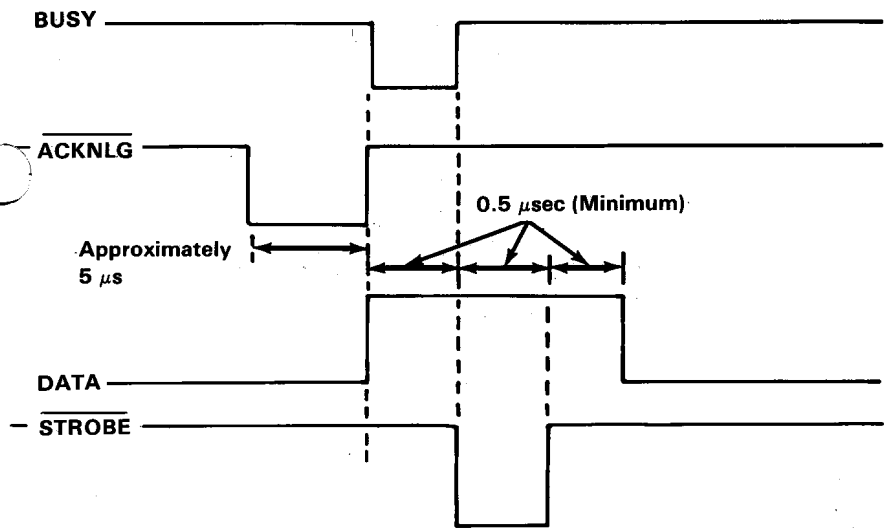



**Parallel Interface Timing Diagram**



## Interface Signals

| | |
|---|---|
| **-Strobe** | STROBE pulse to read data in. Pulse width must be more than 0.5 μs at the receiving terminal. The signal is normally 'high'; however read-in of data is performed at the 'Low' level of this signal. |
| **Data 1-8** | These signals are the first to eight bits of parallel data. Each signal is at a 'high' level when data is a logical 1 and 'low' when data is a logical 0. |
| **-ACKNLG** | Approximately 0.5 μs pulse (low) indicates that data has been received and the printer is ready to accept data. |
| **BUSY** | A 'high' signal indicates that the printer cannot receive data. The signal is 'high' in the following cases:<br>• During data entry |

- During printing operation
- In the "off-line" state
- During printer-error status

| | |
|---|---|
| **PE** | A 'high' signal indicates that the printer is out of paper. |
| **SLCT** | This signal indicates that the printer is in the selected state. |
| **Auto Feed XT** | When this signal is 'low' paper is fed one line after printing. This signal level can be fixed 'low' by DIP switch pin 2-3. |
| **INT** | When this signal is 'low' the printer controller is reset to its initial state and the print buffer is cleared. This signal is normally 'high' and its pulse width must be more than 50 $\mu$s at the receiving terminal. |
| **Error** | This signal is 'low' when the printer is in the "Paper End," "Off Line," and "Error" state. |
| **-SLCTIN** | Data entry to the printer is possible only when this signal is 'low'. This signal can be fixed 'low' by DIP switch 1-8. |

**Notes:**

1. All interface conditions are based on TTL level. Both the rise and fall times of each signal must be less than 0.2 $\mu$s.

2. Data transfer must not be carried out by ignoring the -ACKNLG or BUSY signal. Data transfer can only occur after confirming the -ACKNLG signal or when the BUSY signal is 'low'.

The following figure shows the pin assignment and direction of each signal.

| Signal | Signal Pin # | Return Pin # | Direction |
|---|---|---|---|
| -STROBE | 1 | 19 | In |
| DATA 1 | 2 | 20 | In |
| DATA 2 | 3 | 21 | In |
| DATA 3 | 4 | 22 | In |
| DATA 4 | 5 | 23 | In |
| DATA 5 | 6 | 24 | In |
| DATA 6 | 7 | 25 | In |
| DATA 7 | 8 | 26 | In |
| DATA 8 | 9 | 27 | In |
| -ACKNLG | 10 | 28 | Out |
| BUSY | 11 | 29 | Out |
| PE | 12 | 30 | Out |
| SLCT | 13 | — | Out |
| AUTO FEED XT | 14 | — | In |
| NC | 15 | — | — |
| OV | 16 | — | — |
| CHASSIS GND | 17 | — | — |
| NC | 18 | — | — |
| GND | 19-30 | — | — |
| INT | 31 | — | In |
| ERROR | 32 | — | Out |
| GND | 33 | — | — |
| NC | 34 | — | — |
| | 35 | — | — |
| -SLCT IN | 36 | — | In |

**Pin Assignments**



# Printer Modes

The IBM Graphics Printer can use any of the combinations listed in the following table and the print mode can be changed at any place within the line.

Modes can be selected and combined if they are in the same vertical column.

| Printer Modes | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Normal | X | X | X | | | | | | |
| Compressed | | | | X | X | X | | | |
| Emphasized | | | | | | | X | X | X |
| Double Strike | X | | | X | | | X | | |
| Subscript | | X | | | X | | | X | |
| Superscript | | | X | | | X | | | X |
| Double Width | X | X | X | X | X | X | X | X | X |
| Underline | X | X | X | X | X | X | X | X | X |

**Printer Modes**

# Printer Control Codes

On the following pages are complete codes for printer characters, controls, and graphics. You may want to keep them handy for future reference. The printer codes are listed in ASCII-decimal numeric-order (from NUL which is 0 to DEL, which is 127). The examples given in the Printer-Function descriptions are written in the BASIC language. The "input" description is given when more information is needed for programming considerations.

ASCII decimal values for the printer control codes can be found under "Printer Character Sets."

The Descriptions that follow assume that the printer DIP switches have not been changed from their factory settings.

| Printer code | Printer Function |
|---|---|
| NUL | Null:<br>Used with ESC B and ESC D as a list terminator. NUL is also used with other printer. |

control codes to select options (for example, ESC S).
Example:
LPRINT CHR$ (0);

**BEL** Bell:
Sounds the printer buzzer for 1 second.
Example:
LPRINT CHR$(7);

**HT** Horizontal Tab:
Tabs to the next horizontal tab stop. Tab stops are set with ESC D. Tab stops are set every 8 columns when the printer is powered on.
Example:
LPRINT CHR$(9);

**LF** Line Feed:
Spaces the paper up one line. Line spacing is 1/16-inch unless reset by ESC A, ESC 0, ESC 1, ESC 2, or ESC 3.
Example:
LPRINT CHR$(10);

Form Feed:
Advances the paper to the top of the next page.

> **Note:** The location of the paper, when the printer is powered on, determines the top of the page. The next top of page is 11 inches from that position. ESC C can be used to change the page length.

Example:
LPRINT CHR$(12);

**CR** Carriage Return:
Ends the line that the printer is on and prints the data remaining in the printer buffer. (No Line Feed operation takes place.)

**Note:** IBM Personal Computer BASIC adds a Line Feed unless 128 is added [for example CHR$(141)].

Example:
LPRINT CHR$(13);

**SO** Shift Out (Double Width):
Changes the printer to the Double-Width print-mode.

**Note:** A Carriage Return, Line Feed or DC4 cancels Double-Width print-mode.

Example:
LPRINT CHR$(14);

**SI** Shift In (Compressed):
Changes the printer to the Compressed-Character print-mode. Example:
LPRINT CHR$(15);

**DC2** Device Control 1 (Compressed Off):
Stops printing in the Compressed print-mode.
Example:
LPRINT CHR$(18);

**DC4** Device Control 4 (Double Width Off):
Stops printing in the Double-Width print-mode.
Example:
LPRINT CHR$(20);

**CAN** Cancel:
Clears the printer buffer. Control codes, except SO, remain in effect.
Example:
LPRINT CHR$(24);

**ESC** Escape:
Lets the printer know that the next data sent is a printer command.
Example:
LPRINT CHR$(27);

**ESC -** Escape Minus (Underline)

Format: ESC -;n;
ESC - followed by a 1, prints all of the following data with an underline.
ESC - followed by a 0 (zero), cancels the Underline print-mode.
Example:
LPRINT CHR$(27);CHR$(45);CHR$(1);

**ESC 0** Escape Zero (1/8-Inch Line Feeding)
Changes paper feeding to 1/8-inch.
Example:
LPRINT CHR$(27);CHR$(48);

**ESC 1** Escape One (7/72-Inch Line Feeding)
Changes paper feeding to 7/72-inch.
Example:
LPRINT CHR$(27);CHR$(49);

**ESC 2** Escape Two (Starts Variable Line-Feeding)
ESC 2 is an execution command for ESC A. If no ESC A command has been given, line feeding returns to 1/6-inch.
Example:
LPRINT CHR$(27);CHR$(50);

**ESC 3** Escape Three (Variable Line-Feeding)
Format: ESC 3;n;
Changes the paper feeding to n/216-inch. The example that follows sets the paper feeding to 54/216 (1/4)-inch. The value of n must be between 1 and 255.
Example:
LPRINT CHR$(27);CHR$(51);CHR$(54);

**ESC 6** Escape Six (Select Character Set 2)
Selects Character Set 2. (See "Printer Character set 2")
Example:
LPRINT CHR$(27);CHR$(54);

**ESC 7** Escape Seven (Select Character Set 1)

Selects character set 1. (See "Printer Character Set 1")
Character set 1 is selected when the printer is powered on or reset.
Example:
LPRINT CHR$(27);CHR$(55);

**ESC 8** Escape Eight (Ignore Paper End)
Allows the printer to print to the end of the paper. The printer ignores the Paper End switch.
Example:
LPRINT CHR$(27);CHR$(56);

**ESC 9** Escape Nine (Cancel Ignore Paper End)
Cancels the Ignore Paper End command. ESC 9 is selected when the printer is powered on or reset.
Example:
LPRINT CHR$(27);CHR$(57);

**ESC <** Escape Less Than (Home Head)
The printer head returns to the left margin to print the line following ESC <. This occurs for one line only.
Example:
LPRINT CHR$(27);CHR$(60);

**ESC A** Escape A (Sets Variable Line Feeding)
Format: ESC A;n;
Escape A sets the line-feed to n/72-inch. The example that follows tells the printer to set line feeding to 24/72-inch. ESC 2 must be sent to the printer before the line feeding changes. For example, ESC A;24 (text) ESC 2 (text). The text following ESC A;24 spaces at the previously set line-feed increments. The text following ESC 2 prints with new line-feed increments of 24/72-inch. Any increment between 1/72 and 85/72-inch may be used.

Example:
LPRINT
CHR$(27);CHR$(65);CHR$(24);
CHR$(27);CHR$(50);

**ESC C** Escape C (Set Lines-per-Page)
Format: ESC C;n;
Sets the page length. The ESC C command must have a value following it to specify the length of page desired. (Maximum form length for the printer is 127 lines.) The example below sets the page length to 55 lines. The printer defaults to 66 lines-per-page when powered on or reset.
Example:
LPRINT CHR$(27);CHR$(67);CHR$(55);

Escape C (Set Inches-per-Page)
Format: ESC C;n;m;
Escape C sets the length of the page in inches. This command requires a value of 0 (zero) for n, and a value between 1 and 22 for m.
Example:
LPRINT CHR$(27);CHR$(67);CHR$(0);CHR$(12);

**ESC D** Escape D (Sets Horizontal Tab Stops)
Format: ESC D;n1;n2;...nk;NUL;
Sets the horizontal-tab stop-positions. The example that follows shows the horizontal-tab stop-positions set at printer column positions of 10, 20, and 40. They are followed by CHR$(0), the NUL code. They must also be in ascending numeric order as shown. Tab stops can be set between 1 and 80. When in the Compressed-print mode, tab stops can be set up to 132.
The Graphics Printer can have a maximum of 28 tab stops. The HT (CHR$(9)) is used to execute a tab operation.
Example:

```
LPRINT
CHR$(27);CHR$(68);CHR$(10)
;CHR$(20);CHR$(40);
CHR$(0);
```

**ESC E** Escape E (Emphasized)
Changes the printer to the Emphasized-print mode. The speed of the printer is reduced to half speed during the Emphasized-print mode.
Example:

```
LPRINT CHR$(27);CHR$(69);
```

**ESC F** Escape F (Emphasized Off)
Stops printing in the Emphasized-print mode.
Example:

```
LPRINT CHR$(27);CHR$(70);
```

**ESC G** Escape G (Double Strike)
Changes the printer to the Double-Strike print-mode. The paper is spaced 1/216 of an inch before the second pass of the print head.
Example:

```
LPRINT CHR$(27);CHR$(71);
```

**ESC H** Escape H (Double Strike Off)
Stops printing in the Double-Strike mode.
Example:

```
LPRINT CHR$(27);CHR$(72);
```

**ESC J** Escape J (Sets Variable Line Feeding)
Format: ESC J;n;
When ESC J is sent to the printer, the paper feeds in increments of n/216 of an inch. The value of n must be between 1 and 255. The example that follows gives a line feed of 50/216-inch. ESC J is canceled after the line feed takes place.
Example:

```
LPRINT CHR$(27);CHR$(74);CHR$(50);
```

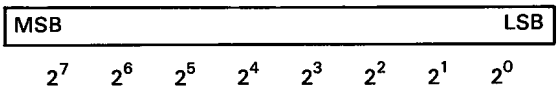
**ESC K** Escape K (480 Bit-Image Graphics Mode)
Format ESC K;n1;n2;v1;v2;...vk;
Changes from the Text mode to the Bit-Image

Graphics mode. n1 and n2 are one byte, which specify the number of bit-image data bytes to be transferred. v1 through vk are the bytes of the bit-image data. The number of bit-image data bytes (k) is equal to n1 +256n2 and cannot exceed 480 bytes. At every horizontal position, each byte can print up to 8 vertical dots. Bit-image data may be mixed with text data on the same line.

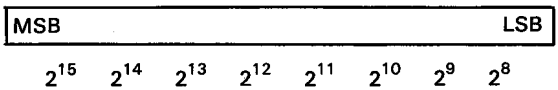
**Note:** Assign values to n1 and n2 as follows:
n1 represents values from 0 - 255.
n2 represents values from 0 - 1 x 256.

MSB is most-significant bit and LSB is least -significant bit.

The following figures show the format.

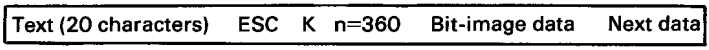



$2^7$ $2^6$ $2^5$ $2^4$ $2^3$ $2^2$ $2^1$ $2^0$



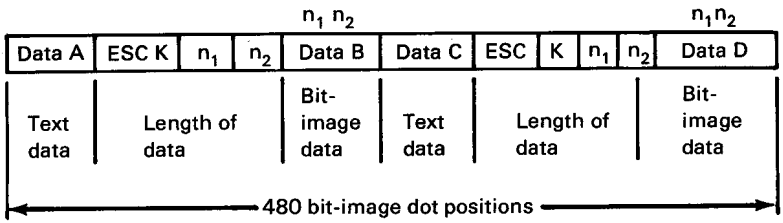

$2^{15}$ $2^{14}$ $2^{13}$ $2^{12}$ $2^{11}$ $2^{10}$ $2^9$ $2^8$

Data sent to the printer.



In text mode, 20 characters in text mode correspond to 120 bit-image positions (20 x 6 = 120). The printable portion left in Bit-Image mode is 360 dot positions (480 - 120 = 360).

Data sent to the printer.




```
Example: 1 'OPEN PRINTER IN RANDOM MODE
WITH LENGTH OF 255
2 OPEN "LPT1:"AS #1
3 WIDTH "LPT1:",255
4 PRINT #1,CHR$(13)+CHR$(10);
5 SLASH$=CHR$(1)+CHR$(02)
+CHR$(04)+CHR$(08)
6 SLASH$=SLASH$+CHR$(16)+CHR$(32)
+CHR&(64)+$CHR$(128)+CHR$(0)
7 GAP$=CHR$(0)+CHR$(0)+CHR$(0)
8 NDOTS=480
9 'ESC K N1 N2
10 PRINT #1,CHR$(27);"K";CHR$(NDOTS
MOD 256);CHR$ (FIX(NDOTS/256));
11 'SEND NDOTS NUMBER OF BIT
IMAGE BYTES
12 FOR I=1 TO NDOTS/12 'NUMBER
OF SLASHES TO
PRINT USING GRAPHICS
13 PRINT #1,SLASH$;GAP$;
```

```
14 NEXT I
15 CLOSE
16 END
```

This example gives you a row of slashes printed in the Bit-Image mode.

**ESC L** Escape L (960-Bit-Image Graphics-Mode)
Format: ESC L;n1;n2;v1;v2;...vk;
Changes from the Text mode to the Bit-Image Graphics mode. The input is similar to ESC K. The 960 Bit-Image mode prints at half the speed of the 480 Bit-Image Graphics mode, but can produce a denser graphic image. The number of bytes of bit-image Data (k) is n1 +256n2 but cannot exceed 960. n1 is in the range of 0 to 255.

**ESC N** Escape N (Set Skip Perforation)
Format ESC N;n;
Sets the Skip Perforation function. The number following ESC N sets the value for the number of lines of Skip Perforation. The example shows a 12-line skip perforation. This prints 54 lines and feeds the paper 12 lines. The value of n must be between 1 and 127. ESC N must be reset anytime the page length (ESC C) is changed.
Example:

```
LPRINT CHR$(27);CHR$(78);CHR$(12);
```

**ESC O** Escape O (Cancel Skip Perforation)
Cancels the Skip Perforation function.
Example:

```
LPRINT CHR$(27);CHR$(79);
```

**ESC S** Escape S (Subscript/Superscript)
Format: ESC S;n;
Changes the printer to the Subscript print mode when ESC S is followed by a 1, as in the example that follows. When ESC S is followed by a 0 (zero), the printer prints in the

Superscript print mode.
Example:
LPRINT CHR$(27);CHR$(83);CHR$(1);

**ESC T** Escape T (Subscript/Superscript Off)
The printer stops printing in the Subscript or Superscript print mode.
Example:
LPRINT CHR$(27);CHR$(84);

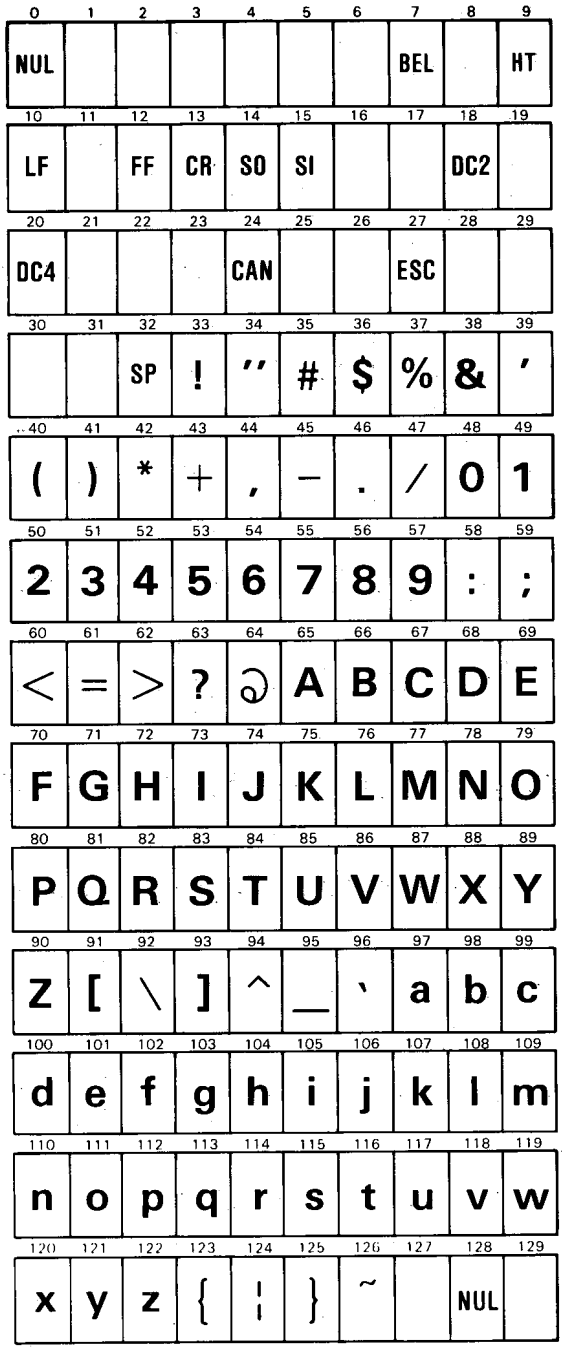
**ESC U** Escape U (Unidirectional Printing)
Format: ESC U;n;
The printer prints from left to right following the input of ESC U;1. When ESC U is followed by a 0 (zero), the left to right printing operation is canceled. The Unidirectional print-mode (ESC U) ensures a more accurate print-start position for better print quality.
Example:
LPRINT CHR$(27);CHR$(85);CHR$(1);

**ESC W** Escape W (Double Width)
Format: ESC W;n;
Changes the printer to the Double-Width print mode when ESC W is followed by a 1. This mode is not canceled by a line-feed operation and must be canceled with ESC W followed by a 0 (zero).
Example:
LPRINT CHR$(27);CHR$(87);CHR$(1);

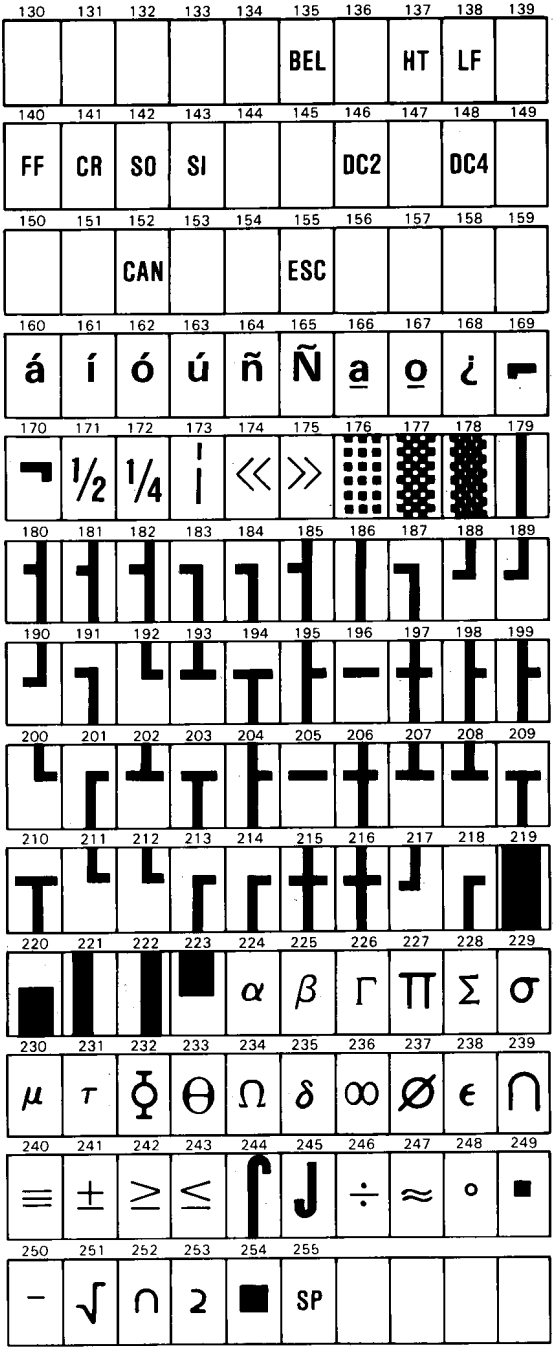
**ESC Y** Escape Y (960 Bit-Image Graphics Mode Normal Speed)
Format: ESC Y n1;n2;v1;v2;...vk;
Changes from the Text mode to the 960 Bit-Image Graphics mode. The printer prints at normal speed during this operation and cannot print dots on consecutive dot position. The input of data is similar to ESC L.

**ESC Z** Escape Z (1920 Bit-Image Graphics Mode)

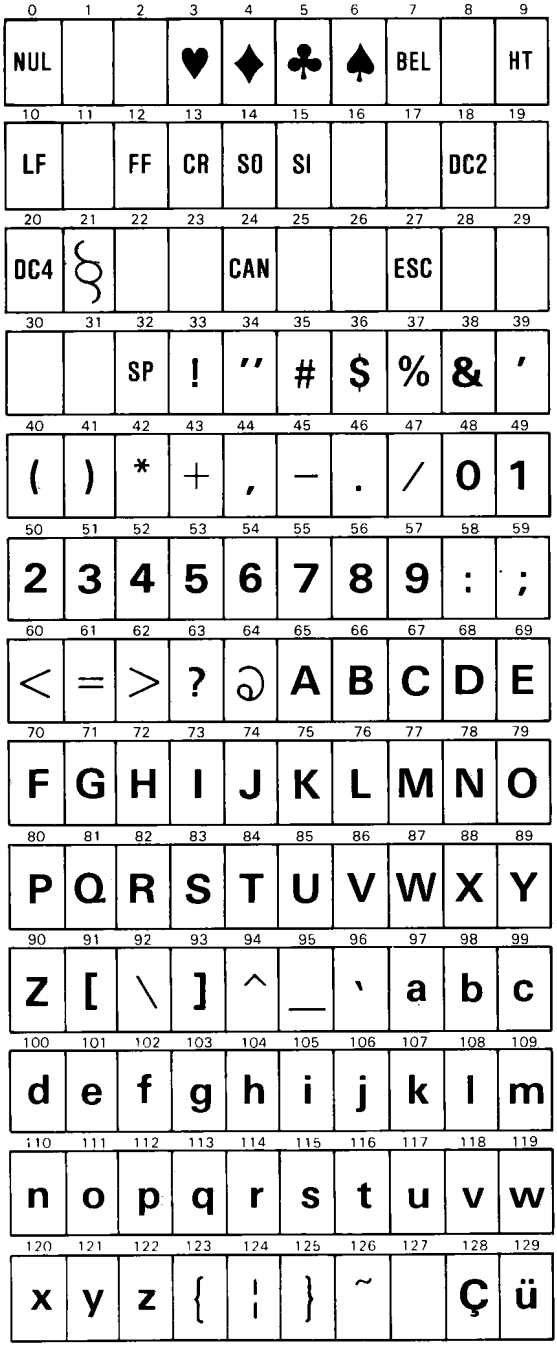
Format: ESC Z;n1;n2;v1;v2;...vk;
Changes from the Text mode to the 1920 Bit-Image Graphics mode. The input is similar to the other Bit-Image Graphics modes. ESC Z can print only every third dot position.



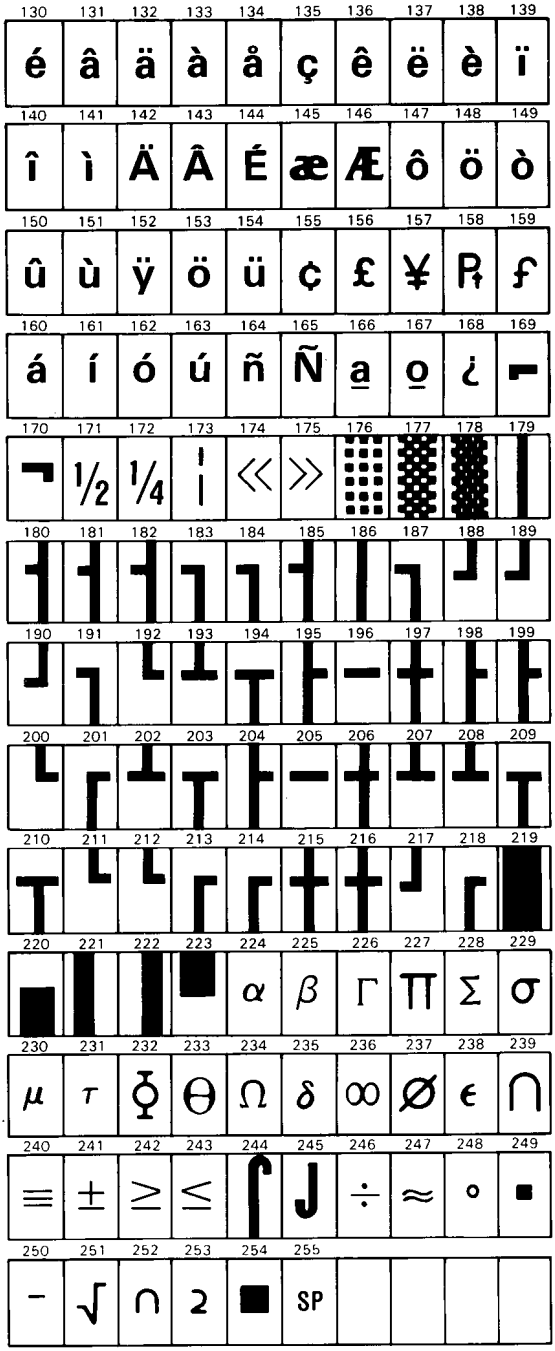

**Printer Character Set 1 (Part 1 of 2)**




**Printer Character Set 1 (Part 2 of 2)**




**Printer Character Set 2 (Part 1 of 2)**



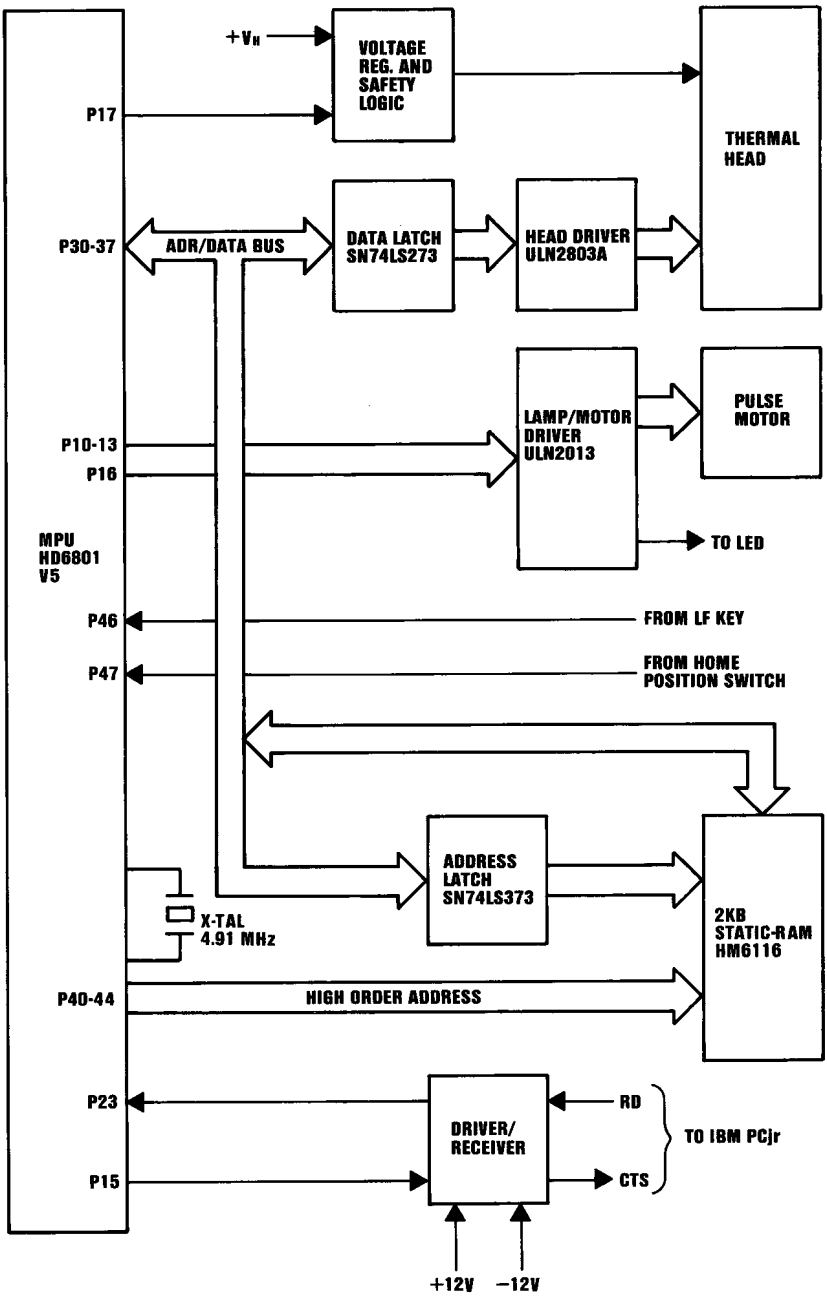

**Printer Character Set 2 (Part 2 of 2)**

# Notes:

# IBM PC Compact Printer

The PC Compact Printer is a stand-alone, tabletop unit that plugs into a standard wall outlet. Using an eight-wire print head, the printer can print characters from the standard ASCII, 96-character, uppercase and lowercase character sets, and prints the characters in a 5-by-7 dot matrix at 56 characters-per-second (cps). It prints in one direction (left-to-right) and has four print modes. In the standard mode, the printer prints 80 characters-per-line; in the compressed mode, 136 characters; in the double-width mode, 40 characters, and in the compressed double-width mode, 68 characters-per-line. The PC Compact Printer can also underline characters, has an extended character-set for international languages, and can accept special characters programmed by the user.

The printer has a 1.89 meter (6-foot), 16-lead, printer cable that connects, through an Amphenol connector, to the serial port (RS-232-C) at the rear of the system unit.


+VH
VOLTAGE REG. AND SAFETY LOGIC
P17
THERMAL HEAD
P30-37
ADR/DATA BUS
DATA LATCH SN74LS273
HEAD DRIVER ULN2803A
LAMP/MOTOR DRIVER ULN2013
PULSE MOTOR
P10-13
P16
TO LED
MPU HD6801 V5
P46
FROM LF KEY
P47
FROM HOME POSITION SWITCH
ADDRESS LATCH SN74LS373
2KB STATIC-RAM HM6116
X-TAL 4.91 MHz
P40-44
HIGH ORDER ADDRESS
P23
DRIVER/ RECEIVER
RD
TO IBM PCjr
P15
CTS
+12V
−12V

# Printer Specifications

| | |
|---|---|
| Print Method: | Thermal, non-impact, Dot-matrix |
| Print Speed: | 56 cps |
| Print Direction: | Left to right only |
| Number of Pins in Print Head: | 8 |
| Line Spacing: | 4.23 mm (1/6 in) |
| Matrix Pattern: | 5 by 7 Dots |
| Character Set: | Full 96-character ASCII with descenders, plus international characters/symbols |
| Graphics: | None |

| Print Modes: | Characters per Inch | Maximum Characters per Line |
|---|---|---|
| Standard | 10 | 80 |
| Double Width | 5 | 40 |
| Compressed | 17.5 | 136 |
| Compressed/ Double Width | 8.75 | 68 |

| | |
|---|---|
| **Paper Feed:** | **Friction Feed** |
| **Paper Width:** | 216 mm (8.5 in) |
| **Copies:** | **Single sheet only** |
| **Paper Path:** | **Top** |
| **System Interface:** | **Serial Data and Control Lines** |
| **Print Color:** | **Black only** |

| | |
|---|---|
| **Environmental Conditions** | |
| Temperature: | 5°C (+41°F) to 40°C (104°F) |
| Humidity: | 10 to 80% non-condensing |
| **Power Requirement** | |
| Voltage: | 110 Vac 60 Hz |
| Current: | 245 mA |
| Power Consumption: | 36 watts |
| Heat Output: | 57.6 kJ (54.6 BTU)/hr (maximum) |
| **Physical Characteristics** | |
| Height: | 88.9 mm (3.5 in) |
| Width: | 312.4 mm (12.3 in) |
| Depth: | 221 mm (8.7 in) |
| Weight: | 2.99 kg (6.6 lb) |
| Power Cable Length: | 1.98 m (6.5 ft) |
| Size: | 28 AWG |
| Printer Cable Length: | 1.83 m (6 ft) |
| Size: | 3 by 18 AWG |

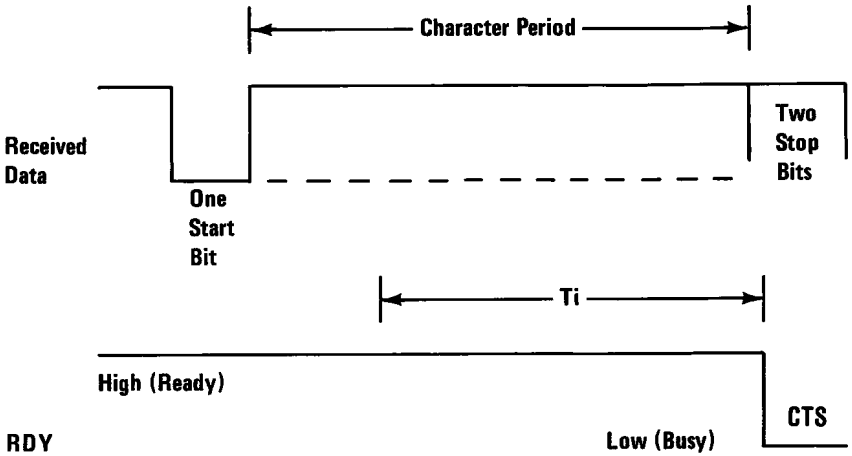
| | |
|---|---|
| Character Set: | ASCII numbers 0 to 31 contain control codes and special characters. ASCII numbers 32 to 127 contain the standard printable characters. ASCII numbers 128 to 175 contain European characters. ASCII numbers 224 to 255 contain math and extra symbols. |

# Serial Interface Description

Specifications:

| | |
|---|---|
| Data Transfer Rate: | 1200 bps (maximum) |
| Synchronization: | internal clocking |
| Handshaking: | CTS (Clear to Send) Pacing |
| Logic Level: | Input data and all interface control- signals are EIA Levels |
| Connector Plug: | 9804 (Amphenol) |

The following figure shows the timing of the Serial Interface.




**Serial Interface Timing Diagram**

# Print Mode Combinations for the PC Compact Printer

The following figure shows the print-mode combinations possible with the PC Compact Printer. Modes shown in the same column can be combined. A print mode can be changed at any time within a line: however, the double-width mode effects the entire line.

| Modes | | | | | |
|---|---|---|---|---|---|
| Standard | XXX | | | | |
| Compressed | | XXX | | XXX | XXX |
| Double-Width | | | XXX | XXX | XXX |
| Underline | XXX | XXX | XXX | | XXX |

# Printer Control Codes and Functions

On the following pages you will find a detailed list of the printer control codes and functions. This list also includes descriptions of the functions and examples of the printer control codes.

The examples (LPRINT statements) given in the detailed descriptions of the printer control codes and functions list, are written in BASIC. Some knowledge of BASIC programming is needed to understand these codes. Some of the printer control codes also show a "Format" description when more information is needed for programming considerations.

| CODE | PRINTER FUNCTION |
|---|---|
| CAN | **Cancel**<br>Clears the printer buffer. Control codes, except SO, remain in effect. Reinitializes the printer to the power on defaults.<br>**LPRINT CHR$(24);** |
| CR | **Carriage Return**<br>Ends the line the printer is on and prints any data remaining in the printer buffer. The logical character position is moved to the left margin. (No Line Feed operation takes place.) Note: IBM Personal Computer BASIC adds a Line Feed unless 128 is added.<br>**LPRINT CHR$(13);** |
| DC2 | **Device Control 2 (Compressed Off)**<br>Stops printing in the Compressed mode.<br>**LPRINT CHR$(18);** |
| DC4 | **Device Control 4 (Double Width Off)**<br>Stops printing in the Double Width mode.<br>**LPRINT CHR$(20);** |
| ESC | **Escape**<br>Informs the printer that the following data is a printer command. (See the following ESC commands.)<br>**LPRINT CHR$(27);** |

**ESC B** **Escape B (Set Vertical Tabs)**
Sets vertical tab stop positions. Up to 64 vertical tab stop positions are recognized by the printer. Tab stop positions must be received in ascending numeric order. The tab stop numbers do not become valid until you type the NUL code. Once vertical tab stops are established, they are valid until new tab stops are specified. (If the printer is reset or switched Off, set tab stops are cleared.) If no tab stop is set, the Vertical Tab command acts as a Line Feed command. ESC B followed only by NUL cancels tab stops. The form length must be set by the ESC C command prior to setting tabs.

```
LPRINT
CHR$(27);CHR$(66);CHR$(10);CHR$(20);
CHR$(40);CHR$(0);
```

**ESC C** **Escape C (Set lines per page)**
Format: ESC C;n; Sets the page length. The ESC C command must be followed by a value to specify the length of page desired. (Maximum form length for the printer is 127 lines.) The following example sets the page length to 55 lines. The printer default is 66 lines per page when switched On or reset.

```
LPRINT CHR$(27);CHR$(67);CHR$(55);
```

**ESC D** **Escape D (Set Horizontal Tab Stops)**
Sets the horizontal tab stop positions. The following example shows the horizontal tab stop positions set at printer column positions of 10, 20 and 40. The horizontal tab stops are followed by CHR$(0), the NUL code. They must also be in ascending numeric order as shown. You can set tab stops between 1 and 80. When in the Compressed print mode, you can set tabs up to column 136. The maximum number of tabs that can be set is 112. HT (CHR$(9)) is used to execute a tab operation.

```
LPRINT
CHR$(27);CHR$(68);CHR$(10)CHR$(20)
CHR$(40);CHR$(0);
```

**ESC K** **Escape K (480 Bit-Image Graphics Mode)**
Format: ESC K;n1;n2; v1; v2;.....vk;
Changes the printer to the Bit-Image Graphics mode. Dot density is 82.5 by 82.5 dots per inch. If the graphics data exceeds the space remaining on the line, the printer ignores the excess data. Only the excess data is lost.

The numbers n1 and n2 specify, in binary form, the number of bit image data bytes to be transferred. Assign values to n1 to represent values from zero to 255 and assign values to n2 to represent values from 0-1 x 256. The total number of bit image data bytes cannot exceed 480. (n1 + (n2 X 256)).

The bit-image data bytes are v1 through vk.

All eight of the print head wires are used to print Bit-image graphics. Each bit of a bit-image data byte represents a dot position within a vertical line. The least significant bit (LSB) represents the bottom dot position, and the most significant bit (MSB) represents the top dot position. For example, if vX is hex 80, the top dot will print only in that vertical position; if vX is hex 01, the bottom dot will print; and if vX is hex FF, all eight dots will print.

| Dot | Bit Number |
|---|---|
| Top | O - - - 8 |
| | O - - - 7 |
| | O - - - 6 |
| | O - - - 5 |
| | O - - - 4 |
| | O - - - 3 |
| | O - - - 2 |
| Bottom | O - - - 1 |

**LPRINT CHR$(27);CHR$(75);n1;n2**

**ESC N** **Escape N (Set Skip Perforation)**
Format: ESC N;n; Sets the Skip Perforation function. The number following ESC N sets the number of lines to be skipped. The example shows a 12-line skip perforation. This command will print 54 lines and feed the paper 12 lines. The value of n must be between 1 and 127. ESC N must be reset anytime the page length (ESC C) is changed. The default for skip perforation is 25.4 mm (1 inch).
**LPRINT CHR$(27);CHR$(78);CHR$(12);**

**ESC O** **Escape O (Cancel Skip Perforation)**
Cancels the Skip Perforation function.
**LPRINT CHR$(27);CHR$(79);**

**ESC R** **Escape R (Clear Tabs)**
Resets all tab stops,both horizontal and vertical to the powered-on defaults.
**LPRINT CHR$(27);CHR$(82);**

**ESC W** **Escape W (Double Width)**
Format: ESC W;n; Changes the printer to the Double Width mode when ESC W is followed by 1. This mode is not canceled by a line feed operation. It is canceled when ESC W is followed by 0 (zero).
**LPRINT CHR$(27);CHR$(87);CHR$(1);**

**ESC 0** **Escape Zero (1/9-Inch Line Feed)**
Changes the line feed to 2.82 mm (1/9 inch).
**LPRINT CHR$(27);CHR$(48);**

**ESC 1** **Escape One (1/9-inch Line Feed)**
Changes the line feed to 2.82 mm (1/9 inch). ESC 1 functions the same as ESC 0.
**LPRINT CHR$(27);CHR$(49);**

**ESC 2** **Escape Two (Start Variable Line Feeding)**
Resets line spacing to 4.23 mm (1/6 inch). This is the powered-on default for vertical line spacing.
**LPRINT CHR$(27);CHR$(50);**

**ESC 5** **Escape Five (Sets Automatic Line Feed)**
With automatic line feed on, when a CR code is received, a line feed automatically follows after the carriage return. ESC 5 (1) sets auto line feed; ESC 5 (0) resets it.
**LPRINT CHR$(27);CHR$(53);**

**ESC -** **Escape Minus (Underline)**
Format: ESC -;n; ESC - followed by 1, prints all of the following data with an underline. ESC - followed by 0 (zero), cancels the Underline print mode.
**LPRINT CHR$(27);CHR(45);CHR$(1); [or CHR$(0);]**

**ESC <** **Escape Less Than (Home Head)**
The print head returns to the left margin to print the line following ESC <. This occurs for one line only.
**LPRINT CHR$(27);CHR$(60);**

**FF** **Form Feed**
Advances the paper to the top of the next page. Note: The location of the paper, when the printer power switch is set to the On position, determines the top of the page. The next top-of-page is 279 mm (11 inches) from that position. ESC C can be used to change the page length. Always separate multiple Form Feed commands with spaces.
**LPRINT CHR$(12);**

**HT** **Horizontal Tab**
Tabs to the next horizontal tab stop. Tab stops are set with ESC D. (Tab stops are automatically set at every 8 columns when the printer power switch is set to the On position.)
**LPRINT CHR$(9);**

**LF** **Line Feed**
Advances the paper one line. Line spacing is 4.23 mm (1/6 inch) unless reset by ESC 0, ESC 1, ESC 2.
**LPRINT CHR$(10);**

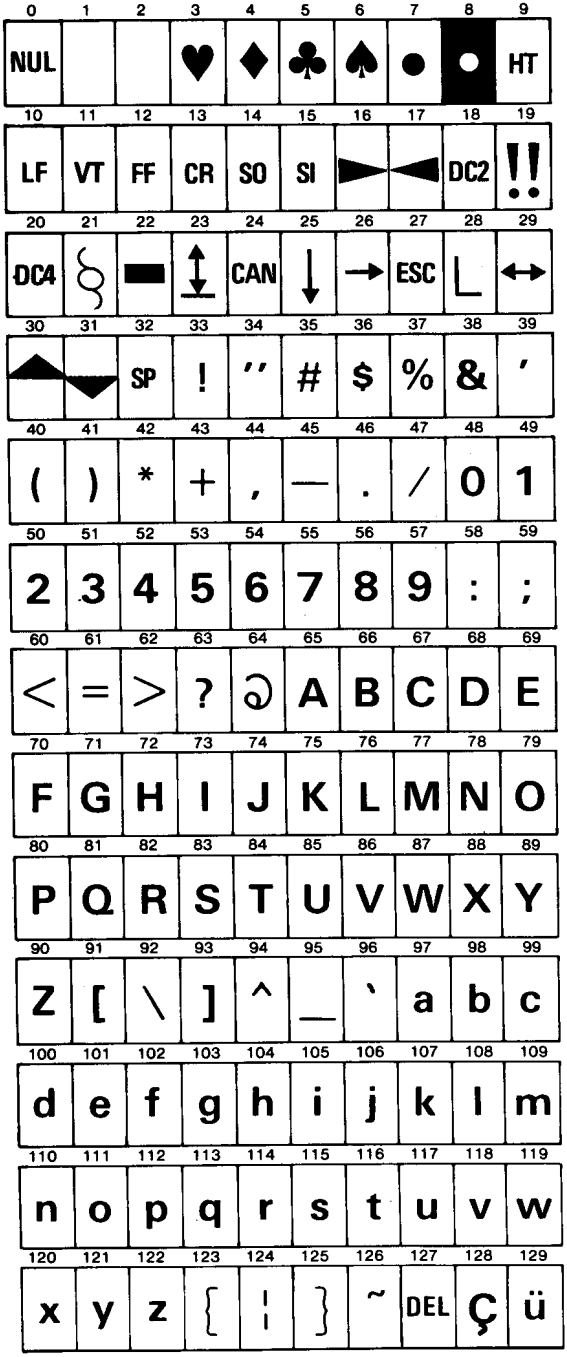
**NUL** **Null**
Used with ESC B and ESC D as terminator for the tab set and clear commands.
**LPRINT CHR$(0);**

**SI** **Shift In (Compressed On)**
Changes the printer to the Compressed Character mode. This command is canceled by a DC2 code (Compressed Off).
**LPRINT CHR$(15);**

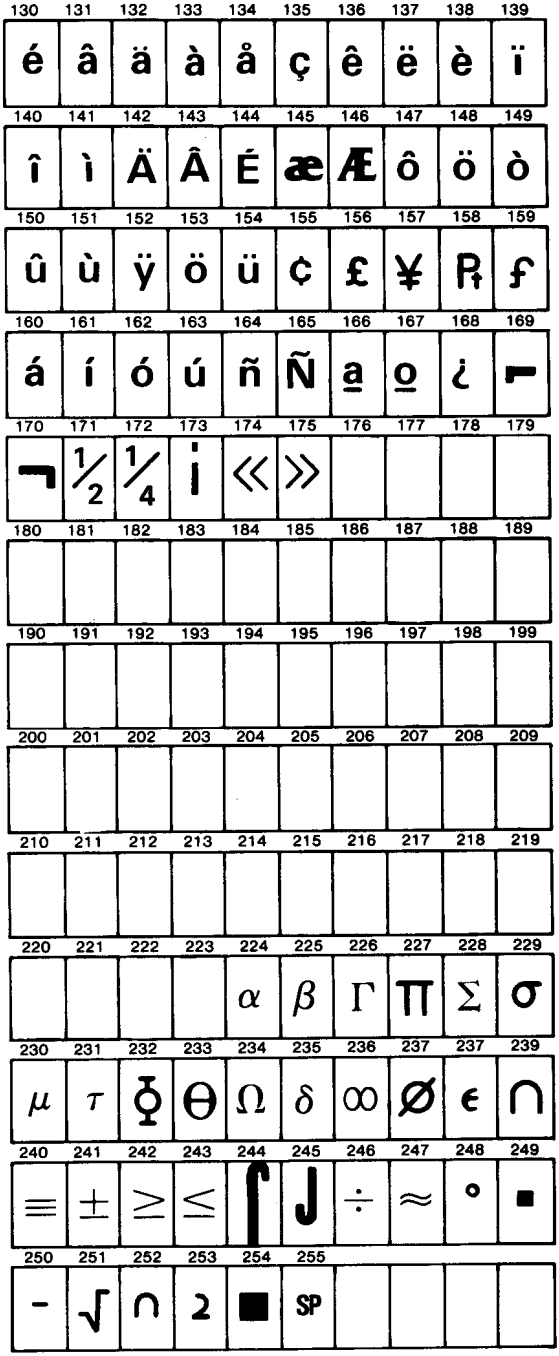
**SO** **Shift Out (Double Width)**
Changes the printer to the Double Width mode. Note: A Carriage Return, Line Feed or DC4 code cancels Double Width mode.
**LPRINT CHR$(14);**

**VT** **Vertical Tab**
Spaces the paper to the next vertical tab position. VT are set by the ESC B sequence. The VT command is the same as the LF command, if no tabs are set. The paper is advanced one line after printing or advanced to the next vertical tab stop.
**LPRINT CHR$(11);**

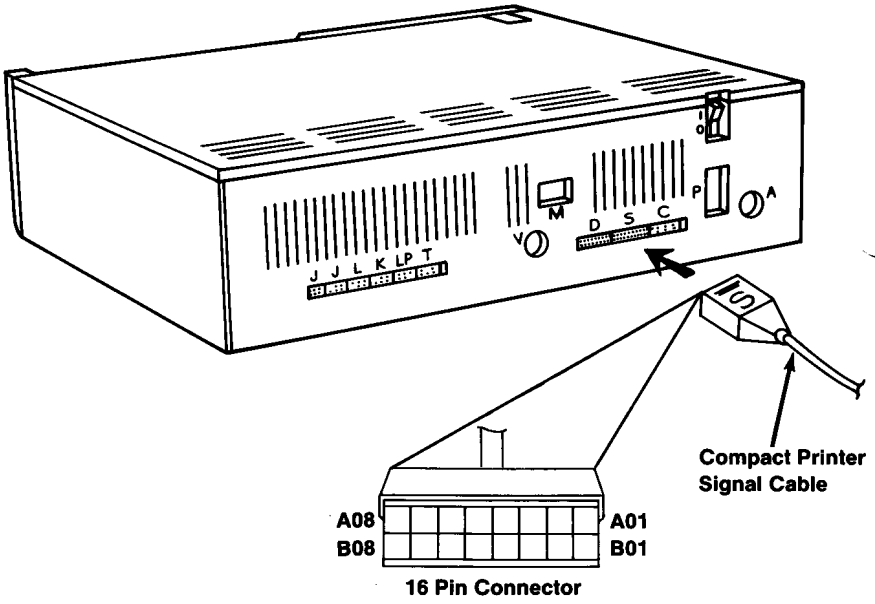
The following charts list the printer control codes and characters in ASCII decimal numeric order, (for example, NUL is 0 and ESC W is 87).

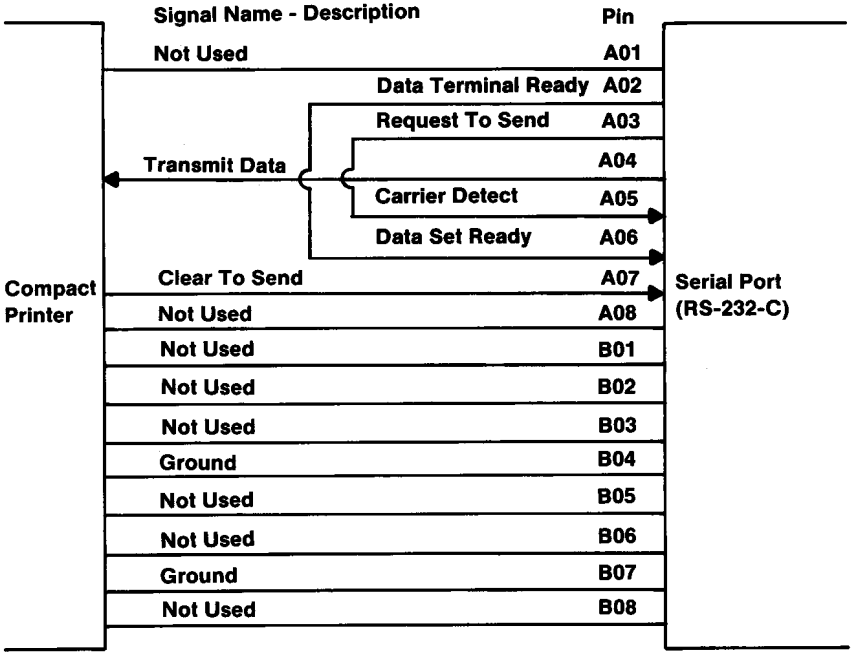



**Character Set (Part 1 of 2)**




**Character Set (Part 2 of 2)**




**Data Terminal Ready Looped in Cable to Data Set Ready**
**Request to Send Looped in Cable to Carrier Detect**

## Connector Specifications

# SECTION 4. COMPATIBILITY WITH THE IBM PERSONAL COMPUTER FAMILY

## Contents

# Notes:

# Compatibility Overview

The IBM PC*jr* is a different Computer than the IBM Personal Computer and IBM Personal Computer XT. Even though it is different, the IBM PC*jr* has a high level of programming compatibility with the IBM Personal Computers. It is possible to create PC*jr* software applications that can run without modification on other IBM Personal Computers. In order to create such programs or to assess if a current program is compatible, you must understand the differences between the Personal Computers in the IBM family and know the proper way to communicate with them.

Normally, it would be impossible for a program written for one computer to run on a different computer since the microprocessors would be different; and the language of the application could not be executed by different processors. In this case, the application would have to be re-written entirely in the language of the other processor. Since the IBM PC*jr* and the other IBM Personal Computers use exactly the same microprocessors (Intel 8088), most assembler language programs need not be modified.

This alone is not enough, since applications normally take advantage of a computers device services (BIOS) and operating system (IBM DOS 2.1). In order to allow for maximum program compatibility, the IBM PC*jr* has maintained all BIOS system interrupts and utilizes the same IBM DOS. This means that applications which use the BIOS and the IBM DOS interrupts on the IBM Personal Computers operate the same on the IBM PC*jr*.

**Note: The BIOS micro-code of the IBM PC*jr* is not identical to that of the IBM Personal Computers. If an application bypasses the BIOS interrupt calls and**

**directly accesses routines and/or storage locations in one system, it may not run in the other system. Some routines may be similar and some BIOS storage locations may be the same. It is strongly recommended that applications use only the BIOS and DOS interrupt interfaces in order to achieve compatibility in the IBM Personal Computer family.**

Using the same language and the BIOS and DOS interfaces go a long way in achieving application compatibility. However, there are still several factors which need to be taken into consideration:

- Timing Dependencies
- Unequal Configurations
- Hardware Differences

# Timing Dependencies

Programs running in user read/write memory normally run slower on the PC*jr* than on the IBM Personal Computers. Programs running in read-only memory (ROM) normally run a little faster on the PC*jr* than on the IBM Personal Computers. This may or may not cause a difference depending upon the application. Most applications are very I/O dependent in which case the execution time is not the critical factor and may not be noticeable. In other cases, the application runs the same but merely take a different amount of time.

If an application has very critical timing dependencies, any timing differences (faster or slower) may adversely affect its usability. Using an application's program execution speed to achieve a desired timing can effect the application. In these cases, the application may need to be modified.

> **Note:** It is strongly recommended not to depend on instruction execution speed to achieve specific application timing. The system timer can provide short interval timing for assembly language programs. Similar timing functions are available in BASIC.

Performance of specific I/O devices (such as diskette or printer) may also differ between the PC*jr* and the other IBM Personal Computers. You should also avoid using timing of any I/O device as a dependency for the application.

# Notes:

# Unequal Configurations

In designing an application to run on both the IBM PC*jr* and the IBM Personal Computers, you need to make sure that the required hardware configuration is available on all machines. This means the application's minimum requirements are met by all IBM Personal Computers.

# Notes:

# Hardware Differences

To be able to run on either computer without change, an application utilizing a specific I/O device must have access to identical devices (or devices with identical operating characteristics and interfaces). The IBM PC*jr* and the IBM Personal Computers have very compatible I/O device capabilities.

The following table lists the hardware features and I/O devices supported by the IBM PC*jr* and the IBM Personal Computers and summarizes the differences:

| Device | PC | PCXT | PC*jr* | PC*jr* Comments |
|---|---|---|---|---|
| Maximum User Memory | 640KB | 640KB | 128KB | Shares user RAM with Video Buffer |
| Cordless Keyboard | No | No | Yes | Scan codes compatible and full 83 key capability |
| 83 Key Keyboard | Yes | Yes | No | Compatible, but Hardware interface differences |
| Diskette Drive | Yes | Yes | Yes | Compatible, but different address and no DMA support |
| Hard Disk File | No | Yes | No | |
| Parallel Printer | Yes | Yes | Yes | Compatible |
| RS 232 Serial Port | Yes | Yes | Yes | Compatible, hex 2F8 address, Interrupt Level 3, Baud-Rate-Frequency divisor difference |
| Game Control | Yes | Yes | Yes | Compatible interface with potential timing differences |
| Cassette | Yes | No | Yes | Compatible |
| Internal Modem | No | No | Yes | Compatible to PC Serial Port hex 3F8 address, Interrupt Level 4, frequency divisor difference |
| IBM Monochrome Display | Yes | Yes | No | |
| Color Graphics and Display | Yes | Yes | Yes | Compatible, with some register differences and enchancements |
| Light Pen | Yes | Yes | Yes | Compatible |

**PC*jr* and Personal Computers Comparison (Part 1 of 2)**

| Device | PC | PCXT | PC*jr* | PC*jr* Comments |
|---|---|---|---|---|
| Attachable Joystick | Yes | Yes | Yes | Compatible |
| 8253 Timer (time of day) | Yes | Yes | Yes | Compatible |
| 8259 Interrupt | Yes | Yes | Yes | Some difference in interrupt levels |
| Internal Sound | Yes | Yes | Yes | Compatible but less frequency response |
| TI 76496 Sound | No | No | Yes | |
| ROM Cartridge Interface | No | No | Yes | |
| Future I/O ROM Architecture | Yes | Yes | Yes | Compatible |

**PC*jr* and Personal Computers Comparison (Part 2 of 2)**



The hardware differences between the IBM PC*jr* and the IBM Personal Computers may lead to incompatibilities depending upon the specific application. Once again; if your application maintains an interface to the Personal Computer Family at the BIOS and DOS interrupt levels, then all hardware differences are handled transparently to your application. If your application goes below the BIOS level and directly addresses the hardware, then there could be an incompatibility.

# User Read/Write Memory

Memory difference can be a problem even with programs written for the same computer, if the available memory is not the same from one machine to the next. Thus, the deciding factor is to state what the minimum memory requirement is for the application, and require that amount on the computer in question.

It is important to understand the memory aspects of the IBM PC*jr* in relationship to that of the IBM Personal Computers. The IBM PC*jr* can be configured for 64K bytes or 128K bytes (with memory expansion). However, this user memory is not all available to the application. The IBM PC*jr* video architecture utilizes a minimum of 16K bytes (in graphic mode) and 2K bytes (in alpha numeric mode) for the screen buffer. Therefore (in graphics mode), the IBM PC*jr* really has 48K bytes or 112K bytes (with memory expansion) available for system software. This is not the case with the IBM Personal Computers, since the color graphics adapter contains a separate 16K byte screen buffer. Thus, a 64K bytes Personal Computer with color graphics (extra 16K bytes) is an 80K byte system compared to a 64K byte IBM PC*jr*. The IBM PC*jr* also has graphic enhancements which allow more than the 16K bytes to be utilized for video screen buffers. If these enhanced features are used in an application, then even less is available for user memory.

Another aspect of available memory is the amount taken away by operating systems and language interpreters. In the case of the IBM DOS, both the IBM PC*jr* and the IBM Personal Computers support the same DOS. If your application requires the BASIC interpreter, then there may be a difference. The IBM Personal Computer Cassette BASIC resides entirely in the system ROM; taking no user memory. However, Disk BASIC or Advanced BASIC utilizes

approximately 10K bytes and 14K bytes respectively from user memory. In the IBM PC*jr*, Advanced BASIC capabilities (cartridge BASIC) reside in ROM, taking no user memory.

As you can see, many items factor into user available memory requirements. The most frequent comparison is for the assembler language or compiled application using a 16K-byte screen buffer operating under DOS 2.1. In this case, an application requiring 64K bytes of user memory on an IBM Personal Computer cannot run on the IBM PC*jr* without its expansion memory (128K byte capability). This is because of the IBM PC*jr* video usage of 16K bytes. Also, any application requiring more than 112K bytes of user memory with DOS 2.1 on the IBM Personal Computers cannot run on an IBM PC*jr*.

## Diskette Capacity/Operation

Since the IBM PC*jr* maximum stand-alone configuration is one diskette drive with a maximum capacity of 360K bytes diskette storage , an IBM PC*jr* application is either limited by this diskette capacity or is impacted by the user having to change diskettes more frequently. The IBM Personal Computers can have multiple diskette drives with a capacity of 360K bytes diskette storage each or even possess hard files with a much larger disk storage capacity. This capacity difference may or may not be a concern depending upon the specific application.

In terms of diskette interfacing, the IBM PC*jr* and the IBM Personal Computers both utilize the NEC $\mu$PD765 floppy diskette controller, but with different hardware addresses, and the IBM PC*jr* does not operate through direct memory access (DMA). Since the IBM PC*jr* does not have DMA capability, application programs

cannot overlap diskette I/O operations. When diskette I/O takes place, the entire system is masked (operator keystrokes and asynchronous communications cannot take place). Therefore, the application must insure that asynchronous operations do not take place while diskette I/O is active.

## IBM PC*jr* Cordless Keyboard

The Cordless Keyboard is unique to the IBM PC*jr*. Even though it does not possess all 83 keys of the IBM Personal Computers' keyboards, it does have the capability to generate all of the scan codes of the 83-key keyboard.

The following shows the additional functions available on the PC*jr*.

| PC*jr* Special Functions | Required Key Combinations |
|---|---|
| Shift screen to the left | Alt + Ctrl + cursor left |
| Shift screen to the right | Alt + Ctrl + cursor right |
| Audio Feedback (System clicks when a key is pressed. | Alt + Ctrl + Caps Lock |
| Customer Diagnostics | Alt + Ctrl + Ins |

**PC*jr* Special Functions**

For more detail see "Keyboard Encoding and Usage" in Section 5.

Since all scan codes can be generated, any special application requirements can be met on the Cordless Keyboard.

The highest level of compatibility to interface to keyboards is through BIOS Interrupt hex 16 (read keystroke). Below that level is risky since there are hardware differences between the PC*jr* keyboard and the IBM Personal Computers' keyboards. The PC*jr* system utilizes the non-maskable (NMI) Interrupt to deserialize the scan codes and pass it to Interrupt hex 48 for compatible mapping to 83-key format. Interrupt level 9 remains a compatible interface for 83-key scan-code handling. It is not recommended to replace Interrupt level 9 even though a high degree of compatibility is maintained. If necessary, analyze this architecture carefully.

## Color Graphics Capability

The IBM PC*jr* color graphic architecture is quite different from that of the IBM Personal Computers. The main difference (as previously discussed) is that the video buffer is taken from main user memory rather than having separate memory for video (as in the IBM Personal Computers). Normally, this would be an incompatibility since applications directly address the color graphics buffer at hex B8000. However, the IBM PC*jr* has special hardware to redirect hex B8000 addressing to any specific 16K-byte block of its user memory. The IBM PC*jr* defaults the video buffer to the high end 16K-byte block of user memory and applications can continue to address the video buffer at hex B8000. In addition all IBM Personal Computers' color graphics adapter modes are BIOS compatible and memory structure (bit map) compatible. These modes are:

| Modes | Requirements |
|---|---|
| Alphanumeric: | |
| 40x25 BW | None |
| 40x25 Color | None |
| 80x25 Color | Note |
| 80x25 BW | None |
| Graphics: | |
| 320x200 4 Color | None |
| 320x200 BW | None |
| 640x200 BW | None |
| **Note:** PC*jr* requires the 64KB Memory and Display Expansion. | |

**Modes Available on the IBM Personal Computers and PC*jr***

In addition the IBM PC*jr* provides some new enhanced graphic modes which are not available to the IBM Personal Computers.

| Modes | Requirements |
|---|---|
| Graphics: | |
| 320x200 16 Color | Note |
| 640x200 4 Color | Note |
| 160x200 16 Color | None |
| **Note:** PC*jr* requires the 64KB Memory and Display Expansion. | |

**Modes Available Only on PC*jr***

The IBM PC*jr* and IBM Personal Computers utilize the 6845 controller, but the hardware interface is not completely the same. Hardware addresses hex 3D8 and

hex 3D9 are not supported by the IBM PC*jr* video interface. Requests using these two addresses are not honored.

Also there are differences in the actual video used by the hardware. BIOS maintains compatibility by using the appropriate PC*jr* video parameters (addressed through Interrupt hex 1D) and maintains all video calls (through Interrupt hex 10). Application can still specify video parameter overrides by modifying Interrupt hex 1D to address their own parameters; however, since there are hardware differences the recommended approach is as follows:

1. Copy the original parameters from the BIOS of the system.
2. Change only those parameters desired.
3. Consider the specific video differences between systems.

Other differences to be aware of are:

- The IBM PC*jr* defaults the colorburst mode to be off, whereas the IBM Personal Computers default colorburst to on. Thus applications should not assume either default but set colorburst mode (through BIOS call) to the desired setting.
- The IBM PC*jr* video supports a full gray scale capability which the IBM Personal Computers do not.
- There can be some color differences between the IBM Personal Computers and the IBM PC*jr*; especially when color mixing techniques are used.

## Black and White Monochrome Display

The IBM PC*jr* does not support the IBM Personal Computers black and white monochrome display. Programs which directly address the IBM Personal Computers monochrome display are not compatible. For example, any direct addressing of the B&W video buffer at hex B8000 is not redirected by the IBM PC*jr*. Applications should support Personal Computer video capabilities through BIOS, and the video buffer address is either transparent to the application or the address is provided indirectly in the BIOS data area.

## RS232 Serial Port and IBM PC*jr* Internal Modem

The IBM PC*jr* serial port address is hex 2F8 and is associated with hardware Interrupt level 3. This is compatible with a second Asynchronous Communications Adapter on the IBM Personal Computers. The Internal Modem address is hex 3F8 and is associated with Interrupt level 4. This is compatible with the first Asynchronous Communications Adapter on the IBM Personal Computers. It is important to note that when the IBM PC*jr* has the Internal Modem installed it is logically COM1 and the RS232 serial port is logically COM2 in BIOS, DOS, and BASIC. Without the Internal Modem installed the RS232 serial port is logically addressed as COM1 in BIOS, DOS, and BASIC even though its address is still hex 2F8 using Interrupt level 3. Other hardware differences on the PC*jr* serial devices are:

- A different frequency divisor is needed to generate baud rate. This is transparent to applications using BIOS to initialize the devices (Interrupt Hex 14).
- No ring indicate capability on the RS232 serial port.

- Asynchronous communications input cannot be overlapped with IBM PC*jr* diskette I/O. Since diskette I/O operates in a non-DMA mode any asynchronous data received during diskette activity may be overrun (and lost). Thus, applications must insure that no diskette activity is active while receiving asynchronous communication data. This can be done by pacing the asynchronous device (tell it to hold from sending ). The ASCII characters XOFF and XON are frequently used by some host computers for this purpose.

# Summary

In summary, the IBM PC*jr* is a member of the IBM Personal Computer family by way of its strong architecture compatibility. The highest degree of application compatibility can be achieved by using a common high level language, and/ or accessing the system only through BIOS and DOS interrupts. It's not recommended to go below the BIOS level even though there are other hardware compatibilities. When it is necessary to design for particular computer differences, the application should determine at execution time which particular computer it is running on. This can be done by inspecting the ROM memory location at segment address hex F000 and offset hex FFFE for the following values

**hex FF** = the IBM Personal Computer
**hex FE** = the IBM Personal Computer XT
**hex FD** = the IBM PC*jr*

Once determined, dual paths would handle any differences.

# Notes:

# SECTION 5. SYSTEM BIOS USAGE

## Contents

# 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 (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, 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.

# Notes:

# BIOS Usage

Access to BIOS is through the software interrupts. Each BIOS entry-point is available through its own interrupt, which can be found in "Personal Computer BIOS Interrupt Vectors", later in this section.

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 hex 12

invokes the BIOS routine for determining memory size and returns the value to the caller.

All parameters passed to and from the BIOS routines go through the 8088 registers. The prologue 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
MOV  DX,LOW_COUNT
INT  1AH               ;set the time.
```

To read 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 prologue 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 | Keyboard NMI | KBDNMI |
| 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 | Vertical retrace | D_EOI |
| 38-3B | E | Diskette Error Handler | 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 | 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 | CRT_CHARH |

**Personal Computer BIOS Interrupt Vectors**

# Vectors with Special Meanings

The following are vectors with special meanings.

## Interrupt Hex 1B - Keyboard Break Address

This vector points to the code to be executed when **Break** is 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 an IRET instruction, so that nothing occurs when **Break** is pressed unless the application program sets a different value.

Control may be retained by this routine, with the following problem. The 'Break' may have occurred during interrupt processing, so that one or more 'End of Interrupt' commands must be issued 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 occurs unless the application modifies the pointer. It is the responsibility of the application to save and restore all registers that are modified.

## Interrupt Hex 1D - Video Parameters

This vector points to a data region containing the parameters required for the initialization of the 6845 CRT Controller. 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. It is recommended that if a programmer wishes to use a different parameter table, that the table contained in ROM be copied to RAM and just modify the values needed for the application.

## 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. It is recommended that if a programmer wishes to use a different parameter table, that the table contained in ROM be copied to RAM and just modify the values needed for the application. The motor start-up-time parameter (parameter 10) is overridden by BIOS to force a 500-ms delay (value 04) if the parameter value is less than 04.

## Interrupt Hex 1F and hex 44 - Graphics Character Pointers

When operating in the graphics modes, the

read/write-character interface forms the character from the ASCII code-point, using a table of dot patterns where each code point is comprised of 8 bytes of graphics information. The table of dot patterns for the first 128 code-points contained in ROM is pointed to by Interrupt Hex 44 and the second table of 128 code-points contained in ROM is pointed to by Interrupt Hex 1F. The user can change this vector to point to his own table of dot patterns. It is the responsibility of the user to restore these vectors to point to the default code-point-tables at the termination of the program.

## Interrupt Hex 48 - Cordless Keyboard Translation

This vector points to the code responsible for translating keyboard scan-codes that are specific to the Cordless Keyboard. The translated scan-codes are then passed to the code pointed to by Interrupt Hex 9 which then handles the 83-key Keyboard scan codes.

## Interrupt Hex 49 - Non-Keyboard Scan-Code Translation-Table Address

This interrupt contains the address of a table used to translate non-keyboard scan-codes (scan codes greater than 85 excluding 255.) If Interrupt hex 48 detects a scan code greater than 85 (excluding 255) it translates it using the table pointed to by Interrupt Hex 49. The address that Interrupt Hex 49 points to can be changed by users to point to their own table if different translations are required.

**Note:** It is recommended that a programmer save default pointers and restore them to their original values when the program has terminated.

# Notes:

# 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 attachments to the system. This includes the optional IBM PC*jr* Internal Modem and the standard RS232 serial-port. Locations hex 408 to 40F contain the base addresses of any parallel printer attachments.

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.

The following is a list of the interrupts reserved for BIOS, DOS, and BASIC.

| 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 |
| 100-115 | 40-43 | Reserved for BIOS |
| 116-119 | 44 | First 128 Graphics Characters |
| 120-131 | 45-47 | Reserves for BIOS |
| 132-135 | 48 | Cordless-Keyboard Translation |
| 136-139 | 49 | Non-keyboard Scan-code Translation Table |
| 140-17F | 50-5F | Reserved for BIOS |
| 100-17F | 40-5F | Reserved for BIOS |
| 180-19F | 60-67 | Reserved for User Software Interrupts |
| 1A0-1FF | 68-7F | Reserved |
| 200-217 | 80-85 | Reserved for Basic |
| 218-3C3 | 86-F0 | Used by Basic Interpreter while BASIC is running |
| 3C4-3FF | F1-FF | Reserved |

**BIOS, BASIC, and DOS Reserved Interrupts**

The following is a list of reserved memory locations.

| Address (Hex) | Mode | Function |
|---|---|---|
| 400-48F | ROM BIOS | See BIOS Listing |
| 490-4EF | | Reserved for System Usage |
| 500-5FF | | Communication Area for any application |
| 500 | DOS | Reserved for DOS and BASIC, 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**

The following is a list of the BASIC workspace variables.

| If you do DEF SEG (Default workspace segment): | Offset (Hex) | 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<br>if 0-no characters in buffer<br>if 1-characters in buffer | 6A | 1 |
| Character color in graphics mode<br>Set to 1, 2, or 3 to get text in colors 1 to 3.<br>Do not set to 0.<br>(Default = 3) | 4E | 1 |

Example

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

**BASIC Workspace Variables**

The following shows the mapping of the BIOS memory

| Starting Address in Hex | |
|---|---|
| 00000 | BIOS Interrupt Vectors |
| 00400 | BIOS Data Area |
| 00500 | User Read/Write Memory |
| A0000 | Reserved for Future Video |
| B8000 | Reserved for Video |
| C0000 | Reserved for Future I/O ROM |
| D0000 | Reserved for Cartridges |
| E0000 | Reserved for Cartridges |
| F0000 | BIOS/ Diagnostics/ Cassette and BASIC Program Area |

**BIOS System Map**

# BIOS Programming Guidelines

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 diskette code, you should 'reset' the drive adapter and retry the operation. A specified number of retries should be required on diskette 'reads' to insure 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 systems.

# 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 Power-On Self-Test (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 C0000 through hex D0000 are scanned in 2K-byte 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:** length (multiple of 2K bytes) - 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

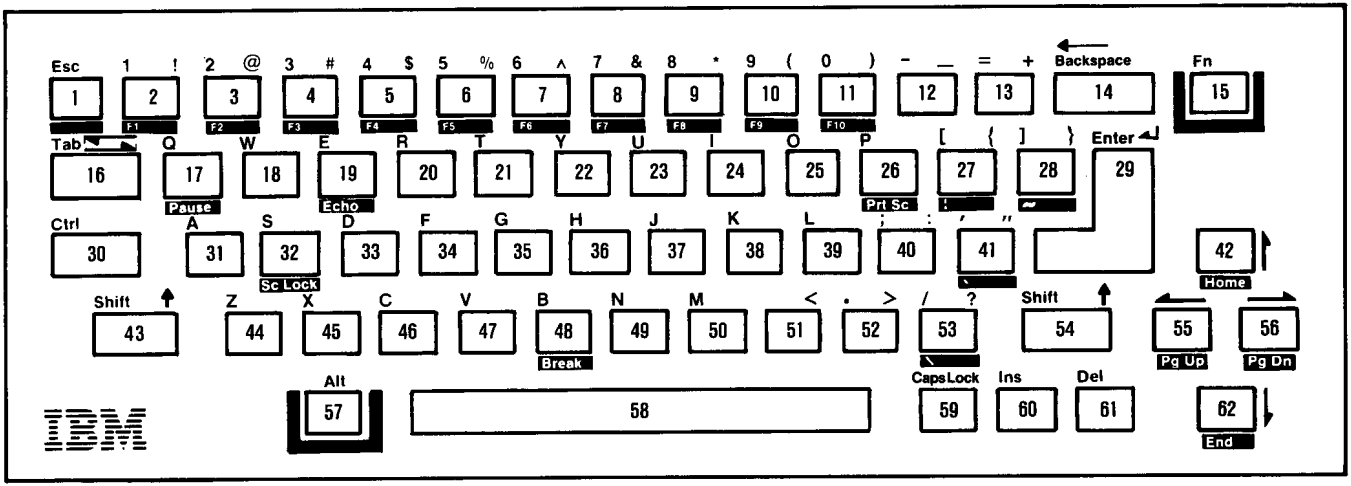
The following explains how the keyboard interacts with BIOS and how 83-key-keyboard functions are accomplished on the Cordless Keyboard.

## Cordless Keyboard Encoding

The KEYBOARD routine provided by IBM in the ROM BIOS is responsible for converting the keyboard scan-codes into what is 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.

The following is the physical layout of the IBM PC*jr* Cordless Keyboard.




**IBM PC*jr* Cordless Keyboard Diagram**

The following are charts of the scan codes for the IBM PC*jr* Cordless Keyboard.

| Key Position | Keyboard Characters | Make Code (Hex) | Break Code (Hex) |
|---|---|---|---|
| 1 | ESC | 1 | 81 |
| 2 | 1/! | 2 | 82 |
| 3 | 2/∂ | 3 | 83 |
| 4 | 3/# | 4 | 84 |
| 5 | 4/$ | 5 | 85 |
| 6 | 5/% | 6 | 86 |
| 7 | 6/≙ | 7 | 87 |
| 8 | 7/& | 8 | 88 |
| 9 | 8/* | 9 | 89 |
| 10 | 9/( | A | 8A |
| 11 | 0/) | B | 8B |
| 12 | -/— | C | 8C |
| 13 | =/+ | D | 8D |
| 14 | BS<— | E | 8E |
| 15 | FN | 54 | D4 |
| 16 | TAB | F | 8F |
| 17 | q/Q | 10 | 90 |
| 18 | w/W | 11 | 91 |
| 19 | e/E | 12 | 92 |
| 20 | r/R | 13 | 93 |
| 21 | t/T | 14 | 94 |
| 22 | y/Y | 15 | 95 |
| 23 | u/U | 16 | 96 |
| 24 | i/I | 17 | 97 |
| 25 | o/O | 18 | 98 |
| 26 | p/P | 19 | 99 |
| 27 | [/{ | 1A | 9A |
| 28 | ]/} | 1B | 9B |
| 29 | ENTER | 1C | 9C |
| 30 | CTRL | 1D | 9D |
| 31 | a/A | 1E | 9E |

**Cordless Keyboard Maxtrix Scan Codes (Part 1 of 2)**

| Key Position | Keyboard Characters | Make Code (Hex) | Break Code (Hex) |
|---|---|---|---|
| 32 | s/S | 1F | 9F |
| 33 | d/D | 20 | A0 |
| 34 | f/F | 21 | A1 |
| 35 | g/G | 22 | A2 |
| 36 | h/H | 23 | A3 |
| 37 | j/J | 24 | A4 |
| 38 | k/K | 25 | A5 |
| 39 | l/L | 26 | A6 |
| 40 | ;/: | 27 | A7 |
| 41 | '/" | 28 | A8 |
| 42 | CUR.UP | 48 | C8 |
| 43 | LF.SHIFT | 2A | AA |
| 44 | z/Z | 2C | AC |
| 45 | x/X | 2D | AD |
| 46 | c/C | 2E | AE |
| 47 | v/V | 2F | AF |
| 48 | b/B | 30 | B0 |
| 49 | n/N | 31 | B1 |
| 50 | m/M | 32 | B2 |
| 51 | ,/< | 33 | B3 |
| 52 | ./> | 34 | B4 |
| 53 | //? | 35 | B5 |
| 54 | RT.SHIFT | 36 | B6 |
| 55 | CUR.LF. | 4B | CB |
| 56 | CUR.RT. | 4D | CD |
| 57 | ALT. | 38 | B8 |
| 58 | SP.BAR | 39 | B9 |
| 59 | CAPS LOCK | 3A | BA |
| 60 | INSERT | 52 | D2 |
| 61 | DELETE | 53 | D3 |
| 62 | CUR.DWN. | 50 | D0 |
| Phantom-Key Scan Code | | 55 | |

**Cordless Keyboard Matrix Scan Codes (Part 2 of 2)**

The Cordless Keyboard is unique to the PC*jr*. Even though it does not possess all 83 keys of the IBM Personal Computer keyboard, it does have a way in which you can cause all of the scan codes of the 83-key keyboard. The following chart shows the mapping of functions between both keyboards:

| IBM Personal Computers 83-key Keyboard Function | IBM PC*jr* Cordless Keyboard Mapping |
|---|---|
| F1-F10 | Function key + 1-0 (F1-F10) |
| Ctrl Break | Function key + B (Break) |
| Ctrl PrtSc (Echo Print) | Function key + E (Echo) |
| Shift PrtSc (Print Screen) | Function key + P (PrtSc) |
| Ctrl NumLock (Pause) | Function key + Q (Pause) |
| Scroll Lock | Function key + S (ScLock) |
| Numeric keypad region: Num Lock (Number keypad 1 through 10 becomes key scan codes.) | Alt + Function key + N (1 through 0 becomes numeric-key scan-codes) |
| PgUp key | Function key + cursor left (PgUp) |
| PgDn key | Function key + cursor right (PgDn) |
| Home key | Function key + cursor up (Home) |
| End key | Function key + cursor down (End) |
| Numeric keypad - sign | Function key plus the - sign |
| Numeric keypad + sign | Function key + = sign |
| \ key | Alt + / |
| ' key | Alt + ' |
| ! key | Alt + [ |
| ~ key | Alt + ] |
| * with PrtSc | Alt + . |
| Numeric keypad . | Shift + Del |
| All 256 extended codes: Alt + numeric value from numeric keypad | NumLock then Alt + numeric value (1 through 0) |

**83-key-Keyboard Function to Cordless-Keyboard Mapping**

## 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, "Characters, Keystrokes,and Color" for the exact codes.

| Key Number | Base Case | Upper Case | Ctrl | Alt | Fn |
|---|---|---|---|---|---|
| 1 | Esc | Esc | Esc | –1 | ** |
| 2 | 1 | ! | –1 | *,***** | (F1) *,*** |
| 3 | 2 | ∂ | Nul (000) | *,***** | (F2) *,*** |
| 4 | 3 | # | –1 | *,***** | (F3) |
| 5 | 4 | $ | –1 | *,***** | (F4) *,*** |
| 6 | 5 | % | –1 | *,***** | (F5) *,*** |
| 7 | 6 | ≙ | RSO (030) | *,***** | (F6) *,*** |
| 8 | 7 | & | –1 | *,***** | (F7) *,*** |
| 9 | 8 | * | –1 | *,***** | (F8) *,*** |
| 10 | 9 | ( | –1 | *,***** | (F9) *,*** |
| 11 | 0 | ) | –1 | *,***** | (F10) *,*** |
| 12 | — | - | US (031) | * | *** |
| 13 | = | + | –1 | * | *** |
| 14 | Backspace (008) | Backspace (008) | DEL (127) | –1 | –1 |
| 15 Fn | –1 | –1 | –1 | –1 | –1 |
| 16 | —>\| (009) | \|<— * | –1 | –1 | –1 |
| 17 | q | Q | DC1 (017) | * | **,*** (Pause) |
| 18 | w | W | ETB (023) | * | –1 |
| 19 | e | E | ENQ (005) | * | **,*** (Echo) |
| 20 | r | R | DC2 (018) | * | –1 |
| 21 | t | T | DC4 (020) | * | –1 |

* - Refer to "Extended Codes" in this section.
** - Refer to "Special Handling" in this section.
*** - Refer to "83-Key Keyboard functions to Cordless Keyboard Mapping Chart."
**** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).
***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

**Cordless-Keyboard Character Codes (Part 1 of 4)**

| Key Number | Base Case | Upper Case | Ctrl | Alt | Fn |
|---|---|---|---|---|---|
| 22 | y | Y | EM (025) | * | -1 |
| 23 | u | U | NAK (021) | * | -1 |
| 24 | i | I | HT (009) | * | -1 |
| 25 | o | O | SI (015) | * | -1 |
| 26 | p | P | DLE (016) | * | **,*** (PrtScreen) |
| 27 | [ | { | Esc (027) | (\|) *** | -1 |
| 28 | ] | } | GS (029) | (~) *** | -1 |
| 29 | CR | CR | LF (010) | -1 | -1 |
| 30 Ctrl | -1 | -1 | -1 | -1 | -1 |
| 31 | a | A | SOH (001) | * | -1 |
| 32 | s | S | DC3 (019) | * | **,*** (Scroll Lock) |
| 33 | d | D | EOT (004) | * | -1 |
| 34 | f | F | ACK (006) | * | -1 |
| 35 | g | G | BELL (007) | * | -1 |
| 36 | h | H | BS (008) | * | -1 |
| 37 | j | J | LF (010) | * | -1 |
| 38 | k | K | VT (011) | * | -1 |
| 39 | l | L | FF (012) | * | -1 |
| 40 | ; | : | -1 | -1 | -1 |
| 41 | ' | " | -1 | (') *** | -1 |

* - Refer to "Extended Codes" in this section.
** - Refer to "Special Handling" in this section.
*** - Refer to "83-Key Keyboard functions to Cordless Keyboard Mapping Chart."
**** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).
***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

**Cordless-Keyboard Character Codes (Part 2 of 4)**

| Key Number | Base Case | Upper Case | Ctrl | Alt | Fn | Alt + Ctrl |
|---|---|---|---|---|---|---|
| 42 | Cur.Up* | 8 **** | -1 | * | **,*** (Home) | |
| 43 Left Shift | -1 | -1 | -1 | -1 | -1 | |
| 44 | z | Z | SUB (026) | * | -1 | |
| 45 | x | X | CAN (024) | * | -1 | |
| 46 | c | C | EXT (003) | * | -1 | |
| 47 | v | V | SYN (022) | * | -1 | |
| 48 | b | B | STX (002) | * | **,*** (Break) | |
| 49 | n | N | SO (014) | *,*** | *** | |
| 50 | m | M | CR (013) | * | -1 | |
| 51 | , | < | -1 | -1 | -1 | |
| 52 | . | > | -1 | (*) * | -1 | |
| 53 | / | ? | -1 | \ | -1 | |
| 54 Right Shift | -1 | -1 | -1 | -1 | | |
| 55 | Cur.L * | 4 **** | * Reverse Word | * | **,*** (PgUp) | ** |
| 56 | Cur.R * | 6 **** | * Advance Word | * | **,*** (PgDn) | ** **** |

* - Refer to "Extended Codes" in this section.
** - Refer to "Special Handling" in this section.
*** - Refer to "83-Key Keyboard functions to Cordless Keyboard Mapping Chart."
**** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).
***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

**Cordless-Keyboard Character Codes (Part 3 of 4)**

| Key Number | Base Case | Upper Case | Ctrl | Alt | Fn | Alt + Ctrl |
|---|---|---|---|---|---|---|
| 57 Alt | -1 | -1 | -1 | -1 | -1 | |
| 58 | Space | Space | Space | Space | Space | |
| 59 Caps Lock | -1 | -1 | -1 | -1 | -1 | ** |
| 60 | Ins. | 0 **** | -1 | * | -1 | ** |
| 61 | Del. * | . **** | -1 | * | -1 | ** |
| 62 | Cur.Dn * | 2 **** | -1 | * | **,*** End | |

* - Refer to "Extended Codes" in this section.
** - Refer to "Special Handling" in this section.
*** - Refer to "83-Key Keyboard functions to Cordless Keyboard Mapping Chart."
**** - Uppercase for cursor keys can be selected by pressing left or right shift or entering the Numlock state (Alt + Fn + N).
***** - When Alt is pressed and the keyboard is in the Numlock state, the upper row of digits is used to enter ASCII codes for generating any character from the extended ASCII character set.

**Cordless-Keyboard Character Codes (Part 4 of 4)**

## Extended Codes

An extended code is used for certain functions that cannot be represented in the standard ASCII code. A character code of 000 (Nul) is returned in AL. This indicates that the system or application program should examine a second code that indicates the actual function. This code is returned in AH. This is the same for both the Cordless Keyboard and 83-key keyboard.

| Second Code | Function |
|---|---|
| 3 | Null Character |
| 15 | ← |
| 16 through 25 | Alt Q, W, E, R, T, Y, U, I, O, P |
| 30 through 38 | Alt A, S, D, F, G, H, J, K, L |
| 44 through 50 | Alt Z, X, C, V, B, N, M |
| 59 through 68 | Fn + 1, 2, 3, 4, 5, 6, 7, 8, 9, 0 (Functions 1 through 10) |
| 71 | Home |
| 72 | Up Arrow |
| 73 | Page Up |
| 75 | ← (Cursor Left) |
| 77 | → (Cursor Right) |
| 79 | End |
| 80 | Down Arrow |
| 81 | Page Down |
| 82 | Ins (Insert) |
| 83 | Del (Delete) |
| 84 through 93 | F11 through F20 (Upper Case F1 through F10) |
| 94 through 103 | F21 through F30 (Ctrl F1 through F10) |
| 104 through 113 | F31 through F40 (Alt F1 through F10) |
| 114 | Fn/E or Ctrl/Fn/P (Start/Stop Echo to Printer) |
| 115 | Ctrl ← (Reverse Word) |
| 116 | Ctrl → (Advance Word) |
| 117 | Ctrl/End [Erase End of Line (EOL)] |
| 118 | Ctrl/PgDn [Erase to End of Screen (EOS)] |
| 119 | Ctrl/Home (Clear Screen and Home) |
| 120 through 131 | Alt/1, 2, 3, 4, 5, 6, 7, 8, 9, 0, -, = (Keys 2 through 13) |
| 132 | Ctrl/PgUp (Top 25 Lines of Text and Home Cur.) |
| 133 through 149 | Reserved |
| 150 through 190 | Reserved for Non-Keyboard Scan Codes |

**Cordless Keyboard Extended Functions**



## Shift States

Most shift states are handled within the KEYBOARD routine, transparent to the system or application

program. The current set of active shift states is 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 thru 13, 16 thru 28, 31 thru 41, and 44 thru 53 to upper case (base case if in Caps Lock state). The **Shift key** temporarily reverses the 'Num Lock' or 'non-Num-Lock' state of keys 42, 55, 56, and 60 thru 62.

### Ctrl

This key temporarily shifts keys 3, 7, 12, 14, 16 thru 28, 30 thru 38, 42, 44 thru 50, 55, and 56 to the Ctrl state. 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, with the **Num Lock key** to cause the 'Pause' function, with the **Alt** and **Cursor Left** or **Right** for 'screen adjustment', with **Alt** and **Ins** to 'activate diagnostics', and with **Alt** and **CapsLock** to 'activate keyboard clicking'. These functions are described in "Special Handling" on the following pages.

### Alt

The **Alt** key temporarily shifts keys 2 thru 13, 17 thru 26, 31 thru 39, and 44 thru 50 to the 'Alternate state'. 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 is also used with keys 27, 28, 41, and 53 to produce the characters under the key.

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 must first put the keyboard in the 'Num Lock' state (concurrently press, first **Alt** then **Fn + n**). Then while holding down the **Alt** key type the decimal value of the character desired using keys 2 thru 11. 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 17 thru 25, 31 thru 39, and 44 thru 50 to 'upper case'. A second press of the **Caps Lock** key reverses the action. **Caps Lock** is handled internal to the KEYBOARD routine.

### Shift-Key Priorities and Combinations

The following keys are listed in descending priority for translation in Interrupt Hex 48 and Interrupt hex 9 respectively:

1. Interrupt Hex 48
   a. **Alt** key
   b. **Ctrl** key
   c. **Shift** key
2. Interrupt Hex 9
   a. **Ctrl**
   b. **Alt**
   c. **Shift**

Of the three keys listed, only **Alt** and **Ctrl** are a valid combination. If any other combination of the three keys is used, only the key with the higher priority is recognized by the system.

# Special Handling

## System Reset

The combination of the **Alt, Ctrl,** and **Del** keys causes the KEYBOARD routine to initiate the equivalent of a 'System Reset'.

## Break

The combination of the **Fn** and **B** keys results in the KEYBOARD routine signaling Interrupt Hex 1A. The extended characters (AL = hex 00, AH = hex 00) are returned.

## Pause

The combination of the **Fn** and **Q** keys causes the KEYBOARD-interrupt routine to loop, waiting for any key to be pressed. This provides a system or application-transparent method of temporarily suspending an operation such as list or print and then resuming the operation by pressing any other key. The key pressed to exit the 'Pause' mode is unused otherwise.

## Print Screen

The combination of the **Fn** and **P** keys results in an interrupt, invoking the PRINT SCREEN routine. This

routine works in the alphanumeric or graphics mode, with unrecognizable characters printing as blanks.

## Scroll Lock

The combination of the **Fn** and **S** key is interpreted by appropriate application programs to indicate that the cursor-control keys should cause 'windowing' over the text rather than cursor movement. Pressing the 'Scroll Lock' combination a second time reverses the action. The KEYBOARD routine simply records the current shift state of 'Scroll Lock'. It is the responsibility of the system or application program to perform the function.

## Functions 1 thru 10

The combination of the **Fn** key (15) and one of keys 2 thru 11 results in the corresponding 'Function' with key 2 being 'F1' up to key 11 being 'F10'.

## Function Lock

Concurrently pressing first the **Fn** key and **Shift** key, and then pressing the **Esc** key causes keys 2 thru 11 to shift to their 'Function' states and remain there until the same combination is pressed again.

## Screen Adjustment

The combination of the **Alt** key, **Ctrl** key, and either the **Left** or **Right** cursor movement key causes the screen to shift one character in the corresponding direction, up to a maximum of four.

## Enable/Disable Keyboard Click

The combination of the **Alt**, **Ctrl**, and **Caps Lock** keys causes the keyboard audio feedback (click) to shift between 'on' and 'off'. The Power-On default is 'off'.

## Run Diagnostics

The combination of the **Alt**, **Ctrl**, and **Ins keys** causes the system diagnostics stored in ROM to be initiated.

## Phantom-Key Scan-Code (Hex 55)

The Phantom-Key scan-code is generated by the keyboard when an invalid combination of three or more keys is pressed. The keys pressed that caused the Phantom-Key scan-code are not put into the keyboard buffer, and are ignored by the keyboard microprocessor. The Phantom-Key scan-code is transmitted to BIOS where it is ignored.

## Other Characteristics

The keyboard buffer is large enough to support a fast typist. If a key is pressed when the buffer is full, the character generated is ignored and the 'bell' is sounded. A larger buffer can be specified by modifying words at labels 'Buffer-Start' (hex 480) and 'Buffer-End' (hex 482) to point to another offset within segment hex 40.

The KEYBOARD routine suppresses the typematic action of the following keys: **Ctrl, Shift, Alt, Caps Lock, Insert**, and **Function.**

| Function | Key Combinations | Description |
|---|---|---|
| System Reset | Alt + Ctrl + Del | Unconditional system reset |
| Break | Fn + B | Breaks program execution |
| Pause | Fn + Q | Resumable pause in program execution |
| Print Screen | Fn + P | |
| Function Lock | Fn and Shift then Esc (Held) concurrently) | Locks the number keys as Function keys (F1-F10) and B, Q, P, E, S, and the cursor control keys to their function states |
| Screen Adjustment | Alt + Ctrl + cursor right or cursor left | Allows the user to adjust the display's image left or right |
| Keyboard Click | Alt + Ctrl + CapsLock | Enables or disables the keyboard audio feedback click |
| Run Diagnostics | Alt + Ctrl + Ins | Initiates system ROM diagnostics |
| Keyboard Adventure Game | Esc | If the first key pressed after the system comes up in Cassette BASIC is Esc (key #1) then the Keyboard Adventure Game will be activated. |
| Cassette Autoload | Ctrl + Esc | If this is the first key sequence after the system comes up in Cassette BASIC then the screen will display 'Load "CAS1:",R followed by a Carriage Return. This allows a cassette program to be automatically loaded. |

**Cordless Keyboard Special Handling**

## Keyboard Usage

"Keyboard Usage" is a set of guidelines of key-usage when performing commonly-used functions.

| Function | Keys | Comment |
|---|---|---|
| Home Cursor | Fn Home | Editors; word processors |
| Return to outermost menu | Fn Home | Menu driven applications |
| Move cursor up | Up Arrow | Full screen editor, word processor |
| Page up, scroll backwards 25 lines | Fn PgUp | Editors; word processors |
| Move cursor left | ← | Text, command entry |
| Move cursor right | → | Text, command entry |
| Scroll to end of text place cursor at end of line | Fn End | Editors; word processors |
| Move cursor down | Down Arrow | Full screen editor, word processor |
| Page down, scroll forwards 25 lines and home | Fn PgDn | Editors; word processors |
| Start/Stop insert text at cursor, shift text right in buffer | Ins | Text, command entry |

**Keyboard - Commonly Used Functions (Part 1 of 3)**

| Function | Keys | Comment |
|---|---|---|
| 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 Fn Home | |
| Scroll up | Up Arrow | In scroll lock mode |
| Scroll down | Down Arrow | In scroll lock mode |
| Scroll left | ← | In scroll lock mode |
| Scroll right | → | In scroll lock mode |
| Delete from cursor to EOL (end of line) | Ctrl Fn End | Text, command entry |
| Exit/Escape | Esc | Editor, 1 level of menu and so on |
| Start/Stop Echo screen to printer | Fn PrtSc | Any time |
| Delete from cursor to EOS (end of screen) | Ctrl Fn PgDn | Text, command entry |
| Advance word | Ctrl → | Text entry |
| Reverse word | Ctrl ← | Text entry |
| Window Right | Ctrl → | When text is too wide to fit the screen |

**Keyboard - Commonly Used Functions (Part 2 of 3)**

| Function | Keys | Comment |
|---|---|---|
| Window Left | Ctrl ← | When text is too wide to fit the screen |
| Enter insert mode | Ins | Line Editor |
| Exit insert mode | Ins | Line Editor |
| Cancel current line | Esc | Command entry, text entry |
| Suspend system (Pause) | Ctrl Fn Pause | Stop list, stop program, and so on. Resumes on any key. |
| Break interrupt | Fn Break | Interrupt current process |
| System reset | Alt Ctrl Del | Reboot |
| Top of document and home cursor | Ctrl Fn PgUp | Editors, word processors |
| Standard function keys | Shift Fn/F1 through Fn/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 through 13 (1 through 9, 0) (–, =) | Line Editor |
| Extra function keys | Alt A through Z | Used when function starts with the same letter as one of the alpha keys. |

**Keyboard - Commonly Used Functions (Part 3 of 3)**

| Function | Key |
|---|---|
| Carriage return | ↵ (Enter) |
| Line feed | Ctrl ↵ (Enter) |
| Bell | Ctrl G |
| Home | Fn Home |
| Cursor up | Up Arrow |
| Cursor down | Down Arrow |
| Cursor left | ← |
| Cursor right | → |
| Advance one word | Ctrl ← |
| Reverse one word | Ctrl → |
| Insert | Ins |
| Delete | Del |
| Clear screen | Ctrl Fn Home |
| Freeze output | Fn Pause |
| Tab advance | → |
| Stop Execution (break) | Fn Break |
| Delete current line | Esc |
| Delete to end of line | Ctrl Fn End |
| Position cursor to end of line | Fn End |

**BASIC Screen Editor Special Functions**

| Function | Key |
|---|---|
| Suspend | Fn Pause |
| Echo to printer | Fn Echo |
| Stop echo to printer | Fn Echo |
| Exit current function (break) | Fn Break |
| Backspace | ← Key 14 |
| Line feed | Ctrl ↵ (Enter) |
| Cancel line | Esc |
| Copy character | Fn F1 or → |
| Copy until match | Fn F2 |
| Copy remaining | Fn F3 |
| Skip character | Del |
| Skip until match | Fn F4 |
| Enter insert mode | Ins |
| Exit insert mode | Ins |
| Make new line the template | Fn F5 |
| String separator in REPLACE | Fn F6 |
| End of file in keyboard input | Fn F6 |

**DOS Special Functions**

# Non-Keyboard Scan-code Architecture

The architecture of the IBM PC*jr* BIOS is designed to also receive scan codes above those generated by the keyboard to accommodate any future device.

The keyboard generates scan codes from hex 1 to 55 and FF. Any scan codes above hex 55 (56 thru 7E for 'make' codes and D6 thru FE for 'break' codes) are processed by BIOS in the following manner:

1. If the incoming 'make' scan code falls within the range of the translate table, whose address is pointed to by BIOS Interrupt Hex 49, it is translated into the corresponding scan code. Any incoming 'break' codes above hex D5 are ignored.

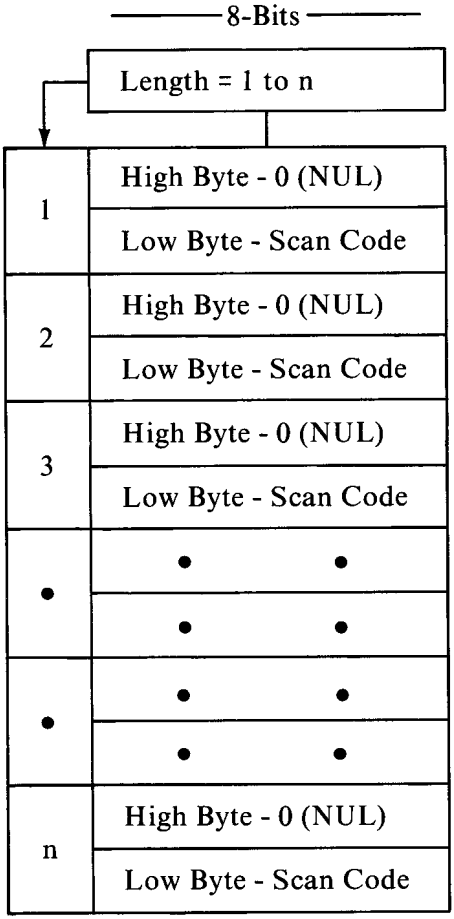
2. If the new translated scan code is less than hex 56, it is processed by BIOS as a keyboard scan-code and the same data is placed in the BIOS keyboard buffer.
3. If the translated scan-code is greater than hex 55 or the incoming scan-code is outside the range of the translate table, hex 40 is added, creating a new extended-scan-code. The new extended-scan-code is then placed in the BIOS keyboard buffer with the character code of 00(null). This utilizes the range hex 96 thru BE for scan codes hex 56 thru 7E respectively.

The default translate-table maps scan codes hex 56 thru 6A to existing keyboard-values. Scan codes hex 6B thru BE are mapped (by adding hex 40) to extended codes of hex AB thru FE, since these are out side the range of the default translate-table.

Users can modify Interrupt Hex 49 to address their own translate table if mapping differences are desired.

The translate table format is:

| | **Description** |
|---|---|
| **0** | Length - The number of non-keyboard scan-codes that are mapped within the table (from 1 to n). |
| **1 to n** | Word with low-order byte representing the scan-code-mapped values relative to the input values in the range of hex 56 thru 7E. |




**Translate Table Format**

With this architecture, all keyboard scan-codes can be intercepted thru Interrupt Hex 9 and all non-keyboard scan-codes can be intercepted thru Interrupt Hex 48.

The following is a chart showing the default values of the translate table in BIOS.

| Length = 20 mapped values | | |
|---|---|---|
| **Input Scan Code** | **Mapped Value** | **Keyboard Character** |
| 86 | 72 | (cursor up) |
| 87 | 73 | PgUp |
| 88 | 77 | (cursor right) |
| 89 | 81 | PgDn |
| 90 | 80 | (cursor down) |
| 91 | 79 | End |
| 92 | 75 | (cursor left) |
| 93 | 71 | Home |
| 94 | 57 | Space |
| 95 | 28 | Enter |
| 96 | 17 | W |
| 97 | 18 | E |
| 98 | 31 | S |
| 99 | 45 | X |
| 100 | 44 | Z |
| 101 | 43 | \ |
| 102 | 30 | A |
| 103 | 16 | Q |
| 104 | 15 | Tab |
| 105 | 1 | Esc |

**Translate Table Default Values**

| Scan Codes (Hex) | Type of Scan Code |
|---|---|
| 1 - 55 | Normal Keyboard Scan Code (Make) |
| 56 - 7E | Non-Keyboard Scan Code (Make) |
| 81 - D5 | Normal Keyboard Scan Code (Break) |
| D6 - FE | Non-Keyboard Scan Code (Break) |
| FF | Keyboard Buffer Full |

**Scan-Code Map**

# Notes:

## Software Algorithms - Interrupt Hex 15

The CASSETTE routine is called by the request type in AH. The address of the bytes to be 'read' from or 'written' to the tape is specified by DS:BX and the number of bytes to be 'read' or 'written' is specified by CX. The actual number of bytes 'read' is returned in DX. The read block and write block 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<br>Read CX bytes into memory starting at Address DS:BX<br>Return actual number of bytes read in DX<br>Return Cassette Status in AH |
| AH = 3 | Write Tape Block<br>Write CX bytes onto cassette starting at Address DS:BX<br>Return Cassette Status in AH |

**AH Request Types**

| 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. | |

**AH Cassette Status**

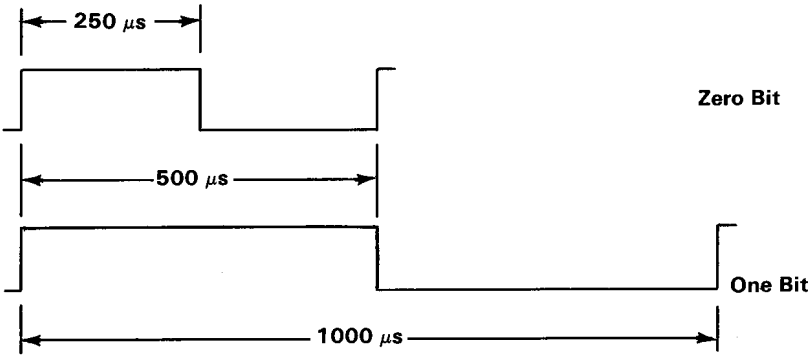
# Cassette Write

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

The WRITE-BLOCK routine 'turns on' the cassette drive motor and 'writes' the leader (256 bytes of all 1's) to the tape, 'writes' a synchronization bit (0), and then 'writes' a synchronization byte (ASCII character hex 16). Next, the routine 'writes' the number of data bytes 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 DS:BX.

The WRITE-BLOCK 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 count register. The timer's period is changed on the fly for each data byte '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. All 8259 interrupts are 'disabled' during cassette-write operations.




**Cassette-Write Timing Chart**

## 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

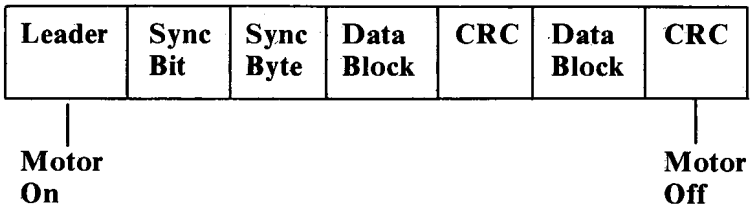
(ASCII character hex 16) is 'read'. If the sync byte is 'read' correctly, the data portion can be 'read'. If a correct sync byte is not found, the routine goes back and searches for the leader again. The data is 'read' a bit-at-a-time and 'assembled' into bytes. After each byte is 'assembled', it is 'written' into memory at location DS: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 exits with the carry flag 'set' to indicate an error and the status of AH 'set' to hex 01. DX contains the number of bytes 'written' into memory.

All 8259 interrupts are 'disabled' during the cassette-'read' operations.

## Data Record Architecture

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




Cassette Write-Block Format

| 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 Detection

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

# Notes:

# Appendixes

## Contents

```
                        ;--------------------------------------------------------
                        ; <CAVEAT EMPTOR>:
                        ;
                        ;       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 THIS  CODE  VIOLATE THE
                        ;       STRUCTURE AND DESIGN OF BIOS.
                        ;--------------------------------------------------------
                        ;---------------------------------------
                        ;                EQUATES
                        ;---------------------------------------
= 0060                  PORT_A          EQU     60H     ; 8255 PORT A ADDR
= 0038                  CPUREG          EQU     38H     ; MASK FOR CPU REG BITS
= 0007                  CRTREG          EQU     7       ; MASK FOR CRT REG BITS
= 0061                  PORT_B          EQU     61H     ; 8255 PORT B ADDR
= 0062                  PORT_C          EQU     62H     ; 8255 PORT C ADDR
= 0063                  CMD_PORT        EQU     63H
= 0089                  MODE_8255       EQU     10001001B
= 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
= 0061                  KB_CTL          EQU     61H     ; CONTROL BITS FOR KEYBOARD
= 03DA                  VGA_CTL         EQU     3DAH    ; VIDEO GATE ARRAY CONTROL PORT
= 00A0                  NMI_PORT        EQU     0A0H    ; NMI CONTROL PORT
= 00B0                  PORT_B0         EQU     0B0H
= 03DF                  PAGREG          EQU     03DFH   ; CRT/CPU PAGE REGISTER
= 0060                  KBPORT          EQU     060H    ; KEYBOARD PORT
= 4000                  DIAG_TABLE_PTR  EQU     4000H
= 2000                  MINI            EQU     2000H
                        ;---------------------------------------
                        ;          DISKETTE EQUATES
                        ;---------------------------------------
= 00F2                  NEC_CTL         EQU     0F2H    ; CONTROL PORT FOR THE DISKETTE
= 00B0                  FDC_RESET       EQU     80H     ; RESETS THE NEC (FLOPPY DISK
                                                        ; CONTROLLER).  0 RESETS,
                                                        ; 1 RELEASES THE RESET
= 0020                  WD_ENABLE       EQU     20H     ; ENABLES WATCH DOG TIMER IN NEC
= 0040                  WD_STROBE       EQU     40H     ; STROBES WATCHDOG TIMER
= 0001                  DRIVE_ENABLE    EQU     01H     ; SELECTS AND ENABLES DRIVE

= 00F4                  NEC_STAT        EQU     0F4H    ; STATUS REGISTER FOR THE NEC
= 0020                  BUSY_BIT        EQU     20H     ; BIT = 0 AT END OF EXECUTION PHASE
= 0040                  DIO             EQU     40H     ; INDICATES DIRECTION OF TRANSFER
= 0080                  RQM             EQU     80H     ; REQUEST FOR MASTER
= 00F5                  NEC_DATA        EQU     0F5H    ; DATA PORT FOR THE NEC
                        ;---------------------------------------
                        ;        8088 INTERRUPT LOCATIONS
                        ;---------------------------------------
0000                    ABS0    SEGMENT AT 0
0008                            ORG     2*4
0008                    NMI_PTR         LABEL   WORD
000C                            ORG     3*4
000C                    INT3_PTR        LABEL   WORD
0014                            ORG     5*4
0014                    INT5_PTR        LABEL   WORD
0020                            ORG     8*4
0020                    INT_PTR         LABEL   DWORD
0040                            ORG     10H*4
0040                    VIDEO_INT       LABEL   WORD
0070                            ORG     1CH*4
0070                    INT1C_PTR       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
0110                            ORG     044H*4
0110                    CSET_PTR        LABEL   DWORD   ; POINTER TO DOT PATTERNS
0120                            ORG     048H*4
0120                    KEY62_PTR       LABEL   WORD    ; POINTER TO 62 KEY KEYBOARD CODE
0124                            ORG     049H*4
0124                    EXST            LABEL   WORD    ; POINTER TO EXT. SCAN TABLE
0204                            ORG     081H*4
0204                    INT81   LABEL   WORD
0208                            ORG     082H*4
0208                    INT82   LABEL   WORD
0224                            ORG     089H*4
0224                    INT89   LABEL   WORD
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
7C00                    ABS0    ENDS
```

```
                        ;--------------------------------------------
                        ; STACK -- USED DURING INITIALIZATION ONLY
                        ;--------------------------------------------
0000                    STACK   SEGMENT AT 30H
0000    80 [                    DW      128 DUP(?)
            ????
                ]

0100                    TOS     LABEL   WORD
0100                    STACK   ENDS
                        ;--------------------------------------
                        ;       ROM BIOS DATA AREAS
                        ;--------------------------------------
0000                    DATA    SEGMENT AT 40H
0000    04 [            RS232_BASE      DW      4 DUP(?) ; ADDRESSES OF RS232 ADAPTERS
            ????
                ]

0008    04 [            PRINTER_BASE    DW      4 DUP(?) ; ADDRESSES OF PRINTERS
            ????
                ]

0010  ????              EQUIP_FLAG      DW      ?       ; INSTALLED HARDWARE
0012  ??                KBD_ERR         DB      ?       ; COUNT OF KEYBOARD TRANSMIT ERRORS
0013  ????              MEMORY_SIZE     DW      ?       ; USABLE MEMORY SIZE IN K BYTES
0015  ????              TRUE_MEM        DW      ?       ; REAL MEMORY SIZE IN K BYTES
                        ;--------------------------------------
                        ;       KEYBOARD DATA AREAS
                        ;--------------------------------------
0017  ??                KB_FLAG         DB      ?
                        ;----- SHIFT FLAG EQUATES WITHIN KB_FLAG
= 0040                  CAPS_STATE      EQU     40H     ; CAPS LOCK STATE HAS BEEN TOGGLED
= 0020                  NUM_STATE       EQU     20H     ; NUM 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
= 0004                  CLICK_ON        EQU     04H     ; INDICATES THAT AUDIO FEEDBACK IS
                                                        ; ENABLED
= 0002                  CLICK_SEQUENCE  EQU     02H     ; OCURRNCE OF ALT-CTRL-CAPSLOCK HAS
                                                        ; OCCURED
0019  ??                ALT_INPUT       DB      ?       ; STORAGE FOR ALTERNATE KEYPAD
                                                        ; ENTRY
001A  ????              BUFFER_HEAD     DW      ?       ; POINTER TO HEAD OF KEYBOARD BUFF
001C  ????              BUFFER_TAIL     DW      ?       ; POINTER TO TAIL OF KEYBOARD BUFF
001E    10 [            KB_BUFFER       DW      16 DUP(?) ; ROOM FOR 15 ENTRIES
            ????
                ]

                        ;------ 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 0 = DRIVE NEEDS RECAL BEFORE
                                                        ;   NEXT SEEK IF BIT IS = 0
003F  ??                MOTOR_STATUS    DB      ?       ; MOTOR STATUS
                                                        ; BIT 0 = DRIVE 0 IS CURRENTLY
                                                        ;   RUNNING
0040  ??                MOTOR_COUNT     DB      ?       ; TIME OUT COUNTER FOR DRIVE
                                                        ; TURN OFF
= 0025                  MOTOR_WAIT      EQU     37      ; 2 SECS OF COUNTS FOR MOTOR
                                                        ; TURN OFF
0041  ??                DISKETTE_STATUS DB      ?       ; RETURN CODE STATUS BYTE
= 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
                                                        ; PROTECTED DISK
= 0002                  BAD_ADDR_MARK   EQU     02H     ; ADDRESS MARK NOT FOUND
= 0001                  BAD_CMD         EQU     01H     ; BAD COMMAND GIVEN TO DISKETTE I/O
0042    07 [            NEC_STATUS      DB      7 DUP(?) ; STATUS BYTES FROM NEC
            ??
                ]

= 0020                  SEEK_END        EQU     20H
= 012C                  THRESHOLD       EQU     300     ; NUMBER OF TIMER-0 TICKS TILL
                                                        ; ENABLE
= 00AF                  PARM0           EQU     0AFH    ; PARAMETER 0 IN THE DISK_PARM
                                                        ; TABLE
= 0003                  PARM1           EQU     3       ; PARAMETER 1
= 0019                  PARM9           EQU     25      ; PARAMETER 9
= 0004                  PARM10          EQU     4       ; PARAMETER 10
```

```
                        ;--------------------------------------------
                        ;       VIDEO DISPLAY DATA AREA
                        ;--------------------------------------------
0049  ??                CRT_MODE        DB      ?       ; CURRENT CRT MODE
004A  ????              CRT_COLS        DW      ?       ; NUMBER OF COLUMNS ON SCREEN
004C  ????              CRT_LEN         DW      ?       ; LENGTH OF REGEN IN BYTES
004E  ????              CRT_START       DW      ?       ; STARTING ADDRESS IN REGEN BUFFER
0050     08 [           CURSOR_POSN     DW      8 DUP(?) ; CURSOR FOR EACH OF UP TO 8 PAGES
                ????
                     ]

0060  ????              CURSOR_MODE     DW      ?       ; CURRENT CURSOR MODE SETTING
0062  ??                ACTIVE_PAGE     DB      ?       ; CURRENT PAGE BEING DISPLAYED
0063  ????              ADDR_6845       DW      ?       ; BASE ADDRESS FOR ACTIVE DISPLAY
                                                        ; CARD
0065  ??                CRT_MODE_SET    DB      ?       ; CURRENT SETTING OF THE
                                                        ; CRT MODE REGISTER
0066  ??                CRT_PALLETTE    DB      ?       ; CURRENT PALETTE MASK SETTING
                        ;--------------------------------------------
                        ;       CASSETTE DATA AREA
                        ;--------------------------------------------
0067  ????              EDGE_CNT        DW      ?       ; TIME COUNT AT DATA EDGE
0069  ????              CRC_REG         DW      ?       ; CRC REGISTER
006B  ??                LAST_VAL        DB      ?       ; LAST INPUT VALUE
                        ;--------------------------------------------
                        ;       TIMER DATA AREA
                        ;--------------------------------------------
006C  ????              TIMER_LOW       DW      ?       ; LOW WORD OF TIMER COUNT
006E  ????              TIMER_HIGH      DW      ?       ; HIGH WORD OF TIMER COUNT
0070  ??                TIMER_OFL       DB      ?       ; TIMER HAS ROLLED OVER SINCE LAST
                                                        ; READ
                        ;--------------------------------------------
                        ;       SYSTEM DATA AREA
                        ;--------------------------------------------
0071  ??                BIOS_BREAK      DB      ?       ; BIT 7=1 IF BREAK KEY HAS BEEN HIT
0072  ????              RESET_FLAG      DW      ?       ; WORD=1234H IF KEYBOARD RESET
                                                        ; UNDERWAY
                        ;--------------------------------------------
                        ;       EXTRA DISKETTE DATA AREAS
                        ;--------------------------------------------
0074  ??                TRACK0          DB      ?
0075  ??                TRACK1          DB      ?
0076  ??                TRACK2          DB      ?
0077  ??                                DB      ?
                        ;--------------------------------------------
                        ;       PRINTER AND RS232 TIME-OUT VARIABLES
                        ;--------------------------------------------
0078     04 [           PRINT_TIM_OUT   DB      4 DUP(?)
                ??
                     ]

007C     04 [           RS232_TIM_OUT   DB      4 DUP(?)
                ??
                     ]

                        ;--------------------------------------------
                        ;       ADDITIONAL KEYBOARD DATA AREA
                        ;--------------------------------------------
0080  ????              BUFFER_START    DW      ?
0082  ????              BUFFER_END      DW      ?
0084  ??                INTR_FLAG       DB      ?       ; FLAG TO INDICATE AN INTERRUPT
                                                        ; HAPPENED
                        ;--------------------------------------------
                        ;  62 KEY KEYBOARD DATA AREA
                        ;--------------------------------------------
0085  ??                CUR_CHAR        DB      ?       ; CURRENT CHARACTER FOR TYPAMATIC
0086  ??                VAR_DELAY       DB      ?       ; DETERMINES WHEN INITIAL DELAY IS
                                                        ; OVER
= 000F                  DELAY_RATE      EQU     0FH     ; INCREASES INITIAL DELAY
0087  ??                CUR_FUNC        DB      ?       ; CURRENT FUNCTION
0088  ??                KB_FLAG_2       DB      ?       ; 3RD BYTE OF KEYBOARD FLAGS
= 0004                  RANGE           EQU     4       ; NUMBER OF POSITIONS TO SHIFT
                                                        ; DISPLAY
                        ;--------------------------------------------
                        ;  BIT ASSIGNMETS FOR KB_FLAG_2
                        ;--------------------------------------------
= 0080                  FN_FLAG         EQU     80H
= 0040                  FN_BREAK        EQU     40H
= 0020                  FN_PENDING      EQU     20H
= 0010                  FN_LOCK         EQU     10H
= 0008                  TYPE_OFF        EQU     08H
= 0004                  HALF_RATE       EQU     04H
= 0002                  INIT_DELAY      EQU     02H
= 0001                  PUTCHAR         EQU     01H
0089  ??                HORZ_POS        DB      ?       ; CURRENT VALUE OF HORIZONTAL
                                                        ; START PARM
008A  ??                PAGDAT          DB      ?       ; IMAGE OF DATA WRITTEN TO PAGREG
0088                    DATA    ENDS
                        ;--------------------------------------------
                        ;       EXTRA DATA AREA
                        ;--------------------------------------------
0000                    XXDATA  SEGMENT AT 50H
0000  ??                STATUS_BYTE     DB      ?
                        ; THE FOLLOWING AREA IS USED ONLY DURING DIAGNOSTICS
                        ; (POST AND ROM RESIDENT)
0001  ??                DCP_MENU_PAGE   DB      ?       ; TO CURRENT PAGE FOR DIAG. MENU
0002  ????              DCP_ROW_COL     DW      ?       ; CURRENT ROW/COLUMN COORDINATES
                                                        ; FOR DIAG MENU
0004  ??                WRAP_FLAG       DB      ?       ; INTERNAL/EXTERNAL 8250 WRAP
                                                        ; INDICATOR
```

```
0005  ??                    MFG_TST         DB      ?       ; INITIALIZATION FLAG
0006  ????                  MEM_TOT         DW      ?       ; WORD EQUIV. TO HIGHEST SEGMENT IN
                                                            ; MEMORY
0008  ????                  MEM_DONES       DW      ?       ; CURRENT SEGMENT VALUE FOR
                                                            ; BACKGROUND MEM TEST
000A  ????                  MEM_DONEO       DW      ?       ; CURRENT OFFSET VALUE FOR
                                                            ; BACKGROUND MEM TEST
000C  ????                  INT1CO          DW      ?       ; SAVE AREA FOR INTERRUPT 1C
                                                            ; ROUTINE
000E  ????                  INT1CS          DW      ?
0010  ??                    MENU_UP         DB      ?       ; FLAG TO INDICATE WHETHER MENU IS
                                                            ; ON SCREEN (FF=YES, 0=NO)
0011  ??                    DONE128         DB      ?       ; COUNTER TO KEEP TRACK OF 128 BYTE
                                                            ; BLOCKS TESTED BY BGMEM
0012  ????                  KBDONE          DW      ?       ; TOTAL K OF MEMORY THAT HAS BEEN
                                                            ; TESTED BY BACKGROUND MEM TEST
                            ;---------------------------------------
                            ;       POST DATA AREA
                            ;---------------------------------------
0014  ????                  IO_ROM_INIT     DW      ?       ; POINTR TO OPTIONAL I/O ROM INIT
                                                            ; ROUTINE
0016  ????                  IO_ROM_SEG      DW      ?       ; POINTER TO IO ROM SEGMENT
0018  ??                    POST_ERR        DB      ?       ; FLAG TO INDICATE ERROR OCCURRED
                                                            ; DURING POST
0019     09 [               MODEM_BUFFER    DB      9 DUP(?) ; MODEM RESPONSE BUFFER
            ??
               ]

                                                            ; (MAX 9 CHARS)
0022  ????                  MFG_RTN         DW      ?       ; POINTER TO MFG. OUTPUT ROUTINE
0024  ????                                  DW      ?
                            ;---------------------------------------
                            ;     SERIAL PRINTER DATA
                            ;---------------------------------------
0026  ????                  SP_FLAG         DW      ?
0028  ??                    SP_CHAR         DB      ?
                                                            ; THE FOLLOWING SIX ENTRIES ARE
                                                            ; DATA PERTAINING TO NEW STICK
0029  ????                  NEW_STICK_DATA  DW      ?       ; RIGHT STICK DELAY
002B  ????                                  DW      ?       ; RIGHT BUTTON A DELAY
002D  ????                                  DW      ?       ; RIGHT BUTTON B DELAY
002F  ????                                  DW      ?       ; LEFT STICK DELAY
0031  ????                                  DW      ?       ; LEFT BUTTON A DELAY
0033  ????                                  DW      ?       ; LEFT BUTTON B DELAY
0035  ????                                  DW      ?       ; RIGHT STICK LOCATION
0037  ????                                  DW      ?       ; UNUSED
0039  ????                                  DW      ?       ; UNUSED
003B  ????                                  DW      ?       ; LEFT STICK POSTITON
003D                        XXDATA  ENDS
                            ;---------------------------------------
                            ;     DISKETTE DATA AREA
                            ;---------------------------------------
0000                        DKDATA  SEGMENT AT 60H
0000  ??                    NUM_DRIVE       DB      ?
0001  ??                    DUAL            DB      ?
0002  ??                    OPERATION       DB      ?
0003  ??                    DRIVE           DB      ?
0004  ??                    TRACK           DB      ?
0005  ??                    HEAD            DB      ?
0006  ??                    SECTOR          DB      ?
0007  ??                    NUM_SECTOR      DB      ?
0008  ??                    SEC             DB      ?
                            ;    FORMAT ID
0009     08 [               TK_HD_SC        DB      8 DUP(0,0,0,0)  ;TRACK,HEAD,SECTOR,NUM OF
            00
            00
            00
            00
               ]

                                                            ; SECTOR
                            ;    BUFFER FOR READ AND WRITE OPERATION
= 0200                      DK_BUF_LEN      EQU     512     ; 512 BYTES/SECTOR
0029  0200 [                READ_BUF        DB      DK_BUF_LEN DUP(0)
            00
               ]

0229  0100 [                WRITE_BUF       DB      (DK_BUF_LEN/2) DUP(6DH,0BH)
            6D
            0B
               ]

                            ;    INFO FLAGS
0429  ??                    REQUEST_IN      DB      ?       ;SELECTION CHARACTER
042A  ??                    DK_EXISTED      DB      ?
042B  ??                    DK_FLAG         DB      ?
042C  ????                  RAN_NUM         DW      ?
042E  ????                  SEED            DW      ?
                            ;    SPEED TEST VARIABLES
0430  ????                  DK_SPEED        DW      ?
0432  ????                  TIM_1           DW      ?
0434  ????                  TIM_L_1         DW      ?
0436  ????                  TIM_2           DW      ?
0438  ????                  TIM_L_2         DW      ?
043A  ????                  FRACT_H         DW      ?
043C  ????                  FRACT_L         DW      ?
043E  ????                  PART_CYCLE      DW      ?
0440  ????                  WHOLE_CYCLE     DW      ?
0442  ????                  HALF_CYCLE      DW      ?
```

```
                        ;       ERROR PARAMETERS
0444  ??                DK_ER_OCCURED   DB      ?       ;ERROR HAS OCCURRED
0445  ??                DK_ER_L1        DB      ?       ;CUSTOMER ERROR LEVEL
0446  ??                DK_ER_L2        DB      ?       ;SERVICE ERROR LEVEL
0447  ??                ER_STATUS_BYTE  DB      ?       ;STATUS BYTE RETURN FROM INT 13H
                        ;       LANGUAGE TABLE
0448  ??                LANG_BYTE       DB      ?       ; PORT 80 TO DETERMINE WHICH
                                                        ; LANGAGE TO USE
0449                    DKDATA  ENDS
                        ;--------------------------------------
                        ;       VIDEO DISPLAY BUFFER
                        ;--------------------------------------
0000                    VIDEO_RAM       SEGMENT AT 0B800H
0000  4000 [                   DB      16384 DUP(?)
              ??
                    ]

4000                    VIDEO_RAM       ENDS
                        ;--------------------------------------
                        ;       ROM RESIDENT CODE               :
                        ;--------------------------------------
0000                    CODE    SEGMENT PAGE
                                ASSUME  CS:CODE,DS:ABS0,ES:NOTHING,SS:STACK

0000  31 35 30 34 30 33         DB      '1504036 COPR. IBM 1981,1983'   ; COPYRIGHT NOTICE
      36 20 43 4F 50 52
      2E 20 49 42 4D 20
      31 39 38 31 2C 31
      39 38 33

0018  0149 R            Z1     DW      L12             ; RETURN POINTERS FOR RTNS CALLED
001D  0157 R                    DW      L14             ; BEFORE STACK INITIALIZED
001F  016D R                    DW      L16
0021  0186 R                    DW      L19
0023  01BA R                    DW      L24
0025  20 4B 42          F3B    DB      ' KB'
0028  0A47 R            EX_O   DW      OFFSET  EB0
002A  0A47 R                    DW      OFFSET  EB0
002C  0ABB R                    DW      OFFSET  TOTLTP0
002E  0AB4 R            EX1    DW      OFFSET  M01
                        ;--------------------------------------
                        ;       MESSAGE AREA FOR POST           :
                        ;--------------------------------------
0030  45 52 52 4F 52    ERROR_ERR       DB      'ERROR' ;GENERAL ERROR PROMPT
0035  41                MEM_ERR         DB      'A'     ;MEMORY ERROR
0036  42                KEY_ERR         DB      'B'     ;KEYBOARD ERROR MSG
0037  43                CASS_ERR        DB      'C'     ;CASSETTE ERROR MESSAGE
0038  44                COM1_ERR        DB      'D'     ;ON-BOARD SERIAL PORT ERR. MSG
0039  45                COM2_ERR        DB      'E'     ;SERIAL PORTION OF MODEM ERROR
003A  46                ROM_ERR         DB      'F'     ;OPTIONAL GENERIC BIOS ROM ERROR
003B  47                CART_ERR        DB      'G'     ;CARTRIDGE ERROR
003C  48                DISK_ERR        DB      'H'     ;DISKETTE ERR
                        ;
003D                    F4      LABEL   WORD            ; PRINTER SOURCE TABLE
003D  0378                      DW      378H
003F  0278                      DW      278H
0041                    F4E     LABEL   WORD
0041                    IMASKS  LABEL       BYTE        ; INTERRUPT MASKS FOR 8259
                                                        ; INTERRUPT CONTROLLER
0041  EF                        DB          0EFH        ; MODEM INTR MASK
0042  F7                        DB          0F7H        ; SERIAL PRINTER INTR MASK
                        ;--------------------------------------
                        ; SETUP                           :
                        ;       DISABLE NMI, MASKABLE INTS.     :
                        ;       SOUND CHIP, AND VIDEO.          :
                        ;       TURN DRIVE 0 MOTOR OFF          :
                        ;--------------------------------------
                                ASSUME  CS:CODE,DS:ABS0,ES:NOTHING,SS:STACK
0043                    RESET   LABEL   FAR
0043  B0 00             START:  MOV     AL,0
0045  E6 A0                     OUT     0A0H,AL         ; DISABLES NMI
0047  FE C8                     DEC     AL              ; SEND FF TO MFG_TESTER
0049  E6 10                     OUT     10H,AL
004B  E4 A0                     IN      AL,0A0H         ; RESET NMI F/F
004D  FA                        CLI                     ; DISABLES MASKABLE INTERRUPTS
                                                        ; DISABLE ATTENUATION IN SOUND CHIP
004E  B8 108F                   MOV     AX,108FH        ; REG ADDRESS IN AH, ATTENUATOR OFF
                                                        ; IN AL
0051  BA 00C0                   MOV     DX,00C0H        ; ADDRESS OF SOUND CHIP
0054  B9 0004                   MOV     CX,4            ; 4 ATTENUATORS TO DISABLE
0057  0A C4             L1:     OR      AL,AH           ; COMBINE REG ADDRESS AND DATA
0059  EE                        OUT     DX,AL
005A  80 C4 20                  ADD     AH,20H          ; POINT TO NEXT REG
005D  E2 F8                     LOOP    L1
005F  B0 A0                     MOV     AL,WD_ENABLE+FDC_RESET  ; TURN DRIVE 0 MOTOR OFF,
                                                        ; ENABLE TIMER
0061  E6 F2                     OUT     0F2H,AL
0063  BA 03DA                   MOV     DX,VGA_CTL      ; VIDEO GATE ARRAY CONTROL
0066  EC                        IN      AL,DX           ; SYNC VGA TO ACCEPT REG
0067  B0 04                     MOV     AL,4            ; SET VGA RESET REG
0069  EE                        OUT     DX,AL           ; SELECT IT
006A  B0 01                     MOV     AL,1            ; SET ASYNC RESET
006C  EE                        OUT     DX,AL           ; RESET VIDEO GATE ARRAY
                        ;--------------------------------------
                        ; TEST 1                          :
                        ;       8088 PROCESSOR TEST             :
                        ; DESCRIPTION                     :
                        ;       VERIFY 8088 FLAGS, REGISTERS    :
                        ;       AND CONDITIONAL JUMPS           :
                        ;                                       :
                        ; MFG. ERROR CODE 0001H           :
                        ;--------------------------------------
```

```
006D  B4 D5                     MOV     AH,0D5H         ; SET SF, CF, ZF, AND AF FLAGS ON
006F  9E                        SAHF
0070  73 4C                     JNC     L4              ; GO TO ERR ROUTINE IF CF NOT SET
0072  75 4A                     JNZ     L4              ; GO TO ERR ROUTINE IF ZF NOT SET
0074  7B 48                     JNP     L4              ; GO TO ERR ROUTINE IF PF NOT SET
0076  79 46                     JNS     L4              ; GO TO ERR ROUTINE IF SF NOT SET
0078  9F                        LAHF                    ; LOAD FLAG IMAGE TO AH
0079  B1 05                     MOV     CL,5            ; LOAD CNT REG WITH SHIFT CNT
007B  D2 EC                     SHR     AH,CL           ; SHIFT AF INTO CARRY BIT POS
007D  73 3F                     JNC     L4              ; GO TO ERR ROUTINE IF AF NOT SET
007F  B0 40                     MOV     AL,40H          ; SET THE OF FLAG ON
0081  D0 E0                     SHL     AL,1            ; SETUP FOR TESTING
0083  71 39                     JNO     L4              ; GO TO ERR ROUTINE IF OF NOT SET
0085  32 E4                     XOR     AH,AH           ; SET AH = 0
0087  9E                        SAHF                    ; CLEAR SF, CF, ZF, AND PF
0088  76 34                     JBE     L4              ; GO TO ERR ROUTINE IF CF ON
                                                        ; GO TO ERR ROUTINE IF ZF ON
008A  78 32                     JS      L4              ; GO TO ERR ROUTINE IF SF ON
008C  7A 30                     JP      L4              ; GO TO ERR ROUTINE IF PF ON
008E  9F                        LAHF                    ; LOAD FLAG IMAGE TO AH
008F  B1 05                     MOV     CL,5            ; LOAD CNT REG WITH SHIFT CNT
0091  D2 EC                     SHR     AH,CL           ; SHIFT 'AF' INTO CARRY BIT POS
0093  72 29                     JC      L4              ; GO TO ERR ROUTINE IF ON
0095  D0 E4                     SHL     AH,1            ; CHECK THAT 'OF' IS CLEAR
0097  70 25                     JO      L4              ; GO TO ERR ROUTINE IF ON
                        ;----- READ/WRITE THE 8088 GENERAL AND SEGMENTATION REGISTERS
                        ;       WITH ALL ONE'S AND ZEROES'S.
0099  B8 FFFF                  MOV     AX,0FFFFH       ; SETUP ONE'S PATTERN IN AX
009C  F9                        STC
009D  8E D8             L2:     MOV     DS,AX           ; WRITE PATTERN TO ALL REGS
009F  8C DB                     MOV     BX,DS
00A1  8E C3                     MOV     ES,BX
00A3  8C C1                     MOV     CX,ES
00A5  8E D1                     MOV     SS,CX
00A7  8C D2                     MOV     DX,SS
00A9  8B E2                     MOV     SP,DX
00AB  8B EC                     MOV     BP,SP
00AD  8B F5                     MOV     SI,BP
00AF  8B FE                     MOV     DI,SI
00B1  73 07                     JNC     L3
00B3  33 C7                     XOR     AX,DI           ; PATTERN MAKE IT THRU ALL REGS
00B5  75 07                     JNZ     L4              ; NO - GO TO ERR ROUTINE
00B7  F8                        CLC
00B8  EB E3                     JMP     L2
00BA  0B C7             L3:     OR      AX,DI           ; ZERO PATTERN MAKE IT THRU?
00BC  74 0C                     JZ      L5              ; YES - GO TO NEXT TEST
00BE  BA 0010           L4:     MOV     DX,0010H        ; HANDLE ERROR
00C1  B0 00                     MOV     AL,0
00C3  EE                        OUT     DX,AL           ; ERROR 0001
00C4  42                        INC     DX
00C5  EE                        OUT     DX,AL
00C6  FE C0                     INC     AL
00C8  EE                        OUT     DX,AL
00C9  F4                        HLT                     ; HALT
00CA                    L5:
                        ;--------------------------------------
                        ; TEST 2                             :
                        ;       8255 INITIALIZATION AND TEST :
                        ; DESCRIPTION                        :
                        ;       FIRST INITIALIZE 8255 PROG.  :
                        ;       PERIPHERAL INTERFACE. PORTS A&B :
                        ;       ARE LATCHED OUTPUT           :
                        ;       BUFFERS. C IS INPUT.         :
                        ; MFG. ERR. CODE =0002H              :
                        ;--------------------------------------
00CA  B0 FE                     MOV     AL,0FEH         ; SEND FE TO MFG
00CC  E6 10                     OUT     10H,AL
00CE  B0 89                     MOV     AL,MODE_8255
00D0  E6 63                     OUT     CMD_PORT,AL     ; CONFIGURES I/O PORTS
00D2  2B C0                     SUB     AX,AX           ; TEST PATTERN SEED = 0000
00D4  8A C4             L6:     MOV     AL,AH
00D6  E6 60                     OUT     PORT_A,AL       ; WRITE PATTERN TO PORT A
00D8  E4 60                     IN      AL,PORT_A       ; READ PATTERN FROM PORT A
00DA  E6 61                     OUT     PORT_B,AL       ; WRITE PATTERN TO PORT B
00DC  E4 61                     IN      AL,PORT_B       ; READ OUTPUT PORT
00DE  3A C4                     CMP     AL,AH           ; DATA AS EXPECTED?
00E0  75 06                     JNE     L7              ; IF NOT, SOMETHING IS WRONG
00E2  FE C4                     INC     AH              ; MAKE NEW DATA PATTERN
00E4  75 EE                     JNZ     L6              ; LOOP TILL 255 PATTERNS DONE
00E6  EB 05                     JMP     SHORT L8        ; CONTINUE IF DONE
00E8  B3 02             L7:     MOV     BL,02H          ; SET ERROR FLAG (BH=00 NOW)
00EA  E9 09BC R                 JMP     E_MSG           ; GO ERROR ROUTINE
00ED  32 C0             L8:     XOR     AL,AL           ;
00EF  E6 60                     OUT     KBPORT,AL       ; CLEAR KB PORT
00F1  E4 62                     IN      AL,PORT_C       ;
00F3  24 08                     AND     AL,00001000B    ; 64K CARD PRESENT?
00F5  B0 1B                     MOV     AL,1BH          ; PORT SETTING FOR 64K SYS
00F7  75 02                     JNZ     L9              ;
00F9  B0 3F                     MOV     AL,3FH          ; PORT SETTING FOR 128K SYS
00FB  BA 03DF           L9:     MOV     DX,PAGREG       ;
00FE  EE                        OUT     DX,AL           ;
00FF  B0 0D                     MOV     AL,00001101B    ;     INITIALIZE OUTPUT PORTS
0101  E6 61                     OUT     PORT_B,AL       ;
```

```
                        ;--------------------------------------------------------
                        ; PART 3
                        ;       SET UP VIDEO GATE ARRAY AND 6845 TO GET MEMORY WORKING
                        ;
                        ;--------------------------------------------------------
0103  B0 FD                     MOV     AL,0FDH
0105  E6 10                     OUT     10H,AL          ;
0107  BA 03D4                   MOV     DX,03D4H        ; SET ADDRESS OF 6845
010A  BB F0A4 R                 MOV     BX,OFFSET VIDEO_PARMS ; POINT TO 6845 PARMS
010D  B9 0010 90                MOV     CX,M0040        ; SET PARM LEN
0111  32 E4                     XOR     AH,AH           ; AH IS REG #
0113  8A C4             L10:    MOV     AL,AH           ; GET 6845 REG #
0115  EE                        OUT     DX,AL
0116  42                       INC     DX              ; POINT TO DATA PORT
0117  FE C4                    INC     AH              ; NEXT REG VALUE
0119  2E: 8A 07                 MOV     AL,CS:[BX]      ; GET TABLE VALUE
011C  EE                        OUT     DX,AL           ; OUT TO CHIP
011D  43                       INC     BX              ; NEXT IN TABLE
011E  4A                       DEC     DX              ; BACK TO POINTER REG
011F  E2 F2                     LOOP    L10
                        ;    START VGA WITHOUT VIDEO ENABLED
0121  BA 03DA                   MOV     DX,VGA_CTL      ; SET ADDRESS OF VGA
0124  EC                        IN      AL,DX           ; BE SURE ADDR/DATA FLAG IS
                                                        ; IN THE PROPER STATE
0125  B9 0005                   MOV     CX,5            ; # OF REGISTERS
0128  32 E4                     XOR     AH,AH           ; AH IS REG COUNTER
012A  8A C4             L11:    MOV     AL,AH           ; GET REG #
012C  EE                        OUT     DX,AL           ; SELECT IT
012D  32 C0                     XOR     AL,AL           ; SET ZERO FOR DATA
012F  EE                        OUT     DX,AL
0130  FE C4                     INC     AH              ; NEXT REG
0132  E2 F6                     LOOP    L11
                        ;-----------------------------------------------
                        ; TEST 4                                      :
                        ;       PLANAR BOARD ROS CHECKSUM TEST        :
                        ; DESCRIPTION                                 :
                        ;       A CHECKSUM TEST IS DONE FOR EACH ROS  :
                        ;       MODULE ON THE PLANAR BOARD TO.        :
                        ;       MFG ERROR CODE =0003H MODULE AT ADDRESS :
                        ;                       F000:0000 ERROR       :
                        ;                       0004H MODULE AT ADDRESS :
                        ;                       F800:0000 ERROR       ;
                        ;-----------------------------------------------
0134  B0 FC                     MOV     AL,0FCH
0136  E6 10                     OUT     10H,AL          ; MFG OUT=FC
                        ; CHECK MODULE AT F000:0 (LENGTH 32K)
0138  33 F6                     XOR     SI,SI           ; INDEX OFFSET WITHIN SEGMENT OF
                                                        ; FIRST BYTE
013A  8C C8                     MOV     AX,CS           ; SET UP STACK SEGMENT
013C  8E D0                     MOV     SS,AX
013E  8E D8                     MOV     DS,AX           ; LOAD DS WITH SEGMENT OF ADDRESS
                                                        ; SPACE OF BIOS/BASIC
0140  B9 8000                   MOV     CX,8000H        ; NUMBER OF BYTES TO BE TESTED, 32K
0143  BC 001B R                 MOV     SP,OFFSET Z1    ; SET UP STACK POINTER SO THAT
                                                        ; RETURN WILL COME HERE
0146  E9 FEEB R                 JMP     ROS_CHECKSUM    ; JUMP TO ROUTINE WHICK PERFORMS
                                                        ; CRC CHECK
0149  74 06             L12:    JZ      L13             ; MODULE AT F000:0 OK, GO CHECK
                                                        ; OTHER MODULE AT F000:8000
014B  BB 0003                   MOV     BX,0003H        ; SET ERROR CODE
014E  E9 09BC R                 JMP     E_MSG           ; INDICATE ERROR
0151  B9 8000            L13:   MOV     CX,8000H        ; LOAD COUNT (SI POINTING TO START
0154  E9 FEEB R                 JMP     ROS_CHECKSUM    ; OF NEXT MODULE AT THIS POINT)
0157  74 06              L14:   JZ      L15             ; PROCEED IF NO ERROR
0159  BB 0004                   MOV     BX,0004H        ; INDICATE ERROR
015C  E9 09BC R                 JMP     E_MSG           ;
015F                     L15:
                        ;-----------------------------------------------
                        ; TEST 5
                        ;       BASE  2K READ/WRITE STORAGE TEST
                        ; DESCRIPTION
                        ;       WRITE/READ/VERIFY DATA PATTERNS
                        ;       AA,55, AND 00 TO 1ST 2K OF STORAGE
                        ;       AND THE 2K JUST BELOW 64K (CRT BUFFER)
                        ;       VERIFY STORAGE ADDRESSABILITY.
                        ;       ON EXIT SET CRT PAGE TO 3. SET
                        ;       TEMPORARY STACK ALSO.
                        ;  MFG. ERROR CODE 04XX FOR SYSTEM BOARD MEM.
                        ;                  05XX FOR 64K ATTRIB. CD. MEM
                        ;                  06XX FOR ERRORS IN BOTH
                        ;                  (XX= ERROR BITS)
                        ;-----------------------------------------------:
015F  B0 FB                     MOV     AL,0FBH
0161  E6 10                     OUT     10H,AL          ; SET MFG FLAG=FB
0163  B9 0400                   MOV     CX,0400H        ; SET FOR 1K WORDS, 2K BYTES
0166  33 C0                     XOR     AX,AX
0168  8E C0                     MOV     ES,AX           ; LOAD ES WITH 0000 SEGMENT
016A  E9 0B59 R                 JMP     PODSTG
016D  75 19              L16:   JNZ     L20             ; BAD STORAGE FOUND
016F  B0 FA                     MOV     AL,0FAH         ; MFG OUT=FA
0171  E6 10                     OUT     10H,AL
0173  B9 0400                   MOV     CX,400H         ; 1024 WORDS TO BE TESTED IN THE
                                                        ; REGEN BUFFER
0176  E4 60                     IN      AL,PORT_A       ; WHERE IS THE REGEN BUFFER?
0178  3C 1B                     CMP     AL,1BH          ; TOP OF 64K?
017A  B8 0F80                   MOV     AX,0F80H        ; SET POINTER TO THERE IF IT IS
017D  74 02                     JE      L18
017F  B4 1F                     MOV     AH,1FH          ; OR SET POINTER TO TOP OF 128K
0181  8E C0              L18:   MOV     ES,AX
0183  E9 0B59 R                 JMP     PODSTG          ;
0186  74 23              L19:   JZ      L23
```

```
0188  B7 04                 L20:    MOV     BH,04H          ; ERROR 04....
018A  E4 62                           IN      AL,PORT_C       ; GET CONFIG BITS
018C  24 08                           AND     AL,00001000B    ; TEST FOR ATTRIB CARD PRESENT
018E  74 06                           JZ      L21             ; WORRY ABOUT ODD/EVEN IF IT IS
0190  8A D9                           MOV     BL,CL
0192  0A DD                           OR      BL,CH           ; COMBINE ERROR BITS IF IT ISN'T
0194  EB 12                           JMP     SHORT L22       ;
0196  80 FC 02              L21:     CMP     AH,02           ; EVEN BYTE ERROR? ERR 04XX
0199  8A D9                           MOV     BL,CL
019B  74 0B                           JE      L22
019D  FE C7                           INC     BH              ; MAKE INTO 05XX ERR
019F  0A DD                           OR      BL,CH           ; MOVE AND POSSIBLY COMBINE
                                                              ; ERROR BITS
01A1  80 FC 01                        CMP     AH,1            ; ODD BYTE ERROR
01A4  74 02                           JE      L22
01A6  FE C7                           INC     BH              ; MUST HAVE BEEN BOTH
                                                              ; - MAKE INTO 06XX
01A8  E9 09BC R             L22:     JMP     E_MSG           ; JUMP TO ERROR OUTPUT ROUTINE
                            ; RETEST HIGH 2K USING B8000 ADDRESS PATH
01AB  B0 F9                 L23:     MOV     AL,0F9H         ; MFG OUT =F9
01AD  E6 10                          OUT     10H,AL
01AF  B9 0400                        MOV     CX,0400H        ; 1K WORDS
01B2  B8 B8B0                        MOV     AX,0B8B0H       ; POINT TO AREA JUST TESTED WITH
                                                             ; DIRECT ADDRESSING
01B5  8E C0                          MOV     ES,AX
01B7  E9 0B59 R                      JMP     PODSTG
01BA  74 06                 L24:     JZ      L25
01BC  B8 0005                        MOV     BX,0005H        ; ERROR 0005
01BF  E9 09BC R                      JMP     E_MSG
                            ;----- SETUP STACK SEG AND SP
01C2  B8 0030               L25:     MOV     AX,0030H        ; GET STACK VALUE
01C5  8E D0                          MOV     SS,AX           ; SET THE STACK UP
01C7  BC 0100 R                      MOV     SP,OFFSET TOS   ; STACK IS READY TO GO
01CA  33 C0                          XOR     AX,AX           ; SET UP DATA SEG
01CC  8E D8                          MOV     DS,AX
                            ;----- SETUP CRT PAGE
01CE  C7 06 0462 R 0007              MOV     DATA_WORD[ACTIVE_PAGE-DATA],07
                            ;----- SET PRELIMINARY MEMORY SIZE WORD
01D4  B8 0040                        MOV     BX,64
01D7  E4 62                          IN      AL,PORT_C       ;
01D9  24 08                          AND     AL,08H          ; 64K CARD PRESENT?
01DB  B0 1B                          MOV     AL,1BH          ; PORT SETTING FOR 64K SYSTEM
01DD  75 05                          JNZ     L26             ; SET TO 64K IF NOT
01DF  83 C3 40                       ADD     BX,64           ; ELSE SET FOR 128K
01E2  B0 3F                          MOV     AL,3FH          ; PORT SETTING FOR 128K SYSTEM
01E4  89 1E 0415 R          L26:     MOV     DATA_WORD[TRUE_MEM-DATA],BX
01E8  A2 048A R                      MOV     DATA_AREA[PAGDAT-DATA],AL
                            ;--------------------------------------------------------
                            ; PART 6
                            ;       INTERRUPTS
                            ; DESCRIPTION
                            ;       32 INTERRUPTS ARE INITIALIZED TO POINT TO A
                            ;       DUMMY HANDLER. THE BIOS INTERRUPTS ARE LOADED.
                            ;       DIAGNOSTIC INTERRUPTS ARE LOADED
                            ;       SYSTEM CONFIGURATION WORD IS PUT IN MEMORY.
                            ;       THE DUMMY INTERRUPT HANDLER RESIDES HERE.
                            ;--------------------------------------------------------
                                    ASSUME  DS:XXDATA
01EB  B8  ---- R                    MOV     AX,XXDATA
01EE  8E D8                         MOV     DS,AX
01F0  C6 06 0005 R F8               MOV     MFG_TST,0F8H    ; SET UP MFG CHECKPOINT FROM THIS
                                                            ; POINT
01F5  E8 E6D8 R                     CALL    MFG_UP          ; UPDATE MFG CHECKPOINT
01F8  C7 06 0022 R 0A61 R           MOV     MFG_RTN,OFFSET MFG_OUT
01FE  8C C8                         MOV     AX,CS
0200  A3 0024 R                     MOV     MFG_RTN+2,AX    ; SET DOUBLEWORD POINTER TO MFG.
                                                            ; ERROR OUTPUT ROUTINE SO DIAGS.
                                                            ; DON'T HAVE TO DUPLICATE CODE
                                    ASSUME  CS:CODE,DS:ABS0
0203  B8 0000                       MOV     AX,0
0206  8E D8                         MOV     DS,AX
                            ;----- SET UP THE INTERRUPT VECTORS TO TEMP INTERRUPT
0208  B9 00FF                       MOV     CX,255          ; FILL ALL INTERRUPTS
020B  2B FF                         SUB     DI,DI           ; FIRST INTERRUPT LOCATION IS 0000
020D  8E C7                         MOV     ES,DI           ; SET ES=0000 ALSO
020F  B8 F815 R             D3:     MOV     AX,OFFSET D11   ; MOVE ADDR OF INTR PROC TO TBL
0212  AB                            STOSW
0213  8C C8                         MOV     AX,CS           ; GET ADDR OF INTR PROC SEG
0215  AB                            STOSW
0216  E2 F7                         LOOP    D3              ; VECTBLO
0218  C7 06 0124 R 109D R           MOV     EXST,OFFSET EXTAB ; SET UP EXT. SCAN TABLE
                            ; SET UP BIOS INTERRUPTS
021E  BF 0040 R                     MOV     DI,OFFSET VIDEO_INT ; SET UP VIDEO INT
0221  0E                            PUSH    CS
0222  1F                            POP     DS              ; PLACE CS IN DS
0223  BE FF03 R                     MOV     SI,OFFSET VECTOR_TABLE+16
0226  B9 0010                       MOV     CX,16
0229  A5                   D4:     MOVSW                   ; MOVE INTERRUPT VECTOR TO LOW
                                                            ; MEMORY
022A  47                            INC     DI
022B  47                            INC     DI              ; POINT TO NEXT VECTOR ENTRY
022C  E2 FB                         LOOP    D4              ; REPEAT FOR ALL 16 BIOS INTERRUPTS
                            ; SET UP DIAGNOSTIC INTERRUPTS
022E  BF 0200                       MOV     DI,0200H        ; START WITH INT. 80H
0231  BE 4000                       MOV     SI,DIAG_TABLE_PTR ; POINT TO ENTRY POINT TABLE
0234  B9 0010                       MOV     CX,16           ; 16 ENTRIES
0237  A5                   D5:     MOVSW                   ; MOVE INTERRUPT VECTOR TO LOW
                                                            ; MEMORY
```

```
0238  47                        INC     DI
0239  47                        INC     DI                ; POINT TO NEXT VECTOR ENTRY
023A  E2 FB                     LOOP    D5                ; REPEAT FOR ALL 16 BIOS INTERRUPTS
023C  8E D9                     MOV     DS,CX             ; SET DS TO ZERO
023E  C7 06 0204 R 1B63 R       MOV     INT81,OFFSET LOCATE1
0244  C7 06 0208 R 1A2A R       MOV     INT82,OFFSET PRNT3
024A  C7 06 0224 R 1BA5 R       MOV     INT89,OFFSET JOYSTICK

                        ;------ SET UP DEFAULT EQUIPMENT DETERMINATION WORD
                        ;       BIT 15,14 = NUMBER OF PRINTERS ATTACHED
                        ;       BIT 13 = 1 = SERIAL PRINTER PRESENT
                        ;       BIT 12 = GAME I/O ATTACHED
                        ;       BIT 11,10,9 = NUMBER OF RS232 CARDS ATTACHED
                        ;       BIT 8 = DMA (0=DMA PRESENT, 1=NO DMA ON SYSTEM
                        ;       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 (10=48K, 11=64K)
                        ;       BIT 1 NOT USED
                        ;       BIT 0 = 1 (IPL DISKETTE INSTALLED)
                        ;--------------------------------------------------
                                ASSUME  CS:CODE,DS:ABS0
0250  BB 1118                   MOV     BX,1118H          ;DEFAULT GAME10,40X25,NO DMA,48K ON
                                                          ; PLANAR
0253  E4 62                     IN      AL,PORT_C
0255  24 08                     AND     AL,08H            ; 64K CARD PRESENT
0257  75 03                     JNZ     D55               ; NO, JUMP
0259  80 CB 04                  OR      BL,4              ; SET 64K ON PLANAR
025C  89 1E 0410 R      D55:    MOV     DATA_WORD[EQUIP_FLAG-DATA],BX
                        ;--------------------------------------------------------
                        ; TEST 7
                        ;       INITIALIZE AND TEST THE 8259 INTERRUPT CONTROLLER CHIP
                        ; MFG ERR. CODE 07XX (XX=00, DATA PATH OR INERNAL FAILURE,
                        ;       XX=ANY OTHER BITS ON=UNEPECTED INTERRUPTS
                        ;--------------------------------------------------------
0260  E8 E6D8 R                 CALL    MFG_UP            ; MFG CODE=F7
                                ASSUME  DS:ABS0,CS:CODE
0263  B0 13                     MOV     AL,13H            ; ICW1 - RESET EDGE SENSE CIRCUIT,
                                                          ;SET SINGLE 8259 CHIP AND ICW4 READ
0265  E6 20                     OUT     INTA00,AL
0267  B0 08                     MOV     AL,8              ; ICW2 - SET INTERRUPT TYPE 8 (8-F)
0269  E6 21                     OUT     INTA01,AL
026B  B0 09                     MOV     AL,9              ; ICW4 - SET BUFFERED MODE/SLAVE
                                                          ;  AND 8086 MODE
026D  E6 21                     OUT     INTA01,AL
                        ;--------------------------------------------------------
                        ;       TEST ABILITY TO WRITE/READ THE MASK REGISTER
                        ;--------------------------------------------------------
026F  B0 00                     MOV     AL,0              ; WRITE ZEROES TO IMR
0271  8A D8                     MOV     BL,AL             ; PRESET ERROR INDICATOR
0273  E6 21                     OUT     INTA01,AL         ; DEVICE INTERRUPTS ENABLED
0275  E4 21                     IN      AL,INTA01         ; READ IMR
0277  0A C0                     OR      AL,AL             ; IMR = 0?
0279  75 18                     JNZ     GERROR            ; NO - GO TO ERROR ROUTINE
027B  B0 FF                     MOV     AL,0FFH           ; DISABLE DEVICE INTERRUPTS
027D  E6 21                     OUT     INTA01,AL         ; WRITE ONES TO IMR
027F  E4 21                     IN      AL,INTA01         ; READ IMR
0281  04 01                     ADD     AL,1              ; ALL IMR BITS ON?
                                                          ; (ADD SHOULD PRODUCE 0)
0283  75 0E                     JNZ     GERROR            ; NO - GO TO ERROR ROUTINE
                        ;--------------------------------------------------------
                        ;       CHECK FOR HOT INTERRUPTS
                        ;--------------------------------------------------------
                        ;       INTERRUPTS ARE MASKED OFF.  NO INTERRUPTS SHOULD OCCUR.
0285  FB                       STI                       ; ENABLE EXTERNAL INTERRUPTS
0286  B9 0050                  MOV     CX,50H
0289  E2 FE            HOT1:   LOOP    HOT1              ; WAIT FOR ANY INTERRUPTS
028B  8A 1E 0484 R              MOV     BL,DATA_AREA[INTR_FLAG-DATA]  ; DID ANY INTERRUPTS
                                                          ; OCCUR?
028F  0A DB                     OR      BL,BL
0291  74 05                     JZ      END_TESTG         ; NO - GO TO NEXT TEST
0293  B7 07             GERROR: MOV     BH,07H            ; SET 07 SECTION OF ERROR MSG
0295  E9 09BC R                 JMP     E_MSG
0298                    END_TESTG:
                        ; FIRE THE DISKETTE WATCHDOG TIMER
0298  B0 E0                     MOV     AL,WD_ENABLE+WD_STROBE+FDC_RESET
029A  E6 F2                     OUT     0F2H,AL
029C  B0 A0                     MOV     AL,WD_ENABLE+FDC_RESET
029E  E6 F2                     OUT     0F2H,AL
                                ASSUME  CS:CODE,DS:ABS0
                        ;--------------------------------------------------------
                        ;       8253 TIMER CHECKOUT
                        ; DESCRIPTION
                        ;       VERIFY THAT THE TIMERS (0, 1, AND 2) FUNCTION PROPERLY.
                        ;       THIS INCLUDES CHECKING FOR STUCK BITS IN ALL THE TIMERS,
                        ;       THAT TIMER 1 RESPONDS TO TIMER 0 OUTPUTS, THAT TIMER 0
                        ;       INTERRUPTS WHEN IT SHOULD, AND THAT TIMER 2'S OUTPUT WORKS
                        ;       AS IT SHOULD.
                        ;       THERE ARE 7 POSSIBLE ERRORS DURING THIS CHECKOUT.
                        ;       BL VALUES FOR THE CALL TO E_MSG INCLUDE:
                        ;       0)  STUCK BITS IN TIMER 0
                        ;       1)  TIMER 1 DOES NOT RESPOND TO TIMER 0 OUTPUT
                        ;       2)  TIMER 0 INTERRUPT DOES NOT OCCUR
                        ;       3)  STUCK BITS IN TIMER 1
                        ;       4)  TIMER 2 OUTPUT INITIAL VALUE IS NOT LOW
                        ;       5)  STUCK BITS IN TIMER 2
                        ;       6)  TIMER 2 OUTPUT DOES NOT GO HIGH ON TERMINAL COUNT
```

```
                        ;--------------------------------------------------------------------------
                        ;       INITIALIZE TIMER 1 AND TIMER 0 FOR TEST
                        ;--------------------------------------------------------------------------
02A0  E8 E6D8 R                 CALL    MFG_UP          ; MFG CKPOINT=F6
02A3  B8 0176                   MOV     AX,0176H        ; SET TIMER 1 TO MODE 3 BINARY
02A6  BB FFFF                   MOV     BX,0FFFFH       ; INITIAL COUNT OF FFFF
02A9  E8 FFE0 R                 CALL    INIT_TIMER      ; INITIALIZE TIMER 1
02AC  B8 0036                   MOV     AX,0036H        ; SET TIMER 0 TO MODE 3 BINARY
                                                        ; INITIAL COUNT OF FFFF
02AF  E8 FFE0 R                 CALL    INIT_TIMER      ; INITIALIZE TIMER 0
                        ;--------------------------------------------------------------------------
                        ;       SET BIT 5 OF PORT A0 SO TIMER 1 CLOCK WILL BE PULSED BY THE
                        ;       TIMER 0 OUTPUT RATHER THAN THE SYSTEM CLOCK.
                        ;--------------------------------------------------------------------------
02B2  B0 20                     MOV     AL,00100000B
02B4  E6 A0                     OUT     0A0H,AL
                        ;--------------------------------------------------------------------------
                        ;       CHECK IF ALL BITS GO ON AND OFF IN TIMER 0 (CHECK FOR STUCK
                        ;          BITS)
                        ;--------------------------------------------------------------------------
02B6  B4 00                     MOV     AH,0            ; TIMER 0
02B8  E8 036C R                 CALL    BITS_ON_OFF     ; LET SUBROUTINE CHECK IT
02BB  73 05                     JNB     TIMER1_NZ       ;NO STUCK BITS (CARRY FLAG NOT SET)
02BD  B3 00                     MOV     BL,0            ; STUCK BITS IN TIMER 0
02BF  E9 0362 R                 JMP     TIMER_ERROR
                        ;--------------------------------------------------------------------------
                        ;       SINCE TIMER 0 HAS COMPLETED AT LEAST ONE COMPLETE CYCLE,
                        ;       TIMER 1 SHOULD BE NON-ZERO.  CHECK THAT THIS IS THE CASE.
                        ;--------------------------------------------------------------------------
02C2                    TIMER1_NZ:
02C2  E4 41                     IN      AL,TIMER+1      ; READ LSB OF TIMER 1
02C4  8A E0                     MOV     AH,AL           ; SAVE LSB
02C6  E4 41                     IN      AL,TIMER+1      ; READ MSB OF TIMER 1
02C8  3D FFFF                   CMP     AX,0FFFFH       ; STILL FFFF?
02CB  75 05                     JNE     TIMER0_INTR     ; NO - TIMER 1 HAS BEEN BUMPED
02CD  B3 01                     MOV     BL,1            ; TIMER 1 WAS NOT BUMPED BY TIMER 0
02CF  E9 0362 R                 JMP     TIMER_ERROR
                        ;--------------------------------------------------------------------------
                        ;       CHECK FOR TIMER 0 INTERRUPT
                        ;--------------------------------------------------------------------------
02D2                    TIMER0_INTR:
02D2  FB                        STI                     ; ENABLE MASKABLE EXT INTERRUPTS
02D3  E4 21                     IN      AL,INTA01
02D5  24 FE                     AND     AL,0FEH         ; MASK ALL INTRS EXCEPT LVL 0
02D7  20 06 0484 R              AND     DATA_AREA[INTR_FLAG-DATA],AL ; CLEAR INT RECEIVED
02DB  E6 21                     OUT     INTA01,AL       ; WRITE THE 8259 IMR
02DD  B9 FFFF                   MOV     CX,0FFFFH       ; SET LOOP COUNT
02E0                    WAIT_INTR_LOOP:
02E0  F6 06 0484 R 01           TEST    DATA_AREA[INTR_FLAG-DATA],1  ; TIMER 0 INT OCCUR?
02E5  75 06                     JNE     RESET_INTRS     ; YES - CONTINUE
02E7  E2 F7                     LOOP    WAIT_INTR_LOOP  ; WAIT FOR INTR FOR SPECIFIED TIME
02E9  B3 02                     MOV     BL,2            ; TIMER 0 INTR DIDN'T OCCUR
02EB  EB 75                     JMP     SHORT TIMER_ERROR
                        ;--------------------------------------------------------------------------
                        ;       HOUSEKEEPING FOR TIMER 0 INTERRUPTS
                        ;--------------------------------------------------------------------------
02ED                    RESET_INTRS:
02ED  FA                        CLI
                        ; SET TIMER INT. TO POINT TO MFG. HEARTBEAT ROUTINE IF IN MFG MODE
02EE  BA 0201                   MOV     DX,201H
02F1  EC                        IN      AL,DX           ; GET MFG. BITS
02F2  24 F0                     AND     AL,0F0H
02F4  3C 10                     CMP     AL,10H          ; SYS TEST MODE?
02F6  74 04                     JE      D6
02F8  0A C0                     OR      AL,AL           ; OR BURN-IN MODE
02FA  75 11                     JNZ     TIME_1
02FC  C7 06 0020 R 188D R   D6: MOV     INT_PTR,OFFSET MFG_TICK ; SET TO POINT TO MFG.
                                                        ; ROUTINE
0302  C7 06 0070 R 188D R       MOV     INT1C_PTR,OFFSET MFG_TICK ; ALSO SET USER TIMER INT
                                                        ; FOR DIAGS. USE
0308  B0 FE                     MOV     AL,0FEH
030A  E6 21                     OUT     INTA01,AL
030C  FB                        STI
                        ;--------------------------------------------------------------------------
                        ;       RESET D5  OF PORT A0 SO THAT THE TIMER 1 CLOCK WILL BE
                        ;       PULSED BY THE SYSTEM CLOCK.
                        ;--------------------------------------------------------------------------
030D  B0 00             TIME_1: MOV     AL,0            ; MAKE AL = 00
030F  E6 A0                     OUT     0A0H,AL
                        ;--------------------------------------------------------------------------
                        ;       CHECK FOR STUCK BITS IN TIMER 1
                        ;--------------------------------------------------------------------------
0311  B4 01                     MOV     AH,1            ; TIMER 1
0313  E8 036C R                 CALL    BITS_ON_OFF
0316  73 04                     JNB     TIMER2_INIT     ; NO STUCK BITS
0318  B3 03                     MOV     BL,3            ; STUCK BITS IN TIMER 1
031A  EB 46                     JMP     SHORT TIMER_ERROR
                        ;--------------------------------------------------------------------------
                        ;       INITIALIZE TIMER 2
                        ;--------------------------------------------------------------------------
031C                    TIMER2_INIT:
031C  B8 02B6                   MOV     AX,02B6H        ; SET TIMER 2 TO MODE 3 BINARY
031F  BB FFFF                   MOV     BX,0FFFFH       ; INITIAL COUNT
0322  E8 FFE0 R                 CALL    INIT_TIMER
                        ;--------------------------------------------------------------------------
                        ;       SET PB0 OF PORT_B OF 8255 (TIMER 2 GATE)
                        ;--------------------------------------------------------------------------
0325  E4 61                     IN      AL,PORT_B       ; CURRENT STATUS
0327  0C 01                     OR      AL,00000001B    ; SET BIT 0 - LEAVE OTHERS ALONE
0329  E6 61                     OUT     PORT_B,AL
```

```
                        ;--------------------------------------------------------------------------
                        ;       CHECK FOR STUCK BITS IN TIMER 2
                        ;--------------------------------------------------------------------------
032B  B4 02                     MOV     AH,2            ; TIMER 2
032D  E8 036C R                 CALL    BITS_ON_OFF
0330  73 04                     JNB     REINIT_T2       ; NO STUCK BITS
0332  B3 05                     MOV     BL,5            ; STUCK BITS IN TIMER 2
0334  EB 2C                     JMP     SHORT TIMER_ERROR
                        ;--------------------------------------------------------------------------
                        ;       RE_INITIALIZE TIMER 2 WITH MODE 0 AND A SHORT COUNT
                        ;--------------------------------------------------------------------------
0336                    REINIT_T2:
                        ; DROP GATE TO TIMER 2
0336  E4 61                     IN      AL,PORT_B       ; CURRENT STATUS
0338  24 FE                     AND     AL,11111110B    ; RESET BIT 0 - LEAVE OTHERS ALONE
033A  E6 61                     OUT     PORT_B,AL
033C  B8 02B0                   MOV     AX,02B0H        ; SET TIMER 2 TO MODE 0 BINARY
033F  BB 000A                   MOV     BX,000AH        ; INITIAL COUNT OF 10
0342  E8 FFE0 R                 CALL    INIT_TIMER
                        ;--------------------------------------------------------------------------
                        ;       CHECK PC5 OF PORT_C OF 8255 TO SEE IF THE OUTPUT OF TIMER 2
                        ;       IS LOW
                        ;--------------------------------------------------------------------------
0345  E4 62                     IN      AL,PORT_C       ; CURRENT STATUS
0347  24 20                     AND     AL,00100000B    ; MASK OFF OTHER BITS
0349  74 04                     JZ      CK2_ON          ; IT'S LOW
034B  B3 04                     MOV     BL,4            ; PC5 OF PORT_C WAS HIGH WHEN IT
034D  EB 13                     JMP     SHORT TIMER_ERROR ; SHOULD HAVE BEEN LOW
                        ; TURN GATE BACK ON
034F  E4 61             CK2_ON: IN      AL,PORT_B       ; CURRENT STATUS
0351  0C 01                     OR      AL,00000001B    ; SET BIT 0 - LEAVE OTHERS ALONE
0353  E6 61                     OUT     PORT_B,AL
                        ;--------------------------------------------------------------------------
                        ;       CHECK PC5 OF PORT_C TO SEE IF THE OUTPUT OF TIMER 2 GOES
                        ;       HIGH
                        ;--------------------------------------------------------------------------
0355  B9 000A                   MOV     CX,000AH        ; WAIT FOR OUTPUT GO HIGH, SHOULD
0358  E2 FE             CK2_LO: LOOP    CK2_LO          ; BE LONGER THAN INITIAL COUNT
035A  E4 62                     IN      AL,PORT_C       ; CURRENT STATUS
035C  24 20                     AND     AL,00100000B    ; MASK OFF ALL OTHER BITS
035E  75 57                     JNZ     POD13_END       ; IT'S HIGH - WE'RE DONE!
0360  B3 06                     MOV     BL,6            ; TIMER 2 OUTPUT DID NOT GO HIGH
                        ;--------------------------------------------------------------------------
                        ;       8253 TIMER ERROR OCCURRED.  SET BH WITH MAJOR ERROR
                        ;       INDICATOR AND CALL E_MSG TO INFORM THE SYSTEM OF THE ERROR.
                        ;       (BL ALREADY CONTAINS THE MINOR ERROR INDICATOR TO TELL
                        ;       WHICH PART OF THE TEST FAILED.)
                        ;--------------------------------------------------------------------------
0362                    TIMER_ERROR:
0362  B7 08                     MOV     BH,8            ; TIMER ERROR INDICATOR
0364  E8 09BC R                 CALL    E_MSG
0367  EB 4E                     JMP     SHORT POD13_END
                        ;--------------------------------------------------------------------------
                        ;       BITS ON/OFF SUBROUTINE - USED FOR DETERMINING IF A
                        ;       PARTICULAR TIMER'S BITS GO ON AND OFF AS THEY SHOULD.
                        ;       THIS ROUTINE ASSUMES THAT THE TIMER IS USING BOTH THE LSB
                        ;       AND THE MSB.
                        ;   CALLING PARAMETER:
                        ;       (AH) = TIMER NUMBER (0, 1, OR 2)
                        ;   RETURNS:
                        ;       (CF) = 1 IF FAILED
                        ;       (CF) = 0 IF PASSED
                        ;       REGISTERS AX, BX, CX, DX, DI, AND SI ARE ALTERED.
                        ;--------------------------------------------------------------------------
0369                    LATCHES LABEL   BYTE
0369  00                       DB      00H             ; LATCH MASK FOR TIMER 0
036A  40                       DB      40H             ; LATCH MASK FOR TIMER 1
036B  80                       DB      80H             ; LATCH MASK FOR TIMER 2
                        ;
036C                    BITS_ON_OFF     PROC    NEAR
036C  33 DB                     XOR     BX,BX           ; INITIALIZE BX REGISTER
036E  33 F6                     XOR     SI,SI           ; 1ST PASS - SI = 0
0370  BA 0040                   MOV     DX,TIMER        ; BASE PORT ADDRESS FOR TIMERS
0373  02 D4                     ADD     DL,AH
0375  BF 0369 R                 MOV     DI,OFFSET LATCHES ; SELECT LATCH MASK
0378  32 C0                     XOR     AL,AL           ; CLEAR AL
037A  86 C4                     XCHG    AL,AH           ; AH -> AL
037C  03 F8                     ADD     DI,AX           ; TIMER LATCH MASK INDEX
                        ; 1ST PASS - CHECKS FOR ALL BITS TO COME ON
                        ; 2ND PASS - CHECKS FOR ALL BITS TO GO OFF
037E                    OUTER_LOOP:
037E  B9 0008                   MOV     CX,8            ; OUTER LOOP COUNTER
0381                    INNER_LOOP:
0381  51                        PUSH    CX              ; SAVE OUTER LOOP COUNTER
0382  B9 FFFF                   MOV     CX,0FFFFH       ; INNER LOOP COUNTER
0385                    TST_BITS:
0385  2E: 8A 05                 MOV     AL,CS:[DI]      ; TIMER LATCH MASK
0388  E6 43                     OUT     TIM_CTL,AL      ; LATCH TIMER
038A  50                        PUSH    AX              ; PAUSE
038B  58                        POP     AX
038C  EC                        IN      AL,DX           ; READ TIMER LSB
038D  0B F6                     OR      SI,SI
038F  75 0D                     JNE     SECOND          ; SECOND PASS
0391  0C 01                     OR      AL,01H          ; TURN LS BIT ON
0393  0A D8                     OR      BL,AL           ; TURN 'ON' BITS ON
0395  EC                        IN      AL,DX           ; READ TIMER MSB
0396  0A F8                     OR      BH,AL           ; TURN 'ON' BITS ON
0398  81 FB FFFF                CMP     BX,0FFFFH       ; ARE ALL TIMER BITS ON?
039C  EB 07                     JMP     SHORT TST_CMP   ; DON'T CHANGE FLAGS
```

```
039E                    SECOND:
039E  22 DB                     AND     BL,AL           ; CHECK FOR ALL BITS OFF
03A0  EC                        IN      AL,DX           ; READ MSB
03A1  22 F8                     AND     BH,AL           ; TURN OFF BITS
03A3  0B DB                     OR      BX,BX           ; ALL OFF?
03A5                    TST_CMP:
03A5  74 07                     JE      CHK_END         ; YES - SEE IF DONE
03A7  E2 DC                     LOOP    TST_BITS        ; KEEP TRYING
03A9  59                        POP     CX              ; RESTORE OUTER LOOP COUNTER
03AA  E2 D5                     LOOP    INNER_LOOP      ; TRY AGAIN
03AC  F9                        STC                     ; ALL TRIES EXHAUSTED - FAILED TEST
03AD  C3                        RET
03AE                    CHK_END:
03AE  59                        POP     CX              ; POP FORMER OUTER LOOP COUNTER
03AF  46                        INC     SI
03B0  83 FE 02                  CMP     SI,2
03B3  75 C9                     JNE     OUTER_LOOP      ; CHECK FOR ALL BITS TO GO OFF
03B5  F8                        CLC                     ; TIMER BITS ARE WORKING PROPERLY
03B6  C3                        RET
03B7                    BITS_ON_OFF     ENDP
03B7                    POD13_END:
                        ;--------------------------------------------------------
                        ;               CRT ATTACHMENT TEST
                        ;
                        ; 1. INIT CRT TO 40X25 - BW
                        ; 2. CHECK FOR VERTICAL AND VIDEO ENABLES, AND CHECK
                        ;    TIMING OF SAME
                        ; 3. CHECK VERTICAL INTERRUPT
                        ; 4. CHECK RED, BLUE, GREEN, AND INTENSIFY DOTS
                        ; 5. INIT TO 40X25 - COLOR
                        ;   MFG. ERROR CODE 09XX (XX-SEE COMMENETS IN CODE)
                        ;--------------------------------------------------------
= A0AC                  MAVT    EQU     0A0ACH          ; MAXIMUM TIME FOR VERT/VERT
                                                        ; (NOMINAL + 10%)
= C460                  MIVT    EQU     0C460H          ; MINIMUM TIME FOR VERT/VERT
                                                        ; (NOMINAL - 10%)
                        ;  NOMINAL TIME IS B286H FOR 60 hz.
= 00C8                  EPF     EQU     200             ; NUMBER OF ENABLES PER FRAME

03B7  E8 E6D8 R                 CALL    MFG_UP          ; MFG CHECKPOINT= F5
03BA  FA                        CLI
03BB  B0 70                     MOV     AL,01110000B    ; SET TIMER 1 TO MODE 0
03BD  E6 43                     OUT     TIM_CTL,AL
03BF  B9 8000                   MOV     CX,8000H
03C2  E2 FE             Q1:     LOOP    Q1              ; WAIT FOR MODE SET TO "TAKE"
03C4  B0 00                     MOV     AL,00H
03C6  E6 41                     OUT     TIMER+1,AL      ; SEND FIRST BYTE TO TIMER
03C8  2B C0                     SUB     AX,AX           ; SET MODE 40X25 - BW
03CA  CD 10                     INT     10H
03CC  B8 0507                   MOV     AX,0507H        ; SET TO VIDEO PAGE 7
03CF  CD 10                     INT     10H             ;
03D1  BA 03DA                   MOV     DX,03DAH        ; SET ADDRESSING TO VIDEO ARRAY
03D4  2B C9                     SUB     CX,CX           ;
                        ; LOOK FOR VERTICAL
03D6  EC                Q2:     IN      AL,DX           ; GET STATUS
03D7  A8 08                     TEST    AL,00001000B    ; VERTICAL THERE YET?
03D9  75 06                     JNE      Q3             ; CONTINUE IF IT IS
03DB  E2 F9                     LOOP    Q2              ; KEEP LOOKING TILL COUNT EXHAUSTED
03DD  B3 00                     MOV     BL,00           ;
03DF  EB 4C                     JMP     SHORT Q115      ; NO VERTICAL = ERROR 0900
                        ; GOT VERTICAL - START TIMER
03E1  32 C0             Q3:     XOR     AL,AL           ;
03E3  E6 41                     OUT     TIMER+1,AL      ; SEND 2ND BYTE TO TIMER TO START
03E5  2B DB                     SUB     BX,BX           ; INIT. ENABLE COUNTER
                        ; WAIT FOR VERTICAL TO GO AWAY
03E7  33 C9                     XOR     CX,CX
03E9  EC                Q4:     IN      AL,DX           ; GET STATUS
03EA  A8 08                     TEST    AL,00001000B    ; VERTICAL STILL THERE?
03EC  74 06                     JZ      Q5              ; CONTINUE IF IT'S GONE
03EE  E2 F9                     LOOP    Q4              ; KEEP LOOKING TILL COUNT EXHAUSTED
03F0  B3 01                     MOV     BL,01H          ;
03F2  EB 39                     JMP     SHORT Q115      ; VERTICAL STUCK ON = ERROR 0901
                        ; NOW START LOOKING FOR ENABLE TRANSITIONS
03F4  2B C9             Q5:     SUB     CX,CX
03F6  EC                Q6:     IN      AL,DX           ; GET STATUS
03F7  A8 01                     TEST    AL,00000001B    ; ENABLE ON YET?
03F9  75 0A                     JNE     Q7              ; GO ON IF IT IS
03FB  A8 08                     TEST    AL,00001000B    ; VERTICAL ON AGAIN?
03FD  75 22                     JNE     Q11             ; CONTINUE IF IT IS
03FF  E2 F5                     LOOP    Q6              ; KEEP LOOKING IF NOT
0401  B3 02                     MOV     BL,02H          ;
0403  EB 28                     JMP     SHORT Q115      ; ENABLE STUCK OFF = ERROR 0902
                        ; MAKE SURE VERTICAL WENT OFF WITH ENABLE GOING ON
0405  A8 08             Q7:    TEST    AL,00001000B    ; VERTICAL OFF?
0407  74 04                     JZ      Q8              ; GO ON IF IT IS
0409  B3 03                     MOV     BL,03H          ;
040B  EB 20                     JMP     SHORT Q115      ; VERTICAL STUCK ON = ERROR 0903
                        ; NOW WAIT FOR ENABLE TO GO OFF
040D  2B C9             Q8:     SUB     CX,CX
040F  EC                Q9:     IN      AL,DX           ; GET STATUS
0410  A8 01                     TEST    AL,00000001B    ; ENABLE OFF YET?
0412  74 06                     JE      Q10             ; PROCEED IF IT IS
0414  E2 F9                     LOOP    Q9              ; KEEP LOOKING IF NOT YET LOW
0416  B3 04                     MOV     BL,04H          ;
0418  EB 13                     JMP     SHORT Q115      ; ENABLE STUCK ON = ERROR 0904
                        ; ENABLE HAS TOGGLED, BUMP COUNTER AND TEST FOR NEXT VERTICAL
041A  43                Q10:    INC     BX              ; BUMP ENABLE COUNTER
041B  74 04                     JZ      Q11             ; IF COUNTER WRAPS, ERROR
041D  A8 08                     TEST    AL,00001000B    ; DID ENABLE GO LOW BECAUSE OF
                                                        ; VERTICAL?
041F  74 D3                     JZ      Q5              ; IF NOT, LOOK FOR ANOTHER ENABLE
                                                        ;   TOGGLE
```

```
                                ; HAVE HAD COMPLETE VERTICAL-VERTICAL CYCLE, NOW TEST RESULTS
0421  B0 40                     Q11:    MOV     AL,40H          ; LATCH TIMER1
0423  E6 43                             OUT     TIM_CTL,AL      ;
0425  81 FB 00C8                        CMP     BX,EPF          ; NUMBER OF ENABLES BETWEEN
                                                                ; VERTICALS O.K.?
0429  74 04                             JE      Q12             ;
042B  B3 05                             MOV     BL,05H          ;
042D  EB 74                     Q115:   JMP     SHORT Q22       ; WRONG # ENABLES = ERROR 0905
042F  E4 41                     Q12:    IN      AL,TIMER+1      ; GET TIMER VALUE LOW
0431  8A E0                             MOV     AH,AL           ; SAVE IT
0433  90                                NOP                     ;
0434  E4 41                             IN      AL,TIMER+1      ; GET TIMER HIGH
0436  86 E0                             XCHG    AH,AL           ;
0438  FB                                STI                     ; INTERRUPTS BACK ON
0439  90                                NOP                     ;
043A  3D A0AC                           CMP     AX,MAVT         ;
043D  7D 04                             JGE     Q13             ;
043F  B3 06                             MOV     BL,06H          ;
0441  EB 60                             JMP     SHORT Q22       ; VERTICALS TOO FAR APART
                                                                ; = ERROR 0906
0443  3D C460                   Q13:    CMP     AX,MIVT         ;
0446  7E 04                             JLE     Q14             ;
0448  B3 07                             MOV     BL,07H          ;
044A  EB 57                             JMP     SHORT Q22       ; VERTICALS TOO CLOSE TOGETHER
                                                                ;   = ERROR 0907
                                ; TIMINGS SEEM O.K., NOW CHECK VERTICAL INTERRUPT (LEVEL 5)
044C  2B C9                     Q14:    SUB     CX,CX           ; SET TIMEOUT REG
044E  E4 21                             IN      AL,INTA01       ;
0450  24 DF                             AND     AL,11011111B    ; UNMASK INT. LEVEL 5
0452  E6 21                             OUT     INTA01,AL       ;
0454  20 06 0484 R                      AND     DATA_AREA[INTR_FLAG-DATA],AL
0458  FB                                STI                     ; ENABLE INTS.
0459  F6 06 0484 R 20           Q15:    TEST    DATA_AREA[INTR_FLAG-DATA],00100000B ; SEE IF INTR.
                                                                ;     5 HAPPENED YET
045E  75 06                             JNZ     Q16             ; GO ON IF IT DID
0460  E2 F7                             LOOP    Q15             ; KEEP LOOKING IF IT DIDN'T
0462  B3 08                             MOV     BL,08H          ;
0464  EB 3D                             JMP     SHORT Q22       ; NO VERTICAL INTERRUPT
                                                                ; = ERROR 0908
0466  E4 21                     Q16:    IN      AL,INTA01       ; DISABLE INTERRUPTS FOR LEVEL 5
0468  0C 20                             OR      AL,00100000B    ;
046A  E6 21                             OUT     INTA01,AL       ;
                                ; SEE IF RED, GREEN, BLUE AND INTENSIFY DOTS WORK
                                ; FIRST, SET A LINE OF REVERSE VIDEO, INTENSIFIED BLANKS INTO VIDEO
                                ; BUFFER
046C  B8 09DB                           MOV     AX,09DBH        ; WRITE CHARS, BLOCKS
046F  BB 077F                           MOV     BX,077FH        ; PAGE 7, REVERSE VIDEO,
                                                                ;    HIGH INTENSITY
0472  B9 0028                           MOV     CX,40           ; 40 CHARACTERS
0475  CD 10                             INT     10H             ;
0477  33 C0                             XOR     AX,AX           ; START WITH BLUE DOTS
0479  2B C9                     Q17:    SUB     CX,CX           ;
047B  EE                                OUT     DX,AL           ; SET VIDEO ARRAY ADDRESS FOR DOTS
                                ; SEE IF DOT COMES ON
047C  EC                        Q18:    IN      AL,DX           ; GET STATUS
047D  A8 10                             TEST    AL,00010000B    ; DOT THERE?
047F  75 08                             JNZ     Q19             ; GO LOOK FOR DOT TO TURN OFF
0481  E2 F9                             LOOP    Q18             ; CONTINUE TESTING FOR DOT ON
0483  B3 10                             MOV     BL,10H          ;
0485  0A DC                             OR      BL,AH           ; OR IN DOT BEING TESTED
0487  EB 1A                             JMP     SHORT Q22       ; DOT NOT COMING ON = ERROR 091X
                                                                ; ( X=0, BLUE; X=1, GREEN;
                                                                ; X=2, RED; X=3, INTENSITY)
                                ; SEE IF DOT GOES OFF
0489  2B C9                     Q19:    SUB     CX,CX
048B  EC                        Q20:    IN      AL,DX           ; GET STATUS
048C  A8 10                             TEST    AL,00010000B    ; IS DOT STILL ON?
048E  74 08                             JE      Q21             ; GO ON IF DOT OFF
0490  E2 F9                             LOOP    Q20             ; ELSE, KEEP WAITING FOR DOT
                                                                ; TO GO OFF
0492  B3 20                             MOV     BL,20H          ;
0494  0A DC                             OR      BL,AH           ; OR IN DOT BEING TESTED
0496  EB 0B                             JMP     SHORT Q22       ; DOT STUCK ON = ERROR 092X
                                                                ; (X=0, BLUE; X=1, GREEN;
                                                                ;  X=2, RED; X=3, INTENSITY)
                                ; ADJUST TO POINT TO NEXT DOT
0498  FE C4                     Q21:    INC     AH              ;
049A  80 FC 04                          CMP     AH,4            ; ALL 4 DOTS DONE?
049D  74 09                             JE      Q23             ; GO END
049F  8A C4                             MOV     AL,AH           ;
04A1  EB D6                             JMP     Q17             ; GO LOOK FOR ANOTHER DOT
04A3  B7 09                     Q22:    MOV     BH,09H          ; SET MSB OF ERROR CODE
04A5  E9 09BC R                         JMP     E_MSG           ;
                                ; DONE WITH TEST RESET TO 40X25 - COLOR
                                        ASSUME  DS:DATA
04A8  E8 138B R                 Q23:    CALL    DDS
04AB  B8 0001                           MOV     AX,0001H        ; INIT TO 40X25 - COLOR
04AE  CD 10                             INT     10H             ;
04B0  B8 0507                           MOV     AX,0507H        ; SET TO VIDEO PAGE 7
04B3  CD 10                             INT     10H             ;
04B5  81 3E 0072 R 1234                 CMP     RESET_FLAG,1234H ; WARM START?
04BB  74 03                             JE      Q24             ; BYPASS PUTTING UP POWER-ON SCREEN
04BD  E8 0C21 R                         CALL    PUT_LOGO        ; PUT LOGO ON SCREEN
```

```
04BD  E8 0C21 R                 CALL    PUT_LOGO          ; PUT LOGO ON SCREEN
04C0  B0 76             Q24:    MOV     AL,01110110B      ; RE-INIT TIMER 1
04C2  E6 43                     OUT     TIM_CTL,AL        ;
04C4  B0 00                     MOV     AL,00H
04C6  E6 41                     OUT     TIMER+1,AL
04C8  90                        NOP
04C9  90                        NOP
04CA  E6 41                     OUT     TIMER+1,AL
                                ASSUME  DS:ABS0
04CC  E8 E6D8 R                 CALL    MFG_UP            ; MFG CHECKPOINT=F4
04CF  33 C0                     XOR     AX,AX
04D1  8E D8                     MOV     DS,AX
04D3  C7 06 0008 R 0F78 R       MOV     NMI_PTR,OFFSET KBDNMI ; SET INTERRUPT VECTOR
04D9  C7 06 0120 R F068 R       MOV     KEY62_PTR,OFFSET KEY_SCAN_SAVE  ; SET VECTOR FOR
                                                          ; POD INT HANDLER
04DF  0E                        PUSH    CS
04E0  58                        POP     AX
04E1  A3 0122 R                 MOV     KEY62_PTR+2,AX
                                ASSUME  DS:DATA
04E4  E8 138B R                 CALL    DDS               ; SET DATA SEGMENT
04E7  BE 001E R                 MOV     SI,OFFSET KB_BUFFER ; SET KEYBOARD PARMS
04EA  89 36 001A R              MOV     BUFFER_HEAD,SI
04EE  89 36 001C R              MOV     BUFFER_TAIL,SI
04F2  89 36 0080 R              MOV     BUFFER_START,SI
04F6  83 C6 20                  ADD     SI,32             ; SET DEFAULT BUFFER OF 32 BYTES
04F9  89 36 0082 R              MOV     BUFFER_END,SI
04FD  E4 A0                     IN      AL,0A0H           ; CLEAR NMI F/F
04FF  B0 80                     MOV     AL,80H            ; ENABLE NMI
0501  E6 A0                     OUT     0A0H,AL           ;
                        ;  IF A KEY IS STUCK, THE BUFFER SHOULD FILL WITH THAT KEY'S CODE
                        ;  THIS WILL BE CHECKED LATER
                        ;--------------------------------------------------
                        ;
                        ;          MEMORY SIZE DETERMINE AND TEST
                        ;  THIS ROUTINE WILL DETERMINE HOW MUCH MEM
                        ;  IS ATTACHED TO THE SYSTEM (UP TO 640KB)
                        ;  AND SET "MEMORY_SIZE" AND "REAL_MEMORY"
                        ;  WORDS IN THE DATA AREA.
                        ;
                        ;  AFTER THIS, MEMORY WILL BE EITHER TESTED
                        ;  OR CLEARED, DEPENDING ON THE CONTENTS OF
                        ;  "RESET_FLAG".
                        ;   MFG. ERROR CODES    -0AXX PLANAR BD ERROR
                        ;                       -0BXX 64K CD ERROR
                        ;                       -0CXX ERRORS IN BOTH
                        ;                        ODD AND EVEN BYTES
                        ;                        IN A 128K SYS
                        ;                       -1YXX MEMORY ABOVE 128K
                        ;                        Y=SEGMENT HAVING TROUBLE
                        ;                        XX= ERROR BITS
                        ;--------------------------------------------------
                                ASSUME  DS:DATA
0503  E8 E6D8 R                 CALL    MFG_UP            ; MFG CHECKPOINT=F3
0506  BB 0040                   MOV     BX,64             ; START WITH BASE 64K
0509  E4 62                     IN      AL,PORT_C         ; GET CONFIG BYTE
050B  A8 08                     TEST    AL,00001000B      ; SEE IF 64K CARD INSTALLED
050D  75 03                     JNE     Q25               ; (BIT 4 WILL BE 0 IF CARD PLUGGED)
050F  83 C3 40                  ADD     BX,64             ; ADD 64K
0512  53                Q25:    PUSH    BX                ; SAVE K COUNT
0513  83 EB 10                  SUB     BX,16             ; SUBTRACT 16K CRT REFRESH SPACE
0516  89 1E 0013 R              MOV     [MEMORY_SIZE],BX  ; LOAD "CONTIGUOUS MEMORY" WORD
051A  5B                        POP     BX
051B  BA 2000                   MOV     DX,2000H          ; SET POINTER TO JUST ABOVE 128K
051E  2B FF                     SUB     DI,DI             ; SET DI TO POINT TO BEGINNING
0520  B9 AA55                   MOV     CX,0AA55H         ; LOAD DATA PATTERN
0523  8E C2             Q26:    MOV     ES,DX             ; SET SEGMENT TO POINT TO MEMORY
                                                          ; SPACE
0525  26: 89 0D                 MOV     ES:[DI],CX        ; SET DATA PATTERN TO MEMORY
0528  B0 0F                     MOV     AL,0FH            ; SET AL TO ODD VALUE
052A  26: 8B 05                 MOV     AX,ES:[DI]        ; GET DATA PATTERN BACK FROM MEM
052D  33 C1                     XOR     AX,CX             ; SEE IF DATA MADE IT BACK
052F  75 0C                     JNZ     Q27               ; NO? THEN END OF MEM HAS BEEN
                                                          ; REACHED
0531  81 C2 1000                ADD     DX,1000H          ; POINT TO BEGINNING OF NEXT 64K
0535  83 C3 40                  ADD     BX,64             ; ADJUST TOTAL MEM. COUNTER
0538  80 FE A0                  CMP     DH,0A0H           ; PAST 640K YET?
053B  75 E6                     JNE     Q26               ; CHECK FOR ANOTHER BLOCK IF NOT
053D  89 1E 0015 R      Q27:    MOV     [TRUE_MEM],BX     ; LOAD "TOTAL MEMORY" WORD
                        ; SIZE HAS BEEN DETERMINED, NOW TEST OR CLEAR ALL OF MEMORY
0541  B8 0004                   MOV     AX,4              ; 4 KB KNOWN OK AT THIS POINT
0544  E8 058C R                 CALL    Q35
0547  BA 0080                   MOV     DX,0080H          ; SET POINTER TO JUST ABOVE
                                                          ; LOWER 2K
054A  B9 7800                   MOV     CX,7800H          ; TEST 30K WORDS (60KB)
054D  8E C2             Q28:    MOV     ES,DX
054F  51                        PUSH    CX                ;
0550  53                        PUSH    BX
0551  50                        PUSH    AX
0552  E8 0B59 R                 CALL    PODSTG            ; TEST OR FILL MEM
0555  74 03                     JZ      Q29
0557  E9 0603 R                 JMP     Q39               ; JUMP IF ERROR
055A  58                Q29:    POP     AX
055B  58                        POP     BX
055C  59                        POP     CX                ; RECOVER
055D  80 FD 78                  CMP     CH,78H            ; WAS THIS A 60 K PASS
0560  9C                        PUSHF
0561  05 003C                   ADD     AX,60             ; BUMP GOOD STORAGE BY 60 KB
0564  9D                        POPF
0565  74 03                     JE      Q30
0567  05 0002                   ADD     AX,2              ;ADD 2 FOR A 62K PASS
056A  E8 058C R         Q30:    CALL    Q35
056D  3B C3                     CMP     AX,BX             ; ARE WE DONE YET?
056F  75 03                     JNE     Q31
0571  E9 0640 R                 JMP     Q43               ; ALL DONE, IF SO
```

```
0574  3D 0080                Q31:    CMP     AX,128          ; DONE WITH 1ST 128K?
0577  74 1E                          JE      Q32             ; GO FINISH REST OF MEM.
0579  BA 0F80                        MOV     DX,0F80H        ; SET POINTER TO FINISH 1ST 64 KB
057C  B9 0400                        MOV     CX,0400H
057F  8E C2                          MOV     ES,DX
0581  50                             PUSH    AX
0582  53                             PUSH    BX
0583  52                             PUSH    DX
0584  E8 0859 R                      CALL    PODSTG          ; GO TEST/FILL
0587  75 7A                          JNZ     Q39             ;
0589  5A                             POP     DX
058A  5B                             POP     BX
058B  58                             POP     AX
058C  05 0002                        ADD     AX,2            ; UPDATE GOOD COUNT
058F  BA 1000                        MOV     DX,1000H        ; SET POINTER TO 2ND 64K BLOCK
0592  B9 7C00                        MOV     CX,7C00H        ; 62K WORTH
0595  EB B6                          JMP     Q28             ; GO TEST IT
0597  BA 2000                Q32:    MOV     DX,2000H        ; POINT TO BLOCK ABOVE 128K
059A  3B D8                   Q33:    CMP     BX,AX           ; COMPARE GOOD MEM TO TOTAL MEM
059C  75 03                           JNE     Q34
059E  E9 0640 R                       JMP     Q43             ; EXIT IF ALL DONE
05A1  B9 4000                 Q34:    MOV     CX,4000H        ; SET FOR 32KB BLOCK
05A4  8E C2                           MOV     ES,DX
05A6  50                              PUSH    AX
05A7  53                              PUSH    BX
05A8  52                              PUSH    DX
05A9  E8 0859 R                       CALL    PODSTG          ; GO TEST/FILL
05AC  75 55                           JNZ     Q39             ;
05AE  5A                              POP     DX
05AF  5B                              POP     BX
05B0  58                              POP     AX
05B1  05 0020                         ADD     AX,32           ; BUMP GOOD MEMORY COUNT
05B4  E8 05BC R                       CALL    Q35             ; DISPLAY CURRENT GOOD MEM
05B7  80 C6 08                        ADD     DH,08H          ; SET POINTER TO NEXT 32K
05BA  EB DE                           JMP     Q33             ; AND MAKE ANOTHER PASS
                              ;---------------------------------------
                              ;    SUBROUTINE FOR PRINTING TESTED
                              ;    MEMORY OK MSG ON THE CRT
                              ; CALL PARMS: AX = K OF GOOD MEMORY
                              ;              (IN HEX)
                              ;---------------------------------------
05BC                          Q35     PROC    NEAR
05BC  E8 138B R                       CALL    DDS             ; ESTABLISH ADDRESSING
05BF  81 3E 0072 R 1234               CMP     RESET_FLAG,1234H ; WARM START?
05C5  74 3B                           JE      Q35E            ; NO PRINT ON WARM START
05C7  53                              PUSH    BX
05C8  51                              PUSH    CX
05C9  52                              PUSH    DX
05CA  50                              PUSH    AX              ; SAVE WORK REGS
05CB  B4 02                           MOV     AH,2            ; SET CURSOR TOWARD THE END OF
05CD  BA 1421                         MOV     DX,1421H        ; ROW 20 (ROW 20, COL. 33)
05D0  B7 07                           MOV     BH,7            ; PAGE 7
05D2  CD 10                           INT     10H
05D4  58                              POP     AX              ;
05D5  50                              PUSH    AX
05D6  BB 000A                         MOV     BX,10           ; SET UP FOR DECIMAL CONVERT
05D9  B9 0003                         MOV     CX,3            ; OF 3 NIBBLES
05DC  33 D2                   Q36:    XOR     DX,DX           ;
05DE  F7 F3                           DIV     BX              ; DEVIDE BY 10
05E0  80 CA 30                        OR      DL,30H          ; MAKE INTO ASCII
05E3  52                              PUSH    DX              ; SAVE
05E4  E2 F6                           LOOP    Q36             ;
05E6  B9 0003                         MOV     CX,3            ;
05E9  58                      Q37:    POP     AX              ; RECOVER A NUMBER
05EA  E8 18BA R                       CALL    PRT_HEX
05ED  E2 FA                           LOOP    Q37
05EF  B9 0003                         MOV     CX,3
05F2  8E 0025 R                       MOV     SI,OFFSET F3B   ; PRINT " KB"
05F5  2E: 8A 04               Q38:    MOV     AL,CS:[SI]
05F8  46                              INC     SI
05F9  E8 18BA R                       CALL    PRT_HEX
05FC  E2 F7                           LOOP    Q38
05FE  58                              POP     AX
05FF  5A                              POP     DX
0600  59                              POP     CX
0601  5B                              POP     BX
0602  C3                    Q35E:   RET
0603                          Q35     ENDP
                              ; ON ENTRY TO MEMORY ERROR ROUTINE, CX HAS ERROR BITS
                              ; AH HAS ODD/EVEN INFO, OTHER USEFUL INFO ON THE STACK
0603  5A                    Q39:    POP     DX              ; POP SEGMENT POINTER TO DX
                                                            ; (HEADING DOWNHILL, DON'T CARE
                                                            ; ABOUT STACK)
0604  81 FA 2000                     CMP     DX,2000H        ; ABOVE 128K (THE SIMPLE CASE)
0608  7C 0E                         JL      Q40             ; GO DO ODD/EVEN-LESS THAN 128K
060A  8A D9                         MOV     BL,CL           ; FORM ERROR BITS ("XX")
060C  0A DD                         OR      BL,CH
060E  B1 04                         MOV     CL,4            ; ROTATE MOST SIGNIFIGANT
                                                            ; NIBBLE OF SEGMET
0610  D2 EE                         SHR     DH,CL           ; TO LOW NIBBLE OF DH
0612  B7 10                         MOV     BH,10H          ;
0614  0A FE                         OR      BH,DH           ; FORM "1Y" VALUE
0616  EB 20                         JMP     SHORT Q42
0618  B7 0A                  Q40:    MOV     BH,0AH          ; ERROR 0A....
061A  E4 62                         IN      AL,PORT_C       ; GET CONFIG BITS
061C  24 08                         AND     AL,00001000B    ; TEST FOR ATTRIB CARD PRESENT
061E  74 06                         JZ      Q41             ; WORRY ABOUT ODD/EVEN IF IT IS
0620  8A D9                         MOV     BL,CL
0622  0A DD                         OR      BL,CH           ; COMBINE ERROR BITS IF IT ISN'T
0624  EB 12                         JMP     SHORT Q42       ;
```

```
0626  80 FC 02                Q41:    CMP     AH,02           ; EVEN BYTE ERROR? ERR 0AXX
0629  8A D9                           MOV     BL,CL
062B  74 0B                           JE      Q42
062D  FE C7                           INC     BH              ; MAKE INTO 0BXX ERR
062F  0A DD                           OR      BL,CH           ; MOVE AND COMBINE ERROR BITS
0631  80 FC 01                       CMP     AH,1            ; ODD BYTE ERROR
0634  74 02                           JE      Q42
0636  FE C7                           INC     BH              ; MUST HAVE BEEN BOTH
                                                              ; - MAKE INTO 0CXX
0638  BE 0035 R               Q42:    MOV     SI,OFFSET MEM_ERR
063B  E8 09BC R                       CALL    E_MSG           ; LET ERROR ROUTINE FIGURE OUT
                                                              ; WHAT TO DO
063E  FA                              CLI                     ;
063F  F4                              HLT
0640                          Q43:
                              ;--------------------------------------------
                              ;       KEYBOARD TEST
                              ; DESCRIPTION
                              ;       NMI HAS BEEN ENABLED FOR QUITE A FEW
                              ;       SECONDS NOW. CHECK THAT NO SCAN CODES
                              ;       HAVE SHOWN UP IN THE BUFFER. (STUCK
                              ;       KEY) IF THEY HAVE, DISPLAY THEM AND
                              ;       POST ERROR.
                              ;       MFG ERR CODE
                              ;       2000 STRAY NMI INTERRUPTS OR KEYBOARD
                              ;       RECEIVE ERRORS
                              ;       21XX  CARD FAILURE
                              ;           XX=01, KB DATA STUCK HIGH
                              ;           XX=02, KB DATA STUCK LOW
                              ;           XX=03, NO NMI INTERRUPT
                              ;       22XX STUCK KEY (XX=SCAN CODE)
                              ;--------------------------------------------
                                      ASSUME  DS:DATA
                              ;----- CHECK FOR STUCK KEYS
0640  E8 E6D8 R                       CALL    MFG_UP          ; MFG CODE=F2
0643  E8 138B R                       CALL    DDS             ; ESTABLISH ADDRESSING
0646  BB 001E R                       MOV     BX,OFFSET KB_BUFFER
0649  8A 07                           MOV     AL,[BX]         ; CHECK FOR STUCK KEYS
064B  0A C0                           OR      AL,AL           ; SCAN CODE = 0?
064D  74 06                           JE      F6_Y            ; YES - CONTINUE TESTING
064F  B7 22                           MOV     BH,22H          ; 22XX ERROR CODE
0651  8A D8                           MOV     BL,AL           ;
0653  EB 0A                           JMP     SHORT F6
0655  80 3E 0012 R 00        F6_Y:   CMP     KBD_ERR,00H     ; DID NMI'S HAPPEN WITH NO SCAN
                                                              ; CODE PASSED?
065A  74 1C                           JE      F7              ; (STRAYS) - CONTINUE IF NONE
065C  BB 2000                         MOV     BX,2000H        ; SET ERROR CODE 2000
065F  BE 0036 R               F6:     MOV     SI,OFFSET KEY_ERR ; GET MSG ADDR
0662  81 3E 0072 R 4321               CMP     RESET_FLAG,4321H ; WARM START TO DIAGS
0668  74 0B                           JE      F6_Z            ; DO NOT PUT UP MESSAGE
066A  81 3E 0072 R 1234               CMP     RESET_FLAG,1234H ; WARM SYSTEM START
0670  74 03                           JE      F6_Z            ; DO NOT PUT UP MESSAGE
0672  E8 09BC R                       CALL    E_MSG           ; PRINT MSG ON SCREEN
0675  E9 06FF R               F6_Z:   JMP     F6_X
                              ; CHECK LINK CARD, IF PRESENT
0678  BA 0201                 F7:     MOV     DX,0201H
067B  EC                              IN      AL,DX           ; CHECK FOR BURN-IN MODE
067C  24 F0                           AND     AL,0F0H
067E  74 7F                           JZ      F6_X            ; BYPASS CHECK IN BURN-IN MODE
0680  E4 62                           IN      AL,PORT_C       ; GET CONFIG. PORT DATA
0682  24 80                           AND     AL,10000000B    ; KEYBOARD CABLE ATTACHED?
0684  74 79                           JZ      F6_X            ; BYPASS  TEST IF IT IS
0686  E4 61                           IN      AL,PORT_B       ;
0688  24 FC                           AND     AL,11111100B    ; DROP SPEAKER DATA
068A  E6 61                           OUT     PORT_B,AL       ;
068C  B0 B6                           MOV     AL,0B6H         ; MODE SET TIMER 2
068E  E6 43                           OUT     TIM_CTL,AL      ;
0690  B0 40                           MOV     AL,040H         ; DISABLE NMI
0692  E6 A0                           OUT     0A0H,AL         ;
0694  B0 20                           MOV     AL,32           ; LSB TO TIMER 2
                                                              ; (APPROX. 40Khz VALUE)
0696  BA 0042                        MOV     DX,TIMER+2
0699  EE                              OUT     DX,AL
069A  2B C0                           SUB     AX,AX
069C  8B C8                           MOV     CX,AX
069E  EE                              OUT     DX,AL           ; MSB TO TIMER 2 (START TIMER)
069F  E4 61                           IN      AL,PORT_B       ;
06A1  0C 01                           OR      AL,1
06A3  E6 61                           OUT     PORT_B,AL       ; ENABLE TIMER 2
06A5  E4 62                   F7_0:   IN      AL,PORT_C       ; SEE IF KEYBOARD DATA ACTIVE
06A7  24 40                           AND     AL,01000000B    ;
06A9  75 06                           JNZ     F7_1            ; EXIT LOOP IF DATA SHOWED UP
06AB  E2 F8                           LOOP    F7_0
06AD  B3 02                           MOV     BL,02H          ; SET NO KEYBOARD DATA ERROR
06AF  EB 49                           JMP     SHORT F6_1
06B1  06                      F7_1:   PUSH    ES              ; SAVE ES
06B2  2B C0                           SUB     AX,AX           ; SET UP SEGMENT REG
06B4  8E C0                           MOV     ES,AX           ; *
06B6  26: C7 06 0008 R F815 R         MOV     ES:[NMI_PTR],OFFSET D11 ; SET UP NEW NMI VECTOR
06BD  A2 0084 R                       MOV     INTR_FLAG,AL    ; RESET INTR FLAG
06C0  E4 61                           IN      AL,PORT_B       ; DISABLE INTERNAL BEEPER TO
06C2  0C 30                           OR      AL,00110000B    ; PREVENT ERROR BEEP
06C4  E6 61                           OUT     PORT_B,AL
06C6  B0 C0                           MOV     AL,0C0H
06C8  E6 A0                           OUT     0A0H,AL         ; ENABLE NMI
06CA  B9 0100                         MOV     CX,0100H        ;
```

```
06CD  E2 FE                   F6_0:   LOOP    F6_0            ; WAIT A BIT
06CF  E4 61                           IN      AL,PORT_B       ; RE-ENABLE BEEPER
06D1  24 CF                           AND     AL,11001111B
06D3  E6 61                           OUT     PORT_B,AL
06D5  A0 0084 R                       MOV     AL,INTR_FLAG    ; GET INTR FLAG
06D8  0A C0                           OR      AL,AL           ; WILL BE NON-ZERO IF NMI HAPPENED
06DA  B3 03                           MOV     BL,03H          ; SET POSSIBLE ERROR CODE
06DC  26: C7 06 0008 R 0F78 R         MOV     ES:[NMI_PTR],OFFSET KBDNMI ; RESET NMI VECTOR
06E3  07                              POP     ES              ; RESTORE ES
06E4  74 14                           JZ      F6_1            ; JUMP IF NO NMI
06E6  B0 00                           MOV     AL,00H          ; DISABLE FEEDBACK CKT
06E8  E6 A0                           OUT     0A0H,AL         ;
06EA  E4 61                           IN      AL,PORT_B       ;
06EC  24 FE                           AND     AL,11111110B    ; DROP GATE TO TIMER 2
06EE  E6 61                           OUT     PORT_B,AL       ;
06F0  E4 62                   F6_2:   IN      AL,PORT_C       ; SEE IF KEYBOARD DATA ACTIVE
06F2  24 40                           AND     AL,01000000B    ;
06F4  74 09                           JZ      F6_X            ; EXIT LOOP IF DATA WENT LOW
06F6  E2 F8                           LOOP    F6_2
06F8  B3 01                           MOV     BL,01H          ; SET KEYBOARD DATA STUCK HIGH ERR
06FA  B7 21                   F6_1:   MOV     BH,21H          ; POST ERROR  "21XX"
06FC  E9 065F R                       JMP     F6              ;
06FF  B0 00                   F6_X:   MOV     AL,00H          ; DISABLE FEEDBACK CKT
0701  E6 A0                           OUT     0A0H,AL         ;
                              ;--------------------------------------------------------
                              ;       CASSETTE INTERFACE 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.
                              ;  MFG. ERROR CODE=2300H (DATA PATH ERROR)
                              ;                 23FF (RELAY FAILED TO PICK)
                              ;--------------------------------------------------------
= 0A9A                        MAX_PERIOD      EQU     0A9AH   ;NOM.+10%
= 08AD                        MIN_PERIOD      EQU     08ADH   ;NOM -10%
                              ;----- TURN THE CASSETTE MOTOR OFF
0703  E8 E6D8 R                      CALL    MFG_OUT         ; MFG CODE=F1
0706  E4 61                          IN      AL,PORT_B
0708  0C 09                          OR      AL,00001001B    ; SET TIMER 2 SPK OUT, AND CASSETTE
070A  E6 61                          OUT     PORT_B,AL       ; OUT BITS ON, CASSETTE MOT OFF
                              ;----- WRITE A BIT
070C  E4 21                          IN      AL,INTA01
070E  0C 01                          OR      AL,01H          ; DISABLE TIMER INTERRUPTS
0710  E6 21                          OUT     INTA01,AL
0712  B0 B6                          MOV     AL,0B6H         ; SEL TIM 2, LSB, MSB, MD 3
0714  E6 43                          OUT     TIMER+3,AL      ; WRITE 8253 CMD/MODE REG
0716  B8 04D2                        MOV     AX,1234         ; SET TIMER 2 CNT FOR 1000 USEC
0719  E6 42                          OUT     TIMER+2,AL      ; WRITE TIMER 2 COUNTER REG
071B  8A C4                          MOV     AL,AH           ; WRITE MSB
071D  E6 42                          OUT     TIMER+2,AL
071F  2B C9                          SUB     CX,CX           ; CLEAR COUNTER FOR LONG DELAY
0721  E2 FE                          LOOP    $               ; WAIT FOR COUNTER TO INIT
                              ;----- READ CASSETTE INPUT
0723  E4 62                          IN      AL,PORT_C       ; READ VALUE OF CASS IN BIT
0725  24 10                          AND     AL,10H          ; ISOLATE FROM OTHER BITS
0727  A2 006B R                      MOV     LAST_VAL,AL
072A  E8 F96F R                      CALL    READ_HALF_BIT   ; TO SET UP CONDITIONS FOR CHECK
072D  E8 F96F R                      CALL    READ_HALF_BIT
0730  E3 3E                          JCXZ    F8              ; CAS_ERR
0732  53                            PUSH    BX              ; SAVE HALF BIT TIME VALUE
0733  E8 F96F R                      CALL    READ_HALF_BIT
0736  58                            POP     AX              ; GET TOTAL TIME
0737  E3 37                          JCXZ    F8              ; CAS_ERR
0739  03 C3                          ADD     AX,BX
073B  3D 0A9A                        CMP     AX,MAX_PERIOD
073E  73 30                          JNC     F8              ; CAS_ERR
0740  3D 08AD                        CMP     AX,MIN_PERIOD
0743  72 2B                          JC      F8
0745  BA 0201                        MOV     DX,201H
0748  EC                             IN      AL,DX
0749  24 F0                          AND     AL,0F0H         ; DETERMINE MODE
074B  3C 10                          CMP     AL,00010000B    ; MFG?
074D  74 04                          JE      F9
074F  3C 40                          CMP     AL,01000000B    ; SERVICE?
0751  75 26                          JNE     T13_END         ; GO TO NEXT TEST IF NOT
                              ; CHECK THAT CASSETTE RELAY IS PICKING (CAN'T DO TEST IN NORMAL
                              ; MODE BECAUSE OF POSSIBILITY OF WRITING ON CASSETTE IF "RECORD"
                              ; BUTTON IS DEPRESSED.)
0753  E4 61                   F9:     IN      AL,PORT_B
0755  8A D0                          MOV     DL,AL           ; SAVE PORT B CONTENTS
0757  24 E5                          AND     AL,11100101B    ; SET CASSETTE MOTOR ON
0759  E6 61                          OUT     PORT_B,AL       ;
075B  33 C9                          XOR     CX,CX
075D  E2 FE                   F91:    LOOP    F91             ; WAIT FOR RELAY TO SETTLE
075F  E8 F96F R                      CALL    READ_HALF_BIT
0762  E8 F96F R                      CALL    READ_HALF_BIT
0765  8A C2                          MOV     AL,DL           ; DROP RELAY
0767  E6 61                          OUT     PORT_B,AL
0769  E3 0E                          JCXZ    T13_END         ; READ_HALF_BIT SHOULD TIME OUT IN
                                                              ; THIS SITUATION
076B  BB 23FF                        MOV     BX,23FFH        ; ERROR 23FF
076E  EB 03                          JMP     SHORT F81
0770                          F8:                             ; CAS_ERR
0770  BB 2300                        MOV     BX,2300H        ; ERR. CODE 2300H
0773  BE 0037 R               F81:    MOV     SI,OFFSET CASS_ERR ; CASSETTE WRAP FAILED
0776  E8 09BC R                      CALL    E_MSG           ; GO PRINT ERROR MSG
0779  E4 21                   T13_END:IN      AL,INTA01
077B  24 FE                          AND     AL,0FEH         ; ENABLE TIMER INTS
077D  E6 21                          OUT     INTA01,AL
077F  E4 A0                          IN      AL,NMI_PORT     ; CLEAR NMI FLIP/FLOP
0781  B0 80                          MOV     AL,80H          ; ENABLE NMI INTERUPTS
0783  E6 A0                          OUT     NMI_PORT,AL
```

```
                        ;--------------------------------------------------------
                        ;       SERIAL PRINTER AND MODEM POWER ON DIAGNOSTIC
                        ;  DESCRIPTION:
                        ;       VERIFIES THAT THE SERIAL PRINTER UART FUNCTIONS PROPERLY.
                        ;       CHECKS IF THE MODEM CARD IS ATTACHED.  IF IT'S NOT, EXITS.
                        ;       VERIFIES THAT THE MODEM UART FUNCTIONS PROPERLY.
                        ;       ERROR CODES RETURNED BY 'UART' RANGE FROM 1 TO 1FH AND ARE
                        ;       REPORTED VIA REGISTER BL.  SEE LISTING OF 'UART' (POD27)
                        ;       FOR POSSIBLE ERRORS.
                        ;       MFG. ERR. CODES  23XX FOR SERIAL PRINTER
                        ;                        24XX FOR MODEM
                        ;--------------------------------------------------------
                                ASSUME  CS:CODE,DS:DATA
                        ;--------------------------------------------------------
                        ;       TEST SERIAL PRINTER INS8250 UART
                        ;--------------------------------------------------------
0785  E8 E6D8 R                  CALL    MFG_UP          ; MFG ROUTINE INDICATOR=F0
0788  BA 02F8                   MOV     DX,02F8H        ; ADDRESS OF SERIAL PRINTER CARD
078B  E8 E831 R                  CALL    UART            ; ASYNCH. COMM. ADAPTER POD
078E  73 06                     JNC     TM              ; PASSED
0790  BE 0038 R                  MOV     SI,OFFSET COM1_ERR ; CODE FOR DISPLAY
0793  E8 09BC R                  CALL    E_MSG           ; REPORT ERROR
                        ;--------------------------------------------------------
                        ;       TEST MODEM INS8250 UART
                        ;--------------------------------------------------------
0796  E8 E6D8 R         TM:     CALL    MFG_UP          ; MFG ROUTINE INDICATOR = EF
0799  E4 62                     IN      AL,PORT_C       ; TEST FOR MODEM CARD PRESENT
079B  24 02                     AND     AL,00000010B    ; ONLY CONCERNED WITH BIT 1
079D  75 0E                     JNE     TM1             ; IT'S NOT THERE - DONE WITH TEST
079F  BA 03F8                   MOV     DX,03F8H        ; ADDRESS OF MODEM CARD
07A2  E8 E831 R                 CALL    UART            ; ASYNCH. COMM. ADAPTER POD
07A5  73 06                     JNC     TM1             ; PASSED
07A7  BE 0039 R                 MOV     SI,OFFSET COM2_ERR ; MODEM ERROR
07AA  E8 09BC R                 CALL    E_MSG           ; REPORT ERROR
07AD                    TM1:
                        ;-----------------------------------------
                        ;       SETUP HARDWARE INT. VECTOR TABLE
                        ;-----------------------------------------
                                ASSUME  CS:CODE,DS:ABS0
07AD  2B C0                     SUB     AX,AX
07AF  8E C0                     MOV     ES,AX
07B1  B9 0008                   MOV     CX,08           ; GET VECTOR CNT
07B4  0E                        PUSH    CS              ; SETUP DS SEG REG
07B5  1F                        POP     DS
07B6  BE FEF3 R                 MOV     SI,OFFSET VECTOR_TABLE
07B9  BF 0020 R                 MOV     DI,OFFSET INT_PTR
07BC  A5                F7A:    MOVSW
07BD  47                        INC     DI              ; SKIP OVER SEGMENT
07BE  47                        INC     DI
07BF  E2 FB                     LOOP    F7A
                        ;----- SET UP OTHER INTERRUPTS AS NECESSARY
                                ASSUME  DS:ABS0
07C1  8E D9                     MOV     DS,CX
07C3  C7 06 0014 R FF54 R       MOV     INT5_PTR,OFFSET PRINT_SCREEN ; PRINT SCREEN
07C9  C7 06 0120 R 10C6 R       MOV     KEY62_PTR,OFFSET KEY62_INT ; 62 KEY CONVERSION
                                                                   ; ROUTINE
07CF  C7 06 0110 R FA6E R       MOV     CSET_PTR,OFFSET CRT_CHAR_GEN ; DOT TABLE
07D5  C7 06 0060 R FFC8 R       MOV     BASIC_PTR,OFFSET BAS_ENT ; CASSETTE BASIC ENTRY
07DB  0E                        PUSH    CS
07DC  58                        POP     AX
07DD  A3 0062 R                 MOV     WORD PTR BASIC_PTR+2,AX ; CODE SEGMENT FOR CASSETTE
                        ;--------------------------------------------------------
                        ; CHECK FOR OPTIONAL ROM FROM C0000 TO F0000 IN 2K BLOCKS
                        ;       (A VALID MODULE HAS '55AA' IN THE FIRST 2 LOCATIONS,
                        ;       LENGTH INDICATOR (LENGTH/512) IN THE 3D LOCATION AND
                        ;       TEST/INIT. CODE STARTING IN THE 4TH LOCATION.)
                        ;       MFG ERR CODE 25XX (XX=MSB OF SEGMENT THAT HAS CRC CHECK)
                        ;--------------------------------------------------------
07E0  B0 01                     MOV     AL,01H
07E2  E6 13                     OUT     13H,AL
07E4  E8 E6D8 R                 CALL    MFG_UP          ; MFG ROUTINE = EE
07E7  BA C000                   MOV     DX,0C000H       ; SET BEGINNING ADDRESS
07EA                    ROM_SCAN_1:
07EA  8E DA                     MOV     DS,DX
07EC  2B DB                     SUB     BX,BX           ; SET BX=0000
07EE  8B 07                     MOV     AX,[BX]         ; GET 1ST WORD FROM MODULE
07F0  53                        PUSH    BX
07F1  5B                        POP     BX              ; BUS SETTLING
07F2  3D AA55                   CMP     AX,0AA55H       ; = TO ID WORD?
07F5  75 05                     JNZ     NEXT_ROM        ; PROCEED TO NEXT ROM IF NOT
07F7  E8 EB51 R                 CALL    ROM_CHECK       ; GO CHECK OUT MODULE
07FA  EB 04                     JMP     SHORT ARE_WE_DONE ; CHECK FOR END OF ROM SPACE
07FC                    NEXT_ROM:
07FC  81 C2 0080                ADD     DX,0080H        ; POINT TO NEXT 2K ADDRESS
0800                    ARE_WE_DONE:
0800  81 FA F000                CMP     DX,0F000H       ; AT F0000 YET?
0804  7C E4                     JL      ROM_SCAN_1      ; GO CHECK ANOTHER ADD. IF NOT
```

```
                        ;--------------------------------------------------------------------
                        ;       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.
                        ;   MFG ERR CODES: 2601 RESET TO DISKETTE CONTROLLER CD. FAILED
                        ;                  2602 RECALIBRATE TO DISKETTE DRIVE FAILED
                        ;                  2603 WATCHDOG TIMER FAILED
                        ;--------------------------------------------------------------------

                                ASSUME  CS:CODE,DS:DATA
0806  E8 E6D8 R                 CALL    MFG_UP                  ; MFG ROUTINE = ED
0809  E8 1388 R                 CALL    DDS                     ; POINT TO DATA AREA
080C  B0 FF                     MOV     AL,0FFH
080E  A2 0074 R                 MOV     TRACK0,AL               ; INIT DISKETTE SCRATCHPADS
0811  A2 0075 R                 MOV     TRACK1,AL
0814  A2 0076 R                 MOV     TRACK2,AL
0817  E4 62                     IN      AL,PORT_C               ; DISKETTE PRESENT?
0819  24 04                     AND     AL,00000100B
081B  74 03                     JZ      F10_0
081D  E9 08A3 R                 JMP     F15                     ; NO - BYPASS DISKETTE TEST
0820  80 0E 0010 R 01    F10_0:  OR      BYTE PTR EQUIP_FLAG,01H ; SET IPL DISKETTE
                                                                ; INDICATOR IN EQUIP. FLAG
0825  83 3E 0072 R 00           CMP     RESET_FLAG,0            ; RUNNING FROM POWER-ON STATE?
082A  75 0E                     JNE     F10                     ; BYPASS WATCHDOG TEST
082C  B0 0A                     MOV     AL,00001010B            ; READ INT. REQUEST REGISTER CMD
082E  E6 20                     OUT     INTA00,AL               ;
0830  E4 20                     IN      AL,INTA00
0832  24 40                     AND     AL,01000000B            ; HAS WATCHDOG GONE OFF?
0834  75 04                     JNZ     F10                     ; PROCEED IF IT HAS
0836  B3 03                     MOV     BL,03H                  ; SET ERROR CODE
0838  EB 33                     JMP     SHORT F13
083A  B0 80             F10:    MOV     AL,FDC_RESET
083C  E6 F2                     OUT     0F2H,AL                 ; DISABLE WATCHDOG TIMER
083E  B4 00                     MOV     AH,0                    ; RESET NEC FDC
0840  8A D4                     MOV     DL,AH                   ; SET FOR DRIVE 0
0842  CD 13                     INT     13H                     ; VERIFY STATUS AFTER RESET
0844  F6 C4 FF                 TEST    AH,0FFH                 ; STATUS OK?
0847  B3 01                     MOV     BL,01H                  ; SET UP POSSIBLE ERROR CODE
0849  75 22                     JNZ     F13                     ; NO - FDC FAILED
                        ;-----  TURN DRIVE 0 MOTOR ON
084B  B0 81                     MOV     AL,DRIVE_ENABLE+FDC_RESET ; TURN MOTOR ON,DRIVE 0
084D  E6 F2                     OUT     0F2H,AL                 ; WRITE FDC CONTROL REG
084F  2B C9                     SUB     CX,CX
0851  E2 FE             F11:    LOOP    F11                     ; WAIT FOR 1 SECOND
0853  E2 FE             F12:    LOOP    F12
0855  33 D2                     XOR     DX,DX                   ; SELECT DRIVE 0
0857  B5 01                     MOV     CH,1                    ; SELECT TRACK 1
0859  88 16 003E R              MOV     SEEK_STATUS,DL
085D  E8 E9FB R                 CALL    SEEK                    ; RECALIBRATE DISKETTE
0860  B3 02                     MOV     BL,02H                  ; ERROR CODE
0862  72 09                     JC      F13                     ; GO TO ERR SUBROUTINE IF ERR
0864  B5 22                     MOV     CH,34                   ; SELECT TRACK 34
0866  E8 E9FB R                 CALL    SEEK                    ; SEEK TO TRACK 34
0869  73 0A                     JNC     F14                     ; OK, TURN MOTOR OFF
086B  B3 02                     MOV     BL,02H
086D  B7 26             F13:    MOV     BH,26H                  ; DSK_ERR:(26XX)
086F  8E 003C R                 MOV     SI,OFFSET DISK_ERR ; GET ADDR OF MSG
0872  E8 09BC R                 CALL    E_MSG                   ; GO PRINT ERROR MSG
0875  B0 82             F14:    MOV     AL,FDC_RESET+02H
0877  E6 F2                     OUT     0F2H,AL
0879  E4 E2                     IN      AL,0E2H
087B  24 06                     AND     AL,00000110B
087D  3C 02                     CMP     AL,00000010B
087F  75 1E                     JNE     F14_1
0881  B0 84                     MOV     AL,FDC_RESET+04H
0883  E6 F2                     OUT     0F2H,AL
0885  E4 E2                     IN      AL,0E2H
0887  24 06                     AND     AL,00000110B
0889  3C 04                     CMP     AL,00000100B
088B  75 12                     JNE     F14_1
088D  E4 E2                     IN      AL,0E2H
088F  24 30                     AND     AL,00110000B
0891  74 0C                     JZ      F14_1
0893  3C 10                     CMP     AL,00010000B
0895  B4 40                     MOV     AH,01000000B
0897  74 02                     JE      F14_2
0899  B4 80                     MOV     AH,10000000B
089B  08 26 0010 R      F14_2:  OR      BYTE PTR EQUIP_FLAG,AH
                        ;-----  TURN DRIVE 0 MOTOR OFF
089F  B0 80             F14_1:  MOV     AL,FDC_RESET    ; TURN DRIVE 0 MOTOR OFF
08A1  E6 F2                     OUT     0F2H,AL
08A3  C6 06 0084 R 00   F15:    MOV     INTR_FLAG,00H   ; SET STRAY INTERRUPT FLAG = 00
08A8  BF 0078 R                 MOV     DI,OFFSET PRINT_TIM_OUT ;SET DEFAULT PRT TIMEOUT
08AB  1E                        PUSH    DS
08AC  07                        POP     ES
08AD  B8 1414                   MOV     AX,1414H        ; DEFAULT=20
08B0  AB                        STOSW
08B1  AB                        STOSW
08B2  B8 0101                   MOV     AX,0101H        ;RS232 DEFAULT=01
08B5  AB                        STOSW
08B6  AB                        STOSW
08B7  E4 21                     IN      AL,INTA01
08B9  24 FE                     AND     AL,0FEH         ; ENABLE TIMER INT. (LVL 0)
08BB  E6 21                     OUT     INTA01,AL
                                ASSUME  DS:XXDATA
08BD  1E                        PUSH    DS
08BE  B8  ---- R                MOV     AX,XXDATA
08C1  8E D8                     MOV     DS,AX
```

```
08C3  80 3E 0018 R 00                 CMP     POST_ERR,00H    ; CHECK FOR "POST_ERR" NON-ZERO
                                      ASSUME  DS:DATA
08C8  1F                              POP     DS
08C9  74 10                           JE      F15A_0          ; CONTINUE IF NO ERROR
08CB  B2 02                           MOV     DL,2            ; 2 SHORT BEEPS (ERROR)
08CD  E8 1A0C R                       CALL    ERR_BEEP
08D0                            ERR_WAIT:
08D0  B4 00                           MOV     AH,00
08D2  CD 16                           INT     16H             ; WAIT FOR "ENTER" KEY
08D4  80 FC 1C                        CMP     AH,1CH
08D7  75 F7                           JNE     ERR_WAIT
08D9  EB 05                           JMP     SHORT F15C
08DB  B2 01                     F15A_0: MOV   DL,1            ; 1 SHORT BEEP (NO ERRORS)
08DD  E8 1A0C R                       CALL    ERR_BEEP
                                ;----- SETUP PRINTER AND RS232 BASE ADDRESSES IF DEVICE ATTACHED
08E0  BD 003D R                 F15C:   MOV   BP,OFFSET F4    ; PRT_SRC_TBL
08E3  33 F6                           XOR     SI,SI
08E5                            F16:                          ; PRT_BASE:
08E5  2E: 8B 56 00                    MOV     DX,CS:[BP]      ; GET PRINTER BASE ADDR
08E9  B0 AA                           MOV     AL,0AAH         ; WRITE DATA TO PORT A
08EB  EE                              OUT     DX,AL
08EC  1E                              PUSH    DS              ; BUS SETTLING
08ED  EC                              IN      AL,DX           ; READ PORT A
08EE  1F                              POP     DS
08EF  3C AA                           CMP     AL,0AAH         ; DATA PATTERN SAME
08F1  75 06                           JNE     F17             ; NO - CHECK NEXT PRT CD
08F3  89 94 0008 R                    MOV     PRINTER_BASE[SI],DX ; YES - STORE PRT BASE ADDR
08F7  46                              INC     SI              ; INCREMENT TO NEXT WORD
08F8  46                              INC     SI
08F9  45                        F17:  INC     BP              ; POINT TO NEXT BASE ADDR
08FA  45                              INC     BP
08FB  83 FD 41                        CMP     BP,OFFSET F4E   ; ALL POSSIBLE ADDRS CHECKED?
08FE  75 E5                           JNE     F16             ; PRT_BASE
0900  33 DB                           XOR     BX,BX           ; SET ADDRESS BASE
0902  BA 03FA                         MOV     DX,03FAH        ; POINT TO INT ID REGISTER
0905  EC                              IN      AL,DX           ; READ PORT
0906  A8 F8                           TEST    AL,0F8H         ; SEEM TO BE AN 8250
0908  75 08                           JNZ     F18
090A  C7 87 0000 R 03F8               MOV     RS232_BASE[BX],3F8H ; SETUP RS232 CD #1 ADDR
0910  43                              INC     BX
0911  43                              INC     BX
0912  C7 87 0000 R 02F8         F18:   MOV    RS232_BASE[BX],2F8H ; SETUP RS232 #2
0918  43                              INC     BX              ; (ALWAYS PRESENT)
0919  43                              INC     BX
                                ;----- SET UP EQUIP FLAG TO INDICATE NUMBER OF PRINTERS AND RS232
                                ;       CARDS
091A  8B C6                           MOV     AX,SI           ; SI HAS 2* NUMBER OF PRINTERS
091C  B1 03                           MOV     CL,3            ; SHIFT COUNT
091E  D2 C8                           ROR     AL,CL           ; ROTATE RIGHT 3 POSITIONS
0920  0A C3                           OR      AL,BL           ; OR IN THE RS232 COUNT
0922  08 06 0011 R                    OR      BYTE PTR EQUIP_FLAG+1,AL ; STORE AS SECOND BYTE
                                ;----- SET EQUIP. FLAG TO INDICATE PRESENCE OF SERIAL PRINTER
                                ; ATTACHED TO ON BOARD RS232 PORT. ---ASSUMPTION--"RTS" IS TIED TO
                                ; "CARRIER DETECT" IN THE CABLE PLUG FOR THIS SPECIFIC PRINTER.
0926  8B C8                           MOV     CX,AX           ; SAVE PRINTER COUNT IN CX
0928  BB 02FE                         MOV     BX,2FEH         ; SET POINTER TO MODEM STATUS REG
092B  BA 02FC                         MOV     DX,2FCH         ; POINT TO MODEM CONTROL REG
092E  2A C0                           SUB     AL,AL
0930  EE                              OUT     DX,AL           ; CLEAR IT
0931  EB 00                           JMP     $+2             ; DELAY
0933  87 D3                           XCHG    DX,BX           ; POINT TO MODEM STATUS REG
0935  EC                              IN      AL,DX           ; CLEAR IT
0936  EB 00                           JMP     $+2             ; DELAY
0938  B0 02                           MOV     AL,02H          ; BRING UP RTS
093A  87 D3                           XCHG    DX,BX           ; POINT TO MODEM CONTROL REG
093C  EE                              OUT     DX,AL           ;
093D  EB 00                           JMP     $+2             ; DELAY
093F  87 D3                           XCHG    DX,BX           ; POINT TO MODEM STATUS REG
0941  EC                              IN      AL,DX           ; GET CONTENTS
0942  A8 08                           TEST    AL,00001000B    ; HAS CARRIER DETECT CHANGED?
0944  74 23                           JZ      F19_A           ; NO, THEN NO PRINTER
0946  A8 01                           TEST    AL,00000001B    ; DID CTS CHANGE? (AS WITH WRAP
                                                              ; CONNECTOR INSTALLED)
0948  75 1F                           JNZ     F19_A           ; WRAP-CONNECTOR ON IF IT DID
094A  2A C0                           SUB     AL,AL           ; SET RTS OFF
094C  87 D3                           XCHG    DX,BX           ; POINT TO MODEM CONTROL REG
094E  EE                              OUT     DX,AL           ; DROP RTS
094F  EB 00                           JMP     $+2             ; DELAY
0951  87 D3                           XCHG    DX,BX           ; MODEM STATUS REG
0953  EC                              IN      AL,DX           ; GET STATUS
0954  24 08                           AND     AL,00001000B    ; HAS CARRIER DETECT CHANGED?
0956  74 11                           JZ      F19_A           ; NO, THEN NO PRINTER
                                ; CARRIER DETECT IS FOLLOWING RTS-INDICATE SERIAL PRINTER ATTACHED
0958  80 C9 20                        OR      CL,00100000B    ;
095B  F6 C1 C0                        TEST    CL,11000000B    ; CHECK FOR NO PARALLEL PRINTERS
095E  75 09                           JNZ     F19_A           ; DO NOTHING IF PARALLEL PRINTER
                                                              ; ATTACHED
0960  80 C9 40                        OR      CL,01000000B    ; INDICATE 1 PRINTER ATTACHED
0963  C7 06 0008 R 02F8               MOV     PRINTER_BASE,2F8H ; STORE ON-BOARD RS232 BASE IN
                                                              ; PRINTER BASE
0969  08 0E 0011 R              F19_A: OR     BYTE PTR EQUIP_FLAG+1,CL ; STORE AS SECOND BYTE
096D  33 D2                           XOR     DX,DX           ; POINT TO FIRST SERIAL PORT
096F  F6 C1 40                        TEST    CL,040H         ; SERIAL PRINTER ATTACHED?
0972  74 18                           JZ      F19_C           ; NO, SKIP INIT
0974  81 3E 0000 R 02F8               CMP     RS232_BASE,02F8H ; PRINTER IN FIRST SERIAL PORT
097A  74 01                           JE      F19_B           ; YES, JUMP
097C  42                              INC     DX              ; NO POINT TO SECOND SERIAL PORT
097D  B8 0087                   F19_B: MOV    AX,87H          ; INIT SERIAL PRINTER
0980  CD 14                           INT     14H
0982  F6 C4 1E                        TEST    AH,1EH          ; ERROR?
0985  75 05                           JNZ     F19_C           ; YES, JUMP
0987  BB 0118                         MOV     AX,0118H        ; SEND CANCEL COMMAND TO
098A  CD 14                           INT     14H             ; ..SERIAL PRINTER
```

```
098C  BA 0201                 F19_C:  MOV     DX,0201H
098F  EC                              IN      AL,DX           ; GET MFG./ SERVICE  MODE INFO
0990  24 F0                           AND     AL,0F0H         ; IS HIGH ORDER NIBBLE = 0?
0992  75 03                           JNZ     F19_1           ; (BURN-IN MODE)
0994  E9 0043 R               F19_0:  JMP     START           ; ELSE GO TO BEGINNING OF POST
0997  3C 20                   F19_1:  CMP     AL,00100000B    ; SERVICE MODE LOOP?
0999  74 F9                           JE      F19_0           ; BRANCH TO START
099B  81 3E 0072 R 4321               CMP     RESET_FLAG,4321H ; DIAG. CONTROL PROGRAM RESTART?
09A1  74 0C                           JE      F19_3           ; NO, GO BOOT
09A3  3C 10                           CMP     AL,00010000B    ; MFG DCP RUN REQUEST
09A5  74 08                           JE      F19_3
09A7  C7 06 0072 R 1234               MOV     RESET_FLAG,1234H ; SET WARM START INDICATOR IN CASE
                                                              ; OF CARTRIDGE RESET
09AD  CD 19                           INT     19H             ; GO TO THE BOOT LOADER
                                      ASSUME  DS:ABS0
09AF  FA                      F19_3:  CLI
09B0  2B C0                           SUB     AX,AX
09B2  8E D8                           MOV     DS,AX           ; RESET TIMER INT.
09B4  C7 06 0020 R FEA5 R             MOV     INT_PTR,OFFSET TIMER_INT
09BA  CD 80                           INT     80H             ; ENTER DCP THROUGH INT. 80H
                              ;-----------------------------------------------------------------
                              ;       THIS SUBROUTINE IS THE GENERAL ERROR HANDLER FOR THE POST
                              ;
                              ; ENTRY REQUIREMENTS:
                              ;       SI = OFFSET(ADDRESS) OF MESSAGE BUFFER
                              ;       BX= ERROR CODE FOR MANUFACTURING OR SERVICE MODE
                              ;       REGISTERS ARE NOT PRESERVED
                              ;       LOCATION "POST_ERR" IS SET NON-ZERO IF AN ERROR OCCURS IN
                              ;       CUSTOMER MODE
                              ;       SERVICE/MANUFACTURING FLAGS AS FOLLOWS: (HIGH NIBBLE OF
                              ;       PORT 201)
                              ;       0000 = MANUFACTURING (BURN-IN) MODE
                              ;       0001 = MANUFACTURING (SYSTEM TEST) MODE
                              ;       0010 = SERVICE MODE (LOOP POST)
                              ;       0100 = SERVICE MODE (SYSTEM TEST)
                              ;-----------------------------------------------------------------
09BC                          E_MSG   PROC    NEAR
09BC  BA 0201                         MOV     DX,201H
09BF  EC                              IN      AL,DX           ; GET MODE BITS
09C0  24 F0                           AND     AL,0F0H         ; ISOLATE BITS OF INTEREST
09C2  75 03                           JNZ     EM0
09C4  E9 0A61 R                       JMP     MFG_OUT         ; MANUFACTURING MODE (BURN-IN)
09C7  3C 10                   EM0:    CMP     AL,00010000B    ;
09C9  75 03                           JNE     EM1
09CB  E9 0A61 R                       JMP     MFG_OUT         ; MFG. MODE (SYSTEM TEST)
09CE  8A F0                   EM1:    MOV     DH,AL           ; SAVE MODE
09D0  80 FF 0A                        CMP     BH,0AH          ; ERROR CODE ABOVE 0AH (CRT STARTED
                                                              ;  DISPLAY POSSIBLE)?
09D3  7C 63                           JL      BEEPS           ; DO BEEP OUTPUT IF BELOW 10H
09D5  53                              PUSH    BX              ; SAVE ERROR AND MODE FLAGS
09D6  56                              PUSH    SI
09D7  52                              PUSH    DX
09D8  B4 02                           MOV     AH,2            ; SET CURSOR
09DA  BA 1521                         MOV     DX,1521H        ; ROW 21, COL.33
09DD  B7 07                           MOV     BH,7            ; PAGE 7
09DF  CD 10                           INT     10H
09E1  BE 0030 R                       MOV     SI,OFFSET ERROR_ERR
09E4  B9 0005                         MOV     CX,5            ; PRINT WORD "ERROR"
09E7  2E: 8A 04               EM_0:   MOV     AL,CS:[SI]
09EA  46                              INC     SI
09EB  E8 18BA R                       CALL    PRT_HEX
09EE  E2 F7                           LOOP    EM_0
                              ; LOOK FOR A BLANK SPACE TO POSSIBLY PUT CUSTOMER LEVEL ERRORS (IN
                              ; CASE OF MULTI ERROR)
09F0  B6 16                           MOV     DH,16H
09F2  B4 02                   EM_1:   MOV     AH,2            ; SET CURSOR
09F4  CD 10                           INT     10H             ; ROW 22,  COL33 (OR ABOVE, IF
                                                              ; MULTIPLE ERRS)
09F6  B4 08                           MOV     AH,8            ; READ CHARACTER THIS POSITION
09F8  CD 10                           INT     10H
09FA  FE C2                           INC     DL              ; POINT TO NEXT POSTION
09FC  3C 20                           CMP     AL,' '          ; BLANK?
09FE  75 F2                           JNE     EM_1            ; GO CHECK NEXT POSITION, IF NOT
0A00  5A                              POP     DX              ; RECOVER ERROR POINTERS
0A01  5E                              POP     SI
0A02  5B                              POP     BX
0A03  80 FE 20                        CMP     DH,00100000B    ; SERVICE MODE?
0A06  74 21                           JE      SERV_OUT        ;
0A08  80 FE 40                        CMP     DH,01000000B    ;
0A0B  74 1C                           JE      SERV_OUT
0A0D  2E: 8A 04               EM_2:   MOV     AL,CS:[SI]      ; GET ERROR CHARACTER
0A10  E8 18BA R                       CALL    PRT_HEX         ; DISPLAY IT
0A13  80 FF 20                        CMP     BH,20H          ; ERROR BELOW 20? (MEM TROUBLE?)
0A16  7D 03                           JNL     EM_2
0A18  E9 0ABB R                       JMP     TOTLTP0         ; HALT SYSTEM IF SO.
                                      ASSUME  DS:XXDATA       ;
0A1B  1E                      EM_2:   PUSH    DS
0A1C  50                              PUSH    AX
0A1D  B8  ---- R                      MOV     AX,XXDATA
0A20  8E D8                           MOV     DS,AX
0A22  88 3E 0018 R                    MOV     POST_ERR,BH     ; SET ERROR FLAG NON-ZERO
0A26  58                              POP     AX
0A27  1F                              POP     DS
                                      ASSUME  DS:NOTHING
0A28  C3                              RET                     ; RETURN TO CALLER
```

```
0A29                            SERV_OUT:
0A29  8A C7                             MOV     AL,BH           ; PRINT MSB
0A2B  53                                PUSH    BX
0A2C  E8 18A9 R                         CALL    XPC_BYTE        ; DISPLAY IT
0A2F  5B                                POP     BX
0A30  8A C3                             MOV     AL,BL           ; PRINT LSB
0A32  E8 18A9 R                         CALL    XPC_BYTE
0A35  E9 0ABB R                         JMP     TOTLTPO
0A38  FA                        BEEPS:  CLI
0A39  8C C8                             MOV     AX,CS           ; SET CODE SEG= STACK SEG
0A3B  8E D0                             MOV     SS,AX           ; (STACK IS LOST, BUT THINGS ARE
                                                                ; OVER, ANYWAY)
0A3D  B2 02                             MOV     DL,2            ; 2 BEEPS
0A3F  BC 0028 R                         MOV     SP,OFFSET EX_O  ; SET DUMMY RETURN
0A42  B3 01                     EB:     MOV     BL,1            ; SHORT BEEP
0A44  E9 FF31 R                         JMP     BEEP            ;
0A47  E2 FE                     EB0:    LOOP    EB0             ; WAIT (BEEPER OFF)
0A49  FE CA                             DEC     DL              ; DONE YET?
0A4B  75 F5                             JNZ     EB              ; LOOP IF NOT
0A4D  80 FF 05                          CMP     BH,05H          ; 64K CARD ERROR?
0A50  75 69                             JNE     TOTLTPO         ; END IF NOT
0A52  80 FE 20                          CMP     DH,00100000B    ; SERVICE MODE?
0A55  74 05                             JE      EB1
0A57  80 FE 40                          CMP     DH,01000000B    ;
0A5A  75 5F                             JNE     TOTLTPO         ; END IF NOT
0A5C  B3 01                     EB1:    MOV     BL,1            ; ONE MORE BEEP FOR 64K ERROR IF IN
                                                                ; SERVICE MODE
0A5E  E9 FF31 R                         JMP     BEEP
0A61                            MFG_OUT:
0A61  FA                                CLI
0A62  E4 61                             IN      AL,PORT_B
0A64  24 FC                             AND     AL,0FCH
0A66  E6 61                             OUT     PORT_B,AL
0A68  BA 0011                           MOV     DX,11H          ; SEND DATA TO  ADDRESSES 11,12
0A6B  8A C7                             MOV     AL,BH           ;
0A6D  EE                                OUT     DX,AL           ; SEND HIGH BYTE
0A6E  42                                INC     DX
0A6F  8A C3                             MOV     AL,BL           ;
0A71  EE                                OUT     DX,AL           ; SEND LOW BYTE
                                ; INIT. ON-BOARD RS232 PORT FOR COMMUNICATIONS W/MFG MONITOR
                                        ASSUME  DS:XXDATA
0A72  B8  ---- R                        MOV     AX,XXDATA
0A75  8E D8                             MOV     DS,AX           ; POINT TO DATA SEGMENT CONTAINING
                                                                ; CHECKPOINT #
0A77  8C C8                             MOV     AX,CS
0A79  8E D0                             MOV     SS,AX           ; SET STACK FOR RTN
0A7B  BC 002E R                         MOV     SP,OFFSET EX1
0A7E  BA 02FB                           MOV     DX,02FBH        ; LINE CONTROL REG. ADDRESS
0A81  E9 F085 R                         JMP     S8250           ; GO SET UP FOR 9600, ODD, 2 STOP
                                                                ; BITS, 8 BITS
0A84  8B CA                     M01:    MOV     CX,DX           ; DX CAME BACK WITH XMIT REG
                                                                ; ADDRESS IN IT
0A86  BA 02FC                           MOV     DX,02FCH        ; MODEM CONTROL REG
0A89  2A C0                             SUB     AL,AL           ; SET DTR AND RTS LOW SO POSSIBLE
                                                                ; WRAP PLUG WON'T CONFUSE THINGS
0A8B  EE                                OUT     DX,AL
0A8C  BA 02FE                           MOV     DX,02FEH        ; MODEM STATUS REG
0A8F  EC                        M02:    IN      AL,DX
0A90  24 10                             AND     AL,00010000B    ; CTS UP YET?
0A92  74 FB                             JZ      M02             ; LOOP TILL IT IS
0A94  4A                                DEC     DX              ; SET DX=2FD (LINE STATUS REG)
0A95  87 D1                             XCHG    DX,CX           ; POINT TO XMIT. DATA REG
0A97  A0 0005 R                         MOV     AL,MFG_TST      ; GET MFG ROUTINE ERROR INDICATOR
0A9A  EE                                OUT     DX,AL           ; (MAY BE WRONG FOR EARLY ERRORS)
0A9B  EB 00                             JMP     $+2             ; DELAY
0A9D  87 D1                             XCHG    DX,CX           ; POINT DX=2FD
0A9F  EC                        M03:    IN      AL,DX           ; TRANSMIT EMPTY?
0AA0  24 20                             AND     AL,00100000B
0AA2  EB 00                             JMP     $+2             ; DELAY
0AA4  74 F9                             JZ      M03             ; LOOP TILL IT IS
0AA6  87 D1                             XCHG    DX,CX
0AA8  8A C7                             MOV     AL,BH           ; GET MSB OF ERROR WORD
0AAA  EE                                OUT     DX,AL
0AAB  EB 00                             JMP     $+2             ; DELAY
0AAD  87 D1                             XCHG    DX,CX
0AAF  EC                        M04:    IN      AL,DX           ; WAIT FOR XMIT EMPTY
0AB0  24 20                             AND     AL,00100000B
0AB2  EB 00                             JMP     $+2             ; DELAY
0AB4  74 F9                             JZ      M04
0AB6  8A C3                             MOV     AL,BL           ; GET LSB OF ERROR WORD
0AB8  87 D1                             XCHG    DX,CX
0ABA  EE                                OUT     DX,AL
0ABB                            TOTLTPO:
0ABB  FA                                CLI                     ; DISABLE INTS.
0ABC  2A C0                             SUB     AL,AL           ;
0ABE  E6 F2                             OUT     0F2H,AL         ; STOP DISKETTE MOTOR
0AC0  E6 A0                             OUT     0A0H,AL         ; DISABLE NMI
0AC2  F4                                HLT                     ; HALT
0AC3  C3                                RET
0AC4                            E_MSG   ENDP
```

```
;--------------------------------------------------------------------------
;   SUBROUTINE TO INITIALIZE INS8250 PORTS TO THE MASTER RESET
;       STATUS.  THIS ROUTINE ALSO TESTS THE PORTS' PERMANENT
;       ZERO BITS.
;   EXPECTS TO BE PASSED:
;       (DX) = ADDRESS OF THE 8250 TRANSMIT/RECEIVE BUFFER
;   UPON RETURN:
;       (CF) = 1  IF ONE OF THE PORTS' PERMANENT ZERO BITS WAS NOT
;                 ZERO (ERR)
;                 (DX) = PORT ADDRESS THAT FAILED TEST
;                 (AL) = MEANINGLESS
;                 (BL) = 2  INTR ENBL REG  BITS NOT 0
;                        3  INTR ID REG BITS NOT 0
;                        4  MODEM CTRL REG  BITS NOT 0
;                        5  LINE STAT REG BITS NOT 0
;              0  IF ALL PORTS' PERMANENT ZERO BITS WERE ZERO
;                 (DX) = TRANSMIT/RECEIVE BUFFER ADDRESS
;                 (AL) = LAST VALUE READ FROM RECEIVER BUFFER
;                 (BL) = 5 (MEANINGLESS)
;       PORTS SET UP AS FOLLOWS ON ERROR-FREE RETURN:
;        XF9 - INTR ENBL REG  = 0          ALL INTERRUPTS DISABLED
;        XFA - INTR ID REG    = 00000001B  NO INTERRUPTS PENDING
;        XFB - LINE CTRL REG  = 0          ALL BITS LOW
;        XFC - MODEM CTRL REG = 0          ALL BITS LOW
;        XFD - LINE STAT REG  = 01100000B  TRANSMITTER HOLDING
;                                          REGISTER AND TRANSMITTER EMPTY ON
;        XFE - MODEM STAT REG = XXXX0000B  WHERE X 'S REPRESENT
;                                          INPUT SIGNALS
;   REGISTERS DX, AL, AND BL ARE ALTERED.  NO OTHER REGISTERS USED.
;--------------------------------------------------------------------------
0AC4                    I8250   PROC    NEAR
0AC4  EC                        IN      AL,DX           ; READ RECVR BUFFER BUT IGNORE
                                                        ; CONTENTS
0AC5  B3 02                     MOV     BL,2            ; ERROR INDICATOR
0AC7  E8 FE9F R                 CALL    RR2             ; READ INTR ENBL REG
0ACA  24 F0                     AND     AL,11110000B    ; BITS 4-7 OFF?
0ACC  75 28                     JNE     AT20            ; NO - ERROR
0ACE  E8 FE9A R                 CALL    RR1             ; READ INTR ID REG
0AD1  24 F8                     AND     AL,11111000B    ; BITS 3-7 OFF?
0AD3  75 21                     JNE     AT20            ; NO
0AD5  42                        INC     DX              ; LINE CTRL REG
0AD6  E8 FE9A R                 CALL    RR1             ; READ MODEM CTRL REG
0AD9  24 E0                     AND     AL,11100000B    ; BITS 5-7 OFF?
0ADB  75 19                     JNE     AT20            ; NO
0ADD  E8 FE9A R                 CALL    RR1             ; READ LINE STAT REG
0AE0  24 80                     AND     AL,10000000B    ; BIT 7 OFF?
0AE2  75 12                     JNE     AT20            ; NO
0AE4  B0 60                     MOV     AL,60H
0AE6  EE                        OUT     DX,AL
0AE7  EB 00                     JMP     $+2             ; I/O DELAY
0AE9  42                        INC     DX              ; MODEM STAT REG
0AEA  32 C0                     XOR     AL,AL
0AEC  EE                        OUT     DX,AL           ; WIRED BITS WILL BE HIGH
0AED  E8 FEA0 R                 CALL    RR3             ; CLEAR BITS 0-3 IN CASE THEY'RE ON
                                                        ; AFTER WRITING TO STATUS REG
0AF0  83 EA 06                  SUB     DX,6            ; RECEIVER BUFFER
0AF3  EC                        IN      AL,DX           ; IN CASE WRITING TO PORTS CAUSED
                                                        ; DATA READY TO GO HIGH!
0AF4  F8                        CLC
0AF5  C3                        RET
0AF6  F9                AT20:   STC                     ; ERROR RETURN
0AF7  C3                        RET
0AF8                    I8250   ENDP
;--------------------------------------------------------------------------
;   SUBROUTINE TO TEST A PARTICULAR 8250 INTERRUPT.  PASS IT THE
;       (BIT # + 1) OF THE STATUS REGISTER THAT IS TO BE TESTED.
;       THIS ROUTINE SETS THAT BIT AND CHECKS TO SEE IF THE CORRECT
;       8250 INTERRUPT IS GENERATED.
;   IT EXPECTS TO BE PASSED:
;       (AH) = BIT # TO BE TESTED
;       (BL) = INTERRUPT IDENTIFIER
;           (0) = RECEIVED DATA AVAILABLE OR TRANSMITTER HOLDING
;                 REGISTER EMPTY INTERRUPT TEST
;           (1) = RECEIVER LINE STATUS OR MODEM STATUS INTERRUPT
;                 TEST
;       (BH) = BITS WHICH DETERMINE WHICH INTERRUPT IS TO BE
;              CHECKED
;           (0) = MODEM STATUS
;           (2) = TRANSMITTER HOLDING REGISTER EMPTY
;           (4) = RECEIVED DATA AVAILABLE
;           (6) = RECEIVER LINE STATUS
;       (CX) = VALUE TO SUBTRACT AND ADD IN ORDER TO REFERENCE THE
;              INTERRUPT IDENTIFICATION REGISTER
;           (3) = RECEIVED DATA AVAILABLE, TRANSMITTER HOLDING
;                 REGISTER AND RECEIVER LINE STATUS INTERRUPTS
;           (4) = MODEM STATUS INTERRUPT
;       (DX) = ADDRESS OF THE LINE STATUS OR MODEM STATUS REGISTER
;              IT RETURNS:
;       (AL) = OFFH IF TEST FAILS - EITHER NO INTERRUPT OCCURRED OR
;              THE WRONG INTERRUPT OCCURRED
;    OR
;       (AL) = CONTENTS OF THE INTERRUPT ID REGISTER FOR RECEIVED
;          DATA AVAILABLE AND TRANSMITTER HOLDING REGISTER
;          EMPTY INTERRUPTS
;          -OR-
;          CONTENTS OF THE LINE STATUS OR MODEM STATUS REGISTER
;              DEPENDING ON WHICH ONE WAS TESTED.
;       (DX) = ADDRESS OF INTERRUPT ID REGISTER FOR RECEIVED DATA
;              AVAILABLE OR TRANSMITTER HOLDING REGISTER EMPTY
;              INTERRUPTS
;    OR
;       (DX) = ADDRESS OF THE LINE STATUS OR DATA SET STATUS
;              REGISTER (DEPENDING ON WHICH INTERRUPT WAS TESTED)
;   NO OTHER REGISTERS ARE ALTERED.
;--------------------------------------------------------------------------
```

```
0AF8                    ICT     PROC    NEAR
0AF8  EC                        IN      AL,DX           ; READ STATUS REGISTER
0AF9  EB 00                     JMP     $+2             ; I/O DELAY
0AFB  0A C4                     OR      AL,AH           ; SET TEST BIT
0AFD  EE                        OUT     DX,AL           ; WRITE IT TO THE STATUS REGISTER
0AFE  2B D1                     SUB     DX,CX           ; POINT TO INTERRUPT ID REGISTER
0B00  51                        PUSH    CX
0B01  2B C9                     SUB     CX,CX           ; WAIT FOR 8250 INTERRUPT TO OCCUR
0B03  EC                AT21:   IN      AL,DX           ; READ INTR ID REG
0B04  A8 01                     TEST    AL,1            ; INTERRUPT PENDING?
0B06  74 02                     JE      AT22            ; YES -RETURN W/ INTERRUPT ID IN AL
0B08  E2 F9                     LOOP    AT21            ; NO - TRY AGAIN
0B0A  59                AT22:   POP     CX              ; AL = 1 IF NO INTERRUPT OCCURRED
0B0B  3A C7                     CMP     AL,BH           ; INTERRUPT WE'RE LOOKING FOR?
0B0D  75 09                     JNE     AT23            ; NO
0B0F  0A DB                     OR      BL,BL           ; DONE WITH TEST FOR THIS INTERRUPT
0B11  74 07                     JE      AT24            ; RETURN W/ CONTENTS OF INTR ID REG
0B13  03 D1                     ADD     DX,CX           ; READ STATUS REGISTER TO CLEAR THE
0B15  EC                        IN      AL,DX           ; INTERRUPT (WHEN BL=1)
0B16  EB 02                     JMP     SHORT AT24      ; RETURN CONTENTS OF STATUS REG
0B18  B0 FF             AT23:   MOV     AL,0FFH         ; SET ERROR INDICATOR
0B1A  C3                AT24:   RET
0B1B                    ICT     ENDP
                        ;--- INT 19 -----------------------------------------------
                        ; BOOT STRAP LOADER
                        ;       TRACK 0, SECTOR 1 IS READ INTO THE
                        ;       BOOT LOCATION (SEGMENT 0, OFFSET 7C00)
                        ;       AND CONTROL IS TRANSFERRED THERE.
                        ;
                        ;       IF THE DISKETTE IS NOT PRESENT OR HAS A
                        ;       PROBLEM LOADING (E.G., NOT READY), AN INT.
                        ;       18H IS EXECUITED. IF A CARTRIDGE HAS VECTORED
                        ;       INT. 18H TO ITSELF, CONTROL WILL BE PASSED TO
                        ;       THE CARTRIDGE.
                        ;--------------------------------------------------------
                                ASSUME  CS:CODE,DS:ABS0
0B1B                    BOOT_STRAP      PROC    NEAR
0B1B  FB                        STI                     ; ENABLE INTERRUPTS
0B1C  2B C0                     SUB     AX,AX           ; SET 40X25 B&W MODE ON CRT
0B1E  CD 10                     INT     10H             ;
0B20  2B C0                     SUB     AX,AX           ; ESTABLISH ADDRESSING
0B22  8E D8                     MOV     DS,AX
                        ;----- SEE IF DISKETTE PRESENT
0B24  E4 62                     IN      AL,PORT_C       ; GET CONFIG BITS
0B26  24 04                     AND     AL,00000100B    ; IS DISKETTE PRESENT?
0B28  75 28                     JNZ     H3              ; NO, THEN ATTEMPT TO GO TO CART.
                        ;----- RESET THE DISK PARAMETER TABLE VECTOR
0B2A  C7 06 0078 R EFC7 R       MOV     WORD PTR DISK_POINTER, OFFSET DISK_BASE
0B30  8C 0E 007A R              MOV     WORD PTR DISK_POINTER+2,CS
                        ;-----  LOAD SYSTEM FROM DISKETTE -- CX HAS RETRY COUNT
0B34  B9 0004                   MOV     CX,4            ; SET RETRY COUNT
0B37  51                H1:     PUSH    CX              ; SAVE RETRY COUNT
0B38  B4 00                     MOV     AH,0            ; RESET THE DISKETTE SYSTEM
0B3A  CD 13                     INT     13H             ; DISKETTE_IO
0B3C  72 0F                     JC      H2              ; IF ERROR, TRY AGAIN
0B3E  B8 0201                   MOV     AX,201H         ; READ IN THE SINGLE SECTOR
0B41  2B D2                     SUB     DX,DX           ; TO THE BOOT LOCATION
0B43  8E C2                     MOV     ES,DX
0B45  BB 7C00 R                 MOV     BX,OFFSET BOOT_LOCN
                                                        ; DRIVE 0, HEAD 0
0B48  B9 0001                   MOV     CX,1            ; SECTOR 1, TRACK 0
0B4B  CD 13                     INT     13H             ; DISKETTE_IO
0B4D  59                H2:     POP     CX              ; RECOVER RETRY COUNT
0B4E  73 04                     JNC     H3A             ; CF SET BY UNSUCCESSFUL READ
0B50  E2 E5                     LOOP    H1              ; DO IT FOR RETRY TIMES
                        ;----- UNABLE TO IPL FROM THE DISKETTE
0B52  CD 18             H3:     INT     18H             ; GO TO BASIC OR CARTRIDGE
                        ;----- IPL WAS SUCCESSFUL
0B54  EA 7C00 ---- R    H3A:    JMP     BOOT_LOCN
0B59                    BOOT_STRAP      ENDP
                        ;--------------------------------------------------------
                        ; THIS ROUTINE PERFORMS A READ/WRITE TEST ON A BLOCK OF
                        ; STORAGE (MAX. SIZE = 32KB). IF "WARM START", FILL
                        ; BLOCK WITH 0000 AND RETURN.
                        ; DATA PATTERNS USED:
                        ;       0->FF ON ONE BYTE TO TEST DATA BUS
                        ;       AAAA,5555,00FF,FF00 FOR ALL WORDS
                        ;       FILL WITH 0000 BEFORE EXIT
                        ;
                        ; ON ENTRY:
                        ;   ES = ADDRESS OF STORAGE TO BE TESTED
                        ;   DS = ADDRESS OF STORAGE TO BE TESTED
                        ;   CX = WORD COUNT OF STORAGE BLOCK TO BE TESTED
                        ;           (MAX. = 8000H (32K WORDS))
                        ;
                        ; ON EXIT:
                        ;   ZERO FLAG = OFF IF STORAGE ERROR
                        ;   IF ZERO FLAG = OFF, THEN CX = XOR'ED BIT PATTERN
                        ;   OF THE EXPECTED DATA PATTERN VS. THE ACTUAL DATA
                        ;   READ. (I.E., A BIT "ON" IN AL IS THE BIT IN ERROR)
                        ;   AH=03 IF BOTH BYTES OF WORD HAVE ERRORS
                        ;   AH=02 IF LOW (EVEN) BYTE HAS ERROR
                        ;   AH=01 IF HI (ODD) BYTE HAS ERROR
                        ; AX,BX,CX,DX,DI,SI ARE ALL DESTROYED.
                        ;--------------------------------------------------------
```

```
0B59                            PODSTG  PROC    NEAR
                                        ASSUME  DS:ABS0
0B59  FC                                CLD                     ; SET DIRECTION TO INCREMENT
0B5A  2B FF                             SUB     DI,DI           ; SET DI=0000 REL. TO START OF SEG
0B5C  2B C0                             SUB     AX,AX           ; INITIAL DATA PATTERN FOR 00-FF
                                                                ; TEST
0B5E  8E D8                             MOV     DS,AX           ; SET DS TO ABS0
0B60  8B 1E 0472 R                      MOV     BX,DATA_WORD[RESET_FLAG-DATA] ; WARM START?
0B64  81 FB 1234                        CMP     BX,1234H
0B68  8C C2                             MOV     DX,ES
0B6A  8E DA                             MOV     DS,DX           ; RESTORE DS
0B6C  75 0B                             JNE     P1
0B6E  F3/ AB                    P12:    REP     STOSW           ; SIMPLE FILL WITH 0 ON WARM-START
0B70  8E D8                             MOV     DS,AX
0B72  89 1E 0472 R                      MOV     DATA_WORD[RESET_FLAG-DATA],BX
0B76  8E DA                             MOV     DS,DX           ; RESTORE DS
0B78  C3                                RET                     ; AND EXIT
0B79  81 FB 4321                P1:     CMP     BX,4321H        ; DIAG. RESTART?
0B7D  74 EF                             JE      P12             ; DO FILL WITH ZEROS
0B7F  88 05                     P2:     MOV     [DI],AL         ; WRITE TEST DATA
0B81  8A 05                             MOV     AL,[DI]         ; GET IT BACK
0B83  32 C4                             XOR     AL,AH           ; COMPARE TO EXPECTED
0B85  74 03                             JZ      PY
0B87  E9 0C0C R                         JMP     P8              ; ERROR EXIT IF MISCOMPARE
0B8A  FE C4                     PY:     INC     AH              ; FORM NEW DATA PATTERN
0B8C  8A C4                             MOV     AL,AH           ;
0B8E  75 EF                             JNZ     P2              ; LOOP TILL ALL 256 DATA PATTERNS
                                                                ;  DONE
0B90  8B E9                             MOV     BP,CX           ; SAVE WORD COUNT
0B92  B8 AAAA                           MOV     AX,0AAAAH       ; LOAD DATA PATTERN
0B95  8B D8                             MOV     BX,AX
0B97  BA 5555                           MOV     DX,05555H       ; LOAD OTHER DATA PATTERN
0B9A  F3/ AB                            REP     STOSW           ; FILL WORDS FROM LOW TO HIGH
                                                                ; WITH AAAA
0B9C  4F                                DEC     DI              ; POINT TO LAST WORD WRITTEN
0B9D  4F                                DEC     DI
0B9E  FD                                STD                     ; SET DIRECTION FLAG TO GO DOWN
0B9F  8B F7                             MOV     SI,DI           ; SET INDEX REGS. EQUAL
0BA1  8B CD                             MOV     CX,BP           ; RECOVER WORD COUNT
0BA3                            P3:                             ; GO FROM HIGH TO LOW
0BA3  AD                                LODSW                   ; GET WORD FROM MEMORY
0BA4  33 C3                             XOR     AX,BX           ; EQUAL WHAT S/B THERE?
0BA6  75 64                             JNZ     P8              ; GO ERROR EXIT IF NOT
0BA8  8B C2                             MOV     AX,DX           ; GET 55 DATA PATTERN
0BAA  AB                                STOSW                   ;  STORE IT IN LOCATION JUST READ
0BAB  E2 F6                             LOOP    P3              ; LOOP TILL ALL BYTES DONE
0BAD  8B CD                             MOV     CX,BP           ; RECOVER WORD COUNT
0BAF  FC                                CLD                     ; BACK TO INCREMENT
0BB0  46                                INC     SI              ; ADJUST PTRS
0BB1  46                                INC     SI
0BB2  8B FE                             MOV     DI,SI
0BB4  8B DA                             MOV     BX,DX           ; S/B DATA PATTERN TO BX
0BB6  BA 00FF                           MOV     DX,00FFH        ; DATA FOR CHECKERBOARD PATTERN
0BB9  AD                        PX:     LODSW                   ; GET WORD FROM MEMORY
0BBA  33 C3                             XOR     AX,BX           ; EQUAL WHAT S/B THERE?
0BBC  75 4E                             JNZ     P8              ; GO ERROR EXIT IF NOT
0BBE  8B C2                             MOV     AX,DX           ; GET OTHER PATTERN
0BC0  AB                                STOSW                   ; STORE IT IN LOCATION JUST READ
0BC1  E2 F6                             LOOP    PX              ; LOOP TILL ALL BYTES DONE
0BC3  8B CD                             MOV     CX,BP           ; RECOVER WORD COUNT
0BC5  FD                                STD                     ; DECREMENT
0BC6  4E                                DEC     SI              ; ADJUST PTRS
0BC7  4E                                DEC     SI
0BC8  8B FE                             MOV     DI,SI
0BCA  8B DA                             MOV     BX,DX           ; S/B DATA PATTERN TO BX
0BCC  F7 D2                             NOT     DX              ; MAKE PATTERN FF00
0BCE  0A D2                             OR      DL,DL           ; FIRST PASS?
0BD0  74 E7                             JZ      PX
0BD2  FC                                CLD                     ; INCREMENT
0BD3  83 C6 04                          ADD     SI,4
0BD6  F7 D2                             NOT     DX
0BD8  8B FE                             MOV     DI,SI
0BDA  8B CD                             MOV     CX,BP
0BDC                            P4:                             ; LOW TO HIGH
0BDC  AD                                LODSW                   ; GET A WORD
0BDD  33 C2                             XOR     AX,DX           ; SHOULD COMPARE TO DX
0BDF  75 28                             JNZ     P8              ; GO ERROR IF NOT
0BE1  AB                                STOSW                   ; WRITE 0000 BACK TO LOCATION
                                                                ; JUST READ
0BE2  E2 F8                             LOOP    P4              ; LOOP TILL DONE
0BE4  FD                                STD                     ; BACK TO DECREMENT
0BE5  4E                                DEC     SI              ; ADJUST POINTER DOWN TO LAST WORD
0BE6  4E                                DEC     SI              ; WRITTEN
                        ; CHECK IF IN SERVICE/MFG MODES, IF SO, PERFORM REFRESH CHECK
0BE7  BA 0201                           MOV     DX,201H         ;
0BEA  EC                                IN      AL,DX           ; GET OPTION BITS
0BEB  24 F0                             AND     AL,0F0H         ;
0BED  3C F0                             CMP     AL,0F0H         ; ALL BITS HIGH=NORMAL MODE
0BEF  74 10                             JE      P6
0BF1  8C C9                             MOV     CX,CS
0BF3  8C D3                             MOV     BX,SS
0BF5  3B CB                             CMP     CX,BX           ; SEE IF IN PRE-STACK MODE
0BF7  74 08                             JE      P6              ; BYPASS RETENTION TEST IF SO
0BF9  B0 18                             MOV     AL,24           ; SET OUTER LOOP COUNT
                        ; WAIT ABOUT 6-8 SECONDS WITHOUT ACCESSING MEMORY
                        ; IF REFRESH IS NOT WORKING PROPERLY, THIS SHOULD
                        ; BE ENOUGH TIME FOR SOME DATA TO GO SOUR.
```

```
0BFB  E2 FE                     P5:     LOOP    P5
0BFD  FE C8                             DEC     AL
0BFF  75 FA                             JNZ     P5
0C01  8B CD                     P6:     MOV     CX,BP           ; RECOVER WORD COUNT
0C03  AD                        P7:     LODSW                   ; GET WORD
0C04  0B C0                             OR      AX,AX           ; = TO 0000
0C06  75 04                             JNZ     P8              ; ERROR IF NOT
0C08  E2 F9                             LOOP    P7              ; LOOP TILL DONE
0C0A  EB 13                             JMP     SHORT P11       ; THEN EXIT
0C0C  8B C8                     P8:     MOV     CX,AX           ; SAVE BITS IN ERROR
0C0E  32 E4                             XOR     AH,AH
0C10  0A ED                             OR      CH,CH           ; HIGH BYTE ERROR?
0C12  74 02                             JZ      P9
0C14  FE C4                             INC     AH              ; SET HIGH BYTE ERROR
0C16  0A C9                     P9:     OR      CL,CL           ; LOW BYTE ERROR?
0C18  74 03                             JZ      P10
0C1A  80 C4 02                          ADD     AH,2
0C1D  0A E4                     P10:    OR      AH,AH           ; SET ZERO FLAG=0 (ERROR INDICATION
0C1F  FC                        P11:    CLD                     ; SET DIR FLAG BACK TO INCREMENT
0C20  C3                                RET                     ; RETURN TO CALLER
0C21                            PODSTG  ENDP
                                ;--------------------------------------------------------
                                ; PUT_LOGO PROCEDURE
                                ;       THIS PROC SETS UP POINTERS AND CALLS THE SCREEN
                                ;    OUTPUT ROUTINE SO THAT THE IBM LOGO, A MESSAGE,
                                ;    AND A COLOR BAR ARE PUT UP ON THE SCREEN.
                                ;    AX,BX, AND DX ARE DESTROYED. ALL OTHERS ARE SAVED
                                ;--------------------------------------------------------
0C21                            PUT_LOGO PROC   NEAR
0C21  1E                                PUSH    DS
0C22  55                                PUSH    BP
0C23  50                                PUSH    AX
0C24  53                                PUSH    BX
0C25  51                                PUSH    CX
0C26  52                                PUSH    DX
0C27  BD 0C4A R                         MOV     BP,OFFSET LOGO  ;
0C2A  BA 8000                           MOV     DX,8000H        ;POINT DH DL AT ROW,COLUMN 0,0
0C2D  B3 1F                             MOV     BL,00011111B    ;ATTRIBUTE OF CHARACTERS TO BE
                                                                ;WRITTEN
0C2F  CD 82                             INT     82H             ;CALL OUTPUT ROUTINE
0C31  B3 00                             MOV     BL,00000000B    ; INITIALIZE ATTRIBUTE
0C33  B2 00                             MOV     DL,0            ; INITIALIZE COLUMN
0C35  B6 94                     AGAIN:  MOV     DH,94H          ; SET LINE
0C37  BD 0CDD R                         MOV     BP,OFFSET COLOR ; OUTPUT GIVEN COLOR BAR
0C3A  CD 82                             INT     82H             ;CALL OUTPUT ROUTINE
0C3C  FE C3                             INC     BL              ; INCREMENT ATTRIBUTE
0C3E  80 FA 20                          CMP     DL,32           ; IS THE COLUMN COUNTER POINTING
                                                                ; PAST 40?
0C41  7C F2                             JL      AGAIN           ; IF NOT, DO IT AGAIN
0C43  5A                                POP     DX
0C44  59                                POP     CX
0C45  5B                                POP     BX
0C46  58                                POP     AX
0C47  5D                                POP     BP              ; RESTORE BP
0C48  1F                                POP     DS              ; RESTORE DS
0C49  C3                                RET
0C4A                            PUT_LOGO ENDP
0C4A  03                        LOGO    DB      LOGO_E - LOGO
0C4B  20 DC                             DB      ' ',220
= 0C4D                          LOGO_E  =       $
0C4D  28 FB                             DB      40,-5
0C4F  28 FB                             DB      40,-5
0C51  02 07 01 09 03 04                 DB      2,7,1,9,3,4,9,4,1,-5
      09 04 01 FB
0C5B  02 07 01 0A 02 05                 DB      2,7,1,10,2,5,7,5,1,-5
      07 05 01 FB
0C65  02 07 01 0B 01 06                 DB      2,7,1,11,1,6,5,6,1,-5
      05 06 01 FB
0C6F  04 03 05 03 03 03                 DB      4,3,5,3,3,3,3,5,3,5,3,-5
      03 05 03 05 03 FB
0C7B  04 03 05 03 03 03                 DB      4,3,5,3,3,3,3,6,1,6,3,-5
      03 06 01 06 03 FB
0C87  04 03 05 08 04 0D                 DB      4,3,5,8,4,13,3,-5
      03 FB
0C8F  04 03 05 07 05 0D                 DB      4,3,5,7,5,13,3,-5
      03 FB
0C97  04 03 05 08 04 0D                 DB      4,3,5,8,4,13,3,-5
      03 FB
0C9F  04 03 05 03 03 03                 DB      4,3,5,3,3,3,3,13,3,-5
      03 0D 03 FB
0CA9  04 03 05 03 03 03                 DB      4,3,5,3,3,3,3,3,1,5,1,3,3,-5
      03 03 01 05 01 03
      03 FB
0CB7  02 07 01 0B 01 05                 DB      2,7,1,11,1,5,2,3,2,5,1,-5
      02 03 02 05 01 FB
0CC3  02 07 01 0A 02 05                 DB      2,7,1,10,2,5,3,1,3,5,1,-5
      03 01 03 05 01 FB
0CCF  02 07 01 09 03 05                 DB      2,7,1,9,3,5,7,5,1,-5
      07 05 01 FB
0CD9  28 FB                             DB      40,-5
0CDB  28 FC                             DB      40,-4
0CDD  02                        COLOR   DB      COLOR_E - COLOR
0CDE  DB                                DB      219
= 0CDF                          COLOR_E =       $
0CDF  02 77 02 77 02 77                 DB      2,121-2,2,121-2,2,121-2,2,121-2,2,-4
      02 77 02 FC

                                        ASSUME  DS:DATA
```

```
;--- 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 4 COLOR
;               (AL)=5  320X200 BW 4 SHADES
;               (AL)=6  640X200 BW 2 SHADES
;               (AL)=7  NOT VALID
;        **** EXTENDED MODES ***
;               (AL)=8  160X200 16 COLOR
;               (AL)=9  320X200 16 COLOR
;               (AL)=A  640X200 4 COLOR
;               *** NOTE BW MODES OPERATE SAME AS COLOR MODES, BUT
;                       COLOR BURST IS NOT ENABLED
;               *** NOTE IF HIGH ORDER BIT IN AL IS SET, THE REGEN
;                       BUFFER IS NOT CLEARED.
;       (AH)=1  SET CURSOR TYPE
;               (CH) =  BITS 4-0 = START LINE FOR CURSOR
;               ** HARDWARE WILL ALWAYS CAUSE BLINK
;               ** SETTING BIT 5 OR 6 WILL CAUSE ERRATIC
;                  BLINKING OR NO CURSOR AT ALL
;               ** IN GRAPHICS MODES, BIT 5 IS FORCED ON TO
;                  DISABLE THE CURSOR
;               (CL) =  BITS 4-0 = END LINE FOR CURSOR
;       (AH)=2  SET CURSOR POSITION
;               (DH,DL) = ROW,COLUMN  (0,0) IS UPPER LEFT
;               (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
;       (AH)=3  READ CURSOR POSITION
;               (BH) = PAGE NUMBER (MUST BE 0 FOR GRAPHICS MODES)
;               ON EXIT (DH,DL) = ROW,COLUMN OF CURRENT CURSOR
;                       (CH,CL) = CURSOR MODE CURRENTLY SET
;       (AH)=4  READ LIGHT PEN POSITION
;               ON EXIT:
;            (AH) = 0 -- LIGHT PEN SWITCH NOT DOWN/NOT TRIGGERED
;            (AH) = 1 -- VALID LIGHT PEN VALUE IN REGISTERS
;                       (DH,DL) = ROW,COLUMN OF CHARACTER LP POSN
;                       (CH) = RASTER LINE (0-199)
;                       (BX) = PIXEL COLUMN (0-319,639)
;       (AH)=5  SELECT ACTIVE DISPLAY PAGE (VALID ONLY FOR
;                       ALPHA MODES)
;               (AL)=NEW PAGE VALUE (0-7 FOR MODES 0&1, 0-3 FOR
;                       MODES 2&3)
;               IF BIT 7 (80H) OF AL=1
;                       READ/WRITE CRT/CPU PAGE REGISTERS
;               (AL) = 80H READ CRT/CPU PAGE REGISTERS
;               (AL) = 81H SET CPU PAGE REGISTER
;                       (BL) = VALUE TO SET
;               (AL) = 82H SET CRT PAGE REGISTER
;                       (BH) = VALUE TO SET
;               (AL) = 83H SET BOTH CRT AND CPU PAGE REGISTERS
;                       (BL) = VALUE TO SET IN CPU PAGE REGISTER
;                       (BH) = VALUE TO SET IN CRT PAGE REGISTER
;               IF BIT 7 (80H) OF AL=1
;                  ALWAYS  RETURNS (BH) = CONTENTS OF CRT PAGE REG
;                                  (BL) = CONTENTS OF CPU PAGE REG
;       (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
;                      CHARACTER (GRAPHICS). SEE NOTE ON WRITE
;                      DOT FOR BIT 7 OF BL = 1.
;       (AH) = 10 (0AH) 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
;               (BL) = COLOR OF CHAR (GRAPHICS)
;                       SEE NOTE ON WRITE DOT FOR BIT 7 OF BL = 1.
```

```
;       FOR READ/WRITE CHARACTER INTERFACE WHILE IN GRAPHICS MODE,
;               THE CHARACTERS ARE FORMED FROM A CHARACTER
;               GENERATOR IMAGE MAINTAINED IN THE SYSTEM ROM.
;               INTERRUPT 44H (LOCATION 00110H) IS USED TO
;               POINT TO THE 1K BYTE TABLE CONTAINING THE
;               FIRST 128 CHARS (0-127).
;               INTERRUPT 1FH (LOCATION 0007CH) IS USED TO
;               POINT TO THE 1K BYTE TABLE CONTAINING 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 (0BH) SET COLOR PALETTE
;               (BH) = PALETTE COLOR ID BEING SET (0-127)
;               (BL) = COLOR VALUE TO BE USED WITH THAT COLOR ID
;                       COLOR ID = 0 SELECTS THE BACKGROUND
;                                    COLOR (0-15)
;                       COLOR ID = 1 SELECTS THE PALETTE TO BE
;                          USED:
;                          2 COLOR MODE:
;                               0 = WHITE FOR COLOR 1
;                               1 = BLACK FOR COLOR 1
;                          4 COLOR MODES:
;                               0 = GREEN, RED, BROWN FOR
;                                   COLORS 1,2,3
;                               1 = CYAN, MAGENTA, WHITE FOR
;                                   COLORS 1,2,3
;                          16 COLOR MODES:
;                               ALWAYS SETS UP PALETTE AS:
;                               BLUE FOR COLOR 1
;                               GREEN FOR COLOR 2
;                               CYAN FOR COLOR 3
;                               RED FOR COLOR 4
;                               MAGENTA FOR COLOR 5
;                               BROWN FOR COLOR 6
;                               LIGHT GRAY FOR COLOR 7
;                               DARK GRAY FOR COLOR 8
;                               LIGHT BLUE FOR COLOR 9
;                               LIGHT GREEN FOR COLOR 10
;                               LIGHT CYAN FOR COLOR 11
;                               LIGHT RED FOR COLOR 12
;                               LIGHT MAGENTA FOR COLOR 13
;                               YELLOW FOR COLOR 14
;                               WHITE FOR COLOR 15
;                  IN 40X25 OR 80X25 ALPHA MODES, THE VALUE SET
;                       FOR PALETTE COLOR 0 INDICATES THE BORDER
;                       COLOR TO BE USED. IN GRAPHIC MODES, IT
;                       INDICATES THE BORDER COLOR AND THE
;                       BACKGROUND COLOR.
;       (AH) = 12 (0CH) 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 (0DH) READ DOT
;               (DX) = ROW NUMBER
;               (CX) = COLUMN NUMBER
;               (AL) RETURNS THE DOT READ
;
; ASCII TELETYPE ROUTINE FOR OUTPUT
;       (AH) = 14 (0EH) 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 (0FH) 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
;       (AH) = 16 (10H) SET PALETTE REGISTERS
;               (AL) = 0 SET PALETTE REGISTER
;                        (BL) = PALETTE REGISTER TO SET (00H - 0FH)
;                        (BH) = VALUE TO SET
;               (AL) = 1 SET BORDER COLOR REGISTER
;                        (BH) = VALUE TO SET
;               (AL) = 2 SET ALL PALETTE REGISTERS AND BORDER
;                        REGISTER
;                       ES:DX POINTS TO A 17 BYTE LIST
;                       BYTES 0 THRU 15 ARE VALUES FOR PALETTE
;                               REGISTERS 0 THRU 15
;                       BYTE 16 IS THE VALUE FOR THE BORDER
;                               REGISTER
;       NOTE:
;       IN MODES USING A 32K REGEN (9 AND A), ACCESS THROUGH THE CPU
;       REGISTER BY USE OF B800H SEGMENT VALUE ONLY REACHES THE
;       FIRST 16K. BIOS USES THE CONTENTS OF THE CPU PAGE REG
;       (BITS 3,4, & 5 OF PAGDAT IN BIOS DATA AREA) TO DERIVE THE
;       PROPER SEGMENT VALUE.
;
;       CS,SS,DS,ES,BX,CX,DX PRESERVED DURING CALL
;       ALL OTHERS DESTROYED
```

```
                        ;--------------------------------------------------
                        ; VIDEO GATE ARRAY REGISTERS
                        ;
                        ;  PORT 3DA OUTPUT
                        ;       REG 0   MODE CONTROL 1 REGISTER
                        ;       01H     +HI BANDWIDTH/-LOW BANDWIDTH
                        ;       02H     +GRAPHICS/-ALPHA
                        ;       04H     +B&W
                        ;       08H     +VIDEO ENABLE
                        ;       10H     +16 COLOR GRAPHICS
                        ;
                        ;       REG 1   PALETTE MASK REISTER
                        ;       01H     PALETTE MASK 0
                        ;       02H     PALETTE MASK 1
                        ;       04H     PALETTE MASK 2
                        ;       08H     PALETTE MASK 3
                        ;
                        ;       REG 2   BORDER COLOR REGISTER
                        ;       01H     BLUE
                        ;       02H     GREEN
                        ;       04H     RED
                        ;       08H     INTENSITY
                        ;
                        ;       REG 3   MODE CONTROL 2 REGISTER
                        ;       01H     RESERVED -- MUST BE ZERO
                        ;       02H     +ENABLE BLINK
                        ;       04H     RESERVED -- MUST BE ZERO
                        ;       08H     +2 COLOR GRAPHICS (640X200 2 COLOR ONLY)
                        ;
                        ;       REG 4   RESET REGISTER
                        ;       01H     +ASYNCHRONOUS RESET
                        ;       02H     +SYNCHRONOUS RESET
                        ;
                        ;       REGS 10 TO 1F   PALETTE REGISTERS
                        ;       01H     BLUE
                        ;       02H     GREEN
                        ;       04H     RED
                        ;       08H     INTENSITY
                        ;
                        ;  VIDEO GATE ARRAY STATUS
                        ;    PORT 3DA INPUT
                        ;       01H     +DISPLAY ENABLE
                        ;       02H     +LIGHT PEN TRIGGER SET
                        ;       04H     -LIGHT PEN SWITCH MADE
                        ;       08H     +VERTICAL RETRACE
                        ;       10H     +VIDEO DOTS
                        ;
                                ASSUME  CS:CODE,DS:DATA,ES:VIDEO_RAM
OCE9                    M0010   LABEL   WORD    ;  TABLE OF ROUTINES WITHIN VIDEO I/O
OCE9  0DA5 R                    DW      OFFSET  SET_MODE
OCEB  E45E R                    DW      OFFSET  SET_CTYPE
OCED  E486 R                    DW      OFFSET  SET_CPOS
OCEF  E520 R                    DW      OFFSET  READ_CURSOR
OCF1  F751 R                    DW      OFFSET  READ_LPEN
OCF3  E4B3 R                    DW      OFFSET  ACT_DISP_PAGE
OCF5  E5D3 R                    DW      OFFSET  SCROLL_UP
OCF7  E63F R                    DW      OFFSET  SCROLL_DOWN
OCF9  F0E4 R                    DW      OFFSET  READ_AC_CURRENT
OCFB  F113 R                    DW      OFFSET  WRITE_AC_CURRENT
OCFD  F12C R                    DW      OFFSET  WRITE_C_CURRENT
OCFF  E543 R                    DW      OFFSET  SET_COLOR
OD01  F187 R                    DW      OFFSET  WRITE_DOT
OD03  F146 R                    DW      OFFSET  READ_DOT
OD05  1992 R                    DW      OFFSET  WRITE_TTY
OD07  E581 R                    DW      OFFSET  VIDEO_STATE
OD09  E685 R                    DW      OFFSET  SET_PALLETTE
= 0022                  M0010L  EQU     $-M0010

OD0B                    VIDEO_IO        PROC    NEAR
OD0B  FB                        STI                     ; INTERRUPTS BACK ON
OD0C  FC                        CLD                     ; SET DIRECTION FORWARD
OD0D  06                        PUSH    ES
OD0E  1E                        PUSH    DS              ; SAVE SEGMENT REGISTERS
OD0F  52                        PUSH    DX
OD10  51                        PUSH    CX
OD11  53                        PUSH    BX
OD12  56                        PUSH    SI
OD13  57                        PUSH    DI
OD14  50                        PUSH    AX              ; SAVE AX VALUE
OD15  8A C4                     MOV     AL,AH           ; GET INTO LOW BYTE
OD17  32 E4                     XOR     AH,AH           ; ZERO TO HIGH BYTE
OD19  D1 E0                     SAL     AX,1            ; #2 FOR TABLE LOOKUP
OD1B  8B F0                     MOV     SI,AX           ; PUT INTO SI FOR BRANCH
OD1D  3D 0022                   CMP     AX,M0010L       ; TEST FOR WITHIN RANGE
OD20  72 04                     JB      C1              ; BRANCH AROUND BRANCH
OD22  58                        POP     AX              ; THROW AWAY THE PARAMETER
OD23  E9 0F70 R                 JMP     VIDEO_RETURN    ; DO NOTHING IF NOT IN RANGE
OD26  E8 138B R         C1:     CALL    DDS
OD29  B8 B800                   MOV     AX,0B800H       ; SEGMENT FOR COLOR CARD
OD2C  80 3E 0049 R 09           CMP     CRT_MODE,9      ; IN MODE USING 32K REGEN
OD31  72 09                     JC      C2              ; NO,JUMP
OD33  8A 26 008A R              MOV     AH,PAGDAT       ; GET COPY OF PAGE REGS
OD37  80 E4 38                  AND     AH,CPUREG       ; ISOLATE CPU REG
OD3A  D0 EC                     SHR     AH,1            ; SHIFT TO MAKE INTO SEGMENT VALUE
OD3C  8E C0             C2:     MOV     ES,AX           ; SET UP TO POINT AT VIDEO RAM AREA
OD3E  58                        POP     AX              ; RECOVER VALUE
OD3F  8A 26 0049 R              MOV     AH,CRT_MODE     ; GET CURRENT MODE INTO AH
OD43  2E: FF A4 OCE9 R          JMP     WORD PTR CS:[SI+OFFSET M0010]
OD48                    VIDEO_IO        ENDP
```

```
                                ;-----------------------------------------
                                ; SET_MODE
                                ;       THIS ROUTINE INITIALIZES THE ATTACHMENT TO
                                ;       THE SELECTED MODE.  THE SCREEN IS BLANKED.
                                ; INPUT
                                ;       (AL) = MODE SELECTED (RANGE 0-B)
                                ; OUTPUT
                                ;       NONE
                                ;-----------------------------------------
0D48                            M0050   LABEL   WORD                    ; TABLE OF REGEN LENGTHS
0D48  0800                              DW      2048                    ; MODE 0 40X25  BW
0D4A  0800                              DW      2048                    ; MODE 1 40X25  COLOR
0D4C  1000                              DW      4096                    ; MODE 2 80X25  BW
0D4E  1000                              DW      4096                    ; MODE 3 80X25  COLOR
0D50  4000                              DW      16384                   ; MODE 4 320X200 4 COLOR
0D52  4000                              DW      16384                   ; MODE 5 320X200 4 COLOR
0D54  4000                              DW      16384                   ; MODE 6 640X200 BW
0D56  0000                              DW      0                       ; MODE 7 INVALID
0D58  4000                              DW      16384                   ; MODE 8 160X200 16 COLOR
0D5A  8000                              DW      32768                   ; MODE 9 320X200 16 COLOR
0D5C  8000                              DW      32768                   ; MODE A 640X200 4 COLOR
                                ;------ COLUMNS
0D5E                            M0060   LABEL   BYTE
0D5E  28 28 50 50 28 28                 DB      40,40,80,80,40,40,80,0,20,40,80
      50 00 14 28 50

                                ;------- TABLE OF GATE ARRAY PARAMATERS FOR MODE SETTING
0D69                            M0070   LABEL   BYTE
                                ;------ SET UP FOR 40X25 BW      MODE 0
0D69  0C 0F 00 02                       DB      0CH,0FH,0,2     ; GATE ARRAY PARMS
= 0004                          M0070L  EQU     $-M0070
                                ;------ SET UP FOR 40X25 COLOR  MODE 1
0D6D  08 0F 00 02                       DB      08H,0FH,0,2     ; GATE ARRAY PARMS
                                ;------ SET UP FOR 80X25 BW     MODE 2
0D71  0D 0F 00 02                       DB      0DH,0FH,0,2     ; GATE ARRAY PARMS
                                ;------ SET UP FOR 80X25 COLOR  MODE 3
0D75  09 0F 00 02                       DB      09H,0FH,0,2     ; GATE ARRAY PARMS
                                ;------ SET UP FOR 320X200  4 COLOR     MODE 4
0D79  0A 03 00 00                       DB      0AH,03H,0,0     ; GATE ARRAY PARMS
                                ;------ SET UP FOR 320X200  BW          MODE 5
0D7D  0E 03 00 00                       DB      0EH,03H,0,0     ; GATE ARRAY PARMS
                                ;------ SET UP FOR 640X200  BW          MODE 6
0D81  0E 01 00 08                       DB      0EH,01H,0,8     ; GATE ARRAY PARMS
                                ;------ SET UP FOR    INVALID           MODE 7
0D85  00 00 00 00                       DB      00H,00H,0,0     ; GATE ARRAY PARMS
                                ;------ SET UP FOR 160X200  16 COLOR    MODE 8
0D89  1A 0F 00 00                       DB      1AH,0FH,0,0     ; GATE ARRAY PARMS
                                ;------ SET UP FOR 320X200  16 COLOR    MODE 9
0D8D  1B 0F 00 00                       DB      1BH,0FH,0,0     ; GATE ARRAY PARMS
                                ;------ SET UP FOR 640X200   4 COLOR    MODE A
0D91  0B 03 00 00                       DB      0BH,03H,0,0     ; GATE ARRAY PARMS
                                ;-------------- TABLES OF PALETTE COLORS FOR 2 AND 4 COLOR MODES
                                ;-----  2 COLOR, SET 0
0D95                            M0072   LABEL   BYTE
0D95  00 0F 00 00                       DB      0,0FH,0,0
= 0004                          M0072L  EQU     $-M0072         ;ENTRY LENGTH
                                ;-----  2 COLOR, SET 1
0D99  0F 00 00 00                       DB      0FH,0,0,0
                                ;-----  4 COLOR, SET 0
0D9D                            M0074   LABEL   BYTE
0D9D  00 02 04 06                       DB      0,2,4,6
                                ;-----  4 COLOR, SET 1
0DA1                            M0075   LABEL   BYTE
0DA1  00 03 05 0F                       DB      0,3,5,0FH
0DA5                            SET_MODE        PROC    NEAR
0DA5  50                                PUSH    AX              ;SAVE INPUT MODE ON STACK
0DA6  24 7F                             AND     AL,7FH          ;REMOVE CLEAR REGEN SWITCH
0DA8  3C 07                             CMP     AL,7            ;CHECK FOR VALID MODES
0DAA  74 04                             JE      C3              ;MODE 7 IS INVALID
0DAC  3C 0B                             CMP     AL,0BH
0DAE  72 02                             JC      C4              ;GREATER THAN A IS INVALID
0DB0  B0 00                     C3:     MOV     AL,0            ;DEFAULT TO MODE 0
0DB2  3C 02                     C4:     CMP     AL,2            ;CHECK FOR MODES NEEDING 128K
0DB4  74 08                             JE      C5
0DB6  3C 03                             CMP     AL,3
0DB8  74 04                             JE      C5
0DBA  3C 09                             CMP     AL,09H
0DBC  72 0A                             JC      C6
0DBE  81 3E 0015 R 0080         C5:     CMP     TRUE_MEM,128    ;DO WE HAVE 128K?
0DC4  73 02                             JNC     C6              ;YES, JUMP
0DC6  B0 00                             MOV     AL,0            ;NO, DEFAULT TO MODE 0
0DC8  BA 03D4                   C6:     MOV     DX,03D4H        ; ADDRESS OF COLOR CARD
0DCB  8A E0                             MOV     AH,AL           ; SAVE MODE IN AH
0DCD  A2 0049 R                         MOV     CRT_MODE,AL     ; SAVE IN GLOBAL VARIABLE
0DD0  89 16 0063 R                      MOV     ADDR_6845,DX    ; SAVE ADDRESS OF BASE
0DD4  8B F8                             MOV     DI,AX           ; SAVE MODE IN DI
0DD6  BA 03DA                           MOV     DX,VGA_CTL      ; POINT TO CONTROL REGISTER
0DD9  EC                                IN      AL,DX           ; SYNC CONTROL REG TO ADDRESS
0DDA  32 C0                             XOR     AL,AL           ; SET VGA REG 0
0DDC  EE                                OUT     DX,AL           ; SELECT IT
0DDD  A0 0065 R                         MOV     AL,CRT_MODE_SET ; GET LAST MODE SET
0DE0  24 F7                             AND     AL,0F7H         ; TURN OFF VIDEO
0DE2  EE                                OUT     DX,AL           ; SET IN GATE ARRAY
```

```
                        ;----- SET DEFAULT PALETTES
0DE3  8B C7                     MOV     AX,DI           ; GET MODE
0DE5  B4 10                     MOV     AH,10H          ; SET PALETTE REG 0
0DE7  BB 0D95 R                 MOV     BX,OFFSET M0072 ; POINT TO TABLE ENTRY
0DEA  3C 06                     CMP     AL,6            ; 2 COLOR MODE?
0DEC  74 0F                     JE      C7              ; YES, JUMP
0DEE  BB 0DA1 R                 MOV     BX,OFFSET M0075 ; POINT TO TABLE ENTRY
0DF1  3C 05                     CMP     AL,5            ; CHECK FOR 4 COLOR MODE
0DF3  74 08                     JE      C7              ; YES, JUMP
0DF5  3C 04                     CMP     AL,4            ; CHECK FOR 4 COLOR MODE
0DF7  74 04                     JE      C7              ; YES JUMP
0DF9  3C 0A                     CMP     AL,0AH          ; CHECK FOR 4 COLOR MODE
0DFB  75 11                     JNE     C9              ; NO, JUMP
0DFD  B9 0004           C7:     MOV     CX,4            ; NUMBER OF REGS TO SET
0E00  8A C4             C8:     MOV     AL,AH           ; GET REG NUMBER
0E02  EE                        OUT     DX,AL           ; SELECT IT
0E03  2E: 8A 07                 MOV     AL,CS:[BX]      ; GET DATA
0E06  EE                        OUT     DX,AL           ; SET IT
0E07  FE C4                     INC     AH              ; NEXT REG
0E09  43                        INC     BX              ; NEXT TABLE VALUE
0E0A  E2 F4                     LOOP    C8
0E0C  EB 0B                     JMP     SHORT C11
                        ;----- SET PALETTES FOR DEFAULT 16 COLOR
0E0E  B9 0010           C9:     MOV     CX,16           ; NUMBER OF PALETTES, AH IS REG
                                                        ; COUNTER
0E11  8A C4             C10:    MOV     AL,AH           ; GET REG NUMBER
0E13  EE                        OUT     DX,AL           ; SELECT IT
0E14  EE                        OUT     DX,AL           ; SET PALETTE VALUE
0E15  FE C4                     INC     AH              ; NEXT REG
0E17  E2 F8                     LOOP    C10
                        ;----- SET UP M0 & M1 IN PAGREG
0E19  8B C7             C11:    MOV     AX,DI           ; GET CURRENT MODE
0E1B  32 DB                     XOR     BL,BL           ; SET UP FOR ALPHA MODE
0E1D  3C 04                     CMP     AL,4            ; IN ALPHA MODE
0E1F  72 08                     JC      C12             ; YES, JUMP
0E21  B3 40                     MOV     BL,40H          ; SET UP FOR 16K REGEN
0E23  3C 09                     CMP     AL,09H          ; MODE USE 16K
0E25  72 02                     JC      C12             ; YES, JUMP
0E27  B3 C0                     MOV     BL,0C0H         ; SET UP FOR 32K REGEN
0E29  BA 03DF           C12:    MOV     DX,PAGREG       ; SET PORT ADDRESS OF PAGREG
0E2C  A0 008A R                 MOV     AL,PAGDAT       ; GET LAST DATA OUTPUT
0E2F  24 3F                     AND     AL,3FH          ; CLEAR M0 & M1 BITS
0E31  0A C3                     OR      AL,BL           ; SET NEW BITS
0E33  EE                        OUT     DX,AL           ; STUFF BACK IN PORT
0E34  A2 008A R                 MOV     PAGDAT,AL       ; SAVE COPY IN RAM
                        ;----- ENABLE VIDEO AND CORRECT PORT SETTING
0E37  8B C7                     MOV     AX,DI           ; GET CURRENT MODE
0E39  32 E4                     XOR     AH,AH           ;   INTO AX REG
0E3B  B9 0004                   MOV     CX,M0070L       ; SET TABLE ENTRY LENGTH
0E3E  F7 E1                     MUL     CX              ; TIMES MODE FOR OFFSET INTO TABLE
0E40  8B D8                     MOV     BX,AX           ; TABLE OFFSET IN BX
0E42  81 C3 0D69 R              ADD     BX,OFFSET M0070 ; ADD TABLE START TO OFFSET
0E46  2E: 8A 27                 MOV     AH,CS:[BX]      ; SAVE MODE SET AND PALETTE
0E49  2E: 8A 47 02              MOV     AL,CS:[BX + 2]  ;   TILL WE CAN PUT THEM IN RAM
0E4D  8B F0                     MOV     SI,AX
0E4F  FA                        CLI                     ; DISABLE INTERRUPTS
0E50  E8 E675 R                 CALL    MODE_ALIVE      ; KEEP MEMORY DATA VALID
0E53  B0 10                     MOV     AL,10H          ; DISABLE NMI AND HOLD REQUEST
0E55  E6 A0                     OUT     NMI_PORT,AL     ;
0E57  BA 03DA                   MOV     DX,VGA_CTL      ;
0E5A  B0 04                     MOV     AL,4            ; POINT TO RESET REG
0E5C  EE                        OUT     DX,AL           ; SEND TO GATE ARRAY
0E5D  B0 02                     MOV     AL,2            ; SET SYNCHRONOUS RESET
0E5F  EE                        OUT     DX,AL           ; DO IT
                        ; WHILE THE GATE ARRAY IS IN RESET STATE, WE CANNOT ACCESS RAM
0E60  8B C6                     MOV     AX,SI           ; RESTORE NEW MODE SET
0E62  80 E4 F7                  AND     AH,0F7H         ; TURN OFF VIDEO ENABLE
0E65  32 C0                     XOR     AL,AL           ; SET UP TO SELECT VGA REG 0
0E67  EE                        OUT     DX,AL           ; SELECT IT
0E68  86 E0                     XCHG    AH,AL           ; AH IS VGA REG COUNTER
0E6A  EE                        OUT     DX,AL           ; SET MODE
0E6B  B0 04                     MOV     AL,4            ; SET UP TO SELECT VGA REG 4
0E6D  EE                        OUT     DX,AL           ; SELECT IT
0E6E  32 C0                     XOR     AL,AL           ;
0E70  EE                        OUT     DX,AL           ; REMOVE RESET FROM VGA
                        ; NOW OKAY TO ACCESS RAM AGAIN
0E71  B0 80                     MOV     AL,80H          ; ENABLE NMI AGAIN
0E73  E6 A0                     OUT     NMI_PORT,AL     ;
0E75  E8 E675 R                 CALL    MODE_ALIVE      ; KEEP MEMORY DATA VALID
0E78  FB                        STI                     ; ENABLE INTERRUPTS
0E79  EB 07                     JMP     SHORT C14
0E7B  8A C4             C13:    MOV     AL,AH           ; GET VGA REG NUMBER
0E7D  EE                        OUT     DX,AL           ; SELECT REG
0E7E  2E: 8A 07                 MOV     AL,CS:[BX]      ; GET TABLE VALUE
0E81  EE                        OUT     DX,AL           ; PUT IN VGA REG
0E82  43                C14:    INC     BX              ; NEXT IN TABLE
0E83  FE C4                     INC     AH              ; NEXT REG
0E85  E2 F4                     LOOP    C13             ; DO ENTIRE ENTRY
                        ;---- SET UP CRT AND CPU PAGE REGS ACCORDING TO MODE & MEMORY SIZE
0E87  8A 03DF                   MOV     DX,PAGREG       ; SET IO ADDRESS OF PAGREG
0E8A  A0 008A R                 MOV     AL,PAGDAT       ; GET LAST DATA OUTPUT
0E8D  24 C0                     AND     AL,0C0H         ; CLEAR REG BITS
0E8F  B3 36                     MOV     BL,36H          ; SET UP FOR GRAPHICS MODE WITH 32K
                                                        ; REGEN
0E91  A8 80                     TEST    AL,80H          ; IN THIS MODE?
0E93  75 0C                     JNZ     C15             ; YES, JUMP
0E95  B3 3F                     MOV     BL,3FH          ; SET UP FOR 16K REGEN AND 128K
                                                        ; MEMORY
0E97  81 3E 0015 R 0080         CMP     TRUE_MEM,128    ; DO WE HAVE 128K?
0E9D  73 02                     JNC     C15             ; YES, JUMP
0E9F  B3 1B                     MOV     BL,1BH          ; SET UP FOR 16K REGEN AND 64K
                                                        ; MEMORY
```

```
0EA1  0A C3                C15:    OR      AL,BL           ; COMBINE MODE BITS AND REG VALUES
0EA3  EE                           OUT     DX,AL           ; SET PORT
0EA4  A2 008A R                    MOV     PAGDAT,AL       ; SAVE COPY IN RAM
0EA7  8B C6                        MOV     AX,SI           ; PUT MODE SET & PALETTE IN RAM
0EA9  8B 26 0065 R                 MOV     CRT_MODE_SET,AH
0EAD  A2 0066 R                    MOV     CRT_PALLETTE,AL
0EB0  E4 61                        IN      AL,PORT_B       ; GET CURRENT VALUE OF 8255 PORT B
0EB2  24 FB                        AND     AL,0FBH         ; SET UP GRAPHICS MODE
0EB4  F6 C4 02                     TEST    AH,2            ; JUST SET ALPHA MODE IN VGA?
0EB7  75 02                        JNZ     C16             ; YES, JUMP
0EB9  0C 04                        OR      AL,4            ; SET UP ALPHA MODE
0EBB  E6 61                C16:    OUT     PORT_B,AL       ; STUFF BACK IN 8255
                           ;------- SET UP 6845
0EBD  1E                           PUSH    DS              ; SAVE DATA SEGMENT VALUE
0EBE  33 C0                        XOR     AX,AX           ; SET UP FOR ABSO SEGMENT
0EC0  8E D8                        MOV     DS,AX           ; ESTABLISH VECTOR TABLE ADDRESSING
                                   ASSUME  DS:ABS0
0EC2  C5 1E 0074 R                 LDS     BX,PARM_PTR     ; GET POINTER TO VIDEO PARMS
                                   ASSUME  DS:CODE
0EC6  8B C7                        MOV     AX,DI           ; GET CURRENT MODE IN AX
0EC8  B9 0010 90                   MOV     CX,M0040        ; LENGTH OF EACH ROW OF TABLE
0ECC  80 FC 02                     CMP     AH,2            ; DETERMINE WHICH TO USE
0ECF  72 10                        JC      C17             ; MODE IS 0 OR 1
0ED1  03 D9                        ADD     BX,CX           ; MOVE TO NEXT ROW OF INIT TABLE
0ED3  80 FC 04                     CMP     AH,4
0ED6  72 09                        JC      C17             ; MODE IS 2 OR 3
0ED8  03 D9                        ADD     BX,CX           ; MOVE TO GRAPHICS ROW OF
                                                           ; INIT_TABLE
0EDA  80 FC 09                     CMP     AH,9
0EDD  72 02                        JC      C17             ; MODE IS 4, 5, 6, 8, OR 9
0EDF  03 D9                        ADD     BX,CX           ; MOVE TO NEXT GRAPHICS ROW OF
                                                           ; INIT_TABLE
                           ;----- BX POINTS TO CORRECT ROW OF INITIALIZATION TABLE
0EE1  50                   C17:    PUSH    AX              ; SAVE MODE IN AH
0EE2  8A 47 02                     MOV     AL,DS:[BX+2]    ; GET HORZ. SYNC POSITION
0EE5  8B 7F 0A                     MOV     DI,WORD PTR DS:[BX+10]  ; GET CURSOR TYPE
0EE8  1E                           PUSH    DS
0EE9  E8 138B R                    CALL    DDS
                                   ASSUME  DS:DATA
0EEC  A2 0089 R                    MOV     HORZ_POS,AL     ; SAVE HORZ. SYNC POSITION VARIABLE
0EEF  89 3E 0060 R                 MOV     CURSOR_MODE,DI  ; SAVE CURSOR MODE
0EF3  50                           PUSH    AX
0EF4  A0 0086 R                    MOV     AL,VAR_DELAY    ; SET DEFAULT OFFSET
0EF7  24 0F                        AND     AL,0FH
0EF9  A2 0086 R                    MOV     VAR_DELAY,AL
0EFC  58                           POP     AX
                                   ASSUME  DS:CODE
0EFD  1F                           POP     DS
0EFE  32 E4                        XOR     AH,AH           ; AH WILL SERVE AS REGISTER NUMBER
                                                           ; DURING LOOP
0F00  BA 03D4                      MOV     DX,03D4H        ; POINT TO 6845
                           ;LOOP THROUGH TABLE, OUTPUTTING REG ADDRESS, THEN VALUE FROM TABLE
0F03  8A C4                C18:    MOV     AL,AH           ; GET 6845 REGISTER NUMBER
0F05  EE                           OUT     DX,AL
0F06  42                           INC     DX              ; POINT TO DATA PORT
0F07  FE C4                        INC     AH              ; NEXT REGISTER VALUE
0F09  8A 07                        MOV     AL,[BX]         ; GET TABLE VALUE
0F0B  EE                           OUT     DX,AL           ; OUT TO CHIP
0F0C  43                           INC     BX              ; NEXT IN TABLE
0F0D  4A                           DEC     DX              ; BACK TO POINTER REGISTER
0F0E  E2 F3                        LOOP    C18             ; DO THE WHOLE TABLE
0F10  58                           POP     AX              ; GET MODE BACK
0F11  1F                           POP     DS              ; RECOVER SEGMENT VALUE
                                   ASSUME  DS:DATA
                           ;------ FILL REGEN AREA WITH BLANK
0F12  33 FF                        XOR     DI,DI           ; SET UP POINTER FOR REGEN
0F14  89 3E 004E R                 MOV     CRT_START,DI    ; START ADDRESS SAVED IN GLOBAL
0F18  C6 06 0062 R 00              MOV     ACTIVE_PAGE,0   ; SET PAGE VALUE
0F1D  5A                           POP     DX              ; GET ORIGINAL INPUT BACK
0F1E  80 E2 80                     AND     DL,80H          ; NO CLEAR OF REGEN ?
0F21  75 1C                        JNZ     C21             ; SKIP CLEARING REGEN
0F23  BA B800                      MOV     DX,0B800H       ; SET UP SEGMENT FOR 16K REGEN AREA
0F26  B9 2000                      MOV     CX,8192         ; NUMBER OF WORDS TO CLEAR
0F29  3C 09                        CMP     AL,09H          ; REQUIRE 32K BYTE REGEN ?
0F2B  72 05                        JC      C19             ; NO, JUMP
0F2D  D1 E1                        SHL     CX,1            ; SET 16K WORDS TO CLEAR
0F2F  BA 1800                      MOV     DX,1800H        ; SET UP SEGMENT FOR 32K REGEN AREA
0F32  8E C2                C19:    MOV     ES,DX           ; SET REGEN SEGMENT
0F34  3C 04                        CMP     AL,4            ; TEST FOR GRAPHICS
0F36  B8 0F20                      MOV     AX,' '+15*256   ; FILL CHAR FOR ALPHA
0F39  72 02                        JC      C20             ; NO_GRAPHICS_INIT
0F3B  33 C0                        XOR     AX,AX           ; FILL FOR GRAPHICS MODE
0F3D  F3/ AB               C20:    REP     STOSW           ; FILL THE REGEN BUFFER WITH BLANKS
                           ;----- ENABLE VIDEO
0F3F  BA 03DA              C21:    MOV     DX,VGA_CTL      ; SET PORT ADDRESS OF VGA
0F42  32 C0                        XOR     AL,AL           ;
0F44  EE                           OUT     DX,AL           ; SELECT VGA REG 0
0F45  A0 0065 R                    MOV     AL,CRT_MODE_SET ; GET MODE SET VALUE
0F48  EE                           OUT     DX,AL           ; SET MODE
                           ;----- DETERMINE NUMBER OF COLUMNS, BOTH FOR ENTIRE DISPLAY
                           ;----- AND THE NUMBER TO BE USED FOR TTY INTERFACE
0F49  32 FF                        XOR     BH,BH
0F4B  8A 1E 0049 R                 MOV     BL,CRT_MODE
0F4F  2E: 8A 87 0D5E R             MOV     AL,CS:[BX + OFFSET M0060]
0F54  32 E4                        XOR     AH,AH
0F56  A3 004A R                    MOV     CRT_COLS,AX     ; NUMBER OF COLUMNS IN THIS SCREEN
```

```
                        ;----- SET CURSOR POSITIONS
0F59  D1 E3                     SHL     BX,1                    ; WORD OFFSET INTO CLEAR LENGTH
                                                                ; TABLE
0F5B  2E: 8B 8F 0048 R          MOV     CX,CS:[BX + OFFSET M0050]  ; LENGTH TO CLEAR
0F60  89 0E 004C R              MOV     CRT_LEN,CX              ; SAVE LENGTH OF CRT
0F64  B9 0008                   MOV     CX,8                    ; CLEAR ALL CURSOR POSITIONS
0F67  BF 0050 R                 MOV     DI,OFFSET CURSOR_POSN
0F6A  1E                        PUSH    DS                      ; ESTABLISH SEGMENT
0F6B  07                        POP     ES                      ;   ADDRESSING
0F6C  33 C0                     XOR     AX,AX
0F6E  F3/ AB                    REP     STOSW                   ; FILL WITH ZEROES
                        ;----- NORMAL RETURN FROM ALL VIDEO RETURNS
0F70                    VIDEO_RETURN:
0F70  5F                        POP     DI
0F71  5E                        POP     SI
0F72  5B                        POP     BX
0F73  59                C22:    POP     CX
0F74  5A                        POP     DX
0F75  1F                        POP     DS
0F76  07                        POP     ES                      ; RECOVER SEGMENTS
0F77  CF                        IRET                            ; ALL DONE
0F78                    SET_MODE        ENDP
                        ;--------------------------------------------------------
                        ;
                        ;
                        ; KBDNMI - KEYBOARD NMI INTERRUPT ROUTINE
                        ;
                        ;       THIS ROUTINE OBTAINS CONTROL UPON AN NMI INTERRUPT, WHICH
                        ;       OCCURS UPON A KEYSTROKE FROM THE KEYBOARD.
                        ;
                        ;       THIS ROUTINE WILL DE-SERIALIZE THE BIT STREAM IN ORDER TO
                        ;       GET THE KEYBOARD SCAN CODE ENTERED.  IT THEN ISSUES INT 41
                        ;       PASSING THE SCAN CODE IN AL TO THE KEY PROCESSOR. UPON RETURN
                        ;       IT RE-ENABLES NMI AND RETURNS TO SYSTEM (IRET).
                        ;
                        ;--------------------------------------------------------
                                ASSUME CS:CODE,DS:DATA
0F78                    KBDNMI  PROC    FAR
                        ;-----------DISABLE INTERRUPTS
0F78  FA                        CLI
                        ;----------SAVE REGS & DISABLE NMI
0F79  56                        PUSH    SI
0F7A  57                        PUSH    DI
0F7B  50                        PUSH    AX                      ; SAVE REGS
0F7C  53                        PUSH    BX
0F7D  51                        PUSH    CX
0F7E  52                        PUSH    DX
0F7F  1E                        PUSH    DS
0F80  06                        PUSH    ES
                        ;----------INIT COUNTERS
0F81  BE 0008                   MOV     SI,8                    ; SET UP # OF DATA BITS
0F84  32 DB                     XOR     BL,BL                   ; INIT. PARITY COUNTER
                        ;----------SAMPLE 5 TIMES TO  VALIDATE START BIT
0F86  32 E4                     XOR     AH,AH
0F88  B9 0005                   MOV     CX,5                    ; SET COUNTER
0F8B  E4 62             I1:     IN      AL,PORT_C               ; GET SAMPLE.
0F8D  A8 40                     TEST    AL,40H                  ; TEST IF 1
0F8F  74 02                     JZ      I2                      ; JMP IF 0
0F91  FE C4                     INC     AH                      ; KEEP COUNT OF 1'S
0F93  E2 F6             I2:     LOOP    I1                      ; KEEP SAMPLING
0F95  80 FC 03                  CMP     AH,3                    ; VALID START BIT ?
0F98  73 03                     JNB     I25                     ; JUMP IF OK
0F9A  EB 5D 90                  JMP     I8                      ; INVALID (SYNC ERROR) NO AUDIO
                                                                ; OUTPUT
                        ;----------VALID START BIT, LOOK FOR TRAILING EDGE
0F9D  B9 0032           I25:    MOV     CX,50                   ; SET UP WATCHDOG TIMEOUT
0FA0  E4 62             I3:     IN      AL,PORT_C               ; GET SAMPLE
0FA2  A8 40                     TEST    AL,40H                  ; TEST IF 0
0FA4  74 05                     JZ      I5                      ; JMP IF TRAILING EDGE FOUND
0FA6  E2 F8                     LOOP    I3                      ; KEEP LOOKING FOR TRAILING EDGE
0FA8  EB 4F 90                  JMP     I8                      ; SYNC ERROR (STUCK ON 1'S)
                        ;----------READ CLOCK TO SET START OF BIT TIME
0FAB  B0 40             I5:     MOV     AL,40H                  ; READ CLOCK
0FAD  E6 43                     OUT     TIM_CTL,AL              ; *
0FAF  90                        NOP                             ; *
0FB0  90                        NOP                             ; *
0FB1  E4 41                     IN      AL,TIMER+1              ; *
0FB3  8A E0                     MOV     AH,AL                   ; *
0FB5  E4 41                     IN      AL,TIMER+1              ; *
0FB7  86 E0                     XCHG    AH,AL                   ; *
0FB9  8B F8                     MOV     DI,AX                   ; SAVE CLOCK TIME IN DI
                        ;----------VERIFY VALID TRANSITION
0FBB  B9 0004                   MOV     CX,4                    ; SET COUNTER
0FBE  E4 62             I6:     IN      AL,PORT_C               ; GET SAMPLE
0FC0  A8 40                     TEST    AL,40H                  ; TEST IF 0
0FC2  75 35                     JNZ     I8                      ; JMP IF INVALID TRANSITION (SYNC)
0FC4  E2 F8                     LOOP    I6                      ; KEEP LOOKING FOR VALID TRANSITION
                        ;----------SET UP DISTANCE TO MIDDLE OF 1ST DATA BIT
0FC6  BA 0220                   MOV     DX,544                  ; 310 USEC AWAY (.838 US / CT)
                        ;-------START LOOKING FOR TIME TO READ DATA BITS AND ASSEMBLE BYTE.
0FC9  E8 1031 R         I7:     CALL    I30
0FCC  BA 020E                   MOV     DX,526                  ; SET NEW DISTANCE TO NEXT HALF BIT
0FCF  50                        PUSH    AX                      ; SAVE 1ST HALF BIT
0FD0  E8 1031 R                 CALL    I30
0FD3  8A C8                     MOV     CL,AL                   ; PUT 2ND HALF BIT IN CL
0FD5  58                        POP     AX                      ; RESTORE 1ST HALF BIT
0FD6  3A C8                     CMP     CL,AL                   ; ARE THEY OPPOSITES ?
0FD8  74 2A                     JE      I9                      ; NO, PHASE ERROR
```

```
0FDA  D0 EF                     ;-----------VALID DATA BIT, PLACE IN SCAN BYTE
0FDC  0A F8                             SHR     BH,1                    ; SHIFT PREVIOUS BITS
0FDE  4E                                OR      BH,AL                   ; OR IN NEW DATA BIT
0FDF  75 E8                             DEC     SI                      ; DECREMENT DATA BIT COUNTER
                                        JNZ     I7                      ; CONTINUE FOR MORE DATA BITS
                                ;-----------WAIT FOR TIME TO SAMPLE PARITY BIT
0FE1  E8 1031 R                         CALL    I30
0FE4  50                                PUSH    AX                      ; SAVE 1ST HALF BIT
0FE5  E8 1031 R                         CALL    I30
0FE8  8A C8                             MOV     CL,AL                   ; PUT 2ND HALF BIT IN CL
0FEA  58                                POP     AX                      ; RESTORE 1ST HALF BIT
0FEB  3A C8                             CMP     CL,AL                   ; ARE THEY OPPOSITES ?
0FED  74 15                             JE      I9                      ; NO, PHASE ERROR
                                ;-----------VALID PARITY BIT, CHECK PARITY
0FEF  80 E3 01                          AND     BL,1                    ; CHECK IF ODD PARITY
0FF2  74 10                             JZ      I9                      ; JMP IF PARITY ERROR
                                ;-----------VALID CHARACTER, SEND TO CHARACTER PROCESSING
0FF4  FB                                STI                             ; ENABLE INTERRUPTS
0FF5  8A C7                             MOV     AL,BH                   ; PLACE SCAN CODE IN AL
0FF7  CD 48                             INT     48H                     ; CHARACTER PROCESSING
                                ;-----------RESTORE REGS AND RE-ENABEL NMI
0FF9  07                        I8:     POP     ES                      ; RESTORE REGS
0FFA  1F                                POP     DS
0FFB  5A                                POP     DX
0FFC  59                                POP     CX
0FFD  5B                                POP     BX
0FFE  E4 A0                             IN      AL,0A0H                 ; ENABLE NMI
1000  58                                POP     AX
1001  5F                                POP     DI
1002  5E                                POP     SI
1003  CF                                IRET                            ; RETURN TO SYSTEM
                                ;-----------PARITY, SYNCH OR PHASE ERROR. OUTPUT MISSED KEY BEEP
1004  E8 138B R                 I9:     CALL    DDS                     ; SETUP ADDRESSING
1007  83 FE 08                          CMP     SI,8                    ; ARE WE ON THE FIRST DATA BIT?
100A  74 E0                             JE      I8                      ; NO AUDIO FEEDBACK (MIGHT BE A
                                                                        ;  GLITCH)
100C  F6 06 0018 R 01                   TEST    KB_FLAG_1,01H           ; CHECK IF TRANSMISSION ERRORS
                                                                        ;  ARE TO BE REPORTED
1011  75 18                             JNZ     I10                     ; 1=DO NOT BEEP, 0=BEEP
1013  BB 0080                          MOV     BX,080H                 ; DURATION OF ERROR BEEP
1016  B9 0048                          MOV     CX,048H                 ; FREQUENCY OF ERROR BEEP
1019  E8 E035 R                       CALL    KB_NOISE                ; AUDIO FEEDBACK
101C  80 26 0017 R F0                   AND     KB_FLAG,0F0H            ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                                        ; SHIFTS
1021  80 26 0018 R 0F                   AND     KB_FLAG_1,0FH           ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                                        ; NUM AND SCROLL SHIFT
1026  80 26 0088 R 1F                   AND     KB_FLAG_2,1FH           ; CLEAR FUNCTION STATES
102B  FE 06 0012 R              I10:    INC     KBD_ERR                 ; KEEP TRACK OF KEYBOARD ERRORS
102F  EB C8                             JMP     SHORT I8                ; RETURN FROM INTERRUPT
1031                            KBDNMI  ENDP
1031                            I30     PROC    NEAR
1031  B0 40                     I31:    MOV     AL,40H                  ; READ CLOCK
1033  E6 43                             OUT     TIM_CTL,AL              ; *
1035  90                                NOP                             ; *
1036  90                                NOP                             ; *
1037  E4 41                             IN      AL,TIMER+1              ; *
1039  8A E0                             MOV     AH,AL                   ; *
103B  E4 41                             IN      AL,TIMER+1              ; *
103D  86 E0                             XCHG    AH,AL                   ; *
103F  8B CF                             MOV     CX,DI                   ; GET LAST CLOCK TIME
1041  2B C8                             SUB     CX,AX                   ; SUB CURRENT TIME
1043  3B CA                             CMP     CX,DX                   ; IS IT TIME TO SAMPLE ?
1045  72 EA                             JC      I31                     ; NO, KEEP LOOKING AT TIME
1047  2B CA                             SUB     CX,DX                   ; UPDATE # OF COUNTS OFF
1049  8B F8                             MOV     DI,AX                   ; SAVE CURRENT TIME AS LAST TIME
104B  03 F9                             ADD     DI,CX                   ; ADD DIFFERENCE FOR NEXT TIME
                                ;-----------START SAMPLING DATA BIT (5 SAMPLES)
104D  B9 0005                           MOV     CX,5                    ; SET COUNTER
                                ;--------------------------------------------------------
                                ;
                                ; SAMPLE LINE
                                ;
                                ;       PORT_C IS SAMPLED CX TIMES AND IF THER ARE 3 OR MORE 1"S
                                ;       THEN 80H IS RETURNED IN AL, ELSE 00H IS RETURNED IN AL.
                                ;       PARITY COUNTER IS MAINTAINED IN ES.
                                ;
                                ;--------------------------------------------------------
1050  32 E4                             XOR     AH,AH                   ; CLEAR COUNTER
1052  E4 62                     I32:    IN      AL,PORT_C               ; GET SAMPLE
1054  A8 40                             TEST    AL,40H                  ; TEST IF 1
1056  74 02                             JZ      I33                     ; JMP IF 0
1058  FE C4                             INC     AH                      ; KEEP COUNT OF 1'S
105A  E2 F6                     I33:    LOOP    I32                     ; KEEP SAMPLING
105C  80 FC 03                          CMP     AH,3                    ; VALID 1 ?
105F  72 05                             JB      I34                     ; JMP IF NOT VALID 1
1061  B0 80                             MOV     AL,080H                 ; RETURN 80H IN AL (1)
1063  FE C3                             INC     BL                      ; INCREMENT PARITY COUNTER
1065  C3                                RET                             ; RETURN TO CALLER
1066  32 C0                     I34:    XOR     AL,AL                   ; RETURN 0 IN AL (0)
1068  C3                                RET                             ; RETURN TO CALLER
1069                            I30     ENDP
```

```
                        ;--------------------------------------------------------------------
                        ;KEY62_INT
                        ;       THE PURPOSE OF THIS ROUTINE IS TO TRANSLATE SCAN CODES AND
                        ;       SCAN CODE COMBINATIONS FROM THE 62 KEY KEYBOARD TO THEIR
                        ;       EQUIVILENTS ON THE 83 KEY KEYBOARD.  THE SCAN CODE IS
                        ;       PASSED IN AL.  EACH SCAN CODE PASSED EITHER TRIGGERS ONE OR
                        ;       MORE CALLS TO INTERRUPT 9 OR SETS FLAGS TO RETAIN KEYBOARD
                        ;       STATUS.  WHEN INTERRUPT 9 IS CALLED THE TRANSLATED SCAN
                        ;       CODES ARE PASSED TO IT IN AL.  THE INTENT OF THIS CODE WAS
                        ;       TO KEEP INTERRUPT 9 INTACT FROM ITS ORIGIN IN THE PC FAMILY
                        ;       THIS ROUTINE IS IN THE FRONT END OF INTERRUPT 9 AND
                        ;       TRANSFORMS A 62 KEY KEYBOARD TO LOOK AS IF IT WERE AN 83
                        ;       KEY VERSION.
                        ;       IT IS ASSUMED THAT THIS ROUTINE IS CALLED FROM THE NMI
                        ;       DESERIALIZATION ROUTINE AND THAT ALL REGISTERS WERE SAVED
                        ;       IN THE CALLING ROUTINE.  AS A CONSEQUENCE ALL REGISTERS ARE
                        ;       DESTROYED.
                        ;--------------------------------------------------------------------
                        ;EQUATES
= 0080                  BREAK_BIT       EQU     80H
= 0054                  FN_KEY          EQU     54H
= 0055                  PHK             EQU     FN_KEY+1
= 0056                  EXT_SCAN        EQU     PHK+1   ; BASE CODE FOR SCAN CODES
                                                        ; EXTENDING BEYOND 83
= 00FF                  AND_MASK        EQU     0FFH    ; USED TO SELECTIVELY REMOVE BITS
= 001F                  CLEAR_FLAGS     EQU     AND_MASK - (FN_FLAG+FN_BREAK+FN_PENDING)
                        ;SCAN CODES
= 0030                  B_KEY           EQU     48
= 0010                  Q_KEY           EQU     16
= 0019                  P_KEY           EQU     25
= 0012                  E_KEY           EQU     18
= 001F                  S_KEY           EQU     31
= 0031                  N_KEY           EQU     49
= 0048                  UP_ARROW        EQU     72
= 0050                  DOWN_ARROW      EQU     80
= 004B                  LEFT_ARROW      EQU     75
= 004D                  RIGHT_ARROW     EQU     77
= 000C                  MINUS           EQU     12
= 000D                  EQUALS          EQU     13
= 000B                  NUM_0           EQU     11
                        ;NEW TRANSLATED SCAN CODES
                        ;--------------------------------------------------
                        ;NOTE:
                        ;       BREAK, PAUSE, ECHO, AND PRT_SCREEN ARE USED AS OFFSETS
                        ;       INTO THE TABLE 'SCAN'. OFFSET = TABLE POSITION + 1.
                        ;--------------------------------------------------
= 0001                  ECHO            EQU     01
= 0002                  BREAK           EQU     02
= 0003                  PAUSE           EQU     03
= 0004                  PRT_SCREEN      EQU     04
= 0046                  SCROLL_LOCK     EQU     70
= 0045                  NUM_LOCK        EQU     69
= 0047                  HOME            EQU     71
= 004F                  END_KEY         EQU     79
= 0049                  PAGE_UP         EQU     73
= 0051                  PAGE_DOWN       EQU     81
= 004A                  KEYPAD_MINUS    EQU     74
= 004E                  KEYPAD_PLUS     EQU     78
                                ASSUME CS:CODE,DS:DATA
                        ;----TABLE OF VALID SCAN CODES
1069                    KBO     LABEL BYTE
1069 30 10 12 19 1F 31          DB B_KEY, Q_KEY, E_KEY, P_KEY, S_KEY, N_KEY
106F 48 50 4B 4D 0C             DB UP_ARROW, DOWN_ARROW, LEFT_ARROW, RIGHT_ARROW, MINUS
1074 0D                         DB EQUALS
= 000C                  KBOLEN          EQU $ - KBO
                        ;----TABLE OF NEW SCAN CODES
1075                    KB1     LABEL BYTE
1075 02 03 01 04 46 45          DB BREAK, PAUSE, ECHO, PRT_SCREEN, SCROLL_LOCK, NUM_LOCK
107B 47 4F 49 51 4A 4E          DB HOME,END_KEY,PAGE_UP,PAGE_DOWN,KEYPAD_MINUS,KEYPAD_PLUS
                        ;--------------------------------------------------
                        ;NOTE:  THERE IS A ONE TO ONE CORRESPONDENCE BETWEEN
                        ;       THE SIZE OF KBO AND KB1.
                        ;--------------------------------------------------
                        ;TABLE OF NUMERIC KEYPAD SCAN CODES
                        ;       THESE SCAN CODES WERE NUMERIC KEYPAD CODES ON
                        ;       THE 83 KEY KEYBOARD.
                        ;--------------------------------------------------
1081                    NUM_CODES LABEL BYTE
1081 4F 50 51 4B 4C 4D          DB 79,80,81,75,76,77,71,72,73,82
     47 48 49 52
                        ;--------------------------------------------------
                        ;TABLE OF SIMULATED KEYSTROKES
                        ;       THIS TABLE REPRESENTS A 4*2 ARRAY.  EACH ROW
                        ;       CONSISTS OF A SEQUENCE OF SCAN CODES WHICH
                        ;       WOULD HAVE BEEN GENERATED ON AN 83 KEY KEYBOARD
                        ;       TO CAUSE THE FOLLOWING FUNCTIONS:
                        ;               ROW 1=ECHO CRT OUTPUT TO THE PRINTER
                        ;               ROW 2=BREAK
                        ;       THE TABLE HAS BOTH MAKE AND BREAK SCAN CODES.
                        ;--------------------------------------------------
108B                    SCAN    LABEL BYTE
108B 1D 37 B7 9D                DB 29,55,183,157        ; CTRL + PRTSC
108F 1D 46 C6 9D                DB 29,70,198,157        ; CTRL + SCROLL-LOCK
```

```
                        ;--------------------------------------------------
                        ;TABLE OF VALID ALT SHIFT SCAN CODES
                        ;       THIS TABLE CONTAINS SCAN CODES FOR KEYS ON THE
                        ;       62 KEY KEYBOARD.  THESE CODES ARE USED IN
                        ;       COMBINATION WITH THE ALT KEY TO PRODUCE SCAN CODES
                        ;       FOR KEYS NOT FOUND ON THE 62 KEY KEYBOARD.
                        ;--------------------------------------------------
1093                    ALT_TABLE       LABEL   BYTE
1093  35 28 34 1A 1B            DB 53,40,52,26,27
= 0005                  ALT_LEN  EQU $ - ALT_TABLE
                        ;--------------------------------------------------
                        ;TABLE OF TRANSLATED SCAN CODES WITH ALT SHIFT
                        ;       THIS TABLE CONTAINS THE SCAN CODES FOR THE
                        ;       KEYS WHICH ARE NOT ON THE 62 KEY KEYBOARD AND
                        ;       WILL BE TRANSLATED WITH ALT SHIFT.  THERE IS A
                        ;       ONE TO ONE CORRESPONDENCE BETWEEN THE SIZES
                        ;       OF ALT_TABLE AND NEW_ALT.
                        ;       THE FOLLOWING TRANSLATIONS ARE MADE:
                        ;               ALT+ / = \
                        ;               ALT+ ' = `
                        ;               ALT+ [ = {
                        ;               ALT+ ] = ~
                        ;               ALT+ . = *
                        ;--------------------------------------------------
1098                    NEW_ALT LABEL BYTE
1098  28 29 37 2B 29            DB 43,41,55,43,41
                        ;--------------------------------------------------
                        ;EXTAB
                        ;       TABLE OF SCAN CODES FOR MAPPING EXTENDED SET
                        ;       OF SCAN CODES (SCAN CODES > 85).  THIS TABLE
                        ;       ALLOWS OTHER DEVICES TO USE THE KEYBOARD INTERFACE.
                        ;       IF THE DEVICE GENERATES A SCAN CODE > 85 THIS TABLE
                        ;       CAN BE USED TO MAP THE DEVICE TO THE KEYBOARD.  THE
                        ;       DEVICE ALSO HAS THE OPTION OF HAVING A UNIQUE SCAN
                        ;       CODE PUT IN THE KEYBOARD BUFFER (INSTEAD OF MAPPING
                        ;       TO THE KEYBOARD). THE EXTENDED SCAN CODE PUT IN THE
                        ;       BUFFER WILL BE CONTINUOUS BEGINNING AT 150.  A ZERO
                        ;       WILL BE USED IN PLACE OF AN ASCII CODE. (E.G. A
                        ;       DEVICE GENERATING SCAN CODE 86 AND NOT MAPPING 86
                        ;       TO THE KEYBOARD WILL HAVE A [150,0] PUT IN THE
                        ;       KEYBOARD BUFFER)
                        ;       TABLE FORMAT:
                        ;       THE FIRST BYTE IS A LENGTH INDICATING THE NUMBER
                        ;       OF SCAN CODES MAPPED TO THE KEYBOARD.  THE REMAINING
                        ;       ENTRIES ARE WORDS. THE FIRST BYTE (LOW BYTE) IS A
                        ;       SCAN CODE AND THE SECOND BYTE (HIGH BYTE) IS ZERO.
                        ;       A DEVICE GENERATING N SCAN CODES IS ASSUMED TO GENERATE THE
                        ;       FOLLOWING STREAM 86,87,88,...,86+(N-1). THE SCAN CODE BYTES
                        ;       IN THE TABLE CORRESPOND TO THIS SET WITH THE FIRST DATA
                        ;       BYTE MATCHING 86, THE SECOND MATCHING 87 ETC.
                        ;   NOTES:
                        ;       (1) IF A DEVICE GENERATES A BREAK CODE, NOTHING IS
                        ;           PUT IN THE BUFFER.
                        ;       (2) A LENGTH OF 0 INDICATES THAT ZERO SCAN CODES HAVE BEEN
                        ;           MAPPED TO THE KEYBOARD AND ALL EXTENDED SCAN CODES WILL
                        ;           BE USED.
                        ;       (3) A DEVICE CAN MAP SOME OF ITS SCAN CODES TO THE KEYBOARD
                        ;           AND HAVE SOME ITS SCAN CODES IN THE EXTENDED SET.
                        ;--------------------------------------------------
1090                    EXTAB   LABEL BYTE
1090  14                        DB      20              ; LENGTH OF TABLE
1091  0048 0049 004D 0051       DW      72,73,77,81,80,79,75,71,57,28
      0050 004F 004B 0047
      0039 001C
10B2  0011 0012 001F 002D       DW      17,18,31,45,44,43,30,16,15,1
      002C 002B 001E 0010
      000F 0001
                        ;
10C6                    KEY62_INT PROC FAR
10C6  FB                        STI
10C7  FC                        CLD                     ; FORWARD DIRECTION
10C8  E8 138B R                 CALL    DDS             ; SET UP ADDRESSING
10CB  8A E0                     MOV     AH,AL           ; SAVE SCAN CODE
10CD  E8 131E R                 CALL    TPM             ; ADJUST OUTPUT FOR USER
                                                        ; MODIFICATION
10D0  73 01                     JNC     KBX0            ; JUMP IF OK TO CONTINUE
10D2  CF                        IRET                    ; RETURN FROM INTERRUPT.
                        ;----EXTENDED SCAN CODE CHECK
10D3  3C FF             KBX0:   CMP     AL,0FFH         ; IS THIS AN OVERRUN CHAR?
10D5  74 6C                     JE      KBO_1           ; PASS IT TO INTERRUPT 9
10D7  24 7F                     AND     AL,AND_MASK-BREAK_BIT ; TURN OFF BREAK BIT
10D9  3C 56                     CMP     AL,EXT_SCAN     ; IS THIS A SCAN CODE > 83
10DB  7C 5F                     JL      KBX4            ; REPLACE BREAK BIT
                        ;----SCAN CODE IS IN EXTENDED SET
10DD  1E                        PUSH    DS
10DE  33 F6                     XOR     SI,SI
10E0  8E DE                     MOV     DS,SI
                                ASSUME  DS:ABS0
10E2  C4 3E 0124 R              LES     DI,DWORD PTR EXST ; GET THE POINTER TO THE EXTENDED
                                                        ; SET
10E6  26: 8A 0D                 MOV     CL,BYTE PTR ES:[DI] ; GET LENGTH BYTE
10E9  1F                        POP     DS
                                ASSUME  DS:DATA
                        ;----DOES SCAN CODE GET MAPPED TO KEYBOARD OR TO NEW EXTENDED SCAN
                        ;       CODES?
10EA  2C 56                     SUB     AL,EXT_SCAN     ; CONVERT TO BASE OF NEW SET
10EC  FE C9                     DEC     CL              ; LENGTH - 1
10EE  3A C1                     CMP     AL,CL           ; IS CODE IN TABLE?
10F0  7F 10                     JG      KBX1            ; JUMP IF SCAN CODE IS NOT IN TABLE
```

```
10F2  47                              ;----GET SCAN CODE FROM TABLE
                                              INC     DI                ; POINT DI PAST LENGTH BYTE
10F3  8B D8                                   MOV     BX,AX
10F5  32 FF                                   XOR     BH,BH             ; PREPARE FOR ADDING TO 16 BIT
                                                                        ; REGISTER
10F7  D1 E3                                   SHL     BX,1
10F9  03 FB                                   ADD     DI,BX             ; OFFSET TO CORRECT TABLE ENTRY
10FB  26: 8A 05                               MOV     AL,BYTE PTR ES:[DI] ; TRANSLATED SCAN CODE IN AL
10FE  3C 56                                   CMP     AL,EXT_SCAN       ; IS CODE IN KEYBOARD SET?
1100  7C 3A                                   JL      KBX4              ; IN KEYBOARD SET, CHECK FOR BREAK
                                      ;----SCAN CODE GETS MAPPED TO EXTENDED SCAN CODES
1102  F6 C4 80                       KBX1:   TEST    AH,BREAK_BIT      ; IS THIS A BREAK CODE?
1105  74 01                                   JZ      KBX2              ; MAKE CODE, PUT IN BUFFER
1107  CF                                      IRET                      ; BREAK CODE, RETURN FROM INTERRUPT
1108  80 C4 40                       KBX2:   ADD     AH,64             ; EXTENDED SET CODES BEGIN AT 150
110B  32 C0                                   XOR     AL,AL             ; ZERO OUT ASCII VALUE (NUL)
110D  8B 1E 001C R                            MOV     BX,BUFFER_TAIL    ; GET TAIL POINTER
1111  8B F3                                   MOV     SI,BX             ; SAVE POINTER TO TAIL
1113  E8 144F R                               CALL    K4                ; INCREMENT TAIL VALUE
1116  3B 1E 001A R                            CMP     BX,BUFFER_HEAD    ; IS BUFFER FULL?
111A  75 19                                   JNE     KBX3              ; PUT CONTENTS OF AX IN BUFFER
                                      ;----BUFFER IS FULL, BEEP AND CLEAR FLAGS
111C  BB 0080                                 MOV     BX,80H            ; FREQUENCY OF BEEP
111F  B9 0048                                 MOV     CX,48H            ; DURATION OF BEEP
1122  E8 E035 R                               CALL    KB_NOISE          ; BUFFER FULL BEEP
1125  80 26 0017 R F0                         AND     KB_FLAG,0F0H      ; CLEAR ALT, CTRL, LEFT AND RIGHT
                                                                        ; SHIFTS
112A  80 26 0018 R 0F                         AND     KB_FLAG_1,0FH     ; CLEAR MAKE OF INS,CAPS_LOCK,NUM
                                                                        ; AND SCROLL
112F  80 26 0088 R 1F                         AND     KB_FLAG_2,1FH     ; CLEAR FUNCTION STATES
1134  CF                                      IRET                      ; DONE WITH INTERRUPT
1135  89 04                           KBX3:   MOV     [SI],AX           ; PUT CONTENTS OF AX IN BUFFER
1137  89 1E 001C R                            MOV     BUFFER_TAIL,BX    ; ADVANCE BUFFER TAIL
113B  CF                                      IRET                      ; RETURN FROM INTERRUPT
113C  80 E4 80                       KBX4:   AND     AH,BREAK_BIT      ; MASK BREAK BIT ON ORIGINAL SCAN
113F  0A C4                                   OR      AL,AH             ; UPDATE NEW SCAN CODE
1141  8A E0                                   MOV     AH,AL             ; SAVE AL IN AH AGAIN
                                      ;----83 KEY KEYBOARD FUNCTIONS SHIFT+PRTSC AND CTRL+NUMLOCK
1143  3C 45                         KB0_1:   CMP     AL,NUM_KEY        ; IS THIS A NUMLOCK?
1145  75 14                                   JNE     KB0_3             ; CHECK FOR PRTSC
1147  F6 06 0017 R 04                         TEST    KB_FLAG,CTL_SHIFT ; IS CTRL KEY BEING HELD DOWN?
114C  74 0A                                   JZ      KB0_2             ; NUMLOCK WITHOUT CTRL, CONTINUE
114E  F6 06 0017 R 08                         TEST    KB_FLAG,ALT_SHIFT ; IS ALT KEY HELD CONCURRENTLY?
1153  75 03                                   JNZ     KB0_2             ; PASS IT ON
1155  E9 12EB R                               JMP     KB16_1            ; PUT KEYBOARD IN HOLD STATE
1158  E9 125C R                      KB0_2:   JMP     CONT_INT          ; CONTINUE WITH INTERRUPT 48H
                                      ;----CHECK FOR PRTSC
115B  3C 37                         KB0_3:   CMP     AL,55             ; IS THIS A PRTSC KEY?
115D  75 11                                   JNZ     KB1_1             ; NOT A PRTSC KEY
115F  F6 06 0017 R 03                         TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; EITHER SHIFT
                                                                        ; ACTIVE?
1164  74 F2                                   JZ      KB0_2             ; PROCESS SCAN IN INT9
1166  F6 06 0017 R 04                         TEST    KB_FLAG,CTL_SHIFT ; IS THE CTRL KEY PRESSED?
116B  75 EB                                   JNZ     KB0_2             ; NOT A VALID PRTSC (PC COMPATIBLE)
116D  E9 1301 R                               JMP     PRTSC             ; HANDLE THE PRINT SCREEN FUNCTION
                                      ;----ALTERNATE SHIFT TRANSLATIONS
1170  8A E0                         KB1_1:   MOV     AH, AL            ; SAVE CHARACTER
1172  24 7F                                   AND     AL, AND_MASK - BREAK_BIT ; MASK BREAK BIT
1174  F6 06 0017 R 08                         TEST    KB_FLAG,ALT_SHIFT ; IS THIS A POTENTIAL TRANSLATION
1179  74 39                                   JZ      KB2
                                      ;----TABLE LOOK UP
117B  0E                                      PUSH    CS
117C  07                                      POP     ES                ; INITIALIZE SEGMENT FOR TABLE LOOK
                                                                        ; UP
117D  BF 1093 R                               MOV     DI,OFFSET ALT_TABLE
1180  B9 0005                                 MOV     CX,ALT_LEN        ; GET READY FOR TABLE LOOK UP
1183  F2/ AE                                  REPNE   SCASB             ; SERACH TABLE
1185  75 2D                                   JNE     KB2               ; JUMP IF MATCH IS NOT FOUND
1187  B9 1094 R                               MOV     CX,OFFSET ALT_TABLE + 1
118A  2B F9                                   SUB     DI,CX             ; UPDATE DI TO INDEX SCAN CODE
118C  2E: 8A 85 1098 R                        MOV     AL,CS:NEW_ALT[DI] ; TRANSLATE SCAN CODE
                                      ;----CHECK FOR BREAK CODE
1191  8A 1E 0017 R                            MOV     BL,KB_FLAG        ; SAVE KB_FLAG STATUS
1195  80 36 0017 R 08                         XOR     KB_FLAG,ALT_SHIFT ; MASK OFF ALT SHIFT
119A  F6 C4 80                                TEST    AH,BREAK_BIT      ; IS THIS A BREAK CHARACTER?
119D  74 02                                   JZ      KB1_2             ; JUMP IF SCAN IS A MAKE
119F  0C 80                                   OR      AL,BREAK_BIT      ; SET BREAK BIT
                                      ;----MAKE CODE, CHECK FOR SHIFT SEQUENCE
11A1  83 FF 03                      KB1_2:   CMP     DI,3              ; IS THIS A SHIFT SEQUENCE
11A4  7C 05                                   JL      KB1_3             ; JUMP IF NOT SHIFT SEQUENCE
11A6  80 0E 0017 R 02                         OR      KB_FLAG,LEFT_SHIFT ; TURN ON SHIFT FLAG
11AB  E6 60                         KB1_3:   OUT     KBPORT,AL
11AD  CD 09                                   INT     9H                ; ISSUE INT TO PROCESS SCAN CODE
11AF  88 1E 0017 R                            MOV     KB_FLAG,BL        ; RESTORE ORIGINAL FLAG STATES
11B3  CF                                      IRET
                                      ;----FUNCTION KEY HANDLER
11B4  3C 54                         KB2:     CMP     AL, FN_KEY        ; CHECK FOR FUNCTION KEY
11B6  75 23                                   JNZ     KB4               ; JUMP IF NOT FUNCTION KEY
11B8  F6 C4 80                                TEST    AH, BREAK_BIT     ; IS THIS A FUNCTION BREAK
11BB  75 0B                                   JNZ     KB3               ; JUMP IF FUNCTION BREAK
11BD  80 26 0088 R 1F                         AND     KB_FLAG_2,CLEAR_FLAGS ; CLEAR ALL PREVIOUS
                                                                        ; FUNCTIONS
11C2  80 0E 0088 R A0                         OR      KB_FLAG_2, FN_FLAG + FN_PENDING
11C7  CF                                      IRET                      ; RETURN FROM INTERRUPT
                                      ;----FUNCTION BREAK
11C8  F6 06 0088 R 20               KB3:     TEST    KB_FLAG_2,FN_PENDING
11CD  75 06                                   JNZ     KB3_1             ; JUMP IF FUNCTION IS PENDING
11CF  80 26 0088 R 1F                         AND     KB_FLAG_2,CLEAR_FLAGS ; CLEAR ALL FLAGS
11D4  CF                                      IRET
11D5  80 0E 0088 R 40               KB3_1:   OR      KB_FLAG_2,FN_BREAK ; SET BREAK FLAG
11DA  CF                            KB3_2:   IRET                      ; RETURN FROM INTERRUPT
```

```
                        ;----CHECK IF FUNCTION FLAG ALREADY SET
11DB  3C 55             KB4:    CMP     AL,PHK          ; IS THIS A PHANTOM KEY?
11DD  74 F8                     JZ      KB3_2           ; JUMP IF PHANTOM SEQUENCE
11DF  F6 06 0088 R 90   KB4_0:  TEST    KB_FLAG_2,FN_FLAG+FN_LOCK ; ARE WE IN FUNCTION
                                                        ; STATE?
11E4  75 21                     JNZ     KB5
                        ;----CHECK IF NUM_STATE IS ACTIVE
11E6  F6 06 0017 R 20           TEST    KB_FLAG,NUM_STATE
11EB  74 16                     JZ      KB4_1           ; JUMP IF NOT IN NUM_STATE
11ED  3C 0B                     CMP     AL,NUM_0        ; ARE WE IN NUMERIC KEYPAD REGION?
11EF  77 12                     JA      KB4_1           ; JUMP IF NOT IN KEYPAD
11F1  FE C8                     DEC     AL              ; CHECK LOWER BOUND OF RANGE
11F3  74 0E                     JZ      KB4_1           ; JUMP IF NOT IN RANGE (ESC KEY)
                        ;----TRANSLATE SCAN CODE TO NUMERIC KEYPAD
11F5  FE C8                     DEC     AL              ; AL IS OFFSET INTO TABLE
11F7  BB 1081 R                 MOV     BX,OFFSET NUM_CODES
11FA  2E: D7                    XLAT    CS:NUM_CODES    ; NEW SCAN CODE IS IN AL
11FC  80 E4 80                  AND     AH,BREAK_BIT    ; ISOLATE BREAK BIT ON ORIGINAL
                                                        ; SCAN CODE
11FF  0A C4                     OR      AL,AH           ; UPDATE KEYPAD SCAN CODE
1201  EB 59                     JMP     SHORT CONT_INT  ; CONTINUE WITH INTERRUPT
1203  8A C4            KB4_1:  MOV     AL,AH           ; GET BACK BREAK BIT IF SET
1205  EB 55                     JMP     SHORT CONT_INT
                        ;----CHECK FOR VALID FUNCTION KEY
1207  3C 0B             KB5:    CMP     AL, NUM_0       ; CHECK FOR RANGE OF INTEGERS
1209  77 2D                     JA      KB7             ; JUMP IF NOT IN RANGE
120B  FE C8                     DEC     AL              ; CHECK FOR ESC KEY (=1)
120D  75 25                     JNZ     KB6             ; NOT ESCAPE KEY, RANGE OF INTEGERS
                        ;----ESCAPE KEY, LOCK KEYBOARD IN FUNCTION LOCK
120F  F6 C4 80                  TEST    AH,BREAK_BIT    ; IS THIS A BREAK CODE?
1212  75 30                     JNZ     KB8             ; NO PROCESSING FOR ESCAPE BREAK
1214  F6 06 0088 R 80           TEST    KB_FLAG_2,FN_FLAG ; TOGGLES ONLY WHEN FN HELD
                                                        ; CONCURRENTLY
1219  74 29                     JZ      KB8             ; NOT HELD CONCURRENTLY
121B  F6 06 0088 R 40           TEST    KB_FLAG_2,FN_BREAK ; HAS THE FUNCTION KEY BEEN
                                                        ; RELEASED?
1220  75 22                     JNZ     KB8             ; CONTINUE IF RELEASED.  PROCESS AS
                                                        ; ESC
1222  F6 06 0017 R 03           TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; EITHER SHIFT?
1227  74 1B                     JZ      KB8             ; NOT HELD DOWN
1229  80 36 0088 R 10           XOR     KB_FLAG_2,FN_LOCK ; TOGGLE STATE
122E  80 26 0088 R 1F           AND     KB_FLAG_2,CLEAR_FLAGS ; TURN OFF OTHER STATES
1233  CF                       IRET                    ; RETURN FROM INTERUPT
                        ;----SCAN CODE IN RANGE 1 -> 0
1234  04 3A             KB6:    ADD     AL, 58          ; GENERATE CORRECT SCAN CODE
1236  EB 3E                     JMP     SHORT KB12      ; CLEAN-UP BEFORE RETURN TO KB_INT
                        ;----CHECK TABLE FOR OTHER VALID SCAN CODES
1238  0E                KB7:    PUSH    CS
1239  07                        POP     ES              ; ESTABLISH ADDRESS OF TABLE
123A  BF 1069 R                 MOV     DI, OFFSET KB0  ; BASE OF TABLE
123D  B9 000C                   MOV     CX, KB0LEN      ; LENGTH OF TABLE
1240  F2/ AE                    REPNE   SCASB           ; SEARCH TABLE FOR A MATCH
1242  74 1D                     JE      KB10            ; JUMP IF MATCH
                        ;----ILLEGAL CHARACTER
1244  F6 06 0088 R 40   KB8:    TEST    KB_FLAG_2,FN_BREAK ; HAS BREAK OCCURED?
1249  74 0F                     JZ      KB9             ; FUNCTION KEY HAS NOT BEEN
                                                        ; RELEASED
124B  F6 C4 80                  TEST    AH,BREAK_BIT    ; IS THIS A BREAK OF AN ILLEGAL
124E  75 0A                     JNZ     KB9             ; DON'T RESET FLAGS ON ILLEGAL
                                                        ; BREAK
1250  80 26 0088 R 1F   KB85:   AND     KB_FLAG_2,CLEAR_FLAGS ; NORMAL STATE
1255  C6 06 0087 R 00           MOV     CUR_FUNC,0      ; RETRIEVE ORIGINAL SCAN CODE
                        ;----FUNCTION BREAK IS NOT SET
125A  8A C4             KB9:    MOV     AL,AH           ; RETRIEVE ORIGINAL SCAN CODE
125C                    CONT_INT:
125C  E6 60                     OUT     KBPORT,AL
125E  CD 09                     INT     9H              ; ISSUE KEYBOARD INTERRUPT
1260                    RET_INT:
1260  CF                        IRET
                        ;----BEFORE TRANSLATION CHECK FOR ALT+FN+N_KEY AS NUM LOCK
1261  3C 31             KB10:   CMP     AL,N_KEY        ; IS THIS A POTENTIAL NUMLOCK?
1263  75 07                     JNE     KB10_1          ; NOT A NUMKEY, TRANSLATE IT
1265  F6 06 0017 R 08           TEST    KB_FLAG,ALT_SHIFT ; ALT HELD DOWN ALSO?
126A  74 D8                     JZ      KB8             ; TREAT AS ILLEGAL COMBINATION
126C  B9 106A R         KB10_1: MOV     CX, OFFSET KB0 + 1 ; GET OFFSET TO TABLE
126F  2B F9                     SUB     DI, CX          ; UPDATE INDEX TO NEW SCAN CODE
                                                        ; TABLE
1271  2E: 8A 85 1075 R          MOV     AL, CS:KB1[DI]  ; MOV NEW SCAN CODE INTO REGISTER
                        ;----TRANSLATED CODE IN AL OR AN OFFSET TO THE TABLE "SCAN"
1276  F6 C4 80           KB12:   TEST    AH,BREAK_BIT    ; IS THIS A BREAK CHAR?
1279  74 35                     JZ      KB13            ; JUMP IF MAKE CODE
                        ;----CHECK FOR TOGGLE KEY
127B  3C 45                     CMP     AL,NUM_LOCK     ; IS THIS A NUM LOCK?
127D  74 04                     JZ      KB12_1          ; JUMP IF TOGGLE KEY
127F  3C 46                     CMP     AL,SCROLL_LOCK  ; IS THIS A SCROLL LOCK?
1281  75 08                     JNZ     KB12_2          ; JUMP IF NOT A TOGGLE KEY
1283  0C 80           KB12_1:  OR      AL,80H          ; TURN ON BREAK BIT
1285  E6 60                     OUT     KBPORT,AL
1287  CD 09                     INT     9H              ; TOGGLE STATE
1289  24 7F                     AND     AL,AND_MASK-BREAK_BIT ; TURN OFF BREAK BIT
128B  F6 06 0088 R 40   KB12_2: TEST    KB_FLAG_2,FN_BREAK ; HAS FUNCTION BREAK OCCURED?
1290  74 11                     JZ      KB12_3          ; JUMP IF BREAK HAS NOT OCCURED
1292  3A 06 0087 R              CMP     AL,CUR_FUNC     ; IS THIS A BREAK OF OLD VALID
                                                        ; FUNCTION
1296  75 C8                     JNE     RET_INT         ; ALLOW FURTHER CURRENT FUNCTIONS
1298  80 26 0088 R 1F          AND     KB_FLAG_2,CLEAR_FLAGS
129D                    KB12_20:
129D  C6 06 0087 R 00           MOV     CUR_FUNC,0      ; CLEAR CURRENT FUNCTION
12A2  CF                        IRET                    ; RETURN FROM INTERRUPT
```

```
12A3  3A 06 0087 R            KB12_3: CMP     AL,CUR_FUNC      ; IS THIS BREAK OF FIRST FUNCTION?
12A7  75 B7                           JNE     RET_INT          ; IGNORE
12A9  80 26 0088 R DF                 AND     KB_FLAG_2,AND_MASK-FN_PENDING ; TURN OFF PENDING
                                                               ; FUNCTION
12AE  EB ED                           JMP     KB12_20          ; CLEAR CURRENT FUNCTION AND RETURN
                                    ;----VALID MAKE KEY HAS BEEN PRESSED
12B0  F6 06 0088 R 40         KB13:   TEST    KB_FLAG_2,FN_BREAK ; CHECK IF FUNCTION KEY HAS BEEN
                                                               ; PRESSED
12B5  74 0D                           JZ      KB14_1           ; JUMP IF NOT SET
                                    ;----FUNCTION BREAK HAS ALREADY OCCURED
12B7  80 3E 0087 R 00                 CMP     CUR_FUNC,0       ; IS THIS A NEW FUNCTION?
12BC  74 06                           JZ      KB14_1           ; INITIALIZE NEW FUNCTION
12BE  38 06 0087 R                    CMP     CUR_FUNC,AL      ; IS THIS NON-CURRENT FUNCTION
12C2  75 8C                           JNZ     KB85             ;JUMP IF NO FUNCTION IS PENDING
                                                               ; ..TO RETRIEVE ORIGINAL SCAN CODE
                                    ;----CHECK FOR SCAN CODE GENERATION SEQUENCE
12C4  A2 0087 R               KB14_1: MOV     CUR_FUNC,AL      ; INITIALIZE CURRENT FN
12C7  3C 04                   KB16:   CMP     AL,PRT_SCREEN    ; IS THIS A SIMULATED SEQUENCE?
12C9  7F 91                           JG      CONT_INT         ; JUMP IF THIS IS A SIMPLE
                                                               ; TRANSLATION
12CB  74 34                           JZ      PRTSC            ; DO THE PRINT SCREEN FUNCTION
12CD  3C 03                           CMP     AL,PAUSE         ; IS THIS THE HOLD FUNCTION?
12CF  74 1A                           JZ      KB16_1           ; DO THE PAUSE FUNCTION
                                    ;----BREAK OR ECHO
12D1  FE C8                           DEC     AL               ; POINT AT BASE
12D3  D0 E0                           SHL     AL,1
12D5  D0 E0                           SHL     AL,1             ; MULTIPLY BY 4
12D7  98                              CBW
12D8  2E: 8D 36 108B R                LEA     SI,SCAN          ; ADDRESS SEQUENCE OF SIMULATED
                                                               ; KEYSTROKES
12DD  03 F0                           ADD     SI,AX            ; UPDATE TO POINT AT CORRECT SET
12DF  B9 0004                         MOV     CX,4             ; LOOP COUNTER
12E2                          GENERATE:
12E2  2E: AC                          LODS    SCAN             ; GET SCAN CODE FROM TABLE
12E4  E6 60                           OUT     KBPORT,AL
12E6  CD 09                           INT     9H               ; PROCESS IT
12E8  E2 F8                           LOOP    GENERATE         ; GET NEXT
12EA  CF                              IRET
                                    ;----PUT KEYBOARD IN HOLD STATE
12EB  F6 06 0018 R 08         KB16_1: TEST    KB_FLAG_1,HOLD_STATE ; CANNOT GO IN HOLD STATE IF
                                                               ; ITS ACTIVE
12F0  75 0E                           JNZ     KB16_2           ; DONE WITH INTERRUPT
12F2  80 0E 0018 R 08                 OR      KB_FLAG_1,HOLD_STATE ; TURN ON HOLD FLAG
12F7  E4 A0                           IN      AL,NMI_PORT      ; RESET KEYBOARD LATCH
12F9  F6 06 0018 R 08         HOLD:   TEST    KB_FLAG_1,HOLD_STATE ; STILL IN HOLD STATE?
12FE  75 F9                           JNZ     HOLD             ; CONTINUE LOOPING UNTIL KEY IS
                                                               ; PRESSED
1300  CF                      KB16_2: IRET                     ; RETURN FROM INTERRUPT 48H
                                    ;----PRINT SCREEN FUNCTION
1301  F6 06 0018 R 08        PRTSC:  TEST    KB_FLAG_1,HOLD_STATE ; IS HOLD STATE IN PROGRESS?
1306  74 06                           JZ      KB16_3           ; OK TO CONTINUE WITH PRTSC
1308  80 26 0018 R F7                AND     KB_FLAG_1,0FFH-HOLD_STATE ; TURN OFF FLAG
130D  CF                              IRET
130E  83 C4 06                KB16_3: ADD     SP,3*2           ; GET RID OF CALL TO INTERRUPT 48H
1311  07                             POP     ES               ; POP REGISTERS THAT AREN'T
                                                               ; MODIFIED IN INT5
1312  1F                              POP     DS
1313  5A                              POP     DX
1314  59                              POP     CX
1315  5B                              POP     BX
1316  E4 A0                           IN      AL,NMI_PORT      ; RESET KEYBOARD LATCH
1318  CD 05                           INT     5H               ; ISSUE INTERRUPT
131A  58                              POP     AX
131B  5F                              POP     DI
131C  5E                              POP     SI               ; POP THE REST
131D  CF                              IRET
131E                          KEY62_INT ENDP
                                    ;--------------------------------------------------------------
                                    ;TYPAMATIC
                                    ;       THIS ROUTINE WILL CHECK KEYBOARD STATUS BITS IN KB_FLAG_2
                                    ;       AND DETERMINE WHAT STATE THE KEYBOARD IS IN.  APPROPRIATE
                                    ;       ACTION WILL BE TAKEN.
                                    ;INPUT
                                    ;       AL= SCAN CODE OF KEY WHICH TRIGGERED NON-MASKABLE INTERRUPT
                                    ;OUTPUT
                                    ;       CARRY BIT = 1 IF NO ACTION IS TO BE TAKEN.
                                    ;       CARRY BIT = 0 MEANS SCAN CODE IN AL SHOULD BE PROCESSED
                                    ;                     FURTHER.
                                    ;       MODIFICATIONS TO THE VARIABLES CUR_CHAR AND VAR_DELAY ARE
                                    ;       MADE.  ALSO THE PUTCHAR BIT IN KB_FLAG_2 IS TOGGLED WHEN
                                    ;       THE KEYBOARD IS IN HALF RATE MODE.
                                    ;--------------------------------------------------------------
131E                          TPM     PROC    NEAR
131E  53                              PUSH    BX
131F  38 06 0085 R                    CMP     CUR_CHAR,AL      ; IS THIS A NEW CHARACTER?
1323  74 31                           JZ      TP2              ; JUMP IF SAME CHARACTER
                                    ;----NEW CHARACTER CHECK FOR BREAK SEQUENCES
1325  A8 80                           TEST    AL,BREAK_BIT     ; IS THE NEW KEY A BREAK KEY?
1327  74 12                           JZ      TP0              ; JUMP IF NOT A BREAK
1329  24 7F                           AND     AL,07FH          ; CLEAR BREAK BIT
132B  38 06 0085 R                    CMP     CUR_CHAR,AL      ; IS NEW CHARACTER THE BREAK OF
                                                               ; LAST MAKE?
132F  8A C4                           MOV     AL,AH            ; RETRIEVE ORIGINAL CHARACTER
1331  75 05                           JNZ     TP               ; JUMP IF NOT THE SAME CHARACTER
1333  C6 06 0085 R 00                 MOV     CUR_CHAR,00      ; CLEAR CURRENT CHARACTER
1338  F8                      TP:     CLC                      ; CLEAR CARRY BIT
1339  5B                              POP     BX
133A  C3                              RET                      ; RETURN
```

```
                        ;----INITIALIZE A NEW CHARACTER
133B  A2 0085 R         TP0:    MOV     CUR_CHAR,AL     ; SAVE NEW CHARACTER
133E  80 26 0086 R F0           AND     VAR_DELAY,0F0H  ; CLEAR VARIABLE DELAY
1343  80 26 0088 R FE           AND     KB_FLAG_2,0FEH  ; INITIAL PUTCHAR BIT AS ZERO
1348  F6 06 0088 R 02           TEST    KB_FLAG_2,INIT_DELAY ; ARE WE INCREASING THE
                                                        ; INITIAL DELAY?
134D  74 E9                     JZ      TP              ; DEFAULT DELAY
134F  80 0E 0086 R 0F           OR      VAR_DELAY,DELAY_RATE ; INCREASE DELAY BY 2X
1354  EB E2                     JMP     SHORT TP
                        ;----CHECK IF WE ARE IN TYPAMATIC MODE AND IF DELAY IS OVER
1356  F6 06 0088 R 08   TP2:    TEST    KB_FLAG_2,TYPE_OFF ; IS TYPAMATIC TURNED OFF?
135B  75 2B                     JNZ     TP4             ; JUMP IF TYPAMATIC RATE IS OFF
135D  8A 1E 0086 R              MOV     BL,VAR_DELAY    ; GET VAR_DEALY
1361  80 E3 0F                  AND     BL,0FH          ; MASK OFF HIGH ORDER(SCREEN RANGE)
1364  0A DB                     OR      BL,BL           ; IS INITIAL DELAY OVER?
1366  74 0D                     JZ      TP3             ; JUMP IF DELAY IS OVER
1368  FE CB                     DEC     BL              ; DECREASE DELAY WAIT BY ANOTHER
                                                        ; CHARACTER
136A  80 26 0086 R F0           AND     VAR_DELAY,0F0H
136F  08 1E 0086 R              OR      VAR_DELAY,BL
1373  EB 13                     JMP     SHORT TP4
                        ;----CHECK IF TIME TO OUTPUT CHAR
1375  F6 06 0088 R 04   TP3:    TEST    KB_FLAG_2,HALF_RATE ; ARE WE IN HALF RATE MODE
137A  74 BC                     JZ      TP              ; JUMP IF WE ARE IN NORMAL MODE
137C  80 36 0088 R 01           XOR     KB_FLAG_2,PUTCHAR ; TOGGLE BIT
1381  F6 06 0088 R 01           TEST    KB_FLAG_2,PUTCHAR ; IS IT TIME TO PUT OUT A CHAR
1386  75 B0                     JNZ     TP              ; NOT TIME TO OUTPUT CHARACTER
1388                    TP4:                            ; SKIP THIS CHARACTER
1388  F9                        STC                     ; SET CARRY FLAG
1389  5B                        POP     BX
138A  C3                        RET
138B                    TPM     ENDP
                        ;--------------------------------------------------------
                        ; THIS SUBROUTINE SETS DS TO POINT TO THE BIOS DATA AREA
                        ; INPUT: NONE
                        ; OUTPUT: DS IS SET
                        ;--------------------------------------------------------
138B                    DDS     PROC    NEAR
138B  50                        PUSH    AX
138C  B8 0040                   MOV     AX,40H
138F  8E D8                     MOV     DS,AX
1391  58                        POP     AX
1392  C3                        RET
1393                    DDS     ENDP
                        ;--- INT 1A ---------------------------------------------
                        ; TIME_OF_DAY/SOUND SOURCE SELECT
                        ;       THIS ROUTINE ALLOWS THE CLOCK TO BE SET/READ.
                        ;       AN INTERFACE FOR SETTING THE MULTIPLEXER FOR
                        ;       AUDIO SOURCE IS ALSO PROVIDED
                        ;
                        ; 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
                        ;   (AH) = 80H  SET UP SOUND MULTIPLEXER
                        ;       AL =(SOURCE OF SOUND) --> "AUDIO OUT" OR RF MODULATOR
                        ;            00 = 8253 CHANNEL 2
                        ;            01 = CASSETTE INPUT
                        ;            02 = "AUDIO IN" LINE ON I/O CHANNEL
                        ;            03 = COMPLEX SOUND GENERATOR CHIP
                        ;
                        ; 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
1393                    TIME_OF_DAY     PROC    FAR
1393  FB                        STI                     ; INTERRUPTS BACK ON
1394  1E                        PUSH    DS              ; SAVE SEGMENT
1395  E8 138B R                 CALL    DDS
1398  80 FC 80                  CMP     AH,80H          ; AH=80
139B  74 2E                     JE      T4A             ; MUX_SET-UP
139D  0A E4                     OR      AH,AH           ; AH=0
139F  74 07                     JZ      T2              ; READ_TIME
13A1  FE CC                     DEC     AH              ; AH=1
13A3  74 16                     JZ      T3              ; SET_TIME
13A5  FB                T1:     STI                     ; INTERRUPTS BACK ON
13A6  1F                        POP     DS              ; RECOVER SEGMENT
13A7  CF                        IRET                    ; RETURN TO CALLER
13A8  FA                T2:     CLI                     ; NO TIMER INTERRUPTS WHILE READING
13A9  A0 0070 R                 MOV     AL,TIMER_OFL
13AC  C6 06 0070 R 00           MOV     TIMER_OFL,0     ; GET OVERFLOW, AND RESET THE FLAG
13B1  8B 0E 006E R              MOV     CX,TIMER_HIGH
13B5  8B 16 006C R              MOV     DX,TIMER_LOW
13B9  EB EA                     JMP     T1              ; TOD_RETURN
13BB  FA                T3:     CLI                     ; NO INTERRUPTS WHILE WRITING
13BC  89 16 006C R              MOV     TIMER_LOW,DX
13C0  89 0E 006E R              MOV     TIMER_HIGH,CX   ; SET THE TIME
13C4  C6 06 0070 R 00           MOV     TIMER_OFL,0     ; RESET OVERFLOW
13C9  EB DA                     JMP     T1              ; TOD_RETURN
```

```
13CB  51                T4A:    PUSH    CX                      ;
13CC  B1 05                     MOV     CL,5                    ;
13CE  D2 E0                     SAL     AL,CL                   ; SHIFT PARM BITS LEFT 5 POSITIONS
13D0  86 C4                     XCHG    AL,AH                   ; SAVE PARM
13D2  E4 61                     IN      AL,PORT_B               ; GET CURRENT PORT SETTINGS
13D4  24 9F                     AND     AL,10011111B            ; ISOLATE MUX BITS
13D6  0A C4                     OR      AL,AH                   ; COMBINE PORT BITS/PARM BITS
13D8  E6 61                     OUT     PORT_B,AL               ; SET PORT TO NEW VALUE
13DA  59                        POP     CX
13DB  EB C8                     JMP     T1                      ; TOD_RETURN
13DD                    TIME_OF_DAY     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
                        ;       (AH)=3  SET TYPAMATIC RATES.  THE TYPAMATIC RATE CAN BE
                        ;               CHANGED USING THE FOLLOWING FUNCTIONS:
                        ;               (AL)=0      RETURN TO DEFAULT.  RESTORES ORIGINAL
                        ;                           STATE. I.E. TYPAMATIC ON, NORMAL INITIAL
                        ;                           DELAY, AND NORMAL TYPAMATIC RATE.
                        ;               (AL)=1      INCREASE INITIAL DELAY.  THIS IS THE
                        ;                           DELAY BETWEEN THE FIRST CHARACTER AND
                        ;                           THE BURST OF TYPAMATIC CHARS.
                        ;               (AL)=2      HALF_RATE.  SLOWS TYPAMATIC CHARACTERS
                        ;                           BY ONE HALF.
                        ;               (AL)=3      COMBINES AL=1 AND AL=2.  INCREASES
                        ;                           INITIAL DELAY AND SLOWS TYPAMATIC
                        ;                           CHARACTERS BY ONE-HALF.
                        ;               (AL)=4      TURN OFF TYPAMATIC CHARACTERS.  ONLY THE
                        ;                           FIRST CHARACTER IS HONORED.  ALL OTHERS
                        ;                           ARE IGNORED.
                        ;               AL IS RANGE CHECKED.  IF AL<0 OR AL>4 THE STATE
                        ;               REMAINS THE SAME.
                        ;               ***NOTE***  EACH TIME THE TYPAMATIC RATES ARE
                        ;               CHANGED ALL PREVIOUS STATES ARE REMOVED.  I.E. IF
                        ;               THE KEYBOARD IS IN THE HALF RATE MODE AND YOU WANT
                        ;               TO ADD AN INCREASE IN TYPAMATIC DELAY, YOU MUST
                        ;               CALL THIS ROUTINE WITH AH=3 AND AL=3.
                        ;       (AH)=4  ADJUST KEYBOARD BY THE VALUE IN AL AS FOLLOWS:
                        ;               (AL)=0      TURN OFF KEYBOARD CLICK.
                        ;               (AL)=1      TURN ON KEYBOARD CLICK.
                        ;               AL IS RANGE CHECKED.  THE STATE IS UNALTERED IF
                        ;               AL <> 1,0.
                        ; OUTPUT
                        ;       AS NOTED ABOVE, ONLY AX AND FLAGS CHANGED
                        ;       ALL REGISTERS RETAINED
                        ;--------------------------------------------------------------------------
13DD                    KEYBOARD_IO     PROC    FAR
                        ASSUME  CS:CODE,DS:DATA
13DD  FB                        STI                             ; INTERRUPTS BACK ON
13DE  1E                        PUSH    DS                      ; SAVE CURRENT DS
13DF  53                        PUSH    BX                      ; SAVE BX TEMPORARILY
13E0  E8 1388 R                 CALL    DDS                     ; POINT DS AT BIOS DATA SEGMENT
13E3  0A E4                     OR      AH,AH                   ; AH=0
13E5  74 0A                     JZ      K1                      ; ASCII_READ
13E7  FE CC                     DEC     AH                      ; AH=1
13E9  74 1E                     JZ      K2                      ; ASCII_STATUS
13EB  FE CC                     DEC     AH                      ; AH=2
13ED  74 2B                     JZ      K3                      ; SHIFT_STATUS
13EF  EB 2E                     JMP     SHORT   K3_1
                        ;------ READ THE KEY TO FIGURE OUT WHAT TO DO
13F1                    K1:                                     ; ASCII READ
13F1  FB                        STI                             ; INTERRUPTS BACK ON DURING LOOP
13F2  90                        NOP                             ; ALLOW AN INTERRUPT TO OCCUR
13F3  FA                        CLI                             ; INTERRUPTS BACK OFF
13F4  8B 1E 001A R              MOV     BX,BUFFER_HEAD          ; GET POINTER TO HEAD OF BUFFER
13F8  3B 1E 001C R              CMP     BX,BUFFER_TAIL          ; TEST END OF BUFFER
13FC  74 F3                     JZ      K1                      ; LOOP UNTIL SOMETHING IN BUFFER
13FE  8B 07                     MOV     AX,[BX]                 ; GET SCAN CODE AND ASCII CODE
1400  E8 144F R                 CALL    K4                      ; MOVE POINTER TO NEXT POSITION
1403  89 1E 001A R              MOV     BUFFER_HEAD,BX          ; STORE VALUE IN VARIABLE
1407  EB 43                     JMP     SHORT   RET_INT16
                        ;------ ASCII STATUS
1409  FA                K2:     CLI                             ; INTERRUPTS OFF
140A  8B 1E 001A R              MOV     BX,BUFFER_HEAD          ; GET HEAD POINTER
140E  3B 1E 001C R              CMP     BX,BUFFER_TAIL          ; IF EQUAL (Z=1) THEN NOTHING THERE
1412  8B 07                     MOV     AX,[BX]
1414  FB                        STI                             ; INTERRUPTS BACK ON
1415  5B                        POP     BX                      ; RECOVER REGISTER
1416  1F                        POP     DS                      ; RECOVER SEGMENT
1417  CA 0002                   RET     2                       ; THROW AWAY FLAGS
                        ;------ SHIFT STATUS
141A  A0 0017 R           K3:   MOV     AL,KB_FLAG              ; GET THE SHIFT STATUS FLAGS
141D  EB 2D                     JMP     SHORT   RET_INT16
```

```
141F  FE CC
1421  74 1A
1423  FE CC
1425  75 25
1427  0A C0
1429  75 07
142B  80 26 0018 R FB
1430  EB 1A
1432  3C 01
1434  75 16
1436  80 0E 0018 R 04
143B  EB 0F

143D  3C 04
143F  7F 0B
1441  80 26 0088 R F1
1446  D0 E0
1448  08 06 0088 R
144C
144C  5B
144D  1F
144E  CF
144F

144F
144F  43
1450  43
1451  3B 1E 0082 R
1455  75 04
1457  8B 1E 0080 R
145B  C3
145C

145C
145C  52
145D  3A 45 46 38 1D
1462  2A 36
= 0008

1464
1464  80
1465  40 20 10 08 04
146A  02 01

146C  1B FF 00 FF FF FF
      1E FF
1474  FF FF FF 1F FF 7F
      FF 11
147C  17 05 12 14 19 15
      09 0F
1484  10 1B 1D 0A FF 01
      13
148B  04 06 07 08 0A 0B
      0C FF FF
1494  FF FF 1C 1A 18 03
      16 02
149C  0E 0D FF FF FF FF
      FF FF
14A4  20 FF

14A6
14A6  5E 5F 60 61 62 63
      64 65
14AE  66 67 FF FF 77 FF
      84 FF
14B6  73 FF 74 FF 75 FF
      76 FF
14BE  FF

14BF
14BF  1B 31 32 33 34 35
      36 37 38 39 30 2D
      3D 08 09
14CE  71 77 65 72 74 79
      75 69 6F 70 5B 5D
      0D FF 61 73 64 66
      67 68 6A 6B 6C 3B
      27
14E7  60 FF 5C 7A 78 63
      76 62 6E 6D 2C 2E
      2F FF 2A FF 20
14F8  FF

14F9
14F9  1B 21 40 23 24 25
      5E 26 2A 28 29 5F
      2B 08 00
1508  51 57 45 52 54 59
      55 49 4F 50 7B 7D
      0D FF 41 53 44 46
      47 48 4A 4B 4C 3A
      22
1521  7E FF 7C 5A 58 43
      56 42 4E 4D 3C 3E
      3F FF 00 FF 20 FF
```

```
;------- ADJUST KEY CLICK
K3_1:   DEC     AH
        JZ      K3_3            ; AH=3, ADJUST TYPAMATIC
        DEC     AH              ; RANGE CHECK FOR AH=4
        JNZ     RET_INT16       ; ILLEGAL FUNCTION CALL
        OR      AL,AL           ; TURN OFF KEYBOARD CLICK?
        JNZ     K3_2            ; JUMP FOR RANGE CHECK
        AND     KB_FLAG_1,AND_MASK-CLICK_ON ; TURN OFF CLICK
        JMP     SHORT RET_INT16
K3_2:   CMP     AL,1            ; RANGE CHECK
        JNE     RET_INT16       ; NOT IN RANGE, RETURN
        OR      KB_FLAG_1,CLICK_ON ; TURN ON KEYBOARD CLICK
        JMP     SHORT   RET_INT16
;------ SET TYPAMATIC
K3_3:   CMP     AL,4            ; CHECK FOR CORRECT RANGE
        JG      RET_INT16       ; IF ILLEGAL VALUE IN AL IGNORE
        AND     KB_FLAG_2,0F1H  ; MASK OFF ANY OLD TYPAMATIC STATES
        SHL     AL,1            ; SHIFT TO PROPER POSITION
        OR      KB_FLAG_2,AL
RET_INT16:
        POP     BX              ; RECOVER REGISTER
        POP     DS              ; RECOVER REGISTER
        IRET                    ; RETURN TO CALLER
KEYBOARD_IO     ENDP
;------ INCREMENT A BUFFER POINTER
K4      PROC    NEAR
        INC     BX              ; MOVE TO NEXT WORD IN LIST
        INC     BX
        CMP     BX,BUFFER_END   ; AT END OF BUFFER?
        JNE     K5              ; NO, CONTINUE
        MOV     BX,BUFFER_START ; YES, RESET TO BUFFER BEGINNING
K5:     RET
K4      ENDP
;------ TABLE OF SHIFT KEYS AND MASK VALUES
K6      LABEL   BYTE
        DB      INS_KEY         ; INSERT KEY
        DB      CAPS_KEY,NUM_KEY,SCROLL_KEY,ALT_KEY,CTL_KEY
        DB      LEFT_KEY,RIGHT_KEY
K6L     EQU     $-K6
;------ SHIFT_MASK_TABLE
K7      LABEL   BYTE
        DB      INS_SHIFT       ; INSERT MODE SHIFT
        DB   CAPS_SHIFT,NUM_SHIFT,SCROLL_SHIFT,ALT_SHIFT,CTL_SHIFT
        DB      LEFT_SHIFT,RIGHT_SHIFT
;------ SCAN CODE TABLES
K8              DB      27,-1,0,-1,-1,-1,30,-1

                DB      -1,-1,-1,31,-1,127,-1,17

                DB      23,5,18,20,25,21,9,15

                DB      16,27,29,10,-1,1,19

                DB      4,6,7,8,10,11,12,-1,-1

                DB      -1,-1,28,26,24,3,22,2

                DB      14,13,-1,-1,-1,-1,-1,-1

                DB      ' ',-1
;-------- CTL TABLE SCAN
K9      LABEL   BYTE
                DB      94,95,96,97,98,99,100,101

                DB      102,103,-1,-1,119,-1,132,-1

                DB      115,-1,116,-1,117,-1,118,-1

                DB      -1
;------- LC TABLE
K10     LABEL   BYTE
                DB      01BH,'1234567890-=',08H,09H

                DB      'qwertyuiop[]',0DH,-1,'asdfghjkl;',027H

                DB      60H,-1,5CH,'zxcvbnm,./',-1,'*',-1,' '

                DB      -1
;------ UC TABLE
K11     LABEL   BYTE
                DB      27,'!@#$',37,05EH,'&*()_+',08H,0

                DB      'QWERTYUIOP{}',0DH,-1,'ASDFGHJKL:"'

                DB      07EH,-1,'|ZXCVBNM<>?',-1,0,-1,' ',-1
```

```
1533                                    ;------ UC TABLE SCAN
1533  54 55 56 57 58 59 5A     K12      LABEL   BYTE
                                        DB      84,85,86,87,88,89,90
153A  5B 5C 5D                          DB      91,92,93
                                        ;------ ALT TABLE SCAN
153D                            K13     LABEL   BYTE
153D  68 69 6A 6B 6C                    DB      104,105,106,107,108
1542  6D 6E 6F 70 71                    DB      109,110,111,112,113
                                        ;------ NUM STATE TABLE
1547                            K14     LABEL   BYTE
1547  37 38 39 2D 34 35 36 2B 31 32 33 30 2E
                                        DB      '789-456+1230.'

                                        ;------ BASE CASE TABLE
1554                            K15     LABEL   BYTE
1554  47 48 49 FF 4B FF 4D             DB      71,72,73,-1,75,-1,77
1558  FF 4F 50 51 52 53                 DB      -1,79,80,81,82,83
                                        ;------ KEYBOARD INTERRUPT ROUTINE
1561                            KB_INT  PROC    FAR
1561  FB                                STI                     ; ALLOW FURTHER INTERRUPTS
1562  50                                PUSH    AX
1563  53                                PUSH    BX
1564  51                                PUSH    CX
1565  52                                PUSH    DX
1566  56                                PUSH    SI
1567  57                                PUSH    DI
1568  1E                                PUSH    DS
1569  06                                PUSH    ES
156A  FC                                CLD                     ; FORWARD DIRECTION
156B  E8 138B R                         CALL    DDS
156E  8A E0                             MOV     AH,AL           ; SAVE SCAN CODE IN AH
                                        ;------ TEST FOR OVERRUN SCAN CODE FROM KEYBOARD
1570  3C FF                            CMP     AL,0FFH         ; IS THIS AN OVERRUN CHAR?
1572  75 1B                            JNZ     K16             ; NO, TEST FOR SHIFT KEY
1574  BB 0080                          MOV     BX,80H          ; DURATION OF ERROR BEEP
1577  B9 0048                          MOV     CX,48H          ; FREQUENCY OF TONE
157A  E8 E035 R                         CALL    KB_NOISE        ; BUFFER FULL BEEP
157D  80 26 0017 R F0                   AND     KB_FLAG,0F0H    ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                                ; SHIFTS
1582  80 26 0018 R 0F                   AND     KB_FLAG_1,0FH   ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                                ; ,NUM AND SCROLL SHIFT
1587  80 26 0098 R 1F                   AND     KB_FLAG_2,1FH   ; CLEAR FUNCTION STATES
158C  E9 164A R                         JMP     K26             ; END OF INTERRUPT
                                        ;------ TEST FOR SHIFT KEYS
158F                            K16:                            ; TEST_SHIFT
158F  24 7F                             AND     AL,07FH         ; TURN OFF THE BREAK BIT
1591  0E                                PUSH    CS
1592  07                                POP     ES              ; ESTABLISH ADDRESS OF SHIFT TABLE
1593  BF 145C R                         MOV     DI,OFFSET K6    ; SHIFT KEY TABLE
1596  B9 0008                          MOV     CX,K6L          ; LENGTH
1599  F2/ AE                           REPNE   SCASB           ; LOOK THROUGH THE TABLE FOR A
                                                                ; MATCH
159B  8A C4                             MOV     AL,AH           ; RECOVER SCAN CODE
159D  74 03                             JE      K17             ; JUMP IF MATCH FOUND
159F  E9 163A R                         JMP     K25             ; IF NO MATCH, THEN SHIFT NOT FOUND
                                        ;------ SHIFT KEY FOUND
15A2  81 EF 145D R              K17:    SUB     DI,OFFSET K6+1  ; ADJUST PTR TO SCAN CODE MATCH
15A6  2E: 8A A5 1464 R                  MOV     AH,CS:K7[DI]    ; GET MASK INTO AH
15AB  A8 80                             TEST    AL,80H          ; TEST FOR BREAK KEY
15AD  75 51                             JNZ     K23             ; BREAK_SHIFT_FOUND
                                        ;------ SHIFT MAKE FOUND, DETERMINE SET OR TOGGLE
15AF  80 FC 10                          CMP     AH,SCROLL_SHIFT
15B2  73 07                             JAE     K18             ; IF SCROLL SHIFT OR ABOVE, TOGGLE
                                                                ; KEY
                                        ;------ PLAIN SHIFT KEY, SET SHIFT ON
15B4  08 26 0017 R                      OR      KB_FLAG,AH      ; TURN ON SHIFT BIT
15B8  E9 164A R                         JMP     K26             ; INTERRUPT_RETURN
                                        ;------ TOGGLED SHIFT KEY, TEST FOR 1ST MAKE OR NOT
15BB                            K18:                            ; SHIFT-TOGGLE
15BB  F6 06 0017 R 04                   TEST    KB_FLAG, CTL_SHIFT ; CHECK CTL SHIFT STATE
15C0  75 78                             JNZ     K25             ; JUMP IF CTL STATE
15C2  3C 52                             CMP     AL, INS_KEY     ; CHECK FOR INSERT KEY
15C4  75 22                             JNZ     K22             ; JUMP IF NOT INSERT KEY
15C6  F6 06 0017 R 08                   TEST    KB_FLAG, ALT_SHIFT ; CHECK FOR ALTERNATE SHIFT
15CB  75 6D                             JNZ     K25             ; JUMP IF ALTERNATE SHIFT
15CD  F6 06 0017 R 20                   TEST    KB_FLAG, NUM_STATE ; CHECK FOR BASE STATE
15D2  75 0D                             JNZ     K21             ; JUMP IF NUM LOCK IS ON
15D4  F6 06 0017 R 03                   TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT ;
15D9  74 0D                             JZ      K22             ; JUMP IF BASE STATE
15DB                            K20:                            ; NUMERIC ZERO, NOT INSERT KEY
15DB  B8 5230                           MOV     AX, 5230H       ; PUT OUT AN ASCII ZERO
15DE  E9 17EC R                         JMP     K57             ; BUFFER_FILL
15E1                            K21:                            ; MIGHT BE NUMERIC
15E1  F6 06 0017 R 03                   TEST    KB_FLAG, LEFT_SHIFT+ RIGHT_SHIFT
15E6  74 F3                             JZ      K20             ; JUMP NUMERIC, NOT INSERT
15E8                            K22:                            ; SHIFT TOGGLE KEY HIT; PROCESS IT
15E8  84 26 0018 R                      TEST    AH,KB_FLAG_1    ; IS KEY ALREADY DEPRESSED
15EC  75 5C                             JNZ     K26             ; JUMP IF KEY ALREADY DEPRESSED
15EE  08 26 0018 R                      OR      KB_FLAG_1,AH    ; INDICATE THAT THE KEY IS
                                                                ; DEPRESSED
15F2  30 26 0017 R                      XOR     KB_FLAG,AH      ; TOGGLE THE SHIFT STATE
15F6  3C 52                             CMP     AL,INS_KEY      ; TEST FOR 1ST MAKE OF INSERT KEY
15F8  75 50                             JNE     K26             ; JUMP IF NOT INSERT KEY
15FA  B8 5200                           MOV     AX,INS_KEY*256  ; SET SCAN CODE INTO AH, 0 INTO AL
15FD  E9 17EC R                         JMP     K57             ; PUT INTO OUTPUT BUFFER
```

```
                        ;------ BREAK SHIFT FOUND
1600                    K23:                            ; BREAK-SHIFT-FOUND
1600  80 FC 10                  CMP     AH,SCROLL_SHIFT ; IS THIS A TOGGLE KEY
1603  73 1A                     JAE     K24             ; YES, HANDLE BREAK TOGGLE
1605  F6 D4                     NOT     AH              ; INVERT MASK
1607  20 26 0017 R              AND     KB_FLAG,AH      ; TURN OFF SHIFT BIT
160B  3C B8                     CMP     AL,ALT_KEY+80H  ; IS THIS ALTERNATE SHIFT RELEASE
160D  75 3B                     JNE     K26             ; INTERRUPT_RETURN
                        ;------ ALTERNATE SHIFT KEY RELEASED, GET THE VALUE INTO BUFFER
160F  A0 0019 R                 MOV     AL,ALT_INPUT
1612  32 E4                     XOR     AH,AH           ; SCAN CODE OF 0
1614  88 26 0019 R              MOV     ALT_INPUT,AH    ; ZERO OUT THE FIELD
1618  0A C0                     OR      AL,AL           ; WAS THE INPUT=0?
161A  74 2E                     JE      K26             ; INTERRUPT_RETURN
161C  E9 17F5 R                 JMP     K58             ; IT WASN'T, SO PUT IN BUFFER
161F                    K24:                            ; BREAK-TOGGLE
161F  3C BA                     CMP     AL,CAPS_KEY+BREAK_BIT ; SPECIAL CASE OF TOGGLE KEY
1621  75 0F                     JNE     K24_1           ; JUMP AROUND POTENTIAL UPDATE
1623  F6 06 0018 R 02           TEST    KB_FLAG_1,CLICK_SEQUENCE
1628  74 08                     JZ      K24_1           ; JUMP IF NOT SPECIAL CASE
162A  80 26 0018 R FD           AND     KB_FLAG_1,AND_MASK-CLICK_SEQUENCE ; MASK OFF MAKE
                                                        ; OF CLICK
162F  EB 19 90                  JMP     K26             ; INTERRUPT IS OVER
                        ;------ BREAK OF NORMAL TOGGLE
1632  F6 D4             K24_1:  NOT     AH              ; INVERT MASK
1634  20 26 0018 R              AND     KB_FLAG_1,AH    ; INDICATE NO LONGER DEPRESSED
1638  EB 10                     JMP     SHORT K26       ; INTERRUPT_RETURN
                        ;------ TEST FOR HOLD STATE
163A                    K25:                            ; NO-SHIFT-FOUND
163A  3C 80                     CMP     AL,80H          ; TEST FOR BREAK KEY
163C  73 0C                     JAE     K26             ; NOTHING FOR BREAK CHARS FROM HERE
                                                        ; ON
163E  F6 06 0018 R 08           TEST    KB_FLAG_1,HOLD_STATE ; ARE WE IN HOLD STATE?
1643  74 0E                     JZ      K28             ; BRANCH AROUND TEST IF NOT
1645  80 26 0018 R F7           AND     KB_FLAG_1,NOT HOLD_STATE ; TURN OFF THE HOLD STATE
                                                        ; BIT
164A                    K26:                            ; INTERRUPT-RETURN
164A  07                        POP     ES
164B  1F                        POP     DS
164C  5F                        POP     DI
164D  5E                        POP     SI
164E  5A                        POP     DX
164F  59                        POP     CX
1650  5B                        POP     BX
1651  58                        POP     AX              ; RESTORE STATE
1652  CF                        IRET                    ; RETURN, INTERRUPTS BACK ON WITH
                                                        ; FLAG CHANGE
                        ;------ NOT IN  HOLD STATE, TEST FOR SPECIAL CHARS
1653                    K28:                            ; NO-HOLD-STATE
1653  F6 06 0017 R 08           TEST    KB_FLAG,ALT_SHIFT ; ARE WE IN ALTERNATE SHIFT
1658  75 03                     JNZ     K29             ; JUMP IF ALTERNATE SHIFT
165A  E9 1749 R                 JMP     K38             ; JUMP IF NOT ALTERNATE
                        ;------ TEST FOR ALT+CTRL KEY SEQUENCES
165D                    K29:                            ; TEST-RESET
165D  F6 06 0017 R 04           TEST    KB_FLAG,CTL_SHIFT ; ARE WE IN CONTROL SHIFT ALSO
1662  74 69                     JZ      K31             ; NO_RESET
1664  3C 53                     CMP     AL,DEL_KEY      ; SHIFT STATE IS THERE, TEST KEY
1666  75 09                     JNE     K29_1           ; NO_RESET
                        ;------ CTL-ALT-DEL HAS BEEN FOUND, DO I/O CLEANUP
1668  C7 06 0072 R 1234        MOV     RESET_FLAG, 1234H ; SET FLAG FOR RESET FUNCTION
166E  E9 0043 R                 JMP     NEAR PTR RESET  ; JUMP TO POWER ON DIAGNOSTICS
1671  3C 52             K29_1:  CMP     AL,INS_KEY      ; CHECK FOR RESET WITH DIAGNOSTICS
1673  75 09                     JNE     K29_2           ; CHECK FOR OTHER
                                                        ; ALT-CTRL-SEQUENCES
                        ;------ ALT-CTRL-INS HAS BEEN FOUND
1675  C7 06 0072 R 4321        MOV     RESET_FLAG,4321H ; SET FLAG FOR DIAGNOSTICS
167B  E9 0043 R                 JMP     NEAR PTR RESET  ; LEVEL 1 DIAGNOSTICS
167E  3C 3A             K29_2:  CMP     AL,CAPS_KEY     ; CHECK FOR KEYBORAD CLICK TOGGLE
1680  75 13                     JNE     K29_3           ; CHECK FOR SCREEN ADJUSTMENT
                        ;------ ALT+CTRL+CAPSLOCK HAS BEEN FOUND
1682  F6 06 0018 R 02          TEST    KB_FLAG_1,CLICK_SEQUENCE
1687  75 C1                     JNZ     K26             ; JUMP IF SEQUENCE HAS ALREADY
                                                        ; OCCURED
1689  80 36 0018 R 04           XOR     KB_FLAG_1,CLICK_ON ; TOGGLE BIT FOR AUDIO KEYSTROKE
                                                        ; FEEDBACK
168E  80 0E 0018 R 02           OR      KB_FLAG_1,CLICK_SEQUENCE ; SET CLICK_SEQUENCE STATE
1693  EB B5                     JMP     SHORT K26       ; INTERRUPT IS OVER
1695  3C 4D             K29_3:  CMP     AL,RIGHT_ARROW  ; ADJUST SCREEN TO THE RIGHT?
1697  75 12                     JNE     K29_4           ; LOOK FOR RIGHT ADJUSTMENT
1699  E8 186E R                 CALL    GET_POS         ; GET THE # OF POSITIONS SCREEN IS
                                                        ; SHIFTED
169C  3C FC                     CMP     AL,0-RANGE      ; IS SCREEN SHIFTED AS FAR AS
                                                        ; POSSIBLE?
169E  7C AA                     JL      K26             ; OUT OF RANGE
16A0  FE 0E 0089 R              DEC     HORZ_POS        ; SHIFT VALUE TO THE RIGHT
16A4  FE C8                     DEC     AL              ; DECREASE RANGE VALUE
16A6  E8 187A R                 CALL    PUT_POS         ; RESTORE STORAGE LOCATION
16A9  EB 14                     JMP     SHORT K29_5     ; ADJUST
16AB  3C 4B             K29_4:  CMP     AL,LEFT_ARROW   ; ADJUST SREEN TO THE LEFT?
16AD  75 1E                     JNE     K31             ; NOT AN ALT_CTRL SEQUENCE
16AF  E8 186E R                 CALL    GET_POS         ; GET NUMBER OF POSITIONS SCREEN IS
                                                        ; SHIFTED
16B2  3C 04                     CMP     AL,RANGE        ; IS SCREEN SHIFTED AS FAR AS
                                                        ; POSSIBLE?
16B4  7F 94                     JG      K26
16B6  FE 06 0089 R              INC     HORZ_POS        ; SHIFT SCREEN TO THE LEFT
16BA  FE C0                     INC     AL              ; INCREASE NUMBER OF POSITIONS
                                                        ; SCREEN IS SHIFTED
16BC  E8 187A R                 CALL    PUT_POS         ; PUT POSTION BACK IN STORAGE
```

```
16BF  B0 02                   K29_5:  MOV     AL,2                    ; ADJUST
16C1  BA 03D4                         MOV     DX,3D4H                 ; ADDRESS TO CRT CONTROLLER
16C4  EE                              OUT     DX,AL
16C5  A0 0089 R                       MOV     AL,HORZ_POS             ; COLUMN POSITION
16C8  42                              INC     DX                      ; POINT AT DATA REGISTER
16C9  EE                              OUT     DX,AL                   ; MOV POSITION
16CA  E9 164A R                       JMP     K26
                              ;------ IN ALTERNATE SHIFT, RESET NOT FOUND
16CD                          K31:                                    ; NO-RESET
16CD  3C 39                           CMP     AL,57                   ; TEST FOR SPACE KEY
16CF  75 29                           JNE     K32                     ; NOT THERE
16D1  B0 20                           MOV     AL,' '                  ; SET SPACE CHAR
16D3  E9 17EC R                       JMP     K57                     ; BUFFER_FILL
                              ;-----  ALT-INPUT-TABLE
16D6                          K30     LABEL   BYTE
16D6  52 4F 50 51 4B 4C       K30     DB      82,79,80,81,75,76,77
      4D
16DD  47 48 49                        DB      71,72,73                ; 10 NUMBERS ON KEYPAD
                              ;------ SUPER-SHIFT-TABLE
16E0  10 11 12 13 14 15               DB      16,17,18,19,20,21,22,23 ; A-Z TYPEWRITER CHARS
      16 17
16E8  18 19 1E 1F 20 21               DB      24,25,30,31,32,33,34,35
      22 23
16F0  24 25 26 2C 2D 2E               DB      36,37,38,44,45,46,47,48
      2F 30
16F8  31 32                           DB      49,50
                              ;------ LOOK FOR KEY PAD ENTRY
16FA                          K32:                                    ; ALT-KEY-PAD
16FA  BF 16D6 R                       MOV     DI,OFFSET K30           ; ALT-INPUT-TABLE
16FD  B9 000A                         MOV     CX,10                   ; LOOK FOR ENTRY USING KEYPAD
1700  F2/ AE                         REPNE   SCASB                   ; LOOK FOR MATCH
1702  75 13                          JNE     K33                     ; NO_ALT_KEYPAD
1704  81 EF 16D7 R                    SUB     DI,OFFSET K30+1         ; DI NOW HAS ENTRY VALUE
1708  A0 0019 R                       MOV     AL,ALT_INPUT            ; GET THE CURRENT BYTE
170B  B4 0A                           MOV     AH,10                   ; MULTIPLY BY 10
170D  F6 E4                           MUL     AH
170F  03 C7                           ADD     AX,DI                   ; ADD IN THE LATEST ENTRY
1711  A2 0019 R                       MOV     ALT_INPUT,AL            ; STORE IT AWAY
1714  E9 164A R                       JMP     K26                     ; THROW AWAY THAT KEYSTROKE
                              ;------ LOOK FOR SUPERSHIFT ENTRY
1717                          K33:                                    ; NO-ALT-KEYPAD
1717  C6 06 0019 R 00                 MOV     ALT_INPUT,0             ; ZERO ANY PREVIOUS ENTRY INTO
                                                                      ; INPUT
171C  B9 001A                         MOV     CX,26                   ; DI,ES ALREADY POINTING
171F  F2/ AE                          REPNE   SCASB                   ; LOOK FOR MATCH IN ALPHABET
1721  75 05                          JNE     K34                     ; NOT FOUND, FUNCTION KEY OR OTHER
1723  32 C0                          XOR     AL,AL                   ; ASCII CODE OF ZERO
1725  E9 17EC R                       JMP     K57                     ; PUT IT IN THE BUFFER
                              ;------ LOOK FOR TOP ROW OF ALTERNATE SHIFT
1728                          K34:                                    ; ALT-TOP-ROW
1728  3C 02                          CMP     AL,2                    ; KEY WITH '1' ON IT
172A  72 0C                          JB      K35                     ;  NOT ONE OF INTERESTING KEYS
172C  3C 0E                          CMP     AL,14                   ;  IS IT IN THE REGION?
172E  73 08                          JAE     K35                     ;  ALT-FUNCTION
1730  80 C4 76                       ADD     AH,118                  ;  CONVERT PSUEDO SCAN CODE TO
                                                                      ; RANGE
1733  32 C0                          XOR     AL,AL                   ;  INDICATE AS SUCH
1735  E9 17EC R                       JMP     K57                     ;  BUFFER_FILL
                              ;------ TRANSLATE ALTERNATE SHIFT PSEUDO SCAN CODES
1738                          K35:                                    ; ALT-FUNCTION
1738  3C 3B                          CMP     AL,59                   ; TEST FOR IN TABLE
173A  73 03                          JAE     K37                     ; ALT-CONTINUE
173C                          K36:                                    ; CLOSE-RETURN
173C  E9 164A R                       JMP     K26                     ; IGNORE THE KEY
173F                          K37:                                    ; ALT-CONTINUE
173F  3C 47                          CMP     AL,71                   ; IN KEYPAD REGION
1741  73 F9                          JAE     K36                     ; IF SO, IGNORE
1743  BB 153D R                       MOV     BX,OFFSET K13           ; ALT SHIFT PSEUDO SCAN TABLE
1746  E9 1863 R                       JMP     K63                     ; TRANSLATE THAT
                              ;------ NOT IN ALTERNATE SHIFT
1749                          K38:                                    ; NOT-ALT-SHIFT
1749  F6 06 0017 R 04                 TEST    KB_FLAG,CTL_SHIFT       ; ARE WE IN CONTROL SHIFT?
174E  74 34                           JZ      K44                     ; NOT-CTL-SHIFT
                              ;------ CONTROL SHIFT, TEST SPECIAL CHARACTERS
                              ;------ TEST FOR BREAK AND PAUSE KEYS
1750  3C 46                          CMP     AL,SCROLL_KEY           ; TEST FOR BREAK
1752  75 19                          JNE     K41                     ; NO-BREAK
1754  8B 1E 001A R                    MOV     BX,BUFFER_HEAD          ; GET CURRENT BUFFER HEAD
1758  C6 06 0071 R 80                MOV     BIOS_BREAK,80H          ; TURN ON BIOS_BREAK BIT
175D  CD 1B                          INT     1BH                     ; BREAK INTERRUPT VECTOR
175F  2B C0                          SUB     AX,AX                   ; PUT OUT DUMMY CHARACTER
1761  89 07                          MOV     [BX],AX                 ; PUT DUMMY CHAR AT BUFFER HEAD
1763  E8 144F R                       CALL    K4                      ; UPDATE BUFFER POINTER
1766  89 1E 001C R                    MOV     BUFFER_TAIL,BX          ; UPDATE TAIL
176A  E9 164A R                       JMP     K26                     ; DONE WITH INTERUPT
176D                          K41:                                    ; NO-PAUSE
                              ;------ TEST SPECIAL CASE KEY 55
176D  3C 37                          CMP     AL,55
176F  75 06                          JNE     K42                     ; NOT-KEY-55
1771  B8 7200                        MOV     AX,114*256              ; START/STOP PRINTING SWITCH
1774  EB 76 90                       JMP     K57                     ; BUFFER_FILL
```

```
                                ;------ SET UP TO TRANSLATE CONTROL SHIFT
1777                            K42:                            ; NOT-KEY-55
1777 BB 146C R                          MOV     BX,OFFSET K8    ; SET UP TO TRANSLATE CTL
177A 3C 3B                              CMP     AL,59           ; IS IT IN TABLE?
177C 72 6A                              JB      K56             ; YES, GO TRANSLATE CHAR
                                                                ; CTL-TABLE-TRANSLATE
177E BB 14A6 R                          MOV     BX,OFFSET K9    ; CTL TABLE SCAN
1781 E9 1863 R                          JMP     K63             ; TRANSLATE_SCAN
                                ;------ NOT IN CONTROL SHIFT
1784                            K44:                            ; NOT-CTL-SHIFT
1784 3C 47                              CMP     AL,71           ; TEST FOR KEYPAD REGION
1786 73 1F                              JAE     K48             ; HANDLE KEYPAD REGION
1788 F6 06 0017 R 03                    TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
178D 74 4E                              JZ      K54             ; TEST FOR SHIFT STATE
                                ;------ UPPER CASE, HANDLE SPECIAL CASES
178F 3C 0F                              CMP     AL,15           ; BACK TAB KEY
1791 75 05                              JNE     K46             ; NOT-BACK-TAB
1793 B8 0F00                            MOV     AX,15*256       ; SET PSEUDO SCAN CODE
1796 EB 54                              JMP     SHORT K57       ; BUFFER_FILL
1798                            K46:                            ; NOT-PRINT-SCREEN
1798 3C 3B                              CMP     AL,59           ; FUNCTION KEYS
179A 72 06                              JB      K47             ; NOT-UPPER-FUNCTION
179C BB 1533 R                          MOV     BX,OFFSET K12   ; UPPER CASE PSEUDO SCAN CODES
179F E9 1863 R                          JMP     K63             ; TRANSLATE_SCAN
17A2                            K47:                            ; NOT-UPPER-FUNCTION
17A2 BB 14F9 R                          MOV     BX,OFFSET K11   ; POINT TO UPPER CASE TABLE
17A5 EB 41                              JMP     SHORT K56       ; OK, TRANSLATE THE CHAR
                                ;------ KEYPAD KEYS, MUST TEST NUM LOCK FOR DETERMINATION
17A7                            K48:                            ; KEYPAD-REGION
17A7 F6 06 0017 R 20                    TEST    KB_FLAG,NUM_STATE ; ARE WE IN NUM_LOCK?
17AC 75 21                              JNZ     K52             ; TEST FOR SURE
17AE F6 06 0017 R 03                    TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; ARE WE IN SHIFT
                                                                ; STATE
17B3 75 21                              JNZ     K53             ; IF SHIFTED, REALLY NUM STATE
                                ;------ BASE CASE FOR KEYPAD
17B5                            K49:                            ; BASE-CASE
17B5 3C 4A                              CMP     AL,74           ; SPECIAL CASE FOR A COUPLE OF KEYS
17B7 74 0C                              JE      K50             ; MINUS
17B9 3C 4E                              CMP     AL,78
17BB 74 0D                              JE      K51
17BD 2C 47                              SUB     AL,71           ; CONVERT ORIGIN
17BF BB 1554 R                          MOV     BX,OFFSET K15   ; BASE CASE TABLE
17C2 E9 1865 R                          JMP     K64             ; CONVERT TO PSEUDO SCAN
17C5 B8 4A2D                    K50:    MOV     AX,74*256+'-'   ; MINUS
17C8 EB 22                              JMP     SHORT K57       ; BUFFER_FILL
17CA B8 4E2B                    K51:    MOV     AX,78*256+'+'   ; PLUS
17CD EB 1D                              JMP     SHORT K57       ; BUFFER_FILL
                                ;------ MIGHT BE NUM LOCK, TEST SHIFT STATUS
17CF                            K52:                            ; ALMOST-NUM-STATE
17CF F6 06 0017 R 03                    TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT
17D4 75 0F                              JNZ     K49             ; SHIFTED TEMP OUT OF NUM STATE
17D6                            K53:                            ; REALLY_NUM_STATE
17D6 2C 46                              SUB     AL,70           ; CONVERT ORIGIN
17D8 BB 1547 R                          MOV     BX,OFFSET K14   ; NUM STATE TABLE
17DB EB 0B                              JMP     SHORT K56       ; TRANSLATE_CHAR
                                ;------ PLAIN OLD LOWER CASE
17DD                            K54:                            ; NOT-SHIFT
17DD 3C 3B                              CMP     AL,59           ; TEST FOR FUNCTION KEYS
17DF 72 04                              JB      K55             ; NOT-LOWER-FUNCTION
17E1 32 C0                              XOR     AL,AL           ; SCAN CODE IN AH ALREADY
17E3 EB 07                              JMP     SHORT K57       ; BUFFER_FILL
17E5                            K55:                            ; NOT-LOWER-FUNCTION
17E5 BB 148F R                          MOV     BX,OFFSET K10   ; LC TABLE
                                ;------ TRANSLATE THE CHARACTER
17E8                            K56:                            ; TRANSLATE-CHAR
17E8 FE C8                              DEC     AL              ; CONVERT ORIGIN
17EA 2E: D7                             XLAT    CS:K11          ; CONVERT THE SCAN CODE TO ASCII
                                ;------ PUT CHARACTER INTO BUFFER
17EC                            K57:                            ; BUFFER-FILL
17EC 3C FF                              CMP     AL,-1           ; IS THIS AN IGNORE CHAR?
17EE 74 1F                              JE      K59             ; YES, DO NOTHING WITH IT
17F0 80 FC FF                           CMP     AH,-1           ; LOOK FOR -1 PSEUDO SCAN
17F3 74 1A                              JE      K59             ; NEAR_INTERRUPT_RETURN
                                ;------ HANDLE THE CAPS LOCK PROBLEM
17F5                            K58:                            ; BUFFER-FILL-NOTEST
17F5 F6 06 0017 R 40                    TEST    KB_FLAG,CAPS_STATE ; ARE WE IN CAPS LOCK STATE?
17FA 74 20                              JZ      K61             ; SKIP IF NOT
                                ;------ IN CAPS LOCK STATE
17FC F6 06 0017 R 03                    TEST    KB_FLAG,LEFT_SHIFT+RIGHT_SHIFT ; TEST FOR SHIFT
                                                                ; STATE
1801 74 0F                              JZ      K60             ; IF NOT SHIFT, CONVERT LOWER TO
                                                                ; UPPER
                                ;------ CONVERT ANY UPPER CASE TO LOWER CASE
1803 3C 41                              CMP     AL,'A'          ; FIND OUT IF ALPHABETIC
1805 72 15                              JB      K61             ; NOT_CAPS_STATE
1807 3C 5A                              CMP     AL,'Z'
1809 77 11                              JA      K61             ; NOT_CAPS_STATE
180B 04 20                              ADD     AL,'a'-'A'      ; CONVERT TO LOWER CASE
180D EB 0D                              JMP     SHORT K61       ; NOT_CAPS_STATE
180F                            K59:                            ; NEAR-INTERRUPT-RETURN
180F E9 164A R                          JMP     K26             ; INTERRUPT_RETURN
                                ;------ CONVERT ANY LOWER CASE TO UPPER CASE
1812                            K60:                            ; LOWER-TO-UPPER
1812 3C 61                              CMP     AL,'a'          ; FIND OUT IF ALPHABETIC
1814 72 06                              JB      K61             ; NOT_CAPS_STATE
1816 3C 7A                              CMP     AL,'z'
1818 77 02                              JA      K61             ; NOT_CAPS_STATE
181A 2C 20                              SUB     AL,'a'-'A'      ; CONVERT TO UPPER CASE
```

```
181C                                    K61:                            ; NOT-CAPS-STATE
181C  8B 1E 001C R                             MOV     BX,BUFFER_TAIL  ; GET THE END POINTER TO THE BUFFER
1820  8B F3                                    MOV     SI,BX           ; SAVE THE VALUE
1822  E8 144F R                                CALL    K4              ; ADVANCE THE TAIL
1825  3B 1E 001A R                             CMP     BX,BUFFER_HEAD  ; HAS THE BUFFER WRAPPED AROUND?
1829  75 1D                                    JNE     K61_1           ; BUFFER_FULL_BEEP
182B  53                                       PUSH    BX              ; SAVE BUFFER_TAIL
182C  BB 0080                                  MOV     BX,080H         ; DURATION OF ERROR BEEP
182F  B9 0048                                  MOV     CX,48H          ; FREQUENCY OF ERROR BEEP HALF TONE
1832  E8 E035 R                                CALL    KB_NOISE        ; OUTPUT NOISE
1835  80 26 0017 R F0                         AND     KB_FLAG,0F0H    ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                                       ; SHIFTS
183A  80 26 0018 R 0F                          AND     KB_FLAG_1,0FH   ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                                       ; ,NUM AND SCROLL SHIFT
183F  80 26 0088 R 1F                          AND     KB_FLAG_2,1FH   ; CLEAR FUNCTION STATES
1844  5B                                       POP     BX              ; RETRIEVE BUFFER TAIL
1845  E9 164A R                                JMP     K26             ; RETURN FROM INTERRUPT
1848  F6 06 0018 R 04                 K61_1:  TEST    KB_FLAG_1,CLICK_ON ; IS AUDIO FEEDBACK ENABLED?
184D  74 0B                                    JZ      K61_2           ; NO, JUST PUT IN BUFFER
184F  53                                       PUSH    BX              ; SAVE BUFFER_TAIL VALUE
1850  BB 0001                                  MOV     BX,1H           ; DURATION OF CLICK
1853  B9 0010                                  MOV     CX,10H          ; FREQUENCY OF CLICK
1856  E8 E035 R                                CALL    KB_NOISE        ; OUTPUT AUDIO FEEDBACK OF KEY
                                                                       ; STROKE
1859  5B                                       POP     BX              ; RETRIEVE BUFFER_TAIL VALUE
185A  89 04                             K61_2:  MOV     [SI],AX         ; STORE THE VALUE
185C  89 1E 001C R                             MOV     BUFFER_TAIL,BX  ; MOVE THE POINTER UP
1860  E9 164A R                                JMP     K26             ; INTERRUPT_RETURN
                                       ;------ TRANSLATE SCAN FOR PSEUDO SCAN CODES
1863                                    K63:                            ; TRANSLATE-SCAN
1863  2C 3B                                    SUB     AL,59           ; CONVERT ORIGIN TO FUNCTION KEYS
1865                                    K64:                            ; TRANSLATE-SCAN-ORGD
1865  2E: D7                                   XLAT    CS:K9           ; CTL TABLE SCAN
1867  8A E0                                    MOV     AH,AL           ; PUT VALUE INTO AH
1869  32 C0                                    XOR     AL,AL           ; ZERO ASCII CODE
186B  E9 17EC R                                JMP     K57             ; PUT IT INTO THE BUFFER
186E                                    KB_INT  ENDP
                                       ;--------------------------------------------------------------------------
                                       ;GET_POS
                                       ;       THIS ROUTINE WILL SHIFT THE VALUE STORED IN THE HIGH NIBBLE
                                       ;       OF THE VARIABLE VAR_DELAY TO THE LOW NIBBLE.
                                       ;INPUT
                                       ;       NONE.  IT IS ASSUMED THAT DS POINTS AT THE BIOS DATA AREA
                                       ;OUTPUT
                                       ;       AL CONTAINS THE SHIFTED VALUE.
                                       ;--------------------------------------------------------------------------
186E                                    GET_POS PROC    NEAR
186E  51                                       PUSH    CX              ; SAVE SHIFT REGISTER
186F  A0 0086 R                                MOV     AL,BYTE PTR VAR_DELAY ; GET STORAGE LOCATION
1872  24 F0                                    AND     AL,0F0H         ; MASK OFF LOW NIBBLE
1874  B1 04                                    MOV     CL,4            ; SHIFT OF FOUR BIT POSITIONS
1876  D2 F8                                    SAR     AL,CL           ; SHIFT THE VALUE SIGN EXTENDED
1878  59                                       POP     CX              ; RESTORE THE VALUE
1879  C3                                       RET
187A                                    GET_POS ENDP
                                       ;--------------------------------------------------------------------------
                                       ;PUT_POS
                                       ;       THIS ROUTINE WILL TAKE THE VALUE IN LOW ORDER NIBBLE IN
                                       ;       AL AND STORE IT IN THE HIGH ORDER OF VAR_DELAY
                                       ;INPUT
                                       ;       AL CONTAINS THE VALUE FOR STORAGE
                                       ;OUTPUT
                                       ;       NONE.
                                       ;--------------------------------------------------------------------------
187A                                    PUT_POS PROC    NEAR
187A  51                                       PUSH    CX              ; SAVE REGISTER
187B  B1 04                                    MOV     CL,4            ; SHIFT COUNT
187D  D2 E0                                    SHL     AL,CL           ; PUT IN HIGH ORDER NIBBLE
187F  8A 0E 0086 R                             MOV     CL,BYTE PTR VAR_DELAY ; GET DATA BYTE
1883  80 E1 0F                                 AND     CL,0FH          ; CLEAR OLD VALUE IN HIGH NIBBLE
1886  0A C1                                    OR      AL,CL           ; COMBINE HIGH AND LOW NIBBLES
1888  A2 0086 R                                MOV     BYTE PTR VAR_DELAY,AL ; PUT IN POSITION
188B  59                                       POP     CX              ; RESTORE REGISTER
188C  C3                                       RET
188D                                    PUT_POS ENDP
                                       ;--------------------------------------------------------------------------
                                       ; MANUFACTURING ACTIVITY SIGNAL ROUTINE - INVOKED THROUGH THE TIMER
                                       ; TICK ROUTINE DURING MANUFACTRUING ACTIVITIES . (ACCESSED THROUGH
                                       ; INT 1CH)
                                       ;--------------------------------------------------------------------------
188D                                    MFG_TICK        PROC    FAR
188D  50                                       PUSH    AX
188E  2B C0                                    SUB     AX,AX           ; SEND A 00 TO PORT 13 AS A
                                                                       ; ACTIVITY SIGNAL
1890  E6 13                                    OUT     13H,AL
1892  E4 61                                    IN      AL,PORT_B       ; FLIP SPEAKER DATA TO OPPOSITE
                                                                       ; SENSE
1894  8A E0                                    MOV     AH,AL           ; SAVE ORIG SETTING
1896  80 E4 9D                                 AND     AH,10011101B    ; MAKE SURE MUX IS -> RIGHT AND
                                                                       ; ISOLATE SPEAKER BIT
1899  F6 D0                                    NOT     AL              ; FLIP ALL BITS
189B  24 02                                    AND     AL,00000010B    ; ISOLATE SPEAKER DATA BIT (NOW IN
                                                                       ; OPPOSITE SENSE)
189D  0A C4                                    OR      AL,AH           ; COMBINE WITH ORIG. DATA FROM
                                                                       ; PORT B
189F  0C 10                                    OR      AL,00010000B    ; AND DISABLE INTERNAL SPEAKER
18A1  E6 61                                    OUT     PORT_B,AL
18A3  B0 20                                    MOV     AL,20H          ; EOI TO INTR. CHIP
18A5  E6 20                                    OUT     20H,AL
18A7  58                                       POP     AX
18A8  CF                                       IRET
18A9                                    MFG_TICK        ENDP
```

```
                        ;--------------------------------------------------------------
                        ;       CONVERT AND PRINT ASCII CODE
                        ;
                        ;       AL MUST CONTAIN NUMBER TO BE CONVERTED.
                        ;       AX AND BX DESTROYED.
                        ;--------------------------------------------------------------
18A9                    XPC_BYTE        PROC    NEAR
18A9  50                        PUSH    AX              ; SAVE FOR LOW NIBBLE DISPLAY
18AA  B1 04                     MOV     CL,4            ; SHIFT COUNT
18AC  D2 E8                     SHR     AL,CL           ; NIBBLE SWAP
18AE  E8 18B4 R                 CALL    XLAT_PR         ; DO THE HIGH NIBBLE DISPLAY
18B1  58                        POP     AX              ; RECOVER THE NIBBLE
18B2  24 0F                     AND     AL,0FH          ; ISOLATE TO LOW NIBBLE
                                                        ; FALL INTO LOW NIBBLE CONVERSION
18B4                    XLAT_PR PROC    NEAR            ; CONVERT 00-0F TO ASCII CHARACTER
18B4  04 90                     ADD     AL,090H         ; ADD FIRST CONVERSION FACTOR
18B6  27                        DAA                     ; ADJUST FOR NUMERIC AND ALPHA
                                                        ; RANGE
18B7  14 40                     ADC     AL,040H         ; ADD CONVERSION AND ADJUST LOW
                                                        ; NIBBLE
18B9  27                        DAA                     ; ADJUST HIGH NIBBLE TO ASCII RANGE
18BA                    PRT_HEX PROC    NEAR
18BA  53                        PUSH    BX
18BB  B4 0E                     MOV     AH,14           ; DISPLAY CHARACTER IN AL
18BD  B7 00                     MOV     BH,0
18BF  CD 10                     INT     10H             ; CALL VIDEO_IO
18C1  5B                        POP     BX
18C2  C3                        RET
18C3                    PRT_HEX ENDP
18C3                    XLAT_PR ENDP
18C3                    XPC_BYTE        ENDP
                        ;CONTROL IS PASSED HERE WHEN THERE ARE NO PARALLEL PRINTERS
                        ;ATTACHED. CX HAS EQUIPMENT FLAG,DS POINTS AT DATA (40H)
                        ;DETERMINE WHICH RS232 CARD (0,1) TO USE
18C3                    REPRINT PROC NEAR
18C3  2B D2             B1_A:   SUB     DX,DX           ;ASSUME TO USE CARD 0
18C5  F6 C5 04                  TEST    CH,00000100B    ;UNLESS THERE ARE TWO CARDS
18C8  74 01                     JE      B10_1           ;IN WHICH CASE,
18CA  42                        INC     DX              ;USE CARD 1
                        ;DETERMINE WHICH FUNCTION IS BEING CALLED
18CB  0A E4             B10_1:  OR      AH,AH           ;TEST FOR AH = 0
18CD  74 41                     JZ      B12             ;GO PRINT CHAR
18CF  FE CC                     DEC     AH              ;TEST FOR AH = 1
18D1  74 1D                     JZ      B11             ;GO DO INIT
18D3  FE CC                     DEC     AH              ;TEST FOR AH = 2
18D5  75 16                     JNZ     SHORT B10_3     ;IF NOT VALID, RETURN
                                                        ;ELSE...
                        ;GET STATUS FROM RS232 PORT
18D7  50                        PUSH    AX              ;SAVE AL
18D8  B4 03                     MOV     AH,03H          ;USE THE GET COMMO PORT
18DA  CD 14                     INT     014H            ;STATUS FUNCTION OF INT14
18DC  E8 1925 R                 CALL    FAKE            ;FAKE WILL MAP ERROR BITS FROM
                                                        ;RS232 TO CORRESPONDING ONES
                                                        ;FOR THE PRINTER
18DF  58                        POP     AX              ;RESTORE AL
18E0  0A F6                     OR      DH,DH           ;CHECK IF ANY FLAGS WERE SET
18E2  74 07                     JZ      B10_2
18E4  8A E6                     MOV     AH,DH           ;MOVE FAKED ERROR CONDITION TO AH
18E6  80 E4 FE                  AND     AH,0FEH
18E9  EB 02                     JMP     SHORT B10_3     ;THEN RETURN
18EB  B4 90             B10_2:  MOV     AH,090H         ;MOVE IN STATUS FOR 'CORRECT'
                                                        ; RETURN
18ED  E9 F00D R         B10_3:  JMP     B1
                        ;INIT COMMO PORT    --- DX HAS WHICH CARD TO INIT.
                        ;MOVE TIME OUT VALUE FROM PRINTER TO RS232 TIME OUT VALUE
18F0  8B F2             B11:    MOV     SI,DX           ;SI GETS OFFSET INTO THE TABLE
18F2  A0 0078 R                 MOV     AL,PRINT_TIM_OUT
18F5  04 0A                     ADD     AL,0AH          ; INCREASE DELAY
18F7  88 84 007C R              MOV     RS232_TIM_OUT[SI],AL
18FB  50                        PUSH    AX              ;SAVE AL
18FC  B0 87                     MOV     AL,087H         ;SET INIT FOR:  1200 BAUD
                                                        ;               8 BIT WRD LNG
                                                        ;               NO PARITY
                                                        ;               2 STOP BITS
18FE  2A E4                     SUB     AH,AH           ;AH=0 IS COMMO INIT FUNCTION
1900  CD 14                     INT     014H            ;DO INIT
1902  E8 1925 R                 CALL    FAKE            ;FAKE WILL MAP ERROR BITS FROM
                                                        ;RS232 TO CORRESPONDING ONES
                                                        ;FOR THE PRINTER
1905  58                        POP     AX              ;RESTORE AL
1906  8A E6                     MOV     AH,DH           ;IF DH IS RETURNED ZERO, MEANING
1908  0A E4                     OR      AH,AH           ;NO ERRORS RETURN IT FOR THAT'S THE
                                                        ; 'CORRECT' RETURN FROM AN ERROR
                                                        ; FREE INIT
190A  74 E1                     JE      B10_3
190C  B4 A8                     MOV     AH,0A8H
190E  EB DD                     JMP     SHORT B10_3     ;THEN RETURN
```

```
                        ;PRINT CHAR TO SERIAL PORT
                        ;DX = RS232 CARD TO BE USED: AL HAS CHAR TO BE PRINTED
1910  50                B12:    PUSH    AX              ;SAVE AL
1911  B4 01                     MOV     AH,01           ;1 IS SEND A CHAR DOWN COMMO LINE
1913  CD 14                     INT     014H            ;SEND THE CHAR
1915  E8 1925 R                 CALL    FAKE            ;FAKE WILL MAP ERROR BITS FROM
                                                        ;RS232 TO CORRESPONDING ONES
                                                        ;FOR THE PRINTER
1918  58                        POP     AX              ;RESTORE AL
1919  0A F6                     OR      DH,DH           ;SEE IF NO ERRORS WERE RETURNED
1918  74 04                     JZ      B12_1
191D  8A E6                     MOV     AH,DH           ;IF THERE WERE ERRORS, RETURN THEM
191F  EB CC                     JMP     SHORT B10_3     ;AND RETURN
1921  B4 10              B12_1: MOV     AH,010H         ;PUT 'CORRECT' RETURN STATUS IN AH
1923  EB C8                     JMP     SHORT B10_3     ;AND RETURN
1925                    REPRINT ENDP
                        ;THIS PROC MAPS THE ERRORS RETURNED FROM A BIOS INT14 CALL
                        ;TO THOSE 'LIKE THAT' OF AN INT17 CALL
                        ;BREAK,FRAMING,PARITY,OVERRUN ERRORS ARE LOGGED AS I/O
                        ;ERRORS AND A TIME OUT IS MOVED TO THE APPROPIATE BIT
1925                    FAKE    PROC NEAR
1925  32 F6                     XOR     DH,DH           ;CLEAR FAKED STATUS FLAGS
1927  F6 C4 1E                  TEST    AH,011110B      ;CHECK FOR BREAK,FRAMING,PARITY
                                                        ;OVERRUN
192A  74 03                     JZ      B13_1           ;ERRORS. IF NOT THEN CHECK FOR
                                                        ; TIME OUT.
192C  B6 08                     MOV     DH,01000B       ;SET BIT 3 TO INDICATE 'I/O ERROR'
192E  C3                        RET                     ;AND RETURN
192F  F6 C4 80            B13_1: TEST    AH,080H         ;TEST FOR TIME OUT ERROR RETURNED
1932  74 02                     JZ      B13_2           ;IF NOT TIME OUT, RETURN
1934  B6 09                     MOV     DH,09H          ;IF TIME OUT
1936  C3                  B13_2: RET
1937                    FAKE    ENDP
                        ;--------------------------------------------------------------------
                        ;NEW_INT9
                        ;       THIS ROUTINE IS THE INTERRUPT 9 HANDLER WHEN THE MACHINE IS
                        ;       FIRST POWERED ON AND CASSETTE BASIC IS GIVEN CONTROL.  IT
                        ;       HANDLES THE FIRST KEYSTROKES ENTERED FROM THE KEYBOARD AND
                        ;       PERFORMS "SPECIAL" ACTIONS AS FOLLOWS:
                        ;               IF ESC IS THE FIRST KEY ENETERED MINI-WELCOME IS
                        ;                  EXECUTED
                        ;               IF CTRL-ESC IS THE FIRST SEQUENCE "LOAD CAS1:,R" IS
                        ;                  EXECUTED GIVING THE USER THE ABILITY TO BOOT
                        ;                  FROM CASSETTE
                        ;       AFTER THESE KEYSTROKES OR AFTER ANY OTHER KEYSTROKES THE
                        ;       INTERRUPT 9 VECTOR IS CHANGED TO POINT AT THE REAL
                        ;       INTERRUPT 9 ROUTINE.
                        ;--------------------------------------------------------------------
1937                    NEW_INT_9  PROC FAR
1937  3C 01                     CMP     AL,1            ; IS THIS AN ESCAPE KEY?
1939  74 10                     JE      ESC_KEY         ; JUMP IF AL=ESCAPE KEY
193B  3C 1D                     CMP     AL,29           ; ELSE, IS THIS A CONTROL KEY?
193D  74 06                     JE      CTRL_KEY        ; JUMP IF AL=CONTROL KEY
193F  E8 E01B R                 CALL    REAL_VECTOR_SETUP ; OTHERWISE, INITIALIZE REAL
                                                        ; INT 9 VECTOR
1942  CD 09                     INT     9H              ; PASS THE SCAN CODE IN AL
1944  CF                        IRET                    ; RETURN TO INTERRUPT 48H
1945                    CTRL_KEY:
1945  80 0E 0017 R 04           OR      KB_FLAG,04H     ; TURN ON CTRL SHIFT IN KB_FLAG
194A  CF                        IRET                    ; RETURN TO INTERRUPT
194B                    ESC_KEY:
194B  F6 06 0017 R 04           TEST    KB_FLAG,04H     ; HAS CONTROL SHIFT OCCURED?
1950  74 29                     JE      ESC_ONLY        ; NO.  ESCAPE ONLY
                        ;CONTROL ESCAPE HAS OCCURED, PUT MESSAGE IN BUFFER FOR CASSETTE
                        ; LOAD
1952  C6 06 0017 R 00           MOV     KB_FLAG,0       ; ZERO OUT CONTROL STATE
1957  1E                        PUSH    DS
1958  07                        POP     ES              ; INITIALIZE ES FOR BIOS DATA
1959  1E                        PUSH    DS              ; SAVE OLD DS
195A  0E                        PUSH    CS              ; POINT DS AT CODE SEGMENT
195B  1F                        POP     DS
195C  BE 1983 R                 MOV     SI,OFFSET CAS_LOAD ; GET MESSAGE
195F  BF 001E R                 MOV     DI,OFFSET KB_BUFFER ; POINT AT KEYBOARD BUFFER
1962  B9 000F 90                MOV     CX,CAS_LENGTH   ; LENGTH OF CASSETTE MESSAGE
1966  AC                  T_LOOP: LODSB                   ; GET ASCII CHARACTER FROM MESSAGE
1967  AB                        STOSW                   ; PUT IN KEYBOARD BUFFER
1968  E2 FC                     LOOP    T_LOOP
196A  1F                        POP     DS              ; RETRIEVE BIOS DATA SEGMENT
                        ;------ INITIALIZE QUEUE SO MESSAGE WILL BE REMOVED FROM BUFFER
196B  C7 06 001A R 001E R       MOV     BUFFER_HEAD,OFFSET KB_BUFFER
1971  C7 06 001C R 003C R       MOV     BUFFER_TAIL,OFFSET KB_BUFFER+(CAS_LENGTH*2)
                        ;--------------------------------------------------------------------
                        ;***NOTE***
                        ;       IT IS ASSUMED THAT THE LENGTH OF THE CASSETTE MESSAGE IS
                        ;       LESS THAN OR EQUAL TO THE LENGTH OF THE BUFFER.  IF THIS IS
                        ;       NOT THE CASE THE BUFFER WILL EVENTUALLY CONSUME MEMORY.
                        ;--------------------------------------------------------------------
1977  E8 E01B R                 CALL    REAL_VECTOR_SETUP
197A  CF                        IRET
197B                    ESC_ONLY:
197B  E8 E01B R                 CALL    REAL_VECTOR_SETUP
197E  B9 2000                   MOV     CX,MINI
1981  FF E1                     JMP     CX              ; ENTER THE WORLD OF KEYBOARD CAPER
                        ;------ MESSAGE FOR OUTPUT WHEN CONTROL-ESCAPE IS ENTERED AS FIRST
                        ;       KEY SEQUENCE
1983                    CAS_LOAD        LABEL   BYTE
1983  4C 4F 41 44 20 22         DB 'LOAD "CAS1:",R'
      43 41 53 31 3A 22
      2C 52
1991  0D                        DB  13
= 000F                  CAS_LENGTH EQU $ - CAS_LOAD
1992                    NEW_INT_9  ENDP
```

```
                        ;--------------------------------------------------
                        ; WRITE_TTY
                        ; THIS INTERFACE PROVIDES A TELETYPE LIKE INTERFACE TO THE
                        ;   VIDEO CARD.  THE INPUT CHARACTER IS WRITTEN TO THE CURRENT
                        ;   CURSOR POSITION, AND THE CURSOR IS MOVED TO THE NEXT POSITION.
                        ;   IF THE CURSOR LEAVES THE LAST COLUMN OF THE FIELD, THE COLUMN
                        ;   IS SET TO ZERO, AND THE ROW VALUE IS INCREMENTED.  IF THE ROW
                        ;   ROW VALUE LEAVES THE FIELD, THE CURSOR IS PLACED ON THE LAST
                        ;   ROW, FIRST COLUMN, AND THE ENTIRE SCREEN IS SCROLLED UP ONE
                        ;   LINE.  WHEN THE SCREEN IS SCROLLED UP, THE ATTRIBUTE FOR FILLING
                        ;   THE NEWLY BLANKED LINE IS READ FROM THE CURSOR POSITION ON THE
                        ;   PREVIOUS LINE BEFORE THE SCROLL, IN CHARACTER MODE.  IN
                        ;   GRAPHICS MODE, 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
1992                    WRITE_TTY       PROC    NEAR
1992  50                        PUSH    AX              ; SAVE REGISTERS
1993  50                        PUSH    AX              ; SAVE CHAR TO WRITE
1994  8A 3E 0062 R              MOV     BH,ACTIVE_PAGE  ; GET CURRENT PAGE SETTING
1998  53                        PUSH    BX              ; SAVE IT
1999  8A DF                     MOV     BL,BH           ; IN BL
199B  32 FF                     XOR     BH,BH
199D  D1 E3                     SAL     BX,1            ; CONVERT TO WORD OFFSET
199F  8B 97 0050 R              MOV     DX,[BX+OFFSET CURSOR_POSN] ; GET CURSOR POSITION
19A3  5B                        POP     BX              ; RECOVER CURRENT PAGE
19A4  58                        POP     AX              ; RECOVER CHAR
                        ;----- DX NOW HAS THE CURRENT CURSOR POSITION
19A5  3C 08                     CMP     AL,8            ; IS IT A BACKSPACE?
19A7  74 50                     JE      U8              ; BACK_SPACE
19A9  3C 0D                     CMP     AL,0DH          ; IS IT A CARRIAGE RETURN?
19AB  74 54                     JE      U9              ; CAR_RET
19AD  3C 0A                     CMP     AL,0AH          ; IS IT A LINE FEED
19AF  74 15                     JE      U10             ; LINE_FEED
19B1  3C 07                     CMP     AL,07H          ; IS IT A BELL
19B3  74 50                     JE      U11             ; BELL
                        ;----- WRITE THE CHAR TO THE SCREEN
19B5  B4 0A                     MOV     AH,10           ; WRITE CHAR ONLY
19B7  B9 0001                   MOV     CX,1            ; ONLY ONE CHAR
19BA  CD 10                     INT     10H             ; WRITE THE CHAR
                        ;----- POSITION THE CURSOR FOR NEXT CHAR
19BC  FE C2                     INC     DL
19BE  3A 16 004A R              CMP     DL,BYTE PTR CRT_COLS ; TEST FOR COLUMN OVERFLOW
19C2  75 31                     JNZ     U7              ; SET_CURSOR
19C4  32 D2                     XOR     DL,DL           ; COLUMN FOR CURSOR
                        ;---- LINE FEED
19C6                    U10:
19C6  80 FE 18                  CMP     DH,24
19C9  75 28                     JNZ     U6              ; SET_CURSOR_INC
                        ;----- SCROLL REQUIRED
19CB  B4 02                     MOV     AH,2
19CD  CD 10                     INT     10H             ; SET THE CURSOR
                        ;----- DETERMINE VALUE TO FILL WITH DURING SCROLL
19CF  A0 0049 R                 MOV     AL,CRT_MODE     ; GET THE CURRENT MODE
19D2  3C 04                     CMP     AL,4
19D4  72 04                     JC      U2              ; READ-CURSOR
19D6  32 FF                     XOR     BH,BH           ; FILL WITH BACKGROUND
19D8  EB 06                     JMP     SHORT U3        ; SCROLL-UP
19DA  B4 08             U2:     MOV     AH,8
19DC  CD 10                     INT     10H             ; READ CHAR/ATTR AT CURRENT CURSOR
19DE  8A FC                     MOV     BH,AH           ; STORE IN BH
19E0  B8 0601           U3:     MOV     AX,601H         ; SCROLL ONE LINE
19E3  2B C9                     SUB     CX,CX           ; UPPER LEFT CORNER
19E5  B6 18                     MOV     DH,24           ; LOWER RIGHT ROW
19E7  8A 16 004A R              MOV     DL,BYTE PTR CRT_COLS ; LOWER RIGHT COLUMN
19EB  FE CA                     DEC     DL
19ED  CD 10             U4:     INT     10H             ; SCROLL UP THE SCREEN
19EF  58                U5:     POP     AX              ; RESTORE THE CHARACTER
19F0  E9 0F70 R                 JMP     VIDEO_RETURN    ; RETURN TO CALLER
19F3  FE C6             U6:     INC     DH              ; NEXT ROW
19F5  B4 02             U7:     MOV     AH,2
19F7  EB F4                     JMP     U4              ; ESTABLISH THE NEW CURSOR
                        ;----- BACK SPACE FOUND
19F9  0A D2             U8:     OR      DL,DL           ; ALREADY AT END OF LINE
19FB  74 F8                     JE      U7              ; SET_CURSOR
19FD  FE CA                     DEC     DL              ; NO -- JUST MOVE IT BACK
19FF  EB F4                     JMP     U7              ; SET_CURSOR
                        ;----- CARRIAGE RETURN FOUND
1A01  32 D2             U9:     XOR     DL,DL           ; MOVE TO FIRST COLUMN
1A03  EB F0                     JMP     U7              ; SET_CURSOR
                        ;----- BELL FOUND
1A05  B3 02             U11:    MOV     BL,2            ; SET UP COUNT FOR BEEP
1A07  E8 FF31 R                 CALL    BEEP            ; SOUND THE POD BELL
1A0A  EB E3                     JMP     U5              ; TTY_RETURN
1A0C                    WRITE_TTY       ENDP
```

```
                        ;--------------------------------------------------------------------------
                        ;       THIS PROCEDURE WILL ISSUE SHORT TONES TO INDICATE FAILURES
                        ;       THAT  1: OCCUR BEFORE THE CRT IS STARTED, 2: TO CALL THE
                        ;       OPERATORS ATTENTION TO AN ERROR AT THE END OF POST, OR
                        ;       3: TO SIGNAL THE SUCCESSFUL COMPLETION OF POST
                        ; ENTRY PARAMETERS:
                        ;       DL = NUMBER OF APPROX. 1/2 SEC TONES TO SOUND
                        ;--------------------------------------------------------------------------
1A0C                    ERR_BEEP  PROC  NEAR
1A0C  9C                        PUSHF                   ; SAVE FLAGS
1A0D  53                        PUSH    BX
1A0E  FA                        CLI                     ; DISABLE SYSTEM INTERRUPTS
1A0F                    G3:                             ; SHORT_BEEP:
1A0F  B3 01                     MOV     BL,1            ; COUNTER FOR A SHORT BEEP
1A11  E8 FF31 R                 CALL    BEEP            ; DO THE SOUND
1A14  E2 FE             G4:     LOOP    G4              ; DELAY BETWEEN BEEPS
1A16  FE CA                     DEC     DL              ; DONE WITH SHORTS
1A18  75 F5                     JNZ     G3              ; DO SOME MORE
1A1A  E2 FE             G5:     LOOP    G5              ; LONG DELAY BEFORE RETURN
1A1C  E2 FE             G6:     LOOP    G6
1A1E  5B                        POP     BX              ; RESTORE ORIG CONTENTS OF BX
1A1F  9D                        POPF                    ; RESTORE FLAGS TO ORIG SETTINGS
1A20  C3                        RET                     ; RETURN TO CALLER
1A21                    ERR_BEEP        ENDP
                         LIST
                                ASSUME  CS:CODE,DS:DATA
E000                            ORG     0E000H
E000  31 35 30 34 30 33         DB      '1504037 COPR. IBM 1981,1983'  ; COPYRIGHT NOTICE
      37 20 43 4F 50 52
      2E 20 49 42 4D 20
      31 39 38 31 2C 31
      39 38 33

                        ;--------------------------------------------------------------------------
                        ;REAL_VECTOR_SETUP
                        ;
                        ;       THIS ROUTINE WILL INITIALIZE THE INTERRUPT 9 VECTOR TO
                        ;       POINT AT THE REAL INTERRUPT ROUTINE.
                        ;--------------------------------------------------------------------------
E01B                    REAL_VECTOR_SETUP       PROC       NEAR
E01B  50                        PUSH    AX              ; SAVE THE SCAN CODE
E01C  53                        PUSH    BX
E01D  06                        PUSH    ES
E01E  33 C0                     XOR     AX,AX           ; INITIALIZE TO POINT AT VECTOR
                                                        ; SECTOR(0)
E020  8E C0                     MOV     ES,AX
E022  BB 0024                   MOV     BX,9H*4H        ; POINT AT INTERRUPT 9
E025  26: C7 07 1561 R          MOV     WORD PTR ES:[BX],OFFSET KB_INT ; MOVE IN OFFSET OF
                                                        ; ROUTINE
E02A  43                        INC     BX              ; ADD 2 TO BX
E02B  43                        INC     BX
E02C  0E                        PUSH    CS              ; GET CODE SEGMENT OF BIOS (SEGMENT
                                                        ; RELOCATEABLE)
E02D  58                        POP     AX
E02E  26: 89 07                 MOV     WORD PTR ES:[BX],AX ; MOVE IN SEGMENT OF ROUTINE
E031  07                        POP     ES
E032  5B                        POP     BX
E033  58                        POP     AX
E034  C3                        RET
E035                    REAL_VECTOR_SETUP       ENDP
                        ;--------------------------------------------------------------------------
                        ;KB_NOISE
                        ;       THIS ROUTINE IS CALLED WHEN GENERAL BEEPS ARE REQUIRED FROM
                        ;       THE SYSTEM.
                        ;INPUT
                        ;       BX=LENGH OF THE TONE
                        ;       CX=CONTAINS THE FREQUENCY
                        ;OUTPUT
                        ;       ALL REGISTERS ARE MAINTAINED.
                        ;HINTS
                        ;       AS CX GETS LARGER THE TONE PRODUCED GETS LOWER IN PITCH.
                        ;
                        ;--------------------------------------------------------------------------
E035                    KB_NOISE        PROC    NEAR
E035  FB                        STI
E036  50                        PUSH    AX
E037  53                        PUSH    BX
E038  51                        PUSH    CX
E039  E4 61                     IN      AL,061H         ; GET CONTROL INFO
E03B  50                        PUSH    AX              ; SAVE
E03C                    LOOPO1:
E03C  24 FC                     AND     AL,0FCH         ; TURN OFF TIMER GATE AND SPEAKER
                                                        ; DATA
E03E  E6 61                     OUT     061H,AL         ; OUTPUT TO CONTROL
E040  51                        PUSH    CX              ; HALF CYCLE TIME FOR TONE
E041  E2 FE             LOOPO2: LOOP    LOOPO2          ; SPEAKER OFF
E043  0C 02                     OR      AL,2            ; TURN ON SPEAKER BIT
E045  E6 61                     OUT     061H,AL         ; OUTPUT TO CONTROL
E047  59                        POP     CX
E048  51                        PUSH    CX              ; RETRIEVE FREQUENCY
E049  E2 FE             LOOPO3: LOOP    LOOPO3          ; ANOTHER HALF CYCLE
E04B  4B                        DEC     BX              ; TOTAL TIME COUNT
E04C  59                        POP     CX              ; RETRIEVE  FREQ.
E04D  75 ED                     JNZ     LOOPO1          ; DO ANOTHER CYCLE
E04F  58                        POP     AX              ; RECOVER CONTROL
E050  E6 61                     OUT     061H,AL         ; OUTPUT THE CONTROL
E052  59                        POP     CX
E053  5B                        POP     BX
E054  58                        POP     AX
E055  C3                        RET
E056                    KB_NOISE        ENDP
E05B                            ORG     0E05BH
E05B  E9 0043 R                 JMP     NEAR PTR RESET
```

```
;--------------------------------------------------------------------------
;       CHARACTER GENERATOR GRAPHICS FOR 320X200 AND 640X200
;       GRAPHICS FOR CHARACTERS 80H THROUGH FFH
;--------------------------------------------------------------------------

E05E                            CRT_CHARH       LABEL   BYTE
E05E  78 CC C0 CC 78 18                 DB      078H,0CCH,0C0H,0CCH,078H,018H,00CH,078H ; D_80
      0C 78
E066  00 CC 00 CC CC CC                 DB      000H,0CCH,000H,0CCH,0CCH,0CCH,07EH,000H ; D_81
      7E 00
E06E  1C 00 78 CC FC C0                 DB      01CH,000H,078H,0CCH,0FCH,0C0H,078H,000H ; D_82
      78 00
E076  7E C3 3C 06 3E 66                 DB      07EH,0C3H,03CH,006H,03EH,066H,03FH,000H ; D_83
      3F 00
E07E  CC 00 78 0C 7C CC                 DB      0CCH,000H,078H,00CH,07CH,0CCH,07EH,000H ; D_84
      7E 00
E086  E0 00 78 0C 7C CC                 DB      0E0H,000H,078H,00CH,07CH,0CCH,07EH,000H ; D_85
      7E 00
E08E  30 30 78 0C 7C CC                 DB      030H,030H,078H,00CH,07CH,0CCH,07EH,000H ; D_86
      7E 00
E096  00 00 78 C0 C0 78                 DB      000H,000H,078H,0C0H,0C0H,078H,00CH,038H ; D_87
      0C 38
E09E  7E C3 3C 66 7E 60                 DB      07EH,0C3H,03CH,066H,07EH,060H,03CH,000H ; D_88
      3C 00
E0A6  CC 00 78 CC FC C0                 DB      0CCH,000H,078H,0CCH,0FCH,0C0H,078H,000H ; D_89
      78 00
E0AE  E0 00 78 CC FC C0                 DB      0E0H,000H,078H,0CCH,0FCH,0C0H,078H,000H ; D_8A
      78 00
E0B6  CC 00 70 30 30 30                 DB      0CCH,000H,070H,030H,030H,030H,078H,000H ; D_8B
      78 00
E0BE  7C C6 38 18 18 18                 DB      07CH,0C6H,038H,018H,018H,018H,03CH,000H ; D_8C
      3C 00
E0C6  E0 00 70 30 30 30                 DB      0E0H,000H,070H,030H,030H,030H,078H,000H ; D_8D
      78 00
E0CE  C6 38 6C C6 FE C6                 DB      0C6H,038H,06CH,0C6H,0FEH,0C6H,0C6H,000H ; D_8E
      C6 00
E0D6  30 30 00 78 CC FC                 DB      030H,030H,000H,078H,0CCH,0FCH,0CCH,000H ; D_8F
      CC 00

E0DE  1C 00 FC 60 78 60                 DB      01CH,000H,0FCH,060H,078H,060H,0FCH,000H ; D_90
      FC 00
E0E6  00 00 7F 0C 7F CC                 DB      000H,000H,07FH,00CH,07FH,0CCH,07FH,000H ; D_91
      7F 00
E0EE  3E 6C CC FE CC CC                 DB      03EH,06CH,0CCH,0FEH,0CCH,0CCH,0CEH,000H ; D_92
      CE 00
E0F6  78 CC 00 78 CC CC                 DB      078H,0CCH,000H,078H,0CCH,0CCH,078H,000H ; D_93
      78 00
E0FE  00 CC 00 78 CC CC                 DB      000H,0CCH,000H,078H,0CCH,0CCH,078H,000H ; D_94
      78 00
E106  00 E0 00 78 CC CC                 DB      000H,0E0H,000H,078H,0CCH,0CCH,078H,000H ; D_95
      78 00
E10E  78 CC 00 CC CC CC                 DB      078H,0CCH,000H,0CCH,0CCH,0CCH,07EH,000H ; D_96
      7E 00
E116  00 E0 00 CC CC CC                 DB      000H,0E0H,000H,0CCH,0CCH,0CCH,07EH,000H ; D_97
      7E 00
E11E  00 CC 00 CC CC 7C                 DB      000H,0CCH,000H,0CCH,0CCH,07CH,00CH,0F8H ; D_98
      0C F8
E126  C3 18 3C 66 66 3C                 DB      0C3H,018H,03CH,066H,066H,03CH,018H,000H ; D_99
      18 00
E12E  CC 00 CC CC CC CC                 DB      0CCH,000H,0CCH,0CCH,0CCH,0CCH,078H,000H ; D_9A
      78 00
E136  18 18 7E C0 C0 7E                 DB      018H,018H,07EH,0C0H,0C0H,07EH,018H,018H ; D_9B
      18 18
E13E  38 6C 64 F0 60 E6                 DB      038H,06CH,064H,0F0H,060H,0E6H,0FCH,000H ; D_9C
      FC 00
E146  CC CC 78 FC 30 FC                 DB      0CCH,0CCH,078H,0FCH,030H,0FCH,030H,030H ; D_9D
      30 30
E14E  F8 CC CC FA C6 CF                 DB      0F8H,0CCH,0CCH,0FAH,0C6H,0CFH,0C6H,0C7H ; D_9E
      C6 C7
E156  0E 1B 18 3C 18 18                 DB      00EH,01BH,018H,03CH,018H,018H,0D8H,070H ; D_9F
      D8 70

E15E  1C 00 78 0C 7C CC                 DB      01CH,000H,078H,00CH,07CH,0CCH,07EH,000H ; D_A0
      7E 00
E166  38 00 70 30 30 30                 DB      038H,000H,070H,030H,030H,030H,078H,000H ; D_A1
      78 00
E16E  00 1C 00 78 CC CC                 DB      000H,01CH,000H,078H,0CCH,0CCH,078H,000H ; D_A2
      78 00
E176  00 1C 00 CC CC CC                 DB      000H,01CH,000H,0CCH,0CCH,0CCH,07EH,000H ; D_A3
      7E 00
E17E  00 F8 00 F8 CC CC                 DB      000H,0F8H,000H,0F8H,0CCH,0CCH,0CCH,000H ; D_A4
      CC 00
E186  FC 00 CC EC FC DC                 DB      0FCH,000H,0CCH,0ECH,0FCH,0DCH,0CCH,000H ; D_A5
      CC 00
E18E  3C 6C 6C 3E 00 7E                 DB      03CH,06CH,06CH,03EH,000H,07EH,000H,000H ; D_A6
      00 00
E196  38 6C 6C 38 00 7C                 DB      038H,06CH,06CH,038H,000H,07CH,000H,000H ; D_A7
      00 00
E19E  30 00 30 60 C0 CC                 DB      030H,000H,030H,060H,0C0H,0CCH,078H,000H ; D_A8
      78 00
E1A6  00 00 00 FC C0 C0                 DB      000H,000H,000H,0FCH,0C0H,0C0H,000H,000H ; D_A9
      00 00
E1AE  00 00 00 FC 0C 0C                 DB      000H,000H,000H,0FCH,00CH,00CH,000H,000H ; D_AA
      00 00
E1B6  C3 C6 CC DE 33 66                 DB      0C3H,0C6H,0CCH,0DEH,033H,066H,0CCH,00FH ; D_AB
      CC 0F
E1BE  C3 C6 CC DB 37 6F                 DB      0C3H,0C6H,0CCH,0DBH,037H,06FH,0CFH,003H ; D_AC
      CF 03
E1C6  18 18 00 18 18 18                 DB      018H,018H,000H,018H,018H,018H,018H,000H ; D_AD
      18 00
E1CE  00 33 66 CC 66 33                 DB      000H,033H,066H,0CCH,066H,033H,000H,000H ; D_AE
      00 00
E1D6  00 CC 66 33 66 CC                 DB      000H,0CCH,066H,033H,066H,0CCH,000H,000H ; D_AF
      00 00
```

```
E1DE  22 88 22 88 22 88 22 88          DB      022H,088H,022H,088H,022H,088H,022H,088H  ; D_B0
E1E6  55 AA 55 AA 55 AA 55 AA          DB      055H,0AAH,055H,0AAH,055H,0AAH,055H,0AAH  ; D_B1
E1EE  DB 77 DB EE DB 77 DB EE          DB      0DBH,077H,0DBH,0EEH,0DBH,077H,0DBH,0EEH  ; D_B2
E1F6  18 18 18 18 18 18 18 18          DB      018H,018H,018H,018H,018H,018H,018H,018H  ; D_B3
E1FE  18 18 18 18 F8 18 18 18          DB      018H,018H,018H,018H,0F8H,018H,018H,018H  ; D_B4
E206  18 18 F8 18 F8 18 18 18          DB      018H,018H,0F8H,018H,0F8H,018H,018H,018H  ; D_B5
E20E  36 36 36 36 F6 36 36 36          DB      036H,036H,036H,036H,0F6H,036H,036H,036H  ; D_B6
E216  00 00 00 00 FE 36 36 36          DB      000H,000H,000H,000H,0FEH,036H,036H,036H  ; D_B7
E21E  00 00 F8 18 F8 18 18 18          DB      000H,000H,0F8H,018H,0F8H,018H,018H,018H  ; D_B8
E226  36 36 F6 06 F6 36 36 36          DB      036H,036H,0F6H,006H,0F6H,036H,036H,036H  ; D_B9
E22E  36 36 36 36 36 36 36 36          DB      036H,036H,036H,036H,036H,036H,036H,036H  ; D_BA
E236  00 00 FE 06 F6 36 36 36          DB      000H,000H,0FEH,006H,0F6H,036H,036H,036H  ; D_BB
E23E  36 36 F6 06 FE 00 00 00          DB      036H,036H,0F6H,006H,0FEH,000H,000H,000H  ; D_BC
E246  36 36 36 36 FE 00 00 00          DB      036H,036H,036H,036H,0FEH,000H,000H,000H  ; D_BD
E24E  18 18 F8 18 F8 00 00 00          DB      018H,018H,0F8H,018H,0F8H,000H,000H,000H  ; D_BE
E256  00 00 00 00 F8 18 18 18          DB      000H,000H,000H,000H,0F8H,018H,018H,018H  ; D_BF

E25E  18 18 18 18 1F 00 00 00          DB      018H,018H,018H,018H,01FH,000H,000H,000H  ; D_C0
E266  18 18 18 18 FF 00 00 00          DB      018H,018H,018H,018H,0FFH,000H,000H,000H  ; D_C1
E26E  00 00 00 00 FF 18 18 18          DB      000H,000H,000H,000H,0FFH,018H,018H,018H  ; D_C2
E276  18 18 18 18 1F 18 18 18          DB      018H,018H,018H,018H,01FH,018H,018H,018H  ; D_C3
E27E  00 00 00 00 FF 00 00 00          DB      000H,000H,000H,000H,0FFH,000H,000H,000H  ; D_C4
E286  18 18 18 18 FF 18 18 18          DB      018H,018H,018H,018H,0FFH,018H,018H,018H  ; D_C5
E28E  18 18 1F 18 1F 18 18 18          DB      018H,018H,01FH,018H,01FH,018H,018H,018H  ; D_C6
E296  36 36 36 36 37 36 36 36          DB      036H,036H,036H,036H,037H,036H,036H,036H  ; D_C7
E29E  36 36 37 30 3F 00 00 00          DB      036H,036H,037H,030H,03FH,000H,000H,000H  ; D_C8
E2A6  00 00 3F 30 37 36 36 36          DB      000H,000H,03FH,030H,037H,036H,036H,036H  ; D_C9
E2AE  36 36 F7 00 FF 00 00 00          DB      036H,036H,0F7H,000H,0FFH,000H,000H,000H  ; D_CA
E2B6  00 00 FF 00 F7 36 36 36          DB      000H,000H,0FFH,000H,0F7H,036H,036H,036H  ; D_CB
E2BE  36 36 37 30 37 36 36 36          DB      036H,036H,037H,030H,037H,036H,036H,036H  ; D_CC
E2C6  00 00 FF 00 FF 00 00 00          DB      000H,000H,0FFH,000H,0FFH,000H,000H,000H  ; D_CD
E2CE  36 36 F7 00 F7 36 36 36          DB      036H,036H,0F7H,000H,0F7H,036H,036H,036H  ; D_CE
E2D6  18 18 FF 00 FF 00 00 00          DB      018H,018H,0FFH,000H,0FFH,000H,000H,000H  ; D_CF

E2DE  36 36 36 36 FF 00 00 00          DB      036H,036H,036H,036H,0FFH,000H,000H,000H  ; D_D0
E2E6  00 00 FF 00 FF 18 18 18          DB      000H,000H,0FFH,000H,0FFH,018H,018H,018H  ; D_D1
E2EE  00 00 00 00 FF 36 36 36          DB      000H,000H,000H,000H,0FFH,036H,036H,036H  ; D_D2
E2F6  36 36 36 36 3F 00 00 00          DB      036H,036H,036H,036H,03FH,000H,000H,000H  ; D_D3
E2FE  18 18 1F 18 1F 00 00 00          DB      018H,018H,01FH,018H,01FH,000H,000H,000H  ; D_D4
E306  00 00 1F 18 1F 18 18 18          DB      000H,000H,01FH,018H,01FH,018H,018H,018H  ; D_D5
E30E  00 00 00 00 3F 36 36 36          DB      000H,000H,000H,000H,03FH,036H,036H,036H  ; D_D6
E316  36 36 36 36 FF 36 36 36          DB      036H,036H,036H,036H,0FFH,036H,036H,036H  ; D_D7
E31E  18 18 FF 18 FF 18 18 18          DB      018H,018H,0FFH,018H,0FFH,018H,018H,018H  ; D_D8
E326  18 18 18 18 F8 00 00 00          DB      018H,018H,018H,018H,0F8H,000H,000H,000H  ; D_D9
E32E  00 00 00 00 1F 18 18 18          DB      000H,000H,000H,000H,01FH,018H,018H,018H  ; D_DA
E336  FF FF FF FF FF FF FF FF          DB      0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH,0FFH  ; D_DB
E33E  00 00 00 00 FF FF FF FF          DB      000H,000H,000H,000H,0FFH,0FFH,0FFH,0FFH  ; D_DC
E346  F0 F0 F0 F0 F0 F0 F0 F0          DB      0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H,0F0H  ; D_DD
E34E  0F 0F 0F 0F 0F 0F 0F 0F          DB      00FH,00FH,00FH,00FH,00FH,00FH,00FH,00FH  ; D_DE
E356  FF FF FF FF 00 00 00 00          DB      0FFH,0FFH,0FFH,0FFH,000H,000H,000H,000H  ; D_DF
```

```
E35E  00 00 76 DC C8 DC              DB      000H,000H,076H,0DCH,0C8H,0DCH,076H,000H  ; D_E0
      76 00
E366  00 78 CC F8 CC F8              DB      000H,078H,0CCH,0F8H,0CCH,0F8H,0C0H,0C0H  ; D_E1
      C0 C0
E36E  00 FC CC C0 C0 C0              DB      000H,0FCH,0CCH,0C0H,0C0H,0C0H,0C0H,000H  ; D_E2
      C0 00
E376  00 FE 6C 6C 6C 6C              DB      000H,0FEH,06CH,06CH,06CH,06CH,06CH,000H  ; D_E3
      6C 00
E37E  FC CC 60 30 60 CC              DB      0FCH,0CCH,060H,030H,060H,0CCH,0FCH,000H  ; D_E4
      FC 00
E386  00 00 7E D8 D8 D8              DB      000H,000H,07EH,0D8H,0D8H,0D8H,070H,000H  ; D_E5
      70 00
E38E  00 66 66 66 66 7C              DB      000H,066H,066H,066H,066H,07CH,060H,0C0H  ; D_E6
      60 C0
E396  00 76 DC 18 18 18              DB      000H,076H,0DCH,018H,018H,018H,018H,000H  ; D_E7
      18 00
E39E  FC 30 78 CC CC 78              DB      0FCH,030H,078H,0CCH,0CCH,078H,030H,0FCH  ; D_E8
      30 FC
E3A6  38 6C C6 FE C6 6C              DB      038H,06CH,0C6H,0FEH,0C6H,06CH,038H,000H  ; D_E9
      38 00
E3AE  38 6C C6 C6 6C 6C              DB      038H,06CH,0C6H,0C6H,06CH,06CH,0EEH,000H  ; D_EA
      EE 00
E3B6  1C 30 18 7C CC CC              DB      01CH,030H,018H,07CH,0CCH,0CCH,078H,000H  ; D_EB
      78 00
E3BE  00 00 7E DB DB 7E              DB      000H,000H,07EH,0DBH,0DBH,07EH,000H,000H  ; D_EC
      00 00
E3C6  06 0C 7E DB DB 7E              DB      006H,00CH,07EH,0DBH,0DBH,07EH,060H,0C0H  ; D_ED
      60 C0
E3CE  38 60 C0 F8 C0 60              DB      038H,060H,0C0H,0F8H,0C0H,060H,038H,000H  ; D_EE
      38 00
E3D6  78 CC CC CC CC CC              DB      078H,0CCH,0CCH,0CCH,0CCH,0CCH,0CCH,000H  ; D_EF
      CC 00

E3DE  00 FC 00 FC 00 FC              DB      000H,0FCH,000H,0FCH,000H,0FCH,000H,000H  ; D_F0
      00 00
E3E6  30 30 FC 30 30 00              DB      030H,030H,0FCH,030H,030H,000H,0FCH,000H  ; D_F1
      FC 00
E3EE  60 30 18 30 60 00              DB      060H,030H,018H,030H,060H,000H,0FCH,000H  ; D_F2
      FC 00
E3F6  18 30 60 30 18 00              DB      018H,030H,060H,030H,018H,000H,0FCH,000H  ; D_F3
      FC 00
E3FE  0E 1B 1B 18 18 18              DB      00EH,01BH,01BH,018H,018H,018H,018H,018H  ; D_F4
      18 18
E406  18 18 18 18 18 D8              DB      018H,018H,018H,018H,018H,0D8H,0D8H,070H  ; D_F5
      D8 70
E40E  30 30 00 FC 00 30              DB      030H,030H,000H,0FCH,000H,030H,030H,000H  ; D_F6
      30 00
E416  00 76 DC 00 76 DC              DB      000H,076H,0DCH,000H,076H,0DCH,000H,000H  ; D_F7
      00 00
E41E  38 6C 6C 38 00 00              DB      038H,06CH,06CH,038H,000H,000H,000H,000H  ; D_F8
      00 00
E426  00 00 00 18 18 00              DB      000H,000H,000H,018H,018H,000H,000H,000H  ; D_F9
      00 00
E42E  00 00 00 00 18 00              DB      000H,000H,000H,000H,018H,000H,000H,000H  ; D_FA
      00 00
E436  0F 0C 0C 0C EC 6C              DB      00FH,00CH,00CH,00CH,0ECH,06CH,03CH,01CH  ; D_FB
      3C 1C
E43E  78 6C 6C 6C 6C 00              DB      078H,06CH,06CH,06CH,06CH,000H,000H,000H  ; D_FC
      00 00
E446  70 18 30 60 78 00              DB      070H,018H,030H,060H,078H,000H,000H,000H  ; D_FD
      00 00
E44E  00 00 3C 3C 3C 3C              DB      000H,000H,03CH,03CH,03CH,03CH,000H,000H  ; D_FE
      00 00
E456  00 00 00 00 00 00              DB      000H,000H,000H,000H,000H,000H,000H,000H  ; D_FF
      00 00

                                     ASSUME  CS:CODE,DS:DATA
                                ;--------------------------------------------------
                                ; SET_CTYPE
                                ;       THIS ROUTINE SETS THE CURSOR VALUE
                                ; INPUT
                                ;       (CX) HAS CURSOR VALUE CH-START LINE, CL-STOP LINE
                                ; OUTPUT
                                ;       NONE
                                ;--------------------------------------------------
E45E                            SET_CTYPE       PROC    NEAR
E45E  80 FC 04                          CMP     AH,4                    ; IN GRAPHICS MODE?
E461  72 03                             JC      C23X                    ; NO, JUMP
E463  80 CD 20                          OR      CH,20H                  ; YES, DISABLE CURSOR
E466  B4 0A                     C23X:   MOV     AH,10                   ; 6845 REGISTER FOR CURSOR SET
E468  89 0E 0060 R                      MOV     CURSOR_MODE,CX          ; SAVE IN DATA AREA
E46C  E8 E472 R                         CALL    C23                     ; OUTPUT CX REG
E46F  E9 0F70 R                         JMP     VIDEO_RETURN
                                ;THIS ROUTINE OUTPUTS THE CX REGISTER TO THE 6845 REGS NAMED IN AH
E472  8B 16 0063 R              C23:    MOV     DX,ADDR_6845            ; ADDRESS REGISTER
E476  8A C4                             MOV     AL,AH                   ; GET VALUE
E478  EE                                OUT     DX,AL                   ; REGISTER SET
E479  42                                INC     DX                      ; DATA REGISTER
E47A  8A C5                             MOV     AL,CH                   ; DATA
E47C  EE                                OUT     DX,AL
E47D  4A                                DEC     DX
E47E  8A C4                             MOV     AL,AH
E480  FE C0                             INC     AL                      ; POINT TO OTHER DATA REGISTER
E482  EE                                OUT     DX,AL                   ; SET FOR SECOND REGISTER
E483  42                                INC     DX
E484  8A C1                             MOV     AL,CL                   ; SECOND DATA VALUE
E486  EE                                OUT     DX,AL
E487  C3                                RET                             ; ALL DONE
E488                            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
                        ;--------------------------------------------
E488                    SET_CPOS        PROC    NEAR
E488  8A CF                     MOV     CL,BH
E48A  32 ED                     XOR     CH,CH                   ; ESTABLISH LOOP COUNT
E48C  D1 E1                     SAL     CX,1                    ; WORD OFFSET
E48E  8B F1                     MOV     SI,CX                   ; USE INDEX REGISTER
E490  89 94 0050 R              MOV     [SI+OFFSET CURSOR_POSN],DX ; SAVE THE POINTER
E494  38 3E 0062 R              CMP     ACTIVE_PAGE,BH
E498  75 05                     JNZ     C24                     ; SET_CPOS_RETURN
E49A  8B C2                     MOV     AX,DX                   ; GET ROW/COLUMN TO AX
E49C  E8 E4A2 R                 CALL    C25                     ; CURSOR_SET
E49F  E9 0F70 R         C24:    JMP     VIDEO_RETURN
E4A2                    SET_CPOS        ENDP
                        ;------ SET CURSOR POSITION, AX HAS ROW/COLUMN FOR CURSOR
E4A2                    C25     PROC    NEAR
E4A2  E8 E5C2 R                 CALL    POSITION                ; DETERMINE LOCATION IN REGEN
                                                                ; BUFFER
E4A5  8B C8                     MOV     CX,AX
E4A7  03 0E 004E R              ADD     CX,CRT_START            ; ADD IN THE START ADDRESS FOR THIS
                                                                ; PAGE
E4AB  D1 F9                     SAR     CX,1                    ; DIVIDE BY 2 FOR CHAR ONLY COUNT
E4AD  B4 0E                     MOV     AH,14                   ; REGISTER NUMBER FOR CURSOR
E4AF  E8 E472 R                 CALL    C23                     ; OUTPUT THE VALUE TO THE 6845
E4B2  C3                        RET
E4B3                    C25     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
                        ;--------------------------------------------
E4B3                    ACT_DISP_PAGE   PROC    NEAR
E4B3  A8 80                     TEST    AL,080H                 ; CRT/CPU PAGE REG FUNCTION
E4B5  75 24                     JNZ     SET_CRTCPU              ; YES, GO HANDLE IT
E4B7  A2 0062 R                 MOV     ACTIVE_PAGE,AL          ; SAVE ACTIVE PAGE VALUE
E4BA  8B 0E 004C R              MOV     CX,CRT_LEN              ; GET SAVED LENGTH OF REGEN BUFFER
E4BE  98                        CBW                             ; CONVERT AL TO WORD
E4BF  50                        PUSH    AX                      ; SAVE PAGE VALUE
E4C0  F7 E1                     MUL     CX                      ; DISPLAY PAGE TIMES REGEN LENGTH
E4C2  A3 004E R                 MOV     CRT_START,AX            ; SAVE START ADDRESS FOR LATER USE
E4C5  8B C8                     MOV     CX,AX                   ; START ADDRESS TO CX
E4C7  D1 F9                     SAR     CX,1                    ; DIVIDE BY 2 FOR 6845 HANDLING
E4C9  B4 0C                     MOV     AH,12                   ; 6845 REGISTER FOR START ADDRESS
E4CB  E8 E472 R                 CALL    C23
E4CE  58                        POP     BX                      ; RECOVER PAGE VALUE
E4CF  D1 E3                     SAL     BX,1                    ; *2 FOR WORD OFFSET
E4D1  8B 87 0050 R              MOV     AX,[BX + OFFSET CURSOR_POSN] ; GET CURSOR FOR THIS
                                                                ; PAGE
E4D5  E8 E4A2 R                 CALL    C25                     ; SET THE CURSOR POSITION
E4D8  E9 0F70 R                 JMP     VIDEO_RETURN
                        ;--------------------------------------------------------
                        ; SET_CRTCPU
                        ;       THIS ROUTINE READS OR WRITES THE CRT/CPU PAGE REGISTERS
                        ;
                        ; INPUT
                        ;       (AL) = 83H      SET BOTH CRT AND CPU PAGE REGS
                        ;                 (BH) =  VALUE TO SET IN CRT PAGE REG
                        ;                 (BL) =  VALUE TO SET IN CPU PAGE REG
                        ;       (AL) = 82H      SET CRT PAGE REG
                        ;                 (BH) =  VALUE TO SET IN CRT PAGE REG
                        ;       (AL) = 81H      SET CPU PAGE REG
                        ;                 (BL) =  VALUE TO SET IN CPU PAGE REG
                        ;       (AL) = 80H      READ CURRENT VALUE OF CRT/CPU PAGE REGS
                        ;
                        ; OUTPUT
                        ;       ALL FUNCTIONS RETURN
                        ;                 (BH) = CURRENT CONTENTS OF CRT PAGE REG
                        ;                 (BL) = CURRENT CONTENTS OF CPU PAGE REG
                        ;--------------------------------------------------------
E4DB                    SET_CRTCPU:
E4DB  8A E0                     MOV     AH,AL                   ; SAVE REQUEST IN AH
E4DD  BA 03DA                   MOV     DX,VGA_CTL              ; SET ADDRESS OF GATE ARRAY
E4E0  EC                C26:    IN      AL,DX                   ; GET STATUS
E4E1  24 08                     AND     AL,08H                  ; VERTICAL RETRACE?
E4E3  74 FB                     JZ      C26                     ; NO, WAIT FOR IT
E4E5  BA 03DF                   MOV     DX,PAGREG               ; SET IO ADDRESS OF PAGE REG
E4E8  A0 008A R                 MOV     AL,PAGDAT               ; GET DATA LAST OUTPUT TO REG
E4EB  80 FC 80                  CMP     AH,80H                  ; READ FUNCTION REQUESTED?
E4EE  74 27                     JZ      C29                     ; YES, DON'T SET ANYTHING
E4F0  80 FC 84                  CMP     AH,84H                  ; VALID REQUEST?
E4F3  73 22                     JNC     C29                     ; NO, PRETEND IT WAS A READ REQUEST
E4F5  F6 C4 01                  TEST    AH,1                    ; SET CPU REG?
E4F8  74 0D                     JZ      C27                     ; NO, GO SEE ABOUT CRT REG
E4FA  D0 E3                     SHL     BL,1                    ; SHIFT VALUE TO RIGHT BIT POSITION
E4FC  D0 E3                     SHL     BL,1
E4FE  D0 E3                     SHL     BL,1
E500  24 C7                     AND     AL,NOT CPUREG           ; CLEAR OLD CPU VALUE
E502  80 E3 38                  AND     BL,CPUREG               ; BE SURE UNRELATED BITS ARE ZERO
E505  0A C3                     OR      AL,BL                   ; OR IN NEW VALUE
```

```
E507  F6 C4 02                C27:    TEST    AH,2            ; SET CRT REG?
E50A  74 07                           JZ      C28             ; NO, GO RETURN CURRENT SETTINGS
E50C  24 F8                           AND     AL,NOT CRTREG   ; CLEAR OLD CRT VALUE
E50E  80 E7 07                        AND     BH,CRTREG       ; BE SURE UNRELATED BITS ARE ZERO
E511  0A C7                           OR      AL,BH           ; OR IN NEW VALUE
E513  EE                      C28:    OUT     DX,AL           ; SET NEW VALUES
E514  A2 008A R                       MOV     PAGDAT,AL       ; SAVE COPY IN RAM
E517  8A D8                   C29:    MOV     BL,AL           ; GET CPU REG VALUE
E519  80 E3 38                        AND     BL,CPUREG       ; CLEAR EXTRA BITS
E51C  D0 FB                           SAR     BL,1            ; RIGHT JUSTIFY IN BL
E51E  D0 FB                           SAR     BL,1
E520  D0 FB                           SAR     BL,1
E522  8A F8                           MOV     BH,AL           ; GET CRT REG VALUE
E524  80 E7 07                        AND     BH,CRTREG       ; CLEAR EXTRA BITS
E527  5F                              POP     DI              ; RESTORE SOME REGS
E528  5E                              POP     SI
E529  58                              POP     AX              ; DISCARD SAVED BX
E52A  E9 0F73 R                       JMP     C22             ; RETURN
E52D                          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
                              ;--------------------------------------------
E52D                          READ_CURSOR     PROC    NEAR
E52D  8A DF                           MOV     BL,BH
E52F  32 FF                           XOR     BH,BH
E531  D1 E3                           SAL     BX,1            ; WORD OFFSET
E533  8B 97 0050 R                    MOV     DX,[BX+OFFSET CURSOR_POSN]
E537  8B 0E 0060 R                    MOV     CX,CURSOR_MODE
E53B  5F                              POP     DI
E53C  5E                              POP     SI
E53D  5B                              POP     BX
E53E  58                              POP     AX              ; DISCARD SAVED CX AND DX
E53F  58                              POP     AX
E540  1F                              POP     DS
E541  07                              POP     ES
E542  CF                              IRET
E543                          READ_CURSOR     ENDP
                              ;--------------------------------------------
                              ; SET COLOR
                              ;       THIS ROUTINE WILL ESTABLISH THE BACKGROUND COLOR, THE
                              ;       OVERSCAN COLOR, AND THE FOREGROUND COLOR SET FOR GRAPHICS
                              ; INPUT
                              ;       (BH) HAS COLOR ID
                              ;               IF BH=0, THE BACKGROUND COLOR VALUE IS SET
                              ;                       FROM THE LOW BITS OF BL (0-31)
                              ;                       IN GRAPHIC MODES, BOTH THE BACKGROUND AND
                              ;                       BORDER ARE SET. IN ALPHA MODES, ONLY THE
                              ;                       BORDER IS SET.
                              ;               IF BH=1, THE PALETTE SELECTION IS MADE
                              ;                       BASED ON THE LOW BIT OF BL:
                              ;                          2 COLOR MODE:
                              ;                               0 = WHITE FOR COLOR 1
                              ;                               1 = BLACK FOR COLOR 1
                              ;                          4 COLOR MODES:
                              ;                               0 = GREEN, RED, YELLOW FOR
                              ;                                   COLORS 1,2,3
                              ;                               1 = BLUE, CYAN, MAGENTA FOR
                              ;                                   COLORS 1,2,3
                              ;                          16 COLOR MODES:
                              ;                               ALWAYS SETS UP PALETTE AS:
                              ;                               BLUE FOR COLOR 1
                              ;                               GREEN FOR COLOR 2
                              ;                               CYAN FOR COLOR 3
                              ;                               RED FOR COLOR 4
                              ;                               MAGENTA FOR COLOR 5
                              ;                               BROWN FOR COLOR 6
                              ;                               LIGHT GRAY FOR COLOR 7
                              ;                               DARK GRAY FOR COLOR 8
                              ;                               LIGHT BLUE FOR COLOR 9
                              ;                               LIGHT GREEN FOR COLOR 10
                              ;                               LIGHT CYAN FOR COLOR 11
                              ;                               LIGHT RED FOR COLOR 12
                              ;                               LIGHT MAGENTA FOR COLOR 13
                              ;                               YELLOW FOR COLOR 14
                              ;                               WHITE FOR COLOR 15
                              ;       (BL) HAS THE COLOR VALUE TO BE USED
                              ; OUTPUT
                              ;       THE COLOR SELECTION IS UPDATED
                              ;--------------------------------------------
E543                          SET_COLOR       PROC    NEAR
E543  BA 03DA                         MOV     DX,VGA_CTL      ; I/O PORT FOR PALETTE
E546  EC                      C30:    IN      AL,DX           ; SYNC UP VGA FOR REG ADDRESS
E547  A8 08                           TEST    AL,8            ; IS VERTICAL RETRACE ON?
E549  74 FB                           JZ      C30             ; NO, WAIT UNTIL IT IS
E54B  0A FF                           OR      BH,BH           ; IS THIS COLOR 0?
E54D  75 19                           JNZ     C31             ; OUTPUT COLOR 1
```

```
                        ;------ HANDLE COLOR 0 BY SETTING THE BACKGROUND COLOR
                        ;               AND BORDER COLOR
E54F  80 3E 0049 R 04           CMP     CRT_MODE,4              ; IN ALPHA MODE?
E554  72 06                     JC      C305                    ; YES, JUST SET BORDER REG
E556  B0 10                     MOV     AL,10H                  ; SET PALETTE REG 0
E558  EE                        OUT     DX,AL                   ; SELECT VGA REG
E559  8A C3                     MOV     AL,BL                   ; GET COLOR
E55B  EE                        OUT     DX,AL                   ; SET IT
E55C  B0 02                     MOV     AL,2                    ; SET BORDER REG
E55E  EE                        OUT     DX,AL                   ; SELECT VGA BORDER REG
E55F  8A C3                     MOV     AL,BL                   ; GET COLOR
E561  EE                        OUT     DX,AL                   ; SET IT
E562  A2 0066 R                 MOV     CRT_PALLETTE,AL         ; SAVE THE COLOR VALUE
E565  E9 0F70 R                 JMP     VIDEO_RETURN
                        ;------ HANDLE COLOR 1 BY CHANGING PALETTE REGISTERS
E568  A0 0049 R         C31:    MOV     AL,CRT_MODE             ; GET CURRENT MODE
E56B  B9 0D95 R                 MOV     CX,OFFSET M0072         ; POINT TO 2 COLOR TABLE ENTRY
E56E  3C 06                     CMP     AL,6                    ; 2 COLOR MODE?
E570  74 0F                     JE      C33                     ; YES, JUMP
E572  3C 04                     CMP     AL,4                    ; 4 COLOR MODE?
E574  74 08                     JE      C32                     ; YES, JUMP
E576  3C 05                     CMP     AL,5                    ; 4 COLOR MODE?
E578  74 04                     JE      C32                     ; YES, JUMP
E57A  3C 0A                     CMP     AL,0AH                  ; 4 COLOR MODE?
E57C  75 20                     JNE     C36                     ; NO, GO TO 16 COLOR SET UP
E57E  B9 0D9D R         C32:    MOV     CX,OFFSET M0074         ; POINT TO 4 COLOR TABLE ENTRY
E581  D0 CB             C33:    ROR     BL,1                    ; SELECT ALTERNATE SET?
E583  73 03                     JNC     C34                     ; NO, JUMP
E585  83 C1 04                  ADD     CX,M0072L               ; POINT TO NEXT ENTRY
E588  8B D9             C34:    MOV     BX,CX                   ; TABLE ADDRESS IN BX
E58A  43                        INC     BX                      ; SKIP OVER BACKGROUND COLOR
E58B  B9 0003                   MOV     CX,M0072L-1             ; SET NUMBER OF REGS TO FILL
E58E  B4 11                     MOV     AH,11H                  ; AH IS REGISTER COUNTER
E590  8A C4             C35:    MOV     AL,AH                   ; GET REG NUMBER
E592  EE                        OUT     DX,AL                   ; SELECT IT
E593  2E: 8A 07                 MOV     AL,CS:[BX]              ; GET DATA
E596  EE                        OUT     DX,AL                   ; SET IT
E597  FE C4                     INC     AH                      ; NEXT REG
E599  43                        INC     BX                      ; NEXT TABLE VALUE
E59A  E2 F4                     LOOP    C35
E59C  EB 0D                     JMP     SHORT C38
E59E  B4 11             C36:    MOV     AH,11H                  ; AH IS REGISTER COUNTER
E5A0  B9 000F                   MOV     CX,15                   ; NUMBER OF PALETTES
E5A3  8A C4             C37:    MOV     AL,AH                   ; GET REG NUMBER
E5A5  EE                        OUT     DX,AL                   ; SELECT IT
E5A6  EE                        OUT     DX,AL                   ; SET PALETTE VALUE
E5A7  FE C4                     INC     AH                      ; NEXT REG
E5A9  E2 F8                     LOOP    C37
E5AB  32 C0             C38:    XOR     AL,AL                   ; SELECT LOW REG TO ENABLE VIDEO
                                                                ; AGAIN
E5AD  EE                        OUT     DX,AL
E5AE  E9 0F70 R                 JMP     VIDEO_RETURN
E5B1                    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
                        ;--------------------------------------------------
E5B1                    VIDEO_STATE     PROC    NEAR
E5B1  8A 26 004A R              MOV     AH,BYTE PTR CRT_COLS ; GET NUMBER OF COLUMNS
E5B5  A0 0049 R                 MOV     AL,CRT_MODE     ; CURRENT MODE
E5B8  8A 3E 0062 R              MOV     BH,ACTIVE_PAGE  ; GET CURRENT ACTIVE PAGE
E5BC  5F                        POP     DI              ; RECOVER REGISTERS
E5BD  5E                        POP     SI              ;
E5BE  59                        POP     CX              ; DISCARD SAVED BX
E5BF  E9 0F73 R                 JMP     C22             ; RETURN TO CALLER
E5C2                    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
                        ;--------------------------------
E5C2                    POSITION        PROC    NEAR
E5C2  53                        PUSH    BX              ; SAVE REGISTER
E5C3  8B D8                     MOV     BX,AX
E5C5  8A C4                     MOV     AL,AH           ; ROWS TO AL
E5C7  F6 26 004A R              MUL     BYTE PTR CRT_COLS       ; DETERMINE BYTES TO ROW
E5CB  32 FF                     XOR     BH,BH
E5CD  03 C3                     ADD     AX,BX           ; ADD IN COLUMN VALUE
E5CF  D1 E0                     SAL     AX,1            ; * 2 FOR ATTRIBUTE BYTES
E5D1  5B                        POP     BX
E5D2  C3                        RET
E5D3                    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
E5D3                    SCROLL_UP       PROC    NEAR
E5D3  8A D8                     MOV     BL,AL                   ; SAVE LINE COUNT IN BL
E5D5  80 FC 04                  CMP     AH,4                    ; TEST FOR GRAPHICS MODE
E5D8  72 03                     JC      C39                     ; HANDLE SEPARATELY
E5DA  E9 F259 R                 JMP     GRAPHICS_UP
E5DD                    C39:                                    ; UP_CONTINUE
E5DD  53                        PUSH    BX                      ; SAVE FILL ATTRIBUTE IN BH
E5DE  8B C1                     MOV     AX,CX                   ; UPPER LEFT POSITION
E5E0  E8 E609 R                 CALL    SCROLL_POSITION         ; DO SETUP FOR SCROLL
E5E3  74 20                     JZ      C44                     ; BLANK_FIELD
E5E5  03 F0                     ADD     SI,AX                   ; FROM ADDRESS
E5E7  8A E6                     MOV     AH,DH                   ; # ROWS IN BLOCK
E5E9  2A E3                     SUB     AH,BL                   ; # ROWS TO BE MOVED
E5EB  E8 E62F R         C40:    CALL    C45                     ; MOVE ONE ROW
E5EE  03 F5                     ADD     SI,BP
E5F0  03 FD                     ADD     DI,BP                   ; POINT TO NEXT LINE IN BLOCK
E5F2  FE CC                     DEC     AH                      ; COUNT OF LINES TO MOVE
E5F4  75 F5                     JNZ     C40                     ; ROW_LOOP
E5F6  58                C41:    POP     AX                      ; RECOVER ATTRIBUTE IN AH
E5F7  B0 20                     MOV     AL,' '                  ; FILL WITH BLANKS
E5F9  E8 E638 R         C42:    CALL    C46                     ; CLEAR THE ROW
E5FC  03 FD                     ADD     DI,BP                   ; POINT TO NEXT LINE
E5FE  FE CB                     DEC     BL                      ; COUNTER OF LINES TO SCROLL
E600  75 F7                     JNZ     C42                     ; CLEAR_LOOP
E602  E9 0F70 R         C43:    JMP     VIDEO_RETURN
E605  8A DE             C44:    MOV     BL,DH                   ; GET ROW COUNT
E607  E8 ED                     JMP     C41                     ; GO CLEAR THAT AREA
E609                    SCROLL_UP       ENDP
                        ;----- HANDLE COMMON SCROLL SET UP HERE
E609                    SCROLL_POSITION PROC    NEAR
E609  E8 E5C2 R                 CALL    POSITION                ; CONVERT TO REGEN POINTER
E60C  03 06 004E R              ADD     AX,CRT_START            ; OFFSET OF ACTIVE PAGE
E610  8B F8                     MOV     DI,AX                   ; TO ADDRESS FOR SCROLL
E612  8B F0                     MOV     SI,AX                   ; FROM ADDRESS FOR SCROLL
E614  2B D1                     SUB     DX,CX                   ; DX = #ROWS, #COLS IN BLOCK
E616  FE C6                     INC     DH
E618  FE C2                     INC     DL                      ; INCREMENT FOR 0 ORIGIN
E61A  32 ED                     XOR     CH,CH                   ; SET HIGH BYTE OF COUNT TO ZERO
E61C  8B 2E 004A R              MOV     BP,CRT_COLS             ; GET NUMBER OF COLUMNS IN DISPLAY
E620  03 ED                     ADD     BP,BP                   ; TIMES 2 FOR ATTRIBUTE BYTE
E622  8A C3                     MOV     AL,BL                   ; GET LINE COUNT
E624  F6 26 004A R              MUL     BYTE PTR CRT_COLS       ; DETERMINE OFFSET TO FROM
                                                                ; ADDRESS
E628  03 C0                     ADD     AX,AX                   ; *2 FOR ATTRIBUTE BYTE
E62A  06                        PUSH    ES                      ; ESTABLISH ADDRESSING TO REGEN
                                                                ; BUFFER
E62B  1F                        POP     DS                      ;  FOR BOTH POINTERS
E62C  0A DB                     OR      BL,BL                   ; 0 SCROLL MEANS BLANK FIELD
E62E  C3                        RET                             ; RETURN WITH FLAGS SET
E62F                    SCROLL_POSITION ENDP
                        ;------ MOVE_ROW
E62F                    C45     PROC    NEAR
E62F  8A CA                     MOV     CL,DL                   ; GET # OF COLS TO MOVE
E631  56                        PUSH    SI
E632  57                        PUSH    DI                      ; SAVE START ADDRESS
E633  F3/ A5                    REP     MOVSW                   ; MOVE THAT LINE ON SCREEN
E635  5F                        POP     DI
E636  5E                        POP     SI                      ; RECOVER ADDRESSES
E637  C3                        RET
E638                    C45     ENDP
                        ;------ CLEAR_ROW
E638                    C46     PROC    NEAR
E638  8A CA                     MOV     CL,DL                   ; GET # COLUMNS TO CLEAR
E63A  57                        PUSH    DI
E63B  F3/ AB                    REP     STOSW                   ; STORE THE FILL CHARACTER
E63D  5F                        POP     DI
E63E  C3                        RET
E63F                    C46     ENDP
                        ;--------------------------------------
                        ; SCROLL_DOWN
                        ;       THIS ROUTINE MOVES THE CHARACTERS WITHIN A DEFINED
                        ;       BLOCK DOWN ON THE SCREEN, FILLING THE TOP LINES
                        ;       WITH A DEFINED CHARACTER
                        ; INPUT
                        ;       (AH) = CURRENT CRT MODE
                        ;       (AL) = NUMBER OF LINES TO SCROLL
                        ;       (CX) = UPPER LEFT CORNER OF REGION
                        ;       (DX) = LOWER RIGHT CORNER OF REGION
                        ;       (BH) = FILL CHARACTER
                        ;       (DS) = DATA SEGMENT
                        ;       (ES) = REGEN SEGMENT
                        ; OUPUT
                        ;       NONE -- SCREEN IS SCROLLED
                        ;--------------------------------------
E63F                    SCROLL_DOWN     PROC    NEAR
E63F  FD                        STD                             ; DIRECTION FOR SCROLL DOWN
E640  8A D8                     MOV     BL,AL                   ; LINE COUNT TO BL
E642  80 FC 04                  CMP     AH,4                    ; TEST FOR GRAPHICS
E645  72 03                     JC      C47
E647  E9 F305 R                 JMP     GRAPHICS_DOWN
E64A  53                C47:    PUSH    BX                      ; SAVE ATTRIBUTE IN BH
E64B  8B C2                     MOV     AX,DX                   ; LOWER RIGHT CORNER
E64D  E8 E609 R                 CALL    SCROLL_POSITION         ; GET REGEN LOCATION
E650  74 1F                     JZ      C51
E652  2B F0                     SUB     SI,AX                   ; SI IS FROM ADDRESS
E654  8A E6                     MOV     AH,DH                   ; GET TOTAL # ROWS
E656  2A E3                     SUB     AH,BL                   ; COUNT TO MOVE IN SCROLL
```

```
E658  E8 E62F R       C48:    CALL    C45             ; MOVE ONE ROW
E65B  2B F5                   SUB     SI,BP
E65D  2B FD                   SUB     DI,BP
E65F  FE CC                   DEC     AH
E661  75 F5                   JNZ     C48
E663  58              C49:    POP     AX              ; RECOVER ATTRIBUTE IN AH
E664  B0 20                   MOV     AL,' '
E666  E8 E63B R       C50:    CALL    C46             ; CLEAR ONE ROW
E669  2B FD                   SUB     DI,BP           ; GO TO NEXT ROW
E66B  FE CB                   DEC     BL
E66D  75 F7                   JNZ     C50
E66F  EB 91                   JMP     C43             ; SCROLL_END
E671  8A DE           C51:    MOV     BL,DH
E673  EB EE                   JMP     C49
E675                  SCROLL_DOWN     ENDP
                      ;--------------------------------------------
                      ; MODE_ALIVE
                      ;       THIS ROUTINE READS 256 LOCATIONS IN MEMORY AS EVERY OTHER
                      ;       LOCATION IN 512 LOCATIONS.  THIS IS TO INSURE THE DATA
                      ;       INTEGRITY OF MEMORY DURING MODE CHANGES.
                      ;--------------------------------------------
E675                  MODE_ALIVE      PROC    NEAR
E675  50                      PUSH    AX              ;SAVE USED REGS
E676  56                      PUSH    SI
E677  51                      PUSH    CX
E678  33 F6                   XOR     SI,SI
E67A  B9 0100                 MOV     CX,256
E67D  AC              C52:    LODSB
E67E  46                      INC     SI
E67F  E2 FC                   LOOP    C52
E681  59                      POP     CX
E682  5E                      POP     SI
E683  58                      POP     AX
E684  C3                      RET
E685                  MODE_ALIVE      ENDP
                      ;--------------------------------------------------------
                      ; SET_PALLETTE
                      ;       THIS ROUTINE WRITES THE PALETTE REGISTERS
                      ; INPUT
                      ;       (AL) = 0        SET PALETTE REG
                      ;              (BH) =   VALUE TO SET
                      ;              (BL) =   PALETTE REG TO SET
                      ;       (AL) = 1        SET BORDER COLOR REG
                      ;              (BH) =   VALUE TO SET
                      ;       (AL) = 2        SET ALL PALETTE REGS AND BORDER REG
                      ;       NOTE: REGISTERS ARE WRITE ONLY.
                      ;--------------------------------------------------------
E685                  SET_PALLETTE    PROC    NEAR
E685  50                      PUSH    AX
E686  8B F4                   MOV     SI,SP
E688  36: 8B 44 0C            MOV     AX,SS:[SI+12]   ; GET SEG FROM STACK
E68C  8E C0                   MOV     ES,AX
E68E  8B F2                   MOV     SI,DX           ; OFFSET IN SI
E690  BA 03DA                 MOV     DX,VGA_CTL      ; SET VGA CONTROL PORT
E693  EC              C53:    IN      AL,DX           ; GET VGA STATUS
E694  24 08                   AND     AL,08H          ; IN VERTICAL RETRACE?
E696  75 FB                   JNZ     C53             ; YES, WAIT FOR IT TO GO AWAY
E698  EC              C54:    IN      AL,DX           ; GET VGA STATUS
E699  24 08                   AND     AL,08H          ; IN VERITCAL RETRACE?
E69B  74 FB                   JZ      C54             ; NO, WAIT FOR IT
E69D  58                      POP     AX
E69E  0A C0                   OR      AL,AL           ; SET PALETTE REG?
E6A0  74 0C                   JZ      C55             ; YES, GO DO IT
E6A2  3C 02                   CMP     AL,2            ; SET ALL REGS?
E6A4  74 17                   JE      C57             ;
E6A6  3C 01                   CMP     AL,1            ; SET BORDER COLOR REG?
E6A8  75 2B                   JNE     C59             ; NO, DON'T DO ANYTHING
E6AA  B0 02                   MOV     AL,2            ; SET BORDER COLOR REG NUMBER
E6AC  EB 06                   JMP     SHORT C56
E6AE  8A C3           C55:    MOV     AL,BL           ; GET DESIRED REG NUMBER IN AL
E6B0  24 0F                   AND     AL,0FH          ; STRIP UNUSED BITS
E6B2  0C 10                   OR      AL,10H          ; MAKE INTO REAL REG NUMBER
E6B4  EE              C56:    OUT     DX,AL           ; SELECT REG
E6B5  8A C7                   MOV     AL,BH           ; GET DATA IN AL
E6B7  EE                      OUT     DX,AL           ; SET NEW DATA
E6B8  32 C0                   XOR     AL,AL           ; SET REG 0 SO DISPLAY WORKS AGAIN
E6BA  EE                      OUT     DX,AL
E6BB  EB 18                   JMP     SHORT C59
E6BD  B4 10           C57:    MOV     AH,10H          ; AH IS REG COUNTER
E6BF  8A C4           C58:    MOV     AL,AH           ; REG ADDRESS IN AL
E6C1  EE                      OUT     DX,AL           ; SELECT IT
E6C2  26: 8A 04               MOV     AL,BYTE PTR ES:[SI]     ;GET DATA
E6C5  EE                      OUT     DX,AL           ; PUT IN VGA REG
E6C6  46                      INC     SI              ; NEXT DATA BYTE
E6C7  FE C4                   INC     AH              ; NEXT REG
E6C9  80 FC 20                CMP     AH,20H          ; LAST PALETTE REG?
E6CC  72 F1                   JB      C58             ; NO, DO NEXT ONE
E6CE  B0 02                   MOV     AL,2            ; SET BORDER REG
E6D0  EE                      OUT     DX,AL           ; SELECT IT
E6D1  26: 8A 04               MOV     AL,BYTE PTR ES:[SI]     ; GET DATA
E6D4  EE                      OUT     DX,AL           ; PUT IN VGA REG
```

```
E6D4  EE                        OUT     DX,AL           ; PUT IN VGA REG
E6D5  E9 0F70 R                 JMP     VIDEO_RETURN    ; ALL DONE
E6D8                    C59:    JMP     VIDEO_RETURN    ; ALL DONE
E6D8                    SET_PALLETTE    ENDP
E6D8                    MFG_UP  PROC    NEAR
E6D8  50                        PUSH    AX
E6D9  1E                        PUSH    DS
                                ASSUME  DS:XXDATA
E6DA  B8  ---- R                MOV     AX,XXDATA
E6DD  8E D8                     MOV     DS,AX
E6DF  A0 0005 R                 MOV     AL,MFG_TST      ; GET MFG CHECKPOINT
E6E2  E6 10                     OUT     10H,AL          ; OUTPUT IT TO TESTER
E6E4  FE C8                     DEC     AL              ; DROP IT BY 1 FOR THE NEXT TEST
E6E6  A2 0005 R                 MOV     MFG_TST,AL
                                ASSUME  DS:ABS0
E6E9  1F                        POP     DS
E6EA  58                        POP     AX
E6EB  C3                        RET
E6EC                    MFG_UP  ENDP
                                ASSUME  CS:CODE,DS:DATA
E6F2                            ORG     0E6F2H
E6F2  E9 0B1B R                 JMP     NEAR PTR BOOT_STRAP
                        ;-------------------------------------------------------------------------
                        ;   SUBROUTINE TO SET UP CONDITIONS FOR THE TESTING OF 8250 AND
                        ;       8259 INTERRUPTS.  ENABLES MASKABLE EXTERNAL INTERRUPTS,
                        ;       CLEARS THE 8259 INTR RECEIVED FLAG BIT, AND ENABLES THE
                        ;       DEVICE'S 8259 INTR (WHICHEVER IS BEING TESTED).
                        ;   IT EXPECTS TO BE PASSED:
                        ;       (DS) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED
                        ;       (DI) = OFFSET OF THE INTERRUPT BIT MASK
                        ;   UPON RETURN:
                        ;       INTR_FLAG BIT FOR THE DEVICE = 0
                        ;   NO REGISTERS ARE ALTERED.
                        ;-------------------------------------------------------------------------
E6F5                    SUI     PROC    NEAR
E6F5  50                        PUSH    AX
E6F6  FB                        STI                     ; ENABLE MASKABLE EXTERNAL
                                                        ; INTERRUPTS
E6F7  2E: 8A 25                 MOV     AH,CS:[DI]      ; GET INTERRUPT BIT MASK
E6FA  20 26 0084 R              AND     INTR_FLAG,AH    ; CLEAR 8259 INTERRUPT REC'D FLAG
                                                        ; BIT
E6FE  E4 21                     IN      AL,INTA01       ; CURRENT INTERRUPTS
E700  22 C4                     AND     AL,AH           ; ENABLE THIS INTERRUPT, TOO
E702  E6 21                     OUT     INTA01,AL       ; WRITE TO 8259 (INTERRUPT
                                                        ; CONTROLLER)
E704  58                        POP     AX
E705  C3                        RET
E706                    SUI     ENDP
                        ;-------------------------------------------------------------------------
                        ;   SUBROUTINE WHICH CHECKS IF A 8259 INTERRUPT IS GENERATED BY THE
                        ;       8250 INTERRUPT.
                        ;   IT EXPECTS TO BE PASSED:
                        ;       (DI) = OFFSET OF INTERRUPT BIT MASK
                        ;       (DS) = ADDRESS OF SEGMENT WHERE INTR_FLAG IS DEFINED.
                        ;   IT RETURNS:
                        ;       (CF) = 1 IF NO INTERRUPT IS GENERATED
                        ;              0 IF THE INTERRUPT OCCURRED
                        ;       (AL) = COMPLEMENT OF THE INTERRUPT MASK
                        ;   NO OTHER REGISTERS ARE ALTERED.
                        ;-------------------------------------------------------------------------
E706                    C5059   PROC    NEAR
E706  51                        PUSH    CX
E707  2B C9                     SUB     CX,CX           ; SET PROGRAM LOOP COUNT
E709  2E: 8A 05                 MOV     AL,CS:[DI]      ; GET INTERRUPT MASK
E70C  34 FF                     XOR     AL,0FFH         ; COMPLEMENT MASK SO ONLY THE INTR
                                                        ; TEST BIT IS ON
E70E  84 06 0084 R      AT25:   TEST    INTR_FLAG,AL    ; 8259 INTERRUPT OCCUR?
E712  75 03                     JNE     AT27            ; YES - CONTINUE
E714  E2 F8                     LOOP    AT25            ; WAIT SOME MORE
E716  F9                        STC                     ; TIME'S UP - FAILED
E717  59                AT27:  POP     CX
E718  C3                        RET
E719                    C5059   ENDP
                        ;-------------------------------------------------------------------------
                        ;   SUBROUTINE TO WAIT FOR ALL ENABLED 8250 INTERRUPTS TO CLEAR (SO
                        ;       NO INTRS WILL BE PENDING).  EACH INTERRUPT COULD TAKE UP TO
                        ;       1 MILLISECOND TO CLEAR. THE INTERRUPT IDENTIFICATION
                        ;       REGISTER WILL BE CHECKED UNTIL THE INTERRUPT(S) IS CLEARED
                        ;       OR A TIMEOUT OCCURS.
                        ;   EXPECTS TO BE PASSED:
                        ;       (DX) = ADDRESS OF THE INTERRUPT ID REGISTER
                        ;   RETURNS:
                        ;       (AL) = CONTENTS OF THE INTR ID REGISTER
                        ;       (CF) = 1  IF INTERRUPTS ARE STILL PENDING
                        ;              0  IF NO INTERRUPTS ARE PENDING (ALL CLEAR)
                        ;   NO OTHER REGISTERS ARE ALTERED.
                        ;-------------------------------------------------------------------------
E719                    W8250C  PROC    NEAR
E719  51                        PUSH    CX
E71A  2B C9                     SUB     CX,CX
E71C  EC                AT28:   IN      AL,DX           ; READ INTR ID REG
E71D  3C 01                     CMP     AL,1            ; INTERRUPTS STILL PENDING?
E71F  74 05                     JE      AT29            ; NO - GOOD FINISH
E721  E2 F9                     LOOP    AT28            ; KEEP TRYING
E723  F9                        STC                     ; TIME'S UP - ERROR
E724  EB 01                     JMP     SHORT AT30
E726  F8                AT29:  CLC
E727  59                AT30:  POP     CX
E728  C3                        RET
E729                    W8250C  ENDP
```

```
                ;-----INT 14----------------------------------------------
                ;RS232_IO
                ;       THIS ROUTINE PROVIDES BYTE STREAM I/O TO THE COMMUNICATIONS
                ;       PORT ACCORDING TO THE PARAMETERS:
                ;       (AH)=0  INITIALIZE THE COMMUNICATIONS PORT
                ;               (AL) HAS PARMS FOR INITIALIZATION
                ;
                ;---7-------6-------5-------4-------3-------2---------1-------0----
                ;-------- BAUD RATE ---:::----PARITY----:::-STOPBIT-:::--WORD LENGTH--
                ;
                ;  000 - 110              XO - 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 - 4800
                ;               ON RETURN,  THE RS232 INTERRUPTS ARE DISABLED AND
                ;                           CONDITIONS ARE 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, REFELECTING THE CURRENT STATUS OF
                ;                       THE LINE.
                ;       (AH)=2  RECEIVE A CHARACTER IN (AL) FROM COMMO LINE BEFORE
                ;                       RETURNING TO CALLER
                ;               ON EXIT,  AH HAS THE CURRENT LINE STATUS, AS SET BY
                ;                       THE STATUS ROUTINE, EXCEPT THAT THE ONLY
                ;                       BITS LEFT ON, ARE THE ERROR BITS
                ;                       (7,4,3,2,1). IN THIS CASE, THE TIME OUT BIT
                ;                       INDICATES DATA SET READY WAS NOT RECEIVED.
                ;                       THUS, AH IS NON ZERO ONLY WHEN AN ERROR
                ;                       OCCURRED.(NOTE: IF THE TIME-OUT BIT IS SET,
                ;                       OTHER BITS IN AH MAY NOT BE RELIABLE.)
                ;       (AH)=3  RETURN THE COMMO PORT STATUS IN (AX)
                ;               AH CONTAINS THE LINE CONTROL STATUS
                ;               BIT 7 = TIME OUT
                ;               BIT 6 = TRANS SHIFT REGISTER EMPTY
                ;               BIT 5 = TRAN HOLDING REGISTER EMPTY
                ;               BIT 4 = BREAK DETECT
                ;               BIT 3 = FRAMING ERROR
                ;               BIT 2 = PARITY ERROR
                ;               BIT 1 = OVERRUN ERROR
                ;               BIT 0 = DATA READY
                ;               AL CONTAINS THE MODEM STATUS
                ;               BIT 7 = RECIEVED 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 RS232_TIM_OUT (BYTE) CONTAINS OUTER LOOP COUNT
                ;       VALUE FOR TIMEOUT (DEFAULT=1)
                ;OUTPUT
                ;               AX      MODIFIED ACCORDING TO PARMS OF CALL
                ;               ALL OTHERS UNCHANGED
                ;-----------------------------------------------
E729                    ASSUME  CS:CODE,DS:DATA
E729                    ORG     0E729H
E729 03F9       A1      LABEL   WORD
E72B 02EA               DW      1017    ; 110 BAUD      ; TABLE OF INIT VALUE
E72D 0175               DW      746     ; 150
E72F 00BA               DW      373     ; 300
E731 005D               DW      186     ; 600
E733 002F               DW      93      ; 1200
E735 0017               DW      47      ; 2400
E737 0017               DW      23      ; 4800
E739                    DW      23      ; 4800
                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 8B F2              MOV     SI,DX           ; RS232 VALUE TO SI
E742 8B FA              MOV     DI,DX           ; AND TO DI (FOR TIMEOUTS)
E744 D1 E6              SHL     SI,1            ; WORD OFFSET
E746 E8 1388 R          CALL    DDS             ; POINT TO BIOS DATA SEGMENT
E749 8B 94 0000 R       MOV     DX,RS232_BASE[SI] ; GET BASE ADDRESS
E74D 0B D2              OR      DX,DX           ; TEST FOR 0 BASE ADDRESS
E74F 74 13              JZ      A3              ; RETURN
E751 0A E4              OR      AH,AH           ; TEST FOR (AH)=0
E753 74 16              JZ      A4              ; COMMUN INIT
E755 FE CC              DEC     AH              ; TEST FOR (AH)=1
E757 74 47              JZ      A5              ; SEND AL
E759 FE CC              DEC     AH              ; TEST FOR (AH)=2
E75B 74 6C              JZ      A12             ; RECEIVE INTO AL
E75D FE CC              DEC     AH              ; TEST FOR (AH)=3
E75F 75 03              JNZ     A3
E761 E9 E7F3 R          JMP     A18             ; COMMUNICATION STATUS
```

```
E764                                    A3:                             ; RETURN FROM RS232
E764  5B                                        POP     BX
E765  59                                        POP     CX
E766  5F                                        POP     DI
E767  5E                                        POP     SI
E768  5A                                        POP     DX
E769  1F                                        POP     DS
E76A  CF                                        IRET                    ; RETURN TO CALLER, NO ACTION
                                                ;------ INITIALIZE THE COMMUNICATIONS PORT
E76B  8A E0                             A4:     MOV     AH,AL           ; SAVE INIT PARMS IN AH
E76D  83 C2 03                                  ADD     DX,3            ; POINT TO 8250 CONTROL REGISTER
E770  B0 80                                     MOV     AL,80H
E772  EE                                        OUT     DX,AL           ; SET DLAB=1
                                        ;------ DETERMINE BAUD RATE DIVISOR
E773  8A D4                                     MOV     DL,AH           ; GET PARMS TO DL
E775  B1 04                                     MOV     CL,4
E777  D2 C2                                     ROL     DL,CL
E779  81 E2 000E                                AND     DX,0EH          ; ISOLATE THEM
E77D  BF E729 R                                 MOV     DI,OFFSET A1    ; BASE OF TABLE
E780  03 FA                                     ADD     DI,DX           ; PUT INTO INDEX REGISTER
E782  8B 94 0000 R                              MOV     DX,RS232_BASE[SI] ; POINT TO HIGH ORDER OF DIVISOR
E786  42                                        INC     DX
E787  2E: 8A 45 01                              MOV     AL,CS:[DI]+1    ; GET HIGH ORDER OF DIVISOR
E78B  EE                                        OUT     DX,AL           ; SET MS OF DIV TO 0
E78C  4A                                        DEC     DX
E78D  2E: 8A 05                                 MOV     AL,CS:[DI]      ; GET LOW ORDER OF DIVISOR
E790  EE                                        OUT     DX,AL           ; SET LOW OF DIVISOR
E791  83 C2 03                                  ADD     DX,3
E794  8A C4                                     MOV     AL,AH           ; GET PARMS BACK
E796  24 1F                                     AND     AL,01FH         ; STRIP OFF THE BAUD BITS
E798  EE                                        OUT     DX,AL           ; LINE CONTROL TO 8 BITS
E799  4A                                        DEC     DX
E79A  4A                                        DEC     DX
E79B  B0 00                                     MOV     AL,0
E79D  EE                                        OUT     DX,AL           ; INTERRUPT ENABLES ALL OFF
E79E  EB 53                                     JMP     SHORT A18       ; COM_STATUS
                                        ;------ SEND CHARACTER IN (AL) OVER COMMO LINE
E7A0                                    A5:
E7A0  50                                        PUSH    AX              ; SAVE CHAR TO SEND
E7A1  83 C2 04                                  ADD     DX,4            ; MODEM CONTROL REGISTER
E7A4  B0 03                                     MOV     AL,3            ; DTR AND RTS
E7A6  EE                                        OUT     DX,AL           ; DATA TERMINAL READY, REQUEST TO
                                                                        ; SEND
E7A7  42                                        INC     DX              ; MODEM STATUS REGISTER
E7A8  42                                        INC     DX
E7A9  B7 30                                     MOV     BH,30H          ; DATA SET READY & CLEAR TO SEND
E7AB  E8 E802 R                                 CALL    WAIT_FOR_STATUS ; ARE BOTH TRUE?
E7AE  74 08                                     JE      A9              ; YES, READY TO TRANSMIT CHAR
E7B0  59                                A7:     POP     CX
E7B1  8A C1                                     MOV     AL,CL           ; RELOAD DATA BYTE
E7B3  80 CC 80                          A8:     OR      AH,80H          ; INDICATE TIME OUT
E7B6  EB AC                                     JMP     A3              ; RETURN
E7B8                                    A9:                             ; CLEAR_TO_SEND
E7B8  4A                                        DEC     DX              ; LINE STATUS REGISTER
E7B9  B7 20                                     MOV     BH,20H          ; IS TRANSMITTER READY
E7BB  E8 E802 R                                 CALL    WAIT_FOR_STATUS ; TEST FOR TRANSMITTER READY
E7BE  75 F0                                     JNZ     A7              ; RETURN WITH TIME OUT SET
E7C0  83 EA 05                                  SUB     DX,5            ; DATA PORT
E7C3  59                                        POP     CX              ; RECOVER IN CX TEMPORARILY
E7C4  8A C1                                     MOV     AL,CL           ; MOVE CHAR TO AL FOR OUT, STATUS
                                                                        ; IN AH
E7C6  EE                                        OUT     DX,AL           ; OUTPUT CHARACTER
E7C7  EB 9B                                     JMP     A3              ; RETURN
                                        ;------ RECEIVE CHARACTER FROM COMMO LINE
E7C9  83 C2 04                          A12:    ADD     DX,4            ; MODEM CONTROL REGISTER
E7CC  B0 01                                     MOV     AL,1            ; DATA TERMINAL READY
E7CE  EE                                        OUT     DX,AL
E7CF  42                                        INC     DX              ; MODEM STATUS REGISTER
E7D0  42                                        INC     DX
E7D1  B7 20                                     MOV     BH,20H          ; DATA SET READY
E7D3  E8 E802 R                                 CALL    WAIT_FOR_STATUS ; TEST FOR DSR
E7D6  75 DB                                     JNZ     A8              ; RETURN WITH ERROR
E7D8  4A                                        DEC     DX              ; LINE STATUS REGISTER
E7D9  EC                                A16:    IN      AL,DX
E7DA  A8 01                                     TEST    AL,1            ; RECEIVE BUFFER FULL
E7DC  75 09                                     JNZ     A17             ; TEST FOR REC. BUFF. FULL
E7DE  F6 06 0071 R 80                           TEST    BIOS_BREAK,80H  ; TEST FOR BREAK KEY
E7E3  74 F4                                     JZ      A16             ; LOOP IF NO BREAK KEY
E7E5  EB CC                                     JMP     A8              ; SET TIME OUT ERROR
E7E7  24 1E                             A17:    AND     AL,00011110B    ; TEST FOR ERROR CONDITIONS ON RECV
                                                                        ; CHAR
E7E9  8A E0                                     MOV     AH,AL
E7EB  8B 94 0000 R                              MOV     DX,RS232_BASE[SI] ; DATA PORT
E7EF  EC                                        IN      AL,DX           ; GET CHARACTER FROM LINE
E7F0  E9 E764 R                                 JMP     A3              ; RETURN
                                        ;------ COMMO PORT STATUS ROUTINE
E7F3  8B 94 0000 R                      A18:    MOV     DX,RS232_BASE[SI]
E7F7  83 C2 05                                  ADD     DX,5            ; CONTROL PORT
E7FA  EC                                        IN      AL,DX           ; GET LINE CONTROL STATUS
E7FB  8A E0                                     MOV     AH,AL           ; PUT IN AH FOR RETURN
E7FD  42                                        INC     DX              ; POINT TO MODEM STATUS REGISTER
E7FE  EC                                        IN      AL,DX           ; GET MODEM CONTROL STATUS
E7FF  E9 E764 R                                 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
                                        ;--------------------------------------------
```

```
E802
E802  8A 9D 007C R
E806  2B C9
E808  EC
E809  8A E0
E80B  22 C7
E80D  3A C7
E80F  74 08
E811  E2 F5
E813  FE CB
E815  75 EF
E817  0A FF
E819
E819  C3
E81A
E81A


E81A
E81A  B0 40
E81C  E6 43
E81E  50
E81F  58
E820  E4 41
E822  8A E0
E824  50
E825  58
E826  E4 41
E828  86 C4
E82A  C3
E82B
E82E
E82E  E9 13DD R
```

```
WAIT_FOR_STATUS PROC    NEAR
        MOV     BL,RS232_TIM_OUT[DI] ;LOAD OUTER LOOP COUNT
WFS0:   SUB     CX,CX
WFS1:   IN      AL,DX           ;GET STATUS
        MOV     AH,AL           ; MOVE TO AH
        AND     AL,BH           ;ISOLATE BITS TO TEST
        CMP     AL,BH           ;EXACTLY = TO MASK
        JE      WFS_END         ;RETURN WITH ZERO FLAG ON
        LOOP    WFS1            ;TRY AGAIN
        DEC     BL
        JNZ     WFS0
        OR      BH,BH           ;SET ZERO FLAG OFF
WFS_END:
        RET
WAIT_FOR_STATUS ENDP
RS232_IO        ENDP
;--------------------------------------------------------------------------
;THIS ROUTINE WILL READ TIMER1.  THE VALUE READ IS RETURNED IN AX.
;--------------------------------------------------------------------------
READ_TIME PROC NEAR
        MOV     AL,40H          ; LATCH TIMER1
        OUT     TIM_CTL,AL
        PUSH    AX              ; WAIT FOR 8253 TO INIT ITSELF
        POP     AX
        IN      AL,TIMER+1      ; READ LSB
        MOV     AH,AL           ; SAVE IT IN HIGH BYTE
        PUSH    AX              ; WAIT FOR 8253 TO INIT ITSELF
        POP     AX
        IN      AL,TIMER+1      ; READ MSB
        XCHG    AL,AH           ; PUT BYTES IN PROPER ORDER
        RET
READ_TIME       ENDP
        ORG     0E82EH
        JMP     NEAR PTR KEYBOARD_IO
;--------------------------------------------------------------------------
;ASYNCHRONOUS COMMUNICATIONS ADAPTER POWER ON DIAGNOSTIC TEST
;DESCRIPTION:
;  THIS SUBROUTINE PERFORMS A THOROUGH CHECK OUT OF AN INS8250 LSI
;  CHIP.
;  THE TEST INCLUDES:
;  1)  INITIALIZATION OF THE CHIP TO ASSUME ITS MASTER RESET STATE.
;  2)  READING REGISTERS FOR KNOWN PERMANENT ZERO BITS.
;  3)  TESTING THE INS8250 INTERRUPT SYSTEM AND THAT THE 8250
;      INTERRUPTS TRIGGER AN 8259 (INTERRUPT CONTROLLER) INTERRUPT.
;  4)  PERFORMING THE LOOP BACK TEST:
;      A)  TESTING WHAT WAS WRITTEN/READ AND THAT THE TRANSMITTER
;          HOLDING REG EMPTY BIT AND THE RECEIVER INTERRUPT WORK
;          PROPERLY.
;      B)  TESTING IF CERTAIN BITS OF THE DATA SET CONTROL REGISTER
;          ARE 'LOOPED BACK' TO THOSE IN THE DATA SET STATUS
;          REGISTER.
;      C)  TESTING THAT THE TRANSMITTER IS IDLE WHEN TRANSMISSION
;          TEST IS FINISHED.
; THIS SUBROUTINE EXPECTS TO HAVE THE FOLLOWING PARAMETER PASSED:
;   (DX)= ADDRESS OF THE INS8250 CARD TO TEST.
;    NOTE:  THE ASSUMPTION HAS BEEN MADE THAT THE MODEM ADAPTER IS
;           ----  LOCATED AT 03F8H; THE SERIAL PRINTER AT 02F8H.
; IT RETURNS:
;  (CF) = 1 IF ANY PORTION OF THE TEST FAILED
;       = 0 IF TEST PASSED
;  (BX) = FAILURE KEY FOR ERROR MESSAGE (ONLY VALID IF TEST FAILED)
;     (BH) = 23H  SERIAL PRINTER ADAPTER TEST FAILURE
;          = 24H  MODEM ADAPTER TEST FAILURE
;     (BL) = 2    PERMANENT ZERO BITS IN INTERRUPT ENABLE REGISTER
;                 WERE INCORRECT
;            3    PERMANENT ZERO BITS IN INTERRUPT IDENTIFICATION
;                 REGISTER WERE INCORRECT
;            4    PERMANENT ZERO BITS IN DATA SET CONTROL REGISTER
;                 WERE INCORRECT
;            5    PERMANENT ZERO BITS IN THE LINE STATUS REGISTER
;                 WERE INCORRECT
;            6    RECEIVED DATA AVAILABLE INTERRUPT TEST FAILED
;                 (THE INTERRUPT WAS NOT GENERATED)
;           16H   RECEIVED DATA AVAILABLE INTERRUPT FAILED TO CLEAR
;            7    RESERVED FOR REPORTING THE TRANSMITTER HOLDING
;                 REGISTER EMPTY INTERRUPT TEST FAILED
;                 (NOT USED AT THIS TIME BECAUSE OF THE DIFFERENCES
;                 BETWEEN THE 8250'S WHICH WILL BE USED)
;          17H   TRANSMITTER HOLDING REG EMPTY INTR FAILED TO CLEAR
;           8-B  RECEIVER LINE STATUS INTERRUPT TEST FAILED
;                (THE INTERRUPT WAS NOT GENERATED)
;                8 - OVERRUN ERROR
;                9 - PARITY ERROR
;                A - FRAMING ERROR
;                B - BREAK INTERRUPT ERROR
;          18-1B RECEIVER LINE STATUS INTERRUPT FAILED TO CLEAR
;           C-F  MODEM STATUS INTERRUPT TEST FAILED
;                (THE INTERRUPT WAS NOT GENERATED)
;                C - DELTA CLEAR TO SEND ERROR
;                D - DELTA DATA SET READY ERROR
;                E - TRAILING EDGE RING INDICATOR ERROR
;                F - DELTA RECEIVE LINE SIGNAL DETECT ERROR
;
```

```
                        ;           1C-1F MODEM STATUS INTERRUPT FAILED TO CLEAR
                        ;            10H  AN 8250 INTERRUPT OCCURRED AS EXPECTED, BUT NO
                        ;                 8259 (INTR CONTROLLER) INTERRUPT WAS GENERATED
                        ;            11H  DURING THE TRANSMISSION TEST, THE TRANSMITTER
                        ;                 HOLDING REGISTER WAS NOT EMPTY WHEN IT SHOULD
                        ;                 HAVE BEEN.
                        ;            12H  DURING THE TRANSMISSION TEST, THE RECEIVED DATA
                        ;                 AVAILABLE INTERRUPT DIDN'T OCCUR.
                        ;            13H  TRANSMISSION ERROR - THE CHARACTER RECEIVED
                        ;                 DURING LOOP MODE WAS NOT THE SAME AS THE ONE
                        ;                 TRANSMITTED
                        ;            14H  DURING TRANSMISSION TEST, THE 4 DATA SET CONTROL
                        ;                 OUTPUTS WERE NOT THE SAME AS THE 4 DATA SET
                        ;                 CONTROL INPUTS.
                        ;            15H  THE TRANSMITTER WAS NOT IDLE AFTER THE TRANS-
                        ;                 MISSION TEST COMPLETED.
                        ;     ON EXIT:
                        ;         - THE MODEM OR SERIAL PRINTER'S 8259 INTERRUPT (WHICHEVER
                        ;           DEVICE WAS TESTED) IS DISABLED.
                        ;         - THE 8250 IS IN THE MASTER RESET STATE.
                        ;     ONLY THE DS REGISTER IS PRESERVED - ALL OTHERS ARE ALTERED.
                        ;-----------------------------------------------------------------
= 0084                  WRAP    EQU     84H             ; LOOP BACK TRANSMISSION TEST
                                                        ; INTERRUPT VECTOR ADDRESS
                                                        ; (IN DIAGNOSTICS)
                                ASSUME  CS:CODE,DS:DATA
E831                    UART    PROC    NEAR
E831  1E                        PUSH    DS
E832  E4 21                     IN      AL,INTA01       ; CURRENT ENABLED INTERRUPTS
E834  50                        PUSH    AX              ; SAVE FOR EXIT
E835  0C 01                     OR      AL,00000001B    ; DISABLE TIMER INTR DURING THIS
                                                        ; TEST
E837  E6 21                     OUT     INTA01,AL
E839  9C                        PUSHF                   ; SAVE CALLER'S FLAGS (SAVE INTR
                                                        ; FLAG)
E83A  52                        PUSH    DX              ; SAVE BASE ADDRESS OF ADAPTER CARD
E83B  E8 138B R                 CALL    DDS             ; SET UP 'DATA' AS DATA SEGMENT
                                                        ; ADDRESS
                        ;-----------------------------------------------------------------
                        ;       INITIALIZE PORTS FOR MASTER RESET STATES AND TEST PERMANENT
                        ;       ZERO DATA BITS FOR CERTAIN PORTS.
                        ;-----------------------------------------------------------------
E83E  E8 0AC4 R                 CALL    I8250
E841  73 03                     JNC     AT1             ; ALL OK
E843  E9 E94B R                 JMP     AT14            ; A PORT'S ZERO BITS WERE NOT ZERO!
                        ;-----------------------------------------------------------------
                        ;       INS8250 INTERRUPT SYSTEM TEST
                        ;       ONLY THE INTERRUPT BEING TESTED WILL BE ENABLED.
                        ;-----------------------------------------------------------------
                        ;       SET DI AND SI FOR CALLS TO 'SUI'
E846  BF 0041 R         AT1:    MOV     DI,OFFSET IMASKS ; BASE ADDRESS OF INTERRUPT MASKS
E849  33 F6                     XOR     SI,SI           ; MODEM INDEX
E84B  80 FE 02                  CMP     DH,2            ; OR SERIAL?
E84E  75 02                     JNE     AT2             ; NO - IT'S MODEM
E850  46                        INC     SI              ; IT'S SERIAL PRINTER
E851  47                        INC     DI              ; SERIAL PRINTER 8259 MASK ADDRESS
                        ;       RECEIVED DATA AVAILABLE INTERRUPT TEST
E852  E8 E6F5 R         AT2:    CALL    SUI             ; SET UP FOR INTERRUPTS
E855  FE C3                     INC     BL              ; ERROR REPORTER (INIT. IN I8250)
E857  42                        INC     DX              ; POINT TO INTERRUPT ENABLE
                                                        ; REGISTER
E858  B0 01                     MOV     AL,1            ; ENABLE RECEIVED DATA AVAILABLE
                                                        ; INTR
E85A  EE                        OUT     DX,AL
E85B  53                        PUSH    BX              ; SAVE ERROR REPORTER
E85C  83 C2 04                  ADD     DX,4            ; POINT TO LINE STATUS REGISTER
E85F  B4 01                     MOV     AH,1            ; SET RECEIVER DATA READY BIT
E861  BB 0400                   MOV     BX,0400H        ; INTR TO CHECK, INTR IDENTIFIER
E864  B9 0003                   MOV     CX,3            ; INTERRUPT ID REG 'INDEX'
E867  E8 0AF8 R                 CALL    ICT             ; PERFORM TEST FOR INTERRUPT
E86A  5B                        POP     BX              ; RESTORE ERROR INDICATOR
E86B  3C FF                     CMP     AL,0FFH         ; INTERRUPT ERROR OCCUR?
E86D  74 36                     JE      AT4             ; YES
E86F  E8 E706 R                 CALL    C5059           ; GENERATE 8259 INTERRUPT?
E872  72 33                     JC      AT5             ; NO
E874  4A                        DEC     DX
E875  4A                        DEC     DX              ; RESET INTR BY READING RECR BUFR
E876  EC                        IN      AL,DX           ; DON'T CARE ABOUT THE CONTENTS!
E877  42                        INC     DX
E878  42                        INC     DX              ; INTR ID REG
E879  E8 E719 R                 CALL    W8250C          ; WAIT FOR INTR TO CLEAR
E87C  73 03                     JNC     AT3             ; OK
E87E  E9 E948 R                 JMP     AT13            ; DIDN'T CLEAR
                        ;-----------------------------------------------------------------
                        ; TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT TEST
                        ;       THIS TEST HAS BEEN MODIFIED BECAUSE THE DIFFERENT 8250'S
                        ;       THAT MAY BE USED IN PRODUCING THIS PRODUCT DO NOT FUNCTION
                        ;       THE SAME DURING THE STANDARD TEST OF THIS INTERRUPT
                        ;       (STANDARD BEING THE SAME METHOD FOR TESTING THE OTHER
                        ;       POSSIBLE 8250 INTERRUPTS).  IT IS STILL VALID FOR TESTING
                        ;       IF AN 8259 INTERRUPT IS GENERATED IN RESPONSE TO THE 8250
                        ;       INTERRUPT AND THAT THE 8250 INTERRUPT CLEARS AS IT SHOULD.
                        ;
                        ;       IF THE TRANSMITTER HOLDING REGISTER EMPTY INTERRUPT IS NOT
                        ;       GENERATED WHEN THAT INTERRUPT IS ENABLED, IT IS NOT TREATED
                        ;       AS AN ERROR.  HOWEVER, IF THE INTERRUPT IS GENERATED, IT
                        ;       MUST GENERATE AN 8259 INTERRUPT AND CLEAR PROPERLY TO PASS
                        ;       THIS TEST.
                        ;-----------------------------------------------------------------
```

```
E881  E8 E6F5 R          AT3:    CALL    SUI             ; SET UP FOR INTERRUPTS
E884  FE C3                      INC     BL              ; BUMP ERROR REPORTER
E886  4A                         DEC     DX              ; POINT TO INTERRUPT ENABLE
                                                         ; REGISTER
E887  B0 02                      MOV     AL,2            ; ENABLE XMITTER HOLDING REG EMPTY
                                                         ; INTR
E889  EE                         OUT     DX,AL
E88A  EB 00                      JMP     $+2             ; I/O DELAY
E88C  42                         INC     DX              ; INTR IDENTIFICATION REG
E88D  2B C9                      SUB     CX,CX
E88F  EC                 AT31:   IN      AL,DX           ; READ IT
E890  3C 02                      CMP     AL,2            ; XMITTER HOLDING REG EMPTY INTR?
E892  74 04                      JE      AT32            ; YES
E894  E2 F9                      LOOP    AT31
E896  EB 11                      JMP     SHORT AT6       ; THE INTR DIDN'T OCCUR - TRY NEXT
                                                         ; TEST
E898                     AT32:                           ; THE INTR DID OCCUR
E898  E8 E706 R                  CALL    C5059           ; GENERATE 8259 INTERRUPT?
E89B  72 0A                      JC      AT5             ; NO
E89D  E8 E719 R                  CALL    W8250C          ; WAIT FOR THE INTERRUPT TO CLEAR
                                                         ; (IT SHOULD ALREADY BE CLEAR
                                                         ; BECAUSE 'ICT' READ THE INTR ID
                                                         ; REG)
E8A0  73 07                      JNC     AT6             ; IT CLEARED
E8A2  E9 E948 R                  JMP     AT13            ; ERROR
E8A5  EB 7E              AT4:    JMP     SHORT AT11      ; AVOID OUT OF RANGE JUMPS
E8A7  EB 7A              AT5:    JMP     SHORT AT10
                         ;--------------------------------------------------------------
                         ; RECEIVER LINE STATUS INTERRUPT TEST
                         ;       THERE ARE 4 BITS WHICH COULD GENERATE THIS INTERRUPT.
                         ;       EACH ONE IS TESTED INDIVIDUALLY.
                         ;       WHEN:   AH      TESTING
                         ;               --      -------
                         ;                2      OVERRUN
                         ;                4      PARITY
                         ;                8      FRAMING
                         ;               10H     BREAK INTR
                         ;--------------------------------------------------------------
E8A9  4A                 AT6:    DEC     DX              ; POINT TO INTERRUPT ENABLE
                                                         ; REGISTER
E8AA  B0 04                      MOV     AL,4            ; ENABLE RECEIVER LINE STATUS INTR
E8AC  EE                         OUT     DX,AL
E8AD  83 C2 04                   ADD     DX,4            ; POINT TO LINE STATUS REGISTER
E8B0  B9 0003                    MOV     CX,3            ; INTR ID REG 'INDEX'
E8B3  BD 0004                    MOV     BP,4            ; LOOP COUNTER
E8B6  B4 02                      MOV     AH,2            ; INITIAL BIT TO BE TESTED
E8B8  E8 E6F5 R          AT7:    CALL    SUI             ; SET UP FOR INTERRUPTS
E8BB  FE C3                      INC     BL              ; BUMP ERROR REPORTER
E8BD  53                         PUSH    BX              ; SAVE IT
E8BE  BB 0601                    MOV     BX,0601H        ; INTR TO CHECK, INTR IDENTIFIER
E8C1  E8 0AF8 R                  CALL    ICT             ; PERFORM TEST FOR INTERRUPT
E8C4  5B                         POP     BX
E8C5  24 1E                      AND     AL,00011110B    ; MASK OUT BITS THAT DON'T MATTER
E8C7  3A C4                      CMP     AL,AH           ; TEST BIT ON?
E8C9  75 5A                      JNE     AT11            ; NO
E8CB  E8 E706 R                  CALL    C5059           ; GENERATE 8259 INTERRUPT?
E8CE  72 53                      JC      AT10            ; NO
E8D0  83 EA 03                   SUB     DX,3            ; INTR ID REG
E8D3  E8 E719 R                  CALL    W8250C          ; WAIT FOR THE INTR TO CLEAR
E8D6  72 70                      JC      AT13            ; IT DIDN'T
E8D8  4D                         DEC     BP              ; ALL FOUR BITS TESTED?
E8D9  74 07                      JE      AT8             ; YES - GO ON TO NEXT TEST
E8DB  D0 E4                      SHL     AH,1            ; GET READY FOR NEXT BIT
E8DD  83 C2 03                   ADD     DX,3            ; LINE STATUS REGISTER
E8E0  EB D6                      JMP     AT7             ; TEST NEXT BIT
                         ;--------------------------------------------------------------
                         ; MODEM STATUS INTERRUPT TEST
                         ;       THERE ARE 4 BITS WHICH COULD GENERATE THIS INTERRUPT.
                         ;       THEY ARE TESTED INDIVIDUALLY.
                         ;       WHEN:   AH      TESTING
                         ;               --      -------
                         ;                1      DELTA CLEAR TO SEND
                         ;                2      DELTA DATA SET READY
                         ;                4      TRAILING EDGE RING INDICATOR
                         ;                8      DELTA RECEIVE LINE SIGNAL DETECT
                         ;--------------------------------------------------------------
E8E2  83 C2 04           AT8:    ADD     DX,4            ; MODEM STATUS REGISTER
E8E5  EC                         IN      AL,DX           ; CLEAR DELTA BITS THAT MAY BE ON
                                                         ; BECAUSE OF DIFFERENCES AMONG
                                                         ; 8250'S.
E8E6  EB 00                      JMP     $+2             ; I/O DELAY
E8E8  83 EA 05                   SUB     DX,5            ; INTERRUPT ENABLE REGISTER
E8EB  B0 08                      MOV     AL,8            ; ENABLE MODEM STATUS INTERRUPT
E8ED  EE                         OUT     DX,AL
E8EE  83 C2 05                   ADD     DX,5            ; POINT TO MODEM STATUS REGISTER
E8F1  B9 0004                    MOV     CX,4            ; INTR ID REG 'INDEX'
E8F4  BD 0004                    MOV     BP,4            ; LOOP COUNTER
E8F7  B4 01                      MOV     AH,1            ; INITIAL BIT TO BE TESTED
E8F9  E8 E6F5 R          AT9:    CALL    SUI             ; SET UP FOR INTERRUPTS
E8FC  FE C3                      INC     BL              ; BUMP ERROR INDICATOR
E8FE  53                         PUSH    BX              ; SAVE IT
E8FF  BB 0001                    MOV     BX,0001H        ; INTR TO CHECK, INTR IDENTIFIER
E902  E8 0AF8 R                  CALL    ICT             ; PERFORM TEST FOR INTERRUPT
E905  5B                         POP     BX
E906  24 0F                      AND     AL,00001111B    ; MASK OUT BITS THAT DON'T MATTER
E908  3A C4                      CMP     AL,AH           ; TEST BIT ON?
E90A  75 19                      JNE     AT11            ; NO
E90C  E8 E706 R                  CALL    C5059           ; GENERATE 8259 INTERRUPT?
E90F  72 12                      JC      AT10            ; NO
E911  83 EA 04                   SUB     DX,4            ; INTR ID REG
```

```
E914  E8 E719 R                 CALL    W8250C          ; WAIT FOR INTERRUPT TO CLEAR
E917  72 2F                     JC      AT13            ; IT DIDN'T
E919  4D                        DEC,    BP
E91A  74 0B                     JE      AT12            ; ALL FOUR BITS TESTED - GO ON
E91C  D0 E4                     SHL     AH,1            ; GET READY FOR NEXT BIT
E91E  83 C2 04                  ADD     DX,4            ; MODEM STATUS REGISTER
E921  EB D6                     JMP     AT9             ; TEST NEXT BIT
                        ;-------------------------------------------------------------------
                        ;       POSSIBLE 8259 INTERRUPT CONTROLLER PROBLEM
                        ;-------------------------------------------------------------------
E923  B3 10             AT10:   MOV     BL,10H          ; SET ERROR REPORTER
E925  EB 24             AT11:   JMP     SHORT AT14
                        ;-------------------------------------------------------------------
                        ;       SET 9600 BAUD RATE AND DEFINE DATA WORD AS HAVING 8
                        ;       BITS/WORD, 2 STOP BITS, AND ODD PARITY.
                        ;-------------------------------------------------------------------
E927  42                AT12:   INC     DX              ; LINE CONTROL REGISTER
E928  E8 F085 R                 CALL    S8250
                        ;-------------------------------------------------------------------
                        ;       SET DATA SET CONTROL WORD TO BE IN LOOP MODE
                        ;-------------------------------------------------------------------
E92B  83 C2 04                  ADD     DX,4
E92E  EC                        IN      AL,DX           ; CURRENT STATE
E92F  EB 00                     JMP     $+2             ; I/O DELAY
E931  0C 10                     OR      AL,00010000B    ; SET BIT 4 OF DATA SET CONTROL REG
E933  EE                        OUT     DX,AL
E934  EB 00                     JMP     $+2             ; I/O DELAY
E936  42                        INC     DX
E937  42                        INC     DX              ; MODEM STATUS REG
E938  EC                        IN      AL,DX           ; CLEAR POSSIBLE MODEM STATUS
                                                        ; INTERRUPT WHICH COULD BE CAUSED
                                                        ; BY THE OUTPUT BITS BEING LOOPED
                                                        ; TO THE INPUT BITS
E939  EB 00                     JMP     $+2             ; I/O DELAY
E93B  83 EA 06                  SUB     DX,6            ; RECEIVER BUFFER
E93E  EC                        IN      AL,DX           ; DUMMY READ TO CLEAR DATA READY
                                                        ; BIT IF IT WENT HIGH ON WRITE TO
                                                        ; MCR
                        ;-------------------------------------------------------------------
                        ;       PERFORM THE LOOP BACK TEST
                        ;-------------------------------------------------------------------
E93F  42                        INC     DX              ; INTR ENBL REG
E940  B0 00                     MOV     AL,0            ; SET FOR INTERNAL WRAP TEST
E942  CD 84                     INT     WRAP            ; DO LOOP BACK TRANSMISSION TEST
E944  B1 00                     MOV     CL,0            ; ASSUME NO ERRORS
E946  73 05                     JNC     AT15            ; WRAP TEST PASSED
E948  80 C3 10          AT13:   ADD     BL,10H          ; ERROR INDICATOR
                        ;-------------------------------------------------------------------
                        ;       AN ERROR WAS ENCOUNTERED SOMEWHERE DURING THE TEST
                        ;-------------------------------------------------------------------
E94B  B1 01             AT14:   MOV     CL,1            ; SET FAIL INDICATOR
                        ;-------------------------------------------------------------------
                        ;       HOUSEKEEPING:  RE-INITIALIZE THE 8250 PORTS (THE LOOP BIT
                        ;       WILL BE RESET), DISABLE THIS DEVICE INTERRUPT, SET UP
                        ;       REGISTER BH IF AN ERROR OCCURRED, AND SET OR RESET THE
                        ;       CARRY FLAG.
                        ;-------------------------------------------------------------------
E94D  5A                AT15:   POP     DX              ; GET BASE ADDRESS OF 8250 ADAPTER
E94E  53                        PUSH    BX              ; SAVE ERROR CODE
E94F  E8 0AC4 R                 CALL    I8250           ; RE-INITIALIZE 8250 PORTS
E952  5B                        POP     BX
E953  2E: 8A 25                 MOV     AH,CS:[DI]      ; GET DEVICE INTERRUPT MASK
E956  20 26 0084 R              AND     INTR_FLAG,AH    ; CLEAR DEVICE'S INTERRUPT FLAG BIT
E95A  80 F4 FF                  XOR     AH,0FFH         ; FLIP BITS
E95D  E4 21                     IN      AL,INTA01       ; GET CURRENT INTERRUPT PORT
E95F  0A C4                     OR      AL,AH           ; DISABLE THIS DEVICE INTERRUPT
E961  E6 21                     OUT     INTA01,AL
E963  9D                        POPF                    ; RE-ESTABLISH CALLER'S INTERRUPT
                                                        ; FLAG
E964  0A C9                     OR      CL,CL           ; ANY ERRORS?
E966  74 0C                     JE      AT17            ; NO
E968  B7 24                     MOV     BH,24H          ; ASSUME MODEM ERROR
E96A  80 FE 02                  CMP     DH,2            ; OR IS IT SERIAL?
E96D  75 02                     JNE     AT16            ; IT'S MODEM
E96F  B7 23                     MOV     BH,23H          ; IT'S SERIAL PRINTER
E971  F9                AT16:   STC                     ; SET CARRY FLAG TO INDICATE ERROR
E972  EB 01                     JMP     SHORT AT18
E974  F8                AT17:   CLC                     ; RESET CARRY FLAG - NO ERRORS
E975  58                AT18:   POP     AX              ; RESTORE ENTRY ENABLED INTERRUPTS
E976  E6 21                     OUT     INTA01,AL       ; DEVICE INTRS RE-ESTABLISHED
E978  1F                        POP     DS              ; RESTORE REGISTER
E979  C3                        RET
E97A                    UART    ENDP
E987                            ORG     0E987H
E987  E9 1561 R                 JMP     NEAR PTR KB_INT
                        ;-----------------------------------------------
                        ; 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
                        ;-----------------------------------------------
```

```
E98A                    NEC_OUTPUT      PROC    NEAR
E98A  52                        PUSH    DX              ; SAVE REGISTERS
E98B  51                        PUSH    CX
E98C  BA 00F4                   MOV     DX,NEC_STAT     ; STATUS PORT
E98F  33 C9                     XOR     CX,CX           ; COUNT FOR TIME OUT
E991  EC                J23:    IN      AL,DX           ; GET STATUS
E992  A8 40                     TEST    AL,DIO          ; TEST DIRECTION BIT
E994  74 0C                     JZ      J25             ; DIRECTION OK
E996  E2 F9                     LOOP    J23
E998                    J24:                            ; TIME_ERROR
E998  80 0E 0041 R 80           OR      DISKETTE_STATUS,TIME_OUT
E99D  59                        POP     CX
E99E  5A                        POP     DX              ; SET ERROR CODE AND RESTORE REGS
E99F  58                        POP     AX              ; DISCARD THE RETURN ADDRESS
E9A0  F9                        STC                     ; INDICATE ERROR TO CALLER
E9A1  C3                        RET                     ;
E9A2  33 C9             J25:    XOR     CX,CX           ; RESET THE COUNT
E9A4  EC                J26:    IN      AL,DX           ; GET THE STATUS
E9A5  A8 80                     TEST    AL,RQM          ; IS IT READY?
E9A7  75 04                     JNZ     J27             ; YES, GO OUTPUT
E9A9  E2 F9                     LOOP    J26             ; COUNT DOWN AND TRY AGAIN
E9AB  EB EB                     JMP     J24             ; ERROR CONDITION
E9AD                    J27:                            ; OUTPUT
E9AD  8A C4                     MOV     AL,AH           ; GET BYTE TO OUTPUT
E9AF  42                        INC     DX              ; DATA PORT IS 1 GREATER THAN
                                                        ; STATUS PORT
E9B0  EE                        OUT     DX,AL           ; OUTPUT THE BYTE
E9B1  59                        POP     CX              ; RECOVER REGISTERS
E9B2  5A                        POP     DX
E9B3  C3                        RET                     ; CY = 0 FROM TEST INSTRUCTION
E9B4                    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 --
                        ;   BL = INDEX OF BYTE TO BE FETCHED # 2
                        ;        IF THE LOW BIT OF BL IS ON, THE BYTE IS IMMEDIATELY
                        ;        OUTPUT TO THE NEC CONTROLLER
                        ; EXIT --
                        ;   AH = THAT BYTE FROM BLOCK
                        ;   BX = DESTROYED
                        ;--------------------------------------------
E9B4                    GET_PARM        PROC    NEAR
E9B4  1E                        PUSH    DS              ; SAVE SEGMENT
E9B5  56                        PUSH    SI              ; SAVE REGISTER
E9B6  2B C0                     SUB     AX,AX           ; ZERO TO AX
E9B8  32 FF                     XOR     BH,BH           ; ZERO BH
E9BA  8E D8                     MOV     DS,AX
                                ASSUME  DS:ABS0
E9BC  C5 36 0078 R              LDS     SI,DISK_POINTER ; POINT TO BLOCK
E9C0  D1 EB                     SHR     BX,1            ; DIVIDE BX BY 2, AND SET FLAG FOR
                                                        ; EXIT
E9C2  9C                        PUSHF                   ; SAVE OUTPUT BIT
E9C3  8A 20                     MOV     AH,[SI+BX]      ; GET THE BYTE
E9C5  83 FB 01                  CMP     BX,1            ; IS THIS THE PARM WITH DMA
                                                        ; INDICATOR
E9C8  75 05                     JNZ     J27_1
E9CA  80 CC 01                  OR      AH,1            ; TURN ON NO DMA BIT
E9CD  EB 0C                     JMP     SHORT J27_2
E9CF  83 FB 0A          J27_1:  CMP     BX,10           ; MOTOR STARTUP DELAY?
E9D2  75 07                     JNE     J27_2
E9D4  80 FC 04                  CMP     AH,4            ; GREATER THAN OR EQUAL TO 1/2 SEC?
E9D7  7D 02                     JGE     J27_2           ; YES, OKAY
E9D9  B4 04                     MOV     AH,4            ; NO, FORCE 1/2 SECOND DELAY
E9DB  9D                J27_2:  POPF                    ; GET OUTPUT BIT
E9DC  5E                        POP     SI              ; RESTORE REGISTER
E9DD  1F                        POP     DS              ; RESTORE SEGMENT
                                ASSUME  DS:DATA
E9DE  72 AA                     JC      NEC_OUTPUT      ; IF FLAG SET, OUTPUT TO CONTROLLER
E9E0  C3                        RET                     ; RETURN TO CALLER
E9E1                    GET_PARM        ENDP
                        ;--------------------------------------------------
                        ; BOUND_SETUP
                        ;       THIS ROUTINE SETS UP BUFFER ADDRESSING FOR READ/WRITE/VERIFY
                        ;       OPERATIONS.
                        ; INPUT
                        ;       ES HAS ORIGINAL BUFFER SEGMENT VALUE
                        ;       BP POINTS AT BASE OF SAVED PARMETERS ON STACK
                        ; OUTPUT
                        ;       ES HAS SEGMENT WHICH WILL ALLOW 64K ACCESS.  THE
                        ;       COMBINATION ES:DI  AND DS:SI POINT TO THE BUFFER.  THIS
                        ;       CALCULATED ADDRESS WILL ALWAYS ACCESS 64K OF MEMORY.
                        ;       BX DESTOYED
                        ;--------------------------------------------------
```

```
E9E1                    BOUND_SETUP     PROC    NEAR
E9E1  51                        PUSH    CX                      ; SAVE REGISTERS
E9E2  8B 5E 0C                  MOV     BX,[BP+12]              ; GET OFFSET OF BUFFER FROM STACK
E9E5  53                        PUSH    BX                      ; SAVE OFFSET TEMPORARILY
E9E6  B1 04                     MOV     CL,4                    ; SHIFT COUNT
E9E8  D3 EB                     SHR     BX,CL                   ; SHIFT OFFSET FOR NEW SEGMENT
                                                                ; VALUE
E9EA  8C C1                     MOV     CX,ES                   ; PUT ES IN REGISTER SUITABLE FOR
                                                                ; ADDING TO
E9EC  03 CB                     ADD     CX,BX                   ; GET NEW VALUE FOR ES
E9EE  8E C1                     MOV     ES,CX                   ; UPDATE THE ES REGISTER
E9F0  5B                        POP     BX                      ; RECOVER ORIGINAL OFFSET
E9F1  81 E3 000F                AND     BX,0000FH               ; NEW OFFSET
E9F5  8B F3                     MOV     SI,BX                   ; DS:SI POINT AT BUFFER
E9F7  8B FB                     MOV     DI,BX                   ; ES:DI POINT AT BUFFER
E9F9  59                        POP     CX
E9FA  C3                        RET
E9FB                    BOUND_SETUP     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
                        ;--------------------------------------------------------
E9FB                    SEEK    PROC    NEAR
E9FB  56                        PUSH    SI                      ; SAVE REGISTER
E9FC  53                        PUSH    BX                      ; SAVE REGISTER
E9FD  51                        PUSH    CX
E9FE  BE 0074 R                 MOV     SI,OFFSET TRACK0 ; BASE OF CURRENT HEAD POSITIONS
EA01  B0 01                     MOV     AL,1                    ; ESTABLISH MASK FOR RECAL
EA03  8A CA                     MOV     CL,DL                   ; USE DRIVE AS A SHIFT COUNT
EA05  81 E1 00FF                AND     CX,0FFH                 ; MASK OFF HIGH BYTE
EA09  03 F1                     ADD     SI,CX                   ; POINT SI AT CORRECT DRIVE
EA0B  D2 C0                     ROL     AL,CL                   ; GET MASK FOR DRIVE
                        ;------ SI CONTAINS OFFSET FOR CORRECT DRIVE, AL CONTAINS BIT MASK
                        ;       IN POSITION 0,1 OR 2
EA0D  59                        POP     CX                      ; RESTORE PARAMETER REGISTER
EA0E  BB EA66 R                 MOV     BX,OFFSET J32   ; SET UP ERROR RECOVERY ADDRESS
EA11  53                        PUSH    BX                      ; NEEDED FOR ROUTINE NEC_OUTPUT
EA12  84 06 003E R              TEST    SEEK_STATUS,AL  ; TEST DRIVE FOR RECAL
EA16  75 1B                     JNZ     J28                     ; NO_RECAL
EA18  08 06 003E R              OR      SEEK_STATUS,AL  ; TURN ON THE NO RECAL BIT IN FLAG
EA1C  80 3C 00                  CMP     BYTE PTR[SI],0  ; LAST REFERENCED TRACK=0?
EA1F  74 12                     JZ      J28                     ; YES IGNORE RECAL
EA21  B4 07                     MOV     AH,07H                  ; RECALIBRATE COMMAND
EA23  E8 E98A R                 CALL    NEC_OUTPUT
EA26  8A E2                     MOV     AH,DL                   ; RECAL REQUIRED ON DRIVE IN DL
EA28  E8 E98A R                 CALL    NEC_OUTPUT              ; OUTPUT THE DRIVE NUMBER
                        ;------ HEAD IS MOVING TO CORRECT TRACK
EA2B  E8 EA6F R                 CALL    CHK_STAT_2              ; GET THE STATUS OF RECALIBRATE
EA2E  72 39                     JC      J32_2                   ; SEEK_ERROR
EA30  C6 04 00                  MOV     BYTE PTR[SI],0
                        ;----- DRIVE IS IN SYNCH WITH CONTROLLER, SEEK TO TRACK
EA33  8A 04             J28:    MOV     AL,BYTE PTR[SI] ; GET THE PCN
EA35  2A C5                     SUB     AL,CH                   ; GET SEEK_WAIT VALUE
EA37  74 2C                     JZ      J31_1                   ; ALREADY ON CORRECT TRACK
EA39  B4 0F                     MOV     AH,0FH                  ; SEEK COMMAND TO NEC
EA3B  E8 E98A R                 CALL    NEC_OUTPUT
EA3E  8A E2                     MOV     AH,DL                   ; DRIVE NUMBER
EA40  E8 E98A R                 CALL    NEC_OUTPUT
EA43  8A E5                     MOV     AH,CH                   ; TRACK NUMBER
EA45  E8 E98A R                 CALL    NEC_OUTPUT
EA48  E8 EA6F R                 CALL    CHK_STAT_2              ; GET ENDING INTERRUPT AND SENSE
                                                                ; STATUS
                        ;----- WAIT FOR HEAD SETTLE
EA4B  9C                        PUSHF                           ; SAVE STATUS FLAGS
EA4C  51                        PUSH    CX                      ; SAVE REGISTER
EA4D  B3 12                     MOV     BL,18                   ; HEAD SETTLE PARAMETER
EA4F  E8 E9B4 R                 CALL    GET_PARM
EA52                    J29:                                    ; HEAD_SETTLE
EA52  B9 0226                   MOV     CX,550                  ; 1 MS LOOP
EA55  0A E4                     OR      AH,AH                   ; TEST FOR TIME EXPIRED
EA57  74 06                     JZ      J31
EA59  E2 FE             J30:    LOOP    J30                     ; DELAY FOR 1 MS
EA5B  FE CC                     DEC     AH                      ; DECREMENT THE COUNT
EA5D  EB F3                     JMP     J29                     ; DO IT SOME MORE
EA5F  59                J31:    POP     CX                      ; RESTORE REGISTER
EA60  9D                        POPF
EA61  72 06                     JC      J32_2
EA63  88 2C                     MOV     BYTE PTR[SI],CH
EA65  5B                J31_1:  POP     BX                      ; GET RID OF DUMMY RETURN
EA66                    J32:                                    ; SEEK_ERROR
EA66  5B                        POP     BX                      ; RESTORE REGISTER
EA67  5E                        POP     SI                      ; UPDATE CORRECT
EA68  C3                        RET                             ; RETURN TO CALLER
EA69  C6 04 FF          J32_2:  MOV     BYTE PTR[SI],0FFH ; UNKNOWN STATUS ABOUT SEEK
                                                                ; OPERATION
EA6C  5B                        POP     BX                      ; GET RID OF DUMMY RETURN
EA6D  EB F7                     JMP     SHORT J32
EA6F                    SEEK    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
                        ;--------------------------------------------------
EA6F                    CHK_STAT_2      PROC    NEAR
EA6F  53                        PUSH    BX                      ; SAVE REGISTERS
EA70  56                        PUSH    SI
EA71  33 DB                     XOR     BX,BX                   ; NUMBER OF SENSE INTERRUPTS TO
                                                                ; ISSUE
EA73  BE EA88 R                 MOV     SI,OFFSET J33_3 ; SET UP DUMMY RETURN FROM
                                                                ; NEC_OUTPUT
EA76  56                        PUSH    SI                      ; PUT ON STACK
EA77  B4 08             J33_2:  MOV     AH,08H          ; SENSE INTERUPT STATUS
EA79  E8 E98A R                 CALL    NEC_OUTPUT      ; ISSUE SENSE INTERUPT STATUS
EA7C  E8 EAA0 R                 CALL    RESULTS         ;
EA7F  72 10                     JC      J35             ; NEC TIME OUT, FLAGS SET IN
                                                                ; RESULTS
EA81  A0 0042 R                 MOV     AL,NEC_STATUS   ; GET STATUS
EA84  A8 20                     TEST    AL,SEEK_END     ; IS SEEK OR RECAL OPERATION DONE?
EA86  75 0D                     JNZ     J35_1           ; JUMP IF EXECUTION OF SEEK OR
                                                                ; RECAL DONE
EA88  4B                J33_3: DEC     BX              ; DEC LOOP COUNTER
EA89  75 EC                     JNZ     J33_2           ; DO ANOTHER LOOP
EA8B  80 0E 0041 R 80           OR      DISKETTE_STATUS,TIME_OUT
EA90  F9                J34:   STC                     ; RETURN ERROR INDICATION FOR
                                                                ; CALLER
EA91  5E                J35:   POP     SI              ; RESTORE REGISTERS
EA92  5E                       POP     SI
EA93  5B                       POP     BX
EA94  C3                       RET
                        ;-----SEEK END HAS OCCURED,  CHECK FOR NORMAL TERMINATION
EA95  24 C0             J35_1: AND     AL,0C0H         ; MASK NORMAL TERMINATION BITS
EA97  74 F8                    JZ      J35             ; JUMP IF NORMAL TERMINATION
EA99  80 0E 0041 R 40          OR      DISKETTE_STATUS,BAD_SEEK
EA9E  EB F0                    JMP     J34
EAA0                    CHK_STAT_2      ENDP
                        ;--------------------------------------------------
                        ; RESULTS
                        ;       THIS ROUTINE WILL READ ANYTHING THAT THE NEC CONTROLLER
                        ;       HAS TO SAY FOLLOWING AN INTERRUPT.
                        ;       IT IS ASSUMED THAT THE NEC DATA PORT = NEC STATUS PORT + 1.
                        ; 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
                        ;--------------------------------------------------
EAA0                    RESULTS PROC    NEAR
EAA0  FC                        CLD
EAA1  BF 0042 R                 MOV     DI,OFFSET NEC_STATUS ; POINTER TO DATA AREA
EAA4  51                        PUSH    CX              ; SAVE COUNTER
EAA5  52                        PUSH    DX
EAA6  53                        PUSH    BX
EAA7  B3 07                     MOV     BL,7            ; MAX STATUS BYTES
                        ;------ WAIT FOR REQUEST FOR MASTER
EAA9                    J38:                            ; INPUT_LOOP
EAA9  33 C9                     XOR     CX,CX           ; COUNTER
EAAB  BA 00F4                   MOV     DX,NEC_STAT     ; STATUS PORT
EAAE                    J39:                            ; WAIT FOR MASTER
EAAE  EC                        IN      AL,DX           ; GET STATUS
EAAF  A8 80                     TEST    AL,080H         ; MASTER READY
EAB1  75 0C                     JNZ     J40A            ; TEST_DIR
EAB3  E2 F9                     LOOP    J39             ; WAIT_MASTER
EAB5  80 0E 0041 R 80           OR      DISKETTE_STATUS,TIME_OUT
EABA                    J40:                            ; RESULTS_ERROR
EABA  F9                        STC                     ; SET ERROR RETURN
                        ;------ RESULT OPERATION IS DONE
EABB  5B                J44:   POP     BX
EABC  5A                       POP     DX
EABD  59                       POP     CX
EABE  C3                       RET
                        ;------ TEST THE DIRECTION BIT
EABF  EC                J40A:  IN      AL,DX           ; GET STATUS REG AGAIN
EAC0  A8 40                    TEST    AL,040H         ; TEST DIRECTION BIT
EAC2  75 07                    JNZ     J42             ; OK TO READ STATUS
EAC4                    J41:                            ; NEC_FAIL
EAC4  80 0E 0041 R 20          OR      DISKETTE_STATUS,BAD_NEC
EAC9  EB EF                    JMP     J40             ; RESULTS_ERROR
                        ;------ READ IN THE STATUS
EACB                    J42:                            ; INPUT_STAT
EACB  42                       INC     DX              ; POINT AT DATA PORT
EACC  EC                       IN      AL,DX           ; GET THE DATA
EACD  88 05                    MOV     [DI],AL         ; STORE THE BYTE
EACF  47                       INC     DI              ; INCREMENT THE POINTER
EAD0  B9 000A                  MOV     CX,10           ; LOOP TO KILL TIME FOR NEC
EAD3  E2 FE             J43:   LOOP    J43
EAD5  4A                       DEC     DX              ; POINT AT STATUS PORT
EAD6  EC                       IN      AL,DX           ; GET STATUS
EAD7  A8 10                    TEST    AL,010H         ; TEST FOR NEC STILL BUSY
EAD9  74 E0                    JZ      J44             ; RESULTS DONE
EADB  FE CB                    DEC     BL              ; DECREMENT THE STATUS COUNTER
EADD  75 CA                    JNZ     J38             ; GO BACK FOR MORE
EADF  EB E3                    JMP     J41             ; CHIP HAS FAILED
```

```
                        ;--------------------------------------------------
                        ; 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
                        ;--------------------------------------------------
EAE1                    NUM_TRANS       PROC    NEAR
EAE1  A0 0045 R                 MOV     AL,NEC_STATUS+3 ; GET CYLINDER ENDED UP ON
EAE4  3A 46 0B                  CMP     AL,[BP+11]      ; SAME AS WE STARTED
EAE7  A0 0047 R                 MOV     AL,NEC_STATUS+5 ; GET ENDING SECTOR
EAEA  74 07                     JZ      J45             ; IF ON SAME CYL, THEN NO ADJUST
EAEC  B3 08                     MOV     BL,8
EAEE  E8 E9B4 R                 CALL    GET_PARM        ; GET EOT VALUE
EAF1  8A C4                     MOV     AL,AH           ;  INTO AL
EAF3  FE C0             J45:    INC     AL              ; USE EOT+1 FOR CALCULATION
EAF5  2A 46 0A                  SUB     AL,[BP]+10      ; SUBTRACT START FROM END
EAF8  88 46 0E                  MOV     [BP+14],AL
EAFB  C3                        RET
EAFC                    NUM_TRANS       ENDP
EAFC                    RESULTS ENDP
                        ;--------------------------------------------------
                        ; DISABLE
                        ;       THIS ROUTINE WILL DISABLE ALL INTERRUPTS EXCEPT FOR
                        ;       INTERRUPT 6 SO WATCH DOG TIME OUT CAN OCCUR IN ERROR
                        ;       CONDITIONS.
                        ; INPUT
                        ;       NONE
                        ; OUTPUT
                        ;       NONE
                        ;       ALL REGISTERS REMAIN INTACT
                        ;--------------------------------------------------
EAFC                    DISABLE         PROC    NEAR
EAFC  50                        PUSH    AX
                        ;------ DISABLE ALL INTERRUPTS AT THE 8259 LEVEL EXCEPT DISKETTE
EAFD  E4 21                     IN      AL,INTA01       ; READ CURRENT MASK
EAFF  89 46 10                  MOV     [BP+16],AX      ; SAVE MASK ON THE SPACE ALLOCATED
                                                        ; ON THE STACK
EB02  B0 BF                     MOV     AL,0BFH         ; MASK OFF ALL INTERRUPTS EXCEPT
                                                        ; DISKETTE
EB04  E6 21                     OUT     INTA01,AL       ; OUTPUT MASK TO THE 8259
EB06  E8 E9E1 R                 CALL    BOUND_SETUP     ; SETUP REGISTERS TO ACCESS BUFFER
EB09  58                        POP     AX
EB0A  C3                        RET
EB0B                    DISABLE         ENDP
                        ;--------------------------------------------------
                        ; ENABLE
                        ;       THIS PROC ENABLES ALL INTERRUPTS.  IT ALSO SETS THE 8253 TO
                        ;       THE MODE REQUIRED FOR KEYBOARD DATA DESERIALIZATION.
                        ;       BEFORE THE LATCH FOR KEYBOARD DATA IS RESET, BIT 0 OF THE
                        ;       8255 IS READ TO DETERMINE WHETHER ANY KEYSTROKES OCCURED
                        ;       WHILE THE SYSTEM WAS MASKED OFF.
                        ; INPUT
                        ;       NONE
                        ; OUTPUT
                        ;       AL=1 MEANS A KEY WAS STRUCK DURING DISKETTE I/O.  (OR NOISE
                        ;            ON THE LINE)
                        ;       AL=0 MEANS THAT NO KEY WAS PRESSED.
                        ;       AX IS DESTROYED.  ALL OTHER REGISTERS REMAIN INTACT.
                        ;--------------------------------------------------
EB0B                    ENABLE          PROC    NEAR
EB0B  52                        PUSH    DX              ; SAVE DX
                        ;------ RETURN TIMER1 TO STATE NEEDED FOR KEYBOARD I/O
EB0C  B0 76                     MOV     AL,01110110B    ;
EB0E  E6 43                     OUT     TIM_CTL,AL
EB10  50                        PUSH    AX
EB11  58                        POP     AX              ; WAIT FOR 8253 TO INITIALIZE
                                                        ; ITSELF
EB12  B0 FF                     MOV     AL,0FFH         ; INITIAL VALUE FOR 8253
EB14  E6 41                     OUT     TIMER+1,AL      ; LSB
EB16  50                        PUSH    AX
EB17  58                        POP     AX              ; WAIT
EB18  E6 41                     OUT     TIMER+1,AL      ; MSB
                        ;------ CHECK IF ANY KEYSTROKES OCCURED DURING DISKETTE TRANSFER
EB1A  8E 46 10                  MOV     ES,[BP+16]      ; GET ORIGINAL ES VALUE FROM THE
                                                        ; STACK
EB1D  E4 62                     IN      AL,62H          ; READ PORT C OF 8255
EB1F  24 01                     AND     AL,01H          ; BIT=1 MEANS KESTROKE HAS OCCURED
EB21  50                        PUSH    AX              ; SAVE IT ON THE STACK
                        ;------ ENABLE NMI INTERRUPTS
EB22  E4 A0                     IN      AL,NMI_PORT     ; RESET LATCH
EB24  B0 80                     MOV     AL,80H          ; MASK TO ENABLE NMI
EB26  E6 A0                     OUT     NMI_PORT,AL     ; ENABLE NMI
                        ;-----  ENABLE ALL INTERRUPTS WHICH WERE ENABLED BEFORE TRANSFER
EB28  8B 46 10                  MOV     AX,[BP+16]      ; GET MASK FROM THE STACK
EB2B  E6 21                     OUT     INTA01,AL
EB2D  58                        POP     AX              ; PASS BACK KEY STROKE FLAG
EB2E  5A                        POP     DX
EB2F  FB                        STI
EB30  C3                        RET
EB31                    ENABLE          ENDP
```

```
                        ;--------------------------------------------
                        ;CLOCK_WAIT
                        ;       THIS PROCEDURE IS CALLED WHEN THE TIME OF DAY
                        ;       IS BEING UPDATED.  IT WAITS IF TIMER0 IS ALMOST
                        ;       READY TO WRAP UNTIL IT IS SAFE TO READ AN ACCURATE
                        ;       TIMER1.
                        ;INPUT
                        ;       NONE.
                        ;OUTPUT
                        ;       NONE. AX IS DESTROYED.
                        ;--------------------------------------------
EB31                    CLOCK_WAIT      PROC    NEAR
EB31  32 C0                     XOR     AL,AL           ; READ MODE TIMER0 FOR 8253
EB33  E6 43                     OUT     TIM_CTL,AL      ; OUTPUT TO THE 8253
EB35  50                        PUSH    AX
EB36  58                        POP     AX              ; WAIT FOR 8253 TO INITIALIZE
                                                        ; ITSELF
EB37  E4 40                     IN      AL,TIMER0       ; READ LEAST SIGNIFICANT BYTE
EB39  86 C4                     XCHG    AL,AH           ; SAVE IT
EB3B  E4 40                     IN      AL,TIMER0       ; READ MOST SIGNIFICANT BYTE
EB3D  86 C4                     XCHG    AL,AH           ; REARRANGE FOR PROPER ORDER
EB3F  3D 012C                   CMP     AX,THRESHOLD    ; IS TIMER0 CLOSE TO WRAPPING?
EB42  72 ED                     JC      CLOCK_WAIT      ; JUMP IF CLOCK IS WITHIN THRESHOLD
EB44  C3                        RET                     ; OK TO READ TIMER1
EB45                    CLOCK_WAIT      ENDP
                        ;--------------------------------------------------------------------
                        ;GET_DRIVE
                        ;       THIS ROUTINE WILL CALCULATE A BIT MASK FOR THE DRIVE WHICH
                        ;       IS SELECTED BY THE CURRENT INT 13 CALL.  THE DRIVE SELECTED
                        ;       CORRESPONDS TO THE BIT IN THE MASK,  I.E. DRIVE ZERO
                        ;       CORRESPONDS TO BIT ZERO AND A 01H IS RETURNED. THE BIT IS
                        ;       CALCULATED BY ACCESSING THE PARAMETERS PASSED TO INT 13
                        ;       WHICH WERE SAVED ON THE STACK.
                        ;INPUT
                        ;       BYTE PTR[BP] MUST POINT TO DRIVE FOR SELECTION.
                        ;OUPTUT
                        ;       AL CONTAINS THE BIT MASK.  ALL OTHER REGISTERS ARE INTACT
                        ;--------------------------------------------------------------------
EB45                    GET_DRIVE       PROC    NEAR
EB45  51                        PUSH    CX              ; SAVE REGISTER.
EB46  8A 4E 00                  MOV     CL,BYTE PTR[BP] ; GET DRIVE NUMBER
EB49  B0 01                     MOV     AL,1            ; INITIALIZE AL WITH VALUE FOR
                                                        ; SHIFTING
EB4B  D2 E0                     SHL     AL,CL           ; SHIFT BIT POSITION BY DRIVE
                                                        ; NUMBER (DRIVE IN RANGE 0-2)
EB4D  24 07                     AND     AL,07H          ; ONLY THREE DRIVES ARE SUPPORTED.
                                                        ;  RANGE CHECK
EB4F  59                        POP     CX              ; RESTORE REGISTERS
EB50  C3                        RET
EB51                    GET_DRIVE       ENDP
                        ;--------------------------------------------
                        ; THIS ROUTINE  CHECKS OPTIONAL ROM MODULES (CHECKSUM
                        ; FOR MODULES FROM C0000->D0000, CRC CHECK FOR CARTRIDGES
                        ; (D0000->F0000)
                        ; IF CHECK IS OK, CALLS INIT/TEST CODE IN MODULE
                        ;    MFG ERROR CODE= 25XX (XX=MSB OF SEGMENT IN ERROR)
                        ;--------------------------------------------
EB51                    ROM_CHECK       PROC    NEAR
EB51  2B F6                     SUB     SI,SI           ; SET SI TO POINT TO BEGINNING
                                                        ; (REL. TO DS)
EB53  2A C0                     SUB     AL,AL           ; ZERO OUT AL
EB55  8A 67 02                  MOV     AH,[BX+2]       ; GET LENGTH INDICATOR
EB58  D1 E0                     SHL     AX,1            ; FORM COUNT
EB5A  50                        PUSH    AX              ; SAVE COUNT
EB5B  81 FA D000                CMP     DX,0D000H       ; SEE IF POINTER IS BELOW D000
EB5F  9C                        PUSHF                   ; SAVE RESULTS
EB60  B1 04                     MOV     CL,4            ; ADJUST
EB62  D3 E8                     SHR     AX,CL           ;
EB64  03 D0                     ADD     DX,AX           ; SET POINTER TO NEXT MODULE
EB66  9D                        POPF                    ; RECOVER FLAGS FROM POINTER RANGE
                                                        ; CHECK
EB67  59                        POP     CX              ; RECOVER COUNT IN CX REGISTER
EB68  52                        PUSH    DX              ; SAVE POINTER
EB69  7C 07                     JL      ROM_1           ; DO ARITHMETIC CHECKSUM IF BELOW
                                                        ; D0000
EB6B  E8 FE71 R                 CALL    CRC_CHECK       ; DO CRC CHECK
EB6E  74 2B                     JZ      ROM_CHECK_1     ; PROCEED IF OK
EB70  EB 05                     JMP     SHORT ROM_2     ; ELSE POST ERROR
EB72  E8 FEEB R         ROM_1:  CALL    ROS_CHECKSUM    ; DO ARITHMETIC CHECKSUM
EB75  74 24                     JZ      ROM_CHECK_1     ; PROCEED IF OK
EB77  BA 1626           ROM_2:  MOV     DX,1626H        ; POSITION CURSOR, ROW 22, COL 38
EB7A  B4 02                     MOV     AH,2
EB7C  B7 07                     MOV     BH,7
EB7E  CD 10                     INT     10H
EB80  8C DA                     MOV     DX,DS           ; RECOVER DATA SEG
EB82  8A C6                     MOV     AL,DH           ;
EB84  E8 18A9 R                 CALL    XPC_BYTE        ; DISPLAY MSB OF DATA SEG
EB87  8A DE                     MOV     BL,DH           ; FORM XX VALUE OF ERROR CODE
EB89  B7 25                     MOV     BH,25H          ; FORM 25 PORTION
EB8B  80 FE D0                  CMP     DH,0D0H         ; IN CARTRIDGE SPACE?
EB8E  BE 003B R                 MOV     SI,OFFSET CART_ERR
EB91  7D 03                     JGE     ROM_CHECK_0     ;
EB93  BE 003A R                 MOV     SI,OFFSET ROM_ERR
EB96                    ROM_CHECK_0:
EB96  E8 09BC R                 CALL    E_MSG           ; GO ERROR ROUTINE
EB99  EB 16                     JMP     SHORT ROM_CHECK_END     ; AND EXIT
EB9B                    ROM_CHECK_1:
EB9B  B8  ---- R                MOV     AX,XXDATA       ; SET ES TO POINT TO XXDATA AREA
EB9E  8E C0                     MOV     ES,AX           ;
EBA0  26: C7 06 0014 R 0003     MOV     ES:IO_ROM_INIT,0003H  ; LOAD OFFSET
EBA7  26: 8C 1E 0016 R          MOV     ES:IO_ROM_SEG,DS      ; LOAD SEGMENT
EBAC  26: FF 1E 0014 R          CALL    DWORD PTR ES:IO_ROM_INIT ; CALL INIT./TEST ROUTINE
```

```
EBB1                            ROM_CHECK_END:
EBB1  5A                                POP     DX              ; RECOVER POINTER
EBB2  C3                                RET                     ; RETURN TO CALLER
EBB3                            ROM_CHECK       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 REQD ON
                                ;               ALL DRIVES
                                ;       (AH)=1  READ THE STATUS OF THE SYSTEM INTO (AL)
                                ;               DISKETTE_STATUS FROM LAST OP'N IS USED
                                ;       REGISTERS FOR READ/WRITE/VERIFY/FORMAT
                                ;       (DL) - DRIVE NUMBER (0-3 ALLOWED, VALUE CHECKED)
                                ;       (DH) - HEAD NUMBER (0-1 ALLOWED, NOT VALUE CHECKED)
                                ;       (CH) - TRACK NUMBER (0-39, NOT VALUE CHECKED)
                                ;       (CL) - SECTOR NUMBER (1-8, NOT VALUE CHECKED, NOT USED FOR
                                ;              FORMAT)
                                ;       (AL) - NUMBER OF SECTORS ( MAX = 8, NOT VALUE CHECKED, NOT
                                ;              USED FOR FORMAT, HOWEVER, CANNOT BE ZERO!!!)
                                ;       (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  06                                PUSH    ES              ; SAVE ES
EC5B  50                                PUSH    AX              ; ALLOCATE ONE WORD OF STORAGE FOR
                                                                ; TIMER1 INITIAL VALUE
EC5C  50                                PUSH    AX              ; ALLOCATE ONE WORD ON STACK FOR
                                                                ; USE IN PROCS ENABLE AND DISABLE.
                                                                ;  WILL HOLD 8259 MASK.
EC5D  50                                PUSH    AX              ; SAVE COMMAND AND N_SECTORS
EC5E  53                                PUSH    BX              ; SAVE ADDRESS
EC5F  51                                PUSH    CX
EC60  1E                                PUSH    DS              ; SAVE SEGMENT REGISTER VALUE
EC61  56                                PUSH    SI              ; SAVE ALL REGISTERS DURING
                                                                ; OPERATION
EC62  57                                PUSH    DI
EC63  55                                PUSH    BP
EC64  52                                PUSH    DX
EC65  8B EC                             MOV     BP,SP           ; SET UP POINTER TO HEAD PARM
EC67  E8 138B R                         CALL    DDS             ; SET DS=DATA
EC6A  E8 EC90 R                         CALL    J1              ; CALL THE REST TO ENSURE DS
                                                                ; RESTORED
EC6D  B3 04                             MOV     BL,4            ; GET THE MOTOR WAIT PARAMETER
EC6F  E8 E984 R                         CALL    GET_PARM
EC72  88 26 0040 R                      MOV     MOTOR_COUNT,AH  ; SET THE TIMER COUNT FOR THE MOTOR
EC76  8A 26 0041 R                      MOV     AH,DISKETTE_STATUS      ; GET STATUS OF OPERATION
EC7A  88 66 0F                          MOV     [BP+15],AH      ; RETURN STATUS IN AL
EC7D  5A                                POP     DX              ; RESTORE ALL REGISTERS
EC7E  5D                                POP     BP
EC7F  5F                                POP     DI
EC80  5E                                POP     SI
EC81  1F                                POP     DS
EC82  59                                POP     CX
EC83  5B                                POP     BX              ; RECOVER OFFSET
EC84  58                                POP     AX
EC85  83 C4 04                          ADD     SP,4            ; DISCARD DUMMY SPACE FOR 8259 MASK
EC88  07                                POP     ES              ; RECOVER SEGMENT
EC89  80 FC 01                          CMP     AH,1            ; SET THE CARRY FLAG TO INDICATE
EC8C  F5                                CMC                     ;  SUCCESS OR FAILURE
EC8D  CA 0002                           RET     2               ; THROW AWAY SAVED FLAGS
```

```
EC90                                    DISKETTE_IO     ENDP
EC90                                    J1      PROC    NEAR
EC90  8A F0                                     MOV     DH,AL           ; SAVE # SECTORS IN DH
EC92  80 26 003F R 7F                           AND     MOTOR_STATUS,07FH       ; INDICATE A READ OPERATION
EC97  0A E4                                     OR      AH,AH           ; AH=0
EC99  74 27                                     JZ      DISK_RESET
EC9B  FE CC                                     DEC     AH              ; AH=1
EC9D  74 74                                     JZ      DISK_STATUS
EC9F  C6 06 0041 R 00                           MOV     DISKETTE_STATUS,0 ; RESET THE STATUS INDICATOR
ECA4  80 FA 02                                  CMP     DL,2            ; TEST FOR DRIVE IN 0-2 RANGE
ECA7  77 13                                     JA      J3              ; ERROR IF ABOVE
ECA9  FE CC                                     DEC     AH              ; AH=2
ECAB  74 6D                                     JZ      DISK_READ
ECAD  FE CC                                     DEC     AH              ; AH=3
ECAF  75 03                                     JNZ     J2              ; TEST_DISK_VERF
ECB1  E9 ED3D R                                 JMP     DISK_WRITE
ECB4                                    J2:                             ; TEST_DISK_VERF
ECB4  FE CC                                     DEC     AH              ; AH=4
ECB6  74 62                                     JZ      DISK_VERF
ECB8  FE CC                                     DEC     AH              ; AH=5
ECBA  74 62                                     JZ      DISK_FORMAT
ECBC                                    J3:                             ; BAD_COMMAND
ECBC  C6 06 0041 R 01                           MOV     DISKETTE_STATUS,BAD_CMD ; ERROR CODE, NO SECTORS
                                                                        ; TRANSFERRED
ECC1  C3                                        RET                     ; UNDEFINED OPERATION
ECC2                                    J1      ENDP
                                        ;------ RESET THE DISKETTE SYSTEM
ECC2                                    DISK_RESET      PROC    NEAR
ECC2  BA 00F2                                   MOV     DX,NEC_CTL      ; ADAPTER CONTROL PORT
ECC5  FA                                        CLI                     ; NO INTERRUPTS
ECC6  A0 003F R                                 MOV     AL,MOTOR_STATUS ; FIND OUT IF MOTOR IS RUNNING
ECC9  24 07                                     AND     AL,07H          ; DRIVE BITS
ECCB  EE                                        OUT     DX,AL           ; RESET THE ADAPTER
ECCC  C6 06 003E R 00                           MOV     SEEK_STATUS,0   ; SET RECAL REQUIRED ON ALL DRIVES
ECD1  C6 06 0041 R 00                           MOV     DISKETTE_STATUS,0 ; SET OK STATUS FOR DISKETTE
ECD6  0C 80                                     OR      AL,FDC_RESET    ; TURN OFF RESET
ECD8  EE                                        OUT     DX,AL           ; TURN OFF THE RESET
ECD9  FB                                        STI                     ; REENABLE THE INTERRUPTS
ECDA  BE ECFA R                                 MOV     SI,OFFSET J4_2  ; DUMMY RETURN FOR
ECDD  56                                        PUSH    SI              ; PUSH RETURN IF ERROR
                                                                        ; IN NEC_OUTPUT
ECDE  B9 0010                                   MOV     CX,10H          ; NUMBER OF SENSE INTERRUPTS TO
                                                                        ; ISSUE
ECE1  B4 08                             J4_0:   MOV     AH,08H          ; COMMAND FOR SENSE INTERRUPT
                                                                        ; STATUS
ECE3  E8 E98A R                                 CALL    NEC_OUTPUT      ; OUTPUT  THE SENSE INTERRUPT
                                                                        ; STATUS
ECE6  E8 EAA0 R                                 CALL    RESULTS         ; GET STATUS FOLLOWING COMPLETION
                                                                        ; OF RESET
ECE9  A0 0042 R                                 MOV     AL,NEC_STATUS   ; IGNORE ERROR RETURN AND DO OWN
                                                                        ; TEST
ECEC  3C C0                                     CMP     AL,0C0H         ; TEST FOR DRIVE READY TRANSITION
ECEE  74 12                                     JZ      J7              ; EVERYTHING OK
ECF0  E2 EF                                     LOOP    J4_0            ; RETRY THE COMMAND
ECF2  80 0E 0041 R 20                   J4_1:   OR      DISKETTE_STATUS,BAD_NEC ; SET ERROR CODE
ECF7  5E                                        POP     SI
ECF8  EB 18                                     JMP     SHORT J8
ECFA  BE ECFA R                         J4_2:   MOV     SI,OFFSET J4_2  ; NEC_OUTPUT FAILED, RETRY THE
                                                                        ; SENSE INTERRUPT
ECFD  56                                        PUSH    SI              ; OFFSET OF BAD RETURN IN
                                                                        ; NEC_OUTPUT
ECFE  E2 E1                                     LOOP    J4_0            ; RETRY
ED00  EB F0                                     JMP     SHORT J4_1
                                        ;------ SEND SPECIFY COMMAND TO NEC
ED02  5E                                J7:     POP     SI              ; GET RID OF DUMMY ARGUMENT
ED03  B4 03                                     MOV     AH,03H          ; SPECIFY COMMAND
ED05  E8 E98A R                                 CALL    NEC_OUTPUT      ; OUTPUT THE COMMAND
ED08  B3 01                                     MOV     BL,1            ; STEP RATE TIME AND HEAD UNLOAD
ED0A  E8 E9B4 R                                 CALL    GET_PARM        ; OUTPUT TO THE NEC CONTROLLER
ED0D  B3 03                                     MOV     BL,3            ; PARM1 HEAD LOAD AND NO DMA
ED0F  E8 E9B4 R                                 CALL    GET_PARM        ;  TO THE NEC CONTROLLER
ED12                                    J8:                             ; RESET_RET
ED12  C3                                        RET                     ; RETURN TO CALLER
ED13                                    DISK_RESET      ENDP
                                        ;------ DISKETTE STATUS ROUTINE
ED13                                    DISK_STATUS     PROC    NEAR
ED13  A0 0041 R                                 MOV     AL,DISKETTE_STATUS
ED16  88 46 0E                                  MOV     BYTE PTR[BP+14],AL ; PUT STATUS ON STACK, IT WILL
                                                                        ; POP IN AL
ED19  C3                                        RET
ED1A                                    DISK_STATUS     ENDP
                                        ;------ DISKETTE VERIFY
ED1A                                    DISK_VERF       LABEL   NEAR
                                        ;------ DISKETTE READ
ED1A                                    DISK_READ       PROC    NEAR
ED1A                                    J9:                             ; DISK_READ_CONT
ED1A  B4 46                                     MOV     AH,046H         ; SET UP READ COMMAND FOR NEC
                                                                        ; CONTROLLER
ED1C  EB 26                                     JMP     SHORT RW_OPN    ; GO DO THE OPERATION
ED1E                                    DISK_READ       ENDP
                                        ;------ DISKETTE FORMAT
ED1E                                    DISK_FORMAT     PROC    NEAR
ED1E  80 0E 003F R 80                           OR      MOTOR_STATUS,80H ; INDICATE A WRITE OPERATION
ED23  B4 4D                                     MOV     AH,04DH         ; ESTABLISH THE FORMAT COMMAND
ED25  EB 1D                                     JMP     SHORT RW_OPN    ; DO THE OPERATION
```

```
ED27                                    J10:                                    ; CONTINUATION OF RW_OPN FOR FMT
ED27  B3 07                                     MOV     BL,7                    ; GET THE
ED29  E8 E9B4 R                                 CALL    GET_PARM                ;  BYTES/SECTOR VALUE TO NEC
ED2C  B3 09                                     MOV     BL,9                    ; GET THE
ED2E  E8 E9B4 R                                 CALL    GET_PARM                ;  SECTORS/TRACK VALUE TO NEC
ED31  B3 0F                                     MOV     BL,15                   ; GET THE
ED33  E8 E9B4 R                                 CALL    GET_PARM                ;  GAP LENGTH VALUE TO NEC
ED36  BB 0011                                   MOV     BX,17                   ; GET THE FILLER BYTE
ED39  53                                        PUSH    BX                      ; SAVE PARAMETER INDEX ON STACK
ED3A  E9 EDCD R                                 JMP     J16                     ;  TO THE CONTROLLER
ED3D                                    DISK_FORMAT     ENDP
                                        ;------ DISKETTE WRITE ROUTINE
ED3D                                    DISK_WRITE      PROC    NEAR
ED3D  80 0E 003F R 80                           OR      MOTOR_STATUS,80H ; INDICATE A WRITE OPERATION
ED42  B4 45                                     MOV     AH,045H                 ; NEC COMMAND TO WRITE TO DISKETTE
ED44                                    DISK_WRITE      ENDP
                                        ;----- ALLOW WRITE ROUTINE TO FALL INTO RW_OPN
                                        ;--------------------------------------
                                        ; RW_OPN
                                        ;       THIS ROUTINE PERFORMS THE READ/WRITE/VERIFY OPERATION
                                        ;--------------------------------------
ED44                                    RW_OPN  PROC    NEAR
ED44  50                                        PUSH    AX                      ; SAVE THE COMMAND
                                        ;------ TURN ON THE MOTOR AND SELECT THE DRIVE
ED45  51                                        PUSH    CX                      ; SAVE THE T/S PARMS
ED46  FA                                        CLI                             ; NO INTERRUPTS WHILE DETERMINING
                                                                                ; MOTOR STATUS
ED47  C6 06 0040 R FF                           MOV     MOTOR_COUNT,0FFH ; SET LARGE COUNT DURING OPERATION
ED4C  E8 EB45 R                                 CALL    GET_DRIVE               ; GET THE DRIVE PARAMETER FROM THE
                                                                                ; STACK
ED4F  84 06 003F R                              TEST    MOTOR_STATUS,AL ; TEST MOTOR FOR OPERATING
ED53  75 1F                                     JNZ     J14                     ; IF RUNNING, SKIP THE WAIT
ED55  80 26 003F R F0                           AND     MOTOR_STATUS,0F0H ; TURN OFF RUNNING DRIVE
ED5A  08 06 003F R                              OR      MOTOR_STATUS,AL         ; TURN ON THE CURRENT MOTOR
ED5E  FB                                        STI                             ; INTERRUPTS BACK ON
ED5F  0C 80                                     OR      AL,FDC_RESET    ; NO RESET.  TURN ON MOTOR
ED61  E6 F2                                     OUT     NEC_CTL,AL
                                        ;------ WAIT FOR MOTOR BOTH READ AND WRITE
ED63  B3 14                                     MOV     BL,20                   ; GET MOTOR START TIME
ED65  E8 E9B4 R                                 CALL    GET_PARM
ED68  0A E4                                     OR      AH,AH                   ; TEST FOR NO WAIT
ED6A                                    J12:                                    ; TEST_WAIT_TIME
ED6A  74 08                                     JZ      J14                     ; EXIT WITH TIME EXPIRED
ED6C  2B C9                                     SUB     CX,CX                   ; SET UP 1/8 SECOND LOOP TIME
ED6E  E2 FE                             J13:    LOOP    J13                     ; WAIT FOR THE REQUIRED TIME
ED70  FE CC                                     DEC     AH                      ; DECREMENT TIME VALUE
ED72  EB F6                                     JMP     J12                     ; ARE WE DONE YET
ED74                                    J14:                                    ; MOTOR_RUNNING
ED74  FB                                        STI                             ; INTERRUPTS BACK ON FOR BYPASS
                                                                                ; WAIT
ED75  59                                        POP     CX
                                        ;------ DO THE SEEK OPERATION
ED76  E8 E9FB R                                 CALL    SEEK                    ; MOVE TO CORRECT TRACK
ED79  58                                        POP     AX                      ; RECOVER COMMAND
ED7A  8A FC                                     MOV     BH,AH                   ; SAVE COMMAND IN BH
ED7C  B6 00                                     MOV     DH,0                    ; SET NO SECTORS READ IN CASE OF
                                                                                ; ERROR
ED7E  73 03                                     JNC     J14_1                   ; IF NO ERROR CONTINUE, JUMP AROUND
                                                                                ; JMP
ED80  E9 EED7 R                                 JMP     J17                     ; CARRY SET JUMP TO MOTOR WAIT
ED83  BE EED7 R                         J14_1:  MOV     SI,OFFSET J17   ; DUMMY RETURN ON STACK FOR
                                                                                ; NEC_OUTPUT
ED86  56                                        PUSH    SI                      ;  SO THAT IT WILL RETURN TO MOTOR
                                                                                ; OFF LOCATION
                                        ;------ SEND OUT THE PARAMETERS TO THE CONTROLLER
ED87  E8 E98A R                                 CALL    NEC_OUTPUT      ; OUTPUT THE OPERATION COMMAND
ED8A  8A 66 01                                  MOV     AH,[BP+1]       ; GET THE CURRENT HEAD NUMBER
ED8D  D0 E4                                     SAL     AH,1                    ; MOVE IT TO BIT 2
ED8F  D0 E4                                     SAL     AH,1
ED91  80 E4 04                                  AND     AH,4                    ; ISOLATE THAT BIT
ED94  0A E2                                     OR      AH,DL                   ; OR IN THE DRIVE NUMBER
ED96  E8 E98A R                                 CALL    NEC_OUTPUT
                                        ;------ TEST FOR FORMAT COMMAND
ED99  80 FF 4D                                  CMP     BH,04DH                 ; IS THIS A FORMAT OPERATION?
ED9C  75 02                                     JNE     J15                     ; NO. CONTINUE WITH R/W/V
ED9E  EB 87                                     JMP     J10                     ; IF SO, HANDLE SPECIAL
EDA0  8A E5                             J15:    MOV     AH,CH                   ; CYLINDER NUMBER
EDA2  E8 E98A R                                 CALL    NEC_OUTPUT
EDA5  8A 66 01                                  MOV     AH,[BP+1]       ; HEAD NUMBER FROM STACK
EDA8  E8 E98A R                                 CALL    NEC_OUTPUT
EDAB  8A E1                                     MOV     AH,CL                   ; SECTOR NUMBER
EDAD  E8 E98A R                                 CALL    NEC_OUTPUT
EDB0  B3 07                                     MOV     BL,7                    ; BYTES/SECTOR PARM FROM BLOCK
EDB2  E8 E9B4 R                                 CALL    GET_PARM                ;  TO THE NEC
EDB5  B3 08                                     MOV     BL,8                    ; EOT PARM FROM BLOCK
EDB7  E8 E9B4 R                                 CALL    GET_PARM                ;  RETURNED IN AH
EDBA  02 4E 0E                                  ADD     CL,[BP+14]      ; ADD CURRENT SECTOR TO NUMBER IN
                                                                                ; TRANSFER
EDBD  FE C9                                     DEC     CL                      ; CURRENT_SECTOR + N_SECTORS - 1
EDBF  8A E1                                     MOV     AH,CL                   ; EOT PARAMETER IS THE CALCULATED
                                                                                ; ONE
EDC1  E8 E98A R                                 CALL    NEC_OUTPUT
EDC4  B3 0B                                     MOV     BL,11                   ; GAP LENGTH PARM FROM BLOCK
EDC6  E8 E9B4 R                                 CALL    GET_PARM                ;  TO THE NEC
EDC9  BB 000D                                   MOV     BX,13                   ; DTL PARM FROM BLOCK
EDCC  53                                        PUSH    BX                      ; SAVE INDEX TO DISK PARAMETER ON
                                                                                ; STACK
```

```
EDCD  FC                      J16:    CLD                     ; FORWARD DIRECTION
                              ;------ START TIMER1 WITH INITIAL VALUE OF FFFF
EDCE  B0 70                           MOV     AL,01110000B    ; SELECT TIMER1,LSB-MSB, MODE 0,
                                                              ; BINARY COUNTER
EDD0  E6 43                           OUT     TIM_CTL,AL      ; INITIALIZE THE COUNTER
EDD2  50                              PUSH    AX
EDD3  58                              POP     AX              ; ALLOW ENOUGH TIME FOR THE 8253 TO
                                                              ; INITIALIZE ITSELF
EDD4  B0 FF                           MOV     AL,0FFH         ; INITIAL COUNT VALUE FOR THE 8253
EDD6  E6 41                           OUT     TIMER+1,AL      ; OUTPUT LEAST SIGNIFICANT BYTE
EDD8  50                              PUSH    AX
EDD9  58                              POP     AX              ; WAIT
EDDA  E6 41                           OUT     TIMER+1,AL      ; OUTPUT MOST SIGNIFACNT BYTE
                              ;------INITIALIZE CX FOR JUMP AFTER LAST PARAMETER IS PASSED TO NEC
EDDC  8A 46 0F                        MOV     AL,[BP+15]      ; RETRIEVE COMMAND PARAMETER
EDDF  A8 01                           TEST    AL,01H          ; IS THIS AN ODD NUMBERED FUNCTION?
EDE1  74 05                           JZ      J16_1           ; JUMP IF NOT ODD NUMBERED
EDE3  B9 EE4E R                       MOV     CX,OFFSET WRITE_LOOP
EDE6  EB 0C                           JMP     SHORT J16_3
EDE8  3C 02                   J16_1:  CMP     AL,2            ; IS THIS A READ?
EDEA  75 05                           JNZ     J16_2           ; JUMP IF VERIFY
EDEC  B9 EE3A R                       MOV     CX,OFFSET READ_LOOP
EDEF  EB 03                           JMP     SHORT J16_3
EDF1  B9 EE20 R               J16_2:  MOV     CX,OFFSET VERIFY_LOOP
                              ;------FINISH INITIALIZATION
EDF4                          J16_3:
                              ;--------------------------------------------------------------
                              ;***NOTE***
                              ;ALL INTERRUPTS ARE ABOUT TO BE DISABLED.  THERE IS A POTENTIAL
                              ;       THAT THIS TIME PERIOD WILL BE LONG ENOUGH TO MISS TIME OF
                              ;       DAY INTERRUPTS.  FOR THIS REASON, TIMER1 WILL BE USED TO
                              ;       KEEP TRACK OF THE NUMBER OF TIME OF DAY INTERRUPTS WHICH
                              ;       WILL BE MISSED. THIS INFORMATION IS USED AFTER THE DISKETTE
                              ;       OPERATION TO UPDATE THE TIME OF DAY.
                              ;--------------------------------------------------------------
EDF4  B0 10                           MOV     AL,10H          ; DISABLE NMI
EDF6  E6 A0                           OUT     NMI_PORT,AL     ; NO KEYBOARD INTERRUPT
EDF8  E8 EB31 R                       CALL    CLOCK_WAIT      ; WAIT IF TIMER0 IS ABOUT TO
                                                              ; INTERRUPT
                              ;------ ENABLE WATCHDOG TIMER
                              ;--------------------------------------------------------------
                              ;***NOTE***
                              ;       GIVEN THE CURRENT SYSTEM CONFIGURATION A  METHOD IS NEEDED
                              ;       TO PULL THE NEC OUT OF "FATAL ERROR" SITUATIONS.  A TIMER
                              ;       ON THE ADAPTER CARD IS PROVIDED WHICH WILL PERFORM THIS
                              ;       FUNCTION. THE WATCHDOG TIMER ON THE ADAPTER CARD IS ENABLED
                              ;       AND STROBED BEFORE THE 8259 INTERRUPT 6 LINE IS ENABLED.
                              ;       THIS IS BECAUSE OF A GLITCH ON THE LINE LARGE ENOUGH TO
                              ;       TRIGGER AN INTERRUPT.
                              ;--------------------------------------------------------------
EDFB  E8 EB45 R                       CALL    GET_DRIVE       ; GET BIT MASK FOR DRIVE
EDFE  BA 00F2                         MOV     DX,NEC_CTL      ; CONTROL PORT TO NEC
EE01  0C E0                           OR      AL,FDC_RESET+WD_ENABLE+WD_STROBE
EE03  EE                              OUT     DX,AL           ; OUTPUT CONTROL INFO FOR
                                                              ; WATCHDOG(WD) ENABLE
EE04  24 A7                           AND     AL,FDC_RESET+WD_ENABLE+7H
EE06  EE                              OUT     DX,AL           ; OUTPUT CONTROL INFO TO STROBE
                                                              ; WATCHDOG
EE07  BA 00F4                         MOV     DX,NEC_STAT     ; PORT TO NEC STATUS
EE0A  B0 20                           MOV     AL,20H          ; SELECT TIMER1 INPUT FROM TIMER0
                                                              ; OUTPUT
EE0C  E6 A0                           OUT     NMI_PORT,AL
                              ;------ READ TIMER1 NOW AND SAVE THE INITIAL VALUE
EE0E  E8 E81A R                       CALL    READ_TIME       ; GET TIMER1 VALUE
EE11  89 46 12                        MOV     [BP+18],AX      ; SAVE INITIAL VALUE FOR CLOCK
                                                              ; UPDATE IN TEMPORAY STORAGE
EE14  E8 EAFC R                       CALL    DISABLE         ; DISABLE ALL INTERRUPTS
                              ;------ NEC BEGINS OPERATION WHEN NEC RECEIVES LAST PARAMETER
EE17  5B                              POP     BX              ; GET PARAMTER FROM STACK
EE18  E8 E984 R                       CALL    GET_PARM        ; OUTPUT LAST PARAMETER TO THE NEC
EE1B  58                              POP     AX              ; CAN NOW DISCARD THAT DUMMY RETURN
                                                              ; ADDRESS
EE1C  06                              PUSH    ES
EE1D  1F                              POP     DS              ; INITIALIZE DS FOR WRITE
EE1E  FF E1                           JMP     CX              ; JUMP TO APPROPRIATE R/W/V LOOP
                              ;--------------------------------------------------------------
                              ;***NOTE***
                              ;       DATA IS TRANSFERRED USING POLLING ALGORITHMS.  THESE LOOPS
                              ;       TRANSFER A DATA BYTE AT A TIME WHILE POLLING THE NEC FOR
                              ;       NEXT DATA BYTE AND COMPLETION STATUS.
                              ;--------------------------------------------------------------
                              ;-----VERIFY OPERATION
EE20                          VERIFY_LOOP:
EE20  EC                              IN      AL,DX           ; READ STATUS
EE21  A8 20                           TEST    AL,BUSY_BIT     ; HAS NEC ENTERED EXECUTION PHASE
                                                              ; YET?
EE23  74 F8                           JZ      VERIFY_LOOP     ; NO, CONTINUE SAMPLING
EE25                          J22_2:
EE25  A8 80                           TEST    AL,RQM          ; IS DATA READY?
EE27  75 07                           JNZ     J22_4           ; JUMP IF DATA TRANSFER IS READY
EE29  EC                              IN      AL,DX           ; READ STATUS PORT
EE2A  A8 20                           TEST    AL,BUSY_BIT     ; ARE WE DONE?
EE2C  75 F7                           JNZ     J22_2           ; JUMP IF MORE TRANSFERS
EE2E  EB 35                           JMP     SHORT OP_END    ; TRANSFER DONE
EE30  42                      J22_4:  INC     DX              ; POINT AT NEC DATA REGISTER
EE31  EC                              IN      AL,DX           ; READ DATA
EE32  4A                              DEC     DX              ; POINT AT NEC STATUS REGISTER
EE33  EC                              IN      AL,DX           ; READ STATUS PORT
EE34  A8 20                           TEST    AL,BUSY_BIT     ; ARE WE DONE?
EE36  75 ED                           JNZ     J22_2           ; CONTINUE
EE38  EB 2B                           JMP     SHORT OP_END    ; WE ARE DONE
```

```
                        ;-----READ OPERATION
EE3A                    READ_LOOP:
EE3A  EC                        IN      AL,DX           ; READ STATUS REGISTER
EE3B  A8 20                     TEST    AL,BUSY_BIT     ; HAS NEC STARTED THE EXECUTION
                                                        ; PHASE?
EE3D  74 F8                     JZ      READ_LOOP       ; HAS NOT STATRED YET
EE3F  EC                J22_5:  IN      AL,DX           ; READ STATUS PORT
EE40  A8 20                     TEST    AL,BUSY_BIT     ; HAS NEC COMPLETED EXECUTION
                                                        ; PHASE?
EE42  74 21                     JZ      OP_END          ; JUMP IF EXECUTION PHASE IS OVER
EE44  A8 80                     TEST    AL,RQM          ; IS DATA READY?
EE46  74 F7                     JZ      J22_5           ; READ THE DATA
EE48  42                        INC     DX              ; POINT AT NEC_DATA
EE49  EC                        IN      AL,DX           ; READ DATA
EE4A  AA                        STOSB                   ; TRANSFER DATA
EE4B  4A                        DEC     DX              ; POINT AT NEC_STATUS
EE4C  EB F1                     JMP     J22_5           ; CONTINUE WITH READ OPERATION
                        ;-----WRITE AND FORMAT OPERATION
EE4E                    WRITE_LOOP:
EE4E  EC                        IN      AL,DX           ; READ NEC STATUS PORT
EE4F  A8 20                     TEST    AL,BUSY_BIT     ; HAS THE NEC ENTERED EXECUTION
                                                        ; PHASE YET?
EE51  74 FB                     JZ      WRITE_LOOP      ; NO, CONTINUE LOOPING
EE53  B9 2080                   MOV     CX,BUSY_BIT*256+RQM
EE56                    J22_7:
EE56  EC                        IN      AL,DX           ; READ STATUS PORT
EE57  84 C5                     TEST    AL,CH           ; IS THE FEC STILL IN THE EXECUTION
                                                        ; PHASE?
EE59  74 0A                     JZ      OP_END          ; JUMP IF EXECUTION PHASE IS DONE.
EE5B  84 C1                     TEST    AL,CL           ; IS THE DATA PORT READY FOR THE
                                                        ; TRANSFER?
EE5D  74 F7                     JZ      J22_7           ; JUMP TO WRITE DATA
EE5F  42                        INC     DX              ; POINT AT DATA REGISTER
EE60  AC                        LODSB                   ; TRANSFER BYTE
EE61  EE                        OUT     DX,AL           ; WRITE THE BYTE ON THE DISKETTE
EE62  4A                        DEC     DX              ; POINT AT THE STATUS REGISTER
EE63  EB F1                     JMP     J22_7           ; CONTINUE WITH WRITE OR FORMAT
                        ;------TRANSFER PROCESS IS OVER
EE65  9C                OP_END: PUSHF                   ; SAVE THE CARRY BIT SET IN
                                                        ; DISK_INT
EE66  E8 EB45 R                 CALL    GET_DRIVE       ; GET BIT MASK FOR DRIVE SELECTION
EE69  0C 80                     OR      AL,FDC_RESET    ; NO RESET, KEEP DRIVE SPINNING
EE6B  BA 00F2                   MOV     DX,NEC_CTL      ;
EE6E  EE                        OUT     DX,AL           ; DISABLE WATCHDOG
                        ;------ UPDATE TIME OF DAY
EE6F  E8 138B R                 CALL    DDS             ; POINT DS AT BIOS DATA SEGMENT
EE72  E8 E831 R                 CALL    CLOCK_WAIT      ; WAIT IF TIMER0 IS CLOSE TO
                                                        ; WRAPPING
EE75  E8 E81A R                 CALL    READ_TIME
EE78  8B 5E 12                  MOV     BX,[BP+18]      ; GET THE INITIAL VALUE OF TIMER1
EE7B  2B C3                     SUB     AX,BX           ; UPDATE NUMBER OF INTERRUPTS
                                                        ; MISSED
EE7D  F7 D8                     NEG     AX              ; PUT IT IN AX
EE7F  50                        PUSH    AX              ; SAVE IT FOR REUSE IN ISSUING USER
                                                        ; TIMER INTERRUPTS
EE80  01 06 006C R              ADD     TIMER_LOW,AX    ; ADD NUMBER OF TIMER INTERRUPTS TO
                                                        ; TIME
EE84  73 04                     JNC     J16_4           ; JUMP IF TIMER_LOW DID NOT SPILL
                                                        ; OVER TO TIMER_HI
EE86  FF 06 006E R              INC     TIMER_HIGH
EE8A  83 3E 006E R 18   J16_4:  CMP     TIMER_HIGH,018H ; TEST FOR COUNT TOTALING 24 HOURS
EE8F  75 19                     JNZ     J16_5           ; JUMP IF NOT 24 HOURS
EE91  81 3E 006C R 00B0         CMP     TIMER_LOW,0B0H  ; LOW VALUE = 24 HOUR VALUE?
EE97  7C 11                     JL      J16_5           ; NOT 24 HOUR VALUE?
                        ;------ TIMER HAS GONE 24 HOURS
EE99  C7 06 006E R 0000         MOV     TIMER_HIGH,0    ; ZERO OUT TIMER_HIGH VALUE
EE9F  81 2E 006C R 00B0         SUB     TIMER_LOW,0B0H  ; VALUE REFLECTS CORRECT TICKS PAST
                                                        ; 00B0H
EEA5  C6 06 0070 R 01           MOV     TIMER_OFL,1     ; INDICATES 24 HOUR THRESHOLD
EEAA  E8 E808 R         J16_5:  CALL    ENABLE          ; ENABLE ALL INTERRUPTS
EEAD  59                        POP     CX              ; CX:=AX, COUNT FOR NUMBER OF USER
                                                        ; TIME INTERRUPTS
EEAE  E3 26                     JCXZ    J16_7           ; IF ZERO DO NOT ISSUE ANY
                                                        ; INTERRUPTS
EEB0  1E                        PUSH    DS              ; SAVE ALL REGISTERS SAVED PRIOR TO
                                                        ; INT 1C CALL FROM TIMERINT
EEB1  50                        PUSH    AX              ; THIS PROVIDES A COMPATIBLE
                                                        ; INTERFACE TO 1C
EEB2  52                        PUSH    DX              ;
EEB3                    J16_6:
EEB3  CD 1C                     INT     1CH             ; TRANSFER CONTROL TO USER
                                                        ; INTERRUPT
EEB5  E2 FC                     LOOP    J16_6           ; DO ALL USER TIMER INTERRUPTS
EEB7  5A                        POP     DX
EEB8  58                        POP     AX
EEB9  1F                        POP     DS              ; RESTORE REGISTERS
                        ;------ CLOCK IS UPDATED AND USER INTERRUPTS 1C HAVE BEEN ISSUED.
                        ;       CHECK IF KEYSTROKE OCCURED
EEBA  0A C0                     OR      AL,AL           ; AL WAS SET DURING CALL TO ENABLE
EEBC  74 1B                     JZ      J16_7           ; NO KEY WAS PRESSED WHILE SYSTEM
                                                        ; WAS MASKED
EEBE  BB 00B0                   MOV     BX,0B0H         ; DURATION OF TONE
EEC1  B9 0048                   MOV     CX,048H         ; FREQUNCY OF TONE
EEC4  E8 E035 R                 CALL    KB_NOISE        ; NOTIFY USER OF MISSED KEYBORAD
                                                        ; INPUT
```

```
                        ;--------CLEAR SHIFT STATES DONT LEAVE POSSIBILTY OF DANGLING STATES
                        ;       OF MISSED BREAKS
EEC7  80 26 0017 R F0           AND     KB_FLAG,0F0H    ; CLEAR ALT,CLRL,LEFT AND RIGHT
                                                        ; SHIFTS
EECC  80 26 0018 R 0F           AND     KB_FLAG_1,0FH   ; CLEAR POTENTIAL BREAK OF INS,CAPS
                                                        ; NUM AND SCROLL SHIFT
EED1  80 26 0088 R 1F           AND     KB_FLAG_2,1FH   ; CLEAR FUNCTION STATES
EED6  9D                J16_7: POPF                    ; GET THE FLAGS
EED7                    J17:
EED7  72 40                     JC      J20
EED9  E8 EAA0 R                 CALL    RESULTS         ; GET THE NEC STATUS
EEDC  72 38                     JC      J20             ; LOOK FOR ERROR
                        ;------ CHECK THE RESULTS RETURNED BY THE CONTROLLER
EEDE  FC                        CLD                     ; SET THE CORRECT DIRECTION
EEDF  BE 0042 R                 MOV     SI,OFFSET NEC_STATUS ; POINT TO STATUS FIELD
EEE2  AC                        LODS    NEC_STATUS      ; GET STO
EEE3  24 C0                     AND     AL,0C0H         ; TEST FOR NORMAL TERMINATION
EEE5  74 58                     JZ      J22             ; OPN_OK
EEE7  3C 40                     CMP     AL,040H         ; TEST FOR ABNORMAL TERMINATION
EEE9  75 25                     JNZ     J18             ; NOT ABNORMAL, BAD NEC
                        ;-----------------------------------------------------------------
                        ;***NOTE***
                        ;       THE CURRENT SYSTEM CONFIGURATION HAS NO DMA.  IN ORDER TO
                        ;       STOP THE NEC AN EOT MUST BE PASSED TO FORCE THE NEC TO HALT
                        ;       THEREFORE, THE STATUS RETURNED BY THE NEC WILL ALWAYS SHOW
                        ;       AN EOT ERROR. IF THIS IS THE ONLY ERROR RETURNED AND THE
                        ;       NUMBER OF SECTORS TRANSFERRED EQUALS THE NUMBER SECTORS
                        ;       REQUESTED IN THIS INTERRUPT CALL THEN THE OPERATION HAS
                        ;       COMPLETED SUCCESSFULLY.  IF AN EOT ERROR IS RETURNED AND THE
                        ;       REQUESTED NUMBER OF SECTORS IS NOT THE NUMBER OF SECTORS
                        ;       TRANSFERRED THEN THE ERROR IS LEGITIMATE.  WHEN THE EOT
                        ;       ERROR IS INVALID THE STATUS BYTES RETURNED ARE UPDATED TO
                        ;       REFLECT THE STATUS OF THE OPERATION IF DMA HAD BEEN PRESENT
                        ;-----------------------------------------------------------------
EEEB  AC                        LODS    NEC_STATUS      ; GET ST1
EEEC  3C 80                     CMP     AL,80H          ; IS THIS THE ONLY ERROR?
EEEE  74 2A                     JE      J21_1           ; NORMAL TERMINATION, NO ERROR
EEF0  D0 E0                     SAL     AL,1            ; NOT EOT ERROR, BYPASS ERROR BITS
EEF2  D0 E0                     SAL     AL,1
EEF4  D0 E0                     SAL     AL,1            ; TEST FOR CRC ERROR
EEF6  B4 10                     MOV     AH,BAD_CRC
EEF8  72 18                     JC      J19             ; RW_FAIL
EEFA  D0 E0                     SAL     AL,1            ; TEST FOR DMA OVERRUN
EEFC  B4 08                     MOV     AH,BAD_DMA
EEFE  72 12                     JC      J19             ; RW_FAIL
EF00  D0 E0                     SAL     AL,1
EF02  D0 E0                     SAL     AL,1            ; TEST FOR RECORD NOT FOUND
EF04  B4 04                     MOV     AH,RECORD_NOT_FND
EF06  72 0A                     JC      J19             ; RW_FAIL
EF08  D0 E0                     SAL     AL,1
EF0A  D0 E0                     SAL     AL,1            ; TEST MISSING ADDRESS MARK
EF0C  B4 02                     MOV     AH,BAD_ADDR_MARK
EF0E  72 02                     JC      J19             ; RW_FAIL
                        ;------ NEC MUST HAVE FAILED
EF10                    J18:                            ; RW-NEC-FAIL
EF10  B4 20                     MOV     AH,BAD_NEC
EF12                    J19:                            ; RW-FAIL
EF12  08 26 0041 R              OR      DISKETTE_STATUS,AH
EF16  E8 EAE1 R                 CALL    NUM_TRANS       ; HOW MANY WERE REALLY TRANSFERRED
EF19                    J20:                            ; RW_ERR
EF19  C3                        RET                     ; RETURN TO CALLER
                        ;------ OPERATION WAS SUCCESSFUL
EF1A                    J21_1:
EF1A  8A 5E 0E                  MOV     BL,[BP+14]      ; GET NUMBER OF SECTORS PASSED
                                                        ; FROM STACK
EF1D  E8 EAE1 R                 CALL    NUM_TRANS       ; HOW MANY GOT MOVED, AL CONTAINS
                                                        ; NUM OF SECTORS
EF20  3A D8                     CMP     BL,AL           ; NUMBER REQUESTED=NUMBER ACTUALLY
                                                        ; TRANSFERRED?
EF22  74 0C                     JE      J21_2           ; TRANSFER SUCCESSFUL
                        ;------ OPERATION ATTEMPTED TO ACCESS DATA PAST REAL EOT.  THIS IS
                        ;       A REAL ERROR
EF24  80 0E 0041 R 04           OR      DISKETTE_STATUS ,RECORD_NOT_FND
EF29  C6 06 0043 R 80           MOV     NEC_STATUS+1,80H ; ST1 GETS CORRECT VALUE
EF2E  F9                        STC
EF2F  C3                        RET
EF30  33 C0             J21_2:  XOR     AX,AX           ; CLEAR AX FOR NEC_STATUS UPDATE
EF32  33 F6                     XOR     SI,SI           ; INDEX TO NEC_STATUS ARRAY
EF34  88 84 0042 R              MOV     NEC_STATUS[SI],AL ; ZERO OUT BYTE, STO
EF38  46                        INC     SI              ; POINT INDEX AT SECOND BYTE
EF39  88 84 0042 R              MOV     NEC_STATUS[SI],AL ; ZERO OUT BUYE, ST1
EF3D  EB 03                     JMP     SHORT J21_3     ; OPN_OK
EF3F  E8 EAE1 R         J22:    CALL    NUM_TRANS
EF42  32 E4             J21_3:  XOR     AH,AH                   ; NO ERRORS
EF44  C3                        RET
EF45                    RW_OPN  ENDP
                        ;--------------------------------------------
                        ; DISK_INT
                        ;       THIS ROUTINE HANDLES THE DISKETTE INTERRUPT. AN INTERRUPT
                        ;       WILL OCCUR ONLY WHEN THE ONE-SHOT TIMER IS FIRED.  THIS
                        ;       OCCURS IN AN ERROR SITUATION.  THIS ROUTINE SETS ERRORS IN
                        ;       THE DISKETTE STATUS BYTE AND DISABLES THE ONE-SHOT TIMER.
                        ;       THEN THE RETURN ADDRESS ON THE STACK IS CHANGED TO RETURN
                        ;       TO THE OP_END LABEL.
                        ; INPUT
                        ;       NONE.
                        ; OUTPUT
                        ;       NONE.  DS POINTS AT BIOS DATA AREA.  CARRY FLAG IS SET SO
                        ;       THAT ERROR WILL BE CAUGHT IN THE ENVIRONMEMT RETURNED TO.
                        ;--------------------------------------------
```

```
EF57                            ORG     0EF57H
EF57                    DISK_INT        PROC    FAR
EF57  1E                        PUSH    DS
EF58  50                        PUSH    AX
EF59  52                        PUSH    DX              ; SAVE REGISTER
EF5A  55                        PUSH    BP              ; SAVE THE BP REGISTER
EF5B  E8 1388 R                 CALL    DDS             ; SETUP DS TO POINT AT BIOS DATA
                                ;------ CHECK IF INTERRUPT OCCURED IN INT13 OR WHETHER IT IS A
                                ;       SPURIOUS INTERRUPT
EF5E  8B EC                     MOV     BP,SP           ; POINT BP AT STACK
EF60  0E                        PUSH    CS              ; WAS IT IN THE BIOS AREA
EF61  58                        POP     AX
EF62  3B 46 0A                  CMP     AX,WORD PTR[BP+10] ; GET INTERRUPTED SEGMENT
EF65  75 48                     JNE     DI3             ; NOT IN BIOS, ERROR CONDITION
EF67  8B 46 08                  MOV     AX,WORD PTR[BP+8] ; GET IP ON THE STACK
EF6A  3D EE20 R                 CMP     AX,OFFSET VERIFY_LOOP ; RANGE CHECK IP FOR DISK
                                                        ; TRANSFER
EF6D  7C 40                     JL      DI3             ; BELOW TRANSFER CODE
EF6F  3D EE66 R                 CMP     AX,OFFSET OP_END+1 ; UPPER RANGE OF TRANSFER CODE
EF72  7D 3B                     JGE     DI3             ; ABOVE RANGE OF WATCHDOG TERRAIN
                                ;-------VALID DISKETTE INTERRUPT CHANGE RETURN ADDRESS ON STACK TO
                                ;       PULL OUT OF LOOP
EF74  C7 46 08 EE65 R           MOV     WORD PTR[BP+8],OFFSET OP_END
EF79  81 4E 0C 0001             OR      WORD PTR[BP+12],1 ; TURN ON CARRY FLAG IN FLAGS ON
                                                        ; STACK
                                ;--------------------------------------------------------------
                                ;***NOTE***
                                ; A WRITE PROTECTED DISKETTE WILL ALWAYS GET STUCK IN WRITE LOOP
                                ; WAITING FOR BEGINNING OF EXECUTION PHASE.  WHEN THE WATCHDOG
                                ; FIRES AND THE STATUS IN PORT NEC_STAT = DXH (X MEANS DON'T CARE)
                                ; STATUS FROM THE RESULT PHASE IS AVAILABLE.  THE STATUS IS READ
                                ; AND WRITE PROTECT IS  CHECKED FOR.
                                ;--------------------------------------------------------------
EF7E  BA 00F4                   MOV     DX,NEC_STAT
EF81  EC                        IN      AL,DX           ; GET NEC STATUS BYTE
EF82  24 F0                     AND     AL,0F0H         ; MASK HIGH NIBBLE
EF84  3C D0                     CMP     AL,0D0H         ; IS EXECUTION PHASE DONE
EF86  75 14                     JNE     DI1             ; STUCK IN LOOP
EF88  E8 EAA0 R                 CALL    RESULTS         ; GET STATUS OF OPERATION
EF8B  BE 0042 R                 MOV     SI,OFFSET NEC_STATUS ; ADDRESS OF BYTES RETURNED BY
                                                        ; NEC
EF8E  8A 44 01                  MOV     AL,[SI+1]       ; GET ST1
EF91  A8 02                     TEST    AL,02H          ; WRITE PROTECT SIGNAL ACTIVE?
EF93  74 07                     JZ      DI1             ; TIME OUT ERROR
EF95  80 0E 0041 R 03           OR      DISKETTE_STATUS,WRITE_PROTECT
EF9A  EB 13                     JMP     SHORT DI3
                                ;------TIME OUT ERROR
EF9C  80 0E 0041 R 80           DI1:    OR      DISKETTE_STATUS,TIME_OUT
EFA1  C6 06 003E R 00           MOV     SEEK_STATUS,0   ; SET RECAL ON DRIVES
                                ;------ RESET THE NEC AND DISABLE WATCHDOG
EFA6  BA 00F2                   DI2:    MOV     DX,NEC_CTL      ; ADDRESS TO NEC CONTROL PORT
EFA9  5D                        POP     BP              ; POINT BP AT BASE OF STACKED
                                                        ; PARAMETERS
EFAA  E8 EB45 R                 CALL    GET_DRIVE       ; RESET ADAPTER AND DISABLE WD
EFAD  55                        PUSH    BP              ; RESTORE FOR RETURNED CALL
EFAE  EE                        OUT     DX,AL
EFAF  B0 20                     DI3:    MOV     AL,EOI          ; GIVE EOI TO 8259
EFB1  E6 20                     OUT     INTA00,AL
EFB3  5D                        POP     BP
EFB4  5A                        POP     DX
EFB5  58                        POP     AX
EFB6  1F                        POP     DS
EFB7  CF                        IRET                    ; RETURN FROM INTERRUPT
EFB8                    DISK_INT        ENDP
                                ;--------------------------------------------------
                                ; DISK_BASE
                                ;  THIS IS THE SET OF PARAMETERS REQUIRED FOR
                                ;  DISKETTE OPERATION.  THEY ARE POINTED AT BY THE
                                ;  DATA VARIABLE DISK_POINTER.  TO MODIFY THE PARAMETERS,
                                ;  BUILD ANOTHER PARAMETER BLOCK AND POINT AT IT
                                ;--------------------------------------------------
EFC7                            ORG     0EFC7H
EFC7                    DISK_BASE       LABEL   BYTE
EFC7  CF                        DB      11001111B       ; SRT=C, HD UNLOAD=0F - 1ST SPECIFY
                                                        ; BYTE
EFC8  03                        DB      3               ; HD LOAD=1, MODE=NO 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)
```