



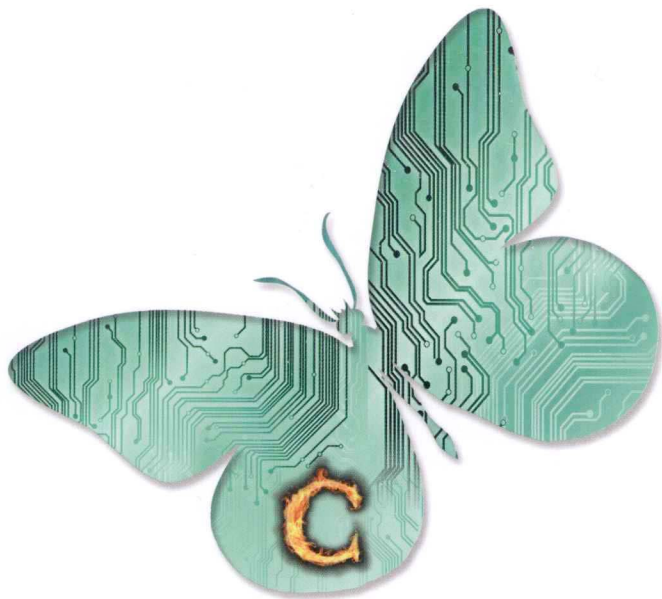
第一本深入剖析STM32官方库及其使用的权威指南

从流水灯剖析到 μ C/OS-III移植，零死角深入STM32库开发

配套业内最流行的野火STM32开发板，提供完整的工程文件和源代码，极具可操作性



单片机与嵌入式



库

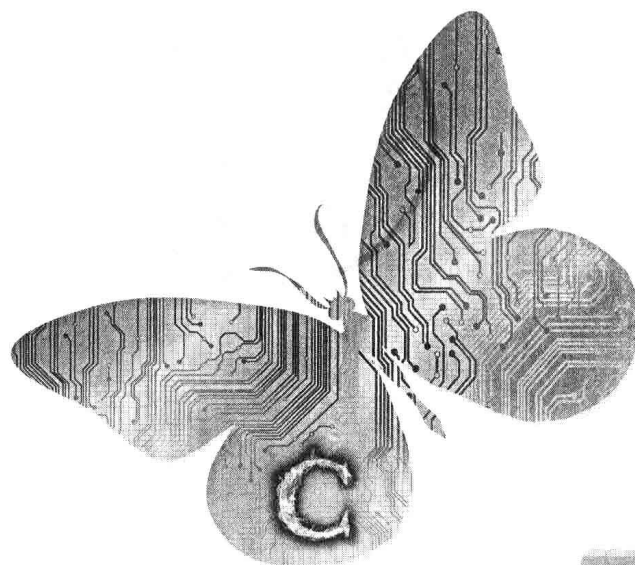
STM32库 开发实战指南

刘火良 杨森 编著



机械工业出版社
China Machine Press

单片机与嵌入式



库

STM32库 开发实战指南

刘火良 杨森 编著



机械工业出版社
China Machine Press

图书在版编目 (CIP) 数据

STM32库开发实战指南 / 刘火良, 杨森编著. —北京: 机械工业出版社, 2013.5

ISBN 978-7-111-42637-0

I. S… II. ①刘… ②杨… III. 微控制器 - 系统开发 - 指南 IV. TP332.3-62

中国版本图书馆CIP数据核字 (2013) 第109248号

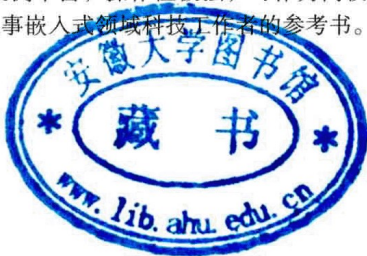
版权所有·侵权必究

封底无防伪标均为盗版

本书法律顾问 北京市展达律师事务所

本书基于STM32F103芯片, 紧紧围绕“库”的分析和使用展开。在大量实例的基础上, 本书对于如何综合运用固件库开发项目给出了具体的范例; 在固件库的使用和学习的基础上, 又进一步讲解了结合嵌入式实时操作系统、TCP/IP协议栈进行嵌入式系统开发的方法, 让读者循序渐进、系统地掌握基于STM32官方库进行开发的方法。

本书内容翔实, 案例丰富, 操作性极强, 可作为高校电子信息、通信工程、信息工程等相关专业的教材, 也适合作为从事嵌入式领域科技工作者的参考书。



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: 余洁

北京市荣盛彩色印刷有限公司印刷

2013年6月第1版第1次印刷

186mm × 240mm · 31印张

标准书号: ISBN 978-7-111-42637-0

ISBN 978-7-89433-959-1 (光盘)

定 价: 69.00元 (附光盘)

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) 88378991 88361066

投稿热线: (010) 88379604

购书热线: (010) 68326294 88379649 68995259

读者信箱: hzsj@hzbook.com

前 言

单片机是对 8/16 位 MCU（微控制器）的另外一种叫法。传统的 8/16 位单片机，久经岁月的洗礼，仍然在工业控制应用中大放光芒。然而，现在的工程师面对的更多的工业控制产品需求是多功能、易用界面、低功耗以及多任务等。基于这样的需求，以往的 8/16 位单片机已不能满足要求，工程师必须寻找新的符合要求的 MCU。工程师虽然可以选择诸如 ARM7、ARM9 这类速度更快的 32 位 MCU，但是鉴于对成本和开发门槛等种种考虑，它们还是不能满足需求。正是看准了这个市场，ARM 公司推出了其全新的基于 ARMv7 架构的 32 位 Cortex-M3 微控制器内核。紧随其后，ST（意法半导体）公司就推出了基于 Cortex-M3 内核的 MCU——STM32。STM32 凭借其产品线的多样化、极高的性价比、简单易用的开发方式，迅速在众多 Cortex-M3 MCU 中脱颖而出，成为最闪亮的一颗新星。STM32 一上市就迅速占领了中低端 MCU 市场，颇有星火燎原之势，这与它倡导的基于固件库的开发方式密不可分。采用库开发的方式可以快速上手，仅通过调用库里面的 API（应用程序接口）就可以迅速搭建一个大型的程序，写出各种用户所需的应用，这就大大降低了学习的门槛和开发周期。然而，又因为在开发中只是调用 API，而忽略了底层寄存器的操作，库开发被习惯了寄存器开发方式的工程师指为“于浮沙筑高台”，没有学习的价值。这种看法具有一定的片面性，他显然没有意识到这是一种全新的学习方法。试问，对于初学者，面对一个 32 位且有如此多寄存器的单片机，如果还像我们以往操作 8 位机，通过配置寄存器的方式来实现，那会是多么繁杂的一项工作？除此之外，库的开发方式自顶向下，它是迈向更高端嵌入式 Linux 开发的一个垫脚石。

库开发已成主流，这是不争的事实。STM32 固件库之所以流行并被大家所喜爱，可以归结为以下两个原因：

（1）技术潮流

1) 于个人：库开发大大地降低了学习的门槛，提高了学习的效率，使个人初步了解了大的程序设计，是一种自顶向下的学习方法，可以从上层的 API 层层跟踪到底层，可以彻彻底底地了解寄存器，了解 CPU 的内存分布，再到启动代码、开发环境的配置等。如果再深究，还会涉及编译器甚至工具链。库的学习，可不仅仅是简单地调用 API，我们需要去分析这个库是如何构建的，是如何从内存到寄存器，寄存器到结构体，结构体到更各层的 API，再到层层外设的文件关

联。这里面涉及了太多的 C 语言的知识，如关键字、宏、结构体、指针、类型转换、条件编译、断言、内联函数等。这些知识的学习，又岂能说是“于浮沙筑高台”。如果你的 C 语言还停留在基本语法的阶段，那么通过对库的使用和学习，你的 C 语言将会得到脱胎换骨的提升。学会了库开发还可以快速地迁移到 ST 其他系列单片机的学习，如 STM32F207、STM32F407，这些 MCU 的固件库基本上都是兼容的。反观如此庞大的固件库，还要相互兼容，细心的人一定可以从中获益良多！

2) 于公司：在公司产品开发中，产品上市速度是非常重要的成功因素，库开发可以极大地缩短产品研发周期，以便快速抢占市场。而且库让程序的维护成本更低，程序的升级更快捷。用库来开发，真可谓事半功倍，一箭双雕。

(2) 市场趋势

有人曾经质疑 STM32 的固件库降低了 MCU 的性能，然而，他却并没有考虑到 STM32 的性能和资源已经不是传统的 8/16 单片机可比的了。强大的硬件应该与消耗这些资源的软件相匹配，否则资源就被浪费了。硬件和软件是相辅相成、共同促进的。所以硬件改善后，工程师对于 MCU 的关注应该从全局着眼。

本书采用 MDK 开发环境，全部例程基于 3.5.0 版本的固件库讲解，不是简单地调用库，而是试图通过对固件库的使用详细讲解什么是库、为什么使用库、怎样使用库等一系列问题，进而引导读者使用高效率的库开发方法。

本书采用原理分析、代码讲解、实验运用这三点连线的讲解方式，循序渐进，适合在校大学生和科研机构开发人员学习使用。全书分为四个部分，第一部分（第 1～5 章）是库开发初级篇，涉及入门的两个主题。一个是嵌入式工程师成长之路，属于方法论的问题，涉及了一个工程师从学生时代开始，在每一个不同的阶段应该学习什么、该如何进阶等。另一个是通过对库的了解和 GPIO 的学习，让读者快速掌握 STM32 的开发方法，这是入门的第一步。第二部分（第 6～16 章）是库开发中级篇，讲解了 STM32 各个外设的使用，是学习的一个进阶阶段，也是 STM32 学习的重中之重。第三部分（第 17～25 章）是库开发高级篇，是 STM32 各个外设的实战演练，如 MP3、液晶、摄像头、Wi-Fi 等，是属于项目实战的例子，一般可直接用于工程项目的开发。第四部分（第 26～28 章）是库开发系统篇，这是嵌入式系统开发的必经之路，是区别裸与不裸的分水岭；这部分讲解了 μ C/OS 最新版本 μ C/OS-III 在 STM32 中的移植，通过该移植应用实践，相信可以为以后进阶到 WinCE、Linux 操作系统的学习打下坚实的基础。

致 谢

首先要感谢本书的策划编辑张国强先生，是他对 STM32 的关注促成了这本书的出版，同时在我们撰写书稿时对本书提出了宝贵的写作建议，并对书稿进行了仔细审阅。其次要感谢野火

工作室的成员廖锦松、曾云清等人提供的部分例程及资料；感谢好友何卓波对书稿内容的校正工作。

由于本书涉及的知识面广，时间又仓促，限于笔者的水平和经验，疏漏之处在所难免，恳请专家和读者批评指正，可以发送邮件到 wildfireteam@163.com 与作者进行交流，或者到阿莫论坛野火 M3 专区 <http://www.amobbs.com> 进行讨论。

刘火良 杨森

目 录

前 言

第一部分 库开发初级篇

第 1 章 为什么学习 STM32 2

- 1.1 嵌入式技术知识结构..... 2
- 1.2 嵌入式工程师成长之路..... 3
- 1.3 为什么学习 STM32..... 4
- 1.4 如何学习 STM32..... 4

第 2 章 初识 STM32 固件库 5

- 2.1 STM32 神器之库开发..... 5
 - 2.1.1 什么是 STM32 库..... 5
 - 2.1.2 为什么采用库开发..... 6
- 2.2 STM32 结构及库层次关系 7
 - 2.2.1 CMSIS 标准 7
 - 2.2.2 库目录、文件简介..... 8
 - 2.2.3 STM32 固件库文件间的关系 14
 - 2.2.4 使用库帮助文档..... 15

第 3 章 GPIO 入门之流水灯 18

- 3.1 安装 MDK 18
- 3.2 建立工程模板 19
 - 3.2.1 新建工程 19
 - 3.2.2 配置 J-LINK 硬件调试 25

3.3 如何编译和下载程序..... 27

- 3.3.1 如何编译程序..... 27
- 3.3.2 如何下载程序..... 27

第 4 章 深入分析流水灯例程 30

- 4.1 STM32 的 GPIO..... 30
- 4.2 STM32 的地址映射..... 33
 - 4.2.1 温故而知新
——stm32f10x.h 文件..... 33
 - 4.2.2 外设基地址 35
 - 4.2.3 总线外设基地址 36
 - 4.2.4 寄存器组基地址 37
- 4.3 STM32 固件库对寄存器的封装 38
- 4.4 STM32 的时钟系统..... 39
 - 4.4.1 时钟树 & 时钟源 39
 - 4.4.2 高速外部时钟 41
 - 4.4.3 HCLK、FCLK、PCLK1、
PCLK2 42
- 4.5 LED 具体代码分析 42
 - 4.5.1 实验描述及工程文件清单 42
 - 4.5.2 配置工程环境 43
 - 4.5.3 编写用户文件 44
 - 4.5.4 初始化结构体
——GPIO_InitTypeDef 类型 46
 - 4.5.5 初始化库函数——GPIO_Init() 47

4.5.6	开启外设时钟	48
4.5.7	控制 I/O 输出高、低电平	52
4.5.8	led.h 文件	52
4.5.9	main 文件	53
4.6	GPIO_Init() 函数的实现	55
4.6.1	规范的位操作方法	55
4.6.2	GPIO_Init() 实现代码分析	55
4.6.3	再论开发方式	60
4.7	开发步骤总结	61

第 5 章 调试程序 62

5.1	MDK 软件仿真调试	62
5.2	使用 J-LINK 进行硬件调试	64
5.2.1	硬件调试	64
5.2.2	软件编译过程	65
5.3	MDK 使用小技巧	66

第二部分 库开发中级篇

第 6 章 GPIO 再举例之

按键实验 70

6.1	GPIO 的 8 种工作模式	70
6.1.1	4 种输入模式	71
6.1.2	4 种输出模式	71
6.2	按键实验分析	72
6.3	按键代码分析	72
6.3.1	实验描述及工程文件清单	72
6.3.2	配置工程环境	73
6.3.3	main 文件	73
6.3.4	GPIO 初始化配置	74
6.3.5	利用固件库的数据类型	75
6.3.6	实现 LED 反转	77
6.3.7	实验现象	77

第 7 章 EXTI 之按键中断实验 78

7.1	STM32 的中断和异常	78
7.2	NVIC 中断控制器	81
7.2.1	NVIC 结构体成员	81
7.2.2	抢占优先级和响应优先级	82
7.2.3	NVIC 的优先级组	83
7.3	EXTI 外部中断	83
7.4	中断检测按键实验分析	84
7.4.1	实验描述及工程文件清单	84
7.4.2	配置工程环境	85
7.4.3	main 文件	86
7.4.4	配置外部中断	86
7.4.5	AFIO 时钟	87
7.4.6	NVIC 初始化配置	88
7.4.7	EXTI 初始化配置	89
7.4.8	编写中断服务函数	89
7.4.9	实验现象	91

第 8 章 串口通信 (USART) 92

8.1	异步串口通信协议	92
8.2	直通线和交叉线	93
8.3	串口工作过程分析	94
8.3.1	波特率控制	94
8.3.2	收发控制	96
8.3.3	数据存储转移	96
8.4	串口通信实验分析	96
8.4.1	实验描述及工程文件清单	96
8.4.2	配置工程环境	97
8.4.3	main 文件	97
8.4.4	USART 初始化配置	98
8.4.5	printf() 函数重定向	101
8.4.6	USART1_printf() 函数	103
8.4.7	实验现象	106

第 9 章 库函数开发小结	107	第 12 章 SysTick (系统滴答定时器)	139
9.1 初始化	107	12.1 SysTick——操作系统的心跳	139
9.2 数据输入输出	108	12.2 SysTick 工作分析	140
9.3 状态位、标志位	108	12.3 使用 SysTick 精确延时 实验分析	141
9.3.1 事件	109	12.3.1 实验描述及工程文件清单	142
9.3.2 标志位的检查与清除	109	12.3.2 配置工程环境	142
9.4 外设函数分类	110	12.3.3 main 文件	143
第 10 章 DMA——为 CPU 减负	112	12.3.4 配置并启动 SysTick	143
10.1 DMA 功能简介	112	12.3.5 定时时间的计算	147
10.2 DMA 工作分析	112	12.3.6 编写中断服务函数	147
10.3 DMA 实例之串口通信	113	12.3.7 使用 SysTick 测量时间的 功能	149
10.3.1 实验描述及工程文件清单	113	12.3.8 实验现象	149
10.3.2 配置工程环境	114	第 13 章 STM32 定时器	150
10.3.3 main 文件	114	13.1 定时器功能简介	150
10.3.4 DMA 初始化	115	13.2 定时器工作分析	150
10.3.5 使用 DMA 中断	121	13.2.1 基本定时器	150
10.3.6 实验现象	123	13.2.2 通用定时器	150
第 11 章 ADC 实验 (DMA 方式)	124	13.2.3 高级定时器	155
11.1 ADC 简介	124	13.3 PWM 输出实例分析	157
11.2 STM32 的 ADC 主要技术指标	124	13.3.1 实验描述及工程文件清单	157
11.3 ADC 工作过程分析	125	13.3.2 配置工程环境	157
11.4 ADC 采集数据实例 (采用 DMA 模式)	126	13.3.3 main 文件	158
11.4.1 实验描述及工程文件清单	127	13.3.4 定时器初始化	159
11.4.2 配置工程环境	128	13.3.5 实验现象	164
11.4.3 main 文件	128	第 14 章 I²C 接口	168
11.4.4 ADC 初始化	129	14.1 I ² C 协议简介	168
11.4.5 计算电压值	138	14.1.1 物理层	168
11.4.6 实验现象	138		

14.1.2	协议层	169
14.2	STM32 的 I ² C 特性及架构	170
14.2.1	I ² C 接口特性	170
14.2.2	I ² C 架构	170
14.3	I ² C 接口读写 EEPROM 实验	171
14.3.1	实验描述及工程文件清单	171
14.3.2	配置工程环境	171
14.3.3	main 文件	172
14.3.4	I ² C 接口初始化	173
14.3.5	对 EEPROM 的读写操作	177
14.3.6	使用 I ² C 读写 EEPROM	
	流程总结	186
14.3.7	实验现象	186
第 15 章 SPI 模块 188		
15.1	SPI 协议简介	188
15.1.1	SPI 信号线	188
15.1.2	SPI 模式	189
15.2	STM32 的 SPI 特性及架构	190
15.2.1	STM32 的 SPI 特性	190
15.2.2	STM32 的 SPI 架构分析	190
15.3	SPI 接口读取 Flash 实例分析	191
15.3.1	实验描述及工程文件清单	192
15.3.2	配置工程环境	193
15.3.3	main 文件	193
15.3.4	SPI 初始化	195
15.3.5	控制 Flash 的命令	199
15.3.6	读取厂商 ID	202
15.3.7	擦除 Flash 内容	203
15.3.8	向 Flash 写入数据	207
15.3.9	从 Flash 读取数据	210
15.3.10	小结	211
15.3.11	实验现象	211

第 16 章 CAN 控制器 212

16.1	CAN 协议简介	212
16.1.1	物理层	212
16.1.2	CAN 的报文种类及结构	213
16.1.3	同步	215
16.2	STM32 的 CAN 特性及架构	217
16.2.1	CAN 特性	217
16.2.2	CAN 架构	218
16.3	双 CAN 通信实验分析	219
16.3.1	实验描述及工程文件清单	219
16.3.2	配置工程环境	220
16.3.3	main 文件	221
16.3.4	配置 CAN 接口	223
16.3.5	打包报文	232
16.3.6	发送报文	234
16.3.7	接收报文、编写中断 服务函数	234
16.3.8	实验小结	236
16.3.9	实验现象	237

第三部分 库开发高级篇

第 17 章 SDIO 之 SD 卡驱动 240

17.1	SD 协议简介	240
17.1.1	卡的种类	240
17.1.2	SDIO 基本架构	241
17.2	STM32 的 SDIO 接口	241
17.2.1	从 SDIO 的时钟说起	242
17.2.2	SDIO 的命令格式	242
17.2.3	数据传输格式	243
17.3	SD 卡读写实验分析	243
17.3.1	实验描述及工程文件清单	243
17.3.2	配置工程环境	244

17.3.3	main 文件	246
17.3.4	SDIO 初始化	247
17.3.5	卡的上电识别流程	249
17.3.6	卡的初始化流程	256
17.3.7	对 SD 卡进行读写	259
17.3.8	原版官方驱动例程的 bug	263
17.3.9	实验现象	264

第 18 章 文件系统之 FATFS_R0.09 265

18.1	什么是文件系统	265
18.2	FATFS 文件系统简介	266
18.2.1	FATFS 的目录结构	266
18.2.2	FATFS 帮助文档	266
18.2.3	FATFS 源码	267
18.3	移植 FATFS 文件系统实验	267
18.3.1	实验描述及工程文件清单	267
18.3.2	配置工程环境	269
18.3.3	为文件系统添加底层驱动	270
18.3.4	添加简体中文和长文件名支持	274
18.3.5	main 文件	274
18.3.6	实验现象	277

第 19 章 MP3 播放器 278

19.1	MP3 文件探秘	278
19.1.1	文件格式	278
19.1.2	MP3 文件的原始数据	278
19.1.3	MP3 文件格式	279
19.2	VS1003 硬件解码芯片	279
19.2.1	VS1003 芯片简介	280
19.2.2	TDA1308 芯片	280
19.3	MP3 播放器实验	280
19.3.1	实验描述及工程文件清单	280

19.3.2	配置工程环境	282
19.3.3	main 文件	283
19.3.4	控制 VS1003 进入准备状态	284
19.3.5	播放 MP3 文件	286
19.3.6	STM32 的堆栈	291
19.3.7	实验现象	294

第 20 章 USB 大容量存储器实例 295

20.1	USB 协议分析	295
20.1.1	协议版本	295
20.1.2	USB 电气特性	295
20.1.3	USB 通信模型	296
20.1.4	USB 枚举	298
20.2	STM32 的 USB 控制器	299
20.3	USB 读取 SD 卡——模拟 U 盘实验	301
20.3.1	实验描述及工程文件清单	301
20.3.2	配置工程环境	302
20.3.3	USB 固件库说明	303
20.3.4	main 文件	305
20.3.5	基本配置	306
20.3.6	USB 初始化	308
20.3.7	中断服务函数	310
20.3.8	BOT 和 SCSI 协议	313
20.3.9	实验现象	316

第 21 章 LCD 触摸屏画板 317

21.1	LCD 控制器简介	317
21.1.1	ILI9341 控制器结构	317
21.1.2	像素点的数据格式	317
21.1.3	ILI9341 的通信时序	319
21.2	用 STM32 驱动 LCD	320
21.2.1	FSMC 简介	320

21.2.2 用 FSMC 模拟 8080 时序	322	23.2 OV7670 介绍	366
21.3 触摸屏感应原理	322	23.2.1 OV7670 功能框架	366
21.4 TSC2046 触摸屏控制器	323	23.2.2 OV7670 管脚封装	367
21.5 LCD 触摸屏画板实验	323	23.3 SCCB 总线	368
21.5.1 实验描述及工程文件清单	323	23.3.1 SCCB 接口定义	368
21.5.2 配置工程环境	325	23.3.2 SCCB 时序描述	370
21.5.3 main 文件	326	23.4 摄像头模块	372
21.5.4 初始化 FSMC 模式	327	23.4.1 摄像头模块硬件介绍	372
21.5.5 FSMC 模拟 8080 读写 参数、命令	332	23.4.2 OV7670 输出时序	372
21.5.6 液晶屏画点函数	334	23.4.3 FIFO 时序	375
21.5.7 触摸屏校正	338	23.4.4 摄像头的驱动原理	376
21.5.8 检测触点、画点	341	23.5 摄像头驱动实验	377
21.5.9 实验现象	342	23.5.1 实验描述及工程文件清单	377
第 22 章 字库及 BMP 图片 显示	343	23.5.2 配置工程环境	379
22.1 什么是字模	343	23.5.3 main 文件	379
22.2 制作字模	344	23.5.4 SCCB 总线的软件实现	380
22.3 BMP 图片格式	347	23.5.5 初始化 OV7670	386
22.4 显示中英文及 BMP 图片实验	351	23.5.6 采集并显示图像	388
22.4.1 实验描述及工程文件清单	351	23.5.7 实验现象	393
22.4.2 配置工程环境	352	第 24 章 以太网及 LwIP 协议栈 移植	394
22.4.3 main 文件	352	24.1 互联网模型	394
22.4.4 显示汉字	353	24.2 以太网	395
22.4.5 在 SD 卡上读取与保存 BMP 图像	358	24.2.1 PHY 层	395
22.4.6 实验现象	364	24.2.2 MAC 子层	396
第 23 章 OV7670 摄像头驱动	365	24.2.3 以太网控制器	397
23.1 摄像头的分类	365	24.3 MAC 之上的网络层	398
23.1.1 数字摄像头与模拟摄像头的 区别	365	24.3.1 为什么在 MAC 之上 还有分层	398
23.1.2 CCD 与 CMOS 的区别	365	24.3.2 TCP/IP 协议中各层次的 功能	398
		24.3.3 LwIP 协议栈	400
		24.4 ENC28J60+LwIP 以太网实验	401

24.4.1	实验描述及工程文件清单	401
24.4.2	配置工程环境	402
24.4.3	main 文件	403
24.4.4	LwIP 对底层数据结构的封装	404
24.4.5	初始化协议栈	408
24.4.6	LwIP 对底层操作的封装	410
24.4.7	轮询和计时	415
24.4.8	opt.h 文件和 debug	416
24.4.9	LwIP 应用	420
24.4.10	网页服务器	421
24.4.11	实验现象	426

第 25 章 Wi-Fi 模块 EMW3180 驱动 430

25.1	资料与工具下载	430
25.2	EMW3180 简介	430
25.3	EMW3180 驱动实验	434
25.3.1	实验描述及工程文件清单	434
25.3.2	配置工程环境	435
25.3.3	EMSP_API 函数	435
25.3.4	API 函数一览	436
25.3.5	main 文件	439
25.3.6	em380c_hal.c 文件	441
25.3.7	实验现象	445

第四部分 库开发系统篇

第 26 章 $\mu\text{C}/\text{OS-III}$ 及其源代码介绍 448

26.1	$\mu\text{C}/\text{OS}$ 简介	448
------	----------------------------	-----

26.1.1	操作系统与裸机的区别	448
26.1.2	$\mu\text{C}/\text{OS}$ 实时操作系统	448
26.2	$\mu\text{C}/\text{OS-III}$ 与 $\mu\text{C}/\text{OS-II}$ 的主要区别	450
26.3	$\mu\text{C}/\text{OS-III}$ 源码	450
26.4	$\mu\text{C}/\text{OS-III}$ 工程架构	452

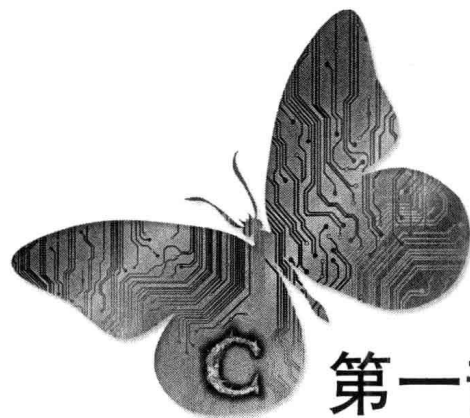
第 27 章 移植 $\mu\text{C}/\text{OS-III}$ 到 STM32 454

27.1	搭建 $\mu\text{C}/\text{OS}$ 工程文件结构	454
27.2	修改 $\mu\text{C}/\text{OS}$ 代码	459
27.2.1	修改 os_cpu.h 文件	459
27.2.2	修改 os_cpu_c.c	459
27.2.3	修改 os_cpu_a.asm 文件	460
27.2.4	修改 cpu_a.asm 文件	461
27.2.5	修改 startup_stm32f10x_hd.s 文件	462
27.2.6	修改 stm32f10x_it.c 文件	463
27.3	编写用户文件	464
27.3.1	编写 includes.h 文件	464
27.3.2	编写 BSP 相关文件	465
27.3.3	创建任务	466
27.4	配置 $\mu\text{C}/\text{OS-III}$	468

第 28 章 运行多任务 473

28.1	创建用户任务	473
28.2	编写用户代码	476
28.3	任务执行流程	479

参考文献 482



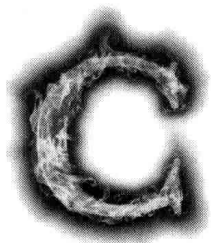
第一部分 库开发初级篇

大多数技术书籍会告诉读者如何掌握某一门技术，但少有讨论与职业规划相关的问题，例如应该学什么以及为什么学。在本书的第一部分我们将与大家共同探讨这些问题，引导读者思考为什么学习 STM32。

初级篇可以帮助初学者快速上手 STM32，以点亮 LED 灯的实例，从软件工程的角度深入剖析什么是固件库、为什么使用固件库和怎样使用固件库；从 STM32 固件库、新建工程、编译和下载程序出发，了解如何操作 GPIO，让新手步步为营，尽享 STM32 的学习乐趣。

我们对初学者的要求是具有基本的单片机基础，如 51、AVR 等，曾使用 C 语言写过单片机程序，但不需精通。读者在学习 STM32 的时候，无需太担心自己的基础，学习这门技术的决心和持之以恒的耐心才是掌握这门技术的法宝。

- ❑ 第 1 章 为什么学习 STM32
- ❑ 第 2 章 初识 STM32 固件库
- ❑ 第 3 章 GPIO 入门之流水灯
- ❑ 第 4 章 深入分析流水灯例程
- ❑ 第 5 章 调试程序



第 1 章

为什么学习 STM32

本章系统介绍了嵌入式工程师的技术成长路线，并详细介绍技术路线中的岗位设置和知识结构，让读者对于嵌入式工程师有一个全面系统地了解，并在此基础上引导工程师规划自己职业生涯。在本章的最后，作为承上启下的内容，从为什么学习 STM32 和如何学习 STM32 这两个话题入手，引导读者开始对于 STM32 的库开发方式有个初步地理解。

1.1 嵌入式技术知识结构

嵌入式技术是专用计算机系统技术，它以应用为核心，以计算机技术为基础，软硬件均可裁剪，适用在对功能、稳定性、功耗有严格要求的系统之中。嵌入式技术的开发人员需要对整个计算机体系（从底层硬件到软件操作系统）都有了解，而在这个体系之中，每个部分都可以分出一些小领域，因而技术要求很高，见图 1-1。

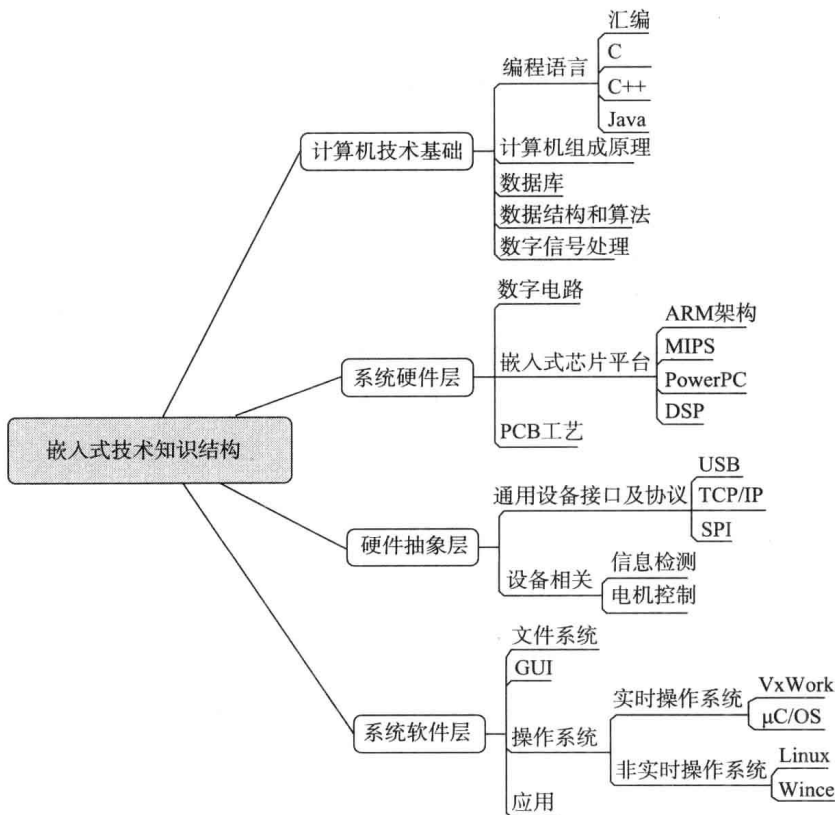


图 1-1 嵌入式技术知识结构

这个图只是粗略地概括了嵌入式技术的知识结构，但从中已经可以看出它涉及的知识面非常广，难怪众多学生甚至技术人员总是“迷茫”。不少电子专业出身的嵌入式技术人员主要从事硬件抽象层（中间层）的开发，这一层是沟通嵌入式系统的硬件层和软件操作系统的桥梁，因而主要的工作是开发驱动程序、板级应用支持、协调软硬件的开发，因而对软硬件都要有深入的了解。

1.2 嵌入式工程师成长之路

1. 从学生成为工程师

若希望从事硬件抽象层的开发，应该如何学习这些知识，才能从学生过渡到工程师呢？见图 1-2，对于希望成为其他方向的嵌入式技术人员也可以参考。

从图 1-2 可以看出，越往上层深入，就越接近于纯软件开发，但这并不代表嵌入式技术人员就不需要了解硬件，相反，上层的知识都是以底层为基础的，很多人说的“做嵌入式软件开发至少要读懂原理图”就是这个道理。

2. 职业规划

在嵌入式技术领域的公司，除了工程师还分很多职业岗位。一般公司的研发部门职位见图 1-3。

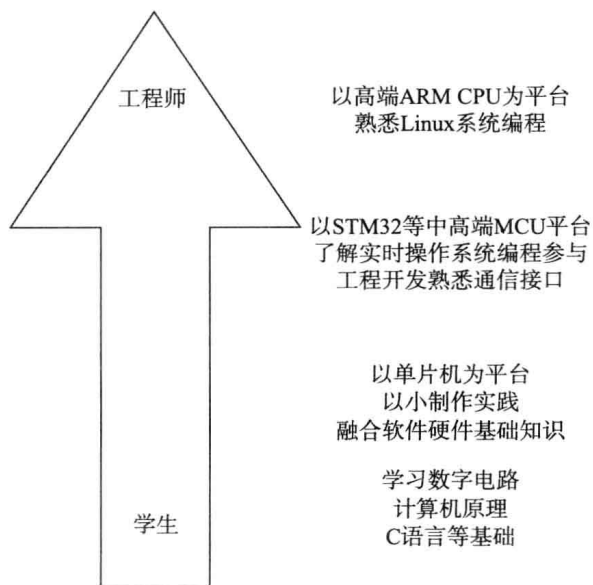


图 1-2 从事硬件抽象层开发的工程师成长之路

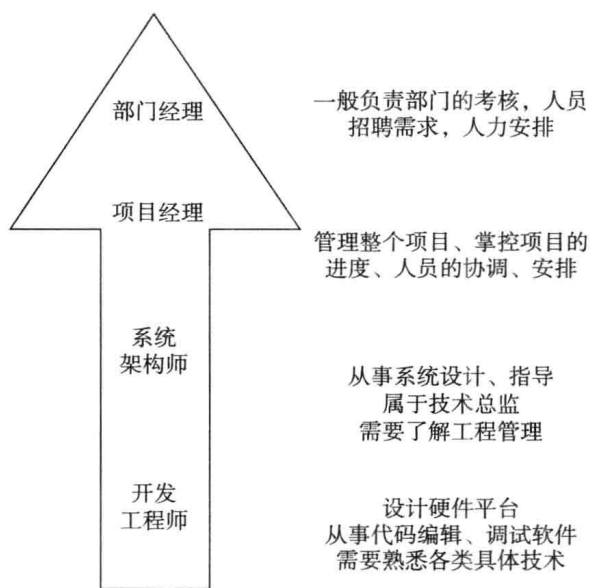


图 1-3 嵌入式工程师的职业成长路线

一般需要 3 ~ 5 年过渡到下一级的岗位，在小公司里项目经理一般也兼任部门经理。部门经理不一定要懂技术，并不是非由项目经理升职而成。直接与技术相关的是开发工程师和系统架构师，开发工程师会针对嵌入式技术的不同领域有不同的区分。在小公司里，熟悉软硬件的跨领域工程师很受欢迎，而大公司则分工明确，更看重在某领域研究得深入的开发工程师。作为系统架构师，则需要熟悉整个嵌入式领域，能够协调不同领域的开发工程师进行项目开发。

对于职业规划，不同的人有不同的见解，情况千差万别，以上所述仅供读者参考。

1.3 为什么学习 STM32

可以发现，在嵌入式领域 STM32 芯片介于低端和高端之间，它相对于普通的 8/16 位机有更多的片上外设，更先进的内核架构，可以运行 $\mu\text{C}/\text{OS}$ 等实时操作系统；相对于可运行 Linux 操作系统的高端 CPU，其成本低，实时性强。这个定位使得 STM32 不仅占领了大部分中端控制器的市场，更是成为提升开发者技术的优良过渡平台，为后续的学习打下坚实的基础。

1.4 如何学习 STM32

因为 STM32 的开发方式较普通的单片机开发还是有很大的不同，所以学习时要注意如下几点：

- 1) 转变思维，适应使用固件库的开发方式，加强运用 C 语言的能力，建立工程意识。
- 2) 熟悉 Cortex-M 系列芯片架构，了解 CMSIS 标准，熟悉 STM32 的总线架构。
- 3) 掌握 I²C、SPI、SDIO、CAN、TCP/IP 等各种通信协议，掌握了这些协议，开发软件驱动就变得相对容易了。

上面有关的内容本书都会详细讲解，但“纸上得来终觉浅，绝知此事要躬行”，读者亲自编程实践是不能少的。

初级篇可以帮助初学者快速上手 STM32，写出自己的应用程序。以点亮 LED 灯的实例，从软件工程的角度深入剖析什么是固件库、为什么使用固件库和怎样使用固件库；从 STM32 固件库、新建工程、编译和下载程序出发，了解如何操作 GPIO，让新手步步为营，尽享 STM32 的学习乐趣。

我们对初学者的要求是具有基本的单片机基础，如 51、AVR 等，曾使用 C 语言写过单片机程序，但不需精通。读者在学习 STM32 的时候，无需太担心自己的基础，我们更需要的是学习的勇气，需要的是拿下 STM32 的决心。试问，我们刚开始学习最简单的单片机的时候，是不是也没基础呢，是不是因此就停止了自己学习的脚步了呢？不是的。我们需要做的是认定一个目标，行动起来，坚持朝向目标的苦行，其中艰辛芳华，唯你自知。