Lab 2: Software Configuration and Simple Parallel I/O

Micah Galos - SID: 862082339
Joseph Ayala - SID: 862080244

Due: October 26th, 2020 - 11:59PM

Abstract[JDA]

The objective of this lab is to familiarize ourselves further with the Kinetis software and manipulate peripherals that can be attached to the K64FRDM Board which would aid in our understanding of Digital I/O. The lab begins with installing more firmware to communicate with the board and navigating through the User interface (**PART 1 & PART 2**) to then start our first program with the Kinetis software and K64 FRDM Board. Within the next section of the lab (**PART 3**)that was successfully taken and transferred over to run on board the K64. Lastly we were able to build and produce (**PART 4**) a simple parallel input/output led circuit that would work within the provided constraints.

Experiment System Specification [JDA]

What is to be designed and implemented?
In the course of this lab the main system that would be developed and implemented would be in **PART 4** of the lab where a parallel input/output led bars are outputting different LED outputs relative to an input from the four channel dip switch. Other items that were to be taken into consideration when implementing the hardware were the pins that would be used for inputting and outputting separate header pins had to be soldered on in order for better accessibility to other pins of the FRDM-K64F.

From following the project description in the lab manual Micah was able to produce the following circuit seen in figure 2. There were some issues in terms of implementing the hardware as the pins were problematic in terms of connecting with the breadboard. Beyond these issues however the parallel counters were able to communicate with the microcontroller accordingly and behave as prescribed in the lab manual in terms of behaving appropriately with the inputs from the DIP switches.

SCHEMATIC [JDA]

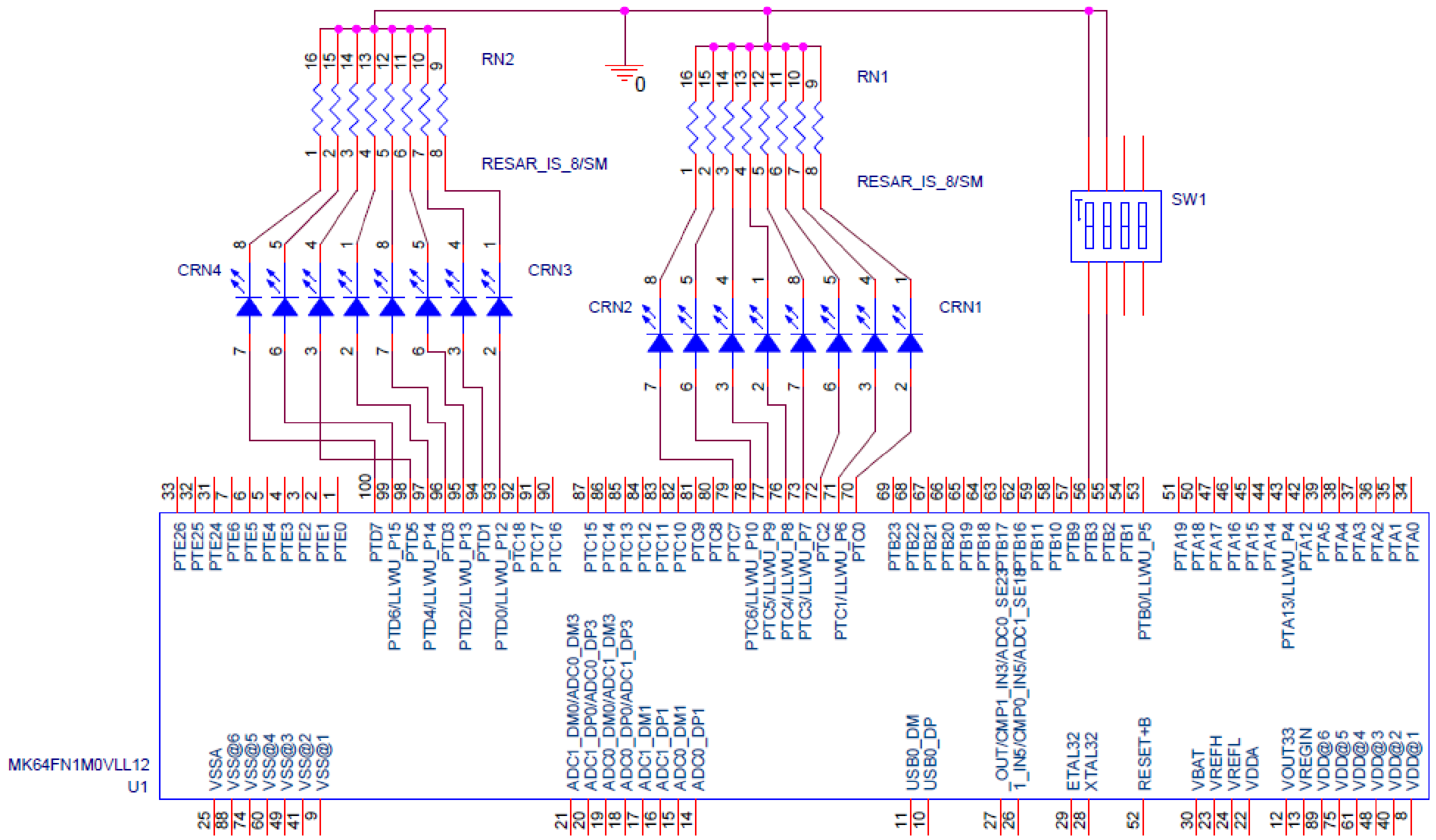


Figure 2
Schematic of PART 4

Breadboard Design

- Thanks to my partner Joseph for soldering the remainder of Port D’s pins on both of our board in preparation of the lab - Micah Galos

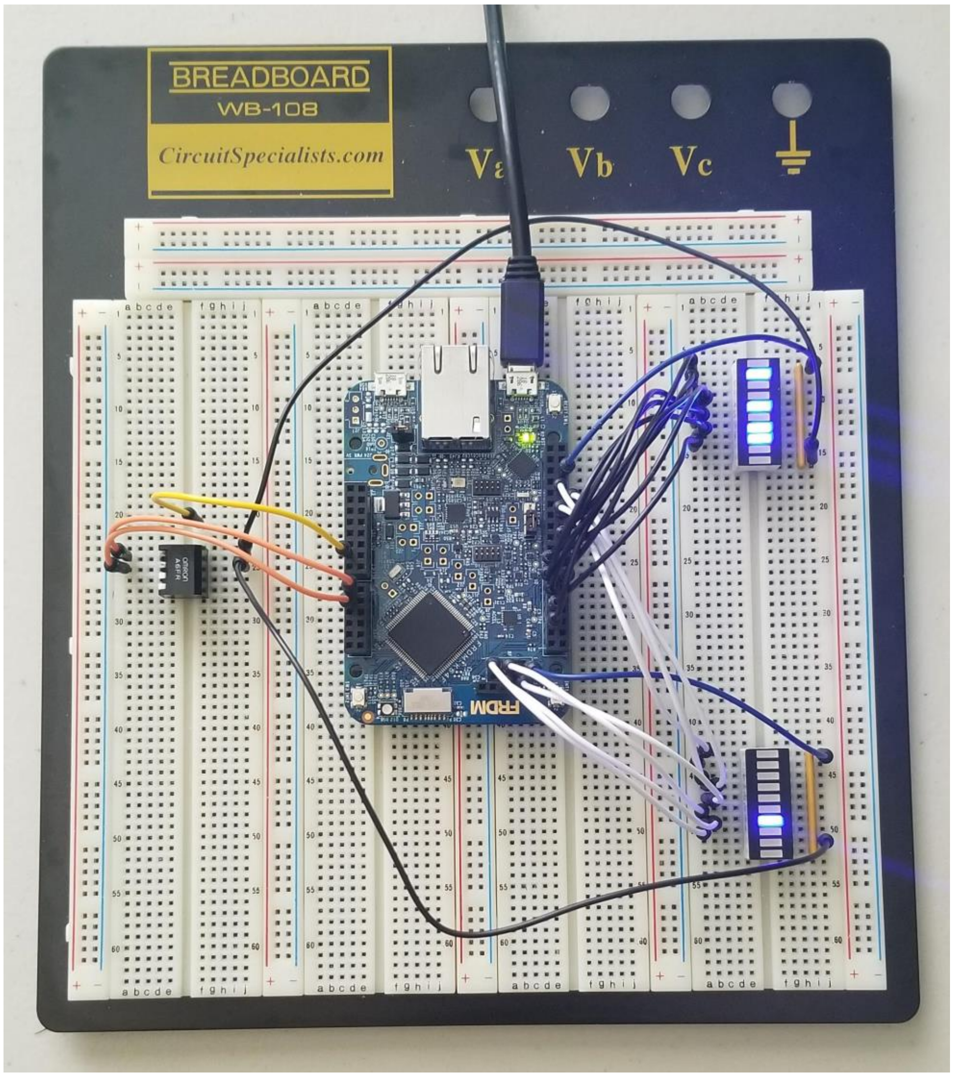
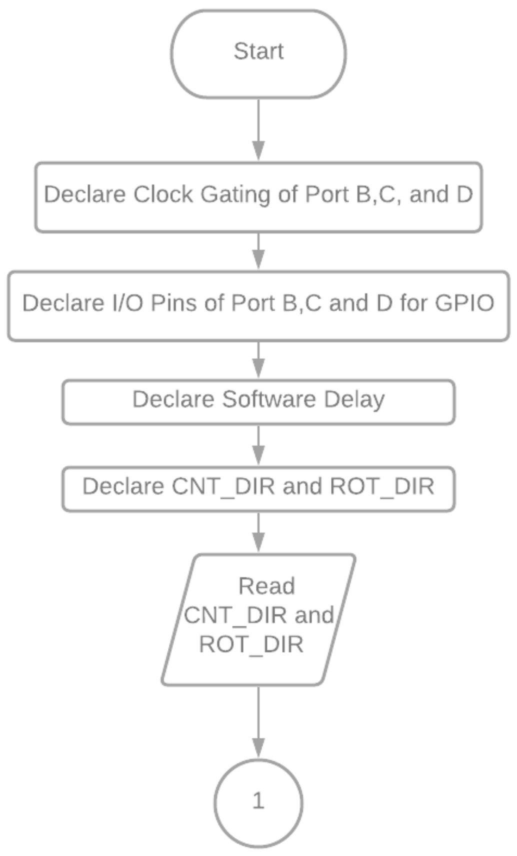
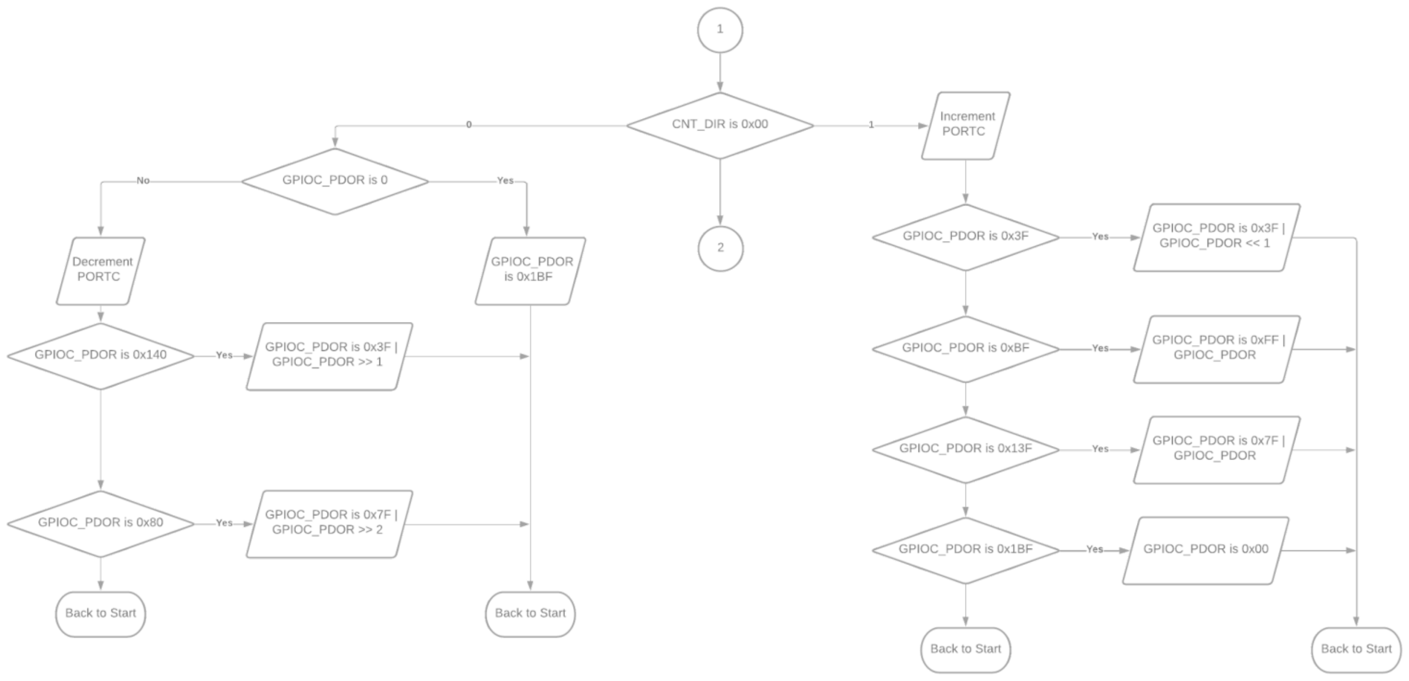
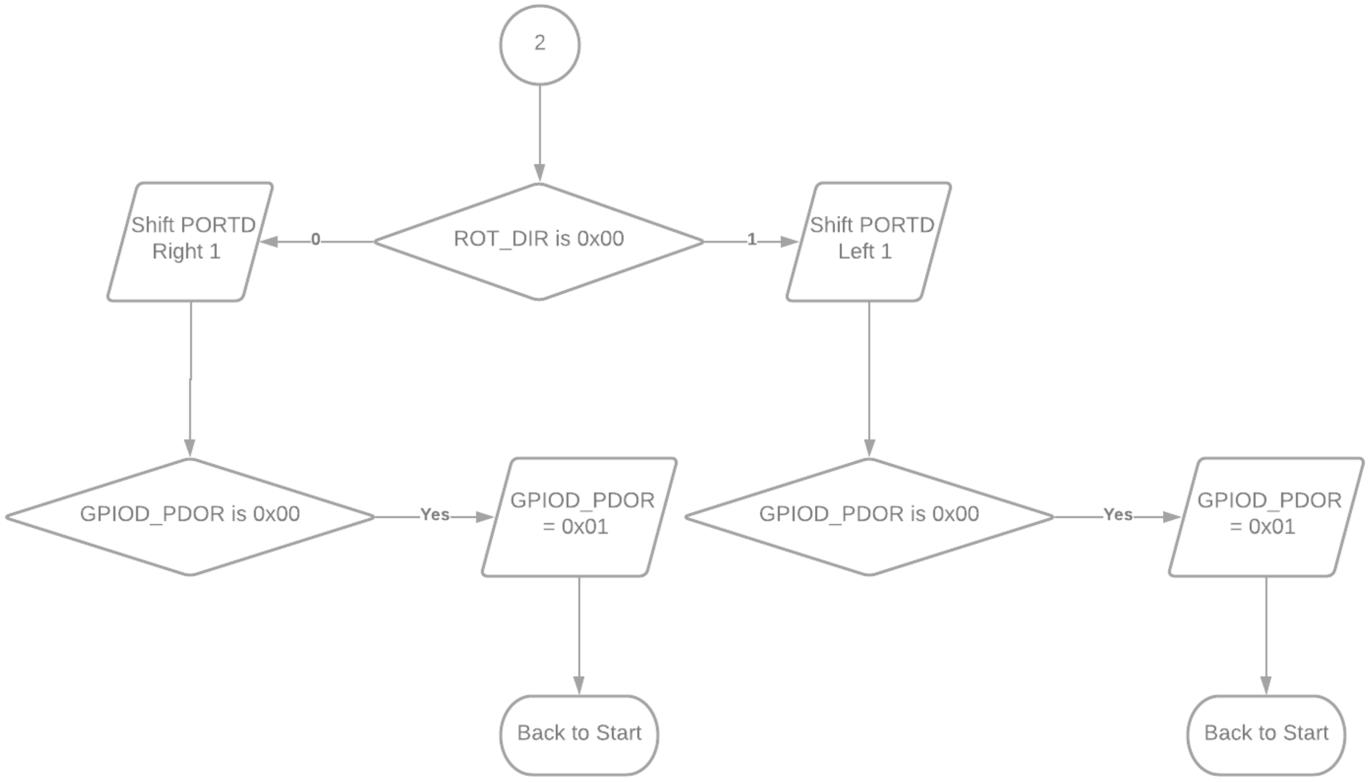


Figure 2
Physical Implementation

Flowchart Diagram[Micah Galos]


Start
Declare Clock Gating of Port B,C, and D
Declare I/O Pins of Port B,C and D for GPIO
Declare Software Delay
Declare CNT_DIR and ROT_DIR
Read CNT_DIR and ROT_DIR
1





Software Design[Micah Galos]

```
#include "fsl_device_registers.h"
void software_delay(unsigned long delay)
{
     while (delay > 0) delay--;
}
int main(void)
{
     /* Configure Clock Gating for Ports B,C, and D; */
     SIM_SCGC5 |= SIM_SCGC5_PORTB_MASK;
     SIM_SCGC5 |= SIM_SCGC5_PORTC_MASK;
     SIM_SCGC5 |= SIM_SCGC5_PORTD_MASK;

     /* NOTE: These are for opening pins of ports */
     /* Configure Port D Pins 0-7 for GPIO; */
     PORTD_GPCLR = 0x00FF0100;

     /* Configure Port C Pins 0-5 and 7-8 for GPIO; */
     PORTC_GPCLR = 0x01BF0100;

     /* Configure Port B Pin 0 & 1 for GPIO; */
     PORTB_GPCLR = 0x000C0100;

     /* NOTE: GPIOx_PDDR for I/O Pin Configuration */
```

```c
/* Configure Port B Pin 2, 3 for Input; */
GPIOB_PDDR |= (0 << 2);
GPIOB_PDDR |= (0 << 3);

/* Configure Port D Pins 0-7 for Output; */
GPIOD_PDDR |= 0x000000FF;

/* Configure Port C Pins 0-5 and 7-8 for Output; */
GPIOC_PDDR |= 0x000001BF;

/* Initialize Port C to 0; */
GPIOC_PDOR |= 0x00;

/* Initialize Port D such that only 1 bit is ON; */
GPIOD_PDOR |= 0x10;

 /*Delay Value*/
//unsigned long Delay = 0x10000; // Fast
unsigned long Delay = 0x100000; // Slow
unsigned long CNT_DIR = 0; unsigned long ROT_DIR = 0;

while(1) {
     software_delay(Delay); /*Wait Delay Value*/
     /* Read Port B*/
     CNT_DIR = ~(GPIOB_PDIR) & 0x04;
     ROT_DIR = ~(GPIOB_PDIR) & 0x08;

     /* Counter PORTC using PORTB*/
     if(CNT_DIR == 0x00){
          GPIOC_PDOR++; // Increment

          /*  Hold lower 6 bits and left shift bits 6,7 to 7,8
           *  to simulate pin 6 output on LED via bit manipulation
           */
          if(GPIOC_PDOR == 0x3F){
               GPIOC_PDOR;
               GPIOC_PDOR = 0x3F | (GPIOC_PDOR << 1);
               software_delay(Delay); /*Wait Delay Value*/
               GPIOC_PDOR;
          }
          if(GPIOC_PDOR == 0xBF){
               GPIOC_PDOR;
               GPIOC_PDOR = (0xFF | GPIOC_PDOR);
               software_delay(Delay); /*Wait Delay Value*/
               GPIOC_PDOR++;
          }
```

```
        if(GPIOC_PDOR == 0x13F){
            GPIOC_PDOR;
            GPIOC_PDOR = (0x7F | GPIOC_PDOR);
            software_delay(Delay); /*Wait Delay Value*/
            GPIOC_PDOR++;
        }
        if(GPIOC_PDOR == 0x1BF){
            software_delay(Delay); /*Wait Delay Value*/
            GPIOC_PDOR++;
            GPIOC_PDOR = 0x00;
        }
    }
    else{
        if(GPIOC_PDOR == 0x00){
            software_delay(Delay); /*Wait Delay Value*/
            GPIOC_PDOR = 0x1BF;
        }
        GPIOC_PDOR--; // Decrement

        /*  Hold lower 6 bits and right shift bits 7,8 to 6,7
         *  to simulate pin 6 output on LED via bit manipulation
         */
        if(GPIOC_PDOR == 0x140){
            GPIOC_PDOR;
            GPIOC_PDOR = (0x3F) | (GPIOC_PDOR >> 1);
            software_delay(Delay); /*Wait Delay Value*/
            GPIOC_PDOR;
        }
        if(GPIOC_PDOR == 0x80){
            GPIOC_PDOR;
            GPIOC_PDOR = 0x1F | (GPIOC_PDOR >> 2);
            software_delay(Delay); /*Wait Delay Value*/
            GPIOC_PDOR;
        }
    }

    /* Rotator PORTD using PORTB */
    if(ROT_DIR == 0x00){
        GPIOD_PDOR = GPIOD_PDOR << 1; // Shift Left 1
        if(GPIOD_PDOR == 0x100){
            GPIOD_PDOR = 0x01;
        }
    }
    else{
        GPIOD_PDOR = GPIOD_PDOR >> 1; // Shift Right 1
        if(GPIOD_PDOR == 0x00){
```

```
                    GPIOD_PDOR = 0x80;
                }
            }
        }
        return 0;
}
```

Technical Problems[Micah Galos]

There were some problems which occurred in the lab, one is soldering our boards with header pins in order to connect our jumper cables to Port D Pins [4:7] in favor of the manual. The other issue was the DIP switch itself because the pins are so short, the signal will not be received unless re-slotted into the board.

Questions[Micah Galos]

1) What is the size of your .elf file for PART 4? 2) What do you think is the actual size of your program code? (hint: check log messages in the Console tab when you build the project)

According to the logs shows:

```
   text      data       bss       dec       hex   filename
   2312       108      2076      4496      1190   Lab_2_Experiment.elf
```

Looking at the actual file size of Part 4, is 1,927 kilobytes, which is relatively close to the hex value in the logs--likewise with the text value. Perhaps it is because of the included comments there is a variance in size between saved locally and building from the Kinetics environment

3) Consider the following C source code:

```
#define PTX_BASE (0x400FF040u)
#define PTX ((GPIO_Type *)PTX_BASE)
#define GPIO_PDOR_REG(PTX_BASE) ((base)->PDOR)
#define GPIO_PDDR_REG(PTX_BASE) ((base)->PDDR)
#define GPIOX_PDOR GPIO_PDOR_REG(PTX)
#define GPIOX_PDDR GPIO_PDDR_REG(PTX)
```

From this code, which I/O port of K64F is referred to by GPIOX_PDOR and GPIOX_PDDR? Why? (Hint: must be A, B, C, D, or E. Also, check MK64F12.h)

From the MK64F12 header file, the first line in the given code indicates it is Port B's base address. Lines 2 through 5 from the question, all contain the same defined headers for their respective port. Therefore, I/O port B is referred to GPIOX_PDOR and GPIO_PDDR from this instance.

Conclusion

In the end, we overcame those technical issues during the process of finishing the lab. Not so much the DIP switch because the pins are too short to receive an input with our breadboards, but we made the best out of using the component. The toughest part of this lab book-keeping the hex value and tobit manipulation when Port C’s LED reached the sixth bit. Since we skipped the pin 6 of Port C, we wrote if-statements when the counter reached a certain hex value we keep the lower six bits and shift the current count. The new output would be when you OR the two hex values together.