第 3 章　STM32 嵌入式处理器

STM32 系列处理器是一款基于专为要求高性能、低成本、低功耗的嵌入式应用专门设计的 ARM Cortex-M3 内核。按性能分成两个不同的系列：STM32F103“基本型”系列和 STM32F101“增强型”系列。

该 ARM 处理器片上资源丰富，是专门为汽车电子、手持设备等应用提供的高性价比解决方案。

本章首先介绍开发的基于 ARM Cortex-M3 内核 STM32F103VB 芯片的评估板，然后对 STM32F103XX 系列处理器进行简单介绍，并针对 STM32F103XX 系列芯片上的基本功能模块进行详细介绍。通过本章内容的学习，读者可以初步了解并掌握 ARM 嵌入式系统的组成以及片上硬件资源的基本知识。

【本章重点】

- STM32F103XX 系列芯片的结构。
- STM32F103XX 系列芯片中的系统控制模块。
- STM32F103XX 系列芯片中的中断。

【本章难点】

- STM32F103XX 系列芯片中的系统控制模块。
- STM32F103XX 系列芯片中的中断。

3.1　STM32F103XX 系统简介

STM32 是一款 ARM 微控制器产品系列的总称。目前这个系列处理器中已经包含了很多子系列，分别介绍如下：

- STM32 小容量产品，即闪存存储器容量在 16～32KB 的 STM32 微处理器。
- STM32 中容量产品，即闪存存储器容量在 64～128KB 的 STM32 微处理器。
- STM32 大容量产品，即闪存存储器容量在 256～512KB 的 STM32 微处理器。
- STM32 互联型产品。

此外，从功能上划分，该系列处理器又可以分为：

(1) STM32F101XX 系列单片机。

(2) STM32F102XX 系列单片机。

(3) STM32F103XX 系列单片机。

3.1.1　STM32F103XX 系列处理器芯片

STM32F103XX 系列单片机是基于高性能 32 位 RISC 的 ARM Cortex-M3 内核，最高工作频率为 72MHz。片上集成了高速存储器和通过 APB 总线连接的众多外设和输入输出接口，其中片上存储器 Flash 最多可达 512KB，SRAM 最多可达 64KB。此外，所有的设备都提供了标准的通信接口，如 2 个 I^2C 接口、3 个 SPI 接口和 5 个 USART 接口。除此之外，STM32F103XX 系列处理器还集成了 2 个 12b 的 ADC、1 个 12b 的双通道 DAC、11 个 16b 的计时器。由此可见，使用 STM32F103XX 系列处理器就可以比较容易地实现较高的指令吞吐量和丰富的系统资源，具体介绍如下。

1. 系统内核

(1) 在 STM32F103XX 系列处理器中，使用了 ARM 32b 的 Cortex-M3 内核，最高工作频率可达 72MHz，代码吞吐量高达 1.25DMIPS/MHz。

(2) 支持单周期指令乘法和硬件除法。

2. 存储器

(1) 片上集成了 32～512KB 的 Flash 存储器。

(2) 片上集成了 6～64KB 的 SRAM 存储器。

3. 系统时钟、复位和电源管理

(1) 2.0～3.6V 的电源供电以及数字输入输出端口的驱动电压。

(2) 系统上电复位(Power On Reset，POR)、系统掉电复位(Power Down Reset，PDR)以及可变成的电压探测器 PVD。

(3) 支持 4～16MHz 的晶振。

(4) 处理器内部集成了出厂前调校的 8MHz 内部振荡电路。

(5) 支持用于 CPU 时钟的锁相环(Phase Locked Loop，PLL)。

(6) 带校准用于实时时钟(Real-Time Clock，RTC)的 32kHz 的晶振。

4. 低功耗

(1) 处理器提供 3 种低功耗模式：休眠模式、停机模式和待机模式。

(2) 支持用于 RTC 以及备份寄存器供电的 VBAT。

5. 2 个 12b 的 μs 级 AD 转换器

(1) AD 转换器测量范围：0～3.6V，16 路采集通道。

(2) 双通道采样并支持采样数据保存。

(3) 处理器片上集成一个温度传感器。

6. 直接内存存取(Direct Memory Access，DMA)

(1) 支持 12 通道 DAM 控制器。

(2) 支持对如下外设的 DMA 操作：定时器 Timer、模拟数字转换器 ADC、数字模拟转换器 DAC、I^2C 通信接口、SPI 通信接口以及 USART 通信接口。

7. 最多支持 112 个快速输入输出端口

(1) 根据不同的处理器型号,STM32 分别支持 26、37、51、80 以及 112 引脚的数字输入输出端口,且所有端口均可被映射为 16 个外部中断向量。

(2) 除了模拟信号输入引脚外,其他所有引脚均可容忍 5V 电压的输入信号。

8. 调试模式

(1) 处理器支持串行调试接口 SWD。

(2) 处理器支持 JTAG 调制接口。

9. 支持 11 个定时器

(1) 支持 4 个 16b 的定时器,且每个定时器具有 4 个 IC/OC/PWM 或者脉冲计数器。

(2) 支持 2 个 16b 的 6 通道高级控制定时器,最多 6 个通道可用于 PWM 信号输出。

(3) 支持 2 个看门狗定时器(独立看门狗定时器与串口看门狗定时器)。

(4) 支持 SysTick 定时器,即 24b 的倒数计数器。

(5) 支持 2 个 16b 的基本定时器,用于驱动数字模拟转换器 DAC。

10. 支持 13 个通信接口

(1) 支持 2 个 I^2C 接口,分别为 SMBμs 和 PMBμs。

(2) 支持 5 个 USART 接口。

(3) 支持 3 个速率为 18Mb/s 的 SPI 接口,与 I^2C 接口复用。

(4) 支持 2 个 CAN 2.0 接口。

(5) 支持 1 个 USB 2.0 接口。

(6) 支持 1 个 SDIO 接口。

3.1.2 STM32F103XX 器件信息

在 STM32 单片机中,根据微处理器中片上硬件资源的不同可以将其分为不同的系列。这也就需要用户在进行 ARM 嵌入式系统设计之前,根据系统对硬件资源的实际需求进行 STM32 处理器的选型操作。在表 3.1 中,选取了 STM32 单片机中 STM32F103XX 系列中常见的处理器型号及其资源列表,用户可以对照该表进行 ARM 处理器的选型操作。

从表 3.1 中可以看出,即使是同一个 STM32 处理器系列,也存在各种不同的具体型号。这些不同型号的处理器主要是在片上集成的硬件资源上有比较大的差别。

用户在进行 ARM 嵌入式开发的过程中,必须根据系统的实际需求,参考表 3.1 中各个类型处理器中所集成的硬件资源,选择不同型号的 ARM 处理器。只有充分考虑到系统设计的各个要求,才能选择正确型号的单片机,既能够满足 ARM 系统对硬件资源的需要,也不浪费资源,降低系统的设计成本。

表 3.1 STM32F103XX 器件选型表

引脚	型 号	ROM	RAM	16b 通用	16b 高级	16b 基本	看门狗	RTC	SPI	I^2C	USART	USB/CAN	I^2S	SDIO	ADC	DAC	IO	封 装
36	STM32F103T4	16	6	2(8/8/8)	1(4/4/6)		2	1	1	1	2	1/1			2		26	QFN36
	STM32F103T6	32	10	2(8/8/8)	1(4/4/6)		2	1	1	1	2	1/1			2		26	QFN36
	STM32F103T8	64	20	3(12/12/12)	1(4/4/6)		2	1	1	1	2	1/1			2		26	QFN36
48	STM32F103C4	16	6	2(8/8/8)	1(4/4/6)		2	1	1	1	2	1/1			2		37	LQFP48
	STM32F103C6	32	10	2(8/8/8)	1(4/4/6)		2	1	1	1	2	1/1			2		37	LQFP48
	STM32F103C8	64	20	3(12/12/12)	1(4/4/6)		2	1	1	1	2	1/1			2		37	LQFP48
	STM32F103CB	128	20	3(12/12/12)	1(4/4/6)		2	1	1	1	2	1/1			2		37	LQFP48
64	STM32F103R4	16	6	2(8/8/8)	1(4/4/6)		2	1	1	1	2	1/1			2		51	LQFP64
	STM32F103R6	32	10	2(8/8/8)	1(4/4/6)		2	1	1	1	2	1/1			2		51	LQFP64
	STM32F103R8	64	20	3(12/12/12)	1(4/4/6)		2	1	2	2	3	1/1			2		51	LQFP64
	STM32F103RB	128	20	3(12/12/12)	1(4/4/6)		2	1	2	2	3	1/1			2		51	LQFP64
	STM32F103RC	256	48	4(16/16/16)	2(8/8/12)	2	2	1	3	2	5	1/1	2	1	3		51	LQFP64
	STM32F103RD	384	64	4(16/16/16)	2(8/8/12)	2	2	1	3	2	5	1/1	2	1	3		51	LQFP64
	STM32F103RE	512	64	4(16/16/16)	2(8/8/12)	2	2	1	3	2	5	1/1	2	1	3		51	LQFP64
100	STM32F103V8	64	20	3(12/12/12)	1(4/4/6)		2	1	2	2	3	1/1			2		80	LQFP100
	STM32F103VB	128	20	3(12/12/12)	1(4/4/6)		2	1	2	2	3	1/1			2		80	LQFP100
	STM32F103VC	256	48	4(16/16/16)	2(8/8/12)	2	2	1	3	2	5	1/1	2	1	3	1	80	LQFP100
	STM32F103VD	384	64	4(16/16/16)	2(8/8/12)	2	2	1	3	2	5	1/1	2	1	3	1	80	LQFP100
	STM32F103VE	512	64	4(16/16/16)	2(8/8/12)	2	2	1	3	2	5	1/1	2	1	3	1	80	LQFP100
144	STM32F103ZC	256	48	4(16/16/16)	2(8/8/12)	2	2	1	3	2	5	1/1	2	1	3	1	80	LQFP144
	STM32F103ZD	384	64	4(16/16/16)	2(8/8/12)	2	2	1	3	2	5	1/1	2	1	3	1	80	LQFP144
	STM32F103ZC	512	64	4(16/16/16)	2(8/8/12)	2	2	1	3	2	5	1/1	2	1	3	1	80	LQFP144

3.2 STM32F103XX 引脚信息

对于不同型号的 STM32F103XX 处理器，虽然可能在系统硬件资源上类似，但在引脚以及封装上并不是完全一致的。

以处理器的封装为例，对于不同的 STM32 处理器，即使硬件资源一样，也可以存在不同类型的封装，典型的有 LQFP48、LQFP64、LQFP100、LQFP144、QFN36、BGA100、BGA144 等，如图 3.1 所示。较小的封装和极低的功耗使 STM32F103XX 系列单片机可以理想地应用于小型 ARM 嵌入式系统中，如汽车电子、手持娱乐媒体等。

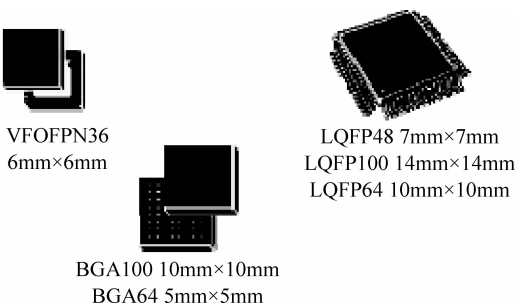

图 3.1 STM32 处理器的封装

当然，除了处理器封装的差异外，同一款型号的处理器芯片也可能存在不同的引脚数目，上述封装中的 LQFP48、LQFP64、LQFP100、LQFP144，其封装符号中最后的数字，就是当前处理器芯片的引脚数目。用户可以通过表 3.2 查看不同封装的处理器芯片中各个引脚的定义。

表 3.2 STM32F103XX 系列处理器引脚功能定义

芯片引脚						引脚名称	引脚类型	复位后功能	可选(复用)功能	
LFBGA100	LQFP48	TGBGA64	LQFP64	LQFP100	VFQFPN36				默认功能	映射功能
A3				1		PE2	IO	PE2	TRACECK	
B3				2		PE3	IO	PE3	TRACED0	
C3				3		PE4	IO	PE4	TRACED1	
D3				4		PE5	IO	PE5	TRACED2	
E3				5		PE6	IO	PE6	TRACED3	
B2	1	B2	1	6		V_{BAT}	S	V_{BAT}		
A2	2	A2	2	7		PC13/TAMPER-RTC	IO	PC13	TAMPER_RTC	
A1	3	A1	3	8		PC14/OSC32_IN	IO	PC14	OSC32_IN	
B1	4	B1	4	9		PC15/OSC32_OUT	IO	PC15	OSC32_OUT	

续表

芯片引脚						引脚名称	引脚类型	复位后功能	可选(复用)功能	
LFBGA100	LQFP48	TGBGA64	LQFP64	LQFP100	VFQFPN36				默认功能	映射功能
C2				10		Vss_5	S	Vss_5		
D2				11		V_{DD}_5	S	V_{DD}_5		
C1	5	C1	5	12	2	OSC_IN	I	OSC_IN		
D1	6	D1	6	13	3	OSC_OUT	O	OSC_OUT		
E1	7	E1	7	14	4	NRST	IO	NRST		
F1		E3	8	15		PC0	IO	PC0	ADC12_IN10	
F2		E2	9	16		PC1	IO	PC1	ADC12_IN11	
E2		F2	10	17		PC2	IO	PC2	ADC12_IN12	
F3			11	18		PC3	IO	PC3	ADC12_IN13	
G1	8	F1	12	19	5	Vss_A	S	Vss_A		
H1				20		V_{REF-}	S	V_{REF-}		
J1		G1		21		V_{REF+}	S	V_{REF+}		
K1	9	H1	13	22	6	V_{DDA}	S	V_{DDA}		
G2	10	G2	14	23	7	PA0-WKUP	IO	PA0	WKUP/USART2_CTS/ADC12_IN0/TIM2_CH1_ETR	
H2	11	H2	15	24	8	PA1	IO	PA1	USART2_RTS/ADC12_IN1/TIM2_CH2	
J2	12	F3	16	25	9	PA2	IO	PA2	USART2_TX/ADC12_IN2/TIM2_CH3	
K2	13	G3	17	26	10	PA3	IO	PA3	USART2_RX/ADC12_IN3/TIM2_CH4	
E4		C2	18	27		Vss_4	S	Vss_4		
F4		D2	19	28		V_{DD}_4	S	V_{DD}_4		
G3	14	H3	20	29	11	PA4	IO	PA4	SPI1_NSS/USART2_CK/ADC12_IN4	
H3	15	F4	21	30	12	PA5	IO	PA5	SPI1_SCK/ADC12_IN5	
J3	16	G4	22	31	13	PA6	IO	PA6	SPI1_MISO/ADC12_IN6/TIM3_CH1	TIM1_BKIN
K3	17	H4	23	32	14	PA7	IO	PA7	SPI1_MOSI/ADC12_IN7/TIM3_CH2	TIM1_CH1N
G4		H5	24	33		PC4	IO	PC4	ADC12_IN14	
H4		H6	25	34		PC5	IO	PC5	ADC12_IN15	
J4	18	F5	26	35	15	PB0	IO	PB0	ADC12_IN8/TIM3_CH3	TIM1_CH2N
K4	19	G5	27	36	16	PB1	IO	PB1	ADC12_IN9/TIM3_CH4	TIM1_CH3N
G5	20	G6	28	37	17	PB2	IO		PB2/BOOT1	
H5				38		PE7	IO	PE7		TIM1_ETR

续表

芯片引脚						引脚名称	引脚类型	复位后功能	可选(复用)功能	
LFBGA100	LQFP48	TGBGA64	LQFP64	LQFP100	VFQFPN36				默认功能	映射功能
J5				39		PE8	IO	PE8		TIM1_CH1N
K5				40		PE9	IO	PE9		TIM1_CH1
G6				41		PE10	IO	PE10		TIM1_CH2N
H6				42		PE11	IO	PE11		TIM1_CH2
J6				43		PE12	IO	PE12		TIM1_CH3N
K6				44		PE13	IO	PE13		TIM1_CH3
G7				45		PE14	IO	PE14		TIM1_CH4
H7				46		PE15	IO	PE15		TIM1_BKIN
J7	21	G7	29	47		PB10	IO	PB10	I2C2_SCL/USART3_TX	TIM2_CH3
K7	22	H7	30	48		PB11	IO	PB11	I2C2_SDA/USART3_RX	TIM2_CH4
E7	23	D6	31	49	18	Vss_1	S	Vss_1		
F7	24	E6	32	50	19	V_{DD}_1	S	V_{DD}_1		
K8	25	H8	33	51		PB12	IO	PB12	SPI2_NSS/I2C2_SMBAI/USART3_CK/TIM1_BKIN	
J8	26	G8	34	52		PB13	IO	PB13	SPI2_SCK/USART3_CTS/TIM1_CHIN	
H8	27	F8	35	53		PB14	IO	PB14	SPI2_MISO/USART3_RTS/TIM1_CH2N	
G8	28	F7	36	54		PB15	IO	PB15	SPI2_MOSI/TIM1_CH3N	
K9				55		PD8	IO	PD8		USART3_TX
J9				56		PD9	IO	PD9		USART3_RX
H9				57		PD10	IO	PD10		USART3_CK
G9				58		PD11	IO	PD11		USART3_CTS
K10				59		PD12	IO	PD12		TIM1_CH1/USART3_RTS
J10				60		PD13	IO	PD13		TIM4_CH2
H10				61		PD14	IO	PD14		TIM4_CH3
G10				62		PD15	IO	PD15		TIM4_CH4

续表

芯片引脚						引脚名称	引脚类型	复位后功能	可选(复用)功能	
LFBGA100	LQFP48	TGBGA64	LQFP64	LQFP100	VFQFPN36				默认功能	映射功能
F10		F6	37	63		PC6	IO	PC6		TIM3_CH1
E10		E7	38	64		PC7	IO	PC7		TIM3_CH2
F9		E8	39	65		PC8	IO	PC8		TIM3_CH3
E9		D8	40	66		PC9	IO	PC9		TIM3_CH4
D9	29	D7	41	67	20	PA8	IO	PA8	USART1_CK/TIM1_CH1/MCO	
C9	30	C7	42	68	21	PA9	IO	PA9	USART1_TX/TIM1_CH2	
D10	31	C6	43	69	22	PA10	IO	PA10	USART1_RX/TIM1_CH3	
C10	32	C8	44	70	23	PA11	IO	PA11	USART1_CTS/USBDM/CAN_RX/TIM1_CH4	
B10	33	B8	45	71	24	PA12	IO	PA12	USART1_RTS/USBDP/CAN_TX/TIM1_ETR	
A10	34	A8	46	72	25	PA13	IO	JTMS/SWDIO		PA13
F8				73		Not Connect				
E6	35	D5	47	74	26	Vss_2	S	Vss_2		
F6	36	E5	48	75	27	V_{DD}_2	S	V_{DD}_2		
A9	37	A7	49	76	28	PA14	IO	JTCK/SWCLK		PA14
A8	38	A6	50	77	29	PA15	IO	JTDI		TIM2_CH1_ETR/PA15/SPI1_NSS
B9		B7	51	78		PC10	IO	PC10		USART3_TX
B8		B6	52	79		PC11	IO	PC11		USART3_RX
C8		C5	53	80		PC12	IO	PC12		USART3_CK
D8	5	C1	5	81	2	PD0	IO	OSC_IN		CAN_RX
E8	6	D1	6	82	3	PD1	IO	OSC_OUT		CAN_TX
B7		B5	54	83		PD2	IO	PD2	TIM3_ETR	
C7				84		PD3	IO	PD3		USART2_CTS
D7				85		PD4	IO	PD4		USART2_RTS
B6				86		PD5	IO	PD5		USART2_TX
C6				87		PD6	IO	PD6		USART2_RX
D6				88		PD7	IO	PD7		USART2_CK
A7	39	A5	55	89	30	PB3	IO	JTDO		PB3/TRACESWO/TIM2_CH2/SPI1_SCK

续表

芯片引脚						引脚名称	引脚类型	复位后功能	可选(复用)功能	
LFBGA100	LQFP48	TGBGA64	LQFP64	LQFP100	VFQFPN36				默认功能	映射功能
A6	40	A4	56	90	31	PB4	IO	NJTRST		PB4/TIM3_CH1/SPI1_MISO
C5	41	C4	57	91	32	PB5	IO	PB5	I2C1_SMBAI	TIM3_CH2/SPI1_MOSI
B5	42	D3	58	92	33	PB6	IO	PB6	I2C1_SCL/TIM4_CH1	USART1/TX
A5	43	C3	59	93	34	PB7	IO	PB7	I2C1_SDA/TIM4_CH2	USART1/RX
D5	44	B4	60	94	35	BOOT0	I	BOOT0		
B4	45	B3	61	95		PB8	IO	PB8	TIM4_CH3	I2C1_SCL/CAN_RX
A4	46	A3	62	96		PB9	IO	PB9	TIM4_CH4	I2C1_SDA/CAN_TX
D4				97		PE0	IO	PE0	TIM4_ETR	
C4				98		PE1	IO	PE1		
E5	47	D4	63	99	36	Vss_3	S	Vss_3		
F5	48	E4	64	100	1	V_{DD}_3	S	V_{DD}_3		

在 STM32F103XX 系列芯片中,绝大部分的引脚都具有 1 个以上的功能,如表 3.2 所示。在实际的工程应用中,用户需要将这些具有复用功能的引脚配置为用户所需要的功能。例如,同样作为数字输入输出口,用户可以将引脚配置为模拟信号输入、数字信号输入、数字信号输出等模式。

一般而言,STM32 处理器中的引脚绝大部分都可以容忍 5V 电压的上限,但作为模拟信号输入的引脚则最高不得超过 3.3V 电压。因此,在进行 ADC 操作的电路设计中,需要特别留意。

3.3 STM32F103XX 的内部结构

在 STM32 系列 ARM 处理器中,包含一个支持 JTAG 仿真的 Cortex-M3 处理器,与片内的存储器控制器接口的局部总线、与中断控制器接口的高性能总线(Advanced High Performance Bus,AHB)和连接片内外设功能的 VLSI 外设总线(VLSI Peripheral Bus,VPB),且 AHB 与 VPB 通过桥相连。

STM32F103XX 芯片中其他的外设功能,除了中断控制器 DMA 以外,都连接到 VPB 总线上。

3.3.1　STM32F103XX 芯片总体结构

通常情况下，STM32F103XX 系列处理器的系统主要包括：

（1）4 个驱动单元，分别为 Cortex-M3 内核指令总线 I-bus、数据总线 D-bus 以及系统总线 S-bus；除此之外，还包含一个通用 DMA，即 GP-DMA。

（2）3 个被动单元，分别为内部 SRAM、内部闪存存储器以及 AHB 到 APB 桥。该桥主要用来连接所有的 APB 设备。

STM32F103XX 系列处理器的总体结构如图 3.2 所示。内部总线和两条 APB 总线将片上系统和外部设备资源紧密地连接起来，其中内部总线是主系统总线，连接 CPU、存储器和系统时钟信号灯。APB1 总线连接高速外设，APB2 总线连接系统外设和中断控制。

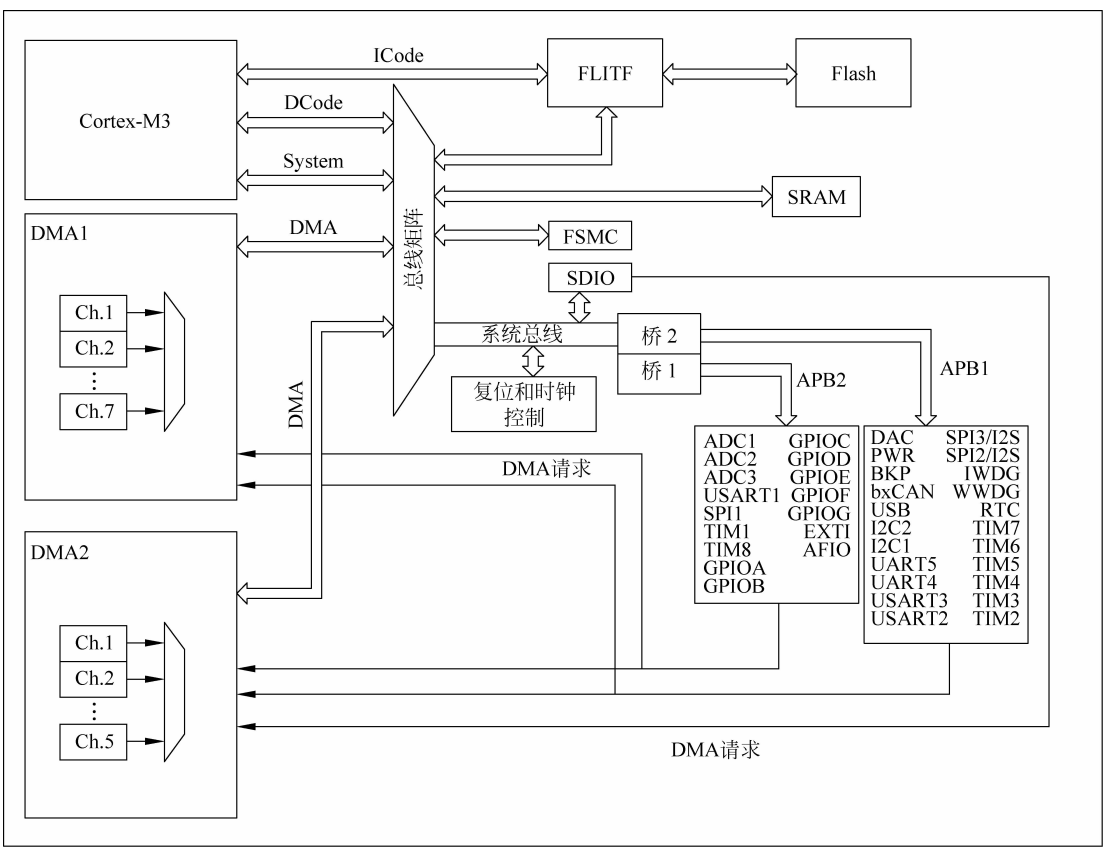

图 3.2　STM32F103XX 系列处理器的总体结构框图

在 STM32F103XX 系列处理器中，通用数字输入输出 IO 端口最多包括 PA、PB、PC、PD、PE、PF 和 PG 这 7 个 16b 的端口。其他的外设接口引脚都是通过与数字 IO 端口的引脚功能复用实现的。在表 3.2 中的 AF 即表示功能复用引脚。

3.3.2　STM32F103XX 片上 FLASH 程序存储器

在 STM32F103XX 系列处理器上集成了 FLASH 存储器系统。该存储器可以作为程

序代码或者数据的存储。需要说明的是，对 FLASH 存储器的编程可以通过以下几种方式实现：

(1) 通过内置串行 JTAG 接口。

(2) 通过在线系统编程(In System Programming，ISP)，即 USART0 通信接口。

(3) 通过应用编程(In Application Programming，IAP)。

在 ARM 处理器中，如果用户使用应用编程的方式进行程序的下载和擦除时，可以在程序运行的同时对 FLASH 进行擦除或编程，这样就位数据存储和现场固件的升级带来了比较大的灵活性。

3.3.3 STM32F103XX 片内静态 RAM

不同型号的 STM32F103XX 系列处理器内都集成了不同大小的静态 RAM，可以用作程序代码和数据变量的存储。主要说明的是，SRAM 可以分别支持 8b、16b 和 32b 的数据访问。

以 SRAM 中的 1B 寻址的存储器为例，对存储器进行字和半字访问时将忽略地址对准，访问被选址的自然对准值。通常，对存储器进行字访问时忽略地址位 0 和 1，半字访问时则忽略地址位 0。因此，有效的读写操作要求半字数据访问的地址线 0 为 0，即 ADDR&0xFFFFFFFE，字数据访问的地址线 0 和地址线 1 都必须全设置为 0，即 ADDR&0xFFFFFFFC。

SRAM 控制器包含一个回写缓冲区，主要用于防止 CPU 在连续的写数据操作时发生异常。一般，写缓冲区总是保存接收到的最后 1B 的数据。该数据只有在特定的请求回写指令的条件下才可以重新写入 SRAM。

在 ARM 处理器发生复位的时候，实际的 SRAM 中的内容并不能真实反映最近的一次写数据操作，这是在复位后检查 SRAM 的时候必须要注意的。同理，通过对一个存储单元执行两次相同的写操作可以保证复位后数据的写入。或者，也可以通过在进入空闲或者掉电模式前进行虚写操作(dummy write)来保证最后的数据在复位后被真正写入 SRAM。

3.4 STM32F103XX 存储器映射

在 STM32 系列处理器中，与传统的 ARM 处理器的存储架构相比，有着明显的不同，具体如下：

(1) STM32 处理器中的存储器映射是预先定义好的，并且规定了不同位置上的存储器使用不同的总线。

(2) 在 STM32 处理器中，数据的存储可以通过“位带”(bit-band)的方式来实现，需要说明的是，位带操作仅适用于一些特殊的存储器区域中。

(3) 在 STM32 处理器中，存储器系统支持非对齐访问和互斥访问。

(4) 在 STM32 处理器中，存储器系统支持小端配置和大端配置。

3.4.1　存储系统中的大/小端配置

ARM 处理器中的大端模式和小端模式是字节寻址存储器存储的两种方式，采取何种方式是根据最低有效字节与相邻较高有效字节相比较，并通过判断是存储在较低地址还是最高地址的结果来区分。

在小端存储方式是将较低字节存放在较低地址，而大端方式则是将较低字节存放在较高的地址，具体如图 3.3 所示。

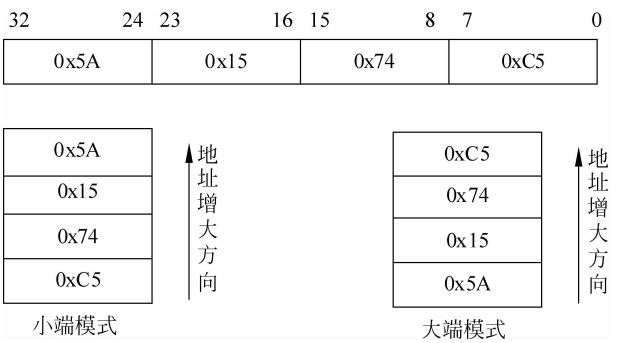

图 3.3　ARM 存储器中的大端模式和小端模式

通常情况下，STM32 系列处理器是在复位时确定使用大小端的工作模式，且在运行的过程中不允许对其进行修改。在绝大多数情况下，STM32 系列处理器都使用了小端模式以避免不必要的麻烦。在这里，也同样推荐用户在没有特殊情况下都使用小端模式。

在 STM32F103XX 处理器中，只有一个固定的存储器映射。因此可以方便地实现软件代码在各种不同型号的 STM32 系列处理器之间的移植。需要说明的是，STM32 系列处理器有关存储器空间的划分是比较粗略的，它允许芯片制造厂商按照各自的需要灵活分配存储器空间，以适应各种不同的应用场合。

3.4.2　系统存储器的映射

在 STM32F103XX 系列存储器中，内部地址空间的大小为 4GB。用户编写的程序代码可以在代码区、内部 SRAM 区以及外部扩展的 RAM 区中执行，具体的结构分布如图 3.4 所示。由于在 STM32 系统中，指令总线和数据总线是分开的，因此，建议用户将程序代码放置到代码区。这样就可以使得取址操作和数据访问操作各自使用自己的总线，而不会发生冲突。

在 STM32 处理器的内核中，内部 SRAM 的大小为 512MB，主要用于让芯片制造厂商能够连接到片上的 SRAM。这个区域中的数据可以通过系统总线进行访问。

需要注意的是，在内部 SRAM 区域的底部，存在一个 1MB 的空间——“位带区”，如图 3.4 所示。该“位带区”具有一个 32MB 且与之对应的“位带别名区”，可容纳 8MB 个 bit 型变量。显然，位带区中对应的是最低 1MB 的地址范围，而位带别名区中的每一个字(2B，16b)对应于位带区中的每一个位(bit)。

位带操作只适用于数据的访问，不能用于指令的取地址操作。用户可以通过位带功能，将多个位(bit)数据打包在一个单一的字(word)中，同时可以从位带别名区中对其直接进行

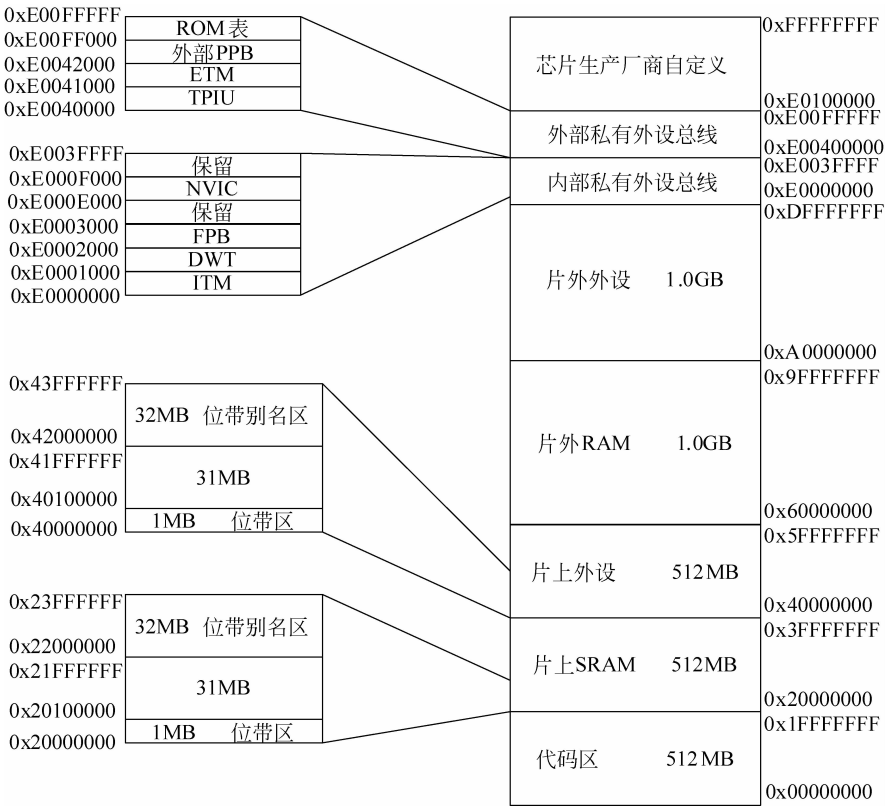

图 3.4 ARM 存储器映射

访问。

在 ARM 处理器地址空间中，处于片上 SRAM 上方的是片上外设。与内部 SRAM 类似的是，片上外设同样也具有一个位带区即 32MB 的位带别名区，这样可以提高访问外部设备的速度。需要注意的是，在外设区不可以执行任何用户的指令。

在 ARM 存储体系中，除了上述的存储空间外，还包含了 2 个 1GB 范围的地址空间，主要用于连接外部 RAM 和外部设备。需要说明的是，在这 2 个 1GB 范围的地址空间中，并不存在位带区。两者的差别在于外部 RAM 区允许执行指令，而外部设备区则不可以执行任何指令。

在最上层的部分存在一个 0.5GB 的私有地址范围，主要用于存放 ARM 内核。包括系统级组件、内部私有外设总线、外部私有外设总线，以及芯片制造商提供的系统外设。

在 STM32 系列处理中，私有外设总线有 2 条：

(1) AHB 私有外设总线，即只用于处理器内部的 AHB 外设，主要包含嵌套中断向量控制 NVIC、Flash 修补断点 FPB、数据观测和跟踪 DWT 和执行跟踪宏单元 ITM。

(2) APB 私有外设总线，既用于处理器内部的 APB 设备，也用于非 ARM 内核以外的设备。

在 STM32 系列处理器允许芯片生产厂商在 APH 私有总线上额外添加一些片上 APB 外设，并可以通过 APB 接口进行访问。

需要说明的是，上述对于系统存储器的映射只是一个粗线条的模板，ARM 芯片生产厂商会基于此映射提供更详细的地址空间配置说明，以说明片上外设的具体分布、系统 RAM 和 ROM 的容量以及位置信息。

3.4.3　系统存储器的访问属性

在 ARM 体系结构中，除了对系统中的存储空间进行粗略的映射划分外，还对系统存储器的访问定义了 4 种属性，分别为：

(1) 可否缓冲(bufferable)。

(2) 可否缓存(cacheable)。

(3) 可否执行(executable)。

(4) 可否共享(sharable)。

在 STM32 系列单片机中，用户可以通过内存保护单元 MUP 配置不同的存储区，并且覆盖默认的访问属性。需要注意的是，在 STM32 系列处理器中并没有配备缓存，更没有缓存控制器，但支持用户在处理器外围添加缓存。

通常情况下，如果处理器支持外部内存，则芯片生产厂商还会附加支持一个内存控制器。它可以根据存储器可否缓存的设置来管理片内和片外 RAM 的访问操作。

1. 系统代码区(0x00000000～0x1FFFFFFF)

在系统存储器的代码区中，用户可以执行相应的代码指令。该代码区中缓存的属性为 WT(Write Through，写通)，即不可缓存。此外，在前面章节中已经介绍过系统的代码区域还可以用于数据存储。用户可以通过数据总线接口实现对该代码区上数据的操作，且在该代码区上的写操作是可缓冲的。

在该区域中，缓存的属性为写通 WT，即写操作将“穿透”中途的缓存，直接到达最终的存储器目的地址中。因此在写通操作中，高速缓存(cache)并没有起到缓存作用，而只是让写操作的结果立即生效。

2. 系统 SRAM 区(0x20000000～0x3FFFFFFF)

在 ARM 处理器存储系统中，对 SRAM 区域的写操作是可缓冲的，并且可以选择 WB-WA(Write Back，Write Allocated)缓存属性。在该 SRAM 区域中，既可以直接运行程序代码，也允许把代码复制到系统的内存中执行。后者经常用在系统的固件升级过程中。

在 SRAM 区域中，存储系统的属性为写回 Write Back，即写入的数据先被暂时存放在系统缓存中，等到需要使用的时候再写入到最终目的地址中。这与系统高速缓存的功能类似，可以用于改善数据传送的效率，减少对主存储器的访问操作。

3. 系统片上外设区(0x40000000～0x5FFFFFFF)

片上外设区主要用于处理器片上外设，因此该区域的写操作是不可缓存的。也不可以在该区域执行指令。在 ARM 公司提供的数据手册中，对该属性定义为 XN，即 eXecute Never。

4. 系统外部 RAM 区(0x60000000～0x9FFFFFFF)

系统外部 RAM 区大致可以分为两个部分，外部 RAM 区前半段(0x60000000～0x7FFFFFFF)和外部 RAM 区后半段(0x60000000～0x9FFFFFFF)。

在前半段 RAM 区域中可以布设片上 RAM 或片外 RAM。用户可以在该外部 RAM

区域执行代码指令操作，并且对该区域的数据操作属性为可缓存，即 WB-WA。片上外设区主要用于处理器片上外设，因此该区域的写操作是不可缓存的。也不可以在该区域执行指令。在 ARM 公司提供的数据手册中，对该属性定义为 XN。

在后半段 RAM 区域中，基本的功能和属性与前半段 RAM 区域基本类似，唯一的不同在于区域的访问属性为不可缓存 WT。

5. 系统外部外设区(0xA0000000～0xDFFFFFFF)

系统外部外设区与外部 RAM 区的基本结构类似，也大致可以分为两个部分：外部外设区前半段(0xA0000000～0xBFFFFFFF)和外部 RAM 区后半段(0xC0000000～0xDFFFFFFF)。但 RAM 区后半段的功能和前半段的功能完全一致，只是习惯上将它分为两个部分而已。

系统外部外设区主要用于片外外设的寄存器，也可以用于多核系统中的内存共享。同样，该外设区也是一个不可执行代码的区域。

6. 系统区(0xE0000000～0xFFFFFFFF)

存储结构的系统区可以分为私有外设区域和芯片生产厂商指定功能的区域。在该区域中，用户也不可执行代码。由于系统区涉及系统运行的关键数据，所以对系统区的访问都是严格序列化的，既不可缓存，也不可缓冲。而对于芯片生产厂商指定功能的区域，则可以进行缓存和缓冲。

3.4.4 系统存储器的地址重映射

所谓存储器地址重映射的概念是：每一个存储器组在存储器映射中有一个“物理”位置，从本质上来说，它是一个地址范围，在该范围内用户可以写入程序代码，每一个存储器空间的容量都固定在同一个位置，这样就不需要将代码设计成在不同的范围内运行。在实际的工程设计过程中，为了让用户可以更好地利用存储空间，可以修改部分寄存器的定位映射，即改变默认的存储位置。

下面以 STM32 系列 ARM 处理器中的引脚重映射为例来说明有关地址重映射的使用。在 ARM 处理器中，每一个内置外设都具有若干个输入输出引脚。通常情况下，这些输出引脚的位置是固定不变的。为了能让用户更好地安排布线的走向以及引脚的功能，STM32 系列处理器提供了外设引脚重映射的概念，即一个外设的引脚除了具有默认的引脚编号外，还可以通过设置重映射寄存器的方式，将这个外设的引脚映射到其他的引脚位置，如图 3.5 所示。

引脚						引脚名称	类型	I/O 级	主要功能(复位后)	复用功能	
LFBGA144	LFBGA100	WLCSP64	LQFP64	LQFP100	LQFP144					默认	重新映射
J7	21	G7	29	47		PB10	I/O	FT	PB10	I2C2_SCL/USART3_TX	TIM2_CH3
K7	22	H7	30	48		PB11	I/O	FT	PB11	I2C2_SDA/USART3_RX	TIM2_CH4

图 3.5 STM32 处理器中的地址重映射

从图 3.5 中可以看出，串口 USART3_TX 默认的引脚为 PB10，USART3_RX 的默认输出引脚是 PB11，但经过系统地址重映射之后，可以将 USART3_TX 的引脚修改为 PD8，USART3_RX 的引脚修改为 PD9。

除此之外，STM32 系列处理器中绝大部分内置外设都具有地址重映射的功能，例如串口通信 USART、定时器 Timer、通信口 CAN、SPI 以及 I^2C 等。用户可以参考不同型号的 STM 数据手册。

在 STM32 系列处理器的地址重映射功能中，除了上述介绍的单个重映射外，还可以支持多个地址的重映射功能。这里同样以 USART3 串口为例，介绍在 STM32 处理器中多个地址的重映射，如图 3.6 所示。

复用功能	USART3_REMAP[1：0]＝“00”(不重复映射)	USART3_REMAP[1：0]＝“01”(部分重复映射)	USART3_REMAP[1：0]＝“11”(全部重复映射)
USART3_TX	PB10	RC10	PD8
USART3_RX	PB11	PC11	PD9
USART3_CK	PB12	PC12	PD10
USART3_CTS	PB13		PD11
USART3_RTS	PB14		PD12

图 3.6　STM32 处理器中 USART 多个地址重映射

从图 3.6 可以看出，串口 USART3_TX 默认的引脚为 PB10，USART3_RX 的默认输出引脚是 PB11。根据引脚寄存器的配置，可以将 USART3_TX 重映射到 PC10，USART3_RX 重映射到 PC11，还可以将 USART3_TX 重映射到 PD8，USART3_RX 重映射到 PD9。

3.4.5　系统存储中止的异常

如果用户访问存储器结构中的保留地址或者未分配区域的地址，则 STM32 处理器会产生一个中止异常。除此之外，用户对 AHB 或 VPB 外设地址执行任何指令操作，也会导致存储操作的中止异常。

在现有的 VPB 外设地址空间中，对未定义地址的访问不会产生数据中止异常。每一个外设内的地址译码被限制为外设内部需要判别的已定义的寄存器。需要注意的是，只有在用户对存储单元执行非法操作时，ARM 处理器才会将预取址中止标志与对应的非法指令一起保存到流水线并中止处理，即处理器在读取指令的时候仅仅设置中止标志，直到实际执行指令的时候才会干预并中止非法指令。

3.5　STM32F103XX 的系统控制模块

本节将介绍有关 STM32 系列处理器中的控制模块。通过了解这些控制模块的寄存器不仅仅对设计 ARM 嵌入式系统大有帮助，更能在调试的过程中加强对嵌入式硬件的了解。

在 ARM 处理器中，系统控制模块主要包括几个系统特性寄存器和控制寄存器。这些寄存器具有与特定外设器件无关的功能，主要包括晶体振荡器、外部中断输入、存储器映射

控制、锁相环 PPL、功率控制、复位电路、VPB 分频器、唤醒定时器等。这些系统控制模块的具体功能都取决于自身的寄存器。在表 3.3 中可以查看 STM32 系列处理器中系统控制模块功能相关的引脚配置。

表 3.3 系统控制模块的引脚

引脚名称	数据方向	功 能 描 述
X1	输入	晶振输入,即振荡器和内部时钟发生器电路的输入
X2	输出	晶振输出,振荡放大器的输出
EINT0	输入	外部中断输入 0,通用型外部中断输入引脚,低电平有效,该引脚也可以将处理器从空闲或掉电模式中唤醒
EINT1	输入	外部中断输入 1,通用型外部中断输入引脚,低电平有效,该引脚也可以将处理器从空闲或掉电模式中唤醒
EINT2	输入	外部中断输入 2,通用型外部中断输入引脚,低电平有效,该引脚也可以将处理器从空闲或掉电模式中唤醒
EINT3	输入	外部中断输入 3,通用型外部中断输入引脚,低电平有效,该引脚也可以将处理器从空闲或掉电模式中唤醒
/RST	输入	外部复位输入,该引脚上的低电平将芯片复位 i,使得 IO 得到初始化

在 ARM 嵌入式系统中,用户可以通过对应的寄存器实现对上述表 3.3 中系统控制模块的操作,具体如表 3.4 所示。

表 3.4 系统控制寄存器功能描述

寄存器名称	功能描述	访问属性	复位初始值
外部中断			
EXTIN	外部中断标志寄存器	RW	0
INTWAKE	外部中断唤醒寄存器	RW	0
EXTMODE	外部中断方式寄存器	RW	0
EXTPOLAR	外部中断极性寄存器	RW	0
存储器映射控制			
MEMMAP	存储器映射控制寄存器	RW	0
锁相环			
PLLCON	PLL 控制寄存器	RW	0
PLLCFG	PLL 配置寄存器	RW	0
PLLSTAT	PLL 状态寄存器	RO	0
PLLFEED	PLL 馈送寄存器	WO	NA
功率控制			
PCON	功率控制寄存器	RW	0
PCONP	外设功率控制	RW	0x3BE
VPB 分频器			
VPBDIV	VPB 分频控制寄存器	RW	0
复位			
RSID	复位源识别寄存器	RW	0

3.5.1　晶体振荡器

STM32 系列单片机晶振输入端 XTAL1 可接收 1～72MHz 占空比为 50%的时钟信号，如图 3.7 所示。

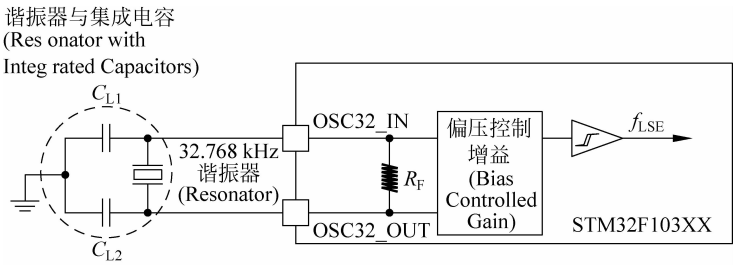

图 3.7　STM32 系列处理器的晶振电路

系统的时钟主要由以下几个方法获取：

(1) HSI，高速内部时钟，即 RC 振荡器，时钟频率为 8MHz。

(2) HSE，高速外部时钟，可外接石英、陶瓷谐振器，或者接外部时钟源，频率范围为 4～16MHz。

(3) LSI，低速内部时钟，即 RC 振荡器，频率为 40kHz。

(4) LSE，低速外部时钟，外接频率为 32.768kHz 的石英晶体。

(5) PLL，锁相环倍频输出，其中锁相环的时钟输入源可以选择为 HIS/2、HSE 或者 HSE/2。倍频时钟可以选择 2～16 的整数倍，但其输出频率最高不得超过 72MHz。

其中 40kHz 的 LSI 供独立看门狗 IWDG 使用。除此之外，还可以被选择为实时时钟 RTC 的时钟源。通常而言，实时时钟 RTC 的时钟源还可以选择 LSE，或者 HSE 的 128 分频。用户可以通过寄存器 RTCSEL[1:0]来选择实时时钟 RTC 的时钟源。

需要说明的是，振荡器输出频率称为 Fosc。为了便于频率符号的书写和描述，ARM 处理器中的时钟频率通常称为 cclk。在没有使用 PLL 的情况下，Fosc 与 cclk 在数值上是一致的。

STM32 系列处理器的振荡器可以工作在两种模式下：从属模式(外接输入时钟源)和振荡模式(外接振荡电路)，具体如图 3.8 所示。

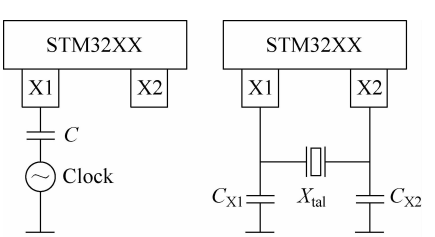

图 3.8　STM32XX 处理器的晶振电路

在从属模式下，输入信号时钟的引脚与一个 100pf 的电容相连，且输入信号时钟的幅值应当不小于 200mV，X2 引脚悬空不连接。如果用户使用时钟的从属模式，则输入信号时钟的频率被限制在 4～16MHz。

此外，系统时钟还可以工作在振荡模式下，具体的电路连接如图 3.8 所示。由于在 ARM 处理器的内部已经集成了一个反馈电阻，所以用户只需要在外部连接一个晶振和两个起振电容就可以形成基本的振荡电路了。

在外接振荡电路的工作模式下，晶体振荡器与起振电容的参数要根据具体输出的频率

范围来确定。用户可以根据表 3.5 中的内容进行选择。

表 3.5 振荡模式下电容的取值

基本振荡频率/MHz	最大晶体串联电阻	外部负载起振电容/pf
1～5	NA	10
	NA	20
	300R	30
5～10	300R	10
	300R	20
	300R	30
10～15	300R	10
	220R	20
	140R	30
15～20	220R	10
	140R	20
	80R	30
20～25	160R	10
	90R	20
	50R	30
25～30	130R	10
	50R	20
	NA	30

3.5.2 外部中断输入

在 STM32 系列处理器中支持 1～240 外部中断输入，具体数值由芯片生产厂商在设计芯片的时候决定，并可以用于将处理器从掉电模式唤醒。

在 ARM 嵌入式系统中，用户可将多个引脚同时连接到同一路外部中断，此时，外部中断逻辑根据中断方式位以及中断极性标志位的参数设置，分别以如下方式处理：

(1) 低电平有效的激活方式，选用外部中断 EINT 功能的全部引脚状态都连接到一个正电平逻辑的“与”门。

(2) 高电平有效的激活方式，选用外部中断 EINT 功能的全部引脚状态都连接到一个正电平逻辑的“或”门。

需要说明的是，当多个 EINT 引脚连接到逻辑“或”门时，用户可以在中断服务程序中通过设置相应的参数寄存器来判别中断的来源，如表 3.6 所示。

表 3.6 STM32 处理器中的中断寄存器

中断寄存器	寄存器描述	寄存器功能描述
AFIO_EXTICR1	外部中断配置寄存器 1	用于配置外部中断寄存器 1 的输入源
AFIO_EXTICR2	外部中断配置寄存器 2	用于配置外部中断寄存器 2 的输入源
AFIO_EXTICR3	外部中断配置寄存器 3	用于配置外部中断寄存器 3 的输入源
AFIO_EXTICR4	外部中断配置寄存器 4	用于配置外部中断寄存器 4 的输入源
EXTI_IMR	中断屏蔽寄存器	屏蔽中断线上的中断请求

续表

中断寄存器	寄存器描述	寄存器功能描述
EXTI_EMR	事件屏蔽寄存器	屏蔽中断线上的事件请求
EXTI_RTSR	上升沿触发选择寄存器	用于配置中断线上的上升沿触发事件
EXTI_FTSR	下降沿触发选择寄存器	用于配置中断线上的下降沿触发事件
EXTI_SWIER	软件中断事件寄存器	用于配置中断线上的软件中断
EXTI_PR	中断挂起寄存器	当外部中断发生了选择的边沿事件时，寄存器对应操作位将被置 1。在该操作位写 1 可以清除当前标志位，也可以通过改变边沿检测的极性进行清除

在 ARM 嵌入式进入掉电模式并允许总线或引脚上的一个或多个事件能够使得处理器恢复正常工作，用户在程序代码中应该对引脚的外部中断功能进行重新编程，选择合适的中断方式、中断极性以及掉电模式。唤醒处理器时软件则恢复引脚复用的外围功能。

3.5.3　系统的启动模式

STM32 处理器支持了 3 种系统启动模式，且对应的存储介质均是芯片内置的。在每个 STM32 的芯片上都有两个引脚 BOOT0 和 BOOT1，这两个引脚在芯片复位时的电平状态决定了芯片复位后从哪个区域开始执行程序，具体如表 3.7 所示。

表 3.7　STM32XX 系列处理器的启动模式

启动模式的引脚		启 动 模 式	功 能 说 明
BOOT1	BOOT0		
X	0	用户闪存存储器	将用户闪存存储器选为系统启动区域
0	1	系统存储器	将系统存储器选为系统启动区域
1	1	片上 SRAM	将片上 SRAM 选为系统启动区域

STM32 系列处理器在上电复位后，在系统时钟 SYSCLK 的第 4 个上升沿，BOOT 引脚的电平状态将被系统锁存。用户可以通过设置 BOOT0 和 BOOT1 引脚的状态来设置芯片复位后的启动模式。

(1) 用户闪存启动模式，即系统从芯片内置的 Flash 中启动。在 STM32 系统中，闪存存储器映射到启动空间 0x00000000，但用户仍然可以在原有地址 0x08000000 对其进行访问。换句话说，闪存存储器中的内容可以分别从 0x00000000 和 0x08000000 两个地址区域进行访问。

(2) SRAM 启动模式，即系统从芯片内置的 RAM 区启动，相当于计算机的内存。系统将从 0x20000000 开始的地址区访问 SRAM。

(3) 系统存储器启动模式，即从芯片内部一块特定的区域的启动。STM32 处理器芯片出厂时在这个区域预置了一段 Bootloader，即 ISP 程序。这个区域的内容在芯片出厂后用户不可以修改或擦除，是一个 ROM 区；系统存储器映射到启动空间 0x00000000，但用户仍然可以在原有地址 0x1FFFF000 对其进行访问。

- BOOT1＝x，BOOT0＝0：从用户闪存启动，这是正常的工作模式。
- BOOT1＝0，BOOT0＝1：从系统存储器启动，这种模式启动的程序功能由厂家设置。

• BOOT1＝1，BOOT0＝1：从内置 SRAM 启动，这种模式可以用于调试。

需要说明的是，STM32 处理器从待机模式退出时，BOOT 引脚上的电平将被系统重新锁存。因此，在待机模式下，BOOT 引脚应当继续保持最初的启动配置。在启动延迟之后，处理器将从地址 0x00000000 读取堆栈顶部的地址，并从启动存储器的 0x00000004 指向的地址开始执行代码。

3.5.4 系统锁相环 PLL

在 STM32 处理器中，PLL 的主要时钟源可以由以下两种时钟提供，产生倍频时钟信号：

(1) HIS 时钟除以 2。

(2) HSE 时钟或通过一个可配置分频器的 PLL2 时钟。

其中，PLL2 和 PLL3 由 HSE 通过一个可配置的分频器提供时钟，具体结构如图 3.9 所示。

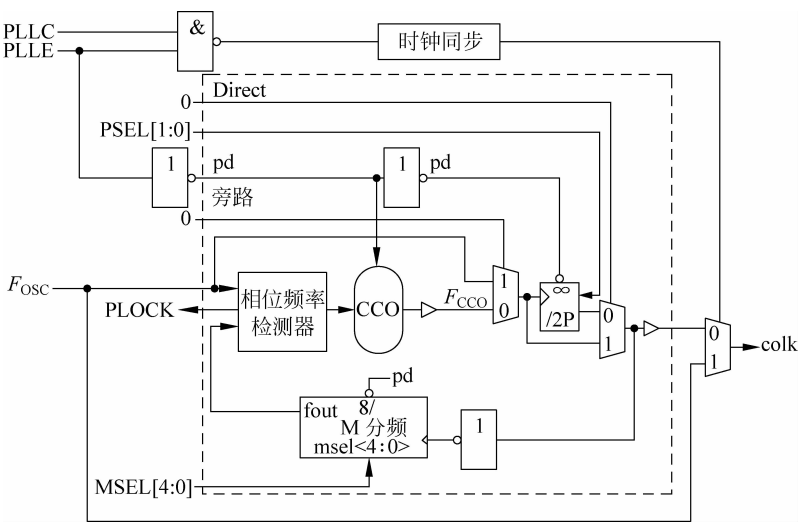

图 3.9 PLL 内部结构

用户必须在使能每一个 PLL 之前完成对 PLL 参数的配置，包括时钟源的选择、预分频系数和倍频系数等。同时，也应该在这些输入时钟信号稳定之后才能对 PLL 进行使能操作。一旦系统 PLL 被成功使能，这些配置参数将不能再被改变。

如果用户希望改变 PLL 的输入时钟源时，必须先选中新的时钟源，即完成对新时钟源的参数配置，之后才能关闭 PLL 原有的时钟源。有关 STM32 系列处理器中 PLL 的特性参数，可以通过表 3.8 中的内容进行查阅。

表 3.8 STM32 处理器中 PLL 的特性参数

时钟标号	时钟参数	参数取值			单位
		最小值	典型值	最大值	
f_{PLL_IN}	PLL 输入时钟信号	1	8	25	MHz
	PLL 输入时钟占空比	40		60	%
f_{PLL_OUT}	PLL 倍频输出时钟信号	16		72	MHz
t_{LOCK}	PLL 锁存时间			200	μs

3.5.5　系统休眠与低功耗

当 STM32 处理器或系统电源复位后，ARM 处理器将处于运行状态。如果用户不需要 CPU 继续运行，如系统在等待某个外部时间，可以利用多种方法使系统进入低功耗模式以节省系统功耗。用户可以根据最低电源消耗、最快启动时间以及可用的唤醒源等条件，选定一个最佳的低功耗模式。

在 STM32F103XX 系列处理器中，系统提供了 3 种低功耗的模式：

(1) 休眠模式，即系统内核停止运行，而所有的外部设备，包括系统核心外设，如系统中断、系统时钟等仍然正常运行。

(2) 停机模式，即系统所有时钟全部停止。

(3) 待机模式，即关闭为内核供电的 1.8V 电源。

用户可以通过表 3.9 查看系统所提供的 3 种低功耗模式的特点。

表 3.9　系统低功耗模式

节电模式	进入节电模式的寄存器操作	唤醒操作	1.8V 内核区域的时钟	VDD 电源区域的时钟	电压调节器
休眠模式	WFI	任何中断	CPU 时钟关闭，对其他时钟以及 ADC 时钟无影响	无影响	开
	WFE	唤醒事件			
停机模式	PDDS 和 LPDS 位 + SLEEPDEEP 位 + WFI 或 WFE	外部中断	所有使用 1.8V 区域的时钟均关闭，HIS 和 HSE 的振荡器均关闭		在低功耗模式下可依据电源控制寄存器(PWR_CR)的参数设定，进行开/关设置
待机模式	PDDS 位 + SLEEPDEEP 位 + WFI 或 WFE	WKUP 引脚的上升沿，RTC 警告时事件，NRST 引脚上的外部复位，IWDG 复位			关

当然，为了进一步降低系统的功耗，用户还可以通过关闭系统时钟，关闭 APB 和 AHB 总线上未被使用的外设时钟，同样可以达到降低系统功耗的目的。

当 STM32 处理器工作在正常运行模式下，用户可以通过对预分频寄存器进行编程，以降低系统时钟(SYSCLK、HCLK、PCLK1 和 PCLK2)的速度。在系统进入休眠模式前，用户也可以利用分频器来降低外部设备的时钟。

1. 休眠模式

用户可以通过执行 WFI 或者 WFE 指令使得系统进入休眠状态。根据 STM32 系列处理器中控制寄存器 SLEEPONEXIT 位的数值，有两种选项可以提供给用户用于选择休眠模式进入机制：

(1) SLEEP-NOW,即如果 SLEEPONEXIT 位被清 0,当 WRI 或 WFE 指令被执行的时候,STM32 处理器立即进入休眠模式。

(2) SLEEP-ON-EXIT,即如果 SLEEPONEXIT 位被置 1,当 WRI 或 WFE 指令被执行的时候,系统从最低优先级的中断处理程序中退出后,STM32 处理器立即进入休眠模式。

用户可以通过 WFI 或 WFE 指令将 ARM 处理器进入休眠模式,但对于这两条不同的休眠指令,需要不同的外部条件才能将处理器从休眠模式中唤醒,如表 3.10 所示。

表 3.10 休眠模式的进入与唤醒

休眠模式	SLEEP-NOW 模式	SLEEP-ON-EXIT 模式
进入模式	在以下条件下执行等待中断 WFI 或等待事件 WFE 指令: (1) SLEEPDEEP=0; (2) SLEEPONEXIT=0	在以下条件下执行等待中断 WFI 指令: (1) SLEEPDEEP=0; (2) SLEEPONEXIT=0
退出模式	如果执行 WFI 指令进入休眠模式:中断 如果执行 WFE 指令进入休眠模式:唤醒事件	中断

如果用户使用 WFI 指令将 ARM 处理器进入休眠模式,则任意一个被嵌套向量中断控制器响应的外设中断都可以将系统从休眠模式唤醒。

如果用户使用 WFE 指令将 ARM 处理器进入休眠模式,则用户需要使用唤醒事情才能将系统从休眠模式唤醒。而唤醒事件可以主要通过以下两种方式产生:

(1) 在外设控制寄存器中使能一个中断,而不是在嵌套向量中断控制器 NVIC 中使能中断,并且在 ARM 系统控制寄存器中使能 SEVONPEND 位。需要注意的是,当 ARM 处理器从 WFE 指令中唤醒后,外设的中断挂起位和外设的 NVIC 中断的挂起位将被清除。

(2) 配置一个外部或者内部的 EXIT 中断线为事件模式。当 ARM 处理器从 WFE 指令中唤醒后,由于与事件中断线对应的挂起位没有被置 1,因此用户也不必清除外设的中断挂起位或者外设的 NVIC 中断通道挂起位。

通常情况下,由于在中断的进入和退出过程中几乎不存在时间上的损失,因此将系统从休眠模式中唤醒所需的时间比较短。

2. 停机模式

在 STM32 系列芯片中,停机模式是在 ARM 处理器深度休眠模式基础上结合了对外设时钟控制机制,在停机模式下电压调节器可以运行在正常或者低功耗模式。处理器工作在停机模式下的时候,1.8V 供电区域内的所有时钟都被停止,PLL、HIS 以及 HSE 振荡器的功能也被禁止,只有 SRAM 和寄存器中的数值被保留。除此之外,在停机模式下,处理器中所有的引脚都保持在原有运行状态下的数值。用户可以根据表 3.11 中的内容进入或退出停机模式。

表 3.11　停机模式的进入与唤醒

停机模式	进 入 模 式	退 出 模 式
进入/退出	在以下条件下执行等待中断 WFI 或等待事件 WFE 指令： (1) SLEEPDEEP=1 (2) 电源控制寄存器 PWR_CR 中 PDDS=0 (3) 设置电源控制寄存器 PWR_CR 中 LPDS 位选择电压调节器的模式	如果执行 WFI 指令进入停机模式：设置任意一个中断线为中断模式，同时必须在 NVIC 中使能相应的外部中断向量； 如果执行 WFE 指令进入停机模式：设置任意一个外部中断线为事件模式
唤醒延迟	HIS RC 唤醒时间 + 电压调节器从低功耗唤醒的时间	

在停机模式下，通过设置电源控制寄存器 PWR_CR 的 LPDS 位使得内部调节器进入低功耗模式，可以使整个 ARM 嵌入式系统具有更低的系统功耗；如果当前处理器正在对片上闪存进行编程，则需要等待系统完成对内存的访问才会进入停机模式；同样，如果处理器正在对 APB 进行访问，则需要等待系统完成对 APB 访问结束后，系统才会进入停机模式。

系统在进入停机模式后，如果在进入该模式前 ADC 和 DAC 没有被关闭，则这些设备仍然会继续消耗电流。用户可以通过设置相应的控制寄存器来关闭这些外设。此外，在停机模式下，电压调节器也处于低功耗模式下，当系统从停机模式退出时，将会消耗额外的一段时间用于设备的启动。用户可以在进入低功耗模式前，保持内部电压调节器开启的状态，这样在退出停机模式时，可以减少时间上的消耗，但相应的系统功耗会增加。

3. 待机模式

在 STM32 系列处理器中，待机模式可以实现系统的最低功耗。该模式在处理器深度休眠模式的基础上关闭电压调节器。除了 1.8V 的供电区域被停止供电外，PLL、HIS 和 HSE 振荡器也被停止供电，正常运行状态下暂存在 SRAM 和寄存器中的内容也不会被保存，只有备份寄存器和待机电路部分维持最低的系统供电。停机模式的进入与唤醒如表 3.12 所示。

表 3.12　停机模式的进入与唤醒

待 机 模 式	条 件 说 明
进入	在以下条件下执行等待中断 WFI 或等待事件 WFE 指令： (1) SLEEPDEEP=1 (2) 电源控制寄存器 PWR_CR 中 PDDS=1 (3) 电源控制/状态寄存器 PWR_CSR 中 WUF=0
退出	以下 4 个条件满足任何一个： (1) WKUP 引脚接收到上升沿信号 (2) RTC 闹钟事件的上升沿信号 (3) NRST 复位引脚上接收到复位信号 (4) IWDG 复位
唤醒延迟	复位阶段时由于电压调节器启动所消耗的时间

用户可以根据表 3.12 中的内容进入或退出待机模式。从表格中可以看出，当一个外部 NRST 复位信号、IWDG 复位信号、WKUP 引脚上的上升沿或者 RTC 闹钟事件的上升沿触发时，处理器将从待机模式退出。处理器从待机模式唤醒后，除电源控制/状态寄存器

PWR_CSR 外，其他所有的寄存器将复位，即从待机模式唤醒后的代码执行等效于系统复位后的执行。

需要提醒用户注意的是，在待机模式下，处理器除了以下几个引脚外，所有的 IO 引脚均处于高阻状态：

(1) 复位引脚 NRST，该引脚在待机模式下始终有效。

(2) 被使能的唤醒引脚。

(3) 设置为防止侵入或校准输出时的 TAMPER 引脚。

在通常情况下，如果用户在进行代码调试的过程中使得处理器进入停止或待机模式，此时由于处理器内核失去了时钟信号，因此系统将失去调试连接。

4. 低功耗模式下的自动唤醒

当处理器工作在上述 3 种低功耗模式下时，不一定要完全依赖于外部的中断来唤醒系统退出低功耗模式。同样，用户也可以通过 RTC 来唤醒低功耗模式下的处理器。

为了使用 RTC 闹钟时间将系统从低功耗模式下唤醒，必须进行如下配置：

(1) 将外部中断线 17 设置为上升沿触发。

(2) 配置 RTC 使其可以产生 RTC 闹钟事件。

通过对备份区域控制寄存器 RCC_BDCR 中 RTCSEL[1:0]位的设置，RTC 中的两个时钟源就可以实现上述唤醒功能，分别为：

(1) 低功耗 32.768kHz 的外部晶振 LSE，该时钟源为系统低功耗模式的唤醒提供了一个精确而功耗极低的时间基准，在典型环境下其功耗小于 1μA。

(2) 低功耗内部 RC 振荡器 LSI RC。

如果系统设计对成本比较敏感，使用系统内部的 RC 振荡器 LSI RC 作为时钟源，则可以节省一个外部 32.768kHz 晶振的成本。但 RC 振荡器会增加系统的电源消耗。

3.5.6 系统复位

在 STM32F103XX 系列单片机中，系统支持 3 种复位形式，分别为系统复位、上电复位和备份区域复位。处理器在接收到复位信号后将所有寄存器还原到初始状态。

1. 系统复位

系统复位将初始化并还原所有的寄存器至最初的状态。此时，除了时钟控制器 RCC_CSR 寄存器中的复位标志位和备份区域中的寄存器外，所有的寄存器都会复位。

可以通过以下任意一个时间来产生系统复位，可以通过查看 RCC_CSR 控制寄存器中的复位标志位来识别具体复位事件的来源，具体为：

(1) NRST 引脚上检测到低电平，该复位方式也称为外部复位。

(2) 窗口看门狗计数终止，该复位方式也称为 WWDG 复位。

(3) 独立看门狗计数终止，该复位方式也称为 IWDG 复位。

(4) 软件复位，该复位方式也称为 SW 复位。

通过将 ARM 系统中断应用和复位控制寄存器的 SYSRESETREQ 位设置为 1，可以实现系统的软件复位。

(5) 低功耗管理复位。

有关系统低功耗管理方式的复位，可以通过以下两种方式产生：

- 在进入待机模式时产生低功耗管理复位：可以通过将用户选择字节中的 nRST_STDBY 位设置为 1 实现系统复位。此时，即使当前系统执行了进入待机模式的命令，处理器也被复位而不是进入待机模式。
- 在进入停机模式的时候产生低功耗管理复位：通过将用户选择字节中的 nRST_STOP 位设置为 1 实现系统复位。此时，即使当前系统执行了进入停机模式的命令，处理器也同样被复位而不是进入停机模式。

2. 电源复位

STM32 系列处理器可以有两种方式：

(1) 上电 POR/掉电 PDR 复位。

(2) 从待机模式中返回。

具体的系统硬件结构如图 3.10 所示。

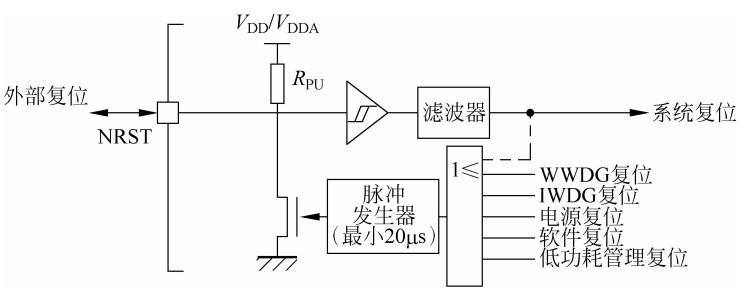

图 3.10　外部复位电路

在发生电源复位时，处理器中除了备份寄存器外所有的寄存器都将被复位。在图 3.10 中，复位信号与处理器的 RESET 引脚相连，并在复位过程中保持低电平。在发生电源复位后，系统复位的入口矢量被固定在地址 0x00000004，即系统会从该地址重新运行用户的程序代码。

在复位过程中，芯片内部的复位信号会在 NRST 引脚上输出，脉冲发生器保证每一个复位源都能保持至少 20μs 的脉冲延时。当 NRST 引脚被外部复位信号拉低并产生外部复位时，处理器将产生复位脉冲。

3. 备份区域复位

在 STM32 系列处理器中，备份区域支持两个专门的复位操作。需要说明的是，备份区域的复位操作只会影响备份区域的寄存器。

可以通过以下两种方式产生备份区域的复位操作：

(1) 软件方式产生备份区域复位：对于备份区域复位操作可以由设置备份区域控制寄存器 RCC_BDCR 中的 BDRST 位产生。

(2) 在 VDD 和 VBAT 两者均掉电的前提下，VDD 和 VBAT 上电将触发备份区域复位。

由于 STM32 系列处理器集成了内部上电复位电路，因此用户在使用复位电路时，需要在外部接一个 10kΩ 的上拉电阻，具体如图 3.11 所示。

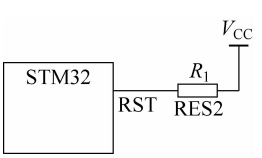

图 3.11　STM32 内部复位电路

通常情况下，在设计电路的时候可以使用外部复位。为

了提高系统复位的可靠性，可以使用专用的复位芯片，如 Sipex 公司的 SP708S 等复位芯片，可以通过查阅各个公司的具体芯片手册来了解相应的复位芯片及应用电路。在图 3.12 中以带复位电路的存储芯片 CAT1025 实现系统的外部复位操作。

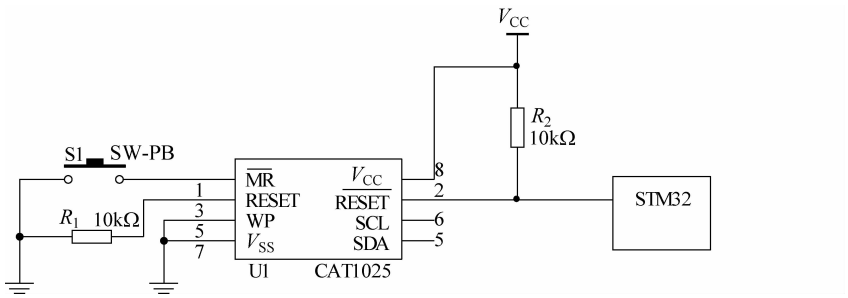

图 3.12 CAT1025 的外部复位

3.5.7 系统时钟分频

在 STM32 处理器中，可以通过 VPB 分频器(VPB Divider)对系统的时钟信号进行分频处理。VPB 分频器决定处理器时钟 CCLK 与外设器件所使用的时钟 PCLK 之间的关系。

通常情况下，VPB 分频器具有两个用途，具体如下：

(1) 通过 VPB 总线为外部设备提供必需的 PCLK 时钟以使外设可以在兼顾 ARM 处理器速度的条件下工作。为了实现这个功能，VPB 总线时钟速率可以降低到处理器时钟速率 CCLK 的 1/2 或者 1/4。由于系统 VPB 总线必须在上电后才能正常工作，所以 VPB 总线在系统复位后默认的运行状态时以 1/4 的速度运行。

(2) VPB 分频器可以使在不需要任何外设全速运行的时候降低系统的功耗。

VPB 分频器与振荡器以及 ARM 处理器时钟的连接框图如图 3.13 所示。由于 VPB 分频器连接到 PLL 输出，PLL 在空闲模式下保持有效。

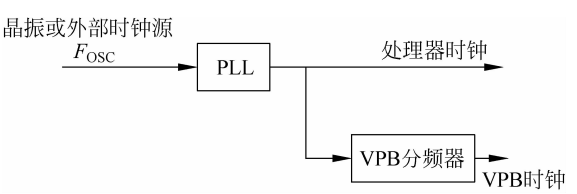

图 3.13 VPB 分频器的连接框图

3.5.8 系统掉电检测与控制

在 STM32 系列处理器中，支持一个对 V_{DD} 引脚电压的二级检测。可以通过可编程电压检测器 PVD 对 VDD 的电压与电源控制寄存器 PWR_CR 中的 PLS[2:0]位中的数据进行比较，以监控电源电压。电源控制寄存器 PWR_CR 中的 PLS[2:0]位中的数据主要用来选择监控电压的阈值。

电源控制/状态寄存器 PWR_CSR 中的 PVDO 标志用来判别电源电压 V_{DD} 是高于还是低于 PVD 的电压阈值。该判决事件会引发一个外部中断，并通过内部连接到外部中断的

第 16 根线。因此，在使用可编程电压检测器 PVD 的时候，必须先将外部中断的操作使能，否则即使电压检测产生中断，也不会被激活。

在对电源电压检测的过程中，如果电源电压 V_{DD}下降到 PVD 阈值以下或者当电源 V_{DD}上升到 PVD 阈值之上时，根据外部中断第 16 根线的上升/下降边沿触发设置，系统会产生一个 PVD 中断，具体如图 3.14 所示。

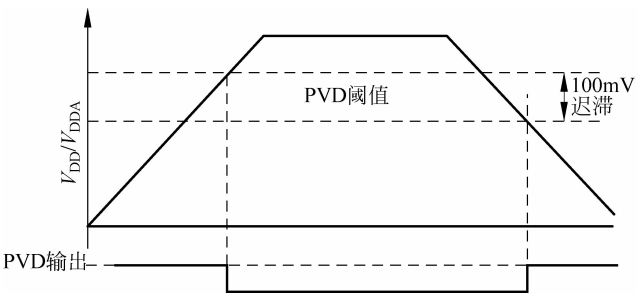

图 3.14　PVD 阈值与 PVD 的输出波形

3.6　STM32F103XX 向量中断控制器

前面已经多次提到，向量中断控制器 NVIC 是 ARM 嵌入式系统中不可分割的一部分。它与系统 ARM 内核的逻辑紧密耦合在一起，共同完成对系统中断的响应，可以通过存储器映射的方式实现对 NVIC 寄存器的访问操作。在 ARM 系统中，NVIC 除了包含控制寄存器和中断处理的控制逻辑外，还包含 MPU、SysTick 定时器以及调试控制相关的寄存器。

3.6.1　中断的概念与类型

所谓中断，是指处理器停止当前正在执行的程序去执行另外一个更为“紧急”的事件。例如，外部中断和处理器执行一个未定义的指令都会产生一个处理器的中断操作。在 STM32 系列处理器中，系统可以支持 7 种类型的中断，共 240 个外部中断输入 IRQ。

ARM 处理器中支持的 7 种类型中断，如表 3.13 所示，其中列出了 ARM 处理器中所有的中断异常模式。当系统发生中断的时候，ARM 处理器程序将跳转到对应中断程序的入口地址开始执行。这些中断程序的入口地址就是“中断向量”。

表 3.13　常见 ARM 中断的类型

中 断 类 型	工 作 模 式	中断入口地址	中断优先级
复位	管理模式	0x00000000	1
未定义指令	未定义	0x00000004	6
软件中断	管理模式	0x00000008	6
预取中断	中止	0x0000000C	5
数据中断	中止	0x00000010	2
外部中断	IRQ 外部中断	0x00000018	4
快速中断	FIQ 快速中断	0x0000001C	3

1. 复位中断

在 STM32 系列处理器中，当处理器检测到复位信号时，将立即停止执行当前的指令，并进入管理模式，跳转到 ARM 状态，并禁止所有的快速中断和外部中断。最后根据 ARM 处理器具体配置的不同，程序跳转到复位入口地址 0x00000000 处开始执行。

2. 未定义指令中断

当 ARM 处理器执行写处理器的时候没有响应，或者 ARM 处理器执行了一条没有定义的指令代码，这时系统会产生一个未定义指令异常中断。当系统出现未定义指令时，ARM 处理器进入未定义模式，并切换到 ARM 状态，禁止所有外部中断，并根据 ARM 处理器配置的不同，跳转到未定义异常入口地址 0x00000004。

3. 软件中断

当用户正在执行软件中的中断指令 SWI 时，系统将产生中断异常。此时 ARM 处理器进入管理模式，并切换到 ARM 状态，禁止所有正常中断，并根据 ARM 处理器配置的不同，跳转到未定义中断入口地址 0x00000008。需要说明的是，使用软件中断异常可以使 ARM 处理器通过软件的方式切换到管理工作模式，从而访问在用户工作模式下受保护的系统资源。

4. 预取中断

当存储系统发出中止信号后，ARM 处理器仍然执行取得的无效指令时，系统将产生预取中断异常。此时，处理器进入中止模式，并切换到 ARM 状态，禁止所有正常中断，并根据 ARM 处理器配置的不同，跳转到中止异常中断入口地址 0x0000000C。

5. 数据中断

当系统发出中止信号后，指令对数据的访问无法得到有效的响应，此时系统会产生一个数据中断。产生数据中断的时候，ARM 处理器进入中止模式，并切换到 ARM 状态，禁止所有正常中断，并根据 ARM 处理器配置的不同，跳转到数据中断入口地址 0x00000010。

6. 外部中断

一般，所有的 ARM 处理器基本都支持外部中断，即都具有 IRQ 输入引脚。当这些引脚上的信号满足中断触发条件，即高电平触发、低电平触犯、上升沿触发、下降沿触发等，系统会产生外部中断 IRQ 异常。

当 CPSR 寄存器中的 I 位被设置为 1 的时候，处理器会禁止 IRQ 中断。当 ARM 处理器检测到外部的 IRQ 中断信号时，则进入 IRQ 模式，切换到 ARM 状态，禁止所有正常中断，并根据 ARM 处理器配置的不同，跳转到外部中断入口地址 0x00000018。

7. 快速中断

在 ARM 处理器中，除了可以支持普通的外部中断 IRQ 外，还支持一类特殊的中断模式，即快速中断 FIQ。快速中断 FIQ 相对于外部中断 IRQ 具有更高的优先级，通常用于数据传输、通道处理等实时性要求更高的场合。

可以通过将 CPSR 寄存器中的 I 位设置为 1，以禁止系统的快速中断 FIQ 功能。当处理器检测到快速中断 FIQ 时，ARM 处理器进入 FIQ 模式，切换到 ARM 状态，禁止所有正常中断、外部中断以及快速中断，并根据 ARM 处理器配置的不同，跳转到快速中断入口地址 0x0000001C。

3.6.2　外部中断/事件控制器的特点与结构

对于 STM32 系列处理器，外部中断/事件控制器由 20 个产生中断/事件请求的边沿检测器组成。处理器引脚上的每一个输入端口线都可以独立地配置成输入中断类型(脉冲或挂起)以及对应的触发事件(上升沿或下降沿或者双边沿触发)。同样，每一个输入端口线也可以独立地被屏蔽，具体的系统结构如图 3.15 所示。

外部中断/事件 EXTI 控制器的主要特性如下：

(1) 每一个中断/事件都可以独立地被触发或屏蔽。

(2) 每一个中断/事件都具备专用的状态位。

(3) 最多可以支持 20 个软件中断/事情请求。

(4) 可检测到脉冲宽度低于 APB2 时钟宽度的外部信号。

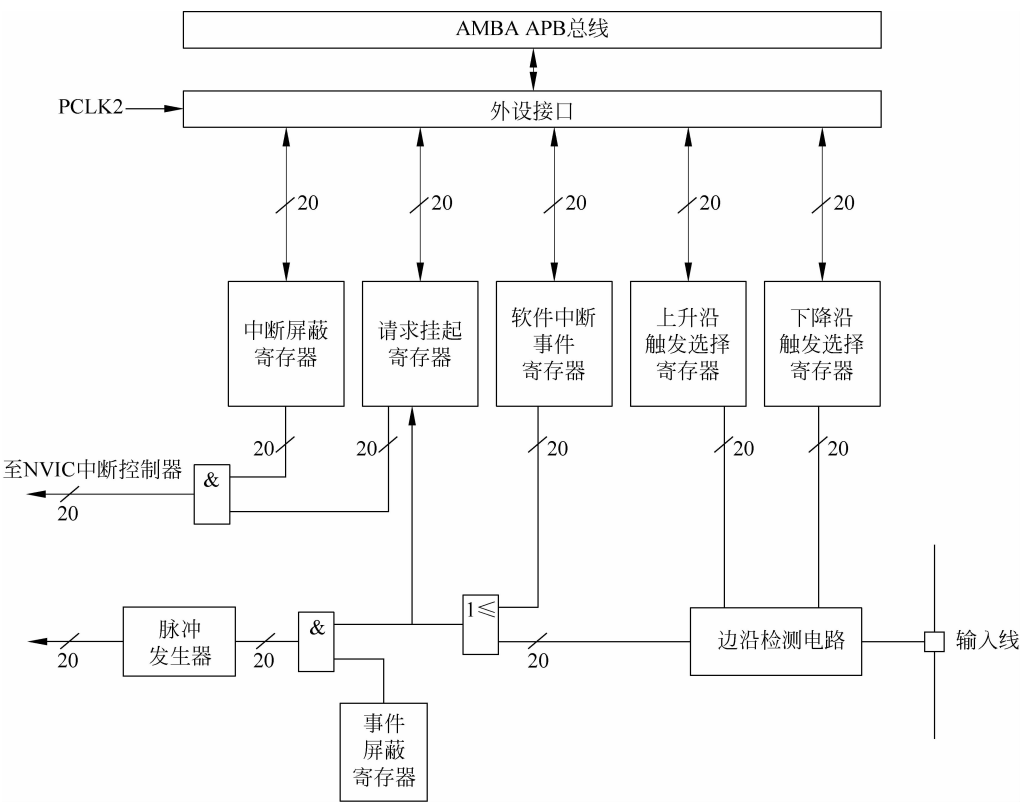

图 3.15　STM32 处理器的中断结构图

在 STM32 系列处理器中，如果用户希望产生“中断”，则必须先配置好芯片中断的引脚线，并对其进行“使能”操作。根据需要的边沿检测设置两个触发寄存器，同时在中断屏蔽寄存器的相应位写 1 以允许中断的请求。当外部中断线上发生了相应的触发边沿信号后，系统产生一个中断请求，对应的挂起标志位也会被设置为 1。可以将中断挂起寄存器中对应的标志位设置为 1，以清除当前的中断请求。

在 STM32 系列处理器中，如果希望产生“事件”，则必须先配置并完成对事件线的使能操作。通过设置两个触发寄存器来完成对边沿检测的配置，同时在事件屏蔽寄存器的相应

位写 1 以允许事情请求操作。当事件线上发生了对应的边沿信号时，系统产生一个事件请求脉冲，对应的挂起位并不会被置 1。

1. 硬件中断的配置

可以通过下面的步骤来配置多个线路作为中断源，具体操作如下：

(1) 在 EXTI_IMR 寄存器中配置多个线路中断的屏蔽位。

(2) 在 EXTI_RTSR 寄存器和 EXTI_FTSR 寄存器中配置所选择的中断线的触发选择位。

(3) 配置对应到外部中断控制器 EXTI 的 NVIC 中断通道的使能和屏蔽位，使多个中断线中的请求可以被及时响应。

2. 硬件事件的配置

对于系统中的事件处理，可以通过以下几个步骤来实现对硬件事件参数的配置：

(1) 通过 EXTI_EMR 寄存器配置多个事件线的屏蔽位。

(2) 通过 EXTI_RTSR 寄存器和 EXTI_FTSR 寄存器配置事件线的触发选择器。

3. 软件中断/事件的配置

对于系统中的软件中断/事件处理，可以通过以下几个步骤来实现对软件中断/事件的配置：

(1) 通过 EXTI_EMR 寄存器和 EXTI_IMR 寄存器配置多个中断/事件线的屏蔽位。

(2) 通过 EXTI_SWIER 寄存器配置软件中断寄存器的请求位。

在图 3.16 中，列出了外部中断与通用 IO 口之间的硬线连接。可以通过 AFIO_EXTICRx 配置 GPIO 端口上的外部中断/事件。

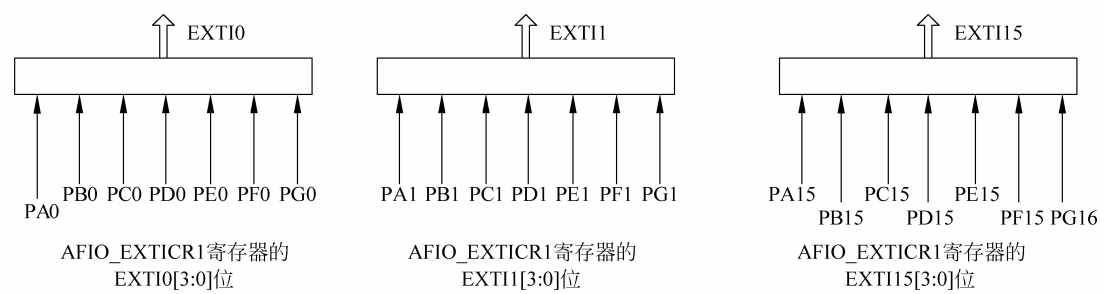

图 3.16 STM32 外部中断与通用 IO 口的映像

特别需要注意的是，在通过 EXTI 配置寄存器 AFIO_EXTICRx 配置 EXTI 线路上的 GPIO 前，必须先使能 AFIO 时钟，具体如下：

- EXTIx[3：0] = 0000 时，选择端口 A 的 x 号引脚。
- EXTIx[3：0] = 0001 时，选择端口 B 的 x 号引脚。
- EXTIx[3：0] = 0010 时，选择端口 C 的 x 号引脚。
- EXTIx[3：0] = 0011 时，选择端口 D 的 x 号引脚。
- EXTIx[3：0] = 0100 时，选择端口 E 的 x 号引脚。
- EXTIx[3：0] = 0101 时，选择端口 F 的 x 号引脚。
- EXTIx[3：0] = 0110 时，选择端口 G 的 x 号引脚。

除了图 3.16 中描述的 0～15 中断线外，还有另外 4 个特别的 EXTI 中断线可供使用，

具体的连接方式如下：

(1) EXTI 线 16 连接到可编程电压检测器 PVD 输出。

(2) EXTI 线 17 连接到实时时钟 RTC 闹钟事件。

(3) EXTI 线 18 连接到 USB 唤醒事件。

(4) EXTI 线 19 连接到以太网唤醒事件。

3.6.3　EXTI 的寄存器

在使用 ARM 处理器的中断前，必须通过 EXTI 相应的寄存器对其各个参数进行配置。需要注意的是，用户在设置寄存器的过程中，必须采用字(word)的方式对其进行操作。

1. 中断屏蔽寄存器

在 STM32 系列处理器中，中断屏蔽寄存器 EXTI_IMR 主要用于设置中断线上的中断屏蔽操作，即 Interrupt Mask on line X。由于 STM32 系列处理器是 32b 的处理器，因此中断屏蔽寄存器的宽度也为 32b，具体内容如表 3.14 所示。

表 3.14　中断屏蔽寄存器 EXTI_IMR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												MR19	MR18	MR17	MR16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0

从表 3.14 中可以看出，中断屏蔽寄存器 EXTI_IMR 中位[31：20]是系统保留位，且必须时钟保持为复位状态；位[19：0]用于设置对应中断线上的中断屏蔽，MRx 表示将中断线 x 上的中断屏蔽位。若 MRx=0，表示屏蔽来自线 x 上的中断请求；若 MRx=1，表示开放来自线 x 上的中断请求。

2. 事件屏蔽寄存器

在 STM32 系列处理器中，事件屏蔽寄存器 EXTI_EMR 主要用于设置中断线上的事件屏蔽操作，即 Event Mask on line X。由于 STM32 系列处理器是 32b 的处理器，因此事件屏蔽寄存器的宽度也为 32b，具体内容如表 3.15 所示。

表 3.15　事件屏蔽寄存器 EXTI_EMR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												MR19	MR18	MR17	MR16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MR15	MR14	MR13	MR12	MR11	MR10	MR9	MR8	MR7	MR6	MR5	MR4	MR3	MR2	MR1	MR0

从表 3.15 中可以看出，中断屏蔽寄存器 EXTI_EMR 中位[31：20]是系统保留位，且必须时钟保持为复位状态；位[19：0]用于设置对应中断线上的事件屏蔽，MRx 表示将中断线 x 上的事件屏蔽位。若 MRx=0，表示屏蔽来自线 x 上的事件请求；若 MRx=1，表示开放来自线 x 上的事件请求。

3. 上升沿触发选择寄存器

在 STM32 系列处理器中，上升沿触发选择寄存器 EXTI_RTSR 主要用于设置中断线上触发脉冲类型为上升沿（即 rising trigger event configuration bit of line X）。由于 STM32 系列处理器是 32b 的处理器，因此上升沿触发选择寄存器的宽度也为 32b，具体内容如表 3.16 所示。

表 3.16 上升沿触发选择寄存器 EXTI_RTSR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												TR19	TR18	TR17	TR16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0

从表 3.16 中可以看出，上升沿触发选择寄存器 EXTI_RTSR 中位[31：20]是系统保留位，且必须时钟保持为复位状态；位[19：0]用于设置对应中断线上的触发方式，TRx 表示中断线 x 上的上升沿触发事件配置。若 MRx＝0，表示禁止输入线 x 上的上升沿中断或事件的触发；若 MRx＝1，表示允许输入线 x 上的上升沿中断或事件的触发。

需要注意的是，外部唤醒线都是边沿触发的，在这些信号线上不能出现毛刺信号。另外，在对中断屏蔽寄存器 EXTI_RTSR 进行写操作的时候，外部中断线上的上升沿触发信号不能被识别，挂起位也不会被置位。在同一个中断线上，可以同时将其设置为上升沿触发和下降沿触发，即任何一个边沿都可以触发系统的外部中断。

4. 下降沿触发选择寄存器

在 STM32 系列处理器中，下降沿触发选择寄存器 EXTI_FTSR 主要用于设置中断线上触发脉冲类型为下降沿（即 falling trigger event configuration bit of line X）。由于 STM32 系列处理器是 32b 的处理器，因此下降沿触发选择寄存器的宽度也为 32b，具体内容如表 3.17 所示。

从表 3.17 中可以看出，下降沿触发选择寄存器 EXTI_RTSR 中位[31：20]是系统保留位，且必须时钟保持为复位状态；位[19：0]用于设置对应中断线上的触发方式，TRx 表示中断线 x 上的下降沿触发事件配置。若 MRx＝0，表示禁止输入线 x 上的下降沿中断或事件的触发；若 MRx＝1，表示允许输入线 x 上的下降沿中断或事件的触发。

表 3.17 下降沿触发选择寄存器 EXTI_FTSR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												TR19	TR18	TR17	TR16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TR15	TR14	TR13	TR12	TR11	TR10	TR9	TR8	TR7	TR6	TR5	TR4	TR3	TR2	TR1	TR0

需要注意的是，外部唤醒线同样也是边沿触发的，在这些信号线上也不能出现毛刺信号。另外，在对中断屏蔽寄存器 EXTI_FTSR 进行写操作的时候，外部中断线上的下降沿触发信号不能被识别，挂起位也不会被置位。在同一个中断线上，也可以同时将其设置为上升

沿触发和下降沿触发，即任何一个边沿都可以触发系统的外部中断。

5. 软件中断事件寄存器

在 STM32 系列处理器中，软件中断事件寄存器 EXTI_SWIER 主要用于设置中断线上的软件中断，即 software interrupt on line X。由于 STM32 系列处理器是 32b 的处理器，因此软件中断事件寄存器的宽度也为 32b，具体内容如表 3.18 所示。

表 3.18　软件中断事件寄存器 EXTI_SWIER

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												SWI19	SWI18	SWI17	SWI16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWI15	SWI14	SWI13	SWI12	SWI11	SWI10	SWI9	SWI8	SWI7	SWI6	SWI5	SWI4	SWI3	SWI2	SWI1	SWI0

从表 3.18 中可以看出，软件中断事件寄存器 EXTI_SWIER 中位[31：20]是系统保留位，且必须时钟保持为复位状态；位[19：0]用于设置对应中断线上的软件中断事件，SWIx 表示中断线 x 上的软件中断事件配置。若 SWIx＝0，可以通过对该位写 1，实现将 EXTI_PR 中相应的位挂起。此时，如果在中断屏蔽寄存器 EXTI_IMR 和事件中断寄存器 EXTI_EMR 中允许该位产生中断，则系统将产生一个中断。

6. 挂起寄存器

在 STM32 系列处理器中，挂起寄存器 EXTI_PR 主要用于识别中断线上的中断请求，即 pending bit。由于 STM32 系列处理器是 32b 的处理器，因此挂起寄存器的宽度也为 32b，具体内容如表 3.19 所示。

表 3.19　挂起寄存器 EXTI_PR

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
保留												PR19	PR18	PR17	PR16
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0

从表 3.19 中可以看出，挂起寄存器 EXTI_PR 中位[31：20]是系统保留位，且必须时钟保持为复位状态；位[19：0]用于识别对应中断线上的中断事件，PRx 表示中断线 x 上的挂起标志位。若 PRx＝0，表示没有发生触发请求；若 PRx＝1，表示发生了触发请求。需要注意的是，当在外部中断线上发生了对应的边沿触发事件时，则对应的 PRx 位将被设置为 1。可以通过在该位中再次写入 1 将其清除，也可以通过改变边沿检测的极性（上升沿触发或下降沿触发）对其进行清除；若写 0，则对该位不会产生影响。

3.6.4　中断的处理过程

在 STM32 系列处理器中，中断异常的进入和退出都需要进行一系列的操作，例如对当前程序执行的状态、配置参数等进行保存，以方便中断结束后程序的返回。此外，在进入中断之前，还需要关闭所有的外部中断源，以防止在处理中断的过程中又出现中断请求，即中

断的嵌套。

在 ARM 处理器进入中断异常之前，具体需要执行的操作如下：

(1) 在链接寄存器 LR 中保存中断异常的返回地址信息。需要说明的是，如果进入中断异常前，STM32 处理器工作在 ARM 模式下，则 LR 寄存器中保存的是当前指令的下一条指令的地址；如果进入中断异常钱，STM32 处理器工作在 Thumb 模式下，则 LR 寄存器中保存的则是当前 PC 指针的偏移量。

(2) 将 CPSR 寄存器中的数值复制到 SPSR 寄存器中，以保存当前各个标志寄存器中的状态。

(3) 设置 CPSR 寄存器中的参数。当处理器进入中断异常时，需要将 CPSR 寄存器中的 I 标志位设置为 1，以禁止外部的其他中断，防止中断的嵌套。

(4) 程序跳转到中断异常入口处执行中断处理程序。

在 ARM 处理器执行完中断处理命令后，退出中断异常前，也需要进行一系列的操作，例如恢复进入异常前处理器的状态，以及程序跳转到中断异常之前的程序地址等。

在 ARM 处理器退出中断异常之前，具体需要执行的操作如下：

(1) 清除 CPSR 寄存器中的中断禁止标志位，开放外部中断。

(2) 将 LR 寄存器中的数值减去相应的偏移量，并存储到 PC 中。需要注意的是，不同的中断向量具有不同的偏移地址量。

(3) 将 SPSR 寄存器中的数值复制到 CPSR 寄存器中，其中还包括对处理器工作状态的还原。