EIE3810 Microprocessor System Design Laboratory

Laboratory Report #1

Name: 马腾飞

Student ID: 121090406

Date: Sep 28,2023

The Chinese University of Hong Kong, Shenzhen

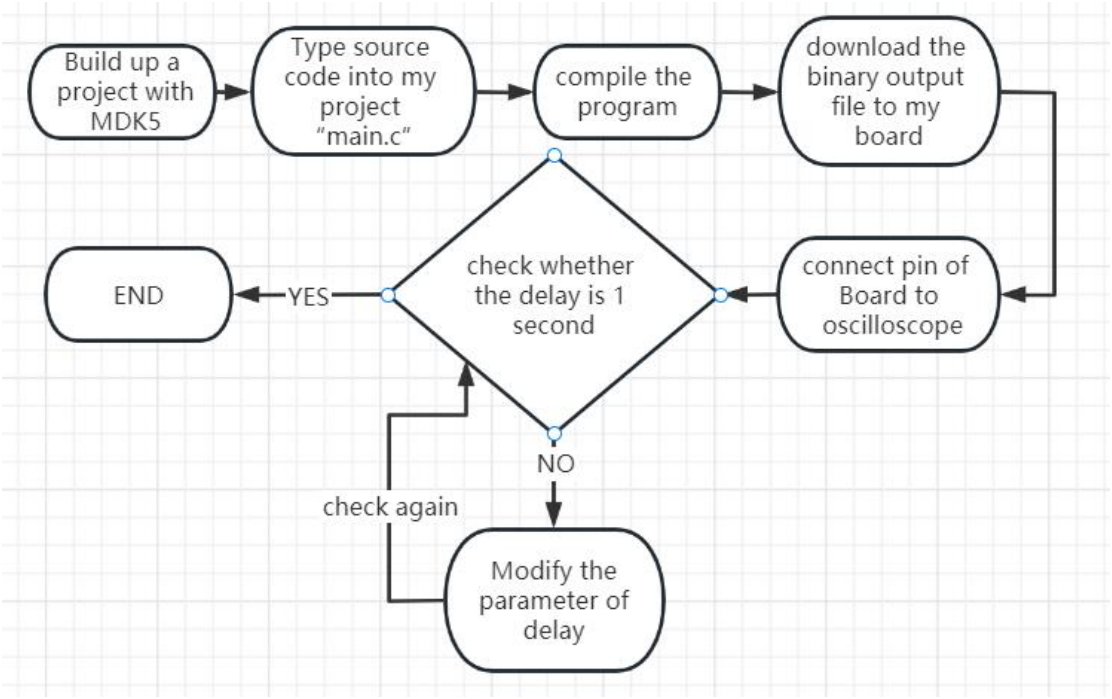
There are what experiments I have done during this lab as follows.

- Experiment 1: Set a GPIO as an output and drive a LED with standard peripheral library.
- Experiment 2: Read a key from GPIO input and drive an LED with a standard peripheral library
- Experiment 3: Set a GPIO as an output and drive an LED with register setting
- Experiment 4: Read a Key from GPIO input and drive an LED with register setting
- Experiment 5: Create my own library for the project board

1. Experiment 1

1.1 Design

1.1.1 Designed program flowchart

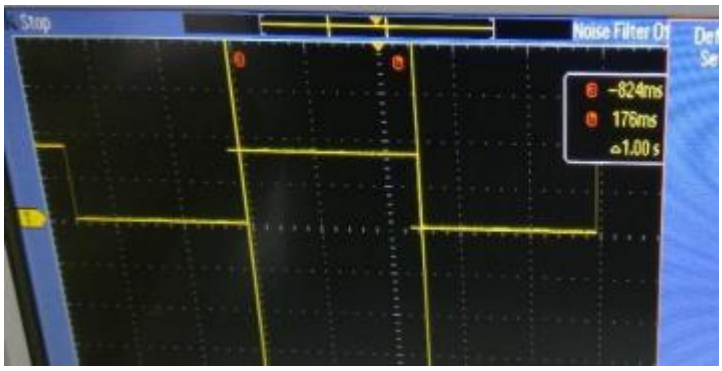


1.1.2 Explanation of source code

```
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);//enable PB
  GPIO_InitStructure.GPIO_Pin = GPIO_Pin_5;             //set as Pin5
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;     //set as 50MHz
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;      //set output mode as push-pull
  GPIO_Init(GPIOB, &GPIO_InitStructure);                //initialize PB5
  GPIO_SetBits(GPIOB, GPIO_Pin_5);                      //set PB5 as high
  while(1)
  {
    GPIO_ResetBits(GPIOB, GPIO_Pin_5);                  //set PB5 as low
    Delay(10000000);                                    //delay 1 second
    GPIO_SetBits(GPIOB, GPIO_Pin_5);                    //set PB5 as high
    Delay(10000000);                                    //delay 1 second
  }
}
```

a. PB5 connects to LED0.
b. PB5 low means LED0 on.
c. Choose 10000000 delay is 1 second.

1.2 Result



In the oscilloscope, there is periodical square wave with period 2 seconds. And the level changes every 1 second. The LED0 does as well.

1.3 Questions

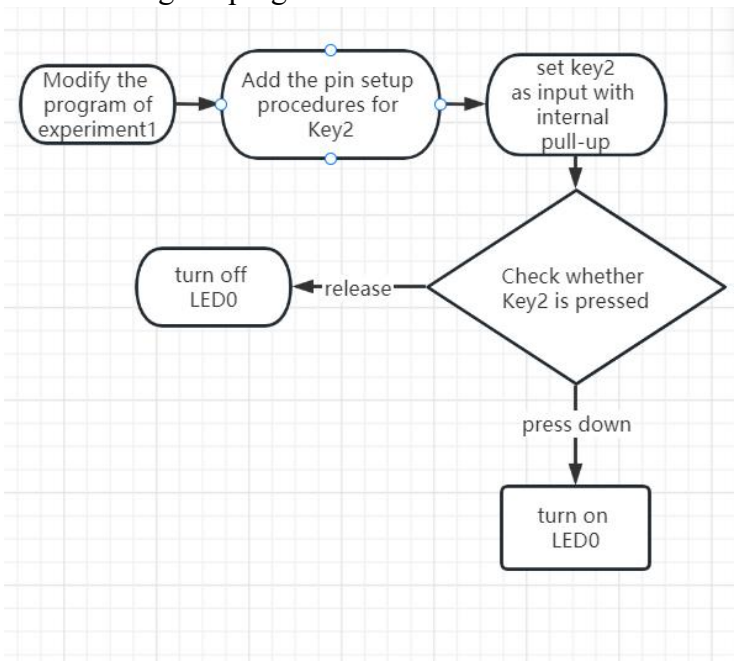
1.3.1 How do you validate that your error is less than 5%, i.e. the error in time periods for LED0 on and off should be less than 50ms?

1.3.2 Use oscilloscope. Coordinate two lines to align with two near edges, and check the time difference. If it is 1.00 sec, it is done. If not, assume the period changes linearly with delay and calculate the right delay. If still not, change delay little by little (eg. 100 by 100).

2. Experiment 2

2.1 Design

2.1.1 Designed program flowchart



2.1.2 Explanation of source code

```
int main(void)
{
  //Initialize PE2
  GPIO_InitTypeDef GPIO_PE2;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOE, ENABLE); //enable PE
  GPIO_PE2.GPIO_Pin = GPIO_Pin_2;
  GPIO_PE2.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_PE2.GPIO_Mode = GPIO_Mode_IPU;   // set PE2 as pull-up input
  GPIO_Init(GPIOB, &GPIO_PE2);

  //Initialize PB5
  GPIO_InitTypeDef GPIO_PB5;
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOB, ENABLE); //enable PB
  GPIO_PB5.GPIO_Pin = GPIO_Pin_5;
  GPIO_PB5.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_PB5.GPIO_Mode = GPIO_Mode_Out_PP;  //set PB5 as push-pull output
  GPIO_Init(GPIOB, &GPIO_PB5);
  GPIO_SetBits(GPIOB, GPIO_Pin_5); //set PB5 as high

  while(1)
  {
    if (GPIO_ReadInputDataBit(GPIOE, GPIO_Pin_2) == 0) //PE2 is connected to Key2, Pressing Key2 makes PE2 low.
    {
      GPIO_ResetBits(GPIOB, GPIO_Pin_5); //when pressing Key2, PB5 (LED0) is lit.
    }
    else
    {
      GPIO_SetBits(GPIOB, GPIO_Pin_5); //when releasing Key2, PB5(LED0) is off.
    }
  }
}
```

a. Initialize PE2 and PB5

b. Use a conditional statement to express the relation between LED0 and Key2

2.2 Result

When pressing Key2, LED0 is on. When releasing Key2, LED0 is off.

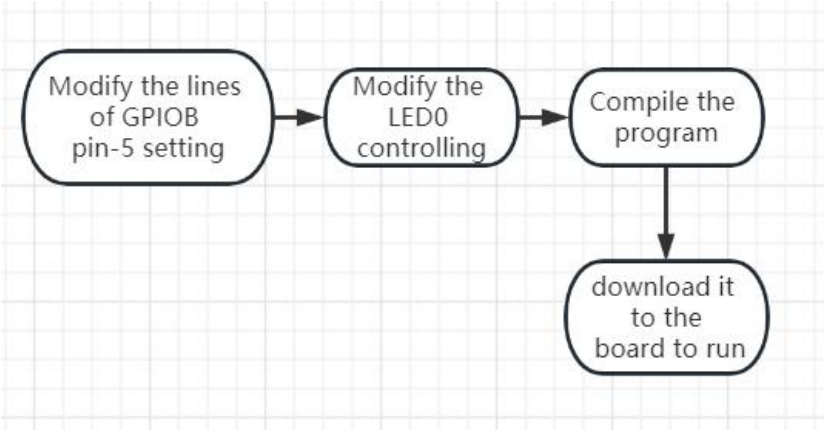
2.3 Questions

2.3.1 There no questions in the handout.

3. Experiment 3

3.1 Design

3.1.1 Designed program flowchart



3.1.2 Explanation of source code

```
int main(void)
{
  RCC->APB2ENR|=1<<3; //enable PB
  //set PB5 as push-pull output
  GPIOB->CRL &=0xFF0FFFFF;
  GPIOB->CRL|=0x00300000;

  while(1)
  {
    GPIOB->BRR=1<<5;//reset PB5(LED0)
    Delay(10000000);//delay 1 sec
    GPIOB->BSRR=1<<5;//set PB5(LED0)
    Delay(10000000);//delay 1 sec
  }
}
```

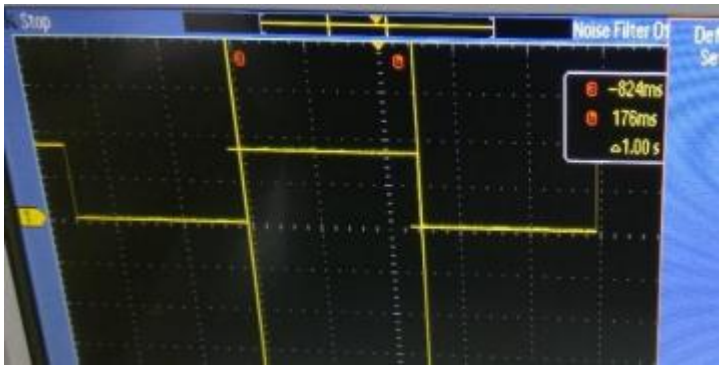
a. APB2ENR is used to enable GPIO, eg. Enable PB

b. CRL is to set output mode, in this case, we set the sixth(count from 0) port(from right) as 0b0011, which means push-pull output mode.

c. BRR means “reset”.

d. BSRR means “set”.

3.2 Result



In the oscilloscope, there is periodical square wave with period 2 seconds. And the level changes every 1 second. The LED0 does as well.

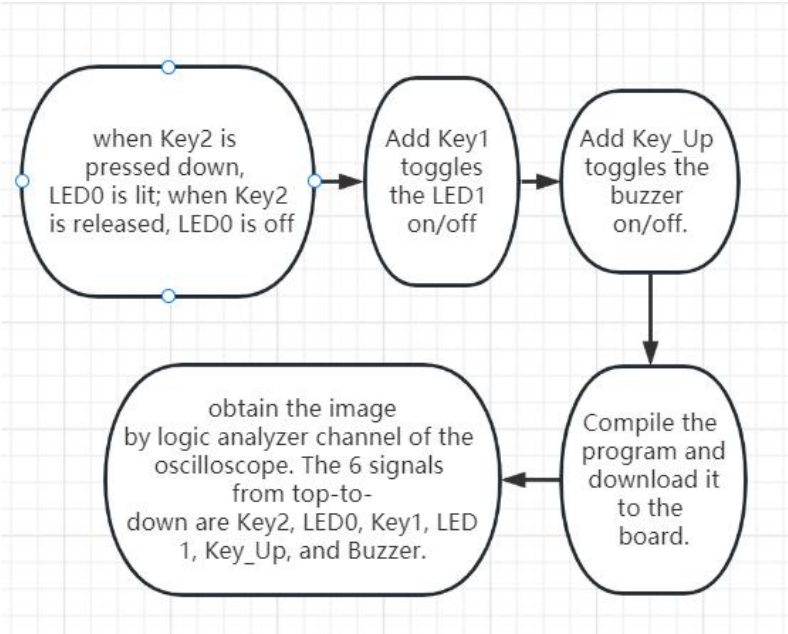
3.3 Questions

3.3.1 There no questions in the handout.

4. Experiment 4

4.1 Design

4.1.1 Designed program flowchart



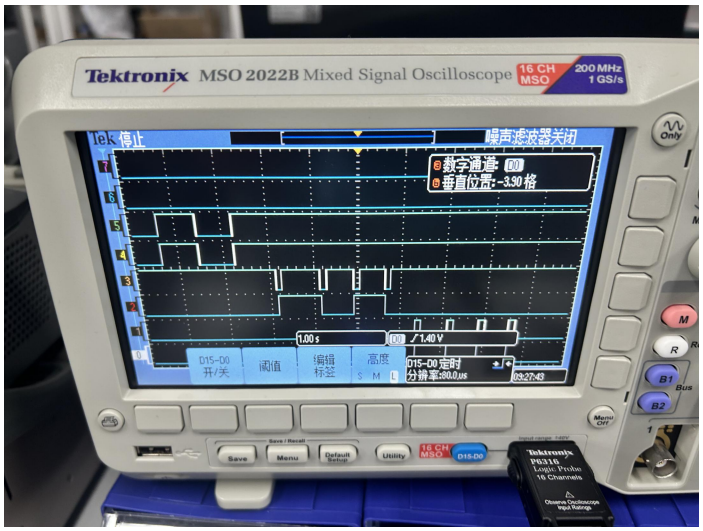
4.1.2 Explanation of source code

```
int main(void)
{
  //start
  RCC->APB2ENR|=1<<2;//PAstart
  RCC->APB2ENR|=1<<3;//PBstart
  RCC->APB2ENR|=1<<6;//PEstart

  //mode
  //PB5
  GPIOB->CRL &=0xFF0FFFFF;
  GPIOB->CRL|=0x00300000;//mode_out_pp
  //PE5
  GPIOE->CRL &=0xFF0FFFFF;
  GPIOE->CRL|=0x00300000;//mode_out_pp
  //PB8 buzzer
  GPIOB->CRH &=0xFFFFFFF0;
  GPIOB->CRH|=0x00000003;//mode_out_pp

//PE2
GPIOE->CRL &=0xFFFFF0FF;
GPIOE->CRL|=0x00000800;//mode_ipu
GPIOE->ODR |=1<<2;//PE2
//PE3
GPIOE->CRL &=0xFFFF0FFF;
GPIOE->CRL|=0x00008000;//mode_ipu
GPIOE->ODR |=1<<3;//PE3
//PA0
GPIOA->CRL &=0xFFFFFFF0;
GPIOA->CRL|=0x00000008;//mode_ipu
GPIOA->ODR |=1<<1;//PA0
```

```
while(1)
{
  //key2-LED0: PE2-PB5
  if (((GPIOE->IDR>>2)&0x1) == 1)//release Key2
  {
    GPIOB->BSRR|=1<<5;//PB5=1 : LED0 is off
  }
  else
  {
    GPIOB->BRR|=1<<5;//PB5=0 : LED0 is lit
  }

  //key1-LED1: PE3-PE5
  if (((GPIOE->IDR>>3)&0x1) == 0)//press Key1
  {
    for(int i = 0; i < 200000; i++);//delay for turbulence at the beginning of pressing

    if (((GPIOE->IDR>>3)&0x1) == 0)
    {
      //change status of LED1 when pressing Key1
      if ((GPIOE->ODR>>5&0x1) == 1) GPIOE->BRR|=1<<5;
      else GPIOE->BSRR|=1<<5;

      while(((GPIOE->IDR>>3)&0x1) == 0);//wait for continuous pressing

      for(int i = 0; i < 200000; i++);//delay for turbulence at the beginning of releasing
    }
  }

    //key_up-buzzer: PA0-PB8
    if (((GPIOA->IDR)&0x1) == 1)//press Key_up
    {
      for(int i = 0; i < 80000; i++);//delay for turbulence at the beginning of pressing

      if (((GPIOA->IDR)&0x1) == 1)
      {
        //change status of buzzer when pressing Key_up
        if ((GPIOB->ODR>>8&0x1) == 1) GPIOB->BRR|=1<<8;
        else GPIOB->BSRR|=1<<8;

        while(((GPIOA->IDR)&0x1) == 1);//wait for continuous pressing

        for(int i = 0; i < 80000; i++);//delay for turbulence at the beginning of releasing
      }
    }

  }
}
```

a. Initialize buzzer, keys, LEDs like experiment3
b. Use delay to deal with turbulence.
c. Notice the comments behind codes

4.2 Result



From top to down, they are Key2, LED0, Key1, LED1, Key_up and buzzer.
When pressing Key2, LED0 is on. When releasing Key2, LED0 is off.
Key1 toggles LED1, i.e. press and release Key1, and LED1 status is then flipped.
Key_up toggles buzzer, i.e. press and release Key_up, and buzzer status is then changed.

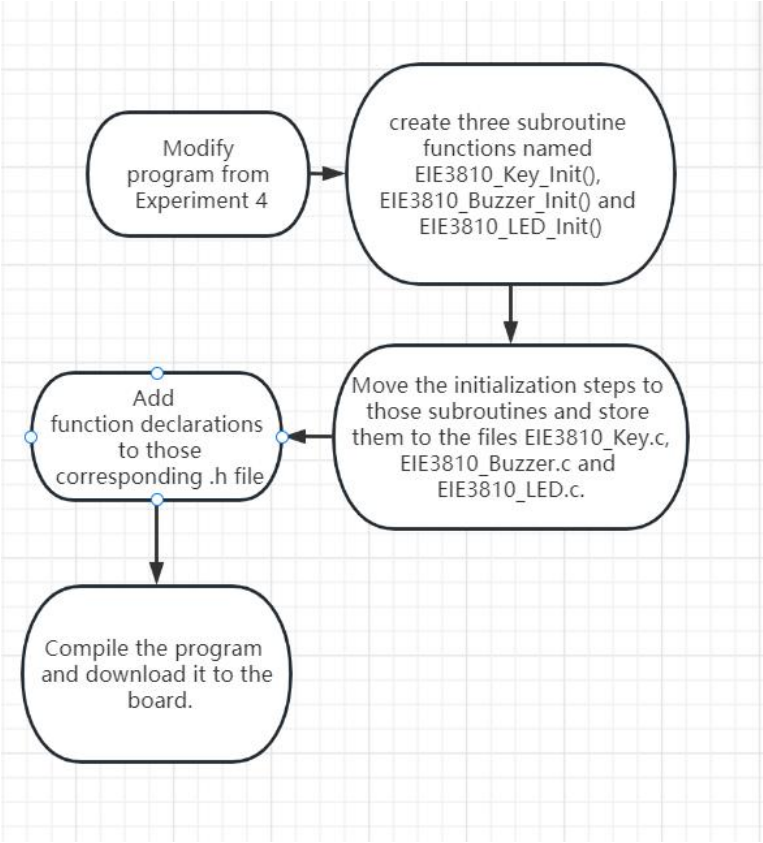
4.3 Questions

4.3.1 There no questions in the handout.

5. Experiment 5

5.1 Design

5.1.1 Designed program flowchart

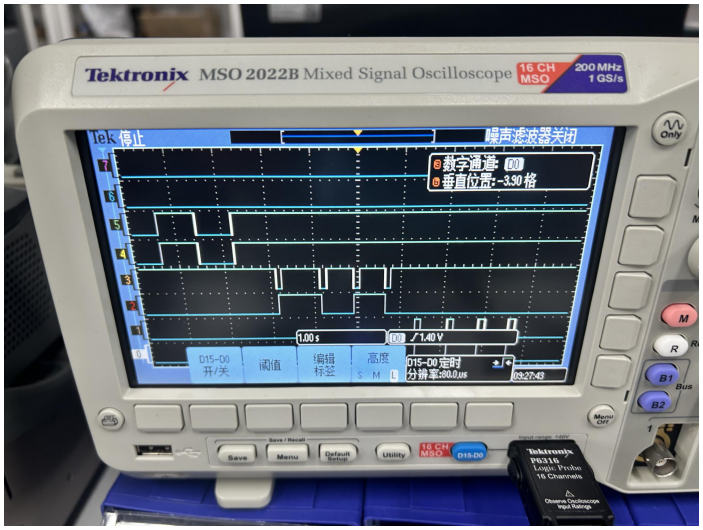


5.1.2 Explanation of source code

```
int main(void)
{
  //initialize device
  EIE3810_LED_Init();
  EIE3810_Buzzer_Init();
  EIE3810_Key_Init();

  while(1)
  {
    if(EIE3810_Read_Key2() == 1)//release Key2
    {
      EIE3810_TurnOff_LED0();//turn off LED0
    }
    else
    {
      EIE3810_TurnOn_LED0();//turn on LED0
    }

    if(EIE3810_Read_Key1() == 0)//press Key1
    {
      EIE3810_toggle_LED1();//toggle LED1
    }

    if(EIE3810_Read_KeyUp() == 1)//press Key_up
    {
      EIE3810_toggle_Buzzer();//toggle buzzer
    }
  }
}
```

The only difference to experiment4 is using subroutines. Copy the content to functions, so we can just use the functions directly.

5.2 Result



From top to down, they are Key2, LED0, Key1, LED1, Key_up and buzzer.
When pressing Key2, LED0 is on. When releasing Key2, LED0 is off.
Key1 toggles LED1, i.e. press and release Key1, and LED1 status is then flipped.
Key_up toggles buzzer, i.e. press and release Key_up, and buzzer status is then changed.

5.3 Questions

5.3.1 There no questions in the handout.

6. Conclusion

I first write codes to drive GPIO to light LED and noise buzzer with both standard library and register setting. It is very exciting and fantastic. I have learned how to use STM32F103ZET6 and firmware library. Besides, I understand how codes control the STM32 by registers. We can compare STM32 to computer. Buzzer is like sound equipment, LED is like screen, keys are like keyboard, my codes are like operating system. In some sense, I figure out how computer works.