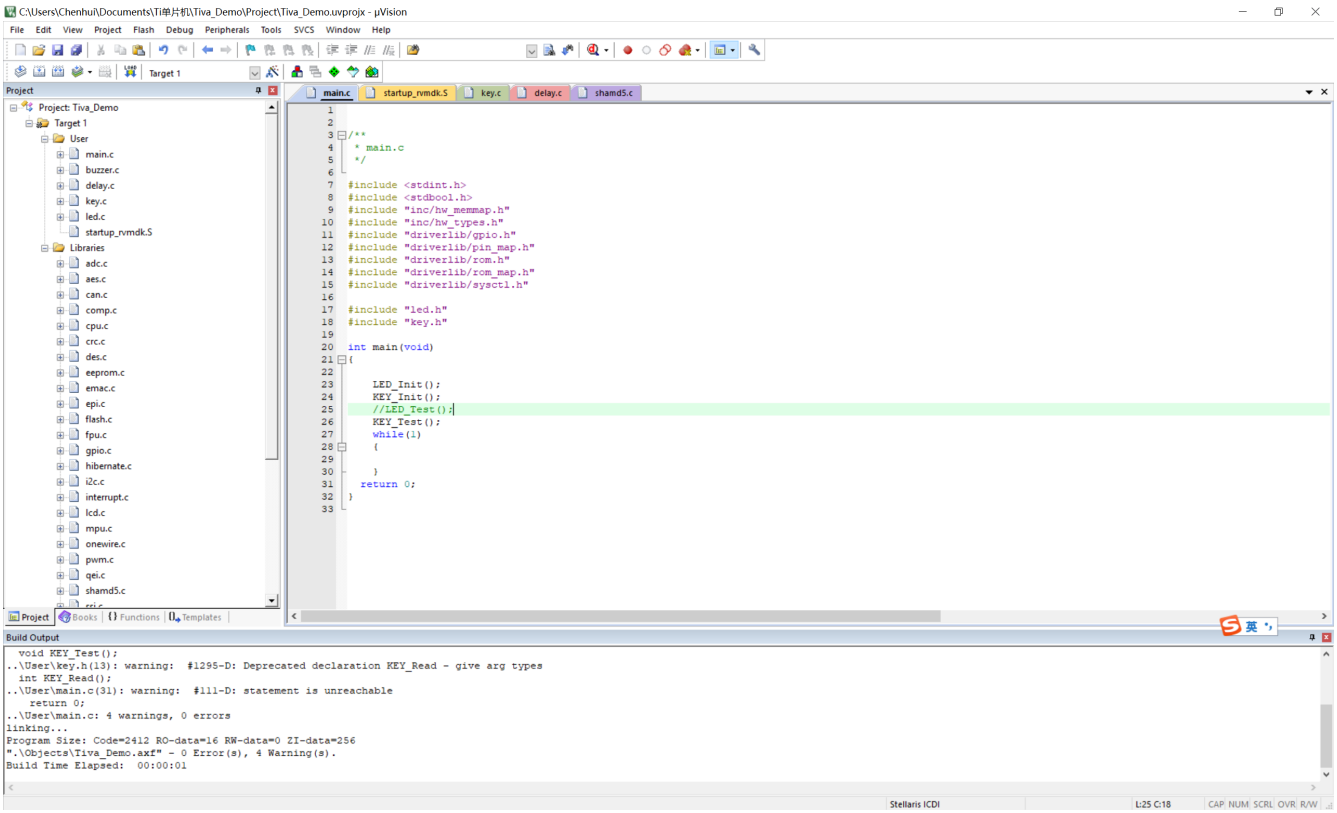
实验一

- 环境准备

  由于CCS使用不便，便使用Keil5开发环境。



- 代码

```
//LED初始化
void LED_Init(void)
{
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);
    GPIODirModeSet(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3,
GPIO_DIR_MODE_OUT);
    GPIOPadConfigSet(GPIO_PORTF_BASE, GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3 ,
 GPIO_STRENGTH_8MA_SC, GPIO_PIN_TYPE_STD);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOL);
    GPIODirModeSet(GPIO_PORTL_BASE,GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|
                   GPIO_PIN_3|GPIO_PIN_4, GPIO_DIR_MODE_OUT);
    GPIOPadConfigSet(GPIO_PORTL_BASE, GPIO_PIN_0|GPIO_PIN_1|GPIO_PIN_2|GPIO_PIN_3|
                   GPIO_PIN_4 , GPIO_STRENGTH_8MA_SC, GPIO_PIN_TYPE_STD);

}
```

```
//LED开启
void LED_Open(int i)
{
    switch(i)
    {
        case 1:
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, GPIO_PIN_1);
            break;
        case 2:
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, GPIO_PIN_2);
            break;
        case 3:
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, GPIO_PIN_3);
            break;
        case 4:
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, GPIO_PIN_0);
            break;
        case 5:
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_1, GPIO_PIN_1);
            break;
        case 6:
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_2, GPIO_PIN_2);
            break;
        case 7:
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_3, GPIO_PIN_3);
            break;
        case 8:
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_4, GPIO_PIN_4);
            break;
        default:
            break;

    }
}

//LED关闭
void LED_Close(int i)
{
    switch(i)
    {
        case 1:
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0);
            break;
        case 2:
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_2, 0);
            break;
        case 3:
            GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_3, 0);
            break;
        case 4:
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_0, 0);
            break;
        case 5:
```

```
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_1, 0);
            break;
        case 6:
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_2, 0);
            break;
        case 7:
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_3, 0);
            break;
        case 8:
            GPIOPinWrite(GPIO_PORTL_BASE, GPIO_PIN_4, 0);
            break;
        default:
            break;

    }
}

//LED测试代码
void LED_Test(void)
{
    int i=0,t=0;
    for(;;)
    {
        i++;
        if(i>8)
        {
            i=1;
        }
        LED_Open(i);
        delay();
        LED_Close(i);
        //实现流水灯效果
    }
}
```

实验二

- 代码

```
//KEY初始化
void KEY_Init()
{
    //4个行引脚配置成输入模式，用来检测电平高低
    //4个列引脚配置成输出模式，用来拉高拉低电平
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOP);
    GPIOPinTypeGPIOInput(GPIO_PORTP_BASE, GPIO_PIN_2);
    GPIOPadConfigSet(GPIO_PORTP_BASE, GPIO_PIN_2 , GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);
```

```
    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPION);
    GPIOPinTypeGPIOInput(GPIO_PORTN_BASE, GPIO_PIN_3|GPIO_PIN_2);
    GPIOPadConfigSet(GPIO_PORTN_BASE, GPIO_PIN_3|GPIO_PIN_2 , GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOD);
    GPIODirModeSet(GPIO_PORTD_BASE, GPIO_PIN_1, GPIO_DIR_MODE_OUT);
    GPIOPadConfigSet(GPIO_PORTD_BASE, GPIO_PIN_1 , GPIO_STRENGTH_8MA_SC,
GPIO_PIN_TYPE_STD);
    GPIOPinWrite(GPIO_PORTD_BASE, GPIO_PIN_1, GPIO_PIN_1);
    GPIOPinTypeGPIOInput(GPIO_PORTD_BASE, GPIO_PIN_0);
    GPIOPadConfigSet(GPIO_PORTN_BASE, GPIO_PIN_0 , GPIO_STRENGTH_2MA,
GPIO_PIN_TYPE_STD_WPU);


    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOH);
    GPIODirModeSet(GPIO_PORTH_BASE, GPIO_PIN_3|GPIO_PIN_2, GPIO_DIR_MODE_OUT);
    GPIOPadConfigSet(GPIO_PORTH_BASE, GPIO_PIN_3|GPIO_PIN_2 , GPIO_STRENGTH_8MA_SC,
GPIO_PIN_TYPE_STD);
    GPIOPinWrite(GPIO_PORTH_BASE, GPIO_PIN_3|GPIO_PIN_2, GPIO_PIN_3|GPIO_PIN_2);

    SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOM);
    GPIODirModeSet(GPIO_PORTM_BASE, GPIO_PIN_3, GPIO_DIR_MODE_OUT);
    GPIOPadConfigSet(GPIO_PORTM_BASE, GPIO_PIN_3, GPIO_STRENGTH_8MA_SC,
GPIO_PIN_TYPE_STD);
    GPIOPinWrite(GPIO_PORTM_BASE, GPIO_PIN_3, GPIO_PIN_3);

}

//KEY键值读取函数
int KEY_Read()
{
    GPIOPinWrite(GPIO_PORTD_BASE,GPIO_PIN_1, 0);
    if(0==GPIOPinRead(GPIO_PORTP_BASE, GPIO_PIN_2))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTP_BASE, GPIO_PIN_2))
        {
            return 1;
        }

    }
    if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_3))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_3))
        {
            return 5;
        }

    }
```

```
    if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_2))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_2))
        {
            return 9;
        }

    }

    if(0==GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0))
        {
            return 13;
        }

    }

    GPIOPinWrite(GPIO_PORTD_BASE,GPIO_PIN_1, GPIO_PIN_1);


    GPIOPinWrite(GPIO_PORTH_BASE,GPIO_PIN_3, 0);
    if(0==GPIOPinRead(GPIO_PORTP_BASE, GPIO_PIN_2))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTP_BASE, GPIO_PIN_2))
        {
            return 2;
        }

    }
    if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_3))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_3))
        {
            return 6;
        }

    }

    if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_2))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_2))
        {
            return 10;
        }

    }
```

```
    if(0==GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0))
        {
            return 14;
        }

    }

    GPIOPinWrite(GPIO_PORTH_BASE,GPIO_PIN_3, GPIO_PIN_3);


    GPIOPinWrite(GPIO_PORTH_BASE,GPIO_PIN_2, 0);
    if(0==GPIOPinRead(GPIO_PORTP_BASE, GPIO_PIN_2))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTP_BASE, GPIO_PIN_2))
        {
            return 3;
        }

    }
    if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_3))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_3))
        {
            return 7;
        }

    }

    if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_2))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_2))
        {
            return 11;
        }

    }

    if(0==GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0))
        {
            return 15;
        }

    }
```

```
    GPIOPinWrite(GPIO_PORTH_BASE,GPIO_PIN_2, GPIO_PIN_2);

    GPIOPinWrite(GPIO_PORTM_BASE,GPIO_PIN_3, 0);
    if(0==GPIOPinRead(GPIO_PORTP_BASE, GPIO_PIN_2))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTP_BASE, GPIO_PIN_2))
        {
            return 4;
        }

    }
    if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_3))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_3))
        {
            return 8;
        }

    }

    if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_2))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTN_BASE, GPIO_PIN_2))
        {
            return 12;
        }

    }

    if(0==GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0))
    {
        delay();
        if(0==GPIOPinRead(GPIO_PORTD_BASE, GPIO_PIN_0))
        {
            return 16;
        }

    }

    GPIOPinWrite(GPIO_PORTM_BASE,GPIO_PIN_3, GPIO_PIN_3);
    return 0;

}

//按键控制流水灯（轮询）
void KEY_Test(void)
{
    int i=0,t=0;
    int flag = 0;
    for(;;)
```

```
    {
        t=KEY_Read();
        if(t==1)
        {
            flag=1;
        }
        if(t==2)
        {
            flag=0;
        }
        if(flag == 1){
        i++;
        if(i>8)
        {
            i=1;
        }
        LED_Open(i);
        delay();
        LED_Close(i);
        }
    }

}
```