参照实验课 Profile_4C123 工程文件

1. 系统时钟 50MHZ,使用 systick 定时器定时，调用 SysTick_Init(304)，传递参数 period =304，得到 164 kHz 中断频率，如果我们需要 100KHZ 的中断频率，那传递参数 period=？，

答案：

Period = 50MHZ/100KHZ=500

2. Timer0A 的初值设定由两部分构成，一个是预分频 TIMER0_TAPR_R = 49，和分频值 TIMER0_TAILR_R=5-1，实际分频为 50 乘 5，得到 200KHz 的中断频率，现在我们设定 TIMER0_TAPR_R = 9，现在想得到 50KHZ 的中断频率，那么 TIMER0_TAILR_R==？

答案：

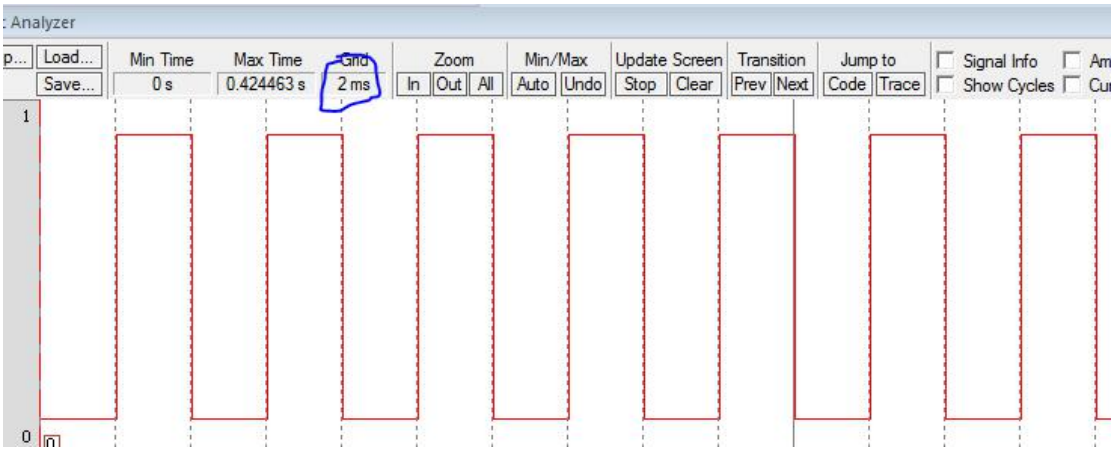
50KHz = 50MHz/(1+9)/(TIMER0_TAILR_R+1)

TIMER0_TAILR_R = 50MHz/50KHz/10-1=100-1=99

实验部分：

系统时钟中断以及中断优先级抢占现象分析实验

1. 系统时钟中断



关键代码分析：

```
void SysTick_Init(uint32_t period){long sr;
  sr = StartCritical();
  NVIC_ST_CTRL_R = 0;         // disable SysTick during setup
  NVIC_ST_RELOAD_R = period-1;// reload value
  NVIC_ST_CURRENT_R = 0;      // any write to current clears it
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x40000000; // priority 2
                              // enable SysTick with core clock and interrupts
  NVIC_ST_CTRL_R = NVIC_ST_CTRL_ENABLE+NVIC_ST_CTRL_CLK_SRC+NVIC_ST_CTRL_INTEN;
  EndCritical(sr);
}
```

系统时钟初始化函数设置了重装值和当前值，将中断允许信号打开，并且设置了系统时钟中断的优先级

NVIC_SYS_PRI3_R 的高 8 位分配给了系统时钟的优先级定义，系统时钟只占用前三位，0x40000000 代表将系统时钟优先级设置为 010 即为 2

```
// Executed every 12.5ns*(period
void SysTick_Handler(void){
  Counts = Counts + 1;
  PF2 ^= 0x04;                //
}
```

中断处理程序中 PF2 翻转一次，即每当系统时钟中断发生时，PF2 都会发生翻转。

2. 中断的优先级抢占

1）修改优先级之前

系统时钟的优先级=2

```
void SysTick_Init(uint32_t period){
  Counts = 0;
  NVIC_ST_CTRL_R = 0;               // disable SysTick during setup
  NVIC_ST_RELOAD_R = period - 1; // reload value
  NVIC_ST_CURRENT_R = 0;          // any write to current clears it
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x40000000; //priority 2
  NVIC_ST_CTRL_R = 0x00000007;   // enable with core clock and interrupts
}
```

定时器 A 优先级=3

```
void Timer0A_Init(unsigned short period){ volatile uint32_t delay;
  SYSCTL_RCGCTIMER_R |= 0x01;       // 0) activate timer0
  delay = SYSCTL_RCGCTIMER_R;       // allow time to finish activating
  TIMER0_CTL_R &= ~0x00000001;      // 1) disable timer0A during setup
  TIMER0_CFG_R = 0x00000004;        // 2) configure for 16-bit timer mode
  TIMER0_TAMR_R = 0x00000002;       // 3) configure for periodic mode
  TIMER0_TAILR_R = period - 1;      // 4) reload value
  TIMER0_TAPR_R = 49;               // 5) 1us timer0A
  TIMER0_ICR_R = 0x00000001;        // 6) clear timer0A timeout flag
  TIMER0_IMR_R |= 0x00000001;       // 7) arm timeout interrupt
  NVIC_PRI4_R = (NVIC_PRI4_R&0x00FFFFFF)|0x60000000; // 8) priority 3
  NVIC_EN0_R = NVIC_EN0_INT19;      // 9) enable interrupt 19 in NVIC
  TIMER0_CTL_R |= 0x00000001;       // 10) enable timer0A
}
```

所以理论上应该有系统时钟中断在定时器 A 中断时抢占中断现象

表现在波形上为：PF4 在 PF3 置位时置位，并因此延长了 PF3 的置位时间。

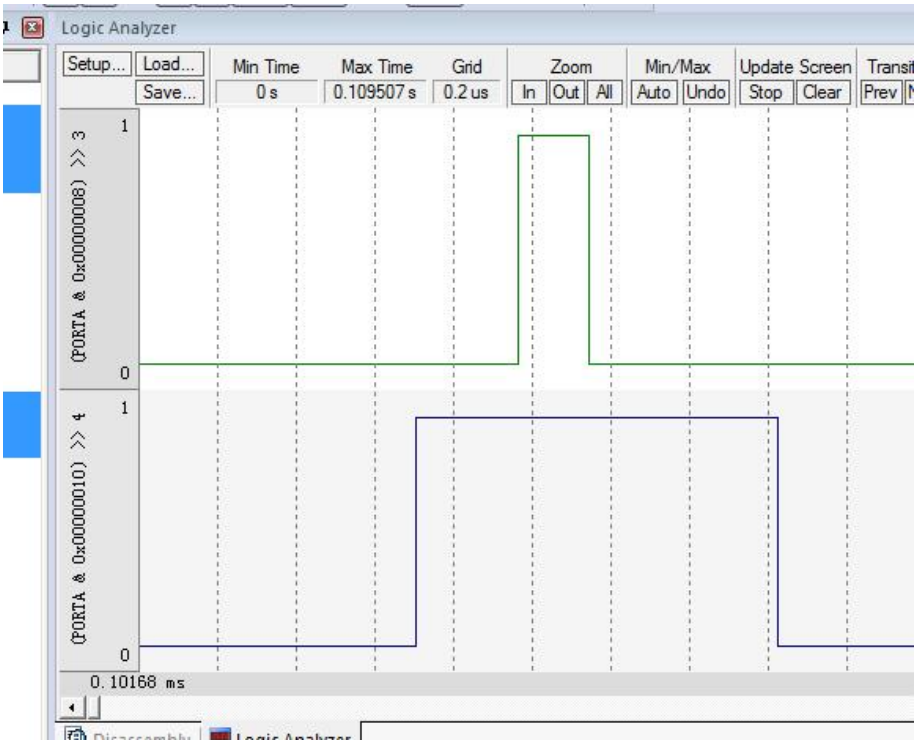
2）修改优先级之后：

将系统时钟优先级修改为 4 即 100，表现在 NVIC_SYS_PRI3_R 寄存器上就是 0X80000000

定时器 A 优先级不变，保持为 3

```
void SysTick_Init(uint32_t period){
  Counts = 0;
  NVIC_ST_CTRL_R = 0;            // disable SysTick during setup
  NVIC_ST_RELOAD_R = period - 1; // reload value
  NVIC_ST_CURRENT_R = 0;         // any write to current clears it
  NVIC_SYS_PRI3_R = (NVIC_SYS_PRI3_R&0x00FFFFFF)|0x80000000; //priority 2
  NVIC_ST_CTRL_R = 0x00000007;   // enable with core clock and interrupts
}
```

重新编译链接，运行项目后发现波形有所不同了



PF3 的置位出现在 PF4 置位和复位之间，并因此延长了 PF4 的置位时间，表明定时器 A 中断抢占了系统时钟中断。

结果表明优先级设定数值越小，中断优先级越大