## 1.TIMER输出PWM基本概念

脉冲宽度调制(PWM)，是英文“Pulse Width Modulation”的缩写，简称脉宽调制，是利用微处理器的数字输出来对模拟电路进行控制的一种非常有效的技术。简单一点，就是对脉冲宽度的控制。一般用来控制步进电机的速度等等。

STM32的定时器除了TIM6和TIM7之外，其他的定时器都可以用来产生PWM输出，其中高级定时器TIM1和TIM8可以同时产生7路的PWM输出，而通用定时器也能同时产生4路的PWM输出。

### 1.1 PWM输出模式

STM32的PWM输出有两种模式，模式1和模式2，由TIMx\_CCMRx寄存器中的OCxM位确定的（“110”为模式1，“111”为模式2）。模式1和模式2的区别如下：

110：PWM模式1－在向上计数时，一旦TIMx\_CNT<TIMx\_CCR1时通道1为有效电平，否则为无效电平；在向下计数时，一旦TIMx\_CNT>TIMx\_CCR1时通道1为无效电平(OC1REF=0)，否则为有效电平(OC1REF=1)。

111：PWM模式2－在向上计数时，一旦TIMx\_CNT<TIMx\_CCR1时通道1为无效电平，否则为有效电平；在向下计数时，一旦TIMx\_CNT>TIMx\_CCR1时通道1为有效电平，否则为无效电平。

由此看来，模式1和模式2正好互补，互为相反，所以在运用起来差别也并不太大。

而从计数模式上来看，PWM也和TIMx在作定时器时一样，也有向上计数模式、向下计数模式和中心对齐模式，关于3种模式的具体资料，可以查看《STM32参考手册》的“14.3.9 PWM模式”一节，在此就不详细赘述了。

### 1.2PWM输出管脚

PWM的输出管脚是确定好的，具体的引脚功能可以查看《STM32参考手册》的“8.3.7 定时器复用功能重映射”一节。在此需要强调的是，不同的TIMx有分配不同的引脚，但是考虑到管脚复用功能，STM32提出了一个重映像的概念，就是说通过设置某一些相关的寄存器，来使得在其他非原始指定的管脚上也能输出PWM。但是这些重映像的管脚也是由参考手册给出的。比如说TIM3的第2个通道，在没有重映像的时候，指定的管脚是PA.7，如果设置部分重映像之后，TIM3\_CH2的输出就被映射到PB.5上了，如果设置了完全重映像的话，TIM3\_CH2的输出就被映射到PC.7上了。

### 1.3PWM输出信号

PWM输出的是一个方波信号，信号的频率是由TIMx的时钟频率和TIMx\_ARR预分频器所决定的，具体设置方法在前面一个学习笔记中有详细的交代。而输出信号的占空比则是由TIMx\_CRRx寄存器确定的。其公式为“占空比=(TIMx\_CRRx/TIMx\_ARR)\*100%”，因此，可以通过向CRR中填入适当的数来输出自己所需的频率和占空比的方波信号。

## 2.TIMER输出PWM实现步骤

1.设置RCC时钟；

2.设置GPIO时钟；

3.设置TIMx定时器的相关寄存器；

4.设置TIMx定时器的PWM相关寄存器。

第1步设置RCC时钟已经在前文中给出了详细的代码，在此就不再多说了。需要注意的是通用定时器TIMx是由APB1提供时钟，而GPIO则是由APB2提供时钟。注意，如果需要对PWM的输出进行重映像的话，还需要开启引脚复用时钟AFIO。

第2步设置GPIO时钟时，GPIO模式应该设置为复用推挽输出GPIO\_Mode\_AF\_PP，如果需要引脚重映像的话，则需要用GPIO\_PinRemapConfig()函数进行设置。

第3步设置TIMx定时器的相关寄存器时，和前一篇学习笔记一样，设置好相关的TIMx的时钟和技术模式等等。具体设置参看“TIMER基本定时功能”的学习笔记。

第4步设置PWM相关寄存器，首先要设置PWM模式（默认情况下PWM是冻结的），然后设置占空比（根据前面所述公式进行计算），再设置输出比较极性：当设置为High时，输出信号不反相，当设置为Low时，输出信号反相之后再输出。最重要是是要使能TIMx的输出状态和使能TIMx的PWM输出使能。

相关设置完成之后，就可以通过TIM\_Cmd()来打开TIMx定时器，从而得到PWM输出了。

## 3.TIMER输出PWM源代码

由于我现在手上的奋斗开发板是将PB.5接到LED上，因此需要使用TIM3的CH2通道，并且要进行引脚重映像。打开TIM3之后，PWM输出，使得LED点亮，通过改变PWM\_cfg()中的占空比可以调节LED的亮度。

## 4.程序附录

1. #include "stm32f10x\_lib.h"
2. void RCC\_cfg();
3. void GPIO\_cfg();
4. void TIMER\_cfg();
5. void PWM\_cfg();
6. //占空比，取值范围为0-100
7. int dutyfactor = 50;
8. int main()
9. {
10. int Temp;
11. RCC\_cfg();
12. GPIO\_cfg();
13. TIMER\_cfg();
14. PWM\_cfg();
15. //使能TIM3计时器，开始输出PWM
16. TIM\_Cmd(TIM3, ENABLE);
17. while(1);
18. }
19. void RCC\_cfg()
20. {
21. //定义错误状态变量
22. ErrorStatus HSEStartUpStatus;
23. //将RCC寄存器重新设置为默认值
24. RCC\_DeInit();
25. //打开外部高速时钟晶振
26. RCC\_HSEConfig(RCC\_HSE\_ON);
27. //等待外部高速时钟晶振工作
28. HSEStartUpStatus = RCC\_WaitForHSEStartUp();
29. if(HSEStartUpStatus == SUCCESS)
30. {
31. //设置AHB时钟(HCLK)为系统时钟
32. RCC\_HCLKConfig(RCC\_SYSCLK\_Div1);
33. //设置高速AHB时钟(APB2)为HCLK时钟
34. RCC\_PCLK2Config(RCC\_HCLK\_Div1);
35. //设置低速AHB时钟(APB1)为HCLK的2分频
36. RCC\_PCLK1Config(RCC\_HCLK\_Div2);
37. //设置FLASH代码延时
38. FLASH\_SetLatency(FLASH\_Latency\_2);
39. //使能预取指缓存
40. FLASH\_PrefetchBufferCmd(FLASH\_PrefetchBuffer\_Enable);
41. //设置PLL时钟，为HSE的9倍频 8MHz \* 9 = 72MHz
42. RCC\_PLLConfig(RCC\_PLLSource\_HSE\_Div1, RCC\_PLLMul\_9);
43. //使能PLL
44. RCC\_PLLCmd(ENABLE);
45. //等待PLL准备就绪
46. while(RCC\_GetFlagStatus(RCC\_FLAG\_PLLRDY) == RESET);
47. //设置PLL为系统时钟源
48. RCC\_SYSCLKConfig(RCC\_SYSCLKSource\_PLLCLK);
49. //判断PLL是否是系统时钟
50. while(RCC\_GetSYSCLKSource() != 0x08);
51. }
52. //开启TIM3的时钟
53. RCC\_APB1PeriphClockCmd(RCC\_APB1Periph\_TIM3,ENABLE);
54. //开启GPIOB的时钟和复用功能
55. RCC\_APB2PeriphClockCmd(RCC\_APB2Periph\_GPIOB | RCC\_APB2Periph\_AFIO,ENABLE);
56. }
57. void GPIO\_cfg()
58. {
59. GPIO\_InitTypeDef GPIO\_InitStructure;
60. /\*\*\*\*\*\*初始化连接LED的引脚PB5\*\*\*\*\*\*\*\*\*\*\*\*\*/
61. //部分映射，将TIM3\_CH2映射到PB5
62. //
63. GPIO\_PinRemapConfig(GPIO\_FullRemap\_TIM3, ENABLE);
64. GPIO\_PinRemapConfig(GPIO\_PartialRemap\_TIM3, ENABLE);
65. //选择引脚5
66. GPIO\_InitStructure.GPIO\_Pin = GPIO\_Pin\_5;
67. //输出频率最大50MHz
68. GPIO\_InitStructure.GPIO\_Speed = GPIO\_Speed\_50MHz;
69. //复用推挽输出
70. GPIO\_InitStructure.GPIO\_Mode = GPIO\_Mode\_AF\_PP;
71. GPIO\_Init(GPIOB,&GPIO\_InitStructure);
72. }
73. void TIMER\_cfg()
74. {
75. TIM\_TimeBaseInitTypeDef TIM\_TimeBaseStructure;
76. //重新将Timer设置为缺省值
77. TIM\_DeInit(TIM3);
78. //采用内部时钟给TIM3提供时钟源
79. TIM\_InternalClockConfig(TIM3);
80. //预分频系数为0，即不进行预分频，此时TIMER的频率为72MHz
81. TIM\_TimeBaseStructure.TIM\_Prescaler = 0;
82. //设置时钟分割
83. TIM\_TimeBaseStructure.TIM\_ClockDivision = TIM\_CKD\_DIV1;
84. //设置计数器模式为向上计数模式
85. TIM\_TimeBaseStructure.TIM\_CounterMode = TIM\_CounterMode\_Up;
86. //设置计数溢出大小，每计7200个数就产生一个更新事件，即PWM的输出频率为10kHz
87. TIM\_TimeBaseStructure.TIM\_Period = 7200 - 1;
88. //将配置应用到TIM3中
89. TIM\_TimeBaseInit(TIM3,&TIM\_TimeBaseStructure);
90. }
91. void PWM\_cfg()
92. {
93. TIM\_OCInitTypeDef TimOCInitStructure;
94. //设置缺省值
95. TIM\_OCStructInit(&TimOCInitStructure);
96. //PWM模式1输出
97. TimOCInitStructure.TIM\_OCMode = TIM\_OCMode\_PWM1;
98. //设置占空比，占空比=(CCRx/ARR)\*100%或(TIM\_Pulse/TIM\_Period)\*100%
99. TimOCInitStructure.TIM\_Pulse = dutyfactor \* 7200 / 100;
100. //TIM输出比较极性高
101. TimOCInitStructure.TIM\_OCPolarity = TIM\_OCPolarity\_High;
102. //使能输出状态
103. TimOCInitStructure.TIM\_OutputState = TIM\_OutputState\_Enable;
104. //TIM3的CH2输出
105. TIM\_OC2Init(TIM3, &TimOCInitStructure);
106. //设置TIM3的PWM输出为使能
107. TIM\_CtrlPWMOutputs(TIM3,ENABLE);
108. }