Riley Ruckman
TCES 460, Wi20
Lab 10

Video Links

- Part a: https://drive.google.com/file/d/1AiIX4qsEgddg80zEvi5DT2wSHzxamjyj/view?usp=sharing
- Part b: https://drive.google.com/file/d/1gKaBI0Hc1DDULKr49SFG45DKTfyr2JDv/view?usp=sharing

Part a:

Code:

```
// Riley Ruckman
// TCES 460, Wi21
// Lab 10 - Part a

//Create definitions of PORTF registers
#define GPIO_PORTF_DATA_R      (*((volatile unsigned long *)0x400253FC))
#define GPIO_PORTF_DIR_R       (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_AFSEL_R     (*((volatile unsigned long *)0x40025420))
#define GPIO_PORTF_PUR_R       (*((volatile unsigned long *)0x40025510))
#define GPIO_PORTF_DEN_R       (*((volatile unsigned long *)0x4002551C))
#define GPIO_PORTF_LOCK_R      (*((volatile unsigned long *)0x40025520))
#define GPIO_PORTF_CR_R        (*((volatile unsigned long *)0x40025524))
#define GPIO_PORTF_AMSEL_R     (*((volatile unsigned long *)0x40025528))
#define GPIO_PORTF_PCTL_R      (*((volatile unsigned long *)0x4002552C))
#define SYSCTL_RCGC2_R         (*((volatile unsigned long *)0x400FE108))

//Create definitions for PORTE registers
#define GPIO_PORTE_DATA_R      (*((volatile unsigned long *)0x400243FC))
#define GPIO_PORTE_DIR_R       (*((volatile unsigned long *)0x40024400))
#define GPIO_PORTE_AFSEL_R     (*((volatile unsigned long *)0x40024420))
#define GPIO_PORTE_PUR_R       (*((volatile unsigned long *)0x40024510))
#define GPIO_PORTE_DEN_R       (*((volatile unsigned long *)0x4002451C))
#define GPIO_PORTE_LOCK_R      (*((volatile unsigned long *)0x40024520))
#define GPIO_PORTE_CR_R        (*((volatile unsigned long *)0x40024524))
#define GPIO_PORTE_AMSEL_R     (*((volatile unsigned long *)0x40024528))
#define GPIO_PORTE_PCTL_R      (*((volatile unsigned long *)0x4002452C))

//Create definitions for UART5 registers
```

```
#define SYSCTL_RCGCUART_R      (*((volatile unsigned long *)0x400FE618))
#define UART5_IBRD_R           (*((volatile unsigned long *)0x40011024))
#define UART5_FBRD_R           (*((volatile unsigned long *)0x40011028))
#define UART5_CTL_R            (*((volatile unsigned long *)0x40011030))
#define UART5_LCRH_R           (*((volatile unsigned long *)0x4001102C))
#define UART5_DR_R             (*((volatile unsigned long *)0x40011000))
#define UART5_FR_R             (*((volatile unsigned long *)0x40011018))
#define UART5_CC_R             (*((volatile unsigned long *)0x40011FC8))

//Create definitions for timer0 registers
#define TIMER0_CFG_R           (*((volatile unsigned long *)0x40030000))
#define TIMER0_TAMR_R          (*((volatile unsigned long *)0x40030004))
#define TIMER0_CTL_R           (*((volatile unsigned long *)0x4003000C))
#define TIMER0_RIS_R           (*((volatile unsigned long *)0x4003001C))
#define TIMER0_ICR_R           (*((volatile unsigned long *)0x40030024))
#define TIMER0_TAILR_R         (*((volatile unsigned long *)0x40030028))
#define TIMER0_TAPR_R          (*((volatile unsigned long *)0x40030038))
#define SYSCTL_RCGCTIMER_R     (*((volatile unsigned long *)0x400FE604))

// Function Prototypes
void timer0A_delayMs(int ttime);
void timer0_InIt(void);
void PortF_Init(void);
void HC05_Init(void);
char BLT_Read(void);
void BLT_Write(unsigned char data);
void BLT_Write_Str(char *s);

int main(void)
{
                // Initialize PORTF and UART5 ports/features
                PortF_Init();
                HC05_Init();

   while(1)
   {
                        // Reads new data from Rx FIFO
                        char c = BLT_Read();

                        // If 'A' is received, a message is sent back and the onboard LED blinks
RED twice at 1 Hz.
                        if(c == 'A') {
                                BLT_Write_Str("Hello World!! RED LED ON\n");
                                for (int i = 0; i < 2; i++) {
```

```
                                GPIO_PORTF_DATA_R = 0x02;
                                timer0A_delayMs(1000);
                                GPIO_PORTF_DATA_R = 0x00;
                                timer0A_delayMs(1000);
                        }
                // If 'B' is received, a message is sent back and the onboard LED blinks
BLUE twice at 1 Hz.
                } else if(c == 'B') {
                        BLT_Write_Str("Hello World!! BLUE LED ON\n");
                        for (int i = 0; i < 2; i++) {
                                GPIO_PORTF_DATA_R = 0x04;
                                timer0A_delayMs(1000);
                                GPIO_PORTF_DATA_R = 0x00;
                                timer0A_delayMs(1000);
                        }
                // If 'C' is received, a message is sent back and the onboard LED blinks
RED-BLUE-GREEN at 1 Hz.
                } else if(c == 'C') {
                        BLT_Write_Str("BYE World!! RGB LEDs ON\n");
                        for (int i = 0; i < 3; i++) {
                                // Since RED, BLUE, and GREEN are 0x2, 0x4, and 0x8,
respectively, shifting
                                // 2 by i will give the correct values for a
RED-BLUE-GREEN sequence./
                                GPIO_PORTF_DATA_R = 2 << i;
                                timer0A_delayMs(1000);
                                GPIO_PORTF_DATA_R = 0x00;
                                timer0A_delayMs(1000);
                        }
                }
   }
}

//Initializes timer0
//Used for delay
void timer0_InIt(void) {

        SYSCTL_RCGCTIMER_R |= 0x01;          /* enable clock to Timer0 */

        TIMER0_CTL_R = 0x00;             /* disable Timer before initialization */
        TIMER0_CFG_R = 0x04;                    /* 16-bit option */
        TIMER0_TAMR_R = 0x02;                   /* periodic mode and down-counter */
        TIMER0_TAILR_R = 50000 - 1;             /* Timer A interval load value register */
        TIMER0_ICR_R = 0x01;             /* clear the TimerA timeout flag*/
```

```
	TIMER0_CTL_R |= 0x01;			/* enable Timer A after initialization */
	TIMER0_TAPR_R = 0;                                                  // Prescalar
value.. Can extend the cycle time max 255 times
}

// Subroutine to initialize Port F
void PortF_Init(void) {

	volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000020;     // F clock
  delay = SYSCTL_RCGC2_R;           // reading register adds a delay
  GPIO_PORTF_LOCK_R = 0x4C4F434B;   // unlock PortF
  GPIO_PORTF_CR_R = 0x0E;           // allow changes to PF3-PF1
  GPIO_PORTF_AMSEL_R = 0x00;        // disable analog function
  GPIO_PORTF_PCTL_R = 0x00000000;   // GPIO clear bit PCTL
  GPIO_PORTF_DIR_R = 0x0E;          // PF3,PF2,PF1 output
  GPIO_PORTF_AFSEL_R = 0x00;        // no alternate function
  GPIO_PORTF_PUR_R = 0x00;          // disable pull-up resistors
  GPIO_PORTF_DEN_R = 0x0E;          // enable digital pins PF3-PF1
}

// Subroutine to initialize UART5
void HC05_Init(void) {

	SYSCTL_RCGC2_R |= 0x10;// Enable clock for PORTE
	SYSCTL_RCGCUART_R |= 0x20; //Enable clock for UART5

	timer0A_delayMs(1);

	GPIO_PORTE_AMSEL_R = 0x00;    // Disable analog function
	GPIO_PORTE_DEN_R = 0x30;      // PE4, PE5 are digital pins
	GPIO_PORTE_AFSEL_R = 0x30;    // Enable alternate functions for PE4, PE5
	GPIO_PORTE_PCTL_R = 0x00110000;      // PE4 = UART5Rx, PE5 = UART5Tx

	UART5_CTL_R = 0x00;       //Disable UART5 module
	UART5_IBRD_R = 325;       // For 9600 baud rate with f = 50MHz, integer = 325
	UART5_FBRD_R = 33;        // For 9600 baud rate with f = 50MHz, fraction = 33
	UART5_CC_R = 0x00;        // Select system clock
	UART5_LCRH_R = 0x60;      // 8-bit, no parity, 1 stop bit, no FIFO
	UART5_CTL_R = 0x301;      // Enable Rx, Tx, and UART5
}

// Receives and reads char from Rx FIFO
char BLT_Read(void){
```

```
	char data;
	while((UART5_FR_R & 0x10) != 0);  // Wait until Rx buffer is not full
	data = UART5_DR_R;		// Copy Rx data to local data variable
	return (unsigned char) data;	// Return data
}

// Writes a char to Tx FIFO to send
void BLT_Write(unsigned char data){

	while((UART5_FR_R & 0x20) != 0);  // Wait until Tx buffer is not full
	UART5_DR_R = data;		// Copy data into Tx
}

// Writes a string to Tx FIFO by repeatedly calling BLT_Write for each char in the string
void BLT_Write_Str(char* s) {

	// Writes string's contents to Tx buffer, one-by-one
	while(*s) {
		BLT_Write(*(s++));
	}
}

/* multiple of millisecond delay using periodic mode */
void timer0A_delayMs(int ttime)
{
	timer0_InIt();
	int i;

	for(i = 0; i < ttime; i++) {
		while ((TIMER0_RIS_R & 0x01) == 0);	/* wait for TimerA timeout flag */
		TIMER0_ICR_R = 0x01;	/* clear the TimerA timeout flag */
	}
}
```

Screenshots:

```
1  // Riley Ruckman
2  // TCES 460, Wi21
3  // Lab 10 - Part a
4
5  //Create definitions of PORTF registers
6  #define GPIO_PORTF_DATA_R       (*((volatile unsigned long *)0x400253FC))
7  #define GPIO_PORTF_DIR_R        (*((volatile unsigned long *)0x40025400))
8  #define GPIO_PORTF_AFSEL_R      (*((volatile unsigned long *)0x40025420))
9  #define GPIO_PORTF_PUR_R        (*((volatile unsigned long *)0x40025510))
10 #define GPIO_PORTF_DEN_R        (*((volatile unsigned long *)0x4002551C))
11 #define GPIO_PORTF_LOCK_R       (*((volatile unsigned long *)0x40025520))
12 #define GPIO_PORTF_CR_R         (*((volatile unsigned long *)0x40025524))
13 #define GPIO_PORTF_AMSEL_R      (*((volatile unsigned long *)0x40025528))
14 #define GPIO_PORTF_PCTL_R       (*((volatile unsigned long *)0x4002552C))
15 #define SYSCTL_RCGC2_R          (*((volatile unsigned long *)0x400FE108))
16
17 //Create definitions for PORTE registers
18 #define GPIO_PORTE_DATA_R       (*((volatile unsigned long *)0x400243FC))
19 #define GPIO_PORTE_DIR_R        (*((volatile unsigned long *)0x40024400))
20 #define GPIO_PORTE_AFSEL_R      (*((volatile unsigned long *)0x40024420))
21 #define GPIO_PORTE_PUR_R        (*((volatile unsigned long *)0x40024510))
22 #define GPIO_PORTE_DEN_R        (*((volatile unsigned long *)0x4002451C))
23 #define GPIO_PORTE_LOCK_R       (*((volatile unsigned long *)0x40024520))
24 #define GPIO_PORTE_CR_R         (*((volatile unsigned long *)0x40024524))
25 #define GPIO_PORTE_AMSEL_R      (*((volatile unsigned long *)0x40024528))
26 #define GPIO_PORTE_PCTL_R       (*((volatile unsigned long *)0x4002452C))
27
28 //Create definitions for UART5 registers
29 #define SYSCTL_RCGCUART_R       (*((volatile unsigned long *)0x400FE618))
30 #define UART5_IBRD_R            (*((volatile unsigned long *)0x40011024))
31 #define UART5_FBRD_R            (*((volatile unsigned long *)0x40011028))
32 #define UART5_CTL_R             (*((volatile unsigned long *)0x40011030))
33 #define UART5_LCRH_R            (*((volatile unsigned long *)0x4001102C))
34 #define UART5_DR_R              (*((volatile unsigned long *)0x40011000))
35 #define UART5_FR_R              (*((volatile unsigned long *)0x40011018))
36 #define UART5_CC_R              (*((volatile unsigned long *)0x40011FC8))
37
38 //Create definitions for timer0 registers
39 #define TIMER0_CFG_R            (*((volatile unsigned long *)0x40030000))
40 #define TIMER0_TAMR_R           (*((volatile unsigned long *)0x40030004))
41 #define TIMER0_CTL_R            (*((volatile unsigned long *)0x4003000C))
42 #define TIMER0_RIS_R            (*((volatile unsigned long *)0x4003001C))
43 #define TIMER0_ICR_R            (*((volatile unsigned long *)0x40030024))
44 #define TIMER0_TAILR_R          (*((volatile unsigned long *)0x40030028))
45 #define TIMER0_TAPR_R           (*((volatile unsigned long *)0x40030038))
46 #define SYSCTL_RCGCTIMER_R      (*((volatile unsigned long *)0x400FE604))
47
48 // Function Prototypes
49 void timer0A_delayMs(int ttime);
```

```c
void timer0_InIt(void);
void PortF_Init(void);
void HC05_Init(void);
char BLT_Read(void);
void BLT_Write(unsigned char data);
void BLT_Write_Str(char *s);

int main(void)
{
    // Initialize PORTF and UART5 ports/features
    PortF_Init();
    HC05_Init();

    while(1)
    {
      // Reads new data from Rx FIFO
      char c = BLT_Read();

      // If 'A' is received, a message is sent back and the onboard LED blinks RED twice at 1 Hz.
      if(c == 'A') {
        BLT_Write_Str("Hello World!! RED LED ON\n");
        for (int i = 0; i < 2; i++) {
          GPIO_PORTF_DATA_R = 0x02;
          timer0A_delayMs(1000);
          GPIO_PORTF_DATA_R = 0x00;
          timer0A_delayMs(1000);
        }
      // If 'B' is received, a message is sent back and the onboard LED blinks BLUE twice at 1 Hz.
      } else if(c == 'B') {
        BLT_Write_Str("Hello World!! BLUE LED ON\n");
        for (int i = 0; i < 2; i++) {
          GPIO_PORTF_DATA_R = 0x04;
          timer0A_delayMs(1000);
          GPIO_PORTF_DATA_R = 0x00;
          timer0A_delayMs(1000);
        }
      // If 'C' is received, a message is sent back and the onboard LED blinks RED-BLUE-GREEN at 1 Hz.
      } else if(c == 'C') {
        BLT_Write_Str("BYE World!! RGB LEDs ON\n");
        for (int i = 0; i < 3; i++) {
          // Since RED, BLUE, and GREEN are 0x2, 0x4, and 0x8, respectively, shifting
          // 2 by i will give the correct values for a RED-BLUE-GREEN sequence./
          GPIO_PORTF_DATA_R = 2 << i;
          timer0A_delayMs(1000);
          GPIO_PORTF_DATA_R = 0x00;
          timer0A_delayMs(1000);
        }
      }
```

```
 98 146     UART5_IBRD_R = 325; // For 9600 baud rate with f = 50MHz, integer = 325
 99 147     UART5_FBRD_R = 33;  // For 9600 baud rate with f = 50MHz, fraction = 33
100 148     UART5_CC_R = 0x00;  // Select system clock
101 149     UART5_LCRH_R = 0x60;  // 8-bit, no parity, 1 stop bit, no FIFO
102 150     UART5_CTL_R = 0x301;  // Enable Rx, Tx, and UART5
103 151   }
104 152
105 153   // Receives and reads char from Rx FIFO
106 154   char BLT_Read(void){
107 155
108 156     char data;
109 157     while((UART5_FR_R & 0x10) != 0);  // Wait until Rx buffer is not full
110 158     data = UART5_DR_R;  // Copy Rx data to local data variable
111 159     return (unsigned char) data;  // Return data
112 160   }
113 161
114 162   // Writes a char to Tx FIFO to send                                                    mes
115 163   void BLT_Write(unsigned char data){
116 164
117 165     while((UART5_FR_R & 0x20) != 0);  // Wait until Tx buffer is not full
118 166     UART5_DR_R = data;  // Copy data into Tx
119 167   }
120 168
121 169   // Writes a string to Tx FIFO by repeatedly calling BLT_Write for each char in the string
122 170   void BLT_Write_Str(char* s) {
123 171
124 172     // Writes string's contents to Tx buffer, one-by-one
125 173     while(*s) {
126 174       BLT_Write(*(s++));
127 175     }
128 176   }
129 177
130 178   /* multiple of millisecond delay using periodic mode */
131 179   void timer0A_delayMs(int ttime)
132 180   {
133 181     timer0_InIt();
134 182     int i;
135 183
136 184     for(i = 0; i < ttime; i++) {
137 185       while ((TIMER0_RIS_R & 0x01) == 0);      /* wait for TimerA timeout flag */
138 186       TIMER0_ICR_R = 0x01;      /* clear the TimerA timeout flag */
139 187     }
140 188   }
141 189
    GPIO_PORTE_AMSEL_R = 0x00;  // Disable analog function
    GPIO_PORTE_DEN_R = 0x30;  // PE4, PE5 are digital pins
    GPIO_PORTE_AFSEL_R = 0x30;  // Enable alternate functions for PE4, PE5
    GPIO_PORTE_PCTL_R = 0x00110000; // PE4 = UART5Rx, PE5 = UART5Tx

    UART5_CTL_R = 0x00; //Disable UART5 module
```

Part b:

Code:

```
// Riley Ruckman
// TCES 460, Wi21
// Lab 10 - Part b

#include <stdint.h>
#include <stdio.h>

//////////////////////////////////////////////////////////////////

/*

                                                    */
/*
                ADC
                                                */
/*

                                                    */

//Create definitions for ADC0 registers
#define SYSCTL_RCGCADC_R            (*((volatile unsigned long *)0x400FE638))
#define ADC0_ACTSS_R                    (*((volatile unsigned long *)0x40038000))
#define ADC0_EMUX_R                     (*((volatile unsigned long *)0x40038014))
#define ADC0_SSMUX3_R               (*((volatile unsigned long *)0x400380A0))
#define ADC0_SSCTL3_R               (*((volatile unsigned long *)0x400380A4))
#define ADC0_PSSI_R                     (*((volatile unsigned long *)0x40038028))
#define ADC0_RIS_R                          (*((volatile unsigned long
*)0x40038004))
#define ADC0_SSFIFO3_R              (*((volatile unsigned long *)0x400380A8))
#define ADC0_ISC_R                          (*((volatile unsigned long
*)0x4003800C))
#define ADC0_PC_R       (*((volatile unsigned long *)0x40038FC4))

#define ADC0_IM_R       (*((volatile unsigned long *)0x40038008))
#define NVIC_EN0_R      (*((volatile unsigned long *)0xE000E100))

#define NVIC_PRI4_R     (*((volatile unsigned long *)0xE000E410))

//Create definitions for PORT D registers
#define SYSCTL_RCGC2_R              (*((volatile unsigned long *)0x400FE108))
```

```
#define GPIO_PORTD_DATA_R     (*((volatile unsigned long *)0x400073FC))
#define GPIO_PORTD_DIR_R            (*((volatile unsigned long *)0x40007400))
#define GPIO_PORTD_AFSEL_R    (*((volatile unsigned long *)0x40007420))
#define GPIO_PORTD_PUR_R            (*((volatile unsigned long *)0x40007510))
#define GPIO_PORTD_DEN_R            (*((volatile unsigned long *)0x4000751C))
#define GPIO_PORTD_LOCK_R     (*((volatile unsigned long *)0x40007520))
#define GPIO_PORTD_CR_R             (*((volatile unsigned long *)0x40007524))
#define GPIO_PORTD_AMSEL_R    (*((volatile unsigned long *)0x40007528))
#define GPIO_PORTD_PCTL_R     (*((volatile unsigned long *)0x4000752C))

//Create definitions for Timer1 registers
#define TIMER1_CFG_R                  (*((volatile unsigned long *)0x40031000))
#define TIMER1_TAMR_R           (*((volatile unsigned long *)0x40031004))
#define TIMER1_CTL_R                  (*((volatile unsigned long *)0x4003100C))
#define TIMER1_RIS_R                  (*((volatile unsigned long *)0x4003101C))
#define TIMER1_ICR_R                  (*((volatile unsigned long *)0x40031024))
#define TIMER1_TAILR_R          (*((volatile unsigned long *)0x40031028))
#define TIMER1_TAPR_R           (*((volatile unsigned long *)0x40031038))

//////////////////////////////////////////////////////////////

/*

                                              */
/*
            UART
                        */
/*

                                              */

//Create definitions for UART5 registers
#define SYSCTL_RCGCUART_R     (*((volatile unsigned long *)0x400FE618))
#define UART5_IBRD_R        (*((volatile unsigned long *)0x40011024))
#define UART5_FBRD_R        (*((volatile unsigned long *)0x40011028))
#define UART5_CTL_R         (*((volatile unsigned long *)0x40011030))
#define UART5_LCRH_R        (*((volatile unsigned long *)0x4001102C))
#define UART5_DR_R          (*((volatile unsigned long *)0x40011000))
#define UART5_FR_R          (*((volatile unsigned long *)0x40011018))
#define UART5_CC_R          (*((volatile unsigned long *)0x40011FC8))

//Create definitions for Port E registers
#define GPIO_PORTE_DATA_R     (*((volatile unsigned long *)0x400243FC))
#define GPIO_PORTE_DIR_R            (*((volatile unsigned long *)0x40024400))
```

```
#define GPIO_PORTE_PUR_R        (*((volatile unsigned long *)0x40024510))
#define GPIO_PORTE_DEN_R        (*((volatile unsigned long *)0x4002451C))
#define GPIO_PORTE_CR_R         (*((volatile unsigned long *)0x40024524))
#define GPIO_PORTE_AMSEL_R    (*((volatile unsigned long *)0x40024528))
#define GPIO_PORTE_AFSEL_R    (*((volatile unsigned long *)0x40024420))
#define GPIO_PORTE_PCTL_R     (*((volatile unsigned long *)0x4002452C))

//////////////////////////////////////////////////////////////////

/*

                                                */
/*
            LEDs
                                */
/*

                                                */

//Create definitions of PORTF registers
#define GPIO_PORTF_DATA_R     (*((volatile unsigned long *)0x400253FC))
#define GPIO_PORTF_DIR_R      (*((volatile unsigned long *)0x40025400))
#define GPIO_PORTF_AFSEL_R    (*((volatile unsigned long *)0x40025420))
#define GPIO_PORTF_PUR_R      (*((volatile unsigned long *)0x40025510))
#define GPIO_PORTF_DEN_R      (*((volatile unsigned long *)0x4002551C))
#define GPIO_PORTF_LOCK_R     (*((volatile unsigned long *)0x40025520))
#define GPIO_PORTF_CR_R       (*((volatile unsigned long *)0x40025524))
#define GPIO_PORTF_AMSEL_R    (*((volatile unsigned long *)0x40025528))
#define GPIO_PORTF_PCTL_R     (*((volatile unsigned long *)0x4002552C))

//////////////////////////////////////////////////////////////////

/*

                                                */
/*
Normal-Use Timer
        */
/*

                                                */

//Create definitions for Timer0 registers
#define TIMER0_CFG_R                    (*((volatile unsigned long *)0x40030000))
```

```
#define TIMER0_TAMR_R                   (*((volatile unsigned long *)0x40030004))
#define TIMER0_CTL_R                        (*((volatile unsigned long *)0x4003000C))
#define TIMER0_RIS_R                        (*((volatile unsigned long *)0x4003001C))
#define TIMER0_ICR_R                        (*((volatile unsigned long *)0x40030024))
#define TIMER0_TAILR_R                  (*((volatile unsigned long *)0x40030028))
#define TIMER0_TAPR_R                   (*((volatile unsigned long *)0x40030038))
#define SYSCTL_RCGCTIMER_R    (*((volatile unsigned long *)0x400FE604))

//////////////////////////////////////////////////////////////////

/*

                                                */
/*                      Function Prototypes & Global Variable Initialization                    */
/*

                                                */

void ADC_Init(void);
void timer0A_delayMs(int ttime);
void timer0_Init(void);
void ADC0SS3_Handler(void);
void PortF_Init(void);
void HC05_Init(void);
char BLT_Read(void);
void BLT_Write(unsigned char data);
void BLT_Write_Str(char *s);
static int temperature = 0;
static int oldTemperature;

//////////////////////////////////////////////////////////////////

int main(void){

        // Initialize necessary ports/modules
        ADC_Init();
        HC05_Init();
        PortF_Init();
        while(1){

                // Reads new data from Rx FIFO
                char c = BLT_Read();

                // If read data from UART is 'A'
```

```
                if(c == 'A') {
                        // Converts temperature into str then writes it to the Tx FIFO
                        char stemp[5];
                        sprintf(stemp, "%d", temperature);
                        BLT_Write_Str("Current Temperature is: ");
                        BLT_Write_Str(stemp);

                        // Calculates temperature difference between new and previous reading
                        // and converts it into a str
                        int tempDifference = temperature - oldTemperature;
                        char stempDif[5];
                        sprintf(stempDif, "%d", abs(tempDifference));          // Takes absolute
value of difference for sending to user

                        // Based off the temperature difference, an unique message will be sent
                        if (tempDifference > 0) {
                                BLT_Write_Str("\nTemperature increase: ");
                                BLT_Write_Str(stempDif);
                        } else if (tempDifference < 0) {
                                BLT_Write_Str("\nTemperature decrease: ");
                                BLT_Write_Str(stempDif);
                        } else {
                                BLT_Write_Str("\nTemperature unchanged");
                        }
                        BLT_Write_Str("\n");
                }
        }
}

// Interrupt Handler for ADC0SS3 interrupt
void ADC0SS3_Handler(void) {

        oldTemperature = temperature;
        // temperature = ((ADC0_SSFIFO3_R & 0xFFF) - 500)/10;
        // Converts 12-bit conversion result in SS3's FIFO to Celsius, and then Fahrenheit
        temperature = (int)((((((ADC0_SSFIFO3_R & 0xFFF) - 500) / 10) * 1000) * (9.0/5.0)) +
32);

        // If the temperature difference is at most -2, then the LED blinks RED twice at 1 Hz.
        // This is to simulate a signal being sent to a heater to turn on
        if ((temperature - oldTemperature) <= -2) {
                        for (int i = 0; i < 2; i++) {
                        GPIO_PORTF_DATA_R = 0x02;
                        timer0A_delayMs(1000);
```

```
                GPIO_PORTF_DATA_R = 0x00;
                timer0A_delayMs(1000);
            }
        // If the temperature difference is at least 2, then the LED blinks GREEN twice at 1 Hz.
        // This is to simulate a signal being sent to a cooler to turn on
        } else if ((temperature - oldTemperature) >= 2) {
            for (int i = 0; i < 2; i++) {
                GPIO_PORTF_DATA_R = 0x08;
                timer0A_delayMs(1000);
                GPIO_PORTF_DATA_R = 0x00;
                timer0A_delayMs(1000);
            }
        }

        // Clears flag that were set due to ADC0 conversion
        // completion and TIMER0 timeout condition, respectively
        ADC0_ISC_R |= 0x8; /* clear completion flag */
        TIMER1_ICR_R = 0x01; /* clear the TimerA timeout flag */
}

/* multiple of millisecond delay using periodic mode */
void timer0_Init(void){

        SYSCTL_RCGCTIMER_R |= 0x01; /* enable clock to Timer0 */
        TIMER0_CTL_R = 0x00; /* disable Timer before initialization */
        TIMER0_CFG_R = 0x04; /* 16-bit option */
        TIMER0_TAMR_R = 0x02; /* periodic mode and down-counter */
        TIMER0_TAILR_R = 50000 - 1; /* Timer A interval load value register */
        TIMER0_ICR_R = 0x1; /* clear the TimerA timeout flag*/
        TIMER0_CTL_R |= 0x01; /* enable Timer A after initialization */
        TIMER0_TAPR_R = 0; // Prescalar value.. Can extend the cycle time max 256 times
}

// Subroutine to initialize Port F
void PortF_Init(void) {

        volatile unsigned long delay;
  SYSCTL_RCGC2_R |= 0x00000020;     // F clock
  delay = SYSCTL_RCGC2_R;           // reading register adds a delay
  GPIO_PORTF_LOCK_R = 0x4C4F434B;   // unlock PortF
  GPIO_PORTF_CR_R = 0x0E;           // allow changes to PF3-PF1
  GPIO_PORTF_AMSEL_R = 0x00;        // disable analog function
  GPIO_PORTF_PCTL_R = 0x00000000;   // GPIO clear bit PCTL
  GPIO_PORTF_DIR_R = 0x0E;          // PF3,PF2,PF1 output
```

```
  GPIO_PORTF_AFSEL_R = 0x00;       // no alternate function
  GPIO_PORTF_PUR_R = 0x00;         // disable pull-up resistors
  GPIO_PORTF_DEN_R = 0x0E;         // enable digital pins PF3-PF1
}

// Initialize ADC0 module and PORTD for ADC use
void ADC_Init(void){

        SYSCTL_RCGC2_R |= 0x08;
        SYSCTL_RCGCADC_R |= 0x01; // Enable clock ADC0

        /* initialize PD1 for AIN6 input */
        GPIO_PORTD_AFSEL_R |= 0x2; /* enable alternate function */
        GPIO_PORTD_DEN_R &= ~0x2; /* disable digital function */
        GPIO_PORTD_AMSEL_R |= 0x2; /* enable analog function */

        /* initialize ADC0 */
        ADC0_ACTSS_R &= ~0x8; /* disable SS3 during configuration */
        ADC0_EMUX_R = 0x5000; /* timer trigger conversion */
        ADC0_SSMUX3_R = 0x6; /* get input from channel 6 */
        ADC0_SSCTL3_R |= 0x6; /* take one sample at a time, set flag at 1st sample */
        ADC0_PC_R = 0x00;                   /* sets sampling rate to 125 kHz */

        ADC0_IM_R |= 0x08; /* enables interrupt mask for SS3 in ADC0 */
        NVIC_EN0_R = 0x00020000; /*  Enable interrupt 17 in NVIC */
        NVIC_PRI4_R = (NVIC_PRI4_R & 0xFFFF0FFF) | 0x00004000; /*  priority 2 */
        ADC0_ACTSS_R |= 0x8; /* enable ADC0 sequencer 3 */

        ADC0_PSSI_R = 0x01;         // Start conversion

        // Initialize Timer1
        SYSCTL_RCGCTIMER_R |= 0x02;     /* enable clock to Timer1 */

        TIMER1_CTL_R = 0x00;          /* disable Timer before initialization */
        TIMER1_CFG_R = 0x00;                    /* 32-bit option */
        TIMER1_TAMR_R = 0x02;                   /* periodic mode and down-counter */
        TIMER1_TAILR_R = 500000000 - 1; /* Timer A interval load value register */
        TIMER1_ICR_R = 0x1;                 /* clear the Timer A timeout flag*/
        TIMER1_TAPR_R = 0;                                                      // Prescalar
value.. Can extend the cycle time max 255 times

        TIMER1_CTL_R |= 0x21;                   /* enable Timer A after initialization */
}
```

```
// Subroutine to initialize UART5
void HC05_Init(void) {

	SYSCTL_RCGC2_R |= 0x10;// Enable clock for PORTE
	SYSCTL_RCGCUART_R |= 0x20; //Enable clock for UART5

	timer0A_delayMs(1);

	GPIO_PORTE_AMSEL_R = 0x00;    // Disable analog function
	GPIO_PORTE_DEN_R = 0x30;      // PE4, PE5 are digital pins
	GPIO_PORTE_AFSEL_R = 0x30;    // Enable alternate functions for PE4, PE5
	GPIO_PORTE_PCTL_R = 0x00110000;      // PE4 = UART5Rx, PE5 = UART5Tx

	UART5_CTL_R = 0x00;       //Disable UART5 module
	UART5_IBRD_R = 325;       // For 9600 baud rate with f = 50MHz, integer = 325
	UART5_FBRD_R = 33;        // For 9600 baud rate with f = 50MHz, fraction = 33
	UART5_CC_R = 0x00;        // Select system clock
	UART5_LCRH_R = 0x60;      // 8-bit, no parity, 1 stop bit, no FIFO
	UART5_CTL_R = 0x301;      // Enable Rx, Tx, and UART5
}

// Receives and reads char from Rx FIFO
char BLT_Read(void){

	char data;
	while((UART5_FR_R & 0x10) != 0);  // Wait until Rx buffer is not full
	data = UART5_DR_R;          // Copy Rx data to local data variable
	return (unsigned char) data;  // Return data
}

// Writes a char to Tx FIFO to send
void BLT_Write(unsigned char data){

	while((UART5_FR_R & 0x20) != 0);  // Wait until Tx buffer is not full
	UART5_DR_R = data;          // Copy data into Tx
}

// Writes a string to Tx FIFO by repeatedly calling BLT_Write for each char in the string
void BLT_Write_Str(char* s) {

	// Writes string's contents to Tx buffer, one-by-one
	while(*s) {
		BLT_Write(*(s++));
	}
```

```
}

/* multiple of millisecond delay using periodic mode */
void timer0A_delayMs(int ttime)
{
        timer0_Init();
        int i;

        for(i = 0; i < ttime; i++) {
                while ((TIMER0_RIS_R & 0x01) == 0);      /* wait for TimerA timeout flag */
                TIMER0_ICR_R = 0x01;      /* clear the TimerA timeout flag */
        }
}
```

Screenshots:

```
1  // Riley Ruckman
2  // TCES 460, Wi21
3  // Lab 10 - Part b
4
5  #include <stdint.h>
6  #include <stdio.h>
7
8  ////////////////////////////////////////////////////////////////////
9
10 /*                                                                */
11 /*                            ADC                                 */
12 /*                                                                */
13
14 //Create definitions for ADC0 registers
15 #define SYSCTL_RCGCADC_R     (*((volatile unsigned long *)0x400FE638))
16 #define ADC0_ACTSS_R         (*((volatile unsigned long *)0x40038000))
17 #define ADC0_EMUX_R          (*((volatile unsigned long *)0x40038014))
18 #define ADC0_SSMUX3_R        (*((volatile unsigned long *)0x400380A0))
19 #define ADC0_SSCTL3_R        (*((volatile unsigned long *)0x400380A4))
20 #define ADC0_PSSI_R          (*((volatile unsigned long *)0x40038028))
21 #define ADC0_RIS_R           (*((volatile unsigned long *)0x40038004))
22 #define ADC0_SSFIFO3_R       (*((volatile unsigned long *)0x400380A8))
23 #define ADC0_ISC_R           (*((volatile unsigned long *)0x4003800C))
24 #define ADC0_PC_R            (*((volatile unsigned long *)0x40038FC4))
25
26 #define ADC0_IM_R            (*((volatile unsigned long *)0x40038008))
27 #define NVIC_EN0_R           (*((volatile unsigned long *)0xE000E100))
28 #define NVIC_PRI4_R          (*((volatile unsigned long *)0xE000E410))
29
30 //Create definitions for PORT D registers
31 #define SYSCTL_RCGC2_R       (*((volatile unsigned long *)0x400FE108))
32 #define GPIO_PORTD_DATA_R    (*((volatile unsigned long *)0x400073FC))
33 #define GPIO_PORTD_DIR_R     (*((volatile unsigned long *)0x40007400))
34 #define GPIO_PORTD_AFSEL_R   (*((volatile unsigned long *)0x40007420))
35 #define GPIO_PORTD_PUR_R     (*((volatile unsigned long *)0x40007510))
36 #define GPIO_PORTD_DEN_R     (*((volatile unsigned long *)0x4000751C))
37 #define GPIO_PORTD_LOCK_R    (*((volatile unsigned long *)0x40007520))
38 #define GPIO_PORTD_CR_R      (*((volatile unsigned long *)0x40007524))
39 #define GPIO_PORTD_AMSEL_R   (*((volatile unsigned long *)0x40007528))
40 #define GPIO_PORTD_PCTL_R    (*((volatile unsigned long *)0x4000752C))
41
42 //Create definitions for Timer1 registers
43 #define TIMER1_CFG_R         (*((volatile unsigned long *)0x40031000))
44 #define TIMER1_TAMR_R        (*((volatile unsigned long *)0x40031004))
45 #define TIMER1_CTL_R         (*((volatile unsigned long *)0x4003100C))
46 #define TIMER1_RIS_R         (*((volatile unsigned long *)0x4003101C))
47 #define TIMER1_ICR_R         (*((volatile unsigned long *)0x40031024))
48 #define TIMER1_TAILR_R       (*((volatile unsigned long *)0x40031028))
49 #define TIMER1_TAPR_R        (*((volatile unsigned long *)0x40031038))
```

```
50
51  ////////////////////////////////////////////////////////////////////
52
53  /*                                                                */
54  /*                              UART                              */
55  /*                                                                */
56
57  //Create definitions for UART5 registers
58  #define SYSCTL_RCGCUART_R       (*((volatile unsigned long *)0x400FE618))
59  #define UART5_IBRD_R            (*((volatile unsigned long *)0x40011024))
60  #define UART5_FBRD_R            (*((volatile unsigned long *)0x40011028))
61  #define UART5_CTL_R             (*((volatile unsigned long *)0x40011030))
62  #define UART5_LCRH_R            (*((volatile unsigned long *)0x4001102C))
63  #define UART5_DR_R              (*((volatile unsigned long *)0x40011000))
64  #define UART5_FR_R              (*((volatile unsigned long *)0x40011018))
65  #define UART5_CC_R              (*((volatile unsigned long *)0x40011FC8))
66
67  //Create definitions for Port E registers
68  #define GPIO_PORTE_DATA_R   (*((volatile unsigned long *)0x400243FC))
69  #define GPIO_PORTE_DIR_R    (*((volatile unsigned long *)0x40024400))
70  #define GPIO_PORTE_PUR_R    (*((volatile unsigned long *)0x40024510))
71  #define GPIO_PORTE_DEN_R    (*((volatile unsigned long *)0x4002451C))
72  #define GPIO_PORTE_CR_R     (*((volatile unsigned long *)0x40024524))
73  #define GPIO_PORTE_AMSEL_R  (*((volatile unsigned long *)0x40024528))
74  #define GPIO_PORTE_AFSEL_R  (*((volatile unsigned long *)0x40024420))
75  #define GPIO_PORTE_PCTL_R   (*((volatile unsigned long *)0x4002452C))
76
77  ////////////////////////////////////////////////////////////////////
78
79  /*                                                                */
80  /*                              LEDs                              */
81  /*                                                                */
82
83  //Create definitions of PORTF registers
84  #define GPIO_PORTF_DATA_R       (*((volatile unsigned long *)0x400253FC))
85  #define GPIO_PORTF_DIR_R        (*((volatile unsigned long *)0x40025400))
86  #define GPIO_PORTF_AFSEL_R      (*((volatile unsigned long *)0x40025420))
87  #define GPIO_PORTF_PUR_R        (*((volatile unsigned long *)0x40025510))
88  #define GPIO_PORTF_DEN_R        (*((volatile unsigned long *)0x4002551C))
89  #define GPIO_PORTF_LOCK_R       (*((volatile unsigned long *)0x40025520))
90  #define GPIO_PORTF_CR_R         (*((volatile unsigned long *)0x40025524))
91  #define GPIO_PORTF_AMSEL_R      (*((volatile unsigned long *)0x40025528))
92  #define GPIO_PORTF_PCTL_R       (*((volatile unsigned long *)0x4002552C))
93
94  ////////////////////////////////////////////////////////////////////
95
96  /*                                                                */
97  /*                        Normal-Use Timer                        */
```

```
 98  /*                                                                      */
 99
100  //Create definitions for Timer0 registers
101  #define TIMER0_CFG_R         (*((volatile unsigned long *)0x40030000))
102  #define TIMER0_TAMR_R        (*((volatile unsigned long *)0x40030004))
103  #define TIMER0_CTL_R         (*((volatile unsigned long *)0x4003000C))
104  #define TIMER0_RIS_R         (*((volatile unsigned long *)0x4003001C))
105  #define TIMER0_ICR_R         (*((volatile unsigned long *)0x40030024))
106  #define TIMER0_TAILR_R       (*((volatile unsigned long *)0x40030028))
107  #define TIMER0_TAPR_R        (*((volatile unsigned long *)0x40030038))
108  #define SYSCTL_RCGCTIMER_R   (*((volatile unsigned long *)0x400FE604))
109
110  ////////////////////////////////////////////////////////////////////////
111
112  /*                                                                      */
113  /*      Function Prototypes & Global Variable Initialization            */
114  /*                                                                      */
115
116  void ADC_Init(void);
117  void timer0A_delayMs(int ttime);
118  void timer0_Init(void);
119  void ADC0SS3_Handler(void);
120  void PortF_Init(void);
121  void HC05_Init(void);
122  char BLT_Read(void);
123  void BLT_Write(unsigned char data);
124  void BLT_Write_Str(char *s);
125
126  static int temperature = 0;
127  static int oldTemperature;
128
129  ////////////////////////////////////////////////////////////////////////
130
131  int main(void){
132
133    // Initialize necessary ports/modules
134    ADC_Init();
135    HC05_Init();
136    PortF_Init();
137    while(1){
138
139      // Reads new data from Rx FIFO
140      char c = BLT_Read();
141
142      // If read data from UART is 'A'
143      if(c == 'A') {
144        // Converts temperature into str then writes it to the Tx FIFO
145        char stemp[5];
```

```
146         sprintf(stemp, "%d", temperature);
147         BLT_Write_Str("Current Temperature is: ");
148         BLT_Write_Str(stemp);
149
150         // Calculates temperature difference between new and previous reading
151         // and converts it into a str
152         int tempDifference = temperature - oldTemperature;
153         char stempDif[5];
154         sprintf(stempDif, "%d", abs(tempDifference)); // Takes absolute value of difference for sending to user
155
156         // Based off the temperature difference, an unique message will be sent
157         if (tempDifference > 0) {
158           BLT_Write_Str("\nTemperature increase: ");
159           BLT_Write_Str(stempDif);
160         } else if (tempDifference < 0) {
161           BLT_Write_Str("\nTemperature decrease: ");
162           BLT_Write_Str(stempDif);
163         } else {
164           BLT_Write_Str("\nTemperature unchanged");
165         }
166         BLT_Write_Str("\n");
167       }
168     }
169   }
170
171   // Interrupt Handler for ADC0SS3 interrupt
172   void ADC0SS3_Handler(void) {
173
174     oldTemperature = temperature;
175     // temperature = ((ADC0_SSFIFO3_R & 0xFFF) - 500)/10;
176     // Converts 12-bit conversion result in SS3's FIFO to Celsius, and then Fahrenheit
177     temperature = (int)((((((ADC0_SSFIFO3_R & 0xFFF) * (3.3/4096.0) * 1000.0) - 500) / 10) * (9.0/5.0)) + 32);
178
179     // If the temperature difference is at most -2, then the LED blinks RED twice at 1 Hz.
180     // This is to simulate a signal being sent to a heater to turn on
181     if ((temperature - oldTemperature) <= -2) {
182       for (int i = 0; i < 2; i++) {
183         GPIO_PORTF_DATA_R = 0x02;
184         timer0A_delayMs(1000);
185         GPIO_PORTF_DATA_R = 0x00;
186         timer0A_delayMs(1000);
187       }
188     // If the temperature difference is at least 2, then the LED blinks GREEN twice at 1 Hz.
189     // This is to simulate a signal being sent to a cooler to turn on
190     } else if ((temperature - oldTemperature) >= 2) {
191       for (int i = 0; i < 2; i++) {
192         GPIO_PORTF_DATA_R = 0x08;
193         timer0A_delayMs(1000);
```

```
194         GPIO_PORTF_DATA_R = 0x00;
195         timer0A_delayMs(1000);
196       }
197     }
198
199     // Clears flag that were set due to ADC0 conversion
200     // completion and TIMER0 timeout condition, respectively
201     ADC0_ISC_R |= 0x8; /* clear completion flag */
202     TIMER1_ICR_R = 0x01; /* clear the TimerA timeout flag */
203   }
204
205   /* multiple of millisecond delay using periodic mode */
206   void timer0_Init(void){
207
208     SYSCTL_RCGCTIMER_R |= 0x01; /* enable clock to Timer0 */
209     TIMER0_CTL_R = 0x00; /* disable Timer before initialization */
210     TIMER0_CFG_R = 0x04; /* 16-bit option */
211     TIMER0_TAMR_R = 0x02; /* periodic mode and down-counter */
212     TIMER0_TAILR_R = 50000 - 1; /* Timer A interval load value register */
213     TIMER0_ICR_R = 0x1; /* clear the TimerA timeout flag*/
214     TIMER0_CTL_R |= 0x01; /* enable Timer A after initialization */
215     TIMER0_TAPR_R = 0; // Prescalar value.. Can extend the cycle time max 256 times
216   }
217
218   // Subroutine to initialize Port F
219   void PortF_Init(void) {
220
221     volatile unsigned long delay;
222     SYSCTL_RCGC2_R |= 0x00000020;     // F clock
223     delay = SYSCTL_RCGC2_R;           // reading register adds a delay
224     GPIO_PORTF_LOCK_R = 0x4C4F434B;   // unlock PortF
225     GPIO_PORTF_CR_R = 0x0E;           // allow changes to PF3-PF1
226     GPIO_PORTF_AMSEL_R = 0x00;        // disable analog function
227     GPIO_PORTF_PCTL_R = 0x00000000;   // GPIO clear bit PCTL
228     GPIO_PORTF_DIR_R = 0x0E;          // PF3,PF2,PF1 output
229     GPIO_PORTF_AFSEL_R = 0x00;        // no alternate function
230     GPIO_PORTF_PUR_R = 0x00;          // disable pull-up resistors
231     GPIO_PORTF_DEN_R = 0x0E;          // enable digital pins PF3-PF1
232   }
233
234   // Initialize ADC0 module and PORTD for ADC use
235   void ADC_Init(void){
236
237     SYSCTL_RCGC2_R |= 0x08; // Enable clock PORTD
238     SYSCTL_RCGCADC_R |= 0x01; // Enable clock ADC0
239
240     /* initialize PD1 for AIN6 input */
241     GPIO_PORTD_AFSEL_R |= 0x2; /* enable alternate function */
```

```
  GPIO_PORTD_DEN_R &= ~0x2; /* disable digital function */
  GPIO_PORTD_AMSEL_R |= 0x2; /* enable analog function */

  /* initialize ADC0 */
  ADC0_ACTSS_R &= ~0x8; /* disable SS3 during configuration */
  ADC0_EMUX_R = 0x5000; /* timer trigger conversion */
  ADC0_SSMUX3_R = 0x6; /* get input from channel 6 */
  ADC0_SSCTL3_R |= 0x6; /* take one sample at a time, set flag at 1st sample */
  ADC0_PC_R = 0x00;     /* sets sampling rate to 125 kHz */

  ADC0_IM_R |= 0x08; /* enables interrupt mask for SS3 in ADC0 */
  NVIC_EN0_R = 0x00020000; /*  Enable interrupt 17 in NVIC */
  NVIC_PRI4_R = (NVIC_PRI4_R & 0xFFFF0FFF) | 0x00004000; /*  priority 2 */
  ADC0_ACTSS_R |= 0x8; /* enable ADC0 sequencer 3 */

  ADC0_PSSI_R = 0x01; // Start conversion

  // Initialize Timer1
  SYSCTL_RCGCTIMER_R |= 0x02;     /* enable clock to Timer1 */

  TIMER1_CTL_R = 0x00;            /* disable Timer before initialization */
  TIMER1_CFG_R = 0x00;            /* 32-bit option */
  TIMER1_TAMR_R = 0x02;           /* periodic mode and down-counter */
  TIMER1_TAILR_R = 500000000 - 1; /* Timer A interval load value register */
  TIMER1_ICR_R = 0x1;             /* clear the Timer A timeout flag*/
  TIMER1_TAPR_R = 0;              // Prescalar value.. Can extend the cycle time max 255 times

  TIMER1_CTL_R |= 0x21;           /* enable Timer A after initialization */
}

// Subroutine to initialize UART5
void HC05_Init(void) {

  SYSCTL_RCGC2_R |= 0x10; // Enable clock for PORTE
  SYSCTL_RCGCUART_R |= 0x20; //Enable clock for UART5

  timer0A_delayMs(1);

  GPIO_PORTE_AMSEL_R = 0x00;  // Disable analog function
  GPIO_PORTE_DEN_R = 0x30;  // PE4, PE5 are digital pins
  GPIO_PORTE_AFSEL_R = 0x30;  // Enable alternate functions for PE4, PE5
  GPIO_PORTE_PCTL_R = 0x00110000; // PE4 = UART5Rx, PE5 = UART5Tx

  UART5_CTL_R = 0x00; //Disable UART5 module
  UART5_IBRD_R = 325; // For 9600 baud rate with f = 50MHz, integer = 325
  UART5_FBRD_R = 33;  // For 9600 baud rate with f = 50MHz, fraction = 33
  UART5_CC_R = 0x00;  // Select system clock
  UART5_LCRH_R = 0x60;  // 8-bit, no parity, 1 stop bit, no FIFO
```

```
290     UART5_CTL_R = 0x301;  // Enable Rx, Tx, and UART5
291   }
292
293   // Receives and reads char from Rx FIFO
294   char BLT_Read(void){
295
296     char data;
297     while((UART5_FR_R & 0x10) != 0);  // Wait until Rx buffer is not full
298     data = UART5_DR_R;  // Copy Rx data to local data variable
299     return (unsigned char) data;  // Return data
300   }
301
302   // Writes a char to Tx FIFO to send
303   void BLT_Write(unsigned char data){
304
305     while((UART5_FR_R & 0x20) != 0);  // Wait until Tx buffer is not full
306     UART5_DR_R = data;  // Copy data into Tx
307   }
308
309   // Writes a string to Tx FIFO by repeatedly calling BLT_Write for each char in the string
310   void BLT_Write_Str(char* s) {
311
312     // Writes string's contents to Tx buffer, one-by-one
313     while(*s) {
314       BLT_Write(*(s++));
315     }
316   }
317
318   /* multiple of millisecond delay using periodic mode */
319   void timer0A_delayMs(int ttime)
320   {
321     timer0_Init();
322     int i;
323
324     for(i = 0; i < ttime; i++) {
325       while ((TIMER0_RIS_R & 0x01) == 0);      /* wait for TimerA timeout flag */
326       TIMER0_ICR_R = 0x01;      /* clear the TimerA timeout flag */
327     }
328   }
329
```