

嵌入式系统及应用实验报告

学号	202018526	姓名	高树林	专业年级	人工智能 2020 级
实验地点	南楼中栋 108 室		实验时间	2023-05-19	
实验名称	实验三 定时器及 PWM 实验				
实验学时	2	实验类型	■验证型 □设计型 ■创新型 ■综合型		
实验目的	1.熟悉定时中断计时的工作及编程方法。 2. 掌握 PWM 编程方法。				
实验内容	1.验证 Timer 定时器编程方法和 Timer 定时中断的处理过程； 2.设计不同的定时控制不同的灯，形成变频的流水灯。 3.区别 Timer 定时器其它三种工作模式（PWM 输出、输入捕获和输出比较）的不同				
实验任务 及 实验过程	1.验证 （1）从华水学堂资料中，下载第 7 章源代码，将工程\04-Software\CH07\中的工程 Timer-STM32L431 导入编译器。理解定时器嵌入式编程过程，以及中断处理的过程。 请写出程序功能和运行流程： 1. 功能： 2. 初始化系统：禁止总中断、停止看门狗定时器，给局部变量和全局变量赋初值，初始化用户外设模块（蓝灯和定时器）。 3. 在控制台输出一些提示信息。 4. 进入主循环，该循环将一直执行。 5. 在主循环中，通过定时器中断实现蓝灯的闪烁：每 20 毫秒触发一次定时器中断处理程序。当发生新的一秒时，切换蓝灯的状态（点亮或关闭），并打印当前的时间（时、分、秒）。 6. 这段代码的主要功能是控制蓝灯以一定的频率进行闪烁，并输出当前的时间。通过定时器中断和状态标志（mFlag）的切换，实现了蓝灯的周期性闪烁。这个代码示例可能是用于学习和演示定时器中断的使用，并通过控制蓝灯的亮灭来验证定时器的工作正常。 7. 代码： 8. 代码的解释： 9. 声明了一些将在主函数中使用的局部变量，包括 mFlag（主循环使用的临时变量）和 mSec（记当前秒的值）。 10. 禁止总中断并停止看门狗定时器。 11. 给局部变量 mFlag 赋值为 'A'，表示初始状态为 'A'。 12. 给全局变量数组 gTime 赋初值，其中 gTime[0] 表示小时，gTime[1] 表示分钟，gTime[2] 表示秒，并将当前秒的值赋给 mSec。 13. 初始化用户外设模块，包括初始化蓝灯和定时器。 14. 启用定时器中断。 15. 开启总中断。 16. 输出一些提示信息。 17. 进入主循环，该循环将一直执行。在循环中，通过比较当前秒和上一次记录的秒（mSec），判断是否发生了新的一秒。如果发生了新的一秒，则切换灯的状态，并更新 mSec。如果灯状态标志为 'A'，则点亮蓝灯，并打印				

当前时间（时、分、秒），然后将状态标志改为 'L'。否则，如果灯状态标志不为 'A'，则关闭蓝灯，并打印当前时间，然后将状态标志改为 'A'。
运行结果截图：（请贴上你的运行截图）



（2）从华水学堂资料中，下载第 7 章源代码，将工程\04-Software\CH07\中的工程 PWM-STM32L431 导入编译器。理解定时器 PWM 输出嵌入式编程过程。请写出程序功能和运行流程：

代码：

```
1. //=====
   =====
2. //文件名称：main.c（应用工程主函数）
3. //框架提供：SD-Arm（sumcu.suda.edu.cn）
4. //版本更新：2017.08，2020.05
5. //功能描述：见本工程的<01_Doc>文件夹下 Readme.txt 文件
6. //=====
   =====
7.
8. #define GLOBLE_VAR
9. #include "includes.h" //包含总头文件
10. void Delay_ms(uint16_t u16ms);
11. //-----
   -----
12. //声明使用到的内部函数
13. //main.c 使用的内部函数声明处
14.
15. //-----
   -----
16. //主函数，一般情况下可以认为程序从此开始运行（实际上有启动过程见
   书稿）
17. int main(void)
18. {
19.     //（1）=====启动部分（开头）
   =====
```

```

20. // (1.1) 声明 main 函数使用的局部变量
21. uint8_t mFlag; //灯的状态标志
22. uint8_t Flag;
23.
24. // (1.2) 【不变】关总中断
25. DISABLE_INTERRUPTS;
26.
27. // (1.3) 给主函数使用的局部变量赋初值
28. mFlag='A'; //灯的状态标志
29. Flag=1;
30. mFlag=0;
31. // (1.4) 给全局变量赋初值
32.
33. // (1.5) 用户外设模块初始化
34. gpio_init(LIGHT_BLUE,GPIO_OUTPUT,LIGHT_OFF); //初始
    化蓝灯
35. pwm_init(PWM_USER,1500,1000,50.0,PWM_CENTER,PWM_MINUS
    ); //PWM 输出初始化
36.
37. // (1.6) 使能模块中断
38.
39. // (1.7) 【不变】开总中断
40. ENABLE_INTERRUPTS;
41.
42. printf("-----\n");
43. printf("金葫芦提
    示: \n");
44. printf(" (1) 蓝灯每秒闪烁一次\n");
45. printf(" (2) 串口输出 PWM 的高低电平\n");
46. printf(" (3) 可通过 PWM-测试程序-C#2019 观察波形变化
    \n");
47. printf("-----\n");
48. //for(;;) { } //在此打桩，理解蓝色发光二极管为何亮起
    来了？
49.
50. // (1) =====启动部分（结尾）
    =====
51.
52. // (2) =====主循环部分（开头）
    =====
53. for(;;) //for(;;)（开头）
54. {

```

```

55.     mFlag=gpio_get(PWM_USER);
56.     if((mFlag==1)&&(Flag==1))
57.     {
58.         printf("高电平: 1\n");
59.         Flag=0;
60.         gpio_reverse(LIGHT_BLUE);
61.     }
62.     else if((mFlag==0)&&(Flag==0))
63.     {
64.
65.         printf("低电平: 0\n");
66.         Flag=1;
67.         gpio_reverse(LIGHT_BLUE);
68.     }
69. } //for(;;)结尾
70. // (2) =====主循环部分 (结尾)
71. =====
71. }
72.
73. //=====以下为主函数调用的子函数存放处
74. =====
74. //=====
75. //函数名称: Delay_ms
76. //函数返回: 无
77. //参数说明: 无
78. //功能概要: 延时 - 毫秒级
79. //=====
80. void Delay_ms(uint16_t u16ms)
81. {
82.     uint32_t u32ctr;
83.     for(u32ctr = 0; u32ctr < 8000*u16ms; u32ctr++)
84.     {
85.         __ASM("NOP");
86.     }
87. }
88. //=====
89. /*
90. 知识要素:
91. (1) main.c 是一个模板, 该文件所有代码均不涉及具体的硬件和环境,
    通过调用构件
92. 实现对硬件的干预。

```

```
93. (2) 本文件中标有【不变】的地方为系统保留，此类代码与具体项目无
    关，不宜删除。
94. (3) 本文件中对宏 GLOBLE_VAR 进行了定义，所以在包含"includes.h"
    头文件时，会定
95. 义全局变量，在其他文件中包含"includes.h"头文件时，
96. 编译时会自动增加 extern
97. */
```

解释：

1. 然后是一个函数的声明：void Delay_ms(uint16_t u16ms);，这是一个延时函数，用于实现毫秒级的延时。
2. 主函数 main() 开始。在这之前有一些初始化和配置的步骤。
3. 声明了两个局部变量 mFlag 和 Flag，并初始化了其中的一些变量。
4. 关闭总中断。
5. 初始化一些外设模块，比如初始化蓝灯和 PWM 输出。
6. 开启总中断。
7. 输出一些提示信息。
8. 主循环部分开始，使用一个无限循环 for(;;)。
9. 在主循环中，通过检查 mFlag 的值来确定输出的是高电平还是低电平，然后通过串口输出相应的信息，并改变标志 Flag 的值以切换状态。同时，还会反转蓝灯的状态。
10. 主循环结束。
11. 接下来是延时函数 Delay_ms() 的实现，通过空指令 NOP 来进行延时。

功能：

1. 初始化阶段：

声明并初始化了两个局部变量 mFlag 和 Flag。

关闭总中断。

初始化蓝灯和 PWM 输出。

开启总中断。

输出一些提示信息。

2. 主循环阶段：

在一个无限循环中进行以下操作：

通过检查 mFlag 的值，确定输出的是高电平还是低电平。

根据电平状态，通过串口输出相应的信息。

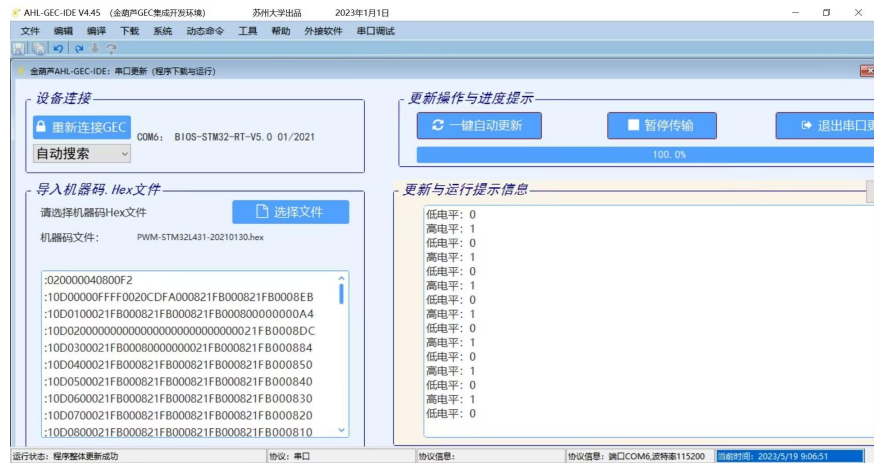
改变标志 Flag 的值以切换状态。

反转蓝灯的状态。

3. 延时函数：

提供了一个延时函数 Delay_ms()，用于实现毫秒级的延时。

总体来说，该代码的功能是控制一个蓝色 LED 灯的闪烁，并通过串口输出相应的 PWM 波形的高低电平状态。代码通过不断改变 mFlag 和 Flag 的值，以及反转蓝灯的状态，来模拟 PWM 波形的变化，并通过串口输出观察波形变化。
运行结果截图：（请贴上你的运行截图）



2. 设计

请参考已有的工程，使用一个或多个定时器，使用定时器中断让红绿蓝灯
变频闪烁，形成流水灯。

核心代码解析和运行结果截图：

代码：

```
1. #include "includes.h"           //包含总头文件
2.
3. #define LED_RED    0
4. #define LED_GREEN  1
5. #define LED_BLUE   2
6.
7. #define DELAY_MS   500
8.
9. void TIM2_IRQHandler(void);
10. void Delay_ms(uint16_t u16ms);
11.
12. int main(void)
13. {
14.     // 初始化 LED
15.     gpio_init(LED_RED, GPIO_OUTPUT, LIGHT_OFF);
16.     gpio_init(LED_GREEN, GPIO_OUTPUT, LIGHT_OFF);
17.     gpio_init(LED_BLUE, GPIO_OUTPUT, LIGHT_OFF);
18.
19.     // 初始化定时器
20.     TIM2_init(1000); // 设置定时器 2 的计数频率为 1kHz
21.
22.     // 启用定时器 2 中断
23.     NVIC_EnableIRQ(TIM2_IRQn);
24.
25.     // 启动定时器 2
```

```

26.     TIM2_start();
27.
28.     while (1) {
29.         // 主程序中无需执行其他操作
30.     }
31. }
32.
33. // 定时器中断处理函数
34. void TIM2_IRQHandler(void)
35. {
36.     static uint8_t led_index = LED_RED;
37.     static uint16_t delay_counter = DELAY_MS;
38.
39.     // 清除定时器中断标志
40.     TIM2_clear_interrupt_flag();
41.
42.     // 更新 LED 状态
43.     switch (led_index) {
44.         case LED_RED:
45.             gpio_reverse(LED_RED);
46.             break;
47.         case LED_GREEN:
48.             gpio_reverse(LED_GREEN);
49.             break;
50.         case LED_BLUE:
51.             gpio_reverse(LED_BLUE);
52.             break;
53.         default:
54.             break;
55.     }
56.
57.     // 更新延时计数器
58.     if (delay_counter > 0) {
59.         delay_counter--;
60.     } else {
61.         delay_counter = DELAY_MS;
62.         led_index = (led_index + 1) % 3; // 循环切换红绿蓝
        灯
63.     }
64. }
65.
66. // 延时函数
67. void Delay_ms(uint16_t u16ms)
68. {

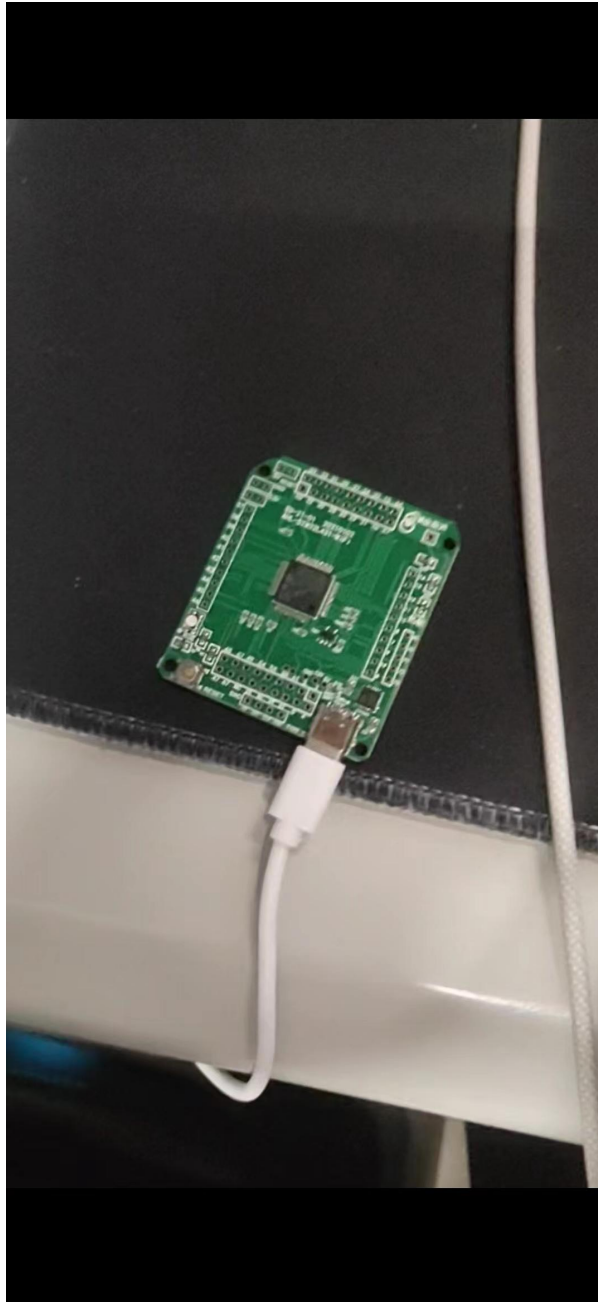
```

```
69.     uint32_t u32ctr;  
70.     for (u32ctr = 0; u32ctr < 8000 * u16ms; u32ctr++) {  
71.         __ASM("NOP");  
72.     }  
73. }
```

使用定时器 2 和定时器中断来实现红绿蓝灯的变频闪烁。具体的实现步骤如下：

1. 包含总头文件并定义红绿蓝灯的引脚和延时时间。
2. 初始化 LED 引脚为输出模式，初始状态为灭。
3. 初始化定时器 2，设置计数频率为 1kHz。
4. 启用定时器 2 中断。
5. 启动定时器 2 计数。
6. 进入主循环，在主循环中无需执行其他操作。
7. 定义定时器 2 中断处理函数 TIM2_IRQHandler:
8. 清除定时器 2 的中断标志。
9. 根据 led_index 变量切换红绿蓝灯的状态。
10. 更新 delay_counter 变量，控制切换时间。
11. 定义延时函数 Delay_ms，用于实现毫秒

结果：



3.实践性问答题

(1) 假定系统时钟频率为48MHz, Timer16 定时器的最大中断定时时间是多少？
如果想周期性定时更长时间，如何实现？

最大中断定时时间 = (最大计数值 / 时钟频率) = $(65535 / 48000000)$ 秒 ≈
1.3659 毫秒

如果想要实现更长的周期性定时，可以考虑以下几种方法：

1. 使用多个定时器：可以使用多个定时器，并将它们的中断时间累加，以实现更长的定时周期。例如，可以使用两个 16 位定时器，每个定时器的最大中断定时时间为 1.3659 毫秒，然后将它们的中断时间相加，以获得更长的周期。
2. 分频器 (Prescaler)：许多定时器具有分频器功能，可以通过将时钟频率分

频来延长定时器的周期。通过设置适当的分频器值，可以将定时器的时钟频率降低到所需的范围，从而延长定时器的最大中断定时时间。

3. 外部时钟源：有些定时器允许使用外部时钟源来驱动定时器，而不是使用系统时钟。使用更低频率的外部时钟源可以增加定时器的最大中断定时时间。

(2)给出定时器 TIM2 的 PTA5 输出 PWM 波形的基本编程步骤。

步骤 1：初始化 GPIO 引脚：

配置 PTA5 引脚为复用功能以使用定时器功能。

配置 PTA5 引脚的输出模式。

步骤 2：初始化定时器 TIM2：

使能 TIM2 的时钟。

配置 TIM2 的工作模式为 PWM 模式。

配置 PWM 的周期和占空比。

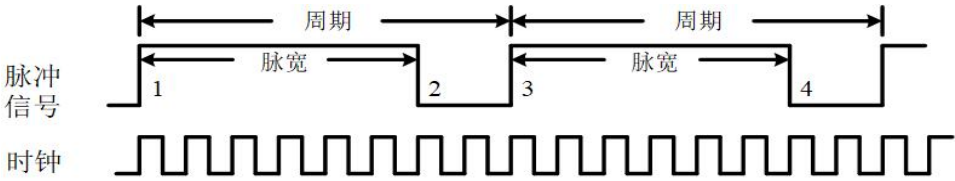
步骤 3：启动定时器 TIM2：

启动 TIM2 的计数器。

代码：

```
1. // 步骤 1：初始化 GPIO 引脚
2. // 配置 PTA5 引脚为复用功能
3. GPIO_PinConfigure(GPIOA, 5, GPIO_FUNC_ALT3);
4.
5. // 配置 PTA5 引脚的输出模式
6. GPIO_PinMode(GPIOA, 5, GPIO_MODE_OUTPUT);
7.
8. // 步骤 2：初始化定时器 TIM2
9. // 使能 TIM2 的时钟
10. SIM->SCGC |= SIM_SCGC_TIM2_MASK;
11.
12. // 配置 TIM2 的工作模式为 PWM 模式
13. TIM2->CnSC[0] = TIM_CnSC_MSB_MASK | TIM_CnSC_ELSB_MASK;
14.
15. // 配置 PWM 的周期和占空比
16. TIM2->MOD = 1000; // 设置周期为 1000 个计数单位
17. TIM2->CONTROLS[0].CnV = 500; // 设置占空比为 50%
18.
19. // 步骤 3：启动定时器 TIM2
20. TIM2->SC |= TIM_SC_CLKEN_MASK; // 启动 TIM2 的计数器
```

(3)请写出定时器输入捕获功能实现图3-1中脉冲信号的周期和脉宽的测量原理。（提示：被测周期或脉宽可能小于，也可能大于定时器的溢出周期）

	<div data-bbox="400 203 1358 383"></div> <div data-bbox="668 405 1085 436">图 1 脉冲信号输入捕捉过程示意图</div> <p>原理说明：</p> <ol style="list-style-type: none">1. 首先，脉冲信号通过定时器的输入捕获功能接入到定时器模块。2. 当脉冲信号的起始边沿到达时，定时器开始计数，并将计数器清零。3. 当脉冲信号的结束边沿到达时，定时器停止计数，并记录当前的计数值。4. 计数器的值即为脉冲信号的高电平时间（脉宽）。5. 根据脉冲信号的起始边沿和结束边沿之间的时间间隔（计数器的值），可以计算出脉冲信号的周期。 <p>通过定时器的输入捕获功能，可以方便地测量脉冲信号的周期和脉宽，从而实现对脉冲信号的精确计时和测量。</p>
<p>实验总结 （遇到的问题、解决过程、收获等）</p>	<p>通过本次的实验，我收获了以下几点：</p> <ol style="list-style-type: none">1. 理解定时器的基本原理：在进行定时器实验之前，我花了一些时间深入学习和理解定时器的工作原理。了解定时器的计数原理、中断机制以及不同工作模式的特点对于正确配置和使用定时器非常重要。2. 熟悉定时器的寄存器配置：在实验过程中，我学习了定时器相关寄存器的功能和配置方法。了解这些寄存器的作用和使用方法，可以方便地设置定时器的工作模式、计数值、中断使能等参数。3. 实践调试：实验过程中，我通过实际编写代码并在开发板上调试，加深了对定时器的理解。反复调试代码，观察定时器输出的波形，以及调整定时器参数，验证不同的中断定时时间和占空比对 PWM 波形的影响。4. 理解 PWM 的应用：通过实验，我了解了 PWM 的应用场景和作用。PWM 技术可以用于控制电机的速度、调节 LED 的亮度、实现音频输出等。理解 PWM 的原理和应用，可以为后续的电子设计和控制提供基础。5. 掌握错误处理和调试技巧：在实验过程中，我遇到了一些问题，例如定时器配置错误、波形输出异常等。通过仔细检查代码和寄存器配置，以及利用调试工具和示波器进行观察和分析，我成功解决了这些问题。这些经验对于后续的电子设计和故障排除非常有帮助。 <p>定时器及 PWM 实验是一次很有收获的实践经验。通过实际动手操作和调试，我不仅加深了对定时器和 PWM 的理解，还掌握了一些调试技巧和解决问题的方法。这些实验心得对于我的专业学习和未来的电子设计工作都具有重要意义。</p>

备注：此表格，可以根据自己的实验情况，进行格式调整。