实验3：  
基本外设 - 端口I/O

北卡罗莱纳州立大学  
电子与计算机工程系

Alexander G. Dean博士  
2016年1月11日，版本1.1



**

*内容*

[概述 4](#_Toc460840104)

[设置 4](#_Toc460840105)

[打开和配置项目 4](#_Toc460840106)

[下载代码 4](#_Toc460840107)

[针对其他LED的程序修改 4](#_Toc460840108)

[为LED LD2、LD3和LD4配置输出端口位 4](#_Toc460840109)

[LED测试代码 5](#_Toc460840110)

[代码验证 5](#_Toc460840111)

[LED状态初始化 5](#_Toc460840112)

[针对另一个开关的程序修改 5](#_Toc460840113)

[为开关BTN2配置输入端口位 5](#_Toc460840114)

[开关测试代码 6](#_Toc460840115)

[代码验证 6](#_Toc460840116)

[针对可维护性的程序修改 6](#_Toc460840117)

[代码更改 6](#_Toc460840118)

[代码验证 7](#_Toc460840119)

[用于扫描LED的程序修改 7](#_Toc460840120)

[代码更改 7](#_Toc460840121)

[代码验证 8](#_Toc460840122)

[完成 8](#_Toc460840123)

# 概述

在本实验练习中，您将编写和测试代码以配置I/O端口，进而点亮chipKIT Wi-FIRE板上的其他三个LED和另一个开关。随后，您需要添加函数以在扫描模式下点亮LED。

讲师注意事项：在分发给学生前，应删除MySolution样式（红色斜体）的文本。

# 设置

## 打开和配置项目

1. 使用6引脚插座或电缆将PICkit 3调试器插入chipKIT板的JP1（ICSP）中。确保PICkit 3调试器上的白色三角指向JP1的引脚1（方形焊盘）。
2. 使用USB电缆将PICkit 3调试器和chipKIT Wi-Fire板连接到PC。
3. 在PC上启动MPLAB X IDE程序。
4. 在MPLAB X IDE的**“File”（文件）**菜单中，选择**“Open Project…”（打开项目…）**，然后选择名为**Lab3\_Port\_IO**的项目。
5. 单击“扫帚和锤子”图标（位于MPLAB X IDE窗口的顶部）以清除并编译主项目。   
   
6. MPLAB X IDE右下角窗格中的输出选项卡将显示这些操作的进度和结果。请注意消息“BUILD SUCCESSFUL”（编译成功），它指示总体状态。

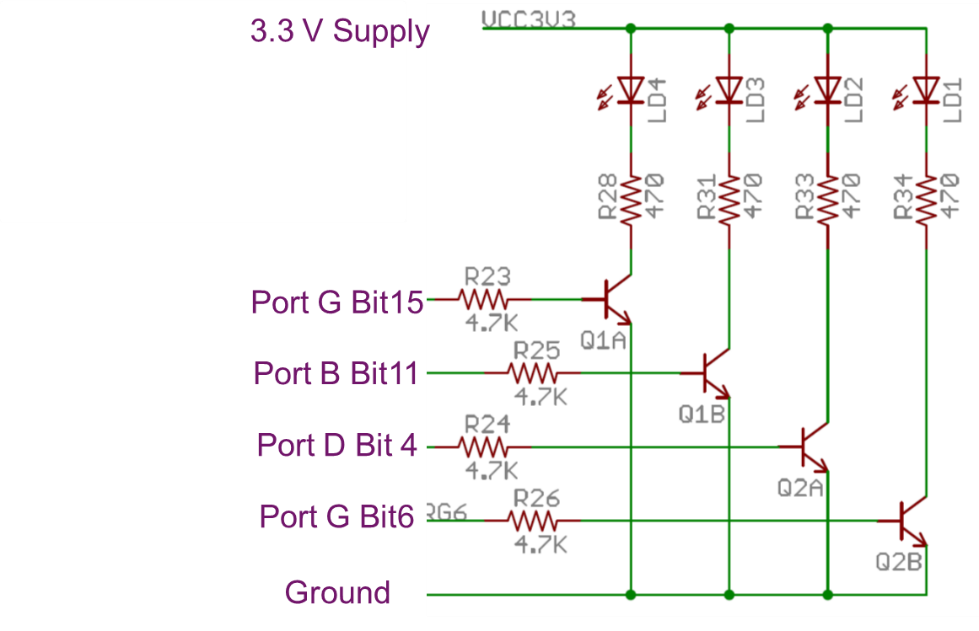
## 下载代码

1. 现在，您可以将程序下载到chipKIT中。单击绿色向下箭头图标（位于MPLAB X IDE窗口的顶部）完成此操作。   
    
2. PICkit3上的状态指示灯将在编程期间呈红色和绿色闪烁状态，然后在完成时保持绿色常亮状态。“Output”（输出）窗口将打开名为**“PICkit 3 debugger”（PICkit 3调试器）**的选项卡以显示编程状态。在此，我们可以看到器件已成功擦除和编程。
3. 完成编程后，程序将开始在chipKIT Wi-FIre板上运行。验证第一个绿色LED（LD1）是否闪烁，并验证在按下PB1时其闪烁速度是否发生变化。

# 针对其他LED的程序修改

## 为LED LD2、LD3和LD4配置输出端口位

1. 要支持其他三个LED，需要修改初始化代码。紧跟注释“Initialize outputs for other LEDs”之后向InitApp函数添加代码。根据以下原理图确定将哪个端口和位用于各个LED。



## LED测试代码

1. 紧跟注释“Turn on LEDs for testing”（将点亮全部四个LED）之后向InitApp函数添加代码。
2. 紧跟注释“Stop program here”之后向InitApp函数添加以下代码。这是一个无限循环，在断电或复位前程序将一直执行。

while (1);

## 代码验证

1. 编译、下载和运行程序。四个LED应点亮并应保持点亮状态。
2. 编辑代码以注释掉或删除用于LED测试的无限循环。

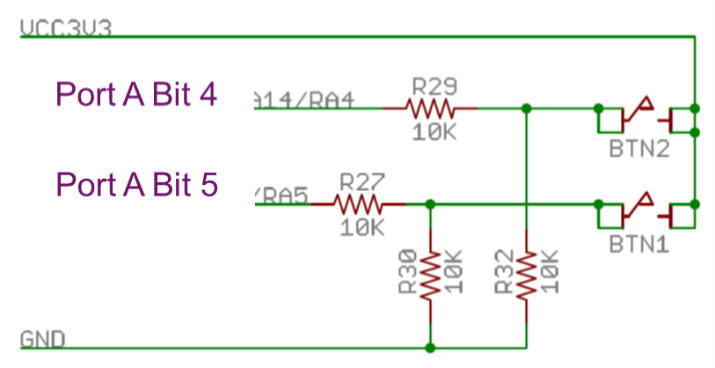
## LED状态初始化

1. 修改InitApp中的代码以在初始化时熄灭LED。

# 针对另一个开关的程序修改

## 为开关BTN2配置输入端口位

1. 要支持开关BTN2，需编写初始化代码。紧跟注释“Initialize input for BTN2”之后向InitApp函数添加代码。根据以下原理图确定将哪个端口和位用于各个开关。



## 开关测试代码

1. 紧跟注释“Test switches”之后向InitApp函数添加以下代码。这是一个无限循环，按下BTN1时将点亮LD1，按下BTN2时将点亮LD2。

while (1) {

LATGbits.LATG6 = PORTAbits.RA5;

LATDbits.LATD4 = PORTAbits.RA4;

};

## 代码验证

1. 编译、下载和运行程序。只要按下相应开关（BTN1和BTN2），LED LD1和LD2便应点亮。
2. 编辑代码以注释掉或删除用于开关测试的无限循环。

# 针对可维护性的程序修改

1. 在程序中显式引用端口名称和位编号会降低开发速度，并引入出错风险。此外，这还会增加更改这些分配的难度，因为每次在程序中使用端口位时都必须进行修改。我们将采用编译器的预处理器来为LED和开关端口连接定义更有用的名称。

## 代码更改

1. 在注释“Add I/O name macros here”之后向文件user.h（而非user.c）添加以下代码。这为端口数据位定义了友好的名称。由于LED为输出，因此我们将写入端口的锁存器以更改相应值。由于按钮为输入，因此我们将从端口寄存器中读取相应值。

#define LD1\_PORT\_BIT LATGbits.LATG6

#define LD2\_PORT\_BIT LATDbits.LATD4

#define LD3\_PORT\_BIT LATBbits.LATB11

#define LD4\_PORT\_BIT LATGbits.LATG15

#define BTN1\_PORT\_BIT PORTAbits.RA5

#define BTN2\_PORT\_BIT PORTAbits.RA4

还可为端口控制寄存器位（TRIS和ANSEL）定义类似的名称，但益处会相应降低，因为这部分只使用一次（初始化期间）。实际端口数据位在代码中的访问频率很高，因此名称定义有助于节省开发时间。此外，我们需要牢记，如果我们更改了设备的某个端口或位，则应更改#define和初始化函数InitApp中的代码。

1. 修改user.c中的代码以将这些名称用于端口数据位。

## 代码验证

1. 验证是否正确编译程序。
2. 下载并运行程序，验证其是否仍能正确运行。

# 用于扫描LED的程序修改

1. 我们添加一个名为Scan\_LEDs的函数以反复扫描LED。按下开关BTN1将更改扫描速度。按下开关BTN2后，全部四个LED应立即熄灭。

## 代码更改

1. 在user.c末尾输入以下代码。

void Scan\_LEDs(void) {

int LED\_state = 1; // 1 on (initial value), 0 off

int delay\_count=1000000;

while (1) {

if (/\* your code here \*/ == 1) { // switch 1 is pressed

delay\_count = 300000;

} else { // switch 1 is not pressed

delay\_count = 1000000;

}

if (/\* your code here \*/ == 1) { // switch 2 is pressed

// Turn off all the LEDs

LD1\_PORT\_BIT = 0;

LD2\_PORT\_BIT = 0;

LD3\_PORT\_BIT = 0;

LD4\_PORT\_BIT = 0;

} else {

LD1\_PORT\_BIT = LED\_state;

Delay(delay\_count);

LD2\_PORT\_BIT = LED\_state;

Delay(delay\_count);

LD3\_PORT\_BIT = LED\_state;

Delay(delay\_count);

LD4\_PORT\_BIT = LED\_state;

Delay(delay\_count);

// next time, set LEDs to opposite state

LED\_state = 1 - LED\_state;

}

}

}

1. 将Scan\_LEDs中的注释（/\* your code here \*/）替换为开关的名称以进行读取。
2. 在main函数（文件main.c中）中，将对Flash\_LED函数的调用替换为对Scan\_LEDs的调用。
3. 通过在标有“User Function Prototypes”（用户函数原型）部分的末尾添加此函数原型来修改user.h文件。

void Scan\_LEDs(void);

## 代码验证

1. 验证是否正确编译程序。
2. 下载和运行程序。观察程序的行为，尤其是按下开关时的变化。
3. 如果不按下任何开关，则会发生什么变化？

LED应从左（LD1）至右（LD4）依次点亮，然后按照同样的顺序熄灭。此循环应无限重复。

1. 如果仅按下BTN1，则会发生什么变化？

LED应按照之前的情况进行扫描，但速度更快。

1. 如果仅按下BTN2，则会发生什么变化？

按下此开关后，LED应立即熄灭。

1. 如果同时按下BTN1和BTN2，则会发生什么变化？

按下这两个开关后，LED应立即熄灭。

1. 如果先按下BTN1再按下BTN2，同时不释放BTN1，则会发生什么变化？

LED应快速闪烁，直至按下BTN2，此时所有LED都会熄灭。

1. 如果先按下BTN2再按下BTN1，同时不释放BTN2，则会发生什么变化？

按下这两个开关后，LED应立即熄灭。

# 完成

1. 现在，您可按下“Exit”（退出）按钮（或Shift+F5）、关闭MPLAB X IDE并断开PICkit 3调试器和chipKIT板来停止调试。   
   