Display()

The **Liquid Crystal Library** allows you to control LCD displays that are compatible with the Hitachi HD44780 driver. There are many of them out there, and you can usually tell them by the 16-pin interface.

This example sketch shows how to use the display() and noDisplay() methods to turn on and off the display. The text to be displayed will still be preserved when you use noDisplay() so it's a quick way to blank the display without losing everything on it.

Hardware Required

- Arduino or Genuino Board
- LCD Screen (compatible with Hitachi HD44780 driver)
- pin headers to solder to the LCD display pins
- 10k ohm potentiometer
- 220 ohm resistor
- hook-up wires
- breadboard

Circuit

Before wiring the LCD screen to your Arduino or Genuino board we suggest to solder a pin header strip to the 14 (or 16) pin count connector of the LCD screen, as you can see in the image above.
To wire your LCD screen to your board, connect the following pins:

- LCD RS pin to digital pin 12
- LCD Enable pin to digital pin 11
- LCD D4 pin to digital pin 5
- LCD D5 pin to digital pin 4
- LCD D6 pin to digital pin 3
- LCD D7 pin to digital pin 2

Additionally, wire a 10k pot to +5V and GND, with it's wiper (output) to LCD screens VO pin (pin3). A 220 ohm resistor is used to power the backlight of the display, usually on pin 15 and 16 of the LCD connector

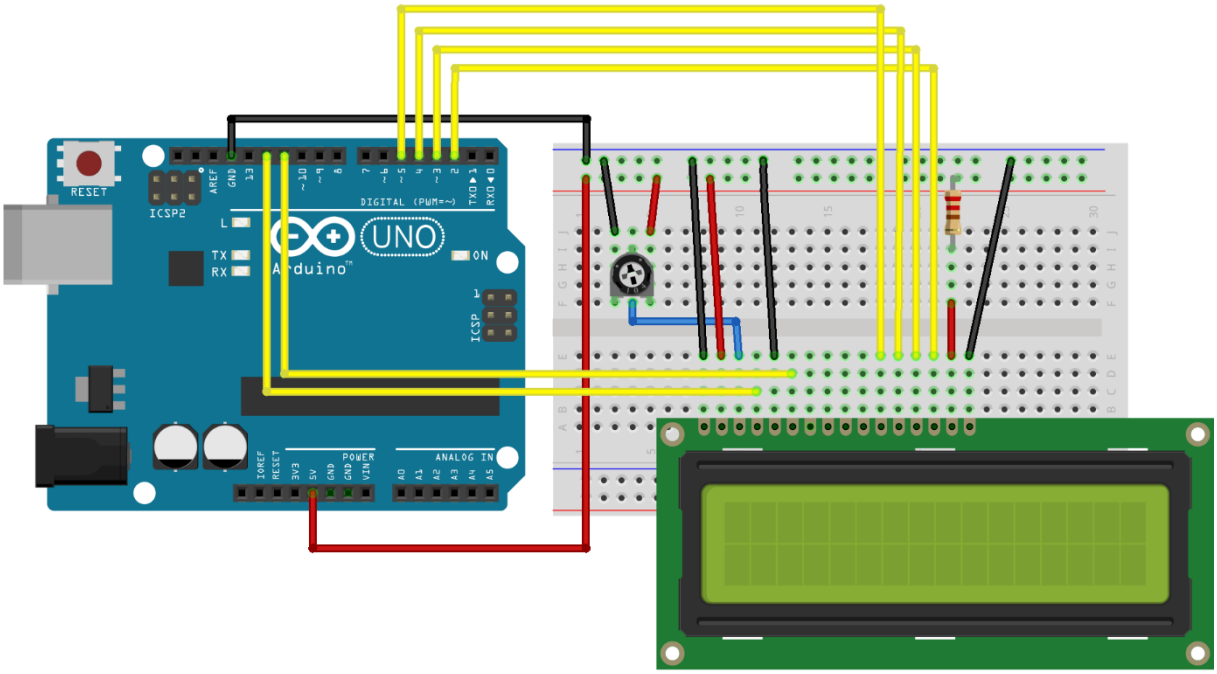
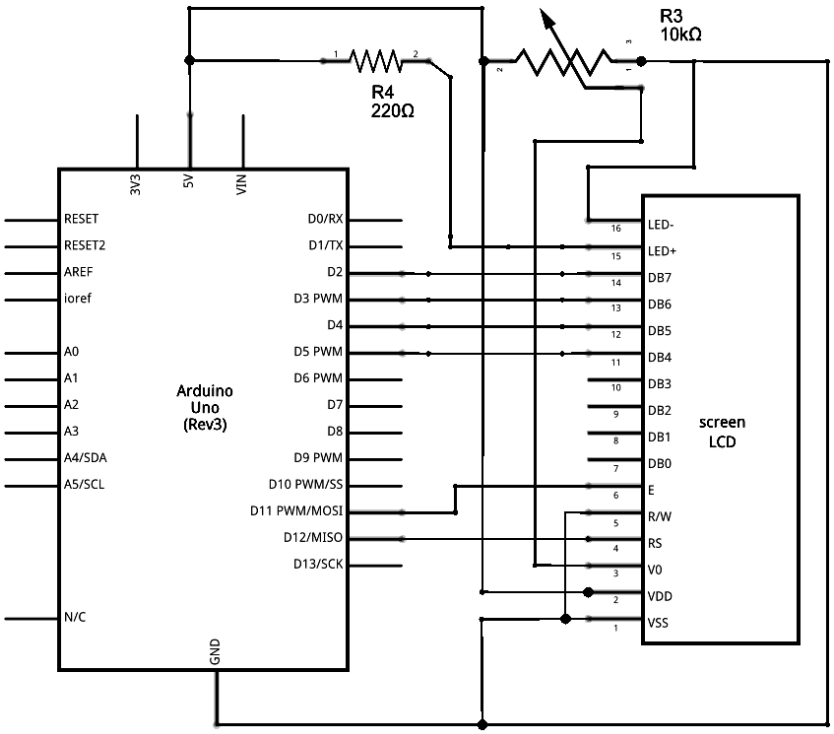


image developed using Fritzing. For more circuit examples, see the Fritzing project page

Schematic



Code in Assembly:

```
; ***************************************************************************
;   LCD-AVR-4d.asm  - Use an HD44780U based LCD with an Atmel ATmega processor
;
;   Copyright (C) 2013 Donald Weiman    (weimandn@alfredstate.edu)
;
;   This program is free software: you can redistribute it and/or modify
;   it under the terms of the GNU General Public License as published by
;   the Free Software Foundation, either version 3 of the License, or
;   (at your option) any later version.
;
;   This program is distributed in the hope that it will be useful,
;   but WITHOUT ANY WARRANTY; without even the implied warranty of
;   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
;   GNU General Public License for more details.
;
;   You should have received a copy of the GNU General Public License
;   along with this program.  If not, see <http://www.gnu.org/licenses/>.
;
; ***************************************************************************
;        File:    LCD-AVR-4d.asm
;        Date:    September 8, 2013
;
;      Target:    ATmega328
;   Assembler:    Atmel AvrAssembler2 (AVR Studio 6)
;      Author:    Donald Weiman
;
;    Hardware:    Arduino, Boarduino, or equivalent, LCD Module
;
;     Summary:    4-bit data interface, busy flag not implemented.
;                 Any LCD pin can be connected to any available I/O port.
;                 Includes a simple write string routine.
;
; ****************************** Program Notes ******************************
;
;           This program uses a 4-bit data interface but does not use the
;             busy flag to determine when the LCD controller is ready.  The
;             LCD RW line (pin 5) is not connected to the uP and it must be
;             connected to GND for the program to function.
;
;           All time delays are longer than those specified in most datasheets
;             in order to accommodate slower than normal LCD modules.  This
;             requirement is well documented but almost always ignored.  The
;             information is in a note at the bottom of the right hand
;             (Execution Time) column of the instruction set.
;
; ***************************************************************************
;
;           The four data lines as well as the two control lines may be
;             implemented on any available I/O pin of any port.  These are
;             the connections used for this program:
;
```

```
;                 -----------                    ----------
;                | ATmega328 |                  |   LCD    |
;                |           |                  |          |
;                |        PD7|----------------->|D7        |
;                |        PD6|----------------->|D6        |
;                |        PD5|----------------->|D5        |
;                |        PD4|----------------->|D4        |
;                |           |                  |D3        |
;                |           |                  |D2        |
;                |           |                  |D1        |
;                |           |                  |D0        |
;                |           |                  |          |
;                |        PB1|----------------->|E         |
;                |           |          GND --->|RW        |
;                |        PB0|----------------->|RS        |
;                 -----------                    ----------
;
; ***************************************************************************

/*                                      ; Studio 6 handles this by default
;.nolist
;.include "m328def.inc"
;.list
*/

.equ    fclk                = 16000000      ; system clock frequency (for delays)

; register usage
.def    temp                = R16           ; temporary storage

; LCD interface (should agree with the diagram above)
;   make sure that the LCD RW pin is connected to GND
.equ    lcd_D7_port         = PORTD         ; lcd D7 connection
.equ    lcd_D7_bit          = PORTD2
.equ    lcd_D7_ddr          = DDRD

.equ    lcd_D6_port         = PORTD         ; lcd D6 connection
.equ    lcd_D6_bit          = PORTD3
.equ    lcd_D6_ddr          = DDRD

.equ    lcd_D5_port         = PORTD         ; lcd D5 connection
.equ    lcd_D5_bit          = PORTD4
.equ    lcd_D5_ddr          = DDRD

.equ    lcd_D4_port         = PORTD         ; lcd D4 connection
.equ    lcd_D4_bit          = PORTD5
.equ    lcd_D4_ddr          = DDRD

.equ    lcd_E_port          = PORTB         ; lcd Enable pin
.equ    lcd_E_bit           = PORTB3
.equ    lcd_E_ddr           = DDRB

.equ    lcd_RS_port         = PORTB         ; lcd Register Select pin
.equ    lcd_RS_bit          = PORTB4
.equ    lcd_RS_ddr          = DDRB
```

```
; LCD module information
.equ    lcd_LineOne         = 0x00          ; start of line 1
.equ    lcd_LineTwo         = 0x40          ; start of line 2
;.equ   lcd_LineThree        = 0x14          ; start of line 3 (20x4)
;.equ   lcd_lineFour         = 0x54          ; start of line 4 (20x4)
;.equ   lcd_LineThree        = 0x10          ; start of line 3 (16x4)
;.equ   lcd_lineFour         = 0x50          ; start of line 4 (16x4)

; LCD instructions
.equ    lcd_Clear           = 0b00000001    ; replace all characters with ASCII 'space'
.equ    lcd_Home            = 0b00000010    ; return cursor to first position on first
line
.equ     lcd_EntryMode         = 0b00000110     ; shift cursor from left to right on
read/write
.equ    lcd_DisplayOff      = 0b00001000    ; turn display off
.equ    lcd_DisplayOn    = 0b00001100    ; display on, cursor off, don't blink character
.equ    lcd_FunctionReset   = 0b00110000    ; reset the LCD
.equ    lcd_FunctionSet4bit = 0b00101000    ; 4-bit data, 2-line display, 5 x 7 font
.equ    lcd_SetCursor       = 0b10000000    ; set cursor position

; ***************************** Reset Vector ******************************
.org    0x0000
    jmp     start                           ; jump over Interrupt Vectors, Program ID
etc.

;***************************** Program ID ********************************
.org    INT_VECTORS_SIZE

program_author:
.db         "HELLO THIS IS",0

program_version:
.db         "MY LCD PROGRAM",0,0

program_date:
.db         "Oct 02, 2018",0

; ***************************** Main Program Code ************************
start:
; initialize the stack pointer to the highest RAM address
    ldi     temp,low(RAMEND)
    out     SPL,temp
    ldi     temp,high(RAMEND)
    out     SPH,temp

; configure the microprocessor pins for the data lines
    sbi     lcd_D7_ddr, lcd_D7_bit          ; 4 data lines - output
    sbi     lcd_D6_ddr, lcd_D6_bit
    sbi     lcd_D5_ddr, lcd_D5_bit
    sbi     lcd_D4_ddr, lcd_D4_bit

; configure the microprocessor pins for the control lines
    sbi     lcd_E_ddr,  lcd_E_bit           ; E line - output
    sbi     lcd_RS_ddr, lcd_RS_bit          ; RS line - output
```

```
; initialize the LCD controller as determined by the equates (LCD instructions)
    call    lcd_init_4d                     ; initialize the LCD display for a 4-bit
interface

; display the first line of information
    ldi     ZH, high(program_author)  ; point to the information that is to be displayed
    ldi      ZL, low(program_author)
    ldi     temp, lcd_LineOne      ; point to where the information should be displayed
    call     lcd_write_string_4d
; display the second line of information
    ldi     ZH, high(program_version) ; point to the information that is to be displayed
    ldi      ZL, low(program_version)
    ldi     temp, lcd_LineTwo      ; point to where the information should be displayed
    call     lcd_write_string_4d

; endless loop
here:
    rjmp    here

; ***************************** End of Main Program Code ******************

; ============================== 4-bit LCD Subroutines =====================
; Name:     lcd_init_4d
; Purpose:  initialize the LCD module for a 4-bit data interface
; Entry:    equates (LCD instructions) set up for the desired operation
; Exit:     no parameters
; Notes:    uses time delays instead of checking the busy flag

lcd_init_4d:
; Power-up delay
    ldi     temp, 100                       ; initial 40 mSec delay
    call    delayTx1mS

; IMPORTANT - At this point the LCD module is in the 8-bit mode and it is expecting to
receive
;    8 bits of data, one bit on each of its 8 data lines, each time the 'E' line is
pulsed.
;
;  Since the LCD module is wired for the 4-bit mode, only the upper four data lines
are connected to
;      the microprocessor and the lower four data lines are typically left open.
Therefore, when
;    the 'E' line is pulsed, the LCD controller will read whatever data has been set
up on the upper
;    four data lines and the lower four data lines will be high (due to internal pull-
up circuitry).
;
;  Fortunately the 'FunctionReset' instruction does not care about what is on the lower
four bits so
;    this instruction can be sent on just the four available data lines and it will be
interpreted
;     properly by the LCD controller.  The 'lcd_write_4' subroutine will accomplish
this if the
;    control lines have previously been configured properly.
```

```
; Set up the RS and E lines for the 'lcd_write_4' subroutine.
    cbi     lcd_RS_port, lcd_RS_bit        ; select the Instruction Register (RS low)
    cbi     lcd_E_port, lcd_E_bit          ; make sure E is initially low

; Reset the LCD controller.
    ldi     temp, lcd_FunctionReset        ; first part of reset sequence
    call    lcd_write_4
    ldi     temp, 10                       ; 4.1 mS delay (min)
    call    delayTx1mS

    ldi     temp, lcd_FunctionReset        ; second part of reset sequence
    call    lcd_write_4
    ldi     temp, 200                      ; 100 uS delay (min)
    call    delayTx1uS

    ldi     temp, lcd_FunctionReset        ; third part of reset sequence
    call    lcd_write_4
    ldi     temp, 200                      ; this delay is omitted in the data sheet
    call    delayTx1uS

; Preliminary Function Set instruction - used only to set the 4-bit mode.
; The number of lines or the font cannot be set at this time since the controller is
still in the
;    8-bit mode, but the data transfer mode can be changed since this parameter is
determined by one
;   of the upper four bits of the instruction.
    ldi     temp, lcd_FunctionSet4bit      ; set 4-bit mode
    call    lcd_write_4
    ldi     temp, 80                       ; 40 uS delay (min)
    call    delayTx1uS

; Function Set instruction
    ldi     temp, lcd_FunctionSet4bit      ; set mode, lines, and font
    call    lcd_write_instruction_4d
    ldi     temp, 80                       ; 40 uS delay (min)
    call    delayTx1uS

; The next three instructions are specified in the data sheet as part of the
initialization routine,
;   so it is a good idea (but probably not necessary) to do them just as specified and
then redo them
;   later if the application requires a different configuration.

; Display On/Off Control instruction
    ldi     temp, lcd_DisplayOff           ; turn display OFF
    call    lcd_write_instruction_4d
    ldi     temp, 80                       ; 40 uS delay (min)
    call    delayTx1uS

; Clear Display instruction
    ldi     temp, lcd_Clear                ; clear display RAM
    call    lcd_write_instruction_4d
    ldi     temp, 4                        ; 1.64 mS delay (min)
    call    delayTx1mS
```

```
; Entry Mode Set instruction
    ldi     temp, lcd_EntryMode             ; set desired shift characteristics
    call    lcd_write_instruction_4d
    ldi     temp, 80                        ; 40 uS delay (min)
    call    delayTx1uS

; This is the end of the LCD controller initialization as specified in the data sheet,
but the display
;   has been left in the OFF condition.This is a good time to turn the display back
ON.

; Display On/Off Control instruction
    ldi     temp, lcd_DisplayOn             ; turn the display ON
    call    lcd_write_instruction_4d
    ldi     temp, 80                        ; 40 uS delay (min)
    call    delayTx1uS
    ret

; ---------------------------------------------------------------------------
; Name:     lcd_write_string_4d
; Purpose:  display a string of characters on the LCD
; Entry:    ZH and ZL pointing to the start of the string
;           (temp) contains the desired DDRAM address at which to start the display
; Exit:     no parameters
; Notes:    the string must end with a null (0)
;           uses time delays instead of checking the busy flag

lcd_write_string_4d:
; preserve registers
    push    ZH                              ; preserve pointer registers
    push    ZL

; fix up the pointers for use with the 'lpm' instruction
    lsl     ZL                               ; shift the pointer one bit left for the
lpm instruction
    rol     ZH

; set up the initial DDRAM address
    ori     temp, lcd_SetCursor            ; convert the plain address to a set cursor
instruction
    call   lcd_write_instruction_4d         ; set up the first DDRAM address
    ldi     temp, 80                        ; 40 uS delay (min)
    call    delayTx1uS

; write the string of characters
lcd_write_string_4d_01:
    lpm     temp, Z+                        ; get a character
    cpi     temp,  0                        ; check for end of string
    breq    lcd_write_string_4d_02          ; done

; arrive here if this is a valid character
    call    lcd_write_character_4d          ; display the character
    ldi     temp, 80                        ; 40 uS delay (min)
    call    delayTx1uS
    rjmp    lcd_write_string_4d_01          ; not done, send another character
```

```
; arrive here when all characters in the message have been sent to the LCD module
lcd_write_string_4d_02:
    pop     ZL                          ; restore pointer registers
    pop     ZH
    ret

; ------------------------------------------------------------------------
; Name:     lcd_write_character_4d
; Purpose:  send a byte of information to the LCD data register
; Entry:    (temp) contains the data byte
; Exit:     no parameters
; Notes:    does not deal with RW (busy flag is not implemented)

lcd_write_character_4d:
    sbi     lcd_RS_port, lcd_RS_bit         ; select the Data Register (RS high)
    cbi     lcd_E_port, lcd_E_bit           ; make sure E is initially low
    call    lcd_write_4                     ; write the upper 4-bits of the data
    swap    temp                            ; swap high and low nibbles
    call    lcd_write_4                     ; write the lower 4-bits of the data
    ret

; ------------------------------------------------------------------------
; Name:     lcd_write_instruction_4d
; Purpose:  send a byte of information to the LCD instruction register
; Entry:    (temp) contains the data byte
; Exit:     no parameters
; Notes:    does not deal with RW (busy flag is not implemented)

lcd_write_instruction_4d:
    cbi     lcd_RS_port, lcd_RS_bit         ; select the Instruction Register (RS low)
    cbi     lcd_E_port, lcd_E_bit           ; make sure E is initially low
    call    lcd_write_4                     ; write the upper 4-bits of the instruction
    swap    temp                            ; swap high and low nibbles
    call    lcd_write_4                     ; write the lower 4-bits of the instruction
    ret

; ------------------------------------------------------------------------
; Name:     lcd_write_4
; Purpose:  send a nibble (4-bits) of information to the LCD module
; Entry:    (temp) contains a byte of data with the desired 4-bits in the upper nibble
;           (RS) is configured for the desired LCD register
;           (E) is low
;           (RW) is low
; Exit:     no parameters
; Notes:    use either time delays or the busy flag

lcd_write_4:
; set up D7
    sbi     lcd_D7_port, lcd_D7_bit         ; assume that the D7 data is '1'
    sbrs    temp, 7                         ; check the actual data value
    cbi     lcd_D7_port, lcd_D7_bit     ; arrive here only if the data was actually '0'
```

```
; set up D6
    sbi     lcd_D6_port, lcd_D6_bit         ; repeat for each data bit
    sbrs    temp, 6
    cbi     lcd_D6_port, lcd_D6_bit

; set up D5
    sbi     lcd_D5_port, lcd_D5_bit
    sbrs    temp, 5
    cbi     lcd_D5_port, lcd_D5_bit

; set up D4
    sbi     lcd_D4_port, lcd_D4_bit
    sbrs    temp, 4
    cbi     lcd_D4_port, lcd_D4_bit

; write the data
                                        ; 'Address set-up time' (40 nS)
    sbi     lcd_E_port, lcd_E_bit       ; Enable pin high
    call    delay1uS                    ; implement 'Data set-up time' (80 nS) and
'Enable pulse width' (230 nS)
    cbi     lcd_E_port, lcd_E_bit       ; Enable pin low
    call    delay1uS                     ; implement 'Data hold time' (10 nS) and
'Enable cycle time' (500 nS)
    ret

; ============================== End of 4-bit LCD Subroutines ===============

; ============================== Time Delay Subroutines =====================
; Name:     delayYx1mS
; Purpose:  provide a delay of (YH:YL) x 1 mS
; Entry:    (YH:YL) = delay data
; Exit:     no parameters
; Notes:    the 16-bit register provides for a delay of up to 65.535 Seconds
;           requires delay1mS

delayYx1mS:
    call    delay1mS                    ; delay for 1 mS
    sbiw    YH:YL, 1                    ; update the the delay counter
    brne    delayYx1mS                  ; counter is not zero

; arrive here when delay counter is zero (total delay period is finished)
    ret

; ---------------------------------------------------------------------------
; Name:     delayTx1mS
; Purpose:  provide a delay of (temp) x 1 mS
; Entry:    (temp) = delay data
; Exit:     no parameters
; Notes:    the 8-bit register provides for a delay of up to 255 mS
;           requires delay1mS

delayTx1mS:
    call    delay1mS                    ; delay for 1 mS
    dec     temp                        ; update the delay counter
    brne    delayTx1mS                  ; counter is not zero
```

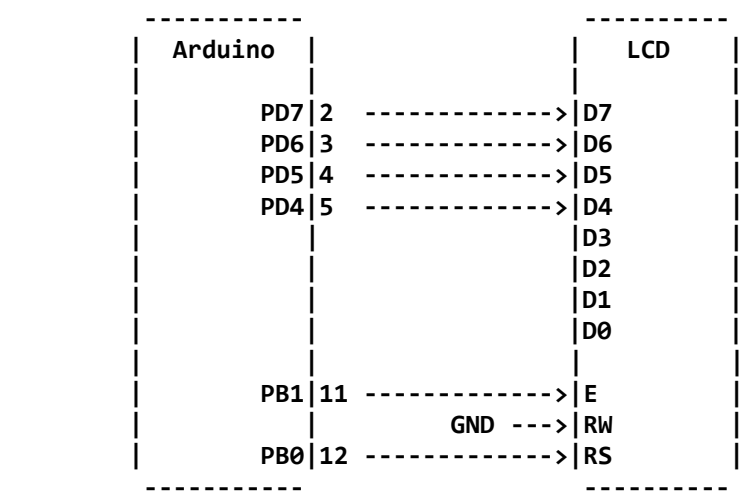
```
; arrive here when delay counter is zero (total delay period is finished)
    ret

; ------------------------------------------------------------------------
; Name:     delay1mS
; Purpose:  provide a delay of 1 mS
; Entry:    no parameters
; Exit:     no parameters
; Notes:    chews up fclk/1000 clock cycles (including the 'call')

delay1mS:
    push    YL                              ; [2] preserve registers
    push    YH                              ; [2]
    ldi     YL, low (((fclk/1000)-18)/4)    ; [1] delay counter
    ldi     YH, high(((fclk/1000)-18)/4)    ; [1]

delay1mS_01:
    sbiw    YH:YL, 1                        ; [2] update the the delay counter
    brne    delay1mS_01                     ; [2] delay counter is not zero

; arrive here when delay counter is zero
    pop     YH                              ; [2] restore registers
    pop     YL                              ; [2]
    ret                                     ; [4]

; ------------------------------------------------------------------------
; Name:     delayTx1uS
; Purpose:  provide a delay of (temp) x 1 uS with a 16 MHz clock frequency
; Entry:    (temp) = delay data
; Exit:     no parameters
; Notes:    the 8-bit register provides for a delay of up to 255 uS
;           requires delay1uS

delayTx1uS:
    call    delay1uS                        ; delay for 1 uS
    dec     temp                            ; decrement the delay counter
    brne    delayTx1uS                      ; counter is not zero

; arrive here when delay counter is zero (total delay period is finished)
    ret

; ------------------------------------------------------------------------
; Name:     delay1uS
; Purpose:  provide a delay of 1 uS with a 16 MHz clock frequency
; Entry:    no parameters
; Exit:     no parameters
; Notes:    add another push/pop for 20 MHz clock frequency

delay1uS:
    push    temp                  ; [2] these instructions do nothing except consume
clock cycles
    pop     temp                ; [2]
    push    temp                ; [2]
    pop     temp                ; [2]
```

```
    ret                         ; [4]

; ============================== End of Time Delay Subroutines ==============
```

CODE IN C:

```
/************************************************************************
    LCD_AVR_4d.ino  - Use an HD44780U based LCD with an Arduino
     Copyright (C) 2013 Donald Weiman    (weimandn@alfredstate.edu)
     This program is free software: you can redistribute it and/or modify
    it under the terms of the GNU General Public License as published by
    the Free Software Foundation, either version 3 of the License, or
    (at your option) any later version.
     This program is distributed in the hope that it will be useful,
    but WITHOUT ANY WARRANTY; without even the implied warranty of
    MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
    GNU General Public License for more details.
     You should have received a copy of the GNU General Public Licensealong
     with this   program.  If not, see <http://www.gnu.org/licenses/>.
*/
/************************************************************************
         File:    LCD_AVR_4d.ino
         Date:    September 16, 2013

       Target:    Arduino (ATmega328)
     Compiler:    Arduino IDE (v1.0.5)
       Author:    Donald Weiman

      Summary:    4-bit data interface, busy flag not implemented.
                  Any LCD pin can be connected to any available I/O port.
                  Includes a simple write string routine.
*/
/****************************** Program Notes ******************************
            This program uses an 4-bit data interface but does not use the
              busy flag to determine when the LCD controller is ready.  The
              LCD RW line (pin 5) is not connected to the uP and it must be
              connected to GND for the program to function.

            All time delays are longer than those specified in most datasheets
              in order to accommodate slower than normal LCD modules.  This
              requirement is well documented but almost always ignored.  The
              information is in a note at the bottom of the right hand
              (Execution Time) column of the instruction set.


  ************************************************************************
```

```
The four data lines as well as the two control lines may be
  implemented on any available I/O pin of any port.  These are
  the connections used for this program:

The numbers shown next to the Arduino are the Arduino pin
numbers, not the IC pin numbers.
```




```
  **************************************************************************/

// LCD interface (should agree with the diagram above)
//   make sure that the LCD RW pin is connected to GND
uint8_t  lcd_D7_ArdPin   =  2;                  // lcd D7 connection
uint8_t  lcd_D6_ArdPin   =  3;
uint8_t  lcd_D5_ArdPin   =  4;
uint8_t  lcd_D4_ArdPin   =  5;

uint8_t  lcd_E_ArdPin    =  11;                  // lcd Enable pin
uint8_t  lcd_RS_ArdPin   =  12;                  // lcd Register Select pin

// LCD module information
#define lcd_LineOne     0x00                    // start of line 1
#define lcd_LineTwo     0x40                    // start of line 2
//#define   lcd_LineThree   0x14                  // start of line 3 (20x4)
//#define   lcd_lineFour    0x54                  // start of line 4 (20x4)
//#define   lcd_LineThree   0x10                  // start of line 3 (16x4)
//#define   lcd_lineFour    0x50                  // start of line 4 (16x4)

// LCD instructions
#define lcd_Clear           0b00000001          // replace all characters with ASCII
'space'
#define lcd_Home            0b00000010          // return cursor to first position on
first line
```

```
#define lcd_EntryMode       0b00000110          // shift cursor from left to right on
read/write
#define lcd_DisplayOff      0b00001000          // turn display off
#define lcd_DisplayOn       0b00001100         // display on, cursor off, don't blink
character
#define lcd_FunctionReset   0b00110000          // reset the LCD
#define lcd_FunctionSet4bit 0b00101000          // 4-bit data, 2-line display, 5 x 7
font
#define lcd_SetCursor       0b10000000          // set cursor position

// Program ID
uint8_t program_author[]   = "HOLA AMUY";
uint8_t program_version[]  = "TU PROGRAMA LCD";
uint8_t program_date[]     = "Oct 02, 2018";


/****************************** Main Program Code ************************/

void setup (void)
{
// configure the microprocessor pins for the data lines
    pinMode(lcd_D7_ArdPin, OUTPUT);                 // 8 data lines - output
    pinMode(lcd_D6_ArdPin, OUTPUT);
    pinMode(lcd_D5_ArdPin, OUTPUT);
    pinMode(lcd_D4_ArdPin, OUTPUT);

// configure the microprocessor pins for the control lines
    pinMode(lcd_E_ArdPin, OUTPUT);                  // E line - output
    pinMode(lcd_RS_ArdPin, OUTPUT);                 // RS line - output

// initialize the LCD controller as determined by the defines (LCD instructions)
    lcd_init_4d();                                  // initialize the LCD display for
an 8-bit interface

// display the first line of information
    lcd_write_string_4d(program_author);

// set cursor to start of second line
    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineTwo);
    delayMicroseconds(80);                          // 40 uS delay (min)

// display the second line of information
    lcd_write_string_4d(program_version);
}

// endless loop
void loop()
  {
  }
/****************************** End of Main Program Code *****************/

/*============================== 4-bit LCD Functions =====================*/
/*
  Name:     lcd_init_4d
  Purpose:  initialize the LCD module for a 8-bit data interface
```

```
  Entry:    equates (LCD instructions) set up for the desired operation
  Exit:     no parameters
  Notes:    uses time delays rather than checking the busy flag
*/
void lcd_init_4d(void)
{
// Power-up delay
    delay(100);                                     // initial 40 mSec delay

 // IMPORTANT - At this point the LCD module is in the 8-bit mode and it is expecting
to receive
//   8 bits of data, one bit on each of its 8 data lines, each time the 'E' line is
pulsed.
//
// Since the LCD module is wired for the 4-bit mode, only the upper four data lines
are connected to
//    the microprocessor and the lower four data lines are typically left open.
Therefore, when
//   the 'E' line is pulsed, the LCD controller will read whatever data has been set
up on the upper
//   four data lines and the lower four data lines will be high (due to internal pull-
up circuitry).
//
// Fortunately the 'FunctionReset' instruction does not care about what is on the lower
four bits so
//   this instruction can be sent on just the four available data lines and it will be
interpreted
//    properly by the LCD controller.  The 'lcd_write_4' subroutine will accomplish
this if the
//   control lines have previously been configured properly.

// Set up the RS and E lines for the 'lcd_write_4' subroutine.
    digitalWrite(lcd_RS_ArdPin, LOW);              // select the Instruction Register
(RS low)
    digitalWrite(lcd_E_ArdPin, LOW);                // make sure E is initially low

// Reset the LCD controller
    lcd_write_4(lcd_FunctionReset);                 // first part of reset sequence
    delay(10);                                      // 4.1 mS delay (min)

    lcd_write_4(lcd_FunctionReset);                 // second part of reset sequence
    delayMicroseconds(200);                         // 100uS delay (min)

    lcd_write_4(lcd_FunctionReset);                 // third part of reset sequence
    delayMicroseconds(200);                           // this delay is omitted in the
data sheet

// Preliminary Function Set instruction - used only to set the 4-bit mode.
// The number of lines or the font cannot be set at this time since the controller is
still in the
//  8-bit mode, but the data transfer mode can be changed since this parameter is
determined by one
//  of the upper four bits of the instruction.

    lcd_write_4(lcd_FunctionSet4bit);               // set 4-bit mode
```

```
    delayMicroseconds(80);                          // 40 uS delay (min)

// Function Set instruction
    lcd_write_instruction_4d(lcd_FunctionSet4bit);  // set mode, lines, and font
    delayMicroseconds(80);                          // 40uS delay (min)

//  The next three instructions are specified in the data sheet as part of the
initialization routine,
//  so it is a good idea (but probably not necessary) to do them just as specified and
then redo them
//  later if the application requires a different configuration.

// Display On/Off Control instruction
    lcd_write_instruction_4d(lcd_DisplayOff);       // turn display OFF
    delayMicroseconds(80);                          // 40 uS delay (min)

// Clear Display instruction
    lcd_write_instruction_4d(lcd_Clear);            // clear display RAM
    delay(4);                                       // 1.64 mS delay (min)

// ; Entry Mode Set instruction
    lcd_write_instruction_4d(lcd_EntryMode);      // set desired shift characteristics
    delayMicroseconds(80);                          // 40 uS delay (min)

// This is the end of the LCD controller initialization as specified in the data sheet,
but the display
//  has been left in the OFF condition.  This is a good time to turn the display back
ON.

// Display On/Off Control instruction
    lcd_write_instruction_4d(lcd_DisplayOn);        // turn the display ON
    delayMicroseconds(80);                          // 40 uS delay (min)
}

/*.....................................................................
  Name:     lcd_write_string_4d
; Purpose:  display a string of characters on the LCD
  Entry:    (theString) is the string to be displayed
  Exit:     no parameters
  Notes:    uses time delays rather than checking the busy flag
*/
void lcd_write_string_4d(uint8_t theString[])
{
    volatile int i = 0;                             // character counter*/
    while (theString[i] != 0)
    {
        lcd_write_character_4d(theString[i]);
        i++;
        delayMicroseconds(80);                          // 40 uS delay (min)
    }
}

/*.....................................................................
  Name:     lcd_write_character_4d
  Purpose:  send a byte of information to the LCD data register
```

```
  Entry:    (theData) is the information to be sent to the data register
  Exit:     no parameters
  Notes:    does not deal with RW (busy flag is not implemented)
*/
void lcd_write_character_4d(uint8_t  theData)
{
    digitalWrite(lcd_RS_ArdPin, HIGH);               // select the Data Register (RS
high)
    digitalWrite(lcd_E_ArdPin, LOW);                // make sure E is initially low
    lcd_write_4(theData);                            // write the upper 4-bits of the
data
    lcd_write_4(theData << 4);                       // write the lower 4-bits of the
data
}

/*.....................................................................
  Name:     lcd_write_instruction_4d
  Purpose:  send a byte of information to the LCD instruction register
  Entry:    (theInstruction) is the information to be sent to the instruction register
  Exit:     no parameters
  Notes:    does not deal with RW (busy flag is not implemented)
*/
void lcd_write_instruction_4d(uint8_t  theInstruction)
{
    digitalWrite(lcd_RS_ArdPin, LOW);              // select the Instruction Register
(RS low)
    digitalWrite(lcd_E_ArdPin, LOW);                // make sure E is initially low
    lcd_write_4(theInstruction);                     // write the upper 4-bits of the
data
    lcd_write_4(theInstruction << 4);                // write the lower 4-bits of the
data
}

/*.....................................................................
  Name:     lcd_write_4
  Purpose:  send a nibble of information to the LCD module
  Entry:    (theByte) contains the information to be sent to the desired LCD register
            RS is configured for the desired LCD register
            E is low
            RW is low
  Exit:     no parameters
  Notes:    the desired information (4-bits) must be in the upper half of (theByte)
            use either time delays or the busy flag
*/
void lcd_write_4(uint8_t  theByte)
{
    digitalWrite(lcd_D7_ArdPin,LOW);                        // assume that data is '0'
    if (theByte & 1<<7) digitalWrite(lcd_D7_ArdPin,HIGH);   // make data = '1' if
necessary

    digitalWrite(lcd_D6_ArdPin,LOW);                       // repeat for each data bit
    if (theByte & 1<<6) digitalWrite(lcd_D6_ArdPin,HIGH);

    digitalWrite(lcd_D5_ArdPin,LOW);
    if (theByte & 1<<5) digitalWrite(lcd_D5_ArdPin,HIGH);
```

```
    digitalWrite(lcd_D4_ArdPin,LOW);
    if (theByte & 1<<4) digitalWrite(lcd_D4_ArdPin,HIGH);

// write the data
                                                  // 'Address set-up time' (40 nS)
    digitalWrite(lcd_E_ArdPin,HIGH);              // Enable pin high
    delayMicroseconds(1);                           // implement 'Data set-up time'
(80 nS) and 'Enable pulse width' (230 nS)
    digitalWrite(lcd_E_ArdPin,LOW);               // Enable pin low
    delayMicroseconds(1);   // implement 'Data hold time' (10 nS) and 'Enable cycle
time' (500 nS)
}
```