```
/*************************************************************************
<For Windows Atmel Studio>
cd "Atmel Studio₩7.0₩ATmega328P₩LCD₩LCD₩Debug"
OR cd "C:₩Users₩Insoo₩Documents₩Atmel Studio₩7.0₩ATmega328P₩LCD₩LCD₩Debug"

avrdude -c usbtiny -P usb -p atmega328p -U flash:w:LCD.hex:i

 */

/************************************************************
 Target MCU & clock speed: ATmega328P @ 1Mhz internal
 Name    : main.c
 C modules of this project, LCD:
    main.c globals.c intrpt.c strFunc.c util.c
 custom Headers:
    defines.h externs.h
 Author  : Insoo Kim (insoo@hotmail.com)
 Created : May 15, 2015
 Updated : Oct 9, 2016 (On Atmel Studio 7)

 Description: Get system compile time & date and display on LCD 2*16
    Button toggling to turn on or off the backlight of LCD

 HEX size[Byte]: 3340 out of 32K (all modules built together) w/prev. version of
   Atmel Studio
    5308 w/ATmel Studio 7

 Ref:
    Donald Weiman    (weimandn@alfredstate.edu)
    http://web.alfredstate.edu/weimandn/programming/lcd/ATmega328/
      LCD_code_gcc_4d.html
 ************************************************************/


#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <avr/io.h>
#include <avr/interrupt.h>
#include <avr/wdt.h>
#include <avr/sleep.h>
#include "externs.h"
#include "defines.h"
#include <util/delay.h>


/***************************** Main Program Code ************************/
int main(void)
{
    config();
    //ioinit(); //dht11
    //init_devices();
    initINT();
    parseCompileTime();
    lcd_dispRealClock();
    // Use the Power Down sleep mode
```

```
    set_sleep_mode(SLEEP_MODE_PWR_DOWN);

    // endless loop
    while(1)
    {
      // go to sleep and wait for interrupt...
      sleep_mode();
      //sysClockTest();
      //lcd_dispRealClock();
      //lcd_showDHT11();
      //_delay_ms(1000);
      //lcd_dispProgInfo();
      //_delay_ms(1000);


        /*
        PORTB |= _BV(debug_PIN);
        _delay_ms(500);
        PORTB &= ~_BV(debug_PIN);
        _delay_ms(500);
        */
    }
    return 0;
}
/****************************** End of Main Program Code *****************/
void chkButtonAndToggleBacklight()
{
    uint8_t valSwitch;

    valSwitch = tactile_Switch_port & _BV(tactile_Switch_bit);
    _delay_ms(200);

    //if switch is pressed,
    if (valSwitch == 0)
        //toggle backlight by allow K to touch ground
        lcd_Backlight_port ^= _BV(lcd_Backlight_bit);
}//chkButtonAndToggleBacklight

void config()
{
// configure the microprocessor pins for the data lines
// 4 data lines - output
    lcd_D7_ddr |= _BV(lcd_D7_bit);
    lcd_D6_ddr |= _BV(lcd_D6_bit);
    lcd_D5_ddr |= _BV(lcd_D5_bit);
    lcd_D4_ddr |= _BV(lcd_D4_bit);

// LCD backlight cathode pin (K) - Output
    lcd_Backlight_ddr |= _BV(lcd_Backlight_bit);
    //turn off LCD backlight
    lcd_Backlight_port |= _BV(lcd_Backlight_bit);

//Tactile switch - Input
    tactile_Switch_ddr &= ~_BV(tactile_Switch_bit);

// configure the microprocessor pins for the control lines
```

```
// E line - output
    lcd_E_ddr |= _BV(lcd_E_bit);
// RS line - output
    lcd_RS_ddr |= _BV(lcd_RS_bit);

// LCD VDD pin - Output
    //lcd_VDD_ddr |= _BV(lcd_VDD_bit);
    //turn off LCD VDD
    //lcd_VDD_port &= ~_BV(lcd_VDD_bit);
    //turn on LCD VDD
    //lcd_VDD_port |= _BV(lcd_VDD_bit);

// initialize the LCD controller as determined by the defines (LCD instructions)
// initialize the LCD display for a 4-bit interface
    lcd_init_4d();

}//config

void turnOnLCDBacklight()
{
    //turn on LCD backlight
    //  by giving 0 volt to K of LCD
    lcd_Backlight_port &= ~_BV(lcd_Backlight_bit);
}//turnOnLCDBacklight

void turnOffLCDBacklight()
{
    //turn off LCD backlight
    //  by giving 5 volt to K of LCD
    lcd_Backlight_port |= _BV(lcd_Backlight_bit);
}//turnOffLCDBacklight

void lcd_SysTime()
{
// set cursor to start of first line
    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineOne);
    _delay_us(DELAY_INST);                          // 40 uS delay (min)
    lcd_write_string_4d((uint8_t *)__TIME__);
    _delay_us(DELAY_INST);                          // 40 uS delay (min)

// set cursor to start of first line
    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineTwo);
    _delay_us(DELAY_INST);                          // 40 uS delay (min)
    lcd_write_string_4d((uint8_t *)__DATE__);
    _delay_us(DELAY_INST);                          // 40 uS delay (min)

}//lcd_SysTime

void lcd_dispOFF()
{
// set LCD off
    lcd_write_instruction_4d(lcd_DisplayOff);
    _delay_us(DELAY_INST);                          // 40 uS delay (min)
}//lcd_dispOFF

void lcd_dispON()
```

```
{
// set LCD off
    lcd_write_instruction_4d(lcd_DisplayOn);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
}//lcd_dispON

void lcd_dispRealClock()
{
    char strSec[3], strMin[3], strHour[3];
    char strYear[3], strMonth[3], strDate[3];
    char strDay[10];
// set cursor to start of first line
    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineOne);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
// display the first line of information
    itoa(hour, strHour, 10);
    itoa(min, strMin, 10);
    itoa(sec, strSec, 10);
    lcd_write_string_4d((uint8_t *)strHour);
    lcd_write_string_4d((uint8_t *)":");
    lcd_write_string_4d((uint8_t *)strMin);
    lcd_write_string_4d((uint8_t *)":");
    lcd_write_string_4d((uint8_t *)strSec);
    lcd_write_string_4d((uint8_t *)"           ");
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)

// set cursor to start of first line
    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineTwo);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
    itoa(year, strYear, 10);
    itoa(month, strMonth, 10);
    itoa(date, strDate, 10);
    lcd_write_string_4d((uint8_t *)strYear);
    lcd_write_string_4d((uint8_t *)"/");
    lcd_write_string_4d((uint8_t *)strMonth);
    lcd_write_string_4d((uint8_t *)"/");
    lcd_write_string_4d((uint8_t *)strDate);
    lcd_write_string_4d((uint8_t *)" ");

    switch (day)
    {
        case 0:
            strcpy(strDay,"Sun");
            break;
        case 1:
            strcpy(strDay,"Mon");
            break;
        case 2:
            strcpy(strDay,"Tue");
            break;
        case 3:
            strcpy(strDay,"Wed");
            break;
        case 4:
            strcpy(strDay,"Thu");
            break;
```

```
        case 5:
            strcpy(strDay,"Fri");
            break;
        case 6:
            strcpy(strDay,"Sat");
            break;
    }
    lcd_write_string_4d((uint8_t *)strDay);
    lcd_write_string_4d((uint8_t *)"     ");
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)


}//lcd_dispRealClock

void lcd_dispAccumulatedTime()
{
    char strSec[3], strMin[3], strHour[3];

// set cursor to start of 2nd line
    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineTwo);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
// display the first line of information
    itoa(accumulatedHour, strHour, 10);
    itoa(accumulatedMin, strMin, 10);
    itoa(accumulatedSec, strSec, 10);
    lcd_write_string_4d((uint8_t *)strHour);
    lcd_write_string_4d((uint8_t *)":");
    lcd_write_string_4d((uint8_t *)strMin);
    lcd_write_string_4d((uint8_t *)":");
    lcd_write_string_4d((uint8_t *)strSec);
    lcd_write_string_4d((uint8_t *)"       ");
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
}//lcd_dispAccumulatedTime

void lcd_showDHT11()
{
    int8_t temperature, humidity;
    char str[3];

    //dht_getdata(&temperature, &humidity);
    //dht_gettemperaturehumidity(&temperature, &humidity);

    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineOne);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)

    lcd_write_string_4d((uint8_t *)"Temp:");

    //lcd_write_4(temperature);
    //sprintf(str, "%d", temperature);
    //temperature = 19;
    itoa(temperature,str, 10);
    lcd_write_string_4d((uint8_t *)str);
    lcd_write_string_4d((uint8_t *)"    ");
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)

    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineTwo);
```

```
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
    lcd_write_string_4d((uint8_t *)"Humidity:");

    itoa(humidity,str, 10);
    lcd_write_string_4d((uint8_t *)str);
    lcd_write_string_4d((uint8_t *)"        ");
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
}//lcd_showDHT11

void lcd_dispProgInfo()
{
// set cursor to start of first line
    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineOne);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
// 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);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)

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

    _delay_ms(1000);

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

// display the second line of information
    lcd_write_string_4d(program_date);

    _delay_ms(2000);

}//lcd_dispProgInfo

void lcd_dispMenu()
{
// set cursor to start of first line
    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineOne);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
// display the first line of information
    lcd_write_string_4d(menu_str1);

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

// display the second line of information
    lcd_write_string_4d(menu_str2);

    _delay_ms(3000);

}//lcd_dispMenu
```

```
void lcd_dispWords(uint8_t i)
{
    uint8_t n, wordLen;
    uint8_t *words0=0, *words1=0;
    //uint8_t str[3];
    // set cursor to start of first line
        lcd_write_instruction_4d(lcd_SetCursor | lcd_LineOne);
        _delay_us(DELAY_INST);                                  // 40 uS delay
          (min)
    // display the first line of information
        //words = malloc(320);
        switch (min%4)
        {
            case 0:
            case 1:
                words0 = words000[i][0];
                words1 = words000[i][1];
                break;
            case 2:
            case 3:
                words0 = words001[i][0];;
                words1 = words001[i][1];
                break;
        }
        //lcd_write_string_4d(words000[i][0]);
        lcd_write_string_4d(words0);
        //wordLen = strlen((char *)words000[i][0]);
        wordLen = strlen((char *)words0);
        for (n=0; n<(LCD_MAXCOL-wordLen); n++)
            lcd_write_character_4d((uint8_t)0x20);
        //lcd_write_string_4d(program_author);

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

    // display the second line of information
        //lcd_write_string_4d(words000[i][1]);
        lcd_write_string_4d(words1);
        //utoa(sizeof(words000[0][0]), str, 10); //16
        //utoa(sizeof(words000[0]), str, 10); //32
        //utoa(sizeof(words), str, 10); //?
        //utoa(strlen(words000[i][0]), str, 10);
        //lcd_write_string_4d(str);
        //wordLen = strlen((char *)words000[i][1]);
        wordLen = strlen((char *)words1);
        for (n=0; n<(LCD_MAXCOL-wordLen); n++)
            lcd_write_character_4d((uint8_t)0x20);
        //lcd_write_string_4d(program_date);

        //_delay_ms(2000);

}//lcd_dispWords

void lcd_testString()
```

```
{
// set cursor to start of first line
    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineOne);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
// 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);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)

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

    _delay_ms(1000);

// set cursor to start of second line
    lcd_write_instruction_4d(lcd_SetCursor | lcd_LineTwo);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
// display the first line of information
    lcd_write_string_4d(program_date);

    _delay_ms(1000);

    lcd_write_instruction_4d(lcd_DisplayOff);
    _delay_us(DELAY_INST);                                  // 40 uS delay (min)
    _delay_ms(1000);
    lcd_write_instruction_4d(lcd_DisplayOn);

}//lcd_testString

/*============================== 4-bit LCD Functions =====================*/
/*
  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 rather than checking the busy flag
*/
void lcd_init_4d(void)
{
// Power-up delay
// initial 40 mSec delay
    _delay_ms(100);

// 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.
    lcd_RS_port &= ~_BV(lcd_RS_bit);                // select the Instruction
      Register (RS low)
    lcd_E_port &= ~_BV(lcd_E_bit);                  // make sure E is initially low

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

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

    lcd_write_4(lcd_FunctionReset);                 // third part of reset sequence
    _delay_us(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
    _delay_us(DELAY_INST);                                  // 40uS delay (min)

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

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

```
// ; Entry Mode Set instruction
    lcd_write_instruction_4d(lcd_EntryMode);        // set desired shift
      characteristics
    _delay_us(DELAY_INST);                                  // 40uS 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
    _delay_us(DELAY_INST);                                  // 40uS 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++;
        _delay_us(DELAY_INST);                              // 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)
{
    lcd_RS_port |= _BV(lcd_RS_bit);                 // select the Data Register (RS
       high)
    lcd_E_port &= ~_BV(lcd_E_bit);                  // 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)
{
    lcd_RS_port &= ~_BV(lcd_RS_bit);                // select the Instruction
      Register (RS low)
    lcd_E_port &= ~_BV(lcd_E_bit);                  // 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 byte of information to the LCD module
  Entry:    (theByte) is 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:    use either time delays or the busy flag
*/
void lcd_write_4(uint8_t theByte)
{
    lcd_D7_port &= ~_BV(lcd_D7_bit);                        // assume that data is
      '0'
    if (theByte & 1<<7) lcd_D7_port |= _BV(lcd_D7_bit);     // make data = '1' if
      necessary

    lcd_D6_port &= ~_BV(lcd_D6_bit);                        // repeat for each data
       bit
    if (theByte & 1<<6) lcd_D6_port |= _BV(lcd_D6_bit);

    lcd_D5_port &= ~_BV(lcd_D5_bit);
    if (theByte & 1<<5) lcd_D5_port |= _BV(lcd_D5_bit);

    lcd_D4_port &= ~_BV(lcd_D4_bit);
    if (theByte & 1<<4) lcd_D4_port |= _BV(lcd_D4_bit);

// write the data
                                                        // 'Address set-up time' (40
                    nS)
    lcd_E_port |= _BV(lcd_E_bit);                       // Enable pin high
    _delay_us(1);                                       // implement 'Data set-up
      time' (80 nS) and 'Enable pulse width' (230 nS)
    lcd_E_port &= ~_BV(lcd_E_bit);                      // Enable pin low
    _delay_us(1);                                       // implement 'Data hold
      time' (10 nS) and 'Enable cycle time' (500 nS)
}
```