EECS 2032 Lab 8 Report

Jorra Singh - 215709876
Fabian Cojman - 216299976

Problem Statement (Part 1) :

For part 1 of this lab we were told to watch the given videos and learn the basics of our microcontroller and the MCUXpresso compiler. After this was done, student were able to complete the actual lab tasks.

Part 1 of this lab required students to write a small program that allowed a user to control the LED’s on the board. The board we were told to use had 2 LED’s and 2 switches.The two switches, switch 1 and switch 3 which henceforth be referred to as SW1 and SW3. The colours of the LED’s were red and green and will be referred to here on out as LEDR and LEDG respectively.

The first part of our lab had the following requirements:

- Initial states of the two LED’s must be off.
- If you press SW1 and SW3 is not pressed then both LED’s should turn on.
- If you press SW3 and and SW1 is pressed, LEDG turns on, and LEDR is off.
- If both SW1 And SW3 are not pressed, the state of both LED’s are off.
- If both SW1 and SW3 are pressed, releasing either one of the switches will make LEDG off and LEDR on.

Problem Statement (Part 2) :

For the second part of the lab, we are to know that the microcontroller that we have has an on board light sensor. Using this sensor, we are to get its output, to the input of the ADC. After this, we must convert it to a digital value and display it on the monitor. The demonstration of its functionality can be done by using a flashlight to trigger the sensor, or using one's hand to partially cover the sensor and trigger it.

Code Part 1:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MKL43Z4.h"
#include "fsl_debug_console.h"
/* TODO: insert other include files here. */

/* TODO: insert other definitions and declarations here. */

/*
 * @brief   Application entry point.
 */
int main(void) {

      /* Init board hardware. */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitBootPeripherals();

    /* Init FSL debug console. */
    BOARD_InitDebugConsole();
    PRINTF("Hello World\n");

    //turns on the ports
    SIM->SCGC5 |= ((1<<9) | (1<<11) | (1<<12) | (1<<13));

    //configures portA
    PORTA->PCR[4] = 0x103; //port A pin 4 GPIO(mux = 001) PS=PE=1 pull up
resistor
    //configures portC
    PORTC->PCR[4] = 0x103; //port C pin 4 GPIO(mux = 001) PS=PE=1 pull up
resistor
    //configures portD
    PORTD->PCR[5] = 0x100; //port D pin 5 GPIO (mux = 1) PS=PE=0 no pull up
or down
    //configures portE
    PORTE->PCR[31] = 0x100; //port E pin 31 GPIO (mux = 1) PS=PE=0 no pull
up or down

    //sets the port function
    PTA->PDDR &= ~(0x10); // set bit 4 of port A to 0 (input)
    PTC->PDDR &= ~(0x10);
    PTD->PDDR |= (1 << 5);// set bit 5 of port D to 1 (pin 5 is output)
    PTE->PDDR |= (1 << 31);
    //sets the default state to off
    PTD->PDOR = (1<<5); // set G_LED off
    PTE->PDOR = (1<<31); // set R_LED off

    bool temp = false;

    while(1) {

      volatile int SW1 = PTA->PDIR & (1<<4); //SW1
      volatile int SW3 = PTC->PDIR & (1<<4); //SW3

        if(SW1 == 1 && SW3 == 0){
            PTD->PDOR = 0;
            PTE->PDOR = 0;
         }else{
            PTD->PDOR = (1<<5);
            PTE->PDOR = (1<<31);
        }


        if(SW1 == 1 && SW3 == 1){
            temp = true;
        }

        while(temp){
            SW1 = PTA->PDIR & (1<<4); //SW1
            SW3 = PTC->PDIR & (1<<4); //SW3

            PTD->PDOR = 0;
            PTE->PDOR = (1<<31);

            if((SW1 == 0 && SW3 == 1) || (SW1 == 1 && SW3 == 0)){
                  PTE->PDOR = 0;
                  PTD->PDOR = (1<<31);
            }else{
                  temp = false;
            }

        }


    }

    return 0;
}
```

Code Part 2:

```
#include <stdio.h>
#include "board.h"
#include "peripherals.h"
#include "pin_mux.h"
#include "clock_config.h"
#include "MKL43Z4.h"
#include "fsl_debug_console.h"
/* TODO: insert other include files here. */

/* TODO: insert other definitions and declarations here. */

/*
 * @brief   Application entry point.
 */
int main(void) {

      /* Init board hardware. */
    BOARD_InitBootPins();
    BOARD_InitBootClocks();
    BOARD_InitBootPeripherals();
      /* Init FSL debug console. */
    BOARD_InitDebugConsole();

    PRINTF("Start Sample");

    SIM->SCGC5 |= (1<<13); //turn on clock to port E

    PORTE->PCR[22] &= ~(0x700); //Set mux bits to 0

    SIM->SCGC6 |= 0x8000000; //enable ADC

    ADC0->CFG1 = 0x00000008; //ADC mode 10 ie 10bit mode

    ADC0->SC2 = 0x00000001; //use default V-ref

    while(1) {

      ADC0->SC1[0] = 0x03; // channel 7 selected

      while(!(ADC0->SC1[0] & 0x04)) { } //wait for MC to complete conversion
the 3rd bit should be 1 when complete

      PRINTF("%d\n", ADC0->R[0]); //print the conversion
    }

    return 0 ;

}
```

Diagrams:

State Diagram (written in pseudo) :

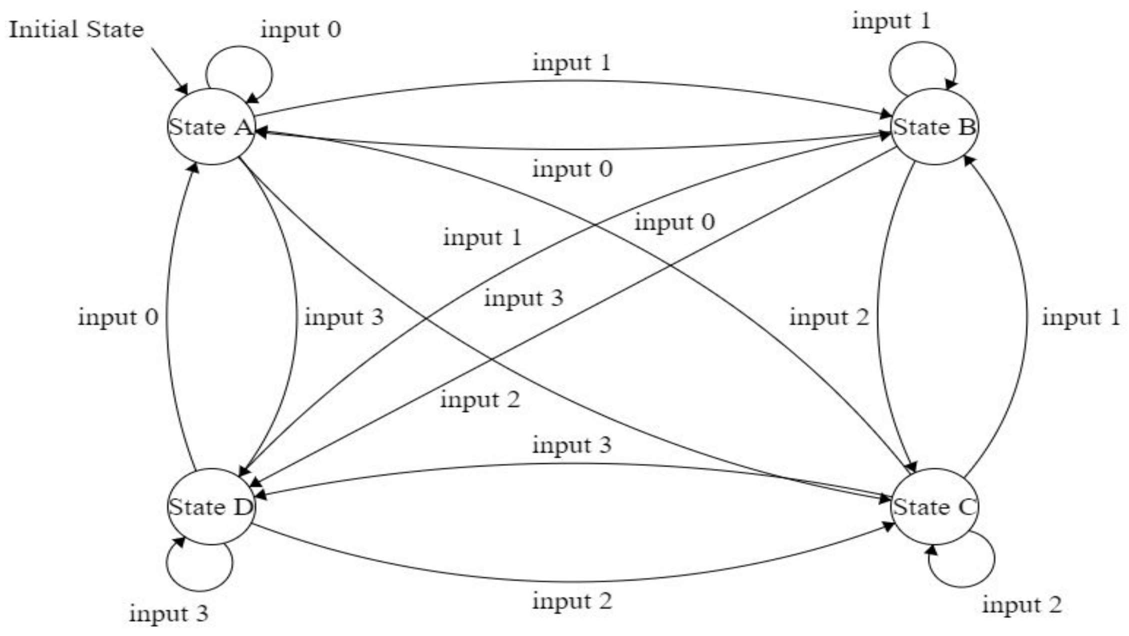
For this lab we have four possible states. They are listed below.

State A: LEDG is off and LEDR is off
State B: LEDG is on and LEDR is on
State C: LEDG is on and LEDR is off
State D: LEDG is off and LEDR is on

We also have four possible inputs that we can anticipate. Their combinations are listed below…
Input 0: SW1 Is down and SW3 is also down
Input 1: SW1 is up and SW3 is also up
Input 2: SW1 is up and SW3 is down
Input 3: SW1 is down and SW3 is up

The above four inputs represent the above four states that we can experience. The State diagram for these is shown below.

State Diagram:



<u>Explanation of State Diagram :</u>

For this lab, we have two LEDs and two switches. The two LED’s are red and green and will henceforth be referred to as LED_R and LED_G respectively. The two switches in this lab are switch 1 and switch 3. They will henceforth be referred to as SW1 and SW3 respectively.

State A represents both LED_R and LED_G in their OFF state. This is the initial state of the machine. From this initial state, all the possible combinations are listed above. Starting at the initial state, State A, we can have four different inputs. They will henceforth be referred to as input 0, input 1, input 2, and input 3. Their functions are detailed above. From State A, we can choose to do either of the four inputs. Input 0 loops us back around to State A, ie. nothing changes. Input 1 allows us to move from State A to State B. Input 2 allows us to skip State B and go directly to State C. Likewise, input 3 allows us to skip both State B and C, and go directly to State D. All states have similar combinations that take them to one of the four possible states. They all have an input that loops them back to themselves, ie. do nothing.

Schematics of FRDM-KL43Z

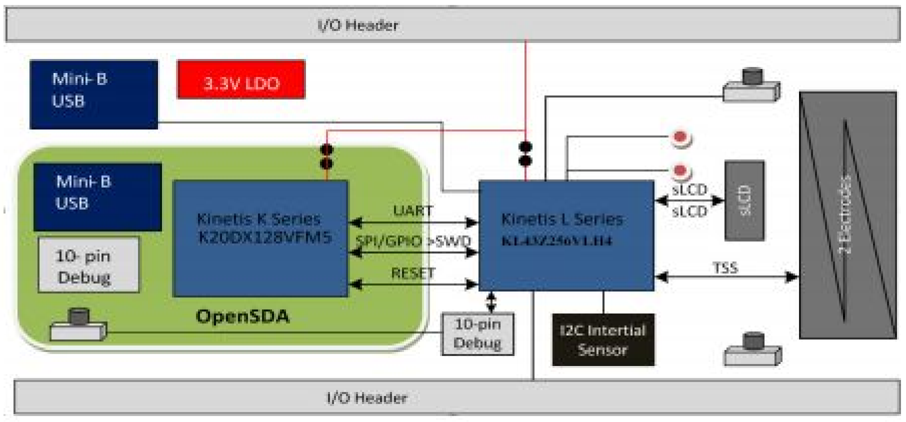


Figure 1: layout of FRDM-KL43X

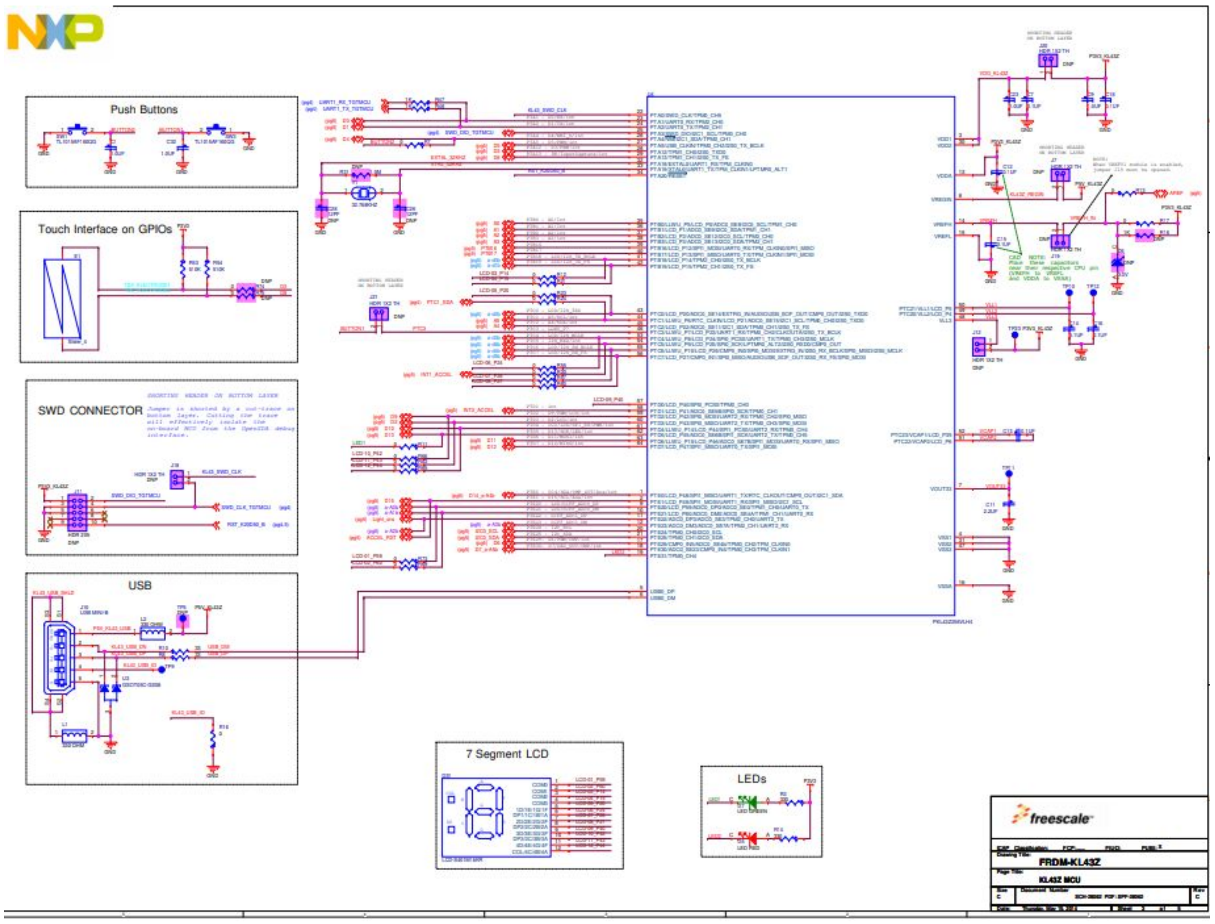


Figure 2: schematic of FRDM-KL43X