定时器/计数器

1. 实验目的

- 掌握 STC15W4K32S4 系列 MCU 定时器/计数器的原理。
- 掌握 5 个定时器/计数器外设相关寄存器配置及程序设计。

2. 实验内容

- 编写程序实现单个定时器中断的程序设计。
- 编写程序实现多个定时器中断的程序设计。

3. 硬件设计

3.1. TIMER 概念介绍

定时器（timer）几乎是每个 MCU 必有的重要外设之一，可用于定时、精确延时、计数等等，在检测、控制领域有广泛应用。

定时器运行时不占用 CPU 时间，配置好之后，可以与 CPU 并行工作，实现精确的定时和计数，并且可以通过软件控制其是否产生中断，使用起来灵活方便。通常 MCU 在介绍定时器外设时，总是和计数器（counter）一起出现，所以我们有必要先了解一下定时器和计数器的区别。

定时器和计数器实际都是通过计数器来计数，定时器是对周期不变的脉冲计数（一般来自于系统时钟），由计数的个数和脉冲的周期即可计算出时间，同时，通过一个给定的预期值（即比较值，对应预期的计数值，也就是预期时间），当计数值达到预期值时产生中断，这样就实现了定时，应用程序通过设置不同的预期值实现不同时长的定时。

计数器是对某一事件进行计数，这个事件每发生一次，计数值加/减 1，而这个事件的产生可能是没有规律的。也就是计数器的用途是对事件的发生次数进行计数，由计数值来反映事件产生的次数。

✧ 注：不同 MCU 定时器外设使用的计数器位数是个重要的参数，STC15W4K32S4 系列 MCU 的定时器使用的是 16 位计数器(由定时器高 8 位寄存器和定时器低 8 位寄存器组合起来实现)。

3.2. STC15W4K32S4 系列单片机定时器/计数器介绍

STC15W4K32S4 系列单片机有 5 个 16 位的定时器/计数器，即定时器/计数器 T0、定时器/计数器 T1、定时器/计数器 T2、定时器/计数器 T3、定时器/计数器 T4。

这些定时器/计数器外设有 2 种工作方式：定时方式（定时器）和计数方式（计数器）。可通过特殊功能寄存器（SFR）设置相应的 C/T 控制位，来选择定时器/计数器工作在哪种方式。

定时器/计数器外设的核心部件是一个加法计数器，其本质是对脉冲信号进行计数。不同的是，定时器的脉冲信号来自于系统时钟，而计数器的脉冲信号来自于单片机特定外部输入引脚。STC15W4K32S4 系列单片机计数器特定外部输入引脚如下表。

表 1：单片机计数器外部输入引脚分配

Tx	对应 IO 口	功能描述	说明	备注
T0	P3.4	计数器 0 外部输入引脚	非独立 GPIO	nRF24L01 模块接口
T1	P3.5	计数器 1 外部输入引脚	非独立 GPIO	RS485 电路
T2	P3.1	计数器 2 外部输入引脚	非独立 GPIO	CH340 电路
T3	P0.5	计数器 3 外部输入引脚	非独立 GPIO	OLED、TFT 屏接口等
T4	P0.7	计数器 4 外部输入引脚	非独立 GPIO	LCD12864/LCD1602 接口

✧ 注：独立 GPIO 表示开发板没有其他的电路使用这个 GPIO，非独立 GPIO 说明开发板有其他电路用到了该 GPIO。针对非独立 GPIO 使用时需特别注意。

STC15W4K32S4 系列单片机定时器/计数器通过相关寄存器的 C/T 位选择计数器输入脉冲信号来源，也即选择了工作方式是定时方式还是计数方式。后配置相关寄存器位控制输入脉冲信号至计数器，计数器溢出后产生中断，也可通过特定引脚输出产生溢出时钟。定时器/计数器结构原理示意图如下。

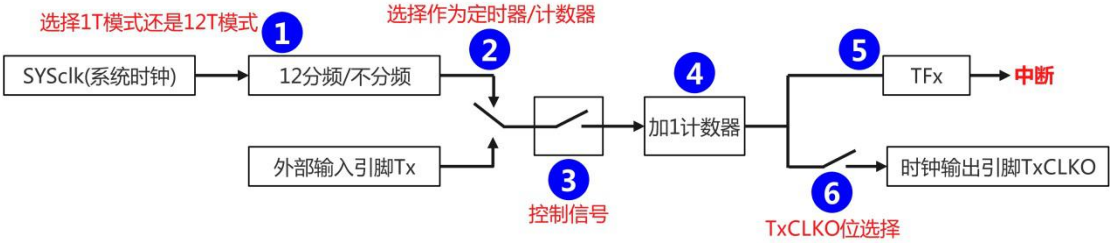

图 1：定时器/计数器结构原理示意图

✧ 注：12T 模式是选择对系统时钟 12 分频，1T 模式是选择对系统时钟不分频。另外，该计数器是递增计数器，不具有递减功能。

■ 定时器/计数器工作在定时方式（定时器）时：

1、 系统时钟进行输入计数，每输入一个脉冲，计数值加 1，当计数到计数器为全 1 时，再输入一个脉冲就使计数值回零，同时从最高位溢出一个脉冲使特殊功能寄存器 TCON 的 TFx 位置 1（T2、T3、T4 没有该寄存器位 TFx），作为计数器的溢出中断标志。

2、 由于计数脉冲的周期是固定的，所以脉冲数乘以脉冲周期就是定时时间，或者称定时溢出时间（关于定时溢出时间计算公式后有详述）。

3、 定时器可作为串口通信时的波特率发生器，需配置相关寄存器。

4、 可通过寄存器的 TxCLKO 位选择特定时钟输出引脚输出脉冲信号，该脉冲信号的频率等于计数器溢出率的一半。

表 2：单片机计数器时钟输出引脚分配

TxCLKO	对应 IO 口	功能描述	说明	备注
T0CLKO	P3.5	T0 时钟输出引脚	非独立 GPIO	RS485 电路
T1CLKO	P3.4	T1 时钟输出引脚	非独立 GPIO	nRF24L01 模块接口
T2CLKO	P3.0	T2 时钟输出引脚	非独立 GPIO	CH340 电路
T3CLKO	P0.4	T3 时钟输出引脚	非独立 GPIO	OLED、TFT 屏接口等
T4CLKO	P0.6	T4 时钟输出引脚	非独立 GPIO	LCD12864/LCD1602 接口

✧ 注：独立 GPIO 表示开发板没有其他的电路使用这个 GPIO，非独立 GPIO 说明开发板有其他电路用到了该 GPIO。针对非独立 GPIO 使用时需特别注意。

■ 定时器/计数器工作在计数方式（计数器）时：

1、 单片机特定引脚输入的外部脉冲信号源进行输入计数，每输入一个从 1 到 0 的负跳变的脉冲时，计数值加 1，当计数到计数器为全 1 时，再输入一个脉冲就使计数值回零，同时从最高位溢出一个脉冲使特殊功能寄存器 TCON 的 TFx 位置 1，作为计数器的溢出中断标志。

2、 外部输入信号的最高允许频率不能大于系统时钟频率 SYSclk 的 1/4，比如 CPU 运行的系统时钟为 11.0592MHz，允许外部最高输入信号频率为 11.0592MHz/4 = 2.7648MHz，如果频率高于这个值，输入信号的部分脉冲在检测过程中会被丢失，导致测量得到的频率比真实频率低。

3、 由于系统每个时钟对外部计数器引脚采样 1 次，当前一次采样到外部引脚为高电平而后一次采样到低电平则形成一个负跳变，因此确认外部输入信号的一次负跳变至少需要 2 个系统时钟周期，实际上，引脚输入通道中还有一个同步采样与边沿检测电路，所以外部输入信号的最高允许频率不能大于系统时钟频率 SYSclk 的 1/4。

3.3. 定时器/计数器工作模式

STC15W4K32S4 系列 MCU 不同的定时器/计数器所具有的可供选择的工作模式不同，定时器/计数器 0 和定时器/计数器 1 有多种模式可供选择，选择模式是通过寄存器 TMOD 对应的 M0 位和 M1 位实现。定时器/计数器 2、定时器/计数器 3 和定时器/计数器 4 只有默认的一种工作模式。具体如下面列表所示。

表 3：STC15W4K32S4 系列定时器/计数器工作模式

序号	定时器/计数器工作模式	T0	T1	T2	T3	T4
1	模式 0：16 位自动重装载模式	有	有	有	有	有
2	模式 1：16 位不可重装载模式	有	有			
3	模式 2：8 位自动重装载模式	有	有			
4	模式 3：不可屏蔽 16 位自动重装载模式	有				

✧ 注：模式 0 是 STC 官方推荐学习的定时器/计数器工作模式，也是我们讲解的重点。当 T0 工作在模式 3，T0 可作为实时操作系统用节拍定时器。

下面介绍下定时器/计数器 0 和定时器/计数器 1 使用时非常重要的一个寄存器 TMOD。

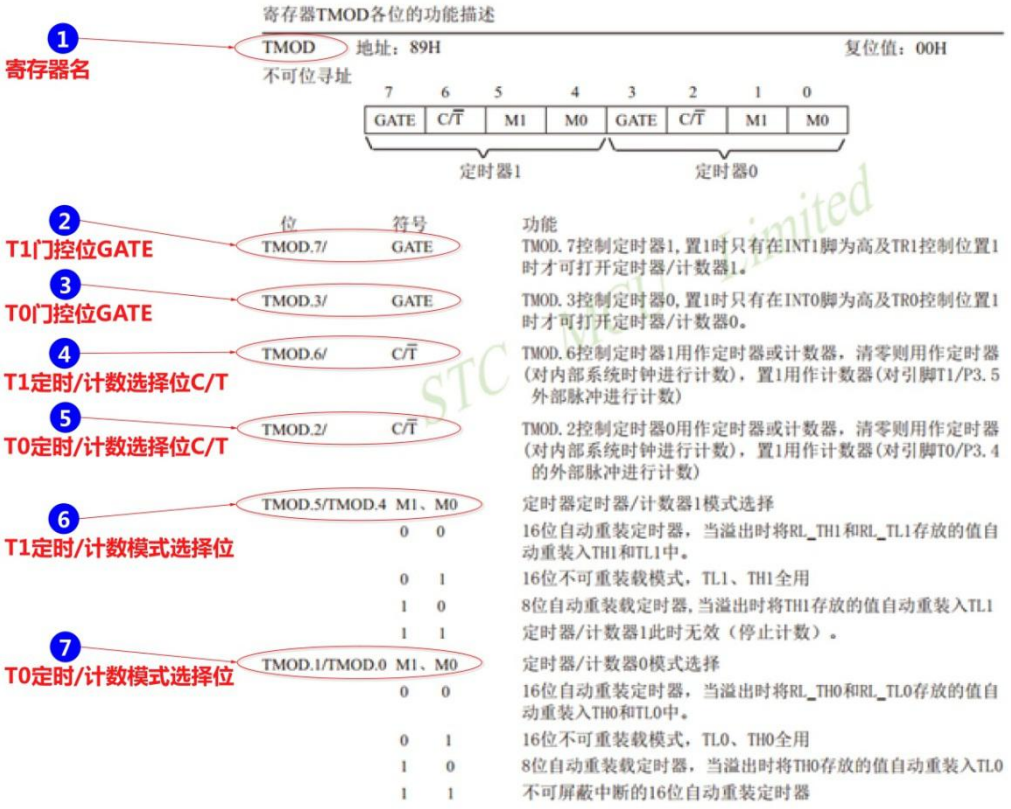

图 2：模式寄存器 TMOD

✧ 注：定时器/计数器的门控位在一些特殊应用中会有使用，一般情况下是将门控位置 0。

■ 定时器/计数器 0 和定时器/计数器 1 工作模式 0 分析。

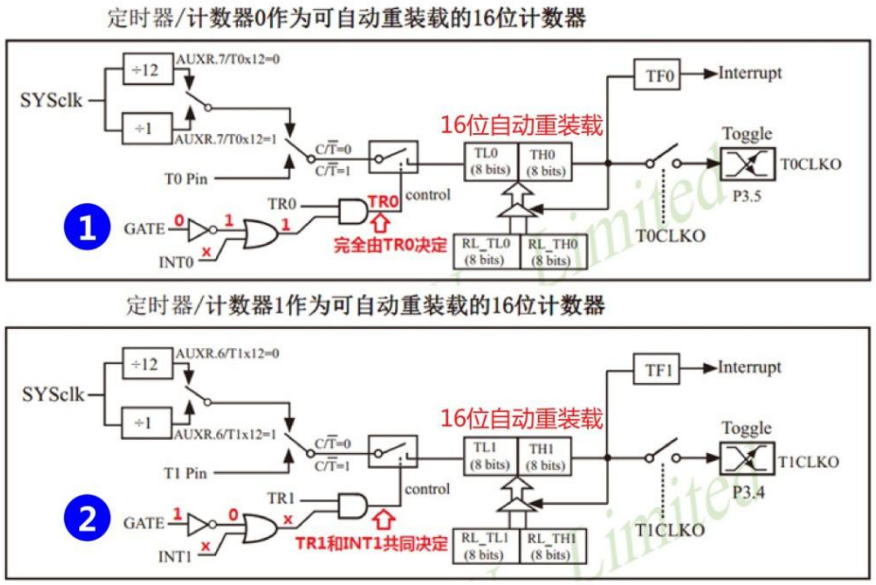

图 3：定时器/计数器 0 和定时器/计数器 1 模式 0 结构图

1） 定时器/计数器 0 和定时器/计数器 1 均有 GATE 门控位为 0、为 1 的情况。

2） 当 GATE=0 时，控制定时器/计数器完全由 TRx（定时器运行控制位）决定。

3） 当 GATE=1 时，控制定时器/计数器不仅由 TRx（定时器运行控制位）决定，还由外部中断引脚上的信号决定。此时，可用于脉宽测量。

4） 图中隐藏 2 个寄存器 RL_THx 和 RL_TLx，RL_THx 和 THx 共用同一个地址，RL_TLx 和 TLx 共用同一个地址。当 Tx 被禁止工作时，写入 THx 和 TLx 的内容会同时被写入 RL_THx 和 RL_TLx 中。当 Tx 开启工作时，准备写入 THx 和 TLx 的内容，其实没有被写入到写入 THx 和 TLx 中，而是写到了 RL_THx 和 RL_TLx 中。这样便巧妙的实现 16 位重装载定时。而读 THx 和 TLx 的内容时，读取的就是 THx 和 TLx 的内容，而不是 RL_THx 和 RL_TLx 中的值。

■ 定时器/计数器 2、定时器/计数器 3 和定时器/计数器 4 工作模式 0 分析。

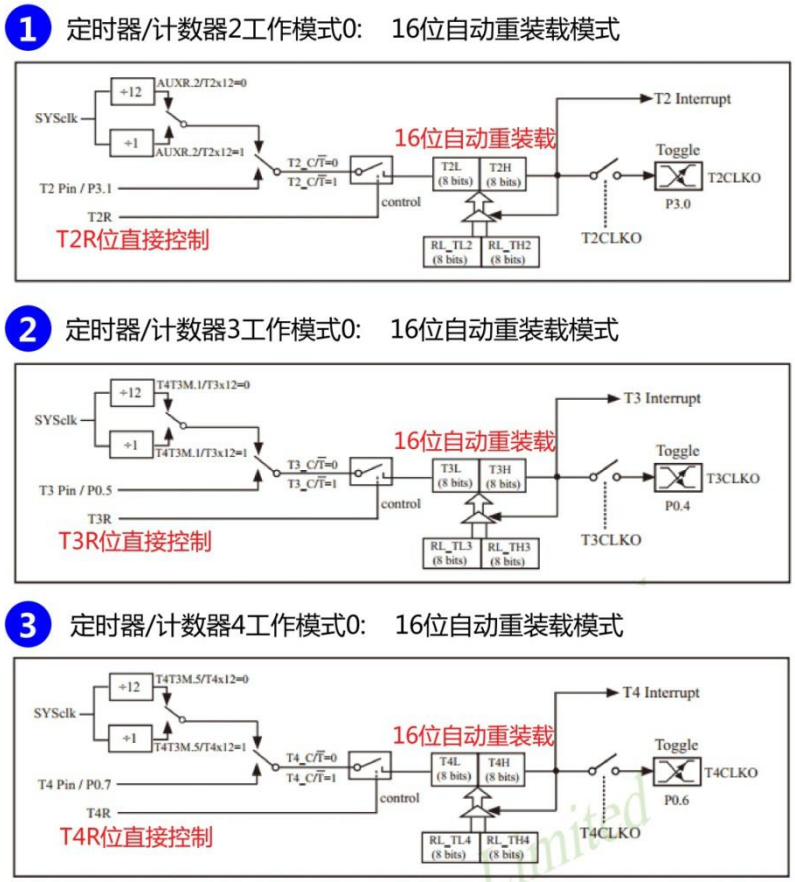

图 4：定时器/计数器 2、定时器/计数器 3 和定时器/计数器 4 模式 0 结构图

1） 定时器/计数器 2、定时器/计数器 3 和定时器/计数器 4 不存在门控位 GATE，对定时器/计数器的控制完全由 TxR（定时器运行控制位）决定。

2） 定时器/计数器 2、定时器/计数器 3 和定时器/计数器 4 有且只有一种模式，即 16 位重装载模式，实现重装载的原理请参考对定时器/计数器 0 和定时器/计数器的分析。

3.4. 定时器/计数器溢出时间计算

STC15W4K32S4 系列单片机作为定时器使用时，由于计数脉冲的周期是固定的，所以溢出前的脉冲数乘以脉冲周期就是定时时间，或者称定时溢出时间。

下面是定时器溢出时间计算公式：

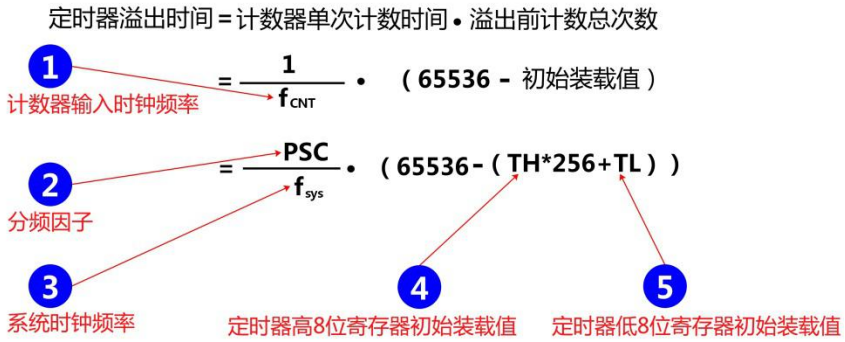

图 5：定时器溢出时间计算公式

✧ 注：公式中的分频因子 PSC 在配置定时器/计数器为 12T 模式时值为 12，在配置定时器/计数器为 1T 模式时值为 1。

举例，配置定时器/计数器为 1T 模式，系统时钟频率为 11.0592MHZ，高 8 位寄存器初始值为 0x28，低 8 位寄存器初始值为 0x00，计算下定时器溢出时间。

1） 十六进制 0x28 转成十进制是 40，十六进制 0x00 转成十进制是 0。这样初始装载值为：256*40+0=10240。

2） 16 位计数器溢出前所计脉冲数为：65536-10240=55296。

3） 分频因子 PSC 在 1T 模式下值为 1。系统时钟频率为 11059200HZ。

4） 定时器溢出时间：55296/11059200=0.005s=5ms。

5） 如果已知定时器溢出时间，计算高 8 位寄存器初始装载值和低 8 位寄存器初始装载值，则是反推过来即可（建议使用软件 STC-ISP 的定时器计算器）。

3.5. 定时器/计数器中断配置步骤

针对 STC15W4K32S4 系列单片机 5 个定时器/计数器外设，软件的配置过程如下：

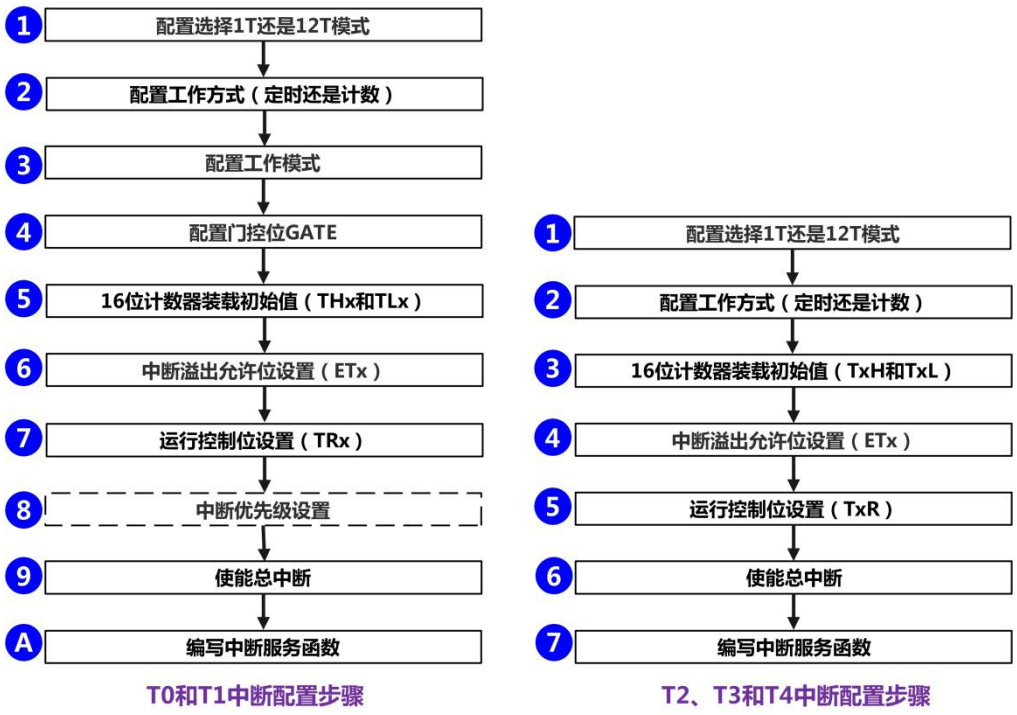

图 6：定时器/计数器中断软件配置步骤

✧ 注：实验例程即是按照上述配置步骤操作寄存器相关位实现，后有详述。

4. 软件设计

4.1. 定时器/计数器寄存器汇集

STC15W4K32S4 系列单片机操作定时器/计数器时会用到 18 个寄存器，如下表所示：

表 4：STC15W4K32S4 系列定时器/计数器使用寄存器汇总

序号	寄存器名	读/写	功能描述
1	TCON	读/写	定时器/计数器中断控制寄存器。
2	TMOD	读/写	定时器/计数器模式寄存器。
3	AUXR	读/写	辅助寄存器 1。
4	AUXR2	读/写	辅助寄存器 2。
5	IE	读/写	中断允许寄存器 1。
6	IE2	读/写	中断允许寄存器 2。
7	IP	读/写	中断优先级寄存器 1。
8	T4T3M	读/写	T3 和 T4 控制寄存器。
9	TL0	读/写	定时器/计数器 0 低 8 位寄存器。

10	TH0	读/写	定时器/计数器 0 高 8 位寄存器。
11	TL1	读/写	定时器/计数器 1 低 8 位寄存器。
12	TH1	读/写	定时器/计数器 1 高 8 位寄存器。
13	T2L	读/写	定时器/计数器 2 低 8 位寄存器。
14	T2H	读/写	定时器/计数器 2 高 8 位寄存器。
15	T3L	读/写	定时器/计数器 3 低 8 位寄存器。
16	T3H	读/写	定时器/计数器 3 高 8 位寄存器。
17	T4L	读/写	定时器/计数器 4 低 8 位寄存器。
18	T4H	读/写	定时器/计数器 4 高 8 位寄存器。

✧ 注：上述寄存器有部分不单单是用于定时器/计数器外设中的，比如寄存器 TCON、IE、IE2 等。

4.2. 寄存器解析

4.2.1. 中断允许寄存器 IE

外部中断允许寄存器 IE 支持位寻址，该寄存器的 B1 和 B3 位是定时器/计数器 0 和定时器/计数器 1 的中断允许位。

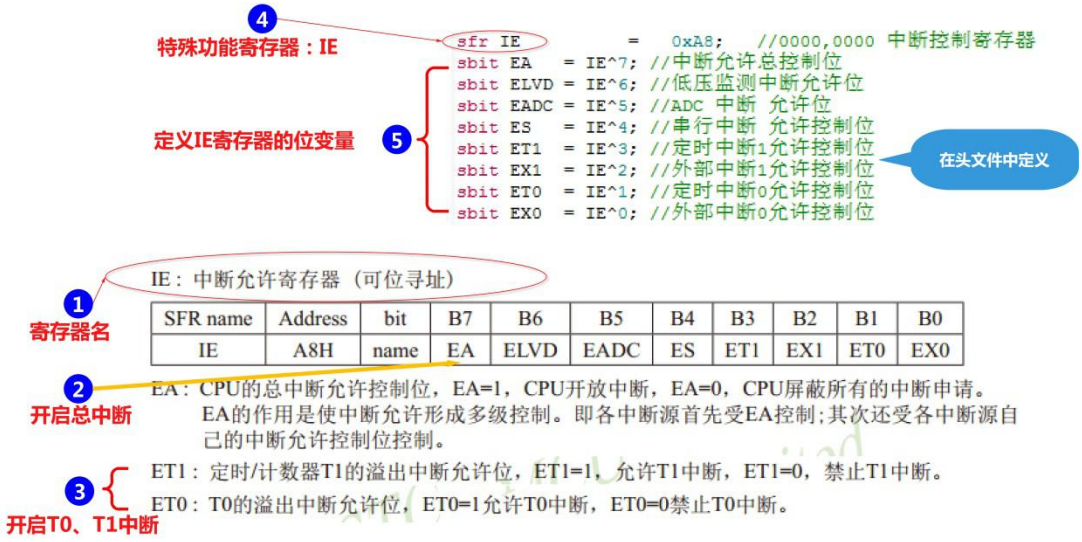

图 7：中断允许寄存器

4.2.2. 中断允许寄存器 IE2

中断允许寄存器 IE2 不支持位寻址，该寄存器的 B2、B5 和 B6 位是定时器/计数器 2、定时器/计数器 3 和定时器/计数器 4 的中断允许位。因为 IE2 寄存器不支持位寻址，所以举例操作该寄存器 B2 位时，不可以直接“ET2=0;”进行操作，参考下图。

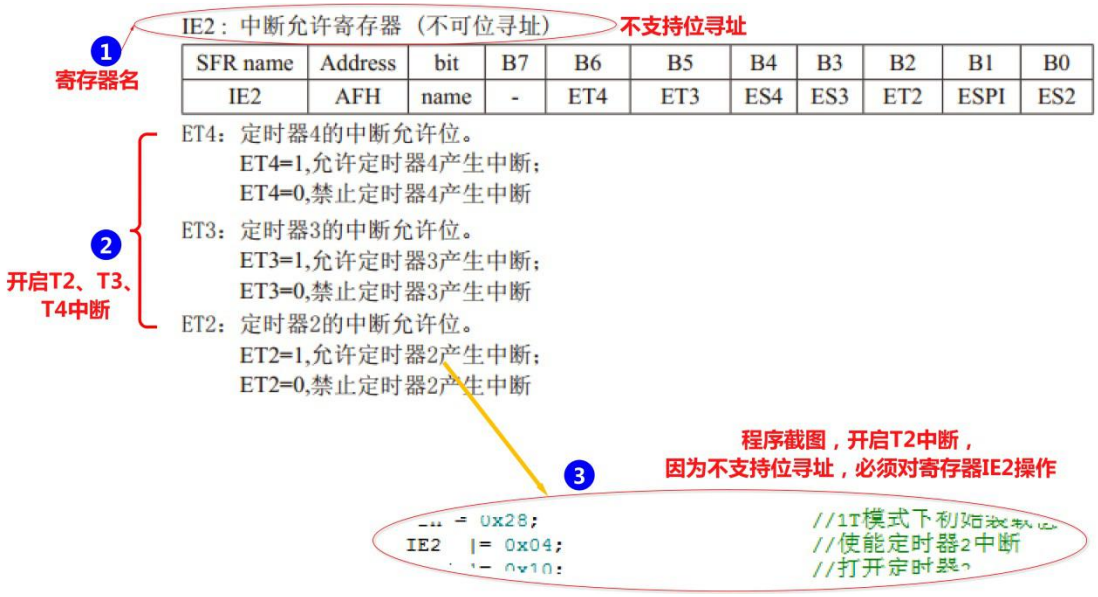

图 8：中断允许寄存器

4.2.3. 定时器/计数器中断控制寄存器 TCON

定时器/计数器中断控制寄存器 TCON 支持位寻址，该寄存器的 B4 位和 B6 位是 T0 和 T1 的运行控制位，寄存器 B5 位和 B7 位是 T0 和 T1 的溢出中断标志，含义如下图。

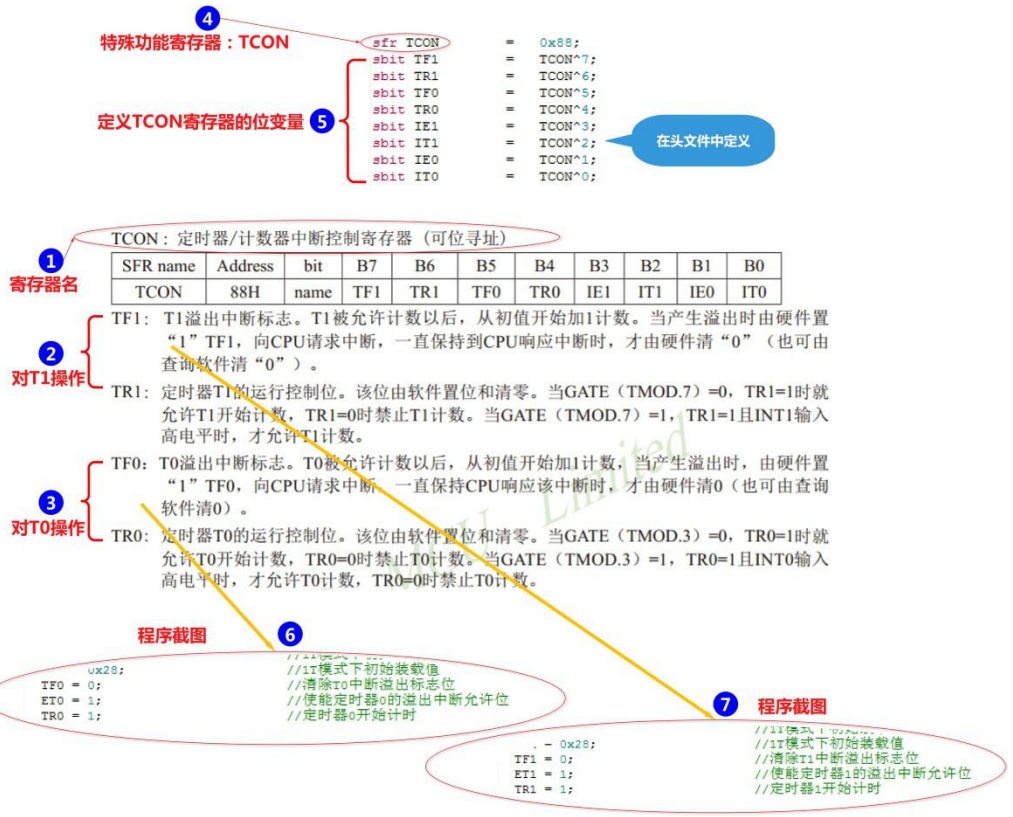

图 9：定时器/计数器中断控制寄存器

✧ 注：TCON 寄存器的低 4 个位是用于外部中断的，在操作该寄存器时一定要按位操作，用不到的位不要操作。

4.2.4. 辅助寄存器 AUXR

辅助寄存器 AUXR 不支持位寻址，该寄存器的 B6 和 B7 位是定时器/计数器 1 和定时器/计数器 0 的速度控制位，寄存器的 B1、B2 和 B3 位是定时器/计数器 2 的速度控制位、工作方式选择位和允许控制位。

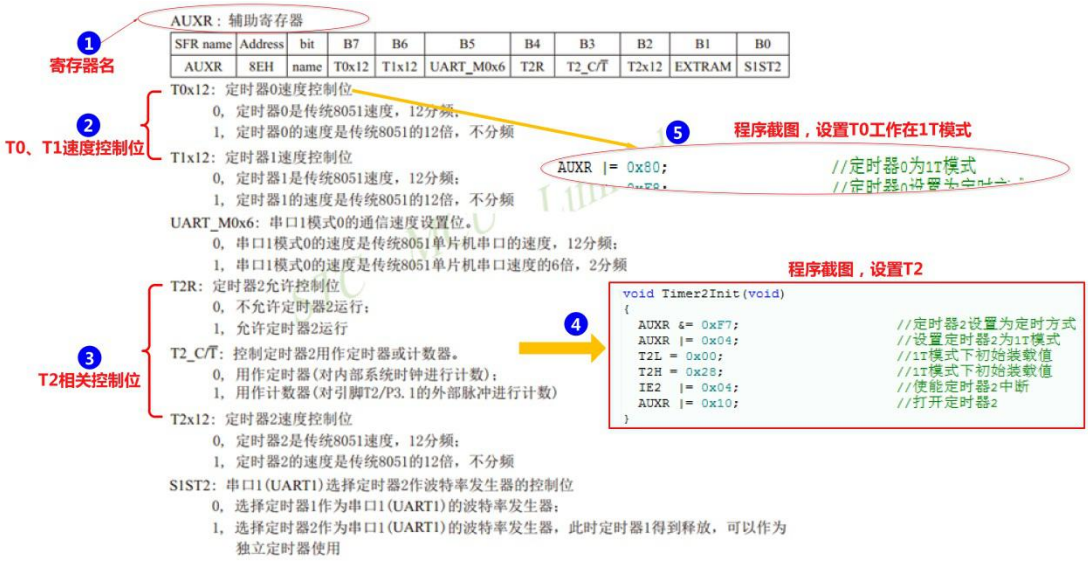

图 10：辅助寄存器

✧ 注：AUXR 寄存器的 B0 和 B5 位在串口外设配置时可能会用到。

4.2.5. 辅助寄存器 AUXR2

辅助寄存器 AUXR2 不支持位寻址，该寄存器的 B0、B1 和 B2 位是定时器/计数器 0、定时器/计数器 1 和定时器/计数器 2 的时钟输出引脚控制位。

图 11：辅助寄存器 2

✧ 注：AUXR2 寄存器的定时器时钟输出功能和定时器工作模式及工作方式（定时还是计数）密切相关，用户暂只需知道有这些关系，待项目应用时再分析频率关系。

4.2.6. 寄存器 T4T3M

寄存器 T4T3M 不支持位寻址，该寄存器的 B0 到 B3 位是定时器/计数器 3 配置过程中会用到的位，寄存器的 B4 到 B7 位是定时器/计数器 4 配置过程中会用到的位。

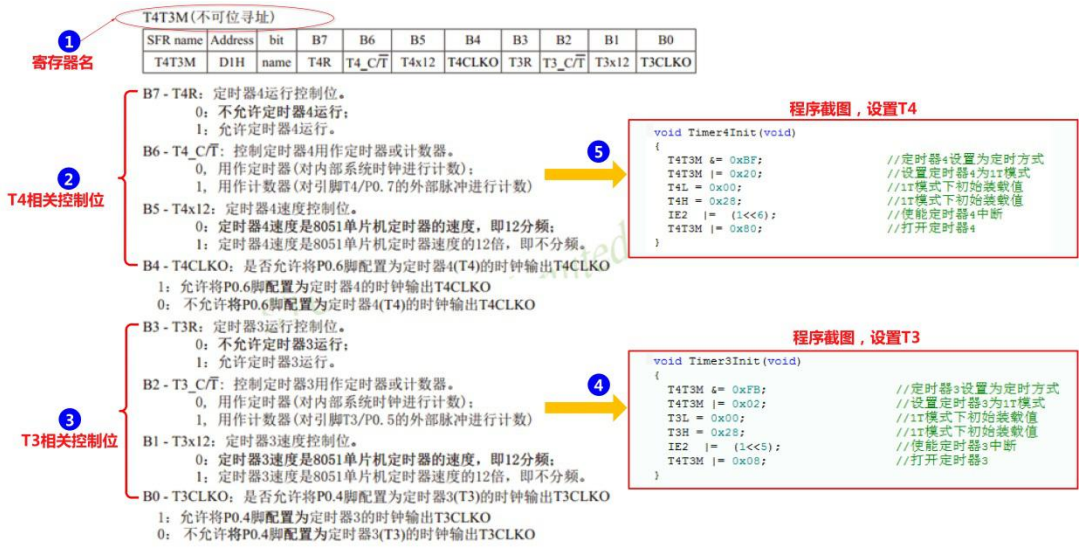

图 12：T3T4M 寄存器

4.2.7. 中断优先级控制寄存器 IP

中断优先级控制寄存器 IP 支持位寻址，该寄存器的 B1 位和 B3 位是设置 T0 和 T1 中断优先级的，含义如下图。需要说明的是 T2、T3 和 T4 是没有中断优先级的。

图 13：中断优先级控制寄存器

4.3. 定时器 0 定时实验

✧ 注：本节的实验源码是在“实验 2-4-1：外部中断 0（下降沿中断方式）”的基础上修改。本节对应的实验源码是：“实验 2-5-1：定时器 0 定时”。

4.3.1. 工程需要用到的 c 文件

本例需要用到的 c 文件如下表所示，工程需要添加下表中的 c 文件。

表 5：实验需要用到的 c 文件

序号	文件名	后缀	功能描述
1	led	.c	包含与用户 led 控制有关的用户自定义函数。
2	timer	.c	外部定时器有关的用户自定义函数。
3	delay	.c	包含用户自定义延时函数。

4.3.2. 头文件引用和路径设置

■ 需要引用的头文件

```
1. #include "delay.h"
2. #include "led.h"
3. #include "timer.h"
```

■ 需要包含的头文件路径

本例需要包含的头文件路径如下表：

表 6：头文件包含路径

序号	路径	描述
1	..\ Source	led.h、timer.h 和 delay.h 头文件在该路径，所以要包含。
2	..\User	15W4KxxS4.h 头文件在该路径，所以要包含。

MDK 中点击魔术棒，打开工程配置窗口，按照下图所示添加头文件包含路径。

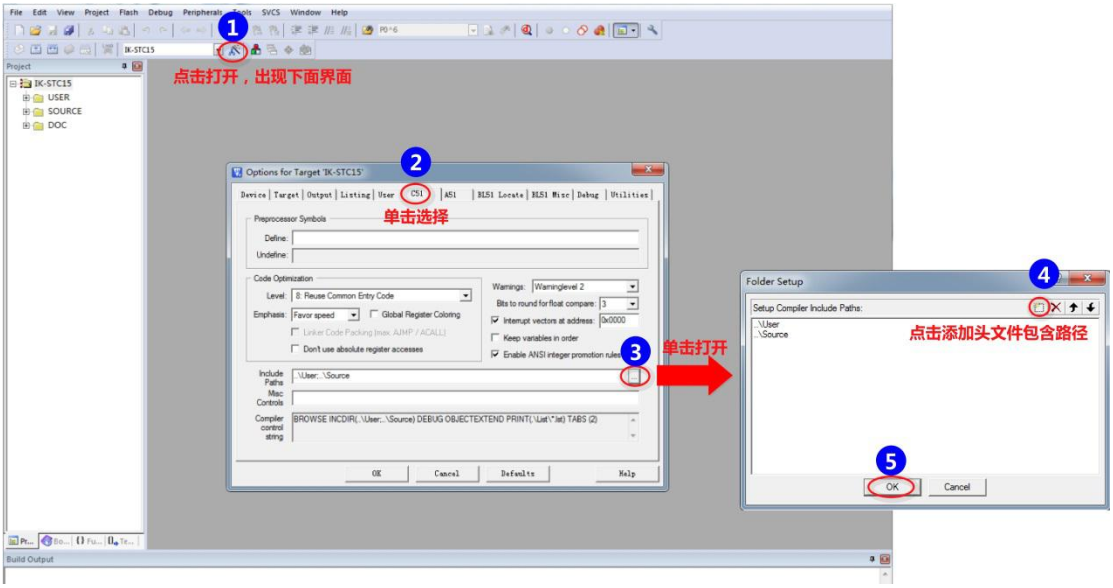

图 14：添加头文件包含路径

4.3.3. 编写代码

首先，在 timer.c 文件中编写定时器 0 的初始化函数 Timer0Init，代码如下。

程序清单：定时器 0 初始化函数

```
/*************************************************************
功能描述：定时器 0 初始化
入口参数：无
返回值：无
*************************************************************/
void Timer0Init(void)
{
    AUXR |= 0x80;                   //定时器 0 为 1T 模式
    TMOD &= 0xF8;                   //定时器 0 设置为定时方式，工作模式为 16 位自动重装模式
    TMOD &= 0xF7;                   //定时器 0 门控位 GATE 设置为 0
    TL0 = 0x00;                     //1T 模式下初始装载值
    TH0 = 0x28;                     //1T 模式下初始装载值
    TF0 = 0;                        //清除 T0 中断溢出标志位
    ET0 = 1;                        //使能定时器 0 的溢出中断允许位
    TR0 = 1;                        //定时器 0 开始计时
}
```

然后，编写定时器 0 中断服务函数，一旦响应中断达到一定次数会执行翻转蓝色指示灯 DS1 的任务，代码如下。

程序清单：中断服务函数

```
/*************************************************************
功能描述：定时器 0 中断服务程序
入口参数：无
返回值：无
*************************************************************/
void timer0_int (void) interrupt TIMER0_VECTOR
{
    cnt++;                      //5ms 进入 1 次中断
    if(cnt == 200)              //200 次中断被响应后，正好 1000ms
    {
        led_toggle(LED_1);      //翻转蓝色指示灯 DS1
        cnt = 0;
    }
    //进入中断时会将定时器中断溢出标志位硬件清零，因此下面一句可以不加的
  TF0 = 0;                    //清除 T0 中断溢出标志位
}
```

最后，在主函数中对 P0.6 口进行模式配置，调用 T0 初始化函数，开启总中断，在主循环中没有任务，蓝色指示灯 DS1 变化来自于中断。

代码清单：主函数

```
int main(void)
{
//////////////////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////////////////
    P0M1 &= 0x3F;   P0M0 &= 0x3F;     //设置 P0.6、P0.7 为准双向口

    Timer0Init();     //定时器 0 初始化
    EA = 1;           //使能总中断
    while (1)
    {
        ;               //无任务，说明 LED 亮灭来自于中断
    }
}
```

4.3.4. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1，请使用短路帽短接开发板 J23 端子的 LED1 和 P06（出厂开发板已默认短接）。

4.3.5. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-5-1：定时器 0 定时”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\Timer0\projec”目录下的工程“Timer0.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“Timer0.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，蓝色 LED 灯间隔 1s 状态翻转一次。

4.4. 定时器 1 定时实验

✧ 注：本节的实验源码是在“实验 2-5-1：定时器 0 定时”的基础上修改。本节对应的实验源码是：“实验 2-5-2：定时器 1 定时”。

4.4.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-5-1：定时器 0 定时”部分。

4.4.2. 编写代码

首先，在 timer.c 文件中编写定时器 1 的初始化函数 Timer1Init，代码如下。

程序清单：定时器 1 初始化函数

```
/*************************************************************
功能描述：定时器 1 初始化
入口参数：无
返回值：无
*************************************************************/
void Timer1Init(void)
{
    AUXR |= 0x40;                   //定时器 1 为 1T 模式
    TMOD &= 0x8F;                   //定时器 1 设置为定时方式，工作模式为 16 位自动重装模式
    TMOD &= 0x7F;                   //定时器 1 门控位 GATE 设置为 0
    TL1 = 0x00;                     //1T 模式下初始装载值
    TH1 = 0x28;                     //1T 模式下初始装载值
    TF1 = 0;                        //清除 T1 中断溢出标志位
    ET1 = 1;                        //使能定时器 1 的溢出中断允许位
    TR1 = 1;                        //定时器 1 开始计时
}
```

然后，编写定时器 1 中断服务函数，一旦响应中断达到一定次数会执行翻转蓝色指示灯 DS1 的任务，代码如下。

程序清单：中断服务函数

```
/*************************************************************
功能描述：定时器 1 中断服务程序
入口参数：无
返回值：无
*************************************************************/
void timer1_int (void) interrupt TIMER1_VECTOR
{
    cnt++;                      //5ms 进入 1 次中断
    if(cnt == 200)              //200 次中断被响应后，正好 1000ms
    {
    led_toggle(LED_1);      //翻转蓝色指示灯 DS1
        cnt = 0;
    }
```

```
    //进入中断时会将定时器中断溢出标志位硬件清零，因此下面一句可以不加的
  TF1 = 0;                    //清除 T1 中断溢出标志位
}
```

最后，在主函数中对 P0.6 口进行模式配置，调用 T1 初始化函数，开启总中断，在主循环中没有任务，蓝色指示灯 DS1 变化来自于中断。

代码清单：主函数

```
int main(void)
{
//////////////////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////////////////
    P0M1 &= 0x3F;   P0M0 &= 0x3F;     //设置 P0.6、P0.7 为准双向口

    Timer1Init();     //定时器 1 初始化
    EA = 1;           //使能总中断
  while (1)
    {
        ;               //无任务，说明 LED 亮灭来自于中断
    }
}
```

4.4.3. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1，请使用短路帽短接开发板 J23 端子的 LED1 和 P06（出厂开发板已默认短接）。

4.4.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-5-2：定时器 1 定时”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\Timer1\projec”目录下的工程“Timer1.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“Timer1.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，蓝色 LED 灯间隔 1s 状态翻转一次。

4.5. 定时器 2 定时实验

✧ 注：本节的实验源码是在“实验 2-5-1：定时器 0 定时”的基础上修改。本节对应的实验源码是：“实验 2-5-3：定时器 2 定时”。

4.5.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-5-1：定时器 0 定时”部分。

4.5.2. 编写代码

首先，在 timer.c 文件中编写定时器 2 的初始化函数 Timer2Init，代码如下。

程序清单：定时器 2 初始化函数

```
/*************************************************************
功能描述：定时器 2 初始化
入口参数：无
返回值：无
*************************************************************/
void Timer2Init(void)
{
    AUXR &= 0xF7;                   //定时器 2 设置为定时方式
    AUXR |= 0x04;                   //设置定时器 2 为 1T 模式
    T2L = 0x00;                     //1T 模式下初始装载值
    T2H = 0x28;                     //1T 模式下初始装载值
    IE2  |= 0x04;                     //使能定时器 2 中断
    AUXR |= 0x10;                   //打开定时器 2
}
```

然后，编写定时器 2 中断服务函数，一旦响应中断达到一定次数会执行翻转蓝色指示灯 DS1 的任务，代码如下。

程序清单：中断服务函数

```
/*************************************************************
功能描述：定时器 2 中断服务程序
入口参数：无
返回值：无
*************************************************************/
void timer2_int (void) interrupt TIMER2_VECTOR
{
    cnt++;          //5ms 进入 1 次中断
    if(cnt == 200)            //200 次中断被响应后，正好 1000ms
    {
    led_toggle(LED_1);      //翻转蓝色指示灯 DS1
        cnt = 0;
```

```
    }
}
```

最后，在主函数中对 P0.6 口进行模式配置，调用 T2 初始化函数，开启总中断，在主循环中没有任务，蓝色指示灯 DS1 变化来自于中断。

代码清单：主函数

```
int main(void)
{
/////////////////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
/////////////////////////////////////////////////////////
    P0M1 &= 0x3F;   P0M0 &= 0x3F;     //设置 P0.6、P0.7 为准双向口

    Timer2Init();     //定时器 2 初始化
    EA = 1;           //使能总中断
    while (1)
    {
        ;               //无任务，说明 LED 亮灭来自于中断
    }
}
```

4.5.3. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1，请使用短路帽短接开发板 J23 端子的 LED1 和 P06（出厂开发板已默认短接）。

4.5.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-5-3：定时器 2 定时”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\Timer2\projec”目录下的工程“Timer2.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“Timer2.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，蓝色 LED 灯间隔 1s 状态翻转一次。

4.6. 定时器 3 定时实验

✧ 注：本节的实验源码是在“实验 2-5-1：定时器 0 定时”的基础上修改。本节对应的实验源码是：“实验 2-5-4：定时器 3 定时”。

4.6.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-5-1：定时器 0 定时”部分。

4.6.2. 编写代码

首先，在 timer.c 文件中编写定时器 3 的初始化函数 Timer3Init，代码如下。

程序清单：定时器 3 初始化函数

```
/*************************************************************
功能描述：定时器 3 初始化
入口参数：无
返回值：无
*************************************************************/
void Timer3Init(void)
{
    T4T3M &= 0xFB;                  //定时器 3 设置为定时方式
    T4T3M |= 0x02;                  //设置定时器 3 为 1T 模式
    T3L = 0x00;                     //1T 模式下初始装载值
    T3H = 0x28;                     //1T 模式下初始装载值
    IE2  |=  (1<<5);                //使能定时器 3 中断
    T4T3M |= 0x08;                  //打开定时器 3
}
```

然后，编写定时器 3 中断服务函数，一旦响应中断达到一定次数会执行翻转蓝色指示灯 DS1 的任务，代码如下。

程序清单：中断服务函数

```
/*************************************************************
功能描述：定时器 3 中断服务程序
入口参数：无
返回值：无
*************************************************************/
void timer3_int (void) interrupt TIMER3_VECTOR
{
    cnt++;          //5ms 进入 1 次中断
    if(cnt == 200)            //200 次中断被响应后，正好 1000ms
    {
        led_toggle(LED_1);      //翻转蓝色指示灯 DS1
        cnt = 0;
```

```
    }
}
```

最后，在主函数中对 P0.6 口进行模式配置，调用 T3 初始化函数，开启总中断，在主循环中没有任务，蓝色指示灯 DS1 变化来自于中断。

代码清单：主函数

```
int main(void)
{
//////////////////////////////////////////////////////////
//注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
//       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
//相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
//         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
//////////////////////////////////////////////////////////
    P0M1 &= 0x3F;   P0M0 &= 0x3F;     //设置 P0.6、P0.7 为准双向口

    Timer3Init();     //定时器 3 初始化
    EA = 1;           //使能总中断
  while (1)
    {
        ;               //无任务，说明 LED 亮灭来自于中断
    }
}
```

4.6.3. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1，请使用短路帽短接开发板 J23 端子的 LED1 和 P06（出厂开发板已默认短接）。

4.6.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-5-4：定时器 3 定时”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\Timer3\projec”目录下的工程“Timer3.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“Timer3.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，蓝色 LED 灯间隔 1s 状态翻转一次。

4.7. 定时器 4 定时实验

✧ 注：本节的实验源码是在“实验 2-5-1：定时器 0 定时”的基础上修改。本节对应的实验源码是：“实验 2-5-5：定时器 4 定时”。

4.7.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-5-1：定时器 0 定时”部分。

4.7.2. 编写代码

首先，在 timer.c 文件中编写定时器 4 的初始化函数 Timer4Init，代码如下。

程序清单：定时器 4 初始化函数

```
/*************************************************************
功能描述：定时器 4 初始化
入口参数：无
返回值：无
*************************************************************/
void Timer4Init(void)
{
    T4T3M &= 0xBF;                  //定时器 4 设置为定时方式
    T4T3M |= 0x20;                  //设置定时器 4 为 1T 模式
    T4L = 0x00;                     //1T 模式下初始装载值
    T4H = 0x28;                     //1T 模式下初始装载值
    IE2  |=  (1<<6);                //使能定时器 4 中断
    T4T3M |= 0x80;                  //打开定时器 4
}
```

然后，编写定时器 4 中断服务函数，一旦响应中断达到一定次数会执行翻转蓝色指示灯 DS1 的任务，代码如下。

程序清单：中断服务函数

```
/*************************************************************
功能描述：定时器 4 中断服务程序
入口参数：无
返回值：无
*************************************************************/
void timer4_int (void) interrupt TIMER4_VECTOR
{
    cnt++;          //5ms 进入 1 次中断
    if(cnt == 200)            //200 次中断被响应后，正好 1000ms
    {
        led_toggle(LED_1);      //翻转蓝色指示灯 DS1
        cnt = 0;
```

```
13.     }
14. }
```

最后，在主函数中对 P0.6 口进行模式配置，调用 T4 初始化函数，开启总中断，在主循环中没有任务，蓝色指示灯 DS1 变化来自于中断。

代码清单：主函数

```
1.  int main(void)
2.  {
3.  ////////////////////////////////////////////////////////
4.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
6.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
7.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
8.  ////////////////////////////////////////////////////////
9.      P0M1 &= 0x3F;   P0M0 &= 0x3F;     //设置 P0.6、P0.7 为准双向口
10.
11.     Timer4Init();     //定时器 4 初始化
12.     EA = 1;           //使能总中断
13.   while (1)
14.     {
15.         ;               //无任务，说明 LED 亮灭来自于中断
16.     }
17. }
```

4.7.3. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1，请使用短路帽短接开发板 J23 端子的 LED1 和 P06（出厂开发板已默认短接）。

4.7.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-5-5：定时器 4 定时”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\Timer4\projec”目录下的工程“Timer4.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“Timer4.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，蓝色 LED 灯间隔 1s 状态翻转一次。

4.8. 多个定时器定时实验

✧ 注：本节的实验源码是在“实验 2-5-1：定时器 0 定时”的基础上修改。本节对应的实验源码是：“实验 2-5-6：多个定时器定时”。

4.8.1. 工程需要用到的 c 文件

本实验需要用到的头文件以及添加头文件包含路径的方法请参考“实验 2-5-1：定时器 0 定时”部分。

4.8.2. 编写代码

首先，在 timer.c 文件中编写定时器 0 和定时器 1 的初始化函数 Timer0Init 和 Timer1Init，代码如下。

程序清单：定时器 0 和定时器 1 初始化函数

```
/*************************************************************
功能描述：定时器 0 初始化
入口参数：无
返回值：无
*************************************************************/
void Timer0Init(void)
{
    AUXR |= 0x80;                   //定时器 0 为 1T 模式
    TMOD &= 0xF8;                   //定时器 0 设置为定时方式，工作模式为 16 位自动重装模式
    TMOD &= 0xF7;                   //定时器 0 门控位 GATE 设置为 0
    TL0 = 0x00;                     //1T 模式下初始装载值
    TH0 = 0x28;                     //1T 模式下初始装载值
    TF0 = 0;                        //清除 T0 中断溢出标志位
    ET0 = 1;                        //使能定时器 0 的溢出中断允许位
    TR0 = 1;                        //定时器 0 开始计时
}
/*************************************************************
功能描述：定时器 1 初始化
入口参数：无
返回值：无
*************************************************************/
void Timer1Init(void)
{
    AUXR |= 0x40;                   //定时器 1 为 1T 模式
    TMOD &= 0x8F;                   //定时器 1 设置为定时方式，工作模式为 16 位自动重装模式
    TMOD &= 0x7F;                   //定时器 1 门控位 GATE 设置为 0
    TL1 = 0x00;                     //1T 模式下初始装载值
    TH1 = 0x28;                     //1T 模式下初始装载值
    TF1 = 0;                        //清除 T1 中断溢出标志位
    ET1 = 1;                        //使能定时器 1 的溢出中断允许位
```

```
    TR1 = 1;                            //定时器 1 开始计时
}
```

然后，编写定时器 0 和定时器 1 的中断服务函数，一旦响应中断达到一定次数会执行翻转对应用户指示灯的操作，代码如下。

程序清单：中断服务函数

```
/*************************************************************
功能描述：定时器 0 中断服务程序
入口参数：无
返回值：无
*************************************************************/
void timer0_int (void) interrupt TIMER0_VECTOR
{
    cnt1++;         //5ms 进入 1 次中断
    if(cnt1 == 100)            //100 次中断被响应后，正好 500ms
    {
        led_toggle(LED_1);      //翻转蓝色指示灯 DS1
        cnt1 = 0;
    }
    //进入中断时会将定时器中断溢出标志位硬件清零，因此下面一句可以不加的
  TF0 = 0;                  //清除 T0 中断溢出标志位
}
/*************************************************************
功能描述：定时器 1 中断服务程序
入口参数：无
返回值：无
*************************************************************/
void timer1_int (void) interrupt TIMER1_VECTOR
{
    cnt2++;                     //5ms 进入 1 次中断
    if(cnt2 == 200)             //200 次中断被响应后，正好 1000ms
    {
        led_toggle(LED_2);      //翻转红色指示灯 DS2
        cnt2 = 0;
    }
    //进入中断时会将定时器中断溢出标志位硬件清零，因此下面一句可以不加的
  TF1 = 0;                  //清除 T1 中断溢出标志位
}
```

最后，在主函数中对 P0.6 和 P0.7 口进行模式配置，调用 T0 和 T1 初始化函数，开启总中断，在主循环中没有任务，蓝色指示灯 DS1 和红色指示灯 DS2 变化来自于中断。

代码清单：主函数

```
1.  int main(void)
2.  {
3.  ////////////////////////////////////////////////////////
4.  //注意: STC15W4K32S4 系列的芯片,上电后所有与 PWM 相关的 IO 口均为
5.  //       高阻态,需将这些口设置为准双向口或强推挽模式方可正常使用
6.  //相关 IO: P0.6/P0.7/P1.6/P1.7/P2.1/P2.2
7.  //         P2.3/P2.7/P3.7/P4.2/P4.4/P4.5
8.  ////////////////////////////////////////////////////////
9.      P0M1 &= 0x3F;   P0M0 &= 0x3F;      //设置 P0.6、P0.7 为准双向口
10.
11.     Timer0Init();      //定时器 0 初始化
12.     Timer1Init();      //定时器 1 初始化
13.     EA = 1;            //使能总中断
14.   while (1)
15.     {
16.         ;                 //无任务，说明 LED 亮灭来自于中断
17.     }
18. }
```

4.8.3. 硬件连接

本实验需要使用 P0.6 驱动蓝色指示灯 DS1、P0.7 驱动红色指示灯 DS2，请使用短路帽短接开发板 J23 端子的 LED1 和 P06、开发板 J23 端子的 LED2 和 P07（出厂开发板已默认短接）。

4.8.4. 实验步骤

1. 解压“…\第 3 部分：配套例程源码\ 1 - 基础实验程序\”目录下的压缩文件“实验 2-5-6：多个定时器定时”，将解压后得到的文件夹拷贝到合适的目录，如“D\STC15”。
2. 启动 Keil C51。
3. 在 Keil C51 中执行“Project→Open Project”打开“…\Timers\projec”目录下的工程“Timers.uvproj”。
4. 点击编译按钮编译工程。注意查看编译输出栏，观察编译的结果，如果有错误，修改程序，直到编译成功为止。编译后生成的 HEX 文件“Timers.hex”位于工程目录下的“Output”文件夹中。
5. 打开 STC-ISP 软件下载程序 。下载使用内部 IRC 时钟，IRC 频率选择为 11.0592MHZ。
6. 程序运行后，蓝色 LED 灯间隔 0.5s 状态翻转一次，红色 LED 灯间隔 1s 状态翻转一次。

■ **思考题：本例可以设置 T0 和 T1 的中断优先级吗？会有什么样的实验现象呢？**