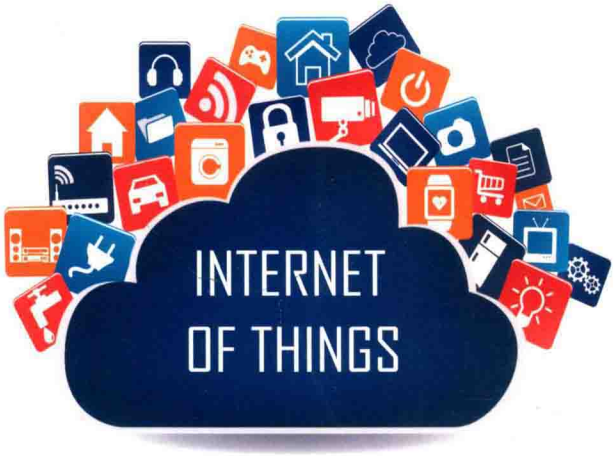
HZ BOOKS
华章IT

自研开源嵌入式实时操作系统RT-Thread核心作者撰写，专业性毋庸置疑
系统剖析嵌入式系统核心设计与实现，掌握物联网操作系统精髓

电子与嵌入式系统
设计丛书

RT-Thread



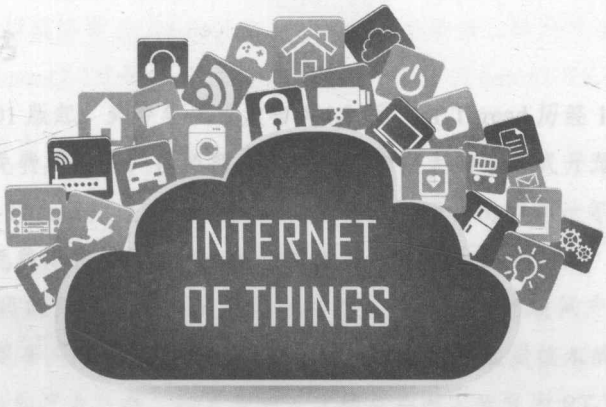
嵌入式实时操作系统

RT-Thread设计与实现

邱祎 熊谱翔 朱天龙 著

机械工业出版社
China Machine Press

电子与嵌入式系统
设计丛书



嵌入式实时操作系统

RT-Thread设计与实现

邱祎 熊谱翔 朱天龙 著

机械工业出版社
China Machine Press

图书在版编目（CIP）数据

嵌入式实时操作系统：RT-Thread 设计与实现 / 邱祎，熊谱翔，朱天龙著 . —北京：机械工业出版社，2019.3
（电子与嵌入式系统设计丛书）

ISBN 978-7-111-61934-5

I. 嵌… II. ①邱… ②熊… ③朱… III. 微控制器 - 系统开发 IV. TP332.3

中国版本图书馆 CIP 数据核字（2019）第 025553 号

嵌入式实时操作系统：RT-Thread 设计与实现

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：郎亚妹　　责任校对：李秋荣
印　　刷：北京市荣盛彩色印刷有限公司　　版　　次：2019 年 3 月第 1 版第 1 次印刷
开　　本：186mm × 240mm　1/16　　印　　张：21.75
书　　号：ISBN 978-7-111-61934-5　　定　　价：89.00 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

前　言

为什么要写这本书

自 2006 年发布 V0.01 版起，到今年正式发布 V4.0 版，RT-Thread 历经 12 年的累积发展，凭借良好的口碑和开源免费的策略，已经拥有了一个国内最大的嵌入式开源社区，积聚了数十万的软件爱好者。RT-Thread 广泛应用于能源、车载、医疗、消费电子等众多行业，已成为国人自主开发、最成熟稳定和装机量最大的开源嵌入式操作系统。

深处于行业之中，我们深刻地感受到近年来国内芯片产业和物联网产业快速崛起的趋势，行业发展迫切需要更多人才，尤其是掌握嵌入式操作系统等底层技术的人才，我们希望通过本书让 RT-Thread 触达更多人群，让更多的人了解集聚国人智慧的 RT-Thread 操作系统，从而让 RT-Thread 赋能更多行业，真正做到“积识成睿，慧泽百川”。

另外，高校学生是 RT-Thread 非常重视的群体，从 2018 年起，RT-Thread 启动了一系列大学生计划，包括送书计划、培训计划、合作开课、赞助竞赛等，以帮助学生了解和学习 RT-Thread，本书编写尽可能做到简单、易懂，让大学生能够轻松上手 RT-Thread。希望本书能够加快 RT-Thread 在高校的普及。

总之，本书的初衷在于降低 RT-Thread 的学习门槛，让更多人能轻松学习、掌握 RT-Thread，从而参与开发 RT-Thread，共同打造开源、开放、小而美的物联网操作系统。

读者对象

- 所有使用 C/C++ 进行编程的开发人员；
- 嵌入式软硬件工程师、电子工程师、物联网开发工程师；
- 高校计算机 / 电子 / 自动化 / 通信类专业学生、老师；
- 其他对嵌入式操作系统感兴趣的人员。

如何阅读本书

为了能够阅读本书，建议先学习 C 语言和 STM32 编程知识，如果有数据结构和面向对象编程基础则更佳。学习本书时，大多数章节都有配套示例代码，这些代码都可以实际运行，建议边阅读边实战，读完一章的同时完成该章示例实验。

本书分为两大部分，共 16 章：第 1 ～ 10 章为内核篇；第 11 ～ 16 章为组件篇。

第 1 ～ 9 章介绍 RT-Thread 内核，首先对 RT-Thread 进行总体介绍，在随后各章中分别介绍 RT-Thread 的线程管理、时钟管理、线程间同步、线程间通信、内存管理、中断管理，每章都有配套的示例代码，这部分示例可运行在 Keil MDK 模拟器环境下，不需要任何硬件。

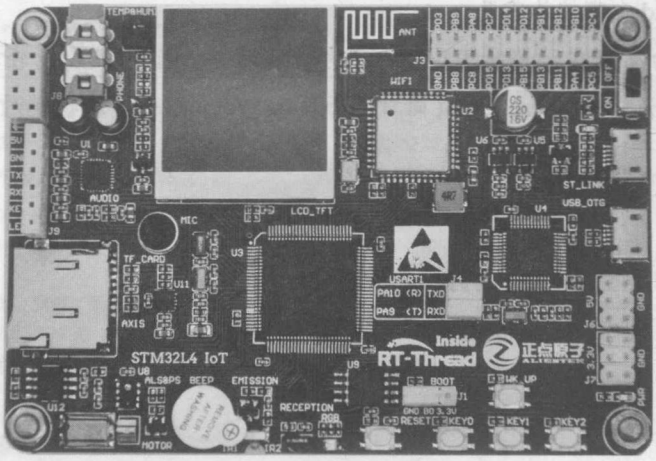
第 10 章介绍 RT-Thread 内核移植，读完本章，可以将 RT-Thread 移植到实际的硬件板上运行。

第 11 ～ 16 章介绍 RT-Thread 组件部分，分别介绍 Env 开发环境、FinSH 控制台、设备管理、文件系统和网络框架，这部分配套示例可以运行在硬件板上，分别完成外设访问、文件系统读写、网络通信功能。

本书配套资料包括实验源码及相关工具软件、硬件资料，可以通过关注微信公众号“RTThread 物联网操作系统”获得。

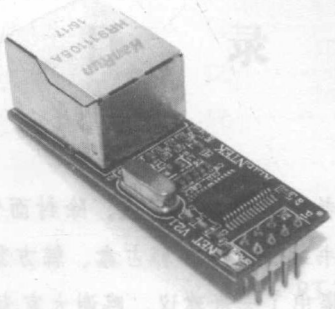
配套硬件

本书配套硬件为 RT-Thread 与正点原子联合开发的 IoT Board 开发板，基于 STM32L475 主芯片，本书组件篇配套的示例代码都基于 IoT Board。



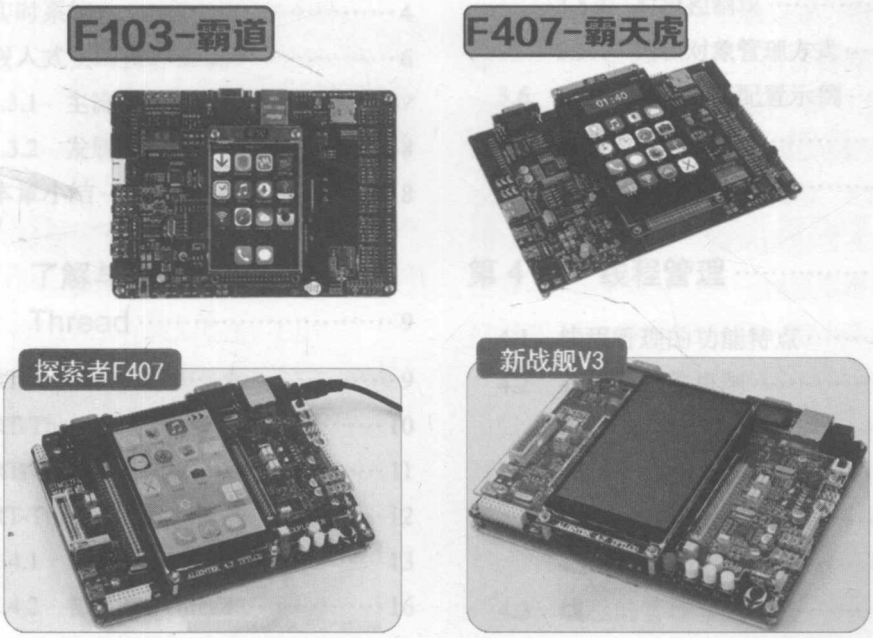
IoT Board 开发板

本书第 16 章需要用到如下图所示的 ENC28J60 模块实现网络示例功能。



ENC28J60 网络模块

如果已经购买其他开发板，如下图所示的野火和正点原子开发板，也可以配合本书进行学习，前提是根据第 10 章的介绍完成开发板上的 RT-Thread 内核移植，然后实现相关的外设驱动。



野火和正点原子开发板

勘误和支持

由于笔者水平有限，编写时间仓促，书中难免会存在一些错误或者不准确的地方，恳

请读者到论坛发帖指正，RT-Thread 官方论坛地址为 https://www.rt-thread.org/qa/。在学习过程中遇到任何问题，也可以发帖交流，期待能够得到你们的真诚反馈，在技术之路上互勉共进。

致谢

本书由诸多 RT-Thread 开发者小伙伴集体完成，除封面作者外，杨洁、罗娇、虞昊迪、张源、邹诚、姚金润也参与了本书编写工作，郭占鑫、韩方黎、杨广亮、赵盼盼等参与了本书校对工作，王卓然对书稿开发提出了宝贵建议，感谢大家为本书出版做出的贡献。

感谢机械工业出版社华章公司编辑高婧雅帮助和引导我们顺利完成全部书稿。

邱 祎

2018 年 11 月

目　录

前言

第一篇　内核篇

第 1 章　嵌入式实时操作系统 …… 2

1.1　嵌入式系统 …… 3
1.2　实时系统 …… 4
1.3　嵌入式实时操作系统 …… 6
1.3.1　主流嵌入式实时操作系统 …… 7
1.3.2　发展趋势 …… 8
1.4　本章小结 …… 8

第 2 章　了解与快速上手 RT-Thread …… 9

2.1　RT-Thread 概述 …… 9
2.2　RT-Thread 的架构 …… 10
2.3　RT-Thread 的获取 …… 11
2.4　RT-Thread 快速上手 …… 12
2.4.1　准备环境 …… 13
2.4.2　初识 RT-Thread …… 16
2.4.3　跑马灯的例子 …… 20
2.5　本章小结 …… 21

第 3 章　内核基础 …… 22

3.1　RT-Thread 内核介绍 …… 22
3.2　RT-Thread 启动流程 …… 24
3.3　RT-Thread 程序内存分布 …… 26
3.4　RT-Thread 自动初始化机制 …… 28
3.5　RT-Thread 内核对象模型 …… 29
3.5.1　静态对象和动态对象 …… 29
3.5.2　内核对象管理架构 …… 31
3.5.3　对象控制块 …… 33
3.5.4　内核对象管理方式 …… 34
3.6　RT-Thread 内核配置示例 …… 36
3.7　常见宏定义说明 …… 38
3.8　本章小结 …… 39

第 4 章　线程管理 …… 40

4.1　线程管理的功能特点 …… 40
4.2　线程的工作机制 …… 41
4.2.1　线程控制块 …… 41
4.2.2　线程的重要属性 …… 42
4.2.3　线程状态切换 …… 45
4.2.4　系统线程 …… 46
4.3　线程的管理方式 …… 46
4.3.1　创建和删除线程 …… 47
4.3.2　初始化和脱离线程 …… 48
4.3.3　启动线程 …… 49
4.3.4　获得当前线程 …… 50
4.3.5　使线程让出处理器资源 …… 50

4.3.6 使线程睡眠 …… 50
4.3.7 挂起和恢复线程 …… 51
4.3.8 控制线程 …… 52
4.3.9 设置和删除空闲钩子 …… 52
4.3.10 设置调度器钩子 …… 53
4.4 线程应用示例 …… 53
4.4.1 创建线程示例 …… 54
4.4.2 线程时间片轮转调度示例 …… 56
4.4.3 线程调度器钩子示例 …… 57
4.5 本章小结 …… 59

第 5 章 时钟管理 …… 60

5.1 时钟节拍 …… 60
5.1.1 时钟节拍的实现方式 …… 60
5.1.2 获取时钟节拍 …… 61
5.2 定时器管理 …… 62
5.2.1 RT-Thread 定时器介绍 …… 62
5.2.2 定时器的工作机制 …… 63
5.2.3 定时器的管理方式 …… 65
5.3 定时器应用示例 …… 69
5.4 高精度延时 …… 72
5.5 本章小结 …… 73

第 6 章 线程间同步 …… 74

6.1 信号量 …… 75
6.1.1 信号量的工作机制 …… 75
6.1.2 信号量控制块 …… 75
6.1.3 信号量的管理方式 …… 76
6.1.4 信号量应用示例 …… 79
6.1.5 信号量的使用场合 …… 85
6.2 互斥量 …… 87
6.2.1 互斥量的工作机制 …… 87
6.2.2 互斥量控制块 …… 89
6.2.3 互斥量的管理方式 …… 89
6.2.4 互斥量应用示例 …… 92
6.2.5 互斥量的使用场合 …… 97
6.3 事件集 …… 97
6.3.1 事件集的工作机制 …… 97
6.3.2 事件集控制块 …… 98
6.3.3 事件集的管理方式 …… 99
6.3.4 事件集应用示例 …… 101
6.3.5 事件集的使用场合 …… 104
6.4 本章小结 …… 104

第 7 章 线程间通信 …… 105

7.1 邮箱 …… 105
7.1.1 邮箱的工作机制 …… 105
7.1.2 邮箱控制块 …… 106
7.1.3 邮箱的管理方式 …… 106
7.1.4 邮箱使用示例 …… 110
7.1.5 邮箱的使用场合 …… 112
7.2 消息队列 …… 113
7.2.1 消息队列的工作机制 …… 113
7.2.2 消息队列控制块 …… 114
7.2.3 消息队列的管理方式 …… 115
7.2.4 消息队列应用示例 …… 118
7.2.5 消息队列的使用场合 …… 121
7.3 信号 …… 123
7.3.1 信号的工作机制 …… 123
7.3.2 信号的管理方式 …… 124
7.3.3 信号应用示例 …… 126
7.4 本章小节 …… 128

第 8 章 内存管理 …… 129

8.1 内存管理的功能特点 …… 129
8.2 内存堆管理 …… 130

8.2.1 小内存管理算法 …… 131
8.2.2 slab 管理算法 …… 132
8.2.3 memheap 管理算法 …… 133
8.2.4 内存堆配置和初始化 …… 134
8.2.5 内存堆的管理方式 …… 134
8.2.6 内存堆管理应用示例 …… 136
8.3 内存池 …… 138
8.3.1 内存池的工作机制 …… 139
8.3.2 内存池的管理方式 …… 140
8.3.3 内存池应用示例 …… 143
8.4 本章小结 …… 145

第 9 章 中断管理 …… 146

9.1 Cortex-M CPU 架构基础 …… 146
9.1.1 寄存器介绍 …… 147
9.1.2 操作模式和特权级别 …… 148
9.1.3 嵌套向量中断控制器 …… 148
9.1.4 PendSV 系统调用 …… 149
9.2 RT-Thread 中断工作机制 …… 149
9.2.1 中断向量表 …… 149
9.2.2 中断处理过程 …… 151
9.2.3 中断嵌套 …… 153
9.2.4 中断栈 …… 154
9.2.5 中断的底半处理 …… 154
9.3 RT-Thread 中断管理接口 …… 156
9.3.1 中断服务程序挂接 …… 157
9.3.2 中断源管理 …… 158
9.3.3 全局中断开关 …… 158
9.3.4 中断通知 …… 160
9.4 中断与轮询 …… 161
9.5 全局中断开关使用示例 …… 162
9.6 本章小结 …… 164

第 10 章 内核移植 …… 165

10.1 CPU 架构移植 …… 165
10.1.1 实现全局中断开关 …… 166
10.1.2 实现线程栈初始化 …… 167
10.1.3 实现上下文切换 …… 168
10.1.4 实现时钟节拍 …… 174
10.2 BSP 移植 …… 175
10.3 内核移植示例 …… 175
10.3.1 准备裸机工程 …… 176
10.3.2 建立 RT-Thread 工程 …… 177
10.3.3 实现时钟管理 …… 179
10.3.4 实现控制台输出 …… 180
10.3.5 实现动态堆内存管理 …… 181
10.3.6 移植到更多开发板 …… 183
10.4 本章小结 …… 184

第二篇 组件篇

第 11 章 Env 辅助开发环境 …… 186

11.1 Env 简介 …… 186
11.2 Env 的功能特点 …… 187
11.3 Env 工程构建示例 …… 189
11.4 构建更多 MDK 工程 …… 196
11.4.1 创建外设示例工程 …… 196
11.4.2 创建文件系统示例工程 … 198
11.4.3 创建网络示例工程 …… 202
11.5 本章小结 …… 206

第 12 章 FinSH 控制台 …… 207

12.1 FinSH 介绍 …… 207
12.2 FinSH 内置命令 …… 209
12.2.1 显示线程状态 …… 210

12.2.2 显示信号量状态 ····· 210
12.2.3 显示事件状态 ····· 210
12.2.4 显示互斥量状态 ····· 210
12.2.5 显示邮箱状态 ····· 211
12.2.6 显示消息队列状态 ····· 211
12.2.7 显示内存池状态 ····· 211
12.2.8 显示定时器状态 ····· 212
12.2.9 显示设备状态 ····· 212
12.2.10 显示动态内存状态 ····· 212
12.3 自定义 FinSH 命令 ····· 213
12.3.1 自定义 msh 命令 ····· 213
12.3.2 自定义 C-Style 命令和变量 ····· 213
12.3.3 自定义命令重命名 ····· 214
12.4 FinSH 功能配置 ····· 214
12.5 FinSH 应用示例 ····· 216
12.5.1 自定义 msh 命令示例 ····· 216
12.5.2 带参数的 msh 命令示例 ····· 217
12.6 本章小结 ····· 218

第 13 章 I/O 设备管理 ····· 219

13.1 I/O 设备介绍 ····· 219
13.1.1 I/O 设备管理框架 ····· 219
13.1.2 I/O 设备模型 ····· 221
13.1.3 I/O 设备类型 ····· 222
13.2 创建和注册 I/O 设备 ····· 223
13.3 访问 I/O 设备 ····· 226
13.3.1 查找设备 ····· 226
13.3.2 初始化设备 ····· 227
13.3.3 打开和关闭设备 ····· 227
13.3.4 控制设备 ····· 228
13.3.5 读写设备 ····· 229
13.3.6 数据收发回调 ····· 229
13.3.7 设备访问示例 ····· 230
13.4 本章小结 ····· 231

第 14 章 通用外设接口 ····· 232

14.1 UART 串口 ····· 232
14.1.1 串口设备管理 ····· 233
14.1.2 创建和注册串口设备 ····· 233
14.1.3 访问串口设备 ····· 235
14.1.4 串口设备使用示例 ····· 235
14.2 GPIO ····· 237
14.2.1 PIN 设备管理 ····· 238
14.2.2 创建和注册 PIN 设备 ····· 238
14.2.3 访问 PIN 设备 ····· 239
14.2.4 PIN 设备使用示例 ····· 242
14.3 SPI 总线 ····· 243
14.3.1 SPI 设备管理 ····· 244
14.3.2 创建和注册 SPI 总线设备 ····· 246
14.3.3 创建和挂载 SPI 从设备 ····· 247
14.3.4 访问 SPI 从设备 ····· 249
14.3.5 特殊使用场景 ····· 254
14.3.6 SPI 设备使用示例 ····· 255
14.4 I2C 总线 ····· 256
14.4.1 I2C 设备管理 ····· 258
14.4.2 创建和注册 I2C 总线设备 ····· 258
14.4.3 访问 I2C 设备 ····· 259
14.4.4 I2C 设备应用示例 ····· 260
14.5 运行设备应用示例 ····· 263
14.5.1 运行 PIN 设备示例 ····· 264
14.5.2 运行 SPI 设备示例 ····· 265

14.5.3 运行 I2C 设备示例 …… 266
14.5.4 运行串口设备示例 …… 266
14.6 本章小结 …… 267

第 15 章 虚拟文件系统 …… 268

15.1 DFS 介绍 …… 268
15.1.1 DFS 架构 …… 269
15.1.2 POSIX 接口层 …… 269
15.1.3 虚拟文件系统层 …… 270
15.1.4 设备抽象层 …… 270
15.2 文件系统挂载管理 …… 271
15.2.1 DFS 组件初始化 …… 271
15.2.2 注册文件系统 …… 271
15.2.3 将存储设备注册为块设备 …… 271
15.2.4 格式化文件系统 …… 272
15.2.5 挂载文件系统 …… 273
15.2.6 卸载文件系统 …… 273
15.3 文件管理 …… 273
15.3.1 打开和关闭文件 …… 273
15.3.2 读写数据 …… 274
15.3.3 重命名 …… 275
15.3.4 获取状态 …… 275
15.3.5 删除文件 …… 275
15.3.6 同步文件数据到存储设备 …… 276
15.3.7 查询文件系统相关信息 …… 276
15.3.8 监视 I/O 设备状态 …… 276
15.4 目录管理 …… 277
15.4.1 创建和删除目录 …… 277
15.4.2 打开和关闭目录 …… 277
15.4.3 读取目录 …… 278
15.4.4 获取目录流的读取位置 …… 278
15.4.5 设置下次读取目录的位置 …… 278
15.4.6 重设读取目录的位置为开头位置 …… 279
15.5 DFS 功能配置 …… 279
15.6 DFS 应用示例 …… 279
15.6.1 准备工作 …… 280
15.6.2 读写文件示例 …… 283
15.6.3 更改文件名称示例 …… 284
15.6.4 获取文件状态示例 …… 285
15.6.5 创建目录示例 …… 286
15.6.6 读取目录示例 …… 286
15.6.7 设置读取目录位置示例 …… 287
15.7 本章小结 …… 289

第 16 章 网络框架 …… 290

16.1 TCP/IP 网络协议简介 …… 290
16.1.1 OSI 参考模型 …… 290
16.1.2 TCP/IP 参考模型 …… 291
16.1.3 TCP/IP 参考模型和 OSI 参考模型的区别 …… 291
16.1.4 IP 地址 …… 292
16.1.5 子网掩码 …… 292
16.1.6 MAC 地址 …… 292
16.2 RT-Thread 网络框架介绍 …… 292
16.3 网络框架工作流程 …… 294
16.3.1 网络协议簇注册 …… 294
16.3.2 网络数据接收流程 …… 295
16.3.3 网络数据发送流程 …… 296
16.4 网络套接字编程 …… 296
16.4.1 TCP socket 通信流程 …… 296

16.4.2 UDP socket 通信流程 ····· 297
16.4.3 创建套接字 ····· 298
16.4.4 绑定套接字 ····· 298
16.4.5 建立 TCP 连接 ····· 299
16.4.6 数据传输 ····· 300
16.4.7 关闭网络连接 ····· 301
16.5 网络功能配置 ····· 302
16.6 网络应用示例 ····· 303
16.6.1 准备工作 ····· 303
16.6.2 TCP 客户端示例 ····· 306
16.6.3 UDP 客户端示例 ····· 310
16.7 本章小结 ····· 312
附录 A menuconfig 配置选项 ····· 313
附录 B SCons 构建系统 ····· 317

第一篇

内 核 篇

第 1 章　嵌入式实时操作系统
第 2 章　了解与快速上手 RT-Thread
第 3 章　内核基础
第 4 章　线程管理
第 5 章　时钟管理
第 6 章　线程间同步
第 7 章　线程间通信
第 8 章　内存管理
第 9 章　中断管理
第 10 章　内核移植

第 1 章
嵌入式实时操作系统

操作系统是指管理和控制计算机硬件与软件资源的计算机程序，是直接运行在计算机上的最基本的系统软件，任何其他软件都必须在操作系统的支持下才能运行。按应用领域来划分，操作系统可分为桌面操作系统、服务器操作系统、移动操作系统和嵌入式操作系统几类。

桌面操作系统是指运行在个人电脑上的操作系统，当前主流的桌面操作系统是微软的 Windows 操作系统，此外，Linux、Mac OS 也是桌面操作系统。

服务器操作系统是指运行在大型服务器上的操作系统，例如云服务器、数据库服务器、网络服务器等，当前主流的服务器操作系统是 Linux，微软的 Windows 服务器操作系统也有部分市场份额。

移动操作系统是指运行在手机、平板、智能电视上的操作系统，谷歌的 Android 和苹果的 iOS 都属于移动操作系统，传统的移动设备，如手机、平板，也属于嵌入式设备，只是随着移动设备使用的芯片越来越强大，它们跟传统的嵌入式设备差异明显变大，因此也将移动操作系统单独分类。

嵌入式操作系统指用在嵌入式系统的操作系统。嵌入式系统使用非常广泛，可以理解为除了服务器、个人电脑、移动设备外的计算机都是嵌入式设备。从军事到民用，从工业控制到网络应用，嵌入式系统在我们的生活中无处不在。以下是一些典型的嵌入式设备举例，图 1-1 中也列出了一些嵌入式操作系统的应用实物的图片。

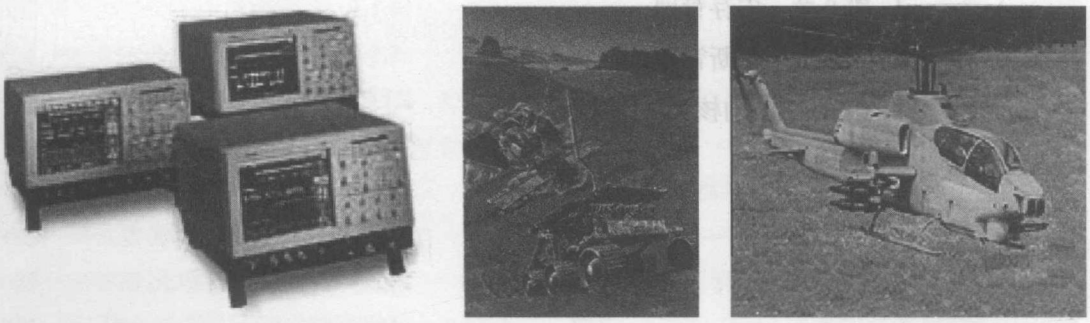


图 1-1　常见嵌入式操作系统的应用

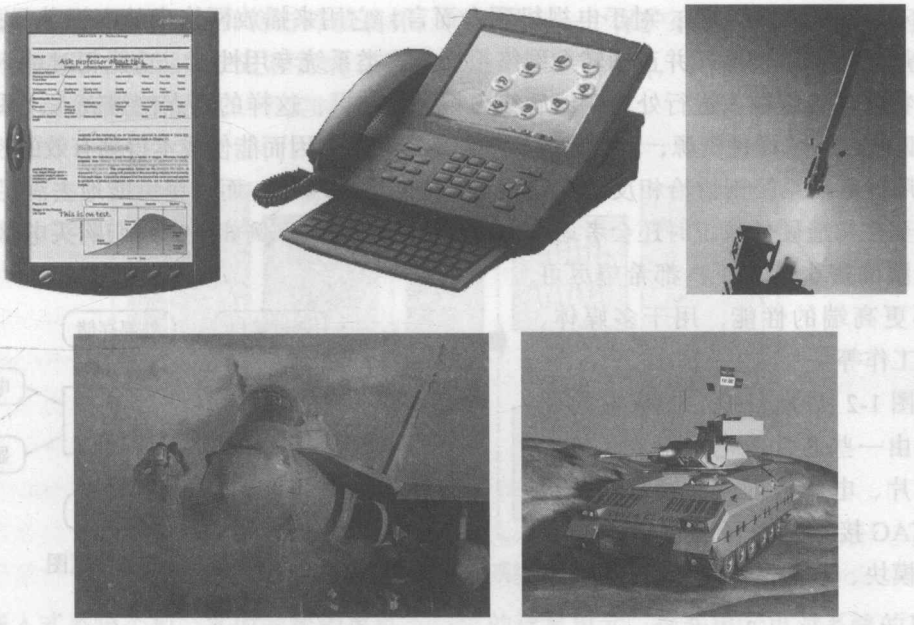


图 1-1 （续）

- **军用：**各种武器控制（火炮控制、导弹控制、智能炸弹制导引爆装置）、坦克、舰艇、轰炸机等陆海空各种军用电子装备，雷达、电子对抗军事通信装备，野战指挥作战使用的各种专用设备等。
- **消费电子：**各种信息家电产品，如数字电视机、机顶盒、数码相机、音响设备、可视电话、家庭网络设备、洗衣机、电冰箱、智能玩具等。
- **工业控制：**各种智能测量仪表、数控装置、可编程控制器、控制机、分布式控制系统、现场总线仪表及控制系统、工业机器人、机电一体化机械设备、汽车电子设备等。
- **网络应用：**网络基础设施、接入设备、移动终端设备、共享单车、水电气表、物联网终端设备等。
- **其他：**各类收款机、POS 系统、电子秤、条形码阅读机、商用终端、银行点钞机、IC 卡输入设备、取款机、自动柜员机、自动服务终端、防盗系统，以及各种银行专业外围设备与各种医疗电子仪器，无一不用到嵌入式系统。

1.1　嵌入式系统

嵌入式系统是一种完全嵌入在装置或设备内部为满足特定需求而设计的计算机系统，生活中常见的嵌入式系统就有：电视机顶盒、路由器、电冰箱、微波炉与移动电话等。它

们都具有某种特定的功能：对于电视机顶盒而言，它用来播放网络中的电视节目；同样，路由器用于选择最优路径并正确转发网络报文。这类系统专用性强、功能相对单一，通常只针对特定的外部输入进行处理，然后给出相应的结果，这样的特点使得嵌入式系统只需具备相匹配的少量硬件资源，就可完成所需的特定功能，因而能使成本得到有效的控制。

通用计算机系统则恰恰相反，它们并不针对特定的需求，而是尽可能地去满足各种需求，甚至在构造硬件系统时还会考虑未来几年的需求变化。例如，在人们购买电脑时，在自身有限的资金情况下，都希望尽可能获得更高端的性能，用于多媒体、游戏及工作等。

如图 1-2 所示，嵌入式系统的硬件设备由一些芯片及电路组成，包括主控芯片、电源管理、开发调试时用到的 JTAG 接口，也可能包含一些数据采集模块、通信模块及音频 / 视频模块等。

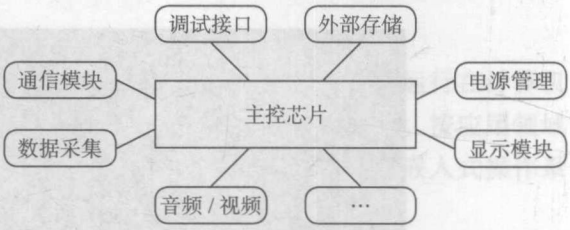


图 1-2 嵌入式系统硬件框图

1.2 实时系统

系统的实时性指的是在固定的时间内正确地对外部事件做出响应。在这段“时间内”，系统内部会做一些处理，例如输入数据的分析计算、加工处理等。而在这段时间之外，系统可能会空闲下来，做一些空余的事。以一个手机终端为例：当一个电话拨入的时候，系统应当及时发出振铃、声音提示以通知主人有来电，询问是否进行接听；而在非电话拨入的时候，人们可以用它进行一些其他操作，例如听音乐、玩游戏等。

从上面的例子我们可以看出，实时系统是一种需求倾向性的系统，对于实时的任务需要在第一时间内做出回应，而对非实时任务则可以在实时事件到达时为之让路——被抢占。所以也可以将实时系统看成是一个等级系统，不同重要性的任务具有不同的优先等级：重要的任务能够优先被响应执行，非重要的任务可以适当往后推迟。

实时计算可以定义成这样一类计算，即系统的正确性不仅取决于计算的逻辑结果，还依赖于产生结果的时间。有两个关键点，即正确地完成和在给定的时间内完成，且两者重要性是等同的。如果计算结果出错，这将不是一个正确的系统，而计算结果正确，但计算所耗费的时间已经偏离需求设定的时间，那么这也不是一个实时系统。图 1-3 描述了一个实时系统。

对于输入的信号、事件，实时系统必须能够在规定的时间内得到正确的响应，而不管这些事件是单一事件、多重事件，还是同步信号或异步信号。

举一个例子说明：假设一颗子弹从 20 米外射向一个玻璃杯，子弹的速度是 v 米 / 秒，