intel

APPLICATION NOTE

AP-62

November 1979

A Low Cost CRT Terminal Using The 8275

John Katausky
Peripherals Applications

1. INTRODUCTION

The purpose of this application note is to provide the reader with the design concepts and factual tools needed to integrate Intel peripherals and microprocessors into a low cost raster scan CRT terminal. A previously published application note, AP-32, presented one possible solution to the CRT design question. This application note expands upon the theme established in AP-32 and demonstrates how to design a functional CRT terminal while keeping the parts count to a minimum.

For convenience, this application note is divided into seven general sections:

1. Introduction
2. CRT Basics
3. 8275 Description
4. Design Background
5. Circuit Description
6. Software Description
7. Appendix

There is no question that microprocessors and LSI peripherals have had a significant role in the evolution of CRT terminals. Microprocessors have allowed design engineers to incorporate an abundance of sophisticated features into terminals that were previously mere slaves to a larger processor. To complement microprocessors, LSI peripherals have reduced component count in many support areas. A typical LSI peripheral easily replaces between 30 and 70 SSI and MSI packages, and offers features and flexibility that are usually not available in most hardware designs. In addition to replacing a whole circuit board of random logic, LSI circuits also reduce the cost and increase the reliability of design. Fewer interconnects increases mechanical reliability and fewer parts decreases the power consumption and hence, the overall reliability of the design. The reduction of components also yields a circuit that is easier to debug during the actual manufacturing phase of a product.

Until the era of advanced LSI circuitry, a typical CRT terminal consisted of 80 to 200 or more SSI and MSI packages. The first microprocessors and peripherals dropped this component count to between 30 and 50 packages. This application note describes a CRT terminal that uses 20 packages.

2. CRT BASICS

The raster scan display gets its name from the fact that the image displayed on the CRT is built up by generating a series of lines (raster) across the face of the CRT. Usually, the beam starts in the upper left hand corner of the display and simultaneously moves left to right and top to bottom to put a series of zig-zag lines on the screen (Fig. 2.1). Two simultaneously operating independent circuits control the vertical and horizontal movement of the beam.

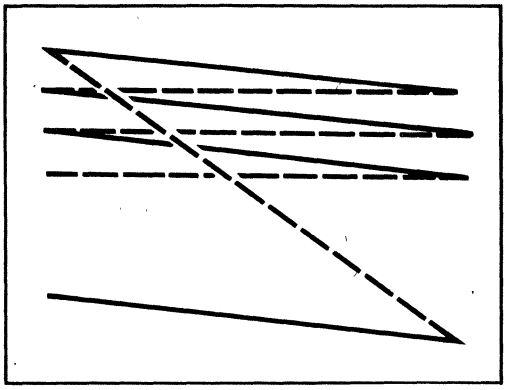


Figure 2-1. Raster Scan

As the electron beam moves across the face of the CRT, a third circuit controls the current flowing in the beam. By varying the current in the electron beam the image on the CRT can be made to be as bright or as dark as the user desires. This allows any desired pattern to be displayed.

When the beam reaches the end of a line, it is brought back to the beginning of the next line at a rate that is much faster than was used to generate the line. This action is referred to as "retrace". During the retrace period the electron beam is usually shut off so that it doesn't appear on the screen.

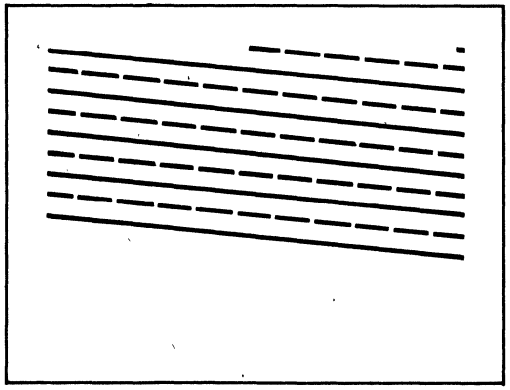
As the electron beam is moving across the screen horizontally, it is also moving downward. Because of this, each successive line starts slightly below the previous line. When the beam finally reaches the bottom right hand corner of the screen, it retraces vertically back to the top left hand corner. The time it takes for the beam to move from the top of the screen to the bottom and back again to the top is usually referred to as a "frame". In the United States, commercial television broadcast use 15,750 Hz as the horizontal sweep frequency (63.5 microseconds per horizontal line) and 60 Hz as the vertical sweep frequency or "frame" (16.67 milliseconds per vertical frame).

Although, the 60 Hz vertical frame and the 15,750 Hz horizontal line are the standards used by commercial broadcasts, they are by no means the only frequency at which CRT's can operate. In fact, many CRT displays use a horizontal scan that is around 18 KHz to 22 KHz and some even exceed 30 KHz. As the

horizontal frequency increases, the number of horizontal lines per frame increases. Hence, the resolution on the vertical axis increases. This increased resolution is needed on high density graphic displays and on special text editing terminals that display many lines of text on the CRT.

Although many CRT's operate at non-standard horizontal frequencies, very few operate at vertical frequencies other than 60 Hz. If a vertical frequency other than 60 Hz is chosen, any external or internal magnetic or electrical variations at 60 Hz will modulate the electron beam and the image on the screen will be unstable. Since, in the United States, the power line frequency happens to be 60 Hz, there is a good chance for 60 Hz interference to exist. Transformers can cause 60 Hz magnetic fields and power supply ripple can cause 60 Hz electrical variations. To overcome this, special shielding and power supply regulation must be employed. In this design, we will assume a standard frame rate of 60 Hz and a standard line rate of 15,750 Hz.

By dividing the 63.5 microsecond horizontal line rate into the 16.67 millisecond vertical rate, it is found that there are 262.5 horizontal lines per vertical frame. At first, the half line may seem a bit odd, but actually it allows the resolution on the CRT to be effectively doubled. This is done by inserting a second set of horizontal lines between the first set (interlacing). In an interlaced system the line sets are not generated simultaneously. In a 60 Hz system, first all of the even-numbered lines are scanned: 0, 2, 4,...524. Then all the odd-numbered lines: 1, 3, 5,... 525. Each set of lines usually contains different data (Fig. 2.2).



EVEN FIELD
ODD FIELD
RETRACE LINES NOT SHOWN

Figure 2-2. Interlaced Scan

Although interlacing provides greater resolution, it also has some distinct disadvantages. First of all, the circuitry needed to generate the extra half horizontal line per frame is quite complex when compared to a noninterlaced design, which requires an integer number of horizontal lines per frame. Next, the overall vertical refresh rate is half that of a noninterlaced display. As a result, flicker may result when the CRT uses high speed phosphors. To keep things as simple as possible, this design uses the noninterlaced approach.

The first thing any CRT controller must do is generate pulses that define the horizontal line timing and the vertical frame timing. This is usually done by dividing a crystal reference source by some appropriate numbers. On most raster scan CRT's the horizontal frequency is very forgiving and can vary by around 500 Hz or so and produce no ill effects. This means that the CRT itself can track a horizontal frequency between 15250 Hz and 16250 Hz, or in other words, there can be 256 to 270 horizontal lines per vertical frame. But, as mentioned earlier, the vertical frequency should be 60 Hz to insure stability.

The characters that are viewed on the screen are formed by a series of dots that are shifted out of the controller while the electron beam moves across the face of the CRT. The circuits that create this timing are referred to as the dot clock and character clock. The character clock is equal to the dot clock divided by the number of dots used to form a character along the horizontal axis and the dot clock is calculated by the following equation:

DOT CLOCK (Hz) = (N + R) * D * L * F

where N is the number of displayed characters per row,

R is the number of retrace character time increments,

D is the number of dots per character,

L is the number of horizontal lines per frame and

F is the frame rate in Hz.

In this design N = 80, R = 20, D = 7, L = 270, and F = 60 Hz. If the numbers are plugged in, the dot clock is found to be 11.34 MHz.

The retrace number, R, may vary from system to system because it is used to establish the margins on the left and right hand sides of the CRT. In this particular design R = 20 was empirically found it be optimum. The number of dots per character may vary depending on the character generator used and the number of dot clocks the designer wants to place between characters. This design uses a 5 X 7 dot matrix and allows 2 dot clock periods between characters (see Fig. 2.3); since 5 + 2 equals 7, we find that D = 7.

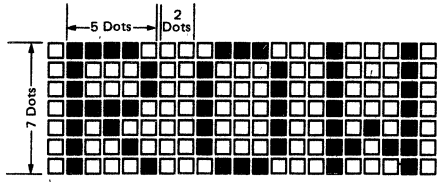


Figure 2-3. 5 X 7 Dot Matrix

The number of lines per frame can be determined by the following equation:

L = (H * Z) + V

where, H is the number of horizontal lines per character,

Z is the number of character lines per frame and

V is the number of horizontal lines during vertical retrace. In this design, a 5 X 7 dot matrix is to be placed on a 7 X 10 field, so H = 10. Also, 25 lines are to be displayed, so Z = 25. As mentioned before, V = 20. When the numbers are plugged into the equation, L is found to be equal to 270 lines per frame.

The designer should be cautioned that these numbers are interrelated and that to guarantee proper operation on a standard raster scan CRT, L should be between 256 and 270. If L does not lie within these bounds the horizontal circuits of the CRT may not be able to lock onto the driving signal and the image will roll horizontally. The chosen L of 270 yields a horizontal frequency of 16,200 KHz on a 60 Hz frame and this number is within the 500 Hz tolerance mentioned earlier.

The V number is chosen to match the CRT in much the same manner as the R number mentioned earlier. When the electron beam reaches the bottom right corner of the screen it must retrace vertically to the top left corner. This retrace action requires time, usually between 900-1200 microseconds. To allow for this, enough horizontal sync times must be inserted during vertical retrace. Twenty horizontal sync times at 61.5 microseconds yield a total of 1234.5 microseconds, which is enough time to allow the beam to return to the top of the screen.

The choices of H and Z largely relate to system design preference. As H increases, the character size along the vertical axis increases. Z is simply the number of lines of characters that are displayed and this, of course, is entirely a system design option.

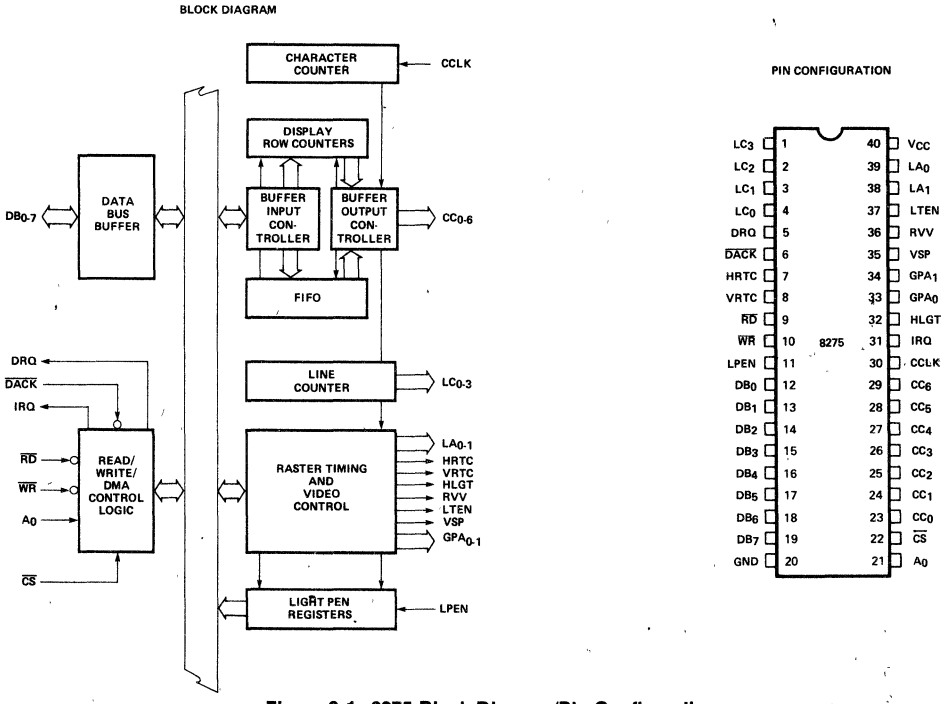


Figure 3-1. 8275 Block Diagram/Pin Configuration

3. 8275 DESCRIPTION

A block diagram and pin configuration of the 8275 are shown in Fig. 3.1. The following is a description of the general capabilities of the 8275.

3.1 CRT DISPLAY REFRESHING

The 8275, having been programmed by the designer to a specific screen format, generates a series of DMA request signals, resulting in the transfer of a row of characters from display memory to the 8275's row buffers. The 8275 presents the character codes to an external character generator ROM by using outputs CCO-CC6. External dot timing logic is then used to transfer the parallel output data from the character generator ROM serially to the video input of the CRT. The character rows are displayed on the CRT one line at a time. Line count outputs LC0-LC3 are applied to the character generator ROM to perform the line selection function. The display process is illustrated in Figure 3.2. The entire process is repeated for each display row. At the beginning of the last displayed row, the 8275 issues an interrupt by setting the IRQ output line. The 8275 interrupt output will normally be connected to the interrupt input of the system central processor.

The interrupt causes the CPU to execute an interrupt service subroutine. The service subroutine typically re-initializes DMA controller parameters for the next display refresh cycle, polls the system keyboard controller, and/or executes other appropriate functions. A block diagram of a CRT system implemented with the 8275 CRT Controller is provided in Figure 3.3. Proper CRT refreshing requires that certain 8275 parameters be programmed prior to the beginning of display operation. The 8275 has two types of programming registers, the Command Registers (CREG) and the Parameter Registers (PREG). It also has a Status Register (SREG). The Command Registers may only be written to and the Status Registers may only be read. The 8275 expects to receive a command followed by a sequence of from 0 to 4 parameters, depending on the command. The 8275 instruction set consist of the eight commands shown in Figure 3.4.

To establish the format of the display, the 8275 provides a number of user programmable display format parameters. Display formats having from 1 to 80 characters per row, 1 to 64 rows per screen, and 1 to 16 horizontal lines per row are available.

In addition to transferring characters from memory

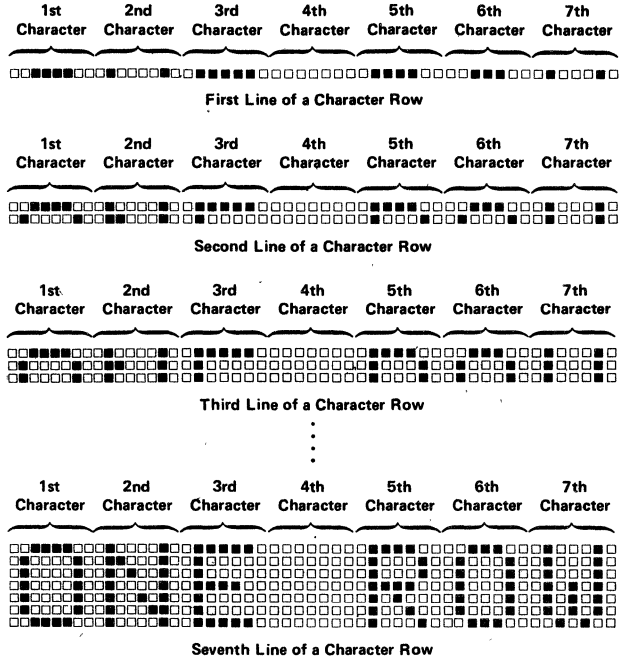


Figure 3-2. 8275 Row Display

AFN-01304A

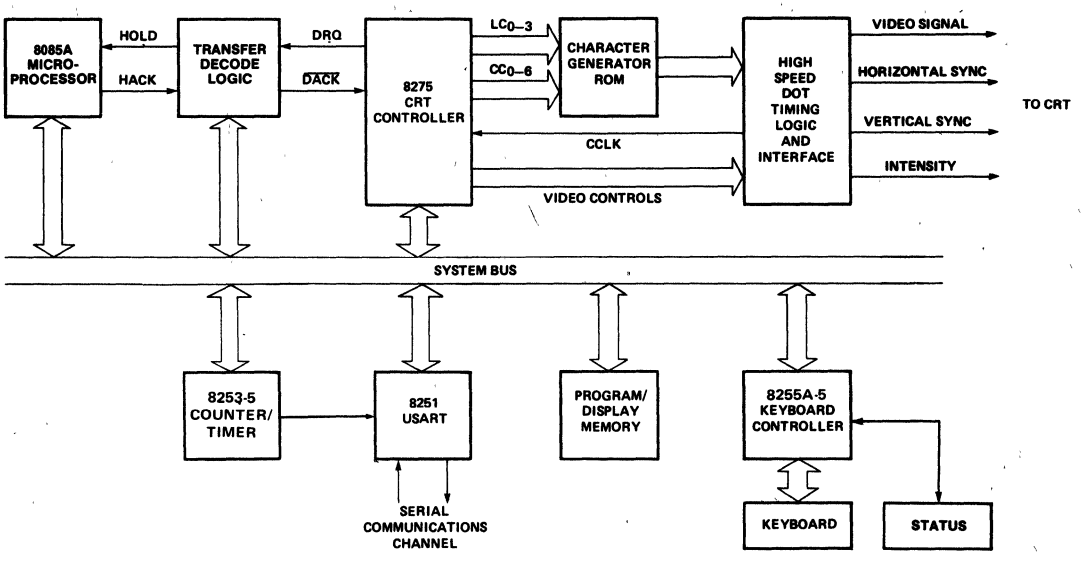


Figure 3-3. CRT System Block Diagram

to the CRT screen, the 8275 features cursor position control. The cursor position may be programmed, via X and Y cursor position registers, to any character position on the display. The user may select from four cursor formats. Blinking or non-blinking underline and reverse video block cursors are available.

3.2 CRT TIMING

The 8275 provides two timing outputs, HRTC and VRTC, which are utilized in synchronizing CRT horizontal and vertical oscillators to the 8275 refresh cycle. In addition, whenever HRTC or VRTC is active, a third timing output, VSP (Video Suppress) is true, providing a blinking signal to the dot timing logic. The dot timing logic will normally inhibit the video output to the CRT during the time when video suppress signal is true. An additional timing output, LTEN (Light Enable) is used to provide the ability to force the video output high regardless of the state of VSP. This feature is used by the 8275 to place a cursor on the screen and to control attribute functions. Attributes will be considered in the next section.

The HLGT (Highlight) output allows an attribute function to increase the CRT beam intensity to a level greater than normal. The fifth timing signal, RVV (Reverse Video) will, when enabled, cause the system video output to be inverted.

COMMAND	NO. OF PARAMETER BYTES	NOTES
RESET	4	Display format parameters required
START DISPLAY	0	DMA operation parameters included in command
STOP DISPLAY	0	—
READ LIGHT PEN	2	—
LOAD CURSOR	2	Cursor X,Y position parameters required
ENABLE INTERRUPT	0	—
DISABLE INTERRUPT	0	—
PRESET COUNTERS	0	Clears all internal counters

Figure 3-4. 8275's Instruction Set

Character attributes were designed to produce the following graphics:

CHARACTER ATTRIBUTE CODE "CCCC"		OUTPUTS LA_1	LA_0	VSP	LTEN	SYMBOL	DESCRIPTION
0000	Above Underline	0	0	1	0		Top Left Corner
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0001	Above Underline	0	0	1	0		Top Right Corner
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0010	Above Underline	0	1	0	0		Bottom Left Corner
	Underline	1	0	0	0		
	Below Underline	0	0	1	0		
0011	Above Underline	0	1	0	0		Bottom Right Corner
	Underline	1	1	0	0		
	Below Underline	0	0	1	0		
0100	Above Underline	0	0	1	0		Top Intersect
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
0101	Above Underline	0	1	0	0		Right Intersect
	Underline	1	1	0	0		
	Below Underline	0	1	0	0		
0110	Above Underline	0	1	0	0		Left Intersect
	Underline	1	0	0	0		
	Below Underline	0	1	0	0		
0111	Above Underline	0	1	0	0		Bottom Intersect
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1000	Above Underline	0	0	1	0		Horizontal Line
	Underline	0	0	0	1		
	Below Underline	0	0	1	0		
1001	Above Underline	0	1	0	0		Vertical Line
	Underline	0	1	0	0		
	Below Underline	0	1	0	0		
1010	Above Underline	0	1	0	0		Crossed Lines
	Underline	0	0	0	1		
	Below Underline	0	1	0	0		
1011	Above Underline	0	0	0	0		Not Recommended *
	Underline	0	0	0	0		
	Below Underline	0	0	0	0		
1100	Above Underline	0	0	1	0		Special Codes
	Underline	0	0	1	0		
	Below Underline	0	0	1	0		
1101	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1110	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						
1111	Above Underline						Illegal
	Underline		Undefined				
	Below Underline						

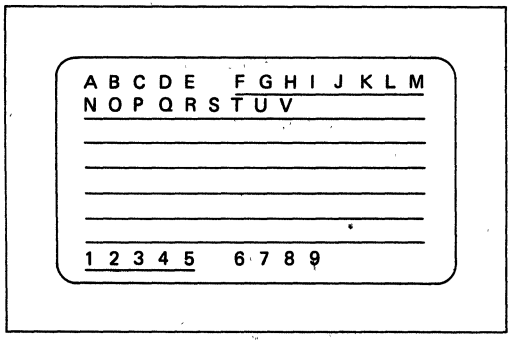
*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character Generator will not be disabled, and an indeterminate character will be generated.

Character Attribute Codes 1101, 1110, and 1111 are illegal.

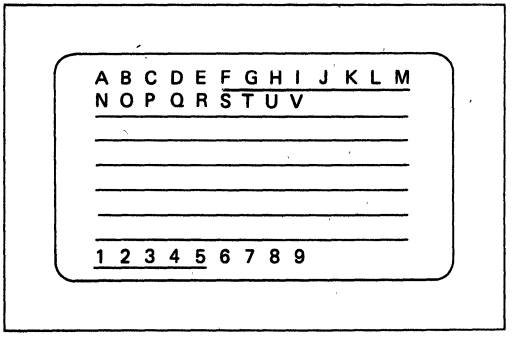
Blinking is active when B = 1.

Highlight is active when H = 1.

Figure 3-5. Character Attributes



EXAMPLE OF THE VISIBLE FIELD ATTRIBUTE MODE (UNDERLINE ATTRIBUTE)



EXAMPLE OF THE INVISIBLE FIELD ATTRIBUTE MODE (UNDERLINE ATTRIBUTE)

Figure 3-6. Field Attribute Examples

3.3 SPECIAL FUNCTIONS

VISUAL ATTRIBUTES—Visual attributes are special codes which, when retrieved from display memory by the 8275, affect the visual characteristics of a character position or field of characters. Two types of visual attributes exist, character attributes and field attributes.

Character Attribute Codes: Character attribute codes can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LAO-LA1), the Video Suppression output (VSP), and the Light Enable output (LTEN). The dot timing logic uses these signals to generate the proper symbols. Character attributes can be programmed to blink or be highlighted individually. Blinking is accomplished with the Video Suppression output (VSP). Blink frequency is equal to the screen refresh frequency divided by 32. Highlighting is accomplished by activating the Highlight output (HGLT). Character attributes were designed to produce the graphic symbols shown in Figure 3.5.

Field Attribute Codes: The field attributes are control codes which affect the visual characteristics for a field of characters, starting at the character following the field attribute code up to, and including, the character which precedes the next field attribute code, or up to the end of the frame.

There are six field attributes:

1. *Blink* — Characters following the code are caused to blink by activating the Video Suppression output (VSP). The blink frequency is equal to the screen refresh frequency divided by 32.
2. *Highlight* — Characters following the code are caused to be highlighted by activating the Highlight output (HGLT).
3. *Reverse Video* — Characters following the code are caused to appear in reverse video format by activating the Reverse Video output (RVV).
4. *Underline* — Characters following the code are caused to be underlined by activating the Light Enable output (LTEN).
5. *General Purpose* — There are two additional 8275 outputs which act as general purpose, independently programmable field attributes. These attributes may be used to select colors or perform other desired control functions.

The 8275 can be programmed to provide visible or invisible field attribute characters as shown in Figure 3.6. If the 8275 is programmed in the visible field attribute mode, all field attributes will occupy a position on the screen. They will appear as blanks caused by activation of the Video Suppression output (VSP). The chosen visual attributes are activated after this blanked character. If the 8275 is programmed in the invisible field attribute mode, the 8275 row buffer FIFOs are activated. The FIFOs effectively lengthen the row buffers by 16 characters, making room for up to 16 field attribute characters per display row. The FIFOs are 126 characters by 7 bits in size. When a field attribute is placed in the row buffer during DMA, the buffer input controller recognizes it and places the next character in the proper FIFO. When a field attribute is placed in the buffer output controller during display, it causes the controller to immediately put a character from the FIFO on the Character Code outputs (CCO-6). The chosen attributes are also activated.

AFN-01304A

LIGHT PEN DETECTION — A light pen consists fundamentally of a switch and light sensor. When the light pen is pressed against the CRT screen, the switch enables the light sensor. When the raster sweep coincides with the light sensor position on the display, the light pen output is input and the row and character position coordinates are stored in two 8275 internal registers. These registers can be read by the microprocessor.

SPECIAL CODES — Four special codes may be used to help reduce memory, software, or DMA overhead. These codes are placed in character positions in display memory.

1. *End Of Row Code* - Activates VSP. VSP remains active until the end of the line is reached. While VSP is active, the screen is blanked.
2. *End Of Row-Stop DMA Code* - Causes the DMA Control Logic to stop DMA for the rest of the row when it is written into the row buffer. It affects the display in the same way as the End of Row Code.
3. *End Of Screen Code* - Activates VSP. VSP remains active until the end of the frame is reached.
4. *End Of Screen-Stop DMA Code* - Causes the DMA Control Logic to stop DMA for the rest of the frame when it is written into the row buffer. It affects the display in the same way as the End of Screen Code.

PROGRAMMABLE DMA BURST CONTROL — The 8275 can be programmed to request single-byte DMA transfers of DMA burst transfers of 2, 4, or 8 characters per burst. The interval between bursts is also programmable. This allows the user to tailor the DMA overhead to fit the system needs.

4. DESIGN BACKGROUND

4.1 DESIGN PHILOSOPHY

Since the cost of any CRT system is somewhat proportional to parts count, arriving at a minimum part count solution without sacrificing performance has been the motivating force throughout this design effort. To successfully design a CRT terminal and keep the parts count to a minimum, a few things became immediately apparent.

1. An 8085 should be used.
2. Address and data buffering should be eliminated.
3. Multi-port memory should be eliminated.
4. DMA should be eliminated.

Decision 1 is obvious, the 8085's on-board clock generator, bus controller and vectored interrupts greatly reduce the overall part count considerably. Decision 2 is fairly obvious; if a circuit can be designed so that loading on the data and address lines is kept to a minimum, both the data and address buffers can be eliminated. This easily saves three to eight packages and reduces the power consumption of the design. Both decisions 3 and 4 require a basic understanding of current CRT design concepts.

In any CRT design, extreme time conflicts are created because all essential elements require access to the bus. The CPU needs to access the memory to control the system and to handle the incoming characters, but, at the same time, the CRT controller needs to access the memory to keep the raster scan display refreshed. To resolve this conflict two common techniques are employed, page buffering and line buffering.

In the page buffering approach the entire screen memory is isolated from the rest of the system. This isolation is usually accomplished with three-state buffers or two line to one line multiplexers. Of course, whenever a character needs to be manipulated the CPU must gain access to the buffered memory and, again, possible contention between the CPU and the CRT controller results. This contention is usually resolved in one of two ways, (1) the CPU is always given priority, or; (2) the CPU is allowed to access the buffered memory only during horizontal and vertical retrace times.

Approach 1 is the easiest to implement from a hardware point of view, but if the CPU always has priority the display may temporarily blink or "flicker" while the CPU accesses the display memory. This, of course, occurs because when the CPU accesses the display memory the CRT controller is not able to retrieve a character, so the display must be blanked during this time. Aesethically, this "flickering" is not desirable, so approach 2 is often used.

The second approach eliminates the display flickering encountered in the previously mentioned technique, but additional hardware is required. Usually the vertical and horizontal blank signals are gated with the buffered memory select lines and this line is used to control the CPU's ready line. So, if the CPU wants to use the buffered memory, its ready line is asserted until horizontal or vertical retrace times. This, of course, will impact the CPU's overall through put.

Both page buffered approaches require a significant amount of additional hardware and for the most part are not well suited for a minimum parts count type of terminal. This guides us to the line buffered approach. This approach eliminates the separate buffered memory for the display, but, at the same time, introduces a few new problems that must be solved.

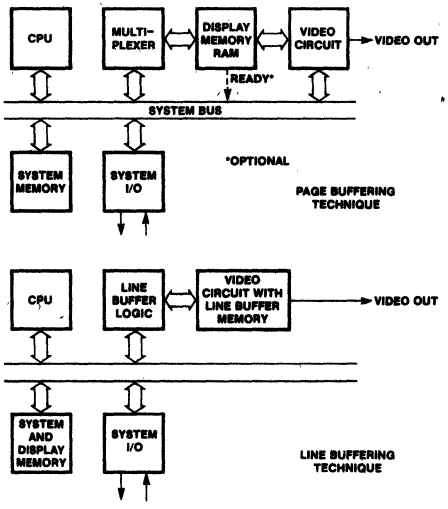


Figure 4-1. Line Buffering Technique

```
CLOCK CYCLES    SEQ     SOURCE STATEMENT

    10          1       PUSH    PSW             ;SAVE A AND FLAGS
    10          2       PUSH    H               ;SAVE H AND L
    10          3       PUSH    D               ;SAVE D AND E
    10          4       LXI     H,0080H         ;ZERO H AND L
    10          5       DAD     SP              ;PUT STACK POINTER IN H AND L
     4          6       XCHG                    ;PUT STACK IN D AND E
    16          7       LHLD    CURAD           ;GET POINTER
     6          8       SPHL                    ;PUT CURRENT LINE INTO SP
     7          9       MVI     A,0C0H          ;SET MASK FOR SIM
     4          10      SIM                     ;SET SPECIAL TRANSFER BIT
   400          11      POP     H               ;DO 40 POPS
     4          12      RRC                     ;SET UP A
     4          13      SIM                     ;GO BACK TO NORMAL MODE
    10          14      LXI     H,0000H         ;ZERO HL
    10          15      DAD     SP              ;ADD STACK
     4          16      XCHG                    ;PUT STACK IN H AND L
     6          17      SPHL                    ;RESTORE STACK
    10          18      LXI     H,LAST          ;PUT BOTTOM DISPLAY IN H AND L
     4          19      XCHG                    ;SWAP REGISTERS
     4          20      MOV     A,D             ;PUT HIGH ORDER IN A
     4          21      CMP     H               ;SEE IF SAME AS H
  7/10          22      JNZ     KPTK            ;IF NOT LEAVE
     4          23      MOV     A,E             ;PUT LOW ORDER IN A
     4          24      CMP     L               ;SEE IF SAME AS L
  7/10          25      JNZ     KPTK            ;IF NOT LEAVE
    10          26      LXI     H,TPDIS         ;LOAD H AND L WITH TOP OF SCREEN MEMORY
    16          27 KPTK: SHLD   CURAD           ;PUT BACK CURRENT ADDRESS
     7          28      MVI     A,18H           ;GET MASK BYTE
     4          29      SIM                     ;SET INTERRUPT MASK
    10          30      POP     D               ;GET D AND E
    10          31      POP     H               ;GET H AND L
    10          32      POP     PSW             ;GET A AND FLAGS
     4          33      EI                      ;ENABLE INTERRUPTS
    10          34      RET                     ;GO BACK

    TOTAL CLOCK CYCLES = 650 (WORST CASE)

    WITH A 6 144 MHZ CRYSTAL TOTAL TIME TO FILL

    ROW BUFFER ON 8275 = 650 * .325 = 211.25 MICROSECONDS
```

Figure 4-2. Routine To Load 8275's Row Buffers

In the line buffered approach both the CPU and the CRT controller share the same memory. Every time the CRT controller needs a new character or line of data, normal processing activity is halted and the CRT controller accesses memory and displays the data. Just how the CRT controller needs to acquire the display data greatly affects the performance of the overall system. Whether the CRT controller needs to gain access to the main memory to acquire a single character or a complete line of data depends on the presence or absence of a separate line or row buffer.

If no row buffer is present the CRT controller must go to the main memory to fetch every character. This of course, is not a very efficient approach because the processor will be forced to relinquish the bus 70% to 80% of the time. So much processor inactivity greatly affects the overall system performance. In fact terminals that use this approach are typically limited to around 1200 to 2400 baud on their serial communication channels. This low baud rate is in general not acceptable, hence this approach was not chosen.

If a separate row buffer is employed the CRT controller only has to access the memory once for each displayed character per line. This forces the processor to relinquish the bus only about 20% to 35% of the time and a full 4800 to 9600 baud can be achieved. Figure 4.1 illustrates these different techniques.

The 8275 CRT controller is ideal for implementing the row buffer approach because the row buffer is contained on the device itself. In fact, the 8275 contains two 80-byte row buffers. The presence of two row buffers allow one buffer to be filled while the other buffer is displaying the data. This dual row buffer approach enhances CPU performance even further.

4.2 USING THE 8275 WITHOUT DMA

Until now the process of filling the row buffer has only been alluded to. In reality, a DMA technique is usually used. This approach was demonstrated in AP-32 where an 8257 DMA controller was mated to an 8275 CRT controller. In order to minimize component count, this design eliminates the DMA controller and its associated circuitry while replacing them with a special interrupt-driven transfer.

The only real concern with using the 8275 in an interrupt-driven transfer mode is speed. Eighty characters must be loaded into the 8275 every 617 microseconds and the processor must also have time to perform all the other tasks that are required. To minimize the overhead associated with loading the characters into the 8275 a special technique was employed. This technique involves setting a special

transfer bit and executing a string of POP instructions. The string of POP instructions is used to rapidly move the data from the memory into the 8275. Figure 4.2 shows the basic software structure.

In this design the 8085's SOD line was used as the special transfer bit. In order to perform the transfer properly this special bit must do two things: (1) turn processor reads into $\overline{DACK}$ plus $\overline{WR}$ for the 8275 and (2) mask processor fetch cycles from the 8275, so that a fetch cycle does not write into the 8275. Conventional logic could have been used to implement this special function, but in this design a small bipolar programmable read only memory was used. Figure 4.3 shows a basic version of the hardware..

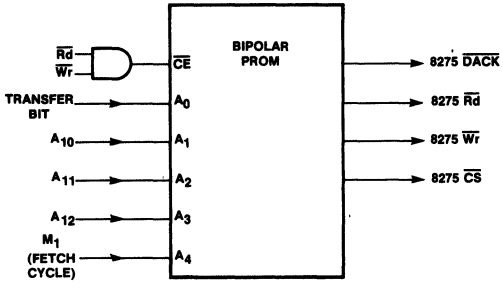


Figure 4-3. Simplified Version of Hardware Decoder

At first, it may seem strange that we are supplying a $\overline{DACK}$ when no DMA controller exist in the system. But the reader should be aware that all Intel peripheral devices that have DMA lines actually use $\overline{DACK}$ as a chip select for the data. So, when you want to write a command or read status you assert $\overline{CS}$ and $\overline{WR}$ or $\overline{RD}$, but when you want to read or write data you assert $\overline{DACK}$ and $\overline{RD}$ or $\overline{WR}$. The peripheral device doesn't "know" if a DMA controller is in the circuit or not. In passing, it should be mentioned that $\overline{DACK}$ and $\overline{CS}$ should not be asserted on the same device at the same time, since this combination yields an undefined result.

This POP technique actually compares quite favorably in terms of time to the DMA technique. One POP instruction transfers two bytes of data to the 8275 and takes 10 CPU clock cycles to execute, for a net transfer rate of one byte every five clock cycles. The DMA controller takes four clock cycles to transfer one byte but, some time is lost in synchronization. So the difference between the two techniques is one clock cycle per byte maximum. If we compare the overall speed of the 8085 to the speed of the 8080 used in AP-32, we find that at 3 MHz we can transfer one byte every 1.67 microseconds using the 8085 and POP technique vs. 2 microseconds per byte for the 2 MHz 8080 using DMA.

5. CIRCUIT DESCRIPTION

5.1 SCOPE OF THE PROJECT

A fully functional, microprocessor-based CRT terminal was designed and constructed using the 8275 CRT controller and the 8085 as the controlling element. The terminal had many of the functions found in existing commercial low-cost terminals and more sophisticated features could easily be added with a modest amount of additional software. In order to minimize component count LSI devices were used whenever possible and software was used to replace hardware.

5.2 SYSTEM TARGET SPECIFICATIONS

The design specifications for the CRT terminal were as follows:

Display Format

- 80 characters per display row
- 25 display rows

Character Format

- 5 X 7 dot matrix character contained within a 7 X 10 matrix
- First and seventh columns blanked
- Ninth line cursor position
- Blinking underline cursor

Special Characters Recognized

- Control characters
- Line feed
- Carriage Return
- Backspace
- Form feed

Escape Sequences Recognized

- ESC, A, Cursor up
- ESC, B, Cursor down
- ESC, C, Cursor right
- ESC, D, Cursor left
- ESC, E, Clear screen
- ESC, H, Home cursor
- ESC, J, Erase to the end of the screen
- ESC, K, Erase the current line

Characters Displayed

- 96 ASCII alphanumeric characters
- Special control characters

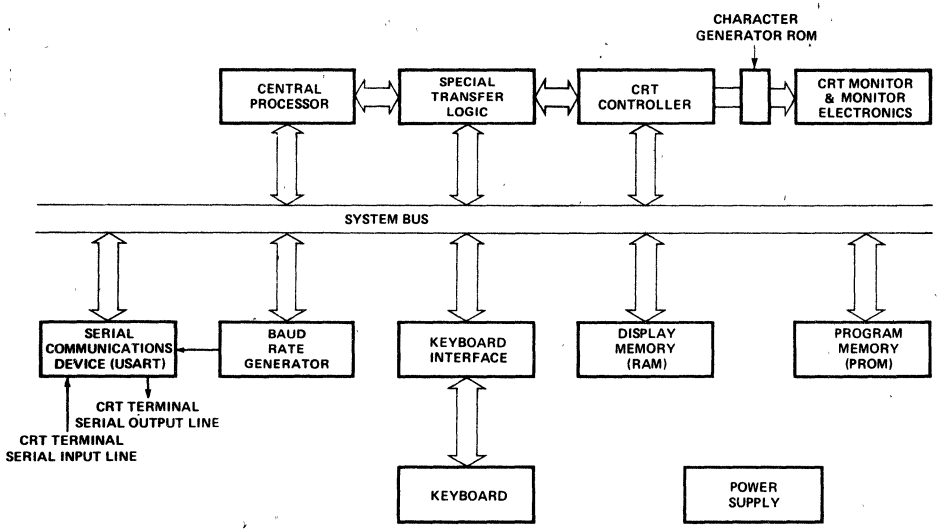


Figure 5-1. CRT Terminal Block Diagram

Characters Transmitted

- 96 ASCII alphanumeric characters
- ASCII control characters

Program Memory

- 2K bytes of 2716 EPROM

Display/Buffer/Stack Memory

- 2K bytes 2114 static memory (4 packages)

Data Rate

- 9600 BAUD using 3MHz 8085

CRT Monitor

- Ball Bros TV-12, 12MHz B.W.

Keyboard

- Any standard un-encoded ASCII keyboard

Screen Refresh Rate

- 60 Hz

5.3 HARDWARE DISCRIPTION

A block diagram of the CRT terminal is shown in Figure 5.1. The diagram shows only the essential system features. A detailed schematic of the CRT is contained in the Appendix. The terminal was constructed on a simple 6" by 6" wire wrap board. Because of the minimum bus loading no buffering of any kind was needed (see Figure 5.2).

The "heart" of the CRT terminal is the 8085 microprocessor. The 8085 initializes all devices in the system, loads the CRT controller, scans the keyboard, assembles the characters to be transmitted, decodes the incoming characters and determines where the character is to be placed on the screen. Clearly, the processor is quite busy.

A standard list of LSI peripheral devices surround the 8085. The 8251A is used as the serial communication link, the 8255A-5 is used to scan the keyboard and read the system variables through a set of

Worst case bus loading:

Data Bus:	8275	20pf
	8255A-5	20pf
	8253-5	20pf
	8253-5	20pf
	8251A	20pf
	2x 2114	10pf
	2716	12pf
	8212	12pf
		114pf max

Only A_8 - A_{15} are important since A_0 - A_7 are latched by the 8212

Address Bus:	4x 2114	20pf
	2716	6pf
		26pf max

This loading assures that all components will be compatible with a 3MHz 8085 and that no wait states will be required

Figure 5-2. Bus Loading

switches, and the 8253 is used as a baud rate generator and as a "horizontal pulse extender" for the 8275.

The 8275 is used as the CRT controller in the system, and a 2716 is used as the character generator. To handle the high speed portion of the terminal the 8275 is surrounded by a small handful of TTL. The program memory is contained in one 2716 EPROM and the data and screen memory use four 2114-type RAMs.

All devices in this system are memory mapped. A bipolar PROM is used to decode all of the addresses for the RAM, ROM, 8275, and 8253. As mentioned earlier, the bipolar prom also turns $\overline{\text{READ}}$s into $\overline{\text{DACK}}$'s and $\overline{\text{WR}}$'s for the 8275. The 8255 and 8253 are decoded by a simple address line chip select method. The total package count for the system is 20, not including the serial line drivers. If this same terminal were designed using the MCS-85 family of integrated circuits, additional part savings could have been realized. The four 2114's could have been replaced by two 8185's and the 8255 and the 2716 program PROM could have been replaced by one 8755. Additionally, since both the 8185 and the 2716 have address latches no 8212 would be needed, so the total parts count could be reduced by three or four packages.

5.4 SYSTEM OPERATION

The 8085 CPU initializes each peripheral to the appropiate mode of operation following system reset. After initialization, the 8085 continually polls the 8251A to see if a character has been sent to the terminal. When a character has been received, the 8085 decodes the character and takes appropriate action. While the 8085 is executing the above "foreground" programs, it is being interrupted once every 617 microseconds by the 8275. This "background" program is used to load the row buffers on the 8275. The 8085 is also interrupted once every frame time, or 16.67 ms, to read the keyboard and the status of the 8275.

As discussed earlier, a special POP technique was used to rapidly move the contents of the display RAM into the 8275's row buffers. The characters are then synchronously transferred to the character code outputs CC0-CC6, connected to the character generator address lines A3-A9 (Figure 5.3). Line count outputs LC0-LC2 from the 8275 are applied to the character generator address lines A0-A2. The 8275 displays character rows one line at a time. The line count outputs are used to determine which line of the character selected by A3-A8 will be displayed. Following the transfer of the first line to the dot timing logic, the line count is incremented and the second line of the character row is selected. This process continues until the last line of the row is transferred to the dot timing logic.

The dot timing logic latches the output of the character generator ROM into a parallel in,serial out synchronous shift register. This shift register is clocked at the dot clock rate (11.34 MHz) and its output constitutes the video input to the CRT.

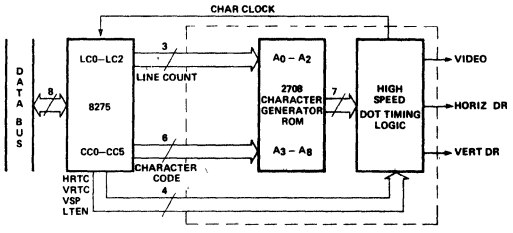


Figure 5-3 Character Generator/Dot Timing Logic Block Diagram

Table 5-1

PARAMETER	RANGE
Vertical Blanking Time (VRTC)	900 μsec nominal
Vertical Drive Pulsewidth	300 μsec ⩽ PW ⩽ 1.4 ms
Horizontal Blanking Time (HRTC)	11 μsec nominal
Horizontal Drive Pulsewidth	25 μsec ⩽ PW ⩽ 30 μsec
Horizontal Repetition Rate	15,750 ±500 pps

5.5 SYSTEM TIMING

Before any specific timing can be calculated it is necessary to determine what constraints the chosen CRT places on the overall timing. The requirements for the Ball Bros. TV-12 monitor are shown in Table 5.1. The data from Table 5.1, the 8275 specifications, and the system target specifications are all that is needed to calculate the system's timing.

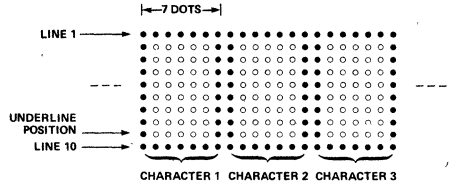


Figure 5-4. Row Format

First, let's select and "match" a few numbers. From our target specifications, we see that each character is displayed on a 7 X 10 field, and is formed by a 5 X 7 dot matrix (Figure 5.4). The 8275 allows the vertical retrace time to be only an integer multiple of the horizontal character line. This means that the total number of horizontal lines in a frame equals 10 times the number of character lines plus the vertical retrace time, which is programmed to be either 1, 2, 3, or 4 character lines. Twenty-five display lines

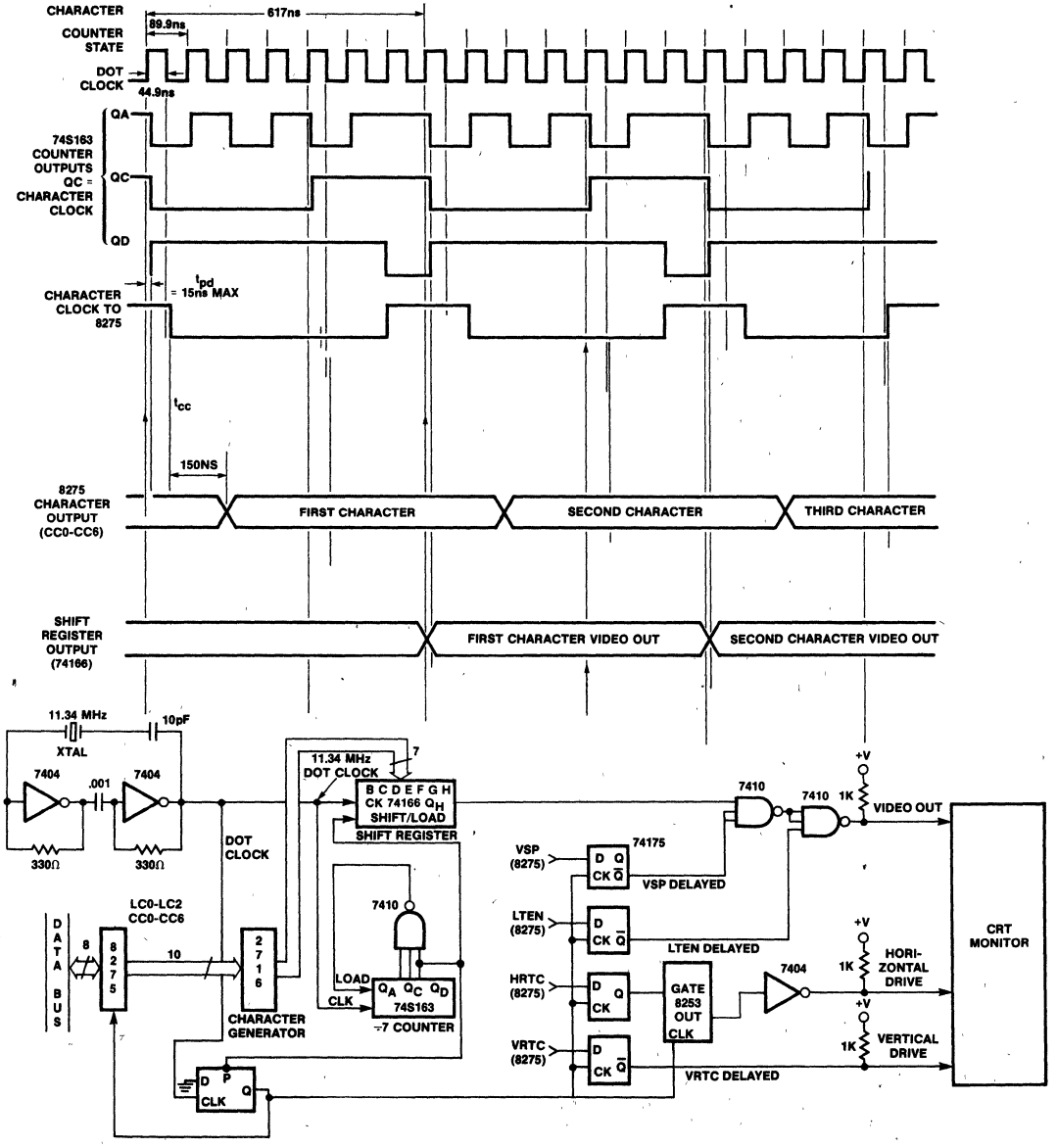


Figure 5-5. Dot Timing Logic

require 250 horizontal lines. So, if we wish to have a horizontal frequency in the neighborhood of 15,750 Hz we must choose either one or two character lines for vertical retrace. To allow for a little more margin at the top and bottom of the screen, two character lines were chosen for vertical retrace. This choice yields a net 250 + 20 = 270 horizontal lines per frame. So, assuming a 60 Hz frame:

60 Hz * 270 = 16,200 Hz (horizontal frequency)

This value falls within our target specification of 15,750 Hz with a 500 Hz variation and also assures timing compatibility with the Ball monitor since, 20 horizontal sync times yield a vertical retract time of:

61.7 microseconds X 20 horizontal sync times = 1.2345 milliseconds

This number meets the nominal VRTC and vertical drive pulse width time for the Ball monitor. A horizontal frequency of 16,200 Hz implies a 1/16,200 = 61.73 microsecond period.

It is now known that the terminal is using 250 horizontal lines to display data and 20 horizontal lines to allow for vertical retrace and that the horizontal frequency is 16,200 Hz. The next thing that needs to be determined is how much time must be allowed for horizontal retrace. Unfortunately, this number depends almost entirely on the monitor used. Usually, this number lies somewhere between 15 and 30 percent of the total horizontal line time, which in this case is 1/16,200 Hz or 61.73 microseconds. Since in most designs a fixed number of characters can be displayed on a horizontal line, it is often useful to express retrace as a given number of character times. In this design, 80 characters can be displayed on a horizontal line and it was empirically found that allowing 20 horizontal character times for retrace gave the best results. So, in reality, there are 100 character times in every given horizontal line, 80 are used to display characters and 20 are used to allow for retrace. It should be noted that if too many character times are used for retrace, less time will be left to display the characters and the display will not "fill out" the screen. Conversely, if not enough character times are allowed for retrace, the display may "run off" the screen.

One hundred character times per complete horizontal line means that each character requires

61.73 microseconds / 100 character times = 617.3 nanoseconds.

If we multiply the 20 horizontal retrace times by the

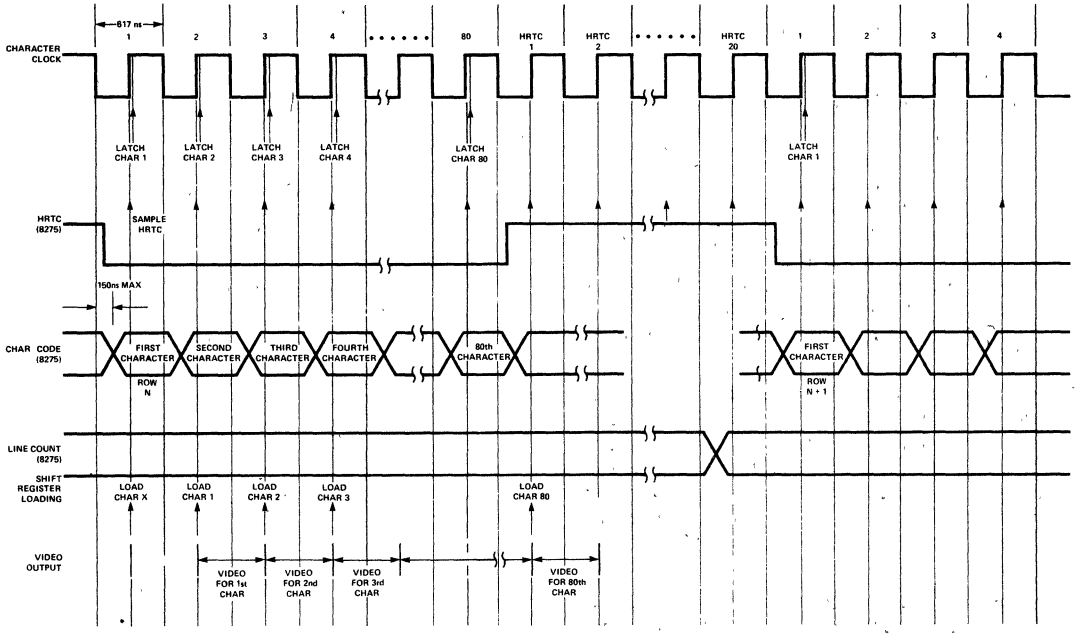


Figure 5-6. CRT System Timing

617.3 nanoseconds needed for each character, we find

617.3 nanoseconds * 20 retrace times = 12.345 microseconds

This value falls short of the 25 to 30 microseconds required by the horizontal drive of the Ball monitor. To correct for this, an 8253 was programmed in the one-shot mode and was used to extend the horizontal drive pulsewidth.

Now that the 617.3 nanosecond character clock period is known, the dot clock is easy to calculate. Since each character is formed by placing 7 dots along the horizontal.

DOT CLOCK PERIOD = 617.3 ns (CHARACTER CLK PERIOD)/ 7 DOTS

DOT CLOCK PERIOD = 88.183 nanoseconds

DOT CLOCK FREQUENCY = 1/PERIOD = 11.34 MHz

Figures 5.5 and 5.6 illustrate the basic dot timing and the CRT system timing, respectively.

6. SYSTEM SOFTWARE

6.1 SOFTWARE OVERVIEW

As mentioned earlier the software is structured on a "foreground-background" basis. Two interrupt-driven routines, FRAME and POPDAT (Fig. 6.1) request service every 16.67 milliseconds and 617 microseconds respectively, frame is used to check the baud rate switches, update the system pointers and decode and assemble the keyboard characters. POPDAT is used to move data from the memory into the 8275's row buffer rapidly.

The foreground routine first examines the line-local switch to see whether to accept data from the USART or the keyboard. If the terminal is in the local mode, action will be taken on any data that is entered through the keyboard and the USART will be ignored on both output and input. If the terminal is in the line mode data entered through the keyboard will be transmitted by the USART and action will be taken on any data read out of the USART.

When data has been entered in the terminal the software first determines if the character received was an escape, line feed, form feed, carriage return, back space, or simply a printable character. If an escape was received the terminal assumes the next received character will be a recognizable escape sequence character. If it isn't no operation is performed.

After the character is decoded, the processor jumps to the routine to perform the required task. Figure 6.2 is a flow chart of the basic software operations; the program is listed in Appendix 6.8.

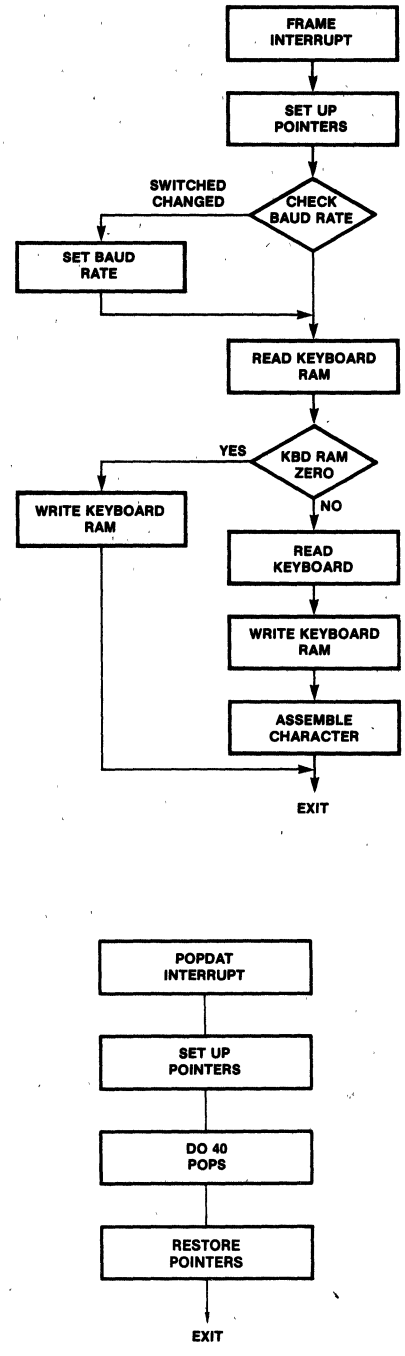


Figure 6-1. Frame and Popdat Interrupt Routines

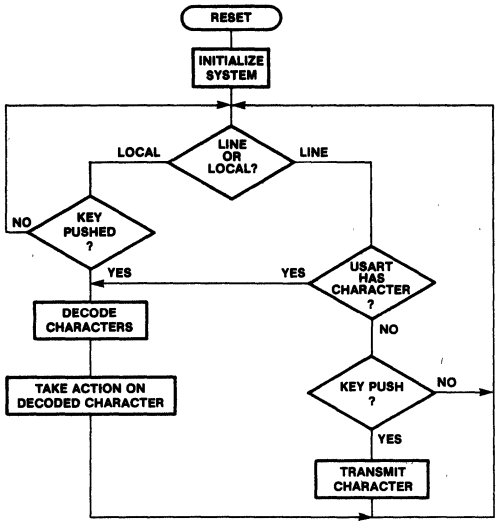


Figure 6-2. Basic Terminal Software

	1st Column	2nd Column ..... 80th Column
ROW 1	0800H	0801H 084FH
ROW 2	0850H	0851H 089FH
ROW 3	08A0H	08A1H 08EFH
ROW 4	08F0H	08F1H 093FH
ROW 5	0940H	0941H 098FH
ROW 6	0990H	0991H 090FH
ROW 7	09E0H	09E1H..... 0A2FH
ROW 8	0A30H	0A31H 0A7FH
ROW 9	0A80H	0A81H 0ACFH
ROW 10	0AD0H	0AD1H 0B1FH
ROW 11	0B20H	0B21H 0B6FH
ROW 12	0B70H	0B71H 0BBFH
ROW 13	0BC0H	0BC1H 0C0FH
ROW 14	0C10H	0C11H 0C5FH
ROW 15	0C60H	0C61H 0CAFH
ROW 16	0CB0H	0CB1H 0CFFH
ROW 17	0D00H	0D01H 0D4FH
ROW 18	0D50H	0D51H 0D9FH
ROW 19	0DA0H	0DA1H 0DEFH
ROW 20	0DF0H	0DF1H 0E3FH
ROW 21	0E40H	0E41H.....0E8FH
ROW 22	0E90H	0E91H.....0EDFH
ROW 23	0EE0H	0EE1H 0F2FH
ROW 24	0F30H	0F31H.....0F7FH
ROW 25	0F80H	0F81H.....0FCFH

Figure 6-3. Screen Display After Initialization

6.2 SYSTEM MEMORY ORGANIZATION

The display memory organization is shown in Figure 6.3. The display begins at location 0800H in memory and ends at location 0FCFH. The 48 bytes of RAM from location 0FD0H to 0FFFH are used as system stack and temporary system storage. 2K bytes of PROM located at 0000H through 07FFH contain the systems program.

6.3 MEMORY POINTERS AND SCROLLING

To calculate the location of a character on the screen, three variables must be defined. Two of these variables are the X and Y position of the cursor (CURSX, CURSY). In addition, the memory address defining the top line of the display must be known, since scrolling on the 8275 is accomplished simply by changing the pointer that loads the 8275's row buffers from memory. So, if it is desired to scroll the display up or down all that must be changed is one 16-bit memory pointer. This pointer is entered into the system by the variable TOPAD (TOP Address) and always defines the top line of the display. Figure 6.4 details screen operation during scrolling.

Subroutines CALCU (Calculate) and ADX (ADd X axis) use these three variables to calculate an absolute memory address. The subroutine CALCU is used whenever a location in the screen memory must be altered.

6.4 SOFTWARE TIMING

One important question that must be asked about the terminal software is, "How fast does it run". This is important because if the terminal is running at 9600 baud, it must be able to handle each received character in 1.04 milliseconds. Figure 6.5 is a flowchart of the subroutine execution times. It should be pointed out that all of the times listed are "worst case" execution times. This means that all routines assume they must do the maximum amount of data manipulation. For instance, the PUT routine assumes that the character is being placed in the last column and that a line feed must follow the placing of the character on the screen.

How fast do the routines need to execute in order to assure operation at 9600 baud? Since POPDAT interrupts occur every 617 microseconds, it is possible to receive two complete interrupt requests in every character time (1042 microseconds) at 9600

ROW 1	0800H	0801H 084FH
ROW 2	0850H	0851H 089FH
ROW 3	08A0H	08A1H 08EFH
ROW 4	08F0H	08F1H 093FH
ROW 5	0940H	0941H 098FH
ROW 6	0990H	0991H 090FH
ROW 7	09E0H	09E1H 0A2FH
ROW 8	0A30H	0A31H 0A7FH
ROW 9	0A80H	0A81H 0ACFH
ROW 10	0AD0H	0AD1H 0B1FH
ROW 11	0B20H	0B21H 0B6FH
ROW 12	0B70H	0B71H 0BBFH
ROW 13	0BC0H	0BC1H 0C0FH
ROW 14	0C10H	0C11H 0C5FH
ROW 15	0C60H	0C61H 0CAFH
ROW 16	0CB0H	0CB1H 0CFFH
ROW 17	0D00H	0D01H 0D4FH
ROW 18	0D50H	0D51H 0D9FH
ROW 19	0DA0H	0DA1H 0DEFH
ROW 20	0DF0H	0DF1H 0E3FH
ROW 21	0E40H	0E41H 0E8FH
ROW 22	0E90H	0E91H 0EDFH
ROW 23	0EE0H	0EE1H 0F2FH
ROW 24	0F30H	0F31H 0F7FH
ROW 25	0F80H	0F81H 0FCFH

After Initialization

ROW 2	0850H	0851H 089FH
ROW 3	08A0H	08A1H 08EFH
ROW 4	08F0H	08F1H 093FH
ROW 5	0940H	0941H 098FH
ROW 6	0990H	0991H 090FH
ROW 7	09E0H	09E1H 0A2FH
ROW 8	0A30H	0A31H 0A7FH
ROW 9	0A80H	0A81H 0ACFH
ROW 10	0AD0H	0AD1H 0B1FH
ROW 11	0B20H	0B21H 0B6FH
ROW 12	0B70H	0B71H 0BBFH
ROW 13	0BC0H	0BC1H 0C0FH
ROW 14	0C10H	0C11H 0C5FH
ROW 15	0C60H	0C61H 0CAFH
ROW 16	0CB0H	0CB1H 0CFFH
ROW 17	0D00H	0D01H 0D4FH
ROW 18	0D50H	0D51H 0D9FH
ROW 19	0DA0H	0DA1H 0DEFH
ROW 20	0DF0H	0DF1H 0E3FH
ROW 21	0E40H	0E41H 0E8FH
ROW 22	0E90H	0E91H 0EDFH
ROW 23	0EE0H	0EE1H 0F2FH
ROW 24	0F30H	0F31H 0F7FH
ROW 25	0F80H	0F81H 0FCFH
ROW 1	0800H	0801H 084FH

After 1 Scroll

ROW 3	08A0H	08A1H 08EFH
ROW 4	08F0H	08F1H 093FH
ROW 5	0940H	0941H 098FH
ROW 6	0990H	0991H 090FH
ROW 7	09E0H	09E1H 0A2FH
ROW 8	0A30H	0A31H 0A7FH
ROW 9	0A80H	0A81H 0ACFH
ROW 10	0AD0H	0AD1H 0B1FH
ROW 11	0B20H	0B21H 0B6FH
ROW 12	0B70H	0B71H 0BBFH
ROW 13	0BC0H	0BC1H 0C0FH
ROW 14	0C10H	0C11H 0C5FH
ROW 15	0C60H	0C61H 0CAFH
ROW 16	0CB0H	0CB1H 0CFFH
ROW 17	0D00H	0D01H 0D4FH
ROW 18	0D50H	0D51H 0D9FH
ROW 19	0DA0H	0DA1H 0DEFH
ROW 20	0DF0H	0DF1H 0E3FH
ROW 21	0E40H	0E41H 0E8FH
ROW 22	0E90H	0E91H 0EDFH
ROW 23	0EE0H	0EE1H 0F2FH
ROW 24	0F30H	0F31H 0F7FH
ROW 25	0F80H	0F81H 0FCFH
ROW 1	0800H	0801H 084FH
ROW 2	0850H	0851H 089FH

After 2 Scrolls

ROW 4	08F0H	08F1H 093FH
ROW 5	0940H	0941H 098FH
ROW 6	0990H	0991H 090FH
ROW 7	09E0H	09E1H 0A2FH
ROW 8	0A30H	0A31H 0A7FH
ROW 9	0A80H	0A81H 0ACFH
ROW 10	0AD0H	0AD1H 0B1FH
ROW 11	0B20H	0B21H 0B6FH
ROW 12	0B70H	0B71H 0BBFH
ROW 13	0BC0H	0BC1H 0C0FH
ROW 14	0C10H	0C11H 0C5FH
ROW 15	0C60H	0C61H 0CAFH
ROW 16	0CB0H	0CB1H 0CFFH
ROW 17	0D00H	0D01H 0D4FH
ROW 18	0D50H	0D51H 0D9FH
ROW 19	0DA0H	0DA1H 0DEFH
ROW 20	0DF0H	0DF1H 0E3FH
ROW 21	0E40H	0E41H 0E8FH
ROW 22	0E90H	0E91H 0EDFH
ROW 23	0EE0H	0EE1H 0F2FH
ROW 24	0F30H	0F31H 0F7FH
ROW 25	0F80H	0F81H 0FCFH
ROW 1	0800H	0801H 084FH
ROW 2	0850H	0851H 089FH
ROW 3	08A0H	08A1H 08EFH

After 3 Scrolls

Figure 6-4. Screen Memory During Scrolling

baud. Each POPDAT interrupt executes in 211 microseconds maximum. This means that each routine must execute in:

1042 - 2 * 211 = 620 microseconds

By adding up the times for any loop, it is clear that all routines meet this speed requirement, with the exception of ESC J. This means that if the terminal is operating at 9600 baud, at least one character time must be inserted after an ESC J sequence.

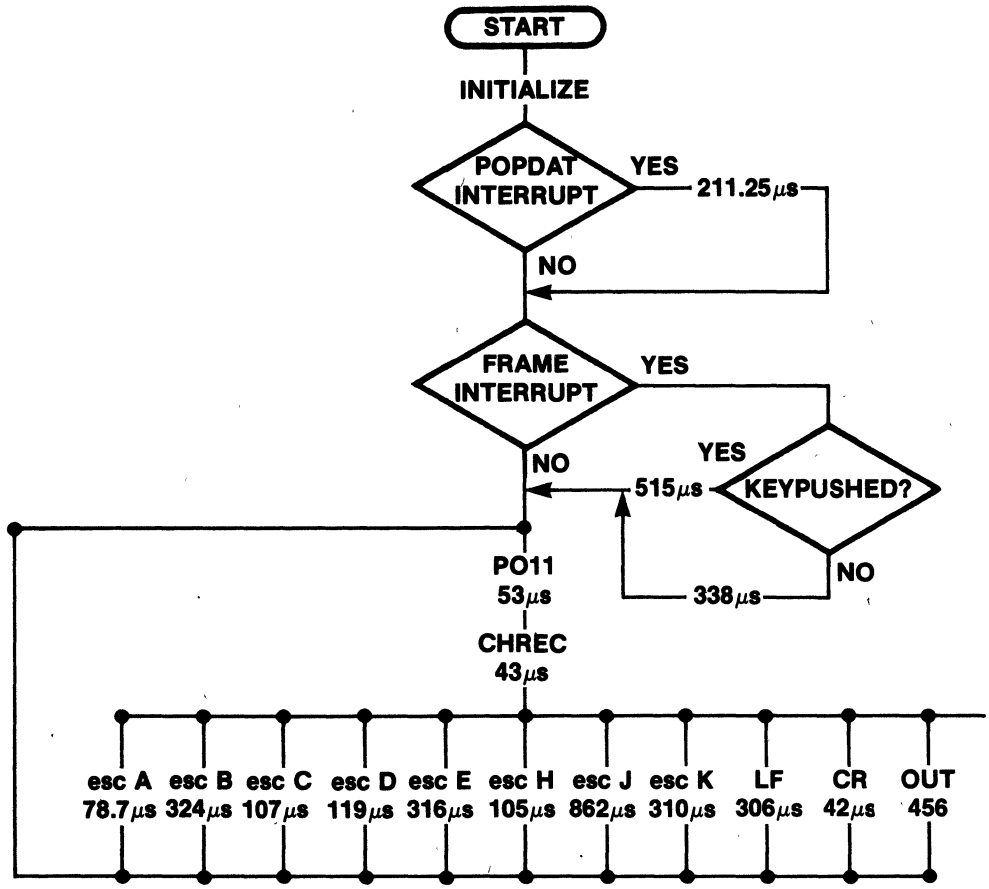
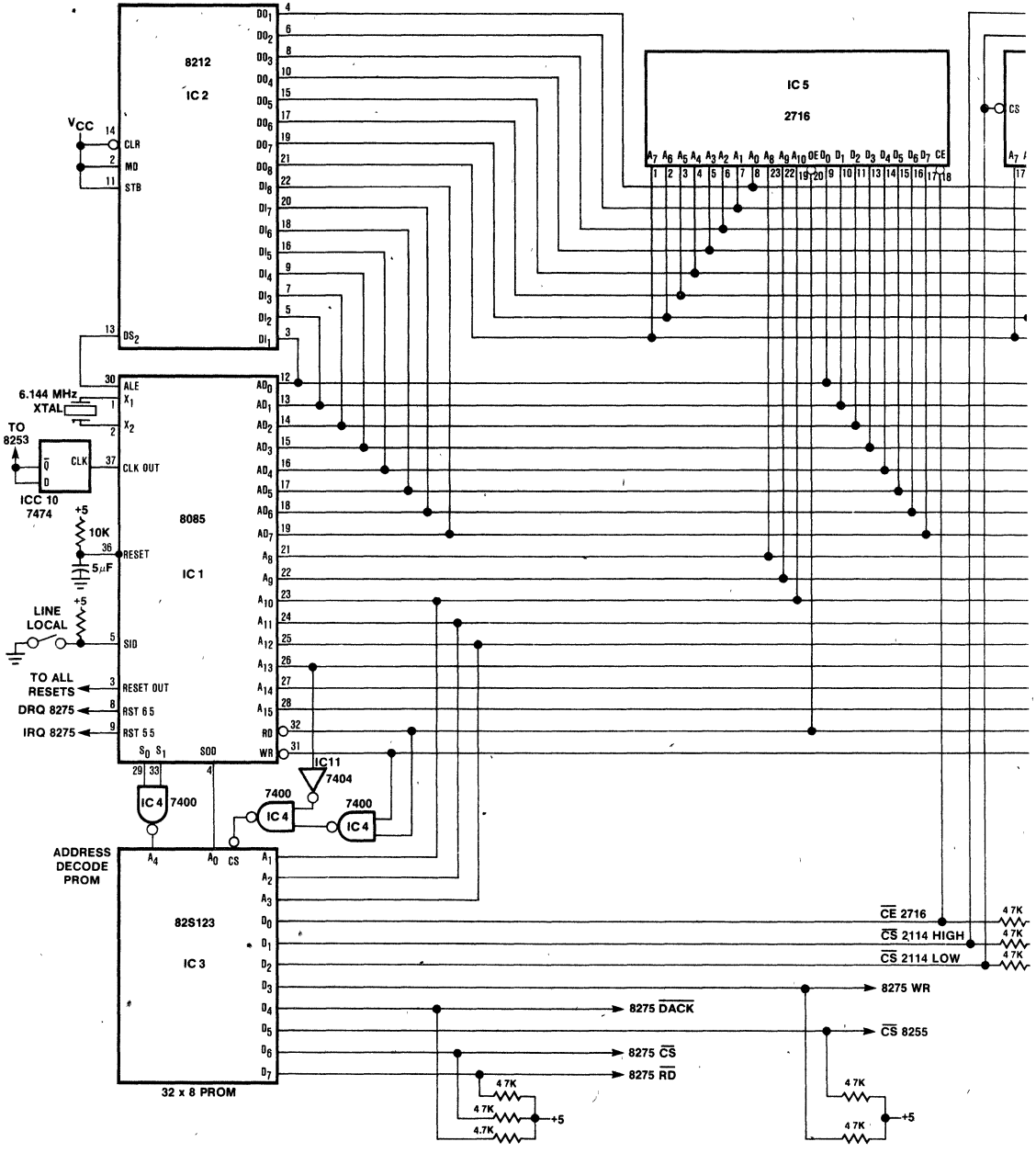
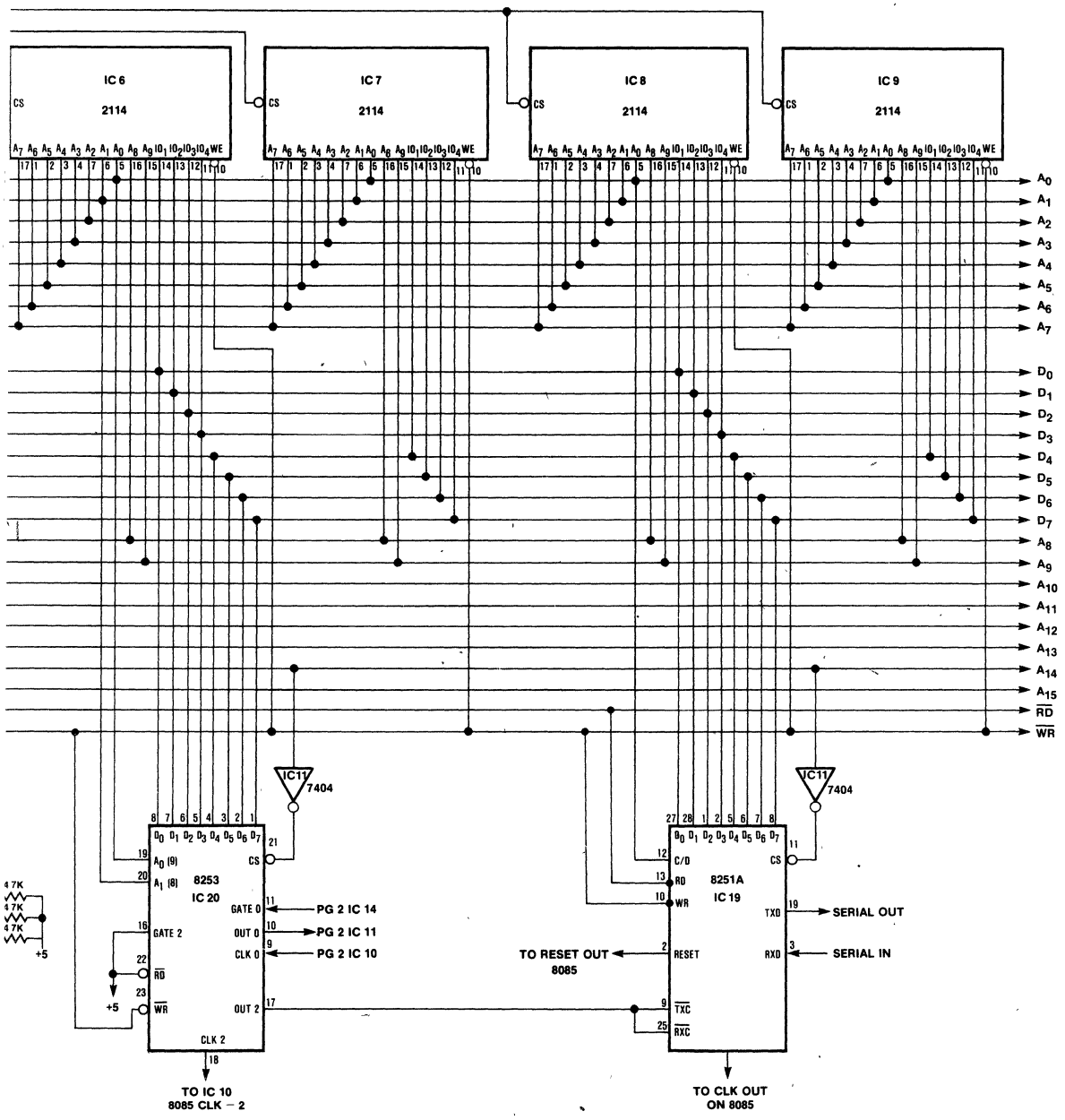


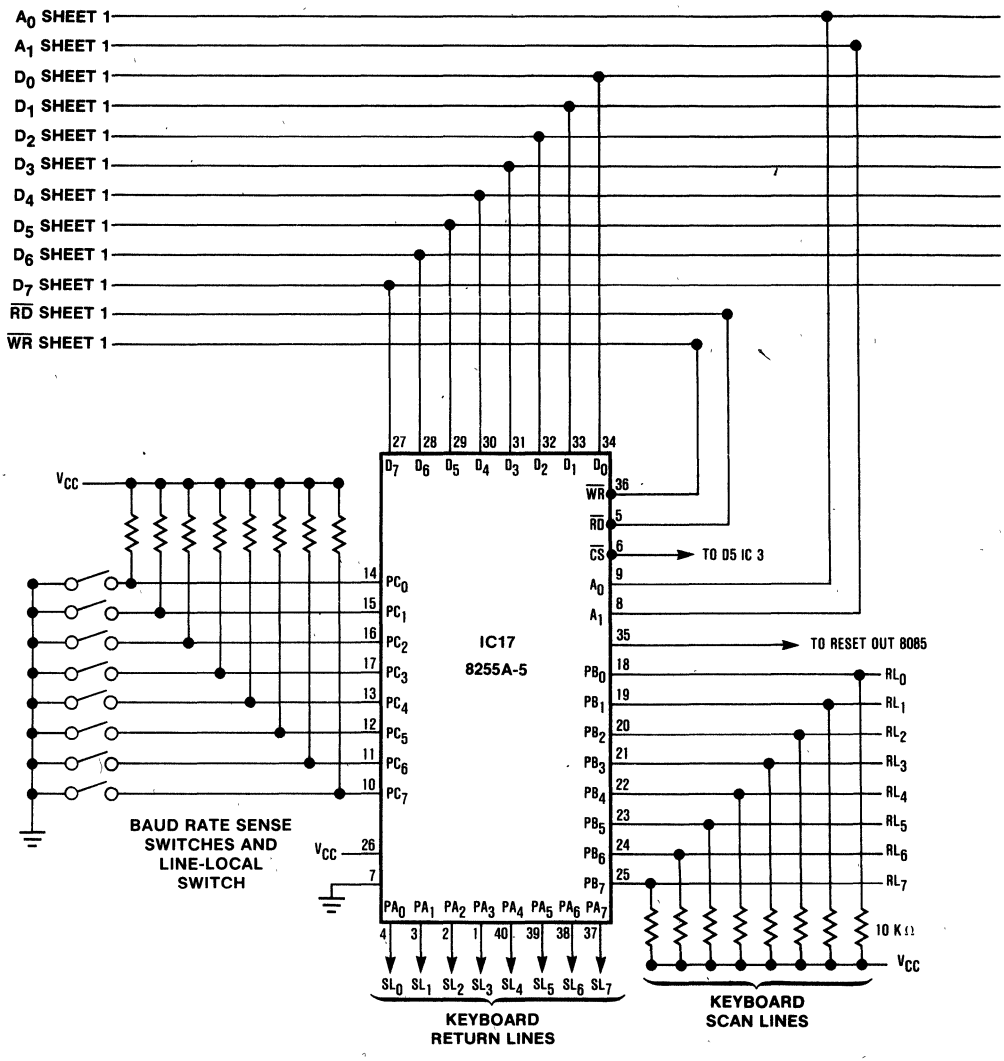
Figure 6-5. Timing Flowchart



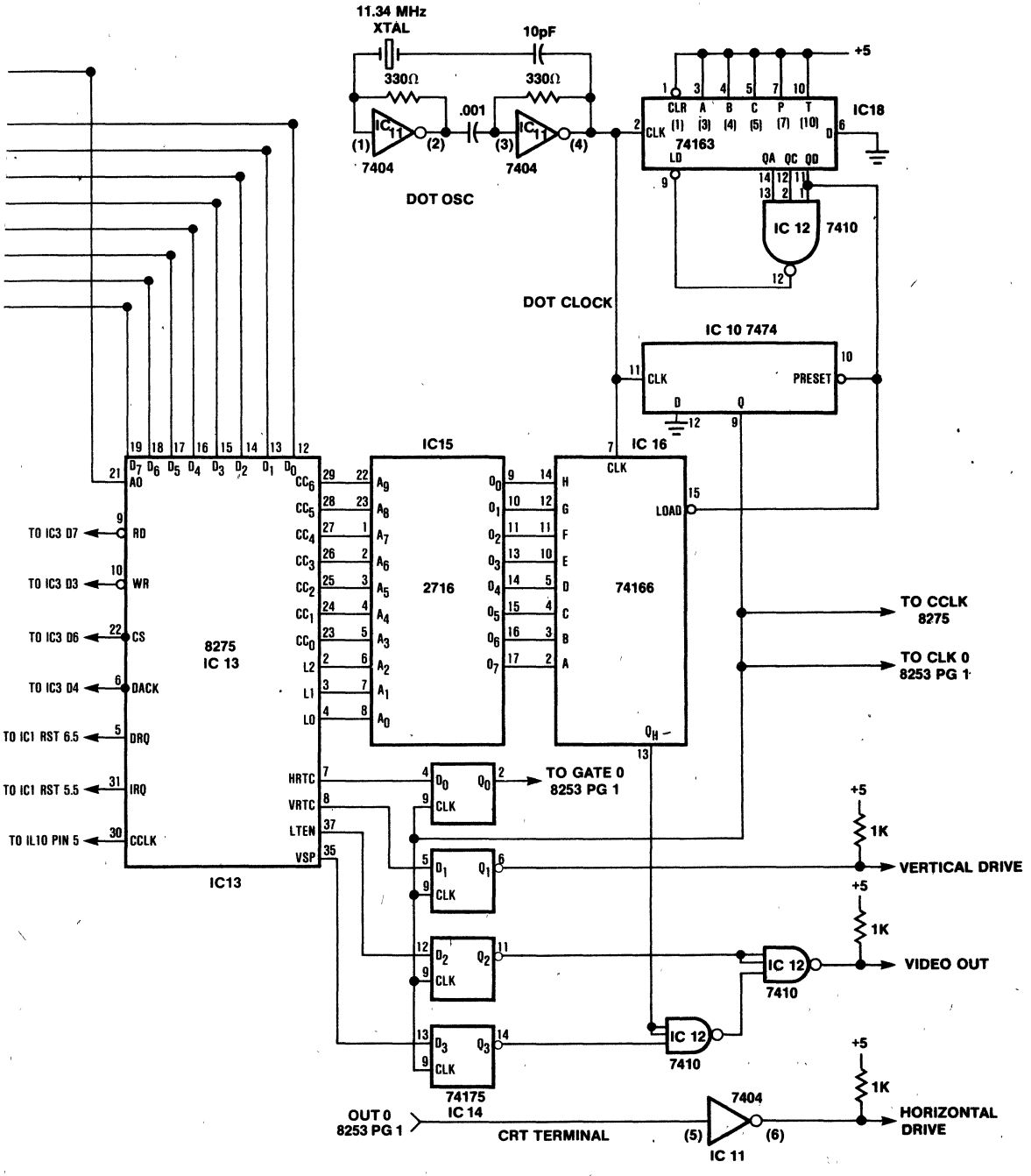
Appendix 7.1
CRT TERMINAL SCHEMATICS

APPLICATIONS





Appendix 7.1
CRT TERMINAL SCHEMATICS


11.34 MHz
XTAL
10pF
330Ω
330Ω
.001
IC 11
(1)
(2)
(3)
(4)
7404
7404
DOT OSC
+5
IC18
CLR A B C P T
(1) (3) (4) (5) (7) (10)
CLK
74163
LD
QA QC QD
IC 12 7410
DOT CLOCK
IC 10 7474
CLK
PRESET
D
Q
IC15
IC 16
2716
74166
CLK
LOAD
8275
IC 13
TO IC3 D7
TO IC3 D3
TO IC3 D6
TO IC3 D4
TO IC1 RST 6.5
TO IC1 RST 5.5
TO IL10 PIN 5
RD
WR
CS
DACK
DRQ
IRQ
CCLK
HRTC
VRTC
LTEN
VSP
IC13
TO CCLK
8275
TO CLK 0
8253 PG 1
TO GATE 0
8253 PG 1
1K
VERTICAL DRIVE
VIDEO OUT
74175
IC 14
OUT 0
8253 PG 1
CRT TERMINAL
HORIZONTAL
DRIVE

Appendix 7.2
KEYBOARD INTERFACE

The keyboard used in this design was a simple unencoded ASCII keyboard. In order to keep the cost to a minimum a simple scan matrix technique was implemented by using two ports of an 8255 parallel I/O device.

When the system is initialized the contents of the eight keyboard RAM locations are set to zero. Once every frame, which is 16.67 milliseconds the contents of the keyboard ram is read and then rewritten with the contents of the current switch matrix. If a non-zero value of one of the keyboard RAM locations is found to be the same as the corresponding current switch matrix, a valid key push is registered and action is taken. By operating the keyboard scan in this manner an automatic debounce time of 16.67 milliseconds is provided.

Figure 7.2A shows the actual physical layout of the keyboard and Figure 7.2B shows how the individual keys were encoded. On Figure 7.2B the scan lines are the numbers on the bottom of each key position and the return lines are the numbers at the top of each key position. The shift, control, and caps lock key were brought in through separate lines of port C of the 8255. Figure 7.3 shows the basic keyboard matrix.

In order to guarantee that two scan lines could not be shorted together if two or more keys are pushed simultaneously, isolation diodes could be added as shown in Figure 7.4.

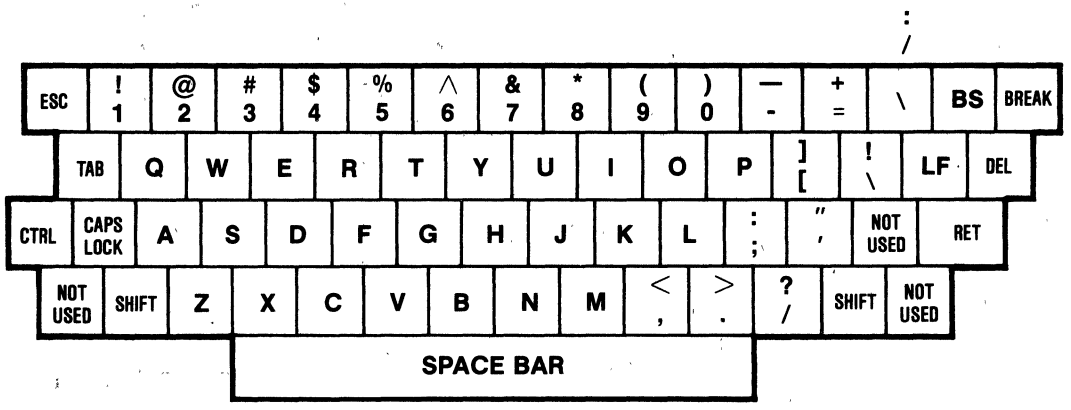
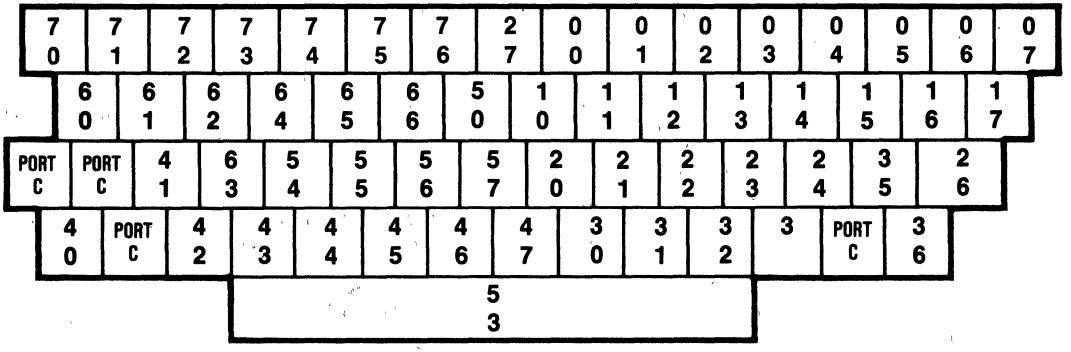


Figure 7-2A. Keyboard Layout



TOP NUMBER = RETURN LINE
BOTTOM NUMBER = SCAN LINE.

Figure 7-2B. Keyboard Encoding

Appendix 7.3
ESCAPE/CONTROL/DISPLAY CHARACTER SUMMARY

	CONTROL CHARACTERS		DISPLAYABLE CHARACTER								ESCAPE SEQUENCE			
BIT	0_{0_0}	0_{0_1}	0_{1_0}	0_{1_1}	1_{0_0}	1_{0_1}	1_{1_0}	1_{1_1}	0_{1_0}	0_{1_1}	1_{0_0}	1_{0_1}	1_{1_0}	1_{1_1}
0000	NUL @	DLE P	SP	ϕ	@	P		P						
0001	SOH A	DC1 Q	!	1	A	Q	A	Q			↑ A			
0010	STX B	DC2 R	"	2	B	R	B	R			↓ B			
0011	ETX C	DC3 S	#	3	C	S	C	S			→ C			
0100	EOT D	DC4 T	$	4	D	T	D	T			← D			
0101	ENQ E	NAK U	%	5	E	U	E	U			CLR E			
0110	ACK F	SYN V	&	6	F	V	F	V						
0111	BEL G	ETB W	'	7	G	W	G	W						
1000	BS H	CAN X	(	8	H	X	H	X			HOME H			
1001	HT I	EM Y	)	9	I	Y	I	Y						
1010	LF J	SUB Z	*	·	J	Z	J	Z			EOS I			
1011	VT K	ESC [	+	;	K	[	K				EL J			
1100	FF L	FS /	,	<	L	\	L							
1101	CR M	GS	-	=	M	]	M							
1110	SO N	RS ^	.	>	N	^	N							
1111	S1 O	US -	/	?	O	-	O							

NOTE Shaded blocks = functions terminal will react to Others can be generated but are ignored up on receipt

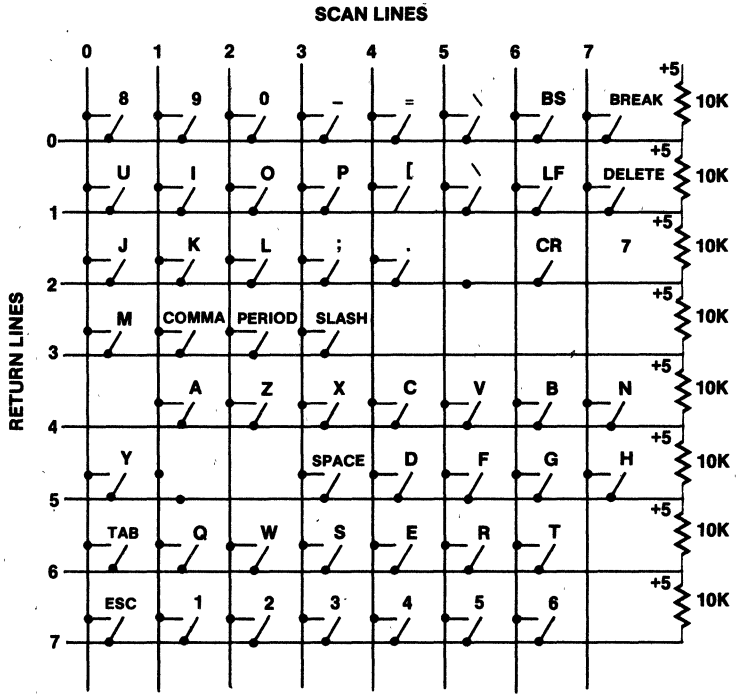


Figure 7-3. Keyboard Matrix

Appendix 7.4
PROM DECODING

As stated earlier, all of the logic necessary to convert the 8275 into a non-DMA type of device was performed by a single small bipolar prom. Besides turning certain processor READS into DACKS and WRITES for the 8275, this 32 by 8 prom decoded addresses for the system ram, rom, as well as for the 8255 parallel I/O port.

Any bipolar prom that has a by eight configuration could function in this application. This particular device was chosen simply because it is the only "by eight" prom available in a 16 pin package. The connection of the prom is shown in detail in Figure 7.5 and its truth table is shown in Figure 7.6. Note that when a fetch cycle (M1) is not being performed, the state of the SOD line is the only thing that determines if memory reads will be written into the 8275's row buffers. This is done by pulling both DACK and WRITE low on the 8275.

Also note that all of the outputs of the bipolar prom MUST BE PULLED HIGH by a resistor. This prevents any unwanted assertions when the prom is disabled.

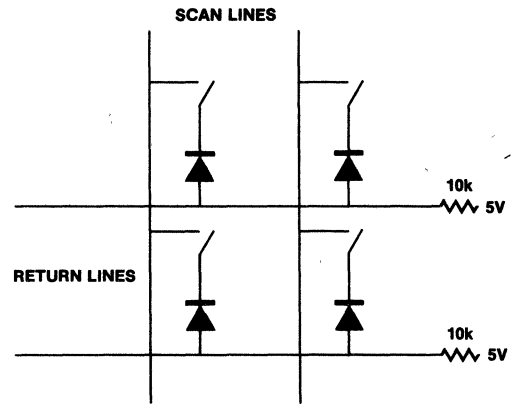


Figure 7-4. Isolating Scan Lines With Diodes

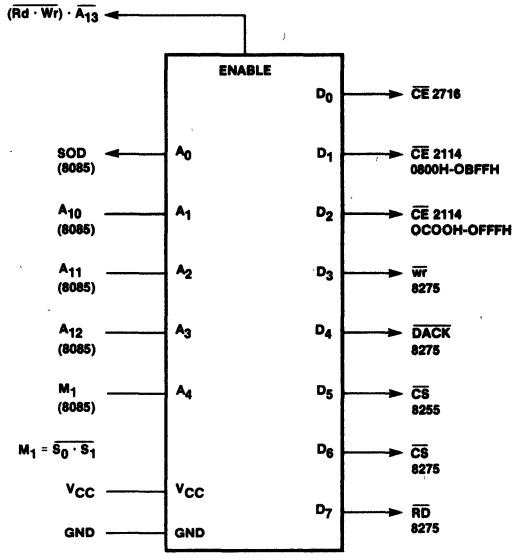


Figure 7-5. Bipolar Prom (825123) Connection

M1	A12	A11	A10	Sod	8275 RD	8275 CS	8255 CS	8275 DACK	8275 WR	2114 H	2114 L	2716
A4	A3	A2	A1	A0	D7	D6	D5	D4	D3	D2	D1	D0
0	0*	0	0	0	1	1	1	1	1	1	1	0
0	0	0	0	1	1	1	1	1	1	1	1	0
0	0	0	1	0	1	1	1	1	1	1	1	0
0	0	0	1	1	1	1	1	1	1	1	1	0
0	0	1	0	0	1	1	1	1	1	1	0	1
0	0	1	0	1	1	1	1	1	1	1	0	1
0	0	1	1	0	1	1	1	1	1	0	1	1
0	0	1	1	1	1	1	1	1	1	0	1	1
0	1	0	0	0	1	0	1	1	1	0	1	1
0	1	0	0	1	1	0	1	1	1	0	1	1
0	1	0	1	0	0	0	1	1	1	1	1	1
0	1	0	1	1	0	0	1	1	1	1	1	1
0	1	1	0	0	1	1	0	1	1	1	1	1
0	1	1	0	1	1	1	0	1	1	1	1	1
0	1	1	1	0	1	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	1	1	1	1	1
1	0	0	0	0	1	1	1	1	1	1	1	0
1	0	0	0	1	1	1	1	0	0	1	1	0
1	0	0	1	0	1	1	1	1	1	1	1	0
1	0	0	1	1	1	1	1	0	0	1	1	0
1	0	1	0	0	1	1	1	1	1	1	0	1
1	0	1	0	1	1	1	1	0	0	1	0	1
1	0	1	1	0	1	1	1	1	1	0	1	1
1	0	1	1	1	1	1	1	0	0	0	1	1
1	1	0	0	0	1	0	1	1	0	1	1	1
1	1	0	0	1	1	0	1	1	0	1	1	1
1	1	0	1	0	0	0	1	1	1	1	1	1
1	1	0	1	1	0	0	1	1	1	1	1	1
1	1	1	0	0	1	1	0	1	1	1	1	1
1	1	1	0	1	1	1	0	1	1	1	1	1
1	1	1	1	0	1	1	0	1	1	1	1	1
1	1	1	1	1	1	1	0	1	1	1	1	1

Figure 7-6. Truth Table Bipolar Prom

Appendix 7.5
CHARACTER GENERATOR

As previously mentioned, the character generator used in this terminal is a 2716 or 2758 EPROM. A 1K by 8 device is sufficient since a 128 character 5 by 7 dot matrix only requires 8K of memory. Any "standard" or custom character generator could have been used.

The three low-order line count outputs (LC0-LC2) from the 8275 are connected to the three low-order address lines of the character generator and the seven character generator outputs (CC0-CC6) are connected to A3-A9 of the character generator. The output from the character generator is loaded into a shift register and the serial output from the shift register is the video output of the terminal.

Now, let's assume that the letter "E" is to be displayed. The ASCII code for "E" is 45H. So, 45H is presented to address lines A2-A9 of the character generator. The scan lines will now count each line from zero to seven to "form" the character as shown in Fig. 7.7. This same procedure is used to form all 128 possible characters.

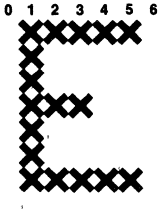
It should be obvious that "custom" character fonts could be made just by changing the bit patterns in the character generator PROM. For reference, Appendix 7.6 contains a HEX dump of the character generator used in this terminal.

45H = 01000101
Address to Prom = 01000101 SL2 SL1 SL0
= 228H - 22FH

Depending on state of Scan lines.

Character generator output

Rom Address	Rom Hex Output	Bit Output*
228H	3E	0 1 2 3 4 5 6 7
229H	02	
22AH	02	
22BH	0E	
22CH	02	
22DH	02	
22EH	3E	
22FH	00	



Bits 0, 6 and 7 are not used.

* note bit output is backward from convention.

Figure 7-7. Character Generation

Appendix 7.6
HEX DUMP OF CHARACTER GENERATOR

```
:100000000100000000000000000000000000000000EF
:10001000000000000000000000000000000000000000E0
:10002000000000000000000000000000000000000000D0
:10003000000000000000000000000000000000000000C0
:10004000000000000000000000000000000000000000B0
:1000500000000000000000000000000000000000000A0
:10006000000000000000000000000000000000000090
:10007000000000000000000000000000000000000080
:10008000000000000000000000000000000000000070
:1000900000000000000082A1C081C2A0830000000BC
:1002A000000000000000000000000000000000000050
:1002B000000000000000000000000000000000000040
:1002C000000000000000000000000000000000000030
:1002D000000000000000000000000000000000000020
:1002E000000000000000000000000000000000000010
:1002F000000000000000000000000000000000000000
:10010000000000000000000000808080808080800BF
:10011000141414000000000143E143E141400C3
:10012000083C0A1C281E08000626100804323000 6D
:10013000040A0A042A122C0708080800000000023
:10014000080402020204080008102020201008000 1
:10015000082A1C081C2A0800080883E0808000 9D
:10016000000000000000808040000003C0000000 3F
:10017000000000000000180000201008040200029
:10018001C22322A26221C00080C0808080 81C0021
:10019001C22201C02023E003E20101820221C00BF
:1001A000101814123E1010003E021E2020221C00C7
:1001B0003804021E22221C003E20100804040400 1
:1001C0001C22221C22221C001C22223C20100E0079
:1001D000000080000080700000000807000080804F3
:1001E0021008040204081000000 03E003E00000059
:1001F00004081020100804001C22201008080000821
:10020001C222A3A1A023C000814222 23E22220012
:10021001E24241C24241E001C22020202221C0074
:10022001E2424242241E003E02020E02023E004C
:10023003E02020E020202003C02023A22223C006E
:10024000222223E222222001C0808080 8081C0044
:10025000702020202020221C0022120A060A122200EE
:10026000202020202023E0022362A2A2222220032
:10027000222262A3222222200 1C22222222221C0092
:10028001E22221E020202001C2222222A122C00FE
:10029001E22221E0A1222003C02021C20201E00E6
:1002A003E0808080808080222222222222 1C00F8
:1002B000222222222214080022222222 2A3522006E
:1002C000222214081422220022222221408080800E4
:1002D0003E2010080402 3E001C0404040 41C0018
:1002E000000204081020000038202020202038 00C0
:1002F000081C2A0808080800000000000000007E12
:10030000890110000000000000 3C203C223C004E
:10031000202 1A2622221E000000380484043800 3B
:10032000202 02C322223C000000 3824BC84B8005B
:10033000382404 0E040404000000BC22223C203CAB
:10034000202 1A2622222200080080808 0890004B
:10035000200020 20A424180202221 20A162200C3
:10036000880808080808900000003 62A2A2222007F
:10037000000 1A2622222200000018242424180 03B
:10038000000 1E22221E020200001C22223C20200D
:10039000000 1A260202020000003804182 01C0087
:1003A00008081C0808089000000022222232 4C0095
:1003B000000022222214080000002222 2A3E1400FB
:1003C000000022140814220000072222 23C20388F
:1003D00000003E1008043E00188889038889190 02F
:1003E000080808080808080C9091219091 0D0051
:1003F00000008C2BB001000000000000000000095
```

Appendix 7.7
COMPOSITE VIDEO

In this design, it was assumed that the monitor required a separate horizontal drive, vertical drive, and video input. However, many monitors require a composite video signal. The schematic shown in Figure 7.8 illustrates how to generate a composite video signal from the output of the 8275.

The dual one-shots are used to provide a small delay and the proper horizontal and vertical pulse to the composite video monitor. The delay introduced in the vertical and horizontal timing is used to "center" the display. VR1 and VR2 control the amount of delay. IC3 is used to mix the vertical and horizontal retrace and Q1 along with the R1, R2, and R3 mix the video and the retrace signal and provide the proper DC levels.

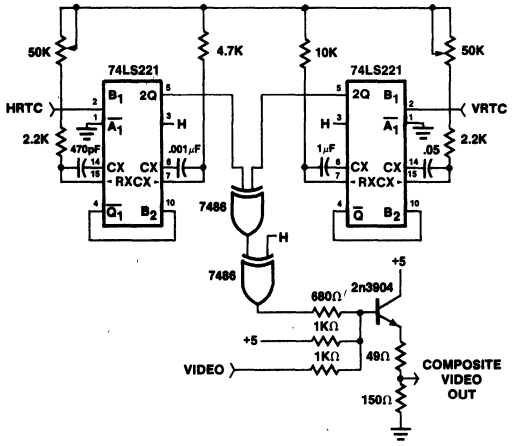


Figure 7-8. Composite Video

Appendix 7.8
SOFTWARE LISTINGS

```
ISIS-II 8080/8085 MACRO ASSEMBLER, X108

LOC  OBJ          SEQ          SOURCE STATEMENT

                    1 $MOD85   MACROFILE
                    2          ;NO DMA 8275 SOFTWARE ALL I/O IS MEMORY MAPPED
                    3          ;SYSTEM ROM 0000H TO 07FFH
                    4          ;SYSTEM RAM 0800H TO 0FFFH
                    5          ;8275 WRITE 1000H TO 13FFH
                    6          ;8275 READ 1400H TO 17FFH
                    7          ;8255 READ/WRITE 1800H TO 1FFF
                    8          ;8253 ENABLED BY A14
                    9          ;8251 ENABLED BY A15
1800                10 PORTA   EQU     1800H          ;8255 PORT A ADDRESS
1801                11 PORTB   EQU     1801H          ;8255 PORT B ADDRESS
1802                12 PORTC   EQU     1802H          ;8255 PORT C ADDRESS
1803                13 CNWD55  EQU     1803H          ;8255 CONTROL PORT ADDRESS
A001                14 USTF    EQU     0A001H         ;8251 FLAGS
A000                15 USTD    EQU     0A000H         ;8251 DATA
6000                16 CNT0    EQU     6000H          ;8253 COUNTER 0
6001                17 CNT1    EQU     6001H          ;8253 COUNTER 1
6002                18 CNT2    EQU     6002H          ;8253 COUNTER 2
6003                19 CNTM    EQU     6003H          ;8253 MODE WORD
1001                20 CRTS    EQU     1001H          ;8275 CONTROL ADDRESS
1000                21 CRTM    EQU     1000H          ;8275 MODE ADDRESS
1401                22 INT75   EQU     1401H          ;8275 INTERRUPT CLEAR
0800                23 TPDIS   EQU     0800H          ;TOP OF DISPLAY RAM
0F80                24 BTDIS   EQU     0F80H          ;BOTTOM OF DISPLAY RAM
0FD0                25 LAST    EQU     0FD0H          ;FIRST BYTE AFTER DISPLAY
0018                26 CURBOT  EQU     18H            ;BOTTOM Y CURSOR
0050                27 LNGTH   EQU     0050H          ;LENGTH OF ONE LINE
0FE0                28 STPTR   EQU     0FE0H          ;LOCATION OF STACK POINTER
                    29         ;
                    30         ;START PROGRAM
                    31         ;ALL VARIABLES ARE INITIALIZED BEFORE ANYTHING ELSE
                    32         ;
0000 F3             33         DI                     ;DISABLE INTERRUPTS
0001 31E00F         34         LXI     SP,STPTR       ;LOAD STACK POINTER
0004 210008         35         LXI     H,TPDIS        ;LOAD H&L WITH TOP OF DISPLAY
0007 22E30F         36         SHLD    TOPAD          ;SET TOP = TOP OF DISPLAY
000A 22E80F         37         SHLD    CURAD          ;STORE THE CURRENT ADDRESS
000D 3E00           38         MVI     A,00H          ;ZERO A
000F 32E10F         39         STA     CURSY          ;ZERO CURSOR Y POINTER
0012 32E20F         40         STA     CURSX          ;ZERO CURSOR X POINTER
0015 32EB0F         41         STA     KBCHR          ;ZERO KBD CHARACTER
0018 32E70F         42         STA     USCHR          ;ZERO USART CHAR BUFFER
001B 32EA0F         43         STA     KEYDWN         ;ZERO KEY DOWN
```

```
001E 32ED0F          44          STA     KEYOK           ;ZERO KEYOK
0021 32EE0F          45          STA     ESCP            ;ZERO ESCAPE
0024 C39800          46          JMP     LPKBD           ;JUMP AND SET EVERYTHING UP
                     47          ;
                     48          ;THIS JUMP VECTOR IS LOCATED AT THE RST 5.5 LOCATION
                     49          ;OF THE 8085. IT IS USED TO READ THE 8275 STATUS AND
                     50          ;READ THE KEYBOARD. THIS ROUTINE IS EXECUTED ONCE EVERY
                     51          ;16.667 MILLISECONDS.
                     52          ;
002C                 53          ORG     002CH
002C C36701          54          JMP     FRAME
                     55          ;
                     56          ;THIS ROUTINE IS LOCATED AT THE RST 6.5 LOCATION OF THE
                     57          ;8085 AND IS USED TO LOAD THE DATA TO BE DISPLAYED INTO
                     58          ;THE 8275. THIS ROUTINE IS EXECUTED ONCE EVERY 617 MICROSECONDS.
                     59          ;
0034                 60          ORG     34H
0034 F5              61 POPDAT:  PUSH    PSW             ;SAVE A AND FLAGS
0035 E5              62          PUSH    H               ;SAVE H AND L
0036 D5              63          PUSH    D               ;SAVE D AND E
0037 210000          64          LXI     H,0000H         ;ZERO H AND L
003A 39              65          DAD     SP              ;PUT STACK POINTER IN H AND L
003B EB              66          XCHG                    ;PUT STACK IN D AND E
003C 2AE80F          67          LHLD    CURAD           ;GET POINTER
003F F9              68          SPHL                    ;PUT CURRENT LINE INTO SP
0040 3EC0            69          MVI     A,0C0H          ;SET MASK FOR SIM
0042 30              70          SIM
                     71          REPT (LNGTH/2)
                     72          POP     H
                     73          ENDM
0043 E1              74+         POP     H
0044 E1              75+         POP     H
0045 E1              76+         POP     H
0046 E1              77+         POP     H
0047 E1              78+         POP     H
0048 E1              79+         POP     H
0049 E1              80+         POP     H
004A E1              81+         POP     H
004B E1              82+         POP     H
004C E1              83+         POP     H
004D E1              84+         POP     H
004E E1              85+         POP     H
004F E1              86+         POP     H
0050 E1              87+         POP     H
0051 E1              88+         POP     H
0052 E1              89+         POP     H
0053 E1              90+         POP     H
0054 E1              91+         POP     H
0055 E1              92+         POP     H
0056 E1              93+         POP     H
0057 E1              94+         POP     H
0058 E1              95+         POP     H
0059 E1              96+         POP     H
005A E1              97+         POP     H
005B E1              98+         POP     H
005C E1              99+         POP     H
005D E1             100+         POP     H
005E E1             101+         POP     H
005F E1             102+         POP     H
0060 E1             103+         POP     H
0061 E1             104+         POP     H
0062 E1             105+         POP     H
0063 E1             106+         POP     H
0064 E1             107+         POP     H
0065 E1             108+         POP     H
0066 E1             109+         POP     H
0067 E1             110+         POP     H
0068 E1             111+         POP     H
0069 E1             112+         POP     H
006A E1             113+         POP     H
006B 0F             114          RRC                     ;SET UP A
006C 30             115          SIM                     ;GO BACK TO NORMAL MODE
006D 210000         116          LXI     H,0000H         ;ZERO HL
0070 39             117          DAD     SP              ;ADD STACK
0071 EB             118          XCHG                    ;PUT STACK IN H AND L
0072 F9             119          SPHL                    ;RESTORE STACK
0073 21D00F         120          LXI     H,LAST          ;PUT BOTTOM DISPLAY IN H AND L
0076 EB             121          XCHG                    ;SWAP REGISTERS
0077 7A             122          MOV     A,D             ;PUT HIGH ORDER IN A
0078 BC             123          CMP     H               ;SEE IF SAME AS H
0079 C28400         124          JNZ     KPTK            ;IF NOT LEAVE
007C 7B             125          MOV     A,E             ;PUT LOW ORDER IN A
007D BD             126          CMP     L               ;SEE IF SAME AS L
007E C28400         127          JNZ     KPTK            ;IF NOT LEAVE
0081 210008         128          LXI     H,TPDIS         ;LOAD H AND L WITH TOP OF SCREEN MEMORY
0084 22E80F         129 KPTK:    SHLD    CURAD           ;PUT BACK CURRENT ADDRESS
0087 3E18           130          MVI     A,18H           ;SET MASK
0089 30             131          SIM                     ;OUTPUT MASK
```

```
008A D1          132          POP     D               ;GET D AND E
008B E1          133          POP     H               ;GET H AND L
008C F1          134          POP     PSW             ;GET A AND FLAGS
008D FB          135          EI                      ;TURN ON INTERRUPTS
008E C9          136          RET                     ;GO BACK
                 137          ;
                 138          ;THIS IS THE EXIT ROUTINE FOR THE FRAME INTERRUPT
                 139          ;
008F 3E18        140 BYPASS:  MVI     A,18H           ;SET MASK
0091 30          141          SIM                     ;OUTPUT THE MASK
0092 C1          142          POP     B               ;GET B AND C
0093 D1          143          POP     D               ;GED D AND E
0094 E1          144          POP     H               ;GET H AND L
0095 F1          145          POP     PSW             ;GET A AND FLAGS
0096 FB          146          EI                      ;ENABLE INTERRUPTS
0097 C9          147          RET                     ;GO BACK
                 148          ;
                 149          ;THIS CLEARS THE AREA OF RAM THAT IS USED
                 150          ;FOR KEYBOARD DEBOUNCE.
                 151          ;
0098 32EF0F      152 LPKBD:   STA     SHCON           ;ZERO SHIFT CONTROL
009B 32F00F      153          STA     RETLIN          ;ZERO RETURN LINE
009E 32F10F      154          STA     SCNLIN          ;ZERO SCAN LINE
                 155          ;
                 156          ;THIS ROUTINE CLEARS THE ENTIRE SCREEN BY PUTTING
                 157          ;SPACE CODES (20H) IN EVERY LOCATION ON THE SCREEN.
                 158          ;
00A1 210008      159          LXI     H,TPDIS         ;PUT TOP OF SCREEN IN HL
00A4 01D00F      160          LXI     B,LAST          ;PUT BOTTOM IN BC
00A7 3620        161 LOOPF:   MVI     M,20H           ;PUT SPACE IN M
00A9 23          162          INX     H               ;INCREMENT POINTER
00AA 7C          163          MOV     A,H             ;GET H
00AB B8          164          CMP     B               ;SEE IF SAME AS B
00AC C2A700      165          JNZ     LOOPF           ;IF NOT LOOP AGAIN
00AF 7D          166          MOV     A,L             ;GET L
00B0 B9          167          CMP     C               ;SEE IF SAME AS C
00B1 C2A700      168          JNZ     LOOPF           ;IF NOT LOOP AGAIN
                 169          ;
                 170          ;8255 INITIALIZATION
                 171          ;
00B4 3E8B        172          MVI     A,8BH           ;MOVE 8255 CONTROL WORD INTO A
00B6 320318      173          STA     CNWD55          ;PUT CONTROL WORD INTO 8255
                 174          ;
                 175          ;8251 INITIALIZATION
                 176          ;
00B9 2101A0      177          LXI     H,USTF          ;GET 8251 FLAG ADDRESS
00BC 3680        178          MVI     M,80H           ;DUMMY STORE TO 8251
00BE 3600        179          MVI     M,00H           ;RESET 8251
00C0 3640        180          MVI     M,40H           ;RESET 8251
00C2 00          181          NOP                     ;WAIT
00C3 36EA        182          MVI     M,0EAH          ;LOAD 8251 MODE WORD
00C5 3605        183          MVI     M,05H           ;LOAD 8251 COMMAND WORD
                 184          ;
                 185          ;8253 INITIALIZATION
                 186          ;
00C7 3E32        187          MVI     A,32H           ;CONTROL WORD FOR 8253
00C9 320360      188          STA     CNTM            ;PUT CONTROL WORD INTO 8253
00CC 3E32        189          MVI     A,32H           ;LSB 8253
00CE 320060      190          STA     CNT0            ;PUT IT IN 8235
00D1 3E00        191          MVI     A,00H           ;MSB 8253
00D3 320060      192          STA     CNT0            ;PUT IT IN 8253
00D6 CDDC00      193          CALL    STBAUD          ;GO DO BAUD RATE
00D9 C3F900      194          JMP     IN75            ;GO DO 8275
                 195          ;
                 196          ;THIS ROUTINE READS THE BAUD RATE SWITCHES FROM PORT C
                 197          ;OF THE 8255 AND LOOKS UP THE NUMBERS NEEDED TO LOAD
                 198          ;THE 8253 TO PROVIDE THE PROPER BAUD RATE.
                 199          ;
00DC 3A0218      200 STBAUD:  LDA     PORTC           ;READ BAUD RATE SWITCHES
00DF E60F        201          ANI     0FH             ;STRIP OFF 4 MSB'S
00E1 32EC0F      202          STA     BAUD            ;SAVE IT
00E4 07          203          RLC                     ;MOVE BITS OVER ONE PLACE
00E5 21C505      204          LXI     H,BDLK          ;GET BAUD RATE LOOK UP TABLE
00E8 1600        205          MVI     D,00H           ;ZERO D
00EA 5F          206          MOV     E,A             ;PUT A IN E
00EB 19          207          DAD     D               ;GET OFFSET
00EC 110360      208          LXI     D,CNTM          ;POINT DE TO 8253
00EF 3EB6        209          MVI     A,0B6H          ;GET CONTROL WORD
00F1 12          210          STAX    D               ;STORE IN 8253
00F2 1B          211          DCX     D               ;POINT AT #2 COUNTER
00F3 7E          212          MOV     A,M             ;GET LSB BAUD RATE
00F4 12          213          STAX    D               ;PUT IT IN 8253
00F5 23          214          INX     H               ;POINT AT MSB BAUD RATE
00F6 7E          215          MOV     A,M             ;GET MSB BAUD RATE
00F7 12          216          STAX    D               ;PUT IT IN 8253
00F8 C9          217          RET                     ;GO BACK
                 218          ;
```

```
                        ;8275 INITIALIZATION
                        ;
00F9 210110     IN75:   LXI     H,CRTS
00FC 3600               MVI     M,00H           ;RESET AND STOP DISPLAY
00FE 2B                 DCX     H               ;HL=1000H
00FF 364F               MVI     M,4FH           ;SCREEN PARAMETER BYTE 1
0101 3658               MVI     M,58H           ;SCREEN PARAMETER BYTE 2
0103 3689               MVI     M,89H           ;SCREEN PARAMETER BYTE 3
0105 36DD               MVI     M,0DDH          ;SCREEN PARAMETER BYTE 4
0107 23                 INX     H               ;HL=1001H
0108 CDB803             CALL    LDCUR           ;LOAD THE CURSOR
010B 36E0               MVI     M,0E0H          ;PRESET COUNTERS
010D 3623               MVI     M,23H           ;START DISPLAY
                        ;
                        ;THIS ROUTINE READS BOTH THE KEYBOARD AND THE USART
                        ;AND TAKES PROPER ACTION DEPENDING ON HOW THE LINE-LOCAL
                        ;SWITCH IS SET
                        ;
010F 3E18       SETUP:  MVI     A,18H           ;SET MASK
0111 30                 SIM                     ;LOAD MASK
0112 FB                 EI                      ;ENABLE INTERRUPTS
                        ;
                        ;READ THE USART
                        ;
0113 20         RXRDY:  RIM                     ;GET LINE LOCAL
0114 E680               ANI     80H             ;IS IT ON OR OFF?
0116 C22101             JNZ     KEYINP          ;LEAVE IF IT IS ON
0119 3A01A0             LDA     USTF            ;READ 8251 FLAGS
011C E602               ANI     02H             ;LOOK AT RXRDY
011E C25C01             JNZ     OK7             ;IF HAVE CHARACTER GO TO WORK
0121 3AEA0F     KEYINP: LDA     KEYDWN          ;GET KEYBOARD CHARACTER
0124 E680               ANI     80H             ;IS IT THERE
0126 C23101             JNZ     KEYS            ;IF KEY IS PUSHED LEAVE
0129 3E00               MVI     A,00H           ;ZERO A
012B 32ED0F             STA     KEYOK           ;CLEAR KEYOK
012E C31301             JMP     RXRDY           ;LOOP AGAIN
0131 3AED0F     KEYS:   LDA     KEYOK           ;WAS KEY DOWN
0134 4F                 MOV     C,A             ;SAVE A IN C
0135 3AEB0F             LDA     KBCHR           ;GET KEYBOARD CHARACTER
0138 B9                 CMP     C               ;IS IT THE SAME AS KEYOK
0139 CA1301             JZ      RXRDY           ;IF SAME LOOP AGAIN
013C 32ED0F             STA     KEYOK           ;IF NOT SAVE IT
013F 32E70F             STA     USCHR           ;SAVE IT
0142 20                 RIM                     ;GET LINE LOCAL
0143 E680               ANI     80H             ;WHICH WAY
0145 CA4B01             JZ      TRANS           ;LEAVE IF LINE
0148 C34E02             JMP     CHREC           ;TIME TO DO SOME WORK
014B 3A01A0     TRANS:  LDA     USTF            ;GET USART FLAGS
014E E601               ANI     01H             ;READY TO TRANSMIT?
0150 CA4B01             JZ      TRANS           ;LOOP IF NOT READY
0153 3AE70F             LDA     USCHR           ;GET CHARACTER
0156 3200A0             STA     USTD            ;PUT IN USART
0159 C30F01             JMP     SETUP           ;LEAVE
015C 3A00A0     OK7:    LDA     USTD            ;READ USART
015F E67F               ANI     07FH            ;STRIP MSB
0161 32E70F             STA     USCHR           ;PUT IT IN MEMORY
0164 C34E02             JMP     CHREC           ;LEAVE
                        ;
                        ;THIS ROUTINE CHECKS THE BAUD RATE SWITCHES, RESETS THE
                        ;SCREEN POINTERS AND READS AND LOOKS UP THE KEYBOARD.
                        ;
0167 F5         FRAME:  PUSH    PSW             ;SAVE A AND FLAGS
0168 E5                 PUSH    H               ;SAVE H AND L
0169 D5                 PUSH    D               ;SAVE D AND E
016A C5                 PUSH    B               ;SAVE B AND C
016B 3A0114             LDA     INT75           ;READ 8275 TO CLEAR INTERRUPT
                        ;
                        ;SET UP THE POINTERS
                        ;
016E 2AE30F             LHLD    TOPAD           ;LOAD TOP IN H AND L
0171 22E80F             SHLD    CURAD           ;STORE TOP IN CURRENT ADDRESS
                        ;
                        ;SET UP BAUD RATE
                        ;
0174 3A0218             LDA     PORTC           ;READ BAUD RATE SWITCHES
0177 E60F               ANI     0FH             ;STRIP OFF 4 MSB'S
0179 47                 MOV     B,A             ;SAVE IN B
017A 3ABC0F             LDA     BAUD            ;GET BAUD RATE
017D B8                 CMP     B               ;SEE IF SAME AS B
017E C4DC00             CNZ     STBAUD          ;IF NOT SAME DO SOMETHING
                        ;
                        ;READ KEYBOARD
                        ;
0181 3AEA0F             LDA     KEYDWN          ;SEE IF A KEY IS DOWN
0184 E640               ANI     40H             ;SET THE FLAGS
0186 C2C201             JNZ     KYDOWN          ;IF KEY IS DOWN JUMP AROUND
0189 CD8F01             CALL    RDKB            ;GO READ THE KEYBOARD
018C C38F00             JMP     BYPASS          ;LEAVE
```

```
018F 21EF0F      307 RDKB:   LXI    H,SHCON      ;POINT HL AT KEYBOARD RAM
0192 3A0218      308         LDA    PORTC        ;GET CONTROL AND SHIFT
0195 77          309         MOV    M,A          ;SAVE IN MEMORY
0196 3EFE        310         MVI    A,0FEH       ;SET UP A
0198 320018      311 LOOPK:  STA    PORTA        ;OUTPUT A
019B 47          312         MOV    B,A          ;SAVE A IN B
019C 3A0118      313         LDA    PORTB        ;READ KEYBOARD
019F 2F          314         CMA                 ;INVERT A
01A0 B7          315         ORA    A            ;SET THE FLAGS
01A1 C2AF01      316         JNZ    SAVKEY       ;LEAVE IF KEY IS DOWN
01A4 78          317         MOV    A,B          ;GET SCAN LINE BACK
01A5 07          318         RLC                 ;ROTATE IT OVER ONE
01A6 DA9801      319         JC     LOOPK        ;DO IT AGAIN
01A9 3E00        320         MVI    A,00H        ;ZERO A
01AB 32EA0F      321         STA    KEYDWN       ;SAVE KEY DOWN
01AE C9          322         RET                 ;LEAVE
01AF 23          323 SAVKEY: INX    H            ;POINT AT RETURN LINE
01B0 2F          324         CMA                 ;PUT A BACK
01B1 77          325         MOV    M,A          ;SAVE RETURN LINE IN MEMORY
01B2 23          326         INX    H            ;POINT H AT SCAN LINE
01B3 70          327         MOV    M,B          ;SAVE SCAN LINE IN MEMORY
01B4 3E40        328         MVI    A,40H        ;SET A
01B6 32EA0F      329         STA    KEYDWN       ;SAVE KEY DOWN
01B9 C9          330         RET                 ;LEAVE
01BA 3E00        331 KYCHNG: MVI    A,00H        ;ZERO 0
01BC 32EA0F      332         STA    KEYDWN       ;RESET KEY DOWN
01BF C38F00      333         JMP    BYPASS       ;LEAVE
01C2 21F10F      334 KYDOWN: LXI    H,SCNLIN     ;GET SCAN LINE
01C5 7E          335         MOV    A,M          ;PUT SCAN LINE IN A
01C6 320018      336         STA    PORTA        ;OUTPUT SCAN LINE TO PORT A
01C9 2B          337         DCX    H            ;POINT AT RETURN LINE
01CA 3A0118      338         LDA    PORTB        ;GET RETURN LINES
01CD B6          339         ORA    M            ;ARE THEY THE SAME?
01CE 2F          340         CMA                 ;INVERT A
01CF B7          341         ORA    A            ;SET FLAGS
01D0 CABA01      342         JZ     KYCHNG       ;IF DIFFERENT KEY HAS CHANGED
01D3 3AEA0F      343         LDA    KEYDWN       ;GET KEY DOWN
01D6 E601        344         ANI    01H          ;HAS THIS BEEN DONE BEFORE?
01D8 C28F00      345         JNZ    BYPASS       ;LEAVE IF IT HAS
01DB 3A0118      346         LDA    PORTB        ;GET RETURN LINE
01DE 06FF        347         MVI    B,0FFH       ;GET READY TO ZERO B
01E0 04          348 UP:     INR    B            ;ZERO B
01E1 0F          349         RRC                 ;ROTATE A
01E2 DAE001      350         JC     UP           ;DO IT AGAIN
01E5 23          351         INX    H            ;POINT H AT SCAN LINES
01E6 7E          352         MOV    A,M          ;GET SCAN LINES
01E7 0EFF        353         MVI    C,0FFH       ;GET READY TO LOOP
01E9 0C          354 UP1:    INR    C            ;START C COUNTING
01EA 0F          355         RRC                 ;ROTATE A
01EB DAE901      356         JC     UP1          ;JUMP TO LOOP
01EE 78          357         MOV    A,B          ;GET RETURN LINES
01EF 07          358         RLC                 ;MOVE OVER ONCE
01F0 07          359         RLC                 ;MOVE OVER TWICE
01F1 07          360         RLC                 ;MOVE OVER THREE TIMES
01F2 B1          361         ORA    C            ;OR SCAN AND RETURN LINES
01F3 47          362         MOV    B,A          ;SAVE A IN B
01F4 3A0218      363         LDA    PORTC        ;GET SHIFT CONTROL
01F7 E640        364         ANI    40H          ;IS CONTROL SET
01F9 4F          365         MOV    C,A          ;SAVE A IN C
01FA 3AEF0F      366         LDA    SHCON        ;GET SHIFT CONTROL
01FD 57          367         MOV    D,A          ;SAVE A IN D
01FE E640        368         ANI    40H          ;STRIP CONTROL
0200 B1          369         ORA    C            ;SET BIT
0201 CA3E02      370         JZ     CNTDWN       ;IF SET LEAVE
0204 3A0218      371         LDA    PORTC        ;READ IT AGAIN,
0207 E620        372         ANI    20H          ;STRIP SHIFT
0209 4F          373         MOV    C,A          ;SAVE A
020A 7A          374         MOV    A,D          ;GET SHIFT CONTROL
020B E620        375         ANI    20H          ;STRIP CONTROL
020D B1          376         ORA    C            ;ARE THEY THE SAME?
020E CA4702      377         JZ     SHDWN        ;IF SET LEAVE
0211 58          378 SCR:    MOV    E,B          ;PUT  TARGET IN E
0212 1600        379         MVI    D,00H        ;ZERO D
0214 210705      380         LXI    H,KYLKUP     ;GET LOOKUP TABLE
0217 19          381         DAD    D            ;GET OFFSET
0218 7E          382         MOV    A,M          ;GET CHARACTER
0219 47          383         MOV    B,A          ;PUT CHARACTER IN B
021A 3A0218      384         LDA    PORTC        ;GET PORTC
021D E610        385         ANI    10H          ;STRIP BIT
021F CA2E02      386         JZ     CAPLOC       ;CAPS LOCK
0222 78          387         MOV    A,B          ;GET A BACK
0223 32EB0F      388 STKEY:  STA    KBCHR        ;SAVE CHARACTER
0226 3EC1        389         MVI    A,0C1H       ;SET A
0228 32EA0F      390         STA    KEYDWN       ;SAVE KEY DOWN
022B C38F00      391         JMP    BYPASS       ;LEAVE
                 392         ;
                 393         ;IF THE CAP LOCK BUTTON IS PUSHED THIS ROUTINE SEES IF
                 394         ;THE CHARACTER IS BETWEEN 61H AND 7AH AND IF IT IS THIS
```

```
                        395          ;ROUTINE ASSUMES THAT THE CHARACTER IS LOWER CASE ASCII
                        396          ;AND SUBTRACTS 20H, WHICH CONVERTS THE CHARACTER TO
                        397          ;UPPER CASE ASCII
                        398          ;
022E 78                 399 CAPLOC:  MOV     A,B             ;GET A BACK
022F FE60               400          CPI     60H             ;HOW BIG IS IT?
0231 DA2302             401          JC      STKEY           ;LEAVE IF IT'S TOO SMALL
0234 FE7B               402          CPI     7BH             ;IS IT TOO BIG
0236 D22302             403          JNC     STKEY           ;LEAVE IF TOO BIG
0239 D620               404          SUI     20H             ;ADJUST A
023B C32302             405          JMP     STKEY           ;STORE THE KEY
                        406          ;
                        407          ;THE ROUTINES SHDWN AND CNTDWN SET BIT 6 AND 7 RESPECTIVLY
                        408          ;IN THE ACC.
                        409          ;
023E 3E80               410 CNTDWN:  MVI     A,80H           ;SET BIT 7 IN A
0240 B0                 411          ORA     B               ;OR WITH CHARACTER
0241 E6BF               412          ANI     0BFH            ;MAKE SURE SHIFT IS NOT SET
0243 47                 413          MOV     B,A             ;PUT IT BACK IN B
0244 C31102             414          JMP     SCR             ;GO BACK
0247 3E40               415 SHDWN:   MVI     A,40H           ;SET BIT 6 IN A
0249 B0                 416          ORA     B               ;OR WITH CHARACTER
024A 47                 417          MOV     B,A             ;PUT IT BACK IN B
024B C31102             418          JMP     SCR             ;GO BACK
                        419          ;
                        420          ;THIS ROUTINE CHECKS FOR ESCAPE CHARACTERS, LF, CR,
                        421          ;FF, AND BACK SPACE
                        422          ;
024E 3AEE0F             423 CHREC:   LDA     ESCP            ;ESCAPE SET?
0251 FE80               424          CPI     80H             ;SEE IF IT IS
0253 CA7B02             425          JZ      ESSQ            ;LEAVE IF IT IS
0256 3AE70F             426          LDA     USCHR           ;GET CHARACTER
0259 FE0A               427          CPI     0AH             ;LINE FEED
025B CAF603             428          JZ      LNFD            ;GO TO LINE FEED
025E FE0C               429          CPI     0CH             ;FORM FEED
0260 CACA03             430          JZ      FMFD            ;GO TO FORM FEED
0263 FE0D               431          CPI     0DH             ;CR
0265 CAAD03             432          JZ      CGRT            ;DO A CR
0268 FE08               433          CPI     08H             ;BACK SPACE
026A CA6E03             434          JZ      LEFT            ;DO A BACK SPACE
026D FE1B               435          CPI     1BH             ;ESCAPE
026F CAA503             436          JZ      ESKAP           ;DO AN ESCAPE
0272 B7                 437          ORA     A               ;CLEAR CARRY
0273 C6E0               438          ADI     0E0H            ;SEE IF CHARACTER IS PRINTABLE
0275 DA7704             439          JC      CHRPUT          ;IF PRINTABLE DO IT
0278 C30F01             440          JMP     SETUP           ;GO BACK AND READ USART AGAIN
                        441          ;
                        442          ;THIS ROUTINE RESETS THE ESCAPE LOCATION AND DECODES
                        443          ;THE CHARACTERS FOLLOWING AN ESCAPE. THE COMMANDS ARE
                        444          ;COMPATABLE WITH INTELS CREDIT TEXT EDITOR
                        445          ;
0278 3E00               446 ESSQ:    MVI     A,00H           ;ZERO A
027D 32EE0F             447          STA     ESCP            ;RESET  ESCP
0280 3AE70F             448          LDA     USCHR           ;GET CHARACTER
0283 FE42               449          CPI     42H             ;DOWN
0285 CAAE02             450          JZ      DOWN            ;MOVE CURSOR DOWN
0288 FE45               451          CPI     45H             ;CLEAR SCREEN CHARACTER
028A CACF02             452          JZ      CLEAR           ;CLEAR THE SCREEN
028D FE4A               453          CPI     4AH             ;CLEAR REST OF SCREEN
028F CAD502             454          JZ      CLRST           ;GO CLEAR THE REST OF THE SCREEN
0292 FE4B               455          CPI     4BH             ;CLEAR LINE CHARACTER
0294 CA2703             456          JZ      CLRLIN          ;GO CLEAR A LINE
0297 FE41               457          CPI     41H             ;CURSOR UP CHARACTER
0299 CA3303             458          JZ      UPCUR           ;MOVE CURSOR UP
029C FE43               459          CPI     43H             ;CURSOR RIGHT CHARACTER
029E CA4503             460          JZ      RIGHT           ;MOVE CURSOR TO THE RIGHT
02A1 FE44               461          CPI     44H             ;CURSOR LEFT CHARACTER
02A3 CA6E03             462          JZ      LEFT            ;MOVE CURSOR TO THE LEFT
02A6 FE48               463          CPI     48H             ;HOME CURSOR CHARACTER
02A8 CA9703             464          JZ      HOME            ;HOME THE CURSOR
02AB C30F01             465          JMP     SETUP           ;LEAVE
                        466          ;
                        467          ;THIS ROUTINE MOVES THE CURSOR DOWN ONE CHARACTER LINE
                        468          ;
02AE 3AE10F             469 DOWN:    LDA     CURSY           ;PUT CURSOR Y IN A
02B1 FE18               470          CPI     CURBOT          ;SEE IF ON BOTTOM OF SCREEN
02B3 CA0F01             471          JZ      SETUP           ;LEAVE IF ON BOTTOM
02B6 3C                 472          INR     A               ;INCREMENT Y CURSOR
02B7 32E10F             473          STA     CURSY           ;SAVE NEW CURSOR
02BA CDB803             474          CALL    LDCUR           ;LOAD THE CURSOR
02BD CDA504             475          CALL    CALCU           ;CALCULATE ADDRESS
02C0 7E                 476          MOV     A,M             ;GET FIRST LOCATION OF THE LINE
02C1 FEF0               477          CPI     0F0H            ;SEE IF CLEAR SCREEN CHARACTER
02C3 C20F01             478          JNZ     SETUP           ;LEAVE IF IT IS NOT
02C6 22E50F             479          SHLD    LOC80           ;SAVE BEGINNING OF THE LINE
02C9 CD1504             480          CALL    CLLINE          ;CLEAR THE LINE
02CC C30F01             481          JMP     SETUP           ;LEAVE
                        482          ;
```

```
                        483          ;THIS ROUTINE CLEARS THE SCREEN.
                        484          ;
02CF CDE403             485 CLEAR:   CALL    CLSCR          ;GO CLEAR THE SCREEN
02D2 C30F01             486          JMP     SETUP          ;GO BACK
                        487          ;
                        488          ;THIS ROUTINE CLEARS ALL LINES BENEATH THE LOCATION
                        489          ;OF THE CURSOR.
                        490          ;
02D5 CDA504             491 CLRST:   CALL    CALCU          ;CALCULATE ADDRESS
02D8 CDCD04             492          CALL    ADX            ;ADD X POSITION
02DB 01204F             493          LXI     B,4F20H        ;PUT SPACE AND LAST X IN B AND C
02DE 3AE20F             494          LDA     CURSX          ;GET X CURSOR
02E1 B8                 495          CMP     B              ;SEE IF AT END OF LINE
02E2 CAEC02             496          JZ      OVR1           ;LEAVE IF X IS AT END OF LINE
02E5 3C                 497 LLP:     INR     A              ;MOVE A OVER ONE X POSITION
02E6 23                 498          INX     H              ;INCREMENT MEMORY POINTER
02E7 71                 499          MOV     M,C            ;PUT A SPACE IN MEMORY
02E8 B8                 500          CMP     B              ;SEE IF A = 4FH
02E9 C2E502             501          JNZ     LLP            ;IF NOT LOOP AGAIN
02EC 01D00F             502 OVR1:    LXI     B,LAST         ;PUT LAST LINE IN BC
02EF 23                 503          INX     H              ;POINT HL TO LAST LINE
02F0 78                 504          MOV     A,B            ;GET B
02F1 BC                 505          CMP     H              ;SAME AS H?
02F2 C2FD02             506          JNZ     CONCL          ;LEAVE IF NOT
02F5 79                 507          MOV     A,C            ;GET C
02F6 BD                 508          CMP     L              ;SAME AS L?
02F7 C2FD02             509          JNZ     CONCL          ;LEAVE IF NOT
02FA 210008             510          LXI     H,TPDIS        ;GET TOP OF DISPLAY
02FD 3AE10F             511 CONCL:   LDA     CURSY          ;GET Y CURSOR
0300 FE18               512          CPI     CURBOT         ;IS IT ON THE BOTTOM
0302 CA0F01             513          JZ      SETUP          ;LEAVE IF IT IS
0305 3C                 514          INR     A              ;MOVE IT DOWN ONE LINE
0306 47                 515          MOV     B,A            ;SAVE CURSOR IN B FOR LATER
0307 115000             516          LXI     D,LNGTH        ;PUT LENGTH OF ONE LINE IN D
030A 36F0               517 CLOOP:   MVI     M,0F0H         ;PUT EOR IN MEMORY
030C 78                 518          MOV     A,B            ;GET CURSOR Y
030D FE18               519          CPI     CURBOT         ;ARE WE ON THE BOTTOM
030F CA0F01             520          JZ      SETUP          ;LEAVE IF WE ARE
0312 3C                 521          INR     A              ;MOVE CURSOR DOWN ONE
0313 19                 522          DAD     D              ;GET NEXT LINE
0314 47                 523          MOV     B,A            ;SAVE A
0315 7C                 524          MOV     A,H            ;PUT H IN A
0316 FE0F               525          CPI     0FH            ;COMPARE TO HIGH LAST
0318 C20A03             526          JNZ     CLOOP          ;LEAVE IF IT IS NOT
031B 7D                 527          MOV     A,L            ;PUT L IN A
031C FED0               528          CPI     0D0H           ;COMPARE TO LOW LAST
031E C20A03             529          JNZ     CLOOP          ;LEAVE IF IT IS NOT
0321 210008             530          LXI     H,TPDIS        ;PUT TOP DISPLAY IN H AND L
0324 C30A03             531          JMP     CLOOP          ;LOOP AGAIN
                        532          ;
                        533          ;THIS ROUTINE CLEARS THE LINE THE CURSOR IS ON.
                        534          ;
0327 CDA504             535 CLRLIN:  CALL    CALCU          ;CALCULATE ADDRESS
032A 22E50F             536          SHLD    LOC80          ;STORE H AND L TO CLEAR LINE
032D CD1504             537          CALL    CLLINE         ;CLEAR THE LINE
0330 C30F01             538          JMP     SETUP          ;GO BACK
                        539          ;
                        540          ;THIS ROUTINE MOVES THE CURSOR UP ONE LINE.
                        541          ;
0333 3AE10F             542 UPCUR:   LDA     CURSY          ;GET Y CURSOR
0336 FE00               543          CPI     00H            ;IS IT ZERO
0338 CA0F01             544          JZ      SETUP          ;IF IT IS LEAVE
033B 3D                 545          DCR     A              ;MOVE CURSOR UP
033C 32E10F             546          STA     CURSY          ;SAVE NEW CURSOR
033F CD8803             547          CALL    LDCUR          ;LOAD THE CURSOR
0342 C30F01             548          JMP     SETUP          ;LEAVE
                        549          ;
                        550          ;THIS ROUTINE MOVES THE CURSOR ONE LOCATION TO THE RIGHT
                        551          ;
0345 3AE20F             552 RIGHT:   LDA     CURSX          ;GET X CURSOR
0348 FE4F               553          CPI     4FH            ;IS IT ALL THE WAY OVER?
034A C25403             554          JNZ     NTOVER         ;IF NOT JUMP AROUND
034D 3AE10F             555          LDA     CURSY          ;GET Y CURSOR
0350 FE18               556          CPI     CURBOT         ;SEE IF ON BOTTOM
0352 CA5903             557          JZ      GD18           ;IF WE ARE JUMP
0355 3C                 558          INR     A              ;INCREMENT Y CURSOR
0356 32E10F             559          STA     CURSY          ;SAVE IT
0359 3E00               560 GD18:    MVI     A,00H          ;ZERO A
035B 32E20F             561          STA     CURSX          ;ZERO X CURSOR
035E CD8803             562          CALL    LDCUR          ;LOAD THE CURSOR
0361 C30F01             563          JMP     SETUP          ;LEAVE
0364 3C                 564 NTOVER:  INR     A              ;INCREMENT X CURSOR
0365 32E20F             565          STA     CURSX          ;SAVE IT
0368 CD8803             566          CALL    LDCUR          ;LOAD THE CURSOR
036B C30F01             567          JMP     SETUP          ;LEAVE
                        568          ;
                        569          ;THIS ROUTINE MOVES THE CURSOR LEFT ONE CHARACTER POSITION
```

```
                        570
                        571 ;
036E 3AE20F             572 LEFT:   LDA     CURSX           ;GET X CURSOR
0371 FE00               573         CPI     00H             ;IS IT ALL THE WAY OVER
0373 C28D03             574         JNZ     NOVER           ;IF NOT JUMP AROUND
0376 3AE10F             575         LDA     CURSY           ;GET CURSOR Y
0379 FE00               576         CPI     00H             ;IS IT ZERO?
037B CA0F01             577         JZ      SETUP           ;IF IT IS JUMP
037E 3D                 578         DCR     A               ;MOVE CURSOR Y UP
037F 32E10F             579         STA     CURSY           ;SAVE IT
0382 3E4F               580         MVI     A,4FH           ;GET LAST X LOCATION
0384 32E20F             581         STA     CURSX           ;SAVE IT
0387 CDB803             582         CALL    LDCUR           ;LOAD THE CURSOR
038A C30F01             583         JMP     SETUP
038D 3D                 584 NOVER:  DCR     A               ;ADJUST X CURSOR
038E 32E20F             585         STA     CURSX           ;SAVE CURSOR X
0391 CDB803             586         CALL    LDCUR           ;LOAD THE CURSOR
0394 C30F01             587         JMP     SETUP           ;LEAVE
                        588         ;
                        589         ;THIS ROUTINE HOMES THE CURSOR.
                        590         ;
0397 3E00               591 HOME:   MVI     A,00H           ;ZERO A
0399 32E20F             592         STA     CURSX           ;ZERO X CURSOR
039C 32E10F             593         STA     CURSY           ;ZERO Y CURSOR
039F CDB803             594         CALL    LDCUR           ;LOAD THE CURSOR
03A2 C30F01             595         JMP     SETUP           ;LEAVE
                        596         ;
                        597         ;THIS ROUTINE SETS THE ESCAPE BIT
                        598         ;
03A5 3E80               599 ESKAP:  MVI     A,80H           ;LOAD A WITH ESCAPE BIT
03A7 32EE0F             600         STA     ESCP            ;SET ESCAPE LOCATION
03AA C30F01             601         JMP     SETUP           ;GO BACK AND READ USART
                        602         ;
                        603         ;THIS ROUTINE DOES A CR
                        604         ;
03AD 3E00               605 CGRT:   MVI     A,00H           ;ZERO A
03AF 32E20F             606         STA     CURSX           ;ZERO  CURSOR X
03B2 CDB803             607         CALL    LDCUR           ;LOAD CURSOR INTO 8275
03B5 C30F01             608         JMP     SETUP           ;POLL USART AGAIN
                        609         ;
                        610         ;THIS ROUTINE LOADS THE CURSOR
                        611         ;
03B8 3E80               612 LDCUR:  MVI     A,80H           ;PUT 80H INTO A
03BA 320110             613         STA     CRTS            ;LOAD CURSOR INTO 8275
03BD 3AE20F             614         LDA     CURSX           ;GET CURSOR X
03C0 320010             615         STA     CRTM            ;PUT IT IN 8275
03C3 3AE10F             616         LDA     CURSY           ;GET CURSOR Y
03C6 320010             617         STA     CRTM            ;PUT IT IN 8275
03C9 C9                 618         RET
                        619         ;
                        620         ;THIS ROUTINE DOES A FORM FEED
                        621         ;
03CA CDE403             622 FMFD:   CALL    CLSCR           ;CALL CLEAR SCREEN
03CD 210008             623         LXI     H,TPDIS         ;PUT TOP DISPLAY IN HL
03D0 22E50F             624         SHLD    LOC80           ;PUT IT IN LOC80
03D3 CD1504             625         CALL    CLLINE          ;CLEAR TOP LINE
03D6 3E00               626         MVI     A,00H           ;ZERO A
03D8 32E20F             627         STA     CURSX           ;ZERO CURSOR X
03DB 32E10F             628         STA     CURSY           ;ZERO CURSOR Y
03DE CDB803             629         CALL    LDCUR           ;LOAD THE CURSOR
03E1 C30F01             630         JMP     SETUP           ;BACK TO USART
                        631         ;
                        632         ;THIS ROUTINE CLEARS THE SCREEN BY WRITING END OF ROW
                        633         ;CHARACTERS INTO THE FIRST LOCATION OF ALL LINES ON
                        634         ;THE SCREEN.
                        635         ;
03E4 3EF0               636 CLSCR:  MVI     A,0F0H          ;PUT EOR CHARACTER IN A
03E6 0618               637         MVI     B,CURBOT        ;LOAD B WITH MAX Y
03E8 04                 638         INR     B               ;GO TO MAX PLUS ONE
03E9 210008             639         LXI     H,TPDIS         ;LOAD H AND L WITH TOP OF RAM
03EC 115000             640         LXI     D,LNGTH         ;MOVE 50H = 80D INTO D AND E
03EF 77                 641 LOADX:  MOV     M,A             ;MOVE EOR INTO MEMORY
03F0 19                 642         DAD     D               ;CHANGE POINTER BY 80D
03F1 05                 643         DCR     B               ;COUNT THE LOOPS
03F2 C2EF03             644         JNZ     LOADX           ;CONTINUE IF NOT ZERO
03F5 C9                 645         RET                     ;GO BACK
                        646         ;
                        647         ;THIS ROUTINE DOES A LINE FEED
                        648         ;
03F6 CDFC03             649 LNFD:   CALL    LNFD1           ;CALL ROUTINE
03F9 C30F01             650         JMP     SETUP           ;POLL FLAGS
                        651         ;
                        652         ;LINE FEED
                        653         ;
03FC 3AE10F             654 LNFD1:  LDA     CURSY           ;GET Y LOCATION OF CURSOR
03FF FE18               655         CPI     CURBOT          ;SEE IF AT BOTTOM OF SCREEN
0401 CA5304             656         JZ      ONBOT           ;IF WE ARE, LEAVE
0404 3C                 657         INR     A               ;INCREMENT A
0405 32E10F             658         STA     CURSY           ;SAVE NEW CURSOR
```

```
0408 CDA504          658          CALL    CALCU           ;CALCULATE ADDRESS
040B 22E50F          659          SHLD    LOC80           ;SAVE TO CLEAR LINE
040E CD1504          660          CALL    CLLINE          ;CLEAR THE LINE
0411 CDB803          661          CALL    LDCUR           ;LOAD THE CURSOR
0414 C9              662          RET                     ;LEAVE
                     663          ;
                     664          ;THIS ROUTINE CLEARS THE LINE WHOSE FIRST ADDRESS
                     665          ;IS IN LOC80. PUSH INSTRUCTIONS ARE USED TO RAPIDLY
                     666          ;CLEAR THE LINE
                     667          ;
0415 F3              668 CLLINE:  DI                      ;NO INTERRUPTS HERE
0416 2AE50F          669          LHLD    LOC80           ;GET LOC80
0419 115000          670          LXI     D,LNGTH         ;GET OFFSET
041C 19              671          DAD     D               ;ADD OFFSET
041D EB              672          XCHG                    ;PUT START IN DE
041E 210000          673          LXI     H,0000H         ;ZERO HL
0421 39              674          DAD     SP              ;GET STACK
0422 EB              675          XCHG                    ;PUT STACK IN DE
0423 F9              676          SPHL                    ;PUT START IN SP
0424 212020          677          LXI     H,2020H         ;PUT SPACES IN HL
                     678          ;
                     679          ;NOW DO 40 PUSH INSTRUCTIONS TO CLEAR THE LINE
                     680          ;
                     681          REPT    (LNGTH/2)
                     682          PUSH    H
                     683          ENDM
0427 E5              684+         PUSH    H
0428 E5              685+         PUSH    H
0429 E5              686+         PUSH    H
042A E5              687+         PUSH    H
042B E5              688+         PUSH    H
042C E5              689+         PUSH    H
042D E5              690+         PUSH    H
042E E5              691+         PUSH    H
042F E5              692+         PUSH    H
0430 E5              693+         PUSH    H
0431 E5              694+         PUSH    H
0432 E5              695+         PUSH    H
0433 E5              696+         PUSH    H
0434 E5              697+         PUSH    H
0435 E5              698+         PUSH    H
0436 E5              699+         PUSH    H
0437 E5              700+         PUSH    H
0438 E5              701+         PUSH    H
0439 E5              702+         PUSH    H
043A E5              703+         PUSH    H
043B E5              704+         PUSH    H
043C E5              705+         PUSH    H
043D E5              706+         PUSH    H
043E E5              707+         PUSH    H
043F E5              708+         PUSH    H
0440 E5              709+         PUSH    H
0441 E5              710+         PUSH    H
0442 E5              711+         PUSH    H
0443 E5              712+         PUSH    H
0444 E5              713+         PUSH    H
0445 E5              714+         PUSH    H
0446 E5              715+         PUSH    H
0447 E5              716+         PUSH    H
0448 E5              717+         PUSH    H
0449 E5              718+         PUSH    H
044A E5              719+         PUSH    H
044B E5              720+         PUSH    H
044C E5              721+         PUSH    H
044D E5              722+         PUSH    H
044E E5              723+         PUSH    H
044F EB              724          XCHG                    ;PUT STACK IN HL
0450 F9              725          SPHL                    ;PUT IT BACK IN SP
0451 FB              726          EI                      ;ENABLE INTERRUPTS
0452 C9              727          RET                     ;GO BACK
                     728          ;
                     729          ;IF CURSOR IS ON THE BOTTOM OF THE SCREEN THIS ROUTINE
                     730          ;IS USED TO IMPLEMENT THE LINE FEED
                     731          ;
0453 2AE30F          732 ONBOT:   LHLD    TOPAD           ;GET TOP ADDRESS
0456 22E50F          733          SHLD    LOC80           ;SAVE IT IN LOC80
0459 115000          734          LXI     D,LNGTH         ;LINE LENGTH
045C 19              735          DAD     D               ;ADD HL + DE
045D 01D00F          736          LXI     B,LAST          ;GET BOTTOM LINE
0460 7C              737          MOV     A,H             ;GET H
0461 B8              738          CMP     B               ;SAME AS B
0462 C26D04          739          JNZ     ARND            ;LEAVE IF NOT SAME
0465 7D              740          MOV     A,L             ;GET L
0466 B9              741          CMP     C               ;SAME AS C
0467 C26D04          742          JNZ     ARND            ;LEAVE IF NOT SAME
046A 210008          743          LXI     H,TPDIS         ;LOAD HL WITH TOP OF DISPLAY
046D 22E30F          744 ARND:    SHLD    TOPAD           ;SAVE NEW TOP ADDRESS
```

```
0470 CD1504          745           CALL    CLLINE          ;CLEAR LINE
0473 CDB803          746           CALL    LDCUR           ;LOAD THE CURSOR
0476 C9              747           RET
                     748           ;
                     749           ;THIS ROUTINE PUTS A CHARACTER ON THE SCREEN AND
                     750           ;INCREMENTS THE X CURSOR POSITION. A LINE FEED IS
                     751           ;INSERTED IF THE INCREMENTED CURSOR EQUALS 81D
                     752           ;
0477 CDA504          753 CHRPUT:   CALL    CALCU           ;CALCULATE SCREEN POSITION
047A 7E              754           MOV     A,M             ;GET FIRST CHARACTER
047B FEF0            755           CPI     0F0H            ;IS IT A CLEAR LINE
047D 22E50F          756           SHLD    LOC80           ;SAVE LINE TO CLEAR
0480 CC1504          757           CZ      CLLINE          ;CLEAR LINE
0483 2AE50F          758           LHLD    LOC80           ;GET LINE
0486 CDCD04          759           CALL    ADX             ;ADD CURSOR X
0489 3AE70F          760           LDA     USC'R           ;GET CHARACTER
048C 77              761           MOV     M,A             ;PUT IT ON SCREEN
048D 3AE20F          762           LDA     CURSX           ;GET CURSOR X
0490 3C              763           INR     A               ;INCREMENT CURSOR X
0491 FE50            764           CPI     LNGTH           ;HAS IT GONE TOO FAR?
0493 C29C04          765           JNZ     OK1             ;IF NOT GOOD
0496 CDFC03          766           CALL    LNFD1           ;DO A LINE FEED
0499 C3AD03          767           JMP     CGRT            ;DO A CR
049C 32E20F          768 OK1:      STA     CURSX           ;SAVE CURSOR
049F CDB803          769           CALL    LDCUR           ;LOAD THE CURSOR
04A2 C30F01          770           JMP     SETUP           ;LEAVE
                     771           ;
                     772           ;THIS ROUTINE TAKES THE TOP ADDRESS AND THE Y CURSOR
                     773           ;LOCATION AND CALCULATES THE ADDRESS OF THE LINE
                     774           ;THAT THE CURSOR IS ON. THE RESULT IS RETURNED IN H
                     775           ;AND L AND ALL REGISTERS ARE USED.
                     776           ;
04A5 21D504          777 CALCU:    LXI     H,LINTAB        ;GET LINE TABLE INTO H AND L
04A8 3AE10F          778           LDA     CURSY           ;GET CURSOR INTO A
04AB 07              779           RLC                     ;SET UP A FOR LOOKUP TABLE
04AC 0600            780           MVI     B,00H           ;ZERO B
04AE 4F              781           MOV     C,A             ;PUT CURSOR INTO A
04AF 09              782           DAD     B               ;ADD LINE TABLE TO Y CURSOR
04B0 7E              783           MOV     A,M             ;PUT LOW LINE TABLE INTO A
04B1 4F              784           MOV     C,A             ;PUT LOW LINE TABLE INTO C
04B2 23              785           INX     H               ;CHANGE MEMORY POINTER
04B3 7E              786           MOV     A,M             ;PUT HIGH LINE TABLE INTO A
04B4 47              787           MOV     B,A             ;PUT HIGH LINE TABLE INTO B
04B5 2100F8          788           LXI     H,0F800H        ;TWOS COMPLEMENT SCREEN LOCATION
04B8 09              789           DAD     B               ;SUBTRACT OFFSET
04B9 EB              790           XCHG                    ;SAVE HL IN DE
04BA 2AE30F          791           LHLD    TOPAD           ;GET TOP ADDRESS IN H AND L
04BD 19              792           DAD     D               ;GET DISPLACED ADDRESS
04BE EB              793           XCHG                    ;SAVE IT IN D
04BF 2130F0          794           LXI     H,0F030H        ;TWOS COMPLEMENT SCREEN LOCATION
04C2 19              795           DAD     D               ;SEE IF WE ARE OFF THE SCREEN
04C3 DAC804          796           JC      FIX             ;IF WE ARE FIX IT
04C6 EB              797           XCHG                    ;GET DISPLACED ADDRESS BACK
04C7 C9              798           RET                     ;GO BACK
04C8 2130F8          799 FIX:      LXI     H,0F830H        ;SCREEN BOUNDRY
04CB 19              800           DAD     D               ;ADJUST SCREEN
04CC C9              801           RET                     ;GO BACK
                     802           ;
                     803           ;THIS ROUTINE ADDS THE X CURSOR LOCATION TO THE ADDRESS
                     804           ;THAT IS IN THE H AND L REGISTERS AND RETURNS THE RESULT
                     805           ;IN H AND L
                     806           ;
04CD 3AE20F          807 ADX:      LDA     CURSX           ;GET CURSOR
04D0 0600            808           MVI     B,00H           ;ZERO B
04D2 4F              809           MOV     C,A             ;PUT CURSOR X IN C
04D3 09              810           DAD     B               ;ADD CURSOR X TO H AND L
04D4 C9              811           RET                     ;LEAVE
                     812           ;
                     813           ;THIS TABLE CONTAINS THE OFFSET ADDRESSES FOR EACH
                     814           ;OF THE 25 DISPLAYED LINES.
                     815           ;
0000                 816 LINTAB:   LINNUM SET 0
                     817           REPT (CURBOT+1)
                     818           DW      TPDIS+(LNGTH*LINNUM)
                     819           LINNUM SET (LINNUM+1)
                     820           ENDM
04D5 0008            821+          DW      TPDIS+(LNGTH*LINNUM)
0001                 822+          LINNUM SET (LINNUM+1)
04D7 5008            823+          DW      TPDIS+(LNGTH*LINNUM)
0002                 824+          LINNUM SET (LINNUM+1)
04D9 A008            825+          DW      TPDIS+(LNGTH*LINNUM)
0003                 826+          LINNUM SET (LINNUM+1)
04DB F008            827+          DW      TPDIS+(LNGTH*LINNUM)
0004                 828+          LINNUM SET (LINNUM+1)
04DD 4009            829+          DW      TPDIS+(LNGTH*LINNUM)
0005                 830+          LINNUM SET (LINNUM+1)
04DF 9009            831+          DW      TPDIS+(LNGTH*LINNUM)
```

```
0006                    832+        LINNUM SET (LINNUM+1)
04E1 E009               833+        DW      TPDIS+(LNGTH*LINNUM)
0007                    834+        LINNUM SET (LINNUM+1)
04E3 300A               835+        DW      TPDIS+(LNGTH*LINNUM)
0008                    836+        LINNUM SET (LINNUM+1)
04E5 800A               837+        DW      TPDIS+(LNGTH*LINNUM)
0009                    838+        LINNUM SET (LINNUM+1)
04E7 D00A               839+        DW      TPDIS+(LNGTH*LINNUM)
000A                    840+        LINNUM SET (LINNUM+1)
04E9 200B               841+        DW      TPDIS+(LNGTH*LINNUM)
000B                    842+        LINNUM SET (LINNUM+1)
04EB 700B               843+        DW      TPDIS+(LNGTH*LINNUM)
000C                    844+        LINNUM SET (LINNUM+1)
04ED C00B               845+        DW      TPDIS+(LNGTH*LINNUM)
000D                    846+        LINNUM SET (LINNUM+1)
04EF 100C               847+        DW      TPDIS+(LNGTH*LINNUM)
000E                    848+        LINNUM SET (LINNUM+1)
04F1 600C               849+        DW      TPDIS+(LNGTH*LINNUM)
000F                    850+        LINNUM SET (LINNUM+1)
04F3 B00C               851+        DW      TPDIS+(LNGTH*LINNUM)
0010                    852+        LINNUM SET (LINNUM+1)
04F5 000D               853+        DW      TPDIS+(LNGTH*LINNUM)
0011                    854+        LINNUM SET (LINNUM+1)
04F7 500D               855+        DW      TPDIS+(LNGTH*LINNUM)
0012                    856+        LINNUM SET (LINNUM+1)
04F9 A00D               857+        DW      TPDIS+(LNGTH*LINNUM)
0013                    858+        LINNUM SET (LINNUM+1)
04FB F00D               859+        DW      TPDIS+(LNGTH*LINNUM)
0014                    860+        LINNUM SET (LINNUM+1)
04FD 400E               861+        DW      TPDIS+(LNGTH*LINNUM)
0015                    862+        LINNUM SET (LINNUM+1)
04FF 900E               863+        DW      TPDIS+(LNGTH*LINNUM)
0016                    864+        LINNUM SET (LINNUM+1)
0501 E00E               865+        DW      TPDIS+(LNGTH*LINNUM)
0017                    866+        LINNUM SET (LINNUM+1)
0503 300F               867+        DW      TPDIS+(LNGTH*LINNUM)
0018                    868+        LINNUM SET (LINNUM+1)
0505 800F               869+        DW      TPDIS+(LNGTH*LINNUM)
0019                    870+        LINNUM SET (LINNUM+1)
                        871         ;
                        872         ;KEYBOARD LOOKUP TABLE
                        873         ;THIS TABLE CONTAINS ALL THE ASCII CHARACTERS
                        874         ;THAT ARE TRANSMITTED BY THE TERMINAL
                        875         ;THE CHARACTERS ARE ORGANIZED SO THAT BITS 0,1 AND 2
                        876         ;ARE THE SCAN LINES, BITS 3,4 AND 5 ARE THE RETURN LINES
                        877         ;BIT 6 IS SHIFT AND BIT 7 IS CONTROL
                        878         ;
0507 38                 879 KYLKUP: DB      38H,39H         ;8 AND 9
0508 39
0509 30                 880         DB      30H,2DH         ;0 AND -
050A 2D
050B 3D                 881         DB      3DH,5CH         ;= AND \
050C 5C
050D 08                 882         DB      08H,00H         ;BS AND BREAK
050E 00
050F 75                 883         DB      75H,69H         ;LOWER CASE U AND I
0510 69
0511 6F                 884         DB      6FH,70H         ;LOWER CASE O AND P
0512 70
0513 5B                 885         DB      5BH,5CH         ;[ AND \
0514 5C
0515 0A                 886         DB      0AH,7FH         ;LF AND DELETE
0516 7F
0517 6A                 887         DB      6AH,6BH         ;LOWER CASE J AND K
0518 6B
0519 6C                 888         DB      6CH,3BH         ;LOWER CASE L AND ;
051A 3B
051B 27                 889         DB      27H,00H         ;' AND NOTHING
051C 00
051D 0D                 890         DB      0DH,37H         ;CR AND 7
051E 37
051F 6D                 891         DB      6DH,2CH         ;LOWER CASE M AND COMMA
0520 2C
0521 2E                 892         DB      2EH,2FH         ;PERIOD AND SLASH
0522 2F
0523 00                 893         DB      00H,00H         ;BLANK AND NOTHING
0524 00
0525 00                 894         DB      00H,00H         ;NOTHING AND NOTHING
0526 00
0527 00                 895         DB      00H,61H         ;NOTHING AND LOWER CASE A
0528 61
0529 7A                 896         DB      7AH,78H         ;LOWER CASE Z AND X
052A 78
052B 63                 897         DB      63H,76H         ;LOWER CASE C AND V
052C 76
052D 62                 898         DB      62H,6EH         ;LOWER CASE B AND N
052E 6E
```

```
052F 79          899          DB          79H,00H          ;LOWER CASE Y AND NOTHING
0530 00
0531 00          900          DB          00H,20H          ;NOTHING AND SPACE
0532 20
0533 64          901          DB          64H,66H          ;LOWER CASE D AND F
0534 66
0535 67          902          DB          67H,68H          ;LOWER CASE G AND H
0536 68
0537 00          903          DB          00H,71H          ;TAB AND LOWER CASE Q
0538 71
0539 77          904          DB          77H,73H          ;LOWER CASE W AND S
053A 73
053B 65          905          DB          65H,72H          ;LOWER CASE E AND R
053C 72
053D 74          906          DB          74H,00H          ;LOWER CASE T AND NOTHING
053E 00
053F 1B          907          DB          1BH,31H          ;ESCAPE AND I
0540 31
0541 32          908          DB          32H,33H          ; 2 AND 3
0542 33
0543 34          909          DB          34H,35H          ; 4 AND 5
0544 35
0545 36          910          DB          36H,00H          ; 6 AND NOTHING
0546 00
0547 2A          911          DB          2AH,28H          ;* AND )
0548 28
0549 29          912          DB          29H,5FH          ;( AND -
054A 5F
054B 2B          913          DB          2BH,00H          ;+ AND NOTHING
054C 00
054D 08          914          DB          08H,00H          ;BS AND BREAK
054E 00
054F 55          915          DB          55H,49H          ;U AND I
0550 49
0551 4F          916          DB          4FH,50H          ;O AND P
0552 50
0553 5D          917          DB          5DH,00H          ;] AND NO CHARACTER
0554 00
0555 0A          918          DB          0AH,7FH          ;LF AND DELETE
0556 7F
0557 4A          919          DB          4AH,4BH          ;J AND K
0558 4B
0559 4C          920          DB          4CH,3AH          ;L AND :
055A 3A
055B 22          921          DB          22H,00H          ;" AND NO CHARACTER
055C 00
055D 0D          922          DB          0DH,26H          ;CR AND &
055E 26
055F 4D          923          DB          4DH,3CH          ;M AND <
0560 3C
0561 3E          924          DB          3EH,3FH          ;> AND ?
0562 3F
0563 00          925          DB          00H,00H          ;BLANK AND NOTHING
0564 00
0565 00          926          DB          00H,00H          ;NOTHING AND NOTHING
0566 00
0567 00          927          DB          00H,41H          ;NOTHING AND A
0568 41
0569 5A          928          DB          5AH,58H          ;Z AND X
056A 58
056B 43          929          DB          43H,56H          ;C AND V
056C 56
056D 42          930          DB          42H,4EH          ;B AND N
056E 4E
056F 59          931          DB          59H,00H          ;Y AND NOTHING
0570 00
0571 00          932          DB          00H,20H          ;NO CHARACTER AND SPACE
0572 20
0573 44          933          DB          44H,46H          ;D AND F
0574 46
0575 47          934          DB          47H,48H          ;G AND H
0576 48
0577 00          935          DB          00H,51H          ;TAB AND Q
0578 51
0579 57          936          DB          57H,53H          ;W AND S
057A 53
057B 45          937          DB          45H,52H          ;E AND R
057C 52
057D 54          938          DB          54H,00H          ;T AND NO CONNECTION
057E 00
057F 1B          939          DB          1BH,21H          ;ESCAPE AND !
0580 21
0581 40          940          DB          40H,23H          ;@ AND #
0582 23
0583 24          941          DB          24H,25H          ;$ AND %
0584 25
0585 5E          942          DB          5EH,00H          ;^ AND NO CONNECTION
```

```
0586 00
                        943     ;
                        944     ;THIS IS WHERE THE CONTROL CHARACTERS ARE LOOKED UP
                        945     ;
0587 00                 946     DB      00H,00H         ;NOTHING
0588 00
0589 00                 947     DB      00H,00H         ;NOTHING
058A 00
058B 00                 948     DB      00H,00H         ;NOTHING
058C 00
058D 00                 949     DB      00H,00H         ;NOTHING
058E 00
058F 15                 950     DB      15H,09H         ;CONTROL U AND I
0590 09
0591 0F                 951     DB      0FH,10H         ;CONTROL O AND P
0592 10
0593 0B                 952     DB      0BH,0CH         ;CONTROL [ AND \
0594 0C
0595 0A                 953     DB      0AH,7FH         ;LF AND DELETE
0596 7F
0597 0A                 954     DB      0AH,0BH         ;CONTROL J AND K
0598 0B
0599 0C                 955     DB      0CH,00H         ;CONTROL L AND NOTHING
059A 00
059B 00                 956     DB      00H,00H         ;NOTHING
059C 00
059D 0D                 957     DB      0DH,00H         ;CR AND NOTHING
059E 00
059F 0D                 958     DB      0DH,00H         ;CONTROL M AND COMMA
05A0 00
05A1 00                 959     DB      00H,00H         ;NOTHING
05A2 00
05A3 00                 960     DB      00H,00H         ;NOTHING
05A4 00
05A5 00                 961     DB      00H,00H         ;NOTHING AND NOTHING
05A6 00
05A7 1A                 962     DB      1AH,18H         ;CONTROL Z AND X
05A8 18
05A9 03                 963     DB      03H,16H         ;CONTROL C AND V
05AA 16
05AB 02                 964     DB      02H,0EH         ;CONTROL B AND N
05AC 0E
05AD 19                 965     DB      19H,00H         ;CONTROL Y AND NOTHING
05AE 00
05AF 00                 966     DB      00H,20H         ;NOTHING AND SPACE
05B0 20
05B1 04                 967     DB      04H,06H         ;CONTROL D AND F
05B2 06
05B3 07                 968     DB      07H,08H         ;CONTROL G AND H
05B4 08
05B5 00                 969     DB      00H,11H         ;NOTHING AND CONTROL Q
05B6 11
05B7 17                 970     DB      17H,13H         ;CONTROL W AND S
05B8 13
05B9 06                 971     DB      06H,12H         ;CONTROL E AND R
05BA 12
05BB 14                 972     DB      14H,00H         ;CONTROL W AND NOTHING
05BC 00
05BD 1B                 973     DB      1BH,1DH         ;ESCAPE AND HOME(CREDIT)
05BE 1D
05BF 1E                 974     DB      1EH,1CH         ;CURSOR UP AND DOWN(CREDIT)
05C0 1C
05C1 14                 975     DB      14H,1FH         ;CURSOR RIGHT AND LEFT(CREDIT)
05C2 1F
05C3 00                 976     DB      00H,00H         ;NOTHING
05C4 00
                        977     ;
                        978     ;LOOK UP TABLE FOR 8253 BAUD RATE GENERATOR
                        979     ;
05C5 00                 980 BDLK:   DB      00H,05H,69H,03H ;75 AND 110 BAUD
05C6 05
05C7 69
05C8 03
05C9 80                 981     DB      80H,02H,40H,01H ;150 AND 300 BAUD
05CA 02
05CB 40
05CC 01
05CD A0                 982     DB      0A0H,00H        ;600 BAUD
05CE 00
05CF 50                 983     DB      50H,00H         ;1200 BAUD
05D0 00
05D1 28                 984     DB      28H,00H         ;2400 BAUD
05D2 00
05D3 14                 985     DB      14H,00H         ;4800 BAUD
05D4 00
05D5 0A                 986     DB      0AH,00H         ;9600 BAUD
05D6 00
```

```
                        987             ;
                        988             ;DATA AREA
                        989             ;
0FE1                    990             ORG     0FE1H
0001                    991 CURSY:      DS      1
0001                    992 CURSX:      DS      1
0002                    993 TOPAD:      DS      2
0002                    994 LOC80:      DS      2
0001                    995 USCHR:      DS      1
0002                    996 CURAD:      DS      2
0001                    997 KEYDWN:     DS      1
0001                    998 KBCHR:      DS      1
0001                    999 BAUD:       DS      1
0001                   1000 KEYOK:      DS      1
0001                   1001 ESCP:       DS      1
0001                   1002 SHCON:      DS      1
0001                   1003 RETLIN:     DS      1
0001                   1004 SCNLIN:     DS      1
                       1005             END

PUBLIC SYMBOLS

EXTERNAL SYMBOLS

USER SYMBOLS
ADX     A 04CD  ARND    A 046D  BAUD    A 0FEC  BDLK    A 05C5  BTDIS   A 0F80  BYPASS  A 008F
CAPLOC  A 022E  CGRT    A 03AD  CHREC   A 024E  CHRPUT  A 0477  CLEAR   A 02CF  CLLINE  A 0415
CLRLIN  A 0327  CLRST   A 02D5  CLSCR   A 03E4  CNT0    A 6000  CNT1    A 6001  CNT2    A 6002
CNTM    A 6003  CNWD55  A 1803  CONCL   A 02FD  CRTM    A 1000  CRTS    A 1001  CURAD   A 0FE8
CURSX   A 0FE2  CURSY   A 0FE1  DOWN    A 02AE  ESCP    A 0FEE  ESKAP   A 03A5  ESSQ    A 027B
FMFD    A 03CA  FRAME   A 0167  GD18    A 0359  HOME    A 0397  IN75    A 00F9  INT75   A 1401
KEYDWN  A 0FEA  KEYINP  A 0121  KEYOK   A 0FED  KEYS    A 0131  KPTK    A 0084  KYCHNG  A 01BA
KYLKUP  A 0507  LAST    A 0FD0  LDCUR   A 03B8  LEFT    A 036E  LINNUM  A 0019  LINTAB  A 04D5
LNFD    A 03F6  LNFD1   A 03FC  LNGTH   A 0050  LOADX   A 03EF  LOC80   A 0FE5  LOOPF   A 00A7
LPKBD   A 0098  NOVER   A 038D  NTOVER  A 0364  OK1     A 049C  OK7     A 015C  ONBOT   A 0453
POPDAT  A 0034  PORTA   A 1800  PORTB   A 1801  PORTC   A 1802  RDKB    A 018F  RETLIN  A 0FF0
RXRDY   A 0113  SAVKEY  A 01AF  SCNLIN  A 0FF1  SCR     A 0211  SETUP   A 010F  SHCON   A 0FEF
STBAUD  A 00DC  STKEY   A 0223  STPTR   A 0FE0  TOPAD   A 0FE3  TPDIS   A 0800  TRANS   A 014B
UP1     A 01E9  UPCUR   A 0333  USCHR   A 0FE7  USTD    A A000  USTF    A A001

ASSEMBLY COMPLETE,  NO ERRORS
```