EECS 3215
Embedded Systems
Lab 7 - Interrupts
Michael Founk 213641204
Sean O’Brien 213735741

Problem Statement

The problem of this lab is that we would like to build a system which processes interrupts and alters the behaviour of the system in response. The default state of the system is that it will blink the green LED every two seconds, and when an interrupt (a button press) is processed, the green LED blinking is interrupted (and paused), and the system toggles the red LED. The difficulty of this lab is in configuring the interrupt response of the board, the basic behaviour of the board like blinking the LEDs and the receiving the pushbutton input is repeated from previous labs.

Top-Level Design

In our solution, we enable the clock to the ports we will be using, and then configure the interrupt using the API functions. The first step is to use the API functions to disable interrupts and disable the interrupt handler. The next step is to disable and enable interrupts manually for the pushbutton. The last step of configuration is to use the API functions again to enable interrupts and the interrupt handler itself. These configurations have enabled us to write an interrupt handler function, which is code that runs when an interrupt occurs. Inside this interrupt handler, we placed the code for blinking the red LED, and inside the main loop we place the code for blinking the green LED.

Code

```
#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"
#include <stdlib.h>

void delay(unsigned int mseconds){
        int c, d;

        for (c = 1; c <= mseconds; c++){
        for(d = 1; d <= 1000; d++)
        {

        }
        }
        }
void PORTC_PORTD_IRQHandler(void){

   PTE->PCOR |= 1<<31u;
```

```
    delay(500);
    PTE->PSOR |= 1<<31u;
    PORTC->PCR[3] |= 1<<24u;
    delay(8000);

}

int main(void) {

        BOARD_InitBootPins();
        BOARD_InitBootClocks();
        BOARD_InitBootPeripherals();
        BOARD_InitDebugConsole();

        SIM->SCGC5 |= SIM_SCGC5_PORTE_MASK; // enable clock to port E
        SIM->SCGC5 |= SIM_SCGC5_PORTD_MASK; // enable clock to port D
        SIM->SCGC5 |= SIM_SCGC5_PORTC_MASK; // enable clock to port C

        __disable_irq();
        NVIC_DisableIRQ(PORTC_PORTD_IRQn);
        PORTC->PCR[3] &= ~(0xF0000);
        PORTC->PCR[3] |= 0xA0103; // then enable ISF flag and Interrupt on falling-edge.
        NVIC_EnableIRQ(PORTC_PORTD_IRQn);
        __enable_irq();

        PORTD->PCR[5] |= 0x100; // enable green led
        PTD->PDDR |= 1<<5u; // set green led to output
        PTD->PSOR |= 1<<5u; // off by default

        PORTE->PCR[31] |= 0x100; // enable red led
        PTE->PDDR |= 1<<31u; // set green led to output
        PTE->PSOR |= 1<<31u; // off by default

        PORTC->PCR[3] |= 0x103; // enable pushbutton
        PTC->PDDR &= 0<<3u; // set pushbutton to input

        while(1) {

                PTD->PCOR |= 1<<5u;
                delay(500);
                PTD->PSOR |= 1<<5u;
                delay(4000);

        }
}
```