**AN-773**

Application Note

# A CRT TERMINAL USING THE M6800 FAMILY

*Prepared by:*
**Joe Roy and Dusty Morris**
Systems Engineering

This Note describes an M6800-based CRT Controller. A display format of 24 rows of 80 characters is featured. A Motorola M3000 Video Monitor is utilized.

Reprinted from Interface Age, Vol. 2, Issue 2, January, 1977.
Copyrighted January, 1977 by McPheters, Wolfe & Jones. All rights reserved.

**MOTOROLA** *Semiconductor Products Inc.*

# A CRT Terminal Using the M6800 Family

*By Joe Roy and Dusty Morris*

**Data Processing Systems Engineering**
**Motorola Inc., Phoenix, AZ**

This article describes a versatile M6800 based CRT Controller for "glass-teletype," smart, programmable, and intelligent CRT terminals. While a complete duplication of the entire package may be beyond the capabilities of most readers, some of the design features should be of particular utility in other construction projects. Of particular interest is the exploitation of the biphase clock architecture of the M6800 system, providing higher throughput, more I/O handling capability, less interference patterns during refresh memory accesses, and easier task-orientated multiple processor implementation in a CRT terminal than is possible with other approaches. Let's look at the features of this CRT terminal:

1. 24 rows of 80 characters.
2. 7 X 9 uppercase characters in a 9 X 12 dot block; shifted lower case through the use of a custom programmed MCM6832 16K Binary ROM.
3. Conventional non-interlaced raster scan.
4. Blink, half-intensity, video invert, underline, and non-display FACS (Field Attribute Codes); embedded FACS with optional widened memory capability.
5. Alternating inverted/non-inverted cursor.
6. 50/60 Hz field rate is logic selectable; display is centered for both 50 and 60 Hz.
7. Transparent accesses of Refresh Memory by VIA and MPU.
8. Limited graphics implementation with no changes in basic design philosophy.
9. Design philosophy facilitates up-grading a simple economical terminal with upward compatible software and hardware to a task-oriented multiple processor intelligent terminal.
10. MPU is unburdened from overhead of refresh memory contention and is free to service keyboard, edit functions, serial communications, and high speed control such as floppy disk.

The basic configuration for a microprocessor-controlled CRT terminal is shown in Figure 1. An MC6800 microprocessor executes the CRT terminal executive firmware routine and jumps to driver subroutines when servicing the keyboard, serial synchronous or asynchronous interface, floppy disk formatter, and other peripherals. Cursor movements, R/W, and all editing functions are programmable and under microprocessor control. Actual refresh of the CRT monitor display is done with a configuration of SSI/MSI hardware called a VIA (Video Interface Adapter). The VIA provides video, vertical sync, and horizontal sync to the Motorola M3000 (or equivalent) monitor. The monitor must meet the requirements specified in Figure 2.

The MPU and VIA share the CRT Refresh Memory. Since the processor clock is derived from the VIA, both are synchronized. As shown in Figure 1 timing diagram, the VIA accesses memory for CRT refresh during clock phase Ø1, while the MPU accesses memory during Ø2. The Refresh Memory is organized in an odd address block and an even address block. Both an even and the adjacent odd address characters are transferred during a Ø1 access, whereas a single character is transferred to the MPU during a Ø2 access. The "odd/even memory" concept allows characters to be pulled from memory at a rate (≃ 2 MHz) sufficient to update the CRT. The "interleaved clocking" of memory makes it look transparent to both the MPU and the VIA. That is, neither delays the access of the other to memory. Consequently, less MPU overhead results than in other approaches.

Note that the interleaved clocking of memory is unaffected by cycle stretching of either MPU Ø1 or Ø2 . . . techniques used for refreshing dynamic memories, synchronizing other I/O, or interfacing slow memory.

The Refresh Memory provides a bidirectional data bus to the MPU and a two byte output bus for screen refresh. The two bytes of display data are pipelined to even and odd latches which are alternately enabled as data to the address inputs of an MCM6832 character ROM. The address is an ASCII character which the

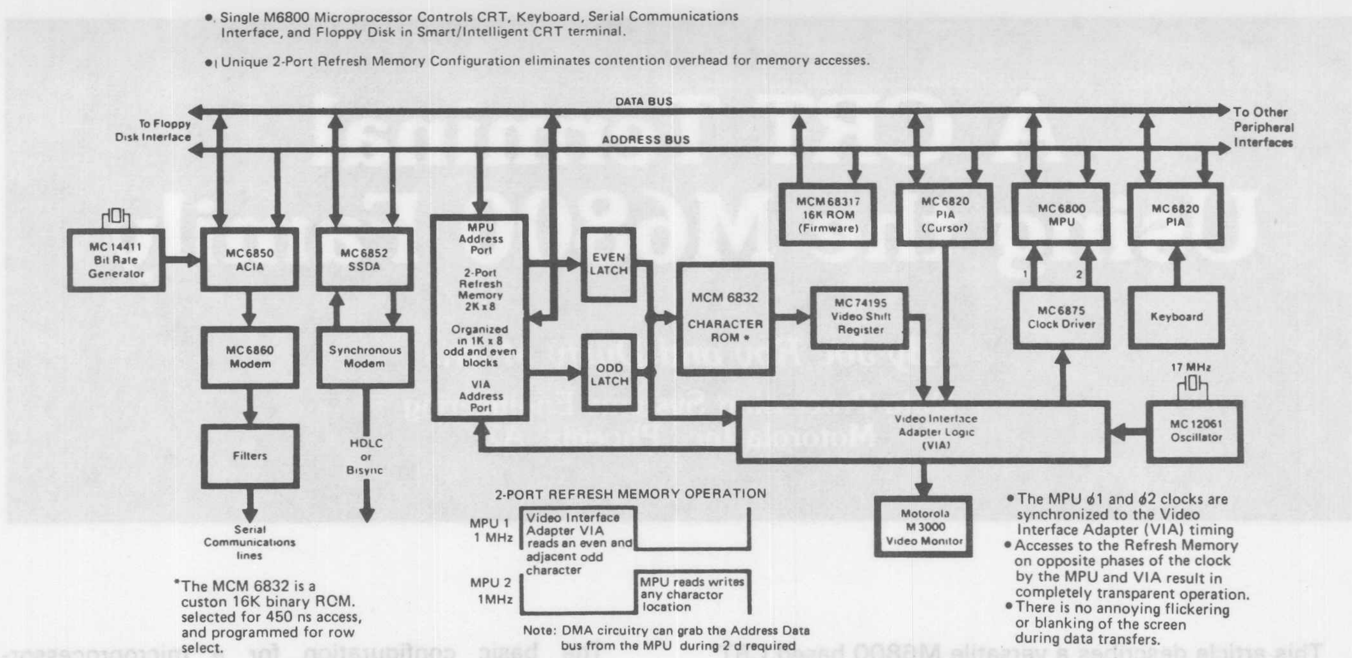



**Figure 1. M6800 Terminal Block Diagram**

| | | | | | |
|---|---|---|---|---|---|
| 1. Character Matrix | (Columns | | 7 | 7 | |
| 2. | (Rows | | 9 | 9 | |
| 3. Character Block | (Columns | | 9 | 9 | |
| 4. | (Rows | | 12 | 12 | |
| 5. Frame (refresh) Rate | | | 50 HZ | 60 HZ | |
| 6. Rows of Characters | | | 24 | 24 | |
| 7. Active Scan Lines (line 6 times line 4) | | | 288 | 288 | |
| 8. Delay before Vertical Sync (No. of scan lines) | | | 1667 µs (31) | 0 | |
| 9. Vertical Sync Width (No. of scan lines) | | 4.52 ms | 215 µs (4) | 215 µs (4) | 1.18 ms |
| 10. Delay after Vertical Sync (No. of scan lines) | | | 2634 µs (49) | 968 µs (18) | |
| 11. Total Scan Lines (Line 7+8+9+10) | | | 372 | 310 | |
| 12. Horizontal Frequency (line rate) (line 11 times line 5) | | | 18.6 KHZ (53.76 µs) | 18.6 KHZ (53.76 µs) | |
| 13. Character Rate (line 12 times line 18) | | | 1.8972 MHZ | 1.8972 MHZ | |
| 14. Characters/Row | | | 80 | 80 | |
| 15. Delay before Horiz. Sync. µsec: | (Character times) | | 2.1 µs (4) | 2.1 µs (4) | |
| 16. Horizontal Sync Width µsec: | (Character times) | | 4.7 µs (9) | 4.7 µs (9) | |
| 17. Delay after Horiz. Sync. µsec: | (Character times) | | 4.7 µs (9) | 4.7 µs (9) | |
| 18. Total character times (line 14+15+16+17) | | | 11.5 µs (102) | 11.5 µs (102) | |
| 19. Clock Rate (line 13 times line 3) | | | 17.074800 MHZ | 17.074800 MHZ | |
| 20. Display size | | | 12" | 12" | |
| 21. Character Time (reciprocal of line 13) | | | 527 ms | 527 ms | |
| 22. MPU Clock (line 19÷2X line 3) | | | 948.6 KHZ | 948.6 KHZ | |
| 23. Clock Time (reciprocal of line19) | | | 58.6 µs | 58.6 µs | |

**Figure 2 Video Monitor Timing (Motorola Display Products M3000 Monitor meets these requirements)**

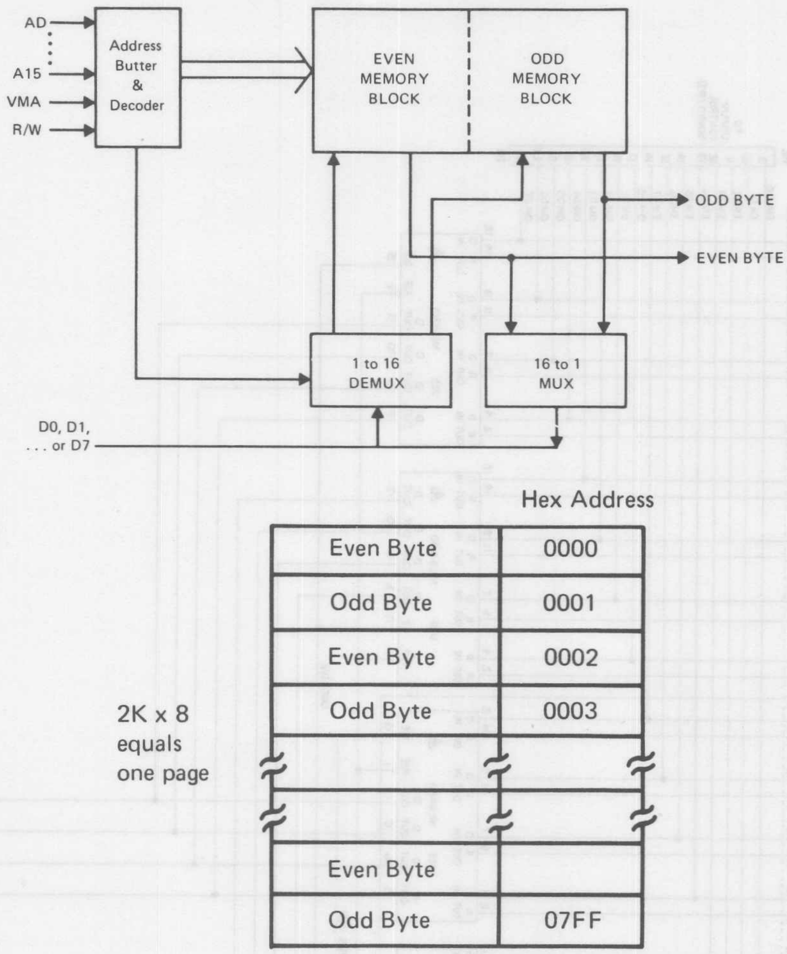



**Figure 3.**

ROM maps into a block of dots. The particular row of dots (0 to 11) in the block is determined by four "row select" inputs from the VIA. In a raster scan system, each ASCII character is presented 12 times to the character ROM with a sequential row select each time in order to paint the complete character on the screen.

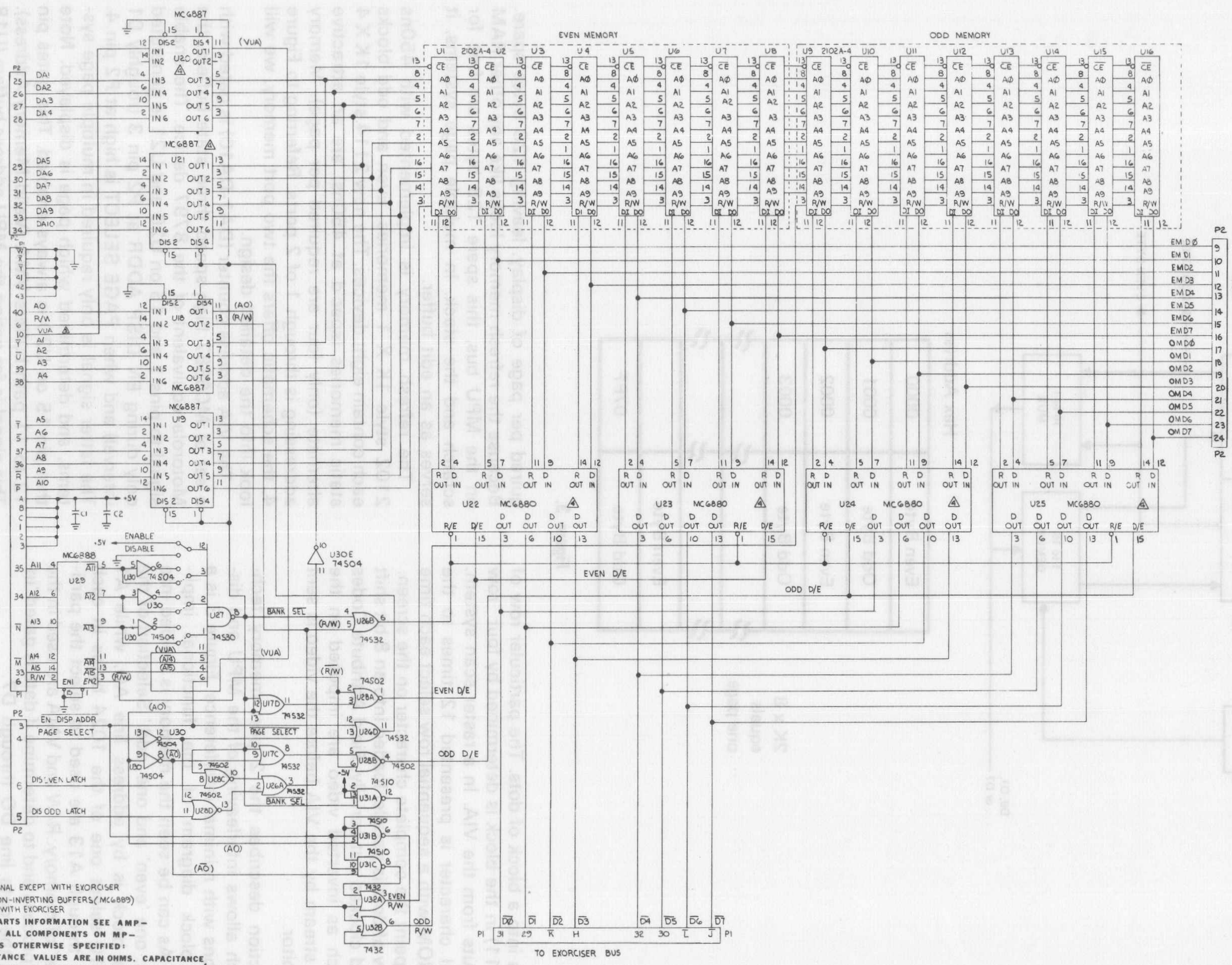
Each row of dots is parallel loaded into an 8 bit shift register and clocked out serially. Field Attribute Codes (FACS) such as inverted video are imposed on the serial data stream by the VIA before the video is sent to the monitor.

This section describes the 2 port memory technique which allows interleaving of the MPU and display functions with minimal interference. Figure 3 is a simplified block diagram of the functional implimentation. As can be seen the memory is divided into two blocks; one 'even' and one 'odd.' Selection of the appropriate block is by address line AO, while A1 through A10 select one of the 1024 bytes in each block. A11 through A13 are used to select the particular "page" of memory. R/W and VMA are used in the read/write process and to determine if data is gated 'in' or 'out' on the data line DO through D7.

As described earlier, the refresh RAM is organized as a 1K X 8 even block and a 1K X 8 odd block. Even and odd blocks are interleaved to form a 2K X 8 "page." Of the 2048 bytes, 1920 (80 times 24) are required per page of display, leaving 128 bytes spare. Because the refresh memory looks like any other RAM on the MPU bus, this spare 128 bytes are free for scratch and the stack. In multiple page systems, it serves as an edit buffer.

The refresh memory is implemented with 450ns 2102 style 1K X 1 memories. Even and odd blocks each contain eight devices. The new 2114 style 1K X 4 static memories (spec'd at 450ns) are an attractive alternate (only four are required per page). Memory addressing is through 1 of 2 ports. Referring to Figure 4, the schematic offers the two port memory, we will look into the detailed design.

The VIA address counter (DA1-DA10) is gated with a set of MC6887 high speed three state buffers, the Motorola equivalent of the 8797 device. The enable signal to pins 1 and 15 on U20 and U21 is generated only during EN DISP ADDR at P2 pin 3 (roughly 01 interval) and when PAGE SELECT is high at P2 pin 4. The latter signal is only required in multiple page systems, and determined which page is displayed. Note that pin 15 on U20 is always enabled. This gates pin 12 to 11 path buffers VMA (Valid Memory Address). This signal is for gating the MPU address buffers (U18 and U19). When used with systems such as the EXORcisor where a block of addresses must be unconditionally protected, this signal should be VUA (Valid Users Address).




⚠ VMA SIGNAL EXCEPT WITH EXORCISER

⚠ USE NON-INVERTING BUFFERS (MC6889) EXCEPT WITH EXORCISER

3. FOR PARTS INFORMATION SEE AMP-

2. MOUNT ALL COMPONENTS ON MP-

1. UNLESS OTHERWISE SPECIFIED: RESISTANCE VALUES ARE IN OHMS. CAPACITANCE VALUES ARE IN MICROFARADS. RESISTORS ARE $\frac{1}{4}$ WATT.

NOTES:

**Figure 4. Display Memory Board**

The selection of U18 and U19 is decoded by U27 which generates BANK Select. The particular bank (1 of 8 pages) is strap selectable (U29 and U30). The ENABLE/DISABLE switch on U27 provides a means of overlaying other chunks of memory with the same address. The other gating for BANK SELECT is VMA (described above), $\overline{\text{A14}}$, $\overline{\text{A15}}$, and $\overline{\text{EN}}$ $\overline{\text{DISP}}$ $\overline{\text{ADDR}}$ (EN MPU or roughly Ø2 interval). AO and R/W from the MPU are gated at all times because pin 15 on U18 is tied to ground.

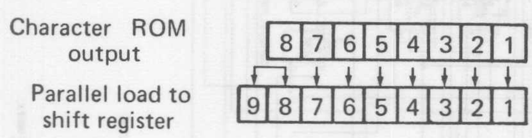
Devices U22 through U25 are MC6880 high speed bidirectional data buffers for multiplexing the Even and Odd Memory bytes into the MPU data bus ($\overline{\text{DO}}$ - $\overline{\text{D7}}$) and vice versa. In systems requiring a non-inverted data bus, MC6880 is replaced with MC6889. During an MPU read, either U22/U23 or U24/U25 is enabled by EVEN D/E or ODD D/E respectively. The buffered LSB of the MPU address, AO, determines which signal is active. During an MPU write, both U22/U23 and U24/U25 are enabled into their respective Even and Odd Memory blocks. The buffered LSB of the MPU address, AO, determines whether EVEN R/W or ODD R/W is active.

The refresh data is an even byte (EMDØ-7) and an odd byte (OMDØ-7).

### Memory for Graphics Applications

Alphanumeric refresh memories are organized on a character basis. Each code stored in memory represents a 7 X 9 pattern in a 9 X 12 dot block. The dot pattern is stored in a character ROM addressed by the character code in the RAM. The repetition of a limited set of symbols on the screen to construct messages makes it possible to use a smaller amount of memory than would be required in a full graphics application where every dot is addressable as a memory location.

A "limited" graphics set of symbols for line drawings and forms is usually implemented with a special character ROM. The nine horizontal dots per character are provided by the ROM. However, most ROMS are organized by eight. It is usually acceptable to get the ninth bit from one of the eight ROM outputs, by parallel load of the 8th bit of ROM into both the eighth and ninth bits of the shift register.




A full graphics capability requires every possible dot on the screen to be stored in memory. Since the pattern is stored directly in RAM, all alphanumeric patterns are generated external to the refresh loop. Accordingly, the character ROM is placed on the MPU bus, and the dual latches drive the shift register directly. As in the alphanumeric controller, the RAM delivers two bytes (even and odd) when addressed by the VIA for refresh. Read and write addressing by the MPU is efficiently handled by bit addressing rather than byte addressing. The complete 64 K address structure of the MC6800 is decoded by hardware; only one of the eight MPU data bits is used for transfers.

A graphics terminal dedicates an MPU to the keyboard and I/O transfers with the refresh memory. All calculations (e.g. vectors), curves, rotations are done in an outboard high speed processor (e.g. microprogrammed bipolar slice). An interface between the MPU and the higher speed processor provides means for control and exchange of input parameters and results.

## DISPLAY CONTROL

The DISPLAY CONTROL consists of circuitry to: sequentially access ASCII characters from the 2-PORT REFRESH MEMORY; generate character row selects; load row patterns into the Parallel-to-Serial Shift Register; serially shift the row pattern through Field Attribute circuits to the monitor as video; provide blanking, horizontal sync, and vertical sync signals; perform cursor compare and generate cursor block.

### Character ROM

The purpose of the Display Control circuitry is to paint the contents of the character ROM at the designated positions on the CRT screen. Custom character ROMS contain 9 X 16 dot matrix patterns for alpha-numeric characters of various domestic and foreign fonts, limited graphics symbols, control characters, or combinations of all from the above. The addresses of the character dot matrix patterns correspond to their ASCII code representation in the case of alphanumeric and control characters. (Assignment is somewhat arbitrary for graphic symbols.)

The particular row of dots in each character dot matrix is selected by four binary row select inputs. Only 12 of the possible 16 rows are utilized in the CRT display being described. The 12 rows are adequate for shifted lower-case characters (g, j, p, q).

Referring to Figure 5 we see that a 16 K binary ROM (e.g. MCM6832) provides 128 possible ASCII characters (only 7 X 12 of the 8 X 16 dot block is used). This is a practical number of characters for most applications. Note the eighth bit of the ASCII code is always available. If it is not used for imbedded Field Attribute Codes (see FAC circuitry description), it can be used to select a second MCM6832 with a different font or graphics. The tri-state capability of the MCM6832 facilitates this mode of operation.

### General Timing

All timing, including MPU Ø1 and Ø2 is derived from an MC12061 crystal oscillator running at the video rate, 17.074800 MHz. The circuit is very stable and always starts from power up. The TTL output of the MC12061 is buffered with a 74S04 before distributing the clock to avoid distortion. The distribution of the clock is critical. Video Clock and $\overline{\text{Video Clock}}$ are fanned out using 74S04 inverters in the same package (for minimum differential propagation delay). Loads are split equally. An alternate distributor is a high speed clock driver with complementary outputs. An important consideration is to keep all 17 MHz logic close together and in proximity to a ground. This will minimize noise and distortion.

There is another method for arriving at 17.074800 MHz. Although it is more expensive, a phase-locked

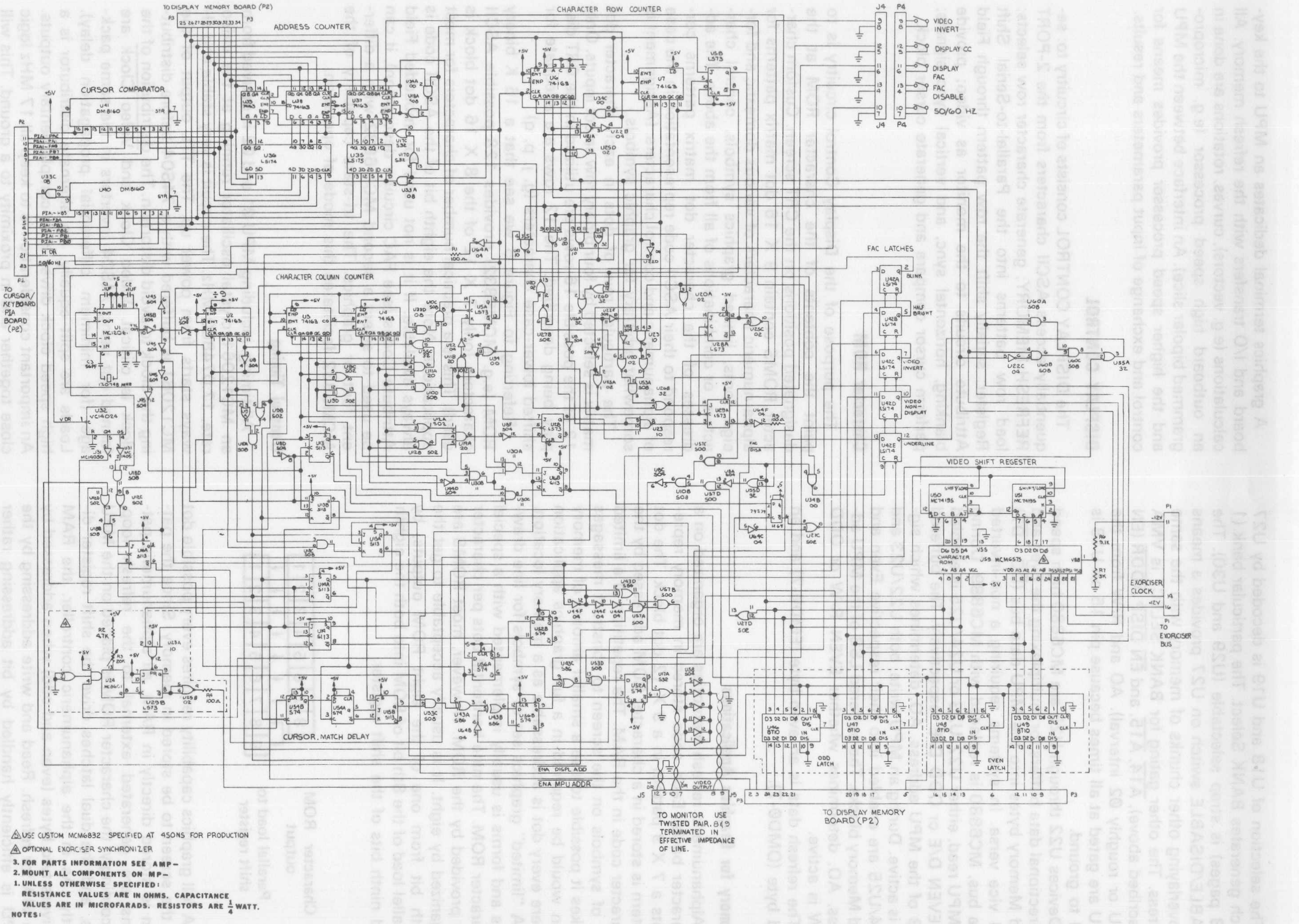



**Figure 5. Schematic — Display Control Board**

loop or phase-locked oscillator is used. The line frequency is the reference and vertical sync the comparison signal. In areas where the line frequency varies radically from nominal, the more expensive system is often required to reduce the visible beat on the CRT screen.

A ÷ 9 counter divides video rate down to a character rate of 1.9 MHz. On/Off decodes from the ÷ 9 counter are resynchronized in 74S113 high speed flip flops. The signals provide the General Timing in Figure 6.

Decodes off the Character Column Counter provide horizontal timing (Figure 7). Vertical timing is from the Character Row Counter (Figure 8).

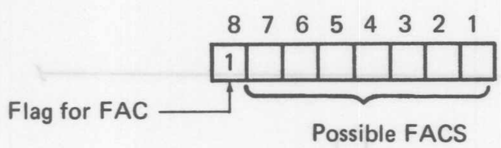
Note the operation of the Address Counter. It must repeat each character address 12 times to paint a complete line of 80 characters on the screen. This requires storing the address of the first character in each line (function of the 74LS latches). Another function of the Address Counter is to advance to address 64 during V blank. (Effectively, this amounts to a start address of 128 since the memory is addressed two bytes at a time.)

Another function of the Display Control Electronics is cursor compare. The contents of the Address Counter are compared with PIA data for coincidence. The DM8160's are exclusive-or comparators. PIA data is a binary address which is manipulated by the MPU for cursor control.

The other circuitry in the Display Control portion generates invert/non-invert cursor block when cursor compare is sensed and furnishes FAC (Field Attribute Code) logic.

### FACS (Field Attribute Codes)

There are two popular methods for handling FACS. In the wide memory method, the memory size is increased by adding bits to each character in memory. Each bit controls a different attribute code for that character. The other method imbeds FAC characters in refresh memory. The eighth bit of the ASCII code is usually decoded as a FAC flag.




When it is a logical one, the other seven bits are latched as FACS. Once latched, the FAC applies to all subsequent characters until another FAC code is decoded. The only exception is at the end of a character line; all FACS are hardware reset. This scheme's advantage is low cost to implement; the disadvantage is the use of a memory location per FAC code. Not only is character density decreased, but the individual characters in a string can not be accented with FACS. It is possible to get around this drawback by stripping FACS from the memory before display, but requires extra hardware and is a programming nightmare. When individually accented characters are required, it is usually better to implement a wider memory.

The wide memory approach to FACS may seem clumsy at first glance . . . the MPU has an 8 bit bus. How are 8 bit MPU transfers done? Construct two pages of memory — a page of ASCII characters 2K X 8, and a page of attributes (2K X 1, 2, 3, 4, . . ., 8). The attribute page is a mask and need only be accessed when the attribute changes or must be read.

For simplicity, the imbedded FAC method was implemented in the CRT terminal. Few changes in Display Control circuitry are required for a wide memory approach.

## CURSOR/KEYBOARD

Referring to Figure 9 we can see how to add a cursor/keyboard interface to this "glass teletype."

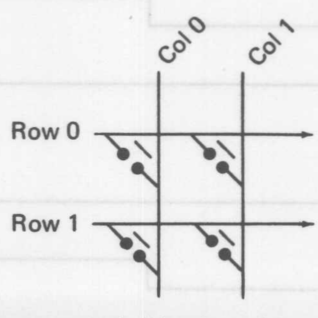
### Cursor

The cursor address is stored as the contents of a PIA. The eight least significant binary bits are PBO - PB7 and the three most significant binary bits are PAO - PA2. There is a one-to-one correspondence between the binary cursor address and a memory location on the screen. The assignment of bits in the PIA is for programming convenience. An STX instruction to the A side of the PIA writes the higher order byte of the index register into the A side of the cursor PIA and the lower order byte into the B side of the cursor PIA (provided the address lines to RSO and RS1 are reversed).

The cursor address is binary rather than X-Y because the binary address manipulations are more frequent. When X-Y addressing is required (e.g. communication interface), a conversion subroutine is called.

### Keyboard

Either a fully-encoded or non-encoded keyboard can be designed. Whichever, a PIA provides the interface to the MPU. In a fully-encoded keyboard, the keyboard hardware generates a strobe and an ASCII encoded character corresponding to the depressed key. All debounce is handled by the keyboard hardware. The strobe pulse applied to the PIA CA or CB inputs causes an interrupt to the MPU. The MPU reads the ASCII character through the PIA and performs the appropriate function. For an alphanumeric character, the MPU writes the data at the present cursor location and increments cursor PIA contents. For control characters, the corresponding commands are executed. (e.g. space, carriage return, insert, delete, etc.)




A non-encoded keyboard is a set of switches wired in a column/row matrix.

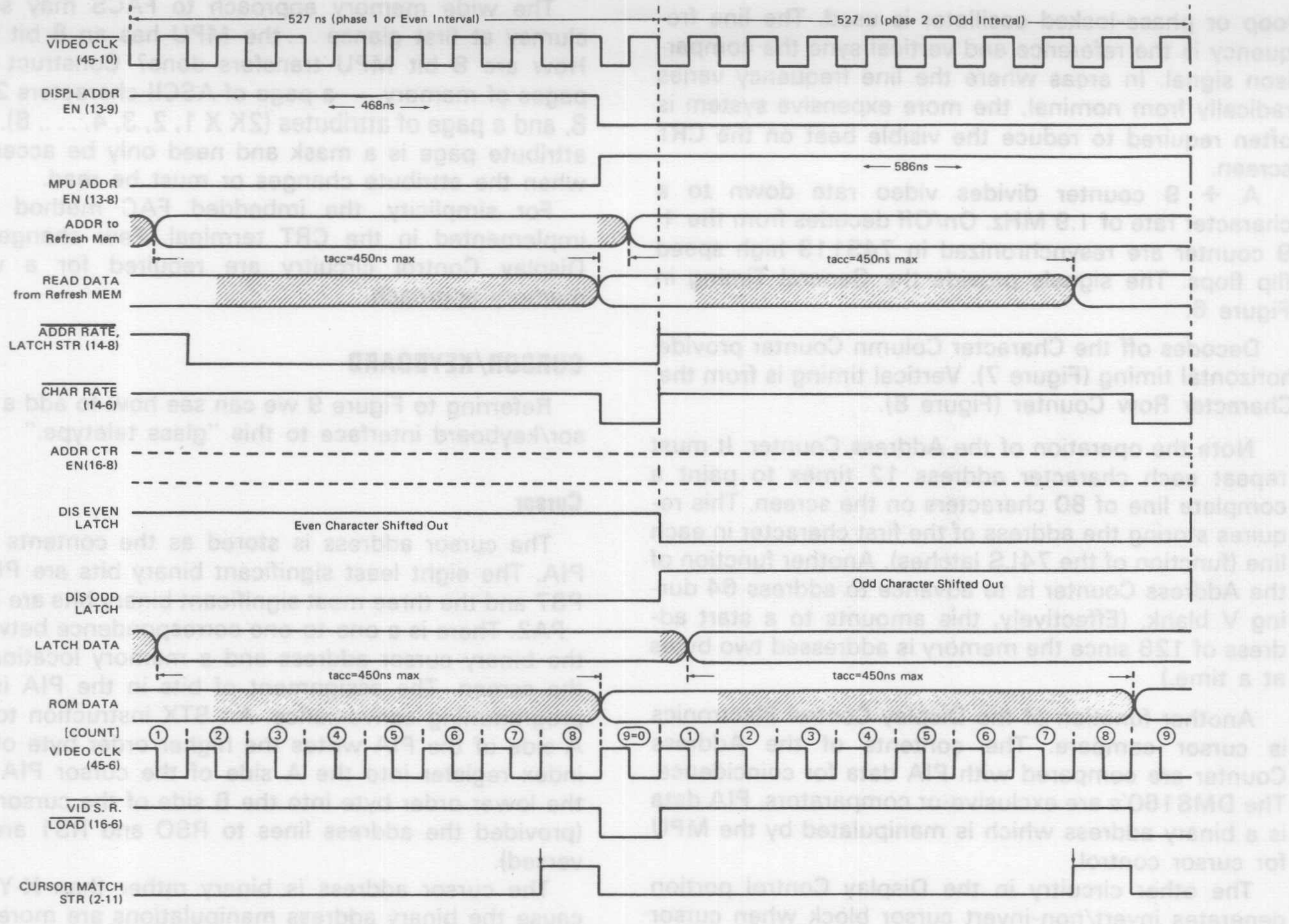



**Figure 6. General Timing**

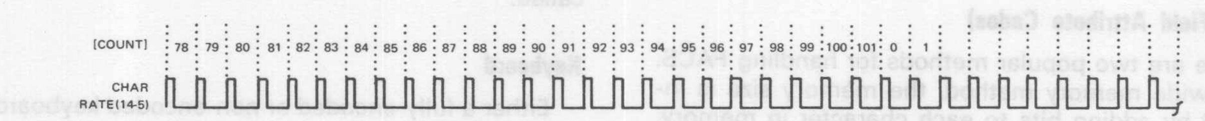



**Figure 7. Horizontal Timing**

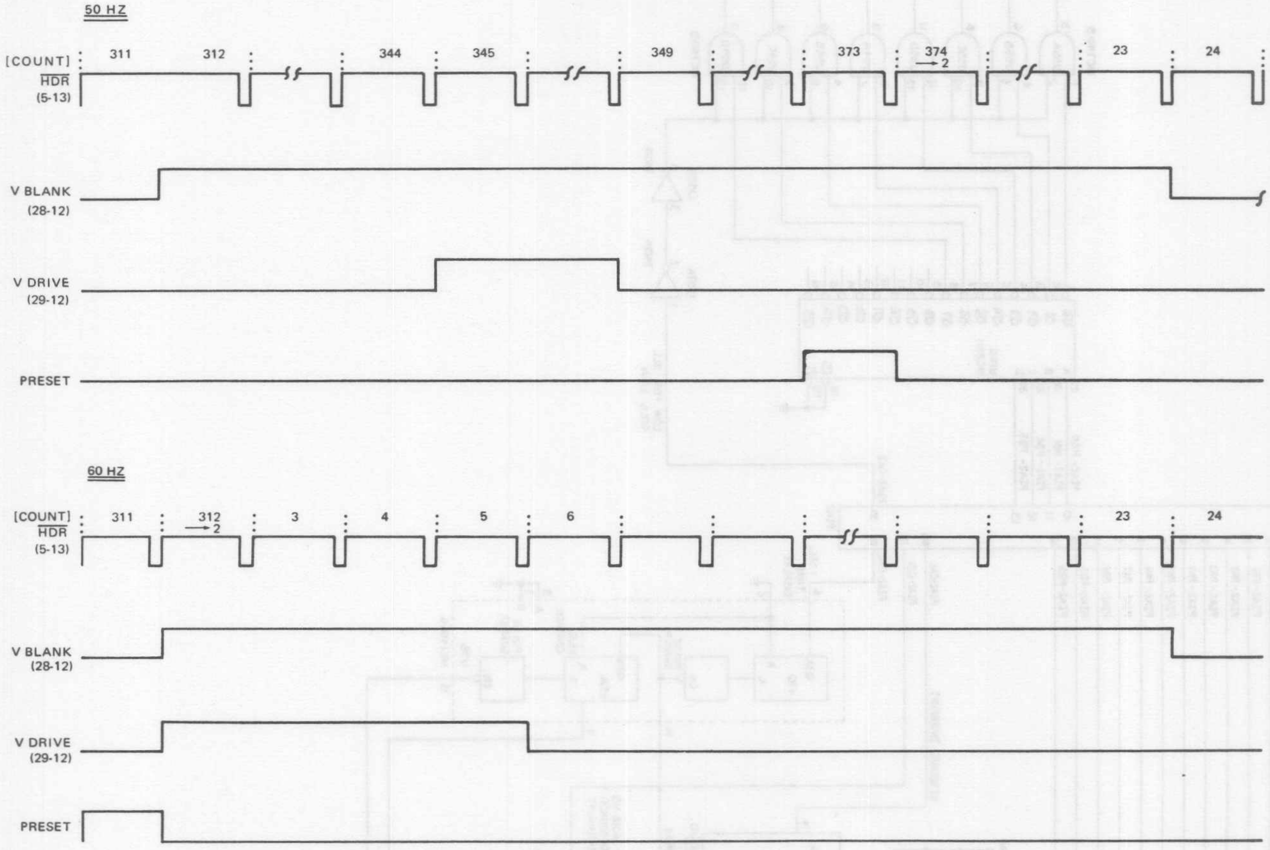
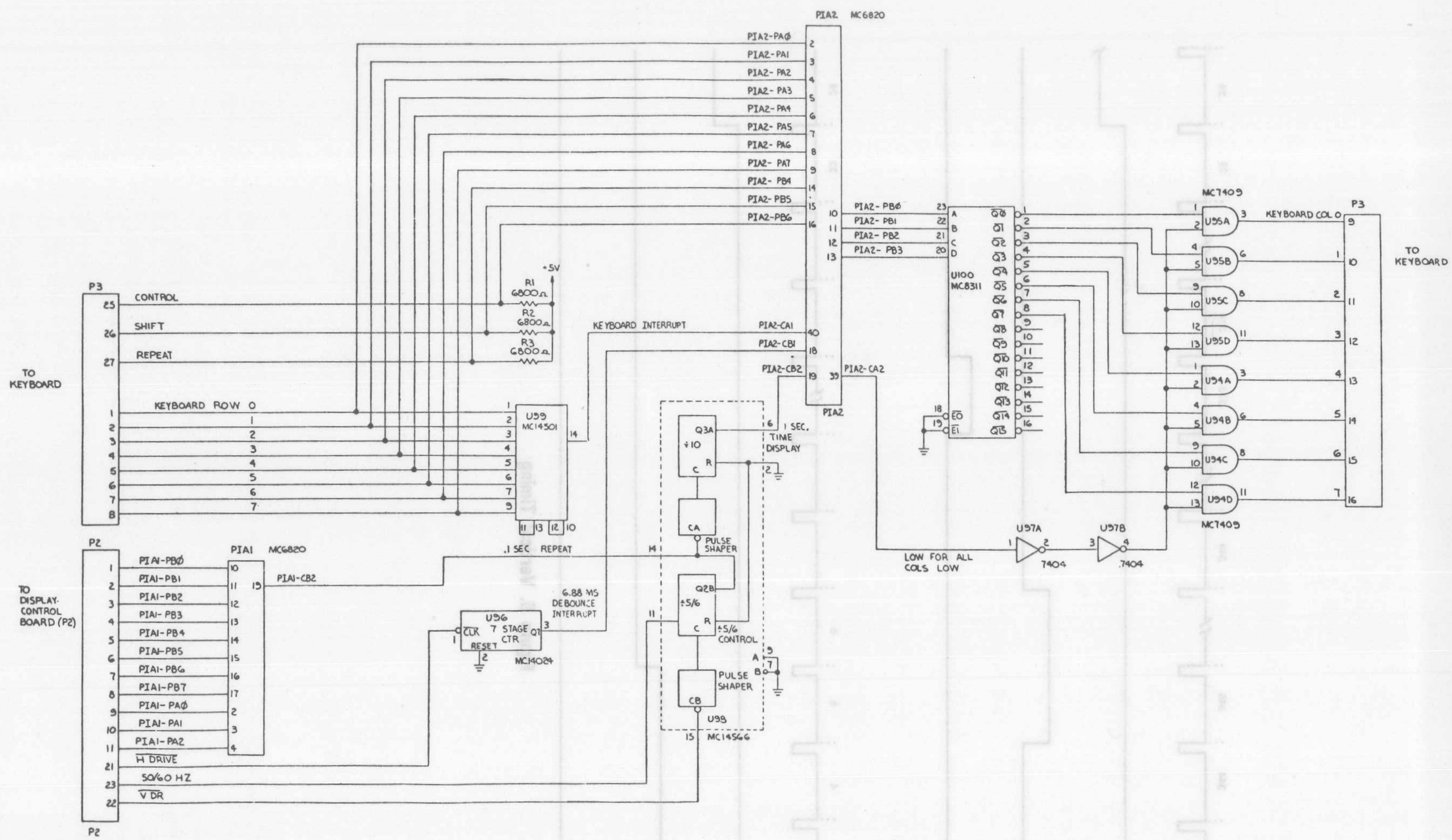



**Figure 8. Vertical Timing**




4. ADDITION TO MEX6820 I/O MODULE
3. FOR PARTS INFORMATION SEE AMP–
2. MOUNT ALL COMPONENTS ON MP–
1. UNLESS OTHERWISE SPECIFIED:
RESISTANCE VALUES ARE IN OHMS. CAPACITANCE VALUES ARE IN MICROFARADS. RESISTORS ARE $\frac{1}{4}$ WATT.
NOTES:

**Figure 9. Cursor/Keyboard PIA Board**

If the columns are scanned one at a time, and the rows read back, the simultaneous depression of any one or two keys can be discerned. For keys, diodes are wired in series with the switch contacts.

The scanning of columns, reading of rows, and switch debounce are under software control. Keys are strategically placed in the matrix so the column and row location easily translate into an ASCII code. Cost savings and flexibility of this non-encoded keyboard versus a fully-encoded keyboard sometimes justifies the additional MPU overhead in a basic CRT terminal.

A non-encoded approach is described here. Referring to the schematic, the keyboard columns are normally held low. When key(s) are depressed, lows appear on the keyboard row inputs. A keyboard interrupt is generated, which starts a scan. Keyboard column outputs are brought low in sequential order. Together with the Shift and Control PIA inputs, the active columns and rows are encoded as an ASCII character and temporarily stored. A 6.88 msec interrupt is provided for debounce and a PIA input for REPEAT (also .1 sec interrupt for this function and a 1 second interrupt for clock functions).

The software for a non-encoded keyboard ranges from simple to quite complex depending on how foolproof the algorithm is.

## Comparison of CRT Terminal Architectures

The central design criteria of a modern CRT terminal is the method used for multiplexing refresh memory between the MPU and the display refresh circuitry. In this section we will discuss time-division multiplexing and priority multiplexing. Throughput versus hardware complexity for different techniques will be analyzed. Schemes which result in missing a character refresh during an MPU transfer are not considered. These techniques result in operator annoyance in many applications (e.g., Key-to-Disk), and are not appropriate for the modern CRT terminal.

### Priority Multiplexing of Refresh Memory

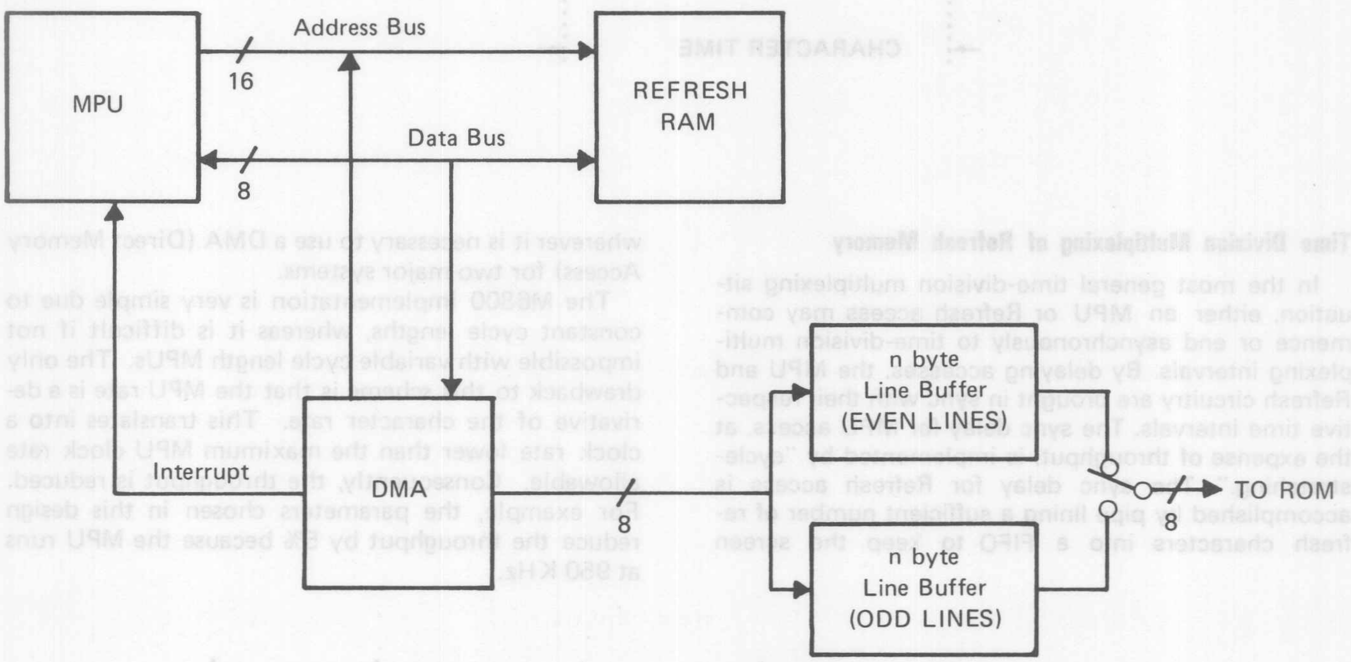
These techniques fall into two general categories:

A. MPU grabs Refresh Memory and locks out Refresh Circuitry.

B. Refresh circuitry grabs Refresh Memory and locks out MPU.

Method A results in zero burden on the MPU; however, a sufficient number of refresh characters must be pipe-lined into FIFO to keep the screen refreshed during an MPU access. The fast memory and complex hardware to accomplish this task is unattractive.

Method B is probably the most widespread technique in use today. The dual line buffer approach is

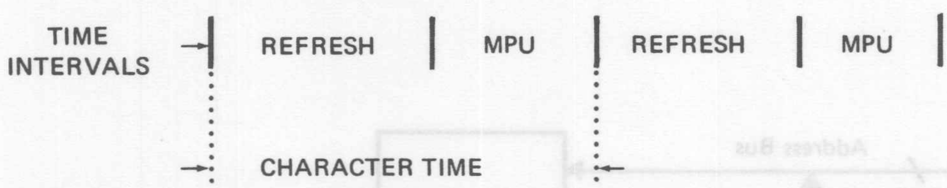



**Dual Line Buffering Technique**

representative of this class. All data for a line on the screen is stored in FIFO's (usually 80 bytes).

While the Odd line buffer is keeping the display refreshed, the Even line buffer is being filled with the next displayed line from Refresh RAM through a DMA channel. The functions are alternately reversed. In an alphanumeric terminal, average MPU burden is between 15 and 30% with peak burden on the MPU of 100% for 80-100 microseconds when loading either line buffer. The MPU is stalled during this transfer and cannot service high speed interfaces (e.g. Floppy Disk). With the addition of extra hardware, the Refresh page of the RAM is isolated from the processor bus during DMA, thus allowing the MPU to continue processing provided it attempts no accesses to memory. In conclusion, the dual-line buffers and DMA configuration is more expensive, has higher parts count, and imposes a severe burden on the MPU compared to the time-division multiplexed technique.

Furthermore, the DMA approach is very inefficient for full graphics, since there are no recirculations of the line buffer as in an alphanumeric display.




## Time Division Multiplexing of Refresh Memory

In the most general time-division multiplexing situation, either an MPU or Refresh access may commence or end asynchronously to time-division multiplexing intervals. By delaying accesses, the MPU and Refresh circuitry are brought in sync with their respective time intervals. The sync delay for MPU access, at the expense of throughput, is implemented by "cycle-stretching." The sync delay for Refresh access is accomplished by pipe lining a sufficient number of refresh characters into a FIFO to keep the screen refreshed during an MPU synchronization delay. The FIFO is typically two bytes deep. The disadvantage of this technique is the requirement for fast memory. It is considerably simplified if the time-division multiplexing is made synchronous with character rate. Sync delay is no longer required for refresh and the FIFO circuitry is eliminated (a single byte latch is still required). Also memory speed is reduced. This simplification is not always acceptable; e.g. it may result in excessive MPU "cycle-stretching" at slow character rates.

The technique for sharing CRT Refresh Memory described in this article is a special case of time-division multiplexing. The access intervals for Refresh and MPU are $\phi 1$ and $\phi 2$, respectively.

As $\phi 1$ and $\phi 2$ clocking signals are outputted even during wait states, no matter how long this MPU is forced into a wait state the screen will remain refreshed. Since there is no overlap of accesses, no contention circuitry is required and the MPU burden is zero at all times. While you may not want or need to duplicate the entire circuitry described in this article, the bi-phase memory access technique may be used wherever it is necessary to use a DMA (Direct Memory Access) for two major systems.

The M6800 implementation is very simple due to constant cycle lengths, whereas it is difficult if not impossible with variable cycle length MPUs. The only drawback to this scheme is that the MPU rate is a derivative of the character rate. This translates into a clock rate lower than the maximum MPU clock rate allowable. Consequently, the throughput is reduced. For example, the parameters chosen in this design reduce the throughput by 5% because the MPU runs at 950 KHz.