嵌入式系统与接口技术
（信工电科教学班）
实验报告

实验 1 时钟选择与 GPIO 操作

成员姓名：强陶

完成时间：2024 年 5 月 26 日

上海交通大学电子信息与电气工程学院

目　　录

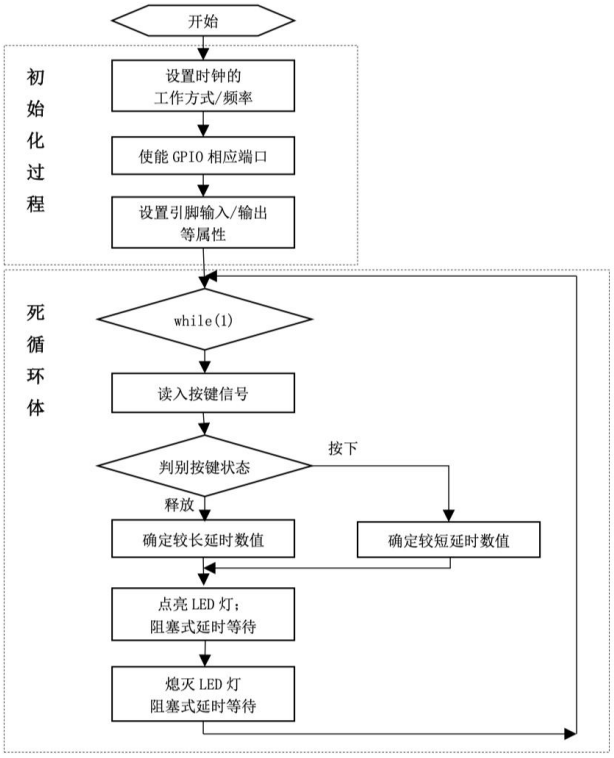
1. 实验目的

熟悉 KEILuVision5，自行建立工程项目

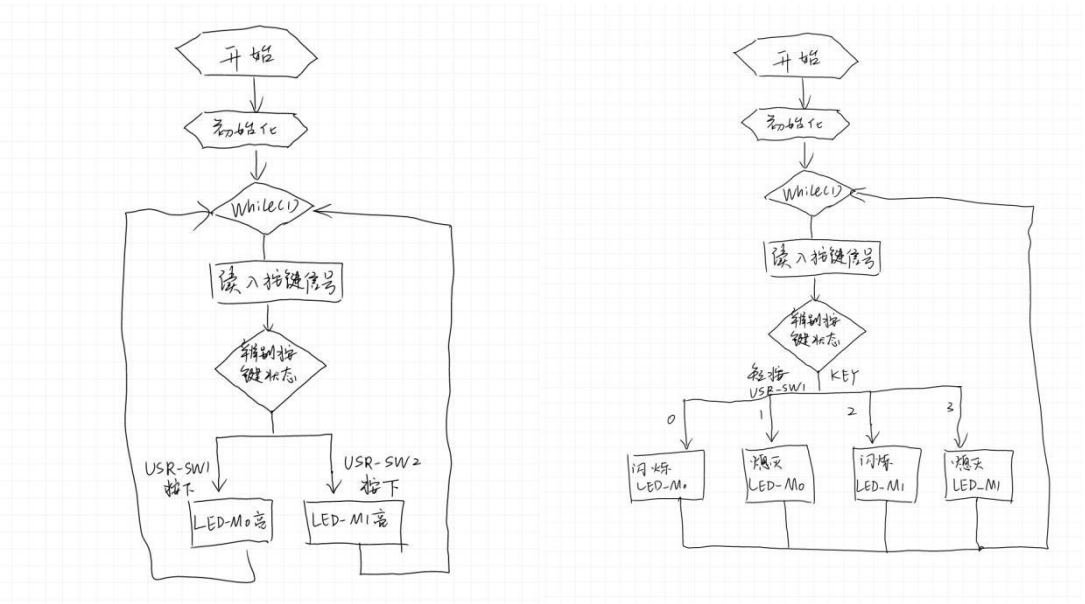
掌握 GPIO 的工作原理，能够结合 GPIO 的输入与输出功能进行实验

理解 CPU 的时钟信号，了解不同时钟对电源消耗的不同。

2. 程序流程示意图



图一、程序总体逻辑



图二、实验二与实验三的逻辑

3. 代码及实现思路分析

3.1 实验要求二

对于实验要求二：

```
read_key_value = GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)  ;
read_key_value_1 = GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1)  ;
PF0_Flash(read_key_value, read_key_value_1);

 if (key_value == 0)           //USR_SW1-PJ0 pressed
   GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);
 else                          //USR_SW1-PJ0 released
   GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0x0);
 if (key_valuel  == 0)          //USR_SW1-PJ0 pressed
   GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, GPIO_PIN_1);
 else                          //USR_SW1-PJ0 released
   GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0x0);
```

用 read_key_value 和 read_key_value_1 分别记录 USR_SW1 和 USR_SW2 是否被按下，将参数传入 PF0_Flash 以后再判定 LED_M0 与 LED_M1 是处于亮或灭的状态。

3.2 实验要求三

对于实验要求三：

```
read_key_value_1 = GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)  ;
if (read_key_value==0 && read_key_value_1==1)
{
  Key=(Key+1)%4;
}
work_1_3(Key);
read_key_value = read_key_value_1;

void work_1_3(uint32_t key_value)
{
  if(key_value==0)
  {
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);      // Turn on the LED.
    Delay(FASTFLASHTIME);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0x0);             // Turn off the LED.
    Delay(FASTFLASHTIME);
  }
  if(key_value==1)
  {
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0x0);
  }
  if(key_value==2)
  {
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, GPIO_PIN_1);      // Turn on the LED.
    Delay(FASTFLASHTIME);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0x0);             // Turn off the LED.
    Delay(FASTFLASHTIME);
  }
  if(key_value==3)
  {
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0x0);
  }
}
```

用 Key 维护此时处于第几次按下 USR_SW1 的状态（从 1 到 4 循环，分别为：闪烁 LED_M0，熄灭 LED_M0，闪烁 LED_M1，熄灭 LED_M1），切换状态的判定条件为：前一刻 USR_SW1 还未

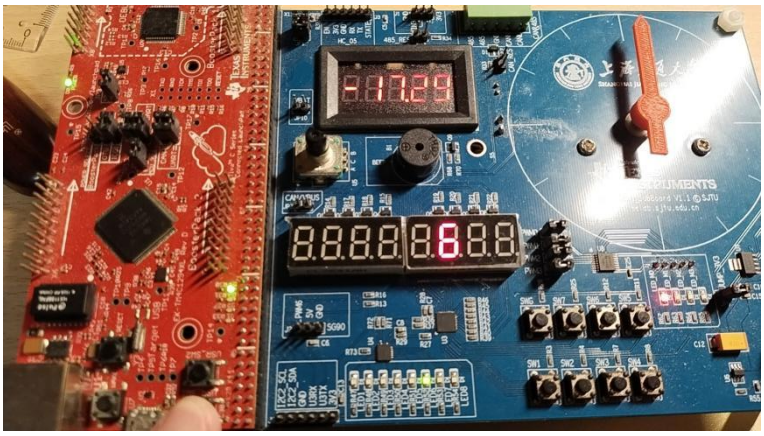
被按下，此刻 USR_SW1 已经被按下。外部系统不断将 Key 传入 work_1_3，再判断此时应该控制板子完成何种操作。

4. 实验结果

4.1 实验二效果

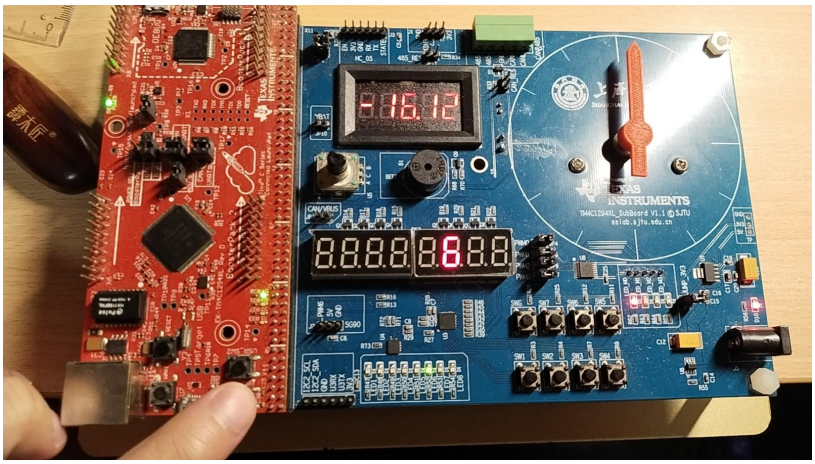
当按下 USR_SW1 键时，点亮 LED_M0，放开时，熄灭 LED_M0

当按下 USR_SW2 键时，点亮 LED_M1，放开时，熄灭 LED_M1

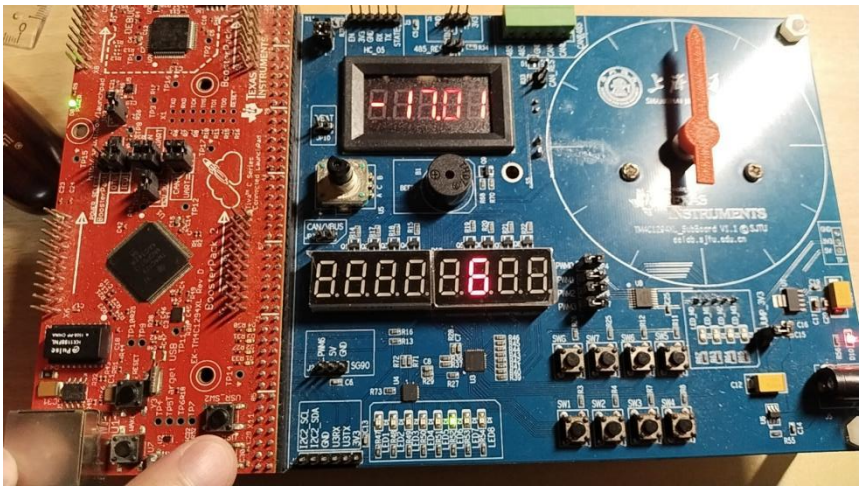


图三、按下 USR_SW1,点亮 LED_M1

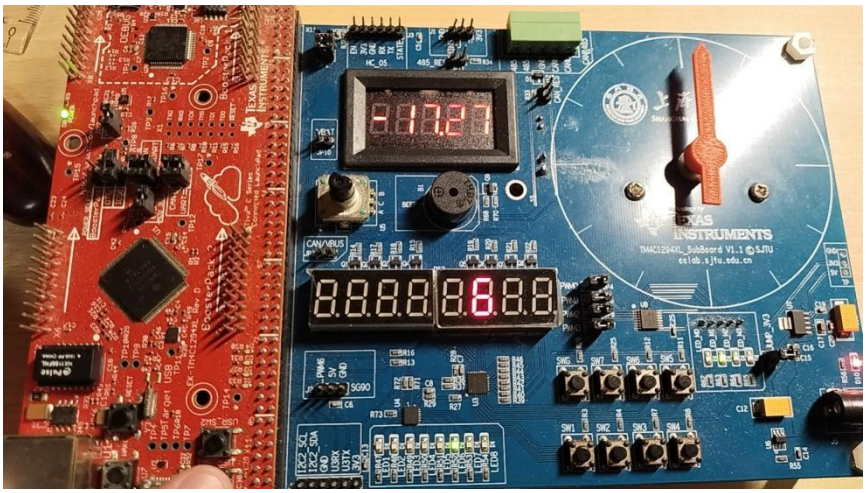
4.2 实验三效果



图四、第一次短按 USR_SW1 键时，闪烁 LED_M0



图五、第二次短按 USR_SW1 键时，熄灭 LED_M0



图六、第三次短按 USR_SW1 键时，闪烁 LED_M1

当第一次短按 USR_SW1 键时，闪烁 LED_M0，

当第二次短按 USR_SW1 键时，熄灭 LED_M0，

当第三次短按 USR_SW1 键时，闪烁 LED_M1，

当第四次短按 USR_SW1 键时，熄灭 LED_M1

5. 感想与收获

在本次实验中，笔者首次使用了 KEILuVision5，并自行完成了建立工程项目的任务。明确了使用 Keil 进行编程的基本思路：在主函数（main）中调用其他已经实现好并完成封装的功能。养成了：分析电路，明确需求，分解操作为基本操作组合；初始化（如设置 GPIO 状态等）；根据需求设置外设状态的基本操作流程习惯。

笔者在此次实验中掌握 GPIO 的工作原理，能够完成 GPIO 的端口配置与初始化，能够结合 GPIO 的输入（GPIOPinWrite）与输出（GPIOPinRead）功能进行实验操作。

6. 讨论题

1. 人为修改内部时钟或外部时钟，如将内部时钟改为 8M，或将外部时钟改为 30M，会有什么结果？

2. 能否将 PLL 时钟调整到外部时钟的频率以下？如将 25M 外部时钟用 PLL 后调整为 20M？

3. 将 PLL 后的时钟调整为最大值 120M，LED 闪烁会有什么变化？为什么？

4. GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0)此函数中，每个函数项的意义。第三个函数项为 GPIO_PIN_0，如果改为 1 或改为 2，或其他值，分别有什么现象？

5. 结合硬件说明 GPIOPinConfigure 行的作用。如果此行注释，在 WATCH 窗口中观察 key_value 值会有什么变化。

6. 为什么在慢闪时，按住 USR_SW1 时，没有马上进入快闪模式。

1.修改内部时钟或外部时钟的结果

内部时钟改为 8M:意味着 MCU 的系统时钟从默认的频率（比如 50MHz）降低到 8MHz。

(1).系统运行速度变慢：所有定时和操作都会变慢。例如，计时器的时间间隔会变长。

(2).功耗降低：低频率下运行会减少功耗。

(3).外设时钟影响：所有依赖系统时钟的外设时钟也会随之变慢。

外部时钟改为 30M:意味着使用外部时钟源提供的 30MHz 作为系统时钟或 PLL 的输入。

(1).系统运行速度改变：如果 PLL 配置得当，系统时钟可能提高或保持稳定。

(2).外设时钟影响：与外部时钟源同步的外设时钟会相应调整。

(3),稳定性和精度：外部时钟源的质量会直接影响系统的稳定性和时间精度。

2.PLL 时钟调整到外部时钟的频率以下

外部时钟 25M，通过 PLL 调整为 20M：理论上可以，但这取决于具体的 PLL 配置能力。

PLL（Phase-LockedLoop）通常用于将一个输入频率乘以某个倍数，再除以另一个倍数来获得目标频率。如果 PLL 可以支持合适的倍频和分频因子，是可以将输入的 25MHz 调整为低于 25MHz 的频率，如 20MHz。

影响：

(1).系统时钟降低：导致系统运行速度变慢。

(2).准确配置：需要正确计算 PLL 倍频和分频值，以得到精确的目标频率。

3.PLL 后的时钟调整为最大值 120M，LED 闪烁的变化和原因

LED 闪烁变化：

(1).频率增加：如果 LED 闪烁由定时器控制，定时器会更快地达到设定值，从而导致 LED 闪烁频率增加。

(2).周期变短：LED 的闪烁周期变短，闪烁速度加快。

原因：

(1)时钟频率增大：系统时钟加快，所有基于时钟的操作都会变快。

(2)定时器和延时函数：在相同的计数值下，定时器和延时函数的时间间隔变短。

4.GPIOPinWrite 函数中各参数的意义和变化

对于函数 GPIOPinWrite(GPIO_PORTF_BASE,GPIO_PIN_0,GPIO_PIN_0)

(1).GPIO_PORTF_BASE：GPIO 端口 F 的基地址。

(2).GPIO_PIN_0：GPIO 端口 F 的第 0 引脚。

(3).GPIO_PIN_0：设置第 0 引脚为高电平。

修改第 3 个参数：

改为 1：设置引脚电平为 1（相当于高电平），引脚输出高电平。

改为 2：设置引脚电平为 2，这对于 GPIO_PIN_0 是不合适的，因为 GPIO_PIN_0 只有高电平和低电平，可能导致不正确的行为或无效操作。

改为其他值：具体值取决于 GPIO 引脚支持的电平值范围和含义，通常 0 为低电平，1 为高电平，其他值可能无效。

5.GPIOPinConfigure 行的作用及其注释影响

作用：GPIOPinConfigure 用于配置 GPIO 引脚的模式，如其工作模式（如数字输入/输出、模拟功能等）。

注释掉的影响：

(1).如果 GPIOPinConfigure 被注释，可能导致引脚没有正确配置为所需的功能。

(2).key_value 可能不会反映实际的引脚状态，可能始终为默认值或未定义状态。

(3).影响调试和故障排除。

6.按住 USR_SW1 时没有马上进入快闪模式的原因

(1).去抖动延迟：按键去抖动处理可能会有延迟，避免按键抖动导致误动作。

(2).软件逻辑延迟：软件可能有一定的延迟逻辑，例如按住一定时间后才进入快闪模式。

(3).状态检测周期：如果程序周期性检测按键状态，周期较长可能导致检测滞后。

7. 源代码

```
#include <stdint.h>
#include <stdbool.h>
#include "hw_memmap.h"
#include "debug.h"
#include "gpio.h"
#include "hw_types.h"
#include "pin_map.h"
#include "sysctl.h"
#define   FASTFLASHTIME        (uint32_t) 300000
#define   SLOWFLASHTIME        (uint32_t) FASTFLASHTIME*20
void      Delay(uint32_t value);
void      S800_GPIO_Init(void);
void      PF0_Flash(uint32_t key_value, uint32_t key_value1);
void      work_1_3(uint32_t key_value);
int main(void)
{
   uint32_t read_key_value=0;
   uint32_t read_key_value_1=0;
   uint32_t Key=3;
   S800_GPIO_Init();
   while(1)
  {
      //1.2

       read_key_value = GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)    ;              //read the PJ0 key
value
      read_key_value_1 = GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_1)  ;
      PF0_Flash(read_key_value, read_key_value_1);


      //1.3
      /*
      read_key_value_1 = GPIOPinRead(GPIO_PORTJ_BASE,GPIO_PIN_0)  ;
      if (read_key_value==0 && read_key_value_1==1)
      {
         Key=(Key+1)%4;
```

```
        }
        work_1_3(Key);
        read_key_value = read_key_value_1;
        */
    }
}

void work_1_3(uint32_t key_value)
{
    if(key_value==0)
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);          // Turn on the LED.
        Delay(FASTFLASHTIME);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0x0);                 // Turn off the LED.
        Delay(FASTFLASHTIME);
    }
    if(key_value==1)
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0x0);
    }
    if(key_value==2)
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, GPIO_PIN_1);          // Turn on the LED.
        Delay(FASTFLASHTIME);
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0x0);                 // Turn off the LED.
        Delay(FASTFLASHTIME);
    }
    if(key_value==3)
    {
        GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0x0);
    }
}
void PF0_Flash(uint32_t key_value, uint32_t key_value1)
{
        /* 1.1
    uint32_t delay_time;
```

```
    if (key_value   == 0)                  //USR_SW1-PJ0 pressed
      delay_time                    = FASTFLASHTIME;
    else                                   //USR_SW1-PJ0 released
      delay_time                    = SLOWFLASHTIME;

    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);        // Turn on the LED.
    Delay(delay_time);
    GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0x0);               // Turn off the LED.
    Delay(delay_time);
  */
  //1.2
    if (key_value   == 0)                  //USR_SW1-PJ0 pressed
      GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, GPIO_PIN_0);
    else                                   //USR_SW1-PJ0 released
      GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_0, 0x0);
    if (key_value1  == 0)                  //USR_SW1-PJ0 pressed
      GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, GPIO_PIN_1);
    else                                   //USR_SW1-PJ0 released
      GPIOPinWrite(GPIO_PORTF_BASE, GPIO_PIN_1, 0x0);
}

void Delay(uint32_t value)
{
  uint32_t ui32Loop;
  for(ui32Loop = 0; ui32Loop < value; ui32Loop++){};
}

void S800_GPIO_Init(void)
{
  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOF);                  //Enable PortF

  while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOF));       //Wait for the GPIO moduleF ready

  SysCtlPeripheralEnable(SYSCTL_PERIPH_GPIOJ);                  //Enable PortJ
  while(!SysCtlPeripheralReady(SYSCTL_PERIPH_GPIOJ));       //Wait for the GPIO moduleJ ready
```

```
  GPIOPinTypeGPIOOutput(GPIO_PORTF_BASE, GPIO_PIN_0 | GPIO_PIN_1);          //Set PF0 as
Output pin

   GPIOPinTypeGPIOInput(GPIO_PORTJ_BASE,GPIO_PIN_0 | GPIO_PIN_1);//Set the PJ0,PJ1 as input
pin

                                    GPIOPadConfigSet(GPIO_PORTJ_BASE,GPIO_PIN_0          |
GPIO_PIN_1,GPIO_STRENGTH_2MA,GPIO_PIN_TYPE_STD_WPU);
}
```