Lab 6
Jacob Hillebrand
CEE-345 Microprocessor System Design
First Lab for FRDM KL25Z

This lab was our first exploration with the Freedom Development Board KL25Z. For this lab, we simply set up the Keil Development Environmnet, and wrote our first program, aptly named BlinkyRED. This program very simply utilized a C file that instructed the microprocessor to blink LEDs.
The code began by sending a signal to the proper registers to enable the processor's clock signal on the ports B and D. Next, it set several GPIO pins (which are also connected to the onboard LED) to output mode by setting the value of the onboard MUX. Then, the PINs to the RED LED were connected, and the LEDs were all set to a default off so they were prepped for the red function. From this point on, the code simply repeatedly called the red function, which turned on and off the red LED every .1 seconds using a delay function specified in a separate header file. The code for this program is shown on the next pages, and the delay header file is on the last page, Fig. 5.

```
#include <MKL25Z4.H>
#include "gpio_defs.h"

/* CEE-345 Microprocessor System Design
Demonstration of simple digital output
Use RGB LED on Freedom Board*/

void Delay(unsigned int time_del) {
    //~1ms * time_del
  volatile int t;

  while (time_del--) {
    for (t=4800; t > 0; t--)
      ;

  }
}

/* Each LED corresponds to a bit on a port
  Red LED connected to Port B (PTB), bit 18 (RED_LED_POS)
  Green LED connected to Port B (PTB), bit 19 (GREEN_LED_POS)
  Blue LED connected to Port D (PTD), bit 1 (BLUE_LED_POS)
Active-Low outputs: Write a 0 to turn on an LED

Turning LEDs on and off
Turn on one LED: PTx->PDOR = ~ MASK(yyy_LED_POS) ;
Turn on two LEDs: PTx->PDOR = ~ (MASK(yyy_LED_POS) | MASK(zzz_LED_POS));
Turn all LEDs off: PTx->PDOR = 0xFFFFFFFF ;*/

void red(void)
{
  //turn on red led
  PTB-> PDOR = ~ MASK(RED_LED_POS);
  //turn off blue LED
  PTD->PDOR = 0xFFFFFFFF;

  //wait for 100 ms
  Delay(100);

  //turn off all LED
  PTB->PDOR = 0xFFFFFFFF;

  //turn off blue LED
  PTD->PDOR = 0xFFFFFFFF;

```

Figure 1: First Snippet

```
45
46    //wait for 100 ms
47    Delay(100);
48
49    //turn on red led
50    PTB-> PDOR = ~ MASK(RED_LED_POS);
51    //turn off blue LED
52    PTD->PDOR = 0xFFFFFFFF;
53
54    //wait for 100 ms
55    Delay(100);
56
57  }
58
59  /*******************************
60   MAIN function
61   *******************************/
62
63
64  int main (void) {
65    //Declare a global variable for debug
66    unsigned int counter = 0;
67
68    /* Configuration steps
69    1. Enable clock to GPIO ports
70    2. Enable GPIO ports (2 step process)
71    3. Set GPIO direction to output
72    4. Ensure LEDs are off */
73
74    //Enable clock to ports B and D
75    /*The symbol SIM_SCGC5 defines the address of the fifth
76    system clock rating register to be at 0x40048038. This and other
77    symbols that represent register addresses (or masks that set or clear
78    specific bits) are defined in the mCU-specific header file, MKL25Z4.h.
79    This header file is provided with the development tools.*/
80
81    SIM->SCGC5 |= SIM_SCGC5_PORTB_MASK | SIM_SCGC5_PORTD_MASK;
```

Figure 2: Second Snippet

```
int main (void) {
  //Declare a global variable for debug
  unsigned int counter = 0;

  /* Configuration steps
  1. Enable clock to GPIO ports
  2. Enable GPIO ports (2 step process)
  3. Set GPIO direction to output
  4. Ensure LEDs are off */

  //Enable clock to ports B and D
  /*The symbol SIM_SCGC5 defines the address of the fifth
  system clock rating register to be at 0x40048038. This and other
  symbols that represent register addresses (or masks that set or clear
  specific bits) are defined in the mCU-specific header file, MKL25Z4.h.
  This header file is provided with the development tools.*/

  SIM->SCGC5 |= SIM_SCGC5_PORTB_MASK | SIM_SCGC5_PORTD_MASK;

  // Make 3 pins GPIO through GPIO setup below
  // The macro PORT_PCR_MUX() is defined in the header
  // file and sets the MUX bits in the corresponding pin control register.
  // two steps process: configure pin multiplexer and select a GPIO pin
  // first, set pin to GPIO by using MUX = 1 in PCR register
  // second, select a GPIO pin
  // "&= ~PORT_PCR_MUX_MASK" below is used to clear bits in the MUX field
  // PORT_PCR_MUX(1) is to configure a port as a GPIO pin and
  // use |= to leave other bits unchanged

  PORTB->PCR[RED_LED_POS] &= ~PORT_PCR_MUX_MASK;
  PORTB->PCR[RED_LED_POS] |= PORT_PCR_MUX(1); // Set PTB18 pin mux to GPIO

  // Set ports to outputs: Port Data Direction Register (PDDR)
  PTB->PDDR |= MASK(RED_LED_POS);

  //Turn off LEDs: Port Set Output Register (PSOR)

  PTB->PSOR = MASK(RED_LED_POS);

  //end of configuration code

  //flash LED repeatedly
```

Figure 3: Third Snippet

```
  //flash LED repeatedly

  while(1) {
    red();
  }

}
```

Figure 4: Fourth Snippet

```c
#ifndef GPIO_DEFS_H
#define GPIO_DEFS_H

//Define a macro for left shifting operation;
// UL: Unsigned Long (data type)

#define MASK(x) (1UL << (x))

// Freedom KL25Z LEDs
#define RED_LED_POS (18)    // on port B
#define GREEN_LED_POS (19)    // on port B
#define BLUE_LED_POS (1)    // on port D


#endif
```

Figure 5: gpio_defs header file