AVR C – 영어단어 학습기(Atmel Studio - ATmega328p English word rotator on LCD)

PN: 23189815

차례

[1 목적 2](#_Toc522004673)

[2 코딩 시나리오 2](#_Toc522004674)

[3 결과 사진 2](#_Toc522004675)

[4 ATmel Studio 7.0 개발 환경 4](#_Toc522004676)

[5 참조 6](#_Toc522004677)

[6 코드 6](#_Toc522004678)

# 목적

Atmel Studio 7.0을 이용해서 아두이노 우노의 MCU인 ATmega328P에 직접 프로그래밍하는 방법을 설명한다. AVR C언어를 이용해서 아트멜 스튜디오에서 작성하고 컴파일된 코드의 hex 실행화일을USBtiny 프로그래머를 이용해서 직접 프로그래밍하는 방법을 설명한다.

아트멜사에서 무료로 제공하는 IDE의 도구에 외부프로그래머(USBtiny)를 연동해서 업로드하는 방법도 설명한다.

# 코딩 시나리오

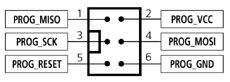
영어 공부를 하면서 만나는 새로운 단어를 평소 공책에 적어두고, 이를 AVR C를 이용해서 2행 16열 문자 LCD 에 주기적으로 나타나게 한다. 코딩하는 재미도 있고, 더불어 심심할 때를 이용해서 영어 단어를 상기시키는 round-robin word memorizer가 탄생한다.

# 결과 사진

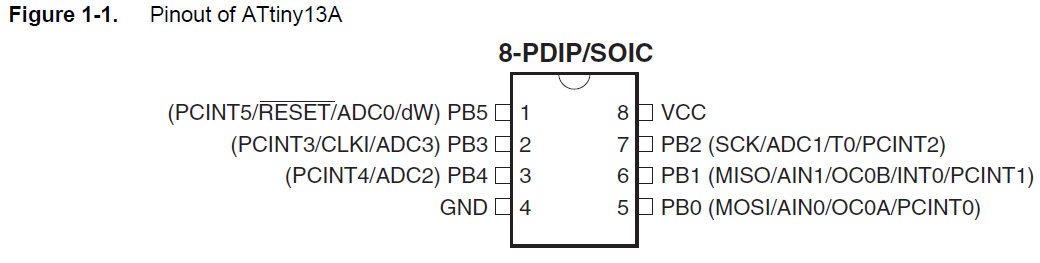
## USBtiny 프로그래머와 연결된 ATmega328p 보드

ATtiny13A 또는 ATtiny 85 MCU를 개발하기 위해 만든, 개발보드의 8핀 칩 소켓을 이용해서 I2C 통신 선로를 연결하였다.

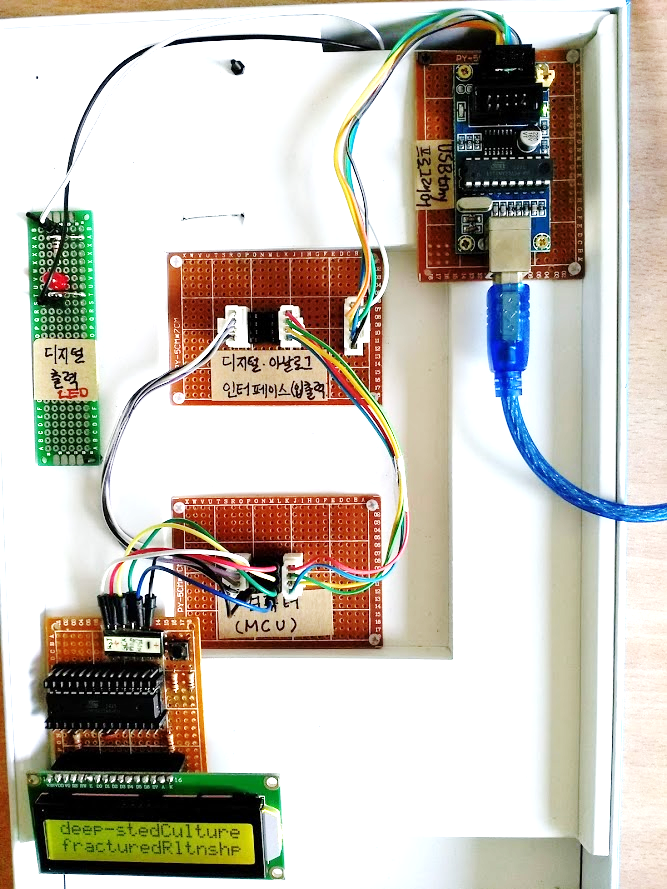
USBtiny 프로그래머의 6핀 소켓 핀아웃 정보



ATtiny13A 8핀 소켓 핀아웃 정보

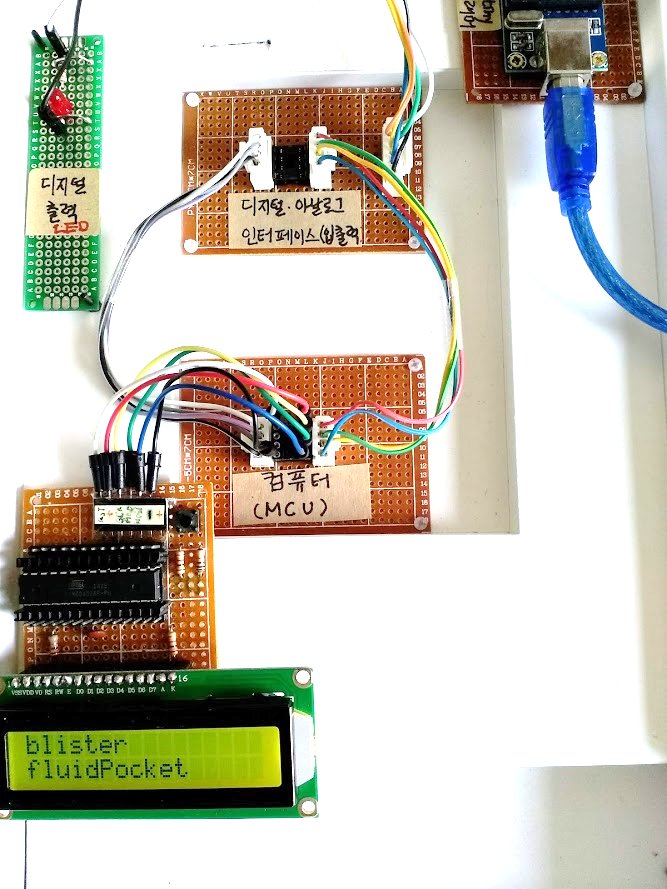


본 실험 사진의 전선은 흰: RST, 빨: Vcc, 노: SCK, 녹: MISO, 파: MOSI, 검: GND

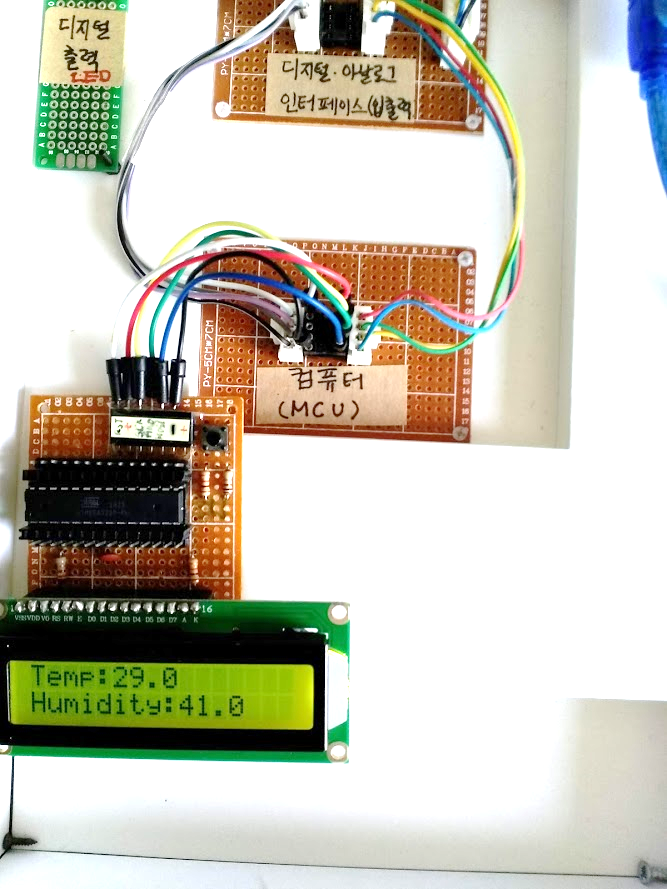


컴퓨터(MCU)보드의 8핀 소켓 위에 라운드 타입 8핀소켓을 덧 씌우고 점퍼선을 꽂으면 안정적으로 점퍼가 고정되고, 소켓 탈착도 편하다.

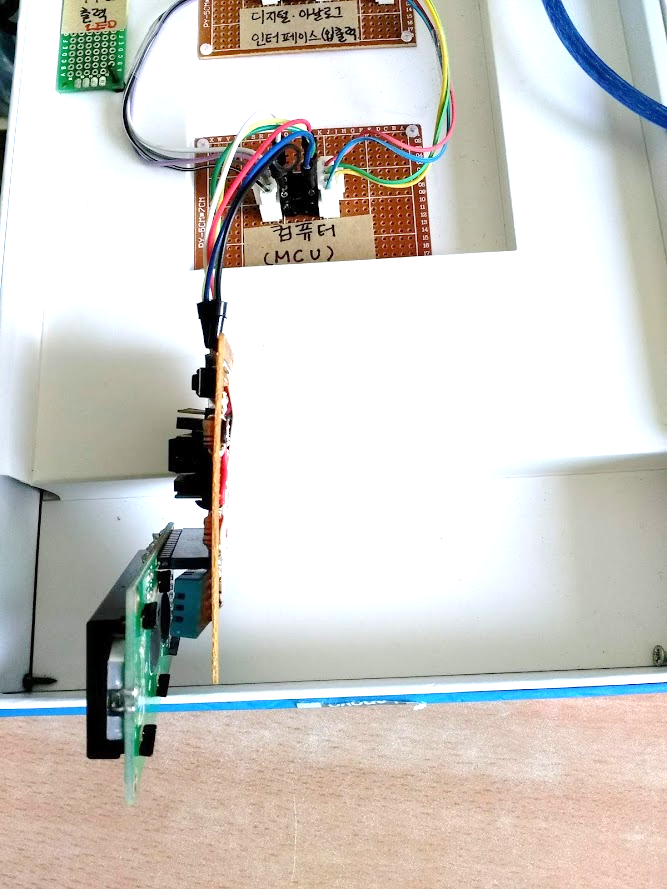
## ATmega328p 보드 확대



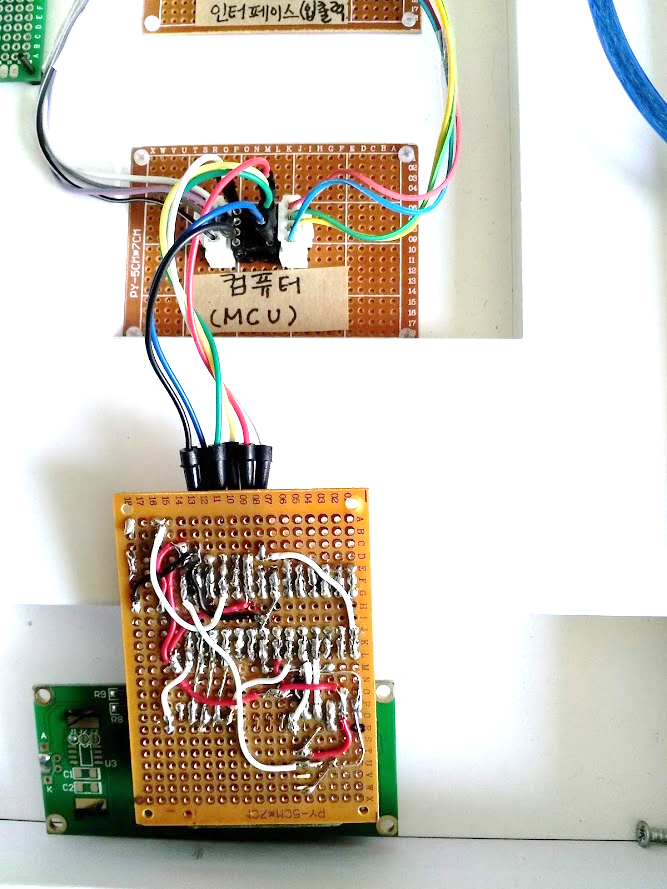
## 온습도 측정



## 측면 (DHT11장착 상태)



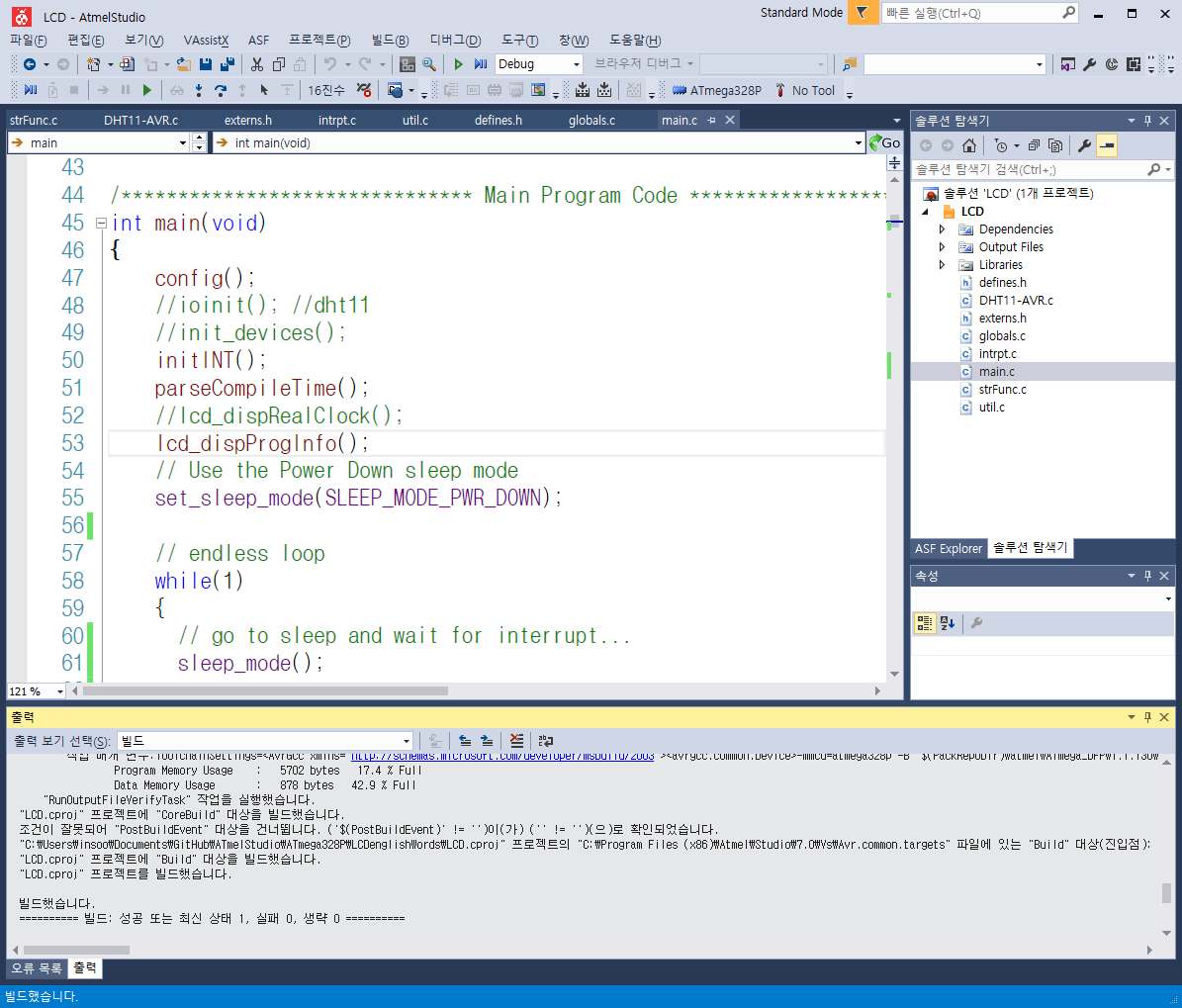
## 뒷면 배선 및 납땜



# ATmel Studio 7.0 개발 환경

## 코딩

F7을 누르면 컴파일



## 업로딩

메뉴 – 도구 – USBtiny : Alt+T, y

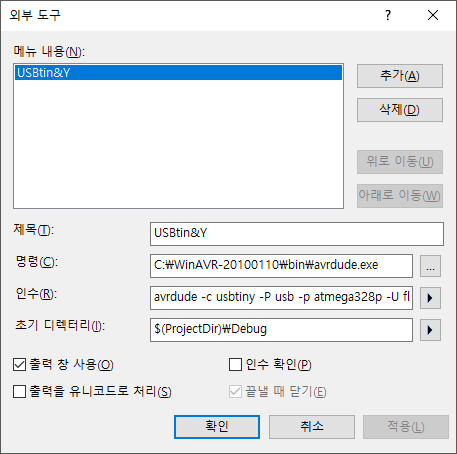
위 단축키는 사전에 도구메뉴의 외부도구 항목에서 설정해 주었기 때문에 가능하다.

제목: USBtin&Y

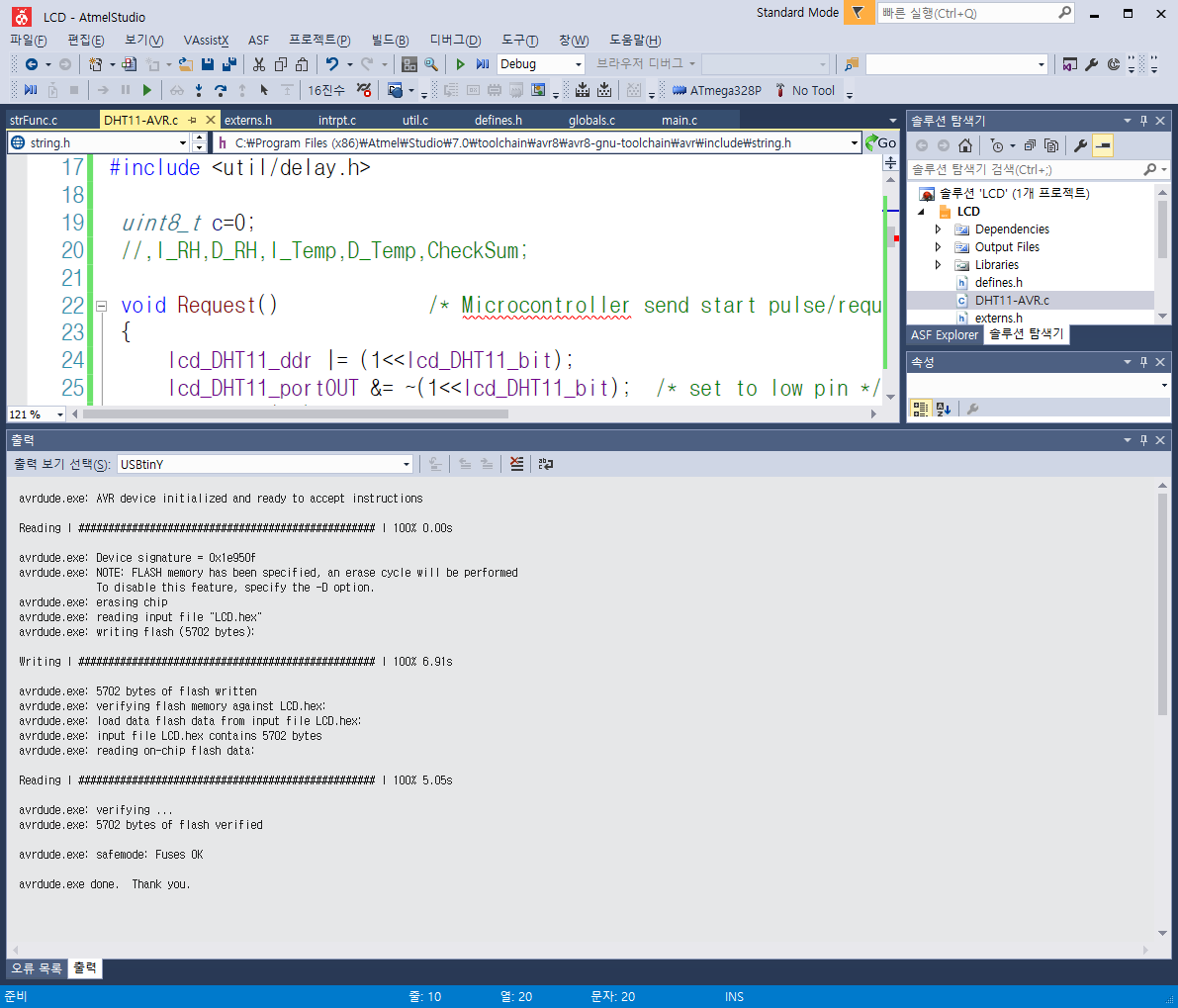
명령: C:\WinAVR-20100110\bin\avrdude.exe

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

초기디렉토리: $(ProjectDir)\Debug



F7으로 컴파일 후 문제가 없음을 확인 한 후, Alt+T, y를 눌러서 USBtiny프로그래머를 통해 ATmega328p로 업로드 하면, Atmel Studio의 출력창에 업로드 진행 경과가 표시된다. 업로드 중에는 USBtiny프로그래머의 빨간 LED가 켜지고, 업로드 완료되면 꺼진다.



# 참조

## DHT11 AVR C코딩

<http://www.electronicwings.com/avr-atmega/dht11-sensor-interfacing-with-atmega16-32>

# 코드

## Main.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

<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"

OR cd "C:\Users\insoo\Box Sync\BoxElec\DIY Electronics\Arduino\Codes 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();

lcd\_dispProgInfo();

// 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

// Ref) http://www.electronicwings.com/avr-atmega/dht11-sensor-interfacing-with-atmega16-32

void lcd\_showDHT11()

{

uint8\_t temperature=0, humidity=0;

char str[3];

uint8\_t humidInt, humidDec, tempInt, tempDec, chkSum;

chkDHT11(&humidInt, &humidDec, &tempInt, &tempDec, &chkSum);

lcd\_write\_instruction\_4d(lcd\_SetCursor | lcd\_LineOne);

\_delay\_us(DELAY\_INST); // 40 uS delay (min)

lcd\_write\_string\_4d((uint8\_t \*)"Temp:");

temperature = tempInt;

itoa(temperature,str, 10);

lcd\_write\_string\_4d((uint8\_t \*)str);

temperature = tempDec;

itoa(temperature,str, 10);

lcd\_write\_string\_4d((uint8\_t \*)".");

lcd\_write\_string\_4d((uint8\_t \*)str);

lcd\_write\_string\_4d((uint8\_t \*)" ");

lcd\_write\_instruction\_4d(lcd\_SetCursor | lcd\_LineTwo);

\_delay\_us(DELAY\_INST); // 40 uS delay (min)

lcd\_write\_string\_4d((uint8\_t \*)"Humidity:");

humidity = humidInt;

itoa(humidity,str, 10);

lcd\_write\_string\_4d((uint8\_t \*)str);

humidity = humidDec;

itoa(humidity,str, 10);

lcd\_write\_string\_4d((uint8\_t \*)".");

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)

}

//--------------------------------------

// TYPEDEFS

typedef uint8\_t byte; // I just like byte & sbyte better

typedef int8\_t sbyte;

// ---------------------------------------------------------------------------

// I2C (TWI) ROUTINES

//

// On the AVRmega series, PA4 is the data line (SDA) and PA5 is the clock (SCL

// The standard clock rate is 100 KHz, and set by I2C\_Init. It depends on the AVR osc. freq.

#define F\_SCL 100000L // I2C clock speed 100 KHz

#define READ 1

#define TW\_START 0xA4 // send start condition (TWINT,TWSTA,TWEN)

#define TW\_STOP 0x94 // send stop condition (TWINT,TWSTO,TWEN)

#define TW\_ACK 0xC4 // return ACK to slave

#define TW\_NACK 0x84 // don't return ACK to slave

#define TW\_SEND 0x84 // send data (TWINT,TWEN)

#define TW\_READY (TWCR & 0x80) // ready when TWINT returns to logic 1.

#define TW\_STATUS (TWSR & 0xF8) // returns value of status register

#define I2C\_Stop() TWCR = TW\_STOP // inline macro for stop condition

void I2C\_Init()

// at 16 MHz, the SCL frequency will be 16/(16+2(TWBR)), assuming prescalar of 0.

// so for 100KHz SCL, TWBR = ((F\_CPU/F\_SCL)-16)/2 = ((16/0.1)-16)/2 = 144/2 = 72.

{

TWSR = 0; // set prescalar to zero

TWBR = ((F\_CPU/F\_SCL)-16)/2; // set SCL frequency in TWI bit register

}

byte I2C\_Detect(byte addr)

// look for device at specified address; return 1=found, 0=not found

{

TWCR = TW\_START; // send start condition

while (!TW\_READY); // wait

TWDR = addr; // load device's bus address

TWCR = TW\_SEND; // and send it

while (!TW\_READY); // wait

return (TW\_STATUS==0x18); // return 1 if found; 0 otherwise

}

byte I2C\_FindDevice(byte start)

// returns with address of first device found; 0=not found

{

for (byte addr=start;addr<0xFF;addr++) // search all 256 addresses

{

if (I2C\_Detect(addr)) // I2C detected?

return addr; // leave as soon as one is found

}

return 0; // none detected, so return 0.

}

void I2C\_Start (byte slaveAddr)

{

I2C\_Detect(slaveAddr);

}

byte I2C\_Write (byte data) // sends a data byte to slave

{

TWDR = data; // load data to be sent

TWCR = TW\_SEND; // and send it

while (!TW\_READY); // wait

return (TW\_STATUS!=0x28);

}

byte I2C\_ReadACK () // reads a data byte from slave

{

TWCR = TW\_ACK; // ack = will read more data

while (!TW\_READY); // wait

return TWDR;

//return (TW\_STATUS!=0x28);

}

byte I2C\_ReadNACK () // reads a data byte from slave

{

TWCR = TW\_NACK; // nack = not reading more data

while (!TW\_READY); // wait

return TWDR;

//return (TW\_STATUS!=0x28);

}

void I2C\_WriteByte(byte busAddr, byte data)

{

I2C\_Start(busAddr); // send bus address

I2C\_Write(data); // then send the data byte

I2C\_Stop();

}

void I2C\_WriteRegister(byte busAddr, byte deviceRegister, byte data)

{

I2C\_Start(busAddr); // send bus address

I2C\_Write(deviceRegister); // first byte = device register address

I2C\_Write(data); // second byte = data for device register

I2C\_Stop();

}

byte I2C\_ReadRegister(byte busAddr, byte deviceRegister)

{

byte data = 0;

I2C\_Start(busAddr); // send device address

I2C\_Write(deviceRegister); // set register pointer

I2C\_Start(busAddr+READ); // restart as a read operation

data = I2C\_ReadNACK(); // read the register data

I2C\_Stop(); // stop

return data;

}

// ---------------------------------------------------------------------------

// DS1307 RTC ROUTINES

#define DS1307 0xD0 // I2C bus address of DS1307 RTC

#define SECONDS\_REGISTER 0x00

#define MINUTES\_REGISTER 0x01

#define HOURS\_REGISTER 0x02

#define DAYOFWK\_REGISTER 0x03

#define DAYS\_REGISTER 0x04

#define MONTHS\_REGISTER 0x05

#define YEARS\_REGISTER 0x06

#define CONTROL\_REGISTER 0x07

#define RAM\_BEGIN 0x08

#define RAM\_END 0x3F

void DS1307\_GetTime(byte \*hours, byte \*minutes, byte \*seconds)

// returns hours, minutes, and seconds in BCD format

{

\*hours = I2C\_ReadRegister(DS1307,HOURS\_REGISTER);

\*minutes = I2C\_ReadRegister(DS1307,MINUTES\_REGISTER);

\*seconds = I2C\_ReadRegister(DS1307,SECONDS\_REGISTER);

if (\*hours & 0x40) // 12hr mode:

\*hours &= 0x1F; // use bottom 5 bits (pm bit = temp & 0x20)

else \*hours &= 0x3F; // 24hr mode: use bottom 6 bits

}

void DS1307\_GetDate(byte \*months, byte \*days, byte \*years)

// returns months, days, and years in BCD format

{

\*months = I2C\_ReadRegister(DS1307,MONTHS\_REGISTER);

\*days = I2C\_ReadRegister(DS1307,DAYS\_REGISTER);

\*years = I2C\_ReadRegister(DS1307,YEARS\_REGISTER);

}

void SetTimeDate()

// simple, hard-coded way to set the date.

{

I2C\_WriteRegister(DS1307,MONTHS\_REGISTER, 0x08);

I2C\_WriteRegister(DS1307,DAYS\_REGISTER, 0x31);

I2C\_WriteRegister(DS1307,YEARS\_REGISTER, 0x13);

I2C\_WriteRegister(DS1307,HOURS\_REGISTER, 0x08+0x40); // add 0x40 for PM

I2C\_WriteRegister(DS1307,MINUTES\_REGISTER, 0x51);

I2C\_WriteRegister(DS1307,SECONDS\_REGISTER, 0x00);

}

## DHT11\_AVR.c

/\*

\* DHT11\_AVR.c

\*

\* Created: 2018-08-13 오후 8:46:46

\* Author: insoo

\* Ref: http://www.electronicwings.com/avr-atmega/dht11-sensor-interfacing-with-atmega16-32

\*/

#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>

uint8\_t c=0;

//,I\_RH,D\_RH,I\_Temp,D\_Temp,CheckSum;

void Request() /\* Microcontroller send start pulse/request \*/

{

lcd\_DHT11\_ddr |= (1<<lcd\_DHT11\_bit);

lcd\_DHT11\_portOUT &= ~(1<<lcd\_DHT11\_bit); /\* set to low pin \*/

\_delay\_ms(20); /\* wait for 20ms \*/

lcd\_DHT11\_portOUT |= (1<<lcd\_DHT11\_bit); /\* set to high pin \*/

}

void Response() /\* receive response from DHT11 \*/

{

lcd\_DHT11\_ddr &= ~(1<<lcd\_DHT11\_bit);

while(lcd\_DHT11\_portIN & (1<<lcd\_DHT11\_bit));

while((lcd\_DHT11\_portIN & (1<<lcd\_DHT11\_bit))==0);

while(lcd\_DHT11\_portIN & (1<<lcd\_DHT11\_bit));

}

uint8\_t Receive\_data() /\* receive data \*/

{

for (int q=0; q<8; q++)

{

while((lcd\_DHT11\_portIN & (1<<lcd\_DHT11\_bit)) == 0); /\* check received bit 0 or 1 \*/

\_delay\_us(30);

if(lcd\_DHT11\_portIN & (1<<lcd\_DHT11\_bit))/\* if high pulse is greater than 30ms \*/

c = (c<<1)|(0x01); /\* then its logic HIGH \*/

else /\* otherwise its logic LOW \*/

c = (c<<1);

while(lcd\_DHT11\_portIN & (1<<lcd\_DHT11\_bit));

}

return c;

}

int chkDHT11(uint8\_t \*humidInt, uint8\_t \*humidDec,uint8\_t \*tempInt, uint8\_t \*tempDec, uint8\_t \*chkSum)

{

//char data[5];

Request(); /\* send start pulse \*/

Response(); /\* receive response \*/

\*humidInt=Receive\_data(); /\* store first eight bit in I\_RH \*/

\*humidDec=Receive\_data(); /\* store next eight bit in D\_RH \*/

\*tempInt=Receive\_data(); /\* store next eight bit in I\_Temp \*/

\*tempDec=Receive\_data(); /\* store next eight bit in D\_Temp \*/

\*chkSum=Receive\_data(); /\* store next eight bit in CheckSum \*/

if ((\*humidInt + \*humidDec + \*tempInt + \*tempDec) != \*chkSum)

{

return (-1);

}

else

{

//itoa(I\_RH,data,10);

return (0);

}

\_delay\_ms(10);

}

## globals.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Target MCU & clock speed: ATmega328P @ 1Mhz internal

Name : globals.c

Author : Insoo Kim (insoo@hotmail.com)

Created : May 15, 2015

Updated : May 17, 2015

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]:

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/pgmspace.h>

#include "defines.h"

uint8\_t wd=0;

// New words

uint8\_t words000[10][2][16] =

{

{//0 "1234512345123456" "1234512345123456"

//"petrified", "cause2bcmStnlike"

"succumb", "fail to resist"

},

{//1 "1234512345123456" "1234512345123456"

"arid", "lckgSuffWtr/Rain"

},

{//2 "1234512345123456" "1234512345123456"

"enliven", "hgtn/intensify"

},

{//3"1234512345123456" "1234512345123456"

"lurch", "mv abruptly"

},

{//4 "1234512345123456" "1234512345123456"

"oblivious to", "lckgCnscsAwrnOf"

},

{//5 "1234512345123456" "1234512345123456"

"wtn a whisker of", "extremelyClose 2"

},

{//6 "1234512345123456" "1234512345123456"

"colloid", "mixWpropBtSolSus"

},

{//7 "1234512345123456" "1234512345123456"

"reticent", "disclined2talk"

},

{//8 "1234512345123456" "1234512345123456"

"pry~apart", "force2getSthOpen"

},

{//9 "1234512345123456" "1234512345123456"

"esoteric", "cnfnd2onyEnlgtnd"

},

}; //words000[10][2][16]

uint8\_t words001[10][2][16] =

{

{//0 "1234512345123456" "1234512345123456"

"snag", "hiddenObstacle"

},

{//1 "1234512345123456" "1234512345123456"

"mission-agnostic", "genApplicable"

},

{//2 "1234512345123456" "1234512345123456"

"deep-stedCulture","fracturedRltnshp"

},

{//3"1234512345123456" "1234512345123456"

"tumult", "loudCnfsdNseByMs"

},

{//4 "1234512345123456" "1234512345123456"

"sobering", "tending2mkSober"

},

{//5 "1234512345123456" "1234512345123456"

"miscarrage", "brthBbyAlrdDd@Wm"

},

{//6 "1234512345123456" "1234512345123456"

"class-actnLwSt", "lawSuitTogether"

},

{//7 "1234512345123456" "1234512345123456"

"blister", "fluidPocket"

},

{//8 "1234512345123456" "1234512345123456"

"titter", "laughQuietNrvsly"

},

{//9 "1234512345123456" "1234512345123456"

"line cook", "a cook inKitchen"

},

}; //words001[10][2][16]

uint8\_t prevMenuCnt;

int8\_t hour=0, min=0, sec=0;

int8\_t year=0, month=0, date=0;

int8\_t monthEndDate, day=0;

int8\_t accumulatedHour=0, accumulatedMin=0,accumulatedSec=0;

//uint8\_t hourlyAdjusted=0;

// Program ID

uint8\_t program\_author[] = "Insoo Kim ";

uint8\_t program\_version[] = "LCD-ATmega328p";

//uint8\_t program\_date[] = "May 23, 2015 "; // Initial Creation Date

//uint8\_t program\_date[] = "Mar 22, 2017 "; // Last Update

//uint8\_t program\_date[] = "Jul 11, 2017 "; // Last Update

uint8\_t program\_date[] = "Aug 13, 2018 "; // Last Update

// Menu info

uint8\_t menu\_str1[] = "1:Rdy Or PrevSel";

uint8\_t menu\_str2[] = "2CK3TH4BL5AJ6AT";

## intrpt.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Target MCU & clock speed: ATmega328P @ 1Mhz internal

Name : intrpt.c

Author : Insoo Kim (insoo@hotmail.com)

Created : May 15, 2015

Updated : May 17, 2015

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]:

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>

/\*

//Function prototypes of outside this code module

extern uint8\_t hour, min, sec;

extern uint8\_t year, month, date;

extern uint8\_t monthEndDate, day;

extern void lcd\_dispRealClock();

extern void lcd\_dispAccumulatedTime();

extern void lcd\_dispProgInfo();

\*/

//Function prototypes of this code module

//void proceedClock();

ISR(PCINT0\_vect)

{

//if (PINB & \_BV(PB6))

// read switch status

if (tactile\_Switch\_port & \_BV(tactile\_Switch\_bit))

{

//Disable PC(Pin Change) interrupt

// to use the button for user menu selection

//For PCINT7-0, DS: Ch 12.2.8

//PCMSK0 &= ~\_BV(PCINT6);

PCMSK0 &= ~\_BV(tactile\_Switch\_bit);

countButton();

//Enable PC(Pin Change) interrupt

//For PCINT7-0, DS: Ch 12.2.8

//PCMSK0 |= \_BV(PCINT6);

PCMSK0 |= \_BV(tactile\_Switch\_bit);

}

\_delay\_ms(200); // for debounce

}//ISR(PCINT0\_vect)

//-----------------------------------

ISR(WDT\_vect)

{

//PORTB |= \_BV(PB4);

proceedClock();

lcd\_dispWords(wd%MAXWORDCNT);

wd++;

//lcd\_dispRealClock();

//PORTB &= ~\_BV(PB4);

}//ISR(WDT\_vect)

//-----------------------------------

void WDT\_Init(void)

{

//disable interrupts

cli();

//MCUSR = 0;

wdt\_disable();

//set up WDT interrupt

//WDTCSR = \_BV(WDCE)|\_BV(WDE);

//Start watchdog timer with 4s prescaller

//WDTCSR |= \_BV(WDIE)|\_BV(WDE)|\_BV(WDP3);

//WDTCSR |= \_BV(WDIE)| \_BV(WDP3);

//WDTCSR |= \_BV(WDIE)|\_BV(WDE)|\_BV(WDP2) | \_BV(WDP1); // 1s

//reset watchdog

wdt\_reset();

wdt\_enable(WDTO\_500MS);

//Enable global interrupts

sei();

}//WDT\_Init

void initINT()

{

cli();

//\*\*\*\* PC(Pin Change) interrupt setting

//ref) https://gist.github.com/Wollw/2598827

// enable PC INT

//GIMSK |= \_BV(PCIE); //Enable PC interrupt

// Enable pin change interrupt for PB3

//PCMSK |= \_BV(PCINT3);

//PCMSK |= \_BV(startPin);

/\*\*

\* Pin Change Interrupt enable on PCINT6 (PB6)

\*/

//For PCINT7-0, DS: Ch 12.2.4

PCICR |= \_BV(PCIE0);

//For PCINT7-0, DS: Ch 12.2.8

//PCMSK0 |= \_BV(PCINT4);

PCMSK0 |= \_BV(tactile\_Switch\_bit);

//\*\*\*\* WDT interrupt setting

check\_wdt();

setup\_wdt();

//set prescale timer

//DS: ch10.9.2 table10.2

//set up WDT interrupt

//WDTCSR |= \_BV(WDCE)| \_BV(WDE);

//WDTCSR = \_BV(WDIE) | \_BV(WDP3) | \_BV(WDP0); // 8s

//WDTCSR = \_BV(WDIE) | \_BV(WDP3); // 4s

//WDTCSR = \_BV(WDIE) | \_BV(WDP2) | \_BV(WDP0); // 0.5s

//wdt\_reset();

//enalbe global interrupt

sei();

}//initINT

/\*

Utmost(!) help to get the WDT of ATmega328p work

http://elegantcircuits.com/2014/10/14/introduction-to-the-avr-watchdog-timer/

\*/

void check\_wdt(void)

{

// If a reset was caused by the Watchdog Timer...

if(MCUSR & \_BV(WDRF))

{

// Clear the WDT reset flag

MCUSR &= ~\_BV(WDRF);

// Enable the WD Change Bit

WDTCSR |= (\_BV(WDCE) | \_BV(WDE));

// Disable the WDT

WDTCSR = 0x00;

}

}//check\_wdt

void setup\_wdt(void){

// Set up Watch Dog Timer for Inactivity

// Enable the WD Change Bit

// Enable WDT interrupt

WDTCSR |= \_BV(WDCE) | \_BV(WDE);

// Set Timeout to ~8 seconds

WDTCSR = \_BV(WDIE) | \_BV(WDP3) | \_BV(WDP0); // 8s

// Set Timeout to ~1 seconds

//WDTCSR = \_BV(WDIE) | \_BV(WDP2) | \_BV(WDP1);

// Set Timeout to ~500 ms

//WDTCSR = \_BV(WDIE) | \_BV(WDP2);

}//setup\_wdt

void init\_devices(void){

//stop errant interrupts until set up

cli(); //disable all interrupts

//timer0\_init();

MCUCR = 0x00;

EICRA = 0x00; //extended ext ints

EIMSK = 0x00;

TIMSK0 = 0x02; //timer 0 interrupt sources

PRR = 0x00; //power controller

sei(); //re-enable interrupts

//all peripherals are now initialized

}

void proceedClock()

{

//WDT interrupt occurs every 8 seconds

sec += 8;

accumulatedSec += 8;

//real-time fetched from compiled time constant \_\_TIME\_\_

//calculate minutes

if (sec >= 60)

{

sec%=60;

min++;

sec += 2;

}

//system run-time acculated time

//calculate minutes

if (accumulatedSec >= 60)

{

accumulatedSec%=60;

accumulatedMin++;

accumulatedSec += 2;

}

//real-time fetched from compiled time constant \_\_TIME\_\_

//calculate hours

if (min >= 60)

{

min%=60;

hour++;

}

//system run-time acculated time

//calculate hours

//if ((accumulatedMin == 60) && (hourlyAdjusted == 0))

if (accumulatedMin == 60)

{

// adjust time by experiments

//hourlyAdjusted = 1;

sec -= 20;

accumulatedMin = 0;

accumulatedHour++;

}

/\*

if (accumulatedMin == 59)

hourlyAdjusted = 0;

\*/

if (hour >= 24)

{

hour=0;

date++;

day++;

if (day >= 7)

day %= 7;

}

switch (month)

{

case 1:

case 3:

case 5:

case 7:

case 8:

case 10:

case 12:

monthEndDate = 31;

break;

case 2:

monthEndDate = 28;

break;

default:

monthEndDate = 30;

}//switch (month)

if (date > monthEndDate)

{

date=1;

month++;

}

if (month > 12)

{

month=1;

year++;

}

}//proceedClock

## strFunc.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Target MCU & clock speed: ATmega328P @ 1Mhz internal

Name : strFunc.c

Author : Insoo Kim (insoo@hotmail.com)

Created : May 15, 2015

Updated : May 17, 2015

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]:

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>

/\*

extern uint8\_t hour, min, sec;

extern uint8\_t year, month, date;

extern uint8\_t monthEndDate, day;

\*/

void parseCompileTime()

{

char \*p;

char sTime[4][3];

char sDate[4][10];

uint8\_t n=0;

uint16\_t yearLong;

p = strtok(\_\_TIME\_\_,":");

while (\*p)

{

strcpy((char \*)&sTime[n++], p);

p=strtok(NULL, ":");

}

hour=atoi(sTime[0]);

min=atoi(sTime[1]);

//give some delay (8 seconds), to compile & upload by human click

//sec=atoi(sTime[2]) + 8;

//if you change power source to battery,

// you'd better put more time allowance

sec=atoi(sTime[2]) + 15;

//hour = atoi(sHour);

n=0;

p = strtok(\_\_DATE\_\_," ");

while (\*p)

{

strcpy((char \*)&sDate[n++], p);

p=strtok(NULL, " ");

}

if ( strcmp(sDate[0], "Jan") == 0 )

month = 1;

else if ( strcmp(sDate[0], "Feb") == 0 )

month = 2;

else if ( strcmp(sDate[0], "Mar") == 0 )

month = 3;

else if ( strcmp(sDate[0], "Apr") == 0 )

month = 4;

else if ( strcmp(sDate[0], "May") == 0 )

month = 5;

else if ( strcmp(sDate[0], "Jun") == 0 )

month = 6;

else if ( strcmp(sDate[0], "Jul") == 0 )

month = 7;

else if ( strcmp(sDate[0], "Aug") == 0 )

month = 8;

else if ( strcmp(sDate[0], "Sep") == 0 )

month = 9;

else if ( strcmp(sDate[0], "Oct") == 0 )

month = 10;

else if ( strcmp(sDate[0], "Nov") == 0 )

month = 11;

else if ( strcmp(sDate[0], "Dec") == 0 )

month = 12;

date=atoi(sDate[1]);

yearLong=atoi(sDate[2]);

year=yearLong%1000;

calcDay();

}//parseCompileTime

void calcDay()

{

day = (date - 17)%7;

}

## util.c

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Target MCU & clock speed: ATmega328P @ 1Mhz internal

Name : utils.c

Author : Insoo Kim (insoo@hotmail.com)

Created : May 17, 2015

Updated : May 17, 2015

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]:

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>

//-----------------------------------

void sysClockTest()

{

PORTB |= \_BV(debug\_PIN);

\_delay\_ms(10);

PORTB &= ~\_BV(debug\_PIN);

\_delay\_ms(10);

}//sysClockTest

//-----------------------------------

void countButton()

{

uint8\_t menuCnt=0;

uint8\_t prevLoop=0, curLoop=0, lapse=0;

uint8\_t loopCnt=0;

uint8\_t val;

uint8\_t DONE=0;

loopCnt=0;

prevLoop=0;

//visual cue to show being ready to get user input of menuCnt

//blinkLED(1);

//Get menuCnt by counting the button press

//If pressing the button within 1 second of interval between each press,

// it will be accumulated as "menuCnt".

//If the interval is over 1 sec, which is menuSelectInterval,

// then DONE is set to 1.

while (!DONE)

{

loopCnt++;

curLoop = loopCnt;

lapse = curLoop - prevLoop;

//menuSelectInterval is

// multiple of \_delay\_ms(halfSec/4)

// which is the unit delay of each while loop

if (lapse > menuSelectInterval)

{

if (menuCnt != 0)

DONE = 1;

else

//wait another 1 sec to get user's menuCnt

;

}

//if user has not chosen menuCnt > 0, and time lapse over 3sec,

//forget about it, and call DONE as 2, and eventually go to sleep.

if ((lapse > menuSelectInterval\*2) && (menuCnt ==0))

DONE = 2;

//check if tactile\_Switch pressed to 0

//AVR equivalent to Arduino digitalRead(tactile\_Switch\_bit)

val = tactile\_Switch\_port & \_BV(tactile\_Switch\_bit);

if (val == 0)

{

// for debounce

\_delay\_ms(halfSec/2);

menuCnt++;

//Pressing the button, lap time calculation should be reset

//to give 1 sec of time to choose menuCnt

prevLoop = loopCnt;

}//if(val == 0)

// should be fast enough to catch button press frequency

// menuSelectInterval is a multiple of times of \_delay\_ms(halfSec/4);

\_delay\_ms(halfSec/4);

}//while(!DONE)

//menuCnt has been set within 3sec of a PCINT occurence

// then, play WDT count for a corresponding alarm period.

if (DONE)

{

if (DONE == 2)

menuCnt = prevMenuCnt;

//visual cue to notifiy user selected menuCnt

//blinkLED(menuCnt);

//\_delay\_ms(halfSec);

//turnOnLCDpower();

//lcd\_dispON();

prevMenuCnt = menuCnt;

switch (menuCnt)

{

case 1:

//lcd\_dispWords();

lcd\_dispRealClock();

\_delay\_ms(1000);

break;

case 2:

\_delay\_ms(1000);

turnOnLCDBacklight();

lcd\_showDHT11();

\_delay\_ms(5000);

turnOffLCDBacklight();

break;

case 3:

turnOnLCDBacklight();

lcd\_dispRealClock();

\_delay\_ms(7000);

turnOffLCDBacklight();

break;

case 4:

//adjustClock();

adjustMin();

adjustHour();

adjustSec();

lcd\_dispRealClock();

\_delay\_ms(1000);

break;

case 5:

lcd\_dispAccumulatedTime();

\_delay\_ms(2000);

break;

case 6:

lcd\_dispProgInfo();

break;

default:

lcd\_dispMenu();

}//switch (menuCnt)

//lcd\_dispOFF();

//turnOffLCDpower();

//reset menuCnt

menuCnt=0;

//Enable watchdog timer interrupts

// and begin counting for alarm

//WDTCR |= \_BV(WDTIE);

}//if (DONE == 1)

}//countButton

void adjustHour()

{

uint8\_t DONE=0, val;

uint8\_t curLoop=0, preLoop=0, lapse=0;

char strHour[3];

while (!DONE)

{

// 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((uint8\_t \*)"Hour: ");

itoa(hour, strHour, 10);

lcd\_write\_string\_4d((uint8\_t \*)strHour);

lcd\_write\_string\_4d((uint8\_t \*)" ");

val = tactile\_Switch\_port & \_BV(tactile\_Switch\_bit);

\_delay\_ms(100);

if (val == 0)

{

hour++;

if (hour > 23)

hour=0;

preLoop = curLoop;

}

curLoop++;

lapse = curLoop - preLoop;

if (lapse > adjustTimeInterval)

DONE = 1;

\_delay\_ms(halfSec/4);

}//while (!DONE)

}//adjustHour

void adjustMin()

{

uint8\_t DONE=0, val;

uint8\_t curLoop=0, preLoop=0, lapse=0;

char strMin[3];

while (!DONE)

{

// 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((uint8\_t \*)"Min: ");

itoa(min, strMin, 10);

lcd\_write\_string\_4d((uint8\_t \*)strMin);

lcd\_write\_string\_4d((uint8\_t \*)" ");

val = tactile\_Switch\_port & \_BV(tactile\_Switch\_bit);

\_delay\_ms(100);

if (val == 0)

{

min++;

if (min > 59)

min=0;

preLoop = curLoop;

}

curLoop++;

lapse = curLoop - preLoop;

if (lapse > adjustTimeInterval)

DONE = 1;

\_delay\_ms(halfSec/4);

}//while (!DONE)

}//adjustMin

void adjustSec()

{

uint8\_t DONE=0, val;

uint8\_t curLoop=0, preLoop=0, lapse=0;

char strSec[3];

while (!DONE)

{

// set cursor to start of first line

lcd\_write\_instruction\_4d(lcd\_SetCursor | lcd\_LineOne);

\_delay\_us(DELAY\_INST); // 40 uS delay (sec)

// display the first line of information

lcd\_write\_string\_4d((uint8\_t \*)"sec: ");

itoa(sec, strSec, 10);

lcd\_write\_string\_4d((uint8\_t \*)strSec);

lcd\_write\_string\_4d((uint8\_t \*)" ");

val = tactile\_Switch\_port & \_BV(tactile\_Switch\_bit);

\_delay\_ms(100);

if (val == 0)

{

sec++;

if (sec > 59)

sec=0;

preLoop = curLoop;

}

curLoop++;

lapse = curLoop - preLoop;

if (lapse > adjustTimeInterval)

DONE = 1;

\_delay\_ms(halfSec/4);

}//while (!DONE)

}//adjustSec

## defines.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Target MCU & clock speed: ATmega328P @ 1Mhz internal

Name : defines.h

Author : Insoo Kim (insoo@hotmail.com)

Created : May 15, 2015

Updated : May 16, 2015

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]:

Ref:

Donald Weiman (weimandn@alfredstate.edu)

http://web.alfredstate.edu/weimandn/programming/lcd/ATmega328/LCD\_code\_gcc\_4d.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//Reference notes from the author that i referd at the begining

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

LCD-AVR-4d.c - 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.c

Date: September 16, 2013

Target: ATmega328

Compiler: avr-gcc (AVR Studio 6)

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 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 |

| | | |

| PD3|---------------->|E |

| | GND --->|RW |

| PD2|---------------->|RS |

----------- ----------

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//DS: Ch.8.2.1 default clock source is 1Mhz

#define F\_CPU 1000000L

#define MAXWORDCNT 10

#define LCD\_MAXCOL 16

#define debug\_PIN PORTB5

#define menuSelectInterval 4

#define adjustTimeInterval 8

#define halfSec 500 // 0.5 second checked by oscilloscope

#define DELAY\_INST 40

// LCD interface (should agree with the diagram above)

// make sure that the LCD RW pin is connected to GND

#define lcd\_D7\_port PORTD // lcd D7 connection

#define lcd\_D7\_bit PORTD7

#define lcd\_D7\_ddr DDRD

#define lcd\_D6\_port PORTD // lcd D6 connection

#define lcd\_D6\_bit PORTD6

#define lcd\_D6\_ddr DDRD

#define lcd\_D5\_port PORTD // lcd D5 connection

#define lcd\_D5\_bit PORTD5

#define lcd\_D5\_ddr DDRD

#define lcd\_D4\_port PORTD // lcd D4 connection

#define lcd\_D4\_bit PORTD4

#define lcd\_D4\_ddr DDRD

#define lcd\_E\_port PORTD // lcd Enable pin

#define lcd\_E\_bit PORTD3

#define lcd\_E\_ddr DDRD

#define lcd\_RS\_port PORTD // lcd Register Select pin

#define lcd\_RS\_bit PORTD2

#define lcd\_RS\_ddr DDRD

#define lcd\_Backlight\_port PORTB

#define lcd\_Backlight\_bit PORTB0

#define lcd\_Backlight\_ddr DDRB

//added Aug 13, 2018 (Begin)

#define lcd\_DHT11\_portOUT PORTB

#define lcd\_DHT11\_portIN PINB

#define lcd\_DHT11\_bit PORTB1

#define lcd\_DHT11\_ddr DDRB

//added Aug 13, 2018 (End)

#define tactile\_Switch\_port PINB

#define tactile\_Switch\_bit PORTB4

#define tactile\_Switch\_ddr DDRB

// 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

## externs.h

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Target MCU & clock speed: ATmega328P @ 1Mhz internal

Name : externs.h

Author : Insoo Kim (insoo@hotmail.com)

Created : May 15, 2015

Updated : May 17, 2015

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]:

Ref:

Donald Weiman (weimandn@alfredstate.edu)

http://web.alfredstate.edu/weimandn/programming/lcd/ATmega328/LCD\_code\_gcc\_4d.html

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

extern uint8\_t wd;

extern uint8\_t words000[10][2][16];

extern uint8\_t words001[10][2][16];

extern uint8\_t prevMenuCnt;

extern uint8\_t program\_author[];

extern uint8\_t program\_version[];

extern uint8\_t program\_date[];

extern uint8\_t menu\_str1[];

extern uint8\_t menu\_str2[];

extern int8\_t dht\_getdata(int8\_t \*, int8\_t \*);

extern int8\_t dht\_getdata\_EXT16MHZ(int8\_t \*, int8\_t \*);

extern int8\_t dht\_getdata\_INT8MHZ(int8\_t \*, int8\_t \*);

extern int8\_t dht\_gettemperaturehumidity(int8\_t \*, int8\_t \*);

extern void ioinit (void);

extern int8\_t hour, min, sec;

extern int8\_t year, month, date;

extern int8\_t monthEndDate, day;

extern int8\_t accumulatedHour, accumulatedMin, accumulatedSec;

//extern uint8\_t hourlyAdjusted;

extern void proceedClock();

extern void WDT\_Init(void);

extern void initINT(void);

extern void check\_wdt(void);

extern void setup\_wdt(void);

extern void init\_devices(void);

extern void parseCompileTime(void);

extern void calcDay(void);

// Function Prototypes

extern void chkButtonAndToggleBacklight();

extern void config();

extern void turnOnLCDBacklight();

extern void turnOffLCDBacklight();

extern void lcd\_SysTime();

extern void lcd\_dispON();

extern void lcd\_dispOFF();

extern void lcd\_dispRealClock();

extern void lcd\_dispAccumulatedTime();

extern void lcd\_showDHT11();

extern void lcd\_testString();

extern void lcd\_dispProgInfo();

extern void lcd\_dispWords(uint8\_t );

extern void lcd\_dispMenu();

extern void lcd\_write\_4(uint8\_t);

extern void lcd\_write\_instruction\_4d(uint8\_t);

extern void lcd\_write\_character\_4d(uint8\_t);

extern void lcd\_write\_string\_4d(uint8\_t \*);

extern void lcd\_init\_4d(void);

//UTIL.C

extern void sysClockTest();

extern void countButton(void);

extern void adjustHour();

extern void adjustMin();

extern void adjustSec();

//DHT11\_AVR.c

extern int chkDHT11(uint8\_t\*, uint8\_t\*, uint8\_t\*, uint8\_t\*, uint8\_t\*);