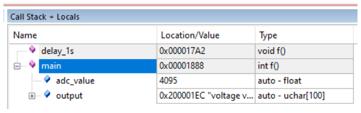
EE447 Preliminary Work Experiment 5

Deniz Soysal 2305332

December 8, 2024

Task 1

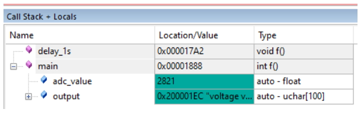
First, we may need to change the clock register if our pin is in a different port. Then we need to change all the gpıo configuration pins AFSEL, AMSEL, DIR, DEN registers according to pin address. For the ADC module, only ACTSS would need to be modified to use a different channel according to our new pin selection.



Call Stack + Locals

Name	Location/Value	Type
delay_1s	0x000017A2	void f()
main	0x00001888	int f()
adc_value	4095	auto - float
output	0x200001EC "voltage v...	auto - uchar[100]

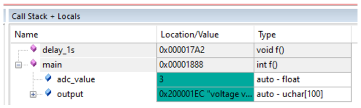
Figure 1: Variable states when 3.3V is connected to PB4.



Call Stack + Locals

Name	Location/Value	Type
delay_1s	0x000017A2	void f()
main	0x00001888	int f()
adc_value	2821	auto - float
output	0x200001EC "voltage v...	auto - uchar[100]

Figure 2: Variable states when 5V is connected to PB4.



Call Stack + Locals

Name	Location/Value	Type
delay_1s	0x000017A2	void f()
main	0x00001888	int f()
adc_value	3	auto - float
output	0x200001EC "voltage v...	auto - uchar[100]

Figure 3: Variable states when 0V is connected to PB4.

```
#include <TM4C123GH6PM.h>
#include <stdio.h>
#include "initialize.h"

int main() {
    float adc_value;

    // initialize PB4 as parallel I/O and ADC0 with ss3
    gpio_init(); // initializing PB4
    adc_init(); // initializing ADC0 module
    UART0_init(); // initializing UART module for printing to termite

    ADC0->PSSI |= (1<<3); // start sampling
    while((ADC0->RIS & 8) == 0)  // wait until sampling is complete
    adc_value = ADC0->SSFIFO3;

    ADC0->ISC |= (1<<3); // clear conversion flag
}
```

Task 2

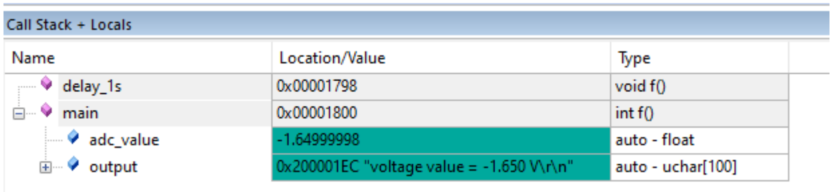
The offset can be given with a simple subtraction operation.

```
adc_value = ADC0->SSFIFO3 - 1.65;
```

Task 3

The final value can be normalized with the following operation.

```
adc_value = ((float)ADC0->SSFIFO3 * 3.3) / 4095.0 - 1.65; // Scale ADC value to voltage
```



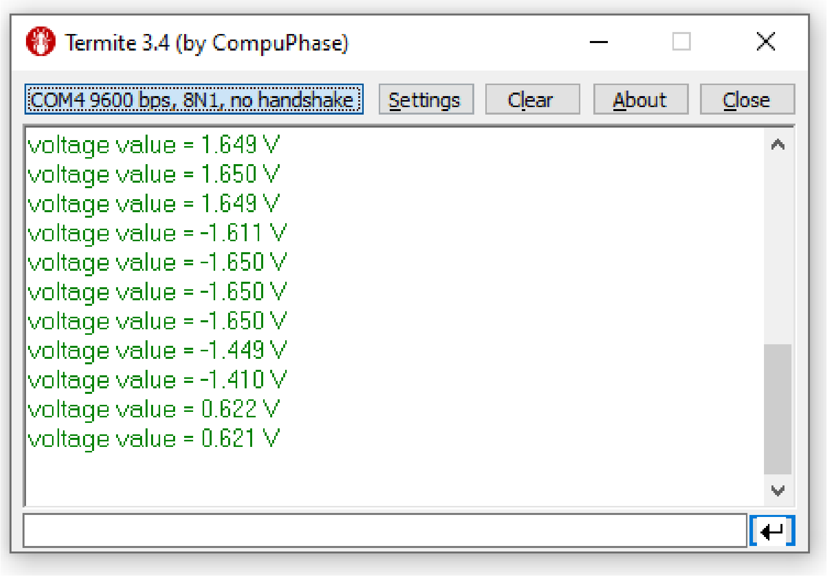
Call Stack + Locals

Name	Location/Value	Type
delay_1s	0x00001798	void f()
main	0x00001800	int f()
adc_value	-1.64999998	auto - float
output	0x200001EC "voltage value = -1.650 V\r\n"	auto - uchar[100]

Figure 4: Output array as part of locals.

Task 4

A normalized ADC input value can be converted and printed on termite with a UART connection. This can be automatized by inserting it into a while loop.



COM4 9600 bps, 8N1, no handshake

```
voltage value = 1.649 V
voltage value = 1.650 V
voltage value = 1.649 V
voltage value = -1.611 V
voltage value = -1.650 V
voltage value = -1.650 V
voltage value = -1.650 V
voltage value = -1.449 V
voltage value = -1.410 V
voltage value = 0.622 V
voltage value = 0.621 V
```

Figure 5: Termite screen when 3.3V, 0V, and 5V is given to PB4.

The following code is the main body of this operation:

```
#include <TM4C123GH6PM.h>
#include <stdio.h>
#include "initialize.h"
#include "write.h"
#include "delay.h"

int main() {
  float adc_value;
 char output[100];

 // initialize PB4 as parallel I/O and ADC0 with ss3
 gpio_init();
 adc_init();
 UART0_init();


 while(1){
  ADC0->PSSI |= (1<<3); // start sampling
  while((ADC0->RIS & 8) == 0)  // wait until sampling is complete
  adc_value = ((float)ADC0->SSFIFO3 * 3.3) / 4095.0 - 1.65; // Scale ADC value to voltage

  ADC0->ISC |= (1<<3); // clear conversion flag

  sprintf(output,"voltage value = %.3f V\r\n", adc_value);
  printstring(output);
  delay_1s();
 }
}
```

Task 5

Using a PWM module and modifying the duty cycle in accordance with the sampled value. By giving the output of PWM to PF2 led can be directly used.

```
#include <TM4C123GH6PM.h>
#include "initialize.h"
#include "delay.h"

int main(){
 pwm_init();
 gpioPF2_init(); //output
 gpioPB4_init(); //input
 adc_init();

 int duty_cycle = 0;
  while(1)
  {
    ADC0->PSSI |= (1<<3); // start sampling
    while((ADC0->RIS & 8) == 0)  // wait until sampling is complete
      duty_cycle = (ADC0->SSFIFO3 - 96) * 4;

   ADC0->ISC |= (1<<3); // clear conversion flag
      if (duty_cycle >= 16000) {duty_cycle = 0;}
      PWM1->_3_CMPA = 16000 - duty_cycle;
      delay_100ms();
  }
}
```

Initializer.h

```
#include <TM4C123GH6PM.h>

void gpioPF2_init(void);
void pwm_init(void);
void gpioPB4_init(void);
void timer0_init(void);
void adc_init(void);

void gpioPF2_init(void){
SYSCTL->RCGCGPIO |= 0x20;
__ASM("NOP");
__ASM("NOP");
__ASM("NOP");

GPIOF->AFSEL |= (1<<2);
GPIOF->PCTL &= ~0x00000F00;
GPIOF->PCTL |= 0x00000500;
GPIOF->DEN |= (1<<2);
}
void pwm_init(void){
SYSCTL->RCGCPWM |= 2;
__ASM("NOP");
__ASM("NOP");
__ASM("NOP");
__ASM("NOP");
__ASM("NOP");
__ASM("NOP");


PWM1->_3_CTL &= ~(1<<0);
PWM1->_3_CTL |= (1<<1);   // count UP
PWM1->_3_GENA = 0x0000008C;
PWM1->_3_LOAD = 16000;
PWM1->_3_CMPA = 8000-1;
PWM1->_3_CTL = 1;
PWM1->ENABLE = 0x40;
}

void gpioPB4_init(void){
    SYSCTL->RCGCGPIO |= (1<<1);
   __ASM("NOP");
   __ASM("NOP");
   __ASM("NOP");

  GPIOB->AFSEL |= (1<<4);
    GPIOB->DIR &= ~(1<<4);
    GPIOB->DEN &= ~(1<<4);
   GPIOB->AMSEL |= (1<<4);
}

void adc_init(void){
    SYSCTL->RCGCADC |= (1<<0);
    __ASM("NOP");
   __ASM("NOP");
   __ASM("NOP");
   __ASM("NOP");
   __ASM("NOP");
   __ASM("NOP");


   ADC0->ACTSS &= ~(1<<3);
   ADC0->EMUX  &= ~0x0000F000;
   ADC0->SSMUX3 = 0xA;
  ADC0->SSCTL3 |= (1<<2)|(1<<1);
   ADC0->PC &= ~0xF;
    ADC0->PC |= 0x01;
   ADC0->ACTSS |= (1<<3);
}
```

delay.h

```
#include <TM4C123GH6PM.h>

void delay_100ms(void);


void delay_100ms(void){
  SYSCTL->RCGCTIMER |= (1<<1);  /*enable clock Timer1 subtimer A in run mode */
    TIMER1->CTL = 0; /* disable timer1 output */
    TIMER1->CFG = 0x4; /*select 16-bit configuration option */
    TIMER1->TAMR = 0x02; /*select periodic down counter mode of timer1 */
    TIMER1->TAPR = 250-1; /* TimerA prescaler value */
    TIMER1->TAILR = 64-1 ; /* TimerA counter starting count down value  */
    TIMER1->ICR = 0x1;          /* TimerA timeout flag bit clears*/
    TIMER1->IMR |=(1<<0); /*enables TimerA time-out  interrupt mask */

    TIMER1->CTL |= 0x01;        /* Enable TimerA module */
    while ((TIMER1->RIS & 0x01) == 0); // Wait for the timeout flag to be set
}
```

Appendix: Complete code of part1-4

```
#include <TM4C123GH6PM.h>
#include <stdio.h>
#include "initialize.h"
#include "write.h"
#include "delay.h"

int main() {
  float adc_value;
 char output[100];

 // initialize PB4 as parallel I/O and ADC0 with ss3
 gpio_init();
 adc_init();
 UART0_init();


 while(1){
  ADC0->PSSI |= (1<<3); // start sampling
  while((ADC0->RIS & 8) == 0)  // wait until sampling is complete
  adc_value = ((float)ADC0->SSFIFO3 * 3.3) / 4095.0 - 1.65; // Scale ADC value to voltage

  ADC0->ISC |= (1<<3); // clear conversion flag

  sprintf(output,"voltage value = %.3f V\r\n", adc_value);
  printstring(output);
  delay_1s();
 }
}

#include <TM4C123GH6PM.h>

void gpio_init(void);
void adc_init(void);

void gpio_init(void){
    SYSCTL->RCGCGPIO |= (1<<1);
   __ASM("NOP");
   __ASM("NOP");
   __ASM("NOP");

  GPIOB->AFSEL |= (1<<4);
    GPIOB->DIR &= ~(1<<4);
    GPIOB->DEN &= ~(1<<4);
   GPIOB->AMSEL |= (1<<4);
}

void adc_init(void){
    SYSCTL->RCGCADC |= (1<<0);
    __ASM("NOP");
   __ASM("NOP");
   __ASM("NOP");
   __ASM("NOP");
   __ASM("NOP");
   __ASM("NOP");
   ADC0->ACTSS &= ~(1<<3);
   ADC0->EMUX  &= ~0x0000F000;
   ADC0->SSMUX3 = 0xA;
  ADC0->SSCTL3 |= (1<<2)|(1<<1);
   ADC0->PC &= ~0xF;
    ADC0->PC |= 0x01;
   ADC0->ACTSS |= (1<<3);
}

#include "TM4C123GH6PM.h"

void UART0_init(void);
void UART0_Transmitter(unsigned char data);
void printstring(char *str);

void UART0_init(void)
{
    SYSCTL->RCGCUART |= 0x01;  /* Enable clock to UART0 */
    SYSCTL->RCGCGPIO |= 0x01;  /* Enable clock to PORTA for PA0/Rx and PA1/Tx */

    UART0->CTL = 0;            /* Disable UART0 module during configuration */
    UART0->IBRD = 104;         /* Integer part for 9600 baud rate */
    UART0->FBRD = 11;          /* Fractional part for 9600 baud rate */
    UART0->CC = 0;             /* Use system clock */
    UART0->LCRH = 0x60;        /* 8-bit data, no parity, 1 stop bit */
    UART0->CTL = 0x301;        /* Enable UART0 module, Rx and Tx */

    GPIOA->DEN = 0x03;         /* Enable digital functions for PA0 and PA1 */
    GPIOA->AFSEL = 0x03;       /* Enable alternate functions for PA0 and PA1 */
    GPIOA->PCTL = 0x11;        /* Configure PA0 and PA1 for UART */
}

void UART0_Transmitter(unsigned char data)
{
    while ((UART0->FR & (1 << 5)) != 0); /* Wait until Tx buffer is not full */
    UART0->DR = data;                    /* Transmit the data */
}

void printstring(char *str)
{
    while (*str)
    {
        UART0_Transmitter(*(str++));    /* Send characters one by one */
    }
}

#include <TM4C123GH6PM.h>

void delay_1s(void);

void delay_1s(void){
  SYSCTL->RCGCTIMER |= (1<<1);  /*enable clock Timer1 subtimer A in run mode */
    TIMER1->CTL = 0; /* disable timer1 output */
    TIMER1->CFG = 0x4; /*select 16-bit configuration option */
    TIMER1->TAMR = 0x02; /*select periodic down counter mode of timer1 */
    TIMER1->TAPR = 250-1; /* TimerA prescaler value */
    TIMER1->TAILR = 64000-1 ; /* TimerA counter starting count down value  */
    TIMER1->ICR = 0x1;          /* TimerA timeout flag bit clears*/
    TIMER1->IMR |=(1<<0); /*enables TimerA time-out  interrupt mask */

    TIMER1->CTL |= 0x01;        /* Enable TimerA module */
    while ((TIMER1->RIS & 0x01) == 0); // Wait for the timeout flag to be set
}
```