SINO WEALTH

SH33F2801

内置高性能电机接口、基于ARM® “星辰”处理器的32位微控制器

1. 特性

- **内核： ARMv8-M架构32位“星辰”处理器**
  - 系统主频最高84MHz
  - 支持DSP和FPU（浮点运算单元）
  - 内置硬件32位乘/除法
  - 指令集兼容Cortex-M4
- **集成数学运算协处理器（MACP）**
  - 内含32位CORDIC运算单元，支持最大IQ31格式正余弦、反正切和求模、乘除法、指数、对数和开方等运算
  - 内置FOC专用硬件SVPWM引擎，支持五段式和七段式
- **存储器**
  - 128K字节\128位宽FLASH程序存储器
  - 16K字节 RAM
  - 4K字节类EEPROM，1K字节OTP区
  - ISP功能
- **时钟、复位和电源管理**
  - 2.4V～5.5V数字供电（VDD）和I/O引脚
  - 上电复位(POR)、掉电检测(BOD)、低电压复位(LVR)
  - 硬件独立看门狗(IWDT)、窗口看门狗(WWDT)
  - 内部集成PLL，无需外接电容，最高输出84MHz高频时钟
  - 4M～12MHz晶体振荡器/陶瓷振荡器
  - 内嵌高频RC振荡器：8MHz（常温0.2%精度）
  - 内嵌低频RC振荡器：128kHz
  - 时钟停振监控
- **专用电机控制模块（MCM）**
  - 提供6路（3对）PWM输出，16位分辨率，每对PWM支持互补或独立模式，输出极性可单独控制
  - 3种计数模式，单次计数模式，边沿对齐模式，中心对齐模式（支持对称与非对称波形）
  - 带死区产生逻辑及故障检测功能
  - PWM周期内可设置多个时刻硬件触发ADC功能
  - 支持硬件相移和逐波限流功能
- **12位高速模数转换器**
  - 2MSPS最高采样速率
  - 16个模拟采样信道，16组结果缓存寄存器
  - 支持单次/连续/同步/独立采样模式和事件触发
  - 针对电机控制独有的单电阻电流采样模式
- **最多45个快速GPIO端口**
  - 所有GPIO内建上拉电阻
  - 45 I/Os TQFP48， 29 I/Os LQFP32
- **8个外部中断输入通道（EXTI0～EXTI7）**
  - 支持边沿触发与电平触发、支持软件触发外部中断
  - 两路中断输入通道可唤醒停机模式
- **内建的低电压复位功能**
  - LVR电压1：4.1V, LVR电压2：2.8V
- **2个32位可编程计数器阵列（PCA）**
  - 能够最多实现6路PWM输出
  - 支持6通道输入捕获功能
  - 支持故障检测功能
- **2个16位基本定时器/计数器（可级联）**
  - 自动重载累加计数器
  - 硬件触发ADC
  - 可在低功耗模式下计数
- **32位正交编码器接口模块（QEI）**
  - 可自动实现高低速信号捕获
  - 内含独立的32位定时器，可单独使用
- **24位自减型系统时间定时器（SYSTICK）**
- **内建代码和数据完整性检测模块（CRC）**
  - 硬件CRC模块，多种生成多项式、代码校验、数据校验
  - CRC数据传输校验
- **多个串行通信接口**
  - 3个通用异步串行口（UART0/1/2）
  - 1个独立串行外设接口（SPI）
  - 1个独立两线串行接口TWI
  - UART和SPI支持FIFO收发（FIFO 0/1/2）
- **CAN控制器**
  - 符合CAN 2.0B规范
  - 15个信息缓冲区，支持远程帧，接收过滤器
- **丰富的片上模拟功能模块（AMOC）**
  - 4个独立的模拟放大器(且一个OP也复用为比较器)
  - 1个独立模拟比较器
- **内置电源电压检测（BOD）**
- **低功耗模式**
  - 睡眠模式（仅CPU停止）
  - 停机模式（CPU停止，外设时钟切断）
- **支持代码全方位保护**
  - 多重读保护、写保护、擦除保护
  - 支持客户安全码保护
  - 调试口切断保护
  - 防止侵入型攻击Flash保护
- **96位芯片唯一识别码**
- **标准SWD烧写/仿真接口**
- **工作环境温度：-40℃～+105℃**
- **封装：TQFP48，LQFP32**

2. 概述

SH33F2801是基于ARM “星辰”处理器的高性能32位微控制器，支持84MHz最高主频，由于采用特殊结构的存储器指令加速模式，可使系统获得高运行效率。

ARM “星辰”处理器具有成熟可靠的系统结构，高效的指令执行效率以及完备的生态系统。SH33F2801充分发挥了ARM “星辰”处理器的优势，提供高集成度、高运算能力的单片集成电路，提供高性价比的微控制器系统解决方案。

在运算能力方面，ARM “星辰”处理器已内置了硬件32位乘除法器，具有强大的计算能力。在此基础上，SH33F2801进一步提供了数学运算协处理单元（MACP），提供硬件CORDIC运算，可实现快速的正余弦、反正切、求模运算及Park变换和逆Park变换；提供硬件SVPWM电路，可实现纯硬件的五段式和七段式SVPWM波形生成。同时SH33F2801内置了一个独立的三相电机专用PWM模块（MCM），还有两个可编程计数器阵列模块（PCA），可实现32位PWM输出和3通道输入捕捉定时功能。SH33F2801还集成了多个运算放大器和模拟比较器，一个12位多通道高速ADC，使之非常适合高性能电机控制应用，尤其适合各种直流无刷电机（BLDC）和永磁同步电机（PMSM）的控制。

SH33F2801提供8/16/32位可选的CRC代码校验和数据校验，SRAM自检算法，支持低功耗应用模式，内置独立看门狗、窗口看门狗、低电压复位电路、掉电检测电路等辅助模块，能够有效提升系统可靠性，可应用于高安规要求场合。

SH33F2801具有代码全方位保护机制，提供多重代码读保护功能、写保护功能、擦除保护功能，并且调试口可做切断保护，支持客户安全码设定功能，还增加了防侵入型攻击Flash的保护功能。

3. 方框图

ARM Star
84MHz
Systick
NVIC
Bus
Matrix
Flash 128KB
E2Prom 4KB
OTP 1KB
SRAM 16KB
AHB / APB Bus
MACP (CORDIC / SVPWM)
CRC / RAMBIST
MCM
PCA0/1
TIM7/8
External Interrupt (0~7)
UARTs 0/1/2
SPI
TWI
CAN
Clock Generator (RC8M / Crystal / PLL)
Power Module
VDD
GND
Reset Circuit
NRST
IWDT
WWDT
GPIOA (16 channel)
PA0~PA15
GPIOB (16 channel)
PB0~PB15
GPIOC (13 channel)
PC0~PC12
ADC
CMPs 0/1
OPAs 0/1/2/3
SW debug port

4. 引脚定义

48引脚TQFP封装

32引脚LQFP封装

SH33F2801
(LQFP32)
PC0/OP3N/TXD1
PB15/OP3P/CP1P1/SS
GND
PB14/T8/CAN_RX/AN6/VREF
VDD
PB13/AN5/CP1P0/SCK/PCA0B
PB12/CP1N/MOSI/PCA1B
PB11/AN4/MISO/CP1OUT
24 23 22 21 20 19 18 17
RXD1/OP3OUT/AN14/PC1 25
PCA0C/OP2OUT/AN13/PC2 26
PCA0B/OP2N/PC3 27
PCA0A/OP2P/PC4 28
OP0P0/AN7/PC8 29
PCA1C/QEA/OP0P1/AN8/PC9 30
PCA1B/QEB/OP0P2/AN9/PC10 31
PCA1A/INDEX/OP0N/AN10/PC11 32
16 PB6/PWM0
15 PB5/PWM1
14 PB4/PWM2
13 PB3/PWM01
12 PB2/PWM11
11 PB1/PWM21
10 PB0/PWMFLT/PCA1ECI/SS/EXTI0
9 PA14/ADTRG/SCL/CAN_TX/TXD1/PCA1C
1 2 3 4 5 6 7 8
EXTI7/PCA1FLT/OP0OUT/AN11/CP0OUT/PC12
NRST
XTAL2/RXD2/CP0OUT/T7/SCL/PA0
ADTRG/XTAL1/EXTI5/TXD2/CP1OUT/T8/SDA/PA2
RXD0/SWDIO/PA7
TXD0/SWCLK/PA8
BOOT/EXTI2/SC1/SCK/CAN_TX/PCA0B/PA10
EXTI1/SDA/CAN_RX/RXD1/PCA1B/PA13

5. 引脚描述

5.1 复用功能映射表

(注：AF0-AF7为软件可修改的8种模式，系统复位后除了调试口SWCLK、SWDIO，其他端口默认为AF0功能全关。)

PORT	SYS	AF0	AF1	AF2	AF3	AF4	AF5	AF6	AF7	TQF P48	LQF P32
PA0	XTAL2	OFF	GPIO		T7	RXD2	SCL	CP0OUT		Y	Y
PA1		OFF	GPIO	SC0	PCA0ECI			CAN_TX		Y	
PA2	XTAL1	OFF	GPIO	ADTRG	T8	TXD2	SDA	CP1OUT		Y	Y
PA3		OFF	GPIO		T7			CAN_RX		Y	
PA4		OFF	GPIO		PCA0FLT		MISO			Y	
PA5		OFF	GPIO	ADTRG	PCA1B		MOSI			Y	
PA6		OFF	GPIO		PCA0B	RXD2	SCK			Y	
PA7	SWDIO	OFF	GPIO			RXD0				Y	Y
PA8	SWCLK	OFF	GPIO			TXD0				Y	Y
PA9		OFF	GPIO		PCA0A	TXD2	SS			Y	
PA10		OFF	GPIO	SC1	PCA0B		SCK	CAN_TX		Y	Y
PA11		OFF	GPIO		PCA0C	RXD0				Y	
PA12		OFF	GPIO		PCA1A	TXD0				Y	
PA13		OFF	GPIO		PCA1B	RXD1	SDA	CAN_RX		Y	Y
PA14		OFF	GPIO	ADTRG	PCA1C	TXD1	SCL	CAN_TX		Y	Y
PA15		OFF	GPIO	SC2	PCA1FLT					Y	
PB0		OFF	GPIO	PWMFLT	PCA1ECI		SS			Y	Y
PB1		OFF	GPIO	PWM21						Y	Y
PB2		OFF	GPIO	PWM11						Y	Y
PB3		OFF	GPIO	PWM01						Y	Y
PB4		OFF	GPIO	PWM2						Y	Y
PB5		OFF	GPIO	PWM1						Y	Y
PB6		OFF	GPIO	PWM0						Y	Y
PB7		OFF	GPIO	SC0				QEA	AN0	Y	
PB8		OFF	GPIO	SC1				QEB	AN1	Y	
PB9		OFF	GPIO	SC2				INDEX	AN2	Y	
PB10		OFF	GPIO						AN3	Y	
PB11		OFF	GPIO				MISO	CP1OUT	AN4	Y	Y
PB12		OFF	GPIO		PCA1B		MOSI		CP1N	Y	Y
PB13		OFF	GPIO		PCA0B		SCK		AN5/CP1P0	Y	Y
PB14		OFF	GPIO		T8			CAN_RX	AN6/VREF	Y	Y
PB15		OFF	GPIO				SS		OP3P/CP1P1	Y	Y
PC0		OFF	GPIO			TXD1			OP3N	Y	Y
PC1		OFF	GPIO			RXD1			AN14/OP3OUT	Y	Y
PC2		OFF	GPIO		PCA0C				AN13/OP2OUT	Y	Y
PC3		OFF	GPIO		PCA0B				OP2N	Y	Y
PC4		OFF	GPIO		PCA0A				OP2P	Y	Y
PC5		OFF	GPIO						AN12/OP1OUT	Y	
PC6		OFF	GPIO	ADTRG					OP1N	Y	
PC7		OFF	GPIO						OP1P_CP1P2	Y	
PC8		OFF	GPIO						AN7/OP0P0/CP0P0	Y	Y
PC9		OFF	GPIO		PCA1C			QEA	AN8/OP0P1/CP0P1	Y	Y
PC10		OFF	GPIO		PCA1B			QEB	AN9/OP0P2/CP0P2	Y	Y
PC11		OFF	GPIO		PCA1A			INDEX	AN10/OP0N/CP0N	Y	Y
PC12		OFF	GPIO		PCA1FLT			CP0OUT	AN11/OP0OUT	Y	Y

5.2 引脚功能描述

引脚编号	类型	说明
I/O端口		
PA0 – PA15	I/O	16位双向I/O端口
PB0 – PB15	I/O	16位双向I/O端口
PC0 – PC12	I/O	13位双向I/O端口
可编程计数器阵列（PCA0～PCA1，支持多通道输入捕获/输出比较/PWM输出）		
PCA0A	I/O	PCA0通道A
PCA0B	I/O	PCA0通道B
PCA0C	I/O	PCA0通道C
PCA0FLT	I	PCA0故障信号输入
PCA0ECI	I	PCA0外部时钟输入
PCA1A	I/O	PCA1通道A
PCA1B	I/O	PCA1通道B
PCA1C	I/O	PCA1通道C
PCA1FLT	I	PCA1故障信号输入
PCA1ECI	I	PCA1外部时钟输入
增量编码器（QEA/QEB/INDEX）		
QEA	I	增量编码器QEA输入
QEB	I	增量编码器QEB输入
INDEX	I	增量编码器INDEX输入
基本定时器（TIM7~8）		
T7	I/O	外部时钟输入/时钟比较输出
T8	I/O	外部时钟输入/时钟比较输出
MCM（MCM，三相电机控制模块）		
PWM0/PWM01 PWM1/PWM11 PWM2/PWM21	O	MCM模块 16BIT PWM互补输出
PWMFLT	I	MCM模块故障输入信号
SC0	I	MCM模块逐波限流输入比较数字信号0
SC1	I	MCM模块逐波限流输入比较数字信号1
SC2	I	MCM模块逐波限流输入比较数字信号2
ADC（ADC, 多通道模数转换器）		
AN0~14	I	ADC独立输入通道（15个）
VREF	I/O	外部参考电压输入接口
ADTRG	I/O	ADC外部采样触发信号输入/内部采样信号输出

中断，复位，时钟，电源		
NRST	I	该引脚上保持10μs以上的低电平，CPU将复位。由于有内建30kΩ上拉电阻连接到V_{DD}，所以仅接一个外部电容即可产生复位脉冲。
BOOT	I	BOOT=1为从Flash启动，BOOT=0为从ISP区启动。
XTAL1	I	谐振器输入
XTAL2	O	谐振器输出
GND	P	数字接地
V_{DD}	P	数字电源（2.4 - 5.5V）
UART（UART0/1/2，通用同步异步串行口）		
RXD0/1/2	I/O	串口0/1/2数据输入
TXD/01/2	O	串口0/1/2数据输出
SPI（串行外设接口）		
MOSI	I/O	SPI主输出从输入引脚
MISO	I/O	SPI主输入从输出引脚
SCK	I/O	SPI串行时钟引脚
SS	I	SPI从设备选择引脚
TWI（两线串行接口）		
SCL	I/O	TWI时钟信号（支持主机/从机模式）
SDA	I/O	TWI数据输入输出
CAN（CAN接口）		
CAN_RX	I/O	CAN模块数据输入
CAN_TX	I/O	CAN模块数据输出
片上模拟比较器(CP0/CP1)		
CP0P0/ CP0P1/ CP0P2 CP1P0/ CP1P1/ CP1P2	I	比较器0/1同相3路输入端
CP0N/CP1N	I	比较器0/1反向输入端
CP0OUT/CP1OUT	O	比较器0/1输出端（数字信号）
片上模拟放大器（OP0/OP1/OP2/OP3，4个独立运放，只支持外部增益模式，OP0具有3路同向输入，3路反向输入）		
OP0P0/OP0P1/OP0P2/ OP1P/OP2P/OP3P	I	放大器0～3同相输入端
OP0N/ OP1N/OP2N/OP3N	I	放大器0～3反向输入端
OP0OUT/OP1OUT/OP2 OUT/OP3OUT	O	放大器0～3输出端
调试接口（SWD，串行双线调试接口）		
SWDIO	I/O	SWD数据输入输出
SWCLK	I	SWD时钟输入

注意：

注1：系统上电这2个接口默认是调试接口，有默认上下拉，具体参考"GPIO"章节。

注2：这2个调试接口可以复用为GPIO口，但注意上电时的上下拉影响，具体参考"SYSCFG"章节。

6. 相关文档

- ARM “星辰”处理器技术参考手册
- ARM “星辰”处理器 通用用户指南（含架构、指令集、核内外设等）
- 下述文档可在Sinowealth网站下载（http://www.sinowealth.com/下载专区）：

数据手册（Data Sheet）： 芯片功能和特性以数据手册为准；

闪存编程手册（Flash Programming Manual）：flash（含内置类EEPROM）的读取、擦除、编程以及保护等介绍；

应用笔记（Application Note）：芯片各模块的应用注意事项、模式典型配置、功能常规用法介绍等；

标准库手册（StdLib Manual）：基于库的形式提供的芯片各模块的应用参考手册；

开发工具包（Development Kit）：芯片开发涉及各项软硬件配套工具，包含各支持ARM “星辰”处理器开发的IDE工具（Keil、IAR等）插件、工程建立向导、行业专用辅助工具、硬件评估平台介绍等。

7. 内核

7.1 ARM® “星辰”处理器

请参考相关文档。

7.2 寄存器组

7.2.1 ARM “星辰”处理器拥有 R0-R15 的寄存器组。其中 R13 作为堆栈指针 SP。通用寄存器

R0-R12都是32位通用寄存器，用于数据操作。

注意：*绝大多数16位Thumb指令只能访问R0-R7，而32位Thumb-2指令可以访问所有寄存器。*

7.2.2 堆栈寄存器

R13：分组的堆栈指针别名，SP_process和SP_main

ARM “星辰”处理器拥有两个分组堆栈指针，任一时刻只能使用其中的一个。

主堆栈指针（MSP）：复位后缺省使用主堆栈指针，所有异常处理使用主堆栈

进程堆栈指针（PSP）：进程模式下使用进程堆栈指针。

堆栈指针的最低两位永远是0，这意味着堆栈总是4字节对齐的。

结束复位后，所有代码都使用主堆栈。异常处理程序（例如SVC）可以通过改变其在退出时使用的EXC_RETURN值来改变线程模式使用的堆栈。所有异常继续使用主堆栈，堆栈指针R13是分组寄存器，在SP_main和SP_process之间切换。在任何时候，进程堆栈和主堆栈中只有一个是可见的，由R13指示。

除了使用从处理者模式（Handler）退出时的EXC_RETURN的值外，在线程模式中，使用MSR指令对CONTROL[1]执行写操作也可以从主堆栈切换到进程堆栈

在ARM编程领域中，凡是打断程序顺序执行的事件，都被称为异常（exception）。除了外部中断外，当有指令执行了“非法操作”，或者访问被禁的内存区间，因各种错误产生的fault，以及不可屏蔽中断发生时，都会打断程序的执行，这些情况统称为异常。在不严格的上下文中，异常与中断也可以混用。另外，程序代码也可以主动请求进入异常状态的（常用于系统调用）。

7.2.3 连接寄存器

R14为连接寄存器，当调用一个子程序时，由R14存储返回地址。

区别于其它处理器内核，ARM为了减少访问内存的次数，把返回地址直接存储在寄存器中。这样足以使很多只有1级子程序调用的代码无需访问内存（堆栈内存），从而提高了子程序调用的效率。如果多于1级，则需要把前一级的R14值压到堆栈里。在ARM上编程时，应尽量只使用寄存器保存中间结果，减少访问内存。

7.2.4 程序计数寄存器

R15为程序计数寄存器，指向当前的程序地址。如果修改它的值，就能改变程序的执行流。

7.3 特殊功能寄存器

ARM “星辰”处理器还在内核水平上搭载了若干特殊功能寄存器，如下表：

表7-1 ARM “星辰”处理器特殊功能寄存器及其功能

寄存器	功能
xPSR	记录ALU标志（0标志，进位标志，负数标志，溢出标志），执行状态，当前正服务的中断号
PRIMASK	除能所有可屏蔽中断——NMI和硬fault可以响应
FAULTMASK	除能所有fault——除了NMI，所有faults及可屏蔽中断都不响应
BASEPRI	除能所有优先级不高于某个具体数值的可屏蔽中断
CONTROL	定义特权状态，并且决定使用哪一个堆栈指针
MSPLIM \PSPLIM	栈指针上限寄存器

7.3.1 程序状态字寄存器组

系统级的处理器状态可分为3类，因此有3个程序状态寄存器。对程序状态寄存器的访问使用MRS和MSR指令，在访问时可以把它们作为单独的寄存器，3个中的任两个组合，或3个组合。这3个寄存器为：应用PSR（APSR），中断PSR（IPSR），执行PSR（EPSR）。

7.3.1.1 应用PSR

应用PSR（APSR）包含条件代码标志。在进入异常之前，ARM “星辰”处理器将条件代码标志保存在堆栈内。可以使用MSR和MRS指令来访问APSR。

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
N	**Z**	**C**	**V**	**Q**	保留										
RW	*RW*	*RW*	*RW*	*RW*	-										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31	**N**	**负数或小于标志** 1：结果为负数或小于 0：结果为正数或大于
30	**Z**	**零标志** 1：结果为0 0：结果为非0
29	**C**	**进位/借位标志** 1：进位或借位 0：没有进位或借位
28	**V**	**溢出标志** 1：溢出 0：没有溢出
27	**Q**	**DSP溢出及粘着饱和（sticky saturation）标志**
26 - 0	保留	-

7.3.1.2 中断PSR

中断PSR（IPSR）包含当前激活的异常的ISR编号。

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留							中断号								
-							*RW*								
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 - 9	保留	**-**
8 - 0	**ISR NUMBER**	占先异常的编号 0 = Thread mode. 1 = Reset. 2 = NMI. 3 = Hard Fault. 4 = Mem Manage. 5 = Bus Fault. 6 = Usage Fault 7 = Secure Fault 8-10 = Reserved. 11 = SVCall. 12 = Debug Monitor. 13 = Reserved. 14 = PendSV. 15 =SysTick 16 = IRQ0 … … 54 = IRQ28 其它= Reserved.

7.3.1.3 执行PSR

执行PSR（EPSR）包含两个重叠的区域：一个是 可中断-可继续（interruptible-continuable）指令（ICI）区，用于被打断的多寄存器加载和存储指令。另一个是用于If-Then（IT）指令的执行状态区，以及T位（Thumb状态位）。

可中断-可继续指令（ICI）区

多寄存器加载（LDM）和存储（STM）操作是可中断的。EPSR的ICI区用来保存从产生中断的点继续执行多寄存器加载和存储操作时所必需的信息。

If-then状态区

EPSR的IT区包含了If-Then指令的执行状态位。不能直接访问EPSR，若想修改EPSR必须发生以下两个事件之一：

在执行LDM或STM指令时产生一次中断

执行If-Then指令

注意：*ICI区和IT区是重叠的，因此，If-Then模块内的多寄存器加载或存储操作不具有可中断-可继续功能。*

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留					ICI/T		T	保留							
-					*RW*		*RW*	-							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	B5	b4	b3	b2	b1	b0
ICT/IT						保留									
RO						-									
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 - 27	保留	-
15 - 12	**ICI**	可中断-可继续的指令位。如果在执行LDM或STM操作时产生一次中断，则LDM或STM操作暂停。EPSR使用位[15:12]来保存该操作中下一个寄存器操作数的编号。在中断响应之后，处理器返回由[15:12]指向的寄存器并恢复操作。如果ICI区指向的寄存器不在指令的寄存器列表中，则处理器对列表中的下一个寄存器（如果有）继续执行LDM/STM操作。
15 - 10:26 - 25	**IT**	If-Then位。它们是If-Then指令的执行状态位。包含If-Then模块的指令数目和它们的执行条件。
24	**T**	T位使用一条可相互作用的指令来清零，这里写入的PC的位0为0。也可以使用异常出栈操作来清零，被压栈的T位为0。当T位为零时执行指令会引起INVSTATE异常。
23 - 16	保留	-
9 - 0	保留	-

LDM和STM操作中的基址寄存器更新

以下是LDM或STM更新基址寄存器的情况：

- 当指令指定了基址寄存器回写操作时，基址寄存器变为更新后的地址。一次中止（abort）能够恢复为原来的基地址。
- 当基址寄存器在 LDM 的寄存器列表中并且不是列表中的最后一个寄存器时，基址寄存器变为被加载的值。

如果出现下列情况，LDM/STM 操作重新开始而不是继续执行：

- LDM/STM 错误
- LDM/STM 指令位于 IT 内

如果 LDM 已经完成基地址加载操作，它会从基地址加载之前的地址继续执行。

保存 xPSR位

在进入异常时，处理器将 3 个状态寄存器组合的信息保存在堆栈内。

7.3.2 中断屏蔽寄存器组

中断屏蔽寄存器组（PRIMASK, FAULTMASK, BASEPRI）

名字	功能描述
PRIMASK	这是个只有1个位的寄存器。当它置1时，就关掉所有可屏蔽的异常，只剩下NMI和硬 fault可以响应。它的缺省值是 0，表示没有关中断。
FAULTMASK	这是个只有1个位的寄存器。当它置 1时，只有NMI 才能响应，所有其它的异常，包括中断和 fault，都无法响应。它的缺省值也是0，表示没有关异常。
BASEPRI	这个寄存器最多有8位（由表达优先级的位数决定）。它定义了被屏蔽优先级的阈值。当它被设成某个值后，所有优先级号大于等于此值的中断都被关（优先级号越大，优先级越低）。但若被设成0，则不关闭任何中断，0 也是缺省值。

要访问 PRIMASK, FAULTMASK 以及 BASEPRI，同样要使用 MRS/MSR 指令,如：

```
MRS   R0,   BASEPRI      ;读取 BASEPRI 到 R0中
MRS   R0,   FAULTMASK  ;读取 FAULTMASK 到 R0中
MRS   R0,   PRIMASK      ;读取 PRIMASK 到 R0中
MSR   BASEPRI,   R0       ;写入 R0到BASEPRI 中
MSR   FAULTMASK, R0    ;写入 R0到FAULTMASK 中
MSR   PRIMASK,   R0       ;写入 R0到PRIMASK 中
```

只有在特权级下，才允许访问这 3个寄存器。

为了快速地开关中断，内核还专门设置了一条 CPS指令，有4种用法

```
CPSID I  ;PRIMASK=1       ;关中断
CPSIE I  ;PRIMASK=0       ;开中断
CPSID F ;FAULTMASK=1   ;关异常
CPSIE F ;FAULTMASK=0   ;开异常
```

7.3.3 控制寄存器

控制寄存器（CONTROL）用于定义特权级别，还用于选择当前使用哪个堆栈指针。

名字	功能描述
CONTROL[1]	**堆栈指针选择** 0=选择主堆栈指针MSP（复位后缺省值） 1=选择进程堆栈指针 PSP 在线程或基础级，可以使用 PSP。在 Handler 模式下，只允许使用MSP，所以此时不得往该位写 1。 (Handler 模式中，CONTROL[1]总是 0。在线程模式中则可以为 0 或 1。仅当处于特权级的线程模式下，此位才可写，其它场合下禁止写此位。改变处理器的模式也有其它的方式：在异常返回时，通过修改 LR 的位 2，也能实现模式切换。)
CONTROL[0]	0=特权级的线程模式 1=用户级的线程模式 Handler模式永远都是特权级的。 (仅当在特权级下操作时才允许写该位。一旦进入了用户级，唯一返回特权级的途径，就是触发一个（软）中断，再由服务例程改写该位。)

CONTROL 寄存器是通过 MRS和 MSR 指令来操作的：

```
MRS   R0,          CONTROL
MSR   CONTROL,  R0
```

7.4 模式、状态和访问方式

7.4.1 工作模式

ARM “星辰”处理器支持两种工作模式，线程模式和处理模式(Handler)：

在复位时处理器进入线程模式，异常返回时也会进入该模式。特权和用户（非特权）代码能够在线程模式下运行。

出现异常时处理器进入处理模式，在处理模式中，所有代码都是特权访问的。

7.4.2 工作状态

ARM “星辰”处理器 处理器有两种工作状态：

Thumb 状态：这是 16 位和 32 位半字对齐的 Thumb 和 Thumb-2 指令的正常执行状态。

调试状态：处理器停机调试时进入该状态。

7.4.3 特权访问和用户访问

代码可以是特权执行或非特权执行。非特权执行时对有些资源的访问受到限制或不允许访问。特权执行可以访问所有资源。处理模式始终是特权访问，线程模式可以是特权或非特权访问。

线程模式在复位之后为特权访问，但可通过 MSR 指令清零 CONTROL[0]，将它配置为用户（非特权）访问。用户访问禁止：

部分指令的使用，例如设置 FAULTMASK 和 PRIMASK 的 CPS 指令。

对系统控制空间（SCS）的大部分寄存器的访问。

当线程模式从特权访问变为用户访问后，本身不能回到特权访问。只有处理操作能够改变线程模式的访问特权。处理模式始终是特权访问的。

7.5 指令集

Mnemonic	Operands	Brief description	Flags
ADC, ADCS	{Rd,} Rn, Op2	Add with Carry	N,Z,C,V
ADD, ADDS	{Rd,} Rn, Op2	Add	N,Z,C,V
ADD, ADDW	{Rd,} Rn, #imm12	Add	-
ADR	Rd, label	Address to Register	-
AND, ANDS	{Rd,} Rn, Op2	Logical AND	N,Z,C
ASR, ASRS	Rd, Rm, <Rs|#n>	Arithmetic Shift Right	N,Z,C
B {cond}	label	Branch {conditionally}	-
BFC	Rd, #lsb, #width	Bit Field Clear	-
BFI	Rd, Rn, #lsb, #width	Bit Field Insert	-
BIC, BICS	{Rd,} Rn, Op2	Bit Clear	N,Z,C
BKPT	#imm8	Breakpoint	-
BL	label	Branch with Link	-
BLX	Rm	Branch indirect with Link and Exchange	-
BLXNS	Rm	Branch indirect with Link and Exchange, Non-secure	-
BX	Rm	Branch and Exchange	-
BXNS	Rm	Branch and Exchange, Non secure	-
CBNZ	Rn, label	Compare and Branch on Non Zero	-
CBZ	Rn, label	Compare and Branch on Zero	-
CDP, CDP2	{*cond*} *coproc*, #*op1*, *Rt*, *CRn*, *CRm*{, #*op2*}	Coprocessor Data Processing	-
CLREX	-	Clear Exclusive	-
CLZ	Rd, Rm	Count Leading Zeros	-
CMN	Rn, Op2	Compare Negative	N,Z,C,V
CMP	Rn, Op2	Compare	N,Z,C,V
CPSID	i	Change Processor State, Disable Interrupts	-
CPSIE	i	Change Processor State, Enable Interrupts	-
DMB	{opt}	Data Memory Barrier	-
DSB	{opt}	Data Synchronization Barrier	-
EOR, EORS	{Rd,} Rn, Op2	Exclusive OR	N,Z,C
FLDMDBX ,FLDMIAX	Rn	FLDMX (Decrement Before, Increment After) loads	-
FSTMDBX,FSTMIAX	Rn	FSTMX (Decrement Before, Increment After) stores	-
ISB	{opt}	Instruction Synchronization Barrier	-
IT	-	If Then condition block	-
LDA	Rd, [Rn]	Load-Acquire Word	
LDAB	Rd, [Rn]	Load-Acquire Byte	
LDAEX	Rd, [Rn]	Load-Acquire Exclusive Word	-
LDAEXB	Rd, [Rn]	Load-Acquire Exclusive Byte	-
LDAEXH	Rd, [Rn]	Load-Acquire Exclusive Halfword	-
LDAH	Rd, [Rn]	Load-Acquire Halfword	-
LDM	Rn{!}, reglist	Load Multiple	-
LDMDB, LDMEA	Rn{!}, reglist	Load Multiple Decrement Before	-
LDMIA, LDMFD	Rn{!}, reglist	Load Multiple, Increment After	-
LDR	Rt, [Rn, Rm {, LSL #shift}]	Load Register Word (register offset)	-
LDR	Rt, label	Load Register Word (literal)	-
LDR, LDRT	Rt, [Rn, #offset]	Load Register Word (immediate offset, unprivileged)	-
LDRB	Rt, [Rn, Rm {, LSL #shift}]	Load Register Byte (register offset)	-
LDRB	Rt, label	Load Register Byte (literal)	-
LDRB, LDRBT	Rt, [Rn, #offset]	Load Register Byte (immediate offset, unprivileged)	-
LDRD	Rt, Rt2, [Rn, #offset]	Load Register Dual (immediate offset)	-
LDRD	Rt, Rt2, label	Load Register Dual (PC relative)	-
LDREX	Rt, [Rn, #offset]	Load Register Exclusive	-
LDREXB	Rt, [Rn]	Load Register Exclusive Byte	-
LDREXH	Rt, [Rn]	Load Register Exclusive Halfword	-

LDRH	Rt, [Rn, Rm {, LSL #shift}]	Load Register Halfword (register offset)	-
LDRH	Rt, label	Load Register Halfword (literal)	-
LDRH, LDRHT	Rt, [Rn, #offset]	Load Register Halfword (immediate offset, unprivileged)	-
LDRSB	Rt, [Rn, Rm {, LSL #shift}]	Load Register Signed Byte (register offset)	-
LDRSB	Rt, label	Load Register Signed Byte (PC-relative)	-
LDRSB, LDRSBT	Rt, [Rn, #offset]	Load Register Signed Byte (immediate offset, unprivileged)	-
LDRSH	Rt, [Rn, Rm {, LSL #shift}]	Load Register Signed Halfword (register offset)	-
LDRSH	Rt, label	Load Register Signed Halfword (PC-relative)	-
LDRSH, LDRSHT	Rt, [Rn, #offset]	Load Register Signed Halfword (immediate offset, unprivileged)	-
LSL, LSLS	Rd, Rm, <Rs|#n>	Logical Shift Left	N,Z,C
LSR, LSRS	Rd, Rm, <Rs|#n>	Logical Shift Right	N,Z,C
MCR,MCR2	{*cond*} *coproc*, #*opc1*, *Rt*, *CRn*, *CRm*{, #*opc2*}	Move to Coprocessor from Register	-
MCRR,MCRR2	{*cond*} *coproc*, #*opc1*, *Rt*, *Rt2*, *CRm*	Move to Coprocessor from two Registers	-
MLA	Rd, Rn, Rm, Ra	Multiply Accumulate	-
MLS	Rd, Rn, Rm, Ra	Multiply and Subtract	-
MOV, MOVS	Rd, Op2	Move	N,Z,C
MOV, MOVS	Rd, Rm	Move (register)	N,Z
MOVT	Rd, #imm16	Move Top	-
MOVW	Rd, #imm16	Move 16-bit constant	N,Z,C
MRC,MRC2	{*cond*} *coproc*, #*opc1*, *Rt*, *CRn*, *CRm*{, #*opc2*}	Move to Register from | Coprocessor	-
MRRC,MRRC2	{*cond*} *coproc*, #*opc1*, *Rt*, *Rt2*, *CRm*	Move to two Registers from Coprocessor.	-
MRS	Rd, spec_reg	Move from Special Register to general register	-
MSR	spec_reg, Rn	Move from general register to Special Register	-
MUL, MULS	{Rd,} Rn, Rm	Multiply	N,Z
MVN, MVNS	Rd, Op2	Bitwise NOT	N,Z,C
NOP	-	No Operation	-
ORN, ORNS	{Rd,} Rn, Op2	Logical OR NOT	N,Z,C
ORR, ORRS	{Rd,} Rn, Op2	Logical OR	N,Z,C
PKHTB, PKHBT	{Rd,} Rn, Rm, {, Op2}	Pack Halfword	-
PLD	[Rn {, #offset}]	Preload Data	-
POP	reglist	Pop registers from stack	-
PUSH	reglist	Push registers onto stack	-
QADD	{Rd,} Rn, Rm	Saturating Add	Q
QADD16	{Rd,} Rn, Rm	Saturating Add 16	-
QADD8	{Rd,} Rn, Rm	Saturating Add 8	-
QASX	{Rd,} Rn, Rm	Saturating Add and Subtract with Exchange	-
QDADD	{Rd,} Rn, Rm	Saturating Double and Add	Q
QDSUB	{Rd,} Rn, Rm	Saturating Double and Subtract	Q
QSAX	{Rd,} Rn, Rm	Saturating Subtract and Add with Exchange	-
QSUB	{Rd,} Rn, Rm	Saturating Subtract	Q
QSUB16	{Rd,} Rn, Rm	Saturating Subtract 16	-
QSUB8	{Rd,} Rn, Rm	Saturating Subtract 8	-
RBIT	Rd, Rn	Reverse Bits	-
REV	Rd, Rn	Reverse byte order in a word	-
REV16	Rd, Rn	Reverse byte order in each halfword	-
REVSH	Rd, Rn	Reverse byte order in bottom halfword and sign extend	-
ROR, RORS	Rd, Rm, <Rs|#n>	Rotate Right	N,Z,C
RRX, RRXS	Rd, Rm	Rotate Right with Extend	N,Z,C
RSB, RSBS	{Rd,} Rn, Op2	Reverse Subtract	N,Z,C,V

SADD16	{Rd,} Rn, Rm	Signed Add 16	GE
SADD8	{Rd,} Rn, Rm	Signed Add 8	GE
SASX	{Rd,} Rn, Rm	Signed Add and Subtract with Exchange	GE
SBC, SBCS	{Rd,} Rn, Op2	Subtract with Carry	N,Z,C,V
SBFX	Rd, Rn, #lsb, #width	Signed Bit Field Extract	-
SDIV	{Rd,} Rn, Rm	Signed Divide	-
SEL	{Rd,} Rn, Rm	Select bytes	GE
SEV	-	Send Event	-
SG	-	Secure Gateway	-
SHADD16	{Rd,} Rn, Rm	Signed Halving Add 16	-
SHADD8	{Rd,} Rn, Rm	Signed Halving Add 8	-
SHASX	{Rd,} Rn, Rm	Signed Halving Add and Subtract with Exchange	-
SHSAX	{Rd,} Rn, Rm	Signed Halving Subtract and Add with Exchange	-
SHSUB16	{Rd,} Rn, Rm	Signed Halving Subtract 16	-
SHSUB8	{Rd,} Rn, Rm	Signed Halving Subtract 8	-
SMLABB, SMLABT, SMLATB, SMLATT	Rd, Rn, Rm, Ra	Signed Multiply Accumulate halfwords	Q
SMLAD, SMLADX	Rd, Rn, Rm, Ra	Signed Multiply Accumulate Dual	Q
SMLAL	RdLo, RdHi, Rn, Rm	Signed Multiply Accumulate Long (32 × 32 + 64), 64-bit result	-
SMLALBB, MLALBT, SMLALTB, SMLALTT	RdLo, RdHi, Rn, Rm	Signed Multiply Accumulate Long, halfwords	-
SMLALD, SMLALDX	RdLo, RdHi, Rn, Rm	Signed Multiply Accumulate Long Dual	-
SMLAWB, SMLAWT	Rd, Rn, Rm, Ra	Signed Multiply Accumulate, word by halfword	Q
SMLSD, SMLSDX	Rd, Rn, Rm, Ra	Signed Multiply Subtract Dual	Q
SMLSLD, SMLSLDX	RdLo, RdHi, Rn, Rm	Signed Multiply Subtract Long Dual	-
SMMLA, SMMLAR	Rd, Rn, Rm, Ra	Signed Most Significant Word Multiply Accumulate	-
SMMLS, SMMLSR	Rd, Rn, Rm, Ra	Signed Most Significant Word Multiply Subtract	-
SMMUL, SMMULR	Rd, Rn, Rm	Signed Most Significant Word Multiply	-
SMUAD, SMUADX	{Rd,} Rn, Rm	Signed Dual Multiply Add	Q.
SMULBB, SMULBT, SMULTB, SMULTT	{Rd,} Rn, Rm	Signed Multiply (halfwords)	-
SMULL	RdLo, RdHi, Rn, Rm	Signed Multiply Long (32 × 32), 64-bit result	-
SMULWB, SMULWT	{Rd,} Rn, Rm	Signed Multiply word by halfword	-
SMUSD, SMUSDX	{Rd,} Rn, Rm	Signed Dual Multiply Subtract	-
SSAT	Rd, #n, Rm {,shift #s}	Signed Saturate	Q
SSAT16	Rd, #n, Rm	Signed Saturate 16	Q
SSAX	{Rd,} Rn, Rm	Signed Subtract and Add with Exchange	GE
SSUB16	{Rd,} Rn, Rm	Signed Subtract 16	GE
SSUB8	{Rd,} Rn, Rm	Signed Subtract 8	GE
STL	Rt, [Rn]	Store-Release Word	-
STLB	Rt, [Rn]	Store-Release Byte	-
STLEX	Rt, Rt [Rn]	Store-Release Exclusive Word	-
STLEXB	Rt, Rt [Rn]	Store-Release Exclusive Byte	-
STLEXH	Rt, Rt [Rn]	Store-Release Exclusive Halfword	-
STLH	Rt, [Rn]	Store-Release Halfword	-
STM	Rn{!}, reglist	Store Multiple	-
STMDB, STMEA	Rn{!}, reglist	Store Multiple Decrement Before	-
STMIA, STMFD	Rn{!}, reglist	Store Multiple Increment After	-
STR	Rt, [Rn, Rm {, LSL #shift}]	Store Register Word (register offset)	-
STR, STRT	Rt, [Rn, #offset]	Store Register Word (immediate offset, unprivileged)	-
STRB	Rt, [Rn, Rm {, LSL #shift}]	Store Register Byte (register offset)	-
STRB, STRBT	Rt, [Rn, #offset]	Store Register Byte (immediate offset, unprivileged)	-
STRD	Rt, Rt2, [Rn, #offset]	Store Register Dual two words	-
STREX	Rd, Rt, [Rn, #offset]	Store Register Exclusive	-
STREXB	Rd, Rt, [Rn]	Store Register Exclusive Byte	-
STREXH	Rd, Rt, [Rn]	Store Register Exclusive Halfword	-

STRH	Rt, [Rn, Rm {, LSL #shift}]	Store Register Halfword (register offset)	-
STRH, STRHT	Rt, [Rn, #offset]	Store Register Halfword (immediate offset, unprivileged)	-
SUB, SUBS	{Rd,} Rn, Op2	Subtract	N,Z,C,V
SUB, SUBW	{Rd,} Rn, #imm12	Subtract	-
SVC	#imm	Supervisor Call	-
SXTAB	{Rd,} Rn, Rm {,ROR #n}	Sign extend 8 bits to 32 and Add	-
SXTAB16	{Rd,} Rn, Rm {,ROR #n}	Sign extend two 8-bit values to 16 and Add	-
SXTAH	{Rd,} Rn, Rm {,ROR #n}	Sign extend 16 bits to 32 and Add	-
SXTB	Rd, Rm {,ROR #n}	Sign extend 8 bits to 32	-
SXTB16	{Rd,} Rm {,ROR #n}	Sign extend 8 bits to 16	-
SXTH	{Rd,} Rm {,ROR #n}	Sign extend a Halfword to 32	-
TBB	[Rn, Rm]	Table Branch Byte	-
TBH	[Rn, Rm, LSL #1]	Table Branch Halfword	-
TEQ	Rn, Op2	Test Equivalence	N,Z,C
TST	Rn, Op2	Test	N,Z,C
TT	Rd, [Rn]	Test Target	-
TTA	Rd, [Rn]	Test Target Alternate Domain	-
TTAT	Rd, [Rn]	Test Target Alternate Domain Unprivileged	-
TTT	Rd, [Rn]	Test Target Unprivileged	-
UADD16	{Rd,} Rn, Rm	Unsigned Add 16	GE
UADD8	{Rd,} Rn, Rm	Unsigned Add 8	GE
UASX	{Rd,} Rn, Rm	Unsigned Add and Subtract with Exchange	GE
UBFX	Rd, Rn, #lsb, #width	Unsigned Bit Field Extract	-
UDF	{c}{q} {#}imm	Permanently Undefined.	-
UDIV	{Rd,} Rn, Rm	Unsigned Divide	-
UHADD16	{Rd,} Rn, Rm	Unsigned Halving Add 16	-
UHADD8	{Rd,} Rn, Rm	Unsigned Halving Add 8	-
UHASX	{Rd,} Rn, Rm	Unsigned Halving Add and Subtract with Exchange	-
UHSAX	{Rd,} Rn, Rm	Unsigned Halving Subtract and Add with Exchange	-
UHSUB16	{Rd,} Rn, Rm	Unsigned Halving Subtract 16	-
UHSUB8	{Rd,} Rn, Rm	Unsigned Halving Subtract 8	-
UMAAL	RdLo, RdHi, Rn, Rm	Unsigned Multiply Accumulate Accumulate Long (32 × 32 + 32 + 32), 64-bit result	-
UMLAL	RdLo, RdHi, Rn, Rm	Unsigned Multiply Accumulate Long (32 × 32 + 64), 64-bit result	-
UMULL	RdLo, RdHi, Rn, Rm	Unsigned Multiply Long (32 × 32), 64-bit result	-
UQADD16	{Rd,} Rn, Rm	Unsigned Saturating Add 16	-
UQADD8	{Rd,} Rn, Rm	Unsigned Saturating Add 8	-
UQASX	{Rd,} Rn, Rm	Unsigned Saturating Add and Subtract with Exchange	-
UQSAX	{Rd,} Rn, Rm	Unsigned Saturating Subtract and Add with Exchange	-
UQSUB16	{Rd,} Rn, Rm	Unsigned Saturating Subtract 16	-
UQSUB8	{Rd,} Rn, Rm	Unsigned Saturating Subtract 8	-
USAD8	{Rd,} Rn, Rm	Unsigned Sum of Absolute Differences	-
USADA8	Rd, Rn, Rm, Ra	Unsigned Sum of Absolute Differences and Accumulate	-
USAT	Rd, #n, Rm{,shift #s}, Ra	Unsigned Saturate	Q
USAT16	Rd, #n, Rm	Unsigned Saturate 16	Q
USAX	{Rd,} Rn, Rm	Unsigned Subtract and Add with Exchange	GE
USUB16	{Rd,} Rn, Rm	Unsigned Subtract 16	GE
USUB8	{Rd,} Rn, Rm	Unsigned Subtract 8	GE
UXTAB	{Rd,} Rn, Rm {,ROR #n}	Rotate, unsigned extend 8 bits to 32 and Add	-
UXTAB16	{Rd,} Rn, Rm {,ROR #n}	Rotate, unsigned extend two 8-bit values to 16 and Add	-
UXTAH	{Rd,} Rn, Rm {,ROR #n}	Rotate, unsigned extend and Add Halfword	-

UXTB	Rd, Rm {,ROR #n}	Unsigned zero-extend Byte	-
UXTB16	{Rd,} Rm {,ROR #n}	Unsigned zero-extend Byte 16	-
UXTH	Rd, Rm {,ROR #n}	Unsigned zero-extend Halfword	-
VABS	.F32 Sd, Sm	Floating-point Absolute	-
VADD	.F32 {Sd,} Sn, Sm	Floating-point Add	-
VCMP	.F32 Sd, <<Sm| #0.0>	Compare two floating-point registers, or one floating point register and zero	N,Z,C,V
VCMPE	.F32 Sd, <<Sm| #0.0>	Compare two floating-point registers, or one floating point register and zero with Invalid Operation check	N,Z,C,V
VCVT	.F32.Tm <Sd>, Sm	Convert from floating-point to integer	-
VCVT	.Td.F32 Sd, Sd, #fbits	Convert from floating-point to fixed point	-
VCVTA	.Tm.F32 <Sd>, Sm	Convert from floating-point to integer with directed rounding to nearest with Ties Away	-
VCVTB VCVTT	.F32.F16 Sd, Sm	Convert half-precision value to single-precision or double-precision	-
VCVTB VCVTT	.F16.F32 Sd, Sm	Convert single-precision or double-precision register to half-precision	-
VCVTM	.Tm.F32 <Sd>, Sm	Convert from floating-point to integer with directed rounding towards Minus infinity	-
VCVTN	.Tm.F32 <Sd>, Sm	Convert from floating-point to integer with directed rounding to nearest with Ties to even	-
VCVTP	.Tm.F32 <Sd>, Sm	Convert from floating-point to integer with directed rounding towards Plus infinity	-
VCVTR	.Tm.F32 <Sd>, Sm	Convert between floating point and integer with rounding.	-
VDIV	.F32 {Sd,} Sn, Sm	Floating-point Divide	-
VFMA	.F32 {Sd,} Sn, Sm	Floating-point Fused Multiply Accumulate	-
VFMS	.F32 {Sd,} Sn, Sm	Floating-point Fused Multiply Subtract	-
VFNMA	.F32 {Sd,} Sn, Sm	Floating-point Fused Negate Multiply Accumulate	-
VFNMS	.F32 {Sd,} Sn, Sm	Floating-point Fused Negate Multiply Subtract	-
VLDM	{mode}{.size} Rn{!}, list	Floating-point Load Multiple extension registers	-
VLDR	.F32 Sd, [<Rn> {, #offset}]	Floating-point Load an extension register from memory (immediate)	-
VLDR	.F32 Sd, <label>	Load an extension register from memory	-
VLDR	.F32 Sd, [PC,#-0]	Load an extension register from memory	-
VLLDM	<c> Rn	Floating-point Lazy Load multiple	-
VLSTM	<c> Rn	Floating-point Lazy Store multiple	-
VMAXNM	.F32 Sd, Sn, Sm	Maximum of two floating point numbers with IEEE754-2008 NaN handling	-
VMINNM	.F32 Sd, Sn, Sm	Minimum of two floating point numbers with IEEE754-2008 NaN handling	-
VMLA	.F32 Sd, Sn, Sm	Floating-point Multiply Accumulate	-
VMLS	.F32 Sd, Sn, Sm	Floating-point Multiply Subtract	-
VMOV	<Sn|Rt>, <Rt|Sn>	Copy core register to single precision	-
VMOV	<Sm|Rt>, <Sm1| Rt2>, <Rt|Sm>, <Rt2|Sm1>	Copy two core registers to two single-precision	-
VMOV	{.size} Dd[x], Rt	Copy core register to scalar	-
VMOV	{.dt} Rt, Dn[x]	Copy scalar to core register	-
VMOV	.F32 Sd, #immm	Floating-point Move immediate	-
VMOV	.F32 Sd, Sd, Sm	Copies the contents of one register to another	-
VMOV	<Dm|Rt>, <Rt| Rt2>, <Rt2|Dm>	Floating-point Move transfers two words between two core registers and a doubleword register	-
VMRS	Rt, FPSCR	Move to core register from floating-point Special Register	N,Z,C,V
VMSR	FPSCR, Rt	Move to floating-point Special Register from core register	-
VMUL	.F32 {Sd,} Sn, Sm	Floating-point Multiply	-
VNEG	.F32 Sd, Sm	Floating-point Negate	-

VNMLA	.F32 Sd, Sn, Sm	Floating-point Multiply Accumulate and Negate	-
VNMLS	.F32 Sd, Sn, Sm	Floating-point Multiply, Subtract and Negate	-
VNMUL	.F32 {Sd,} Sn, Sm	Floating-point Multiply and Negate	-
VPOP	{.size} list	Load multiple consecutive floating-point registers from the stack	-
VPUSH	{.size} list	Store multiple consecutive floating-point registers to the stack	-
VRINTA	.F32 Sd, Sm	Float to integer in floating point format conversion with directed rounding to Nearest with Ties Away	-
VRINTM	.F32 Sd, Sm	Float to integer in floating point format conversion with directed rounding to Minus infinity	-
VRINTN	.F32 Sd, Sm	Float to integer in floating point format conversion with directed rounding to Nearest with Ties to even	-
VRINTP	.F32 Sd, Sm	Float to integer in floating point format conversion with directed rounding to Plus infinity	-
VRINTR	.F32 Sd, Sm	Float to integer in floating point format conversion with rounding towards value specified in FPSCR	-
VRINTX	.F32 Sd, Sm	Float to integer in floating point format conversion with rounding specified in FPSCR	-
VRINTZ	.F32 Sd, Sm	Float to integer in floating point format conversion with rounding towards Zero	-
VSEL	.F32 Sd, Sn, Sm	Select register, alternative to a pair of conditional VMOV	-
VSQRT	.F32 Sd, Sm	Calculates floating-point Square Root	-
VSTM	{mode}{.size} Rn{!}, list	Floating-point Store Multiple	-
VSTR	.F32 Sd, [Rn{, #offset}]	Floating-point Store Register stores an extension register to memory	-
VSUB	F32 {Sd,} Sn, Sm	Floating-point Subtract	-
WFE	-	Wait For Event	-
WFI	-	Wait For Interrupt	-
YIELD	-	Suspend task	-

8. 存储器和总线架构

ARM® ”星辰”处理器的总线接口基于AMBA5 AHB协议，它们的规则参考“AMBA5 AHB 规格”。

8.1 系统架构

SH33F2801主系统由多层AHB总线矩阵互联和驱动，如下图所示：

图8-1 SH33F2801 系统架构框图

AHB外设总线

AHB外设总线挂载在总线矩阵上，一些要求高速并和系统紧密耦合的外设挂载在该总线上。AHB/APB桥也挂载在此总线上。

AHB/APB桥

AHB-APB桥在AHB和APB总线间提供同步连接。APB是降速外设总线，最高速度为42MHz。

在使用一个外设之前，必须设置时钟寄存器AHBENR/APBENR的相应控制位来打开该外设的时钟，具体请参考“复位和时钟控制（RCC）”章节。

8.2 存储器组织

SH33F2801遵循ARM® ”星辰”处理器对存储器组织的基础规定，存储器基本组织架构如下：

表8-1 SH33F2801 基本存储器组织表

地址范围	用途	地址范围细分	描述
0x0000 0000 – 0x0FFF FFFF	片上非易失性代码存储区（Flash）	0x0000 0000 – 0x0001 FFFF	片上128K flash
	片上非易失性系统存储区（System Block）	0x0FFF 0000 – 0x0FFF 0FFF	片上EEProm-Like存储区（共4K）
		保留区域	保留区域
		0x0FFF F800 – 0x0FFF FBFF	片上OTP区（共 1K）
0x2000 0000 – 0x3FFF FFFF	片上易失性数据存储区（SRAM）	0x2000 0000 – 0x2000 3FFF	片上16K SRAM
		0x2000 2000 – 0x3FFF FFFF	保留区域
0x4000 0000 – 0x5FFF FFFF	片上外设区	0x4000 0000 – 0x4001 FFFF	片上APB外设区，每个外设分配1K字节地址空间
		0x4002 0000 – 0x4003 FFFF	保留区域
		0x4004 0000 – 0x400F FFFF	片上AHB外设区，每个外设分配1K字节地址空间
0x6000 0000 – 0x9FFF FFFF	保留区域	-	可在此区域扩展外部RAM设备
0xA000 0000 – 0xDFFF FFFF	保留区域	-	可在此区域扩展外部外设设备
0xE000 0000 – 0xE00F FFFF	ARM “星辰”处理器私有外设区	0xE000 1000 – 0xE000 1FFF	数据观察点与跟踪（DWT）
		0xE000 2000 – 0xE000 2FFF	闪存地址重载与断点单元（FPB）
		0xE000 3000 – 0xE000 DFFF	保留区域
		0xE000 E000 – 0xE000 EFFF	嵌套向量中断控制器（NVIC）
		0xE000 F000 – 0xE004 FFFF	保留区域

8.2.1 寄存器映像

表8-2列出了所有SH33F2801中内置外设（不包含ARM “星辰”处理器设备）的基地址。具体各个外设的寄存器映像请参考该外设“寄存器”章节中的寄存器地址映像表。

外设寄存器地址：

表8-2 SH33F2801 内置外设的基地址

地址	外设	总线
0x4000 0000 ~ 0x4000 03FF	保留	APB总线: 0x4000 0000 - 0x4003 FFFF
0x4000 0400 ~ 0x4000 07FF	PCA0	
0x4000 0800 ~ 0x4000 0BFF	PCA1	
0x4000 0C00 ~ 0x4000 0FFF	TIM7	
0x4000 1000 ~ 0x4000 13FF	TIM8	
0x4000 1400 ~ 0x4000 17FF	QEI	
0x4000 1800 ~ 0x4000 1BFF	UART0	
0x4000 1C00 ~ 0x4000 1FFF	UART1	
0x4000 2000 ~ 0x4000 23FF	UART2	
0x4000 2400 ~ 0x4000 27FF	SPI	
0x4000 2800 ~ 0x4000 2BFF	TWI	
0x4000 2C00 ~ 0x4000 2FFF	CAN	
0x4000 3000 ~ 0x4000 33FF	IWDT	
0x4000 3400 ~ 0x4000 37FF	WWDT	
0x4000 3800 ~ 0x4000 3BFF	AMOC	
0x4000 3C00 ~ 0x4000 3FFF	ADC	
0x4000 4000 ~ 0x4000 43FF	EXIT	
0x4000 4400 ~ 0x4000 47FF	FIFO0	
0x4000 4800 ~ 0x4000 4BFF	FIFO1	
0x4000 4C00 ~ 0x4000 4FFF	FIFO2	
0x4000 5000 ~ 0x4003 FFFF	保留	
0x4004 0000 ~ 0x4004 01FF	GPIO	AHB总线: 0x4004 0000 - 0x4005 FFFF
0x4004 0400 ~ 0x4004 07FF	MCM	
0x4004 0800 ~ 0x4004 0BFF	SYSCFG	
0x4004 0C00 ~ 0x4004 0FFF	RCC	
0x4004 1000 ~ 0x4004 13FF	FLASH	
0x4004 1400 ~ 0x4004 17FF	MACP	
0x4004 1800 ~ 0x4004 1BFF	CRC	
0x4004 1C00 ~ 0x4005 FFFF	保留	

8.2.2 数据存储器

SH33F2801内置16K RAM（SRAM），可以存放数据，也可以运行代码。它可以以字节、半字(16位)或全字(32位)访问。SRAM的起始地址是0x2000 0000。SRAM支持March检测算法。

8.2.3 程序存储器

SH33F2801内置有Flash存储器，分成主存储块和类EEPROM区以及OTP区三部分：

主存储块为128K字节，划分为128个1K字节的扇区。

类EEPROM区为4K字节，用户可通过烧写工具以及SSP（Self-Sector Programming）方式进行操作。

OTP区为1K字节，只能一次写入，写入后不可更改。

8.3 Flash 程序存储器

8.3.1 特性

- Flash 主存储器包括 128 X 1KB 扇区，总共 128KB
- Flash 类 EEPROM 区包括 4 X1KB 扇区，总共 4KB
- Flash OTP 区包括 1 X 1KB 扇区，总共 1KB
- Flash 写操作支持 32 位，16 位写入，且每次写入一个非全 0 数据后，再下次擦除之前，不能再次写入
- 在工作电压范围内都能进行编程和擦除操作
- 支持整体/扇区擦除和编程
- 支持 FLASH 划分区块功能，方便在线升级程序及备份
- Flash 读保护功能，防止非法访问
- Flash 写保护功能，阻止意外操作
- 读保护、写保护、客户安全码多层级保护
- 防止侵入型攻击 Flash

8.3.2 简介

SH33F2801 内置 128K 的可编程 Flash 主程序存储区（Main Program Memory Block），每个扇区 1024 字节，128 个扇区都可单独进行擦除。

4KB 的内置类 EEPROM 存储区，用于存放用户数据，每个扇区 1024 字节。

1KB 的 OTP 区，用于存放出厂初始化数据，**OTP 区数据一旦被写入就无法被擦除**。

Flash 存储器结构见下图。

图8-2 Flash 存储器结构

表8-3　　Flash 存储器结构

Block	Name	Base address	Size（bytes）
Main memory	**Sector 0**	0x0000 0000 - 0x0000 03FF	1024
	Sector 1	0x0000 0400 - 0x0000 07FF	1024
	Sector 2	0x0000 0800 - 0x0000 0BFF	1024
	Sector 3	0x0000 0C00 - 0x0000 0FFF	1024
	Sector 4	0x0000 1000 - 0x0000 13FF	1024
	…	…	…
	…	…	…
	Sector 126	0x0001 F800 - 0x0001 FBFF	1024
	Sector 127	0x0001 FC00 - 0x0001 FFFF	1024
Information Block	**EEPROM Block**	0x0FFF 0000 - 0x0FFF 0FFF	4096
	Protect Block	0x0FFF E000 - 0x0FFF E3FF	1024
	Customer Block	0x0FFF E800 - 0x0FFF EBFF	1024
	OTP Block	0x0FFF F800 - 0x0FFF FBFF	1024
Flash memory Operation registers	**FLASH_ACR**	0x4004 1000 - 0x4004 1003	4
	FLASH_MKYR	0x4004 1004 - 0x4004 1007	4
	FLASH_E2KYR	0x4004 1008 - 0x4004 100B	4
	FLASH_SR	0x4004 100C - 0x4004 100F	4
	FLASH_CR	0x4004 1010 - 0x4004 1013	4
	Reversed	0x4004 1014 - 0x4004 102F	4
	FLASH_CNTR	0x4004 1030 - 0x4004 1033	4
	FLASH_UPCNTR	0x4004 1034 - 0x4004 1037	4
	FLASH_CNTCR	0x4004 1038 - 0x4004 103B	4
	Reversed	0x4004 103C - 0x4004 103F	4
	FLASH_IKYR	0x4004 1040 - 0x4004 1043	4
	Reversed	0x4004 1044 - 0x4004 10FF	188

8.3.3　Flash 存储器功能描述

8.3.3.1　Flash 擦除与编程

系统复位后，Flash 存储器操作寄存器处于锁定状态，所以对 Flash 进行擦除编程操作之前需要对 Flash 存储器操作寄存器进行解锁。不同 Flash 存储区具有不同的解锁寄存器，对不同区域操作需要解锁相应的寄存器。对解锁 Flash 操作也分为两种情况，一种是解锁后可对 Flash 进行单次操作，另一种是解锁后，可对 Flash 进行多次操作。如果 Flash 存储器操作寄存器处于解锁状态，可通过控制 FLASH_CR 寄存器，对 Flash 各存储区进行擦除、编程操作。但为防止 Flash 被误操作，对 Flash 存储器控制寄存器进行访问时，必须保证 Flash 操作定时器的计数值在一个有效的范围内，否则对 Flash 各存储区的操作将被禁止。

对 Flash 存储器，每次可以写入 16 位或 32 位数据。主存储区进行擦除操作后，每 16 位或 32 位仅可以被编程一次，即被编程的 16 位或 32 位数据的各个位如果不是 0，就不可以再次被编程，否则相应错误标志置 1。

主存储区的整片擦除与按扇区擦除，主存储区的编程，类 EEPROM 存储区的按扇区擦除与编程，客户信息区的按扇区擦除与编程，OTP 区的编程，都可以通过运行在 RAM 中的程序或者在 Debug 模式下直接访问 Flash 操作寄存器来实现。

主存储区的按扇区擦除及主存储区的编程、类 EEPROM 存储区的按扇区擦除与编程，客户信息区的按扇区擦除与编程，OTP 区的编程，可以通过主程序区中的程序访问 Flash 控制寄存器来实现。

Flash 中的主程序区可以通过设置读保护防止被非法的读出；同样可以对 Flash 存储区的各扇区设置写保护，防止在程序跑飞的情况下不被意外地改变。

8.3.3.2 Flash 控制寄存器解锁

系统复位后，Flash 存储器操作寄存器进入锁定状态，所以对 Flash 进行擦除编程操作之前需要对 Flash 存储器操作寄存器进行解锁。不同 Flash 存储区具有不同的解锁寄存器，对不同区域操作需要解锁相应的寄存器。对 Flash 解锁需要对相应解锁寄存器写入两个解锁值，第一个键值写入错误或第二个键值写入错误，都不能对 Flash 相应区域进行解锁，并会产生一个总线错误；任意一个区域处于解锁状态，再对解锁寄存器进行解锁，也会产生一个总线错误。对解锁 Flash 操作也分为两种情况，一种是解锁后可对 Flash 进行单次操作，另一种是解锁后，可对 Flash 进行多次操作。如果 Flash 存储器操作寄存器处于解锁状态，可通过控制 FLASH_CR，对 Flash 各存储区进行擦除、编程操作。

复位后，位 MNLCK@FLASH_CR 置为 1。通过先向 FLASH_MKYR 寄存器写入 Flash 解锁值——0x8ACE 0246，再向 FLASH_MKYR 寄存器写入单次操作解锁值 0xC3C3 C3C3 或多次操作解锁值 0xB4B4 B4B4 后，才可以使得 FLASH_CR 解锁。两次正确的写解锁值操作完成后，位 MNLCK@FLASH_CR 被硬件清 0。软件可以将控制位 MNLCK@FLASH_CR 置 1 来锁定主程序区。FLASH_MKYR 寄存器锁定的是主程序区，解锁后，可以通过运行在 RAM 中的程序，主程序区中的程序，Boot 区的程序或者使用调试工具直接访问 Flash 操作寄存器的方式对主程序区进行操作。

特殊信息区的类 EEPROM 区可以被 FLASH_E2KYR 寄存器锁定。解锁过程也是两次写操作，通过先向 FLASH_E2KYR 寄存器写入 Flash 解锁值——0x9BDF 1357，再向 FLASH_E2KYR 寄存器写入单次操作解锁值 0xC3C3 C3C3 或多次操作解锁值 0xB4B4 B4B4 后，才可以使得该区域被解锁，同时控制位 E2LCK@FLASH_CR 被硬件清 0。软件可以将控制位 E2LCK@FLASH_CR 置 1 来锁定特殊信息区的类 EEPROM 区。FLASH_E2KYR 寄存器锁定的是特殊信息区的类 EEPROM 区，该区域解锁后，可以通过运行在 RAM 中的程序，主程序区中的程序，Boot 区的程序或者使用调试工具直接访问 Flash 操作寄存器的方式对其进行操作。

特殊信息区的 OTP 区可以被 FLASH_IKYR 寄存器锁定。解锁过程也是两次写操作，通过先向 FLASH_IKYR 寄存器写入 Flash 解锁值——0xABCD 5678，再向 FLASH_IKYR 寄存器写入单次操作解锁值 0xC3C3 C3C3 或多次操作解锁值 0xB4B4 B4B4 后，才可以使得该区域被解锁，同时控制位 INFLCK@FLASH_CR 被硬件清 0。软件可以将控制位 INFLCK@FLASH_CR 置 1 来锁定特殊信息区的 OTP 区。FLASH_IKYR 寄存器锁定的是特殊信息区的 OTP 区，其中 OTP 区可以通过运行在 RAM 中的程序，主程序区中的程序来实现操作。

对于 Flash 单次解锁操作，当 MNLCK@FLASH_CR 或 E2WRE/INFLCK@FLASH_CR 被清 0 后，只允许对 Flash 进行一次编程或擦除操作，操作完成后 MNLCK@FLASH_CR 或 E2WRE/INFLCK@FLASH_CR 被硬件自动置 1，再次锁定 Flash 相关区块。

对于 Flash 多次解锁操作，MNLCK@FLASH_CR 或 E2WRE/ INFLCK@FLASH_CR 被清 0 后，允许对 Flash 进行多次编程或擦除操作，但操作完成后，必须将 MNLCK@FLASH_CR 或 E2WRE/INFLCK@FLASH_CR 位软件置 1，才能再次锁定 Flash 相关区块。

表8-4 各解锁寄存器解锁值及说明

寄存器	功能	Flash初始解锁值	Flash单次操作解锁值	Flash多次操作解锁值/锁定值	说明
FLASH_MKYR	解锁主程序区	0x8ACE 0246	0xC3C3 C3C3	0xB4B4 B4B4	通过运行在RAM中的程序或主程序区中的程序直接访问
FLASH_E2KYR	解锁特殊信息区的类EEPROM区	0x9BDF 1357	0xC3C3 C3C3	0xB4B4 B4B4	通过运行在RAM中的程序或主程序区中的程序直接访问
FLASH_IKYR	解锁特殊信息区的OTP区	0xABCD 5678	0xC3C3 C3C3	0xB4B4 B4B4	通过运行在RAM中的程序或主程序区中的程序直接访问

8.3.3.3 Flash 操作定时器功能

Flash 控制模块具有一个操作定时器，Flash 操作定时器的计数值是以系统时钟为时钟源，Flash 操作定时器的计数值必须小于 Flash 操作定时器上限值，并且大于零，才可以对 Flash 存储区进行操作，或者向目的地址写入一个 32bit/16bit 数据。如果启动一次 Flash 操作，就是说位 STRT@FLASH_CR 置 1 或者启动对 Flash 的写操作时，Flash 操作定时器计数值不在有效的时间窗口内，Flash 操作不被执行，且位 PGWERR@FLASH_SR 置 1。Flash 操作定时器的具体用法可参考下图。

图8-3 Flash 操作定时器使用流程图

8.3.3.4 主程序区整体擦除

Flash 操作提供了整片擦除功能，可以擦除主存储块区的内容。具体步骤如下：
（1）FLASH_MKYR 依次写入 Flash 解锁值和单次操作解锁值，确保 Flash 主程序区不处于锁定状态；
（2）检查位 BSY@FLASH_SR 来判定 Flash 存储器是否处于运行状态；
（3）当位 BSY@FLASH_SR 为 0 时，写 Flash 主程序区整体擦除命令字到 FLASH_CR 寄存器；
（4）通过将位 STRT@FLASH_CR 置 1 来启动整片擦除操作；
（5）检查位 BSY@FLASH_SR 来判断擦除指令是否执行完毕；
（6）当位 BSY@FLASH_SR 为 0 时，操作完成。
位 EOP@FLASH_SR 预示操作的结束，该位置 1，表示操作执行完毕。所有 Flash 数据擦除后被复位为 0x0000 0000。
流程图如下：

图8-4 主程序区整体擦除

8.3.3.5 主程序区扇区擦除

Flash 存储器的每扇区都可以被独立擦除，不影响其它扇区内容。Flash 操作擦除扇区步骤如下：

（a）单独擦除某一扇区

（1）FLASH_MKYR 依次写入 Flash 解锁值和单次操作解锁值，确保 Flash 主程序区不处于锁定状态；

（2）检查位 BSY@FLASH_SR 来判定 Flash 存储器是否处于运行状态；

（3）当位 BSY@FLASH_SR 为 0 时，写主程序区扇区擦除命令字和待擦除扇区号码到 FLASH_CR 寄存器；

（4）通过将位 STRT@FLASH_CR 置 1 来启动扇区擦除操作；

（5）检查位 BSY@FLASH_SR 来判断擦除指令是否执行完毕；

（6）当位 BSY@FLASH_SR 为 0 时，操作完成。

流程图如下：

图8-5 主程序区按扇区擦除（单一扇区）

（b）擦除多个扇区
（1）FLASH_MKYR 依次写入 Flash 解锁值和多次操作解锁值，确保 Flash 主程序区不处于锁定状态；
（2）检查位 BSY@FLASH_SR 来判定 Flash 存储器是否处于运行状态；
（3）当位 BSY@FLASH_SR 为 0 时，写主程序区扇区擦除命令字和扇区代码到 FLASH_CR 寄存器；
（4）通过将位 STRT@FLASH_CR 置 1 来启动扇区擦除操作；
（5）检查位 BSY@FLASH_SR 来判断擦除指令是否执行完毕；
（6）重复（3）-（5），擦除其它扇区；
（7）位 MNLCK@FLASH_CR 置 1，操作完成。
操作流程图如下图：

图8-6　　主程序区按扇区擦除（多个扇区）

当目标擦除扇区被用来取指令或访问数据时，相应的擦除操作无效，但 Flash 操作将不提供任何通知，因此确保目标擦除扇区地址的正确性是很有必要的。位 EOP@FLASH_SR 预示操作的结束。

8.3.3.6 主程序区编程

Flash 操作提供了一个 32 位字/16 位半字编程功能，用来修改 Flash 主存储块内容。对 Flash 主程序区编程每次可以写入 32 位/16 位，地址必需按 32 位/16 位对齐，若未对齐，相应操作无效，相关错误标志位置 1。当 FLASH_CR 寄存器操作命令字设置为主程序区编程时，在一个 Flash 地址写入一个字或半字将启动一次编程。在编程过程中(位 BSY@FLASH_SR 位为'1')，任何读写 Flash 的操作都会使 CPU 暂停，直到此次 Flash 编程结束。

下面的步骤显示了字编程操作寄存器过程。

（a）单字或半字编程

（1）FLASH_MKYR 依次写入 Flash 解锁值和单次操作解锁值，确保 Flash 主程序区不处于锁定状态；

（2）检查位 BSY@FLASH_SR 来判定 Flash 存储器是否处于运行状态；

（3）当位 BSY@FLASH_SR 为 0 时，写 Flash 主程序区编程操作命令字及烧写操作位宽到 FLASH_CR 寄存器；

（4）将位 STRT@FLASH_CR 置 1，向目的地址写入一个 32 位字/16 位半字；

（5）检查位 BSY@FLASH_SR 来判断编程指令是否执行完毕；

（6）当位 BSY@FLASH_SR 为 0 时，操作完成。

操作流程图如下图：

图8-7 主程序区编程（单字或半字）

（b）连续写入多字编程
（1）FLASH_MKYR 依次写入 Flash 解锁值和多次操作解锁值，确保 Flash 主程序区不处于锁定状态；
（2）检查位 BSY@FLASH_SR 来判定 Flash 存储器是否处于运行状态；
（3）写 Flash 主程序区编程操作命令字及烧写操作位宽到 FLASH_CR 寄存器；
（4）将位 STRT@FLASH_CR 置 1，向目的地址写入一个 32 位字/16 位半字；
（5）检查位 BSY@FLASH_SR 来判断编程指令是否执行完毕；
（6）重复（4）-（5），直到多地址都编程结束；
（7）MNLCK@FLASH_CR 位置 1，操作完成。

图8-8　　主程序区编程（连续写入多字编程）

对 Flash 存储器，每次可以写入 16 位或 32 位数据。主存储区擦除操作后，每 16 位或 32 位仅可以被编程一次，即被编程的 16 位或 32 位数据中有一个位如果不是 0，就不可以再次被烧写，否则编程错误标志位 PGERR@FLASH_SR 被置 1。

如果 Flash 主程序区的某一扇区已经被写保护位保护，但仍对该扇区进行写操作，则写保护错误标志位 WRPRTERR @FLASH_SR 被置 1。

8.3.3.7 类EEPROM 存储区擦除编程

类 EEPROM 存储区擦除编程与主程序区按扇区擦除、编程相类似，不同的是 Flash 解锁寄存器不同，写入 FLASH_CR 寄存器的操作命令字不同，锁定存储区的标志位不同。

类 EEPROM 存储区执行擦除操作后, 每 16 位仅可以被编程一次，即被编程的 16 位数据中有一个位如果不是 0，就不可以再次被烧写，否则编程错误标志位 PGERR@FLASH_SR 被置 1。

8.3.3.8 OTP 区编程

OTP 区只能一次编程，编程后无法擦除，OTP 区的编程与主程序区编程相类似，不同的是 Flash 解锁寄存器不同，写入 FLASH_CR 寄存器的操作命令字不同，锁定存储区的标志位不同。

8.3.3.9 代码保护

Flash 中的主程序区可以通过设置读保护防止被非法读出；同样可以设置写保护防止被非法修改。对于 SH33F2801 而言，写保护的基本单位是 4 个扇区为一个保护单元。

(1)读保护

读保护是通过设置读保护字节，写入后立即生效。

读保护有三个保护层级：无读保护、低级别读保护、高级别读保护。

无读保护：当将读保护寄存器设为 0xA55A 时，不执行读保护，对 Flash 的读、写、擦除操作不受限制。

低级别读保护：当设置读保护寄存器为 0xA55A 和 0xC33C 之外的其它值时，低级别读保护生效，此时读取到的读保护字节为 0xAAAA。

当低级别读保护生效后，只允许用户代码在非调试方式下对 Flash 主程序存储器进行读操作。读保护状态下，在 Flash 主存储器中执行的代码可以对 Flash 进行编程，实现 IAP 或数据存储等功能。需要注意的是**低级别读保护状态下，无法调试 Flash 代码，**但可以在调试模式下,向内置 RAM 装载代码并执行代码指令来解除低级别读保护（整片擦除）,也可以通过使用中颖工具（SWD/ISP）执行整片擦除来解除低级别读保护。

高级别读保护：当将读保护寄存器设为 0xC33C 时，高级别读保护生效。**在高级别读保护下，调试功能关闭，所以请用户谨慎选择高级别读保护功能。**

(2) 写保护

写保护位用于对 Flash 各扇区进行擦除/编程保护，可以阻止对 Flash 存储器的意外操作。写保护字节的 WRP[31:0]位为 1 可以使能指定扇区的写保护功能。SH33F2801 是以 4 个扇区为单位提供保护，对被保护的扇区进行编程或擦除操作将不被执行，同时产生操作错误标志。下表显示了写保护位与具体被保护扇区的映射。

表8-5 写保护位映射表

WP bit	被保护的扇区
WP[0]	Sector0 ~ Sector3
WP[1]	Sector4 ~ Sector7
WP[2]	Sector8 ~ Sector11
-	-
-	-
-	-
WP[30]	Sector120 ~ Sector123
WP[31]	Sector124 ~ Sector127

8.3.3.10 客户安全码

SH33F2801支持客户设置安全码功能，客户可以设置自定义的安全码来锁定烧写仿真工具对芯片的读写擦操作。客户安全码长度为6字节，客户可设置任意非全零的安全码。芯片出厂时，客户安全码默认是全零，当设置的有效安全码烧写入芯片且再次上电后，该芯片必须在正确输入安全码的情况下，才可以通过工具对芯片进行读写擦操作。中颖32Bit量产和仿真工具均支持客户安全码设置操作。

8.4 Flash 访问加速控制

SH33F2801增加了指令预读控制模块，用以解决系统高主频与Flash存储器访问慢的矛盾。对SH33F2801而言，在系统84MHz主频运行时，Flash存储器插入4个等待时间，开启预读缓冲区，可以获得接近零等待的系统执行性能。

8.5 Flash 切分区块功能

为保证片上备份程序与主程序能够快速切换，SH33F2801配置有FLASH切分区块功能。

与FLASH区块切分的相关代码选项参考“27.用户代码选项”。其中：

FLASH_BACKUP为FLASH分区使能位，用于控制FLASH分区功能是否使能；

FLASH_SELECT为FLASH逻辑地址映射切换控制位，仅在FLASH_BACKUP为‘1’时有效；

在Boot Loader以及Main程序中，允许对FLASH_BACKUP和FLASH_SELECT进行“擦除”以及“擦除 - 编程”的操作，此时需要通过将FLASH模块的LOCK寄存器写“0x5AA5”解锁后才可修改。具体操作方式与“客户信息区”烧写/擦除方式相同。

在修改FLASH_SELECT后，可通过软件复位(SW复位)或上电复位，产生Vector复位，使PC指针对应的逻辑地址映射完成切换。

上电复位或其他系统复位后，硬件会自动根据读到的FLASH_BACKUP以及FLASH_SELECT值，对FLASH的逻辑地址进行相应的映射处理。

图8-9 128KB FLASH 区块切分后地址配置框图

当 FLASH_SELECT 为‘0’时，FLASH 区的逻辑地址与物理地址相符；且系统复位后，PC 指针内容为：指向逻辑地址 0000 0000H，物理地址 0000 0000H；当 FLASH_SELECT 为‘1’时，FLASH 区的主 Memory 块和备份 Memory 块的逻辑地址调转，此时：

- Main Memory 块的逻辑地址 = 物理地址 + 0001 0000H
- Backup Memory 块的逻辑地址 = 物理地址 - 0001 0000H

且系统复位后，PC 指针内容为：指向逻辑地址 0000 0000H，物理地址 0001 0000H。

当FLASH_SELECT的值为‘0’时，PC指针指向Main Memory，当PC指针超过Main Memory的逻辑地址范围（0000 0000H~0000 FFFFH）时，则认为程序跑飞，产生总线Fault；当FLASH_SELECT的值为‘1’时，使能FLASH主/备份区块地址交换功能，PC指针指向Backup Memory，当PC指针超过Backup Memory的逻辑地址范围（0000 0000H~0000 FFFFH）时，则认为程序跑飞，产生总线错误。

总之，当FLASH分区块功能开启时，通过FLASH_SELECT可选择对应区块作为PC指针访问区域，若PC指针访问超出选中区块的范围，则认为程序跑飞，发生总线Fault。但需要注意，此时非选中的FLASH区块仍可作为ROM进行读/写操作，只是不允许运行程序。

8.6 寄存器

FLASH 模块寄存器列表 (基地址：0x4004 1000)

地址	寄存器名	说明
0x4004 1000	**ACR**	**Flash访问控制寄存器**
0x4004 1004	**MKYR**	**Flash主程序存储区解锁控制寄存器**
0x4004 1008	**E2KYR**	**类EEPROM存储区解锁控制寄存器**
0x4004 100C	**SR**	**Flash状态和清除寄存器**
0x4004 1010	**CR**	**Flash控制寄存器**
0x4004 101C	**OPR**	**Flash代码选项寄存器**
0x4004 1020	**RPR**	**Flash读保护寄存器**
0x4004 1024	**WRPR**	**Flash写保护寄存器**
0x4004 1030	**CNTR**	**Flash操作定时器计数寄存器**
0x4004 1034	**UPCNTR**	**Flash操作定时器上限值**
0x4004 1038	**CNTCR**	**Flash操作定时器启动寄存器**
0x4004 1040	**IKYR**	**Flash特殊信息存储区解锁控制寄存器**
0x4004 1100	**MEMRMP**	**内存映射选择寄存器**

8.6.1 Flash 访问控制寄存器 (FLASH_ACR)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
LOCK[15:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留				CRST	DCEN	保留	PRFTEN	保留					LATENCY[2:0]		
-				*RW*	*RW*	-	*RW*	-					*RW*		
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

位	符号	说明
31 ~ 16	**LOCK[15:0]**	寄存器锁定位 0x5AA5：解锁此寄存器 其他：锁定此寄存器 配置此寄存器时解锁位必须同时输入0x5AA5，操作完成后自动锁定，读取此LOCK位固定读出0x0000。
15 ~ 12	保留	-
11	**CRST**	复位缓冲区 0：不复位缓冲区 1：复位缓冲区（数据缓冲区和指令缓冲区同时复位） *Note：当数据缓冲区使能或预取缓冲区使能时，对存储区进行编程操作后，需要复位缓冲区，重新加载数据缓冲区或预取缓冲区数据，即CRST需要置1再清零。*
10	**DCEN**	数据缓冲区使能 0：关闭数据缓冲区 1：启用数据缓冲区
9	保留	-
8	**PRFTEN**	预取缓冲区使能 0：关闭预取缓冲区 1：启用预取缓冲区
7 ~ 3	保留	-
2 ~ 0	**LATENCY[2:0]**	**SYSCLK(系统时钟)周期与Flash访问时间配置设置** 000：0个等待状态 001：1个等待状态 010：2个等待状态 011：3个等待状态 100：4个等待状态 101：5个等待状态 110：6个等待状态 111：7个等待状态

8.6.2 Flash 主程序存储区解锁控制寄存器 (FLASH_MKYR)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
MKY[31:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
MKY[31:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	MKY[31:0]	**Flash主程序存储区解锁控制位** 依次写正确的Flash解锁值、操作解锁值到MKY[31:0]可以解锁Flash主程序存储区的操作，解锁成功后，位MNLCK@FLASH_CR被硬件清0，MKY[31:0]硬件清零。 对于Flash单次解锁操作，当位MNLCK@FLASH_CR被清0后，只允许对Flash进行一次编程或擦除操作，操作完成后位MNLCK@FLASH_CR自动置1，再次锁定Flash主程序存储区。 对于Flash多次解锁操作，当位MNLCK@FLASH_CR被清0后，允许对Flash进行多次编程或擦除操作，但操作完成后，必须将位MNLCK@FLASH_CR软件置1，才能再次锁定Flash主程序存储区。 这些位仅能被软件写。

8.6.3 类 EEPROM 存储区解锁控制寄存器 (FLASH_E2KYR)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
E2KY[31:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
E2KY[31:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	E2KY[31:0]	**类EEPROM存储区解锁控制位** 依次写正确的Flash解锁值、操作解锁值到E2KY[31:0]可以解锁类EEPROM区，解锁成功后，位E2WRE@FLASH_CR被硬件置1，E2KY[31:0]硬件清零。 对于Flash单次解锁操作，当解锁成功后，只允许对Flash进行一次编程或擦除操作，操作完成后位E2WRE@FLASH_CR硬件自动清0，再次锁定Flash相关区块。 对于Flash多次解锁操作，当解锁成功后，允许对Flash进行多次编程或擦除操作，但操作完成后，必须将位E2WRE@FLASH_CR软件清0，才能再次锁定Flash相关区块。 这些位仅能被软件写

8.6.4 Flash 状态和清除寄存器 (FLASH_SR)

偏移地址: 0x000C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留									PGWERRC	PGPERRC	WRPRTERRC	FLSERRC	保留	OPERRC	EOPC
-									WO	WO	WO	WO	-	WO	WO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
BSY	保留								PGWERR	PGPERR	WRPRTERR	FLSERR	保留	OPERR	EOP
RO	-								RO	RO	RO	RO	-	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 23	保留	-
22	PGWERRC	编程窗口错误标志清除位 0：无效 1：清除
21	PGPERRC	编程错误标志清除位 0：无效 1：清除
20	WRPRTERRC	写保护错误标志清除位 0：无效 1：清除
19	FLSERRC	Flash硬件错误状态清除位 0：无效 1：清除
18	保留	-
17	OPERRC	Flash操作错误标志清除位 0：无效 1：清除
16	EOPC	操作执行标志位 0：无效 1：清除
15	BSY	忙标志位 该位指示Flash操作正在进行。在Flash操作开始时，该位被设置为‘1’；在操作结束或发生错误时该位被清除为‘0’。
14 ~ 7	保留	-
6	PGWERR	编程窗口错误标志位 该位置‘1’，表示启动一次Flash操作时，即位STRT@FLASH_CR置‘1’或者启动对Flash的写操作时，Flash操作定时器计数值不在有效的时间窗口内，Flash操作不被执行。PGWERRC@FLASH_SR写入‘1’可以清除这位状态。
5	PGPERR	编程错误标志位 试图对已烧写过的地址编程（某一bit已不为0），硬件置‘1’。 PGPERRC@FLASH_SR写入‘1’可以清除这位状态。 PGPERRC@FLASH_SR写入‘1’可以清除这位状态。
4	WRPRTERR	写保护错误标志位 试图对写保护的Flash地址编程/擦除时，硬件置‘1’。 WRPRTERRC@FLASH_SR写入‘1’可以清除这位状态。

3	FLSERR	**Flash硬件错误** Flash操作超时，该错误标志置‘1’。FLSERRC@FLASH_SR写入‘1’可以清除这位状态。
2	保留	-
1	OPERR	**Flash操作错误标志位** Flash操作失败，硬件置‘1’。OPERRC@FLASH_SR写入‘1’可以清除这位状态。具体错误详见PGWERR、PGPERR、WRPRTERR、STAERR错误标志位。
0	EOP	**操作执行标志位** 当Flash执行完编程/擦除操作，硬件置‘1’。EOPC@FLASH_SR写入‘1’可以清除这位状态。 注：每次执行完编程或擦除都会设置EOP@FLASH_SR状态。

8.6.5 Flash 控制寄存器 (FLASH_CR)

偏移地址: 0x0010

复位值: 0x0000 D000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
CMD[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
MNLCK	E2LCK	保留	INFLCK	PSIZE	保留		STRT	保留	SNB[6:0]						
RW1s	*RW1s*	-	*RW1s*	*RW*	-		*RW*	-	*RW*						
1	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	CMD[15:0]	**Flash操作命令字** 0xE619：主程序区扇区擦除(MSE) 0x6E91：主程序区存储单元编程(MPG) 0xB44B：特殊信息区E2Prom区编程(IPG) 0x4BB4：特殊信息区E2Prom区按扇区擦除(类E2Prom)（E2SE） 0xF00F：特殊信息区OTP区编程（OPG）
15	MNLCK	**主程序区锁定标志** 依次向FLASH_MKYR写入正确的Flash解锁值、操作解锁值后，该位被硬件清零，此时才允许对Flash主程序存储区进行擦除、编程操作。 该位只能硬件清零，软件置‘1’。
14	E2LCK	**类EEPROM存储区锁定标志位** 依次向FLASH_E2KYR写入正确的Flash解锁值、操作解锁值后，该位被硬件清零，此时允许对类EEPROM存储区进行擦除、编程操作。 该位只能硬件清零，软件置‘1’。
13	保留	-
12	INFLCK	**特殊信息区锁定标志位(代码保护区、客户信息区、产品信息区和引导区)** 依次向FLASH_IKYR写入正确的Flash解锁值、操作解锁值后，该位被硬件清零，此时允许对特殊信息区(代码保护区、客户信息区、产品信息区和引导区)进行擦除、编程操作。 该位只能硬件清零，软件置‘1’。
11	PSIZE	**烧写操作位宽选择位** 0：32位同时烧写 1：16位同时烧写 注：当选择32Bit同时烧写时，也可以写入16bit数据烧写

10 ~ 9	保留	-
8	STRT	开始标志位 当该位为'1'时将触发一次操作。该位只可由软件置为'1'并在BSY@FLASH_SR变为'1'时清为'0'。
7	保留	-
6 ~ 0	SNB[6:0]	扇区擦除扇区选择位: 0000000：扇区0 0000001：扇区1 0000010：扇区2 …….. 1111110：扇区126 1111111：扇区127 注：该位仅当主程序区扇区擦除或类EEPROM区按扇区擦除时才有效，主程序区按扇区擦除时，每个扇区大小是1024字节；EEPROM区按扇区擦除时，每个扇区大小是1024字节。

8.6.6 Flash 代码选项寄存器 (FLASH_OPR)

偏移地址: 0x001C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
OPT3[7:0]								OPT2[7:0]							
RO								RO							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
OPT1[7:0]								OPT0[7:0]							
RO								RO							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 24	OPT3[7:0]	代码选项3
23 ~ 16	OPT2[7:0]	代码选项2
15 ~ 8	OPT1[7:0]	代码选项1
7 ~ 0	OPT0[7:0]	代码选项0

8.6.7 Flash 读保护寄存器 (FLASH_RPR)

偏移地址: 0x0020

复位值: 0x0000 XXXX

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RDP[15:0]															
RO															
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	RDP[15:0]	读保护 该寄存器包含由复位后客户信息区加载器加载的读保护选项字节。

8.6.8 Flash 写保护寄存器 (FLASH_WRPR)

偏移地址: 0x0024

复位值: 0xXXXX XXXX

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
WRP[31:0]															
RO															
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
WRP[31:0]															
RO															
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

位	符号	说明
31 ~ 0	WRP[31:0]	写保护该寄存器包含由复位后客户信息区加载器加载的写保护字节。 0：写保护失效 1：写保护生效 注意：这些位为只读。

8.6.9 Flash 操作定时器计数寄存器 (FLASH_CNTR)

偏移地址: 0x0030

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
CNT[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CNT[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	CNT[31:0]	**Flash操作定时器计数值寄存器** 基于系统时钟的Flash操作定时器，根据CNTEN@CNTCR位使能或禁能递减计数功能，当计数器值递减至Flash操作定时器上限值后，才允许操作Flash存储器，当Flash操作定时器计数值递减至零后，禁止操作Flash存储器。

8.6.10 Flash 操作定时器上限值 (FLASH_UPCNTR)

偏移地址: 0x0034

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
UPCNT[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
UPCNT[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	UPCNT[31:0]	**Flash操作定时器上限值** Flash操作定时器计数值递减至或初始设置值小于Flash操作定时器上限值后，才允许操作Flash存储器，Flash操作定时器计数值递减至零后，禁止操作Flash存储器。 注：当UPCNT=0且CNT=0时，禁止操作Flash存储器。

8.6.11 Flash 操作定时器启动寄存器 (FLASH_CNTCR)

偏移地址: 0x0038

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留															CNTEN
-															RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 1	保留	-
0	CNTEN	**Flash操作定时器启动位** 0：Flash操作定时器禁能 1：Flash操作定时器使能 注：CNTEN@CNTCR置1，Flash操作定时器递减计数使能，计数递减到零，该位清0，Flash操作定时器禁能。

8.6.12 Flash 特殊信息存储区解锁控制寄存器 (FLASH_IKYR)

偏移地址: 0x0040

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
INFO_KEY[31:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
INFO_KEY[31:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	INFO_KEY[31:0]	**Flash特殊信息存储区解锁控制位** 依次写Flash解锁值，操作解锁值到INFO_KEY[31:0]可以解锁Flash特殊信息区（代码保护区、客户信息区），解锁成功后，位INFLCK@FLASH_CR被硬件清‘0’，INFO_KEY[31:0]硬件清零。 对于Flash单次解锁操作，当解锁成功后，只允许对Flash进行一次编程或擦除操作，操作完成后位INFLCK@FLASH_CR硬件自动置‘1’，再次锁定Flash相关区块。 对于Flash多次解锁操作，当解锁成功后，允许对Flash进行多次编程或擦除操作，但操作完成后，必须将位INFLCK@FLASH_CR软件置‘1’，才能再次锁定Flash相关区块。 这些位仅能被软件写。

8.6.13 内存映射选择寄存器 (FLASH_MEMRMP)

偏移地址: 0x0100

复位值: 0x0000 XXXX

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
MEMMODE[15:0]															
RW															
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	MEMMODE[15:0]	**内存映射选择(Memory mapping selection)** 这些位在复位后由硬件设置，软件可读可写。 0x5AA5: Boot区映射到地址0x0 其他: 主闪存存储器映射到地址0x0 任何时刻只要不满足0x5AA5，即映射到主闪存存储器

9. 电源控制（PWR）

SH33F2801的系统工作电压(VDD)为2.4～5.5V。通过内置的电压调节器提供系统内核电源。SH33F2801内部有一套完整的上电复位和掉电检测电路，后者在一些电池应用中具有重要意义。

9.1 电源与时钟预热

- 内建电源预热计数器消除电源上电不稳定状态
- 内建振荡器预热计数器消除振荡器起振不稳定状态

上电后，SH33F2801会先经过电源上电预热计数过程，等待溢出后再进行振荡器的预热计数过程，溢出后开始运行程序。

表9-1　　电源上电预热计数时间

上电复位（POR）		RESET复位 / LVR复位		IWDT复位（不在停机模式下）		IWDT复位（从停机模式下唤醒）		从停机模式下唤醒（中断或事件唤醒）	
电源上电预热时间	振荡器上电预热时间	电源上电预热时间	振荡器上电预热时间	电源上电预热时间	振荡器上电预热时间	电源上电预热时间	振荡器上电预热时间	电源上电预热时间	振荡器上电预热时间
约8.5ms	约16us	约1ms	约16us	约1ms	无	约1ms	约16us	约270us	约16us

9.2 掉电检测（BOD）

- 通过寄存器选择开启
- 上升和下降都能产生中断
- 可调的检测电压
- 带回差，减少电压波动影响
- 硬件去抖动，去抖时间约为60μs

掉电检测用来监测电源电压，如果电压高于或低于指定值时产生内部标志，并产生中断。主要用于检测电源电压的变化，用于监控系统供电情况。在某些应用中，可用于低电压检测，系统软件可据此采取一些保护措施，如备份数据、保留现场等。

BOD可以唤醒停机。下图给出了在开启上升和下降双向中断（BODMD=10b），带回差条件下的BOD工作波形：

由于具备双向检测功能，BOD的使用变得很简单，典型使用步骤如下：

BOD中断开启步骤：

1）使能BOD模块（BODEN=1）；

2）设置BOD检测档位寄存器VBOD[3:0]；

3）设置BOD双沿检测BODMD[1:0]=10b；

4）清零BOD中断请求标志位BODIF；

5）置位BOD中断允许位BODIE（NVIC相应通道需配合开启）。

在 BOD 中断中，做如下工作：

1）清零BOD中断允许位BODIE；

2）查询BODF，确定刚才发生的是上升还是下降变化；

3）根据2）的查询结果执行相应的操作（如果是单向检测，则不需要查询BODF）；

4）清零BOD中断请求标志位BODIF；

5）置位BOD中断允许位BODIE；

BOD也能引起NMI中断，由IEN_BOD@SYSCFG_SAFR控制位使能。

9.3 低电压复位（LVR）

在交流电或大容量电池应用中，接通大负载后容易导致MCU供电暂时低于定义的工作电压。低电压复位即是保护系统在低于设定电压下产生有效复位。

- 通过代码选项选择开启，并通过代码选项设定 LVR 电压
- LVR 去抖动时间 T_{LVR} 约 60μs
- 当供电电压低于设定电压 V_{LVR} 时，将产生内部复位

LVR功能打开后，具有以下特性（t表示电压低于设定电压V_{LVR}的时间）：

当$V_{DD} \le V_{LVR}$且$t \ge T_{LVR}$时产生系统复位。

当$V_{DD} > V_{LVR}$或$V_{DD} < V_{LVR}$，但持续时间$t < T_{LVR}$时不会产生系统复位。

上图中, V_{DD}为电源电压, V_{LVR}为LVR检测电压, V_{HYS}为回差。

9.4 低功耗管理

SH33F2801支持两种低功耗模式：

- 睡眠模式(Sleep Mode)：基本低功耗模式，MCU内核停止运行，但用到的外设包括内核的核心外设，如NVIC、SysTick等仍在运行
- 停机模式(Stop Mode)：深度低功耗模式，除了内核停机，用到的外设大部分都停止工作，只剩下担负唤醒任务的个别外设可保持运行（通过LSI提供时钟），如IWDT、BOD、LVR等

表9-2 低功耗模式一览

模式	进入	唤醒	唤醒延时	VDD/AVDD电源域
睡眠模式	WFI/WFE，保持SLEEPDEEP=0	任一中断/唤醒事件/复位信号	无	1. ARM “星辰”处理器时钟停止； 2. 振荡器和电源保持原状态，外设时钟和部分模拟外设电源可软件关闭；
停机模式	WFI/WFE，保持SLEEPDEEP=1	任一外部中断/复位信号	1. HSI唤醒时间； 2. HSE和PLL唤醒时间（假设有使用）； 3. 内核供电唤醒时间；	1. HSI、HSE以及PLL均被禁用，使用这部分时钟的外设全部停止； 2. LSI可开启，使用此时钟外设可正常工作； 3. 部分模拟外设电源可软件关闭；

9.4.1 睡眠模式

睡眠模式能够降低系统功耗，在此模式下，程序中止运行，CPU时钟停止，但外部设备时钟继续运行。睡眠模式下，CPU在确定的状态下停止，并在进入睡眠模式前所有CPU的状态都被保存（包括系统时钟寄存器），SRAM和寄存器内容被保留，GPIO保持进入时电平状态。

进入睡眠模式

保持ARM “星辰”处理器系统控制寄存器中的SLEEPDEEP位处于复位状态，执行WFI或WFE指令就可以进入睡眠模式。根据ARM “星辰”处理器系统控制寄存器中的SLEEPONEXIT位的值，有2个CPU选项影响进入睡眠模式：

- SLEEP-NOW：当WFI或WFE被执行时，微控制器立即进入睡眠模式；
- SLEEP-ON-EXIT：如果SLEEP-ON-EXIT位被置位，系统从最低优先级的中断处理程序中退出时，微控制器就立即进入睡眠模式。

说明：SLEEP-ON-EXIT是ARM内核特有的“自动睡眠”的机制——它可以被编程为从中断服务例程返回后立即睡眠。这样一来，处理器的所有工作就只是响应中断，其它时间都在睡眠。

退出睡眠模式

如果执行WFI指令进入睡眠模式，任意一个被嵌套中断向量控制器响应的外设中断都能将系统从睡眠模式唤醒[注]。

如果执行WFE指令进入睡眠模式，则一旦发生唤醒事件时，微处理器都将从睡眠模式退出。唤醒事件可以通过下述方式产生：

- 在外设控制寄存器中使能一个中断，但不在NVIC中使能，并且在ARM “星辰”处理器系统控制寄存器中使能SEVONPEND位，这样就可以用中断信号把系统从WFE中唤醒。当MCU从WFE中唤醒后，外设的中断挂起位和外设的NVIC中断通道挂起位(在NVIC中断清除挂起寄存器中)必须被清除。
- 配置一个外部事件(EXTI)输入，通过外部事件把系统从WFE中唤醒。当MCU从WFE中唤醒后，因为与事件线对应的挂起位未被设置，不必清除外设的中断挂起位或外设的NVIC中断通道挂起位。该模式唤醒所需的时间最短，因为没有时间损失在中断的进入或退出上。

说明：

（1）当从WFI 唤醒时，要根据异常系统的优先级来决定是否唤醒。只有当该中断的优先级比当前优先级要高（如果是在中断服务例程中使用WFI），并且比BASEPRI 掩蔽的高时，才唤醒处理器并执行ISR。但如果PRIMASK 置位，则依然唤醒处理器，而ISR 不执行。

（2）WFE 则不管优先级和掩蔽情况如何，只要SEVONPEND 置位，任何一个事件发生时一定把处理器唤醒。至于是否执行ISR，则与WFI 的情况相同。

关于如何退出睡眠模式，更多的细节参考表9-3和表9-4。

表9-3　　SLEEP-NOW 模式

SLEEP-NOW模式	说明
进入	在以下条件下执行WFI(等待中断)或WFE(等待事件)指令： – SLEEPDEEP = 0 和 – SLEEPONEXIT = 0 参考ARM “星辰”处理器系统控制寄存器。
退出	如果执行WFI进入睡眠模式： 中断：参考中断向量表 如果执行WFE进入睡眠模式： 唤醒事件：参考唤醒事件管理
唤醒延时	无

表9-4　　SLEEP-ON-EXIT 模式

SLEEP-ON-EXIT模式	说明
进入	在以下条件下执行WFI指令： – SLEEPDEEP = 0 和 – SLEEPONEXIT = 1 参考ARM “星辰”处理器系统控制寄存器。 注：为了进入睡眠模式，所有的外部中断的请求位(挂起寄存器(EXTI_PR))标志必须被清除，否则睡眠模式的进入流程将会被跳过，程序继续运行。
退出	中断：参考中断向量表
唤醒延时	无

复位信号（RESET，WDT，BOD、LVR）都能退出睡眠模式。

系统唤醒后，立即恢复CPU时钟，如果中断唤醒还需要执行中断服务程序，随后跳转到进入睡眠模式指令之后的指令。唤醒后SRAM保持不变，而系统寄存器和外设寄存器的值根据不同功能模块设置可能会改变（外设时钟和使能都没关，在睡眠期间还会运行，会有计数值和状态值的变化）。

注：睡眠模式下高频时钟是不停的，唤醒后可以直接从原系统时钟开始运行，不需要时钟切换。

如果是复位信号唤醒，则系统从地址0x0000 0000开始执行，SRAM和寄存器按照复位规则处理。

9.4.2　停机模式

停机模式可以使芯片进入功耗非常低的状态。停机模式将停止CPU和外围设备的所有时钟信号（除LSI可在停机模式下开启），同时还能设置内核电压调节器进入低功耗状态。如果IWDT使能，IWDT模块将继续工作，LVR和BOD电路开启时也处于工作状态。在进入停机模式前所有CPU的状态都被保存，SRAM和寄存器内容被保留，GPIO保持进入时电平状态，但PWM输出（含MCM和PCA模块）将固定被切换到高阻状态。

说明：对数字外设，如UART、SPI、TIMER等，停机时虽然模块使能未关闭，但由于时钟切断，模块处于停止状态，模块引脚电平保持为停机前电平。

对模拟外设，如ADC、OP、CMP，如果模块使能未关闭，停机时其数字部分时钟被关闭停止工作，但模拟部分供电不会自动切断，可根据控制要求手动关闭这部分供电（一般是关闭模块使能）。

PWM模块（含MCM和PCA）停机时除了停止工作外，还会强制切换到高阻输出，以保护外部驱动电路。应用上，为避免停机前后的状态变化影响系统工作，建议用户在进入停机前手动关闭PWM模块，在唤醒后重新初始化PWM后开启。

停机时PLL、HSI和HSE电路都被自动关闭。

系统进入停机模式时，RCC模块的SW、SWS、PLLON、PLLRDY、HSEON、HSERDY、PLLRDYIE、PLLRDYIF、HSERDYIE、HSERDYIF等控制位处于复位状态。

停机模式下，以下模块可以正常运行：

- 独立看门狗（IWDT），一旦启动IWDT，除了系统复位，它不能再被停止[注]；
- 掉电检测（BOD），由用户开启；
- 低频内部RC时钟（LSI），始终开启；
- 唤醒定时器（Wakeup Timer），使用TIM7/TIM8定时器，要求配置LSICLK时钟源，或外部输入时钟；

*注：默认情况下IWDT在停机模式下开启，这种情况下IWDT会唤醒停机。可以通过用户代码选项“**OP_WDTPD**”修改。*

停机模式下时钟安全监控（CSM）模块无效，睡眠模式下CSM可正常工作。

进入停机模式

关于如何进入停机模式，详见表9-5。

- 如果正在进行闪存编程，直到对闪存访问完成，系统才进入停机模式；
- 如果正在进行对AHB/APB的访问，直到对AHB/APB访问完成，系统才进入停机模式；
- 如果在中断服务程序中，或者有中断在排队，系统正常进入停机模式，但EXTI、BOD、Wakeup Timer三种中断除外，这三种中断产生时将跳过停机流程，导致进入停机操作失败，除非清除掉相应的中断请求标志。

表9-5　　STOP 模式

STOP模式	说明
进入	在以下条件下执行WFI(等待中断)或WFE(等待事件)指令： – 设置ARM “星辰”处理器系统控制寄存器中的SLEEPDEEP位 注：为了进入停机模式，所有的外部中断的请求位(挂起寄存器(EXTI_PR))标志必须被清除，否则停机模式的进入流程将会被跳过，程序继续运行。
退出	如果执行WFI进入停机模式： 设置任一外部中断线为中断模式(在NVIC中必须使能相应的外部中断向量)。参见NVIC的“中断向量表”。 如果执行WFE进入停机模式： 设置任一外部中断线为事件模式。参见EXTI的“唤醒事件管理” 。
唤醒延时	HSI RC唤醒时间 + 内核供电从低功耗状态唤醒的时间

退出停机模式

关于如何退出停机模式，详见表9-5。

当一个中断或唤醒事件导致退出停机模式时，HSI会被自动开启，并被选为系统时钟[注]，然后由软件控制逐步开启HSE、PLL（如果有使用）等电路并切换系统时钟至目标时钟源，和系统上电时类似，需要等待电源预热和时钟预热。

注：睡眠模式系统时钟不切换，停机模式系统时钟被切换。

复位信号（RESET，IWDT，BOD、LVR等）能退出停机模式，复位后程序从0x0000 0000地址开始运行。

如果通过中断唤醒退出停机模式，还需要执行中断服务程序，随后跳转到STOP之后的指令。

中断唤醒源包括EXTI中断、BOD掉电中断和Wakeup Timer中断。

如果中断使能未开启，还可以通过EXTI事件、Wakeup Timer溢出事件进行唤醒[注]。

注：一般而言事件和中断的源头可以是同一个，EXTI有中断和事件两条分支，而Wakeup Timer只有中断使能位控制，开启中断使能的结果是唤醒（WFI和WFE）以后还要进入相应中断服务程序。

9.4.3　低功耗模式下的调试

默认情况下，如果在进行调试微处理器时，使微处理器进入低功耗模式，将失去调试连接。这是因为ARM “星辰”处理器的内核失去了时钟。 然而，通过设置DBGMCU_CR寄存器中的某些配置位，可以在使用低功耗模式下调试程序。

更多的细节请参考调试接口有关“低功耗模式下调试支持”的内容。

9.5　寄存器

SYSCFG 模块寄存器列表 (基地址：0x4004 0800)

地址	寄存器名	说明
0x4004 0800	PWRCR	电源控制寄存器
0x4004 0804	PWRSR	电源状态寄存器

9.5.1 电源控制寄存器 (SYSCFG_PWRCR)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留						LVREN	VLVR	BODEN	BODIE	BODMD[1:0]		VBOD[3:0]			
-						*RW*	*RW*	*RW*	*RW*	*RW*		*RW*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
9	**LVREN**	**LVR使能位** 0：禁止低电压复位功能 1：允许低电压复位功能
8	**VLVR**	**LVR复位电压档位设置位** 0: 4.1 V 1: 2.8 V
7	**BODEN**	**BOD允许位** 0：禁止掉电检测 1：允许掉电检测
6	**BODIE**	**BOD中断使能位**
5 ~ 4	**BODMD[1:0]**	**BOD模式选择控制位** 00：仅当VDD电压从小于BOD门限到大于BOD门限时，BODIF标志置1 01：仅当VDD电压从大于BOD门限到小于BOD门限时，BODIF标志置1 10：当VDD电压从大于门限到小于门限，或从小于门限到大于门限，BODIF标志都置1 11：保留
3 ~ 0	**VBOD[3:0]**	**BOD门限电压设置位** 0000: 2.80 V 0001: 2.90 V 0010: 3.00 V 0011: 3.10 V 0100: 3.20 V 0101: 3.30 V 0110: 3.40 V 0111: 3.50 V 1000: 3.60 V 1001: 3.70 V 1010: 3.80 V 1011: 3.90 V 1100: 4.00 V 1101: 4.10 V 1110: 4.20 V 1111: 4.30 V

9.5.2 电源状态寄存器 (SYSCFG_PWRSR)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留														BODF	BODIF
-														*RO*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 2	保留	-
1	**BODF**	**BOD标志位** 0：当前电压高于在VBOD[3:0]中设置的BOD电压 1：当前电压低于在VBOD[3:0]中设置的BOD电压
0	**BODIF**	**BOD中断请求标志** 0：无中断挂起，清除挂起 1：中断挂起

10. 复位和时钟控制（RCC）

10.1 复位

SH33F2801的复位信号分成两种类型：系统复位（SYSRESET）和电源复位（PWRESET）。

10.1.1 系统复位

除了时钟控制器的RCC_RSTSTR寄存器中的复位标志位外，系统复位将复位所有寄存器至它们的复位状态。 当发生以下任一事件时，产生一个系统复位：

- NRST引脚上的低电平(外部复位)
- 独立看门狗计数终止(IWDT复位)
- 窗口看门狗计数终止(WWDT复位)
- 软件复位(SW复位)

可通过查看RCC_RSTSTR控制状态寄存器中的复位状态标志位识别复位事件来源。

10.1.1.1 引脚复位

SH33F2801有一个独立的复位引脚NRST，该引脚上的一段时间低电平将引起系统复位。

10.1.1.2 独立看门狗复位（IWDT）

独立看门狗复位后产生IWDTRSTF@RCC_RSTSTR标志。

具体见“独立看门狗”章节介绍。

10.1.1.3 窗口看门狗复位 （WWDT）

窗口看门狗复位后产生WWDTRSTF@RCC_RSTSTR标志。

具体见“窗口看门狗”章节介绍。

10.1.1.4 软件复位

通过将ARM “星辰”处理器的“应用中断与复位控制寄存器”中的SYSRESETREQ位置‘1’，可实现软件复位（内核和外设）。

10.1.2 电源复位

当以下事件中之一发生时，产生电源复位：

- 上电复位(POR复位)
- 低电压复位（LVR复位）

电源复位将复位所有寄存器。复位入口矢量被固定在地址0x0000_0004。

10.1.2.1 上电复位

SH33F2801内部有一个完整的上电复位（POR）电路，当供电电压达到V_{POR}时系统即能正常工作。当VDD低于指定的限位电压V_{POR}时，系统保持为复位状态。

10.1.2.2 低电压复位

低电压复位（LVR）是可以通过软件使能，当供电电压低于V_{LVR}时，MCU将产生复位。

具体见“低电压复位（LVR）”章节介绍。

10.2 时钟

三种不同的时钟源可被用来驱动系统时钟(SYSCLK)：

- HSI振荡器时钟（8MHz）
- HSE振荡器时钟（4MHz～12MHz）
- PLL时钟 （max：84MHz）

其中PLL时钟可选择HSI和HSE作为时钟源。

图10-1 SH33F2801 的时钟结构框图

用户可通过多个预分频器配置AHB和APB的频率。

- HCLK：AHB域主时钟，最高频率是84MHz，包括ARM “星辰”处理器、内存、MCM等都由HCLK驱动。
- PCLK：APB域主时钟，最高频率是42MHz，APB总线上的外设都由PCLK驱动。

RCC通过AHB时钟(HCLK)8分频后作为SysTick的外部时钟。通过对SysTick控制与状态寄存器的设置，可选择上述时钟或内核时钟作为SysTick时钟源。

10.2.1 HSE 时钟

高速外部时钟信号(HSE)由以下两种时钟源产生：

(1) 外部晶体/陶瓷谐振器：4MHz - 12MHz

4~12Mz外部振荡器（包含晶体谐振器和陶瓷谐振器）可为系统提供更为精确的主时钟。在时钟控制寄存器HSERDY@RCC_CR位用来指示高速外部振荡器是否稳定。在启动时，直到这一位被硬件置‘1’，时钟才被释放出来。如果在时钟中断寄存器RCC_CIENR中允许产生中断，将会产生相应中断。

HSE晶体可以通过设置时钟控制寄存器里HSEON@RCC_CR位被启动和关闭。

为了减少时钟输出的失真和缩短启动稳定时间，晶体/陶瓷谐振器和负载电容器必须尽可能地靠近振荡器引脚。负载电容值必须根据所选择的振荡器来调整。

(2) 外部输入时钟：4MHz – 12MHz

在这个模式里，必须提供外部时钟。它的频率最高可达12MHz。这种模式下用户先要在XTAL1上灌入外部时钟，然后开启HSEON位，此时XTAL2必须悬空。但可以通过设置OSCCFG@SYSCFG_SAFR位为10b来设置XTAL2复用为GPIO口使用。

和外部振荡器一样，外部时钟也存在预热时间和Ready信号。

外部时钟也受CSM监控。

10.2.2 HSI 时钟

HSI时钟信号由内部8MHz的RC振荡器产生，可直接作为系统时钟或作为PLL输入，作为PLL输入时，建议打开PLL预分频因子PLLXTPRE@RCC_CFGR，即启动二分频。

HSI还有如下特性：

- 系统复位后，选择HSI作为系统时钟。
- HSI上电后是常开的（无论系统时钟是否用HSI），在停机模式下关闭。
- HSI RC振荡器的启动时间比HSE晶体振荡器短。
- HSI RC在出厂时会被校准到0.2%(25°C)，在全温度范围不超过1%，系统复位时，工厂校准值被装载到时钟控制寄存器的HSICAL[7:0]位。 如果用户的应用基于不同的电压或环境温度，这将会影响RC振荡器的精度，可以通过时钟控制寄存器里的HSITRIM[2:0]位来进一步调整HSI频率。

10.2.3 PLL

内部PLL可以用来倍频HSI的内部RC输出时钟或HSE的外部晶体振荡器输出时钟。

PLL的设置必须在其被激活前完成。这些参数包括PLLSRC、PLLXTPRE、PLLF、PLLK，一旦PLL被激活，这些参数就不能被改动。

如果PLL中断在时钟中断寄存器里被允许，当PLL准备就绪时，可产生中断申请。

PLL倍频系数由3个参数组成，分别是N、F和K，计算公式是：

$$PRECLK = \frac{PLLCLKIN}{N} \qquad (1)$$

$$VCOCLK = PRECLK \times F \qquad (2)$$

$$PLLCLK = \frac{VCOCLK}{K} \times \frac{1}{2} \qquad (3)$$

N：由预分频因子PLLXTPRE决定，取1或2；

F：由倍频因子PLLF指定，取值范围15～78；

K：由分频因子PLLK指定，取值范围1～8；

对于HSI时钟源，建议预分频因子N取2。

对于HSE时钟源，N取1。

对PLL模块而言，PLLCLKIN输入时钟范围4MHz～12MHz，PRECLK预分频时钟要求满足4MHz～8MHz范围，VCOCLK倍频输出时钟要求满足150MHz～300MHz范围。PLL的设置除了要满足上述时钟范围要求，也需要满足系统主频范围要求（4MHz～84MHz）。

以设置84MHz系统主频为例，

（1） 选择8MHz作为时钟源，设置PLLXTPRE=0（N=1），PLLF=6（F=21），PLLK=0（K=1），则PRECLK=8MHz，VCOCLK=168MHz，输出PLLCLK为：8×21/1/2=84MHz；

（2） 选择8MHz作为时钟源，设置PLLXTPRE=1（N=2），PLLF=27（F=42），PLLK=0（K=1），则PRECLK=4MHz，VCOCLK=168MHz，输出PLLCLK为：4×42/1/2=84MHz。

10.2.4 LSI 时钟

LSI RC提供一个低功耗时钟源，用于停机唤醒模块的驱动，包括IWDT和Wakeup Timer。LSI不能用于系统时钟。LSI时钟频率在128kHz左右，LSI RC不是一个精确的时钟源， 最大误差会达到±40%左右，LSI RC不支持校准。但LSI（RC 128kHz）常温25度实际测量值与理论值的误差已写入0FFF F050H（误差值低8位）和0FFF F051H（误差值高8位）。误差值具体计算公式是(测量值-128000) /100，测量值单位Hz，用户可自行读取使用。

LSI上电后是常开的。在停机模式下也保持开启。

10.2.5 系统时钟(SYSCLK)选择

系统复位后，HSI振荡器被选为系统时钟。只有当目标时钟源准备就绪了(HSE就绪或PLL就绪)，才允许从HSI时钟切换到HSE或PLL时钟。

系统时钟由RCC_CR寄存器的SW控制位设定。

注意：要求在HSE或PLL就绪以后再切换系统时钟（SW控制位），否则有可能会出现控制位与实际时钟源不对应的情况，如以下两种：

（1） CSM未打开，HSE或PLL未就绪就切换到目标时钟，此时切换不会成功，保持在之前时钟状态，但SW控制位已经被修改，出现不对应；

（2） CSM打开，发生时钟异常后系统时钟切换到HSI，此时SW仍然停留在原时钟配置（下节“CSM”模块有详细介绍）。

10.2.6 时钟安全监控（CSM）

为了增强系统的可靠性，防止因时钟失效造成系统出错甚至死机的严重后果，SH33F2801增加了一个时钟安全监控（CSM）模块。CSM监控两个源头：

（1） HSE的振荡情况，包含晶振、陶振或外部时钟输入的情况，一旦发生停振或振荡异常事件（CSM检测频率下限在2MHz左右，上限不作要求），则送出异常标志HSECSMF；

（2） PLL的失锁情况（使用PLL的情况下），一旦发生失锁，则送出异常标志PLLCSMF；

只有对应的模块（HSE和PLL）打开才会进行相应的异常检测，如果HSE关闭或PLL关闭，则CSM检测相应模块失效功能自动关闭。

用户可以在RCC@CISTR来查询相应的中断标志以确定是何种异常。如果上述异常会影响系统时钟，则会申请CSM中断，CSM中断为NMI不可屏蔽中断（由IEN_CSM@SYSCFG.SAFR单独控制使能）。具体系统时钟和异常关系，以及何时产生CSM中断，参见下面的“CSM异常功能定义和处理表”。

CSM可以通过软件设置，CSM开启时只有在HSE就绪或PLL就绪后监控才有效。

表10-1 CSM 异常功能定义和处理表（只针对常规时钟配置情况）

系统时钟	HSE异常情况	PLL异常情况	CSM标志	CSM处理（CSM开启NMI中断）
HSI	0	1	（1）如果PLLON=1，则PLLCSMF=1；如果PLLON=0，则PLLCSMF=0（注1）	（1）HSERDY、HSERDYIF、PLLRDY 、 PLLRDYIF 随 着 HSECSMF 和 PLLCSMF 的 置位而清零，其他flag不变 （2）CSM标志不会产生CSM中断，对系统也不会有其他影响
	1	0	（1）如果HSEON=1，则HSECSMF=1；如果HSEON=0，则HSECSMF=0	
	1	1	（1）如果HSEON=1，则HSECSMF =1；如果HSEON=0，则HSECSMF=0 （2）如果PLLON=1，则PLLCSMF=1，如果PLLON=0，则PLLCSMF=0	
HSI+PLL	0	1	（1）PLLCSMF=1	（1）PLLRDY、PLLRDYIF随着HSECSMF和PLLCSMF的置位而清零，其他flag不变 （2）系统时钟切换到HSI （3）产生电机保护信号 （4）产生CSM中断
	1	0	（1）如果HSEON=1，则HSECSMF=1；如果HSEON=0，则HSECSMF=0	（1）HSERDY、HSERDYIF、随着HSECSMF的置位而清零，其他flag不变 （2）CSM标志不会产生CSM中断，对系统也不会有其他影响
	1	1	（1）PLLCSMF=1 （2）如果HSEON=1，则HSECSMF=1；如果HSEON=0，则HSECSMF=0	（1）HSERDY、HSERDYIF、PLLRDY 、 PLLRDYIF 随 着 HSECSMF 和 PLLCSMF 的 置位而清零，其他flag不变 （2）系统时钟切换到HSI （3）产生电机保护信号 （4）产生CSM中断

HSE	0	1	（1）如果PLLON=1，则PLLCSMF=1；如果PLLON=0，则PLLCSMF=0	（1）PLLRDY、PLLRDYIF随着PLLCSMF的置位而清零，其他flag不变 （2）CSM标志不会产生CSM中断，对系统也不会有其他影响
	1	0	（1）HSECSMF=1	（1）HSERDY、HSERDYIF随着HSECSMF的置位而清零，其他flag不变 （2）系统时钟切换到HSI （3）产生电机保护信号（注2） （4）产生CSM中断
	1	1	（1）HSECSMF=1 （2）如果PLLON=1，则PLLCSMF=1，如果PLLON=0，则PLLCSMF=0	（1）HSERDY、HSERDYIF、PLLRDY、PLLRDYIF随着HSECSMF和PLLCSMF的置位而清零，其他flag不变 （2）系统时钟切换到HSI （3）产生电机保护信号 （4）产生CSM中断
HSE+PLL	0	1	（1）PLLCSMF=1	（1）PLLRDY、PLLRDYIF随着PLLCSMF的置位而清零，其他flag不变。 （2）系统时钟切换到HSI； （3）产生电机保护信号； （4）产生CSM中断；
	1	0	（1）HSECSMF=1	（1）HSERDY、HSERDYIF随着HSECSMF的置位而清零，其他flag不变 （2）系统时钟切换到HSI （3）产生电机保护信号 （4）产生CSM中断
	1	1	（1）PLLCSMF=1 （2）HSECSMF=1 （注3）	（1）HSERDY、HSERDYIF、PLLRDY、PLLRDYIF随着HSECSMF和PLLCSMF的置位而清零，其他flag不变 （2）系统时钟切换到HSI （3）产生电机保护信号 （4）产生CSM中断

注1：严格来讲，PLLON=0时PLL模块关闭，不会发生“PLL异常”情况。此处为了说明完整性，把此种情况也列出来。

注2：即使HSE与HSI频率相同，但在时钟切换时，为了安全起见，也会产生电机保护信号。

注3：在HSE+PLL作为系统时钟时，如果HSE出现异常，有可能会造成PLL也会失锁，即实际失效场景中HSE与PLL之间可能是有关联的。 此处我们以冗余方式对HSE和PLL进行独立检测，即认为二者任一失效即产生CSM中断。

从上表中可以看出，即使HSE没有作为系统时钟，只要HSEON打开，一旦HSE停振，会置位CSM异常标志，但不会产生CSM中断。同样的，即使PLL没有用作系统时钟，只要PLLON打开，一旦PLL失锁，会置位异常标志，但不会产生CSM中断。

发生CSM后，除了HSECSMF和PLLCSMF被置位外，RCC的一些控制位不会清零，如HSEON、PLLON、SW、SWS都保持原值，即此时系统时钟已经切换到HSI时钟，但SW仍保持原设置。

从时钟失效状态恢复时，一旦失效的时钟状态恢复（以READY状态位被置起为准），用户只需要clear CSM flag即可恢复原系统时钟，不需要切换SW（SW保持原设置）。需要注意的是，硬件上并没有在恢复时钟时自动检测READY位，即如果clear CSM flag时目标时钟仍然处于失效状态，系统时钟会先切换到目标时钟，然后再瞬间切换回HSI时钟，这种情况也是不期望的，因此正确的做法是在恢复时钟之前先查询READY位，READY位置起说明对应时钟已恢复，才能clear CSM flag。

因为CSM中断是不可屏蔽中断（通过IEN_CSM开启），因此软件中必须对此进行处理。

下面代码示意了时钟失效与恢复的过程：假设系统时钟是HSI+PLL， PLL失锁引起时钟失效。

```
Void NMIException(void)
{
    if (RCC_GetITStatus(RCC_IT_CSM) != RESET)        //产生了时钟失效中断
    {
        // 系统时钟已经自动切换到HSI
        // 用户可以在此添加应用相关的保护代码
         /* 下面准备恢复时钟，通过查询方式 */
         /*****************************************************************/
          if (RCC_GetINTStatus (RCC_INT_PLLRDY) != RESET)
          {
                    RCC_ClearITPendingBit(RCC_IT_CSM);      // 清除中断挂起位
                                                            // 因为SW仍然保持原值，清除CSM时立刻恢复原时钟
          }
         /*****************************************************************/
    }
}
```

10.2.7 系统节拍定时器（SysTick）

SysTick是一个24位倒计数定时器，在ARM “星辰”处理器中实现。可提供一个固定间隔的时间节拍，在主流程中可作为监控或定时使用。因为所有ARM “星辰”处理器芯片都有SysTick，有助于操作系统或程序进行跨平台移植。

SysTick可使用内核时钟（HCLK），也可使用外部参考时钟（HCLK/8），其节拍周期可由“重装载值寄存器”设定。

SH33F2801给SysTick提供了外部参考时钟，固定为HCLK/8，其节拍标准值固定为105000，即在HCLK为84MHz频率下，产生10ms的时间基准。

10.2.8 HSI 修整功能（HSITRIM）

SH33F2801出厂时HSI已校准到一个很高的精度（常温0.2%精度）， 但在全温度范围，HSI精度会有所下降（上限1%），此时用户可以使用3bit修整控制位HSITRIM[2:0]对HSI进行进一步修整，以达到全温度范围提升精度的目标。

对于一种特殊的情况，即系统安装有外部晶振（精度一般在几十ppm），但运行时又使用HSI作为时钟源，此时可以借助外部晶振HSE去校准HSI，该功能由TRIMRUN@RCC_HSICAL开启，方法是使用HSI去计量4096分频后的HSE时钟，结果存放在TRIMREF[12:0]@RCC_HSICAL中，由用户根据该参数再去修整HSI，从而达到提升HSI精度、补偿电压和温度变化的目的。

如HSE使用8M晶振，则标准情况下TRIMREF将得到4095（表示4096个时钟）。

如果使用PLL，做HSI修整之前最好退出PLL，修整完成后再打开PLL。

10.3 寄存器

RCC 模块寄存器列表 (基地址：0x4004 0C00)

地址	寄存器名	说明
0x4004 0C00	CR	时钟控制寄存器
0x4004 0C04	CFGR	时钟配置寄存器
0x4004 0C08	CIENR	时钟中断使能寄存器
0x4004 0C0C	CISTR	时钟中断状态寄存器
0x4004 0C10	CICLR	时钟中断标志清除寄存器
0x4004 0C14	AHBRSTR	AHB外设复位寄存器
0x4004 0C1C	APB0RSTR	APB外设复位寄存器
0x4004 0C20	AHBENR	AHB外设时钟使能寄存器
0x4004 0C28	APB0ENR	APB外设时钟使能寄存器
0x4004 0C2C	RSTSTR	RESET状态寄存器
0x4004 0C30	RSTCLR	清除RESET状态寄存器
0x4004 0C34	HSICAL	HSI振荡器校准寄存器
0x4004 0C38	RCCLOCK	RCC配置锁定寄存器

10.3.1 时钟控制寄存器 (RCC_CR)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留							CSM ON	PLL RDY	PLL ON	HSE RDY	HSE ON	SWS[1:0]		SW[1:0]	
-							*RW*	*RO*	*RW*	*RO*	*RW*	*RO*		*RW*	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 9	保留	-
8	**CSMON**	**CSM时钟检测功能使能** 0：CSM时钟检测功能关闭 1：CSM时钟检测功能开启
7	**PLLRDY**	**PLL时钟就绪标志** PLL锁定后由硬件置‘1’ 0：PLL未锁定 1：PLL锁定

6	**PLLON**	**PLL使能 (PLL enable)** 由软件置'1'或清零以开启或关闭PLL 当PLL时钟被用作系统时钟时，该位不能被清零 0：PLL关闭 1：PLL使能
5	**HSERDY**	**外部高频振荡器就绪标志** 由硬件置1来指示外部高频振荡器已经稳定 在HSEON位清零后，该位需要6个外部4-12MHz时钟周期清零 0：外部4-12MHz时钟没有就绪 1：外部4-12MHz时钟就绪 说明：该位依赖于HSEON，仅指示外部高频振荡器就绪状态。
4	**HSEON**	**外部高频振荡器使能** 由软件置1或清0 当HSE被直接或间接用作系统时钟时，该位不能被清零 0：HSE振荡器关闭 1：HSE振荡器开启
3 ~ 2	**SWS[1:0]**	**系统时钟切换状态 (System clock switch status)** 由硬件置'1'或清'0'来指示系统时钟源。 00：HSI作为系统时钟 01：HSE作为系统时钟 1x：PLL输出作为系统时钟
1 ~ 0	**SW[1:0]**	**系统时钟切换 (System clock switch)** 由软件置'1'或清'0'来选择系统时钟源。 00：HSI作为系统时钟 01：HSE作为系统时钟 1x：PLL输出作为系统时钟

10.3.2 时钟配置寄存器 (RCC_CFGR)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留											HSE 16M	PLL XTPRE	PLL SRC	PLLF[5:0]	
-											RW	RW	RW	RW	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PLLF[5:0]				PLLK[2:0]			保留			PPRE[2:0]			HPRE[2:0]		
RW				RW			-			RW			RW		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 21	保留	-
20	**HSE16M**	**晶振增强驱动模式选择位** 0: 强驱动模式关闭 1: 强驱动模式开启
19	**PLLXTPRE**	**PLL时钟输入预分频 (Pre-divider for PLL entry)** 由软件置‘1’或清‘0’来分频后作为PLL输入时钟。 只能在关闭PLL时写入此位。 0：不预分频 1：2分频
18	**PLLSRC**	**PLL输入时钟源 (PLL entry clock source)** 由软件置‘1’或清‘0’来选择PLL输入时钟源。 只能在关闭PLL时才能写入此位。 0：HSI时钟作为PLL输入时钟 1：HSE时钟作为PLL输入时钟
17 ~ 12	**PLLF[5:0]**	**PLL倍频因子F (PLL multiplication factor F)** 由软件设置来确定PLL倍频参数 只有在PLL关闭的情况下才可被写入。 000000：F=15 000001：F=16 000010：F=17 ... ... 111101：F=76 111110：F=77 111111：F=78

11 ~ 9	**PLLK[2:0]**	**PLL倍频因子K (PLL multiplication factor K)** 由软件设置来确定PLL倍频参数 只有在PLL关闭的情况下才可被写入。 000：K=1 001：K=2 010：K=3 011：K=4 100：K=5 101：K=6 110：K=7 111：K=8
8 ~ 6	保留	-
5 ~ 3	**PPRE[2:0]**	**低速APB预分频(APB) (APB low-speed prescaler (APB))** 由软件置‘1’或清‘0’来控制低速APB时钟(PCLK)的预分频系数。 000：HCLK不分频 001：HCLK 2分频 010：HCLK 4分频 011：HCLK 8分频 100：HCLK 16分频 其他：HCLK 16分频 警告：软件必须保证APB时钟频率不超过42MHz。
2 ~ 0	**HPRE[2:0]**	**AHB预分频 (AHB Prescaler)** 由软件置‘1’或清‘0’来控制AHB时钟的预分频系数。 000：SYSCLK不分频 001：SYSCLK 2分频 010：SYSCLK 4分频 011：SYSCLK 8分频 100：SYSCLK 16分频 101：SYSCLK 32分频 其他：SYSCLK 32分频

10.3.3 时钟中断使能寄存器 (RCC_CIENR)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留											PLLRDYIE	HSERDYIE	保留		
-											*RW*	*RW*	-		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 5	保留	-
4	**PLLRDYIE**	**PLL就绪中断使能 (PLL ready interrupt enable)** 由软件置‘1’或清‘0’来使能或关闭PLL就绪中断。 0：PLL就绪中断关闭； 1：PLL就绪中断使能。
3	**HSERDYIE**	**HSE就绪中断使能 (HSE ready interrupt enable)** 由软件置‘1’或清‘0’来使能或关闭外部4-12MHz振荡器就绪中断。 0：HSE就绪中断关闭； 1：HSE就绪中断使能。
2 ~ 0	保留	-

10.3.4 时钟中断状态寄存器 (RCC_CISTR)

偏移地址: 0x000C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								CSMPLLF	CSMHSEF	保留	PLLRDYIF	HSERDYIF	保留		
-								*RO*	*RO*	-	*RO*	*RO*	-		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7	**CSMPLLF**	**PLL异常中断标志 (PLL clock security monitor interrupt flag)** PLL失锁时有效，由硬件置‘1’。 由软件通过置‘1’ CSMC位来清除。 0：无PLL异常导致的CSM中断； 1：有PLL异常导致的CSM中断；
6	**CSMHSEF**	**HSE异常中断标志 HSE clock security monitor interrupt flag)** HSE失效时有效，由硬件置‘1’。 由软件通过置‘1’ CSMC位来清除。 0：无HSE失效导致的CSM中断； 1：有HSE失效导致的CSM中断；
5	保留	-
4	**PLLRDYIF**	**PLL就绪中断标志 (PLL ready interrupt flag)** 由硬件置‘1’。 由软件通过置‘1’ PLLRDYC位来清除。 0：无PLL上锁产生的时钟就绪中断； 1：PLL上锁导致时钟就绪中断。
3	**HSERDYIF**	**HSE就绪中断标志 (HSE ready interrupt flag)** 由硬件置‘1’。 由软件通过置‘1’ HSERDYC位来清除。 0：无外部4-12MHz振荡器产生的时钟就绪中断； 1：外部4-12MHz振荡器导致时钟就绪中断。
2 ~ 0	保留	-

10.3.5 时钟中断标志清除寄存器 (RCC_CICLR)

偏移地址: 0x0010

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								CSMC	保留		PLLRDYC	HSERDYC	保留		
-								*WO*	-		*WO*	*WO*	-		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7	**CSMC**	**清除CSM中断 (Clock security monitor interrupt clear)** 由软件置'1'来清除CSM中断标志位CSMIF。 0：无作用； 1：清除CSMPLLF和CSMHSEF中断标志位。
6 ~ 5	保留	-
4	**PLLRDYC**	**清除PLL就绪中断 (PLL ready interrupt clear)** 由软件置'1'来清除PLL就绪中断标志位PLLRDYF。 0：无作用； 1：清除PLL就绪中断标志位PLLRDYF。
3	**HSERDYC**	**清除HSE就绪中断 (HSE ready interrupt clear)** 由软件置'1'来清除HSE就绪中断标志位HSERDYF。 0：无作用； 1：清除HSE就绪中断标志位HSERDYF。
2 ~ 0	保留	-

10.3.6 AHB 外设复位寄存器 (RCC_AHBRSTR)

偏移地址: 0x0014

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留											MACPRST	CRCRST	SYSCFGRST	MCMRST	GPIORST
-											*RW1s*	*RW1s*	*RW1s*	*RW1s*	*RW1s*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 5	保留	-
4	**MACPRST**	**数学协处理器复位 (Math co-processor reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：数学协处理器复位
3	**CRCRST**	**CRC模块复位 (CRC module reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：CRC模块复位
2	**SYSCFGRST**	**系统配置模块复位 (SYSCFG reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：系统配置模块复位 注：无法复位此模块内的one-time寄存器。
1	**MCMRST**	**MCM模块复位 (MCM module reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：MCM模块复位
0	**GPIORST**	**通用I/O端口复位 (GPIO block reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：通用I/O端口配置区复位

10.3.7 APB 外设复位寄存器 (RCC_APB0RSTR)

偏移地址: 0x001C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留														FIFO2RST	FIFO1RST
-														RW1s	RW1s
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
FIFO0RST	保留	CAN0RST	ADCRST	PCA1RST	PCA0RST	AMOCRST	WWDTRST	TWI0RST	SPI0RST	UART2RST	UART1RST	UART0RST	QEIRST	TIM8RST	TIM7RST
RW1s	-	RW1s	RW1s	RW1s	RW1s	RW1s	RW1s	RW1s	RW1s	RW1s	RW1s	RW1s	RW1s	RW1s	RW1s
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 18	保留	-
17	**FIFO2RST**	**FIFO2复位 (FIFO2 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：FIFO2复位
16	**FIFO1RST**	**FIFO1复位 (FIFO1 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：FIFO1复位
15	**FIFO0RST**	**FIFO0复位 (FIFO0 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：FIFO0复位
14	保留	-
13	**CAN0RST**	**CAN0复位 (CAN0 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：CAN0复位
12	**ADCRST**	**ADC复位 (ADC reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：ADC复位
11	**PCA1RST**	**PCA1复位 (PCA1 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：PCA1复位

10	**PCA0RST**	**PCA0复位 (PCA0 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：PCA0复位
9	**AMOCRST**	**片上模拟功能模块复位(Analog module on-chip reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：片上模拟功能模块复位
8	**WWDTRST**	**窗口看门狗复位(Window watchdog reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：窗口看门狗复位
7	**TWI0RST**	**TWI0复位 (TWI0 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：TWI0复位
6	**SPI0RST**	**SPI0复位 (SPI reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：SPI0复位
5	**UART2RST**	**UART2复位 (UART2 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：UART2复位
4	**UART1RST**	**UART1复位 (UART 1 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：UART1复位
3	**UART0RST**	**UART0复位 (UART 0 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：UART0复位
2	**QEIRST**	**QEI复位 (QEI reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：QEI复位
1	**TIM8RST**	**定时器8复位 (Timer 8 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：定时器8复位
0	**TIM7RST**	**定时器7复位 (Timer 7 reset)** 由软件置‘1’，硬件清‘0’ 0：无影响 1：定时器8复位

10.3.8 AHB 外设时钟使能寄存器 (RCC_AHBENR)

偏移地址: 0x0020

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留											MACPEN	CRCEN	SYSCFGEN	MCMEN	GPIOEN
-											*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 5	保留	-
4	**MACPEN**	**数学协处理器时钟使能 (Math co-processor clock enable)** 由软件置‘1’或清‘0’ 0：数学协处理器时钟关闭 1：数学协处理器时钟开启
3	**CRCEN**	**CRC模块时钟使能 (CRC clock enable)** 由软件置‘1’或清‘0’ 0：CRC模块时钟关闭 1：CRC模块时钟开启
2	**SYSCFGEN**	**系统配置模块时钟使能 (SYSCFG clock enable)** 由软件置‘1’或清‘0’ 0：系统配置模块时钟关闭 1：系统配置模块时钟开启 注：无法复位此模块内的one-time寄存器。
1	**MCMEN**	**MCM模块时钟使能 (MCM module clock enable)** 由软件置‘1’或清‘0’ 0：MCM模块时钟关闭 1：MCM模块时钟开启 注：MCM在时钟关闭时输出呈现高阻状态。
0	**GPIOEN**	**通用I/O端口时钟使能 (GPIO block clock enable)** 由软件置‘1’或清‘0’ 0：通用I/O端口时钟关闭 1：通用I/O端口时钟开启

10.3.9 APB 外设时钟使能寄存器 (RCC_APB0ENR)

偏移地址: 0x0028

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留														FIFO 2EN	FIFO 1EN
-														RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
FIFO 0EN	保留	CAN 0EN	ADC EN	PCA 1EN	PCA 0EN	AMO CEN	WW DTE N	TWI0 EN	SPI0 EN	UAR T2E N	UAR T1E N	UAR T0E N	QEIE N	TIM8 EN	TIM7 EN
RW	-	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 18	保留	-
17	**FIFO2EN**	**FIFO2时钟使能 (FIFO2 clock enable)** 由软件置‘1’或清‘0’ 0：FIFO2时钟关闭 1：FIFO2时钟开启
16	**FIFO1EN**	**FIFO1时钟使能 (FIFO1 clock enable)** 由软件置‘1’或清‘0’ 0：FIFO1时钟关闭 1：FIFO1时钟开启
15	**FIFO0EN**	**FIFO0时钟使能 (FIFO0 clock enable)** 由软件置‘1’或清‘0’ 0：FIFO0时钟关闭 1：FIFO0时钟开启
14	保留	-
13	**CAN0EN**	**CAN0时钟使能 (CAN0 clock enable)** 由软件置‘1’或清‘0’ 0：CAN0时钟关闭 1：CAN0时钟开启
12	**ADCEN**	**ADC时钟使能 (ADC clock enable)** 由软件置‘1’或清‘0’ 0：ADC时钟关闭 1：ADC时钟开启
11	**PCA1EN**	**PCA1时钟使能 (PCA1 clock enable)** 由软件置‘1’或清‘0’ 0：PCA1时钟关闭 1：PCA1时钟开启

10	**PCA0EN**	**PCA1时钟使能 (PCA0 clock enable)** 由软件置‘1’或清‘0’ 0：PCA0时钟关闭 1：PCA0时钟开启
9	**AMOCEN**	**片上模拟功能模块时钟使能 (Analog module on-chip clock enable)** 由软件置‘1’或清‘0’ 0：片上模拟功能模块时钟关闭 1：片上模拟功能模块时钟开启
8	**WWDTEN**	**窗口看门狗时钟使能 (Window watchdog clock enable)** 由软件置‘1’或清‘0’ 0：窗口看门狗时钟关闭 1：窗口看门狗时钟开启
7	**TWI0EN**	**TWI0时钟使能 (TWI0 clock enable)** 由软件置‘1’或清‘0’ 0：TWI0时钟关闭 1：TWI0时钟开启
6	**SPI0EN**	**SPI0时钟使能 (SPI0 clock enable)** 由软件置‘1’或清‘0’ 0：SPI0时钟关闭 1：SPI0时钟开启
5	**UART2EN**	**UART2时钟使能 (UART2 clock enable)** 由软件置‘1’或清‘0’ 0：UART2时钟关闭 1：UART2时钟开启
4	**UART1EN**	**UART1时钟使能 (UART1 clock enable)** 由软件置‘1’或清‘0’ 0：UART1时钟关闭 1：UART1时钟开启
3	**UART0EN**	**UART0时钟使能 (UART0 clock enable)** 由软件置‘1’或清‘0’ 0：UART0时钟关闭 1：UART0时钟开启
2	**QEIEN**	**QEI时钟使能 (QEI clock enable)** 由软件置‘1’或清‘0’ 0：QEI时钟关闭 1：QEI时钟开启
1	**TIM8EN**	**定时器8时钟使能 (Timer 8 clock enable)** 由软件置‘1’或清‘0’ 0：定时器8时钟关闭 1：定时器8时钟开启
0	**TIM7EN**	**定时器7时钟使能 (Timer 7 clock enable)** 由软件置‘1’或清‘0’ 0：定时器7时钟关闭 1：定时器7时钟开启

10.3.10 RESET 状态寄存器 (RCC_RSTSTR)

偏移地址: 0x002C

复位值: 0x0000 0004

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留										WWDTRSTF	IWDTRSTF	SWRSTF	PORSTF	LVRSTF	PINRSTF
-										*RO*	*RO*	*RO*	*RO*	*RO*	*RO*
0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0

位	符号	说明
31 ~ 6	保留	-
5	**WWDTRSTF**	**窗口看门狗复位标志 (Window watchdog reset flag)** 在窗口看门狗复位发生时由硬件置'1'；由软件通过写WWDTRSTFC位清除。 0：无窗口看门狗复位发生 1：发生窗口看门狗复位
4	**IWDTRSTF**	**独立看门狗复位标志 (Independent watchdog reset flag)** 在独立看门狗复位发生时由硬件置'1'；由软件通过写IWDTRSTFC位清除。 0：无独立看门狗复位发生 1：发生独立看门狗复位
3	**SWRSTF**	**软件复位标志 (Software reset flag)** 在软件复位发生时由硬件置'1'；由软件通过写SWRSTFC位清除。 0：无软件复位发生 1：发生软件复位
2	**PORSTF**	**上电复位标志 (POR reset flag)** 在上电复位发生时由硬件置'1'；由软件通过写PORSTFC位清除。 0：无上电复位发生 1：发生上电复位
1	**LVRSTF**	**低电压复位标志 (LVR reset flag)** 在低电压复位发生时由硬件置'1'；由软件通过写LVRSTFC位清除。 0：无低电压复位发生 1：发生低电压复位
0	**PINRSTF**	**NRST引脚复位标志 (PIN reset flag)** 在NRST引脚复位发生时由硬件置'1'；由软件通过写PINRSTFC位清除。 0：无NRST引脚复位发生 1：发生NRST引脚复位

10.3.11 清除 RESET 状态寄存器 (RCC_RSTCLR)

偏移地址: 0x0030

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留										WWDTRSTFC	IWDTRSTFC	SWRSTFC	PORSTFC	LVRSTFC	PINRSTFC
-										*WO*	*WO*	*WO*	*WO*	*WO*	*WO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 6	保留	-
5	**WWDTRSTFC**	**清除WWDT复位标志** 由软件置'1'来清除复位标志。 0：无作用 1：清除复位标志
4	**IWDTRSTFC**	**清除IWDT复位标志** 由软件置'1'来清除复位标志。 0：无作用 1：清除复位标志
3	**SWRSTFC**	**清除SOFTWARE复位标志** 由软件置'1'来清除复位标志。 0：无作用 1：清除复位标志
2	**PORSTFC**	**清除POR复位标志** 由软件置'1'来清除复位标志。 0：无作用 1：清除复位标志
1	**LVRSTFC**	**清除LVR复位标志** 由软件置'1'来清除复位标志。 0：无作用 1：清除复位标志
0	**PINRSTFC**	**清除PIN复位标志** 由软件置'1'来清除复位标志。 0：无作用 1：清除复位标志

10.3.12 HSI 振荡器校准寄存器 (RCC_HSICAL)

偏移地址: 0x0034

复位值: 0xXXXX XXXX

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
TRIMRUN	保留		TRIMREF[12:0]												
RW	-		*RO*												
0	0	0	X	X	X	X	X	X	X	X	X	X	X	X	X

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
HSICAL[7:0]								保留					HSITRIM[2:0]		
RO								-					*RW*		
X	X	X	X	X	X	X	X	X	X	X	X	X	0	0	0

位	符号	说明
31	**TRIMRUN**	**HSI用户调整开关，只能软件置1，硬件或软件清0** 0：运算已结束或未启动 1：启动HSI ADJ，运算过程中标志为1,运算完成后标志由硬件清0。 注：允许运算过程中清零RUN以强行结束运算。
30 ~ 29	保留	-
28 ~ 16	**TRIMREF[12:0]**	**HSI用户校准参数值** 在RUN运行完成后，用户可从该寄存器读取HSI用户校准参数值 （HSI计数HSE分频源）
15 ~ 8	**HSICAL[7:0]**	**内部高频RC振荡器校准** 在系统启动时，这些位被自动初始化
7 ~ 3	保留	-
2 ~ 0	**HSITRIM[2:0]**	**内部高频RC振荡器调整** 由软件写入来调整HSI频率，它们被叠加在HSICAL[7:0]数值上 这些位在HSICAL[7:0]的基础上，让用户可以输入一个调整数值，根据电压和温度的变化调整内部HSI RC振荡器的频率 000： +0 （default） 001： +0.25% 010：+0.5% 011：+0.75% 100：-1% 101：-0.75% 110：-0.5% 111：-0.25%

10.3.13 RCC 配置锁定寄存器 (RCC_RCCLOCK)

偏移地址: 0x0038

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
LOCK[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**LOCK[15:0]**	**RCC 配置操作解锁位 (RCC register lock bit)** 0x33CC： 解锁 其他：锁定

10.3.14 SysTick 控制及状态寄存器（SysTick_CTRL）

地址: 0xE000 E010

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															CTFLAG
-															R
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留													CLKSOURCE	TICKINT	ENABLE
-													RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 17	保留	-
16	CTFLAG	计数标志位 如果在上次读取本寄存器后， SysTick 已经计到了 0，则该位为 1。如果读取该位，该位将自动清零。
15 ~ 3	保留	-
2	CLKSOURCE	SysTick时钟源选择位 0：外部时钟源（HCLK/8） 1：内核时钟（HCLK）
1	TICKINT	SysTick异常请求使能 0：SysTick倒数计数到0时无动作 1：SysTick倒数计数到0时产生 SysTick异常请求
0	ENABLE	SysTick 定时器的使能位 0：关闭 1：开启

10.3.15 SysTick 重装载数值寄存器（SysTick_LOAD）

地址: 0xE000 E014
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留								RELOAD[23:16]							
-								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RELOAD[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 24	保留	-
23 ~ 0	**RELOAD[23:0]**	**SysTick重装载数值** 当倒数计数至零时，将被重装载的值。

10.3.16 SysTick 当前数值寄存器（SysTick_VAL）

地址: 0xE000 E018
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留								CURRENT[23:16]							
-								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CURRENT[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 24	保留	-
23 ~ 0	**CURRENT[23:0]**	**SysTick当前数值** 读取时返回当前倒计数的值，写它则使之清零，同时还会清除在SysTick控制及状态寄存器中的COUNTFLAG 标志

10.3.17 SysTick 校准数值寄存器（SysTick_CALIB）

地址: 0xE000 E01C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
NOREF	SKEW	保留						TENMS[23:16]							
R	*R*	-						*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TENMS[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31	**NOREF**	**SysTick时钟源状态位** 1：没有外部参考时钟（HCLK 不可用） 0：外部参考时钟可用
30	**SKEW**	**SysTick节拍标准值状态位** 1：校准值不是准确的10ms 0：校准值是准确的10ms
29 ~ 24	保留	**-**
23 ~ 0	**TENMS[23:0]**	**SysTick节拍标准值** 10ms 间隔的节拍标准值。

11. 看门狗定时器（WDT）

11.1 独立看门狗定时器（IWDT）

独立看门狗定时器是一个12位递减计数器，独立内建RC振荡器作为其时钟源，上电默认硬件关闭，但可以在程序中通过控制位IWDTON@IWDT_CR软件开启。 IWDT也可以通过用户代码选项“OP_IWDT”设置为上电硬件开启。

软件开启方式下，IWDTON@IWDT_CR一旦置位后只能在复位时清除。

IWDT可以工作在停机模式下，因此可以作为唤醒定时器，通过代码选项OP_WDTPD可选择停机时开启或关闭看门狗（默认停机模式下开启）。

在定时器溢出之前，通过对喂狗寄存器IWDT_CLR写入0xAAAA更新计数器以重新开始计数，避免产生溢出。

定时器溢出后将复位芯片，并置位IWDTRSTF@RCC_RSTSTR标志。

独立看门狗工作时钟是LSI，默认溢出时间为4096*32/128 = 1024ms，当然由于LSI的不精确，此溢出时间也是不精确的。

在调试时，通过“DBG_IWDT”可以控制调试时看门狗是否停止工作。

11.2 窗口看门狗定时器（WWDT）

窗口看门狗的喂狗时间是一个时间窗口，用来检测外部干扰或不可预测逻辑条件造成的应用程序背离正常运行序列而产生的软件故障，适合在精确计时窗口起作用的应用程序。 当过早或过迟刷新窗口看门狗时，将产生复位信号。

窗口看门狗是一个8位递减计数器，喂狗区间为一个窗口区，早于窗口喂狗区喂狗将产生超前异常事件，晚于窗口喂狗区将产生延迟异常事件，两者都会引起复位。另外，在计数器到达窗口喂狗区下边界0时为最迟喂狗时刻，可产生WWDT中断，进入WWDT中断后，需要在1个WWDT时钟内喂狗，否则在计数器溢出时产生复位。

窗口看门狗在WWDT_CR中开启，一旦开启，只有在复位时才能关闭。

窗口看门狗工作时钟是PCLK。

图11-1　　窗口看门狗示意图

应用WWDT时，先设置WWDT的窗口重载值WWDTRLR（图中标注RL），窗口值WWDTWTR（图中标注WT），然后开启WWDT，计数器开始运行（计数器初始为重载值，递减计数）。在正常喂狗区通过对喂狗寄存器WWDTCLR写入0x5555进行喂狗。

如设置RL>WT则可实现正常窗口监控功能，可监测超前异常和延迟异常。

如设置RL<=WT，则没有超前异常监控功能，只有延迟异常监控功能，功能类似普通看门狗。

RL=WT=0xFF则窗口最大。

当微控制器进入调试模式时(ARM “星辰”处理器停止)，根据调试模块中的DBG_WWDT 配置位的状态，WWDT的计数器能够继续工作或停止。详见“调试接口”章节。

11.3 IWDT 寄存器

IWDT 模块寄存器列表 (基地址：0x4000 2C00)

地址	寄存器名	说明
0x4000 2C00	CR	独立看门狗控制寄存器
0x4000 2C04	CLR	独立看门狗喂狗寄存器

11.3.1 独立看门狗控制寄存器 (IWDT_CR)

偏移地址: 0x0000
复位值: 0x0000 7FFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
LOCK[15:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
IWDT ON	**IWDTPR[2:0]**			**IWDTRLR[11:0]**											
RW1t	*RW*			*RW*											
0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位	符号	说明
31 ~ 16	**LOCK[15:0]**	**寄存器锁定位** 0x5AA5：解锁此寄存器 其他：锁定此寄存器 配置此寄存器时解锁位必须同时输入0x5AA5，操作完成后自动锁定，读取此LOCK位固定读出0x0。
15	**IWDTON**	**独立看门狗使能，此位由软件置‘1’，但仅能由硬件在复位后清‘0’。** 此控制位只在OP_IWDT=0101b时有效。 0：无意义 1：启用IWDT
14 ~ 12	**IWDTPR[2:0]**	**独立看门狗时钟预分频设置** 000: 预分频因子=4 001: 预分频因子=8 010: 预分频因子=16 011: 预分频因子=32 （默认） 100: 预分频因子=64 101: 预分频因子=128 110: 预分频因子=256 111: 预分频因子=512
11 ~ 0	**IWDTRLR[11:0]**	**独立看门狗重载寄存器** 用于定义看门狗计数器的重装载值。 注：IWDTRLR[11:0]复位值为0x0FFF。

11.3.2 独立看门狗喂狗寄存器 (IWDT_CLR)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
IWDTCLR[15:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**IWDTCLR[15:0]**	独立看门狗喂狗寄存器 0xAAAA：喂狗（用IWDTRLR更新看门狗计数器） 其他：无意义 喂狗完成后读此寄存器固定返回0x0。

11.4 WWDT 寄存器

WWDT 模块寄存器列表 (基地址：0x4000 3000)

地址	寄存器名	说明
0x4000 3000	CR	窗口看门狗控制寄存器
0x4000 3004	SR	窗口看门狗状态寄存器
0x4000 3008	CLR	窗口看门狗喂狗寄存器
0x4000 300C	WTR	窗口看门狗窗口寄存器

11.4.1 窗口看门狗控制寄存器 (WWDT_CR)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
LOCK[15:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
WWDTON	WWDTIE	保留			WWDTPR[2:0]			WWDTRLR[7:0]							
RW1t	*RW*	-			*RW*			*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**LOCK[15:0]**	寄存器锁定位 0x5AA5：解锁此寄存器 其他：锁定此寄存器 配置此寄存器时解锁位必须同时输入0x5AA5，操作完成后自动锁定，读取此LOCK位固定读出0x0000。

15	**WWDTON**	**窗口看门狗使能，此位由软件置'1'，但仅能由硬件在复位后清'0'。** 0：禁止WWDT 1：启用WWDT
14	**WWDTIE**	**窗口看门狗最迟喂狗中断使能，此位由软件置'1'和清'0'。** 0：禁止WWDT中断 1：启用WWDT中断
13 ~ 11	保留	**-**
10 ~ 8	**WWDTPR[2:0]**	**窗口看门狗时钟预分频设置** 000: 预分频因子=1X256 001: 预分频因子=2X256 010: 预分频因子=4X256 011: 预分频因子=8X256 100: 预分频因子=16X256 101: 预分频因子=32X256 110: 预分频因子=64X256 111: 预分频因子=128X256
7 ~ 0	**WWDTRLR[7:0]**	**窗口看门狗重载寄存器** 用于定义看门狗计数器的重装载值。

11.4.2 窗口看门狗状态寄存器 (WWDT_SR)

偏移地址: 0x0004

复位值: 0x0000 00FF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
WWDTIF	保留							**TCNT[7:0]**							
RW	-							*RO*							
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

位	符号	说明
31 ~ 16	保留	**-**
15	**WWDTIF**	**窗口看门狗最迟喂狗中断标志位，此位由软件或硬件置'1'，软件清'0'。** 0：未产生WWDT中断 1：产生了WWDT中断
14 ~ 8	保留	**-**
7 ~ 0	**TCNT[7:0]**	**窗口看门狗计数寄存器** 指示当前看门狗计数值，启动时默认为最大值0xFF。

11.4.3 窗口看门狗喂狗寄存器 (WWDT_CLR)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
WWDTCLR[15:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**WWDTCLR[15:0]**	窗口看门狗喂狗寄存器 0x5555：喂狗（用WWDTRLR更新看门狗计数器） 其他：无意义 喂狗完成后读此寄存器固定返回0x0。

11.4.4 窗口看门狗窗口寄存器 (WWDT_WTR)

偏移地址: 0x000C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
LOCK[15:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								WWDTWTR[7:0]							
-								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**LOCK[15:0]**	寄存器锁定位 0x5AA5：解锁此寄存器 其他：锁定此寄存器 配置此寄存器时解锁位必须同时输入0x5AA5，操作完成后自动锁定，读取此LOCK位固定读出0x0000。
15 ~ 8	保留	-
7 ~ 0	**WWDTWTR[7:0]**	窗口看门狗窗口值寄存器 用于定义看门狗的窗口值。 该寄存器具有写保护，只有LOCK位为0x5AA5时才能写入，读出不受限制。

12. 代码和数据检测（Code and Data Checking）

12.1 CRC 模块（CRC）

12.1.1 CRC 简介

循环冗余校验(CRC)计算单元可根据生成多项式得到任一32位/16位/8位的CRC计算结果。CRC技术主要应用于核实数据传输或者数据存储的正确性和完整性。标准EN/IEC 60335-1需要提供一种核实闪存存储器完整性的方法，利用 CRC计算单元可以在程序运行时计算出软件的标识，与在连接时生成的参考标识比较，检查闪存存储器完整性。在通讯过程中，可将CRC值添加到数据末尾进行传送，接收端将所接收到的数据逐个送入CRC单元，最后判断CRC值是否为0来验证传输数据的完整性。

12.1.2 CRC 主要特性

- 支持多种CRC格式

  （1）CRC-32：(0x04C11DB7)

  生成多项式：$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$

  （2） CRC-16： (0x8005)

  生成多项式：$x^{16}+x^{15}+x^{2}+1$

  （3）CRC-CCITT：(0x1021)

  生成多项式：$x^{16}+x^{12}+x^{5}+1$

  （4）CRC-8: (0x7)：

  生成多项式：$x^{8}+x^{2}+x+1$
- 支持修改初始值 CRC_INIT. 当计算CRC-16时，生成多项式的低16位有效，当计算CRC-8时，生成多项式的低8位有效。
- 可配置输入/输出数据转换，包括字节逆向，字节内位逆向，及1’s complement （ 按位取反)。
- 支持8/16/32位数据输入。输入8位则一次只计算8位数据，输入16位则一次只计算16位数据，输入32位则一次计算32位数据。此输入格式与CRC格式无关。计算数据的基本单元为8位数据，因此当输入多字节数据时是低字节先运算。
- 计算CRC-8需要1个时钟，CRC-16需要2个时钟，CRC-32需要4个时钟

12.1.3 CRC 功能描述

如图12.1所示。计算CRC时首先根据实际情况对CRC_CR进行格式设置。然后设置格式所需的生成多项式及CRC初始值。设置完毕后需要往RELOAD@CRC_CR写1，将初始值载入CRC_DR单元。之后就可以往CRC_DR中写数据然后从CRC_DR中读取计算结果。

数据写入时支持8位，16位，32位写入，但要求写入/读取时必须按字对齐（4字节对齐,即必须是CRC_DR的首地址），否则结果不可预测。每次输入都会得到一个CRC结果。CRC的计算顺序是低位在先。所以当输入转换设置为全0时（即不对写入数据作任何变换），一次写入0x12345678与连续写入0x78，0x56，0x34，0x12得到的CRC结果是一样的。

图12-1　　CRC 逻辑框图

12.1.3.2 CRC 格式转换描述

由于对寄存器操作时可以是8位，16位或32位。所以在对数据预处理时会根据输入数据的差异进行不同的处理。读取时只与CRC Mode有关。

表12-1 数据格式转换描述

	8位写入配置位			**读取配置位**		
MODE	COMPLW (Byte)	RBITW (bit)	RBYTEW (Byte)	COMPLR (Byte)	RBITR (bit)	RBYTER (Byte)
CRC_8	B0 = ~B0	b[0..7]=b[7..0]	忽略	B0 = ~B0	b[0..7]=b[7..0]	忽略
CRC_16	B0 = ~B0	b[0..7]=b[7..0]	忽略	B0 = ~B0 B1=~B1	b[0..7]=b[7..0] b[8..15]=b[15..8]	B0=B1' B1=B0'
CRC_32	B0 = ~B0	b[0..7]=b[7..0]	忽略	B0 = ~B0 B1 = ~B1 B2 = ~B2 B3 = ~B3	b[0..7]=b[7..0] b[8..15]=b[15..8] b[16..23]=b[23..16] b[24..31]=b[31..24]	B0=B3' B3=B0' B1=B2' B2=B1'
	16位写入配置位			**读取配置位**		
MODE	COMPLW	RBITW	RBYTEW	COMPLR	RBITR	RBYTER
CRC_8	B0 = ~B0 B1 = ~B1	b[0..7]=b[7..0] b[8..15]=b[15..8]	B0=B1' B1=B0'	B0 = ~B0	b[0..7]=b[7..0]	忽略
CRC_16	B0 = ~B0 B1 = ~B1	b[0..7]=b[7..0] b[8..15]=b[15..8]	B0=B1' B1=B0'	B0 = ~B0 B1=~B1	b[0..7]=b[7..0] b[8..15]=b[15..8]	B0=B1' B1=B0'
CRC_32	B0 = ~B0 B1 = ~B1	b[0..7]=b[7..0] b[8..15]=b[15..8]	B0=B1' B1=B0'	B0 = ~B0 B1 = ~B1 B2 = ~B2 B3 = ~B3	b[0..7]=b[7..0] b[8..15]=b[15..8] b[16..23]=b[23..16] b[24..31]=b[31..24]	B0=B3' B3=B0' B1=B2' B2=B1'
	32位写入配置位			**读取配置位**		
MODE	COMPLW	RBITW	RBYTEW	COMPLR	RBITR	RBYTER
CRC_8	B0 = ~B0 B1 = ~B1 B2 = ~B2 B3 = ~B3	b[0..7]=b[7..0] b[8..15]=b[15..8] b[16..23]=b[23..16] b[24..31]=b[31..24]	B0=B3' B3=B0' B1=B2' B2=B1'	B0 = ~B0	b[0..7]=b[7..0]	忽略
CRC_16	B0 = ~B0 B1 = ~B1 B2 = ~B2 B3 = ~B3	b[0..7]=b[7..0] b[8..15]=b[15..8] b[16..23]=b[23..16] b[24..31]=b[31..24]	B0=B3' B3=B0' B1=B2' B2=B1'	B0 = ~B0 B1=~B1	b[0..7]=b[7..0] b[8..15]=b[15..8]	B0=B1' B1=B0'
CRC_32	B0 = ~B0 B1 = ~B1 B2 = ~B2 B3 = ~B3	b[0..7]=b[7..0] b[8..15]=b[15..8] b[16..23]=b[23..16] b[24..31]=b[31..24]	B0=B3' B3=B0' B1=B2' B2=B1'	B0 = ~B0 B1 = ~B1 B2 = ~B2 B3 = ~B3	b[0..7]=b[7..0] b[8..15]=b[15..8] b[16..23]=b[23..16] b[24..31]=b[31..24]	B0=B3' B3=B0' B1=B2' B2=B1'

12.1.3.3 常见格式的配置

（1）CRC32的设置 (0x04C11DB7)

$$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^{8}+x^{7}+x^{5}+x^{4}+x^{2}+x+1$$

CRC_INIT : 0xFFFFFFFF
CRC_CR.MODE : 0
CRC_CR.RBITW : 0
CRC_CR.RBYTEW : 0
CRC_CR.COMPLW : 0
CRC_CR.RBITR : 0
CRC_CR.RBYTER : 0
CRC_CR.COMPLR : 0

输入:
8位: 一次计算1个字节
16位: 一次计算2个字节，低字节先算
32位: 一次计算4个字节，低字节先算
CRC结果：CRC_DR的全部32-bit

（2）CRC16 的设置 (0x8005)

$$x^{16}+x^{15}+x^{2}+1$$

CRC_INIT : 0xFFFF
CRC_CR.MODE : 1
CRC_CR.RBITW : 0
CRC_CR.RBYTEW : 0
CRC_CR.COMPLW : 0
CRC_CR.RBITR : 0
CRC_CR.RBYTER : 0
CRC_CR.COMPLR : 0
输入:
8位: 一次计算1个字节
16位: 一次计算2个字节，低字节先算
32位: 一次计算4个字节，低字节先算
CRC结果：CRC_DR的低16位

（3）CRC16-CITT 的设置 (0x1021)

$$x^{16}+x^{12}+x^{5}+1$$

CRC_INIT : 0xFFFF
CRC_CR.MODE : 2
CRC_CR.RBITW : 0
CRC_CR.RBYTEW : 0
CRC_CR.COMPLW : 0
CRC_CR.RBITR : 0
CRC_CR.RBYTER : 0
CRC_CR.COMPLR : 0
输入:
8位: 一次计算1个字节
16位: 一次计算2个字节，低字节先算
32位: 一次计算4个字节，低字节先算
CRC结果：CRC_DR的低16位

（4）CRC8的设置 (0x7)

$$x^{8}+x^{2}+x+1$$

CRC_INIT : 0xFF
CRC_CR.MODE : 3
CRC_CR.RBITW : 0
CRC_CR.RBYTEW : 0
CRC_CR.COMPLW : 0
CRC_CR.RBITR : 0
CRC_CR.RBYTER : 0
CRC_CR.COMPLR : 0
输入:
8位: 一次计算1个字节
16位: 一次计算2个字节，低字节先算
32位: 一次计算4个字节，低字节先算
CRC结果：CRC_DR的低高8位

12.1.4 CRC 寄存器

CRC 模块寄存器列表 (基地址：0x4004 1800)

地址	寄存器名	说明
0x4004 1800	DR	CRC数据寄存器
0x4004 1804	CR	CRC控制寄存器
0x4004 1808	INIT	CRC初始值寄存器

12.1.4.1 CRC 数据寄存器 (CRC_DR)

偏移地址: 0x0000
复位值: 0xFFFF FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
DR[31:0]															
RW															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DR[31:0]															
RW															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位	符号	说明
31 ~ 0	**DR[31:0]**	**CRC数据** 输入时用作写一个新的数据到CRC计算单元，并启动计算.读取时返回上一次CRC计算的结果。允许一次写入8位，16位，32位。

12.1.4.2 CRC 控制寄存器 (CRC_CR)

偏移地址: 0x0004
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RBYTER	RBITR	COMPLR	RBYTEW	RBITW	COMPLW	MODE[1:0]		保留							RELOAD
RW	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*		-							*WO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	**-**
15	**RBYTER**	**对DR中的数据进行字节顺序逆向后再读取** 0: 保持不变 1: 进行字节顺序逆向 (具体变换参考表12-1)
14	**RBITR**	**对DR中的数据进行字节内位顺序逆向后再读取** 0: 保持不变 1: 进行字节内位顺序逆向(具体变换参考表12-1)
13	**COMPLR**	**对DR中的数据进行1's complement变换后再读取.(即对的数据进行按位取反)** 0:保持不变 1:进行1's complement变换

12	**RBYTEW**	**对写入的数据进行字节顺序逆向** 0: 保持不变 1: 进行字节顺序逆向 (具体变换参考表12-1)
11	**RBITW**	**对写入的数据进行字节内位顺序逆向** 0: 保持不变 1: 进行字节内位顺序逆向(具体变换参考表12-1)
10	**COMPLW**	**对写入的数据进行1's complement变换。(即对写入的数据进行按位取反)** 0:保持不变 1:进行1's complement变换
9 ~ 8	**MODE[1:0]**	**CRC算法模式选择，因为用逆向算法，所以生成多项式的实际值是按位逆向的。** 00: CRC-32 (0x04C11DB7) – 选table crc32_04C11DB7 01: CRC-16 (生成多项式为0x8005 -- modbus) – 选table crc16_8005 10: CRC-CITT (生成多项式为0x1021) – 选table crc16_1021 11: CRC-8 (生成多项式为0x7) – 选table crc8_7
7 ~ 1	保留	-
0	**RELOAD**	**将CRC的initial值重新载入到运算寄存器** 1: 写1重新载入。将INIT@CRC_INIT重新载入DR@CRC_DR寄存器，其他寄存器值保持不变。 0: 写0不变，此Bit无法读取，始终为 0。

12.1.4.3 CRC 初始值寄存器 (CRC_INIT)

偏移地址: 0x0008

复位值: 0xFFFF FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
INIT[31:0]															
RW															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
INIT[31:0]															
RW															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位	符号	说明
31 ~ 0	**INIT[31:0]**	**存放CRC初始值** 往RELOAD@CRC_CR写1后，此值会自动载入到DR@CRC_DR寄存器中。 CR_MODE为0，32位有效 CR_MODE为1，低16位有效 CR_MODE为2，低16位有效 CR_MODE为3，低8位有效

13. 通用功能 I/O (GPIO)

13.1 简介

SH33F2801提供最多45个通用I/O口，分成3组端口（A～C），每组最多包含16个输入输出引脚（PA0~PA15，PB0～PB15，PC0～PC12）。每组I/O端口都有独立的配置寄存器和数据寄存器。

每一个I/O口都有复用功能，或者与片上数字外设复用，或者与片上模拟外设复用，或者与外部中断与调试接口复用。

所有I/O都支持弱上拉和推挽输出，个别I/O支持开漏输出。

13.2 主要特性

- 所有端口组都挂载在AHB总线上，支持快速输入输出
- 输入/输出方向控制
- 输入管脚可配置弱上拉（上拉和方向不绑定，即输出时也能配置上拉）
- 输出管脚默认为推挽输出，个别引脚支持开漏输出
- 端口寄存器支持按字和字节操作，提供按位操作寄存器
- 部分端口可用作外部中断，触发沿可设置（需结合EXTI 配置寄存器）
- I/O口可配置为普通输入输出功能/数字外设功能/模拟外设功能
- I/O口配置使用多路选择器，同一时刻只能配置为一种功能
- 支持端口配置锁定功能

图13-1　　I/O 端口基本结构框图

13.3 功能描述

每个I/O模式配置是通过4个基本配置寄存器（MODER、OTYPER、PUR、AFR）完成的，每个寄存器都支持以32位字、16位半字、8位字节的方式读写，用于配置I/O的输入输出模式（方向）、输出推挽/开漏方式（仅部分引脚有开漏）、复用功能选择、端口上拉属性。

I/O端口的基本工作模式如下表所示：

表13-1　　I/O 端口基本工作模式一览

端口功能	AFRL/AFRH	MODER	OTYPE	I/O属性	
全关	AF0 (default)	-	-	输入输出全关	**注1**
GPIO	AF1	0	-	GPIO输入	
		1	0	GPIO推挽输出	
			1	GPIO开漏输出	
数字外设	AF2-AF6	-	0	数字类推挽输出外设	注2
			1	数字类开漏输出外设	
模拟外设	AF6/7	-	-	模拟外设	注3
系统接口	-	-	-	SWD/XTAL	注4

表中 **-** 表示不影响或无意义。

注1：上电后，GPIO默认是输入输出全关状态。

注2：数字类外设的输入输出配置是与外设绑定的，不需要用户配置。但OTYPE和PUPDR需要用户配置。

注3：模拟类外设MODER、OTYPE这两个寄存器没有意义。而PUPDR是允许的，以满足对模拟端口提供弱上拉功能的特殊应用。

注4：系统接口（system port）包含调试接口SWD和外部振荡器接口XTAL两类端口。系统接口也可以复用为GPIO或外设接口。但在作为系统接口使用时上述基本配置寄存器无效，这两种接口在SYSCFG中有进一步说明。

所有端口默认都是推挽方式，仅TWI端口支持推挽/开漏选择，见下表。对于不支持开漏的端口，OTYPE寄存器对应的开漏选择位无效，固定为推挽方式。

表13-2　　支持开漏模式的端口

I/O口	**复用功能**
PA0	SCL
PA2	SDA
PA13	(SDA)
PA14	(SCL)

13.3.1　I/O 引脚的默认配置

在上电复位期间和复位完成后，I/O口处于全关状态，但SWD接口处于开启状态。

SWD的引脚属性是系统设定的，系统复位后，对应的SWD引脚状态设置如下：

- SWDIO：输入并上拉
- SWCLK：输入并下拉

一旦SWD接口被用户代码释放（见控制位SWJCFG@SYSCFG_SAFR），软件可以把这些I/O口作为普通的I/O口使用。此时SWDIO和SWCLK的上下拉自动失效，用户可根据要求配置为上拉或悬空。

与SWD接口不同，外部振荡器接口XTAL1/XTAL2默认是GPIO接口，需在系统启动后由软件开启XTAL接口属性（见控制位OSCCFG@SYSCFG_SAFR），在作为XTAL接口时，端口的上拉功能仍然受寄存器控制，应用时需注意，不要误打开。

13.3.2 I/O 控制寄存器

每个I/O口都有4个寄存器用于配置工作模式：MODER，OTYPER，PUPDR，AFR。

MODER：设置输入输出方向；

OTYPER：设置输出类型，分别为推挽输出、开漏输出(仅个别引脚)；

PUPDR：设置内部弱上拉，在输入和输出时都能设置上拉；

AFR：设置复用功能，共8个选项AF0～AF7，具体见“复用功能映射表”，系统接口不属于AFR设置。

13.3.3 I/O 数据寄存器

每个GPIO口都有2个16位数据寄存器：IDR，ODR。

IDR：输入数据寄存器，对输入数据寄存器的读访问可得到I/O状态。

ODR：输出数据寄存器，写到输出数据寄存器上的值(ODR)输出到相应的I/O引脚。可以以推挽模式或开漏模式使用输出驱动器。

13.3.4 I/O 位置位和位清除

位置位和清除寄存器（BSRR）是1个32位寄存器，允许对输出寄存器（ODR）的每个位进行独立的置位和清除。

BSRR的低16位是置位操作，高16位是清除操作，允许在单次AHB写操作内对同一组端口的不同I/O口进行同时置位和清除操作，如果是对同一个I/O口同时置位和清除，置位操作具有高优先级。

BSRR的读写可以是字、半字或字节，下面的例子说明了如何用字访问来同时同向/反向切换两个I/O口：

；同时同向切换PA.0和PA.1
GPIOA_BSRR = 0x00000003
GPIOA_BSRR = 0x00030000
；同时反向切换PA.0和PA.1
GPIOA_BSRR = 0x00020001
GPIOA_BSRR = 0x00010002

如果用半字的方式来实现同时同向切换，方法如下：
#define GPIOA_BitSet_ADDR = 0xXX ；指向GPIO_BSRR_A的基地址（用于set）
#define GPIOA_BitReset_ADDR = 0xXX + 2 ；指向GPIO_BSRR_A的高半字偏移地址（用于reset）
**(u16*)GPIOA_BitSet_ADDR = 0x03*
**(u16*)GPIOA_BitReset_ADDR = 0x03*

BSRR对ODR的位操作是单次（one-shot）的，不会锁住ODR的相应位，用户仍然可直接对ODR寄存器进行操作。

13.3.5 I/O 锁定机制

锁定机制允许冻结IO配置。可锁定的寄存器包括：OTYPER，PUPDR，AFR。

每一个锁定位LCKy用于锁定一个I/O口，写入锁定位还需要通过向对应的解锁位（LOCK）写入特殊控制字才能写入。

13.3.6 I/O 配置为外部中断/唤醒线

部分端口可以配置为外部中断输入口，注意该功能并不通过AFR寄存器映射，而是通过外部中断配置寄存器（EXTI_CFG，在EXTI模块中）来配置。

注：SH33F2801的外部中断输入口是固定的，为确定的8个端口。

I/O口选择映射为外部中断输入口时，同外设复用类似，输入输出(MODER)不需要用户配置，但如果要用到输入上拉仍然要用户配置。

EXTI功能虽然不是通过AFR映射，但其应用方式和数字外设类似，只要I/O口的输入通道开启（并不局限于AF1，AF2/AF3选项下只要输入通道开启，有关哪些情况输入通道开启见后文GPIO功能描述），且EXTI_CFG对应端口打开，都是可以触发外部中断的。

注：I/O口配置为AF0时EXTI通道是关闭的。

更多的关于外部中断的信息，参考“外部中断/事件控制器(EXTI)”章节。

13.3.7 TTL 输入选择

部分引脚兼容TTL电平输入， SH33F2801定义了10个这种引脚，由TTL功能开关GPIOx_TTLEN寄存器对应开启。

表13-3 支持 TTL 输入的端口

I/O口	复用功能
PA0	RXD2/SCL
PA2	TXD2/SDA
PA6	RXD2/PCA0B
PA9	TXD2/PCA0A
PA11	RXD0/PCA0C
PA12	TXD0/PCA1A
PA13	RXD1/PCA1B
PA14	TXD1/PCA1C
PC0	TXD1
PC1	RXD1

13.3.8 I/O 复用功能

寄存器AFR定义了端口的复用功能，允许同一端口作为多个复用功能的共享输入输出口，每个端口都有一个8输入的I/O多路复用选择器用于选择需要的复用功能，同一时刻每个端口只允许一个外设连接。

8个输入通道（AF0～AF7）分成3类：

- AF0：输入输出全关（默认）；
- AF1：GPIO基本功能；
- AF2-AF6：数字外设复用功能；
- AF6/7：模拟外设复用功能；

输入输出全关复用通道：AF0

这是I/O口上电和复位后的默认配置，此通道是一种特殊状态，强制把PORT口的输入输出通道全部关闭，但允许上拉。

GPIO基本通道：AF1

所有I/O口都配置为输入输出功能状态（GPIO基本功能），如未开启上拉，则默认是输入浮空状态，除了SWD接口（有上拉和下拉）。

SWD接口引脚只有通过SWJCFG@SYSCFG_SAFR关闭SWD功能后才能作为GPIO功能（AF1）使用。

数字外设复用通道：AF2-AF6

数字外设复用功能由硬件自动设置输入输出方向，但输出类型(OTYPE)、上拉(PUPDR)需用户软件配置。

请参考 “复用功能映射表”，以获取详细的引脚复用映射。

说明：对于某些外设要求的双向端口，可由硬件自动切换。外设会保证不受发送时反馈到接收通道的数据的影响（假如开启接收通道的话）。

模拟外设复用通道：AF6/7

此通道用于片上模拟外设，输入输出方向（MODER）和输出类型（OTYPE）配置无意义，但上拉（PUR）在某些应用中有特殊用途。

模拟通道会关闭输出缓冲器及输入施密特触发器，相当于断开了数字输入输出通路，可以避免漏电。

13.3.9 GPIO 输入功能 （AF1）

当I/O端口配置为GPIO输入时：

- 输出通道关闭
- 输入通道开启
- 根据上拉配置的不同，上拉电阻被连接或断开
- 对输入数据寄存器的读访问可得到I/O状态

下图给出了I/O端口位的GPIO输入配置

图13-2　　I/O 端口位的 GPIO 输入配置框图

13.3.10 GPIO 输出功能 （AF1）

当I/O端口被配置为GPIO输出时：

- 输出通道开启
  - 开漏模式：输出寄存器上的‘0’激活N-MOS，而输出寄存器上的’1’将端口置于高阻状态(P-MOS不激活)。
  - 推挽模式：输出寄存器上的‘0’激活N-MOS，而输出寄存器上的’1’将激活P-MOS。
- 输入通道也开启
- 根据上拉配置的不同，上拉电阻被连接或断开
- 对输入数据寄存器的读访问可得到I/O状态
- 对输出数据寄存器的读访问得到最后一次写的值

下图给出了I/O端口位的GPIO输出配置。

图13-3　　I/O 端口位的 GPIO 输出配置框图

13.3.11 数字复用功能 （AF2-AF6）

当I/O端口被配置为数字复用功能时：

- 作为数字复用输入时输出通道被关闭
- 作为数字复用输出时输出通道被打开（但断开GPIO输出数据寄存器）
- 数字复用输入或输出属性是由外设决定的，在功能激活时自动配置
- 无论是输入还是输出，输入通道都是打开的
- 根据上拉配置的不同，上拉电阻被连接或断开
- 读输入数据寄存器时可得到I/O口状态

下图示出了I/O端口位数字复用功能配置：

图13-4　　I/O 端口位的数字复用功能配置框图

如果软件把一个I/O脚配置成数字复用输出功能，但是外设没有被激活，此时输出通道被关闭，输出通道呈现高阻状态。 当然用户可通过配置上拉给出一个确定电平。

说明：GPIO输入功能、GPIO输出功能和数字复用功能的I/O输入通道都是打开的，因此实际上当I/O口设置为默认的GPIO端口（AF1），仅仅是额外打开了I/O输出通道。

13.3.12 模拟复用功能 （AF6/7）

当I/O端口被配置为模拟复用配置时：

- 输出通道被关闭；
- 自动关闭输入通道，实现了每个模拟I/O引脚上的零消耗，施密特触发输出值被强置为‘0’；
- 读取输入数据寄存器时数值为‘0’；
- 允许开启上拉；

下图示出了I/O端口位的模拟复用配置：

图13-5　　I/O 端口位的模拟复用功能配置框图

13.3.13 全关功能 （AF0）

此功能是I/O口的一种特殊状态，通过切断输入通道和输出通道使能位，可以消除由于I/O口误配置导致的漏电问题，也就是说，简单起见，可以把不用的I/O口配置为全关。

上电复位期间和复位完成后，除SWD接口外的所有I/O口都处于全关状态。

13.3.14 I/O 引脚的基本应用

从应用上来看，I/O引脚只有3种功能：GPIO功能、外设功能（模拟/数字）和全关。使用数字外设时，需要根据不同外设类型对端口位进行配置。

一种推荐的外设配置流程为：

（1） 根据复用口的输入输出属性，配置好上拉（一般针对输入，特殊情况也可以对输出），推挽或开漏（输出）；

（2） 开启对应外设，设备待机；

（3） 切换到目标外设通道；

（4） 关闭外设时如有必要再配置上拉以给出确定电平（关闭外设时输出通道会自动切换到高阻）。

实际应用中有些外设（2）和（3）顺序也可以交换，即先切换外设通道，再使能外设，但一般情况下建议按照上述推荐流程操作。

以串口模块UART0举例说明： UART0为数字外设，配置如下：

复用引脚	功能定义	GPIO配置
PA8/TXD0	异步全双工模式，数据发送	推挽输出，关闭上拉 AFR8 = 100b（AF4）
	同步半双工模式，时钟输出	推挽输出，关闭上拉 AFR8 = 100b（AF4）
PA7/RXD0	异步全双工模式，数据接收	浮空输入或带上拉输入 AFR7= 100b（AF4）
	同步半双工模式，双向数据	推挽输出与浮空输入自动切换，关闭上拉或开启上拉 AFR7 = 100b（AF4）

以电机控制模块MCM举例说明： MCM为数字外设，配置如下：

复用引脚	功能定义	GPIO配置
PB6/ PWM0	PWM互补输出通道0	推挽输出，关闭上拉 AFR6 = 010b（AF2）
PB5/ PWM1	PWM互补输出通道1	推挽输出，关闭上拉 AFR5 = 010b（AF2）
PB4/ PWM2	PWM互补输出通道2	推挽输出，关闭上拉 AFR4 = 010b（AF2）
PB3/ PWM01	PWM互补输出通道01	推挽输出，关闭上拉 AFR3 =010b（AF2）
PB2/ PWM11	PWM互补输出通道11	推挽输出，关闭上拉 AFR2 =010b（AF2）
PB1/ PWM21	PWM互补输出通道21	推挽输出，关闭上拉 AFR1 =010b（AF2）
PB0/PWMFLT	PWM保护信号输入	浮空输入 AFR0 = 010b（AF2）

以运放模块OP2举例说明： OP2为模拟外设，配置如下：

复用引脚	功能定义	GPIO配置
PC7/ OP1P/ CP1P2	运放正极性输入	OP1EN=1 AFR7 = 111b（AF7）
PC6/OP1N	运放负极性输入	OP1EN=1 AFR6 = 111b（AF7）
PC5/OP1OUT	运放输出	OP1EN=1 AFR5 = 111b（AF7）

注意：以上配置并未禁止CP1P2的输入使能，如果CP1相应输入通道开启，PC7引脚上的信号会同时进入OP1和CP1。

13.4 寄存器

GPIO 模块寄存器列表 (基地址：0x4004 0000)

地址	寄存器名	说明
0x4004 0000+ 0x080*x+0x000	MODER	端口x模式寄存器 (x代表端口A、B、C，在计算偏移地址时对应取值为0、1、2，下同)
0x4004 0000+ 0x080*x+0x004	IDR	端口x输入数据寄存器
0x4004 0000+ 0x080*x+0x008	ODR	端口x输出数据寄存器
0x4004 0000+ 0x080*x+0x00C	BSRR	端口x位清除/置位寄存器
0x4004 0000+ 0x080*x+0x010	LCKR	端口x配置锁定寄存器
0x4004 0000+ 0x080*x+0x014	IE	端口x输入使能配置寄存器
0x4004 0000+ 0x080*x+0x018	OTYPER	端口x输出方式配置寄存器
0x4004 0000+ 0x080*x+0x01C	ODRVR	端口x输出驱动能力配置寄存器
0x4004 0000+ 0x080*x+0x020	PUPDR	端口x上下拉配置寄存器
0x4004 0000+ 0x080*x+0x024	TTLEN	端口x部分引脚TTL电平选择寄存器
0x4004 0000+ 0x080*x+0x028	AFRL	端口x复用功能寄存器低位
0x4004 0000+ 0x080*x+0x02C	AFRH	端口x复用功能寄存器高位

13.4.1 端口 x 模式寄存器 (GPIOx_MODER)

偏移地址: 0x0000 + 0x080*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
MODER15	MODER14	MODER13	MODER12	MODER11	MODER10	MODER9	MODER8	MODER7	MODER6	MODER5	MODER4	MODER3	MODER2	MODER1	MODER0
RW	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15	**MODER15**	**端口x模式配置** 0：输入（复位状态） 1：输出
14	**MODER14**	**端口x模式配置** 0：输入（复位状态） 1：输出

13	**MODER13**	**端口x模式配置** 0：输入（复位状态） 1：输出
12	**MODER12**	**端口x模式配置** 0：输入（复位状态） 1：输出
11	**MODER11**	**端口x模式配置** 0：输入（复位状态） 1：输出
10	**MODER10**	**端口x模式配置** 0：输入（复位状态） 1：输出
9	**MODER9**	**端口x模式配置** 0：输入（复位状态） 1：输出
8	**MODER8**	**端口x模式配置** 0：输入（复位状态） 1：输出
7	**MODER7**	**端口x模式配置** 0：输入（复位状态） 1：输出
6	**MODER6**	**端口x模式配置** 0：输入（复位状态） 1：输出
5	**MODER5**	**端口x模式配置** 0：输入（复位状态） 1：输出
4	**MODER4**	**端口x模式配置** 0：输入（复位状态） 1：输出
3	**MODER3**	**端口x模式配置** 0：输入（复位状态） 1：输出
2	**MODER2**	**端口x模式配置** 0：输入（复位状态） 1：输出
1	**MODER1**	**端口x模式配置** 0：输入（复位状态） 1：输出
0	**MODER0**	**端口x模式配置** 0：输入（复位状态） 1：输出

13.4.2 端口 x 输入数据寄存器 (GPIOx_IDR)

偏移地址: 0x0004 + 0x080*x

复位值: 0x0000 XXXX

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
IDRy(y=15~0)															
RO															
X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**IDRy(y=15~0)**	**端口x输入数据寄存器 (Port x input data register)** 该寄存器只读，存储端口读到的电平值。

13.4.3 端口 x 输出数据寄存器 (GPIOx_ODR)

偏移地址: 0x0008 + 0x080*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
ODRy(y=15~0)															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**ODRy(y=15~0)**	**端口x输出数据寄存器 (Port x output data register)** 该寄存器可以软件读写。通过后面的BSRR寄存器，ODR能够独立的被按位set和reset，以实现原子位操作。

13.4.4 端口位清除/置位寄存器 (GPIOx_BSRR)

偏移地址: 0x000C + 0x080*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
BRy(y=15~0)															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
BSy(y=15~0)															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**BRy(y=15~0)**	**端口x位清除 (Port x reset bit)** 这些位是只写，读取这些位返回0x0000。 0: 无影响 1: 清除相应的ODR位 注：如果同时设置同一个Port口的BRy和BSy，则BSy优先级高。
15 ~ 0	**BSy(y=15~0)**	**端口x位置位 (Port x set bit)** 这些位是只写，读取这些位返回0x0000。 0: 无影响 1: 置位相应的ODR位 注：如果同时设置同一个Port口的BRy和BSy，则BSy优先级高。

13.4.5 端口 x 配置锁定寄存器 (GPIOx_LCKR)

偏移地址: 0x0010 + 0x080*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
LOCK[15:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
LCK 15	**LCK 14**	**LCK 13**	**LCK 12**	**LCK 11**	**LCK 10**	**LCK 9**	**LCK 8**	**LCK 7**	**LCK 6**	**LCK 5**	**LCK 4**	**LCK 3**	**LCK 2**	**LCK 1**	**LCK 0**
RW	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**LOCK[15:0]**	**LCK控制位操作解锁位** 这些位是只写，读取这些位返回0x0000。 0x5AA5： 解锁 其他：锁定
15	**LCK15**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
14	**LCK14**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
13	**LCK13**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
12	**LCK12**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
11	**LCK11**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）

10	**LCK10**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
9	**LCK9**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
8	**LCK8**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
7	**LCK7**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
6	**LCK6**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
5	**LCK5**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
4	**LCK4**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
3	**LCK3**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
2	**LCK2**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
1	**LCK1**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）
0	**LCK0**	**端口x锁定位** 0: 端口位配置未锁定 1: 端口位配置锁定 该位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要）

13.4.6 端口 x 输入使能配置寄存器 (GPIOx_IE)

偏移地址: 0x0014 + 0x080*x

复位值: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
IE15	IE14	IE13	IE12	IE11	IE10	IE9	IE8	IE7	IE6	IE5	IE4	IE3	IE2	IE1	IE0
RW	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位	符号	说明
31 ~ 16	保留	-
15	**IE15**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
14	**IE14**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
13	**IE13**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
12	**IE12**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
11	**IE11**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
10	**IE10**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
9	**IE9**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
8	**IE8**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
7	**IE7**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
6	**IE6**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
5	**IE5**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
4	**IE4**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）

3	**IE3**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
2	**IE2**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
1	**IE1**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）
0	**IE0**	**端口x输入使能配置** 0: 输入关闭 1: 输入使能（复位状态）

13.4.7 端口 x 输出方式配置寄存器 (GPIOx_OTYPER)

偏移地址: 0x0018 + 0x080*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留	OT14	OT13	保留										OT2	保留	OT0
-	*RW*	*RW*	-										*RW*	-	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 15	保留	-
14	**OT14**	**端口PA14输出方式配置** 0：输出推挽模式（复位状态） 1：输出开漏模式
13	**OT13**	**端口PA13输出方式配置** 0：输出推挽模式（复位状态） 1：输出开漏模式
12 ~ 3	保留	-
2	**OT2**	**端口PA2输出方式配置** 0：输出推挽模式（复位状态） 1：输出开漏模式
1	保留	-
0	**OT0**	**端口PA0输出方式配置** 0：输出推挽模式（复位状态） 1：输出开漏模式

注：OT属性只有PA的4个口具备，PB、PC口此寄存器无意义（地址空间保留，但寄存器不存在）

13.4.8 端口 x 输出驱动能力配置寄存器 (GPIOx_ODRVR)

偏移地址: 0x001C + 0x080*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	保留	-

13.4.9 端口 x 上下拉配置寄存器 (GPIOx_PUPDR)

偏移地址: 0x0020 + 0x080*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留	PHDR15	保留	PHDR14	保留	PHDR13	保留	PHDR12	保留	PHDR11	保留	PHDR10	保留	PHDR9	保留	PHDR8
-	*RW*	-	*RW*	-	*RW*	-	*RW*	-	*RW*	-	*RW*	-	*RW*	-	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留	PHDR7	保留	PHDR6	保留	PHDR5	保留	PHDR4	保留	PHDR3	保留	PHDR2	保留	PHDR1	保留	PHDR0
-	*RW*	-	*RW*	-	*RW*	-	*RW*	-	*RW*	-	*RW*	-	*RW*	-	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31	保留	-
30	**PHDR15**	端口x上拉配置 0：无上拉（复位状态） 1：上拉
29	保留	-
28	**PHDR14**	端口x上拉配置 0：无上拉（复位状态） 1：上拉
27	保留	-
26	**PHDR13**	端口x上拉配置 0：无上拉（复位状态） 1：上拉
25	保留	-
24	**PHDR12**	端口x上拉配置 0：无上拉（复位状态） 1：上拉
23	保留	-

22	**PHDR11**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉
21	保留	-
20	**PHDR10**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉
19	保留	-
18	**PHDR9**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉
17	保留	-
16	**PHDR8**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉
15	保留	-
14	**PHDR7**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉
13	保留	-
12	**PHDR6**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉
11	保留	-
10	**PHDR5**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉
9	保留	-
8	**PHDR4**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉
7	保留	-
6	**PHDR3**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉
5	保留	-
4	**PHDR2**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉
3	保留	-
2	**PHDR1**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉
1	保留	-
0	**PHDR0**	**端口x上拉配置** 0：无上拉（复位状态） 1：上拉

13.4.10 端口 x 部分引脚 TTL 电平选择寄存器 (GPIOBx_TTLEN)

偏移地址: 0x0024 + 0x080*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留	TTLEN14	TTLEN13	TTLEN12	TTLEN11	保留	TTLEN9	保留		TTLEN6	保留			TTLEN2	TTLEN1	TTLEN0
-	*RW*	*RW*	*RW*	*RW*	-	*RW*	-		*RW*	-			*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 15	保留	-
14	**TTLEN14**	**PA14引脚TTL/CMOS输入选择** 0 : 选择CMOS输入 1: 选择TTL输入 TTL参数见“电气特性”，在停机模式下不做任何处理。
13	**TTLEN13**	**PA13引脚TTL/CMOS输入选择** 0 : 选择CMOS输入 1: 选择TTL输入 TTL参数见“电气特性”，在停机模式下不做任何处理。
12	**TTLEN12**	**PA12引脚TTL/CMOS输入选择** 0 : 选择CMOS输入 1: 选择TTL输入 TTL参数见“电气特性”，在停机模式下不做任何处理。
11	**TTLEN11**	**PA11引脚TTL/CMOS输入选择** 0 : 选择CMOS输入 1: 选择TTL输入 TTL参数见“电气特性”，在停机模式下不做任何处理。
10	保留	-
9	**TTLEN9**	**PA9引脚TTL/CMOS输入选择** 0 : 选择CMOS输入 1: 选择TTL输入 TTL参数见“电气特性”，在停机模式下不做任何处理。
8 ~ 7	保留	-
6	**TTLEN6**	**PA6引脚TTL/CMOS输入选择** 0 : 选择CMOS输入 1: 选择TTL输入 TTL参数见“电气特性”，在停机模式下不做任何处理。
5 ~ 3	保留	-
2	**TTLEN2**	**PA2引脚TTL/CMOS输入选择** 0 : 选择CMOS输入 1: 选择TTL输入 TTL参数见“电气特性”，在停机模式下不做任何处理。
1	保留	-
0	**TTLEN0**	**PA0引脚TTL/CMOS输入选择** 0 : 选择CMOS输入 1: 选择TTL输入 TTL参数见“电气特性”，在停机模式下不做任何处理。

注：TTL属性只有部分引脚具备，需对照“TTL输入选择”章节确认具体引脚。

13.4.11 端口 x 复用功能寄存器低位 (GPIOx_AFRL)

偏移地址: 0x0028 + 0x080*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留	AFR7[2:0]			保留	AFR6[2:0]			保留	AFR5[2:0]			保留	AFR4[2:0]		
-	RW			-	RW			-	RW			-	RW		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留	AFR3[2:0]			保留	AFR2[2:0]			保留	AFR1[2:0]			保留	AFR0[2:0]		
-	RW			-	RW			-	RW			-	RW		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31	保留	-
30 ~ 28	**AFR7[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
27	保留	-
26 ~ 24	**AFR6[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
23	保留	-
22 ~ 20	**AFR5[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
19	保留	-
18 ~ 16	**AFR4[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
15	保留	-

14 ~ 12	**AFR3[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
11	保留	-
10 ~ 8	**AFR2[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
7	保留	-
6 ~ 4	**AFR1[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
3	保留	-
2 ~ 0	**AFR0[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7

13.4.12 端口 x 复用功能寄存器高位 (GPIOx_AFRH)

偏移地址: 0x002C + 0x080*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留	AFR15[2:0]			保留	AFR14[2:0]			保留	AFR13[2:0]			保留	AFR12[2:0]		
-	RW			-	RW			-	RW			-	RW		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留	AFR11[2:0]			保留	AFR10[2:0]			保留	AFR9[2:0]			保留	AFR8[2:0]		
-	RW			-	RW			-	RW			-	RW		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31	保留	-
30 ~ 28	**AFR15[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
27	保留	-
26 ~ 24	**AFR14[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
23	保留	-
22 ~ 20	**AFR13[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
19	保留	-
18 ~ 16	**AFR12[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
15	保留	-

14 ~ 12	**AFR11[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
11	保留	-
10 ~ 8	**AFR10[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
7	保留	-
6 ~ 4	**AFR9[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7
3	保留	-
2 ~ 0	**AFR8[2:0]**	端口x复用功能选择 000: AF0 001: AF1 010: AF2 011: AF3 100: AF4 101: AF5 110: AF6 111: AF7

14. 系统配置模块(SYSCFG)

此模块用于系统层面的辅助模块和系统配置。SYSCFG挂在AHB总线上，有独立的使能与复位控制。

14.1 系统配置相关

14.1.1 SWD 和 XTAL 引脚复用功能定义

调试器接口是两线SW接口（SWDIO和SWCLK），称为SWD接口，通过SWJCFG@SYSCFG_SAFR可以配置为GPIO口。一旦配置完成，只能通过复位回到调试接口状态。

外部振荡器引脚XTAL1和XTAL2默认用作GPIO接口，通过OSCCFG@SYSCFG_SAFR可配置为外部振荡器接口或外灌时钟接口，一旦配置完成，只能通过复位回到默认接口状态。

SWD接口和XTAL接口在作为非GPIO接口使用时，GPIO的配置参数无效（可以写入但无效果），包含输入输出方向的配置也无效。

14.1.2 NMI 中断信号源配置

晶振异常事件、掉电事件、外部中断事件都可以配置为产生NMI中断，通过SAFR寄存器的IEN_CSM、IEN_BOD、IEN_EXTI0使能。

14.1.3 调试接口寄存器

DBGCR为调试接口寄存器，此处仅定义了调试模式下看门狗状态以及低功耗模式下的调试选项，具体参见“调试接口”章节。

14.2 寄存器

SYSCFG 模块寄存器列表 (基地址：0x4004 0800)

地址	寄存器名	说明
0x4004 0808	SAFR	系统配置寄存器
0x4004 0810	DBGCR	调试接口控制寄存器

14.2.1 系统配置寄存器 (SYSCFG_SAFR)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
LOCK[15:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								IEN_CSM	IEN_BOD	IEN_EXTI0	保留		SWJCFG	OSCCFG[1:0]	
-								*RW*	*RW*	*RW*	-		*RW1t*	*RW1t*	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**LOCK[15:0]**	**此寄存器操作解锁位 (Control bit Lock register)** 这些位是只写，读取这些位返回0x00。 0x5AA5： 解锁 其他： 锁定 注：写入此寄存器需配合解锁控制，读此寄存器不需要解锁。
15 ~ 8	保留	-
7	**IEN_CSM**	**CSM引起NMI中断使能位**
6	**IEN_BOD**	**BOD引起NMI中断使能位**
5	**IEN_EXTI0**	**EXTI0引起NMI中断使能位**
4 ~ 3	保留	-
2	**SWJCFG**	**串行线引脚复用功能定义** 0：SWDIO/SWCLK引脚为调试端口，不复用为普通I/O； 1：关闭SW调试功能；SWCLK、SWDIO可复用为普通I/O。但SWCLK、SWDIO仅在正常运行状态下可复用为普通I/O，在调试模式状态下仍强制作为调试接口。 注：SWD包含SWDIO/SWCLK引脚。此控制位一旦设置，只有发生复位才能修改。用户需谨慎配置该位，因会导致调试模式需在复位信号配合下才能进入。
1 ~ 0	**OSCCFG[1:0]**	**XTAL1/XTAL2引脚复用功能定义** 00： XTAL1/XTAL2作为GPIO使用（默认） 01：外部振荡器接口（晶振和陶振） 10：XTAL1上外灌时钟，XTAL2作为GPIO使用 11：保留 注：此控制位一旦设置，只有reset才能修改；

14.2.2 调试接口控制寄存器 (SYSCFG_DBGCR)

偏移地址: 0x0010

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
LOCK[15:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DBG_TWI	DBG_SPI	DBG_UART	DBG_MCM	DBG_TIM	DBG_PCA	DBG_WWDT	DBG_IWDT	DBG_QEI	DBG_CAN	保留				DBG_STOP	保留
RW	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	-				*RW*	-
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**LOCK[15:0]**	**此寄存器操作解锁位 (Control bit Lock register)** 0x5AA5： 解锁 其他: 锁定 注：写入此寄存器需配合解锁控制，读此寄存器不需要解锁。
15	**DBG_TWI**	**当内核进入调试状态时TWI停止工作** 0：TWI停止工作 1：TWI仍然正常工作
14	**DBG_SPI**	**当内核进入调试状态时SPI停止工作** 0：SPI停止工作 1：SPI仍然正常工作
13	**DBG_UART**	**当内核进入调试状态时UART停止工作** 0：UART停止工作 1：UART仍然正常工作
12	**DBG_MCM**	**当内核进入调试状态时MCM停止工作** 0：MCM停止工作 1：MCM仍然正常工作
11	**DBG_TIM**	**当内核进入调试状态时TIM7-TIM8模块停止工作** 0：TIM7-TIM8模块停止工作 1：TIM7-TIM8模块仍然正常工作
10	**DBG_PCA**	**当内核进入调试状态时PCA停止工作** 0：PCA停止工作 1：PCA仍然正常工作
9	**DBG_WWDT**	**当内核进入调试状态时调试窗口看门狗停止工作。** 0：窗口看门狗计数器停止工作 1：窗口看门狗计数器仍然正常工作
8	**DBG_IWDT**	**当内核进入调试状态时看门狗停止工作** 0：看门狗计数器停止工作 1：看门狗计数器仍然正常工作

7	**DBG_QEI**	**当内核进入调试状态时QEI停止工作** 0：QEI停止工作 1：QEI仍然正常工作
6	**DBG_CAN**	**当内核进入调试状态时CAN停止工作** 0：CAN停止工作 1：CAN仍然正常工作
5 ~ 2	保留	-
1	**DBG_STOP**	**调试停机模式。** 0：(FCLK开，HCLK开)在停机模式时，FCLK和HCLK时钟由内部RC振荡器提供。当退出停机模式时，软件必需重新配置时钟系统启动PLL，晶振等(与配置此比特位为0时的操作一样) 1：(FCLK关，HCLK关)在停机模式时，时钟控制器禁止一切时钟(包括HCLK和FCLK)。当从停机模式退出时，时钟的配置和复位之后的配置一样(微控制器由8MHz的内部RC振荡器(HSI)提供时钟)。因此，软件必需重新配置时钟控制系统启动PLL，晶振等 注：DBG_STOP=0时HCLK保持开启，HCLK可维持调试接口的连接，即调试器可读取停机前的CPU和各外设的状态信息。另外这种状态下FCLK和HCLK只由HSI提供。
0	保留	-

15. 中断和事件

15.1 可嵌套中断向量控制器（NVIC）

15.1.1 简介

NVIC是ARM “星辰”处理器的主要组成部分，它与CPU紧密结合，降低了中断延时，并允许新进中断可以得到高效处理。

NVIC还可以处理系统异常，包括：复位(Reset)、非屏蔽中断(NMI)、硬件错误(Hard Fault)、内存错误(MemManage Fault)、总线错误(Bus Fault)、使用错误(Usage Fault)、调用系统服务(SVCall)、调试监控(Debug Monitor)、挂起服务(PendSV)及系统嘀嗒(Systick)。

15.1.2 主要特性

- 嵌套向量控制器是ARM “星辰”处理器的一个组成部分
- 高度耦合的中断控制器提供了低中断延时
- 控制处理系统异常及外设中断
- SH33F2801支持29个可屏蔽中断(不包含16个ARM “星辰”处理器的中断)
- 8个可编程中断优先级（可进行优先级分组）
- 低延迟的异常和中断处理
- 中断可屏蔽
- 软件中断

可嵌套中断支持

可嵌套中断支持的作用范围很广，覆盖了所有的外部中断和绝大多数系统异常。这些异常都可以被赋予不同的优先级，当前优先级被存储在 xPSR 的专用字段中。当一个异常发生时，硬件会自动比较该异常的优先级是否比当前的异常优先级更高，如果发现来了更高优先级的异常，处理器就会中断当前的中断服务例程而服务新来的异常——即中断抢占。

向量中断支持

当开始响应一个中断后，内核 会自动定位一张向量表，并且根据中断号从表中找出 ISR 的入口地址，然后跳转过去执行。动态优先级调整支持软件可以在运行时期更改中断的优先级。中断不支持中断重入(reentry)，比如在某 ISR 中修改了自己所对应中断的优先级，而且这个中断又有新的实例处于挂起中（pending），结果是当前中断继续执行，不会自己打断自己，从而没有重入风险。

中断延迟大大缩短

ARM “星辰”处理器 为了缩短中断延迟，引入了好几个新特性。包括自动的现场保护和恢复，以及其它的措施，用于缩短中断嵌套时的 ISR响应延迟。更多信息请参考ARM官方手册。

中断可屏蔽

既可以屏蔽优先级低于某个阈值的中断/异常 (设置 BASEPRI 寄存器)，也可以全体屏蔽(设置 PRIMASK 和 FAULTMASK 寄存器)。这是为了让时间关键（time - critical）任务能在最后期限(deadline)到来前完成，而不被干扰。

15.1.3 中断汇总

SH33F2801中断的所有中断源汇总如下表：

NMI源包含CSM、EXTI0、BOD，有单独的控制位（见SYSCFG_SAFR寄存器）。EXTI0除了NMI入口，还有独立的EXTI0入口，因此需要选择产生NMI还是EXTI0（可以同时使能），BOD也有两个中断入口，而CSM只有NMI中断入口。

9 Exception + 29 HWINT

中断号/异常号	中断源	向量地址	优先级	描述
-16 / 0	-	0x0000	-	堆栈初始指针值
-15 / 1	Reset	0x0004	-3(最高)	复位PC值，异步的
-14 / 2	NMI	0x0008	-2	非屏蔽中断。来自外部中断输入脚（仅EXTI0）及时钟监控信号（CSM），掉电监测信号(BOD)。异步属性。
-13 / 3	HardFault	0x000C	-1	由于优先级的原因或可配置的故障处理被禁止而导致不能将故障激活时的所有类型故障。同步属性。
-12 / 4	MemManage	0x0010	可调整(1)	是同步的。可以用它来支持默认的存储器映射的XN 区域。
-11 / 5	BusFault	0x0014	可调整(1)	预取故障，存储器访问故障，以及其它相关的地址/存储故障。精确时是同步的，不精确时是异步的。
-10 / 6	UsageFault	0x0018	可调整(1)	使用故障。例如，执行未定义的指令或尝试不合法的状态转换。同步属性。
-9 / 7	-	0x001C	-	
-8 / 8	-	0x0020	-	
-7 / 9	-	0x0024	-	
-6 / 10	-	0x0028	-	
-5 / 11	SVCall	0x002C	可调整(1)	利用SVC指令调用系统服务。同步属性。
-4 / 12	DebugMonitor	0x0030	可调整(1)	调试监控，在处理器没有停止时出现。是同步的，但只有在使能时是有效的。 如果它的优先级比当前有效的异常的优先级要低，则不能被激活。
-3 / 13	-	0x0034		
-2 / 14	PendSV	0x0038	可调整(1)	可挂起的系统服务请求。 是异步的， 只能由软件来实现挂起。
-1 / 15	SysTick	0x003C	可调整(1)	系统节拍定时器（tick timer）递减为0。异步属性。
0	WWDT	0x0040	可调整(2)	窗口看门狗定时器中断
1	BOD	0x0044	可调整(2)	掉电检测中断
2	RCC	0x0048	可调整(2)	PLL及外部时钟就绪中断
3	EXTI0	0x004C	可调整(2)	EXTI引脚0中断
4	EXTI1	0x0050	可调整(2)	EXTI引脚1中断
5	EXTI2	0x0054	可调整(2)	EXTI引脚2中断
6	EXTI3	0x0058	可调整(2)	EXTI引脚3中断
7	MCM_FLT	0x005C	可调整(2)	MCM模块PWM故障中断
8	MCM_PWM	0x0060	可调整(2)	MCM模块PWM中断
9	ADC	0x0064	可调整(2)	ADC中断
10	PCA0	0x0068	可调整(2)	PCA0中断
11	PCA1	0x006C	可调整(2)	PCA1中断
12	PCA0_FLT	0x0070	可调整(2)	PCA0故障中断
13	PCA1_FLT	0x0074	可调整(2)	PCA1故障中断
14	CMP0	0x0078	可调整(2)	比较器0中断
15	CMP1	0x007C	可调整(2)	比较器1中断
16	QEI	0x0080	可调整(2)	正交编码器接口(QEI)中断
17	CAN	0x0084	可调整(2)	CAN中断
18	TWI	0x0088	可调整(2)	双线串行接口中断
19	SPI	0x008C	可调整(2)	SPI 中断
20	UART0	0x0090	可调整(2)	UART0中断
21	UART1	0x0094	可调整(2)	UART1中断
22	UART2	0x0098	可调整(2)	UART2中断
23	TIM7	0x009C	可调整(2)	定时器7中断
24	TIM8	0x00A0	可调整(2)	定时器8中断
25	EXTI4_7	0x00A4	可调整(2)	EXTI引脚4~7中断
26	FIFO0	0x00A8	可调整(2)	FIFO0中断
27	FIFO1	0x00AC	可调整(2)	FIFO1中断
28	FIFO2	0x00B0	可调整(2)	FIFO2中断

注：1. 使用 NVIC 系统处理程序优先级寄存器，异常类型的优先级可改变。更多信息请参考ARM官方手册。

2. 使用NVIC中断优先级寄存器， 中断优先级可改变。

对于上述中断源（系统复位中断除外），只有对应的中断屏蔽位和中断使能位都允许触发中断时，对应中断标志被硬件置1的同时才会触发中断，中断响应过程由中断优先级设置和嵌套设置决定。

15.1.4 NVIC 寄存器列表

地址	寄存器	类型	说明	复位值
0xE000E004	ICTR	只读	中断控制类型寄存器	0x00000001
0xE000E010	SCTRL	读/写	系统时钟节拍（SysTick）控制与状态寄存器	0x00000000
0xE000E014	SLOAD	读/写	系统时钟节拍（SysTick）重装值寄存器	不可预测
0xE000E018	SVAL	读/写清除	系统时钟节拍（SysTick）当前值寄存器	不可预测
0xE000E01C	SCALIB	只读	系统时钟节拍（SysTick）校准值寄存器	0x40019A28
0xE000E100	ISER0_31	读/写	Irq0～31使能设置寄存器	0x00000000
0xE000E104	ISER32_63	读/写	Irq32～63使能设置寄存器	0x00000000
0xE000E180	ICER0_31	读/写	Irq0～31使能清除寄存器	0x00000000
0xE000E184	ICER32_63	读/写	Irq32～63使能清除寄存器	0x00000000
0xE000E200	ISPR0_31	读/写	Irq0～31挂起设置寄存器	0x00000000
0xE000E204	ISPR32_63	读/写	Irq32～63挂起设置寄存器	0x00000000
0xE000E280	ICPR0_31	读/写	Irq0～31挂起清除寄存器	0x00000000
0xE000E284	ICPR32_63	读/写	Irq32～63挂起清除寄存器	0x00000000
0xE000E300	IABR0_31	只读	Irq0～31激活位寄存器	0x00000000
0xE000E304	IABR32_63	只读	Irq32～63激活位寄存器	0x00000000
0xE000E400	IPR0_3	读/写	Irq0~3优先级寄存器	0x00000000
0xE000E404	IPR4_7	读/写	Irq4~7优先级寄存器	0x00000000
0xE000E408	IPR8_11	读/写	Irq8~11 优先级寄存器	0x00000000
0xE000E40C	IPR12_15	读/写	Irq12~15优先级寄存器	0x00000000
0xE000E410	IPR16_19	读/写	Irq16~19优先级寄存器	0x00000000
0xE000E414	IPR20_3	读/写	Irq20~23优先级寄存器	0x00000000
0xE000E418	IPR24_27	读/写	Irq24~27优先级寄存器	0x00000000
0xE000E41C	IPR28_31	读/写	Irq28~31优先级寄存器	0x00000000
0xE000E420	IPR32_35	读/写	Irq32~35优先级寄存器	0x00000000
0xE000E424	IPR36_39	读/写	Irq36~39优先级寄存器	0x00000000
0xE000E428	IPR40_43	读/写	Irq40~43优先级寄存器	0x00000000
0xE000E42C	IPR44_47	读/写	Irq44~47优先级寄存器	0x00000000
0xE000E430	IPR48_51	读/写	Irq48~51优先级寄存器	0x00000000
0xE000E434	IPR52_55	读/写	Irq52~55优先级寄存器	0x00000000
0xE000E438	IPR56_59	读/写	Irq56~59优先级寄存器	0x00000000
0xE000E43C	IPR60_63	读/写	Irq60~63优先级寄存器	0x00000000
0xE000ED00	CPUID	只读	CPUID基址寄存器	0x412FC231
0xE000ED04	ICSR	读/写或只读	中断控制状态寄存器	0x00000000
0xE000ED08	VTOR	读/写	向量表偏移寄存器	0x00000000
0xE000ED0C	AIRCR	读/写	应用中断/复位控制寄存器	0x00000000
0xE000ED10	SCR	读/写	系统控制寄存器	0x00000000
0xE000ED14	CCR	读/写	配置控制寄存器	0x00000000
0xE000ED18	SHPR4_7	读/写	系统处理程序4-7优先级寄存器，MemManage为4，BusFault为5，UsageFault为6	0x00000000
0xE000ED1C	SHPR8_11	读/写	系统处理程序8-11优先级寄存器，SVCall为11	0x00000000
0xE000ED20	SHPR12_15	读/写	系统处理程序12-15 优先级寄存器，PendSV为14，SysTick为15	0x00000000
0xE000ED24	SHCSR	读/写	系统处理器控制与状态寄存器	0x00000000
0xE000ED2C	CFSR	读/写	可配置故障状态寄存器	0x00000000
0xE000ED30	HFSR	读/写	硬故障状态寄存器	0x00000000
0xE000ED34	DFSR	读/写	调试故障状态寄存器	0x00000000
0xE000ED38	MMFAR	读/写	存储器管理地址寄存器	不可预测
0xE000ED3C	BFAR	读/写	总线故障地址寄存器	不可预测
0xE000ED40	AFSR	读/写清除	辅助故障状态寄存器	0x00000000
0xE000EF00	STIR	只写	软件触发中断寄存器	-

注：更多以上寄存器的详细描述，请参考ARM官方手册。

15.2 外部中断/事件控制器（EXTI）

外部中断/事件控制器由8个产生事件/中断请求的边沿/电平检测器组成，这8条线连接到特定的GPIO口(具体见引脚图)。每个输入线可以独立地配置输入类型(脉冲或挂起)和对应的触发事件(上升沿/下降沿/双边沿/高电平/低电平/双电平[注])。每个输入线都可以独立地被屏蔽。挂起寄存器保持着状态线的中断请求。

注：双电平会造成主程序陷入EXTI中断中无法退出，硬件上不屏蔽这种设置，由用户软件屏蔽。

15.2.1 主要特性

- 每个中断/事件都有独立的触发和屏蔽
- 每个中断线都有专用的状态位
- 支持8个软件的中断/事件请求
- 输入检测采样参数可调，在正常运行和低功耗模式下都具备去抖能力

图15-1　　外部中断/事件控制器框图

说明：EXTI的请求挂起寄存器和中断屏蔽寄存器和NVIC是独立的。
EXTI0能引起NMI中断，由IEN_EXTI0@SYSCFG_SAFR控制位使能。

15.2.2 唤醒事件管理

SH33F2801可以处理外部或内部事件来唤醒内核(WFE)。唤醒事件可以通过下述配置产生：

在外设的控制寄存器使能一个中断，但不在NVIC中使能，同时在ARM “星辰”处理器的系统控制寄存器中使能SEVONPEND位。当CPU从WFE恢复后，需要清除相应外设的中断挂起位（在EXTI挂起寄存器中）和外设NVIC中断通道挂起位(在NVIC中断挂起清除寄存器中)。

配置一个外部或内部EXTI线为事件模式，当CPU从WFE恢复后，因为对应事件线的挂起位没有被置位，不必清除相应外设的中断挂起位或NVIC中断通道挂起位。使用外部I/O端口作为唤醒事件，请参见GPIO章节的“GPIO配置为外部中断/唤醒线”说明。

EXTI0和EXTI2可以唤醒STOP，其他EXTI无法唤醒STOP。 所有EXTI都可以唤醒SLEEP。

15.2.3 功能说明

要产生中断，必须先配置好并使能中断线。根据需要的边沿检测设置2个触发寄存器，同时在中断屏蔽寄存器的相应位写’1’允许中断请求。当外部中断线上发生了期待的边沿时，将产生一个中断请求，对应的挂起位也随之被置‘1’。在挂起寄存器的对应位写’1’，将清除该中断请求。

如果需要产生事件，必须先配置好并使能事件线。根据需要的边沿检测通过设置2个触发寄存器，同时在事件屏蔽寄存器的相应位写’1’允许事件请求。当事件线上发生了需要的边沿时，将产生一个事件请求脉冲，对应的挂起位不被置‘1’。 通过在软件中断/事件寄存器写’1’，也可以通过软件产生中断/事件请求。

SH33F2801也支持电平触发，通过设置触发模式为电平触发，并设置高电平或低电平触发，EXTI可以工作在电平触发状态，与边沿触发不同，如果触发电平不改变，将会持续触发。

EXTI输入信号支持采样滤波，在正常运行模式下使用APB时钟PCLK作为时钟源，支持采样时钟预分频（最高128分频），并支持多次采样以达到滤波效果。在停机模式下使用LSI作为时钟源[注]。 具体见“EXTI采样控制寄存器”。

注：LSI作为滤波时钟源时由于频率较低，硬件上屏蔽了时钟分频，但多次采样功能是保留的。

硬件中断选择

通过下面的过程来配置8个输入线做为中断源：

- 配置8个中断线的屏蔽位(EXTI_IMR)
- 配置所选中断线的触发选择位(EXTI_RTSR和EXTI_FTSR)；
- 配置对应到外部中断控制器(EXTI)的NVIC中断通道的使能和屏蔽位，使得8个中断线中的请求可以被正确地响应。

硬件事件选择

通过下面的过程，可以配置8个线路为事件源

- 配置8个事件线的屏蔽位(EXTI_EMR)
- 配置事件线的触发选择位(EXTI_RTSR和EXTI_FTSR)

软件中断/事件的选择

8个线路可以被配置成软件中断/事件线。下面是产生软件中断的过程：

- 配置8个中断/事件线屏蔽位(EXTI_IMR, EXTI_EMR)
- 设置软件中断寄存器的请求位(EXTI_SWIER)

15.2.4 外部中断/事件线路映像

8个外部中断固定连接到对应的I/O端口，具体见引脚图。

要开启EXTI线路连接，首先要把I/O口配置为AF1[注]（AF0方式下输入通路关闭，包括外部中断）。然后要开启外部中断/事件线路开关，该开关由EXTI_CFG寄存器相应位控制，EXTIn_EN(n=0~7)为1时使能。

注：AF2/AF3时，I/O口输入通路也是开启的，也可以配置，具体在“GPIO”章节有详细介绍。

表15-1　　EXTI 端口映射表

I/O口	EXTIn_EN	EXTI线	扩展功能
PB0	EXTI0_EN=1	EXTI0	支持引起NMI中断、支持唤醒STOP
PA13	EXTI1_EN=1	EXTI1	-
PA10	EXTI2_EN=1	EXTI2	支持唤醒STOP、支持作为TIM8时钟输入
PA9	EXTI3_EN=1	EXTI3	-
PA6	EXTI4_EN=1	EXTI4	-
PA2	EXTI5_EN=1	EXTI5	-
PA1	EXTI6_EN=1	EXTI6	-
PC12	EXTI7_EN=1	EXTI7	-

15.3 寄存器

EXTI 模块寄存器列表 (基地址：0x4000 4000)

地址	寄存器名	说明
0x4000 4000	IMR	中断屏蔽寄存器
0x4000 4004	EMR	事件屏蔽寄存器
0x4000 4008	TMSR	触发模式选择寄存器
0x4000 400C	RTSR	上升沿/高电平触发模式选择寄存器
0x4000 4010	FTSR	下降沿/低电平触发模式选择寄存器
0x4000 4014	SWIER	软件中断事件寄存器
0x4000 4018	PR	挂起寄存器
0x4000 401C	CFG	外部中断配置低寄存器
0x4000 4020	CFGH	外部中断配置高寄存器
0x4000 4024	SAMP	EXTI采样控制寄存器

15.3.1 中断屏蔽寄存器 (EXTI_IMR)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								IMRy(y=7~0)							
-								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 0	**IMRy(y=7~0)**	**线y上的中断屏蔽** 0: 屏蔽来自线y上的中断请求 1: 开放来自线y上的中断请求

15.3.2 事件屏蔽寄存器 (EXTI_EMR)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								EMRy(y=7~0)							
-								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 0	**EMRy(y=7~0)**	**线y上的事件屏蔽** 0: 屏蔽来自线y上的事件请求 1: 开放来自线y上的事件请求

15.3.3 触发模式选择寄存器 (EXTI_TMSR)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								TMRy(y=7~0)							
-								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 0	**TMRy(y=7~0)**	**线y上的触发模式选择位** 0: 边沿触发 1: 电平触发 注：边沿触发为one-shot方式，触发后通过软件clear pending flag来清除标志位。电平触发为level方式，触发后只能通过外部电平变化来清除标志位，无法通过软件清除。

15.3.4 上升沿/高电平触发模式选择寄存器 (EXTI_RTSR)

偏移地址: 0x000C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								RTRy(y=7~0)							
-								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 0	**RTRy(y=7~0)**	**线y上的上升沿/高电平触发事件配置位** 0：禁止输入线y上的上升沿/高电平触发(中断和事件) 1：允许输入线y上的上升沿/高电平触发(中断和事件)

15.3.5 下降沿/低电平触发模式选择寄存器 (EXTI_FTSR)

偏移地址: 0x0010

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								FTRy(y=7~0)							
-								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 0	**FTRy(y=7~0)**	**线y上的下降沿/低电平触发事件配置位** 0：禁止输入线y上的下降沿/低电平触发(中断和事件) 1：允许输入线y上的下降沿/低电平触发(中断和事件)

15.3.6 软件中断事件寄存器 (EXTI_SWIER)

偏移地址: 0x0014

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								SWIERy(y=7~0)							
-								*WO*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 0	**SWIERy(y=7~0)**	**线y上的软件中断 (Software interrupt on line y)** 当该位为'0'时，写'1'将设置EXTI_PR中相应的挂起位。如果在EXTI_IMR和EXTI_EMR中允许产生该中断，则此时将产生一个中断。 0：无意义 1：置位EXTI_PR相应pending位，引起软件触发 该信号由硬件自动清除。 如果对应的EXTI_PR已经置位，则本次操作无效。

15.3.7 挂起寄存器 (EXTI_PR)

偏移地址: 0x0018

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留								PRCy(y=7~0)							
-								*WO*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								PRy(y=7~0)							
-								*RO*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 24	保留	-
23 ~ 16	**PRCy(y=7~0)**	**挂起清除位** 0：无效 1：清除
15 ~ 8	保留	-
7 ~ 0	**PRy(y=7~0)**	**挂起位 (Pending bit)** 0：没有发生触发请求 1：发生了选择的触发请求 当在外部中断线上发生了选择的触发事件，该位被置‘1’。 注：对电平触发，如果外部电平没有变化，则清除pending位后将立刻又产生pending。

15.3.8 外部中断配置低寄存器 (EXTI_CFG)

偏移地址: 0x001C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								EXTI7_EN	EXTI6_EN	EXTI5_EN	EXTI4_EN	EXTI3_EN	EXTI2_EN	EXTI1_EN	EXTI0_EN
-								*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7	**EXTI7_EN**	**EXTI 7 通道配置开关** 0：EXTI7 输入通道关闭 1：EXTI7 输入通道打开
6	**EXTI6_EN**	**EXTI 6 通道配置开关** 0：EXTI6 输入通道关闭 1：EXTI6 输入通道打开
5	**EXTI5_EN**	**EXTI 5 通道配置开关** 0：EXTI5 输入通道关闭 1：EXTI5 输入通道打开

4	**EXTI4_EN**	**EXTI 4 通道配置开关** 0：EXTI4 输入通道关闭 1：EXTI4 输入通道打开
3	**EXTI3_EN**	**EXTI 3 通道配置开关** 0：EXTI3 输入通道关闭 1：EXTI3 输入通道打开
2	**EXTI2_EN**	**EXTI 2 通道配置开关** 0：EXTI2 输入通道关闭 1：EXTI2 输入通道打开
1	**EXTI1_EN**	**EXTI 1 通道配置开关** 0：EXTI1 输入通道关闭 1：EXTI1 输入通道打开
0	**EXTI0_EN**	**EXTI 0 通道配置开关** 0：EXTI0 输入通道关闭 1：EXTI0 输入通道打开

15.3.9 外部中断配置高寄存器 (EXTI_CFGH)

偏移地址: 0x0020

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	保留	**-**

15.3.10 EXTI 采样控制寄存器 (EXTI_SAMP)

偏移地址: 0x0024

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
PS7[1:0]		SN7[1:0]		PS6[1:0]		SN6[1:0]		PS5[1:0]		SN5[1:0]		PS4[1:0]		SN4[1:0]	
RW		*RW*		*RW*		*RW*		*RW*		*RW*		*RW*		*RW*	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PS3[1:0]		SN3[1:0]		PS2[1:0]		SN2[1:0]		PS1[1:0]		SN1[1:0]		PS0[1:0]		SN0[1:0]	
RW		*RW*		*RW*		*RW*		*RW*		*RW*		*RW*		*RW*	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 30	**PS7[1:0]**	**EXTI7采样时钟预分频比选择位** 00：1/1 01：1/4 10：1/16 11：1/128
29 ~ 28	**SN7[1:0]**	**EXTI7连续采样次数选择位** 00：1 01：2 10：3 11：4
27 ~ 26	**PS6[1:0]**	**EXTI6采样时钟预分频比选择位** 00：1/1 01：1/4 10：1/16 11：1/128
25 ~ 24	**SN6[1:0]**	**EXTI6连续采样次数选择位** 00：1 01：2 10：3 11：4
23 ~ 22	**PS5[1:0]**	**EXTI5采样时钟预分频比选择位** 00：1/1 01：1/4 10：1/16 11：1/128
21 ~ 20	**SN5[1:0]**	**EXTI5连续采样次数选择位** 00：1 01：2 10：3 11：4
19 ~ 18	**PS4[1:0]**	**EXTI4采样时钟预分频比选择位** 00：1/1 01：1/4 10：1/16 11：1/128
17 ~ 16	**SN4[1:0]**	**EXTI4连续采样次数选择位** 00：1 01：2 10：3 11：4
15 ~ 14	**PS3[1:0]**	**EXTI3采样时钟预分频比选择位** 00：1/1 01：1/4 10：1/16 11：1/128
13 ~ 12	**SN3[1:0]**	**EXTI3连续采样次数选择位** 00：1 01：2 10：3 11：4

11 ~ 10	**PS2[1:0]**	**EXTI2采样时钟预分频比选择位** 00：1/1 01：1/4 10：1/16 11：1/128
9 ~ 8	**SN2[1:0]**	**EXTI2连续采样次数选择位** 00：1 01：2 10：3 11：4
7 ~ 6	**PS1[1:0]**	**EXTI1采样时钟预分频比选择位** 00：1/1 01：1/4 10：1/16 11：1/128
5 ~ 4	**SN1[1:0]**	**EXTI1连续采样次数选择位** 00：1 01：2 10：3 11：4
3 ~ 2	**PS0[1:0]**	**EXTI0采样时钟预分频比选择位** 00：1/1 01：1/4 10：1/16 11：1/128
1 ~ 0	**SN0[1:0]**	**EXTI0连续采样次数选择位** 00：1 01：2 10：3 11：4

16. 电机控制模块（Motor Control Module，MCM）

16.1 简介

SH33F2801内建电机控制PWM模块。PWM模块由时基模块（PWM Time Base Block）、波形发生模块（Wave Generator）、死区&极性控制模块（Dead&Polarity Logic）、故障检测（Fault Detect Logic）以及输出控制模块（Output Control Logic）组成，如图16-1所示。

时基模块中通过一个16位计数器（PWM Counter）对PWM时钟（PWM Clock）计数，用该计数器的值与周期寄存器和0比较以产生周期匹配信号与归零信号、与占空比寄存器比较以产生占空比匹配信号，结合波形发生模块产生原始的六路PWM信号Px_O/Px1_O，该原始PWM信号经过死区&极性控制模块后产生带死区和极性的PWM信号Px_D/Px1_D，最后的输出控制模块和故障检测模块决定是否将Px_D/Px1_D信号输出至PWM引脚PWMx/PWMx1上。

图16-1　　PWM 模块简图

16.2 主要特性

- 重要寄存器受保护寄存器PWMRLDEN控制
- 16位时基计数器
- 三种时基计数模式：边沿对齐计数、中心对齐计数和单次计数模式
- 两种波形输出模式：互补模式与独立模式
- 互补模式下提供死区控制逻辑，并支持对称和非对称两种模式
- 提供PWM周期匹配、归零匹配和占空比比较匹配中断、故障保护中断
- 3路互补PWM输出或6路独立PWM输出，输出极性可选择
- 提供故障检测功能可紧急关闭PWM输出
- 提供寄存器重载使能位以保证寄存器重载同步
- 16位预分频、8档后分频
- 可切换手动控制PWM输出
- 一个PWM时基周期内任意四个时刻触发ADC
- PWM最终输出可配置成按不同顺序映射到管脚（参见用户代码选项章节）

16.3 功能

16.3.1 PWM0/1/2 时基模块（PWM Time Base Block）

PWM模块中的时基模块（Time Base Block）主要由一个16位的计数器结合一个预分频器和一个后分频器组成。如图16-2所示，波形发生器（Wave Generator）用时基模块中16位计数器的值与占空比寄存器比较并结合死区/极性逻辑（Dead&Polarity Logic）和输出控制逻辑（Output Control）最终产生PWM波形。图16-2给出了时基模块的逻辑简图。

图16-2　　PWM 时基逻辑简图

16.3.1.1 PWM 工作时钟

PWM时钟（PWM Clock）由AHB时钟（HCLK）分频而来，通过预分频器（Pre-Counter）由寄存器PWMPSQ中决定PWM模块时钟是PCLK的(PWMPSQ+1)分频。

通过软件置位/清零PWMOE寄存器可以使能/禁止PWM时基模块。PWMOE置1后，PWM计数器（PWM Counter）开始对PWM时钟从PWMC值开始向上计数。PWMOE位清0时，预分频的内部计数器、后分频内部计数器和PWM计数器都将清0。

注1：如果为边沿模式，写入PWMC的初值大于等于PWMP的值，则PWMC先从PWMP到0，之后开始递增计数。

注2：如果为中心对齐模式，写入PWMC的初值大于PWMP的值，则PWMC从PWMP开始递减计数。

注3：PWMC在模块停止时被清零一次，之后可以写。

16.3.1.2 PWM 时基工作方式

PWM时基中16位的PWM计数器有3种工作方式，由PWMCON1寄存器中的PTMOD[1:0]位段确定。计数模式的修改需要在PWM模块使能之前完成，如在PWM模块使能后进行计数模式修改可能会得到非预期结果。

16.3.1.3 边沿对齐计数模式

设置PWMCON1中PTMOD位段为00，时基模块中的PWM计数器将工作在边沿对齐计数模式下。此模式下，软件令PWMOE位置1，PWM计数器将从PWMC值开始向上计数，直到与PWM周期寄存器PWMP匹配，然后PWM计数器复位为0并继续向上计数，如此往复。PWM计数器复位为0的时刻即归零时刻（Zero Match）。软件令PWMOE位清0，PWM Counter将停止计数并在下一个PCLK边沿复位为0并停止计数。如设置PWMP = 5，初始PWMC=0，则PWM计数器在此模式下的工作如图16-3所示。

图16-3　　边沿对齐计数模式

16.3.1.4　中心对齐计数模式

设置PWMCON1中PTMOD位段为01，时基模块中的PWM计数器将工作在中心对齐计数模式下。此模式下，软件令PWMOE位置1，PWM计数器将从PWMC值开始向上计数，直到与PWM周期寄存器PWMP匹配，然后PWM计数器开始向下计数直至归0，如此往复。PWM计数器向下计数至零的时刻即归零时刻（Zero Match），PWM计数器与周期寄存器匹配时刻为周期匹配时刻（Period Match）。软件令PWMOE位清0，PWM Counter将停止计数并在下一个PCLK边沿复位为0并停止计数。如设置PWMP = 5，初始PWMC=0，则PWM计数器在此模式下的工作如图16-4所示。

图16-4　　中心对齐计数模式

16.3.1.5　单次计数模式

设置PWMCON1中PTMOD位段为10或11，时基模块中的PWM计数器将工作在单次计数模式下。此模式下，软件令PWMOE位置1，PWM计数器将从PWMC值开始向上计数，直到与PWM周期寄存器PWMP匹配时，PWM计数器复位为0，同时PWMOE位将由硬件清0，PWM输出高阻状态。PWM计数器复位为0的时刻即归零时刻（Zero Match）。

PWM计数器向上计数过程中若软件令PWMOE位清0，则PWM Counter将在下一个PCLK边沿复位为0并停止计数。如设置PWMP = 5，初始PWMC=0，则PWM计数器在此模式下的工作如图16-5所示。

图16-5　　单次计数模式

16.3.1.6 后分频

当不需要每个周期更新PWM的占空比时，后分频将非常有用。如图16-2所示，PWM计数器运行时产生的周期匹配信号（Period Match）和归零信号（Zero Match）经过后分频计数器（Post-Counter），可以实现1/1、1/2、1/3、1/4、1/5、1/6、1/7、1/8分频。通过寄存器PWMCON1中的POSTPS位段可以选择以上八档后分频。令POSTPS = 0则禁止后分频，周期匹配信号和归零信号不会被分频。

当PWMOE置1时，PWM开始计数，同时后分频计数器也开始以周期为单位开始增计数，也即是在每个计数周期计数值归零后分频计数器自动加1，当后分频计数器等于后分频系数设置值时，后分频有效，在该计数周期内产生的周期匹配信号和归零匹配信号有效，可有效触发周期/占空比/死区/比较值寄存器重载，周期中断，归零中断以及比较值匹配触发ADC。

首先，分频后的周期匹配信号（Scaled Period Match）和归零信号（Scaled Zero Match）结合PWMRLDEN寄存器值用于控制周期寄存器PWMP、时钟预分频系数PWMPSQ[15:0]以及占空比寄存器的重载。

其次，分频后的周期匹配信号（Scaled Period Match）和归零信号（Scaled Zero Match）可用于产生归零中断和周期匹配中断，如图16-1所示。详见“时基中断”章节。

最后，PWM比较寄存器触发AD信号也受后分频系数控制。在后分频有效周期内，PWM计数值与PWM比较值相等时可触发ADC。

但是，后分频只对周期匹配信号和归零信号有效，对占空比匹配时刻无效，比如，允许占空比中断和归零信号中断的情况下，设置后分频为8分频时，归零信号进中断会在8个PWM周期后才进一次中断，而占空比匹配信号则仍然会在每个PWM周期触发中断。

16.3.1.7 时基中断

时基中断包括分频后的周期匹配信号所触发的周期中断和分频后的归零信号所触发的归零中断。分频后的周期匹配信号发生时将令寄存器PWMINTF中的PWMPIF位置1，此时若PWMINTEN中的PWMPIE位为1则将触发周期中断；分频后的归零信号发生时将令寄存器PWMINTF中的PWMZIF位置1，此时若PWMINTEN中的PWMZIE位为1则将触发归零中断。PWMPIF位和PWMZIF位需分别由软件PWMPIFC和PWMZIFC写1清除。

PWMCON1中POSTPS段等于0时，PWM计数器的周期匹配信号和归零信号不会被分频，故周期匹配信号每次发生时都将令PWMPIF位置1，归零信号每次发生时都将令PWMZIF位置1。

16.3.1.8 PWM 周期寄存器与预分频系数，后分频系数的自动重载

周期寄存器PWMP有一个缓存寄存器（BUFFER），用户能够读写PWMP但不能操作其缓存寄存器，如图16-2所示。PWM时基计数器运行时实际上是与周期寄存器的BUFFER进行比较以产生周期匹配信号。

时钟预分频系数PWMPSQ也有一个缓存寄存器（BUFFER），用户能够读写PWMPSQ但不能操作其缓存寄存器，如图16-2所示。PCLK实际上是利用预分频系数位段PWMPSQ的BUFFER进行分频以产生PWM时钟（PWM Clock）。

PWMCON1寄存器中的POSTPS位段也有一个缓存寄存器（BUFFER），用户能够读写POSTPS位段但不能操作其缓存寄存器，如图16-2所示。PCLK实际上是利用后分频系数位段POSTPS的BUFFER进行后分频操作。

由于存在缓冲寄存器，因此对周期寄存器，预分频系数和后分频系数寄存器位的读写有特殊规定：

当PWMRLDEN寄存器的值设为0x55时，才允许对这些寄存器进行修改，然后只有当PWMRLDEN寄存器中的值设为0xAA时，分频后的归零信号发生时才会将PWMP寄存器和PWMPSQ锁存至对应的缓存寄存器（BUFFER）中，这样可以避免寄存器修改过程中发生重载，也避免寄存器修改立即生效可能会出现毛刺。因此对上述寄存器的修改，必须要有对PWMRLDEN寄存器的配合操作才能完成。

注：以上3个寄存器的重载时刻不可配置，只能发生在归零匹配。

16.3.2 PWM0/1/2 波形发生模块

PWM波形发生模块指图16-1中的Wave Generator模块。波形发生模块使用PWM时基模块中PWM计数器的值与6个16位占空比值进行比较，从而产生原始的6路PWM波形：P0_O、P01_O、P1_O、P11_O、P2_O和P21_O。该原始6路PWM波形信号经过死区&极性逻辑和输出控制逻辑最终反映在6个PWM输出引脚上，如图16-1所示。

波形发生器的逻辑简图如图16-6所示。

图16-6　　波形发生器逻辑简图

16.3.2.1　PWM0/1/2 输出模式

PWM模块中包含有3个波形发生模块，对应于3对PWM输出：PWM0/PWM01、PWM1/PWM11和PWM2/PWM21，如图16-1所示。对于每对PWM输出可以分别配置为互补模式或独立模式。

16.3.2.2　PWM 互补输出模式

令PWMCON1寄存器中POUTMOD位为0，PWMx/PWMx1工作在互补状态。

当工作在对称波形输出模式下（PWMSYM@PWMCON1=0），16位PWM计数器与占空比寄存器PWMxD比较以产生Px_O和Px1_O波形，因此PWMx和PWMx1引脚上最终输出的PWM波形使用的是同一占空比寄存器PWMxD。在此输出模式下，可由寄存器PWMCON1设置PWMx/PWMx1引脚上输出波形的极性、并可由寄存器PWMDT0x和PWMDT1x设置死区（详见16.3.3节“PWM死区与极性控制模块”）。（x = 0 - 2）

当工作在非对称波形输出模式下（PWMSYM@PWMCON1=1，仅在中心对齐技术模式下有效），16位PWM计数器在增计数过程中与占空比寄存器PWMxD比较以产生Px_O和Px1_O波形，在减计数过程中与占空比寄存器PWMx1D比较以产生Px_O和Px1_O波形。在此输出模式下，可由寄存器PWMCON1设置PWMx/PWMx1引脚上输出波形的极性、并可由寄存器PWMDT0x和PWMDT1x设置死区（详见16.3.3节“PWM死区与极性控制模块”）。（x = 0 - 2）

16.3.2.3　PWM 独立输出模式

令PWMCON1寄存器中POUTMOD位为1，PWMx/PWMx1工作在独立状态，16位PWM计数器与占空比寄存器PWMxD比较以产生Px_O波形，16位PWM计数器与占空比寄存器PWMx1D的值比较以产生Px1_O波形，因此PWMx和PWMx1引脚上最终输出的PWM波形使用的是不同的占空比寄存器PWMxD与PWMx1D。在独立输出模式下，可由寄存器PWMCON1设置PWMx/PWMx1引脚上输出波形的极性、但是寄存器PWMDT0x和PWMDT1x将无效，即独立输出模式下PWM的输出无死区（详见16.3.3节“PWM死区与极性控制模块”）。（x = 0 - 2）

16.3.2.4 PWM 占空比有效区域

在各种模式下，占空比有效区域都由PDCONx@PWMCON1(x=0-2)定义。

当PDCONx=0时，边沿模式或独立模式时，计数值小于占空比寄存器定义为占空比有效区域（DutyZone），其余区域为非占空比区域（None-Duty Zone）；中心模式下，增计数时计数值小于占空比寄存器定义为占空比有效区域（DutyZone），其余区域为非占空比区域（None-Duty Zone），减计数时计数值小于等于占空比寄存器定义为占空比有效区域（DutyZone），其余区域为非占空比区域（None-Duty Zone）。

当PDCONx=1时，边沿模式或独立模式时，计数值大于等于占空比寄存器定义为占空比有效区域（DutyZone），其余区域为非占空比区域（None-Duty Zone）；中心模式下，增计数时计数值大于等于占空比寄存器定义为占空比有效区域（DutyZone），其余区域为非占空比区域（None-Duty Zone），减计数时计数值小于占空比寄存器定义为占空比有效区域（DutyZone），其余区域为非占空比区域（None-Duty Zone）。

16.3.2.5 边沿对齐计数模式下的 PWM0/1/2 原始波形

令PWMCON1中PTMOD = 00，将PWM时基设置为边沿对齐计数模式。在互补输出模式下，PWM原始输出波形Px_O和Px1_O如图16-7所示。PWMOE置1后，Px_O和Px1_O波形变高，PWM计数器与占空比寄存器PWMxD匹配后Px_O和Px1_O波形将变低直至PWM计数器归零，如此往复。

图16-7 边沿对齐计数、互补输出模式下的 PWM 原始波形

在独立输出模式下，PWM原始输出波形Px_O和Px1_O如图16-8所示。PWMOE置1后，Px_O和Px1_O波形变高，PWM计数器与占空比寄存器PWMxD匹配后Px_O波形将变低直至PWM计数器归零；PWM计数器与占空比寄存器PWMx1D匹配后Px1_O波形将变低直至PWM计数器归零，如此往复。

图16-8　　边沿对齐计数、独立输出模式下的 PWM 原始波形

16.3.2.6　中心对齐计数模式下的PWM0/1/2 原始波形

图16-9　　中心对齐计数、互补输出模式下的 PWM 原始波形（对称波形）

令PWMCON1中PTMOD = 01，将PWM时基设置为中心对齐计数模式。

在互补输出模式下，令PWMCON1中PWMSYM = 0，PWM波形发生模块工作在对称波形输出模式，PWM原始输出波形Px_O和Px1_O如图16-9所示。PWMOE置1后，Px_O和Px1_O波形保持为高，PWM计数器增计数时与PWMxD匹配后Px_O和Px1_O波形将变低直至PWM计数器减计数时再次与PWMxD匹配，如此往复。

图16-10　　中心对齐计数、互补输出模式下的 PWM 原始波形（非对称波形）

令PWMCON1中PTMOD = 01，将PWM时基设置为中心对齐计数模式。

在互补输出模式下，令PWMCON1中PWMSYM =1，PWM波形发生模块工作在非对称波形输出模式，PWM原始输出波形Px_O和Px1_O如图16-10所示。PWMOE置1后，Px_O和Px1_O波形保持为高，PWM计数器增计数时与PWMxD匹配后Px_O和Px1_O波形将变低直至PWM计数器减计数时与PWMx1D匹配，如此往复。

图16-11　　中心对齐计数、独立输出模式下的 PWM 原始波形

独立输出模式下，PWM原始输出波形Px_O和Px1_O如图16-11所示。PWMOE置1后，Px_O和Px1_O波形变高，PWM计数器增计数时与PWMxD匹配后Px_O波形将变低直至PWM计数器减计数时再次与PWMxD匹配；PWM计数器增计数时与PWMx1D匹配后Px1_O波形将变低直至PWM计数器减计数时再次与PWMx1D匹配，如此往复。

16.3.2.7　单次计数模式下的PWM0/1/2 原始波形

令PWMCON1中PTMOD = 10或11，将PWM时基设置为单次计数模式。在互补输出模式下，PWM原始输出波形Px_O和Px1_O如图16-12所示。PWMOE置1后，Px_O和Px1_O波形变高，PWM计数器增计数时与PWMxD匹配后Px_O和Px1_O波形将变低。

图16-12　单次计数、互补输出模式下的 PWM 原始波形

在独立输出模式下，PWMOE置1后，Px_O和Px1_O波形变高(PDCONx=0)，PWM计数器增计数时与PWMxD匹配后Px_O波形将变低；PWM计数器增计数时与PWMx1D匹配后Px1_O波形将变低。

16.3.2.8　占空比定义及占空比寄存器的自动重载

如图16-6所示，6个占空比寄存器都带有对应的缓存寄存器（BUFFER），16位的PWM计数器实际上是与BUFFER比较从而控制PWM原始波形的变化。用户能够读写6个占空比寄存器，却不能操作其对应的缓存寄存器。结合图16-2和图16-6可以看出。

PWMCON2寄存器中的PDCONx（x=0-2）位段也有一个缓存寄存器（BUFFER），用户能够读写PDCONx位段但不能操作其缓存寄存器，如图16-6所示。

由于存在缓存寄存器，和周期寄存器一样，对占空比定义和占空比寄存器的读写有特殊规定：

当PWMRLDEN寄存器的值设为0x55时，才允许对占空比定义和占空比寄存器进行修改，然后只有当PWMRLDEN寄存器中中的值设为0xAA时，分频后的归零信号发生时才会将占空比定义和占空比寄存器锁存至对应的缓存寄存器（BUFFER）中，这样可以避免寄存器修改过程中发生重载，也避免寄存器修改立即生效可能会出现毛刺。因此对上述寄存器的修改，必须要有对PWMRLDEN寄存器的配合操作才能完成。

另外，在中心对齐模式下，和周期寄存器不同，占空比定义和占空比寄存器除了可以用归零信号重载外，还可以用周期匹配信号重载，如果将PWMCON2中PDLDEN位置1，则分频后的周期匹配信号自动将占空比定义和6个占空比寄存器锁存至对应的缓存寄存器中（称之为重载）。因此在边沿计数模式和单次计数模式下，由于时基模块只有归零匹配信号，故PWM计数器每次复位为0时都将发生一次占空比定义和占空比寄存器的重载；中心对齐计数模式下，时基模块能产生归零匹配以及周期匹配信号，故PWM计数器每次复位为0以及与周期寄存器匹配时都可以重载占空比定义和占空比寄存器。

16.3.2.9 PWM 占空比中断

PWM0/1/2时基计数器向上计数时，不论在何种模式，当计数值与占空比寄存器PWMxD匹配时能够令PWMINTF寄存器中PTUDxIF位置1，此时若PWMINTEN寄存器中的PTUDxIE位为1，则能够触发占空比中断，如图16-7/图16-9所示。在中心对齐计数模式中，PWM时基计数器向下计数时，当计数值与PWMxD匹配时能够令PWMINTF寄存器中PTDDxIF位置1，此时若PWMINTEN寄存器中的PTDDxIE位为1，则能够触发占空比中断，如图16-9所示。

注1：互补对称输出模式和独立输出模式，占空比中断只是在PWM计数器与PWMxD相关的比较时产生，与PWMx1D相关比较匹配时不会改变占空比中断标志。

注2：互补非对称输出模式，占空比中断在递增计数时PWM计数器与PWMxD相关的比较时产生，在递减计数时PWM计数器与PWMx1D相关的比较时产生。

*注3：当占空比设为0或者周期时，*PTUDxIF位和PTDDxIF位都被置1。

16.3.2.10 PWM0/1/2 占空比寄存器更新模式

6个占空比寄存器PWMxD和PWMx1D可以实现立即更新和固定时刻（周期和归零匹配）更新。

DILDEN@PWMCON2 =1时，修改PWMxD和PWMx1D的设置值，该值重载后将直接更新到缓存寄存器。

DILDEN@PWMCON2 =0时，修改PWMxD和PWMx1D的设置值，该值更新缓存寄存器的方式受PWMCON2寄存器的PDLDEN和ZDLDEN位控制，即可以在周期匹配时更新，也可以在归零匹配时更新。

注意：当DILDEN@PWMCON2 =1时，每个PWM模块时钟都会重载并更新缓存寄存器，即修改是立即生效的。同时，如果设置了PWMCON2寄存器的PDLDEN和ZDLDEN控制位，这两个时刻同样会重载并更新缓存寄存器，即立即更新并不会屏蔽PDLDEN和ZDLDEN功能。同样的，死区寄存器和PDCONx寄存器的立即更新也是如此处理。

立即更新方式下，根据PWMxD和PWMx1D寄存器写入时刻PWM计数值与占空比寄存器新旧设置值的关系，处理结果说明如下：

图16-13 递增计数过程的 PWMxD 和 PWMx1D 寄存器更新(x=0~2)

图16-14　　递减计数过程的 PWMxD 和 PWMx1D 寄存器更新(x=0~2)

注1：立即更新在死区发生时的逻辑处理。

注2：所有的占空比寄存器都在重载之后才生效，即把PWMRLDEN写0xAA之后。

16.3.2.11　事件触发寄存器更新模式

4个PWM事件触发寄存器PWMCMPx可以实现立即更新和固定时刻（周期和归零匹配）更新。

CILDEN@PWMCON2 =1时，修改PWMCMPx的设置值，该值重载后将直接更新到缓存寄存器。

CILDEN@PWMCON2 =0时，修改PWMCMPx的设置值，该值更新缓存寄存器的方式受PWMCON2寄存器的PCMLDEN和DCMLDEN位控制。

16.3.3 PWM 死区与极性控制模块

如图16-1所示，波形发生器（Wave Generator）产生的PWM原始波形Px_O/Px1_O经过死区与极性控制模块（Dead&Polarity Logic）产生最终的PWM波形Px_D/Px1_D。带有死区与极性的PWM信号Px_D/Px1_D由输出控制模块（Output Control Logic）和故障检测模块（Fault Detect Logic）决定是否由引脚PWMx/PWMx1送出，（x = 0 - 2）。

极性控制模块的逻辑简图如图16-15所示。

图16-15　　死区与极性控制逻辑简图

16.3.3.1　PWM 死区的实现

如图16-15所示，死区逻辑通过将PWM原始信号Px_O的上升和Px1_O的下降沿延迟一段时间而实现。PWMCON1寄存器中POUTMOD位为1时，PWM原始信号Px_O/Px1_O不会经过死区逻辑，即PWMx/PWMx1引脚输出为独立模式时，无死区；PWMCON1寄存器中POUTMOD位为0时，PWM原始信号Px_O/Px1_O将经过死区逻辑，即PWMx/PWMx1引脚为互补模式时，将引入死区。死区时间由寄存器PWMDT0x和PWMDT1x决定：

上升沿延迟时间（Rising Edge Delay）= PWMDT0x * TPWM CLOCK

下降沿延迟时间（Falling Edge Delay）= PWMDT1x * TPWM CLOCK

在不同的极性设置下，插入死区后的PWM波形也将不同，详见“不同极性设置下带死区的PWM波”章节。

注意：PWMDT0x = 0时，原始PWM波形Px_O的上升沿不会触发延时；PWMDT1x = 0时，原始PWM波形Px1_O的下降沿也不会触发延时。

16.3.3.2　PWM 极性设置

为了有效说明PWM引脚上波形的极性，将一个PWM周期划分为“占空比时区（Duty Zone）”与“非占空比时区（None-Duty Zone）”，各种时基计数模式下，占空比时区和非占空比时区的划分如图16-7 - 图16-11所示。由寄存器PWMCON1中的位PWMxS/PWMx1S来决定占空比时区期间Px_D和Px1_D的电平。

PWMxS控制Px_D的极性。PWMxS = 0时，Px_D的占空比时区期间为高电平，非占空比时区期间为低电平；PWMxS = 1时，Px_D的占空比时区期间为低电平，非占空比时区期间为高电平。

PWMx1S控制Px1_D的极性。PWMx1S = 0时，Px1_D的占空比时区期间为低电平，非占空比时区期间为高电平；PWMx1S = 1时，Px1_D的占空比时区期间为高电平，非占空比时区期间为低电平。

结合图16-7 - 图16-11所示的PWM原始波形Px_O/Px1_O和图16-15可以清晰的反映死区与极性的实现。

16.3.3.3　死区和死区方式选择位段的自动重载

PWM死区也和占空比、周期一样，有自己的缓存寄存器（BUFFER），自动重载功能类似于占空比的自动重载，可以在周期匹配和归零信号发生时重载。对死区寄存器的修改也必须PWMRLDEN寄存器的值设为0x55时，然后只有当PWMRLDEN寄存器中的值设为0xAA时，重载生效。

16.3.3.4　不同极性设置下带死区的PWM 波形

图16-16给出了一对PWM输出Px_D/Px1_D设置为互补输出模式，所有的极性组合下带有死区的波形。

图16-16　边沿对齐计数模式

注：图中的DR表示上升沿死区，DF表示下降沿死区，三相上升沿和下降沿死区设置成一个值。

16.3.3.5 PWM0/1/2 输出控制与故障检测

PWM输出控制与故障检测模块（Output Control Logic & Fault Detect Logic）用于决定最终的PWM波形Px_D是否在PWMx引脚上输出、Px1_D是否在PWMx1引脚上输出。输出控制与故障检测模块的逻辑简图如图16-17所示。

图16-17 输出控制与故障检测逻辑简图

16.3.4 PWM 故障检测模式

PWM模块中有一个故障检测模块（Fault Detection logic），当FLTCON寄存器中FLT0EN、FLT1EN、FLT2EN位为1时，使能该模块功能，若FLT1EN和FLT2EN都为0，则禁止该模块功能。PWM故障检测主要目的是：当故障发生时（比如过流）可以切断PWM的输出，进入无效驱动状态（可设为高电平、低电平和高阻态），进而达到保护外部功率器件的目的，由于它是由硬件控制，因此响应速度非常快。故障检测模块的逻辑简图如图16-17所示，结合输出控制模块（Output Control Logic）能够实现前述的保护功能。

注1：故障保护在MCM模块时钟关闭时也可以正常工作。

注2：当发生保护后，如果故障恢复，则PWM输出要等周期匹配或者归零匹配时才会恢复。

16.3.4.1 故障信号输入源的选择

可以选择比较器（0/1）的输出或FLT引脚输入作为故障检测输入信号，如图16-17所示。无论故障信号来自比较器0输出、比较器1输出、MCMFLT引脚输入，都可以设置高低电平来保护PWM输出。

如图16-17所示，若FLTxEN（x = 0 - 2） = 1，则一旦对应信号变为FLTxS（x = 0 - 2）所指定的信号，并保持滤波时间（只有MCMFLT引脚才有滤波时间，比较器输出的滤波时间在比较器模块里设置），则六路PWM引脚PWMx/PWMx1立即进入无效驱动状态（x = 0 - 2）。

当使用PWM模块来控制图16-18所示的逆变桥，六个管子为高电平驱动的功率管：设置PWMxS/PWMx1S = 0（x = 0 - 2），则PWM0/1/2的输出在占空比时区期间为高电平，PWM01/11/21的输出在占空比时区期间为低电平（见“PWM极性的设置”章节），设置PWMCON1寄存器中POUTMOD位为0，，即PWM0与PWM01互补输出、PWM1与PWM11互补输出、PWM2与PWM21互补输出。使能故障检测后，若发生故障，则六路PWM进入无效驱动状态（可设为高电平、低电平和高阻态）。

图16-18 PWM 控制逆变桥示例图

16.3.4.2 FLT 引脚上的信号与比较器输出信号的滤波

当FLT2EN@FLTCON = 1时，MCMFLT引脚的输入信号会作为故障检测输入信号使用，此时可以通过设置寄存器FLTCON中的FLT2DEB[3:0]位段来调整对此信号的滤波时间。FLT2DEB[3:0] = 0时，无滤波作用。当FLT2S@FLTCON = 0时，故障检测输入信号的电平由低变高时立即触发保护；FLT2DEB[3:0] = 0 - 15时，可以将滤波时间设置为0~32us共16档，故障检测输入信号由低变高时，高电平需要至少保持FLT2DEB[3:0]定义的时间累加长度，故障检测模块才认为故障检测输入信号的电平变高了，从而触发保护；故障检测输入信号的电平由高变低时，低电平需要至少保持FLT2DEB[3:0]定义的时间累加长度，故障检测模块才认为FLT引脚上的有效故障电平消失了，六路PWM的输出根据保护模式（下节介绍）动作。当FLT2S = 1时，FLT故障检测信号变为低电平有效，情况与FLT2S = 0时相反。根据功率管的特性，恰当的设置滤波时间可以滤除故障检测输入信号上的噪声。

滤波时间设置说明如下：

1.计数器上限设置为滤波常数，下限为0，超出上下限，不执行加减动作

2. 滤波计数器初值设定:若FLT引脚设为高电平有效，计数器初始值为0；若FLT引脚设为低电平有效，则滤波计数器初值设为滤波常数。

3. 滤波器输出稳定需要一个建立时间，大致为滤波常数设置的时间，如滤波常数设置为256，则在256个系统时钟后，滤波器输出稳定，在这之前，滤波器输出不确定。

当选比较器0、1的输出信号作为故障检测输入信号使用，此时比较器0、1的输出滤波时间和算法分别在寄存器CMP0CON、CMP1CON2中设置。（详见运算放大器和比较器模块章节）

16.3.4.3　保护模式

有两种故障检测模式：锁存模式与逐次模式。

(1) 故障保护状态

故障保护后，立即进入无效电平状态，无效电平可以输出高阻态和设定的高低电平。

(2) 锁存模式

设置保护寄存器中FLTM位为0将使能锁存模式。当故障检测模块探测到有效故障信号，六路PWM引脚PWMx/PWMx1立即切换为对应FOUTx设定的无效电平状态。

若有效故障信号没有消失，六路PWM引脚PWMx/PWMx1将保持输出FLTCON.FOUTx的设定电平状态（x = 0,1），软件无法清除FLTSTAT标志位，六路PWM引脚无法恢复正常输出。

当有效故障信号消失后，六路PWM引脚PWMx/PWMx1（x = 0 - 2）亦不会恢复正常输出，只有软件将FLTSTAT位清0后，PWMx/PWMx1才会在FLTSTAT被清零之后的最近一次PWM计数器归0或与周期寄存器PWMP的值匹配时恢复正常输出（边沿对齐模式中，在PWM计数器归0时恢复输出；中心对齐模式中，在PWM计数器与周期寄存器PWMP匹配或归0时恢复输出）。

(3) 逐次模式

设置保护寄存器中FLTM位为1将使能逐次模式。在逐次模式下，PWMx/PWMx1引脚是否输出PWM波形直接由比较器滤波后的输出信号控制。FLTSTAT位也都由滤波之后的故障信号直接控制，若滤波之后的故障信号为高，则FLTSTAT位为1；若滤波之后的故障信号为低，则FLTSTAT位为0。

当故障检测模块探测到有效故障信号，六路PWM引脚PWMx/PWMx1立即输出无效电平状态（x = 0 - 2），FLTSTAT位亦会被硬件置1。

若有效故障信号没有消失，六路PWM引脚PWMx/PWMx1将保持输出无效电平状态（x = 0 - 2），软件无法使得六路PWM引脚恢复正常输出。

当有效故障信号消失后，六路PWM引脚PWMx/PWMx1（x = 0 - 2）会在有效故障信号消失后的最近一次PWM计数器归0或与周期寄存器PWMP的值匹配时自动恢复正常输出，FLTSTAT位也将归0。

注：PWM首次开启前（包含上电），PWM输出固定为高阻态。

16.3.4.4　故障保护中断

发生故障保护时，可以置起中断，进入中断程序。

16.3.5　PWM 异常保护

MCM模块可以对输出电平进行保护，当上下桥同时输出设定的有效电平时，PWM进入异常保护。另外还可以检测时钟停振，当外部晶振停振后，PWM进入异常保护。当发生异常保护时，PWM输出状态可以设定为高电平、低电平和高阻态。

16.3.5.1　PWM 输出电平异常保护

首先设定PWM保护的有效电平，在PWM输出使能寄存器（PWMOE）的允许位设定的PWM输出引脚中，当上下桥同时输出OLSGxx@POSCR设定的PWM有效电平，置起异常标志位，同时可以选择发生保护后PWM输出状态，通过FOUT0@FLTCON和FOUT1@FLTCON设定可选择输出为高电平、低电平和高阻态。如下图，设定有效电平都为低电平，输出保护后输出高阻态。

图16-19　PWM 输出电平比较

注：假如只设定3个下桥输出或者3个上桥输出，永远不会出现输出电平异常保护。

16.3.5.2　振荡器异常保护

如果检测到外部振荡器异常时（见CSM章节），置起异常标志位，同时可以选择是否将PWM输出使能寄存器（PWMOE）允许位设定的PWM输出引脚状态设为保护电平。PWM保护电平可以设定为高电平、低电平和高阻态，通过FOUT0@FLTCON和FOUT1@FLTCON设定。

16.3.5.3 异常保护恢复

系统复位、模块复位或者清除对应的保护标志位即可恢复。

注1：电平异常保护时，当处于异常电平时，即使给异常标志位写0，也不能清除标志位。

注2：异常保护在MCM模块时钟关闭时也可以正常工作。

16.3.5.4 异常保护中断

发生异常保护时，可以置起中断，进入中断程序。

16.3.6 PWM 输出控制模块

16.3.6.1 PWM 输出引脚与 IO 功能复用

PWM引脚PWMx/PWMx1与IO功能复用（x = 0 - 2），由I/O多路选择寄存器控制，具体如图16-17所示。

16.3.6.2 手动控制 PWM 输出

当PWMOE位为1时，若PMANUALCON1寄存器中的PMANUALx/PMANUALx1位为1，则PWMx/PWMx1引脚将输出PMANUALCON2寄存器中POUTx/POUTx1位的值；若PMANUALCON1寄存器中的PMANUALx/PMANUALx1位为0，则PWMx/PWMx1引脚将输出PWM波形Px_D/Px1_D。此功能和PORT口的输出功能类似，唯一的区别是，在手动控制PWM输出模式下，其输出会受到PWM故障检测输入信号控制，而在I/O输出模式下，不受该信号控制。具体如图16-17所示。

更改PMANUALCON1和PMANUALCON2寄存器时，PWMx/PWMx1引脚输出的改变与PCLK同步，输出状态改变立即生效。

16.3.6.3 时基停止时 PWM 引脚的输出

PWMOE位为0时，时基模块将停止工作，具体如图16-17所示。

16.3.7 事件触发功能

PWM 事件可以触发 ADC、寄存器的自动重载和中断的启动。

16.3.7.1 PWM0/1/2 计数器触发 ADC

若寄存器PWMCON2中的CMPx =1 （x=1-4），则在PWM计数期间当计数器值与PWMCMPx （x=1-4）设定值相等时可以自动触发一次ADC序列的转换；若寄存器PWMCON2中的CMPx = 2（x=1-4），则仅在PWM增计数期间当计数器值与PWMCMPx（x=1-4）设定值相等时可以自动触发一次ADC序列的转换（如果ADC模块中的ADCON1寄存器的ADON位、ADC_HARDCHSEL寄存器和ADC_HARDCHLH寄存器相应位被允许，将会启动一次序列转换）；若寄存器PWMCON2中的CMPx = 3 （x=1-4），则仅在PWM减计数期间当计数器值与PWMCMPx （x=1-4）设定值相等时可以自动触发一次ADC序列的转换（如果ADC模块中的ADCON1寄存器的ADON位、ADC_HARDCHSEL寄存器和ADC_HARDCHLH寄存器相应位被允许，将会启动一次序列转换）。PWM比较寄存器触发AD信号也受后分频系数控制。

16.3.7.2 比较寄存器 x 及其启动 AD 控制位(PWMCON2.CMPx[1:0])自动重载

和周期及占空比寄存器一样，比较值寄存器也有对应的缓存寄存器（BUFFER）。

当PWMRLDEN寄存器的值设为0x55时，才允许对比较值寄存器进行修改，然后只有当PWMRLDEN寄存器中中的值设为0xAA时，分频后的归零信号发生时才会将比较值寄存器锁存至对应的缓存寄存器（BUFFER）中，这样可以避免寄存器修改过程中发生重载，也避免寄存器修改立即生效可能会出现毛刺。因此对上述寄存器的修改，必须要有对PWMRLDEN寄存器的配合操作才能完成。

比较值寄存器可以选择是否用归零信号重载，如果将PWMCON2中ZCMLDEN位为1，则分频后的归零匹配信号也会将硬件自动将比较值寄存器锁存至对应的缓存寄存器中（称之为重载）。

另外，在中心对齐模式下，比较值寄存器除了可以用归零信号重载外，还可以用周期匹配信号重载，如果将PWMCON2中PCMLDEN位为1，则分频后的周期匹配信号也会将硬件自动将比较值寄存器锁存至对应的缓存寄存器中（称之为重载）。

因此在边沿计数模式和单次计数模式下，由于时基模块只有归零匹配信号，故PWM计数器每次复位为0时都将发生一次比较值寄存器的重载；中心对齐计数模式下，时基模块能产生归零匹配以及周期匹配信号，故PWM计数器每次复位为0以及与周期寄存器匹配时都可以重载比较值寄存器。

比较器事件触发设置位PWMCON2的CMPx(x=1-4)的重载也受归零信号或周期匹配信号控制。

16.3.7.3 PWM 模块的中断总汇

PWM模块的中断包括时基模块产生的PWM计数器归零中断、PWM计数器值与周期寄存器匹配产生的周期中断（详见“时基中断”章节）以及PWM计数器值与占空比寄存器PWMxD（x = 0 - 2）匹配时产生的占空比中断（详见“占空比中断”章节）。

16.3.8 PWM 占空比饱和功能

饱和模块具有比较输出功能，当设定饱和功能使能（STATEN@PSCON为1）时，写入PWMxD（x=0,1,2,01,11,21）寄存器的值（缓存完之后），开启启动位，功能完成后启动位自动清零。首先比较PWMxD（x=0,1,2,01,11,21）和PWMDMAX寄存器的值，通过设定PSCON寄存器的[MaxSelect]选择位来选择比较输出PWMxD（x=0,1,2,01,11,21）的值。之后比较PWMxD（x=0,1,2,01,11,21）和PWMDMIN两个寄存器的值，通过设定MINSELECT@PSCON选择位来选择比较输出的值。具体实现方式如下面框图：

图16-20　　饱和功能图

16.3.9 PWM 相移功能

在电机控制单电阻方案中，当给定三个占空比中两个相邻大小占空比的差值小于一定值时，无法实现两相电流采样。通过PWM相移功能，可以自动完成相移，从而满足两相电流采样的所需时间。开启相移启动位，功能完成后启动位自动清零。

系统框图如下：

图16-21　　PWM 相移功能图

注：Max/Med/Min值来自PWMxD（x = 0、1、2），经过相移后（SHIFT MODULE），Max/Med/Min赋值给PWMxD（x = 0、1、2），Max1/Med1/Min1赋值给PWMx1D（x = 0、1、2）。

通过START@PSCON启动位来启动相移功能，首先根据象限区分最大值（Max）、中间值（Med）和最小值（Min）占空比，之后经过移位模块处理，得到移位后的三个占空比值，再根据象限关系给定非对称PWM的两个占空比寄存器PWMxD和PWMx1D，输出非对称PWM波实现相移功能。

16.3.9.1　　象限判断功能

电机转动到不同象限，PWM三相占空比（PWM0D、PWM1D和PWM2D）的大小关系不一样，所以要根据三相大小关系输出SECTOR@PSCON的值，对应关系如图16-21所示。通过大小顺序占空比寄存器Buffer来给定比较结果BUFFER的值，所有这些数据都输出到移位模块（Shift Module）中。

16.3.9.2　移位模块

移位模块主要是通过判断比较结果BUFFER值和比较寄存器值大小来输出不同的非对称占空比，实现框图如下：

图16-22　PWM 相移实现总图

图16-23　　PWM 相移实现图 Part1

图16-24　　PWM 相移实现图 Part2

16.3.9.3 饱和相移应用示例

设定：MCM的时钟频率为60Mhz，空调压缩机载波频率为6K，则PWM周期寄存器PWMP = 5000，输出范围限定在(PWMDMIN~PWMDMAX)范围内，设定MAXSELECT[1:0]@PSCON=01b，MINSELECT[1:0]@PSCON=01b，PWMDMAX = 4500，PWMDMIN = 500，PWMDCMP1 = PWMDCMP2 =500，电机当前运行的象限Sector = 4，三相占空比分别为PWM0D = 4200，PWM1D = 4800，PWM2D = 300。

首先开启饱和相移功能，设置PSCON |= 0x101，之后中间过程的各个寄存器和BUFFER值为：经过饱和输出为PWM0D = 4200，PWM1D = 4500，PWM2D = 500，之后Max = 4500，Med = 4200，Min = 500，CMP1 = 300，CMP2 = 3700。经过移位模块后，Max = 4500，Max1 = 4500，Med = 4000，Med1 = 4400，Min = 500，Min1 = 500，然后再通过象限选择六个PWM占空比寄存器分别为，PWM0D = 4000，PWM01D = 4400，PWM1D = 4500，PWM11D = 4500，PWM2D = 500，PWM21D = 500。

原始波形如下：

图16-25 PWM 饱和相移前波形

经过饱和相移后波形如下：

图16-26　　PWM 饱和相移后波形

注1：饱和与相移的启动方式均为软件写启动位，计算完成后硬件自动清零。

注2：如果饱和和相移同时启动，则先做饱和处理，结束之后做相移处理。

注3：寄存器重载是在PWMRLDEN写0xAA后，同时饱和和相移的启动位为0时才重载，如果饱和或者相移正在运行，则要等运行结束才重载。

16.3.10 PWM 逐波限流功能

16.3.10.1 功能介绍

PWM总共有3对输出口，每对输出都可以根据对应的相线电流来实现单独的限流保护。如果对应的SCx引脚的保护电平有效，则进入逐步限流，逐波限流后的无电平则根据FOUT0[1:0]@FLTCON和FOUT1[1:0]@FLTCON来设置。

注：逐波限流功能与故障检测模块可以同时开启。

16.3.10.2 逐波限流应用示例

设定：MCM1控制电机，三相互补带死区输出PWM0、PWM1和PWM2，无效电平都为低电平。SC0保护PWM0，保护时间设定为2400个PLCK2(40us)；SC1保护PWM1，保护时间设定为3600个PLCK2(60us)；SC2保护PWM2，保护时间设定为4800个PLCK2(80us)。示例波形中的保护时间内，PWM输出的粗细虚线部分为如果没有SC保护的波形，实线为保护之后实际波形。

图16-27 逐波限流应用示例

逐步限流SCx引脚滤波配置说明：

1.计数器上限设置为常数的2倍，下限为0，超出上下限，不执行加减动作

2. 滤波计数器初值设定:若逐步限流引脚置为高电平有效，计数器初始值为0；若逐步限流引脚置为低电平有效，则滤波计数器初值设为滤波常数x2。

3. 滤波器输出稳定需要一个建立时间，大致为滤波常数设置的时间，如滤波常数设置为256，则在256个系统时钟后，滤波器输出稳定，在这之前，滤波器输出不确定。

16.3.11 PWM 寄存器的写保护

16.3.11.1 普通寄存器的写保护和重载

PWM模块的寄存器，除了中断标志寄存器PWMINTF、手动控制寄存器PMANUALCON2、FLTCON、POSCR和POSTDCR外，其它寄存器的修改有限制条件，只有当PWMRLDEN = 0x55时才允许软件修改，否则修改无效。

PWM模块中有一些寄存器带有缓存寄存器（BUFFER），包括占空比寄存器、周期寄存器、PWM时钟预分频寄存器PWMPSQ、死区寄存器和PWM事件触发寄存器，只有当PWMRLDEN = 0xAA时缓存寄存器才会被重载。

注：带缓存的寄存器的重载时间点如下：

1、 固定为归零重载的寄存器：周期寄存器 PWMP、PWM 时钟预分频寄存器 PWMPSQ、后分频寄存器 POSTPS[2:0]@PWMCON1。

2、 重载点为归零和周期可设的寄存器：死区寄存器 PWMDTxx(xx=00,10,20,01,11,21)、Duty 选择寄存器 PDCONx@PWMCON1(x=0,1,2)、PWMSYM@PWMCON1、CMPx@PWMCON2(x=1~4)。

3、 重载点为归零、周期和立即更新可设的寄存器：空比寄存器 PWMxD 和 PWMx1D(x=0,1,2)、PWM 事件触发寄存器 PWMCMPx(x=1,2,3,4)。

16.3.11.2 保护寄存器的写保护

PWM模块的保护寄存器，故障检测寄存器FLTCON、异常保护寄存器POSCR和POSTDCR的写入有限制条件，只有在FLTWEN = 0x33CC时才允许软件修改，否则修改无效。

16.3.12 注意事项

16.3.12.1　边沿对齐计数模式中占空比寄存器值为0 或周期值

边沿对齐计数模式中，若占空比寄存器为0或周期值，则根据不同的极性设置，PWMx/PWMx1应输出占空比为0%或100%的波形。令PWMOE = 1、PWMDT0x = 0、PWMDT1x = 0在边沿对齐时基计数、互补输出模式下，不同占空比值的PWM引脚输出的波形、中断标志触发位置、事件触发位置、寄存器重载位置如图16-28 - 图16-30所示。

注意：PWM时基Enable时刻，虽然时基计数器为0，但不会产生归零（zero match）信号，因此不会置PWMZIF标志，也不会产生事件触发信号，而寄存器重载信号在PWMOE= 0时是立即发生的。

图16-28　　边沿对齐计数、互补输出模式

图16-29　　边沿对齐计数、互补输出模式，占空比 = 0

图16-30　　边沿对齐计数、互补输出模式，占空比 = 周期寄存器

16.3.12.2 中心对齐计数模式中占空比寄存器值为0或周期值

中心对齐计数模式中，若占空比寄存器为0或周期值，则根据不同的极性设置，PWMx/PWMx1应输出占空比为0%或100%的波形。令PWMOE = 1、PWMDT0x = 0、PWMDT1x = 0在边沿对齐时基计数、互补输出模式下，不同占空比值的PWM引脚输出的波形、中断标志触发位置、事件触发位置、寄存器重载位置如图16-31 - 图16-33所示。

图16-31　　中心对齐计数、互补输出模式

图16-32　　中心对齐计数、互补输出模式、占空比 = 0

图16-33　中心对齐计数、互补输出模式、占空比 = 周期

16.3.12.3 周期寄存器值为0

不管时基计数器在何种模式，若周期寄存器为0，PWM输出高阻态。此时，所有中断标志都不会产生，也不会产生事件触发信号，但重载信号会产生。

16.3.12.4 仿真状态的PWM 输出

支持在线仿真，在仿真状态运行和正常状态运行没有区别，如果在仿真状态停止运行或单步运行，PWM输出会切换为高阻态，保证不会误触发外部功率管。

16.3.12.5 进入 IDLE 模式，PWM0/1/2 的波形输出状态

进入IDLE模式后，PWM输出口不受影响，比如PWM0口原先输出1K的方波，进入IDLE模式后会继续输出1K的方波，但是如果设置了PWM0/1/2中断，会唤醒IDLE模式，继续执行IDLE后面的程序。

16.3.12.6 进入 STOP 模式，PWM0/1/2 的波形输出状态

进入STOP模式时，如果MCM模块输出使能，则关闭PWMOE，PWM输出口会输出高阻状态，即使设置了PWM0/1/2中断，也不能唤醒掉电模式，掉电模式必须通过其它方式唤醒，具体参考电源管理章节。 在退出STOP模式后需软件打开PWMOE。

注：为避免在从STOP模式下恢复后状态不确定，建议在进入STOP之间软件关闭PWM，在退出STOP后重新配置和打开PWM。

16.4 寄存器

MCM 模块寄存器列表 (基地址：0x4004 0400)

地址	寄存器名	说明
0x4004 0400	PWMOE	PWM 输出使能寄存器
0x4004 0404	PWMCON1	PWM 模块控制寄存器 1
0x4004 0408	PWMCON2	PWM 模块控制寄存器 2
0x4004 040C	PWMP	PWM 周期寄存器
0x4004 0410	PWMC	PWM 计数寄存器
0x4004 0414	PWMPSQ	PWM 时钟预分频寄存器
0x4004 0418	PWM0D	PWM0 占空比寄存器
0x4004 041C	PWM1D	PWM1 占空比寄存器
0x4004 0420	PWM2D	PWM2 占空比寄存器
0x4004 0424	PWM01D	PWM01 占空比寄存器
0x4004 0428	PWM11D	PWM11 占空比寄存器
0x4004 042C	PWM21D	PWM21 占空比寄存器
0x4004 0430	PWMCMP1	事件触发比较寄存器 1
0x4004 0434	PWMCMP2	事件触发比较寄存器 2
0x4004 0438	PWMCMP3	事件触发比较寄存器 3
0x4004 043C	PWMCMP4	事件触发比较寄存器 4
0x4004 0440	PWMDT00	PWM 通道 0 上升沿死区控制寄存器
0x4004 0444	PWMDT01	PWM 通道 0 下降沿死区控制寄存器
0x4004 0448	PWMDT10	PWM 通道 1 上升沿死区控制寄存器
0x4004 044C	PWMDT11	PWM 通道 1 下降沿死区控制寄存器
0x4004 0450	PWMDT20	PWM 通道 2 上升沿死区控制寄存器
0x4004 0454	PWMDT21	PWM 通道 2 下降沿死区控制寄存器
0x4004 0458	PMANUALCON1	PWM0/1/2 手动输出设置寄存器 1
0x4004 045C	PMANUALCON2	PWM0/1/2 手动输出设置寄存器 2
0x4004 0460	FLTCON	FLT 引脚故障检测保护 PWM0/1/2 寄存器
0x4004 0464	POSCR	PWM0/1/2 输出电平保护控制寄存器
0x4004 0470	POSTDCR	PWM0/1/2 振荡器停止检测保护控制位
0x4004 0474	PWMINTEN	PWM 中断使能控制寄存器
0x4004 0478	PWMINTF	PWM 中断标志和清除寄存器
0x4004 047C	PWMRLDEN	寄存器修改和重载控制寄存器
0x4004 0480	PSCON	PWM 饱和/相移控制寄存器
0x4004 0484	PWMDMAX	PWM 占空比上限比较寄存器
0x4004 0488	PWMDMIN	PWM 占空比下限比较寄存器
0x4004 048C	PWMDCMP1	PWM 占空比移相最小采样间隔寄存器 1
0x4004 0490	PWMDCMP2	PWM 占空比移相最小采样间隔寄存器 2
0x4004 0494	SC0CON	逐波限流引脚 0 功能选择寄存器
0x4004 0498	SC1CON	逐波限流引脚 1 功能选择寄存器
0x4004 049C	SC2CON	逐波限流引脚 2 功能选择寄存器
0x4004 04A0	FLTWEN	寄存器修改和重载保护控制寄存器

16.4.1 PWM 输出使能寄存器 (MCM_PWMOE)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留															PWMOE
-															*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 1	保留	-
0	**PWMOE**	**MCM 时基使能控制位（注：PWM0/1/2 共用一个时基）** 0：关闭 MCM（PWM0/1/2）时基 1：打开 MCM（PWM0/1/2）时基

16.4.2 PWM 模块控制寄存器 1 (MCM_PWMCON1)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
POUTMOD	POSTPS[2:0]			PTMOD[1:0]		PWMSYM	PDCON2	PDCON1	PDCON0	PWM21S	PWM11S	PWM01S	PWM2S	PWM1S	PWM0S
RW	*RW*			*RW*		*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15	**POUTMOD**	**PWM0/1/2 输出模式（独立/互补）控制位** 0：整个模块配置为 3 通道互补输出 1：整个模块配置为 6 路独立输出
14 ~ 12	**POSTPS[2:0]**	**后分频系数选择位段** 000：无后分频功能 001：中断、重载信号与事件触发信号将 2 分频 010：中断、重载信号与事件触发信号将 3 分频 011：中断、重载信号与事件触发信号将 4 分频 100：中断、重载信号与事件触发信号将 5 分频 101：中断、重载信号与事件触发信号将 6 分频 110：中断、重载信号与事件触发信号将 7 分频 111：中断、重载信号与事件触发信号将 8 分频

11 ~ 10	**PTMOD[1:0]**	**PWM0/1/2 时基模块中 PWM 计数器工作模式选择位段** 00：边沿对齐计数模式，在此模式下，时基计数器只会发生归零匹配 01：中心对齐计数模式，在此模式下，时基计数器会发生归零匹配和周期匹配 1x：单次计数模式，在此模式下，时基计数器只会发生归零匹配
9	**PWMSYM**	**PWM 波形对称性控制位（仅在中心对齐计数模式的互补输出时有效）** 0：互补输出对称 PWM 波形 1：互补输出非对称 PWM 波形
8	**PDCON2**	**PWM2/PWM21 原始波形 DUTY 区控制位** 0：计数器值小于占空比寄存器时定义为 DUTY 区 1：计数器值大于等于占空比寄存器时定义为 DUTY 区
7	**PDCON1**	**PWM1/PWM11 原始波形 DUTY 区控制位** 0：计数器值小于占空比寄存器时定义为 DUTY 区 1：计数器值大于等于占空比寄存器时定义为 DUTY 区
6	**PDCON0**	**PWM0/PWM01 原始波形 DUTY 区控制位** 0：计数器值小于占空比寄存器时定义为 DUTY 区 1：计数器值大于等于占空比寄存器时定义为 DUTY 区
5	**PWM21S**	**PWM21 输出有效电平（Active Polar）选择位** 0：PWM21 在 Duty 区输出低电平，其余时间输出高电平 1：PWM21 在 Duty 区输出高电平，其余时间输出低电平
4	**PWM11S**	**PWM11 输出有效电平（Active Polar）选择位** 0：PWM11 在 Duty 区输出低电平，其余时间输出高电平 1：PWM11 在 Duty 区输出高电平，其余时间输出低电平
3	**PWM01S**	**PWM01 输出有效电平（Active Polar）选择位** 0：PWM01 在 Duty 区输出低电平，其余时间输出高电平 1：PWM01 在 Duty 区输出高电平，其余时间输出低电平
2	**PWM2S**	**PWM2 输出有效电平（Active Polar）选择位** 0：PWM2 在 Duty 区输出高电平，其余时间输出低电平 1：PWM2 在 Duty 区输出低电平，其余时间输出高电平
1	**PWM1S**	**PWM1 输出有效电平（Active Polar）选择位** 0：PWM1 在 Duty 区输出高电平，其余时间输出低电平 1：PWM1 在 Duty 区输出低电平，其余时间输出高电平
0	**PWM0S**	**PWM0 输出有效电平（Active Polar）选择位** 0：PWM0 在 Duty 区输出高电平，其余时间输出低电平 1：PWM0 在 Duty 区输出低电平，其余时间输出高电平

16.4.3 PWM 模块控制寄存器 2 (MCM_PWMCON2)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留	PCMLDEN	ZCMLDEN	PDLDEN	ZDLDEN	CILDEN	DILDEN	OSYNC	CMP4[1:0]		CMP3[1:0]		CMP2[1:0]		CMP1[1:0]	
-	RW	RW	RW	RW	RW	RW	RW	RW		RW		RW		RW	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 15	保留	-
14	**PCMLDEN**	**PWM 计数器周期匹配时重载事件触发比较寄存器（PWMCMPx，x=1,2,3,4）、启动 AD 方式寄存器(CMP1~4[1:0])使能位（只在中心对齐模式下有效）** 0：周期匹配时刻不允许重载事件触发寄存器（PWMCMPx，x=1,2,3,4） 1：周期匹配时刻允许重载事件触发寄存器（PWMCMPx，x=1,2,3,4）
13	**ZCMLDEN**	**PWM 计数器归零匹配时重载事件触发比较寄存器（PWMCMPx，x=1,2,3,4）、启动 AD 方式寄存器(CMP1~4[1:0])使能位** 0：归零匹配时刻不允许重载事件触发寄存器（PWMCMPx，x=1,2,3,4） 1：归零匹配时刻允许重载事件触发寄存器（PWMCMPx，x=1,2,3,4）
12	**PDLDEN**	**PWM 计数器周期匹配时重载占空比和死区寄存器使能位（只在中心对齐模式下有效）** 0：周期匹配时刻不允许重载占空比寄存器和死区寄存器 1：周期匹配时刻允许重载占空比寄存器和死区寄存器 注：PWM 周期寄存器、后分频寄存器和预分频寄存器这三个寄存器的重载不可配置，只能发生在归 0 匹配
11	**ZDLDEN**	**PWM 计数器归零匹配时重载占空比和死区寄存器使能位** 0：归零匹配时刻不允许重载占空比寄存器和死区寄存器 1：归零匹配时刻允许重载占空比寄存器和死区寄存器 注： PWM 周期寄存器、后分频寄存器和预分频寄存器这三个寄存器在归 0 匹配时刻都会自动重载，不受软件控制
10	**CILDEN**	**PWM 事件触发比较寄存器（PWMCMPx，x=1,2,3,4）立即生效使能位：** 0：与 PWM 周期同步后生效 1：立即生效
9	**DILDEN**	**PWM 占空比寄存器立即生效使能位** 0：与 PWM 周期同步后生效 1：立即生效
8	**OSYNC**	**手动修改 PWMx/PWMx1 输出同步位:（手动修改通过设置 PMANUALCON1/2 寄存器实现,任一寄存器的修改,何时生效都由此寄存器控制）** 0：手动修改后立即生效 1：手动修改与 PWM 周期同步后生效
7 ~ 6	**CMP4[1:0]**	**比较寄存器 4 启动 AD 控制位** 00：不触发启动 AD 01：只要 PWM 计数器与比较寄存器 4 相等时会触发启动 AD 10：只有 PWM 计数器在增计数过程中与比较寄存器 4 相等时会触发启动 AD（仅在 PTMOD = 01 中心对齐模式下有效） 11：只要 PWM 计数器在减计数过程中与比较寄存器 4 相等时会触发启动 AD（仅在 PTMOD = 01，中心对齐模式下有效）
5 ~ 4	**CMP3[1:0]**	**比较寄存器 3 启动 AD 控制位** 00：不触发启动 AD 01：只要 PWM 计数器与比较寄存器 3 相等时会触发启动 AD 10：只有 PWM 计数器在增计数过程中与比较寄存器 3 相等时会触发启动 AD（仅在 PTMOD = 01 中心对齐模式下有效） 11：只要 PWM 计数器在减计数过程中与比较寄存器 3 相等时会触发启动 AD（仅在 PTMOD = 01，中心对齐模式下有效）

3 ~ 2	**CMP2[1:0]**	**比较寄存器 2 启动 AD 控制位** 00：不触发启动 AD 01：只要 PWM 计数器与比较寄存器 2 相等时会触发启动 AD 10：只有 PWM 计数器在增计数过程中与比较寄存器 2 相等时会触发启动 AD（仅在 PTMOD = 01 中心对齐模式下有效） 11：只要 PWM 计数器在减计数过程中与比较寄存器 2 相等时会触发启动 AD（仅在 PTMOD = 01，中心对齐模式下有效）
1 ~ 0	**CMP1[1:0]**	**比较寄存器 1 启动 AD 控制位** 00：不触发启动 AD 01：只要 PWM 计数器与比较寄存器 1 相等时会触发启动 AD 10：只有 PWM 计数器在增计数过程中与比较寄存器 1 相等时会触发启动 AD（仅在 PTMOD = 01 中心对齐模式下有效） 11：只要 PWM 计数器在减计数过程中与比较寄存器 1 相等时会触发启动 AD（仅在 PTMOD = 01，中心对齐模式下有效）

16.4.4 PWM 周期寄存器 (MCM_PWMP)

偏移地址: 0x000C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMP[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	**-**
15 ~ 0	**PWMP[15:0]**	**PWM 周期寄存器**

16.4.5 PWM 计数寄存器 (MCM_PWMC)

偏移地址: 0x0010

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMC[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMC[15:0]**	**PWM 计数寄存器** 注 1：PWMC 只有在停止计数时可以写，其他时间只能读 注 2：PWMC 在模块停止时被清零一次，之后可以写。 注 3：如果模块停止信号和 PWMC 写信号同时来了，则模块停止信号优先级高

16.4.6 PWM 时钟预分频寄存器 (MCM_PWMPSQ)

偏移地址: 0x0014

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMPSQ[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMPSQ[15:0]**	**PWM 时钟为(HCLK/(PWMPSQ+1))**

16.4.7 PWM0/1/2 占空比寄存器 (MCM_PWMnD)(n=0..2)

偏移地址: 0x0018
:0x001C
:0x0020
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMnD[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMnD[15:0]**	**PWMn 占空比寄存器**

16.4.8 PWM01/11/21 占空比寄存器 (MCM_PWMn1D)(n=0..2)

偏移地址: 0x0024
:0x0028
:0x002C
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMnD[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMn1D[15:0]**	**PWMn1 占空比寄存器**

16.4.9 事件触发比较寄存器 1 (MCM_PWMCMPn)(n=1..4)

偏移地址: 0x0030
:0x0034
:0x0038
:0x003C
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMCMPn[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMCMPn[15:0]**	事件触发比较寄存器 n

16.4.10 PWM 通道 0 上升沿死区控制寄存器 (MCM_PWMDT00)

偏移地址: 0x0040
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMDT00[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMDT00[15:0]**	PWM 通道 0 上升沿死区控制寄存器

16.4.11 PWM 通道 0 下降沿死区控制寄存器 (MCM_PWMDT01)

偏移地址: 0x0044
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMDT01[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	**-**
15 ~ 0	**PWMDT01[15:0]**	**PWM 通道 0 下降沿死区控制寄存器**

16.4.12 PWM 通道 1 上升沿死区控制寄存器 (MCM_PWMDT10)

偏移地址: 0x0048
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMDT10[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	**-**
15 ~ 0	**PWMDT10[15:0]**	**PWM 通道 1 上升沿死区控制寄存器**

16.4.13 PWM 通道 1 下降沿死区控制寄存器 (MCM_PWMDT11)

偏移地址: 0x004C
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMDT11[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMDT11[15:0]**	**PWM 通道 1 下降沿死区控制寄存器**

16.4.14 PWM 通道 2 上升沿死区控制寄存器 (MCM_PWMDT20)

偏移地址: 0x0050
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMDT20[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMDT20[15:0]**	**PWM 通道 2 上升沿死区控制寄存器**

16.4.15 PWM 通道 2 下降沿死区控制寄存器 (MCM_PWMDT21)

偏移地址: 0x0054
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMDT21[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMDT21[15:0]**	**PWM 通道 2 下降沿死区控制寄存器**

16.4.16 PWM0/1/2 手动输出设置寄存器 1 (MCM_PMANUALCON1)

偏移地址: 0x0058

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留										PMANUAL21	PMANUAL11	PMANUAL01	PMANUAL2	PMANUAL1	PMANUAL0
-										RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 6	保留	-
5	**PMANUAL21**	**PWM21 口输出控制位** 0：PWM21 口输出 PWM 波形 1：PWM21 口输出由 PMANUALCON2 寄存器中的 POUT21 位控制
4	**PMANUAL11**	**PWM11 口输出控制位** 0：PWM11 口输出 PWM 波形 1：PWM11 口输出由 PMANUALCON2 寄存器中的 POUT11 位控制
3	**PMANUAL01**	**PWM01 口输出控制位** 0：PWM01 口输出 PWM 波形 1：PWM01 口输出由 PMANUALCON2 寄存器中的 POUT01 位控制
2	**PMANUAL2**	**PWM2 口输出控制位** 0：PWM2 口输出 PWM 波形 1：PWM2 口输出由 PMANUALCON2 寄存器中的 POUT2 位控制
1	**PMANUAL1**	**PWM1 口输出控制位** 0：PWM1 口输出 PWM 波形 1：PWM1 口输出由 PMANUALCON2 寄存器中的 POUT1 位控制
0	**PMANUAL0**	**PWM0 口输出控制位** 0：PWM0 口输出 PWM 波形 1：PWM0 口输出由 PMANUALCON2 寄存器中的 POUT0 位控制

16.4.17 PWM0/1/2 手动输出设置寄存器 2 (MCM_PMANUALCON2)

偏移地址: 0x005C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留										POUT21	POUT11	POUT01	POUT2	POUT1	POUT0
-										RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 6	保留	-
5	**POUT21**	**PMANUAL21 = 1 时，决定 PWM21 口的输出电平** 0：输出 0 1：输出 1
4	**POUT11**	**PMANUAL11 = 1 时，决定 PWM11 口的输出电平** 0：输出 0 1：输出 1
3	**POUT01**	**PMANUAL01 = 1 时，决定 PWM01 口的输出电平** 0：输出 0 1：输出 1
2	**POUT2**	**PMANUAL2 = 1 时，决定 PWM2 口的输出电平** 0：输出 0 1：输出 1
1	**POUT1**	**PMANUAL1 = 1 时，决定 PWM1 口的输出电平** 0：输出 0 1：输出 1
0	**POUT0**	**PMANUAL0 = 1 时，决定 PWM0 口的输出电平** 0：输出 0 1：输出 1 注意：首次配置 PMANUALCON1 寄存器时，必须先配置 PMANUALCON2 寄存器，后配置 PMANUALCON1 寄存器。

16.4.18 FLT 引脚故障检测保护 PWM0/1/2 寄存器 (MCM_FLTCON)

偏移地址: 0x0060

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留								FOUT1[1:0]		FOUT0[1:0]		保留	FLT0 STATC	FLT1 STATC	FLT2 STATC
-								RW		RW		-	WO	WO	WO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
FLT0 EN	FLT0 S	FLT0 M	FLT0 STAT	FLT1 EN	FLT1 S	FLT1 M	FLT1 STAT	FLT2DEB[3:0]				FLT2 EN	FLT2 S	FLT2 M	FLT2 STAT
RW	RW	RW	RO	RW	RW	RW	RO	RW				RW	RW	RW	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 24	保留	-
23 ~ 22	**FOUT1[1:0]**	**故障发生后，PWMx1 引脚（下桥三个引脚）输出状态选择位：** 0x：输出高阻态 10：输出低电平 11：输出高电平
21 ~ 20	**FOUT0[1:0]**	**故障发生后，PWMx 引脚（上桥三个引脚）输出状态选择位：** 0x：输出高阻态 10：输出低电平 11：输出高电平
19	保留	-

18	**FLT0STATC**	**FLT0 故障检测标志位清除位** 0：无效 1：清除
17	**FLT1STATC**	**FLT1 故障检测标志位清除位** 0：无效 1：清除
16	**FLT2STATC**	**FLT2 故障检测标志位清除位** 0：无效 1：清除
15	**FLT0EN**	**比较器 0 保护 MCM 控制位** 0：故障检测 1 功能禁止 1：故障检测 1 功能打开
14	**FLT0S**	**故障检测 0 有效电平选择位** 0：故障检测 0 CMP0 输出高电平有效 1：故障检测 0 CMP0 输出低电平有效
13	**FLT0M**	**检测功能模式选择** 0：锁存模式，当检测到故障输入有效，FLTSTAT 被硬件置 1，PWM 将立即停止输出，当故障输入变为无效时状态保持不变，只有当 FLTSTAT 被软件清零后，PWM 波形才会在 PWM 时基计数器归 0 时刻或与周期寄存器 PWMP 的值匹配时恢复输出。（当故障输入一直有效时 FLTSTAT 无法被软件清除） 1：逐次模式，PWM 输出直接由故障检测输入端来控制，如果故障输入有效，FLTSTAT 被硬件置 1，立即关闭 PWM 输出。如果故障输入变为无效，FLTSTAT 自动清 0，PWM0/1/2 波形将在 PWM 时基计数器归 0 时刻自动恢复输出
12	**FLT0STAT**	**故障检测标志位** 0：PWM0/1/2 模块处于正常输出状态 1：出现过流，PWM0/1/2 模块处于停止输出状态，如果在锁存模式，可软件清 0，恢复 PWM 输出 注：FLTSTAT 与 FLTIF 的区别为逐次模式下，FLTSTAT 会被清零
11	**FLT1EN**	**比较器 1 保护 MCM 控制位** 0：故障检测 1 功能禁止 1：故障检测 1 功能打开
10	**FLT1S**	**故障检测 1 有效电平选择位** 0：故障检测 1 CMP1 输出高电平有效 1：故障检测 1 CMP1 输出低电平有效
9	**FLT1M**	**检测功能模式选择** 0; 锁存模式，当检测到故障输入有效，FLTSTAT 被硬件置 1，PWM 将立即停止输出，当故障输入变为无效时状态保持不变，只有当 FLTSTAT 被软件清零后，PWM 波形才会在 PWM 时基计数器归 0 时刻或与周期寄存器 PWMP 的值匹配时恢复输出。（当故障输入一直有效时 FLTSTAT 无法被软件清除） 1; 逐次模式，PWM 输出直接由故障检测输入端来控制，如果故障输入有效，FLTSTAT 被硬件置 1，立即关闭 PWM 输出。如果故障输入变为无效，FLTSTAT 自动清 0，PWM0/1/2 波形将在 PWM 时基计数器归 0 时刻自动恢复输出
8	**FLT1STAT**	**故障检测标志位** 0：PWM0/1/2 模块处于正常输出状态 1：出现过流，PWM0/1/2 模块处于停止输出状态，如果在锁存模式，可软件清 0，恢复 PWM 输出 注：FLTSTAT 与 FLTIF 的区别为逐次模式下，FLTSTAT 会被清零

7 ~ 4	**FLT2DEB[3:0]**	**故障检测 2 输入滤波参数选择** 0000：无滤波 0001：0.5us 0010：1us 0011：1.5us 0100：2us 0101：3us 0110：4us 0111：6us 1000：8us 1001：10us 1010：12us 1011：14us 1100：16us 1101：20us 1110：24us 1111：32us 注 1：上述滤波常数时间不是精确值，仅供参考。 注 2：滤波器算法说明：用 HSI 时钟采样滤波器输入。采用数字积分，如果采样结果为高电平，计数器加 1，计数器结果超出设定的常数，则滤波器输出 1 同时计数器置为滤波常数；如果采样结果为低电平，计数器减 1，计数器结果小于滤波器常数,则滤波器输出 0 同时计数器置为 0。
3	**FLT2EN**	**故障检测 2 功能控制位（注 1）** 0：故障检测 2 功能禁止 1：故障检测 2 功能打开，输入源为 FLT 引脚
2	**FLT2S**	**故障检测 2 有效电平选择位** 0：故障检测 2 FLT 输入高电平有效 1：故障检测 2 FLT 输入低电平有效
1	**FLT2M**	**检测功能模式选择** 0：锁存模式，当检测到故障输入有效，FLTSTAT 被硬件置 1，PWM 将立即停止输出，当故障输入变为无效时状态保持不变，只有当 FLTSTAT 被软件清零后，PWM 波形才会在 PWM 时基计数器归 0 时刻或与周期寄存器 PWMP 的值匹配时恢复输出。（当故障输入一直有效时 FLTSTAT 无法被软件清除） 1：逐次模式，PWM 输出直接由故障检测输入端来控制，如果故障输入有效，FLTSTAT 被硬件置 1，立即关闭 PWM 输出。如果故障输入变为无效，FLTSTAT 自动清 0，PWM0/1/2 波形将在 PWM 时基计数器归 0 时刻自动恢复输出
0	**FLT2STAT**	**故障检测标志位** 0：PWM0/1/2 模块处于正常输出状态 1：出现过流，PWM0/1/2 模块处于停止输出状态，如果在锁存模式，可软件清 0，恢复 PWM 输出 注：FLTSTAT 与 FLTIF 的区别为逐次模式下，FLTSTAT 会被清零

16.4.19 PWM0/1/2 输出电平保护控制寄存器 (MCM_POSCR)

偏移地址: 0x0064

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								OLS EN	保留	OLS G21	OLS G11	OLS G01	OLS G2	OLS G1	OLS G0
-								*RW*	-	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	**-**
7	**OLSEN**	输出电平保护的使能位 0：禁止 1：使能
6	保留	**-**
5	**OLSG21**	输出短路保护中 **PWM21** 的有效电平设定位 0：Low 电平有效 1：High 电平有效
4	**OLSG11**	输出短路保护中 **PWM11** 的有效电平设定位 0：Low 电平有效 1：High 电平有效
3	**OLSG01**	输出短路保护中 **PWM01** 的有效电平设定位 0：Low 电平有效 1：High 电平有效
2	**OLSG2**	输出短路保护中 **PWM2** 的有效电平设定位 0：Low 电平有效 1：High 电平有效
1	**OLSG1**	输出短路保护中 **PWM1** 的有效电平设定位 0：Low 电平有效 1：High 电平有效
0	**OLSG0**	输出短路保护中 **PWM0** 的有效电平设定位 0：Low 电平有效 1：High 电平有效

16.4.20 PWM0/1/2 振荡器停止检测保护控制位 (MCM_POSTDCR)

偏移地址: 0x0070

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								OSTDEN	保留						
-								*RW*	-						
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7	**OSTDEN**	振荡器停止保护 **PWM** 输出使能位 0：禁止 1：使能
6 ~ 0	保留	-

16.4.21 PWM 中断使能控制寄存器 (MCM_PWMINTEN)

偏移地址: 0x0074

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留			OSTDIE	OIE	FLTCMP1IE	FLTCMP0IE	FLTIE	PWMPIE	PWMZIE	PTDD2IE	PTUD2IE	PTDD1IE	PTUD1IE	PTDD0IE	PTUD0IE
-			*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 13	保留	-
12	**OSTDIE**	振荡器停止保护 **PWM** 输出中断使能位 0：禁止中断请求 1：允许中断请求
11	**OIE**	输出短路的中断允许位 0：禁止中断请求 1：允许中断请求
10	**FLTCMP1IE**	**CMP1** 故障中断允许位 0：禁止故障检测中断 1：允许故障检测中断
9	**FLTCMP0IE**	**CMP0** 故障中断允许位 0：禁止故障检测中断 1：允许故障检测中断

8	**FLTIE**	**FLT 引脚故障中断允许位** 0：禁止故障检测中断 1：允许故障检测中断
7	**PWMPIE**	**PWM 时基周期匹配中断允许位（只在中心对齐模式下有效）** 0：禁止 PWM 时基周期匹配中断 1：允许 PWM 时基周期匹配中断
6	**PWMZIE**	**PWM 时基归零中断允许位** 0：禁止 PWM 时基归零中断 1：允许 PWM 时基归零中断
5	**PTDD2IE**	**PWM 时基计数器减计数时与占空比寄存器 PWM2D 匹配时中断使能位 （只在中心对齐模式下有效）** 0：禁止 PWM 计数器减计数时与 PWM2D 匹配时触发中断 1：允许 PWM 计数器减计数时与 PWM2D 匹配时触发中断
4	**PTUD2IE**	**PWM 时基计数器增计数时与占空比寄存器 PWM2D 匹配时中断使能位** 0：禁止 PWM 计数器增计数时与 PWM2D 匹配时触发中断 1：允许 PWM 计数器增计数时与 PWM2D 匹配时触发中断
3	**PTDD1IE**	**PWM 时基计数器减计数时与占空比寄存器 PWM1D 匹配时中断使能位 （只在中心对齐模式下有效）** 0：禁止 PWM 计数器减计数时与 PWM1D 匹配时触发中断 1：允许 PWM 计数器减计数时与 PWM1D 匹配时触发中断
2	**PTUD1IE**	**PWM 时基计数器增计数时与占空比寄存器 PWM1D 匹配时中断使能位** 0：禁止 PWM 计数器增计数时与 PWM1D 匹配时触发中断 1：允许 PWM 计数器增计数时与 PWM1D 匹配时触发中断
1	**PTDD0IE**	**PWM 时基计数器减计数时与占空比寄存器 PWM0D 匹配时中断使能位 （只在中心对齐模式下有效）** 0：禁止 PWM 计数器减计数时与 PWM0D 匹配时触发中断 1：允许 PWM 计数器减计数时与 PWM0D 匹配时触发中断
0	**PTUD0IE**	**PWM 时基计数器增计数时与占空比寄存器 PWM0D 匹配时中断使能位** 0：禁止 PWM 计数器增计数时与 PWM0D 匹配时触发中断 1：允许 PWM 计数器增计数时与 PWM0D 匹配时触发中断

16.4.22 PWM 中断标志和清除寄存器 (MCM_PWMINTF)

偏移地址: 0x0078

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SC2STATC	SC1STATC	SC0STATC	OSTDFC	OSFC	FLTCMP1IFC	FLTCMP0IFC	FLTIFC	PWMPIFC	PWMZIFC	PTDD2IFC	PTUD2IFC	PTDD1IFC	PTUD1IFC	PTDD0IFC	PTUD0IFC
WO	*WO*	*WO*	*WO*	*WO*	*WO*	*WO*	*WO*	*WO*	*WO*	*WO*	*WO*	*WO*	*WO*	*WO*	*WO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SC2STAT	SC1STAT	SC0STAT	OSTDF	OSF	FLTCMP1IF	FLTCMP0IF	FLTIF	PWMPIF	PWMZIF	PTDD2IF	PTUD2IF	PTDD1IF	PTUD1IF	PTDD0IF	PTUD0IF
RO	*RO*	*RO*	*RO*	*RO*	*RO*	*RO*	*RO*	*RO*	*RO*	*RO*	*RO*	*RO*	*RO*	*RO*	*RO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31	**SC2STATC**	**SC2 故障检测标志位清除位** 0：无效 1：清除
30	**SC1STATC**	**SC1 故障检测标志位清除位** 0：无效 1：清除
29	**SC0STATC**	**SC0 故障检测标志位清除位** 0：无效 1：清除
28	**OSTDFC**	**振荡器停止保护 PWM 输出中断标志位清除位** 0：无效 1：清除
27	**OSFC**	**输出短路标志位清除位** 0：无效 1：清除
26	**FLTCMP1IFC**	**CMP1 故障中断标志位清除位** 0：无效 1：清除
25	**FLTCMP0IFC**	**CMP0 故障中断标志位清除位** 0：无效 1：清除
24	**FLTIFC**	**FLT 引脚故障中断标志位清除位** 0：无效 1：清除
23	**PWMPIFC**	**PWM 时基周期匹配中断标志位清除位** 0：无效 1：清除
22	**PWMZIFC**	**PWM 时基归零中断标志位清除位** 0：无效 1：清除
21	**PTDD2IFC**	**PWM 时基计数器减计数时与占空比寄存器 PWM2D 匹配时中断标志位清除位** 0：无效 1：清除

20	**PTUD2IFC**	**PWM 时基计数器增计数时与占空比寄存器 PWM2D 匹配时中断标志位清除位** 0：无效 1：清除
19	**PTDD1IFC**	**PWM 时基计数器减计数时与占空比寄存器 PWM1D 匹配时中断标志位清除位** 0：无效 1：清除
18	**PTUD1IFC**	**PWM 时基计数器增计数时与占空比寄存器 PWM1D 匹配时中断标志位清除位** 0：无效 1：清除
17	**PTDD0IFC**	**PWM 时基计数器减计数时与占空比寄存器 PWM0D 匹配时中断标志位清除位** 0：无效 1：清除
16	**PTUD0IFC**	**PWM 时基计数器增计数时与占空比寄存器 PWM0D 匹配时中断标志位清除位** 0：无效 1：清除
15	**SC2STAT**	**SC3 故障检测标志位** 0：SC3 选定通道处于正常输出状态 1：出现过流，SC3 选定通道处于输出无效电平状态
14	**SC1STAT**	**SC2 故障检测标志位** 0：SC2 选定通道处于正常输出状态 1：出现过流，SC2 选定通道处于输出无效电平状态
13	**SC0STAT**	**SC1 故障检测标志位** 0：SC1 选定通道处于正常输出状态 1：出现过流，SC1 选定通道处于输出无效电平状态
12	**OSTDF**	**振荡器停止保护 PWM 输出中断标志位** 0：无振荡器停振现象 1：发生了振荡器停振
11	**OSF**	**输出短路标志** 0：不同时变为有效电平 1：同时变为有效电平（MCM 中 3 组 2 相输出中，至少有 1 组同时为有效电平）
10	**FLTCMP1IF**	**CMP1 故障中断标志位** 0：无故障检测中断 1：发生故障检测中断
9	**FLTCMP0IF**	**CMP0 故障中断标志位** 0：无故障检测中断 1：发生故障检测中断
8	**FLTIF**	**FLT 引脚故障中断标志位** 0：无故障检测中断 1：发生故障检测中断
7	**PWMPIF**	**PWM 时基周期匹配中断标志位（只在中心对齐模式下有效）** 0：无 PWM 周期匹配中断 1：发生 PWM 周期匹配中断
6	**PWMZIF**	**PWM 时基归零中断标志位** 0：无 PWM 时基归零中断 1：发生 PWM 时基归零中断
5	**PTDD2IF**	**PWM 时基计数器减计数时与占空比寄存器 PWM2D 匹配时中断标志位 （只在中心对齐模式下有效）** 0：未发生中断或发生过但被 PTUD2IF 信号清 0. 1：发生中断

4	**PTUD2IF**	**PWM 时基计数器增计数时与占空比寄存器 PWM2D 匹配时中断标志位** 0：未发生中断或发生过但被 PTDD2IF 信号清 0. 1：发生中断
3	**PTDD1IF**	**PWM 时基计数器减计数时与占空比寄存器 PWM1D 匹配时中断标志位 （只在中心对齐模式下有效）** 0：未发生中断或发生过但被 PTUD1IF 信号清 0. 1：发生中断
2	**PTUD1IF**	**PWM 时基计数器增计数时与占空比寄存器 PWM1D 匹配时中断标志位** 0：未发生中断或发生过但被 PTDD1IF 信号清 0. 1：发生中断
1	**PTDD0IF**	**PWM 时基计数器减计数时与占空比寄存器 PWM0D 匹配时中断标志位 （只在中心对齐模式下有效）** 0：未发生中断或发生过但被 PTUD0IF 信号清 0. 1：发生中断
0	**PTUD0IF**	**PWM 时基计数器增计数时与占空比寄存器 PWM0D 匹配时中断标志位** 0：未发生中断或发生过但被 PTDD0IF 信号清 0. 1：发生中断 注：PTDDxIF 置 1 时，硬件会自动将 PTUDxIF 清零；同样，PTUDxIF 置 1 时，硬件会自动将 PTDDxIF 清零；因此通过判断这两个标志位可以判断目前 PWM 波形处在有效状态还是无效状态。

16.4.23 寄存器修改和重载控制寄存器 (MCM_PWMRLDEN)

偏移地址: 0x007C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								PWMRLDEN[7:0]							
-								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 0	**PWMRLDEN[7:0]**	**寄存器修改和重载控制寄存器** 0x55：允许软件对模块寄存器的修改 0xAA：允许带缓存的寄存器的重载 注：1. PWM 模块的寄存器，除了中断标志寄存器 PWMINTF、手动控制寄存器 PMANUALCON2、FLTCON、POSCR 和 POSTDCR 外，其它寄存器的修改有限制条件，只有当 PWMRLDEN = 0x55 时才允许软件修改，否则修改无效 2. PWM 模块中有一些寄存器带有缓存寄存器，包括占空比寄存器 PWMxD&PWMx1D、周期寄存器 PWMP 和 PWM 时钟预分频寄存器 PWMPSQ、后分频寄存器 POSTPS[2:0]@ PWMCON1、Duty 选择寄存器 PDCONx@PWMCON1、对称性控制寄存器 PWMSYM@PWMCON1、死区寄存器 PWMDTxx、比较寄存器启动 AD 控制位 CMPx@PWMCON2 和 PWM 事件触发寄存器 PWMCMPx，*这些寄存器在被软件修改后只有当 PWMRLDEN = 0xAA 时才被重载。*

16.4.24 PWM 饱和/相移控制寄存器 (MCM_PSCON)

偏移地址: 0x0080

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留				MINSELECT[1:0]		MAXSELECT[1:0]		保留	SECTOR[2:0]			保留	OVER	STATRUN	SHIFTRUN
-				*RW*		*RW*		-	*RO*			-	*RO*	*RW1s*	*RW1s*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 12	保留	-
11 ~ 10	**MINSELECT[1:0]**	**饱和处理时，与下限比较值比较后 PWM 占空比值选择位** 00：任何时候 PWMx 占空比（PWMxD）的值都不变 01：当 PWMxD 小于等于 PWMDMIN 时，PWMxD 被修改为 PWMDMIN；否则不变 1x：当 PWMxD 小于等于 PWMDMIN 时，PWMxD 被修改为 0；否则不变
9 ~ 8	**MAXSELECT[1:0]**	**饱和处理时，与上限比较值比较后 PWM 占空比值选择位** 00：任何时候 PWMx 占空比（PWMxD）的值都不变 01：当 PWMxD 大于等于 PWMDMAX 时，PWMxD 被修改为 PWMDMAX；否则不变 1x：当 PWMxD 大于等于 PWMDMAX 时，PWMxD 被修改为周期值 PWMP；否则不变
7	保留	-
6 ~ 4	**SECTOR[2:0]**	**SVPWM 象限设定位，通过此位可以确定 PWM0D，PWM1D 和 PWM2D 的大小顺序，尤其是有两个值相等时的大小顺序** MAX MED MIN 000：PWM2D PWM1D PWM0D 001：PWM2D PWM0D PWM1D 010：PWM0D PWM2D PWM1D 011：PWM0D PWM1D PWM2D 100：PWM1D PWM0D PWM2D 101：PWM1D PWM2D PWM0D 注 1：三相全相等时，按照 0>1>2 处理 注 2：三相不全相等但两相相等时，按照 0=2,0>2；2=1,2>1；1=0,1>0。
3	保留	-
2	**OVER**	**相移溢出标志位：** 0：没有溢出 1：溢出 注：每运行一次相移，更新一次结果
1	**STATRUN**	**占空比寄存器饱和功能启动位：** 0：无效或者运算完成 1：启动占空比寄存器饱和功能，PWMxD 注：共 6 个占空比寄存器；软件置 1，运算完成后硬件清零；硬件单个 MCM 时钟周期完成
0	**SHIFTRUN**	**相移模块启动位:** 0：无效或者运算完成 1：启动占空比寄存器相移功能 注：软件置 1，运算完成后硬件清零；硬件单个 MCM 时钟周期完成

16.4.25 PWM 占空比上限比较寄存器 (MCM_PWMDMAX)

偏移地址: 0x0084
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMDMAX[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMDMAX[15:0]**	**PWM 占空比上限比较寄存器**

16.4.26 PWM 占空比下限比较寄存器 (MCM_PWMDMIN)

偏移地址: 0x0088
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMDMIN[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMDMIN[15:0]**	**PWM 占空比下限比较寄存器**

16.4.27 PWM 占空比移相最小采样间隔寄存器 1 (MCM_PWMDCMPn)(n=1..2)

偏移地址: 0x008C
:0x0090
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMDCMPn[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**PWMDCMPn[15:0]**	**PWM 占空比移相最小采样间隔寄存器 n**

16.4.28 逐波限流引脚 0 功能选择寄存器 (MCM_SCnCON)(n=0..1)

偏移地址: 0x0094
:0x0098
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留		SCEN[1:0]		SCS	SCPWM2EN	SCPWM1EN	SCPWM0EN	SCDEB[3:0]				SCTIME[3:0]			
-		RW		RW	RW	RW	RW	RW				RW			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 14	保留	-
13 ~ 12	**SCEN[1:0]**	**逐波限流 n 功能控制位** 00：逐波限流 n 功能禁止 01：SCn 引脚引脚做为逐波限流 n 输入 10：CMP0 输出做为逐波限流 n 输入 11：CMP1 输出做为逐波限流 n 输入
11	**SCS**	**SCn 有效电平选择位** 0：逐波限流 SCn 输入高电平有效 1：逐波限流 SCn 输入低电平有效
10	**SCPWM2EN**	**PWM2 保护使能位：** 0：SCn 不保护 PWM2 1：SCn 保护 PWM2
9	**SCPWM1EN**	**PWM1 保护使能位：** 0：SCn 不保护 PWM1 1：SCn 保护 PWM1
8	**SCPWM0EN**	**PWM0 保护使能位：** 0：SCn 不保护 PWM0 1：SCn 保护 PWM0

7 ~ 4	**SCDEB[3:0]**	**逐波限流 SCn 输入滤波参数选择** 0000：0 0001：40*HCLK 0010：80*HCLK 0011：120*HCLK 0100：160*HCLK 0101：240*HCLK 0110：320*HCLK 0111：480*HCLK 1000：640*HCLK 1001：800*HCLK 1010：960*HCLK 1011：1120*HCLK 1100：1280*HCLK 1101：1600*HCLK 1110：1920*HCLK 1111：2560*HCLK 注 1：上述滤波常数时间不是精确值，仅供参考。 注 2：滤波器算法说明：用 HSI 时钟采样滤波器输入。采用数字积分，如果采样结果为高电平，计数器加 1，计数器结果超出设定的常数，则滤波器输出 1 同时计数器置为滤波常数×2；如果采样结果为低电平，计数器减 1，计数器结果小于滤波器常数,则滤波器输出 0 同时计数器置为 0。
3 ~ 0	**SCTIME[3:0]**	**保护时间选择位：** 0000：下一个归零匹配时恢复 0001：1200*HCLK 0010：1600*HCLK 0011：2000*HCLK 0100：2400*HCLK 0101：2800*HCLK 0110：3200*HCLK 0111：4000*HCLK 1000：4800*HCLK 1001：6400*HCLK 1010：8000*HCLK 1011：16000*HCLK 1100：32000*HCLK 1101：48000*HCLK 1110：64000*HCLK 1111：手动恢复 MCM 输出，通过清除 SCnSTATC@PWMINTF

16.4.29 逐波限流引脚 2 功能选择寄存器 (MCM_SC2CON)

偏移地址: 0x009C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留		SCEN[1:0]		SCS	SCPWM2EN	SCPWM1EN	SCPWM0EN	SCDEB[3:0]				SCTIME[3:0]			
-		RW		RW	RW	RW	RW	RW				RW			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 14	保留	-
13 ~ 12	**SCEN[1:0]**	**逐波限流 2 功能控制位** 00：逐波限流 2 功能禁止 01：SC2 引脚引脚做为逐波限流 2 输入 10：CMP0 输出高做为逐波限流 2 输入 11：CMP1 输出做为逐波限流 2 输入
11	**SCS**	**SC2 有效电平选择位** 0：逐波限流 SC2 输入高电平有效 1：逐波限流 SC2 输入低电平有效
10	**SCPWM2EN**	**PWM2 保护使能位：** 0：SC2 不保护 PWM2 1：SC2 保护 PWM2
9	**SCPWM1EN**	**PWM1 保护使能位：** 0：SC2 不保护 PWM1 1：SC2 保护 PWM1
8	**SCPWM0EN**	**PWM0 保护使能位：** 0：SC2 不保护 PWM0 1：SC2 保护 PWM0

7 ~ 4	**SCDEB[3:0]**	**逐波限流 SC3 输入滤波参数选择** 0000：0 0001：40*HCLK 0010：80*HCLK 0011：120*HCLK 0100：160*HCLK 0101：240*HCLK 0110：320*HCLK 0111：480*HCLK 1000：640*HCLK 1001：800*HCLK 1010：960*HCLK 1011：1120*HCLK 1100：1280*HCLK 1101：1600*HCLK 1110：1920*HCLK 1111：2560*HCLK 注 1：上述滤波常数时间不是精确值，仅供参考。 注 2：滤波器算法说明：用 HSI 时钟采样滤波器输入。采用数字积分，如果采样结果为高电平，计数器加 1，计数器结果超出设定的常数，则滤波器输出 1 同时计数器置为滤波常数×2；如果采样结果为低电平，计数器减 1，计数器结果小于滤波器常数,则滤波器输出 0 同时计数器置为 0。
3 ~ 0	**SCTIME[3:0]**	**保护时间选择位：** 0000：下一个归零匹配时恢复 0001：1200*HCLK 0010：1600*HCLK 0011：2000*HCLK 0100：2400*HCLK 0101：2800*HCLK 0110：3200*HCLK 0111：4000*HCLK 1000：4800*HCLK 1001：6400*HCLK 1010：8000*HCLK 1011：16000*HCLK 1100：32000*HCLK 1101：48000*HCLK 1110：64000*HCLK 1111：手动恢复 MCM 输出，通过清除 SC2STATC@PWMINTF

16.4.30 寄存器修改和重载保护控制寄存器 (MCM_FLTWEN)

偏移地址: 0x00A0

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
FLTWEN[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**FLTWEN[15:0]**	寄存器修改和重载保护控制寄存器 0x33CC：允许软件对保护模块寄存器的修改 注：保护寄存器包括 FLTCON、POSCR 和 POSTDCR，这些寄存器的修改有限制条件，只有当 FLTWEN = 0x33CC 时才允许软件修改，否则修改无效

17. 可编程计数器阵列（PCA，PCA0～PCA1）

17.1 简介

可编程计数器阵列 PCA 提供增强的定时器功能，能够提供输入捕捉、比较匹配(输出)、频率可调方波输出、PWM 调制输出四种增强功能。SH33F2801 内置 2 个 PCA 模块（PCA0/1），每个 PCA 模块均提供 3 路输入输出通道。

17.2 主要特性

- 32/16位定时器
- 3路独立的输入输出通道
- 支持输入捕捉、比较匹配(输出)
- 支持频率可调的方波输出
- 32位PWM，4种输出模式输出

17.3 功能描述

可编程计数器阵列PCA由一个基本计数单元、3个32位捕捉/比较模块及一个故障检测模块组成，每个捕捉/比较模块有独立的引脚（PCAxA/PCAxB/PCAxC），其引脚可以作为捕捉信号输入或波形输出。

注：小下标x为PCA序号，例如PCAx（x=0，1），下文将统一使用PCAx，不再说明x的值。

图17-1 PCA 原理框图

计数单元

PCAx由一个32位的周期寄存器PCAx_PR、一个32位32位的计数寄存器PCAx_CNT和一个16位的预分频寄存器PCAx_PSQ组成基本的计数/定时单元。

32位的计数器有2种计数模式，分别为向上计数和中央对齐计数。当寄存器PCAx_CFGR中的SDEN位为0时，计数器工作在向上计数模式，当SDEN为1时，计数器工作在中央对齐计数模式。

在向上计数模式中，计数器从0向上计数到周期寄存器PCAx_PR的值，然后重新从0开始计数并且产生一个计数器溢出事件，同时状态寄存器PCAx_SR中的CIF位会被硬件置1。如果寄存器PCAx_CFGR中的CIE位被软件设置为1，则会产生一个中断。

在中央对齐计数模式中，计数器从0开始向上计数到周期寄存器PCAx_PR值，然后再向下计数到1并且产生一个计数器溢出事件，同时状态寄存器PCAx_SR中的CIF位会被硬件置1。如果寄存器PCAx_CFGR中的CIE位被软件设置为1，则会产生一个中断。

注：向上计数模式也可称作锯齿波模式，中央对齐计数模式也可称作三角波模式。

计数器时钟源

计数器的时钟源可以通过设置寄存器PCAx_CFGR中的CPS[2:0]来选择，分别为PCLK时钟、PCAx_ECI引脚时钟输入和低频RC 128KHz时钟源，可以通过PCAx_CFGR寄存器中的CPS[2:0]位来选择，如下表所示。

CPS[2]	CPS[1]	CPS[0]	
0	0	0	PCLK时钟
0	0	1	外部引脚输入PCAxECI时钟下降沿
0	1	0	LSI(RC 128KHz)
0	1	1	保留项

17.4 工作模式

PCAx 有3路捕捉/比较模块，每个捕捉/比较模块对应1路通道且都可独立配置。可以通过配置各自模块寄存器PCAx_CCMR中SM[1:0]选择该模块工作在以下4种工作模式之一：输入捕捉、比较匹配输出、频率输出、PWM输出模式。寄存器PCAx_CCMR中的FS[1:0]可以对每种工作模式进行更进一步的配置，只有当SM[1:0]=10时，对FS[1:0]设置无效。

工作方式选择如下表所示：

表17-1 PCAx 模式选择表

模式	SDEN	SM[1]	SM[0]	FS[1]	FS[0]	功能说明
Mode0	0	0	0	0	X	正沿触发捕捉（锯齿波）
				1	0	负沿触发捕捉（锯齿波）
				1	1	任意沿触发捕捉（锯齿波）
Mode1	0	0	1	0	X	连续比较匹配(输出)（锯齿波）
				1	X	单次比较匹配(输出)（锯齿波）
Mode2	0	1	0	X	X	频率输出（锯齿波）
Mode3	0	1	1	0	0	16位PWM1模式（锯齿波）
				0	1	32位PWM2模式（锯齿波）
	1			1	0	32位PWM3模式（三角波）
	1			1	1	32位PWM4模式（三角波）
其他	-					PCAx 计数器正确计数，但比较/捕捉模块不工作。

注：X表示任意值;

17.4.1 输入捕捉模式

当寄存器PCAx_CCMR中的SM[1:0]=00b时，捕捉/比较模块工作在输入捕捉模式下，寄存器PCAx_CCMR中的FS[1:0]用于选择触发捕捉的电平变化类型。当FS[1:0]=0Xb时，上升沿捕捉；当FS[1:0]=10b时，下降沿捕捉；当FS[1:0]=11b时，上升/下降沿都可触发捕捉。捕捉发生时计数器PCAx_CNT的值会被装载到对应模块的捕捉/比较寄存器PCAx_CCR中。当捕捉发生时，PCAx_SR中的捕捉/比较标志CCnIF被置1，如果相应模块的CCIE位为1，会产生一个中断请求。CCnIF标志位不能被硬件自动清除，必须用软件向PCAx_SR寄存器中的相应CCIFC位写1清除。

在输入捕捉模式下，可以通过读取寄存器PCAx_CCMR中的TCP位来判断当前发生的捕捉是上升沿捕捉还是下降沿捕捉。当TCP=0时，表示当前发生的是下降沿捕捉；当TCP=1时，表示当前发生的是上升沿捕捉。

发生捕捉时，可以通过设置寄存器PCAx_CCMR中的CC位选择是否清除计数器的值。如果CC为0，发生捕捉时不清除计数器值，如果CC为1，发生捕捉时硬件把当前计数器值保存到PCAx_CCR寄存器后再将计数器的值清0。在实际捕捉输入信号时，如果CC为1，由硬件清除计数器值，那么保存在CCR寄存器中的值就是输入信号的实际脉宽。

每个模块对应的输入通道都有独立的滤波功能，可以设置PCAx_CCMR中的ICF位段选择滤波时间。当ICF[2:0]=0时，对输入信号没有滤波作用。当ICF[2:0]=1~7时，滤波时间分别为40PCLK，80PCLK，160PCLK，…，2560PCLK。输入信号至少要保持滤波时间长度，才能被内部检测电路认为是有效的电平。

图17-2　　PCA 捕捉方式原理框图

17.4.2　比较匹配输出

当PCAx工作于比较匹配模式下时，设置FS[1:0]=0Xb，可以实现连续匹配，计数器会从0到TOP值重复计数。当发生匹配时，模块输出引脚PCAxA/B/C的电平会发生翻转，同时寄存器PCAx_SR中的捕捉/比较标志CCIF会被置1，如果相应模块的CCIE为1，则会触发中断。当FS[1:0]=1Xb时，此模式为单次匹配，即计数器从0计数到PR值后溢出重新计回0且不断重复计数，但是输出引脚PCAxA/B/C的电平也只在第一次匹配时发生翻转，之后就保持翻转后的电平不再改变。相应模块的标志CCIF会被置1，如果CCIE为1，则会触发一次中断。无论是连续匹配还是单次匹配，都可以通过PCAx_CCMR中的TCP位改变输出波形的初始电平，当TCP为0时，输出波形初始电平为高电平，TCP位为1时，输出波形初始电平为低电平。

图17-3　　比较匹配方式原理框图

另外，可以通过设置PCAx_FORCE寄存器FCOn位强制产生一次比较匹配，但是，这样的匹配并不是真正的匹配，既不会置起比较匹配标志位，也不会影响寄存器数值，只是会在PCAxA/B/C输出引脚产生电平翻转。当产生一次强制匹配生效后，FCO位会被硬件自动清0。

17.4.3 频率输出方式

通过设置寄存器CCRM中的SM[1:0]=10b，可以使得相应模块工作在频率可编程方波输出模式，方波通过引脚PCAxA/B/C输出。在此模式下，周期寄存器PR、计数器CNT及捕捉/输出寄存器CCR都由32位分成两个16位的寄存器使用，用高位寄存器（如PRH）和低位寄存器（PRL）区分。

此模式下，周期寄存器低16位PRL固定为0xFFFF，不可修改，PRH与PRL组成的32位数值为周期值，如果PRH设置为0x053C，则周期为0x053CFFFF，当计数器计数到这个值时会溢出然后重新从0开始计数，同时PCAx的溢出标志CIF被置1，如果此时计数器溢出中断允许位CIE为1，则会触发中断。

当捕捉/比较寄存器的低16位CCRL与计数器的低16位CNTL进行比较时，若两者匹配，则PCAxA/B/C/引脚的电平会发生翻转，同时高16位寄存器CCRH的值会被以偏移值的方式加到CCRL上并且立即生效，当CNTL继续计数到CCRL+CCRH时再次发生匹配，输出引脚的电平会发生翻转。只要发生匹配，CCRL的值就会在原有的数值上再加上CCRH的值并立即得到更新。

PCAx工作在频率输出模式下，只有低16位会发生匹配，因此当CCRL的值不断累加超过0xFFFF后就溢出，溢出后的低16位则继续与计数器CNTL比较。

输出引脚PCAxA/B/C的初始电平可以根据TCP位来选择，当TCP为0时，初始电平为高电平，当TCP为1时，初始电平为低电平。

图17-4　　频率输出方式原理框图

17.4.4 PWM 输出

PCAx的每个模块都可以独立地产生脉宽调制（PWM）输出。配置SM[1:0]=11b可以使得比较/捕捉模块工作在PWM模式。在该模式下，通过配置FS[1:0]位可以使比较/捕捉模块输出以下4种PWM波形。

FS[1]	FS[0]	功能说明
0	0	16位PWM1
0	1	32位PWM2
1	0	32 位PWM3
1	1	32 位PWM4

■ 16位PWM1模式

当比较/捕捉模块工作在16位PWM功能时，计数器PCAx_CNT的低16位PCAx_CNTL从0x0向PCAx_PRL递增计数（计数器为锯齿波模式），当PCAx_CNTL溢出时（从0xFFFF到0x0），保存在PCAx_CCRH中的值被自动装载到PCAx_CCRL中，这个过程不需软件干预。

图17-5　　16 位脉宽调制器（PWM）方式原理框图

模块的捕捉/比较寄存器PCAx_CCRH用来改变PWM 输出信号的占空比。当TCP=0时，PCAx计数器（计数器为锯齿波模式）低字节PCAx_CNTL与PCAx_CCRL中的值相等时，PCAxA/B/C引脚上的输出由高电平变为低电平’；当计数器PCAx_CNTL溢出时，PCAxA/B/C引脚上的输出由低电平变为高电平；当TCP=1时，PCAxA/B/C引脚输出极性相反的波形。

■ 32位PWM2模式

在该方式下当TCP =0时，PCAx计数器（计数器为锯齿波模式）与模块的匹配寄存器CCR值匹配时，PCAxA/B/C引脚上的输出由高电平变为低电平；当计数器溢出时，PCAxA/B/C引脚上的输出由低电平变为高电平，当TCP=1时，PCAxA/B/C引脚输出极性相反的波形。实现原理框图如下图17-11所示。

图17-6　　32 位 PWM 模式框图（计数器为锯齿波模式）

■ 32位PWM3模式

在此模式下，寄存器PCAx_CCMR中的TCP位可以用来选择Duty区的电平。当TCP=0时，当计时器（计数器为三角波模式）从0往PCAx_PR计数时若计数器值与PCAx_CCR匹配，PCAxA/B/C引脚上的输出将由高电平变为低电平；而在计时器从周期值往0x0计数时若计数器值与PCAx_CCR匹配，PCAxA/B/C引脚上的输出将由低电平变为高电平。当TCP=1时，PCAxA/B/C引脚输出极性相反的波形。实现原理框图如下图17-13所示。当匹配发生时，匹配标志位CCIF被置1，若中断允许位CCIE为1，则会产生一次PCAx中断。当PCAx从周期值递减计数回到0x0溢出时，PCAx中断标志CIF也会被置1，若中断允许位CIE为1，则会产生一次PCAx中断。

在此模式下，修改PR和CCR的值，将会在计数器运行到PR值时才能生效。因此，如果在计数器从0往PR计数时，修改了PR值和CCR的值，CCR值在计数器从PR值回0x0的这半个周期即可使用，而周期值要在计数器从PR回到0x0后，在新的一个计数周期里被修改的PR值才被使用。因此无论在任何时刻修改PR值，计数器的值都不会突变。

■ 32位PWM4模式

TCP=0时，当计时器（计数器为三角波模式）往周期值PR计数时若计数器值PCAx_CNT与PCAx_CCR匹配，PCAxA/B/C引脚上的输出由高电平变为低电平；而在计时器往0x0计数时若计数器值PCAx_CNT与PCAx_CCR匹配，PCAxA/B/C引脚上的输出由低电平变为高电平。当TCP=1时，PCAxA/B/C引脚输出极性相反的波形。当匹配发生时，匹配标志位PCAxA/B/C被置1，若中断允许，则会产生一次PCAx中断。当计数器从PR递减计数回到0x0溢出时，PCAx中断标志CIF也会被置1，若中断允许，则会产生一次PCAx中断响应。

PWM4与PWM3模式的区别在于PCAx_PR与PCAx_CCR的更新时间不同，因此PWM3可以输出非对称波形，而PWM4总是输出对称波形。这是由于PWM4是在0x0000点更新CCR和PR寄存器的值，而PWM3是在PR点更新CCR和PR寄存器值。

17.5 中断

PCAx有2个独立的中断向量地址，其中计数器溢出和3路输入捕捉/输出比较共用一个中断向量，输出故障检测模块具有独立的一个中断向量。

计数器和3路输入捕捉/输出比较模块虽然共用一个中断向量，但计数器和每路输入捕捉/输出比较模块都具有独立的中断请求源，当计数器溢出时，溢出标志位置1，只要溢出中断允许位CIE为1，即可触发中断，3路输入捕捉/输出比较模块请求中断的过程与计数器一样，均需要开启各自的中断允许位。

当检测到输出故障时，故障检测标志位FLTIF会置1，此时如果故障检测的中断允许位为1，则会触发故障检测中断。

图17-7 PCAx 中断源

17.6 寄存器立即更新

输出比较寄存器CCR和周期寄存器可以通过设置CFGR中的相应位来选择立即生效还是在下一个周期生效。设置CFGR中的CC0UE为1时，写入CCR0的值会立即生效，如果CC0UE为0，则写入CCR0的值在下一个周期生效。

可以设置的寄存器包括CCR0/CCR1/CCR2/PR，其对应的位分别为CFGR中的CC0UE/CC1UE/CC2UE/PUE。

17.7 寄存器锁定

PCAx有两个锁定寄存器，分别为LCKR和FLTWPR。其中FLTWPR只锁定FLTCR寄存器，只有当FLTPWR=0x33CC时，才能修改FLTCR寄存器的值。

LCKR保护除PCAx_SR外的其他寄存器（FLTCR除外），只有当LCKR=0x33CC时，PCAx的寄存器才允许写入。被保护的寄存器包括CR,CFGR,PR,PSC,CNT,CCMR0,CCMR1,CCMR2,CCR0,CCR1,CCR2,ADTR。

17.8 输出故障检测

PCAx中有一个故障检测模块（Fault Detection logic），当PCA工作在Mode1/2/3这三种模式时可以启用此功能。故障检测主要目的是：当故障发生时（比如过流）可以切断PCAx的输出，进入无效驱动状态（可设为高电平、低电平和高阻态），进而达到保护外部功率器件的目的，由于它是由硬件控制，因此响应速度非常快。

注*1*：故障保护在*PCA*模块时钟关闭时也可以正常工作。

17.8.1 故障信号输入源的选择

可以选择比较器（0/1）的输出或FLT引脚输入作为故障检测输入信号。当故障信号来自比较器0/1/的输出时，高电平为有效故障信号，即比较器输出由低变高时输出都将自动变为指定状态。当故障信号来自FLT引脚时，可以通过设置FLT2S的值选择低电平或高电平为有效故障信号。

若FLT1EN = 1，则比较器0或比较器1的输出会作为故障检测输入信号。一旦比较器的输出变高，并保持滤波时间（下节介绍），则A/B/C三路输出立即进入指定的状态。同理，若FLT2EN = 1，则FLT引脚输入作为故障检测输入信号，一旦FLT引脚的输入信号电平与FLT2S设置的有效故障信号电平相同，并保持滤波时间，则A/B/C三路输出立即进入指定的状态。如果FLT2EN = 0，则禁止FLT引脚的故障检测功能，FLT引脚可以作为普通IO使用。

17.8.2 FLT 引脚上的信号与比较器输出信号的滤波

当FLT2EN = 1时，FLT引脚的输入信号会作为故障检测输入信号使用，此时可以通过设置寄存器PCAx_FLTCR中的FLT2DEB[3:0]位段来调整对此信号的滤波时间。FLT2DEB[3:0] = 0时，无滤波作用。当FLT2S = 0时，故障检测输入信号的电平由低变高时立即触发保护；FLT2DEB[3:0] = 0 - 15时，可以将滤波时间设置为0~32us共16档，故障检测输入信号由低变高时，高电平需要至少保持FLT2DEB[3:0]定义的时间长度，故障检测模块才认为故障检测输入信号的电平变高了，从而触发保护；故障检测输入信号的电平由高变低时，低电平需要至少保持FLT2DEB[3:0]定义的时间长度，故障检测模块才认为FLT引脚上的有效故障电平消失了。当FLT2S = 1时，FLT故障检测信号变为低电平有效，情况与FLT2S = 0时相反。

FLT输入引脚滤波器算法说明：用HSI时钟采样滤波器输入。采用数字积分，如果采样结果为高电平，计数器加1，计数器结果超出设定的常数，则滤波器输出1同时计数器置为滤波常数；如果采样结果为低电平，计数器减1，计数器结果小于滤波器常数,则滤波器输出0同时计数器置为0。

注：*1.*计数器上限设置为滤波常数，下限为0，超出上下限，不执行加减动作

2. 滤波计数器初值设定*:*若*FLT*引脚设为高电平有效，计数器初始值为*0*；若*FLT*引脚设为低电平有效，则滤波计数器初值设为滤波常数。

3. 滤波器输出稳定需要一个建立时间，大致为滤波常数设置的时间，如滤波常数设置为*256*，则在*256*个系统时钟后，滤波器输出稳定， 在这之前，滤波器输出不确定。

17.8.3 故障检测同步控制

当寄存器FLTCON中的FLTSYN位为1时，任意PCA模块的FLT引脚检测到故障信号，所有的PCA模块都会根据模块的当前设置进入指定状态。

注：只需设置任意一个模块的FLTSYN位为1，也可同时设置所有的PCA模块的FLTSYN位为1。

17.9 比较触发 ADC 事件

在Mode1/2/3模式时，设置CFGR寄存器中的ADTSEL[1:0]位段可以选择是否触发ADC转换。

当ADTSEL[1:0]=00时，不触发ADC转换；

当ADTSEL[1:0]=01时，在计数器向上计数时，PCAx_ADTR寄存器的数值与计数器值相相匹配时，触发ADC转换；

当ADTSEL[1:0]=10时，在计数器向下计数时，PCAx_ADTR寄存器的数值与计数器值相相匹配时，触发ADC转换；

当ADTSEL[1:0]=11时，无论计数器向下计数还是向下计数，只要PCAx_ADTR寄存器的数值与计数器值相相匹配时，都会触发ADC转换。由于SDEN=0时，计数器只向上计数，因此SDEN为0时，设置ADTSEL[1:0]为10或11，实际效果相当于不触发ADC转换。

17.10 同步启动

17.10.1 PCA 同步使能

多个PCA模块可以通过设置PCAx_CFGR寄存器中的SYN[1:0]来进行同步启动。当PCA0的SYN[1:0]=01时，表示PCA0的使能位PR由PCA1来控制，当PCA1的PEN位为1时，硬件会同时将PCA0的PEN位设置为1。

17.10.2 MCM 同步使能

PCA模块可以通过设置PCAx_CFG寄存器的SYN[1:0]位来进行同步启动。当PCAx的SYN[1:0]=10时，PCAx的使能位PR由MCM模块的使能位控制，即当MCM的使能位为1时，硬件会同时将PCAx的PR位设置为1。

17.11 注意事项

17.11.1 仿真状态下的输出状态

PCA模块在仿真状态下输出与正常运行没有区别，但是如果在仿真状态下CPU处于停止状态（如设置了断点或单步调试等），则输出会自动切换到高阻态，避免损坏外部功率器件。

17.11.2 STOP 模式下的输出状态

当PCA的PEN为1时，进入STOP模式前由硬件将PEN关闭，同时将输出切换为高阻态。当MCU从STOP模式唤醒后，需要由软件再次对PEN写1才能正常输出波形。

17.12 寄存器

PCA 模块寄存器列表 (基地址：0x4000 0400)

地址	寄存器名	说明
0x4000 0400	CR	PCAx 使能寄存器
0x4000 0404	CFGR	PCAx 方式寄存器
0x4000 0408	SR	PCAx 标志寄存器
0x4000 040C	FORCE	PCAx 强制输出控制寄存器
0x4000 0410	CNT	PCAx计数器寄存器
0x4000 0414	PSC	预分频器
0x4000 0418	PR	PCAx 计数周期值
0x4000 041C	CCMR0	PCAx 捕捉/比较模块0控制寄存器
0x4000 0420	CCMR1	PCAx 捕捉/比较模块1控制寄存器
0x4000 0424	CCMR2	PCAx 捕捉/比较模块2控制寄存器
0x4000 0428	CCR0	PCAx 比较/捕捉模块0捕捉值
0x4000 042C	CCR1	PCAx 比较/捕捉模块1捕捉值
0x4000 0430	CCR2	PCAx 比较/捕捉模块2捕捉值
0x4000 0434	LCKR	
0x4000 0438	ADTR	比较触发寄存器
0x4000 043C	FLTWPR	PCA写保护寄存器
0x4000 0440	FLTCR	PWM0/1/2故障检测保护寄存器

17.12.1 PCAx 使能寄存器 (PCAx_CR)

偏移地址: 0x0000
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留															PEN
-															RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 1	保留	-
0	**PEN**	**PCAx 计数器/定时器运行控制位** 0：禁止PCAx计数器/定时器 1：允许PCAx计数器/定时器

17.12.2 PCAx 方式寄存器 (PCAx_CFGR)

偏移地址: 0x0004
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留												SYN[1:0]		ECF[1:0]	
-												RW		RW	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PUE	CC2UE	CC1UE	CC0UE	保留	ADTSEL[1:0]		PIE	CIE	SDEN	保留			CPS[2:0]		
RW	RW	RW	RW	-	RW		RW	RW	RW	-			RW		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 20	保留	-
19 ~ 18	**SYN[1:0]**	**同步使能位** 00：无同步功能 01：PCA模块的PEN位同步置1或清0 10：PEN位的置1与清0由MCM的使能位同步控制 11：保留位 注：当所有PCA模块的SYN均为01时，使能或清0任一个PCA的PEN位，其他PCA的PEN位都由硬件同步使能或清0
17 ~ 16	**ECF[1:0]**	**PCAx_ECI输入信号滤波时间** 00：无滤波功能 01：滤波时间为8*PCLK 10：滤波时间为16*PCLK 11：滤波时间为32*PCLK
15	**PUE**	**PACx周期值立即更新使能位** 0：写入到PR寄存器的值在下一个周期生效 1：写入到PR寄存器的值立即生效

14	**CC2UE**	**PACx比较模块2立即更新使能位** 0：写入到CCR2的值在下一个周期生效 1：写入到CCR2的值立即生效
13	**CC1UE**	**PACx比较模块1立即更新使能位** 0：写入到CCR1的值在下一个周期生效 1：写入到CCR1的值立即生效
12	**CC0UE**	**PACx比较模块0立即更新使能位** 0：写入到CCR0的值在下一个周期生效 1：写入到CCR0的值立即生效
11	保留	-
10 ~ 9	**ADTSEL[1:0]**	**PCAx触发ADC选择位** 00：不触发ADC 01：计数器向上计数可以触发ADC 10：计数器向下计数触发ADC 11：计数器向上/向下计数均可以触发ADC 注：ADTSEL[1:0]为10和11只有在SDEN=1时才有效，SDEN为0时设置ADTSEL[1:0]为1X相当于00
8	**PIE**	**PCAx计数器/定时器周期中断允许位** 0：禁止计数器周期中断 1：当PIF被置位时，允许 PCAx计数器/定时器周期的中断请求 该位是PCAx计数器/定时器周期（PIF）中断的屏蔽位
7	**CIE**	**PCAx计数器/定时器溢出中断允许位** 0：禁止计数器溢出中断 1：当CIF被置位时，允许 PCAx计数器/定时器溢出的中断请求 该位是PCAx计数器/定时器溢出（CIF）中断的屏蔽位
6	**SDEN**	**PCAx 计数模式选择位** 0：PCAx工作在向上计数（锯齿波）模式, 此时该PCAx其它比较捕捉模块的三角模式均被禁止 1：PCAx工作在中央对齐计数（三角波）模式，此时该PCAx其它比较捕捉模块的锯齿波模式均被禁止 当PCAx设定为两种模式中的一种时，比较捕捉模块的另一种斜坡模式即使被配置也是无效的。
5 ~ 3	保留	-
2 ~ 0	**CPS[2:0]**	**PCAx计数器/定时器时钟选择** 000 ： PCLK时钟 001 ： PCAxECI下降沿 010 ： RC128K 011 ： 保留 100 ： 保留 101 ： 保留 110 ： 保留 111 ： 保留

17.12.3 PCAx 标志寄存器 (PCAx_SR)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留								CIFC	保留		PIFC	FLTIFC	CC2IFC	CC1IFC	CC0IFC
-								WO	-		WO	WO	WO	WO	WO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								CIF	保留		PIF	FLTIF	CC2IF	CC1IF	CC0IF
-								RO	-		RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 24	保留	-
23	**CIFC**	**PCAx 计数器/定时器溢出标志清除位** 0：无效 1：清除
22 ~ 21	保留	-
20	**PIFC**	**PCAx 计数器/定时器周期中断标志清除位** 0：无效 1：清除
19	**FLTIFC**	**故障中断标志位清除位** 0：无效 1：清除
18	**CC2IFC**	**PCAx捕捉/比较模块2标志位清除位** 0：无效 1：清除
17	**CC1IFC**	**PCAx捕捉/比较模块1标志位清除位** 0：无效 1：清除
16	**CC0IFC**	**PCAx捕捉/比较模块0标志位清除位** 0：无效 1：清除
15 ~ 8	保留	-
7	**CIF**	**PCAx 计数器/定时器溢出标志** 0：计数器未溢出 1：计数器溢出 当PCAx 计数器/定时器从周期值到0x0000溢出时由硬件置位。在计数器/定时器溢出（CIF）中断被允许时，该位置‘1’将导致 CPU 转向PCAx中断服务程序。该位不能由硬件自动清0，必须用专门的清除位清零。
6 ~ 5	保留	-
4	**PIF**	**PCAx 计数器/定时器周期中断标志** 0：计数器到达周期值 1：计数器未到达周期值 只在SDEN=1时，在计数器值到达周期值会被硬件置位。当SDEN=0时，此位无效，一直为0。当PIE为1时，该位置‘1’将导致 CPU 转向PCAx中断服务程序。该位不能由硬件自动清0，必须用专门的清除位清零。
3	**FLTIF**	**故障中断标志位** 0：无故障检测中断 1：发生故障检测中断

2	**CC2IF**	**CC2IF：PCAx捕捉/比较模块2标志位** 0：未发生比较匹配或捕捉 1：发生了比较匹配或捕捉 在发生一次匹配或捕捉时该位由硬件置位。当CC2IF中断被允许时，该位置‘1’将导致CPU转向PCAx中断服务程序。该位不能由硬件自动清0，必须用专门的清除位清零。
1	**CC1IF**	**CC1IF：PCAx捕捉/比较模块1标志位** 0：未发生比较匹配或捕捉 1：发生了比较匹配或捕捉 在发生一次匹配或捕捉时该位由硬件置位。当CC1IF中断被允许时，该位置‘1’将导致CPU转向PCAx中断服务程序。该位不能由硬件自动清0，必须用专门的清除位清零。
0	**CC0IF**	**CC0IF：PCAx 捕捉/比较模块0标志位** 0：未发生比较匹配或捕捉 1：发生了比较匹配或捕捉 在发生一次匹配或捕捉时该位由硬件置位。当CC0IF中断被允许时，该位置‘1’将导致CPU转向PCAx中断服务程序。该位不能由硬件自动清0，必须用专门的清除位清零。

17.12.4 PCAx 强制输出控制寄存器 (PCAx_FORCE)

偏移地址: 0x000C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留													FCO2	FCO1	FCO0
-													*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 3	保留	**-**
2	**FCO2**	**模块2强制匹配控制位（该位只有当SM[1:0] =01时有效）** 0：不使能强制匹配 1：使能强制匹配，匹配后硬件自动清零
1	**FCO1**	**模块1强制匹配控制位（该位只有当SM[1:0] =01时有效）** 0：不使能强制匹配 1：使能强制匹配，匹配后硬件自动清零
0	**FCO0**	**模块0强制匹配控制位（该位只有当SM[1:0] =01时有效）** 0：不使能强制匹配 1：使能强制匹配，匹配后硬件自动清零

17.12.5 PCAx 计数器寄存器 (PCAx_CNT)

偏移地址: 0x0010
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
CNTH[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CNTL[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**CNTH[15:0]**	**PCAx计数器值高16位**
15 ~ 0	**CNTL[15:0]**	**PCAx计数器值低16位**

17.12.6 预分频器 (PCAx_PSC)

偏移地址: 0x0014
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PSC[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	**-**
15 ~ 0	**PSC[15:0]**	预分频器

17.12.7 PCAx 计数周期值 (PCAx_PR)

偏移地址: 0x0018
复位值: 0xFFFF FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
PRH[15:0]															
RW															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PRL[15:0]															
RW															
1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

位	符号	说明
31 ~ 16	**PRH[15:0]**	**PCAx 计数最大值高字节（MSB）**
15 ~ 0	**PRL[15:0]**	**PCAx 计数最大值低字节（LSB）**

17.12.8 PCAx 捕捉/比较模块 0 控制寄存器 (PCAx_CCMR0)

偏移地址: 0x001C
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留			CSSEL	ICF[2:0]			CC	SM[1:0]		FS[1:0]		CEN	TCP	保留	CCIE
-			RW	RW			RW	RW		RW		RW	RW	-	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 13	保留	-
12	**CSSEL**	**捕获源选择位** 0：选择PCAxA作为捕获输入源 1：选择比较器0的C0OUT0作为捕获输入源 注：此位只在捕获模式下有效（SM=00），其他模式下不能设置为1
11 ~ 9	**ICF[2:0]**	**PCAxA输入信号滤波时间（仅在Mode0模式下有效）** 000：无滤波功能 001：滤波时间为40*PCLK 010：滤波时间为80*PCLK 011：滤波时间为160*PCLK 100：滤波时间为320*PCLK 101：滤波时间为640*PCLK 110：滤波时间为1280*PCLK 111：滤波时间为2560*PCLK
8	**CC**	**捕获发生时清除计数器使能位** 0：发生捕获事件时不清除计数器值 1：发生捕获事件时清除计数器值
7 ~ 6	**SM[1:0]**	**PCAx模式选择位** 00： 捕捉方式 01：比较匹配 10：频率输出 11：PWM输出
5 ~ 4	**FS[1:0]**	**当SM[1:0] =00时：捕捉方式选择位** 0X:捕捉方式下，在正沿触发 10:捕捉方式下，在负沿触发 11：捕捉方式下，在任意沿触发 当SM[1:0] =01时：软件定时方式选择位 0X:连续匹配 1X:单次匹配 当SM[1:0]=11时：PWM方式选择。 00:选择16位PWM1模式 01:选择32位PWM2模式 10:选择32位PWM3模式 11:选择32位PWM4模式 当SM[1:0]=10（频率输出方式）时此位才无效。

3	**CEN**	**比较/捕捉模块功能使能位** 0：禁止比较/捕捉模块0 1：使能比较/捕捉模块0
2	**TCP**	**当SM[1:0] =00时，该位为捕捉信号边沿指示位（只读）** 0：PCAxA通道为下降沿捕捉 1：PCAxA通道为上升沿捕捉 当SM[1:0] =01或10或11时，该位为波形输出控制位 0：初始电平为高电平 1：初始电平为低电平
1	保留	-
0	**CCIE**	**捕捉/比较标志中断允许位** 0：禁止比较捕捉/输出比较中断 1：允许比较捕捉/输出比较中断，当 CCIF位被置1时，允许捕捉/比较标志的中断请求

17.12.9 PCAx 捕捉/比较模块 1 控制寄存器 (PCAx_CCMR1)

偏移地址: 0x0020
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留			CSSEL	ICF[2:0]			CC	SM[1:0]		FS[1:0]		CEN	TCP	保留	CCIE
-			*RW*	*RW*			*RW*	*RW*		*RW*		*RW*	*RW*	-	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 13	保留	-
12	**CSSEL**	**捕获源选择位** 0：选择PCAxB作为捕获输入源 1：选择比较器0的C0OUT1作为捕获输入源 注：此位只在捕获模式下有效（SM=00），其他模式下不能设置为1
11 ~ 9	**ICF[2:0]**	**PCAxB输入信号滤波时间（仅在Mode0模式下有效）** 000：无滤波功能 001：滤波时间为40*PCLK 010：滤波时间为80*PCLK 011：滤波时间为160*PCLK 100：滤波时间为320*PCLK 101：滤波时间为640*PCLK 110：滤波时间为1280*PCLK 111：滤波时间为2560*PCLK
8	**CC**	**捕获发生时清除计数器使能位** 0：发生捕获事件时不清除计数器值 1：发生捕获事件时清除计数器值
7 ~ 6	**SM[1:0]**	**PCAx模式选择位** 00： 捕捉方式 01：软件定时器 10：频率输出 11：PWM输出

5 ~ 4	**FS[1:0]**	**当SM[1:0] =00时：捕捉方式选择位** 0X:捕捉方式下，在正沿触发 10:捕捉方式下，在负沿触发 11：捕捉方式下，在任意沿触发 当SM[1:0] =01时：软件定时方式选择位 0X:连续软件定时模式 1X:单次软件定时模式 当SM[1:0]=11时：PWM方式选择。 00:选择16位PWM1模式 01:选择32位PWM2模式 10:选择32位PWM3模式 11:选择32位PWM4模式 当SM[1:0]=10（频率输出方式）时此位才无效。
3	**CEN**	**比较/捕捉模块功能使能位** 0：禁止比较/捕捉模块1 1：使能比较/捕捉模块1
2	**TCP**	**当SM[1:0] =00时，该位为捕捉信号边沿指示位** 0：PCAxB通道为下降沿捕捉 1：PCAxB通道为上升沿捕捉 当SM[1:0] =01或10或11时，该位为波形输出控制位 0：初始电平为高电平 1：初始电平为低电平
1	保留	**-**
0	**CCIE**	**捕捉/比较标志中断允许位** 0：禁止比较捕捉/输出比较中断 1：允许比较捕捉/输出比较中断，当 CCIF位被置1时，允许捕捉/比较标志的中断请求

17.12.10 PCAx 捕捉/比较模块 2 控制寄存器 (PCAx_CCMR2)

偏移地址: 0x0024
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留			CSSEL	ICF[2:0]			CC	SM[1:0]		FS[1:0]		CEN	TCP	保留	CCIE
-			*RW*	*RW*			*RW*	*RW*		*RW*		*RW*	*RW*	-	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 13	保留	**-**
12	**CSSEL**	**捕获源选择位** 0：选择PCAxC作为捕获输入源 1：选择比较器0的C0OUT2作为捕获输入源 注：此位只在捕获模式下有效（SM=00），其他模式下不能设置为1

11 ~ 9	**ICF[2:0]**	**PCAxC输入信号滤波时间（仅在Mode0模式下有效）** 000：无滤波功能 001：滤波时间为40*PCLK 010：滤波时间为80*PCLK 011：滤波时间为160*PCLK 100：滤波时间为320*PCLK 101：滤波时间为640*PCLK 110：滤波时间为1280*PCLK 111：滤波时间为2560*PCLK
8	**CC**	**捕获发生时清除计数器使能位** 0：发生捕获事件时不清除计数器值 1：发生捕获事件时清除计数器值
7 ~ 6	**SM[1:0]**	**PCAx模式选择位** 00： 捕捉方式 01：软件定时器 10：频率输出 11：PWM输出
5 ~ 4	**FS[1:0]**	**当SM[1:0] =00时：捕捉方式选择位** 0X:捕捉方式下，在正沿触发 10:捕捉方式下，在负沿触发 11：捕捉方式下，在任意沿触发 当SM[1:0] =01时：软件定时方式选择位 0X:连续匹配 1X:单次匹配 当SM[1:0]=11时：PWM方式选择。 00:选择16位PWM1模式 01:选择32位PWM2模式 10:选择32位PWM3模式 11:选择32位PWM4模式 当SM[1:0]=10（频率输出方式）时此位才无效。
3	**CEN**	**比较/捕捉模块功能使能位** 0：禁止比较/捕捉模块2 1：使能比较/捕捉模块2
2	**TCP**	**当SM[1:0] =00时，该位为捕捉信号边沿指示位** 0：PCAxC通道为下降沿捕捉 1：PCAxC通道为上升沿捕捉 当SM[1:0] =01或10或11时，该位为波形输出控制位 0：初始电平为高电平 1：初始电平为低电平
1	保留	-
0	**CCIE**	**捕捉/比较标志中断允许位** 0：禁止比较捕捉/输出比较中断 1：允许比较捕捉/输出比较中断，当 CCIF位被置1时，允许捕捉/比较标志的中断请求

17.12.11 PCAx 比较/捕捉模块 0 捕捉值 (PCAx_CCR0)

偏移地址: 0x0028
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
CCR0H[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CCR0L[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**CCR0H[15:0]**	**PCAx 比较/捕捉模块0寄存器高字节**
15 ~ 0	**CCR0L[15:0]**	**PCAx 比较/捕捉模块0寄存器低字节**

17.12.12 PCAx 比较/捕捉模块 1 捕捉值 (PCAx_CCRn)(n=1..2)

偏移地址: 0x002C
:0x0030
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
CCRnH[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CCRnL[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**CCRnH[15:0]**	**PCAx 比较/捕捉模块n寄存器高字节（MSB）**
15 ~ 0	**CCRnL[15:0]**	**PCAx 比较/捕捉模块n寄存器低字节（LSB）**

17.12.13 寄存器保护控制寄存器 (PCAx_LCKR)

偏移地址: 0x0034
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
KEY[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**KEY[15:0]**	寄存器修改控制寄存器 0x33CC:允许软件对模块受保护的寄存器进行修改 注：受保护的寄存器为 CR,CFGR,PR,PSC,CNT,CCMR0,CCMR1,CCMR2,CCR0,CCR1,CCR2,ADTR

17.12.14 比较触发寄存器 (PCAx_ADTR)

偏移地址: 0x0038

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
DATA[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DATA[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**DATA[31:0]**	触发ADC比较寄存器 此寄存器的数值与计数器匹配时，会触发ADC转换。

17.12.15 PCA 写保护寄存器 (PCAx_FLTWPR)

偏移地址: 0x003C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
KEY[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**KEY[15:0]**	寄存器修改控制寄存器 0x33CC:允许软件对模块受保护的寄存器进行修改 注：受保护的寄存器为FLTCR

17.12.16 PCA 故障检测保护寄存器 (PCAx_FLTCR)

偏移地址: 0x0040

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留	FLTIE	FOUT[1:0]		FLTSYN	FLT1EN	FLT1SEL[1:0]		FLT2DEB[3:0]				FLT2EN	FLT2S	FLTM	FLTSTAT
-	RW	RW		RW	RW	RW		RW				RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 15	保留	-
14	**FLTIE**	**故障中断允许位** 0：禁止故障检测中断 1：允许故障检测中断
13 ~ 12	**FOUT[1:0]**	**故障发生后，输出状态选择位：** 0x：输出高阻态 10：输出低电平 11：输出高电平
11	**FLTSYN**	**故障检测同步控制位** 0：故障检测同步控制禁止 1：故障检测同步控制使能 注：当PCAx的此位为1时，另一个PCA的故障检测（包括故障检测1和故障检测2）发生会同时将此PCA的输出变为预先设置的状态
10	**FLT1EN**	**故障检测1功能控制位** 0：故障检测1功能禁止 1：故障检测1功能打开，输入源可以选择比较器0或比较器1
9 ~ 8	**FLT1SEL[1:0]**	**故障检测1输入源选择控制位** 00：比较器0滤波后的输出作为PCA故障检测1输入源，高电平有效 01：比较器1滤波后的输出作为PCA故障检测1输入源，高电平有效 1X：保留
7 ~ 4	**FLT2DEB[3:0]**	**故障检测2输入滤波参数选择** 0000：无滤波 0001：0.5us 0010：1us 0011：1.5us 0100：2us 0101：3us 0110：4us 0111：6us 1000：8us 1001：10us 1010：12us 1011：14us 1100：16us 1101：20us 1110：24us 1111：32us 注1：上述滤波常数时间不是精确值，仅供参考。

3	**FLT2EN**	**故障检测2功能控制位** 0：故障检测2功能禁止 1：故障检测2功能打开，输入源为FLT引脚
2	**FLT2S**	**故障检测2有效电平选择位** 0：故障检测2 FLT输入高电平有效 1：故障检测2 FLT输入低电平有效
1	**FLTM**	**检测功能模式选择** 0：锁存模式，当检测到故障输入有效，FLTSTAT 被硬件置1，PCA的波形将立即停止输出，当故障输入变为无效时状态保持不变，只有当FLTSTAT被软件清零后，波形才会在发生溢出事件时恢复输出。（当故障输入一直有效时FLTSTAT无法被软件清除） 1：逐次模式，PCA的波形输出直接由故障检测输入端来控制，如果故障输入有效，FLTSTAT被硬件置1，立即关闭PCA的波形输出。如果故障输入变为无效，FLTSTAT自动清0，PCAxA/B/C波形将在发生溢出事件时自动恢复输出
0	**FLTSTAT**	**故障检测标志位** 0：PCA输出模块处于正常输出状态 1：故障状态，PCA输出模块处于停止输出状态

18. 基本定时器（Basic Timer，TIM7～TIM8）

18.1 简介

SH33F2801内置2个基本的16 位定时器，可级联成1个32位的定时器。每个定时器最大能以APB时钟（PCLK）运行，TIM7和TIM8定时器可以STOP模式下运行。

18.2 主要特性

- 16位×2个定时器
- 各定时器进行递增计数
- 各个定时器可以选择独立的时钟源
- 每个定时器可以选择外部时钟源，可做为计数器
- 能够同步运行
- 能够级联成1个32位定时器

18.3 功能

○：能　　—：不能

PCLK为APB时钟，ICLK为时钟源

项目	TIM7	TIM8
时钟源（ICLK）	(1) PCLK (2) T7 (3) 128kHz RC（LSI）	(1) PCLK (2) T8 (3) 128kHz RC（LSI）
时钟源分频	ICLK/(PSQ[15:0]+1)	ICLK/(PSQ[15:0]+1)
周期设定寄存器	TIM7.TPR	TIM8.TPR
输入引脚	T7	T8
输出引脚	T7	T8
级联运行	○	○
A/D转换开始触发	—	TCNT溢出 （**TIM8.TF**）
中断源	TCNT溢出 （**TIM7.TF**）	TCNT溢出 （**TIM8.TF**）

18.3.1 计数器/定时器运行

SH33F2801有2个基本定时器，每个定时器有一个16位数据寄存器TIMx_TCNT（x=7~8），有一个2bit时钟源选择器CLKS[1:0]@TIMx_CR，有一个时钟源分频器TIMx_PSQ[15:0]（x=7~8）。TIMx_CR寄存器的IE位置1能允许定时器TIMx_TCNT（x=7~8）中断。

当开始计时后，定时器开始递增计数，当达到寄存器TIMx.TPR（x=7~8）时溢出，系统置起定时器溢出标志TFx，TIMx_TCNT（x=7~8）被清除。如果定时器x中断被允许，将会产生一个中断。

CLKS[1:0]@TIMx位选择定时器的时钟源：可以选择PCLK、定时器x输入引脚（Tx）的上升沿、内部LSI 128kHz RC和另外一个TIM的溢出标志位。其中只有TIM7可以选择TIM8的溢出标志位作为时钟源（级联）。

STR@TIMx置1时不复位定时器，这意味着如果STR@TIM_CR位置1，定时器寄存器将从TIMx_TCNT开始计数。所以在允许定时器之前，应该设定定时器寄存器的初始值。

当作为定时器应用时，可配置寄存器TIMx_CR中的TC位使定时器x溢出时Tx脚输出电平自动翻转。如果TC被置1，Tx引脚自动设置为输出。

注1：当写入TIMx_TCNT初值大于TIMx_TPR时，TIMx_TCNT会继续向上计数，直到计到0xFFFF后回到0x0000。

注2：TIMx_TCNT可以在定时器运行时修改，在定时器停止时被硬件清零一次。

注3：定时器溢出指的是TIMx_TCNT递增计数过程中与TIMx_TPR相等时。

注4：预分频寄存器只能在定时器停止的时候能够写。

*注5：Tx输入的最快频率1/2*PCLK，滤波时钟选择PCLK。*

注6：Tx输出电平的初始状态与IO口data寄存器的值保持一致。

下图为定时器框图，TIM7和TIM8均可以唤醒低功耗。

图18-1　　定时器框图

注：定时器7和定时器8可以工作在低功耗模式，并且可以以级联的方式唤醒低功耗。

注2：在Stop模式下，Tx口也可以工作，Tx口的滤波用一个128K做一个滤波。

注3：修改TIMx_TCNT时，请确保STR@TIMx_CR为0。

如果CLKS[1:0]@TIMx_CR为00，定时器x不能工作在低功耗模式下。如果CLKS[1:0]@TIMx_CR为01，Tx端口输入外部时钟，定时器x可以工作在普通模式或低功耗模式。当CLKS[1:0]@TIMx_CR为10时，定时器x选择内部128KRC时钟源。

18.3.2　级联运行

级联运行是将2个通道的16位计数器连接为32位计数器的功能。

通过使能TIM7 CR寄存器中的CASCEN位，可以实现级联运行的功能。

级联的两种组合如下表：

组合	高16位	低16位
定时器7和定时器8	TIM7	TIM8

计数器级联运行的例子：

图18-2　　级联运行示例图

注1：级联之后，TIM7被当成一个32位定时器使用，对寄存器的读写操作也可以直接访问32位。

注2：级联时，开启高位TIM(TIM7)的STR@CR位可以同时开始两个Timer。

18.3.3　外部输入时钟源滤波

当CLKS[1:0]@TIMx_CR设置为01时，计数器的时钟由外部Tx引脚的下降沿提供，设置位段ECF@TIMx_CR可以选择不同的滤波参数，当ECF=0时，无滤波作用，当ECF=1,2,3时，滤波时间分别为8PCLK,16PCLK,32PCLK时钟周期。

滤波算法说明：采用数字积分，如果采样结果为高电平，计数器加1，计数器结果超出设定的常数，则滤波器输出1同时计数器置为滤波常数；如果采样结果为低电平，计数器减1，计数器结果小于滤波器常数,则滤波器输出0同时计数器置为0。

18.4 寄存器

TIM 模块寄存器列表 (基地址：0x4000 0C00)

地址	寄存器名	说明
0x4000 0C00	CR	TIM 控制寄存器
0x4000 0C04	TCNT	TIM 计数寄存器
0x4000 0C08	TPR	TIM 周期寄存器
0x4000 0C0C	PSQ	TIM 预分频寄存器
0x4000 0C10	TIMINTF	TIMx 中断标志和清除寄存器

18.4.1 TIM 控制寄存器 (TIMx_CR)

偏移地址: 0x0000
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CASCEN	保留	ECF[1:0]		保留	TC	ETEN	TRIGEN	IE	保留	CLKS[1:0]		OPM	保留		STR
RW	-	*RW*		-	*RW*	*RW*	*RW*	*RW*	-	*RW*		*RW*	-		*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15	**CASCEN**	**TIMx 级联(cascade)使能位(x=7，只在级联的高位 TIM 有效)** 0：禁止 TIM7/TIM8 级联功能 1：使能 TIM7/TIM8 级联功能 注：只有在级联的两个 TIM 都停止计数时才能修改此位
14	保留	-
13 ~ 12	**ECF[1:0]**	**Tx 作为时钟源输入的滤波时间** 00：无滤波功能 01：滤波时间为 8*PCLK 10：滤波时间为 16*PCLK 10：滤波时间为 32*PCLK
11	保留	-
10	**TC**	**比较输出功能允许位** 0：禁止定时器 x 比较输出功能 1：允许定时器 x 比较输出功能 注：当定时器允许比较输出时(TC=1)，时钟源不能选择 Tx(x=7,8)。
9	**ETEN**	**TIMx 溢出事件使能位** 0：禁止定时器溢出唤醒内核 1：允许定时器溢出唤醒内核
8	**TRIGEN**	**TIMx 溢出触发外部模块使能位** 0：定时器 x 溢出禁止触发 ADC 模块 1：定时器 x 溢出允许触发 ADC 模块 注：仅 TIM8 模块可以触发 ADC。

7	**IE**	**TIMx 溢出中断使能位** 0：禁止定时器 TIMx 溢出中断 1：允许定时器 TIMx 溢出中断
6	保留	-
5 ~ 4	**CLKS[1:0]**	**TIMx 时钟源（ICLK）选择位** 00：内部 APB0 时钟（PCLK0） 01：外部时钟源输（Tx） 10：内部 LSI 128KHz RC 11：保留 注：2.级联时高 16 位 TIM 的此位无效，只读属性，与低位 TIM 的此位一样
3	**OPM**	**单脉冲模式** 0：定时器 x 溢出时，定时器不停止 1：定时器 x 溢出时，定时器停止 注：级联时高 16 位 TIM 的此位无效，只读属性，与低位 TIM 的此位一样
2 ~ 1	保留	-
0	**STR**	**TIMx 开始位(x=7,8)** 注 1：可以软件触发，也可以由外部模块触发 注 2：级联时高 16 位 TIM 的此位无效，只读属性，与低位 TIM 的此位一样

18.4.2 TIM 计数寄存器 (TIMx_TCNT)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
TCNTH[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TCNTL[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**TCNTH[15:0]**	**定时器计数值高 16 位：　只有 TIM7 位有效** 注：只有级联时有效，
15 ~ 0	**TCNTL[15:0]**	**定时器计数值低 16 位：　全部 TIM 有效**

注：此寄存器在计数器停止时，被自动清零，且只清一次

18.4.3 TIM 周期寄存器 (TIMx_TPR)

偏移地址: 0x0008
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
TPRH[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TPRL[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**TPRH[15:0]**	**定时器自动重载高 16 位： 只有 TIM7 位有效** 注：只有级联时有效，
15 ~ 0	**TPRL[15:0]**	**定时器自动重载寄存器** 注：低 16 位，全部 TIM 有效

18.4.4 TIM 预分频寄存器 (TIMx_PSQ)

偏移地址: 0x000C
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
PSQH[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PSQL[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**PSQH[15:0]**	**定时器分频器高 16 位： 只有 TIM7 位有效** 注：只有级联时有效。
15 ~ 0	**PSQL[15:0]**	**定时器分频器低 16 位：全部 TIM 有效** 注：定时器时钟为(ICLK/(PSQ[15:0]+1))

注：当写入 TCNT 值时，PSQ 的内部 CNT 值也要清零。

18.4.5 TIMx 中断标志和清除寄存器 (TIMx_TIMINTF)

偏移地址: 0x0010
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															TFC
-															*WO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留															TF
-															*RO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 17	保留	-
16	**TFC**	**TIMx 溢出标志位清除位** 0：无效 1：清除
15 ~ 1	保留	-
0	**TF**	**TIMx 溢出标志位** 0：定时器 x 无溢出，可由软件清 0 1：定时器 x 溢出，由硬件置 1 注：级联之后，32 位溢出才置起标志位

19. 数学协处理单元（Math Co-Processor，MACP）

19.1 简介

SH33F2801的数学协处理单元包含一个32位CORDIC运算单元（简称为CORDIC）和一个电机专用的SVPWM增强引擎（简称为SVPWM）。

CORDIC单元可实现基于圆函数、线性函数、双曲函数迭代的旋转与向量运算，尤其可快速完成高精度的正余弦运算，求模运算，开方运算等。

电机专用SVPWM增强引擎用硬件方式完成电机矢量控制中常用的七段式SVPWM计算，有效提升计算效率，在高性能电机控制中非常有意义。

由于数学协处理单元由独立的硬件实现，可有效节省程序运行时间，减轻CPU负担，提高关键算法的执行效率。

19.2 寄存器列表

MACP 模块寄存器列表 (基地址：0x4004 1400)

地址	寄存器名	说明
0x4004 1400	CORDCSR0	CORDIC 控制状态寄存器组 0
0x4004 1404	OPRDX0	CORDIC 运算操作数 X 寄存器 0
0x4004 1408	OPRDY0	CORDIC 运算操作数 Y 寄存器 0
0x4004 140C	OPRDZ0	CORDIC 运算操作数 Z 寄存器 0
0x4004 1410	CORDCSR1	CORDIC 控制状态寄存器组 1
0x4004 1414	OPRDX1	CORDIC 运算操作数 X 寄存器 1
0x4004 1418	OPRDY1	CORDIC 运算操作数 Y 寄存器 1
0x4004 141C	OPRDZ1	CORDIC 运算操作数 Z 寄存器 1
0x4004 1600	SVCON	SVPWM 运算启动标志寄存器
0x4004 1604	SVUALPHA	SVPWM 输入 Ualpha 值寄存器
0x4004 1608	SVUBETA	SVPWM 输入 Ubeta 值寄存器
0x4004 160C	SVIQN	SVPWM 数据格式配置寄存器
0x4004 1610	SVTA	SVPWM 输出 Ta 值寄存器
0x4004 1614	SVTB	SVPWM 输出 Tb 值寄存器
0x4004 1618	SVTC	SVPWM 输出 Tc 值寄存器
0x4004 161C	SVSECTOR	SVPWM 象限寄存器

19.3 CORDIC 运算单元

19.3.1 主要特性

- 基于CORDIC算法（圆函数迭代、线性函数迭代、双曲函数迭代）实现旋转模式和向量模式
- CORDIC计算采用预处理及后处理逻辑，可支持全范围取值
- CORDIC计算支持IQ26/IQ24/IQ15三种预置数据格式
- CORDIC支持运算结果的K因子自动消除
- 一个计算引擎，两套控制和操作数寄存器

19.3.2 功能描述

19.3.2.1 CORDIC 计算单元

SH33F2801内建一个硬件的CORDIC计算单元，可通过其实现基本CORDIC算法。CORDIC算法是一个迭代过程，每次计算进行**N次迭代，最多N+2个周期内完成（未开启K因子后处理时）**，可用来解三角函数等。

一般CORDIC算法对应如下CORDIC方程：

$$x_{i+1} = x_i - m \cdot d_i \cdot y_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i}$$

$$z_{i+1} = z_i - d_i \cdot e_i$$

其中m=1为圆函数迭代，m=0为线性函数迭代，m=-1为双曲函数迭代。

旋转模式下 $d_i = sign\ (z_i), z_i \to 0$

向量模式下 $d_i = sign\ (y_i), y_i \to 0$

$$e_i = \arctan(2^{-i})$$

19.3.2.2 CORDIC 协处理器的操作

CORDIC协处理器可工作在旋转模式（Rotate）或向量模式（Vector）下，用来计算三角函数，可通过MODE@MACP_CORDCSRx控制位设置旋转或向量模式。

通过ARITH[1:0]@ MACP_CORDCSRx控制位，可选择圆函数迭代、线性函数迭代、双曲函数迭代，分别实现不同的功能，常用的为圆函数迭代，可实现正弦函数、余弦函数、求相角、求模值等常规操作。线性迭代可实现线性函数。双曲迭代可实现双曲正弦、双曲余弦等操作，并可扩展实现开方操作等。

根据需要执行的函数将相应初始值赋值到X，Y，Z数据寄存器中后，启动CORDIC计算（RUN = 1），CORDIC将数据寄存器内容加载到内核计算寄存器中，通过置位RUN启动运算，在计算过程中RUN位一直为1，计算结束后由硬件清零，并自动把计算结果输出到结果寄存器。在运算过程中如果强制清除RUN位，则当前计算立刻停止，计算结果也不输出到结果寄存器（保持前值）。

计算结束后，若有数据溢出，则OVF标志位被置位。

由于CORDIC计算结果带有算法固有的K因子，如希望消除该因子，可开启K因子后处理，也可由用户软件自行计算处理。

CORDIC计算引擎只有一套[注1]，但有两套完全独立的控制和操作数寄存器（组0和组1），两套寄存器可以独立配置工作模式。置位两组寄存器的RUN控制位（RUN0和RUN1在两个寄存器地址，无法同时置位，此处假设RUN0先置位），则组0先运行，组1处于等待状态，在组0运行结束后立刻切换到组1运行。

注1：旋转模式和向量模式只能先后运行。

输入数据寄存器在运行期间无法修改，保证运算数据不会被篡改，但MODE@MACP_CORDCSRx控制位没有做屏蔽，运行期间如果更改MODE位会得到错误的结果，需用户软件保证。

一种典型应用是在主流程中使用寄存器组0进行运算一项运算（旋转或向量），在中断中使用寄存器组1进行运算另一项运算（旋转或向量），由于硬件对两次运算有自动衔接处理，可以减少状态查询和数据输入输出的开销。

19.3.2.3 数据精度和迭代次数

CORDIC允许最高27位数据精度，对应27次迭代，默认数据格式IQ26，输入数据要求在IQ26(-1)～IQ26(+1)之间。

CORDIC还支持16位数据精度，对应16次迭代，默认数据格式IQ15，输入数据要求在IQ15(-1)～IQ15(+1)之间。

对于32位系统，CORDIC操作数寄存器是32位宽的，使用27位CORDIC计算时，实际要求限制输入数据范围为IQ26(-1)～IQ26(+1)之间，CORDIC会直接截取符号位和有效数据位进行运算。

32位输入数据在27位CORDIC计算时的有效数据位

注：如果输入数据是以2的补码格式存储，此处无效数据位就是符号位扩展。

对于16位系统，CORDIC操作数寄存器仍然为32位宽，此时高16位数据无效。

32位输入数据在16位CORDIC计算时的有效数据位

19.3.2.4 数据格式

CORDIC协处理器的初始数据X，Y和Z输入都是有符号数2补码格式。结果数据也是有符号数2补码格式，有一种例外是在向量模式中的X结果数据将会按无符号数存储，这样可防止潜在的X数据结果溢出，这种情况下MSB（最高位）也是数据位。只有工作在向量模式时，X操作结果始终为正。

X和Y输入数据格式可以是整数或定点数，但在任何计算中X和Y数据必须保持一致，如果是定点小数，则必须拥有相同个数的小数位数；而对于Z数据始终是归一化整数值，在旋转模式下**Z作为输入数据时代表角度**，用$[-2^N,(2^N-1)]$代表$[-\pi,\frac{(2^N-1)}{2^N}\pi]$，而在向量模式下**Z作为输出数据时也是代表角度，**角度的分辨率为$\frac{\pi}{2^N}$

输入Z的初始值 = 实数Z初始值（弧度）× $\frac{2^N}{\pi}$

实数Z的结果数据（弧度） = Z 结果数据 × $\frac{\pi}{2^N}$

注：N可取15和26，分别对应16位和27位CORDIC。

需要注意的是，CORDIC的计算结果包含一个固有的，有旋转模式或向量模式引起的增益因子K，在圆函数中K≈1.6467602579。

19.3.2.5 溢出标志及幅度调节

若在CORDIC计算过程中发生了溢出，则会产生OVF标志，有一种例外是在圆函数向量模式中的X结果数据将固定会按无符号数存储，如计算结果在**[0，2^（N+1）]**范围内则不会产生溢出标志。另外CORDCSR的XYMRS位可用来设置CORDIC结果数据的幅度调节，若XYMRS置1则CORDIC迭代结束后，将X和Y的计算结果除以2以后存入X和Y寄存器中，这样可有效防止结果数据溢出。

19.3.2.6 操作模式和对应结果数据

CORDIC方程（以圆函数迭代为例）如下，协处理器操作模式和对应的结果数据见下表，X_{result}，Y_{result}，Z_{result}为最终数据，X, Y, Z表示初始值。

$$x_{i+1} = x_i - d_i \cdot y_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i}$$

$$z_{i+1} = z_i - d_i \cdot e_i$$

CORDIC操作模式和对应结果数据（以圆函数迭代为例）

	旋转模式	向量模式
圆函数 $e_i = \arctan(2^{-i})$	$d_i = sign(z_i), z_i \to 0$	$d_i = sign(y_i), y_i \to 0$
	$X_{result} = K[X\cos(Z) - Y\sin(Z)]$ $Y_{result} = K[X\sin(Z) + Y\cos(Z)]$ $Z_{result} = 0$ 其中K≈1.6467602579	$X_{result} = K\sqrt{X^2 + Y^2}$ $Y_{result} = 0$ $Z_{result} = Z + \arctan(Y/X)$ 其中K≈1.6467602579
	设置 $X = 1/K, Y = 0$ 则 $X_{result} = \cos(Z), Y_{result} = \sin(Z)$ 由于CORDIC内部采用了预处理逻辑，所以X,Y和Z可全范围（$[-2^N, (2^N - 1)]$）取值（N可取15和26）。	设置 $X = X/K, Y = Y/K$ 则向量幅值 $X_{result} = \sqrt{X^2 + Y^2}$ 由于CORDIC内部采用了预处理和后处理逻辑，所以X,Y可全范围（$[-2^N, (2^N - 1)]$）取值（N可取15和26） 设置 $Z = 0$ 则 $Z_{result} = \arctan(Y/X)$ 由于CORDIC内部采用了预处理和后处理逻辑，所以X,Y可全范围（$[-2^N, (2^N - 1)]$）取值（N可取15和26），X=0除外

19.3.2.7 K 因子后处理

如上表所示，CORDIC计算结果中存在K因子，有些应用中该因子需要被消除，可开启K因子后处理功能（KADJ@MACP.CORDCSR）。当设置KADJ=1时，X、Y输出结果乘以1/K（Z输出不受影响）。以圆函数迭代为例：

- 对27位CORDIC，Q26(1/K) = Q26 (0.607252935) = 0x26DD3B6
- 对16位CORDIC，Q15(1/K)= Q15 (0.607252935) = 0x4DBA

K因子后处理在CORDIC运算的输出阶段，在溢出处理之后，因此对前端溢出的数据再进行K因子处理是无意义的。

K因子后处理会额外增加3个周期的运算时间，总运算时间会增加到N+5个。

使用CORDIC圆函数迭代可实现电机矢量控制中常用的Park和iPark算法，需要开启K因子后处理。

PARK公式	IPARK公式
$I_q = \mathrm{I}_\beta \cos(\theta) - \mathrm{I}_\alpha \sin(\theta)$ $I_d = \mathrm{I}_\alpha \cos(\theta) + \mathrm{I}_\beta \sin(\theta)$	$U_\alpha = U_\mathrm{d} \cos(\theta) - U_q \sin(\theta)$ $U_\beta = U_q \cos(\theta) + U_d \sin(\theta)$
$input:$ $X = \mathrm{I}_\beta, Y = \mathrm{I}_\alpha$ $Z = \theta$ $output\ (KADJ = 0):$ $X_{result} = KI_q, Y_{result} = KI_d$ $output\ (KADJ = 1):$ $X_{result} = I_q, Y_{result} = I_d$	$input:$ $X = U_d, Y = U_q$ $Z = \theta$ $output\ (KADJ = 0):$ $X_{result} = KU_\alpha, Y_{result} = KU_\beta$ $output\ (KADJ = 1):$ $X_{result} = U_\alpha, Y_{result} = U_\beta$

19.3.2.8 CORDIC 计算精度

尽管在圆函数旋转模式下，X和Y可以是任意定点格式（Qx或IQx）。但考虑到结果误差及提高分辨率，在进行计算时应尽可能将数值扩大。极端情况下，如输入的X、Y绝对值过小（定点化后），将会造成较大的计算误差。

因此，如果可能，建议用户按最大的数值进行定点化，如对32位系统用IQ26，对16位系统用IQ15。

19.3.2.9 IQ24 数据格式

IQ24 定点格式

CORDIC模块使用27位CORDIC运算内核来支持IQ24数据格式，增加了必要的预处理和后处理。用户只要选择FORMAT=01b，即可由硬件完成这些处理，无需软件处理。

CORDIC不同模式不同数据格式输入输出数值范围（以圆函数迭代为例）

CORDIC 模式	预设数据格式		
	IQ26	**IQ24**	**IQ15**
rotate	input： X：IQ26[-1]～IQ26(+1) Y：IQ26[-1]～IQ26(+1) Z：IQ26[-1]～IQ26(+1)	input： X：IQ24[-4]～IQ24(+4) Y：IQ24[-4]～IQ24(+4) Z：IQ24[-1]～IQ24(+1) cordic 内部自动会对 Z 输入“×4”处理	input： X：IQ15[-1]～IQ15(+1) Y：IQ15[-1]～IQ15(+1) Z：IQ15[-1]～IQ15(+1)
	output：（bypass KADJ） X：IQ26[-1]～IQ26(+1) Y：IQ26[-1]～IQ26(+1) Z：-->0	output：（bypass KADJ） X：IQ24[-4]～IQ24(+4) Y：IQ24[-4]～IQ24(+4) Z：-->0	output：（bypass KADJ） X：IQ15[-1]～IQ15(+1) Y：IQ15[-1]～IQ15(+1) Z：-->0
	output：(enable KADJ) X：IQ26[-1/K]～IQ26(+1/K) Y：IQ26[-1/K]～IQ26(+1/K) Z：-->0	output：(enable KADJ) X：IQ24[-4/K]～IQ24(+4/K) Y：IQ24[-4/K]～IQ24(+4/K) Z：-->0 cordic 内部自动会对 Z 输出“/4”处理（因为 Z 为 0，实际无意义） 注：对 X、Y 内部不会有自动移位处理，移位只针对 Z。	output：(enable KADJ) X：IQ15[-1/K]～IQ15(+1/K) Y：IQ15[-1/K]～IQ15(+1/K) Z：-->0
vector	input： X：IQ26[-1]～IQ26(+1) Y：IQ26[-1]～IQ26(+1) Z：IQ26[-1]～IQ26(+1)	input： X：IQ24[-4]～IQ24(+4) Y：IQ24[-4]～IQ24(+4) Z：IQ24[-1]～IQ24(+1) cordic 内部自动会对 Z 输入“×4”处理	input： X：IQ15[-1]～IQ15(+1) Y：IQ15[-1]～IQ15(+1) Z：IQ15[-1]～IQ15(+1)
	output：（bypass KADJ） X：IQ26[0]～IQ26(+2) Y：-->0 Z：IQ26[-1]～IQ26(+1)	output：（bypass KADJ） X：IQ24[0]～IQ24(+8) Y：-->0 Z：IQ24[-1]～IQ24(+1) cordic 内部自动会对 Z 输出“/4”处理	output：（bypass KADJ） X：IQ15[0]～IQ15(+2) Y：-->0 Z：IQ15[-1]～IQ15(+1)
	output：(enable KADJ) X：IQ26[0]～IQ26(+2/K) Y：-->0 Z：IQ26[-1]～IQ26(+1)	output：(enable KADJ) X：IQ24[0]～IQ24(+8/K) Y：-->0 Z：IQ24[-1]～IQ24(+1) cordic 内部自动会对 Z 输出“/4”处理	output：(enable KADJ) X：IQ15[0]～IQ15(+2/K) Y：-->0 Z：IQ15[-1]～IQ15(+1)

19.3.3 CORDIC 寄存器

19.3.3.1 CORDIC 控制状态寄存器组 0 (MACP_CORDCSRn)(n=0..1)

偏移地址: 0x0000
:0x0010
复位值: 0x0000 1F00

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留	ARITH[1:0]		ITERA[4:0]					MODE	FORMAT[1:0]		保留	XYMRS	KADJ	OVF	RUN
-	RW		RW					RW	RW		-	RW	RW	RO	RW
0	0	0	1	1	1	1	1	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 15	保留	-
14 ~ 13	**ARITH[1:0]**	**CORDIC迭代算法选择** 00：圆函数迭代（默认） 01：线性函数迭代 1x： 双曲函数迭代
12 ~ 8	**ITERA[4:0]**	**迭代次数选择，最大为31，默认按照FORMAT设置** IQ24/IQ26格式下，默认27次迭代 IQ15格式下，建议设置16次，需要用户手动修改 一般迭代次数要和数据格式匹配，但用户也可人为调整迭代次数在精度与速度之间进行权衡，详细的精度情况要参考理论分析，由用户自行确认，此处只给出运算结果不做判断。
7	**MODE**	**CORDIC模式选择位** 0：旋转模式 1：向量模式
6 ~ 5	**FORMAT[1:0]**	**CORDIC运算数据格式** 00：IQ26格式（默认） 01：IQ24格式 1x： IQ15格式 注：IQ26/IQ24都使用27位CORDIC进行运算，IQ15使用16位CORDIC进行运算。 运算数据格式并不局限于上述三种，用户可以使用其他格式输入，但需要对输入和输出进行调整。
4	保留	-
3	**XYMRS**	**CORDIC运算XY幅度调节选择位** 0：最后一次迭代后，X和Y不变作为最终结果 1：最后一次迭代后，X和Y除以2作为最终结果 注：正确设置XYMRS位，可避免结果数据溢出。
2	**KADJ**	**CORDIC运算K因子调节选择位** 0：运算结果不进行1/K调整 1：运算结果乘以1/K，消除K因子 注：K=1.64676
1	**OVF**	**CORDIC运算溢出标志位 (由硬件置1与清0)** 0：无溢出的情况发生 1：有溢出的情况发生 注：运算完成后，该Flag保持，直到模块复位或下次运算启动才清除。

0	**RUN**	**CORDIC运算启动标志位，只能软件置1，硬件或软件清0** 0：运算已结束或未启动 1：启动Cordic运算，运算过程中标志为1,运算完成后标志由硬件清0。 注：允许运算过程中清零RUN以强行结束运算。 RUN启动时会把CORDIC内部flag清零。

19.3.3.2 CORDIC 运算操作数 X 寄存器 0 (MACP_OPRDXn)(n=0..1)

偏移地址: 0x0004
:0x0014
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
OPRDXn[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
OPRDXn[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**OPRDXn[31:0]**	**运算前：** 存放Cordic运算X操作数 运算完成后： 存放Cordic运算X结果

19.3.3.3 CORDIC 运算操作数 Y 寄存器 0 (MACP_OPRDYn)(n=0..1)

偏移地址: 0x0008
:0x0018
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
OPRDYn[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
OPRDYn[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**OPRDYn[31:0]**	**运算前：** 存放Cordic运算Y操作数 运算完成后： 存放Cordic运算Y结果

19.3.3.4 CORDIC 运算操作数 Z 寄存器 0 (MACP_OPRDZn)(n=0..1)

偏移地址: 0x000C
:0x001C
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
OPRDZn[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
OPRDZn[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**OPRDZn[31:0]**	**运算前：** 存放Cordic运算Z操作数 运算完成后： 存放Cordic运算Z结果

19.4 电机专用 SVPWM 增强引擎

19.4.1 SVPWM 功能

SH33F2801包含三相电机的空间矢量脉宽调制（Space Vector Pulse Width Modulation，SVPWM）算法，对应于交流感应电机或永磁同步电机中的三相电压源逆变器的功率器件的一种特殊的开关触发顺序和脉宽大小的组合。这种开关触发顺序和组合将在定子线圈中产生三相互差120°电角度的正弦波电流。

19.4.1.1 SVPWM 算法原理

通过SVPWM的算法产生3相电机电压信号的脉宽调制信号，每相脉宽的产生过程都可简化为几个一次方程。三相逆变器的每相输出都可为两种状态之一，即逆变器输出可连接到正极性母线端或负极性母线端，这使得三相逆变器输出共存在8种可能的状态。其中三相输出全部连接到正极性母线端或负极性母线端的两种状态被视为无效状态，因为此时任意两相之间都不存在线电压。这两种状态在星型图中被绘制为原点。其余六种状态表示为每两个相邻状态间旋转间隔为60度的矢量，如下图所示。

图19-1　　空间矢量调制

空间矢量调制的过程允许通过两个相邻矢量各分量的和来表示任何空间电压矢量。在下图中，UOUT是期望的空间电压矢量。该矢量位于U60和U0之间的区间内。如果在给定PWM周期T期间，U0的输出时间为T1/T而U60的输出时间为T2/T，则整个周期的平均电压值为U_{OUT}。

T0 = 无效矢量

T = T1 + T2 + T0 = PWM 周期

UOUT = (T1/T * U0) + (T2/T * U60)

图19-2　　平均空间矢量调制

T0表示绕组上无有效电压的时间；即施加了无效矢量。在六个区间的每个区间中，一个轴与该区间正好相反，而其它两个轴对称形成该区间的边界。沿着这两个边界轴的矢量分量分别为T1和T2。

19.4.1.2　SVPWM 算法实现过程

首先将静止2轴α-β坐标系变换到定义的静止3轴参考坐标系，需要两个静止坐标系的Uα和Uβ输入。如下所示：

$$\begin{cases} V_{r1} = U_\beta \\ V_{r2} = \left(-U_\beta + \sqrt{3} * U_\alpha\right)/2 \\ V_{r3} = \left(-U_\beta - \sqrt{3} * U_\alpha\right)/2 \end{cases}$$

同时定义三个变量a,b,c，如果Vr1>0，则a=1，否则a=0；如果Vr2>0，则b=1，否则b=0；如果Vr3>0，则c=1，否则c=0。再定义一个变量sector，sector = a*1+b*2+c*4，根据sector不同，把空间分为6个扇区。

另外定义：

$$\begin{cases} X = U_\beta \\ Y = \left(U_\beta + \sqrt{3} * U_\alpha\right)/2 \\ Z = \left(U_\beta - \sqrt{3} * U_\alpha\right)/2 \end{cases}$$

对于不同扇区T1,T2值，有如下对应关系：

表19-1　　扇区与时间对应关系表

Sector	1	2	3	4	5	6
T1	Z	Y	-Z	-X	X	-Y
T2	Y	-X	X	Z	-Y	-Z

在PWM周期T中，矢量T1的输出时间为T1/T，而矢量T2的输出时间为T2/T，在剩余时间内输出无效矢量。无效矢量可以由全部为零组成，为五段式SVPWM波形。也可以由中间为全1两边为全零组成，为七段式SVPWM波形。

19.4.1.3 五段式 SVPWM 算法

下图为五段式SVPWM波形， PWM信号配置为输出中心对齐，这种配置方法在每个周期内可产生1个线间脉冲。

图19-3 周期 T 内的 PWM

能够实现从U_{alpha}，U_{beta}直接计算出Ta，Tb，Tc三相占空比，输入输出的数据格式可以设定，算法流程图如下：

图19-4　　五段式 SVPWM 算法流程图

注1：U_{alpha}、U_{beta}、T1、T2、Ta、Tb和Tc为有符号32位数，IQN和Sector为无符号8位数。

注2：乘法为IQN格式乘法。

注3：SvpwmSector为一个数组，内容为SvpwmSector[]={0,1,5,0,3,2,4,0}

19.4.1.4 七段式 SVPWM 算法

下图为七段式SVPWM波形， PWM信号配置为输出中心对齐，这种配置方法在每个周期内可产生两个线间脉冲，有效开关频率加倍，纹波电流减小，同时并未增加功率器件的开关损耗。

图19-5　　周期 T 内的 PWM

能够实现从Ualpha，Ubeta直接计算出Ta，Tb，Tc三相占空比，输入输出的数据格式可以设定，算法流程图如下：

图19-6　　七段式 SVPWM 算法流程图

注1：U_{alpha}、U_{beta}、T1、T2、Ta、Tb和Tc为有符号32位数，IQN和Sector为无符号8位数。

注2：乘法为IQN格式乘法。

注3：SvpwmSector为一个数组，内容为SvpwmSector[]={0,1,5,0,3,2,4,0}

19.4.1.5 sign 算法

其中sign为符号算法，详细的算法如下：

例如a = sign(x)：

```
if (x > 0)
  a = 1;
else
  a = 0;
```

19.4.1.6 饱和算法

不管五段式还是七段式，都可能存在计算溢出情况，当Ta/Tb/Tc任何一个计算结果超出范围，则都会置起溢出标志位OVER@SVIQN。为了防止计算结果溢出，增加了对Ta,Tb和Tc的饱和处理，通过寄存器SATEN@SVIQN来选择是否使能饱和，详细算法如下：

```
if(Tx < 0)
{
    OVER@SVIQN = 1;
    if (1 == SATEN@SVIQN)
    {
        Tx = 0;
    }
}
else if(Tx > IQN(1))
{
    OVER@SVIQN = 1;
    if(1 == SATEN@SVIQN)
    {
        Tx = IQN(1);
    }
}
```

注1：x=a,b,c

注2：整个SVPWM硬件计算时间为3个AHB CLK。

注3：溢出标志位的值在每次计算时都更新。

19.4.2 SVPWM 寄存器

19.4.2.1 SVPWM 运算启动标志寄存器 (MACP_SVCON)

偏移地址: 0x0200

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留															RUN
-															*RW1s*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 1	保留	-
0	**RUN**	**SVPWM运算启动标志位，只能硬件清0，软件置1** 0：运算已结束或未启动 1：启动SVPWM运算，运算过程中标志为1,运算完成后标志由硬件清0

19.4.2.2 SVPWM 输入 Ualpha 值寄存器 (MACP_SVUALPHA)

偏移地址: 0x0204

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SVUALPHA[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SVUALPHA[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**SVUALPHA[31:0]**	**SVPWM输入Ualpha值**

19.4.2.3 SVPWM 输入 Ubeta 值寄存器 (MACP_SVUBETA)

偏移地址: 0x0208

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SVUBETA[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SVUBETA[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**SVUBETA[31:0]**	**SVPWM输入Ubeta值**

19.4.2.4 SVPWM 数据格式配置寄存器 (MACP_SVIQN)

偏移地址: 0x020C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
OVERCH	OVERBH	OVERAH	OVERCL	OVERBL	OVERAL	SATEN	SVTYP	SVIQN[7:0]							
RO	*RO*	*RO*	*RO*	*RO*	*RO*	*RW*	*RW*	*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15	**OVERCH**	**C相占空比大于IQN溢出标志位：** 0:没有溢出 1:溢出 注：每计算一次更新一次
14	**OVERBH**	**B相占空比大于IQN溢出标志位：** 0:没有溢出 1:溢出 注：每计算一次更新一次
13	**OVERAH**	**A相占空比大于IQN溢出标志位：** 0:没有溢出 1:溢出 注：每计算一次更新一次
12	**OVERCL**	**C相占空比小于0溢出标志位：** 0:没有溢出 1:溢出 注：每计算一次更新一次
11	**OVERBL**	**B相占空比小于0溢出标志位：** 0:没有溢出 1:溢出 注：每计算一次更新一次
10	**OVERAL**	**A相占空比小于0溢出标志位：** 0:没有溢出 1:溢出 注：每计算一次更新一次
9	**SATEN**	**输出饱和使能位：** 0:禁止 1:使能
8	**SVTYP**	**五段式七段式选择位** 0:七段式 1:五段式
7 ~ 0	**SVIQN[7:0]**	**SVPWM数据格式**

19.4.2.5 SVPWM 输出 Ta 值寄存器 (MACP_SVTA)

偏移地址: 0x0210

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SVTA[31:0]															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SVTA[31:0]															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**SVTA[31:0]**	**SVPWM输出Ta值**

19.4.2.6 SVPWM 输出 Tb 值寄存器 (MACP_SVTB)

偏移地址: 0x0214

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SVTB[31:0]															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SVTB[31:0]															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**SVTB[31:0]**	**SVPWM输出Tb值**

19.4.2.7 SVPWM 输出 Tc 值寄存器 (MACP_SVTC)

偏移地址: 0x0218

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SVTC[31:0]															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SVTC[31:0]															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**SVTC[31:0]**	**SVPWM输出Tc值**

19.4.2.8 SVPWM 象限寄存器 (MACP_SVSECTOR)

偏移地址: 0x021C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								SVSECTOR[7:0]							
-								RO							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 0	**SVSECTOR[7:0]**	**SVPWM象限**

20. 模数转换器（Analog to Digital Converter，ADC）

20.1 简介

SH33F2811包含一个单端型12位高速逐次逼近型模数转换器（ADC，Analog-to-Digit Converter），模块图如图20-1所示。ADC基准电压在芯片复位后默认使用AVDD，用户也可以选择外部VREF端口的输入作为基准电压。

ADC有16路模拟输入通道（AN0~AN10，OPxOUT(x=0,1,2,3)，Vbg），可以编入序列中自动进行转换，结果储存在对应的结果寄存器ADDRx（x = 0 - 15）中，每转换一次序列，对应结果寄存器的值更新一次，并作一次比较。结果寄存器和模拟输入通道之间的映像关系可以随意编程，从而组成一个转换序列，而且可以将某一模拟输入通道在序列中重复编程，从而在结果寄存器中获得此模拟输入通道连续多次转换的结果。

带有独有的单电阻采样模式，可以实现不同硬件源触发不同的起始通道和通道个数。同时具有单次转换模式、间断转换模式和连续转换模式。当启动连续转换功能时，序列自动循环启动转换，结果寄存器也不断更新，并将每次的更新值与极限值作比较。

对于单个通道，转换速率最高可达2MSPS，可由寄存器TADC@ADCON2设置ADC时钟频率，TS[3:0]@ADCON2设置ADC每个通道的第一段采样时间。序列中相邻通道之间的时间间隔（亦可当做每个通道的第二段采样时间）可通过寄存器TGAP[2:0]@ADCON2和GAPENx@ADGAPON来设置。

图20-1　　ADC 系统框图

20.2 主要特性

- 逐次逼近方式的12位A/D转换器
- 参考电压可选外接VREF或AVDD，并且外接VREF可以输出给其它模块，例如OP、比较器使用。
- 11路模拟输入口、4个OP输出口、一个参考电压，共16个ADC模拟输入通道
- 启动一次ADC可以自动完成多通道转换（序列），而且每个通道都可以配置为多路模拟输入中的任意一路
- 序列可配置为单通道或多通道，一个序列最多可包含16个通道，转换结果保存在16个结果寄存器中
- 存在四种转换方式：序列转换方式、间断转换方式、连续转换方式和单电阻转换方式
- 序列转换时相邻通道转换之间的时间间隔可由软件设定
- 可由一个MCM模块、两个PCA定时器、一个普通定时器(TIM8)和一个外部引脚的触发信号自动启动AD转换
- 单通道转换速率最高可达2MSPS
- 四种转换方式都带比较功能

20.3 功能

20.3.1 单次转换方式

ADCTU[1:0] = 00，ADC配置为单次序列转换模式，一次转换一个序列。序列由单个通道或多个通道组成，对序列进行转换即对序列中的通道进行逐个转换。在硬件上，使得多个信号在同一时间点上转换得以实现（2通道间最短采样间隔0.5us，可近似看作同时）。

转换的结果储存在对应的结果寄存器中，序列转换完成的同时，把指定的某个结果寄存器的值与ADDGT和ADDLT的值比较，并以标志指示比较的结果。

单次序列转换模式下， ADSOC被置起后开始从通道0开始转换，每次单通道转换结束后ADPCH寄存器自动加1，随后进行下一通道转换，当单次序列转换通道个数达到设定值(ADMAXCH[3:0])时，本次序列转换结束，ADSOC位清零，表明本次序列转换完成，同时自动复位ADPCH寄存器为0；同时ADC_ADINTF中的ADCIF位将由硬件置起，此时若ADCON1中ADCIE位为1（注意：ADC的中断不要被屏蔽），则将触发序列转换完成中断，ADCIF标志需要软件清除。

例：有4路通道需要转换，按优先级排序为AN0，AN2，AN4，AN7，则设置为ADMAXCH[3:0] = 3，SEQCH0 = 0，SEQCH1 = 2，SEQCH2 = 4，SEQCH3 = 7，则每次启动AD转换（ADSOC=1）时，随后从通道0开始的4个通道将依次转换，结果将依次存在ADDR0，ADDR1，ADDR2，ADDR3中。

图20-2　　单次转换方式波形示例图

20.3.2 间断转换方式

ADCTU[1:0] = 01，则ADC工作在间断转换工作模式下，ADC直接从由ADPCH寄存器指定的通道开始转换，当转换个数达到(ADMAXCH[3:0])设定值时，本次序列转换结束。ADC采样通道序号最大为15，因此当转换通道指针寄存器ADPCH超过15时，会自动归0。一个序列转换结束后，如果此时将ADPCH软件改写为n，则下次转换从通道n开始；若不改写ADPCH，则下次转换通道从上次采样结束时的ADPCH值开始，硬件采样结束并不会清零ADPCH的值因此程序对ADPCH寄存器的改写要小心处理（ADC转换期间（ADSOC =1），ADPCH无法修改）。

例如，SEQCH0=2，SEQCH1=4，SEQCH2=7，SEQCH3=3，SEQCH4=5，SEQCH5=6，SEQCH14=1，SEQCH15=0，当ADMAXCH=3时，若当前ADPCH=14，在间断转换方式下，SEQCH14，SEQCH15，SEQCH0和SEQCH1为有效的通道选择寄存器，由此组成的通道序列为AN1，AN0，AN2和AN4，采样结果将依次存在ADDR14，ADDR15，ADDR0，ADDR1中。转换结束后ADPCH值应该为2，而再次启动AD转换时，AN7，AN3，AN5和AN6将依次转换，结果将依次存在ADDR2，ADDR3，ADDR4，ADDR5中，转换结束后ADPCH值应该为6，以此类推。该示例中，各关键信号的时序关系如图20-3所示：

图20-3　　单序列顺序采样模式，间断转换方式的波形示例图

注1：ADPCH只能在ADC不采样（ADSOC等于0）时，同时在间断或单电阻模式下才能被软件写入。
注2：ADPCH在每个通道转换完成把采样结果送出时更改，也就是说ADPCH更改时刻与ADC结果送出时刻一致。

20.3.3 连续转换方式

ADCTU[1:0] = 10选择连续转换方式，转换完一个序列便进行此序列的下一次转换，一直循环此序列转换。连续转换的一些寄存器设置可参考“单次转换方式”。每次序列转换完成的同时，把指定的某个结果寄存器的值与ADDGT和ADDLT的值比较，并以标志指示比较的结果。当软件清除ADSOC位时，AD转换立即停止。

例如，SEQCH0=2，SEQCH1=5，SEQCH2=0，SEQCH3=4，…，SEQCH6=1，SEQCH7=6，当ADMAXCH=3时，在转换开始后， SEQCH0，SEQCH1，SEQCH2和SEQCH3为有效的通道选择寄存器，由此组成的通道序列为AN2，AN5，AN0，AN4。该示例中，各关键信号的时序关系如图20-4所示：

注意：ADPCH的值在采样结束时被清零。

图20-4　　连续转换方式波形示例图

20.3.4 单电阻转换方式

ADCTU[1:0] = 11，则ADC工作在单电阻转换工作模式下。当不同的硬件触发ADC时，直接从由相应的硬件触发源对应的HARDCHSEL寄存器指定通道开始转换，当转换个数达到该硬件触发源设置的转换通道个数HARDCHLH值时，本次序列转换结束。当软件触发ADC时，ADC直接从由ADPCH寄存器指定的通道开始转换，当转换个数达到(ADMAXCH[3:0])设定值时，本次序列转换结束。采样通道序号最大为15，因此当转换通道指针寄存器ADPCH超过15时，会自动归0。采样结束并不会清零ADPCH的值，因此程序对ADPCH寄存器的改写要小心处理（ADC转换期间（ADSOC =1），ADPCH无法修改）。

例如，在单电阻模式下， SEQCH7=3，SEQCH8=5，SEQCH9=6，SEQCH14=1，SEQCH15=0，PWMCMP1@HARDCHLH=1，PWMCMP2@HARDCHLH=2，PWMCMP1@HARDCHSEL=14，PWMCMP2@HARDCHSEL=7。当MCM的PWMCMP1触发ADC时，SEQCH14和SEQCH15为有效的通道选择寄存器，由此组成的通道序列为AN1和AN0，采样结果将依次存在ADDR14和ADDR15中。当MCM的PWMCMP2触发ADC时，SEQCH7，SEQCH8，SEQCH9为有效的通道选择寄存器，由此组成的通道序列AN3，AN5和AN6将依次转换，结果将依次存在ADDR7，ADDR8，ADDR9中，以此类推。该示例中，各关键信号的时序关系如图20-5所示：

图20-5　　单电阻模式转换方式的波形示例图

20.3.5 其它功能描述

20.3.5.1 序列转换中通道的选择设置

一个序列可包含单个或多个通道，将待转换的通道号分别存放在通道寄存器SEQCHx中（x = 0 - 15），寄存器SEQCHx有16组，所以序列转换时，一次最多可转换16路通道。而每次转换的通道数量由寄存器ADCON2中的ADMAXCH[3:0]的值决定。例：ADMAXCH[3:0] = 0，即单通道转换，将SEQCH0中存放的通道进行转换；ADMAXCH[3:0] = 3，则序列中有4路通道，将SEQCH0到SEQCH3中存放的通道依次转换。

通道寄存器SEQCHx（x = 0 - 15）中存放的即是待转换的通道号。例：有3路通道需要转换，按优先级排序为OP1OUT，AN2，AN7，则设置为ADMAXCH[3:0] = 2，SEQCH0 = 12，SEQCH1 = 2，SEQCH2 = 7，将依次转换。

注1：需要转换的模拟通道IO复用必须配置为模拟功能（GPIOx_CFG_AFR设成AF7）。SEQCHx中也可设置相同的通道号，比如将SEQCHx中的值全设置为AN3，结果寄存器中储存的将是不同时间段的AN3的转换值。

注2：在ADC采样时（ADSOC=1）时，SEQCHx不允许被修改。

20.3.5.2 序列转换时通道间隔（Gap Time）设置

序列转换时，从上一个通道转换完毕到下一个通道开始采样时刻之间的时间可以通过寄存器TGAP[2:0]@ADCON2和GAPENx@ADGAPON位段来设置。当TGAP[2:0] = 0或GAPENx=0时，一个通道转换完毕后立即开始下一个通道的采样，之间没有等待时间。

注：第一个通道也可以设置Gap Time，此时间亦可当做采样时间。

每个通道的Gap Time都可以单独设置，所以Gap Time亦可当做每个通道的采样时间。所有通道的Gap Time只能设置成一个值，但是每个通道可单独设置是否使能Gap Time。

20.3.5.3 序列转换模式的启动和停止

通过将ADCON1中的ADON位置1，这样可以使能ADC模块的时钟ADC_CLK，同时令ADON位置1可以令ADC模块中的模拟电路上电。序列转换模式的启动可以分为软件启动和硬件启动。

软件启动，当ADCON1寄存器中的ADSOC位置1，则启动转换。当一次序列转换完成，由硬件将ADSOC清0，同时进行比较（在20.3.6节中详细阐述）。当读取ADSOC为1时，则标志转换进行未完成。如果在转换过程中清零ADSOC位，则终止转换。

硬件启动有4种方式，外部引脚上升沿触发启动、MCM模块触发信号触发启动、PCA模块触发信号触发启动和普通定时器(TIM8)模块触发信号触发启动。外部引脚触发ADC的使能位为ADTGREN@ADCON1，其他模块的触发ADC使能在各自模块中。外部触发引脚（ADTRG）固定为上升沿启动ADC，而且固定滤波系数，要保持2个PCLK时钟。

注1：置起ADCON1中的ADON位置1后，需要等待10us之后再开始采样。

注2：任何硬件触发单次序列、间断转换和连续转换过程中，又来了一次硬件触发，则后一次硬件触发被忽略。

20.3.5.4 序列转换完成中断

序列转换完毕后，ADIF@ADINTF将由硬件置起，此时若对应的ADIE@ADCON1为1，则将触发序列转换完成中断，ADIF@ADINTF位只能由软件把ADIFC@ADINTF置1清除。

20.3.6 比较功能

20.3.6.1 比较用结果寄存器的指定

由寄存器CSEL@ADCMPCON来指定被比较的结果寄存器，若CSEL = n，则结果寄存器ADDRn将与ADDGT和ADDLT的值比较。若设置CSEL = 2，每次序列转换完成的同时将ADDR2的值与ADDGT和ADDLT的值比较。

需要注意的是，若指定一个序列转换不使用的结果寄存器，则不会发生任何比较动作。如上例中令CSEL = 3，转换后未使用结果寄存器ADDR3，故没有任何比较动作发生，标志位ADGIF和ADLIF的值不会发生变化。

***注意：** ADDGT和ADDLT写入值立即生效，比较时会采用ADDGT和ADDLT最近一次的更新值。*

20.3.6.2　比较过程

设CSEL = n，ADDGT = Max，ADDLT = Min，当序列转换完成时，立即将指定结果寄存器ADDRn的值与Max和Min比较。若ADDRn >= Max，则ADCMPCON寄存器中ADGIF位将置1，若ADGIE位为1，则可以触发*一个比较中断*（与序列转换完成共享一个中断向量），ADGIF位将保持为1直到软件将其清除；若ADDRn <= Min，则ADCMPCON寄存器中ADLIF位将置1，若ADLIE位为1，也可以触发一个*比较中断*（与序列转换完成共享一个中断向量），ADLIF位将保持为1直到软件将其清除。

设ADDGT = 0x3DF，ADDLT = 0x8FE，CSEL = n，则ADDRn的值与ADGIF、ADLIF位中值的关系如图20-6的左半部分所示；设ADDGT = 0x8FE，ADDLT = 0x3DF，CSEL = n，则ADDRn的值与ADGIF、ADLIF位中值的关系如图20-6的右半部分所示。

图20-6　　比较过程图

20.3.6.3　比较中断

每次序列转换完成时，都会将指定结果寄存器的值去作比较，如果超限，可以产生一个比较中断。因为比较和序列转换完成是同时发生的，所以中断的产生也是同时。

20.3.7　ADC 转换时间设置

通过寄存器ADCON2可以设置ADC的时钟以及采样时间。通过TADC[3:0]@ADCON2设置ADC的时钟。通过寄存器TS[3:0]@ADCON2、TGAP[2:0]@ADCON2和GAPENx@ADGAPON设置每个通道的采样时间t_{SAMP}，t_{SAMP} = (TS[3:0]+TGAP[2:0]* GAPEN+1) * t_{AD}，具体见ADC寄存器章节。

对于12BIT模式AD转换每个通道的AD转换时间固定为15 * t_{AD}。因此每个通道的总共转换时间 = t_{SAMP} + 15 * t_{AD}。

ADC时钟最快设置成40MHz。

20.3.8　ADC 模块参考电压的设置

可以选择芯片V_{DD}或者V_{REF}引脚上的输入电压作为ADC模块的参考电压。芯片复位后，ADC模块的参考电压为芯片的V_{DD}。令寄存器ADCON1中的REFC位置1将使ADC模块使用V_{REF}引脚上的电压作为参考电压。

20.3.9　ADC 通道与 IO 口功能设置

ADC输入I/O口设为模拟口，具体设置请参考“GPIO”章节。

20.3.10 AD 转换时对传感器输出阻抗的要求

如图20-7所示，为了保证AD转换在一定的转换速率的条件下能转换出精确的AD结果，需要在一定时间内（采样时间Tsamp）对芯片内部采样电容C完成充电。如果在规定的采样时间内，采样电容C未能充满电，即V0 < V1，AD转换结果则会产生精度误差。因为R0（芯片内部电阻）和C（内部采样电容）是常量，已经由芯片内定，能否在转换出允许精度范围内的AD结果就取决于传感器内部等效电阻Rx。设AD转换结果的分辨率为M（12bit结果的分辨率为4096），如果AD转换的精度要求为N，则对传感器电阻的选择有如下要求：

图20-7　　输出阻抗

$$Rx \le -\frac{T}{5\times10^{-12} * Ln\frac{V1-V0}{V1}} - 0.8\times10^{3}$$

上式也可表示为：

$$Rx \le -\frac{T}{5\times10^{-12} * Ln\frac{\mathrm{N}}{\mathrm{M}}} - 0.8\times10^{3}$$

对于12bit模式下即4096分辨率，假如ADC时钟为40M，TS设置成0xF，Tgap关闭，则采样时间为16*Tadc，为0.4us，理论精度误差控制在0.1LSB内时，按上述公式代入有

$$Rx \le -\frac{0.4\times10^{-6}}{5\times10^{-12} Ln\frac{0.1}{4096}} - 0.8\times10^{3} \approx 6.73K\Omega$$

即要满足精度误差在0.1LSB之内，传感器选型时的内部等效电阻要小于6.73KΩ。

注：在上述配置下，并不是说大于6.73 KΩ阻抗的信号源不能使用，只是理论精度误差会超出0.1LSB。理论上阻抗越大精度误差越大。具体可以根据上述公式自行计算。降低采样速率，理论等效阻抗上限相应提高。

20.3.11 注意事项

ADC转换，最好将ADON@ADCON1打开10us后再将ADSOC@ADCON1位置1，进行转换。因为ADC模块打开后有个稳定时间。

在ADSOC为1时，对ADCON2、ADPCH、SEQCHSEL0/1、ADGAPON这些寄存器操作无效。

如需要转换增益放大通路，需要先打开OP功能，OP需要建立时间（约为100us）才能稳定输出稳定，再进行AD转换，转换时选取OPOUT0、OPOUT1、OPOUT2和OPOUT3通路。

转换模拟通路时要先将端口设置为AD通道功能才能顺利转换。

当ADIE，ADLIE，ADGIE的都开启时，ADIF，ADLIF，ADGIF任意一个置位都能引起中断，且共享一个中断向量，通过判位ADIF，ADLIF，ADGIF哪个为1，来确定具体的中断源并定制特定的操作。

20.4 寄存器

ADC 模块寄存器列表 (基地址：0x4000 3C00)

地址	寄存器名	说明
0x4000 3C00	ADCON1	ADC 控制寄存器 1
0x4000 3C04	ADCON2	ADC 控制寄存器 2
0x4000 3C08	ADPCH	ADC 采样转换通道指针寄存器
0x4000 3C0C	ADDR0	ADC 结果寄存器 0
0x4000 3C10	ADDR1	ADC 结果寄存器 1
0x4000 3C14	ADDR2	ADC 结果寄存器 2
0x4000 3C18	ADDR3	ADC 结果寄存器 3
0x4000 3C1C	ADDR4	ADC 结果寄存器 4
0x4000 3C20	ADDR5	ADC 结果寄存器 5
0x4000 3C24	ADDR6	ADC 结果寄存器 6
0x4000 3C28	ADDR7	ADC 结果寄存器 7
0x4000 3C2C	ADDR8	ADC 结果寄存器 8
0x4000 3C30	ADDR9	ADC 结果寄存器 9
0x4000 3C34	ADDR10	ADC 结果寄存器 10
0x4000 3C38	ADDR11	ADC 结果寄存器 11
0x4000 3C3C	ADDR12	ADC 结果寄存器 12
0x4000 3C40	ADDR13	ADC 结果寄存器 13
0x4000 3C44	ADDR14	ADC 结果寄存器 14
0x4000 3C48	ADDR15	ADC 结果寄存器 15
0x4000 3C4C	ADCMPCON	ADC 比较控制寄存器
0x4000 3C50	ADDGT	ADC 上限比较寄存器
0x4000 3C54	ADDLT	ADC 下限比较寄存器
0x4000 3C58	SEQCHSEL0	ADC 通道选择寄存器 1
0x4000 3C5C	SEQCHSEL1	ADC 通道选择寄存器 2
0x4000 3C60	HARDCHSEL	单电阻模式下，硬件触发 ADC 的起始通道选择位
0x4000 3C64	HARDCHLH	单电阻模式下，硬件触发 ADC 的通道长度设置位
0x4000 3C68	ADGAPON	ADC 通道 GAP 控制位
0x4000 3C6C	ADINTF	ADC 中断标志和清除寄存器

20.4.1 ADC 控制寄存器 1 (ADC_ADCON1)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留							ADOUT	ADON	ADIE	ADTGREN	REFC	保留	ADCTU[1:0]		ADSOC
-							RW	RW	RW	RW	RW	-	RW		RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 9	保留	-
8	**ADOUT**	**采样输出位：** 0：禁止通过 ADTRG 引脚输出 ADSOC 状态 1：使能通过 ADTRG 引脚输出 ADSOC 状态
7	**ADON**	**ADC 允许位** 0：禁止 ADCx 模块 1：允许 ADCx 模块
6	**ADIE**	**通道序列转换结束中断使能位** 0：禁止 ADC 转换结束中断 1：使能 ADC 转换结束中断
5	**ADTGREN**	**外部 ADTRG 引脚触发 ADC 使能位** 0：禁止 1：使能
4	**REFC**	**基准电压选位** 0：选择 VDD 为基准电压 1：选择外部 VREF 端口输入为基准电压
3	保留	-
2 ~ 1	**ADCTU[1:0]**	**ADC 通道序列的转换方式控制位** 00：单次序列转换方式 01：间断序列转换方式 10：连续转换方式 11：单电阻采样模式（硬件触发 ADC 起始通道由 HARDCHSEL 选择，采样通道个数由 HARDCHLH 设置）。在此模式下，ADPCH 还是随采样通道的改变而改变。并且软件触发 ADC 模式为间断序列转换方式。 注：修改该控制位，会在下一次启动 AD 转换过程时才产生效果
0	**ADSOC**	**ADC 通道序列启动 AD 转换请求位** 0：未进行序列转换/连续转换或已完成。写 0 可取消现在正在进行的转换。 1：在单次和间断转换方式下，置 1 开始序列转换，在转换过程中保持为 1，转换完成后硬件自动清 0（若转换期间清 0 此位会立即终止序列转换。） 在连续转换方式下，置 1 开始连续转换，则该位不会由硬件清 0，可以由软件清 0 立即终止转换

20.4.2 ADC 控制寄存器 2 (ADC_ADCON2)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留	TGAP [2:0]			TS [3:0]				ADMAXCH[3:0]				TADC[3:0]			
-	*RW*			*RW*				*RW*				*RW*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 15	保留	-
14 ~ 12	**TGAP [2:0]**	**序列中时相邻通道之间时间间隔设置位段** 在一次序列转换中，一个通道转换完毕到下一个通道开始采样之间的时间间隔 000：无等待时间 001：2ADC 时钟周期 010：4ADC 时钟周期 011：8ADC 时钟周期 100：16ADC 时钟周期 101：32ADC 时钟周期 110：64ADC 时钟周期 111：128ADC 时钟周期
11 ~ 8	**TS [3:0]**	**采样时间设置:** 1 tAD ≤ (TS [3:0]+1) * tAD ≤ 16 tAD 注 1：一个通道的总采样时间由两部分组成： TS + TGAP， 此处设置 TS 部分； 注 2： 总采样时间范围：1 tAD ≤ ((TS[3:0])+1+TGAP[2:0]*GAPEN)* tAD ≤ 144 tAD， 其中 TGAP 由 GAPEN 控制是否使能； 注 3：在设置 TS[3:0]前，需估算连接到 ADC 输入引脚的串联电阻，以获得最佳采样精度，当选择 1 * tAD 为采样时间时，请确保连接到 ADC 输入引脚的串联电阻满足输入阻抗需求，详见 20.3.10； 注 4：TADC 设置需保证 ADC 时钟周期 Tadc ≥ 25ns 注 5：12bit 模式一个通道的总共采样转换时间 = 15tAD + 采样时间
7 ~ 4	**ADMAXCH[3:0]**	**ADC 通道序列的总长度设置位（设置值 0~15）：** 一次 AD 转换过程的模拟通道总数为 ADMAXCH1+1
3 ~ 0	**TADC[3:0]**	**ADC 时钟周期选择** 0000：ADC 时钟周期 t_{AD} = 1 t_{PCLK} 0001：ADC 时钟周期 t_{AD} = 2 t_{PCLK} 0010：ADC 时钟周期 t_{AD} = 3 t_{PCLK} 0011：ADC 时钟周期 t_{AD} = 4 t_{PCLK} 0100：ADC 时钟周期 t_{AD} = 5 t_{PCLK} 0101：ADC 时钟周期 t_{AD} = 6 t_{PCLK} 0110：ADC 时钟周期 t_{AD} = 8 t_{PCLK} 0111：ADC 时钟周期 t_{AD} = 12 t_{PCLK} 1000：ADC 时钟周期 t_{AD} = 16 t_{PCLK} 1001：ADC 时钟周期 t_{AD} = 24 t_{PCLK} 1010：ADC 时钟周期 t_{AD} = 32 t_{PCLK} 1011：ADC 时钟周期 t_{AD} = 48 t_{PCLK} 1100：ADC 时钟周期 t_{AD} = 64 t_{PCLK} 1101：ADC 时钟周期 t_{AD} = 128 t_{PCLK} 1110：ADC 时钟周期 t_{AD} = 256 t_{PCLK} 1111：ADC 时钟周期 t_{AD} = 320 t_{PCLK}

20.4.3 ADC 采样转换通道指针寄存器 (ADC_ADPCH)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留												ADPCH[3:0]			
-												*RW*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 4	保留	-
3 ~ 0	**ADPCH[3:0]**	**ADC 采样转换通道指针寄存器** ADC 当前采样通道为 SEQCHn(n=ADPCH[3:0]) 注：ADPCH 只能在 ADC 不采样（ADSOC 等于 0）时，能够被软件写入.

20.4.4 ADC 结果寄存器 0 (ADC_ADDRn)(n=0..15)

偏移地址: 0x000C
:0x0010
:0x0014
:0x0018
:0x001C
:0x0020
:0x0024
:0x0028
:0x002C
:0x0030
:0x0034
:0x0038
:0x003C
:0x0040
:0x0044
:0x0048

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留				ADDRn[11:0]											
-				*RO*											
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 12	保留	-
11 ~ 0	**ADDRn[11:0]**	一个通道转换完毕后，数据立即更新并存放在 **ADDRx（x = 0）**中序列转换一次所有结果寄存器也都更新了一次

20.4.5 ADC 比较控制寄存器 (ADC_ADCMPCON)

偏移地址: 0x004C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								ADGIE	保留	ADLIE	保留	CSEL[3:0]			
-								*RW*	-	*RW*	-	*RW*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7	**ADGIE**	上限比较中断使能位 0：禁止上限比较中断 1：使能上限比较中断
6	保留	-
5	**ADLIE**	下限比较中断使能位 0：禁止下限比较中断 1：使能下限比较中断
4	保留	-
3 ~ 0	**CSEL[3:0]**	用于比较的结果寄存器选择位段 选择 ADDRn 中的值与 ADDGT、ADDLT 比较

20.4.6 ADC 上限比较寄存器 (ADC_ADDGT)

偏移地址: 0x0050

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
GT[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**GT[15:0]**	**ADC 上限比较寄存器** 当一次序列转换完成时，ADCMPCON 寄存器中 CSEL 位段指定结果寄存器的值立刻与 ADDGT 中的值比较，若大于等于 ADDGT 中的值则令 ADCMPCON 中的 ADGIF 位置 1，ADGIF 位将保持为 1 直到软件将其清除

20.4.7 ADC 下限比较寄存器 (ADC_ADDLT)

偏移地址: 0x0054
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
LT[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**LT[15:0]**	**ADC 下限比较寄存器** 当一次序列转换完成时，ADCMPCON 寄存器中 CSEL 位段指定结果寄存器的值立刻与 ADDLT 中的值比较，若小于等于 ADDLT 中的值则令 ADCMPCON 中的 ADLIF 位置 1，ADLIF 位将保持为 1 直到软件将其清除

20.4.8 ADC 通道选择寄存器 1 (ADC_SEQCHSEL0)

偏移地址: 0x0058
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SEQCH7[3:0]				SEQCH6[3:0]				SEQCH5[3:0]				SEQCH4[3:0]			
RW				*RW*				*RW*				*RW*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SEQCH3[3:0]				SEQCH2[3:0]				SEQCH1[3:0]				SEQCH0[3:0]			
RW				*RW*				*RW*				*RW*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 28	**SEQCH7[3:0]**	模拟通道 7 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg

27 ~ 24	**SEQCH6[3:0]**	**模拟通道 6 选择寄存器组，分别指定对应的模拟输入通道：** 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg
23 ~ 20	**SEQCH5[3:0]**	**模拟通道 5 选择寄存器组，分别指定对应的模拟输入通道：** 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg
19 ~ 16	**SEQCH4[3:0]**	**模拟通道 4 选择寄存器组，分别指定对应的模拟输入通道：** 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg

15 ~ 12	**SEQCH3[3:0]**	模拟通道 **3** 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg
11 ~ 8	**SEQCH2[3:0]**	模拟通道 **2** 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg
7 ~ 4	**SEQCH1[3:0]**	模拟通道 **1** 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg

3 ~ 0	**SEQCH0[3:0]**	模拟通道 **0** 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg

20.4.9 ADC 通道选择寄存器 2 (ADC_SEQCHSEL1)

偏移地址: 0x005C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SEQCH15[3:0]				**SEQCH14[3:0]**				**SEQCH13[3:0]**				**SEQCH12[3:0]**			
RW				*RW*				*RW*				*RW*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SEQCH11[3:0]				**SEQCH10[3:0]**				**SEQCH9[3:0]**				**SEQCH8[3:0]**			
RW				*RW*				*RW*				*RW*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 28	**SEQCH15[3:0]**	模拟通道 **15** 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg

27 ~ 24	**SEQCH14[3:0]**	模拟通道 **14** 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg
23 ~ 20	**SEQCH13[3:0]**	模拟通道 **13** 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg
19 ~ 16	**SEQCH12[3:0]**	模拟通道 **12** 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg

15 ~ 12	**SEQCH11[3:0]**	模拟通道 **11** 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg
11 ~ 8	**SEQCH10[3:0]**	模拟通道 **10** 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg
7 ~ 4	**SEQCH9[3:0]**	模拟通道 **9** 选择寄存器组，分别指定对应的模拟输入通道： 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg

3 ~ 0	**SEQCH8[3:0]**	**模拟通道 8 选择寄存器组，分别指定对应的模拟输入通道：** 0000 - AN0 0001 - AN1 0010 - AN2 0011 - AN3 0100 - AN4 0101 - AN5 0110 - AN6 0111 - AN7 1000 - AN8 1001 - AN9 1010 - AN10 1011 - OP0OUT/AN11 1100 - OP1OUT/AN12 1101 - OP2OUT/AN13 1110 - OP3OUT/AN14 1111 - Vbg

20.4.10 单电阻模式下，硬件触发 ADC 的起始通道选择位 (ADC_HARDCHSEL)

偏移地址: 0x0060
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
ADTRG[3:0]				**TIM8SEL[3:0]**				**PCA1SEL[3:0]**				**PCA0SEL[3:0]**			
RW				*RW*				*RW*				*RW*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMCMP4[3:0]				**PWMCMP3[3:0]**				**PWMCMP2[3:0]**				**PWMCMP1[3:0]**			
RW				*RW*				*RW*				*RW*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 28	**ADTRG[3:0]**	**外部引脚 ADTRG 触发 ADC 起始通道选择位：** 从通道 n 开始采样(n=ADTRG[3:0])
27 ~ 24	**TIM8SEL[3:0]**	**TIM8 触发 ADC 起始通道选择位：** 从通道 n 开始采样(n=TIM8[3:0])
23 ~ 20	**PCA1SEL[3:0]**	**PCA1 触发 ADC 起始通道选择位：** 从通道 n 开始采样(n=PCA1[3:0])
19 ~ 16	**PCA0SEL[3:0]**	**PCA0 触发 ADC 起始通道选择位：** 从通道 n 开始采样(n=PCA0[3:0])
15 ~ 12	**PWMCMP4[3:0]**	**MCM 的 PWMCMP4 触发 ADC 起始通道选择位：** 从通道 n 开始采样(n=MCMCMP4[3:0])
11 ~ 8	**PWMCMP3[3:0]**	**MCM 的 PWMCMP3 触发 ADC 起始通道选择位：** 从通道 n 开始采样(n=MCMCMP3[3:0])
7 ~ 4	**PWMCMP2[3:0]**	**MCM 的 PWMCMP2 触发 ADC 起始通道选择位：** 从通道 n 开始采样(n=MCMCMP2[3:0])
3 ~ 0	**PWMCMP1[3:0]**	**MCM 的 PWMCMP1 触发 ADC 起始通道选择位：** 从通道 n 开始采样(n=MCMCMP1[3:0])

20.4.11 单电阻模式下，硬件触发 ADC 的通道长度设置位 (ADC_HARDCHLH)

偏移地址: 0x0064

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
ADTRG[3:0]				TIM8LH[3:0]				PCA1LH[3:0]				PCA0LH[3:0]			
RW				RW				RW				RW			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PWMCMP4[3:0]				PWMCMP3[3:0]				PWMCMP2[3:0]				PWMCMP1[3:0]			
RW				RW				RW				RW			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 28	**ADTRG[3:0]**	**外部引脚 ADTRG 触发 ADC 的通道个数设置位：** 采样 n+1 个通道(n=ADTRG[3:0])
27 ~ 24	**TIM8LH[3:0]**	**TIM8 触发 ADC 的通道个数设置位：** 采样 n+1 个通道(n=TIM8[3:0])
23 ~ 20	**PCA1LH[3:0]**	**PCA1 触发 ADC 的通道个数设置位：** 采样 n+1 个通道(n=PCA1[3:0])
19 ~ 16	**PCA0LH[3:0]**	**PCA0 触发 ADC 的通道个数设置位：** 采样 n+1 个通道(n=PCA0[3:0])
15 ~ 12	**PWMCMP4[3:0]**	**MCM 的 PWMCMP4 触发 ADC 的通道个数设置位：** 采样 n+1 个通道(n=PWMCMP4[3:0])
11 ~ 8	**PWMCMP3[3:0]**	**MCM 的 PWMCMP3 触发 ADC 的通道个数设置位：** 采样 n+1 个通道(n=PWMCMP3[3:0])
7 ~ 4	**PWMCMP2[3:0]**	**MCM 的 PWMCMP2 触发 ADC 的通道个数设置位：** 采样 n+1 个通道(n=PWMCMP2[3:0])
3 ~ 0	**PWMCMP1[3:0]**	**MCM 的 PWMCMP1 触发 ADC 的通道个数设置位：** 采样 n+1 个通道(n=PWMCMP1[3:0])

20.4.12 ADC 通道 GAP 控制位 (ADC_ADGAPON)

偏移地址: 0x0068

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
GAPENy(y=15~0)															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**GAPENy(y=15~0)**	**ADC 通道 y 的 GAP 时间使能位：** 0：禁止 1：使能

20.4.13 ADC 中断标志和清除寄存器 (ADC_ADINTF)

偏移地址: 0x006C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留													ADIFC	ADGIFC	ADLIFC
-													*WO*	*WO*	*WO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留													ADIF	ADGIF	ADLIF
-													*RO*	*RO*	*RO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 19	保留	-
18	**ADIFC**	**两路独立顺序采样模式下，SHB 通道序列转换结束中断标志位清除位** 0：无效 1：清除
17	**ADGIFC**	**上限比较中断标志位清除位** 0：无效 1：清除
16	**ADLIFC**	**下限比较中断标志位清除位** 0：无效 1：清除
15 ~ 3	保留	-
2	**ADIF**	**两路独立顺序采样模式下，SHB 通道序列转换结束中断标志位：** 0：无通道序列转换结束中断发生 1：有通道序列转换结束中断发生
1	**ADGIF**	**上限比较中断标志位** 0：无上限中断发生，CSEL 位段所指定结果寄存器中的最近一次更新值小于 ADDGT 中的值 1：发生上限中断，CSEL 位段所指定结果寄存器中的最近一次更新值大于等于 ADDGT 中的值 该标志置起后需由软件清 0
0	**ADLIF**	**下限比较中断标志位** 0：无下限中断发生，CSEL 位段所指定结果寄存器中的最近一次更新值大于 ADDLT 中的值 1：发生下限中断，CSEL 位段所指定结果寄存器中的最近一次更新值小于等于 ADDLT 中的值 该标志置起后需由软件清 0

21. 正交编码器接口模块（QEI）

21.1 简介

电机控制系统中常使用增量式旋转编码器来实时监测旋转系统的位置和速度。编码器通常有3个输出信号：两个相位互差90度的方波信号QEA、QEB和一个索引信号INDEX。实际应用中，编码器通常安装在旋转轴，如电机转子上，编码器输出的QEA/QEB信号频率与电机旋转速率成正比，如一个1000线的编码器安装在转速为6000rpm的电机上，编码器将产生100KHz的QEA和QEB信号，因此通过测量QEA或QEB输出即可确定电机的转速。

大多数编码器在顺时针旋转时输出的QEA信号超前QEB信号，逆时针旋转时输出的QEA信号滞后QEB信号，每旋转一周输出一个INDEX脉冲信号指示一个绝对位置。索引信号INDEX的脉冲宽度因编码器不同而不同，大体分为门控索引信号和非门控索引信号两大类。门控索引信号的脉冲上升和下降沿必定与正交信号QEA/QEB的4个边沿中的某2个对齐，脉冲宽度为1/4、1/2或整个QEA/QEB信号的周期，而非门控索引信号的边沿与QEA/QEB的边沿无绝对联系。

编码器输出的正交信号QEA/QEB可以有四种状态，典型的增量式旋转编码器输出信号如下图所示：

图21-1 增量式旋转编码器输出信号

下图提供了逻辑框图：

图21-2 正交编码器接口模块 QEI 逻辑框图

21.2 主要特性

- 3个输入引脚：两个相位信号QEA，QEB和一个索引脉冲信号INDEX
- 16位增/减双向位置计数器
- 2种位置计数模式：x2模式和x4模式
- 输入引脚的数字滤波功能
- 提供低转速测量和高转速测量两种应用方案
- 包含一个32位Qtimer

21.3 功能

21.3.1 输入滤波

为提高系统抗干扰能力，QEI模块为3个输入引脚QEA、QEB和INDEX提供了内部数字滤波电路。用户通过QABFEN和QIDXFEN分别控制QEA/QEB和INDEX引脚是否使用滤波功能。

当QABFEN=1时，引脚QEA和QEB的滤波电路有效，QEA和QEB引脚输入信号的电平必须保持至少3个滤波时钟周期的稳定状态才会使滤波输出信号发生变化。滤波时钟由APB时钟分频（PCLK）分频得到，分频值通过QABCPS设置。

当QIDXFEN=1时，引脚INDEX的滤波电路有效，INDEX引脚输入信号的电平必须保持至少3个滤波时钟周期的稳定状态才会使滤波输出信号发生变化。滤波时钟由APB时钟分频（PCLK）分频得到，分频值通过QIDXCPS设置。

当QEA、QEB和INDEX引脚上采样不同的滤波设置状态时，会导致后级电路的三个输入信号的相位发生变化，用户需要合理设置滤波功能，以保证后级电路的工作结果符合设计意图。

为描述方便，后面各节均以关闭滤波电路，且3个输入信号的相位为理想情况进行说明。

21.3.2 正交解码器

QEI模块内建的正交解码器，用于对引脚QEA、QEB的输入信号进行解码，以确定增量式旋转编码器的工作状态。

21.3.2.1 正交解码器的接口信号

从引脚QEA/QEB输入的信号经过滤波电路后首先受QSWAP位控制，再输入给正交解码器进行解码，修改QSWAP控制位可以满足不同的增量式旋转编码器的应用需要。

QSWAP=0，引脚输入的QEA和QEB信号未作交换直接进行解码，常用于QEA相位超前QEB对应正转的旋转编码器；QSWAP=1，引脚输入的QEA和QEB信号先作交换后再进行解码，可用于QEA相位超前QEB对应反转的旋转编码器。以下各节均以QSWAP=0的设置为例进行说明，QSWAP=1的情况可以此类推。

正交解码器输出QCLK信号作为位置计数器QPOSCNT的计数时钟，QCLK的产生受QPMOD[1]控制，不同设置条件下，正交解码器提供了2种QCLK产生模式：x2模式和x4模式。

正交解码器输出QPDIR指示计数方向，用户通过软件读取同名寄存器可判断计数方向。QPDIR=0，QPOSCNT对QCLK进行减计数；QPDIR=1，QPOSCNT对QCLK进行增计数。

21.3.2.2 旋转方向与计数方向

正交解码器根据输入QEA/QEB信号的状态，判断增量式旋转编码器的旋转方向。以[QEA:QEB]为状态字，可能的状态转换过程如下图所示，其中各图符的说明如下：

F 表示QEA相位超前QEB，编码器正向旋转，[QEA:QEB]状态字按照图示顺时针方向转换，该过程中QPDIR将输出1；

R 表示QEB相位超前QEA，编码器反向旋转，[QEA:QEB]状态字按照图示逆时针方向转换，该过程中QPDIR将输出0；

PE 表示QEA与QEB同相位或反相位（编码器正常工作时不应该出现图示对角线状态间的转换），此时解码器会判断出现相位错误，置位中断标志QPEIF。

S 表示QEA与QEB的相位关系没有发生变化。

图21-3 QEA/QEB 状态转换图

21.3.3 位置计数器

QEI模块内建16位位置计数器QPOSCNT，对正交解码器输出的QCLK进行增/减双向计数，计数方向由QPDIR控制，计数模式和复位方式受QPMOD控制。

QPMOD[1]控制位提供2种计数模式，也即QCLK产生模式：x2模式和x4模式。

QPMOD[0]控制位提供2种复位方式：自动循环复位和INDEX索引复位。

21.3.3.1 位置计数器的计数模式

当QPMOD=00/01时，位置计数器工作于x2计数模式，该模式下QCLK在QEA的上升沿和下降沿产生一个脉冲，因此计数器对QEA的双沿进行计数。x2计数模式下计数器的工作时序如下图所示：

图21-4 x2 计数模式下 QPOSCNT 的工作时序

当QPMOD=10/11时，位置计数器工作于x4计数模式，该模式下QCLK在QEA和QEB的上升沿和下降沿均产生一个脉冲，因此计数器对QEA和QEB的双沿进行计数。x4计数模式下计数器的工作时序如下图所示：

图21-5 x4 计数模式下 QPOSCNT 的工作时序

21.3.3.2 位置计数器的复位方式

当QPMOD=00/10时，位置计数器使用索引脉冲INDEX复位方式。该复位方式下，硬件会自动检测INDEX引脚的输入信号，每次INDEX由0变1后出现的第一个满足条件的QCLK计数脉冲，会触发一次复位动作。如果复位动作发生时QPOSCNT处于增计数过程，则QPOSCNT自动复位为0；如果此时QPOSCNT处于减计数过程，则QPOSCNT自动重载QPOSMAX设置值。复位动作还受QIDXEN控制，当QIDXEN=1时，所有复位动作有效，在第一次复位动作后，QPOSCNT将在0~QPOSMAX范围作循环计数。当QIDXEN=0时，所有复位动作均无效，QPOSCNT增计数到QPOSMAX后自动复位为0，减计数到0后自动复位为QPOSMAX，在0~QPOSMAX范围作循环计数。当发生复位动作时，不管QIDXEN设置状态如何，硬件均会置位QEIIF中断标志。

为保证正反转过程计数值的对称性，每次重新打开QEI功能（QEIEN由0变1）或每次写QPMOD为新设置值后，硬件会在此后出现的第一次INDEX复位事件时，记录对应的QEA或QEB的变化沿状态。然后在后续每次INDEX由0变1出现时，按照下表寻找第一个满足条件的复位事件，并触发复位动作。设定QEA和QEB有四种组合状态：S0为00，S1为10，S2为11，S3为01。

重启后第一次复位事件	重启后第n次复位事件(n≠1)	
Sn->Sm	与重启后第一次复位事件同向	Sn->Sm
	与重启后第一次复位事件反向	Sm->Sn

注1：n和m的范围都为为1~4。

注2：假设0-1=3，3+1=0，则m和n的关系为，正转m=n+1，反转m=n-1。

注3：INDEX上升沿开始检测四种状态变化，检测到一个变化后INDEX下降沿结束检测。

以QPMOD=10，QPOSMAX=99为例，若重新打开QEI功能后第一次复位动作发生S3->S0，则各关键信号的时序关系如下图所示（灰色框表示复位事件发生位置）：

图21-6 索引脉冲 INDEX 复位方式的工作时序(x4 计数模式)

当QPMOD=01/11时，位置计数器使用自动循环复位方式。该复位方式下，当QPOSCNT在增计数过程中等于QPOSMAX设置值后，在下一个QCLK时QPOSCNT自动复位为0，然后再继续对QCLK进行增计数；当QPOSCNT在减计数过程中等于0后，在下一个QCLK时QPOSCNT重载QPOSMAX设置值，然后继续减计数。自动循环复位方式不受INDEX信号控制，自动使位置计数器QPOSCNT在0~QPOSMAX范围作循环计数。当QPOSCNT上溢或下溢复位时，硬件自动置位QEIIF标志。

以QPMOD=11，QPOSMAX=99为例，各关键信号的时序关系如下图所示：

图21-7　　自动循环复位方式的工作时序(x4 计数模式)

21.3.4 测速方案

测速有两种方案，低速和高速。在低转速应用中，通过固定QPOSCNT的值去检测对应时间的增量。在高速应用中，则用变化的QPOSCNT值来检测对应时间的增量，提高了测量精度。

21.3.4.1 低转速测速方案

在低转速测速应用方案中，QEI模块使用QTimer作为时间增量的计数器。

注：QTimer正常工作过程中修改QTPSQ为新设置值，会立即改变计数时钟的频率。建议用户在QTimer不计数状态下修改QTPSQ设置值，以保证计数时钟频率的稳定。

当QEI功能打开时，正交解码器输出的每个QCLK可以触发捕捉事件，捕捉事件的发生频率由QTEPS设置。当有效的捕捉事件发生时，硬件自动把QCNT当前计数值锁存到QTMLAT寄存器，同时置位QTCAPIF中断标志（QEIEN=1启动QTimer计数后的第一次捕捉事件不会置位该标志），然后清除QTimer计数器为0，并重新开始增计数。用户在检测到QTCAPIF标志后，可以根据触发事件频率设置（固定的QPOSCNT变化值）和QTMLAT锁存值计算出转速，QTCAPIF标志由用户软件写0清除。

当QTimer计数到QTPR后，硬件自动复位计数器为0并重新开始增计数，同时置位QTIF标志。用户需要准确设置QTEPS，以防止捕捉事件发生前QTimer溢出，通过查询QTIF标志可以判断捕捉发生前QTimer是否已经溢出。当QEIEN=1，QTimer启动状态下修改QTEPS为新设置值，新设置值将在下一次捕捉事件发生后才有效。

低转速测速应用方案中，各关键信号的时序关系如下图所示（QTCAP信号表示发生了捕捉事件，置起QTCAPIF标志位）：

图21-8 低转速测速应用方案中关键信号示意图(QTEPS=0001)

21.3.4.2 高转速测速方案

当电机转速很快时，编码器在一个QCLK期间捕捉得到的QCNT个数比较小，此时得到的转速精度不能满足要求。通过设定QCNTMIN值，在捕捉信号发生的时候，如果QCNT的值小于QCNTMIN，忽略此次捕捉信号，同时QCNT继续增加，但是捕捉次数记录下来。直到在某次捕捉信号发生时QCNT大于等于QCNTMIN，才把把QTCAPIF置起，然后把记录下来的捕捉分频次数放到QTMLAT的高16位中，把QCNT放到QTMLAT的低16位中，并且是一个CLOCK存入32位数据。

如下图示例，设置QCNTMIN为0x1200，QTCAP在前6次分频时，QCNT的值都小于QCNTMIN，在第七次分频时QCNT为0x1234，大于QCNTMIN的0x1200，所以此时触发捕捉溢出，同时把7放到QTMLAT的高16位，把0x1234放到QTMLAT的低16位。

图21-9 高速测速应用方案中关键信号示意图(QTEPS=0001)

21.3.5 QEI 模块的中断

21.3.5.1 QEI 功能的中断

QEI模块使用QEIIE和QEIIF控制和指示QEI功能的中断，该中断只在QEI功能打开（QEIEN=1）时有效。

QPMOD=00/10时，位置计数器使用索引脉冲INDEX复位方式。当发生INDEX复位事件时，硬件自动置位QEIIF标志，如果此时QEIIE=1，则触发相应的中断。中断标志位QEIIF由用户软件写0清除。

QPMOD=01/11时，位置计数器使用自动循环复位方式。该复位方式下，位置计数器QPOSCNT自动在0~QPOSMAX范围作循环计数。当计数上溢或下溢时，硬件自动置位QEIIF标志，如果此时QEIIE=1，则触发相应的中断。中断标志位QEIIF由用户软件对QEIIFC写1清除。

21.3.5.2 计数错误中断

当QEI功能打开（QEIEN=1），且QPMOD=00/10使用索引脉冲INDEX复位方式时，硬件使用QCEIE和QCEIF控制和指示计数错误中断。该中断只在发生第一次复位事件后才有效。

当发生第一次复位事件后，QPOSCNT的计数值范围将为0~QPOSMAX。当此后增计数过程计数值从QPOSMAX变为QPOSMAX+1，或减计数过程计数值从0变为0xFFFF的同时，硬件自动置位QCEIF标志，表示发生计数错误。如果此时QCEIE=1，则触发相应的中断。中断标志位QCEIF由用户软件对QCEIFC写1清除。

21.3.5.3 相位错误中断

当QEI功能打开（QEIEN=1），正交解码器会自动检查[QEA:QEB]的状态转换过程，当检测到QEA与QEB同相位或反相位的同时，解码器会判断出现相位检测错误，置位中断标志QPEIF，如果此时QPEIE=1，则触发相应的中断。中断标志位QPEIF由用户软件对QPEIFC写1清除。

21.3.5.4 QEI(QTimer)捕捉事件中断

当QEI功能打开（QEIEN=1），QTimer作为QEI功能的辅助定时器使用，硬件使用QTCAPIE和QTCAPIF控制和指示捕捉事件中断。

当有效的捕捉事件发生时，硬件自动把QCNT当前计数值锁存到QTMLAT寄存器，同时置位QTCAPIF中断标志（QEIEN=1启动QTimer计数后的第一次捕捉事件不会置位该标志），如果此时QTCAPIE=1，则触发相应的中断。中断标志位QTCAPIF由用户软件对QTCAPIFC写1清除。

21.3.5.5 QTimer 溢出中断

QTimer作为一个独立的32位定时器，计数到QTPR设置值后发生溢出，计数器自动复位为0并重新开始增计数，同时置位QTIF标志。如果此时QTIE =1，则触发相应的中断。中断标志位QTIF由用户软件对QTIFC写1清除。

21.4 寄存器

QEI 模块寄存器列表 (基地址：0x4000 1400)

地址	寄存器名	说明
0x4000 1400	QEICON	QEI控制寄存器
0x4000 1404	QFLTCON	QEI输入引脚滤波控制寄存器
0x4000 1408	QPOSCNT	QEI位置计数寄存器
0x4000 140C	QPOSMAX	QEI最大计数值寄存器
0x4000 1410	QTMLAT	QEI计数值锁存寄存器
0x4000 1414	QCNT	QEI计数寄存器
0x4000 1418	QCNTMIN	QEI计数器在捕捉时最小计数值
0x4000 141C	QTPR	QEI计数器周期寄存器
0x4000 1420	QTPSQ	QEI计数器时钟预分频值
0x4000 1424	QEIINT	QEI中断和状态寄存器
0x4000 1428	QTINTF	QEI中断标志和清除寄存器

21.4.1 QEI 控制寄存器 (QEI_QEICON)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留				QTEPS[3:0]				QPDIR	QIDXS	QSWAP	QIDXEN	QEIEN	QTSR	QPMOD[1:0]	
-				*RW*				*RO*	*RO*	*RW*	*RW*	*RW*	*RW*	*RW*	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 12	保留	-
11 ~ 8	**QTEPS[3:0]**	**QEI(QTimer)捕捉事件分频设置位（仅在QEI功能打开时有效）：** 0000：QCLK/1 0001：QCLK/2 0010：QCLK/4 0011：QCLK/8 0100：QCLK/16 0101：QCLK/32 0110：QCLK/64 0111：QCLK/128 1000：QCLK/256 1001：QCLK/512 1010：QCLK/1024 1011：QCLK/2048 Others：QCLK/2048
7	**QPDIR**	**编码器旋转方向状态位：** 0：QEA信号相位落后QEB，编码器反向旋转，位置计数器减计数 1：QEA信号相位超前QEB，编码器正向旋转，位置计数器增计数
6	**QIDXS**	**INDEX输入信号（QEI正交解码器输入位置）电平状态指示位：** 0：INDEX输入低电平 1：INDEX输入高电平
5	**QSWAP**	**QEA和QEB输入交换选择位：** 0：QEA相与QEB相输入未交换，直接提供给正交解码器 1：QEA相与QEB相输入作交换后再提供给正交解码器
4	**QIDXEN**	**位置计数器复位使能位：（仅在QPMOD=00/10时有效）** 0：索引脉冲INDEX不能复位位置计数器 1：索引脉冲INDEX可以复位位置计数器
3	**QEIEN**	**QEI功能使能控制位：** 0：QEI功能关闭 1：QEI功能打开，具体工作模式受QPMOD控制
2	**QTSR**	**QEI定时器的启动位：** 0：不启动Qtimer 1：启动Qtimer 注：此位为当QEIEN为0时，Qtimer是否工作的开启位，当QEIEN为1时，此位控制无效。
1 ~ 0	**QPMOD[1:0]**	**QEI位置计数器计数选择位：（仅在QEI功能打开时有效）** 00：x2计数模式，使用索引脉冲INDEX复位位置计数器 01：x2计数模式，位置计数器在0~QPOSMAX间自动循环计数 10：x4计数模式，使用索引脉冲INDEX复位位置计数器 11：x4计数模式，位置计数器在0~QPOSMAX间自动循环计数

21.4.2 QEI 输入引脚滤波控制寄存器 (QEI_QFLTCON)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								QIDXFEN	QIDXCPS[2:0]			QABFEN	QABCPS[2:0]		
-								RW	RW			RW	RW		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7	**QIDXFEN**	**INDEX输入引脚滤波使能位：** 0：禁止INDEX输入引脚的滤波功能 1：使能INDEX输入引脚的滤波功能
6 ~ 4	**QIDXCPS[2:0]**	**INDEX引脚滤波时钟分频系数：** 000：1/1 PCLK 001：1/2 PCLK 010：1/4 PCLK 011：1/16 PCLK 100：1/32 PCLK 101：1/64 PCLK 110：1/128 PCLK 111：1/256 PCLK
3	**QABFEN**	**QEA和QEB输入引脚滤波使能位：** 0：禁止QEA和QEB输入引脚的滤波功能 1：使能QEA和QEB输入引脚的滤波功能
2 ~ 0	**QABCPS[2:0]**	**QEA和QEB引脚滤波时钟分频系数：** 000：1/1 PCLK 001：1/2 PCLK 010：1/4 PCLK 011：1/16 PCLK 100：1/32 PCLK 101：1/64 PCLK 110：1/128 PCLK 111：1/256 PCLK

21.4.3 QEI 位置计数寄存器 (QEI_QPOSCNT)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
QPOSCNT[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**QPOSCNT[15:0]**	**16位增/减双向计数器，计数方向由QPDIR位控制**

21.4.4 QEI 最大计数值寄存器 (QEI_QPOSMAX)

偏移地址: 0x000C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
QPOSMAX[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**QPOSMAX[15:0]**	位置计数器的最大计数值寄存器

21.4.5 QEI 计数值锁存寄存器 (QEI_QTMLAT)

偏移地址: 0x0010

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
QTMLAT [31:0]															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
QTMLAT [31:0]															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**QTMLAT [31:0]**	满足条件时，锁存QCNT计数值

21.4.6 QEI 计数寄存器 (QEI_QCNT)

偏移地址: 0x0014

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
QCNT [31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
QCNT [31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**QCNT [31:0]**	**32bit计数器** 注：在QEIEN为1时不能修改

21.4.7 QEI 计数器在捕捉时最小计数值 (QEI_QCNTMIN)

偏移地址: 0x0018

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
QCNTMIN[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**QCNTMIN[15:0]**	当捕捉事件发生时，如果QCNT的值小于QCNTMIN寄存器的值，则此次捕捉事件不发生，同时内部的捕捉分频加1，直到捕捉发生到QCNT的值大于QCNTMIN，则把分频和此时QCNT自动锁存到QTMLAT中。

21.4.8 QEI 计数器周期寄存器 (QEI_QTPR)

偏移地址: 0x001C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
QTPR [31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
QTPR [31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**QTPR [31:0]**	**32bit周期寄存器**

21.4.9 QEI 计数器时钟预分频值 (QEI_QTPSQ)

偏移地址: 0x0020

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
QTPSQ[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	**-**
15 ~ 0	**QTPSQ[15:0]**	定时的分频值

21.4.10 QEI 中断和状态寄存器 (QEI_QEIINT)

偏移地址: 0x0024

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留											QTIE	QTCAPIE	QEIIE	QCEIE	QPEIE
-											*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 5	保留	-
4	**QTIE**	**Qtimer溢出中断使能位：** 0：禁止Qtimer溢出事件发生时触发中断 1：允许Qtimer溢出事件发生时触发中断
3	**QTCAPIE**	**QTimer捕捉事件中断使能位（仅在QEI功能打开时有效）：** 0：禁止捕捉事件发生时触发中断 1：允许捕捉事件发生时触发中断
2	**QEIIE**	**QEI功能中断使能位：** 0：禁止QEI中断 1：使能QEI中断
1	**QCEIE**	**INDEX信号复位位置计数器模式下，错误检测中断使能位：** 0：禁止错误检测中断 1：使能错误检测中断
0	**QPEIE**	**编码器输入相位错误中断使能位，QEA/QEB信号同相或反相时视为相位错误：** 0：禁止相位错误中断 1：使能相位错误中断

21.4.11 QEI 中断标志和清除寄存器 (QEI_QTINTF)

偏移地址: 0x0028

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留											QTIFC	QTCAPIFC	QEIIFC	QCEIFC	QPEIFC
-											*WO*	*WO*	*WO*	*WO*	*WO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留											QTIF	QTCAPIF	QEIIF	QCEIF	QPEIF
-											*RO*	*RO*	*RO*	*RO*	*RO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 21	保留	-
20	**QTIFC**	**Qtimer溢出标志清除位** 0：无效 1：清除
19	**QTCAPIFC**	**QEI(QTimer)捕捉事件中断标志清除位** 0：无效 1：清除
18	**QEIIFC**	**QEI功能中断标志清除位** 0：无效 1：清除
17	**QCEIFC**	**QEI位置计数器计数错误标志清除位** 0：无效 1：清除
16	**QPEIFC**	**QEI解码器相位错误标志清除位** 0：无效 1：清除
15 ~ 5	保留	-
4	**QTIF**	**QTimer溢出标志位：** 0：QCNT没有发生计到QTPR值溢出 1：QCNT发生计到QTPR值溢出
3	**QTCAPIF**	**QEI(QTimer)捕捉事件中断标志位（仅在QEI功能打开时有效）：** 0：无捕捉事件发生 1：发生捕捉事件，硬件已经锁存Qtimer.CNT值到QTMLAT
2	**QEIIF**	**QEI功能中断标志位，QPMOD=00/10时作为INDEX复位事件标志：** 0：未发生INDEX复位事件中断 1：已发生INDEX复位事件中断 QEI功能中断标志位，QPMOD=01/11时作为位置计数上溢或下溢标志： 0：未发生位置计数器上溢或下溢中断 1：已发生位置计数器上溢或下溢中断
1	**QCEIF**	**QEI位置计数器计数错误标志位：** 0：QEI位置计数器未发生计数错位中断 1：QEI位置计数器已发生计数错位中断
0	**QPEIF**	**QEI解码器相位错误标志位：** 0：未发生输入信号相位检测错误中断 1：已发生输入信号相位检测错误中断

22. 片上模拟功能模块（Analog Module On Chip，AMOC）

22.1 简介

SH33F2801片上集成了4个高速运算放大器和2个模拟比较器，其中1个比较器与运算放大器共用。运算放大器通过外部电阻选择放大倍数，放大信号可以直接被ADC采样。比较器都为多输入比较器，同时比较器正端输入与OP的正端输入可以复用，两个功能可以同时实现。

22.2 主要特性

- 集成4个通用运算放大器，放大器的输入输出端都开放，可通过外接电阻调整运放增益
- 放大器输出可直接作为ADC转换模块的输入或者做比较器的同相输入端
- 放大器0可以选择内部增益模式，共有8种增益倍数，最大32倍增益
- 两个比较器都为多输入比较器，正端都有3个外部引脚，也可以选择OP输出
- 比较器输出结果滤波后可以用作MCM和PCA故障检测输入信号
- 比较器反相输入端参考信号可以从外部输入，也可以采用内部参考电压（8档）作为反向输入
- 比较器内置施密特窗口和数字滤波电路，输出可触发上升沿、下降沿和双沿中断

22.3 运算放大器

22.3.1 运放典型应用图

图22-1　　运放典型应用图（外部增益）

图22-2　　运放典型应用图（OP0P0 内部增益）

22.4 比较器

22.4.1 比较器用法

内置2个比较器，比较器0和比较器1。2个比较器的输出都可以作为MCM模块和PCA模块的故障输入检测信号。故障输入检测可以自动迅速关闭PWM输出，从而实现快速保护，也可以用在某些电压/电流闭环中。2个比较器都有集成滤波常数可编程的数字滤波电路，同时提供中断功能，比较适合用作需要精确比较的应用场合，比如直流无刷电机的反电动势过零点检测。比较器1包含内置可选的施密特功能。

比较器0与比较器1的区别在于实现方式不一样，比较器0是通过OP0实现的。

注：

（1）比较器的上电默认状态为关闭。

（2）比较器1的功能和放大器1的功能可以同时实现。

（3）比较器0不能和OP0同时使用。

比较器滤波方法说明：用内部时钟采样滤波器输入，如果采样结果为高电平，计数器加1，直到计数器结果超出设定的常数则滤波器输出1，同时计数器置为滤波常数；如果采样结果为低电平，计数器减1，直到计数器结果为0则滤波器输出0，同时计数器置为0。

注：

（1）计数器上限设置为滤波常数，下限为0，超出上下限，不执行加减动作

（2）滤波计数器初值设定:若比较器中断设为上升沿触发，计数器初始值为0；若设为下降沿触发，则滤波计数器初值设为滤波常数。若设为双沿触发或不触发，则滤波计数器初值设为滤波常数。

（3）滤波器输出稳定需要一个建立时间，大致为滤波常数设置的时间，如滤波常数设置为256，则在256个系统时钟后，滤波器输出稳定，在这之前，滤波器输出不确定，中断标志CxIF可能会被误置1，因此，刚打开滤波器时，需要在滤波器输出稳定后软件清除此中断标志位。

22.4.2 比较器框图

比较器的输出波形的产生流程具体可参考下图：

图22-3　　多通道输入模拟比较器 0

图22-4　　多通道输入模拟比较器 1

比较器1可以通过设置寄存器CxSMT[1:0](x=0/1)设置施密特比较器窗口（迟滞比较器），如下图所示：

图22-5　　比较器施密特窗口

注：图22-3和图22-4中的VREF指的是外部VREF引脚（PB14）。

22.5　寄存器

AMOC 模块寄存器列表 (基地址：0x4000 3800)

地址	寄存器名	说明
0x4000 3800	CMP0CON	比较器 0 控制寄存器
0x4000 3804	CMP1CON	比较器 1 控制寄存器
0x4000 3808	CMPINTF	比较器中断标志和清除寄存器
0x4000 380C	OPCON	放大器控制寄存器 1

22.5.1 比较器 0 控制寄存器 (AMOC_CMP0CON)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留	C0OUT2	C0OUT1	C0OUT0	CMPAUTO	TRGPOL	PWMTRGS[1:0]		CMPGAP[1:0]		CMP0VRS[5:0]					
-	RO	RO	RO	RW	RW	RW		RW		RW					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CMP0EN	CMP0VCMP	CP0NOUTEN	C0OUTEN	C0IES[1:0]		保留		C0NCHS	C0PCHS[2:0]			C0OUT	C0DEB[2:0]		
RW	RW	RW	RW	RW		-		RW	RW			RO	RW		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31	保留	-
30	**C0OUT2**	**比较器 0 使能自动轮流比较功能后， CP0P2(OP0P2)的比较结果**
29	**C0OUT1**	**比较器 0 使能自动轮流比较功能后， CP0P1(OP0P1)的比较结果**
28	**C0OUT0**	**比较器 0 使能自动轮流比较功能后， CP0P0(OP0P0)的比较结果。**
27	**CMPAUTO**	**比较器 0 自动轮流比较功能使能位** 0： 关闭自动轮流比较 1： 打开自动轮流比较 注：该位为 0 时，比较器 0 的输入由 C0PCHS[2:0]位段决定；该位为 1 时，OP0P0~OP0P2 均为比较器 0 的输入
26	**TRGPOL**	**比较器 0 比较功能控制位** 0：仅在 PWM 输出有效期间比较 1：仅在 PWM 输出无效期间比较
25 ~ 24	**PWMTRGS[1:0]**	**比较器 0 同步比较控制位** 00：连续比较模式，无同步比较功能 01：和 PWM0 输出同步 10：和 PWM1 输出同步 11：和 PWM2 输出同步
23 ~ 22	**CMPGAP[1:0]**	**自动轮流比较间隔:** 00：比较间隔为 32 个 APB 时钟 01：比较间隔为 48 个 APB 时钟 10：比较间隔为 64 个 APB 时钟 11：比较间隔为 80 个 APB 时钟
21 ~ 16	**CMP0VRS[5:0]**	**比较器 0 反相端参考电压选择位** xxx：(1+CMP0VRS[5:0])/ 64 *Vcmp 注：负端电压范围为 1/64~64/64
15	**CMP0EN**	**比较器 0 使能控制位** 0：比较器 0 关闭 1：比较器 0 开启
14	**CMP0VCMP**	**比较器 0 反相端参考电压源 Vcmp 选择位：** 0：VDD 1：外接 Vref（PB14）
13	**CP0NOUTEN**	**当比较器 N 端为内部基准源时，比较器 N 端是否输出到 CP0N** 0：不输出 1：输出 注意：只有在 C0NCHS 为 1 时和 OP0EN 为 0 时，此位才有效

12	**C0OUTEN**	**比较器 0I/O 输出口使能** 0：不通过比较器 0 的 CP0OUT 引脚输出，CP0OUT 为普通 IO 1：通过比较器 0 的 CP0OUT 引脚输出 注意：I/O 输出是带滤波功能的，滤波参数为 C0DEB[2:0]的设定值
11 ~ 10	**C0IES[1:0]**	**比较器 0 中断模式选择位** 00：不触发中断标志 01：下降沿触发，比较器 0 输出由高变低时触发中断标志 10：上升沿触发，比较器 0 输出由低变高时触发中断标志 11：双沿触发，比较器 1 输出由高变低时和由低变高时都触发中断标志
9 ~ 8	保留	-
7	**C0NCHS**	**比较器 0 反相输入端选择位** 0：选择 C0N 作为比较器 0 反相输入端 1：选择内部参考 CMP0VRS 作为比较器 0 反相输入端
6 ~ 4	**C0PCHS[2:0]**	**比较器 0 同相输入端选择位** 000：选择 OP0P0 作为比较器 0 同相输入端 001：选择 OP0P1 作为比较器 0 的同相输入端 010：选择 OP0P2 作为比较器 0 的同相输入端 011：保留 100：选择 OP1OUT 作为比较器 0 同相输入端 101：选择 OP2OUT 作为比较器 0 同相输入端 110：选择 OP3OUT 作为比较器 0 的同相输入端 111：保留
3	**C0OUT**	**比较器 0 输出状态标志位（表示经过滤波器以后的状态，只读）** 0：比较器 0 输出低 1：比较器 0 输出高
2 ~ 0	**C0DEB[2:0]**	**比较器 0 输出信号滤波时间** 000：无滤波 001：滤波时间常数为 0.5us 010：滤波时间常数为 1us 011：滤波时间常数为 2us 100：滤波时间常数为 4us 101：滤波时间常数为 8us 110：滤波时间常数为 12us 111：滤波时间常数为 16us 注 1：上述滤波常数时间不是精确值，仅供参考。 注 2：滤波器算法说明：用 HSI 时钟采样滤波器输入。采用数字积分，如果采样结果为高电平，计数器加 1，计数器结果超出设定的常数，则滤波器输出 1 同时计数器置为滤波常数；如果采样结果为低电平，计数器减 1，计数器结果小于滤波器常数,则滤波器输出 0 同时计数器置为 0。

22.5.2 比较器 1 控制寄存器 (AMOC_CMP1CON)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留										CMP1VRS[5:0]					
-										RW					
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CMP1EN	CMP1VCMP	CP1NOUTEN	C1OUTEN	C1IES[1:0]		C1SMT[1:0]		C1NCHS	C1PCHS[2:0]			C1OUT	C1DEB[2:0]		
RW	RW	RW	RW	RW		RW		RW	RW			RO	RW		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 22	保留	-
21 ~ 16	**CMP1VRS[5:0]**	**比较器 1 反相端参考电压选择位** xxx：(1+CMP1VRS[5:0])/ 64 *Vcmp 注：负端电压范围为 1/64~64/64
15	**CMP1EN**	**比较器 1 使能控制位** 0：比较器 1 关闭 1：比较器 1 开启
14	**CMP1VCMP**	**比较器 1 反相端参考电压源 Vcmp 选择位：** 0：VDD 1：外接 Vref（PB14）
13	**CP1NOUTEN**	**当比较器 N 端为内部基准源时，比较器 N 端是否输出到 CP1N** 0：不输出 1：输出 注意：只有在 C1NCHS 为 1 时，此位才有效
12	**C1OUTEN**	**比较器 1I/O 输出口使能** 0：不通过比较器 1 的 CP1OUT 引脚输出，CP1OUT 为普通 IO 1：通过比较器 1 的 CP1OUT 引脚输出 注意：I/O 输出是带滤波功能的，滤波参数为 C1DEB[2:0]的设定值
11 ~ 10	**C1IES[1:0]**	**比较器 1 中断模式选择位** 00：不触发中断标志 01：下降沿触发，比较器 1 输出由高变低时触发中断标志 10：上升沿触发，比较器 1 输出由低变高时触发中断标志 11：双沿触发，比较器 1 输出由高变低时和由低变高时都触发中断标志
9 ~ 8	**C1SMT[1:0]**	**比较器 1 施密特电压选择位：** 00: 没有施密特窗口特性 01: 施密特窗口为 10mv 10: 施密特窗口为 20mv 11: 施密特窗口为 50mv
7	**C1NCHS**	**比较器 1 反相输入端选择位** 0：选择 C1N 作为比较器 1 反相输入端 1：选择内部参考 CM1VRS 作为比较器 1 反相输入端，CP1N 为普通 IO 注：如果用到内部 VREF 需提前 100us 将基准源打开，即将寄存器[AGCON: VREFEN]位置 1

6 ~ 4	**C1PCHS[2:0]**	**比较器 1 同相输入端选择位** 000：选择 C1P0 作为比较器 1 同相输入端 001：选择 C1P1 作为比较器 1 同相输入端 010：选择 C1P2 作为比较器 1 同相输入端 011：选择 OP0OUT 作为比较器 1 的同相输入端 100：选择 OP1OUT 作为比较器 1 的同相输入端 101：选择 OP2OUT 作为比较器 1 的同相输入端 110：选择 OP3OUT 作为比较器 1 的同相输入端 111：保留
3	**C1OUT**	**比较器 1 输出状态标志位（表示经过滤波器以后的状态，只读）** 0：比较器 1 输出低 1：比较器 1 输出高
2 ~ 0	**C1DEB[2:0]**	**比较器 1 输出信号滤波时间** 000：无滤波 001：滤波时间常数为 0.5us 010：滤波时间常数为 1us 011：滤波时间常数为 2us 100：滤波时间常数为 4us 101：滤波时间常数为 8us 110：滤波时间常数为 12us 111：滤波时间常数为 16us 注 1：上述滤波常数时间不是精确值，仅供参考。 注 2：滤波器算法说明：用 HSI 时钟采样滤波器输入。采用数字积分，如果采样结果为高电平，计数器加 1，计数器结果超出设定的常数，则滤波器输出 1 同时计数器置为滤波常数；如果采样结果为低电平，计数器减 1，计数器结果小于滤波器常数,则滤波器输出 0 同时计数器置为 0。

22.5.3 比较器中断标志和清除寄存器 (AMOC_CMPINTF)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留														C1IFC	C0IFC
-														*WO*	*WO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留														C1IF	C0IF
-														*RO*	*RO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 18	保留	-
17	**C1IFC**	**比较器 1 输出上升沿中断标志位清除位** 0：无效 1：清除
16	**C0IFC**	**比较器 0 输出上升沿中断标志位清除位** 0：无效 1：清除
15 ~ 2	保留	-

1	**C1IF**	**比较器 1 输出上升沿中断标志位（经过滤波器以后）** 0：比较器 1 输出没有产生中断 1：比较器 1 输出产生中断请求
0	**C0IF**	**比较器 0 输出上升沿中断标志位（经过滤波器以后）** 0：比较器 0 输出没有产生中断 1：比较器 0 输出发生中断请求

22.5.4 放大器控制寄存器 1 (AMOC_OPCON)

偏移地址: 0x000C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								OP0 GEN	OP0GSEL[2:0]			OP3 EN	OP2 EN	OP1 EN	OP0 EN
-								*RW*	*RW*			*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7	**OP0GEN**	**运算放大器 0 的 OP0P0 引脚的内部增益使能位：** 0：禁止 1：使能
6 ~ 4	**OP0GSEL[2:0]**	**运算放大器 0 增益系数选择：** 000：由内部增益系数确定 X1 001：由内部增益系数确定 X4 010：由内部增益系数确定 X6 011：由内部增益系数确定 X8 100：由内部增益系数确定 X12 101：由内部增益系数确定 X16 110：由内部增益系数确定 X24 111：由内部增益系数确定 X32
3	**OP3EN**	**运算放大器 3 使能控制位** 0：运算放大器 3 关闭 1：运算放大器 3 开启 注：IO 引脚设置成模拟输入，对应 I/O AF 寄存器需配置为模拟 AF。
2	**OP2EN**	**运算放大器 2 使能控制位** 0：运算放大器 2 关闭 1：运算放大器 2 开启 注：IO 引脚设置成模拟输入，对应 I/O AF 寄存器需配置为模拟 AF。
1	**OP1EN**	**运算放大器 1 使能控制位** 0：运算放大器 1 关闭 1：运算放大器 1 开启 注：IO 引脚设置成模拟输入，对应 I/O AF 寄存器需配置为模拟 AF。
0	**OP0EN**	**运算放大器 0 使能控制位** 0：运算放大器 0 关闭 1：运算放大器 0 开启 注：IO 引脚设置成模拟输入，对应 I/O AF 寄存器需配置为模拟 AF。 注 2：如果 OP0EN 开启则 OP0 必须当做 OP 用，只有当 OP0EN 为 0 时，OP0 才能当做比较器用。

23. 串行通信接口

23.1 通用异步收发器（UART）

23.1.1 简介

通用异步收发传输器（Universal Asynchronous Receiver/ Transmitter)，通常称作 UART，是一种异步收发传输器。

23.1.2 主要特性

- 自带波特率发生器
- 1/2 位停止位可设
- UART 有四种工作方式
- 增强功能包括帧出错检测及自动地址识别
- 奇偶校验
- 波特率检测
- 支持 LIN 总线同步间断的硬件产生和检测
- 每个 UART 都可配置 FIFO，支持基于 FIFO 的数据收发

23.1.3 功能描述

23.1.3.1 UART 工作方式

UART 有 4 种工作方式。在通信之前用户必须先初始化 UARTx_CR，选择方式和波特率以及停止位。

在所有四种方式中，在使能了 TEN 后，任何将 UARTx_TDR 作为目标寄存器的写操作都会启动发送。在方式 0 中由条件 RI = 0 和 REN = 1 初始化接收。这会在 TXD 引脚上产生一个时钟信号，然后在 RXD 引脚上移 8 位数据。在其它方式中由输入的起始位初始化接收（如果 RI = 0 和 REN = 1）。外部发送器通信以发送起始位开始。

表23-1 UART 工作方式列表

SM1	SM0	方式	类型	波特率	帧长度	起始位	停止位	第 9 位
0	0	0	同步半双工	PCLK/（48 或 12）	8 位	无	无	无
0	1	1	异步全双工	自带波特率发生器的溢出率/16	10 位(或 11)	1	1(或 2)	无
1	0	2	异步全双工	PCLK/（256 或 128）	11 位(或 12)	1	1(或 2)	0，1
1	1	3	异步全双工	自带波特率发生器的溢出率/16	11 位(或 12)	1	1(或 2)	0，1

方式 0：同步，半双工通讯

方式 0 也称为 UART 的 SPI 模式，支持与外部设备的同步通信。在 TXD 引脚发送移位时钟，在 RXD 引脚上收发串行数据，如果与外部 SPI 设备连接，则 UART 方式 0 作为 SPI 主机。因为只有 RXD 引脚支持数据传输，这种方式只能进行半双工通讯。在发送时从 RXD 引脚送出数据，在接收时从 RXD 引脚接收数据。

在这个方式中，每帧收发 8 位，低位先接收或发送。

通过置 SMOD0 位@UARTx_CR 为 0 或 1，波特率固定为 PCLK 时钟的 1/48 或 1/12。当 SMOD0 位等于 0 时，串行端口以 PCLK 时钟的 1/48 运行。当 SMOD0 位等于 1 时，串行端口以 PCLK 时钟的 1/12 运行。

功能块框图如下图所示。数据通过 RXD 引脚移入和移出串行端口，移位时钟由 TXD 引脚输出。

图23-1　　方式 0 功能块框图

在使能了 TEN@UARTx_CR 后，只要 UARTx_TDR 缓冲寄存器为空，就可以执行写操作。写入 TDR 后的下一个 PCLK 时钟 TX 模块开始发送。在移位时钟的下降沿加载发送数据，移位寄存器的内容逐次从左往右送出到 RXD 线上，当移位寄存器中的所有 8 位都发送后，TX 模块停止发送，然后在下一个 PCLK 时钟的上升沿将数据从 UARTx_TDR（如果不为空）搬移至 SHIFT 寄存器。

由于有 TDR 和 SHIFT 两个寄存器，首次发送时可以连续写入 2 个 byte 数据，在 SHIFT 把第 1 个 byte 发送出去前，TI @UARTx_FR 标志一直保持为 0，即表示 TDR 不为空。

Send Timing of Mode 0

图23-2　　方式 0 发送时序图

REN@UARTx_CR 置 1 和 RI@UARTx_FR 清零（通过读 RDR 或置位 RIC 都可以清零 RI）初始化接收。下一个 PCLK 时钟启动 RX 模块，在移位时钟的上升沿锁存数据，RXD 线上的数据逐次从右往左进入移位寄存器。当所有 8 位数据都进入移位寄存器后，RX 模块停止接收，在下一个 PCLK 时钟的上升沿 RI 置位，直到被软件读取 RDR，RI 随之被清零（或写 RIC 清零 RI）。

注意，RI 未清零时，不会继续送出移位时钟，也就不会继续接收数据，因此 UART 方式 0 不会存在接收溢出的情况。

Receive Timing of Mode 0

图23-3　　方式 0 接收时序图

方式 1：8 位 UART，可变波特率，异步全双工

方式 1 提供 10/11 位全双工异步通信，10 位由一个起始位（逻辑 0），8 个数据位（低位为第一位）和 1 位/2 位停止位（逻辑 1）组成。在接收时，这 8 个数据位存储在 UARTx_RDR 中而停止位储存在 RB8@UARTx_RDR 中。方式 1 中的波特率固定为自带波特率发生器溢出率的 1/16。功能块框图如下图所示：

图23-4　　方式 1 功能块框图

在使能了 TEN@UARTx_CR 后, UARTx_TDR 寄存器为空，写操作可以被执行，实际上 TXD 引脚上发送是从 16 分频计数器中的下一次跳变之后的 PCLK 时钟开始的，因此位时间与 16 分频计数器是同步的。起始位首先在 TXD 引脚上移出，然后是 8 位数据位。在发送移位寄存器中的所有 8 位数据都发送完后，停止位在 TXD 引脚上移出，在停止位发送结束的同时 TI 标志置位。

Send Timing of Mode 1

图23-5　　方式 1 发送时序图(STOP=1BIT)

只有 REN 置位时才允许接收。当 RXD 引脚检测到下降沿时串行口开始接收串行数据。为此，UART 对 RXD 不断采样，采样速率为波特率的 16 倍。当检测下降沿时，16 分频计数器立即复位，这有助于 16 分频计数器与 RXD 引脚上的串行数据位同步。16 分频计数器把每一位的时间分为 16 个状态，在第 7、8、9 状态时，位检测器对 RXD 端的电平进行采样。为抑制噪声，在这 3 个状态采样中至少有 2 次采样值一致数据才被接收。如果所接收的第一位不是 0，说明这位不是一帧数据的起始位，该位被忽略，接收电路被复位，等待 RXD 引脚上另一个下降沿的到来。若起始位有效，则移入移位寄存器，并接着移入其它位到移位寄存器。8 个数据位和 1 个停止位移入之后，移位寄存器的内容被分别装入 RDR 和 RB8 中，RI 置位，但必须满足下列条件：

1. RI = 0
2. 无第 9 位功能的模式或者接收的停止位 = 1

如果这些条件被满足，那么停止位装入 RB8，8 个数据位装入 UARTx_RDR，RI 被置位。否则接收的帧会丢失，继续检测下一个起始位。这时，接收器将重新去探测 RXD 端是否另一个下降沿。用户应将 UARTx_RDR 内数据取走（或用软件写 1 清零 RI），然后才能再次接收，否则接收到 SHIFT 内的数据将会覆盖 UARTx_RDR 内未被取走的数据，且会产生 RXOV 错误标志。

Receive Timing of Mode 1

图23-6　　方式 1 接收时序图(STOP=1BIT)

方式 2：9 位 UART，固定波特率，异步全双工

这个方式使用异步全双工通信中的 11 位/12 位。一帧由一个起始位（逻辑 0），8 个数据位（低位为第一位），一个可编程的第 9 数据位和 1 位/2 位停止位（逻辑 1）组成。方式 2 支持多机通信和硬件地址识别（详见多机通讯章节）。在数据传送时，第 9 数据位（TB8@UARTx_TDR）可以写 0 或 1，该位的设置需要与 TDR@UARTx_TDR 一同写入。第 9 位可用作多机通信中的数据/地址标志位，以及奇偶校验位。当接收到数据时，第 9 数据位移入 RB8 而停止位不保存。通过 SMOD2@UARTx_CR 位选择波特率为 PCLK 的 1/256 或 1/128。功能块框图如下所示：

图23-7　　方式 2 功能块框图

在使能了 TEN@UARTx_CR 后, UARTx_TDR 寄存器为空，写操作可以被执行，同时也将 TB8 载入到发送移位寄存器的第 9 位中。实际上 TXD 发送是从 16 分频计数器中的下一次跳变之后的 PCLK 时钟开始的，因此位时间与 16 分频计数器是同步的。起始位首先在 TXD 引脚上移出，然后是 9 位数据。在发送转换寄存器中的所有 9 位数据都发送完后，停止位在 TXD 引脚上移出，在停止位发送结束时 TI 标志置位。

Send Timing of Mode 2

图23-8　　方式 2 发送时序图(STOP=1BIT)

只有 REN 置位时才允许接收。当 RXD 引脚检测到下降沿时串行口开始接收串行数据。为此，CPU 对 RXD 不断采样，采样速率为波特率的 16 倍。当检测下降沿时，16 分频计数器立即复位。这有助于 16 分频计数器与 RXD 引脚上的串行数据位同步。16 分频计数器把每一位的时间分为 16 个状态，在第 7、8、9 状态时，位检测器对 RXD 端的电平进行采样。为抑制噪声，在这 3 个状态采样中至少有 2 次采样值一致数据才被接收。如果所接收的第一位不是 0，说明这位不是一帧数据的起始位，该位被忽略，接收电路被复位，等待 RXD 引脚上另一个下降沿的到来。若起始位有效，则移入移位寄存器，并接着移入其它位到移位寄存器。9 个数据位和 1 个停止位移入之后，移位寄存器的内容被分别装入 UARTx_RDR 和 RB8 中，RI 置位，但必须满足下列条件：

（1）RI = 0

（2）用户自定义第 9 位模式

或者多机通讯模式下，接收的第 9 位 = 0，且前一 byte 接收的字节为地址且符合约定的从机地址（多机通讯模式）

或者，奇偶校验模式（优先级高于多机通讯模式和用户自定义第九位模式，详见奇偶校验章节）

如果这些条件被满足，那么第 9 位移入 RB8，8 位数据移入 UARTx_RDR，RI 被置位。否则接收的数据帧会丢失。在停止位的当中，接收器回到寻找 RXD 引脚上的另一个下降沿。用户应将 UARTx_RDR 内数据及时取走（或用软件写 1 清除 RI），然后才能再次接收。否则接收到 SHIFT 内的数据将会覆盖 UARTx_RDR 内未被取走的数据，且会产生 RXOV 错误标志。

Receive Timing of Mode 2

图23-9　　方式 2 接收时序图(STOP=1BIT)

方式 3：9 位 UART，可变波特率，异步全双工

方式 3 使用方式 2 的传输协议以及方式 1 的波特率产生方式。

图23-10　　方式 3 功能块框图

关于TI,TC标志的说明

TI@UARTx_FR为发送缓冲区（UARTx_TDR）空标志位。对UARTx_TDR写入数据后，TI将自动清零。但是需要注意的是，在连续发送数据时，第一个数据是直接写穿到发送移位寄存器（SHIFT）里，第二个数据才被写入到UARTx_TDR中。所以要在第二个数据写入到UARTx_TDR后，TI标志位才会清零。（如图23-11所示）

TC@UARTx_FR为帧发送结束标志位。在连续数据帧发送时，如果出现UART TDR和SHIFT同时为空的情况则置位TC，表示当前数据发送完成又无后续数据发送（一般意味着一个连续数据帧发送完成，所以叫帧发送结束标志位）。 对TDR写入，或置位TCC@UARTx_FR都可以清零TC 标志。具体时序如下图所示。

注意：SH33F2801是在最后1个字节的最后一个stop位被送出后才送出TC flag。

图23-11　　TC 工作时序图

23.1.3.2 可微调波特率

UART 自带一个波特率发生器，它实质上就是一个 15 位递增计数器。

图23-12 波特率发生器功能块框图

由图得到，波特率发生器的溢出率为

$$SBRT \quad overflow \quad rate = \frac{PCLK}{BRT}$$

因此，UART 在各模式下的波特率计算公式如下。

在方式 0 中，波特率可编程为 APB 时钟的 1/48 或 1/12，由 SM2 位决定。当 SM2 为 0 时，串行端口在 PCLK 时钟的 1/48 下运行。当 SM2 为 1 时，串行端口在 PCLK 时钟的 1/12 下运行。

在方式 1 和方式 3 中，波特率可微调，精度为一个 PCLK 时钟，公式如下：

$$BaudRate = \frac{PCLK}{16 \times (SBRT + 1) + BFINE}$$

例如：PCLK = 30MHz，需要得到 9600Hz 的波特率，SBRT 和 SFINE 值计算方法如下：

30000000/16/9600 = 195.3125

SBRT = 195 - 1 = 194

由 BaudRate 计算公式：9600 = 30000000/(16 X 195 + BFINE)

得到：BFINE = 5

此微调方式计算出的实际波特率为 9598，误差为 0.02%。

在方式 2 中，波特率固定为 PCLK 时钟的 1/256 或 1/128，由 SMOD2 位@UARTx_CR 中决定。当 SMOD2 位为 0 时，UART 以 PCLK 时钟的 1/256 运行。当 SMOD2 位为 1 时，UART 以 PCLK 时钟的 1/128 运行。

$$BaudRate = 2^{SMOD} \times \left(\frac{PCLK}{256} \right)$$

23.1.3.3 多机通讯

自动（硬件）地址识别

在方式 2 和方式 3 中，MULTIE 为 1，UART 运行状态如下：接收到停止位，RB8 的第 9 位为 1（地址字节标志），且接收到的数据字节符合 UART 的从机地址，UART 将开始接收后续数据字节。

第 9 位为 1 表明该字节是地址而非数据。当主机要发送一组数据给几个从机中的一个时，必须先发送目标从机地址。所有从机等待接收地址字节。自动地址识别的特点是只有地址匹配的从机才能产生后续接收数据的中断标志，硬件完成地址比较。

地址匹配的从机才能继续接收数据字节并产生接收中断标志。地址不匹配的从机不受影响，将继续等待接收和它匹配的地址字节。

使用自动地址识别功能时，主机可以通过调用给定的从机地址选择与一个或多个从机通信。主机使用广播地址可以寻址所有从机。有个特殊功能寄存器 UARTx_ADDR，包含了从机地址（SADDR）和地址屏蔽（SAMR）。从机地址是一个 8 位的字节,SAMR 用于定义 SADDR 各位的有效与否，如果 SAMR 中某一位为 0，则 SADDR 中相应位被忽略，如果 SAMR 中某一位置位，则 SADDR 中相应位将用于产生约定地址。这可以使用户在不改变 SADDR 寄存器中的从机地址的情况下灵活地寻址多个从机。

表23-2　　自动地址识别范例表

	从机 1	**从机 2**
SADDR	10100100	10100111
SAMR（为 0 的位被忽略）	11111010	11111001
约定地址	10100x0x	10100xx1
广播地址（SADDR 或 SAMR）	1111111x	11111111

从机 1 和从机 2 的约定地址最低位是不同的。从机 1 忽略了最低位，而从机 2 的最低位是 1。因此只与从机 1 通讯时，主机必须发送最低位为 0 的地址（10100000）。类似地，从机 1 的第 1 位为 0，从机 2 的第 1 位被忽略。因此，只与从机 2 通讯时，主机必须发送第 1 位为 1 的地址（10100011）。如果主机需要同时与两从机通讯，则第 0 位为 1，第 1 位为 0，第 2 位被两从机都忽略，两个不同的地址用于选定两个从机（1010 0001 和 1010 0101）。

主机可以通过广播地址与所有从机同时通讯。这个地址等于 SADDR 和 SAMR 的位或，结果中的 0 表示该位被忽略。多数情况下，广播地址为 0xFFh，该地址可被所有从机应答。

系统复位后，SADDR 和 SAMR 两个寄存器初始化为 0，这两个结果设定了约定地址和广播地址为 XXXXXXXX（所有位都被忽略）。这有效地去除了多从机通讯的特性，禁止了自动寻址方式。这样的 UART 将对任何地址都产生应答。用户可以按照上面提到的方法实现软件地址识别的多机通讯。

23.1.3.4 帧出错检测

UART 具有帧出错检测功能。错误标志位被置位后，只能通过软件对相应清除位写 1 来清零，尽管后续接收的帧没有任何错误也不会自动清零。该错误标志的置位不会触发中断，不影响串口数据继续发送，只供软件查询使用。

发送冲突

如果 UARTx_TDR 内有数据，用户软件继续写数据到 UARTx_TDR 寄存器时，发送冲突位 TXCOL@UARTx_FR 置位。如果发生了冲突，新数据会被忽略，不能被写入发送缓冲器。该位的清零通过对 TXCOLC@UARTx_FR 写 1 来实现。

接收溢出

UART 接收端存在 RDR 接收数据寄存器和移位寄存器两级，当 UARTx 收到一个字节后，RI 置 1，然后允许接收端最长延迟 1 个字节的串口时间读取 UARTx_RDR 数据，否则 RXOV 置 1。也就是说，当 RI 置 1 后，接收移位寄存器已满且又接收到新数据时，RXOV 置 1。RXOV 置 1 后，后续接收到的数据将继续进入接收移位寄存器中，即接收移位寄存器中的数据会被覆盖，但 RDR 中的数据不会被覆盖。当 UARTx_RDR 中的数据被读取后，等待移位寄存器接收到一帧完整的数据（包括起始位、数据位、停止位）的情况下，才将移位寄存器中的数据移入 UARTx_RDR，以避免错位。因此，一旦 RXOV@UARTx_FR 置 1，就表示有数据被丢弃了，丢失多少个数据是不确定的，这种情况一般要求用户重新进行该数据包的传输。

奇偶校验出错

如果检测到一个错误的奇偶校验位，那么奇偶校验位 PE@UARTx_FR 置位。该位的清零通过对 PEC@UARTx_FR 写 1 来实现。

帧出错

如果检测到一个无效（低）停止位，那么帧出错位 FE@UARTx_FR 置位。该位的清零通过对 FEC@UARTx_FR 写 1 来实现。　方式 1、2、3 都支持帧出错检测。

注意：*在发送之前 TXD，引脚必须被设置为输出高电平。*

23.1.3.5 奇偶校验

奇校验：校验位与传输数据（二进制表示）中‘1’的个数为奇数。

偶校验：校验位与传输数据（二进制表示）中‘1’的个数为偶数。

选择方式 0 和方式 1 时，没有奇偶校验功能。选择方式 2 和方式 3 时，若第 9 位功能选择位设置为奇偶校验位 MULTIE@UARTx_CR 为 2，第九位 TB9 将用为奇偶校验位，该位数据由移位寄存器生成并自动赋值到串口传输的第九位。校验方式选择由 PS@UARTx_CR 设定。

奇偶校验位为硬件自动检测，在每个字节传送完成后，若发生奇偶校验错误自动将就校验错误 PE@UARTx_FR 位置 1，该位由软件对 PEC@UARTx_FR 写 1 实现清零。

注意：无论校验是否正确，数据都会存入 RDR，校验位都会存入 RB8。

23.1.3.6 波特率检测

波特率检测模式由使能位 AUTOR@UARTx_CR 使能。该功能使 RXD 与 CAPTURE 模块连接，可供 RXD 用于自动波特率检测，注意在进入波特率检测模式时，RXD 信号将被内部直连到 CAPTURE，直到退出自动波特率检测模式。

23.1.3.7 LIN（局域互联网）总线模式

SH33F2801 的 LIN 总线模式支持同步间断（SYNCH BREAK）的硬件产生和检测。

LIN 模式通过 LINEN@UART_CR 进行控制。LIN 模式下，使用 UART 的 TXD 和 RXD 引脚作为 LIN 发送接收功能管脚，UART 的模式应选择 MODE1。

图23-13 LIN 总线模式

LIN 发送，传送字节与普通的 UART 类似。依照 LIN 总线协议，先发送同步间断（SBK@UART_CR）作为 LIN 总线上信号传输的开始，然后软件发送 0X55 作为波特率适应的同步域（SYNCH FIELD）数据，最后发送标识符场（IDENT FIELD）来控制总线上的数据通讯。主从机后续通信的数据格式均为 8 位数据（LSB），无校验位。在数据帧末端带有一个校验和场（CHECKSUM FIELD）。校验和场有两种：仅根据数据字节进行计算的校验和被称为传统的校验和，它可以用于 LIN1.3 从机通讯；根据数据字节和标识符进行计算的校验和被称为增强校验和，可以与 LIN2.0 从机进行通讯。主机根据校验和判断有没有达到目的，进而决定要不要再次发起 LIN 总线通讯。

LIN 接收，在 UART 的 RXD 接口有同步间断的检测电路，当检测到长度大于为 11 或 10 个 bit（通过 SBDL@ UART_CR 设置）的低电平（BREAK CHARACTER），再检测到至少 1 个 bit 的高电平（DELIMITER），即表示同步间断检测成功，产生一个 LBD 中断标志，若响应中断允许位 LBDIE 使能则进入中断服务程序。在 LBD 通过 LBDC 写 1 清零的时候需要将接收移位寄存器与接收缓存清空，并且关闭 LINEN,以便接收后续移入的串行数据。如还需要重新检测同步间断信号，需要重新使能 LINEN。

23.1.3.8 利用FIFO收发

对UART增加FIFO收发主要为减少UART对CPU的影响，避免CPU时间过多被低速外设通信设备占用。UART的FIFO功能是由开关控制的，当关闭该功能时，UART就是标准的模块。

UART FIFO 功能开关			
符号	类型	复位	说明
UARXFIFO	W/R	0	UART RX通道FIFO功能开关 0：标准UART 1：带RX FIFO的UART
UATXFIFO	W/R	0	UART TX通道FIFO功能开关 0：标准UART 1：带TX FIFO的UART

UART与FIFO之间的映射是可配置的，每个FIFO可映射到多个UART中的任意一个，硬件上没有做多个FIFO映射到同一个UART的错误检查，需要软件上避免这种情况。

有关映射表请参考“FIFO控制器”章节。

当FIFO功能未使能时，UART为标准的带一级发送和接收缓冲（TDR/RDR）的串行收发器。其操作和标志都和原标准UART一致。

在使能FIFO功能后，由TX FIFO和RX FIFO替换TDR和RDR，但原标准UART的一些控制位和标志位仍然有效。

UART TX FIFO

FIFO使能时， MCU先填充TX FIFO，UART移位寄存器从TX FIFO中取数，当TX FIFO读出 Counter逐步减小（每次减1），并小于指定的Trigger Level时，将置起FIFO发送触发标志位（TXTRIGF）。

TX FIFO写入位宽可选8/16/32位，当选择8位宽写入，FIFO写入 Counter每次加1，16位宽写入每次加2，32位宽写入每次加4，即TX FIFO Counter的基本计数单位是8bits（1个字节）。

TX FIFO的读出位宽固定为8bits，即UART移位寄存器每次都从TX FIFO中取走1个字节。如果涉及9bits UART模式，由原有的控制逻辑产生第9位，和TX FIFO无关。 比如9位奇偶校验模式，直接由TX SHIFT产生奇偶校验位并添加到数据末尾发送。如果是9位多机通信模式或9位自定义模式，第9bit先写入TB8，然后由UART控制器自动添加到数据末尾发送。

UART RX FIFO

FIFO使能时，UART从接收移位寄存器（RX SHIFT）获得数据并保存入RX FIFO，RX FIFO 写入Counter逐步增大，当大于指定的Trigger Level时，将置起FIFO接收触发标志位（RXTRIGF）。

RX FIFO读出位宽可选8/16/32位，当选择8位宽读出，FIFO 读出Counter每次减1，16位宽读出每次减2，32位宽读出每次减4，即RX FIFO Counter的基本计数单位是8bits（1个字节）。

对UART而言，RX FIFO的写入有两种模式：8bits模式和（8+3）bits模式， 通过RXMODE@FIFOx_CR控制位设定。当选择8bits模式时，只把8bits数据存入RX FIFO，FIFO Counter每次加1。 而CPU读取RX FIFO时只需要设置8bit读取位宽即可。

当选择（8+3）bits模式时，UART 把8bits数据存入FIFO前8bit，FIFO后3bit存放第9bit和两个Error Flag，如下表所示：

（8+3）bits RX FIFO Structure

RX FIFO Bit	Function
10	Frame Error Flag
9	Parity Error Flag
8	Ninth Bit（校验位/多机通信位/自定义位）

在RX FIFO使能的情况下，RB8、PE和FE这3个控制位/状态位仍然是有效的。PE和FE每次会根据UART 接收状态更新，但不会根据从FIFO中的读取结果更新（注）。而RB8有不同，因为RB8+RDR地址与RX FIFO地址相同，每次FIFO数据有更新，RB8也会相应更新，即RB8时刻反映着当前操作的数据情况。

注：PE和FE都是硬件置位软件清零，因此CPU如果没有对这两个标志处理，错误标志会一直保留。

CPU要读取（8+3）bits模式数据，除了设置RXMODE=1， 还需要设置RX FIFO的读取位宽为16bits，可读出完整的（8+3）bits，高位补零， 读取Counter每次减1（注）。 而如果设置读取位宽为8bits，则只读出前8bit数据，后3bit被丢弃，读取Counter每次减1。

注：RXMODE控制位除了决定UART往FIFO中写入多少位数据外，还决定了在（8+3）bits模式下从UART向FIFO写入的计数单位是一组接收数据，在（8+3）bits模式下一组数据就是11 bits，因此Counter计数只加1。

关于 UART FIFO 的说明

使能FIFO以后，在FIFO与UART的接口交互中，一些功能标志位会受影响，一些错误标志也会受影响，说明如下表。

表23-3　　开启 FIFO 以后 UART 功能和错误标志的变化

功能	关闭FIFO	开启FIFO
发送缓冲区（TDR）	**发送缓冲区**	**FIFO下无效**
接收缓冲区（RDR）	**接收缓冲区**	**FIFO下无效**
发送中断标志（TI）	初始化为1。TDR为空则置起，TDR为满则清零。 对TDR写入可清零TI，除非数据被写穿到移位寄存器中，则需再写入1 byte。	**FIFO下无效** FIFO引入了TXTRIG标志表示类似功能。 当FIFO中counter减计数达到trigger level时置位TXTRIG标志，该标志通过TXTRIGFC写1清零。
接收中断标志（RI）	初始化为0。 RDR为满则置起，读取RDR，或置位RIC可清零。	**FIFO下无效** FIFO引入了RXTRIG标志表示类似功能。 当FIFO中counter增计数达到trigger level时置位RXTRIG标志，该标志通过RXTRIGFC写1清零。
发送完成标志（TC）	初始化为1。 发送完成，TDR为空且移位寄存器中所有bit都发出后置位。 写TDR（不管是否写穿到移位寄存器中），或置位TCC可清零。	FIFO下有效。 在FIFO下发送完成指FIFO为空且移位寄存器中所有bit都发出。 写FIFO，或置位TCC可清零。
发送冲突（TXCOL）	TDR内有数据，用户继续写TDR时会产生。发生冲突后新写数据会被忽略。 置位TXCOLC可清零。	**FIFO下无效** FIFO未定义类似功能。 一般认为UART发送是软件可控的，完全可通过程序控制避免这种情况。
接收溢出（RXOV）	RDR内数据未被读取，又有接收数据进入时会产生。发生溢出后新接收数据被丢弃。 置位RXOVC可清零。	**FIFO下无效** FIFO引入了OVF标志表示类似功能。 在FIFO已满，而又有UART接收数据进入时会产生。在OVF以后FIFO内数据不变，新进数据会被丢弃。 该标志通过OVFC写1清零。
奇偶校验出错（PE）	奇偶校验错误时置位， 置位PEC可清零。	FIFO下有效。功能相同。
帧出错（FE）	检测到无效（低）停止位时置位。 置位FEC可清零。	FIFO下有效。功能相同。

23.1.3.9 中断控制

UART 的中断（TI,RI）在 UARTx_TDR 为空和 UARTx_RDR 为满时产生。
注意：当 TEN 未使能，TIE 使能时，会触发中断将数据搬移到发送缓存内，但是数据将会在使能 TEN 后才发送。
当包含有数据的一帧发送完成后，并且 TI=1 时，由硬件将 TC 位置 1。
LBD 中断发生在使能 LIN 总线功能后，在 RXD 上检测到 break 信号时发出。

图23-14　　UART 中断控制图

开启 FIFO 后，因为 TDR 和 RDR 被替换，TI 和 RI 中断源是无效的，而 TC 和 LDB 中断源仍然有效。FIFO 的中断请参考“FIFO 控制器”章节。

23.1.4　UART 寄存器

UART 模块寄存器列表 (基地址：0x4000 1800)

地址	寄存器名	说明
0x4000 1800	FR	UART中断标志寄存器
0x4000 1808	TDR	UART发送数据寄存器
0x4000 180C	RDR	UART接收数据寄存器
0x4000 1810	ADDR	UART地址配置寄存器
0x4000 1814	BRT	UART波特率配置寄存器
0x4000 1818	CR	UART控制寄存器

23.1.4.1 UART 中断标志寄存器 (UARTx_FR)

偏移地址: 0x0000

复位值: 0x0000 0006

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留								LBDC	PEC	FEC	RXOVC	TXCOLC	TCC	保留	RIC
-								WO	WO	WO	WO	WO	WO	-	WO
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								LBD	PE	FE	RXOV	TXCOL	TC	TI	RI
-								RO	RO	RO	RO	RO	RO	RO	RO
0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0

位	符号	说明
31 ~ 24	保留	-
23	**LBDC**	**LIN同步间隔中断标志清零位** 0：无操作 1：清零LBD
22	**PEC**	**UART奇偶校验出错标志位** 0：无操作 1：清零PE
21	**FEC**	**UART帧出错标志清零位** 0：无操作 1：清零FE
20	**RXOVC**	**UART接收溢出标志清零位** 0：无操作 1：清零RXOV
19	**TXCOLC**	**UART发送冲突标志清零位** 0：无操作 1：清零TXCOL
18	**TCC**	**UART全部一帧数据发送完毕标志清零位** 0：无操作 1：清零TC 注：TC通过对TCC写入1来清零
17	保留	-
16	**RIC**	**UART一字节数据接收中断标志清零位** 0：无操作 1：清零RI 注：RI通过对RIC写入1来清零，也可以对RDR读操作来清零
15 ~ 8	保留	-
7	**LBD**	**LIN同步间隔中断标志位** 0：未接收到同步间隔打断信号 1：接收到同步间隔打断信号 通过写LBDC清零。
6	**PE**	**UART奇偶校验出错标志位** 0：奇偶校验正确 1：奇偶校验出错，由硬件置位 注：对PEC写1清零

5	**FE**	**UART帧出错标志位** 0：无帧出错 1：帧出错，由硬件置位 注：对FEC写1清零
4	**RXOV**	**UART接收溢出标志位** 0：无接收溢出 1：接收溢出，由硬件置位 注：对RXOVC写1清零
3	**TXCOL**	**UART发送冲突标志位** 0：无发送冲突 1：发送冲突，由硬件置位 注：对TXCOLC写1清零
2	**TC**	**UART全部一帧数据发送完毕标志位** 0：全部一帧数据发送中，无置位 1：全部一帧数据发送完毕，由硬件置位 注：TC通过对TCC写入1来清零
1	**TI**	**UART一字节数据发送中断标志位** 0：单字节数据发送中，无置位 1：单字节数据发送完毕，由硬件置位 注：TI只能通过对TDR写操作来清零
0	**RI**	**UART一字节数据接收中断标志位** 0：单字节数据发接收中，无置位 1：单字节数据接收完毕，由硬件置位 注：RI通过对RIC写入1来清零，也可以对RDR读操作来清零

23.1.4.2 UART 发送数据寄存器 (UARTx_TDR)

偏移地址: 0x0008
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
FIFOTDH[31:9]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
FIFOTDH[31:9]							**TB8**	**TDR[7:0]**							
RW							*RW*	*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 9	**FIFOTDH[31:9]**	**UART FIFO发送数据高位寄存器** (仅在FIFO功能使能且操作位宽为16BIt/32Bit)
8	**TB8**	**在UART的方式2和3下发送的第9位**
7 ~ 0	**TDR[7:0]**	**UART的发送数据寄存器**

23.1.4.3 UART 接收数据寄存器 (UARTx_RDR)

偏移地址: 0x000C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
FIFORDH[31:9]															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
FIFORDH[31:9]							RB8	RDR[7:0]							
RO							RO	RO							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 9	**FIFORDH[31:9]**	**UART FIFO接收数据高位寄存器** (仅在FIFO功能使能且操作位宽为16BIt/32Bit)
8	**RB8**	**在UART的方式2和3下接收的第9位**
7 ~ 0	**RDR[7:0]**	**UART的接收数据寄存器**

23.1.4.4 UART 地址配置寄存器 (UARTx_ADDR)

偏移地址: 0x0010

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SMAR[7:0]								SADDR[7:0]							
RW								RW							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	**-**
15 ~ 8	**SMAR[7:0]**	**位屏蔽寄存器，决定了SADDR的哪些位会被检测**
7 ~ 0	**SADDR[7:0]**	**UART的硬件地址**

23.1.4.5 UART 波特率配置寄存器 (UARTx_BRT)

偏移地址: 0x0014

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留												BFINE[3:0]			
-												RW			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留	SBRT[14:0]														
-	RW														
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 20	保留	-
19 ~ 16	**BFINE[3:0]**	波特率发生器微调数据
15	保留	-
14 ~ 0	**SBRT[14:0]**	波特率发生器调节数据

23.1.4.6 UART 控制寄存器 (UARTx_CR)

偏移地址: 0x0018

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留					UATXFIFO	UARXFIFO	保留	FER	TNEG	RNEG	保留	保留	TEN	REN	LINEN
-					*RW*	*RW*	-	*RW*	*RW*	*RW*	-	-	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SBK	SM[1:0]		MULTIE[1:0]		PS	保留	SMOD0	LBDL	LBDIE	TCIE	TIE	RIE	SMOD2	SBRTEN	STOP
RW	*RW*		*RW*		*RW*	-	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 27	保留	-
26	**UATXFIFO**	**UART TX FIFO功能开关** 0：标准UART 1：带TX FIFO的UART
25	**UARXFIFO**	**UART RX FIFO功能开关** 0：标准UART 1：带RX FIFO的UART
24	保留	-
23	**FER**	**帧错误接收允许位** 0:禁止停止位确认检验，任何停止位都会置位RI 1:允许停止位确认检验，只有有效的停止位(1)才能置位RI
22	**TNEG**	**UART发送极性控制位** 0：发送数据不取反 1：发送数据取反
21	**RNEG**	**UART接收极性控制位** 0：接收数据不取反 1：接收数据取反
20	保留	-
19	保留	-
18	**TEN**	**UART发送功能使能位** 0：禁用发送功能 1：使能发送功能
17	**REN**	**UART接收功能使能位** 0：禁用接收功能 1：使能接收功能
16	**LINEN**	**LIN模式功能使能位** 0：禁止LIN模式 1：使能LIN模式

15	**SBK**	**LIN同步间隔使能位** 0：不发送同步间隔 1：发送同步间隔，发送完成后自动清零
14 ~ 13	**SM[1:0]**	**UART串行方式控制位** 00：方式0，同步方式，固定波特率 01：方式1，8位异步方式，可变波特率 10：方式2，9位异步方式，固定波特率 11：方式3，9位异步方式，可变波特率
12 ~ 11	**MULTIE[1:0]**	**第九位功能选择位** 00：软件自定义 01：多机通讯标志位 1x：奇偶校验位
10	**PS**	**奇偶校验选择位** 0:奇校验 1:偶校验
9	**保留**	**-**
8	**SMOD0**	**UART方式0波特率控制位** 0：在方式0中，波特率为PCLK1时钟的1/48 1：在方式0中，波特率为PCLK1时钟的1/12
7	**LBDL**	**同步间隔检测阈值长度** 0：低电平长度大于10BIT低电平 1：低电平长度大于11BIT低电平
6	**LBDIE**	**LIN打断分隔符中断使能位** 0：禁止LIN打断分隔符中断功能 1：允许LIN打断分隔符中断功能
5	**TCIE**	**全部数据发送完成中断使能位** 0：禁止数据发送完成标志位TC触发中断功能 1：允许数据发送完成标志位TC触发中断功能
4	**TIE**	**单字节传送中断使能位** 0：禁止传输中断标志位TI触发中断 1：允许传输中断标志位TI触发中断
3	**RIE**	**单字节接收中断使能位** 0：禁止接收中断标志位RI触发中断 1：允许接收中断标志位RI触发中断
2	**SMOD2**	**UART方式2波特率控制位** 0：在方式2中，波特率为PCLK1时钟的1/256 1：在方式2中，波特率为PCLK1时钟的1/128
1	**SBRTEN**	**UART波特率发生器使能控制位** 0：禁用波特率发生器 1：使能波特率发生器
0	**STOP**	**停止位位数选择位** 0：选择使用1位停止位 1：选择使用2位停止位

23.2 串行外设接口（SPI）

23.2.1 简介

串行外部设备接口（SPI）是一种高速串行通信接口，允许 MCU 与外围设备（包括其它 MCU）进行全双工，同步串行通讯。

下图所示即为典型的由一个主设备和若干从属外部设备组成的SPI总线网络，主设备通过 3 条线连接所有从设备，主设备控制连接从属设备$\overline{SS}$引脚的 4 个并行端口来选中其中一个从属设备进行通讯。

图23-15　　SPI 总线网络图

23.2.2 主要特性

- 全双工，三线同步传输
- 主从机操作
- 8 个可编程主时钟频率
- 极性相位可编程的串行时钟
- 数据字位宽 8、16、32 可选
- 带 MCU 中断的主模式故障出错标志
- 写入冲突标志保护
- 可选择 LSB 或 MSB 传输
- 可配置 FIFO，支持基于 FIFO 的数据收发

23.2.3 功能描述

23.2.3.1　信号描述

- **主输出从输入（MOSI）**

该路信号连接主设备和从设备。数据通过 MOSI 从主设备串行传送到从设备，主设备输出，从设备输入。

- **主输入从输出（MISO）**

该路信号连接从设备和主设备。数据通过MISO从从设备串行传送到主设备，从设备输出，主设备输入。当SPI配置为从设备并未被选中（$\overline{SS}$引脚为高电平），从设备的MISO引脚处于高阻状态。

- **SPI 串行时钟（SCK）**

SCK信号用作控制MOSI和MISO线上输入输出数据的同步移动。每 8 时钟周期线上传送一个字节。如果从设备未被选中（$\overline{SS}$引脚为高电平），SCK信号被此从设备忽略。

- **从设备选择引脚（$\overline{SS}$）**

每个从属外围设备由一个从选择引脚（$\overline{SS}$引脚）选择，当引脚信号为低电平时，表明该从设备被选中。主设备可以通过软件控制连接于从设备$\overline{SS}$引脚的端口电平选择每个从设备，很明显，只有一个主设备可以驱动通讯网络。为了防止MISO总线冲突，同一时间只允许一个从设备与主设备通讯。在主设备模式中，$\overline{SS}$引脚状态关联SPI标志寄存器SPIx_FR中MODF标志位以防止多个主设备驱动MOSI和SCK。

下列情况，$\overline{SS}$引脚可以作为普通端口或其它功能使用：

（1） 设备配置为主设备，SPI 控制寄存器 SPIx_CR 寄存器的 SSDIS 位置 1。这种配置仅仅存在于通讯网络中只有一个主设备的情况，因此，SPI 状态寄存器 SPSTA 中 MODF 标志位不会被置 1。

（2）设备配置为从设备，SPI控制寄存器SPIx_CR的CPHA位和SSDIS位置 1。这种配置情况存在于只有一个主设备一个从设备的通讯网络中，因此，设备总是被选中的，主设备也不需要控制从设备的$\overline{SS}$引脚选择其作为通讯目标。

注意：*当CPHA = '0'时，$\overline{SS}$ 引脚产生下降沿表示启动发送。*

23.2.3.2　　波特率

在主模式下，SPI 的波特率有八种可选择的频率，分别是内部时钟的 2，4，8，16，32，64，128，256，512,1024 分频，可以通过设定 SPIx_CR 寄存器的 SPR[3:0]位进行选择。

23.2.3.3　　原理描述

下图所示是 SPI 模块的详细结构。

图23-16　　SPI 模块框图

23.2.3.4 工作模式

SPI 可配置为主模式或从属模式中的一种。SPI 模块的配置和初始化通过设置 SPIx_CR 寄存器来完成（建议先配置好各个控制模式位，再使能 SPEN@SPIx_CR）。配置完成后，通过设置 SPIx_CR，SPIx_TDR，SPIx_RDR（串行外围设备数据寄存器）来完成数据传送。

在SPI通讯期间，数据同步地被串行的移进移出。串行时钟线（SCK）使两条串行数据线（MOSI和MISO）上数据的移动和采样保持同步。从设备选择线（$\overline{SS}$）可以独立地选择SPI从属设备；如果从设备没有被选中，则不能参与SPI总线上的活动。

当 SPI 主设备通过 MOSI 线传送数据到从设备时，从设备通过 MISO 线发送数据到主设备作为响应，这就实现了在同一时钟下数据发送和接收的同步全双工传输。对 SPI 数据发送寄存器 SPIx_TDR 进行写操作，当移位寄存器为空时将写入发送移位寄存器，当移位寄存器内有数据时将写入发送缓存 SPIx_TDR。对 SPIx_RDR 寄存器进行读操作将获得接收缓存内的数据。

图23-17 全双工主从互联图

■ 主模式

(1) 模式启动

SPI 主设备控制 SPI 总线上所有数据传送的启动。当 SPIx_CR 寄存器中的 MSTR 位置 1 时，SPI 在主模式下运行，只有一个主设备可以启动传送。

(2) 发送

在 SPI 主模式下，写一个字节数据到 SPI 数据寄存器 SPIx_TDR，若发送移位寄存器为空，数据将会写入发送移位缓冲器，若发送移位寄存器非空，数据将被放在发送缓存 SPIx_TDR 内，当发送移位寄存器发送完毕，数据会从缓存 SPIx_TDR 自动调入移位寄存器，并产生 SPTI 中断。如果发送缓存为满时还向 SPIx_TDR 写数据，那么主 SPI 产生一个 WCOL 信号以表明写入太快。但是在发送缓存和发送移位寄存器中的数据不会受到影响，发送也不会中断。另外如果发送移位寄存器为非空，那么主设备立即按照 SCK 上的 SPI 时钟频率串行地移出发送移位寄存器中的数据到 MOSI 线上。当发送缓存 SPIx_TDR 为空时，SPIx_FR 寄存器中的 SPTI 位被置 1。如果 SPTIE 中断被允许，当 SPTI 位置 1 时，会产生一个中断。SPTI@SPIx_FR 将在 SPIx_TDR 内被填充数据后自动清零。

(3) 接收

当主设备通过 MOSI 线传送数据给从设备时，相对应的从设备同时也通过 MISO 线将其发送移位寄存器的内容传送给主设备的接收移位寄存器，实现全双工操作。从设备接收的数据按照 MSB 或 LSB 优先的传送方向存入主设备的接收移位寄存器。当一个字节的数据完全被移入接收寄存器时，数据将被自动移入 SPIx_RDR，并产生 SPRI 中断。处理器可以通过读 SPIx_RDR 寄存器获得该数据。接收中断标志位 SPRI 在 SPIx_RDR 为满时被置 1，当 SPIx_RDR 内数据被读走为空后，SPRI 将自动清零。如果发生超限（接收到的数据未被及时取走，SPIx_RDR 与接收移位寄存器内都有数据，就试图开始下一次接收），RXOV 位置 1，表示发生数据超限，此时接收移位寄存器保持原有数据并且 SPRI 位置 1，这样直到 SPRI 位被清 0，SPI 主设备将不会接收任何数据。

■ 从模式

(1) 模式启动

当SPIx_CR寄存器中的MSTR位清 0，SPI在从模式下运行。在数据传送之前，从设备的$\overline{SS}$引脚必须被置低，而且必须保持低电平直到一个字节数据传送完毕。

(2) 发送与接收

从属模式下，按照主设备控制的 SCK 信号，数据通过 MOSI 引脚移入，MISO 引脚移出。一个位计数器记录 SCK 的边沿数，当接收移位寄存器移入 8 位数据（一个字节）后开始发送移位寄存器移出 8 位数据（一个字节）。数据可以通过读取 SPIx_RDR 寄存器获得。如果 SPRIE 中断被允许，当 SPRI 置 1 时，也会产生一个中断。

SPI 从设备不能启动数据传送，所以 SPI 从设备必须在主设备开始一次新的数据传送之前将要传送的数据写入发送寄存器 SPIx_TDR（当只写入一个数据时，直接写入发送移位寄存器，当写入两个数据时，分别存放在发送移位寄存器与发送缓存 SPIx_TDR 内）。如果从设备在第一次开始发送之前未写入数据，从设备将传送“0x00”字节给主设备。如果写 SPIx_TDR 操作时，发送缓存 SPIx_TDR 与发送移位寄存器内都有数据，那么 SPI 从设备的 WCOL 标志位置 1，即如果传送缓存寄存器 SPIx_TDR 已经含有数据，SPI 从设备的 WCOL 位置 1，表示写 SPIx_TDR 冲突。但是发送缓存与发送移位寄存器内的数据不受影响，传送也不会被中断。

23.2.3.5 传送形式

通过软件设置 SPIx_CR 寄存器的 CPOL 位和 CPHA 位，用户可以选择 SPI 时钟极性和相位的四种组合方式。CPOL 位定义时钟的极性，即空闲时的电平状态，它对 SPI 传输格式影响不大。CPHA 位定义时钟的相位，即定义允许数据采样移位的时钟边沿。在主从通讯的两个设备中，时钟极性相位的设置应一致。

图23-18 数据传送形式（CPHA = 0）图

如果CPHA = 0，SCK的第一个沿捕获数据，从设备必须在SCK的第一个沿之前将数据准备好，因此，在从机模式下，$\overline{SS}$引脚的下降沿从设备开始发送数据。$\overline{SS}$引脚在每次传送完一个字节之后必须被拉高，在发送下一个字节之前重新设置为低电平，因此当CPHA = 0，SSDIS不起作用。另外, 针对该模式，SPI还添加了SPSFF@SPIx_CR控制位，置位该控制位后，第一byte数据还需要$\overline{SS}$引脚的下降沿载入，从第二byte数据开始将使用前一byte最后一个CLK沿载入数据，以达到连续发送的目的。

图23-19　　数据发送形式（CPHA = 1）图

如果CPHA = 1，主设备在SCK的第一个沿将数据输出到MOSI线上，从设备把SCK的第一个沿作为开始发送信号。用户必须在第一个SCK的第二个沿之前完成写SPIx_TDR的操作。$\overline{SS}$引脚在每个字节数据的传送过程始终保持低电平。这种数据传输形式是一个主设备一个从设备之间通信的首选形式。

图23-20　　CPHA/NSS 时序图

***注意：**当 SPI 用作从设备模式，且 SPIx_CR 寄存器的 CPOL 位清 0，SCK 端口必须设置为输入模式，并在数据传输使能位置 1 前打开上拉电阻。*

23.2.3.6 数据字8、16、32 位可设置

单次传输的数据字长度可通过寄存器 SPIx_CR 的 SPDATL 位配置为 8 位、16 位、32 位。采用 16 位/32 位字长时，移位寄存器与缓存寄存器均被扩展为 16 位/32 位模式，对 SPIx_TDR 与 SPIx_RDR 的读写均为 16 位/32 位操作。（注：TDR、RDR 设计为 32 位寄存器，在 8 位/16 位读写时，自动忽略高位），时序图如下（以 16 位为例）：

数据传送形式（**CPHA = 0**）

数据发送形式（**CPHA = 1**）

23.2.3.7 SPSTA 状态异常情况

SPIx_FR 寄存器中的 MODF,WCOL,RXOV 标志位表示在 SPI 通讯中的出错情况：

(1) 模式故障（MODF）

SPI主模式下的模式故障出错表明$\overline{SS}$引脚上的电平状态与实际的设备模式不一致。SPIx_FR寄存器中MODF位置 1 后，表明系统控制存在多主设备冲突的问题。这种情况下，SPI系统受到如下影响：

产生 SPI 接收/错误 CPU 中断请求，在中断中需要软件完成以下操作；

禁止 SPI，切为从机模式，重新使能 SPI。

注意：以上只是恢复了模式故障节点的基本 SPI 功能，要彻底解决主设备冲突问题（比如还是有多主设备需求），需要从整体进行设计。

当SPIx_CR寄存器的$\overline{SS}$引脚禁止位（SSDIS）清 0，$\overline{SS}$引脚信号为低时，MODF标志位置 1。然而，对于只有一个主设备的系统来说，主设备的$\overline{SS}$引脚被拉低，那决不是另外一个主设备试图驱动网络。这种情况下，为防止MODF置 1，可使SPIx_CR寄存器中的SSDIS位置 1，$\overline{SS}$引脚作为普通I/O口或是其它功能引脚。

重新启动串行通信时，用户必须将 MODF 位软件清 0，将 SPIx_CR 寄存器中的 MSTR 位和 SPIEN 位置 1，重新启动主模式。

(2) 写冲突（WCOL）

在发送数据序列期间写入 SPIx_TDR 寄存器会引起的写冲突，SPIx_FR 寄存器中的 WCOL 位置 1。WCOL 位置 1 不会引起中断，发送也不会中止。WCOL 位需由软件清 0。

(3) 超限情况（RXOV）

主设备或从设备尚未清除 SPRI 位，主或从设备又试图发送几个数据字节时，超限情况发生。在这种情况下，接收移位寄存器保持原有数据，SPRI 置 1，同样 SPI 设备直到 SPRI 被清除后才会再接收数据。在 SPRI 位被清除之前继续调用中断，发送也不会中止。RXOV 位置 1 不会引起中断，RXOV 位需由软件清 0。

23.2.3.8 利用 FIFO 收发

对SPI增加FIFO收发的目的与UART相同，鉴于SPI通信速率要远高于UART，FIFO深度也要相应高一些，SPI只支持FIFO2（8级深度）。

SPI FIFO 功能开关			
符号	类型	复位	说明
FIFOEN	W/R	0	SPI FIFO功能开关 0：标准SPI 1：带FIFO的SPI

注：SPI的FIFO总是成对出现，因此TX FIFO和RX FIFO只需要定义一个功能开关FIFOEN。

SPI与FIFO之间的映射请参考“FIFO控制器”章节。

当FIFO功能未使能时，SPI为标准的带一级发送和接收缓冲（TDR/RDR）的串行收发器。其操作和标志都和原标准SPI一致。

在使能FIFO功能后，由TX FIFO和RX FIFO替换原SPI的TDR和RDR，但原SPI的一些控制位和标志位仍然有效。

SPI TX FIFO

FIFO使能时， MCU先填充TX FIFO，SPI移位寄存器从TX FIFO中取数，当TX FIFO读出 Counter逐步减小（每次减1），并小于指定的Trigger Level时，将置起FIFO发送触发标志位（TXTRIGF）。

TX FIFO写入位宽可选8/16/32位，当选择8位宽写入，FIFO写入 Counter每次加1，16位宽写入每次加2，32位宽写入每次加4，即TX FIFO Counter的基本计数单位是8bits（1个字节）。

TX FIFO的读出位宽可选8/16/32位（对应于SPI的8/16/32位位宽模式），即SPI移位寄存器每次都从TX FIFO中取走1、2或4个字节，相应的Counter也减小1、2或4。

SPI RX FIFO

由于SPI使用的FIFO2模块也可配置用于UART，因此该RX FIFO仍然保留了（8+3）bits的存储结构，但当映射到SPI时就固定为8 bits模式。

FIFO使能时，SPI从接收移位寄存器（RX SHIFT）获得数据并保存入RX FIFO，RX FIFO 写入Counter逐步增大，当大于指定的Trigger Level时，将置起FIFO接收触发标志位（RXTRIGF）。

RX FIFO读出位宽可选8/16/32位，当选择8位宽读出，FIFO 读出Counter每次减1，16位宽读出每次减2，32位宽读出每次减4，即RX FIFO Counter的基本计数单位是8bits（1个字节）。

因为SPI有8/16/32位位宽模式，在FIFO情况下每次SPI接收会接收到8、16或32位数据，相应的FIFO Counter就会增加1、2或4。

关于 SPI FIFO 的说明

使能FIFO以后，在FIFO与SPI的接口交互中，一些功能标志位会受影响，一些错误标志也会受影响，说明如下表。

表23-4　　开启 FIFO 以后 SPI 功能和错误标志的变化

功能	关闭FIFO	开启FIFO
发送缓冲区（TDR）	**发送缓冲区**	**FIFO下无效**
接收缓冲区（RDR）	**接收缓冲区**	**FIFO下无效**
发送中断标志（SPTI）	初始化为1。TDR为空则置起，TDR为满则清零。 对TDR写入可清零SPTI，除非数据被写穿到SPI移位寄存器中，则需再写入1 byte。 置位SPTIC也可清零SPTI。	**FIFO下无效** FIFO引入了TXTRIG标志表示类似功能。 当FIFO中counter减计数达到trigger level时置位TXTRIG标志，该标志通过TXTRIGFC写1清零。
接收中断标志（SPRI）	初始化为0。 RDR为满则置起，读取RDR，或置位SPRIC可清零。	**FIFO下无效** FIFO引入了RXTRIG标志表示类似功能。 当FIFO中counter增计数达到trigger level时置位RXTRIG标志，该标志通过RXTRIGFC写1清零。
发送冲突（WCOL）	TDR内有数据，用户继续写TDR时会产生。发生冲突后新写数据会被忽略。 置位WCOLC可清零。	**FIFO下无效** FIFO未定义类似功能。 一般认为发送是软件可控的，完全可通过程序控制避免这种情况。
接收溢出（RXOV）	RDR内数据未被读取，又有接收数据进入时会产生。发生溢出后新进数据会被忽略。 置位RXOVC可清零。	**FIFO下无效** FIFO引入了OVF标志表示类似功能。 在FIFO已满，而又有接收数据进入时会产生。在OVF以后FIFO内数据不变，新进数据会被忽略。该标志通过OVFC写1清零。
模式故障错误（MODF）	$\overline{SS}$引脚上的电平状态与实际的设备模式不一致时置位。 置位MODFC可清零。	FIFO下有效。功能相同。

23.2.3.9 中断

三种 SPI 状态标志 SPTI、SPRI & MODF 能产生一个 CPU 中断请求。

串行外围设备数据发送接收标志：

SPI 的中断（SPTI、SPRI）在 SPIx_TDR 和 SPIx_RDR 缓存完成一字节数据传输后（发送时为发送寄存器为空，接收时为接收寄存器为满）产生。

模式故障标志，MODF：该位被置 1 表示$\overline{SS}$引脚上的电平与SPI模式不一致的。SSDIS位为 0 并且MODF置 1 将产生SPI接收器/出错CPU中断请求。当SSDIS置 1 时，无MODF中断请求产生。

图23-21 SPI 中断控制图

开启 FIFO 后，因为 SPI 的 TDR 和 RDR 被替换，SPTI 和 SPRI 中断源是无效的，而 MODF 中断源仍然有效。FIFO 的中断请参考“FIFO 控制器”章节。

23.2.4 SPI 寄存器

SPI 模块寄存器列表 (基地址：0x4000 2400)

地址	寄存器名	说明
0x4000 2400	FR	SPI中断标志寄存器
0x4000 2404	TDR	SPI发送数据寄存器
0x4000 2408	RDR	SPI接收数据寄存器
0x4000 240C	CR	SPI控制寄存器

23.2.4.1 SPI 中断标志寄存器 (SPI_FR)

偏移地址: 0x0000
复位值: 0x0000 0002

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留										WCOLC	RXOVC	MODFC	保留	SPTIC	SPRIC
-										*WO*	*WO*	*WO*	-	*WO*	*WO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留										WCOL	RXOV	MODF	BUSY	SPTI	SPRI
-										*RO*	*RO*	*RO*	*RO*	*RO*	*RO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0

位	符号	说明
31 ~ 22	保留	-
21	**WCOLC**	写入冲突标志清除位 0：无效 1：清除
20	**RXOVC**	接收溢出标志清除位 0：无效 1：清除
19	**MODFC**	模式故障标志清除位 0：无效 1：清除
18	保留	-
17	**SPTIC**	该控制位保留，但不再使用
16	**SPRIC**	该控制位保留，但不再使用
15 ~ 6	保留	-
5	**WCOL**	写入冲突标志位 0：已处理或没有写入冲突，由软件对WCOLC位写1清零 1：检测到写入冲突，由硬件置位
4	**RXOV**	接收溢出标志位 0：已处理或无接收溢出，由软件对RXOVC位写1清零 1：发生接收溢出，由硬件置位
3	**MODF**	模式故障标志位 0：无故障，由软件对MODFC位写1清零 1：NSS引脚电平与SPI模式不一致，由硬件置位

2	**BUSY**	**主机BUSY状态标志位** 0：主机处于空闲状态 1：主机处于发送状态，BUSY
1	**SPTI**	**SPI一字节数据传送中断标志位** 0：发送数据寄存器TDR为满的状态，由软件对SPTIC位写1清零 1：发送数据寄存器TDR为空的状态，由硬件置位
0	**SPRI**	**SPI一字节数据接收中断标志位** 0：接收数据寄存器RDR为空的状态，由软件对SPRIC位写1清零 1：接收数据寄存器RDR为满的状态，由硬件置位

23.2.4.2 SPI 发送数据寄存器 (SPI_TDR)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
TDR[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TDR[31:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**TDR[31:0]**	**发送缓存寄存器** 当SPDATL=0,为8位SPI发送缓存寄存器 当SPDATL=1,为16位SPI发送缓存寄存器 当SPDATL=2,为32位SPI发送缓存寄存器

23.2.4.3 SPI 接收数据寄存器 (SPI_RDR)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
RDR[31:0]															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
RDR[31:0]															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 0	**RDR[31:0]**	**接收缓存寄存器** 当SPDATL=0,为8位SPI接收缓存寄存器 当SPDATL=1,为16位SPI接收缓存寄存器 当SPDATL=2,为32位SPI接收缓存寄存器

23.2.4.4 SPI 控制寄存器 (SPI_CR)

偏移地址: 0x000C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留													FIFOEN	SPDATL[1:0]	
-													RW	RW	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
SPSFF	SPIEN	保留	保留	SPTIE	SPRIE	保留	DIR	MSTR	CPHA	CPOL	SSDIS	SPR[3:0]			
RW	RW	-	-	RW	RW	-	RW	RW	RW	RW	RW	RW			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 19	保留	-
18	**FIFOEN**	**SPI RX FIFO功能开关** 0：标准SPI 1：带FIFO的SPI 注：SPI FIFO总是TX/RX成对出现，只需一个使能开关
17 ~ 16	**SPDATL[1:0]**	**数据字长度选择位** 00：每次发送8bit数据 01：每次发送16bit数据 1x： 每次发送32bit数据
15	**SPSFF**	**从机快速发送模式控制位（CPHA=0时有效）** 0:普通模式，从机的SS引脚在每次传送完一个数据（8位或16位）后必须被拉高，发送下一个数据前必须从新拉低SS以载入数据到移位寄存器 1:快速模式，从机发送的第一个数据需要用SS的下降沿来触发，第二个数据则利用上个数据最后一个CLK来加载到移位寄存器（注，若发送缓存内没有数据则不会加载）
14	**SPIEN**	**SPI使能位** 0：关闭SPI 1：使能SPI
13	保留	-
12	保留	-
11	**SPTIE**	**SPI的传送中断标志使能位** 0：禁止传送中断标志位SPTI触发中断 1：允许传送中断标志位SPTI触发中断
10	**SPRIE**	**SPI的接收中断标志使能位** 0：禁止接收中断标志位SPRI触发中断 1：允许接收中断标志位SPRI触发中断
9	保留	-
8	**DIR**	**数据传送方向选择位** 0：MSB优先发送 1：LSB优先发送
7	**MSTR**	**SPI主从设备配置位** 0：配置SPI为从设备 1：配置SPI为主设备
6	**CPHA**	**时钟相位控制位** 0：CK周期的第一沿采集数据 1：CK周期的第二沿采集数据

5	**CPOL**	**时钟极性控制位** 0：在IDLE状态下CK处于低电平 1：在IDLE状态下CK处于高电平
4	**SSDIS**	**NSS引脚控制位** 0：在主/从模式下，打开NSS引脚 1：在主/从模式下，关闭NSS引脚，将不会产生MODF中断请求 在从模式下，如果CPHA=0，该位不起作用
3 ~ 0	**SPR[3:0]**	**时钟分频控制位** 0000：PCLK1 /2 0001：PCLK1 /4 0010：PCLK1 /8 0011：PCLK1 /16 0100：PCLK1 /32 0101：PCLK1 /64 0110：PCLK1 /128 0111：PCLK1 /256 1000：PCLK1 /512 1001：PCLK1 /1024 其他：保留

23.3 两线串行接口（TWI）

23.3.1 简介

TWI（Two-wire Serial Interface）接口是对 I^2C 总线接口的继承和发展，完全兼容 I^2C 总线，具有硬件实现简单、软件设计方便、运行可靠和成本低廉的优点。TWI 由一根时钟线和一根数据线组成，以字节为单位进行传输，可兼容 SMBus 总线规范，自动对字节传输进行处理，并对串行通信进行跟踪。

SCL/SDA 是 TWI 总线的信号线。 SDA 是双向数据线，SCL 是时钟线。在 TWI 总线上传送数据，首先送最高位(MSB)，由主机发出起始信号，然后由主机发送一个或多个字节的数据，数据传送完毕，由主机发出停止信号，完成一次最简单的 TWI 传输。

典型 TWI 总线应用如下图所示，最高支持 128 个不同的器件进行通讯。

图23-22　　典型 TWI 总线应用框图

23.3.2 主要特性

- 两线模式
- OD 输出(需在 GPIO 中配置)，通讯电平不受 VDD 影响，也不会影响到 VDD
- 支持主机模式(Master)和从机模式(Slave)
- 允许发送(Transmitter)和接收(Receiver)
- 支持多主机通讯的仲裁功能
- 具有低电平总线超时判断(Timeout)
- 在睡眠模式下可唤醒系统，不支持停机模式唤醒
- 地址可编程，带多个地址屏蔽位，支持广播功能
- 支持标准模式和快速模式

23.3.3 功能描述

23.3.3.1 总线信号时序

总线初始状态为总线空闲，即 SDA 和 SCL 两条信号线都处于高电平，总线上所有的器件都释放总线，两条信号线各自的上拉电阻把电平拉高。

数据传输中数据线上每一位的传输均需要时钟线上一个脉冲。在时钟高电平时数据线应保持稳定，在时钟低电平时允许数据线改变。但起始条件和停止条件是特殊的，前者指时钟高电平期间的数据线的下降沿，后者指时钟高电平期间的数据线上升沿。

图23-23 TWI 总线的基本位传输特性

图23-24 TWI 总线的起始条件（起始信号）和停止条件（停止信号）

在 TWI 总线中，起始条件和停止条件均由主机发出。

主机可以发起和终结一次传输。当主机发送一个起始条件时开始一次传输， 发送一个停止条件时结束本次传输。在起始条件和停止条件之间，总线定义为“忙碌”状态。其它主机不应该去试图发起传输。在“忙碌”状态下，如果主机再次发送起始条件，则定义为“重复起始条件”，表示主机希望不放弃总线的情况下开始一次新的传输。发送重复起始条件后，总线仍处于“忙碌”状态，一直到总线出现停止条件。鉴于重复起始条件和起始条件性质完全一致，除非特别声明，本文中将采用起始条件来代替两者。

所有数据包（包括地址包）均有 9 位组成，包括 1 个字节和一个应答位。主机负责发出时钟，起始条件和停止条件，接收者负责给出应答信号。接收者通过在第九个时钟脉冲处将数据线拉低发出“应答(ACK)”信号；或维持第九个脉冲处维持高电平表示“不应答(NACK)”信号。当接收方接收到最后一个字节，或因某种原因无法继续接收数据时，应回应“不应答(NACK)”信号。

图23-25 ACK 和 NACK 信号

由于某种原因从机不对主机寻址信号应答时（如从机正在进行实时性处理工作而无法响应总线通信），它必需返回一个 NACK，再由主机产生一个停止信号以结束总线的数据传送。

如果从机对主机进行了应答，但在数据传送一段时间后无法继续接收更多数据时，可以通过返回一个 NACK 通知主机，主机则应发出停止信号以结束数据的继续传送。

当主机接收数据时收到最后一个数据字节后，必须向从机发送一个 NACK 通知从机停止数据发送，然后从机释放 SDA 线，以允许主机产生一个停止信号。

注：TWI 是双向数据通信，一节点要通过 SDA 发送数据时要求其它节点释放 SDA 线，即 SDA 线处于外部上拉状态。

在 TWI 总线中，采用从高到低逐位进行传输。

一次传输通常包括一个起始条件，地址+读/写，一个或多个数据包和一个停止条件。仅包含起始条件和停止条件的数据格式是不合通讯规则的。值得注意的是“线与”结构给主机和从机之间的握手信号提供了方便。当主机相对太快或从机需要处理其它事务时，从机可以通过拉低时钟线来拉长时钟线的低电平时间，从而降低通讯频率。从机可以拉长时钟线的低电平周期但不会影响到时钟线高电平的周期。

在产生应答信号时，SH33F2801 拉低 SDA 信号线。中断标志位置起期间，SH33F2801 拉低 SCL 信号线，释放 SDA 信号线。中断处理完毕后清除 TWINT 标志，释放 SCL 线。

图23-26　　一次典型的 TWI 数据传输时序图

主控器完成一次通信后还想继续占用总线进行通信，不释放总线，就要利用重复起始条件 Rs。它既作为前一次数据传输的结束，又作为后一次传输的开始。

特殊情况下，若需禁止所有发生在 TWI 总线上的通信，可以封锁或关闭总线，具体操作为总线上任一器件将 SCL 锁定在低电平即可。

23.3.3.2 时钟同步

当多个主机同时希望控制总线时，总线将依据“线与”原则决定时钟线高低电平。对于所有参与传输的主机来说，定义清楚每一个时钟脉冲的起始是相当重要的。

时钟线电平的由高到低跳变将导致所有参与传输的器件开始低电平计时。每一个器件计时达到自己低电平要求时释放时钟线，在时钟线变为高电平之前进入高电平等待期；当所有器件均计满低电平周期时，时钟线才变为高电平。之后所有器件开始对高电平进行计时，第一个计满高电平周期的器件将拉低时钟线，进入下一个时钟周期。

图23-27 时钟同步时序图

注：当多个主机同时控制 SCL 线时，“线与”会导致 SCL 线的低电平周期由低电平时间最长的主机决定，而高电平周期由高电平时间最短的主机决定。

23.3.3.3 总线仲裁

主机只有在总线处于“空闲”状态时才能开始一次传输。两个或多个主机可能在最小持续时间($t_{HOLD:STA}$))内同时发送起始条件，结果在总线上会得到一个正常的起始条件，多个主机将开始发送，后续的传输需要进行仲裁。

由于发送起始条件的主机无法知道是否有其它主机在竞争总线，只能靠 SCL 高电平期间对 SDA 线的电平检测来判断，即仲裁逻辑会检查每个 SDA 发送的逻辑 1 是否真正出现在总线上，如果有其它主机传输低电平并把 SDA 线拉低，则传输高电平的主机将失去仲裁，必须放弃总线，这就是基本的仲裁逻辑。

注：这是一种“低电平优先”的仲裁原则，即把总线判给在 SDA 上先发送低电平的主器件。

失去仲裁的主机将从发送状态转为接收状态，并且继续发送时钟脉冲（SCL 上），直到当前传输字节发送完毕。

注： 除了主机发送模式，在主机接收模式中，当主机向从机返回一个 NACK 信号（逻辑 1）时，也有可能发生仲裁丢失，由于 NACK 在串行字节结束时出现，因此模块不会再发送时钟脉冲。

仲裁是按 bit 逐位进行的，它的第一个阶段是比较地址位，如果多个主机寻址相同的器件，则可能会顺利通过地址阶段仲裁，则仲裁将会继续比较数据位（主机发送模式），或者响应位（主机接收模式）。因为总线上的地址和数据信息是由赢得仲裁的主机决定，仲裁过程中不会丢失信息。

如果该主机同时开启了从机模式，在发送地址阶段失去仲裁后应检测线上的地址是否与自己相匹配；如果是对自己的访问，应立即切换到从机模式，接收信息。

每次传输中，仍要检测线上的“重复起始条件”，当检测到并非自己发出的“重复起始条件”时，应立即退出当前传输。

当一个主机发送重复起始条件或停止条件时，另一个主机可能仍然在发送数据，这会造成一个未定的状态（规范要求在帧格式相同位置重发这个重复起始条件或停止条件），为此定义如下 3 种情况下总线不进行仲裁：

- 主机 1 发送重复起始条件，主机 2 发送数据
- 主机 1 发送停止条件，主机 2 发送数据
- 主机 1 发送重复起始条件，主机 2 发送停止条件

另外，仲裁只发生在主机之间，从机不参与仲裁。

图23-28 数据仲裁时序图

23.3.3.4 TWI 通讯模块的详细结构

下图描述了 TWI 通讯模块的详细结构。

图23-29 TWI 模块结构图

总线接口单元

总线接口单元包括数据和地址移位寄存器(TWI_DR)，开始/停止条件控制器，仲裁和总线超时检测单元。

寄存器 TWI_DR 存储了即将发送的数据或地址和接收到的数据和地址。

开始/停止条件控制器负责发送和检测总线上的起始条件，重复起始条件和停止条件。

如果 SH33F2801 已经作为主机开始一次传输，仲裁单元将始终检测是否有仲裁发生。当失去仲裁时，控制单元可以进行合适的动作，并产生相应的状态码。

SH33F2801 在传输数据/地址时，必须在 SCL 由低跳高前维持数据稳定。

SH33F2801 在传输 ACK/NACK 时，在 SCL 由低跳高后产生 TWINT 中断，并在 SCL 由高跳低时拉低 SCL，在 TWINT 中断清零时释放 SCL。

SH33F2801 在传输 ACK/NACK 信号时，若 TWINT 已被清零，SCL 仍为高电平时，SDA 产生跳变，则重新产生 TWINT 中断，状态为 00H。SH33F2801 当前通讯终止，该状态与普通 00H 状态处理一致。

SH33F2801 在传输 ACK/NACK 信号时，若 TWINT 未被清零，SCL 仍为高电平时，SDA 产生跳变，则状态直接切换到 00H，不会再次产生中断。SH33F2801 作为从机进入该状态，则当前通讯终止，可发生 STA 开始主机传输，或重新接受 STA+ADR 对自己地址的访问。SH33F2801 作为主机进入该状态，则当前通讯终止，可发生 STA 开始主机传输，或重新接受 STA+ADR 对自己的访问。

SH33F2801 在当前通讯终止后，不会再参与当前传输。SH33F2801 若作为主机存在，请开启 EFREE 功能，防止进入逻辑死区。

SH33F2801 规定总线维持高电平超过 TFREE =Tsys*TWI_HOC*1024（必须保证 TFREE 大于 t_{SCL}/2（t_{SCL} 为时钟线的周期））所定义的系统时钟个数时为“空闲”状态，释放总线（此处“系统时钟”Tsys 实际为 TWI 模块工作时钟）。该功能仅适用于一个数据包传输过程中（8+1 个位）。SH33F2801 处于从机发送模式，且所传输的第一个字节为低电平时适用该功能。起始条件（STA、RSTA）不适用于该功能。SH33F2801 产生中断，寄存器 TWI_CR 中的 TFREE 会被置位(如果控制位 EFREE 已置位)。

如果时钟线 SCL 被从机拉低时，通讯会暂时中止；而主机也没有办法将时钟线拉高。为解决此问题，TWI 协议规定参与传输的所有器件，将时钟线维持低电平超过 N*Tsys（N 值由 CNT 寄存器决定）所定义的时钟个数时为“总线超时”，寄存器 TWI_FR 中的 TOUT 会被置位(如果控制位 ETOT 已置位)。软件通过此标记位的变化，复位 TWI 模块，释放总线。

频率生成单元

在主机模式下，可通过寄存器 TWI_CR 的 CR[1:0]分频系数及 TWI_BRT[7:0]寄存器来设定通讯频率。

SCL 频率为 f_{TWI} /（16+2*CR*TWI_BRT）。

注：此处 f_{TWI} 为 TWI 模块工作时钟，即 APB1 总线时钟。

地址匹配单元

地址匹配单元检验所收到的地址是否与 TWIADR 中的 7 位地址相匹配，如果地址屏蔽位 TWIAMR 内对应位置 1，则该位地址不检测。如果通用地址使能位 GC 被置位，还将检测是否与通用地址 00H 相匹配。当地址匹配时，控制单元将产生合适的动作及相应的状态码。

控制单元

控制单元监视 TWI 总线，并依据控制寄存器 TWI_CR 的设置进行相应的回应。当 TWI 总线有需要应用层注意的事件时，TWI 中断标志被置起，标明当前事件的状态码会被写入状态寄存器 TWI_STAT。状态寄存器 TWI_STAT 只表示 TWI 通讯中断产生时的通讯状态信息；其它情况下状态寄存器内是一个用于表示没有有效状态码的状态码（F8H）。在中断清除之前，时钟线将维持低电平。应用软件可在处理完任务后才允许 TWI 通讯继续。

23.3.3.5 TWI 操作模式

在指定应用中，TWI 模块可用作主机(master)、从机(slave)或同时用作主机和从机。在从机模式中，TWI 硬件查找从机地址以及通用地址（GC=1）。如果检测到其中一个地址，则产生中断请求。如果处理器想成为总线主机，那么在进入主机模式前硬件必须等到总线空闲，使得从机操作不被中止。如果在主机模式中总线仲裁丢失，则 TWI 模块进入等待状态（纯主机），或者立刻切换到从机模式并在相同的串行传输中检测自身的从机地址（主从机）。

TWI 的四种基本操作模式：

（1）主机发送模式
（2）主机接收模式
（3）从机接收模式
（4）从机发送模式

SH33F2801 的 TWI 通讯是以底层驱动电路和以中断为基础的应用软件共同完成的。诸如接收到一个字节或发送一个起始条件的所有总线事件均会产生一个中断。所以在字节传输期间，应用软件可以进行其它的操作。需注意的是，除了基本的 TWINT 中断外，有时还需要开启总线超时和 SCL 高电平超时检测，这两种情况也会产生中断，三个中断源共享一个中断入口，具体见“中断”章节。

当 TWINT 位置起时，表示一次 TWI 传输已完成，等待应用软件的回应，此时状态寄存器 TWI_STAT 包含了当前的状态。应用软件可通过寄存器 TWI_CR 和 TWI_STAT 决定 TWI 进行哪种通讯。

下面将分别介绍 TWI 通讯的四种主要模式，并对所有可能的状态码进行了描述。下图中有如下缩写：

S ：起始条件
Rs ：重复起始条件
R ：读控制位
W ：写控制位
A ：应答位
$\bar{A}$ ：无应答位
DATA：8 位数据
P ：停止条件
SLA ：从机地址

图中圆形用于表示中断标志已被置起。其中的数字表示当前状态寄存器 TWI_STAT 中的状态码。

在 TWINT 被清除之前，TWI 通讯会暂停，应用软件必须决定是继续通讯还是终止当前传输。对每一个状态码，所需要的软件动作和随后的传输细节均有描述。

- **主机发送模式**

主机发送模式中，主机发送一系列数据到从机。为进入主机发送模式，一个起始条件，随后一个从机地址+写控制字(SLA+W)地址包表示进入主机发送模式(MT)。

通过设置控制寄存器 TWI_CR 中的 TWIEN 和 STA，清除 STO 和 TWINT，TWI 逻辑将检测 TWI 总线并在允许时发出一个起始条件(STA)。当起始条件(STA)传输完毕，通讯中断(TWINT)被置起，状态寄存器(TWI_STAT)为 08H，中断服务程序应将从机地址和写控制字(SLA+W)写入数据寄存器 TWI_DR。在开启下一个传输前清除 TWINT 标志。（注）

当从机地址和写控制字传输完毕并收到一个“应答”信息时，中断(TWINT)被置起，状态寄存器 TWI_STAT 中有几个可能的状态：对主机模式有 18H，20H 和 38H，对从机模式有 68H，78H 和 B0H。

注：TWINT 标志置位以后，数据可以写入控制寄存器，但总线上不会发送。 即使是 START 位也不会发送，因此涉及到重复起始条件的情况（此时可能 TWINT 保持置位状态），写入 START 位以后也要清除 TWINT 标志才能发送出去。

图23-30　　主机发送模式状态图

注：上图中设定SH33F2801是主机。

表23-5　　主机发送模式状态码

状态码	TWI总线和硬件接口状态	应用软件响应					TWI执行的下一个动作
		读/写数据寄存器TWI_DR操作	控制位操作				
			STA	STO	TWINT	AA	
08H	已发送起始条件	写入SLA+W	X	0	0	X	发送SLA+W，接收ACK 或NACK
10H	已发送重复起始条件	写入SLA +W	X	0	0	X	发送SLA+W，接收ACK或NACK
		写入SLA +R	X	0	0	X	发送SLA+R，TWI将切换到主机接收模式
18H	已发送SLA +W；已接收ACK	写入数据字节	0	0	0	X	发送数据，接收ACK或NACK
		无TWI_DR动作	1	0	0	X	发送重复起始条件
			0	1	0	X	发送停止条件；清除STO标志
			1	1	0	X	发送停止条件，之后发送起始条件；STO被清除
20H	已发送SLA +W；已接收NACK	写入数据字节	0	0	0	X	发送数据，接收ACK或NACK
		无TWI_DR动作	1	0	0	X	发送重复起始条件
			0	1	0	X	发送停止条件；清除STO标志
			1	1	0	X	发送停止条件，之后发送起始条件；STO被清除
28H	已发送TWI_DR中数据；已接收ACK	写入数据字节	0	0	0	X	发送数据，接收ACK或NACK
		无TWI_DR动作	1	0	0	X	发送重复起始条件
			0	1	0	X	发送停止条件；清除STO标志
			1	1	0	X	发送停止条件，之后发送起始条件；STO被清除
30H	已发送TWI_DR中数据；已接收NACK	写入数据字节	0	0	0	X	发送数据，接收ACK或NACK
		无TWI_DR动作	1	0	0	X	发送重复起始条件
			0	1	0	X	发送停止条件；清除STO标志
			1	1	0	X	发送停止条件，之后发送起始条件；STO被清除
38H	在SLA+W或数据传输中丢失仲裁位	无TWI_DR动作	0	0	0	X	TWI总线被释放；进入非寻址从机模式
			1	0	0	X	在总线空闲时发送起始条件

■ 主机接收模式

主机接收模式中，主机从从机接收一系列数据。为进入主机接收模式，一个起始条件，随后一个从机地址+读控制字(SLA+R)地址包表示进入主机接受模式(MR)。

通过设置控制寄存器 TWI_CR 中的 TWIEN 和 STA，清除 STO 和 TWINT，TWI 逻辑将检测 TWI 总线并在允许时发出一个起始条件(STA)。当起始条件(STA)传输完毕，通讯中断(TWINT)被置起，状态寄存器(TWI_STAT)为 08H，中断服务程序应将从机地址和读控制字(SLA+R)写入数据寄存器 TWI_DR。在开启下一个传输前清除 TWINT 标志。

当从机地址和写控制字传输完毕并收到一个“应答”信息时，中断(TWINT)被置起，状态寄存器 TWI_STAT 中有几个可能的状态：对主机模式有 40H，48H 和 38H，对从机模式有 68H，78H 和 B0H。

图23-31　　主机接收模式状态图

注：上图中设定SH33F2801是主机。

表23-6　　主机接收模式状态码

状态码	TWI总线和硬件接口状态	应用软件响应					TWI执行的下一个动作
		读/写数据寄存器TWI_DR操作	控制位操作				
			STA	STO	TWINT	AA	
08H	已发送起始条件	写入SLA+R	X	0	0	X	发送SLA+R，接收ACK或NACK
10H	已发送重复起始条件	写入SLA+R	X	0	0	X	发送SLA+R，接收ACK或NACK
		写入SLA+W	X	0	0	X	发送SLA+W，TWI将切换到主机发送模式
38H	发送SLA+R或NACK时失去仲裁	无TWI_DR动作	0	0	0	X	TWI总线被释放；进入非寻址从机模式
			1	0	0	X	在总线空闲时发送起始条件
40H	已发送SLA+R；已接收ACK	无TWI_DR动作	0	0	0	0	接收数据，返回NACK
			0	0	0	1	接收数据，返回ACK
48H	已发送SLA+R；已接收NACK	无TWI_DR动作	1	0	0	X	发送重复起始条件
			0	1	0	X	发送停止条件；清除STO标志
			1	1	0	X	发送停止条件，之后发送起始条件；STO被清除
50H	数据已接收；已回应ACK	读取数据	0	0	0	0	接收数据，返回NACK
			0	0	0	1	接收数据，返回ACK
58H	数据已接收；已回应NACK	读取数据	1	0	0	X	发送重复起始条件
			0	1	0	X	发送停止条件；清除STO标志
			1	1	0	X	发送停止条件，之后发送起始条件；STO被清除

■ 从机发送模式

从机发送模式中，从机发送一系列数据到主机。为初始化从机发送模式，必须对控制寄存器 TWI_CR 和地址寄存器 TWI_ADDR 进行初始化：置位控制寄存器 TWI_CR 中的 TWIEN 和 AA，清除 STA、STO 和 TWINT；地址位 TWI_ADDR 为 SH33F2801 准备相应的地址。如果 GC 置位，SH33F2801 也将响应通用地址(00H)；否则将不响应通用地址。

在 TWIADR 和 TWI_CR 初始化后，SH33F2801 将等待总线对自己地址或通用地址(如果 GC 被置位)的响应。如果方向标志位是“读”，则 TWI 进入从机发送模式，否则将进入从机接收模式。在地址和读标志位接收完毕后，中断标志(TWINT)置位，状态寄存器 TWI_STAT 有效。

在传输中，如果将应答使能位“AA”清零，TWI 将传送最后一个字节，并依据主机接收方发送的应答或不应答信息位进入 C0H 或 C8H 状态。总线将切换到非地址从机模式，不在响应主机传输。从而主机接收方将接收到一串’1’。最后一个字节发送完毕后，如果主机仍需额外的数据(传输“应答”信号)，则进入 C8H 状态。

图23-32　　从机发送模式状态图

注：上图中设定SH33F2801是从机。

表23-7　　从机发送模式状态码

状态码	TWI总线和硬件接口状态	应用软件响应					TWI执行的下一个动作
		读/写数据寄存器TWI_DR操作	控制位操作				
			STA	STO	TWINT	AA	
A8H	已收到自己SLA+R；已经回应ACK	写入数据字节	X	0	0	0	发送最后数据；等待ACK或NACK回应
			X	0	0	1	发送数据；等待ACK或NACK回应
B0H	作为主机发送SLA+R/W时失去仲裁，收到主机SLA+R；已回应ACK	写入数据字节	X	0	0	0	发送最后数据；等待ACK或NACK回应
			X	0	0	1	发送数据；等待ACK或NACK回应
B8H	已发送TWI_DR数据；已接收ACK回应	写入数据字节	X	0	0	0	发送最后数据；等待ACK或NACK回应
			X	0	0	1	发送数据；等待ACK或NACK回应
C0H	已发送TWI_DR数据；已接收NACK回应	无TWI_DR动作	0	0	0	0	切换至非寻址从机模式；不响应自己地址和通用地址
			0	0	0	1	切换至非寻址从机模式；响应自己地址，是否响应通用地址依赖于寄存器TWI_ADDR中GC的设置
			1	0	0	0	切换至非寻址从机模式；不响应自己地址和通用地址；总线空闲时发送“起始条件”
			1	0	0	1	切换至非寻址从机模式；响应自己地址，是否响应通用地址依赖于寄存器TWI_ADDR中GC的设置；总线空闲时发送“起始条件”
C8H	已发送最后一个TWI_DR数据(AA=0)；已接收ACK回应	无TWI_DR动作	0	0	0	0	切换至非寻址从机模式；不响应自己地址和通用地址
			0	0	0	1	切换至非寻址从机模式；响应自己地址，是否响应通用地址依赖于寄存器TWI_ADDR中GC的设置
			1	0	0	0	切换至非寻址从机模式；不响应自己地址和通用地址；总线空闲时发送“起始条件”
			1	0	0	1	切换至非寻址从机模式；响应自己地址，是否响应通用地址依赖于寄存器TWI_ADDR中GC的设置；总线空闲时发送“起始条件”

■ 从机接收模式

从机接收模式中，从机从主机接收一系列数据。为初始化从机接收模式，必须对控制寄存器 TWI_CR 和地址寄存器 TWI_ADDR 进行初始化：置位控制寄存器 TWI_CR 中的 TWIEN 和 AA，清除 STA、STO 和 TWINT；地址寄存器 TWIADR 中高 7 位为 SH33F2801 准备相应的地址。如果 GC 置位，SH33F2801 也将响应通用地址(00H)；否则将不响应通用地址。

在 TWI_ADDR 和 TWI_CR 初始化后，SH33F2801 将等待总线对自己地址或通用地址(如果 GC 被置位)的响应。如果方向标志位是'写'，则 TWI 进入从机接收模式，否则将进入从机发送模式。在地址和写标志位接收完毕后，中断标志(TWINT)置位，状态寄存器 TWI_STAT 有效。

在传输中，如果将应答使能位"AA"清零，TWI 将接收最后一个字节并回应"不应答"信息。回应"不应答"可以表示当前从机无法接收更多字节。当 AA=0 时，SH33F2801 无法回应对自己地址的访问；但仍然监视总线状态，并可以通过 AA=1 恢复对自己地址的响应。可以通过 AA=0 暂时将 SH33F2801 从总线隔离。

图23-33　　从机接收模式图

注：上图中设定SH33F2801是从机。

表23-8　　从机接收模式状态码

状态码	TWI总线和硬件接口状态	应用软件响应					TWI执行的下一个动作
		读/写数据寄存器TWI_DR操作	控制位操作				
			STA	STO	TWINT	AA	
60H	已收到自己SLA+W；已回应ACK	无TWI_DR动作	X	0	0	0	接收数据；发送NACK回应
			X	0	0	1	接收数据；发送ACK回应
68H	作为主机发送SLA+R/W时失去仲裁，收到主机SLA+W；已回应ACK	无TWI_DR动作	X	0	0	0	接收数据；发送NACK回应
			X	0	0	1	接收数据；发送ACK回应
70H	收到主机发送通用地址；已回应 ACK	无TWI_DR动作	X	0	0	0	接收数据；发送NACK回应
			X	0	0	1	接收数据；发送ACK回应
78H	作为主机发送SLA+R/W时失去仲裁，收到主机发送通用地址；已回应ACK	无TWI_DR动作	X	0	0	0	接收数据；发送NACK回应
			X	0	0	1	接收数据；发送ACK回应
80H	处于已寻址状态；已收到数据；已回应ACK	读取数据	X	0	0	0	接收数据；发送NACK回应
			X	0	0	1	接收数据；发送ACK回应
88H	处于已寻址状态；已收到数据；已回应NACK	读取数据	0	0	0	0	切换至非寻址从机模式；不响应自己地址和通用地址
			0	0	0	1	切换至非寻址从机模式；响应自己地址，是否响应通用地址依赖于寄存器TWI_ADDR中GC的设置
			1	0	0	0	切换至非寻址从机模式；不响应自己地址和通用地址；总线空闲时发送“起始条件”
			1	0	0	1	切换至非寻址从机模式；响应自己地址，是否响应通用地址依赖于寄存器TWI_ADDR中GC的设置；总线空闲时发送“起始条件”
90H	处于通用地址已寻址状态；已收到数据；已回应ACK	读取数据	X	0	0	0	接收数据；发送NACK回应
			X	0	0	1	接收数据；发送ACK回应
98H	处于通用地址已寻址状态；已收到数据；已回应NACK	读取数据	0	0	0	0	切换至非寻址从机模式；不响应自己地址和通用地址
			0	0	0	1	切换至非寻址从机模式；响应自己地址，是否响应通用地址依赖于寄存器TWI_ADDR中GC的设置
			1	0	0	0	切换至非寻址从机模式；不响应自己地址和通用地址；总线空闲时发送“起始条件”
			1	0	0	1	切换至非寻址从机模式；响应自己地址，是否响应通用地址依赖于寄存器TWI_ADDR中GC的设置；总线空闲时发送“起始条件”
A0H	作为从机时收到停止条件或重复起始条件	无TWI_DR动作	0	0	0	0	切换至非寻址从机模式；不响应自己地址和通用地址
			0	0	0	1	切换至非寻址从机模式；响应自己地址，是否响应通用地址依赖于寄存器TWI_ADDR中GC的设置
			1	0	0	0	切换至非寻址从机模式；不响应自己地址和通用地址；总线空闲时发送“起始条件”
			1	0	0	1	切换至非寻址从机模式；响应自己地址，是否响应通用地址依赖于寄存器TWI_ADDR中GC的设置；总线空闲时发送“起始条件”

■ 其它模式

除上述状态码外，两个状态码没有明确的 TWI 状态。状态 0F8H 表示由于中断标志 TWINT 未置起，没有相应的状态信息。当中断 TWINT 未置起，即在清除一个状态后到一个新状态建立前均由 0F8H 进行填充。

状态 00H 表示在 TWI 总线通讯中有错误发生，即传输中有非法的起始条件或停止条件发生。例如在传输地址，数据或回应 ACK 应答时有起始条件或停止条件发生。当总线扰乱了内部逻辑时也会产生 00H 状态。当非法状态出现时，会置起中断标志位 TWINT。可通过置起 STO 并清除 TWINT 标志恢复到正常通讯，SH33F2801 将进入非寻址从机模式，并自动清除 STO 标志。数据线和时钟线将被释放，线上无停止条件传送。

表23-9　　其它模式状态码

状态码	TWI总线和硬件接口状态	应用软件响应					TWI执行的下一个动作
		读/写数据寄存器 TWI_DR操作	控制位操作				
			STA	STO	TWINT	AA	
F8H	没有有效状态码；TWINT=0	无TWI_DR动作	无TWI_CR动作				等待或处理当前传输
00H	在主机或寻址从机模式下有非法起始条件或停止条件发送；接口导致TWI内部逻辑混乱	无TWI_DR动作	0	1	0	X	只有内部硬件受影响；释放总线；切换到非寻址从机模式；清除STO

23.3.3.6　支持快速模式 400kbps

TWI 模块兼容两档常用的传输速率：标准模式 100kHz、快速模式 400kHz。通过应用软件配置分频系数 CR[1:0]和比特率配置位 BRT[7:0]来获得两种传输速率。

23.3.3.7　中断

总线超时标志TOUT，SCL高电平超时标志TFREE，TWI通讯中断标志TWINT，三者任一标志置位，都会引起TWI中断(前提是对应的中断使能置位)，三者共享中断向量，标志必须由软件清0。

TOUT：总线超时标志。总线超时功能在TWI模块开启时即使能（无法关闭），一旦发生总线超时就置位TOUT，而中断使能位ETOT则控制总线超时能否引起TWI中断。

注：当TWI总线低电平超过N*Ttwi时TOUT置位。N值由CNT[1:0]@TWI_CR寄存器位决定。

TFREE：SCL高电平超时标志。SCL高电平超时功能在TWI模块开启时即使能（无法关闭），一旦发生SCL高电平超时就置位TFREE，而中断使能位EFREE则控制SCL高电平超时能否引起TWI中断。

注：参与总线传输时，如时钟线高电平超过TFREE=Ttwi* HOC *256所定义的系统时钟个数时硬件置位。HOC值由HOC[7:0]@TWI_HOC寄存器位决定。

TWINT：TWI通讯中断标志。当TWI模块开启时即使能。当TWI状态改变时TWINT置位。但是，进入状态F8不会使TWINT置位，因为在那种情况下中断服务程序不起作用。

当TWINT置位时，SCL线上的串行时钟低电平周期扩展，且串行传输被中止。当SCL为高时，TWINT标志的状态不受影响。

TWINTE：TWI通讯中断使能位，控制TWINT标志是否产生中断请求，如果关闭该中断使能位，则应用软件只能通过查询方式进行TWI通信状态处理。

注：TWI没有总中断控制位，中断使能由ETOT、EFREE、TWINTE三个独立开关控制。

注意：因为总线超时和SCL高电平超时功能强制开启，用户设计TWI协议时必须考虑超时因素，在合理配置超时时间后，在一次TWI通信期间，尽量不要插入长时间的等待以免误触发超时。

23.3.3.8　TWI 通信服务程序开发

SH33F2801 的 TWI 模块需配合用户应用程序才能实现完整的针对 TWI 的通信协议（根据应用的不同通信协议都有不同），保证了最大的应用灵活性。用户应用程序通过 TWI 中断进行驱动，不会影响主流程和其它控制。具体可参考四种主要模式的状态图和状态码进行开发。对于 MCU 最基本的操作 TWI/I2C 接口外设的情况，只需要参考“主机发送模式”进行开发即可，

一个完整的 TWI 服务程序必须执行的操作包括：

复位后 TWI 模块初始化，包括配置从机地址和通用地址位（GC），使能中断，设置串行时钟频率等。

TWI 中断服务，识别状态代码并按分支进入相应的状态服务程序，另外也需要处理超时情况来限制无效的总线或丢失的服务程序。

支持 4 种 TWI 操作模式的多种状态服务程序（I2C 有 26 种），有些模式有一种或几种没用到则相应的状态服务可忽略（只要小心处理，那些状态就不会出现）。

23.3.4 TWI 寄存器

TWI 模块寄存器列表 (基地址：0x4000 2800)

地址	寄存器名	说明
0x4000 2800	FR	TWI 中断标志寄存器
0x4000 2804	STAT	TWI 状态寄存器
0x4000 2808	HOC	TWI 高电平超时检测配置寄存器
0x4000 280C	BRT	TWI 比特率配置寄存器
0x4000 2810	DR	TWI 数据寄存器
0x4000 2814	ADDR	TWI 地址配置寄存器
0x4000 2818	CR	TWI 控制寄存器

23.3.4.1 TWI 中断标志寄存器 (TWI_FR)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留												TOUTC	TFREEC	保留	TWINTC
-												*WO*	*WO*	-	*WO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留												TOUT	TFREE	保留	TWINT
-												*RO*	*RO*	-	*RO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 20	保留	-
19	**TOUTC**	**总线超时标志清除位** 0：无效 1：清除
18	**TFREEC**	**SCL 高电平超时标志清除位** 0：无效 1：清除
17	保留	-
16	**TWINTC**	**TWI 一字节数据传送中断标志清除位** 0：无效 1：清除
15 ~ 4	保留	-
3	**TOUT**	**总线超时标志位** 0：无超时发生，由软件写 1 清零 1：当 TWI 总线低电平超过 N*Tsys 时置位。N 值由 TWITOUT 寄存器决定
2	**TFREE**	**SCL 高电平超时标志位** 0：无超时，由软件写 1 清零 1：参与总线传输时，如时钟线高电平超过 TFREE=Tsys* TWI_HOC *256 所定义的系统时钟个数时硬件置位
1	保留	-
0	**TWINT**	**TWI 一字节数据传送中断标志位** 0：单字节数据传送中，由软件写 1 清零 1：单字节数据传送完毕，由硬件置位

23.3.4.2 TWI 状态寄存器 (TWI_STAT)

偏移地址: 0x0004

复位值: 0x0000 00F8

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								SR[4:0]					保留		
-								RO					-		
0	0	0	0	0	0	0	0	1	1	1	1	1	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 3	**SR[4:0]**	**TWI 串行通讯状态位**
2 ~ 0	保留	-

23.3.4.3 TWI 高电平超时检测配置寄存器 (TWI_HOC)

偏移地址: 0x0008

复位值: 0x0000 00FF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								HOC[7:0]							
-								RW							
0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1

位	符号	说明
31 ~ 8	保留	-
7 ~ 0	**HOC[7:0]**	**TWI 的 SCL 总线高电平超时检测配置位**

23.3.4.4 TWI 比特率配置寄存器 (TWI_BRT)

偏移地址: 0x000C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								BRT[7:0]							
-								RW							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 0	**BRT[7:0]**	**TWI 的比特率配置位**

23.3.4.5 TWI 数据寄存器 (TWI_DR)

偏移地址: 0x0010
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								DR[7:0]							
-								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 0	**DR[7:0]**	**TWI 数据寄存器**

23.3.4.6 TWI 地址配置寄存器 (TWI_ADDR)

偏移地址: 0x0014
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留								TWIAMR[6:0]							保留
-								*RW*							-
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								ADDR[6:0]							GC
-								*RW*							*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 24	保留	-
23 ~ 17	**TWIAMR[6:0]**	**TWI 地址屏蔽位** TWIAMR 的每一位可屏蔽 TWIADR 地址寄存器中相应的地址位。如果屏蔽位置 1，地址匹配逻辑忽略输入的地址位与 TWIADR 中的相应地址位的比较结果；如果屏蔽位置 0，不忽略相应位的比较结果
16 ~ 8	保留	-
7 ~ 1	**ADDR[6:0]**	**TWI 地址配置位**
0	**GC**	**通用地址使能位** 0：禁止响应通用地址 1：允许响应通用地址

23.3.4.7 TWI 控制寄存器 (TWI_CR)

偏移地址: 0x0018

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TWI EN	保留				TWI NTE	ETO T	EFR EE	CNT[1:0]		CR[1:0]		保留	STA	STO	AA
RW	-				RW	RW	RW	RW		RW		-	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15	**TWIEN**	**TWI 使能位** 0：关闭 TWI 1：使能 TWI
14 ~ 11	保留	-
10	**TWINTE**	**TWI 的发送接收中断标志使能位** 0：禁止 TWI 中断标志位 TWINT 触发中断 1：允许 TWI 中断标志位 TWINT 触发中断
9	**ETOT**	**总线超时监控使能位** 0：禁止总线超时判断 1：使能总线超时判断
8	**EFREE**	**SCL 高电平超时监控使能位** 0：禁止 SCL 总线高电平超时判断 1：使能 SCL 总线高电平超时判断
7 ~ 6	**CNT[1:0]**	**总线超时计数控制位** 00: N=25000 01: N=50000 10: N=100000 11: N=200000 N 值在总线超时功能中使用
5 ~ 4	**CR[1:0]**	**TWI 分频系数** 00：64 01：16 10： 4 11： 1
3	保留	-
2	**STA**	**起始位** 0：不发送起始条件 1：当总线空闲时发送起始条件 注：当 STA=1 时发送起始条件并进入主模式，如果已经在主模式，则发送重复起始条件 当 STA 和 STO 都置位时，如果处于主模式，将向总线发送一个停止条件，然后发送一个起始条件。如果接口处于从模式，则产生一个内部停止条件，但不发送到总线上。

1	**STO**	**终止位** 0：不发送终止条件 1：作为主机时发送终止条件；作为从机时不向总线发送终止条件，但状态恢复到非寻址从机状态。硬件将自动清除该标志位 注：在主模式中，当 STO 为 1 时，会发送一个停止条件或在从模式中从错误状态中恢复。当主模式中 STO=1 时，向总线发送停止条件。当总线检测到停止条件时，STO 自动清零。在从模式中，置位 STO 位可从错误状态中恢复。这种情况下不向总线发送停止条件。硬件的表现就好像是接收到一个停止条件并切换到不可寻址的从接收模式。STO 标志由硬件自动清零。
0	**AA**	**声明应答标志位** 0：回复“不应答”信号（SDA 高电平） 1：回复“应答”信号（SDA 低电平）

23.4 FIFO 控制器

23.4.1 简介

FIFO控制器用以匹配高速主系统与低速外围通信接口之间的速度差，通过FIFO的缓冲，可以减少CPU被打断的频率，确保流畅的通信数据流，提升系统整体效率。

对于双工外设（全双工或半双工），FIFO都是成组出现的，每组FIFO包含TX FIFO和RX FIFO两个完全独立的FIFO模块。

SH33F2801有3组FIFO，都是双工外设，具体资源为：

FIFO0：4级深度FIFO，TXFIFO 4×8 bit，RXFIFO 4×（8+3）bit

FIFO1：4级深度FIFO，TXFIFO 4×8 bit，RXFIFO 4×（8+3）bit

FIFO2：8级深度FIFO，TXFIFO 8×8 bit，RXFIFO 8×（8+3）bit

23.4.2 主要特性

- TX FIFO 位宽 8 bit(模块定义 8/16/32 存取位宽对应于 FIFO counter±1/±2/±4)
- RX FIFO 位宽支持 8 bit/(8+N)bit，(8+N)bit 只针对 UART 模块
- 可配置的 FIFO depth (2N，N≥2)，且 RX FIFO 和 TX FIFO 可分别配置
- FIFO 能提供空,非空,满标志
- 提供可编程的 triger level:1/4,1/2,3/4
- 模块根据编程值触发条件后输出信号
- 当发生 overflow 时，丢弃 overflow 数据，提供 overflow 信号
- 提供 flush 操作，复位读写指针
- 提供数据超时信号

23.4.3 功能描述

23.4.3.1 Enable 和 Reset 信号

FIFO有单独的时钟使能与模块信号，前者在RCC中定义，后者在此处定义。时钟使能后即可对FIFO寄存器进行操作，但只有模块使能FIFO才开始工作。

FIFO内部Reset信号用于清零FIFO内数据、清零Counter值和内部指针， 但对FIFO配置寄存器和flag信号无影响。

Control			
符号	类型	复位	说明
TXFRST	RW1s（注）	0	写1 reset TX FIFO，复位完成自动清零
RXFRST	RW1s	0	写1 reset RX FIFO，复位完成自动清零
TXFEN	W/R	0	Enable TX FIFO 0：Enable 1：Disable
RXFEN	W/R	0	Enable RX FIFO 0：Enable 1：Disable

注：RW1s 为可读，写 1 置 1，硬件清零。

FIFO每开关一次，都自动执行一次Reset操作。

23.4.3.2 FIFO 映射

FIFO与外设之间的映射是可配置的，每个FIFO可映射到多个外设中的任意一个。但硬件上没有做多个FIFO映射到同一个外设的错误检查，需要软件上避免这种情况。

FIFO/外设 映射选择			
符号	类型	复位	说明
TXMAP	**W/R**	**0**	**TX FIFO与外设映射选择** 00：UART0 01：UART1 10：UART2 11：保留 或 SPI（注） 注：4级FIFO不支持SPI，保留 8级FIFO支持SPI。
RXMAP	**W/R**	**0**	**RX FIFO与外设映射选择** 00：UART0 01：UART1 10：UART2 11：保留 或 SPI（注） 注：4级FIFO不支持SPI， 8级FIFO支持SPI。

注意某个外设要配置FIFO功能需执行以下步骤：

（1） 开启FIFO时钟，使能FIFO，完成FIFO配置；

（2） 完成FIFO与外设的映射；

（3） 在外设模块中开启FIFO功能，开始使用FIFO；

23.4.3.3 Trigger Level 定义

FIFO Counter

- Reset 后为 0
- 当数据写入并且没有读操作，counter+1
- 当数据被读出并且没有写入操作，counter-1

Trigger信号

- Trigger Level 可配置为 1/4 ,1/2 ,3/4
- RX FIFO：当 counter 值累加等于被编程的 Trigger Level，产生 Trigger 信号
- TX FIFO：当 counter 值递减等于被编程的 Trigger Level，产生 Trigger 信号

Trigger Level			
符号	类型	复位	说明
RXTRIG	R/W	0	The trigger points for RX FIFO 00：RX FIFO >= 1/4 full 01：RX FIFO >= 1/2 full 10：RX FIFO >= 3/4 full 11：RX FIFO 有一笔数据
TXTRIG	R/W	0	The trigger points for TX FIFO 00：TX FIFO <= 1/4 full 01：TX FIFO <= 1/2 full 10：TX FIFO <= 3/4 full 11：TX FIFO empty

23.4.3.4 FIFO 在 CPU 一侧的读写接口

此处定义的FIFO模块数据流是单向的，即对TX FIFO而言，数据从CPU写入到FIFO中，外设从FIFO中取数据。对RX FIFO而言，数据从外设写入到FIFO中，CPU从FIFO中读数据。

对一个FIFO（无论TX 或 RX），其与外设之间的接口是确定的（一般为8位，也有16位情况）。而与CPU之间的接口位宽是可配置的，可以8/16/32位配置。CPU读写FIFO的数据接口是一个固定地址。

Write/Read BitWidth			
符号	类型	复位	说明
RXRDBITW	**W/R**	**0**	**RX FIFO读取数据的位宽(注)** **0：8 bits　　1：16 bits** **2：32bits　　3：Reserved**
TXWRBITW	**W/R**	**0**	**TX FIFO写入数据的位宽** **0：8 bits　　1：16 bits** **2：32bits　　3：Reserved**

注：对于8+N配置，要想读出后面的N bit，需要配置为 16 bits读取，其中高位补零。

23.4.3.5 FIFO 与外设的映射

FIFO的另一侧就是通信外设，FIFO与外设并不绑定，用户可进行配置。一旦配置完成，FIFO就替换掉外设原来的收发缓冲区（如果有的话），但是也仅仅影响数据接口，外设的底层收发控制还是由外设自身完成， FIFO只是在读写数据时参与而已。具体FIFO如何结合外设进行控制，请参考带FIFO功能的外设模块相关章节。

外设与FIFO的映射配置在外设模块中操作。一旦某个FIFO在外设中被映射，软件需要确保该FIFO时钟和使能已经被打开，硬件对此不做检查。

23.4.3.6 FIFO Status 描述

- 当 FIFO counter 值等于 FIFO 深度时，提供满 FULL 标志
- 当 FIFO counter 值等于 0 时，提供空 EMPTY 标志
- 当 FIFO counter 值不等于 0 时，提供非空 NON-EMPTY 标志

Status			
符号	类型	复位	说明
TXEMPTY	R	0	当TX FIFO空时，TXEMPTY =1 当TX FIFO内有数据时，RXEMPTY =0
TXFULL	R	0	当TX FIFO 数据满时，TXFULL=1
RXEMPTY	R	0	当RX FIFO空时，RXEMPTY =1 当RX FIFO内有数据时，RXEMPTY =0
RXFULL	R	0	当RX FIFO 数据满时，RXFULL=1

23.4.3.7 FIFO 控制过程

对TX FIFO，MCU先填充TX FIFO，同时外设从TX FIFO中取数，当TX FIFO Counter逐步减小，并小于指定的Trigger Level时，将置起FIFO触发标志位（TXTRIGF）。

TX FIFO写入位宽可选8/16/32位，当选择8位宽写入，FIFO Counter每次加1，16位宽写入每次加2，32位宽写入每次加4，即TX FIFO Counter的基本计数单位是字节。

举例：TX FIFO 深度为4 bytes，Trigger Level配置为1/2，设置TX FIFO写入位宽是8bits，在MCU写满TX FIFO后不再操作，外设每取走一个字节，TX FIFO Counter减1，当TX FIFO Counter减至2时(1/2 FIFO Depth)，硬件将置位TXTRIGF，并发出中断（如果使能），提醒MCU TX FIFO已经半空，需要进一步向TX FIFO中补充数据。

对RX FIFO，外设向RX FIFO填充，RX FIFO Counter逐步增大，当大于指定的Trigger Level时，将置起FIFO接收触发标志位（RXTRIGF）。

RX FIFO读出位宽可选8/16/32位，当选择8位宽读出，FIFO Counter每次减1，16位宽读出每次减2，32位宽读出每次减4，即RX FIFO Counter的基本计数单位是字节。

举例：RX FIFO深度为4 bytes， Trigger Level配置为1/2，设置RX FIFO读出位宽是8bits ，则RX模块开始填充RX FIFO，每填充一笔数据，RX FIFO Counter加1，当RX FIFO Counter增至2时(1/2 FIFO Depth)，硬件将置位RXTRIGF，并发出中断（如果使能），提醒RX FIFO已经半满，需要MCU从RX FIFO中读取数据。

23.4.3.8 FIFO 中断

每组FIFO有3个中断源，共享一个中断入口。每个中断源都有独立的使能位。中断标志位使用专门的中断清除控制位进行清除。

IRQ			
符号	类型	复位	说明
OVF	R	0	RX FIFO内数据已满，又收到新的数据时，产生此flag，同时丢弃新的数据，FIFO内数据不变
RXTRIGF	R	0	RX FIFO counter累加至trigger point时，产生此flag
TXTRIGF	R	0	TX FIFO counter减少至trigger point时，产生此flag

23.4.4 FIFO4L 寄存器

FIFO4L 模块寄存器列表 (基地址：0x4000 4400)

地址	寄存器名	说明
0x4000 4400	FR	FIFO中断和状态寄存器
0x4000 4404	CR	FIFO控制寄存器

23.4.4.1 FIFO 中断和状态寄存器 (FIFOx_FR)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留													RXOVFC	TXTRIGFC	RXTRIGFC
-													*WO*	*WO*	*WO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								TXFULL	RXFULL	TXEMPT	RXEMPT	保留	RXOVF	TXTRIGF	RXTRIGF
-								*RO*	*RO*	*RO*	*RO*	-	*RO*	*RO*	*RO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 19	保留	-
18	**RXOVFC**	**RX FIFO内数据满冲突标志清零位**
17	**TXTRIGFC**	**TX FIFO产生计数触发标志清零位**
16	**RXTRIGFC**	**RX FIFO产生计数触发标志清零位**
15 ~ 8	保留	-
7	**TXFULL**	**TX FIFO满状态标志**
6	**RXFULL**	**RX FIFO满状态标志**
5	**TXEMPT**	**TX FIFO空状态标志**
4	**RXEMPT**	**RX FIFO空状态标志**
3	保留	-
2	**RXOVF**	**RX FIFO内数据满冲突标志**
1	**TXTRIGF**	**TX FIFO产生计数触发标志**
0	**RXTRIGF**	**RX FIFO产生计数触发标志**

23.4.4.2 FIFO 控制寄存器 (FIFOx_CR)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留												TXFEN	RXFEN	TXFRST	RXFRST
-												RW	RW	RW1S	RW1S
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TXTRIG[1:0]		RXTRIG[1:0]		TXWRBITW[1:0]		RXRDBITW[1:0]		TXMAP[1:0]		RXMAP[1:0]		RXMODE	RXOVFIE	TXTRIGIE	RXTRIGIE
RW		RW		RW		RW		RW		RW		RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 20	保留	-
19	**TXFEN**	**TX FIFO使能位** 0: 禁止 1: 使能
18	**RXFEN**	**RX FIFO使能位** 0: 禁止 1: 使能
17	**TXFRST**	**TX FIFO复位控制** 写1 reset TX FIFO，复位完成自动清零
16	**RXFRST**	**RX FIFO复位控制** 写1 reset RX FIFO，复位完成自动清零
15 ~ 14	**TXTRIG[1:0]**	**TX FIFO触发点配置** The trigger points for TX FIFO 00：TX FIFO <= 1/4 full 01：TX FIFO <= 1/2 full 10：TX FIFO <= 3/4 full 11：TX FIFO empty
13 ~ 12	**RXTRIG[1:0]**	**RX FIFO触发点配置** 00：RX FIFO >= 1/4 full 01：RX FIFO >= 1/2 full 10：RX FIFO >= 3/4 full 11：RX FIFO 有一笔数据
11 ~ 10	**TXWRBITW[1:0]**	**TX FIFO写入数据的位宽** 0：8 bits 1：16 bits 2：32bits 3：Reserved
9 ~ 8	**RXRDBITW[1:0]**	**RX FIFO读出数据的位宽** 0：8 bits 1：16 bits 2：32bits 3：Reserved
7 ~ 6	**TXMAP[1:0]**	**TX FIFO与外设映射选择** 00：UART0 01：UART1 10：UART2

		11：保留 （或SPI） 注：4级FIFO不支持SPI，8级FIFO支持SPI
5 ~ 4	**RXMAP[1:0]**	**RX FIFO与外设映射选择** 00：UART0 01：UART1 10：UART2 11：保留（或SPI） 注：4级FIFO不支持SPI，8级FIFO支持SPI
3	**RXMODE**	**RX FIFO MODE选择位** 0: 8bits Mode 1：(8+N)bits Mode 注：此位只在UART外设下有意义，在SPI外设下无意义
2	**RXOVFIE**	**RX FIFO内数据满冲突中断使能位** 0: 禁止 1: 使能
1	**TXTRIGIE**	**TX FIFO产生计数触发中断使能位** 0: 禁止 1: 使能
0	**RXTRIGIE**	**RX FIFO产生计数触发中断使能位** 0: 禁止 1: 使能

23.4.5 FIFO8L 寄存器

FIFO8L 模块寄存器列表 (基地址：0x4000 4C00)

地址	寄存器名	说明
0x4000 4C00	FR	FIFO中断和状态寄存器
0x4000 4C04	CR	FIFO控制寄存器

23.4.5.1 FIFO 中断和状态寄存器 (FIFO8L_FR)

偏移地址: 0x0000

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留													RXOVFC	TXTRIGFC	RXTRIGFC
-													*WO*	*WO*	*WO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								TXFULL	RXFULL	TXEMPT	RXEMPT	保留	RXOVF	TXTRIGF	RXTRIGF
-								*RO*	*RO*	*RO*	*RO*	-	*RO*	*RO*	*RO*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 19	保留	-
18	**RXOVFC**	**RX FIFO内数据满冲突标志清零位**
17	**TXTRIGFC**	**TX FIFO产生计数触发标志清零位**
16	**RXTRIGFC**	**RX FIFO产生计数触发标志清零位**
15 ~ 8	保留	-
7	**TXFULL**	**TX FIFO满状态标志**
6	**RXFULL**	**RX FIFO满状态标志**
5	**TXEMPT**	**TX FIFO空状态标志**
4	**RXEMPT**	**RX FIFO空状态标志**
3	保留	-
2	**RXOVF**	**RX FIFO内数据满冲突标志**
1	**TXTRIGF**	**TX FIFO产生计数触发标志**
0	**RXTRIGF**	**RX FIFO产生计数触发标志**

23.4.5.2 FIFO 控制寄存器 (FIFO8L_CR)

偏移地址: 0x0004

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留												TXFEN	RXFEN	TXFRST	RXFRST
-												RW	RW	RW1S	RW1S
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TXTRIG[1:0]		RXTRIG[1:0]		TXWRBITW[1:0]		RXRDBITW[1:0]		TXMAP[1:0]		RXMAP[1:0]		RXMODE	RXOVFIE	TXTRIGIE	RXTRIGIE
RW		RW		RW		RW		RW		RW		RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 20	保留	-
19	**TXFEN**	**TX FIFO使能位** 0: 禁止 1: 使能
18	**RXFEN**	**RX FIFO使能位** 0: 禁止 1: 使能
17	**TXFRST**	**TX FIFO复位控制** 写1 reset TX FIFO，复位完成自动清零
16	**RXFRST**	**RX FIFO复位控制** 写1 reset RX FIFO，复位完成自动清零
15 ~ 14	**TXTRIG[1:0]**	**TX FIFO触发点配置** The trigger points for TX FIFO 00：TX FIFO <= 1/4 full 01：TX FIFO <= 1/2 full 10：TX FIFO <= 3/4 full 11：TX FIFO empty
13 ~ 12	**RXTRIG[1:0]**	**RX FIFO触发点配置** 00：RX FIFO >= 1/4 full 01：RX FIFO >= 1/2 full 10：RX FIFO >= 3/4 full 11：RX FIFO 有一笔数据
11 ~ 10	**TXWRBITW[1:0]**	**TX FIFO写入数据的位宽** 0：8 bits 1：16 bits 2：32bits 3：Reserved
9 ~ 8	**RXRDBITW[1:0]**	**RX FIFO读出数据的位宽** 0：8 bits 1：16 bits 2：32bits 3：Reserved

7 ~ 6	**TXMAP[1:0]**	**TX FIFO与外设映射选择** 00：UART0 01：UART1 10：UART2 11：SPI 注：8级FIFO支持SPI
5 ~ 4	**RXMAP[1:0]**	**RX FIFO与外设映射选择** 00：UART0 01：UART1 10：UART2 11：SPI
3	**RXMODE**	**RX FIFO MODE选择位** 0: 8bits Mode 1：(8+N)bits Mode 注：此位只在UART外设下有意义，在SPI外设下无意义
2	**RXOVFIE**	**RX FIFO内数据满冲突中断使能位** 0: 禁止 1: 使能
1	**TXTRIGIE**	**TX FIFO产生计数触发中断使能位** 0: 禁止 1: 使能
0	**RXTRIGIE**	**RX FIFO产生计数触发中断使能位** 0: 禁止 1: 使能

24. CAN 控制器

24.1 简介

CAN 控制器支持 CAN2.0A 和 CAN2.0B 协议。因此发送与接收都支持标准的 11 位 ID，也支持 29 位扩展 ID。数据在 CAN 和 CPU 之间是通过 15 个信箱缓冲区进行传送的，而这些缓冲区可以被单独设置为接收或发送缓冲区。每个缓冲区都有一个状态及控制寄存器，用来指示当前的状态以及设置缓冲区的功能。当接收到一个有效帧或者一帧数据成功发送时，可以产生一个中断。另外，如果发生总线错误也能触发中断。

收到的帧信息必须通过 2 个过滤器掩码中的一个才能被正确接收。过滤掩码可以被设置为只接收一个单独的 ID 或一组 ID。其中缓冲区 14 单独使用一个信息过滤器。远程传送请求可以被自动的重配置为在发送之后进行接收或者在接收之后进行发送。优先级控制器允许每个缓冲区都能配为 16 级优先级，其中包含最低和最高优先级，全部缓冲区的优先级一共有 240 种。

CAN 控制器有一个 16 位的时间戳计数器可以提供实时应用功能，计数器的内容可以被接收或发送缓冲区获取。计数器可以通过总线网络进行同步，这个同步机制允许计数器在接收或发送一个缓冲区信息后被清零。

CAN 控制器挂在 APB 总线上。CPU 通过修改寄存器来控制 CAN 模块。包括 CAN 的波特率初始化，引脚上的电平以及使能/失能 CAN 控制器。CAN 控制器具有一系列的诊断特性，如回环测试，监听模式及错误诊断。

24.2 主要特性

CAN 控制器实现以下特性：

- 符合CAN 2.0B规范
  - 标准的数据帧和远程帧
  - 扩展的数据帧和远程帧
  - 0 – 8字节数据长度
  - 高达1Mbps的可编程波特率
- 15个信息缓冲区，每个都可单独配置为接收或发送缓冲区
  - 信息缓冲区是16位的双向RAM
  - 缓冲区可用作为基本CAN路径
- 支持远程帧
  - 接收到远程发送请求后自动发送
  - 发送完一个远程发送请求后自动接收
- 接收过滤器
  - 两个过滤器：全局的接受掩码和单个缓冲区标识符
  - 其中一个缓冲区（14）使用单独的过滤过程
- 可编程的发送优先级
- 中断功能
  - 所有的信息缓冲区（接收/发送/错误）共用一个中断向量
  - 每个中断源可以被单独的使能或禁止
- 16位时间戳计数器，用于成功接收或发送。输入/输出引脚具有推挽功能
- 诊断功能
  - 错误识别
  - 回环模式和监听模式，用于测试和初始化

24.3 功能描述

CAN 控制器由三个部分组成：CAN 内核，接口管理以及信息存储器。

图24-1　　CAN 控制器框图

24.3.2 CAN 信号

内部及外部 CAN 收发器引脚描述如下表

芯片级信号	CAN 顶层信号	输入/输出	描述
CANTX	cantx_pin	输出	传送数据到 CAN 总线 注：cantx_pin 的电平逻辑可由寄存器 CGCR.CTX 设置
CANRX	canrx_pin	输入	从 CAN 总线接收数据 注：canrx_pin 的电平逻辑可由寄存器 CGCR.CRX 设置

24.3.3 CAN 内核

内核实现的 CAN 协议功能包括位填充，CRC 计算/校验和错误管理。它根据总线规则控制传送逻辑以及产生错误信号。另外，它还将 CPU 并行发送的数据流转换为串行的 CAN 总线数据。

24.3.4 接口管理

接口管理是由寄存器和接口管理处理单元实现的。寄存器区为 CAN 控制器提供来自于 CPU 的控制信息，反过来也将 CAN 控制器的状态信息返回给 CPU。同时，还向 CPU 触发中断。

接口管理处理单元是一个状态机，执行 CPU 的发送与接收命令。并控制多个消息缓冲区和 RX/TX 移位寄存器之间的数据传输。

24.3.5 信息缓冲区

15 个信息缓冲区被重映射到 RAM，用来发送和接收 CAN 总线上的数据。每个缓冲区都有 8 个 16 位的寄存器。其中一个寄存器包含着信息缓冲区的配置和当前状态的控制状态信息，其他的寄存器主要用于信息辨识，最多 8 字节的数据以及时间戳信息。 在接收过程中，收到的信息首先被存放在隐藏的缓冲区中，直到信息被正确识别。然后这个缓冲区的内容被拷贝到第一个符合接受此 ID 的缓冲区中。

24.3.6 时钟发生器

CAN 的时钟预分频结构如下图所示。它对输入的 APB 时钟进行分频，分频系数在寄存器 CTIM 中定义。分频后的时钟被称为时间份额，同时被定义为一个时间份额（tq）长度。

关于 CTIM 寄存器的描述请参考 CAN Timing Register(CTIM)章节。

图24-2　　位速率发生器

注意：PSC是时钟预分频器的数值，TSEG1和TSEG2是段1和段2的时间长度。

总线时钟可以按以下公式计算得到：

$$busclock = \frac{pclk}{(PSC)x(1 + TSEG1 + TSEG2)}$$

PSC，TSEG1和TSEG2的数值是在PSC寄存器中定义的：
PSC = PSC[6:0] + 2
TSEG1 = TSEG1[3:0] + 1
TSEG2 = TSEG2[2:0] + 1

24.4 信息传输

CAN 控制器能访问 15 个独立的信息缓冲区（地址被映射在 RAM 区）。每个信息缓冲区都是由 8 个 16 位的 RAM 区组成，可以被单独的配置为一个接收信息缓冲区或发送信息缓冲区。

一个专用的接收过滤器允许用户设置每个缓冲区用于接收单个 ID 或接收一组 ID。缓冲区 14 使用独立的过滤过程，此过程能提供建立基本 CAN 路径。

对于接收到数据或远程帧，CAN 控制器遵循“匹配优先”原则，这意味着给定的信息帧只会有一个缓冲区接收到，即被最先与帧 ID 匹配到的缓冲区接收到。

帧的传输可以通过用户软件写状态寄存器和优先级寄存器来初始化。另一种安排传输的方法是自动应答远程帧。在后一种情况下，CAN 控制器为每一个缓冲区设置传输，以响应具有给定标识和符合掩码设置的远程帧。这意味着单个远程帧能够轮询多个符合配置为响应触发远程传输请求的缓冲区。

24.5 验收过滤

两个 32 位掩码 GMASK 和 BMASK 用于过滤来自 CAN 总线不需要的消息，掩码和缓冲区控制关系如下图。

缓冲区区 0~13 是通过全局过虑掩码（GMASK）和每个缓冲区的 ID 对输入信息进行接受过滤。

缓冲区 14 通过单独的过滤掩码（BMASK）和此缓冲区的 ID 对输入信息进行接受过滤。缓冲区缓冲区缓冲区

图24-3　　接收过滤器结构图

24.5.1 过滤缓冲区 0 到 13

一旦收到的对象在隐藏缓冲区中等待被复制到缓冲区，CAN 控制器会扫描被配置为接收缓冲区的所有缓冲区来匹配过滤掩码。从缓冲区 0 开始按升序检查缓冲区 0 到 13，然后隐藏缓冲区的内容被复制到匹配到过滤掩码的第一个缓冲区中。

全局过滤掩码（GMASK）中保持‘1’的位表示“忽略”每个缓冲区标识中的相应位，无论这个缓冲区标识中相应的位是“0”还是“1”都可以接受。

这就能为每个缓冲区配置成只接受单个 ID 或接受一组 ID。以下两个例子说明了不同之处

例 1：接受单个标识符

如果全局掩码设置为 0x00，则传入消息的接收过滤仅由单个缓冲区 ID 确定，这意味着每个缓冲区只接受一个消息 ID。

图24-4　　接收单一 ID

例 2：接收识别组

设置为‘1’的全局掩码寄存器中的位将缓冲区 ID 中的相应位状态更改为“忽略”（“X”）。因此，接受与非“忽略”位匹配的所有消息。

图24-5　　接收一组 ID

24.5.2 过滤缓冲区 14

缓冲区 14 使用单独的过滤路径。对于此缓冲区，接收过滤由缓冲区 ID 与基本过滤掩码一起建立。此基本掩码使用与全局掩码相同的方法。将某位设置为‘1’会将缓冲区 ID 中的相关位更改为“忽略”位。因此，基本掩码允许该缓冲区接收大量不频繁的消息。

注意：如果 BMASK 寄存器等于 GMASK 寄存器，则缓冲区 14 的使用方法与缓冲区 0 至 13 相同。

缓冲区 0 到 13 的扫描优先级高于缓冲区 14。如果一个消息已经被缓冲区 0 到 13 匹配接收则不会再被缓冲区 14 检查。

24.5.3 BUFFLOCK 功能

通过将配置寄存器中的 BUFFLOCK 位置 1，接收缓冲区在接收到一个有效帧后会自动锁定。CPU 读取数据并在缓冲区状态字段中写入 RX_READY 后，可以解锁缓冲区。使用锁定功能，用户可以将具有相同标识符或相同标识符组的多条消息保存到多个缓冲区中。在这种情况下，如果具有最高优先级的缓冲区已经接收到消息并且被锁定，那么具有次高优先级的缓冲区也能接收消息（假设两个缓冲区使用相同的接受过滤掩码）。

如下图所示，使能锁定功能时，可以接收多条具有相同 ID 的消息。缓冲区 0、1、13 和 14 的过滤掩码被设置为可接受该消息。缓冲区 0 将接收第一个输入帧，然后锁定缓冲区 0，那么缓冲区 1 将接收下一帧，依此类推。如果所有匹配的接收缓冲区已满且都被锁定，则下一帧将不会被任何缓冲区接收。

图24-6　　使能 BUFFLOCK 的消息存储

24.5.4 接收结构

所有接收到帧在验证有效之前会先存储在隐藏的接收缓冲区中。（接收消息的验证点是倒数第二个 EOF 位。）然后将接收到的标识符与每个缓冲区 ID 以及相应的掩码和状态进行比较。一旦验证通过，隐藏缓冲区的所有内容都将被复制到匹配的消息缓冲区中，如下图所示。

注意：不允许在 CAN 使能后通过 CPU 访问隐藏的接收缓冲区。当 CAN 控制器使能后缓冲区禁止写操作。

图24-7　　接收缓冲结构

24.5.5 数据帧

接收到的数据帧存储在从缓冲区 0 开始的第一个匹配的接收缓冲区中。例如，消息与缓冲区 5 匹配，那么将会从隐藏缓冲区复制到缓冲区 5，同时 RX 请求被清除，CAN 控制器也不会尝试将数据帧与其他任何缓冲区匹配。

隐藏缓冲区的所有内容都将被复制到相应的接收缓冲区中，包括接收到的消息 ID 以及接收到的数据长度代码（DLC）。

注意：当某些掩码位设置为“忽略”时，接收到的数据帧 ID 会覆盖当前在接收缓冲区内的消息 ID。接收帧的 DLC 会替换接收缓冲区原数据帧的 DLC。接收消息的 DLC 不与已存在于消息缓冲区的 CNSTAT 寄存器中的 DLC 进行比较。这意味着 CNSTAT 寄存器的 DLC 代码表示实际属于最新接收消息的数据字节数。

24.5.6 远程帧

CAN 控制器可以通过两种方式处理远程帧。首先，通过将缓冲区配置为 RX_READY 并设置 ID 位（包括 RTR 位），可以像数据帧一样接收远程帧。在这种情况下，相同的过程适用于 “数据帧”章节中所述。其次，远程帧可以触发一个或多个消息缓冲区以在接收后发送数据帧。这个过程描述在“应答远程帧”章节中。

24.5.7 接收时序

一旦 CAN 控制器收到 CAN 总线上的显性位，就会启动接收过程。如果全局掩码或基本掩码匹配，则接收到的数据标识符和数据将存储在隐藏接收缓冲区中。接收数据后，CAN 控制器尝试匹配缓冲区 0 ~14 的缓冲区 ID。在接收到第 6 个 EOF 位后，此时数据如果被验证有效，将被复制到缓冲区中。无论数据长度如何，每帧的复制过程至少需要 17 个 PCLK 时钟（参见 “CPU 访问 CAN 寄存器/存储器”章节）。下图说明接收时序。

图24-8　　接收时序

为了指示数据帧在隐藏缓冲区中等待，在复制数据帧期间相应缓冲区的 BUSY 位 ST [0]会被置 1。当数据复制到缓冲区后，CAN 控制器会立即清除 BUSY 位。复制过程完成后，CAN 控制器将状态字段更改为 RX_FULL。反过来，CPU 应在处理数据时将状态字段更改为 RX_READY。在 CPU 将状态更改为 RX_READY 之前如果同一缓冲区接收到新对象时，CAN 控制器会将状态更改为 RX_OVERRUN，以指示至少有一个帧已被新数据帧覆盖。下表总结了 CAN 控制器的当前状态和最终更新。

表24-2　　在 BUSY 期间写入缓冲状态代码

Current Status	Resulting Status
RX_READY	RX_FULL
RX_NOT_ACTIVE	RX_NOT_ACTIVE
RX_FULL	RX_OVERRUN

在 BUSY 位置位期间，除了状态字段外，所有对接收缓冲区的写操作都无效。如果在 BUSY 激活状态时更改状态位，则 CAN 控制器会更新状态（如上表所示）。

缓冲区状态由 CNSTAT 寄存器中的 ST [3：0]位指示和控制（详见缓冲区状态控制寄存器章节）。

24.5.8 接收程序

用户必须执行以下过程来初始化用于接收 CAN 消息的缓冲区：

（1）配置接收掩码（分别为 GMASK 或 BMASK）。

（2）配置缓冲区 ID。

（3）将消息缓冲区状态配置为 RX_READY。

图24-9　　缓冲读取流程图（缓冲锁定禁止）

为了读取收到的消息，CPU 必须执行以下步骤。

注意：第一步仅适用于使用轮询来获取接收缓冲区的状态。对于中断驱动的接收例程，可以省略第一步。

（1）读取接收缓冲区的状态（CNSTAT）。

如果状态为 RX_READY - 没有收到消息 - 退出。

如果状态为 RX_BUSY - 隐藏接收缓冲区的复制过程尚未完成- 再次读取 CNSTAT。

如果缓冲区配置为 RX_READY 并且其中断被使能，则一旦缓冲区收到消息并进入 RX_FULL 状态，它将立即产生中断。在这种情况下，应遵循下述步骤。

（2）在此阶段可以读取状态，以区别是否是因新消息覆盖最初接收的消息而触发的中断。
（3） 将 RX_READY 写入 CNSTAT。
（4）从消息缓冲区中读取 ID /数据和对象控制位（DLC / RTR）。
（5）再次读取缓冲区状态并检查它不是 RX_BUSYx。如果是，请重复此步骤，直到 RX_BUSYx 消失。
（6）如果缓冲区状态为 RX_FULL 或 RX_OVERRUN，则复制一条或多条新消息。在这种情况下，请从步骤 2 开始。
（7）如果状态仍为 RX_READY（由 CPU 在步骤 2 设置），则清除中断标志位并退出。

启用 BUFFLOCK 功能时（参见 “BUFFLOCK 功能”章节），没有必要在读取缓冲区过程中去检查是否接收到新的消息，因为在接收到第一个有效帧之后该缓冲区已经被锁定。从已被锁定的缓冲区中读取数据的过程如下图所示。

（1）从消息缓冲区中读取 ID /数据和对象控制（DLC / RTR）。
（2）将 RX_READY 写入 CNSTAT。
（3）清除中断挂起位并退出。

图24-10　　缓冲读取流程图（缓冲锁定使能）

24.5.9 RX 缓冲区状态

如下图所示，一旦用户将缓冲区状态从 RX_NOT_ACTIVE 设置为 RX_READY，就可以启动接收过程。

CNSTAT 寄存器的状态部分设置为 0000b 至 0010b。收到消息时，在数据帧从隐藏接收缓冲区复制到消息缓冲区的过程中缓冲区会显示忙状态（RX_BUSYx）.之后缓冲区状态会变为 RX_FULL。现在 CPU 可以读取缓冲区内数据并将缓冲区状态重置为 RX_READY，或者在 CPU 读取缓冲区之前接收新的数据帧。在第二种情况下，缓冲区状态将自动更改为 RX_OVERRUN，以指示至少有一条消息丢失。在复制过程中，缓冲区状态在短时间内再次变为 RX_BUSYx，但在这种情况下，CNSTAT 状态部分将变为 0101b，表明缓冲区之前是 RX_FULL（0100b）。在读取最后收到的消息后，CPU 可以将缓冲区重置为 RX_READY。

注：仅在禁用BUFFLOCK功能时才进行转换

图24-11　　接收缓冲状态

24.6 传输结构

为了传输 CAN 消息，用户必须先将缓冲区状态更改为 TX_NOT_ACTIVE 然后再配置消息缓冲区。如果缓冲区状态控制寄存器（CNSTAT）的 ST[3]位设置为‘1’，则缓冲区配置为发送功能。在 TX_NOT_ACTIVE 状态下，缓冲区已准备好从 CPU 接收数据。在接收到所有传输数据（ID，数据字节，DLC 和 PRI）之后，CPU 可以通过将 TX_ONCE 写入缓冲区状态寄存器来开始传输。在传输过程中，缓冲区的状态为 TX_BUSYx。成功传输后，CAN 控制器将缓冲区状态重置为 TX_NOT_ACTIVE。

当传输过程失败时，缓冲区状态保持为 TX_BUSYx，控制器会重新传输，直到数据帧成功传送或者 CPU 取消传输请求为止。

24.6.1 发送远程帧

为了向其他 CAN 节点发送远程传输请求（远程帧），用户需要将消息标识符的 RTR 位设置为‘1’（参见“远程存储消息“章节）并将消息缓冲区的状态更改为 TX_ONCE。成功传输此远程帧后，此消息缓冲区将自动进入 RX_READY 状态，并准备好接收相应的响应数据。注意，需要将缓冲区的掩码的 RTR / XRTR 位设置为接收数据帧（RTR = 0），该缓冲区之前已被配置为发送远程帧（RTR = 1）。

24.6.2 应答远程帧

如果 CPU 将 TX_RTR 写入缓冲区状态寄存器，则缓冲区等待远程帧。当远程帧通过一个或多个缓冲区的接收过滤掩码时，此缓冲区状态变为 TX_ONCE_RTR，缓冲区内容被传输，之后，CAN 控制器再次将 TX_RTR 写入状态码寄存器。

如果 CPU 将 TX_ONCE_RTR 写入缓冲区状态，则发送缓冲区内容，成功发送后，缓冲区进入“等待远程帧”条件（TX_RTR）。

24.6.3 传输调度

将 TX_ONCE 写入缓冲区状态后，传输开始，BUSY 位置 1。一旦缓冲区处于 TX_BUSY 状态，除了 CNSTAT 寄存器的 ST [3：1]位之外，CPU 不可以访问缓冲区。随着当前帧的 CRC 字段的开始，CAN 控制器寻找另一个缓冲区发送请求，并通过将缓冲区状态从 TX_ONCE 改为 TX_BUSY 来为下一次发送选择具有最高优先级的缓冲区。只要下一帧的传输尚未开始，该发送请求可以由 CPU 取消，或者可以被具有更高优先级的缓冲区的另一发送请求覆盖。这意味着，在当前帧的 CRC 字段的开始和下一帧的传输开始之间，有两个缓冲区(当前缓冲区和当前调度为下一次传输的缓冲区)处于 BUSY 状态。为了取消下一帧的发送请求，CPU 必须将缓冲区状态改变为 TX_NOT_ACTIVE。当发送请求被另一个更高优先级缓冲区的请求覆盖时，CAN 控制器将缓冲区状态从 TX_BUSY 更改为 TX_ONCE。因此，此传输请求将保持未决状态。下图进一步说明传输时序。

图24-12　　数据传输

如果传输失败或仲裁丢失，则传输停止，并在中断接收或错误指令完成后继续（见下图）。在这种情况下，会接着选择新的缓冲区并再次执行传输。

注意：取消的消息可能会被具有更高优先级的缓冲区的 TX 请求延迟。当 TX_BUSY 为高时，用户无法更改消息缓冲区对象的内容。在所有情况下，将忽略写入 BUSY 位的操作。

图24-13　　数据传输期间丢失的仲裁

24.6.4 传输优先级

CAN 控制器能够在两条消息间不释放总线的情况下生成预定消息流，从而可以优化性能。它将在发送完上一条消息后立即对总线进行仲裁，并且只有在仲裁丢失的情况下才会释放总线。

如果计划传输多个缓冲区，则优先级由消息缓冲区编号和 CNSTAT 寄存器中的优先级代码构成。优先级的 8 位值由 4 位 TXPRI 值和 4 位缓冲区编号（0 ~14）组成，如下图所示。数字值越低表示优先级越高。

图24-14　　发送优先级编码

例 3：当所有缓冲区的优先级设置为 TXPRI = 0 时的发送优先级配置

表24-3　　发送优先级（TXPRI=0）

TXPRI	Buffer Number	PRI	TX Priority
0	0	0	highest
0	1	1	
:	:	:	:
:	:	:	:
0	14	14	lowest

示例 4：当 TXPRI 与缓冲区号不同时发送优先级配置

表24-4　　发送优先级（TXPRI 不为 0）

TXPRI	Buffer Number	PRI	TX Priority
14	0	224	lowest
13	1	209	
12	2	194	
11	3	179	
10	4	164	
9	5	149	
8	6	134	
7	7	119	
6	8	104	
5	9	89	
4	10	74	
3	11	59	
2	12	44	
1	13	29	
0	14	14	highest

注意：如果两个缓冲区具有相同的优先级（PRI），则具有较低缓冲区编号的缓冲区将获得更高的优先级。

24.6.5 传输过程

CAN 消息的传输必须按如下方式执行（参见下图）.

（1） 将 CNSTAT 状态字段配置为 TX_NOT_ACTIVE。

如果状态是 TX_BUSY，则先前的传输请求仍然处于挂起状态，并且用户无法访问缓冲区的数据内容。在这种情况下，用户可以选择等待，直到缓冲区再次可用（如图所示）。还可以选择退出更新过程,直到缓冲区数据已传输并产生中断，或传输因错误而中止后。

（2）加载缓冲区标识符和数据寄存器

（对于远程帧，应设置标识符的 RTR 位，数据字节可以省略。）

（3）将 CNSTAT 状态字段配置为所需的值：

TX ONCE：触发单帧的传输过程。

TX_ONCE_RTR：触发单个数据帧的传输，然后等待接收的远程帧触发连续的数据帧。

TX_RTR：等待远程帧触发数据帧的传输。

在 CNSTAT 状态字段中写入 TX_ONCE 或 TX_ONCE_RTR 将设置 CAN 控制器的内部发送请求。

如果缓冲区配置为 TX_RTR 并且接收到远程帧，则将自动发送此寻址到的缓冲区的数据内容，无需 CPU 操作。

图24-14　　写缓冲流程

24.6.6 TX 缓冲区状态

用户加载缓冲寄存器（数据，ID，DLC，PRI）并将缓冲区状态从 TX_NOT_ACTIVE 设置为 TX_ONCE，TX_RTR 或 TX_ONCE_RTR 后，可以开始传输。

当 CPU 写入 TX_ONCE 时，一旦 CAN 控制器调度该缓冲区为下一次传输缓冲区，缓冲区就变为 TX_BUSY。该帧成功发送后，缓冲区自动复位为 TX_NOT_ACTIVE（发送的是数据帧时），或者复位为 TX_READY（发送的是远程帧时）。

当 CPU 将消息缓冲区配置为 TX_ONCE_RTR 时，消息缓冲区首先发送其数据内容。在传输期间，缓冲区状态为 1111b，因为 CPU 将 1110b 写入 CNSTAT 寄存器的状态部分。成功传输后，缓冲区进入 TX_RTR 状态并等待远程帧。当消息缓冲区收到远程帧时，它返回 TX_ONCE_RTR 状态，发送其数据字节，然后返回 TX_RTR。如果 CPU 将 1010b 写入缓冲区状态部分，则消息缓冲区进入 TX_RTR 状态，但在等待远程帧之前不发送其数据字节。

下图说明了可能的发送缓冲区状态。

图24-15　　传输缓冲状态

24.7 中断

24.7.1 中断条件

CAN 控制器可以访问 CPU 中的一个中断向量。中断过程可以初始化为以下来源：

■ CAN 数据传输

（1） 缓冲区中接收到有效数据帧（缓冲区状态从 RX_READY 变为 RX_FULL 或 RX_OVERRUN）。

（2） 成功传输数据帧（缓冲区状态从 TX_ONCE 变为 TX_NOT_ACTIVE 或 RX_READY）。

（3） 成功响应远程帧（缓冲区状态从 TX_ONCE_RTR 变为 TX_RTR）。

（4） 传输调度（缓冲区状态从 TX_RTR 变为 TX_ONCE_RTR。）

■ CAN 错误条件

（1） 检测到 CAN 错误（设置 CIPND 寄存器中的 CEIPND 位以及错误诊断寄存器 CEDIAG 中的相应位）。

可以在寄存器(CIEN)中单独使能/禁止对每个消息缓冲区的接收/发送中断。消息缓冲区的挂起标志位于寄存器(CIPND)（只读）中，可以通过复位寄存器(CICLR)中的标志来清除。

24.7.2 最高优先级中断代码

为了减少 CIPND 寄存器的解码时间，将具有最高优先级的缓冲中断请求作为中断状态代码放入 CSTPND 寄存器的 IST [3：0]部分。

可以在 CAN 中断使能寄存器（CIEN）中单独使能或禁止每个缓冲区的中断以及错误中断。一旦发生中断条件，每个中断请求都由 CAN 中断挂起寄存器（CIPND）中的标志指示。当使能当前最高优先级中断请求的中断代码逻辑时，该中断将被转换为 CAN 状态挂起寄存器（CSTPND）的 IST [3：0]位。通过将 CAN 中断清除寄存器（CICLR）中的相应位置“1”，可以清除中断请求。

下图说明了 CAN 控制器中断管理。

图24-16 CAN 中断管理

优先级最高的中断源转换为 IRQ 和 IST [3：0]位，如下表所示。

表24-5　　最高优先级中断代码（ICEN = 0XFFFF）

CAN Interrupt Request	IRQ	IST[3]	IST[2]	IST[1]	IST[0]
No request	0	0	0	0	0
Error Interrupt	1	0	0	0	0
Buffer 0	1	0	0	0	1
Buffer 1	1	0	0	1	0
Buffer 2	1	0	0	1	1
Buffer 3	1	0	1	0	0
Buffer 4	1	0	1	0	1
Buffer 5	1	0	1	1	0
Buffer 6	1	0	1	1	1
Buffer 7	1	1	0	0	0
Buffer 8	1	1	0	0	1
Buffer 9	1	1	0	1	0
Buffer 10	1	1	0	1	1
Buffer 11	1	1	1	0	0
Buffer 12	1	1	1	0	1
Buffer 13	1	1	1	1	0
Buffer 14	1	1	1	1	1

■　用法提示

（1）代码部分 IST [3：0]可以在中断处理程序中用作置换，以便跳转到相关的子程序。

（2）CAN 中断代码使能（CICEN）寄存器可用于 CAN 中断处理程序，如果用户首先要处理所有接收缓冲区中断，之后是所有发送缓冲区中断。在这种情况下，用户可以首先仅使能所有接收缓冲区中断，按其优先级顺序对所有待处理的中断请求进行编码，扫描和服务。然后，用户更改 CICEN 寄存器以禁用所有接收缓冲区且使能所有发送中断，再根据优先级处理所有的发送中断。

24.8 时间戳计数器

CAN 控制器具有一个自由运行的 16 位定时器（CTMR），以 CAN 总线上可识别的每一位时间递增。在成功发送或接收消息之后, ACK 时间份额期间该定时器的值会被捕获到消息缓冲区的 TSTP 寄存器中。下图显示了时间戳计数器的简化框图。

图24-17　　时间戳计数器

通过从缓冲区 0 接收或发送消息，可以在 CAN 网络上同步定时器。在这种情况下，缓冲区 0 的 TSTP 寄存器在消息的 ACK 时间份额期间捕获当前 CTMR 值（如上所述），之后 CTMR 重置为 0x0000。可以通过 CGCR.TSTAMP 位启用或禁用同步。

24.9 存储器组织

CAN 控制器占用内存地址空间中的 144 个半字（16 位）。该空间分为用于消息缓冲区的 15 * 8 + 8（保留）字和用于控制和状态的 14 + 2（保留）字。这 144 个字是 32 位对齐的。（CPU APB 只能进行 32 位的字访问,其中的高 16 位数据没有意义。）

24.9.1 CPU 访问 CAN 寄存器/存储器

消息缓冲区占用的所有内存位置由 CPU 和 CAN 控制器（双端口 RAM）共享。CAN 控制器和 CPU 具有对该存储器的单周期访问，CPU 具有优先权。但是，如果发生访问竞争，则每个周期都会改变对内存的访问，直到竞争解决为止。该内部访问仲裁对用户是透明的。

如果缓冲区在接收对象期间忙（在进行从隐藏接收缓冲区复制的过程）或被调度传输，则 CPU 无法对缓冲区的数据内容进行写操作。写入状态/控制字节和读取整个缓冲区的访问是始终被允许的。。

所有配置和状态寄存器只能由 CAN 控制器或 CPU 访问。这些寄存器提供单周期固定宽度访问，无任何潜在的等待状态。

24.9.2 寄存器分布

下一节中的所有寄存器描述都使用以下布局。每个字的高 16 位没有使用,所有对其的操作都被忽略.

bit 32	bit 16	bit 15	... bit number ...	bit 0
		... bit name ...		
		... reset value ...		
... always reads 0 / writes are ignored		... CPU access ... r = register bit is read only w = register bit is write only r/w = register bit is read/write		

24.9.3 消息缓冲区组织

消息缓冲区是 CAN 和 CPU 之间的通信接口，用于 CAN 帧的发送和接收。 RAM 位置的固定地址有 15 个消息缓冲区。如下表所示, 每个缓冲区由 2 个用于标识符的字，4 个用于最多 8 个 CAN 数据字节的字，1 个用于时间戳的字，1 个用于数据长度，发送优先级和缓冲区状态代码的字。

表24-6　　消息缓冲区组织

ADDR	Buffer Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1c	ID1	XI28 ID10	XI27 ID9	XI26 ID8	XI25 ID7	XI24 ID6	XI23 ID5	XI22 ID4	XI21 ID3	XI20 ID2	XI19 ID1	XI18 ID0	SRR RTR	IDE	XI17	XI16	XI15
0x18	ID0	XI14	XI13	XI12	XI11	XI10	XI9	XI8	XI7	XI6	XI5	XI4	XI3	XI2	XI1	XI0	RTR
0x14	DATA0	Data 1.7	Data 1.6	Data 1.5	Data 1.4	Data 1.3	Data 1.2	Data 1.1	Data 1.0	Data 2.7	Data 2.6	Data 2.5	Data 2.4	Data 2.3	Data 2.2	Data 2.1	Data 2.0
0x10	DATA1	Data 3.7	Data 3.6	Data 3.5	Data 3.4	Data 3.3	Data 3.2	Data 3.1	Data 3.0	Data 4.7	Data 4.6	Data 4.5	Data 4.4	Data 4.3	Data 4.2	Data 4.1	Data 4.0
0x0c	DATA2	Data 5.7	Data 5.6	Data 5.5	Data 5.4	Data 5.3	Data 5.2	Data 5.1	Data 5.0	Data 6.7	Data 6.6	Data 6.5	Data 6.4	Data 6.3	Data 6.2	Data 6.1	Data 6.0
0x08	DATA3	Data 7.7	Data 7.6	Data 7.5	Data 7.4	Data 7.3	Data 7.2	Data 7.1	Data 7.0	Data 8.7	Data 8.6	Data 8.5	Data 8.4	Data 8.3	Data 8.2	Data 8.1	Data 8.0
0x04	TSTP	TSTP15	TSTP14	TSTP13	TSTP12	TSTP11	TSTP10	TSTP9	TSTP8	TSTP7	TSTP6	TSTP5	TSTP4	TSTP3	TSTP2	TSTP1	TSTP0
0x00	CNTSTAT	DLC3	DLC2	DLC1	DLC0	reserved				PRI3	PRI2	PRI1	PRI0	ST3	ST2	ST1	ST0

24.9.4 缓冲状态/控制寄存器（CNSTAT）

缓冲区状态，缓冲区优先级和数据长度代码通过操作缓冲区状态/控制寄存器的内容来控制。 CPU 和 CAN 控制器可以访问该寄存器。

bit 15	bit 14	bit 13	bit 12	bit 11	bit 10	bit 9	bit 8	bit 7	bit 6	bit 5	bit 4	bit 3	bit 2	bit 1	bit 0
DLC[3:0]				reserved				PRI[3:0]				ST[3:0]			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r/w															

缓冲区状态 ST[3:0]

CNSTAT 寄存器有一个状态部分 ST[3:0]，其包含缓冲区的状态信息，如下表所示。此部分可以被 CAN 控制器修改。

表24-7　　CNSTAT 寄存器的缓冲区状态部分

ST[3] (DIR)	ST[2]	ST[1]	ST[0] (BUSY)	Buffer Status
0	0	0	0	RX_NOT_ACTIVE
0	0	0	1	reserved for RX_BUSY [1]
0	0	1	0	RX_READY
0	0	1	1	RX_BUSY0[2]
0	1	0	0	RX_FULL
0	1	0	1	RX_BUSY1[2]
0	1	1	0	RX_OVERRUN
0	1	1	1	RX_BUSY2[2]
1	0	0	0	TX_NOT_ACTIVE
1	0	0	1	reserved for TX_BUSY[3]
1	1	0	0	TX_ONCE
1	1	0	1	TX_BUSY0[4]
1	0	1	0	TX_RTR (automatic response to a remote frame)
1	0	1	1	reserved for TX_BUSY1[5]
1	1	1	0	TX_ONCE_RTR (changes to TX_RTR after transmission)
1	1	1	1	TX_BUSY2[4]

注 1：此情况表示用户将 RX_NOT_ACTIVE 写入了数据复制过程仍处于活动状态的缓冲区。
注 2：RX_BUSYx 表示正在进行复制,有三种可能的情况：
（1）首次复制数据 RX_READY - > RX_BUSY0
（2）数据被第二次复制 RX_FULL - > RX_BUSY1
（3） 数据被复制第三次或更多次 RX_OVERRUN - > RX_BUSY2
注 3：该状态表示用户将 TX_NOT_ACTIVE 写入了被调度为发送或正在发送的缓冲区。
注 4：TX_BUSYx 表示缓冲区被调度用于传输或正在进行传输; 它可以是两种状态之一：
（1）一个消息正在等待传输或正在传输
（2）一个自动应答正在等待传输或正在传输
注 5： 这种情况不会发生。

ST [0]位用作缓冲区忙指示。当 BUSY 位置 1 时，除了 CNTSTAT 寄存器的低字节外，禁止任何对缓冲区的写操作。如果缓冲区数据当前正从隐藏缓冲区复制，或者消息被调度传输或当前正在传输，则 CAN 控制器会设置此位。 CAN 控制器在状态更新时一定会复位该位。

PRI [3:0] 包含消息缓冲区的用户定义发送优先级代码。

DLC [3:0]确定接收帧或发送帧的数据字节数。对于发送，应根据发送数据的字节数来设置这些位。对于接收，这些位表示消息缓冲区中接收的有效数据的字节数。下表显示 DLC [3：0]的可能位组合，数据长度为 0 到 8 个字节。

表24-8　　数据长度编码

Number of Data Bytes	DLC[3]	DLC[2]	DLC[1]	DLC[0]
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0

注意：即使数据长度代码设置为大于 8 的值，接收/发送的最大数据字节数也是 8.因此，如果数据长度代码大于或等于 8 字节，则位 DLC [2:0]被忽略。

24.9.5 存储标准消息

在处理标准帧期间，扩展标识符位（IDE）设置为‘0’。位 ID1 [3:0]，ID0 [15:0]是”忽略”位。

表24-9　　具有 8 个数据字节的标准帧

ADDR	Buffer Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1c	ID1	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR	IDE	don't care		
0x18	ID0	don't care															
0x14	DATA0	Data 1.7	Data 1.6	Data 1.5	Data 1.4	Data 1.3	Data 1.2	Data 1.1	Data 1.0	Data 2.7	Data 2.6	Data 2.5	Data 2.4	Data 2.3	Data 2.2	Data 2.1	Data 2.0
0x10	DATA1	Data 3.7	Data 3.6	Data 3.5	Data 3.4	Data 3.3	Data 3.2	Data 3.1	Data 3.0	Data 4.7	Data 4.6	Data 4.5	Data 4.4	Data 4.3	Data 4.2	Data 4.1	Data 4.0
0x0c	DATA2	Data 5.7	Data 5.6	Data 5.5	Data 5.4	Data 5.3	Data 5.2	Data 5.1	Data 5.0	Data 6.7	Data 6.6	Data 6.5	Data 6.4	Data 6.3	Data 6.2	Data 6.1	Data 6.0
0x08	DATA3	Data 7.7	Data 7.6	Data 7.5	Data 7.4	Data 7.3	Data 7.2	Data 7.1	Data 7.0	Data 8.7	Data 8.6	Data 8.5	Data 8.4	Data 8.3	Data 8.2	Data 8.1	Data 8.0
0x04	TSTP	TSTP 15	TSTP 14	TSTP 13	TSTP 12	TSTP 11	TSTP 10	TSTP 9	TSTP 8	TSTP 7	TSTP 6	TSTP 5	TSTP 4	TSTP 3	TSTP 2	TSTP 1	TSTP 0
0x00	CNTSTAT	DLC3	DLC2	DLC1	DLC0	reserved				PRI3	PRI2	PRI1	PRI0	ST3	ST2	ST1	ST0

IDE

标识符扩展 - IDE 设置为‘0’以指示消息是使用 11 个标识符位的标准帧。如果 IDE 设置为‘1’，则存储在缓冲区中的对象将作为扩展帧处理。

RTR

远程传输请求 - RTR 设置为‘1’以指示该消息是远程帧。对于数据帧，RTR 位应设置为‘0’。

ID[10:0]

ID 位段 ID[10:0]用于 11 个标准帧标识符位。

存储的消息少于 8 字节数据

不处理这些没有被用作传输的字节。如果是传输对象，则将忽略这些字节中的数据。如果是接收对象，则这些字节中的数据将被无效的数据覆盖。

24.9.6 存储扩展消息

如果 IDE 位被设置为‘1’，则缓冲区处理扩展帧。扩展 ID 的存储遵循下表中的描述。 SRR 替换标准帧中的 RTR 位，并应作为‘1’发送。

表24-10　具有 8 个数据字节的扩展消息

ADDR	Buffer Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1c	ID1	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	SRR	IDE	ID17	ID16	ID15
0x18	ID0	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
0x14	DATA0	Data 1.7	Data 1.6	Data 1.5	Data 1.4	Data 1.3	Data 1.2	Data 1.1	Data 1.0	Data 2.7	Data 2.6	Data 2.5	Data 2.4	Data 2.3	Data 2.2	Data 2.1	Data 2.0
0x10	DATA1	Data 3.7	Data 3.6	Data 3.5	Data 3.4	Data 3.3	Data 3.2	Data 3.1	Data 3.0	Data 4.7	Data 4.6	Data 4.5	Data 4.4	Data 4.3	Data 4.2	Data 4.1	Data 4.0
0x0c	DATA2	Data 5.7	Data 5.6	Data 5.5	Data 5.4	Data 5.3	Data 5.2	Data 5.1	Data 5.0	Data 6.7	Data 6.6	Data 6.5	Data 6.4	Data 6.3	Data 6.2	Data 6.1	Data 6.0
0x08	DATA3	Data 7.7	Data 7.6	Data 7.5	Data 7.4	Data 7.3	Data 7.2	Data 7.1	Data 7.0	Data 8.7	Data 8.6	Data 8.5	Data 8.4	Data 8.3	Data 8.2	Data 8.1	Data 8.0
0x04	TSTP	TSTP 15	TSTP 14	TSTP 13	TSTP 12	TSTP 11	TSTP 10	TSTP 9	TSTP 8	TSTP 7	TSTP 6	TSTP 5	TSTP 4	TSTP 3	TSTP 2	TSTP 1	TSTP 0
0x00	CNTSTAT	DLC3	DLC2	DLC1	DLC0	reserved				PRI3	PRI2	PRI1	PRI0	ST3	ST2	ST1	ST0

SRR

替代远程请求。 SRR 仅用于替换标准帧中的 RTR 位。如果缓冲区配置为发送带有扩展标识符的消息，则用户应将 SRR 位应设置为‘1’。它就会在 CAN 总线上被监控接收。

IDE

标识符扩展。 IDE 设置为‘0’以指示消息是使用 11 个标识符位的标准帧。如果 IDE 设置为‘1’，则存储在缓冲区中的对象将作为扩展帧处理。

RTR

远程传输请求。 RTR 设置为‘1’以指示该消息是远程帧。对于数据帧，RTR 位应设置为‘0’。

ID[28:0]

用于构建扩展帧的 29 位标识符。

24.9.7 存储远程消息

在远程帧传输期间，不必关心缓冲寄存器 DATA [3：0]的值。如果是发送远程帧，这些寄存器的内容则被忽略。如果是接收到远程帧，则这些寄存器的内容将会被无效数据覆盖。表格中显示了具有扩展标识符的远程帧的消息缓冲区设置结构，如下表所示.

表24-11　　扩展的远程框架

ADDR	Buffer Register	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x1c	ID1	ID28	ID27	ID26	ID25	ID24	ID23	ID22	ID21	ID20	ID19	ID18	SRR	IDE	ID17	ID16	ID15
0x18	ID0	ID14	ID13	ID12	ID11	ID10	ID9	ID8	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0	RTR
0x14	DATA0	don't care															
0x10	DATA1	don't care															
0x0c	DATA2	don't care															
0x08	DATA3	don't care															
0x04	TSTP	TSTP 15	TSTP 14	TSTP 13	TSTP 12	TSTP 11	TSTP 10	TSTP 9	TSTP 8	TSTP 7	TSTP 6	TSTP 5	TSTP 4	TSTP 3	TSTP 2	TSTP 1	TSTP 0
0x00	CNTSTAT	DLC3	DLC2	DLC1	DLC0	reserved				PRI3	PRI2	PRI1	PRI0	ST3	ST2	ST1	ST0

SSR

替代远程请求。 SRR 仅用于替换标准帧中的 RTR 位。用户应将 SRR 位应设置为‘1’。

IDE

标识符扩展。IDE 设置为‘0’以指示消息是使用 11 个标识符位的标准帧。如果 IDE 设置为‘1’，则存储在缓冲区中的对象将作为扩展帧处理。

RTR

远程传输请求。 RTR 设置为‘1’以指示该消息是远程帧。对于数据帧，RTR 位应设置为‘0’。

ID[28：0]

ID 比特 0 到比特 28 用于构建扩展帧的 29 位标识符。 ID1 寄存器中的 ID28 至 ID18 用于 11 个标准帧标识符位。

24.10 寄存器

CAN 模块寄存器列表 (基地址:0x4000 2C00)

地址	寄存器名	说明
0x4000 2C00	BUF0_CNTSTAT	缓冲区0状态寄存器
0x4000 2C04	BUF0_TSTP	缓冲区0时间戳寄存器
0x4000 2C08	BUF0_DATA3	缓冲区0数据3寄存器
0x4000 2C0C	BUF0_DATA2	缓冲区0数据2寄存器
0x4000 2C10	BUF0_DATA1	缓冲区0数据1寄存器
0x4000 2C14	BUF0_DATA0	缓冲区0数据0寄存器
0x4000 2C18	BUF0_ID0	缓冲区0标识符0寄存器
0x4000 2C1C	BUF0_ID1	缓冲区0标识符1寄存器
0x4000 2C20	BUF1_CNTSTAT	缓冲区1状态寄存器
0x4000 2C24	BUF1_TSTP	缓冲区1时间戳寄存器
0x4000 2C28	BUF1_DATA3	缓冲区1数据3寄存器
0x4000 2C2C	BUF1_DATA2	缓冲区1数据2寄存器
0x4000 2C30	BUF1_DATA1	缓冲区1数据1寄存器
0x4000 2C34	BUF1_DATA0	缓冲区1数据0寄存器
0x4000 2C38	BUF1_ID0	缓冲区1标识符0寄存器

0x4000 2C3C	BUF1_ID1	缓冲区1标识符1寄存器
0x4000 2C40	BUF2_CNTSTAT	缓冲区2状态寄存器
0x4000 2C44	BUF2_TSTP	缓冲区2时间戳寄存器
0x4000 2C48	BUF2_DATA3	缓冲区2数据3寄存器
0x4000 2C4C	BUF2_DATA2	缓冲区2数据2寄存器
0x4000 2C50	BUF2_DATA1	缓冲区2数据1寄存器
0x4000 2C54	BUF2_DATA0	缓冲区2数据0寄存器
0x4000 2C58	BUF2_ID0	缓冲区2标识符0寄存器
0x4000 2C5C	BUF2_ID1	缓冲区2标识符1寄存器
0x4000 2C60	BUF3_CNTSTAT	缓冲区3状态寄存器
0x4000 2C64	BUF3_TSTP	缓冲区3时间戳寄存器
0x4000 2C68	BUF3_DATA3	缓冲区3数据3寄存器
0x4000 2C6C	BUF3_DATA2	缓冲区3数据2寄存器
0x4000 2C70	BUF3_DATA1	缓冲区3数据1寄存器
0x4000 2C74	BUF3_DATA0	缓冲区3数据0寄存器
0x4000 2C78	BUF3_ID0	缓冲区3标识符0寄存器
0x4000 2C7C	BUF3_ID1	缓冲区3标识符1寄存器
0x4000 2C80	BUF4_CNTSTAT	缓冲区4状态寄存器
0x4000 2C84	BUF4_TSTP	缓冲区4时间戳寄存器
0x4000 2C88	BUF4_DATA3	缓冲区4数据3寄存器
0x4000 2C8C	BUF4_DATA2	缓冲区4数据2寄存器
0x4000 2C90	BUF4_DATA1	缓冲区4数据1寄存器
0x4000 2C94	BUF4_DATA0	缓冲区4数据0寄存器
0x4000 2C98	BUF4_ID0	缓冲区4标识符0寄存器
0x4000 2C9C	BUF4_ID1	缓冲区4标识符1寄存器
0x4000 2CA0	BUF5_CNTSTAT	缓冲区5状态寄存器
0x4000 2CA4	BUF5_TSTP	缓冲区5时间戳寄存器
0x4000 2CA8	BUF5_DATA3	缓冲区5数据3寄存器
0x4000 2CAC	BUF5_DATA2	缓冲区5数据2寄存器
0x4000 2CB0	BUF5_DATA1	缓冲区5数据1寄存器
0x4000 2CB4	BUF5_DATA0	缓冲区5数据0寄存器
0x4000 2CB8	BUF5_ID0	缓冲区5标识符0寄存器
0x4000 2CBC	BUF5_ID1	缓冲区5标识符1寄存器
0x4000 2CC0	BUF6_CNTSTAT	缓冲区6状态寄存器
0x4000 2CC4	BUF6_TSTP	缓冲区6时间戳寄存器
0x4000 2CC8	BUF6_DATA3	缓冲区6数据3寄存器
0x4000 2CCC	BUF6_DATA2	缓冲区6数据2寄存器
0x4000 2CD0	BUF6_DATA1	缓冲区6数据1寄存器
0x4000 2CD4	BUF6_DATA0	缓冲区6数据0寄存器
0x4000 2CD8	BUF6_ID0	缓冲区6标识符0寄存器
0x4000 2CDC	BUF6_ID1	缓冲区6标识符1寄存器
0x4000 2CE0	BUF7_CNTSTAT	缓冲区7状态寄存器
0x4000 2CE4	BUF7_TSTP	缓冲区7时间戳寄存器
0x4000 2CE8	BUF7_DATA3	缓冲区7数据3寄存器
0x4000 2CEC	BUF7_DATA2	缓冲区7数据2寄存器
0x4000 2CF0	BUF7_DATA1	缓冲区7数据1寄存器
0x4000 2CF4	BUF7_DATA0	缓冲区7数据0寄存器
0x4000 2CF8	BUF7_ID0	缓冲区7标识符0寄存器
0x4000 2CFC	BUF7_ID1	缓冲区7标识符1寄存器
0x4000 2D00	BUF8_CNTSTAT	缓冲区8状态寄存器

0x4000 2D04	BUF8_TSTP	缓冲区8时间戳寄存器
0x4000 2D08	BUF8_DATA3	缓冲区8数据3寄存器
0x4000 2D0C	BUF8_DATA2	缓冲区8数据2寄存器
0x4000 2D10	BUF8_DATA1	缓冲区8数据1寄存器
0x4000 2D14	BUF8_DATA0	缓冲区8数据0寄存器
0x4000 2D18	BUF8_ID0	缓冲区8标识符0寄存器
0x4000 2D1C	BUF8_ID1	缓冲区8标识符1寄存器
0x4000 2D20	BUF9_CNTSTAT	缓冲区9状态寄存器
0x4000 2D24	BUF9_TSTP	缓冲区9时间戳寄存器
0x4000 2D28	BUF9_DATA3	缓冲区9数据3寄存器
0x4000 2D2C	BUF9_DATA2	缓冲区9数据2寄存器
0x4000 2D30	BUF9_DATA1	缓冲区9数据1寄存器
0x4000 2D34	BUF9_DATA0	缓冲区9数据0寄存器
0x4000 2D38	BUF9_ID0	缓冲区9标识符0寄存器
0x4000 2D3C	BUF9_ID1	缓冲区9标识符1寄存器
0x4000 2D40	BUF10_CNTSTAT	缓冲区10状态寄存器
0x4000 2D44	BUF10_TSTP	缓冲区10时间戳寄存器
0x4000 2D48	BUF10_DATA3	缓冲区10数据3寄存器
0x4000 2D4C	BUF10_DATA2	缓冲区10数据2寄存器
0x4000 2D50	BUF10_DATA1	缓冲区10数据1寄存器
0x4000 2D54	BUF10_DATA0	缓冲区10数据0寄存器
0x4000 2D58	BUF10_ID0	缓冲区10标识符0寄存器
0x4000 2D5C	BUF10_ID1	缓冲区10标识符1寄存器
0x4000 2D60	BUF11_CNTSTAT	缓冲区11状态寄存器
0x4000 2D64	BUF11_TSTP	缓冲区11时间戳寄存器
0x4000 2D68	BUF11_DATA3	缓冲区11数据3寄存器
0x4000 2D6C	BUF11_DATA2	缓冲区11数据2寄存器
0x4000 2D70	BUF11_DATA1	缓冲区11数据1寄存器
0x4000 2D74	BUF11_DATA0	缓冲区11数据0寄存器
0x4000 2D78	BUF11_ID0	缓冲区11标识符0寄存器
0x4000 2D7C	BUF11_ID1	缓冲区11标识符1寄存器
0x4000 2D80	BUF12_CNTSTAT	缓冲区12状态寄存器
0x4000 2D84	BUF12_TSTP	缓冲区12时间戳寄存器
0x4000 2D88	BUF12_DATA3	缓冲区12数据3寄存器
0x4000 2D8C	BUF12_DATA2	缓冲区12数据2寄存器
0x4000 2D90	BUF12_DATA1	缓冲区12数据1寄存器
0x4000 2D94	BUF12_DATA0	缓冲区12数据0寄存器
0x4000 2D98	BUF12_ID0	缓冲区12标识符0寄存器
0x4000 2D9C	BUF12_ID1	缓冲区12标识符1寄存器
0x4000 2DA0	BUF13_CNTSTAT	缓冲区13状态寄存器
0x4000 2DA4	BUF13_TSTP	缓冲区13时间戳寄存器
0x4000 2DA8	BUF13_DATA3	缓冲区13数据3寄存器
0x4000 2DAC	BUF13_DATA2	缓冲区13数据2寄存器
0x4000 2DB0	BUF13_DATA1	缓冲区13数据1寄存器
0x4000 2DB4	BUF13_DATA0	缓冲区13数据0寄存器
0x4000 2DB8	BUF13_ID0	缓冲区13标识符0寄存器
0x4000 2DBC	BUF13_ID1	缓冲区13标识符1寄存器
0x4000 2DC0	BUF14_CNTSTAT	缓冲区14状态寄存器
0x4000 2DC4	BUF14_TSTP	缓冲区14时间戳寄存器
0x4000 2DC8	BUF14_DATA3	缓冲区14数据3寄存器

0x4000 2DCC	BUF14_DATA2	缓冲区14数据2寄存器
0x4000 2DD0	BUF14_DATA1	缓冲区14数据1寄存器
0x4000 2DD4	BUF14_DATA0	缓冲区14数据0寄存器
0x4000 2DD8	BUF14_ID0	缓冲区14标识符0寄存器
0x4000 2DDC	BUF14_ID1	缓冲区14标识符1寄存器
0x4000 2E00	CGCR	CAN全局配置寄存器
0x4000 2E04	CTIM	CAN时序寄存器
0x4000 2E08	GMSKX	CAN全局屏蔽寄存器
0x4000 2E0C	GMSKB	CAN全局屏蔽寄存器
0x4000 2E10	BMSKX	CAN基本屏蔽寄存器
0x4000 2E14	BMSKB	CAN基本屏蔽寄存器
0x4000 2E18	CIEN	CAN中断使能寄存器
0x4000 2E1C	CIPND	CAN中断标志寄存器
0x4000 2E20	CICLR	CAN中断标志位清除寄存器
0x4000 2E24	CICEN	CAN中断码使能寄存器
0x4000 2E28	CSTPND	CAN状态挂起寄存器
0x4000 2E2C	CANEC	CAN错误计数器寄存器
0x4000 2E30	CEDIAG	CAN错误诊断寄存器
0x4000 2E34	CTMR	CAN定时器寄存器

24.10.1 缓冲区 n 状态/控制寄存器 (CAN_BUFn_CNTSTAT)(n=0..14)

偏移地址: 0x0000
:0x0020
:0x0040
:0x0060
:0x0080
:0x00A0
:0x00C0
:0x00E0
:0x0100
:0x0120
:0x0140
:0x0160
:0x0180
:0x01A0
:0x01C0

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DLC[3:0]				保留				PRI[3:0]				ST[3:0]			
RW				-				*RW*				*RW*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 12	**DLC[3:0]**	数据长度
11 ~ 8	保留	-
7 ~ 4	**PRI[3:0]**	发送优先级
3 ~ 0	**ST[3:0]**	缓冲区n的状态，参考缓冲状态/控制寄存器（CNSTAT）

24.10.2 缓冲区 n 时间戳寄存器 (CAN_BUFn_TSTP)(n=0..14)

偏移地址: 0x0004
:0x0024
:0x0044
:0x0064
:0x0084
:0x00A4
:0x00C4
:0x00E4
:0x0104
:0x0124
:0x0144
:0x0164
:0x0184
:0x01A4
:0x01C4

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
TSTP[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**TSTP[15:0]**	保存时间戳定时器(CTMR)的值

24.10.3 缓冲区 n 数据 3 寄存器 (CAN_BUFn_DATA3)(n=0..14)

偏移地址: 0x0008
:0x0028
:0x0048
:0x0068
:0x0088
:0x00A8
:0x00C8
:0x00E8
:0x0108
:0x0128
:0x0148
:0x0168
:0x0188
:0x01A8
:0x01C8

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DATAH[7:0]								DATAL[7:0]							
RW								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 8	**DATAH[7:0]**	数据的第7个字节
7 ~ 0	**DATAL[7:0]**	数据的第8个字节

24.10.4 缓冲区 n 数据 2 寄存器 (CAN_BUFn_DATA2)(n=0..14)

偏移地址: 0x000C
:0x002C
:0x004C
:0x006C
:0x008C
:0x00AC
:0x00CC
:0x00EC
:0x010C
:0x012C
:0x014C
:0x016C
:0x018C
:0x01AC
:0x01CC

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DATAH[7:0]								DATAL[7:0]							
RW								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 8	**DATAH[7:0]**	数据的第5个字节
7 ~ 0	**DATAL[7:0]**	数据的第6个字节

24.10.5 缓冲区 n 数据 1 寄存器 (CAN_BUFn_DATA1)(n=0..14)

偏移地址: 0x0010
:0x0030
:0x0050
:0x0070
:0x0090
:0x00B0
:0x00D0
:0x00F0
:0x0110
:0x0130
:0x0150
:0x0170
:0x0190
:0x01B0
:0x01D0

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DATAH[7:0]								DATAL[7:0]							
RW								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 8	**DATAH[7:0]**	数据的第3个字节
7 ~ 0	**DATAL[7:0]**	数据的第4个字节

24.10.6 缓冲区 n 数据 0 寄存器 (CAN_BUFn_DATA0)(n=0..14)

偏移地址: 0x0014
:0x0034
:0x0054
:0x0074
:0x0094
:0x00B4
:0x00D4
:0x00F4
:0x0114
:0x0134
:0x0154
:0x0174
:0x0194
:0x01B4
:0x01D4

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DATAH[7:0]								DATAL[7:0]							
RW								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 8	**DATAH[7:0]**	数据的第1个字节
7 ~ 0	**DATAL[7:0]**	数据的第2个字节

24.10.7 缓冲区 n 标识符 0 寄存器 (CAN_BUFn_ID0)(n=0..14)

偏移地址: 0x0018
:0x0038
:0x0058
:0x0078
:0x0098
:0x00B8
:0x00D8
:0x00F8
:0x0118
:0x0138
:0x0158
:0x0178
:0x0198
:0x01B8
:0x01D8

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
XID[14:0]															XRTR
RW															*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 1	**XID[14:0]**	扩展帧的ID14~ID0
0	**XRTR**	扩展帧的远程请求位,置1表示此消息为远程帧

24.10.8 缓冲区 n 标识符 1 寄存器 (CAN_BUFn_ID1)(n=0..14)

偏移地址: 0x001C
:0x003C
:0x005C
:0x007C
:0x009C
:0x00BC
:0x00DC
:0x00FC
:0x011C
:0x013C
:0x015C
:0x017C
:0x019C
:0x01BC
:0x01DC

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
IDXID[10:0]											RTR SRR	IDE	XID[2:0]		
RW											RW	RW	RW		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 5	**IDXID[10:0]**	标准帧的ID10~ID0, 扩展帧的ID28~ID18
4	**RTRSRR**	扩展帧中替代远程帧请求位（SRR）, 标准帧中此位是远程帧请求位(RTR）
3	**IDE**	扩展帧指示位,0表示标准帧,使用11位ID,1表示一个扩展帧,用29位ID
2 ~ 0	**XID[2:0]**	扩展帧的ID17~ID15

24.10.9 CAN 全局配置寄存器 (CAN_CGCR)

偏移地址: 0x0200
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留				EIT	DIA GEN	INTE RNA L	LOO PBA CK	IGN ACK	LO	DDI R	TST PEN	BUF FLO CK	CTX	CRX	CAN EN
-				RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW	RW
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 12	保留	-
11	**EIT**	**错误中断类型** 0：每个总线错误EIPND位都置起 1：只有当错误状态（CSTPND.NS）改变时EIPND才置起
10	**DIAGEN**	**诊断功能使能位** 0：正常模式 1：诊断功能使能 (诊断功能包括LO，IGNACK，LOOPBACK，INTERNAL模式)
9	**INTERNAL**	**内部模式** 0：正常模式 1：内部模式使能。TX和RX引脚内部相连。
8	**LOOPBACK**	**回环模式** 0：正常模式 1：回环模式使能
7	**IGNACK**	**忽略ACK位** 0：正常模式 1：发送数据后忽略ACK位
6	**LO**	**监听模式(Listen Only).** 0：允许接收/发送数据 1：只允许接收数据，不发送消息或应答
5	**DDIR**	**数据存储方向控制位** 0：先接收到的存储在高位地址，后接收到的存储在低位地址 1：先接收到的存储在低位地址，后接收到的存储在高位地址
4	**TSTPEN**	**时间同步使能位。** 0：时间同步失能，时间戳计数器值不会在接收或发送消息后复位 1：时间同步使能，时间戳计数器值在接收或发送消息后复位
3	**BUFFLOCK**	**缓冲区锁定位,使能后buffer在成功接收后锁定buffer. 通过写RX_READY到缓冲状态寄存器解锁。** 0：禁止缓冲区锁定功能 1：允许缓冲区锁定功能
2	**CTX**	**发送电平配置** 0：显性电平为0，隐性电平为1 1：显性电平为1，隐性电平为0
1	**CRX**	**接收电平配置** 0：显性电平为0，隐性电平为1 1：显性电平为1，隐性电平为0
0	**CANEN**	**CAN模块使能允许位** 0：CAN模块失能 1：CAN模块使能

24.10.10 CAN 时序寄存器 (CAN_CTIM)

偏移地址: 0x0204

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
PSC[6:0]							SJW[1:0]		TSEG1[3:0]				TSEG2[2:0]		
RW							*RW*		*RW*				*RW*		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 9	**PSC[6:0]**	预分频设置 0000000：2 0000001：3 0000010：4 0000011：5 … 1111101：127 1111110：128 1111111：128
8 ~ 7	**SJW[1:0]**	同步跳转宽度 00：1tq 01：2tq 10：3tq 11：4tq
6 ~ 3	**TSEG1[3:0]**	相位1 0000：不推荐 0001：2tq 0010：3tq 0011：4tq 0100：5tq 0101：6tq 0110：7tq 0111：8tq 1000：9tq 1001：10tq 1010：11tq 1011：12tq 1100：13tq 1101：14tq 1110：15tq 1111：16tq

2 ~ 0	**TSEG2[2:0]**	**相位2** 000：1tq 001：2tq 010：3tq 011：4tq 100：5tq 101：6tq 110：7tq 111：8tq

24.10.11 CAN 全局屏蔽寄存器 (CAN_GMSKX)

偏移地址: 0x0208

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
XGM[14:0]															XRTR
RW															*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 1	**XGM[14:0]**	**扩展帧ID14~ID0全局屏蔽位** 0：此输入ID位对应的消息ID位必须与消息中的相应ID位一致 1：此输入ID位对应的消息ID位允许1或0
0	**XRTR**	**扩展远程帧屏蔽位** 0：需匹配ID中对应位 1：忽略ID中对应位

24.10.12 CAN 全局屏蔽寄存器 (CAN_GMSKB)

偏移地址: 0x020C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
GMXGM[28:18]											RTRSRR	IDE	XGM[17:15]		
RW											*RW*	*RW*	*RW*		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 5	**GMXGM[28:18]**	**标准帧时ID10~ID0局屏蔽位，扩展帧时ID28~ID18全局屏蔽位** 0: 此输入ID位对应的消息ID位必须与消息中的相应ID位一致 1：此输入ID位对应的消息ID位允许1或0
4	**RTRRSS**	**标准帧时远程帧屏蔽位，扩展帧时远程替代位屏蔽位** 0：需匹配ID中对应位 1：忽略ID中对应位
3	**IDE**	**扩展帧屏蔽位** 0：需匹配ID中对应位 1：忽略ID中对应位
2 ~ 0	**XGM[17:15]**	**扩展帧ID17~ID15全局屏蔽位** 0: 此输入ID位对应的消息ID位必须与消息中的相应ID位一致 1：此输入ID位对应的消息ID位允许1或0）

24.10.13 CAN 基本屏蔽寄存器 (CAN_BMSKX)

偏移地址: 0x0210

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
XBM[14:0]															XRTR
RW															*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 1	**XBM[14:0]**	**扩展帧ID14~ID0屏蔽位 (仅对缓冲区14有效)** 0: 此输入ID位对应的消息ID位必须与消息中的相应ID位一致 1：此输入ID位对应的消息ID位允许1或0
0	**XRTR**	**扩展远程帧屏蔽位(仅对缓冲区14有效)** 0：需匹配ID中对应位 1：忽略ID中对应位

24.10.14 CAN 基本屏蔽寄存器 (CAN_BMSKB)

偏移地址: 0x0214

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
BMXBM[28:18]											RTRSRR	IDE	XBM[17:15]		
RW											*RW*	*RW*	*RW*		
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 5	**BMXBM[28:18]**	**标准帧时ID10~ID0局屏蔽位，扩展帧时ID28~ID18全局屏蔽位 (仅对缓冲区14有效)** 0: 此输入ID位对应的消息ID位必须与消息中的相应ID位一致 1：此输入ID位对应的消息ID位允许1或0
4	**RTRSRR**	**标准帧时远程帧屏蔽位，扩展帧时远程替代位屏蔽位 (仅对缓冲区14有效)** 0：需匹配ID中对应位 1：忽略ID中对应位
3	**IDE**	**扩展帧屏蔽位(仅对缓冲区14有效)** 0：需匹配ID中对应位 1：忽略ID中对应位
2 ~ 0	**XBM[17:15]**	**扩展帧ID17~ID15全局屏蔽位(仅对缓冲区14有效)** 0: 此输入ID位对应的消息ID位必须与消息中的相应ID位一致 1：此输入ID位对应的消息ID位允许1或0

24.10.15 CAN 中断使能寄存器 (CAN_CIEN)

偏移地址: 0x0218

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
EIEN	**IEN14**	**IEN13**	**IEN12**	**IEN11**	**IEN10**	**IEN9**	**IEN8**	**IEN7**	**IEN6**	**IEN5**	**IEN4**	**IEN3**	**IEN2**	**IEN1**	**IEN0**
RW	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15	**EIEN**	**错误中断使能位** 0：禁止错误中断 1：允许错误中断，且当TEC或REC有变化时会产生一个错误中断
14	**IEN14**	**缓冲区14中断使能位** 0：禁止中断 1：允许中断
13	**IEN13**	**缓冲区13中断使能位** 0：禁止中断 1：允许中断
12	**IEN12**	**缓冲区12中断使能位** 0：禁止中断 1：允许中断
11	**IEN11**	**缓冲区11中断使能位** 0：禁止中断 1：允许中断
10	**IEN10**	**缓冲区10中断使能位** 0：禁止中断 1：允许中断

9	**IEN9**	**缓冲区9中断使能位** 0：禁止中断 1：允许中断
8	**IEN8**	**缓冲区8中断使能位** 0：禁止中断 1：允许中断
7	**IEN7**	**缓冲区7中断使能位** 0：禁止中断 1：允许中断
6	**IEN6**	**缓冲区6中断使能位** 0：禁止中断 1：允许中断
5	**IEN5**	**缓冲区5中断使能位** 0：禁止中断 1：允许中断
4	**IEN4**	**缓冲区4中断使能位** 0：禁止中断 1：允许中断
3	**IEN3**	**缓冲区3中断使能位** 0：禁止中断 1：允许中断
2	**IEN2**	**缓冲区3中断使能位** 0：禁止中断 1：允许中断
1	**IEN1**	**缓冲区1中断使能位** 0：禁止中断 1：允许中断
0	**IEN0**	**缓冲区0中断使能位** 0：禁止中断 1：允许中断

24.10.16 CAN 中断标志寄存器 (CAN_CIPND)

偏移地址: 0x021C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
EIPND	**IPND14**	**IPND13**	**IPND12**	**IPND11**	**IPND10**	**IPND9**	**IPND8**	**IPND7**	**IPND6**	**IPND5**	**IPND4**	**IPND3**	**IPND2**	**IPND1**	**IPND0**
RO	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15	**EIPND**	**错误中断挂起标志位** 0：TEC/REC错误状态没有改变 1：TEC/REC错误状态发生改变，产生错误中断
14	**IPND14**	**缓冲区14中断挂起标志位** 0：没有中断挂起 1：有中断挂起
13	**IPND13**	**缓冲区13中断挂起标志位** 0：没有中断挂起 1：有中断挂起
12	**IPND12**	**缓冲区12中断挂起标志位** 0：没有中断挂起 1：有中断挂起
11	**IPND11**	**缓冲区11中断挂起标志位** 0：没有中断挂起 1：有中断挂起
10	**IPND10**	**缓冲区10中断挂起标志位** 0：没有中断挂起 1：有中断挂起
9	**IPND9**	**缓冲区9中断挂起标志位** 0：没有中断挂起 1：有中断挂起
8	**IPND8**	**缓冲区8中断挂起标志位** 0：没有中断挂起 1：有中断挂起
7	**IPND7**	**缓冲区7中断挂起标志位** 0：没有中断挂起 1：有中断挂起
6	**IPND6**	**缓冲区6中断挂起标志位** 0：没有中断挂起 1：有中断挂起
5	**IPND5**	**缓冲区5中断挂起标志位** 0：没有中断挂起 1：有中断挂起
4	**IPND4**	**缓冲区4中断挂起标志位** 0：没有中断挂起 1：有中断挂起
3	**IPND3**	**缓冲区3中断挂起标志位** 0：没有中断挂起 1：有中断挂起
2	**IPND2**	**缓冲区2中断挂起标志位** 0：没有中断挂起 1：有中断挂起
1	**IPND1**	**缓冲区1中断挂起标志位** 0：没有中断挂起 1：有中断挂起
0	**IPND0**	**缓冲区0中断挂起标志位** 0：没有中断挂起 1：有中断挂起

24.10.17 CAN 中断标志位清除寄存器 (CAN_CICLR)

偏移地址: 0x0220

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
EICLR	ICLR14	ICLR13	ICLR12	ICLR11	ICLR10	ICLR9	ICLR8	ICLR7	ICLR6	ICLR5	ICLR4	ICLR3	ICLR2	ICLR1	ICLR0
WO	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15	**EICLR**	**错误中断清除位** 0：不改变错误挂起位EIPND 1：复位错误挂起位EIPND
14	**ICLR14**	**缓冲区14中断清除位** 0：不改变中断挂起位 1：复位挂起位
13	**ICLR13**	**缓冲区13中断清除位** 0：不改变中断挂起位 1：复位挂起位
12	**ICLR12**	**缓冲区12中断清除位** 0：不改变中断挂起位 1：复位挂起位
11	**ICLR11**	**缓冲区11中断清除位** 0：不改变中断挂起位 1：复位挂起位
10	**ICLR10**	**缓冲区10中断清除位** 0：不改变中断挂起位 1：复位挂起位
9	**ICLR9**	**缓冲区9中断清除位** 0：不改变中断挂起位 1：复位挂起位
8	**ICLR8**	**缓冲区8中断清除位** 0：不改变中断挂起位 1：复位挂起位
7	**ICLR7**	**缓冲区7中断清除位** 0：不改变中断挂起位 1：复位挂起位
6	**ICLR6**	**缓冲区6中断清除位** 0：不改变中断挂起位 1：复位挂起位
5	**ICLR5**	**缓冲区5中断清除位** 0：不改变中断挂起位 1：复位挂起位
4	**ICLR4**	**缓冲区4中断清除位** 0：不改变中断挂起位 1：复位挂起位

3	**ICLR3**	**缓冲区3中断清除位** 0：不改变中断挂起位 1：复位挂起位
2	**ICLR2**	**缓冲区2中断清除位** 0：不改变中断挂起位 1：复位挂起位
1	**ICLR1**	**缓冲区1中断清除位** 0：不改变中断挂起位 1：复位挂起位
0	**ICLR0**	**缓冲区0中断清除位** 0：不改变中断挂起位 1：复位挂起位

24.10.18 CAN 中断码使能寄存器 (CAN_CICEN)

偏移地址: 0x0224

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
EICEN	ICEN14	ICEN13	ICEN12	ICEN11	ICEN10	ICEN9	ICEN8	ICEN7	ICEN6	ICEN5	ICEN4	ICEN3	ICEN2	ICEN1	ICEN0
RW	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15	**EICEN**	**错误中断码使能位** 0：错误中断挂起不显示在中断码寄存器（CSTPND）中 1：错误中断挂起显示在中断码寄存器（CSTPND）中
14	**ICEN14**	**缓冲区14中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
13	**ICEN13**	**缓冲区13中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
12	**ICEN12**	**缓冲区12中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
11	**ICEN11**	**缓冲区11中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
10	**ICEN10**	**缓冲区10中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
9	**ICEN9**	**缓冲区9中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中

8	**ICEN8**	**缓冲区8中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
7	**ICEN7**	**缓冲区7中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
6	**ICEN6**	**缓冲区6中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
5	**ICEN5**	**缓冲区5中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
4	**ICEN4**	**缓冲区4中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
3	**ICEN3**	**缓冲区3中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
2	**ICEN2**	**缓冲区2中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
1	**ICEN1**	**缓冲区1中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中
0	**ICEN0**	**缓冲区0中断码使能位** 0：中断挂起不显示在中断码寄存器(CSTPND)中 1：中断挂起显示在中断码寄存器(CSTPND)中

24.10.19 CAN 状态挂起寄存器 (CAN_CSTPND)

偏移地址: 0x0228

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								NS[2:0]			IRQ	IST[3:0]			
-								*RO*			*RO*	*RO*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 8	保留	-
7 ~ 5	**NS[2:0]**	**CAN节点状态** 000：未激活 010：错误激活状态 011：错误警告 10X：被动错误 11X：总线关闭

4	**IRQ**	**中断请求位** 0：无中断请求 1：有中断请求
3 ~ 0	**IST[3:0]**	**最高优先级中断码** **当IRQ=0时表示无中断请求，当IRQ=1时中断码如下** 0000：错误中断 0001：buff0 0010：buff1 0011：buff2 0100：buff3 0101：buff4 0110：buff5 0111：buff6 1000：buff7 1001：buff18 1010：buff19 1011：buff10 1100：buff11 1101：buff12 1110：buff13 1111：buff14

24.10.20 CAN 错误计数器寄存器 (CAN_CANEC)

偏移地址: 0x022C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
REC[7:0]								TEC[7:0]							
RW								*RW*							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 8	**REC[7:0]**	接收错误计数器
7 ~ 0	**TEC[7:0]**	发送错误计数器

24.10.21 CAN 错误诊断寄存器 (CAN_CEDIAG)

偏移地址: 0x0230

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留	DRIVE	MON	CRCERR	STUFF	TXE	EBID[5:0]						EFID[3:0]			
-	*RO*	*RO*	*RO*	*RO*	*RO*	*RO*						*RO*			
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 15	保留	-
14	**DRIVE**	出错时**CANTX**引脚上的输出电平。（注：接收者不会驱动总线，除了在**ACK**及置错误标志期间）
13	**MON**	出错时**CANRX**引脚上的输入电平
12	**CRCERR**	**CRC**校验状态位 0：没有发生CRC校验错误 1：发生了CRC校验错误
11	**STUFF**	位填充错误标志位 0：没有发生位填充错误 1：发生了位填充错误
10	**TXE**	发送错误 0：错误发生时CAN模块作为接收方 1：错误发生时CAN模块作为发送方
9 ~ 4	**EBID[5:0]**	出错位的位置
3 ~ 0	**EFID[3:0]**	错误状态 0000：错误 0001：错误删除 0010：错误回显 0011：总线空闲 0100：响应 0101：帧结束 0110：间歇 0111：发送挂起 1000：帧开始 1001：仲裁 1010：扩展帧 1011：扩展仲裁 1100：R0/R1 1101：数据长度 1110：数据帧 1111：校验

24.10.22 CAN 定时器寄存器 (CAN_CTMR)

偏移地址: 0x0234

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留															
-															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
CTMR[15:0]															
RW															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	保留	-
15 ~ 0	**CTMR[15:0]**	当前时间戳计数值

25. 器件电子签名

产品唯一身份标识寄存器（96 位），非常适合如下应用：

- 用来作为序列号(例如产品序列号满足终端应用)
- 用来作为密码，在编写闪存时，将此唯一标识与软件加解密算法结合使用，提高代码在闪存存储器内的安全性。
- 用来激活带安全机制的自举过程

96 位的产品唯一身份标识所提供的参考号码对任意一个 SH33F2801 微控制器，在任何情况下都是唯一的。用户在何种情况下，都不能修改这个身份标识。

这个 96 位的产品唯一身份标识，按照用户不同的用法，可以以字节(8 位)为单位读取，也可以以半字(16 位)或者全字(32 位)。

96 位识别码的地址是 0x0FFF F910 - 0x0FFF F91B，建议使用全部 96bit 作为产品唯一 ID。

26. 调试接口

26.1 主要特性

SH33F2801 使用 ARM “星辰”处理器，该内核内含硬件调试模块，支持复杂的调试操作。硬件调试模块允许内核在取指(指令断点)或访问数据(数据断点)时停止。内核停止时，内核的内部状态和系统的外部状态都是可以查询的。完成查询后，内核和外设可以被复原，程序将继续执行。

SH33F2801 支持 SW 调试接口，当 SH33F2801 微控制器连接到调试器并开始调试时，调试器将使用内核的硬件调试模块进行调试操作。

ARM “星辰”处理器提供集成的片上调试功能。它由以下部分组成：

- SW-DP：串行调试端口
- AHP-AP：AHB 访问端口
- FPB：闪存指令断点
- DWT：数据触发

有关调试接口的更多信息请参考ARM官方手册，以及ARM开发工具集技术参考手册。

26.2 SW 调试端口(Serial Wire)

SH33F2801 内核集成了串行调试接口(SW-DP)。这是标准的 ARM CoreSight 调试接口的精简版，串行调试接口(SW-DP)为 AHP-AP 模块提供 2 针(时钟＋数据)接口。

26.3 SW 调试端口引脚

SH33F2801 的 2 个普通 I/O 口可用作 SW-DP 接口引脚。这些引脚在所有的封装里都存在。

表26-1 SW 调试端口引脚

SW-DP 端口引脚名称	SW 调试接口		引脚分配
	类型	调试功能	
SWDIO	输入/输出	串行数据输入/输出	PA7
SWCLK	输入	串行时钟	PA8

26.4 利用串行接口并释放不用的调试脚作为普通 I/O 口

复位(SYSRESET 或 PORESET)后，属于 SW-DP 的所有个引脚都立即被初始化为可被调试器使用的专用引脚。 然而，SH33F2801 微控制器可以把 SW 接口再复用为 GPIO 口，详细内容请参考“SYSCFG”章节相关介绍。

对于用户软件设计，应注意在复位后，这些专用引脚仍然处于带上拉的输入(SWDIO)，带下拉的输入(SWCLK)，并持续一段时间，直到用户代码释放这些引脚。

表26-2 SW_DP 引脚分配

SWJCFG[0]	配置为调试专用的引脚	SWD 接口的 I/O 口分配		SW 调试接口是否可用
		SWDIO	SWCLK	
0	SW-DP 引脚复位状态	调试专用	调试专用	可用
1	SW-DP 接口禁止	用户可用（注 1）	用户可用（注 1）	可用

注 1：当在调试模式下，SWDIO 和 SWCLK 引脚仍被配置成调试接口，且修改相应的普通 I/O 口配置寄存器是无效的。

26.5 MCU 调试模块

MCU 调试模块协助调试器提供以下功能：

- 低功耗模式
- 在断点时提供 MCM、PCA、TIM、UART、SPI、TWI、IWDT、WDT、CAN 的时钟控制

26.5.1 低功耗模式的调试支持

使用 WFI 和 WFE 可以进入低功耗模式。MCU 支持多种低功耗模式，分别可以关闭 CPU 时钟，或降低 CPU 的能耗。但为使用户在低功耗模式下调试代码，那在调试期间就不能关闭 FCLK 和 HCLK，即设置 DBG_STOP=0，因此在调试模式下进入低功耗模式后，FCLK 和 HCLK 默认是工作的。 如果用户需要在调试期间进入真正的低功耗模式，可以通过调试器或软件设置 DBG_STOP=1。

注意，DBG_STOP=0 时只是保证在调试模式下进 STOP 后可以读取各外设的寄存器，但外设的工作时钟仍然是关闭的，另外，这种情况下 HCLK 由内部 HSI 提供时钟。

26.5.2 带定时功能的模块调试支持

在产生断点时，有必要根据各模块的不同用途选择相应计数器的工作模式，对于 MCM、PCA、TIM、UART、SPI、TWI、IWDT、WWDT、CAN 各个模块，在产生断点时，可选择各模块的计数器是否继续计数。

26.6 寄存器

SYSCFG 模块寄存器列表 (基地址：0x4004 0800)

地址	寄存器名	说明
0x4004 0808	SAFR	系统配置寄存器
0x4004 0810	DBGCR	调试接口控制寄存器

26.6.1 系统配置寄存器 (SYSCFG_SAFR)

偏移地址: 0x0008

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
LOCK[15:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
保留								IEN_CSM	IEN_BOD	IEN_EXTI0	保留		SWJCFG	OSCCFG[1:0]	
-								*RW*	*RW*	*RW*	-		*RW1t*	*RW1t*	
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**LOCK[15:0]**	**此寄存器操作解锁位 (Control bit Lock register)** 这些位是只写，读取这些位返回0x00。 0x5AA5： 解锁 其他： 锁定 注：写入此寄存器需配合解锁控制，读此寄存器不需要解锁。
15 ~ 8	保留	**-**
7	**IEN_CSM**	**CSM引起NMI中断使能位**

6	**IEN_BOD**	**BOD引起NMI中断使能位**
5	**IEN_EXTI0**	**EXTI0引起NMI中断使能位**
4 ~ 3	保留	**-**
2	**SWJCFG**	**串行线引脚复用功能定义** 这些位只可由软件写(读这些位，将返回未定义的数值)，用于配置SW和跟踪复用功能的I/O口。系统复位后的默认状态是启用SW但没有跟踪功能，这种状态下可以通过SWCLK和SWDIO脚上的特定信号选择SWD模式。为释放部分或全部调试口用做普通I/O口，用户软件在复位后可设置SWJCFG[2:0]。 注：SWD包含SWDIO/SWCLK引脚，而SWO为ITM跟踪输出。 0：SWDIO/SWCLK引脚为调试端口，不复用为普通I/O； 1：关闭SW调试功能；SWCLK、SWDIO可复用为普通I/O。但SWCLK、SWDIO仅在正常运行状态下可复用为普通I/O，在调试模式状态下仍强制作为调试接口。 注：此控制位一旦设置，只有reset才能修改。 用户需谨慎配置该位，因部分选项会导致调试模式需在Reset信号配合下才能进入。
1 ~ 0	**OSCCFG[1:0]**	**XTAL1/XTAL2引脚复用功能定义** 00： XTAL1/XTAL2作为GPIO使用（默认） 01：外部振荡器接口（晶振和陶振） 10：XTAL1上外灌时钟，XTAL2作为GPIO使用 11：保留 注：此控制位一旦设置，只有reset才能修改；

26.6.2 调试接口控制寄存器 (SYSCFG_DBGCR)

偏移地址: 0x0010

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
LOCK[15:0]															
WO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

b15	b14	b13	b12	b11	b10	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0
DBG_TWI	**DBG_SPI**	**DBG_UART**	**DBG_MCM**	**DBG_TIM**	**DBG_PCA**	**DBG_WWDT**	**DBG_IWDT**	**DBG_QEI**	**DBG_CAN**	**保留**				**DBG_STOP**	**保留**
RW	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	-				*RW*	-
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	**LOCK[15:0]**	**此寄存器操作解锁位 (Control bit Lock register)** 这些位是只写，读取这些位返回0x0000。 0x5AA5： 解锁 其他: 锁定 注：写入此寄存器需配合解锁控制，读此寄存器不需要解锁。

15	**DBG_TWI**	**当内核进入调试状态时TWI停止工作** 0：TWI停止工作 1：TWI仍然正常工作
14	**DBG_SPI**	**当内核进入调试状态时SPI停止工作** 0：SPI停止工作 1：SPI仍然正常工作
13	**DBG_UART**	**当内核进入调试状态时UART停止工作** 0：UART停止工作 1：UART仍然正常工作
12	**DBG_MCM**	**当内核进入调试状态时MCM停止工作** 0：MCM停止工作 1：MCM仍然正常工作
11	**DBG_TIM**	**当内核进入调试状态时TIM7-TIM8模块停止工作** 0：TIM7-TIM8模块停止工作 1：TIM7-TIM8模块仍然正常工作
10	**DBG_PCA**	**当内核进入调试状态时PCA停止工作** 0：PCA停止工作 1：PCA仍然正常工作
9	**DBG_WWDT**	**当内核进入调试状态时调试窗口看门狗停止工作。** 0：窗口看门狗计数器停止工作 1：窗口看门狗计数器仍然正常工作
8	**DBG_IWDT**	**当内核进入调试状态时看门狗停止工作** 0：看门狗计数器停止工作 1：看门狗计数器仍然正常工作
7	**DBG_QEI**	**当内核进入调试状态时QEI停止工作** 0：QEI停止工作 1：QEI仍然正常工作
6	**DBG_CAN**	**当内核进入调试状态时CAN停止工作** 0：CAN停止工作 1：CAN仍然正常工作
5 ~ 2	保留	-
1	**DBG_STOP**	**调试停机模式。** 0：(FCLK开，HCLK开)在停机模式时，FCLK和HCLK时钟由内部RC振荡器提供。当退出停机模式时，软件必需重新配置时钟系统启动PLL，晶振等(与配置此比特位为0时的操作一样) 1：(FCLK关，HCLK关)在停机模式时，时钟控制器禁止一切时钟(包括HCLK和FCLK)。当从停机模式退出时，时钟的配置和复位之后的配置一样(微控制器由8MHz的内部RC振荡器(HSI)提供时钟)。因此，软件必需重新配置时钟控制系统启动PLL，晶振等 注：DBG_STOP=0时HCLK保持开启，但外部设备时钟被屏蔽，即外设模块仍然进入了STOP状态。HCLK可维持调试接口的连接，即调试器可读取停机前的CPU和各外设的状态信息。另外这种状态下FCLK和HCLK只由HSI提供。
0	保留	-

27. 用户代码选项

OP_IWDT：

0101：软件控制看门狗（默认）
其他：看门狗常开，无法关闭

OP_WDTPD：

0：停机模式下禁止独立看门狗工作
1：停机模式下允许独立看门狗工作（默认）

OP_MCMPIN：

MCM模块 PWM波形输出顺序：

引脚 / 选项	PB6	PB5	PB4	PB3	PB2	PB1
000（默认）	PWM无输出					
001	PWM0	PWM1	PWM2	PWM01	PWM11	PWM21
010	PWM21	PWM11	PWM01	PWM2	PWM1	PWM0
011	PWM0	PWM01	PWM1	PWM11	PWM2	PWM21
1XX	PWM21	PWM2	PWM11	PWM1	PWM01	PWM0

FLASH_BACKUP：

0：FLASH区不分区块（默认）
1：FLASH区分为两块，Main Memory以及Backup Memory

FLASH_SELECT：（只有当FLASH_BACKUP为1时有效）

0：PC指针从Main Memory区取址（默认）
1：PC指针从Backup Memory区取址

OP_ISP:

0：使能ISP（默认）
1：禁止ISP

28. 电气特性

直流供电电压. . . . . –0.3V to +6.0V
输入/输出电压. . . . . GND-0.3V to VDD+0.3V
工作环境温度. . . . . .-40℃ to +105℃
存储温度. . . . . .–55℃ to +125℃
FLASH存储器写/擦除操作 . . . . . 0℃ to +85℃

***注释**

如果器件的工作条件超过左列“极限参数”的范围，将造成器件永久性破坏。只有当器件工作在说明书所规定的范围内时功能才能得到保障。器件在极限参数列举的条件下工作将会影响到器件工作的可靠性。

直流电气特性 (VDD = 2.4 – 5.5V, TA = 25°C, 除非另有说明)

参数	符号	最小值	典型值	最大值	单位	条件
数字工作电压	V_{DD}	2.4	5.0	5.5	V	4MHz ≤ f_{sys} ≤ 84MHz
工作电流	I_{OP1}	-	20	35	mA	f_{sys} = **84MHz**，PLL开，V_{DD}= 5.0V，PLL时钟源使用晶振（8MHz）。所有输出引脚无负载，所有输入引脚不浮动 CPU打开(执行本地循环跳转)，IWDT打开，LVR打开，关闭所有外设时钟
	I_{OP2}	-	6	10	mA	f_{sys} = 8MHz，PLL关，V_{DD}=5.0V，使用内部RC振荡器。所有输出引脚无负载，所有输入引脚不浮动 CPU打开(执行本地循环跳转)，IWDT打开，LVR打开，关闭所有外设时钟
	I_{OP3}	-	35	45	mA	f_{sys} = **84MHz**，PLL开，V_{DD}= 5.0V，PLL时钟源使用晶振（8MHz）。所有输出引脚无负载，所有输入引脚不浮动 CPU打开(执行本地循环跳转)，IWDT打开，LVR打开，**开启所有外设时钟**
	I_{OP4}	-	8	12	mA	f_{sys} = 8MHz，PLL关，V_{DD}= 5.0V，使用内部RC振荡器。所有输出引脚无负载，所有输入引脚不浮动 CPU打开(执行本地循环跳转)，IWDT打开，LVR打开，**开启所有外设时钟**
待机电流 (睡眠模式)	I_{SB1}	-	6	10	mA	f_{sys} = 8MHz，PLL关，V_{DD}= 5.0V，使用内部RC振荡器。所有输出引脚无负载，所有输入引脚不浮动 CPU停止((执行WFI进入sleep)，IWDT关闭，LVR打开，关闭所有外设时钟
待机电流 (停机模式)	I_{SB2}	-	10	20	μA	V_{DD}= 5.0V，所有输出引脚无负载，所有输入引脚不浮动，CPU停止(执行WFI进入stop)，IWDT关闭，LVR关闭，关闭其它所有功能
LVR电流	I_{LVR1}	-	1	-	μA	LVR打开，LVR电平 = 4.1V/2.8V
IWDT 电流	I_{IWDT}	-	-	1	uA	所有输出引脚无负载；独立看门狗打开 V_{DD} =5.0V
输入低电压1	V_{IL1}	GND		0.2 X V_{DD}	V	I/O 端口（所有端口都是施密特输入） V_{DD} = 2.4 - 5.5V
输入高电压1	V_{IH1}	0.8 X V_{DD}	-	V_{DD}	V	I/O 端口（所有端口都是施密特输入） V_{DD} = 2.4 - 5.5V
输入低电压2	V_{IL2}	GND	-	0.15 X V_{DD}		仅NRST引脚 V_{DD} = 2.4 – 5.5V
输入高电压2	V_{IH2}	0.85 X V_{DD}	-	V_{DD}		仅NRST引脚 V_{DD} = 2.4 – 5.5V

输入低电压3	V_{IL3}	GND	-	0.8	V	仅表13-3引脚具备（输入高低压窗口0.4V） V_{DD} = 4.5 - 5.5V，TTL输入功能打开
		GND	-	0.15 X V_{DD}	V	仅表13-3引脚具备（输入高低压窗口0.4V） V_{DD} = 2.4 - 4.5V，TTL输入功能打开
输入高电压3	V_{IH3}	2.0	-	V_{DD}	V	仅表13-3引脚具备（输入高低压窗口0.4V） V_{DD} = 4.5 - 5.5V，TTL输入功能打开
		0.25 X V_{DD}+0.8	-	V_{DD}	V	仅表13-3引脚具备（输入高低压窗口0.4V） V_{DD} = 2.4 - 4.5V，TTL输入功能打开
输入漏电流	I_{IL}	-1		1	uA	输入无上拉, V_{IN}= V_{DD} or GND
上拉电阻*	R_{PH1}	-	30	-	kΩ	V_{DD} =5.0V, V_{IN}=GND
复位引脚上拉电阻	R_{RPH}	-	10	-	kΩ	V_{DD} = 5.0V，V_{IN} = GND
输出高电压	V_{OH}	V_{DD} – 0.7	-	-	V	I/O端口, I_{OH} = -10mA, V_{DD} = 5.0V
输出低电压	V_{OL}	-	-	GND + 0.6	V	I/O 端口, I_{OL} = 12mA, V_{DD} = 5.0V

注意：

1. ""表示典型值下的数据是在5.0V，25°C下测得的，除非另有说明。*

2. 流过V_{DD}的最大电流值须小于150mA。流过GND的最大电流值须小于200mA。

3. 设计保证，实际测试时未包含温度。.

高速12BIT模/数转换器电气特性（1LSB = V_{DD}/4096）

转换速率最高2MSPS。（V_{DD} = 2.7V～5.5V，GND = 0V，T_A = +25°C，除非另有说明）

参数	符号	最小值	典型值	最大值	单位	条件
工作电压范围	V_{AD}	2.7	5.0	5.5	V	
精度	N_R	-	12	-	bit	V_{REF} = 5.0V
A/D输入电压	V_{AIN}	GND	-	V_{REF}	V	
A/D等效输入电阻	R_{AIN}	2	-	-	MΩ	V_{IN} = 5.0V
外部模拟参考电压	V_{REF}	2.5	-	V_{DD}	V	
ADC采样信号源推荐阻抗	Z_{AIN}	-	2.5	-	kΩ	典型值以 1MSPS 转换速率，误差 2LSB 测量，具体参考“章节 20.3.10”公式
		-	-	1	MΩ	采样速率越低，支持阻抗越大，在 1KSPS 转换速率，误差 1LSB 测量
A/D转换电流	I_{AD}	-	1.5	3	mA	ADC模块工作，V_{DD} = 5.0V
微分非线性误差	D_{LE}	-	±0.5	±2	LSB	V_{DD} = 5.0V，V_{REF} = V_{DD}，ADC CLK = 40MHz
积分非线性误差	I_{LE}	-	±1	±3	LSB	V_{DD} = 5.0V，V_{REF} = V_{DD}，ADC CLK = 40MHz
满刻度误差	E_F	-	±3	±5	LSB	V_{DD} = 5.0V，V_{REF} = V_{DD}，ADC CLK = 40MHz
偏移量误差	E_Z	-	±3	±7	LSB	V_{DD} = 5.0V，V_{REF} = V_{DD}，ADC CLK = 40MHz
总绝对误差	E_{AD}	-	-	±8	LSB	V_{DD} = 5.0V，V_{REF} = V_{DD}，ADC CLK = 40MHz
ADC工作时钟	f_{ADC}	1	-	40	MHz	V_{DD} = 5.0V，V_{REF} = V_{DD}
			-	30		V_{DD} = 3.3V，V_{REF} = V_{DD}
ADC采样时间	t_{SAMP}	0.125	-	-	μs	V_{DD} = 5.0V，V_{REF} = V_{DD}
		0.2	-	-		V_{DD} = 3.3V，V_{REF} = V_{DD}
ADC转换速率	F_{CON}	-	-	2	MSPS	V_{DD} = 5.0V，V_{REF} = V_{DD}
		-	-	1.4		V_{DD} = 3.3V，V_{REF} = V_{DD}
总转换时间	T_{CON}	16	-	159	t_{AD}	（1 t_{AD} ～ 16 t_{AD}）+t_{GAP} + t_{COMP}

比较器0/1电气特性（V_{DD} = 2.7 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明）

参数	符号	最小值	典型值	最大值	单位	条件
输入失调电压	V_{IO}	-	2	3	mV	
输入共模电压范围	V_{ICM}	0	-	V_{DD} - 1.2	V	
小信号响应时间	T_{RS1}	-	1	2	μs	V_{DD} = 5V，CMPxN = 1.20V，CMPxP = 1.15V Step to 1.25V
大信号响应时间	T_{RS2}	-	0.3	0.5	μs	V_{DD} = 5V，CMPxN = 1V，CMPxP = 0V Step to 2V
比较器施密特窗口1	V_{SMT1}	-	0	-	mV	V_{DD} = 5V，无施密特窗口，下降沿窗口
比较器施密特窗口2	V_{SMT2}	-	15	20	mV	V_{DD} = 5V，施密特窗口10mV，下降沿窗口
比较器施密特窗口3	V_{SMT3}	-	30	40	mV	V_{DD} = 5V，施密特窗口20mV，下降沿窗口
比较器施密特窗口4	V_{SMT4}	-	50	70	mV	V_{DD} = 5V，施密特窗口50mV，下降沿窗口
内部反相端参考源精度	V_{N1}	-	-	3	mV	V_{DD} = 5V, 内部N端电压小于300mV
		-	-	1%	-	V_{DD} = 5V, 内部N端电压大于300mV

放大器0/1/2/3电气特性（V_{DD} = 2.7 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明）

参数	符号	最小值	典型值	最大值	单位	条件
输入失调电压	V_{IO}	-	2	3	mV	T_A = 25°C
输入共模电压	V_{ICM}	0	-	V_{DD} - 1.2	V	
温度系数		-	0.015%	-		V_{OUT} = 2.5V，温度变每化1°C，输出电压的变化率
转换速率	SR	5	-	-	V/us	电压跟随器模式
输出电压范围	V_{OUR}	0.1	-	V_{DD} - 0.5	V	
电压抑制比	SVR	65	80	-	dB	直流特性
共模抑制比	CMRR	60	80	-	dB	直流特性
增益带宽	BW	-	12	-	MHz	电压跟随器模式
内部增益系数精度	Gain1	-	0.5%	1%	-	VDD=5V, 放大倍数1-16倍
	Gain2	-	1%	2%	-	VDD=5V, 放大倍数24、32倍

交流电气特性（V_{DD} = 2.4 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明）

参数	符号	最小值	典型值	最大值	单位	条件
复位脉冲宽度	t_{RESET}	100	-	-	µs	低电平有效
频率精度 (RC)	|Δ F|/F	-	-	0.2	%	内部RC振荡器频率精度:（1024个周期求平均值） $\|F_{RC}$ – 8MHz|/8MHz X 100% （V_{DD} = 2.4 – 5.5V，T_A = 25°C）
	|Δ F|/F	-	-	0.5	%	内部RC振荡器频率精度:（1024个周期求平均值） $\|F_{RC}$ – 8MHz|/8MHz X 100% （V_{DD} = 2.4 - 5.5V，T_A = -10°C至70°C， 设计保证，不在生产中测试）
	|Δ F|/F	-	-	1	%	内部RC振荡器频率精度:（1024个周期求平均值） $\|F_{RC}$ – 8MHz|/8MHz X 100% （V_{DD} = 2.4 - 5.5V，T_A = -40°C至+105°C， 设计保证，不在生产中测试）

PLL特性（V_{DD} = 2.4 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明）

参数	符号	最小值	典型值	最大值	单位	条件
PLL输入时钟频率	f_{PLL_IN}	4	-	12	MHz	
PLL输入时钟Duty	D_{PLL_IN}	40	-	60	%	
PLL输出时钟频率	f_{PLL_OUT}			84	MHz	
PLL建立时间	T_{PLL}		2^{17}		T_{OSC}	
PLL频率稳定性			±0.05	±0.1	%	（1024个周期求平均值 V_{DD} = 2.4 - 5.5V，T_A = 25°C）

FLASH特性（VDD = 2.4 - 5.5V，GND = 0V，TA = +25°C，除非另有说明）

参数	符号	最小值	典型值	最大值	单位	条件
读操作速度	T_{READ}	-	30	-	ns	32位读出
烧写操作速度	T_{PROG}	-	20	-	us	32位写入
扇区擦除	T_{SECTOR_ERASE}	-	2	-	ms	单个扇区
整体擦除速度	T_{TOTAL_ERASE}	-	10	-	ms	
编程/擦除次数	N_{END}					
程序区	-	10	-	-	千次	
类EEPROM区	-	100	-	-	千次	
数据保存年限	t_{RET}	10	-	-	年	

掉电检测（BOD）电气特性（V_{DD} = 2.4 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明）

参数	符号	最小值	典型值	最大值	单位	条件
BOD电压范围	V_{BOD}	2.8	-	4.3	V	
BOD门限等级	$\|\Delta V_{BOD}\|$	-	100	-	mV	V_{BOD} = 2.8V～4.3V
BOD电压检测迟滞窗口	V_{SMTBD1}	-	50	-	mV	V_{BOD} = 2.8V～4.3V
BOD去抖动时间	T_{BOD}	-	60	-	μs	

低电压复位（LVR）电气特性（V_{DD} = 2.4 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明）

参数	符号	最小值	典型值	最大值	单位	条件
LVR电压1	V_{LVR1}	4.0	4.1	4.2	V	LVR1使能
LVR电压2	V_{LVR2}	2.7	2.8	2.9	V	LVR2使能
辅助LVR	V_{LVR}	-	2.3	-	V	内部辅助LVR（始终有效）
LVR电压检测迟滞窗口	V_{SMTLV}	-	50	-	mV	
LVR低电压复位宽度	T_{LVR}	-	60	-	μs	

29. 订购信息

产品编号	封装
SH33F2801U/048UR	TQFP48
SH33F2801P/032PR	LQFP32

30. 封装信息

TQFP48外形尺寸

单位: 英寸/毫米

Symbol	Dimensions in inches		Dimensions in mm	
	MIN	MAX	MIN	MAX
A	---	0.047	---	1.2
A1	0.002	0.006	0.05	0.15
A2	0.035	0.041	0.9	1.05
D	0.270	0.281	6.85	7.15
E	0.270	0.281	6.85	7.15
H_D	0.346	0.362	8.8	9.2
H_E	0.346	0.362	8.8	9.2
b	0.005	0.011	0.15	0.27
e	0.020 TYP		0.500 TYP	
c	0.004	0.008	0.090	0.200
L	0.018	0.030	0.45	0.75
L1	0.033	0.045	0.85	1.15
θ2	0°	10°	0°	10°

注意:

1. 封装尺寸不包括模的毛边凸起或门毛刺。
2. 如无特殊规定，容差为±0.1毫米。
3. 共面性：0.1毫米。
4. 控制尺寸为毫米。

Notice:

1.Both package length and width do not include mold flash.
2.Tolerance is ±0.1mm if not specified.
3.Coplanarity:0.1mm max
4.Controlling dimension:mm

LQFP32外形尺寸

单位: 英寸/毫米

Symbol	Dimensions in inches		Dimensions in mm	
	MIN	MAX	MIN	MAX
A	0.057	0.065	1.45	1.65
A1	0.000	0.008	0.01	0.21
A2	0.051	0.059	1.30	1.50
D	0.268	0.281	6.80	7.15
E	0.268	0.281	6.80	7.15
H_D	0.346	0.362	8.80	9.20
H_E	0.346	0.362	8.80	9.20
b	0.010	0.018	0.25	0.45
e	0.031 TYP		0.8TYP	
c	0.004	0.009	0.09	0.22
L	0.016	0.031	0.40	0.78
L1	0.035	0.043	0.90	1.10
θ2	0°	10°	0°	10°

注意:

1. 封装尺寸不包括模的毛边凸起或门毛刺。
2. 如无特殊规定，容差为±0.1毫米。
3. 共面性：0.1毫米。
4. 控制尺寸为毫米。

Notice:

1.Both package length and width do not include mold flash.
2.Tolerance is ±0.1mm if not specified.
3.Coplanarity:0.1mm max
4.Controlling dimension:mm

31. 规格更改记录

版本	记录	日期
1.0	初始版本	2021年10月

重要声明

本手册为中颖电子股份有限公司及其关联公司（“公司”）的财产。本手册，包括本手册中描述的本公司的任何产品（“产品”），均为本公司根据相关可适用法律或条约所拥有。本公司保留该等法律和条约下的所有权利，不授予其专利、版权、商标或其他知识产权下的任何许可。

本手册内的任何技术信息，包括功能介绍和原理图，不应理解为使用或执行任何知识产权的许可。本手册若引述相关第三方的名称和品牌（如有）等为其各自所有者的财产，仅供识别用途。

本公司不对本手册或任何产品作任何明示或暗示的保证，包括但不限于对适销性和适合特定用途的暗示保证。本公司不承担因超出规格书或我司产品标准的保证范围使用本手册所述任何产品而产生的任何责任。除适用协议中明确规定的定制产品外，产品仅为普通商业、工业、个人和/或家庭应用而设计、开发和制造。禁止用于军事、国防、核能、医疗以及可能导致人身伤害、死亡，或是环境破坏等领域。用户应采取任何和所有行动，确保按照适用的法律法规使用和销售产品。

半导体产品自身存在一定的失效概率。为防止因故障或误工作而产生的人身损害、火灾事故或其他社会性损害，请注意冗余设计、消防设计以及其他安全防护设计。特别说明：参考应用电路不保证能够适用于特定应用的量产。

若用户违反上述声明，本公司不承担全部或部分责任，用户应在此免除本公司及其供应商和/或分销商因产品的所有非预期用途有关的任何索赔、损害或其他责任；与此同时，用户应赔偿并确保本公司及其供应商和/或分销商免受因产品的任何非预期用途的使用而产生的与之相关的所有索赔、成本、损害赔偿和其他责任，包括人身伤害或死亡的索赔。

本手册中的信息仅与产品有关。本公司保留随时对本手册及所述的产品和服务进行更改、修改或改进的权利，恕不另行通知。订购前建议用户咨询销售代表。

本公司对本手册拥有最终解释权。

目录

1. 特性 1

2. 概述 2

3. 方框图 3

4. 引脚定义 4

5. 引脚描述 6

5.1 复用功能映射表 6

5.2 引脚功能描述 7

6. 相关文档 9

7. 内核 10

7.1 ARM® "星辰"处理器 10

7.2 寄存器组 10

7.2.1 ARM "星辰"处理器拥有R0-R15的寄存器组。其中R13作为堆栈指针SP。通用寄存器 10

7.2.2 堆栈寄存器 10

7.2.3 连接寄存器 10

7.2.4 程序计数寄存器 10

7.3 特殊功能寄存器 11

7.3.1 程序状态字寄存器组 11

7.3.2 中断屏蔽寄存器组 14

7.3.3 控制寄存器 14

7.4 模式、状态和访问方式 15

7.4.1 工作模式 15

7.4.2 工作状态 15

7.4.3 特权访问和用户访问 15

7.5 指令集 16

8. 存储器和总线架构 22

8.1 系统架构 22

8.2 存储器组织 23

8.2.1 寄存器映像 24

8.2.2 数据存储器 24

8.2.3 程序存储器 24

8.3 Flash程序存储器 25

8.3.1 特性 25

8.3.2 简介 25

8.3.3 Flash存储器功能描述 26

8.4 Flash 访问加速控制 34

8.5 Flash切分区块功能.....35

8.6 寄存器.....36
8.6.1 Flash访问控制寄存器 (FLASH_ACR).....37
8.6.2 Flash主程序存储区解锁控制寄存器 (FLASH_MKYR).....38
8.6.3 类EEPROM存储区解锁控制寄存器 (FLASH_E2KYR).....38
8.6.4 Flash状态和清除寄存器 (FLASH_SR).....39
8.6.5 Flash控制寄存器 (FLASH_CR).....40
8.6.6 Flash代码选项寄存器 (FLASH_OPR).....41
8.6.7 Flash读保护寄存器 (FLASH_RPR).....41
8.6.8 Flash写保护寄存器 (FLASH_WRPR).....42
8.6.9 Flash操作定时器计数寄存器 (FLASH_CNTR).....42
8.6.10 Flash操作定时器上限值 (FLASH_UPCNTR).....43
8.6.11 Flash操作定时器启动寄存器 (FLASH_CNTCR).....43
8.6.12 Flash特殊信息存储区解锁控制寄存器 (FLASH_IKYR).....44
8.6.13 内存映射选择寄存器 (FLASH_MEMRMP).....44

9. 电源控制（PWR）.....45

9.1 电源与时钟预热.....45

9.2 掉电检测（BOD）.....46

9.3 低电压复位（LVR）.....47

9.4 低功耗管理.....47
9.4.1 睡眠模式.....48
9.4.2 停机模式.....49
9.4.3 低功耗模式下的调试.....50

9.5 寄存器.....50
9.5.1 电源控制寄存器 (SYSCFG_PWRCR).....51
9.5.2 电源状态寄存器 (SYSCFG_PWRSR).....52

10. 复位和时钟控制（RCC）.....53

10.1 复位.....53
10.1.1 系统复位.....53
10.1.2 电源复位.....53

10.2 时钟.....54
10.2.1 HSE时钟.....55
10.2.2 HSI时钟.....55
10.2.3 PLL.....56
10.2.4 LSI时钟.....56
10.2.5 系统时钟(SYSCLK)选择.....56
10.2.6 时钟安全监控（CSM）.....57
10.2.7 系统节拍定时器（SysTick）.....59
10.2.8 HSI修整功能（HSITRIM）.....59

10.3 寄存器.....60
10.3.1 时钟控制寄存器 (RCC_CR).....60
10.3.2 时钟配置寄存器 (RCC_CFGR).....62
10.3.3 时钟中断使能寄存器 (RCC_CIENR).....64
10.3.4 时钟中断状态寄存器 (RCC_CISTR).....65

10.3.5 时钟中断标志清除寄存器 (RCC_CICLR) 66
10.3.6 AHB外设复位寄存器 (RCC_AHBRSTR) 67
10.3.7 APB外设复位寄存器 (RCC_APB0RSTR) 68
10.3.8 AHB外设时钟使能寄存器 (RCC_AHBENR) 70
10.3.9 APB外设时钟使能寄存器 (RCC_APB0ENR) 71
10.3.10 RESET状态寄存器 (RCC_RSTSTR) 73
10.3.11 清除RESET状态寄存器 (RCC_RSTCLR) 74
10.3.12 HSI振荡器校准寄存器 (RCC_HSICAL) 75
10.3.13 RCC配置锁定寄存器 (RCC_RCCLOCK) 76
10.3.14 SysTick控制及状态寄存器（SysTick_CTRL） 77
10.3.15 SysTick重装载数值寄存器（SysTick_LOAD） 78
10.3.16 SysTick当前数值寄存器（SysTick_VAL） 78
10.3.17 SysTick校准数值寄存器（SysTick_CALIB） 79

11. 看门狗定时器（WDT） 80

11.1 独立看门狗定时器（IWDT） 80

11.2 窗口看门狗定时器（WWDT） 80

11.3 IWDT寄存器 81
11.3.1 独立看门狗控制寄存器 (IWDT_CR) 81
11.3.2 独立看门狗喂狗寄存器 (IWDT_CLR) 82

11.4 WWDT寄存器 82
11.4.1 窗口看门狗控制寄存器 (WWDT_CR) 82
11.4.2 窗口看门狗状态寄存器 (WWDT_SR) 83
11.4.3 窗口看门狗喂狗寄存器 (WWDT_CLR) 84
11.4.4 窗口看门狗窗口寄存器 (WWDT_WTR) 84

12. 代码和数据检测（Code and Data Checking） 85

12.1 CRC模块（CRC） 85
12.1.1 CRC简介 85
12.1.2 CRC 主要特性 85
12.1.3 CRC 功能描述 86
12.1.4 CRC寄存器 89

13. 通用功能I/O (GPIO) 91

13.1 简介 91

13.2 主要特性 91

13.3 功能描述 91
13.3.1 I/O引脚的默认配置 92
13.3.2 I/O控制寄存器 93
13.3.3 I/O数据寄存器 93
13.3.4 I/O位置位和位清除 93
13.3.5 I/O锁定机制 93
13.3.6 I/O配置为外部中断/唤醒线 93
13.3.7 TTL输入选择 94
13.3.8 I/O复用功能 94
13.3.9 GPIO输入功能 （AF1） 95

13.3.10 GPIO输出功能 （AF1）.....95
13.3.11 数字复用功能 （AF2-AF6） 96
13.3.12 模拟复用功能 （AF6/7）.....97
13.3.13 全关功能 （AF0）.....97
13.3.14 I/O引脚的基本应用.....98

13.4 寄存器.....99
13.4.1 端口x模式寄存器 (GPIOx_MODER) 99
13.4.2 端口x输入数据寄存器 (GPIOx_IDR).....101
13.4.3 端口x输出数据寄存器 (GPIOx_ODR) 101
13.4.4 端口位清除/置位寄存器 (GPIOx_BSRR) 101
13.4.5 端口x配置锁定寄存器 (GPIOx_LCKR).....102
13.4.6 端口x输入使能配置寄存器 (GPIOx_IE).....104
13.4.7 端口x输出方式配置寄存器 (GPIOx_OTYPER) 105
13.4.8 端口x输出驱动能力配置寄存器 (GPIOx_ODRVR).....106
13.4.9 端口x上下拉配置寄存器 (GPIOx_PUPDR).....106
13.4.10 端口x部分引脚TTL电平选择寄存器 (GPIOBx_TTLEN).....108
13.4.11 端口x复用功能寄存器低位 (GPIOx_AFRL).....109
13.4.12 端口x复用功能寄存器高位 (GPIOx_AFRH) 111

14. 系统配置模块(SYSCFG).....113

14.1 系统配置相关.....113
14.1.1 SWD和XTAL引脚复用功能定义.....113
14.1.2 NMI中断信号源配置.....113
14.1.3 调试接口寄存器.....113

14.2 寄存器.....113
14.2.1 系统配置寄存器 (SYSCFG_SAFR).....114
14.2.2 调试接口控制寄存器 (SYSCFG_DBGCR) 115

15. 中断和事件 117

15.1 可嵌套中断向量控制器（NVIC） 117
15.1.1 简介.....117
15.1.2 主要特性.....117
15.1.3 中断汇总.....117
15.1.4 NVIC寄存器列表.....119

15.2 外部中断/事件控制器（EXTI） 120
15.2.1 主要特性.....120
15.2.2 唤醒事件管理.....121
15.2.3 功能说明.....121
15.2.4 外部中断/事件线路映像 122

15.3 寄存器.....122
15.3.1 中断屏蔽寄存器 (EXTI_IMR) 123
15.3.2 事件屏蔽寄存器 (EXTI_EMR) 123
15.3.3 触发模式选择寄存器 (EXTI_TMSR).....124
15.3.4 上升沿/高电平触发模式选择寄存器 (EXTI_RTSR).....124
15.3.5 下降沿/低电平触发模式选择寄存器 (EXTI_FTSR) 125
15.3.6 软件中断事件寄存器 (EXTI_SWIER) 125
15.3.7 挂起寄存器 (EXTI_PR).....126

15.3.8 外部中断配置低寄存器 (EXTI_CFG) 126
15.3.9 外部中断配置高寄存器 (EXTI_CFGH) 127
15.3.10 EXTI采样控制寄存器 (EXTI_SAMP) 127

16. 电机控制模块（Motor Control Module，MCM） 130

16.1 简介 130

16.2 主要特性 131

16.3 功能 131
16.3.1 PWM0/1/2时基模块（PWM Time Base Block） 131
16.3.2 PWM0/1/2波形发生模块 135
16.3.3 PWM死区与极性控制模块 144
16.3.4 PWM故障检测模式 146
16.3.5 PWM异常保护 148
16.3.6 PWM输出控制模块 149
16.3.7 事件触发功能 149
16.3.8 PWM占空比饱和功能 150
16.3.9 PWM相移功能 151
16.3.10 PWM逐波限流功能 157
16.3.11 PWM寄存器的写保护 158
16.3.12 注意事项 159

16.4 寄存器 164
16.4.1 PWM输出使能寄存器 (MCM_PWMOE) 165
16.4.2 PWM模块控制寄存器1 (MCM_PWMCON1) 165
16.4.3 PWM模块控制寄存器2 (MCM_PWMCON2) 166
16.4.4 PWM周期寄存器 (MCM_PWMP) 168
16.4.5 PWM计数寄存器 (MCM_PWMC) 169
16.4.6 PWM时钟预分频寄存器 (MCM_PWMPSQ) 169
16.4.7 PWM0/1/2占空比寄存器 (MCM_PWMnD)(n=0..2) 170
16.4.8 PWM01/11/21占空比寄存器 (MCM_PWMn1D)(n=0..2) 170
16.4.9 事件触发比较寄存器1 (MCM_PWMCMPn)(n=1..4) 171
16.4.10 PWM通道0上升沿死区控制寄存器 (MCM_PWMDT00) 171
16.4.11 PWM通道0下降沿死区控制寄存器 (MCM_PWMDT01) 172
16.4.12 PWM通道1上升沿死区控制寄存器 (MCM_PWMDT10) 172
16.4.13 PWM通道1下降沿死区控制寄存器 (MCM_PWMDT11) 173
16.4.14 PWM通道2上升沿死区控制寄存器 (MCM_PWMDT20) 173
16.4.15 PWM通道2下降沿死区控制寄存器 (MCM_PWMDT21) 173
16.4.16 PWM0/1/2手动输出设置寄存器1 (MCM_PMANUALCON1) 174
16.4.17 PWM0/1/2手动输出设置寄存器2 (MCM_PMANUALCON2) 174
16.4.18 FLT引脚故障检测保护PWM0/1/2寄存器 (MCM_FLTCON) 175
16.4.19 PWM0/1/2输出电平保护控制寄存器 (MCM_POSCR) 178
16.4.20 PWM0/1/2振荡器停止检测保护控制位 (MCM_POSTDCR) 179
16.4.21 PWM中断使能控制寄存器 (MCM_PWMINTEN) 179
16.4.22 PWM中断标志和清除寄存器 (MCM_PWMINTF) 181
16.4.23 寄存器修改和重载控制寄存器 (MCM_PWMRLDEN) 183
16.4.24 PWM饱和/相移控制寄存器 (MCM_PSCON) 184
16.4.25 PWM占空比上限比较寄存器 (MCM_PWMDMAX) 185
16.4.26 PWM占空比下限比较寄存器 (MCM_PWMDMIN) 185
16.4.27 PWM占空比移相最小采样间隔寄存器1 (MCM_PWMDCMPn)(n=1..2) 185

16.4.28 逐波限流引脚0功能选择寄存器 (MCM_SCnCON)(n=0..1) 186
16.4.29 逐波限流引脚2功能选择寄存器 (MCM_SC2CON) 188
16.4.30 寄存器修改和重载保护控制寄存器 (MCM_FLTWEN) 190

17. 可编程计数器阵列（PCA，PCA0～PCA1） 191

17.1 简介 191

17.2 主要特性 191

17.3 功能描述 191

17.4 工作模式 192
17.4.1 输入捕捉模式 192
17.4.2 比较匹配输出 193
17.4.3 频率输出方式 194
17.4.4 PWM输出 195

17.5 中断 196

17.6 寄存器立即更新 197

17.7 寄存器锁定 197

17.8 输出故障检测 197
17.8.1 故障信号输入源的选择 197
17.8.2 FLT 引脚上的信号与比较器输出信号的滤波 197
17.8.3 故障检测同步控制 197

17.9 比较触发ADC事件 198

17.10 同步启动 198
17.10.1 PCA同步使能 198
17.10.2 MCM同步使能 198

17.11 注意事项 198
17.11.1 仿真状态下的输出状态 198
17.11.2 STOP模式下的输出状态 198

17.12 寄存器 198
17.12.1 PCAx 使能寄存器 (PCAx_CR) 199
17.12.2 PCAx 方式寄存器 (PCAx_CFGR) 199
17.12.3 PCAx 标志寄存器 (PCAx_SR) 201
17.12.4 PCAx 强制输出控制寄存器 (PCAx_FORCE) 202
17.12.5 PCAx计数器寄存器 (PCAx_CNT) 203
17.12.6 预分频器 (PCAx_PSC) 203
17.12.7 PCAx 计数周期值 (PCAx_PR) 203
17.12.8 PCAx 捕捉/比较模块0控制寄存器 (PCAx_CCMR0) 204
17.12.9 PCAx 捕捉/比较模块1控制寄存器 (PCAx_CCMR1) 205
17.12.10 PCAx 捕捉/比较模块2控制寄存器 (PCAx_CCMR2) 206
17.12.11 PCAx 比较/捕捉模块0捕捉值 (PCAx_CCR0) 208
17.12.12 PCAx 比较/捕捉模块1捕捉值 (PCAx_CCRn)(n=1..2) 208
17.12.13 寄存器保护控制寄存器 (PCAx_LCKR) 208
17.12.14 比较触发寄存器 (PCAx_ADTR) 209
17.12.15 PCA写保护寄存器 (PCAx_FLTWPR) 209
17.12.16 PCA故障检测保护寄存器 (PCAx_FLTCR) 210

18. 基本定时器（Basic Timer，TIM7～TIM8）.....212

18.1 简介.....212

18.2 主要特性.....212

18.3 功能.....212
18.3.1 计数器/定时器运行.....212
18.3.2 级联运行.....213
18.3.3 外部输入时钟源滤波.....213

18.4 寄存器.....214
18.4.1 TIM控制寄存器 (TIMx_CR).....214
18.4.2 TIM计数寄存器 (TIMx_TCNT).....215
18.4.3 TIM周期寄存器 (TIMx_TPR).....216
18.4.4 TIM预分频寄存器 (TIMx_PSQ).....216
18.4.5 TIMx中断标志和清除寄存器 (TIMx_TIMINTF).....217

19. 数学协处理单元（Math Co-Processor，MACP）.....218

19.1 简介.....218

19.2 寄存器列表.....218

19.3 CORDIC运算单元.....218
19.3.1 主要特性.....218
19.3.2 功能描述.....219
19.3.3 CORDIC寄存器.....224

19.4 电机专用SVPWM增强引擎.....227
19.4.1 SVPWM功能.....227
19.4.2 SVPWM寄存器.....234

20. 模数转换器（Analog to Digital Converter，ADC）.....238

20.1 简介.....238

20.2 主要特性.....240

20.3 功能.....240
20.3.1 单次转换方式.....240
20.3.2 间断转换方式.....241
20.3.3 连续转换方式.....242
20.3.4 单电阻转换方式.....243
20.3.5 其它功能描述.....243
20.3.6 比较功能.....244
20.3.7 ADC转换时间设置.....245
20.3.8 ADC模块参考电压的设置.....245
20.3.9 ADC通道与IO口功能设置.....245
20.3.10 AD转换时对传感器输出阻抗的要求.....246
20.3.11 注意事项.....246

20.4 寄存器.....247
20.4.1 ADC控制寄存器1 (ADC_ADCON1).....248
20.4.2 ADC控制寄存器2 (ADC_ADCON2).....249
20.4.3 ADC采样转换通道指针寄存器 (ADC_ADPCH).....250

20.4.4 ADC结果寄存器0 (ADC_ADDRn)(n=0..15) 250
20.4.5 ADC比较控制寄存器 (ADC_ADCMPCON) 251
20.4.6 ADC上限比较寄存器 (ADC_ADDGT) 251
20.4.7 ADC下限比较寄存器 (ADC_ADDLT) 252
20.4.8 ADC通道选择寄存器1 (ADC_SEQCHSEL0) 252
20.4.9 ADC通道选择寄存器2 (ADC_SEQCHSEL1) 255
20.4.10 单电阻模式下，硬件触发ADC的起始通道选择位 (ADC_HARDCHSEL) 258
20.4.11 单电阻模式下，硬件触发ADC的通道长度设置位 (ADC_HARDCHLH) 259
20.4.12 ADC通道GAP控制位 (ADC_ADGAPON) 259
20.4.13 ADC中断标志和清除寄存器 (ADC_ADINTF) 260

21. 正交编码器接口模块（QEI） 261

21.1 简介 261

21.2 主要特性 262

21.3 功能 262
21.3.1 输入滤波 262
21.3.2 正交解码器 262
21.3.3 位置计数器 264
21.3.4 测速方案 267
21.3.5 QEI模块的中断 268

21.4 寄存器 269
21.4.1 QEI控制寄存器 (QEI_QEICON) 269
21.4.2 QEI输入引脚滤波控制寄存器 (QEI_QFLTCON) 271
21.4.3 QEI位置计数寄存器 (QEI_QPOSCNT) 272
21.4.4 QEI最大计数值寄存器 (QEI_QPOSMAX) 272
21.4.5 QEI计数值锁存寄存器 (QEI_QTMLAT) 272
21.4.6 QEI计数寄存器 (QEI_QCNT) 273
21.4.7 QEI计数器在捕捉时最小计数值 (QEI_QCNTMIN) 273
21.4.8 QEI计数器周期寄存器 (QEI_QTPR) 274
21.4.9 QEI计数器时钟预分频值 (QEI_QTPSQ) 274
21.4.10 QEI中断和状态寄存器 (QEI_QEIINT) 275
21.4.11 QEI中断标志和清除寄存器 (QEI_QTINTF) 275

22. 片上模拟功能模块（Analog Module On Chip，AMOC） 277

22.1 简介 277

22.2 主要特性 277

22.3 运算放大器 277
22.3.1 运放典型应用图 277

22.4 比较器 278
22.4.1 比较器用法 278
22.4.2 比较器框图 278

22.5 寄存器 279
22.5.1 比较器0控制寄存器 (AMOC_CMP0CON) 280
22.5.2 比较器1控制寄存器 (AMOC_CMP1CON) 282
22.5.3 比较器中断标志和清除寄存器 (AMOC_CMPINTF) 283

22.5.4 放大器控制寄存器1 (AMOC_OPCON) 284

23. 串行通信接口 285

23.1 通用异步收发器（UART） 285
23.1.1 简介....285
23.1.2 主要特性....285
23.1.3 功能描述....285
23.1.4 UART寄存器....298

23.2 串行外设接口（SPI） 304
23.2.1 简介....304
23.2.2 主要特性....304
23.2.3 功能描述....304
23.2.4 SPI寄存器....313

23.3 两线串行接口（TWI） 317
23.3.1 简介....317
23.3.2 主要特性....317
23.3.3 功能描述....318
23.3.4 TWI寄存器....333

23.4 FIFO控制器....338
23.4.1 简介....338
23.4.2 主要特性....338
23.4.3 功能描述....338
23.4.4 FIFO4L寄存器....342
23.4.5 FIFO8L寄存器....345

24. CAN控制器....348

24.1 简介....348

24.2 主要特性....348

24.3 功能描述....348
24.3.2 CAN信号....350
24.3.3 CAN内核....350
24.3.4 接口管理....350
24.3.5 信息缓冲区....350
24.3.6 时钟发生器....350

24.4 信息传输....351

24.5 验收过滤....351
24.5.1 过滤缓冲区0到13....352
24.5.2 过滤缓冲区14....353
24.5.3 BUFFLOCK功能....353
24.5.4 接收结构....354
24.5.5 数据帧....354
24.5.6 远程帧....354
24.5.7 接收时序....355
24.5.8 接收程序....356
24.5.9 RX缓冲区状态....358

24.6 传输结构 358
24.6.1 发送远程帧 358
24.6.2 应答远程帧 359
24.6.3 传输调度 359
24.6.4 传输优先级 360
24.6.5 传输过程 361
24.6.6 TX缓冲区状态 362

24.7 中断 363
24.7.1 中断条件 363
24.7.2 最高优先级中断代码 363

24.8 时间戳计数器 365

24.9 存储器组织 365
24.9.1 CPU访问CAN寄存器/存储器 365
24.9.2 寄存器分布 365
24.9.3 消息缓冲区组织 366
24.9.4 缓冲状态/控制寄存器（CNSTAT） 366
24.9.5 存储标准消息 369
24.9.6 存储扩展消息 370
24.9.7 存储远程消息 371

24.10 寄存器 371
24.10.1 缓冲区n状态/控制寄存器 (CAN_BUFn_CNTSTAT)(n=0..14) 374
24.10.2 缓冲区n时间戳寄存器 (CAN_BUFn_TSTP)(n=0..14) 375
24.10.3 缓冲区n数据3寄存器 (CAN_BUFn_DATA3)(n=0..14) 375
24.10.4 缓冲区n数据2寄存器 (CAN_BUFn_DATA2)(n=0..14) 376
24.10.5 缓冲区n数据1寄存器 (CAN_BUFn_DATA1)(n=0..14) 377
24.10.6 缓冲区n数据0寄存器 (CAN_BUFn_DATA0)(n=0..14) 377
24.10.7 缓冲区n标识符0寄存器 (CAN_BUFn_ID0)(n=0..14) 378
24.10.8 缓冲区n标识符1寄存器 (CAN_BUFn_ID1)(n=0..14) 379
24.10.9 CAN全局配置寄存器 (CAN_CGCR) 379
24.10.10 CAN时序寄存器 (CAN_CTIM) 381
24.10.11 CAN全局屏蔽寄存器 (CAN_GMSKX) 382
24.10.12 CAN全局屏蔽寄存器 (CAN_GMSKB) 382
24.10.13 CAN基本屏蔽寄存器 (CAN_BMSKX) 383
24.10.14 CAN基本屏蔽寄存器 (CAN_BMSKB) 383
24.10.15 CAN中断使能寄存器 (CAN_CIEN) 384
24.10.16 CAN中断标志寄存器 (CAN_CIPND) 385
24.10.17 CAN中断标志位清除寄存器 (CAN_CICLR) 387
24.10.18 CAN中断码使能寄存器 (CAN_CICEN) 388
24.10.19 CAN状态挂起寄存器 (CAN_CSTPND) 389
24.10.20 CAN错误计数器寄存器 (CAN_CANEC) 390
24.10.21 CAN错误诊断寄存器 (CAN_CEDIAG) 391
24.10.22 CAN定时器寄存器 (CAN_CTMR) 392

25. 器件电子签名 393

26. 调试接口 394

26.1 主要特性 394

26.2 SW调试端口(Serial Wire) *394*

26.3 SW调试端口引脚 *394*

26.4 利用串行接口并释放不用的调试脚作为普通I/O口 *394*

26.5 MCU调试模块 *395*
26.5.1 低功耗模式的调试支持 395
26.5.2 带定时功能的模块调试支持 395

26.6 寄存器 *395*
26.6.1 系统配置寄存器 (SYSCFG_SAFR) 395
26.6.2 调试接口控制寄存器 (SYSCFG_DBGCR) 396

27. 用户代码选项 398

28. 电气特性 399

直流电气特性 (VDD = 2.4 – 5.5V, TA = 25°C, 除非另有说明) 399
高速12BIT模/数转换器电气特性（1LSB = V_{DD}/4096） 401
比较器0/1电气特性（V_{DD} = 2.7 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明） 401
放大器0/1/2/3电气特性（V_{DD} = 2.7 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明） 402
交流电气特性（V_{DD} = 2.4 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明） 402
PLL特性（V_{DD} = 2.4 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明） 402
FLASH特性（VDD = 2.4 - 5.5V，GND = 0V，TA = +25°C，除非另有说明） 403
掉电检测（BOD）电气特性（V_{DD} = 2.4 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明） 403
低电压复位（LVR）电气特性（V_{DD} = 2.4 - 5.5V，GND = 0V，T_A = +25°C，除非另有说明） 403

29. 订购信息 404

30. 封装信息 405

31. 规格更改记录 407