Graphics/Alphanumerics Systems Using the DP8350

National Semiconductor
Application Note 243
Mike Evans
July 1986

This Application Note summarizes some CRT terminal circuits, each with an increasing degree of graphics capability, and then goes into detail to describe a system having full graphics capability, with all dots individually programmable. All these applications use the DP8350 CRT Controller.

Here are some of the features of the full graphics system.

Hardware Features

- The hardware is designed for a 24 row by 80 column display, with 7 dots per column and 10 lines per row
- All ICs are made by National Semiconductor
- Low I.C. cost, all parts readily available
- Fits on one standard BLC80 (SBC80) card
- System performance only limited by software
- 8080 Mnemonics—usable with STARPLEX™ or Intellec® Development Systems
- All graphics programs very fast
  Example: One dot takes 500 μs maximum to plot
- During display time, each 7-dot cycle may be shared by the microprocessor
- 8-bit word comprises MSB as attribute and next 7 bits as 7 dot word of a character line
- Can input display data serially or parallel
- Can output display data serially or parallel
- Baud rate programmable from 110 to 56k baud
- Can be used as slave to main system
- Can copy characters from alphanumeric ROM or symbol EPROM
- 13 kbytes of RAM available for user software or back-up display storage
- Analog inputs—joystick or waveforms
- Easily expandable to color graphics

Software Features

- The software is programmed for any display configuration of rows, columns, dots per column and lines per row. The hardware is designed for a 24 row by 80 column display, with 7 dots per column and 10 lines per row
- Can perform most dumb terminal functions, including scrolling
- Simultaneous display of alphanumerics and graphics
- Identical terminals can display same information with inputs from either
- Can save displays in computer storage
- Can load displays from computer storage
- Can erase any part of display or all of it
- Can draw a rectangle linking any 2 horizontal and 2 vertical coordinates
- Can transfer in 1/10th second max any area of display to any other area or to/from backup display storage
- Smooth movement of subject in any direction
- Immediate display of fixed diagrams
- In-system emulation of programs available

The DP8350 CRT Controller provides incrementing video addresses starting from the Top of Page address, or from a new Row Start address. These addresses and the Cursor address are loaded into their respective registers from the address bus. All video control signals are provided by the 8350, so that apart from the crystal oscillator, no extra video circuitry is required.

The DP8350 has so far been considered to be useable only in dumb terminals, whereas in fact it is easy to adapt it to more complex terminals with full graphics capability. Following is a summary of the functions of the various combinations of alphanumerics/graphics displays beginning with a dumb terminal using a monitor with 24 x 80 characters.

Dumb Terminal

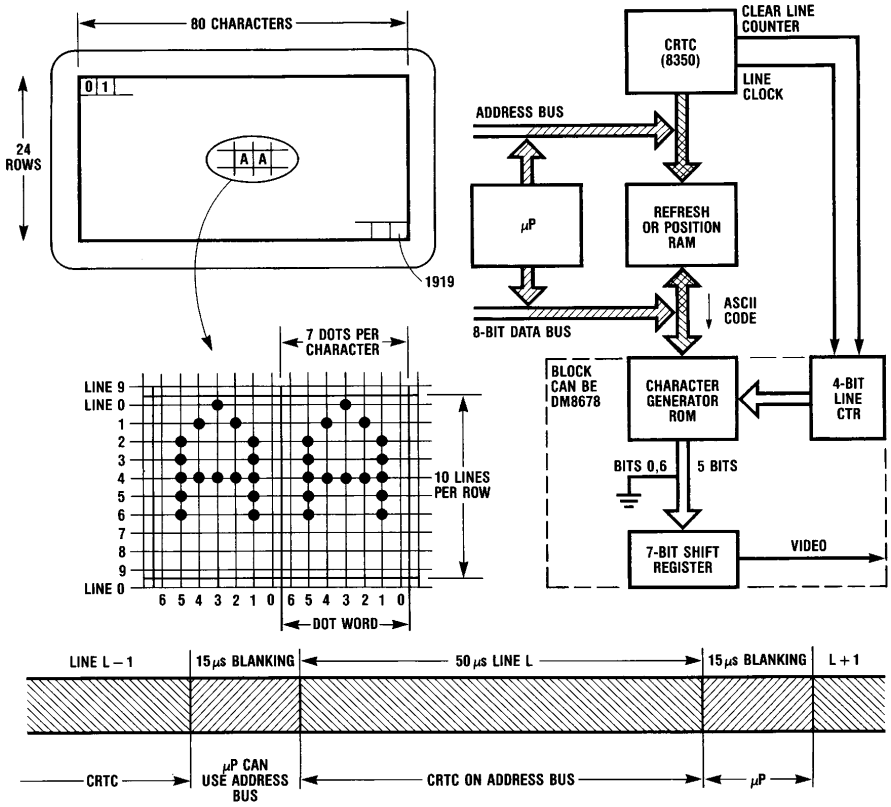
The basic dumb terminal design is shown in *Figure 1*. Usually the microprocessor loads the Character Position RAM (or Refresh RAM) only during horizontal or vertical blanking, or during the last 3 lines of a row. The CRTC then sequentially addresses this RAM during display time. The ASCII data from this RAM (for the character selected) is outputted to the ROM of the Character Generator. The 7-dot word of this character for the line being displayed is then loaded into a shift register, and shifted out as video to the monitor during the next 7-dot cycle.

The logical choice of CRT Controller for this simple CRT terminal is the DP8350. The most common application is for a 24 row by 80 column display with the character field comprised of 10 lines each of 7 dots. The character itself occupies 7 lines each of 5 dots, leaving 3 lines for vertical character spacing, and 2 dots for horizontal character spacing.

Refer to AN-198 and AN-199 for further information on alphanumeric applications of CRTs.

STARPLEX™ is a trademark of National Semiconductor Corp.
Intellec® is a registered trademark of Intel Corp.

 TL/F/5868 RRD-B30M105/Printed in U. S. A.



Disadvantages for Graphics TL/F/5868–1

- Only Characters in the Character Generator ROM can be selected.
- Characters not continuous to adjacent fields.
- Microprocessor thru-put 30% of maximum—not desirable for graphics.

FIGURE 1. Simplest CRT Terminal

Alphanumeric Characters with Extra Symbols

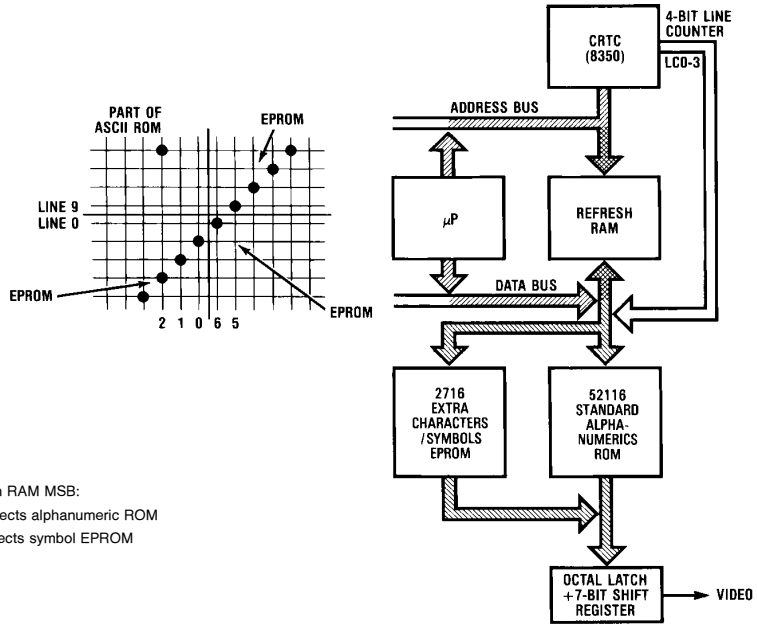
When characters or symbols are required that are different from those in the ROM, then an extra EPROM such as the 2716 can be added as shown in *Figure 2*. The standard characters can be selected from a separate ROM such as the 52116 which contains all 128 standard ASCII characters. The EPROM is preprogrammed with additional characters or symbols. The 8350 outputs sequential addresses to the Refresh RAM, and each address is two dot cycles ahead of the shifting dot word.

The data out from the RAM must be valid 150 ns after each address change. The MSB of the data selects ROM or EPROM, and the remaining 7 bits select the character. The line of the character is decoded from the 4-bit line counter outputs coming from the 8350. The ROM/EPROM now has 640–150 ns (>450 ns) to output the valid dot word. This has to be latched into an octal latch and held for one dot cycle before it can be loaded into the 7-bit shift register. The dots are then shifted out in the dot cycle.

Limited Graphics Terminal

To be able to generate any graphics symbol, a character RAM must replace the fixed ROM characters. Characters or symbols can be loaded into the RAM as required from a ROM or a pre-programmed EPROM like the 2716 (refer to *Figure 3*). But now, new graphics characters can be written into the RAM from the Microprocessor. These can either be derived internally from the μP or obtained directly from peripherals (such as serially to an Asynchronous Communications Element like the INS8250, or parallel from an external I/O port).

This limited graphics application thus requires two RAMs, the Refresh RAM (or Character Position RAM), and the Character RAM. The Refresh RAM outputs the selected character address, and the 8350 line counter outputs select the line in the Character RAM. The 7 dots outputted from this RAM are latched into the Octal Latch and held for one dot cycle. The 8th bit of data can be used as an attribute control bit. The 7 LSBs are then loaded into the 7-bit shift register.



Refresh RAM MSB:

0 Selects alphanumeric ROM

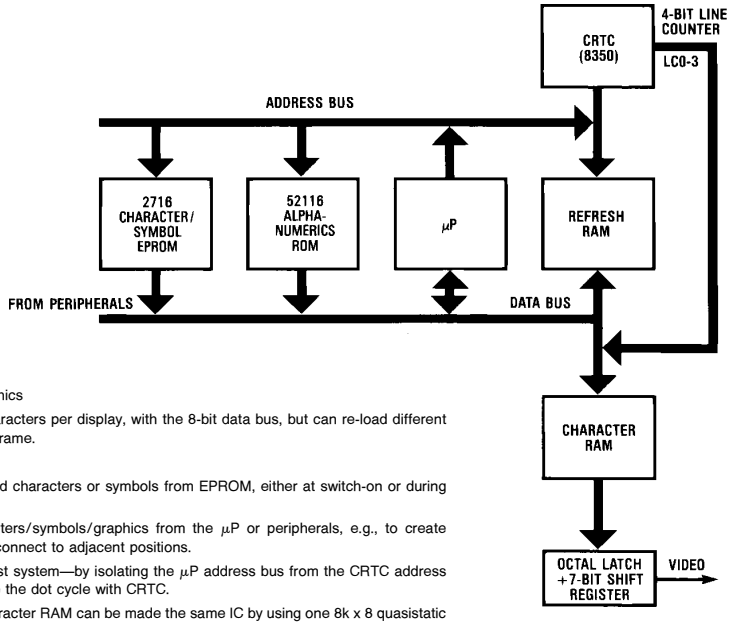
1 selects symbol EPROM

TL/F/5868–2

Disadvantages for Graphics

- Fixed graphics possible with continuous display, but limited to 128 different characters, and 128 standard alphanumerics, for all 1920 positions.
- Also it is possible to change characters/symbols once the EPROM has been programmed.
- The Microprocessor is still slow thru-put.

FIGURE 2. Fixed Character ROM + Symbol EPROM



Disadvantages for Graphics

- Only 256 possible characters per display, with the 8-bit data bus, but can re-load different characters for a new frame.

Advantages

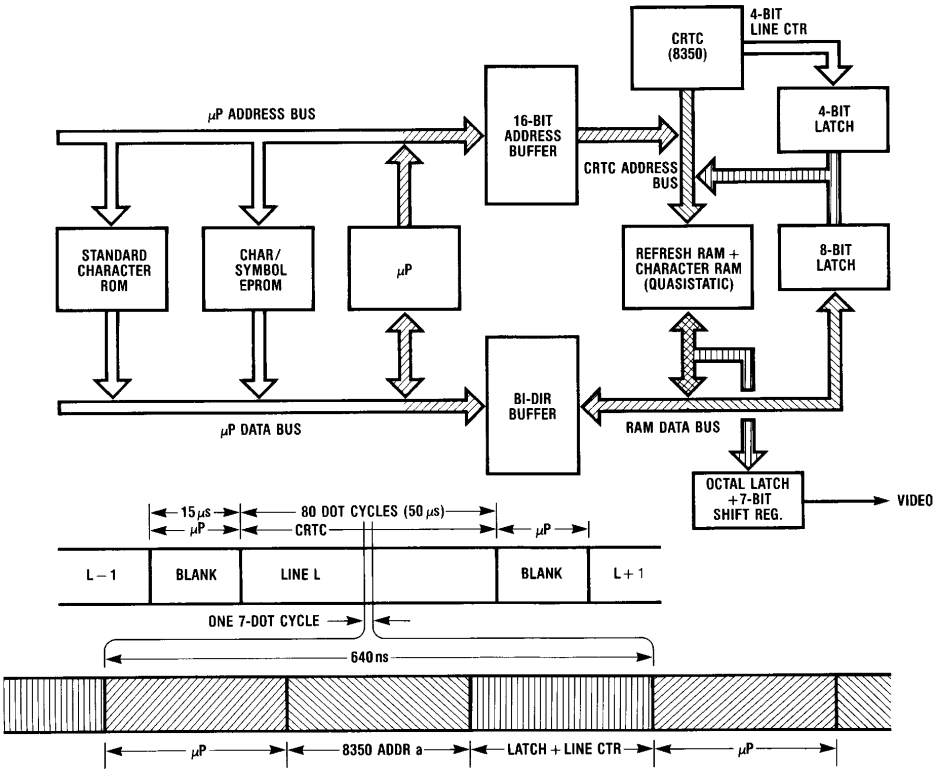
- Can now load standard characters or symbols from EPROM, either at switch-on or during normal running.
- Can also load characters/symbols/graphics from the μP or peripherals, e.g., to create graphics drawings to connect to adjacent positions.
- Can now be a very fast system—by isolating the μP address bus from the CRTC address bus, the μP can share the dot cycle with CRTC.
- Refresh RAM and character RAM can be made the same IC by using one 8k x 8 quasistatic RAM.

TL/F/5868–3

FIGURE 3. Character RAM with ROM/EPROM Look-Up

The first quarter of an 8k x 8 RAM can be used as a Refresh RAM for the 1920 character positions. The RAM data outputs containing the character address can then be latched into an octal TRI-STATE latch. If the 8350 address bus is then disabled, the octal TRI-STATE latch can feed back to the RAM second half address inputs, along with the enabled 8350 line counter outputs. The data out from the RAM now contains the next 7-dot word to be displayed and this is then loaded into the shift register. This takes the last two thirds of the dot cycle, the first third is for the μP. With the fast cycle time of the quasistatic RAMs this 3 part cycle can easily be accomplished in one 7-dot cycle. (Refer to *Figure 4*).

With the method just described it is only possible to display 256 different characters for any one page, because each character consists of 10 lines, almost filling the second half of the quasistatic RAM. If this is acceptable, then a limited graphics terminal can be easily implemented using a microprocessor, with one 2716 instruction set EPROM, one 52116 character ROM, one 2716 symbol EPROM, one DP8350 CRT Controller, the NMC4864 quasistatic RAM, and a DM74166 shift register. The logic and drive circuitry required to control the sequencing comprises a further 15 SSI ICs. This application has not yet been built, awaiting availability of the quasistatic RAMs.



TL/F/5868–4

Advantages

- Only one IC, an 8k x 8 quasistatic RAM, used for both the refresh RAM and character RAM.
- Fast, although μP may be in the wait state for a maximum of 600 ns. This is no problem because the fastest μP instruction cycle is 1 μs, so there will be no effect on maximum thru-put.

Disadvantages

- No quasistatics available at the time of writing.
- Full graphics capability not possible.

FIGURE 4. Limited Graphics Using a Buffered CRTC Address Bus and a Quasistatic RAM

Full Graphics Capability

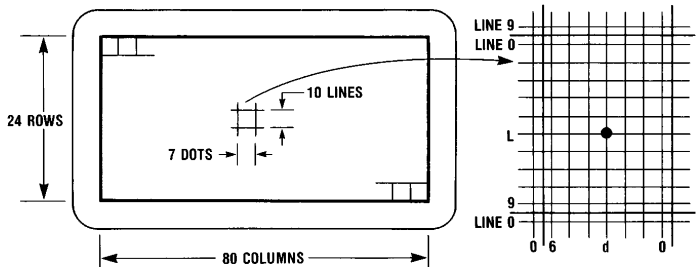
We need to be able to select any dot on the display, for full graphics capability, while still using the CRT controller to sequence every line of every row, as it does in the simple terminal (See *Figure 5*).

With the standard 24 x 80 character display, full graphics can be achieved by using a 24 (rows) by 80 (columns) by 10 (lines) address RAM, and selecting the 7 dots as the data word for the character position on the display and the line of that character position.

This means that alphanumeric characters can be displayed in exactly the same format as with a simple terminal, by copying the character from ROM or EPROM into the selected 10 line by 7-dot field, line by line.

Full graphics capability is also easily implemented once the relevant software algorithms have been determined.

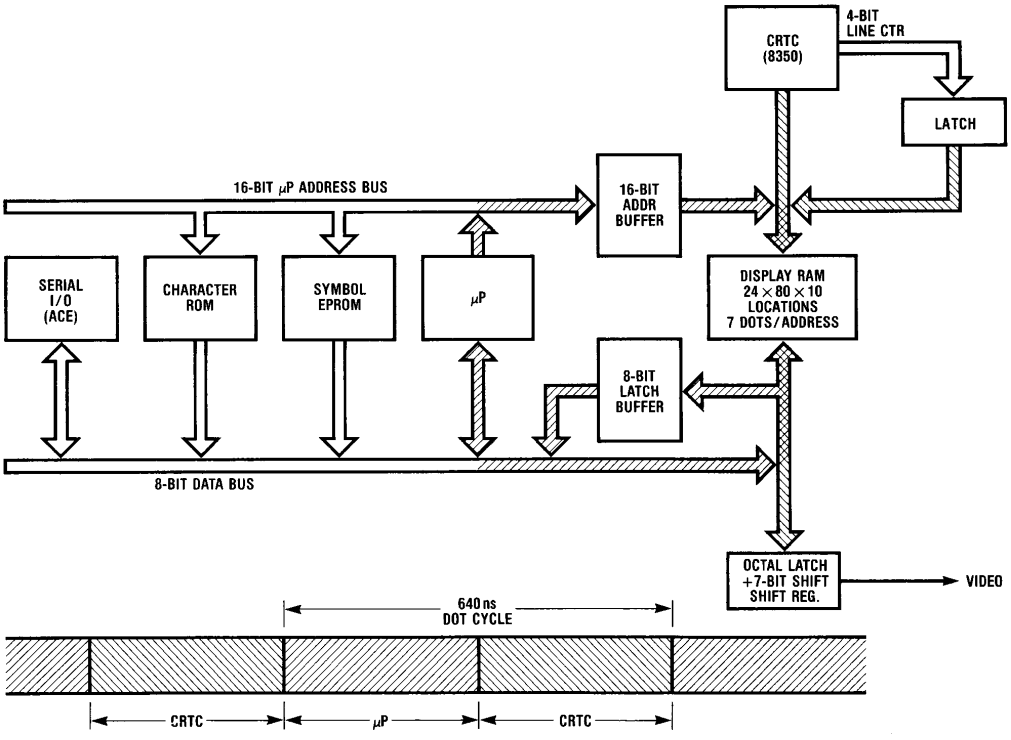
So for full graphics, every dot is one bit of memory. There is no refresh RAM, refer to *Figure 6*. The CRTC scans through the Display RAM, a line at a time for each row on the CRT, causing the RAM outputs to be read every 7-dot cycle. The RAM output is shifted out two dot cycles later. The microprocessor may write into the Display RAM each 7-dot word, with ''1's'' representing dots.



TL/F/5868–5

Dot is at Line L, Dot d character position is at Row r, Column c.

FIGURE 5. Full Graphics Capability Requires Individual Dot Selection



TL/F/5868–6

FIGURE 6. Full Graphics System

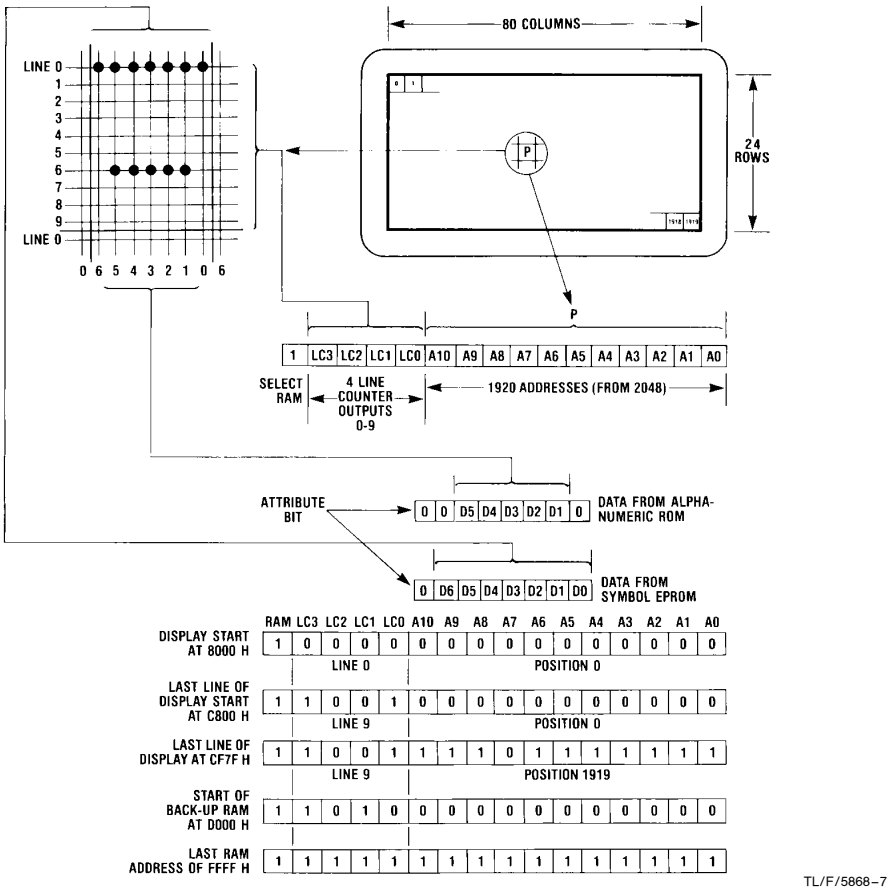


FIGURE 7. RAM Addressing

CRTC Address Bus Configuration

The particular RAM address to be written into is determined by its 10 x 7 character field position and the selected line of that field; refer to *Figure 7*.

The 11 least significant addresses A_0 to A_{10} contain character position information from position 0 to 1919, and the next 4 addresses A_{11} to A_{14} are the 8350 line counter outputs via a TRI-STATE buffer. The most significant bit, A_{15} is used to select the RAM when HI, and the EPROM's and peripherals when LO.

Graphics Design Criteria

In the simple CRT applications, the microprocessor is used mainly to re-write the Refresh RAM as new information is fed in, either from the keyboard, or from the main computer (via ACE). The μP can still be used in this application for alphanumerics/graphics, but it is also desirable if it can perform graphics computations, such as drawing, lines from the inputted coordinates.

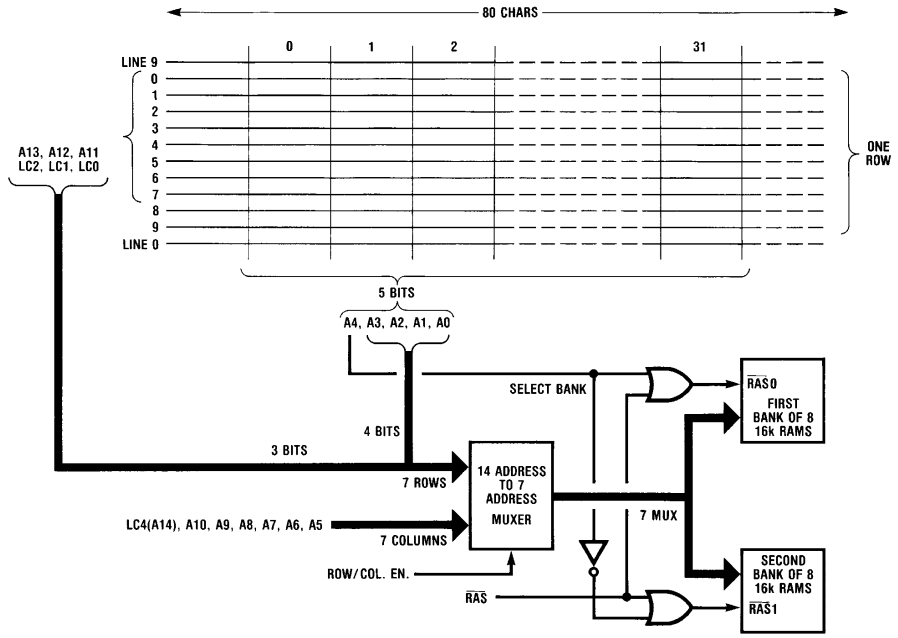
This requires the microprocessor to be able to write 7 dot words quickly to the Display RAM. The best way to implement this is to time multiplex the dot cycle with the CRTC so that whenever the μP requires access to the Display RAM, it merely waits for its slot in the next dot cycle, which could be up to 640 ns later. The information is either written or read after 360 ns, that is a maximum of 1 μs after the memory access request, which is fast enough. Now the μP no longer has to wait for blanking to be able to operate, it continues its normal operation and only enters the WAIT state during RAM access. Although this is for up to 1 μs, in fact it is in general invisible because the μP memory access takes at least 700 ns.

The Microprocessor

The 8080A-2 was chosen for the following reasons:

- FAST—Takes 21.84 MHz (2 x 8350 frequency) divided by 9 (in the 8224), to give a clock cycle of 2.427 MHz, i.e., 0.41 μs per microcycle, or 1.6 μs for a short instruction
- Software can be developed on STARPLEX™ or Intellec Development Systems
- 8080A-2, DP8224 and DP8238 are low cost
- Associated circuitry previously designed in Application Note AN-199

Note the DP8238 has advanced MEMW more—desirable so that the microprocessor can go into the WAIT state earlier in the write cycle.



TL/F/5868–8

FIGURE 8. Automatic RAM Refresh

Interrupts

The INS8259 is ideal as an Interrupt Controller, because most interrupt signals in the system are positive going, saving D-type flip-flops. It can also be used to mask off interrupts when necessary.

Interrupt Priority

1) *Horizontal Sync* from the 8350, highest priority if row start has to be quickly changed, normally masked off
2) *Paralleled 8-Bit I/O Port,* highest priority if CRTC card is part of a master system, otherwise masked off
3) *Vertical Sync* from the 8350, normally highest priority, need to quickly change the Top of Page register for scrolling, to change the display before the new frame begins
4) *ACE,* INS8250—during serial block transfers this will take highest priority
5) *Keyboard*—the time to press the keys is much longer than the interrupt wait time so can be low priority
6) *A/D Converter*—time for conversion is 100 μs so again can be low priority

Display RAM

The system requires a RAM with 24 x 80 x 10 addresses, each of 8 bits (representing 7 + 1 attribute bit), and a cycle time of 640 ns/2 or 320 ns. Using static RAMs 19.2 kbytes would require 40 ICs, whereas using dynamic RAMs 16 ICs are necessary, totaling 32 kbytes. This leaves 13 kbytes available as spare RAM.

Advantages of Dynamic RAMs

- Only 16, 16-pin packages instead of 40, 18-pin packages
- Less than $10 for 16,000 bits
- Fast access and cycle times using the MM5290-2 (average cycle time is 320 ns). Even faster times with the 5V only 16k MM5295
- Standby current only 5% of operating current
- Less average power dissipation than for static RAMs

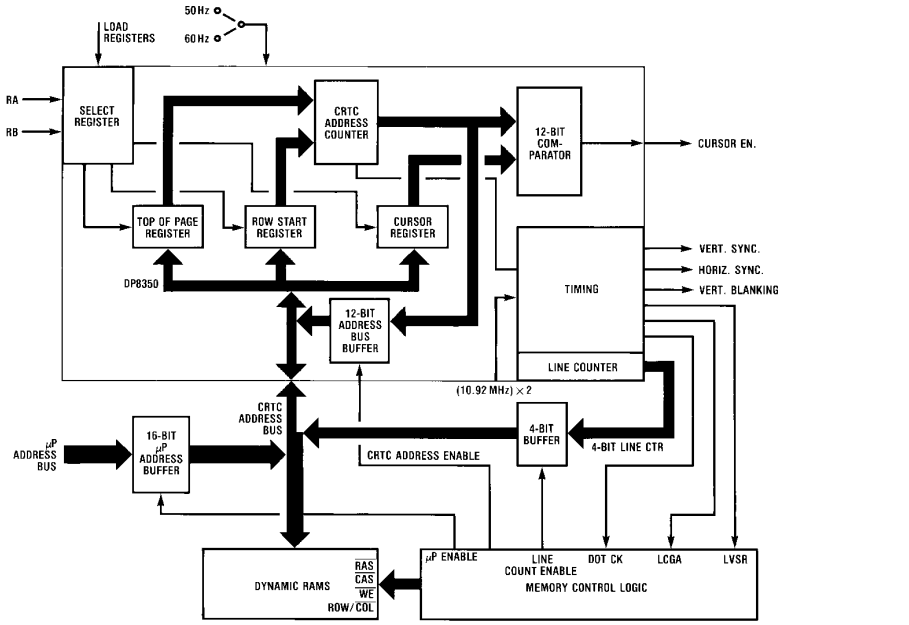
This means average power dissipation is 30 mA x 12V x ½ x ½ x 16 or 1.5W for all 16 packages (only one bank is accessed per cycle by the CRTC for half the dot cycle time). For 40, 4k x 1 static RAMs, average power is 80 mA x 50V x 40 or 16W. Note that if the MM5295 5V, 16k x 1 dynamic RAM is selected, power dissipation will be even further reduced, with access and cycle times about half the 3 rail version.

Disadvantages

- Not easy to interface to
- Need to be refreshed every 2 ms—see ''Refreshing of Dynamic RAMs''
- 3 supply rails needed, +12V, +5V, −5V, but these are already required for the 8080

Refreshing of the Dynamic RAMs

With 16k dynamic RAMs all 128 rows of every RAM have to be refreshed every 2 ms maximum to maintain valid data. It is possible to manipulate the addressing of the CRTC address bus to the dynamic RAM multiplexed address bus, so that there is no need for a separate refresh counter. This is because for any display row, the 8350 sequences all 80 characters, starting at line 0 and ending at line 9. Thus we can use the 3 least significant bits of the line counter outputs (A_{11}, A_{12}, A_{13}, from LC0, LC1, LC2) for three of the dynamic RAM row address bits, (corresponding to lines 0 to 7 of each display row), and the four least significant bits of the character position address (A_0 to A_3) for the remaining four RAM row address bits. See *Figure 8*.



TL/F/5868–9

FIGURE 9. 8350 Block Diagram

Unfortunately, because 19k addresses are required, it is necessary to use two banks of RAM (8 RAMs in each bank), giving a total of 32k x 8. This leaves 13 kbytes available for scratch pad, display storage, and in-system emulation of programs. Therefore each row of this second bank of dynamic RAMs also has to be refreshed. By using address bit A_4 to select the bank, all rows of the dynamic RAMs are therefore refreshed every 32 characters, which in fact is eight lines, or in effect one row of the display. The worst case is when the 32 characters are split over two display rows. There is no problem during the vertical blanking because the 8350 still outputs incrementing addresses and LCGA continues to activate the control logic. So refreshing still continues during blanking. Thus the longest period any RAM row goes without a refresh cycle is 65 μs per line x 10 lines per row x 2 rows = 1.3 ms, which is still within the 2 ms maximum at 70°C. *In other words, dynamic RAM refreshing is automatically performed by the 8350 sequencing the address and no extra circuitry is required.*

CRT Controller

A graphics/alphanumeric CRT Controller requires the following (See *Figure 9*):

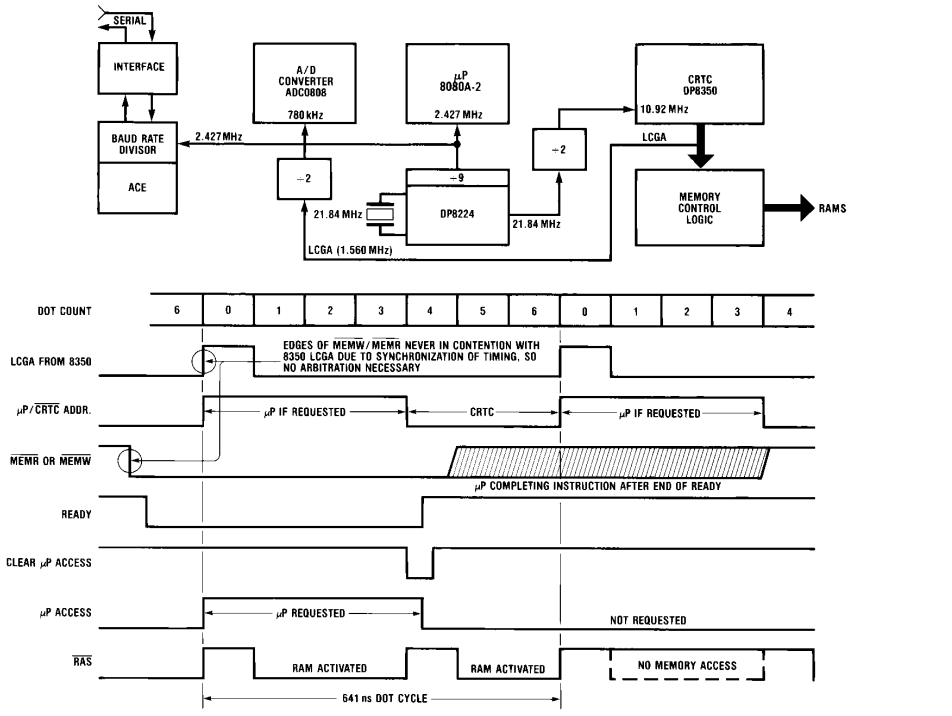
1) All monitor signals provided—the 8350 provides Vert/Horiz sync and vertical blanking
2) Cursor signal—the 8350 has cursor enable
3) Fast dot clock, a 7-dot cycle clock continuous, and a shift register clock only during display—the 8350 has dot clock, LCGA and LVSR
4) Line counter output 4-bit, TRI-STATE—the 8350 has line counter output (but not TRI-STATE)
5) Ability to set top of page, row start and cursor reg at any time—the 8350 can do this using LD REG, RA and RB inputs during the time the μP is on the CRCT address bus. RA and RB can be data bus bits DB0 and DB1, and LD REG can be decoded from the address bus
6) 50 Hz or 60 Hz capability—the 8350 has a frequency select input
7) Incrementing position address, TRI-STATE—the 8350 has this, with a maximum enable/disable time of 30 ns

   This parameter is important in this application where it is necessary to switch the memory from the CRTC address to the microprocessor address, and back to the CRTC address all in one 7 dot cycle of 640 ns. Other CRT controllers are not capable of enabling and disabling the CRTC address so quickly.

Hence the DP8350 requires no extra circuitry apart from a Quad Latch to disable the Line Counter outputs. The 8350 has internal ROMs which determine how many rows (24), columns (80), lines per row (10), and dots per column (7). Versions of the 8350 are available with other combinations.

System Timing

The standard timing for the dumb terminal type of application is shown in the timing section of *Figure 1*, with the microprocessor inactive during display time. This is undesirable for graphics applications where full use of the micro-



TL/F/5868–10

FIGURE 10. System Timing Control Circuit and Diagram

processor is required for computations and peripheral control with very fast baud rate. To determine the timing sequence it is first necessary to calculate the CRTC frequency required for the dot clock.

CRTC Frequency = d × [c + (characters during horizontal blanking)] × [(r × l) + (lines during vertical blanking)] × (line input frequency)

where d = dot per character,

c = columns on display

r = rows on display

l = lines per row

For the standard 8350,

$$f = 7 \times (80 + 20) \times [(24 \times 10) + 20] \times 60 \text{ Hz}$$

$$= 7 \times 100 \times 260 \times 60 \text{ Hz} = 10.92 \text{ MHz}$$

This is too slow for the DP8224 which divides the crystal frequency by 9 to provide the clock to the microprocessor. The 8224 frequency can therefore be 21.84 MHz as in *Figure 10*, and this is divided by 2 to provide the 8350 dot clock of 10.92 MHz, or 91.6 ns per dot. A 7-dot cycle is 641 ns or 1.560 MHz. This is divided by 2, i.e., 780 kHz, to provide a clock frequency for the A/D converter.

The 8080 frequency is 21.84/9 MHz or 2.427 MHz. This frequency is also applied to the ACE to provide the clock for the Baud Rate divider. The baud rate is determined from 2.427 MHz/(16 x Baud Divisor).

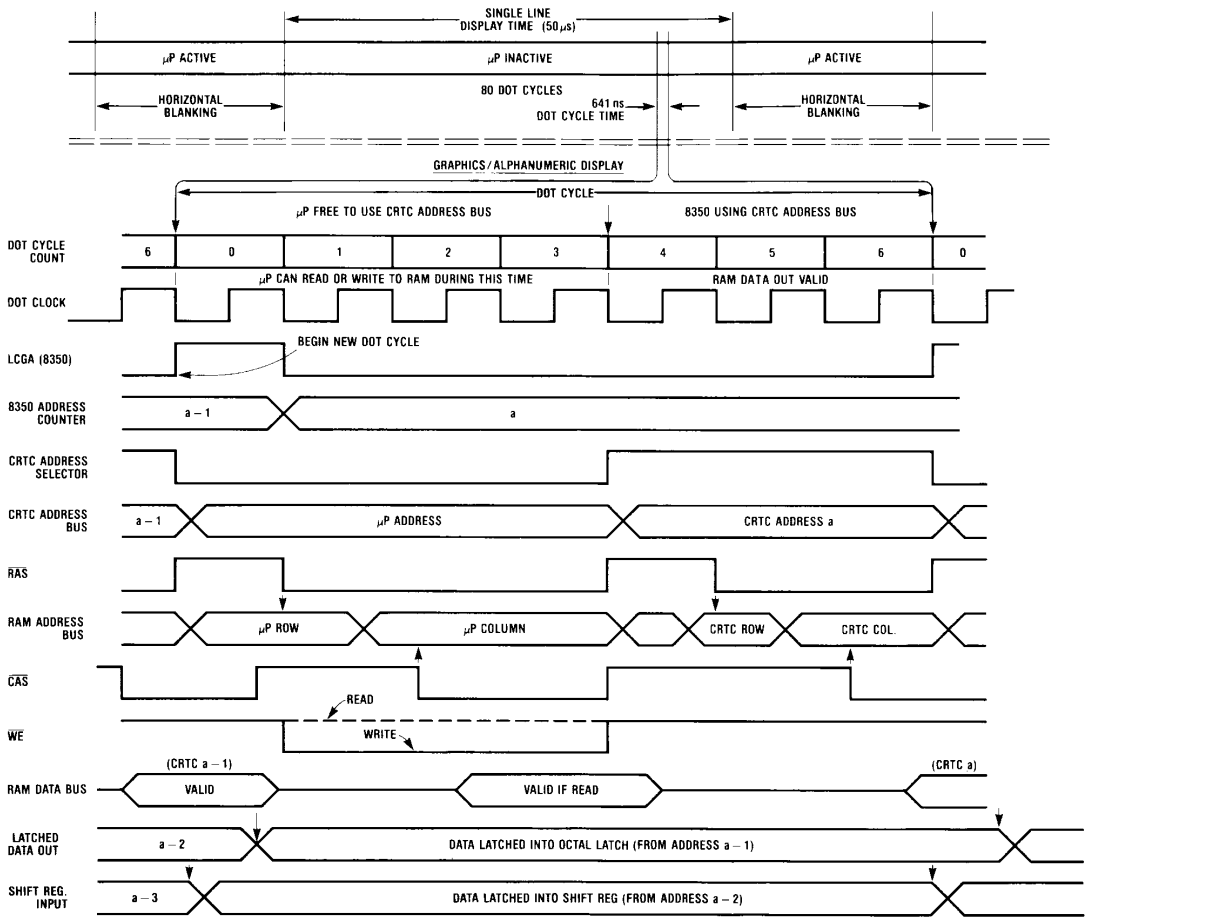
7 Dot Cycle Timing

Figure 11 shows how the 7-dot cycle time of 641 ns can be time multiplexed into two separate control sequences; Microprocessor and CRTC. It is necessary that the new 7-dot word to be displayed, is available at the commencement of the dot cycle shift. Therefore the 8350 must access the CRTC address bus for the second half of the 7-dot cycle, in fact for the last 3 dots of the cycle. This allows the time period taken by the first 4 dots to be used by the microprocessor, so that the microprocessor address output appears on the CRTC address bus for the first four dots, but only if a μP access is requested.

The CRTC 15-bit address is timed multiplexed into 7 rows and 7 columns to be applied to the dynamic RAMs, using multiplexer-drivers, with bit A4 selecting the bank. It is therefore necessary to latch in first the rows with RAS (Row Address Strobe), and then the columns with CAS (Column Address Strobe), for both the μP half-cycle and the CRTC half-cycle. All the set-up and hold times are met by the circuitry of *Figure 12*. If the μP is not requesting RAM access during its half-cycle, the RAS does not occur, although CAS still does. This is because RAS enables CAS internally in the dynamic RAM's, so that if RAS does not occur, the CAS has no effect and the RAM remains in standby mode. This is also the case in selecting the banks with RAS0 or RAS1.

In the second half-cycle, the CRTC always reads the RAM, so WE remains HI, but in the first half-cycle the μP may request a READ or WRITE. WE remains HI for READ, and for WRITE remains LO while RAS is low. Note that the 8350 outputs the address word two dot cycles in advance, and therefore it is necessary to latch and then hold the dot word for one dot cycle. It is then latched into the 7-bit parallel-in serial-out shift register. The 8th bit from the latch can be used as an attribute bit.

Simple Alphanumeric Display



TL/F/5868–11

FIGURE 11. Memory Control Logic Timing Diagram

Figure 12 shows the Memory Control Logic required to correctly sequence the control signals and busses to the dynamic RAMs and associated components. The interfacing from the 8080 microprocessor (via signals MEMR and MEMW) is such that whenever the μP requests to read or write to the dynamic RAMS, the μP Access Flip-flops access the RAMs at the start of the next μP cycle. At the end of these four dots, the information has either been latched into an 8-bit latch (for READ), or written into the RAMS (for WRITE). The READY signal goes active at this time which ensures that valid information is read at the end of the μP cycle; refer to *Figure 10*. Also the fact that MEMR and MEMW occur at fixed intervals relative to the dot cycle signal, LCGA, means that system contention cannot occur. Therefore there is no need for arbitration between these two signals when a microprocessor cycle is requested.

This also applies when selecting the 8350 to change Top of Page, Row Start and Cursor. To select any of these 3 registers, the μP data bus bits D0 and D1 are connected to R_A and R_B to select the required register.

The information to be latched into the selected register has to be valid on the CRTC address bus. Because this is time shared with the 8350 address counter, which outputs the incrementing display addresses during the second half of every 7-dot cycle, the CRTC register information has to be valid for the first half of the next dot cycle. The CRTC is selected with DS6/7 and MEMW, so that REGISTER LOAD occurs just after the CRTC register information becomes valid on the CRTC address bus. The 8350 spec requires that the address be valid 250 ns before REGISTER LOAD trailing edge (old data sheets do not state this), and that R_A and R_B are valid at the leading edge. Note that the 8350 internal address counter can be enabled or disabled within 30 ns of the ADDRESS ENABLE changing state.

All the Logic for Memory Control is Schottky, due to the very fast timing required in the system. Note that the cycle time of the CRTC half-cycle is 270 ns, which is less than the 320 ns specified for the MM5290-2. This parameter is specified at 320 ns for power dissipation reasons, and because the μP is not fast enough to use its half-cycle every 7-dot cycle or 641 ns, the average cycle time is greater than 320 ns.

System Configuration

Figures 13 and *14* together show the system block diagram. The peripheral components of *Figure 13* are used with the microprocessor circuitry of *Figure 14*. The right hand half of *Figure 14* is equivlent to the circuitry of *Figure 12*.

The LS138 address decoder is used for both I/O and memory addressing. Referring to *Figure 15* address map, the peripherals are designated as I/O, and the EPROMs, ROM, CRTC and dynamic RAMs as memory. With address bit A15 HI, the 32k dynamic RAM block is selected. With address bits A14 and A15 LO, the LS138 outputs are selected. A11, A12, and A13 are decoded to select which one of the LS138 outputs goes LO, so that when memory is addressed, each section is 2 kbytes. This includes the CRTC which requires 4 kbytes from 3000H to 3FFFH for 2 pages. The top four address bits select the CRTC and the remaining 12 address bits are latched into the selected register.

When addressing I/O, address bits A0–A7 also appear respectively on A8 to 15, so that with A6 and A7 LO, i.e., I/O address 00H to 3FH, each LS138 output is now 8 bytes selected by A3, A4, and A5. Bits A0, A1, and A2 are then connected as required to the peripherals, to select the addressed byte.

PERIPHERALS

I/O Port

In 00H or OUT 00H select the 8-bit parallel I/O port, which basically is two octal latches with TRI-STATE outputs. The 8 output bits may be connected to a master 8-bit data bus. When an external 8-bit data word is latched into the input octal latch, an interrupt causes this to be enabled on the μP data bus, when acknowledged with the instruction IN 00H. To output to the master databus, OUT 00H causes the μP data to be latched into the output latch and this also provides an external interrupt to the master system. The master can then read this data by enabling the output octal latch. Data can be transferred fast because the I/O port normally has the highest priority interrupt (IR 3 of the 8259), when required.

Interrupt Controller INS8250

This was also mentioned in an earlier section. At initialization, it is set up to remain in the fully nested mode, so that only higher priority interrups may interrupt an existing interrupt. Otherwise a lower priority interrupt has to wait for the higher one to finish. Normally the horizontal sync interrupt to IR2 is masked off if there is no need to change ROW START or soft scroll display data off the screen line by line. The I/O address to select the 8259 can be either 10H or 11H; refer to the 82569 data sheet and the software to determine whether A0 is '0' or '1'. Each interrupt routine has to end with a SET END OF INTERRUPT instruction.

Keyboard

The instruction IN 18H reads ASCII data on the keyboard after a keyboard interrupt has been acknowledged.

Serial I/O Using the ACE INS8250

The 8250 with its associated EIA RS 232 interface allows serial data to be received or transmitted 8 bits at a time, with the instructions IN 20H or OUT 20H. The baud rate is previously determined as described in the software section. Other ACE registers may be accessed, by connecting A0, A1 and A2 of the μP address bus to the same designations on ACE, so that ACE addresses are from 20H to 26H. During block transfers, such as dumping a picture on the screen into an external memory, or loading from the memory, the higher priority inputs can be masked off for fast transfers.

Baud Rate Switch

See 'Baud Rate' for application, the instruction OUT 28H will read the 4 switch positions.

A/D Converter ADC0808

This 8 analog channel, 8-bit A/D converter, has first to be initialized to commence a conversion on one of the channels. Address bits A0, A1 and A2 are used to select the channel, so that instruction OUT 3 nH starts a conversion on INPUT n. The conversion takes about 100 μs with the 780 kHz clock, so the μP can continue operating during conversion. The END OF CONVERSION signal then interrupts the μP, which when acknowledged reads the 8-bit data with the instruction IN 3 nH, although n is not important in reading the A/D.

The A/D converter being only one 28-pin chip, is ideal for demonstrating the graphics capabilities of the system. For instance, an x-y joystick can be connected to INPUT 0 and INPUT 1, so that the movement of the joystick draws on the screen.

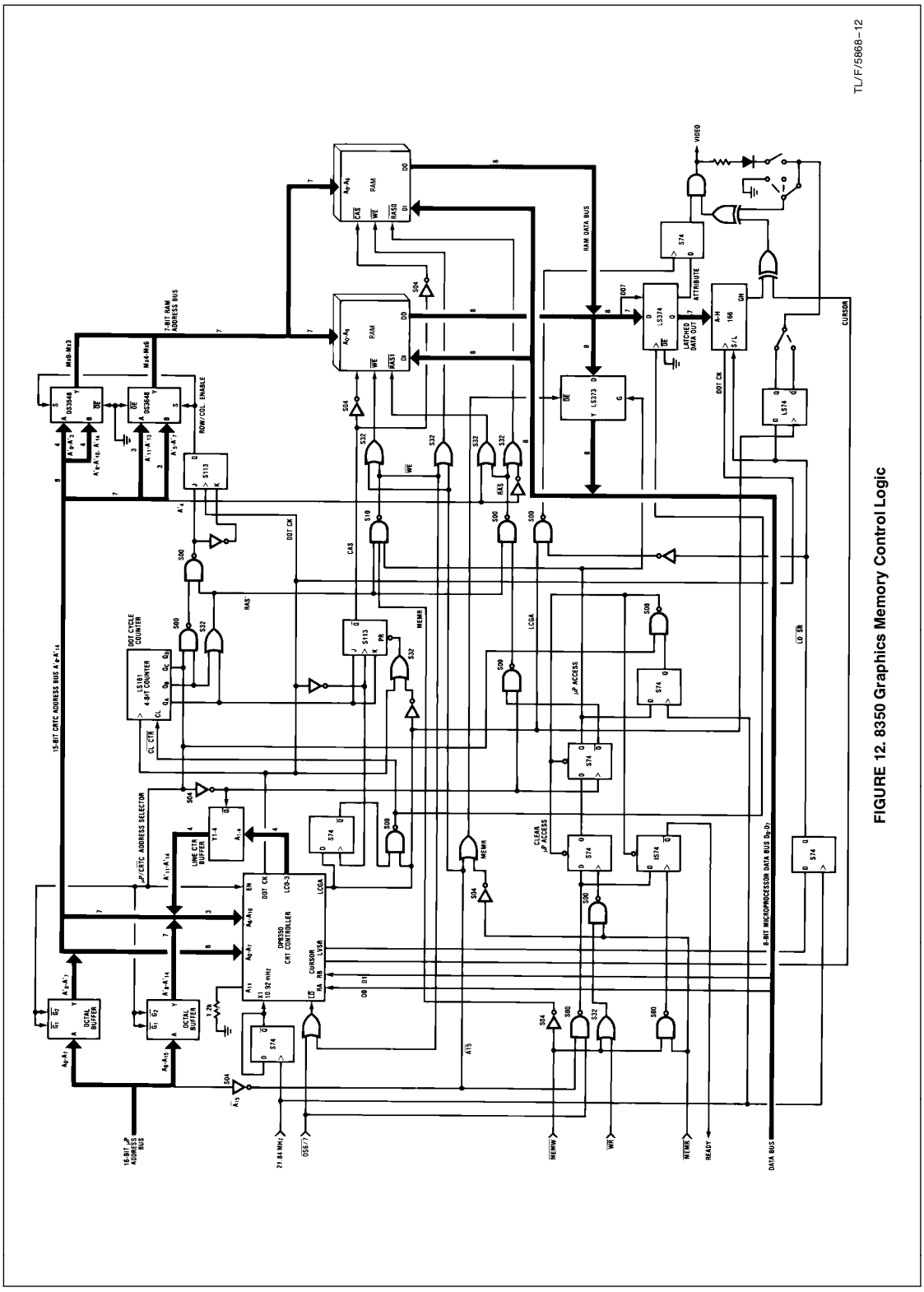


FIGURE 12. 8350 Graphics Memory Control Logic

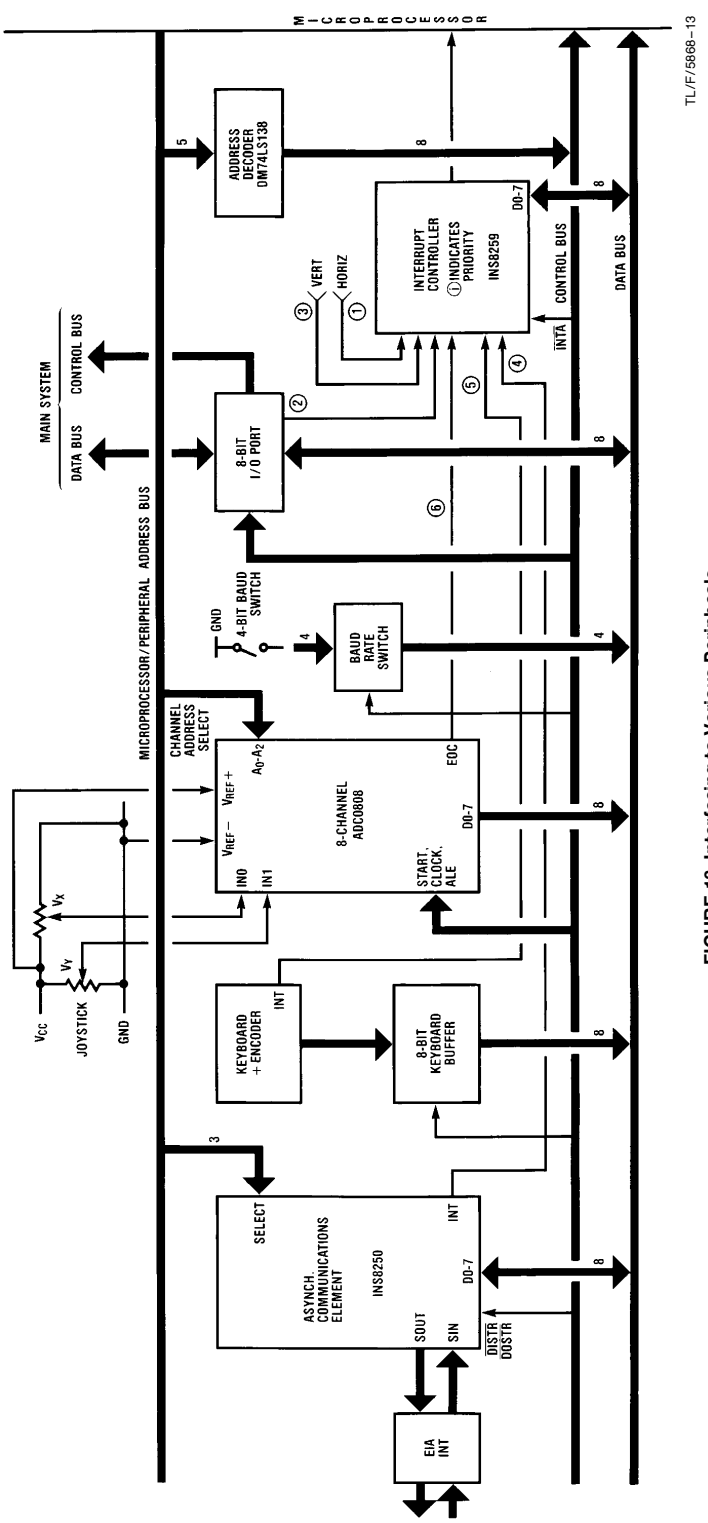
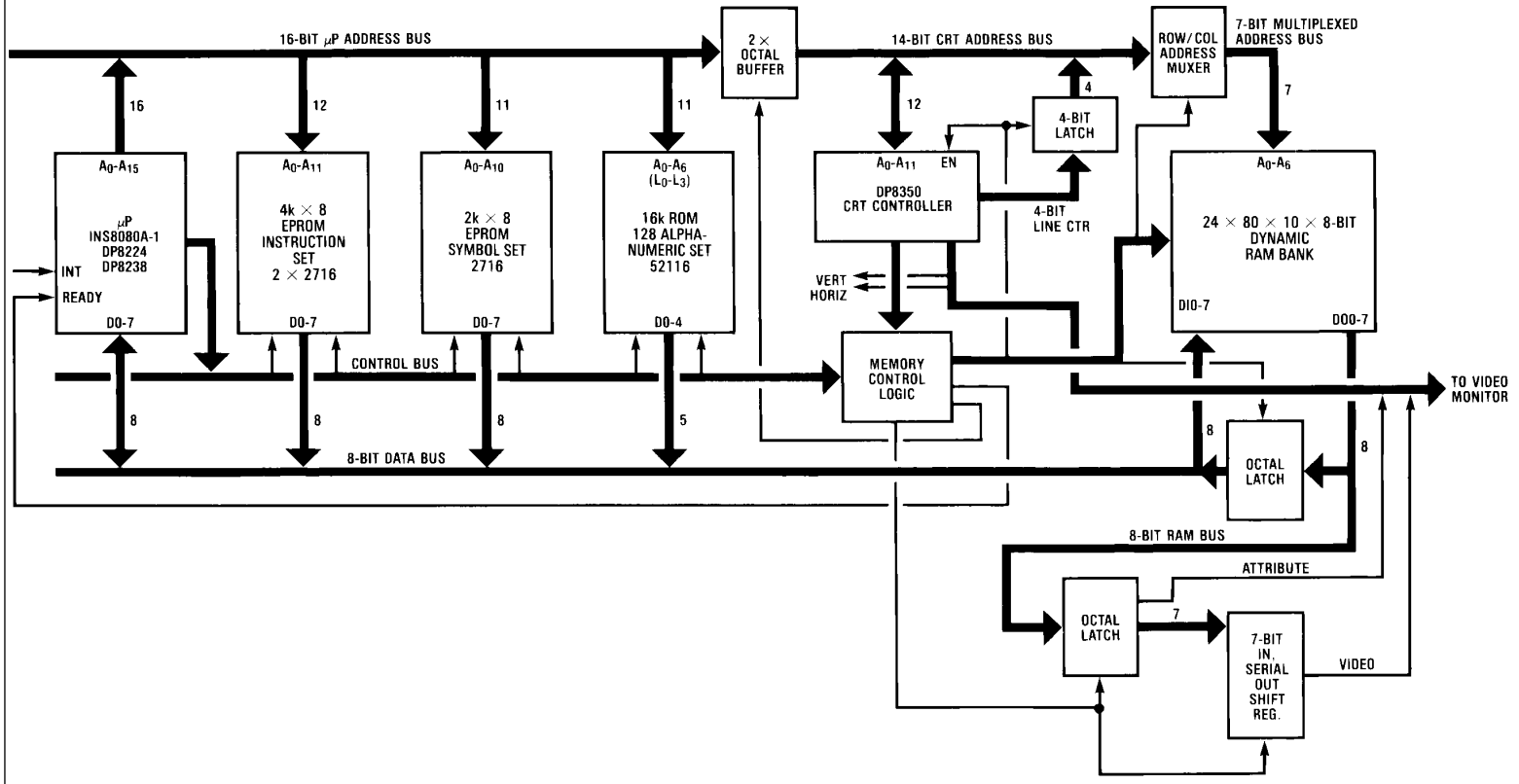
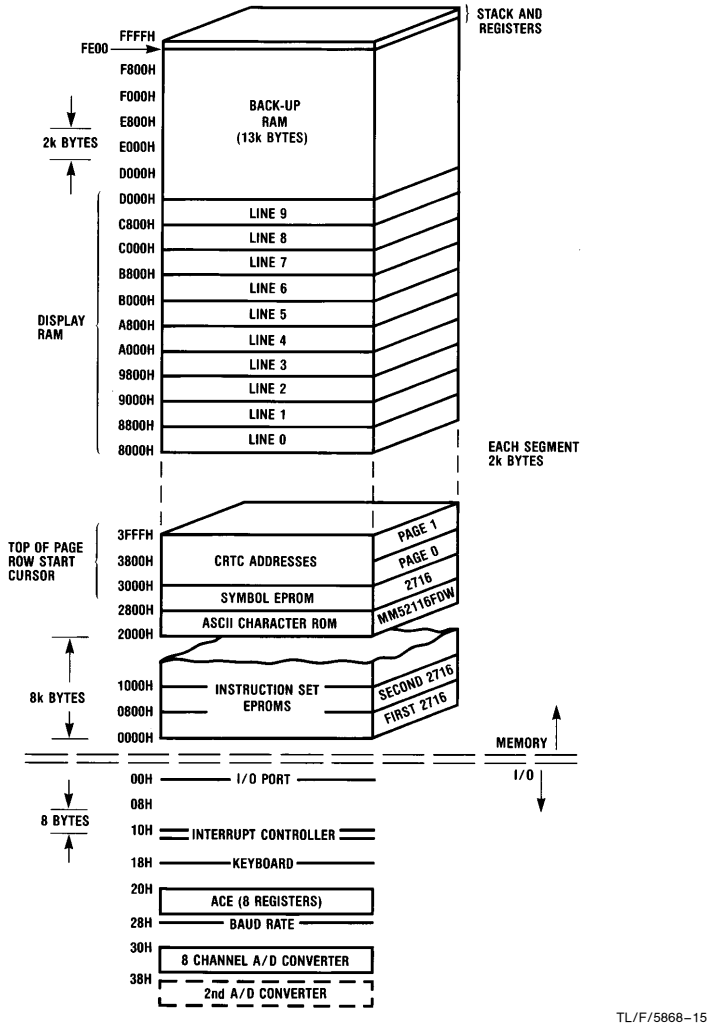


FIGURE 13. Interfacing to Various Peripheals



TL/F/5868–14

FIGURE 14. Complex Alphanumeric/Graphics/Symbol Display Terminal



TL/F/5868–15

FIGURE 15 I/O and Memory Map

System Operation and Software

The software was developed purely for demonstration purposes to show the versatility and power of the system. All the software has been tested, but the system could be much more powerful with additional software. The 13 kbytes of back-up RAM are also useful in this respect. The software was developed on National's STARPLEX Development System. The instruction set so far is just under 4 kbytes, so two 2716's are used, but these may be replaced by 2732's if the chip select pins are reconnected, so that extension up to 8 kbytes is possible with no extra IC sockets.

Parameter Definitions

The software is structured as in *Figure 16*. The philosophy was to make it versatile, easy to understand, and easy to modify or add to.

The registers are stored in the dynamic RAMs starting at FE00H in the non-display section. The Top of Stack is also in the RAMs at the highest location, FFFFH. This allows for about 240 nested two-byte PUSHes or CALLs, which is comfortable. Any register may easily be relocated merely by changing its address, similarly any new registers may be added to the list.

The addresses of the various memory and I/O locations are also listed and defined in the front section so these can be changed as desired.

For complete versatility, the display parameters are also listed in the front section so that any different value of parameters from those listed need be changed only in this section. The values of the parameters or constants are those of the standard DP8350 around which the hardware has been designed.

Thus by defining most parameters in the software once, at the beginning, the subsequent routines/subroutines will be valid for different applications and should not need to be altered, merely added to. Not many macros were used in order to save EPROM instruction space.

Interrupt Entry Locations

These are in 8-byte increments beginning at 0010H. The 16 bytes before this are saved for power-up initialization to disable interrupts and set Top of Stack.

Each interrupt location calls that interrupt subroutine. At the end of the subroutine, the system returns to output an END OF INTERRUPT to the 8259, and then returns to the original subroutine in progress when the higher priority interrupt occurred. If no interrupt was in progress, the program returns to the WAIT LOOP which enables all unmasked interrupts to the μP.

Look-Up Tables

This has three sections. First, the BAUD RATE DIVISOR look-up table contains all the 16-bit divisors required for baud rates from 110 baud to 19k baud.

The next look-up table contains PROGRAM LABELS, used in the SEARCH FOR PROGRAM subroutine. The first row contains all the first characters of the program labels, the second row contains the second character, up to the fourth row contains the fourth character. Each program consists of four characters.

The third table is the address list so that once the SEARCH subroutine has located the desired label, it alters the program counter to the equivalent section of this table, which then calls up the program requested.

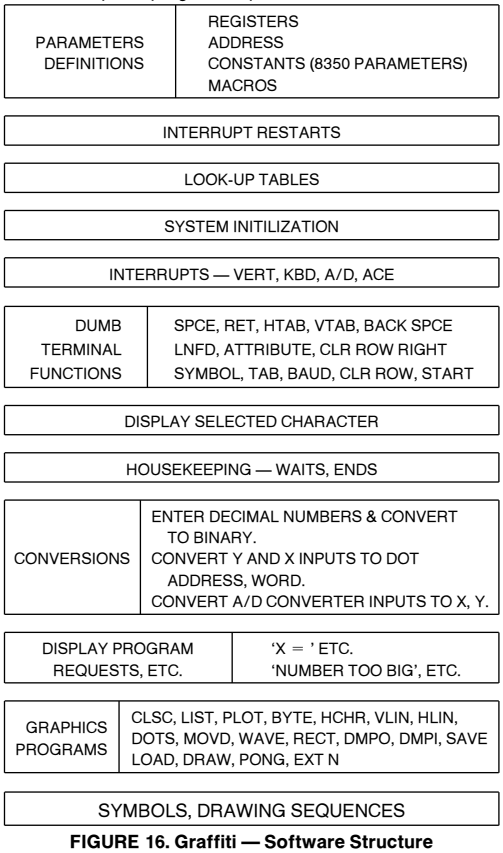


FIGURE 16. Graffiti — Software Structure

System Initialization

After disabling interrupts and setting Top of Stack, the 32k addresses of RAM are cleared one byte at a time, so that the screen is blank within half a second of switch-on. The cursor is then homed to the first character position. First the Top of Page register is set to 0 in the CRTC and then the cursor register is set to 0, both in the RAM and the CRTC. The column count is also reset.

The ACE is next set up including the baud rate (see Baud Rate section). Next, the Interrupt Controller is set up, and after this the system enters the WAIT LOOP system, enabling the interrupts to wait for an interrupt.

Interrupts

1) *Horizontal Interrupt* is normally masked off but may be unmasked for two reasons: either during scrolling, so that each row can be soft scrolled off the screen a line at a time, or during editing to delete a row, so that a jump in ROW START to the next row has to occur every frame at this new row. This new row must be loaded after horizontal blanking of the last line before the jump row is to begin.

   Note that if the ROW START register is not loaded, each row start address is the last display address incremented.

2) *Port Interrupt* is normally masked off, but must be unmasked if transfer of data to a master system is necessary.

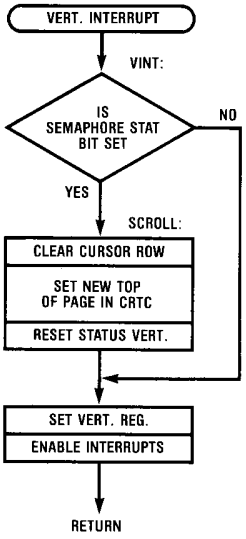
3) *Vertical Interrupt* is used for two purposes. One is to scroll the display by one row, once the scroll semaphore bit has been set in one of the associated subroutines, this is begun at vertical interrupt so that screen flicker does not occur. The other is to change a graphics display every frame so that smooth transition of a subject across the screen is attained. An example of this is the program PONG. The flow chart for Vertical Interrupt is shown in *Figure 17*.

4) *ACE Interrupt* is by far the most complex because data received by this chip then has to be operated on to determine what action to take. The flow chart for ACE Interrupt is shown in *Figure 18*. Assuming the interrupt is because ACE has received data available, the ASCII data is checked for a function input. If not a function, but a program is already in progress awaiting inputted data, then this character is saved in the Input Character register. If the character was entered while the cursor was in the first four positions of a row, then the character is saved in a register determined by the column position of the cursor. This saves the character to recall it in a look-up comparison later, while searching for a program. Unless the ASCII code was a function, the character is displayed in the cursor position (see Displaying Characters).

   If the ASCII code entered is a function, then first CARRIAGE RETURN is checked for. If negative, then all the other functions are checked for and if positive, that particular function is executed. If the input is in fact a carriage return, then a check is made to see if the cursor was in the 5th position, signifying a four character graphics program has been requested. The system then goes to search in a look-up table for a program corresponding to the four ASCII characters entered in order. If a program is found, the system then calls the requested graphics program and executes it. If not, then a carriage return is executed.

5) *Keyboard Interrupt* in most systems is a simple subroutine, merely accepting the ASCII data word from the keyboard and outputting it to the ACE (see Displaying Characters).

6) *A/D Converter Interrupt* sets the A/D semaphore bit.



TL/F/5868–16

FIGURE 17. Vertical Interrupt Flow Chart

Functions

A number of dumb terminal functions are available with the present software: carriage return, line feed, advance cursor, backspace, up cursor, tab 8 positions, clear row, clear row right of cursor, scroll up one row, and selecting attributes. Attributes available are half-intensity characters and character inversion. Each 7-dot location has its own attribute bit.

Baud Rate

The 4-bit BAUD SWITCH is used to select the BAUD RATE at switch on, or during operation if CTL E is entered. The μP then reads the switch setting and loads the corresponding 16-bit BAUD RATE DIVISOR into the INS8250 Asynchronous Communications Element. Baud rates from 110 to 19k are available, and up to 56k is feasible if the 8080A-2 μP is selected for fast data rates.

Displaying Characters

When a key is depressed, the keyboard outputs the ASCII code of the key selected, which is read by the μP when the keyboard interrupt is acknowledged. The μP then outputs the same ASCII data to the INS8250 ACE to be transmitted serially via the RS232 interface. This can be connected to a main computer, or an identical terminal, or back to the serial input of the ACE. When the ACE receives the returning 8 bits, it outputs a RECEIVED DATA AVAILABLE INTERRUPT or RCDA. The received data is then read by the μP, which selects the ASCII character from the 128 character ROM using the ASCII code as address. The alphanumeric character is copied line by line into the dynamic RAM in the position of the cursor.

Initially all RAM locations are '0' and the dots are written as '1', in a 7-dot word. See *Figure 19*. Then every frame, as the 8350 scans each line, the 7-dot word for each character position is latched from the RAM into the 7-bit shift register, and outputted serially during the next 7-dot cycle so that each '1' appears s a dot. The standard ASCII characters are displayed in a 7 line by 5 dot format or font. The 7 lines are copied line by line into the first 7 lines of the 10 line character field, leaving lines 7 through 9 as vertical spacing between characters. Data bits 1 through 5 are used for characters, leaving dots 0 and 6 as spacing between adjacent characters. The keyed character then appears on the screen and the cursor is incremented to the next position.

Additional Symbols

An additional 2716 EPROM with pre-programmed electronic symbols can be selected instead of the ROM, so that circuit diagrams can be drawn on the screen. Each symbol in the EPROM can be 10 lines of up to 7 dots so that each character may be continuous into the next—a necessity for circuit diagrams. The EPROM is selected by typing CTL Z on the keyboard and then a key, which can be either upper or lower case. This then displays the appropriate symbol in a similar manner to an alphanumeric character. To return to alphanumerics again, another CTL Z is required from the keyboard.

Two sequences are also stored in this EPROM, at addresses 1D00H and 1E00H. When either of these are called up by the program DRAW, a circuit diagram is drawn on the screen. This is an efficient way of storing circuit diagrams. Each circuit sequence, requires about 200 bytes, which is not a lot to cover most of the screen, much less than the 19 kbytes normally required to save every dot.

Although the symbol EPROM was programmed for electronic symbols, other kinds of symbols may be programmed into this EPROM, such as mechanical symbols.

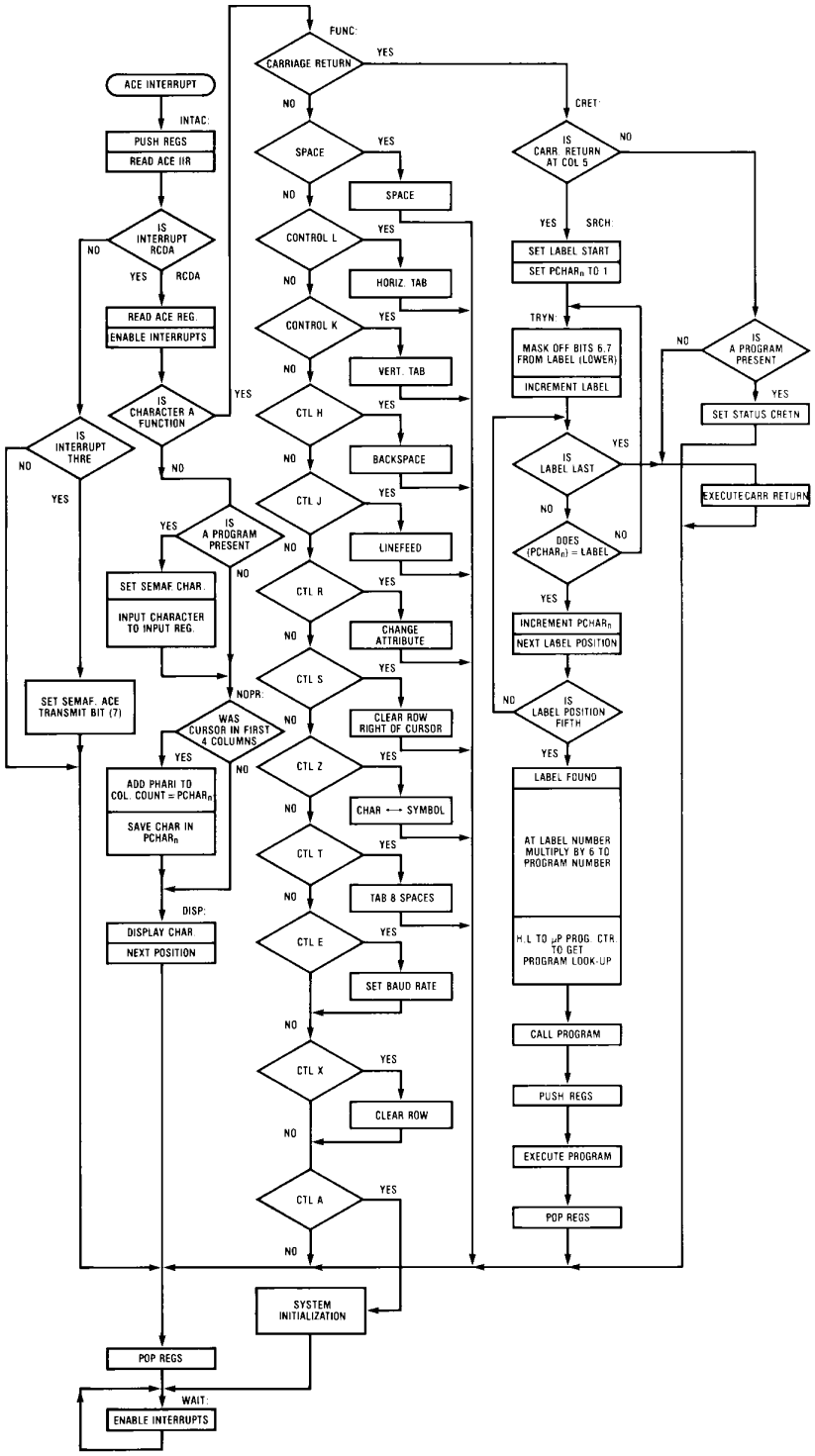
Programming this EPROM is not easy. Assuming ASCII characters are to be used to select each symbol, then the addresses A_6, A_5, A_4 must be 100, 101, 110, 111 corresponding to ASCII codes 4XH to 7XH, where X is address A_3, A_2, A_1, A_0. The 4 lines LC 3, LC 2, LC 1 and LC 0 go to address bits A_{10}, A_9, A_8, A_7. The EPROM is selected with 00011B to A_{15}, A_{14}, A_{13}, A_{12}, A_{11}. In other words, to select the first line of character (41H), the address would be 1841H, and for the second line 18CH etc.

Locating the Position of a Dot

The standard DP8350 displays 80 horizontal characters for each of 24 rows, each character field comprising 10 lines of 7 dots. Thus there are 80 x 7 or 560 horizontal dots and 24 x 10 or 240 vertical dots in the display. Let the value of the horizontal dot position be x, where $0 \leq x < 560$, and y be the vertical dot position, where $0 \leq y < 240$. Refer to *Figure 20*.

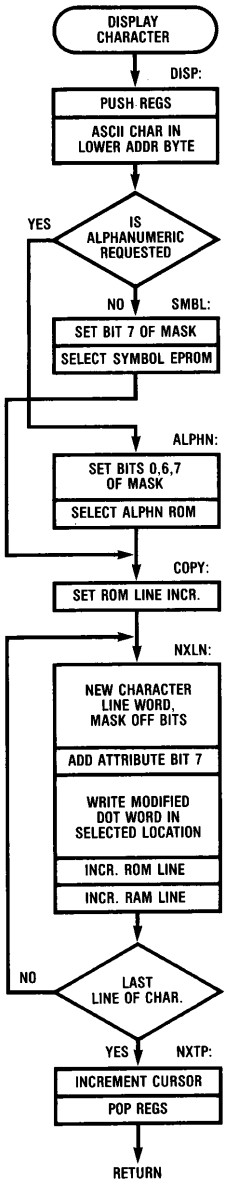
If the x and y values are inputted to the microprocessor, it can then compute the character position, the line number and the dot position number. First, the Row Number r is INTEGER (y/10). This then has to be multiplied by 80 to produce the ROW START number. The Column Number then has to be added to this to obtain the Character Position Number, where the Column Position c is INTEGER (x/7). The line of the row is $(y - r)$, and the dot number is $(x - c)$ for the computed character position.

For the 8080 microprocessor, multiplication and division of numbers is laborious and time consuming. It is therefore easier to use the program subroutine shown in *Figure 21* to compute the character position, line number, and dot number. A separate subroutine then computes the dot word from the dot number. This 7-dot word is then ORed with the word already in the computed dynamic RAM location. All this can be demonstrated using the program PLOT.



TL/F/5868–17

FIGURE 18. ACE Interrupt Flow Chart



TL/F/5868–18

FIGURE 19. Displaying Character

This computation takes an average of 300 μs and a maximum of 500 μs. Hence up to 3,000 dots can be plotted per second for any values of x and y to create a graphics display.

A good demonstration of the graphics capability is to connect an x–y joystick to two analog inputs of the A/D converter and by selecting the program MOVD (move dot), moving the joystick. The joystick can be moved quickly from one extreme to another and all dots on the way are displayed. This program can also use a dot as the cursor, using the joystick to select its position, and then to depress keys whenever a desired character is required at the character position of the dot.

Dot Word Transfers

With the use of the ACE, it is possible to unload the contents of the RAM into either an identical terminal to copy the display, or to store it in a main computer. It can then be recalled from the computer at a later date and re-loaded into the RAM to be displayed. Or if desired, sections of the display can be transferred. Copying from or to the display can be fast, because 7 dots are read or written at a time. An example of this is to use the programs SAVE and LOAD. A section of the display (such as a circuit diagram) can be saved in the back-up RAM, and then loaded back on to the display in a different area. The diagram appears almost instantly.

This extra 13 kbytes of back-up RAM can also be used as additional memory for in system emulation of programs, or for powerful computing capability for graphics calculations.

Graphics Programs

To perform various graphics functions it was decided to select the necessary software with four letter program labels, followed by a carriage return. As long as the label derived starts at the first column of a row, the program requested is called up and executed. Some programs request information from the operator. PLOT is an example of this, where the values Y1 and X1 are requested by the display. The user types in the desired values in normal decimal, signifying the end of the number with a carriage return. After both y and x have been entered, the program continues, in this case plotting a dot at Y1, X1.

Conversion of Entered Decimal Numbers

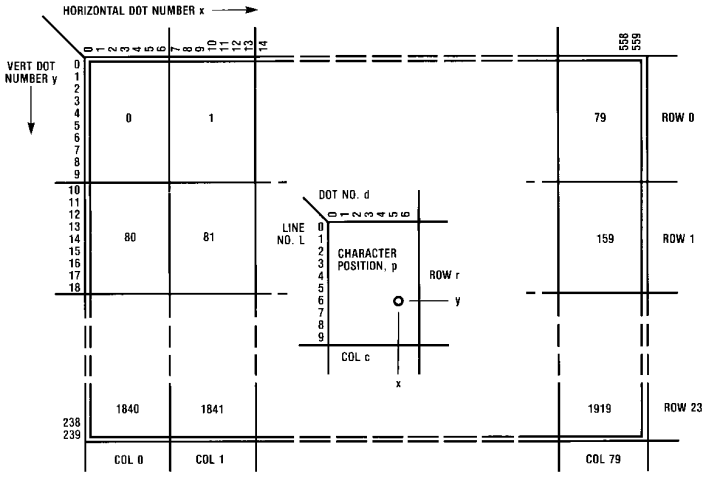
The conversion of the decimal numbers entered and saved in the Input Number registers, is performed by the subroutine ENTR. First the last decimal number entered (obviously units) is tested for ASCII number units and then saved. The second number (tens) if entered, is then tested and decremented until 0 is reached, and each decrement, 10 is added to the total number. Then the third number (hundreds) is tested and decremented to 0, each time 100 is added to the total. At the end of the conversion, H, L register contains the total number in binary. This is then saved in the respective register.

Conversion of 2 Hexadecimal Characters to an 8-Bit Word

This subroutine takes 2 ASCII characters each in the range 0 to 9, A to F, and converts them to a binary word. First, the 3 ASCII code bits are masked off the number first entered. This is shifted left 4 times and added to the masked off 4 bits of the second number entered. This 8-bit word is now 7 dots plus one attribute bit. With this method, it is easy to write/read words quickly on to the display, in the selected location. This can be demonstrated with the program DMPI (dump-in) as in the next two sections.

Display Loading

The starting address is first entered (anywhere from 8000 H to FDFFH) by keying in the first two hex numbers when requested by B = (byte), no carriage return, then the last two hex numbers. This is repeated for the end address. The bytes are then entered 2 ASCII characters at a time. If the addresses are between 8000H and CFFFH, the words will appear on the display. For example, 7F will appear as 7 dots, or 83 will appear as the 2 right-hand dots with attribute. In this way a picture can be loaded on the display.



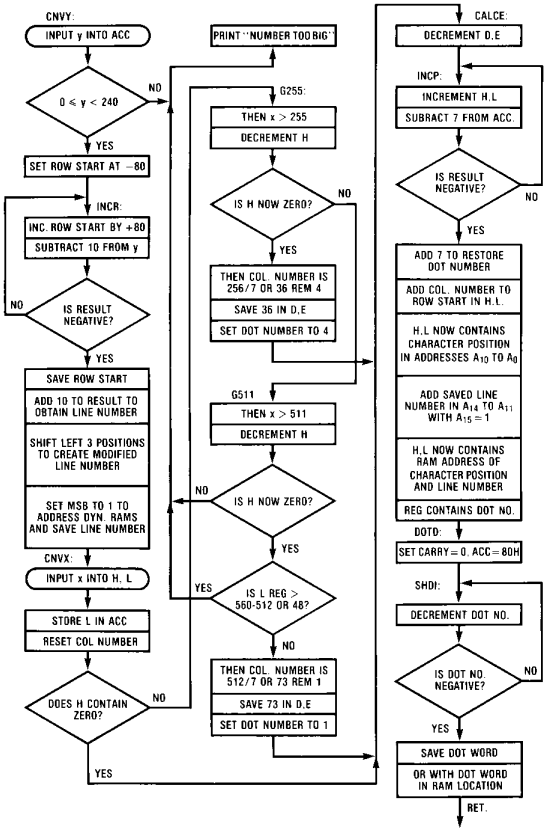
TL/F/5868–19

ROW r = INTEGER (y/10), LINE L = y–r

COLUMN c = INTEGER (X/7), DOT NUMBER d = x–c

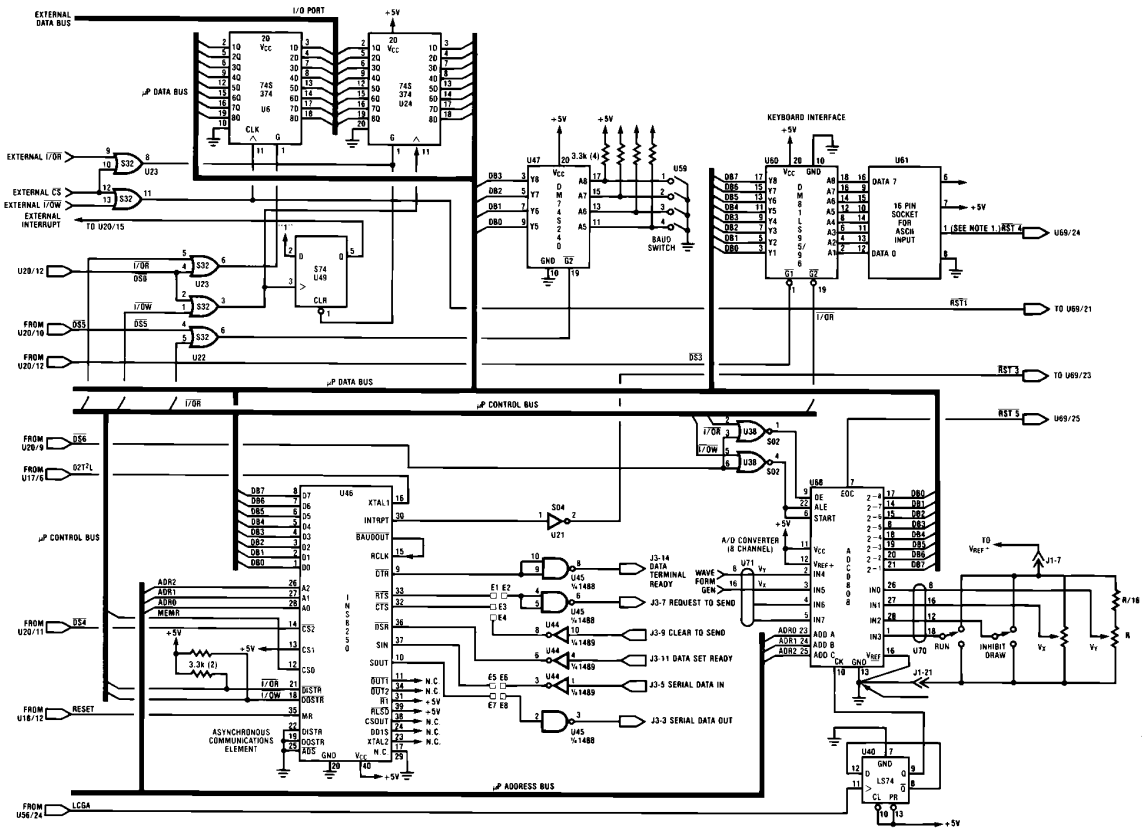
DOT AT LOCATION x, y IS IN CHARACTER POSITION p, LINE L, DOT NO. d, WHERE p = 80r + c

FIGURE 20. CRT Display/8350 Character Address Positions



TL/F/5868–20

FIGURE 21. Flow Chart to Add Dot x, y to Display



TL/F/5868–21

Note 1: Strobe Input must be positive (+)

Note 2: All devices bypassed with 0.1 μF 50 VDC capacitors between supply voltages and ground.

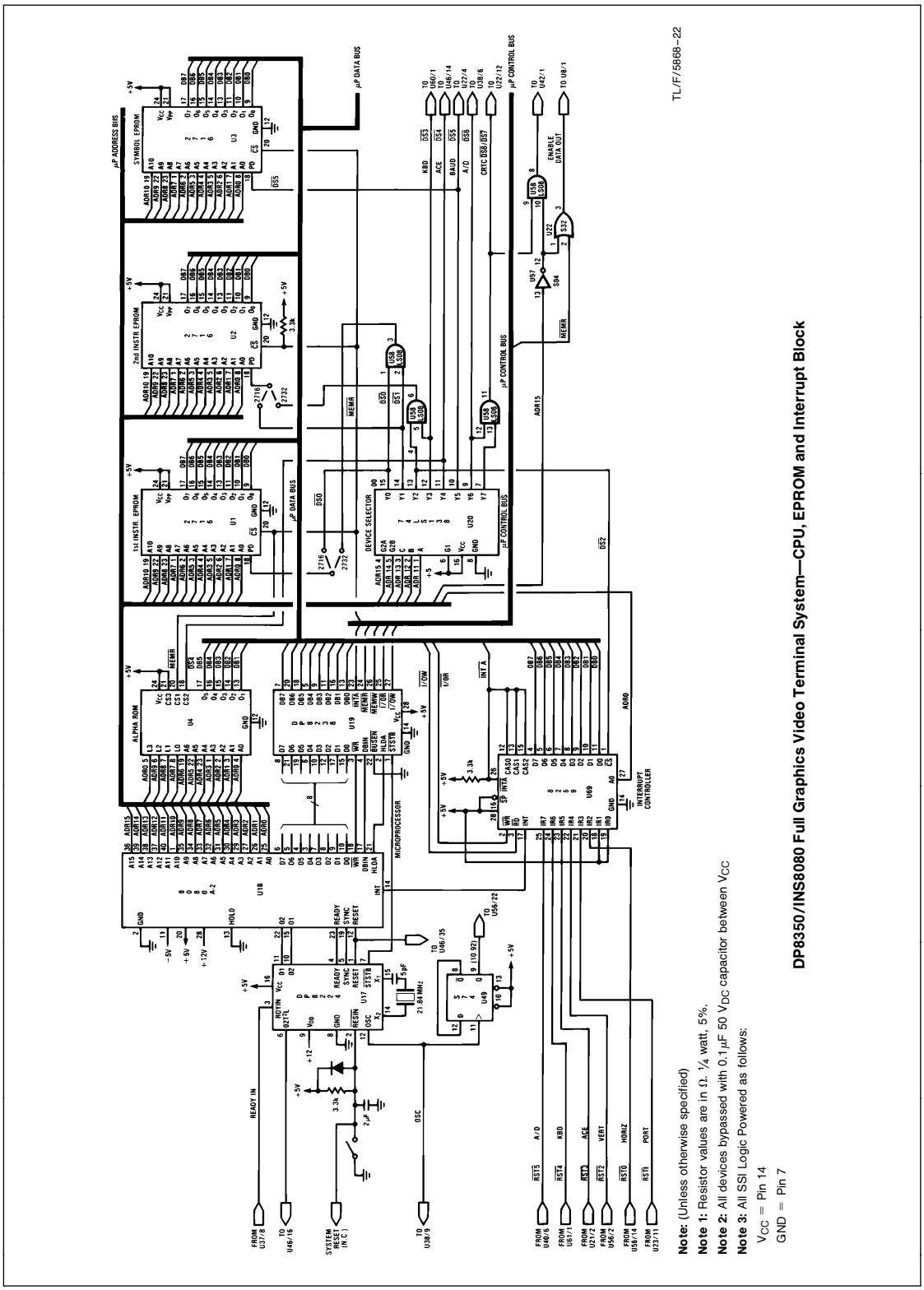
Note 3: Unless otherwise specified, SSI Logic Power as follows:

V_{CC} = Pin 14

GND = Pin 7

Note 4: For serial transfer to other systems Link E1–E2, E3–E4, E5–E6 and E7–E8 for stand alone use, connect E1–E3 and E5–E7.

DP8350/INS8080 Full Graphics Video Terminal System—Peripheral Interface Block



Note: (Unless otherwise specified)

Note 1: Resistor values are in Ω. ¼ watt, 5%.

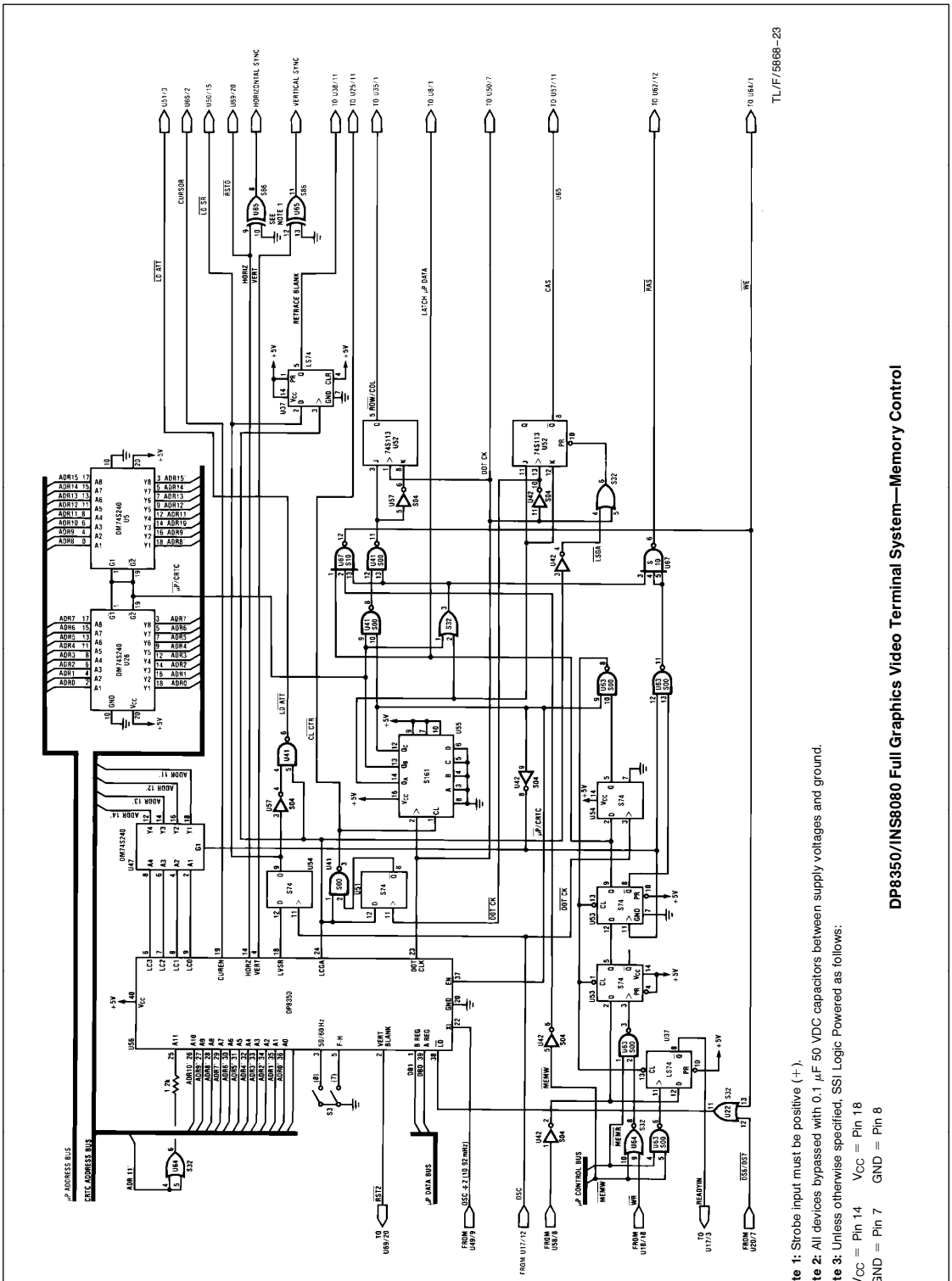
Note 2: All devices bypassed with 0.1 μF 50 V_{DC} capacitor between V_{CC}

Note 3: All SSI Logic Powered as follows:

V_{CC} = Pin 14

GND = Pin 7

DP8350/INS8080 Full Graphics Video Terminal System—CPU, EPROM and Interrupt Block



Note 1: Strobe input must be positive (+).

Note 2: All devices bypassed with 0.1 µF 50 VDC capacitors between supply voltages and ground.

Note 3: Unless otherwise specified, SSI Logic Powered as follows:

V_{CC} = Pin 14 V_{CC} = Pin 18

GND = Pin 7 GND = Pin 8

DP8350/INS8080 Full Graphics Video Terminal System—Memory Control

Use of Back-Up RAM

If the DMPI addresses are between D000H and FDFFH, the information is stored in the back-up RAM. This is useful for in-system emulation, for example. By calling up the program EXT0 (External 0), if a program has previously been loaded in the back-up RAM, starting at address D000H, this program will then be executed after EXT0, carriage return. Another use of this section of RAM is the storage of different sequences of circuit diagrams other than those in the symbol EPROM. The program DRAW can then call up the starting address.

Additional Software

The power and versatility of this system is easily demonstrated with the existing software. This can be added to as required with new software, calling up existing subroutines where possible. Up to 4 kbytes of additional software can be incorporated without any hardware modifications (other than moving two links to select 2732s instead of 2716s).

Conclusion

So using all National Semiconductor ICs, at a cost of a few hundred dollars, the hardware for an intelligent terminal with full graphics capability can be fitted on one BLC80/SBC80 size card. The design is easily expandable to systems requiring color. The biggest modification is to the memory; instead of one bit per dot. 3 bits ae required for blue, green, and red, to give 8 possible combinations per dot. A small number of extra logic ICs are required, as are minor additions to the software. To select the color, a CTL key can be used followed by the code of the color. This color will then be written until changed by the CTL key. A differential CTL key followed by a number could previously have set the background.

Present Capabilities of Alphanumerics/Graphics System

Dumb Terminal Functions

**All 128 ASCII Characters Displayable	
**Space	
**Carriage Return	
**Horizontal Tab	(→) or (CTL L)
**Backspace	(←) or (CTL H)
**Linefeed	(↓) or (CTL I)
**Vertical Tab	(↑) or (CTL K)
**Select/Deselect Attribute	CTL R) with SW-1 or S-3
**Tab 8 Spaces	(Tab)(CTL T)
**Clear Cursor Row	(CTL X)
**Clear Row Right of Cursor	(CTL S)
**Initialize System	(CTL A)
**Select Baud Rate from 110 to 19,200 Using S-1 and CTL E	
**Scrolling Upwards	

Non-Standard Character/Symbol Selection

**By selecting CTL Z, symbols can be displayed for each key of the keyboard, including shifted and control keys. Also can deselect back to standard ASCII characters with CTL Z.

Graphics Programs

*CLSC: Clears screen only, leaving 13k back-up RAM unaffected

*LIST: Lists all graphics programs.

*PLOT: Plots a dot at X, Y, X is the number of horizontal dot positions from the left of screen, from 0 to 7 x 80 for the 8350, i.e., $0 \leq x \leq 559$. Y is the number of vertical dots from the top of the screen, from 0 to 10 x 24, i.e., $\leq 0 \leq y$ 239. The operator keys in the decimal values of Y and then X when requested by the display.

*VLIN: Draws a vertical line between Y1 and Y2 at X. These values are entered decimally by keyboard when requested by the display.

*HLIN: Draws a horizontal line between X1 and X2 at Y1. These values are entered decimally by keyboard when requested by the display.

*RECT: Draws a rectangle linking lines X1, X2, Y1, and Y2.

*PONG: Bounces a dot around the screen between the four walls of the display.

*DRAW: Draws a diagram on the screen from a sequence of operations saved in ASCII code in memory. The START address of the sequence is determined by the first four hexadecimal characters entered on the keyboard. The address 1D00H selects a DC voltage restoring circuit sequence located in the symbol EPROM. Address 1E00H selects a logic circuit and waveforms. Test sequences can be loaded into back-up RAM using program 'DMPI' at the starting and end address entered. This start address is then called up by the 'DRAW.' The end address must contain 0 (zero).

*SAVE: Saves in the back-up RAM a section of display contained within rows R1 to R2, and columns C1 to C2. These values are entered decimally by keyboard when requested by display. The start address in the back-up RAM is selected by the first four hexadecimal characters entered on the keyboard.

*LOAD: Loads from the back-up RAM to a section of display bounded by R1, R2, C1, C2. These values are entered decimally by keyboard when requested by the display. The back-up RAM start address is selected as in SAVE.

*DOTS: Plots N dots on any line Y1 at positions X1, Y1; . . . XN, Y1; and then any new line entered in decimal by the operator. Ends the program by entering 0 (zero) when the next Y1 is requested.

*MOVD: Uses the 8-channel 8-bit A/D converter to monitor the voltages on X–Y joystick, an inhibit-draw switch, and an exit-program switch. In the DRAW mode consecutive dots are plotted to create a picture as described by the movement of the joystick. All these input signals are connected to the first A/D socket. In the inhibit DRAW mode the dot is moved around by the joystick as a cursor, and by keying in from the keyboard the desired character, this character will appear in the character field of the dot. This moving dot can be used to erase existing dots, or erase characters by keying 'SPACE' in the desired position. To exit the program, set the EXIT program switch in EXIT-DRAW mode with the Inhibit-Draw Switch in INHIBIT.

*WAVE: Uses the A/D Converter to create waveforms on the screen when the signals are connected to the second A/D socket.

*DMPO: Unloads any part of RAM to an external system starting at an address keyed in by the operator in hexadecimal characters (four) and ending at another similarly entered address. The RAM is unloaded 7 dots at a time per line of character and converted to two ASCII characters and then transmitted serially.

*DMPI: Loads any part of RAM from an external source (or the keyboard) starting at an address-selected by the first four hexadecimal characters entered on the keyboard and ending at another similarly entered address. The RAM is loaded 7 dots at a time per line of character, keyed in by two hexadecimal characters, for each word. The addresses selected can be display addresses 8000H to CFFFH or back-up RAM addresses D000 to FFFFH (warning: FE00H upwards are registers and FFFFH downwards are stack). Thus a complete picture could be loaded on to the display. Alternately a program could be loaded into back-up RAM at EXT0 (D000H), EXT1 (D800H), EXT2 (E000H), or EXT3 (F000H). The characters 'EXTn' can then be typed in on the keyboard and this will then select the instructions beginning at address EXTn. Thus in-system emulation is easily accomplished.

*EXT0: Executes a program beginning at RAM address D000H. The program must previously have been entered using DMPI selecting D000H as the starting address.

*EXT1: As EXT0 but starts at D800H.

*EXT2: As EXT0 but starts at E000H.

*EXT3: As EXT0 but starts at F000H.

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408