intel®

APPLICATION NOTE

AP-32

November 1977

CRT Terminal Design Using The Intel® 8275 and 8279

John Murray and George Alexy
Microcomputer Applications

PRICE $1.00
9800576A

RELATED INTEL PUBLICATIONS

"MCS-80™ USER's MANUAL"

"MEMORY DESIGN HANDBOOK"

" PERIPHERALS DATA NOTEBOOK "

CRT Terminal Design Using The Intel® 8275 and 8279

Contents

1. INTRODUCTION

The purpose of this application note is to provide the reader with the conceptual and factual tools needed to apply the 8275 Programmable CRT Controller and 8279 Programmable Keyboard/Display Interface in CRT system design. The 8275 Controller is designed to interface CRT raster scan displays with Intel® Microcomputer Products. Its primary functions include refreshing the CRT display by buffering information from display memory and generating horizontal and vertical timing signals used for CRT synchronization. The programmable features of the 8275 allow it to be interfaced to almost any raster scan display with a minimum of external hardware. In addition, visual attribute features allow the implementation of specialized graphic display functions and display enhancement operations. The 8279 Keyboard Interface provides key scanning, debounce, and buffering features required for interfacing CRT terminal keyboards to the system processor. Two key or N-key rollover is provided. The use of these devices in a microcomputer based CRT terminal yields substantial savings in component count, printed circuit board area, and power consumption.

The application note is divided into five sections:

1. Introduction
2. CRT System Design Concepts
3. Component Description
4. CRT System Design Example
5. Appendix

Readers desiring an overview of CRT system design should consider reading the first three sections of the application note. Individuals requiring an in-depth knowledge of CRT system design should read the first three sections, then proceed to the design example. The design example consists of a description of the design of a complete CRT terminal. Both hardware and software aspects of the design are included. It will be assumed in Section 4 that the reader is familiar with the 8275, 8279, and 8257 data sheets, and the operation of the 8080A microprocessor.

2. CRT SYSTEM DESIGN CONCEPTS

2.1 CRT OPERATION

In order to fully understand the CRT terminal design process, it is necessary to consider the fundamentals of CRT operation. A typical CRT Monitor is shown in Figure 2-1. The CRT consists of an evacuated glass structure having a phosphorescent coating on the inner surface of the rectangular frontal region (screen). A filament contained in the narrow cylindrical region (neck) of the CRT heats the cathode, causing the cathode to give off electrons by thermionic emission. Heating is accomplished by applying a low voltage source across the filament leads. A high voltage source applied between the cathode and the screen electrode (anode) accelerates the electrons toward the screen. The electron beam, upon striking the phosphorescent inner surface of the screen, produces light. To control the point at which the beam strikes the screen, two primary deflection techniques are utilized. The first technique, electromagnetic deflection, involves applying a current through a deflection coil placed around the neck of the CRT. The resulting magnetic field forces the electron beam to be deflected in proportion to the magnitude of the applied current. Electrostatic deflection involves placing deflection electrodes in the neck of the CRT perpendicular to the electron beam. An applied voltage changes the position of the beam accordingly.

2.2 MONITOR OPERATION

A CRT monitor consists of a CRT and the electronics required for positioning the beam in the desired manner. A block diagram of the control electronics contained within a typical CRT monitor is provided in Figure 2-2.

The horizontal oscillator is designed to move the electron beam horizontally across the CRT screen and then return the beam rapidly to its original position. As the beam is moved horizontally, the vertical oscillator causes the beam to be deflected vertically. The net result of these operations is to move the beam in a manner shown in Figure 2-3. If the intensity of the electron beam is modulated in a controlled manner as the beam sweeps across the screen, it is possible to display pictorial information on the CRT screen surface. It will be assumed that the monitor in question will be used for displaying alphanumeric characters or graphic symbols. In this case, the electron beam will be turned on to display a light region on the screen and turned off to display a dark region. Display information appearing at the video input to the CRT is applied through the video amplifier to a control grid located in the neck of the CRT. The magnitude of the video signal determines whether the electron beam will be on or off.

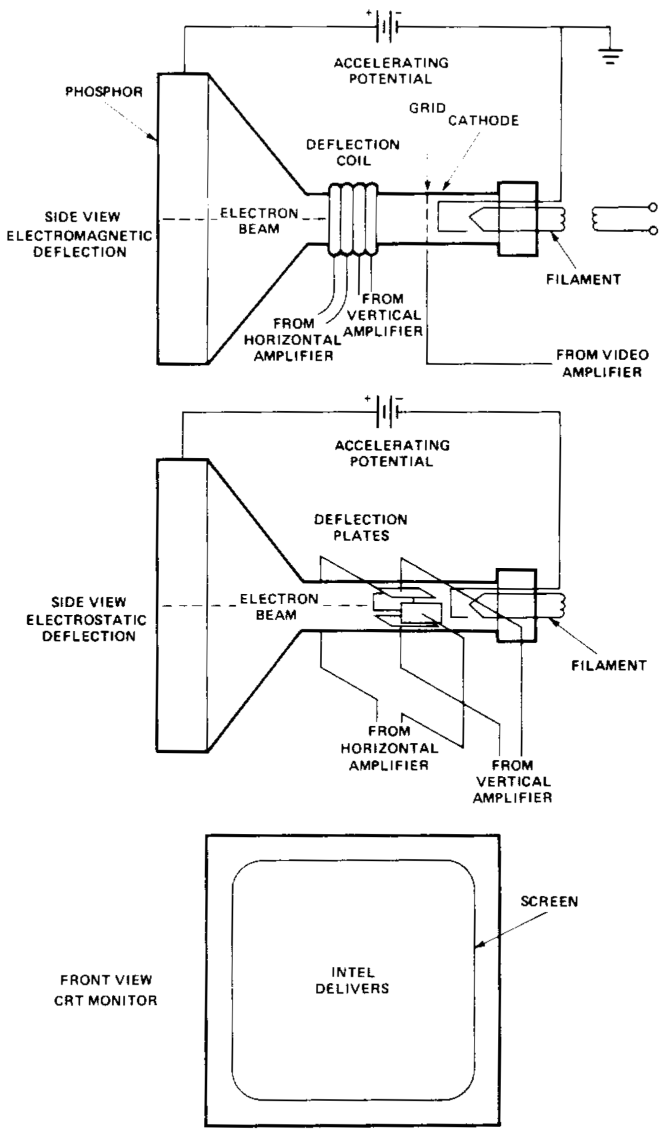


Figure 2-1. CRT Monitor

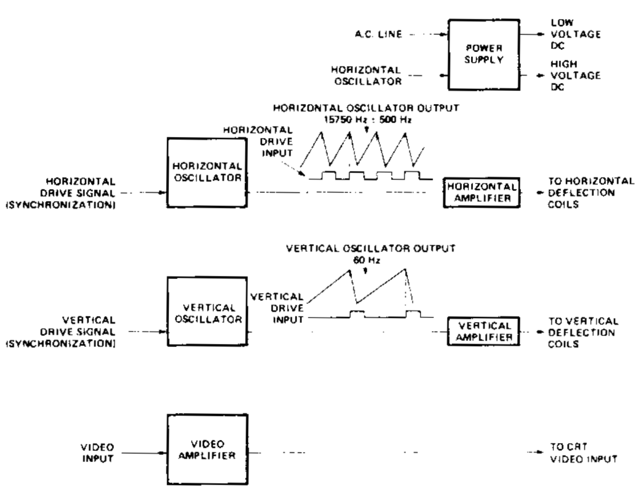


Figure 2-2. CRT Monitor Electronics

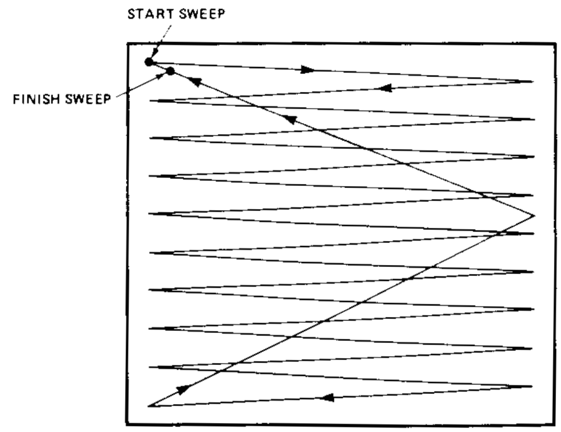


Figure 2-3. CRT Monitor Raster

2.3 CRT TERMINAL DESCRIPTION

A CRT terminal consists basically of a CRT monitor, monitor control electronics, memory for storing display information, logic to control information transfer to and from external devices and between internal devices, and a keyboard. The fundamental operations performed by a CRT terminal consist of the display of information contained in internal memory on the CRT screen, communication with manual data entry devices such as keyboards or light pens, and communication with external intelligent devices such as computers or data communication terminals. Typical CRT terminal communication functions are illustrated in Figure 2-4.

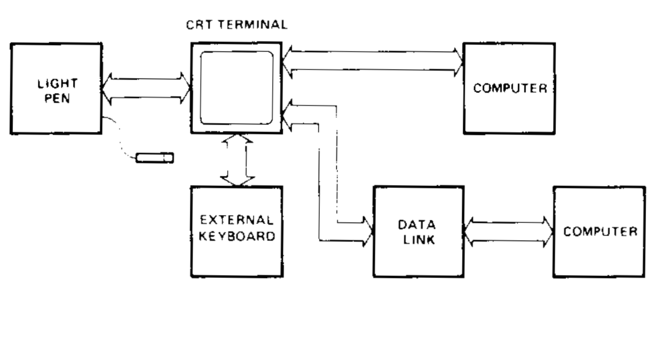


Figure 2-4. CRT Terminal Communications

2.4 CRT TERMINAL IMPLEMENTATION

A typical microprocessor-based CRT terminal is presented in block diagram form in Figure 2-5. The terminal consists of the CRT monitor, monitor electronics, memory for storing the information to be displayed, a serial communication device, keyboard, keyboard interface device, CRT controller, central processor and associated program memory, and a DMA device. The primary function of the CRT controller is to refresh the display. It does this by controlling the periodic transfer of information from display memory to the CRT screen. The central processor unit (CPU) coordinates the transfer of information to and from the terminal peripheral devices and external devices. When information from an external device is received by the terminal, the central processor performs character recognition and handling functions, display memory management functions, and cursor control functions. The CPU also interrogates the keyboard interface device. If a key depression is detected by the keyboard interface device, the CPU responds by transmitting the ASCII character representing the key to the terminal serial output line via the serial communication device. A direct memory access (DMA) device is required in the system to effect the necessary memory to screen data transfer rate.

The CRT terminal control functions under consideration may be implemented with LSI devices at a considerable cost savings over earlier terminal designs using MSI and SSI components. This cost savings is complemented by an increase in the number of features which can be incorporated in terminal designs. The additional features stem from the programmable nature of the devices. In addition, utilizing a microprocessor as the terminal controller allows considerable intelligence to be built into the terminal for decision making, computational, and control functions. The design example presented in Section 4 of the application note illustrates the use of the 8275 Programmable CRT Controller and 8279 Keyboard Controller in a typical terminal design. In the following section, the 8275 and 8279 are considered in depth.

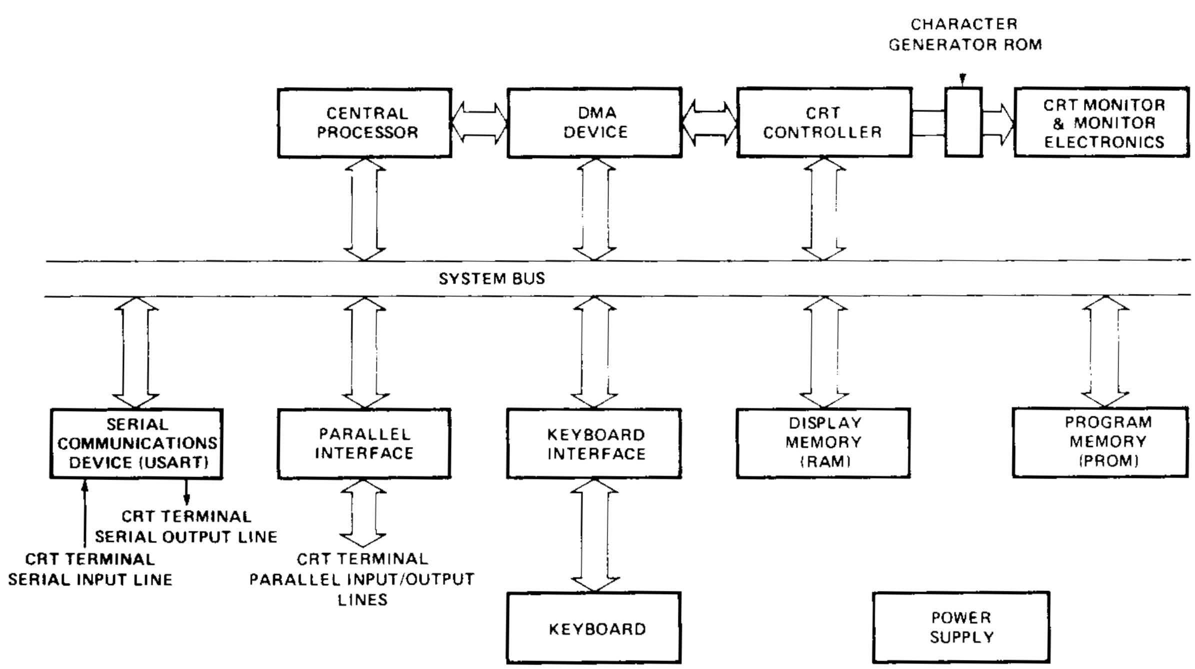


Figure 2-5. CRT Terminal Block Diagram

3. COMPONENT DESCRIPTION

3.1 8275

The block diagram and pin configuration for the 8275 Programmable CRT Controller are presented in Figure 3-1. The 8275 provides the following general capabilities:

1. *CRT Display Refreshing* – The 8275, having been programmed to a specific screen format, generates a series of DMA request signals, resulting in the transfer of a row of characters from display memory, via the 8257 DMA Controller, to the 8275's row buffers. The 8275 presents the character codes to an external character generator ROM. The 8275 character code outputs CC0–CC6 are used for this purpose. External dot timing logic is then utilized 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 graphically illustrated in Figure 3-2. The entire process is repeated for each display row. At the beginning of the last display row, the 8275 issues an interrupt via the INT 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 consists of 8 commands:

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

In order 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 from 1 to 16 horizontal lines per row are available.

In addition to transferring characters from memory 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 4 cursor formats. Blinking or non-blinking underline and reverse video block cursors are available.

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 are active, a third timing output, VSP (Video Suppress) is true, providing a blanking 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 utilized 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.

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 are codes that can be used to generate graphics symbols without the use of a character generator. This is accomplished by selectively activating the Line Attribute outputs (LA0–LA1), the Video Suppression output (VSP), and the Light Enable output. 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 3-4.

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 3-5. 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 16 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 (CC0–6). The chosen attributes are also activated.

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 activated. If the output of the light pen is presented to the 8275 LPEN input, the row and character position coordinates are stored in two 8275 internal registers. These registers can be read on command by the microprocessor.

<u>SPECIAL CODES</u> – 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.

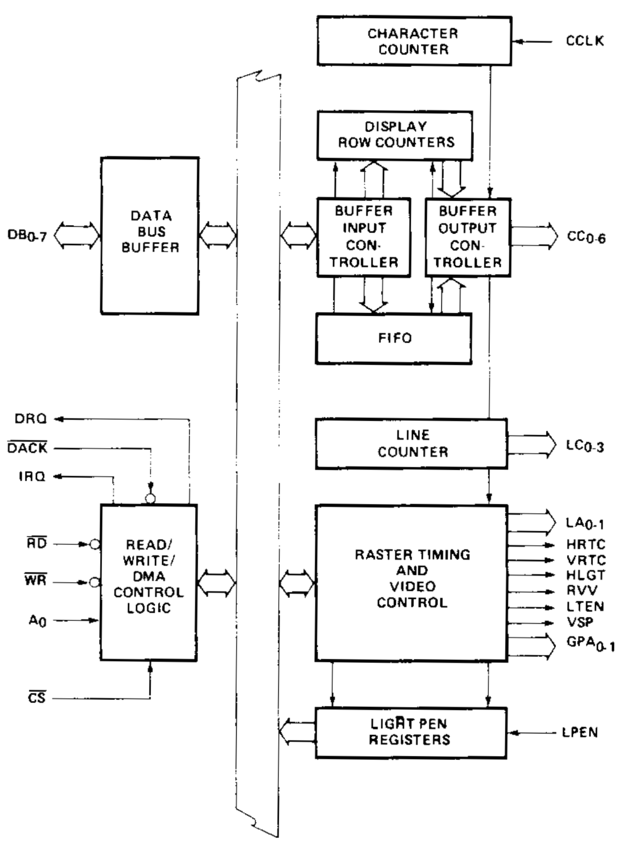
3. *End of Screen Code* –

   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.

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

BLOCK DIAGRAM



PIN CONFIGURATION

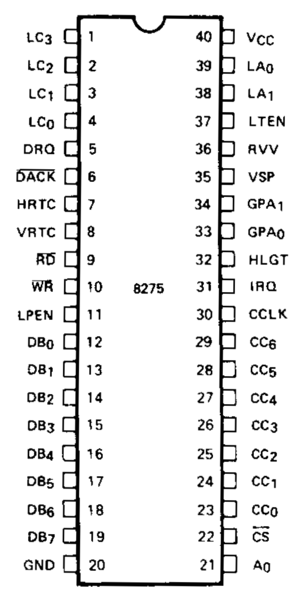


Figure 3-1. 8275 Block Diagram/Pin Configuration

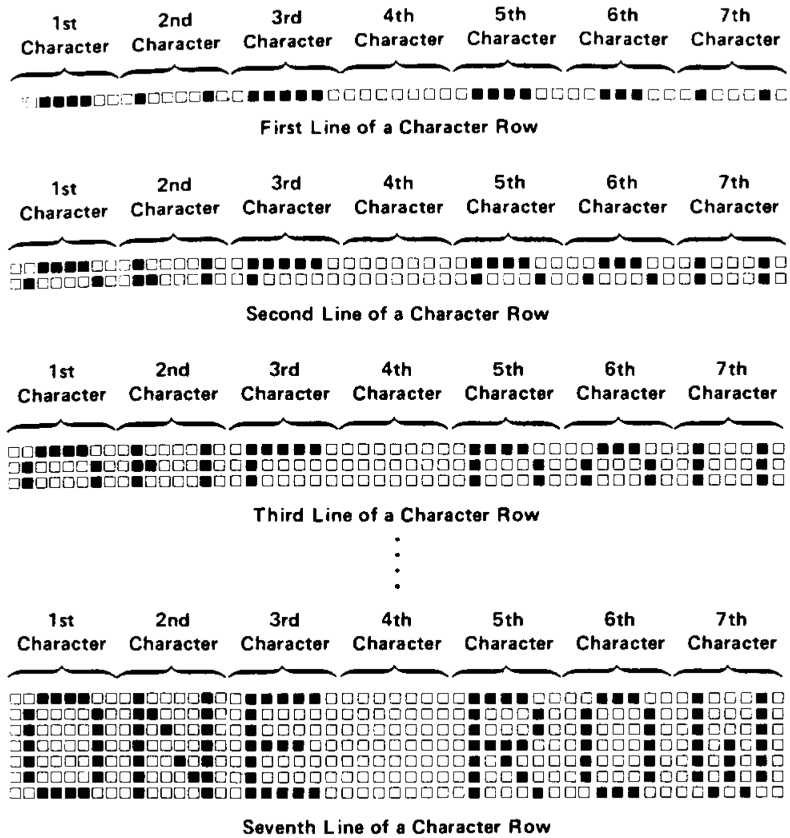


Figure 3-2. 8275 Row Display

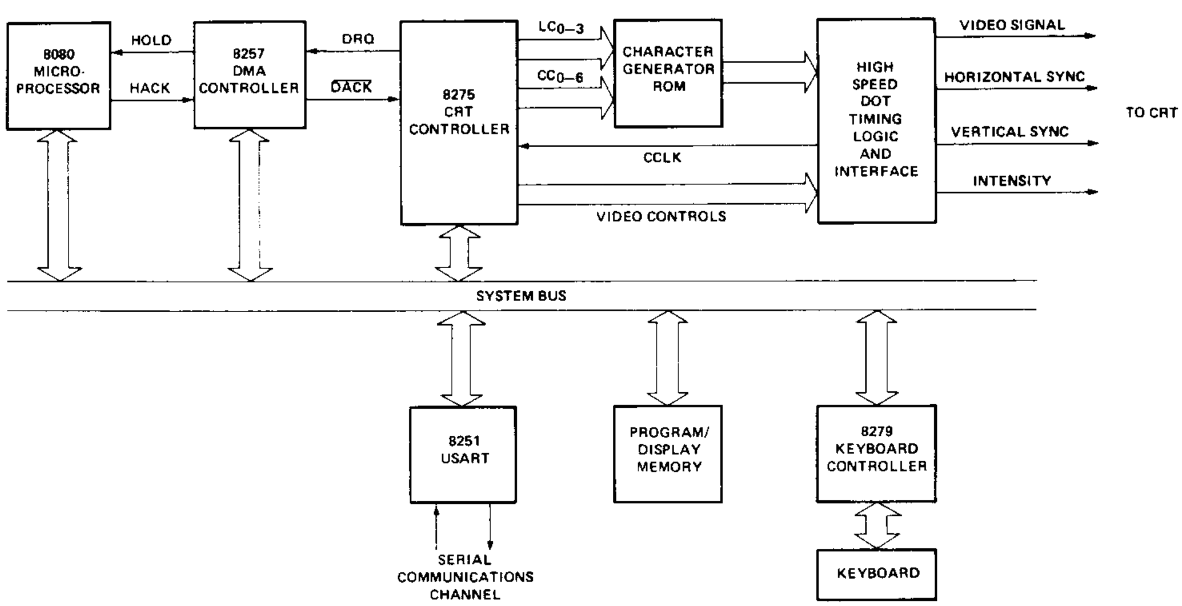


Figure 3-3. CRT System Block Diagram

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						

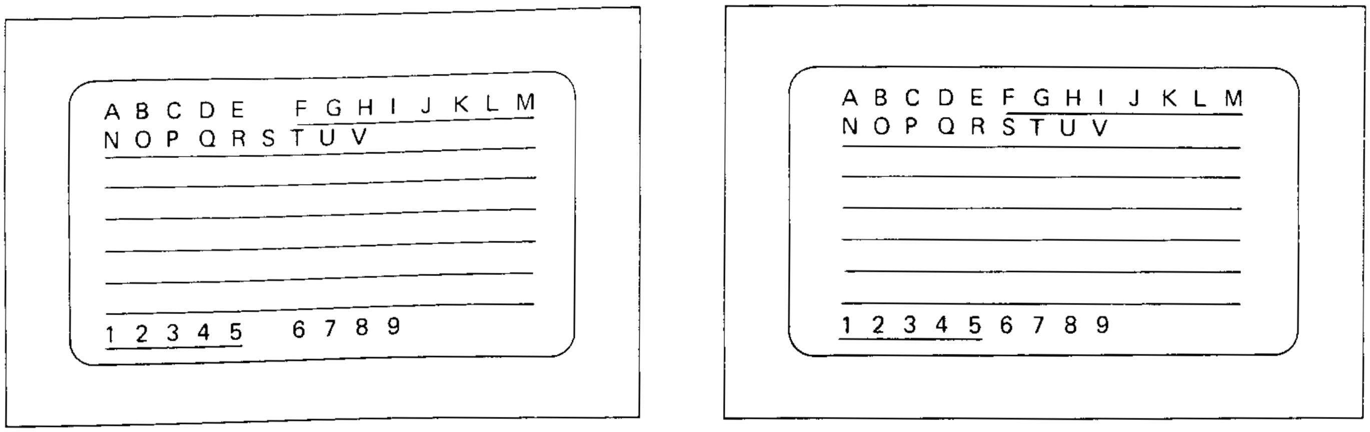
*Character Attribute Code 1011 is not recommended for normal operation. Since none of the attribute outputs are active, the character Generator will not be disabled, and an indeterminate character will be generated.

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

Blinking is active when B = 1.

Highlight is active when H = 1.

Figure 3-4. Character Attributes



EXAMPLE OF THE VISIBLE FIELD ATTRIBUTE MODE (UNDERLINE ATTRIBUTE)

EXAMPLE OF THE INVISIBLE FIELD ATTRIBUTE MODE (UNDERLINE ATTRIBUTE)

Figure 3-5. Field Attribute Examples

3.2 8279

The 8279 Programmable Keyboard/Display Interface block diagram and pin configuration are shown in Figure 3-6. The 8279 will be utilized in the CRT design example for performing keyboard scanning, key debounce, and data bus interface functions. Only features associated with these functions will be described in this section. The reader is referred to the 8279 data sheet for information on display control, sensor matrix mode operation, and strobed input mode operation. A detailed description of the 8279 keyboard scanning, debounce, and data bus interface functions follows.

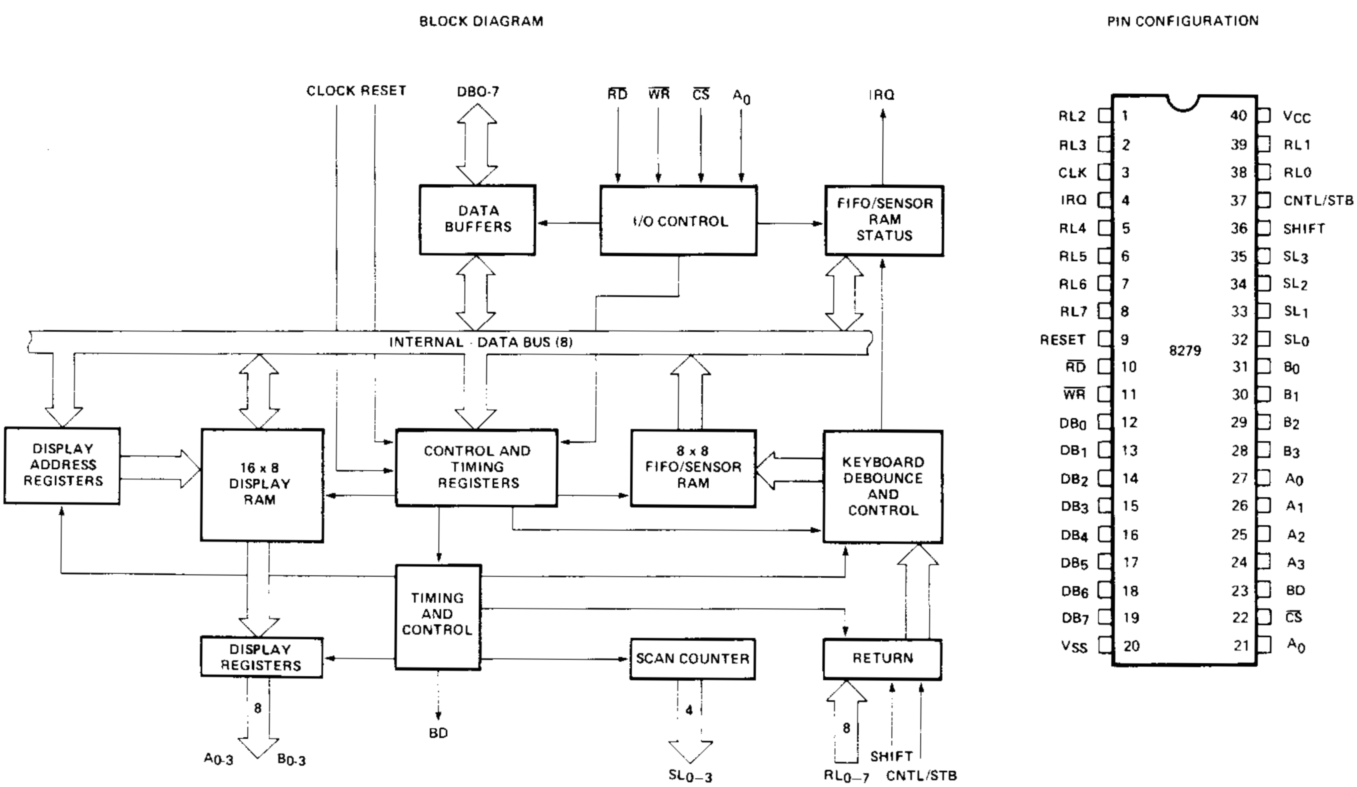


Figure 3-6. 8279 Pin Configuration and Block Diagram

The primary functions of the 8279 in the CRT system application include scanning the 64 key keyboard, determining if a key has been depressed, and, when polled by the system processor, transmitting the address of the key in the keyboard matrix to the master processor. Alternately, the interrupt line from the 8279 may be used to inform the CPU of a key depression. A block diagram of the 8279 interface, as implemented in the CRT system design example, is provided in Figure 3-7. The keyboard controller initiates the keyboard scanning process by transmitting keyboard scan line selection information over output lines SL_0-SL_2. The data may be encoded or decoded depending on the mode programmed. Assuming encoded mode is selected, the SL_0-SL_2 lines are connected to the input of a 3-line to 8-line decoder as shown in Figure 3-7. The decoder outputs are connected to the keyboard row inputs. Only one decoder output will be enabled for a given set of input conditions. The keyboard column outputs are connected to the 8279 return line inputs RL_0-RL_7. The eight return lines are buffered and latched by the 8279. These lines are scanned by the internal logic of the 8279, looking for a key depression in the selected row. If the debounce circuit detects a key depression, it waits approximately 10 ms to determine if the key remains down. If it does, the address of the key in the matrix plus the status of the shift and control lines are transferred to the 8279 FIFO. The FIFO data format is shown in Figure 3-8. The FIFO will hold up to eight data bytes; that is, up to eight key depressions may occur prior to a CPU initiated read operation. The number of characters entered into the FIFO is indicated by the character count contained within the FIFO status word. When a key depression is detected, the 8279 interrupt line goes high, and the FIFO status is modified to reflect the number of characters contained in the FIFO. The CPU may determine the occurrence of a key depression in one of two ways: The 8279 interrupt line may be connected to the interrupt input line of the CPU, forcing the CPU to call an interrupt service routine which reads the FIFO character. An alternate approach requires the CPU to periodically poll the 8279, reading the FIFO status word. If the FIFO character count is non-zero, indicating that at least one character is present in the FIFO, the CPU then reads the FIFO contents. This approach will be utilized in the CRT design example. A read operation places the contents of the FIFO on the system data bus and decrements the FIFO character count, contained within the FIFO status word, by one.

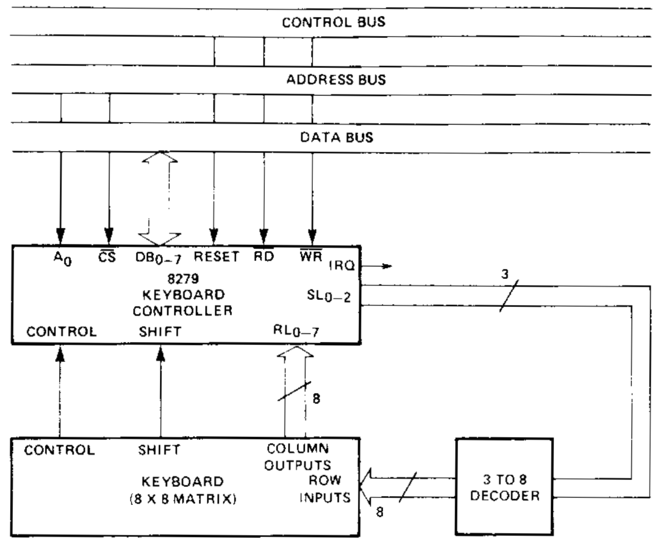


Figure 3-7. 8279 Interface

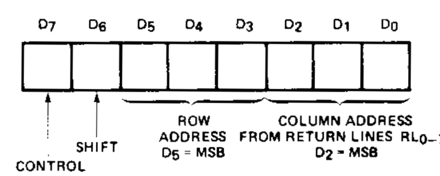


Figure 3-8. FIFO Data Byte Format

4. CRT SYSTEM DESIGN EXAMPLE

4.1 SCOPE OF THE PROJECT

A fully operational, microcomputer-based CRT terminal was designed and constructed utilizing the 8275 CRT Controller and 8279 Keyboard Controller as the basic system elements. The terminal incorporates the majority of the functions found in existing dedicated computer terminals. An Intel® 8080A microprocessor was utilized as the CPU in the design. The recently announced Intel® 8085 microprocessor constitutes an ideal processor for future CRT terminal designs. LSI devices were utilized in the design whenever possible in order to minimize component count.

4.2 SYSTEM SPECIFICATIONS

The specifications for the CRT terminal design are as follows:

Display Format

- 80 characters/display row
- 25 display rows

Character Format (Figure 4-1)

- 5X7 character contained within a 7X10 matrix, 1st and 10th lines blanked, 1st and 7th columns blanked, 9th line cursor position, blinking underline cursor.

Characters Recognized

- Displayable characters: 64 ASCII upper-case alphanumeric characters
- Control characters:
  Line feed, Control J
  Carriage return, Control M
  Back space, Control H
- Escape Sequences:
  Cursor up, ESC, A
  Cursor down, ESC, B
  Cursor right, ESC, C
  Cursor left, ESC, D
  Clear screen, ESC, E
  Home, ESC, H
  Erase to end of screen, ESC, J
  Erase line, ESC, K

Characters Transmitted

- 64 ASCII upper-case alphanumeric characters
- ASCII Control Character set
- ASCII Escape Sequence set

Program Memory

- 2K bytes, 2716 EPROM

Display/Buffer/Stack Memory

- 2K bytes, 2114 static RAM

Data Rate

- 4800 BAUD maximum using 8080A

CRT Monitor

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

Keyboard

- Microswitch hall effect keyboard, open collector outputs

Scrolling Capability

- Scroll up feature implemented with 8257 DMA Controller

Screen Refresh Rate

- 60 Hz

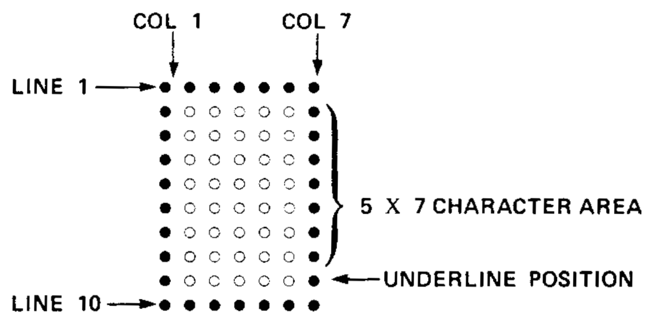


Figure 4-1. Character Format

4.3 SYSTEM HARDWARE DESIGN

4.3.1 General Considerations

A block diagram of the CRT terminal is presented in Figure 4-2. The diagram includes only essential system features. A detailed schematic of the CRT terminal is contained in the appendix. The terminal was constructed using an Intel® SDK-80 microcomputer kit and an Intel® SBC 905 prototyping board. The standard 8080 bus structure incorporated in the SDK-80 kit allowed the CRT terminal to be implemented with minimum buffering.

In the ensuing discussion of CRT terminal operation, it will be assumed that the terminal normally communicates with a remote device, such as an Intel® MDS microcomputer development system. Communication will take place in the full duplex mode. The CRT terminal, upon transmitting a character to the remote device, will remain idle until a character is received from the external device. Transmission of a character to the remote device is initiated by depressing a key on the keyboard. Character transmission to the CRT terminal from the remote device is assumed to be asynchronous with respect to terminal operation.

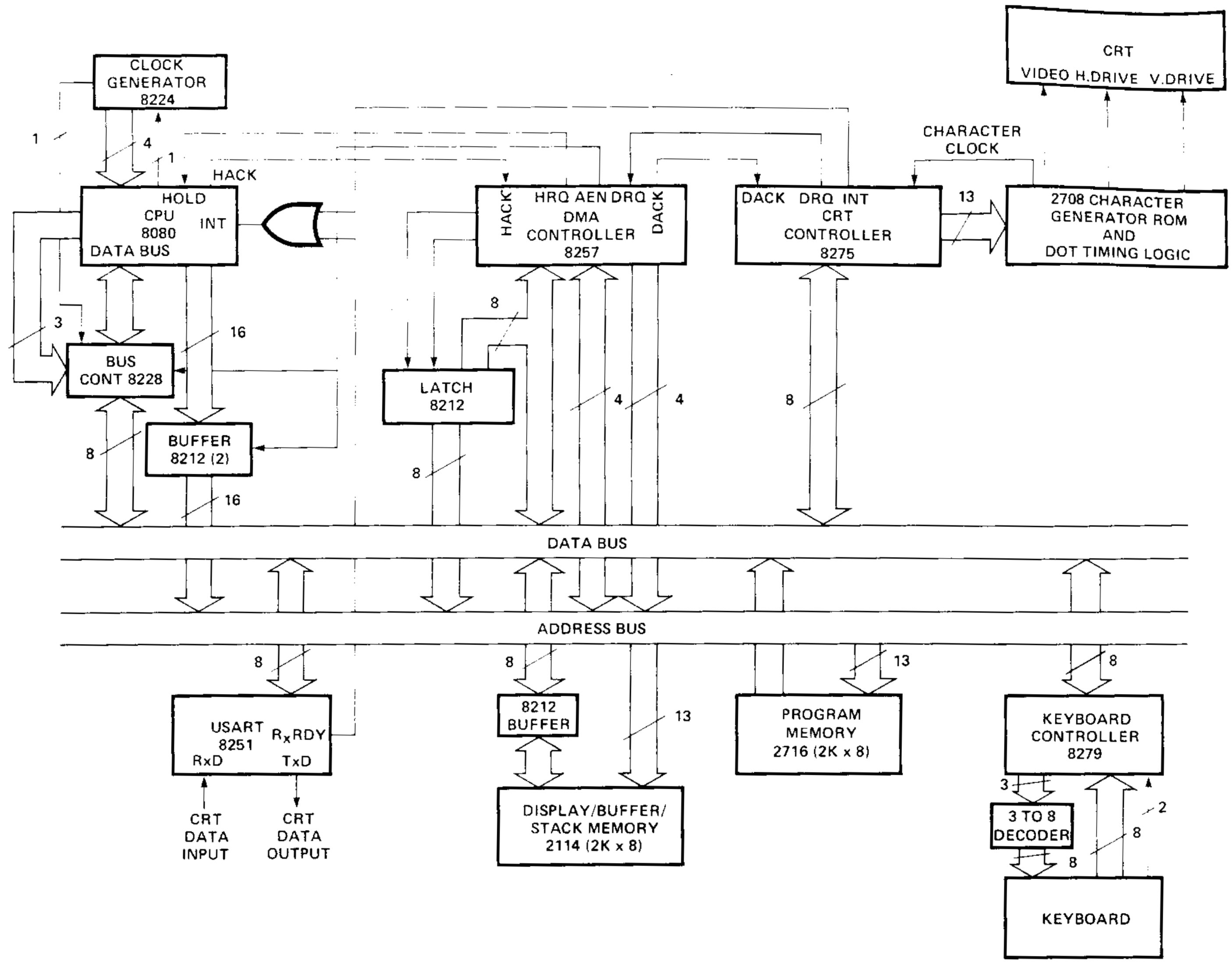


Figure 4-2. CRT Terminal Block Diagram

4.3.2 Operation

The 8080A CPU initializes each peripheral to the appropriate mode of operation following system reset. Upon receiving a character from a remote device, the 8251 USART issues an interrupt to the CPU. The CPU calls the interrupt service subroutine, which polls both the 8275 and 8251 to determine the source of the interrupt. Having determined that the 8251 issued the interrupt, the CPU calls the READ/STORE USART character subroutine, reads the USART character, and stores the character in buffer memory. The character recognition subroutine is called next. This routine determines whether the character is a displayable character, a control character, or a character in an escape sequence. Assuming the character is a displayable character, the CPU places the character in display memory at the location corresponding to the present cursor position, advances the cursor, modifies the display memory pointers, and, if required, performs the operations necessary for scrolling. If the received character is a control character or escape sequence character requiring cursor and display memory pointer changes, these functions are carried out. Escape sequences which involve erasing a portion of the display are also handled via the appropriate subroutines.

In order to place characters contained in display memory on the CRT display screen, the 8275 CRT Controller must first transfer the display characters, via the 8257 DMA Controller, to the 8275's row buffers. It should be noted that the 8257 DMA Controller is required to achieve the data transfer rate necessary for CRT refreshing. Display characters are then transferred from the 8275 row buffers to the character code outputs CC0–CC5. The character code outputs are applied to the character generator address lines A3–A8 (Figure 4-3). Line count outputs LC0–LC2 from the 8275 are applied to character generator address lines A0–A2. It should be noted that the 8275 displays character rows one line at a time. The line count outputs are utilized 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. The process continues until the last line of the row under consideration is transferred to the dot timing logic.

The dot timing logic latches the 6-bit character code and 3-bit line count from the 8275 on positive transitions of the character clock and transfers this information to the character generator ROM. In systems requiring a greater number of lines/character, the fourth line count output would also be used. The 7-bit ROM output corresponds to the 7 dots which make up a line segment for a particular character. The ROM output is loaded into a parallel input-serial output shift register. The shift register is clocked at the dot clock rate (11.34 MHz) continuously. The shift register output constitutes the video input to the CRT. The character code outputs select the character to be displayed at a given character position in the display row. The character set consists of $2^6=64$ ASCII upper case alphanumeric characters.

The row by row transfer of character data from display memory to the 8275 continues until the beginning of the last display row. At this time the 8275 issues an interrupt to the CPU. The CPU polls both the 8275 and 8251. Having determined that the interrupt originated with the 8275, the CPU calls the 8275 interrupt subroutine. The 8275 interrupt subroutine re-initializes the 8257 DMA Controller starting address and terminal count parameters and polls the 8279 Keyboard Controller to determine if a key depression has occurred. If a key has been depressed, the CPU reads the key position data from the 8279, performs a table lookup, and transmits the appropriate ASCII character to the CRT data output via the 8251 USART. It should be noted that interrupts are generated by the 8275 every 16.67 ms for a 60 Hz screen refresh rate.

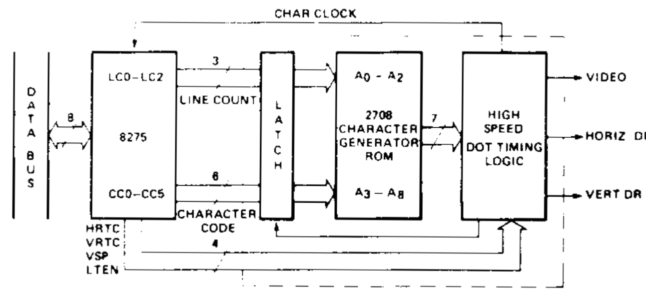


Figure 4-3. Character Generator/Dot Timing Logic Block Diagram

4.3.3 System Timing

The CRT terminal display raster is shown in Figure 4-4. It can be seen from the figure that a display row is composed of 10 lines. The Total Line Time consists of the display portion of the line plus the Horizontal Blanking Time. Row Time is equal to the number of lines per row multiplied by the Total Line Time. The Total Screen Time (1/Refresh Rate) is equal to the Row Time multiplied by the number of display rows plus the Row Time intervals associated with vertical blanking. Specifications for the BALL BROS. monitor show that there are constraints on the Vertical Blanking Time, Horizontal Blanking Time, and Horizontal Oscillator Repetition Rate. These constraints are summarized in Table 4-1.

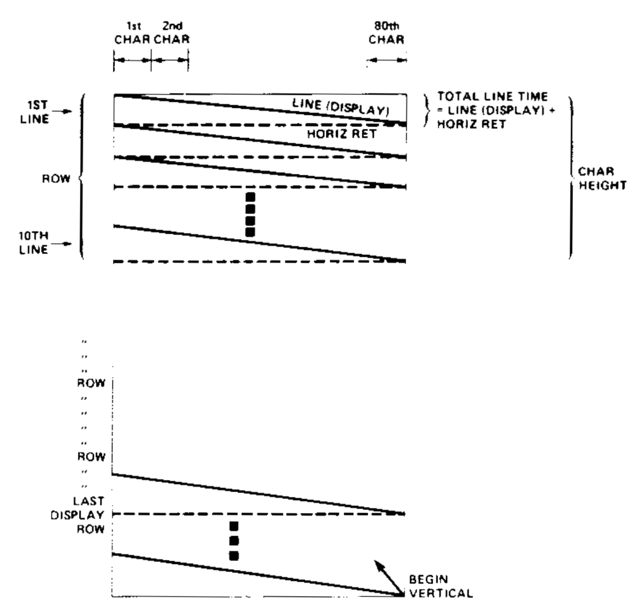


Figure 4-4. CRT Display Raster

Table 4-1

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

Given the constraints in Table 4-1 and the Refresh Rate specification of 60 Hz, the Vertical Retrace Row Count and Horizontal Retrace Character Count parameters required by the 8275 CRT Controller may be calculated:

$$\text{Total Screen Time} = \frac{1}{\text{Refresh rate}} = \frac{1}{60\text{ Hz}}$$
$$= 0.01667 \text{ sec}$$

Also,

$$\text{Total Screen Time} = (\text{Row Time})(\#\text{ of Display Rows}) + \text{Vertical Blanking Time (VRTC)}$$

Vertical Blanking Time (VRTC) must be an integral number of Row Times (between 1 and 4).

Therefore,

$$0.016667 \text{ sec} = (\text{Row Time})(25) + \text{VRTC}$$
$$= (\text{Row Time})(25) + \text{N (Row Time)}$$

If N is selected to be 2, the following result is obtained:

$$\text{Row Time} = 6.17284 \times 10^{-4} \text{ sec}$$

Therefore,

$$\text{VRTC} = (2)(\text{Row Time}) = 12.3457 \times 10^{-4} \text{ sec}$$
$$= 1.23457 \text{ ms}$$

Since the Vertical Blanking Time, nominally 900 μsec, falls within the constraints for the Vertical Drive Pulsewidth, the VRTC output from the 8275 may be used directly for the Vertical Drive Pulse. The 8275 will be programmed for a Vertical Retrace Row Count of 2.

In order to calculate the Horizontal Retrace Character Count, it is necessary to consider the row format as defined in the specifications. Figure 4-5 shows three adjacent characters in a row. The row, as shown, is composed of 10 Lines/Row and 7 Dots/Line/Character. Given that the Row Time is 617.284 μsec, the Total Line Time may be calculated as follows:

$$\text{Total Line Time} = \frac{\text{Row Time}}{\#\text{ Lines/Row}}$$
$$= \frac{617.284 \times 10^{-6}\text{sec}}{10}$$
$$= 61.7284 \times 10^{-6} \text{ sec}$$
$$= 61.7284\ \mu\text{sec}$$

The Total Line Time is composed of the display portion of the line plus the Horizontal Blanking Time (HRTC).

$$\text{Total Line Time} = 61.7284 \times 10^{-6} \text{ sec}$$
$$= 80 \left(\frac{\text{Character Time}}{\text{line}}\right) + \text{HRTC}$$

Horizontal Blanking Time (HRTC) must be an integral number of Character Times/Line.

Then

$$61.7284 \times 10^{-6} \text{ sec} = 80 \left(\frac{\text{Character Time}}{\text{line}}\right) + \text{M} \left(\frac{\text{Character Time}}{\text{line}}\right)$$

If M is selected to be 20, the following result is obtained:

$$\left(\frac{\text{Character Time}}{\text{line}}\right) = \frac{61.7284 \times 10^{-6}}{80 + 20}$$
$$= 6.1728 \times 10^{-7} \text{ sec}$$
$$= 617.284 \text{ ns}$$

This value defines the period of the 8275 character clock.

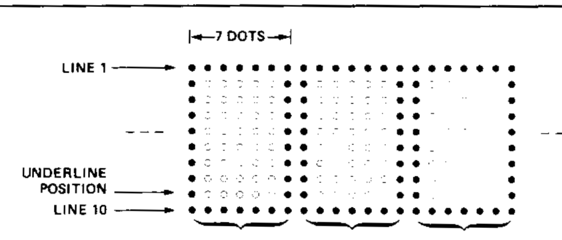


Figure 4-5. Row Format

The Horizontal Blanking Time (HRTC) is calculated as follows:

$$\text{HRTC} = 20\,(617.284\text{ ns})$$
$$= 12.3456\,\mu\text{sec (nominal value 11 }\mu\text{sec)}$$

The 8275 will be programmed for a Horizontal Retrace Character Count of 20. Since the specifications call for a Horizontal Drive Pulsewidth of 25–30 μsec, an external oneshot is required. The oneshot is triggered by the leading edge of HRTC.

Using the value for the Character Time/Line, the Dot Clock Rate may be established. It should be noted that the clock is used to shift data from the parallel in-serial out shift register (contained in the dot timing logic) to the CRT video input. The system character clock is also derived from the Dot Clock.

The dot clock is calculated as follows:

$$\left(\frac{\text{Dot Time}}{\text{line}}\right) = \frac{\left(\frac{\text{Character Time}}{\text{line}}\right)}{\#\,\text{dots/character}}$$
$$= \frac{6.17284 \times 10^{-7}}{7}\text{ sec}$$
$$= 8.8183 \times 10^{-8}\text{ sec}$$
$$= 88.183\text{ ns}$$

$$\text{Dot Clock Frequency} = \frac{1}{\frac{\text{Dot Time}}{\text{Line}}} = 11.34\text{ MHz}$$

The Horizontal Oscillator Repetition Rate may be calculated as follows:

$$f_{\text{Horiz}} = \frac{1}{\text{Total Line Time}} = \frac{1}{61.7284 \times 10^{-6}\text{ sec}}$$
$$= 16{,}200\text{ Hz}$$

This value falls within the system specification of 15,750 ±500 pps.

4.3.4 Dot Timing Logic

The primary function of the dot timing logic, illustrated in Figure 4-6, is to transfer the output of the character generator ROM to the video input of the CRT. Due to the high data transfer rate (11.34 MHz), logic external to the 8275 is required for this function. The data transfer operation is accomplished as follows: The character generator ROM output is applied to the parallel input lines of the 74166 shift register, the shift register is loaded synchronously with respect to the positive-going edge of the character clock, and data is clocked out of the 74166 serial input at the dot clock frequency. The 74166 output is applied, through appropriate gating logic, to the CRT video input. In addition to the previously described functions, the dot timing logic provides the timing signals required for transferring characters from the 8275 character code and line count outputs to the character generator ROM, implements the video suppress and light enable gating functions, and generates the system dot and character clocks.

In order to understand the dot timing logic design process, it is necessary to refer to Figure 4-6 and Figure 4-7.

It can be seen from the timing waveforms of Figure 4-7 that the character code output from the 8275 will be valid 150 ns (worst case) after the negative-going edge of the character clock. The character generator ROM output will be valid, assuming a direct connection between the 8275 and the ROM, 450 ns (worst case) after the character code appears at the address inputs. Total delay from the negative-going edge of the character clock until ROM output data becomes available is then 600 ns. Given the character clock width of 617 ns and external logic propagation delays and setup times, it becomes difficult to latch the ROM output for the first display character during the first character clock period. In order to alleviate this situation, a data pipelining technique is utilized. The timing for this technique is shown in Figure 4-7. A latch, introduced between the 8275 and the character generator ROM as shown in Figure 4-6, samples character code and line count data from the 8275 1/2 dot clock (45 ns) after the positive-going edge of the character clock. Data from the latch is applied to the character generator ROM address lines yielding, after a 450 ns delay (worst case), the appropriate 7-bit code at the ROM output. ROM data is loaded into the 74166 shift register on the next positive-going edge of the character clock. This technique effectively delays the video output from the shift register by 1½ character clocks, but eliminates the difficulties in sampling the ROM data within the first character clock period. Due to the video delay associated with this technique, it is also necessary to delay all signals affecting the video output and CRT timing. These signals include HRTC, VRTC, VSP, and

LTEN. The delay is accomplished using a two-stage shift register constructed with edge triggered D flip-flops (74175). The system dot clock (11.34 MHz) is obtained by dividing the 22.68 MHz output from the 8224 clock generator by two. The dot clock is utilized to clock the 74166 output shift register and is divided by 7, using a 74S163 counter, to produce the system character clock. It should be noted that the use of a bipolar character generator PROM such as the Intel® 3604 or 3608 will reduce the external dot timing logic package count due to the reduced access time.

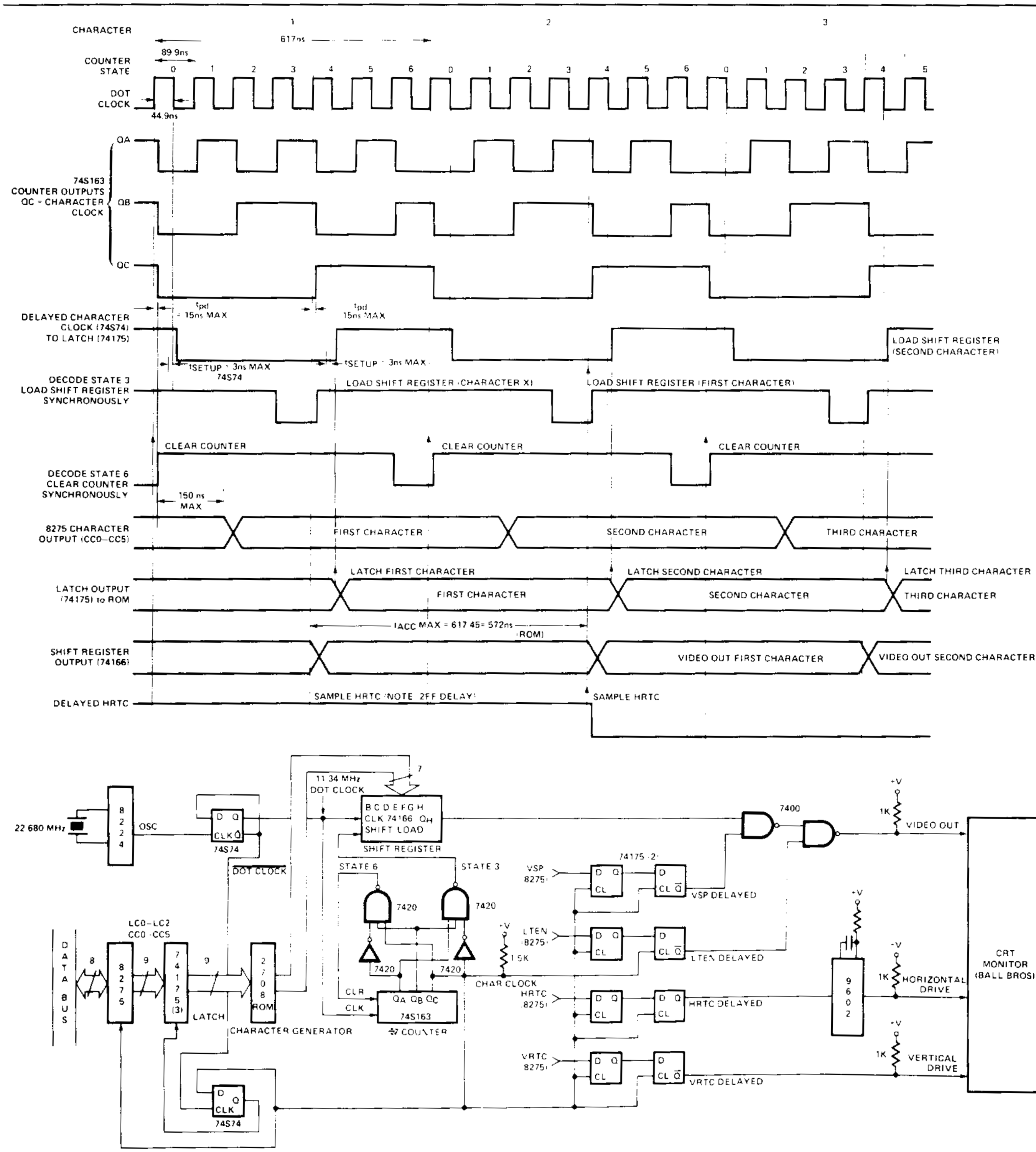


Figure 4-6. Dot Timing Logic

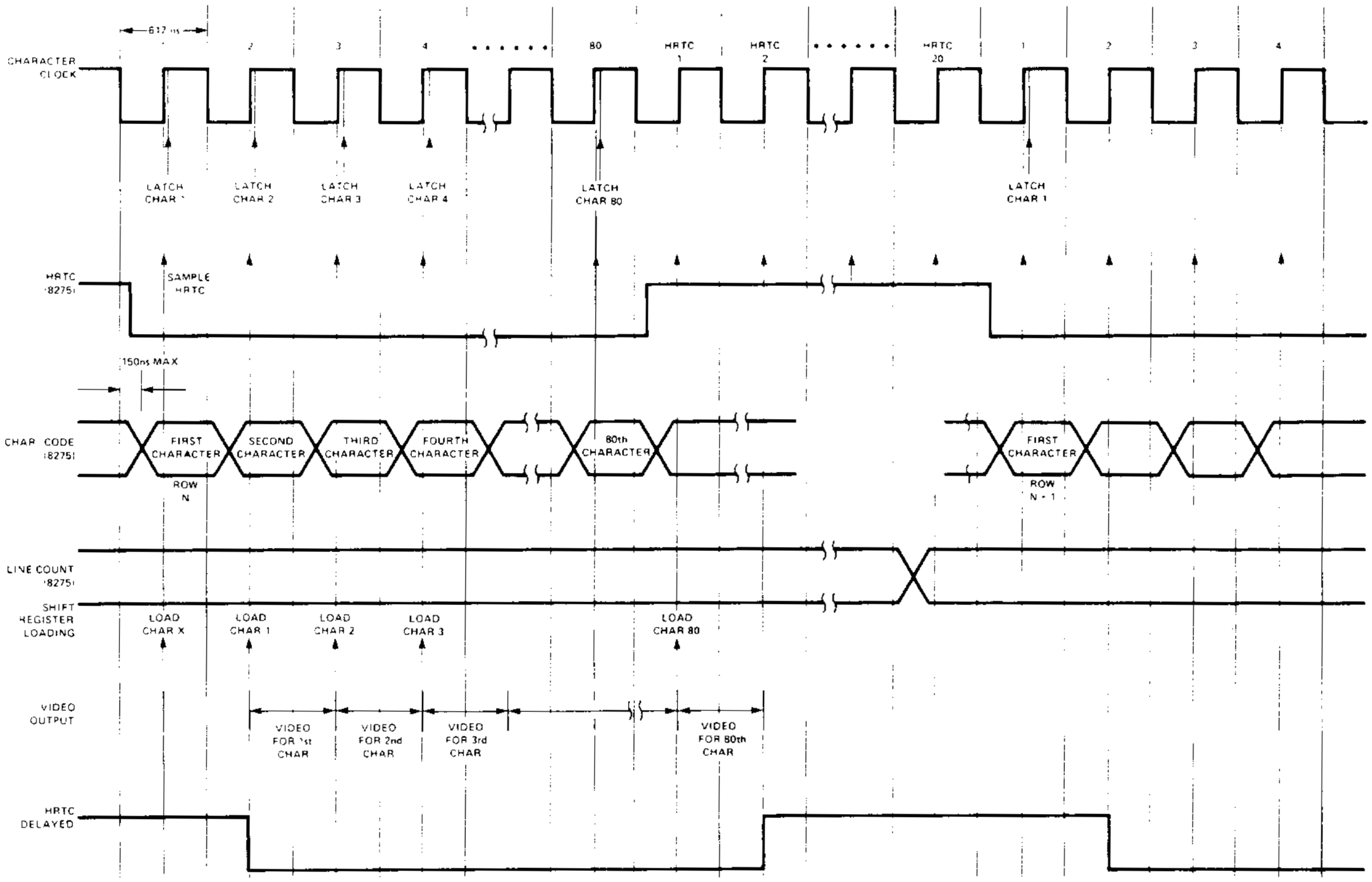


Figure 4-7. CRT System Timing

4.3.5 Keyboard Interface Design

The keyboard interface, Figure 4-8, consists of the 8279 Keyboard Controller and the decoding logic necessary for scanning the keyboard matrix. The 8279 SL_0–SL_2 output lines are decoded by the 74S138 decoder. The eight output lines from the decoder select 1 of 8 keyboard matrix rows for testing by the 8279. The keyboard matrix column output lines are connected to the 8279 return lines, RL_0–RL_7. Open collector outputs presented by individual keys within the matrix eliminate the need for isolation diodes when two keys in a given column are depressed. Two-key rollover was chosen as the operating mode for the 8279.

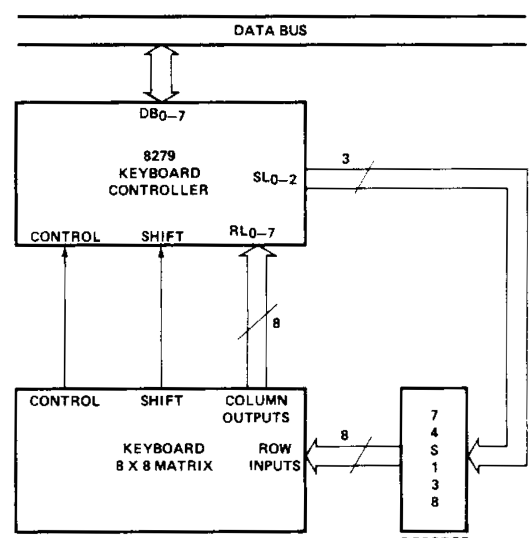


Figure 4-8. Keyboard Interface

4.3.6 System Memory Design

The system memory, illustrated in Figure 4-9, consists of one 2716 EPROM used for program storage and four 2114 RAMs used for display memory, buffer memory, and system stack. The 2114 4K static RAM was chosen for the design because of its 1K X 4 organization, ease of use, and availability. Buffering between RAM memory and the system data bus was used to minimize bus loading.

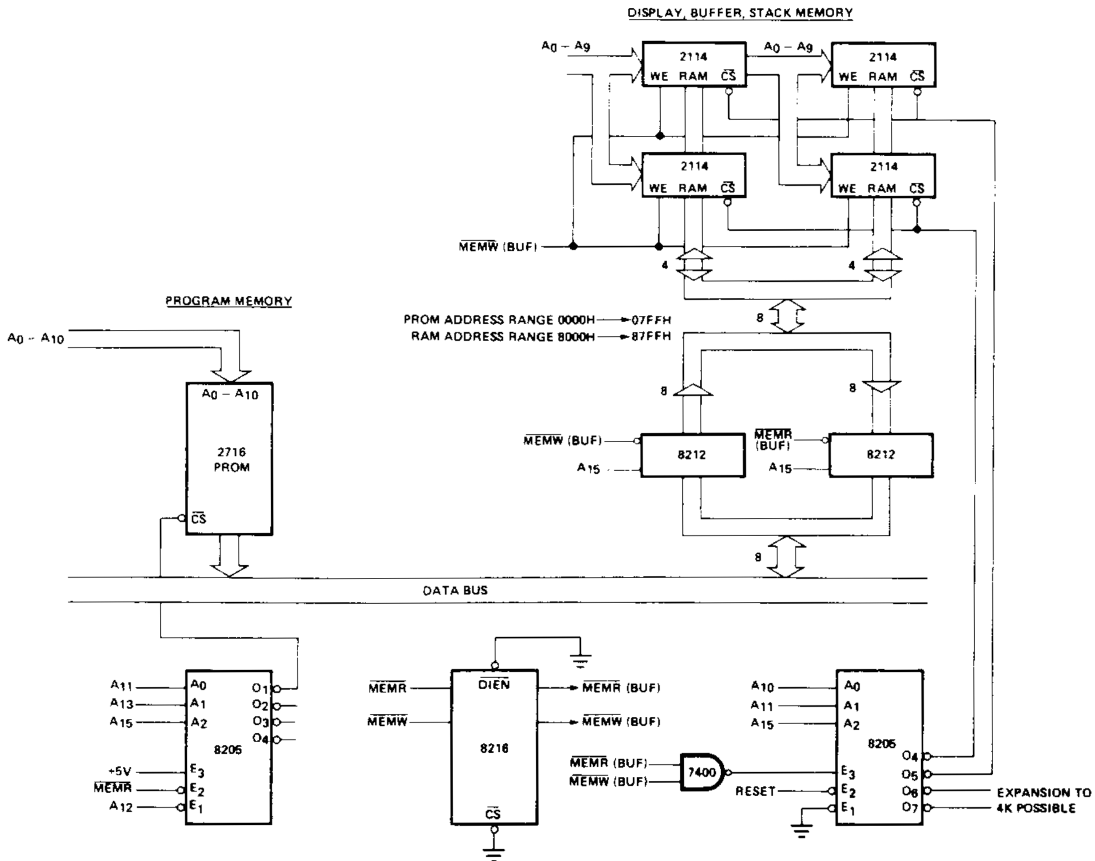


Figure 4-9. System Memory

4.4 SYSTEM SOFTWARE DESIGN

4.4.1 General Considerations

The approach taken in presenting the system software design is as follows: First, the software development process will be outlined. A discussion of system software operation will then be undertaken. Software operation will be followed by a detailed presentation of system subroutines.

4.4.2 Software Development

Software development was accomplished using the following tools:

1. Intel® MDS microcomputer development system
2. Intel® dual floppy disc system
3. Intel® ICE-80 In-Circuit Emulator
4. Intel® ISIS II disc operating system

The MDS was utilized in conjunction with the dual floppy disc system for program editing, assembly, relocation, and loading functions.

The ICE module was used extensively for loading assembled routines into the prototype system RAM and debugging program errors. While in the emulation mode, the ICE processor controlled the operation of the CRT system. During debugging, emulation proceeded normally until certain user specified break conditions occurred, at which time ICE entered the interrogation mode. During interrogation mode all processor functions, including DMA, ceased, allowing the user to access and display CPU register contents, status, and up to 44 previous machine cycles, system memory contents, and I/O device data.

4.4.3 Operation

The fundamental operations performed by the CRT system software are presented in Figure 4-10. Extensive use of subroutines in implementing major software functions resulted in readily understandable software. Debugging operations were also simplified as a result of the software structure. At system reset, the central processor interrupt system is disabled, the program counter is set to zero, and peripheral reset functions are carried out. Following reset, the system software initializes all peripherals, clears buffer memory, initializes special buffer locations, fills display memory with space codes, and enables interrupts. The processor then loops until an interrupt arrives from the 8275 or 8251. When the processor detects the occurrence of an interrupt, the instruction being executed is completed, an RST 7 vector is placed on the system data bus, and the RST 7 call instruction is executed, forcing a jump to the starting address of the 8275/8251 interrupt polling routine. Once the polling routine establishes the source of the interrupt, program flow continues along one of the two possible paths shown in Figure 4-10. An 8275 interrupt causes the 8257 DMA Controller to be re-initialized, the 8279 Keyboard Controller to be serviced, and, if a key depression has occurred, a character to be transmitted to the terminal output. An interrupt from the 8251 will first cause the USART character to be read and stored in memory. The system software then examines the character to determine whether it is a displayable character, a control code, or the first or second character in an escape sequence. After determining the nature of the character, an appropriate subroutine is called. Following the completion of the routines associated with an 8275/8251 interrupt, interrupts are re-enabled and a return instruction executed. The CPU then loops until the receipt of an interrupt. In order to appreciate the operation of the system software in detail, it is necessary to consider the following items:

1. System memory organization.
2. The relationship between character position on the screen and screen pointers Row Count, Column Count, and memory pointer Top.
3. The relationship between memory pointers Row Count, Column Count and the 8275 cursor X and Y position registers.
4. Scrolling concepts, including the relation between scrolling, display memory, and the memory pointer Top.

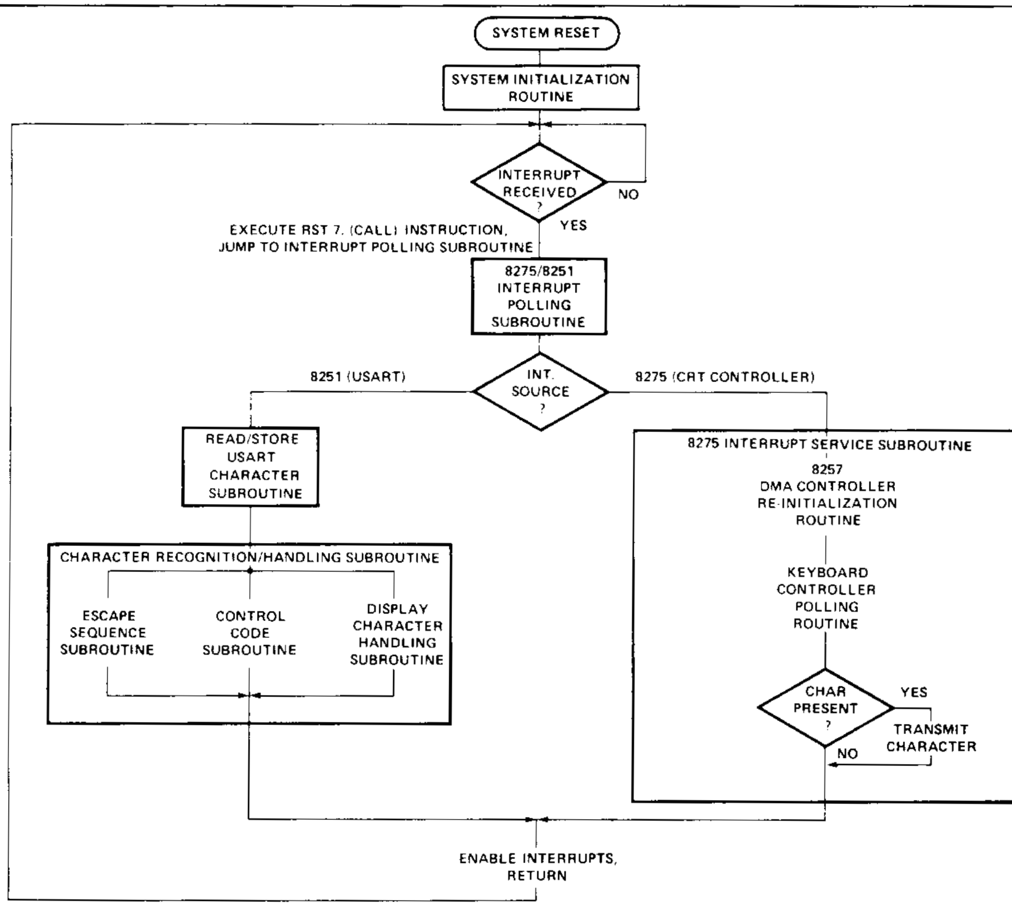


Figure 4-10. CRT Software Operations

System Memory Organization

System memory organization is shown in Figure 4-11. It should be noted that an additional 2K block of RAM was utilized for program memory (rather than PROM) during the software development/debug phase of system design.

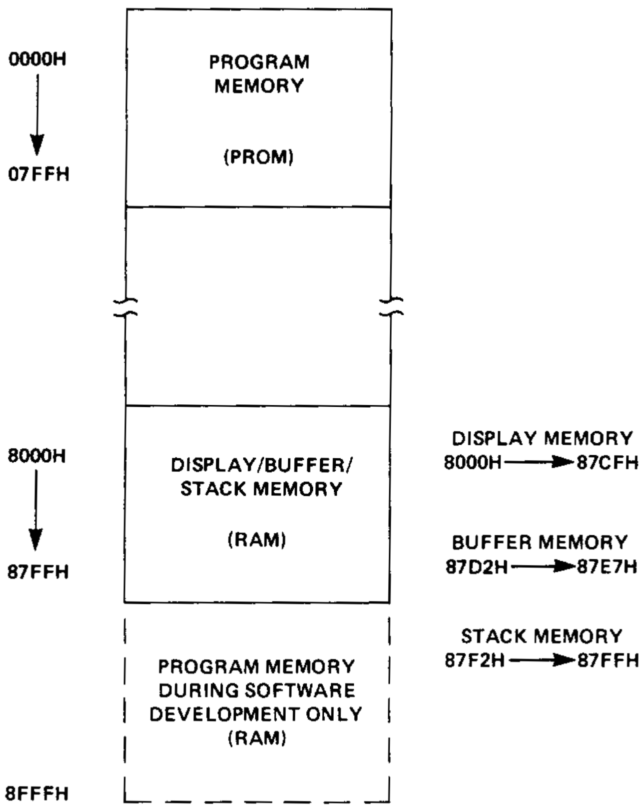


Figure 4-11. System Memory Organization

Character Position/Screen Pointer Relationships

To define the location of a character on the screen, two pointers, Row Count and Column Count, were created in memory. The relationship between character location on the screen and the two pointers is illustrated in Figure 4-12. Row Count and Column Count are stored in memory locations RCTAD and CCTAD, respectively. Row Count represents the position of the first character in a given row. For the first row, Row Count = 0000H. For the second row, Row Count = 0050H. Column Count represents the specific column in which the character is located. Character position on the screen may be calculated by adding the Row Count to the Column Count; e.g., the highlighted character in Figure 4-12 is located at AOH + 03H = A3H.

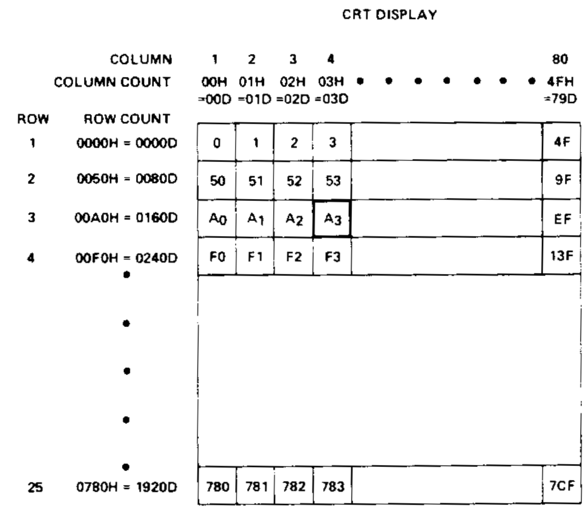


Figure 4-12. Character Location/Pointer Relationship

Memory Pointer/8275 Cursor Position Register Relationship

It was necessary to establish a relationship between Row Count and Column Count pointers and the 8275 Cursor X and Y Position registers for the cursor generated by the 8275 to be loaded at the appropriate position on the screen. This relationship is summarized in Table 4-2.

The value transferred to the 8275 for the Cursor X Position is identical to the Column Count. A new parameter, Cursor Y Position, stored at memory location CURSY, was also established. For a given Row Count value, a value for Cursor Y Position is defined. This value is transferred to the 8275 Cursor Y Position register.

It is necessary to introduce an additional parameter, Top, which will be used in conjunction with Row Count and Column Count to determine the location in display memory at which an incoming display character will be stored. The location at which a given character will be stored (assuming no more than 2000 characters have been entered since initialization) is calculated by adding TOP + Row Count + Column Count, where TOP is assumed to be 8000H, the starting location of display memory shown in Figure 4-11. Following system initialization, characters will be entered in display memory starting at memory location 8000H. The 2000th character will be entered at location 87CFH. Upon entering the 2001st character, a scrolling condition exists and TOP will be modified to point to memory address 8050H. An in-depth discussion of scrolling is presented in the next section.

Table 4-2

SCREEN POINTER/8275 CURSOR X,Y POSITION REGISTER RELATIONSHIP

ROW	ROW COUNT VALUE	CURSOR Y POSITION REGISTER VALUE	COLUMN	COLUMN COUNT VALUE	CURSOR X POSITION REGISTER VALUE
1	0000H	00H	1	00H	00H
2	0050H	01H	2	01H	01H
3	00A0H	02H	3	02H	02H
4	00F0H	03H	4	03H	03H
25	0780H = 1920D	18H = 24D	80	4FH = 79D	4FH = 79D

Scrolling

Scrolling is implemented in the CRT system design by shifting the entire display up by 1 row when a scrolling condition occurs. Scrolling will occur when certain cursor manipulation functions are exercised or when a character is entered in the last CRT display position, indicating a full memory page condition exists. Character entry will be used as the vehicle for explaining scrolling in the following discussion.

Characters are normally entered sequentially in display memory. When the 2000th character has been entered, display memory capacity has been attained; i.e., a full page condition exists. At this point, scrolling will take place. For scrolling to take place, DMA channel 2, the channel used to extract characters from display memory, must be re-initialized to the appropriate starting address and terminal count values. The memory pointer TOP will be used to establish the starting address for channel 2. Prior to scrolling, TOP = 8000H, the starting address of display memory. Each scrolling operation causes 80D (50H) to be added to TOP, moving the pointer, as shown in Figure 4-13b, to the beginning of the following row in display memory. It should be recalled that TOP, in conjunction with Row Count and Column Count determines the insertion address for incoming display characters. The net effect of modifying TOP is to shift the information being displayed on the CRT up by 1 row; i.e., scrolling is accomplished. Prior to scrolling, the terminal count value for DMA channel 2 is equal in magnitude to the display memory length −1 or 87CFH − 8000H. The actual value sent to the terminal count register is 87CFH − 8000H + 8000H. The addition of 8000H sets bit 14 in the terminal count register to a 1, indicating a DMA read operation. If scrolling is to be implemented, the terminal count value must be modified to 87CFH − TOP + 8000H. Characters transferred by channel 2 include those characters located from the address specified by TOP to the end of display memory. In order to transfer the characters from the beginning of display memory through the address immediately prior to TOP, the autoload feature of the 8257 DMA controller is utilized. When DMA channel 2 reaches terminal count, following the transfer of characters from TOP to the end of display memory, the starting address and terminal count parameters stored in the DMA channel 3 registers are loaded into channel 2. DMA operations resume in channel 2 using the channel 3 parameters. To accomplish the desired channel 3 operations, it is only necessary to re-initialize the channel 3 starting address to the beginning address of display memory, and the terminal count value to 87CFH, the maximum terminal count for a 2000-byte display memory space. These processes are performed during DMA re-initialization following an 8275 interrupt. New text entry following scrolling is illustrated in Figure 4-13. BOTTOM, a parameter corresponding to the address of the first character in the last row to be displayed, is utilized during clear to end of screen operations.

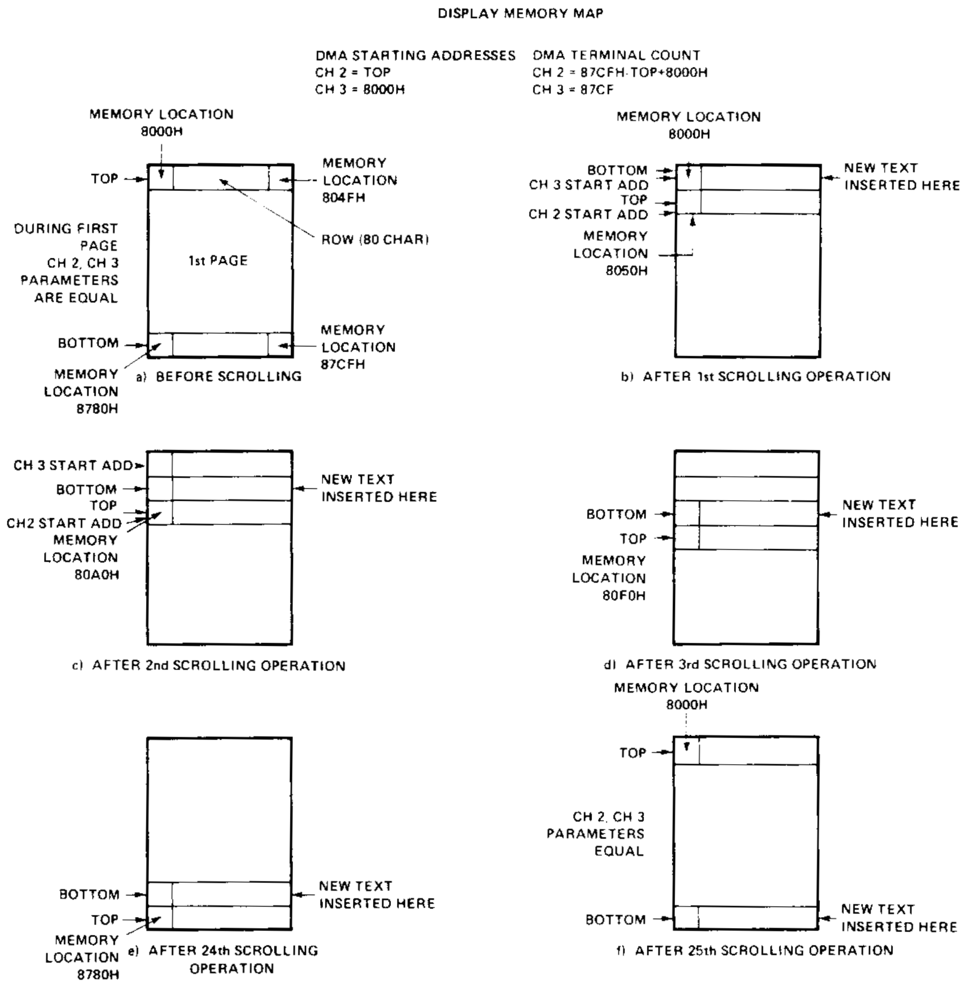


Figure 4-13. Pointer Manipulation During Scrolling

4.4.4 System Subroutines

System Initialization Routine (CRTGO)

The system initialization routine, Figure 4-14, establishes a starting point for system operation. The 8251 USART is initialized to transmit to and receive characters from an external device. The 8279 Keyboard Controller, at system reset, comes up in the two-key rollover mode. It is therefore only necessary to set up the Keyboard Controller internal operating frequency during initialization. Assuming a desired internal operating frequency of approximately 100 kHz and a 2.048 MHz system clock, the frequency divider chain is programmed to divide by 21. The 8275 initialization parameters are determined from the original CRT system specifications and vertical retrace Row Count/ Horizontal Retrace Character Count calculations previously performed. The delayed line number feature allows the use of only 3 line count outputs to determine which of 10 possible lines in a character row will be displayed. Given that the underline placement position is set to the ninth row, the top and bottom lines of the character are automatically blanked, leaving, effectively, 8 unique lines for display. The 8275 cursor position registers are initialized to zero, forcing the cursor to the upper left-hand corner of the display. The preset counters command resets all 8275 counters to zero and stops the 8275 counters until another command is issued. The 8275 is then started by a start display command. An interrupt will be generated from the 8275 approximately 15 ms later. Interrupts are enabled following the 8275 start command. Interrupts were disabled prior to this time to insure that the central processor did not react to erroneous interrupts from the 8275 generated prior to 8275 initialization. The processor, following initialization, waits in a loop until the arrival of an interrupt from the 8275 or 8251.

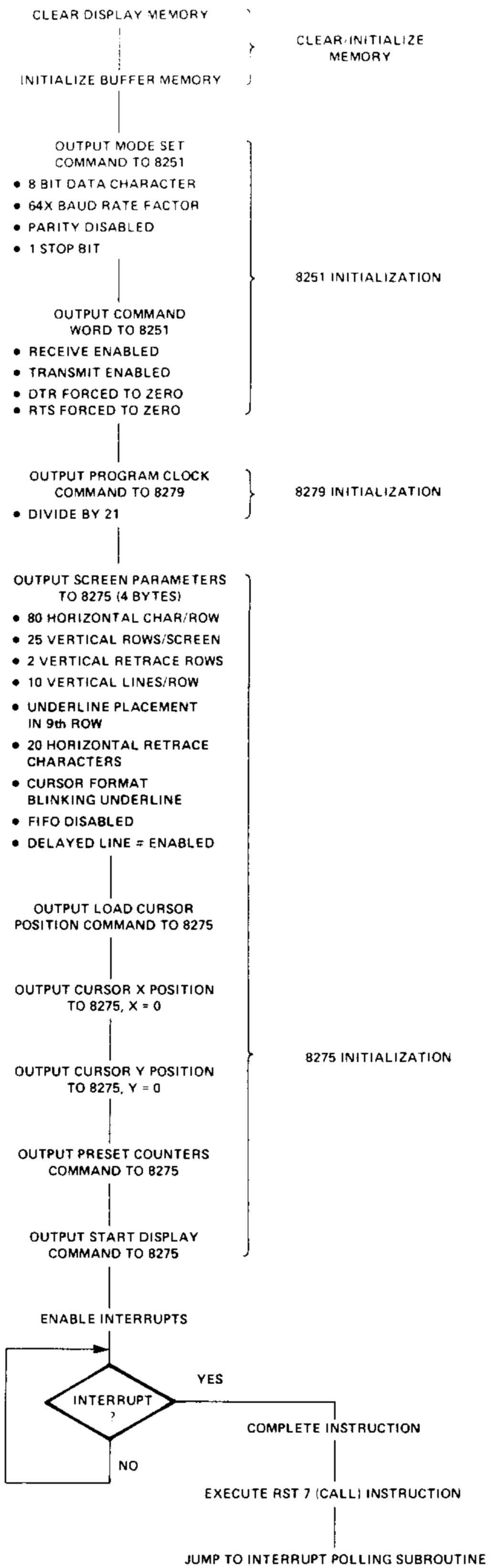


Figure 4-14. System Initialization Routines

Interrupt Polling Subroutine (Poll)

The interrupt polling subroutine, Figure 4-15, tests to determine the source of the interrupt. If the interrupt originated with the 8275, the 8275 interrupt service subroutine is called. Following completion of the subroutine, interrupts are re-enabled, and a return executed. An interrupt issued from the 8251 forces subroutine calls to the read/store USART character subroutine and the character recognition/handling subroutine. Interrupts are re-enabled at the completion of the character recognition/handling routine. A return operation follows.

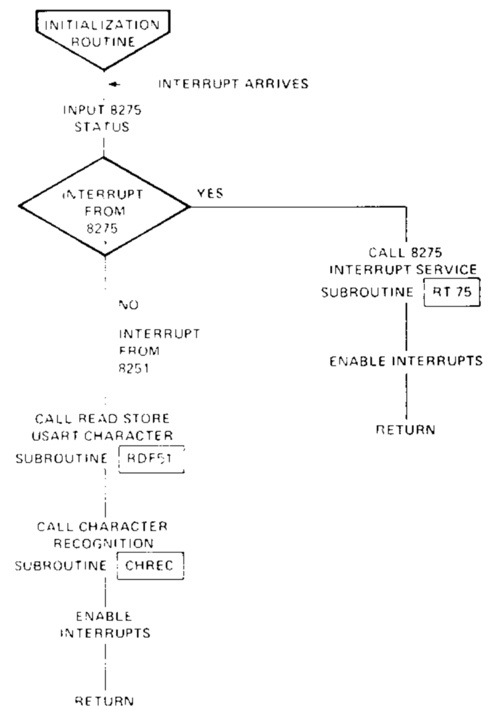


Figure 4-15. Interrupt Polling Subroutine (POLL)

8275 Interrupt Service Subroutine (RT 75)

The 8275 interrupt service subroutine, Figure 4-16, re-initializes the 8257 DMA Controller, then tests the 8279 FIFO status. If a character has been transmitted from the keyboard to the Keyboard Controller, a table lookup operation is performed to obtain the correct ASCII code for the character, and the character is transmitted.

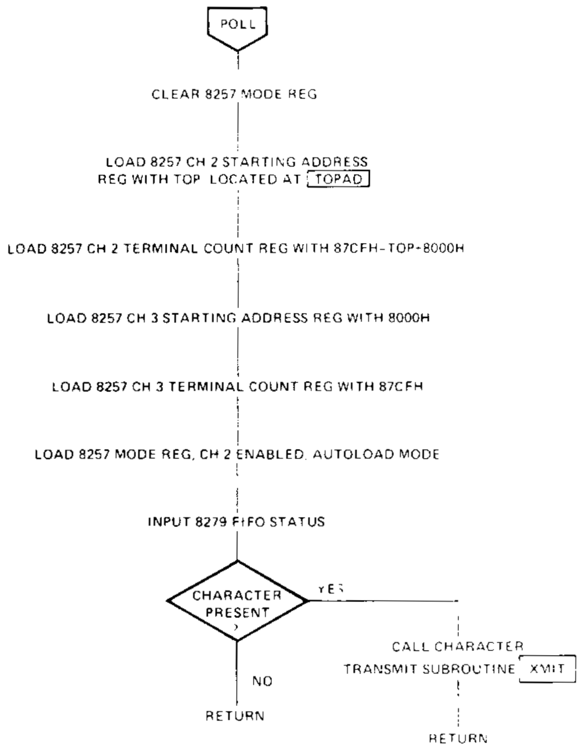


Figure 4-16. 8275 Interrupt Service Subroutine (RT75)

USART Read/Store Subroutine (RDF 51)

The read/store USART character subroutine, Figure 4-17, moves a character from the USART to the CPU, masks off the upper-most bit, and stores the character in system buffer memory.

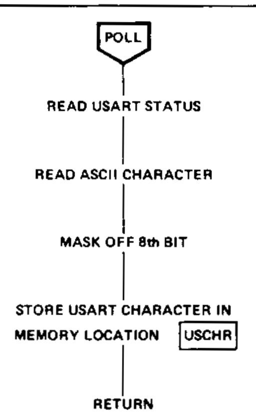


Figure 4-17. READ/STORE USART Character Subroutine (RDF51)

Character Recognition/Handling Subroutine (CHREC)

The character recognition/handling subroutine, Figure 4-18, examines the masked USART character to determine whether the character is a displayable character, control code, or the first or second character in an escape sequence. A call to the appropriate subroutine follows the decision-making process. If the character is the first character in an escape sequence, the escape sequence flag is set and the processor loops until a second character is received. The character immediately following the ESC character is examined by the escape code handling subroutine and a jump to an escape code routine follows. If the character is a displayable character or control code, the appropriate subroutine is called.

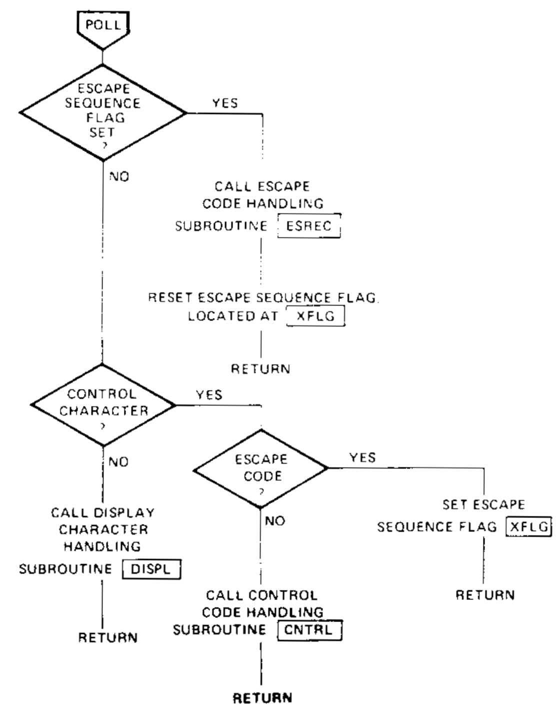


Figure 4-18. Character Recognition/Handling Subroutine (CHREC)

Escape Sequence Subroutine (ESREC)

The escape sequence subroutine, Figure 4-19, performs a masking operation on the USART character, shifts the result by one bit position, and adds this value to the base address of the escape sequence lookup table, BSETI. The lookup table contains starting addresses for each of the escape sequence routines. This address is jammed into the program counter and the routine executed. A summary of escape sequence functions is given in Appendix 5.2.

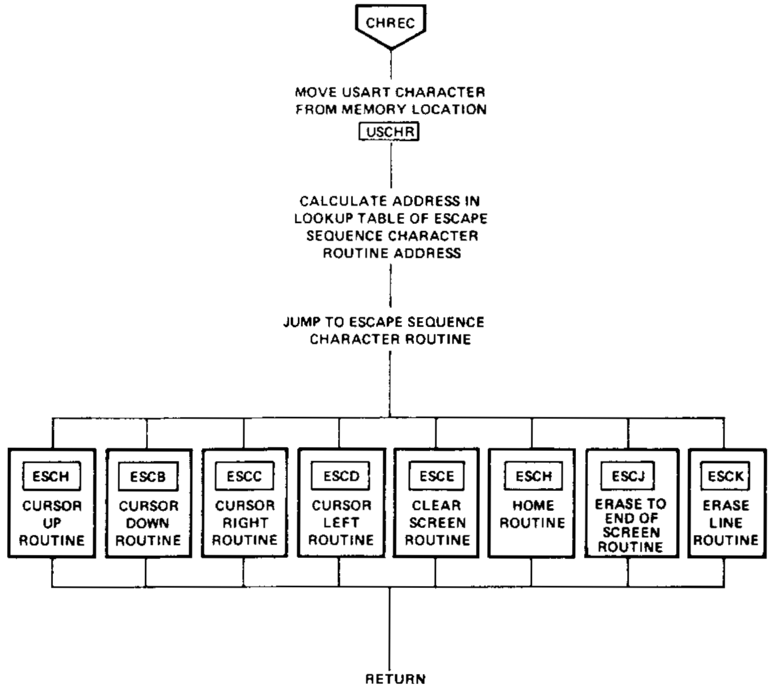


Figure 4-19. Escape Sequence Subroutine (ESREC)

Control Code Subroutine (CNTRL)

The control code subroutine, Figure 4-20, involves, conceptually, the same procedures executed by the escape sequence subroutine. A summary of control code functions is given in Appendix 5.2.

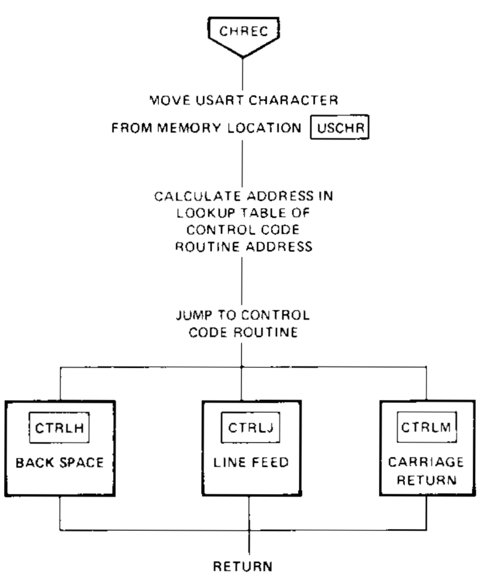


Figure 4-20. Control Code Subroutine (CNTRL)

Display Character Handling Subroutine (DISPL)

The display character handling subroutine, Figure 4-21, determines if the cursor is located in the last column of the row, the last display position, or elsewhere and calls the appropriate subroutines.

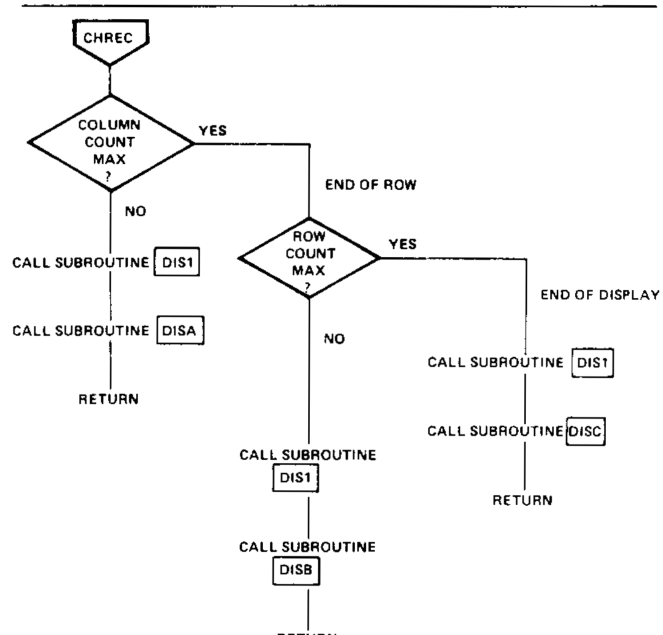


Figure 4-21. Display Character Handling Subroutine (DISPL)

Display Subroutine One (DIS1)

Display subroutine one, Figure 4-22, calculates the location in memory at which the display character is to be inserted. If the location calculation results in an address outside of the display memory bounds, appropriate compensation action is taken. Prior to inserting the display character in memory, the first character position in the row in which the character will be located is examined. If an End of Row character (EOR) is found, the row in question will be blanked by the 8275. It is necessary to clear the row by filling it with space codes (Fill Subroutine), then insert the display character in the desired location. If no EOR character is found, insertion proceeds without further software intervention.

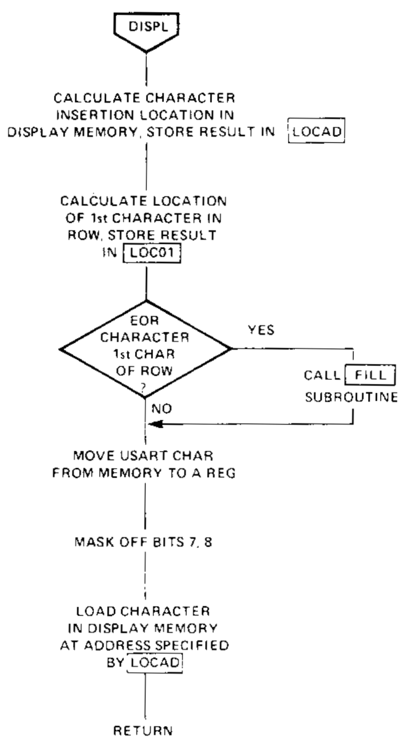


Figure 4-22. Display Subroutine 1 (DIS1)

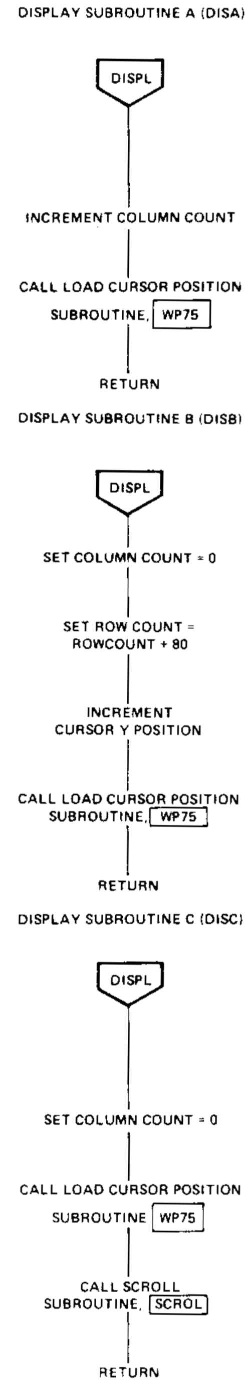


Figure 4-23. Display Subroutines — A (DISA), B (DISB), C (DISC)

Display Subroutines A, B, C (DISA, DISB, DISC)

Display subroutines A, B, and C, Figure 4-23, modify the appropriate display memory pointers. The modifications are based on the present cursor location, as determined by subroutine DISPL. The resulting cursor position data is transferred to the 8275 Cursor X and Y Position registers. If DISC is called, a scrolling operation occurs.

Cursor Up Routine (ESCA)

The cursor up routine, Figure 4-24, determines if the cursor is located in the first display row. If it is, the Row Count and Column Count values are modified, and the cursor is moved to the last display row with no change in X position. If the cursor is not in the top row, the row up subroutine is called.

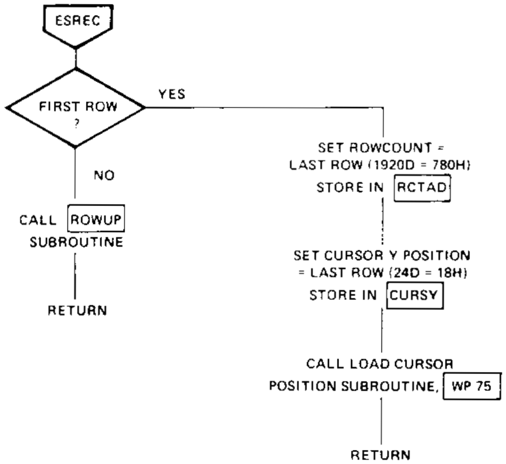


Figure 4-24. Cursor Up Routine (ESCA)

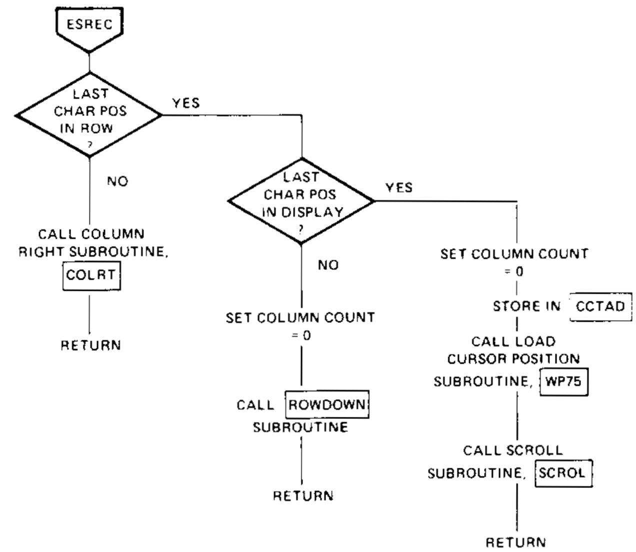


Figure 4-26. Cursor Right Routine (ESCC)

Cursor Down Routine (ESCB)

The cursor down routine, Figure 4-25, determines if the cursor is located in the last display row. If it is, the scroll subroutine is called. No modification of cursor position is called for. If the cursor is not located in the last display row, the row down subroutine is called.

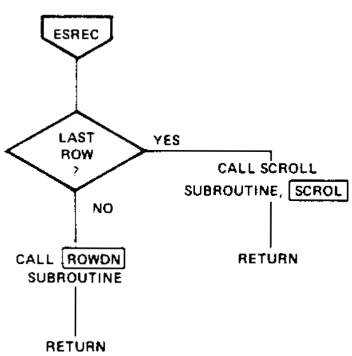


Figure 4-25. Cursor Down Routine (ESCB)

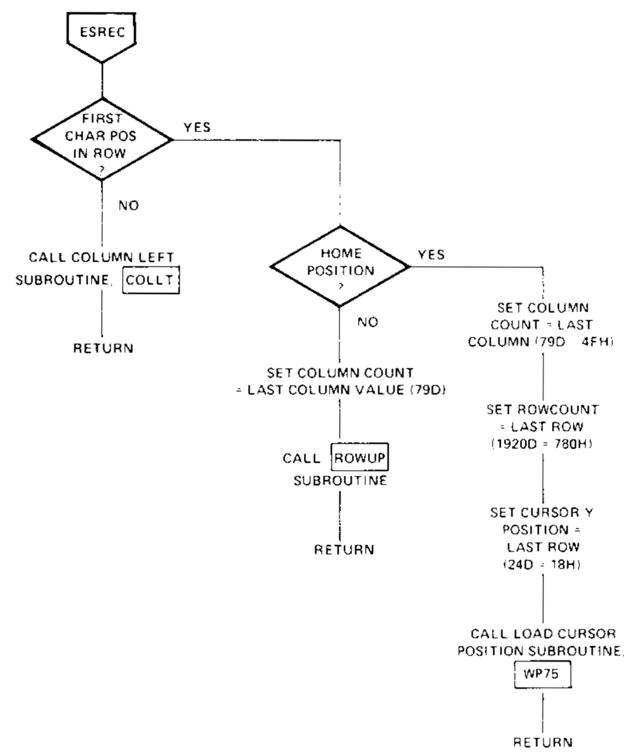


Figure 4-27. Cursor Left Routine (ESCD)

Cursor Right Routine (ESCC)

The cursor right routine tests the cursor location and moves the cursor as described in Figure 4-26. If the cursor is in the last display position, a scrolling operation occurs. 8275 Cursor X and Y Position registers are updated accordingly.

Cursor Left Routine (ESCD)

The cursor left routine tests the cursor location and moves the cursor as described in Figure 4-27.

Clear Screen Routine (ESCE)

Several possibilities existed for implementing the clear screen function. The simplest of these techniques involves filling the display memory with space codes. This technique, although conceptually simple, requires several milliseconds to implement.

The End-of-Row character (EOR) recognized by the 8275 allows the clear screen feature to be executed in a considerably shorter time span. During the clear screen routine, Figure 4-28, EOR characters are placed in the first character position of each row in display memory. Since the EOR character blanks the entire display row when placed in the first character position of the row, the use of EOR characters in each row blanks the entire screen. All pointers are cleared during the clear screen operation.

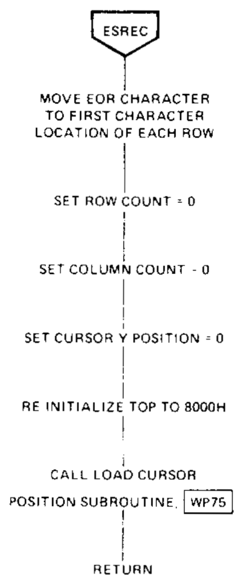


Figure 4-28. Clear Screen Routine (ESCE)

Home Routine (ESCH)

The home routine, Figure 4-29, resets the Row Count, Column Count and Cursor Y Position buffers to zero, but does not affect the value of TOP.

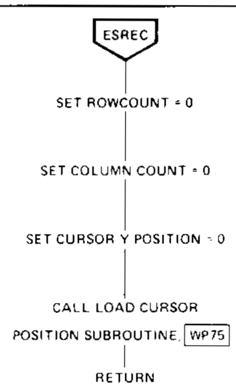


Figure 4-29. Home Routine (ESCH)

Erase to End of Screen Routine (ESCJ)

The erase to end of screen routine, Figure 4-30, inserts End of Row characters (EOR) in display memory in the same fashion as the clear screen routine. The fundamental difference between the routines is that the erase to end of screen routine must insert EOR characters selectively. Only rows from the present display row until the last display row, pointed to by BOTTOM, receive EOR characters. It should be noted that the pointer BOTTOM changes dynamically with scrolling operations.

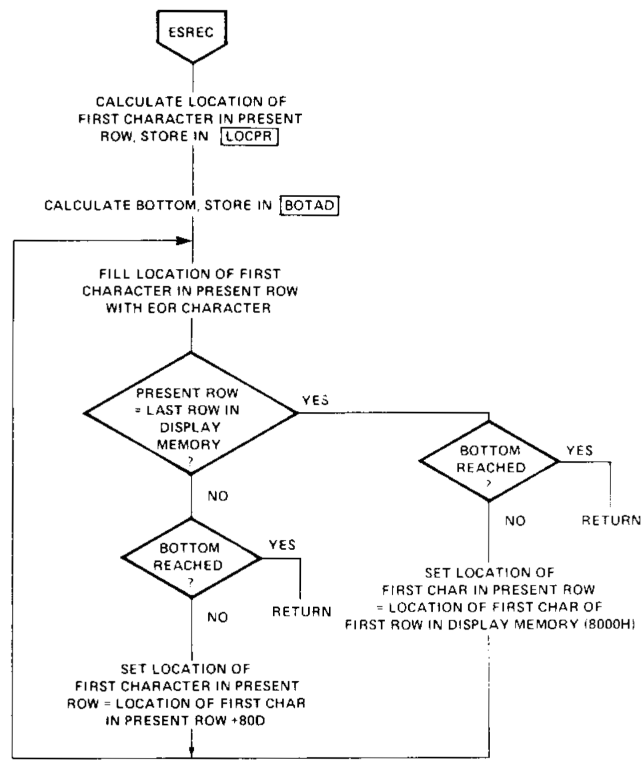


Figure 4-30. Erase to End of Screen Routine (ESCJ)

Erase Line Routine (ESCK)

The erase line routine, Figure 4-31, calculates the location of the first character in the current display row, stores the location in buffer memory, and calls the fill subroutine, which fills the row with space codes.

Backspace Routine (CTRLH)

See cursor left routine.

Line Feed Routine (CTRLJ)

See cursor down routine.

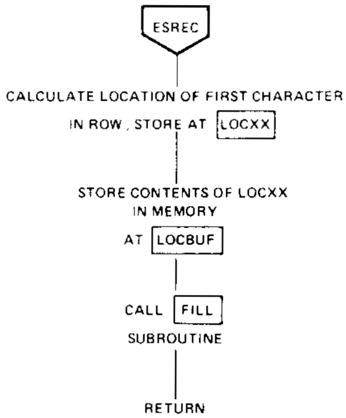


Figure 4-31. Erase Line Routine (ESCK)

Carriage Return Routine (CTRLM)

The carriage return routine, Figure 4-32, clears the column count and updates the 8275 cursor position registers.

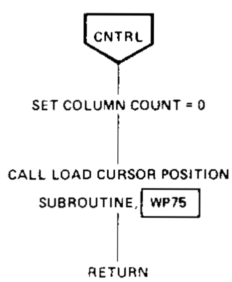


Figure 4-32. Carriage Return Routine (CTRLM)

Row Up, Row Down Subroutines (ROW UP, ROW DOWN)

The row up subroutine, Figure 4-33, subtracts 80D from the Row Count value, decrements the Cursor Y Position pointer, and updates the 8275 Cursor Position registers. The row down subroutine, Figure 4-34, differs in that 80D is added to Row Count.

Column Right, Column Left Subroutines (COLRT, COLLT)

The column right subroutine, Figure 4-35, increments the Column Count pointer and updates the 8275 cursor position registers. The column left subroutine, Figure 4-36, differs in that the Column Count is decremented.

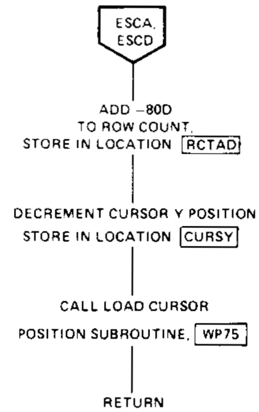


Figure 4-33. Row Up Subroutine (ROWUP)

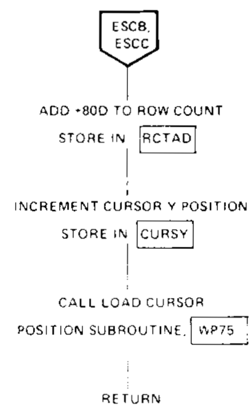


Figure 4-34. Row Down Subroutine (ROWDN)

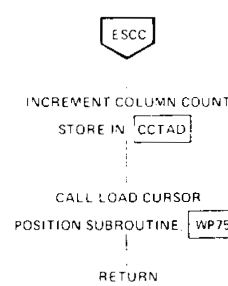


Figure 4-35. Column Right Subroutine (COLRT)

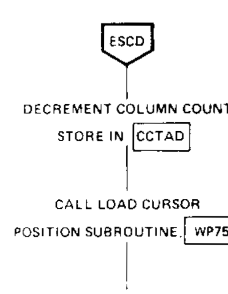


Figure 4-36. Column Left Subroutine (COLLT)

Scroll Subroutine (SCROL)

The scroll subroutine, Figure 4-37, fills the row in display memory pointed to by TOP with space characters via the fill subroutine, then modifies the value of TOP. TOP is utilized by the 8275 service subroutine in re-initializing the 8257 DMA controller.

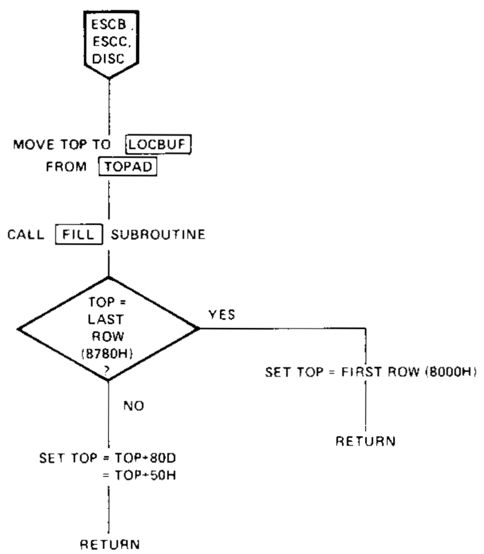


Figure 4-37. Scroll Subroutine (SCROL)

Fill Subroutine (FILL)

The fill subroutine, Figure 4-38, calculates the location of the last character in the current display row, plus one character position, by adding 80D = 50H to the location of the first character in the current display row. The current stack pointer value is saved, then the stack pointer is loaded with the location of the last character in the current display row, plus one character position. The B and C registers of the CPU are loaded with space characters and 40 PUSH B operations performed. This technique provides a rapid means (275 μsec) of filling a given row with space codes.

Load Cursor Position Subroutine (WP 75)

The load cursor position subroutine, Figure 4-39, transfers the contents of the Column Count and cursor Y position pointers to the 8275 cursor X position and cursor Y position registers, respectively.

The relationship between system subroutines is presented in Appendix 5.3. Software timing considerations are covered in Appendix 5.4.

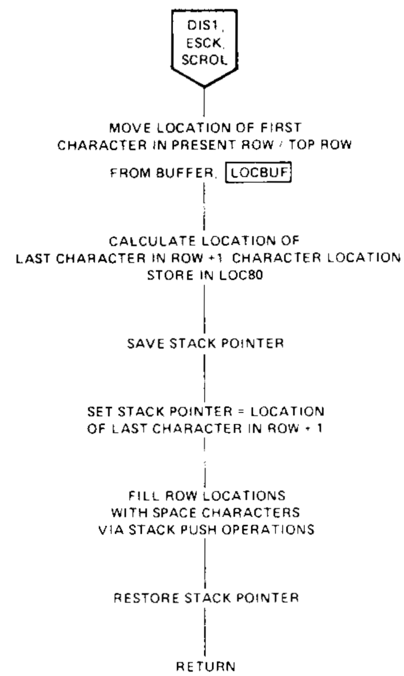


Figure 4-38. Fill Subroutine (FILL)

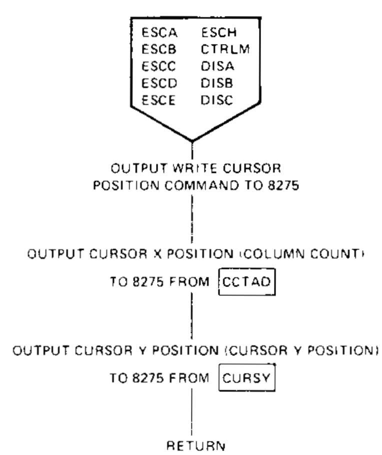
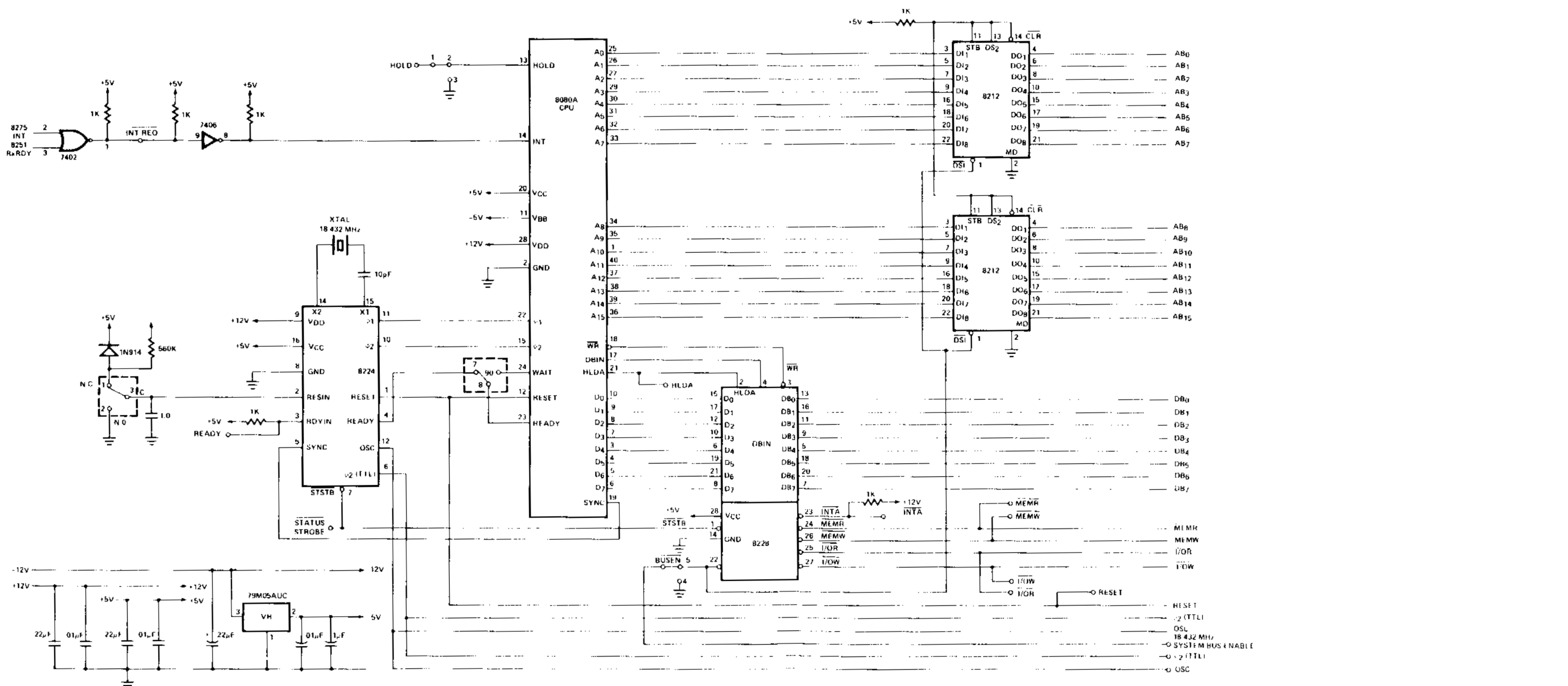
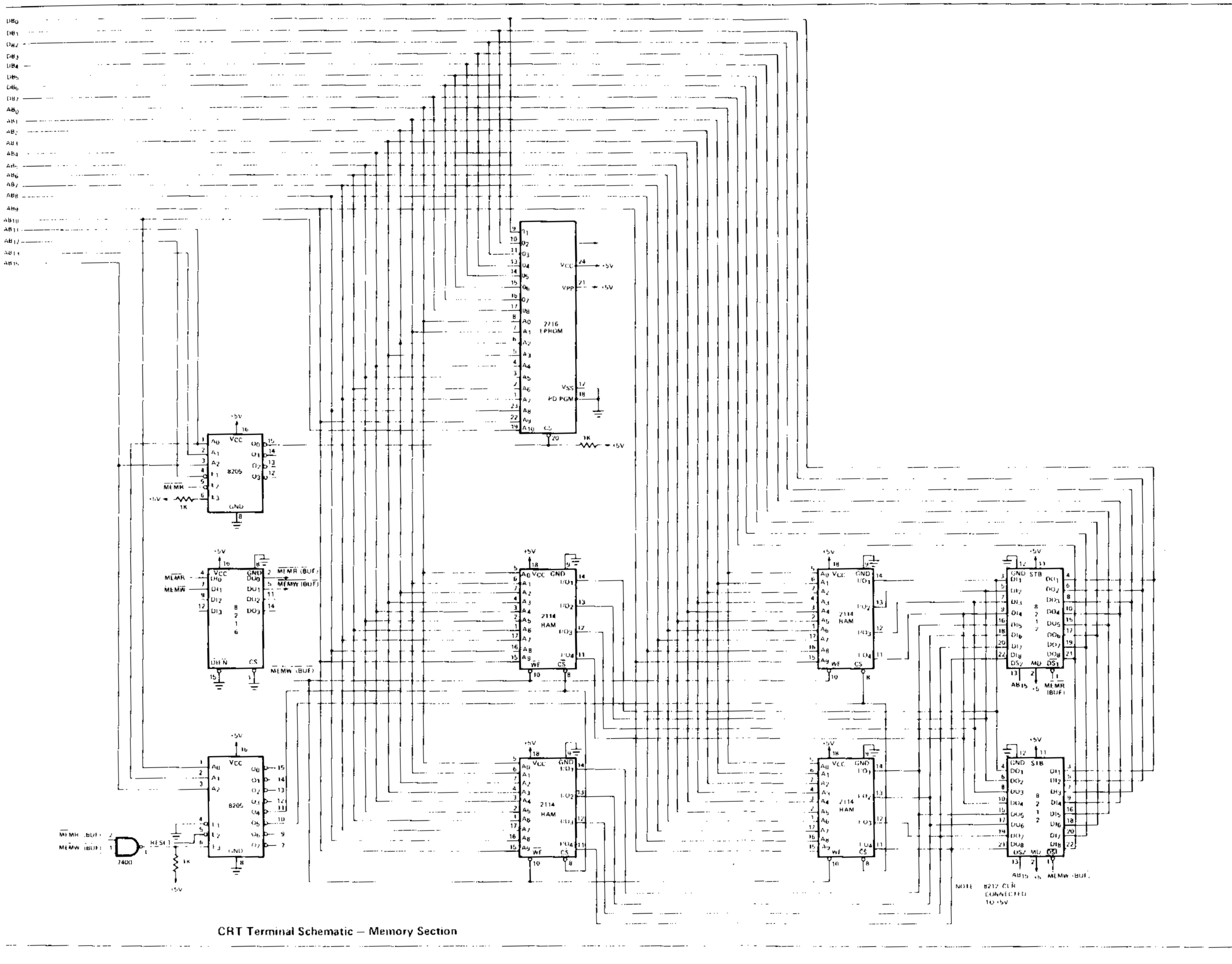


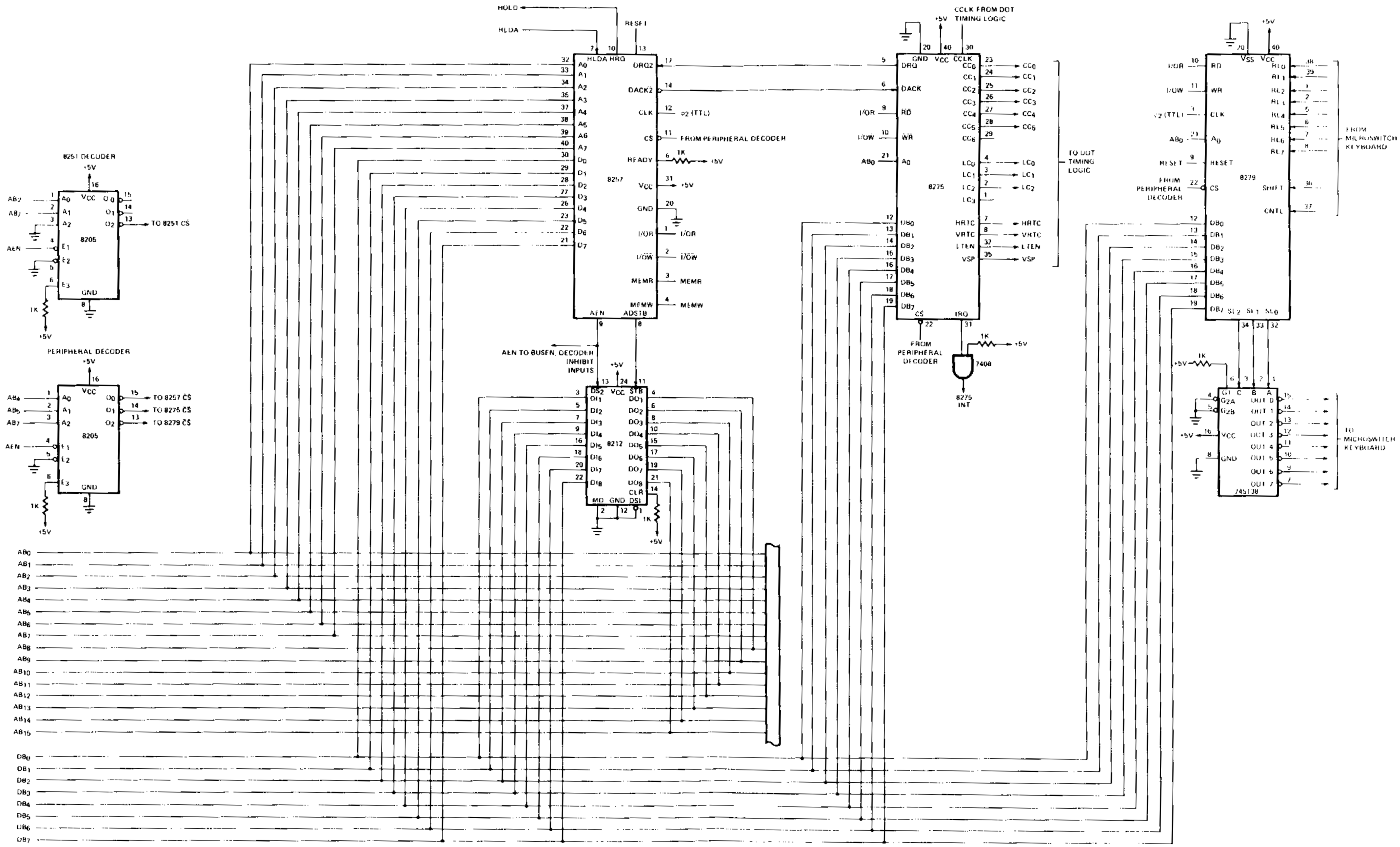
Figure 4-39. Load Cursor Position Subroutine (WP75)



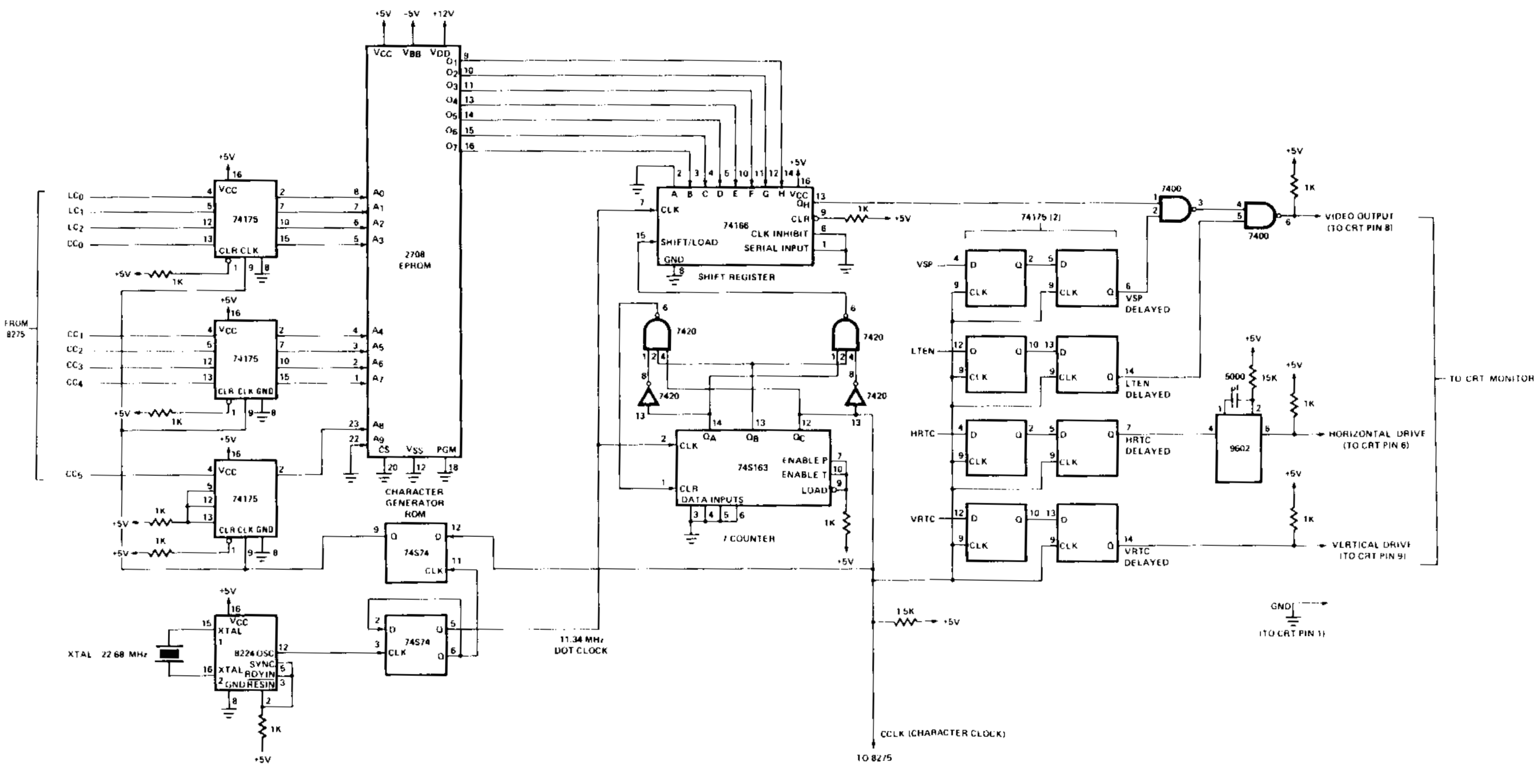
CRT Terminal Schematic – CPU Section



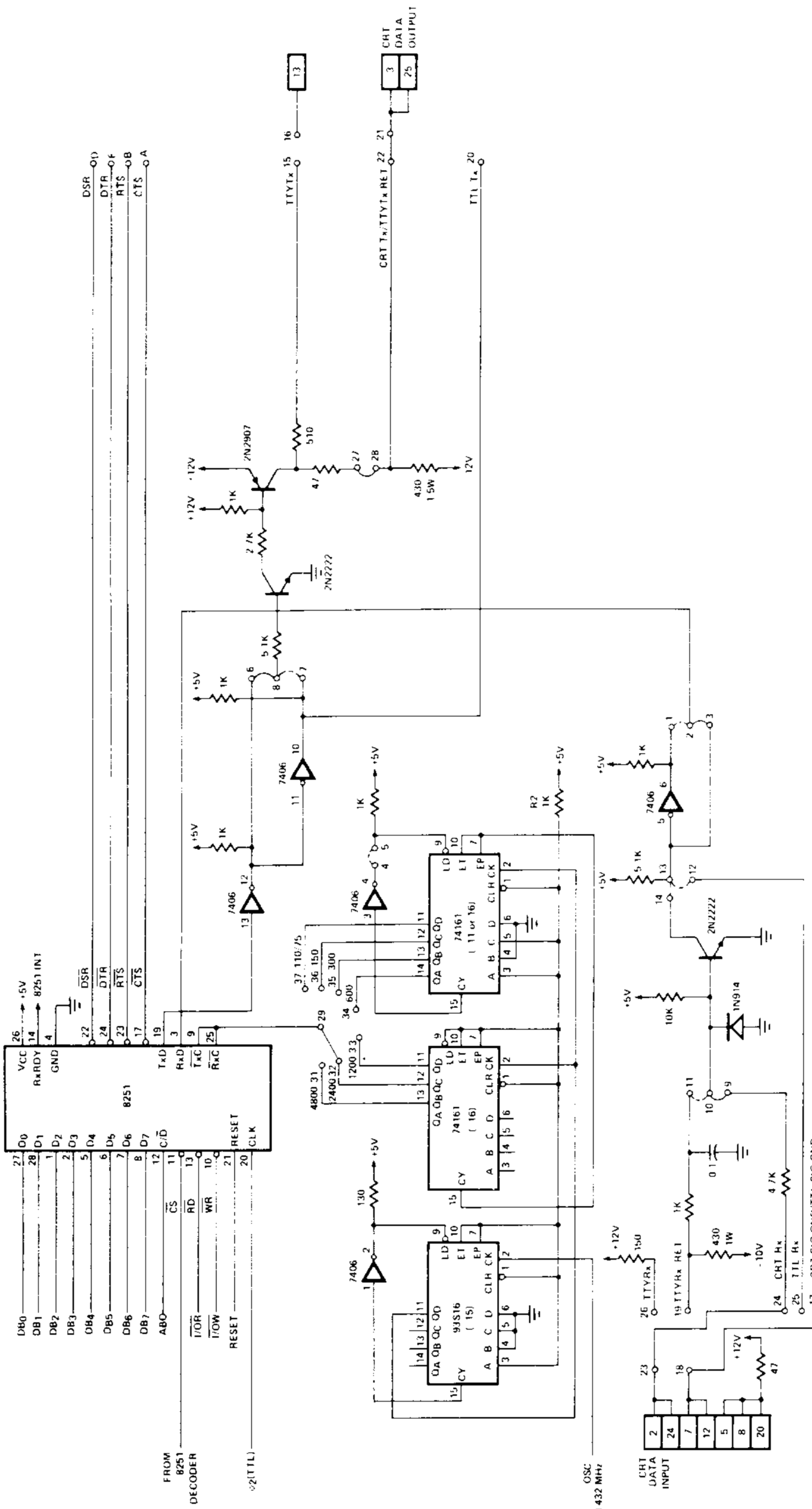
CRT Terminal Schematic – Memory Section



CRT Terminal Schematic – Peripherals Section



CRT Terminal Schematic – Dot Timing Logic Section



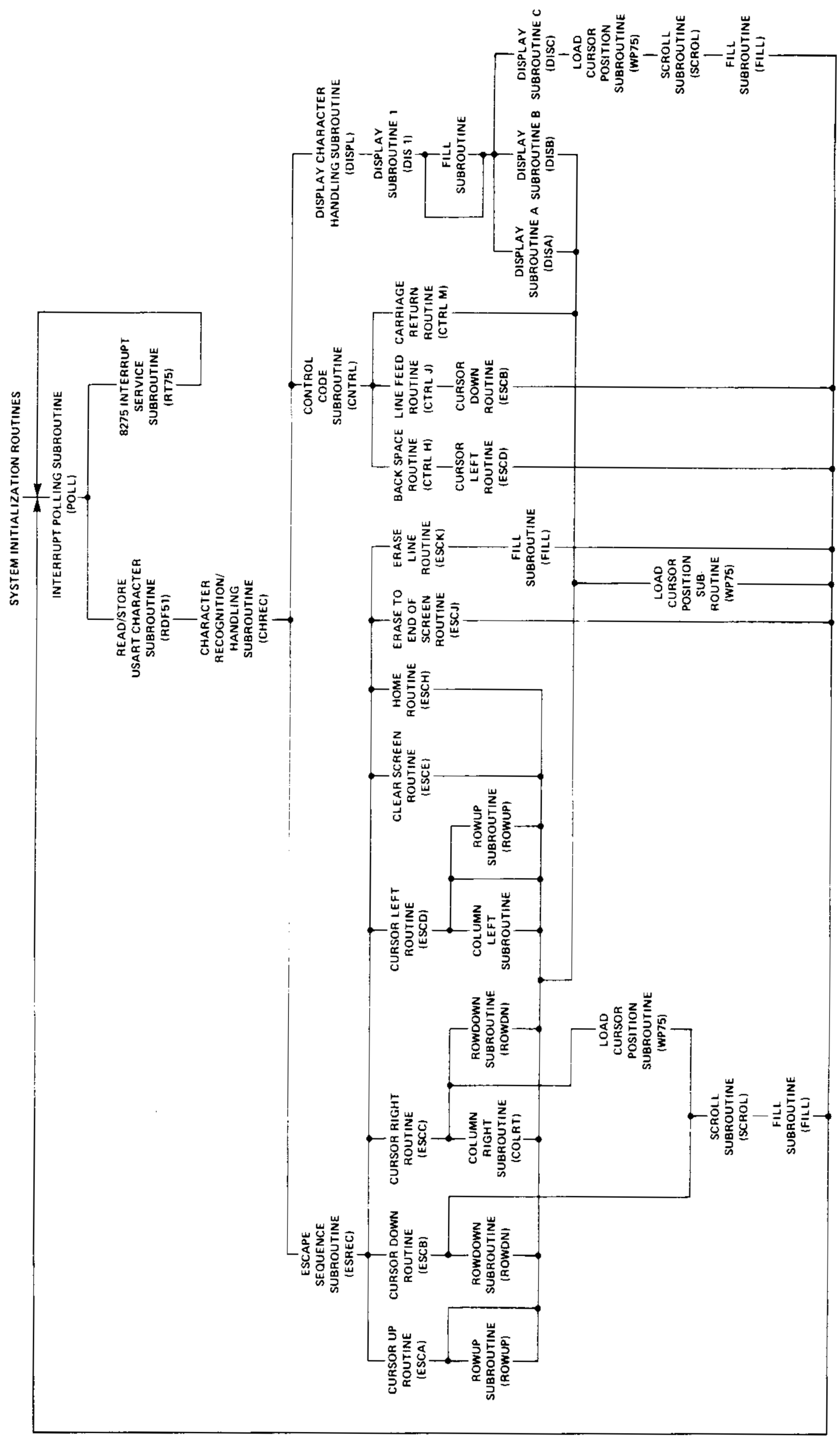
SERIAL COMMUNICATIONS SECTION

Appendix 5.2
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	ø	@	P								
0001	SOH A	DC1 Q	!	1	A	Q					↑ A			
0010	STX B	DC2 R	"	2	B	R					↓ B			
0011	ETX C	DC3 S	#	3	C	S					→ C			
0100	EOT D	DC4 T	$	4	D	T					← D			
0101	ENQ E	NAK U	%	5	E	U					CLR E			
0110	ACK F	SYN V	&	6	F	V								
0111	BEL G	ETB W	'	7	G	W								
1000	BS H	CAN X	(	8	H	X					HOME H			
1001	HT I	EM Y	)	9	I	Y								
1010	LF J	SUB Z	*	:	J	Z					EOS I			
1011	VT K	ESC [	+	;	K	[					EL J			
1100	FF L	FS /	,	<	L	\								
1101	CR M	GS	–	=	M	]								
1110	SO N	RS \	.	>	N	^								
1111	S1 O	US –	/	?	O	–								

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

Appendix 5.3
SUBROUTINE INTERRELATIONSHIPS



Appendix 5.4
SOFTWARE TIMING

Subroutine execution times are summarized in the flowchart provided in Figure 5-1. The values shown represent the number of clock cycles required for the execution of a given routine. The actual routine execution time is obtained by multiplying the number of clock cycles/routine by the time/clock cycle. For a 2.048 MHz system clock, the time/clock cycle is 0.4883 μsec. It should be noted that the values indicated represent worst-case execution times. In order to appreciate the meaning of the subroutine execution times, it is necessary to consider two factors:

1. The time available for the CPU to execute instructions between DMA operations.
2. The maximum rate at which data characters are presented to the CPU for processing.

CPU availability during a complete display frame is illustrated in Figure 5-2. Available CPU processing time, per character, at 4800 baud, during the DMA active portion of the display frame, is illustrated in Figure 5-3. It can be seen from Figure 5-3 that 1443 μsec are available for processing each character during the DMA active portion of the frame. Total CPU processing time during the DMA inactive portion of the frame may be seen from Figure 5-2 to be 1234 μsec. This value encompasses the time to process the 8275 interrupt and perform character handling functions.

Using the information contained in Figure 5-1, the maximum execution time* for a given character handling routine is 802 μsec. Since this value is less than 1.443 msec, proper timing is assured. Using the maximum character handling routine execution time and the time required for 8275 interrupt processing, the maximum CPU availability requirement during the DMA inactive portion of the frame may be calculated. This value corresponds to 802 μsec + 253 μsec (8275 interrupt processing) or 1055 μsec. Since this value is less than 1234 μsec, proper timing is assured.

*see notes, Figure 5-1.

Appendix 5.5
VISUAL ATTRIBUTE IMPLEMENTATION CONSIDERATIONS

In order to utilize the visual attribute features of the 8275, it is necessary to modify the CRT system hardware and software functions accordingly.

Hardware modifications necessary to implement character attributes are illustrated in Figure 5-4. The attribute outputs LA0–LA1 selectively control the data transferred to the output shift register.

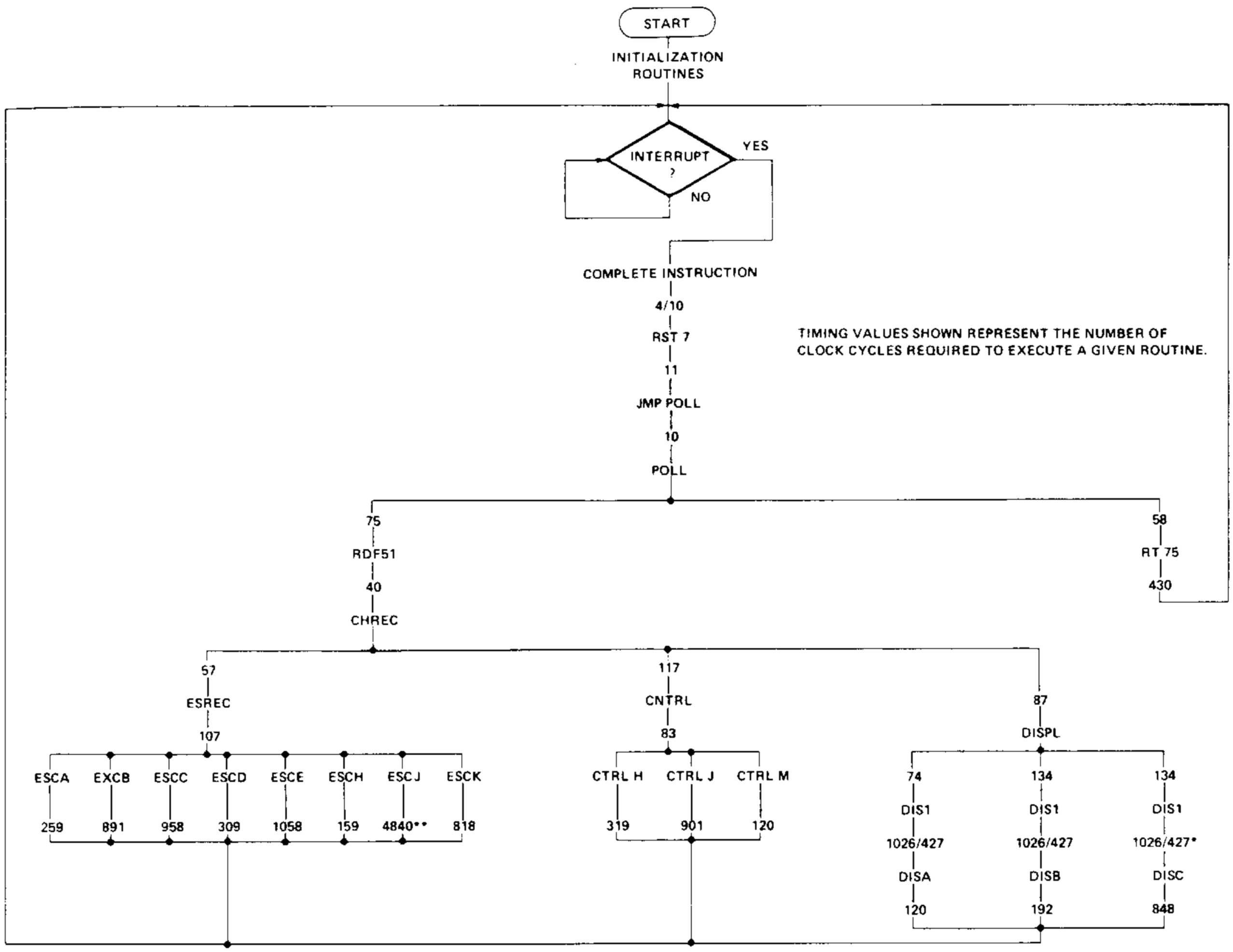
The software memory management scheme presented in the Application Note must be modified in order to accommodate attribute features. An outline of the software considerations involved when using the attribute features is presented as follows:

1. Attributes, as described in the 8275 Data Sheet, occupy character locations in display memory. Since the number of attributes per display row may be variable, the linear mapping relationship between character position on the screen and memory pointers Top, Row Count, and Column Count no longer exists. It is necessary to keep track of the number of attribute characters in each row and their specific location when modifying pointer values.

2. The increased number of character locations required will force the user to incorporate additional display RAM.

3. Since the total number of characters in display memory may be variable when attributes are utilized, it is necessary to modify the starting address and terminal count values for the DMA channels as required.

4. Character insertion and deletion operations may be handled through block transfer operations or through the use of extended display memory row segments.



* UNDER NORMAL OPERATING CONDITIONS, 427 CLOCK CYCLES REPRESENTS THE WORST CASE EXECUTION TIME FOR THIS ROUTINE.

** IT IS NECESSARY FOR THE REMOTE DEVICE TO WAIT APPROXIMATELY 2.5 ms FOLLOWING THE TRANSMISSION OF AN ESCJ CHARACTER BEFORE RESUMING TRANSMISSION.

Figure 5-1. Subroutine Execution Times Flowchart

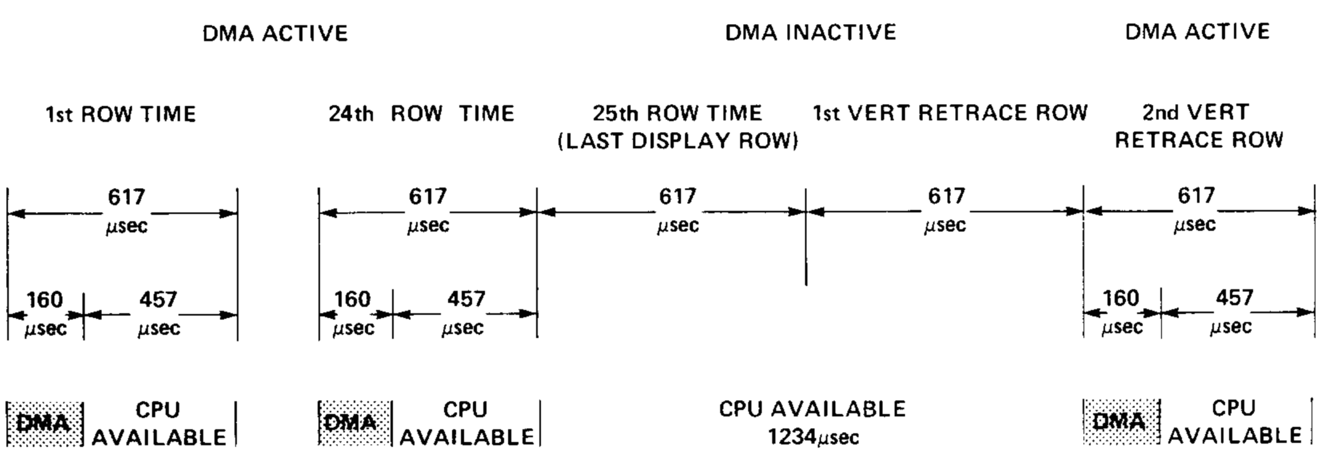


Figure 5-2. CPU Availability

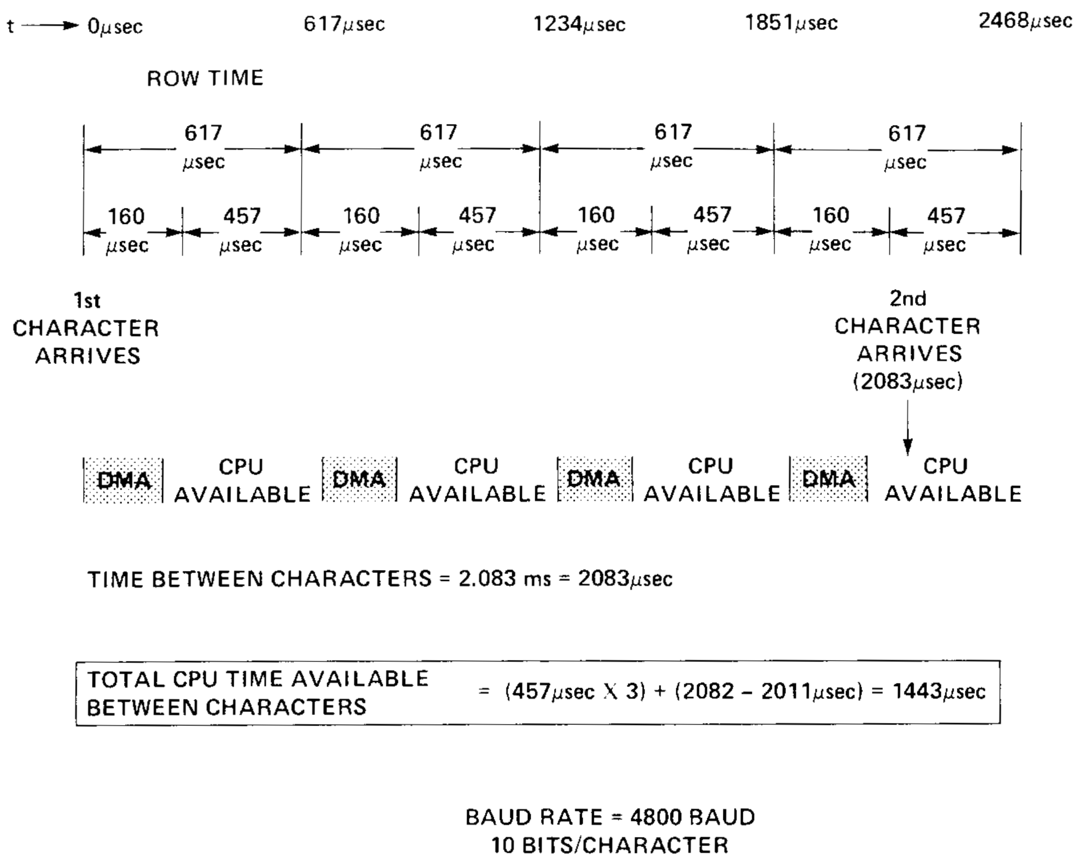


Figure 5-3. CPU Availability/Character at 4800 Baud (DMA Active)

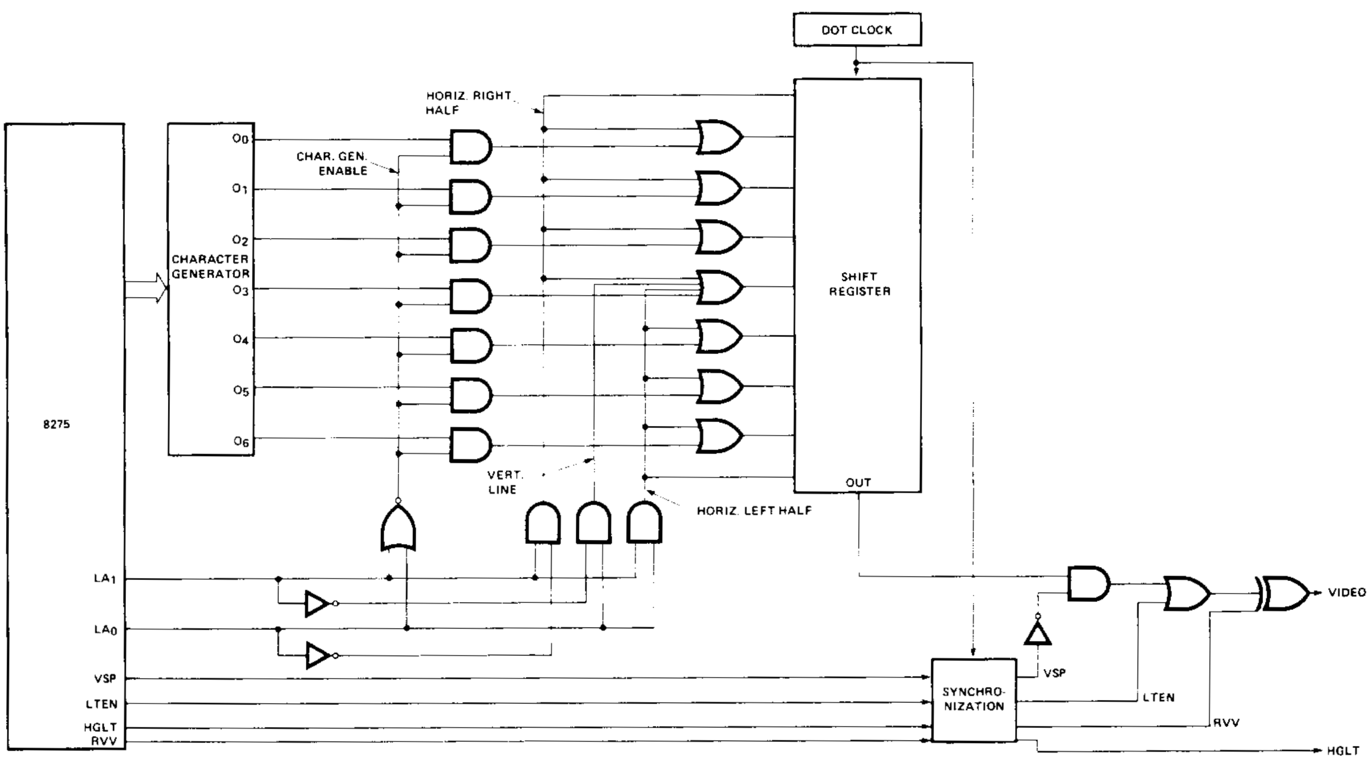


Figure 5-4. Typical Character Attribute Logic

SOFTWARE LISTINGS

```
LOC  OBJ          SOURCE STATEMENT

             ;8275/8279 CRT SYSTEM SOFTWARE
                     ;
                     ;SYSTEM EQUATES
                     ;
00FB         CNCTL   EQU     0FBH           ;8251 CONTROL ADDRESS
00FA         CNIN    EQU     0FAH           ;8251 INPUT DATA ADD
00FA         CNOUT   EQU     0FAH           ;8251 OUTPUT DATA ADD
006F         KCOM    EQU     6FH            ;8279 COMMAND ADDRESS
006E         KDAT    EQU     6EH            ;8279 DATA ADDRESS
005F         CRCOM   EQU     5FH            ;8275 COMMAND ADDRESS
005E         CRDAT   EQU     5EH            ;8275 DATA ADDRESS
0044         PC2SA   EQU     44H            ;8257 CH 2 START ADD PORT
0045         PC2TC   EQU     45H            ;8257 CH 2 TERM COUNT PORT
0046         PC3SA   EQU     46H            ;8257 CH 3 STARTING ADD PORT
0047         PC3TC   EQU     47H            ;8257 CH 3 TERM COUNT PORT
0000         MDC57   EQU     00H            ;8257 MODE CLEAR
0084         MDS57   EQU     84H            ;8257 MODE SET (AUTOLOAD, CH 2 ENABLED)
0048         PMD57   EQU     48H            ;8257 MODE SET PORT
                     ;
                     ;SYSTEM INITIALIZATION ROUTINES
                     ;
                     ;
0000 C34000          JMP     CRTGO          ;JUMP TO START OF MAIN ROUTINE
                     ;
0038                 ORG     0038H
                     ;
0038 C3C900          JMP     POLL           ;JUMP TO START OF INT SERVICE ROUTINE
                     ;
0040                 ORG     0040H
                     ;
0040 F3      CRTGO:  DI                     ;DISABLE INTERRUPTS
0041 31FF87          LXI     SP, 87FFH      ;LOAD STACK POINTER
                     ;
                     ;MEMORY CLEAR ROUTINE
                     ;
0044 210080          LXI     H,8000H        ;LOAD H&L WITH START ADD OF DISPLAY MEM
0047 3E20    THETA:  MVI     A, 20H         ;LOAD A WITH SPACE CHAR CODE
0049 77              MOV     M,A            ;LOAD SPACE CHAR IN MEM
004A 7D              MOV     A,L            ;MOVE LOW ADD BYTE TO A
004B FECF            CPI     0CFH           ;COMPARE WITH 0CFH
004D CA5400          JZ      NXT1           ;IF COMPARRISON JMP TO NXT1
0050 23              INX     H              ;INCREMENT H&L
0051 C34700          JMP     THETA          ;JMP TO THETA, CONT LOADING MEMORY
0054 7C      NXT1:   MOV     A,H            ;MOVE UP ADD BYTE TO A
0055 FE87            CPI     87H            ;COMPARE WITH 87H
0057 CA5E00          JZ      NXT2           ;IF COMPARRISON ADD=LAST DISPLAY
                                            ;ADD, THEREFORE, JMP TO NXT2
005A 23              INX     H              ;INCREMENT H&L
005B C34700          JMP     THETA          ;JMP TO THETA, CONT LOADING MEMORY
                     ;
                     ;POINTER/BUFFER CLEAR ROUTINE
                     ;
005E 210000  NXT2:   LXI     H,0000H        ;ZERO H&L
0061 22D387          SHLD    RCTAD          ;ZERO ROW COUNT
0064 22E287          SHLD    LOCBUF         ;ZERO BUFFER
0067 22D887          SHLD    LOCAD          ;ZERO CHARACTER LOCATION
006A 22DA87          SHLD    LOC01          ;ZERO LOC OF 1ST CHAR IN ROW
006D 22DC87          SHLD    LOC80          ;ZERO LOC OF 80TH CHAR IN ROW
0070 22DE87          SHLD    LOCXX          ;ZERO PRESENT LOC OF 1ST CHAR IN ROW
0073 22E087          SHLD    LOCPR          ;ZERO PRESENT LOC OF 1ST CHAR IN ROW
0076 210080          LXI     H,8000H        ;LOAD H&L WITH 8000H
0079 22D687          SHLD    TOPAD          ;SET TOP = 8000H
007C 218087          LXI     H,8780H        ;LOAD H&L WITH 8780H
007F 22E687          SHLD    BOTAD          ;SET BOT = 8780H
0082 3E00            MVI     A,00H          ;ZERO A
0084 32D287          STA     CCTAD          ;ZERO COLUMN COUNT
0087 32D587          STA     CURSY          ;ZERO CURSOR Y POINTER
008A 32E487          STA     XFLG           ;ZERO ESC SEQ FLAG
008D 32E587          STA     USCHR          ;ZERO USART CHAR BUFFER
                     ;
                     ;8251 INITIALIZATION ROUTINE
                     ;
0090 3E4F            MVI     A,4FH          ;MODE SET VALUE TO A
0092 D3FB            OUT     CNCTL          ;OUPUT VALUE
0094 3E27            MVI     A,27H          ;COMMAND WORD TO A
0096 D3FB            OUT     CNCTL          ;OUTPUT VALUE
                     ;
                     ;8279 INITIALIZATION ROUTINE
                     ;
0098 3E35            MVI     A,35H          ;OUTPUT PROG CLOCK, DIV BY 21
009A D36F            OUT     KCOM
                     ;
                     ;8275 INITIALIZATION ROUTINE
                     ;
```

```
009C 3E00          MVI    A,00H      ;RESET AND STOP DISPLAY
009E D35F          OUT    CRCOM
00A0 3E4F          MVI    A,4FH      ;SCREEN PARAM BYTE 1
00A2 D35E          OUT    CRDAT
00A4 3E58          MVI    A,58H      ;              BYTE 2
00A6 D35E          OUT    CRDAT
00A8 3E89          MVI    A,89H      ;              BYTE 3
00AA D35E          OUT    CRDAT
00AC 3ED9          MVI    A,0D9H     ;              BYTE 4
00AE D35E          OUT    CRDAT
00B0 3E80          MVI    A,80H      ;LOAD CURSOR POSITION
00B2 D35F          OUT    CRCOM
00B4 3E00          MVI    A,00H      ;CURSOR X POSITION
00B6 D35E          OUT    CRDAT
00B8 3E00          MVI    A,00H      ;CURSOR Y POSITION
00BA D35E          OUT    CRDAT
00BC 3EE0          MVI    A,0E0H     ;PRESET COUNTERS
00BE D35F          OUT    CRCOM
00C0 3E23          MVI    A,23H      ;START DISPLAY
00C2 D35F          OUT    CRCOM
00C4 FB            EI                ;ENABLE INTERRUPTS
00C5 00      LOOP: NOP
00C6 C3C500        JMP    LOOP
                   ;
                   ;
                   ;8275/8251 INTERRUPT POLLING ROUTINE
                   ;
                   ;
00C9 DB5F    POLL: IN     CRCOM      ;READ 8275 STATUS, CLEARING INT
00CB E620          ANI    20H        ;MASK STATUS, SAVE INT REQ BIT
00CD CAD500        JZ     AGGIE      ;IF STATUS=1, SERVICE 8275
                   ;
00D0 CD7304 GIGEM: CALL   RT75       ;CALL 8275 INT SERVICE SUBROUTINE
00D3 FB            EI                ;ENABLE INTERRUPTS
00D4 C9            RET               ;RETURN
                   ;
00D5 CDDD00 AGGIE: CALL   RDF51      ;CALL READ USART CHAR ROUTINE
00D8 CDE500        CALL   CHREC      ;CALL CHARACTER RECOG/HANDLING ROUTINE
00DB FB            EI                ;ENABLE INTERRUPTS
00DC C9            RET               ;RETURN
                   ;
                   ;USART READ/STORE CHAR SUBROUTINE
                   ;
00DD DBFA   RDF51: IN     CNIN       ;READ ASCII CHAR FROM USART, RESETTING RXRDY
00DF E67F          ANI    7FH        ;MASK BIT 8,SAVE BITS 1-7
00E1 32E587        STA    USCHR      ;STORE USART CHAR IN MEMORY
00E4 C9            RET               ;RETURN
                   ;
                   ;CHARACTER RECOGNITION/HANDLING SUBROUTINE
                   ;
00E5 3AE487 CHREC: LDA    XFLG       ;LOAD A WITH ESC SEQ FLAG
00E8 E6FF          ANI    0FFH       ;SET/RESET ZERO BIT
00EA CAF100        JZ     NXTX       ;IF ONE,CHAR=2ND CHAR IN ESC SEQ
00ED CD0F01        CALL   ESREC      ;CALL ESC SEQ SUBROUTINE
00F0 C9            RET               ;RETURN
00F1 3AE587  NXTX: LDA    USCHR      ;LOAD USART CHAR IN A
00F4 E660          ANI    60H        ;MASK BITS 1-5,&8,SAVING BITS 6&7
00F6 CAFD00        JZ     NXTY       ;IF ZERO CHAR=CONTROL CHAR
                                     ;IF ONE CHAR=DISPLAY CHAR
00F9 CD4B03        CALL   DISPL      ;CALL DISPLAY CHAR SUBROUTINE
00FC C9            RET               ;RETURN
00FD 3AE587  NXTY: LDA    USCHR      ;LOAD  USART CHAR IN A
0100 E610          ANI    10H        ;MASK OFF BITS,SAVE BIT 5
0102 C20901        JNZ    NXTZ       ;IF ZERO CONT CHAR=CONT CODE
                                     ;IF ONE CONT CHAR=ESC CODE
0105 CD2701        CALL   CNTRL      ;CALL CONTROL CODE SUBROUTINE
0108 C9            RET               ;RETURN
0109 21E487  NXTZ: LXI    H,XFLG     ;LOAD H&L WITH ADD OF ESC SEQ FLAG
010C 3601          MVI    M,01H      ;SET ESC SEQ FLAG
010E C9            RET               ;RETURN
                   ;
                   ;ESCAPE SEQUENCE SUBROUTINE
                   ;
010F 3E00   ESREC: MVI    A,00H      ;ZERO A
0111 32E487        STA    XFLG       ;RESET ESC SEQ FLAG
0114 3AE587        LDA    USCHR      ;LOAD USART CHAR IN A
0117 E60F          ANI    0FH        ;MASK BITS 5-8
0119 07            RLC               ;SHIFT LEFT,YIELDING OFFSET
011A 21D004        LXI    H,BSET1    ;LOAD BASE ADD OF TABLE 1 IN H&L
011D 110000        LXI    D,0000H    ;ZERO D&E
0120 5F            MOV    E,A        ;LOAD OFFSET IN E
0121 19            DAD    D          ;ADD OFFSET TO BASE, RESULT IN H&L
0122 5E            MOV    E,M        ;MOVE LOW BYTE OF ROUTINE ADD TO E
0123 23            INX    H          ;INCREMENT COMPUTED ADDRESS
0124 56            MOV    D,M        ;MOVE UP BYTE OF ROUTINE ADD TO D
0125 EB            XCHG              ;EXCHANGE D&E WTIH H&L
0126 E9            PCHL              ;LOAD PC WITH ROUTINE ADD, JMP TO ROUTINE
                   ;
                   ;CONTROL CODE SUBROUTINE
```

```
                        ;
0127 3AE587   CNTRL:    LDA     USCHR       ;LOAD USART CHAR IN A
012A E606               ANI     06H         ;MASK CHAR, SAVE BITS 2-3
012C 21F004             LXI     H,BSET2     ;LOAD BASE ADD OF TABLE 2 IN H&L
012F 110000             LXI     D,0000H     ;CLEAR D&E
0132 5F                 MOV     E,A         ;LOAD OFFSET IN E
0133 19                 DAD     D           ;ADD OFFSET TO BASE, RESULT IN H&L
0134 5E                 MOV     E,M         ;MOVE LOW BYTE OF ROUTINE ADD TO E
0135 23                 INX     H           ;INCREMENT COMPUTED ADDRESS
0136 56                 MOV     D,M         ;MOVE UP BYTE OF ROUTINE ADD TO D
0137 EB                 XCHG                ;EXCHANGE D&E WITH H&L
0138 E9                 PCHL                ;LOAD PC WITH ROUTINE ADD, JMP TO ROUTINE
                        ;
                        ;CURSOR UP ROUTINE
                        ;
0139 2AD387   ESCA:     LHLD    RCTAD       ;LOAD ROWCOUNT IN H&L
013C 7D                 MOV     A,L         ;MOVE LOW BYTE OF ROWCOUNT TO A
013D FE00               CPI     00H         ;COMPARE BYTE WITH 00H
013F CA4601             JZ      ALPHA       ;IF BYTE=0 CONTINUE COMPARRISON
0142 CD0803             CALL    ROWUP       ;CALL ROWUP SUBROUTINE
0145 C9                 RET                 ;RETURN
0146 7C       ALPHA:    MOV     A,H         ;MOVE UP BYTE OF ROWCOUNT TO A
0147 FE00               CPI     00H         ;COMPARE BYTE WITH 00H
0149 CA5001             JZ      BETA        ;IF BYTE=0,ROWCOUNT=FIRST ROW
014C CD0803             CALL    ROWUP       ;CALL ROWUP SUBROUTINE
014F C9                 RET                 ;RETURN
0150 218007   BETA:     LXI     H,0780H     ;LOAD H&L WITH ROWCOUNT=LAST ROW VALUE (1920D)
0153 22D387             SHLD    RCTAD       ;STORE 0780H IN ROWCOUNT BUFFER
0156 3E18               MVI     A,18H       ;LOAD A WITH CURSOR Y POS=LAST ROW VALUE (240)
0158 32D587             STA     CURSY       ;STORE 18H IN CURSOR Y POS BUFFER
015B CD3C03             CALL    WP75        ;CALL LOAD CURSOR POSITION SUBROUTINE
015E C9                 RET
                        ;
                        ;CURSOR DOWN ROUTINE
                        ;
015F 2AD387   ESCB:     LHLD    RCTAD       ;LOAD ROWCOUNT IN H&L
0162 7D                 MOV     A,L         ;MOVE LOW BYTE OF ROWCOUNT TO A
0163 FE80               CPI     80H         ;COMPARE BYTE WITH 80H
0165 CA6C01             JZ      GAMMA       ;IF BYTE=80H, CONTINUE COMPARRISON
0168 CD1A03             CALL    ROWDN       ;CALL ROWDOWN SUBROUTINE
016B C9                 RET                 ;RETURN
016C 7C       GAMMA:    MOV     A,H         ;MOVE UP BYTE OF ROWCOUNT TO A
016D FE07               CPI     07H         ;COMPARE BYTE WITH 07H
016F CA7601             JZ      DELTA       ;IF BYTE=07H, ROWCOUNT=LAST ROW
0172 CD1A03             CALL    ROWDN       ;CALL ROWDOWN SUBROUTINE
0175 C9                 RET                 ;RETURN
0176 CD3C03   DELTA:    CALL    WP75        ;CALL LOAD CURSOR POSITION SUBROUTINE
0179 CD0B04             CALL    SCROL       ;CALL SCROLL SUBROUTINE
017C C9                 RET                 ;RETURN
                        ;
                        ;CURSOR RIGHT ROUTINE
                        ;
017D 3AD287   ESCC:     LDA     CCTAD       ;LOAD COLUMN COUNT IN A
0180 FE4F               CPI     4FH         ;COMPARE BYTE WITH 4FH
0182 CA8901             JZ      ZETA        ;IF BYTE=4FH, COLUMN COUNT =LAST
                                            ;CHARACTER POS IN ROW
0185 CD3403             CALL    COLRT       ;CALL COLUMN RIGHT SUBROUTINE
0188 C9                 RET                 ;RETURN
0189 2AD387   ZETA:     LHLD    RCTAD       ;LOAD ROWCOUNT IN H&L
018C 7D                 MOV     A,L         ;MOVE LOW BYTE OF ROWCOUNT TO A
018D FE80               CPI     80H         ;COMPARE BYTE WITH 80H
018F C29B01             JNZ     CCTOA       ;IF BYTE=80H, CONTINUE COMPARRISON
0192 7C                 MOV     A,H         ;MOVE UP BYTE OF ROWCOUNT TO A
0193 FE07               CPI     07H         ;COMPARE BYTE WITH 07H
0195 C29B01             JNZ     CCTOA       ;IF BYTE=07H,ROWCOUNT=LAST ROW
0198 C3A401             JMP     CCTOB       ;JUMP TO CCTOB
019B 3E00     CCTOA:    MVI     A,00H       ;ZERO A
019D 32D287             STA     CCTAD       ;ZERO COLUMN COUNT
01A0 CD1A03             CALL    ROWDN       ;CALL ROWDOWN SUBROUTINE
01A3 C9                 RET                 ;RETURN
01A4 3E00     CCTOB:    MVI     A,00H       ;ZERO A
01A6 32D287             STA     CCTAD       ;ZERO COLUMN COUNT BUFFER
01A9 CD3C03             CALL    WP75        ;CALL LOAD CURSOR POSITION SUBROUTINE
01AC CD0B04             CALL    SCROL       ;CALL SCROLL SUBROUTINE
01AF C9                 RET                 ;RETURN
                        ;
                        ;CURSOR LEFT ROUTINE
                        ;
01B0 3AD287   ESCD:     LDA     CCTAD       ;LOAD COLUMN COUNT IN A
01B3 FE00               CPI     00H         ;COMPARE BYTE WITH 00H
01B5 CABC01             JZ      NXTA        ;IF BYTE=0,COLUMN COUNT =FIRST CHAR POS IN ROW
01B8 CD2C03             CALL    COLLT       ;CALL COLUMN LEFT SUBROUTINE
01BB C9                 RET                 ;RETURN
01BC 2AD387   NXTA:     LHLD    RCTAD       ;LOAD ROWCOUNT IN H&L
01BF 7D                 MOV     A,L         ;LOAD LOW BYTE OF ROWCOUNT IN A
01C0 FE00               CPI     00H         ;COMPARE BYTE WITH 00H
01C2 C2CE01             JNZ     CCTMA       ;IF BYTE=0,CONTINUE COMPARRISON
01C5 7C                 MOV     A,H         ;LOAD UP BYTE OF ROWCOUNT IN A
01C6 FE00               CPI     00H         ;COMPARE BYTE WITH ZERO
01C8 C2CE01             JNZ     CCTMA       ;IF BYTE=0,HOME POS CONDITION EXISTS
```

```
01CB C3D701          JMP    CCTMB          ;JUMP TO CCTMB
01CE 3E4F    CCTMA:  MVI    A,4FH          ;LOAD A WITH 4FH
01D0 32D287          STA    CCTAD          ;SET COLUMN COUNT=4FH=79D
01D3 CD0803          CALL   ROWUP          ;CALL ROWUP SUBROUTINE
01D6 C9              RET                   ;RETURN
01D7 218007  CCTMB:  LXI    H,0780H        ;LOAD H&L WITH ROWCOUNT=780H=1920D
01DA 22D387          SHLD   RCTAD          ;SET ROWCOUNT =1920D
01DD 3E4F            MVI    A,4FH          ;LOAD A WITH 4FH
01DF 32D287          STA    CCTAD          ;SET COLUMN COUNT=4FH=79D
01E2 3E18            MVI    A,18H          ;LOAD A WITH 18H
01E4 32D587          STA    CURSY          ;SET CURSOR Y POINTER=18H=24D
01E7 CD3C03          CALL   WP75           ;CALL LOAD CURSOR POSITION SUBROUTINE
01EA C9              RET                   ;RETURN
                     ;
                     ;HOME ROUTINE
                     ;
01EB 210000  ESCH:   LXI    H,0000H        ;ZERO H&L
01EE 22D387          SHLD   RCTAD          ;SET ROWCOUNT=0
01F1 3E00            MVI    A,00H          ;ZERO A
01F3 32D287          STA    CCTAD          ;SET COLUMN COUNT=0
01F6 32D587          STA    CURSY          ;SET CURSOR Y POINTER=0
01F9 CD3C03          CALL   WP75           ;CALL LOAD CURSOR POSITION SUBROUTINE
01FC C9              RET                   ;RETURN
                     ;
                     ;ERASE LINE ROUTINE
                     ;
01FD 2AD687  ESCK:   LHLD   TOPAD          ;LOAD TOP IN H&L
0200 EB              XCHG                  ;STORE TOP IN D&E
0201 2AD387          LHLD   RCTAD          ;LOAD ROWCOUNT IN H&L
0204 19              DAD    D              ;ADD TOP+ROWCOUNT, RESULT IN H&L
0205 22DE87          SHLD   LOCXX          ;STORE RESULT IN MEM
                     ;
0208 3E87            MVI    A,87H          ;LOAD 87H IN A
020A BC              CMP    H              ;COMPARE H WITH 87H
020B D21402          JNC    FRODO          ;IF NO CARRY, CONTINUE
020E CD2A02          CALL   COMRX          ;IF CARRY,CALL COMPENSATION ROUTINE
0211 C32002          JMP    BILBO          ;JUMP TO BILBO
0214 C22002  FRODO:  JNZ    BILBO          ;IF NOT EQUAL END COMPARRISON
0217 3ECF            MVI    A,0CFH         ;LOAD CFH IN A
0219 BD              CMP    L              ;COMPARE L WITH CFH
021A D22002          JNC    BILBO          ;IF NO CARRY,LOCXX LESS THAN OR EQ TO 87CFF
021D CD2A02          CALL   COMRX          ;IF CARRY, CALL COMPENSATION ROUTINE
0220 2ADE87  BILBO:  LHLD   LOCXX          ;LOAD LOC OF FIRST CHAR IN ROW IN H&L
0223 22E287          SHLD   LOCBUF         ;STORE LOCXX IN BUFFER
0226 CD3204          CALL   FILL           ;CALL FILL ROW WITH SP CHAR SUBROUTINE
0229 C9              RET                   ;RETURN
                     ;
                     ;COMPENSATION SUBROUTINE COMRX
                     ;
022A 2ADE87  COMRX:  LHLD   LOCXX          ;LOAD LOCXX IN H&L
022D 1130F8          LXI    D,0F830H       ;LOAD COMPENSATION VALUE IN D&E
0230 19              DAD    D              ;ADD D&E TO H&L
0231 22DE87          SHLD   LOCXX          ;STORE RESULT IN LOCXX
0234 C9              RET
                     ;
                     ;CLEAR SCREEN ROUTINE
                     ;
0235 3EF0    ESCE:   MVI    A,0F0H         ;MOVE EOR CHAR TO A
0237 0619            MVI    B,19H          ;MOVE LOOP CTR START VALUE =19H=25D TO B
0239 115000          LXI    D,50H          ;MOVE 80D=50H TO D&E
023C 210080          LXI    H,8000H        ;MOVE 8000H TO H&L
                     ;
023F 77      LOADX:  MOV    M,A            ;MOVE EOR CHARACTER TO MEM
0240 19              DAD    D              ;ADD 80D=50H TO ADDRESS IN H&L
0241 05              DCR    B              ;DECREMENT B
0242 C23F02          JNZ    LOADX          ;CONTINUE LOOPING IF B NOT ZERO
                     ;
0245 210000          LXI    H,0000H        ;ZERO H&L
0248 22D387          SHLD   RCTAD          ;ZERO ROWCOUNT
024B 210080          LXI    H,8000H
024E 22D687          SHLD   TOPAD
0251 218087          LXI    H,8780H
0254 22E687          SHLD   BOTAD
0257 3E00            MVI    A,00H          ;ZERO A
0259 32D287          STA    CCTAD          ;ZERO COLUMN COUNT
025C 32D587          STA    CURSY          ;ZERO CURSOR Y POS
025F 32E487          STA    XFLG
0262 CD3C03          CALL   WP75           ;CALL LOAD CURSOR POSITION SUBROUTINE
0265 C9              RET
                     ;
                     ;ERASE TO END OF SCREEN ROUTINE
                     ;
0266 2AD687  ESCJ:   LHLD   TOPAD          ;LOAD TOP IN H&L
0269 EB              XCHG                  ;STORE TOP IN D&E
026A 2AD387          LHLD   RCTAD          ;LOAD ROW COUNT IN H&L
026D 19              DAD    D              ;ADD TOP+ROWCOUNT, YIELDING LOC OF
                                           ;FIRST CHAR IN PRESENT ROW
026E 22E087          SHLD   LOCPR          ;STORE LOCATION IN MEM
                     ;
```

```
0271 3E87            MVI    A,87H          ;LOAD 87H IN A
0273 BC              CMP    H              ;COMPARE H WITH 87H
0274 D27D02          JNC    VAR            ;IF NO CARRY, CONTINUE COMPARRISON
0277 CDEE02          CALL   COMRY          ;CALL COMPENSATION ROUTINE
027A C38902          JMP    FIN            ;JUMP TO FIN
027D C28902   VAR:   JNZ    FIN            ;IF NOT EQUAL END COMPARRISON
0280 3ECF            MVI    A,0CFH         ;LOAD CFH IN A
0282 BD              CMP    L              ;COMPARE L WITH CFH
0283 D28902          JNC    FIN            ;IF NO CARRY,LOCPR LESS THAN OR EQ TO 87CFH
0286 CDEE02          CALL   COMRY          ;CALL COMPENSATION ROUTINE
                     ;
                     ;
0289 2AD687   FIN:   LHLD   TOPAD          ;LOAD TOP IN H&L
028C 7D              MOV    A,L            ;MOVE L TO A
028D FE00            CPI    00H            ;COMPARE BYTE TO 00H
028F C2A102          JNZ    TROLL          ;IF NO COMPARRISON, JUMP TO TROLL
0292 7C              MOV    A,H            ;MOVE H TO A
0293 FE80            CPI    80H            ;COMPARE BYTE WITH 80H
0295 C2A102          JNZ    TROLL          ;IF NO COMPARRISON,JUMP TO TROLL
0298 218087          LXI    H,8780H        ;IF COMPARRISON,SET BOT=8780H
029B 22E687          SHLD   BOTAD
029E C3AB02          JMP    GNOME          ;JUMP TO GNOME
02A1 11B0FF   TROLL: LXI    D,0FFB0H       ;LOAD -80D=0FFB0H IN D&E
02A4 2AD687          LHLD   TOPAD          ;LOAD TOP IN H&L
02A7 19              DAD    D              ;ADD -80D TO TOP
02A8 22E687          SHLD   BOTAD
                     ;
02AB 3EF0     GNOME: MVI    A,0F0H         ;LOAD A WITH EOR CHAR (LOOP START)
                     ;
02AD 2AE087          LHLD   LOCPR          ;LOAD LOCPR IN H&L
02B0 77              MOV    M,A            ;MOVE EOR CHAR TO MEM
                     ;
02B1 7D              MOV    A,L            ;MOVE L TO A
02B2 FE80            CPI    80H            ;COMPARE YTE WITH 80H
02B4 C2D502          JNZ    WIZAR          ;IF NO COMPARRISON,JMP TO WIZAR
02B7 7C              MOV    A,H            ;MOVE H TO A
02B8 FE87            CPI    87H            ;COMPARE BYTE WITH 87H
02BA C2D502          JNZ    WIZAR          ;IF NO COMPARRISON, JMP TO WIZAR
                                           ;IF COMPARRISON,PROCEED TO GZONK
02BD EB       GZONK: XCHG                  ;STORE PRESENT LOC IN D&E
02BE 2AE687          LHLD   BOTAD          ;LOAD BOT IN H&L
02C1 7D              MOV    A,L            ;MOVE L TO A
02C2 BB              CMP    E              ;COMPARE E WITH A
02C3 C2CC02          JNZ    FUN            ;IF NO COMP, JUMP TO FUN
02C6 7C              MOV    A,H            ;MOVE H TO A
02C7 BA              CMP    D              ;COMPARE D WITH A
02C8 C2CC02          JNZ    FUN            ;IF NO COMP, JMP TO FUN
                                           ;IFCOMPARRISON,RETURN
02CB C9              RET                   ;RETURN
02CC 210080   FUN:   LXI    H,8000H        ;LOAD H&L WITH 8000H
02CF 22E087          SHLD   LOCPR          ;SET LOCPR =8000H
02D2 C3AB02          JMP    GNOME
                     ;
02D5 EB       WIZAR: XCHG                  ;STORE LOCPR IN D&E
02D6 2AE687          LHLD   BOTAD          ;LOAD BOT IN H&L
02D9 7D              MOV    A,L            ;MOVE L TO A
02DA BB              CMP    E              ;COMPARE E WITH A
02DB C2E402          JNZ    NUF            ;IF NO COMP,JMP TO NUF
02DE 7C              MOV    A,H            ;MOVE H TO A
02DF BA              CMP    D              ;COMPARE D WITH A
02E0 C2E402          JNZ    NUF            ;IF NO COMP, JMP TO NUF
                                           ;IF COMPARRISON,RETURN
02E3 C9              RET                   ;RETURN
02E4 215000   NUF:   LXI    H,50H          ;LOAD 80D=50H IN H&L
02E7 19              DAD    D              ;ADD 80D TO LOCPR (LOCPR IN D&E)
02E8 22E087          SHLD   LOCPR          ;STORE LOCPR IN MEM
02EB C3AB02          JMP    GNOME          ;JUMP TO GNOME
                     ;
                     ;COMPENSATION SUBROUTINE COMRY
                     ;
02EE 2AE087   COMRY: LHLD   LOCPR          ;LOAD LOCPR IN H&L
02F1 1130F8          LXI    D,0F830H               ;LOAD COM VALUE IN D&E
02F4 19              DAD    D              ;ADD COMPENSATION TO LOCPR
02F5 22E087          SHLD   LOCPR          ;STORE LOCPR IN MEM
02F8 C9              RET                   ;RETURN
                     ;LINE FEED ROUTINE
                     ;
02F9 C35F01   CTRLJ: JMP    ESCB
                     ;
                     ;CARRIAGE RETURN ROUTINE
                     ;
02FC 3E00     CTRLM: MVI    A,00H          ;ZERO A
02FE 32D287          STA    CCTAD          ;SET COLUMN COUNT=0
0301 CD3C03          CALL   WP75           ;CALL LOAD CURSOR POSITION SUBROUTINE
0304 C9              RET                   ;RETURN
                     ;
                     ;BACK SPACE ROUTINE
                     ;
0305 C3B001   CTRLH: JMP    ESCD
```

```
                        ;
                        ;ROWUP SUBROUTINE
                        ;
0308 2AD387   ROWUP:    LHLD    RCTAD           ;LOAD ROWCOUNT IN H&L
030B 11B0FF             LXI     D,0FFB0H        ;MOVE -80D=0FFB0H (2'S COMP) TO D&E
030E 19                 DAD     D               ;ADD -80D TO ROWCOUNT
030F 22D387             SHLD    RCTAD           ;STORE RESULT IN ROWCOUNT BUFFER

0312 21D587             LXI     H,CURSY         ;LOAD CURSOR Y POINTER ADDRESS IN H&L
0315 35                 DCR     M               ;DECREMENT CURSOR Y POINTER
0316 CD3C03             CALL    WP75            ;CALL LOAD CURSOR POSITION SUBROUTINE
0319 C9                 RET                     ;RETURN
                        ;
                        ;ROWDOWN SUBROUTINE
                        ;
031A 2AD387   ROWDN:    LHLD    RCTAD           ;LOAD ROWCOUNT IN H&L
031D 115000             LXI     D,50H           ;MOVE +80D=50H TO D&E
0320 19                 DAD     D               ;ADD +80D TO ROWCOUNT
0321 22D387             SHLD    RCTAD           ;STORE RESULT IN ROWCOUNT
0324 21D587             LXI     H,CURSY         ;LOAD CURSOR Y POINTER ADDRESS IN H&L
0327 34                 INR     M               ;INCREMENT CURSOR Y POINTER
0328 CD3C03             CALL    WP75            ;CALL LOAD CURSOR POSITION SUBROUTINE
032B C9                 RET                     ;RETURN
                        ;
                        ;COLUMN LEFT SUBROUTINE
                        ;
032C 21D287   COLLT:    LXI     H,CCTAD         ;LOAD COLUMN COUNT ADDRESS IN H&L
032F 35                 DCR     M               ;DECREMENT COLUMN COUNT
0330 CD3C03             CALL    WP75            ;CALL LOAD CURSOR POSITION SUBROUTINE
0333 C9                 RET                     ;RETURN
                        ;
                        ;COLUMN RIGHT SUBROUTINE
                        ;
0334 21D287   COLRT:    LXI     H,CCTAD         ;LOAD COLUMN COUNT ADDRESS IN H&L
0337 34                 INR     M               ;INCREMENT COLUMN COUNT
0338 CD3C03             CALL    WP75            ;CALL LOAD CURSOR POSITION SUBROUTINE
033B C9                 RET                     ;RETURN
                        ;
                        ;LOAD CURSOR POSITION SUBROUTINE
                        ;
033C 3E80     WP75:     MVI     A,80H           ;LOAD A WITH 80H, LOAD CURSOR POSITION COMMAND
033E D35F               OUT     CRCOM
0340 3AD287             LDA     CCTAD           ;LOAD A WITH CURSOR X POSITION
0343 D35E               OUT     CRDAT
0345 3AD587             LDA     CURSY           ;LOAD A WITH CURSOR Y POSITION
0348 D35E               OUT     CRDAT
034A C9                 RET                     ;RETURN
                        ;
                        ;
                        ;DISPLAY CHARACTER HANDLING SUBROUTINE
                        ;
034B 3AD287   DISPL:    LDA     CCTAD           ;LOAD COLUMN COUNT IN H&L
034E FE4F               CPI     4FH             ;COMPARE BYTE WITH 4FH=79D
0350 CA5A03             JZ      CTA             ;IF BYTE=4FH,COLUMN COUNT=LAST CHAR-
                                                ;ACTER IN ROW
0353 CD7E03             CALL    DIS1            ;CALL DIS1 SUBROUTINE
0356 CDBB03             CALL    DISA            ;CALL DISA SUBROUTINE
0359 C9                 RET                     ;RETURN
035A 2AD387   CTA:      LHLD    RCTAD           ;LOAD ROWCOUNT IN H&L
035D 7D                 MOV     A,L             ;LOAD LOW BYTE OF ROWCOUNT IN H$L
035E FE80               CPI     80H             ;COMPARE BYTE WITH 80H
0360 CA6A03             JZ      CTB             ;IF BYTE=80H,CONTINUE COMPARRISON
0363 CD7E03             CALL    DIS1            ;CALL DIS1 SUBROUTINE
0366 CDC303             CALL    DISB            ;CALL DISB SUBROUTINE
0369 C9                 RET                     ;RETURN
036A 7C       CTB:      MOV     A,H             ;MOVE UP BYTE OF ROWCOUNT TO H&L
036B FE07               CPI     07H             ;COMPARE BYTE WITH 07H
036D CA7703             JZ      CTC             ;IF BYTE=07H,END OF DISPLAY COND EXISTS
0370 CD7E03             CALL    DIS1            ;CALL DIS1 SUBROUTINE
0373 CDC303             CALL    DISB            ;CALL DISB SUBROUTINE
0376 C9                 RET                     ;RETURN
0377 CD7E03   CTC:      CALL    DIS1            ;CALL DIS1 SUBROUTINE
037A CDDA03             CALL    DISC            ;CALL DISC SUBROUTINE
037D C9                 RET                     ;RETURN
                        ;
                        ;SUBROUTINE DIS1
                        ;
037E 2AD687   DIS1:     LHLD    TOPAD           ;LOAD TOP IN H&L
0381 EB                 XCHG                    ;STORE TOP IN D&E
0382 2AD387             LHLD    RCTAD           ;LOAD ROWCOUNT IN H&L
0385 19                 DAD     D               ;ADD TOP+ROWCOUNT,  RESULT IN H&L
0386 22DA87             SHLD    LOC01           ;STORE LOCATION OF FIRST CHAR IN ROW
0389 EB                 XCHG                    ;STORE TOP+ROWCOUNT IN D&E
038A 210000             LXI     H,0000H         ;ZERO H&L
038D 3AD287             LDA     CCTAD           ;LOAD COLUMN COUNT IN A
0390 6F                 MOV     L,A             ;MOVE COLUMN COUNT TO L
0391 19                 DAD     D               ;CALCULATE LOCATION=
                                                ;TOP+ROWCOUNT+COLUMN COUNT,RESULT IN H&L
0392 22D887             SHLD    LOCAD           ;STORE LOCATION IN MEMORY
0395 3E87               MVI     A,87H           ;LOAD 87H IN A
```

```
0397 BC                    CMP     H            ;COMPARE H WITH 87H
0398 D2A103                JNC     NXTCM        ;IF NO CARRY,CONTNUE COMPARRISON
039B CDE603                CALL    COMRT        ;IF CARRY, CALL COMPENSATION ROUTINE
039E C3AD03                JMP     XSTAD        ;JUMP TO XSTAD
03A1 C2AD03        NXTCM:  JNZ     XSTAD        ;IF NOT EQUAL,END COMPARRISON
03A4 3ECF                  MVI     A,0CFH       ;LOAD 0CFH IN A
03A6 BD                    CMP     L            ;COMPARE L WITH 0CFH
03A7 D2AD03                JNC     XSTAD        ;IF NO CARRY, LOCATION LESS THAN
                                                ;OR EQUAL TO 87CFH
03AA CDE603                CALL    COMRT        ;IF CARRY, CALL COMPENSATION ROUTINE
03AD CDFB03        XSTAD:  CALL    EORT         ;CALL END OF ROW CHAR TEST ROUTINE
03B0 21E587                LXI     H,USCHR      ;LOAD USART CHAR ADD IN H&L
03B3 7E                    MOV     A,M          ;MOVE USART CHAR TO A
03B4 E63F                  ANI     3FH          ;MASK OFF UPPER 2 BITS OF CHAR
03B6 2AD887                LHLD    LOCAD        ;LOAD LOCATION IN H&L
03B9 77                    MOV     M,A          ;MOVE CHARACTER TO CHARACTER
                                                ;LOCATION IN DISPLAY MEMORY
03BA C9                    RET                  ;RETURN
                           ;
                           ;SUBROUTINE DISA
                           ;
03BB 21D287        DISA:   LXI     H,CCTAD      ;LOAD COLUMN COUNT ADD IN H&L
03BE 34                    INR     M            ;INCREMENT COLUMN COUNT
03BF CD3C03                CALL    WP75         ;CALL LOAD CURSOR POSITION SUBROUTINE
03C2 C9                    RET                  ;RETURN
                           ;
                           ;SUBROUTINE DISB
                           ;
03C3 3E00          DISB:   MVI     A,00H        ;ZERO A
03C5 32D287                STA     CCTAD        ;ZERO COLUMN COUNT
03C8 2AD387                LHLD    RCTAD        ;LOAD ROWCOUNT IN H&L
03CB 115000                LXI     D,50H        ;LOAD 80D=50H IN D&E
03CE 19                    DAD     D            ;ADD +80 TO ROWCOUNT
03CF 22D387                SHLD    RCTAD        ;STORE ROWCOUNT IN MEMORY
03D2 21D587                LXI     H,CURSY      ;LOAD CURSOR Y POSITION ADDRESS IN H&L
03D5 34                    INR     M            ;INCREMENT CURSOR Y POSITION
03D6 CD3C03                CALL    WP75         ;CALL LOAD CURSOR POSITION SUBROUTINE
03D9 C9                    RET                  ;RETURN
                           ;
                           ;SUBROUTINE DISC
                           ;
03DA 3E00          DISC:   MVI     A,00H        ;ZERO A
03DC 32D287                STA     CCTAD        ;ZERO COLUMN COUNT
03DF CD3C03                CALL    WP75         ;CALL LOAD CURSOR POSITION SUBROUTINE
03E2 CD0B04                CALL    SCROL
03E5 C9                    RET                  ;RETURN
                           ;
                           ;ADDRESS COMPENSATION SUBROUTINE
                           ;
03E6 2AD887        COMRT:  LHLD    LOCAD        ;LOAD CHARACTER LOCATION
03E9 1130F8                LXI     D,0F830H     ;LOAD COMPENSATION VALUE IN D&E
03EC 19                    DAD     D            ;ADD COMPENSATION TO LOCATION
03ED 22D887                SHLD    LOCAD        ;STORE MODIFIED LOCATION IN MEMORY
                           ;
03F0 2ADA87                LHLD    LOC01        ;LOAD LOCATION OF FIRST CHAR
                                                ;IN ROW IN H&L
03F3 1130F8                LXI     D,0F830H     ;LOAD COMPENSATION VALUE IN H&L
03F6 19                    DAD     D            ;ADD COMPENSATION TO LOC01
03F7 22DA87                SHLD    LOC01        ;STORE MODIFIED LOC01 IN MEMORY
03FA C9                    RET                  ;RETURN
                           ;
                           ;END OF ROW TEST ROUTINE
                           ;
03FB 2ADA87        EORT:   LHLD    LOC01        ;LOAD LOCATION OF FIRST CHAR
                                                ;IN ROW IN H&L
03FE 7E                    MOV     A,M          ;MOVE FIRST CHAR IN ROW TO A REG
03FF FEF0                  CPI     0F0H         ;COMPARE CHAR WITH 0F0 (END OF ROW CHAR)
0401 C20A04                JNZ     XIT          ;IF NO COMPARRISON, EXIT
0404 22E287                SHLD    LOCBUF       ;STORE FIRST CHAR IN ROW ADD IN LOCBUF
0407 CD3204                CALL    FILL         ;CALL FILL ROW WITH SPACE CODES SUBROUTINE
040A C9            XIT:    RET                  ;RETURN
                           ;
                           ;SCROLL SUBROUTINE
                           ;
040B 2AD687        SCROL:  LHLD    TOPAD        ;LOAD TOP IN H&L
040E 22E287                SHLD    LOCBUF       ;STORE FIRST CHAR IN ROW ADD IN LOCBUF
0411 CD3204                CALL    FILL         ;CALL FILL ROW WITH SPACE CODES SUBROUTINE
0414 2AD687                LHLD    TOPAD        ;MOVE TOP TO H&L
0417 7D                    MOV     A,L          ;MOVE LOWER BYTE OF TOP TO A
0418 FE80                  CPI     80H          ;COMPARE TOP WITH MAX VALUE
041A C22A04                JNZ     DUCK         ;IF NO COMPARRISON EXISTS, CONTINUE SCROL
041D 7C                    MOV     A,H          ;MOVE UPPER BYTE OF TOP TO A
041E FE87                  CPI     87H          ;COMPARE TOP WITH MAX VALUE
0420 C22A04                JNZ     DUCK         ;IF NO COMPARRISON EXISTS, CONTINUE SCROL
                                                ;IF COMPARRISON, TOP=MAX VALUE=8780H
0423 210080                LXI     H,8000H      ;IF COMPARRISON, MODIFY TOP TO TOP=8000H
0426 22D687                SHLD    TOPAD        ;STORE MODIFIED TOPAD IN MEMORY
0429 C9                    RET                  ;RETURN
042A 115000        DUCK:   LXI     D,50H        ;MOVE 80D=50H TO D&E
```

```
042D 19                     DAD     D           ;ADD 80D=50H TO TOP
042E 22D687                 SHLD    TOPAD       ;STORE MODIFIED TOPAD IN MEMORY
0431 C9                     RET                 ;RETURN
                            ;
                            ;FILL SUBROUTINE
                            ;
0432 2AE287         FILL:   LHLD    LOCBUF      ;LOAD LOCATION OF FIRST CHAR IN ROW
                                                ;OR FIRST CHAR IN TOP ROW IN H&L
0435 115000                 LXI     D,50H       ;LOAD 80D=50H IN D&E
0438 19                     DAD     D           ;CALCULATE LOCATION OF LAST CHAR IN ROW
0439 22DC87                 SHLD    LOC80       ;STORE LOCATION OF LAST CHAR IN ROW IN MEMORY
043C 012020                 LXI     B,2020H     ;LOAD SPACE CHARACTERS IN B&C
043F 210000                 LXI     H,0000H     ;ZERO H&L
0442 39                     DAD     SP          ;ADD SP TO H&L, TRANSFERRING SP TO H&L
0443 EB                     XCHG                ;STORE STACK POINTER IN D&E
0444 2ADC87                 LHLD    LOC80       ;LOAD LOCATION OF LAST CHAR IN ROW IN H&L
0447 F9                     SPHL                ;LOAD LAST CHAR LOCATION IN SP
                            ;
0448 C5                     PUSH B              ;EXECUTE THE LIST OF PUSH B COMMANDS TO
0449 C5                     PUSH B              ;FILL THE LINE WITH BLANK CHARACTERS
044A C5                     PUSH B
044B C5                     PUSH B
044C C5                     PUSH B
044D C5                     PUSH B
044E C5                     PUSH B
044F C5                     PUSH B
0450 C5                     PUSH B
0451 C5                     PUSH B
0452 C5                     PUSH B
0453 C5                     PUSH B
0454 C5                     PUSH B
0455 C5                     PUSH B
0456 C5                     PUSH B
0457 C5                     PUSH B
0458 C5                     PUSH B
0459 C5                     PUSH B
045A C5                     PUSH B
045B C5                     PUSH B
045C C5                     PUSH B
045D C5                     PUSH B
045E C5                     PUSH B
045F C5                     PUSH B
0460 C5                     PUSH B
0461 C5                     PUSH B
0462 C5                     PUSH B
0463 C5                     PUSH B
0464 C5                     PUSH B
0465 C5                     PUSH B
0466 C5                     PUSH B
0467 C5                     PUSH B
0468 C5                     PUSH B
0469 C5                     PUSH B
046A C5                     PUSH B
046B C5                     PUSH B
046C C5                     PUSH B
046D C5                     PUSH B
046E C5                     PUSH B
046F C5                     PUSH B
0470 EB                     XCHG                ;STACK POINTER TRANSFERRED TO H&L
0471 F9                     SPHL                ;RESTORE STACK
0472 C9                     RET                 ;RETURN
                            ;
                            ;
                            ;
                            ;
                            ;8275 INTERRUPT SERVICE SUBROUTINE
                            ;
                            ;8257 REINITIALIZATION
                            ;
0473 3E00           RT75:   MVI     A,MDC57     ;MOVE MODE CLEAR COMMAND TO A
0475 D348                   OUT     PMD57       ;OUTPUT MODE CLEAR COMMAND TO 8257
                            ;
0477 2AD687                 LHLD    TOPAD       ;LOAD TOP IN H&L
047A 7D                     MOV     A,L         ;LOAD CH 2 START ADD, LOW BYTE, IN A
047B D344                   OUT     PC2SA       ;OUTPUT CH 2 START ADD TO 8257
047D 7C                     MOV     A,H         ;LOAD CH 2 START ADD, UP BYTE, IN A
047E D344                   OUT     PC2SA       ;OUTPUT CH 2 START ADD TO 8257
                            ;
0480 7D                     MOV     A,L         ;LOAD LOW BYTE OF TOP IN A
0481 2F                     CMA                 ;COMPLEMENT A
0482 6F                     MOV     L,A         ;LOAD COMPLEMENTED VALUE IN L
0483 7C                     MOV     A,H         ;LOAD UP BYTE OF TOP IN A
0484 2F                     CMA                 ;COMPLEMENT A
0485 67                     MOV     H,A         ;LOAD COMPLEMENTED VALUE IN H
0486 23                     INX     H           ;INCREMENT H&L, YIELDING 2'S COMPLEMENT
                                                ;OF TOP IN A
0487 11CF87                 LXI     D,87CFH     ;LOAD 87CFH IN D&E
048A 19                     DAD     D           ;ADD H&L TO D&E, YIELDING 87CFH-TOP
048B 110080                 LXI     D,8000H     ;LOAD D&E WITH 8000H
048E 19                     DAD     D           ;ADD 8000H TO 87CF-TOP
```

```
048F 7D              MOV    A,L           ;MOVE LOW BYTE OF CH 2 TC TO A
0490 D345            OUT    PC2TC         ;OUTPUT CH 2 TC TO 8257
0492 7C              MOV    A,H           ;MOVE UP BYTE OF CH 2 TC TO A
0493 D345            OUT    PC2TC         ;OUTPUT CH 2 TC TO 8257
                     ;
0495 210080          LXI    H,8000H       ;LOAD 8000HIN H&L
0498 7D              MOV    A,L           ;MOVE LOW BYTE OF CH 3 START ADD TO A
0499 D346            OUT    PC3SA         ;OUTPUT CH 3 START ADD TO 8257
049B 7C              MOV    A,H           ;MOVE UP BYTE OF CH 3 START ADD TO A
049C D346            OUT    PC3SA         ;OUTPUT CH 3 START ADD TO 8257
                     ;
049E 21CF87          LXI    H,87CFH       ;LOAD CH 3 TC VALUE IN H&L
04A1 7D              MOV    A,L           ;MOVE L TO A
04A2 D347            OUT    PC3TC         ;OUTPUT CH 3 TC TO 8257
04A4 7C              MOV    A,H           ;MOVE H TO A
04A5 D347            OUT    PC3TC         ;OUTPUT CH 3 TC TO 8257
04A7 3E84            MVI    A,MDS57       ;LOAD A WITH MODE SET VALUE
04A9 D348            OUT    PMD57         ;OUTPUT MODE SET TO 8257
                     ;
                     ;
                     ;KEYBOARD POLLING ROUTINE
                     ;
04AB DB6F     KPOLL: IN     KCOM          ;INPUT FIFO STATUS
04AD E607            ANI    07H           ;MASK STATUS, SAVE BITS 0-2
04AF CAB504          JZ     ZIP           ;TEST FOR CHARACTER PRESENT
04B2 CDB604          CALL   XMIT          ;CALL CHARACTER TRANSMIT ROUTINE
04B5 C9       ZIP:   RET                  ;RETURN
                     ;
                     ;CHARACTER TRANSMIT SUBROUTINE
                     ;
04B6 DB6E     XMIT:  IN     KDAT          ;INPUT FIFO CHARACTER
04B8 EEC0            XRI    0C0H          ;INVERT TOP 2 BITS
04BA 21F804          LXI    H,BSET3       ;LOAD BASE ADD OF TABLE 3 IN H&L
04BD 110000          LXI    D,0000H       ;ZERO D&E
04C0 5F              MOV    E,A           ;LOAD E WITH CHARACTER FROM FIFO
04C1 19              DAD    D             ;CALCULATE ADD IN LOOKUP TABLE
                                          ;CONTAINING ASCII CHARACTERS
                                          ;CORRESPONDING TO KEY POSITION IN MATRIX
04C2 DBFB     USZ:   IN     CNCTL         ;INPUT USART STATUS
04C4 E601            ANI    01H           ;MASK STATUS, SAVE TRANSMITTER READY BIT
04C6 CAC204          JZ     USZ           ;TEST READY BIT
04C9 7E              MOV    A,M           ;MOVE ASCII CHAR TO A
04CA E67F            ANI    7FH           ;MASK BIT 7
04CC D3FA            OUT    CNOUT         ;OUTPUT CHAR FROM USART
04CE C9              RET                  ;RETURN
                     ;
                     ;DUMY ROUTINE DEFINITION
                     ;
04CF C9       DUMY:  RET                  ;RETURN
                     ;
                     ;
                     ;
                     ;
                     ;TABLE DEFINITION AREA
                     ;
                     ;
                     ;
04D0 CF04     BSET1: DW     DUMY
04D2 3901            DW     ESCA
04D4 5F01            DW     ESCB
04D6 7D01            DW     ESCC
04D8 B001            DW     ESCD
04DA 3502            DW     ESCE
04DC CF04            DW     DUMY
04DE CF04            DW     DUMY
04E0 EB01            DW     ESCH
04E2 CF04            DW     DUMY
04E4 6602            DW     ESCJ
04E6 FD01            DW     ESCK
04E8 CF04            DW     DUMY
04EA CF04            DW     DUMY
04EC CF04            DW     DUMY
04EE CF04            DW     DUMY
                     ;
                     ;
04F0 0503     BSET2: DW     CTRLH
04F2 F902            DW     CTRLJ
04F4 FC02            DW     CTRLM
04F6 CF04            DW     DUMY
                     ;
                     ;
04F8 30       BSET3: DB     30H           ;DUMMY CHARACTER
04F9 30              DB     30H
04FA 30              DB     30H
04FB 30              DB     30H
04FC 30              DB     30H
04FD 30              DB     30H
04FE 30              DB     30H
04FF 30              DB     30H
```

```
0500 30        800        DB      30H
0501 30        801        DB      30H
0502 30        802        DB      30H
0503 30        803        DB      30H
0504 30        804        DB      30H
0505 30        805        DB      30H
0506 30        806        DB      30H
0507 30        807        DB      30H
0508 1B        808        DB      1BH                 ; ESC
0509 0A        809        DB      0AH                 ; LF
050A 2C        810        DB      2CH                 ; ,
050B 0D        811        DB      0DH                 ; CR
050C 20        812        DB      20H                 ; SP
050D 7F        813        DB      7FH                 ; DEL
050E 2E        814        DB      2EH                 ; .
050F 2F        815        DB      2FH                 ; /
0510 5A        816        DB      5AH                 ; Z
0511 58        817        DB      58H                 ; X
0512 4D        818        DB      4DH                 ; M
0513 56        819        DB      56H                 ; V
0514 30        820        DB      30H
0515 43        821        DB      43H                 ; C
0516 4E        822        DB      4EH                 ; N
0517 42        823        DB      42H                 ; B
0518 30        824        DB      30H                 ; 0
0519 2D        825        DB      2DH                 ; -
051A 4F        826        DB      4FH                 ; O
051B 4C        827        DB      4CH                 ; L
051C 39        828        DB      39H                 ; 9
051D 3A        829        DB      3AH                 ; :
051E 50        830        DB      50H                 ; P
051F 3B        831        DB      3BH                 ; ;
0520 53        832        DB      53H                 ; S
0521 44        833        DB      44H                 ; D
0522 4B        834        DB      4BH                 ; K
0523 47        835        DB      47H                 ; G
0524 41        836        DB      41H                 ; A
0525 46        837        DB      46H                 ; F
0526 4A        838        DB      4AH                 ; J
0527 48        839        DB      48H                 ; H
0528 57        840        DB      57H                 ; W
0529 45        841        DB      45H                 ; E
052A 49        842        DB      49H                 ; I
052B 54        843        DB      54H                 ; T
052C 51        844        DB      51H                 ; Q
052D 52        845        DB      52H                 ; R
052E 55        846        DB      55H                 ; U
052F 59        847        DB      59H                 ; Y
0530 32        848        DB      32H                 ; 2
0531 33        849        DB      33H                 ; 3
0532 38        850        DB      38H                 ; 8
0533 35        851        DB      35H                 ; 5
0534 31        852        DB      31H                 ; 1
0535 34        853        DB      34H                 ; 4
0536 37        854        DB      37H                 ; 7
0537 36        855        DB      36H                 ; 6
0538 30        856        DB      30H
0539 30        857        DB      30H
053A 30        858        DB      30H
053B 30        859        DB      30H
053C 30        860        DB      30H
053D 30        861        DB      30H
053E 30        862        DB      30H
053F 30        863        DB      30H
0540 30        864        DB      30H
0541 30        865        DB      30H
0542 30        866        DB      30H
0543 30        867        DB      30H
0544 30        868        DB      30H
0545 30        869        DB      30H
0546 30        870        DB      30H
0547 30        871        DB      30H
0548 30        872        DB      30H
0549 30        873        DB      30H
054A 3C        874        DB      3CH                 ; <
054B 30        875        DB      30H
054C 30        876        DB      30H
054D 30        877        DB      30H
0517 42        823        DB      42H                 ; B
0518 30        824        DB      30H                 ; 0
0519 2D        825        DB      2DH                 ; -
051A 4F        826        DB      4FH                 ; O
051B 4C        827        DB      4CH                 ; L
051C 39        828        DB      39H                 ; 9
051D 3A        829        DB      3AH                 ; :
051E 50        830        DB      50H                 ; P
051F 3B        831        DB      3BH                 ; ;
0520 53        832        DB      53H                 ; S
0521 44        833        DB      44H                 ; D
0522 4B        834        DB      4BH                 ; K
```

```
0523 47          835     DB      47H           ; G
0524 41          836     DB      41H           ; A
0525 46          837     DB      46H           ; F
0526 4A          838     DB      4AH           ; J
0527 48          839     DB      48H           ; H
0528 57          840     DB      57H           ; W
0529 45          841     DB      45H           ; E
052A 49          842     DB      49H           ; I
052B 54          843     DB      54H           ; T
052C 51          844     DB      51H           ; Q
052D 52          845     DB      52H           ; R
052E 55          846     DB      55H           ; U
052F 59          847     DB      59H           ; Y
0530 32          848     DB      32H           ; 2
0531 33          849     DB      33H           ; 3
0532 38          850     DB      38H           ; 8
0533 35          851     DB      35H           ; 5
0534 31          852     DB      31H           ; 1
0535 34          853     DB      34H           ; 4
0536 37          854     DB      37H           ; 7
0537 36          855     DB      36H           ; 6
0538 30          856     DB      30H
0539 30          857     DB      30H
053A 30          858     DB      30H
053B 30          859     DB      30H
053C 30          860     DB      30H
053D 30          861     DB      30H
053E 30          862     DB      30H
053F 30          863     DB      30H
0540 30          864     DB      30H
0541 30          865     DB      30H
0542 30          866     DB      30H
0543 30          867     DB      30H
0544 30          868     DB      30H
0545 30          869     DB      30H
0546 30          870     DB      30H
0547 30          871     DB      30H
0548 30          872     DB      30H
0549 30          873     DB      30H
054A 3C          874     DB      3CH           ; <
054B 30          875     DB      30H
054C 30          876     DB      30H
054D 30          877     DB      30H
054E 3E          878     DB      3EH           ; >
054F 3F          879     DB      3FH           ; ?
0550 30          880     DB      30H
0551 30          881     DB      30H
0552 5D          882     DB      5DH           ; ]
0553 30          883     DB      30H
0554 30          884     DB      30H
0555 30          885     DB      30H
0556 30          886     DB      30H
0557 30          887     DB      30H
0558 30          888     DB      30H
0559 3D          889     DB      3DH           ; =
055A 30          890     DB      30H
055B 5C          891     DB      5CH           ; \
055C 29          892     DB      29H           ; )
055D 2A          893     DB      2AH           ; *
055E 30          894     DB      30H
055F 2B          895     DB      2BH           ; +
0560 30          896     DB      30H
0561 30          897     DB      30H
0562 5B          898     DB      5BH           ; [
0563 30          899     DB      30H
0564 30          900     DB      30H
0565 30          901     DB      30H
0566 30          902     DB      30H
0567 30          903     DB      30H
0568 30          904     DB      30H
0569 30          905     DB      30H
056A 30          906     DB      30H
056B 30          907     DB      30H
056C 30          908     DB      30H
056D 30          909     DB      30H
056E 30          910     DB      30H
056F 30          911     DB      30H
0570 22          912     DB      22H           ; "
0571 23          913     DB      23H           ; #
0572 28          914     DB      28H           ; (
0573 25          915     DB      25H           ; %
0574 21          916     DB      21H           ; !
0575 24          917     DB      24H           ; $
0576 27          918     DB      27H           ; '
0577 26          919     DB      26H           ; &
0578 30          920     DB      30H
0579 30          921     DB      30H
057A 30          922     DB      30H
057B 30          923     DB      30H
057C 30          924     DB      30H
```

```
057D 30        925      DB    30H
057E 30        926      DB    30H
057F 30        927      DB    30H
0580 30        928      DB    30H
0581 30        929      DB    30H
0582 30        930      DB    30H
0583 30        931      DB    30H
0584 30        932      DB    30H
0585 30        933      DB    30H
0586 30        934      DB    30H
0587 30        935      DB    30H
0588 30        936      DB    30H
0589 30        937      DB    30H
058A 30        938      DB    30H
058B 30        939      DB    30H
058C 30        940      DB    30H
058D 30        941      DB    30H
058E 30        942      DB    30H
058F 30        943      DB    30H
0590 1A        944      DB    1AH           ; SUB
0591 18        945      DB    18H           ; CAN
0592 0D        946      DB    0DH           ; CR
0593 16        947      DB    16H           ; SYN
0594 30        948      DB    30H
0595 03        949      DB    03H           ; ETX
0596 0E        950      DB    0EH           ; SO
0597 02        951      DB    02H           ; STX
0598 30        952      DB    30H
0599 1F        953      DB    1FH           ; US
059A 0F        954      DB    0FH           ; SI
059B 0C        955      DB    0CH           ; FF
059C 30        956      DB    30H
059D 30        957      DB    30H
059E 10        958      DB    10H           ; DLE
059F 30        959      DB    30H
05A0 13        960      DB    13H           ; DC3
05A1 04        961      DB    04H           ; EOT
05A2 0B        962      DB    0BH           ; VT
05A3 07        963      DB    07H           ; BEL
05A4 01        964      DB    01H           ; SOH
05A5 06        965      DB    06H           ; ACK
05A6 0A        966      DB    0AH           ; LF
05A7 08        967      DB    08H           ; BS
05A8 17        968      DB    17H           ; ETB
05A9 05        969      DB    05H           ; ENQ
05AA 09        970      DB    09H           ; HT
05AB 14        971      DB    14H           ; DC4
05AC 11        972      DB    11H           ; DC1
05AD 12        973      DB    12H           ; DC2
05AE 15        974      DB    15H           ; NAK
05AF 19        975      DB    19H           ; EM
05B0 30        976      DB    30H
05B1 30        977      DB    30H
05B2 30        978      DB    30H
05B3 30        979      DB    30H
05B4 30        980      DB    30H
05B5 30        981      DB    30H
05B6 30        982      DB    30H
05B7 30        983      DB    30H
05B8 30        984      DB    30H
05B9 30        985      DB    30H
05BA 30        986      DB    30H
05BB 30        987      DB    30H
05BC 30        988      DB    30H
05BD 30        989      DB    30H
05BE 30        990      DB    30H
05BF 30        991      DB    30H
05C0 30        992      DB    30H
05C1 30        993      DB    30H
05C2 30        994      DB    30H
05C3 30        995      DB    30H
05C4 30        996      DB    30H
05C5 30        997      DB    30H
05C6 30        998      DB    30H
05C7 30        999      DB    30H
05C8 30       1000      DB    30H
05C9 30       1001      DB    30H
05CA 30       1002      DB    30H
05CB 30       1003      DB    30H
05CC 30       1004      DB    30H
05CD 30       1005      DB    30H
05CE 30       1006      DB    30H
05CF 30       1007      DB    30H
05D0 30       1008      DB    30H
05D1 30       1009      DB    30H
05D2 1D       1010      DB    1DH           ; GS
05D3 30       1011      DB    30H
05D4 30       1012      DB    30H
05D5 30       1013      DB    30H
```