1．编码标准

FreeRTOS 的核心源代码遵从 MISRA 编码标准指南。这个标准篇幅稍长，你可以在 MISRA 官方网站花少量钱买到，这里不再复制任何标准。

FreeRTOS 源代码不符合 MISRA 标准的项目如下所示：

- 有两个 API 函数有多个返回点。MISRA 编码标准强制规定：一个函数在其结尾应该有单一的返回点。
- 指针算数运算，在创建任务时，为了兼容 8、16、20、24、32 位总线，不可避免的使用了指针算数运算。MISRA 编码标准强制规定：指针的算术运算只能用在指向数组或数组元素的指针上。
- 默认情况下，跟踪宏为空语句，因此不符合 MISRA 的规定。MISRA 编码标准强制规定：预处理指令在句法上应该是有意义的。

FreeRTOS 可以在很多不同编译器中编译，其中的一些编译器比同类有更高级特性。因为这个原因，FreeRTOS 不使用任何非 C 语言标准的特性或语法。一个例外情况是头文件 stdint.h。在文件夹 FreeRTOS/Source/include 下包含一个叫做 stdint.readme 的文件，如果你的编译器不提供 stdint 类型定义，可以将 stdint.readme 文件重命名为 stdint.h。

2 命名规则

RTOS 内核和演示例程源代码使用以下规则：

1> 变量

- uint32_t 类型的变量使用前缀 ul，这里’u’表示’unsigned’，’l’表示’long’
- uint16_t 类型的变量使用前缀 us，这里’u’表示’unsigned’，’s’表示’short’
- uint8_t 类型的变量使用前缀 uc，这里’u’表示’unsigned’，’c’表示’char’
- 非 stdint 类型的变量使用前缀 x，比如基本的 Type_t 和 TickType_t 类型，这些类型在移植层定义，定义成符合处理器架构的最高效类型；
- 非 stdint 类型的无符号变量使用前缀 ux，比如 UbaseType_t（unsigned BaseType_t）
- size_t 类型的变量使用前缀 x；
- 枚举类型变量使用前缀 e
- 指针类型变量在类型基础上附加前缀 p，比如指向 uint16_t 的指针变量前缀为 pus
- 与 MISRA 指南一致，char 类型变量仅被允许保存 ASCII 字符，前缀为 c
- 与 MISRA 指南一致，char *类型变量仅允许指向 ASCII 字符串，前缀为 pc

2> 函数

- 在文件作用域范围的函数前缀为 prv
- API 函数的前缀为它们的返回类型，当返回为空时，前缀为 v
- API 函数名字起始部分为该函数所在的文件名。比如 vTaskDelete 函数定义在 tasks.c，并且该函数返回空。

3> 宏

- 宏的名字起始部分为该宏定义所在的文件名的一部分。比如 configUSE_PREEMPTION 定义在 FreeRTOSConfig.h 文件中。
- 除了前缀，宏剩下的字母全部为大写，两个单词间用下划线（'_'）隔开。

3 数据类型

只有 stdint.h 和 RTOS 自己定义的数据类型可以使用，但也有例外情况，如下所示：

- char：与 MISRA 编码标准指南一致，char 类型变量仅被允许保存 ASCII 字符
- char *：与 MISRA 编码标准指南一致，char *类型变量仅允许指向 ASCII 字符串。当标准库函数期望一个 char *参数时，这样做可以消除一些编译器警告；特别是考虑到有些编译器将 char 类型当做 signed 类型，还有些编译器将 char 类型当做 unsigned 类型。

有三种类型会在移植层定义，它们是：

- TickType_t：如果 configUSE_16_BIT_TICKS 为非零（条件为真），TickType_t 定义为无符号 16 位类型。如果 configUSE_16_BIT_TICKS 为零（条件为假），TickType_t 定义为无符号 32 位类型。注：32 位架构的微处理器应设置 configUSE_16_BIT_TICKS 为零。
- BaseType_t：定义为微处理器架构效率最高的数据类型。比如，在 32 位架构处理器上，BaseType_t 应该定义为 32 位类型。在 16 位架构处理器上，BaseType_t 应该定义为 16 位类型。如果 BaseType_t 定义为 char，对于函数返回值一定要确保使用的是 signed char，否则可能造成负数错误。
- UbaseType_t：这是一个无符号 BaseType_t 类型

3.4 风格指南

- 缩进：缩进使用制表符，一个制表符等于 4 个空格。
- 注释：注释单行不超过 80 列，特殊情况除外。不使用 C++风格的双斜线（//）注释
- 布局：FreeRTOS 的源代码被设计成尽可能的易于查看和阅读。下面的代码片中，第一部分展示文件布局，第二部分展示 C 代码设计格式。