Embedded Systems
Postlab

By Rajendra Singh (111601017), final year, CSE, IIT PALAKKAD

30 August 2019

Objective

To learn how to setup and operate the General Purpose Input/Output (GPIO) pins of the KL25Z Microcontroller using ARM Cortex M0+ assembly programming.

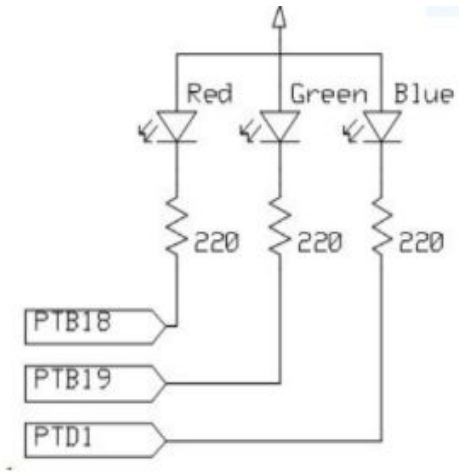
Abstract

In this lab, we tried to setup LED of different colours using the pins of the FRDM-KL25Z board. The board provides 53 pins which can be used for assigned peripheral special functions (e.g. UART IO, DAC outputs, ADC inputs, etc... For the above purpose, we need to configure the pins to work as General Purpose Input / Output Pins (GPIO) pins. Pins are grouped into five ports labeled A through E. The architecture allows for ports to have up to 32-bits but some ports have less than 32-bits due to limitations on the number of pins available in the device package. Once the pins are initiated as GPIO, then the pins can be masked to give different combinations of red, green, blue.

Introduction

In this lab, Keil microcontroller development kit (MDK) featuring µVision IDE will be used as the software development tool set for the Cortex M0+ processor. Keil µVision IDE combines project management, run-time environment, build facilities, source code editing, and program debugging in a single powerful environment. Through this software, we can run the code line-by-line and visualize the changes in the board. In this lab, we are given a task to blink red/green/blue led one after other.

There is an LED on the FRDM-KL25Z is a tri-color red/green/blue (RGB LED) device. Red, Green and Blue colour depends on the pin **PTB18, PTB19 and PTD1** respectively. Using the above instructions, these pins acts as GPIO. To turn on LED, GPIOs must be driven low and vice-versa as per circuit shown below. Here, in this circuit the common anode is tied to VDD. The three cathodes are tied through resistors to GPIOs as shown below. Because the LED is wired with the common anode to VDD, the GPIOs must be driven low to run on the LED color and driven high to turn off the LED.



LED Color	GPIO
RED	PTB18
GREEN	PTB19
BLUE	PTD1

7 Combinations formed:

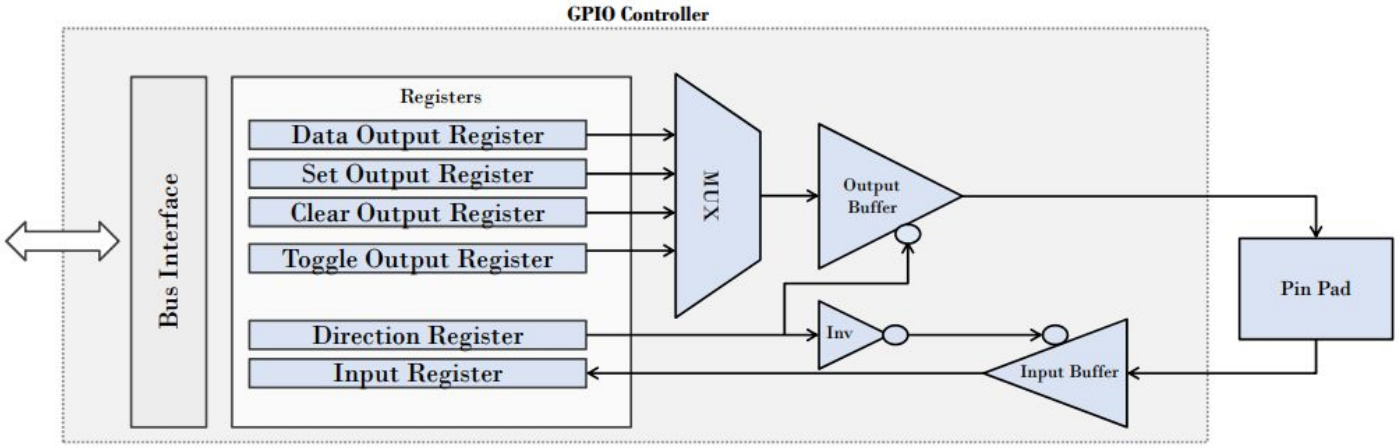
COLOUR	PIN PTB18	PIN PTB19	PIN PTD1
Red	Yes	-	-
Green	-	Yes	-
Blue	-	-	Yes
Yellow (Red + Green)	Yes	Yes	-
Magenta (Red + Blue)	Yes	-	Yes
Cyan (Green + Blue)	-	Yes	Yes
White (Red+Green+Blue)	Yes	Yes	Yes

Table 4.1

Foremost, to turn on or off a LED, these three pins are required to be configured as GPIO - output. To configure a pin as GPIO, three steps are required to be followed:

1. Clock for each port has to be enabled i.e. changing value in the register SIM_SCGC5
2. Making required pin as GPIO i.e. changing value in the register PORTx_PCRn, where **'x'** is the port and **'n'** is the pin
3. Making required pin as Input or Output i.e. changing value in the register GPIOx_PDDR , where **'x'** is the port.

If the GPIO is configured as an input, the state of the pin can be read using the GPIOx_PDIR register. If the GPIO is configured as an output, the state of pin can be controlled with GPIOx_PDOR , GPIOx_PSOR, GPIOx_PCOR and GPIOx_PTOR registers.



For port 'x',

GPIOx_PDOR:	Data Output Register
GPIOx_PSOR:	Set Output Register
GPIOx_PCOR:	Clear Output Register
GPIOx_PTOR:	Toggle Output Register
GPIOx_PDIR:	Data Input Register
GPIOx_PDDR:	Data Direction Register

Here, we are configuring the GPIO as an output. To turn on the LED, we will make the required pin as high in GPIOx_PCOR register and similarly, to turn off the LED, we will make that pin high in GPIOx_PSOR register. Similarly, the pin can toggle between high and low by enabling GPIOx_PTOR register.

EXERCISE 1

This aim of this exercise is to illustrate the use of GPIO pins as outputs. In this problem, GPIO pins are used to control the blinking of the LED of the microcontroller.

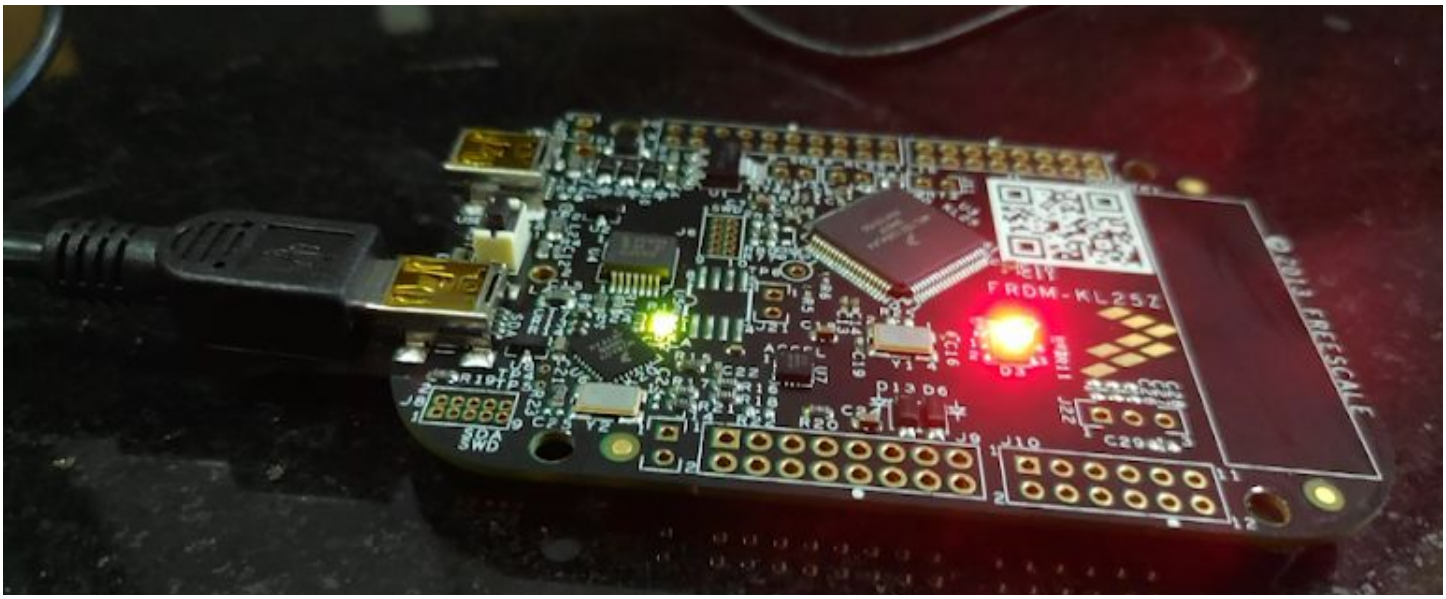
=> Executed the below code, and analyse how the various registers (including PORT, GPIO, SIM general purpose registers) change.
Below is well commented code where for each color led we written the helper functions like initialise, on, off and toggle. What each function does is explained in below code.

```c
#include //INCLUDING LIBRARY
//====================red B18============================//
//=======init========//
void led_red_init(){
  SIM->SCGC5 |=(1<<10); //TO ACTIVATE PORT B OR ACTIVE PORT B CLOCK
  //SET 8,9,10 = 001 TO GPIO
  PORTB->PCR[18]|=(1<<8); //SETTING 8TH BIT TO 1
  PORTB->PCR[18]&=0xFFFFF9FF; //SETTING 9TH, 10TH BIT TO 0, OTHER UNCHANGED
  PTB->PDDR |= (1<<18); //18TH BIT = 1, TO ACTIVATE 18 PIN
}
//=======ON========//
void led_red_on(){ //!!! on on low
  PTB->PCOR |= (1<<18 ); //CLEAR 18PIN VALUE
}
//=======OFF========//
void led_red_off(){
  PTB->PDOR |= (1<<18); //SET 18TH PIN VALUE
}
//=======TOGGLE========//
void led_red_toggle(){
  PTB->PTOR |= (1<<18); //TOGGLE 18TH PIN VALUE
}
//============================green B19============================//
//=======init========//
void led_green_init(){
  SIM->SCGC5 |=(1<<10); //TO ACTIVATE PORT B
  PORTB->PCR[19]|=(1<<8); //SETTING 8TH BIT TO 1
  PORTB->PCR[19]&=0xFFFFF9FF; //SETTING 9TH, 10TH BIT TO 0, OTHER UNCHANGED
  PTB->PDDR |= (1<<19); //19TH BIT = 1
}
//=======ON========//
void led_green_on(){
  PTB->PCOR |= (1<<19 ); //CLEAR 19PIN VALUE
}
//=======OFF========//
void led_green_off(){
  PTB->PDOR |= (1<<19); //SET 19 PIN VALUE
}
//=======TOGGLE========//
void led_green_toggle(){
  PTB->PTOR |= (1<<19); //TOGGLE 19TH PIN VALUE
}
//============================blue D1============================//
//=======init========//
void led_blue_init(){
  SIM->SCGC5 |=(1<<12); //TO ACTIVATE PORT D
  PORTD->PCR[1]|=(1<<8); //SETTING 8TH BIT TO 1
  PORTD->PCR[1]&=0xFFFFF9FF; //SETTING 9TH, 10TH BIT TO 0, OTHER UNCHANGED
  PTD->PDDR |= (1<<1); //1TH BIT = 1
}
//=======ON========//
void led_blue_on(){
  PTD->PCOR |= (1<<1 ); //CLEAR 1ST PIN VALUE
}
//=======OFF========//
void led_blue_off(){
  PTD->PDOR |= (1<<1); //SET 1ST PIN VALUE
}
//=======TOGGLE========//
void led_blue_toggle(){
  PTD->PTOR |= (1<<1); //TOGGLE 1ST PIN VALUE
}

//============================DELAY============================//
void delay(long long int d){
  while(d--);
}
//============================MAIN============================//
int main(void){
  SystemCoreClockUpdate(); //updating clock from PLL

  long long int n;          //NOUMBER OF BLINK
```

C:\Users\student\Documents\EmebbedSystem\eslab4\main.c

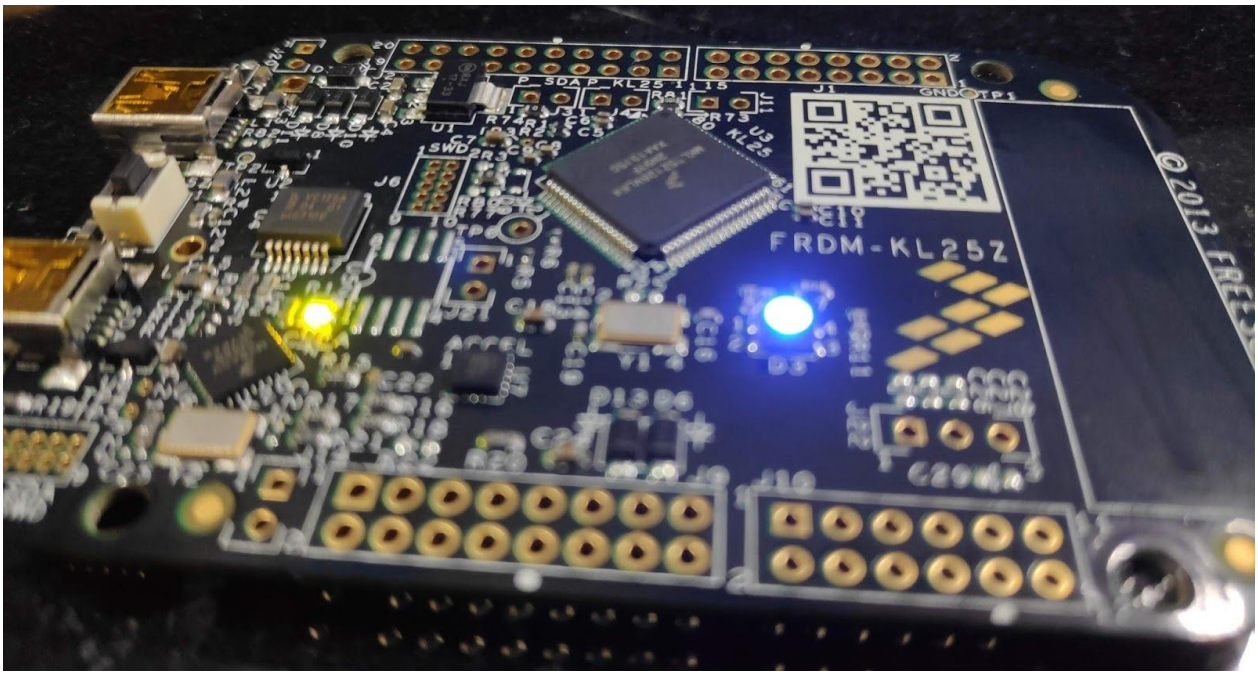
```
 73
 74      //INIT ALL LED
 75      led_red_init();
 76      led_green_init();
 77      led_blue_init();
 78
 79      // ========== BLINK RED FOR 100 TIMES ===========//
 80      led_red_on();
 81      n = 1e2;
 82      while(n--){
 83          led_red_toggle();
 84          delay(1e6);
 85      }
 86      led_red_off();
 87
 88      // ========== BLINK GREEN FOR 100 TIMES ===========//
 89      led_green_on();
 90      n = 1e2;
 91      while(n--){
 92          led_green_toggle();
 93          delay(1e6);
 94      }
 95      led_green_off();
 96
 97      // ========== BLINK BLUE FOR 100 TIMES ===========//
 98      led_blue_on();
 99      n = 1e2;
100      while(n--){
101          led_blue_toggle();
102          delay(1e6);
103      }
104      led_blue_off();
105
106      // ========== BLINK ALL ONE AFTER OTHER FOR 100 TIMES ===========//
107      n = 1e2;
108      while(n--){
109          led_red_on();
110          delay(1e6);
111          led_red_off();
112          delay(1e6);
113
114          led_green_on();
115          delay(1e6);
116          led_green_off();
117          delay(1e6);
118
119          led_blue_on();
120          delay(1e6);
121          led_blue_off();
122          delay(1e6);
123      }
124
125
126
127  }
```

Red:



Green:

- Blue



Conclusion

- Setting up correct amount of Delay plays an important role to visualize. For blinking, different delays can be given. A very small delay won't be seen easily.
- For different colours, masking has to be done for every single colour it has. It was rather fun to make different combinations.

Reference

1) Google
2) Slides on GPIOs, given in this Course
3) **Cortex M0+ Technical Reference Manual:**

   This document describes the functionality and the effects of functional options on the behavior of the Cortex M0+ processor.
   http://infocenter.arm.com/help/topic/com.arm.doc.ddi0484b/DDI0484B_cortex_m0p_r0p 0_trm.pdf