

嵌入式开发系列

# Cortex-M3 + $\mu$ C/OS-II 嵌入式系统开发入门与应用

华清远见嵌入式培训中心 陈瑶 李佳 宋宝华 编著

详细介绍Cortex-M3的内核结构、编程模式及其开发工具

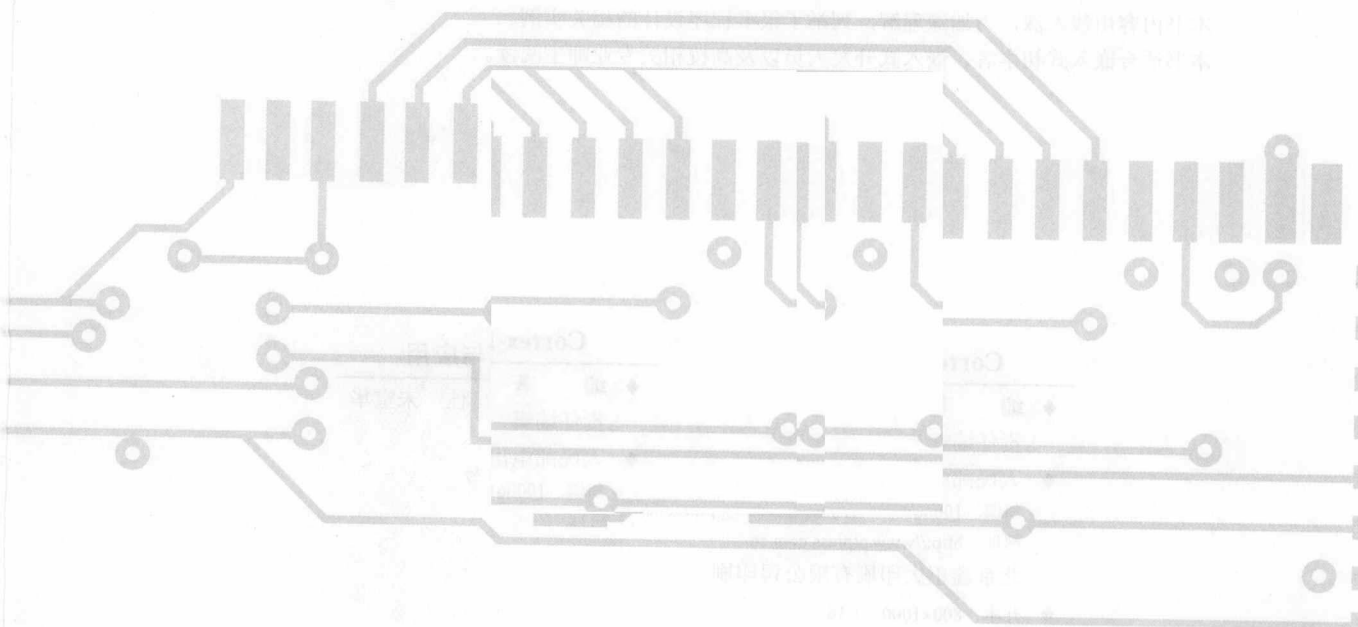
从 $\mu$ COS-II在Cortex-M3上的移植入手，介绍在Cortex-M3平台上用 $\mu$ COS-II操作系统开发的方法

通过一个Cortex-M3 +  $\mu$ C/OS-II完整应用实例全面了解开发过程

嵌入式开发系列

# Cortex-M3 + $\mu$ C/OS-II 嵌入式系统开发入门与应用

华清远见嵌入式培训中心 陈瑶 李佳 宋宝华 编著



人民邮电出版社

北京

## 图书在版编目(CIP)数据

Cortex-M3 + uC/OS-II嵌入式系统开发入门与应用 /  
陈瑶, 李佳, 宋宝华编著. — 北京: 人民邮电出版社,  
2010.8

ISBN 978-7-115-23105-5

I. ①C… II. ①陈… ②李… ③宋… III. ①微处理  
器—系统设计 IV. ①TP332

中国版本图书馆CIP数据核字(2010)第107900号

## 内 容 提 要

本书首先讲解了 Cortex-M3 处理器的架构及 RealView MDK 开发平台的相关知识, 然后对 $\mu$ C/OS-II 的内核移植、驱动开发、应用程序开发进行了介绍, 最后综合运用所学知识介绍了一个 $\mu$ C/OS-II 的程序设计实例。

本书内容由浅入深, 为加深理解, 列举了很多程序设计的相关实例。

本书适合嵌入式初学者、嵌入式开发人员以及高校相关专业师生阅读。

### Cortex-M3+ $\mu$ C/OS-II 嵌入式系统开发入门与应用

◆ 编 著 华清远见嵌入式培训中心 陈 瑶 李 佳 宋宝华  
责任编辑 黄 焱

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号  
邮编 100061 电子函件 315@ptpress.com.cn  
网址 <http://www.ptpress.com.cn>  
北京鑫正大印刷有限公司印刷

◆ 开本: 800×1000 1/16

印张: 19.5

字数: 428 千字

印数: 1—3 500 册

2010 年 8 月第 1 版

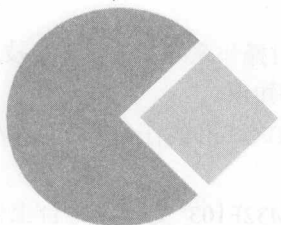
2010 年 8 月北京第 1 次印刷

ISBN 978-7-115-23105-5

定价: 39.80 元

读者服务热线: (010)67132692 印装质量热线: (010)67129223

反盗版热线: (010)67171154



## 前言

### 写作背景

Cortex-M3 是 ARM 公司推出的最新的针对微控制器应用的内核,它提供了业界领先的高性能和低成本的解决方案,将成为 MCU (Micro Controller Unit, 微控制单元) 应用的热点和主流。Cortex-M3 是一个 32 位的核,在传统的单片机领域中,有一些不同于通用 32 位 CPU 应用的要求。谭军博士曾说,在工控领域,用户要求具有更快的中断速度,而 Cortex-M3 采用了 Tail-Chaining 中断技术,完全基于硬件进行中断处理,最多可减少 12 个时钟周期,在实际应用中可减少 70% 的中断,完全符合用户的需求。

Cortex-M3 采用了新型的单线调试 (Single Wire) 技术,专门拿出一个引脚来做调试,从而节约了大笔调试工具的费用。同时, Cortex-M3 中还集成了大部分存储器、控制器,这样工程师可以直接在 MCU 外连接 Flash,降低了设计和应用的难度。

Cortex-M3 继承了单片机操作方便、调试简单等特点,同时在运算速度上比单片机有了很大的提高,这就具备了操作系统所需要的运行条件。

### 本书内容

本书以  $\mu\text{C}/\text{OS-II}$  实时操作为例,由浅入深地介绍了 Cortex-M3 内核结构及以  $\mu\text{C}/\text{OS-II}$  为基础的复杂系统的实现。全书共分两部分。第 1 部分详细介绍了 Cortex-M3 的内核结构、编程模式及其开发工具;第 2 部分从  $\mu\text{C}/\text{OS-II}$  在 Cortex-M3 上的移植入手,用浅显易懂的语言讲述了在 Cortex-M3 平台上用  $\mu\text{C}/\text{OS-II}$  操作系统开发的方法。具体安排如下。

第 1 部分包括第 1 章~第 5 章。

第 1 章 走进 ARM 微处理器。介绍了 ARM 处理器的发展及 ARM V7 的体系结构和新增指令。

第 2 章 Cortex-M3 处理器体系结构。介绍了 Cortex-M3 的编程模式、存储器结构、异常处理和电源管理。

第 3 章 ARM 开发环境 RealView MDK 平台搭建。介绍了常用的 ARM 开发工具、RealView MDK 的使用、仿真工具 ULINK2 等知识。

第 4 章 基于 RealView 开发环境的嵌入式软件开发。介绍了 RealView 编译器的默认行为、

映像文件存储器映射调整、复位和初始化等知识。

第5章 STM32F103 处理器内部资源 C 编程与实例。介绍了 I/O 控制模块 C 编程与实例、中断控制模块 C 编程与实例、A/D 转换和 DMA 编程实例、实时钟 RTC 编程实例、串行外设接口 SPI 编程实例、CAN 总线编程实例、窗口看门狗 WWDG 实例、UART 编程实例。

第2部分包括第6章~第8章。

第6章 μC/OS-II 操作系统基础及其移植开发初步。介绍了实时操作系统基本原理与技术、μC/OS-II 应用程序开发、μC/OS-II 在 STM32F103 处理器上的移植等知识。

第7章 μC/OS-II 的内核机制。介绍了 μC/OS-II 的内核结构、μC/OS-II 的任务管理、μC/OS-II 的时间管理、任务之间通信与同步、内存管理等知识。

第8章 基于 μC/OS-II 的程序设计实例。介绍了一个在 STM32F103 处理器平台上使用 μC/OS-II 实时操作系统的程序设计实例。

#### 作者简介

陈瑶

华清远见特邀顾问，任教于北京工业大学计算机学院，计算机软件与理论系。长期以来从事软件工程和嵌入式开发方面的教学与科研工作。曾于 2004 年作为访问学者在美国 University of Massachusetts 进修。参加过多项国家自然科学基金、北京市自然科学基金项目，发表多篇论文。

李佳

著名 ARM 培训讲师，有丰富的 ARM 培训经验，多年来一直致力于 ARM 的研究与应用，有多年的基于 ARM 的嵌入式开发经验，曾在不同的 ARM 处理器平台上开发过众多领域的应用项目。著有国内第一本 ARM 手册类图书《ARM 系列处理器应用技术完全手册》。

由于本书涉及的知识领域广泛且变化日新月异，加之编者水平有限及时间仓促，难免有差错和不足之处，敬请专家和读者批评指正。

宋宝华

华清远见金牌讲师，著有《Linux 设备驱动开发详解》，译有《Essential Linux Device Drivers》，曾为思科、阿尔卡特、朗讯、四方、意法半导体等知名企业提供嵌入式 Linux 企业培训和技术服务，受到企业广泛好评。活跃于 Linux 社区，已给 Linux 内核贡献了逾 2 万行代码。

#### 相关学习资料

嵌入式学习资料：

<http://download.farsight.com.cn/download>

单片机学习资料：

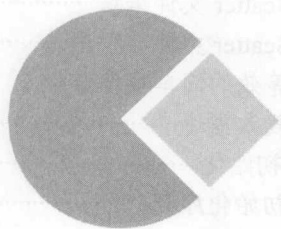
<http://www.coronabook.com>

Linux 学习资料：

<http://www.linuxdriver.cn>

编者

2010.7



# 目 录

## 第 1 部分

|                                      |    |
|--------------------------------------|----|
| 第 1 章 走进 ARM 微处理器 .....              | 3  |
| 1.1 ARM 体系结构的源头——<br>ARM 公司 .....    | 3  |
| 1.2 ARM 处理器的发展历程 .....               | 4  |
| 1.3 ARM 体系结构 .....                   | 5  |
| 1.3.1 ARM Cortex 处理器技术<br>特点 .....   | 6  |
| 1.3.2 ARM Cortex 系列处理器<br>比较 .....   | 7  |
| 1.4 ARM v7 新增指令 .....                | 12 |
| 1.4.1 位操作指令 .....                    | 13 |
| 1.4.2 比特反转指令 .....                   | 13 |
| 1.4.3 16-bit 常数操作 .....              | 13 |
| 1.4.4 调转表指令 .....                    | 14 |
| 1.4.5 IT 指令 .....                    | 14 |
| 1.4.6 检查是否为零并跳转<br>指令 (CBZ) .....    | 14 |
| 第 2 章 Cortex-M3 处理器体系结构 .....        | 16 |
| 2.1 Cortex-M3 综述 .....               | 16 |
| 2.2 Cortex-M3 编程模式 .....             | 17 |
| 2.2.1 Cortex-M3 的工作模式和<br>工作状态 ..... | 18 |
| 2.2.2 特权访问和用户访问 .....                | 18 |
| 2.2.3 Cortex-M3 的寄存器组织 .....         | 18 |
| 2.2.4 Cortex-M3 的数据类型 .....          | 19 |
| 2.2.5 Cortex-M3 的存储器格式 .....         | 20 |
| 2.3 Cortex-M3 的存储器管理 .....           | 21 |
| 2.3.1 存储器映射 .....                    | 21 |
| 2.3.2 Bit-banding 机制 .....           | 23 |
| 2.4 Cortex-M3 的异常处理 .....            | 24 |
| 2.4.1 异常优先级 .....                    | 26 |
| 2.4.2 异常处理的堆栈使用 .....                | 27 |
| 2.4.3 Cortex-M3 特有的异常<br>处理机制 .....  | 29 |
| 2.4.4 异常退出 .....                     | 30 |
| 2.4.5 复位异常 .....                     | 31 |
| 2.4.6 中止 (Abort) 异常 .....            | 34 |
| 2.5 Cortex-M3 的电源管理 .....            | 37 |
| 2.5.1 SLEEPING .....                 | 37 |
| 2.5.2 SLEEPDEEP .....                | 38 |

|       |                                             |    |       |                                    |     |
|-------|---------------------------------------------|----|-------|------------------------------------|-----|
| 2.6   | 嵌套向量中断控制器 NVIC .....                        | 38 | 4.2.2 | 从最终代码映像中去掉<br>Semihosting .....    | 74  |
| 2.7   | 存储器保护单位 MPU .....                           | 41 | 4.3   | 映像文件存储器映射调整 .....                  | 74  |
| 2.8   | Cortex-M3 开发平台——<br>STM32V100 评估板介绍 .....   | 43 | 4.3.1 | 关于分散加载 .....                       | 74  |
| 第 3 章 | ARM 开发环境 RealView MDK<br>平台搭建 .....         | 46 | 4.3.2 | Scatter 文件语法 .....                 | 78  |
| 3.1   | 常用 ARM 开发工具 .....                           | 46 | 4.3.3 | Scatter 文件典型用法 .....               | 85  |
| 3.1.1 | 免费平台 GNU .....                              | 47 | 4.3.4 | 等效的简单映像分散<br>载入描述 .....            | 91  |
| 3.1.2 | RealView Development<br>Suite 开发工具 .....    | 48 | 4.4   | 复位和初始化 .....                       | 95  |
| 3.1.3 | ARM 硬件仿真器 Multi-<br>ICE 与 Multi-Trace ..... | 49 | 4.4.1 | 初始化序列 .....                        | 95  |
| 3.1.4 | Multi 2000 .....                            | 50 | 4.4.2 | 向量表 .....                          | 96  |
| 3.1.5 | Hitool for ARM .....                        | 50 | 4.4.3 | ROM/RAM 重映射 .....                  | 97  |
| 3.1.6 | RealView MDK .....                          | 50 | 4.4.4 | 与局部存储器设置有关的<br>考虑事项 .....          | 100 |
| 3.2   | RealView MDK 概述 .....                       | 51 | 4.4.5 | 栈指针初始化 .....                       | 100 |
| 3.3   | RealView MDK 的使用 .....                      | 54 | 4.4.6 | 硬件初始化 .....                        | 102 |
| 3.3.1 | 创建工程 .....                                  | 54 | 第 5 章 | STM32F103 处理器内部资源<br>C 编程与实例 ..... | 104 |
| 3.3.2 | 编译、链接工程 .....                               | 57 | 5.1   | I/O 控制模块 C 编程与实例 .....             | 104 |
| 3.3.3 | 程序调试 .....                                  | 59 | 5.1.1 | 实例内容与目标 .....                      | 104 |
| 3.3.4 | 工程选项设置 .....                                | 60 | 5.1.2 | I/O 控制模块操作原理 .....                 | 104 |
| 3.4   | 仿真工具——ULINK2 .....                          | 61 | 5.1.3 | I/O 控制实例操作步骤 .....                 | 109 |
| 3.5   | RealView MDK 工程实例 .....                     | 62 | 5.1.4 | I/O 控制实例参考程序及<br>说明 .....          | 109 |
| 第 4 章 | 基于 RealView 开发环境的嵌入式<br>软件开发 .....          | 66 | 5.2   | 中断控制模块 C 编程与实例 .....               | 112 |
| 4.1   | RealView 编译器的缺省行为 .....                     | 66 | 5.2.1 | 实例内容与目标 .....                      | 112 |
| 4.1.1 | Semihosting .....                           | 66 | 5.2.2 | 中断模块 NVIC 的操作<br>原理 .....          | 113 |
| 4.1.2 | C 库结构 .....                                 | 69 | 5.2.3 | 中断控制实例操作步骤 .....                   | 115 |
| 4.1.3 | 默认存储器映射 .....                               | 70 | 5.2.4 | 中断控制实例参考程序及<br>说明 .....            | 116 |
| 4.1.4 | 链接程序放置规则 .....                              | 71 | 5.3   | A/D 转换和 DMA 编程实例 .....             | 119 |
| 4.1.5 | 应用程序启动 .....                                | 71 | 5.3.1 | 实例内容与目标 .....                      | 119 |
| 4.2   | 调整 C 库使其适应目标硬件 .....                        | 73 | 5.3.2 | A/D 转换控制器与 DMA                     |     |
| 4.2.1 | C 库函数重定向 .....                              | 73 |       |                                    |     |

|                                         |     |                                 |     |
|-----------------------------------------|-----|---------------------------------|-----|
| 控制器操作原理 .....                           | 119 | 5.6 CAN 总线编程实例 .....            | 149 |
| 5.3.3 A/D 转换和 DMA 编程<br>实例操作步骤 .....    | 126 | 5.6.1 实例内容与目标 .....             | 149 |
| 5.3.4 A/D 转换和 DMA 编程<br>实例参考程序及说明 ..... | 126 | 5.6.2 CAN 总线操作原理 .....          | 149 |
| 5.4 实时钟 RTC 编程实例 .....                  | 133 | 5.6.3 CAN 总线编程实例操作<br>步骤 .....  | 156 |
| 5.4.1 实例内容与目标 .....                     | 133 | 5.6.4 CAN 总线实例参考程序及<br>说明 ..... | 157 |
| 5.4.2 STM32F103 实时钟操作<br>原理 .....       | 134 | 5.7 窗口看门狗 WWDG 实例 .....         | 163 |
| 5.4.3 实时钟 RTC 编程实例<br>操作步骤 .....        | 135 | 5.7.1 实例内容与目标 .....             | 163 |
| 5.4.4 实时钟 RTC 编程实例<br>参考程序及说明 .....     | 136 | 5.7.2 WWDG 操作原理 .....           | 164 |
| 5.5 串行外设接口 SPI 编程实例 .....               | 139 | 5.7.3 看门狗实例操作步骤 .....           | 166 |
| 5.5.1 实例内容与目标 .....                     | 139 | 5.7.4 看门狗实例参考程序及<br>说明 .....    | 166 |
| 5.5.2 SPI 接口操作原理 .....                  | 139 | 5.8 UART 编程实例 .....             | 169 |
| 5.5.3 SPI 接口实例操作步骤 .....                | 144 | 5.8.1 实例内容与目标 .....             | 169 |
| 5.5.4 SPI 接口实例参考程序及<br>说明 .....         | 144 | 5.8.2 UART 编程原理 .....           | 169 |
|                                         |     | 5.8.3 串口编程实例操作步骤 .....          | 180 |
|                                         |     | 5.8.4 串口编程实例参考程序及<br>说明 .....   | 180 |

## 第 2 部分

### 第 6 章 $\mu\text{C/OS-II}$ 操作系统基础及其

移植开发初步 ..... 189

#### 6.1 实时操作系统基本原理与 技术 .....

6.1.1 实时操作系统基本特征 ..... 189

6.1.2 实时操作系统的关键技术  
指标 .....

6.1.3 实时操作系统基本术语 ..... 191

#### 6.2 $\mu\text{C/OS-II}$ 应用程序开发 .....

6.2.1  $\mu\text{C/OS-II}$  的变量类型 ..... 192

6.2.2 应用程序的基本结构 ..... 193

6.2.3  $\mu\text{C/OS-II}$  API 介绍 ..... 194

#### 6.2.4 $\mu\text{C/OS-II}$ 多任务实现

机制 ..... 195

#### 6.3 $\mu\text{C/OS-II}$ 在 STM32F103

处理器上的移植 .....

6.3.1 移植条件 ..... 196

6.3.2 移植步骤 ..... 197

### 第 7 章 $\mu\text{C/OS-II}$ 的内核机制 ..... 203

#### 7.1 $\mu\text{C/OS-II}$ 的内核结构 .....

7.1.1 临界段 .....

7.1.2 任务 .....

7.1.3 任务状态 .....

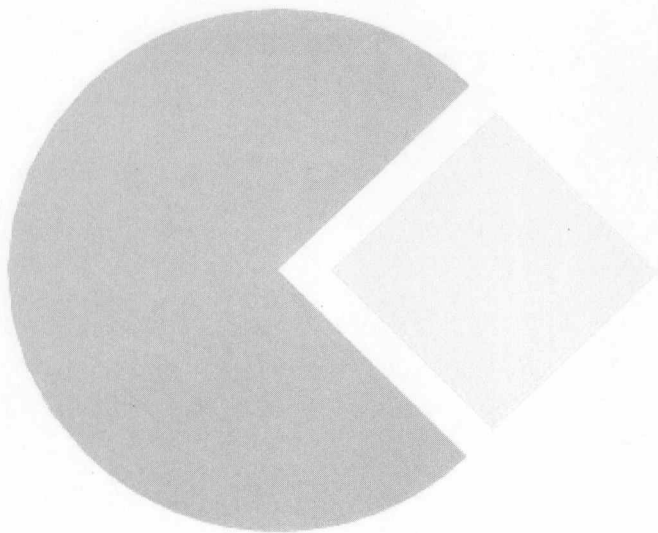
7.1.4 任务调度 .....

|       |                     |     |
|-------|---------------------|-----|
| 7.1.5 | 中断处理                | 208 |
| 7.1.6 | $\mu$ C/OS-II 初始化   | 209 |
| 7.1.7 | $\mu$ C/OS-II 启动    | 209 |
| 7.2   | $\mu$ C/OS-II 的任务管理 | 211 |
| 7.2.1 | 任务建立                | 211 |
| 7.2.2 | 任务堆栈                | 214 |
| 7.2.3 | 堆栈检验                | 214 |
| 7.2.4 | 改变任务优先级             | 217 |
| 7.2.5 | 挂起任务                | 220 |
| 7.2.6 | 恢复任务                | 222 |
| 7.2.7 | 删除任务                | 224 |
| 7.2.8 | 请求删除任务              | 227 |
| 7.3   | $\mu$ C/OS-II 的时间管理 | 229 |
| 7.3.1 | 任务延时                | 230 |
| 7.3.2 | 时/分/秒延时函数           | 231 |
| 7.3.3 | 恢复延时任务              | 233 |
| 7.3.4 | 系统时间                | 235 |
| 7.4   | 任务之间通信与同步           | 236 |
| 7.4.1 | 事件控制块 ECB           | 237 |
| 7.4.2 | 初始化任务控制块            | 240 |
| 7.4.3 | 使任务进入就绪态            | 240 |
| 7.4.4 | 使任务进入等待某事件发生状态      | 243 |
| 7.4.5 | 由于等待超时而将任务置为就绪态     | 243 |
| 7.4.6 | 信号量                 | 244 |
| 7.4.7 | 邮箱                  | 252 |
| 7.4.8 | 消息队列                | 261 |

|       |        |     |
|-------|--------|-----|
| 7.5   | 内存管理   | 274 |
| 7.5.1 | 内存控制块  | 274 |
| 7.5.2 | 建立内存分区 | 275 |
| 7.5.3 | 分配内存块  | 278 |
| 7.5.4 | 释放内存块  | 279 |
| 7.5.5 | 等待内存块  | 280 |

## 第 8 章 基于 $\mu$ C/OS-II 的程序设计

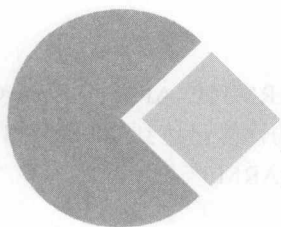
|                       |     |
|-----------------------|-----|
| 实例                    | 283 |
| 8.1 实例介绍              | 283 |
| 8.2 实例分析              | 283 |
| 8.2.1 实例任务划分          | 283 |
| 8.2.2 实例任务设计与优先级分配    | 284 |
| 8.3 任务实现详解            | 285 |
| 8.3.1 键盘任务            | 286 |
| 8.3.2 显示任务            | 293 |
| 8.3.3 使用延时函数的采样任务     | 296 |
| 8.3.4 使用时钟节拍钩子函数的采样任务 | 297 |
| 8.3.5 使用定时中断的采样任务     | 299 |
| 8.3.6 使用高优先级中断的采样任务   | 301 |
| 8.3.7 串口发送任务          | 302 |
| 参考文献                  | 304 |



# 第 1 部分

- 第 1 章 走进 ARM 微处理器
- 第 2 章 Cortex-M3 处理器体系结构
- 第 3 章 ARM 开发环境 RealView MDK 平台搭建
- 第 4 章 基于 Realview 开发环境的嵌入式软件开发
- 第 5 章 STM32F103 处理器内部资源 C 编程与实例





## 第1章 走进 ARM 微处理器

第一片 ARM 处理器的开发距今已近 20 年了,在这 20 年中它已经发展成为世界领先的 32 位嵌入式处理器核。第一片 ARM 核是由 Acorn Computers Limited 公司在 1983~1985 年开发的。这是一个总部设在剑桥的小型英国公司,当时大约有 400 名职工,处理器是使用很简单的设计工具设计的。行为模型是以 BASIC 写的寄存器传输级描述,而且需用手工转换成逻辑图。

ARM 公司本身不生产芯片,靠转让设计许可由合作伙伴公司来生产各具特色的芯片。ARM 商业模式的强大之处在于其价格合理,在全世界范围的合作伙伴超过 100 个,其中包括许多半导体的著名公司。ARM 公司专注于设计,其内核耗电少、成本低、功能强,特有 16/32 位双指令集。ARM 已成为移动通信、手持计算和多媒体数字消费等嵌入式解决方案的 RISC 实际标准。

### 1.1 ARM 体系结构的源头——ARM 公司

1990 年 11 月,ARM 成立于英国,原名 Advanced RISC Machine 有限公司,是苹果电脑、Acorn 电脑集团和 VLSI Technology 的合资企业。Acorn 曾推出世界首个商用单芯片 RISC 处理器,而苹果电脑当时希望将 RISC 技术应用于自身系统,ARM 的微处理器新标准因此应运而生。ARM 成功地研制了首个低成本 RISC 架构,迅速在市场上崭露头角。与此同时 RISC 结构的竞争对手都着眼于提高性能,发展高端工作站处理器的 RISC 结构。

1991 年 ARM 推出首颗嵌入式 RISC 核心——ARM6™系列处理器,不久 VLSI 率先获得授权,一年后夏普和 GEC Plessey 也成为授权用户。1993 年德州仪器和 Cirrus Logic 亦签署了授权协议,从此 ARM 的知识产权产品和授权用户都急剧扩大。1993 年 Nippon Investment and Finance (NIF) 成为 ARM 股东后,ARM 开始向全球拓展,分别在亚洲和欧洲等设立了办事处。1998 年 4 月 ARM 在伦敦证券交易所和纳斯达克交易所上市。

现如今的 ARM 公司, 已发展成为一家全球性大公司, 在 3 大洲 8 个国家设有分支机构, ARM 中国—安谋咨询上海有限公司也在 2002 年 7 月在中国上海成立。

## 1.2 ARM 处理器的发展历程

ARM 处理器当前有 7 个产品系列: ARM7、ARM9、ARM9E、ARM10E、ARM11、SecurCore 和 Cortex 系列。其中 Cortex 系列为最近推出的产品, Cortex A8 的性能已达到 2000MIPS。进一步产品来自于合作伙伴, 例如 Intel Xscale 微体系结构和 StrongARM 产品。

根据适用的领域不同, ARM 处理器分为 3 类。

- 嵌入式实时系统应用处理器: 主要应用于对实时性要求较高的系统, 如网络存储、自动化控制、工业监控等领域。
- 应用系统平台处理器: 该系列处理器常和操作系统相结合, 应用于消费电子、音视频处理等对计算性能要求较高的领域。
- 安全应用系列处理器: 该系列处理器主要应用于智能卡、SIM 卡、缴费终端等领域。SecurCore 系列是专门为安全设备而设计的。

表 1.1 所示为各系列所包含的不同处理器及各处理器的特点。

表 1.1 各系列所包含的不同处理器及其特点

|                    | Cache 大小  | 是否存在紧耦合内存 | 内存管理                  | 总线类型               | 是否支持 Thumb | 是否支持 DSP | 是否支持 Jazelle |
|--------------------|-----------|-----------|-----------------------|--------------------|------------|----------|--------------|
| 应用系列处理器            |           |           |                       |                    |            |          |              |
| Cortex-A8          | 可配        | -         | MMU + Trust Zone      | AMBA 3 AXI         | 是          | 是        | 是            |
| Cortex-A9 (MPCore) | 16KB/64KB |           | MMU + Trust Zone      | AMBA 3 AXI         | 是          | 是        | 是            |
| Cortex-A9 (单核)     | 16KB/64KB |           | MMU + Trust Zone      | AMBA 3 AXI         | 是          | 是        | 是            |
| ARM11 MPCore       | 可配        | -         | MMU + cache coherency | 1 × 或 2 × AMBA AXI | 是          | 是        | 是            |
| ARM1136J (F) S     | 可配        | 是         | MMU                   | 5 × AHB            | 是          | 是        | 是            |
| ARM1136J Z (F) S   | 可配        | 是         | MMU + Trust Zone      | 4 × AHB            | 是          | 是        | 是            |
| ARM720T            |           |           |                       |                    | 否          | 否        | 否            |

续表

|                  | Cache 大小  | 是否存在紧耦合内存 | 内存管理 | 总线类型                | 是否支持 Thumb | 是否支持 DSP | 是否支持 Jazelle |
|------------------|-----------|-----------|------|---------------------|------------|----------|--------------|
| ARM920T          | 16KB/16KB | -         | MMU  | AHB                 | 是          | 否        | 否            |
| ARM922T          | 8KB/8KB   | -         | MMU  | AHB                 | 是          | 否        | 否            |
| ARM926EJ-S       | 可配        | 是         | MMU  | 2 × AHB             | 是          | 是        | 是            |
| 嵌入式处理器           |           |           |      |                     |            |          |              |
| Cortex-M0        |           |           |      |                     | 是          | 否        | 否            |
| Cortex-M1        |           | 是         |      | AMBA AHB Lite + APB | 是          | 否        | 否            |
| Cortex-M3        | -         | -         | MPU  | 3 × AHB Lite + APB  | 是          | 否        | 否            |
| Cortex R4        | 0KB~64KB  | 可配        | MPU  | AMBA 3 AXI          | 是          | 是        | 否            |
| ARM1156T2 (F) -S | 可配        | 是         | MPU  | 3 × AXI             | 是          | 是        | 否            |
| ARM7EJ-S         | -         | -         | -    | 是                   | 是          | 是        | 是            |
| ARM7TDMI         | -         | -         | -    | 是                   | 是          | 否        | 否            |
| ARM7TDMI-S       | -         | -         | -    | 是                   | 是          | 否        | 否            |
| ARM946E-S        | 可配        | 是         | MPU  | AHB                 | 是          | 是        | 否            |
| ARM966E-S        | -         | 是         | -    | AHB                 | 是          | 是        | 否            |
| ARM968E-S        | n/a       | 是         | DMA  | AHB-Lite            | 是          | 是        | 否            |
| 安全系列处理器          |           |           |      |                     |            |          |              |
| SC300            |           |           | MPU  |                     | 是          | 否        | 否            |
| SC100            |           |           | MPU  |                     | 是          | 否        | 否            |
| SC200            |           |           | MPU  |                     | 是          | 是        | 是            |

### 1.3 ARM 体系结构

ARM 体系结构的发展经历了下面几个关键时期。

- ARMv4: 目前 ARM 支持的最老版本的体系结构, 它只支持 32 位的 ARM 指令, 在这之前的体系结构已经被 ARM 公司废除。这一体系结构的处理器包括 ARM<sup>TM</sup> 和 Intel 的 StrongARM。

- ARMv4T: ARMv4T 结构增加了 Thumb 指令支持, 它继承了 ARMv4 结构的所有优点, 并且可以产生密度更高的 Thumb 代码。

- ARMv5TE: 1999 年, ARM 发布了带有 DSP 指令的 ARMv5TE 架构。该结构不但在原有架构上增加了新的 Thumb 指令, 并且在 ARM 架构中引入了 DSP 处理, 增加了 ARM 在多媒体市场的竞争力。

- ARMv5TEJ: 2000 年, ARM 发布的 ARMv5TEJ 架构引入了具有 Java 处理能力的 Jazelle 技术, 摆脱了 ARM 对软 Java 虚拟机的依赖, 为 ARM 进军手持移动设备奠定了基础。

- ARMv6: ARMv6 体系结构正式发布于 2001 年, 该架构不但支持单指令多数据处理 (SIMD, Single Instruction Multiple Data) 指令, 而且还引入了 Thumb-2 和 TrustZone 技术。该架构第一款成熟处理器为 ARM1136J(F)-S, 2002 年正式上市, 随后 ARM 公司于 2003 年相继发布了 ARM1156T2(F)-S 和 ARM1176JZ(F)-S 处理器。

- ARMv7: ARMv7 处理器以全新的 Cortex 命名问世, 并且根据应用的领域不同, 分为 3 个独立方向同时发展——支持虚拟内存管理的 A 系列, 具有更高实时性的 R 系列和针对微控制领域和低端应用的 M 系列。

图 1.1 所示为 ARM 体系结构的发展过程。

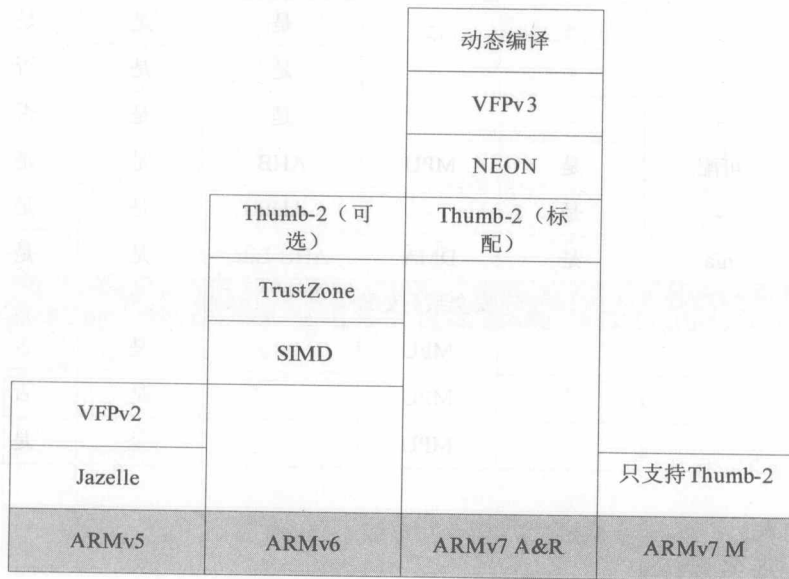


图 1.1 ARM 体系结构的发展过程

### 1.3.1 ARM Cortex 处理器技术特点

ARM Cortex 系列处理器采用全新的 ARM v7 架构, 共分为 Cortex-M、Cortex-R 和 Cortex-A3 类。

ARM v7 架构是在 ARM v6 架构的基础上衍生而来的。该架构采用了 32-bit 指令和 16-bit 指令相混合的 Thumb-2 技术,该技术是在 ARM 的 Thumb 代码压缩技术的基础上发展起来的,并且保证了对现存 ARM 解决方案的完全兼容。Thumb-2 技术比纯 32 位代码少使用 31% 的内存,减小了系统开销。同时能够提供比已有的基于 Thumb 技术的解决方案高出 38% 的性能。

ARM v7 架构还采用了 NEON 技术,将 DSP 和媒体处理能力提高了近 4 倍,并支持改良的浮点运算,满足下一代 3D 图形、游戏物理应用以及传统嵌入式控制应用的需求。

此外,ARM v7 架构还支持改良的运行环境,以迎合不断增加的 JIT (Just In Time 实时生产) 和 DAC (Dynamic Adaptive Compilation, 动态自适应编译) 技术的使用。

同时,ARM v7 架构在设计时充分考虑到了与早期的 ARM 处理器的软件兼容性。ARM Cortex-M 系列支持的 Thumb-2 指令集 (Thumb 指令集的扩展集),完全兼容为 ARMv7 体系结构之前的处理器编写的代码。ARM Cortex-M 系列处理器通过一个前向的转换方式与 ARM Cortex-R 系列微处理器完全兼容。ARM Cortex-M 系列系统代码 (如实时操作系统) 可以很容易地移植到基于 ARM Cortex-R 系列的系统上。ARM Cortex-A 和 Cortex-R 系列处理器还支持 ARM 32 位指令集,向后完全兼容早期的 ARM 处理器,包括从 1995 年发布的 ARM7TDMI 处理器到 2002 年发布的 ARMII 处理器系列。

### 1.3.2 ARM Cortex 系列处理器比较

#### 1. ARM Cortex-M 处理器技术特点

ARM Cortex-M3 处理器是为存储器和处理器的尺寸对产品成本影响极大的各种应用专门开发设计的,其结构如图 1.2 所示。它整合了多种技术,减少使用内存,并在极小的 RISC 内核上提供低功耗和高性能。ARM Cortex-M3 处理器是使用门数最少的 ARM CPU,相对于过去的设计大大减小了芯片面积,可减小装置的体积或采用更低成本的工艺进行生产。仅 33 000 门的内核,其性能可达 1.2 DMIPS/MHz。此外,基本系统外设还具备高度集成化的特点,集成了许多紧耦合系统外设,合理利用了芯片空间,使系统满足了下一代产品的控制需求。

ARM Cortex-M3 处理器不仅使用了先进的哈佛微架构,执行 32 位的 Thumb-2 指令集,同时包含高效的系统外设——Nested Vectored Interrupt Controller (NVIC) 和 Arbiter 总线。Cortex-M3 处理器的技术方案在测试和实例应用中表现出较高的性能:在台机电 180nm 工艺下,芯片性能达 1.2 DMIPS/MHz,时钟频率高达 100MHz。Cortex-M3 处理器还实现了 Tail-Chaining 中断技术。该技术是一项完全基于硬件的中断处理技术,最多可减少 12 个时钟周期数,在实际应用中可减少 70% 的中断处理时间。同时,该架构还推出了新的单线调试技术,避免使用多引脚进行 JTAG 调试,并全面支持 RealView 编译器和 RealView 调试产品。先进的 Realview 工具向设计者提供模拟、创建虚拟模型、编译软件、调试、验证和测试等基于 ARM v7 架构的系统功能。

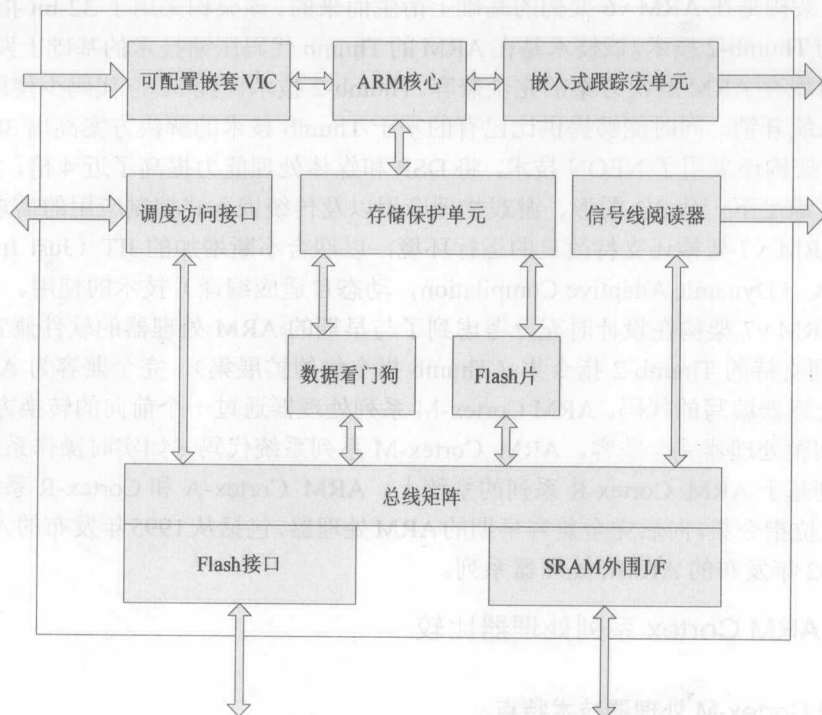


图 1.2 ARM Cortex-M3 处理器结构

综上所述，为微控制器应用而开发的 Cortex-M3 系列处理器与其他 ARM 处理器相比具有以下特点。

- 实现单周期 Flash 应用最优化。
- 准确快速的中断处理。中断处理不超过 12 周期，使用末尾连锁（Tail-chaining）机制仅用 6 个周期。
- 拥有低功耗时钟门控（Clock Gating）的 3 种睡眠模式。
- 具有单周期乘法和乘法累加指令。
- Thumb-2 混合的 16/32 位固有指令集，无模式切换。
- 包括数据观察点和 Flash 补丁在内的高级调试功能。
- 可以使用原子位操作。在一个单一指令中读取/修改/编写数据某一位。
- 1.25DMIPS/MHz 的处理速度（ARM7 为 0.9DMIPS/MHz，而 ARM9 为 1.1DMIPS/MHz），使其在低功耗的处理器市场上占有一席之地。

## 2. ARM Cortex-R 处理器技术特点

ARM Cortex-R 系列处理器目前包括 ARM Cortex-R4 和 ARM Cortex-R4F 两个型号，该