

深圳大学实验报告

课程名称： 智能网络与计算

实验项目名称： 实验二： STM32 定时器应用开发

学院： 计算机与软件学院

专业： 计算机科学与技术

指导教师： 车越岭

报告人： 吴嘉楷 学号： 2022150168 班级： 国际班

实验时间： 2024.10.10, 2024.10.24

实验报告提交时间： 2024 年 10 月 11 日

教务处制

实验二： STM32 定时器应用开发

实验目的与要求：

实验目的：

1. 理解定时器的工作原理。
2. 掌握定时器库函数的使用。
3. 完成定时器程序编写。
4. 掌握定时器程序调试以及寄存器查看。

实验要求：

硬件环境： PC 机 Pentium 处理器双核 2GHz 以上，内存 4GB 以上

操作系统： Windows7 64 位及以上操作系统

实验器材： xLab 未来实验平台 Plus 节点

实验配件： xLab 未来实验平台 J-Link 仿真器、12V 电源

方法、步骤：

1. STM32 硬件的准备与连接

参见“附录 A”，进行操作。

2. STM32 代码下载与调试

- 1) 打开实验代码中 06-Timer\Project 目录下的 TIMER.eww 工程。
- 2) 使用 IAR 开发环境打开电子时钟实验程序并阅读 readme 文件。
- 3) 编译代码查看程序是否有误。
- 4) 将程序通过 J-Link 调试工具下载到 Plus 节点中，IAR 进入调试页面。
- 5) 点击 IAR 的执行按钮(GO)执行程序，从 Plus 节点上查看实验现象。
- 6) 通过 Watch 窗口 查看 LED 控制标志位 led_status 参数。在下图所示处打上断点，运行程序，经过 1S 后程序执行到断点处，观察 led_status 状态。
- 7) 通过 Register 窗口查看 TIM3 的计数寄存器计数值。运行程序，执行到断点处，查看 TIM3_CNT 数值变化。

3. 实验现象记录及描述

D3、D4 灯一秒转换一次状态，且两灯的状态保持相反。

4. 改变频率重新观察实验现象

原本 5000 个时钟周期转换一次状态，现修改为 50 个时钟周期，使得 D3、D4 灯 0.01 秒转换一次状态。

实验过程及内容:

1. STM32 硬件的准备与连接

将 j-link 仿真器的 USB 端连接电脑，另一端连接 plus 节点的调试接口:



图 1 STM32 硬件的连接图

2. STM32 代码下载与调试

1) 打开实验代码中 06-Timer\Project 目录下的 TIMER.eww 工程。

名称	类型	压缩大小
settings	文件夹	
TIMER.dep	DEP 文件	4 KB
TIMER.ewd	EWD 文件	5 KB
TIMER.ewp	EWP 文件	4 KB
TIMER.ewt	EWT 文件	1 KB
TIMER.eww	EWB 文件	1 KB

图 2.1 打开 TIMER.eww 工程

2) 使用 IAR 开发环境打开电子时钟实验程序并阅读 readme 文件。

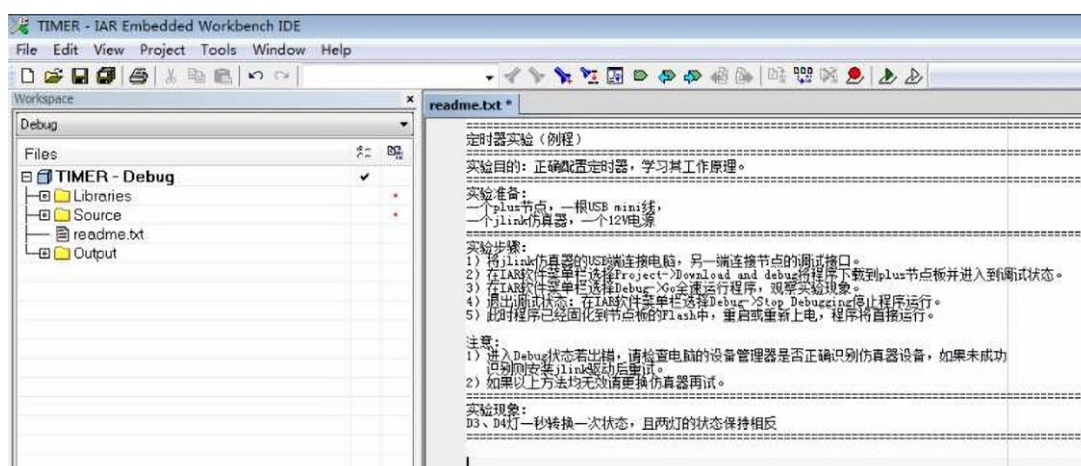



图 2.2 IAR 打开电子时钟实验程序并阅读 readme 文件

3) 编译代码查看程序是否有误。

在此之前需要使用实验资料中提供的破解工具  IARkg_Unis.exe 进行软件激活，否则编译会报错。

4) 将程序通过 J-Link 调试工具下载到 Plus 节点中，IAR 进入调试页面。

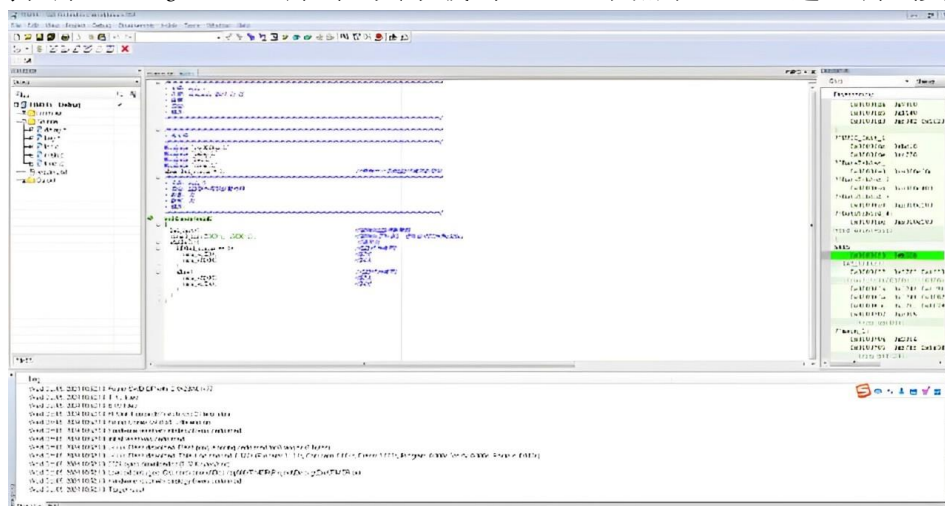


图 3 点击 download and Debug 后的结果

5) 点击 IAR 的执行按钮 (GO) 执行程序，从 Plus 节点上查看实验现象。如下图所示，D3、D4 灯每隔一秒转换一次状态，交替闪烁：

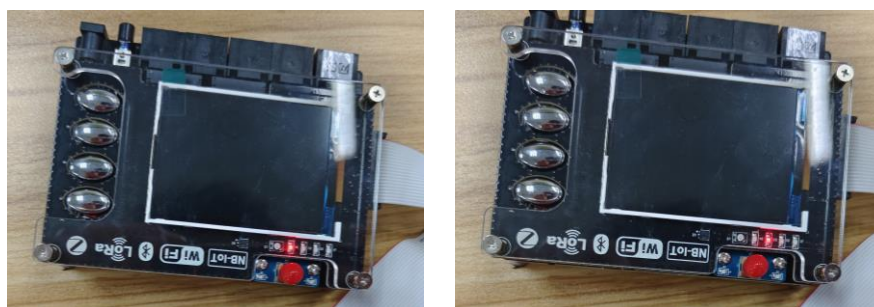


图 4、图 5 左图为 D4 亮起，右图为 D3 亮起

我们对 main.c 程序进行分析，不难发现 D3、D4 灯状态转换的原因：

```
void main(void)
{
    led_init();                                //初始化LED控制管脚
    timer3_init(5000-1, 16800-1);              //初始化定时器3，设置溢出时间为1000ms
    while(1){                                  //循环体
        if(led_status == 0){                  //LED灯为状态0
            turn_on(D3);                      //D3开
            turn_off(D4);                     //D4关
        }
        else{                                 //LED灯为状态1
            turn_off(D3);                      //D3关
            turn_on(D4);                       //D4开
        }
    }
}
```

图 5.2 D3、D4 灯转换的代码逻辑

程序会根据 led_status 变量的值来控制 D3、D4 灯的开关，并且，二者每次只有一个灯会亮起，另外一个则会关闭。

6) 通过 Watch 窗口 查看 LED 控制标志位 led_status 参数。
首先, 在下图所示处打上断点, 并在 watch 视图中新建 led_status 变量:

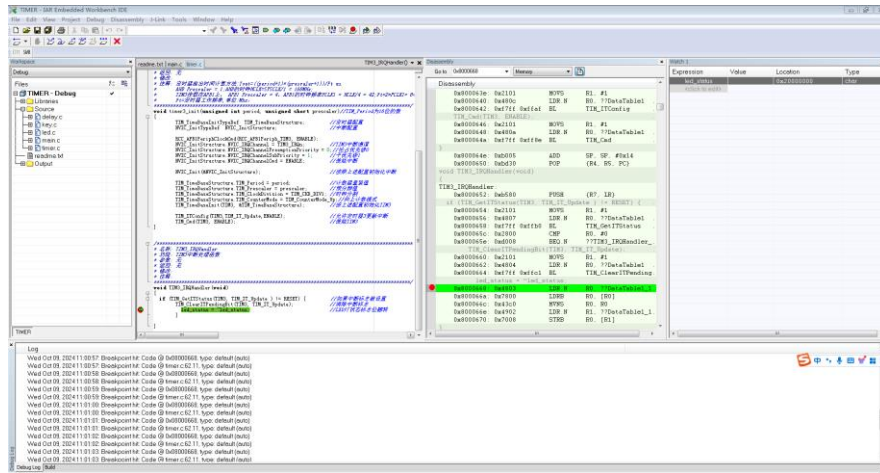


图 6.1 在绿色行处打断电

然后, 运行程序, 经过 1S 后程序执行到断点处, 观察 led_status 状态:

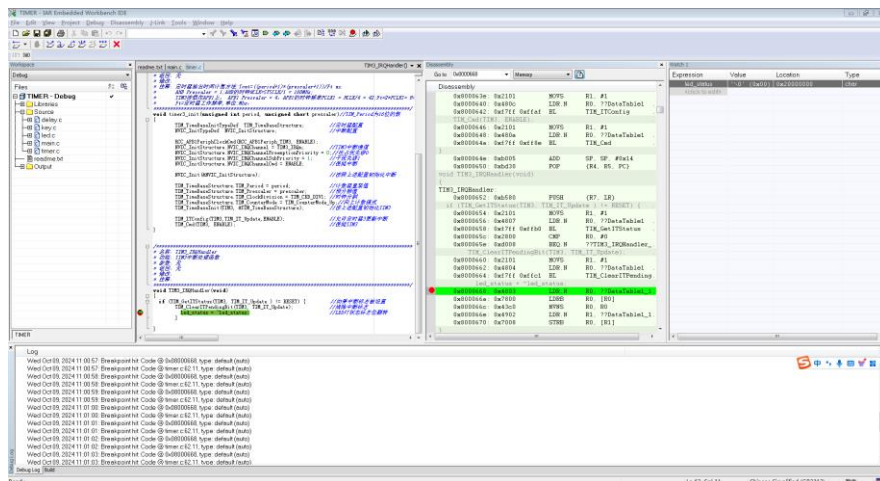


图 6.2 led_status 状态为 ‘\0’

由上图可见, led_status 的状态为 ‘\0’, 与运行程序前的空值是等效的。
对此, 我们对代码进行分析:

```
void TIM3_IRQHandler(void)
{
    if (TIM_GetITStatus(TIM3, TIM_IT_Update) != RESET) { //如果中断标志被设置
        TIM_ClearITPendingBit(TIM3, TIM_IT_Update); //清除中断标志
        led_status = ~led_status; //LED灯状态标志位翻转
    }
}
```

图 6.3 led_status 状态变化逻辑

我们在红圈所在的代码行设置了断点, 于是程序执行到当前行时暂停, 即 led_status 还没有执行翻转操作。因此, 状态未发生改变。

7) 通过 Register 窗口查看 TIM3 的计数寄存器计数值。

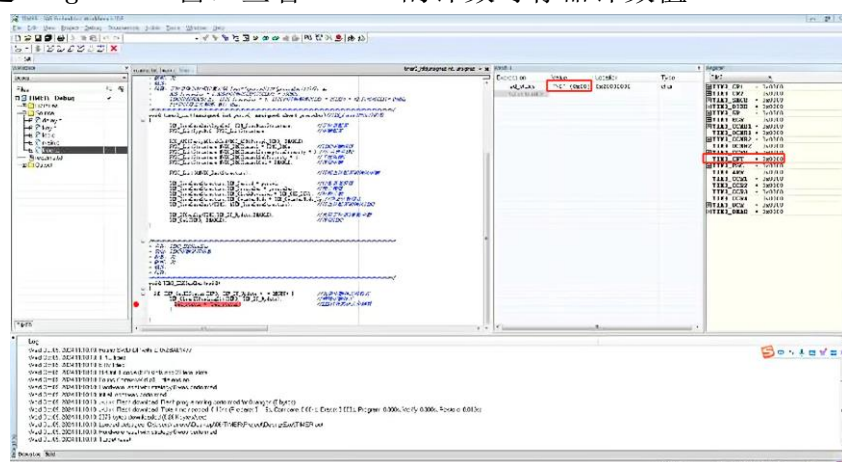


图7 查看 TIM3 的计数寄存器计数值

由图7可见，TIM3_CNT的初始值为0，led_status也为0x0000。

8) 运行程序，执行到断点处，查看 TIM3_CNT 数值变化。

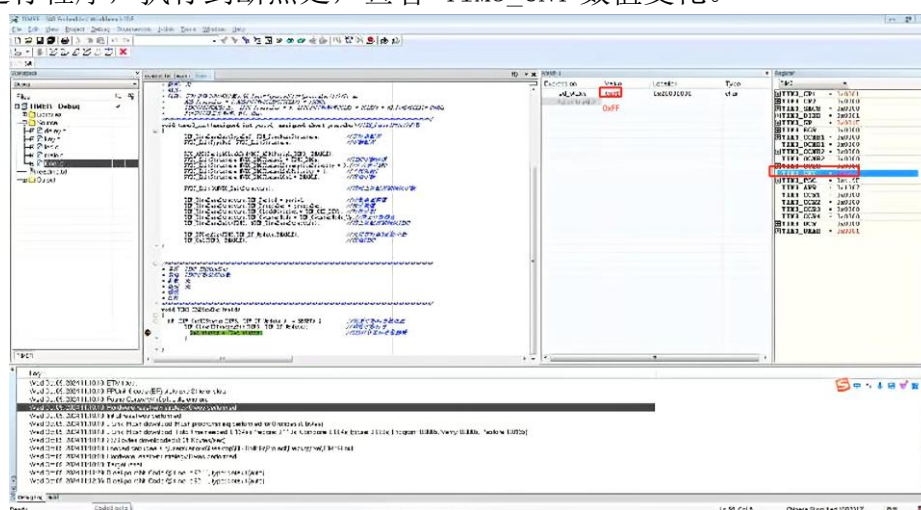


图8 查看 TIM3_CNT 数值变化

由图可见 TIM3_CNT 的数值发生了变化，这是因为每次执行当前的时钟周期数都在改变，TIM3 的计数值当然也在改变。

3. 实验现象记录及描述

D3、D4 灯一秒转换一次状态，且两灯的状态保持相反。

4. 改变频率重新观察实验现象

如何改变频率呢？

观察 main 函数可以发现，我们使用 timer3_init 函数设置了溢出时间：

```
led_init(); //初始化LED控制引脚
timer3_init(5000-1, 16800-1); //初始化定时器3，设置溢出时间为1000ms
while(1){ //循环体
    if(led_status == 0){ //LED灯为状态0
```

图9 main 函数中设置溢出时间的逻辑

进一步的，我们深入 timer3_init 函数进行探索：

```
* 参数: period: 自动重装值。 prescaler: 时钟预分频数
* 返回: 无
* 修改:
* 注释: 定时器溢出时间计算方法: Tout=((period+1)*(prescaler+1))/Ft us.
*   AHB Prescaler = 1; AHB的时钟HCLK=SYSCLK/1 = 168MHz;
*   TIM3挂载在APB1上, APB1 Prescaler = 4, APB1的时钟频率PCLK1 = HCLK/4 = 42MHz
*   Ft=定时器工作频率,单位:Mhz,
*****
void timer3_init(unsigned int period unsigned short prescaler)//TIM_Period为
{
    TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;    //定时器配置
    NVIC_InitTypeDef NVIC_InitStructure;              //中断配置

    RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM3, ENABLE);
    NVIC_InitStructure.NVIC_IRQChannel = TIM3_IRQn;    //TIM3中断通道
    NVIC_InitStructure.NVIC_IRQChannelPreemptionPriority = 0; //抢占优先级
    NVIC_InitStructure.NVIC_IRQChannelSubPriority = 1;   //子优先级1
    NVIC_InitStructure.NVIC_IRQChannelCmd = ENABLE;     //使能中断

    NVIC_Init(&NVIC_InitStructure);                  //按照上述配置初始化中断

    TIM_TimeBaseStructure.TIM_Period = period;         //计数器重装值
```

图 10 timer3_init 函数分析

由图 10 可见，timer3_init 函数的第一个参数值越大，溢出时间越大，那么 D3 和 D4 灯的转换周期就越长。为了缩短转换周期，我们可以缩小 period 参数的值！

原本 5000 个时钟周期转换一次状态，现修改为 50 个时钟周期，使得 D3、D4 灯 0.01 秒转换一次状态。

修改频率：我们将原本 5000 个时钟周期转换一次状态修改为 50 个，这将使得 D3 和 D4 灯的转换频率加快 100 倍。

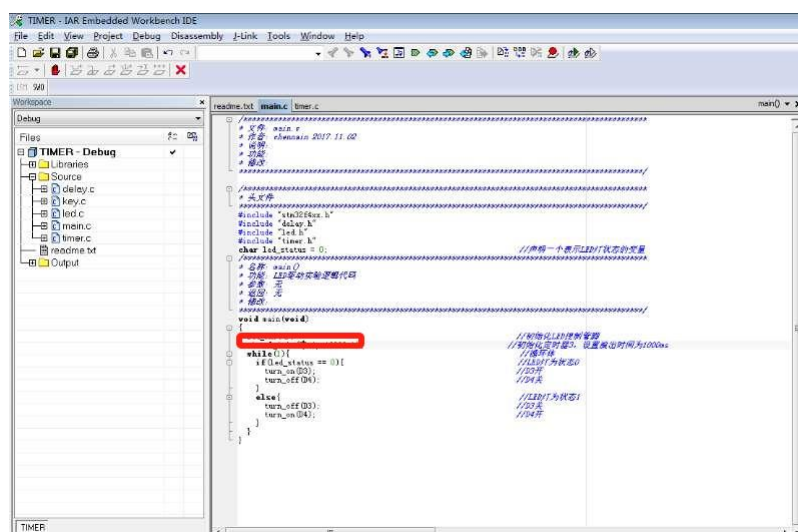


图 11 修改时钟周期

实验现象：（由于频率过快，人眼会看到 D3、D4 灯同时亮起）

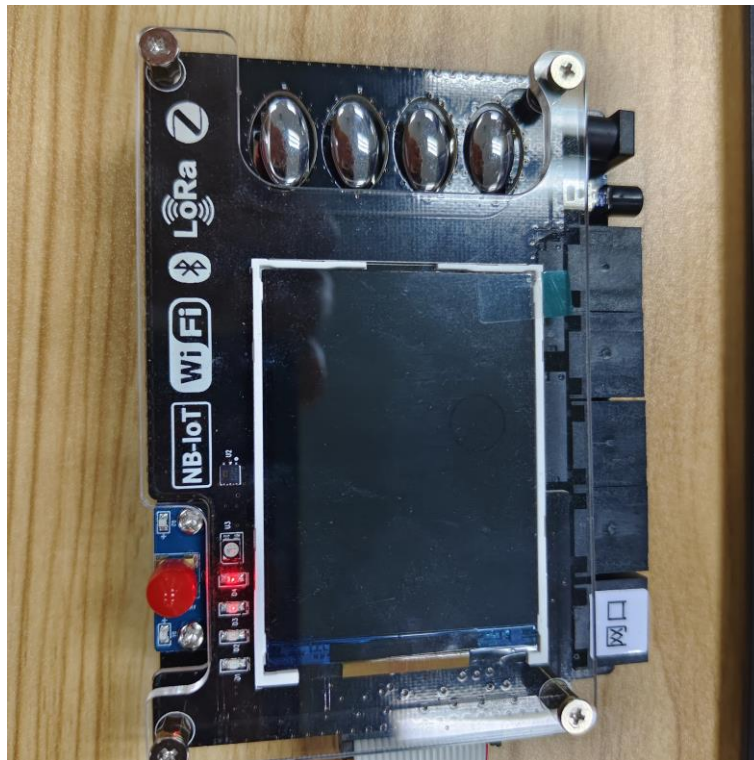


图 12 D4、D3 快速交替闪烁

注：可能由于网络问题，在网页版微信传输助手上进行图片传输的过程中发生了图片质量的损失，导致以上截屏图片有些失真，对此我尝试了多种方法进行图片的修复，但结果还是不尽人意。

实验结论：

在本次实验中，我们成功地实现了 STM32 定时器的应用开发。通过实验，我们验证了定时器的工作原理，并掌握了程序调试以及寄存器查看的方法。实验中，我们使用了 xLab 未来实验平台 Plus 节点和 J-Link 仿真器，通过 IAR 开发环境编译并下载代码到 STM32 微控制器中。

实验结果显示，D3 和 D4 两个 LED 灯能够按照预期一秒转换一次状态，且两灯的状态保持相反。通过改变定时器的时钟周期，我们观察到 LED 灯的闪烁频率也随之改变，从每秒闪烁一次变为每 0.01 秒闪烁一次。这一现象符合定时器工作原理，即通过设定不同的时钟周期来控制定时器的溢出时间，从而实现不同的定时功能。

心得体会：

通过本次实验，我对 STM32 定时器的工作原理有了更深入的理解，同时也提高了我在嵌入式系统开发方面的实践能力。在实验过程中，我学会了如何使用 IAR 开发环境进行代码的编写、编译和调试，以及如何通过 J-Link 仿真器将程序下载到硬件中进行测试。

此外，我还学习了如何通过观察寄存器的值来调试程序，这对于理解程序的运行机制和排查问题非常有帮助。

实验中也遇到了不少挑战：

1. 软件激活问题：

在编译代码时，由于没有使用实验资料中提供的破解工具进行软件激活，导致编译报错。

解决方案：我按照实验资料的指导，下载并安装了破解工具，成功激活了软件，之后编译过程就顺利完成了。

2. 硬件连接问题：

在连接 STM32 硬件时，发现 LED 灯没有按预期闪烁，初步判断可能是硬件连接不当。

解决方案：我重新检查了硬件连接，确保所有的接线都正确无误，并确认了电源供应稳定，之后 LED 灯开始按预期工作。

3. 图片传输质量问题：

在将实验过程截图传输到网页版微信传输助手时，图片出现了失真。

解决方案：我尝试了多种图片格式和压缩设置，虽然结果不尽人意，但也让我从中汲取了经验和教训。以后，可以考虑使用其他传输工具，如电子邮件或云存储服务，以避免图片质量损失。

总的来说，这次实验不仅加深了我对定时器应用的理解，也提升了我的编程和调试技能。我相信这些知识和技能将在我未来的学习和工作中发挥重要作用。

指导教师批阅意见：

成绩评定： 分

指导教师签字:

年 月 日

备注: