<<IERG3810>>

<<Microcontroller and Embedded Systems Laboratory>>

Report on Experiment <<4>>

<<Interrupt >>

Group: 19

Member: Chan Kai Yin (CUID: 1155124983)

Ip Tsun Yu (CUID: 1155144668)

Submission Date: 18-03-2022

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	18-03-2022
Signature	Name	Date
Derek	Ip Tsun Yu	18-03-2022
Signature	Name	Date

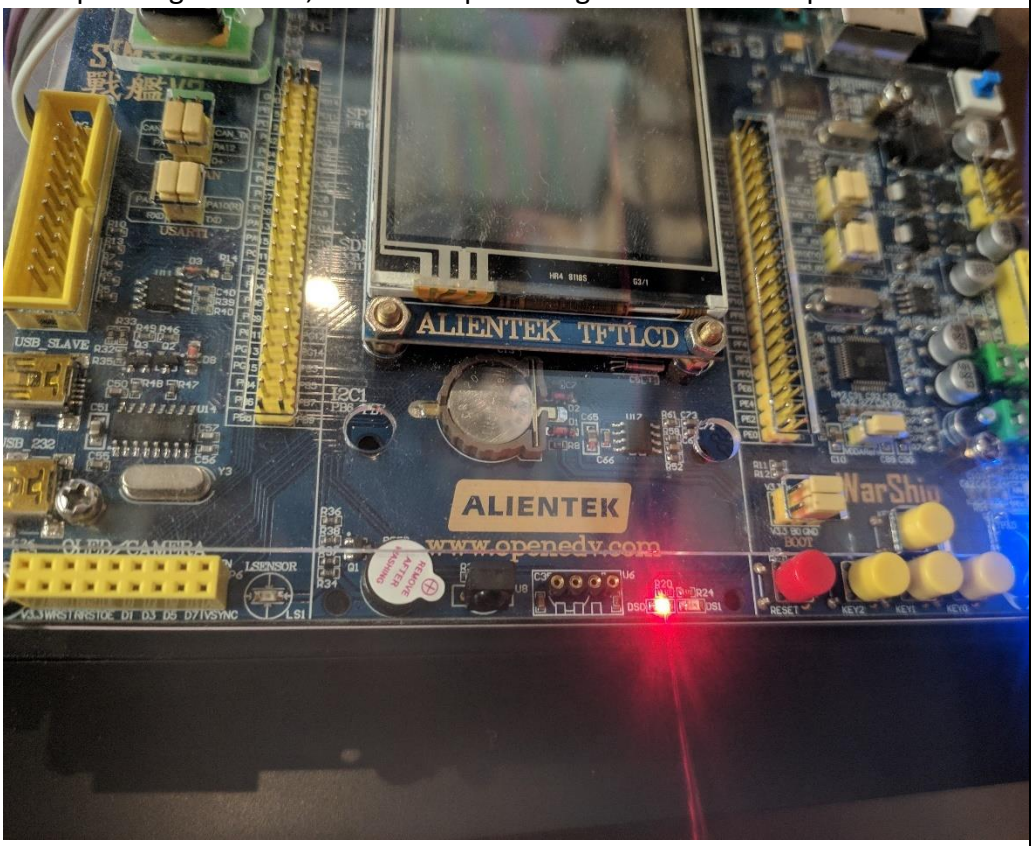
I. OBJECTIVES

- To study the External Interrupt setting of Cortex-M3.
- To study Synchronous protocol with PS2 keyboard.
- Using interrupt to receive key stroke data from a PS2 keyboard

II. DATA ANALYSIS

Experiment 4.1

After pressing the KEY2, the ds1 stop blinking and the ds0 lit up and blink:



```
#include "stm32f10x.h"
#include "IERG3810_LED.h"
#include "IERG3810_Buzzer.h"
#include "IERG3810_KEY.h"
#include "IERG3810_USART.h"
#include "IERG3810_Clock.h"
#include "FONT.H"
#include "SevenSegments.h"
#include "CFONT.H"

void IERG3810_clock_tree_init(void);
void IERG3810_USART2_init(u32, u32);
void IERG3810_USART1_init(u32, u32);
void Delay(u32);
void USART_print(u8, char *);

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

void IERG3810_key2_ExtInit(){
	RCC->APB2ENR |= 1 << 6;
	GPIOE ->CRL &= 0xFFFFF0FF;
	GPIOE ->CRL |= 0x00000800;
	GPIOE ->BSRR = 1 << 2;
	RCC->APB2ENR |= 0x01;
	AFIO ->EXTICR[0] &= 0xFFFFF0FF;
	AFIO ->EXTICR[0] |= 0x00000400;
	EXTI ->IMR |= 1 << 2;
	EXTI ->FTSR |= 1 << 2;
	//EXTI ->RTSR |= 1 << 2;

	NVIC ->IP[8] = 0x65;
	NVIC ->ISER[0] &= ~(1 << 8);
	NVIC ->ISER[0] |= (1 << 8);

}

void IERG3810_NVIC_SetPriorityGroup(u8 prigroup){
	u32 temp, temp1;
	temp1 = prigroup & 0x00000007;
	temp1 <<= 8;
	temp = SCB ->AIRCR;
	temp &= 0x0000F8FF;
	temp |= 0x05FA0000;
	temp |= temp1;
```

```
	SCB ->AIRCR = temp;
}

void EXTI2_IRQHandler(void){
	u8 i;
	for(i = 0; i < 10; i++){
		GPIOB ->BRR = 1 << 5;
		Delay(1000000);
		GPIOB ->BSRR = 1 << 5;
		Delay(1000000);
	}
	EXTI ->PR = 1 << 2;
}

u32 sheep = 0;
int main(void){
	IERG3810_clock_tree_init();
	//IERG3810_USART2_init(36,9600);
	IERG3810_LED_Init();
	IERG3810_NVIC_SetPriorityGroup(5);
	IERG3810_key2_ExtInit();
	//USART_print(2, "123457890");
	GPIOB ->BSRR = 1 << 5; //DS0 OFF

	while(1){
		//USART_print(2, "---ABCDEF    ");
		Delay(5000000);
		GPIOE ->BRR |= 1 << 5; //DS1 ON
		Delay(5000000);
		GPIOE ->BSRR |= 1 << 5;//DS1 OFF
		sheep++;
	}
}
```

Experiment 4.2

```
#include "stm32f10x.h"
#include "IERG3810_LED.h"
#include "IERG3810_Buzzer.h"
#include "IERG3810_KEY.h"
#include "IERG3810_USART.h"
#include "IERG3810_Clock.h"
#include "FONT.H"
#include "SevenSegments.h"
#include "CFONT.H"

void IERG3810_clock_tree_init(void);
void IERG3810_USART2_init(u32, u32);
void IERG3810_USART1_init(u32, u32);
void Delay(u32);
void USART_print(u8, char *);

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

void IERG3810_key2_ExtInit(){
	RCC->APB2ENR |= 1 << 6;
	GPIOE ->CRL &= 0xFFFFF0FF;
	GPIOE ->CRL |= 0x00000800;
	GPIOE ->BSRR = 1 << 2;
	RCC->APB2ENR |= 0x01;
	AFIO ->EXTICR[0] &= 0xFFFFF0FF;
	AFIO ->EXTICR[0] |= 0x00000400;
	EXTI ->IMR |= 1 << 2;
	EXTI ->FTSR |= 1 << 2;
	//EXTI ->RTSR |= 1 << 2;

	NVIC ->IP[8] = 0x65;
	NVIC ->ISER[0] &= ~(1 << 8);
	NVIC ->ISER[0] |= (1 << 8);

}

void IERG3810_NVIC_SetPriorityGroup(u8 prigroup){
	u32 temp, temp1;
	temp1 = prigroup & 0x00000007;
	temp1 <<= 8;
	temp = SCB ->AIRCR;
	temp &= 0x0000F8FF;
	temp |= 0x05FA0000;
	temp |= temp1;
	SCB ->AIRCR = temp;
```

```
}

void EXTI2_IRQHandler(void){
	u8 i;
	for(i = 0; i < 10; i++){
	 GPIOB ->BRR = 1 << 5;
	 Delay(1000000);
	 GPIOB ->BSRR = 1 << 5;
	 Delay(1000000);
	}
	EXTI ->PR = 1 << 2;
}

void EXTI0_IRQHandler(void){
	u8 i;
	for(i = 0; i < 10; i++){
	 GPIOE ->BRR = 1 << 5; // on
	 Delay(1000000);
	 GPIOE ->BSRR = 1 << 5; // off
	 Delay(1000000);
	}
	EXTI ->PR = 1 << 0;
}

void IERG3810_keyUP_ExtInit(){
	//PA0, KeyUp press  = high, EXTI-0
	RCC->APB2ENR |= 1 << 2;
	GPIOA ->CRL &= 0xFFFFFFF0;
	GPIOA ->CRL |= 0x00000008;
	GPIOA ->ODR |= 1 << 4;

	RCC->APB2ENR |= 0x1;
	AFIO ->EXTICR[0] &= 0xFFFFFFF0; //EXTI-0
	AFIO ->EXTICR[0] |= 0x00000000; //EXTI-0, 0000: PA[x] pin

	EXTI ->IMR |= 0x1; //0001 not mask on line 0
	EXTI ->FTSR |= 0x1; // Falling trigger enabled

	NVIC ->IP[6] = 0x75; //priority = 0x95
	NVIC ->ISER[0] &= ~(1 << 6);  //enable IRQ 6
	NVIC ->ISER[0] |= (1 << 6);      //IRQ 6
}

u32 sheep = 0;
int main(void){
	IERG3810_clock_tree_init();
	//IERG3810_USART2_init(36,9600);
	IERG3810_LED_Init();
	IERG3810_NVIC_SetPriorityGroup(5);
	IERG3810_key2_ExtInit();
	IERG3810_keyUP_ExtInit();
	//USART_print(2, "123457890");
```

```
	GPIOB ->BSRR = 1 << 5; //DS0 OFF

	while(1){
	 //USART_print(2, "   ---ABCDEF      ");
	 //Delay(5000000);
	 //GPIOE ->BRR |= 1 << 5; //DS1 ON
	 //Delay(5000000);
	 //GPIOE ->BSRR |= 1 << 5;//DS1 OFF
	 sheep++;
	}
}
```

Experiment 4.3

```
void IERG3810_keyUP_ExtInit(){
	//PA0, KeyUp press  = high, EXTI-0
	RCC->APB2ENR |= 1 << 2;
	GPIOA ->CRL &= 0xFFFFFFF0;
	GPIOA ->CRL |= 0x00000008;
	GPIOA ->ODR |= 1 << 4;

	RCC->APB2ENR |= 0x1;
	AFIO ->EXTICR[0] &= 0xFFFFFFF0; //EXTI-0
	AFIO ->EXTICR[0] |= 0x00000000; //EXTI-0, 0000: PA[x] pin

	EXTI ->IMR |= 0x1; //0001 not mask on line 0
	EXTI ->FTSR |= 0x1; // Falling trigger enabled

	NVIC ->IP[6] = 0x95; //priority = 0x95
	NVIC ->ISER[0] &= ~(1 << 6);     //enable IRQ 6
	NVIC ->ISER[0] |= (1 << 6); //IRQ 6
}
```

Experiment 4.4

```
Testing with different

void IERG3810_keyUP_ExtInit(){
	//PA0, KeyUp press  = high, EXTI-0
	RCC->APB2ENR |= 1 << 2;
	GPIOA ->CRL &= 0xFFFFFFF0;
	GPIOA ->CRL |= 0x00000008;
	GPIOA ->ODR |= 1 << 4;

	RCC->APB2ENR |= 0x1;
	AFIO ->EXTICR[0] &= 0xFFFFFFF0; //EXTI-0
	AFIO ->EXTICR[0] |= 0x00000000; //EXTI-0, 0000: PA[x] pin

	EXTI ->IMR |= 0x1; //0001 not mask on line 0
	EXTI ->FTSR |= 0x1; // Falling trigger enabled

	NVIC ->IP[6] = 0x35;
	NVIC ->ISER[0] &= ~(1 << 6);    //enable IRQ 6
	NVIC ->ISER[0] |= (1 << 6); //IRQ 6
}
```

Experiment 4.5

```
#include "stm32f10x.h"
#include "IERG3810_LED.h"
#include "IERG3810_Buzzer.h"
#include "IERG3810_KEY.h"
#include "IERG3810_USART.h"
#include "IERG3810_Clock.h"
#include "FONT.H"
#include "SevenSegments.h"
#include "CFONT.H"

void IERG3810_clock_tree_init(void);
void IERG3810_USART2_init(u32, u32);
void IERG3810_USART1_init(u32, u32);
void Delay(u32);
void USART_print(u8, char *);

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

void IERG3810_key2_ExtInit(){
```

```
	// EXTI-2
	RCC->APB2ENR |= 1 << 6;
	GPIOE ->CRL &= 0xFFFFF0FF;
	GPIOE ->CRL |= 0x00000800;
	GPIOE ->BSRR = 1 << 2;
	RCC->APB2ENR |= 0x01;
	AFIO ->EXTICR[0] &= 0xFFFFF0FF;
	AFIO ->EXTICR[0] |= 0x00000400;
	EXTI ->IMR |= 1 << 2;
	EXTI ->FTSR |= 1 << 2;
	//EXTI ->RTSR |= 1 << 2;

	NVIC ->IP[8] = 0x65;
	NVIC ->ISER[0] &= ~(1 << 8);
	NVIC ->ISER[0] |= (1 << 8);

}

void IERG3810_NVIC_SetPriorityGroup(u8 prigroup){
	u32 temp, temp1;
	temp1 = prigroup & 0x00000007;
	temp1 <<= 8;
	temp = SCB ->AIRCR;
	temp &= 0x0000F8FF;
	temp |= 0x05FA0000;
	temp |= temp1;
	SCB ->AIRCR = temp;
}

void EXTI2_IRQHandler(void){
	u8 i;
	for(i = 0; i < 10; i++){
		GPIOB ->BRR = 1 << 5;
		Delay(1000000);
		GPIOB ->BSRR = 1 << 5;
		Delay(1000000);
	}
	EXTI ->PR = 1 << 2;
}

void EXTI0_IRQHandler(void){
	u8 i;
	for(i = 0; i < 10; i++){
		GPIOE ->BRR = 1 << 5; // on
		Delay(1000000);
		GPIOE ->BSRR = 1 << 5; // off
		Delay(1000000);
	}
```

```
	EXTI ->PR = 1 << 0;
}



void IERG3810_keyUP_ExtInit(){
	//PA0, KeyUp press  = high, EXTI-0
	RCC->APB2ENR |= 1 << 2;
	GPIOA ->CRL &= 0xFFFFFFF0;
	GPIOA ->CRL |= 0x00000008;
	GPIOA ->ODR |= 1 << 4;

	RCC->APB2ENR |= 0x1;
	AFIO ->EXTICR[0] &= 0xFFFFFFF0; //EXTI-0
	AFIO ->EXTICR[0] |= 0x00000000; //EXTI-0, 0000: PA[x] pin

	EXTI ->IMR |= 0x1; //0001 not mask on line 0
	EXTI ->FTSR |= 0x1; // Falling trigger enabled

	NVIC ->IP[6] = 0x95; //priority = 0x95
	NVIC ->ISER[0] &= ~(1 << 6);    //enable IRQ 6 for EXTI0
	NVIC ->ISER[0] |= (1 << 6); //IRQ 6
}

void IERG3810_PS2key_ExtInit(){
	// PS2 data : PC10, PS2 CLK: PC11

	RCC->APB2ENR |= 1 << 4;
	GPIOC ->CRH &= 0xFFFF00FF;//PC10, PC11
	GPIOC ->CRH |= 0x00008800; // 1000

	GPIOC ->BSRR = 1 << 11; // set hight
	GPIOC ->BSRR = 1 << 10;

	RCC->APB2ENR |= 0x01;
	AFIO ->EXTICR[2] &= 0xFFFF0FFF; //EXTI11
	AFIO ->EXTICR[2] |= 0x00002000;// 0010: PC11 pin

	EXTI ->IMR |= 1 << 11; // Event request from Line x is not masked
	EXTI ->FTSR |= 1 << 11;// Falling trigger enabled

	NVIC ->IP[40] = 0x65; //priority for IRQ 40
	NVIC ->ISER[1] |= (1 << 8); //enable IRQ 40 for EXTI[15:10]
}

u32 sheep = 0;
u32 timeout = 10000;
u32 ps2key = 0;
```

```
u32 tmp = 0;
u32 ps2count = 0;
u8 ps2dataReady = 0;
u8 key_stack[2];


void EXTI15_10_IRQHandler(void){

	if (ps2count > 0 && ps2count < 9){ //1:8 , bit 0:7 data

		tmp = ps2key >>= 1; //right shift 1
		if ((GPIOC->IDR)&(1<<10)){
			tmp |= 0x80;
		}
	ps2key = tmp;
	ps2count++;
	}

	else {
		ps2count++;
	}


	Delay(10);
	EXTI->PR = 1 << 11;
}

int main(void){
	IERG3810_clock_tree_init();
	//IERG3810_USART2_init(36,9600);
	IERG3810_LED_Init();
	IERG3810_NVIC_SetPriorityGroup(5);
	IERG3810_key2_ExtInit();
	IERG3810_keyUP_ExtInit();
	IERG3810_PS2key_ExtInit();
	//USART_print(2, "123457890");
	GPIOB ->BSRR = 1 << 5; //DS0 OFF
	GPIOE ->BSRR = 1 << 5; //DS1 OFF


while(1){

		sheep++;
		if(ps2count >= 11){
			if(ps2key==0X73){
				GPIOB->BSRR = 1 << 21;
			}
			else{
```

```
                    GPIOB->BSRR = 1 << 5;
                }
                if(ps2key==0X74){

                    GPIOE->BSRR = 1 << 21;
                }
                else{
                    GPIOE->BSRR = 1 << 5;
                }
                EXTI ->PR = 1 << 11;
            }
            if(timeout == 0){
                timeout = 20000;
                ps2key = 0;
                ps2count = 0;
            }
            timeout --;

        }
}
```

III. DISCUSSION

Exp 4.1

<Question: Press KEY2 once after reset. Explain your observation.>

After pressing KEY2, ds1 stops blinking and ds0 lights up and blinks. After a while, ds0 will light up and ds1 will flash back. If I press KEY2 multiple times, ds0 will keep flashing until I stop and return ds1 to light up.

<Question: Describe the given programs and explain your observation in lab report.>

1. The IERG3810_key2_ExtiInit() function:

It configures GPIOE pin 2. Enable AFIO clock. Select interrupt source (EXTI-2). Select the source trigger with falling edge. Set the interrupt priority level as 0x65 and enable register with IRQ8.

2. IERG3810_NVIC_SetPriorityGroup() function:

It set priority group equal to 5. Only get the last 3 bit information from the priority group. Then 8 bits shifted left to set the priority group.

3. EXTI2_IRQHandler() function:

The function specifies the event when we trigger the KEY2 which is lit up the ds0.

Exp 4.2

<Question: Press KEY2 once after reset. Explain your observation>

The ds0 lights up and blinks.

<Question: Press KEYUP during KEY2 handler is running. Explain your observation.>

KEY2 will light ds0 red light and blink. Before it finishes, pressing KEYUP will not interrupt the ds0 process, it will wait for ds0 to finish and ds1 will blink. Because they belong to the same preempt level. KEYUP with priority 0x75 cannot interrupt KEY2 with priority 0x65.

<Question: Press KEY2 during KEYUP handler is running. Explain your observation.>

Reverse as the above question.

The KEYUP will lit up the ds1 green light and blinks. Before it finishes, pressing KEY2 will not interrupt the ds1 process, it will wait for ds1 to finish and ds0 will blink. Because they belong to the same subpriority. KEY2 with priority 0x65 cannot interrupt KEYUP with priority 0x75.

<Question: Does nested interrupt function as expected?>

No. Same conclusion as above, the setting of priority level in ExtInit() function has decided whether the interrupt can occur. When two level are the same, which are 0x65 and 0x75, no interrupt occurs.

Exp 4.3

<Question: Press KEY2 once after reset. Explain your observation.>

The ds0 lights up and blinks.

<Question: Press KEYUP during KEY2 handler is running. Explain your observation.>

Ds0 blink till it finishes then the ds1 blink.

<Question: Press KEY2 during KEYUP handler is running. Explain your

observation.>
Interrupt occurs. The ds1 blinks first, after pressing KEY2 the ds1 process stops and switches to ds0 blinking until done and then back to ds1.

<Question: Does nested interrupt function as expected?>
Yes. KEY2 with priority level 0x65 is higher than KEYUP with 0x95.

Exp 4.4
<Question: Press KEY2 once after reset. Explain your observation.>
The ds0 lights up and blinks.

<Question: Press KEYUP during KEY2 handler is running. Explain your observation.>
Interrupt occurs. The ds0 blinks first, after pressing KEYUP the ds0 process stops and switches to ds1 blinking until done and then back to ds0.

<Question: Press KEY2 during KEYUP handler is running. Explain your observation.>
Ds1 blink till it finishes then the ds0 blink.

<Question: Does nested interrupt function as expected?>
Yes. KEYUP with priority level 0x35 is higher than KEY2 with 0x65.

<Question: Explain your observations in your report when the KEYUP priority is set to 0x65, 0x75, 0x95 and 0x35.>
The KEY2 priority is set to 0x65. The KEYUP can interrupt the KEY2 when the priority is higher than the 0x65, which the number should be lower than the 0x65 which is 0x35.

<Question: How to set the pre-empty priority to two levels? What are the results of KEYUP priority when it is 0x65, 0x75, 0x35 or 0x95? Explain them in your lab report.>
Change the priority group to 6 which only have two pre-empty priority and 8 subpriority level. When priority level of KEY2 = 0x65, only KEYUP with 0x35 can interrupt.

IV. SUMMARY

We learn the External Interrupt setting of Cortex-M3

V. DIVISION OF WORK

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

VI. REFERENCES