Interfacing an SX Microcontroller to a Hitachi HD44780 LCD Display

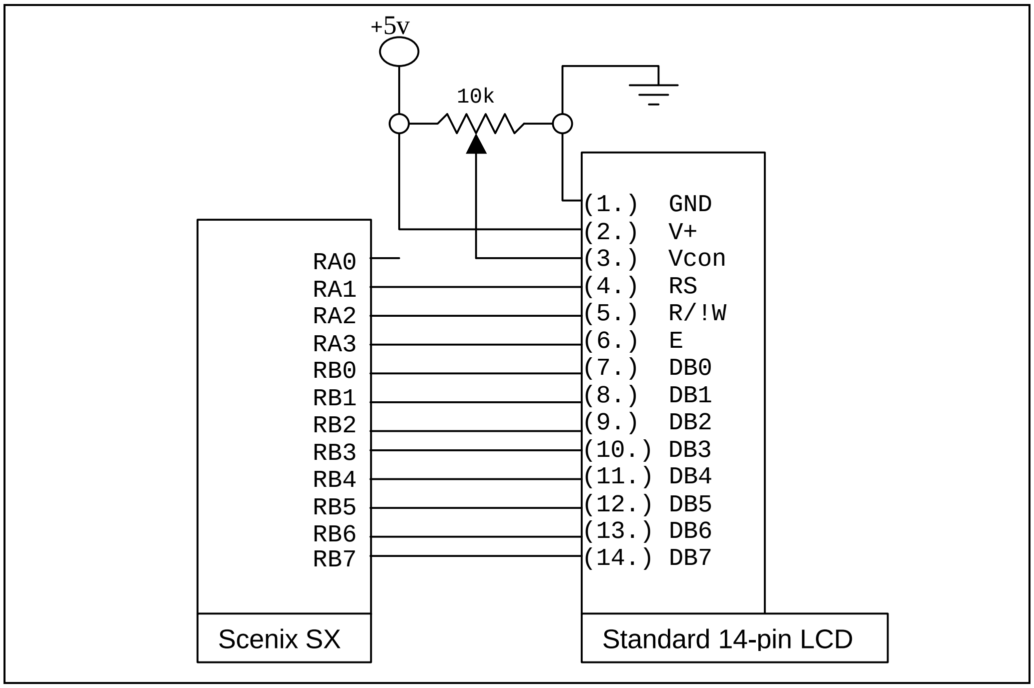
Application Note: Simple Interface between Scenix SX and Hitachi-HD44780 Driven Display.

Introduction

This application note describes two simple way to interface a Scenix SX microcontroller and a Hitachi HD44780-Driven Display, through a 4-bit and an 8-bit interface. The Hitachi HD44780 LCD driver is one of the most common LCD controllers, and is very easy to find at surplus electronics stores.

The HD44780 controller IC

The HD44780 IC is a self contained LCD driver, designed to interface with microcontrollers/microprocessors. Its interface is either 4 or 8 bits. The IC has built-in Display Data RAM (DDRAM) to store the displayed characters, as well as Character Generator Ram (CGRAM), which can hold custom, user-designed characters. This application note deals only with writing characters to the DDRAM, the most common usage.



Connecting the SX to the LCD's *standard* 14-pin connector (Example only. This diagram can be used with both the 8-bit and 4-bit interface techniques, although the DB0-DB3 connections are not necessary for the 4-bit interface. The example programs *`lcd8xmpl.src`* and *`lcd4xmpl.src`* use this layout and are available from www.scenix.com.)

Most LCD's using the HD44780 driver chip use this industry-standard pin-out. Check datasheet to be sure:

PIN	NAME	OPERATION
1	Vss	(-) Ground
2	Vcc	(+) Power
3	Vee	Contrast Adjust. Connect to Potentiometer
4	RS	Data/!Instruction… 0 = Instruction input, 1 = Data input
5	R/!W	Read/!Write… 0 = Write, 1 = Read
6	E	Enable signal. Active High (Read). Negative edge triggers input latch (Write).
7	DB0	Data Bus Line 0 (LSB)
8	DB1	Data Bus Line 1
9	DB2	Data Bus Line 2
10	DB3	Data Bus Line 3
11	DB4	Data Bus Line 4
12	DB5	Data Bus Line 5
13	DB6	Data Bus Line 6
14	DB7	Data Bus Line 7 (MSB)

HD44780 Instruction Set

From *Hitachi Liquid Crystal Display Module* databook.

Instruction	RS	R/!W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	Description	EXE TIME
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears display memory and returns the cursor to the home position. (Address 0)	82us - 1.64ms
Return Home	0	0	0	0	0	0	0	0	1	*	Returns the cursor to the home position (Address 0) and shifts the display back to its original position. Does not change DDRAM contents.	40us - 1.6ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	S	Sets direction that the cursor moves and whether or not to shift the display. Write and read.	40us - 1.64ms
Display ON/OFF	0	0	0	0	0	0	0	D	C	B	D = Display ON/OFF C = Cursor ON/OFF B = Blinking Cursor	40us
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	*	*	Moves the cursor and shifts the display without changing DD RAM contents.	40us
Function Set	0	0	0	0	1	DL	N	F	*	*	DL = Interface Data Length N = Number of Display Lines F = Character Font	40us
Set CG RAM Address	0	0	0	1	CG RAM Address						Sets the CG RAM address. CG RAM data is sent/received after this command.	40us
Set DD RAM Address	0	0	1	DD RAM Address							Sets the DD RAM Address. DD RAM is sent/received after this command.	40us
Read Busy Flag and Address Counter Contents	0	1	BF	Address Counter Contents							Reads the Busy Flag (BF), indicating an internal operation is in progress, as well as the contents of the address counter.	1us
Write Data to CG/DD RAM	1	0	Data to Write								Writes data into DDRAM or CGRAM, depending on current Address.	40us
Read Data from CG/DD RAM	1	1	Data to Read								Reads data from DDRAM or CGRAM, depending on current Address.	40us

Setting Bit	Definition	Setting Bit	Definition

I/D = 1
I/D = 0
	Increment
	Decrement
		BF = 1
		BF = 0
			Internal Operation in progress.
			Instructions can be accepted.

S = 1
S = 0
	Display Shift
	No Display Shift
		R/L = 1
		R/L = 0
			Right Shift
			Left Shift

D = 1
D = 0
	Display ON
	Display OFF
		DL = 1
		DL = 0
			8 - Bit Interface
			4 - Bit Interface

C = 1
C = 0
	Cursor ON
	Cursor OFF
		N = 1
		N = 0
			2 Line Display
			1 Line Display

B = 1
B = 0
	Blink ON
	Blink OFF
		F = 1
		F = 0
			5*10 dot matrix
			5*7 dot matrix

S/C = 1 S/C = 0	Display Shift Cursor Movement		

Initializing the LCD

On power-up, the LCD needs several milliseconds to initialize. In the example code, about 5ms is used:

```
lcd_init

            mov        W,#0                                ; Delays for 5.1ms at 50MIPS
            call       delay
```

After this initial delay, the program initializes the RA and RB ports to outputs, and waits until the LCD has finished intializing. When the intialization process has completed, the program begins sending commands to the LCD, using the lcd_write_command subroutine.

The first command, except for busy flag/address read, to be written to the LCD after power-up should always be "Function Set," which chooses the interface data length, the number of data lines, and character font. If this command is not issued first, no function instruction except changing the interface data length can be executed.

After sending each command, the example programs use the lcd_wait_busy subroutine to wait for the LCD to finish processing the last command.

```
; Set up the LCD I/O first.  RA0-RA3 are all outputs, as are RB0-RB7

            mov        W, #00h
            mov        lcd_control, W                      ; Set up the latches for when this register is switched to output.
            mov        !lcd_control, W                     ; Switch RA to all outputs, with a 0000 appearing on the pins (Enable is low)
            mov        !lcd_data, W                        ; Switch RB to all outputs.  (for initialization routine)

; First, set the data length, number of display lines, and character font.
;------------------------------------------------------------------------------------------------------
;           RS-RA2 R/!W-RA3 DB7-RB7DB6-RB6     DB5-RB5     DB4-RB4    DB3-RB3     DB2-RB2     DB1-RB1    DB0-RB0     Execution Time
;           0          0           0           0           1          DL          N           F          *           *           40us
;------------------------------------------------------------------------------------------------------
; DL--Interface Data Length                    0 = 4-bit interface                1 = 8-bit interface
; N --Number of Display Lines                  0 = 1 line                         1 = 2 lines
; F --Character Font                           0 = 5*7 dots                                   1 = 5*10 dots

            call       lcd_wait_busy           ; wait for the LCD to finish initializing
            mov        W, #00111000b
            call       lcd_write_command       ; set for for 8 bits, 2 lines, and 5*7 dots
            call       lcd_wait_busy           ; Wait until the LCD is finished processing.


; Next, turn the display on, turn the cursor on, and turn cursor blink on (so we know LCD is alive)
;------------------------------------------------------------------------------------------------------
;           RS-RA2 R/!W-RA3 DB7-RB7DB6-RB6     DB5-RB5     DB4-RB4    DB3-RB3     DB2-RB2     DB1-RB1    DB0-RB0     Execution Time
;           0          0           0           0           0          0           1           D          C           B           40us
;------------------------------------------------------------------------------------------------------
; D --Display ON/OFF control                   0 = Display OFF                                1 = Display ON
; C --Cursor ON/OFF control                    0 = Cursor OFF                                 1 = Cursor ON
; B --Blink ON/OFF control                     0 = Blink OFF                                  1 = Blink ON

            clr        W
            call       lcd_write_command
            call       lcd_wait_busy                       ; Display off

            mov        W, #00001111b
            call       lcd_write_command       ; turn display on, cursor on, and blink on..
            call       lcd_wait_busy                       ; Wait until the LCD is finished processing.

; Next, set display so that the cursor moves as characters are entered.
;------------------------------------------------------------------------------------------------------
;           RS-RA2 R/!W-RA3 DB7-RB7DB6-RB6     DB5-RB5     DB4-RB4    DB3-RB3     DB2-RB2     DB1-RB1    DB0-RB0     Execution Time
;           0          0           0           0           0          1           S/C         R/L        *           *           40us
;------------------------------------------------------------------------------------------------------
; S/C--Cursor move/Display Shift   0 = Cursor Move                                1 = Shift Display
; R/L--Shift Direction                         0 = Shift left                                 1 = Shift right

            mov        W, #00010000b
            call       lcd_write_command       ; set for cursor move and display shift.
            call       lcd_wait_busy                       ; Wait until the LCD is finished processing.

; Next, set entry mode (cursor move direction, shift or no shift).
;------------------------------------------------------------------------------------------------------
;           RS-RA2 R/!W-RA3 DB7-RB7DB6-RB6     DB5-RB5     DB4-RB4    DB3-RB3     DB2-RB2     DB1-RB1    DB0-RB0     Execution Time
;           0          0           0           0           0          0           0           1          I/D         S           40us ~ 1.64ms
;------------------------------------------------------------------------------------------------------
; I/D--Increment/Decrement address 0 = Decrement Cursor Address       1 = Increment Cursor Address
; S  --Display shift                           0 = No shift                                   1 = Shift

            mov        W, #00000110b
            call       lcd_write_command       ; set for incrementing address and no shift..
            call       lcd_wait_busy                       ; Wait until the LCD is finished processing.

            ret        ; Return from lcd_init subroutine.

;*************************************************************************
; End of lcd_init subroutine.
;*************************************************************************
```

Writing Commands and Data

The lcd_write_command and lcd_write_data subroutines use the same core code. The only difference is that the lcd_write_command subroutine clears the LCD's RS pin, whereas the lcd_write_data subroutine sets it.

In 8-bit data mode (lcd8xmpl.src), the write and wait_busy subroutines differ from 4-bit data mode (lcd4xmpl.src).

Writing to the LCD in 8-bit data mode requires the SX to

- set up RS
- set R/!W to LO
- put data on DB7-DB0
- pulse the Enable pin

Writing to the LCD in 4-bit data mode requires the SX to

- set up RS
- set R/W to LO
- put the most significant four bits of data on DB7-DB4
- pulse the Enable pin
- put the least significant four bits of data on DB7-DB4
- pulse the Enable pin

```
;-----------------------------------------------------------------------------------------------------
lcd_write_command
;-----------------------------------------------------------------------------------------------------
; This function writes the command in W to the LCD display, using the 8-bit interface.  The procedure is:
; 1.  Clear RS
; 2.  Set up R/!W
; 3.  Write the data to the port
;-----------------------------------------------------------------------------------------------------
         clrb     lcd_RS            ; Drive RS low so LCD knows to write COMMAND.
         jmp      lcd_write         ; goto WRITE code

lcd_write_data
;-----------------------------------------------------------------------------------------------------
; This function writes the data in W to the LCD display, using the 8-bit interface.
; 1.  Set RS
; 2.  Set up R/!W
; 3.  Write the data to the port
;-----------------------------------------------------------------------------------------------------

         setb     lcd_RS            ; Drive RS high so LCD knows to write DATA.

lcd_write
         mov      lcd_data,W        ; Write the data in W to the port latches.
         mov      W,#000h           ; Write zeroes to the control register to switch the data pins to outputs.
         mov      !lcd_data,W
         clrb     lcd_RW            ; Drive R/!W low so LCD knows to WRITE.
         call     nopdel
         call     nopdel
         setb     lcd_E             ; Pulse LCD's enable pin.
         call     nopdel
         call     nopdel
         clrb     lcd_E             ; Force LCD to latch the data present on the data bus.
         call     nopdel
         call     nopdel
         ret
;-----------------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------------
```

Waiting for the Busy Flag

The busy flag indicates that the LCD is busy completing a task and is not ready for new data or commands. To check the busy flag, the SX needs to clear RS, to set R/!W, and to set Enable. The LCD will return the status of the busy flag.

Checking the busy flag in 8-bit data mode requires the SX to:

- Change the port pins to inputs.
- Set RS low.
- Set R/!W high.
- Pulse Enable.
- Read the data at the input pins (the MSB is 1 if the LCD is busy, otherwise it is zero.)

Checking the busy flag in 4-bit data mode requires the SX to:

- Change the port pins to inputs.
- Set RS low.
- Set R/!W high.
- Pulse Enable.
- Read the data at the input pins (Only DB7-DB4 will return data. DB7 is HI if the LCD is busy, otherwise it is LO)
- Pulse Enable again. This causes the LCD to read out the lower 4 bits of the current DDRAM address on DB7-DB4.

Example of 8-bit Busy Flag check.

```
lcd_wait_busy
; waits until the LCD is ready to accept a command.
;------------------------------------------------------------------------------------------------------
;        RS-RA2 R/!W-RA3 DB7-RB7   DB6-RB6  DB5-RB5  DB4-RB4  DB3-RB3  DB2-RB2  DB1-RB1  DB0-RB0  Exe Time
;        0        1       BF       * ------------------DDRAM Address-------------- *    1us
;------------------------------------------------------------------------------------------------------

         mov      W, #0FFh ; write ones to the control register to switch the data pins to inputs.
         mov      !lcd_data,W
         clrb     lcd_RS           ; clear RS for instruction
         setb     lcd_RW           ; set for READ.
         call     nopdel
         call     nopdel
         setb     lcd_E            ; set enable high to read busy flag
         call     nopdel
         call     nopdel           ; wait for the LCD to tx data.
         mov      W,lcd_data
         clrb     lcd_E            ; clear LCD enable
         call     nopdel
         call     nopdel
         and      W, #080h         ; test W for zero (Z is cleared if LCD is busy)
         sb       Z
         jmp      lcd_wait_busy
         setb     lcd_RW
         mov      W,#00h
         mov      !lcd_data,W      ; Switch the data pins back to outputs
         call     nopdel
         call     nopdel
         call     nopdel
         ret                       ; return from subroutine
```

```
;*************************************************************************
;
; Author:  Chris Fogelklou for Scenix Semiconductor, Inc.(chris.fogel@scenix.com)
; Written:  Thursday, August 20, 1998.
; Modified: Wednesday, August 26, 1998.
;
; This is simple code to demonstrate how to use an SX chip to interface
; with an LCD display.  It initializes the display and infinitely loops,
; printing "Hi. " to the display.  This code will work with any type of
; HITACHI HD44780 driven display (1*16, 2*16, 1*20, etc...).  It is not a
; virtual peripheral, as it does not efficiently use the processor.
; (It contains a wait loop, as well as several delays).  This example
; code is simply a good program to build upon.  There is a virtual
; peripheral for LCD under development, which will use the MCU
; efficiently.  (Check www.scenix.com for updates.)
;
;
; lcd_init
; Because the LCD should only need to be initialized once, the LCD_Init
; routine does not return until it is fully completed.  Comments in
; LCD_Init suggest changes for any number of different settings.
; (eg. more/fewer display lines, cursor direction, display shifting...)
; CALLS:
;  -lcd_write_command
;  -lcd_write_data
;  -lcd_wait_busy
;  -delay
;
;
; lcd_write_command
; This subroutine is called to write a command to the LCD, such as
; 'clear screen and return home'.  The command to be written is passed in
; inside the W register.
; CALLS:
;  -nopdel
;  -delay
;
;
; lcd_write_data
; This subroutine is called to write data to the LCD, such as a character
; to be displayed.  Like lcd_write_command, lcd_write_data accepts the
; data in the W register.
; CALLS:
;  -nopdel
;  -delay
;
;
; lcd_wait_busy
; This subroutine does not return until the LCD is ready to accept more
; data/commands.
; CALLS:
;  -nopdel
;  -delay
;
;
; nopdel
; A simple subroutine containing 8 nops, returning after the nops.
;
;
; delay
; This subroutine delays for (w-1)*20us at 50MIPS, (w-1)*1ms at 1MIPS
;
;
; REGISTER USAGE
; The only registers used in this program are
; dlycnt1
; dlycnt2
; in the "delay_regs" bank
;
;*************************************************************************
;*************************************************************************
; Assembler Directives...
;*************************************************************************
                    device    pins28,pages1,banks8,oschs    ; 28 pin package,
                                                                      ; 1 page program,
                                                                      ; 8 banks RAM,
                                                                      ; HS oscillator.
                    device    stackx,optionx,turbo                    ; stack extend,
                                                                      ; option extend, turbo.
                    id        'LCD VP'
                    reset     reset_entry                   ; Jump to reset_entry on reset.
                    FREQ      50000000                      ; 50MHz target frequency.

;*************************************************************************
; Pin Definitions
;*************************************************************************
```

```
lcd_control         =         ra

lcd_RS              =         ra.1                ; 0 = instruction, 1 = data
lcd_RW              =         ra.2                ; 0 = write, 1 = read
lcd_E               =         ra.3                ; 1,1-->0 is the LCD enable

lcd_data  =         rb

lcd_DB0             =         rb.0                ; DB0 = Data bus line 0 (LSB)
lcd_DB1             =         rb.1
lcd_DB2             =         rb.2
lcd_DB3             =         rb.3
lcd_DB4             =         rb.4
lcd_DB5             =         rb.5
lcd_DB6             =         rb.6
lcd_DB7             =         rb.7                ; DB7 = Data bus line 7 (MSB)

;*************************************************************************
; Variables
;*************************************************************************
                    org       8

                    org       10h

                    org       30h                 ;LCD Virtual Peripheral variables

delay_regs          =         $

dlycnt1             ds        1
dlycnt2             ds        1

                    org       0

;*************************************************************************
; Interrupt routine - virtual peripherals
;*************************************************************************

interrupt
          reti
;*************************************************************************




;*************************************************************************
; LCD initialization code.
; This code should be called at the beginning of the program to
; initialize the LCD display.  It only needs to be called once.
;*************************************************************************
lcd_init

          mov       W,#0                          ; Delays for 5.1ms at 50MIPS
          call      delay
          mov       W,#0                          ; Delays for 5.1ms at 50MIPS
          call      delay
          mov       W,#0                          ; Delays for 5.1ms at 50MIPS
          call      delay

; Set up the LCD I/O first.  RA0-RA3 are all outputs, as are RB0-RB7

          mov       W, #00h
          mov       lcd_control, W                ; Set up the latches for when this register is switched to output.
          mov       !lcd_control, W               ; Switch RA to all outputs, with a 0000 appearing on the pins (Enable is
low)
          mov       !lcd_data, W                  ; Switch RB to all outputs.  (for initialization routine)

; First, set the data length, number of display lines, and character font.
;----------------------------------------------------------------------------------------------------
;         RS-RA2 R/!W-RA3 DB7-RB7       DB6-RB6   DB5-RB5   DB4-RB4   DB3-RB3   DB2-RB2   DB1-RB1   DB0-RB0   Execution Time
;         0         0         0         0         1         DL        N         F         *         *         40us
;----------------------------------------------------------------------------------------------------
; DL--Interface Data Length             0 = 4-bit interface           1 = 8-bit interface
; N --Number of Display Lines           0 = 1 line                              1 = 2 lines
; F --Character Font                    0 = 5*7 dots                            1 = 5*10 dots

          mov       W, #00111000b
          call      lcd_write_command   ; set for for 8 bits, 2 lines, and 5*7 dots
          call      lcd_wait_busy                 ; Wait until the LCD is finished processing.


; Next, turn the display on, turn the cursor on, and turn cursor blink on (so we know LCD is alive)
;----------------------------------------------------------------------------------------------------
;         RS-RA2 R/!W-RA3 DB7-RB7       DB6-RB6   DB5-RB5   DB4-RB4   DB3-RB3   DB2-RB2   DB1-RB1   DB0-RB0   Execution Time
;         0         0         0         0         0         0         1         D         C         B         40us
;----------------------------------------------------------------------------------------------------
```

```
; D --Display ON/OFF control            0 = Display OFF                         1 = Display ON
; C --Cursor ON/OFF control             0 = Cursor OFF                          1 = Cursor ON
; B --Blink ON/OFF control              0 = Blink OFF                           1 = Blink ON

          clr       W
          call      lcd_write_command
          call      lcd_wait_busy                 ; Display off

          mov       W, #00001111b
          call      lcd_write_command   ; turn display on, cursor on, and blink on..
          call      lcd_wait_busy       ; Wait until the LCD is finished processing.

; Next, set display so that the cursor moves as characters are entered.
;-----------------------------------------------------------------------------------------------------
;         RS-RA2 R/!W-RA3 DB7-RB7       DB6-RB6   DB5-RB5   DB4-RB4   DB3-RB3   DB2-RB2   DB1-RB1   DB0-RB0   Execution Time
;         0         0         0         0         0         1         S/C       R/L       *         *         40us
;-----------------------------------------------------------------------------------------------------
; S/C--Cursor move/Display Shift        0 = Cursor Move                         1 = Shift Display
; R/L--Shift Direction                            0 = Shift left                          1 = Shift right

          mov       W, #00010000b
          call      lcd_write_command   ; set for cursor move and display shift.
          call      lcd_wait_busy                 ; Wait until the LCD is finished processing.

; Next, set entry mode (cursor move direction, shift or no shift).
;-----------------------------------------------------------------------------------------------------
;         RS-RA2 R/!W-RA3 DB7-RB7       DB6-RB6   DB5-RB5   DB4-RB4   DB3-RB3   DB2-RB2   DB1-RB1   DB0-RB0   Execution Time
;         0         0         0         0         0         0         0         1         I/D       S         40us ~ 1.64ms
;-----------------------------------------------------------------------------------------------------
; I/D--Increment/Decrement address      0 = Decrement Cursor Address  1 = Increment Cursor Address
; S  --Display shift                              0 = No shift                            1 = Shift

          mov       W, #00000110b
          call      lcd_write_command   ; set for incrementing address and no shift..
          call      lcd_wait_busy                 ; Wait until the LCD is finished processing.

          ret       ; Return from lcd_init subroutine.

;*************************************************************************
; End of lcd_init subroutine.
;*************************************************************************



;-----------------------------------------------------------------------------------------------------
lcd_write_command
;-----------------------------------------------------------------------------------------------------
; This function writes the command in W to the LCD display, using the 8-bit interface.  The procedure is:
; 1.  Clear RS
; 2.  Set up R/!W
; 3.  Write the data to the port
;-----------------------------------------------------------------------------------------------------
          clrb      lcd_RS              ; Drive RS low so LCD knows to write COMMAND.
          jmp       lcd_write ; goto WRITE code

lcd_write_data
;-----------------------------------------------------------------------------------------------------
; This function writes the data in W to the LCD display, using the 8-bit interface.
; 1.  Set RS
; 2.  Set up R/!W
; 3.  Write the data to the port
;-----------------------------------------------------------------------------------------------------

          setb      lcd_RS              ; Drive RS high so LCD knows to write DATA.

lcd_write
          mov       lcd_data,W          ; Write the data in W to the port latches.
          mov       W,#000h             ; Write zeroes to the control register to switch the data pins to outputs.
          mov       !lcd_data,W
          clrb      lcd_RW              ; Drive R/!W low so LCD knows to WRITE.
          call      nopdel
          call      nopdel
          setb      lcd_E               ; Pulse LCD's enable pin.
          call      nopdel
          call      nopdel
          clrb      lcd_E               ; Force LCD to latch the data present on the data bus.
          call      nopdel
          call      nopdel
          ret
;-----------------------------------------------------------------------------------------------------
;-----------------------------------------------------------------------------------------------------

lcd_wait_busy
; waits until the LCD is ready to accept a command.
;-----------------------------------------------------------------------------------------------------
;         RS-RA2 R/!W-RA3 DB7-RB7       DB6-RB6   DB5-RB5   DB4-RB4   DB3-RB3   DB2-RB2   DB1-RB1   DB0-RB0   Execution Time
;         0         1         BF        * ------------------DDRAM Address-------------- *                     1us
;----------------------------------------------------------------------------------------------------

          mov       W, #0FFh  ; write ones to the control register to switch the data pins to inputs.
          mov       !lcd_data,W
          clrb      lcd_RS              ; clear RS for instruction
          setb      lcd_RW              ; set for READ.
          call      nopdel
          call      nopdel
          setb      lcd_E               ; set enable high to read busy flag
          call      nopdel
          call      nopdel              ; wait for the LCD to tx data.
          mov       W,lcd_data
          clrb      lcd_E               ; clear LCD enable
          call      nopdel
          call      nopdel
          and       W, #080h  ; test W for zero (Z is cleared if LCD is busy)
          sb        Z
          jmp       lcd_wait_busy
          setb      lcd_RW
          mov       W,#00h
          mov       !lcd_data,W         ; Switch the data pins back to outputs
          call      nopdel
          call      nopdel
          call      nopdel
          ret                           ; return from subroutine

nopdel    ;         returns to main program in 11 cycles (11us@1MIPS) from call
;----------------------------------------------------------------------------------------------------
          nop
          nop
          nop
          nop
          nop
          nop
          nop
          nop
          ret
;----------------------------------------------------------------------------------------------------

delay; (delays for [((w-1) * 1ms )] at 1MIPS, or [((w-1) * 20us)] at 50MIPS ... 0<=W<=255)
;*****************************************************************************************************
; This function delays for ((W-1)*20us), plus/minus a few ns
;*****************************************************************************************************
          bank      delay_regs
          mov       dlycnt1,W
;
delay1
          decsz     dlycnt1;
          jmp       loop1;
          ret       ;

          loop1
          mov       w,#166;
          mov       dlycnt2,W;

loop;
          nop
          nop
          nop
          decsz     dlycnt2;
          jmp       loop;
          jmp       delay1;
;****************************************************************************************************

;         Main Code
reset_entry
main2
          call      lcd_init
          mov       W, #001h            ; Clear the screen
          call      lcd_write_command
          call      lcd_wait_busy
Hiloop
          mov       W, #'H'                       ; Write "H"
          call      lcd_write_data
          call      lcd_wait_busy
          mov       W, #'i'                       ; Write "i"
          call      lcd_write_data
          call      lcd_wait_busy
          mov       W, #'.'                       ; Write "."
          call      lcd_write_data
          call      lcd_wait_busy
          mov       W, #' '                       ; Write " "
          call      lcd_write_data
          call      lcd_wait_busy
          mov       W,#255
          call      delay               ; Delay 5.1ms @ 50MIPS
          mov       W,#255
          call      delay               ; Delay 5.1ms @ 50MIPS
          mov       W,#255
          call      delay               ; Delay 5.1ms @ 50MIPS
          mov       W,#255
          call      delay               ; Delay 5.1ms @ 50MIPS

          jmp       Hiloop
```