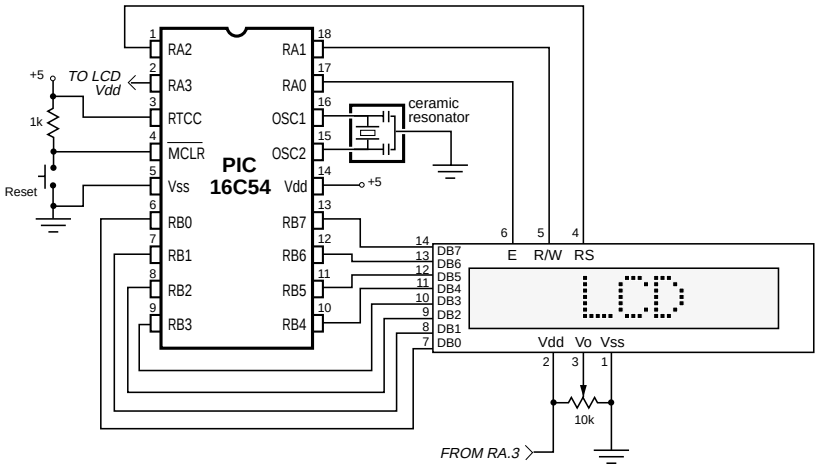
Driving an LCD Display

Introduction. This application note shows how to interface PIC microcontrollers to common Hitachi liquid-crystal display (LCD) modules. The program, in TechTools assembly language, writes text to the display, reads display status, and creates custom character patterns.

Background. LCD modules based on the Hitachi 44780 controller are plentiful and inexpensive, and range in size from 8 to 80 characters. While a complete description of these LCDs' features and operation is beyond the scope of this application note, here are the highlights:



Hitachi LCD modules display the standard ASCII character set, plus selected Japanese, Greek, and math symbols. They operate from a single-ended 5-volt supply, and communicate with a bus or controller through 11 input/output (I/O) lines. The data lines are tri-state; they go into a high-impedance state when the LCD is not enabled.

The three control lines 'control' the LCD. The enable (E) line determines whether the LCD listens to the other control and data lines. When disabled, the LCD ignores all data and control signals. When enabled, the LCD checks the state of the other two control lines and responds accordingly.

The read/write (R/W) line determines whether the LCD reads bits from the data lines, or writes bits to them.

Register-select (RS) determines whether the LCD treats data as instructions or characters. Here is the truth table for the control lines:

E	0	LCD disabled.
	1	LCD enabled.
R/W	0	Write to LCD.
	1	Read from LCD.
RS	0	Instructions.
	1	Characters/bytes.

Writing to the LCD requires the basic steps listed below. (Reading from the LCD follows the same sequence, but the R/W bit must be set.)

- Clear the R/W bit.
- Set or clear the RS bit as appropriate.
- Set the E bit (E=1).
- Clear the E bit (E=0).

When power is applied to the LCD, it resets itself and waits for instructions. Typically these instructions turn on the display, turn on the cursor, and set the display to print from left to right.

Once the LCD is initialized, it can receive data or instructions. If it receives a character, it prints it on the screen and moves the cursor one character to the right. The cursor marks the next location at which a character will be printed. The LCD's internal processing is similar. A memory pointer determines where the next byte will be stored. When a new byte arrives, the pointer advances. To write to sequential locations, establish the starting address and then write one byte after another.

Characters are stored in data display (DD) RAM. Regardless of the number of characters visible on the display, the LCD has 80 bytes of DD RAM. Characters in off-screen RAM can be made visible by scrolling the display.

The LCD also has 64 bytes of character-generator (CG) RAM. Data in

CG RAM determines the bit maps of the characters corresponding to codes 0 through 7 (in normal ASCII, these are control codes). To download bit maps to the LCD, first set the CG RAM address to the desired starting point (usually 0), and then write the bytes to the LCD. Because the pointer increments with each write, you don't need to keep specifying addresses. Figure 2 is an example pattern definition. The program listing shows how to define custom characters.

Address in Character Generator RAM	Bit Map	Data	ÒretwĆ Data
0000		01111	15
0001		10001	17
0010		10001	17
0011		01111	15
0100		00001	1
0101		00001	1
0110		00001	1
0111		00000	0

Figure 2. Programming a custom character pattern.

Before you can write to DD RAM after defining custom characters, the program must set a DD RAM address. The LCD reads and writes whichever RAM bank (DD or CG) was last specified in a set-address instruction. Once you have set an address in DD RAM, the next data write will display a character at the corresponding location on the screen.

Until now, we have talked about reading and writing to the LCD as though it were regular memory. It's not. The LCD's controller takes 40 to 120 microseconds (µs) to complete a read or write. Other operations can take as long as 5 milliseconds. To avoid making the PIC wait a worst-case delay between operations, the LCD has a 1µs instruction that reads the address counter and a busy flag. When the busy flag is set (1), the LCD cannot handle a read or write. The program listing includes a subroutine (blip_E) that makes sure the busy flag is cleared (0) before talking to the LCD.

The address returned along with the busy flag is either the DD or CG RAM pointer, depending on which address was last set.

Figure 3 is a list of LCD instructions for reading and writing memory. Some other useful instructions appear as constants in the beginning of the program listing.

How it works. The circuit in figure 1 interfaces a PIC to an LCD module. When the power is turned on, or the circuit is reset, the PIC initializes the LCD, downloads four custom characters, and prints “TechTools” forward, "Tools" backward and ends with a "smile".

The potentiomenter connected to the LCD’s Vo pin controls contrast. If the display is hard to read, appears blank, or is filled with black pixels, adjust this control.

Power to the LCD is controlled by a PIC I/O bit. In this way, the PIC ensures that the 5-volt supply is up and stable before switching on the LCD. If the circuit used the PIC’s sleep mode, it could shut down the LCD to save approximately 1.5 mA. If you use this feature, make sure that the

Set DD RAM address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	1	A	A	A	A	A	A	A

Set CG RAM address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	0	0	1	A	A	A	A	A	A

Read busy flag and address

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
0	1	Bsy	A	A	A	A	A	A	A

Write data to RAM (CG or DD, most recently set)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	0	D	D	D	D	D	D	D	D

Read data from RAM (CG or DD, most recently set)

RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0
1	1	D	D	D	D	D	D	D	D

Figure 3. Commonly used memory operations.
(A = address; D = data; Bsy = busy)

data and control lines are cleared to 0's or set to input before putting the PIC to sleep. Otherwise, leakage through the LCD's protection diodes might continue to power it.

Program listing. This program may be downloaded from our Internet ftp site at ftp.tech-tools.com. The ftp site may be accessed directly or through our web site at http://www.tech-tools.com.

```
; PROGRAM: Drive Liquid Crystal Display (LCD_DRVR.SRC)
; This program initializes a Hitachi LCD module, defines a set of four custom
; characters, and displays the message 'TechTools' forward and backward (in
; custom, mirror-reading letters as many as would fit). It includes a
; subroutine, blip_E, that handles all the required handshaking to send
; data or instructions to LCD.

                device  pic16c54,xt_osc,wdt_off,protect_off
                reset   start

LCD_pwr         =       ra.3    ;+5 to LCD module
RS              =       ra.2    ;0 = write, 1 = read
RW              =       ra.1    ;0 = instruction, 1 = data
E               =       ra.0    ;0 = disable, 1 = enable
data            =       rb      ;Data to LCD
count           =       16      ;Number of characters in demo string.
char_cnt        =       32      ;Number of bytes in custom character
                                ;definition

; Declare constants for common LCD instructions. To perform the functions
; listed below, clear bit RS and send #constant to the display. Remember
; to set RS again before sending characters to the LCD.

clear           =       1       ;Clears the display (fills with blanks)
home            =       2       ;Returns display to the home position.
shift_l         =       24      ;Shifts display to the left.
shift_r         =       28      ;Shifts display to the right.
crsr_l          =       16      ;Moves cursor to the left.
crsr_r          =       20      ;Moves cursor to the right.
no_crsr =       8               ;Turns off the cursor.
blink_c         =       11      ;Blinks whole character to indicate
                                ;cursor position.

                org     8       ;Start of available RAM
```

```
temp            ds      1       ;Temporary counter.
temp2           ds      1       ;Pass data or instructions to blip_E.
counter         ds      1       ;Index variable.

        org 0           ;Start of code space
start           mov     ra, #0  ;Initialize ports, power LCD and wait
                                ;for it to reset.
                mov     rb, #0
                mov     !ra,#0h         ;Set control lines to output
                mov     !rb,#0h         ;Set data lines to output
                setb    LCD_pwr
                call    wait
                mov     temp2,#00110000b ;Initialize LCD: set 8-bit, 1-line
                                        ;operation
                call    blip_E
                mov     temp2,#00001110b
                call    blip_E
                mov     temp2,#00000110b
                call    blip_E

                mov     temp2, #01000000b ;Write to CG RAM:
                                          ;start at address 0.
                call    blip_E
                setb    RS              ;Set RS to send data.
                mov     counter, #0

:stuff          mov     w, counter
                call    my_chars        ;Get next byte.
                mov     temp2, w        ;Write byte to CG RAM.
                call    blip_E
                inc     counter
                cjb     counter, #char_cnt, :stuff

                clrb    RS              ;Clear RS to send instruction.

                mov     temp2, #10000000b
                                        ;Address 0 of DD RAM.
                call    blip_E
                setb    RS

send_msg        mov     counter, #0     ;Send the message string to the LCD.
:loop           mov     w,counter
                call    msg
                mov     temp2,w
                call    blip_E
                inc     counter
                cjb     counter,#count,:loop
```

```
:loop2      jmp     :loop2    ;Endless loop. Reset PIC to run
                              ;program again.

            ; Write data or instructions (in variable temp2) to the LCD.
blip_E      movb    pa2, RS   ;Store current state of RS in unused bit.
            clrb    RS                ;Clear RS to send instruction.
            setb    RW                ;Set RW to read.
            mov     !rb, #255         ;Port rb: all inputs.

:loop       setb    E                 ;Enable the LCD.
            nop
            mov     temp, data        ;Put LCD data (rb) into temp.
            clrb    E                 ;Disable LCD.
            snb     temp.7            ;Is the LCD busy?
            jmp     :loop             ;Yes, try again later.
            movb    RS, pa2           ;No, send the data or instruction.
            clrb    RW
            mov     !rb, #0
            mov     data, temp2
            setb    E
            nop
            clrb    E
            ret

            ;Create a delay for LCD power-on reset.
wait        mov     temp2,#200
:loop       djnz    temp,:loop
            djnz    temp2,:loop
            ret

            ;Table of ASCII and custom characters to display.
msg         jmp     pc+w
            retw'T','e','c','h','T','o','o','l','s',0,1,2,2,'T','  ',3

            ;Table of data for custom characters.
my_charsjmp pc+w
            retw  0,0,14,1,14,16,15,0     ; Backwards s
            retw  6,4,4,4,4,4,14,0        ; Backwards l
            retw  0,0,14,17,17,17,14,0    ; Graphic small o
            retw  0,10,0,17,14,0,0,0      ; Graphic smile
```