IERG3810 Lab4 Interrupt

HU, Han Session C No board number (board bought by myself)

Experiment 4.1: EXTI-2 interrupt and on board KEY2

Observation of procedure 4.1.3:

LED DS0 will flash 10 times when KEY2 is pressed down (triggered at falling edge), while LED DS1 will just be flashing automatically (5 times slower than the DS0 flash frequency).

Program description:

In the given programs, we first setup the interrupt controller of STM32 with EXTI interrupt. Since KEY2 is used here, GPIOE is configured here to control the LED. Then AFIO is used to control external interrupt configuration register and source input. For NVIC, since it is the interrupt register, we set IP[8] as it corresponds to KEY2 and 0x65 is the respective priority level. For the `NVIC_SetPriorityGroup()` function, it sets the priority group of the interrupts and the value in SCB is store via `SCB->AIRCR=temp;`.

Experiment 4.2: Experiment 4.2: EXTI interrupt and on board KEYUP

Observation of procedure 4.2.6:

LED DS0 will flash 10 times when KEY2 is pressed down.

Observation of procedure 4.2.7:

LED DS1 will flash 10 times after LED DS0 finishing flashing 10 times.

Observation of procedure 4.2.8:

LED DS0 will flash 10 times after LED DS1 finishing flashing 10 times.

Explanation on the nested interrupt function:

The nested interrupt function didn't act as expected. This is due to the priority level set to KEY2 and KEYUP. The priority level of KEY2 is 0x65 and priority level of KEYUP is 0x75. The first two bits of both priority levels are 0x01. As PRIGROUP is set to be 5, these two priority level is in the same priority group. Therefore, the nested interrupt function will not be executed.

Experiment 4.3: Test interrupts priority (I)

Observation of procedure 4.3.3:

LED DS0 will flash 10 times when KEY2 is pressed down.

Observation of procedure 4.3.4:

LED DS1 will flash 10 times after LED DS0 finishing flashing 10 times.

Observation of procedure 4.3.5:

LED DS1 will first begin to flash. When KEY2 is pressed down, its interrupt will be handled and DS1 will stop flashing. Then, LED DS0 will begin to flash 10 times. After DS0 finishing flashing, DS1 will continue flashing until it flashes 10 times in total.

Explanation on the nested interrupt function:

In this experiment, the nested interrupt function works. The priority level of KEY2 is 0x65 and KEYUP is 0x95. Hence, KEY2 has a higher priority level. If KEYUP is pressed down when KEY2 interrupt is being handled, it will be pended until KEY2 handler is finished. However, if KEY2 is pressed down when KEYUP interrupt is handled, it will interrupt KEYUP handler. If KEY2 is pressed for more than once, KEYUP handler will be pended until all KEY2 interrupts are handled.

Experiment 4.4: Test interrupts priority (II)

Observation of procedure 4.4.3:

LED DS0 will flash 10 times when KEY2 is pressed down.

Observation of procedure 4.4.4:

LED DS0 will first begin to flash. When KEYUP is pressed down, its interrupt will be handled and DS0 will stop flashing. Then, LED DS1 will begin to flash 10 times. After DS1 finishing flashing, DS0 will continue flashing until it flashes 10 times in total.

Observation of procedure 4.4.5:

LED DS0 will flash 10 times after LED DS1 finishing flashing 10 times.

Explanation on the nested interrupt function:

In this experiment, the nested interrupt function works. The priority level of KEY2 is 0x65 and KEYUP is 0x35. Hence, KEYUP has a higher priority level. If KEY2 is pressed down when KEYUP interrupt is being handled, it will be pended until KEYUP handler is finished. However, if KEYUP is pressed down when KEY2 interrupt is handled, it will interrupt KEY2 handler. If KEYUP is pressed for more than once, KEY2 handler will be pended until all KEYUP interrupts are handled.

Explanation on different priority settings:

Smaller value represents higher priority level (i.e. 0x35 > 0x65 > 0x75 >0x95). When 0x35 priority level is set, it has the highest priority and hence can interrupt KEY2 but not vice versa. When 0x65 priority level is set, KEY2 and KEYUP have the same priority level, hence they cannot interrupt each other. When 0x75 or 0x95 is set, KEYUP has a lower priority than KEY2, therefore it cannot interrupt KEY2 but vice versa.

Explanation on 4.4.8 Question:

We can set the pre-empty priority to two levels by setting the PRIGROUP to 6. Under PRIGROUP 6 setting, then 0x35, 0x65 and 0x75 will have the same pre-empty priority. Hence, only if 0x95 is set can KEY2 interrupt KEYUP through nested interrupt. In other scenarios, they cannot interrupt each other.

Experiment 4.5: PS2 Keyboard

My design of `EXTI15_10_IRQHandler(void)`:

```
1  void EXTI15_10_IRQHandler(void){
2      // student design here
3      if(ps2count>0 && ps2count <9){ //check the data
   bits
4          ps2key |= (GPIOC->IDR & 0x00000400)>>(10-
   ps2count+1);
5      }
6      ps2count++;
7      Delay(10); //We found that the processor is too
   fast and get error.
8      // A short delay can eliminate the error.
9      EXTI->PR = 1<<11; //Clear this exception pending
   bit
10 }
```

The input data of PS2 keyboard is checked whether from 0 to 9 and store them in ps2key.

My design of `void IERG3810_PS2key_ExtiInit(void)`:

```
1  void IERG3810_PS2key_ExtiInit(void){
2      //PS2 Keyboard CLK at PC11, EXTI10:15, IRQ#40
3      RCC->APB2ENR|=1<<4;
4      GPIOC->CRH &= 0xFFFF0FFF;
5      GPIOC->CRH |= 0x00008000;
6      RCC->APB2ENR |= 0x01;
7      AFIO->EXTICR[2] &= 0xFFFF0FFF;
8      AFIO->EXTICR[2] |= 0x00002000;
9      EXTI->IMR |= 1<<11;
10     EXTI->FTSR |=1<<11;
11
12     NVIC->IP[40] = 0x95; //set priority of this
   interrupt
```

```
    NVIC->ISER[1] |= (1<<8);

    //PS2 Keyboard DAT at PC10
    RCC->APB2ENR|=1<<4;
    GPIOC->CRH &= 0xFFFFF0FF;
    GPIOC->CRH |= 0x00000800;
}
```

This function is set according to the respective connection through PC11 and PC10 of CLK and DAT of PS/2 keyboard.

My design of `int main(void)`:

```
int main(void){
    IERG3810_clock_tree_init();
    IERG3810_LED_Init();
    IERG3810_NVIC_SetPriorityGroup(5); //set PRIGROUP
    IERG3810_key2_ExtiInit(); //Init KEY2 as an interrupt input
    IERG3810_keyUP_ExtiInit(); //Init KEYUP as an interrupt input
    IERG3810_PS2key_ExtiInit(); //Init PS2 keyboard as an interrupt input
    DS0_OFF;
    DS1_OFF;
    while(1){
        //USART_print(2, " --- ABCDEF ");

        sheep++; //COUNT SHEEP
        //if PS2 keyboard received data correctly
        if(ps2count>=11){
            //EXTI->IMR &= ~(1<<11); optional, suspend interrupt
            //-- student design program here
      if(ps2key_prev==0x70&&ps2key==0x70){
        DS0_ON;
      }
      if(ps2key_prev==0x69&&ps2key==0x69){
        DS1_ON;
      }
      if(ps2key_prev==0xF0&&ps2key==0x70){
        DS0_OFF;
      }
      if(ps2key_prev==0xF0&&ps2key==0x69){
        DS1_OFF;
      }
            ps2key_prev=ps2key;
      ps2count=0;
      ps2key=0;
            EXTI->PR = 1<<11; //Clear this exception pending bit
```

```
            //EXTI->IMR |= (1<<11); optional, resume interrupt
        } //end of "if PS2 keyboard received data correctly"
        timeout--;

        if (timeout ==0){ //Clear PS2 keyboard data when timeout
            timeout=20000;
            ps2key=0;
            ps2count=0;
        } // end of "clear PS2 keyboard data when timeout"
    }
}
```

Here, four checks are conducted and the key 0 (0x70),1 (0x69) in the keyboard and the release signal (0xF0) are used. If a consecutive input of 0 is received, DS0 will be on. Similarly, if a consecutive input of 1 is received, DS1 will be on. If a combination of release and 0 is received, DS0 will be off. Similarly, if a combination of release and 1 is received, DS1 will be off.