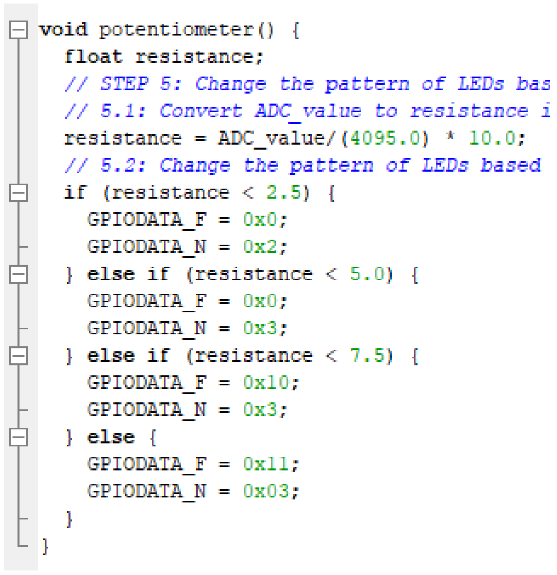
ECE/CSE 474 Lab Report 3

By Timothy Yi (2065634) and Jonathan Wang (2236141)

Task 1a

Procedure:

1. Initialize the potentiometer by inserting it on a breadboard and connecting one end to ground and the other to a 3.3V power supply.
2. Connect the middle pin in the potentiometer to a GPIO pin on the TIVA board.
3. Fill out provided template code, Lab3_Inits, to initialize ADC0, the on-board LEDs, and Timer0 to trigger ADC0 at 1 Hz. Configure any required registers to the header file.
4. Fill out the ADC0 handler function provided on the template.
5. Fill the rest of the provided template to convert the value in the ADC to resistance in kilo-ohms and light up certain LEDs based on the resistance. Shown on the right.

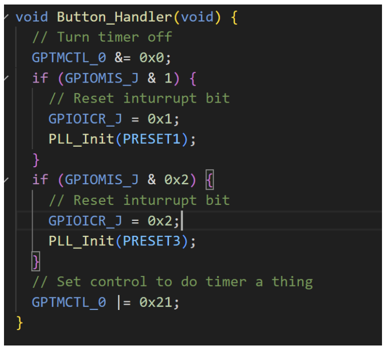


```
void potentiometer() {
  float resistance;
  // STEP 5: Change the pattern of LEDs bas
  // 5.1: Convert ADC_value to resistance i
  resistance = ADC_value/(4095.0) * 10.0;
  // 5.2: Change the pattern of LEDs based
  if (resistance < 2.5) {
    GPIODATA_F = 0x0;
    GPIODATA_N = 0x2;
  } else if (resistance < 5.0) {
    GPIODATA_F = 0x0;
    GPIODATA_N = 0x3;
  } else if (resistance < 7.5) {
    GPIODATA_F = 0x10;
    GPIODATA_N = 0x3;
  } else {
    GPIODATA_F = 0x11;
    GPIODATA_N = 0x03;
  }
}
```

Task 1b

Procedure:

1. In this case, we reused code from our Lab 2 buttons code and modified our initialization code in step 2.14 to be able to sample from the on-board thermometer.
2. Our button handler code made use of the given PLL_Init code given to us with the 3 given presets to create a code that could automatically switch between given clock states.
3. Additionally, we added a method temperature() that could printf and flush the printf buffer to the on-board terminal



```
void Button_Handler(void) {
  // Turn timer off
  GPTMCTL_0 &= 0x0;
  if (GPIOMIS_J & 1) {
    // Reset inturrupt bit
    GPIOICR_J = 0x1;
    PLL_Init(PRESET1);
  }
  if (GPIOMIS_J & 0x2) {
    // Reset inturrupt bit
    GPIOICR_J = 0x2;
    PLL_Init(PRESET3);
  }
  // Set control to do timer a thing
  GPTMCTL_0 |= 0x21;
}
```

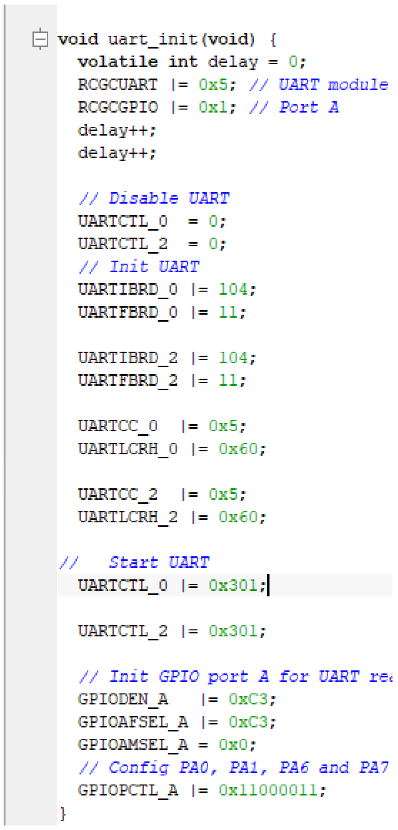
Results:

For task 1a, we were able to utilize the ADC built in the TIVA board to get the resistance values of a potentiometer and utilize the built-in LEDs to indicate the resistance. When the resistance was 0-2.5, LED 1 turned on. When the resistance was between 2.5 and 5.0, LEDs 1 and 2 would turn on. When the resistance was between 5.0 and 7.5, LEDS 1, 2 and 3 would turn on. When the resistance was 7.5 to 10.0, all the LEDs would turn on.

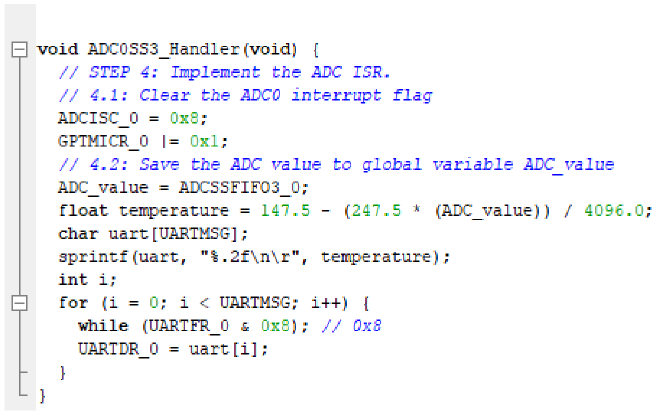
Task 2a:

Procedure:

1. Initialize UART 0 as shown on the right (This also initializes UART 2) to be able to read and write. Include any necessary registers inside the header file as well.
2. Inside the ADC0 handler function used in Task 1b, include code so that the temperature value from the ADC will be sent to PuTTY one character at a time. Shown below.
3. Go to device manager and find which COM port the “Stellaris Virtual Serial Port” is located.
4. Open up PuTTY and choose the serial connection type and change the serial line to the appropriate COM port.
5. Once everything is configured on PuTTY and it is running, run the program and verify that temperature reading are showing up on PuTTY.



```c
void uart_init(void) {
  volatile int delay = 0;
  RCGCUART  |= 0x5; // UART module
  RCGCGPIO  |= 0x1; // Port A
  delay++;
  delay++;

  // Disable UART
  UARTCTL_0   = 0;
  UARTCTL_2   = 0;
  // Init UART
  UARTIBRD_0  |= 104;
  UARTFBRD_0  |= 11;

  UARTIBRD_2  |= 104;
  UARTFBRD_2  |= 11;

  UARTCC_0   |= 0x5;
  UARTLCRH_0  |= 0x60;

  UARTCC_2   |= 0x5;
  UARTLCRH_2  |= 0x60;

//   Start UART
  UARTCTL_0  |= 0x301;

  UARTCTL_2  |= 0x301;

  // Init GPIO port A for UART rea
  GPIODEN_A    |= 0xC3;
  GPIOAFSEL_A  |= 0xC3;
  GPIOAMSEL_A = 0x0;
  // Config PA0, PA1, PA6 and PA7
  GPIOPCTL_A  |= 0x11000011;
}
```

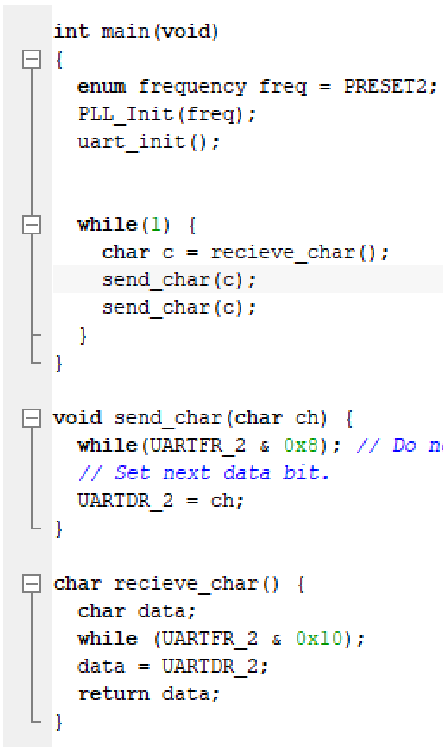


```c
void ADC0SS3_Handler(void) {
  // STEP 4: Implement the ADC ISR.
  // 4.1: Clear the ADC0 interrupt flag
  ADCISC_0 = 0x8;
  GPTMICR_0  |= 0x1;
  // 4.2: Save the ADC value to global variable ADC_value
  ADC_value = ADCSSFIFO3_0;
  float temperature = 147.5 - (247.5 * (ADC_value)) / 4096.0;
  char uart[UARTMSG];
  sprintf(uart, "%.2f\n\r", temperature);
  int i;
  for (i = 0; i < UARTMSG; i++) {
    while (UARTFR_0 & 0x8); // 0x8
    UARTDR_0 = uart[i];
  }
}
```

Task 2b

Procedure:

1. Connect the bluetooth module to the TIVA board by connecting the ground and power supply accordingly and attaching the modules Rx pin to the UARTs (in our case, UART 2) Tx pin and vice-versa for the Tx pin on the module.
2. Connect the bluetooth module to your computer.
3. Initialize UART 2 as we did for UART 0 on task 1a. Be sure to do the same for the GPIO registers.
4. Go to device manager and find the COM port associated with the bluetooth module.
5. Open PuTTY as done in task 2a but using the COM port associated with the bluetooth module instead. Check the bluetooth module to see if it stops blinking which indicates that it is connected correctly.
6. Create a program that receives a character written on the PuTTY terminal and sends it back to the terminal. Shown on the right.
7. Run the program and open up the PuTTY terminal with the correct serial COM port associated with the bluetooth module and type on the terminal.



```
int main(void)
{
  enum frequency freq = PRESET2;
  PLL_Init(freq);
  uart_init();


  while(1) {
    char c = recieve_char();
    send_char(c);
    send_char(c);
  }
}

void send_char(char ch) {
  while(UARTFR_2 & 0x8); // Do n
  // Set next data bit.
  UARTDR_2 = ch;
}

char recieve_char() {
  char data;
  while (UARTFR_2 & 0x10);
  data = UARTDR_2;
  return data;
}
```

Results:

For task 2a, we were able to get the temperature readings from the previous task using the ADC, Task 1b, and instead of reading it on our IAR terminal, we were able to read it on the PuTTY terminal.
For task 2b, we were able to create a “Return-to-Sender” program that would return the same character that was typed on the PuTTY terminal. In our case, we made the function send back the typed character twice for verification.