Michael Ghisilieri

CPE301 – SPRING 2016

Design Assignment 6

**DO NOT REMOVE THIS PAGE DURING SUBMISSION:**

The student understands that all required components should be submitted in complete for grading of this assignment.

|  |  |  |  |
| --- | --- | --- | --- |
| **NO** | **SUBMISSION ITEM** | **COMPLETED (Y/N)** | **MARKS**  **(/MAX)** |
| 0. | COMPONENTS LIST AND FLOW CHART |  |  |
| 1. | INITIAL CODE OF TASK 1 |  |  |
| 2. | SCHEMATIC |  |  |
| 3. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 4. | SCREENSHOT OF EACH DEMO |  |  |
| 5. | VIDEO LINKS OF EACH DEMO |  |  |
| 6. | GITHUB LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| 0. | COMPONENTS LIST AND FLOW CHART |  |  |

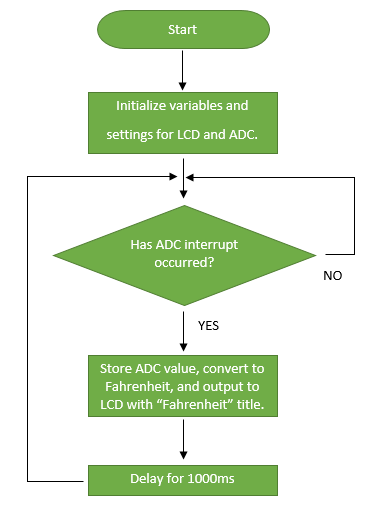
**COMPONENTS:**

ATmega328P Chip x1

LM34 Temperature Sensor x1

Nokia 5110 GLCD x1

**FLOWCHART:**



|  |  |  |  |
| --- | --- | --- | --- |
| 1. | INITIAL CODE OF TASK 1 |  |  |

/\*

\* DA6\_Task1.c

\*

\* Created: 4/21/2016 5:53:10 PM

\* Author : Michael

\*/

#define *F\_CPU* 8000000UL // declare clock frequency as 8MHz for delay function

#include <avr/io.h> // library for I/O

#include <stdint.h> // needed for uintX\_t variables

#include <avr/interrupt.h> // library for ADC interrupt

#include <util/delay.h> // library for \_delay\_ms() function

#include "nokia5110.h" // nokia 5110 file for its functions

void delay (); // function declaration for delay

// 32 bit integer to hold ADC value during conversion to Fahrenheit value

volatile *uint32\_t* temperature;

volatile *uint16\_t* ADCvalue; // Global variable, set to volatile if used with ISR

volatile char temp\_out; // current digit of temperature being transmitted

int main(void)

{

nokia\_lcd\_init(); // initialize the LCD

nokia\_lcd\_clear(); // clear the LCD

ADMUX = 0; // use ADC0

ADMUX |= (1 << REFS0); // use AVcc as the reference

ADCSRA |= (1 << ADPS2) | (1 << ADPS1); // 64 prescale for 16Mhz

ADCSRA |= (1 << ADATE); // Set ADC Auto Trigger Enable

ADCSRB = 0; // 0 for free running mode

ADCSRA |= (1 << ADEN); // Enable the ADC

ADCSRA |= (1 << ADIE); // Enable Interrupts

ADCSRA |= (1 << ADSC); // Start the ADC conversion

sei();

while(1) // loop forever and wait for ADC interrupt

{

}

return 0;

}

ISR(ADC\_vect)

{

ADCvalue = ADC; // read all bits of ADC (10) and store into ADCvalue

temperature = ADCvalue; // copy value into new variable in case ADC overwrites it

// equation used to convert ADC value is:

// (ADC register \* 5 \* 100) / 1024

// which is the ADC multiplied by the reference voltage multiplied by 100

// since the LM34 sensor gives its value in mV and then divide by 1024 since

// a ten bit register will have a max value of 2^10 or 1024

temperature = temperature\*5;

temperature = temperature\*100;

temperature = temperature/1024;

temp\_out = temperature / 10; // dividing 2 digit temp (i.e. 75 or 80) will give tens digit

nokia\_lcd\_set\_cursor(0, 0); // set cursor to beginning of LCD

nokia\_lcd\_write\_string("Fahrenheit", 1); // print "Fahrenheit" with size scale of 1

nokia\_lcd\_set\_cursor(0, 10); // move to second line

nokia\_lcd\_write\_char(temp\_out+'0', 4); // print tens digit with size scale of 4

temp\_out = temperature % 10; // % 10 will give the ones digit of a 2 digit temp

nokia\_lcd\_write\_char(temp\_out+'0', 4); // print ones digit with size scale of 4

nokia\_lcd\_render(); // render to display all the writes from above

delay(); // delay for 1000 ms or 1 second

}

void delay () {

int i; // declare i for counter

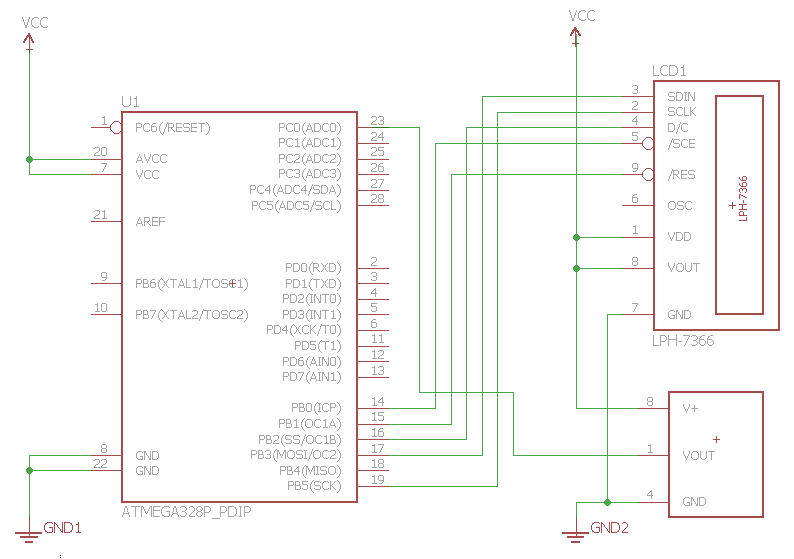
for(i = 0; i < 100; i++) { // loop 100 times

*\_delay\_ms*(10); // delay 10 ms

} // total delay = 100\*10ms = 1000ms

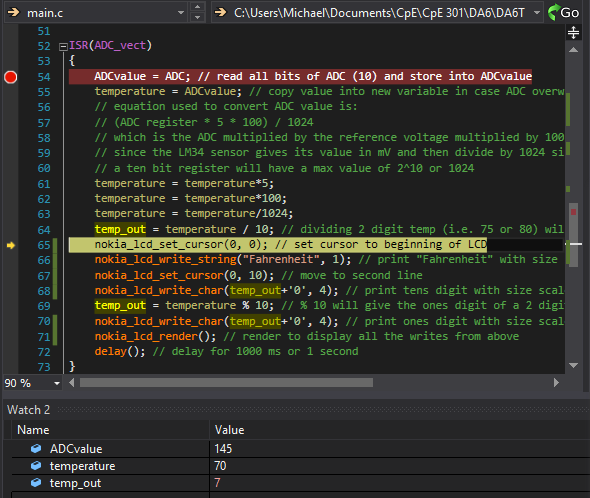
}

|  |  |  |  |
| --- | --- | --- | --- |
| 2. | SCHEMATIC |  |  |

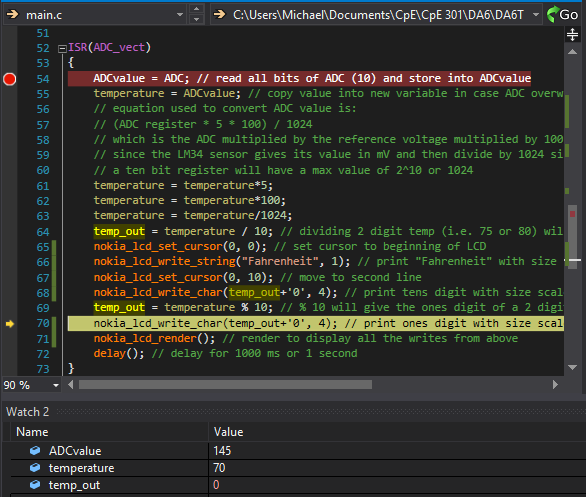


|  |  |  |  |
| --- | --- | --- | --- |
| 3. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |

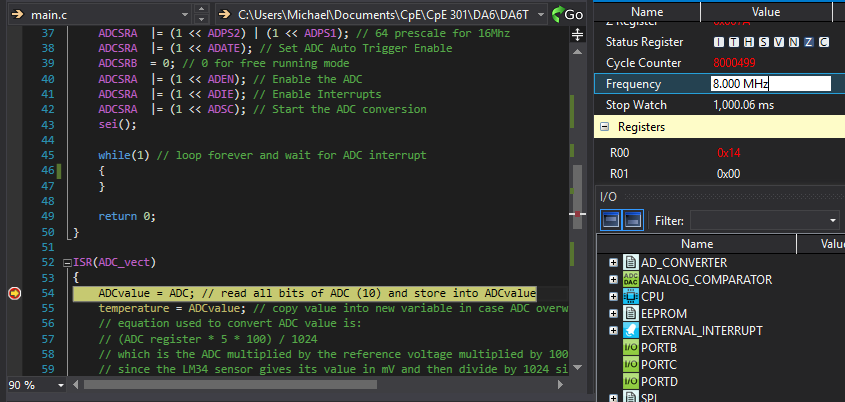
TASK 1: Giving a random value to ADCvalue, which would be straight from the ADC, of 145 will cause the equation used in this interrupt to calculate a temperature of 70 degrees Fahrenheit. The first expression for temp\_out will grab the tens digit of ‘7’ and convert it to a char when writing it to the screen with *nokia\_lcd\_write\_char*.



The next thing to be output to the GLCD will be the ones digit which can be seen as ‘0’ with the next temp\_out expression.

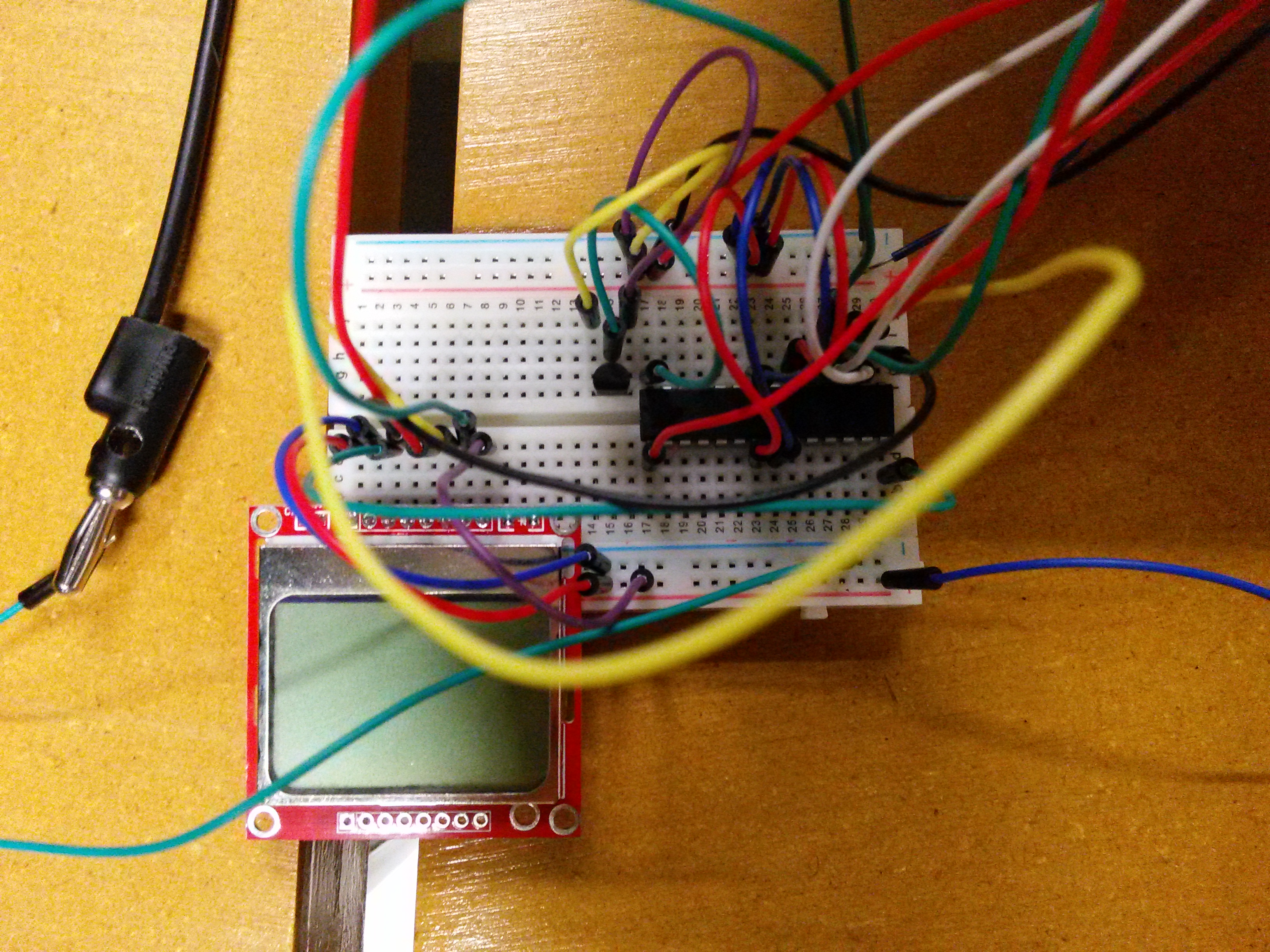


The time taken to get back into the ADC interrupt from the delay is 1,000.06 ms which is **1.00006** seconds. This is very close to being 1 second for displaying the temperature although there is no update time that is specified.



|  |  |  |  |
| --- | --- | --- | --- |
| 4. | SCREENSHOT OF EACH DEMO |  |  |

TASK 1: Breadboard with two power supplies connected for 3.3V (bottom rail) and 5V (top rail).



|  |  |  |  |
| --- | --- | --- | --- |
| 5. | VIDEO LINKS OF EACH DEMO |  |  |
| <https://youtu.be/l-LBvTEG7XM> | | | |
| 6. | GITHUB LINK OF THE DA |  |  |
| https://github.com/michael-ghisilieri/CpE301\_DAs.git | | | |

**Student Academic Misconduct Policy**

<http://studentconduct.unlv.edu/misconduct/policy.html>

“This assignment submission is my own, original work”.

Michael Ghisilieri