第6章 串行通信模块及第一个中断程序结构

6.3 ARM Cortex-M4F中断机制及MSP432中断编程步骤

6.3 ARM Cortex-M4F中断机制及MSP432中断编程步骤

6.3.1 1）中断与异常的基本含义

- 异常（exception）是CPU强行从正常的程序运行切换到由某些内部或外部条件所要求的处理任务上去，这些任务的紧急程度是优先于CPU正在运行的任务。引起异常的外部条件通常来自外围设备、硬件断点请求、访问错误和复位等；引起异常的内部条件通常为指令、不对界错误、违反特权级和跟踪等。

6.3.1 1）中断与异常的基本含义

- 中断（interrupt）来自CPU外围设备的强行任务切换请求称为中断，软件上表现为将程序计数器（PC）指针强制转到中断服务程序入口地址运行。CPU对复位、中断、异常具有同样的处理过程，可统称为中断。

6.3.1 2）中断源、中断向量表与中断向量号

- 中断源：可以引起CPU产生中断的外部器件。
- 中断向量表：一个CPU通常可以识别多个中断源，每个中断源产生中断后，分别要运行相应的中断服务例程（Interrupt Service Routine ，ISR），这些中断服务例程ISR的起始地址（叫做中断向量地址）放在一段连续的存储区域内，这个存储区被称之为中断向量表。实际上，中断向量表是一个指针数组，内容是中断服务例程ISR的首地址。
- 中断向量号：CPU能够识别每个中断源的编号。

6.3.1 3）中断服务例程ISR

- 中断提供了一种机制，来打断当前正在运行的程序，并且保存当前CPU状态（CPU内部寄存器），转而去运行一个中断处理程序，然后恢复CPU状态，以便恢复CPU到运行中断之前的状态，同时使得中断前的程序得以继续运行。中断时打断当前正在运行的程序，而转去运行的一个中断处理程序，通常被称为中断服务例程（Interrupt Service Routine），简称ISR，通常也被称为中断处理函数。

6.3.1 4）中断优先级、可屏蔽中断和不可屏蔽中断

- 根据中断是否可以通过程序设置的方式被屏蔽，可将中断划分为可屏蔽中断和不可屏蔽中断两种。
- 可屏蔽中断是指可通过程序设置的方式决定不响应该中断，即该中断被屏蔽了。
- 不可屏蔽中断是指不能通过程序方式关闭的中断。

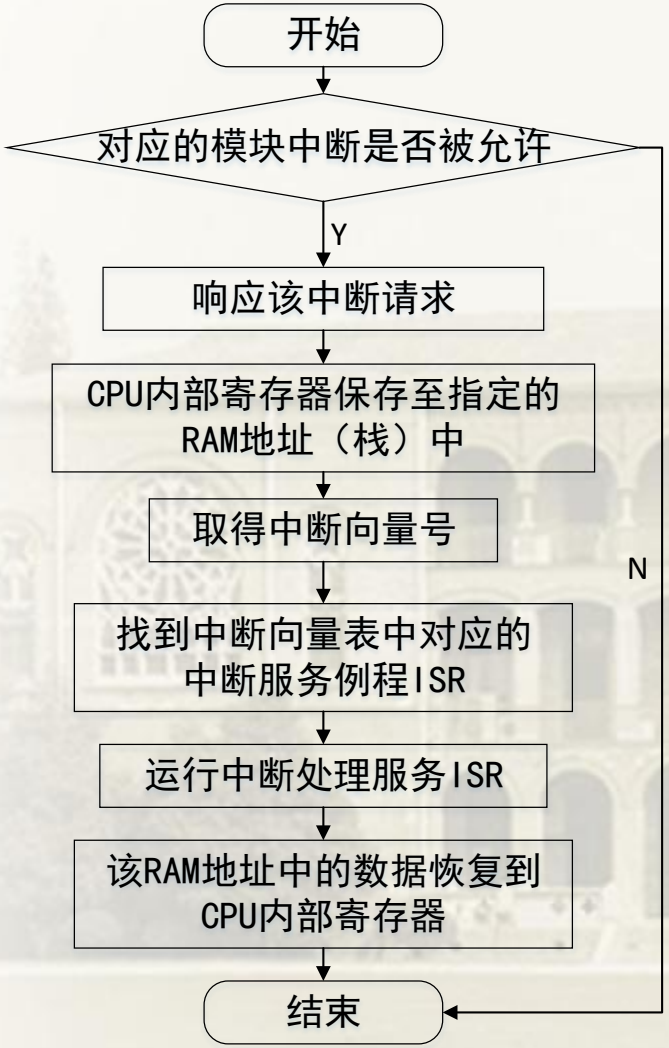
6.3.1 5）中断请求

当某一中断源需要CPU为其服务时，它将会向CPU发出中断请求信号（一种电信号）。中断控制器获取中断源硬件设备的中断向量号 ，并通过识别的中断向量号将对应硬件中断源模块的中断状态寄存器中的“中断请求位”置位，以便CPU知道何种中断请求来了。

6.3.1 6）中断采样（检测）

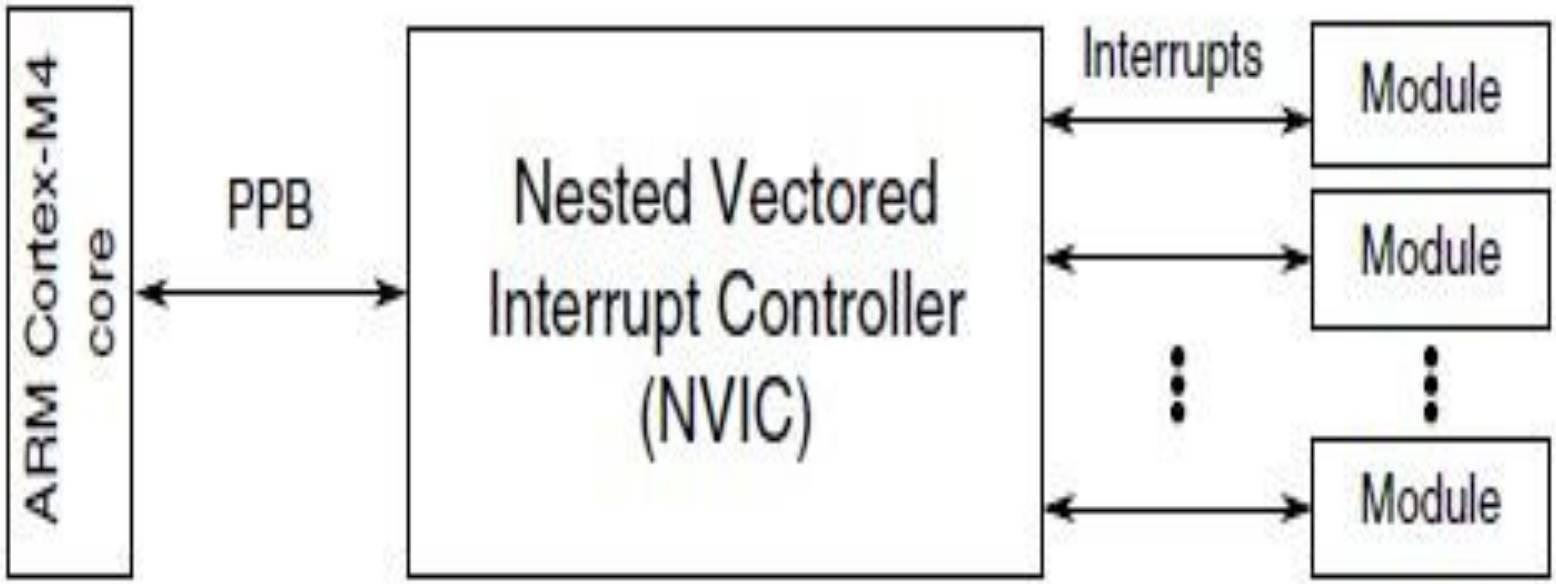
CPU在每条指令结束的时候将会检查中断请求或者系统是否满足异常条件，为此，多数CPU专门在指令周期中使用了中断周期。在中断周期中，CPU将会检测系统中是否有中断请求信号，若此时有中断请求信号，则CPU将会暂停当前运行的任务，转而去对中断事件进行响应，若系统中没有中断请求信号则继续运行当前任务

6.3.1 7）中断响应和中断处理

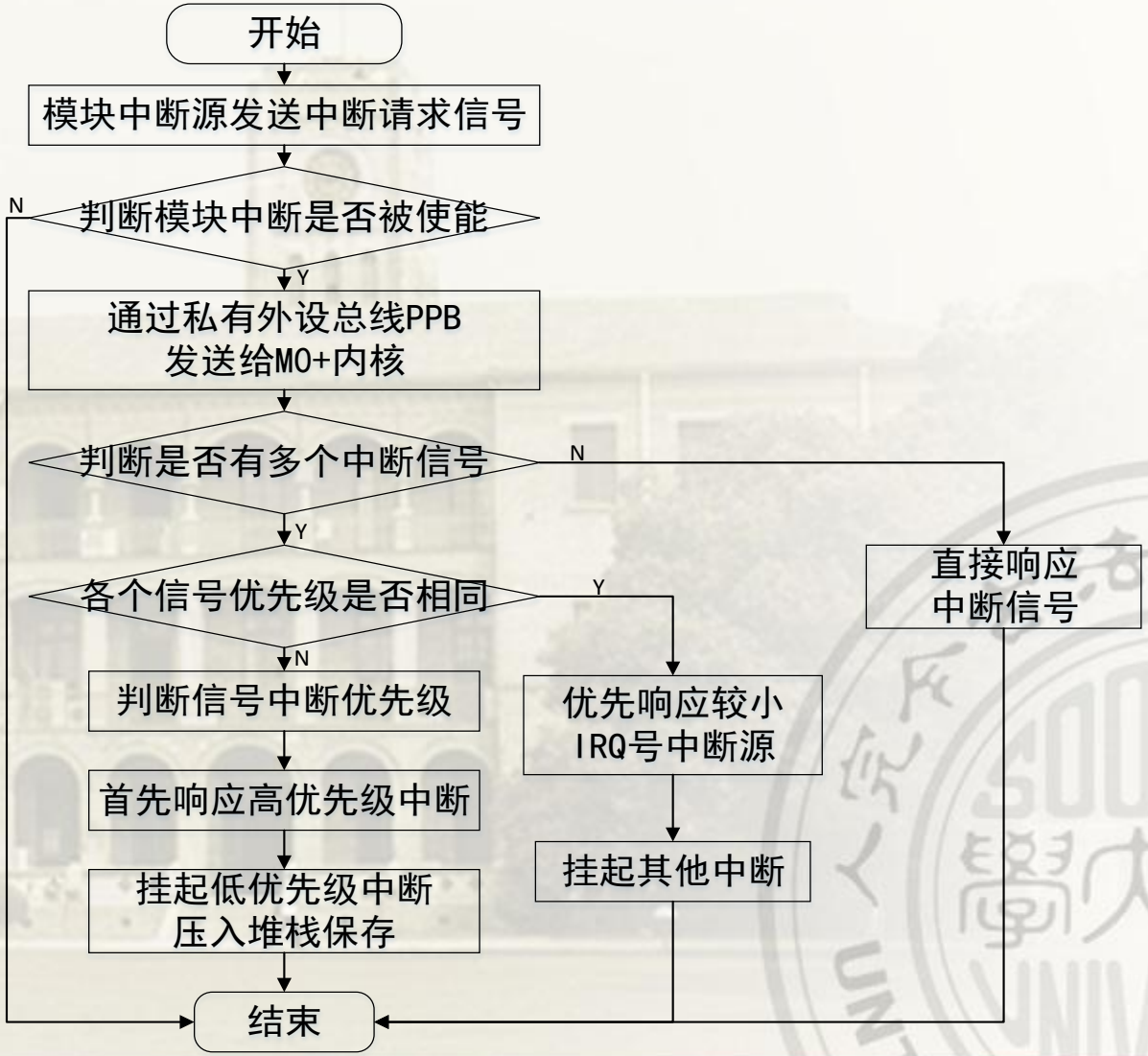


- 首先CPU会查找中断源所对应的模块中断是否被允许，若被允许，则响应该中断请求。
- 中断响应的过程要求CPU保存当前环境的“上下文（context）”于堆栈中。通过中断向量号找到中断向量表中对应的中断服务例程ISR，转而去运行中断处理服务ISR。
- 在调用ISR之前，将CPU内部寄存器保存至指定的RAM地址（栈）中，在中断结束后再将该RAM地址中的数据恢复到CPU内部寄存器中，从而使中断前后程序的“运行现场”没有任何变化。

6.3.2 1）M0+中断结构及中断过程



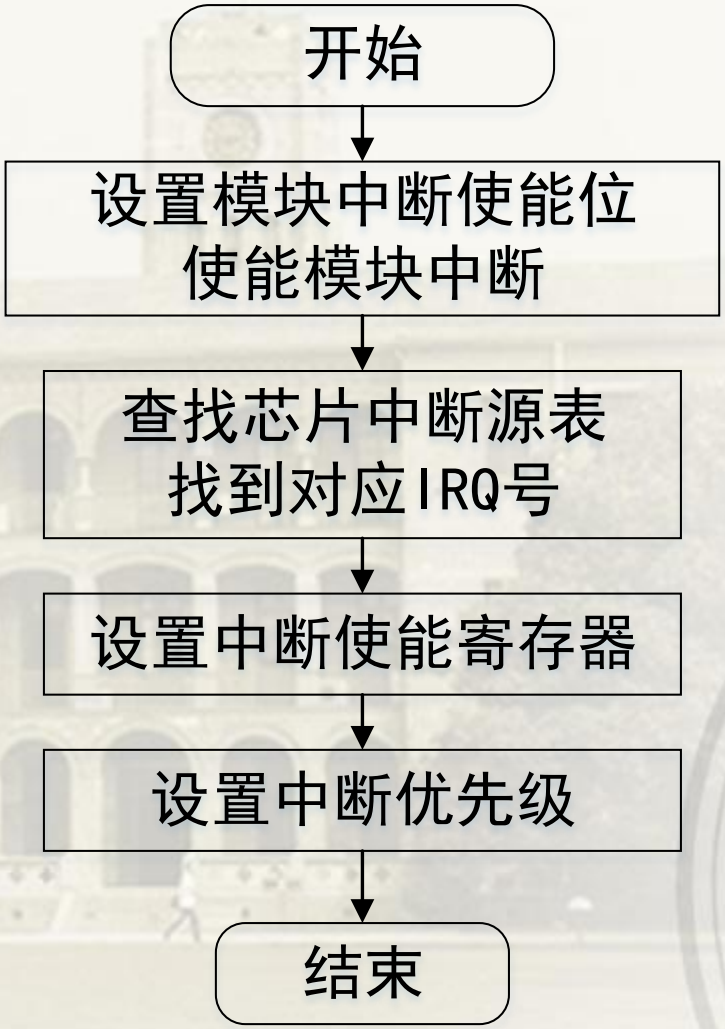
6.3.2 1）M0+中断结构及中断过程



6.3.2 2）M4F嵌套中断向量控制器NVIC内部寄存器简介

- 中断使能寄存器（NVIC_ISER）
- 中断除能寄存器（NVIC_ICER）
- 挂起/清除挂起寄存器（NVIC_ISPR/NVIC_ICPR）
- 优先级寄存器（NVIC_IPR0-NVIC_IPR15）

6.3.2 3）非内核中断初始化设置步骤



6.3.3 1）main.c文件中的工作—串口初始化、使能模块中断、开总中断

- （1）在“在初始化外设模块”位置调用uart构件中的初始化函数

```
uart_init (UART_0, 9600);     //波特率使用9600
```

- （2）在“初始化外设模块”位置调用uart构件中的使能模块中断函数

```
uart_enable_re_int(UART_0); //使能串口0接收中断
```

- （3）在“开总中断”位置调用common.h文件中的开总中断宏函数

```
ENABLE_INTERRUPTS;     //开总中断
```

6.3.3 2）在中断向量表中找到相应中断服务例程的函数名

- 在startup_msp432p401r_ccs.c文件的中断向量表中找到相应中断服务例程的函数名。
- 在中断向量表中找到串口0接收中断服务例程的函数名是EUSCIA0_IRQHandler。

6.3.3 3）进行中断功能的编程

```
void    EUSCIA0_IRQHandler (void)
{
   //串口中断服务子程序
}
```

6.3.3 4）串口中断服务例程设计

```
//=========================中断函数服务例程=========================
//串口 0 接收中断服务例程
void EUSCIA0_IRQHandler (void)
{
    uint_8 ch, flag;
    flag = 1;
    DISABLE_INTERRUPTS;      //关总中断
    ch = uart_re1(UART_0, &flag);    //调用接收一个字节的函数
    if ( flag)                       //若收到一个字节
    {
        uart_send1(UART_0, ch);      //向原串口发回一个字节
    }
    ENABLE_INTERRUPTS;    //开总中断
}
```

谢 谢！