Martin Jaime

CPE301 – SPRING 2016

Design Assignment 3

**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 CONNECTION BLOCK DIAGRAM w/ PINS |  |  |
| 1. | INITIAL CODE OF TASK 1/A |  |  |
| 2. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 2/B |  |  |
| 3. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 3/C |  |  |
| 4. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 4/D |  |  |
| 5. | INCREMENTAL / DIFFERENTIAL CODE OF TASK 5/E |  |  |
| 6. | SCHEMATICS |  |  |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |
| 8. | SCREENSHOT OF EACH DEMO |  |  |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| 10. | GOOGLECODE LINK OF THE DA |  |  |
|  |  |  |  |
|  |  |  |  |

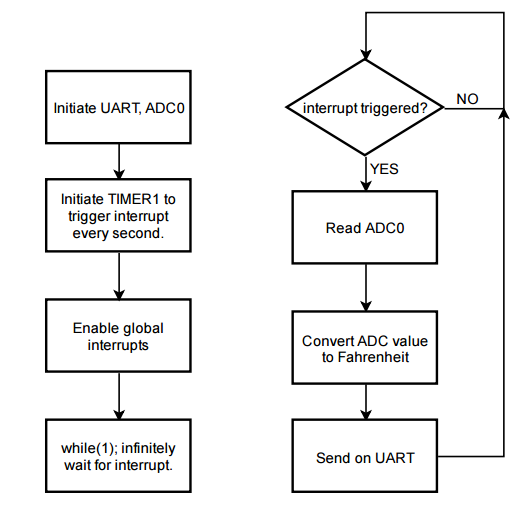
|  |  |  |  |
| --- | --- | --- | --- |
| 0. | COMPONENTS LIST AND CONNECTION BLOCK DIAGRAM w/ PINS |  |  |

* Atmega328P
* LM34
* Pololu USB AVR programmer for serial communication
* AVR Studio 7

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

Task 1: Write a C AVR program that will monitor the LM34/35 connected to an Analog pin to display the temperature in F on the serial terminal every 1 sec. Use a timer with interrupt for the 1 sec delay. You can use a FTDI chip for serial to USB conversion.

**Flow chart:**



#define *F\_CPU* 8000000UL

#include <stdio.h>

#include <avr/io.h>

#include <util/delay.h>

#include <avr/interrupt.h>

#define BAUDRATE 9600 // set baudrate

#define ASYNCH\_NORM\_PRESCALER (*F\_CPU*/16/BAUDRATE - 1) // prescaler value

int USART0\_sendChar(char, *FILE*\*); // Send character on USART0

void USART0\_init (void); // Initialize USART0

void ADC0\_init(); // to initiate ADC

void TIMER1\_init(); // initiate timer1

// reset stream pointer

// http://www.gnu.org/savannah-checkouts/non-gnu/avr-libc/user-manual/group\_\_avr\_\_stdio.html

*FILE* USART0\_stream = *FDEV\_SETUP\_STREAM*(USART0\_sendChar, *NULL*, *\_FDEV\_SETUP\_WRITE*);

int main(void)

{

*stdout* = &USART0\_stream; // change standard output to point to a USART stream

USART0\_init(); // Initiate USART0

ADC0\_init(); // Initiate ADC0

TIMER1\_init(); // Initiate TIMER1

sei(); // enable global interrupts

*printf*("\n"); // Go to the next line in case current line printed junk.

while(1); // infinite loop. Wait for interrupt.

return 0;

}

ISR(TIMER1\_COMPA\_vect) // interrupt routine for Output Compare Match A.

// ISR that triggers at TIMER1\_COMPT\_vect

{

*uint16\_t* ADC\_value = ADC; // read ADC conversion

float temperature\_F;

c

temperature\_F = (ADC\_value / 1024.0) \* 5000;

// 10 mV per degree Fahrenheit

temperature\_F = temperature\_F / 10; // Convert to Fahrenheit

*printf*("Temp(F) LM34: %3.2f\r", temperature\_F);

}

void TIMER1\_init()

// TIMER1\_init initiates TIMER1 for 1 second delays to trigger an interrupt at 1 Hz

{

// WGM = 0100

TCNT1 = 0;

OCR1A = 2\**F\_CPU*/2/256-1;

TIMSK1 = (1<<OCIE1A);

TCCR1A = 0x00;

TCCR1B = (1<<WGM12) | (1<<CS12) | (0<<CS11) | (0<<CS10);

}

void ADC0\_init()

// ADC0\_init will initiate ADC acquisition at ADC0

{

DDRC &= ~(0<<DDC0); // Clear bit 0 of DDRC

ADMUX = 0; // use ADC0

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

ADCSRA |= (1 << ADPS2) | (1 << ADPS1)/\* | (1 << ADPS0)\*/; // 128 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 << ADSC); // Start the ADC conversion

}

int USART0\_sendChar(char data, *FILE* \*stream)

/\*

\* Procedure to send a single character over USART0. If character is linefeed, reset

\* line.

\* Assumes ASCII code.

\*/

{

if(data == '\n') // If linefeed, also print a return.

{

while(! (UCSR0A & (1<<UDRE0)) );// Wait for data register to be available.

UDR0 = '\r'; // send return char to data register.

}

while(! (UCSR0A & (1<<UDRE0)) ); // Wait for data register to be available.

UDR0 = data; // send char to UART.

return 0;

}

void USART0\_init (void)

/\*

\* Procedure to initialize USART0 asynchronous with enabled RX/TX, 8 bit data,

\* no parity, and 1 stop bit.

\*/

{

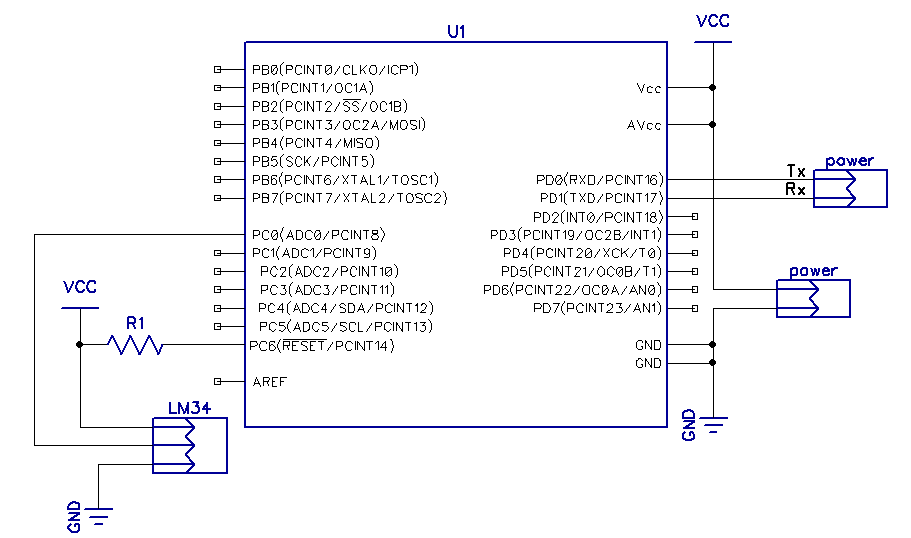
UCSR0B = (1<<TXEN0) | (1<<RXEN0); // enable transmit/receive

UCSR0C = (1<<UCSZ01) | (1<<UCSZ00); // asynchronous, 8N1

UBRR0L = ASYNCH\_NORM\_PRESCALER; // To set 9600 baud rate with 8MHz clock

}

|  |  |  |  |
| --- | --- | --- | --- |
| 6. | SCHEMATICS |  |  |



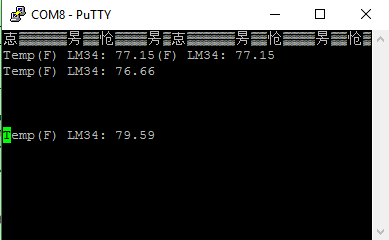
|  |  |  |  |
| --- | --- | --- | --- |
| 7. | SCREENSHOTS OF EACH TASK OUTPUT |  |  |

TASK 1:

At a frequency of 1Hz, the current captured temperature was sent over UART and printed on the serial terminal. By using

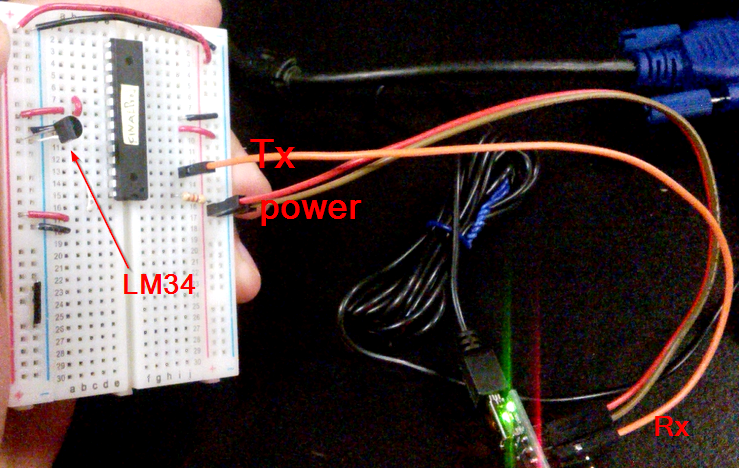
*printf*("Temp(F) LM34: %3.2f\r", temperature\_F);

The information could be printed on a single line since the return character ‘\r’ made the next incoming text print over the old.



|  |  |  |  |
| --- | --- | --- | --- |
| 8. | SCREENSHOT OF EACH DEMO |  |  |

TASK 1:



|  |  |  |  |
| --- | --- | --- | --- |
| 9. | VIDEO LINKS OF EACH DEMO |  |  |
| <https://drive.google.com/file/d/0B4ItVBjMqlnybmIzOGVwRk1VemM/view?usp=sharing> | | | |
| 10. | GITHUB REPOSITORY LINK OF THE DA |  |  |
| <https://github.com/martinjaime/CpE301_Assignments2016S.git> | | | |

**Student Academic Misconduct Policy**

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

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

Martin Jaime