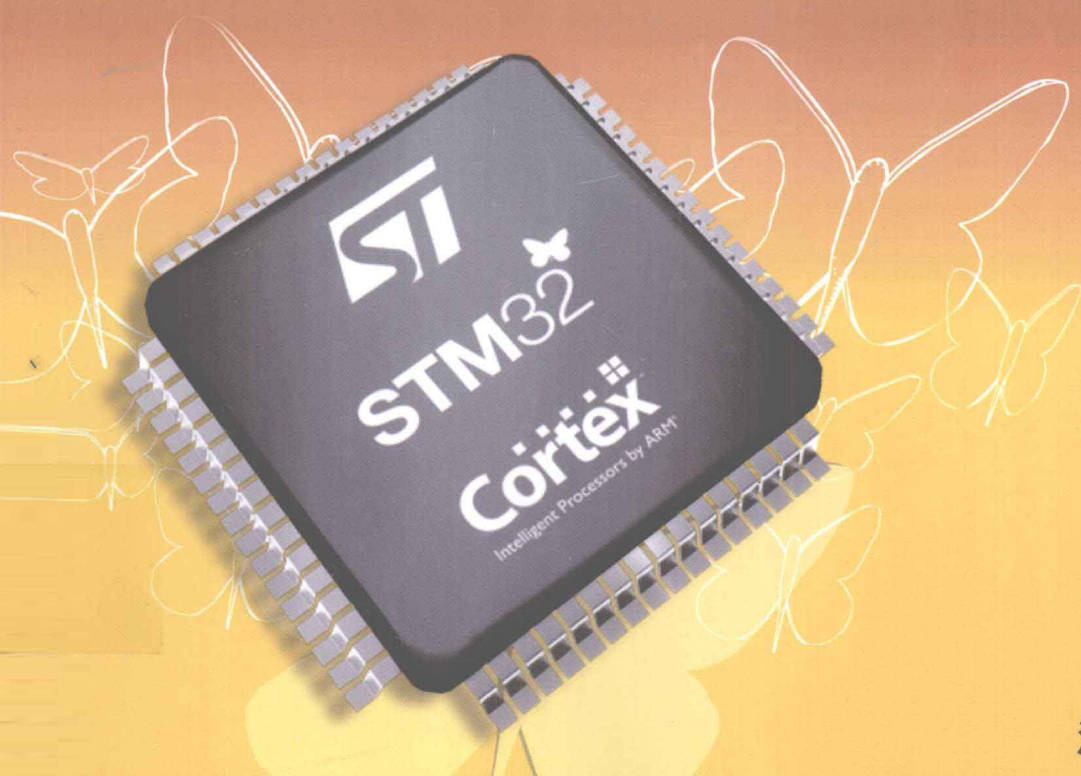


STM32F系列ARM Cortex-M3核 微控制器开发与应用

喻金钱 喻斌 编著

ARM Cortex-M3 ARM Cortex-M3 ARM Cortex-M3 ARM Cortex-M3 ARM Cortex-M3



清华大学出版社

STM32F 系列 ARM Cortex-M3 核 微控制器开发与应用

喻金钱 喻 斌 编著

清华大学出版社

北 京

内 容 简 介

本书从实际应用需求和开发过程中所遇到的问题出发,介绍了STM32F系列ARM芯片内外设和各个功能模块的应用。

本书没有涉及有关芯片的存储结构系统构架、指令集等理论性的知识,而是从最基本的应用要求出发,结合大量实例,依托库函数,详细讲解I/O接口、异步串口、系统时基定时器、SPI接口、RTC、看门狗、定时器、I2C接口、CAN接口和模数转化器等外设接口的使用方法。本书注重实际操作和开发中的细节,对在开发过程中容易出错的情况作出提醒,并与读者分享作者在实际开发中的一些经验和感想,为有单片机和C语言基础的读者打开了通向嵌入式开发的大门。

本书可作为单片机爱好者的学习用书,也可作为嵌入式应用工程技术人员的学习和培训用书,同时可作为大学生学习单片机的教材。

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

STM32F 系列 ARM Cortex-M3 核微控制器开发与应用/喻金钱,喻斌编著. —北京:清华大学出版社,2011.4

ISBN 978-7-302-24442-4

I. ①S… II. ①喻… ②喻… III. ①微控制器 IV. ①TP332.3

中国版本图书馆 CIP 数据核字(2010)第 249554 号

责任编辑:钟志芳 朱 俊

封面设计:张 岩

版式设计:文森时代

责任校对:柴 燕

责任印制:李红英

出版发行:清华大学出版社

地 址:北京清华大学学研大厦 A 座

<http://www.tup.com.cn>

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

印 刷 者:清华大学印刷厂

装 订 者:三河市金元印装有限公司

经 销:全国新华书店

开 本:185×260 印 张:32 字 数:739 千字

附光盘 1 张

版 次:2011 年 4 月第 1 版

印 次:2011 年 4 月第 1 次印刷

印 数:1~4000

定 价:58.00 元

产品编号:039295-01

前 言

单片机市场的规模越来越大，到 2010 年出货量在 20G 片左右。世界各大半导体公司纷纷亮出自己的得意之作，提供各具特色的器件和架构。意法半导体集团（ST）是世界第五大半导体公司，当前推出一个新的 32 位微控制器系列产品——STM32 系列。该系列产品所用微处理器是 ARM 公司为高性能、低成本、低功耗的嵌入式应用专门设计的 ARM Cortex-M3 内核。

由于各行各业对于单片机的要求在不断提高，处理器必须在不增加功耗的条件下，处理更多的任务。处理器间的互联也变得复杂，串口、USB、SPI、I2C、CAN 等一系列的硬件接口一个都不能少。在软件上，应用程序的功能也在不断创新，要求有更高的运算速度，更强的实时能力，更炫的图形界面……STM32 系列产品得益于 Cortex-M3 在架构上进行的多项改进，包括提升性能的同时又提高了代码密度的 Thumb-2 指令集和大幅度提高中断响应的紧耦合嵌套向量中断控制器，所有这些功能都同时具有业界最优的功耗水平。这些性能的不断提提高满足了业界对单片机的需求。许多曾经只能求助于高级 32 位处理器或 DSP 的软件设计，都能在 STM32 上运行得很顺畅。

本书根据笔者多年使用 STM32 的实际经验与体会，结合库函数，以实现其功能为主线，强调实例应用和开发调试过程的特点进行编写。本书并没有介绍每个寄存器的详细功能和具体使用（这在 STM32F 的技术手册中有详细的描述），而是通过对具体实例的讲解和剖析，结合厂家提供的固件库，简单轻松地实现所要达到的功能，让每一个工程师都能使用 STM32F 系列芯片，并能把这系列芯片的功能发挥到极致。只要会 C 语言，通过这本书的学习，读者不必去弄懂底层硬件结构，就能使用 ARM，就可以很好地使用 STM32F 系列进行开发、调试工作。

本书主要内容

本书在编写过程中，强调实用、易用和有用。全书共分为三大部分，第 1 部分介绍开发平台和实验平台，这是后面学习的基础；第 2 部分介绍 STM32F 芯片的各个功能模块的特点、库函数，然后用具体的实例详细介绍如何使用这些库函数实现功能模块的不同应用；第 3 部分是基于 STM32F 常用系统设计的实例应用。

第 1 部分是本书的基础，需要熟练掌握，只有这样才可以有效提高开发效率，减少错误的发生。

第 2 部分是本书的重点，共有 13 章，分别介绍各个功能模块，这 13 章采用了相同的讲解模式，首先介绍该模块的功能，随即介绍能实现这些功能的库函数，最后用多个实例详细讲解如何使用这些库函数实现模块功能。该部分是本书最有特色的部分，也是读者最感兴趣的部分，其中讲解了 LED 灯驱动实例、流水灯实例、按键实例，串口发送数据实例，

中断接收数据方式实例、串口队列实例、嘀嗒实例、实用按键实例、外部中断实例、中断嵌套实例、万年历实例、后备域数据读写实例、I2C 与 24C02 通信实例、单通道 ADC 采样实例、温度采样实例、内部 Flash 读写实例、定时器简单定时实例、PWM 输出实例、独立看门狗实例、窗口看门狗实例和 CAN 接口实例等二十多个实例。

第 3 部分是本书的提高部分，讲解了常用系统设计实例。这些实例在实际应用中经常遇到，本书详细讲解了其思路和逻辑方法，这也是本书的特色部分，包括 GPS 数据解析、NOR Flash 数据储存方案和 2.4G 无线数据传输等高级应用。

通过使用本固件函数库，无须深入掌握细节，用户也可以轻松应用每一个外设。因此，使用本固件函数库可以大大减少用户的程序编写时间，进而降低开发成本。

本书以具体的功能实例为基础，引导读者分析实例并实现这些功能。在开发调试中，一步一步地解决问题、实现功能，并把一个复杂的问题划分成一个一个好解决的小问题，一个一个地解决，最后实现整个功能。这也是本书作者着力介绍的一种解决问题的方法。

读者定位

本书要求读者有基本的 C 语言基础，不需要有硬件方面的知识。通过本书的学习，可以让一个初学者很快进入到嵌入式开发的大门。

同时，本书可作为嵌入式应用工程技术人员的学习和培训用书，也可作为企业内部培训教材，当然如果作为大学单片机教材，也会起到事半功倍的教学效果。

如果您是一个单片机的入门者，那么这本书可以说是为您量身打造的。您只需要按照本书的编排顺序，一章一章地往后学，一个实例一个实例地理解和编写即可。本书后面的内容是以前面的内容为基础的，像堆积木一样，一点一点地把功能进行堆积。等您学完这本书，您就可以成为一个嵌入式的熟手了。

如果您是单片机方面的高手，想通过本书学习使用 STM32 系列的芯片，您只需要熟悉开发平台和实验平台，然后了解每个外设功能模块是如何初始化的即可。本书第 3 部分内容您也可以看看，或许能给您带来意想不到的收获。

本书配套资料

本书配套光盘中有书中各个实例的源代码，这些源代码都在实验板上验证通过。希望广大读者不要只是把源代码一烧了之，而是应该尝试自己编写这些软件，因为只有经过不断的实践，才能获得真知。

为了帮助广大读者更快地进入到嵌入式开发中，作者将提供与本书配套的实验板。当然读者也可以根据本书提供的原理图来自行搭建，或使用其他实验板，依据其硬件更改相应的实例代码。作者在此提供一个经过验证的可靠的硬件平台，是为了让读者能在开始时绕过硬件屏障，全心学习 STM32F 系列芯片功能外设。当读者掌握了这些技能后，依据本书中作者提供的版图尺寸和需注意的细节，完全可以设计出实现自己所需功能的、性能优异的电路板。

整个开发系统的搭建只需要一台 PC、一个实验板、一条串口延长线即可，不需要昂贵的仿真器或下载器。

在本书的编写过程中，得到了家人的理解和大力支持，并得到了清华大学出版社钟志芳老师的大力支持，师荣老师对全书进行了辛苦的校对，在此一并表示感谢。

由于本书涉及的知识领域日新月异，加上作者水平有限及时间仓促，难免有差错和不足之处，希望广大读者批评指正。有任何建议和意见，可以和我联系，我的邮箱为 tonda@126.com。

喻金钱

2011 年 1 月

目 录

第 1 部分 基础篇

第 1 章 开发板硬件结构	2	1.3.2 硬件资源使用	19
1.1 电路原理图	2	第 2 章 编译开发环境的建立	21
1.2 原理图说明	5	2.1 下载和安装 EWARM	21
1.2.1 电源电路	5	2.2 IDE 界面简介	24
1.2.2 系统复位电路	5	2.3 生成一个新项目	25
1.2.3 时钟电路	6	2.3.1 建立项目文件目录, 复制公共文件	25
1.2.4 JTAG 接口电路	6	2.3.2 生成新的工作区	25
1.2.5 串口电路	6	2.3.3 生成新项目	26
1.2.6 键盘电路	7	2.3.4 给项目添加文件	27
1.2.7 LED 灯电路	8	2.4 修改应用文件	28
1.2.8 I2C 接口电路	8	2.5 配置项目选项	29
1.2.9 ADC 电路	9	2.5.1 通用选项设置	30
1.2.10 USB 电路	10	2.5.2 C/C++ 编译器选项设置	30
1.2.11 CAN 电路	10	2.5.3 Assembler 选项设置	31
1.2.12 语音采集和播放电路	10	2.5.4 Output Converter 选项设置	31
1.2.13 SPI 接口电路	11	2.5.5 Linker 选项设置	32
1.2.14 电动机驱动板接口电路	13	2.6 Flash Loader Demo 下载器介绍	33
1.3 开发板元器件布局图	14	2.7 力源 STM32F 的 ISP 下载器	36
1.3.1 跳线器说明	14	2.8 串口调试助手介绍	37

第 2 部分 应用篇

第 3 章 通用和复用功能 I/O 口	40	3.1.10 模拟输入配置	43
3.1 概述	40	3.2 库函数	43
3.1.1 通用 I/O	40	3.2.1 函数 GPIO_Init	43
3.1.2 单独的位设置或位清除	41	3.2.2 函数 GPIO_SetBits	45
3.1.3 外部中断/唤醒线	41	3.2.3 函数 GPIO_ResetBits	45
3.1.4 复用功能	41	3.2.4 函数 GPIO_WriteBit	46
3.1.5 软件重新映射 I/O 复用功能	41	3.2.5 函数 GPIO_Write	46
3.1.6 GPIO 锁定机制	41	3.2.6 函数 GPIO_ReadOutputDataBit	47
3.1.7 输入配置	42	3.2.7 函数 GPIO_ReadOutputData	47
3.1.8 输出配置	42	3.2.8 函数 GPIO_ReadInputDataBit	48
3.1.9 复用功能配置	42	3.2.9 函数 GPIO_ReadInputData	48

3.3 I/O 端口的外设映射.....	49	4.1.1 USART 主要特性.....	79
3.3.1 将 OSC32_IN/OSC32_OUT 作为 PC14/PC15 端口.....	49	4.1.2 USART 功能概述.....	80
3.3.2 将 OSC_IN/OSC_OUT 引脚作为 PD0/PD1 端口.....	49	4.1.3 发送器.....	80
3.3.3 CAN 复用功能重映射.....	50	4.1.4 接收器.....	82
3.3.4 JTAG/SWD 复用功能重映射.....	50	4.1.5 分数比特率的产生.....	83
3.3.5 ADC 复用功能重映射.....	51	4.1.6 多处理器通信.....	84
3.3.6 定时器复用功能重映射.....	51	4.1.7 LIN 模式.....	84
3.3.7 USART 复用功能重映射.....	53	4.1.8 USART 同步模式.....	85
3.3.8 I2C1 复用功能重映射.....	53	4.1.9 单线半双工通信.....	87
3.3.9 SPI1 复用功能重映射.....	53	4.1.10 智能卡.....	87
3.4 位运算.....	54	4.1.11 IrDA SIR ENDEC 功能块.....	89
3.4.1 移位运算.....	54	4.1.12 USART 中断请求.....	90
3.4.2 按位与运算.....	57	4.1.13 USART 模式配置.....	91
3.4.3 按位或运算.....	57	4.2 USART 串口库函数介绍.....	91
3.4.4 取反运算.....	58	4.2.1 函数 USART_Init.....	91
3.4.5 异或运算.....	58	4.2.2 函数 USART_Cmd.....	94
3.5 I/O 口输出实例 1——控制 LED 灯.....	59	4.2.3 函数 USART_ITConfig.....	94
3.5.1 实例要求.....	59	4.2.4 函数 USART_SendData.....	95
3.5.2 硬件基础.....	59	4.2.5 函数 USART_ReceiveData.....	95
3.5.3 软件结构.....	59	4.2.6 函数 USART_GetFlagStatus.....	96
3.5.4 实例代码.....	60	4.2.7 函数 USART_ClearFlag.....	97
3.5.5 编译下载和调试.....	68	4.2.8 函数 USART_GetITStatus.....	97
3.6 I/O 口输出实例 2——流水灯.....	68	4.3 不同型号芯片的 USART 串口 复用重映射.....	98
3.6.1 实例要求.....	68	4.3.1 引脚为 36 的系列芯片和引脚为 48 和 64, 容量为 32KB 的芯片.....	98
3.6.2 硬件基础.....	68	4.3.2 引脚为 48 的中容量芯片.....	99
3.6.3 软件结构.....	69	4.3.3 引脚为 64 的中容量芯片.....	99
3.6.4 实例代码.....	70	4.3.4 引脚为 100 的中容量芯片.....	100
3.6.5 编译下载和调试.....	71	4.3.5 引脚为 64 的大容量芯片.....	101
3.7 I/O 口输入实例——按键输入 1.....	72	4.3.6 引脚为 144 的系列芯片和 引脚为 100 的高容量芯片.....	102
3.7.1 实例要求.....	72	4.4 USART 通信实例 1——串口 发送数据.....	102
3.7.2 硬件基础.....	72	4.4.1 实例要求.....	102
3.7.3 软件结构.....	72	4.4.2 硬件基础.....	103
3.7.4 实例代码.....	73	4.4.3 软件结构.....	103
3.7.5 编译下载和调试.....	75	4.4.4 实例代码.....	106
3.8 I/O 口输入实例——按键输入 2.....	76	4.4.5 编译下载和调试.....	108
第 4 章 USART 串口的一般应用.....	79	4.5 USART 通信实例 2——中断 接收数据方式.....	109
4.1 USART 介绍.....	79		

4.5.1 实例要求	109	6.3.2 函数 NVIC_PriorityGroupConfig	134
4.5.2 硬件基础	109	6.3.3 函数 NVIC_Init	135
4.5.3 软件结构	109	6.3.4 函数 NVIC_SetVectorTable	138
4.5.4 实例代码	110	6.4 外部中断控制器库函数介绍	138
4.5.5 编译下载和调试	112	6.4.1 函数 EXTI_DeInit	138
4.6 使用队列收发数据实例	112	6.4.2 函数 EXTI_Init	139
第 5 章 系统时基定时器	115	6.4.3 函数 EXTI_GenerateSWInterrupt	141
5.1 概述	115	6.4.4 函数 EXTI_GetFlagStatus	141
5.2 库函数介绍	115	6.4.5 函数 EXTI_ClearFlag	141
5.2.1 函数 SysTick_CLKSourceConfig	115	6.4.6 函数 EXTI_GetITStatus	142
5.2.2 函数 SysTick_SetReload	116	6.5 外部中断实例	142
5.2.3 函数 SysTick_CounterCmd	116	6.5.1 实例目的	142
5.2.4 函数 SysTick_ITConfig	117	6.5.2 实例要求	142
5.2.5 函数 SysTick_GetCounter	117	6.5.3 硬件基础	143
5.3 系统时基定时器实例 1——嘀嗒		6.5.4 软件结构	143
实例	118	6.5.5 实例代码	144
5.3.1 实例要求	118	6.5.6 编译下载和调试	146
5.3.2 软件结构	118	6.6 中断嵌套实例	146
5.3.3 实例代码	119	6.6.1 实例目的	146
5.3.4 编译下载和调试	120	6.6.2 实例要求	146
5.4 系统时基定时器实例 2——有实际		6.6.3 硬件基础	146
应用意义的键盘实例	120	6.6.4 软件结构	146
5.4.1 实例要求	120	6.6.5 实例代码	148
5.4.2 软件结构	120	6.6.6 编译下载和调试	149
5.4.3 实例代码	121	第 7 章 复位和系统时钟	151
5.4.4 编译下载和调试	124	7.1 复位	151
第 6 章 外部中断和中断控制器	127	7.1.1 系统复位	151
6.1 嵌套向量中断控制器	127	7.1.2 电源复位	152
6.1.1 概述	127	7.1.3 备份区域复位	152
6.1.2 中断和异常向量	127	7.2 时钟	152
6.1.3 中断优先级介绍	129	7.2.1 HSE 时钟	153
6.2 外部中断/事件控制器	131	7.2.2 HSI 时钟	154
6.2.1 EXTI 控制器的主要特征	131	7.2.3 PLL	154
6.2.2 唤醒事件管理	131	7.2.4 LSE 时钟	155
6.2.3 功能说明	132	7.2.5 LSI 时钟	155
6.2.4 外部中断/事件线路映射	132	7.2.6 系统时钟选择	156
6.3 NVIC 库函数介绍	133	7.2.7 时钟安全系统	156
6.3.1 函数 NVIC_DeInit	133	7.2.8 RTC 时钟	156
		7.2.9 看门狗时钟	157

7.2.10 时钟输出	157	8.3 RTC 实时时钟库函数介绍	177
7.3 外设时钟	157	8.3.1 函数 RTC_ITConfig	177
7.4 RCC 库函数	158	8.3.2 函数 RTC_EnterConfigMode	178
7.4.1 函数 RCC_DeInit	158	8.3.3 函数 RTC_ExitConfigMode	178
7.4.2 函数 RCC_HSEConfig	158	8.3.4 函数 RTC_GetCounter	179
7.4.3 函数 RCC_WaitForHSEStartUp	159	8.3.5 函数 RTC_SetCounter	179
7.4.4 函数 RCC_PLLConfig	159	8.3.6 函数 RTC_SetPrescaler	180
7.4.5 函数 RCC_PLLCmd	161	8.3.7 函数 RTC_SetAlarm	180
7.4.6 函数 RCC_SYSCLKConfig	161	8.3.8 函数 RTC_WaitForLastTask	181
7.4.7 函数 RCC_GetSYSCLKSource	162	8.3.9 函数 RTC_WaitForSynchro	181
7.4.8 函数 RCC_HCLKConfig	162	8.3.10 函数 RTC_GetFlagStatus	182
7.4.9 函数 RCC_PCLK1Config	163	8.3.11 函数 RTC_ClearFlag	182
7.4.10 函数 RCC_PCLK2Config	164	8.3.12 函数 RTC_GetITStatus	183
7.4.11 函数 RCC_USBCLKConfig	164	8.4 BKP 后备域库函数介绍	183
7.4.12 函数 RCC_ADCCLKConfig	165	8.4.1 函数 BKP_DeInit	183
7.4.13 函数 RCC_LSEConfig	165	8.4.2 函数 BKP_TamperPinLevelConfig	184
7.4.14 函数 RCC_RTCCLKConfig	166	8.4.3 函数 BKP_TamperPinCmd	184
7.4.15 函数 RCC_RTCCLKCmd	167	8.4.4 函数 BKP_WriteBackupRegister	185
7.4.16 函数 RCC_AHBPeriph		8.4.5 函数 BKP_ReadBackupRegister	186
ClockCmd	167	8.4.6 函数 BKP_ClearITPendingBit	186
7.4.17 函数 RCC_APB1Periph		8.5 实时时钟实例——万年历	186
ClockCmd	168	8.5.1 实例目的	186
7.4.18 函数 RCC_APB2Periph		8.5.2 实例要求	187
ClockCmd	169	8.5.3 硬件基础	187
7.4.19 函数 RCC_GetFlagStatus	170	8.5.4 软件结构	187
7.5 系统时钟的建立	171	8.5.5 实例代码	188
7.5.1 如何建立时钟	171	8.5.6 编译下载和调试	191
7.5.2 实例代码	171	8.6 在后备域中保存数据	192
第 8 章 实时时钟和备份寄存器	173	8.6.1 实例要求	192
8.1 RTC 简介	173	8.6.2 硬件基础	192
8.1.1 主要特性	173	8.6.3 软件结构	192
8.1.2 复位过程	173	8.6.4 实例代码	193
8.1.3 概述	173	8.6.5 编译下载和调试	194
8.1.4 基本操作	174	第 9 章 通用 SPI 的一般应用	195
8.1.5 后备域和 RTC 供电	175	9.1 SPI 简介	195
8.1.6 低功耗模式下的自动唤醒	176	9.1.1 SPI 特征	195
8.2 BKP 简介	176	9.1.2 SPI 引脚描述	196
8.2.1 BKP 特性	176	9.1.3 数据传输模式	197
8.2.2 侵入检测	177	9.1.4 SPI 从模式	198

9.1.5 SPI 主模式	198	10.3.7 函数 I2C_OwnAddress2Config	228
9.1.6 状态标志	199	10.3.8 函数 I2C_DualAddressCmd	229
9.1.7 利用 DMA 的 SPI 通信	200	10.3.9 函数 I2C_GeneralCallCmd	229
9.1.8 SPI 中断	200	10.3.10 函数 I2C_ITConfig	230
9.2 SPI 库函数介绍	200	10.3.11 函数 I2C_SendData	230
9.2.1 函数 SPI_DeInit	200	10.3.12 函数 I2C_ReceiveData	231
9.2.2 函数 SPI_Init	201	10.3.13 函数 I2C_Send7bitAddress	231
9.2.3 函数 SPI_Cmd	203	10.3.14 函数 I2C_ReadRegister	232
9.2.4 函数 SPI_ITConfig	204	10.3.15 函数 I2C_SoftwareResetCmd	233
9.2.5 函数 SPI_DMAMCmd	205	10.3.16 函数 I2C_GetLastEvent	233
9.2.6 函数 SPI_SendData	205	10.3.17 函数 I2C_CheckEvent	234
9.2.7 函数 SPI_ReceiveData	206	10.4 I2C 读写 24C02	235
9.2.8 函数 SPI_GetFlagStatus	206	10.4.1 实例要求	235
9.2.9 函数 SPI_ClearFlag	207	10.4.2 硬件基础	235
9.2.10 函数 SPI_GetITStatus	207	10.4.3 24C02 器件介绍	235
9.2.11 函数 SPI_ClearITPendingBit	208	10.4.4 软件结构	237
9.3 SPI 使用	208	10.4.5 实例代码	238
9.3.1 SPI 初始化	208	10.4.6 编译下载和调试	241
9.3.2 SPI 主机发送/接收数据	210	第 11 章 ADC 的一般应用	243
9.3.3 完整的初始化和收发数据代码	211	11.1 ADC 介绍	243
第 10 章 I2C 接口的一般应用	213	11.1.1 ADC 主要特征	243
10.1 I2C 简介	213	11.1.2 ADC 功能描述	243
10.2 I2C 功能描述	213	11.1.3 校准	246
10.2.1 模式选择	214	11.1.4 数据对齐	247
10.2.2 通信流	214	11.1.5 可编程的通道采样时间	247
10.2.3 I2C 从模式	214	11.1.6 外部触发转换	247
10.2.4 I2C 主模式	216	11.1.7 DMA 请求	249
10.2.5 错误条件	219	11.1.8 双 ADC 模式	249
10.2.6 SDA/SCL 线控制	219	11.1.9 温度传感器	250
10.2.7 SMBus	220	11.1.10 ADC 中断	251
10.2.8 DMA 请求	222	11.2 实现 ADC 最佳精度	251
10.2.9 I2C 中断请求	223	11.2.1 ADC 模块自身相关的误差	251
10.3 I2C 库函数介绍	225	11.2.2 与环境相关的 ADC 误差	254
10.3.1 函数 I2C_DeInit	225	11.2.3 如何减小与外部环境相关的 ADC 误差	257
10.3.2 函数 I2C_Init	225	11.3 ADC 库函数介绍	265
10.3.3 函数 I2C_Cmd	227	11.3.1 函数 ADC_DeInit	265
10.3.4 函数 I2C_GenerateSTART	227	11.3.2 函数 ADC_Init	265
10.3.5 函数 I2C_GenerateSTOP	228	11.3.3 函数 ADC_Cmd	267
10.3.6 函数 I2C_AcknowledgeConfig	228		

11.3.4	函数 ADC_DMACmd.....	267
11.3.5	函数 ADC_ITConfig.....	268
11.3.6	函数 ADC_ResetCalibration.....	268
11.3.7	函数 ADC_GetReset CalibrationStatus	269
11.3.8	函数 ADC_StartCalibration	269
11.3.9	函数 ADC_GetCalibrationStatus ...	270
11.3.10	函数 ADC_SoftwareStart ConvCmd.....	270
11.3.11	函数 ADC_DiscModeChannel CountConfig	270
11.3.12	函数 ADC_DiscModeCmd	271
11.3.13	函数 ADC-RegularChannel Config	271
11.3.14	函数 ADC_ExternalTrig ConvConfig	273
11.3.15	函数 ADC_GetConversionValue.....	273
11.3.16	函数 ADC_GetDualMode ConversionValue	273
11.3.17	函数 ADC_AutoInjected ConvCmd.....	274
11.3.18	函数 ADC_InjectedDisc ModeCmd.....	274
11.3.19	函数 ADC_ExternalTrigInjected ConvConfig	275
11.3.20	函数 ADC_ExternalTrigInjected ConvCmd.....	276
11.3.21	函数 ADC_SoftwareStartInjected ConvCmd.....	276
11.3.22	函数 ADC_GetSoftwareStartInjected ConvStatus.....	276
11.3.23	函数 ADC_InjectedChannle Config	277
11.3.24	函数 ADC_InjectedSequencer LengthConfig.....	277
11.3.25	函数 ADC_SetInjectedOffset.....	278
11.3.26	函数 ADC_GetInjected ConversionValue	279
11.3.27	函数 ADC_TampSensor VrefintCmd.....	279
11.3.28	函数 ADC_GetFlagStatus	279

11.3.29	函数 ADC_ClearFlag.....	280
11.4	ADC 数据采集实例 1——单通道 数据采集.....	281
11.4.1	实例要求	281
11.4.2	硬件基础	281
11.4.3	软件结构	281
11.4.4	实例代码	283
11.4.5	编译下载和调试	287
11.5	ADC 数据采集实例 2——芯片 温度采集.....	288
11.5.1	实例要求	288
11.5.2	硬件基础	288
11.5.3	软件结构	288
11.5.4	实例代码	288
11.5.5	编译下载和调试	292

第 12 章 嵌入式闪存的基本操作..... 293

12.1	嵌入式闪存介绍	293
12.1.1	特性	293
12.1.2	闪存模块组织	293
12.1.3	读操作	294
12.1.4	闪存编程和擦除控制器	295
12.2	FLASH 固件库函数介绍	300
12.2.1	函数 FLASH_SetLatency	300
12.2.2	函数 FLASH_HalfCycle AccessCmd.....	301
12.2.3	函数 FLASH_PrefetchBufferCmd ...	302
12.2.4	函数 FLASH_Unlock.....	302
12.2.5	函数 FLASH_Lock	303
12.2.6	函数 FLASH_ErasePage.....	303
12.2.7	函数 FLASH_EraseAllPages	303
12.2.8	函数 FLASH_EraseOptionBytes ...	304
12.2.9	函数 FLASH_ProgramWord.....	304
12.2.10	函数 FLASH_ProgramHalfWord ...	305
12.2.11	函数 FLASH_ProgramOption ByteData	305
12.2.12	函数 FLASH_EnableWrite Protection.....	306
12.2.13	函数 FLASH_ReadOut Protection.....	307

12.2.14 函数 FLASH_UserOption ByteConfig.....	308	13.2.13 函数 TIM_EncoderInterface Config	345
12.2.15 函数 FLASH_GetUser OptionByte.....	309	13.2.14 函数 TIM_ARRPreloadConfig	346
12.2.16 函数 FLASH_GetWriteProtection OptionByte.....	309	13.2.15 函数 TIM_CCPreloadControl	347
12.2.17 函数 FLASH_GetReadOut ProtectionStatus.....	310	13.2.16 函数 TIM_OC1PreloadConfig.....	347
12.2.18 函数 FLASH_GetPrefetch BufferStatus	310	13.2.17 函数 TIM_OC2PreloadConfig.....	348
12.2.19 函数 FLASH_ITConfig.....	311	13.2.18 函数 TIM_OC3PreloadConfig.....	348
12.2.20 函数 FLASH_GetFlagStatus.....	311	13.2.19 函数 TIM_OC4PreloadConfig.....	349
12.2.21 函数 FLASH_ClearFlag.....	312	13.2.20 函数 TIM_SelectOutputTrigger ...	349
12.2.22 函数 FLASH_GetStatus.....	313	13.2.21 函数 TIM_SelectSlaveMode.....	350
12.2.23 函数 FLASH_WaitFor LastOperation	313	13.2.22 函数 TIM_SelectMaster SlaveMode	350
12.3 FLASH 读写实例	314	13.2.23 函数 TIM_SetCounter.....	351
12.3.1 实例要求	314	13.2.24 函数 TIM_SetAutoreload.....	351
12.3.2 硬件基础	314	13.2.25 函数 TIM_GetCounter	352
12.3.3 软件结构	314	13.2.26 函数 TIM_GetPrescaler	352
12.3.4 实例代码	314	13.2.27 函数 TIM_GetFlagStatus	353
12.3.5 编译下载和调试	316	13.2.28 函数 TIM_ClearFlag	354
第 13 章 定时器的一般应用	317	13.2.29 函数 TIM_GetITStatus	354
13.1 定时器功能简介	317	13.2.30 函数 TIM_ClearITPendingBit	354
13.1.1 TIMx 主要功能.....	317	13.3 TIME 应用实例 1——简单定时器 应用	355
13.1.2 TIMx 功能描述.....	318	13.3.1 实例要求	355
13.2 定时器固件库函数介绍.....	333	13.3.2 硬件基础	355
13.2.1 函数 TIM_DeInit.....	333	13.3.3 软件结构	355
13.2.2 函数 TIM_TimeBaseInit	334	13.3.4 实例代码	360
13.2.3 函数 TIM_OC1Init.....	335	13.3.5 编译下载和调试	363
13.2.4 函数 TIM_OC2Init.....	337	13.4 TIME 应用实例 2——使用一个 定时器产生 4 路不同占空比的 PWM	363
13.2.5 函数 TIM_OC3Init.....	338	13.4.1 实例目的	363
13.2.6 函数 TIM_OC4Init.....	339	13.4.2 实例要求	363
13.2.7 函数 TIM_ICInit	339	13.4.3 硬件基础	363
13.2.8 函数 TIM_BDTRConfig	341	13.4.4 软件结构	363
13.2.9 函数 TIM_Cmd	343	13.4.5 实例代码	366
13.2.10 函数 TIM_CtrlPWMOutputs	343	13.4.6 编译下载和调试	368
13.2.11 函数 TIM_ITConfig.....	344	13.5 TIME 应用实例 3——使用定时器 产生 4 路不同占空比和频率的 PWM.....	368
13.2.12 函数 TIM_SelectInputTrigger.....	345		

13.5.1 实例目的	368
13.5.2 实例要求	368
13.5.3 硬件基础	369
13.5.4 软件结构	369
13.5.5 实例代码	373
13.5.6 编译下载和调试	375
13.6 TIME 应用实例 4——定时器同步	375
13.6.1 实例目的	375
13.6.2 实例要求	375
13.6.3 软件结构	376
13.6.4 实例代码	378
13.6.5 编译下载和调试	381

第 14 章 独立看门狗和窗口看门狗

定时器	382
14.1 独立看门狗一般特性介绍	382
14.1.1 IWDG 主要特性	382
14.1.2 IWDG 功能描述	383
14.1.3 硬件看门狗	383
14.1.4 寄存器访问保护	383
14.2 窗口看门狗一般特性介绍	383
14.2.1 WWDG 主要特性	384
14.2.2 WWDG 功能描述	384
14.3 独立看门狗库函数介绍	385
14.3.1 函数 IWDG_WriteAccessCmd	386
14.3.2 函数 IWDG_SetPrescaler	386
14.3.3 函数 IWDG_SetReload	387
14.3.4 函数 IWDG_ReloadCounter	387
14.3.5 函数 IWDG_Enable	388
14.4 窗口看门狗库函数介绍	388
14.4.1 函数 WWDG_DeInit	388
14.4.2 函数 WWDG_SetPrescaler	388
14.4.3 函数 WWDG_SetWindow Value	389
14.4.4 函数 WWDG_EnableIT	390
14.4.5 函数 WWDG_SetCounter	390
14.4.6 函数 WWDG_Enable	390
14.4.7 函数 WWDG_GetFlagStatus	391
14.4.8 函数 WWDG_ClearFlag	391
14.5 独立看门狗实例	392

14.5.1 实例目的	392
14.5.2 实例要求	392
14.5.3 硬件基础	392
14.5.4 软件结构	392
14.5.5 实例代码	393
14.5.6 编译下载和调试	394
14.6 窗口看门狗实例	395
14.6.1 实例目的	395
14.6.2 实例要求	395
14.6.3 硬件基础	395
14.6.4 软件结构	395
14.6.5 实例代码	396
14.6.6 编译下载和调试	398

第 15 章 控制器局域网 CAN 的

一般应用	400
15.1 CAN 介绍	400
15.1.1 bxCAN 主要特点	400
15.1.2 总体描述	401
15.1.3 bxCAN 工作模式	401
15.1.4 bxCAN 发送处理	403
15.1.5 时间触发通信模式	404
15.1.6 接收管理	404
15.1.7 标识符过滤	405
15.1.8 报文存储	407
15.1.9 出错管理	407
15.1.10 位时间特性	408
15.1.11 bxCAN 中断	408
15.2 bxCAN 库函数介绍	409
15.2.1 函数 CAN_DeInit	409
15.2.2 函数 CAN_Init	409
15.2.3 函数 CAN_FilterInit	411
15.2.4 函数 CAN_StructInit	413
15.2.5 函数 CAN_ITConfig	414
15.2.6 函数 CAN_Transmit	415
15.2.7 函数 CAN_TransmitStatus	416
15.2.8 函数 CAN_CancelTransmit	416
15.2.9 函数 CAN_FIFORelease	417
15.2.10 函数 CAN_MessagePending	417
15.2.11 函数 CAN_Receive	418
15.2.12 函数 CAN_Sleep	419

15.2.13 函数 CAN_WakeUp.....	419	15.3.1 实例目的	422
15.2.14 函数 CAN_GetFlagStatus	420	15.3.2 实例要求	422
15.2.15 函数 CAN_ClearFlag	420	15.3.3 硬件基础	422
15.2.16 函数 CAN_GetITStatus	421	15.3.4 软件结构	423
15.2.17 函数 CAN_ClearITPendingBit	421	15.3.5 实例代码	424
15.3 CAN 收发数据实例	422	15.3.6 编译下载和调试	427

第 3 部分 提高篇

第 16 章 GPS 数据解析	430	18.1.1 低功耗短距离无线通信概述	464
16.1 GPS 数据协议解析	430	18.1.2 短距离低功耗无线应用	464
16.1.1 GPS 概述	430	18.2 2.4G 无线模块介绍	465
16.1.2 GPS 介绍	430	18.2.1 芯片的架构	465
16.1.3 GPS 数据结构介绍	431	18.2.2 芯片主要特点	465
16.2 GPS 数据解析实例	434	18.2.3 功能概述	466
16.2.1 实例要求	434	18.2.4 寄存器介绍	470
16.2.2 硬件基础	434	18.3 无线模块相关 API 函数集	471
16.2.3 软件结构	434	18.3.1 无线芯片的 SPI 接口	471
16.2.4 实例代码	435	18.3.2 复位	473
第 17 章 串行 Flash 数据储存方案	450	18.3.3 无线模块功能 API 软件函数集	474
17.1 串行 Flash 概述	450	18.4 数据发送/接收的时序	477
17.1.1 SST25VF016B 概述	450	18.4.1 数据发送/接收时序	477
17.1.2 SST25VF016B 引脚说明	451	18.4.2 无线发送/接收数据 API 函数	478
17.1.3 SST25VF016B 接口电路	451	18.5 读无线芯片寄存器实例	479
17.2 API 软件包	452	18.5.1 实例目的	479
17.2.1 软件包结构	452	18.5.2 实例要求	479
17.2.2 SPI 初始化	452	18.5.3 硬件基础	479
17.2.3 读数据 API	455	18.5.4 软件结构	480
17.2.4 写数据 API	456	18.5.5 实例代码	480
17.2.5 Flash 擦除 API	457	18.6 双向无线数据收发实例	484
17.2.6 读 ID	459	18.6.1 实例目的	484
17.3 Flash 数据读写实例	459	18.6.2 实例要求	484
17.3.1 实例目的	460	18.6.3 硬件基础	485
17.3.2 实例要求	460	18.6.4 实例代码	485
17.3.3 硬件基础	460	18.6.5 编译下载和调试	489
17.3.4 实例代码	460	附录 A IAR 工程转 MDK 工程	490
17.3.5 编译下载和调试	463	附录 B ARM 处理器: 选择 ARM7 还是 Cortex-M3	492
第 18 章 2.4G 低功耗短距离无线模块应用	464		
18.1 低功耗短距离无线通信	464		

第 1 部分

基 础 篇

第 1 章 开发板硬件结构

第 2 章 编译开发环境的建立

第 1 章 开发板硬件结构

OpenM3V 开发板是作者专门为本书设计的硬件原型，采用了 ST 公司基于 M3 核的 STM32F103VB，可通过 ISP 下载，也可以通过 JTAG 方式调试和下载。

开发板上提供了众多的功能部件，都是工程师在实际应用中常用并必须要使用的模块，充分使用这些模块能尽可能地发挥 STM32 系列的性能。这些功能模块包括键盘和 LED 灯功能部件；I2C 方式接口的 EEPROM 储存器电路；两个 RS-232 串口电路；简单 AD 采集电路，语音 AD 采集电路；CAN 接口电路；USB 接口电路；JTAG 接口电路；后备供电电路；SPI 方式接口的 Flash 储存器接口电路模块，SPI 方式接口的 SD 卡电路，SPI 方式接口的 128×64 点阵液晶接口电路，SPI 方式接口 2.4G 无线通信模块接口电路，SPI 方式接口的 779~928MHz 频段无线模块接口电路；PWM 方式调光电路，PWM 方式语音输出电路及连接直流无刷电动机驱动板的接口电路等，同时结合灵活的跳线，所有的 I/O 口都可以单独引出，可以极大地方便读者进行嵌入式开发。

1.1 电路原理图

OpenM3V 开发板硬件原理图如图 1-1-1~图 1-1-5 所示。

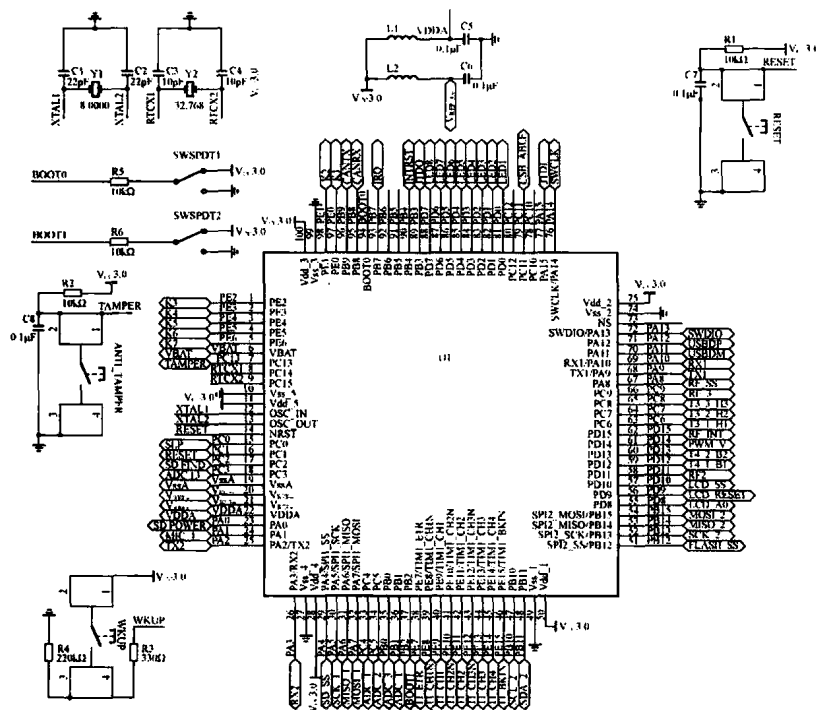


图 1-1-1 芯片最小系统部分