STM32 上 RTOS 的中断管理

一. 中断管理体系

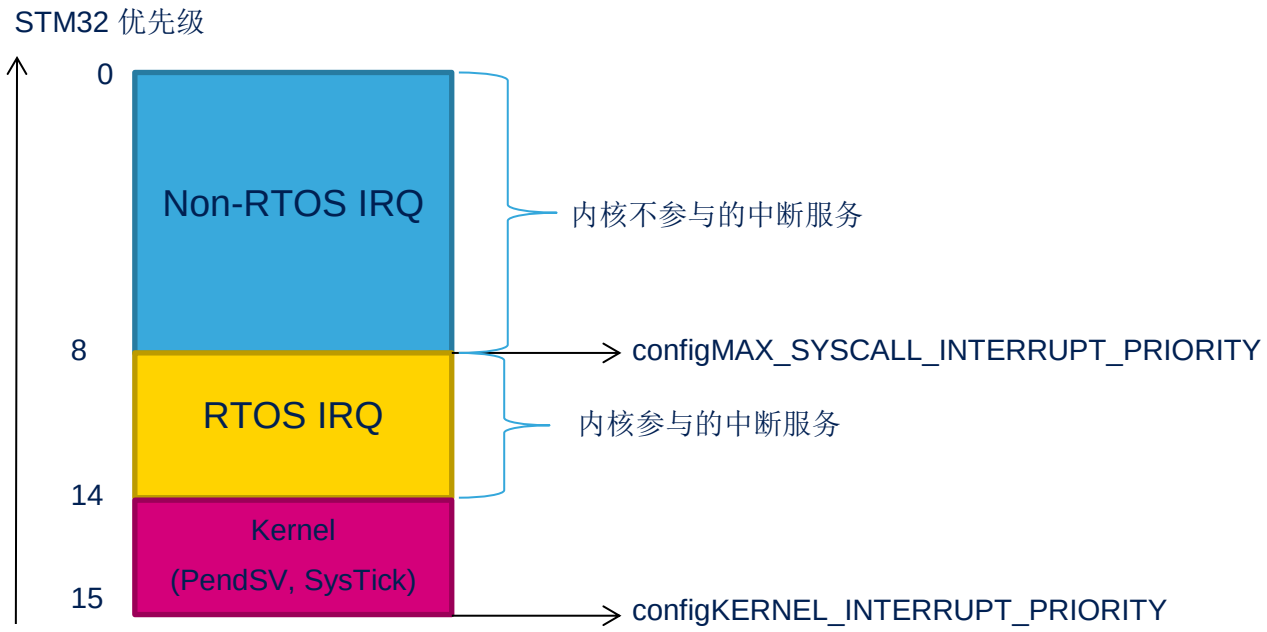

图 1 STM32 的中断服务

本文以 STM32F1，F2，F3，F4 为例（采用 Cortex-M3，M4 内核），内核支持中断嵌套（最多可设置 256 个中断优先级）。STM32 只使用其中的 16 个优先级。

如图 1 所示，RTOS 一般不会将优先级分组，但是会设置为 3 类，优先级最低的中断（级别 15）由 SysTick，PendSV 所使用；中断级别 8~14 的服务程序可以调用内核提供的进程间通信函数，但是此类中断服务程序会受到内核的影响，中断响应可能被推迟（在进入临界区后，CPU 会设置为忽略为 8~15 的优先级中断请求，但中断控制器会锁存这些请求，在告别临界区后重新打开中断便可立即产生中断请求）；级别 0~7 的中断服务程序不使用内核提供的任何函数，即内核不会影响这些中断，因此其中断延迟时间是非常短的。

二. 中断延迟的实例

1. 测试简介

初始化 GPIO PA0 为 EXTI 中断，进入临界区后按 PA0（中断源），中断不会即时响应，离开临界区后此中断会被响应。

```
/* 配置 PA0 为 EXTI 中断 */
EXTILine0_Config();

CPU_SR_Save();          // 进入临界区

for (i=0;i<200;i++)
```

```
  {
    for (j=0;j<50000;j++);
  }

  CPU_SR_Restore();       //离开临界区
```

临界区实现如下，此临界区实现屏蔽掉所有中断优先级的中断，注意在 M3/M4 内核下，可以屏蔽一定级别的中断。

```
__asm void CPU_SR_Save(void)
{
     MRS     R0, PRIMASK                ; Set prio int mask to mask all (except faults)
     CPSID   I
     BX      LR
}

__asm void CPU_SR_Restore(void)
{
     MSR     PRIMASK, R0
     BX      LR
}
```

三. FreeRTOS 下的设置

FreeRTOSConfig.h 中

```
#define configLIBRARY_LOWEST_INTERRUPT_PRIORITY                 0xf
#define configLIBRARY_MAX_SYSCALL_INTERRUPT_PRIORITY            8
```