接触 STM32 快一年了，断断续续的学了点东西。学来学去总是有种心里没底的感觉。寄存器和库函数都有接触过。先用的正点原子的 STM32 不完全手册学的寄存器感觉很难就没有学下去。后来转学库函数，用库函数做了几个小项目。以为自己会了。但有机会接触到了一个大的项目的源程序，因为人家使用的是寄存器发现自己看的很痛苦。所以下定决心学一学寄存器，还有一个想法，在 STM32 上可以使用库函数，其他的也许也有库函数，但芯片的都是操作寄存器，现在学习寄存器和看寄存器手册，以后遇到别的没有库函数的芯片也不会太困难了。

刚学寄存器，写本文是一来是督促自己学习，并加深理解。二来是以一个菜鸟的理解方式学习后面会持续更新。我知道不是很全面，现在的首要目标是实现功能，优化以后在实践中积累。但是**有错误一定要指出**。共同**学习共同成长。**

本笔记是根据《STM32 不完全手册》实验顺序，参考《STM32 中文参考手册》进行的

一．点亮 LED 或灭掉 LED

想要点亮一个 LED 首先需要知道要知道有哪些寄存器需要配置

1． 配置相关寄存器

1).与 LED 连接的相关引脚的时钟配置 ——RCC->APB2ENR

2).引脚功能初始化——GPIOx->CRL 或者 GPIOx->CRH

3).使用 GPIOx->BSRS 点亮 LED 与灭掉 LED

2.相关寄存器配置

第一步 时钟初始化

从从上面可以看出有 3 个寄存器需要我们来配置。首先是 **RCC->APB2ENR 这个寄存器的功能是使能挂在 APB2 时钟线上的设备的时钟的**（我们在使用某一个功能时要先使能对应的时钟，用哪个功能就开哪个的时钟，这样可以减少功耗）

APB2ENR 的每一位对应一个功能的时钟使能。比如上面标号为 15 对应的功能是 ADC3 时钟的使能。标号 14 是串口 1 的时钟使能。

6.3.7 APB2 外设时钟使能寄存器(RCC_APB2ENR)

偏移地址：0x18

复位值：0x0000 0000

访问：字，半字和字节访问

通常无访问等待周期。但在APB2总线上的外设被访问时，将插入等待状态直到APB2的外设访问结束。

注：当外设时钟没有启用时，软件不能读出外设寄存器的数值，返回的数值始终是0x0。

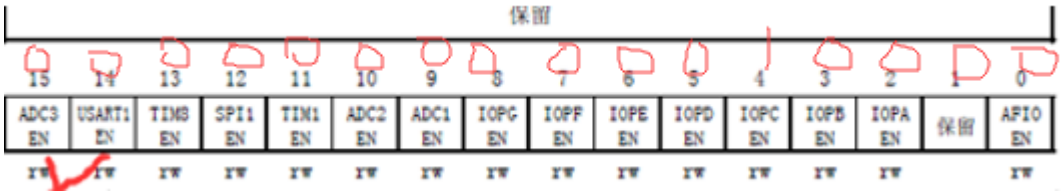
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ADC3 EN	USART1 EN	TIM8 EN	SPI1 EN	TIM1 EN	ADC2 EN	ADC1 EN	IOPG EN	IOPF EN	IOPE EN	IOPD EN	IOPC EN	IOPB EN	IOPA EN	保留	AFIO EN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw		rw

位31:16	保留，始终读为0。
位15	**ADC3EN**：ADC3接口时钟使能 (ADC 3 interface clock enable) 由软件置'1'或清'0' 0：ADC3接口时钟关闭； 1：ADC3接口时钟开启。
位14	**USART1EN**：USART1时钟使能 (USART1 clock enable) 由软件置'1'或清'0' 0：USART1时钟关闭； 1：USART1时钟开启。
位13	**TIM8EN**：TIM8定时器时钟使能 (TIM8 Timer clock enable) 由软件置'1'或清'0' 0：TIM8定时器时钟关闭； 1：TIM8定时器时钟开启。
位12	**SPI1EN**：SPI1时钟使能 (SPI 1 clock enable) 由软件置'1'或清'0' 0：SPI1时钟关闭； 1：SPI1时钟开启。
位11	**TIM1EN**：TIM1定时器时钟使能 (TIM1 Timer clock enable) 由软件置'1'或清'0'

从说明中我们可以看出写 0 关闭时钟，写 1 开启时钟。

例如：我这里 LED 对应的是 GPIOC 的 1 脚;就要把 GPIOC 的时钟使能，也就是把标号为 4 的那位写入 1 就行；



为了不影响其他位原来的值我们使用下面的语句来向标号为 4 的位置写入 1

```
RCC->APB2ENR&=0xFFFFFFEF;//把 GPIOC 对应的位清 0
RCC->APB2ENR|=0X00000010;//再把 GPIOC 对应的位置 1 使能时钟
```

第二步 引脚初始化

引脚初始化首先要知道有哪些设置模式

输入模式

（1）浮空输入——浮空输入，可以做 KEY 识别

（2）带上拉输入——IO 内部上拉电阻输入

（3）带下拉输入—— IO 内部下拉电阻输入
（4） 模拟输入——应用 ADC 模拟输入，或者低功耗下省电
输出模式
（5）开漏输出——IO 输出 0 接 GND，IO 输出 1，悬空，需要外接上拉电阻，才能实现输出高电平。当输出为 1 时，IO 口的状态由上拉电阻拉高电平，但由于是开漏输出模式，这样 IO 口也就可以由外部电路改变为低电平或不变。可以读 IO 输入电平变化，实现 C51 的 IO 双向功能
（6）推挽输出——IO 输出 0-接 GND， IO 输出 1 -接 VCC，读输入值是未知的
（7）复用功能的推挽输出——片内外设功能（I2C 的 SCL,SDA）
（8）复用功能的开漏输出——片内外设功能（TX1,MOSI,MISO.SCK.SS）
(以上内容来自 http://blog.sina.com.cn/s/blog_6c9bac050101djjz.html)
根据上面的介绍我们选择推挽输出比较方便，也可以是浮空开漏输出模式自己接上拉电阻

8.2.1 端口配置低寄存器(GPIOx_CRL) (x=A..E)

偏移地址：0x00

复位值：0x4444 4444

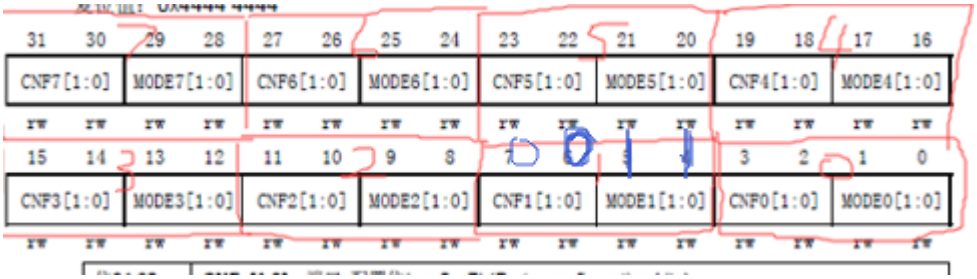
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CNF7[1:0]		MODE7[1:0]		CNF6[1:0]		MODE6[1:0]		CNF5[1:0]		MODE5[1:0]		CNF4[1:0]		MODE4[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNF3[1:0]		MODE3[1:0]		CNF2[1:0]		MODE2[1:0]		CNF1[1:0]		MODE1[1:0]		CNF0[1:0]		MODE0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

位31:30 27:26 23:22 19:18 15:14 11:10 7:6 3:2	**CNFy[1:0]**：端口x配置位(y = 0...7) (Port x configuration bits) 软件通过这些位配置相应的I/O端口，请参考表17端口位配置表。 在输入模式(MODE[1:0]=00)： 00：模拟输入模式 01：浮空输入模式(复位后的状态) 10：上拉/下拉输入模式 11：保留 在输出模式(MODE[1:0]>00)： 00：通用推挽输出模式 01：通用开漏输出模式 10：复用功能推挽输出模式 11：复用功能开漏输出模式
位29:28 25:24 21:20 17:16 13:12 9:8, 5:4 1:0	**MODEy[1:0]**：端口x的模式位(y = 0...7) (Port x mode bits) 软件通过这些位配置相应的I/O端口，请参考表17端口位配置表。 00：输入模式(复位后的状态) 01：输出模式，最大速度10MHz 10：输出模式，最大速度2MHz 11：输出模式，最大速度50MHz

这个寄存器确实看的有点晕。不仔细看再配合下面的说明就完全看不懂了。我们知道一组 **GPIO 中有 16 个脚 0~15。CRL 是配置 0~7 引脚的， CRH 是配置 8~15 脚的 CRH 和 CRL 结构是一样**。每个引脚配置分成两个部分 mode 和 CNF。从上图可以看出 MODE 主要配置输入还是输出的。CNF 是配置输出或输入的功能。比如 MODE=00 CNF=00 就是模拟输入，MODE=01,CNF=00 则是通用推挽输出最大速度是 10MHz

我的引脚是 PC1 所以要配置 CRL 中 4~7 位对应得 MODE 和 CNF;

现在我们要点亮 LED 就需要输出那么 MODE=11 输出模式,最大速度 50MHz，CNF=00 推挽输出



我们只要把 CRL 中的 1 脚对应的 4~7 位设置成 0011 就是 16 进制表示中的倒数第二位设置为 3 就可以了。

为了不影响其他位我还是使用先清除对应位在向该位写入的方法

```
GPIOC->CRL&=0XFFFFFF0F;
GPIOC->CRL|=0X00000030;
```

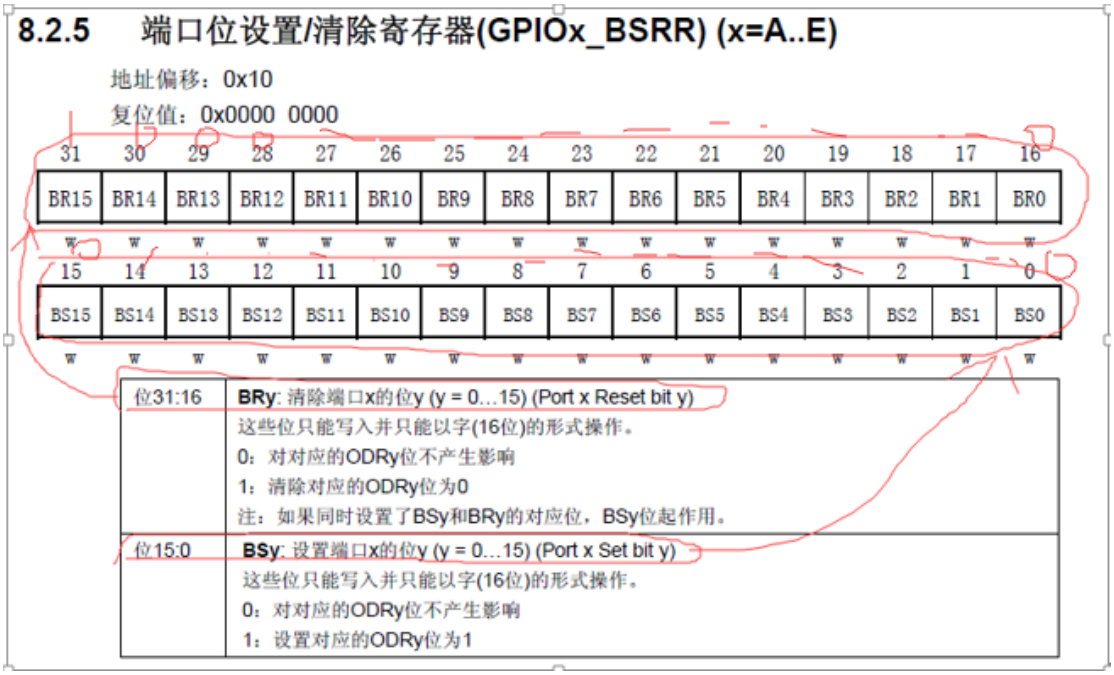
注：这里设置和 RCC 设置不同 RCC 是对每一位设置，这里是每 4 位操作从先清除写入位置就知道了

```
RCC->APB2ENR&=0xFFFFFFEF;//把 GPIOC 对应的位清 0
GPIOC->CRL&=0XFFFFFF0F;
```

前面的只清除了 1 位而后一个清除了 4 位；

第三步 控制 LED 亮灭

到这里就已经配置好了 LED 对应的引脚了。下面就是控制灯的亮灭了



从图中可以看出上面一排的 16 位写 1 就是把 0~15 引脚中的对应的引脚设为低电平。下面一排某位写 1 就是把对应的引脚设为高电平。

可以这么理解：下面的写 1 就是 1，上面的写 1 就是 0；

比如我要设置 PC1 为低，我们就在上面 17 的位置位或上 1 即可；

写成语句就是

```
GPIOC->BSRR|=0X00020000;
```

至此我们点亮 GPIOC.1 对应的 LED 灯的程序就写好了如下

参考代码 LED------PC1

```
#include "stm32f10x.h"
int main()
{
//设置引脚对应时钟
    RCC->APB2ENR&=0xFFFFFFEF;
    RCC->APB2ENR|=0X00000010;
//配置引脚模式
    GPIOC->CRL&=0xFFFFFFFF;
    GPIOC->CRL|=0x00000030;
//点亮 LED
    GPIOC->BSRR|=0x00020000;
//或灭掉 LED
//  GPIOC->BSRR|=0X00000020;
}
```