# 实验1：GPIO外设实验

## 实验目的

* 结合数据手册，理解直接对GPIO相关寄存器进行配置的过程；
* 深入理解标准库函数设计原理，包括函数的用途和与相关寄存器之间的关系。
* 学习代码设计逻辑和书写规范，复习嵌入式C语言；

## 实验内容

* 查询开发板硬件原理图，LED0、LED1均属于共阳接法，可知与LED0、LED1相接的端口PB5、PE5输出低电平时LED灯亮，反之输出高电平时，LED灯灭。
* 参考实验例程和《STM32F1开发指南-库函数版本\_V3.0》，对使能某一IO口具体步骤分为：首先使能所挂在总线的时钟，开启时钟；依次对结构体GPIO\_InitStructure的成员变量GPIO\_Pin、GPIO\_Mode、GPIO\_Speed赋值来选择所要配置的IO口、输入输出模式和IO口翻转速度；通过GPIO\_init()函数完成GPIO端口初始化，将对GPIO\_InitStructure结构体变量的赋值进行解析并与相应的寄存器进行匹配。
* 通过 GPIO\_SetBits()和GPIO\_ResetBits()两个函数可以控制输出端口置0和置1，从而达到LED灯闪烁的效果。

## 程序框图



图3.1 主要程序流程框图

## 主要程序

此部分给出了针对IO输出端口配置过程中需要涉及到的必须代码片段。主要是对所挂载时钟的使能和对InitStructure结构体成员变量的赋值。

1. **以下对LED\_Init()函数做详细分析：**
2. **void** LED\_Init(**void**)
3. {
4. GPIO\_InitTypeDef GPIO\_InitStructure;
5. RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB|RCC\_APB2Periph\_ GPIOE, ENABLE);
6. GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_5;
7. GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_Out\_PP;
8. GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;
9. GPIO\_Init(GPIOB, &GPIO\_InitStructure);
10. GPIO\_SetBits(GPIOB,GPIO\_Pin\_5);
12. GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_5;
13. GPIO\_Init(GPIOE, &GPIO\_InitStructure);
14. GPIO\_SetBits(GPIOE,GPIO\_Pin\_5);
15. }

* **GPIO\_InitTypeDef GPIO\_InitStructure**—— 声明新变量GPIO\_InitStructure，并指向原结构体成员，其中包括GPIO\_Pin、GPIO\_Speed、GPIO\_Mode。同时GPIO\_Speed、GPIO\_Mode又分别为GPIOSpeed\_TypeDef和GPIOMode\_TypeDef声明的变量。

GPIOSpeed\_TypeDef结构体包括枚举形式的三个变量GPIO\_Speed\_10MHz、GPIO\_Speed\_2MHz和GPIO\_Speed\_50MHz，分别对应数据手册中可配置的寄存器GPIOx\_CRL和GPIOx\_CRH中的三种速度。

GPIOMode\_TypeDef结构体括枚举形式的八个变量GPIO\_Mode\_AIN、GPIO\_Mode\_IN\_FLOATING、GPIO\_Mode\_IPD、GPIO\_Mode\_IPU、GPIO\_Mode\_Out\_OD、GPIO\_Mode\_Out\_PP、GPIO\_Mode\_AF\_OD、GPIO\_Mode\_AF\_PP，分别对应数据手册中可配置的寄存器GPIOx\_CRL和GPIOx\_CRH中的八种输入输出模式。在GPIO\_Mode的定义中，输出模式下的定义的高四位均为0x1.而输入模式下设置为0/2/4， 根据此处的区别会方便后续程序处理。

* **RCC\_APB2PeriphClockCmd()函数：**

1. **void**RCC\_APB2PeriphClockCmd(uint32\_t RCC\_APB2Periph,FunctionalState NewState)
2. {
3. assert\_param(IS\_RCC\_APB2\_PERIPH(RCC\_APB2Periph));
4. assert\_param(IS\_FUNCTIONAL\_STATE(NewState));
5. **if** (NewState != DISABLE)
6. RCC->APB2ENR |= RCC\_APB2Periph;
7. **else**
8. RCC->APB2ENR &= ~RCC\_APB2Periph;
9. }

#define RCC ((RCC\_TypeDef \*) RCC\_BASE) 该代码表示RCC指向地址RCC\_BASE，RCC\_BASE存放的数据类型为RCC\_TypeDef。

由宏定义：

#define RCC\_BASE (AHBPERIPH\_BASE + 0x1000)

#define AHBPERIPH\_BASE (PERIPH\_BASE + 0x20000)

#define PERIPH\_BASE ((uint32\_t)0x40000000)

可得RCC寄存器起始地址为0x40021000。

又由于RCC\_TypeDef是结构体，且RCC\_TypeDef结构体包含\_\_IO uint32\_t CR;\_\_IO uint32\_t CFGR;\_\_IO uint32\_t CIR;\_\_IO uint32\_t APB2RSTR;\_\_IO uint32\_t APB1RSTR;\_\_IO uint32\_t AHBENR;\_\_IO uint32\_t APB2ENR;\_\_IO uint32\_t APB1ENR;\_\_IO uint32\_t BDCR;\_\_IO uint32\_t CSR;变量。所以RCC->APB2ENR所指向的地址为0x40021000+0x018=0x40021018对应于数据手册RCC\_APB2ENR寄存器地址。

#define RCC\_APB2Periph\_GPIOE ((uint32\_t)0x00000040)对应于RCC\_APB2ENR寄存器的第6位，由软件置‘1’或清‘0’开启或关闭IO端口E时钟。

* **GPIO\_Init ()函数**：在对GPIO\_InitStructure结构体变量赋值完成后，调用GPIO\_Init ()函数来对成员变量含义进行解析，然后赋值给相应的端口配置寄存器。（此部分解析穿插在代码之中）

1. **void** GPIO\_Init(GPIO\_TypeDef\* GPIOx, GPIO\_InitTypeDef\* GPIO\_InitStruct)
2. {
3. uint32\_t currentmode = 0x00, currentpin = 0x00, pinpos = 0x00, pos = 0x00;
4. uint32\_t tmpreg = 0x00, pinmask = 0x00;
5. //检测所输入的参数是否有效（即是否为所定义的结构体变量）
6. assert\_param(IS\_GPIO\_ALL\_PERIPH(GPIOx));
7. assert\_param(IS\_GPIO\_MODE(GPIO\_InitStruct->GPIO\_Mode));
8. assert\_param(IS\_GPIO\_PIN(GPIO\_InitStruct->GPIO\_Pin));
9. /\*---------------------------- GPIO Mode Configuration -----------------\*/
10. currentmode = ((uint32\_t)GPIO\_InitStruct->GPIO\_Mode) & ((uint32\_t)0x0F);
11. //只留GPIOMode\_TypeDef成员变量值的后四位赋值给参数currentmode，用于后续程序判断模式
12. **if** ((((uint32\_t)GPIO\_InitStruct->GPIO\_Mode) & ((uint32\_t)0x10)) != 0x00)
13. {   //判断所输入参数4-7位是否为1，若为1则是输出模式
14. //若为输出模式，则判断所输入的速度参数是否有效
15. assert\_param(IS\_GPIO\_SPEED(GPIO\_InitStruct->GPIO\_Speed));
16. //将速度参数赋值给currentmode参数（GPIO\_Mode结构体中所设置的值低二位均为0）
17. currentmode |= (uint32\_t)GPIO\_InitStruct->GPIO\_Speed;
18. }
19. /\*---------------------------- GPIO CRL Configuration ------------------\*/
20. **if** (((uint32\_t)GPIO\_InitStruct->GPIO\_Pin & ((uint32\_t)0x00FF)) != 0x00)
21. {  //判断GPIO\_Pin参数后8位是否不全为0（即判断是否是Pin0~Pin7引脚）
22. tmpreg = GPIOx->CRL;
23. //备份原CRL寄存器的值，以便在此次配置过程中不影响原来已经配置好的位
24. **for** (pinpos = 0x00; pinpos < 0x08; pinpos++)
25. {
26. pos = ((uint32\_t)0x01) << pinpos;
27. currentpin = (GPIO\_InitStruct->GPIO\_Pin) & pos;
28. //对GPIO\_Pin的值进行遍历比较，判断所需要配置的引脚位
29. **if** (currentpin == pos)
30. {
31. pos = pinpos << 2;
32. //将pinpos向左位移两位。即将位置乘以4，对应于CRL寄存器每4位对应一个端口
33. pinmask = ((uint32\_t)0x0F) << pos;
34. tmpreg &= ~pinmask;
35. //对即将配置的CRL寄存器的相应位置清空
36. tmpreg |= (currentmode << pos);
37. //将设置好的currentmode放置到指定的位置
38. **if** (GPIO\_InitStruct->GPIO\_Mode == GPIO\_Mode\_IPD)
39. {
40. GPIOx->BRR = (((uint32\_t)0x01) << pinpos);
41. }  //判断若为下拉模式，则设置BRR寄存器（ODR端口输出寄存器）为0，此处设置
42. 和输出端口硬件部分的NMOS和PMOS有关。
43. **else**
44. {
45. **if** (GPIO\_InitStruct->GPIO\_Mode == GPIO\_Mode\_IPU)
46. {
47. GPIOx->BSRR = (((uint32\_t)0x01) << pinpos);
48. }  //判断若为上拉模式，则设置BSRR寄存器（ODR端口输出寄存器）为1，若此
49. 处设置ODR寄存器为0，则上拉电阻会因为内部连线直接接地，不仅增加功
50. 耗，同时也会影响功能。
51. }
52. }
53. }
54. GPIOx->CRL = tmpreg;
55. }

其余部分代码为CRH（高16位）寄存器的配置过程，和CRL寄存器配置过程大体相同，此处不再赘述。

1. **GPIO\_SetBits()函数：**
2. **void** GPIO\_ResetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)
3. {
4. assert\_param(IS\_GPIO\_ALL\_PERIPH(GPIOx));
5. assert\_param(IS\_GPIO\_PIN(GPIO\_Pin));
6. GPIOx->BSRR = GPIO\_Pin;
7. }

GPIOx->BSRR与 上述举例RCC->APB2ENR用法相同，此处不再赘述，对BSRR寄存器赋值，可同时配置相应的ODR寄存位（端口输出寄存器）置0。以此来控制输出端口输出高电平。

1. **GPIO\_ResetBits()函数：**
2. **void** GPIO\_ResetBits(GPIO\_TypeDef\* GPIOx, uint16\_t GPIO\_Pin)
3. {
4. assert\_param(IS\_GPIO\_ALL\_PERIPH(GPIOx));
5. assert\_param(IS\_GPIO\_PIN(GPIO\_Pin));
6. GPIOx->BRR = GPIO\_Pin;
7. }

GPIOx->BRR与 上述举例RCC->APB2ENR用法相同，此处不再赘述，对BRR寄存器赋值，可同时配置相应的ODR寄存位（端口输出寄存器）置0。以此来控制输出端口输出低电平。

1. **直接对寄存器进行位操作：**
2. **void** LED\_Init(**void**)
3. {
4. RCC->APB2ENR|=1<<3;  //使能 PORTB 时钟
5. RCC->APB2ENR|=1<<6;  //使能 PORTE 时钟
6. GPIOB->CRL&=0XFF0FFFFF;
7. GPIOB->CRL|=0X00300000;//PB.5 推挽输出
8. GPIOB->ODR|=1<<5;  //PB.5 输出高
9. GPIOE->CRL&=0XFF0FFFFF;
10. GPIOE->CRL|=0X00300000;//PE.5 推挽输出
11. GPIOE->ODR|=1<<5;  //PE.5 输出高
12. }

此处的代码，包括对APB总线的使能和GPIOB、GPIOE端口的模式进行了配置，该部分简单的代码已经包含了上述库函数所有预表达的意思。

直接对相应的控制寄存器进行位的赋值操作，使得代码非常简介并且高效，但是带来的唯一缺点是，会增加开发者的困难，每次进行配置都需要对照数据手册详细对比，无疑是增加了工作量，相比于标准好的库函数来说，后者更加人性化，方便调用和配置。

## 实验现象

程序烧录后，可以观察到开发板上的LED0和LED1已经开始按程序设定方式闪烁。

 

