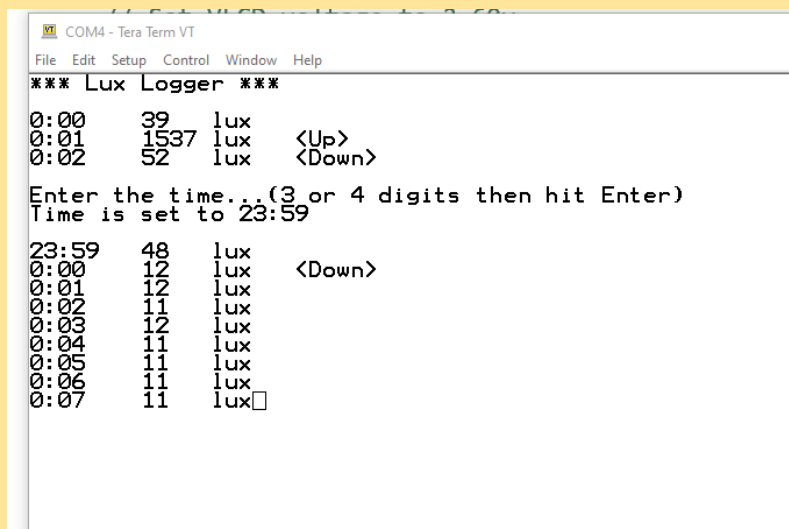


LUX LOGGER PROJECT

As I have worked alongside light sensors and embedded software throughout both my schoolwork and internship, this is a perfect combination of both. Towards the beginning of my internship, I learned about different communication protocols such as Inter-Integrated Circuit (I2C) and Universal Asynchronous Rx/Tx and how the hardware wiring is completed. This became quite helpful later in my school course, Embedded Systems, when I began to learn how to execute these communication modes with the software. With both combined, I learned how to fully implement these communication protocols on both the hardware and software areas.

This project required the usage of the MSP430FR6989 Launchpad, alongside the BoosterPack from Texas Instruments. The BoosterPack has a light sensor that can be read and written to using I2C. In order to make this data visually useful to users, I was required to output the data to the screen using UART.



```
COM4 - Tera Term VT
File Edit Setup Control Window Help
*** Lux Logger ***
0:00 39 lux
0:01 1537 lux <Up>
0:02 52 lux <Down>
Enter the time... (3 or 4 digits then hit Enter)
Time is set to 23:59
23:59 48 lux
0:00 12 lux <Down>
0:01 12 lux
0:02 11 lux
0:03 12 lux
0:04 11 lux
0:05 11 lux
0:06 11 lux
0:07 11 lux
```

Another requirement of the project was to detect when the next reading was either above or below the previous reading by more than 20 lux.

For reference, a well lit desk will have a lux (light) reading of approximately 400 lux.

Finally, the project required a timer that would ask for a new reading every second. In addition, the user could press a button located on the LaunchPad and type in a time with the keyboard. The software reads this keyboard input through UART.

Overall, this project helped me better understand the objects and software I work with at my job from day to day.

Below are thinking maps and screenshots of sections of code for reference.

```

void Initialize_I2C(void) {
    // Configure the MCU in Master mode
    // Configure pins to I2C functionality
    // (UCB1SDA same as P4.0) (UCB1SCL same as P4.1)
    // (P4SEL1=11, P4SEL0=00) (P4DIR=xx)
    P4SEL1 |= (BIT1|BIT0);
    P4SEL0 &= ~(BIT1|BIT0);
    // Enter reset state and set all fields in this register to zero
    UCB1CTLW0 = UCSWRST;
    // Fields that should be nonzero are changed below
    // (Master Mode: UCMST) (I2C mode: UCMODE_3) (Synchronous mode: UCSYNC)
    // (UCSSEL 1:ACLK, 2,3:SMCLK)
    UCB1CTLW0 |= UCMST | UCMODE_3 | UCSYNC | UCSSEL_3;
    // Clock frequency: SMCLK/8 = 1 MHz/8 = 125 KHz
    UCB1BRW = 8;
    // Chip Data Sheet p. 53 (Should be 400 KHz max)
    // Exit the reset mode at the end of the configuration
    UCB1CTLW0 &= ~UCSWRST;
}

// Configures ACLK to 32 KHz crystal
void config_ACLK_to_32KHz_crystal() {
    // By default, ACLK runs on LFMODCLK at 5MHz/128 = 39 KHz

```

```

void uart_write_uint16 (unsigned int n){

    //integer used to hold rightmost number in digit
    int k=0;
    //counter used to advance to next component in array
    int place=0;

    // Do while loop to extract each digit
    do{
        //extracts rightmost number
        k=(n%10);
        //saves rightmost number in value of counter of array
        UART_Display[place]=k;
        //shifts number to the right one place
        n = n/10;
        //advances array placeholder
        place++;
    }while(n>0);

    k=0;

    for(k=place-1;k>=0;k--){
        uart_write_char('0'+UART_Display[k]);
    }
}

```

7.3 APPLICATION: LUX LOGGER

Below is my code and thinking guide for how I completed the code for 7.3.

Requirements:

1. Read sensor every minute and print to module with timestamp
2. At the start AND every time the lux goes out of range, update the range +/- 10 of lux value
3. Print up or down if out of range
4. Button s2 is pressed, then user can set time.

Must be able to detect either 3 numbers then ENTER key OR 4 numbers then ENTER key

Timer starts based on HH:MM

Save MM to int a

Increment seconds

If ss = 59,

S = 0

MM++

If MM = 59

MM=0

HH++

If HH >23

HH=0

If MM = a + 1

Read sensor

Save result

Print time (only hour and minute)

Print sensor result

Button S2 is pressed

User sets time

3 or 4 number format

A challenging part of this section was being able to successfully read either 3 or 4 keyboard inputs. One of the main things I did to achieve this was change the uart_read_char() function. Instead of checking once for a reading, I changed it to "blocking" style by using a while loop instead of an if statement. This means my function now polled until a new byte was received.