实验一　GPIO 实验

1. 首先，按照工程建立流程新建一个新的工程(见相关 PPT)。
2. 了解 GPIO 的原理(见附录 1)。
3. 为方便大家实验，第一次实验提供必要的头文件与可参考使用的部分头文件与宏定义以供参考，如下：

   (1) 必要：

```
#include "lm4f232h5qd.h"
#include <stdbool.h>
#include <stdint.h>
#include <Stdlib.h>
#include <String.h>
#include "driverlib/fpu.h"
#include "inc/hw_types.h"
#include "driverlib/sysctl.h"
#include "driverlib/rom.h"
```

   (2) 选用：

```
#include "driverlib/gpio.h"
#include "inc/hw_memmap.h"
#define   SYS_CLOCK_KHZ     80000
// KEY1 地址 = 基地址(GPIO_PORTG_BASE) + 偏移量(GPIO_PIN_0)
#define   KEY1   ROM_GPIOPinRead(GPIO_PORTG_BASE,GPIO_PIN_0)
```

4. 选择按键，并进行配置。比如：**KEY1**(KEY1 由 CPU 控制，而 KEY2~3 是 PCF8574 控制的，还有一个 WAKE 键)。选定完按键之后，对照着《硬件图》进行寻找，看看是用什么 IO 口进行控制的，弄清楚之后就开始进行 IO 口的配置了。

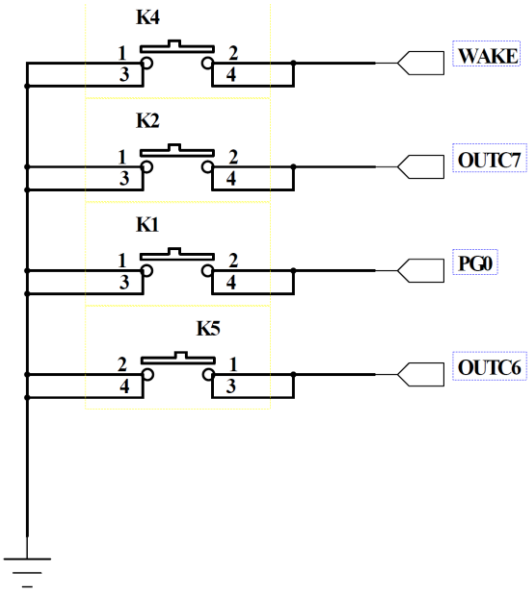

图 1　按键原理图

5. 对引脚进行寄存器的配置最主要的方法是**“软硬件结合”**，通过在《芯片手册》查询各个寄存器的作用以及各个位的含意(《芯片手册》第 10.5 节)，进行

编程，从而达到控制效果。例如：KEY1 是通过GPIO－PG0进行控制的，我们就对其配置如下：

(1) //时钟门控控制寄存器，**片选**，选择并启动 GPIOG，并提供时钟
SYSCTL_RCGCGPIO_R |= 0x40;

(2) //数据**方向**寄存器，0 为引脚方向输入，1 为引脚方向输出
//注意：通过 GPIO_PORTG_DIR_R 的写法即可访问 PG0 口的 DIR 寄存器
GPIO_PORTG_DIR_R |= 0x00;

(3) //**复用**功能选择寄存器，复位，引脚不复用,仅用作 GPIO 功能
GPIO_PORTG_AFSEL_R |= 0x00;

(4) //**数字使能**寄存器，启用 PG 管脚数字功能
GPIO_PORTG_DEN_R |= 0xFF;

(5) //**上拉**电阻选择寄存器，PG0 引脚上拉，此引脚置位时为 CPU 内部的弱上拉
GPIO_PORTG_PUR_R |= 0x01;

(6) //**数据**寄存器(这是实际干活儿的)，8 位控制 PH0~7 的数值
GPIO_PORTH_DATA_R =0x20;

6. GPIO 同样可以用于对 **LED** 灯、PCF8574 芯片(实验 3，引脚复用)等等进行控制，其配置过程跟 KEY 一样。对于本实验所要用到的 LED 灯，使用步进电机模块的 MOTO0~MOTO3 接口进行控制，具体的接线请参考《硬件图》。请根据上述配置方法自行配置 LED 灯的引脚。

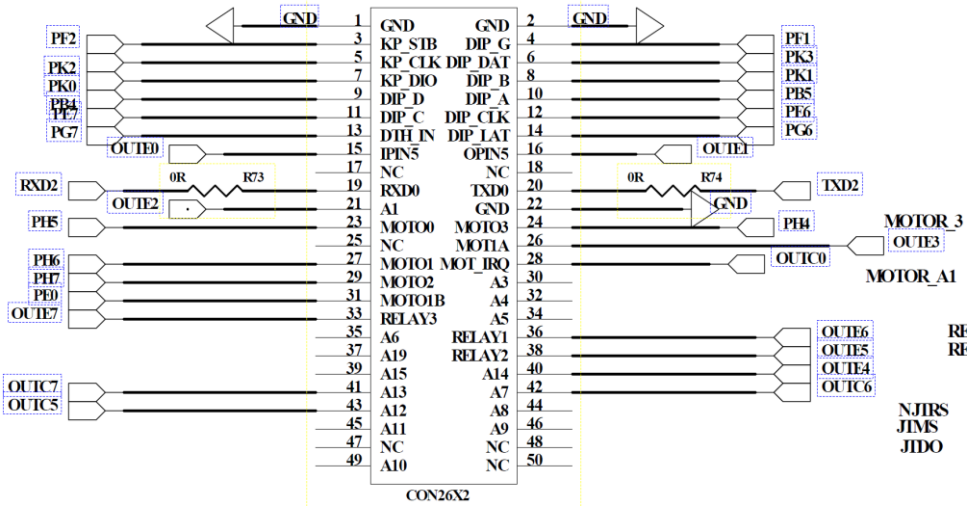

图 2　步进电机原理图

7. 完成上述步骤之后根据实验要求自行编写主程序，整体的程序主要由：**定义+配置+主函数**组成。
8. 通过 J-Link 下载程序，完成测试。

附录 1：GPIO 原理简述

GPIO 的原理分为入门知识、工作模式以及 GPIO 寄存器三部分讲述：

A. 入门知识
   a. GPIO 全称 general purpose intput output(通用输入输出端口)。GPIO 端口可通过程序配置成输入或者输出。
   b. TM4C123 的引脚中，有部分是做 GPIO 使用，部分是电源引脚/复位引脚/启动模式引脚/晶振引脚/调试下载引脚。
   c. 部分引脚除了当 GPIO 使用外，还可以**复用**为外设功能引脚(比如I^2C，实验三)。

B. 8 种工作模式(详见《STM32 八种 IO 口模式区别》)
   a. 4 种输入模式：
      输入浮空
      输入上拉
      输入下拉
      模拟输入
   b. 4 种输出模式(带上下拉)：
      开漏输出(带上拉或者下拉)
      开漏复用功能(带上拉或者下拉)
      推挽式输出(带上拉或者下拉)
      推挽式复用功能(带上拉或者下拉)

C. GPIO 寄存器(见《芯片手册》)

附录 2：上本次实验时钟配置程序

```
void sysclock_cfg()
{
 uint32_t ui32Delay, ui32RCC, ui32RCC2;
 ui32RCC = SYSCTL_RCC_R;
 ui32RCC2 = SYSCTL_RCC2_R;
 ui32RCC |= SYSCTL_RCC_BYPASS;
 ui32RCC &= ~(SYSCTL_RCC_USESYSDIV);
 ui32RCC2 |= SYSCTL_RCC2_BYPASS2;
 SYSCTL_RCC_R = ui32RCC;
 SYSCTL_RCC2_R = ui32RCC2;
 SYSCTL_RCC_R    &= ~(0x01);
 for(ui32Delay = 524288; ui32Delay > 0; ui32Delay--)
 {
 }
 ui32RCC    = SYSCTL_RCC_R;
 ui32RCC2 = SYSCTL_RCC2_R;
 ui32RCC    &= ~(0x000007c0 | 0x00000030);
 ui32RCC    |= 0x00000440;
 SYSCTL_RCC_R    = ui32RCC;
 SYSCTL_RCC2_R &= ~(0x00000070);
 SYSCTL_RCC2_R    |= (0x80000000);
```

```
    for(ui32Delay = 32768; ui32Delay > 0; ui32Delay--)
    {
    }
    SYSCTL_MISC_R |= 0x00000040;
    SYSCTL_RCC2_R &= ~(0x00002000);
    SYSCTL_RCC_R    &= ~(0x00002000);
    ui32RCC    = SYSCTL_RCC_R;
    ui32RCC &= ~(0x07800000 | 0x00400000);
    ui32RCC |= 0x07800000;
    ui32RCC2 = SYSCTL_RCC2_R;
    ui32RCC2 &= ~(0x1f800000 | 0x00400000);
    ui32RCC2 |= (0x41000000);
    for(ui32Delay = 32768; ui32Delay > 0; ui32Delay--)
    {
        if(SYSCTL_RIS_R & 0x00000040)
        {
            break;
        }
    }
    ui32RCC    &= ~(0x00000800);
    ui32RCC2 &= ~(0x00000800);
    SYSCTL_RCC_R    = ui32RCC;
    SYSCTL_RCC2_R = ui32RCC2;
    for(ui32Delay = 0; ui32Delay < 10000; )
    {
        ui32Delay++;
    }
}
```