



从0到1教你写FreeRTOS内核，详解FreeRTOS源码如何使用。
由浅入深，基于野火STM32全系列开发板，提供完整的源代码，极具可操作性。



RTOS

FreeRTOS内核实现 与应用开发实战指南

基于STM32

刘火良 杨森 编著



机械工业出版社
China Machine Press



野火嵌入式系列



FreeRTOS内核实现与应用开发实战指南

基于STM32

刘火良 杨森 编著



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

FreeRTOS 内核实现与应用开发实战指南: 基于 STM32 / 刘火良, 杨森编著. —北京: 机械工业出版社, 2019.2

(电子与嵌入式系统设计丛书)

ISBN 978-7-111-61825-6

I. F… II. ①刘… ②杨… III. 微控制器-系统开发-指南 IV. TP332.3-62

中国版本图书馆 CIP 数据核字 (2019) 第 010054 号

FreeRTOS 内核实现与应用开发实战指南 基于 STM32

出版发行: 机械工业出版社 (北京市西城区百万庄大街 22 号 邮政编码: 100037)

责任编辑: 赵亮宇

责任校对: 李秋荣

印刷: 北京诚信伟业印刷有限公司

版次: 2019 年 3 月第 1 版第 1 次印刷

开本: 186mm × 240mm 1/16

印张: 31

书号: ISBN 978-7-111-61825-6

定价: 99.00 元

凡购本书, 如有缺页、倒页、脱页, 由本社发行部调换

客服热线: (010) 88379426 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzit@hzbook.com

版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问: 北京大成律师事务所 韩光 / 邹晓东

前言

如何学习本书

本书系统讲解 FreeRTOS，共分为两个部分。第一部分重点讲解 FreeRTOS 的原理实现，从 0 开始，不断迭代，教你把 FreeRTOS 的内核写出来，让你彻底学会任务是如何定义的、系统是如何调度的（包括底层的汇编代码讲解）、多优先级是如何实现的等操作系统最深层次的知识。当你拿到本书开始学习的时候，你一定会惊讶，原来 RTOS（Real Time Operating System，实时操作系统）的学习并没有那么复杂，反而是那么有趣，原来自己也可以写 RTOS，成就感瞬间爆棚。

当彻底掌握第一部分的知识之后，再学习其他 RTOS，可以说十分轻松。纵观现在市面上流行的几种 RTOS，它们的内核实现差异不大，只需要深入研究其中一种即可，没有必要对每一种 RTOS 都深入地研究源码。但如果时间允许，看一看也并无坏处。第二部分重点讲解 FreeRTOS 的移植、内核中每个组件的应用，比起第一部分，这部分内容掌握起来应该比较容易。

全书内容循序渐进，不断迭代，尤其在第一部分，前一章多是后一章的基础，建议从头开始阅读，不要进行跳跃式阅读。在学习时务必做到两点：一是不能一味地看书，要把代码和书本结合起来学习，一边看书，一边调试代码。如何调试代码呢？即单步执行每一条程序，看程序的执行流程和执行的效果与自己所想的是否一致。二是在每学完一章之后，必须将配套的例程重写一遍（切记不要复制，哪怕是一个分号。但可以照书录入），做到举一反三，确保真正理解。在自己写的时候肯定会错漏百出，要认真纠错，好好调试，这是你提高编程能力的最好机会。记住，编写程序不是一气呵成的，而是要一步一步地调试。

本书的编写风格

本书第一部分以 FreeRTOS V9.0.0 官方源码为蓝本，抽丝剥茧，不断迭代，教你如何从 0 开始把 FreeRTOS 内核写出来。书中涉及的数据类型、变量名称、函数名称、文件名称、

文件存放的位置都完全按照 FreeRTOS 官方的方式来实现。学完这本书之后，你可以无缝地切换到原版的 FreeRTOS 中使用。要注意的是，在实现的过程中，某些函数中会去掉一些形参和冗余的代码，只保留核心的功能，但这并不会影响学习。

本书第二部分主要介绍 FreeRTOS 的移植和内核组件的使用，不会再去深入讲解源码，而是着重讲解如何应用，如果对第一部分不感兴趣，可以跳过第一部分，直接进入第二部分的学习。

本书还有姊妹篇——《RT-Thread 内核实现与应用开发实战指南：基于 STM32》^①，两本书的编写风格、内容框架和章节命名与排序基本一致，语言阐述类似，且涉及 RTOS 抽象层的理论部分也相同，不同之处在于 RTOS 的实现原理、内核源码的讲解和上层 API 的使用，这些内容才是重点部分，是读者学习的核心。例如，虽然两本书的第一部分的章节名称基本类似，但内容不同，因为针对的 RTOS 不一样。其中，关于新建 FreeRTOS 工程和裸机系统与多任务（线程）系统的描述属于 RTOS 抽象层的理论部分，不具体针对某个 RTOS，所以基本一样。第二部分中，对于什么是任务（线程）、阻塞延时和信号量的应用等 RTOS 抽象层的理论讲解也基本类似，但是具体涉及这两个 RTOS 的原理实现和代码讲解时则完全不同。

如果读者已经学习了其中一本书，再学习另外一本的话，那么涉及 RTOS 抽象层的理论部分可跳过，只需要把精力放在 RTOS 内核的实现和源码 API 的应用方面。因为现有的 RTOS 在理论层基本都是相通的，但在具体的代码实现上各有特点，所以可以用这两本书进行互补学习，掌握了其中一本书的知识，再学习另外一本书定会得心应手，事半功倍。

本书的参考资料及配套硬件

关于本书的参考资料和配套硬件的信息，请参考本书附录部分。

本书的技术论坛

如果在学习过程中遇到问题，可以到野火电子论坛 www.firebbs.cn 发帖交流，开源共享，共同进步。

鉴于作者水平有限，书中难免有错漏之处，热心的读者也可把勘误发送到论坛上以便改进。祝你学习愉快，FreeRTOS 的世界，野火与你同行。

^① 此书由机械工业出版社出版，书号为 978-7-111-61366-4。——编辑注

引言

为什么要学习 RTOS

当我们进入嵌入式这个领域时，首先接触的往往是单片机编程，单片机编程又首选 51 单片机来入门。这里面说的单片机编程通常都是指裸机编程，即不加入任何 RTOS 的编程。常用的 RTOS 有国外的 FreeRTOS、 $\mu\text{C}/\text{OS}$ 、RTX 和国内的 FreeRTOS、Huawei LiteOS 和 AliOS-Things 等，其中，开源且免费的 FreeRTOS 的市场占有率最高。

在裸机系统中，所有的程序基本都是自己写的，所有的操作都是在一个无限的大循环中实现。现实生活中的很多中小型电子产品中用的都是裸机系统，而且能够满足需求。但是为什么还要学习 RTOS 编程，要涉及一个操作系统呢？一是因为项目需求，随着产品要实现的功能越来越多，单纯的裸机系统已经不能完美地解决问题，反而会使编程变得更加复杂，如果想降低编程的难度，可以考虑引入 RTOS 实现多任务管理，这是使用 RTOS 的最大优势；二是出于学习的需要，必须学习更高级的技术，实现更好的职业规划，为将来能有更好的职业发展做准备，而不是一味拘泥于裸机编程。作为一个合格的嵌入式软件工程师，学习是永远不能停歇的，时刻都得为将来做准备。书到用时方恨少，希望当机会来临时，你不要有这种感觉。

为了帮大家厘清 RTOS 编程的思路，本书会在第 3 章简单地分析这两种编程方式的区别，我们将这个区别称为“学习 RTOS 的命门”，只要掌握这一关键内容，以后的 RTOS 学习可以说是易如反掌。在讲解这两种编程方式的区别时，我们主要讲解方法论，不会涉及具体的代码，即主要还是通过伪代码来讲解。

如何学习 RTOS

裸机编程和 RTOS 编程的风格有些不一样，而且有很多人说学习 RTOS 很难，这就导致想要学习的人一听到 RTOS 编程就在心里忌惮三分，结果就是“出师未捷身先死”。

那么到底如何学习 RTOS 呢？最简单的方法就是在别人移植好的系统上，先看看 RTOS

中 API 的使用说明，然后调用这些 API 实现自己想要的功能，完全不用关心底层的移植，这是最简单、快速的入门方法。这种方法有利有弊。如果是做产品，好处是可以快速地实现功能，将产品推向市场，赢得先机；弊端是当程序出现问题时，因对 RTOS 不够了解，会导致调试困难。如果想系统地学习 RTOS，那么只会简单地调用 API 是不可取的，我们应该深入学习其中一款 RTOS。

目前市场上的 RTOS，其内核实现方式差异不大，我们只需要深入学习其中一款即可。万变不离其宗，只要掌握了一款 RTOS，以后换到其他型号的 RTOS，使用起来自然也是得心应手。那么如何深入地学习一款 RTOS 呢？这里有一个非常有效但也十分难的方法，就是阅读 RTOS 的源码，深入研究内核和每个组件的实现方式。这个过程枯燥且痛苦。但为了能够学到 RTOS 的精华，还是很值得一试的。

市面上虽然有一些讲解相关 RTOS 源码的图书，但如果基础知识掌握得不够，且先前没有使用过该款 RTOS，那么只看源码还是会非常枯燥，并且不能从全局掌握整个 RTOS 的构成和实现。

现在，我们采用一种全新的方法来教大家学习一款 RTOS，既不是单纯地介绍其中的 API 如何使用，也不是单纯地拿里面的源码一句句地讲解，而是从 0 开始，层层叠加，不断完善，教大家如何把一个 RTOS 从 0 到 1 写出来，让你在每一个阶段都能享受到成功的喜悦。在这个 RTOS 实现的过程中，只需要具备 C 语言基础即可，然后就是跟着本书笃定前行，最后定有所成。

选择什么 RTOS

用来教学的 RTOS，我们不会完全从头写一个，而是选取目前国内外市场占有率很高的 FreeRTOS 为蓝本，将其抽丝剥茧，从 0 到 1 写出来。在实现的过程中，数据类型、变量名、函数名称、文件类型等都完全按照 FreeRTOS 里面的写法，不会再重新命名。这样学完本书之后，就可以无缝地过渡到 FreeRTOS 了。

目 录

前 言 引 言

第一部分 从 0 到 1 教你写 FreeRTOS 内核

第 1 章 初识 FreeRTOS 2

1.1 FreeRTOS 版权 2

1.2 FreeRTOS 收费问题 2

1.2.1 FreeRTOS 2

1.2.2 OpenRTOS 2

1.2.3 SaveRTOS 3

1.3 FreeRTOS 资料获取 3

1.3.1 获取源码 3

1.3.2 获取书籍 4

1.3.3 快速入门 4

1.4 FreeRTOS 的编程风格 5

1.4.1 数据类型 5

1.4.2 变量名 6

1.4.3 函数名 6

1.4.4 宏 7

1.4.5 格式 7

第 2 章 新建 FreeRTOS 工程—— 软件仿真 8

2.1 新建本地工程文件夹 8

2.2 使用 KEIL 新建工程 8

2.2.1 New Project 9

2.2.2 Select Device for Target 9

2.2.3 Manage Run-Time

Environment 10

2.3 在 KEIL 工程中新建文件组 11

2.4 在 KEIL 工程中添加文件 11

2.5 调试配置 13

2.5.1 设置软件仿真 13

2.5.2 修改时钟大小 13

2.5.3 添加头文件路径 13

第 3 章 裸机系统与多任务系统 15

3.1 裸机系统 15

3.1.1 轮询系统 15

3.1.2 前后台系统 16

3.2 多任务系统 17

第 4 章 数据结构——列表与列表项 20

4.1 C 语言链表 20

4.1.1 单向链表 20

4.1.2 双向链表 22

4.1.3 链表与数组的对比 22

4.2 FreeRTOS 中链表的实现 23

4.2.1 实现链表节点 23

4.2.2 实现链表根节点 25

4.3 链表节点插入实验	31	6.6 临界段代码的应用	70
4.4 实验现象	34	6.7 实验现象	71
第 5 章 任务的定义与任务切换	35	第 7 章 空闲任务与阻塞延时	72
5.1 本章目标	35	7.1 实现空闲任务	72
5.2 什么是任务	36	7.1.1 定义空闲任务的栈	72
5.3 创建任务	37	7.1.2 定义空闲任务的任 务控制块	73
5.3.1 定义任务栈	37	7.1.3 创建空闲任务	73
5.3.2 定义任务函数	38	7.2 实现阻塞延时	74
5.3.3 定义任务控制块	39	7.2.1 vTaskDelay() 函数	74
5.3.4 实现任务创建函数	40	7.2.2 修改 vTaskSwitchContext() 函数	75
5.4 实现就绪列表	45	7.3 SysTick 中断服务函数	77
5.4.1 定义就绪列表	45	7.4 SysTick 初始化函数	78
5.4.2 就绪列表初始化	45	7.5 main() 函数	80
5.4.3 将任务插入就绪列表	46	7.6 实验现象	83
5.5 实现调度器	49	第 8 章 多优先级	84
5.5.1 启动调度器	49	8.1 支持多优先级的方法	84
5.5.2 任务切换	54	8.2 查找最高优先级的就绪任务相关 代码	85
5.6 main() 函数	58	8.2.1 通用方法	87
5.7 实验现象	61	8.2.2 优化方法	87
5.8 本章涉及的汇编指令	64	8.3 修改代码以支持多优先级	89
第 6 章 临界段的保护	65	8.3.1 修改任务控制块	89
6.1 什么是临界段	65	8.3.2 修改 xTaskCreateStatic() 函数	89
6.2 Cortex-M 内核快速关中断指令	65	8.3.3 修改 vTaskStartScheduler() 函数	93
6.3 关中断	66	8.3.4 修改 vTaskDelay() 函数	94
6.3.1 不带返回值的关中断 函数	66	8.3.5 修改 vTaskSwitchContext() 函数	95
6.3.2 带返回值的关中断函数	67		
6.4 开中断	67		
6.5 进入 / 退出临界段的宏	68		
6.5.1 进入临界段	68		
6.5.2 退出临界段	69		

8.3.6 修改 xTaskIncrementTick() 函数	96
8.4 main() 函数	97
8.5 实验现象	100

第 9 章 任务延时列表 102

9.1 任务延时列表的工作原理	102
9.2 实现任务延时列表	103
9.2.1 定义任务延时列表	103
9.2.2 任务延时列表初始化	103
9.2.3 定义 xNextTaskUnblock- Time	103
9.2.4 初始化 xNextTaskUnblock- Time	104
9.3 修改代码以支持任务延时列表	104
9.3.1 修改 vTaskDelay() 函数	105
9.3.2 修改 xTaskIncrementTick() 函数	107
9.3.3 修改 taskRESET_READY_ PRIORITY() 函数	109
9.4 main() 函数	110
9.5 实验现象	110

第 10 章 时间片 111

10.1 时间片测试实验	111
10.2 main.c 文件	112
10.3 实验现象	115
10.4 原理分析	116
10.4.1 taskSELECT_HIGHEST_ PRIORITY_TASK() 函数	116
10.4.2 taskRESET_READY_ PRIORITY() 函数	117

10.5 修改代码以支持优先级	118
10.5.1 修改 xPortSysTick- Handler() 函数	118
10.5.2 修改 xTaskIncrement- Tick() 函数	119

第二部分 FreeRTOS 内核应用开发

第 11 章 移植 FreeRTOS 到 STM32 ... 124

11.1 获取 STM32 的裸机工程模板	124
11.2 下载 FreeRTOS V9.0.0 源码	124
11.3 FreeRTOS 文件夹内容	126
11.3.1 FreeRTOS 文件夹	126
11.3.2 FreeRTOS-Plus 文件夹	128
11.3.3 HTML 文件	129
11.4 向裸机工程中添加 FreeRTOS 源码	129
11.4.1 提取 FreeRTOS 最简 源码	129
11.4.2 复制 FreeRTOS 到裸机 工程根目录	130
11.4.3 复制 FreeRTOSConfig.h 文件到 User 文件夹	131
11.4.4 添加 FreeRTOS 源码到 工程组文件夹	131
11.5 修改 FreeRTOSConfig.h 文件	133
11.5.1 FreeRTOSConfig.h 文件 内容	133
11.5.2 修改 FreeRTOSConfig.h 文件	143
11.6 修改 stm32f10x_it.c 文件	147
11.7 修改 main.c 文件	151
11.8 下载验证	152

第 12 章 任务	153	13.3 两种方法的适用情况	179
12.1 硬件初始化	153	13.4 FreeRTOS 的启动流程	179
12.2 创建单任务——SRAM 静态内存	155	13.4.1 创建任务函数 xTaskCreate()	179
12.2.1 定义任务函数	155	13.4.2 开启调度器函数 vTaskStartScheduler()	181
12.2.2 空闲任务与定时器任务 栈函数实现	155	13.4.3 main() 函数	185
12.2.3 定义任务栈	157	第 14 章 任务管理	188
12.2.4 定义任务控制块	157	14.1 任务的基本概念	188
12.2.5 静态创建任务	158	14.2 任务调度器的基本概念	188
12.2.6 启动任务	159	14.3 任务状态的概念	189
12.2.7 main.c 文件	159	14.4 任务状态迁移	190
12.3 下载验证 SRAM 静态内存单任务	164	14.5 常用的任务函数	191
12.4 创建单任务——SRAM 动态内存	164	14.5.1 任务挂起函数	191
12.4.1 动态内存空间堆的来源	164	14.5.2 任务恢复函数	195
12.4.2 定义任务函数	165	14.5.3 任务删除函数	203
12.4.3 定义任务栈	166	14.5.4 任务延时函数	207
12.4.4 定义任务控制块指针	166	14.6 任务的设计要点	215
12.4.5 动态创建任务	166	14.7 任务管理实验	216
12.4.6 启动任务	167	14.8 实验现象	221
12.4.7 main.c 文件	167	第 15 章 消息队列	222
12.5 下载验证 SRAM 动态内存单任务	171	15.1 消息队列的基本概念	222
12.6 创建多任务——SRAM 动态内存	171	15.2 消息队列的运作机制	222
12.7 下载验证 SRAM 动态内存多任务	175	15.3 消息队列的阻塞机制	223
第 13 章 FreeRTOS 的启动流程	176	15.4 消息队列的应用场景	224
13.1 “万事俱备，只欠东风”法	176	15.5 消息队列控制块	224
13.2 “小心翼翼，十分谨慎”法	177	15.6 常用的消息队列函数	226
		15.6.1 消息队列动态创建 函数	226
		15.6.2 消息队列静态创建函数	232
		15.6.3 消息队列删除函数	233

15.6.4	向消息队列发送消息函数	234	17.3	互斥量的应用场景	287
15.6.5	从消息队列读取消息函数	244	17.4	互斥量的运作机制	287
15.7	消息队列注意事项	251	17.5	互斥量控制块	288
15.8	消息队列实验	252	17.6	互斥量函数	289
15.9	实验现象	256	17.6.1	互斥量创建函数 xSemaphoreCreateMutex()	289
第 16 章	信号量	258	17.6.2	递归互斥量创建函数 xSemaphoreCreateRecursiveMutex()	292
16.1	信号量的基本概念	258	17.6.3	互斥量删除函数 vSemaphoreDelete()	293
16.1.1	二值信号量	258	17.6.4	互斥量获取函数 xSemaphoreTake()	293
16.1.2	计数信号量	259	17.6.5	递归互斥量获取函数 xSemaphoreTakeRecursive()	299
16.1.3	互斥信号量	259	17.6.6	互斥量释放函数 xSemaphoreGive()	301
16.1.4	递归信号量	259	17.6.7	递归互斥量释放函数 xSemaphoreGiveRecursive()	304
16.2	二值信号量的应用场景	260	17.7	互斥量实验	307
16.3	二值信号量的运作机制	260	17.7.1	模拟优先级翻转实验	307
16.4	计数信号量的运作机制	261	17.7.2	互斥量降低优先级翻转危害实验	312
16.5	信号量控制块	262	17.8	实验现象	318
16.6	常用的信号量函数	263	17.8.1	模拟优先级翻转实验现象	318
16.6.1	信号量创建函数	263	17.8.2	互斥量降低优先级翻转危害实验现象	318
16.6.2	信号量删除函数	268	第 18 章	事件	320
16.6.3	信号量释放函数	268	18.1	事件的基本概念	320
16.6.4	信号量获取函数	271			
16.7	信号量实验	273			
16.7.1	二值信号量同步实验	273			
16.7.2	计数信号量实验	277			
16.8	实验现象	282			
16.8.1	二值信号量实验现象	282			
16.8.2	计数信号量实验现象	283			
第 17 章	互斥量	284			
17.1	互斥量的基本概念	284			
17.2	互斥量的优先级继承机制	284			

18.2	事件的应用场景	321	19.6.4	软件定时器任务	358
18.3	事件的运作机制	321	19.6.5	软件定时器删除函数	365
18.4	事件控制块	323	19.7	软件定时器实验	366
18.5	事件函数	323	19.8	实验现象	371
18.5.1	事件创建函数 xEvent-GroupCreate()	323	第 20 章	任务通知	372
18.5.2	事件删除函数 vEvent-GroupDelete()	325	20.1	任务通知的基本概念	372
18.5.3	事件组置位函数 xEvent-GroupSetBits() (任务)	326	20.2	任务通知的运作机制	372
18.5.4	事件组置位函数 xEvent-GroupSetBitsFromISR() (中断)	330	20.3	任务通知的数据结构	373
18.5.5	等待事件函数 xEvent-GroupWaitBits()	332	20.4	任务通知函数	374
18.5.6	清除事件组指定位函数 xEventGroupClearBits() 与 xEventGroupClearBitsFromISR()	337	20.4.1	发送任务通知函数	374
18.6	事件实验	338	20.4.2	获取任务通知函数	391
18.7	实验现象	343	20.5	任务通知实验	398
第 19 章	软件定时器	344	20.5.1	任务通知代替消息队列	398
19.1	软件定时器的基本概念	344	20.5.2	任务通知代替二值信号量	404
19.2	软件定时器的应用场景	345	20.5.3	任务通知代替计数信号量	409
19.3	软件定时器的精度	345	20.5.4	任务通知代替事件组	414
19.4	软件定时器的运作机制	346	20.6	实验现象	419
19.5	软件定时器控制块	348	20.6.1	任务通知代替消息队列实验现象	419
19.6	软件定时器函数	349	20.6.2	任务通知代替二值信号量实验现象	420
19.6.1	软件定时器创建函数	349	20.6.3	任务通知代替计数信号量实验现象	420
19.6.2	软件定时器启动函数	352	20.6.4	任务通知代替事件组实验现象	421
19.6.3	软件定时器停止函数	356	第 21 章	内存管理	422
			21.1	内存管理的基本概念	422
			21.2	内存管理的应用场景	423

21.3 内存管理方案详解	424	22.3 中断延迟的概念	459
21.3.1 heap_1.c	424	22.4 中断管理的应用场景	460
21.3.2 heap_2.c	428	22.5 ARM Cortex-M 的中断管理	460
21.3.3 heap_3.c	436	22.6 中断管理实验	462
21.3.4 heap_4.c	438	22.7 实验现象	470
21.3.5 heap_5.c	448		
21.4 内存管理实验	451	第 23 章 CPU 利用率统计	471
21.5 实验现象	455	23.1 CPU 利用率的基本概念	471
第 22 章 中断管理	456	23.2 CPU 利用率的作用	471
22.1 异常与中断的基本概念	456	23.3 CPU 利用率统计	472
22.1.1 中断的介绍	457	23.4 CPU 利用率统计实验	473
22.1.2 和中断相关的术语	457	23.5 实验现象	478
22.2 中断管理的运作机制	458	附录	479

本部分以 FreeRTOS Nano 为蓝本，并不打算：取而代之，教大家如何从零开始写 FreeRTOS 与出来。这一部分着重讲解 FreeRTOS 实现的过程，当你学完这部分之后，再来重新使用 FreeRTOS 或者其他 RTOS，相信得心应手，不仅知其然，而且知其所以然。在源码实现的过程中，提及的数据类型、变量名称、函数名称、文件名等以及文件的存放目录等会完全按照 FreeRTOS 的来写呢，一些不必要的注释则会删除，但并不影响我们理解整个程序实现的原理。

本部分几乎每一章都是有一章的基础，环环相扣，逐渐揭开 FreeRTOS 的神秘面纱，读起来会有一种豁然开朗的感觉。如果把代码看做一餐，你真正得出的效果与书中给出的一样，亲从心里油然而生的成就感确实就要爆棚，恨不得一下子把本书读完，真是让人看了还想看，看了还想看。

第一部分

从 0 到 1 教你写 FreeRTOS 内核

本部分以 FreeRTOS Nano 为蓝本，抽丝剥茧，不断迭代，教大家如何从 0 开始把 FreeRTOS 写出来。这一部分着重讲解 FreeRTOS 实现的过程，当你学完这部分之后，再来重新使用 FreeRTOS 或者其他 RTOS，将会得心应手，不仅知其然，而且知其所以然。在源码实现的过程中，涉及的数据类型、变量名称、函数名称、文件名称以及文件的存放目录都会完全按照 FreeRTOS 的来实现，一些不必要的代码将会剔除，但并不会影响我们理解整个操作系统的功能。

本部分几乎每一章都是前一章的基础，环环相扣，逐渐揭开 FreeRTOS 的神秘面纱，读起来会有一种豁然开朗的感觉。如果把代码都敲一遍，仿真时得出的效果与书中给出的一样，那从心里油然而生的成就感简直就要爆棚，恨不得一下子把本书读完，真是让人看了还想看，读了还想读。

第 1 章

初识 FreeRTOS

1.1 FreeRTOS 版权

FreeRTOS 由美国的 Richard Barry 于 2003 年发布，Richard Barry 是 FreeRTOS 的拥有者和维护者，在过去的十多年中 FreeRTOS 历经了 9 个版本，与众多半导体厂商合作密切，有数百万开发者，是目前市场占有率最高的 RTOS。

FreeRTOS 于 2018 年被亚马逊收购，改名为 AWS FreeRTOS，版本号升级为 V10，且开源协议也由原来的 GPLv2+ 修改为 MIT，与 GPLv2+ 相比，MIT 更加开放，你完全可以理解为完全免费。V9 以前的版本还是维持原样，V10 版本相比于 V9 就是加入了一些物联网相关的组件，内核基本不变。亚马逊收购 FreeRTOS 也是为了进军物联网和人工智能领域。本书还是以 V9 版本来讲解。

1.2 FreeRTOS 收费问题

1.2.1 FreeRTOS

FreeRTOS 是一款“开源免费”的实时操作系统，遵循的是 GPLv2+ 的许可协议。这里说到的开源，指的是可以免费获取 FreeRTOS 的源代码，且当你的产品使用了 FreeRTOS 而没有修改 FreeRTOS 内核源码时，你的产品的全部代码都可以闭源，不用开源，但是当你修改了 FreeRTOS 内核源码时，就必须将修改的这部分开源，反馈给社区，其他应用部分不用开源。免费的意思是无论你是个人还是公司，都可以免费地使用，不需要花费一分钱。

1.2.2 OpenRTOS

FreeRTOS 和 OpenRTOS 拥有的代码是一样的，但是可从官方获取的服务却是不一样的。FreeRTOS 号称免费，OpenRTOS 号称收费，它们的具体区别如表 1-1 所示。

表 1-1 FreeRTOS 开源授权与 OpenRTOS 商业授权的区别

比较的项目	FreeRTOS	OpenRTOS
是否免费	是	否
可否商业使用	是	是
是否需要版权费	否	否
是否提供技术支持	否	是
是否被法律保护	否	是
是否需要开源工程代码	否	否
是否需要开源修改的内核源码	是	否
是否需要声明产品使用了 FreeRTOS	如果发布源码,则需要声明	否
是否需要提供 FreeRTOS 的整个工程代码	如果发布源码,则需要提供	否

1.2.3 SaveRTOS

SaveRTOS 也基于 FreeRTOS,但是 SaveRTOS 为某些特定的领域做了安全相关的设计,有关 SaveRTOS 获得的安全验证具体如表 1-2 所示。SaveRTOS 需要收费。

表 1-2 SaveRTOS 获得的安全方面的验证

行业类目	验证编号
工控	IEC 61508
铁路	EN 50128
医疗	IEC 62304/FDA 510K
核工业	IEC 61513、IEC62138、ASME NQA-1
汽车电子	ISO 26262
加工业	IEC 61511
航空	DO178B

1.3 FreeRTOS 资料获取

FreeRTOS 的源码和相应的官方书籍均可从官网 www.freertos.org 获得,如图 1-1 所示为官网首页。

1.3.1 获取源码

单击图 1-1 中的 Download Source 按钮,可以下载 FreeRTOS 最新版本的源码。如果想下载以往版本,可从托管网址 <https://sourceforge.net/projects/freertos/files/FreeRTOS/> 下载。截至本章编写时,FreeRTOS 已经更新到 V10.0.1,具体如图 1-2 所示。