普通高等教育“十三五”规划教材
ST首批认证“STM32精品课程”教材
“蓝桥杯”嵌入式设计与开发竞赛培训教材

ARM Cortex-M3 系统设计与实现 ——STM32基础篇（第2版）

◆ 郭书军 编著

中国工信出版集团
電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

普通高等教育“十三五”规划教材
ST 首批认证“STM32 精品课程”教材
“蓝桥杯”嵌入式设计与开发竞赛培训教材

ARM Cortex-M3
系统设计与实现
——STM32 基础篇
（第 2 版）

郭书军　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书以 STM32 系列 32 位 Flash MCU 为例，以“蓝桥杯”嵌入式设计与开发竞赛训练板为硬件平台，以“一切从简单开始”为宗旨，介绍 ARM Cortex-M3 系统的设计与实现。

全书分为 10 章，第 1 章简单介绍 STM32 MCU 和 SysTick 的结构；第 2、3 章以一个简单的嵌入式系统设计为例，详细介绍 SysTick、GPIO 和 USART 的应用设计；第 4、5 章分别介绍 SPI 和 I^2C 的结构和设计实例；第 6、7 章分别介绍 TIM 和 ADC 的结构和设计实例；第 8、9 章分别介绍 NVIC 和 DMA 的结构和设计实例；第 10 章介绍竞赛扩展板的使用。书后附有实验指导，以方便实验教学。

本书所有设计程序均为原创，并经过两轮学生实验的改进，内容简单易懂，特别适合初学者学习参考，也可以作为嵌入式系统设计教材供电子、通信和自动化等相关专业人员使用。

图书在版编目（CIP）数据

ARM Cortex-M3 系统设计与实现. STM32 基础篇 / 郭书军编著. —2 版. —北京：电子工业出版社，2018.10
普通高等教育“十三五”规划教材
ISBN 978-7-121-35198-3

Ⅰ. ①A… Ⅱ. ①郭… 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 234009 号

策划编辑：赵玉山
责任编辑：刘真平
印　　刷：北京七彩京通数码快印有限公司
装　　订：北京七彩京通数码快印有限公司
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：787×1 092　1/16　印张：15.75　字数：403.2 千字
版　　次：2014 年 1 月第 1 版
　　2018 年 10 月第 2 版
印　　次：2018 年 10 月第 1 次印刷
定　　价：48.00 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 zlts@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：（010）88254556，zhaoys@phei.com.cn。

序　言

世界万物，智能互联，这是当下产业界正在推动的新一代技术发展和服务的方向，万物互联后产生的大数据可以进一步提升社会效率和推动产业升级，将产生巨大的社会价值。

产业升级，技术创新，离不开与时俱进的人才。

人才的培养，高等学校是最大的培养基地。

作为致力于长期服务中国市场、为中国的产业发展提供最新技术产品的公司，意法半导体一直为中国的用户提供最前沿的技术，推动生态系统的建设，为用户提供从芯片到方案的支持。

为了向产业界提供有技术的人才，我们从数年前就开始系统性地和高校开展人才培养计划，这个计划包含下列 3 个方面：

（1）推动精品课程建设：协助高校课程改革，将最前沿的技术和产品带入教学和实验中，让学生接触体验到最新技术，为以后就业打好基础。

（2）实施 TTT（老师培训老师）项目：邀请有开课经验的老师开展培训，帮助打算开课的老师提升信心，分享教育经验和体会。

（3）开展大学生智能互联校园创新大赛：让学生通过大赛进一步夯实所学的知识，在一个公平的环境中模拟企业项目，提升自身能力和信心。

在过去数年的探索中，我们惊喜地发现已经有众多的老师在人才培养方面取得了优异的成果，并且积极分享和持续优化、全方位推动高校课程改革和人才培养。

北方工业大学电子信息工程学院的郭书军老师就是其中一位，他在本科生和研究生教育方面，一直倡导课程和时代技术发展紧密结合，把市场主流的技术带进课堂，从 2010 年开始把 STM32 作为嵌入式系统设计课程的主要教学载体，升级课程体系，同时鼓励学生积极参加各项竞赛，以赛代练，提高技术能力。同时，郭书军老师也为工信部人才交流中心举办的“蓝桥杯”嵌入式设计与开发竞赛做出了巨大的贡献。

喜闻郭书军老师对《ARM Cortex-M3 系统设计与实现——STM32 基础篇》进行改版优化，将硬件平台更新为竞赛训练板，并在原有寄存器编程的基础上添加了库函数介绍和库函数编程。后来又增加了实验的视频演示，更方便大家学习和实验。新一版教材凝聚了郭书军老师的辛勤付出，希望为广大学生带来一本优质的教材，也为其他院校老师提供很好的借鉴模板。

曹锦东
意法半导体（中国）投资有限公司
中国区微控制器市场及应用总监
2018 年 8 月

前　言

《ARM Cortex-M3 系统设计与实现——STM32 基础篇》出版发行后，由于其简单实用的特点，受到读者的欢迎。虽然寄存器编程更有利于理解硬件原理，但有一定难度，限制了它的使用范围。为了惠及更多读者，更为了作为“蓝桥杯”嵌入式设计与开发竞赛的培训教材，本书将硬件平台更新为竞赛训练板，并在原有寄存器编程的基础上添加了库函数介绍和库函数编程。竞赛扩展板推出后，又增加了竞赛扩展板各功能模块的使用介绍。后来又增加了实验的视频演示，可通过扫描二维码打开观看（目录中标压缩二维码图标的章节含二维码），更方便大家学习和实验。

全书分为 10 章，以竞赛试题为主线，依次介绍 GPIO、USART、SPI、I^2C、TIM、ADC、NVIC、DMA 的结构和程序设计与实现，最后介绍竞赛扩展板各功能模块的使用。

第 1 章介绍 STM32 MCU 和 SysTick 的结构，重点介绍复位和时钟控制（RCC）库函数和 SysTick 库函数，方便后续章节的使用。

第 2 章和第 3 章分别在介绍 GPIO、USART 结构和库函数的基础上，以嵌入式竞赛训练板为硬件平台，使用库函数和寄存器两种软件设计方法，介绍 GPIO 和 USART 的软件设计与实现方法，包括新建工程、新建并添加 C 语言源文件、添加库文件、生成目标程序文件、调试和运行目标程序等，重点介绍使用仿真器和调试器调试及运行目标程序的步骤和方法。

第 4 章和第 5 章分别介绍 SPI、I^2C 的结构和库函数及程序设计与实现。SPI 的编程操作和 USART 相似，软件设计实例主要实现了 SPI 的环回。I^2C 的编程操作相对复杂一些，设计实例用两种方法实现了通过 I^2C 读/写 2 线串行 EEPROM。

第 6 章和第 7 章分别介绍 TIM、ADC 的结构和库函数及程序设计。TIM 设计实例实现了 1s 定时、矩形波输出和矩形波测量程序设计等，ADC 设计实例用 ADC 规则通道实现外部输入模拟信号的模数转换和用 ADC 注入通道实现内部温度传感器的温度测量等。

第 8 章和第 9 章分别介绍 NVIC、DMA 的结构和库函数及设计实例。中断和 DMA 是高效的数据传送控制方式，对前面介绍的接口和设备数据传送查询方式稍做修改即可实现中断功能，再结合 DMA 可以实现数据的批量传送。

第 10 章介绍竞赛扩展板各功能模块的使用，包括数码管、ADC 按键、湿度传感器、温度传感器和加速度传感器的使用。

书末附有 STM32 库函数、引脚功能、训练板和扩展板介绍等实用资料供读者参考，还包含 8 个实验指导以方便实验教学。

本书所有设计程序均为原创，并在竞赛训练板和 Keil 4.12 环境下测试通过。

参与本书编写和程序调试的还有王玉花、刘哲、王硕、孟群升和田香。在本书的编写过程中，得到意法半导体（中国）投资有限公司中国区微控制器市场及应用总监曹锦东先生的大力支持，他在百忙中为本书撰写了序言；在本书的出版过程中，得到北方工业大学的资助及电子工业出版社赵玉山先生和刘真平女士的支持，在此一并表示衷心的感谢。

由于编著者水平所限，书中难免会有不妥之处，敬请广大读者批评指正。

E-mail：cortex_m3@126.com，QQ 群：STM32 学习（489189201）。

编著者

2018 年 2 月

目　录

第 1 章　STM32 MCU 简介

STM32 系列 32 位 Flash 微控制器基于 ARM Cortex-M 系列处理器，旨在为 MCU 用户提供新的开发自由度。它包括一系列 32 位产品，具有高性能、实时功能、数字信号处理、低功耗与低电压操作特性，同时还保持了集成度高和易于开发的特点。无可比拟且品种齐全的 STM32 产品基于行业标准内核，提供了大量工具和软件选项，使该系列产品成为小型项目和完整平台的理想选择。

作为一个主流的微控制器系列，STM32 满足工业、医疗和消费电子市场的各种应用需求。凭借这个产品系列，ST 在全球的 ARM Cortex-M 微控制器中处于领先地位，同时树立了嵌入式应用的里程碑。该系列最大化地集成了高性能与一流外设和低功耗、低电压工作特性，在可以接受的价格范围内提供简单的架构和易用的工具。

该系列包含 5 个产品线，它们之间引脚、外设和软件相互兼容：

- 基本型系列 STM32F101：36MHz 最高主频，具有高达 1MB 的片上闪存
- USB 基本型系列 STM32F102：48MHz 最高主频，具有全速 USB 模块
- 增强型系列 STM32F103：72MHz 最高主频，具有高达 1MB 的片上闪存，集成电机控制、USB 和 CAN 模块
- 互联型系列 STM32F105/107：72MHz 最高主频，具有以太网 MAC、CAN 及 USB 2.0 OTG 功能

本书以增强型系列 STM32F103 为核心，介绍 STM32 MCU 的设计应用。

1.1　STM32 MCU 结构

STM32 MCU 由控制单元、从属单元和总线矩阵三大部分组成，控制单元和从属单元通过总线矩阵相连接，如图 1.1 所示。

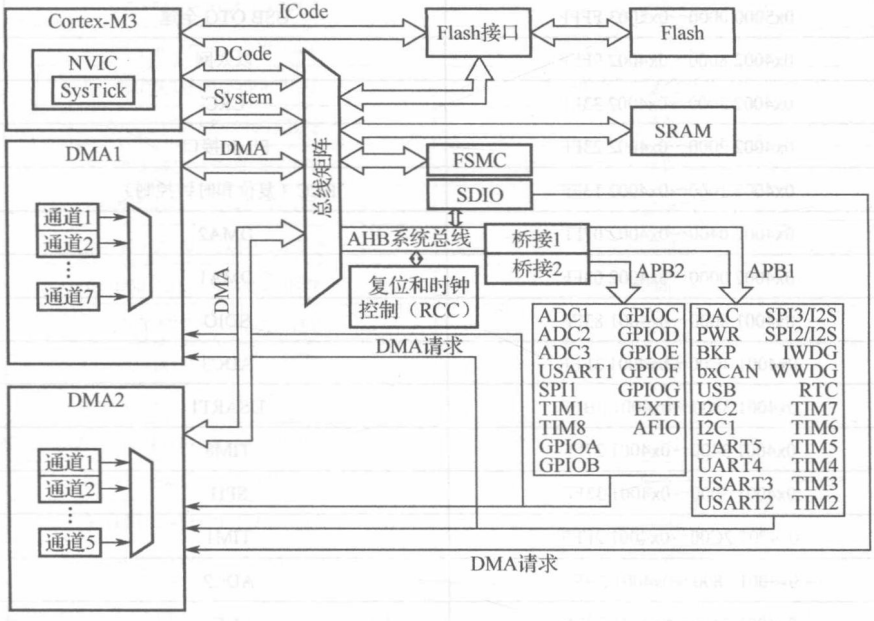

图 1.1　STM32 MCU 结构

控制单元包括 Cortex-M3 内核和两个 DMA 控制器（DMA1 和 DMA2）。其中 Cortex-M3 内核通过指令总线 ICode 从 Flash 中读指令，通过数据总线 DCode 与存储器交换数据，通过系统总线 System（设备总线）、高性能系统总线 AHB 和高级设备总线 APB 与设备交换数据。

从属单元包括存储器（Flash 和 SRAM 等）和设备（连接片外设备的接口和片内设备）。其中设备通过 AHB-APB 桥接器和总线矩阵与控制单元相连接，与 APB1 相连的是低速设备（最高频率 36MHz），与 APB2 相连的是高速设备（最高频率 72MHz）。

连接片外设备的接口有并行接口和串行接口两种，并行接口即通用 I/O 接口 GPIO，串行接口有通用同步/异步收发器接口 USART、串行设备接口 SPI、内部集成电路总线接口 I^2C、通用串行总线接口 USB 和控制器局域网络接口 CAN 等。

片内设备有定时器 TIM、模数转换器 ADC 和数模转换器 DAC 等，其中定时器包括高级控制定时器 TIM1/8、通用定时器 TIM2-5、基本定时器 TIM6/7、实时钟 RTC、独立看门狗 IWDG 和窗口看门狗 WWDG 等。

系统复位后，除 Flash 接口和 SRAM 时钟开启外，所有设备都被关闭，使用前必须设置时钟使能寄存器（RCC_APBENR）开启设备时钟。

1.2 STM32 MCU 存储器映像

STM32 MCU 的程序存储器、数据存储器和输入/输出端口寄存器被组织在同一个 4GB 的线性地址空间内，存储器映像如表 1.1 所示。

表 1.1 STM32 MCU 存储器映像表

地址范围		设备名称	备注
0xE000 0000～0xE00FFFFF（1MB）		内核设备	
内核设备	0xE000E100～0xE000E4EF	NVIC（嵌套矢量中断控制）	详见表 8.2
	0xE000E010～0xE000E01F	SysTick（系统滴答定时器）	详见表 1.9
0x4000 0000～0x5FFFFFFF（512MB）		片上设备	
AHB	0x5000 0000～0x5003 FFFF	USB OTG 全速	
	0x4002 8000～0x4002 9FFF	以太网	
	0x4002 3000～0x4002 33FF	CRC	
	0x4002 2000～0x4002 23FF	Flash 接口	
	0x4002 1000～0x4002 13FF	RCC（复位和时钟控制）	详见表 1.2
	0x4002 0400～0x4002 07FF	DMA2	
	0x4002 0000～0x4002 03FF	DMA1	详见表 9.2
	0x4001 8000～0x4001 83FF	SDIO	
APB2	0x4001 3C00～0x4001 3FFF	ADC3	
	0x4001 3800～0x4001 3BFF	USART1	详见表 3.3
	0x4001 3400～0x4001 37FF	TIM8	
	0x4001 3000～0x4001 33FF	SPI1	详见表 4.2
	0x4001 2C00～0x4001 2FFF	TIM1	详见表 6.2
	0x4001 2800～0x4001 2BFF	ADC2	详见表 7.2
	0x4001 2400～0x4001 27FF	ADC1	详见表 7.2

续表

地址范围		设备名称	备　注
APB2	0x4001 2000～0x4001 23FF	GPIOG	
	0x4001 1C00～0x4001 1FFF	GPIOF	
	0x4001 1800～0x4001 1BFF	GPIOE	
	0x4001 1400～0x4001 17FF	GPIOD	
	0x4001 1000～0x4001 13FF	GPIOC	详见表 2.1
	0x4001 0C00～0x4001 0FFF	GPIOB	详见表 2.1
	0x4001 0800～0x4001 0BFF	GPIOA	详见表 2.1
	0x4001 0400～0x4001 07FF	EXTI	详见表 8.6
	0x4001 0000～0x4001 03FF	AFIO	
APB1	0x4000 7400～0x4000 77FF	DAC	
	0x4000 7000～0x4000 73FF	PWR（电源控制）	
	0x4000 6C00～0x4000 6FFF	BKP （后备寄存器）	
	0x4000 6800～0x4000 6BFF	bxCAN2	
	0x4000 6400～0x4000 67FF	bxCAN1	
	0x4000 6000～0x4000 63FF	USB/CAN 共享的 512B SRAM	
	0x4000 5C00～0x4000 5FFF	USB 全速设备寄存器	
	0x4000 5800～0x4000 5BFF	I2C2	详见表 5.2
	0x4000 5400～0x4000 57FF	I2C1	详见表 5.2
	0x4000 5000～0x4000 53FF	UART5	
	0x4000 4C00～0x4000 4FFF	UART4	
	0x4000 4800～0x4000 4BFF	USART3	
	0x4000 4400～0x4000 47FF	USART2	
	0x4000 4000～0x4000 43FF	保留	
	0x4000 3C00～0x4000 3FFF	SPI3/I2S3	
	0x4000 3800～0x4000 3BFF	SPI2/I2S2	
	0x4000 3400～0x4000 37FF	保留	
	0x4000 3000～0x4000 33FF	IWDG （独立看门狗）	
	0x4000 2C00～0x4000 2FFF	WWDG （窗口看门狗）	
	0x4000 2800～0x4000 2BFF	RTC	详见表 6.20
	0x4000 1400～0x4000 17FF	TIM7	
	0x4000 1000～0x4000 13FF	TIM6	
	0x4000 0C00～0x4000 0FFF	TIM5	
	0x4000 0800～0x4000 0BFF	TIM4	
	0x4000 0400～0x4000 07FF	TIM3	
	0x4000 0000～0x4000 03FF	TIM2	
0x2000 0000～0x3FFF FFFF (512MB)		SRAM	
0x00000000～0x1FFF FFFF (512MB)		FLASH	
FLASH	0x1FFF F800～0x1FFF F80F	选择字节	
	0x1FFF F000～0x1FFF F7FF	系统存储器	
	0x0800 0000～0x0801 FFFF	主存储器	

存储器映像在 stm32f10x_map.h（V2.0.1）或 stm32f10x.h（V3.5.0）中定义。两者的主要区别

是数据类型定义不同，如寄存器类型定义前者使用 VU32，后者使用_IO uint32_t。

1.3 STM32 MCU 系统时钟树

STM32 MCU 系统时钟树由系统时钟源、系统时钟 SYSCLK 和设备时钟等部分组成，如图 1.2 所示。

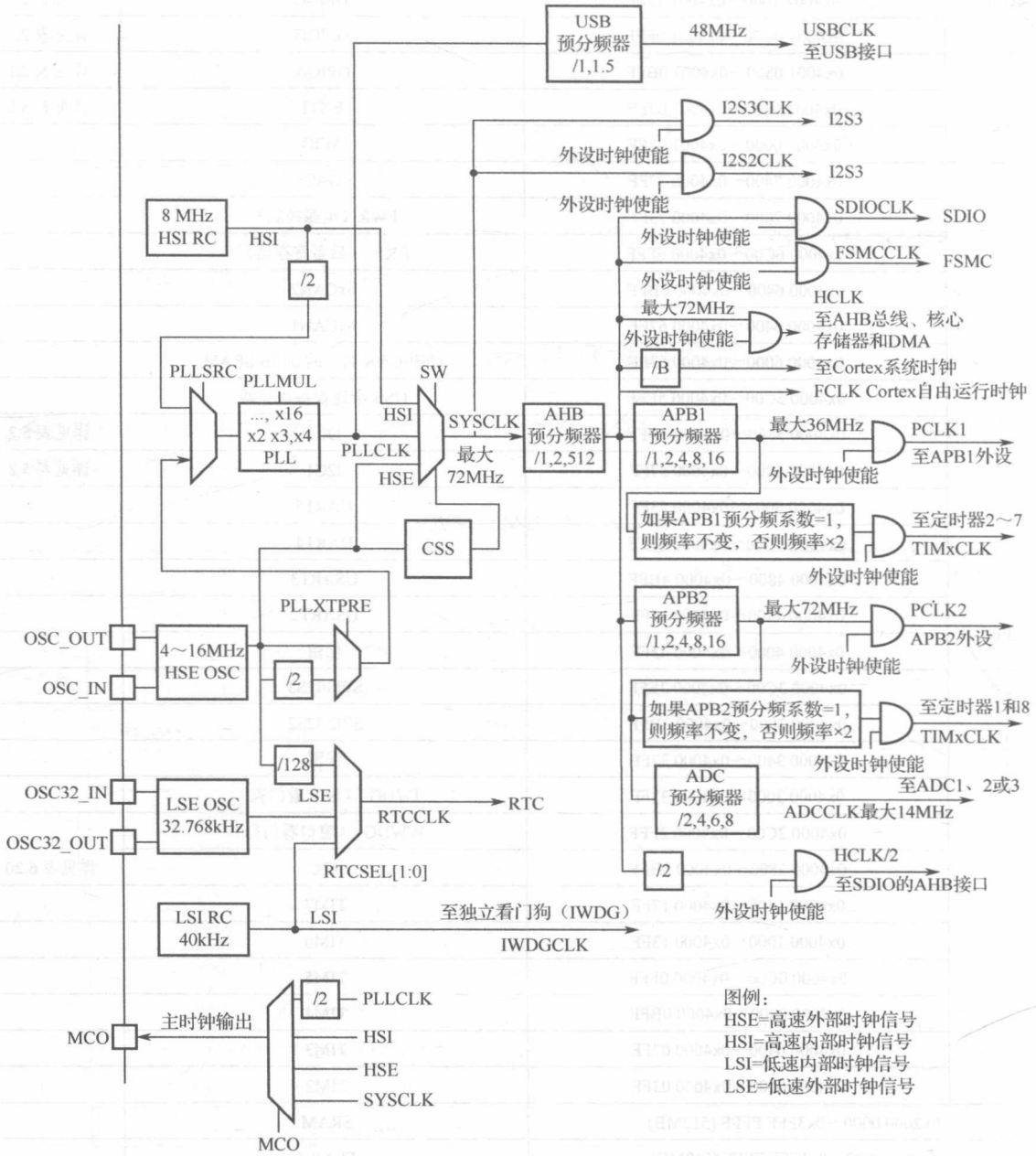

图 1.2 STM32 MCU 系统时钟树

系统时钟源有 4 个：高速外部时钟 HSE（4～16MHz）、低速外部时钟 LSE（32.768kHz）、高速内部时钟 HSI（8MHz）和低速内部时钟 LSI（40kHz），其中外部时钟用晶体振荡器 OSC 实现，内部时钟用 RC 振荡器实现。

系统时钟 SYSCLK（最大 72MHz）可以是 HSE 或 HSI，也可以是 HSE 或 HSI 通过锁相环 2～16 倍频后的锁相环时钟 PLLCLK。系统复位后的系统时钟为 HSI，这就意味着即使没有 HSE 系统也能正常工作，只是 HSI 的精度没有 HSE 高。

SYSCLK 经 AHB 预分频器分频后得到 AHB 总线时钟 HCLK（最大 72MHz），HCLK 经 APB1/APB2 预分频器分频后得到 APB1/APB2 总线时钟 PCLK1（最大 36MHz）和 PCLK2（最大 72MHz），PCLK1 和 PCLK2 分别为相连的设备提供设备时钟。

系统时钟树中的时钟选择、预分频值和外设时钟使能等都可以通过对复位和时钟控制（RCC）寄存器编程实现，复位和时钟控制（RCC）寄存器如表 1.2 所示（RCC 的基地址是 0x4002 1000）。

表 1.2　复位和时钟控制（RCC）寄存器

偏移地址	名　称	类　型	复 位 值	说　明
0x00	CR	读/写	0x0000 XX83	时钟控制寄存器（HSIRDY=1，HSION=1，详见表 1.3）
0x04	CFGR	读/写	0x0000 0000	时钟配置寄存器（SYSCLK=HSI，AHB、APB1 和 APB2 均不分频，即频率均为 8MHz，定时器时钟频率也为 8MHz，ADC 时钟为 APB2/2，即频率为 4MHz，详见表 1.4）
0x08	CIR	读/写	0x0000 0000	时钟中断寄存器（禁止所有中断）
0x0C	APB2RSTR	读/写	0x0000 0000	APB2 设备复位寄存器
0x10	APB1RSTR	读/写	0x0000 0000	APB1 设备复位寄存器
0x14	AHBENR	读/写	0x0000 0014	AHB 设备时钟使能寄存器（开启 Flash 接口和 SRAM 时钟）
0x18	APB2ENR	读/写	0x0000 0000	APB2 设备时钟使能寄存器（关闭所有 APB2 设备时钟，详见表 1.5）
0x1C	APB1ENR	读/写	0x0000 0000	APB1 设备时钟使能寄存器（关闭所有 APB1 设备时钟，详见表 1.6）
0x20	BDCR	读/写	0x0000 0000	备份域控制寄存器（详见表 1.7）
0x24	CSR	读/写	0x0C00 0000	控制状态寄存器（上电复位，NRST 引脚复位，详见表 1.8）

复位和时钟控制（RCC）寄存器结构体在 stm32f10x_map.h（V2.0.1）中定义如下：

```
typedef struct
{
  vu32 CR;                    // 时钟控制寄存器
  vu32 CFGR;                  // 时钟配置寄存器
  vu32 CIR;                   // 时钟中断寄存器
  vu32 APB2RSTR;              // APB2 设备复位寄存器
  vu32 APB1RSTR;              // APB1 设备复位寄存器
  vu32 AHBENR;                // AHB 设备时钟使能寄存器
  vu32 APB2ENR;               // APB2 设备时钟使能寄存器
  vu32 APB1ENR;               // APB1 设备时钟使能寄存器
  vu32 BDCR;                  // 备份域控制寄存器
  vu32 CSR;                   // 控制状态寄存器
} RCC_TypeDef;
```

1.3.1　时钟控制

时钟控制主要包括 HSI 使能、HSE 使能和 PLL 使能等，时钟控制寄存器（CR）如表 1.3 所示（保留位未列出）。

表 1.3　时钟控制寄存器（CR）

位	名　称	类　型	复 位 值	说　明
0	HSION	读/写	1	高速内部时钟使能：0—关闭时钟，1—开启时钟
1	HSIRDY	读	1	高速内部时钟就绪：0—时钟未就绪，1—时钟就绪
7:3	HSITRIM[4:0]	读/写	10000	高速内部时钟调整
15:8	HSICAL[7:0]	读	XXXXXXXX	高速内部时钟校准
16	HSEON	读/写	0	高速外部时钟使能：0—关闭时钟，1—开启时钟
17	HSERDY	读	0	高速外部时钟就绪：0—时钟未就绪，1—时钟就绪
18	HSEBYP	读/写	0	高速外部时钟旁路：0—时钟未旁路，1—时钟旁路
19	CSSON	读/写	0	时钟安全系统使能：0—关闭检测，1—开启检测
24	PLLON	读/写	0	PLL 使能：0—关闭 PLL，1—开启 PLL
25	PLLRDY	读	0	PLL 就绪：0—PLL 未就绪，1—PLL 就绪

常用与时钟控制有关的 RCC 库函数在 stm32f10x_rcc.h（V2.0.1）中声明如下：

```
void RCC_HSEConfig(u32 RCC_HSE);
ErrorStatus RCC_WaitForHSEStartUp(void);
void RCC_HSICmd(FunctionalState NewState);
void RCC_PLLCmd(FunctionalState NewState);
```

1）配置 HSE

```
void RCC_HSEConfig(u32 RCC_HSE);
```

参数说明：

★ RCC_HSE：HSE 配置，在 stm32f10x_rcc.h 中定义如下：

```
#define RCC_HSE_ON                    ((u32)0x00010000)   // 开启 HSE
#define RCC_HSE_Bypass                ((u32)0x00040000)   // 旁路 HSE
```

RCC_HSEConfig()函数的核心语句是：

```
RCC->CR &= CR_HSEON_Reset;
RCC->CR &= CR_HSEBYP_Reset;
RCC->CR |= CR_HSEON_Set;
RCC->CR |= CR_HSEBYP_Set | CR_HSEON_Set;
```

CR_HSEON_Reset、CR_HSEBYP_Reset、CR_HSEON_Set 和 CR_HSEBYP_Set 在 stm32f10x_rcc.c 中定义如下：

```
#define CR_HSEBYP_Reset               ((u32)0xFFFBFFFF)   // 旁路 HSE 复位
#define CR_HSEBYP_Set                 ((u32)0x00040000)   // 旁路 HSE 置位
#define CR_HSEON_Reset                ((u32)0xFFFEFFFF)   // 开启 HSE 复位
#define CR_HSEON_Set                  ((u32)0x00010000)   // 开启 HSE 置位
```

2）等待 HSE 启动

```
ErrorStatus RCC_WaitForHSEStartUp(void);
```

返回值：SUCCESS—HSE 就绪，ERROR—HSE 未就绪

RCC_WaitForHSEStartUp()函数的核心语句是：

```
HSEStatus = RCC_GetFlagStatus(RCC_FLAG_HSERDY);
```

RCC_GetFlagStatus()函数的说明参加 1.3.6（2）。

3）使能 HSI

```
void RCC_HSICmd(FunctionalState NewState);
```

参数说明：

★ NewState：HSI 新状态，ENABLE（1）—允许，DISABLE（0）—禁止

4）使能 PLL

```
void RCC_PLLCmd(FunctionalState NewState);
```

参数说明：

★ NewState：HSI 新状态，ENABLE（1）—允许，DISABLE（0）—禁止

1.3.2 时钟配置

时钟配置主要包括系统时钟切换、AHB 预分频、APB1 预分频、APB2 预分频、ADC 预分频、PLL 倍频和时钟输出选择等，时钟配置寄存器（CFGR）如表 1.4 所示。

表 1.4 时钟配置寄存器（CFGR）

位	名　称	类　型	复 位 值	说　明
1:0	SW[1:0]	读/写	00	系统时钟切换：00—HSI，01—HSE，10—PLLCLK，11—不可用
3:2	SWS[1:0]	读	00	系统时钟切换状态：00—HSI，01—HSE，10—PLLCLK，11—不可用
7:4	HPRE[3:0]	读/写	0000	AHB 预分频：0XXX—SYSCLK，1000—SYSCLK/2，1001—SYSCLK/4，1010—SYSCLK/8，1011—SYSCLK/16，1100—SYSCLK/64，1101—SYSCLK/128，1110—SYSCLK/256，1111—SYSCLK/512
10:8	PPRE1[2:0]	读/写	000	APB1 预分频：0XX—HCLK，100—HCLK/2，101—HCLK/4，110—HCLK/8，111—HCLK/16
13:11	PPRE2[2:0]	读/写	000	APB2 预分频：0XX—HCLK，100—HCLK/2，101—HCLK/4，110—HCLK/8，111—HCLK/16
15:14	ADCPRE[1:0]	读/写	00	ADC 预分频：00—PCLK2/2，01—PCLK2/4，10—PCLK2/6，11—PCLK2/8
16	PLLSRC	读/写	0	PLL 输入时钟源：0—HSI/2，1—HSE 或 HSE/2
17	PLLXTPRE	读/写	0	PLL 外部输入分频：0—HSE，1—HSE/2
21:18	PLLMUL[3:0]	读/写	0000	PLL 倍频：0000~1110—2~16 倍频，1111—16 倍频
22	USBPRE	读/写	0	USB 预分频：0—PLLCLK/1.5，1—PLLCLK
26:24	MCO[2:0]	读/写	000	时钟输出选择：0XX—不输出，100—SYSCLK，101—HSI，110—HSE，111—PLLCLK/2

常用与时钟配置有关的 RCC 库函数在 stm32f10x_rcc.h（V2.0.1）中声明如下：

```
void RCC_PLLConfig(u32 RCC_PLLSource, u32 RCC_PLLMul);
void RCC_SYSCLKConfig(u32 RCC_SYSCLKSource);
u8 RCC_GetSYSCLKSource(void);
void RCC_HCLKConfig(u32 RCC_SYSCLK);
```

```
void RCC_PCLK1Config(u32 RCC_HCLK);
void RCC_PCLK2Config(u32 RCC_HCLK);
void RCC_ADCCLKConfig(u32 RCC_PCLK2);
void RCC_GetClocksFreq(RCC_ClocksTypeDef* RCC_Clocks);
void RCC_MCOConfig(u8 RCC_MCO);
```

1）配置 PLL

```
void RCC_PLLConfig(u32 RCC_PLLSource, u32 RCC_PLLMul);
```

参数说明：

★ RCC_PLLSource：PLL 时钟源，在 stm32f10x_rcc.h 中定义如下：

```
#define RCC_PLLSource_HSI_Div2      ((u32)0x00000000)   // HSI 分频 2
#define RCC_PLLSource_HSE_Div1      ((u32)0x00010000)   // HSE 分频 1
#define RCC_PLLSource_HSE_Div2      ((u32)0x00030000)   // HSE 分频 2
```

★ RCC_PLLMul：PLL 倍频，在 stm32f10x_rcc.h 中定义如下：

```
#define RCC_PLLMul_2                ((u32)0x00000000)   // PLL 倍频 2
#define RCC_PLLMul_3                ((u32)0x00040000)   // PLL 倍频 3
#define RCC_PLLMul_4                ((u32)0x00080000)   // PLL 倍频 4
#define RCC_PLLMul_5                ((u32)0x000C0000)   // PLL 倍频 5
#define RCC_PLLMul_6                ((u32)0x00100000)   // PLL 倍频 6
#define RCC_PLLMul_7                ((u32)0x00140000)   // PLL 倍频 7
#define RCC_PLLMul_8                ((u32)0x00180000)   // PLL 倍频 8
#define RCC_PLLMul_9                ((u32)0x001C0000)   // PLL 倍频 9
#define RCC_PLLMul_10               ((u32)0x00200000)   // PLL 倍频 10
#define RCC_PLLMul_11               ((u32)0x00240000)   // PLL 倍频 11
#define RCC_PLLMul_12               ((u32)0x00280000)   // PLL 倍频 12
#define RCC_PLLMul_13               ((u32)0x002C0000)   // PLL 倍频 13
#define RCC_PLLMul_14               ((u32)0x00300000)   // PLL 倍频 14
#define RCC_PLLMul_15               ((u32)0x00340000)   // PLL 倍频 15
#define RCC_PLLMul_16               ((u32)0x00380000)   // PLL 倍频 16
```

RCC_PLLConfig()函数的核心语句是：

```
tmpreg |= RCC_PLLSource | RCC_PLLMul;
RCC->CFGR = tmpreg;
```

2）配置 SYSCLK

```
void RCC_SYSCLKConfig(u32 RCC_SYSCLKSource);
```

参数说明：

★ RCC_SYSCLKSource：SYSCLK 时钟源，在 stm32f10x_rcc.h 中定义如下：

```
#define RCC_SYSCLKSource_HSI      ((u32)0x00000000)      // SYSCLK=HSI
#define RCC_SYSCLKSource_HSE      ((u32)0x00000001)      // SYSCLK=HSE
#define RCC_SYSCLKSource_PLLCLK   ((u32)0x00000002)      // SYSCLK=PLLCLK
```

RCC_SYSCLKConfig()函数的核心语句是：

```
tmpreg |= RCC_SYSCLKSource;
RCC->CFGR = tmpreg;
```

3）获取 SYSCLK 时钟源

```
u8 RCC_GetSYSCLKSource(void);
```

返回值：SYSCLK 时钟源，0—HSI，4—HSE，8—PLLCLK

RCC_GetSYSCLKSource()函数的核心语句是：

```
return ((u8)(RCC->CFGR & CFGR_SWS_Mask));
```

CFGR_SWS_Mask 在 stm32f10x_rcc.c 中定义如下：

```
#define CFGR_SWS_Mask                    ((u32)0x0000000C)   // SYSCLK 屏蔽
```

4）配置 HCLK

```
void RCC_HCLKConfig(u32 RCC_SYSCLK);
```

参数说明：

★ RCC_SYSCLK：SYSCLK 分频，在 stm32f10x_rcc.h 中定义如下：

```
#define RCC_SYSCLK_Div1          ((u32)0x00000000)   // SYSCLK 分频 1
#define RCC_SYSCLK_Div2          ((u32)0x00000080)   // SYSCLK 分频 2
#define RCC_SYSCLK_Div4          ((u32)0x00000090)   // SYSCLK 分频 4
#define RCC_SYSCLK_Div8          ((u32)0x000000A0)   // SYSCLK 分频 8
#define RCC_SYSCLK_Div16         ((u32)0x000000B0)   // SYSCLK 分频 16
#define RCC_SYSCLK_Div64         ((u32)0x000000C0)   // SYSCLK 分频 64
#define RCC_SYSCLK_Div128        ((u32)0x000000D0)   // SYSCLK 分频 128
#define RCC_SYSCLK_Div256        ((u32)0x000000E0)   // SYSCLK 分频 256
#define RCC_SYSCLK_Div512        ((u32)0x000000F0)   // SYSCLK 分频 512
```

RCC_HCLKConfig()函数的核心语句是：

```
tmpreg |= RCC_SYSCLK;
RCC->CFGR = tmpreg;
```

5）配置 PCLK1

```
void RCC_PCLK1Config(u32 RCC_HCLK);
```

参数说明：

★ RCC_HCLK：HCLK 分频，在 stm32f10x_rcc.h 中定义如下：

```
#define RCC_HCLK_Div1            ((u32)0x00000000)   // HCLK 分频 1
#define RCC_HCLK_Div2            ((u32)0x00000400)   // HCLK 分频 2
#define RCC_HCLK_Div4            ((u32)0x00000500)   // HCLK 分频 4
#define RCC_HCLK_Div8            ((u32)0x00000600)   // HCLK 分频 8
#define RCC_HCLK_Div16           ((u32)0x00000700)   // HCLK 分频 16
```

RCC_PCLK1Config()函数的核心语句是：

```
tmpreg |= RCC_HCLK;
RCC->CFGR = tmpreg;
```

6）配置 PCLK2

```
void RCC_PCLK2Config(u32 RCC_HCLK);
```

参数说明：

★ RCC_HCLK：HCLK 分频，在 stm32f10x_rcc.h 中定义如下：

```
#define RCC_HCLK_Div1              ((u32)0x00000000)   // HCLK 分频 1
#define RCC_HCLK_Div2              ((u32)0x00000400)   // HCLK 分频 2
#define RCC_HCLK_Div4              ((u32)0x00000500)   // HCLK 分频 4
#define RCC_HCLK_Div8              ((u32)0x00000600)   // HCLK 分频 8
#define RCC_HCLK_Div16             ((u32)0x00000700)   // HCLK 分频 16
```

RCC_PCLK2Config()函数的核心语句是：

```
tmpreg |= RCC_HCLK<< 3;
RCC->CFGR = tmpreg;
```

7）配置 ADCCLK

```
void RCC_ADCCLKConfig(u32 RCC_PCLK2);
```

参数说明：

★ RCC_PCLK2：PCLK2 分频，在 stm32f10x_rcc.h 中定义如下：

```
#define RCC_PCLK2_Div2             ((u32)0x00000000)   // PCLK2 分频 2
#define RCC_PCLK2_Div4             ((u32)0x00004000)   // PCLK2 分频 4
#define RCC_PCLK2_Div6             ((u32)0x00008000)   // PCLK2 分频 6
#define RCC_PCLK2_Div8             ((u32)0x0000C000)   // PCLK2 分频 8
```

RCC_ADCCLKConfig()函数的核心语句是：

```
tmpreg |= RCC_PCLK2;
RCC->CFGR = tmpreg;
```

8）获取时钟频率

```
void RCC_GetClocksFreq(RCC_ClocksTypeDef* RCC_Clocks);
```

参数说明：

★ RCC_Clocks：时钟结构体指针，时钟结构体在 stm32f10x_rcc.h 中定义如下：

```
typedef struct
{
  u32 SYSCLK_Frequency;          // SYSCLK 频率
  u32 HCLK_Frequency;          // HCLK 频率
  u32 PCLK1_Frequency;          // PCLK1 频率
  u32 PCLK2_Frequency;          // PCLK2 频率
  u32 ADCCLK_Frequency;          // ADCCLK 频率
} RCC_ClocksTypeDef;
```

9）配置时钟输出

```
void RCC_MCOConfig(u8 RCC_MCO);
```