普通高等教育“十二五”电子信息类规划教材

亮点嵌入式系列

基于STM32的嵌入式系统原理与设计

卢有亮 编著

免费电子课件

机械工业出版社
CHINA MACHINE PRESS

普通高等教育“十二五”电子信息类规划教材

亮点嵌入式系列

基于 STM32 的嵌入式系统原理与设计

卢有亮　编著

机械工业出版社

本书内容包括了基于 STM32 的嵌入式系统设计的原理、硬件设计、软件开发及工程实践。在原理部分针对最关键的部分给出了较为详细的解析。在硬件设计部分设计分析了作者开发的 STM32 开发板的详细电路。在软件编程部分不仅引领读者使用库函数编程实现对外设的编程，部分环节还使用寄存器方式实现，另外给出了双缓冲乒乓操作等高级编程方法，及利用 uIP、FatFs 等编程实现网络通信和文件系统。最后在工程实践部分给出了工程实例。本书的硬件和代码由作者设计及编程。

本书适合于计算机、电子、通信、机电、自动化及相关专业的本、专科学生及研究生，也适合于广大嵌入式系统开发工程技术人员、电子技术研究人员。

本书交流论坛：http：//www. eeboard. com/bp

责任编辑邮箱：jinacmp@ 163. com

图书在版编目（CIP）数据

基于 STM32 的嵌入式系统原理与设计/卢有亮编著. —北京：机械工业出版社，2013. 11

普通高等教育“十二五”电子信息类规划教材

ISBN 978 - 7 - 111 - 44416 - 9

Ⅰ. ①基… Ⅱ. ①卢… Ⅲ. ①微型计算机 - 系统设计 - 高等学校 - 教材 Ⅳ. ①TP360. 21

中国版本图书馆 CIP 数据核字（2013）第 246235 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

策划编辑：吉 玲 责任编辑：吉 玲 崔利平 刘丽敏

版式设计：霍永明 责任校对：程俊巧

封面设计：张 静 责任印制：张 楠

北京京丰印刷厂印刷

2014 年 1 月第 1 版 · 第 1 次印刷

184mm × 260mm · 15. 5 印张 · 379 千字

标准书号：ISBN 978 - 7 - 111 - 44416 - 9

定价：33. 00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务

社服务中心：(010)88361066

销售一部：(010)68326294

销售二部：(010)88379649

读者购书热线：(010)88379203

网络服务

教材网：http：//www. cmpedu. com

机工官网：http：//www. cmpbook. com

机工官博：http：//weibo. com/cmp1952

封面无防伪标均为盗版

前　言

嵌入式系统设计飞速的发展，渗透到社会生活的各个方面。STM32 以其较高的性能和优越的性价比，已经成为单片机市场的主流之一。本书基于 STM32，对嵌入式系统的原理、设计、编程进行讲解，并给出一个工程实例。另外，开发了 STM32 开发板做为实验平台、搭建了一个交流论坛，旨在构造一个给力的学习平台。

在笔者的第一部著作《嵌入式实时操作系统 μC/OS 原理与实践》于 2012 年 2 月出版之后，受到了读者的欢迎，全部代码在 PC 下虚拟运行，代码和 PPT 可以在博客下载。在这种情况下，虽然可以学到 μC/OS 的代码，但是缺少在嵌入式设备上的实践。另外，笔者认为微机原理课程上有些内容过于抽象，因此，笔者开始构思设计硬件平台。STM32 系列基于为要求高性能、低成本、低功耗的嵌入式应用专门设计的 ARM Cortex-M3 内核，是现在最热门的嵌入式系统之一。因此，最后的选型是 STM32。

编写这本书的目的就是要提供一本在嵌入式系统设计方面读者读的懂的，从原理到硬件设计与调试，再到软件开发和工程实践的好书！本书中大量的实例可以作为实验课的内容。

本书第 1 章介绍了 STM32 的基本原理，这个原理不是原理大全，而是对于开发最关键的部分，例如地址映射、时钟树、中断、DMA、FSMC 等部分给出了较为详细的解析，而对于电源管理等部分只作了简单介绍。

第 2 章介绍的是硬件设计。笔者成功地设计了亮点 STM32 开发板，在这个基础上，给出了硬件设计的解析，对于读者设计嵌入式系统硬件具备参考价值，市面上其他的嵌入式系统设计的书籍是很少有这方面内容的，而且软件的编程离不开硬件的逻辑。

第 3 章是 STM32 软件开发。这里的编程是驱动的编程加应用开发。所有的章节都配备实例，这些实例都是被亮点 STM32 开发板用户验证了的。在第 3 章的最开始部分是“我的第一个基于固件库的工程”，是快速学习入门的好帮手，笔者做到了尽量细致。后面章节的双缓冲操作、网络开发组件 uIP、文件系统组件 FatFs 又是具有一定的难度和极大的应用价值。因此，这一部分由浅入深，覆盖了软件开发的大部分环节。

第 4 章是具体的工程项目实例，有需求、设计和实现，希望读者能通过它找到做工程的方法并巩固所学的知识，并提高工程开发能力。

附录 A 是亮点 STM32 开发板资源，这对软件编程的重要性显而易见，对于端口的设置，请查附录 A。

附录 B 是实验教学安排，可作为高校实验教学使用的一个参考。

因篇幅限制，本书并没有包含 μC/OS 和 μC/GUI，这些内容将出现在后续的亮点嵌入式系列书籍中介绍。

感谢机械工业出版社吉玲对我的大力支持，在成都与我对书籍的写作进行了大量的交流，并促成了亮点嵌入式在爱板网论坛安家。另外，特别感谢给予我很多建议、支持和帮助的朋友。

资源：

本书配套的开发板在淘宝（http：//brightpoint. taobao. com）有售。如果采用其他开发板或用户自己的目标板，只需修改头文件中的配置信息即可使用本书的例程。

读者还可以在以下地址进行学习和交流：

博客：http：//blog. sina. com. cn/u/2630123921

交流论坛：http：//www. eeboard. com/bp

编　者

2013 年于成都

目　录

前言

第 1 章　STM32 基本原理 …… 1

1.1　STM32 性能和结构 …… 1

1.1.1　总体性能 …… 1

1.1.2　系统结构分析 …… 2

1.1.3　芯片封装和引脚概述 …… 4

1.2　Cortex-M3 处理器 …… 5

1.2.1　Cortex-M3 的定位和应用 …… 5

1.2.2　Cortex-M3 处理器结构 …… 6

1.2.3　Cortex-M3 寄存器 …… 7

1.2.4　堆栈 …… 10

1.3　STM32 储存地址映射 …… 11

1.4　引脚功能描述 …… 15

1.5　电源连接 …… 16

1.6　复位和时钟控制（RCC） …… 17

1.6.1　复位 …… 17

1.6.2　时钟源 …… 17

1.6.3　时钟管理寄存器 …… 20

1.7　输入/输出端口 …… 20

1.7.1　常规输入/输出 GPIO …… 20

1.7.2　GPIO 复用 …… 21

1.8　模-数转换器和数-模转换器 …… 22

1.8.1　模-数转换器 …… 22

1.8.2　数-模转换器 …… 23

1.9　中断 …… 24

1.9.1　STM32 的中断通道和中断向量处理 …… 25

1.9.2　STM32 的外部中断 …… 28

1.9.3　STM32 的中断优先级分组 …… 28

1.10　DMA …… 30

1.10.1　DMA 解析 …… 30

1.10.2　DMA 通道和请求 …… 31

1.11　定时器 …… 34

1.11.1　系统滴答定时器（SysTick） …… 34

1.11.2　STM32 常规定时器 …… 39

1.12　同步串行口 SPI 和 I^2C …… 40

1.12.1　SPI …… 40

1.12.2　I^2C …… 43

1.13　同步异步收发器 …… 45

1.14　灵活的 FSMC …… 46

1.14.1　FSMC 概述 …… 46

1.14.2　FSMC 控制液晶控制器 …… 48

习题 1 …… 48

第 2 章　硬件设计 …… 50

2.1　STM32 主板设计 …… 50

2.1.1　MCU 及其周围电路设计 …… 50

2.1.2　USB 转串口电路设计 …… 51

2.1.3　TTL 转 RS232 电路设计 …… 52

2.1.4　网络端口电路 …… 53

2.1.5　SPI FLASH 端口电路 …… 54

2.1.6　I^2C 端口电路 …… 54

2.1.7　TF 卡端口电路 …… 55

2.1.8　按键、LED 显示电路和其他端口 …… 55

2.2　液晶屏与触摸屏控制板设计 …… 56

2.2.1　带触摸 TFT 液晶屏 …… 56

2.2.2　TFT LCD 屏的时序 …… 58

2.2.3　触摸屏 …… 58

2.2.4　TFT LCD 的背光 LED …… 59

2.2.5　TFT LCD 控制器 RA8875 …… 60

2.2.6　TFT 液晶控制板具体设计 …… 65

习题 2 …… 71

第 3 章　STM32 软件开发 …… 72

3.1　STM32 软件开发环境 …… 72

3.1.1　MDK Keil 开发环境 …… 73

3.1.2　串口编程软件 ISP …… 74

3.1.3　JLINK …… 76

3.2　使用固件库开发我的第一个工程 …… 81

3.2.1　获得和理解固件库 …… 81

3.2.2　我的第一个工程 …… 83

3.3　操作 GPIO 和管理中断 …… 99

3.3.1　GPIO 寄存器 …… 99

3.3.2　GPIO 库函数 …… 102

3.3.3　嵌套向量中断控制器 NVIC

库函数 …… 107
3.3.4 外部中断/事件管理库函数 …… 108
3.3.5 带按键控制的流水灯实验 …… 109
3.4 串口通信和 DMA 编程 …… 115
3.4.1 串行异步通信 USART 库函数 …… 115
3.4.2 一个串口发送和中断接收例程的实现 …… 116
3.4.3 DMA 库函数 …… 118
3.4.4 使用 DMA 和双缓冲乒乓操作实现串口接收、发送 …… 119
3.5 SPI 与 I^2C 编程 …… 125
3.5.1 SPI 库函数 …… 125
3.5.2 SPI FLASH 原理 …… 126
3.5.3 SPI FLASH 编程实验 …… 130
3.5.4 TF 卡编程 …… 135
3.5.5 I^2C 编程及实例 …… 145
3.6 液晶屏及触摸屏编程 …… 152
3.6.1 FSMC 端口配置和简单图形显示 …… 152
3.6.2 触摸屏编程 …… 161
3.6.3 汉字输出 …… 163
3.6.4 图片显示和操作 …… 173
3.7 定时器编程 …… 185
3.7.1 SysTick 编程实验 …… 185
3.7.2 定时器库函数 …… 187
3.7.3 定时器编程实验 …… 189
3.8 DAC 和 ADC 编程 …… 192
3.8.1 DAC 库函数 …… 192
3.8.2 DAC 编程 …… 193
3.8.3 ADC 库函数 …… 195
3.8.4 DMA 方式 ADC 采集实验 …… 197
3.9 网络编程 …… 201
3.9.1 网络端口芯片 ENC28J60 驱动 …… 201
3.9.2 uIP 编程 …… 207
3.9.3 使用 uIP 实现 Ping 响应 …… 210
3.9.4 AD 采集和网络 UDP 传输 …… 214
3.9.5 TCP 接收和发送实验 …… 219
习题 3 …… 220
第 4 章 工程项目实例 …… 222
4.1 需求分析 …… 222
4.1.1 需求 …… 222
4.1.2 分析 …… 222
4.2 工程设计 …… 223
4.2.1 整体设计 …… 223
4.2.2 从机硬件端口设计 …… 223
4.2.3 从机软件设计 …… 225
4.3 软件开发 …… 225
4.3.1 宏和变量定义 …… 225
4.3.2 主程序编程 …… 226
设计题 …… 228
附录 …… 229
附录 A 亮点 STM32 开发板资源 …… 229
附录 B 实验教学安排 …… 237
参考文献 …… 239

第 1 章　STM32 基本原理

学习 STM32 需要从原理入手，学好 STM32 基本原理便可为全套的硬件、软件、操作系统、工程实践学习打下一个良好的基础。STM32 基本原理是整本书的快速入门知识或基石。但是本部分并不是原理大全，对于一般的情况，在学习了本书后，并不需要学习一本专门的原理方面的书籍。

无论是做顶层的开发还是做驱动开发，都需要掌握基本的原理。如果没有涉足这个领域，那么本章起到一个入门的作用。同时，本章的内容也可以作为工程开发、操作系统移植等方面的参考。

本章的第一部分是 STM32 的性能和结构，然后是对 Cortex-M3 处理器的分析，之后讲述的是 STM32 储存地址映射，引脚功能描述，电源连接，复位和时钟控制（RCC），输入/输出端口，ADC 和 DAC，中断，DMA，定时器，同步串行口 SPI 和 I^2C，以及同步异步收发器，最后介绍的是灵活的 FSMC。通过本章的学习，既可以掌握 STM32 的全貌，也可以学习到关于 Cortex-M3 处理器内核和 STM32 器件的细节信息。

1.1　STM32 性能和结构

STM32 具有比较高的性价比，且具有很高的市场占有率，下面介绍总体性能。

1.1.1　总体性能

笔者对器件的选型为高密度的 STM32F103VET6，目的是能适合一般项目的需要，而且价格可控制在 30 元以下，避免由于 FLASH 和 RAM 太小造成的瓶颈。因为要运行 μC/OS 和 μC/GUI，需要一定的 FLASH 和 SRAM。VET6 中：V 的含义为 100pins，即 100 个引脚；E 表示 512KB 的 FLASH；T 表示 LQFP 封装；6 表示 -40 ~ 85℃的温度范围。

STM32F103VET6 的整体性能见表 1-1。

表 1-1　STM32F103VET6 的整体性能

项目	解　读
内核	ARM 32-bit Cortex-M3 CPU 核
最高频率	72MHz
处理能力	1.25DMIPS/MHz(在 1MHz 的时钟下,每秒可执行 125 万条整数运算指令)
FLASH	512KB FLASH 存储器
SRAM	64KB SRAM
电源和 I/O 输入电压范围	2.0 ~ 3.6V
模-数转换器(ADC)	3 个 12 位 ADC,16 通道
数-模转换器(DAC)	2 个 12 位 DAC,2 通道

（续）

项目	解　读
GPIO	80 个，可忍受 5V 的高压
调试	串口调试（SWD）和 JTAG 端口
定时器	8 个，TM1 ~ TM8
通信端口	13 个，包括 5 个串口、2 个 I^2C、3 个 SPI、1 个 CAN、1 个 USB、1 个 SDIO
FSMC	有

从表 1-1 来看，STM32F103VET6 的整体性能相对于价格（小于 30 元）来说较优秀。查看低密度和中密度系列的器件手册，STM32 性能有所缩减，但价格也降低了（STM32F103RBT6 价格不到 10 元）。由于 STM32 系列之间的全兼容性，相同封装可以 STM32 替换。STM32 性价比高是其流行的主要原因，而即便是同时使用了操作系统、图形用户端口、TCP/IP 协议栈、FAT32 文件系统等，STM32F103VET6 也可以应付，对于一般的应用足够了。

对于其他型号的 STM32 芯片，在官网查看其器件，下载对应的器件数据手册，可以和 STM32F103VET6 进行比较。差异性主要是 FLASH 和 SRAM 的尺寸，封装形式，AD、DA 及通信端口的个数等。

需要注意，1.25DMIPS/MHz 可以理解为处理能力，是在 1MHz 的时钟下，每秒执行整数指令的条数是 1.25M 条，也就是 1 个时钟周期执行超过一条的整数运算指令。这是因为采用了多级流水线的结构。这个指标，对于 ARM7 是 0.9DMIPS/MHz，ARM9 是 1.1DMIPS/MHz。这里的 D 是整数运算指令的意思，也就是采用整数指令进行测试得到的结果。这个处理能力相对于 STM32 比较便宜的价格来说是足够优秀的了。

要掌握 STM32 系统，首先要对系统的结构进行分析。

1.1.2　系统结构分析

STM32 系统结构框图如图 1-1 所示。该框图中包括了 STM32 的最核心的设备。图 1-1 中，Cortex-M3 内核是大脑，通过指令总线 ICode、数据总线 DCode、系统总线 System 与存储器和各种外设打交道，而总线矩阵实现了开关操作功能。多 DMA 通道是 STM32 系统的重要特征，有利于让 Cortex-M3 内核从繁重的数据转移工作中解放出来。高速总线 AHB 和低速总线 APB 之间通过 AHB-APB 桥进行交互。

图 1-1 上标明的数字处分别是：

（1）Cortex-M3 CPU 所在之处，是司令部，是大脑。

（2）总线矩阵，所谓矩阵就是矩阵开关，就如同电话接线员，根据请求进行接线。其核心功能是进行总线之间的连接。说得专业一点，它具有仲裁功能，由 4 个主动部件（DCode 总线、系统总线、DMA1 总线、DMA2 总线）及 4 个被动部件（FLASH 端口、SRAM、FSMC、AHB _ APB 桥）构成。

（3）闪存 FLASH 通过 FLASH 端口连接 CPU。FLASH 端口有两条路到 CPU，一条是传送指令的 ICode 总线，这样，FLASH 中的指令直接通过 ICode 总线送到 CPU 进行处理。另外一条是将 FLASH 的数据线通过总线矩阵接数据总线 DCode 连接到 CPU。

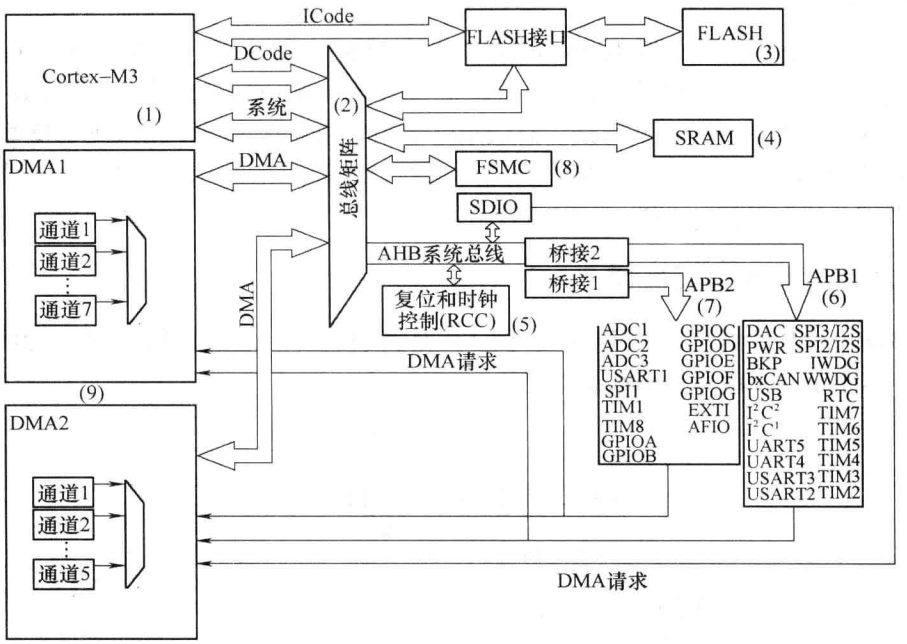

图 1-1　STM32 系统结构

（4）静态存储器（SRAM）通过总线矩阵连接 CPU。SRAM 是数据存放的地方，堆栈等也在这里，所以其速度和容量是关键的因素。通常直接在 FLASH 中执行代码，毕竟 SRAM 容量有限，但是也可以选择将程序从 FLASH 转移到 SRAM 来执行。

（5）复位和时钟控制（RCC），它是高速设备，连接在高速的 AHB 总线上。RCC 是一整套的时钟管理设备，通过对与之相关的寄存器的配置，可以设置 RCC 的工作模式，例如选择内部还是外部的时钟，选择高速还是低速的时钟，时钟分频或倍频的比率等。

（6）低速 APB1 外设，通过 APB1 总线接 APB 桥 2，然后通过 AHB 系统总线接矩阵开关，最后连接到 CPU。低速外设的速度上限是 36Mbit/s。串口、SPI、I^2C 及大部分的定时器都在其中。

（7）APB2 外设，也是 APB 外设，通过 APB2 总线接 APB 桥 1，然后通过高速 AHB 系统总线接矩阵开关，最后连接到 CPU。APB2 外设的速度上限是 72Mbit/s。GPIO 口、ADC、定时 1 和定时器 8 在其中。

（8）可变静态存储控制器（FSMC）。FSMC 具有软骨功，柔韧性非常强。FSMC 支持不同的静态存储器，具有多种存储器操作方法，并支持代码从 FSMC 扩展的存储器直接运行。通过对与 FSMC 相关的特殊功能寄存器的设置，FSMC 能够根据不同的外部存储器，发出相应的数据、地址、控制信号来匹配外部存储器，从而使得 STM32 能够应用各种不同类型、不同速度的外部静态存储器。由于 FSMC 的这种特性，可以降低系统设计的复杂性。通过 FSMC 可以以总线的方式与液晶控制器通信，从而驱动高精度大屏幕液晶，因此，FSMC 经常被应用于液晶控制器的管理。

（9）2 个 DMA 通道。DMA 技术在 STM32 中得到了较好的应用，采用 DMA 方式可以极大地减少 CPU 的负担。因为有了 DMA，CPU 不需要做所有的事。DMA 控制器通过 DMA 总

线连接到总线矩阵，再通过总线矩阵来与其他设备进行互联。

归纳一下 4 条总线。总线 ICode 是用来传输 FLASH 指令的，也用于预取指令。总线 DCode 是连接 FLASH 的数据线。系统总线连接着 Cortex-M3 的系统总线（外设总线）。DMA 总线是 DMA 控制器进行控制的总线。总线矩阵使用轮换算法进行仲裁来决定这些总线的连接。通过矩阵开关之后，AHB 系统总线是高速的与外设通信的总线，再通过桥接芯片后，将 AHB 系统总线与低速的 APB 总线进行连接。

1.1.3　芯片封装和引脚概述

STM32F103VE 的封装是 LQFP100，图 1-2 是其引脚图。通过比较发现，LQFP100 封装的 STM32F103 不同子型号引脚都是相同的。

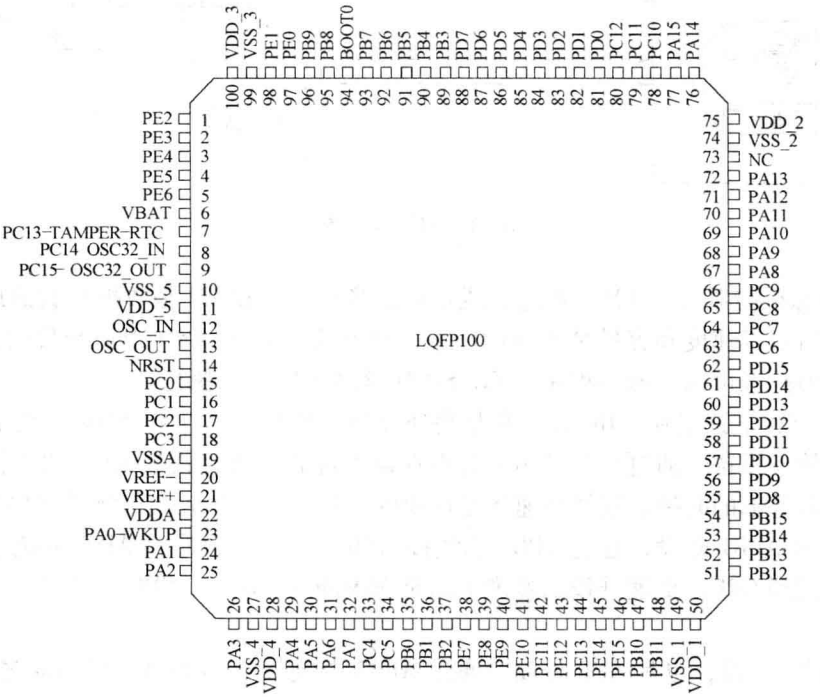

图 1-2　STM32F103VE 引脚图

要进行硬件设计必须对每个引脚的功能和电气特性都很清楚，引脚描述在器件数据手册中对应的是引脚及其描述（Pinouts and Pin Descriptions）部分。

其中，6 脚 VBAT 为电池供电端，在未提供电源（VDD 不满足要求或无电压）时为实时时钟 RTC、备份寄存器和外部时钟 32kHz 晶体振荡器供电。当不使用电池的时候必须将其连接到 VDD。

所有的 VDD 都是电源输入端，所有的 VSS 都接地。这些电源和地引脚一共 5 组（VSS1、VDD1）～（VSS5、VDD5），共用 10 个引脚。

GPIO 为常规输入/输出端口，占据了大量的引脚，共 80 个。该类引脚以 P 打头，都是

16位的，分为5个组，分别为PA0～PA15，PB0～PB15，PC0～PC15，PD0～PD15，PE0～PE15。这些端口大都有多重功能，既可作为I/O又可编程为其他功能。

OSC _ IN和OSC _ OUT是外部时钟输入引脚，共用2个引脚。

NRST为外部复位引脚。该引脚上的低电平将导致系统复位。

VREF－和VREF＋为ADC和DAC提供参考电压，VSSA和VDDA为ADC和DAC提供电源，共用4个引脚。如果项目需要高精度的数模转换和模数转换，应该将其连接到外部参考源。

BOOT0和BOOT1为芯片启动模式配置引脚，其中BOOT1与PB2复用引脚。

未用引脚NC1个。

综合统计GPIO引脚80个，电源引脚10个，其他引脚10个，合计引脚数量为100个。

那么，SPI引脚、FSMC引脚、USART引脚等都到哪里去了呢？不用担心，STM32可对引脚功能进行编程，大多数的引脚都可以通过编程定义其功能，称之为复用。

其中，特殊的复用引脚有脚7、8、9，它们在作GPIO使用时，电流只能达到3mA，频率不能超过2MHz。如脚7（PC13-TAMPER-RTC）这样的引脚是复用引脚，默认时为TAMPER-RTC，也可以配置为PC13（GPIO的13位）。引脚8和9为PC14-OSC32-IN和PC15-OSC32-OUT，默认时为外部时钟输入。当它们在作I/O使用时，电流也只能达到3mA。

其他引脚则没有这样的限制，在作I/O使用时可以具有较高的输出能力（一般为25mA）。至于这些引脚都可以配置为什么样的复用功能，在厂家器件手册中有详细的说明。

接下来一节进入核心——Cortex-M3处理器。

1.2 Cortex-M3处理器

STM32采用的是32位处理器核Cortex-M3，Cortex-M3中的各种寄存器是主要的编程对象。即使使用库函数，还是需要对Cortex-M3的结构有所了解。本节简明扼要地研究Cortex-M3处理器。

因为Cortex-M3是ARM公司的知识产权。请在ARM公司网站获取资料，主要资料是Cortex-M3技术参考手册。

1.2.1 Cortex-M3的定位和应用

从图1-3可见，嵌入式处理器核Cortex-M3的容量（Capability）和执行功能（Performance Functionality）都居中，但其性价比是当今较好的品种之一，也是现在较流行的品种之一。嵌入式处理器主要着重于在各种功耗敏感型应用中提供具有高确定性的实时行为。这些处理器通常执行实时操作系统（RTOS）和用户开发的应用程序代码。既然是运行RTOS，那么μC/OS比较适用于在装载了Cortex-M3的STM32上运行，Linux就不合适。

以Cortex-M3核为中心的处理器的应用范围ARM公司定义为商业微控制器、汽车控制系统、电动机控制系统、大型家用电器控制器、无线传感器网络、有线传感器网络、大容量存储控制器、打印机、网络设备，当然不在这8项里的应用也有很多，例如，环境监测、各种测控系统甚至航空航天系统等。读者在掌握了这门技术之后，根据所在单位的特点，自然会应用在需要的领域。

下面介绍的是 Cortex-M3 处理器结构。

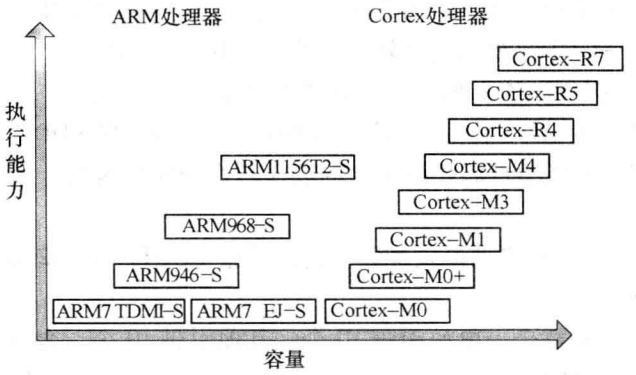

图 1-3　Cortex-M3 定位

1.2.2　Cortex-M3 处理器结构

Cortex-M3 处理器结构如图 1-4 所示。

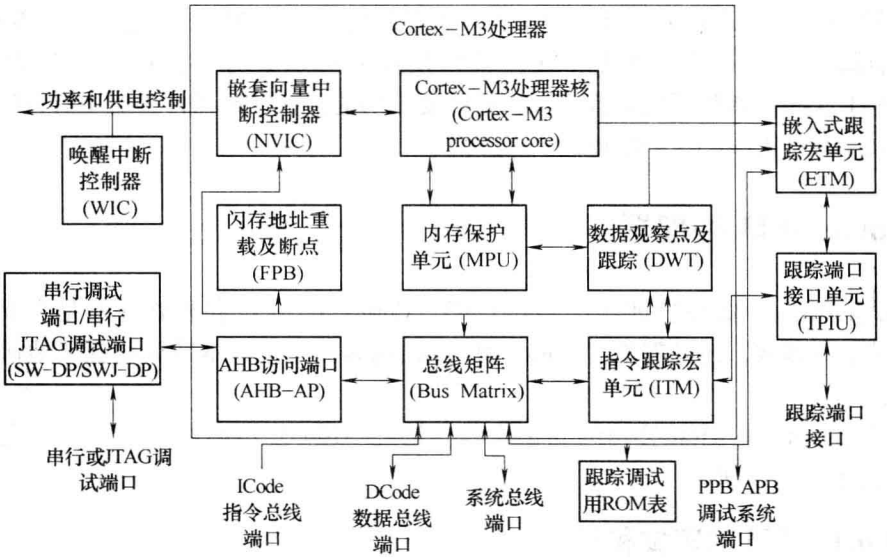

图 1-4　Cortex-M3 处理器结构

Cortex-M3 处理器结构中各个部分的解释和功能如下：

嵌套向量中断控制器（NVIC）：负责中断控制。该控制器和内核是紧耦合的，提供可屏蔽、可嵌套、动态优先级的中断管理。

Cortex M3 处理器核（Cortex-M3 processor core）：Cortex-M3 处理器核是处理器的核心所在。

闪存地址重载及断点（FPB）：实现硬件断点以及代码空间到系统空间的映射。

内存保护单元（MPU）：内存保护单元 MPU 的主要作用是实施存储器的保护，它能够在系统或程序出现异常而非正常地访问不应该访问的存储空间时，通过触发异常中断而达到

提高系统可靠性的目的。STM32 系统并没有使用该单元。

数据观察点及跟踪单元（DWT）：调试中用于数据观察功能。

AHB 访问端口（AHB-AP）：高速总线 AHB 访问端口将 SW/SWJ 端口的命令转换为 AHB 的命令传送。

总线矩阵（Bus Matrix）：Cortex-M3 总线矩阵，CPU 内部的总线通过总线矩阵连接到外部的 ICode、DCode 及系统总线。

指令跟踪宏单元（ITM）：可以产生时间戳数据包并插入到跟踪数据流中，用于帮助调试器求出各事件的发生时间。

唤醒中断控制器（WIC）：WIC 可以使处理器和 NVIC 处于一个低功耗睡眠的模式。

嵌入式跟踪宏单元（ETM）：调试中用于处理指令跟踪。

串行调试端口/串行 JTAG 调试端口（SW-DP/SWJ-DP）：串行调试的端口。

跟踪端口接口单元（TPIU）：跟踪端口的接口单元。用于向外部跟踪捕获硬件发送调试停息的接口单元，作为来自 ITM 和 ETM 的 Cortex-M3 内核跟踪数据与片外跟踪端口之间的桥接。

Cortex-M3 既然是 32 位处理器核，地址线、数据线都是 32 位的。采用了哈佛结构。哈佛这种并行结构将程序指令和数据分开进行存储，其优点是因为在一个机器周期内处理器可以并行获得执行字和操作数，提高了执行速度。简单理解一下，指令存储器和其他数据存储器采用不同的总线（ICode 和 DCode 总线），可并行取得指令和数据。

1.2.3　Cortex-M3 寄存器

Cortex-M3 寄存器部分对于以后的编程是非常重要的，尤其是当进行 μC/OS 移植需要写汇编的时候，须直接编程操作这些寄存器，因此需要先认识这些寄存器。表 1-2 列出了 Cortex-M3 内核拥有的寄存器。

表 1-2　Cortex-M3 寄存器

寄存器		名　称	说　明
R0		通用寄存器（16 位 THUMB 指令及 32 位 THUMB2 指令都可访问）	通用寄存器
R1			
R2			
R3			
R4			
R5			
R6			
R7			
R8		通用寄存器（仅 32 位 THUMB2 指令可访问）	
R9			
R10			
R11			
R12			
R13	MSP	主程序堆栈指针	用于 OS 内核和堆栈处理
	PSP	应用程序堆栈指针	用于应用程序
R14		链接寄存器	存放子程序返回地址
R15		程序指针	存放程序地址

（续）

寄存器		名　称	说　明
程序状态寄存器 xPSR	APSR	ALU 标志寄存器	存放上条指令结果的标志，包括 N、Z、C、V、Q 位
	IPSR	中断号寄存器	存放中断号
	EPSR	执行状态寄存器	含 T 位，在 CM3 中 T 位必须是 1。含 ICI 位，记录下一个即将传送的寄存器是哪一个
PRIMASK		中断关闭寄存器	为 1 时关闭所有可屏蔽中断
FAULTMASK		异常关闭寄存器	为 1 时屏蔽除 NMI 外所有异常
BASEPRI		屏蔽优先级寄存器	定义屏蔽优先级的阀值，所有优先级大于该值的中断被关闭
CONTROL		状态控制寄存器	定义特权级别，选择堆栈指针

R0 ~ R12 是最基本的寄存器，是通用寄存器。R0 ~ R7 所有的指令都可以访问，R8 ~ R12 则不是，需要 32 位的 Thumb2 指令才能访问。

R13 是堆栈指针寄存器。这样的寄存器其实有两个，一个是 MSP，一个是 PSP，但是同一时间点上，只能使用其中的一个。MSP 是系统复位时使用的，也是中断服务程序使用的；PSP 则是可以由用户程序使用的。Cortex-M3 有两种模式，特权模式和线程模式。普通应用程序代码，例如用户主程序 main 或用户函数运行在线程模式，可以使用 MSP 或 PSP。而异常处理代码即中断服务程序 ISR 代码运行在特权模式，只能使用 MSP。

MSP 和 PSP 的不同也为实时操作系统的使用提供了便利，不经过特殊设置，不带操作系统的程序永远使用 MSP，中断服务程序和用户程序使用同一个堆栈。当使用了如 μC/OS 等操作系统的时候，每个任务有自己的堆栈，让用户程序使用 PSP，中断服务程序使用 MSP。

R14 是链接寄存器，是用来存储返回地址的。如果主程序调用函数，就要把当前的地址保存到 R14。需要注意的是，在中断中，R14 是另有含义的，被称为 EXC _ RETURN，含义和在用户程序中是完全不同的。位 31 ~ 位 4 全为 1，低 4 位中位 3 为 1 表示返回后进入线程模式，为 0 表示返回后进入特权模式 HANDLER。位 2 为 1 表示返回后使用 PSP，为 0 表示返回后使用 MSP。位 1 必须为 0。位 0 为 1 表示返回 Thumb 状态，在 Cortex-M3 处理器核时必须为 1，为 0 表示返回 ARM 状态。通过对 R14 的设置，可以在中断返回后，选择使用 MSP 或 PSP，这在嵌入式实时操作系统 μC/OS 移植的时候会被使用到。

EXC-RETURN 按位说明见表 1-3。

表 1-3　EXC _ RETURN 按位说明

位	值	
	0	1
31 ~ 4	非法	必为 1
3	从中断返回后将进入特权模式	从中断返回后将进入线程模式
2	从中断返回后使用 MSP	从中断返回后使用 PSP
1	保留，必为 0	非法
0	返回 ARM 状态	返回 Thumb 状态，Cortex-M3 下必为 1

R15 是程序寄存器，里面是程序的地址。如果对它进行修改，就改变了程序的走向。

其他的寄存器是特殊功能寄存器，其中程序状态寄存器 xPSR 是一个 32 位的寄存器，但又可以划分为 3 个子寄存器，31 ~ 27 位为 APSR，26 ~ 9 位为 EPSR，8 ~ 0 位为 IPSR。PRIMASK、FAULTMASK、BASEPRI 都是用来设置中断屏蔽的。CONTROL 寄存器（状态控制寄存器）用于定义特权级别，选择堆栈。

Cortex-M3 操作模式如图 1-5 所示。

图 1-5　Cortex-M3 操作模式

说到 CONTROL 寄存器，就会说到特权级和用户级，特权模式和线程模式。其实所谓特权模式，就是中断服务程序运行，其他的都是在线程模式。当 STM32 开始运行的时候，首先进入复位中断程序，这是在特权模式，然后，从中断中返回，进入用户程序执行，就进入了线程模式。

例如，程序在主程序中运行，循环点亮流水灯，是出于线程模式。这时串行接口（串口）有数据送过来，串口中断发生，将进入中断服务程序 ISR，就进入特权模式。从串口中断中返回，继续控制流水灯，就回到线程模式。

特权模式下，必然运行在特权级，可以访问所有的资源，如所有的寄存器，而在用户级则不能，功能受限，例如不能够访问 CONTROL 寄存器，经过测试在用户级访问特殊功能寄存器指令无任何执行效果。用户程序在执行的时候可以处在特权级也可以在用户级，在没有进行设置的时候，是运行在特权级的。由特权级到用户级的方法就是可以通过设置 CONTROL 寄存器，但是要从用户级回到特权级却只有通过中断服务程序，因为在用户级是不能够设置 CONTROL 寄存器的，而中断服务程序是运行在特权级，可以通过设置前面分析的 R14（在堆栈中变身为 EXC _ RETURN）寄存器，在中断返回后改变特权级及使用的堆栈！

如表 1-4 所示，CONTROL 有 2 位，CONTROL［0］为 0 表示是处于特权级，为 1 表示处于用户级。只有在特权级，才可以将其修改为 1，进入了用户级的线程模式。CONTROL［1］也是如此，其为 0 表示选择主堆栈指针 MSP，为 1 表示选择线程堆栈指针 PSP。笔者设计了让用户程序离开特权模式的代码如代码 1-1 所示。

表 1-4 CONTROL 寄存器按位说明

位	值	
	0	1
1	选择主堆栈指针 MSP	选择线程堆栈指针 PSP
0	特权级的线程模式	用户级的线程模式

代码 1-1 离开特权模式的代码

```
LeaveSpecialPower
    MRS R0, CONTROL
    ORR R0, #1
    MSR CONTROL, R0
    BX      LR
```

使用 JLINK 在线调试，跟踪每一条语句，查看寄存器的值。MRS 语句将特殊功能寄存器 CONTROL 的值赋给 R0，R0 的值是 0，然后将 R0 与 1 按位或运算，R0 的值从 0 变为 1，然后用 MSR 将 R0 的值赋给 CONTROL，CONTROL 的值变为 1。这样，就进入了用户级的线程模式。这之后读者可以加入一些语句来测试，再使用 MRS、MSR 指令不会有任何的效果，说明用户级下对特殊功能寄存器进行了保护。如果要再返回特权级，必须进中断服务程序，并在中断服务程序通过设置 EXC _ RETURN 寄存器来达到修改返回后的模式和使用的堆栈。

1.2.4 堆栈

堆栈这种后进先出的数据结构在编程中具有重要的意义，在函数调用的时候，将寄存器的值顺序压入堆栈，当程序返回的时候，将寄存器的值从堆栈中逆序弹出，这样寄存器的值不会发生变化，就不会产生错误。进入中断的时候也是这个道理

Cortex-M3 使用的是向下增长的满栈。所谓向下增长是指向低地址增长，所谓满栈是指堆栈指针 SP 指向最后一个被压入的数据。另外，Cortex-M3 使用小端模式，高字节在高地址。堆栈以字（4B）为单位，那么 Cortex-M3 堆栈的说明图如图 1-6 所示。

在图 1-6 中，堆栈的栈顶是 0x20007FF8，堆栈中已经压入了 2 个字 0x0102030A 和 0x00000009。图的下半部分说明了在小端模式下，0x0102030A 按字节存储的方法及每个字节存储的方法，一个字中数值的高字节存储在高地址，一个字节中数值的高位存储在高位。从 0x20007FFC 地址开始存储 0x0102030A，采用小端模式的时候，低字节存储在低地址 0x20007FFC，即 0x20007FFC 的内容为 0x0A，而高地址 0x20007FFF 存储的是最高字节的 0x01。

当压入（PUSH）4 个字后，堆栈向低地址增长，栈顶变为 0x20007FE8。当再弹出（POP）3 个字后，栈顶变为 0x20007FF4。PUSH 的过程是 SP 先减 4，然后将要压入的数据存储到 SP 指示的堆栈地址空间，POP 的过程则相反。于是，图 1-6 中，当堆栈是空的时候，SP 的值应该是 0x20008000。

另外，Cortex-M3 具有主堆栈指针 MSP 和进程堆栈指针 PSP，支持双堆栈机制。当使用