MOTOROLA
Semiconductor Products Inc.

AN-851
Application Note

MOTOROLA MC6845 CRTC SIMPLIFIES VIDEO DISPLAY CONTROLLERS

Prepared by
Charles Melear and Jack Browne
Microprocessor Applications Engineering
Austin, Texas

The need for displaying visual information by the general business community has found widespread applications. Banks, airports, department stores, and other businesses need rapid display of visual information at points of sale and points of use. Much of this information is generated by people who have only a limited knowledge of the electronics involved. Therefore, they must rely on the equipment used to automatically receive data, digest it, and display it on a video monitor. Systems could range in complexity from those which display only a few lines of data to complicated word processors. Historically, character printers gave way to line printers. However, obtaining hard copy is cumbersome and slow, and a considerable amount of paper is used. Much of this information is used only momentarily and then discarded, such as inventory checks or airport flight schedules. The efficiency of low cost, high performance video monitors have

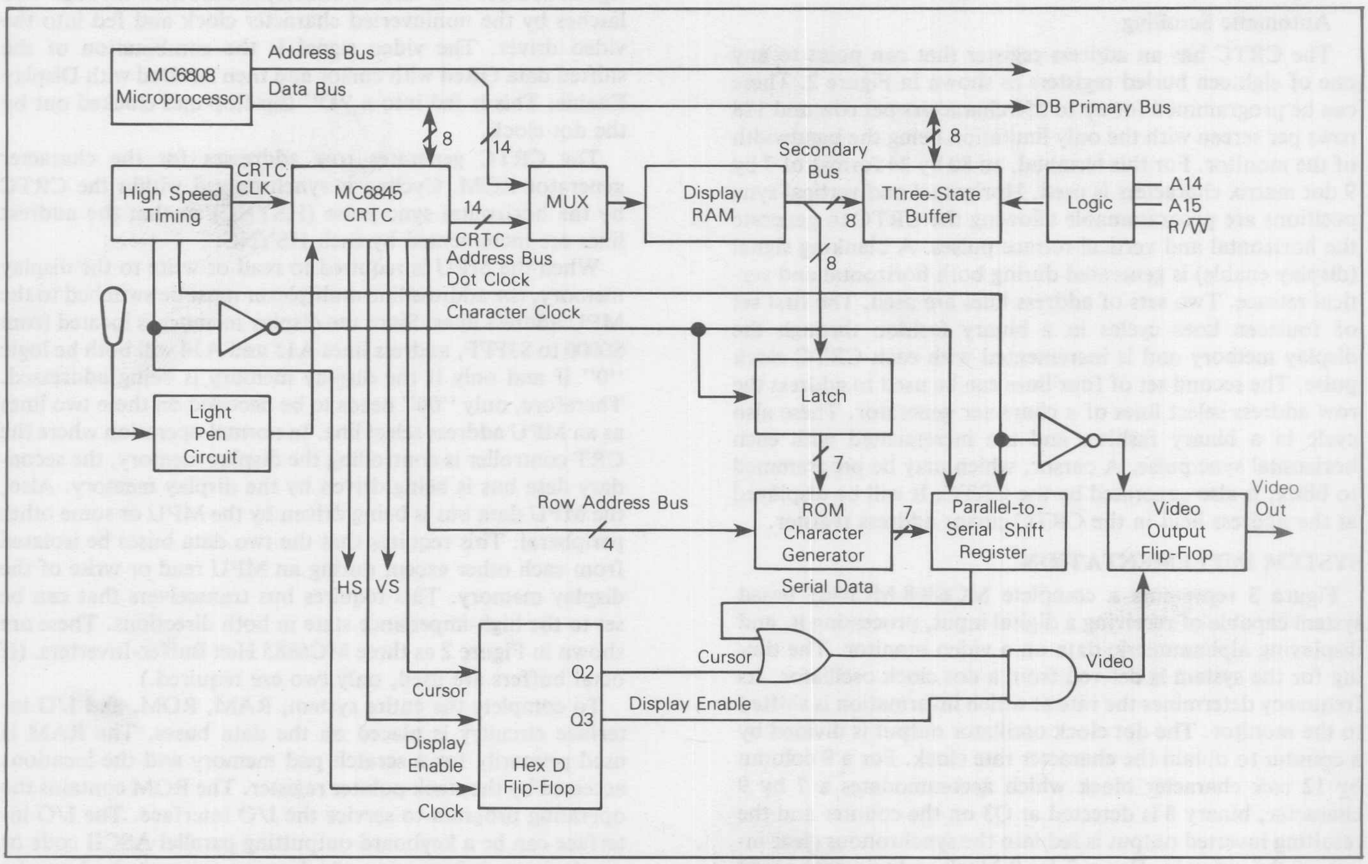

FIGURE 1 — CRT Controller Application

made the transition from hard copy to visual display even more advantageous. As video monitors have come into general use, the requirement for cost savings in the controller has intensified. LSI circuits have been appearing which meet that need.

The Motorola MC6845 CRT controller (CRTC) can economically solve many of the problems encountered with video monitor displays. This is acomplished by using an innovative design aimed at complete control of the monitor with intervention by the MPU only when new information is put into the display memory. The problems to be solved by the MC6845 in a raster scan video display controller are: cost, number of required components, amount of intervention by MPU, timing and synchronization of signals, and software, among others.

Today, CRT controllers can be built using an MC6845 which require approximately 25 ICs plus the extra chips required for memory. This number represents only a fraction of the parts required just a few years ago when SSI and MSI logic devices were used. CRT controllers were built using SSI and MSI logic devices which required well over one hundred ICs. With the MC6845 approach, the number of ICs can be reduced to approximately 25 plus those required for memory.

To illustrate the capabilities of an MC6845 based terminal, the software and "rough" hardware considerations used in its design are discussed. The terminal, as shown in Figure 1, has the following features:

Blinking Cursor	Move Cursor Up One Line
Carriage Return	Paging
Backspace	Home Cursor
Line Feed	Clear Screen
Automatic Scrolling	

The CRTC has an address register that can point to any one of eighteen buried registers as shown in Figure 2. These can be programmed for up to 256 characters per row and 128 rows per screen with the only limitation being the bandwidth of the monitor. For this terminal, an 80 by 24 format of 7 by 9 dot matrix characters is used. Horizontal and vertical sync positions are programmable allowing the CRTC to generate the horizontal and vertical retrace pulses. A blanking signal (display enable) is generated during both horizontal and vertical retrace. Two sets of address lines are used. The first set of fourteen lines cycles in a binary fashion through the display memory and is incremented with each CRTC clock pulse. The second set of four lines can be used to address the row address select lines of a character generator. These also cycle in a binary fashion and are incremented with each horizontal sync pulse. A cursor, which may be programmed to blink, is also generated by the CRTC. It will be displayed at the address held in the CRTC cursor address register.

SYSTEM IMPLEMENTATION

Figure 3 represents a complete MC6808-MC6845 based system capable of receiving a digital input, processing it, and displaying alphanumeric data on a video monitor. The timing for the system is derived from a dot clock oscillator. Its frequency determines the rate at which information is shifted to the monitor. The dot clock oscillator output is divided by a counter to obtain the character rate clock. For a 9 column by 12 row character block which accommodates a 7 by 9 character, binary 8 is detected at Q3 on the counter and the resulting inverted output is fed into the synchronous clear input of the counter. For a 7 by 9 block, a logic gate could detect binary 6 on Q0, Q1, and Q2. It is important to use a counter with a synchronous clear so the clear pulse will be one dot clock period wide. The character clock (generated by the rising edge of Q3) serves as a shift/load signal for the output shift register and a clock to latch data from the display memory. The CRTC clock (generated by the trailing edge of Q2) is used to clock the MC6845 CRTC. Each character rate clock increments the address lines (MA0-MA13) of the MC6845. The display memory must be capable of being controlled by either the MPU or the CRTC. Therefore, the address lines for both devices (A0-A13 and MA0-MA13) are routed through multiplexers such as the SN74LS157. The MPU takes control of the display memory only when a new character is to be written. The output of the multiplexer addresses the memory.

As shown in Figure 3, the 8K $\times$ 8 static display memory requires 10 address lines for the address bus of the memory elements and 3 address lines for the 3-to-8 line decoder which drives the chip selects of the memory elements. The output of the display memory is fed into an 8-bit latch (74LS374) and is clocked into the latch on the next character clock. This latch helps to prevent address line jitter which could present spurious data to the character generator ROM. The character clock is used to latch data into the SN74LS374. This creates a one character clock delay from the time that an address becomes valid to the memory until data is presented to the character generator ROM. The character clock is also used to load the parallel word from the character generator ROM into the shift register, producing a second character clock delay. Once the shift register is loaded the dot clock is used to serially shift data from the shift register to the video driver.

In order to synchronize both the display enable and cursor output with the shift register output, a two CRTC clock delay must be imposed. Both signals are synchronous with the CRTC address lines. To implement this delay, the two signals (cursor and display enable) are clocked through two latches by the noninverted character clock and fed into the video driver. The video signal is the combination of the shifted data ORed with cursor and then ANDed with Display Enable. This is fed into a "D" flip-flop and clocked out by the dot clock.

The CRTC generates row addresses for the character generator ROM. Cycling is synchronized within the CRTC by the horizontal sync pulse (HSYNC) so that the address lines are incremented by each HSYNC.

When the MPU is required to read or write to the display memory, the address line multiplexer must be switched to the MPU address lines. Since the display memory is located from $0000 to $3FFF, address lines A15 and A14 will both be logic "0" if and only if the display memory is being addressed. Therefore, only "00" needs to be decoded on these two lines as an MPU address select line. In normal operation where the CRT controller is controlling the display memory, the secondary data bus is being driven by the display memory. Also, the MPU data bus is being driven by the MPU or some other peripheral. This requires that the two data buses be isolated from each other except during an MPU read or write of the display memory. This requires bus transceivers that can be set to the high-impedance state in both directions. These are shown in Figure 2 as three MC6885 Hex Buffer-Inverters. (If octal buffers are used, only two are required.)

To complete the entire system, RAM, ROM, and I/O interface circuitry is placed on the data buses. The RAM is used primarily for a scratch pad memory and the locations accessed by the stack pointer register. The ROM contains the operating program to service the I/O interface. The I/O interface can be a keyboard outputting parallel ASCII code or row/column information. As long as some method can be programmed to receive digital data and transfer it onto the data bus, the CRT controller, using an MC6845, can display that information on a video display.

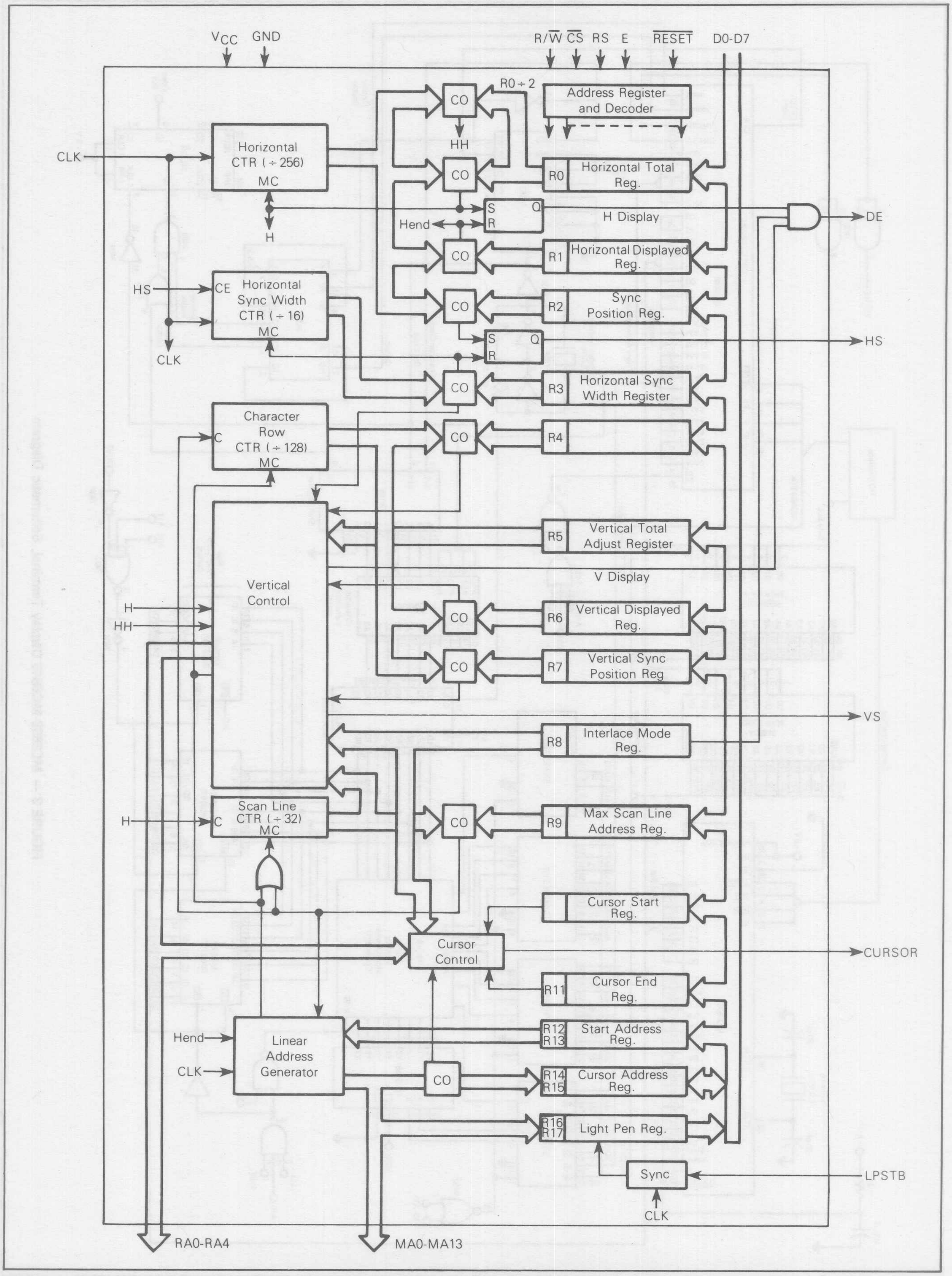

FIGURE 2 — MC6845 CRT Controller Block Diagram

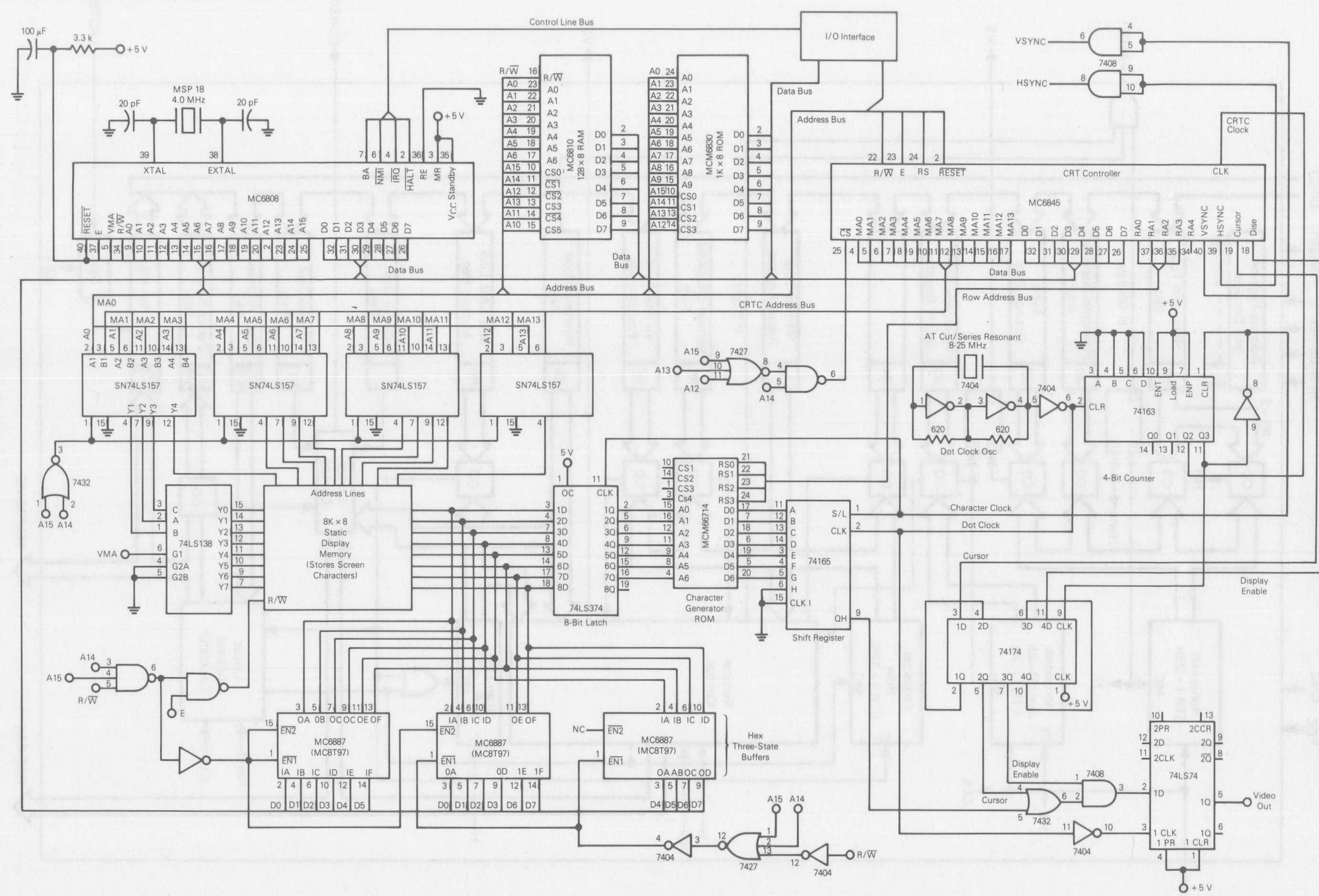

FIGURE 3 — MC6808-MC6845 Display Terminal, Schematic Diagram

DEVICE IMPLEMENTATION

The MC6845 CRTC has 18 programmable registers (R0-R17 in Figure 2) that control: the horizontal and vertical sync, number of characters per row, number of scan lines per row, number of rows per screen, the portion of memory to be displayed, cursor format and position, and the choice of one of three interlace modes.

The first four registers, R0 through R3, are concerned with the horizontal format. These registers determine the number of characters to be displayed, their width, and horizontal position. Programming considerations are based on the period of the monitor, i.e., the sweep plus retrace time. Also, the horizontal sync pulse should occur slightly after the beam is driven past the right-hand side of the screen. It is important to note that the beam is overdriven on the left side of the screen as well as the right. This means that a certain time elapses between the horizontal sync pulse and when the beam sweeps onto the screen from the left and is at the position for it to start displaying data.

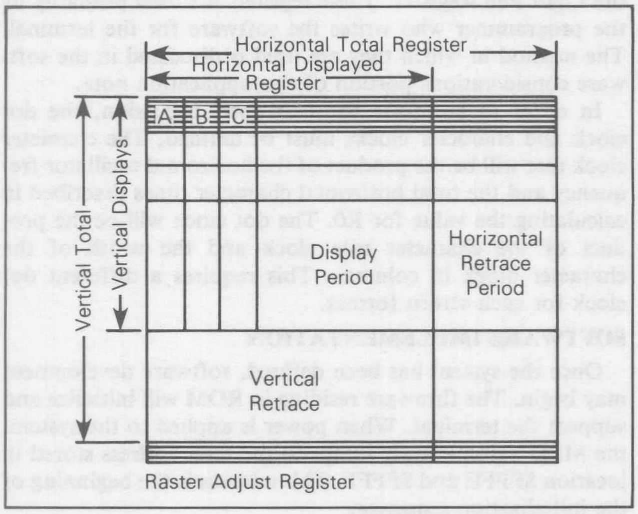

FIGURE 4 — Monitor Period Divided Into Character Times

The period of the monitor should be divided into character times (see Figure 4). This will define the width of a character block and this value will be stored in the Horizontal Total Register (R0). A video monitor will require about 20% of the period to be reserved for retrace (see Figure 5), as opposed to about 35% for a TV. This means that the Horizontal Displayed Register (R1), which contains the number of characters to be displayed per row, will not usually exceed about 80% of the value in R0. If R0 contains a very small number, each character will be very wide. Likewise, if R0 contains a large number, the characters will be very narrow. The Horizontal Sync Position Register (R2) is programmed in character times and should be positioned such that it will occur slightly after the beam is driven past the right margin of the screen. The Horizontal Sync Width Register (R3), programmed in character times, should provide sufficient width to allow the discharge of the circuitry driving the horizontal sweep. It should be noted that the value in R0 usually exceeds the sum of the values in R2 and R3. This is to allow for the time required for the beam to sweep onto the screen from the left margin since it could be overdriven to the left.

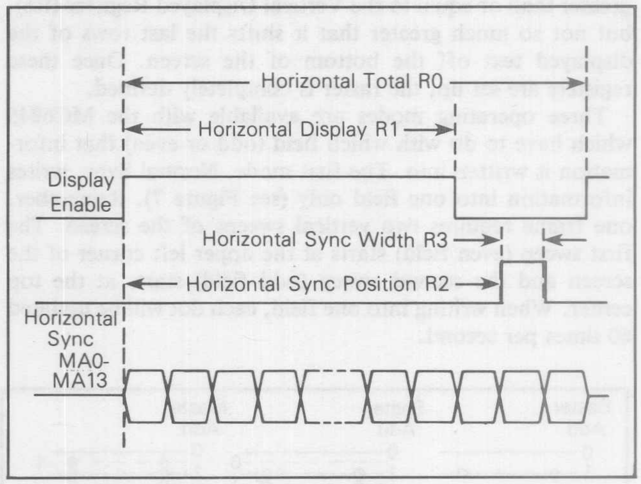

FIGURE 5 — Horizontal Timing

Four registers, R4-R7, are used to set up the vertical format (see Figure 6). The frequency of the horizontal oscillator and the vertical refresh rate must be known. Generally, the vertical refresh rate is 60 Hz. The horizontal frequency, usually 15,750 Hz, divided by the frame refresh rate is equal to the total number of scan lines per frame. The vertical sync pulse requires 16 scan lines. This means that the programmer cannot use the total number of scan lines for information display. A character block which contains the character to be displayed, plus spacing columns to the right and additional scan lines on the bottom, is chosen by the programmer. Typically, a character generator ROM with a 7×9 matrix element will be placed in a 9×12 character block. The Vertical Total Register (R4) contains the number of character rows per screen which is equal to the total number of scan lines divided by the height of the character block. This height is programmed in scan lines and placed in the Max Scan Line Address Register (R9). The number of scan lines left over is written into the Vertical Adjust Register (R5). All scan lines must be accounted for so the CRT controller will exactly match the vertical refresh rate; otherwise, the display will "swim" or have a wavy motion. The Vertical Displayed Register (R6) contains the number of character rows that the programmer wishes to be displayed. The Vertical Sync Position Register (R7) contains the position of the vertical sync pulse. This number, programmed in character times, must be

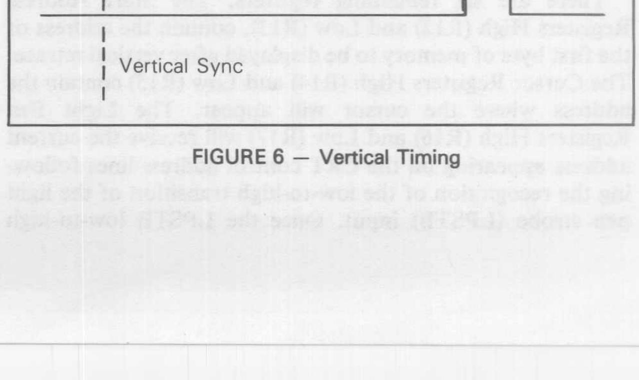

FIGURE 6 — Vertical Timing

greater than or equal to the Vertical Displayed Register (R6), but not so much greater that it shifts the last rows of the displayed text off the bottom of the screen. Once these registers are set up, the raster is completely defined.

Three operating modes are available with the MC6845 which have to do with which field (odd or even) that information is written into. The first mode, Normal Sync, writes information into one field only (see Figure 7). Remember, one frame requires two vertical sweeps of the screen. The first sweep (even field) starts at the upper left corner of the screen and the second sweep (odd field) starts at the top center. When writing into one field, each dot will be updated 60 times per second.

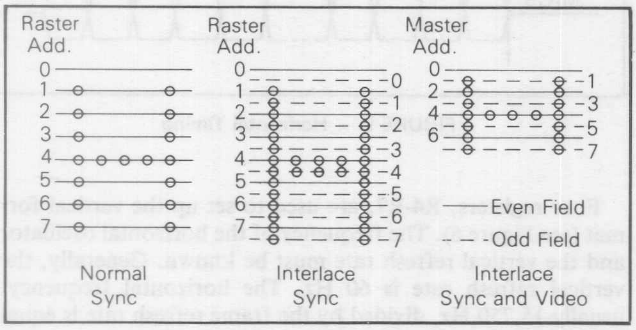

FIGURE 7 — Interlace Mode (R8)

The second mode, Interlace Sync, writes in both fields. The odd field is an exact duplicate of the even field. Essentially, the same information is written twice. This has the advantage of making the letters appear to have solid vertical lines thus improving resolution. However, each dot is now refreshed only 30 times per second which may cause an objectionable flicker on the screen. This flicker cannot be perceived by all people due to variances in eye sight. Also, the persistance of the phosphor will moderate the effect of the flicker.

The third mode, Interlace Sync and Video, also writes in both fields. However, one half the character is written in each field. This means an eight row character block in this mode will have four scan lines in the even field and four in the odd field making a character only half the height of the other two modes. This allows the highest screen density for the MC6845. These modes are programmed in the Interlace Mode Register (R8).

The MC6845 also controls the cursor format and blink rate (see Figure 8). Each character row has a certain number of scan lines as defined by the Max Scan Line Address Register (R9). The Cursor Start Register (R10) specifies on which scan line the cursor begins. Also, this register contains a bit that specifies whether the cursor will blink or not blink. Another bit specifies the blink rate which is either one sixteenth or one thirty second of the field rate.

The Cursor End Register (R11) specifies the scan line at the bottom of the cursor. If the same number is specified for both the starting and ending scan line, the cursor will be one line tall.

There are six remaining registers. The Start Address Registers High (R12) and Low (R13), contain the address of the first byte of memory to be displayed after vertical retrace. The Cursor Registers High (R14) and Low (R15) contain the address where the cursor will appear. The Light Pen Registers High (R16) and Low (R17) will receive the current address appearing on the CRT control address lines following the recognition of the low-to-high transition of the light pen strobe (LPSTB) input. Once the LPSTB low-to-high

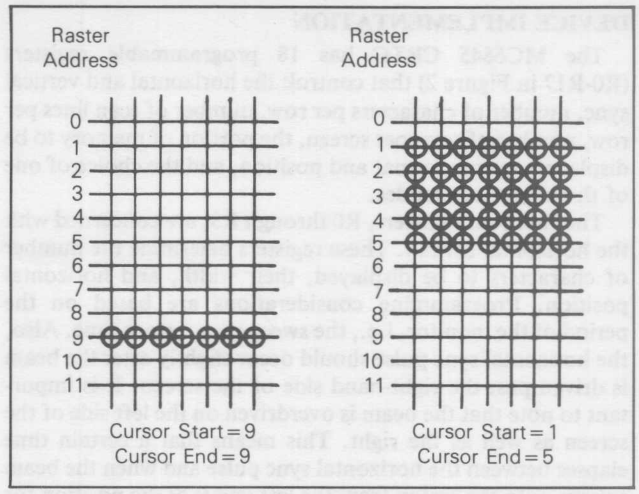

FIGURE 8 — Cursor Start and End Register

transition is recognized, the next low-to-high CRTC clock transition latches the address information and loads it into the Light Pen Register. These registers are used primarily by the programmer who writes the software for the terminal. The method in which they are used is discussed in the software considerations portion of this application note.

In order to complete the hardware discussion, the dot clock and character clocks must be defined. The character clock rate will be the product of the horizontal oscillator frequency and the total horizontal character times described in calculating the value for R0. The dot clock will be the product of the character rate clock and the width of the character block in columns. This requires a different dot clock for each screen format.

SOFTWARE IMPLEMENTATION

Once the system has been defined, software development may begin. The firmware residing in ROM will initialize and support the terminal. When power is applied to the system, the MPU automatically jumps to the reset address stored in location $FFFE and $FFFF. This address is the beginning of the initialization sequence.

After a power-on-reset, the display memory is initialized (to avoid a flash of false data), the eighteen buried registers of the CRT controller are initialized, and characters are accepted from the keyboard. Some control characters will be decoded to implement the following features:

Carriage Return	Move Cursor Up One Line
Backspace	Paging
Line Feed	Home Cursor
Clear Screen	

Scrolling up or down will be done automatically as required.

The software was developed using the concepts of structured programming. The first two routines which were written support the hardware development and debugging. The first routine is named CHARGN and its flowchart is shown in Figure 9. This routine initializes the display memory with successive ASCII character codes which help identify addressing problems. The second routine is named CRTINT and initializes the CRT controller (see flowchart in Figure 10). The register values to implement an 80 by 24 display are read from the ROM and stored into the buried registers of the CRT controller. Again, it is important to initialize the display memory prior to initializing the MC6845, to avoid a flash of false data. After the system has been initialized by running this program (as listed in Figure 11), waveforms, timing, and data may be checked, thus speeding the design phase.

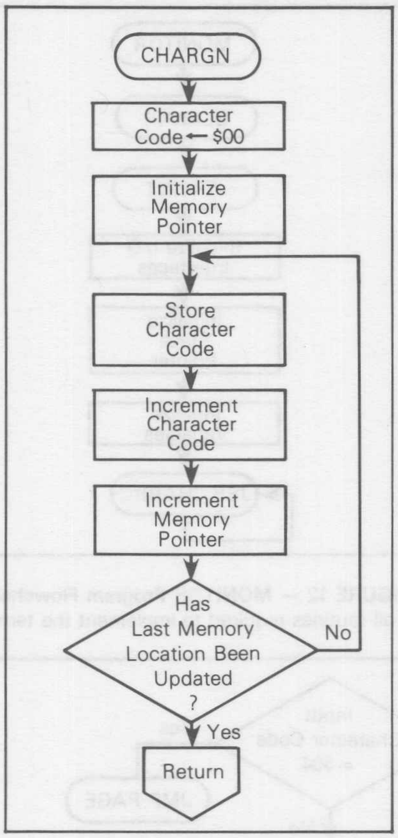

FIGURE 9 — CHARGN Subroutine Flowchart
Loads ASCII character codes into display memory.

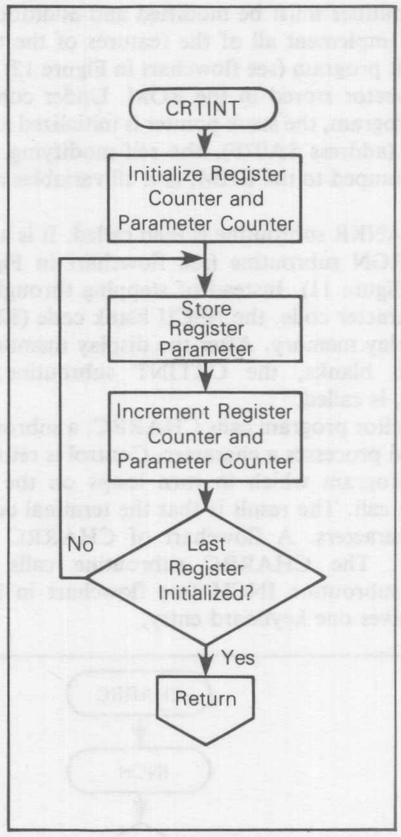

FIGURE 10 — CRTINT Subroutine Flowchart
Initializes the CRTC registers with the previously calculated values stored in the ROM.

```
PAGE  001  BOOT      .SA:1

00001             4000  A CRTCAD EQU    $4000
00002             4001  A CRTCRG EQU    $4001
00003A E3FE                    ORG    $E3FE
00004A E3FE       E0    A             FCB    $E0,0
00005A E000                    ORG    $E000
00006A E000 4F            CHARGN CLRA          FILL SCREEN WITH CHARACTER
00007A E001 CE 0000  A         LDX    #$0000
00008A E004 A7 00    A CHAR    STAA   0,X      STORE CHARACTER
00009A E006 4C                 INCA            GET NEXT CHARACTER
00010A E007 08                 INX             MOVE TO NEXT LOCATION
00011A E008 8C 1000  A         CPX    #$1000   FINISHED
00012A E00B 26 F7 E004         BNE    CHAR     FINISHED?
00013A E00D 5F            CRTINT CLRB          INITIALIZE CRTC
00014A E00E CE E022  A         LDX    #TABLE
00015A E011 F7 4000  A CRTIN1  STAB   CRTCAD   SELECT CRTC REGISTER
00016A E014 A6 00    A         LDAA   0,X
00017A E016 B7 4001  A         STAA   CRTCRG
00018A E019 08                 INX
00019A E01A 5C                 INCB
00020A E01B C1 10    A         CMPB   #$10
00021A E01D 26 F2 E011         BNE    CRTIN1
00022A E01F 01            LOOPER NOP
00023A E020 20 FD E01F         BRA    LOOPER
00024A E022       30    A TABLE  FCB    $30,$26,$2B,$02,$14,$01
00025A E028       12    A        FCB    $12,$13,$00,$0B,$40,$08,$00,$00,$00
00026                          END
TOTAL ERRORS 00000--00000
```

FIGURE 11 — CRT DEM Listing
This program, resident in PROM, will initialize the display memory with successive ASCII characters. This will allow initial checkout of the hardware.

These routines must be modified and additional routines written to implement all of the features of the terminal. A MONITOR program (see flowchart in Figure 12) is called by the reset vector stored in the ROM. Under control of the monitor program, the stack pointer is initialized at the end of the RAM (address $A07F), the self-modifying sections of code are dumped to the RAM, and all variables are initialized.

The BLANKR subroutine is then called. It is a revision of the CHARGN subroutine (see flowchart in Figure 9 and listing in Figure 11). Instead of stepping through the entire ASCII character code, the ASCII blank code ($20) is stored in the display memory. After the display memory has been filled with blanks, the CRTINT subroutine, discussed previously, is called.

The monitor program calls CHARRC, a subroutine which accepts and processes a character. Control is returned to the monitor program which in turn loops on the CHARRC subroutine call. The result is that the terminal continuously accepts characters. A flowchart of CHARRC appears in Figure 13. The CHARRC subroutine calls the input character subroutine INCH (see flowchart in Figure 14), which receives one keyboard entry.

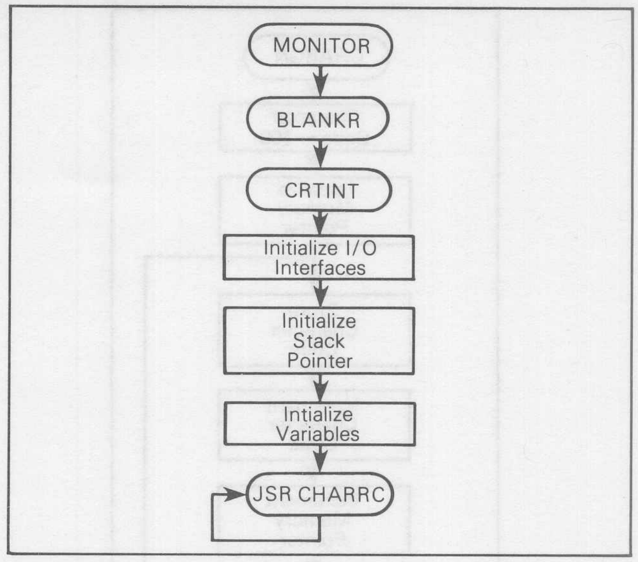

FIGURE 12 — MONITOR Program Flowchart
Calls all routines required to implement the terminal.

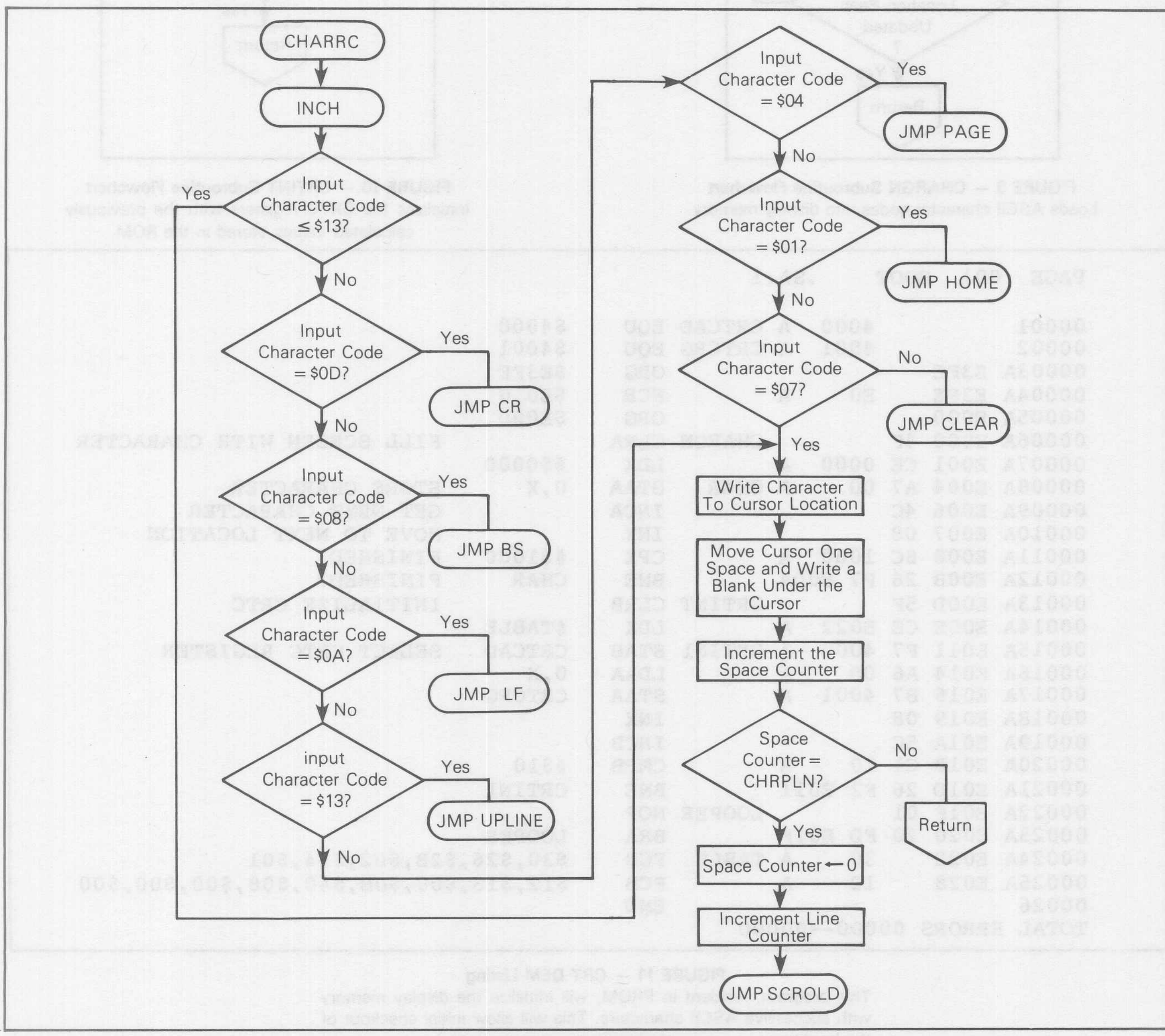

FIGURE 13 — CHARRC Subroutine
Accepts characters from keyboard, moves cursor, and decodes all special characters.

The special functions are implemented using control characters which are not normally utilized by CRT terminals. Table 1 lists the feature and its control character and indicates which routine processes the command. Each time one of the special characters is received, a jump to the appropriate routine occurs. All characters received from the keyboard, except for the special control characters, are written to the current cursor location, the cursor is moved one space, and a blank is written under the cursor.

To facilitate carriage returns, a space counter (SPACES) is used. It keeps track of the cursor displacement from the beginning of the current line. The counter (SPACES) is used whenever a carriage return key is pressed. The cursor is moved back to the beginning of the line by subtracting the number of spaces from the Cursor Registers (R14 and R15). A line feed is then generated by adding the number of characters per line to the Cursor Register.

The CRT controller treats the screen memory as a linear array such that the last space of a line and the first space of the next line are located at adjacent memory locations. When the cursor is at the end of a line and another character is input, the cursor moves to the first of the next line. The space counter (SPACES) must be reset.

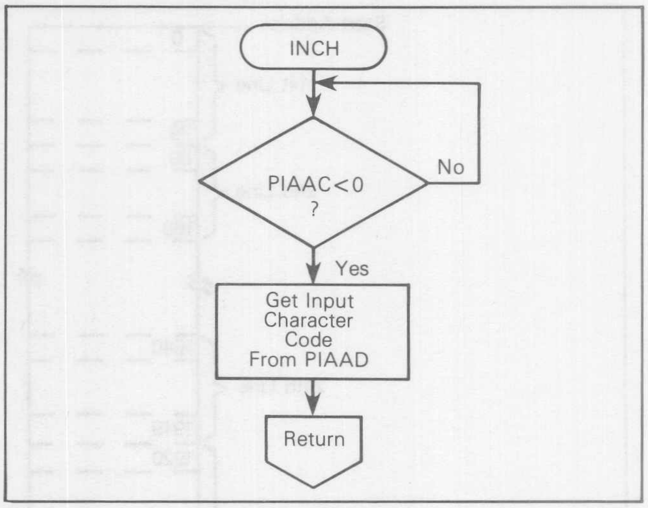

FIGURE 14 — INCH Subroutine Flowchart
Polls PIA A Control Register until IRQA1 is set, then the data is retrieved from the PIA A Data Register.

TABLE 1 — Subroutine Implementation of Terminal Features

Feature	Keyboard Entry	Subroutine Name	Flowcharted in Figure	Result
Scroll Up	None	SCROLU	15b	Called whenever a line feed is generated. Will add a line to bottom of screen when necessary.
Carriage Return	CR Key	CR	16	Generates carriage return, calls LF.
Line Feed	LF Key	LF	17	Generates line feed, calls SCROLU.
Back Space	© H	BS	18	Generates back space and blanks under cursor, calls SCROLD when cursor moves back to previous line.
Move Cursor Up One Line	© $	UPLINE	19	Moves cursor up one line, calls SCROLD.
Move to Next Page	© D	PAGE	20	Moves to same place on next page.
Home Cursor	© A	HOME	21	Moves cursor.
Clear Screen	© G	CLEAR	22	Clears page starting at cursor.
Scroll Down	None	SCROLD	23	Called whenever cursor moves back one line. Adds a new line to top of screen when necessary.

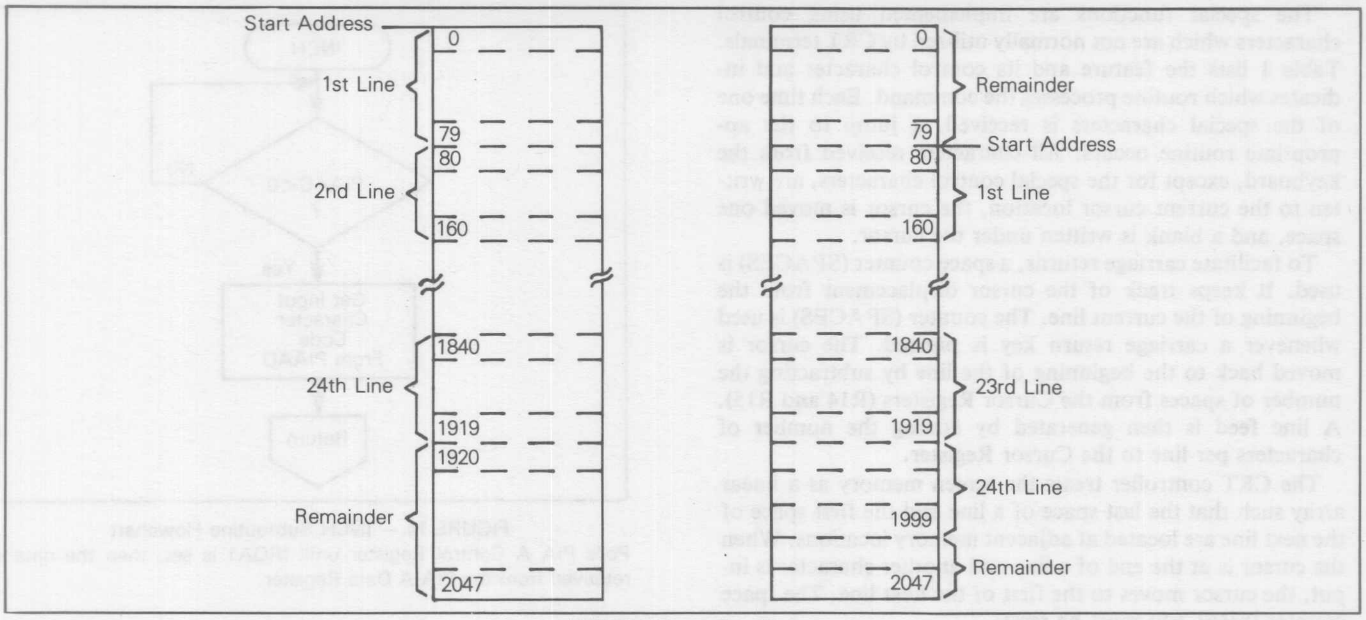

FIGURE 15a — Scrolling

Performed by changing the Start Address in R12 and R13 in the CRTC. This example shows how an 80 x 24 display is scrolled up one line.

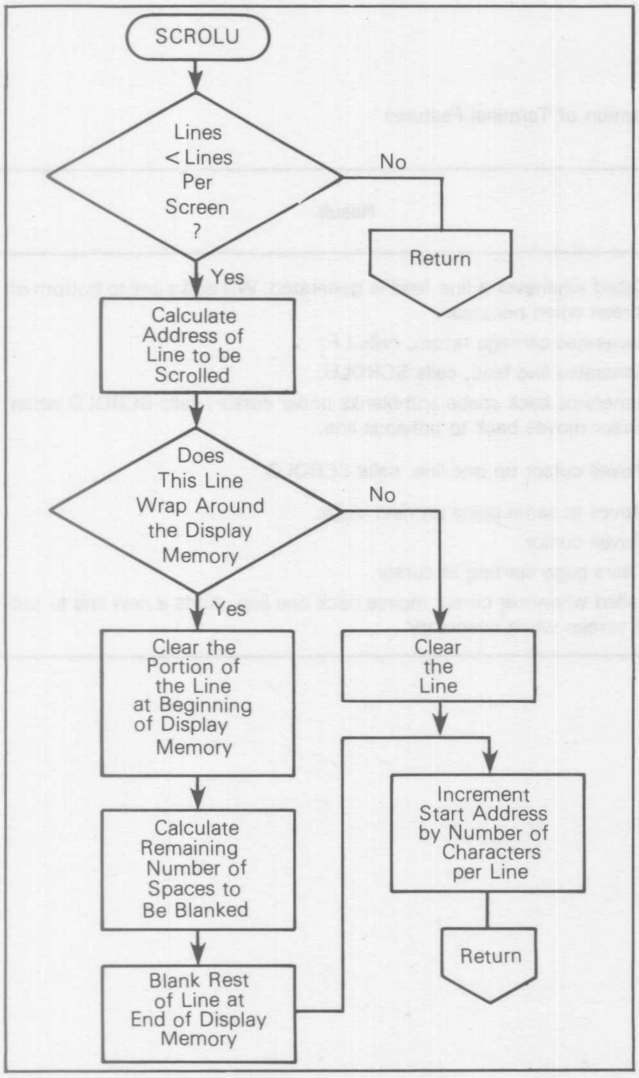

FIGURE 15b — SCROLU Subroutine Flowchart

The 14-bit cursor address is checked to see if cursor has moved off the screen. If so, the 14-bit start address is incremented to add a new line (with the cursor) at the bottom.

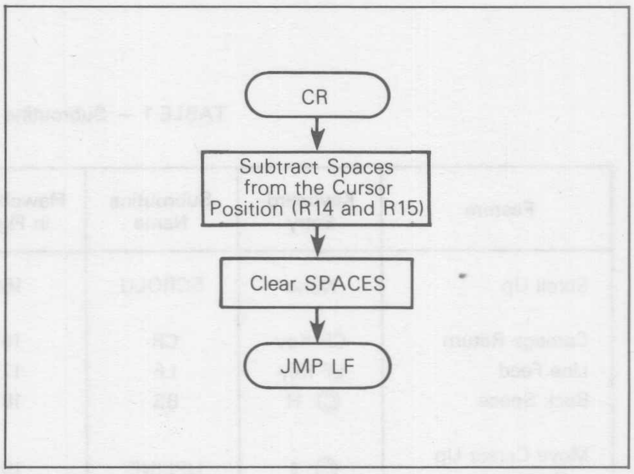

FIGURE 16 — CR Subroutine Flowchart

Generates a cursor return by subtracting SPACES (the space counter) from the current cursor position in R14 and R15 of the CRT. Jumps to LF to generate a line feed.

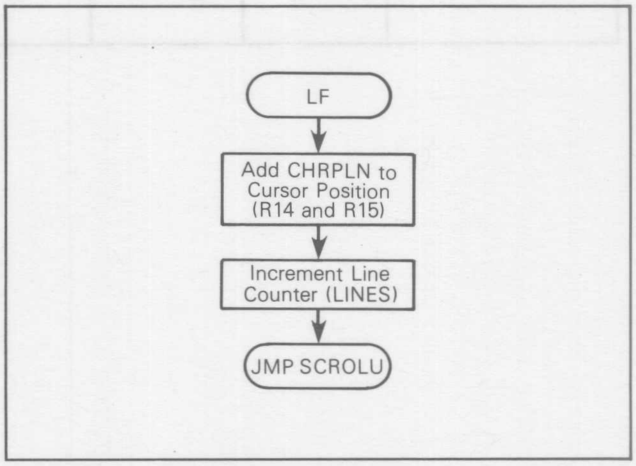

FIGURE 17 — LF Subroutine Flowchart

Generates a line feed by adding the number of characters per line to the current cursor position stored in R14 and R15 of the CRTC. Jumps to SCROLU to see if a new line should be scrolled on the page.

Whenever SPACES is reset, the scroll up routine (SCROLU) is called to determine if the cursor is still on the CRT screen. If the cursor has moved off the bottom of the CRT screen, then the Start Address Registers (R12 and R13) are adjusted to scroll a new line in at the bottom of the screen. The SCROLU routine is illustrated in Figure 15a and flowcharted in Figure 15b.

Flowcharts, describing implementations of the special features listed in Table 1, are presented in Figures 15-23. Notes at the bottom of each figure explain the algorithms employed.

When the routine to generate a line feed LF (flowcharted in Figure 17) is called, the cursor is moved down one line. Because this may move the cursor off the screen, the SCROLU routine, to scroll up one line, is called. Similarly, whenever the backspace routine or the routine to move the cursor up one line (UPLINE, see flowchart in Figure 19) is called, the cursor may be moved back to the previous line. This may also move the cursor off the top of the screen requiring the routine which scrolls down one line (SCROLD, see flowchart in Figure 23) to be called. The scrolling, whether up or down, is implemented by modifying the starting address stored in CRTC Registers R12 and R13. Scrolling up is implemented by adding or subtracting the number of characters per line to the start address. Note that the CRTC Cursor Registers R14 and R15 are the only read/write registers. This requires the use of a variable to retain the current start address duplicated in R12 and R13 (write only).

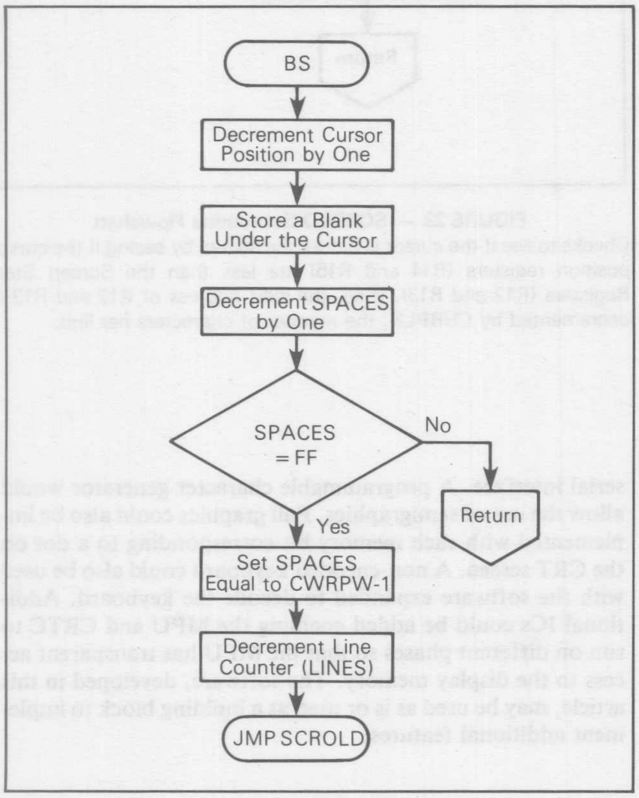

FIGURE 18 — BS Subroutine Flowchart

Backspaces and blanks under cursor. Jumps to SCROLD and checks if the cursor has moved off the top of the screen.

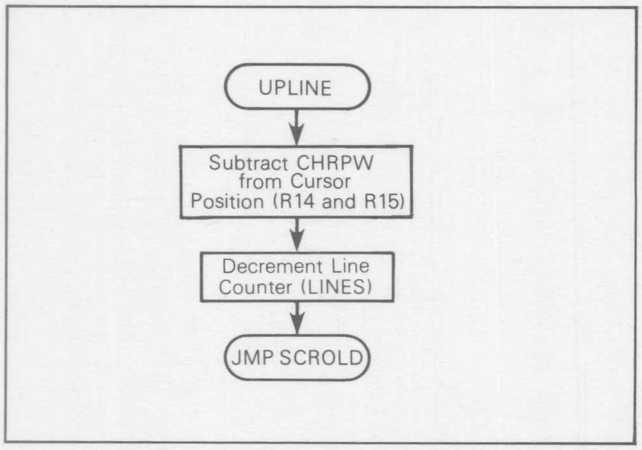

FIGURE 19 — UPLINE Subroutine Flowchart

Moves the cursor up one line by subtracting the number of characters per line from current cursor position stored in R14 and R15 of the CRTC. Jumps to SCROLD to check if the cursor has moved off the top of the screen.

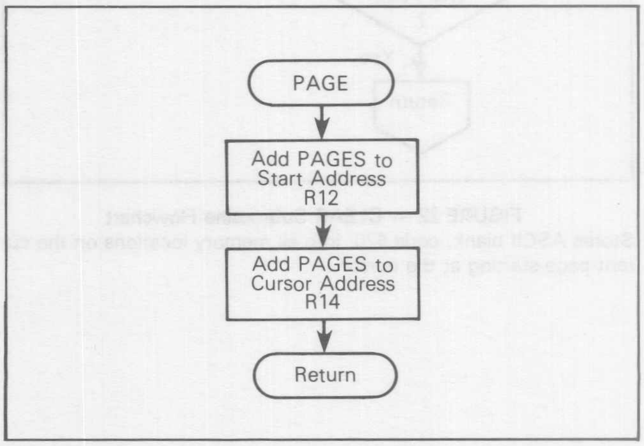

FIGURE 20 — PAGE Subroutine Flowchart

Moves to the same position on the next page by adding PAGES to the high order byte of the starting address (R12) and the high order byte of the cursor position (R14). PAGES multiplied by $100 equals the number of characters per page.

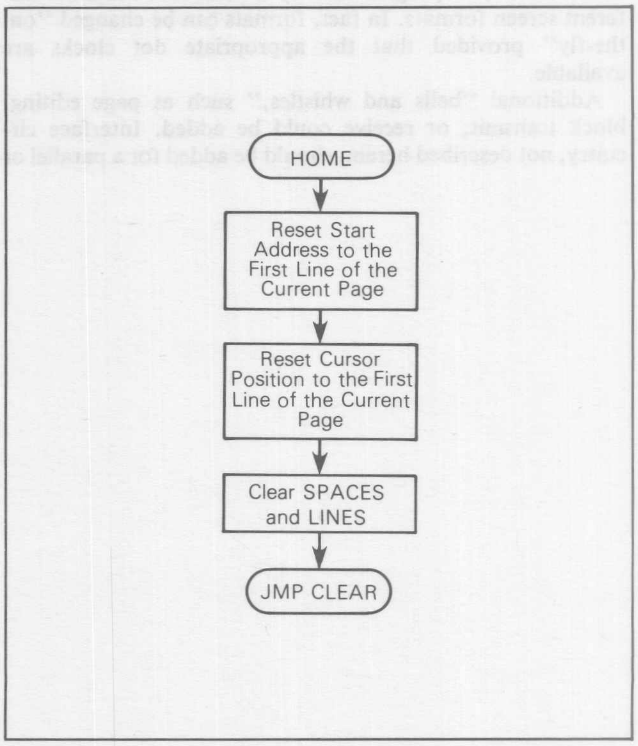

FIGURE 21 — HOME Subroutine Flowchart

Reset start address and cursor position to the beginning of the current page, then clear SPACES and jump to CLEAR to put blanks in each display memory element of the current page.

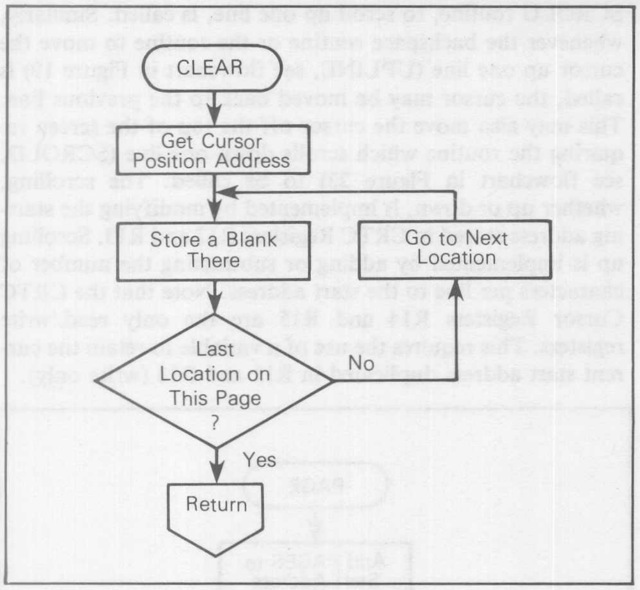

FIGURE 22 — CLEAR Subroutine Flowchart
Stores ASCII blank, code $20, into all memory locations on the current page starting at the cursor.

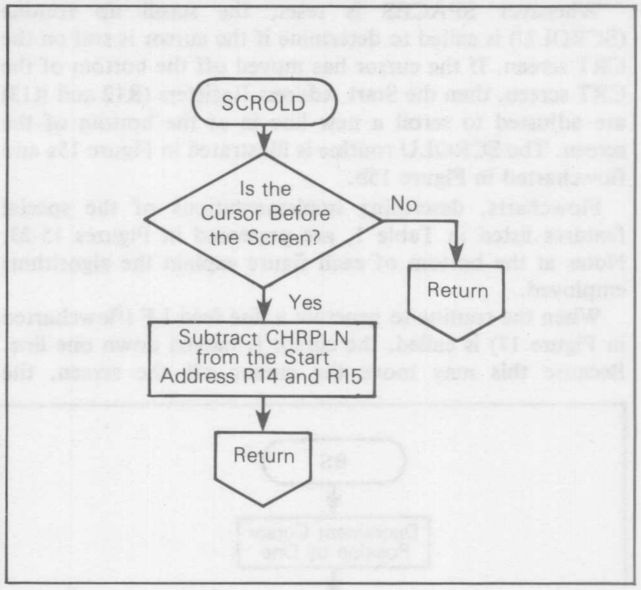

FIGURE 23 — SCROLD Subroutine Flowchart
Checks to see if the cursor is before the screen by seeing if the cursor position registers (R14 and R15) are less than the Screen Start Registers (R12 and R13). If so, the start address of R12 and R13 is decremented by CHRPLN, the number of characters per line.

A complete listing of the software appears in Figure 24 and will implement all the described features. A semi-structured approach is utilized to simplify changes or additions. The MC6845 CRTC supplies the video and sync pulses to the CRT and may be programmed by the MC6808 MPU for different screen formats. In fact, formats can be changed "on-the-fly" provided that the appropriate dot clocks are available.

Additional "bells and whistles," such as page editing, block transmit, or receive could be added. Interface circuitry, not described herein, should be added for a parallel or serial interface. A programmable character generator would allow the use of semigraphics. Full graphics could also be implemented with each memory bit corresponding to a dot on the CRT screen. A non-encoded keyboard could also be used with the software expanded to decode the keyboard. Additional ICs could be added enabling the MPU and CRTC to run on different phases so that the MPU has transparent access to the display memory. The software, developed in this article, may be used as is or used as a building block to implement additional features.

```
PAGE  001  CRTC     .SA:1  CRTC

                                 NAM    CRTC
                       ***********************************************
                       *      HARDWARE CONFIGURATION
                       *         ACIA                 $FCF4
                       *         ROM                  $E000
                       *         RAM                  $A000
                       *         CRTC                 $4000
                       *         SCREEN MEMORY        $0000
                       ***********************************************
                       *
                       *      SET UP PERIPHERAL ADDRESSES
         FCF4   A ACIACS EQU    $FCF4      ACIA CONTROL/STATUS REG
         FCF5   A ACIADA EQU    ACIACS+1   ACIA DATA REGISTER
         3000   A CRTCAD EQU    $3000      CRTC ADDRESS REGISTER
         3001   A CRTCRG EQU    CRTCAD+1   CRTC DATA REGISTER
                       *
                       *      SET CONSTANTS
         4000   A SCRNST EQU    $4000      SCREEN STARTING ADDRESS
         47D0   A SCRNND EQU    SCRNST+2000 SCREEN END ADDRESS
         0040   A MOVE   EQU    $40        SCREEN OFFSET
         0004   A PAGESZ EQU    $04        CHARACTERS PER PAGE
         00FC   A PGMASK EQU    $FC        MASK TO GET CURRENT PAGE
         0002   A SCRNH  EQU    $02        CHARACTERS ON SCREEN
         00AB   A SCRNL  EQU    $AB
                       *
                       *      DEFINE VARIABLE LACATIONS
                       *
         A000   A RAM    EQU    $A000      RAM STARTING ADDRESS
         A001   A CHARH  EQU    RAM+1
         A002   A CHARL  EQU    RAM+2      CHARACTER POINTER L
         A006   A BLANKH EQU    RAM+6
         A007   A BLANKL EQU    RAM+7      BLANK POINTER L
         A006   A BSPOSH EQU    BLANKH     BACK SPACE POSITION H
         A007   A BSPOSL EQU    BLANKL     BACK SPACE POSITION L
         A00A   A INDEX  EQU    RAM+10     HOME UP POINTER
         A00E   A COMPR  EQU    RAM+14     HOME END POINTER
         A011   A SPACES EQU    RAM+17     SPACE COUNTER
         A012   A STARTH EQU    RAM+18     DISPLAY START ADDRESS H
         A013   A STARTL EQU    RAM+19     DISPLAY START ADDRESS L
         A014   A ENDH   EQU    RAM+20     END OF SCREEN
         A015   A ENDL   EQU    RAM+21     END OF SCREEN
         A016   A CHARLN EQU    RAM+22     CHARACTERS PER LINE
A E000                   ORG    $E000      STARTING ROM ADDRESS
                       ***********************************************
                       *         MONITOR PROGRAM
                       *          INITIALIZES THE STACK POINTER
                       *          INITIALIZES THE SELF-MODIFYING CODE
                       *          INITIALIZES THE DISPLAY MEMORY
                       *          INITIALIZES THE CRTC
                       *          ACCEPTS INPUT CHARACTERS
                       ***********************************************
A E000 8E A07F  A        LDS    #$A07F     INITIALIZE STACK POINTER
                       *
                       *    INITIALIZE THE SELF-MODIFYING CODE IN RAM
                       *
A E003 4F                CLRA              ZERO A ACCUMULATOR
A E004 B7 A001  A        STAA   CHARH
A E007 B7 A002  A        STAA   CHARL
```

FIGURE 24 — Complete Listing of CRTC Software

```
PAGE  002  CRTC     .SA:1  CRTC

A E00A B7 A006  A          STAA   BLANKH    ZERO BLANKH/BSPOSH POINTER
A E00D B7 A007  A          STAA   BLANKL    ZERO BLANKL/BSPOSL POINTER
A E010 B7 A00A  A          STAA   INDEX
A E013 B7 A00B  A          STAA   INDEX+1
A E016 B7 A00E  A          STAA   COMPR
A E019 B7 A00F  A          STAA   COMPR+1
A E01C B7 A011  A          STAA   SPACES
A E01F B7 A012  A          STAA   STARTH
A E022 B7 A013  A          STAA   STARTL
A E025 B7 A014  A          STAA   ENDH
A E028 B7 A015  A          STAA   ENDL
A E02B 86 B7    A          LDAA   #$B7      STORE "STA A" OP CODE
A E02D B7 A000  A          STAA   RAM
A E030 B7 A005  A          STAA   RAM+5
A E033 86 86    A          LDAA   #$86      STORE "LDA A" OP CODE
A E035 B7 A003  A          STAA   RAM+3
A E038 86 20    A          LDAA   #$20      STORE ASCII "BLANK"
A E03A B7 A004  A          STAA   RAM+4
A E03D 86 39    A          LDAA   #$39      STORE "RTS" OP CODE
A E03F B7 A008  A          STAA   RAM+8
A E042 B7 A00C  A          STAA   RAM+12
A E045 B7 A010  A          STAA   RAM+16
A E048 86 CE    A          LDAA   #$CE      STORE "LDX" OP CODE
A E04A B7 A009  A          STAA   RAM+9
A E04D 86 8C    A          LDAA   #$8C      STORE "CPX" OP CODE
A E04F B7 A00D  A          STAA   RAM+13
A E052 86 26    A          LDAA   #$26      SET NO. CHAR PER LINE
A E054 B7 A016  A          STAA   RAM+22
A E057 8D 06 E05F          BSR    BLANKR    FILL SCREEN WITH BLANKS
A E059 8D 12 E06D          BSR    CRTINT    INITIALIZE CRTC
A E05B 8D 32 E08F RUN      BSR    CHARRC    RUN PROGRAM
A E05D 20 FC E05B          BRA    RUN
                  *****************************************************
                  *    BLANKR SUBROUTINE FILLS DISPLAY MEMORY WITH
                  *    BLANK CODE ($20).
                  *****************************************************
A E05F 86 20    A BLANKR   LDAA   #$20      INITIALIZE SCREEN MEMORY
A E061 CE 4000  A          LDX    #SCRNST   DISPLAY START ADDRESS
A E064 A7 00    A BLANK1   STAA   0,X       STORE CHARACTER
A E066 08                  INX              NEXT SCREEN LOCATION
A E067 8C 47D0  A          CPX    #SCRNND   FINISHED?
A E06A 26 F8 E064          BNE    BLANK1
A E06C 39                  RTS
                  *****************************************************
                  * CRINT SUBROUTINE INITIALIZES CRTC BY LOADING
                  * THE BURRIED RIGISTERS.
                  *****************************************************
A E06D 5F         CRTINT   CLRB             INITIALIZE CRTC
A E06E CE E07F  A          LDX    #TABLE
A E071 F7 3000  A CRT      STAB   CRTCAD    SELECT CRTC REGISTER
A E074 A6 00    A          LDAA   0,X       GET TABLE VALUE
A E076 B7 3001  A          STAA   CRTCRG    STORE CRTC PARAMETER
A E079 08                  INX              GET NEXT TABLE VALUE
A E07A 5C                  INCB             SELECT NEXT CRTC REGISTER
A E07B C1 10    A          CMPB   #$10      LAST CRTC REGISTER
A E07D 26 F2 E071          BNE    CRT
                  *
                  * TABLE OF VAU
```

FIGURE 24 — Complete Listing of CRTC Software
(Continued)

```
PAGE  003  CRTC     .SA:1  CRTC

00117                    *
00118A E07F     30     A TABLE  FCB     $30        R0  HORIZONTAL TOTLA
00119A E080     26     A        FCB     $26        R1  HORIZONTAL DISPLAYED
00120A E081     2B     A        FCB     $2B        R2  HORIZONTAL SYNC POS.
00121A E082     02     A        FCB     $02        R3  HORIZONTAL SYNC WIDTH
00122A E083     14     A        FCB     $14        R4  VERTICAL TOTAL
00123A E084     01     A        FCB     $01        R5  VERTICAL TOTAL ADJUST
00124A E085     12     A        FCB     $12        R6  VERTICAL DISPLAYED
00125A E086     13     A        FCB     $13        R7  VERTICAL SYNC POSITION
00126A E087     00     A        FCB     $00        R8  INTERLACE MODE
00127A E088     0B     A        FCB     $0B        R9  MAX SCAN LINE ADDRESS
00128A E089     40     A        FCB     $40        R10 CURSOR START ADDRESS
00129A E08A     08     A        FCB     $08        R11 CURSOR END ADDRESS
00130A E08B     00     A        FCB     $00        R12 START ADDRESS H
00131A E08C     00     A        FCB     $00        R13 START ADDRESS L
00132A E08D     00     A        FCB     $00        R14 START ADDRESS H
00133A E08E     00     A        FCB     $00        R15 START ADDRESS L
00134                    ************************************************
00135                    *   CHARRC SUBROUTINE ACCEPTS KEYBOARD INPUT,DECO
00136                    *   SPECIAL FEATURES AND CONTROLS THE CURSOR.
00137                    ************************************************
00138A E08F 8D 7F E110 CHARRC BSR     INCH       GET INPUT
00139A E091 81 13    A          CMPA    #$13       DECODE SPECIAL CHARACTERS
00140A E093 23 02 E097          BLS     DECODE
00141A E095 20 31 E0C8          BRA     CURSE      NOT A SPECIAL CHARACTER
00142A E097 81 0D    A DECODE CMPA    #$0D
00143A E099 26 03 E09E          BNE     DEC1
00144A E09B 7E E177  A          JMP     CR         CARRIAGE RETURN?
00145A E09E 81 08    A DEC1   CMPA    #$08
00146A E0A0 26 03 E0A5          BNE     DEC2
00147A E0A2 7E E1AF  A          JMP     BS         BACKSPACE?
00148A E0A5 81 0A    A DEC2   CMPA    #$0A
00149A E0A7 26 03 E0AC          BNE     DEC3
00150A E0A9 7E E191  A          JMP     LF         LINEFED?
00151A E0AC 81 13    A DEC3   CMPA    #$13
00152A E0AE 26 03 E0B3          BNE     DEC4
00153A E0B0 7E E1EF  A          JMP     UPLINE     MOVE CURSOR UP ONE LINE?
00154A E0B3 81 04    A DEC4   CMPA    #$04
00155A E0B5 26 03 E0BA          BNE     DEC5
00156A E0B7 7E E20C  A          JMP     PAGE       NEXT PAGE?
00157A E0BA 81 01    A DEC5   CMPA    #01
00158A E0BC 26 03 E0C1          BNE     DEC6
00159A E0BE 7E E22A  A          JMP     HOME       HOME CURSOR
00160A E0C1 81 07    A DEC6   CMPA    #07
00161A E0C3 26 03 E0C8          BNE     CURSE
00162A E0C5 7E E258  A          JMP     CLEAR      CLEAR SCREEN?
00163A E0C8 C6 0F    A CURSE  LDAB    #$0F       GET CURSOR ADDRESS L
00164A E0CA F7 3000  A          STAB    CRTCAD
00165A E0CD F6 3001  A          LDAB    CRTCRG
00166A E0D0 F7 A002  A          STAB    CHARL      SAVE CHARACTER ADDRESS
00167A E0D3 5C                  INCB
00168A E0D4 F7 3001  A          STAB    CRTCRG
00169A E0D7 F7 A007  A          STAB    BLANKL     SAVE CURSOR ADDRESS FOR BL
00170A E0DA C6 0E    A          LDAB    #$0E       GET CURSOR ADDRESS H
00171A E0DC F7 3000  A          STAB    CRTCAD
00172A E0DF F6 3001  A          LDAB    CRTCRG
00173A E0E2 CA 40    A          ORAB    #MOVE      MOVE CURSOR TO DISPLAY ADD
00174A E0E4 F7 A001  A          STAB    CHARH      SAVE CHARACTER ADDRESS
```

FIGURE 24 — Complete Listing of CRTC Software
(Continued)

```
PAGE 004  CRTC    .SA:1  CRTC

A E0E7 F6 A007  A          LDAB    BLANKL    BLANKL=0?
A E0EA 26 06 E0F2          BNE     NOCARY
A E0EC F6 A001  A          LDAB    CHARH     INCREMENT IF CARRY REQUIRE
A E0EF 5C                  INCB
A E0F0 20 03 E0F5          BRA     CARRYD
A E0F2 F6 A001  A NOCARY   LDAB    CHARH     INCREMENT IF CARRY REQUIRE
A E0F5 F7 3001  A CARRYD   STAB    CRTCRG    UPDATE CURSOR
A E0F8 F7 A006  A          STAB    BLANKH    BLNAK UNDER CURSOR
                  *
                  *       RAM IS A SECTION OF SELF-MODIFYING CODE WHI
                  *       STORES THE CHARACTER, IN THE A REGISTER, AT
                  *       THE PRESENT CURSOR LOCATION.
                  ***********************************************
A E0FB BD A000  A          JSR     RAM       SAVE CHARACTER
A E0FE 7C A011  A          INC     SPACES    INCREMENT SPACE COUNTER
A E101 F6 A016  A          LDAB    CHARLN    AUTOMATIC CR?
A E104 F1 A011  A          CMPB    SPACES
A E107 2E 06 E10F          BGT     NOSCRL
A E109 7F A011  A          CLR     SPACES
A E10C 7E E120  A SCRLOL   JMP     SCROLU    CHECH FOR SCROLL UP
A E10F 39         NOSCRL   RTS
                  ***********************************************
                  *      INCH SUBROUTINE POLLS THE ACIA UNTIL A CHARA
                  *      IS RECEIVED THEN MASKS THE PARITY BIT AND
                  *      IGNORS RUBOUTS.
                  ***********************************************
A E110 B6 FCF4  A INCH     LDAA    ACIACS
A E113 47                  ASRA              READY?
A E114 24 FA E110          BCC     INCH      RECEIVED NOT READY
A E116 B6 FCF5  A          LDAA    ACIADA    INPUT CHARACTER
A E119 84 7F    A          ANDA    #$7F      RESET PARITY BIT
A E11B 81 7F    A          CMPA    #$7F
A E11D 27 F1 E110          BEQ     INCH      RUBOUT IGNOR
A E11F 39                  RTS
                  ***********************************************
                  *      SCROLU SUBROUTINE CHECKS TO SEE IT THE CURSO
                  *      MOVED OFF THE BOTTOM OF THE SCREEN. IF SO A
                  *      NEW LINE IS SCROLLED ONTO THE SCREEN.
                  ***********************************************
A E120 B6 A013  A SCROLU   LDAA    STARTL    SET UP END OF SCREEN ADDRE
A E123 9B AB    A          ADDA    SCRNL
A E125 B7 A015  A          STAA    ENDL
A E128 24 04 E12E          BCC     FIND
A E12A 86 01    A          LDAA    #01
A E12C 20 01 E12F          BRA     FIND1
A E12E 4F         FIND     CLRA
A E12F BB A012  A FIND1    ADDA    STARTH
A E132 9B 02    A          ADDA    SCRNH
A E134 B7 A014  A          STAA    ENDH
A E137 C6 0E    A          LDAB    #$0E      GET CURSOR ADDRESS H
A E139 F7 3000  A          STAB    CRTCAD
A E13C F6 3001  A          LDAB    CRTCRG
A E13F 11                  CBA
A E140 22 10 E152          BHI     EQUAL1
A E142 B6 A015  A          LDAA    ENDL      CHECK LOW ADDRESS
A E145 C6 0F    A          LDAB    #$0F      GET CURSOR ADDRESS L
A E147 F7 3000  A          STAB    CRTCAD
A E14A F6 3001  A          LDAB    CRTCRG
```

FIGURE 24 — Complete Listing of CRTC Software
(Continued)

```
PAGE  005  CRTC    .SA:1  CRTC

00233A E14D 11                   CBA
00234A E14E 27 02 E152           BEQ    EQUAL1
00235A E150 23 01 E153           BLS    CHANGE
00236A E152 39           EQUAL1 RTS
00237A E153 86 0D      A CHANGE LDAA   #$0D       INCREMENT START ADDRESS
00238A E155 B7 3000  A           STAA   CRTCAD
00239A E158 F6 A013  A           LDAB   STARTL
00240A E15B FB A016  A           ADDB   CHARLN     SCROLL UP ONE LINE
00241A E15E F7 3001  A           STAB   CRTCRG
00242A E161 F7 A013  A           STAB   STARTL
00243A E164 25 01 E167           BCS    CARRY      CARRY?
00244A E166 39                   RTS
00245A E167 C6 0C      A CARRY  LDAB   #$0C       INCREMENT START ADDRESS H
00246A E169 F7 3000  A           STAB   CRTCAD
00247A E16C F6 A012  A           LDAB   STARTH
00248A E16F 5C                   INCB
00249A E170 F7 3001  A           STAB   CRTCRG
00250A E173 F7 A012  A           STAB   STARTH
00251A E176 39                   RTS               CHECK TO SEE IF TI IS OK
00252                  ***********************************************************
00253                  *      CR SUBROUTINE SUBTRACTS SPACE COUNTER FROM
00254                  *      CURSOR POSITION TO GENERATE A CARRIAGE RETU
00255                  *      AND THEN CALLS LINEFD.
00256                  ***********************************************************
00257A E177 86 0F      A CR     LDAA   #$0F       GET CURSOR ADDRESS L
00258A E179 B7 3000  A           STAA   CRTCAD
00259A E17C F6 3001  A           LDAB   CRTCRG
00260A E17F F0 A011  A           SUBB   SPACES     GENERATE CR
00261A E182 F7 3001  A           STAB   CRTCRG
00262A E185 24 07 E18E           BCC    YES        NO CARRY?
00263A E187 4A                   DECA              ELSE DECREMENT CURSOR H
00264A E188 B7 3000  A           STAA   CRTCAD
00265A E18B 7A 3001  A           DEC    CRTCRG
00266A E18E 7F A011  A YES      CLR    SPACES     INITIALIZE SPACE COUNTER
00267                  ***********************************************************
00268                  *   LINEFD SUBRFOUTINE MOVES THE CURSOR DOWN ONE L
00269                  *   BY ADDING THE NUMBER OF CHARACTERS.LINE,CHRPLN
00270                  *   CURRENT CURSOR LOCATION.  SCROLU IS THEN CALLE
00271                  ***********************************************************
00272A E191 86 0F      A LF     LDAA   #$0F       GET CURSOR ADDRESS L
00273A E193 B7 3000  A           STAA   CRTCAD
00274A E196 F6 3001  A           LDAB   CRTCRG
00275A E199 FB A016  A           ADDB   CHARLN     GENERATE LINE FEED
00276A E19C 24 0B E1A9           BCC    NCARRY     CARRY?
00277A E19E F7 3001  A           STAB   CRTCRG
00278A E1A1 4A                   DECA
00279A E1A2 B7 3000  A           STAA   CRTCAD
00280A E1A5 F6 3001  A           LDAB   CRTCRG
00281A E1A8 5C                   INCB
00282A E1A9 F7 3001  A NCARRY   STAB   CRTCRG
00283A E1AC 7E E120  A           JMP    SCROLU
00284                  ***********************************************************
00285                  *   BS SUBROUTINE MOVES CURSOR BACK ONE LINE IF TH
00286                  *   CURSOR MOVES TO THE PREVIOUS LINE THEN SCROLD
00287                  *   IS CALLED TO SEE IF A NEW LINE SHOULD BE ADDED
00288                  *   AT THE TOP OF THE SCREEN.
00289                  ***********************************************************
00290A E1AF 86 0F      A BS     LDAA   #$0F       GET CURSOR ADDRESS L
```

FIGURE 24 — Complete Listing of CRTC Software
(Continued)

```
PAGE  006  CRTC     .SA:1  CRTC

00291A E1B1 B7 3000   A          STAA   CRTCAD
00292A E1B4 F6 3001   A          LDAB   CRTCRG
00293A E1B7 5A                   DECB          BACK UP CURSOR
00294A E1B8 F7 3001   A          STAB   CRTCRG
00295A E1BB 4A                   DECA          SELECT CURSOR H
00296A E1BC B7 3000   A          STAA   CRTCAD
00297A E1BF F7 A007   A          STAB   BSPOSL SAVE BACK SPACE POSITION L
00298A E1C2 C1 FF     A          CMPB   #$FF   CARRY?
00299A E1C4 27 05 E1CB           BEQ    DECR
00300A E1C6 F6 3001   A          LDAB   CRTCRG
00301A E1C9 20 07 E1D2           BRA    NODECR
00302A E1CB F6 3001   A DECR     LDAB   CRTCRG IF SO DECREMENT CURSOR H
00303A E1CE 5A                   DECB
00304A E1CF F7 3001   A          STAB   CRTCRG
00305A E1D2 CA 40     A NODECR   ORAB   #MOVE  MOVE TO SCREEN MEMORY
00306A E1D4 F7 A006   A          STAB   BSPOSH SAVE BACK SPACE POSITION H
00307A E1D7 BD A003   A          JSR    RAM+3  BLANK UNDER CURSOR
00308A E1DA 7A A011   A          DEC    SPACES DECREMENT SPACE COUNTER
00309A E1DD B6 A011   A          LDAA   SPACES BACK TO PREVIOUS LINE?
00310A E1E0 81 FF     A          CMPA   #$FF
00311A E1E2 27 01 E1E5           BEQ    CALLER
00312A E1E4 39                   RTS
00313A E1E5 B6 A016   A CALLER   LDAA   CHARLN RESET SPACE COUNTER
00314A E1E8 4A                   DECA
00315A E1E9 B7 A011   A          STAA   SPACES
00316A E1EC 7E E284   A          JMP    SCROLD
00317                   ***********************************************
00318                   *    UPLINE SUBROUTINE MOVES THE CURSOR UP ONE
00319                   *    LINE THEN CALLS SCROLD.
00320                   ***********************************************
00321A E1EF 86 0F     A UPLINE   LDAA   #$0F   GET CURSOR ADDRESS L
00322A E1F1 B7 3000   A          STAA   CRTCAD
00323A E1F4 F6 3001   A          LDAB   CRTCRG
00324A E1F7 F0 A016   A          SUBB   CHARLN GENERATE UPLINE
00325A E1FA 24 0B E207           BCC    NOOCRY CARRY?
00326A E1FC F7 3001   A          STAB   CRTCRG
00327A E1FF 4A                   DECA          GET CURSOR H
00328A E200 B7 3000   A          STAA   CRTCAD
00329A E203 F6 3001   A          LDAB   CRTCRG SUBTRACT CARRY
00330A E206 5A                   DECB
00331A E207 F7 3001   A NOOCRY   STAB   CRTCRG
00332A E20A 20 78 E284           BRA    SCROLD
00333                   ***********************************************
00334                   *     PAGE SINE MOVE THE CURSOR TO THE NEXT PAGE.
00335                   ***********************************************
00336A E20C 86 0C     A PAGE     LDAA   #$0C   GET SCREEN START ADDRESS H
00337A E20E B7 3000   A          STAA   CRTCAD
00338A E211 F6 A012   A          LDAB   STARTH
00339A E214 DB 04     A          ADDB   PAGESZ MOVE TO NEXT PAGE
00340A E216 F7 3001   A          STAB   CRTCRG
00341A E219 F7 A012   A          STAB   STARTH
00342A E21C 86 0E     A          LDAA   #$0E   GET CURSOR ADDRESS H
00343A E21E B7 3000   A          STAA   CRTCAD
00344A E221 F6 3001   A          LDAB   CRTCRG
00345A E224 DB 04     A          ADDB   PAGESZ MOVE CURSOR TO NEXT PAGE
00346A E226 F7 3001   A          STAB   CRTCRG
00347A E229 39                   RTS
00348                   ***********************************************
```

FIGURE 24 — Complete Listing of CRTC Software
(Continued)

```
PAGE  007  CRTC     .SA:1  CRTC

00349                      *   HOME SUBROUTINE MOVES THE CURSOR TO THE BEGIN
00350                      *   OF THE PRESENT PAGE AND CALLS CLEAR.
00351                      *************************************************
00352A E22A 86 0E     A HOME    LDAA   #$0E       GET CURSOR ADDRESS H
00353A E22C B7 3000   A         STAA   CRTCAD
00354A E22F F6 3001   A         LDAB   CRTCRG
00355A E232 D4 FC     A         ANDB   PGMASK     GET PAGE ADDRESS
00356A E234 F7 3001   A         STAB   CRTCRG     MOVE CURSOR
00357A E237 F7 A012   A         STAB   STARTH     START AT FIRST OF PAGE
00358A E23A 86 0C     A         LDAA   #$0C
00359A E23C B7 3000   A         STAA   CRTCAD
00360A E23F F7 3001   A         STAB   CRTCRG
00361A E242 4C                  INCA              SELECT CURSOR L
00362A E243 B7 3000   A         STAA   CRTCAD
00363A E246 4F                  CLRA
00364A E247 B7 3001   A         STAA   CRTCRG
00365A E24A B7 A013   A         STAA   STARTL     START AT FIRST OF PAGE
00366A E24D C6 0F     A         LDAB   #$0F
00367A E24F F7 3000   A         STAB   CRTCAD
00368A E252 B7 3001   A         STAA   CRTCRG
00369A E255 B7 A011   A         STAA   SPACES     ZERO SPACE COUNTER
00370                      *************************************************
00371                      *    CLEAR SUBROUTINE CLEARS PRESENT PAGE PAST TH
00372                      *    CURSOR BY STORING ASCII BLANDS ($20) INTO
00373                      *    SCREEN MEMORY.
00374                      *************************************************
00375A E258 86 0E     A CLEAR   LDAA   #$0E       GET CURSOR ADDRESS H
00376A E25A B7 3000   A         STAA   CRTCAD
00377A E25D F6 3001   A         LDAB   CRTCRG
00378A E260 D4 FC     A         ANDB   PGMASK     LOCATE CURSOR PAGE ADDRESS
00379A E262 CB 40     A         ADDB   #MOVE      ADD OFFSET
00380A E264 F7 A00A   A         STAB   INDEX      SAVE START ADDRESS
00381A E267 DB 04     A         ADDB   PAGESZ     SAVE END ADDRESS
00382A E269 F7 A00E   A         STAB   COMPR
00383A E26C 4C                  INCA              SET UP LOW ADDRESS
00384A E26D B7 3000   A         STAA   CRTCAD
00385A E270 F6 3001   A         LDAB   CRTCRG
00386A E273 F7 A00B   A         STAB   INDEX+1
00387A E276 BD A009   A         JSR    RAM+9      INDEX REGISTER PAGE ADDRES
00388A E279 86 20     A BLANK   LDAA   #$20       ASCII BLANK
00389A E27B A7 00     A         STAA   0,X        STORE BLANK
00390A E27D 08                  INX               NEXT SCREEN CHARACTER
00391A E27E BD A00D   A         JSR    RAM+13     CHECK INDEX REGISTER
00392A E281 26 F6 E279          BNE    BLANK
00393A E283 39                  RTS
00394                      *************************************************
00395                      *   SCROLD SUBROUTINE CHECKS TO SEE IF THE CURSOR
00396                      *   MOVED OFF THE TOP OF THE SCREEN.  IF SO A NEW
00397                      *   IS SCROLLED DOWN ONTO THE SCREEN.
00398                      *************************************************
00399A E284 B6 A012   A SCROLD  LDAA   STARTH     CURSOR BEFORE SCREEN?
00400A E287 C6 0E     A         LDAB   #$0E       GET CURSOR ADDRESS H
00401A E289 F7 3000   A         STAB   CRTCAD
00402A E28C F6 3001   A         LDAB   CRTCRG
00403A E28F 11                  CBA
00404A E290 22 12 E2A4          BHI    BEFORE
00405A E292 27 01 E295          BEQ    EQUAL2
00406A E294 39                  RTS               HIGH ADDRESS DOESN'T MATCH
```

FIGURE 24 — Complete Listing of CRTC Software
(Continued)

Motorola reserves the right to make changes to any products herein to improve reliability, function or design. Motorola does not assume any liability arising out of the application or use of any product or circuit described herein; neither does it convey any license under its patent rights nor the rights of others.

MOTOROLA *Semiconductor Products Inc.*

3501 ED BLUESTEIN BLVD., AUSTIN, TEXAS 78721 • A SUBSIDIARY OF MOTOROLA INC.

A14955 PRINTED IN USA 12/81 IMPERIAL LITHO C01778 18,000 AN851

```
PAGE  008  CRTC     .SA:1  CRTC

00407A E295 B6 A013   A EQUAL2 LDAA    STARTL     IS CURSOR BEFORE THE SCREE
00408A E298 C6 0F     A        LDAB    #$0F       GET CURSOR ADDRESS LOW
00409A E29A F7 3000   A        STAB    CRTCAD
00410A E29D F6 3001   A        LDAB    CRTCRG
00411A E2A0 11                 CBA
00412A E2A1 22 01 E2A4         BHI     BEFORE
00413A E2A3 39          EXIT   RTS
00414A E2A4 86 0D     A BEFORE LDAA    #$0D       MOVE BACK ONE LINE
00415A E2A6 B7 3000   A        STAA    CRTCAD
00416A E2A9 F6 A013   A        LDAB    STARTL
00417A E2AC F0 A016   A        SUBB    CHARLN
00418A E2AF F7 3001   A        STAB    CRTCRG
00419A E2B2 F7 A013   A        STAB    STARTL
00420A E2B5 25 01 E2B8         BCS     CRYSET     CARRY SET?
00421A E2B7 39                 RTS
00422A E2B8 4A          CRYSET DECA               IF SO DECREMENT STARTH
00423A E2B9 B7 3000   A        STAA    CRTCAD
00424A E2BC F6 A012   A        LDAB    STARTH
00425A E2BF 5A                 DECB
00426A E2C0 F7 3001   A        STAB    CRTCRG
00427A E2C3 F7 A012   A        STAB    STARTH
00428A E2C6 39                 RTS
00429                          END
TOTAL ERRORS 00000--00000


    FCF4 ACIACS 00012*00013 00201
    FCF5 ACIADA 00013*00204
    E2A4 BEFORE 00404 00412 00414*
    E279 BLANK  00388*00392
    E064 BLANK1 00097*00100
    A006 BLANKH 00031*00033 00059 00182
    A007 BLANKL 00032*00034 00060 00169 00175
    E05F BLANKR 00087 00095*
    E1AF BS     00147 00290*
    A006 BSPOSH 00033*00306
    A007 BSPOSL 00034*00297
    E1E5 CALLER 00311 00313*
    E167 CARRY  00243 00245*
    E0F5 CARRYD 00179 00181*
    E153 CHANGE 00235 00237*
    A001 CHARH  00029*00057 00174 00177 00180
    A002 CHARL  00030*00058 00166
    A016 CHARLN 00042*00190 00240 00275 00313 00324 00417
    E08F CHARRC 00089 00138*
    E258 CLEAR  00162 00375*
    A00E COMPR  00036*00063 00064 00382
    E177 CR     00144 00257*
    E071 CRT    00108*00114
    3000 CRTCAD 00014*00015 00108 00164 00171 00225 00231 00238 00246
                00258 00264 00273 00279 00291 00296 00322 00328 00337
                00343 00353 00359 00362 00367 00376 00384 00401 00409
                00415 00423
    3001 CRTCRG 00015*00110 00165 00168 00172 00181 00226 00232 00241
                00249 00259 00261 00265 00274 00277 00280 00282 00292
                00294 00300 00302 00304 00323 00326 00329 00331 00340
                00344 00346 00354 00356 00360 00364 00368 00377 00385
```

FIGURE 24 — Complete Listing of CRTC Software
(Continued)

```
PAGE  009  CRTC     .SA:1  CRTC
                 00402 00410 00418 00426
   E06D CRTINT   00088 00106*
   E2B8 CRYSET   00420 00422*
   E0C8 CURSE    00141 00161 00163*
   E09E DEC1     00143 00145*
   E0A5 DEC2     00146 00148*
   E0AC DEC3     00149 00151*
   E0B3 DEC4     00152 00154*
   E0BA DEC5     00155 00157*
   E0C1 DEC6     00158 00160*
   E097 DECODE   00140 00142*
   E1CB DECR     00299 00302*
   A014 ENDH     00040*00068 00223
   A015 ENDL     00041*00069 00216 00229
   E152 EQUAL1   00228 00234 00236*
   E295 EQUAL2   00405 00407*
   E2A3 EXIT     00413*
   E12E FIND     00217 00220*
   E12F FIND1    00219 00221*
   E22A HOME     00159 00352*
   E110 INCH     00138 00201*00203 00207
   A00A INDEX    00035*00061 00062 00380 00386
   E191 LF       00150 00272*
   0040 MOVE     00020*00173 00305 00379
   E1A9 NCARRY   00276 00282*
   E0F2 NOCARY   00176 00180*
   E1D2 NODECR   00301 00305*
   E207 NOOCRY   00325 00331*
   E10F NOSCRL   00192 00195*
   E20C PAGE     00156 00336*
   0004 PAGESZ   00021*00339 00345 00381
   00FC PGMASK   00022*00355 00378
   A000 RAM      00028*00029 00030 00031 00032 00035 00036 00037 00038
                 00039 00040 00041 00042 00071 00072 00074 00076 00078
                 00079 00080 00082 00084 00086 00188 00307 00387 00391
   E05B RUN      00089*00090
   E10C SCRLOL   00194*
   0002 SCRNH    00023*00222
   00AB SCRNL    00024*00215
   47D0 SCRNND   00019*00099
   4000 SCRNST   00018*00019 00096
   E284 SCROLD   00316 00332 00399*
   E120 SCROLU   00194 00214*00283
   A011 SPACES   00037*00065 00189 00191 00193 00260 00266 00308 00309
                 00315 00369
   A012 STARTH   00038*00066 00221 00247 00250 00338 00341 00357 00399
                 00424 00427
   A013 STARTL   00039*00067 00214 00239 00242 00365 00407 00416 00419
   E07F TABLE    00107 00118*
   E1EF UPLINE   00153 00321*
   E18E YES      00262 00266*
```

FIGURE 24 — Complete Listing of CRTC Software
(Continued)