FreeRTOS学习3

1 FreeRTOS中断管理

1.1 中断

即CPU打断正常运行的程序，转而去处理紧急的事件。

- 中断优先级：对于STM32来说，在8位寄存器中，用高4位来表示中断优先级，即最大有16级。

  数值越小，优先级越高

- 中断优先级分组：一共5种分配方式，不同模式决定了给抢占优先级和子优先级分别分配的位数

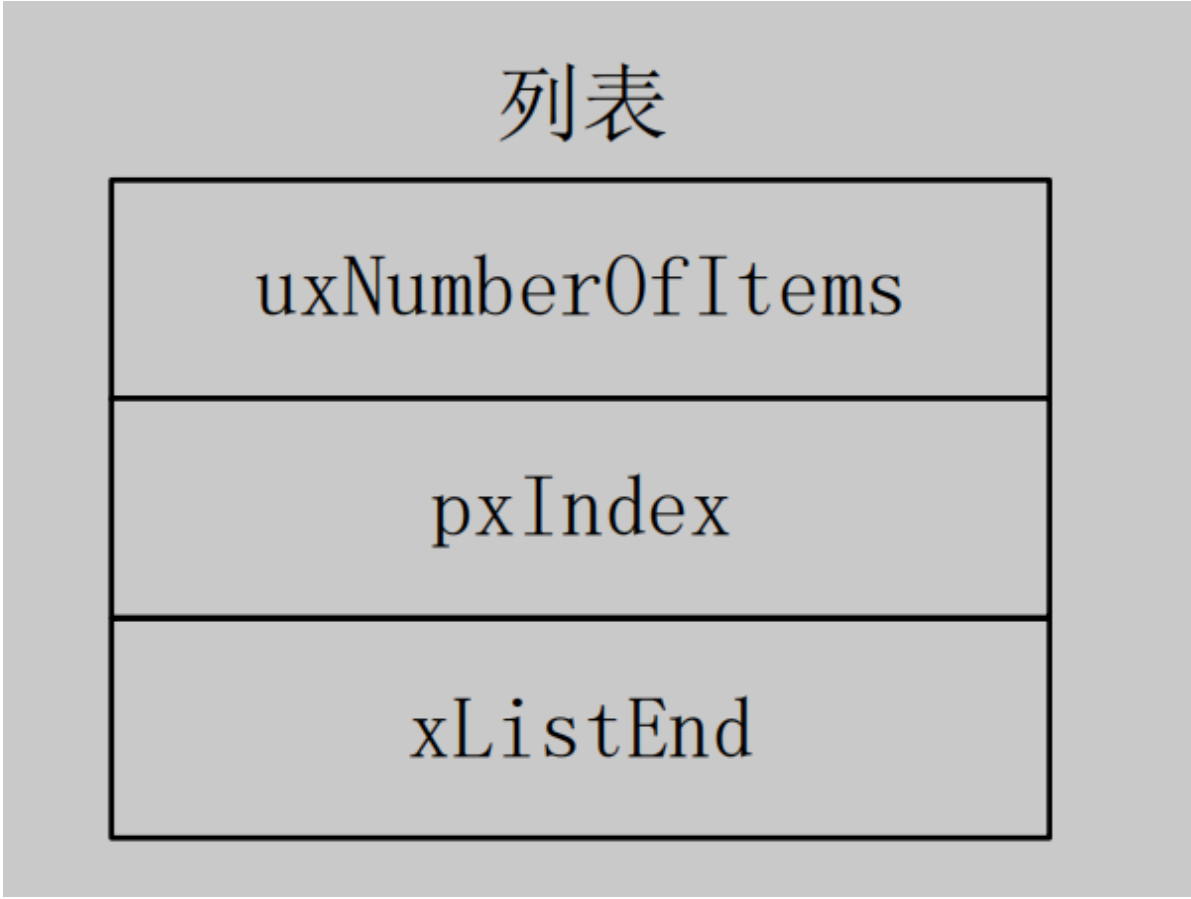
优先级分组	抢占优先级	子优先级	优先级配置寄存器高 4 位
NVIC_PriorityGroup_0	0 级抢占优先级	0-15 级子优先级	0bit 用于抢占优先级 4bit 用于子优先级
NVIC_PriorityGroup_1	0-1 级抢占优先级	0-7 级子优先级	1bit 用于抢占优先级 3bit 用于子优先级
NVIC_PriorityGroup_2	0-3 级抢占优先级	0-3 级子优先级	2bit 用于抢占优先级 2bit 用于子优先级
NVIC_PriorityGroup_3	0-7 级抢占优先级	0-1 级子优先级	3bit 用于抢占优先级 1bit 用于子优先级
NVIC_PriorityGroup_4	0-15 级抢占优先级	0 级子优先级	4bit 用于抢占优先级 0bit 用于子优先级

1.2中断管理

2 列表和列表项

2.1 列表

- 列表是FreeRTOS中的一个数据结构，概念上与链表相似

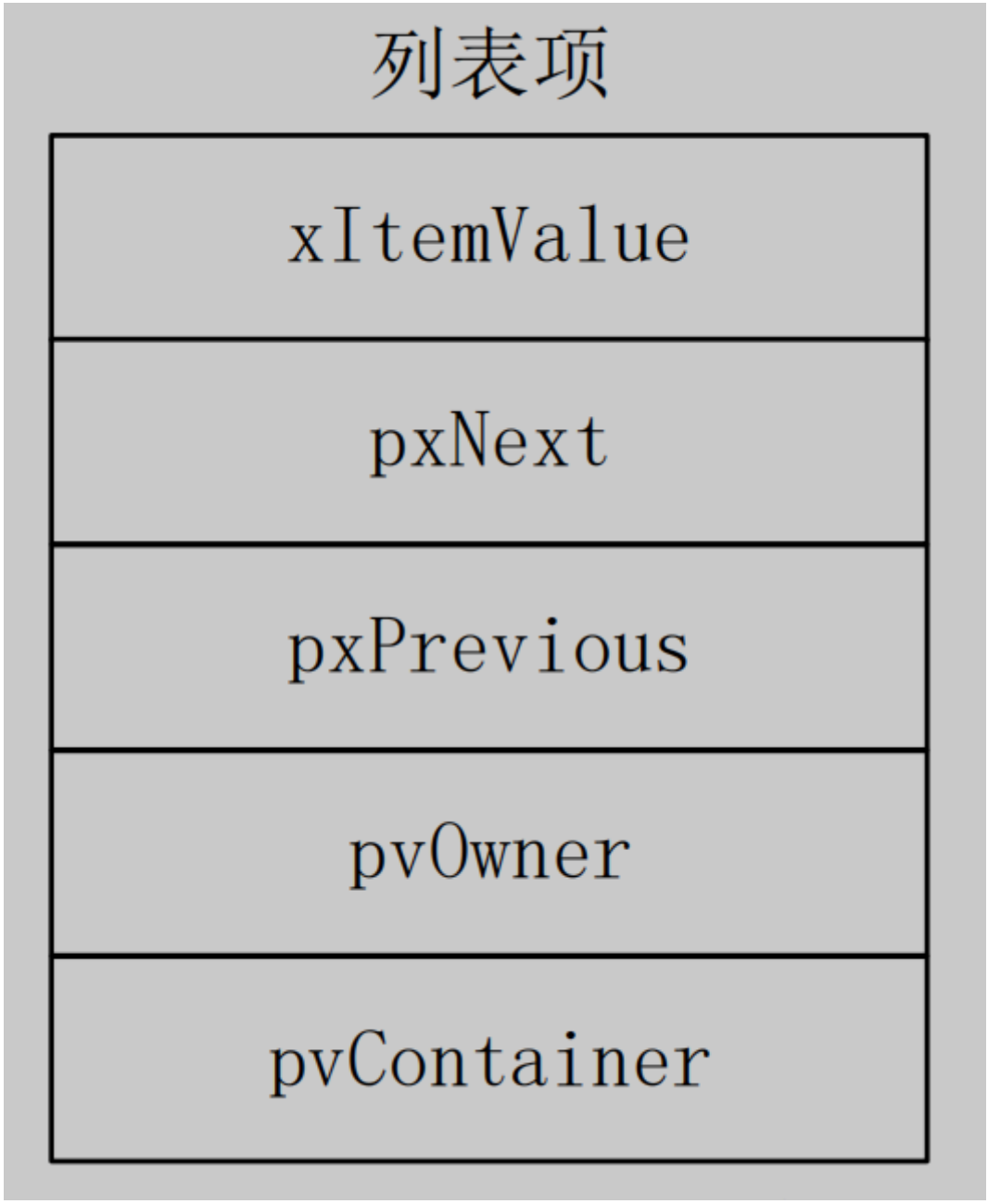


其中，**uxNumberOfItems**表示**列表中列表项的数量**；**pxIndex**表示**当前列表项索引号**，**xListEnd**属性是一个列表项，是**迷你列表项**，表示列表结束。

- **在FreeRTOS中，列表是一个双向环形链表**

2.2 列表项

- 列表项是存放在列表里的项目（相当于结点）
- 列表项间的地址不连续



xItemValue为**列表项值**，**pxNext**指向**下一个列表项**，**xPrevious**指向**前一个列表项**，**pvOwner**记录**此列表项归谁所有**，**pvContainer**记录**此列表项属于哪个列表**

2.3 列表相关API函数

- **vListInitialise()**：初始化列表
- **vListInitialiseItem()**：初始化列表项
- **vListInsertEnd()**：列表末尾插入列表项
- **vListInsert()**：列表插入列表项
- **uxListRemove()**：列表移除列表项

3 任务调度器

3.1 开启任务调度器

函数名称：vTaskStartScheduler()，用于启动任务调度器。

eg：之前的demo都是先创建一个*start_task*的开始任务，然后调用函数**vTaskStartScheduler()**

4 任务切换

本质：CPU寄存器的切换

任务A切换到任务B时：

- step1：暂停A的执行，并将任务A的寄存器保存到任务堆栈，称为保存现场
- step2：将任务B的各个寄存器值（来自于任务堆栈）恢复到CPU寄存器中，称为恢复现场