源程序下载地址：
http://www.buaapress.com.cn的“下载专区”

STM32F207高性能网络型MCU
嵌入式系统设计

廖义奎　编著

北京航空航天大学出版社
BEIHANG UNIVERSITY PRESS

STM32F207 高性能网络型 MCU 嵌入式系统设计

廖义奎　编著

北京航空航天大学出版社

内 容 简 介

本书介绍了 STM32F2 系列处理器的特点与应用，共 16 章，分别讲解 ARM Cortex 处理器概述、从 STM32F1 到 STM32F2 的硬件兼容性设计、从 STM32F1 到 STM32F2 的程序设计、STM32F2 固件库的使用、STM32F2 的启动原理及时钟控制、STM32F2 新增的 FSMC 接口及 LCD 屏控制、STM32F2 新增的日历功能及应用、STM32F2 中断及 SysTick 应用、STM32F2 增强的闹钟、时间戳与篡改检测、STM32F2 增强的定时器、STM32F2 新增的 ETH 以太网接口及 LwIP 应用、STM32F2 新增的 DCMI 数码相机接口及应用、STM32F2 增强的 USART 接口与应用、STM32F2 增强的 ADC 模块及应用、一步一步设计自己的嵌入式操作系统、一步一步设计自己的嵌入式 GUI 库。

本书配套资料中附有所有章节的源程序。本书适合于嵌入式开发人员作为开发参考资料，也适合于高校师生作为单片机、嵌入式系统课程的教材和教学参考书。

图书在版编目(CIP)数据

STM32F207 高性能网络型 MCU 嵌入式系统设计 / 廖义奎编著. -- 北京：北京航空航天大学出版社，2012.9
ISBN 978-7-5124-0921-7

Ⅰ.①S… Ⅱ.①廖… Ⅲ.①微处理器—系统设计
Ⅳ.①TP332

中国版本图书馆 CIP 数据核字(2012)第 198069 号

STM32F207 高性能网络型 MCU 嵌入式系统设计

廖义奎 编著

责任编辑 沈韶华

*

北京航空航天大学出版社出版发行

北京市海淀区学院路 37 号(邮编 100191) http://www.buaapress.com.cn

发行部电话：(010)82317024 传真：(010)82328026

读者信箱：emsbook@gmail.com 邮购电话：(010)82316936

涿州市新华印刷有限公司印装 各地书店经销

*

开本：710×1 000 1/16 印张：31 字数：679 千字

2012 年 9 月第 1 版 2012 年 9 月第 1 次印刷 印数：4 000 册

ISBN 978-7-5124-0921-7 定价：59.00 元

前　言

1. 从 STM32F1 到 STM32F2

意法半导体于 2011 年下半年开始量产 STM32F2 系列处理器，采用最新的 90 nm工艺生产新一代的 STM32 产品。90 nm 工艺带来更高的效率，极致发挥了 Cortex－M3 的性能，具有更低的动态功耗、更多灵活创新的高性能外设及更高的集成度。

- 极致的运行速度表现在以 120 MHz 高速运行时可达到 150 DMIPS 的处理能力，自适应实时闪存加速器使得 STM32F2 可以在片内闪存上以 120 MHz 的高速零等待地执行代码。
- 更多的存储空间表现在高达 1 MB 的片上闪存，高达 128 KB 的内嵌 SRAM。具有灵活的高速外部存储器接口 FSMC，用于扩展片外存储器和外设。

2. 注重分析 STM32F2 新增功能的应用

STM32F2 新增的日历、FSMC LCD 接口、ETH 以太网接口、DCMI 视频接口等功能模块是其优势，也是其重点应用目标。因此，介绍这些功能模块的特点与应用方法是本书的主要目标。

3. 硬件与软件结合

硬件与软件是嵌入式系统开发中不可缺少的两个组成部分，缺一不可。本书详细介绍了 STM32F1 与 STM32F2 之间的硬件兼容性设计。

由于 STM32F2 与 STM32F1 引脚上基本兼容，因此在设计时可以采用与 STM32F1 兼容的方案，其优点是在不花额外代价的情况下，一次设计可同时获得 STM32F2 和 STM32F1 两种 PCB 板。另外，采用兼容的设计方案，还有利于在设计过程中，充分利用原来已经熟悉和成熟的 STM32F1 电路作为参考，可以更加快速和可靠地进行设计。

4. GCC 编译器

GCC 编译器是一套以 GPL 及 LGPL 许可证发行的开源、自由软件。GCC 编译器是移植到中央微控制器架构以及操作系统最多的编译器。由于 GCC 已成为 GNU 系统的官方编译器（包括 GNU/Linux 家族），它也成为编译与建立其他操作系统的主要编译器，包括 Linux 系列、BSD 系列、Mac OS X、NeXTSTEP 与 BeOS 等。

GCC 通常是跨平台软件首选的编译器。有别于一般局限于特定系统与执行环境的编译器，GCC 在所有平台上都使用同一个前端处理程序，产生一样的中间代码，此中间代码在各个不同的平台上都一致，并可输出正确无误的最终代码。

GCC 功能强大、性能优越，并且开放源代码，可以免费使用从而降低开发成本。因此，GCC 是本书重点介绍的内容，也是书中例子所用的编译器。

5. 以 C++面向对象的思路编程

本书大部分程序以 C++面向对象的思路进行编程，具有分类与封装、隐藏、权限、继承与代码重用、接口编程等特点。

(1) 分类与封装

以往的单片机程序通常采用 C 语言来编写，常遇到代码宏定义过多、函数名繁杂、代码杂乱等问题。例如，把一大堆代码放在一起，这些代码既没归类也没级别，致使整个程序结构变得既复杂又混乱。

采用 C++的方式很容易解决这一问题，在 C++里用这些函数进行封装的方法就是定义一个类，C++中的类名就相当于给封装在一起的函数和变量起的名字。同时，解决了多个函数共用非全局变量的问题。

(2) 隐　藏

在单片机程序中常遇到这样的问题，就是有些端口是输入端口，有些是输出端口，那么输入端口就不应该执行向端口写入数据的操作。但是，在 C 语言程序中，无法从语言规则上限制这样的操作，也就是不管用户程序中直接写入寄存器还是调用写入函数，程序在编译时都不会报错，因为编译器并不知道输入端口不能写入。要限制这样的操作，只能由程序编写人员自己把握，但这样经常会出现失误。在 C++中，可以利用类的隐藏和屏蔽等特点来解决这类问题。

(3) 权　限

在单片机程序中常遇到这样的情况，当修改某个变量的值时，需要同时修改另一个变量的值，或者同时需要调用另一个函数来处理某些操作。例如某个程序需完成这样的功能：在修改程序中代表系统时间是多少秒钟的变量时，如果修改的秒数大于 60，这时应该同时需要两种附加的办法来解决大于 60 的问题，而不能直接保留这个大于 60 的数值在秒变量之中。上述问题，可以采用封装一个时间类，把秒变量封装为私有成员，不允许类外直接操作，再提供一个公有的成员函数来变更秒时间。在该成员函数之中可以包括上述两种附加秒处理方式之中的一种，程序其他地方修改秒时间时，只能通过该成员函数来修改而不能直接修改秒变量，这样就不会出现秒变量超过 60 的错误了。这种方法即是程序中操作权限的设置方法。

(4) 继承与代码重用

提高单片机程序代码的重用率，是编程人员一个共同的追求。在 C++中，可以通过类的继承等方式来实现。例如可以把端口的操作封装到 CGpio 类中，对于所需要的 LED 操作和 Key 操作是两个不同种类的端口操作。对于 LED，所需要的是执行 On 操作和 Off 操作。对于 Key，所需要的是读取当前是 isUp 状态还是 isDown 状态。因此，采用 CLed 和 CKey 这两个独立的类来封装会更加合适。由于 CLed 和 CKey 都属于端口操作之中的一种，所以都具有 CGpio 类所有的特征，让 CLed 和 CKey 都重新实现一次 CGpio 类的功能显然没必要，并且会出现大量的重

复代码。因此，采用继承的方式更加合适，让 CLed 和 CKey 都继承于 CGpio 类，这样它们就具有了 CGpio 类所有的特征了。

(5) 接口编程

接口编程是嵌入式系统中非常重要的组成部分，特别是在编写驱动程序、嵌入式操作系统、通用程序库等之中，都需要进行接口编程。在 C 语言中，也可以提供接口编程方式，一般是通过 struct 结构来实现。struct 结构相当于 C++中的一个类，但只能提供定义公有类型（public）成员的特殊类，不能像C++、JAVA 那么严格地实现接口编程方式。

6. 尽量全面介绍 STM32F2 的特点与应用

① 本书重点介绍 STM32F2 新增功能，包括日历功能及应用、FSMC 接口及 LCD 屏控制、ETH 以太网接口及 LwIP 应用、DCMI 数码相机接口及应用。

② 详细介绍了 STM32F2 增强的功能，包括闹钟时间戳与篡改检测、定时器、USART 接口与应用、ADC 模块及应用。

③ 详细介绍了 STM32F1 与 STM32F2 兼容性设计，包括从 STM32F1 到 STM32F2 的硬件兼容性设计、从 STM32F1 到 STM32F2 的程序设计。

④ 详细介绍了 STM32F2 的基本编程原理，包括 STM32F2 固件库的使用、STM32F2 的启动原理及时钟控制、STM32F2 中断及 SysTick 应用。

⑤ 详细介绍了 STM32F2 的扩展应用，包括嵌入式操作系统基础、嵌入式 GUI 设计与应用。

另外，还用一整章详细介绍了各种常见的新型 ARM Cortex 处理器的特点与功能。

7. STM32F2 新增的 ETH 以太网接口及 LwIP 应用

在以嵌入式操作系统为核心的网络应用系统和网络设备中，微控制器一般可以选择 ARM9、ARM11、Cortex－A8、Cortex－A9 等，其特点一是内部带有以太网 MAC 控制器；二是运行速度快，一般主频都在 400 MHz 以上，可满足高速的通信数据处理要求；三是带有 MMC 单元，可运行功能强大的 Windows CE、Linux 等嵌入式操作系统，既方便实现复杂任务的管理，又带有完善的网络支持。

在面向快速网络应用系统和网络设备中，例如视频监控系统，要求提供一种低成本、低功耗、高速的网络通信环境，微控制器一般可以选择内部带有以太网 MAC 控制器的 Cortex－M3、Cortext－M4 微控制器（例如 STM32F107 系列、STM32F207 系列、STM32F407 系列等），内部集成了以太网 10/100 MB MAC 模块，支持 10/100 MB 自适应网络应用。

在面向慢速网络应用系统和网络设备中，对数据传输的速度要求不高，通常只需要完成现场传感器数据采集与传输、远程设备控制等功能时，这时可选择内部不带以太网 MAC 控制器的微控制器，并使用外加一个专用的以太网模块来实现。这些模块常见的有 ENC28J60、CP2200、ENC28J60、W5100 等，具有成本低、接口简单、使用方便等特点。

第 11 章介绍了 STM32F2 新增的 ETH 以太网接口的功能与特点，以及基于 LwIP 协议栈的以太网应用程序开发。

8. STM32F2 新增的 DCMI 视频接口及应用

摄像头与图像采集已经成为手持电子产品中的标准配置。STM32F2 新增的 DCMI 数码相机接口是一个能够接收高速数据流接口，支持 8、10、12 或 14 位的 CMOS 摄像头模块的同步并行数据。它支持 YCbCr4：2：2、RGB565、逐行扫描视频和压缩数据（JPEG）等数据格式。

OV7670 图像传感器具有体积小、工作电压低等特点，提供单片 VGA 摄像头和影像微控制器的所有功能。通过 SCCB 总线控制可以输出整帧、子采样、取窗口等方式的各种分辨率 8 位影像数据。

STM32F2 新增的 DCMI 视频接口及 OV7670 摄像头应用开发在本书第 12 章有详细的介绍。

9. 如何使用本书

本书的读者需要有一定的 C/C++、单片机以及电子线路设计基础。本书适合于从事 ARM 嵌入式开发的工程人员以及 STM32 的初学者；也适合于原来从事 8 位、16 位 MCU 开发，而又需要跨到 32 位 MCU 平台的研发人员；同时也适合于高校师生作为课程设计、毕业设计以及电子设计竞赛的培训和指导教材；也适合于作为本、专科单片机、嵌入式系统相关课程的教材或实验指导书。

配套资料中包括了所有章节的程序代码，读者可以直接从北航出版社网站（www. buaapress. com. cn）的“下载专区”下载使用。

如果读者在使用本书时遇到相关技术问题，或者对本书介绍的 ARM 开发板感兴趣或有疑问，可以通过电子邮件与作者联系（javawebstudio@163. com），作者将尽最大努力与读者共同解决学习过程中和开发过程中遇到的问题，共同进步。

10. 致　谢

本书的编写过程中，陆才志、苏宇、梁创英、许金、玉黄荣、王继平、韦运忠、徐卫怡、蓝艺峥、苏金秀分别审阅了本书全部或部分章节，在此表示衷心感谢。

本书在编写过程中参考了大量的文献资料，一些资料来自互联网和一些非正式出版物，书后的参考文献无法一一列举，在此对原作者表示诚挚的谢意。

限于作者水平，并且编写时间比较仓促，书中难免存在错误和疏漏之处，敬请读者批评指正。

编　者

2012. 7

目 录

第 1 章 ARM Cortex 处理器概述 …… 1

1.1 ARM 处理器分类 …… 1

1.2 ARM Cortex 处理器 …… 2

1.3 Cirtex-M0 处理器 …… 3

1.3.1 概述 …… 3

1.3.2 NUC100 系列处理器 …… 4

1.3.3 NXP Cortex-M0 处理器 …… 5

1.4 Cortex-M1 处理器 …… 7

1.4.1 概述 …… 7

1.4.2 Cortex-M1 应用 …… 8

1.5 Coretx-M3 处理器 …… 9

1.5.1 概述 …… 9

1.5.2 AT91SAM3U 系列处理器 …… 10

1.5.3 LPC1800 系列处理器 …… 11

1.6 Coretx-M4 处理器 …… 13

1.6.1 概述 …… 13

1.6.2 Kinetis 系列处理器 …… 14

1.6.3 LPC4300 系列处理器 …… 16

1.6.4 STM32F4 系列处理器 …… 19

1.7 Cortex-A8 处理器 …… 21

1.8 Cortex-A9 处理器 …… 21

1.9 Cortex-A15 处理器 …… 24

1.9.1 Cortex-A15 内核简介 …… 24

1.9.2 OMAP 5 处理器 …… 25

第 2 章 从 STM32F1 到 STM32F2 的硬件兼容性设计 …… 29

2.1 STM32F1 及 STM32F2 系列处理器 …… 29

2.1.1 STM32F1 系列处理器 …… 29

2.1.2 STM32F2 系列处理器 …… 30

2.1.3 STM32F1 与 STM32F2 的区别 …… 33

2.2 STM32F1 与 STM32F2 之间的兼容性设计 …… 34
2.3 STM32F207 最小系统设计 …… 37
2.3.1 最小系统电路设计 …… 37
2.3.2 电源电路设计 …… 42
2.3.3 按键与 LED 电路设计 …… 47
2.3.4 时钟、复位、引导配置以及 SWD 接口电路设计 …… 49
2.3.5 通信接口电路设计 …… 53
2.3.6 其他外设电路设计 …… 60
2.4 图像传感器及接口 …… 62
2.4.1 图像传感器 …… 62
2.4.2 OV7670 摄像头 …… 63
2.4.3 CMOS 摄像头接口 …… 65
2.5 以太网接口 …… 66
2.5.1 STM32F2 以太网模块介绍 …… 66
2.5.2 SMI、MII 和 RMII 接口 …… 66
2.5.3 STM32F207 以太网接口电路设计 …… 71
2.6 引脚安排汇总 …… 73
第 3 章 从 STM32F1 到 STM32F2 的程序设计 …… 78
3.1 从 STM32F1 到 STM32F2 …… 78
3.1.1 STM32F1 与 STM32F2 在开发工具版本上的差别 …… 78
3.1.2 STM32F1 与 STM32F2 系列的 IP 总线之间的映射差异 …… 78
3.1.3 STM32F1 和 STM32F2 AHB/APB 桥时钟差异 …… 81
3.1.4 STM32F1 和 STM32F2 在寄存器上的差别 …… 84
3.1.5 STM32F1 和 STM32F2 在 GPIO 上的差异 …… 86
3.2 基于 Keil 的第一个 STM32F207 程序 …… 89
3.2.1 创建一个 Keil 新项目 …… 89
3.2.2 添加主程序 …… 92
3.2.3 配置 Flash Download …… 93
3.2.4 在 RealView MDK 中调试程序 …… 94
3.2.5 与 STM32F1 的比较 …… 95
3.3 第一个基于 GCC 的 STM32F207 程序 …… 98
3.3.1 软件环境 …… 98
3.3.2 编写 STM32 的 C 语言程序 …… 100
3.3.3 用 GCC 编译 STM32 程序 …… 104
3.3.4 在 Obtain_Studio 中编译 Hello World 程序 …… 105

3.4 使用 C++开发 STM32F2 程序 …… 106
3.5 位操作方式 …… 107
第 4 章 STM32F2 固件库的使用 …… 110
4.1 STM32F2xx 标准外设库 …… 110
4.1.1 STM32F2xx 标准外设库结构 …… 110
4.1.2 如何使用标准外设库 …… 121
4.2 在 RealView MDK 中使用 STM32 固件库 …… 123
4.2.1 STM32 固件库应用 …… 123
4.2.2 STM32 固件库应用程序分析 …… 125
4.3 在 GCC 中应用 STM32 固件库 …… 133
4.3.1 STM32F2 固件库 GCC 项目模板 …… 133
4.3.2 Obtain_Studio 集成开发系统常用技巧 …… 137
第 5 章 STM32F2 的启动原理及时钟控制 …… 141
5.1 STM32F2 启动原理 …… 141
5.1.1 STM32F2 启动过程分析 …… 141
5.1.2 STM32F2 物理重新映射 …… 143
5.1.3 STM32 软件复位与功耗控制 …… 144
5.2 STM32F2 时钟控制(RCC) …… 147
5.2.1 STM32F2 时钟树 …… 147
5.2.2 F2 与 F1 系列 RCC 主要区别 …… 149
5.2.3 RCC PLL 配置寄存器与 RCC 时钟配置寄存器 …… 153
5.2.4 采用 STM32F2xx-RevA-Z_Clock_Configuration 进行时钟配置 …… 159
5.3 RCC 的应用 …… 159
5.3.1 RCC 的配置方法 …… 159
5.3.2 STM32F2 固件库中的时钟初始化的实现 …… 161
5.3.3 在主程序中调用 STM32F2 固件库时钟初始化函数 …… 164
5.4 系统配置控制器(SYSCFG) …… 165
第 6 章 STM32F2 新增的 FSMC 接口及 LCD 屏控制 …… 169
6.1 STM32F2 新增的 FSMC 接口 …… 169
6.1.1 STM32F1 与 STM32F2 的 FSMC 接口比较 …… 169
6.1.2 AHB 总线接口 …… 171
6.2 LCD 驱动芯片 …… 171
6.2.1 LCD 接口 …… 171
6.2.2 Ili9xxx 系列 TFT 驱动芯片 …… 172
6.3 基于 FSMC 的 TFT 驱动程序设计 …… 177

6.3.1 FSMC 与 TFT 端口连接与端口映射 …… 177
6.3.2 FSMC 与 TFT 的内存空间映射与操作 …… 179
6.3.3 FSMC 初始化 …… 182
6.3.4 TFT 初始化 …… 187
6.3.5 TFT 基本显示函数的实现 …… 190
第 7 章 STM32F2 新增的日历功能及应用 …… 195
7.1 STM32F2 实时时钟 …… 195
7.1.1 RTC 简介 …… 195
7.1.2 STM32F2 与 STM32F1 在 RTC 上的区别 …… 195
7.1.3 STM32F2 实时时钟结构 …… 196
7.1.4 STM32F2 实时时钟固件库 …… 197
7.2 日历功能测试程序 …… 200
7.2.1 日历功能测试程序 …… 200
7.2.2 日历时钟源 …… 201
7.2.3 日历配置 …… 204
7.2.4 日历值的写入与读取 …… 209
第 8 章 STM32F2 中断及 SysTick 应用 …… 213
8.1 STM32F2 中断 …… 213
8.2 STM32F2 用户程序中断向量表 …… 216
8.3 SysTick 时钟及中断处理 …… 226
8.3.1 关于 SysTick …… 226
8.3.2 SysTick 测试程序 …… 228
8.3.3 SysTick 程序分析 …… 230
8.4 STM32F2 中断向量管理器 …… 234
8.4.1 NVIC 嵌套中断向量控制器 …… 234
8.4.2 深入了解 STM32F2 的 NVIC 优先级 …… 238
第 9 章 STM32F2 增强的闹钟、时间戳与篡改检测 …… 242
9.1 STM32F2 闹钟功能 …… 242
9.1.1 概述 …… 242
9.1.2 闹钟测试程序的实现 …… 247
9.2 STM32F2 唤醒功能 …… 251
9.2.1 STM32F2 定期唤醒定时器 …… 251
9.2.2 唤醒功能测试程序 …… 255
9.3 时间戳功能 …… 257
9.3.1 概述 …… 257

9.3.2 时间戳测试程序 …… 259
9.4 STM32F2 与 STM32F1 在备份寄存器上的区别 …… 263
9.5 篡改检测 …… 264
9.5.1 概述 …… 264
9.5.2 备份寄存器与篡改检测测试程序 …… 265
9.6 STM32F2 RTC 的数字校准 …… 269
第 10 章 STM32F2 增强的定时器 …… 272
10.1 STM32F2 定时器的种类 …… 272
10.1.1 SysTick 定时器 …… 272
10.1.2 RTC 定时器 …… 272
10.1.3 通用定时器(TIM2～TIM5) …… 273
10.1.4 通用定时器(TIM9～TIM14) …… 273
10.1.5 基本定时器(TIM6、TIM7) …… 274
10.1.6 高级控制定时器(TIM1 及 TIM8) …… 274
10.1.7 独立看门狗 …… 275
10.1.8 窗口看门狗 …… 276
10.2 STM32F2 通用定时器计数模式 …… 277
10.2.1 时基单元 …… 277
10.2.2 计数器模式-向上计数模式 …… 277
10.2.3 计数器模式-向下计数模式 …… 278
10.2.4 计数器模式-中心对齐模式(向上/向下计数) …… 278
10.3 STM32F2 通用定时器基本应用 …… 279
10.4 通用定时器工作模式 …… 283
10.4.1 概述 …… 283
10.4.2 STM32F2 通用定时器模式举例 …… 285
第 11 章 STM32F2 新增的 ETH 以太网接口及 LwIP 应用 …… 289
11.1 STM32F2 与 STM32F1 以太网模块的差异 …… 289
11.2 LwIP …… 292
11.2.1 概述 …… 292
11.2.2 LwIP 主要模块 …… 292
11.2.3 LwIP TCP 协议工作过程 …… 302
11.2.4 LwIP UDP 协议工作过程 …… 303
11.3 LwIP 的移植 …… 305
11.3.1 LwIP 下载 …… 305
11.3.2 LwIP 网络设备驱动程序文件 ethernetif.c …… 306

11.3.3 STM32F207 以太网接口初始化 …… 307
11.4 LwIP 协议栈的 httpserver 测试程序 …… 317
11.5 LwIP 协议栈的 udp_echo_client 测试程序 …… 319
第 12 章 STM32F2 新增的 DCMI 数码相机接口及应用 …… 325
12.1 STM32F2 新增的 DCMI 数码相机接口 …… 325
12.1.1 概述 …… 325
12.1.2 DCMI 的接口 …… 326
12.1.3 DCMI 固件库函数 …… 327
12.2 OV7670 摄像头 …… 329
12.2.1 概述 …… 329
12.2.2 OV7670 工作原理 …… 331
12.3 CMOS 摄像头测试程序 …… 332
12.4 深入 CMOS 摄像头驱动程序原理 …… 335
12.4.1 SCCB 协议 …… 335
12.4.2 SCCB 协议驱动程序设计 …… 337
12.4.3 CMOS 摄像头驱动程序设计 …… 341
第 13 章 STM32F2 增强的 USART 接口与应用 …… 352
13.1 STM32F2 的 USART 接口 …… 352
13.1.1 概述 …… 352
13.1.2 USART 波特率的计算方法 …… 354
13.1.3 发送器 …… 356
13.1.4 接收器 …… 358
13.2 USART 通用串口程序设计 …… 360
13.2.1 USART 固件库函数 …… 360
13.2.2 USART 数据发送与接收程序设计 …… 362
13.2.3 中断方式的数据接收程序设计 …… 365
13.2.4 在 LCD 屏幕上显示 USART 收发数据接 …… 368
第 14 章 STM32F2 增强的 ADC 模块及应用 …… 370
14.1 STM32F2 增强的 ADC 模块 …… 370
14.1.1 概述 …… 370
14.1.2 STM32F2 和 STM32F1 的 ADC 差异 …… 371
14.1.3 STM32F2 的 ADC 固件库函数 …… 374
14.2 STM32 ADC 测试程序 …… 378
14.3 STM32 ADC 程序分析 …… 380

第 15 章 一步一步设计自己的嵌入式操作系统 …… 392
15.1 嵌入式操作系统 …… 392
15.1.1 概述 …… 392
15.1.2 实时操作系统 …… 392
15.1.3 常见的嵌入式操作系统 …… 393
15.2 自己设计一个简单的实时系统 …… 398
15.2.1 操作系统最核心的任务切换 …… 398
15.2.2 实时任务切换基础 …… 403
15.2.3 最简单的操作系统 …… 411
15.2.4 最简单操作系统原理分析 …… 413
15.2.5 为操作系统加上任务休眠功能 …… 419
15.2.6 任务调度策略 …… 422
15.2.7 内存分配技术 …… 426
15.2.8 任务的同步 …… 430
15.2.9 任务间通信 …… 431
15.3 C++实时开源操作系统 scmRTOS …… 432
15.3.1 概述 …… 432
15.3.2 scmRTOS 测试程序 …… 435
15.3.3 把 scmRTOS 应用于前面章节的例子之中 …… 442
第 16 章 一步一步设计自己的嵌入式 GUI 库 …… 445
16.1 嵌入式 GUI …… 445
16.1.1 概述 …… 445
16.1.2 常见的嵌入式 GUI …… 446
16.2 嵌入式 GUI 设计基础 …… 449
16.3 嵌入式 GUI 设计实例 …… 451
16.3.1 最简单的窗口程序 …… 451
16.3.2 嵌入式 GUI 的仿真 …… 453
16.3.3 带消息处理的 GUI 测试程序 …… 456
16.3.4 在 main 函数里处理消息的方式 …… 458
16.4 控件应用程序 …… 459
16.4.1 窗口的控件 …… 459
16.4.2 控件应用程序设计 …… 461
16.5 智能手机桌面风格的应用程序 …… 465
16.6 嵌入式 GUI 底层的设计 …… 469
参考文献 …… 482

第 1 章

ARM Cortex 处理器概述

1.1 ARM 处理器分类

目前的嵌入式系统绝大多数都采用了以 ARM 为内核架构的微处理器。ARM 公司自 1990 年成立以来，一直以一个设计并出售 IP(Intelligence Property)Code 的身份出售其知识产权，其在 32 位 RISC(Reduced Instruction Set Computer)CPU 的开发设计上不断取得了突破，其设计的微处理器结构已经从 v1 发展到了 v7，ARM 处理器的核心及构架如表 1-1 所列。

表 1-1 ARM 处理器的核心及构架

构 架	核 心
v1	ARM1
v2	ARM2
v2a	ARM2As、ARM3
v3	ARM6、ARM600、ARM610、ARM7、ARM700、ARM710
v4	StrongARM、ARM8、ARM810
v4T	ARM7TDMI、ARM7TDMI-S、ARM720T、ARM740T、ARM7EJ、ARM9TDMI、ARM920T、ARM922T、ARM940T
v5TE	ARM9E-S、ARM10TDMI、ARM1020E
v6	ARM1136J(F)-S，ARM1176JZ(F)-S、ARM11、MPCore
v6T2	ARM1156T2(F)-S
v7	ARM Cortex-M、ARM Cortex-R、ARM Cortex-A

现在主流的构架包括 ARMv4、ARMv5、ARMv6 以及 ARMv7 等。基于这 4 种架构的 ARM 微处理器又可分为 ARM7(注：不包括 v3 架构部分)、ARM9、ARM9E、ARM10E、SecurCore、Xscale(Intel)、StrongARM(Intel 处理器技术和 ARM 构架的结合体)、ARM11 以及 Cortex 等几个主流系列，如表 1-2 所列。

表 1-2 ARM 主流内核架构分类说明

系 列	架 构	类型(核)	应用领域
ARM7	ARMv4T	ARM7TDMI、ARM7TDMI-S	对价格和功耗敏感的消费应用，如便携式手持设备、工业控制、网络设备、消费电子等
ARM9	ARMv4T	ARM9TDMI、ARM920T、ARM922T、ARM940T	无线设备、仪器仪表、安全系统、机顶盒、高端打印机、数字照相机、数字摄像机、音频、视频多媒体设备等
ARM9E	ARMv5TE	ARM926EJ-S、ARM946E-S、ARM966E	弥补了 ARM9 的不足，适用于需要高速数字信号处理、大量浮点运算、高密度运算的场合
ARM10E	RMv5TE	ARM1020E、ARM1022E、ARM1026EJ-S	工业控制、通信和信息系统等高端应用领域，包括高端的无线设备、基站设备、视频游戏机、高清成像系统
Xscale	ARMv5TE	PXA270、PXA271、PXA272	高端手持设备、多媒体设备、网络、存储、远程接入服务等领域
ARM11	ARMv6/ARMv6T2	ARM1136J、ARM1176、ARM11、MPCore、ARM1156T2	高端领域的应用：数字电视、机顶盒、游戏机、智能手机、音视频多媒体设备、无线通信基站设备、汽车电子产品
Cortex	ARMv7	Cortex-M0/M3、Cortex-M4、Cortex-R4、Cortex-A8、Cortex-A9/A15	M：当前 8/16 位 MCU 的换代产品；R：硬盘、打印机、汽车电子、网络和影像系统；A：成本敏感的手机、汽车控制系统、智能家电等

注：① T—Thumb 状态，支持 16 位压缩指令集；

② M—内嵌硬件乘法器，支持长乘法运算；

③ D—支持片上 Debug；

④ I—嵌入的 ICE(In Circuit Emulation)，支持片上断点和调试点；

⑤ E—支持 DSP 指令集；

⑥ J—支持 Java 指令集。

1.2 ARM Cortex 处理器

2005 年 3 月，ARM 公司公布了其最新的 ARMv7 架构的技术细节，定义了三大分工明确的系列："A"系列面向尖端的基于虚拟内存的操作系统和用户应用；"R"系列针对实时系统；"M"系列对微控制器和低成本应用提供优化。

ARMv7 架构采用了 Thumb-2 技术，是在 ARM 业界领先的 Thumb 代码压缩技

术的基础上发展起来，并且保持了对已存 ARM 解决方案的代码兼容性。Thumb-2 技术比纯 32 位代码少使用 31%的内存，降低了系统开销，同时却能够提供比已有的基于 Thumb 技术的解决方案高出 38%的性能表现。ARMv7 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 图形和游戏物理应用以及传统的嵌入式控制应用的需求。

ARM NEON 技术的通用 SIMD 引擎可有效处理当前和将来的多媒体格式。NEON 技术可加速多媒体和信号处理算法(如视频编码/解码、2D/3D 图形、游戏、音频和语音处理、图像处理技术、电话和声音合成)，其性能至少为 ARMv5 性能的 3 倍，为 ARMv6 SIMD 性能的 2 倍。

NEON 技术是 64/128 位单指令多数据流(SIMD)指令集，用于新一代媒体和信号处理应用加速。NEON 技术下执行 MP3 音频解码器，CPU 频率可低于 10 MHz；运行 GSM AMR 语音数字编解码器，CPU 频率仅为 13 MHz。新技术包含指令集、独立的寄存器及可独立的执行硬件。NEON 支持 8 位、16 位、32 位、64 位整数及单精度浮点 SIMD 操作，以进行音频/视频、图像和游戏处理。

ARM Cortex 处理器系列基于 ARMv7 架构，从尺寸和性能方面，既有少于 3.3 万个门电路的 ARM Cortex-M 系列，也有高性能的 ARM Cortex-A 系列。ARMv7 架构确保了与早期的 ARM 处理器之间良好的兼容性，既保护了客户在软件方面的投资，又为已存的系统设计提供了转换便捷。

(1) ARM Cortex-A 系列

ARM Cortex-A 针对日益增长的运行，包括 Linux、Windows CE 和 Symbian 在内的消费电子和无线产品。

(2) ARM Cortex-R 系列

ARM Cortex-R 针对需要运行实时操作系统来进行控制应用的系统，包括汽车电子、网络和影像系统。

(3) ARM Cortex-M 系列

ARM Cortex-M 为那些对开发费用非常敏感同时对性能要求不断增加的嵌入式应用而设计，如微控制器、汽车车身控制系统和各种大型家电。

1.3　Cirtex-M0 处理器

1.3.1　概述

Cortex-M0 处理器是市场上现有的核面积最小、能耗最低、最节能的 ARM 处理器。该处理能耗非常低、门数量少、代码占用空间小，使得 MCU 开发人员能够以 8 位处理器的价位，获得 32 位处理器的性能。超低门数还使其能够用于模拟信号设备和混合信号设备及 MCU 应用中，可明显节约系统成本。