# 同济大学电子与信息工程学院报告


**课程名称：嵌入式系统**

**实验名称：定时器编程——简单定时器**

**任课教师：宋春林**

**姓名：斯提凡**

**学号： 1656038**

**小组成员：郭立宇、孙成龙**

# 实验四 定时器编程---简单定时器

## 一、实验目的

1. 了解 STM32 的时钟系统

2. 掌握定时器的基本使用方法

## 二、实验基本要求

1. 认真阅读和掌握本实验的程序。

2. 按实验要求编写程序并调试运行。

3. 保存与记录实验结果，并进行分析总结。

## 三、实验要点

实验环境

硬件：PC 机一台，P4 2.06CPU/40GHD/512M RAM 以上配置，STM32F103 开发板一套。

软件：PC 机操作系统为 Windows7，程序开发调试环境为 Keil C。

## 四、实验学时数

本次实验共 2 学时。

## 五、实验内容

利用定时器（Timer2）实现对小灯的定时亮灭控制。

## 六、实验步骤

1. 新建工程目录：打开 PC 机，在 D 盘新建目录“D：\Timer”
2. 拷贝固件函数库：将光盘中示例代码工程目录下的“library”目录拷贝到新建的工程目录下。
3. 新建工程：打开 Keil C，Project 菜单选择 New uVision Project，新建一个工程。将工程命名为“Timer”，并保存在 D 盘新建的目录中。
4. 选择芯片型号：在芯片型号数据库选择弹出式菜单选项中，选择通用 CPU 数据库,点击“OK”。在弹出的目标设备型号选择菜单中，选中“STMicroelectronics”公司，在出现的下拉式列表中选择“STM32F103VC”，点击“OK”。在弹出的是否添加启动文件对话框中选择“否”不添加。
5. 添加启动文件：右键点击项目资源管理器中的“Target1Source Group1”文件夹，在弹出的菜单中选择“Add File to Group ’Source Group 1’”将启动文件“stm32f10x_vector.s”，“cortexm3_macro.s”添加至项目。
6. 新建用户程序：点击主菜单“File-->New”，新建一个文件，命名为“Timer.c”保存至项目文件夹。重复第 5 步，将用户程序文件添加到工程中。
7. 添加 Include 语句：通过“Include”语句，将以上代码使用到的库件函数头文件添加到文件。
   a) #include"./library/src/stm32f10x_it.h"

b) #include"./library/src/stm32f10x_nvic.h"
c) #include"./library/src/stm32f10x_gpio.h"
d) #include"./library/src/stm32f10x_flash.h"
e) #include"./library/src/stm32f10x_rcc.h"
f) #include"./library/src/stm32f10x_exti.h"
g) #include"./library/src/stm32f10x_tim.h"
h) #include "stdio.h"
i) #include "stm32f10x_lib.h"

8. 将相应的库文件添加到工程

a) stm32f10x_it.c
b) stm32f10x_nvic.c
c) stm32f10x_gpio.c
d) stm32f10x_flash.c
e) stm32f10x_rcc.c
f) stm32f10x_exti.c
g) stm32f10x_tim.c

9. 建立 Main 函数：输入 int main(void)，建立入口函数，并添加 while(1) 循环，使 main 函数不会退出。

10. 初始化时钟：定义“ErrorStatus”类结构体“ErrorStatus HSEStartUpStatus”，并在 main 里添加时钟系统初始化函数 RCC_Configuration();

11. 配置 GPIO 端口：添加代码，定义“GPIO_InitTypeDef”类结构体“GPIO_InitStructure”，将 GPIO 口 B 的第 9 号管脚配置为推挽输出模式，速度为 50M 并初始化端口。

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOB, &GPIO_InitStructure);
```

12. 配置 Timer2 中断线优先级：添加代码，定义“NVIC_InitTypeDef”类结构“NVIC_InitStruct”，将时钟中断的抢占中断先优级与响应优先级都设置为 0，并初始化中断。

```
NVIC_InitStruct.NVIC_IRQChannel = TIM2_IRQChannel;
```

```
NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0;
NVIC_InitStruct.NVIC_IRQChannelSubPriority = 0;
NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
NVIC_Init(&NVIC_InitStruct);
```

13. 配置时钟参数：添加代码，定义“TIM_TimeBaseInitTypeDef” 类结构 “TIM_TimeBaseStructure”；计算分频器数值与计数器计数值，配置定时器定时参数，时钟向上计数，允许中断发生。

```
TIM_DeInit( TIM2);
TIM_TimeBaseStructure.TIM_Period = 35999;
TIM_TimeBaseStructure.TIM_Prescaler = 999;
TIM_TimeBaseStructure.TIM_CounterMode = TIM_CounterMode_Up;
TIM_TimeBaseInit(TIM2, & TIM_TimeBaseStructure);
TIM_ClearFlag(TIM2, TIM_FLAG_Update);
TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
TIM_Cmd(TIM2, ENABLE);
```

14. 添加中断响应函数：打开“stm32f10x_it.c”文件，找到时钟 2 中断响应函数“void TIM2_IRQHandler(void)”。在函数中添加中断响应代码：

```
if(GPIO_ReadOutputDataBit(GPIOB, GPIO_Pin_9)==0)
GPIO_SetBits(GPIOB, GPIO_Pin_9);
else
GPIO_ResetBits(GPIOB, GPIO_Pin_9);
TIM_ClearFlag(TIM2, TIM_FLAG_Update);
```

15. 连接硬件：取出开发板，用 J-Link 将开发板连接至 PC 机，并给开发板上电。

16. 配置编译环境：点击项目 Option 菜单，在弹出式菜单中选择 Debug 菜单，配置 J-Llink。选择 Utilities 菜单配置 J-Llink 并选择 Flash 类型为“STM32F10x Med-density Flash”。

17. 编译并下载：编译程序，编译完成后点击下载按钮，将程序下载至开发板。

18. 察看结果：上电察看时钟中断运行情况。

# 实验告 1 定时器编程---简单定时器

实验内容：

1. 使用定时器，让 PB8、PB9、PC12、PC13 四个小灯轮流点亮，时间间隔为 500ms。（按照步骤列出各部分代码清单）
2. 使用定时器，让 PB8、PB9、PC12、PC13 四个小灯轮流点亮，PB8、PB9 间隔为 200ms，PB9、PC12 间隔为 500ms，PC12、PC13 间隔为 1s，PC13、 PB8 间隔为 1.5s。（按照步骤列出各部分代码清单）

## 七、实验代码

### 实验基本代码

```
#include"./library/src/stm32f10x_it.h"
 #include"./library/src/stm32f10x_nvic.h"
 #include"./library/src/stm32f10x_gpio.h"
 #include"./library/src/stm32f10x_flash.h"
 #include"./library/src/stm32f10x_rcc.h"
 #include"./library/src/stm32f10x_exti.h"
 #include"./library/src/stm32f10x_tim.h"
 #include "stdio.h"
 #include "stm32f10x_lib.h"
 ErrorStatus ErrorStatusHSEStartUpStatus;
 GPIO_InitTypeDef GPIO_InitStructure;
 NVIC_InitTypeDef NVIC_InitStruct;
 TIM_TimeBaseInitTypeDef TIM_TimeBaseStructure;
```

```
 ErrorStatus HSEStartUpStatus;
  void RCC_Configuration(void)
{
RCC_DeInit();
RCC_HSEConfig(RCC_HSE_ON);
HSEStartUpStatus = RCC_WaitForHSEStartUp();
if(HSEStartUpStatus == SUCCESS)
  {
RCC_HCLKConfig(RCC_SYSCLK_Div1);
    RCC_PCLK2Config(RCC_HCLK_Div1);
    RCC_PCLK1Config(RCC_HCLK_Div2);
FLASH_SetLatency(FLASH_Latency_2);
FLASH_PrefetchBufferCmd(FLASH_PrefetchBuffer_Enable);
RCC_PLLConfig(RCC_PLLSource_HSE_Div1, RCC_PLLMul_9);
RCC_PLLCmd(ENABLE);
while(RCC_GetFlagStatus(RCC_FLAG_PLLRDY) == RESET)     ;
     RCC_SYSCLKConfig(RCC_SYSCLKSource_PLLCLK);
     while(RCC_GetSYSCLKSource() != 0x08);
  }

  /* Enable GPIOA~E and AFIO clocks */
                RCC_APB2PeriphClockCmd(RCC_APB2Periph_GPIOA|
```

```
RCC_APB2Periph_GPIOB                    |RCC_APB2Periph_GPIOC|
RCC_APB2Periph_GPIOD|RCC_APB2Periph_GPIOE|
RCC_APB2Periph_AFIO, ENABLE);

            RCC_APB2PeriphClockCmd(RCC_APB2Periph_USART1,
ENABLE);

  /* TIM1 clock enable */
  RCC_APB2PeriphClockCmd(RCC_APB2Periph_TIM1, ENABLE);

  /* TIM2 clock enable */
  RCC_APB1PeriphClockCmd(RCC_APB1Periph_TIM2, ENABLE);

  /* ADC1 clock enable */
                RCC_APB2PeriphClockCmd(RCC_APB2Periph_ADC1,
ENABLE);
}

 int main(void)
 {
   RCC_Configuration();
   GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
```

```
  GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
  GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
  GPIO_Init(GPIOB, &GPIO_InitStructure);
  NVIC_InitStruct.NVIC_IRQChannel = TIM2_IRQChannel;
  NVIC_InitStruct.NVIC_IRQChannelPreemptionPriority = 0;
  NVIC_InitStruct.NVIC_IRQChannelCmd = ENABLE;
  NVIC_Init(&NVIC_InitStruct);
  TIM_DeInit( TIM2);
  TIM_TimeBaseStructure.TIM_Period = 35999;
  TIM_TimeBaseStructure.TIM_Prescaler = 999;
                    TIM_TimeBaseStructure.TIM_CounterMode      =
TIM_CounterMode_Up;
  TIM_TimeBaseInit(TIM2, & TIM_TimeBaseStructure);
  TIM_ClearFlag(TIM2, TIM_FLAG_Update);
  TIM_ITConfig(TIM2, TIM_IT_Update, ENABLE);
  TIM_Cmd(TIM2, ENABLE);
  while(1)
  {
  }
 }
```

**运行结果：**

**B9LED 灯闪烁，时间间隔为 1s**

**进阶实验：**

**一、%%与基本实验相比，LED 灯增加为 4 个，轮流闪烁，只需要控制时钟，控制 4 个轮流的不同 GPIO 端口状态即可完成实验**

**（1）重新配置 GPIO 端口，修改代码为**

```
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_9;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOB, &GPIO_InitStructure);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_8;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOB, &GPIO_InitStructure);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_12;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOC, &GPIO_InitStructure);
GPIO_InitStructure.GPIO_Pin = GPIO_Pin_13;
GPIO_InitStructure.GPIO_Mode = GPIO_Mode_Out_PP;
GPIO_InitStructure.GPIO_Speed = GPIO_Speed_50MHz;
GPIO_Init(GPIOC, &GPIO_InitStructure);
```

**（2）修改中断代码**

```
int flag=0;
```

```
void TIM2_IRQHandler(void)
{

if(flag==1)
{
 GPIO_ResetBits(GPIOB, GPIO_Pin_8);
 GPIO_SetBits(GPIOB, GPIO_Pin_9);
 GPIO_SetBits(GPIOC, GPIO_Pin_12);
 GPIO_SetBits(GPIOC, GPIO_Pin_13);

TIM_ClearFlag(TIM2, TIM_FLAG_Update);

}
else if(flag==2)
{
      GPIO_SetBits(GPIOB, GPIO_Pin_8);
 GPIO_ResetBits(GPIOB, GPIO_Pin_9);
 GPIO_SetBits(GPIOC, GPIO_Pin_12);
 GPIO_SetBits(GPIOC, GPIO_Pin_13);
TIM_ClearFlag(TIM2, TIM_FLAG_Update);
}
else if(flag==3)
```

```
{
   GPIO_SetBits(GPIOB, GPIO_Pin_8);
 GPIO_SetBits(GPIOB, GPIO_Pin_9);
 GPIO_ResetBits(GPIOC, GPIO_Pin_12);
 GPIO_SetBits(GPIOC, GPIO_Pin_13);
TIM_ClearFlag(TIM2, TIM_FLAG_Update);
}
else
{
 GPIO_SetBits(GPIOB, GPIO_Pin_8);
 GPIO_SetBits(GPIOB, GPIO_Pin_9);
 GPIO_SetBits(GPIOC, GPIO_Pin_12);
 GPIO_ResetBits(GPIOC, GPIO_Pin_13);
TIM_ClearFlag(TIM2, TIM_FLAG_Update);
}
flag=(flag+1)%4;
}
```

**运行后实验结果与实验要求相符**

**二、%%与上实验相比较，时钟频率改变为不同的 4 个状态，取公约数为时钟频率，取 32 个不同的时钟频率显示 4 个不同 GPIO 端口状态可以完成实验**

**（1）修改时钟为 100ms，代码为**

```
   TIM_TimeBaseStructure.TIM_Period = 35999;
   TIM_TimeBaseStructure.TIM_Prescaler = 199;
```

**（2）修改中断代码为：**

```
int flag=0;
void TIM2_IRQHandler(void)
{

if(flag>=0&&flag<2)
{
 GPIO_ResetBits(GPIOB, GPIO_Pin_8);
 GPIO_SetBits(GPIOB, GPIO_Pin_9);
 GPIO_SetBits(GPIOC, GPIO_Pin_12);
 GPIO_SetBits(GPIOC, GPIO_Pin_13);

TIM_ClearFlag(TIM2, TIM_FLAG_Update);

}
else if(flag>=2&&flag<7)
{
      GPIO_SetBits(GPIOB, GPIO_Pin_8);
 GPIO_ResetBits(GPIOB, GPIO_Pin_9);
 GPIO_SetBits(GPIOC, GPIO_Pin_12);
```

```
 GPIO_SetBits(GPIOC, GPIO_Pin_13);
TIM_ClearFlag(TIM2, TIM_FLAG_Update);
}
else if(flag>=7&&flag<17)
{
   GPIO_SetBits(GPIOB, GPIO_Pin_8);
 GPIO_SetBits(GPIOB, GPIO_Pin_9);
 GPIO_ResetBits(GPIOC, GPIO_Pin_12);
 GPIO_SetBits(GPIOC, GPIO_Pin_13);
TIM_ClearFlag(TIM2, TIM_FLAG_Update);
}
else
{
 GPIO_SetBits(GPIOB, GPIO_Pin_8);
 GPIO_SetBits(GPIOB, GPIO_Pin_9);
 GPIO_SetBits(GPIOC, GPIO_Pin_12);
 GPIO_ResetBits(GPIOC, GPIO_Pin_13);
TIM_ClearFlag(TIM2, TIM_FLAG_Update);
}
flag=(flag+1)%32;
}
```

**（3）运行后实验结果与实验要求相符**