**《计算系统设计与实现》**

实验报告

**姓名： 常添**

**班级： 2203102**

**学号： 2022111699**

**哈尔滨工业大学计算学部**

**2025年3月**

### 实验1 环境建立与GPIO实验

**一、实验目的（2分）**

 掌握 Tiva C LaunchPad 上 GPIO（通用输入输出）引脚的基本配置方法。

 学会使用 GPIO 控制板载 LED 的点亮和按键输入的检测。

 熟悉 TivaWare 程序库的使用，从而简化对硬件资源的操作。

**二、实验环境（2分）**

 硬件环境: TI Tiva C LaunchPad (TM4C123GH6PM)。

 软件环境: Code Composer Studio (CCS) 12.8 集成开发环境，TivaWare\_C\_Series-2.2.0.295。

 连接方式: 使用 USB 线将 LaunchPad 与计算机连接，实现供电与在电路板上进行在线调试。

**三、实验内容**

1. **实验一：控制LED灯交替闪烁**

**任务**：通过GPIO控制开发板上的三色LED灯交替闪烁。

**代码**：// 实验一代码

#include <stdint.h>

#include <stdbool.h>

#include "inc/hw\_memmap.h"

#include "driverlib/sysctl.h"

#include "driverlib/gpio.h"

int main(void)

{

**// 1. 设置时钟（5分）**

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOF))

{

// 等待 GPIOF 准备就绪

}

**// 2. 配置端口F的引脚1和引脚2为数字输出模式（5分）**

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2);

**// 3. 编写程序，使led红蓝闪烁（10分）**

while(1)

{

// 3.1 使红灯 (PF1) 亮

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2, GPIO\_PIN\_1);

SysCtlDelay(2000000);

// 3.2 使蓝灯 (PF2) 亮

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_1 | GPIO\_PIN\_2, GPIO\_PIN\_2);

SysCtlDelay(2000000);

}

}

**结果**：（2分）

* + - 将上述程序烧写到 TM4C123 LaunchPad 上运行后，可以观察到开发板上的红色 LED (PF1) 与蓝色 LED (PF2) 交替点亮，并伴有一定延时。

1. **实验二：利用按键控制LED灯**
   * **任务**：通过按键控制LED灯的亮灭。
   * **代码**：

// 实验二代码

#include <stdint.h>

#include "inc/tm4c123gh6pm.h"

int main(void){

uint8\_t ui8LEDOn = 0;

**volatile** uint32\_t ui32Loop;

**// 使能GPIO端口F的时钟（5分）**

SysCtlPeripheralEnable(SYSCTL\_PERIPH\_GPIOF);

while(!SysCtlPeripheralReady(SYSCTL\_PERIPH\_GPIOF))

{

// 等待外设就绪

}

**// 配置PF2为输出（连接到LED）（5分）**

GPIOPinTypeGPIOOutput(GPIO\_PORTF\_BASE, GPIO\_PIN\_2);

**// 配置PF4为输入（连接到按钮）（5分）**

GPIOPinTypeGPIOInput(GPIO\_PORTF\_BASE, GPIO\_PIN\_4);

GPIOPadConfigSet(GPIO\_PORTF\_BASE, GPIO\_PIN\_4,

GPIO\_STRENGTH\_2MA, GPIO\_PIN\_TYPE\_STD\_WPU);

**//编写程序，完成按键控制led的亮灭任务（10分）**

while(1)

{

// 如果检测到按键被按下 (PF4=0)

if(GPIOPinRead(GPIO\_PORTF\_BASE, GPIO\_PIN\_4) == 0)

{

// 延时约 1 秒 (假设系统时钟为 80MHz，SysCtlDelay(N) => 3\*N 个时钟周期)

SysCtlDelay(10000000 / 3);

// 再次检查是否依旧按下，若仍为低电平则视为“长按”

if(GPIOPinRead(GPIO\_PORTF\_BASE, GPIO\_PIN\_4) == 0)

{

// 翻转 LED 状态

ui8LEDOn = !ui8LEDOn;

if(ui8LEDOn)

{

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, GPIO\_PIN\_2);

}

else

{

GPIOPinWrite(GPIO\_PORTF\_BASE, GPIO\_PIN\_2, 0);

}

// 等待按键松开 (PF4 回到 1)

while(GPIOPinRead(GPIO\_PORTF\_BASE, GPIO\_PIN\_4) == 0);

}

}

}

}

**结果**：（2分）

将程序下载到开发板后：

初始状态下，PF2（LED）可能是灭的；

当长按 (按住约 1 秒) SW1 (PF4) 并松开后，LED 状态翻转；

再次长按约 1 秒后松开，LED 再次切换；

每次长按松开都会切换该 LED 的亮灭。

1. **思考题（5分）**
2. **问题**：在实验一中，如果LED灯闪烁频率过快。如何解决？给出具体措施。

回答: 可以通过增加延时的方法让闪烁频率降低。例如，增大 SysCtlDelay() 的参数使延时时间变长；或者使用定时器中断来更精确地控制闪烁周期。

1. **问题**：在实验二中，如何提升按键的去抖动效果？

回答: 可以采用软件或硬件去抖。软件去抖可以在检测到按键按下时，等待数毫秒后再次读取按键状态，以确认按键确实被按下。硬件去抖则可以添加 RC 低通滤波器，或者专用去抖电路。也可以借助定时器中断或更完善的状态机处理机制来避免抖动影响。

1. **实验总结（2分）**

通过这两个实验，我学习了如何在 Tiva C LaunchPad 上使用 GPIO 来驱动 LED 和检测按键输入，并进一步熟悉了 TivaWare 提供的简化函数（如 SysCtlPeripheralEnable()、GPIOPinWrite() 等）。实验一重点在于掌握输出引脚的配置与延时控制。实验二则加入了对输入按键的检测，以及“长按”逻辑的简单实现。

这两部分实验为后续更复杂的功能开发（如使用中断、定时器或更多外设）奠定了基础。在实际工程中，针对按键去抖动、定时精度等需求往往需要更进一步的改进或更精确的硬件/软件方案。