Ronobir Mookherjee

**Design Assignment 5/Midterm Part 1**

Description:

Use a temperature sensor to measure the ambient room temperature and display on the computer using serial transmission and interface with uC using ADC.

**Code:**

*/\**

*\* DesignAssigment5Part1.c*

*\**

*\* Submission for part 1*

*Part 1 and 2 were combined into one project, and are being separated and submitted individually.*

*The purpose of part 1 is to get the value of the Temperature using ADC once per second and outputting through USART connection to a terminal in the window.*

*The three components used include the Temperature sensor LM34 from TI, the Polulu Programmer with TTL serial communications capability and the Atmega328p*

*\* Author: Ronobir Mookherjee*

*\*/*

*/\* Below is the standard setup for an AVR project, with preprocessor directives\*/*

#include <stdlib.h>

#include <alloca.h>

#include <assert.h>

#include <ctype.h>

#include <errno.h>

#include <inttypes.h>

#include <math.h>

#include <setjmp.h>

#include <stdint.h>

#include <stdio.h>

#include <string.h>

#include <avr/boot.h>

#include <avr/io.h>

#include <avr/cpufunc.h>

#include <avr/eeprom.h>

#include <avr/fuse.h>

#include <avr/interrupt.h>

#include <avr/lock.h>

#include <avr/power.h>

#include <avr/pgmspace.h>

#include <avr/sfr\_defs.h>

#include <avr/sleep.h>

#include <avr/version.h>

#include <util/crc16.h>

#include <util/parity.h>

#define FOSC 8000000 *// Clock Speed of the system*

#define BAUD 9600 *// Transmission rate for Baud Transmission*

#define MYUBRR FOSC/16/BAUD -1 *// Formula for Transmission of the Baud Rate Register*

volatile uint8\_t DataADC;

void USART0Init(void){

*/\*Set baud rate \*/*

UBRR0H |= (MYUBRR >> 8); *// shifts high bits from the lower to the higher register*

UBRR0L |= MYUBRR; *//Sets the Baud rate value*

UCSR0B |= (1 << RXEN0) | (1 << TXEN0); *// Enable receiver and transmitter*

UCSR0C |= (1 << UCSZ01) | (1 << UCSZ00); *// Set frame: 8data, 1 stop*

}

void USARTSendDataByte(unsigned char blank){

while(!(UCSR0A&(1<<UDRE0))){ *// Standard procedure to check if UDR buffer is empty*

}

UDR0 = blank; *//Sends data from the function call through serial transmission*

}

void InitialzeIO(void){

*// Debug ports to check if all functions are working*

DDRC = (0 << DDC0); *//Input Port for ADC Temp Sensor*

*/\*PORTC |= (1<<PORTC0);\*/*

DDRD |= (1<<DDD2)|(1<<DDD6); *// Debug Ports*

}

void InitTimer(void) *//Timer used for ADC conversion, set to 1s*

{

OCR1A = 8000; *//1s value at 8mhz clock rate*

TIMSK1 |= (1 << OCIE1A); *//enables CTC mode interrupts*

TCCR1B |= (1 << WGM12); *//sets to Mode 4 or CTC mode*

}

void startTimer(void) *//function call that starts timer*

{

TCCR1B |= (1 << CS12) | (1 << CS10); *//clock prescaler that divides clock 1024 for the system clock*

}

void stopTimer(void) *//stops timer*

{

TCCR1B &= ~((1 << CS12) | (1 << CS10));

TIMSK1 &=~(1<<OCIE0A);

}

void InitADC(){ *//initializes ADC*

*//select Vref = AVCC and left adjust result setting to 8 bits*

ADMUX = (0<<REFS1)|(0<<REFS0)|(1<<ADLAR);

*//set prescaler, enable autoriggering, enable adc interrupt, and enable adc*

ADCSRA |= (1<<ADPS1)|(1<<ADPS2)|(1<<ADPS0)|(1<<ADATE)|(1<<ADIE)|(1<<ADEN);

ADCSRB |= (0<<ADTS1)|(1<<ADTS0)|(1<<ADTS2); *//Set clock division value to 256*

}

void SetADCChannel(int ADCchannel){ *//function to select which ADC channel to pickup from for output going to the ADCMUX*

if (ADCchannel == 5)

{

ADMUX = (0<<MUX1)|(0<<MUX3)|(1<<MUX2)|(1<<MUX0);

}

else if (ADCchannel == 0)

{

ADMUX = (0<<MUX1)|(0<<MUX3)|(0<<MUX2)|(0<<MUX0);

}

else

{

ADMUX = (0<<MUX1)|(1<<MUX3)|(0<<MUX2)|(0<<MUX0);

}

}

void startADC(void) *//Starts ADC transmission*

{

ADCSRA |= (1<<ADSC);

}

void StopADC(void)

{

ADCSRA &= ~((1<<ADEN)|(1<<ADIE));

}

int main(void) *//main body calls all other functions to initialize the proper values of the Transmission protocol, timer and ADC module, then starts them*

{

USART0Init();

InitialzeIO();

SetADCChannel(0);

InitADC();

InitTimer();

startTimer();

startADC();

sei(); *// enables interrupts*

while(1)

{

*// Main loop*

}

}

ISR(ADC\_vect) *//After 1 second, ADC conversion takes place*

{

while(ADIF == 0){ *//check to see if ADCH and ADCL have been populated before proceeding*

};

DataADC = ADCH \* 2; *// is the value of the temperature sensor captured and stored in the variable*

USARTSendDataByte(DataADC);

TIFR1 = (0<<OCF1A)|(0<<OCF1B); *//disables flag to reset interrupt and start again*

StopADC(); *//stops ADC to restart back to main.*

}

Flowchart:

Setup of Part 1 of Design Assignment 

