<<IERG3810>>

<<Microcontroller and Embedded Systems Laboratory>>

Report on Experiment <<1>>

<<General Purpose Inputs/Outputs (GPIO) >>

Group: 19

Member: Chan Kai Yin (CUID: 1155124983)

Ip Tsun Yu (CUID: 1155144668)

Submission Date: 11-02-2022

Disclaimer

I declare that the assignment here submitted is original except for source material explicitly acknowledged, and that the same or related material has not been previously submitted for another course. I also acknowledge that I am aware of University policy and regulations on honesty in academic work, and of the disciplinary guidelines and procedures applicable to breaches of such policy and regulations, as contained in the website http://www.cuhk.edu.hk/policy/academichonesty/

Jesse	Chan Kai Yin	11-02-2022
Signature	Name	Date
Derek	Ip Tsun Yu	11-02-2022
Signature	Name	Date

I. OBJECTIVES

To study the setting of GPIOs as inputs and outputs.

To study the project in C language with Standard Firmware Library for registers' settings.

Create own libraries for project board.

II. DATA ANALYSIS

Experiment 1.1

```
#include "stm32f10x.h"

void Delay(u32 count)
{
	u32 i;
	for ( i =0; i<count; i++);

}

int main(void)
{

	GPIO_InitTypeDef GPIO_InitStructure;
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init (GPIOB, & GPIO_InitStructure);
	GPIO_SetBits(GPIOB, GPIO_Pin_5);
	while(1){

		GPIO_ResetBits(GPIOB, GPIO_Pin_5);
		Delay(7000000);
		GPIO_SetBits(GPIOB, GPIO_Pin_5);
		Delay(7000000);

	}

}
```

The KED DS0 is blink. Setting the delay function with 7000000, it is close to 1s in real time.

Experiment 1.2

DS0 (Led 0) lit up when pressing the key2:

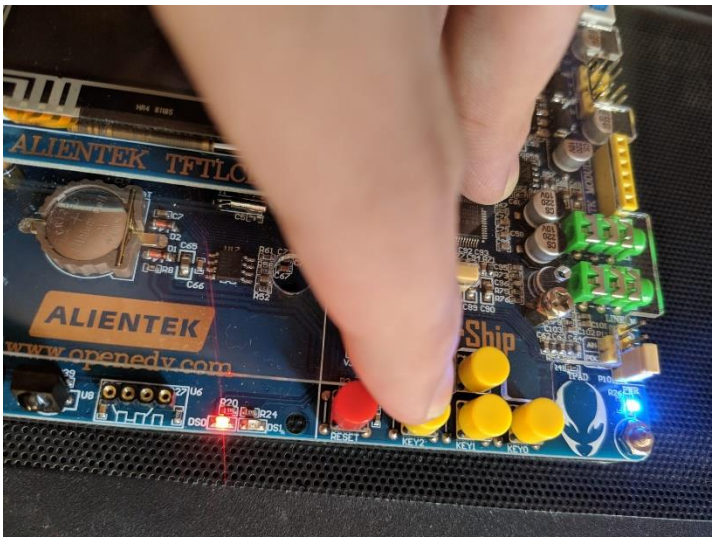

Figure 1 Pressing Key2

```
#include "stm32f10x.h"

void Delay(u32 count)
{
	u32 i;
	for ( i =0; i<count; i++);

}
int main(void)
{
	GPIO_InitTypeDef GPIO_InitStructure;
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE);
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init (GPIOB, & GPIO_InitStructure);
	/* Conf Key, GPIO Pin */
	RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE, ENABLE);
	GPIO_InitStructure.GPIO_Pin = GPIO_Pin_2;
	GPIO_InitStructure.GPIO_Mode = GPIO_Mode_IPU;
	GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
	GPIO_Init (GPIOE, & GPIO_InitStructure);
	while(1)
	{
		if (!GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_2))
	{

	  GPIO_ResetBits(GPIOB, GPIO_Pin_5);

	}
	else
	{
                                GPIO_SetBits(GPIOB, GPIO_Pin_5);
	}
	 }
}
```

Experiment 1.3

```
#include "stm32f10x.h"

void Delay(u32 count)
{
	u32 i;
	for ( i =0; i<count; i++);
}
int main(void)
{
	RCC ->APB2ENR |= 1 << 3;
	GPIOB->CRL &= 0xFF0FFFFF;
	GPIOB->CRL |= 0x00300000;
	while(1)
	{
		GPIOB->BRR = 1 << 5;
		Delay(1000000);
		GPIOB->BSRR = 1 << 5;
		Delay(1000000);
	}


}
```

Having the same effect as in Exp1.1. The ds0 Led 0 is blinking.

Experiment 1.4

```
#include "stm32f10x.h"

void Delay(u32 count)
{
	u32 i;
	for ( i =0; i<count; i++);

}
```

```
int main(void)
{
	//	led ds0: Pin 5
	RCC ->APB2ENR |= 1 << 3;
	GPIOB->CRL &= 0xFF0FFFFF;
	GPIOB->CRL |= 0x00300000;

	//	led ds1 : Pin 5
	RCC ->APB2ENR |= 1 << 6;
	GPIOE->CRL &=0xFF0FFFFF;
	GPIOE->CRL |=0x00300000;
	GPIOE->BSRR = 1 << 5;

	//	key1 : Pin 3
	RCC ->APB2ENR |= 1 << 6;
	GPIOE->CRL &= 0xFFFF0FFF;
	GPIOE->CRL |= 0x00008000;
	GPIOE->BSRR|=1<<3;

	//	key2 : Pin 2
	RCC ->APB2ENR |= 1 << 6;
	GPIOE->CRL &= 0xFFFFF0FF;
	GPIOE->CRL |= 0x00000800;
	GPIOE->BSRR|=1<<2;

	// key_up : Pin 0
	RCC ->APB2ENR |= 1 << 2;
	GPIOA->CRL &=0xFFFFFFF0;
	GPIOA->CRL |=0x00000008;



	//	buzzer : Pin 8
	RCC ->APB2ENR |= 1 << 3;
	GPIOB->CRH &=0xFFFFFFF0;
	GPIOB->CRH |=0x00000003;

	while(1)
```

```
{

    // check key2 input and trigger led ds0
    if (!(GPIOE -> IDR & 1 << 2))
    {
        GPIOB->BRR = 1 << 5;
    }
    else
    {
        GPIOB ->BSRR = 1 << 5;
    }

    //check key 1 and trigger led ds1
    if (!(GPIOE -> IDR & 1 << 3))
    {
        if (GPIOE -> ODR & 1 << 5){ // check ds1 state
            GPIOE -> BRR = 1 << 5;
        }
        else{
            GPIOE -> BSRR = 1 << 5;
        }
        Delay(5000000);
    }

    // key_up
    if (GPIOA -> IDR & 1 << 0)
    {
        if (GPIOB -> ODR & 1 << 8){
            GPIOB -> BRR = 1 << 8;
        }
        else{
            GPIOB -> BSRR = 1 << 8;
        }
        Delay(5000000);
    }

}
```

```
}
```

Init state:

Figure 2 Do nothing

LED DS0 lit when KEY2 is pressed down:

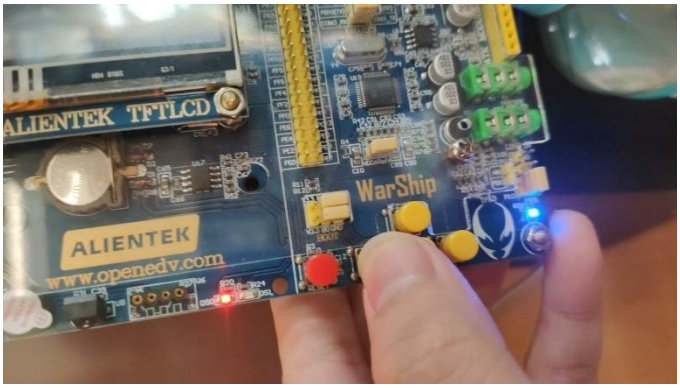

Figure 3 Pressing Key2

KEY1 toggles the LED DS1 on/off:

Figure 4 Ds1 is on after toggle

KEY_UP toggles the buzzer on off. Using the same logic as the Key1 toggles led ds1, difficult to demo without sound.

Experiment 1.5

```
#include "stm32f10x.h"
#include "IERG3810_LED.h"
#include "IERG3810_Buzzer.h"
#include "IERG3810_KEY.h"

void Delay(u32 count)
{
	u32 i;
	for ( i =0; i<count; i++);

}

int main(void)
{
	IERG3810_LED_Init();
	IERG3810_KEY_Init();
	IERG3810_Buzzer_Init();


	while(1)
	{

		// check key2 input and trigger led ds0
		if (!(GPIOE -> IDR & 1 << 2))
		{
			GPIOB->BRR = 1 << 5;
		}
		else
		{
			GPIOB ->BSRR = 1 << 5;
		}

		//check key 1 and trigger led ds1
		if (!(GPIOE -> IDR & 1 << 3))
		{
```

```
                if (GPIOE -> ODR & 1 << 5){ // check ds1 state
                    GPIOE -> BRR = 1 << 5;
                }
                else{
                    GPIOE -> BSRR = 1 << 5;
                }
                Delay(5000000);
            }

            // key_up
            if (GPIOA -> IDR & 1 << 0)
            {
                if (GPIOB -> ODR & 1 << 8){
                    GPIOB -> BRR = 1 << 8;
                }
                else{
                    GPIOB -> BSRR = 1 << 8;
                }
                Delay(5000000);
            }
        }
}
```

Create three files named IERG3810_KEY, IERG3810_Buzzer and IERG3810_LED. Have the same effect as Exp1.4.

III. DISCUSSION

Finding port problem:

For Exp 1.2 – 1.4, finding the corresponding pin number and GOIP port required reading the manual of the board.

In Exp 1.3 Register setting:

RCC ->APB2ENR |= 1 << 3

RCC clock of Port-B enable is bit-3 of RCC_APB2ENR

GPIOB->CRL &= 0xFF0FFFFF

Reset state for GPIOB-pin 5, it cleans pin 5 by &= operation and keep other

configuration unchanged.
GPIOB->CRL |= 0x00300000
3 = 0b0011. Set GPIOB-pin 5 in output mode with last two bits(11) and set it as general purpose output push-pull in output mode with first two bits(00)

GPIOB->BRR = 1 << 5
Bit reset in GPIOB-pin 5. 1 << 5 is refer as 0b0000 0000 0000 0000 0000 0000 0010 0000. This set GPIOB-pin5 to low.

GPIOB->BSRR = 1 << 5
Bit set in GPIOB-pin 5. 1 << 5 is refer as 0b0000 0000 0000 0000 0000 0000 0010 0000. This set GPIOB-pin5 to high.

Key bouncing problem in Exp 1.4:
I add a delay function for prevent the retrigger within a very short time. Compare to using hardware de-bouncing to solve the problem, using the software de-bouncing is much cheaper and easier.

IV. SUMMARY

We learn the setting of GPIOs as inputs and outputs. Using C language with Standard Firmware Library for registers' settings to make different operations. Learn to transfer C language to register settings. Create my own libraries for project board.

V. DIVISION OF WORK

<Lab work: Jesse & Derek, Report writing: Jesse>

VI. REFERENCES