嵌入式系统实验报告

实验名称：	中断与多任务调度
姓　　名：	陈姝仪
学　　号：	2018211507
学 院(系)：	计算机学院
专　　业：	网络工程
指导教师：	刘健培

2020 年 12 月 31 日

1 实验目的

- 通过 FSM4 实验板了解实验的软硬件环境，熟悉 MDK 开发环境的使用。
- 学习查阅文档和数据手册，获取需要的信息。
- 学习使用 STM32 定时器的基本操作方式。
- 掌握 STM32 中断处理方式。
- 学习基本的多任务处理方式。

2 实验环境

- FS-STM32F407 开发平台
- ST-Link 仿真器
- RealView MDK5.23 集成开发软件
- PC 机 Window7/8/10 (32/64bit)
- 串口调试工具

3 实验要求

在实验 1、2、3 的基础上扩展中断功能，并实现多任务调度。

- 将实验 1 的上下文切换功能扩展为多任务调度功能，并通过定时器中断实现时间片轮转调度。(相当于一个“微型”的嵌入式操作系统内核。)
- 将实验 2 的按键读取从轮询方式扩展为中断方式，其余功能不变。
- 将实验 3 的串口字符收发从从轮询方式扩展为中断方式，其余功能不变
- 将实验 2、实验 3 的功能实现为本实验中的任务，实现多任务并发运行。
    - Led 闪烁
    - 串口 shell 输入输出
    - 扫描按键

4 实验原理

14.3.2 计数器模式

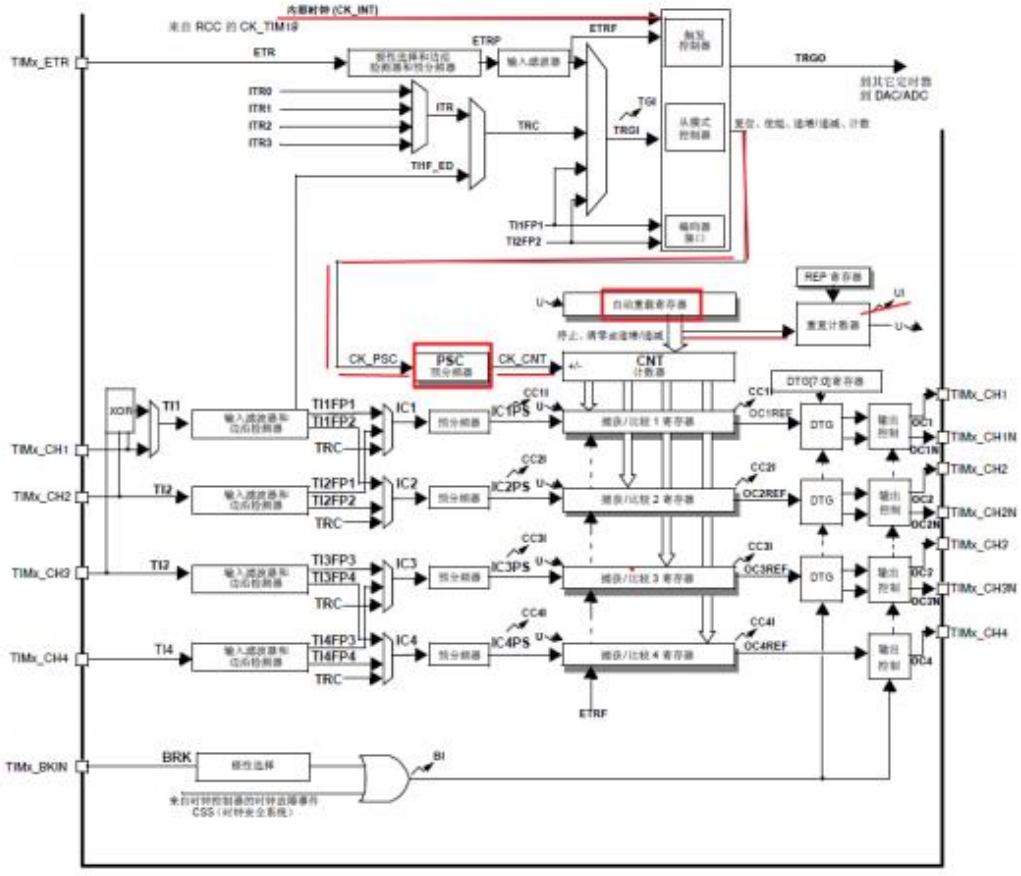
递增计数模式

在递增计数模式下，计数器从 0 计数到自动重载值（TIMx_ARR 寄存器的内容），然后重新从 0 开始计数并生成计数器上溢事件。

如果使用重复计数器，则当递增计数的重复次数达到重复计数器寄存器中编程的次数加一次 (TIMx_RCR+1) 后，将生成更新事件 (UEV)。否则，将在每次计数器上溢时产生更新事件。

将 TIMx_EGR 寄存器的 UG 位置 1（通过软件或使用从模式控制器）时，也将产生更新事件。

其原理如下图：



计数值的计算公式：

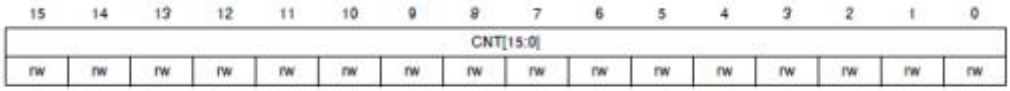
计数器时钟频率 (CK_CNT) 等于 f_{CK_PSC} / (PSC[15:0] + 1)

14.4.10 TIM1 和 TIM8 计数器 (TIMx_CNT)

TIM1&TIM8 counter

偏移地址：0x24

复位值：0x0000



位 15:0 **CNT[15:0]**：计数器值 (Counter value)

14.4.11 TIM1 和 TIM8 预分频器 (TIMx_PSC)

TIM1&TIM8 prescaler

偏移地址：0x28

复位值：0x0000

位 15:0 **PSC[15:0]**：预分频器值 (Prescaler value)

计数器时钟频率 (CK_CNT) 等于 f_{CK_PSC} / (PSC[15:0] + 1)。

PSC 包含每次发生更新事件（包括计数器通过 TIMx_EGR 寄存器中的 UG 位清零时，或在配置为“复位模式”时通过触发控制器清零时）时要装载到活动预分频器寄存器的值。

14.4.12 TIM1 和 TIM8 自动重载寄存器 (TIMx_ARR)

TIM1&TIM8 auto-reload register

偏移地址：0x2C

复位值：0x0000

位 15:0 **ARR[15:0]**：自动重载值 (Auto-reload value)

ARR 为要装载到实际自动重载寄存器的值。

有关 ARR 更新和行为的详细信息，请参见*第 331 页的第 14.3.1 节：时基单元*。

当自动重载值为空时，计数器不工作。

本实验主要使用后台轮询与前台中断驱动的程序结构：

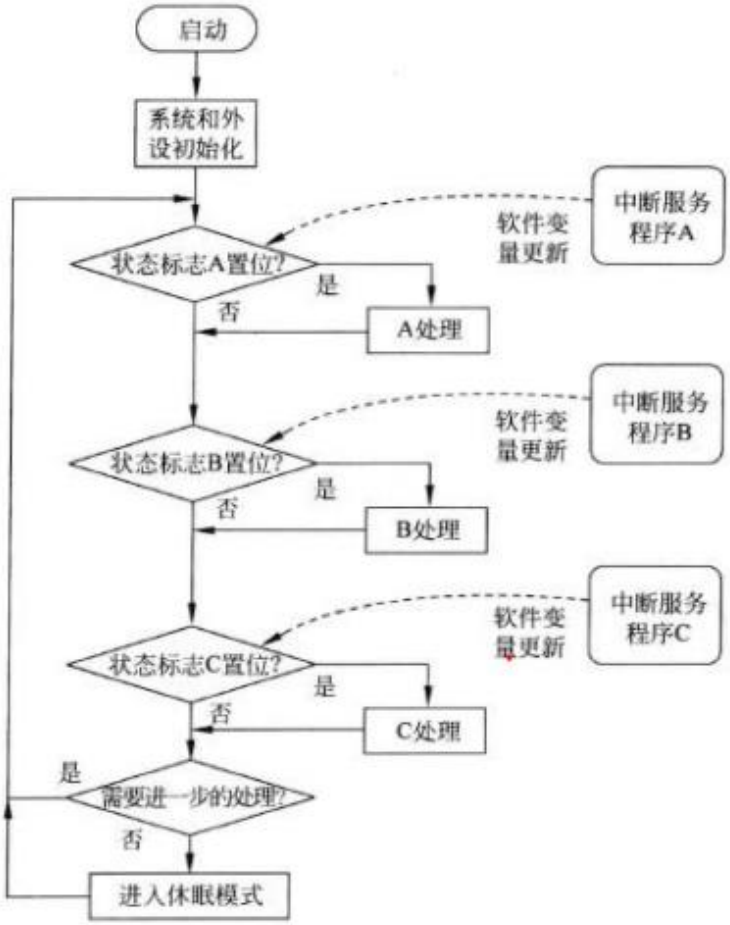


图 2.10　使用轮询和中断驱动两种方式的应用

5 实验步骤

5.1 基础部分

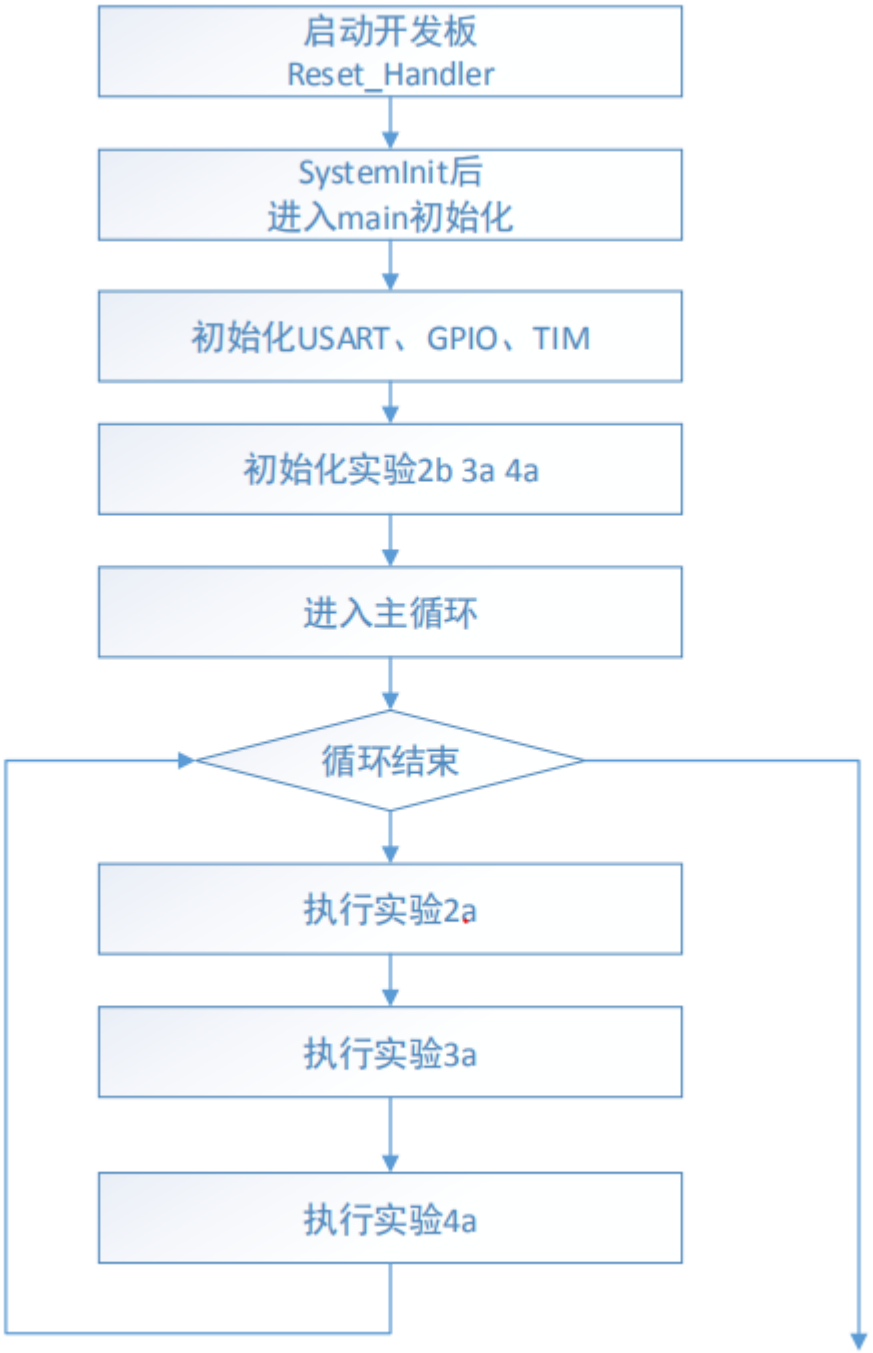
1. 下载并打开老师给的代码 emlab2020-lab4.rar

2. 连接好实验板的相关线路，打开电源，打开串口工具并打开响应串口。

3. 在 lab_main.c 中找到实验入口 lab4_a_main()，根据函数调用，理解一个个按顺序调用的函数，观察程序的执行过程和现象。

6 实验方案与实现

6.1 软件结构



6.2 源代码

- Lab4_a:

```
//led10/D10 flash
int flag=0;
static void led_flash(){
  //trace_printf("current time: %d\r\n", HAL_GetTick());s
  if(flag)
  {
    led_on(1);
    flag=0;
  }
  else
  {
    led_off(1);
    flag=1;
  }
}
```

7 实验结果与分析

基础部分的实验结果是将代码烧录到实验板后，led 1 以 1s 为 闪烁。

8 实验总结

一开始没有正确理解 task_add 的参数表中 period 的意思，period==500,是每 500ms 执行一次 led_flash 函数的意思，我们只需实现每次进入 led_flash 函数的时候，改变相应灯的状态即可。

```
void exp4_a_init(){
    //add task
    task_add(led_flash, 0, 500);
    task_add_oneshot(alarm, 1000);
}
```

而不是写得如此复杂：

```
//led10/D10 flash
unsigned int tim2=0;
int clks = 0;
static void led_flash(){
    unsigned int tim1= HAL_GetTick();

    led_off(1);
    led_off(2);
    led_off(3);

        if(tim1 -tim2 >= 1000){
            tim2=tim1;
            if(clks){
                led_off(0);
                clks = 0;
            }
            else{
                led_on(0);
                clks = 1;
            }
    }
//  # error "Not Implemented!"
}
```