HEWLETT PACKARD

# APPLICATION NOTE 1001

# Interfacing the HDSP-2000 to Microprocessor Systems

## INTRODUCTION

Over the past two years, the need for alphanumeric displays has grown very rapidly due to the extensive use of microprocessors in new systems design. The presence of the microprocessor in such systems substantially simplifies the traditionally difficult task of designing an alphanumeric display into a system. This task is further simplified by using a display element such as the HDSP-2000 which has in one package a four character display, as well as most of the basic electronics necessary to drive the display. Depending upon overall systems configuration, microprocessor time available to dedicate to display support, and the type of information to be displayed, one may choose several different partitioning schemes to drive such a display.

This note will deal with four different techniques (see Figure 1) for interfacing the HDSP-2000 display to microprocessor systems:

1. The REFRESH CONTROLLER interrupts the microprocessor at a 500 Hz rate to request refresh data for the display.
2. The DECODED DATA CONTROLLER accepts 5 x 7 matrix data from the microprocessor and then automatically refreshes the display with the same information until new data is supplied by the microprocessor.
3. The RAM CONTROLLER accepts ASCII data and interfaces like a RAM to the microprocessor.
4. The DISPLAY PROCESSOR CONTROLLER (HDSP-247X series) employs a dedicated single chip microprocessor as a data display/control/keyboard interface which has many of the features of a complete terminal.

The interface techniques depicted are specifically for the 8080A or 6800 microprocessor families. Extension of these techniques to other processors should be a relatively simple software chore with little or no hardware changes required.

## COMPARISON OF INTERFACE TECHNIQUES

The choice of a particular interface is an important consideration because it affects the design of the entire microprocessor system. The REFRESH CONTROLLER provides the lowest cost interface because it uses the microprocessor to provide ASCII decoding and display strobing. Because the ASCII decoder is located within the microprocessor system, the designer has total control over the display font within the program. This feature is particularly important when the system will be used to display different languages and special graphic symbols. However, the REFRESH CONTROLLER requires a significant amount of microprocessor time. Furthermore, while the interrupt allows the refresh program to operate asynchronously from the main program, this technique limits some of the software techniques that can be used in the main program.

The DECODED DATA CONTROLLER requires microprocessor interaction only when the display message is changed. Like the REFRESH CONTROLLER, the ASCII decoder is located within the microprocessor program. However, the time required to decode the ASCII string and store the resulting 5 x 7 display data into the interface requires several milliseconds of microprocessor time.

The RAM CONTROLLER also requires interaction from the microprocessor system only when the display message is changed. Because the ASCII decoder is located within the display interface, the microprocessor requires much less time to load a new message into the display.

The DISPLAY PROCESSOR CONTROLLER, the HDSP-247X series, is the most powerful interface. The software within the DISPLAY PROCESSOR CONTROLLER further reduces the microprocessor interaction by providing more powerful left and right data entry modes compared to the RAM entry mode of the DECODED DATA and RAM CONTROLLERS. The DISPLAY PROCESSOR CONTROLLER can also provide features such as a Blinking Cursor, Editing Commands, and a Data Out function. One version of the DISPLAY PROCESSOR CONTROLLER allows the user to provide a custom ASCII decoder for applications needing a special character font.

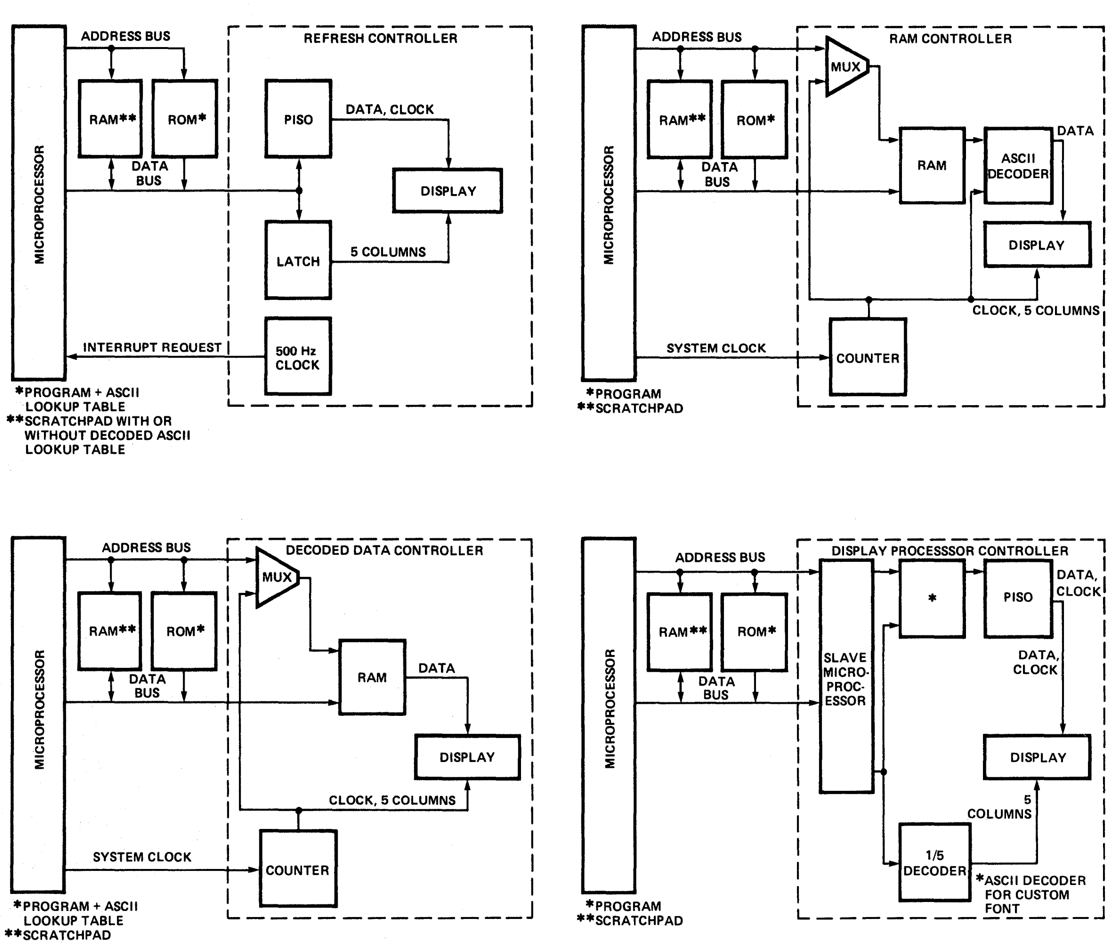
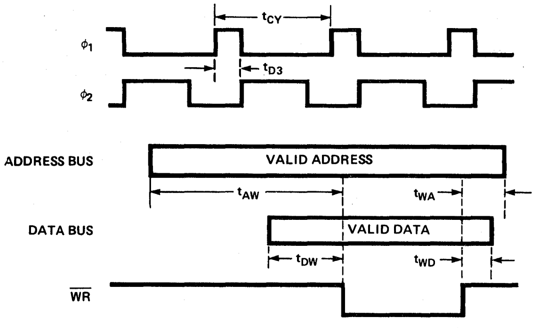


**Figure 1. Four Different Techniques to Interface the HDSP-2000 Alphanumeric Display to a Microprocessor System**

## MICROPROCESSOR OVERVIEW

In order to effectively utilize the interface techniques listed above, an understanding of microprocessor fundamentals is required. A microprocessor system usually consists of a microprocessor, ROM memory, RAM memory, and some specific I/O interface. The microprocessor performs the desired system function by executing a program stored within the ROM. The RAM memory is used to provide a stack for the microprocessor, as well as a temporary scratchpad memory. The I/O interface consists of circuitry that is used as an input to the system as well as an output from the system. The alphanumeric display subsystem would be considered part of this interface. The microprocessor interfaces to this system through an Address Bus, a Data Bus, and a Control Bus. The Address Bus consists of several outputs from the microprocessor ($A_0$, $A_1$...$A_n$) which collectively specify a binary number. This number or "address" uniquely specifies each word in the ROM memory, RAM memory, and I/O interface. The Data Bus consists of several lines from the microprocessor which are used both as inputs and outputs. The Data Bus serves as an input during a memory or I/O read operation and as an output for a memory or I/O write operation. The Control Bus provides the required signals and timing to the rest of the microprocessor system to distinguish a memory read from a memory write, and in some systems an I/O read from an I/O write. These control lines and the timing between the Address, Data, and Control Buses vary for different microprocessors.

For the 8080A microprocessor, the Address Bus consists of 16 lines, the Data Bus consists of 8 lines, and the Control Bus consists of several lines including DBIN (Data Bus In), $\overline{\text{WR}}$ (Write), and clock signals $\phi_1$ and $\phi_2$. DBIN and $\overline{\text{WR}}$ are used to specify a memory read or write. The 8080A microprocessor provides several other control lines which are usually decoded with DBIN and $\overline{\text{WR}}$ to generate composite control signals $\overline{\text{MEM R}}$ (Memory Read), $\overline{\text{MEM W}}$ (Memory Write), $\overline{\text{I/O R}}$ (I/O Read), and $\overline{\text{I/O W}}$ (I/O Write). Since the alphanumeric display subsystem is an output of the microprocessor system, the timing between the Address Bus, Data Bus, and $\overline{WR}$ is of particular significance. This timing is generalized in Figure 2.



| 8080 MICROPROCESSOR WITH 8228 CLOCK | MINIMUM TIMES (ns) | | | |
|---|---|---|---|---|
| | $t_{AW}$ | $t_{WA}$ | $t_{DW}$ | $t_{WD}$ |
| 8080A, $t_{CY}$ = 480 | 740 | 90 | 230 | 90 |
| 8080A-2, $t_{CY}$ = 380 | 560 | 80 | 140 | 80 |
| 8080A-1, $t_{CY}$ = 320 | 470 | 70 | 110 | 70 |

$t_{AW} = 2t_{CY} - t_{D3} - [140(A), 130(A\text{-}2), 110(A\text{-}1)]$

$t_{WA} = t_{WD} = t_{D3} + 10$

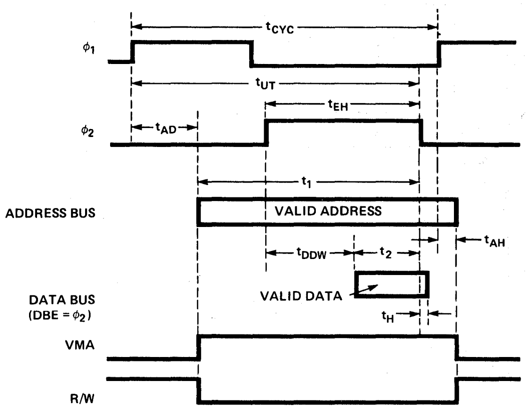
$t_{DW} = t_{CY} - t_{D3} - [170(A), 170(A\text{-}2), 150(A\text{-}1)]$

From INTEL Component Data Catalog, 1978

**Figure 2. Memory Write Timing for the Intel 8080A Microprocessor Family**

The 6800 microprocessor has a 16 line Address Bus, 8 line Data Bus, and a Control Bus that includes the signals VMA (Valid Memory Address), R/W (Read/Write), DBE (Data Bus Enable), and clock signals $\phi_1$ and $\phi_2$. R/W specifies either a memory read or write while VMA is used in conjunction with R/W to specify a Valid Memory Address. DBE gates the internal data bus of 6800 into the Data Bus. In many applications, DBE is connected to $\phi_2$. The timing between the Address Bus, Data Bus, VMA, and R/W (when DBE = $\phi_2$) is shown in Figure 3. Additional data hold time, $t_H$, can be achieved by delaying $\phi_2$ to the microprocessor or by extending DBE beyond the falling edge of $\phi_2$.

The ASCII to 5 x 7 dot matrix decoder used by the REFRESH CONTROLLER and DECODED DATA CONTROLLER is located within the microprocessor program. This decoder requires 640 bytes of storage to decode the 128 character ASCII set. The decoder used by these controllers is formatted so that the first 128 bytes contain column 1 information; the next 128 bytes contain column 2 information, etc. Each byte of this decoder is formatted such that $D_6$ through $D_0$ contain Row 7 through Row 1 display data respectively. The data is coded so that a HIGH bit would turn the corresponding 5 x 7 display dot ON. This decoder table is shown in Figure 20. The resulting 5 x 7 dot matrix display font is shown in the HDSP-2471 data sheet.



| 6800 MICROPROCESSOR | MINIMUM TIMES (ns) | | | |
|---|---|---|---|---|
| | $t_1$ | $t_{AH}$ | $t_2$ | $t_H$ |
| 6800, $t_{CYC}$ = 1000 | 630 | 30 | 225 | 10 |
| 68A00, $t_{CYC}$ = 666 | 420 | 30 | 80 | 10 |
| 68B00, $t_{CYC}$ = 500 | 290 | 30 | 60 | 10 |

$t_1(\text{MIN}) = t_{UT}(\text{MIN}) - t_{AD}(\text{MAX})$

$t_2(\text{MIN}) = t_{EH}(\text{MIN}) - t_{DDW}(\text{MAX})$

From MOTOROLA Semiconductor MC6800 Data Sheet (DS9471), 1978

**Figure 3. Memory Write Timing for the Motorola 6800 Microprocessor Family**

## REFRESH CONTROLLER

The REFRESH CONTROLLER circuit depicted in Figure 4 is designed for interface to either 6800 or 8080A microprocessors. This circuit operates by interrupting the microprocessor every two milliseconds to request a new block of display data and column select data. Display data is loaded from the data bus into the serial input of the HDSP-2000 via a 74165 parallel in, serial out shift register. The 74LS293 counter and associated gates insure that only seven clock pulses are delivered to the shift register and the HDSP-2000 for each word loaded. Column Select data is loaded into a 74174 latch which, in turn, drives the column switch transistors. The circuit timing relative to the microprocessor clock and I/O is depicted in Figure 5.

The 6800 software necessary to support this interface is divided into two separate subroutines, "RFRSH" and "LOAD" (Figure 6). This approach is desirable to minimize microprocessor involvement during display refresh. The subroutine "RFRSH" loads a new set of decoded display data from the microprocessor scratchpad memory into the interface at each interrupt request. The subroutine "LOAD" is utilized to decode a string of 32 ASCII characters into 5 x 7 formatted display data and store this data in the scratchpad memory used by "RFRSH".

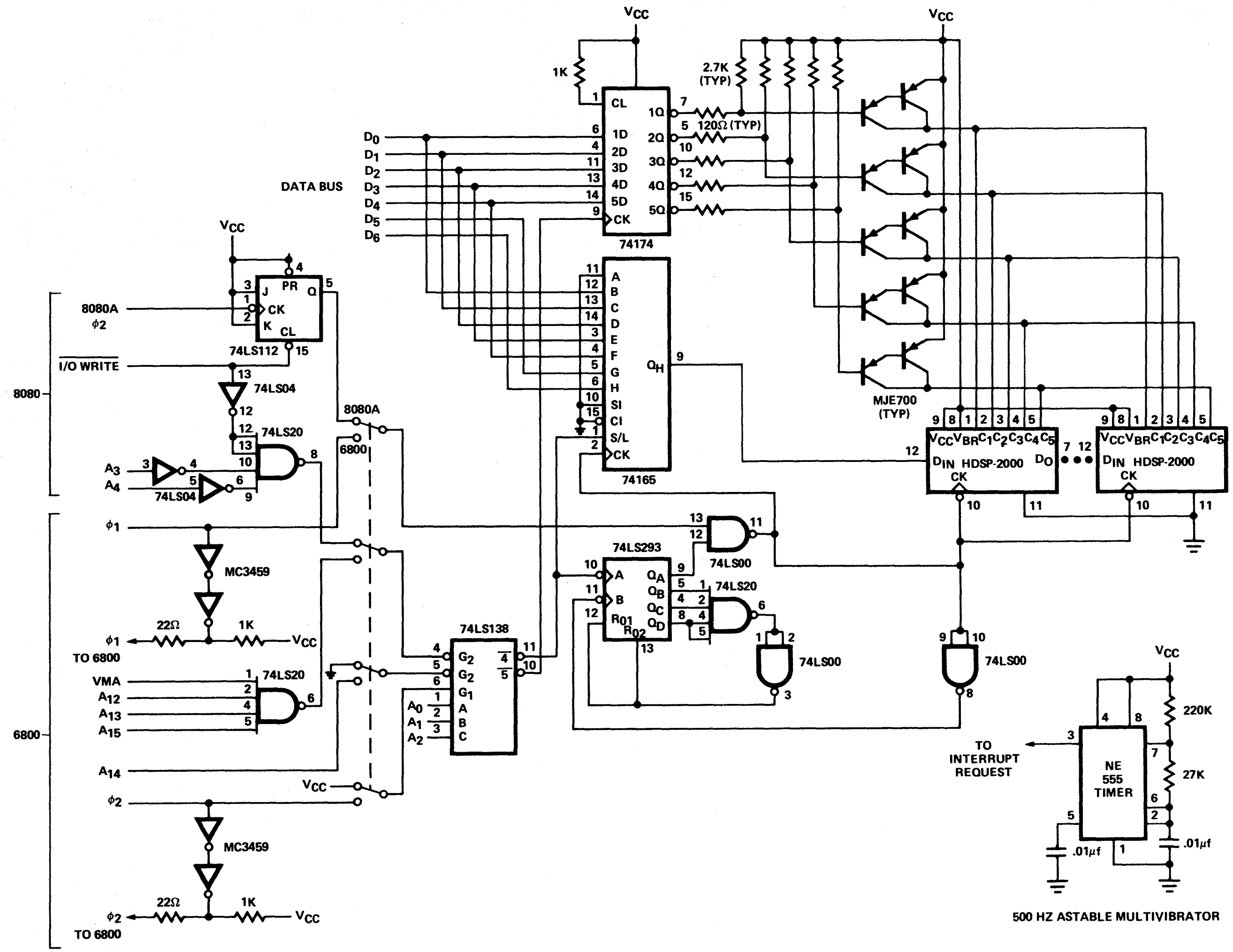


**Figure 4. 6800 or 8080A Microprocessor Interface to the HDSP-2000 REFRESH CONTROLLER**

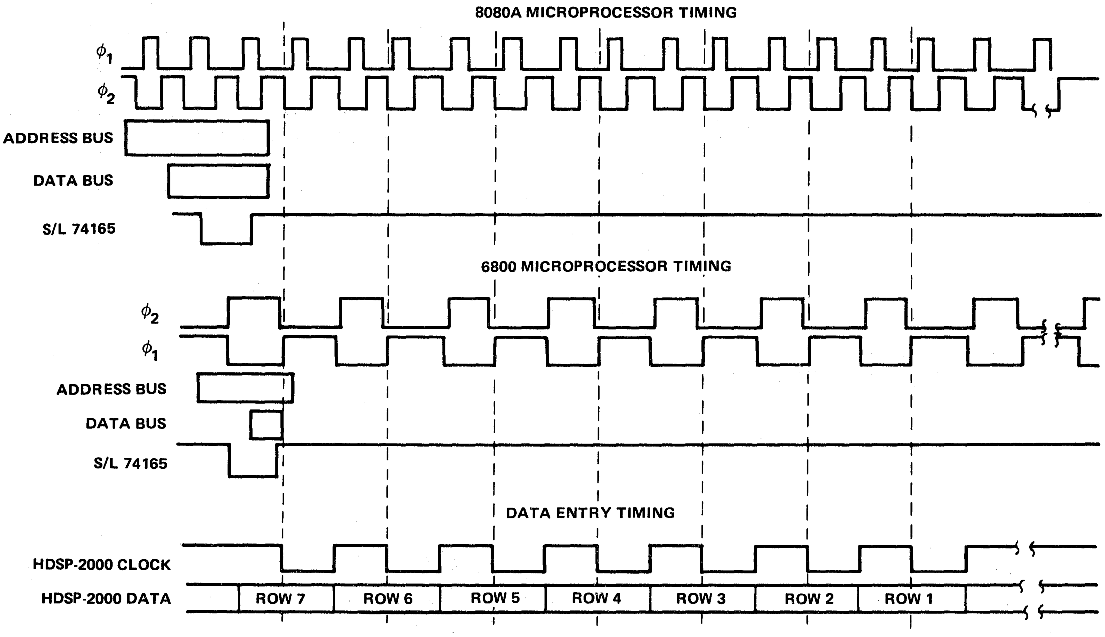


**Figure 5. REFRESH CONTROLLER Timing**

Figures 7a and 7b depict two different software routines for interfacing the REFRESH CONTROLLER to an 8080A microprocessor. The two subroutines shown in Figure 7a are functional replacements for the 6800 program shown in Figure 6. The programs shown in Figures 6 and 7a require a 5n byte scratchpad memory where n is the display length. The routine in Figure 7b eliminates this scratchpad memory by decoding and loading data each time a new interrupt request is received.

Because the microprocessor system is interrupted every 2ms, proper software design is especially important for the REFRESH CONTROLLER. The use of the scratchpad memory significantly reduces the time required to refresh the display. The fastest program, shown in Figure 6, uses in-line code to access data from the buffer and output it to the display. This program requires 3.7% + .50n% of the available microprocessor time for a 1MHz clock. The program shown in Figure 7a is similar to the one shown in Figure 6, except that it uses a program loop instead of the in-line code. This program uses 5.4% + .93n% of the microprocessor time for a 2MHz clock. These programs utilize a subroutine "LOAD" which is called whenever the display message is changed. This subroutine executes in 10.2ms and 7.5ms respectively for Figure 6 and Figure 7a. The program in Figure 7b uses 7.6% + 1.35n% of the microprocessor time for a 2MHz clock. A 50% reduction in the previously described microprocessor times can be achieved by using faster versions of the 6800 and 8080A microprocessors.

```
        OBJECT
LOC      CODE       SOURCE STATEMENTS

                    *
                    *
        BF 05       CDVR      EQU     $BF05
        BF 04       RDVR      EQU     $BF04
        06 00       DECDR     EQU     $0600
0000                POINT     RMB     2
0002                COLMN     RMB     1
0003                COUNT     RMB     2

0005    00 AD       ASCII     FDB     DATA
0007                DISPNT    RMB     2
0009                DCRPNT    RMB     2
000B                COLCNT    RMB     1
000C                DIGCNT    RMB     1

000D                BUFFR     RMB     160
00AD                DATA      RMB     32

0400                          ORG     $0400
0400    86 FF       RFRSH     LDA A   I, $FF
0402    B7 BF 05              STA A   E, CDVR
0405    DE 00                 LDX     D, POINT
0407    A6 00       LOOPHH    LDA A   X, 0
0409    B7 BF 04              STA A   E, RDVR
040C    A6 01                 LDA A   X, 1
040E    B7 BF 04              STA A   E, RDVR
                                  •
                                  •
                                  •
04A2    A6 1F                 LDA A   X, 31
04A4    B7 BF 04              STA A   E, RDVR
04A7    96 02                 LDA A   D, COLMN
04A9    B7 BF 05              STA A   E, CDVR
04AC    81 EF                 CMP A   I, $EF
04AE    27 10                 BEQ     LOOPB
04B0    D6 00                 LDA B   D, POINT +1
04B2    CB 20                 ADD B   I, 32
04B4    D7 00                 STA B   D, POINT +1
04B6    24 03                 BCC     LOOPA
04B8    7C 00 00              INC     E, POINT
04BB    0D          LOOPA     SEC
04BC    79 00 02              ROL     E, COLMN
04BF    3B                    RTI
04C0    CE 00 0D    LOOPB     LDX     I, BUFFR
04C3    DF 00                 STX     D, POINT
04C5    DE 03                 LDX     D, COUNT
04C7    09                    DEX
04C8    DF 03                 STX     D, COUNT
04CA    86 FE                 LDA A   I, $FE
04CC    97 02                 STA A   D, COLMN
04CE    3B                    RTI

04CF    5F          LOAD      CLR B
04D0    CE 00 0D              LDX     I, BUFFR
04D3    DF 07                 STX     D, DISPNT
04D5    86 06                 LDA A   I, <DECDR
04D7    97 09                 STA A   D, DCRPNT
04D9    86 05                 LDA A   I, 5
04DB    97 0B                 STA A   D, COLCNT
04DD    86 20       LOOP1     LDA A   I, 32
04DF    97 0C                 STA A   D, DIGCNT
04E1    9B 06                 ADD A   D, ASCII+1
04E3    24 03                 BCC     LOOP2
04E5    7C 00 05              INC     E, ASCII
04E8    97 06       LOOP2     STA A   D, ASCII+1
04EA    DE 05       LOOP3     LDX     D, ASCII
04EC    09                    DEX
04ED    A6 00                 LDA A   X, 0
04EF    DF 05                 STX     D, ASCII
04F1    1B                    ABA
04F2    97 0A                 STA A   D, DCRPNT+1
04F4    DE 09                 LDX     D, DCRPNT
04F6    A6 00                 LDA A   X, 0
04F8    DE 07                 LDX     D, DISPNT
04FA    A7 00                 STA A   X, 0
04FC    08                    INX
04FD    DF 07                 STX     D, DISPNT
04FF    7A 00 0C              DEC     E, DIGCNT
0502    26 E6                 BNE     LOOP3
0504    CB 80                 ADD B   I, $80
0506    24 03                 BCC     LOOP4
0508    7C 00 09              INC     E, DCRPNT
050B    7A 00 0B    LOOP4     DEC     E, COLCNT
050E    26 CD                 BNE     LOOP1
0510    39                    RTS
```

**Figure 6. 6800 Microprocessor Program Utilizing a 160 Byte RAM Buffer that Interfaces to the REFRESH CONTROLLER**

```
        OBJECT
LOC      CODE       SOURCE STATEMENTS

0004                RDVR      EQU     0004H
0005                CDVR      EQU     0005H
E500                DECDR     EQU     0E500H

                              ORG     0E000H
E000    05 E0       POINT     DW      BUFFR
E002    FE          COLMN     DB      0FEH
E003    FF FF       COUNT     DW      0FFFFH
E005    00          BUFFR     DS      160

                              ORG     0E0A5H
E0A5    A7 E0       ASCII     DW      DATA
E0A7    00          DATA      DS      32

                              ORG     0E400H
E400    F5          RFRSH     PUSH    PSW
E401    C5                    PUSH    B
E402    E5                    PUSH    H
E403    2A 00 E0              LHLD    POINT
E406    06 20                 MVI     B, 32
E408    3E FF                 MVI     A, 0FFH
E40A    D3 05                 OUT     CDVR
F40C    7E          LOOP      MOV     A, M
E40D    D3 04                 OUT     RDVR
E40F    23                    INX     H
E410    05                    DCR     B
E411    C2 0C E4              JNZ     LOOP
E414    3A 02 E0              LDA     COLMN
E417    D3 05                 OUT     CDVR
E419    FE EF                 CPI     0EFH
E41B    CA 28 E4              JZ      FIRST
E41E    22 00 E0              SHLD    POINT
E421    07                    RLC
E422    32 02 E0              STA     COLMN
E425    C3 3A E4              JMP     END
E428    21 05 E0    FIRST     LXI     H, BUFFR
E42B    22 00 E0              SHLD    POINT
E42E    3E FE                 MVI     A, 0FEH
E430    32 02 E0              STA     COLMN
E433    2A 03 E0              LHLD    COUNT
E436    2B                    DCX     H
E437    22 03 E0              SHLD    COUNT
E43A    E1          END       POP     H
E43B    C1                    POP     B
E43C    F1                    POP     PSW
E43D    C9                    RET

E43E    11 24 E0    LOAD      LXI     D, BUFFR+31
E441    0E 20                 MVI     C, 32
E443    2A A5 E0    LOOP1     LHLD    ASCII
E446    7E                    MOV     A, M
E447    23                    INX     H
E448    22 A5 E0              SHLD    ASCII
E44B    26 E5                 MVI     H, DECDR/256
E44D    6F                    MOV     L, A
E44E    06 05                 MVI     B, 5
E450    7E          LOOP2     MOV     A, M
E451    12                    STAX    D
E452    7D                    MOV     A, L
E453    C6 80                 ADI     80H
E455    6F                    MOV     L, A
E456    D2 5A E4              JNC     LOOP3
E459    24                    INR     H
E45A    7B          LOOP3     MOV     A, E
E45B    C6 20                 ADI     32
E45D    5F                    MOV     E, A
E45E    05                    DCR     B
E45F    C2 50 E4              JNZ     LOOP2
E462    7B                    MOV     A, E
E463    C6 5F                 ADI     5FH
E465    5F                    MOV     E, A
E466    0D                    DCR     C
E467    C2 43 E4              JNZ     LOOP1
E46A    C9                    RET
```

**Figure 7a. 8080A Microprocessor Program Utilizing a 160 Byte RAM Buffer that Interfaces to the REFRESH CONTROLLER**

```
        OBJECT
LOC      CODE      SOURCE STATEMENTS

0004               RDVR    EQU    0004H
0005               CDVR    EQU    0005H
E500               DECDR   EQU    0E500H

                           ORG    0E000H
E000  07  E0       ASCII   DW     DATA
E002  FE           COLMN   DB     0FEH
E003  FF  FF       COUNT   DW     0FFFFH
E005  00  E5       BASE    DW     DECDR
E007  00           DATA    DS     32

                   ORG            0E400H
E400  F5           RFRSH   PUSH   PSW
E401  C5                   PUSH   B
E402  D5                   PUSH   D
E403  E5                   PUSH   H
E404  2A  05  E0           LHLD   BASE
E407  EB                   XCHG
E408  2A  00  E0           LHLD   ASCII
E40B  01  1F  00           LXI    B, 31
E40E  09                   DAD    B
E40F  43                   MOV    B, E
E410  0E  20               MVI    C, 32
E412  3E  FF               MVI    A, 0FFH
E414  D3  05               OUT    CDVR
E416  78           LOOP    MOV    A, B
E417  86                   ADD    M
E418  5F                   MOV    E, A
E419  1A                   LDAX   D
E41A  D3  04               OUT    RDVR
E41C  2B                   DCX    H
E41D  0D                   DCR    C
E41E  C2  16  E4           JNZ    LOOP
E421  EB                   XCHG
E422  3A  02  E0           LDA    COLMN
E425  D3  05               OUT    CDVR
E427  FE  EF               CPI    0EFH
E429  CA  3B  E4           JZ     FIRST
E42C  07                   RLC
E42D  32  02  E0           STA    COLMN
E430  68                   MOV    L, B
E431  01  80  00           LXI    B, 0080H
E434  09                   DAD    B
E435  22  05  E0           SHLD   BASE
E438  C3  4D  E4           JMP    END
E43B  3E  FE       FIRST   MVI    A, 0FEH
E43D  32  02  E0           STA    COLMN
E440  21  00  E5           LXI    H, DECDR
E443  22  05  E0           SHLD   BASE
E446  2A  03  E0           LHLD   COUNT
E449  2B                   DCX    H
E44A  22  03  E0           SHLD   COUNT
E44D  E1           END     POP    H
E44E  D1                   POP    D
E44F  C1                   POP    B
E450  F1                   POP    PSW
E451  C9                   RET
```

**Figure 7b. 8080A Microprocessor Program that Decodes a 32 Character ASCII String Prior to Loading into the REFRESH CONTROLLER**

## DECODED DATA CONTROLLER

The DECODED DATA CONTROLLER circuit schematic for a 32 character display is depicted in Figure 8. The circuit is specifically designed for interface to an 8080A microprocessor. This circuit is designed to accept and store in local memory all of the display data for a 32 character HDSP-2000 display (1120 bits). The microprocessor loads 160 bytes of display data into the two 1K x 1 RAM's via the 74165 parallel in, serial out shift register. Each byte of data represents one column of display data. The counter string automatically generates the proper address location for each serial bit of data after initialization by $\overline{\text{MEM W}}$, the character address, and the desired column. Once the loading is complete, the counter sequentially loads and displays each column (224 bits) of data at a 90Hz rate (2MHz input clock rate). The timing for this circuit is shown in Figure 9. The software required to decode a 32 character ASCII string is shown in Figure 10. This program decodes the 32 ASCII characters into 160 bytes of display data which are then stored in the controller. The program requires about 6.6ms, for a 2MHz clock, to decode and load the message into the DECODED DATA CONTROLLER.

## RAM CONTROLLER

The RAM CONTROLLER (Figure 11a) is designed to accept ASCII coded data for storage in a local 128 x 8 RAM. After the microprocessor has loaded the RAM, local scanning circuitry controls the decoding of the ASCII, the display data loading, and the column select function. With minor modification, the circuit can be utilized for up to 128 display characters. The RAM used in this circuit is an MCM6810P with the Address and Data inputs isolated via 74LS367 tristate buffers. This allows the RAM to be accessed either by the microprocessor or by the local electronics. The protocol is arranged such that the microprocessor always takes precedence over the local scanning electronics. The "Write" cycle timing for the RAM CONTROLLER is depicted in Figure 11b. This circuit, as with the DECODED DATA CONTROLLER, requires no microprocessor time once the local RAM has been loaded with the desired data.

## DISPLAY PROCESSOR CONTROLLER

The previously mentioned interface techniques provide only for the display of ASCII coded data. Such important features as a blinking cursor, editing routines, and character addressing must be provided by other subroutines in the microprocessor software. The DISPLAY PROCESSOR CONTROLLER is a system which utilizes a dedicated 8048 single chip microprocessor to provide these important features. This controller, as depicted in Figure 12, is a series of printed circuit board subsystems available from Hewlett-Packard under the following part numbers:

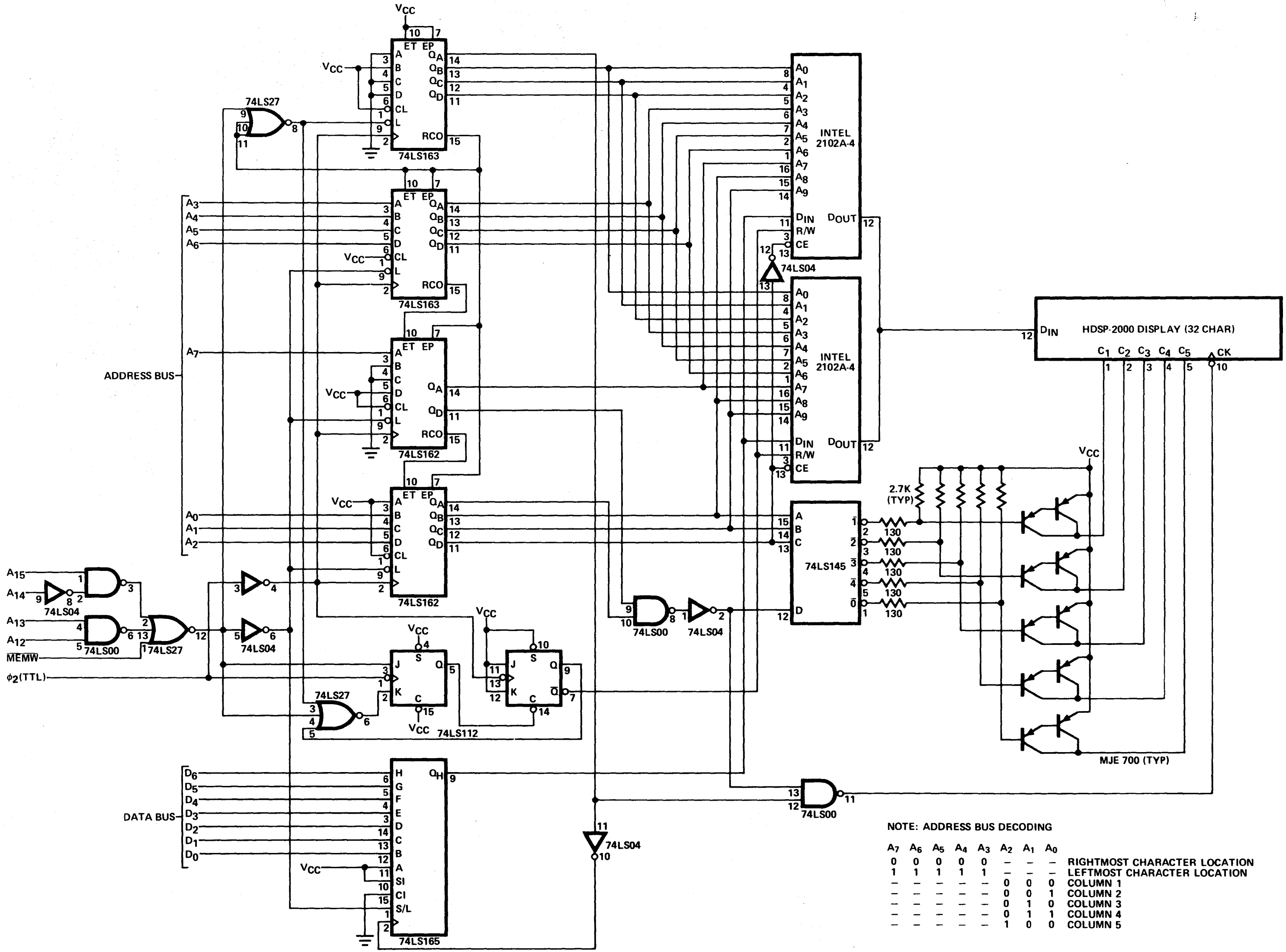
HDSP-2470 — Controller with 64 character ASCII to 5 x 7 decoder

HDSP-2471 — Controller with 128 character universal ASCII to 5 x 7 decoder

HDSP-2472 — Controller with socket for user supplied custom coded ROM/PROM/EPROM.

All of the controllers have the following features:

- Choice of character string length:
  4-48 characters in increments of four characters
- Four modes of data entry
  Left Entry
  Right Entry
  RAM Entry (≤ 32 characters only)
  Block Entry
- Flashing Cursor — Left Entry Only
- Data Out (≤ 32 characters only)
- Edit Functions
  Clear Display ⎤ RIGHT ⎤
  Backspace Cursor ⎦ ENTRY ⎥
  Forwardspace Cursor ⎥ LEFT ENTRY
  Insert ⎥
  Delete ⎦



NOTE: ADDRESS BUS DECODING

| A7 | A6 | A5 | A4 | A3 | A2 | A1 | A0 | |
|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | – | – | – | RIGHTMOST CHARACTER LOCATION |
| 1 | 1 | 1 | 1 | 1 | – | – | – | LEFTMOST CHARACTER LOCATION |
| – | – | – | – | – | 0 | 0 | 0 | COLUMN 1 |
| – | – | – | – | – | 0 | 0 | 1 | COLUMN 2 |
| – | – | – | – | – | 0 | 1 | 0 | COLUMN 3 |
| – | – | – | – | – | 0 | 1 | 1 | COLUMN 4 |
| – | – | – | – | – | 1 | 0 | 0 | COLUMN 5 |

**Figure 8. 8080A Microprocessor Interface to the HDSP-2000 DECODED DATA CONTROLLER**

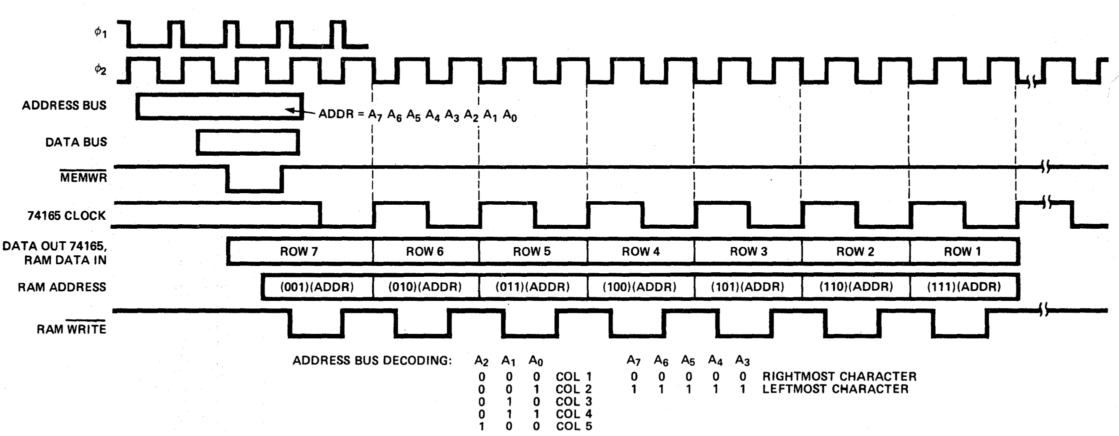


**Figure 9. Data Entry Timing for DECODED DATA CONTROLLER**

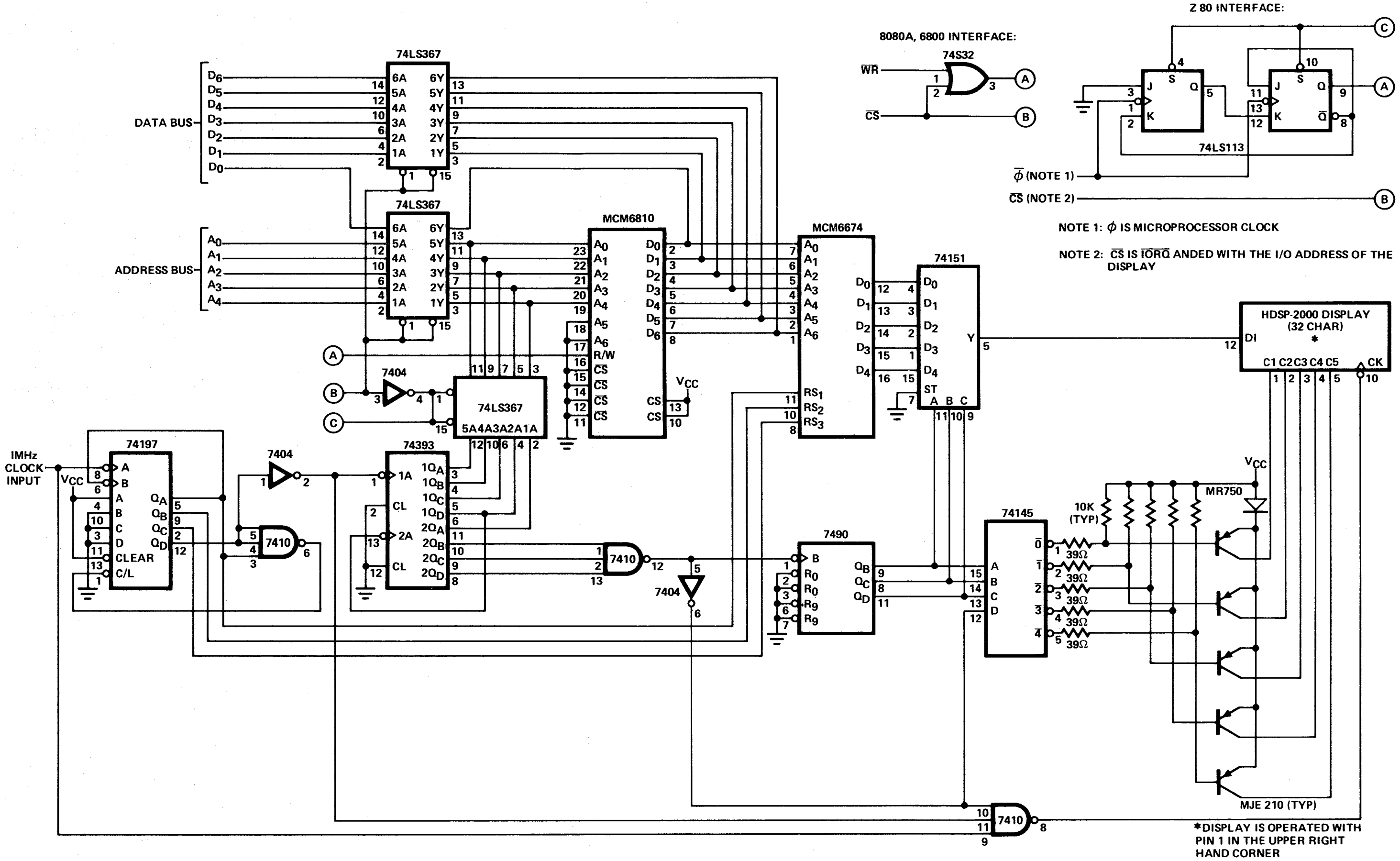
```
        OBJECT
LOC     CODE        SOURCE STATEMENTS

B000                DISPL    EQU    0B000H
E500                DECDR    EQU    0E500H

                             ORG    0E000H
E000    02  E0      ASCII    DW     DATA
E002    00                   DS     32

                             ORG    0E400H
E400    11  F8  B0  LOAD     LXI    D, DISPL+00F8H
E403    0E  20               MVI    C, 32
E405    2A  00  E0  LOOP1    LHLD   ASCII
E408    7E                   MOV    A, M
E409    23                   INX    H
E40A    22  00  E0           SHLD   ASCII
E40D    26  E5               MVI    H, DECDR/256
E40F    6F                   MOV    L, A
E410    06  05               MVI    B, 5
E412    7E          LOOP2    MOV    A, M
E413    12                   STAX   D
E414    13                   INX    D
E415    7D                   MOV    A, L
E416    C6  80               ADI    80H
E418    6F                   MOV    L, A
E419    D2  1D  E4           JNC    LOOP3
E41C    24                   INR    H
E41D    05          LOOP3    DCR    B
E41E    C2  12  E4           JNZ    LOOP2
E421    7B                   MOV    A, E
E422    D6  0D               SUI    13
E424    5F                   MOV    E, A
E425    0D                   DCR    C
E426    C2  05  E4           JNZ    LOOP1
E429    C9                   RET
```

**Figure 10. 8080A Microprocessor Program that Decodes a 32 Character ASCII String Prior to Loading into the DECODED DATA CONTROLLER**

These controllers have been designed to eliminate the burden of data handling between keyboard, display, and microprocessor. The product data sheet describes the technical function of the controllers in detail.

Interfacing the controller to microprocessor systems depends on the needs of the particular application. Figures 13a and 13b depict latched interfaces from a master microprocessor to the HDSP-247X series of controllers. These interfaces are utilized to avoid having the master processor wait for the controller to accept data.

In sophisticated systems, it may be desirable to have the HDSP-247X controller handle all of the keyboard/display interface while the microprocessor reads edited messages from the controller DATA OUT port. This function can be achieved through the use of peripheral interface adapters (PIA) available from the microprocessor manufacturers. Figure 14 depicts a 6800 based system in which data may enter the display from either a keyboard or a microprocessor. This interface uses a 6821 PIA configured so that $PB_7$ controls whether the microprocessor or keyboard enters data into the controller. The 6800 program is shown in Figure 15. Subroutine "LOAD" uses $CA_1$ and $CA_2$ to provide a data entry handshake that allows the 6800 to load data into the controller as fast as the controller can accept it. After the prompting message has been loaded, the microprocessor turns the control of data entry over to the keyboard. A signal from the keyboard ("ER" in the example) sets a flag within the 6821. Depending on how the 6821 is configured, the microprocessor can either test the flag or allow the flag to automatically interrupt the microprocessor. Subroutine "READ" would then be used to read the DATA OUT



NOTE 1: φ IS MICROPROCESSOR CLOCK

NOTE 2: CS IS IORQ ANDED WITH THE I/O ADDRESS OF THE DISPLAY

*DISPLAY IS OPERATED WITH PIN 1 IN THE UPPER RIGHT HAND CORNER

**Figure 11a. 8080A Microprocessor Interface to the HDSP-2000 RAM CONTROLLER**

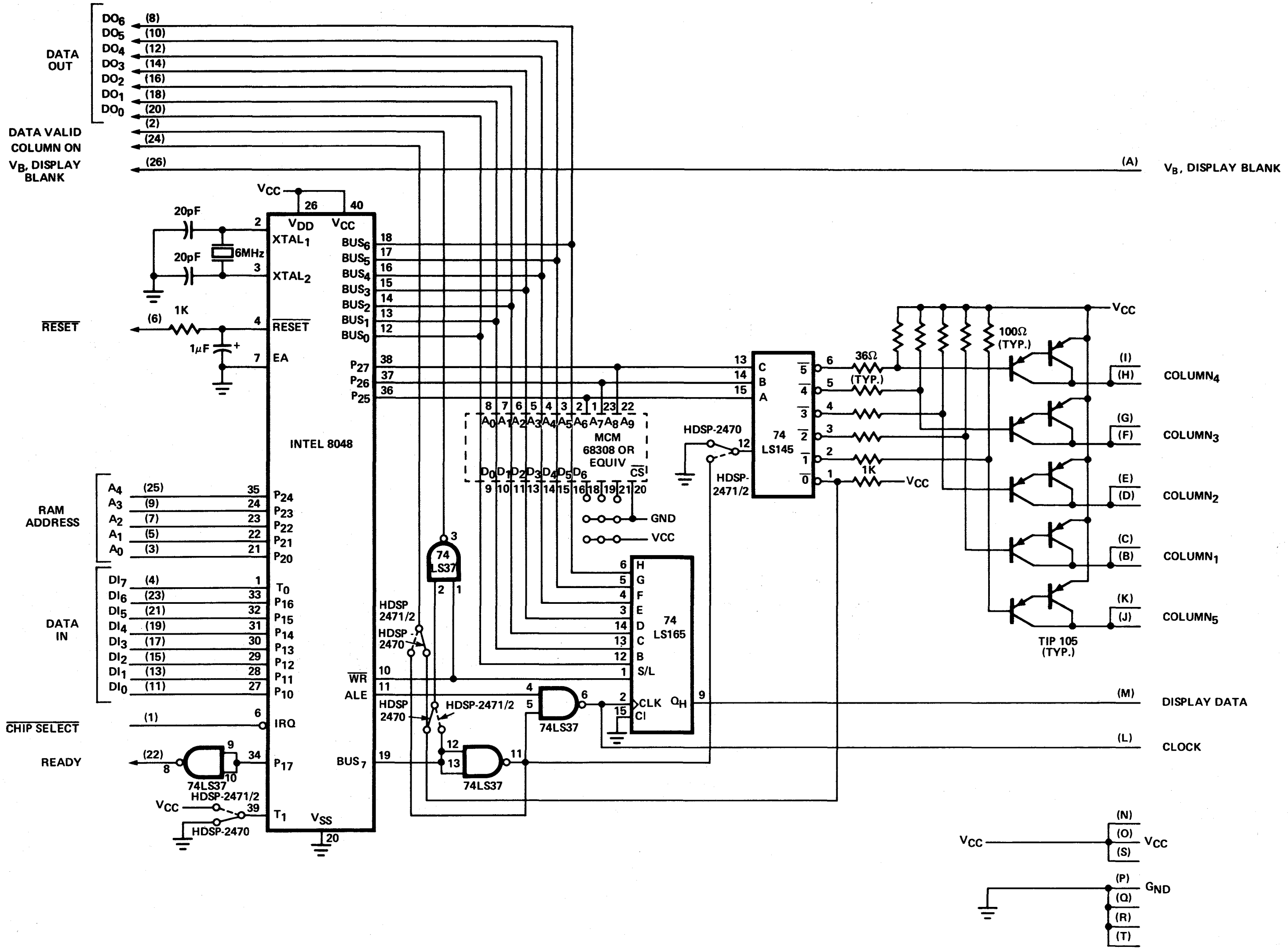
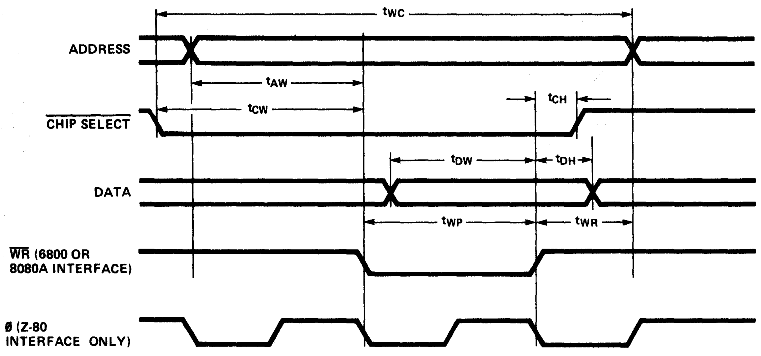
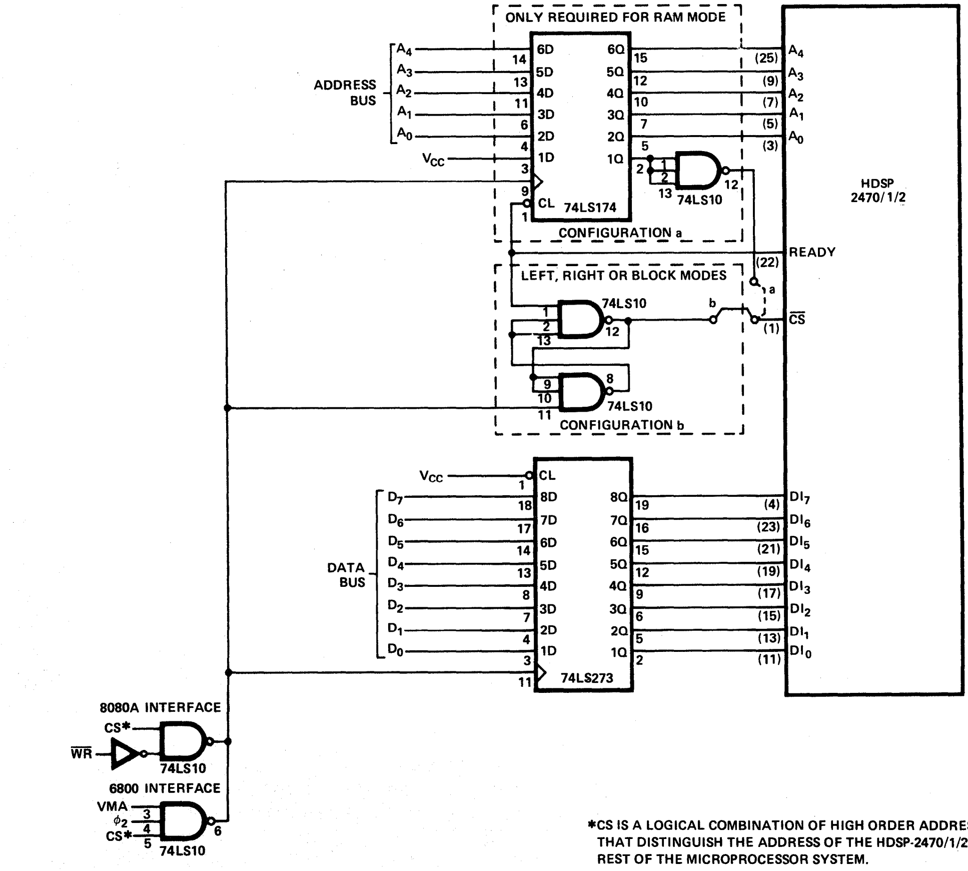


**Figure 12. HDSP-2470/-2471/-2472 DISPLAY PROCESSOR CONTROLLER**



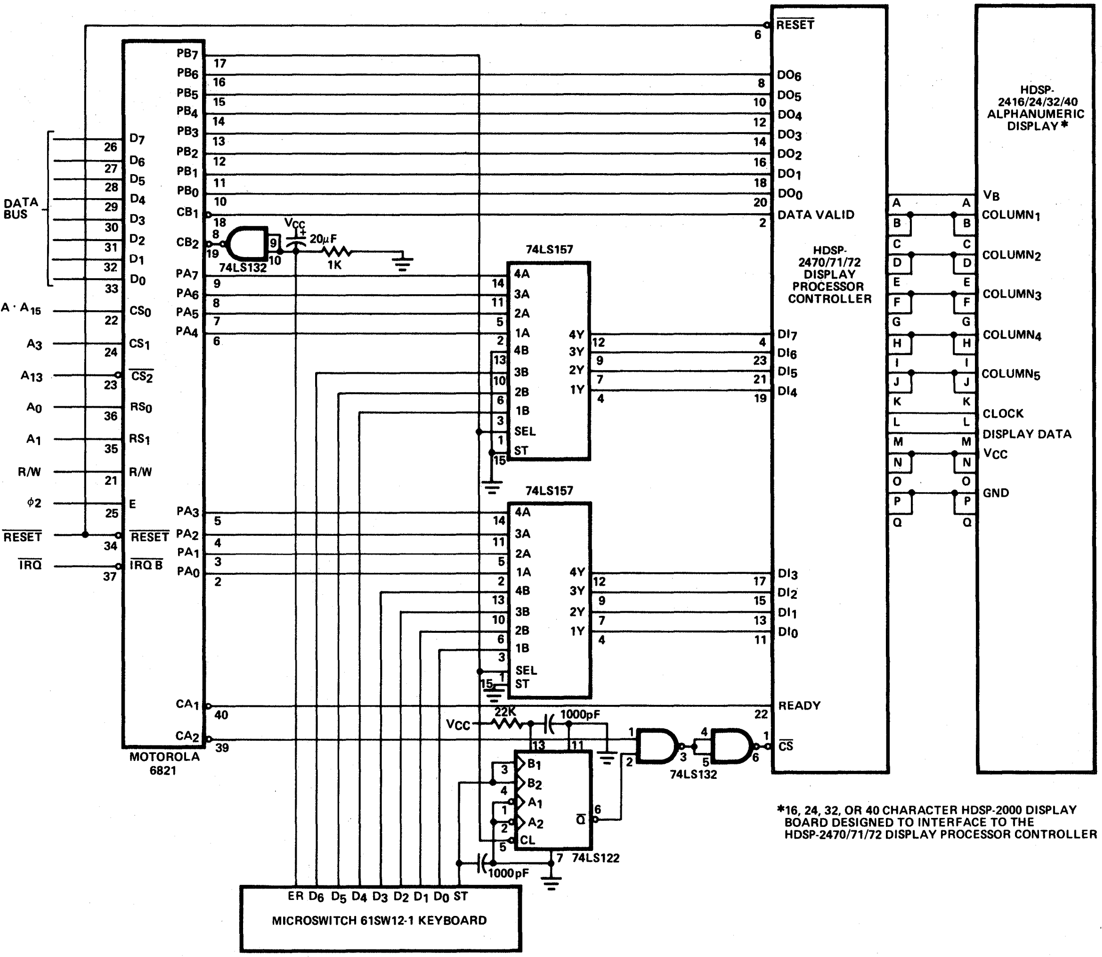
| PARAMETER | SYMBOL | MIN. |
|---|---|---|
| WRITE CYCLE | $t_{WC}$ | 390ns |
| WRITE DELAY | $t_{AW}$ | 65ns |
| CHIP ENABLE TO WRITE | $t_{CW}$ | 65ns |
| DATA SETUP | $t_{DW}$ | 220ns |
| DATA HOLD | $t_{DH}$ | 20ns |
| WRITE PULSE | $t_{WP}$ | 310ns |
| WRITE RECOVERY | $t_{WR}$ | 10ns |
| CHIP ENABLE HOLD | $t_{CH}$ | 20ns |

**Figure 11b. Memory Write Timing for the HDSP-2000 RAM CONTROLLER**



*CS IS A LOGICAL COMBINATION OF HIGH ORDER ADDRESS BITS THAT DISTINGUISH THE ADDRESS OF THE HDSP-2470/1/2 FROM THE REST OF THE MICROPROCESSOR SYSTEM.

**Figure 13. Latched Interface to the HDSP-2470/-2471/-2472 DISPLAY PROCESSOR CONTROLLER**



*16, 24, 32, OR 40 CHARACTER HDSP-2000 DISPLAY BOARD DESIGNED TO INTERFACE TO THE HDSP-2470/71/72 DISPLAY PROCESSOR CONTROLLER

**Figure 14. 6800 Microprocessor Interface Utilizing a 6820 PIA for an HDSP-2470/-2471/-2472 Alphanumeric Terminal**

outputs from the controller into the microprocessor system. The microprocessor uses the $CB_1$ input of the 6821 PIA to determine when to read each of the 34 data output words into the system.

A similar PIA interface for the 8080A microprocessor is depicted in Figures 16 and 17.

The HDSP-247X series of controllers are programmed to default to "Left Entry" mode for a 32 character string of displays. If some other entry mode or string length is desired, it is necessary to either load the appropriate control word from the microprocessor or to provide a control word during POWER ON RESET. The controller will read the DATA IN lines during RESET and interpret the contents as the control word. The circuit depicted in Figure 18 can be utilized to load any desired preprogrammed word into the HDSP-247X controller, during power on.

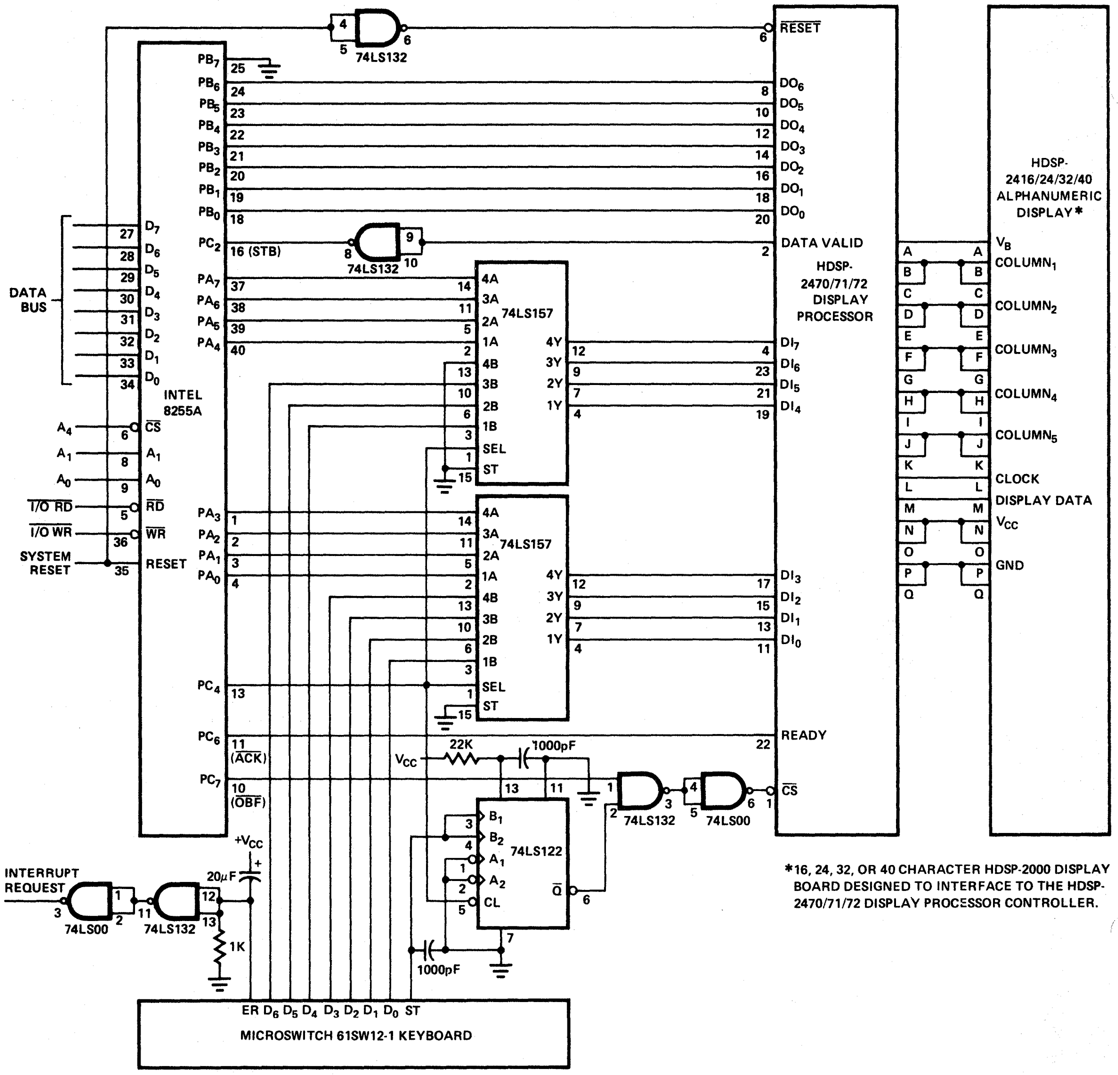
Under certain operating conditions, it may be desirable to vary the brightness of displays controlled by the HDSP-247X controllers. The circuit depicted in Figure 19 may be utilized to provide manual brightness control of the display through pulse width modulation. Automatic control may be achieved by substituting an appropriate value photoconductor for potentiometer $R_1$.

```
*  PORT CONFIGURATION:
*  1.   PORT A:
                PA0-PA7 OUTPUTS TO DATA IN OF HDSP-247X
*               CA1  (INPUT) MODE 00 SET FLAG NEG EDGE OF READY
*               CA2  (OUTPUT) MODE 100 CLEARED MPU READ PRA, SET
*               NEG  EDGE OF READY
*  1.   PORT B:
*               PB0-PB6 INPUTS DATA TO 6800 FROM DATA OUT OF HDSP-247X
*               CB1  (INPUT) MODE 00 SETS FLAG NEG EDGE OF DATA VALID
*               CB2  (INPUT) MODE 000 SETS FLAG NEG EDGE OF ER KEY
*               CB2  (INPUT) MODE 001 SETS FLAG NEG EDGE OF ER KEY
                CAUSING IRQ
*               PB7  (OUTPUT) LOW ENABLES PA0-PA7 TO MUX
*                             HIGH ENABLES  KEYBOARD TO MUX
*

LOC   OBJECT     CODE      SOURCE STATEMENT

      8008       PRA       EQU     $8008
      8008       DRA       EQU     $8008
      8009       CRA       EQU     $8009
      800A       PRB       EQU     $800A
      800A       DRB       EQU     $800A
      800B       CRB       EQU     $800B

                           ORG     $0000
0000             MESSAGE   RMB     2

                           ORG     $0100
0100             STATUS    RMB     1
0101             CURSOR    RMB     1
0102             DATA      RMB     32

                           ORG     $0400
0400  CE  0100   READ      LDX     I, STATUS
0403  B6  800A   LOOP1     LDA  A  E, PRB          CLEAR CB1 AND CB2
0406  5F                   CLR  B
0407  5C         LOOP2     INC  B
0408  B6  800B             LDA  A  E, CRB
040B  2A  FA               BPL     LOOP2           WAIT FOR DATA VALID
040D  C1  0A               CMP  B  I, 10
040F  23  F2               BLS     LOOP1
0411  C6  21               LDA  B  I, 33
0413  B6  800A   LOOP3     LDA  A  E, PRB          READ AND CLEAR CB1
0416  84  7F               AND  A  I, $7F
0418  A7  00               STA  A  X, 0            STORE IN RAM
041A  B6  800B   LOOP4     LDA  A  E, CRB
041D  2A  FB               BPL     LOOP4           WAIT FOR DATA VALID
041F  08                   INX
0420  5A                   DEC  B
0421  26  F0               BNE     LOOP3           READ DATA
0423  B6  800A             LDA  A  E, PRB
0426  84  7F               AND  A  I, $7F
0428  A7  00               STA  A  X, 0
042A  39                   RTS

042B  DE  00     LOAD      LDX     D, MESSGE
042D  A6  00     LOOP10    LDA  A  X, 0
042F  08                   INX
0430  81  FF               CMP  A  I, $FF          LAST WORD IN STRING
0432  27  0D               BEQ     ENDL            JUMP WHEN DONE
0434  B7  8008             STA  A  E, PRA
0437  7D  8008             TST     E, PRA         CLEAR CA1  AND CA2
043A  B6  8009   LOOP11    LDA  A  E, CRA
043D  2A  FB               BPL     LOOP11          WAIT
043F  20  EC               BRA     LOOP10
0441  DF  00     ENDL      STX     D, MESSGE
0443  39                   RTS

                           ORG     $0500
0500  7F  8009   START     CLR     E, CRA
0503  7F  800B             CLR     E, CRB
0506  86  FF               LDA  A  I, $FF
0508  B7  8008             STA  A  E, DRA
050B  86  24               LDA  A  I, $24
050D  B7  8009             STA  A  E, CRA
0510  86  80               LDA  A  I, $80
0512  B7  800A             STA  A  E, DRB
0515  86  04               LDA  A  I, $04
0517  B7  800B             STA  A  E, CRB

                 *  PROCEDURE TO LOAD HDSP-247X SYSTEM
051A  0E                   CLI
051B  7F  800A             CLR     E, PRB      DISABLE KEYBD FROM MUX
051E  BD  042B             JSR     E, LOAD

                 *  PROCEDURE TO READ DATA OUT OF HDSP-247X SYSTEM
0521  7D  800A             TST     E, PRB     CLEAR CB1, CB2
0524  86  80               LDA  A  I, $80
0526  B7  800A             STA  A  E, PRB      ENABLE KEYBD TO MUX
0529  86  0C               LDA  A  I, $0C
042B  B7  800B             STA  A  E, CRB      ENABLE  IRQ,
052E  0F                   SEI                 IRQ CAUSE JSR TO READ
```

**Figure 15. 6800 Microprocessor Program that Interfaces to the Circuit shown in Figure 14.**

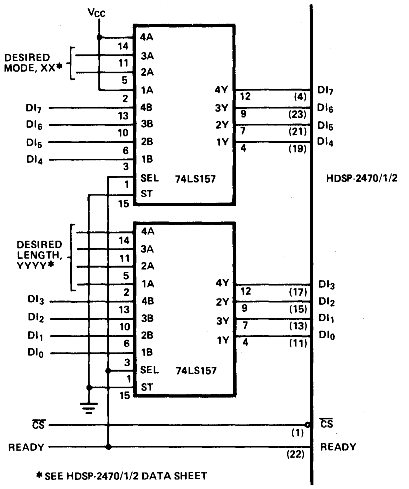
```
*  PORT CONFIGURATION:
*  1.   PORT A (MODE 1 OUTPUT):
*               PA0-PA7 OUTPUTS TO DATA IN OF HDSP-247X
*               PC7  (OBF) OUTPUT; TO CHIP SELECT
*               PC6 (ACK) INPUT; TO READY
*               FLAG PC7 (OBF) CLEARED BY OUTPUT; SET BY READY
*
*  2.   PORT B (MODE 1 INPUT):
*               PB0-PB6 INPUTS DATA FROM DATA OUT OF HDSP-247X
*               PC2 (STB) INPUT; LOADS DATA ON NEG EDGE OF DATA VALID
*               FLAG PCO (INTR) CLEARED BY INPUT; SET BY DATA VALID
*
*  3.   PORT C:
*               PC4 OUTPUT; LOW ENABLES PA0-PA7 TO HDSP-247X
*                            HIGH ENABLES KEYBOARD TO HDSP-247X

LOC   OBJECT      CODE      SOURCE STATEMENTS

000C              PA        EQU     0CH
000D              PB        EQU     0DH
000E              PC        EQU     0EH
000F              CNTRL     EQU     0FH

                            ORG     0E000H
E000  02  E0      ASCII     DW      TEXT
E002  00          TEXT      DS      32

                            ORG     0E100H
E100  00          STAT      DB      0
E101  00          ADDR      DB      0
E102  00          DATA      DS      32

                            ORG     0E400H
E400  F3          READ      DI
E401  F5                    PUSH    PSW
E402  E5                    PUSH    H
E403  C5                    PUSH    B
E404  0E  20                MVI     C, 32
E406  21  00 E1             LXI     H, STAT     FIRST WORD
E409  DB  0D                IN      PB          CLEAR INTR
E40B  06  00      LOOP1     MVI     B, 0
E40D  DB  0E      LOOP2     IN      PC
E40F  04                    INR     B
E410  1F                    RAR
E411  D2  0D E4             JNC     LOOP2       WAIT UNTIL INTR IS SET
E414  3E  0A                MVI     A, 10
E416  B8                    CMP     B
E417  DB  0D                IN      PB
E419  D2  0B E4             JNC     LOOP1       WAIT UNTIL STATUS WORD
E41C  77          LOOP3     MOV     M, A        STORE IN RAM
E41D  23                    INX     H
E41E  DB  0E      LOOP4     IN      PC
E420  1F                    RAR
E421  D2  1E E4             JNC     LOOP4       WAIT UNTIL INTR IS SET
E424  DB  0D                IN      PB
E426  0D                    DCR     C
E427  C2  1C E4             JNZ     LOOP3
E42A  77                    MOV     M, A        STORE LAST WORD
E42B  C1                    POP     B
E42C  E1                    POP     H
E42D  F1                    POP     PSW
E42E  FB                    EI
E42F  C9                    RET

E430  2A  00 E0   LOAD      LHLD    ASCII       FIRST WORD OF MESSAGE
E433  7E          LOOP5     MOV     A, M
F434  FE  FF                CPI     0FFH        CHECK TO SEE IF DONE
E436  CA  45 E4             JZ      ENDL
E439  D3  0C                OUT     PA          OUTPUT TO DISPLAY
E43B  23                    INX     H
E43C  DB  0E      LOOP6     IN      PC
E43E  17                    RAL
E43F  D2  3C E4             JNC     LOOP6       WAIT
E442  C3  33 E4             JMP     LOOP5       NEXT WORD
E445  23          ENDL      INX     H
E446  22  00 E0             SHLD    ASCII
E449  C9                    RET

E44A  3E  A7      START     MVI     A, 0A7H     PA OUTPUT, PB INPUT
E44C  D3  0F                OUT     CNTRL
E44E  3E  0C                MVI     A, 0CH      CLEAR INTE A
E450  D3  0F                OUT     CNTRL
E452  3E  05                MVI     A, 05H
E454  D3  0F                OUT     CNTRL       SET INTE B

                  *  PROCEDURE TO LOAD HDSP-247X SYSTEM
E456  3E  08                MVI     A, 08H
E458  D3  0F                OUT     CNTRL      ENABLE A SIDE OF MUX
E45A  CD  30 E4             CALL    LOAD

                  *  PROCEDURE TO READ DATA OUT OF HDSP-247X SYSTEM
E45D  3E  09                MVI     A, 09H
E45F  D3  0F                OUT     CNTRL      ENABLE B SIDE OF MUX
E461  FB                    EI                 INT MUST CALL READ
```

**Figure 16. 8080A Microprocessor Program that Interfaces to the Circuit shown in Figure 17.**



*16, 24, 32, OR 40 CHARACTER HDSP-2000 DISPLAY BOARD DESIGNED TO INTERFACE TO THE HDSP-2470/71/72 DISPLAY PROCESSOR CONTROLLER.

**Figure 17. 8080A Microprocessor Interface Utilizing an 8255 PIA for an HDSP-2470/-2471/-2472 Alphanumeric Terminal**



*SEE HDSP-2470/1/2 DATA SHEET

**Figure 18. External Circuitry to Load a Control Word into the HDSP-2470/-2471/-2472 Alphanumeric System upon Reset**

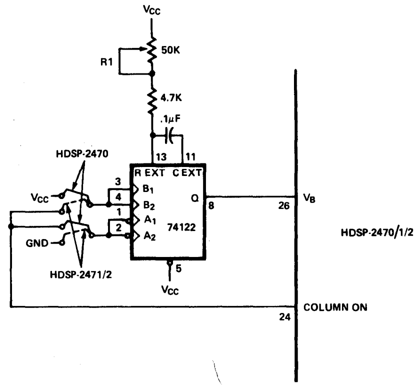


**Figure 19. External Circuitry to Vary Luminous Intensity of the HDSP-2470/-2471/-2472 Alphanumeric Display System**

| DECODER ADDRESS FOR FIG. 7a, 7b, 10 | DECODER ADDRESS FOR FIG. 6 | HDSP-2471 ROM ADDRESS | HEXIDECIMAL DATA | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| E500 | 0600 | 080 | 08 | 30 | 45 | 7D | 7D | 38 | 7E | 30 | 60 | 1E | 3E | 62 | 40 | 08 | 38 | 41 | COLUMN$_1$ |
| | | 090 | 10 | 18 | 5E | 78 | 38 | 78 | 38 | 3C | 38 | 3C | 38 | 08 | 20 | 12 | 48 | 01 | |
| | | 0A0 | 00 | 00 | 00 | 14 | 24 | 23 | 36 | 00 | 00 | 00 | 08 | 08 | 00 | 08 | 00 | 20 | |
| | | 0B0 | 3E | 00 | 62 | 22 | 18 | 27 | 3C | 01 | 36 | 06 | 00 | 00 | 00 | 14 | 41 | 06 | |
| | | 0C0 | 3E | 7E | 7F | 3E | 7F | 7F | 7F | 3E | 7F | 00 | 20 | 7F | 7F | 7F | 7F | 3E | |
| | | 0D0 | 7F | 3E | 7F | 26 | 01 | 3F | 07 | 7F | 63 | 03 | 61 | 00 | 02 | 41 | 04 | 40 | |
| | | 0E0 | 00 | 38 | 7F | 38 | 38 | 38 | 08 | 08 | 7F | 00 | 20 | 00 | 00 | 78 | 7C | 38 | |
| | | 0F0 | 7C | 18 | 00 | 48 | 04 | 3C | 1C | 3C | 44 | 04 | 44 | 00 | 00 | 00 | 08 | 2A | |
| E580 | 0680 | 100 | 1C | 48 | 29 | 09 | 09 | 44 | 01 | 4A | 50 | 04 | 49 | 14 | 3C | 7C | 44 | 63 | COLUMN$_2$ |
| | | 110 | 08 | 24 | 61 | 14 | 44 | 15 | 45 | 43 | 45 | 41 | 42 | 08 | 7E | 19 | 7E | 12 | |
| | | 120 | 00 | 5F | 03 | 7F | 2A | 13 | 49 | 0B | 00 | 41 | 2A | 08 | 58 | 08 | 30 | 10 | |
| | | 130 | 51 | 42 | 51 | 41 | 14 | 45 | 4A | 71 | 49 | 49 | 36 | 5B | 08 | 14 | 22 | 01 | |
| | | 140 | 41 | 09 | 49 | 41 | 41 | 49 | 09 | 41 | 08 | 41 | 40 | 08 | 40 | 02 | 04 | 41 | |
| | | 150 | 09 | 41 | 09 | 49 | 01 | 40 | 18 | 20 | 14 | 04 | 51 | 00 | 04 | 41 | 02 | 40 | |
| | | 160 | 07 | 44 | 48 | 44 | 44 | 54 | 7E | 14 | 08 | 44 | 40 | 7F | 41 | 04 | 08 | 44 | |
| | | 170 | 14 | 24 | 7C | 54 | 3E | 40 | 20 | 40 | 28 | 48 | 64 | 08 | 00 | 41 | 04 | 55 | |
| E600 | 0700 | 180 | 3E | 45 | 11 | 11 | 05 | 44 | 29 | 4D | 48 | 04 | 49 | 08 | 20 | 04 | 44 | 55 | COLUMN$_3$ |
| | | 190 | 78 | 7E | 01 | 15 | 45 | 14 | 44 | 42 | 44 | 40 | 40 | 2A | 02 | 15 | 49 | 7C | |
| | | 1A0 | 00 | 00 | 00 | 14 | 7F | 08 | 56 | 07 | 3E | 3E | 1C | 3E | 38 | 08 | 30 | 08 | |
| | | 1B0 | 49 | 7F | 49 | 49 | 12 | 45 | 49 | 09 | 49 | 49 | 36 | 3B | 14 | 14 | 14 | 51 | |
| | | 1C0 | 5D | 09 | 49 | 41 | 41 | 49 | 09 | 41 | 08 | 7F | 40 | 14 | 40 | 0C | 08 | 41 | |
| | | 1D0 | 09 | 51 | 19 | 49 | 7F | 40 | 60 | 18 | 08 | 78 | 49 | 7F | 08 | 7F | 7F | 40 | |
| | | 1E0 | 0B | 44 | 44 | 44 | 44 | 54 | 09 | 54 | 04 | 7D | 44 | 10 | 7F | 18 | 04 | 44 | |
| | | 1F0 | 24 | 14 | 08 | 54 | 44 | 40 | 40 | 30 | 10 | 30 | 54 | 36 | 77 | 36 | 08 | 2A | |
| E680 | 0780 | 200 | 7F | 40 | 29 | 21 | 05 | 38 | 2E | 49 | 50 | 38 | 49 | 10 | 20 | 7C | 3C | 49 | COLUMN$_4$ |
| | | 210 | 08 | 24 | 61 | 14 | 3C | 15 | 3D | 43 | 45 | 41 | 42 | 1C | 02 | 12 | 41 | 12 | |
| | | 220 | 00 | 00 | 03 | 7F | 2A | 64 | 20 | 00 | 41 | 00 | 2A | 08 | 00 | 08 | 00 | 04 | |
| | | 230 | 45 | 40 | 49 | 49 | 7F | 45 | 49 | 05 | 49 | 29 | 00 | 00 | 22 | 14 | 08 | 09 | |
| | | 240 | 55 | 09 | 49 | 41 | 41 | 49 | 09 | 51 | 08 | 41 | 40 | 22 | 40 | 02 | 10 | 41 | |
| | | 250 | 09 | 21 | 29 | 49 | 01 | 40 | 18 | 20 | 14 | 04 | 45 | 41 | 10 | 00 | 02 | 40 | |
| | | 260 | 00 | 3C | 44 | 44 | 48 | 54 | 02 | 54 | 04 | 40 | 3D | 28 | 40 | 04 | 04 | 44 | |
| | | 270 | 24 | 7C | 04 | 54 | 20 | 20 | 20 | 40 | 28 | 08 | 4C | 41 | 00 | 08 | 10 | 55 | |
| E700 | 0800 | 280 | 00 | 30 | 45 | 7D | 79 | 44 | 10 | 30 | 60 | 40 | 3E | 60 | 1C | 02 | 04 | 41 | COLUMN$_5$ |
| | | 290 | 04 | 18 | 5E | 78 | 40 | 78 | 40 | 3C | 38 | 3C | 38 | 08 | 02 | 00 | 42 | 01 | |
| | | 2A0 | 00 | 00 | 00 | 14 | 12 | 62 | 50 | 00 | 00 | 00 | 08 | 08 | 00 | 08 | 00 | 02 | |
| | | 2B0 | 3E | 00 | 46 | 36 | 10 | 39 | 30 | 03 | 36 | 1E | 00 | 00 | 41 | 14 | 00 | 06 | |
| | | 2C0 | 1E | 7E | 36 | 22 | 3E | 41 | 01 | 72 | 7F | 00 | 3F | 41 | 40 | 7F | 7F | 3E | |
| | | 2D0 | 06 | 5E | 46 | 32 | 01 | 3F | 07 | 7F | 63 | 03 | 43 | 41 | 20 | 00 | 04 | 40 | |
| | | 2E0 | 00 | 40 | 38 | 20 | 7F | 08 | 00 | 3C | 78 | 00 | 00 | 44 | 00 | 78 | 78 | 38 | |
| | | 2F0 | 18 | 40 | 04 | 20 | 00 | 7C | 1C | 3C | 44 | 04 | 44 | 00 | 00 | 00 | 08 | 2A | |

**Figure 20. 128 Character ASCII Decoder Table Used by the 6800 Refresh Program in Figure 6, 8080A Refresh Programs in Figures 7a, 7b, and 10, and the HDSP-2471 DISPLAY PROCESSOR CONTROLLER. Decoded 5x7 Display Font is shown in the HDSP-247X Data Sheet**