

本书提供视频教程和源代码

嵌入式技术与应用 丛书

跟工程师学嵌入式开发

基于STM32和 μ C/OS-III

谭贵 易确 熊立宇 编著



中国工信出版集团



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

嵌入式技术与应用 丛书

跟工程师学嵌入式开发

基于STM32和 μ C/OS-III

谭贵 易确 熊立宇 编著

电子工业出版社

Publishing House of Electronics Industry

北京•BEIJING

内 容 简 介

本书选用的 STM32 芯片基于 ARM Cortex-M3 体系结构,根据基于 MCU 的嵌入式技术实际应用需求,合理地选择了多种常用的重要外设接口,如 USART、SPI、I2C、FSCM、SDIO 总线、以太网等,结合丰富的实例及工程源代码,由浅入深、系统全面地介绍嵌入式系统的底层工作原理。在此过程中,通过穿插多个综合示例的讲解,如命令行外壳程序 Shell、eFat 文件系统、Telnet 远程控制、 μ C/OS-III 实时操作系统的移植过程,无论是嵌入式的初学者,还是有一定开发经验的工程师都能从中获益,使读者既能系统全面地掌握嵌入式开发所需的软硬件知识,又能锻炼他们的综合开发能力,为将来从事嵌入式开发方面的工作奠定坚实的基础。

本书可作为高等学校电子、计算机、自动化控制类等相关专业的教材,也可供工程师、嵌入式爱好者及自学人员阅读。

本书提供视频教程和工程源代码,读者可登录华信教育资源网(www.hxedu.com.cn)免费注册后下载。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

跟工程师学嵌入式开发:基于 STM32 和 μ C/OS-III/谭贵,易确,熊立宇编著. —北京:电子工业出版社, 2017.10

(嵌入式技术与应用丛书)

ISBN 978-7-121-32725-4

I. ①跟… II. ①谭… ②易… ③熊… III. ①微处理器—系统设计 IV. ①TP332

中国版本图书馆 CIP 数据核字(2017)第 228902 号

责任编辑:田宏峰

印 刷:三河市君旺印务有限公司

装 订:三河市君旺印务有限公司

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本:787×1092 1/16 印张:28 字数:716 千字

版 次:2017 年 10 月第 1 版

印 次:2017 年 10 月第 1 次印刷

定 价:88.00 元

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888, 88258888。

质量投诉请发邮件至 zltts@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

本书咨询联系方式: tianhf@phei.com.cn。

FOREWORD

前言

进入 21 世纪以来,随着微电子技术、计算机技术及网络通信等技术的深入发展,整个社会信息化的程度越来越高,不用说智能手机,各种信息设备,如应用于医疗健康领域的智能心电仪、智能血糖仪,工业生产领域的自动测试装置、机器人手臂,安防领域的指纹识别、人脸识别技术,智能家居里的智能空调、冰箱、电表,甚至军事领域中的精确制导武器、红外热成像眼镜、自动跟踪……都无一例外地具有“智慧”的大脑,它们的应用已深入我们生活的每个角落,改变着我们的生活方式。

在上面所提及的应用中,其“智慧”的大脑,实质就是一套套嵌入式系统,它们由不同的硬件和软件组成。这里所说的“不同”,一是指构成嵌入式系统的硬件核心,可能是由基于不同厂商的 SoC 芯片所拓展设计的实用电路,如 ST 公司的 STM8、STM32 系列 SoC、NXP 公司的 LPC 系列 SoC;二是指嵌入式系统的软件构成,除去 SoC 片上外设的必要驱动外,还有管理这些驱动和应用的操作系统,如 Embedded Linux、 μ C/OS-III 和文件管理系统(如 FatFs)等。

目前,嵌入式应用处理器多采用 ARM 体系结构。ARM 公司为了细分市场,将其芯片按应用领域分为 Cortex-A、Cortex-R、Cortex-M 三个系列。Cortex-A 系列芯片带有 MMU(内存管理单元)、MPU(内存保护单元)部件,主要针对复杂的嵌入式应用,如智能手机、平板电脑及高档成像设备等;Cortex-R 系列面向实时应用领域,如精密机器控制、炉温监控;Cortex-M 系列则面向传统的单片机市场,其子系列 M0~M4 涵盖了从 8 位到 32 位单片机的所有应用,与传统的 C51 系列单片机相比,功能更强大。因此,学习嵌入式开发技术,应本着“循序渐进,由简单到复杂”的原则,首先学习基于 Cortex-M 系列的单片机是最好的入门选择,在此基础上,进阶学习 Cortex-A 系列就显得“自然而游刃有余”。

本书就以基于 CM3 内核架构(Cortex-M3)的芯片 STM32F103ZET6(意法半导体公司 ST,基于 MCU 应用的 32 位芯片系列,简称 STM32)为讲解线索,合理选择实际应用中广泛使用的 USART、I2C、SPI、SDIO、以太网等接口,结合 ST 公司提供的库函数,通过一个个具有实际使用价值的案例代码,详细介绍每种接口的工作原理、驱动配置和综合应用。在学习本书之前,如果读者对 CM3 体系结构有一定的了解,当然最好;如果没有此类的背景知识,也不用担心会影响对本书的学习。笔者在讲解过程中,会在涉及需要 CM3 体系结构知识的地方,自然而然地引入相应知识点的介绍。

本书对章节、知识点的安排有以下三个显著特点,以便读者快速、高效地掌握基于“STM32+ μ C/OS-III 结构”的嵌入式开发。

在章节的安排上,遵循“先总体,后细节”的原则。第 1 章以一个跑马灯实验作为引

子，为读者介绍了嵌入式开发所涉及的基本概念和流程，如 GPIO 配置、事件的轮询和中断处理机制等，以及基于 STM32 库开发所涉及的库文件组织和 CMSIS 标准、开发工具。建立了嵌入式开发的基本过程等轮廓性认识之后，第 2 章自然地切入 STM32 系列芯片的框架结构，包括总线、外设地址空间映射和时钟树。只有清楚了这三者之间的关系，才有可能对后面章节所讲解的外设工作过程和相关操作有深刻的理解。在此基础上，第 3 章开启了嵌入式系统的启动之旅，透彻地分析了基于 CM3 核的芯片系统之启动过程。有了前面三章的基础，随后的章节则以“先简单，后复杂”的原则逐一介绍 GPIO、外部中断线、USART、DMA、I2C、SDIO 等外设的结构原理及相应的驱动代码。

其次，对每一种外设的讲解，除了遵循“先总体，后细节”的原则之外，采用“以实验现象为驱动（每章的第一节首先呈现给读者一个最终的实验结果画面）”的思路，一步步进入外设的内部世界。这样的安排有利于激起读者对未知世界的强烈兴趣，随着对外设“先总后细”的逐层深入，最终使读者彻底理解并掌握每种外设“实验现象”背后的逻辑。

最后，本书除了讲解每种外设的工作原理和驱动代码之外，还穿插了三个有实用价值的综合案例，以拓展读者对外设应用的认识和理解，以及必要的知识面。第一个综合实例是基于 USART 接口的 Shell（俗称“外壳”，类似于 Linux 的 bash）程序，通过它可以“对外设的操作”封装为一个个 Shell 命令，以随时执行。因此，Shell 程序贯穿本书的始末。第二个综合实例是 Telnet 远程登录服务程序，该程序底层硬件是基于 SPI 总线的以太网芯片 ENC2860，上层使用 uIP 协议栈来完成常用的 TCP/IP 功能，如 ping、ICMP、IP、TCP 等。“麻雀虽小，五脏俱全”，通过实现这样的服务程序，不但可以使读者理解和掌握 Telnet 协议的工作原理及过程，而且有利于进一步学习理解 TCP/IP 协议栈代码实现。最后一个综合示例实现了使用 μ C/OS-III 操作系统来管理前面所讲解的硬件，充分利用操作系统的任务通信、消息传递等机制来提升硬件系统的运行效能。 μ C/OS-III 系统结构紧凑，代码量小，容易理解掌握，通过移植和应用 μ C/OS-III，使读者在掌握系统应用场景的同时，加深理解操作系统内部的工作机理，为后续进一步学习基于 Linux 的嵌入式开发打下基础。

由于社会信息化日趋明显，必定导致包括 STM32 在内的 MCU 的应用越来越多。希望本书能为渴望进入嵌入式开发领域的人员提供一个好的入门指引，为后续深入嵌入式开发的高级应用奠定基础。作为学习教材，本书每一章的实验代码都由笔者在 Keil MDK 开发环境中调试通过，读者可以放心学习使用。同时，由于笔者水平有限，书中难免会存在对相关知识点理解不够准确之处，敬请读者批评指正。

参与编写本书的人员还有我的同事易确，他负责本书所有的实验电路设计；熊立宇，负责完成最后三章的初稿编写及全书的校验工作。十分感谢他们的辛勤付出！

在本书的编写过程中，得到了电子工业出版社的田宏峰老师的悉心支持，在此表示衷心感谢，同时感谢他为我提供了一个这样施展自己特长机会；十分感谢我的同事彭丽兰，是她在我工作忙碌的时候，分担了我的工作，使我能够安心写作；最后想表达对我的家人，特别是朋友熊姬珠的谢意，是她们给予我精神上的鼓励，才使我得以完成这“马拉松”式的写作。

谭 贵

2017 年 8 月于深圳

CONTENTS

目录

第 1 章 开发利器：STM32 库和 MDK Keil

1.1 学习启航：闪烁的跑马灯	1
1.1.1 实验结果呈现	1
1.1.2 实验分析	2
1.1.3 配置 GPIO 引脚	5
1.1.4 实验控制逻辑	6
1.2 STM32 库结构和 CMSIS 标准	8
1.2.1 STM32 库层次结构	9
1.2.2 CMSIS 层次结构	9
1.2.3 STM32 库结构中的文件关系	10
1.2.4 STM32 库函数命名规则	13
1.2.5 STM32 库常见的几个状态类型	13
1.3 工程开发环境设置	14
1.3.1 有关 MDK	14
1.3.2 使用 MDK 建立工程的步骤	15

第 2 章 STM32 体系结构

2.1 总线与通信接口	25
2.1.1 总线组成	25
2.1.2 重要的总线术语	26
2.2 STM32 功能框架	27
2.2.1 系统组成	27
2.2.2 总线单元及挂接设备	28
2.3 STM32 存储器映射	29
2.3.1 独立编址	30
2.3.2 统一编址（存储器映像编址）	31
2.3.3 CM3 外设地址空间映射	32
2.3.4 地址空间映射详解	34
2.4 STM32 时钟结构	39
2.4.1 STM32F103ZET6 的时钟树	39

2.4.2 时钟树二级框架·····	40
2.4.3 时钟启用过程·····	41
2.5 系统时钟树与地址空间映射的关系·····	43

第3章

3.1 CM3 的复位序列·····	44
3.1.1 堆栈·····	45
3.1.2 向量表·····	47
3.2 STM32 启动代码分析·····	49
3.3 STM32 系统时钟初始化·····	52
3.3.1 时钟源的选择·····	52
3.3.2 系统时钟设置·····	56
3.4 程序运行环境初始化函数__main()·····	60
3.4.1 回顾编译和链接过程·····	60
3.4.2 映像文件的组成·····	61
3.4.3 映像的加载过程·····	63
3.4.4 由 MDK 集成环境自动生成的分散加载文件·····	65
3.4.5 __main()函数的作用·····	66

第4章

4.1 实验结果预览: LED 跑马灯·····	68
4.2 GPIO 基本知识·····	68
4.2.1 GPIO 分组管理及其引脚·····	69
4.2.2 GPIO 工作模式及其配置·····	69
4.2.3 GPIO 引脚的写入和读出·····	71
4.3 实验代码解析·····	74
4.3.1 实验现象原理分析·····	74
4.3.2 源代码分析·····	78
4.4 创建工程·····	81
4.4.1 建立工程目录结构·····	81
4.4.2 导入源代码文件·····	81
4.4.3 编译执行·····	82
4.5 编译调试·····	82
4.5.1 调试方法·····	82
4.5.2 栈和变量观察窗口·····	83
4.5.3 运行程序并调试: 一个函数一个断点·····	84
4.5.4 运行程序并调试: 多个函数多个断点·····	86

第5章

5.1 实验结果预览: LED 跑马灯_中断控制·····	90
-------------------------------	----

5.2	异常与中断	91
5.2.1	Cortex-M3 的异常向量	91
5.2.2	异常向量表	92
5.3	NVIC 与中断控制	93
5.3.1	NVIC 简述	93
5.3.2	NVIC 与外部中断	93
5.3.3	NVIC 中断的优先级	94
5.3.4	NVIC 初始化	95
5.4	EXTI 基本知识	97
5.4.1	EXTI 简介	97
5.4.2	EXTI 控制器组成结构	97
5.4.3	GPIO 引脚到 EXTI_Line 的映射	100
5.4.4	EXTI_Line 到 NVIC 的映射	102
5.5	实验代码解析	103
5.5.1	工程源码的逻辑结构	103
5.5.2	实验代码软硬件原理	104
5.5.3	实验代码分析	107
5.6	创建工程	109
5.6.1	建立工程目录结构	109
5.6.2	导入源代码文件	109
5.6.3	编译执行	110
5.7	编译调试	111
5.7.1	打开内存窗口	111
5.7.2	设置断点	111
5.7.3	运行程序并调试	112

第6章 USART 接口 115

6.1	实验结果预览	115
6.1.1	实验准备工作	115
6.1.2	实验现象描述	116
6.2	USART 基本知识	117
6.2.1	串行异步通信协议	117
6.2.2	USART 与接口标准 RS-232	118
6.3	STM32 USART 结构	119
6.3.1	USART 工作模式	119
6.3.2	精简的 USART 结构	119
6.3.3	USART 单字节收发过程	120
6.4	USART 寄存器位功能定义	121
6.4.1	状态寄存器 (USART_SR)	121

6.4.2	数据寄存器 (USART_DR)	122
6.4.3	控制寄存器 1 (USART_CR1)	122
6.4.4	控制寄存器 2 (USART_CR2)	123
6.4.5	控制寄存器 3 (USART_CR3)	123
6.4.6	分数波特率寄存器 USART_BRR	124
6.4.7	USART 模块寄存器组	125
6.4.8	USART 模块初始化函数	126
6.4.9	USART 常用函数功能说明	127
6.5	USART 实验代码分析	128
6.5.1	实验电路 (硬件连接关系)	128
6.5.2	工程源代码文件层次结构	130
6.5.3	应用层 (主程序控制逻辑)	131
6.5.4	用户驱动层	133
6.5.5	函数 printf() 重定向	135
6.6	创建工程	135
6.6.1	建立工程目录结构	135
6.6.2	创建文件组和导入源文件	136
6.6.3	编译执行	137

第 7 章 USART 综合应用：命令行外壳程序 Shell 138

7.1	实验结果预览	138
7.2	基于 USART 的 I/O 函数	139
7.2.1	字符及字符串获取函数: xgetc() 和 xgets()	139
7.2.2	字符及字符串打印函数: xputc() 和 xputs()	141
7.3	可变参数输出函数 xprintf()	142
7.3.1	可变参数	142
7.3.2	可变参数宏的使用与作用	143
7.3.3	用可变参数宏实现自己的格式化输出函数 xprintf()	144
7.4	Shell 外壳	145
7.4.1	Shell 命令管理结构	146
7.4.2	Shell 命令解析过程	147
7.4.3	命令函数之参数解析	150
7.5	建立工程, 编译和运行	151
7.5.1	创建和配置工程	151
7.5.2	编译执行	153

第 8 章 I2C 接口 154

8.1	实验结果预览：轮询写入/读出 EEPROM 数据	154
8.2	I2C 总线协议	155

8.2.1	总线特点	155
8.2.2	I2C 应用结构	155
8.2.3	总线信号时序分析	156
8.3	STM32 I2C 模块	158
8.3.1	I2C 组成框图	158
8.3.2	I2C 主模式工作流程	159
8.3.3	I2C 中断及 DMA 请求	161
8.4	I2C EEPROM 读写示例及分析	162
8.4.1	示例电路连接	162
8.4.2	app.c 文件中的 main() 函数	163
8.4.3	eeeprom.h 文件	166
8.4.4	eeeprom.c 文件	167
8.4.5	shell.c 文件	174
8.5	建立工程, 编译及运行	175
8.5.1	创建和配置工程	175
8.5.2	编译执行	176

第 9 章 DMA 接口

9.1	实验结果预览	177
9.2	通用 DMA 的作用及特征	178
9.3	STM32 DMA 基本知识	178
9.3.1	DMA 与系统其他模块关系图	178
9.3.2	STM32 DMA 组成	179
9.4	实验示例分析	183
9.4.1	main.c 文件中的 main() 函数	184
9.4.2	USART1 的初始化	184
9.4.3	DMA 通道中断处理函数	189
9.4.4	sysTick 中断处理函数	190
9.4.5	DMA 通道配置的其他寄存器	191
9.4.6	DMA 用户测试命令及其执行函数	192
9.5	建立工程, 编译和执行	193
9.5.1	建立以下工程文件夹	194
9.5.2	创建文件组和导入源文件	194
9.5.3	编译运行	194

第 10 章 实时时钟 RTC

10.1	实验结果预览	195
10.2	STM32 RTC 模块	196
10.2.1	STM32 后备供电区域	196

10.2.2	RTC 组成	199
10.3	RTC 实验设计与源码分析	204
10.3.1	硬件连接和 GPIO 资源	204
10.3.2	实验源代码逻辑结构	204
10.3.3	源代码分析	205
10.4	建立工程, 编译和执行	212
10.4.1	建立以下工程文件夹	212
10.4.2	创建文件组和导入源文件	212
10.4.3	编译执行	213

第 11 章 系统定时器 SysTick 214

11.1	SysTick 简述	214
11.2	SysTick 工作过程	214
11.3	SysTick 寄存器位功能定义	215
11.3.1	控制和状态寄存器: STK_CTRL	215
11.3.2	重载寄存器: STK_LOAD	216
11.3.3	当前计数值寄存器: STK_VAL	217
11.3.4	校正寄存器: STK_CALIB	217
11.3.5	SysTick 模块寄存器组	217
11.3.6	配置 SysTick 定时器	218
11.4	基于 SysTick 的延时函数代码分析	220
11.4.1	实现原理	220
11.4.2	实现代码分析	220
11.4.3	基于 SysTick 延时的 LED 闪烁命令	223
11.5	建立工程, 编译和执行	224
11.5.1	建立以下工程文件夹	224
11.5.2	创建文件组和导入源文件	224
11.5.3	编译运行	226

第 12 章 SPI 接口 227

12.1	实验现象预览: 轮询写入/读出 SPI Flash 数据	227
12.2	SPI 总线协议	228
12.2.1	总线信号及其应用结构	228
12.2.2	SPI 内部结构与工作原理	229
12.3	STM32 SPI 模块	231
12.3.1	SPI 组成框图	231
12.3.2	STM32 SPI 主模式数据收发过程	232
12.3.3	SPI 中断及 DMA 请求	234
12.4	W25Q128FV 规格说明	234

12.4.1	W25Q128FV 状态和控制管理	235
12.4.2	W25Q128FV 常用指令	236
12.5	程序入口与 SPI 初始化代码	237
12.5.1	实验硬件资源	237
12.5.2	工程入口文件 main.c	238
12.5.3	spiflash.c 文件中的 spiFlash_Init()函数	239
12.6	SPI Flash 测试代码分析	243
12.6.1	spiflash.c 文件中的 SPI Flash 测试函数 spiTest()	244
12.6.2	SPI Flash ID 读取函数 sFLASH_readID()	245
12.6.3	扇区擦除函数 sFLASH_eraseSector()	246
12.6.4	Flash 页写函数 sFLASH_writePage()	246
12.6.5	Flash 读函数 sFLASH_readBuffer()	247
12.6.6	Flash 字节发送函数 sFLASH_SendByte()	248
12.7	向 Shell 添加 SPI 测试指令 spitest	249
12.8	建立工程, 编译和执行	250
12.8.1	建立以下工程文件夹	250
12.8.2	创建文件组和导入源文件	250
12.8.3	编译运行	252

第 13 章 网络接口：以太网 253

13.1	网络体系结构简介	253
13.1.1	三种网络模型	253
13.1.2	以太网标准 (Ethernet)	256
13.2	ENC28J60 知识	257
13.2.1	ENC28J60 概述	257
13.2.2	控制寄存器	259
13.2.3	以太网缓冲器	260
13.2.4	PHY 寄存器	261
13.2.5	ENC28J60 SPI 指令集	261
13.2.6	ENC28J60 初始化	263
13.2.7	使用 ENC28J60 收发数据	268
13.2.8	ENC28J60 驱动代码总结	272
13.3	uIP 协议栈简介	274
13.3.1	uIP 特性	274
13.3.2	uIP 应用接口	275
13.3.3	uIP 的初始化及配置函数	277
13.3.4	uIP 的主程序循环	277
13.4	uIP 移植分析	279
13.4.1	下载 uIP1.0 版源码文件	279

13.4.2	理解两个中间层文件与应用层和协议层之间的关系	280
13.4.3	添加 uIP 协议栈后的工程文件组	285

第 14 章 综合示例：基于 uIP 的 Telnet 服务

14.1	实验现象预览	286
14.2	Telnet 远程登录协议	287
14.2.1	Telnet 概述	287
14.2.2	Telnet 协议主要技术	288
14.2.3	Telnet 命令	288
14.3	Telnetd 服务框架及实现	290
14.3.1	本实验 Telnetd 服务框架	290
14.3.2	Telnetd 服务框架的实现	291
14.4	上层应用与 uIP 协议的接口：telnetd_appcall()	304
14.5	建立工程，编译和运行	309
14.5.1	创建和配置工程	309
14.5.2	编译执行	311

第 15 章 SDIO 总线协议与 SD 卡操作

15.1	SD 卡简介	312
15.1.1	SD 卡家族	312
15.1.2	SD 卡引脚功能定义	313
15.1.3	SD 卡内部组成	314
15.1.4	SD 卡容量规格	315
15.1.5	SDIO 接口规范和总线应用拓扑	315
15.2	SD 协议	316
15.2.1	工作模式与状态	316
15.2.2	命令和响应格式	316
15.2.3	卡识别模式	317
15.2.4	数据传输模式	320
15.3	STM32 SDIO 控制器	322
15.3.1	控制器总体结构描述	322
15.3.2	SDIO 适配器模块	323
15.3.3	SDIO AHB 接口	325
15.4	工程入口及配置	326
15.4.1	实验硬件资源	326
15.4.2	工程入口文件 main.c	327
15.5	SDIO 初始化	328
15.5.1	SD 卡上电初始化函数 SD_PowerON()	330
15.5.2	SD 卡规格信息获取函数 SD_InitializeCards()	336

15.6	SDIO 卡测试代码分析	339
15.6.1	块擦除	340
15.6.2	多块写	342
15.6.3	多块读	345
15.7	建立工程, 编译和运行	348
15.7.1	建立以下工程文件夹	348
15.7.2	创建文件组和导入源文件	348
15.7.3	编译执行	349

第 16 章 移植文件系统 FatFs

16.1	实验现象预览: 基于 Shell 的文件系统命令	350
16.2	FatFs 文件系统	351
16.2.1	FatFs 特点	351
16.2.2	FatFs 在设备系统中的层次与接口	351
16.3	移植 FatFs 文件系统	352
16.3.1	FatFs 源代码结构	352
16.3.2	基于 FatFs 应用的常用数据类型说明	353
16.3.3	FatFs 的移植	355
16.4	FatFs 文件系统应用示例分析	357
16.4.1	工程源代码逻辑	357
16.4.2	工程源代码分析	358
16.5	建立工程, 编译和运行	363
16.5.1	创建和配置工程	363
16.5.2	编译执行	364

第 17 章 无线接入: Wi-Fi 模块 ESP8266 应用

17.1	无线技术标准: IEEE 802.11	365
17.1.1	IEEE 802.11 简介	365
17.1.2	无线局域网的组网拓扑	366
17.1.3	无线接入过程的三个阶段	367
17.2	ESP-WROOM-02 模组	368
17.2.1	ESP-WROOM-02 性能参数	368
17.2.2	ESP-WROOM-02 与主机系统的电路连接	369
17.3	ESP-WROOM-02 指令集	370
17.3.1	ESP8266 AT 常用指令	370
17.3.2	使用 ESP-WROOM-02 进行真实通信	373
17.4	封装 ESP-WROOM-02 的配置函数	375
17.4.1	ESP-WROOM-02 的初始化函数	375
17.4.2	ESP-WROOM-02 的配置函数	377

17.4.3	优化 USART 接收缓存的数据结构	379
17.4.4	ESP-WROOM-02 的 Shell 操作命令	381
17.5	建立工程, 编译和运行	384
17.5.1	工程程序文件	384
17.5.2	创建和配置工程	384
17.5.3	编译执行	385

第 18 章 移植 μ C/OS-III 操作系统 387

18.1	μ C/OS-III 基础	387
18.1.1	μ C/OS-III 简介	387
18.1.2	μ C/OS-III 内核组成架构	388
18.2	μ C/OS-III 任务基础	390
18.2.1	任务状态	390
18.2.2	任务控制块和就绪任务表	391
18.2.3	创建任务	391
18.2.4	任务同步与通信	393
18.3	μ C/OS-III 的信号量	393
18.3.1	信号量分类及其应用	393
18.3.2	信号量工作方式	394
18.3.3	信号量应用操作步骤	396
18.4	μ C/OS-III 的消息队列	396
18.4.1	消息队列工作模型	397
18.4.2	消息队列应用操作步骤	397
18.5	μ C/OS-III 的事件标志组	398
18.5.1	事件标志组工作模型	398
18.5.2	事件标志组应用操作步骤	399
18.6	信号量、消息队列和事件标志组综合示例	399
18.6.1	综合示例任务关系图	400
18.6.2	任务代码头文件 task.h	400
18.6.3	任务代码 C 文件 task.c	402
18.6.4	中断异常处理文件 stm32f10x_it.c	409
18.7	μ C/OS-III 移植	410
18.7.1	μ C/OS-III 源码组织架构	410
18.7.2	简化 μ C/OS-III 源码组织架构	411
18.7.3	建立基于 μ C/OS-III 的工程	412
18.7.4	μ C/OS-III 综合示例运行效果	414

第 19 章 基于 μ C/OS-III 的信息系统 415

19.1	系统功能描述	415
------	--------	-----

19.1.1 系统任务划分 415

19.1.2 系统实际运行效果 415

19.2 系统任务设计分析 417

19.2.1 Shell 任务 417

19.2.2 LED 灯闪烁任务 420

19.2.3 事件监测任务 420

19.2.4 系统统计任务 422

19.2.5 无线通信处理任务 425

19.3 工程源代码（文件）整合 428

19.3.1 主文件 main.c 428

19.3.2 任务头文件 task.h 428

19.3.3 includes.h 文件 429

19.3.4 任务实现文件 task.c 430

19.4 建立工程，编译和运行 430

19.4.1 建立工程源代码结构 430

19.4.2 建立文件组，导入源文件 430

19.4.3 编译执行 431

参考文献 432

第1章

开发利器：STM32 库和 MDK Keil

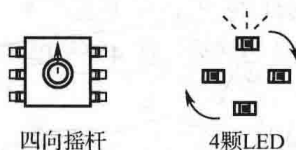
“工欲善其事，必先利其器”，对于干技术活的工程人员来说，对这句话有比一般人更深刻的理解。尤其是软件开发领域，可供使用的工具众多，选择一个好的开发平台，不但可以有效地管理工程文件、配置工程参数，而且可更快捷有效地发现并解决程序设计中出现的问题。

本章以一个简单有趣的外设实验为线索，为您讲解本书将要使用的“利器”——MDK Keil 开发环境的使用方法，包括工程建立、源文件管理、编译配置等。同时通过这个实验，为读者首先建立一个开发流程的初步印象，为后续学习打下基础，并一步步地进入趣味无穷的嵌入式世界。

1.1 学习启航：闪烁的跑马灯

对于任何刚开始接触嵌入式技术的人来说，对学习对象的直观感受是十分重要的。所谓“第一印象决定了以后的交往”，在技术领域也有这种效应。换句话说，首先得要让他对所学的东西有初步的兴趣。本节内容的目的就是如此，通过对一个四向（上下左右）摇杆不同方向的按钮来控制 4 颗 LED 灯闪烁频率和方向。

1.1.1 实验结果呈现



实验操作：

- 步骤 A：当将摇杆向上推时，4 颗 LED 灯以顺时针方向间隔 1 s 依次闪烁。
- 步骤 B：将摇杆向下推，4 颗 LED 以逆时针方向间隔 1 s 闪烁。
- 步骤 C：将摇杆向左推，加快 LED 的闪烁频率。
- 步骤 D：将摇杆向右推，降低 LED 的闪烁频率。