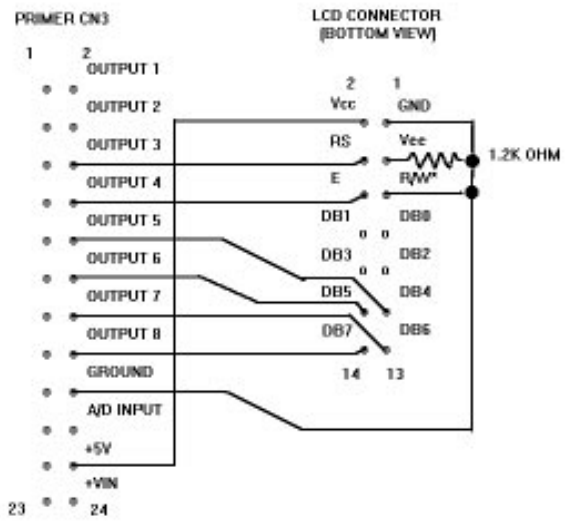
Application 7: Controlling an LCD Module

Purpose: To demonstrate writing characters and cursor positioning on an LCD Module display.

Discussion:

There are many LCD Module display manufacturers and most use the same 14 pin dual row header interface and the same controller chip, the HD44780. These modules display characters only, not graphics (with the exception that you can simulate graphics by dynamically defining your own characters). You may find these displays in surplus catalogs, or parts catalogs such as DIGI-KEY. Some example parts are:

DIGI-KEY Part.	Description	(Call 1-800-DIGI-KEY)
OP116-ND	OPTREX 16x1 standard LCD dot matrix module	
VT216-ND	Varitronix Ltd 16x2 standard LCD dot matrix module	



The HD44780 controller has two registers: one for data and one for commands. The data register allows you to write characters to the display, define your own characters and read display memory. The command register allows writing of several commands relating to display control and initialization and also reading the controller's status and address counter. In the interest of simplicity we will write to the controller registers in this application.

The controller can transfer data in 8 or 4 bit mode, so we will use it in 4 bit mode since we have only 8 output ports and we need at least 4 to transfer data (DB4 to DB7) and 2 for the control lines (RS and E).

```
;
; LCD DRIVER CODE
;
OPORT    EQU      11H         ;OUTPUT PORT
IPORT    EQU      12H         ;INPUT PORT
KEYIN    EQU      0BH         ;SERVICE FOR READING KEYPAD
MOS      EQU      1000H       ;MOS CALL ADDRESS

;
; OPORT BITS ARE DEFINED AS FOLLOWS:
;  7   6   5   4   3   2   1   0
; DB7 DB6 DB5 DB4  E   RS (not used)
```

```
;
          ORG       0FF01H
          MVI       A,11110011B ; RS, E, = 0.
          OUT       OPORT

          ; RESET CODE
          CALL      DELAY
          CALL      DELAY
          MVI       A,30H
          CALL      DLNOUT
          CALL      DLNOUT
          CALL      DLNOUT

          ; INIT CODE
          MVI       A,00100000B ;SET 4 BIT MODE
          CALL      DLNOUT

          MVI       A,00101000B ;SET 4 BIT, 2 LINE, 5 BY 7 DOTS
          CALL      OUTCMD
          MVI       A,00001000B ;DISPLAY OFF
          CALL      OUTCMD
          MVI       A,00000001B ;DISPLAY ON
          CALL      OUTCMD
          MVI       A,00001110B ;TURN ON DISPLAY, CURSOR, AND BLINK.
          CALL      OUTCMD
          MVI       A,00000110B ;ENTRY MODE SET. INC. W/CURSOR MOVEMENT
          CALL      OUTCMD

          LXI       H,TSTSTR
          CALL      SHWSTR

LOOP:     NOP
          NOP
          NOP
          NOP
          NOP                   ;THESE ARE PLACE HOLDERS

          MVI       C,KEYIN
          CALL      MOS         ;GET A KEY
          MVI       A,'0'
          ADD       L           ;CONVERT 0 TO 9 IN L TO ASCII
          CALL      OUTDTA      ;DISPLAY THE CHAR
          JMP       LOOP

TSTSTR:   DB 'The Primer.',0

;
; Show the string pointed to by HL.  When 0 is encountered the program exits
; returning HL pointing to the byte after the 0.
;
SHWSTR:   MOV       A,M         ;READ STRING
          INX       H           ;CHANGE POINTER
          ORA       A           ;SEE IF A=0
          RZ                    ;EXIT IF END OF STRING
          CALL      OUTDTA      ;DISPLAY CHARACTER
          JMP       SHWSTR

;
; Send A to the LCD with RS=1, high nibble first and low second.
;
OUTDTA:   MVI       E,0100B     ;SET RS
          JMP       OBYT1
;
; Send A to the LCD with RS=0, high nibble first and low second.
;
OUTCMD:   MVI       E,0         ;RS=0
OBYT1:    MOV       B,A         ;SAVE IN B
          ANI       0F0H        ;MASK OFF LOW NIBBLE
```

```
         ORA      E           ;MAYBE MODIFY RS
         CALL     DLNOUT      ;SEND IT
         MOV      A,B
         ADD      A
         ADD      A
         ADD      A
         ADD      A           ;LOWER IS MOVED TO UPPER, PADDING 0'S
         ORA      E           ;MAYBE MODIFY RS
         CALL     DLNOUT
         RET

;
; This delays and falls through to OUTNIB
;
DLNOUT:  CALL     DELAY

;
; Send data in A to the LCD.  Assumes bits 0 to 3 have been properly set.
;
OUTNIB:  PUSH     PSW
         ANI      11110111B   ;CLEAR E
         OUT      OPORT       ;SEND NIBBLE
         ORI      1000B       ;SET E BIT
         OUT      OPORT
         ANI      11110111B   ;CLEAR E BIT
         OUT      OPORT
         POP      PSW
         RET

;
; 5ms time delay for 8085 is 24 t states
;
DELAY:   PUSH     PSW         ;approx 5ms for 3.072 MHZ clock
         PUSH     H
         LXI      H,641
DLAY2:   DCX      H           ;6 T STATES
         MOV      A,H         ;4 T STATES
         ORA      L           ;4 T STATES
         JNZ      DLAY2       ;10 T STATES
         POP      H
         POP      PSW
         RET
```

Program Description:

According to the schematic, the output port controls the LCD and the port bits are connected as follows:

```
output port bits:      7   6   5   4   3   2   1   0
LCD header pins:      DB7 DB6 DB5 DB4  E   RS (not used)
```

The routine OUTNIB assumes the upper nibble of A has the value you want to output and bit 2 (RS) is set to 0 for a command or 1 for data. This value is output first with bit 3 (E) low, then high, then low again. The E input when brought high momentarily causes the data input to RS and DB4 through DB7 to be accepted by the LCD controller. DLNOUT works the same except a 5mS delay (provided by DELAY) occurs before executing OUTNIB.

DELAY is called because the method we used to interface to the LCD Module prevents us from reading the LCD module. This in turn prevents us from reading the busy flag which tells us the LCD controller is busy executing a command and cannot receive another yet. DELAY gets us around this problem because it takes longer to execute than any of the LCD controller's instructions insuring that the LCD will not be busy by the time it is finished. In the initialization section some longer delays are needed, so DELAY is called repeatedly.

OUTCMD and OUTDTA use the same core routine but they select RS of 0 and 1 respectively. This core routine takes the byte in A and breaks it into two nibbles and sends them to DLNOUT (high nibble first).

The main routine does the hardware reset for the HD44780, followed by the display mode setup. Then SHWSTR sends the ASCII string pointed to by HL to the display via OUTDTA, and then the MOS subroutine KEYIN is called to get a key from the keypad and the key is translated to ASCII and sent to the display (via OUTDTA) and then it loops back to get another key.

Connect Primer connector CN3 to the LCD according to the schematic and then enter the following program. When you run the program "The Primer._" should be shown on the display and when you press one of keys "0" to "9" they will be shown on the display, with each new character displayed to the right of the previous.

Eventually if you press the keys enough times you will eventually run out of display area. The characters are now being stored in an area that is not being displayed. If you have a 2 line display and you send enough characters, they will start showing up on the second line and after more are sent they will eventually show up on the first line.

ADDRESS	DATA	DESCRIPTION	
FF01	3E	MVI	A,F3
FF02	F3		
FF03	D3	OUT	11
FF04	11		
FF05	CD	CALL	FF8D
FF06	8D		
FF07	FF		
FF08	CD	CALL	FF8D
FF09	8D		
FF0A	FF		
FF0B	3E	MVI	A,30
FF0C	30		
FF0D	CD	CALL	FF7B
FF0E	7B		
FF0F	FF		
FF10	CD	CALL	FF7B
FF11	7B		
FF12	FF		
FF13	CD	CALL	FF7B
FF14	7B		
FF15	FF		
FF16	3E	MVI	A,20
FF17	20		
FF18	CD	CALL	FF7B
FF19	7B		
FF1A	FF		
FF1B	3E	MVI	A,28
FF1C	28		
FF1D	CD	CALL	FF68
FF1E	68		
FF1F	FF		
FF20	3E	MVI	A,08
FF21	08		
FF22	CD	CALL	FF68
FF23	68		
FF24	FF		
FF25	3E	MVI	A,01
FF26	01		
FF27	CD	CALL	FF68
FF28	68		
FF29	FF		
FF2A	3E	MVI	A,0E
FF2B	0E		
FF2C	CD	CALL	FF68
FF2D	68		
FF2E	FF		
FF2F	3E	MVI	A,06
FF30	06		
FF31	CD	CALL	FF68
FF32	68		
FF33	FF		
FF34	21	LXI	H,FF4D
FF35	4D		
FF36	FF		
FF37	CD	CALL	FF59
FF38	59		
FF39	FF		
FF3A	00	NOP	
FF3B	00	NOP	
FF3C	00	NOP	
FF3D	00	NOP	
FF3E	00	NOP	
FF3F	0E	MVI	C,0B
FF40	0B		

ADDRESS	DATA	DESCRIPTION	
FF41	CD	CALL	1000
FF42	00		
FF43	10		
FF44	3E	MVI	A,30
FF45	30		
FF46	85	ADD	L
FF47	CD	CALL	FF63
FF48	63		
FF49	FF		
FF4A	C3	JMP	FF3A
FF4B	3A		
FF4C	FF		
FF4D	54	"T"	
FF4E	68	"h"	
FF4F	65	"e"	
FF50	20	" "	
FF51	50	"P"	
FF52	72	"r"	
FF53	69	"i"	
FF54	6D	"m"	
FF55	65	"e"	

```
FF56        72    "r"                          FF9A        C9    RET
FF57        2E    "."
FF58        00    (end marker)
FF59        7E    MOV    A,M
FF5A        23    INX    H
FF5B        B7    ORA    A
FF5C        C8    RZ
FF5D        CD    CALL   FF63
FF5E        63
FF5F        FF
FF60        C3    JMP    FF59
FF61        59
FF62        FF
FF63        1E    MVI    E,04
FF64        04
FF65        C3    JMP    FF6A
FF66        6A
FF67        FF
FF68        1E    MVI    E,00
FF69        00
FF6A        47    MOV    B,A
FF6B        E6    ANI    F0
FF6C        F0
FF6D        B3    ORA    E
FF6E        CD    CALL   FF7B
FF6F        7B
FF70        FF
FF71        78    MOV    A,B
FF72        87    ADD    A
FF73        87    ADD    A
FF74        87    ADD    A
FF75        87    ADD    A
FF76        B3    ORA    E
FF77        CD    CALL   FF7B
FF78        7B
FF79        FF
FF7A        C9    RET
FF7B        CD    CALL   FF8D
FF7C        8D
FF7D        FF
FF7E        F5    PUSH   PSW
FF7F        E6    ANI    F7
FF80        F7
ADDRESS     DATA  DESCRIPTION
FF81        D3    OUT    11
FF82        11
FF83        F6    ORI    08
FF84        08
FF85        D3    OUT    11
FF86        11
FF87        E6    ANI    F7
FF88        F7
FF89        D3    OUT    11
FF8A        11
FF8B        F1    POP    PSW
FF8C        C9    RET
FF8D        F5    PUSH   PSW
FF8E        E5    PUSH   H
FF8F        21    LXI    H,0281
FF90        81
FF91        02
FF92        2B    DCX    H
FF93        7C    MOV    A,H
FF94        B5    ORA    L
FF95        C2    JNZ    FF92
FF96        92
FF97        FF
FF98        E1    POP    H
FF99        F1    POP    PSW
```

In the next example we will modify the program to use the Set DD RAM Address command which will in effect allow us to control the cursor position. Modify the following addresses and run the program. You will see that each key typed will show up on the screen in the same place even though it is still automatically incrementing the cursor position. This is because the address is set for that cursor position after the cursor has been incremented.

You may want to experiment with different cursor positions. If you have a 2 line display, you can move the cursor to line 2 by sending 10000000b + 40h (C0h) to OUTCMD, where 10000000b is the command for Set DD RAM Address and 40h is the offset for line 2.

ADDRESS	**DATA**	**DESCRIPTION**
FF3A	3E	MVI A,8B
FF3B	8B	
FF3C	CD	CALL FF68
FF3D	68	
FF3E	FF	