HEWLETT PACKARD

APPLICATION NOTE 1003

Interfacing 18 Segment Displays to Microprocessors

INTRODUCTION

Over the past four years, the need for alphanumeric displays has grown very rapidly due to the extensive use of microprocessors in new system designs. The HDSP-6508 and HDSP-6300 alphanumeric displays were developed to provide a low cost, easy-to-use alternative to 5x7 dot matrix displays. These displays use an 18 segment display font that includes a centered decimal point and colon for increased readability. This font is capable of displaying the 64 character ASCII subset (numbers, punctuation symbols, and upper case alphabet) as well as many special purpose symbols. The HDSP-6504 and HDSP-6508 are 3.81 mm (0.150") red 4 or 8 character displays in a dual-in-line package. The HDSP-6300 is a 3.56 mm (0.140") red 8 character display in a dual-in-line package. The HDSP-6508 has character-to-character spacing on 6.35 mm (0.250") centers while the HDSP-6300 has character-to-character spacing on 5.08 mm (0.200") centers. Paralleling the development of these alphanumeric displays have been the introduction of several new display interface circuits that simplify the use of the 18 segment display. These circuits include an ASCII to 18 segment decoder/driver and improved NPN Darlington digit drivers that are designed to interface directly to 5 volt digital logic. This Application Note deals with several techniques to interface the 18 segment display to microprocessor systems. Depending upon the overall system configuration, microprocessor time available to dedicate to display support, and the type of information to be displayed, the system designer would choose the best interface technique to drive an 18 segment display.

DISPLAY INTERFACE TECHNIQUES

This application note will deal with four different techniques, as shown in Figure 1a-d, for interfacing the HDSP-6508 and HDSP-6300 displays to microprocessor systems.

1a. The REFRESH CONTROLLER interfaces the microprocessor system to a multiplexed LED display. The controller periodically interrupts the microprocessor and after each interrupt, the microprocessor supplies new display data for the next refresh cycle of the display.

1b. The DECODED DATA CONTROLLER refreshes a multiplexed LED display independently from the microprocessor system. A local RAM stores decoded display data. This data is continuously read from the RAM and then used to refresh the display. Whenever the display message is changed, the microprocessor decodes each character in software and writes the decoded data into the local RAM.

1c. The CODED DATA CONTROLLER also refreshes a multiplexed LED display independently from the microprocessor system. The local RAM stores ASCII data which is continuously read from the RAM, decoded, and used to refresh the display. The display message is changed by writing new ASCII characters within the local RAM.

1d. The DISPLAY PROCESSOR CONTROLLER uses a separate microprocessor to drive the LED display. This microprocessor provides ASCII storage, ASCII decode, and display refresh independently from the main microprocessor system. Software within the dedicated microprocessor provides many powerful features not available in the other controllers. The main microprocessor updates the LED display by sending new ASCII characters to the slave microprocessor.

COMPARISON OF INTERFACE TECHNIQUES

The choice of a particular interface is an important consideration because it affects the design of the entire microprocessor system. Each interface requires one or more memory or I/O addresses. These addresses are generated by decoding the microprocessor address bus. The display decoder can be located within the microprocessor program or as circuitry within the display interface. Location of the display decoder within the microprocessor program gives the designer total control of the display font within the program. This feature can be particularly important if the display will be used to display different languages and special graphics symbols. The interface technique chosen may limit or interfere with some programming techniques used in the rest of the microprocessor program. For example, the use of an

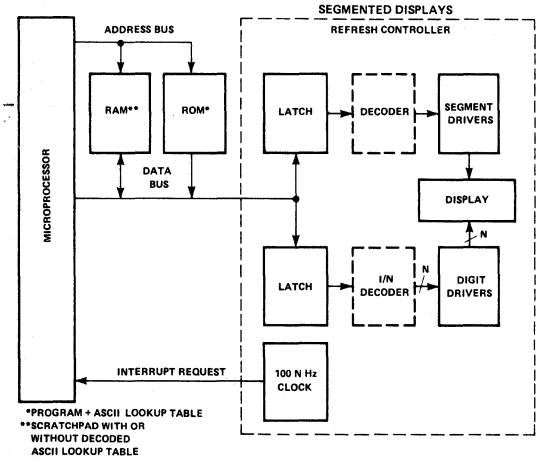


Figure 1a. REFRESH CONTROLLER Display Interface

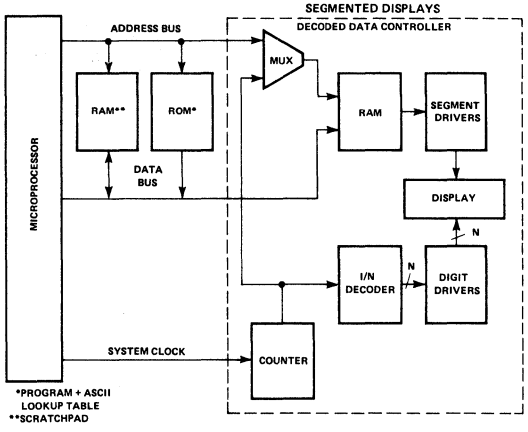


Figure 1b. DECODED DATA CONTROLLER Display Interface

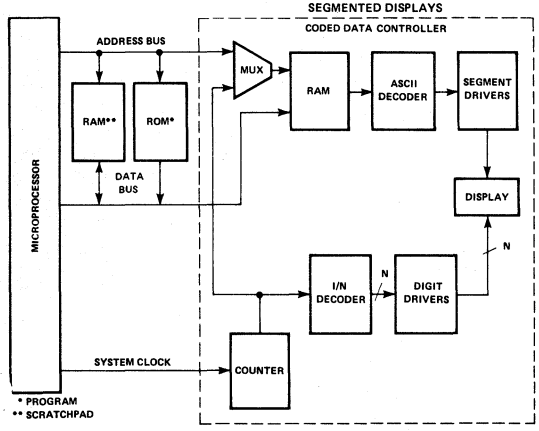


Figure 1c. CODED DATA CONTROLLER Display Interface

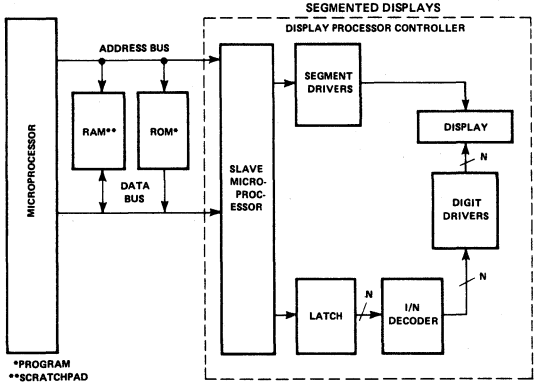


Figure 1d. DISPLAY PROCESSOR CONTROLLER Display Interface

interrupt may restrict the use of some programming techniques used in the interruptable portions of the microprocessor program.

The REFRESH CONTROLLER requires continuous interaction from the microprocessor system. Since the microprocessor actively strobes the LED display, the display interface circuitry is reduced. Generally, this technique provides the lowest hardware cost for any given display length. The display decoder can be located either within the microprocessor program or as circuitry within the interface. Display strobing is accomplished through use of the microprocessor interrupt circuitry. Demands upon microprocessor time are directly proportional to display length.

The DECODED DATA CONTROLLER and CODED DATA CONTROLLER require microprocessor interaction only when the display message is changed. Both techniques employ a local RAM memory that is continuously scanned by the display interface electronics. For the DECODED DATA CONTROLLER, the display decoder is located within the microprocessor software and the local RAM stores decoded display data. The CODED DATA CONTROLLER includes the display decoder within the display interface circuitry and the local RAM stores ASCII data. Since ASCII data is more compact than decoded display data, the CODED DATA CONTROLLER uses a smaller RAM than the DECODED DATA CONTROLLER. Both techniques allow the microprocessor to individually

change each display character by a memory or I/O write to a specific display address. These interface techniques can accept new data at a very high rate.

The DISPLAY PROCESSOR CONTROLLER, like the previously defined CODED and DECODED DATA CONTROLLERS, requires microprocessor interaction only when the display message is changed. By using a dedicated microprocessor, the DISPLAY PROCESSOR CONTROLLER provides many additional display features. These features include multiple entry modes, a blinking cursor, editing commands, and a data output function. The software with the DISPLAY PROCESSOR CONTROLLER further reduces microprocessor interaction by providing more sophisticated data entry modes compared to the RAM entry mode provided by the DECODED DATA and CODED DATA CONTROLLERS. The display decoder can either be designed into the dedicated display microprocessor or can be located within a separate PROM. The use of a PROM allows the user to provide a special character font with additional circuitry. The DISPLAY PROCESSOR CONTROLLER does not allow as high a data entry rate as either the DECODED DATA or CODED DATA CONTROLLERS.

MICROPROCESSOR OPERATION

In order to effectively utilize the interface techniques outlined in the following sections, an understanding of microprocessor fundamentals is required. A brief description of microprocessor fundamentals is included in the following section. A microprocessor system usually consists of a microprocessor, ROM memory, RAM memory, and a specific I/O interface as outline in Figure 2. The microprocessor performs the desired system function by executing a program stored within the ROM. The RAM memory provides temporary storage for the microprocessor system. The I/O interface consists of circuitry that is used as an input to the system or as an output from the system. The microprocessor interfaces to this system through an address bus, data bus, and control bus. The address bus consists of several outputs (A_0, A_1,...A_n) from the microprocessor 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 serves as an input to the microprocessor during a memory or input read and as an output from the microprocessor during a memory or output write. The control bus provides the required timing and signals to 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 bus, data bus, and control bus vary for different microprocessors.

The address, data, and control buses provide the flow of instructions and data into the microprocesor. Program execution consists of a series of memory reads (instruction fetches) which are sometimes followed by a memory read or write (instruction execution). The microprocessor performs a memory read by outputting the memory address of the word to be read on the address bus. This address uniquely specifies a word within the memory system. The microprocessor also outputs a signal on the control bus, which instructs the memory system to perform a memory read. The address selects one memory element, either RAM or ROM, within the memory system. Then, the desired word within the selected memory element is gated on the data bus by the read signal. Meanwhile, the unselected memory elements tristate their output lines so that only the selected memory element is active on the data bus. After sufficient delay, the microprocessor reads the word that appears on the data bus. Similarly, for a memory write, the microprocessor outputs the memory address of the word to be written on the address bus. After sufficient delay, the microprocessor outputs a signal on the control bus, which instructs the memory system to perform a memory write.

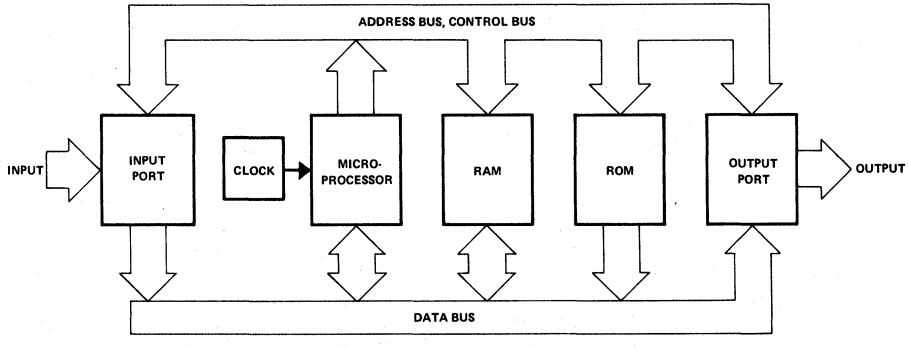


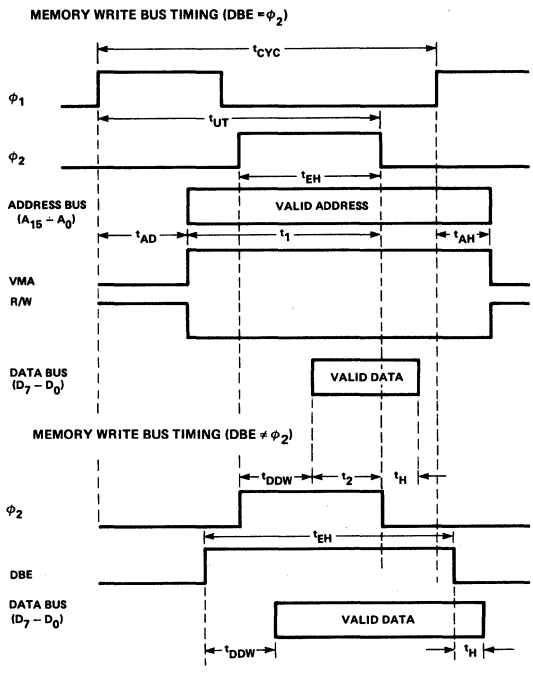
Figure 2. Block Diagram of a Typical Microprocessor System

The microprocessor also outputs the desired memory word on the data bus. The address selects one RAM memory element within the memory system. The write signal causes the memory element to read the word on the data bus and store it at the desired location. After the write cycle has been completed, the new word will have replaced the previous word within the RAM memory. During the memory write, outputs from the unselected memory elements remain tristated so that only the microprocessor is active on the data bus. These control lines and the timing for the address bus, data bus, and control bus vary for different microprocessors.

Some microprocessors, such as the Motorola 6800 microprocessor family, handle memory and I/O in exactly the same way. Memory and I/O occupy a common address space and are accessed by the same instructions. With this type of microprocessor, the hardware decoding of the address bus determines whether the read or write is to a memory or I/O element. Other microprocessors, such as the Intel 8080A, Intel 8085A, and the Zilog Z-80 have separate address spaces for memory and I/O. These microprocessors use different instructions for a memory access or an I/O access and provide signals on the control bus to distinguish between memory and I/O. One advantage of this approach is that the I/O address space can be made smaller to simplify device decoding. However, the I/O instructions that are available are usually not as powerful as the memory reference instructions. Of course, the user can always locate specific I/O devices within the memory address space through proper decoding of the address and control buses. This would allow these I/O devices to be accessed with memory reference instructions.

The 6800 microprocessor family has a 16 line address bus, 8 line data bus, and a control bus that includes the signals VMA (Valid Memory Address), R/$\overline{W}$ (Read/Write), $\overline{DBE}$ (Data Bus Enable), and clock signals ϕ_1 and ϕ_2. R/$\overline{W}$ specifies either a memory read or write while VMA is used in conjunction with R/$\overline{W}$ to specify a valid memory address. DBE gates the internal data bus of the 6800 to the external data bus. In many applications, DBE is connected to ϕ_2. Additional data hold time, t_H, can be achieved by delaying ϕ_2 to the microprocessor or by extending DBE beyond the falling edge of ϕ_2. The timing between the address bus, data bus, VMA, and R/$\overline{W}$ for a memory write is shown in Figure 3.

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), $\overline{WR}$ (Write), SYNC (Synchronizing Signal), READY, and clock signals ϕ_1 and ϕ_2. DBIN and $\overline{WR}$ are used to specify a read or write operation. The 8080A microprocessor distinguishes memory from I/O through the use of a status word that precedes every machine cycle. When SYNC is high, the status word should be loaded into an octal latch on the positive edge of ϕ_1. The outputs from the latch can then be decoded to specify whether the machine cycle is a memory write, memory read, I/O write, or I/O read. The Intel 8228 or 8238 System Controller provides this status latch and additionally encodes the outputs of the status latch with DBIN and $\overline{WR}$ to generate four timing signals $\overline{MEM\ R}$ (Memory Read), $\overline{MEM\ W}$ (Memory Write), $\overline{I/O\ R}$ (I/O Read), and $\overline{I/O\ W}$ (I/O Write). However, the 8228 and 8238 do not provide the outputs of the status latch. The timing between the address bus, data bus, $\overline{WR}$, and SYNC



6800 MICROPROCESSOR	MINIMUM TIMES (nS)				
	t_{DDW} (MAX)	t_1	t_{AH}	t_2	t_H
6800, t_{CYC} = 1000	225	630	30	225	10
68A00, t_{CYC} = 666	200	420	30	80	10
68B00, t_{CYC} = 500	160	290	30	60	10

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

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

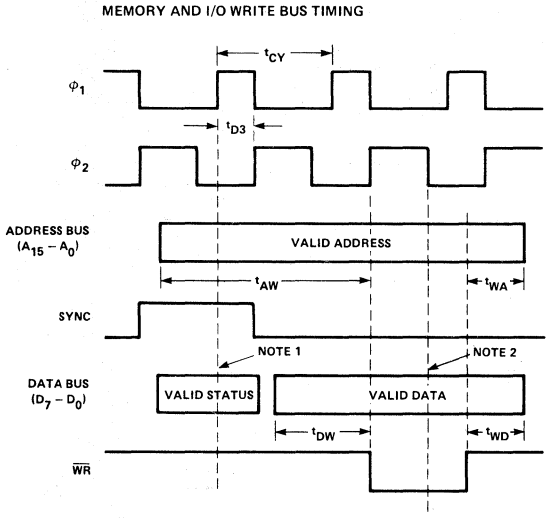
From MOTOROLA SEMICONDUCTOR MC6800 Data Sheet (DS9471), 1978

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

for both a memory write and an I/O write is shown in Figure 4. The 8080A also provides an input, READY, which allows the memory system to extend the time the address and data bus is valid by integral clock cycles.

REFRESH CONTROLLERS

Figure 5 shows a REFRESH CONTROLLER for a 16 character 18 segment alphanumeric display. The circuit operates by interrupting the microprocessor at a 1600 Hz rate. Following each interrupt, the microprocessor responds by outputting a new ASCII character to the Texas Instruments AC5947 ASCII to 18 Segment Decoder/Driver and a new digit word to the 74LS174. The character font for the AC5947 is shown in Figure 6. The outputs of the 74LS174 are decoded such that digit word 00_{16} turns the leftmost display character on, digit word $0F_{16}$ turns the rightmost display character on, and digit word $1F_{16}$ turns all digits off. The interface can be expanded to 24 characters with an additional Signetics NE590 driver. This change would also require modifications in I_F peak, and the interrupt rate.



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

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

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

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

From INTEL Component Data Catalog, 1978

NOTE 1: Status Word should be loaded into an octal latch when SYNC = 1 on positive edge of ϕ_1.

NOTE 2: Additional wait cycles can be inserted here. A wait cycle is added by forcing READY low prior to the falling edge of ϕ_2 during the clock cycle preceeding the falling edge of $\overline{WR}$.

Figure 4. Memory and I/O Write Timing for the Intel 8080A Microprocessor Family

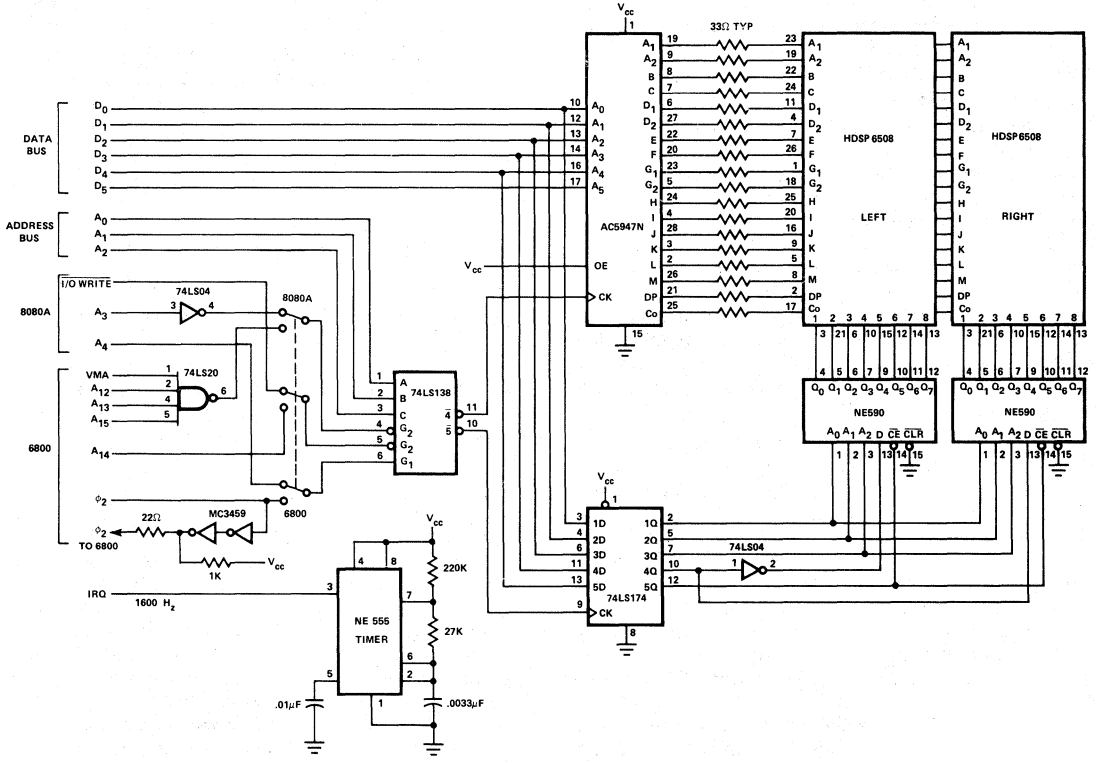


Figure 5. 6800 or 8080A Microprocessor Interface to the HDSP-6508 REFRESH CONTROLLER Utilizing the Texas Instruments AC5947 ASCII to 18 Segment Decoder/Driver

BITS	D_3	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
	D_2	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
	D_1	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
	D_0	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1
D_6 D_5 D_4	HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 1 0	2	(space)	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0 1 1	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
1 0 0	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1 0 1	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	↑	←

Figure 6. 18 Segment Display Font for the Texas Instruments AC5947 ASCII to 18 Segment Decoder/Driver

A 6800 microprocessor program that interfaces to this REFRESH controller is shown in Figure 7. Following each interrupt, the program "RFRSH" is executed. The program uses a scratch pad register "POINT" that points to the location within a 16 byte ASCII message of the next ASCII character to be stored in the display interface. The scratch pad register "DIGIT" contains the next digit word to be loaded into the display interface. The program interfaces to the circuit through two memory or I/O addresses. A memory write to address "SEG" writes a six bit word into the AC5947, and a memory write to address "DIG" writes a five bit word into the 74LS174. To prevent undesirable ghosting, the digit drivers are turned off prior to loading the next ASCII character into the AC5947. After sufficient delay, the next digit is turned on. Registers "POINT" and "DIGIT" are then updated by the program. Following execution of the "RTI" instruction, execution of the main program is resumed. A similar program written for an 8080A microprocessor is shown in Figure 8. The 6800 microprocessor program shown in Figure 7 operated with a 1 MHz clock requires 0.11% + 0.72n% of the available microprocessor time to refresh the display at a 100 Hz refresh rate, where n is the display length. The 8080A microprocessor program shown in Figure 8 when operated with a 2 MHz clock requires 0.31% + 0.96n% of the available microprocessor time to refresh the display at a 100 Hz refresh rate, where n is the display length. For example, the 16 character display shown in Figure 5

```
LOC   OBJECT CODE        SOURCE STATEMENTS
      BF04       SEG     EQU   $BF04
      BF05       DIG     EQU   $BF05

0000  0003       POINT   FDB   DATA
0002  00         DIGIT   FCB   0
0003             DATA    RMB   16

0400             RFRSH   ORG   $0400
0400  DE  00              LDX   D,POINT
0402  E6  00              LDA B X,0
0404  86  1F              LDA A I,$1F
0406  B7  BF05            STA A E,DIG
0409  F7  BF04            STA B E,SEG
040C  96  02              LDA A D,DIGIT
040E  81  0F              CMP A I,15
0410  27  0A              BEQ   LOOP1
0412  7C  0002            INC   E,DIGIT
0415  08                  INX
0416  B7  BF05            STA A E,DIG
0419  DF  00              STX   D,POINT
041B  3B                  RTI
041C  7F  0002  LOOP1     CLR   E,DIGIT
041F  F6  0001            LDA B E,POINT+1
0422  B7  BF05            STA A E,DIG
0425  C0  0F              SUB B I,15
0427  D7  01              STA B D,POINT+1
0429  24  03              BCC   LOOP2
042B  7A  0000            DEC   E,POINT
042E  3B        LOOP2     RTI
```

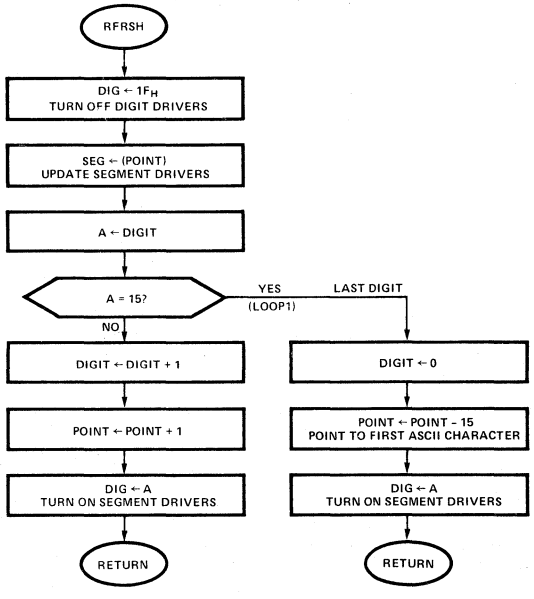


Figure 7. 6800 Microprocessor Program and Flowchart that Interfaces to the REFRESH CONTROLLER Shown in Figure 5

```
LOC   OBJECT CODE        SOURCE STATEMENTS

001C                SEG   EQU   001CH
001D                DIG   EQU   001DH

                          ORG   0E000H
E000  03  E0        POINT DW    DATA
E002  00            DIGIT DB    00H
E003  00            DATA  DS    16

                          ORG   0E400H
E400  F5            RFRSH PUSH  PSW
E401  E5                  PUSH  H
E402  2A  00E0            LHLD  POINT
E405  3E  1F              MVI   A,1FH
E407  D3  1D              OUT   DIG
E409  7E                  MOV   A,M
E40A  D3  1C              OUT   SEG
E40C  3A  02E0            LDA   DIGIT
E40F  D3  1D              OUT   DIG
E411  FE  0F              CPI   15
E413  CA  21E4            JZ    LOOP1
E416  3C                  INR   A
E417  32  02E0            STA   DIGIT
E41A  23                  INX   H
E41B  22  00E0      LOOP2 SHLD  POINT
E41E  E1                  POP   H
E41F  F1                  POP   PSW
E420  C9                  RET
E421  3E  00        LOOP1 MVI   A,0
E423  32  02E0            STA   DIGIT
E426  7D                  MOV   A,L
E427  D6  0F              SUI   15
E429  6F                  MOV   L,A
E42A  D2  1BE4            JNC   LOOP2
E42D  25                  DCR   H
E42E  C3  1BE4            JMP   LOOP2
```

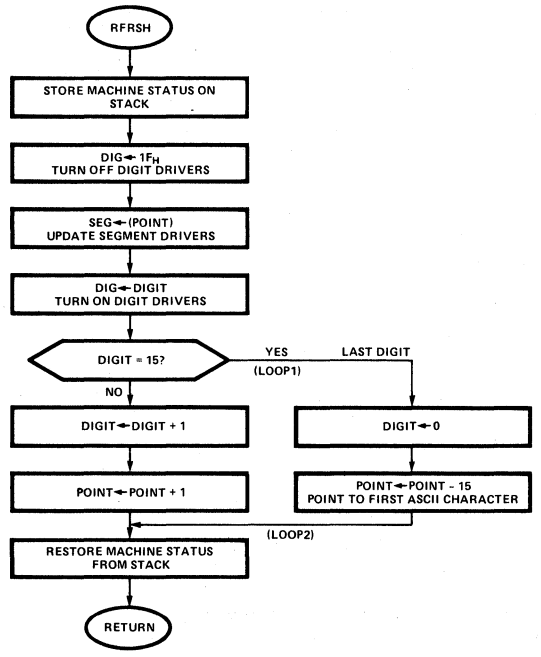


Figure 8. 8080A Microprocessor Program and Flowchart that Interfaces to the REFRESH CONTROLLER Shown in Figure 5

requires 11.6% of the 6800 microprocessor time or 15.7% of the 8080A microprocessor time to refresh the display at a 100 Hz refresh rate. Faster versions of the 6800 and 8080A microprocessors can reduce this microprocessor time by 50%.

DECODED CONTROLLERS

Figure 9 shows a DECODED DATA CONTROLLER designed for a 32 character 18 segment alphanumeric display. To simplify the circuitry, the display is configured as a 14 segment display with decimal point and colon. This allows each display character to be specified by two 8 bit words. One possible display font is shown in Figure 10. The Motorola 6810 RAM stores 64 bytes of display data that are continually read and displayed. The display data is organized within the RAM such that addresses A_5, A_4, A_3, A_2, and A_1 specify the desired character and address A_0 differentiates between the two words of display data for each character. The display data is formatted such that word 0 (D_7—D_0) is decoded as G_2, G_1, F, E, D, C, B, and A; and word 1 (D_7—D_0) is decoded as COLON, DP, M, L, K, J, I, and H. The display data is coded low true such that a low output turns the appropriate segment on. Strobing of the display is accomplished with the 74LS14 oscillator and 74LS393 counter. The counter continuously reads display data from the RAM and enables the appropriate digit driver. The time allotted to each digit is broken into four segments. During the first segment of time, the display is turned off and work 0 is read from the RAM and stored in the 74LS273 octal register. During the next three segments of time, word 1 is read from the RAM and the display is turned on. Thus, the display duty factor is (1/32) (3/4) or 1/42.6. For values of R and C specified, the display is strobed at a 130 Hz refresh rate.

Data is entered into the RAM from the address and data bus of the microprocessor via two control lines, Chip Select and Write. When Chip Select goes low, the address generated by the counter is disabled and the microprocessor address and data bus is gated to the RAM. Then, after sufficient delay, the Write input is pulsed, which stores the data within the RAM. The data entry timing for the 18 segment DECODED DATA CONTROLLER is shown in Figure 11. Because of the requirement that the address inputs of the 6810 RAM must be stable prior to the falling edge of Write, Chip Select should go low for time t_{CW} prior to the falling edge of Write. To guarantee that the address and data inputs of the RAM remain stable until after Write goes high, Chip Select should remain low for time t_{CH} following the rising edge of Write. This requirement for two separate timing signals is also required for the CODED DATA CONTROLLER shown in Figure 15. Because this interface timing is somewhat more difficult than the previously described circuits, the following methods are presented for interfacing to commonly used microprocessors.

Interface to the 6800 microprocessor family is accomplished by NANDing together VMA and some specified combination of high order address lines to generate Chip Select and using ϕ_2 to generate Write.

For the 8080A and 8085A microprocessor families, the limited flexibility of the output instruction requires that the 18 segment DECODED DATA CONTROLLER must be addressed as memory instead of I/O. The 8080A micro-

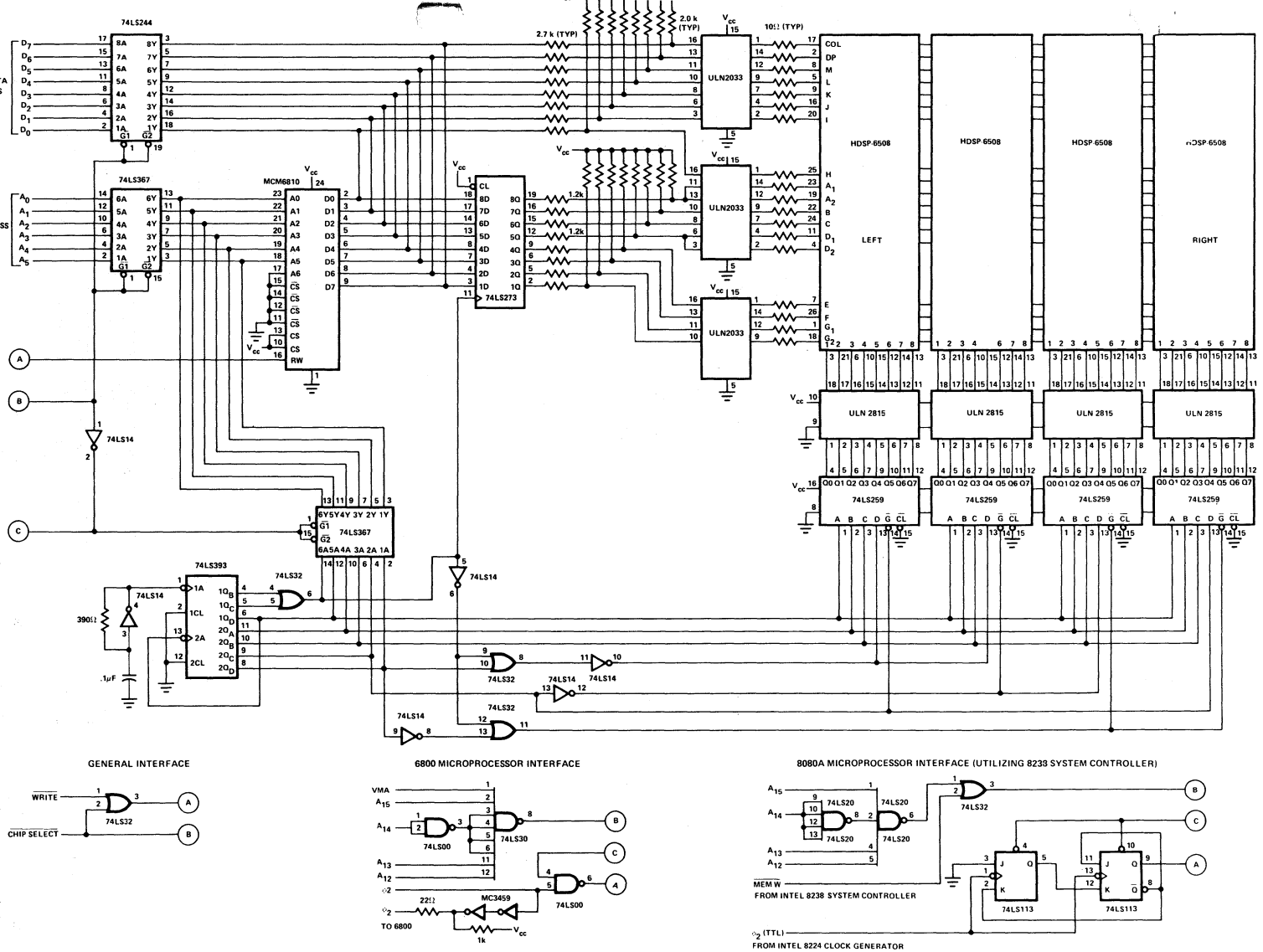


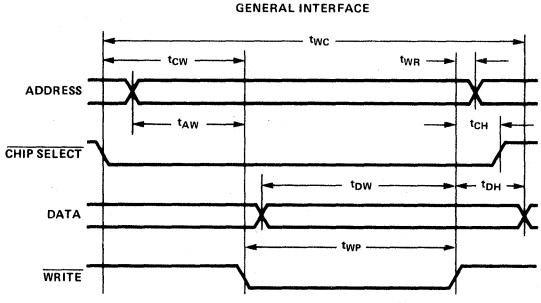
Figure 9. 6800, 8080A, and General Interface to the HDSP-6508 DECODED DATA CONTROLLER

BITS	D_3 D_2 D_1 D_0	0 0 0 0	0 0 0 1	0 0 1 0	0 0 1 1	0 1 0 0	0 1 0 1	0 1 1 0	0 1 1 1	1 0 0 0	1 0 0 1	1 0 1 0	1 0 1 1	1 1 0 0	1 1 0 1	1 1 1 0	1 1 1 1
D_6 D_5 D_4	HEX	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0 1 0	2	(space)	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0 1 1	3	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
1 0 0	4	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1 0 1	5	P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_

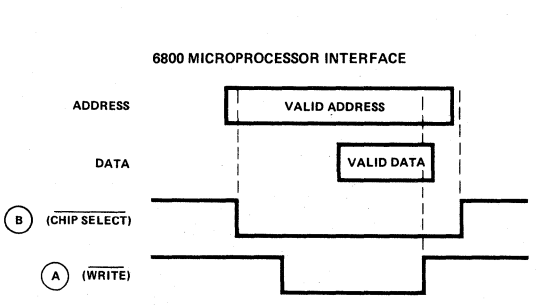
Figure 10. One Possible 16 Segment Display Font (14 Segments Plus Decimal Point and Colon) for the DECODED DATA CONTROLLER Shown in Figure 9.

processor requires an external status latch to hold status information provided during program execution. This status latch function can be implemented with an octal register such as the Intel 8212 or 74LS273. A Memory $\overline{\text{Write}}$ signal can be generated by NORing together all outputs of this status latch. This signal can then be NANDed with some specified combination of high order address lines to generate $\overline{\text{Chip Select}}$. The 8080A $\overline{\text{WR}}$ output can then be connected to $\overline{\text{Write}}$. The Intel 8238 System Controller, which is commonly used with the 8080A microprocessor, prevents direct access to the outputs of the status latch. An example of an interfacing to a system utilizing the 8238 is illustrated in Figure 9. $\overline{\text{MEM W}}$ from the 8238 is inverted and then NANDed with some specified combination of high order address lines to generate $\overline{\text{Chip Select}}$. The 74LS113 generates $\overline{\text{Write}}$ from the microprocessor clock, $\phi 2$ (TTL).

Interface to the 8085A microprocessor family can be accomplished by inverting the I/O/$\overline{\text{M}}$ output and NANDing the resulting signal with the S_0 output and some specified combination of high order address lines to generate $\overline{\text{Chip Select}}$. The $\overline{\text{WR}}$ output from the microprocessor is connected directly to $\overline{\text{Write}}$.



PARAMETER	SYMBOL	MIN.
WRITE CYCLE	t_{WC}	425ns
WRITE DELAY	t_{AW}	65ns
CHIP ENABLE TO WRITE	t_{CW}	65ns
DATA SETUP	t_{DW}	210ns
DATA HOLD	t_{DH}	35ns
WRITE PULSE	t_{WP}	325ns
WRITE RECOVERY	t_{WR}	25ns
CHIP ENABLE HOLD	t_{CH}	35ns



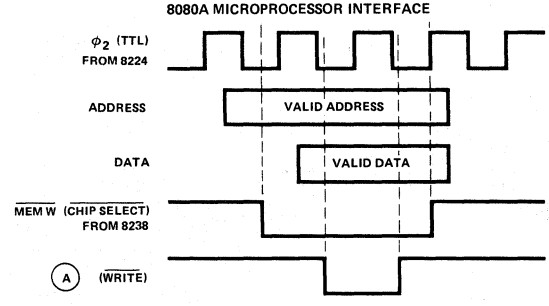


Figure 11. Data Entry Timing for the DECODED DATA CONTROLLER Shown in Figure 9

```
LOC   OBJECT CODE          SOURCE STATEMENTS

      BF00          DSPLY  EQU   $BF00
      0600          DECDR  EQU   $0600

0000  0006          ASCII  FDB   MESSGE
0002  BF00          PAD1   FDB   DSPLY
0004  0600          PAD2   FDB   DECDR
0006                MESSGE RMB   32

0400                       ORG   $0400
0400  CE  BF00       LOAD   LDX   I,DSPLY
0403  DF  02                STX   D,PAD1
0405  CE  0600              LDX   I,DECDR
0408  DF  04                STX   D,PAD2
040A  DE  00       LOOP1   LDX   D,ASCII
040C  A6  00               LDA A X,0
040E  08                   INX
040F  DF  00               STX   D,ASCII
0411  48                   ASL A
0412  97  05               STA A D,PAD2+1
0414  DE  04               LDX   D,PAD2
0416  A6  00               LDA A X,0
0418  E6  01               LDA B X,1
041A  DE  02               LDX   D,PAD1
041C  A7  00               STA A X,0
041E  08                   INX
041F  E7  00               STA B X,0
0421  08                   INX
0422  DF  02               STX   D,PAD1
0424  8C  BF40             CPX   I,DSPLY+64
0427  26  E1               BNE   LOOP1
0429  39                   RTS
```

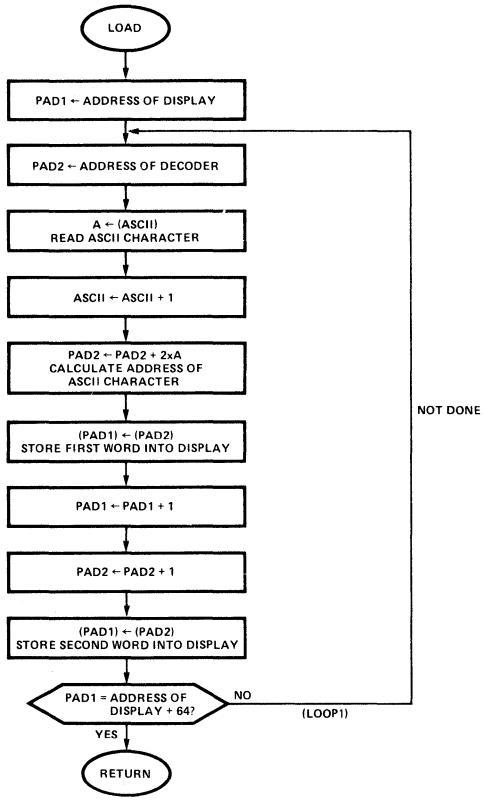


Figure 12. 6800 Microprocessor Program and Flowchart that Interfaces to the DECODED DATA CONTROLLER Shown in Figure 9

The simplest interface to the Z-80 microprocessor family is accomplished by addressing the 18 segment DECODED DATA CONTROLLER as I/O instead of memory. An example of this interface is shown in Figure 15. The $\overline{IORQ}$ output is inverted and NANDed with some specified combination of address lines to generate $\overline{Chip\ Select}$. The 74LS113 circuit generates $\overline{Write}$ from the inverted microprocessor clock $\overline{\phi}$.

A 6800 microprocessor program that interfaces to the 18 segment DECODED DATA CONTROLLER is shown in Figure 12. This program decodes 32 ASCII characters and stores the resulting decoded display data within the display. The scratch pad register "ASCII" points to the location of the next ASCII character to be decoded. The program reads the first ASCII character, increments the point, "ASCII," and then looks up two words of display data within the 64 character ASCII look-up table "DECDR." These words of display data are then stored at the two addresses for the leftmost display location. Subsequent ASCII characters are decoded, and stored at the appropriate address within the display until all 32 characters have been decoded. After the program is finished, the pointer "ASCII" will have been incremented by 32. This program requires 2.4 ms for a 1 MHz clock to decode and load 32 ASCII characters into the 18 segment DECODED DATA CONTROLLER. The corresponding 8080A microprocessor program is shown in Figure 13. This program requires 1.4 ms for a 2 MHz clock to decode and load 32 ASCII characters into the 18 segment DECODED DATA CONTROLLER.

The 64 character ASCII font shown in Figure 10 can be generated using the table shown in Figure 14. This ASCII decoder uses two 8 bit words to represent each ASCII character. The format of the decoder is consistent with either the 6800 microprocessor program shown in Figure 12 or the 8080A microprocessor program shown in Figure 13.

CODED DATA CONTROLLERS

Figure 15 shows a CODED DATA CONTROLLER designed for a 32 character 18 segment alphanumeric display. Operation of this circuit is similar to the DECODED DATA CONTROLLER shown in Figure 9 except that the Motorola 6810 RAM stores 32 six bit ASCII words and the Texas Instruments AC5947 decodes this ASCII data into 18 segment display data. The resulting display font is shown in Figure 6. Strobing of the display is accomplished by the 74LS14 oscillator and 74LS393 counter. Because the long propagation delay through the AC5947 tends to cause display ghosting, the display is

```
LOC   OBJECT CODE          SOURCE STATEMENTS

BF00                DSPLY  EQU   0BF00H

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

                           ORG   0E400H
E400  01   00BF     LOAD   LXI   B,DSPLY
E403  11   00E5            LXI   D,DECDR
E406  2A   00E0            LHLD  ASCII
E409  7E            LOOP1  MOV   A,M
E40A  23                   INX   H
E408  07                   RLC
E40C  5F                   MOV   E,A
E40D  1A                   LDAX  D
E40E  02                   STAX  B
E40F  13                   INX   D
E410  03                   INX   B
E411  1A                   LDAX  D
E412  02                   STAX  B
E413  03                   INX   B
E414  79                   MOV   A,C
E415  FE   40              CPI   64
E417  C2   09E4            JNZ   LOOP1
E41A  22   00E0            SHLD  ASCII
E41D  C9                   RET
```

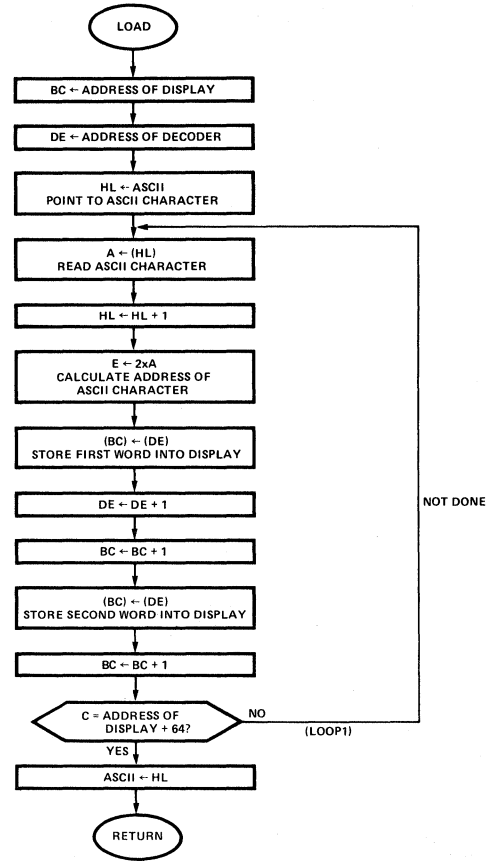


Figure 13. 8080A Microprocessor Program and Flowchart that Interfaces to the DECODED DATA CONTROLLER Shown in Figure 9

ASCII	SYMBOL	WORD 0	WORD 1
20	(SPACE)	FF	FF
21	!	FF	BD
22	"	DF	FD
23	#	36	ED
24	$	12	ED
25	%	1B	D2
26	&	F2	CA
27	'	FF	FD
28	(	FF	F3
29	)	FF	DE
2A	*	3F	C0
2B	+	3F	ED
2C	,	FF	DF
2D	-	3F	FF
2E	.	FF	BF
2F	/	FF	DB
30	0	C0	DB
31	1	FF	ED
32	2	24	FF
33	3	30	FF
34	4	19	FF
35	5	96	F7
36	6	02	FF
37	7	F8	FF
38	8	00	FF
39	9	18	FF
3A	:	FF	3F
3B	;	FF	5F
3C	<	7F	FB
3D	=	37	FF
3E	>	BF	FE
3F	?	7C	EF

ASCII	SYMBOL	WORD 0	WORD 1
40	@	44	FD
41	A	08	FF
42	B	70	ED
43	C	C6	FF
44	D	F0	ED
45	E	86	FF
46	F	8E	FF
47	G	42	FF
48	H	09	FF
49	I	F6	ED
4A	J	E1	FF
4B	K	8F	F3
4C	L	C7	FF
4D	M	C9	FA
4E	N	C9	F6
4F	O	C0	FF
50	P	0C	FF
51	Q	C0	F7
52	R	0C	F7
53	S	12	FF
54	T	FE	ED
55	U	C1	FF
56	V	CF	DB
57	W	C9	D7
58	X	FF	D2
59	Y	FF	EA
5A	Z	F6	DB
5B	[	7F	F3
5C	\	FF	F6
5D	]	BF	DE
5E	^	FF	D7
5F	_	F7	FF

Figure 14. 64 Character ASCII Decoder Table for the Microprocessor Programs Shown in Figures 12 and 13. 18 Segment Display Font is Shown in Figure 10.

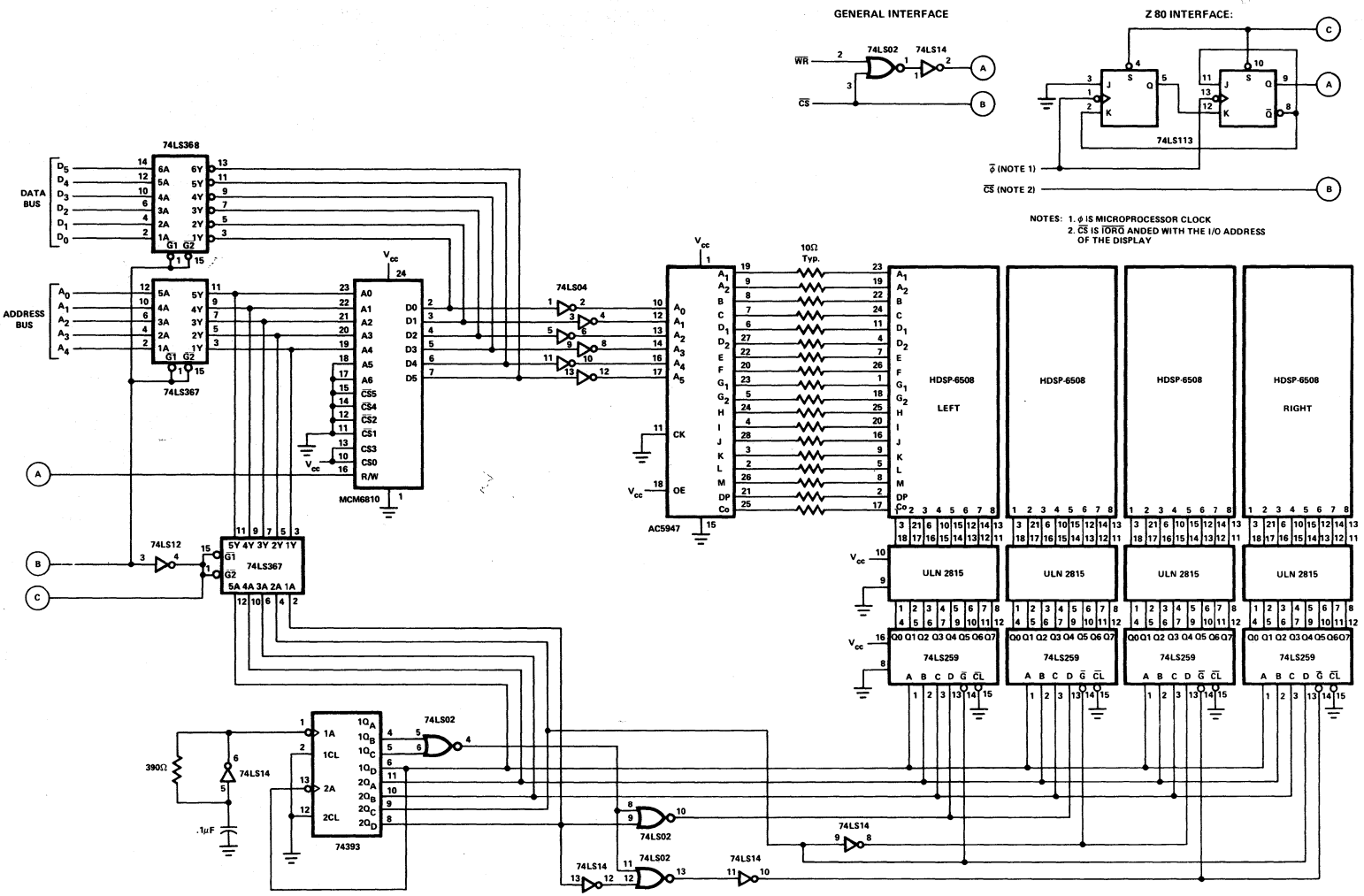
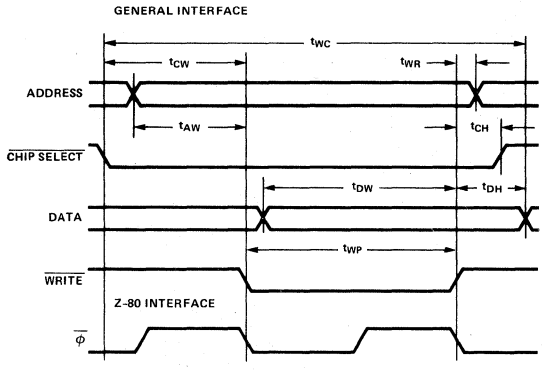


Figure 15. General Interfaces to the HDSP-6508 CODED DATA CONTROLLER



PARAMETER	SYMBOL	MIN.
WRITE CYCLE	t_{WC}	455ns
WRITE DELAY	t_{AW}	65ns
CHIP ENABLE TO WRITE	t_{CW}	65ns
DATA SETUP	t_{DW}	215ns
DATA HOLD	t_{DH}	50ns
WRITE PULSE	t_{WP}	340ns
WRITE RECOVERY	t_{WR}	40ns
CHIP ENABLE HOLD	t_{CH}	50ns

Figure 16. Data Entry Timing for the CODED DATA CONTROLLER Shown in Figure 15

blanked momentarily after each new character is read from the RAM. This is accomplished by breaking the total time allotted for each digit into four segments. During the first segment, the display is turned off to allow data to ripple through the AC5947 and during the next three segments, the display is turned on. The resulting display duty factor is (1/32) (3/4) or 1/42.6. The display is strobed at a 130 Hz refresh rate.

Data is entered into the RAM from the address and data bus of the microprocessor via two control lines Chip Select and Write. When Chip Select goes low, the address from the counter is tristated and the microprocessor address bus and data bus is gated to the RAM. Then after sufficient delay, the Write input is pulsed, which stores the data within the RAM. Data entry timing for the 18 segment CODED DATA CONTROLLER is shown in Figure 16. Since this timing is very similar to the DECODED DATA CONTROLLER shown in Figure 9, interface to the various microprocessor families is the same as described in the section on DECODED DATA CONTROLLERS.

DISPLAY PROCESSOR CONTROLLERS

The DISPLAY PROCESSOR CONTROLLER provides a powerful, smart interface which performs many of the functions normally found in a small terminal. The DISPLAY PROCESSOR CONTROLLER is designed around a slave microprocessor or custom LSI integrated circuit that provides display storage and multiplexing with a very minimum of circuit complexity. The simplest DISPLAY PROCESSOR CONTROLLER designed for a 16 digit 18 segment alphanumeric display is shown in Figure 17. This circuit is designed around the Intel 8279 Programmable Keyboard/Display Interface. This LSI chip contains the circuitry necessary to interface directly to a microprocessor bus and provides a 16 x 8 RAM, programmable scan counter, and keyboard debounce and control logic. While the 8279 is specifically designed for 7 segment displays, inclusion of the Texas Instruments AC5947 ASCII to 18 segment decoder/driver allows the use of an 18 segment alphanumeric display. The 8279 Keyboard/Display Controller interfaces to a microprocessor via an eight line bidirectional Data Bus, control lines RD (Read), WR (Write), CS (Chip Select), A_0 (Command/Data), RESET, IRQ (Interrupt Request), and a clock input, CLK. The display is scanned by outputs A_{0-3} and B_{0-3} which are connected to the inputs of the AC5947, and outputs SL_{0-3} which are connected to the digit scanning circuitry. The 74LS122 is used to provide interdigit blanking to prevent display ghosting. In addition to display scanning, the 8279 also has the ability to scan many different types of encoded or decoded keyboards, X-Y matrix keyboards, or provide a strobed data input to the microprocessor. The 8279 provides for either block data entry, where data enters from left to right across the display overflowing to the leftmost display location; right data entry, where data enters at the righthand side of the display and previous data shifts toward the left; and RAM data entry, where a four bit field in the control word specifies the address at which the next data word will be written. The 8279 allows data written into the display to be read by the microprocessor, and provides commands to either blank or clear the display.

The HDSP-8716/-8724/-8732/-8740 DISPLAY PROCESSOR CONTROLLER shown in Figure 18 is designed to provide a flexible 18 segment display interface for displays up to 40 characters in length. This circuit utilizes a dedicated Intel 8048 single chip microprocessor to provide features such as a blinking cursor, display editing routines, multiple data entry modes, variable display string length, and data out. This controller is available as a series of printed circuit board subsystems of 16, 24, 32, and 40 characters in length. The user interfaces to the 8048 microprocessor through eight Data In inputs, six Address inputs, a Chip Select input, Reset input, Blank input, six Data Out outputs, Data Valid output, Refresh output, and Clock output. The software within the 8048 microprocessor provides four data entry modes — Left Entry with a blinking cursor, Right Entry, Block Entry, and RAM Entry. The Data Out port allows the user to read the ASCII data stored within the display, determine the configured data entry mode and display length, and locate the position of the cursor within the display. Since the Data Out port is separate from the Data In port, the 18 segment DISPLAY PROCESSOR CONTROLLER can be used for text editing independent of the main microprocessor system. In Left Entry mode, the controller provides the Clear, Carriage Return, Backspace, Forwardspace, Insert, and Delete editing functions; while in Right Entry mode, the controller provides Clear and Backspace editing functions. The controller can also be expanded into multiple line panels.

The 8048 microprocessor interfaces to the display via the Port 2 output. The output is configured to enable the microprocessor to send a six bit word to one of three destinations as selected by P_{26} and P_{27}. The PROG output

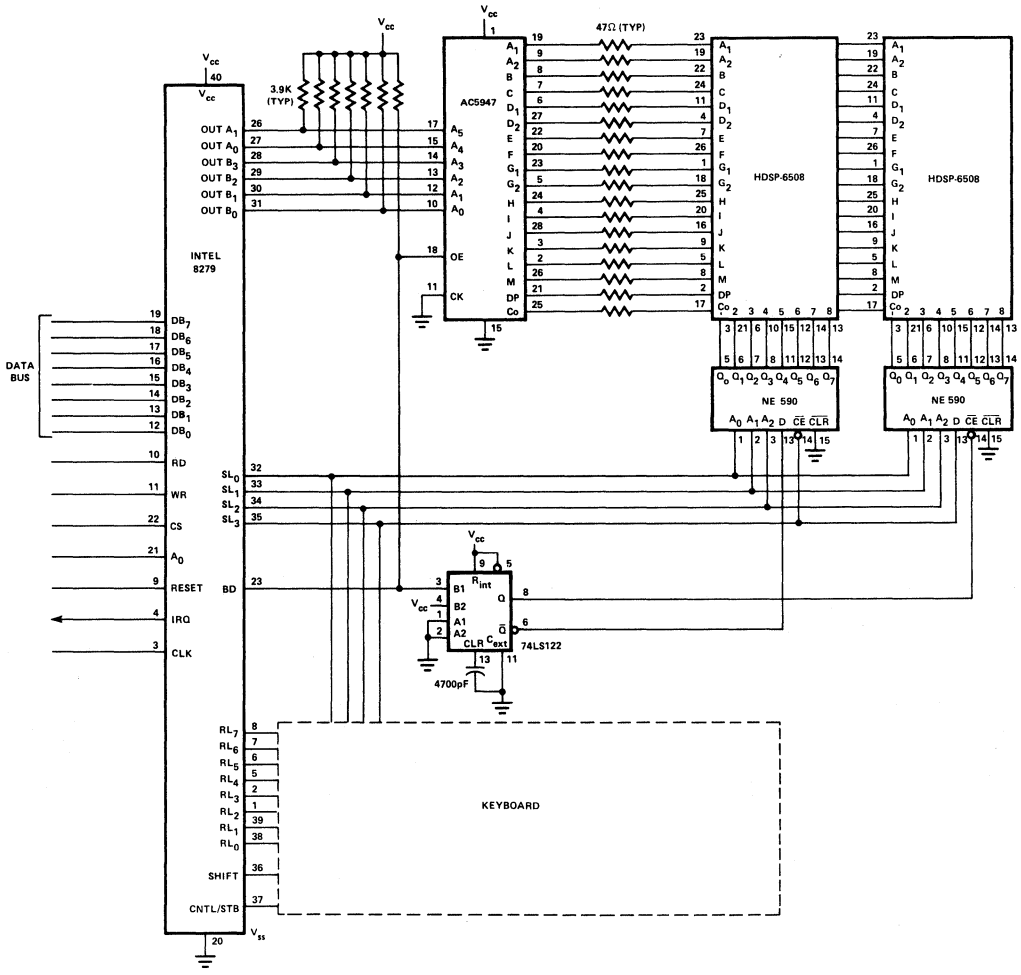


Figure 17. HDSP-6508 DISPLAY PROCESSOR CONTROLLER Utilizing the Intel 8279 Programmable Keyboard Display Interface

is then used to store this word at the specified destination. Destination$_0$ is the 74LS174 hex register. The outputs of this register are decoded by the 74LS259 addressable latches and Sprague ULN 2815 digit drivers. Output $3F_{16}$ is decoded to turn on the rightmost display digit while the address of the leftmost display digit varies from 18_{16} for a 40 character display to 30_{16} for a 16 character display. Destination$_1$ is the AC5947 18 segment decoder/driver. The positive edge of $\overline{\text{PROG}}$ stores a six bit ASCII code within the AC5947. Because destination$_1$ is pulsed once every time a digit is refreshed, this output is also used as the $\overline{\text{Refresh}}$ output. Destination$_2$ is the Data Valid output of the Data Out port. Thus, Data Out actually consists of a series of six bit words that are sent to Destination$_2$. Display refresh is accomplished by first turning off the digit drivers by outputting a 0_{16} to the 74LS174. Then a new ASCII character is stored within the AC5947. Finally, a new digit word is stored within the 74LS174. The actual time that each digit is on varies according to the configured display length so as to provide a fixed 100 Hz refresh rate.

Interfacing the DISPLAY PROCESSOR CONTROLLER shown in Figure 18 to microprocessor systems depends on the needs of the particular application. Since the information on the Data In and Address inputs is loaded into the controller through a program within the 8048 microprocessor, the time required to read these inputs varies from about 100 to 700 microseconds. A latch as shown in the HDSP-8716/-8724/-8732/-8740 Data Sheet can be used as a buffer between these inputs and the data bus and address bus of the main microprocessor system. The latch provides temporary storage to avoid making the main microprocessor wait for the DISPLAY PROCESSOR CONTROLLER to accept data.

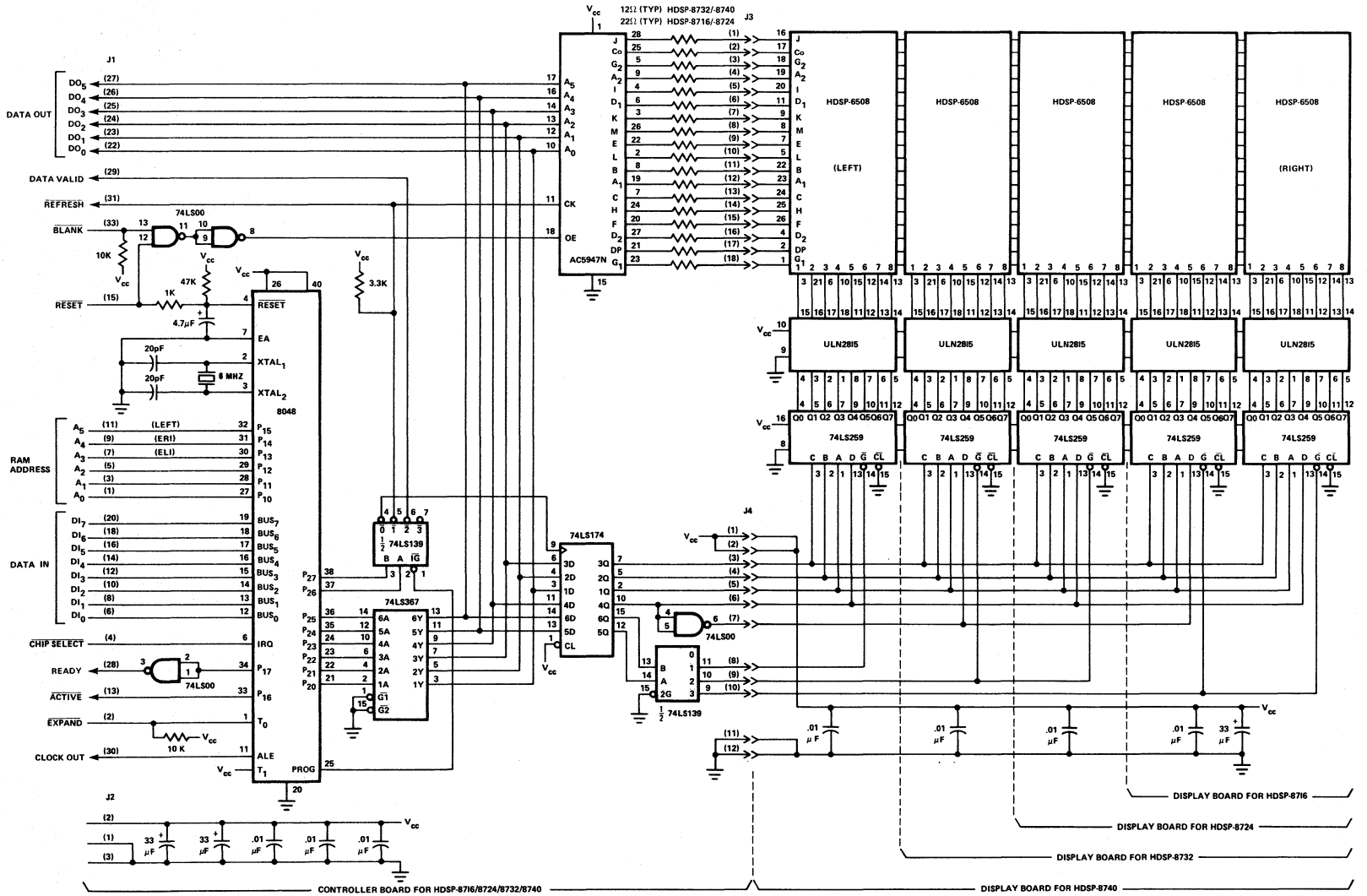


Figure 18. HDSP-8716/-8724/-8732/-8740 DISPLAY PROCESSOR CONTROLLER

The 18 segment DISPLAY PROCESSOR CONTROLLER shown in Figure 18 can also be interfaced to the main microprocessor system through a Peripheral Interface Adapter (PIA). The Data In inputs of the controller would be connected to an output port of the PIA. In RAM Entry mode, the Address inputs of the controller would be connected to another output port of the PIA. The PIA provides a handshake back to the main microprocessor system that tells when the DISPLAY PROCESSOR CONTROLLER is ready to accept another data input word from the main microprocessor. This allows the microprocessor to load data into the controller at the highest possible rate. A PIA can also be used to allow the 18 segment DISPLAY PROCESSOR CONTROLLER to act as a buffer between a keyboard and the main microprocessor. In this configuration, the main processor could output a prompting message to the user via the DISPLAY PROCESSOR CONTROLLER. The user could then enter data from the keyboard into the display utilizing the controller's editing capability. After the message has been entered and edited, the user would instruct the main microprocessor to read the final edited message from the Data Out port. One port from the PIA can be used to control the Data In inputs of the DISPLAY PROCESSOR CONTROLLER and another port of the PIA can be used to read the Data Out port. Figure 19 shows a 6800 microprocessor system using a Motorola 6821 PIA to control the DISPLAY PROCESSOR CONTROLLER shown in Figure 18. The PB_7 output of the PIA determines whether data is entered into the controller

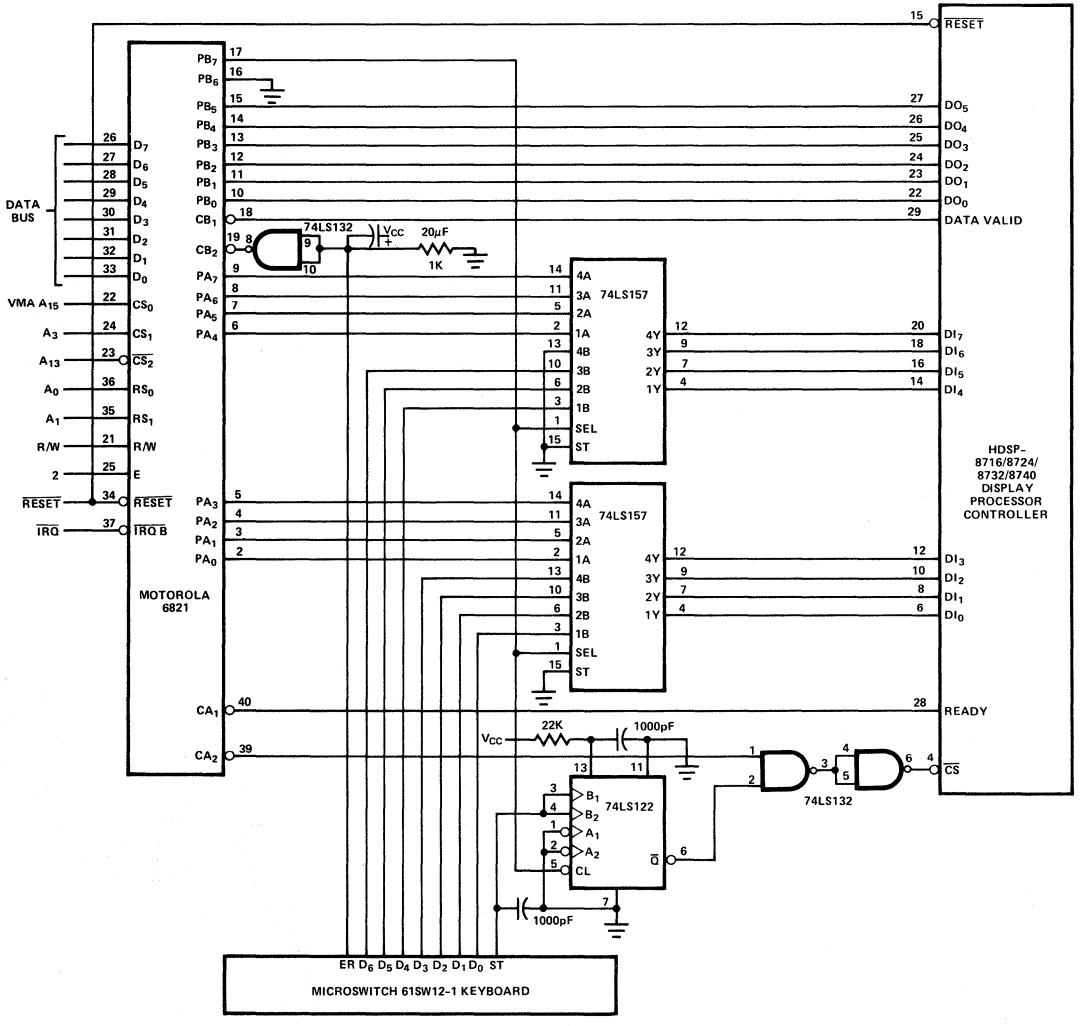


Figure 19. 6800 Microprocessor Interface to the DISPLAY PROCESSOR CONTROLLER Shown in Figure 18 Utilizing a Motorola 6821 PIA

```
* PORT CONFIGURATION:
* 1. PORT A:
*        PA0-PA7 OUTPUTS TO DATA IN OF HDSP-87XX
*        CA1 (INPUT) MODE 00 SETS FLAG NEG EDGE OF READY
*        CA2 (OUTPUT) MODE 100 CLEARED MPU READ PRA, SET
*        NEGATIVE EDGE OF READY
* 2. PORT B:
*        PB0-PB5 INPUTS DATA TO 6800 FROM DATA OUT OF HDSP
*        CB1 (INPUT) MODE 10 SETS FLAG POS EDGE OF DATA VA
*        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 AND KEY

          8008      PRA     EQU     $8008
          8008      DRA     EQU     $8008
          8009      CRA     EQU     $8009
          800A      PRB     EQU     $800A
          800A      DRB     EQU     $800A
          800B      CRB     EQU     $800B
          0028      LENGTH  EQU     40          MUST BE SAME AS LENGTH

0000                        ORG     $0000
0000 0002           MESSGE  FDB     TEXT

0100                        ORG     $0100
0100                STATUS  RMB     1
0101                CURSOR  RMB     1
0102                DATA    RMB     40


0400                        ORG     $0400
0400 CE 0100        READ    LDX     I,STATUS
0403 7F 800A                CLR     E,PRB       ENABLE PORT A TO MUX
0406 86 FF                  LDA A   I,$FF
0408 B7 8008                STA A   E,PRA       BEGIN DATA OUT SEQUENCE
040B 7D 8008                TST     E,PRA       CLEAR CA1 AND CA2
040E 7D 800A                TST     E,PRB       CLEAR CB1 AND CB2
0411 C6 2A                  LDA B   I,LENGTH+2
0413 B6 800B        LOOP1   LDA A   E,CRB
0416 2A FB                  BPL     LOOP1       WAIT FOR DATA VALID
0418 B6 800A                LDA A   E,PRB
041B 84 3F                  AND A   I,$3F
041D A7 00                  STA A   X,0         STORE IN RAM
041F 08                     INX
0420 5A                     DEC B
0421 26 F0                  BNE     LOOP1       NEXT DATA OUT WORD
0423 7D 8008                TST     E,PRA       CLEAR CA1 AND CA2
0426 B6 8009        LOOP2   LDA A   E,CRA
0429 2A FB                  BPL     LOOP2       WAIT UNTIL READY
042B 39                     RTS

042C DE 00          LOAD    LDX     D,MESSGE
042E A6 00          LOOP10  LDA A   X,0
0430 08                     INX
0431 81 FF                  CMP A   I,$FF
0433 27 0D                  BEQ     ENDL        JUMP WHEN DONE
0435 B7 8008                STA A   E,PRA
0438 7D 8008                TST     E,PRA       CLEAR CA1 AND CA2
043B B6 8009        LOOP11  LDA A   E,CRA
043E 2A FB                  BPL     LOOP11      WAIT
0440 20 EC                  BRA     LOOP10
0442 DF 00          ENDL    STX     D,MESSGE
0444 39                     RTS

0500                        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 06                  LDA A   I,$06
0517 B7 800B                STA A   E,CRB
051A 0E             MAIN    CLI
051B 7F 800A                CLR     E,PRB       DISABLE KEYBOARD FROM MUX
051E BD 042C                JSR     E,LOAD
0521 7D 800A                TST     E,PRB       CLEAR CB1, CB2
0524 86 80                  LDA A   I,$80
0526 B7 800A                STA A   E,PRB       ENABLE KEYBOARD TO MUX
0529 86 0E                  LDA A   I,$0E
052B B7 800B                STA A   E,CRB       ENABLE IRG
052E 0F                     SEI                 IRQ CAUSES JSR TO READ
```

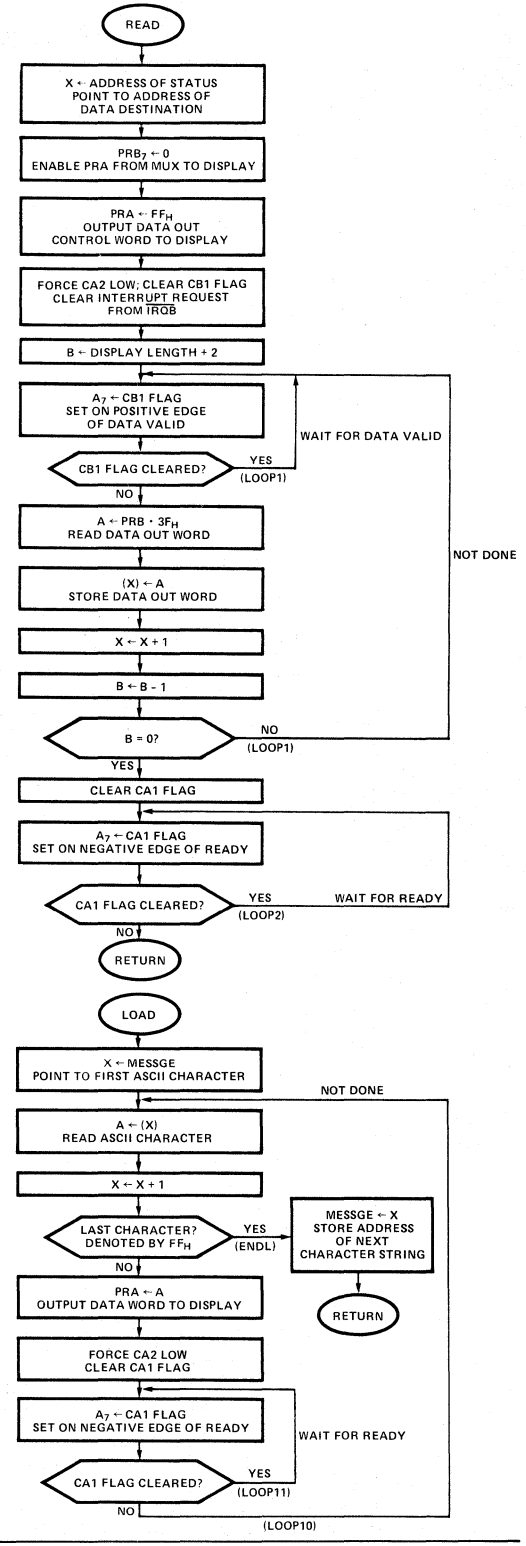


Figure 20. 6800 Microprocessor Program and Flowchart that Interfaces to the Circuit Shown in Figure 19

from the microprocessor system or from the keyboard. Control lines CA_1 and CA_2 are used to provide a data entry handshake to allow data to be loaded into the controller at the highest possible rate. Data is read into the main microprocessor system through Port B of the PIA using the CB_1 input as a data strobe.

The 6800 microprocessor program shown in Figure 20 is used to operate the PIA interface described in Figure 19. The microprocessor program following "START" is used to initialize the 6821 PIA. Once initialized, the PIA can be used either to load data into the controller via the main microprocessor, allow data to be loaded into the controller via the keyboard, or to read data from the Data Out port into the main microprocessor. The instruction CLR E, PRB at location $051B_{16}$ forces PB_7 low to connect the outputs of Port A to the Data In inputs of the controller.

Subroutine "LOAD" then loads a series of eight bit words into the controller. "LOAD" continues to output words until it reads an FF_{16} to denote the end of the prompting message. The instruction sequence LDA A I, $80 and STA A E, PRB at location 0526_{16} forces PB_7 high to connect the output of the keyboard to the Data In inputs of the controller. In this mode, the user can enter or edit data into the DISPLAY PROCESSOR CONTROLLER. The 4B input of the 74LS157 has been grounded to prevent the keyboard from loading a control word into the DISPLAY PROCESSOR CONTROLLER. The instructions LDA A I, $0E and STA A E, CRB at location $052B_{16}$ enables the "ER" key on the keyboard to interrupt the microprocessor when the edited message is complete. Subroutine "READ" would then be used to read data into the 6800 system. First, subroutine "READ" outputs a special control word,

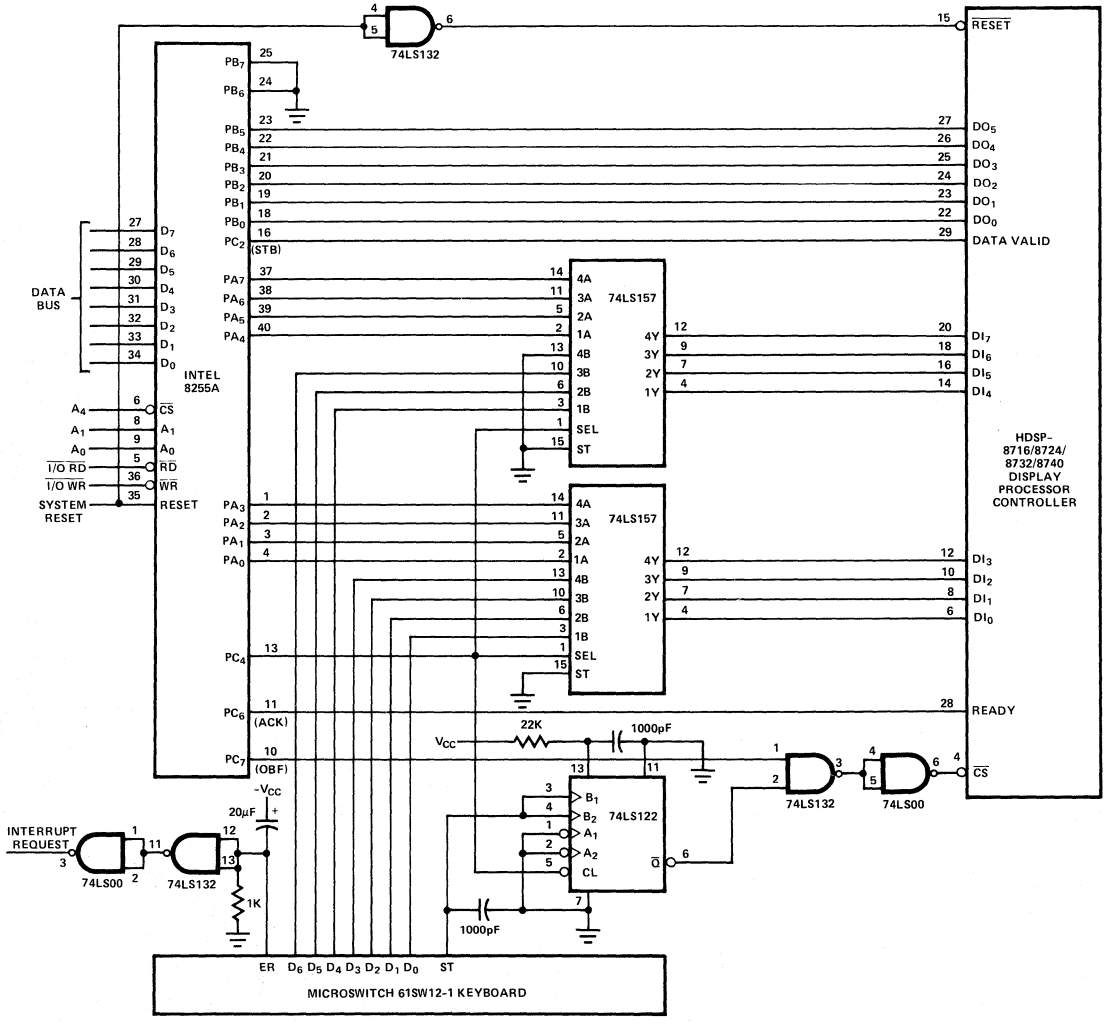


Figure 21. 8080A Microprocessor Interface to the DISPLAY PROCESSOR CONTROLLER Shown in Figure 18 Utilizing an Intel 8255 PIA

```
;THIS PROGRAM IS WRITTEN IN 8080 ASSEMBLY LANGUAGE.
;THIS PROGRAM USES AN 8255 PIA TO ACCESS THE HDSP-87XX
;ALPHANUMERIC DISPLAY SYSTEM.
;
;PORT CONFIGURATION:
;1. PORT A (MODE 1 OUTPUT):
;       PA0-PA7 OUTPUTS TO DATA IN OF HDSP-87XX
;       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-87XX
;       PC2 (STB) INPUT; LOADS DATA ON POS EDGE OF DATA VALID
;       FLAG PC0 (INTR) CLEARED BY INPUT; SET BY DATA VALID
;
;3. PORT C:
;       PC4 OUTPUT; LOW ENABLES PA0-PA7 TO HDSP-87XX
;                   HIGH ENABLES KEYBOARD TO HDSP-87XX

000C            PA      EQU     0CH
000D            PB      EQU     0DH
000E            PC      EQU     0EH
000F            CNTRL   EQU     0FH
0028            LENTH   EQU     40          MUST BE DISPLAY LENGTH
                        ORG     0E000H
E000 02 E0      ASCII   DW      TEXT
E002 00         TEXT    DS      40
                ;
                        ORG     0E100H
E100 00         STAT    DB      0
E101 00         ADDR    DB      0
E102 00         DATA    DB      0
                ;
                        ORG     0E400H
E400 F3         READ    DI
E401 F5                 PUSH    PSW
E402 E5                 PUSH    H
E403 C5                 PUSH    B
E404 3E 08              MVI     A,08H
E406 D3 0F              OUT     CNTRL       ENABLE A SIDE OF MUX
E408 3E FF              MVI     A,0FFH
E40A D3 0C              OUT     PA          BEGIN DATA OUT SEQUENCE
E40C 0E 2A              MVI     C,LENTH+2
E40E 21 00E1            LXI     H,STAT      FIRST WORD
E411 DB 0D              IN      PB          CLEAR INTR
E413 DB 0E      LOOP1   IN      PC
E415 1F                 RAR
E416 D2 13E4            JNC     LOOP1       WAIT UNTIL INTR IS SET
E419 DB 0D              IN      PB
E41B 77                 MOV     M,A         STORE IN RAM
E41C 23                 INX     H
E41D 0D                 DCR     C
E41E C2 13E4            JNZ     LOOP1       READ LENGTH+2 WORDS
E421 DB 0E      LOOP2   IN      PC
E423 17                 RAL
E424 D2 21E4            JNC     LOOP2       WAIT UNTIL READY
E427 C1                 POP     B
E428 E1                 POP     H
E429 F1                 POP     PSW
E42A FB                 EI
E42B C9                 RET
                ;
E42C 2A 00E0    LOAD    LHLD    ASCII       FIRST WORD OF MESSAGE
E42F 7E         LOOP5   MOV     A,M
E430 FE FF              CPI     0FFH        CHECK TO SEE IF DONE
E432 CA 44E4            JZ      ENDL
E435 D3 0C              OUT     PA          OUTPUT TO DISPLAY
E437 23                 INX     H
E438 DB 0E      LOOP6   IN      PC
E43A 17                 RAL
E43B D2 38E4            JNC     LOOP6       WAIT
E43E 00                 NOP
E43F 00                 NOP
E440 00                 NOP
E441 C3 2FE4            JMP     LOOP5       NEXT WORD
E444 23         ENDL    INX     H
E445 22 00E0            SHLD    ASCII
E448 C9                 RET
                ;
E449 3E A7      START   MVI     A,0A7H      PA AS M1 OUTPUT, PB AS M1 INPUT
E44B D3 0F              OUT     CNTRL
E44D 3E 0C              MVI     A,0CH       CLEAR INTE A
E44F D3 0F              OUT     CNTRL
E451 3E 05              MVI     A,05H
E453 D3 0F              OUT     CNTRL       SET INTE B
                ;
                ;PROCEDURE TO LOAD HDSP-87XX SYSTEM
                ;
E455 3E 08              MVI     A,08H
E457 D3 0F              OUT     CNTRL       ENABLE A SIDE OF MUX
E459 CD 2CE4            CALL    LOAD
                ;
                ;PROCEDURE TO READ DATA OUT OF HDSP-87XX SYSTEM
                ;
E45C 3E 09              MVI     A,09H
E45E D3 0F              OUT     CNTRL       ENABLE B SIDE OF MUX
E460 FB                 EI
                ;                           INTERRUPT MUST CALL READ
```

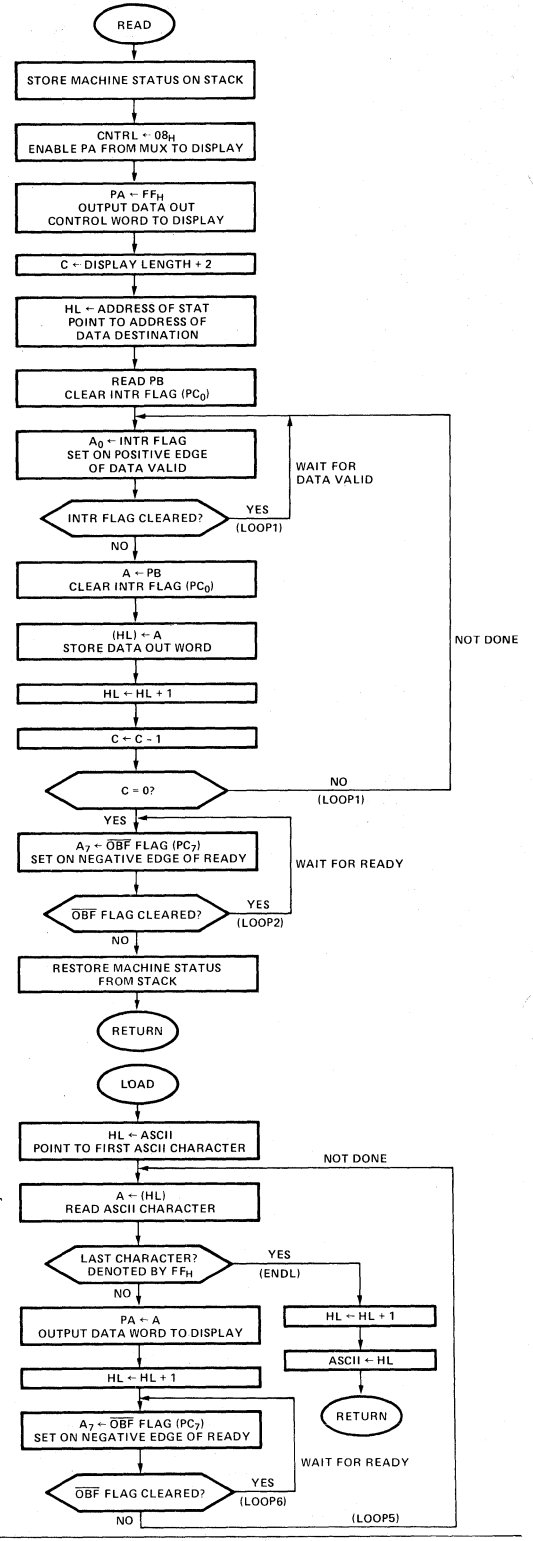


Figure 22. 8080A Microprocessor Program and Flowchart that Interfaces to the Circuit Shown in Figure 21

FF_{16}, to the Data In inputs of the DISPLAY PROCESSOR CONTROLLER. This control word causes the controller to begin its data output sequence. The controller outputs a series of data output words that define the configured entry mode and display length, location of the cursor, and the ASCII text stored within the DISPLAY PROCESSOR CONTROLLER. "LOOP 1" within the program continuously reads the Data Valid output and waits until the controller outputs the STATUS word. This STATUS word, the subsequent CURSOR ADDRESS word, and the string of ASCII characters are then stored in consecutive words of scratch pad memory starting at address "STATUS."

A similar PIA interface designed for an 8080A microprocessor system that uses an Intel 8255A PIA is shown in Figure 21. This interface operates in much the same way as the 6821 PIA interface that was previously described. The PC_4 output of the PIA determines whether the Data In inputs of the 18 segment DISPLAY PROCESSOR CONTROLLER shown in Figure 18 are connected to the PIA or to the keyboard. Control lines PC_6 and PC_7 are used to provide a data entry handshake between the 8080A microprocessor and the DISPLAY PROCESSOR CONTROLLER. Data is read into the 8080A microprocessor system through Port B of the PIA using PC_2 as the data strobe.

The 8080A microprocessor program shown in Figure 22 is used to operate the PIA interface described in Figure 21. The microprocessor program following "START" is used to initialize the 8255A PIA. The instructions MVI A, 08H and OUT CNTRL at location $E457_{16}$ force PC_4 low to connect Port A of the PIA to the Data In inputs of the DISPLAY PROCESSOR CONTROLLER. Subroutine "LOAD" would then be used to load a prompting message into the controller. The instructions MVI A, 09H and OUT CNTRL at location $E45E_{16}$ connect the keyboard to the Data In inputs of the controller. In this mode, the user can enter data into the DISPLAY PROCESSOR CONTROLLER, or to edit an existing line. Subroutine "READ" would then be used to read the data from the Data Out port into the 8080A microprocessor system.

Subroutine "READ" begins the data output sequence by outputting the special control word FF_H to the Data In inputs of the DISPLAY PROCESSOR CONTROLLER. Then, the subroutine reads the series of data output words that are outputted by the controller and stores them in consecutive words of scratch pad memory starting at address STAT.