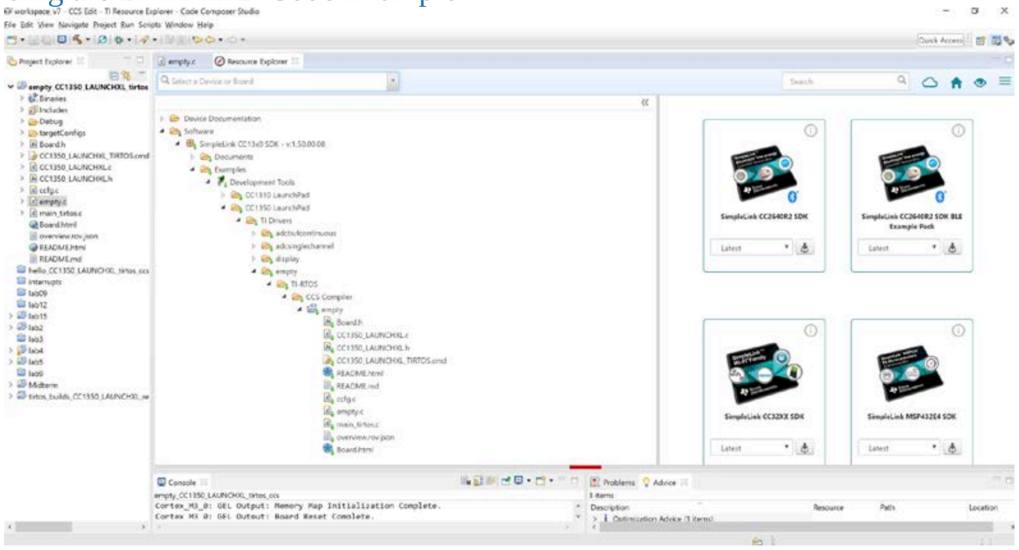
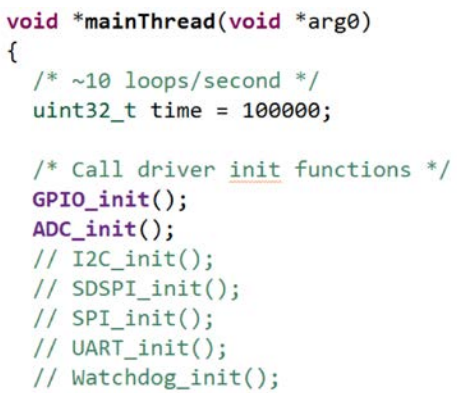
Matthew Hoffman
CPE 403 – Advanced Embedded Systems
CC1350 Lab 2

## Task 1: Finding the Blink LED Code Example



```
empty_CC1350_LAUNCHXL_tirtos_ccs
Cortex_M3_0: GEL Output: Memory Map Initialization Complete.
Cortex_M3_0: GEL Output: Board Reset Complete.
```

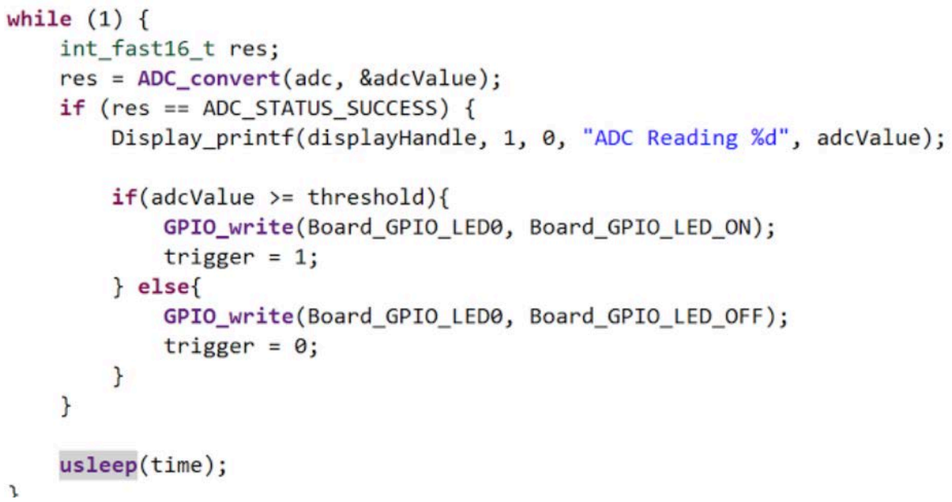
**Figure 1 – Task 1.a Empty Template**



```
void *mainThread(void *arg0)
{
    /* ~10 loops/second */
    uint32_t time = 100000;

    /* Call driver init functions */
    GPIO_init();
    ADC_init();
    // I2C_init();
    // SDSPI_init();
    // SPI_init();
    // UART_init();
    // Watchdog_init();
```

**Figure 2 – Task 1.b mainThread Function**



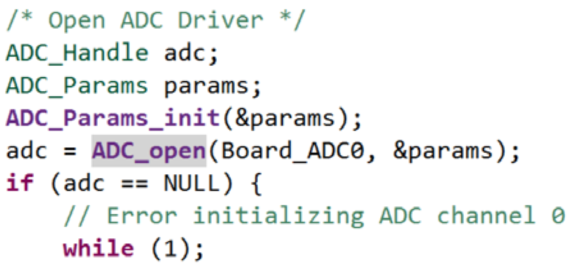
```c
while (1) {
    int_fast16_t res;
    res = ADC_convert(adc, &adcValue);
    if (res == ADC_STATUS_SUCCESS) {
        Display_printf(displayHandle, 1, 0, "ADC Reading %d", adcValue);

        if(adcValue >= threshold){
            GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_ON);
            trigger = 1;
        } else{
            GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_OFF);
            trigger = 0;
        }
    }

    usleep(time);
}
```

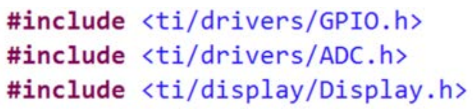
**Figure 3 – Task 1.c ADC Conversion**

## Task 2: Modify the example so LED is ON only if an ADC reading exceeds a threshold



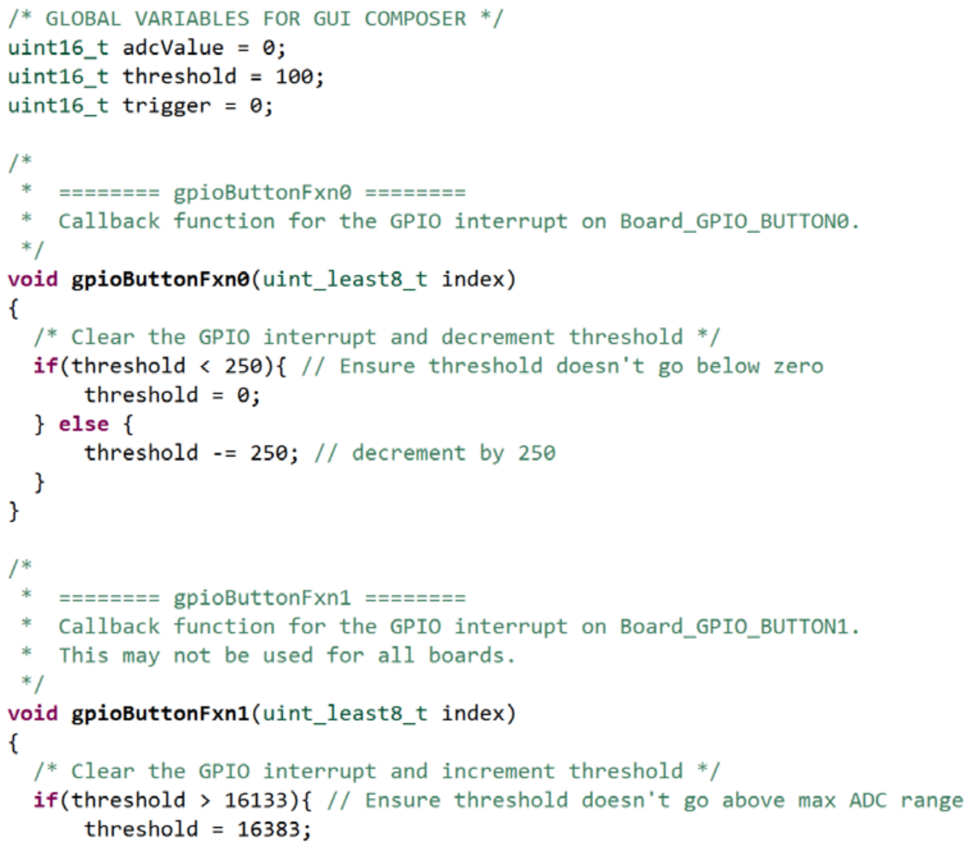
```c
/* Open ADC Driver */
ADC_Handle adc;
ADC_Params params;
ADC_Params_init(&params);
adc = ADC_open(Board_ADC0, &params);
if (adc == NULL) {
    // Error initializing ADC channel 0
    while (1);
```

**Figure 4 – Task 2.a ADC Handle**



```c
#include <ti/drivers/GPIO.h>
#include <ti/drivers/ADC.h>
#include <ti/display/Display.h>
```

**Figure 5 – Task 2.b GPIO, ADC, Display Includes**



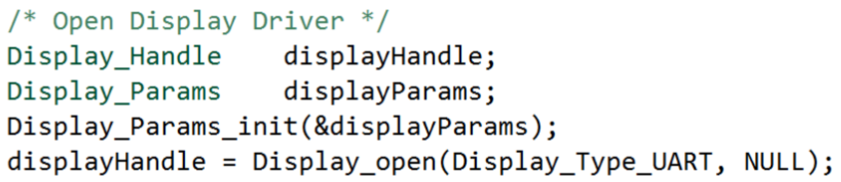
```c
/* GLOBAL VARIABLES FOR GUI COMPOSER */
uint16_t adcValue = 0;
uint16_t threshold = 100;
uint16_t trigger = 0;

/*
 *  ======== gpioButtonFxn0 ========
 *  Callback function for the GPIO interrupt on Board_GPIO_BUTTON0.
 */
void gpioButtonFxn0(uint_least8_t index)
{
  /* Clear the GPIO interrupt and decrement threshold */
  if(threshold < 250){ // Ensure threshold doesn't go below zero
      threshold = 0;
  } else {
      threshold -= 250; // decrement by 250
  }
}

/*
 *  ======== gpioButtonFxn1 ========
 *  Callback function for the GPIO interrupt on Board_GPIO_BUTTON1.
 *  This may not be used for all boards.
 */
void gpioButtonFxn1(uint_least8_t index)
{
  /* Clear the GPIO interrupt and increment threshold */
  if(threshold > 16133){ // Ensure threshold doesn't go above max ADC range
      threshold = 16383;
```

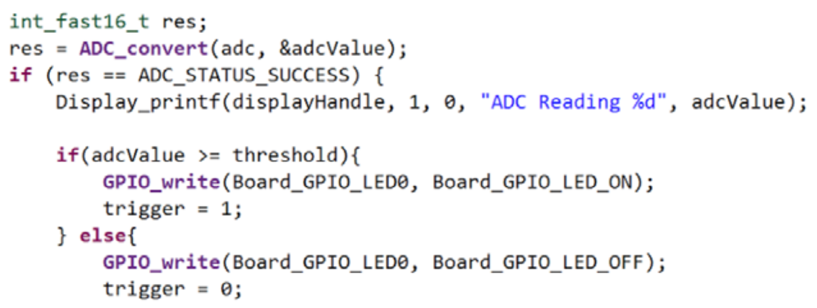
**Figure 6 – Task 2.c Added-in Callback Functions**

## Task 3: Adding a serial UART transmission to report ADC readings



```c
/* Open Display Driver */
Display_Handle    displayHandle;
Display_Params    displayParams;
Display_Params_init(&displayParams);
displayHandle = Display_open(Display_Type_UART, NULL);
```

**Figure 7 – Task 3.a Display Driver**

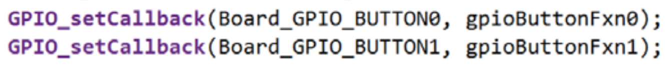


```
int_fast16_t res;
res = ADC_convert(adc, &adcValue);
if (res == ADC_STATUS_SUCCESS) {
    Display_printf(displayHandle, 1, 0, "ADC Reading %d", adcValue);

    if(adcValue >= threshold){
        GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_ON);
        trigger = 1;
    } else{
        GPIO_write(Board_GPIO_LED0, Board_GPIO_LED_OFF);
        trigger = 0;
```

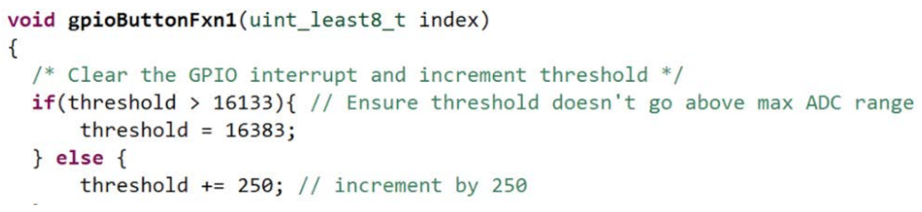
**Figure 8 – Task 3.b ADC Conversion**

## Task 4: Adding GPIO interrupts to our base example



```
GPIO_setCallback(Board_GPIO_BUTTON0, gpioButtonFxn0);
GPIO_setCallback(Board_GPIO_BUTTON1, gpioButtonFxn1);
```

**Figure 9 – Task 4.a Button Callback Functions**



```
void gpioButtonFxn1(uint_least8_t index)
{
  /* Clear the GPIO interrupt and increment threshold */
  if(threshold > 16133){ // Ensure threshold doesn't go above max ADC range
      threshold = 16383;
  } else {
      threshold += 250; // increment by 250
```

**Figure 10 – Task 4.b Interrupt Threshold Code**