ARM®Cortex™-M0

32 位微处理器

SWM320 系列 MCU

库函数指南

华芯微特科技有限公司

Synwit Technology Co., Ltd.

目 录

1 概述

1.1 文档结构

本文档是 SWM320 系列驱动参考手册。系统级软件开发人员可以使用 SWM320 系列驱动来代替直接使用寄存器的编程方式进行快速的应用软件开发，这可以大大减少总的开发时间。

在本文档中，对于每一个驱动应用接口，会提供一个关于该驱动应用接口的描述、使用和示例代码。完整的驱动例程和驱动源码在 SWM320 系列的 BSP（板级支持包）里。

本文档分为若干个章节，第一章是概述。第二章是详细的驱动描述，包括：SYSCON 驱动，PORTCON 驱动，GPIO 驱动，外部中断 EXTI，TIMER 驱动，WDT 驱动，UART 驱动，I2C 驱动，SPI 驱动，PWM 驱动，ADC 驱动，DMA 驱动，CAN 驱动, RTC 驱动和 FLASH 驱动。

1.2 相关文档

《华芯微特SWM320数据手册》。

1.3 缩略语和术语

- ADC 模数转换器
- AHB 增强型高性能总线
- AMBA 增强型微控制器总线架构
- APB 增强型外围设备总线
- BOD 欠压检测
- FIFO 先进先出
- FLASH 存储控制器
- GPIO 通用输入/输出
- I2C 内部集成电路
- PLL 锁相环
- PWM 脉宽调制
- SPI 串行外围设备接口
- UART 通用异步收/发器

1.4 数据类型定义

在我们的驱动中所有的基本数据类型定义遵循 ANSI C 的定义并且和 ARM CMSIS（Cortex-M 软件接口标准）兼容。功能相关的枚举数据类型在各个相应的章节中定义。基本数据类型定义如下表。

类型	定义	描述
int8_t	signed char	8 位有符号整数
int16_t	signed short	16 位有符号整数
int32_t	signed int	32 位有符号整数
uint8_t	unsigned char	8 位无符号整数
uint16_t	unsigned short	16 位无符号整数
uint32_t	unsigned int	32 位无符号整数

2 功能描述

2.1 系统管理（SYSCON）

2.1.1 特性

- 时钟控制
- 工作模式选择
- 休眠使能
- 低功耗计数器设置
- 端口唤醒设置
- BOD 掉电级别控制
- 复位控制及状态
- UID

2.1.2 常量定义

常量名	值	描述
SYS_CLK_24MHz	0	内部高频 24MHz RC 振荡器
SYS_CLK_6MHz	1	内部高频 6MHz RC 振荡器
SYS_CLK_48MHz	2	内部高频 48MHz RC 振荡器
SYS_CLK_12MHz	3	内部高频 12MHz RC 振荡器
SYS_CLK_32KHz	4	内部高频 32KHz RC 振荡器
SYS_CLK_XTAL	5	外部 XTAL 晶体振荡器（2-30MHz）

__HSI	32000000UL	高速内部时钟
__LSI	32000UL	低速内部时钟
__HSE	32000000UL	高速外部时钟

2.1.3 函数

SystemCoreClockUpdate

原型：void SystemCoreClockUpdate(void)

```
/*****************************************************************************
* 函数名称: SystemCoreClockUpdate()
* 功能说明: This function is used to update the variable SystemCoreClock and must be called whenever the core clock is changed
* 输　　入: 无
* 输　　出: 无
* 注意事项: 无
*****************************************************************************/
```

SystemInit

原型：void SystemInit(void)

```
/*****************************************************************************
* 函数名称: SystemInit()
* 功能说明: The necessary initializaiton of systerm
* 输　　入: 无
* 输　　出: 无
* 注意事项: 无
*****************************************************************************/
```

2.2 引脚功能配置（PORTCON）

2.2.1 介绍

端口控制模块主要包括管脚输入使能，管脚功能配置， I/O 上拉、下拉、开漏配置。本系列所有型号 PORTCON 模块操作均相同，部分型号无对应管脚时，对应寄存器位无效。

2.2.2 特性

- 可将 UART/I2C/SPI/PWM/COUNTER/CAN 功能配置至任意 I/O 引脚
- 支持上拉、 下拉、 开漏功能
- 配置管脚输入使能

2.2.3 常量定义

常量名	值	描述
PORTA	0	端口 A 功能引脚
PORTB	1	端口 B 功能引脚
PORTC	2	端口 C 功能引脚
PORTM	3	端口 M 功能引脚
PORTN	4	端口 N 功能引脚
PORTP	5	端口 P 功能引脚
PORTA_PIN0_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN0_FUNMUX	1	数字模块功能引脚
PORTA_PIN0_SWCLK	2	SWD 下载接口的时钟线引脚
PORTA_PIN1_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN1_FUNMUX	1	数字模块功能引脚
PORTA_PIN1_SWDIO	2	SWD 下载接口的数据线引脚
PORTA_PIN2_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN2_FUNMUX	1	数字模块功能引脚
PORTA_PIN3_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN3_FUNMUX	1	数字模块功能引脚
PORTA_PIN4_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN4_FUNMUX	1	数字模块功能引脚
PORTA_PIN5_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN5_FUNMUX	1	数字模块功能引脚
PORTA_PIN6_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN6_FUNMUX	1	数字模块功能引脚
PORTA_PIN7_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN7_FUNMUX	1	数字模块功能引脚
PORTA_PIN8_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN8_FUNMUX	1	数字模块功能引脚
PORTA_PIN9_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN9_FUNMUX	1	数字模块功能引脚
PORTA_PIN9_ADC0_IN7	3	ADC0 模块的输入通道 7 引脚
PORTA_PIN10_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN10_FUNMUX	1	数字模块功能引脚
PORTA_PIN10_ADC0_IN6	3	ADC0 模块的输入通道 6 引脚
PORTA_PIN11_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN11_FUNMUX	1	数字模块功能引脚
PORTA_PIN11_ADC0_IN5	3	ADC0 模块的输入通道 5 引脚
PORTA_PIN12_GPIO	0	数字 GPIO 功能引脚
PORTA_PIN12_ADC0_IN4	3	ADC0 模块的输入通道 4 引脚
PORTB_PIN0_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN0_FUNMUX	1	数字模块功能引脚

常量名	值	描述
PORTB_PIN0_SD_DETECT	2	SD 卡检测功能引脚
PORTB_PIN1_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN1_FUNMUX	1	数字模块功能引脚
PORTB_PIN1_SD_CLK	2	SD 卡时钟功能引脚
PORTB_PIN2_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN2_FUNMUX	1	数字模块功能引脚
PORTB_PIN2_SD_CMD	2	SD 卡命令功能引脚
PORTB_PIN3_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN3_FUNMUX	1	数字模块功能引脚
PORTB_PIN3_SD_D0	2	SD 卡数据功能引脚
PORTB_PIN4_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN4_FUNMUX	1	数字模块功能引脚
PORTB_PIN4_SD_D1	2	SD 卡数据功能引脚
PORTB_PIN5_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN5_FUNMUX	1	数字模块功能引脚
PORTB_PIN5_SD_D2	2	SD 卡数据功能引脚
PORTB_PIN6_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN6_FUNMUX	1	数字模块功能引脚
PORTB_PIN6_SD_D3	2	SD 卡数据功能引脚
PORTB_PIN7_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN7_FUNMUX	1	数字模块功能引脚
PORTB_PIN7_SD_D4	2	SD 卡数据功能引脚
PORTB_PIN8_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN8_FUNMUX	1	数字模块功能引脚
PORTB_PIN8_SD_D5	2	SD 卡数据功能引脚
PORTB_PIN9_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN9_FUNMUX	1	数字模块功能引脚
PORTB_PIN9_SD_D6	2	SD 卡数据功能引脚
PORTB_PIN10_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN10_FUNMUX	1	数字模块功能引脚
PORTB_PIN10_SD_D7	2	SD 卡数据功能引脚
PORTB_PIN11_GPIO	0	数字 GPIO 功能引脚
PORTB_PIN11_FUNMUX	1	数字模块功能引脚
PORTB_PIN12_GPIO	0	数字 GPIO 功能引脚
PORTC_PIN0_GPIO	0	数字 GPIO 功能引脚
PORTC_PIN0_FUNMUX	1	数字模块功能引脚
PORTC_PIN1_GPIO	0	数字 GPIO 功能引脚
PORTC_PIN1_FUNMUX	1	数字模块功能引脚
PORTC_PIN2_GPIO	0	数字 GPIO 功能引脚
PORTC_PIN2_FUNMUX	1	数字模块功能引脚
PORTC_PIN3_GPIO	0	数字 GPIO 功能引脚
PORTC_PIN3_FUNMUX	1	数字模块功能引脚

常量名	值	描述
PORTC_PIN4_GPIO	0	数字 GPIO 功能引脚
PORTC_PIN4_FUNMUX	1	数字模块功能引脚
PORTC_PIN4_ADC1_IN3	3	ADC1 模块的输入通道 3 引脚
PORTC_PIN5_GPIO	0	数字 GPIO 功能引脚
PORTC_PIN5_FUNMUX	1	数字模块功能引脚
PORTC_PIN5_ADC1_IN2	3	ADC1 模块的输入通道 2 引脚
PORTC_PIN6_GPIO	0	数字 GPIO 功能引脚
PORTC_PIN6_FUNMUX	1	数字模块功能引脚
PORTC_PIN6_ADC1_IN1	3	ADC1 模块的输入通道 1 引脚
PORTC_PIN7_GPIO	0	数字 GPIO 功能引脚
PORTC_PIN7_FUNMUX	1	数字模块功能引脚
PORTC_PIN7_ADC1_IN0	3	ADC1 模块的输入通道 0 引脚
PORTM_PIN0_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN0_FUNMUX	1	数字模块功能引脚
PORTM_PIN0_NORFL_D15	2	NORFLASH 数据输出引脚
PORTM_PIN1_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN1_FUNMUX	1	数字模块功能引脚
PORTM_PIN1_NORFL_D14	2	NORFLASH 数据输出引脚
PORTM_PIN2_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN2_FUNMUX	1	数字模块功能引脚
PORTM_PIN2_NORFL_D13	2	NORFLASH 数据输出引脚
PORTM_PIN3_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN3_FUNMUX	1	数字模块功能引脚
PORTM_PIN3_NORFL_D12	2	NORFLASH 数据输出引脚
PORTM_PIN4_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN4_FUNMUX	1	数字模块功能引脚
PORTM_PIN4_NORFL_D11	2	NORFLASH 数据输出引脚
PORTM_PIN5_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN5_FUNMUX	1	数字模块功能引脚
PORTM_PIN5_NORFL_D10	2	NORFLASH 数据输出引脚
PORTM_PIN6_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN6_FUNMUX	1	数字模块功能引脚
PORTM_PIN6_NORFL_D9	2	NORFLASH 数据输出引脚
PORTM_PIN7_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN7_FUNMUX	1	数字模块功能引脚
PORTM_PIN7_NORFL_D8	2	NORFLASH 数据输出引脚
PORTM_PIN8_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN8_FUNMUX	1	数字模块功能引脚
PORTM_PIN8_NORFL_D7	2	NORFLASH 数据输出引脚
PORTM_PIN9_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN9_FUNMUX	1	数字模块功能引脚
PORTM_PIN9_NORFL_D6	2	NORFLASH 数据输出引脚

常量名	值	描述
PORTM_PIN10_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN10_FUNMUX	1	数字模块功能引脚
PORTM_PIN10_NORFL_D5	2	NORFLASH 数据输出引脚
PORTM_PIN11_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN11_FUNMUX	1	数字模块功能引脚
PORTM_PIN11_NORFL_D4	2	NORFLASH 数据输出引脚
PORTM_PIN12_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN12_FUNMUX	1	数字模块功能引脚
PORTM_PIN12_NORFL_D3	2	NORFLASH 数据输出引脚
PORTM_PIN13_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN13_FUNMUX	1	数字模块功能引脚
PORTM_PIN13_NORFL_D2	2	NORFLASH 数据输出引脚
PORTM_PIN14_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN14_FUNMUX	1	数字模块功能引脚
PORTM_PIN14_NORFL_D1	2	NORFLASH 数据输出引脚
PORTM_PIN15_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN15_FUNMUX	1	数字模块功能引脚
PORTM_PIN15_NORFL_D0	2	NORFLASH 数据输出引脚
PORTM_PIN16_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN16_FUNMUX	1	数字模块功能引脚
PORTM_PIN16_NORFL_OEN	2	NORFLASH 数据输出使能引脚
PORTM_PIN17_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN17_FUNMUX	1	数字模块功能引脚
PORTM_PIN17_NORFL_WEN	2	NORFLASH 数据写使能引脚
PORTM_PIN18_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN18_FUNMUX	1	数字模块功能引脚
PORTM_PIN18_NORFL_CSN	2	NORFLASH 功能选通引脚
PORTM_PIN19_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN19_FUNMUX	1	数字模块功能引脚
PORTM_PIN19_SDRAM_CSN	2	外部 SRAM 功能选通引脚
PORTM_PIN20_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN20_FUNMUX	1	数字模块功能引脚
PORTM_PIN20_SRAM_CSN	2	外部 SRAM 功能选通引脚
PORTM_PIN21_GPIO	0	数字 GPIO 功能引脚
PORTM_PIN21_FUNMUX	1	数字模块功能引脚
PORTM_PIN21_SDRAM_CKE	2	外部 SRAM 时钟使能功能引脚
PORTN_PIN0_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN0_FUNMUX	1	数字模块功能引脚
PORTN_PIN0_LCD_D0	2	LCD 数据输出引脚
PORTN_PIN0_ADC1_IN4	3	ADC1 模块的输入通道 4 引脚
PORTN_PIN1_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN1_FUNMUX	1	数字模块功能引脚

常量名	值	描述
PORTN_PIN1_LCD_D1	2	LCD 数据输出引脚
PORTN_PIN1_ADC1_IN5	3	ADC1 模块的输入通道 5 引脚
PORTN_PIN2_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN2_FUNMUX	1	数字模块功能引脚
PORTN_PIN2_LCD_D2	2	LCD 数据输出引脚
PORTN_PIN2_ADC1_IN6	3	ADC1 模块的输入通道 6 引脚
PORTN_PIN3_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN3_FUNMUX	1	数字模块功能引脚
PORTN_PIN3_LCD_D3	2	LCD 数据输出引脚
PORTN_PIN4_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN4_FUNMUX	1	数字模块功能引脚
PORTN_PIN4_LCD_D4	2	LCD 数据输出引脚
PORTN_PIN5_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN5_FUNMUX	1	数字模块功能引脚
PORTN_PIN5_LCD_D5	2	LCD 数据输出引脚
PORTN_PIN6_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN6_FUNMUX	1	数字模块功能引脚
PORTN_PIN6_LCD_D6	2	LCD 数据输出引脚
PORTN_PIN7_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN7_FUNMUX	1	数字模块功能引脚
PORTN_PIN7_LCD_D7	2	LCD 数据输出引脚
PORTN_PIN8_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN8_FUNMUX	1	数字模块功能引脚
PORTN_PIN8_LCD_D8	2	LCD 数据输出引脚
PORTN_PIN9_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN9_FUNMUX	1	数字模块功能引脚
PORTN_PIN9_LCD_D9	2	LCD 数据输出引脚
PORTN_PIN10_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN10_FUNMUX	1	数字模块功能引脚
PORTN_PIN10_LCD_D10	2	LCD 数据输出引脚
PORTN_PIN11_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN11_FUNMUX	1	数字模块功能引脚
PORTN_PIN11_LCD_D11	2	LCD 数据输出引脚
PORTN_PIN12_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN12_FUNMUX	1	数字模块功能引脚
PORTN_PIN12_LCD_D12	2	LCD 数据输出引脚
PORTN_PIN13_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN13_FUNMUX	1	数字模块功能引脚
PORTN_PIN13_LCD_D13	2	LCD 数据输出引脚
PORTN_PIN14_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN14_FUNMUX	1	数字模块功能引脚
PORTN_PIN14_LCD_D14	2	LCD 数据输出引脚

常量名	值	描述
PORTN_PIN15_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN15_FUNMUX	1	数字模块功能引脚
PORTN_PIN15_LCD_D15	2	LCD 数据输出引脚
PORTN_PIN16_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN16_FUNMUX	1	数字模块功能引脚
PORTN_PIN16_LCD_RD	2	LCD 读数据功能引脚
PORTN_PIN16_LCD_DOTCK	2	LCD 像素时钟
PORTN_PIN17_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN17_FUNMUX	1	数字模块功能引脚
PORTN_PIN17_LCD_CS	2	LCD 片选功能引脚
PORTN_PIN17_LCD_VSYNC	2	LCD 帧同步
PORTN_PIN18_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN18_LCD_RS	2	LCD 指令/数据功能引脚
PORTN_PIN18_LCD_DATEN	2	LCD 数据使能引脚
PORTN_PIN19_GPIO	0	数字 GPIO 功能引脚
PORTN_PIN19_LCD_WR	2	LCD 写数据功能引脚
PORTN_PIN19_LCD_HSYNC	2	LCD 行同步
PORTP_PIN0_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN0_FUNMUX	1	数字模块功能引脚
PORTP_PIN0_NORFL_A0	2	NORFLASH 地址线 0
PORTP_PIN1_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN1_FUNMUX	1	数字模块功能引脚
PORTP_PIN1_NORFL_A1	2	NORFLASH 地址线 1
PORTP_PIN2_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN2_FUNMUX	1	数字模块功能引脚
PORTP_PIN2_NORFL_A2	2	NORFLASH 地址线 2
PORTP_PIN2_SD_D7	3	SD 卡数据功能引脚
PORTP_PIN3_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN3_FUNMUX	1	数字模块功能引脚
PORTP_PIN3_NORFL_A3	2	NORFLASH 地址线 3
PORTP_PIN3_SD_D6	3	SD 卡数据功能引脚
PORTP_PIN4_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN4_FUNMUX	1	数字模块功能引脚
PORTP_PIN4_NORFL_A4	2	NORFLASH 地址线 4
PORTP_PIN4_SD_D5	3	SD 卡数据功能引脚
PORTP_PIN5_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN5_FUNMUX	1	数字模块功能引脚
PORTP_PIN5_NORFL_A5	2	NORFLASH 地址线 5
PORTP_PIN5_SD_D4	3	SD 卡数据功能引脚
PORTP_PIN6_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN6_FUNMUX	1	数字模块功能引脚
PORTP_PIN6_NORFL_A6	2	NORFLASH 地址线 6

常量名	值	描述
PORTP_PIN6_SD_D3	3	SD 卡数据功能引脚
PORTP_PIN7_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN7_FUNMUX	1	数字模块功能引脚
PORTP_PIN7_NORFL_A7	2	NORFLASH 地址线 7
PORTP_PIN7_SD_D2	3	SD 卡数据功能引脚
PORTP_PIN8_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN8_FUNMUX	1	数字模块功能引脚
PORTP_PIN8_NORFL_A8	2	NORFLASH 地址线 8
PORTP_PIN8_SD_D1	3	SD 卡数据功能引脚
PORTP_PIN9_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN9_FUNMUX	1	数字模块功能引脚
PORTP_PIN9_NORFL_A9	2	NORFLASH 地址线 9
PORTP_PIN9_SD_D0	3	SD 卡数据功能引脚
PORTP_PIN10_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN10_FUNMUX	1	数字模块功能引脚
PORTP_PIN10_NORFL_A10	2	NORFLASH 地址线 10
PORTP_PIN10_SD_CMD	3	SD 卡功能命令引脚
PORTP_PIN11_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN11_FUNMUX	1	数字模块功能引脚
PORTP_PIN11_NORFL_A11	2	NORFLASH 地址线 11
PORTP_PIN11_SD_CLK	3	SD 卡时钟功能引脚
PORTP_PIN12_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN12_FUNMUX	1	数字模块功能引脚
PORTP_PIN12_NORFL_A12	2	NORFLASH 地址线 12
PORTP_PIN12_SD_DETECT	3	SD 卡检测功能引脚
PORTP_PIN13_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN13_FUNMUX	1	数字模块功能引脚
PORTP_PIN13_NORFL_A13	2	NORFLASH 地址线 13
PORTP_PIN13_SDRAM_CLK	2	SDRAM 系统时钟输入引脚
PORTP_PIN14_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN14_FUNMUX	1	数字模块功能引脚
PORTP_PIN14_NORFL_A14	2	NORFLASH 地址线 14
PORTP_PIN14_SDRAM_CAS	2	SDRAM 列有效功能选择
PORTP_PIN15_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN15_FUNMUX	1	数字模块功能引脚
PORTP_PIN15_NORFL_A15	2	NORFLASH 地址线 15
PORTP_PIN15_SDRAM_RAS	2	SDRAM 行有效功能选择
PORTP_PIN16_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN16_FUNMUX	1	数字模块功能引脚
PORTP_PIN16_NORFL_A16	2	NORFLASH 地址线 16
PORTP_PIN16_SDRAM_LDQ	2	SDRAM 低字节使能引脚
PORTP_PIN17_GPIO	0	数字 GPIO 功能引脚

常量名	值	描述
PORTP_PIN17_FUNMUX	1	数字模块功能引脚
PORTP_PIN17_NORFL_A17	2	NORFLASH 地址线 17
PORTP_PIN17_SDRAM_UDQ	2	SDRAM 高字节使能引脚
PORTP_PIN18_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN18_FUNMUX	1	数字模块功能引脚
PORTP_PIN18_NORFL_A18	2	NORFLASH 地址线 18
PORTP_PIN19_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN19_FUNMUX	1	数字模块功能引脚
PORTP_PIN19_NORFL_A19	2	NORFLASH 地址线 19
PORTP_PIN20_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN20_FUNMUX	1	数字模块功能引脚
PORTP_PIN20_NORFL_A20	2	NORFLASH 地址线 20
PORTP_PIN20_SDRAM_BA0	2	SDRAM Bank 选择引脚
PORTP_PIN21_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN21_FUNMUX	1	数字模块功能引脚
PORTP_PIN21_NORFL_A21	2	NORFLASH 地址线 21
PORTP_PIN21_SDRAM_BA1	2	SDRAM Bank 选择引脚
PORTP_PIN22_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN22_FUNMUX	1	数字模块功能引脚
PORTP_PIN22_NORFL_A22	2	NORFLASH 地址线 22
PORTP_PIN23_GPIO	0	数字 GPIO 功能引脚
PORTP_PIN23_FUNMUX	1	数字模块功能引脚
PORTP_PIN23_NORFL_A23	2	NORFLASH 地址线 23
FUNMUX0_UART0_RXD	100	UART0 接收数据引脚
FUNMUX0_UART1_RXD	101	UART1 接收数据引脚
FUNMUX0_UART2_RXD	102	UART2 接收数据引脚
FUNMUX0_UART3_RXD	103	UART3 接收数据引脚
FUNMUX0_I2C0_SCL	105	I2C0 模块的时钟引脚
FUNMUX0_I2C1_SCL	106	I2C1 模块的时钟引脚
FUNMUX0_PWM0A_OUT	107	PWM 模块第 0 组 A 路输出引脚
FUNMUX0_PWM2A_OUT	108	PWM 模块第 2 组 A 路输出引脚
FUNMUX0_PWM4A_OUT	109	PWM 模块第 4 组 A 路输出引脚
FUNMUX0_PWM0B_OUT	110	PWM 模块第 0 组 B 路输出引脚
FUNMUX0_PWM2B_OUT	111	PWM 模块第 2 组 B 路输出引脚
FUNMUX0_PWM4B_OUT	112	PWM 模块第 4 组 B 路输出引脚
FUNMUX0_PWM_BREAK	113	PWM 模块刹车信号
FUNMUX0_TIMR0_IN	114	TIMER0 模块输入捕获引脚
FUNMUX0_TIMR2_IN	115	TIMER2 模块输入捕获引脚
FUNMUX0_CAN_RX	116	CAN 模块接收信号
FUNMUX0_SPI0_SSEL	117	SPI0 模块的片选功能引脚
FUNMUX0_SPI0_MOSI	118	SPI0 模块的主机发送功能引脚
FUNMUX0_SPI1_SSEL	119	SPI1 模块的片选功能引脚

常量名	值	描述
FUNMUX0_SPI1_MOSI	120	SPI1 模块的主机发送功能引脚
FUNMUX0_UART0_CTS	121	UART0 发送流控功能引脚
FUNMUX0_UART1_CTS	122	UART1 发送流控功能引脚
FUNMUX0_UART2_CTS	123	UART2 发送流控功能引脚
FUNMUX0_UART3_CTS	124	UART3 发送流控功能引脚
FUNMUX1_UART0_TXD	100	UART0 发送数据引脚
FUNMUX1_UART1_TXD	101	UART1 发送数据引脚
FUNMUX1_UART2_TXD	102	UART2 发送数据引脚
FUNMUX1_UART3_TXD	103	UART3 发送数据引脚
FUNMUX1_I2C0_SDA	105	I2C0 模块的数据引脚
FUNMUX1_I2C1_SDA	106	I2C1 模块的数据引脚
FUNMUX1_PWM1A_OUT	107	PWM 模块第 1 组 A 路输出引脚
FUNMUX1_PWM3A_OUT	108	PWM 模块第 3 组 A 路输出引脚
FUNMUX1_PWM5A_OUT	109	PWM 模块第 5 组 A 路输出引脚
FUNMUX1_PWM1B_OUT	110	PWM 模块第 1 组 B 路输出引脚
FUNMUX1_PWM3B_OUT	111	PWM 模块第 3 组 B 路输出引脚
FUNMUX1_PWM5B_OUT	112	PWM 模块第 5 组 B 路输出引脚
FUNMUX1_PULSE_IN	113	脉冲捕获功能引脚
FUNMUX1_TIMR1_IN	114	TIMER1 模块输入捕获引脚
FUNMUX1_TIMR3_IN	115	TIMER3 模块输入捕获引脚
FUNMUX1_CAN_TX	116	CAN 模块发送信号
FUNMUX1_SPI0_SCLK	117	SPI0 模块的时钟线引脚
FUNMUX1_SPI0_MISO	118	SPI0 模块的主机接收功能引脚
FUNMUX1_SPI1_SCLK	119	SPI1 模块的时钟线引脚
FUNMUX1_SPI1_MISO	120	SPI1 模块的主机接收功能引脚
FUNMUX1_UART0_RTS	121	UART0 接收流控功能引脚
FUNMUX1_UART1_RTS	122	UART1 接收流控功能引脚
FUNMUX1_UART2_RTS	123	UART2 接收流控功能引脚
FUNMUX1_UART3_RTS	124	UART3 接收流控功能引脚

2.2.4 函数

PORT_Init

原型：void PORT_Init(PORT_TypeDef * PORTx, uint32_t n, uint32_t func, uint32_t digit_in_en)

```
/*****************************************************************************
* 函数名称: PORT_Init()
* 功能说明:     端口引脚功能选择，可用的功能见"SWM3200_port.h"文件
* 输     入: uint32_t PORTx          指定 PORT 端口，PORTA、PORTB、PORTC、PORTM、PORTN、PORTP
*           uint32_t n              指定 PORT 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
*           uint32_t func           指定端口引脚要设定的功能，其可取值见
```

"SWM3200_port.h"文件
* uint32_t digit_in_en 数字输入使能
* 输 出: 无
* 注意事项: 当引脚标号 n 为偶数时，func 取值只能是 FUNMUX0 开头的，如 FUNMUX0_UART0_RXD
* 当引脚标号 n 为奇数时，func 取值只能是 FUNMUX1 开头的，如 FUNMUX1_UART0_TXD
*****/

2.3 通用 I/O（GPIO）

2.3.1 介绍

通用输入输出模块主要功能包括数据控制、中断控制功能。使用前需使能对应 GPIO 模块时钟。

2.3.2 特性

- 最高 100 个独立 IO
- 每个 IO 均可触发中断
- 中断触发条件可配置，支持电平触发/沿触发
- 沿触发中断可配置为上升沿/下降沿/双沿触发
- 每个 IO 均支持上拉/下拉/开漏功能

2.3.3 函数

GPIO_Init

原型：void GPIO_Init(GPIO_TypeDef * GPIOx, uint32_t n, uint32_t dir, uint32_t pull_up, uint32_t pull_down)

```
/*****************************************************************************
* 函数名称: GPIO_Init()
* 功能说明:  引脚初始化，包含引脚方向、上拉电阻、下拉电阻、开漏输出
* 输  入: GPIO_TypeDef * GPIOx  指定 GPIO 端口，有效值包括 GPIOA、GPIOB、GPIOC、GPIOM、GPION、GPIOP
*  uint32_t n  指定 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
*  uint32_t dir  引脚方向，0 输入  1 输出
*  uint32_t pull_up  上拉电阻，0 关闭上拉  1 开启上拉
*  uint32_t pull_down  下拉电阻，0 关闭下拉  1 开启下拉
* 输  出: 无
* 注意事项: 无
```

**********/

GPIO_SetBit

原型：void GPIO_SetBit(GPIO_TypeDef * GPIOx, uint32_t n)

```
/*****************************************************************************
* 函数名称: GPIO_SetBit()
* 功能说明:     将参数指定的引脚电平置高
* 输    入: GPIO_TypeDef * GPIOx    指示 GPIO 端口，有效值包括 GPIOA、GPIOB、GPIOC、GPIOM、GPION、GPIOP
*           uint32_t n          指定 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

GPIO_ClrBit

原型：void GPIO_ClrBit(GPIO_TypeDef * GPIOx, uint32_t n)

```
/*****************************************************************************
* 函数名称:     GPIO_ClrBit()
* 功能说明:     将参数指定的引脚电平置低
* 输    入: GPIO_TypeDef * GPIOx    指定 GPIO 端口，有效值包括 GPIOA、GPIOB、GPIOC、GPIOM、GPION、GPIOP
*           uint32_t n          指定 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

GPIO_InvBit

原型：void GPIO_InvBit(GPIO_TypeDef * GPIOx, uint32_t n)

```
/*****************************************************************************
* 函数名称: GPIO_InvBit()
* 功能说明:     将参数指定的引脚电平反转
* 输    入: GPIO_TypeDef * GPIOx    指定 GPIO 端口，有效值包括 GPIOA、GPIOB、GPIOC、GPIOM、GPION、GPIOP
*           uint32_t n          指定 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

GPIO_GetBit

原型：uint32_t GPIO_GetBit(GPIO_TypeDef * GPIOx, uint32_t n)

```
/*****************************************************************************
**********
```

```
* 函数名称:      GPIO_GetBit()
* 功能说明:      读取参数指定的引脚的电平状态
* 输      入: GPIO_TypeDef * GPIOx          指定 GPIO 端口，有效值包括 GPIOA、GPIOB、
GPIOC、GPIOM、GPION、GPIOP
*                uint32_t n                 指定 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ...
PIN22、PIN23
* 输      出: 参数指定的引脚的电平状态      0 低电平      1 高电平
* 注意事项: 无
*******************************************************************************/
```

GPIO_SetBits

原型：void GPIO_SetBits(GPIO_TypeDef * GPIOx, uint32_t n, uint32_t w)

```
/*******************************************************************************
* 函数名称: GPIO_SetBits()
* 功能说明: 将参数指定的从 n 开始的 w 位连续引脚的电平置高
* 输      入: GPIO_TypeDef * GPIOx          指定 GPIO 端口，有效值包括 GPIOA、GPIOB、
GPIOC、GPIOM、GPION、GPIOP
*                uint32_t n                 指定 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ...
PIN22、PIN23
*                uint32_t w                 指定要将引脚电平置高的引脚的个数
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

GPIO_ClrBits

原型：void GPIO_ClrBits(GPIO_TypeDef * GPIOx, uint32_t n, uint32_t w)

```
/*******************************************************************************
* 函数名称:      GPIO_ClrBits()
* 功能说明: 将参数指定的从 n 开始的 w 位连续引脚的电平置低
* 输      入: GPIO_TypeDef * GPIOx          指定 GPIO 端口，有效值包括 GPIOA、GPIOB、
GPIOC、GPIOM、GPION、GPIOP
*                uint32_t n                 指定 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ...
PIN22、PIN23
*                uint32_t w                 指定要将引脚电平置低的引脚的个数
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

GPIO_InvBits

原型：void GPIO_InvBits(GPIO_TypeDef * GPIOx, uint32_t n, uint32_t w)

```
/*******************************************************************************
*************************************************************
* 函数名称: GPIO_InvBits()
* 功能说明: 将参数指定的从 n 开始的 w 位连续引脚的电平反转
* 输      入: GPIO_TypeDef * GPIOx          指定 GPIO 端口，有效值包括 GPIOA、GPIOB、
```

GPIOC、GPIOM、GPION、GPIOP
* uint32_t n 指定 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
* uint32_t w 指定要将引脚电平反转的引脚的个数
* 输 出: 无
* 注意事项: 无
***/

GPIO_GetBits

原型：uint32_t GPIO_GetBits(GPIO_TypeDef * GPIOx, uint32_t n, uint32_t w)

```
/*******************************************************************************
*************************************************************
* 函数名称:  GPIO_GetBits()
* 功能说明: 读取参数指定的从 n 开始的 w 位连续引脚的电平状态
* 输  入: GPIO_TypeDef * GPIOx  指定 GPIO 端口，有效值包括 GPIOA、GPIOB、GPIOC、GPIOM、GPION、GPIOP
*  uint32_t n  指定 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
*  uint32_t w  指定要将引脚电平置高的引脚的个数
* 输  出: 参数指定的从 n 开始的 w 位连续引脚的电平状态  0 低电平  1 高电平
*  返回值的第 0 位表示引脚 n 的电平状态、返回值的第 1 位表示引脚 n+1 的电平状态... ...返回值的第 w 位表示引脚 n+w 的电平状态
* 注意事项: 无
*******************************************************************************/
```

2.4 外部中断（EXTI）

2.4.1 常量定义

常量名	值	描述
EXTI_FALL_EDGE	0x00	下降沿触发中断
EXTI_RISE_EDGE	0x01	上升沿触发中断
EXTI_BOTH_EDGE	0x02	双边沿触发中断
EXTI_LOW_LEVEL	0x10	低电平触发中断
EXTI_HIGH_LEVEL	0x11	高电平触发中断

2.4.2 函数

EXTI_Init

原型：void EXTI_Init(GPIO_TypeDef * GPIOx, uint32_t n, uint32_t mode)

```
/*******************************************************************************
* 函数名称: EXTI_Init()
* 功能说明:  指定引脚外部中断初始化
```

* 输　　入: GPIO_TypeDef * GPIOx　　指定产生外部中断的 GPIO 端口，有效值包括 GPIOA、GPIOB、GPIOC、GPIOM、GPION、GPIOP
*　　uint32_t n　　指定产生外部中断的 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
*　　uint32_t mode　　有效值有 EXTI_FALL_EDGE、EXTI_RISE_EDGE、EXTI_BOTH_EDGE、EXTI_LOW_LEVEL、EXTI_HIGH_LEVEL
* 输　　出: 无
* 注意事项: 由于 GPIOA、GPIOB、GPIOC、GPIOM 的 PIN0--7 引脚即可以接入 NVIC 中的引脚中断（如 GPIOA0_IRQn），也可以接入 NVIC 的组中断（GPIOA_IRQn），所以不在此函数中调用 NVIC_EnableIRQ() 使能 NVIC 中断，从而可以根据需要调用 NVIC_EnableIRQ(GPIOA0_IRQn)和 NVIC_EnableIRQ(GPIOA_IRQn)
***/

EXTI_Open

原型：void EXTI_Open(GPIO_TypeDef * GPIOx, uint32_t n)

```
/*****************************************************************************
* 函数名称:　　EXTI_Open()
* 功能说明:　　指定引脚外部中断打开（即使能）
* 输　　入: GPIO_TypeDef * GPIOx　　指定产生外部中断的 GPIO 端口，有效值包括 GPIOA、GPIOB、GPIOC、GPIOM、GPION、GPIOP
*　　uint32_t n　　指定产生外部中断的 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
* 输　　出: 无* 注意事项: 无
*****************************************************************************/
```

EXTI_Close

原型：void EXTI_Close(GPIO_TypeDef * GPIOx, uint32_t n)

```
/*****************************************************************************
* 函数名称:　　EXTI_Close()
* 功能说明:　　指定引脚外部中断关闭（即禁能）
* 输　　入: GPIO_TypeDef * GPIOx　　指定产生外部中断的 GPIO 端口，有效值包括 GPIOA、GPIOB、GPIOC、GPIOM、GPION、GPIOP
*　　uint32_t n　　指定产生外部中断的 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
* 输　　出: 无
* 注意事项: 无
*****************************************************************************/
```

EXTI_State

原型：uint32_t EXTI_State(GPIO_TypeDef * GPIOx, uint32_t n)

```
/*****************************************************************************
* 函数名称:　　EXTI_State()
* 功能说明:　　指定引脚是否触发了中断
* 输　　入: GPIO_TypeDef * GPIOx　　指定产生外部中断的 GPIO 端口，有效值包括 GPIOA、GPIOB、GPIOC、GPIOM、GPION、GPIOP
```

```
*                uint32_t n                  指定产生外部中断的 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
* 输      出: uint32_t      1 此引脚触发了中断      0 此引脚未触发中断
* 注意事项: 无
*******************************************************************************/
```

EXTI_RawState

原型：uint32_t EXTI_RawState(GPIO_TypeDef * GPIOx, uint32_t n)

```
/*******************************************************************************
* 函数名称:      EXTI_RawState()
* 功能说明:      指定引脚是否满足过/了中断触发条件，当此中断关闭时可通过调用此函数以查询的方式检测引脚上是否满足过/了中断触发条件
* 输      入: GPIO_TypeDef * GPIOx      指定产生外部中断的 GPIO 端口，有效值包括 GPIOA、GPIOB、GPIOC、GPIOM、GPION、GPIOP
*                uint32_t n                  指定产生外部中断的 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
* 输      出: uint32_t      1 此引脚满足过/了中断触发条件      0 此引脚未满足过/了中断触发条件
* 注意事项: 无
*******************************************************************************/
```

EXTI_Clear

原型：void EXTI_Clear(GPIO_TypeDef * GPIOx, uint32_t n)

```
/*******************************************************************************
* 函数名称:      EXTI_Clear()
* 功能说明:      指定引脚外部中断清除（即清除中断标志，以免再次进入此中断）
* 输      入: GPIO_TypeDef * GPIOx      指定产生外部中断的 GPIO 端口，有效值包括 GPIOA、GPIOB、GPIOC、GPIOM、GPION、GPIOP
*                uint32_t n                  指定产生外部中断的 GPIO 引脚，有效值包括 PIN0、PIN1、PIN2、... ... PIN22、PIN23
* 输      出: 无
* 注意事项: 只能清除边沿触发中断的标志，电平触发中断的标志无法清除，只能在引脚电平不符合中断触发条件后硬件自动清除
*******************************************************************************/
```

2.5 加强型定时器（TIMER）

2.5.1 介绍

SWM320 系列所有型号 TIMER 操作均相同。使用前需使能 TIMER 模块时钟。

每个 TIMER 模块均具备定时器功能（使用片内时钟作为计数基准）和计数器功能（使用片外时钟作为计数基准）。

6 路 TIMER 模块支持级联操作，TIMER1 可使用 TIMER0 溢出作为计数源，扩展计数周期，以此类推，即最高可支持 192bit 位宽定时器。

1 路 32 位脉宽捕捉计数器，针对外部输入信号实现捕捉功能。

2.5.2 特性

- 6 路 32 位通用定时器
- 1 路 32 位脉冲宽度测量计数器
- 可单独配置计时触发条件为内部时钟或者外部输入
- 检测脉冲极性可配
- 每路中段可以单独使能
- 支持级联功能，最高支持 192bit 定时器

2.5.3 常量定义

常量名	值	描述
TIMR_MODE_TIMER	0	定时器模式
TIMR_MODE_COUNTER	1	计数器模式

2.5.4 函数

TIMR_Init

原型：void TIMR_Init(TIMR_TypeDef * TIMRx, uint32_t mode, uint32_t period, uint32_t int_en)

```
/*****************************************************************************
* 函数名称: TIMR_Init()
* 功能说明:     TIMR 定时器/计数器初始化
* 输    入: TIMR_TypeDef * TIMRx     指定要被设置的定时器，有效值包括 TIMR0、TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
*           uint32_t mode           TIMR_MODE_TIMER  定时器模式
TIMR_MODE_COUNTER  计数器模式
*           uint32_t period          定时/计数周期
*           uint32_t int_en          中断使能
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

TIMR_Start

原型：void TIMR_Start(TIMR_TypeDef * TIMRx)

```
/*****************************************************************************
* 函数名称:     TIMR_Start()
* 功能说明:     启动定时器，从初始值开始计时/计数
* 输    入: TIMR_TypeDef * TIMRx     指定要被设置的定时器，可取值包括 TIMR0、
```

TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
* 输　　出: 无
* 注意事项: 无
***/

TIMR_Stop

原型：void TIMR_Stop(TIMR_TypeDef * TIMRx)

```
/*****************************************************************************
* 函数名称:      TIMR_Stop()
* 功能说明:      停止定时器
* 输　　入: TIMR_TypeDef * TIMRx      指定要被设置的定时器，可取值包括 TIMR0、
TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
* 输　　出: 无
* 注意事项: 无
*****************************************************************************/
```

TIMR_Halt

原型：void TIMR_Halt(TIMR_TypeDef * TIMRx)

```
/*****************************************************************************
* 函数名称:      TIMR_Halt()
* 功能说明:      暂停定时器，计数值保持不变
* 输　　入: TIMR_TypeDef * TIMRx      指定要被设置的定时器，可取值包括 TIMR0、
TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
* 输　　出: 无
* 注意事项: 无
*****************************************************************************/
```

TIMR_Resume

原型：void TIMR_Resume(TIMR_TypeDef * TIMRx)

```
/*****************************************************************************
* 函数名称:      TIMR_Resume()
* 功能说明:      恢复定时器，从暂停处继续计数
* 输　　入: TIMR_TypeDef * TIMRx      指定要被设置的定时器，可取值包括 TIMR0、
TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
* 输　　出: 无
* 注意事项: 无
*****************************************************************************/
```

TIMR_SetPeriod

原型：void TIMR_SetPeriod(TIMR_TypeDef * TIMRx, uint32_t period)

```
/*****************************************************************************
* 函数名称: TIMR_SetPeriod()
* 功能说明:      设置定时/计数周期
* 输　　入: TIMR_TypeDef * TIMRx      指定要被设置的定时器，可取值包括 TIMR0、
TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
```

```
*                   uint32_t period                 定时/计数周期
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

TIMR_GetPeriod

原型：uint32_t TIMR_GetPeriod(TIMR_TypeDef * TIMRx)

```
/*******************************************************************************
* 函数名称: TIMR_GetPeriod()
* 功能说明:     获取定时/计数周期
* 输      入: TIMR_TypeDef * TIMRx      指定要被设置的定时器，可取值包括 TIMR0、TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
* 输      出: uint32_t                   当前定时/计数周期
* 注意事项: 无
*******************************************************************************/
```

TIMR_GetCurValue

原型：uint32_t TIMR_GetCurValue(TIMR_TypeDef * TIMRx)

```
/*******************************************************************************
* 函数名称:     TIMR_GetCurValue()
* 功能说明:     获取当前计数值
* 输      入: TIMR_TypeDef * TIMRx      指定要被设置的定时器，可取值包括 TIMR0、TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
* 输      出: uint32_t                   当前计数值
* 注意事项: 无
*******************************************************************************/
```

TIMR_INTEn

原型：void TIMR_INTEn(TIMR_TypeDef * TIMRx)

```
/*******************************************************************************
* 函数名称:     TIMR_INTEn()
* 功能说明:     使能中断
* 输      入: TIMR_TypeDef * TIMRx      指定要被设置的定时器，可取值包括 TIMR0、TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

TIMR_INTDis

原型：void TIMR_INTDis(TIMR_TypeDef * TIMRx)

```
/*******************************************************************************
* 函数名称: TIMR_INTDis()
* 功能说明:     禁能中断
* 输      入: TIMR_TypeDef * TIMRx      指定要被设置的定时器，可取值包括 TIMR0、TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
```

```
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

TIMR_INTClr

原型：void TIMR_INTClr(TIMR_TypeDef * TIMRx)

```
/*******************************************************************************
* 函数名称:      TIMR_INTClr()
* 功能说明:      清除中断标志
* 输      入: TIMR_TypeDef * TIMRx      指定要被设置的定时器，可取值包括 TIMR0、TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

TIMR_INTStat

原型：uint32_t TIMR_INTStat(TIMR_TypeDef * TIMRx)

```
/*******************************************************************************
* 函数名称: TIMR_INTStat()
* 功能说明:      获取中断状态
* 输      入: TIMR_TypeDef * TIMRx      指定要被设置的定时器，可取值包括 TIMR0、TIMR1、TIMR2、TIMR3、TIMR4、TIMR5
* 输      出: uint32_t                  0 TIMRx 未产生中断      1 TIMRx 产生了中断
* 注意事项: 无
*******************************************************************************/
```

2.6 看门狗定时器（WDT）

2.6.1 介绍

看门狗定时器（WDT）主要用于控制程序流程正确。SWM320 系列所有型号 WDT 操作均相同。使用前需使能对应 WDT 模块时钟。

2.6.2 特性

- 产生计数器溢出复位信号，复位信号使能可配置
- 具有 32 位计数位宽，可灵活配置宽范围的溢出周期
- 具有中断功能

2.6.3 常量定义

常量名	值	描述
WDT_MODE_RESET	0	计数器减小到 0 时产生复位
WDT_MODE_INTERRUPT	1	计数器减小到 LOAD/4 时产生中断

2.6.4 函数

WDT_Init

原型：void WDT_Init(WDT_TypeDef * WDTx, uint32_t peroid, uint32_t mode)

```
/*****************************************************************************
* 函数名称:      WDT_Init()
* 功能说明:      WDT 看门狗初始化
* 输      入: WDT_TypeDef * WDTx          指定要被设置的看门狗，有效值包括 WDT
*             uint32_t peroid             取值 0--4294967295，单位为单片机系统时钟周期
*             uint32_t mode               WDT_MODE_RESET  超时产生复位
WDT_MODE_INTERRUPT  超时产生中断
* 输      出: 无
* 注意事项: 复位使能时中断不起作用，因为计数周期结束时芯片直接复位了，无法响应
中断
*****************************************************************************/
```

WDT_Start

原型：void WDT_Start(WDT_TypeDef * WDTx)

```
/*****************************************************************************
* 函数名称:      WDT_Start()
* 功能说明:      启动指定 WDT，开始倒计时
* 输      入: WDT_TypeDef * WDTx          指定要被设置的看门狗，有效值包括 WDT
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

WDT_Stop

原型：void WDT_Stop(WDT_TypeDef * WDTx)

```
/*****************************************************************************
* 函数名称:      WDT_Stop()
* 功能说明:      关闭指定 WDT，停止倒计时
* 输      入: WDT_TypeDef * WDTx          指定要被设置的看门狗，有效值包括 WDT
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

WDT_Feed

原型：void WDT_Feed(WDT_TypeDef * WDTx)

```
/*****************************************************************************
* 函数名称:      WDT_Feed()
* 功能说明:      喂狗，重新从装载值开始倒计时
* 输      入: WDT_TypeDef * WDTx          指定要被设置的看门狗，有效值包括 WDT
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

WDT_GetValue

原型：int32_t WDT_GetValue(WDT_TypeDef * WDTx)

```
/*****************************************************************************
* 函数名称:      WDT_GetValue()
* 功能说明:      获取指定看门狗定时器的当前倒计时值
* 输      入: WDT_TypeDef * WDTx          指定要被设置的看门狗，有效值包括 WDT
* 输      出: int32_t                     看门狗当前计数值
* 注意事项: 无
*****************************************************************************/
```

WDT_INTClr

原型：void WDT_INTClr(WDT_TypeDef * WDTx)

```
/*****************************************************************************
* 函数名称:      WDT_INTClr()
* 功能说明:      中断标志清除
* 输      入: WDT_TypeDef * WDTx          指定要被设置的看门狗，有效值包括 WDT
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

WDT_INTStat

原型：uint32_t WDT_INTStat(WDT_TypeDef * WDTx)

```
/*****************************************************************************
*************************************************************
* 函数名称:      WDT_INTStat()
* 功能说明:      中断状态查询
* 输      入: WDT_TypeDef * WDTx          指定要被设置的看门狗，有效值包括 WDT
* 输      出: int32_t                     1 发生中断溢出      0 未发生中断溢出
* 注意事项: 无
*****************************************************************************/
```

2.7 UART 接口控制器（UART）

2.7.1 介绍

不同型号具备 UART 数量可能不同。使用前需使能对应 UART 模块时钟。

UART 模块支持波特率配置，最高速度可达到模块时钟 16 分频。具备深度为 9 的 FIFO，同时提供了多种中断供选择。

2.7.2 特性

- 支持标准的 UART 协议
- 支持全双工模式
- 支持波特率可配置
- 支持 8 位/9 位数据格式选择
- 可配置的奇偶校验位
- 支持 1 位/2 位停止位选择
- 支持波特率自动调整
- 深度为 9 字节的发送和接收 FIFO
- 支持 break 操作自动检测
- 支持接收超时中断
- 支持 LIN 模式
- 支持自动流控功能

2.7.3 常量定义

常量名	值	描述
UART_DATA_8BIT	0	8 位数据位
UART_DATA_9BIT	1	9 位数据位
常量名	值	描述
UART_PARITY_NONE	0	无校验位
UART_PARITY_ODD	1	奇校验位
UART_PARITY_EVEN	3	偶校验位
UART_PARITY_ONE	5	校验位常为 1
UART_PARITY_ZERO	7	校验位常为 0
常量名	值	描述
UART_STOP_1BIT	0	1 位停止位
UART_STOP_2BIT	1	2 位停止位
常量名	值	描述
UART_RTS_1BYTE	0	当接收 FIFO 中剩下 1 个空位时，将 RTS 拉高
UART_RTS_2BYTE	1	当接收 FIFO 中剩下 2 个空位时，将 RTS 拉高

UART_RTS_4BYTE	2	当接收 FIFO 中剩下 4 个空位时，将 RTS 拉高
UART_RTS_6BYTE	3	当接收 FIFO 中剩下 6 个空位时，将 RTS 拉高
UART_ABR_RES_OK	1	自动波特率校准成功
UART_ABR_RES_ERR	2	自动波特率校准失败

2.7.4 类型定义

UART_InitStructure

```
typedef struct {
    uint32_t Baudrate;

    uint8_t  DataBits;

    uint8_t  Parity;

    uint8_t  StopBits;

    uint8_t  RXThreshold;
    uint8_t  RXThresholdIEn;

    uint8_t  TXThreshold;
    uint8_t  TXThresholdIEn;

    uint8_t  TimeoutTime;
    uint8_t  TimeoutIEn;
} UART_InitStructure;
```

成员	描述
DataBits	数据位位数，可取值 UART_DATA_8BIT、UART_DATA_9BIT
Parity	奇偶校验位，可取值 UART_PARITY_NONE、UART_PARITY_ON、UART_PARITY_EVEN、UART_PARITY_ON、UART_PARITY_ZERO
StopBits	停止位位数，可取值 UART_STOP_1BIT、UART_STOP_2BIT
RXThreshold	取值 0--7
RXThresholdIEn	当 RX FIFO 中数据个数 >= RXThreshold 时触发中断
TXThreshold	取值 0--7

TXThresholdIEn	当 TX FIFO 中数据个数 <= TXThreshold 时触发中断
TimeoutTime	超时时长 = TimeoutTime/(Baudrate/10) 秒
TimeoutIEn	超时中断，超过 TimeoutTime/(Baudrate/10) 秒没有在 RX 线上接收到数据时触发中断

2.7.5 函数

UART_Init

原型：void UART_Init(UART_TypeDef * UARTx, UART_InitStructure * initStruct)

```
/*****************************************************************************
* 函数名称:      UART_Init()
* 功能说明:      UART 串口初始化
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
*               UART_InitStructure * initStruct      包含 UART 串口相关设定值的结构体
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

UART_Open

原型：void UART_Open(UART_TypeDef * UARTx)

```
/*****************************************************************************
* 函数名称:      UART_Open()
* 功能说明:      UART 串口打开
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

UART_Close

原型：void UART_Close(UART_TypeDef * UARTx)

```
/*****************************************************************************
* 函数名称:      UART_Close()
* 功能说明:      UART 串口关闭
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

UART_WriteByte

原型：void UART_WriteByte(UART_TypeDef * UARTx, uint8_t data)

```
/*****************************************************************************
* 函数名称:      UART_WriteByte()
* 功能说明:      发送一个字节数据
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，可取值包括 UART0、UART1、UART2、UART3、UART4
*                uint8_t data             要发送的字节
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

UART_ReadByte

原型：uint32_t UART_ReadByte(UART_TypeDef * UARTx, uint32_t * data)

```
/*****************************************************************************
* 函数名称:      UART_ReadByte()
* 功能说明:      读取一个字节数据，并指出数据是否 Valid
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，可取值包括 UART0、UART1、UART2、UART3、UART4
*           uint32_t * data             接收到的数据
* 输      出: uint32_t                  1 数据 Valid      0 数据错误（帧错误、奇偶校验错误）
* 注意事项: 无
*****************************************************************************/
```

UART_IsTXBusy

原型：uint32_t UART_IsTXBusy(UART_TypeDef * UARTx)

```
/*****************************************************************************
* 函数名称:      UART_IsTXBusy()
* 功能说明:      UART 是否正在发送数据
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输      出: uint32_t                  1 UART 正在发送数据      0 数据已发完
* 注意事项: 无
*****************************************************************************/
```

UART_IsRXFIFOEmpty

原型：uint32_t UART_IsRXFIFOEmpty(UART_TypeDef * UARTx)

```
/*****************************************************************************
* 函数名称:      UART_IsRXFIFOEmpty()
* 功能说明:      接收 FIFO 是否为空，如果不空则说明其中有数据可以读取
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输      出: uint32_t                  1 接收 FIFO 空      0 接收 FIFO 非空
* 注意事项: 无
*****************************************************************************/
```

UART_IsTXFIFOFull

原型：uint32_t UART_IsTXFIFOFull(UART_TypeDef * UARTx)

```
/****************************************************************************************************
* 函数名称:      UART_IsTXFIFOFull()
* 功能说明:      发送 FIFO 是否为满，如果不满则可以继续向其中写入数据
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输    出: uint32_t                  1 发送 FIFO 满      0 发送 FIFO 不满
* 注意事项: 无
****************************************************************************************************/
```

UART_SetBaudrate

原型：void UART_SetBaudrate(UART_TypeDef * UARTx, uint32_t baudrate)

```
/****************************************************************************************************
* 函数名称:      UART_SetBaudrate()
* 功能说明:      设置波特率
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
*           uint32_t baudrate         要设置的波特率
* 输    出: 无
* 注意事项: 不要在串口工作时更改波特率，使用此函数前请先调用 UART_Close()关闭串口
****************************************************************************************************/
```

UART_GetBaudrate

原型：uint32_t UART_GetBaudrate(UART_TypeDef * UARTx)

```
/****************************************************************************************************
* 函数名称:      UART_GetBaudrate()
* 功能说明:      查询波特率
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输    出: uint32_t                  当前波特率
* 注意事项: 无
****************************************************************************************************/
```

UART_ CTSConfig

原型：void UART_CTSConfig(UART_TypeDef * UARTx, uint32_t enable, uint32_t polarity)

```
/****************************************************************************************************
* 函数名称:      UART_CTSConfig()
* 功能说明:      UART CTS 流控配置
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
*           uint32_t enable           1 使能 CTS 流控      0 禁止 CTS 流控
```

*　　uint32_t polarity　　0 CTS 输入为低表示可以发送数据　　1 CTS 输入为高表示可以发送数据
* 输　　出: 无
* 注意事项: 无
***/

UART_ CTSLineState

原型：uint32_t UART_CTSLineState(UART_TypeDef * UARTx)

```
/*******************************************************************************
* 函数名称:　　UART_CTSLineState()
* 功能说明:　　UART CTS 线当前状态
* 输　　入: UART_TypeDef * UARTx　　指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输　　出: uint32_t　　0 CTS 线当前为低电平　　1 CTS 线当前为高电平
* 注意事项: 无
*******************************************************************************/
```

UART_ RTSConfig

原型：void UART_RTSConfig(UART_TypeDef * UARTx, uint32_t enable, uint32_t polarity, uint32_t threshold)

```
/*******************************************************************************
* 函数名称:　　UART_RTSConfig()
* 功能说明:　　UART RTS 流控配置
* 输　　入: UART_TypeDef * UARTx　　指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
*　　uint32_t enable　　1 使能 RTS 流控　　0 禁止 RTS 流控
*　　uint32_t polarity　　0 RTS 输出低表示可以接收数据　　1 RTS 输出高表示可以接收数据
*　　uint32_t threshold　　RTS 流控的触发阈值，可取值 UART_RTS_1BYTE、UART_RTS_2BYTE、UART_RTS_4BYTE、UART_RTS_6BYTE
* 输　　出: 无
* 注意事项: 无
*******************************************************************************/
```

UART_ RTSLineState

原型：uint32_t UART_RTSLineState(UART_TypeDef * UARTx)

```
/*******************************************************************************
* 函数名称:　　UART_RTSLineState()
* 功能说明:　　UART RTS 线当前状态
* 输　　入: UART_TypeDef * UARTx　　指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输　　出: uint32_t　　0 RTS 线当前为低电平　　1 RTS 线当前为高电平
* 注意事项: 无
*******************************************************************************/
```

UART_ LINConfig

原 型： void UART_LINConfig(UART_TypeDef * UARTx, uint32_t detectedIEn, uint32_t generatedIEn)

```
/*****************************************************************************
* 函数名称:      UART_LINConfig()
* 功能说明:      UART LIN 功能配置
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
*                uint32_t detectedIEn      检测到 Break 中断使能
*                uint32_t generatedIEn     Break 发送完成中断使能
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

UART_ LINGenerate

原型：void UART_LINGenerate(UART_TypeDef * UARTx)

```
/*****************************************************************************
* 函数名称:      UART_LINGenerate()
* 功能说明:      UART LIN 产生/发送 Break
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

UART_ LINIsDetected

原型：uint32_t UART_LINIsDetected(UART_TypeDef * UARTx)

```
/*****************************************************************************
* 函数名称:      UART_LINIsDetected()
* 功能说明:      UART LIN 是否检测到 Break
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输      出: uint32_t                  1 检测到 LIN Break      0 未检测到 LIN Break
* 注意事项: 无
*****************************************************************************/
```

UART_ LINIsGenerated

原型：uint32_t UART_LINIsGenerated(UART_TypeDef * UARTx)

```
/*****************************************************************************
* 函数名称:      UART_LINIsGenerated()
* 功能说明:      UART LIN Break 是否发送完成
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输      出: uint32_t                  1 LIN Break 发送完成      0 LIN Break 发送未完成
```

```
* 注意事项: 无
*******************************************************************************/
```

UART_ABRStart

原型：void UART_ABRStart(UART_TypeDef * UARTx, uint32_t detectChar)

```
/*******************************************************************************
* 函数名称:      UART_ABRStart()
* 功能说明:      UART 自动波特率检测开始
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
*                uint32_t detectChar          用于自动检测、计算波特率的检测字符
*                                             8 位数据时可取值：0xFF、0xFE、0xF8、0x80，分别表示发送方必须发送 0xFF、0xFE、0xF8、0x80
*                                             9 位数据时可取值：0x1FF、0x1FE、0x1F8、0x180，分别表示发送方必须发送 0x1FF、0x1FE、0x1F8、0x180
* 输      出: 无
* 注意事项: 自动波特率检测时不能开启奇偶校验
*******************************************************************************/
```

UART_ABRIsDone

原型：uint32_t UART_ABRIsDone(UART_TypeDef * UARTx)

```
/*******************************************************************************
* 函数名称:      UART_ABRIsDone()
* 功能说明:      UART 自动波特率是否完成
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输      出: uint32_t                  0 未完成      UART_ABR_RES_OK 已完成，且成功      UART_ABR_RES_ERR 已完成，但失败、出错
* 注意事项: 无
*******************************************************************************/
```

UART_INTRXThresholdEn

原型：void UART_INTRXThresholdEn(UART_TypeDef * UARTx)

```
/*******************************************************************************
* 函数名称:      UART_INTRXThresholdEn()
* 功能说明:      当 RX FIFO 中数据个数 >= RXThreshold 时 触发中断
* 输      入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

UART_INTRXThresholdDis

原型：void UART_INTRXThresholdDis(UART_TypeDef * UARTx)

```
/*******************************************************************************
```

```
* 函数名称:      UART_INTRXThresholdDis()
* 功能说明:      当 RX FIFO 中数据个数 >= RXThreshold 时 不触发中断
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

UART_INTRXThresholdStat

原型：uint32_t UART_INTRXThresholdStat(UART_TypeDef * UARTx)

```
/*****************************************************************************
* 函数名称:      UART_INTRXThresholdStat()
* 功能说明:      是否 RX FIFO 中数据个数 >= RXThreshold
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输    出: uint32_t                  1 RX FIFO 中数据个数 >= RXThreshold          0 RX FIFO 中数据个数 < RXThreshold
* 注意事项: RXIF = RXTHRF & RXIE
*****************************************************************************/
```

UART_INTTXThresholdEn

原型：void UART_INTTXThresholdEn(UART_TypeDef * UARTx)

```
/*****************************************************************************
* 函数名称:      UART_INTTXThresholdEn()
* 功能说明:      当 TX FIFO 中数据个数 <= TXThreshold 时 触发中断
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

UART_INTTXThresholdDis

原型：void UART_INTTXThresholdDis(UART_TypeDef * UARTx)

```
/*****************************************************************************
* 函数名称:      UART_INTTXThresholdDis()
* 功能说明:      当 TX FIFO 中数据个数 <= TXThreshold 时 不触发中断
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

UART_INTTXThresholdStat

原型：uint32_t UART_INTTXThresholdStat(UART_TypeDef * UARTx)

```
/*****************************************************************************
```

```
* 函数名称:      UART_INTTXThresholdStat()
* 功能说明:      是否 TX FIFO 中数据个数 <= TXThreshold
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输    出: uint32_t                  1 TX FIFO 中数据个数 <= TXThreshold          0 TX FIFO 中数据个数 > TXThreshold
* 注意事项: TXIF = TXTHRF & TXIE
*******************************************************************************/
```

UART_INTTimeoutEn

原型：void UART_INTTimeoutEn(UART_TypeDef * UARTx)

```
/*******************************************************************************
* 函数名称:      UART_INTTimeoutEn()
* 功能说明:      接收发生超时时 触发中断
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输    出: 无
* 注意事项: 无
*******************************************************************************/
```

UART_INTTimeoutDis

原型：void UART_INTTimeoutDis(UART_TypeDef * UARTx)

```
/*******************************************************************************
* 函数名称:      UART_INTTimeoutDis()
* 功能说明:      接收发生超时时 不触发中断
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输    出: 无
* 注意事项: 无
*******************************************************************************/
```

UART_INTTimeoutStat

原型：uint32_t UART_INTTimeoutStat(UART_TypeDef * UARTx)

```
/*******************************************************************************
* 函数名称:      UART_INTTimeoutStat()
* 功能说明:      是否发生了接收超时，即超过 TimeoutTime/(Baudrate/10) 秒没有在 RX 线上接收到数据时触发中断
* 输    入: UART_TypeDef * UARTx      指定要被设置的 UART 串口，有效值包括 UART0、UART1、UART2、UART3
* 输    出: uint32_t                  1 发生了接收超时          0 未发生接收超时
* 注意事项: 无
*******************************************************************************/
```

2.8 I2C 总线控制器（主/从）

2.8.1 介绍

I2C 总线采用串行数据线(SDA)和串行时钟线(SCL)传输数据。 I2C 总线的设备端口为开漏输出，所以必须在接口外接上拉电阻。不同型号 I2C 模块数量可能不同。使用前需使能对应 I2C 模块时钟。

2.8.2 特性

- 支持主机/从机模式
- 支持 7 位或 10 位地址
- 波特率可配置
- 支持中断功能

2.8.3 类型定义

I2C_InitStructure

```
typedef struct {
    uint8_t  Master;
    uint8_t  Addr7b;

    uint32_t MstClk;
    uint8_t  MstIEn;
} I2C_InitStructure;
```

成员	描述
Master	1：主机模式，0：从机模式
Addr7b	1：7 位地址，0：10 位地址
MstClk	主机传输时钟频率
MstIEn	主机模式中断使能

2.8.4 函数

I2C_Init

原型：void I2C_Init(I2C_TypeDef * I2Cx, I2C_InitStructure * initStruct)

```
/*****************************************************************************
* 函数名称:      I2C_Init()
* 功能说明:      I2C 初始化
* 输    入: I2C_TypeDef * I2Cx          指定要被设置的 I2C，有效值包括 I2C0、I2C1
*           I2C_InitStructure * initStruct    包含 I2C 相关设定值的结构体
* 输    出: 无
```

```
* 注意事项: 模块只能工作于主机模式
*******************************************************************************/
```

I2C_Open

原型：void I2C_Open(I2C_TypeDef * I2Cx)

```
/*******************************************************************************
* 函数名称:      I2C_Open()
* 功能说明:      I2C 打开，允许收发
* 输      入: I2C_TypeDef * I2Cx          指定要被设置的 I2C，有效值包括 I2C0、I2C1
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

I2C_Close

原型：void I2C_Close(I2C_TypeDef * I2Cx)

```
/*******************************************************************************
* 函数名称:      I2C_Close()
* 功能说明:      I2C 关闭，禁止收发
* 输      入: I2C_TypeDef * I2Cx          指定要被设置的 I2C，有效值包括 I2C0、I2C1
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

2.9 串行外设接口（SPI）控制器

2.9.1 介绍

不同型号 SPI 模块数量可能不同。使用前需使能对应 SPI 模块时钟。

SPI 模块支持 SPI 模式及 SSI 模式。 SPI 模式下支持 MASTER 模式及 SLAVE 模式。具备深度为 8 的 FIFO，速率及帧宽度可灵活配置。

2.9.2 特性

- 全双工串行同步收发
- 可编程时钟极性和相位
- 支持 MASTER 模式和 SLAVE 模式
- MASTER 模式下最高传输速度支持主时钟 2 分频
- 数据宽度支持 4BIT 至 16BIT
- 具备深度为 8 的接收和发送 FIFO

2.9.3 常量定义

常量名	值	描述
SPI_FORMAT_SPI	0	Motorola SPI 格式
SPI_FORMAT_TI_SSI	1	TI SSI 格式
常量名	值	描述
SPI_FIRST_EDGE	0	第一个时钟沿开始采样
SPI_SECOND_EDGE	1	第二个时钟沿开始采样
常量名	值	描述
SPI_LOW_LEVEL	0	空闲时时钟线保持低电平
SPI_HIGH_LEVEL	1	空闲时时钟线保持高电平
常量名	值	描述
SPI_CLKDIV_4	0	SPI_CLK = SYS_CLK / 4
SPI_CLKDIV_8	1	SPI_CLK = SYS_CLK / 8
SPI_CLKDIV_16	2	SPI_CLK = SYS_CLK / 16
SPI_CLKDIV_32	3	SPI_CLK = SYS_CLK / 32
SPI_CLKDIV_64	4	SPI_CLK = SYS_CLK / 64
SPI_CLKDIV_128	5	SPI_CLK = SYS_CLK / 128
SPI_CLKDIV_256	6	SPI_CLK = SYS_CLK / 256
SPI_CLKDIV_512	7	SPI_CLK = SYS_CLK / 512
常量名	值	描述
SPI_FORMAT_SPI	0	

2.9.4 类型定义

SPI_InitStructure

```
typedef struct {
    uint8_t  FrameFormat;
    uint8_t  SampleEdge;
    uint8_t  IdleLevel;
    uint8_t  FrameSize;
    uint8_t  Master;
    uint8_t  clkDiv;

    uint8_t  RXHFullIEn;
    uint8_t  RXFullIEn;
    uint8_t  TXHFullIEn;
    uint8_t  TXEmptyIEn;
} SPI_InitStructure;
```

成员	描述
FrameFormat	帧格式：SPI_FORMAT_SPI、SPI_FORMAT_TI_SSI
SampleEdge	在 SPI 帧格式下，选择数据采样边沿：SPI_FIRST_EDGE、SPI_SECOND_EDGE

IdleLevel	在 SPI 帧格式下，选择空闲时（无数据传输时）时钟线的电平：SPI_LOW_LEVEL、SPI_HIGH_LEVEL
FrameSize	帧长度，有效值 4-16
Master	1 主机模式　　0 从机模式
clkDiv	SPI_CLK = SYS_CLK / clkDiv，有效值：SPI_CLKDIV_4、SPI_CLKDIV_8、... ... 、SPI_CLKDIV_512
RXHFullIEn	接收 FIFO 半满中断使能
RXFullIEn	接收 FIFO 满　中断使能
TXHFullIEn	发送 FIFO 半满中断使能
TXEmptyIEn	发送 FIFO 空中断使能

2.9.5 函数

SPI_Init

原型：void SPI_Init(SPI_TypeDef * SPIx, SPI_InitStructure * initStruct)

```
/*****************************************************************************
* 函数名称:      SPI_Init()
* 功能说明:      SPI 同步串行接口初始化，包括帧长度设定、时序设定、速度设定、中断设定、FIFO 触发设定
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
*               SPI_InitStructure * initStruct      包含 SPI 相关设定值的结构体
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_Open

原型：void SPI_Open(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_Open()
* 功能说明:      SPI 打开，允许收发
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_Close

原型：void SPI_Close(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_Close()
* 功能说明:      SPI 关闭，禁止收发
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_Read

原型：uint32_t SPI_Read(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_Read()
* 功能说明:      读取一个数据
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: uint32_t                    读取到的数据
* 注意事项: 无
*****************************************************************************/
```

SPI_Write

原型：void SPI_Write(SPI_TypeDef * SPIx, uint32_t data)

```
/*****************************************************************************
* 函数名称:      SPI_Write()
* 功能说明:      写入一个数据
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
*                uint32_t                 要写入的数据
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_WriteWithWait

原型：void SPI_WriteWithWait(SPI_TypeDef * SPIx, uint32_t data)

```
/*****************************************************************************
* 函数名称:      SPI_WriteWithWait()
* 功能说明:      写入一个数据并等待数据完全发送出去
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1、SPI1
*                uint32_t                 要写入的数据
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_ReadWrite

原型：uint32_t SPI_ReadWrite(SPI_TypeDef * SPIx, uint32_t data)

```
/*****************************************************************************
* 函数名称: SPI_ReadWrite()
* 功能说明: 发送一个数据，并返回发送过程中接收到的
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
*                uint32_t data             要发送的数据
* 输      出: uint32_t                    接收到的数据
* 注意事项: 对于同一个 SPI 模块，此函数不应与 SPI_Write()混着用，因为 SPI_Write()不清除 SPI_STAT_RFNE 状态
*****************************************************************************/
```

SPI_IsRXEmpty

原型：uint32_t SPI_IsRXEmpty(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_IsRXEmpty()
* 功能说明:      接收 FIFO 是否空，如果不空则可以继续 SPI_Read()
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: uint32_t                    1 接收 FIFO 空      0 接收 FIFO 非空
* 注意事项: 无
*****************************************************************************/
```

SPI_IsTXFull

原型：uint32_t SPI_IsTXFull(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_IsTXFull()
* 功能说明:      发送 FIFO 是否满，如果不满则可以继续 SPI_Write()
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: uint32_t                    1 发送 FIFO 满      0 发送 FIFO 不满
* 注意事项: 无
*****************************************************************************/
```

SPI_IsTXEmpty

原型：uint32_t SPI_IsTXEmpty(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_IsTXEmpty()
* 功能说明:      发送 FIFO 是否空
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: uint32_t                    1 发送 FIFO 空      0 发送 FIFO 非空
* 注意事项: 无
*****************************************************************************/
```

SPI_INTRXHalfFullEn

原型：void SPI_INTRXHalfFullEn(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTRXHalfFullEn()
* 功能说明:      接收 FIFO 半满中断使能
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_INTRXHalfFullDis

原型：void SPI_INTRXHalfFullDis(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTRXHalfFullDis()
* 功能说明:      接收 FIFO 半满中断禁止
```

```
* 输    入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_INTRXHalfFullStat

原型：uint32_t SPI_INTRXHalfFullStat(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:    SPI_INTRXHalfFullStat()
* 功能说明:    接收 FIFO 半满中断状态
* 输    入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输    出: uint32_t                  1 接收 FIFO 达到半满    0 接收 FIFO 未达到半满
* 注意事项: 无
*****************************************************************************/
```

SPI_INTRXFullEn

原型：void SPI_INTRXFullEn(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:    SPI_INTRXFullEn()
* 功能说明:    接收 FIFO 满中断使能
* 输    入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_INTRXFullDis

原型：void SPI_INTRXFullDis(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:    SPI_INTRXFullDis()
* 功能说明:    接收 FIFO 满中断禁止
* 输    入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_INTRXFullStat

原型：uint32_t SPI_INTRXFullStat(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:    SPI_INTRXFullStat()
* 功能说明:    接收 FIFO 满中断状态
* 输    入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输    出: uint32_t                  1 接收 FIFO 满    0 接收 FIFO 未满
* 注意事项: 无
*****************************************************************************/
```

SPI_INTRXOverflowEn

原型：void SPI_INTRXOverflowEn(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTRXOverflowEn()
* 功能说明:      接收 FIFO 溢出中断使能
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_INTRXOverflowDis

原型：void SPI_INTRXOverflowDis(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTRXOverflowDis()
* 功能说明:      接收 FIFO 溢出中断禁止
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_INTRXOverflowClr

原型：void SPI_INTRXOverflowClr(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTRXOverflowClr()
* 功能说明:      接收 FIFO 溢出中断标志清除
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_INTRXOverflowStat

原型：uint32_t SPI_INTRXOverflowStat(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTRXOverflowStat()
* 功能说明:      接收 FIFO 溢出中断状态
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: uint32_t                    1 接收 FIFO 溢出      0 接收 FIFO 未溢出
* 注意事项: 无
*****************************************************************************/
```

SPI_INTTXHalfFullEn

原型：void SPI_INTTXHalfFullEn(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTTXHalfFullEn()
* 功能说明:      发送 FIFO 半满中断使能
```

```
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_INTTXHalfFullDis

原型：void SPI_INTTXHalfFullDis(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTTXHalfFullDis()
* 功能说明:      发送 FIFO 半满中断禁止
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_INTTXHalfFullStat

原型：uint32_t SPI_INTTXHalfFullStat(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTTXHalfFullStat()
* 功能说明:      发送 FIFO 半满中断状态
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: uint32_t                  1  发送 FIFO 达到半满      0  发送 FIFO 未达到半满
* 注意事项: 无
*****************************************************************************/
```

SPI_INTTXEmptyEn

原型：void SPI_INTTXEmptyEn(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTTXEmptyEn()
* 功能说明:      发送 FIFO 空中断使能
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_INTTXEmptyDis

原型：void SPI_INTTXEmptyDis(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTTXEmptyDis()
* 功能说明:      发送 FIFO 空中断禁止
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

SPI_INTTXEmptyStat

原型：uint32_t SPI_INTTXEmptyStat(SPI_TypeDef * SPIx)

```
/*****************************************************************************
* 函数名称:      SPI_INTTXEmptyStat()
* 功能说明:      发送 FIFO 空中断状态
* 输      入: SPI_TypeDef * SPIx          指定要被设置的 SPI，有效值包括 SPI0、SPI1
* 输      出: uint32_t                    1 发送 FIFO 空      0 发送 FIFO 非空
* 注意事项: 无
*****************************************************************************/
```

2.10 脉冲宽度调制（PWM）发生器

2.10.1介绍

通过配置 BCTRLx 寄存器，可配置各路 PWM 初始电平。ENABLE 寄存器 EN 位使能后，对应通道起始输出电平即为该寄存器对应通道配置电平。

2.10.2特性

- 6 组 16 位宽 PWM 控制，最多可产生 12 路 PWM 信号
- 支持互补、中心对称、单步模式
- 最高支持输入时钟 8 分频
- 提供高电平起始中断及周期结束中断
- 支持死区设置
- 可选择初始输出电平选择
- 支持刹车功能
- 支持硬件自动触发 ADC 采样

2.10.3常量定义

常量名	值	描述
PWM_CLKDIV_1	0	PWM_CLK = SYS_CLK
PWM_CLKDIV_8	1	PWM_CLK = SYS_CLK / 8
常量名	**值**	**描述**
PWM_MODE_INDEP	0	A 路和 B 路为两路独立输出
PWM_MODE_COMPL	1	A 路和 B 路为一路互补输出
PWM_MODE_INDEP_CALIGN	3	A 路和 B 路为两路独立输出，中心对齐
PWM_MODE_COMPL_CALIGN	4	A 路和 B 路为一路互补输出，中心

		对齐
常量名	值	描述
PWM_CH_A	0	PWM 通道 A
PWM_CH_B	1	PWM 通道 B

2.10.4类型定义

PWM_InitStructure

```
typedef struct {
    uint8_t    clk_div;

    uint8_t    mode;

    uint16_t cycleA;
    uint16_t hdutyA;
    uint8_t    deadzoneA;
    uint8_t    initLevelA;

    uint16_t cycleB;
    uint16_t hdutyB;
    uint8_t    deadzoneB;
    uint8_t    initLevelB;

    uint8_t    HEndAIEn;
    uint8_t    NCycleAIEn;
    uint8_t    HEndBIEn;
    uint8_t    NCycleBIEn;
} PWM_InitStructure;
```

成员	描述
clk_div	PWM_CLKDIV_1、PWM_CLKDIV_8
mode	PWM_MODE_INDEP、PWM_MODE_COMPL、 PWM_MODE_INDEP_CALIGN、 PWM_MODE_COMPL_CALIGN
cycleA	A 路周期
hdutyA	A 路占空比
deadzoneA	A 路死区时长，取值 0--63
initLevelA	A 路初始输出电平，0 低电平　　1 高电平
cycleB	B 路周期
hdutyB	B 路占空比
deadzoneB	B 路死区时长，取值 0--63
initLevelB	B 路初始输出电平，0 低电平　　1 高电平
HEndAIEn	A 路高电平结束中断使能
NCycleAIEn	A 路新周期开始中断使能

HEndBIEn	B 路高电平结束中断使能
NCycleBIEn	B 路新周期开始中断使能

2.10.5函数

PWM_Init

原型：void PWM_Init(PWM_TypeDef * PWMx, PWM_InitStructure * initStruct)

```
/*****************************************************************************
* 函数名称:      PWM_Init()
* 功能说明:      PWM 初始化
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括
PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*                PWM_InitStructure * initStruct      包含 PWM 相关设定值的结构体
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

PWM_Start

原型：void PWM_Start(PWM_TypeDef * PWMx, uint32_t chA, uint32_t chB)

```
/*****************************************************************************
* 函数名称:      PWM_Start()
* 功能说明:      启动 PWM，开始 PWM 输出
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、
PWM1、PWM2、PWM3、PWM4、PWM5
*                uint32_t chA          0  通道 A 不启动          1  通道 A 启动
*                uint32_t chB          0  通道 B 不启动          1  通道 B 启动
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

PWM_Stop

原型：void PWM_Stop(PWM_TypeDef * PWMx, uint32_t chA, uint32_t chB)

```
/*****************************************************************************
* 函数名称:      PWM_Stop()
* 功能说明:      关闭 PWM，停止 PWM 输出
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、
PWM1、PWM2、PWM3、PWM4、PWM5
*                uint32_t chA          0  通道 A 不关闭          1  通道 A 关闭
*                uint32_t chB          0  通道 B 不关闭          1  通道 B 关闭
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

PWM_SetCycle

原型：void PWM_SetCycle(PWM_TypeDef * PWMx, uint32_t chn, uint16_t cycle)

```
/****************************************************************************
* 函数名称:      PWM_SetCycle()
* 功能说明:      设置周期
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*               uint32_t chn              选择要设置哪个通道，有效值：PWM_CH_A、PWM_CH_B
*               uint16_t cycle            要设定的周期值
* 输      出: 无
* 注意事项: 无
****************************************************************************/
```

PWM_GetCycle

原型：uint16_t PWM_GetCycle(PWM_TypeDef * PWMx, uint32_t chn)

```
/****************************************************************************
* 函数名称:      PWM_GetCycle()
* 功能说明:      获取周期
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*           uint32_t chn                选择要查询哪个通道，有效值：PWM_CH_A、PWM_CH_B
* 输      出: uint16_t                    获取到的周期值
* 注意事项: 无
****************************************************************************/
```

PWM_SetHDuty

原型：void PWM_SetHDuty(PWM_TypeDef * PWMx, uint32_t chn, uint16_t hduty)

```
/****************************************************************************
*************************************************************
* 函数名称:      PWM_SetHDuty()
* 功能说明:      设置高电平时长
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*           uint32_t chn                选择要设置哪个通道，有效值：PWM_CH_A、PWM_CH_B
*           uint16_t hduty              要设定的高电平时长
* 输      出: 无
* 注意事项: 无
****************************************************************************/
```

PWM_GetHDuty

原型：uint16_t PWM_GetHDuty(PWM_TypeDef * PWMx, uint32_t chn)

```
/****************************************************************************
```

```
* 函数名称:      PWM_GetHDuty()
* 功能说明: 获取高电平时长
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*           uint32_t chn                  选择要查询哪个通道，有效值：PWM_CH_A、PWM_CH_B
* 输      出: uint16_t                    获取到的高电平时长
* 注意事项: 无
*******************************************************************************/
```

PWM_SetDeadzone

原型：void PWM_SetDeadzone(PWM_TypeDef * PWMx, uint32_t chn, uint8_t deadzone)

```
/*******************************************************************************
* 函数名称:      PWM_SetDeadzone()
* 功能说明:      设置死区时长
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*           uint32_t chn                  选择要设置哪个通道，有效值：PWM_CH_A、PWM_CH_B
*           uint8_t deadzone              要设定的死区时长
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

PWM_GetDeadzone

原型：uint8_t PWM_GetDeadzone(PWM_TypeDef * PWMx, uint32_t chn)

```
/*******************************************************************************
* 函数名称:      PWM_GetDeadzone()
* 功能说明: 获取死区时长
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2
*           uint32_t chn                  选择要查询哪个通道，有效值：PWM_CH_A、PWM_CH_B
* 输      出: uint8_t                     获取到的死区时长
* 注意事项: 无
*******************************************************************************/
```

PWM_IntNCycleEn

原型：void PWM_IntNCycleEn(PWM_TypeDef * PWMx, uint32_t chn)

```
/*******************************************************************************
* 函数名称:      PWM_GetDeadzone()
* 功能说明: 获取死区时长
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*           uint32_t chn                  选择要查询哪个通道，有效值：PWM_CH_A、PWM_CH_B
* 输      出: uint8_t                     获取到的死区时长
* 注意事项: 无
```

**********/

PWM_IntNCycleDis

原型：void PWM_IntNCycleDis(PWM_TypeDef * PWMx, uint32_t chn)

```
/*****************************************************************************
* 函数名称:      PWM_IntNCycleDis()
* 功能说明: 新周期开始中断禁能
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*           uint32_t chn                  选择要设置哪个通道，有效值：PWM_CH_A、PWM_CH_B
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

PWM_IntNCycleClr

原型：void PWM_IntNCycleClr(PWM_TypeDef * PWMx, uint32_t chn)

```
/*****************************************************************************
* 函数名称:      PWM_IntNCycleClr()
* 功能说明: 新周期开始中断标志清除
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*           uint32_t chn                  选择要设置哪个通道，有效值：PWM_CH_A、PWM_CH_B
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

PWM_IntNCycleStat

原型：uint32_t PWM_IntNCycleStat(PWM_TypeDef * PWMx, uint32_t chn)

```
/*****************************************************************************
* 函数名称:      PWM_IntNCycleStat()
* 功能说明: 新周期开始中断是否发生
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*           uint32_t chn                  选择要设置哪个通道，有效值：PWM_CH_A、PWM_CH_B
* 输      出: uint32_t                    1 新周期开始中断已发生      0 新周期开始中断未发生
* 注意事项: 无
*****************************************************************************/
```

PWM_IntHEndEn

原型：void PWM_IntHEndEn(PWM_TypeDef * PWMx, uint32_t chn)

```
/*****************************************************************************
* 函数名称:      PWM_IntHEndEn()
* 功能说明: 高电平结束中断使能
* 输      入: PWM_TypeDef * PWMx          指定要被设置的 PWM，有效值包括 PWM0、
```

PWM1、PWM2、PWM3、PWM4、PWM5
* uint32_t chn 选择要设置哪个通道，有效值：PWM_CH_A、PWM_CH_B
* 输 出: 无
* 注意事项: 无
***/

PWM_IntHEndDis

原型：void PWM_IntHEndDis(PWM_TypeDef * PWMx, uint32_t chn)

```
/*******************************************************************************
* 函数名称:  PWM_IntHEndDis()
* 功能说明: 高电平结束中断禁能
* 输  入: PWM_TypeDef * PWMx  指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*  uint32_t chn  选择要设置哪个通道，有效值：PWM_CH_A、PWM_CH_B
* 输  出: 无
* 注意事项: 无
*******************************************************************************/
```

PWM_IntHEndClr

原型：void PWM_IntHEndClr(PWM_TypeDef * PWMx, uint32_t chn)

```
/*******************************************************************************
* 函数名称:  PWM_IntHEndClr()
* 功能说明: 高电平结束中断标志清除
* 输  入: PWM_TypeDef * PWMx  指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*  uint32_t chn  选择要设置哪个通道，有效值：PWM_CH_A、PWM_CH_B
* 输  出: 无
* 注意事项: 无
*******************************************************************************/
```

PWM_IntHEndStat

原型：uint32_t PWM_IntHEndStat(PWM_TypeDef * PWMx, uint32_t chn)

```
/*******************************************************************************
* 函数名称:  PWM_IntHEndStat()
* 功能说明: 高电平结束中断是否发生
* 输  入: PWM_TypeDef * PWMx  指定要被设置的 PWM，有效值包括 PWM0、PWM1、PWM2、PWM3、PWM4、PWM5
*  uint32_t chn  选择要设置哪个通道，有效值：PWM_CH_A、PWM_CH_B
* 输  出: uint32_t  1 高电平结束中断已发生  0 高电平结束中断未发生
* 注意事项: 无
*******************************************************************************/
```

2.11 模拟数字转换器（ADC）

2.11.1介绍

ADC 即模数转换器，作用是将模拟信号转变为数字信号。本系列所有型号 SAR ADC 操作均相同，两个 12 位逐次逼近型模拟数字转换器最多支持 8 通道，使用前需使能 SAR ADC 模块时钟。

2.11.2特性

- 12-bits 分辨率
- 最多 8 路输入通道
- 最高 1MSPS 转换速率
- 支持 7 档内部参考电平，最低 100mV 量程
- 支持单次模式和连续模式
- 灵活的转换启动方式
- 支持软件、 PWM 启动
- 每个通道都有自己独立的转换结果数据寄存器和转换完成、数据溢出状态寄存器
- 每个通道都有自己独立的转换完成中断使能和数据溢出中断使能

2.11.3常量定义

常量名	值	描述
ADC_CH0	0x01	ADC 输入通道 0
ADC_CH1	0x02	ADC 输入通道 1
ADC_CH2	0x04	ADC 输入通道 2
ADC_CH3	0x08	ADC 输入通道 3
ADC_CH4	0x10	ADC 输入通道 4
ADC_CH5	0x20	ADC 输入通道 5
ADC_CH6	0x40	ADC 输入通道 6
ADC_CH7	0x80	ADC 输入通道 7
ADC_CLKSRC_HRC	1	内部高频 RC 时钟
ADC_CLKSRC_VCO_DIV16	2	压控振荡器频率 16 分频
ADC_CLKSRC_VCO_DIV32	3	压控振荡器频率 32 分频
ADC_CLKSRC_VCO_DIV64	4	压控振荡器频率 64 分频
ADC_AVG_SAMPLE1	0	一次启动采样、转换 1 次
ADC_AVG_SAMPLE2	1	一次启动连续采样、转换 2 次，并计算两次结果的平均值作为转换结果
ADC_AVG_SAMPLE4	3	一次启动连续采样、转换 4 次，并计算两次

常量名	值	描述
		结果的平均值作为转换结果
ADC_AVG_SAMPLE8	7	一次启动连续采样、转换 8 次，并计算两次结果的平均值作为转换结果
ADC_AVG_SAMPLE16	15	一次启动连续采样、转换 16 次，并计算两次结果的平均值作为转换结果
ADC_TRIGSRC_SW	0	软件触发，即 ADC->START.GO 写 1 启动转换
ADC_TRIGSRC_PWM	1	PWM 触发 ADC
PGA_VCM_INTERNAL	1	PGA 输入共模电平由内部电路产生，ADC_REFP 和 ADC_REFN 可悬空
PGA_VCM_EXTERNAL	0	PGA 输入共模电平由外部引脚提供，(ADC_REFP + ADC_REFN) 电平值须与量程相同
ADC_IN_RANGE_100mV	0	PGA 输入量程 100mV，PGA 增益 25.1dB
ADC_IN_RANGE_150mV	1	PGA 输入量程 150mV，PGA 增益 21.6dB
ADC_IN_RANGE_500mV	2	PGA 输入量程 500mV，PGA 增益 11.1dB
ADC_IN_RANGE_1V2	3	PGA 输入量程 1.2V，PGA 增益 3.5dB
ADC_IN_RANGE_1V8	4	PGA 输入量程 1.8V，PGA 增益 0 dB
ADC_IN_RANGE_2V5	5	PGA 输入量程 2.5V，PGA 增益-2.9dB
ADC_IN_RANGE_3V3	6	PGA 输入量程 3.3V，PGA 增益-5.3dB
ADC_IN_VCM_50mV	0	PGA 输入共模电压 50mV
ADC_IN_VCM_75mV	1	PGA 输入共模电压 75mV
ADC_IN_VCM_250mV	2	PGA 输入共模电压 250mV
ADC_IN_VCM_600mV	3	PGA 输入共模电压 600mV
ADC_IN_VCM_900mV	4	PGA 输入共模电压 900mV
ADC_IN_VCM_1V25	5	PGA 输入共模电压 1.25V
ADC_IN_VCM_1V65	6	PGA 输入共模电压 1.65V

2.11.4类型定义

ADC_InitStructure

```
typedef struct {
    uint8_t clk_src;
    uint8_t clk_div;
    uint8_t pga_gain;
    uint8_t channels;
    uint8_t samplAvg;
    uint8_t trig_src;
    uint8_t Continue;
                                //          0 单次转换模式，转换完成后 START
位自动清除停止转换
    uint8_t EOC_IEn;
```

```
    uint8_t OVF_IEn;
    uint8_t HFULL_IEn;
    uint8_t FULL_IEn;
} ADC_InitStructure;
```

成员	描述
clk_src	ADC 转换时钟源：ADC_CLKSRC_HRC、ADC_CLKSRC_VCO_DIV16、ADC_CLKSRC_VCO_DIV32、ADC_CLKSRC_VCO_DIV32
clk_div	ADC 转换时钟分频，取值 1--31
pga_gain	ADC 前置可编程增益放大器增益值，ADC_IN_RANGE_100mV、... ... 、ADC_IN_RANGE_3V3
channels	ADC 转换通道选中，ADC_CH0、ADC_CH1、... ... 、ADC_CH7 及其组合（即“按位或”运算）
samplAvg	采样取平均，触发启动 ADC 转换后，ADC 在一个通道上连续采样、转换多次，并将它们的平均值作为该通道转换结果
trig_src	ADC 触发方式：ADC_TRIGSRC_SW、ADC_TRIGSRC_PWM、ADC_TRIGSRC_TIMR2、ADC_TRIGSRC_TIMR3
Continue	在软件触发模式下：1 连续转换模式，启动后一直采样、转换，直到软件清除 START 位
EOC_IEn	EOC 中断使能，可针对每个通道设置，其有效值为 ADC_CH0、ADC_CH1、... ... 、ADC_CH7 及其组合（即“按位或”运算）
OVF_IEn	OVF 中断使能，可针对每个通道设置，其有效值为 ADC_CH0、ADC_CH1、... ... 、ADC_CH7 及其组合（即“按位或”运算）

2.11.5函数

ADC_Init

原型：void ADC_Init(ADC_TypeDef * ADCx, ADC_InitStructure * initStruct)

```
/*****************************************************************************
* 函数名称: ADC_Init()
* 功能说明:     ADC 模数转换器初始化
* 输     入: ADC_TypeDef * ADCx          指定要被设置的 ADC，有效值包括 ADC0、ADC1
*           ADC_InitStructure * initStruct          包含 ADC 各相关定值的结构体
* 输     出: 无
* 注意事项: 无
*****************************************************************************/
```

ADC_Open

原型：void ADC_Open(ADC_TypeDef * ADCx)

```
/*****************************************************************************
* 函数名称:     ADC_Open()
* 功能说明:     ADC 开启，可以软件启动、或硬件触发 ADC 转换
* 输     入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
```

```
* 输    出: 无
* 注意事项: 无
*******************************************************************************/
```

ADC_Close

原型：void ADC_Close(ADC_TypeDef * ADCx)

```
/*******************************************************************************
* 函数名称:     ADC_Close()
* 功能说明:     ADC 关闭，无法软件启动、或硬件触发 ADC 转换
* 输    入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
* 输    出: 无
* 注意事项: 无
*******************************************************************************/
```

ADC_Start

原型：void ADC_Start(ADC_TypeDef * ADCx)

```
/*******************************************************************************
* 函数名称:     ADC_Start()
* 功能说明:     软件触发模式下启动 ADC 转换
* 输    入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
* 输    出: 无
* 注意事项: 无
*******************************************************************************/
```

ADC_Stop

原型：void ADC_Stop(ADC_TypeDef * ADCx)

```
/*******************************************************************************
* 函数名称:     ADC_Stop()
* 功能说明:     软件触发模式下停止 ADC 转换
* 输    入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
* 输    出: 无
* 注意事项: 无
*******************************************************************************/
```

ADC_Read

原型：uint32_t ADC_Read(ADC_TypeDef * ADCx, uint32_t chn)

```
/*******************************************************************************
* 函数名称:     ADC_Read()
* 功能说明:     从指定通道读取转换结果
* 输    入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*           uint32_t chn               要读取转换结果的通道，有效值 ADC_CH0、
ADC_CH1、... ... 、ADC_CH7
* 输    出: uint32_t                   读取到的转换结果
* 注意事项: 无
*******************************************************************************/
```

ADC_IsEOC

原型：uint32_t ADC_IsEOC(ADC_TypeDef * ADCx, uint32_t chn)

```
/*****************************************************************************
* 函数名称:      ADC_IsEOC()
* 功能说明:      指定通道是否 End Of Conversion
* 输      入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*                uint32_t chn                要查询状态的通道，有效值 ADC_CH0、
ADC_CH1、... ... 、ADC_CH7
* 输      出: uint32_t                  1 该通道完成了转换      0 该通道未完成转换
* 注意事项: 无
*****************************************************************************/
```

ADC_ChnSelect

原型：void ADC_ChnSelect(ADC_TypeDef * ADCx, uint32_t chns)

```
/*****************************************************************************
* 函数名称:      ADC_ChnSelect()
* 功能说明:      ADC 通道选通，模数转换会在选通的通道上依次采样转换
* 输      入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*                uint32_t chns                要选通的通道，有效值 ADC_CH0、
ADC_CH1、... ... 、ADC_CH7 及其组合（即“按位或”运算）
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

ADC_SetGain

原型：void ADC_SetGain(ADC_TypeDef * ADCx, uint32_t gain)

```
/*****************************************************************************
* 函数名称:      ADC_SetGain()
* 功能说明:      ADC 设置 PGA 增益
* 输      入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*                uint32_t gain              PGA 增益，可取值 ADC_IN_RANGE_100mV、
ADC_IN_RANGE_150mV、... ...、ADC_IN_RANGE_2V5、ADC_IN_RANGE_3V3
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

ADC_SetGainAndVCM

原型：void ADC_SetGainAndVCM(ADC_TypeDef * ADCx, uint32_t gain, uint32_t ivcm)

```
/*****************************************************************************
* 函数名称:      ADC_SetGainAndVCM()
* 功能说明:      ADC 设置 PGA 增益和 PGA 输入共模电压
* 输      入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*                uint32_t gain              PGA 增益，可取值 ADC_IN_RANGE_100mV、
ADC_IN_RANGE_150mV、... ...、ADC_IN_RANGE_2V5、ADC_IN_RANGE_3V3
```

```
*                uint32_t ivcm                PGA 输入共模电压，可取值
ADC_IN_VCM_50mV、ADC_IN_VCM_75mV、... ...、ADC_IN_VCM_1V25、
ADC_IN_VCM_1V65
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

ADC_IntEOCEn

原型：void ADC_IntEOCEn(ADC_TypeDef * ADCx, uint32_t chn)

```
/*******************************************************************************
* 函数名称:      ADC_IntEOCEn()
* 功能说明:      转换完成中断使能
* 输      入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*                uint32_t chn              要设置的通道，有效值 ADC_CH0、ADC_CH1、... ... 、
ADC_CH7
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

ADC_IntEOCDis

原型：void ADC_IntEOCDis(ADC_TypeDef * ADCx, uint32_t chn)

```
/*******************************************************************************
* 函数名称:      ADC_IntEOCDis()
* 功能说明:      转换完成中断禁止
* 输      入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*                uint32_t chn              要设置的通道，有效值 ADC_CH0、ADC_CH1、... ... 、
ADC_CH7
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

ADC_IntEOCClr

原型：void ADC_IntEOCClr(ADC_TypeDef * ADCx, uint32_t chn)

```
/*******************************************************************************
* 函数名称:      ADC_IntEOCClr()
* 功能说明:      转换完成中断标志清除
* 输      入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*                uint32_t chn              要设置的通道，有效值 ADC_CH0、ADC_CH1、... ... 、
ADC_CH7
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

ADC_IntEOCStat

原型：uint32_t ADC_IntEOCStat(ADC_TypeDef * ADCx, uint32_t chn)

```
/*****************************************************************************
* 函数名称:      ADC_IntEOCStat()
* 功能说明:      转换完成中断状态
* 输      入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*               uint32_t chn              要查询的通道，有效值 ADC_CH0、ADC_CH1、…… 、
ADC_CH7
* 输      出: uint32_t                    1 该通道完成了转换      0 该通道未完成转换
* 注意事项: 无
*****************************************************************************/
```

ADC_IntOVFEn

原型：void ADC_IntOVFEn(ADC_TypeDef * ADCx, uint32_t chn)

```
/*****************************************************************************
* 函数名称:      ADC_IntOVFEn()
* 功能说明:      数据溢出中断使能
* 输      入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*               uint32_t chn              要设置的通道，有效值 ADC_CH0、ADC_CH1、…… 、
ADC_CH7
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

ADC_IntOVFDis

原型：void ADC_IntOVFDis(ADC_TypeDef * ADCx, uint32_t chn)

```
/*****************************************************************************
* 函数名称:      ADC_IntOVFDis()
* 功能说明:      数据溢出中断禁止
* 输      入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*               uint32_t chn              要设置的通道，有效值 ADC_CH0、ADC_CH1、…… 、
ADC_CH7
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

ADC_IntOVFClr

原型：void ADC_IntOVFClr(ADC_TypeDef * ADCx, uint32_t chn)

```
/*****************************************************************************
* 函数名称:      ADC_IntOVFClr()
* 功能说明:      数据溢出中断标志清除
* 输      入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*               uint32_t chn              要设置的通道，有效值 ADC_CH0、ADC_CH1、…… 、
ADC_CH7
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

ADC_IntOVFStat

原型：uint32_t ADC_IntOVFStat(ADC_TypeDef * ADCx, uint32_t chn)

```
/*****************************************************************************
* 函数名称:      ADC_IntOVFStat()
* 功能说明:      数据溢出中断状态
* 输    入: ADC_TypeDef * ADCx          指定要被设置的 ADC，可取值包括 ADC
*           uint32_t chn               要查询的通道，有效值 ADC_CH0、ADC_CH1、…… 、
ADC_CH7
* 输    出: uint32_t                   1 有通道溢出    0 没有通道溢出
* 注意事项: 无
*****************************************************************************/
```

2.12 直接内存存取（DMA）控制器

2.12.1介绍

DMA 主要功能在于完成两个 AHB Master 口之间的数据搬移，支持系统内存与系统内存、系统内存与片上外设（NORFLC/SDRAMC/SRAMC）间的数据搬运，外设与外设之间无法直接进行数据交换，需先从外设 A 搬移到系统内存，然后再从系统内存搬移到外设 B。搬运过程中无需占用内核资源，从而节省了内核资源可供其他操作使用。

2.12.2特性

- Master 接口支持 AMBA 2.0 AHB Lite
- Master 接口支持 SINGLE 和 INCR4 传输
- Master 接口支持 BYTE 和 WORD 操作
- Slave 接口为 AMBA 2.0 AHB 接口
- Slave 接口支持 WORD 操作
- 最大通道数为 8
- 支持二种地址变化方式：递增，固定
- 支持 memory to memory，memory to peripheral，peripheral to peripheral 三种握手方式

2.12.3常量定义

常量名	值	描述
DMA_CH0	0	读 SPI 通道
DMA_CH1	1	写 SPI 通道
DMA_CH2	2	读 ADC 通道

2.12.4函数

DIV_Init

原型：void DIV_Init(DIV_TypeDef * DIVx)

```
/*****************************************************************************
* 函数名称:      DIV_Init()
* 功能说明:      硬件除法器初始化
* 输      入: DIV_TypeDef * DIVx      指定要被设置的硬件除法器，有效值包括 DIV
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

DMA_CH_Config

原型：void DMA_CHM_Config(uint32_t chn, uint32_t src_addr, uint32_t src_addr_incr, uint32_t dst_addr, uint32_t dst_addr_incr, uint32_t num_word, uint32_t int_en)

```
/*****************************************************************************
* 函数名称: DMA_CHM_Config()
* 功能说明:      DMA 通道配置，用于存储器间（如 Flash 和 RAM 间）搬运数据
* 输      入: uint32_t chn              指定要配置的通道，有效值有 DMA_CH0、
DMA_CH1、DMA_CH1
*           uint32_t src_addr          源地址，必须字对齐，即地址的最低 2 位必须是 00
*           uint32_t src_addr_incr     0 固定地址      1 地址递增
*           uint32_t dst_addr          目的地址，必须字对齐，即地址的最低 2 位必须是
00
*           uint32_t dst_addr_incr     0 固定地址      1 地址递增
*           uint32_t num_word            要搬运的数据字数，最大 1024
*           uint32_t int_en              中断使能，1 数据搬运完成后产生中断      0 数
据搬运完成后不产生中断
* 输      出: 无
* 注意事项: 搬运数据量以字为单元，不是字节
*****************************************************************************/
```

DMA_CH_Open

原型：void DMA_CH_Open(uint32_t chn)

```
/*****************************************************************************
* 函数名称: DMA_CH_Open()
* 功能说明:      DMA 通道打开
* 输      入: uint32_t chn              指定要配置的通道,有效值有 DMA_CH0、DMA_CH1、
DMA_CH1
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

DMA_CH_Close

原型：void DMA_CH_Close(uint32_t chn)

```
/*****************************************************************************
* 函数名称: DMA_CH_Close()
* 功能说明:     DMA 通道关闭
* 输    入: uint32_t chn                指定要配置的通道，有效值有 DMA_CH0、DMA_CH1、DMA_CH1
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

DMA_CH_INTEn

原型：void DMA_CH_INTEn(uint32_t chn)

```
/*****************************************************************************
* 函数名称: DMA_CH_INTEn()
* 功能说明:     DMA 中断使能，数据搬运完成后触发中断
* 输    入: uint32_t chn                指定要配置的通道，有效值有 DMA_CH0、DMA_CH1、DMA_CH1
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

DMA_CH_INTDis

原型：void DMA_CH_INTDis(uint32_t chn)

```
/*****************************************************************************
* 函数名称: DMA_CH_INTDis()
* 功能说明:     DMA 中断禁止，数据搬运完成后不触发中断
* 输    入: uint32_t chn                指定要配置的通道，有效值有 DMA_CH0、DMA_CH1、DMA_CH1
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

DMA_CH_INTClr

原型：void DMA_CH_INTClr(uint32_t chn)

```
/*****************************************************************************
* 函数名称: DMA_CH_INTClr()
* 功能说明:     DMA 中断标志清除
* 输    入: uint32_t chn                指定要配置的通道，有效值有 DMA_CH0、DMA_CH1、DMA_CH1
* 输    出: 无
* 注意事项: 无
*****************************************************************************/
```

DMA_CH_INTStat

原型：uint32_t DMA_CH_INTStat(uint32_t chn)

```
/*****************************************************************************
* 函数名称: DMA_CH_INTStat()
* 功能说明:     DMA 中断状态查询
* 输    入: uint32_t chn                指定要配置的通道，有效值有 DMA_CH0、DMA_CH1、DMA_CH1
* 输    出: uint32_t                    1 数据搬运完成    0 数据搬运未完成
* 注意事项: 无
*****************************************************************************/
```

2.13 局域网控制器（CAN）

2.13.1介绍

本系列所有型号 CAN 模块操作均相同。使用前需使能 CAN 模块时钟。与物理层相连需要连接额外的硬件收发器。

2.13.2特性

- 支持协议 2.0A(11bit 标识符)和 2.0B（ 29bit 标识符）
- 支持最大 1 Mbit/s 的比特率
- 提供 64 字节的接收 FIFO
- 提供两个 16 位或 1 个 32 位的滤波器
- 提供可掩蔽中断
- 为自检操作提供可编程环回模式

2.13.3常量定义

常量名	值	描述
CAN_FRAME_STD	0	标准帧格式
CAN_FRAME_EXT	1	扩展帧格式
CAN_MODE_NORMAL	0	常规模式
CAN_MODE_LISTEN	1	监听模式
CAN_MODE_SELFTEST	2	自测模式
CAN_BS1_1tq	0	1 time quantum
CAN_BS1_2tq	1	2 time quantum
CAN_BS1_3tq	2	3 time quantum
CAN_BS1_4tq	3	4 time quantum
CAN_BS1_5tq	4	5 time quantum

常量名	值	描述
CAN_BS1_6tq	5	6 time quantum
CAN_BS1_7tq	6	7 time quantum
CAN_BS1_8tq	7	8 time quantum
CAN_BS1_9tq	8	9 time quantum
CAN_BS1_10tq	9	10 time quantum
CAN_BS1_11tq	10	11 time quantum
CAN_BS1_12tq	11	12 time quantum
CAN_BS1_13tq	12	13 time quantum
CAN_BS1_14tq	13	14 time quantum
CAN_BS1_15tq	14	15 time quantum
CAN_BS1_16tq	15	16 time quantum
CAN_BS2_1tq	0	1 time quantum
CAN_BS2_2tq	1	2 time quantum
CAN_BS2_3tq	2	3 time quantum
CAN_BS2_4tq	3	4 time quantum
CAN_BS2_5tq	4	5 time quantum
CAN_BS2_6tq	5	6 time quantum
CAN_BS2_7tq	6	7 time quantum
CAN_BS2_8tq	7	8 time quantum
CAN_SJW_1tq	0	1 time quantum
CAN_SJW_2tq	1	2 time quantum
CAN_SJW_3tq	2	3 time quantum
CAN_SJW_4tq	3	4 time quantum
CAN_FILTER_16b	0	两个 16 位过滤器
CAN_FILTER_32b	1	一个 32 位过滤器

2.13.4类型定义

CAN_InitStructure

```
typedef struct {
    uint8_t    Mode;
    uint8_t    CAN_BS1;
    uint8_t    CAN_BS2;
    uint8_t    CAN_SJW;
    uint32_t Baudrate;
    uint8_t    FilterMode;
    union {
        uint32_t FilterMask32b;
        struct {
            uint16_t FilterMask16b1;
            uint16_t FilterMask16b2;
        };
```

```
    };
    union {
        uint32_t FilterCheck32b;
        struct {
            uint16_t FilterCheck16b1;
            uint16_t FilterCheck16b2;
        };
    };
    uint8_t   RXNotEmptyIEn;
    uint8_t   RXOverflowIEn;
    uint8_t   ArbitrLostIEn;
    uint8_t   ErrPassiveIEn;
} CAN_InitStructure;
```

成员	描述
Mode	CAN_MODE_NORMAL、CAN_MODE_LISTEN、CAN_MODE_SELFTEST
CAN_BS1	CAN_BS1_1tq、CAN_BS1_2tq、... ... 、CAN_BS1_16tq
CAN_BS2	CAN_BS2_1tq、CAN_BS2_2tq、... ... 、CAN_BS2_8tq
CAN_SJW	CAN_SJW_1tq、CAN_SJW_2tq、CAN_SJW_3tq、CAN_SJW_4tq
Baudrate	波特率，即位传输速率，取值 1--1000000
FilterMode	CAN_FILTER_16b、CAN_FILTER_32b
FilterMask32b	FilterCheck & (~FilterMask) == ID & (~FilterMask)的 Message 通过过滤
RXNotEmptyIEn	接收 FIFO 非空，有数据可读
RXOverflowIEn	接收 FIFO 溢出，有数据丢失
ArbitrLostIEn	控制器丢失仲裁变成接收方
ErrPassiveIEn	接收/发送错误计数值达到 127

CAN_RXMessage

```
typedef struct {
    uint32_t id;
    uint8_t   remote;
    uint8_t   data[8];
    uint8_t   size;
} CAN_RXMessage;
```

成员	描述
id	消息 ID
remote	消息是否为远程帧
size	接收到的数据个数

2.13.5 函数

CAN_Init

原型：void CAN_Init(CAN_TypeDef * CANx, CAN_InitStructure * initStruct)

```
/*****************************************************************************
* 函数名称:      CAN_Init()
* 功能说明:      CAN 接口初始化
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
*               CAN_InitStructure * initStruct      包含 CAN 接口相关设定值的结构体
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

CAN_Open

原型：void CAN_Open(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_Open()
* 功能说明:      CAN 接口打开
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

CAN_Close

原型：void CAN_Close(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_Close()
* 功能说明:      CAN 接口关闭
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

CAN_Transmit

原型：void CAN_Transmit(CAN_TypeDef * CANx, uint32_t format, uint32_t id, uint8_t data[], uint32_t size, uint32_t once)

```
/*****************************************************************************
* 函数名称:      CAN_Transmit()
* 功能说明:      CAN 发送数据
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
*               uint32_t format          CAN_FRAME_STD 标准帧      CAN_FRAME_EXT
扩展帧
*               uint32_t id              消息 ID
*               uint8_t data[]           要发送的数据
```

```
*                uint32_t size          要发送的数据的个数
*                uint32_t once          只发送一次，即使发送失败（仲裁丢失、发送出错、NAK）也不尝试重发
* 输    出: 无
* 注意事项: 无
*******************************************************************************/
```

CAN_TransmitRequest

原型：void CAN_TransmitRequest(CAN_TypeDef * CANx, uint32_t format, uint32_t id, uint32_t once)

```
/*******************************************************************************
* 函数名称:     CAN_TransmitRequest()
* 功能说明:     CAN 发送远程请求，请求远程节点发送数据
* 输    入: CAN_TypeDef * CANx     指定要被设置的 CAN 接口，有效值包括 CAN
*                uint32_t format        CAN_FRAME_STD 标准帧    CAN_FRAME_EXT 扩展帧
*                uint32_t id            消息 ID
*                uint32_t once          只发送一次，即使发送失败（仲裁丢失、发送出错、NAK）也不尝试重发
* 输    出: 无
* 注意事项: 无
*******************************************************************************/
```

CAN_Receive

原型：void CAN_Receive(CAN_TypeDef * CANx, CAN_RXMessage *msg)

```
/*******************************************************************************
* 函数名称:     CAN_Receive()
* 功能说明:     CAN 接收数据
* 输    入: CAN_TypeDef * CANx     指定要被设置的 CAN 接口，有效值包括 CAN
*                CAN_RXMessage *msg     接收到的消息存储在此结构体变量中
* 输    出: 无
* 注意事项: 无
*******************************************************************************/
```

CAN_TXComplete

原型：uint32_t CAN_TXComplete(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:     CAN_TXComplete()
* 功能说明:     发送是否完成
* 输    入: CAN_TypeDef * CANx     指定要被设置的 CAN 接口，有效值包括 CAN
* 输    出: uint32_t              1 已经完成    0 还未完成
* 注意事项: 发送被 Abort 也会触发发送完成，但不会触发发送成功
*******************************************************************************/
```

CAN_TXSuccess

原型：uint32_t CAN_TXSuccess(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_TXSuccess()
* 功能说明:      发送是否成功
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: uint32_t                1 发送成功      0 发送失败
* 注意事项: 无
*****************************************************************************/
```

CAN_AbortTransmit

原型：void CAN_AbortTransmit(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_AbortTransmit()
* 功能说明:      终止发送
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 正在进行的发送无法终止，但执行此命令后若发送失败不会再重发
*****************************************************************************/
```

CAN_TXBufferReady

原型：uint32_t CAN_TXBufferReady(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_TXBufferReady()
* 功能说明:      TX Buffer 是否准备好可以写入消息
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: uint32_t                1 已准备好      0 未准备好
* 注意事项: 无
*****************************************************************************/
```

CAN_RXDataAvailable

原型：uint32_t CAN_RXDataAvailable(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_RXDataAvailable()
* 功能说明:      RX FIFO 中是否有数据可读出
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: uint32_t                1 有数据可读出      0 没有数据
* 注意事项: 无
*****************************************************************************/
```

CAN_SetBaudrate

原型：void CAN_SetBaudrate(CAN_TypeDef * CANx, uint32_t baudrate, uint32_t CAN_BS1, uint32_t CAN_BS2, uint32_t CAN_SJW)

```
/*****************************************************************************
* 函数名称:      CAN_SetBaudrate()
```

```
* 功能说明:      设置波特率
* 输    入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
*           uint32_t baudrate      波特率，即位传输速率
*           uint32_t CAN_BS1       CAN_BS1_1tq、CAN_BS1_2tq、... ... 、
CAN_BS1_16tq
*           uint32_t CAN_BS2       CAN_BS2_1tq、CAN_BS2_2tq、... ... 、
CAN_BS2_8tq
*           uint32_t CAN_SJW       CAN_SJW_1tq、CAN_SJW_2tq、CAN_SJW_3tq、
CAN_SJW_4tq
* 输    出: 无
* 注意事项: 设置前需要先调用 CAN_Close()关闭 CAN 模块
*****************************************************************************/
```

CAN_SetFilter32b

原型：void CAN_SetFilter32b(CAN_TypeDef * CANx, uint32_t check, uint32_t mask)

```
/*****************************************************************************
* 函数名称:      CAN_SetFilter32b()
* 功能说明:      设置接收滤波器，1 个 32 位滤波器
* 输    入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
*           uint32_t check          与 mask 一起决定了接收到的 Message 是否是自己需
要的：check & (~mask) == ID & (~mask)的 Message 通过过滤
*           uint32_t mask
* 输    出: 无
* 注意事项: 设置前需要先调用 CAN_Close()关闭 CAN 模块
*****************************************************************************/
```

CAN_SetFilter16b

原型：void CAN_SetFilter16b(CAN_TypeDef * CANx, uint16_t check1, uint16_t mask1, uint16_t check2, uint16_t mask2)

```
/*****************************************************************************
* 函数名称:      CAN_SetFilter16b()
* 功能说明:      设置接收滤波器，2 个 16 位滤波器
* 输    入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
*           uint16_t check1         与 mask 一起决定了接收到的 Message 是否是自己需
要的：check & (~mask) == ID & (~mask)的 Message 通过过滤
*           uint16_t mask1
*           uint16_t check2
*           uint16_t mask2
* 输    出: 无
* 注意事项: 设置前需要先调用 CAN_Close()关闭 CAN 模块
*****************************************************************************/
```

CAN_INTRXNotEmptyEn

原型：void CAN_INTRXNotEmptyEn(CAN_TypeDef * CANx)

```
/*****************************************************************************
```

```
* 函数名称:      CAN_INTRXNotEmptyEn()
* 功能说明:      当 RX FIFO 中有数据时（非空）触发中断使能
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

CAN_INTRXNotEmptyDis

原型：void CAN_INTRXNotEmptyDis(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTRXNotEmptyDis()
* 功能说明:      当 RX FIFO 中有数据时（非空）触发中断禁止
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

CAN_INTRXNotEmptyStat

原型：uint32_t CAN_INTRXNotEmptyStat(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTRXNotEmptyStat()
* 功能说明:      RX FIFO 非空中断是否触发
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: uint32_t                1 已触发      0 未触发
* 注意事项: 无
*******************************************************************************/
```

CAN_INTTXBufEmptyEn

原型：void CAN_INTTXBufEmptyEn(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTTXBufEmptyEn()
* 功能说明:      当 TX Buffer 空时触发中断使能
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

CAN_INTTXBufEmptyDis

原型：void CAN_INTTXBufEmptyDis(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTTXBufEmptyDis()
* 功能说明:      当 TX Buffer 空时触发中断禁止
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
```

```
*****************************************************************************/
```

CAN_INTTXBufEmptyStat

原型：uint32_t CAN_INTTXBufEmptyStat(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_INTTXBufEmptyStat()
* 功能说明:      TX Buffer 空中断是否触发
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: uint32_t                1 已触发      0 未触发
* 注意事项: 无
*****************************************************************************/
```

CAN_INTErrWarningEn

原型：void CAN_INTErrWarningEn(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_INTErrWarningEn()
* 功能说明:      TXERR/RXERR 计数值达到 Error Warning Limit 时触发中断使能
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

CAN_INTErrWarningDis

原型：void CAN_INTErrWarningDis(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_INTErrWarningDis()
* 功能说明:      TXERR/RXERR 计数值达到 Error Warning Limit 时触发中断禁止
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

CAN_INTErrWarningStat

原型：uint32_t CAN_INTErrWarningStat(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_INTErrWarningStat()
* 功能说明:      TXERR/RXERR 计数值达到 Error Warning Limit 中断是否触发
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: uint32_t                1 已触发      0 未触发
* 注意事项: 无
*****************************************************************************/
```

CAN_INTRXOverflowEn

原型：void CAN_INTRXOverflowEn(CAN_TypeDef * CANx)

```
/*****************************************************************************
```

```
* 函数名称:      CAN_INTRXOverflowEn()
* 功能说明:      RX FIFO 溢出时触发中断使能
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

CAN_INTRXOverflowDis

原型：void CAN_INTRXOverflowDis(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTRXOverflowDis()
* 功能说明:      RX FIFO 溢出时触发中断禁止
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

CAN_INTRXOverflowStat

原型：uint32_t CAN_INTRXOverflowStat(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTRXOverflowStat()
* 功能说明:      RX FIFO 溢出中断是否触发
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: uint32_t                1 已触发      0 未触发
* 注意事项: 无
*******************************************************************************/
```

CAN_INTRXOverflowClear

原型：void CAN_INTRXOverflowClear(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTRXOverflowClear()
* 功能说明:      RX FIFO 溢出中断清除
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

CAN_INTWakeupEn

原型：void CAN_INTWakeupEn(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTWakeupEn()
* 功能说明:      唤醒事件触发中断使能
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
```

```
*****************************************************************************/
```

CAN_INTWakeupDis

原型：void CAN_INTWakeupDis(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_INTWakeupDis()
* 功能说明:      唤醒事件触发中断禁止
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

CAN_INTWakeupStat

原型：uint32_t CAN_INTWakeupStat(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_INTWakeupStat()
* 功能说明:      唤醒事件中断是否触发
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: uint32_t                 1  已触发      0  未触发
* 注意事项: 无
*****************************************************************************/
```

CAN_INTErrPassiveEn

原型：void CAN_INTErrPassiveEn(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_INTErrPassiveEn()
* 功能说明:      TXERR/RXERR 计数值达到 127 时中断使能
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

CAN_INTErrPassiveDis

原型：void CAN_INTErrPassiveDis(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_INTErrPassiveDis()
* 功能说明:      TXERR/RXERR 计数值达到 127 时中断禁止
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

CAN_INTErrPassiveStat

原型：uint32_t CAN_INTErrPassiveStat(CAN_TypeDef * CANx)

```
/*****************************************************************************
```

```
* 函数名称:      CAN_INTErrPassiveStat()
* 功能说明:      TXERR/RXERR 计数值达到 127 中断是否触发
* 输    入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输    出: uint32_t                1 已触发      0 未触发
* 注意事项: 无
*******************************************************************************/
```

CAN_INTArbitrLostEn

原型：void CAN_INTArbitrLostEn(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTArbitrLostEn()
* 功能说明:      仲裁失败中断使能
* 输    入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输    出: 无
* 注意事项: 无
*******************************************************************************/
```

CAN_INTArbitrLostDis

原型：void CAN_INTArbitrLostDis(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTArbitrLostDis()
* 功能说明:      仲裁失败中断禁止
* 输    入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输    出: 无
* 注意事项: 无
*******************************************************************************/
```

CAN_INTArbitrLostStat

原型：uint32_t CAN_INTArbitrLostStat(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTArbitrLostStat()
* 功能说明:      仲裁失败中断是否触发
* 输    入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输    出: uint32_t                1 已触发      0 未触发
* 注意事项: 无
*******************************************************************************/
```

CAN_INTBusErrorEn

原型：void CAN_INTBusErrorEn(CAN_TypeDef * CANx)

```
/*******************************************************************************
* 函数名称:      CAN_INTBusErrorEn()
* 功能说明:      总线错误中断使能
* 输    入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输    出: 无
* 注意事项: 无
```

```
*****************************************************************************/
```

CAN_INTBusErrorDis

原型：void CAN_INTBusErrorDis(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_INTBusErrorDis()
* 功能说明:      总线错误中断禁止
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

CAN_INTBusErrorStat

原型：uint32_t CAN_INTBusErrorStat(CAN_TypeDef * CANx)

```
/*****************************************************************************
* 函数名称:      CAN_INTBusErrorStat()
* 功能说明:      总线错误中断是否触发
* 输      入: CAN_TypeDef * CANx      指定要被设置的 CAN 接口，有效值包括 CAN
* 输      出: uint32_t                1  已触发      0  未触发
* 注意事项: 无
*****************************************************************************/
```

2.14 实时时钟（RTC）

2.14.1介绍

RTC 控制器用于提供给用户实时的时间信息与日期信息。

2.14.2特性

- 可自由设置日期（年、月、周、日）和时间（时、分、秒）
- 可自由设置闹钟（周、 时、分、秒）
- 自动识别当前设置年份是否为闰年
- 支持 RTC 时钟校正功能
- 支持 RTC 中断从 SLEEP 模式下唤醒芯片

2.14.3常量定义

常量名	值	描述
RTC_SUN	0x01	周计数器__周日
RTC_MON	0x02	周计数器__周一

RTC_TUE	0x04	周计数器__周二
RTC_WED	0x08	周计数器__周三
RTC_THU	0x10	周计数器__周四
RTC_FRI	0x20	周计数器__周五
RTC_SAT	0x40	周计数器__周六

2.14.4类型定义

RTC_InitStructure

```
typedef struct {
    uint16_t Year;
    uint8_t  Month;
    uint8_t  Date;
    uint8_t  Hour;
    uint8_t  Minute;
    uint8_t  Second;
    uint8_t  SecondIEn;
    uint8_t  MinuteIEn;
} RTC_InitStructure;
```

成员	描述
Year	年计数器
Month	月计数器
Date	日期计数器
Hour	时计数器
Minute	分计数器
Second	秒计数器
SecondIEn	秒中断使能
MinuteIEn	分中断使能

RTC_AlarmStructure

```
typedef struct {
    uint8_t  Days;
    uint8_t  Hour;
    uint8_t  Minute;
    uint8_t  Second;
    uint8_t  AlarmIEn;
} RTC_AlarmStructure;
```

成员	描述
Days	闹钟计数器周设置
Hour	闹钟计数器时设置
Minute	闹钟计数器分设置
Second	闹钟计数器秒设置
AlarmIEn	闹钟使能

RTC_DateTime

```
typedef struct {
    uint16_t Year;
    uint8_t  Month;
    uint8_t  Date;
    uint8_t  Day;
    uint8_t  Hour;
    uint8_t  Minute;
    uint8_t  Second;
} RTC_DateTime;
```

成员	描述
Year	获取当前年份
Month	获取当前月份
Date	获取当前日期
Day	获取当前星期
Hour	获取当前时
Minute	获取当前分
Second	获取当前秒

2.14.5 函数

RTC_Init

原型：void RTC_Init(RTC_TypeDef * RTCx, RTC_InitStructure * initStruct)

```
/*****************************************************************************
* 函数名称:      RTC_Init()
* 功能说明:      RTC 初始化
* 输      入: RTC_TypeDef * RTCx      指定要被设置的 RTC，有效值包括 RTC
*                RTC_InitStructure * initStruct      包含 RTC 相关设定值的结构体
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

RTC_Start

原型：void RTC_Start(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_Start()
* 功能说明:      启动 RTC
* 输      入: RTC_TypeDef * RTCx      指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

RTC_Stop

原型：void RTC_Stop(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_Stop()
* 功能说明:      停止 RTC
* 输      入: RTC_TypeDef * RTCx      指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

RTC_GetDateTime

原型：void RTC_GetDateTime(RTC_TypeDef * RTCx, RTC_DateTime * dateTime)

```
/*****************************************************************************
* 函数名称:      RTC_GetDateTime()
* 功能说明:      获取当前的时间和日期
* 输      入: RTC_TypeDef * RTCx      指定要被设置的 RTC，有效值包括 RTC
*                RTC_DateTime * dateTime      获取到的时间、日期值存入此指针指向的结构体
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

RTC_AlarmSetup

原型：void RTC_AlarmSetup(RTC_TypeDef * RTCx, RTC_AlarmStructure * alarmStruct)

```
/*****************************************************************************
* 函数名称:      RTC_AlarmSetup()
* 功能说明:      RTC 闹钟设定
* 输      入: RTC_TypeDef * RTCx      指定要被设置的 RTC，有效值包括 RTC
*                RTC_AlarmStructure * alarmStruct      包含 RTC 闹钟设定值的结构体
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

calcWeekDay

原型：static uint32_t calcWeekDay(uint32_t year, uint32_t month, uint32_t date)

```
/*****************************************************************************
* 函数名称:      calcWeekDay()
* 功能说明:      计算指定年、月、日是星期几
* 输      入: uint32_t year          年
*                uint32_t month          月
*                uint32_t date          日
* 输      出: uint32_t              0  星期日      1  星期一      ... ...      6  星期六
* 注意事项: 无
*****************************************************************************/
```

RTC_IntSecondEn

原型：void RTC_IntSecondEn(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntSecondEn()
* 功能说明:      秒中断使能
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

RTC_IntSecondDis

原型：void RTC_IntSecondDis(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntSecondDis()
* 功能说明:      秒中断禁止
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

RTC_IntSecondClr

原型：void RTC_IntSecondClr(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntSecondClr()
* 功能说明:      秒中断标志清除
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

RTC_IntSecondStat

原型：uint32_t RTC_IntSecondStat(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntSecondStat()
* 功能说明:      秒中断状态
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: uint32_t                    1 秒中断发生      0 秒中断未发生
* 注意事项: 无
*****************************************************************************/
```

RTC_IntMinuteEn

原型：void RTC_IntMinuteEn(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntMinuteEn()
* 功能说明:      分中断使能
```

```
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

RTC_IntMinuteDis

原型：void RTC_IntMinuteDis(RTC_TypeDef * RTCx)

```
/*******************************************************************************
* 函数名称:      RTC_IntMinuteDis()
* 功能说明:      分中断禁止
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

RTC_IntMinuteClr

原型：void RTC_IntMinuteClr(RTC_TypeDef * RTCx)

```
/*******************************************************************************
* 函数名称:      RTC_IntMinuteClr()
* 功能说明:      分中断标志清除
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

RTC_IntMinuteStat

原型：uint32_t RTC_IntMinuteStat(RTC_TypeDef * RTCx)

```
/*******************************************************************************
* 函数名称:      RTC_IntMinuteStat()
* 功能说明:      分中断状态
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: uint32_t                    1 分中断发生      0 分中断未发生
* 注意事项: 无
*******************************************************************************/
```

RTC_IntHourEn

原型：void RTC_IntHourEn(RTC_TypeDef * RTCx)

```
/*******************************************************************************
* 函数名称:      RTC_IntHourEn()
* 功能说明:      时中断使能
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

RTC_IntHourDis

原型：void RTC_IntHourDis(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntHourDis()
* 功能说明:      时中断禁止
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

RTC_IntHourClr

原型：void RTC_IntHourClr(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntHourClr()
* 功能说明:      时中断标志清除
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

RTC_IntHourStat

原型：uint32_t RTC_IntHourStat(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntHourStat()
* 功能说明:      时中断状态
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: uint32_t                    1 时中断发生      0 时中断未发生
* 注意事项: 无
*****************************************************************************/
```

RTC_IntDateEn

原型：void RTC_IntDateEn(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntDateEn()
* 功能说明:      日中断使能
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

RTC_IntDateDis

原型：void RTC_IntDateDis(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntDateDis()
* 功能说明:      日中断禁止
```

```
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

RTC_IntDateClr

原型：void RTC_IntDateClr(RTC_TypeDef * RTCx)

```
/*******************************************************************************
* 函数名称:      RTC_IntDateClr()
* 功能说明:      日中断标志清除
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

RTC_IntDateStat

原型：uint32_t RTC_IntDateStat(RTC_TypeDef * RTCx)

```
/*******************************************************************************
* 函数名称:      RTC_IntDateStat()
* 功能说明:      日中断状态
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: uint32_t                    1 日中断发生      0 日中断未发生
* 注意事项: 无
*******************************************************************************/
```

RTC_IntAlarmEn

原型：void RTC_IntAlarmEn(RTC_TypeDef * RTCx)

```
/*******************************************************************************
* 函数名称:      RTC_IntAlarmEn()
* 功能说明:      闹钟中断使能
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

RTC_IntAlarmDis

原型：void RTC_IntAlarmDis(RTC_TypeDef * RTCx)

```
/*******************************************************************************
* 函数名称:      RTC_IntAlarmDis()
* 功能说明:      闹钟中断禁止
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*******************************************************************************/
```

RTC_IntAlarmClr

原型：void RTC_IntAlarmClr(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntAlarmClr()
* 功能说明:      闹钟中断标志清除
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

RTC_IntAlarmStat

原型：uint32_t RTC_IntAlarmStat(RTC_TypeDef * RTCx)

```
/*****************************************************************************
* 函数名称:      RTC_IntAlarmStat()
* 功能说明:      闹钟中断状态
* 输      入: RTC_TypeDef * RTCx          指定要被设置的 RTC，可取值包括 RTC
* 输      出: uint32_t                    1 闹钟中断发生      0 闹钟中断未发生
* 注意事项: 无
*****************************************************************************/
```

2.15 FLASH

2.15.1介绍

FLASH 通过内置 IAP 函数进行擦除及写入。 IAP 函数作为片内驻留程序，其提供了针对 flash 的相关操作。

2.15.2函数

FLASH_Erase

原型：void FLASH_Erase(uint32_t addr)

```
/*****************************************************************************
*************************************************************
* 函数名称: FLASH_Erase()
* 功能说明:      片内 Flash 擦除
* 输      入: uint32_t addr              擦除地址
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

FLASH_Write

原型：void FLASH_Write(uint32_t addr, uint32_t buff[], uint32_t size)

```
/*****************************************************************************
```

```
*****************************************************************
* 函数名称: FLASH_Write()
* 功能说明:      片内 Flash 写入
* 输      入: uint32_t addr                写入地址
*                uint32_t buff[]              要写入的数据
*                uint32_t size                要写入数据的个数，字为单位
* 输      出: 无
* 注意事项: 无
*****************************************************************************/
```

3 版本记录

版本	修改日期	说明
V1.00	2016.8.25	文档发布

Important Notice

Synwit Products are neither intended nor warranted for usage in systems or equipment, any malfunction or failure of which may cause loss of human life, bodily injury or severe property damage. Such applications are deemed, “Insecure Usage”.

Insecure usage includes, but is not limited to: equipment for surgical implementation, atomic energy control instruments, airplane or spaceship instruments, the control or operation of dynamic, brake or safety systems designed for vehicular use, traffic signal instruments, all types of safety devices, and other applications intended to support or sustain life.

All Insecure Usage shall be made at customer’s risk, and in the event that third parties lay claims to Synwit as a result of customer’s Insecure Usage, customer shall indemnify the damages and liabilities thus incurred by Synwit.