Ömer Karslıoğlu

MICROPROCESSORS LABORATORY 7

Introduction

General Purpose of This Laboratory :

The general purpose of this laboratory application is to do the project, which reads the data we have obtained as analog with the lm35 over the analog digital converter (ADC) peripheral and sends it to the computer with UART serial communication, showing the ambient temperature on the PUTTY screen and showing the situation with LEDs in terms of visuality.

Pin/Port Selection :

I have made my connections as in the image below with little differences.
I did not connect the Vcc part because the USB TTL converter is fed from the computer. I just connected the GND line.
I didn't use PF4 as the leds are connected to PF3, PF2 and PF1 in Tiva tm4c123 launch pad.
I connected the TX-RX of the USB-TTL converter to PE4(TX)-PE5(RX) and middle leg of LM35 to PE3 pin .

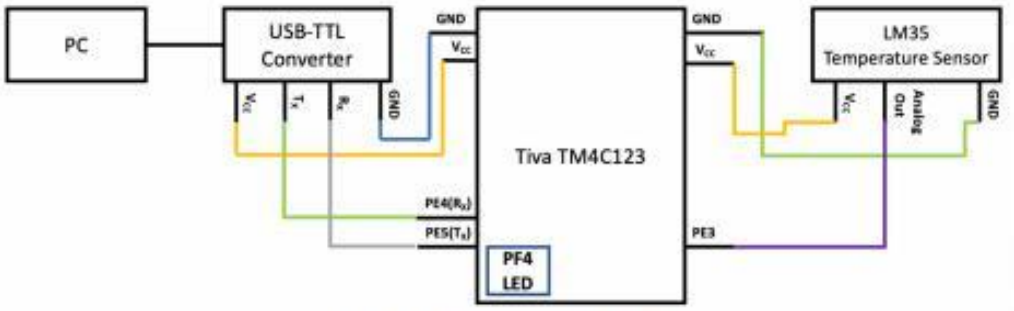
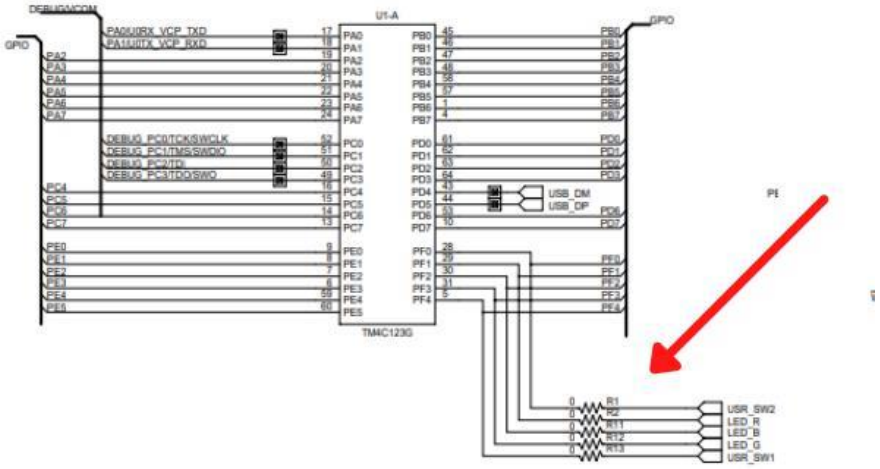


Figure-8: Experiment Setup

The image from EE226 Lab. Manual



The image from Tiva c4 series user's manual [1] from page 20 (Schematic Image)

Result And Discussion

The main purpose of the code I have written and this application I have prepared is to send the value received from the temperature sensor (LM35) to the computer via UART serial communication.It would be good to start my story about this laboratory application by explaining how I use the temperature (LM35) sensor.

The working principle of the LM35 [2] is as follows: The output voltage increases by 10mV for each degree. With this value, the required ratio is established and the temperature of the external environment is converted from analog to digital [3] , that is, to values that we can see digitally.Based on this information , I had to activate the analog peripheral of our TIVA TM4C123GH6PM launch pad first before I could read data from the LM35 . That's why I made a function definition called "analogInit() ". In this function I set the RCGCGPIO register to PE3 to activate the clock of the first head PE3 pin and activate the RCGCADC register and the ADC peripheral. I set my GPIO configs to read analog from PE3. One of the GPIO registers, the AMSEL register played an important role here.

Sequencer	Number of Samples	Depth of FIFO
SS3	1	1
SS2	4	4
SS1	4	4
SS0	8	8

Each ADC module contains 4 sequencer modules. Thanks to these modules, I can take samples from one or more ADC inputs at a time. Sequencer modules store the received samples according to their fifo capacity.As seen in the table above, there are sequencer modules with 1 to 8 warehouses. SS3 will do the job for me as I will be reading analog from a single input.I chose SS3 by setting the ACTSS register. I selected a software event as a start conversion trigger with EMUX register.TM4C123G microcontroller provides 12 analog channels. ADCSSMUXn registers (where n=0, 1,2,3) select analog channels for sample sequencers. For example, if we want to use SS3 and analog channel A0, this line will select AN0 analog channel for sample sequencer 3 or SS3. Based on this information , I chose the A0 peripheral .
I set the SSCTL register to take one sample at a time and set the flag in the first sample.Thus, I have completed the ADC configuration.In the main loop, I started the sampling operation with the PSSI register and checked that it is active. When analogue read data is converted to a digital number, it is retained in the FIFO. After getting the value we read from here, I cleared the flag and finished the analog reading process.I put the value I read into the adc_value variable. To convert this analog value that I read later to the temperature value (according to the information I learned from the datasheet), I applied the following process : (adc_value * 330)/4095 . Finally, I sent the values I read to the computer via UART serial communication and light the leds according to the situations.

Now, let me briefly explain how I use the UART [4] serial communication protocol and peripheral on the launch pad.By short definition, UART is a hardware communication protocol that uses asynchronous serial communication with configurable speed. Asynchronous means there is no clock signal to synchronize the output bits from the transmitting device going to the receiving end.It uses two buses, TX transmitting data and RX receiving data .

Let me now explain how I initialize the UART peripheral.While performing the UART communication, the baudrate (data transport rate) must be set first after the clock configurations connected to the UART peripharel are made.I assigned 104 to the IBRD register and 11 to the FBRD register to set the baudrate to 9600.
Since the data bit length is usually 8 and the frame is set as one stop bit, I have configured it this way.That's why I assigned 0x60 to the LCRH register. Thus, my data length became 8 bits.I declared that with CC register it will work with system clock of UART.Then I activated the UART with the CTL register. Finally ,I determined my tx & rx pins by configuring GPIO AFSEL and GPIO PCTL registers.

UART Functions :
printChar(unsigned char data) : If transmitter bus is available , it sends the data to UART Data register .
printingString(char *str) : If transmitter bus is available , it sends the data to UART Data register as byte byte .

Conclusion

I used the experience I gained with UART in the previous laboratory application in this laboratory application. I set the "startup" file for the UART. I gained more knowledge and experience with UART in the TIVA TM4C123 launch pad.

In addition, I learned in detail how to configure the analog digital converter, how to initial it and how to process the read value. At the same time, I think that I had the opportunity to learn the working principle of the LM35 sensor and how to process the read value.

References

- [1] https://www.ti.com/lit/ug/spmu296/spmu296.pdf?ts=1623072786863
- [2] https://www.ti.com/lit/ds/symlink/lm35.pdf
- [3] http://class.ece.iastate.edu/cpre288/lectures/lect12_13.pdf
- [4] https://www.analog.com/en/analog-dialogue/articles/uart-a-hardware-communication-protocol.html#

MY YOUTUBE VIDEO FOR THE LAB 7

https://youtu.be/pBQ6Ek_jWVg

Appendix

```
// Omer Karslioglu-17050211061
#include "TM4C123GH6PM.h"
#include <stdio.h>

#define red 0x02
#define blue 0x04
#define green 0x08

#define PORTF_DATA  (*((volatile unsigned long *)0x40025038))

void Delay(unsigned long counter);
void PortF_init(void);
void UART5_init(void);
void UART5_Transmitter(unsigned char data);
void printstring(char *str);

char mesg[12];
float tempature;

int main(void)
{
   unsigned int adc_value;
          PortF_init();
   UART5_init();

   SYSCTL->RCGCGPIO |= (1<<4);
   SYSCTL->RCGCADC |= (1<<0);

   GPIOE->AFSEL |= (1<<3);
   GPIOE->DEN &= ~(1<<3);
   GPIOE->AMSEL |= (1<<3);

   ADC0->ACTSS &= ~(1<<3);
   ADC0->EMUX &= ~0xF000;
   ADC0->SSMUX3 = 0;
   ADC0->SSCTL3 |= (1<<1)|(1<<2);
   ADC0->ACTSS |= (1<<3);


   while(1)
   {
      ADC0->PSSI |= (1<<3);
      while((ADC0->RIS & 8) == 0) ;
      adc_value = ADC0->SSFIFO3;
      ADC0->ISC = 8;


                  tempature = (adc_value * 330)/4095;
                  sprintf(mesg, "\r\nTempature = %0.1f Celcius", tempature);
                  printstring(mesg);
      Delay(2000);

                        if(tempature<=24 && tempature>=21){
                              PORTF_DATA = red;
```

```
                        }
                        else if (tempature<=27){
                                PORTF_DATA = blue;
                        }
                        else if (tempature<=30){
                                PORTF_DATA = green;
                        }
                        else{
                                PORTF_DATA=0x00;
                        }

        }
}
void PortF_init(void)
{
  SYSCTL->RCGCGPIO |= 0x20;        // initialize clock of Port F
  GPIOF->DIR |=0x0E;         // set PF4 as a digial output pin
  GPIOF->DEN |=0x1F;
  GPIOF->DATA = 0;
}
void UART5_init(void)
{
    SYSCTL->RCGCUART |= 0x20;
    SYSCTL->RCGCGPIO |= 0x10;
    Delay(1);

    UART5->CTL = 0;
    UART5->IBRD = 104;
    UART5->FBRD = 11;
    UART5->CC = 0;
    UART5->LCRH = 0x60;
    UART5->CTL = 0x301;


    GPIOE->DEN = 0x30;
    GPIOE->AFSEL = 0x30;
    GPIOE->AMSEL = 0;
    GPIOE->PCTL = 0x00110000;
}
void UART5_Transmitter(unsigned char data)
{

    while((UART5->FR & (1<<5)) != 0);
    UART5->DR = data;

}

void printstring(char *str)
{
  while(*str)
  {

        UART5_Transmitter(*(str++));

  }
```

```
}
void Delay(unsigned long counter)
{

  unsigned long i = 0;

  for(i=0; i< counter*1000; i++);

}
```