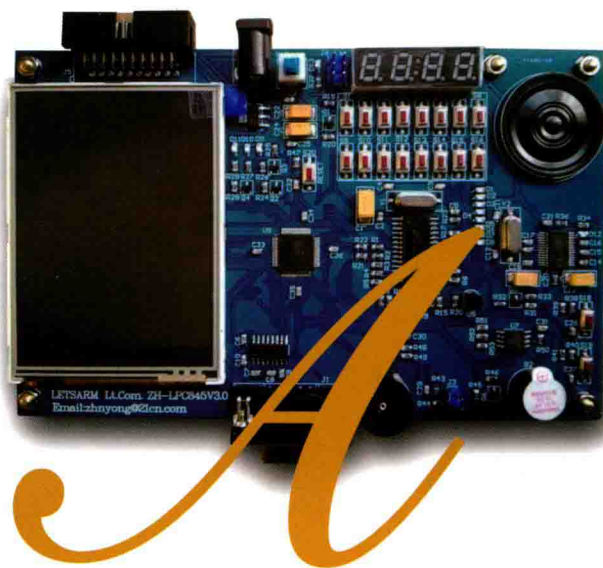


清華

开发者书库



Principles and Applications of ARM Embedded Microprocessor
Based on Cortex-M0 + Core LPC84X and μ C / OS-III Operating System

ARM嵌入式微控制器 原理与应用

基于Cortex-M0+内核LPC84X
与 μ C/OS-III操作系统

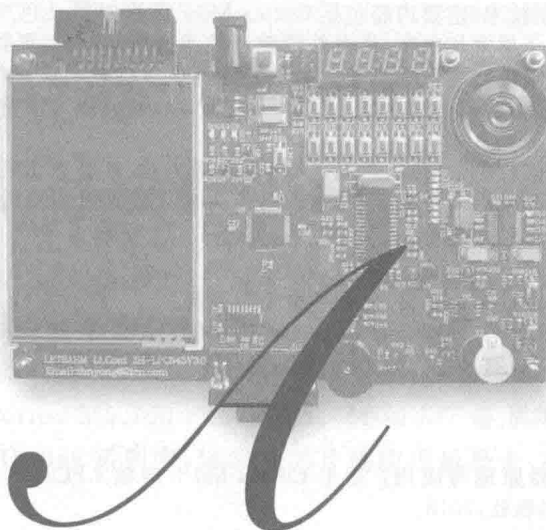
张 勇◎编著
ZhangYong

清华大学出版社



清华

开发者书库



Principles and Applications of ARM Embedded Microprocessor

Based on Cortex-M0 + CoreLPC84X and μ C / OS-III Operating System

ARM嵌入式微控制器 原理与应用

基于Cortex-M0+内核LPC84X
与 μ C/OS-III操作系统

张勇◎编著

ZhangYong

清华大学出版社

北京

内 容 简 介

ARM Cortex-M0+内核微控制器以其高性能、极低功耗和易用性等特点成为替代传统 8051 架构单片机的首选微控制器,其中以 NXP 公司 LPC84X 系列微控制器因其处理速度快、存储空间大和片内外设资源丰富而最有代表性。Micrium 公司 $\mu\text{C}/\text{OS-II}/\text{III}$ 系统软件是在全球范围内被广泛加载到微控制器上的嵌入式实时操作系统。本书结合微控制器 LPC84X 与嵌入式实时操作系统 $\mu\text{C}/\text{OS-II}/\text{III}$ 详细讲述了 ARM 微控制器原理与应用技术,主要内容包括 Cortex-M0+微控制器、LPC84X 硬件电路系统、Keil MDK 集成开发环境、Cortex-M0+异常与中断、片内外设驱动技术、 $\mu\text{C}/\text{OS-II}/\text{III}$ 移植、 $\mu\text{C}/\text{OS-II}/\text{III}$ 任务、信号量与互斥信号量以及消息邮箱与消息队列等。本书的特色在于理论与应用结合紧密且实例丰富,对学习基于 Cortex-M0+微控制器和实时操作系统 $\mu\text{C}/\text{OS-II}/\text{III}$ 等领域的嵌入式设计与应用开发技术,都具有很强的指导意义和参考价值。

本书可作为普通高等院校电子信息、通信工程、物联网工程、计算机工程、软件工程、自动控制和智能仪器等相关专业的高年级本科生或研究生教材,也可作为嵌入式系统设计爱好者和工程开发人员的参考用书。

本书封面贴有清华大学出版社防伪标签,无标签者不得销售。

版权所有,侵权必究。侵权举报电话:010-62782989 13701121933

图书在版编目(CIP)数据

ARM 嵌入式微控制器原理与应用:基于 Cortex-M0+内核 LPC84X 与 $\mu\text{C}/\text{OS-III}$ 操作系统/张勇编著. —北京:清华大学出版社,2018

(清华开发者书库)

ISBN 978-7-302-49961-9

I. ①A… II. ①张… III. ①微处理器—系统设计 IV. ①TP332

中国版本图书馆 CIP 数据核字(2018)第 060325 号

责任编辑:王一玲 赵 凯

封面设计:李召霞

责任校对:梁 毅

责任印制:宋 林

出版发行:清华大学出版社

网 址: <http://www.tup.com.cn>, <http://www.wqbook.com>

地 址:北京清华大学学研大厦 A 座

邮 编:100084

社总机:010-62770175

邮 购:010-62786544

投稿与读者服务:010-62776969, c-service@tup.tsinghua.edu.cn

质量反馈:010-62772015, zhiliang@tup.tsinghua.edu.cn

课件下载: <http://www.tup.com.cn>, 010-62795954

印装者:三河市金元印装有限公司

经 销:全国新华书店

开 本:186mm×240mm

印 张:28.75

字 数:683 千字

版 次:2018 年 6 月第 1 版

印 次:2018 年 6 月第 1 次印刷

印 数:1~1500

定 价:79.00 元

产品编号:079204-01

前言

PREFACE

当前,ARM 微控制器正在逐步替代传统 8051 架构单片机而成为嵌入式系统的核心控制器。2010 年以后,ARM 公司主推 Cortex 系列内核,Cortex 系列分为 M 系、R 系和 A 系,其中,A 系是高性能内核,用于基于 Android 操作系统的智能手机和平板电脑,支持 ARM、Thumb 和 Thumb-2 指令集;R 系为微处理器内核,支持 ARM、Thumb 和 Thumb-2 指令集;M 系为低功耗微控制器内核,仅支持 Thumb-2 指令集,诞生于 2004 年,最早推出的内核为 Cortex-M3,目前有 Cortex-M0、M0+、M1、M3、M4 和 M7 等,用于支持快速中断的嵌入式实时应用系统中。在 Cortex 系列中,M 系列芯片的应用量最大,每年的应用量为几十亿块。

在 Cortex-M 系列中,M0 和 M0+内核都是极低功耗内核,M0+内核的功耗比 M0 内核更低(ARM 公司公布的功耗数据为 $11.2\mu\text{W}/\text{MHz}$),被誉为全球功耗最低的微控制器内核,主要应用在控制和检测领域,涵盖了传统 8051 单片机的应用领域,比传统 8051 单片机在处理速度、功耗、片上外设灵活多样性、中断数量与中断反应能力、编程与调试等诸多方面都有更大优势,M0+内核的代表芯片如 NXP 公司的 LPC845 微控制器。

基于 ARM Cortex-M0+微控制器的软件开发有两种方式,即传统的芯片级别的应用软件开发和加载嵌入式实时操作系统的应用软件开发。芯片级软件开发方式使用 C 语言函数管理硬件外设驱动和实现用户功能,称之为面向函数的程序设计方式;加载嵌入式操作系统的应用软件开发使用嵌入式操作系统管理硬件外设和存储资源,借助于用户任务实现用户功能,称之为面向任务的程序设计方式。由于 Cortex-M0+微控制器片内 RAM 空间丰富,一般在 8KB 以上,适宜加载嵌入式实时操作系统(RTOS) $\mu\text{C}/\text{OS-II}$ 或 $\mu\text{C}/\text{OS-III}$ 。在 Cortex-M0+微控制器上加载了 RTOS 后,将显著加速项目的开发进度。

本书主要以 Cortex-M0+内核 LPC845 微控制器为例,在介绍了 Cortex-M0+内核组成原理和 LPC84X 微控制器芯片结构后,详细介绍了 LPC845 典型硬件系统及其片上外设的驱动方法,基于面向函数的程序设计方法介绍了 LED 灯、蜂鸣器、按键、数码管、温度显示(DS18B20)、串口通信、模数转换器(ADC)、存储器访问、LCD 屏显示和触摸屏输入等外设驱动程序设计技术;然后,详细介绍了嵌入式实时操作系统 $\mu\text{C}/\text{OS-II}$ 和 $\mu\text{C}/\text{OS-III}$ 在 LPC845 微控制器上的移植与应用技术,包括用户任务、信号量与互斥信号量、消息邮箱与消息队列等组件应用程序设计方法,重点在于阐述面向任务的程序设计方法及其优越性。

本书讲义经过多名教师应用,理论学时宜为 32 学时,实验学时为 32 学时。建议讲述内

容为第1~12章(第一篇与第二篇的全部章节),选学内容为第13~16章(第三篇内容),按书中章节顺序讲述。作者巧妙地组织了书中的全部实例,使得全部实例代码均是完整的。因此,要求读者必须在掌握了前面章节实例的基础上,才能学习后面章节的实例。对于自学本书的嵌入式爱好者而言,要求至少具有数字电路、模拟电路、C语言程序设计等课程的基础知识,并建议使用LPC845学习板辅助学习,以增加学习乐趣。本书的全部源代码可从百度网盘上下载(链接: <https://pan.baidu.com/s/1bRB6XK>, 密码: 4kou), 请购买本书同步阅读。

本书具有以下三个方面的特色:

其一,公布了基于LPC845微控制器为核心的开源硬件平台,对嵌入式硬件开发具有很强的指导作用。

其二,全书工程实例丰富,通过完整的工程实例详细讲述了函数级别与任务级别的程序设计方法,对于嵌入式系统应用软件开发具有颇强的指导意义。

其三,结合LPC845硬件平台,详细讲述了嵌入式实时操作系统 $\mu\text{C}/\text{OS}-\text{II}/\text{III}$ 的任务管理和系统组件应用方法,对学习和应用 $\mu\text{C}/\text{OS}-\text{II}/\text{III}$ 具有良好的可借鉴性。

感谢恩智浦(NXP)中国公司辛华峰经理对本书编写的关心与支持;感谢北京博创智联科技有限公司陆海军总经理对本书编写的关心与支持;感谢广州天嵌计算机科技有限公司梁传智总经理对本书编写的关心与支持;感谢清华大学出版社的辛勤工作;感谢我的爱人贾晓天在资料检索和LPC845学习板焊装调试方面所做的大量工作;感谢阅读了作者已出版的教材并反馈了宝贵意见的读者们。本书的编写通俗易懂,其自学门槛较以往的教材大大降低。

由于作者水平有限,书中难免会有纰漏之处,敬请同行专家和读者朋友批评指正(联系方式: 15270015009@qq.com)。

免责声明: 知识的发展和科技的进步是多元的。本书内容上广泛引用的知识点均罗列于参考文献中,主要为LPC845用户手册、LPC845芯片手册、Cortex-M0+技术手册、嵌入式实时操作系统 $\mu\text{C}/\text{OS}-\text{II}/\text{III}$ 、Keil MDK集成开发环境、ULINK2仿真资料和Altium Designer软件等内容,所有这些引用内容的知识产权归相关公司所有。本书内容仅用于教学目的,旨在推广ARM Cortex-M0+内核LPC845微控制器、嵌入式实时操作系统 $\mu\text{C}/\text{OS}-\text{II}/\text{III}$ 和Keil MDK集成开发环境等,禁止任何单位和个人摘抄或扩充本书内容用于出版发行,严禁将本书内容用于商业场合。

张 勇

2018年4月于

江西财经大学枫林园

目录

CONTENTS

第一篇 LPC84X 典型硬件系统与芯片级软件设计

第 1 章 ARM Cortex-M0+内核	3
1.1 ARM Cortex-M0+内核特点	3
1.2 ARM Cortex-M0+内核架构	4
1.3 ARM Cortex-M0+存储器配置	5
1.4 ARM Cortex-M0+内核寄存器	7
1.4.1 内核寄存器	7
1.4.2 系统控制寄存器	8
1.5 SysTick 定时器	13
1.6 Cortex-M0+异常	15
1.7 嵌套向量中断控制器	16
1.8 本章小结	18
第 2 章 LPC84X 微控制器	19
2.1 LPC845 微控制器特点与引脚配置	19
2.2 LPC845 微控制器内部结构	30
2.3 LPC845 存储器配置	32
2.4 LPC845 NVIC 中断	33
2.5 I/O 口配置 IOCON	36
2.6 通用目的输入输出 GPIO	39
2.7 系统配置模块 SYSCON	42
2.8 本章小结	55
第 3 章 LPC845 典型硬件平台	56
3.1 LPC845 核心电路	57
3.2 电源电路	58

3.3	LED 驱动电路与蜂鸣器驱动电路	59
3.4	串口通信电路	59
3.5	用户按键与用户接口和 ADC 电路	60
3.6	DS18B20 电路	61
3.7	ZLG7289B 电路	61
3.8	SWD、ISP 和复位电路	64
3.9	LCD 屏与触摸屏接口电路	65
3.10	存储器电路	66
3.11	声码器电路	67
3.12	本章小结	67
第 4 章 LED 灯与蜂鸣器控制		68
4.1	LED 灯控制	68
4.1.1	LPC845 GPIO 口读写访问	69
4.1.2	Keil MDK 工程框架	71
4.2	LPC845 异常管理	87
4.2.1	LPC845 异常	87
4.2.2	LED 灯闪烁工程	89
4.3	NVIC 中断管理	93
4.3.1	多速率定时器 MRT	93
4.3.2	MRT 定时器中断实例	97
4.4	蜂鸣器工作原理	100
4.5	LPC845 外部中断	102
4.5.1	外部中断与模式匹配工作原理	102
4.5.2	LPC845 外部中断实例	112
4.5.3	LPC845 模式匹配实例	117
4.6	本章小结	119
第 5 章 按键与数码管显示		120
5.1	ZLG7289B 工作原理	120
5.2	DS18B20 工作原理	124
5.3	按键与数码管实例	132
5.4	本章小结	142
第 6 章 串口通信与声码器		143
6.1	串口通信	143

6.1.1	LPC845 串口工作原理	143
6.1.2	串口通信实例	152
6.2	声码器	157
6.2.1	声码器工作原理	157
6.2.2	声码器实例	160
6.3	本章小结	168
第 7 章	ADC 与存储器访问	169
7.1	LPC845 微控制器 ADC	169
7.1.1	ADC 工作原理	169
7.1.2	ADC 工程实例	174
7.2	AT24C128 存储器	179
7.2.1	AT24C128 访问方法	179
7.2.2	AT24C128 访问实例	182
7.3	W25Q64 存储器	190
7.3.1	W25Q64 存储器访问方法	190
7.3.2	LPC845 微控制器 SPI 模块	192
7.3.3	W25Q64 访问实例	196
7.4	本章小结	208
第 8 章	触摸屏与 LCD 屏	209
8.1	电阻式触摸屏驱动原理	209
8.2	电阻式触摸屏实例	215
8.3	LCD 屏驱动原理	217
8.4	LCD 屏实例	239
8.5	本章小结	244

第二篇 嵌入式实时操作系统 $\mu\text{C}/\text{OS-II}$

第 9 章	$\mu\text{C}/\text{OS-II}$ 系统与移植	247
9.1	$\mu\text{C}/\text{OS-II}$ 系统移植	247
9.2	$\mu\text{C}/\text{OS-II}$ 系统结构与配置	260
9.3	$\mu\text{C}/\text{OS-II}$ 系统任务	266
9.3.1	空闲任务	267
9.3.2	统计任务	267
9.3.3	定时器任务	268

9.4 本章小结	268
第 10 章 $\mu\text{C}/\text{OS-II}$ 任务管理	269
10.1 $\mu\text{C}/\text{OS-II}$ 用户任务	269
10.2 $\mu\text{C}/\text{OS-II}$ 多任务工程实例	274
10.3 统计任务实例	286
10.4 系统定时器	290
10.5 本章小结	293
第 11 章 信号量与互斥信号量	294
11.1 $\mu\text{C}/\text{OS-II}$ 信号量	294
11.2 $\mu\text{C}/\text{OS-II}$ 互斥信号量	296
11.3 信号量与互斥信号量实例	298
11.4 本章小结	310
第 12 章 消息邮箱与消息队列	311
12.1 $\mu\text{C}/\text{OS-II}$ 消息邮箱	311
12.2 $\mu\text{C}/\text{OS-II}$ 消息队列	313
12.3 消息邮箱与消息队列实例	315
12.4 本章小结	324

第三篇 嵌入式实时操作系统 $\mu\text{C}/\text{OS-III}$

第 13 章 $\mu\text{C}/\text{OS-III}$ 系统与移植	327
13.1 $\mu\text{C}/\text{OS-III}$ 发展历程	327
13.2 $\mu\text{C}/\text{OS-III}$ 特点	329
13.3 $\mu\text{C}/\text{OS-III}$ 应用领域	334
13.4 $\mu\text{C}/\text{OS-III}$ 系统组成	335
13.4.1 $\mu\text{C}/\text{OS-III}$ 配置文件	338
13.4.2 $\mu\text{C}/\text{OS-III}$ 内核文件	343
13.5 $\mu\text{C}/\text{OS-III}$ 自定义数据类型	352
13.6 $\mu\text{C}/\text{OS-III}$ 移植	354
13.7 本章小结	362
第 14 章 $\mu\text{C}/\text{OS-III}$ 任务管理	363
14.1 用户任务	363

14.1.1	任务堆栈与优先级	365
14.1.2	任务控制块	365
14.1.3	任务工作状态	370
14.1.4	用户任务创建过程	372
14.2	多任务工程实例	374
14.3	统计任务	390
14.4	定时器任务	391
14.5	本章小结	395
第 15 章	信号量、任务信号量和互斥信号量	396
15.1	信号量	396
15.1.1	信号量工作方式	396
15.1.2	信号量实例	397
15.2	任务信号量	409
15.2.1	任务信号量工作方式	409
15.2.2	任务信号量实例	409
15.3	互斥信号量	413
15.3.1	互斥信号量工作方式	413
15.3.2	互斥信号量实例	414
15.4	本章小结	418
第 16 章	消息队列与任务消息队列	420
16.1	消息队列	420
16.1.1	消息队列工作方式	422
16.1.2	消息队列实例	424
16.2	任务消息队列	432
16.2.1	任务消息队列工作方式	432
16.2.2	任务消息队列实例	433
16.3	本章小结	440
附录 A	文件 my25q64.c	442
附录 B	工程项目索引	448
参考文献		450

第一篇 LPC84X典型硬件系统与芯片级软件设计



本篇包括第1~8章,为全书的硬件系统与芯片级软件设计部分,将依次介绍 ARM Cortex-M0+内核、LPC84X 微控制器、LPC845 典型硬件系统电路和面向函数的程序设计方法。在本篇中,借助于实例和 Keil MDK 集成开发环境,先后详细阐述以下板级外设的芯片级工程程序设计方法:

- (1) LED 灯与蜂鸣器控制;
- (2) 按键与数码管显示(包含 DS18B20 温度传感器访问技术);
- (3) 串口通信(包含 SYN6288 声码器访问技术);
- (4) 模数转换器(ADC)和存储器访问技术;
- (5) 240×320 像素分辨率彩色 LCD 屏与触摸屏控制。

需要说明的是,全书中使用了 Keil MDK 5.24a 版本,该版本为 2017 年 7 月发布的最新版本。建议读者使用最新的 Keil MDK 版本(由于版本升级而导致的工程兼容性问题,请与作者联系)。

第 1 章



ARM Cortex-M0+ 内核

ARM 是 Advanced RISC Machine(高级精简指令集机器)的缩写,现为 ARM 公司的注册商标。ARM Cortex-M0+内核属于 ARM 公司推出的 Cortex-M 系列内核之一,相对于高性能的 Cortex-M3 内核而言,它具有体积小、功耗低和控制灵活等特点,主要针对传统单片机的控制与显示等嵌入式系统应用。本章将介绍 Cortex-M0+内核的特点、架构、存储器配置和内核寄存器等内容。

1.1 ARM Cortex-M0+ 内核特点

Cortex-M0+内核基于 ARMv6-M 体系结构,使用 ARMv6-M 汇编语言指令集,它具有以下特点:

(1) Cortex-M0+内核包含极低数量的门电路,是目前全球功耗最低的内核,特别适用于对功耗要求苛刻的嵌入式系统应用场合。

(2) 支持 32 位长的 Thumb-2 扩展指令集和 16 位长的 Thumb 指令集,代码的执行效率远远高于 8 位长的单片机汇编指令。

(3) 支持单周期的 I/O(输入/输出)口访问,对外设的控制速度快。

(4) 具有低功耗工作模式,在内核空闲时可以使其进入低功耗模式,从而极大地节约电能;当需要内核工作时,通过与内核紧耦合的快速中断唤醒单元使其进入正常工作模式。

(5) 内核中的各个组件采用模块化结构,通过精简的高性能总线(AHB-Lite)连接在一起,内核中的功耗管理单元可以动态配置各个组件的工作状态,可根据需要使某些空闲的组件处于掉电模式,以尽可能地减少功耗。

(6) 可执行代码保存在 Flash 存储区中,Cortex-M0+内核支持从 Flash 中以极低的功耗快速读取指令,并以极低的功耗(工作在一个相对较低的 CPU 时钟下)在内核中高速执行代码。

(7) Cortex-M0+内核支持硬件乘法器,硬件乘法器最早出现在 DSP(数字信号处理器)芯片中,与加法器协同工作,并称为乘加器,是指在一个 CPU 时钟周期内,用硬件电路直接实现 $A \times B + C$ 的三操作数运算。这里 Cortex-M0+支持的硬件乘法器可以在一个 CPU 时

钟周期内实现 $A \times B$ 的二操作数运算。

(8) Cortex-M0+内核的每条汇编指令的执行周期是确定的；中断处理的时间是确定的、高效的，且具有快速中断处理能力，特别适用于对实时性要求苛刻的智能控制场合。

(9) 支持二线的串行调试接口(SWD)，只需要使用芯片的二根引脚就可以实现对Cortex-M0+内核芯片的在线仿真与调试，通过SWD可以向芯片的Flash存储器固化程序代码，且具有指令跟踪执行功能。而绝大多数的传统单片机是不能在线仿真调试的，因此基于单片机的工程测试复杂且周期漫长。

(10) Cortex-M0+内核不是物理形态的微控制器芯片，而是属于知识产权(IP)，所以常被称为IP软核。目前全球至少有200多家大型半导体公司购买了ARM公司的IP核，生产集成了IP核的微控制器芯片(称为流片，流片测试成功后进入芯片量产阶段)。所有集成了Cortex-M0+内核的微控制器芯片，均可使用相同的集成开发环境(如Keil公司的MDK和IAR公司的EWARM等)和相同的仿真器(如ULink2和JLink V8等)进行软件开发。事实上，几乎全部的ARM芯片都使用相同的开发环境和仿真器，但是对于传统的单片机而言，不同半导体厂商生产的单片机所用的开发环境和编程下载器往往不相同。

1.2 ARM Cortex-M0+ 内核架构

相对于8位字长的传统8051单片机而言，Cortex-M0+内核是32位字长的微控制器内核，其内部总线宽度为32位，指令和数据传输能力大大提升。Cortex-M0+内核架构如图1-1所示。

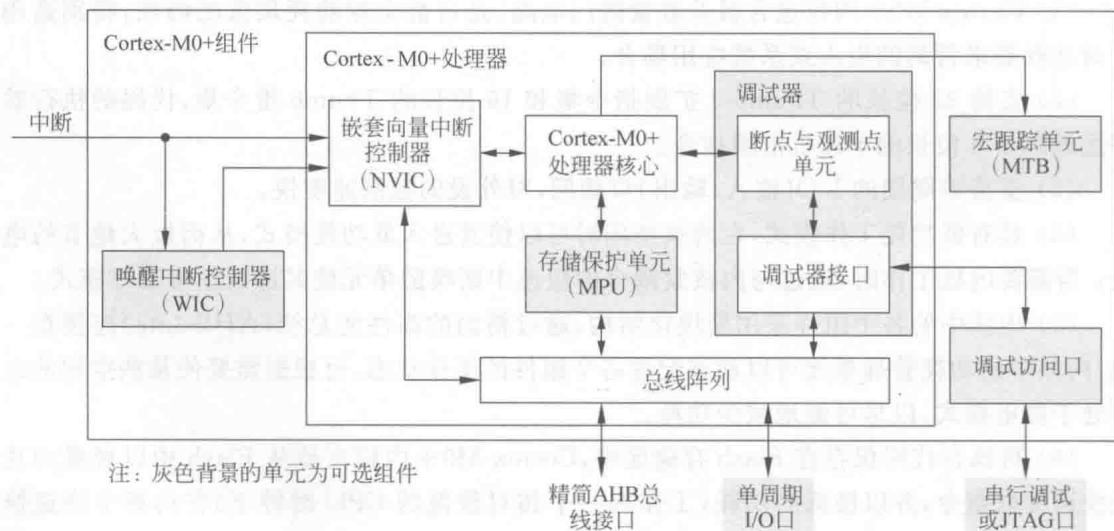


图 1-1 Cortex-M0+ 内核架构

由图1-1可知，Cortex-M0+内核由Cortex-M0+处理器和三个可选的组件，即唤醒中断控制器(WIC)、宏跟踪单元(MTB)和调试访问口组成，Cortex-M0+处理器包括Cortex-

M0+处理器核心、嵌套向量中断控制器(NVIC)和两个可选的组件,即存储保护单元(MPU)和调试器组成,其中调试器又包括断点与观测点单元和调试器接口。Cortex-M0+内核与外部通过总线阵列和中断进行通信,其中中断为单向输入口,与总线阵列相连接的精简高性能总线(AHB)接口以及可选的单周期 I/O(输入/输出)口和串行调试或 JTAG 口均为双向口。

32 位的 Cortex-M0+处理器核心是计算和控制中心,采用了两级流水的冯·诺依曼结构,执行 ARMv6-M 指令集(即 16 位长的 Thumb-2 指令集,含 32 位长的扩展指令),集成了一个单周期的乘法器(用于高性能芯片中)或一个 32 周期的乘法器(用于低功耗芯片中)。

Cortex-M0+内核具有很强的中断处理能力,一个优先级可配置的嵌套向量中断控制器(NVIC)直接与 Cortex-M0+处理器核心相连接,通过这种紧耦合的嵌套向量中断控制器,可以实现不可屏蔽中断、中断尾链、快速中断响应、睡眠态唤醒中断和四级中断优先级。这里的“中断尾链”是指当有多个中断被同时触发时,优先级高的中断响应完成后,不需要进行运行环境的恢复,而是直接运行优先级次高的中断,全部中断响应完后,再恢复运行环境。如果没有中断尾链功能,则处理器在响应一个中断前,先进行入栈操作,保存当前中断触发时的运行环境,然后处理器暂停当前程序的执行去响应中断,响应完中断后,进行出栈操作,恢复响应中断前的环境(即使程序计数器指针 PC 指向被中断的程序位置处)继续执行原来的程序,接着重复这些操作响应下一个中断。因此,没有中断尾链功能时,两个连续响应的中断间需要插入一次出栈和一次入栈操作(即恢复前一个运行环境和保存后一个运行环境),而具有中断尾链功能时,这两次堆栈操作均被省略掉了。

Cortex-M0+内核具有很强的调试能力,通过调试访问口,外部的串行调试或 JTAG 调试口与 Cortex-M0+处理器的调试器相连接,调试器直接与 Cortex-M0+处理器核心连接,还通过它与宏跟踪单元相连接。因此,通过串行调试或 JTAG 调试口可以访问 Cortex-M0+内核的全部资源,包括调试或跟踪程序代码的执行,还可以检查代码的执行结果。

存储保护单元(MPU)可以对存储器的某些空间设定访问权限,使用特定的程序代码才能访问这些空间,普通的程序代码则无权访问,这样可以有效地保护关键的程序代码存储区或数据区不受意外访问(例如病毒)的侵害。

图 1-1 中,相对于 Cortex-M0+处理器核心而言,其余的组件均称为 Cortex-M0+内核的外设,所有这些组件均为知识产权(IP)核。半导体厂商在这个 IP 核的基础上添加中断发生器、存储器、与精简 AHB 总线(通过高级外设总线 APB)相连接的多功能芯片外设和 I/O(输入/输出)口等,即可以得到特定功能的微控制器芯片。

1.3 ARM Cortex-M0+ 存储器配置

Cortex-M0+存储空间的最大访问能力为 2^{32} 字节,即 4GB。对于 Cortex-M0+而言,8 位(8 bit)为一个字节,16 位称为半字,32 位称为字。以字为单位,Cortex-M0+的存储空间的最大访问能力为 2^{30} 字,即从 $0 \sim 2^{30} - 1$ 字。一般地,访问地址习惯采用字节地址,此时

Cortex-M0+的存储空间配置如图 1-2 所示。

0xFFFF FFFF	系统专用设备	系统专用设备区域为芯片生产者专用的外设映射空间
0xE000 0000 0xDFFF FFFF		
0xC000 0000 0xBFFF FFFF	私有设备	私有设备区域是针对特定芯片的外设映射空间
0xA000 0000 0x9FFF FFFF		
0x8000 0000 0x7FFF FFFF	共享设备	共享设备区域是公用的外设映射空间
0x6000 0000 0x5FFF FFFF		
0x4000 0000 0x3FFF FFFF	RAM	该RAM区域为慢速RAM区, 具有“写通”(Write Through)特性
0x2000 0000 0x1FFF FFFF		
0x0000 0000	Code	该RAM区域为快速RAM区, 具有“写回”(Write Back)特性
	片上外设	片上外设区域为片上外设的寄存器区域, 通过访问该区域内的寄存器管理片上外设资源
	SRAM	SRAM区域为片上RAM区, 主要用于存放数据
		Code区域为片上ROM或Flash区, 主要用于存放程序代码。中断向量表位于0x0起始地址处

图 1-2 Cortex-M0+存储空间配置

由图 1-2 可知, 4GB 的 Cortex-M0+ 映射存储空间被分成 8 个相同大小的空间, 每个空间为 0.5GB。这 8 个空间中, 位于地址范围 0x0000 0000~0x1FFF FFFF 的 Code 空间对应着集成在芯片上的 ROM 或 Flash 存储器, 主要用于保存可执行的程序代码, 也可用于保存数据, 其中中断向量表位于地址 0x0 起始的地址空间中, 一般占有几十至几百个字节。位于地址范围 0x2000 0000~0x3FFF FFFF 的 SRAM 区域, 对应着集成在芯片中的 RAM 存储器, 主要用于保存数据, 保存的数据在芯片掉电后丢失。位于地址范围 0x4000 0000~0x5FFF FFFF 的片上外设区域, 存储着片上外设的访问寄存器, 通过读或写这些寄存器, 可实现对片上外设的访问和控制。

位于地址范围 0x6000 0000~0x7FFF FFFF 的 RAM 区域属于快速 RAM 区, 具有“写回”特性的缓存区, 而位于地址范围 0x8000 0000~0x9FFF FFFF 的 RAM 区域属于慢速 RAM 区, 具有“写通”特性的缓存区。这两个 RAM 区都有对应的缓存(Cache), 所谓的“写回”是指当 Cortex-M0+ 内核向 RAM 区写入数据时, 不是直接将数据写入 RAM, 而是写入

更快速的缓存中,当缓存满了或者总线空闲时,缓存自动将数据写入 RAM 区中。所谓的“写通”是指当 Cortex-M0+内核向 RAM 区写入数据时,通过缓存直接将数据写入 RAM 区中。因此,“写回”特性的 RAM 中的数据有可能与其缓存中的数据不同,当数据较少时,数据将保存在缓存中,而不用写入 RAM 中;而“写通”特性的 RAM 中的数据与缓存中相应的数据是相同的。其实,Cortex-M0+的 Code 区和 SRAM 区也有缓存,前者是“写通”特性的缓存,后者是“写回”特性的缓存机制。

Cortex-M0+支持两种存储模式,即小端模式和字节保序的大端模式。不妨设 Addr 为一个字地址所谓的字地址是指地址的最低两位为 0,即地址能被 4 整除,如 0x0000 0000、0x0000 0004、0x0000 0008 等都是字地址。同样,把地址的最低一位为 0 的地址称为半字地址,如 0x0000 0002、0x0000 0004 等。显然,字地址都是半字地址。设 DataA 为一个字,包括 4 字节,从高字节到低字节依次记为 DataA[31:24]、DataA[23:16]、DataA[15:8]和 DataA[7:0],设 DataB 为一个半字,包括 2 字节,依次记为 DataB[15:8]和 DataB[7:0],则在两种存储模式下,将 DataA 或 DataB 保存在 Addr 地址的情况如图 1-3 所示。

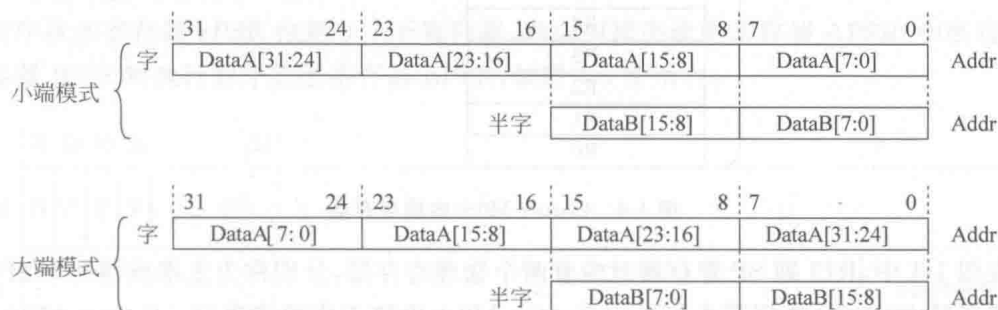


图 1-3 小端与字节保序的大端存储模式示例

由图 1-3 可知,小端模式下,数据字的高字节存储在字地址的高端,数据字的低字节存储在字地址的低端;而大端模式下刚好相反,数据字的高字节存储在字地址的低端,数据字的低字节存储在字地址的高端。所谓的字节保序是指在大端模式下,每个字节内的位的顺序保持从高到低不变。

1.4 ARM Cortex-M0+ 内核寄存器

在 Cortex-M0+微控制器中,全部的寄存器都是 32 位的,本节将重点介绍 Cortex-M0+的内核寄存器,这些寄存器直接服务于处理器内核,用于保存计算和控制的运算数和操作码。此外,本节还将介绍 Cortex-M0+内核的系统控制寄存器。需要指出的是,内核寄存器只能用汇编语言访问,C 语言(即使其指针功能)无法访问。

1.4.1 内核寄存器

Cortex-M0+微控制器共有 16 个内核寄存器,分别记为 R0~R15,其中 R0~R12 为 32