SINO WEALTH

SH32F284

基于ARM®Cortex™-M3内核的32位通用电机主控芯片

1. 特性

- 采用ARM Cortex-M3内核的32位MCU
  - 系统最高84MHz主频
  - M3内核内置硬件32位乘/除法
  - 高速存储器和指令加速，提高执行效率
  - 内置存储器保护单元（MPU）
- 集成数学运算协处理器（MACP）
  - 内置硬件CORDIC运算器，支持16位、24位正余弦、反正切、求模运算、Park变换与反变换
  - 内置硬件IQ除法器，支持硬件32位IQ格式除法
  - 内置FOC专用硬件SVPWM引擎，支持五段式和七段式波形生成
- 存储器
  - 最大128K字节FLASH程序存储器
  - 最大8K字节SRAM+16K字节CRAM（也用作DRAM）
  - 最大6K字节内置类EEPROM区
  - 包含2K字节OTP（一次编程）区
  - 支持对代码全方位保护：包括两级读保护、写保护、擦除保护、客户安全码保护等
- 时钟、复位和电源管理
  - 2.4V - 5.5V供电范围，覆盖3V/5V广泛系统
  - 上电复位、掉电检测、低电压复位
  - 硬件独立看门狗（IWDT）、窗口看门狗（WWDT）
  - 4M - 16MHz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）
  - 集成PLL，无需外接电容，最高输出84MHz时钟
  - 时钟停振监控
- 2个完全独立12位高速模数转换器
  - 最高2M SPS采样速率
  - 支持16个模拟采样通道（其中11个采样引脚ANx）
  - 可配置序列转换，每个ADC配有8个AD转换结果寄存器
  - 支持PWM同步触发AD采样
  - 支持单次/间断/连续采样模式，支持事件触发AD采样
  - ADC基准源可选择内部基准电压或外部输入基准电压
- DMA控制器
  - 8个独立的DMA通道
  - 支持内存与外设之间任意组合的传输
  - ADC/MCM/GPT/UART/SPI等模块都支持DMA操作
- 最多58个快速GPIO端口（GPIOx）
  - 64PIN有58个、48PIN有42个I/O端口
  - 支持I/O功能重映射
  - 所有I/O内建上拉/下拉电阻
  - 4档驱动能力，部分引脚提供强灌电流能力
  - 部分引脚兼容TTL电平输入
  - 支持I/O位操作
- 16个外部中断输入通道（EXTIx）
  - 任何一个GPIO都可作为外部中断输入通道
  - 支持边沿触发、电平触发、软件触发
- 1个电机专用控制模块（MCM）
  - 提供6路（或3对）PWM输出，16位分辨率
  - 支持互补/独立输出模式，支持单次/边沿对齐/中心对齐计数模式，支持对称与非对称波形生成
  - 集成死区逻辑、故障保护、逻辑异常保护、逐波限流等
  - 每个MCM集成4个事件比较器，可由事件触发ADC采样，提供4个灵活的采样时刻点
  - 支持相移功能，适合单电阻采样
- 3个通用PWM定时器（GPTx）
  - 每个GPT提供2路PWM，16位分辨率
  - 支持单脉冲、锯齿波、三角波输出
  - 双缓冲寄存器，支持对称与非对称波形生成
  - 支持死区产生逻辑和PWM故障保护功能
  - 支持GPT的同步运行，可组合为32bit定时器
  - 3个GPT支持最多6路PWM输出、输入捕捉或比较输出
- 4个16位基本定时器/计数器（TIMx）
  - 可级联组合成2个32位定时器/计数器
  - 自动重载累加计数器
- 1个正交编码器接口模块（QEI），包含独立32位定时器
- 多个串行通信接口
  - 3个通用异步串行口（UARTx）
  - 1个串行外设接口（SPI）
  - 1个两线串行接口（TWI），主从模式，兼容I^2C
- 其他片上模拟功能模块
  - 3个独立高速模拟放大器（OPx）
  - 3个高速模拟比较器（CMPx）
  - 内置高精度基准电压源，可作为ADC和比较器基准
  - 内置温度传感器
- 硬件CRC模块，8/16/32位，可用于代码校验、数据校验
- 硬件支持的RAMBIST功能
  - 支持March-C/March-X检测算法
  - 可在不中断用户程序情况下对SRAM进行检测
- LVR功能（3档）、BOD功能（16档）
- 96位芯片唯一识别码
- 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）
- 48PIN支持2线SWD接口，64PIN支持2线/4线接口
- 工作环境温度：-40℃～+105℃
- 封装：LQFP64，TQFP48

2. 概述

SH32F284是基于ARM Cortex-M3内核的通用32位微控制器，最高支持84MHz主频，由于采用存储器指令加速结构，可使系统获得近乎零等待的运行效率。

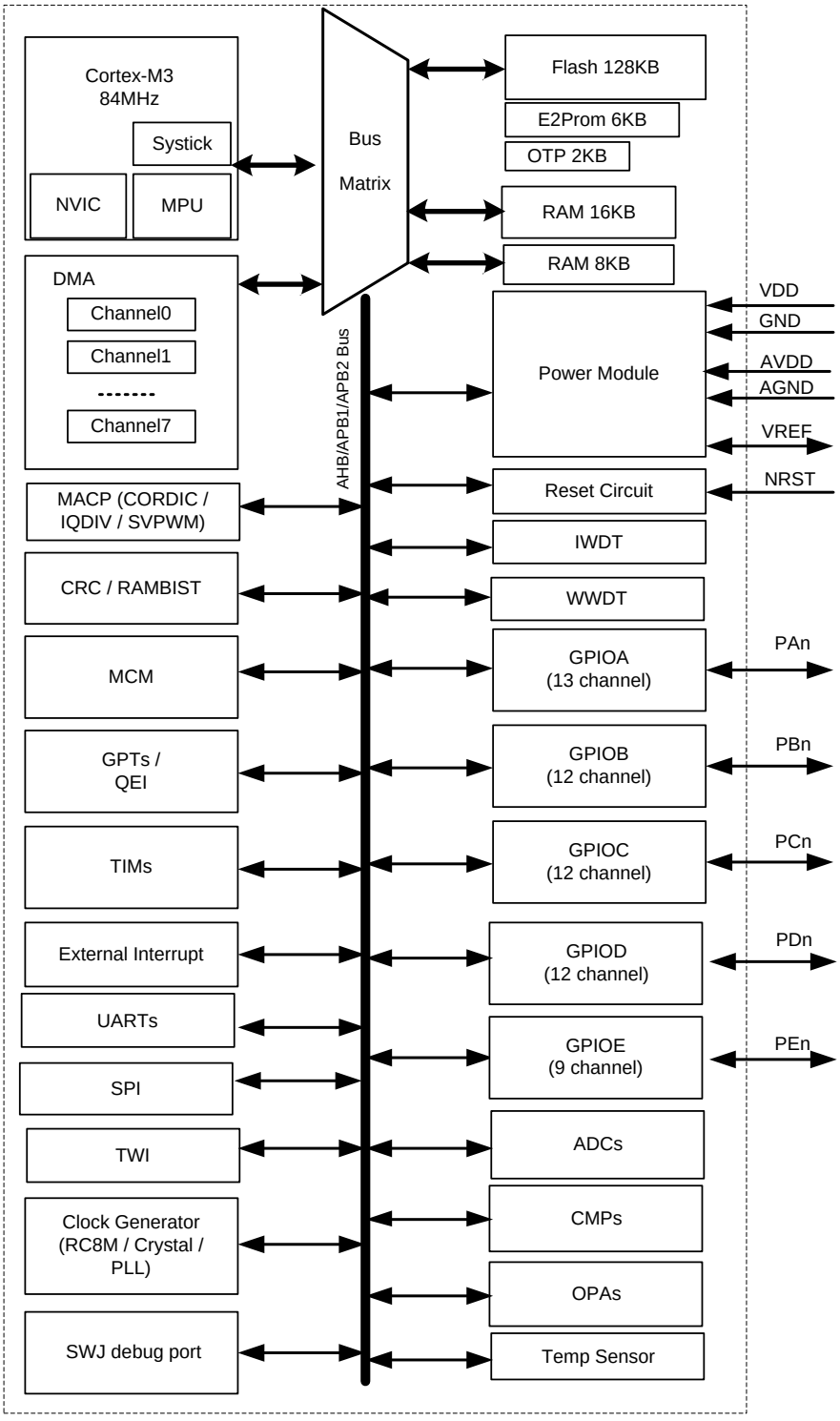
SH32F284是32位工控型微控制器，主要应用在电机控制应用和其他工控领域，包括各种直流无刷电机、永磁同步电机、高性能变频器、高效逆变器等。

SH32F284内置了硬件CORDIC运算器、硬件32位定点IQDIV除法器（Cortex-M3内核已内置32位乘除法计算单元，而IQDIV可进一步加速除法运算）、硬件SVPWM生成器，用以实现电机控制常用的坐标旋转、正余弦计算、反正切计算、求模计算、Park变换和逆变换等，硬件SVPWM生成器可简化FOC方案中五段式和七段式SVPWM波形的生成。

SH32F284内置了一个电机专用PWM模块（MCM），三个通用PWM发生器（GPT0/1/2，带比较捕捉功能），集成了三个高速运算放大器和三个模拟比较器，两个12位2M SPS多通道ADC，内置温度传感电路和参考电压发生电路等。

SH32F284提供8/16/32位CRC代码校验和数据校验，提供内存自检模块（RAMBIST），独立看门狗，窗口看门狗，低电压复位电路，掉电检测电路，时钟停振检测等模块，提供基于客户安全码的代码保护，保证系统的可靠性和代码的安全性。

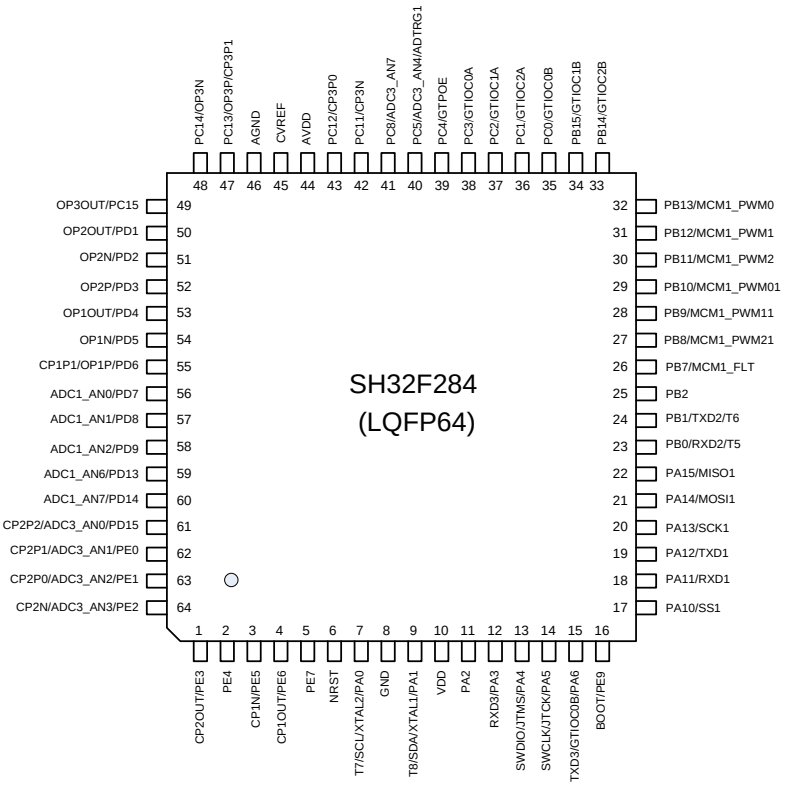
3. 方框图


Cortex-M3
84MHz
Systick
NVIC
MPU
DMA
Channel0
Channel1

Channel7
Bus
Matrix
AHB/APB1/APB2 Bus
Flash 128KB
E2Prom 6KB
OTP 2KB
RAM 16KB
RAM 8KB
Power Module
VDD
GND
AVDD
AGND
VREF
Reset Circuit
NRST
IWDT
WWDT
GPIOA
(13 channel)
PAn
GPIOB
(12 channel)
PBn
GPIOC
(12 channel)
PCn
GPIOD
(12 channel)
PDn
GPIOE
(9 channel)
PEn
ADCs
CMPs
OPAs
Temp Sensor
MACP (CORDIC /
IQDIV / SVPWM)
CRC / RAMBIST
MCM
GPTs /
QEI
TIMs
External Interrupt
UARTs
SPI
TWI
Clock Generator
(RC8M / Crystal /
PLL)
SWJ debug port

4. 引脚定义

64脚LQFP封装



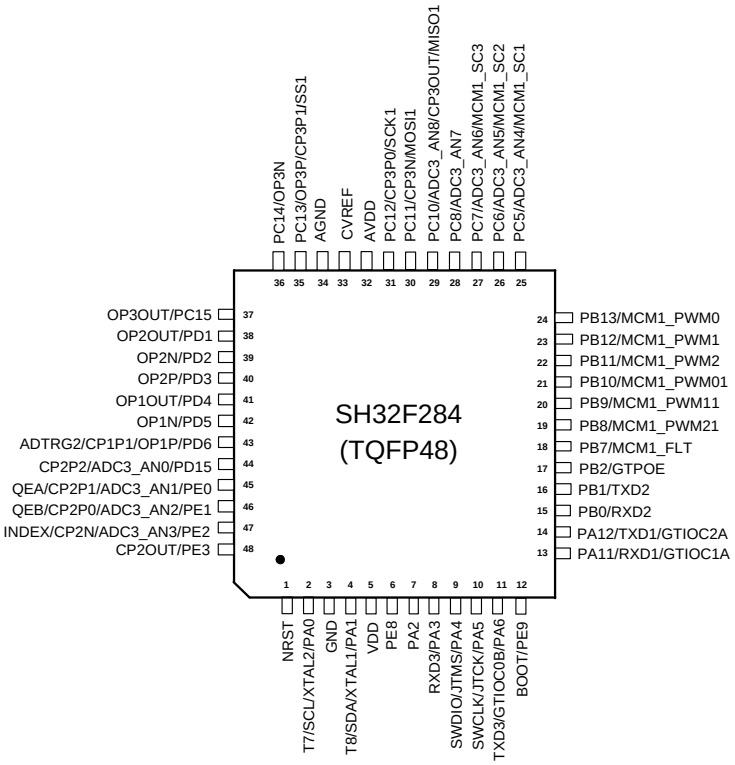
注意：

*(1)上图中每个引脚只列出了一种复用选项，芯片支持灵活的引脚重映射，比如TXD1功能可映射到PA12、PD14、PE7三个引脚之一，但在图中只标注到了PA12引脚上。请参考随后的"**复用功能映射表**"获得全部的重映射信息。*

(2)除仿真端口（SWJ）外，所有引脚上电默认是GPIO口模式。

(3)有关复用功能与引脚重映射详细内容请参考"通用功能I/O（GPIO）"章节。

48脚TQFP封装



注意：*个别TQFP48的引脚在LQFP64是不存在的（请见"**复用功能映射表**"高亮标注的4个引脚），应用时注意。*

复用功能映射表（Alternate Function Mapping）

LQFP64	TQFP48	AF0	AF1	-	AF3	-	AF5	AF6	AF7	AF8	AF9	AF10	AF14	AF15
		GPIO	MCM		GPT		TIM	QEI	UART	SPI	TWI			
●	●	PA0					T7		RXD3		SCL1			OFF
●	●	PA1					T8		TXD3		SDA1			OFF
●	●	PA2			GTIOC2B				RXD1					OFF
●	●	PA3			GTIOC1B				RXD3					OFF
●	●	PA4												OFF
●	●	PA5												OFF
●	●	PA6			GTIOC0B				TXD3					OFF
●	○	PA10			GTIOC0A					SS1				OFF
●	●	PA11			GTIOC1A				RXD1					OFF
●	●	PA12			GTIOC2A				TXD1					OFF
●	○	PA13			GTIOC0B					SCK1				OFF
●	○	PA14			GTIOC1B					MOSI1				OFF
●	○	PA15			GTIOC2B					MISO1				OFF
●	●	PB0					T5		RXD2		SDA1			OFF
●	●	PB1					T6		TXD2		SCL1			OFF
●	●	PB2			GTPOE									OFF
●	●	PB7	MCM1_FLT											OFF
●	●	PB8	MCM1_PWM21											OFF
●	●	PB9	MCM1_PWM11											OFF
●	●	PB10	MCM1_PWM01											OFF
●	●	PB11	MCM1_PWM2											OFF
●	●	PB12	MCM1_PWM1						RXD2					OFF
●	●	PB13	MCM1_PWM0						TXD2					OFF
●	○	PB14			GTIOC2B									OFF
●	○	PB15			GTIOC1B									OFF
●	○	PC0			GTIOC0B									OFF
●	○	PC1			GTIOC2A									OFF
●	○	PC2			GTIOC1A									OFF
●	○	PC3			GTIOC0A									OFF
●	○	PC4			GTPOE									OFF
●	●	PC5	MCM1_SC1					QEA				ADTRG1	ADC3_AN4	OFF
○	●	PC6	MCM1_SC2					QEB					ADC3_AN5	OFF
○	●	PC7	MCM1_SC3					INDEX				ADTRG3	ADC3_AN6	OFF
●	●	PC8											ADC3_AN7	OFF
○	●	PC10			GTPOE					MISO1		CP3OUT	ADC3_AN8	OFF
●	●	PC11			GTIOC2B					MOSI1			CP3N	OFF
●	●	PC12			GTIOC1B					SCK1			CP3P0	OFF
●	●	PC13			GTIOC0B					SS1			OP3P/CP3P1	OFF
●	●	PC14			GTIOC0A								OP3N	OFF
●	●	PC15			GTIOC1A								OP3OUT	OFF
●	●	PD1											OP2OUT	OFF
●	●	PD2											OP2N	OFF
●	●	PD3											OP2P	OFF
●	●	PD4											OP1OUT	OFF
●	●	PD5										ADTRG3	OP1N	OFF
●	●	PD6											OP1P/CP1P1	OFF
●	○	PD7										ADTRG1	ADC1_AN0	OFF
●	○	PD8											ADC1_AN1	OFF
●	○	PD9							TXD3				ADC1_AN2	OFF
●	○	PD13								SCK1			ADC1_AN6	OFF
●	○	PD14							TXD1	MOSI1	SCL1		ADC1_AN7	OFF
●	●	PD15							RXD1	MISO1	SDA1		ADC3_AN0/CP2P2	OFF
●	●	PE0						QEA					ADC3_AN1/CP2P1	OFF
●	●	PE1						QEB					ADC3_AN2/CP2P0	OFF
●	●	PE2						INDEX					ADC3_AN3/CP2N	OFF
●	●	PE3					T5					CP2OUT		OFF
●	○	PE4					T6							OFF
●	○	PE5					T7						CP1N	OFF
●	○	PE6					T8					CP1OUT		OFF
●	○	PE7			GTPOE				TXD1			CP3OUT		OFF
○	●	PE8												
●	●	PE9												OFF

5. 引脚描述

引脚编号	类型	说明
I/O端口		
PA.n	I/O	PA组双向I/O端口 ***注：****PA.n引脚编号并不连续，具体参考"引脚定义"，下同。*
PB.n	I/O	PB组双向I/O端口
PC.n	I/O	PC组双向I/O端口
PD.n	I/O	PD组双向I/O端口
PE.n	I/O	PE组双向I/O端口 ***注：****LQFP64封装：可用PE0 - PE7，PE9（共9个引脚，无PE8）* *TQFP48封装：可用PE0 - PE3，PE8，PE9（共6个引脚，含PE8）*
通用PWM定时器（GPT0 - GPT2，支持多通道输入捕获/输出比较/PWM或脉冲输出）		
GTIOC0A	I/O	GPT定时器0通道A输入捕获/输出比较/PWM或脉冲输出
GTIOC0B	I/O	GPT定时器0通道B输入捕获/输出比较/PWM或脉冲输出
GTIOC1A	I/O	GPT定时器1通道A输入捕获/输出比较/PWM或脉冲输出
GTIOC1B	I/O	GPT定时器1通道B输入捕获/输出比较/PWM或脉冲输出
GTIOC2A	I/O	GPT定时器2通道A输入捕获/输出比较/PWM或脉冲输出
GTIOC2B	I/O	GPT定时器2通道B输入捕获/输出比较/PWM或脉冲输出
GTPOE	I	GPT故障保护信号输入
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（TIM5 - TIM8）		
T5 - T8	I/O	外部时钟输入/时钟比较输出
MCM（三相电机控制模块）		
MCM1_PWM0/PWM01 MCM1_PWM1/PWM11 MCM1_PWM2/PWM21	O	MCM1模块16BIT PWM互补输出
MCM1_FLT	I	MCM1模块16BIT PWM模块故障输入信号
MCM1_SC1	I	MCM1模块逐波限流输入比较数字信号1
MCM1_SC2	I	MCM1模块逐波限流输入比较数字信号2
MCM1_SC3	I	MCM1模块逐波限流输入比较数字信号3
ADC（ADC1/ADC3，多通道模数转换器）		
ADC1_ANn	I	ADC1的5个独立输入通道 ***注：****AN.n引脚编号并不连续，具体参考"引脚定义"，下同。*
ADC3_ANn	I	ADC3的6个独立输入通道
VREF	I/O	ADC内部参考电压滤波电容接口，或内部参考电压输出接口，或外部参考电压输入接口（如使用内部参考电压，请在此引脚上接0.01uF电容并联1uF电容，把1uF电容增大有助于改善稳压效果，但不建议超过10uF）
ADTRG1/ADTRG3	I	ADC外部采样触发信号输入

续上表

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，内建30kΩ上拉电阻连接到V_{DD}，所以仅接一个外部0.1uF电容即可产生复位脉冲（如需再加外部上拉电阻，请加4.7kΩ - 10kΩ）
BOOT	I	保留用作启动模式选择引脚。 ***注：****此引脚仅在提供ISP功能的产品中有意义，BOOT=0为正常从闪存启动，BOOT=1为从系统存储器启动。*
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）
AGND	P	模拟接地
AV_{DD}	P	模拟电源（2.7 - 5.5V）
UART（UART1/2/3，通用同步异步串行口）		
RXD1/2/3	I/O	串口1/2/3数据输入
TXD1/2/3	O	串口1/2/3数据输出
SPI（SPI1，串行外设接口）		
MOSI1	I/O	SPI1主输出从输入引脚
MISO1	I/O	SPI1主输入从输出引脚
SCK1	I/O	SPI1串行时钟引脚
SS1	I	SPI1从设备选择引脚
TWI（TWI1，两线串行接口，兼容I2C）		
SCL1	I/O	TWI1时钟信号（支持主机/从机模式）
SDA1	I/O	TWI1数据输入输出
片上模拟比较器(CP1/CP2/CP3)		
CP1P1（注） CP2P0/1/2 CP3P0/1	I	比较器1/2/3同相输入端 ***注：****CP1P0固定接OP1OUT，不支持外部引脚输入。*
CP1N/CP2N/CP3N	I	比较器1/2/3反向输入端
CP1OUT/CP2OUT/CP3OUT	O	比较器1/2/3输出端（数字信号）
片上模拟放大器（OP1/OP2/OP3）		
OP1P/OP2P/OP3P	I	放大器1 - 3同相输入端
OP1N/OP2N/OP3N	I	放大器1 - 3反向输入端
OP1OUT/OP2OUT/OP3OUT	O	放大器1 - 3输出端
调试接口（SWDP/JTAG，串行双线调试接口和标准JTAG接口）		
JTRST	I	JTAG复位信号 ***注1：****系统上电这5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5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。*
JTDI	I	JTAG数据输入
JTMS/SWDIO	I/O	JTAG模式选择/SWDP数据输入输出
JTCK/SWCLK	I	JTAG时钟输入/SWDP时钟输入
JTDO/SWO	O	JTAG数据输出/SWDP跟踪数据输出

6. 相关文档

- Cortex™-M3技术参考手册，可按下述链接下载：

http://infocenter.arm.com/help/topic/com.arm.doc.100165_0201_00_en/arm_cortexm3_processor_trm_100165_0201_00_en.pdf

- Cortex™-M3 Device通用用户指南（含架构、指令集、核内外设等），可按下述链接下载：

http://infocenter.arm.com/help/topic/com.arm.doc.dui0552a/DUI0552A_cortex_m3_dgug.pdf

- 以下参考文档和资料可从Sinowealth公司或其他关联渠道处获得，Sinowealth保留在文档更新时不另行通知的权利，请定期查询Sinowealth网站或关联渠道以获取版本更新信息及差异对照表。
  - ✓ 数据手册（Data Sheet）：即本手册，芯片最基础最权威的说明文档；
  - ✓ 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、常规用法介绍等；
  - ✓ 标准库手册（StdLib Manual）：与外设库配合的说明文档；
  - ✓ 开发工具包（Development Kit）：芯片开发入门所涉及的各项软硬件配套工具。包括针对主流IDE开发环境（Keil、IAR等）的插件、Project建立向导、电机专用辅助工具、基于本芯片的硬件开发评估平台等。

7. 内核

7.1 ARM®的 CortexTM-M3 核心

ARM®Cortex™-M3处理器是行业领先的32位RISC处理器内核，适用于具有较高确定性的实时应用，它经过专门开发，适用于广泛的高性能低成本平台应用（包括微控制器、工业控制系统以及无线网络和传感器）。Cortex™-M3处理器具有出色的计算性能以及对事件的优异系统响应能力，该处理器提供额外的代码效率，同时可应对实际中对低动态和静态功率需求的挑战。Cortex™-M3采用了哈佛结构，拥有独立的指令总线和数据总线，可以让取指与数据访问并行不悖，从而提升了性能。

SH32F284使用ARM®Cortex™-M3最新的r2p1版本内核。

有关内核的更多信息请参考ARM官方技术手册“Cortex-M3 Technical Reference Manual - Revision r2p1”。（汉译版本为“Cortex-M3技术参考手册”）

7.2 寄存器组

Cortex-M3处理器拥有R0-R15的寄存器组。其中R13作为堆栈指针SP。SP有两个，但在同一时刻只能有一个可用（Banked寄存器）。

7.2.1 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

Cortex-M3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，当呼叫一个子程序时，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

Cortex-M3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 Cortex-M3 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

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，Cortex-M3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**	**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 基础级别=0 NMI=2 SVCall = 11 INTISR[0]=16 INTISR[1]=17 . . INTISR[15]=31 . . INTISR[239]=255

7.3.1.3 执行PSR

执行PSR（EPSR）包含两个重叠的区域：

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。

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/IT		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 -	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

25		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，CM3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。

	(Cortex‐M3的 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

Cortex-M3 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

Cortex-M3 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

7.5.1 16 位 Cortex-M3 指令汇总

汇编指令	操作
ADC <Rd>, <Rm>	寄存器值与寄存器值及C标志相加
ADD <Rd>, <Rn>, #<immed_3>	3位立即数与寄存器值相加
ADD <Rd>, #<immed_8>	8位立即数与寄存器值相加
ADD <Rd>, <Rn>, <Rm>	低寄存器值与低寄存器值相加
ADD <Rd>, <Rm>	高寄存器值与低或高寄存器值相加
ADD <Rd>, PC, #<immed_8>*4	PC加4（8位立即数）
ADD <Rd>, SP, #<immed_8>*4	SP 加4（8位立即数）
ADD <Rd>, SP, #<immed_7>*4或 ADD SP, SP, #<immed_7>*4	SP 加4（7位立即数）
AND <Rd>, <Rm>	寄存器值按位与
ASR <Rd>, <Rm>, #<immed_5>	算术右移，移位次数取决于立即数值
ASR <Rd>, <Rs>	算术右移，移位次数取决于寄存器中的值
B<cond><target address>	条件分支
B<tartet address>	无条件分支
BIC <Rd>, <Rs>	位清零
BKPT <immed_8>	软件断点
BL <Rm>	带链接分支
CBNZ <Rn>, <label>	比较结果不为零时分支
CBZ <Rn>, <Rm>	比较结果为零时分支
CMN <Rn>, <Rm>	将寄存器值取反与另一个寄存器值比较
CMP <Rn>, #<immed_8>	与8位立即数比较
CMP <Rn>, <Rm>	寄存器比较
CMP <Rn>, <Rm>	高寄存器与高或低寄存器比较
CPS <effect>, <iflags>	改变处理器状态
CPY <Rd>, <Rm>	将高或低寄存器的值复制到另一个高或低寄存器中
EOR <Rd>, <Rm>	寄存器的值按位异或
IT<cond> IT<x><cond> IT<x><y><cond> IT<x><y><z><cond>	以下一条指令为条件，以下面两条指令为条件，以下面三条指令为条件，以下面四条指令为条件
LDMIA <Rn>!, <register>	多个连续的存储器字加载
LDR <Rd>, [<Rn>, #<immed_5*4>]	将基址寄存器与5位立即数偏移的和的地址处的数据加载到寄存器中
LDR <Rd>, [<Rn>, <Rm>]	将基址寄存器与寄存器偏移的和的地址处的数据加载到寄存器中
LDR <Rd>, [PC, #<immed_8>*4]	将PC 与8位立即数偏移的和的地址处的数据加载到寄存器中
LDR <Rd>, [SP, #<immed_8>*4]	将SP与 8位立即数偏移的和的地址处的数据加载到寄存器中
LDRB <Rd>, [<Rn>, #<immed_5>]	将寄存器与5位立即数偏移的和的地址处的字节[7:0]加载到寄存器中
LDRB <Rd>, [<Rn>, <Rm>]	将寄存器与寄存器偏移的和的地址处的字节[7:0]加载到寄存器中
LDRH <Rd>, [<Rn>, #<immed_5>*2]	将寄存器与 5 位立即数偏移的和的地址处的半字[15:0]加载到寄存器中
LDRH <Rd>, [<Rn>, <Rm>]	将寄存器与寄存器偏移的和的地址处的半字[15:0]加载到寄存器中
LDRSB <Rd>, [<Rn>, <Rm>]	将寄存器与寄存器偏移的和的地址处的带符号字节[7:0]加载到寄存器中
LDRSH <Rd>, [<Rn>, <Rm>]	将寄存器与寄存器偏移的和的地址处的带符号半字[15:0]加载到寄存器中
LSL <Rd>, <Rm>, #<immed_5>	逻辑左移，移位次数取决于立即数值
LSL <Rd>, <Rs>	逻辑左移，移位次数取决于寄存器中的值

LSR <Rd>, <Rm>, #<immed_5>	逻辑右移，移位次数取决于立即数值
LSR <Rd>, <Rs>	逻辑右移，移位次数取决于寄存器中的值
MOV <Rd>, #<immed_8>	将8位立即数传送到目标寄存器
MOV <Rd>, <Rn>	将低寄存器值传送给低目标寄存器
MOV <Rd>, <Rm>	将高或低寄存器值传送给高或低目标寄存器
MUL <Rd>, <Rm>	寄存器值相乘
MVN <Rd>, <Rm>	将寄存器值取反后传送给目标寄存器
NEG <Rd>, <Rm>	将寄存器值取负并保存在目标寄存器中
NOP <C>	无操作
ORR <Rd>, <Rm>	将寄存器值按位作逻辑或操作
POP <寄存器>	寄存器出栈
POP <寄存器，PC>	寄存器和PC出栈
PUSH <registers>	寄存器压栈
PUSH <registers, LR>	寄存器和LR压栈
REV <Rd>, <Rn>	将字内的字节逆向（reverse）并复制到寄存器中
REV16 <Rd>, <Rn>	将两个半字内的字节逆向并复制到寄存器中
REVSH <Rd>, <Rn>	将低半字[15:0]内的字节逆向并将符号位扩展，复制到寄存器中。
ROR <Rd>, <Rs>	循环右移，移位次数由寄存器中的值标识
SBC <Rd>, <Rm>	寄存器中的值减去寄存器值和C标志
SEV <c>	发送事件
STMIA <Rn>!, <registers>	将多个寄存器字保存到连续的存储单元中
STR <Rd>, [<Rn>, #<immed_5>*4]	将寄存器字保存到寄存器与5位立即数偏移的和的地址中
STR <Rd>, [<Rn>, <Rm>]	将寄存器字保存到寄存器地址中
STR <Rd>, [SP, #<immed_8> * 4]	将寄存器字保存到SP与8位立即数偏移的和的地址中
STRB <Rd>, [<Rn>, #<immed_5>]	将寄存器字节[7:0]保存到寄存器与5位立即数偏移的和的地址中
STRB <Rd>, [<Rn>, <Rm>]	将寄存器字节[7:0]保存到寄存器地址中
STRH <Rd>, [<Rn>, #<immed_5> * 2]	将寄存器半字[15:0]保存到寄存器与5位立即数偏移的和的地址中
STRH <Rd>, [<Rn>, #<immed_5> * 2]	将寄存器半字[15:0]保存到寄存器地址中
STRH <Rd>, [<Rn>, #<immed_5> * 2]	寄存器值减去3位立即数
SUB <Rd>, #<immed_8>	寄存器值减去8位立即数
SUB <Rd>, <Rn>, <Rm>	寄存器值减去寄存器值
SUB SP, #<immed_7> * 4	SP减4（7位立即数）
SVC <immed_8>	操作系统服务调用，带8位立即数调用代码
SXTB <Rd>, <Rm>	从寄存器中提取字节[7:0]，传送到寄存器中，并用符号位扩展到32位
SXTH <Rd>, <Rm>	从寄存器中提取半字[15:0]，传送到寄存器中，并用符号位扩展到32位
TST <Rn>, <Rm>	将寄存器与另一个寄存器相与，测试寄存器中的置位的位
UXTB <Rd>, <Rm>	从寄存器中提取字节[7:0]，传送到寄存器中，并用零位扩展到32位
UXTH <Rd>, <Rm>	从寄存器中提取半字[15:0]，传送到寄存器中，并用零位扩展到32位
WFE <c>	等待事件
WFI <c>	等待中断

7.5.2 32 位 Coxtex-M3 指令汇总

汇编指令	操作
ADC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12>	寄存器值与12位立即数及C位相加
ADC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}	寄存器值与移位后的寄存器值及C位相加
ADD{S}.W <Rd>, <Rn>,#<modify_constant(immed_12)>	寄存器值与12位立即数相加
ADD{S}.W <Rd>, <Rm>{, <shift>}	寄存器值与移位后的寄存器值相加
ADDW.W <Rd>, <Rn>, #<immed_12>	寄存器值与12位立即数相加
AND{S}.W <Rd>, <Rn>, #<modify_constant(immed_12>	寄存器值与12位立即数按位与
AND{S}.W <Rd>, <Rn>, Rm>{, <shift>}	寄存器值与移位后的寄存器值按位与
ASR{S}.W <Rd>, <Rn>, <Rm>	算术右移，移位次数取决于寄存器值
B{cond}.W <label>	条件分支
BFC.W <Rd>, #<lsb>, #<width>	位区清零
BFI.W <Rd>, <Rn>, #<lsb>, #<width>	将一个寄存器的位区插入另一个寄存器中
BIC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>	12位立即数取反与寄存器值按位与
BIC{S}.W <Rd>, <Rn>, {, <shift>}	移位后的寄存器值取反与寄存器值按位与
BL <label>	带链接的分支
BL<c><label>	带链接的分支（立即数）
B.W <label>	无条件分支
CLZ.W <Rd>, <Rn>	返回寄存器值中零的数目
CMN.W <Rn>, #<modify_constant(immed_12)>	寄存器值与12位立即数两次取反后的值比较
CMN.W <Rn>, <Rm>{, <shift>}	寄存器值与移位后的寄存器值两次取反后的值比较
CMP.W <Rn>, #<modify_constant(immed_12)>	寄存器值与12位立即数比较
CMP.W <Rn>, <Rm>{, <shift>}	寄存器值与移位后的寄存器值比较
DMB <c>	数据存储器排序（barrier）
DSB <c>	数据同步排序（barrier）
EOR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>	寄存器值与12位立即数作异或操作
EOR{S}.W <Rd>, <Rn>, <Rm>{, <shift>}	寄存器值与移位后的寄存器值作异或操作
ISB <c>	指令同步排序（barrier）
LDM{IA|DB}.W <Rn>{!}, <registers>	多存储器寄存器加载，加载后加1或加载前减1
LDR.W <Rxf>, [<Rn>, #<offset_12>]	保存寄存器地址与12位立即数偏移的和的地址处的数据字
LDR.W PC, [<Rn>, #<offset_12>]	将寄存器地址与12位立即数偏移的和的地址处的数据字保存到PC中
LDR.W PC, #<+/-<offset_8>	将基址寄存器地址的8位立即数偏移的地址处的数据字保存到PC中，后索引
LDR.W <Rxf>, [<Rn>], #+/–<offset_8>	保存基址寄存器地址的8位立即数偏移的地址处的数据字，后索引
LDR.W <Rxf>, [<Rn>, #<+/–<offset_8>]!	保存基址寄存器地址的8位立即数偏移的地址处的数据字，前索引
LDR.W PC, [<Rn>, #+/–<offset_8>]!	将基址寄存器地址的8位立即数偏移的地址处的数据字保存到PC中，前索引
LDR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]	保存寄存器地址左移0，1，2或3个位置后的地址处的数据字
LDR.W PC, [<Rn>, <Rm>{, LSL #<shift>}]	将寄存器地址左移0，1，2或3个位置后的地址处的数据字保存到PC中
LDR.W <Rxf>, [PC, #+/–<offset_12>]	保存PC地址的12位立即数偏移的地址处的数据字
LDR.W PC, [PC, #+/–<offset_12>]	将PC地址的12位立即数偏移的地址处的数据字保存到PC中
LDRB.W <Rxf>, [<Rn>, #<offset_12>]	保存基址寄存器地址与12位立即数偏移的和的地址处的字节[7:0]
LDRB.W <Rxf>. [<Rn>], #+/-<offset_8>	保存基址寄存器地址的8位立即数偏移的地址处的字节[7:0]，后索引
LDRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]	保存寄存器地址左移0，1，2或3个位置后的地址处的字节[7:0]
LDRB.W <Rxf>, [<Rn>, #<+/–<offset_8>]!	保存基址寄存器地址的8位立即数偏移的地址处的字节[7:0]，前索引
LDRB.W <Rxf>, [PC, #+/–<offset_12>]	保存PC地址的12位立即数偏移的地址处的字节

LDRD.W <Rxf>, <Rxf2>, [<Rn>, #+/–<offset_8> * 4]{!}	保存寄存器地址8位偏移4的地址处的双字，前索引
LDRD.W <Rxf>, <Rxf2>, [<Rn>], #+/–<offset_8> * 4	保存寄存器地址8位偏移4的地址处的双字，后索引
LDRH.W <Rxf>, [<Rn>, #<offset_12>]	保存基址寄存器地址与12位立即数偏移的和的地址处的半字[15:0]
LDRH.W <Rxf>, [<Rn>, #<+/–<offset_8>]!	保存基址寄存器地址的8位立即数偏移的地址处的半字[15:0]，前索引
LDRH.W <Rxf>. [<Rn>], #+/-<offset_8>	保存基址寄存器地址的8位立即数偏移的地址处的半字[15:0]，后索引
LDRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]	保存基址寄存器地址左移0，1，2或3个位置后的地址处的半字[15:0]
LDRH.W <Rxf>, [PC, #+/–<offset_12>]	保存PC地址的12位立即数偏移的地址处的半字
LDRSB.W <Rxf>, [<Rn>, #<offset_12>]	保存基址寄存器地址与12位立即数偏移的和的地址处的带符号字节[7:0]
LDRSB.W <Rxf>. [<Rn>], #+/-<offset_8>	保存基址寄存器地址的8位立即数偏移的地址处的带符号字节[7:0]，后索引
LDRSB.W <Rxf>, [<Rn>, #<+/–<offset_8>]!	保存基址寄存器地址的8位立即数偏移的地址处的带符号字节[7:0]，前索引
LDRSB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]	保存寄存器地址左移0，1，2或3个位置后的地址处的带符号字节[7:0]
LDRSB.W <Rxf>, [PC, #+/–<offset_12>]	保存PC地址的12位立即数偏移的地址处的带符号字节
LDRSH.W <Rxf>, [<Rn>, #<offset_12>]	保存基址寄存器地址与12位立即数偏移的和的地址处的带符号半字[15:0]
LDRSH.W <Rxf>. [<Rn>], #+/-<offset_8>	保存基址寄存器地址的8位立即数偏移的地址处的带符号半字[15:0]，后索引
LDRSH.W <Rxf>, [<Rn>, #<+/–<offset_8>]!	保存基址寄存器地址的8位立即数偏移的地址处的带符号半字[15:0]，前索引
LDRSH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]	保存寄存器地址左移0，1，2或3个位置后的地址处的带符号半字[15:0]
LDRSH.W <Rxf>, [PC, #+/–<offset_12>]	保存PC地址的12位立即数偏移的地址处的带符号半字
LSL{S}.W <Rd>, <Rn>, <Rm>	逻辑左移，移位次数由寄存器中的值标识
LSR{S}.W <Rd>, <Rn>, <Rm>	逻辑右移，移位次数由寄存器中的值标识
MLA.W <Rd>, <Rn>, <Rm>, <Racc>	将两个带符号或无符号的寄存器值相乘，并将低32位与寄存器值相加
MLS.W <Rd>, <Rn>, <Rm>, <Racc>	将两个带符号或无符号的寄存器值相乘，并将低32位与寄存器值相减
MOV{S}.W <Rd>, #<modify_constant(immed_12)>	将12位立即数传送到寄存器中
MOV{S}.W <Rd>, <Rm>{, <shift>}	将移位后的寄存器值传送到寄存器中
MOVT.W <Rd>, #<immed_16>	将16位立即数传送到寄存器的高半字[31:16]中
MOVW.W <Rd>, #<immed_16>	将16位立即数传送到寄存器的低半字[15:0]中，并将高半字[31:16]清零
MRS<c><Rd>, <psr>	将状态传送到寄存器中
MSR<c><psr>_<fields>,<Rn>	传送到状态寄存器中
MUL.W <Rd>, <Rn>, <Rm>	将两个带符号或不带符号的寄存器值相乘
NOP.W	无操作
ORN{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>	将寄存器值与12位立即数作逻辑“或非”操作
ORN[S}.W <Rd>, <Rn>, <Rm>{, <shift>}	将寄存器值与移位后的寄存器值作逻辑“或非”操作
ORR{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)	将寄存器值与12位立即数作逻辑“或”操作
ORR{S}.W <Rd>, <Rn>, <Rm>{, <shift>}	将寄存器值与移位后的寄存器值作逻辑 “或”操作
RBIT.W <Rd>, <Rm>	将位顺序逆向
REV.W <Rd>, <Rm>	将字内的字节逆向
REV16.W <Rd>, <Rn>	将每个半字内的字节逆向
REVSH.W <Rd>, <Rn>	将低半字内的字节逆向并用符号扩展
ROR{S}.W <Rd>, <Rn>, <Rm>	循环右移，移位次数取决于寄存器中的值
RSB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>	寄存器值与12位立即数相减

RSB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}	寄存器值与移位后的寄存器值相减
SBC{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>	寄存器值与12位立即数及C位相减
SBC{S}.W <Rd>, <Rn>, <Rm>{, <shift>}	寄存器值与移位后的寄存器值及C位相减
SBFX.W <Rd>, <Rn>, #<lsb>, #<width>	将所选的位复制到寄存器中并用符号扩展
SDIV<c><Rd>,<Rn>,<Rm>	带符号除法
SEV<c>	发送事件
SMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm>	将带符号半字相乘并用符号扩展到2个寄存器值
SMULL.W <RdLo>, <RdHi>, <Rn>, <Rm>	两个带符号寄存器值相乘
SSAT <c><Rd>, #<imm>, <Rn>{, <shift>}	带符号饱和操作
STM{IA|DB}.W <Rn>{!}, <registers>	多个寄存器字保存到连续的存储单元中
STR.W <Rxf>, [<Rn>, #<offset_12>]	寄存器字保存到寄存器地址与12位立即数偏移的和的地址中
STR.W <Rxf>, [<Rn>], #+/–<offset_8>	寄存器字保存到寄存器地址的8位立即数偏移的地址中，后索引
STR.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]	寄存器字保存到寄存器地址移位0，1，2或3个位置的地址中
STR{T}.W <Rxf>, [<Rn>, #+/–<offset_8>]{!}	寄存器字保存到寄存器地址的8位立即数偏移的地址中，前索引
STRB{T}.W <Rxf>, [<Rn>, #+/–<offset_8>]{!}	寄存器字节[7:0]保存到寄存器地址的8位立即数偏移的地址中，前索引
STRB.W <Rxf>, [<Rn>, #<offset_12>]	寄存器字节[7:0]保存到寄存器地址与12位立即数偏移的和的地址中
STRB.W <Rxf>, [<Rn>], #+/–<offset_8>	寄存器字节[7:0]保存到寄存器地址的8位立即数偏移的地址中，后索引
STRB.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]	寄存器字节保存到寄存器地址移位0，1，2或3个位置的地址中
STRD.W <Rxf>, <Rxf2>, [<Rn>, #+/–<offset_8> * 4]{!}	存储双字，前索引
STRD.W <Rxf>, <Rxf2>, [<Rn>], #+/–<offset_8> * 4	存储双字，后索引
STRH.W <Rxf>, [<Rn>, #<offset_12>]	寄存器半字[15:0]保存到寄存器地址与12位立即数偏移的和的地址中
STRH.W <Rxf>, [<Rn>, <Rm>{, LSL #<shift>}]	寄存器半字保存到寄存器地址移位0，1，2或3个位置的地址中
STRH{T}.W <Rxf>, [<Rn>, #+/–<offset_8>]{!}	寄存器半字保存到寄存器地址的8位立即数偏移的地址中，前索引
STRH.W <Rxf>, [<Rn>], #+/–<offset_8>	寄存器半字保存到寄存器地址的8位立即数偏移的地址中，后索引
SUB{S}.W <Rd>, <Rn>, #<modify_constant(immed_12)>	寄存器值与12位立即数相减
SUB{S}.W <Rd>, <Rn>, <Rm>{, <shift>}	寄存器值与移位后的寄存器值相减
SUBW.W <Rd>, <Rn>, #<immed_12>	寄存器值与12位立即数相减
SXTB.W <Rd>, <Rm>{, <rotation>}	将字节符号扩展到32位
SXTH.W <Rd>, <Rm>{, <rotation>}	将半字符号扩展到32位
TBB [<Rn>, <Rm>]	表格分支字节
TBH [<Rn>, <Rm>, LSL #1]	表格分支半字
TEQ.W <Rn>, #<modify_constant(immed_12)>	寄存器值与12位立即数作逻辑“异或”操作
TEQ.W <Rn>, <Rm>{, <shift}	寄存器值与移位后的寄存器值作逻辑 “异或”操作
TST.W <Rn>, #<modify_constant(immed_12)>	寄存器值与12位立即数作逻辑“与”操作
TST.W <Rn>, <Rm>{, <shift>}	寄存器值与移位后的寄存器值作逻辑“与”操作
UBFX.W <Rd>, <Rn>, #<lsb>, #<width>	将寄存器的位区复制到寄存器中，并用零扩展到32位
UMLAL.W <RdLo>, <RdHi>, <Rn>, <Rm>	无符号除法 UDIV<c><Rd>,<Rn>,<Rm>两个无符号寄存器值相乘并与两个寄存器值相加
USAT <c><Rd>, #<imm>, <Rn>{, <shift>}	两个无符号寄存器值相乘 UMULL.W <RdLo>, <RdHi>, <Rn>, <Rm>无符号饱和操作
UXTB.W <Rd>, <Rm>{, <rotation>}	将无符号字节复制到寄存器中并用零扩展到32位
UXTH.W <Rd>, <Rm>{, <rotation>}	将无符号半字复制到寄存器中并用零扩展到32位
WFE.W	等待事件
WFI.W	等待中断

8. 存储器和总线架构

ARM® Cortex®-M3的总线接口基于AHB-Lite和APB协议，它们的规则参考“AMBA2.0 规格（第4版）”。

8.1 系统架构

SH32F284主系统由多层AHB总线矩阵互联和驱动，如下图所示：

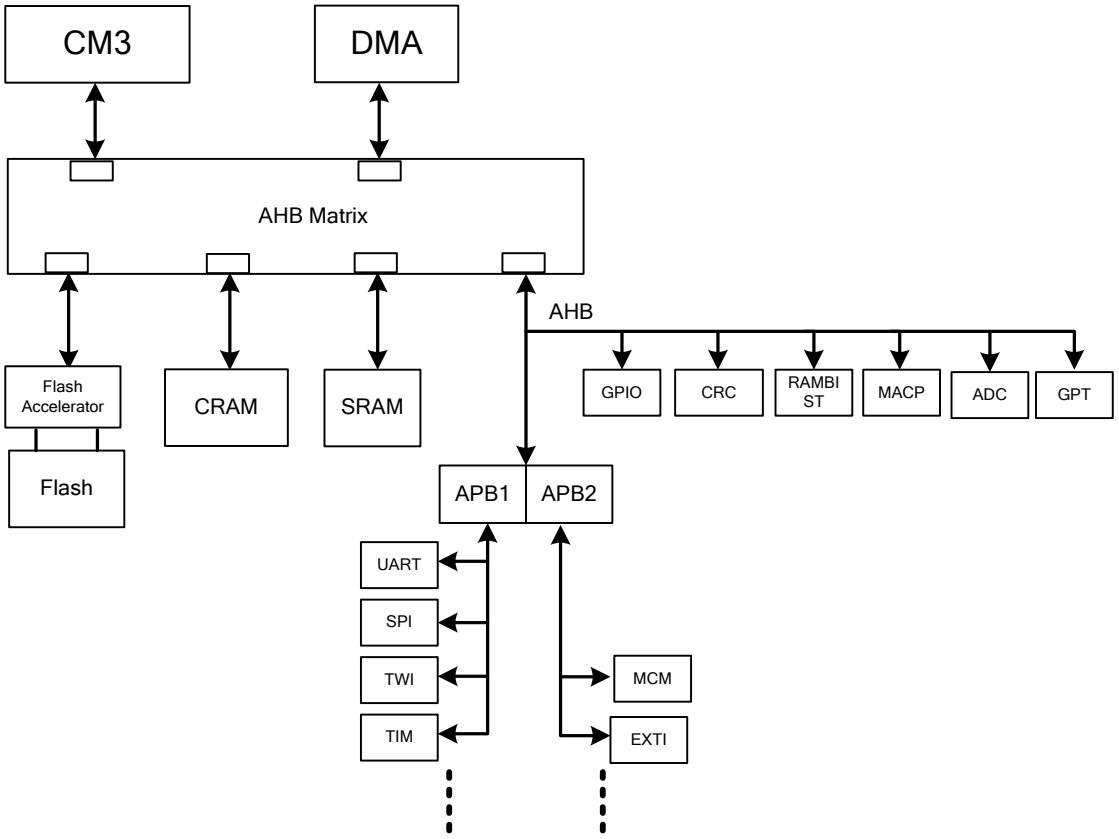


图8-1 SH32F284 系统架构框图

8.1.1 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。

8.1.2 AHB/APB 桥

两个AHB-APB桥在AHB和2个APB总线间提供同步连接。APB2和APB1都是专用外设总线，前者要求设置为AHB时钟的二分之一或更低，后者要求设置为AHB时钟的四分之一或更低。

在使用一个外设之前，必须设置时钟寄存器AHBENR/APB1ENR/APB2ENR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

SH32F284遵循ARM® Cortex™-M3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 SH32F284 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 17FF	片上EEProm-Like存储区（共6K）
		保留区域	保留区域
		0x0FFF F800 – 0x0FFF FFFF	片上OTP区（共2K）
0x1000 0000 – 0x1FFF FFFF	片上易失性数据存储区2（CRAM）	0x1000 0000 – 0x1000 3FFF	片上16K CRAM（可作为DRAM）
0x2000 0000 – 0x3FFF FFFF	片上易失性数据存储区1（SRAM）	0x2000 0000 – 0x2000 1FFF	片上8K SRAM （位于数据位带区）
		0x2200 0000 – 0x23FF FFFF	数据位带别名区， 每一个地址映像为位带区的一个位
0x4000 0000 – 0x5FFF FFFF	片上外设区	0x4000 0000 – 0x4001 FFFF	片上APB1外设区，每个外设分配1K字节地址空间 （位于外设位带区）
		0x4002 0000 – 0x4003 FFFF	片上APB2外设区，每个外设分配1K字节地址空间 （位于外设位带区）
		0x4004 0000 – 0x400F FFFF	片上AHB外设区，每个外设分配1K字节地址空间 （位于外设位带区）
		0x4200 0000 – 0x43FF FFFF	外设位带别名区，每一个地址映像为位带区的一个位
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	Cortex-M3私有外设区	0xE000 0000 – 0xE000 0FFF	指令跟踪宏单元（ITM）
		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 – 0xE003 FFFF	保留区域
		0xE004 0000 – 0xE004 0FFF	跟踪端口接口单元（TPIU）

8.2.1 寄存器映像

表8-2列出了所有SH32F284中内置外设（不包含Cortex-M3内核设备）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：外设基地址+寄存器偏移地址。

表8-2 SH32F284 内置外设的基地址

基地址	外设	总线
0x4000 0000 ~ 0x4000 03FF	保留	APB1总线: 0x4000 0000 - 0x4001 F3FF
0x4000 0400 ~ 0x4000 07FF	TIM5	
0x4000 0800 ~ 0x4000 0BFF	TIM6	
0x4000 0C00 ~ 0x4000 0FFF	TIM7	
0x4000 1000 ~ 0x4000 13FF	TIM8	
0x4000 1400 ~ 0x4000 1BFF	保留	
0x4000 1C00 ~ 0x4000 1FFF	QEI	
0x4000 2000 ~ 0x4000 23FF	UART1	
0x4000 2400 ~ 0x4000 27FF	UART2	
0x4000 2800 ~ 0x4000 2BFF	UART3	
0x4000 2C00 ~ 0x4000 2FFF	SPI1	
0x4000 3000 ~ 0x4000 33FF	保留	

0x4000 3400 ~ 0x4000 37FF	TWI1	
0x4000 3800 ~ 0x4000 4BFF	保留	
0x4000 4C00 ~ 0x4000 4FFF	IWDT	
0x4000 5000 ~ 0x4000 53FF	WWDT	
0x4000 5400 ~ 0x4000 57FF	保留	
0x4000 5800 ~ 0x4000 5BFF	AMOC	
0x4001 F000 ~ 0x4001 F3FF	保留	
0x4002 0000 ~ 0x4002 03FF	MCM1	APB2总线: 0x4002 0000 - 0x4002 1FFF
0x4002 0400 ~ 0x4002 0FFF	保留	
0x4002 1000 ~ 0x4002 13FF	EXTI	
0x4002 1400 ~ 0x4002 1FFF	保留	
0x4004 0000 ~ 0x4004 001F	GPIOA_CFG	AHB总线: 0x4004 0000 - 0x4005 03FF
0x4004 0020 ~ 0x4004 003F	GPIOB_CFG	
0x4004 0040 ~ 0x4004 005F	GPIOC_CFG	
0x4004 0060 ~ 0x4004 007F	GPIOD_CFG	
0x4004 0080 ~ 0x4004 009F	GPIOE_CFG	
0x4004 0400 ~ 0x4004 041F	GPIOA_IOD	
0x4004 0420 ~ 0x4004 043F	GPIOB_IOD	
0x4004 0440 ~ 0x4004 045F	GPIOC_IOD	
0x4004 0460 ~ 0x4004 047F	GPIOD_IOD	
0x4004 0480 ~ 0x4004 049F	GPIOE_IOD	
0x4004 0800 ~ 0x4004 0FFF	保留	
0x4004 1000 ~ 0x4004 13FF	ADC1	
0x4004 1400 ~ 0x4004 17FF	保留	
0x4004 1800 ~ 0x4004 1BFF	ADC3	
0x4004 1C00 ~ 0x4004 1FFF	保留	
0x4004 2000 ~ 0x4004 23FF	GPT	
0x4004 2400 ~ 0x4004 27FF	GPT0	
0x4004 2800 ~ 0x4004 2BFF	GPT1	
0x4004 2C00 ~ 0x4004 2FFF	GPT2	
0x4004 3000 ~ 0x4004 3FFF	保留	
0x4004 4000 ~ 0x4004 43FF	SYSCFG	
0x4004 4400 ~ 0x4004 47FF	RCC	
0x4004 4800 ~ 0x4004 4BFF	DMA	
0x4004 4C00 ~ 0x4004 4FFF	保留	
0x4004 5000 ~ 0x4004 53FF	FLASH	
0x4004 5400 ~ 0x4004 57FF	MACP	
0x4004 5800 ~ 0x4004 5BFF	CRC	
0x4004 5C00 ~ 0x4004 5FFF	RAMBIST	
0x4005 0000 ~ 0x4005 03FF	保留	

8.2.2 数据存储器 1（SRAM）

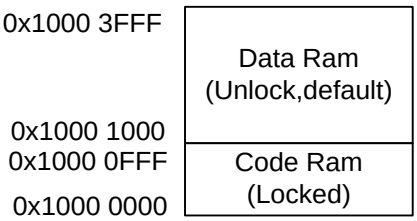
SH32F284内置8K数据RAM（SRAM）。它可以以字节、半字(16位)或全字(32位)访问。SRAM的起始地址是0x2000 0000。SRAM支持March检测算法。

8.2.3 数据存储器 2（CRAM）

SH32F284内置16K程序RAM（CRAM），该CRAM可作为Code Ram使用，也可作为Data Ram（本文简称为DRAM）使用，做Code Ram使用时用于加载核心程序，实现核心程序在RAM中运行，做Data Ram时用法和SRAM相同。CRAM的起始地址是0x1000 0000。

CRAM提供一个锁定方式用于保护程序代码，当设置为锁定方式时，CRAM变为只读，通过CRAMLCKn@SYSCFG_CRAMLOCK控制位设置。当不锁定时，CRAM就是普通的Data Ram。

如下例中通过设置4KB为Code Ram，其它保持为Data Ram，设置如下：



把核心程序加载到Code Ram的操作由用户软件完成，具体用法可参考SH32F284的技术文档。

CRAM支持March检测算法。

8.2.4 位带

Cortex™-M3存储器映像包括两个位带(bit-band)区。这两个位带区的每一位（1位）将映射到位带别名区中的一个字(32位)，在位带别名区写入一个字具有对位带区的目标位执行读-改-写操作的相同效果。

位带区：支持位带操作的地址区

位带别名区：对别名地址的访问最终作用到位带区的访问上（这中间有一个地址映像过程）

在SH32F284里，SRAM区和片上外设区全部都分配在位带区，即都支持位带操作。

■ 对SRAM 位带区的某个比特，记它所在字节地址为A,位序号为n(0<=n<=7)，则该比特在别名区的地址为：

位带别名地址 = 0x22000000 + ((A-0x20000000)*8+n)*4 = 0x22000000+(A-0x20000000)*32 + n*4

■ 对于片上外设位带区的某个比特，记它所在字节的地址为A,位序号为n(0<=n<=7)，则该比特在别名区的地址为：

位带别名地址 = 0x42000000 + ((A-0x40000000)*8+n)*4 = 0x42000000+(A-0x40000000)*32 + n*4

例子：

下面的例子说明如何映像SRAM区地址为0x20000300字节的位2：

位带别名地址= 0x22000000 + (0x300×32) + (2×4) = 0x22006008

对0x22006008地址的写操作与对SRAM中地址0x20000300字节的位2执行读-改-写操作有着相同的效果。读0x22006008地址返回SRAM中地址0x20000300字节的位2的值(0x01 或 0x00)。

注意：Cortex-M3中存储空间是按字节编址的，因此位带别名区的一个字（32位）占用4个字节地址，即根据位带映射得到的位带别名地址，其最低2位必然是0。

有关位带的更多信息请参考ARM官方手册”Cortex-M3 Technical Reference Manual - Revision r2p1”。（汉译版本为“Cortex-M3技术参考手册”）

8.2.5 程序存储器

SH32F284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64个2K字节的扇区

类EEPROM区为6K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2K字节，只能一次写入，写入后不可更改

有关Flash程序存储器的详细信息，请参考下一节“Flash程序存储器”。

8.3 Flash 程序存储器

8.3.1 特性

- Flash 主存储器包括 64 X 2KB 扇区，总共 128KB
- Flash 类 EEPROM 区包括 3 X 2KB 扇区，总共 6KB
- Flash OTP 区包括 1 X 2KB 扇区，总共 2KB
- Flash 写操作支持 32 位，16 位写入
- 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
- Flash 读保护功能，防止非法访问
- Flash 写保护功能，阻止意外操作
- Flash 客户安全码保护，更高层级的用户鉴定保护

8.3.2 简介

SH32F284 内置 128K 的可编程 Flash 主程序存储区（Main Program Memory Block），每个扇区 2048 字节，64 个扇区都可单独进行擦除。

6KB 的内置类 EEPROM 存储区，用于存放用户数据，每个扇区 2048 字节。

2KB 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。

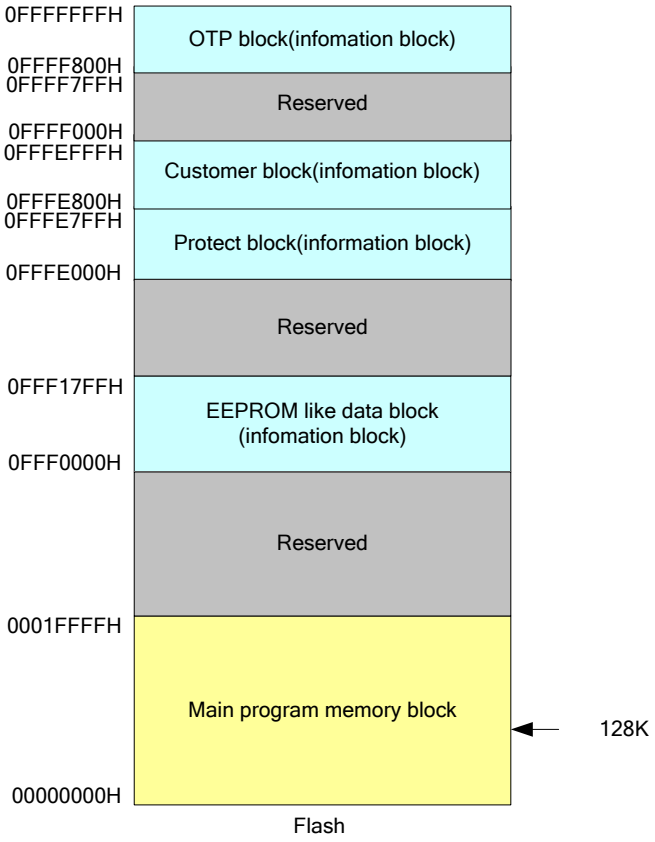


图8-2　　Flash 存储器结构

表8-3 Flash 存储器结构

Block	Name	Base address	Size（bytes）
Main memory	**Sector 0**	0x0000 0000 - 0x0000 07FF	2048
	Sector 1	0x0000 0800 - 0x0000 0FFF	2048
	Sector 2	0x0000 1000 - 0x0000 017FF	2048
	…	…	…
	…	…	…
	Sector 62	0x0001 F000 - 0x0001 F7FF	2048
	Sector 63	0x0001 F800 - 0x0001 FFFF	2048
Information Block	**EEPROM Block**	0x0FFF 0000 - 0x0FFF 17FF	6144
	Protect Block	0x0FFF E000 - 0x0FFF E7FF	2048
	Customer Block	0x0FFF E800 - 0x0FFF EFFF	2048
	OTP Block	0x0FFF F800 - 0x0FFF FFFF	2048
Flash memory Operation registers	**FLASH_ACR**	0x4002 8000 - 0x4002 8003	4
	FLASH_MKYR	0x4002 8004 - 0x4002 8007	4
	FLASH_E2KYR	0x4002 8008 - 0x4002 800B	4
	FLASH_SR	0x4002 800C - 0x4002 800F	4
	FLASH_CR	0x4002 8010 - 0x4002 8013	4
	Reversed	0x4002 8014 - 0x4002 802F	4
	FLASH_CNTR	0x4002 8030 - 0x4002 8033	4
	FLASH_UPCNTR	0x4002 8034 - 0x4002 8037	4
	FLASH_CNTCR	0x4002 8038 - 0x4002 803B	4
	Reversed	0x4002 803C - 0x4002 803F	4
	FLASH_IKYR	0x4002 8040 - 0x4002 8043	4
	Reversed	0x4002 8044 - 0x4002 80FF	188

8.3.3 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。对于 128KB 的 Flash 存储区，写保护的基本单位是 4 个扇区为一个保护单元。

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。

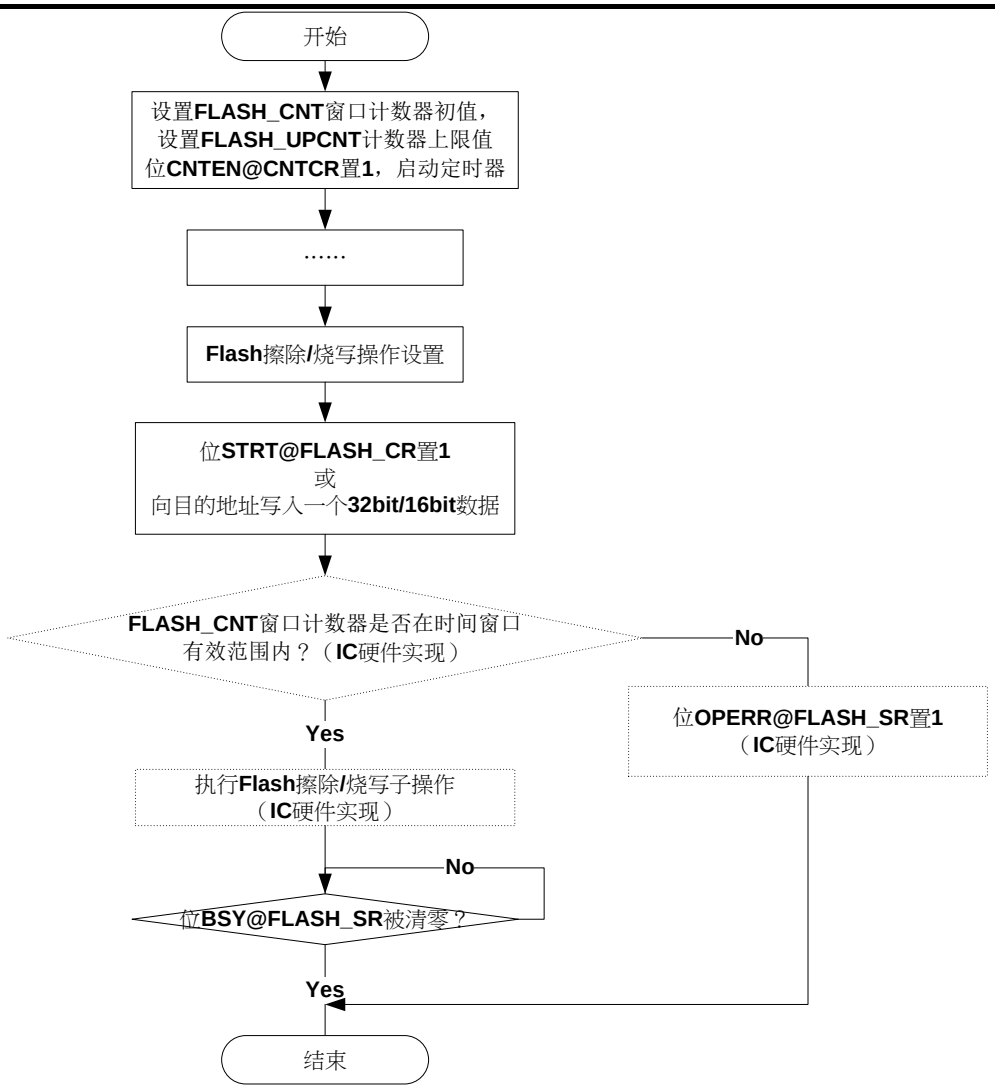


图8-3　　Flash 操作定时器使用流程图

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：

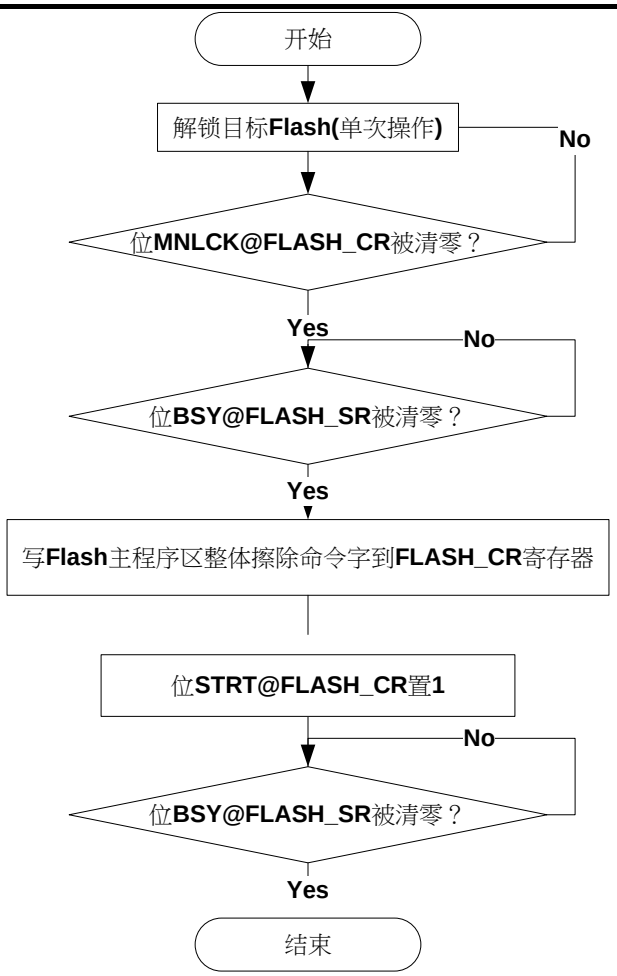


图8-4　　主程序区整体擦除

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：

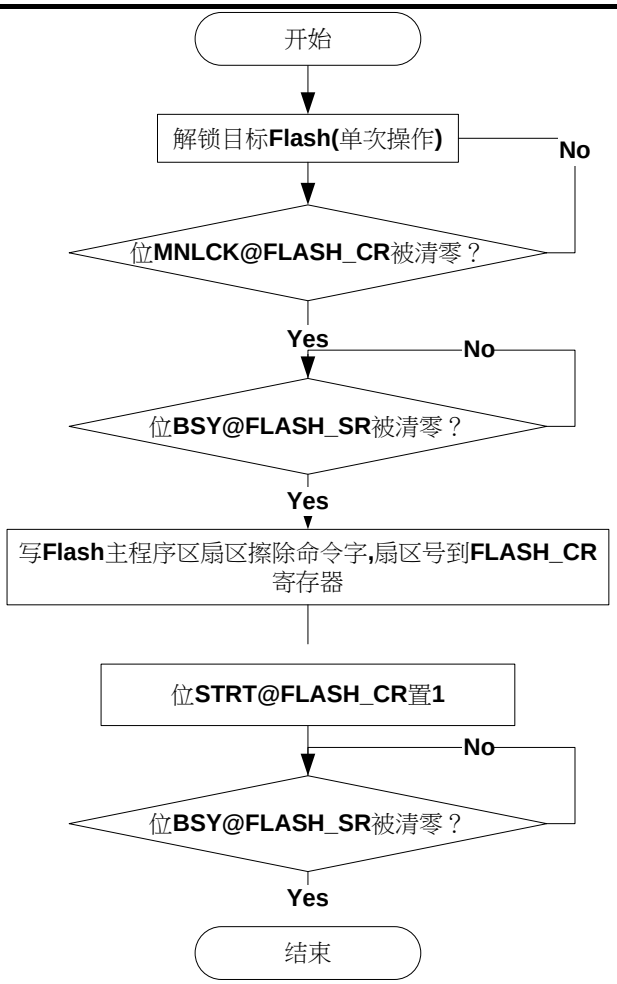


图8-5　　主程序区按扇区擦除（单一扇区）

（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：

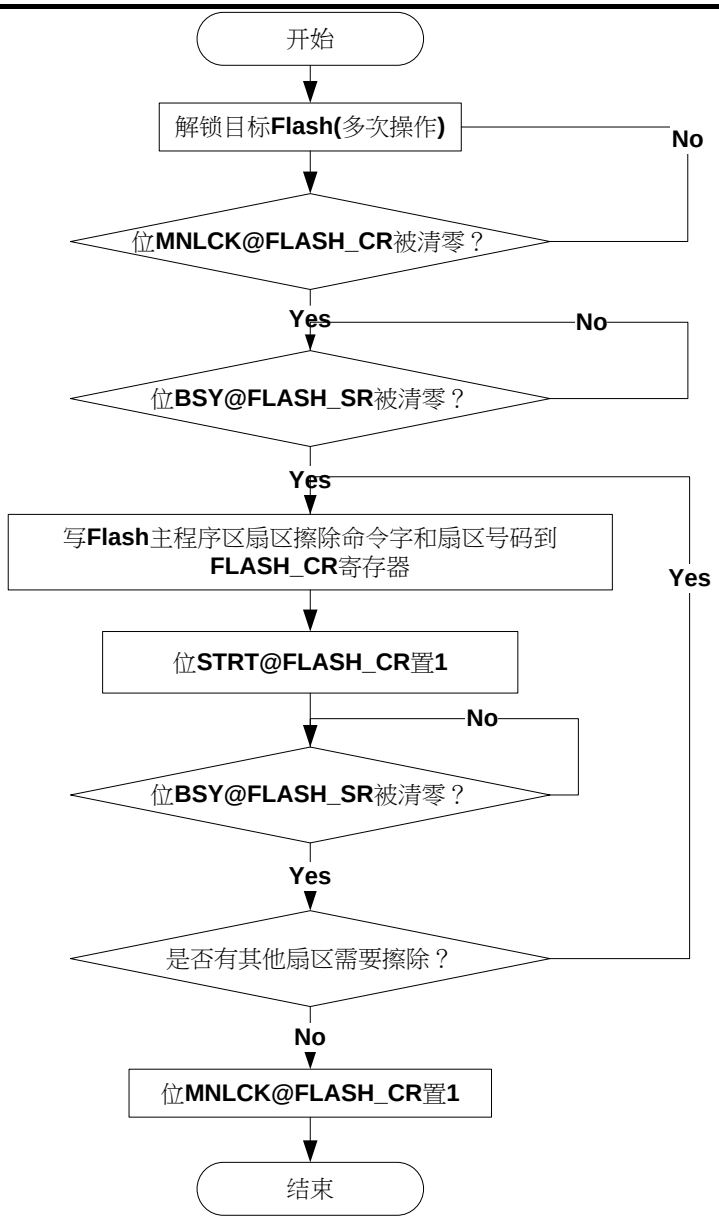


图8-6　　主程序区按扇区擦除（多个扇区）

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）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：

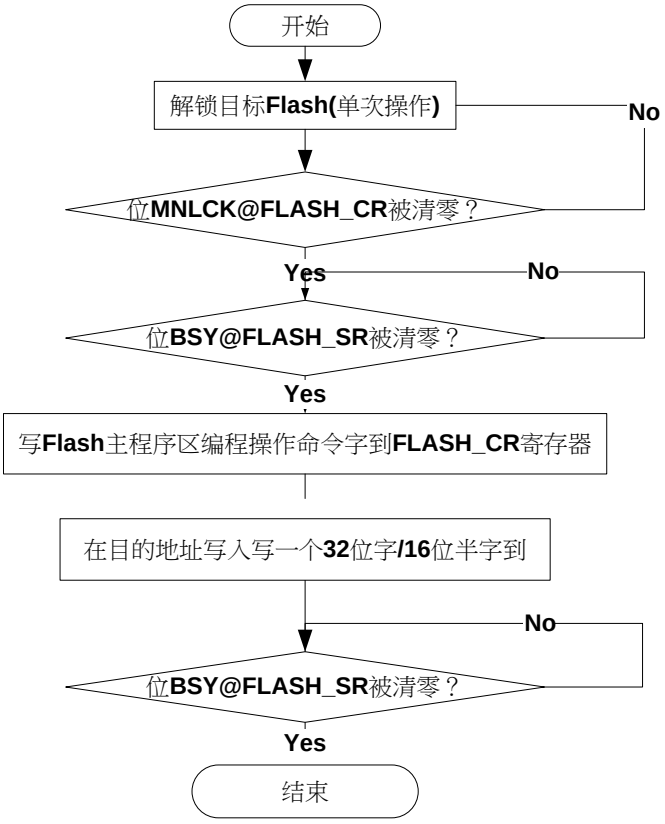


图8-7 主程序区编程（单字或半字）

（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）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。

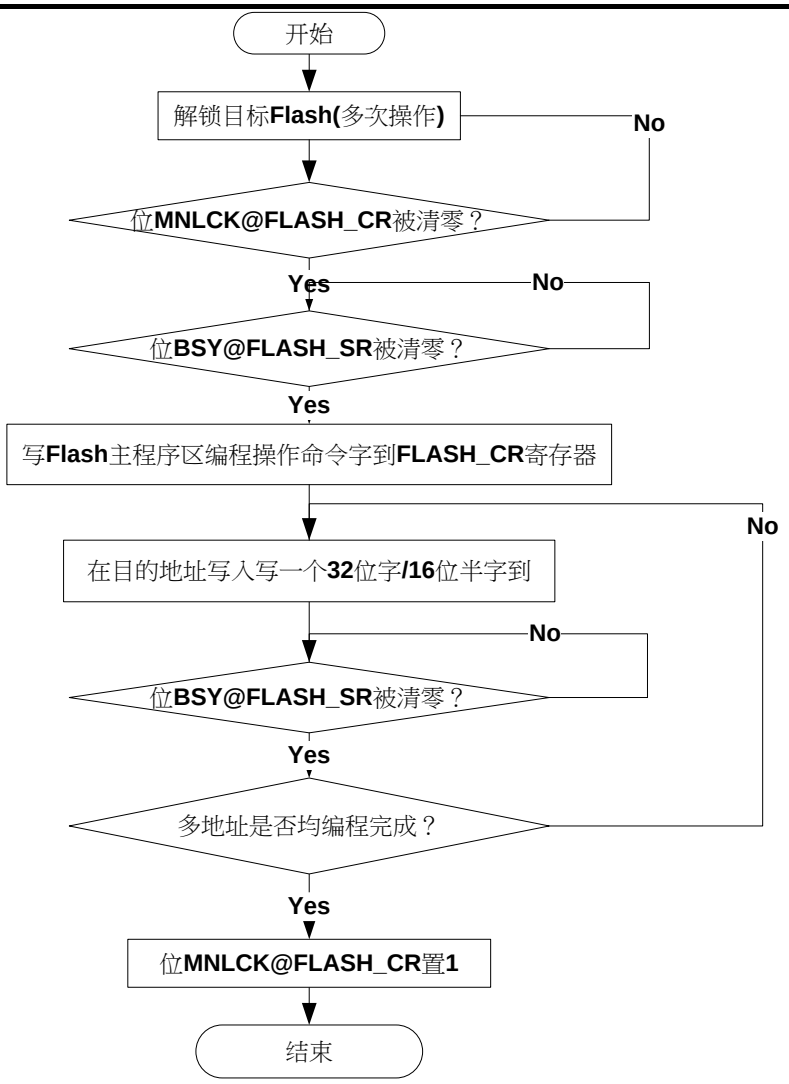


图8-8　　主程序区编程（连续写入多字编程）

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。

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 SH32F284 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。低级别读保护状态下，在调试模式下（JTAG/SWD）通过 DMA 或其它任何方式间接地访问 Flash 主存储器的操作也被禁止。

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。在低级别保护下，可以在调试模式下（JTAG/SWD）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（JTAG/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（JTAG/SWD）关闭。

(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[15:0]位为 1 可以使能指定扇区的写保护功能。SH32F284 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[15]	Sector62 ~ Sector63

（3）客户安全码

SH32F284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。

使用32位产品线量产工具（ProWriter-32）和仿真工具（SinoLink）都可对客户安全码进行设置，一旦设置完成，必须要安全码匹配才可进一步操作。

8.3.3.10　　访问加速控制

SH32F284增加了Flash访问加速控制模块，用以解决系统高主频与Flash存储器访问慢的矛盾。通过高速预读缓冲区，可以提前将指令及数据加载到缓冲区中，从而降低或消除Flash存储器与系统主频不匹配造成的影响。

对SH32F284而言，在系统84MHz主频运行时，要求设定Flash存储器等待周期为2（Flash LATENCY = 2），并开启预取缓冲区、指令缓冲区、数据缓冲区，可获得最佳执行性能，近似于Flash存储器零等待。

8.3.4　寄存器

FLASH 模块寄存器列表 (基地址:0x0x4004 5000)

地址	寄存器名	说明
0x4004 5000	ACR	Flash访问控制寄存器
0x4004 5004	MKYR	Flash主程序存储区解锁控制寄存器
0x4004 5008	E2KYR	类EEPROM存储区解锁控制寄存器
0x4004 500C	SR	Flash状态和清除寄存器
0x4004 5010	CR	Flash控制寄存器
0x4004 5020	FLASH_RPR	Flash读保护寄存器
0x4004 5024	FLASH_WRPR	Flash 写保护寄存器
0x4004 5030	CNTR	Flash操作定时器计数寄存器
0x4004 5034	UPCNTR	Flash操作定时器上限值
0x4004 5038	CNTCR	Flash操作定时器启动寄存器
0x4004 5040	IKYR	Flash特殊信息存储区解锁控制寄存器
0x4004 5100	FLASH_MEMRMP	内存映射选择寄存器

8.3.4.1 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	ICEN	PRFTEN	保留					LATENCY[2:0]		
-				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]**	**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）
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	**ICEN**	**指令缓冲区使能** 0：关闭指令缓冲区 1：启用指令缓冲区
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

8.3.4.2 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

8.3.4.3 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

8.3.4.4 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								**STAERRC**	**PGWERRC**	**PGPERRC**	**WRPRTERRC**	**FLSERRC**	**保留**	**OPERRC**	**EOPC**
-								*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
BSY	**保留**							**STAERR**	**PGWERR**	**PGPERR**	**WRPRTERR**	**FLSERR**	**保留**	**OPERR**	**EOP**
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 ~ 24	**保留**	**-**

23	**STAERRC**	**Flash状态错误标志清除位** 0：无效 1：清除
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 ~ 8	**保留**	**-**
7	**STAERR**	**Flash状态错误标志位** 在Flash未正常解锁的情况下，当试图对Flash擦除或编程，该位硬件置'1'。STAERRC@FLASH_SR写入'1'可以清除这位状态。 在STRT@FLASH_CR未置'1'的情况下，试图对Flash编程，该位硬件置'1'。STAERRC@FLASH_SR写入'1'可以清除这位状态。
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。

8.3.4.5 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） 其它：保留值
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'。 在一次不成功的解锁操作后，下次系统复位前，该位不能再被改变。
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(OTP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OTP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
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 …….. 0111110：扇区62 0111111：扇区63 1xxxxxx：保留 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2048字节；EEPROM区按扇区擦除时，每个扇区大小是2048字节。

8.3.4.6 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。

8.3.4.7 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。

8.3.4.8 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。

8.3.4.9 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。

8.3.4.10 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。

8.3.4.11 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特殊信息区中的OTP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。

8.3.4.12 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）

SH32F284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。模拟电路工作电压(AVDD)不能超过VDD±0.3V。

SH32F284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

- 内建电源预热功能，上电时约 16ms，唤醒时约 1ms
- 内建振荡器预热功能，约 16us

上电后，SH32F284会先经过电源预热，和振荡器预热，然后才开始运行程序。

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
- BOD可以唤醒停机模式

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 去抖动时间约 60μs
- 当供电电压低于设定电压时，将产生内部复位

9.4 低功耗管理

SH32F284可以通过以下几个方面降低系统功耗：

- 使用低电压供电；
- 降低系统时钟；
- 关闭 APB 和 AHB 总线上未被使用的外设时钟（默认是关闭的）；
- 降低 GPIO 和外设接口电路上电流；

SH32F284支持两种低功耗模式：

- 睡眠模式：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：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9-1　　低功耗模式一览

模式	进入	唤醒	唤醒延时
睡眠模式	WFI/WFE，保持SLEEPDEEP=0	任一中断/唤醒事件/复位信号	无
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； 3. 内核供电唤醒时间；

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。

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（LSI时钟除外），并使供电模块进入低功耗状态。因为IWDT时钟源是LSI，在停机模式下IWDT可以继续工作，LVR和BOD在停机模式下也可以正常工作（有独立开关，如果不需要可以关闭）。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（MCM和GPT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（MCM和GPT）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在停机模式下是开启的（前提是IWDT开启），这种情况下停机模式会被IWDT唤醒。可以通过用户代码选项"**OP_WDTPD**"设置为不开启，避免IWDT唤醒停机的情况。*

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。

9.5 寄存器

SYSCFG 模块寄存器列表 (基地址:0x0x4004 4000)

地址	寄存器名	说明
0x4004 4000	PWRCR	电源控制寄存器
0x4004 4004	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								BODEN	BODIE	BODMD[1:0]		VBOD[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 ~ 8	保留	-
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

SH32F284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_CSR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_CSR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

SH32F284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_CSR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_CSR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Cortex-M3内核的“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）。该寄存器中的另一个控制位VECTRESET则只复位Cortex-M3内核，不复位外设。

更多信息请参考ARM官方手册”Cortex-M3 Technical Reference Manual - Revision r2p1”。（汉译版本为“Cortex-M3技术参考手册”）

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。

10.1.2.1 上电复位

SH32F284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）是一种强制性保护复位，可以通过用户代码选项“OP_LVR”开启，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～16MHz）
- PLL时钟（最高84MHz）

其中PLL时钟可选择HSI和HSE作为时钟源。

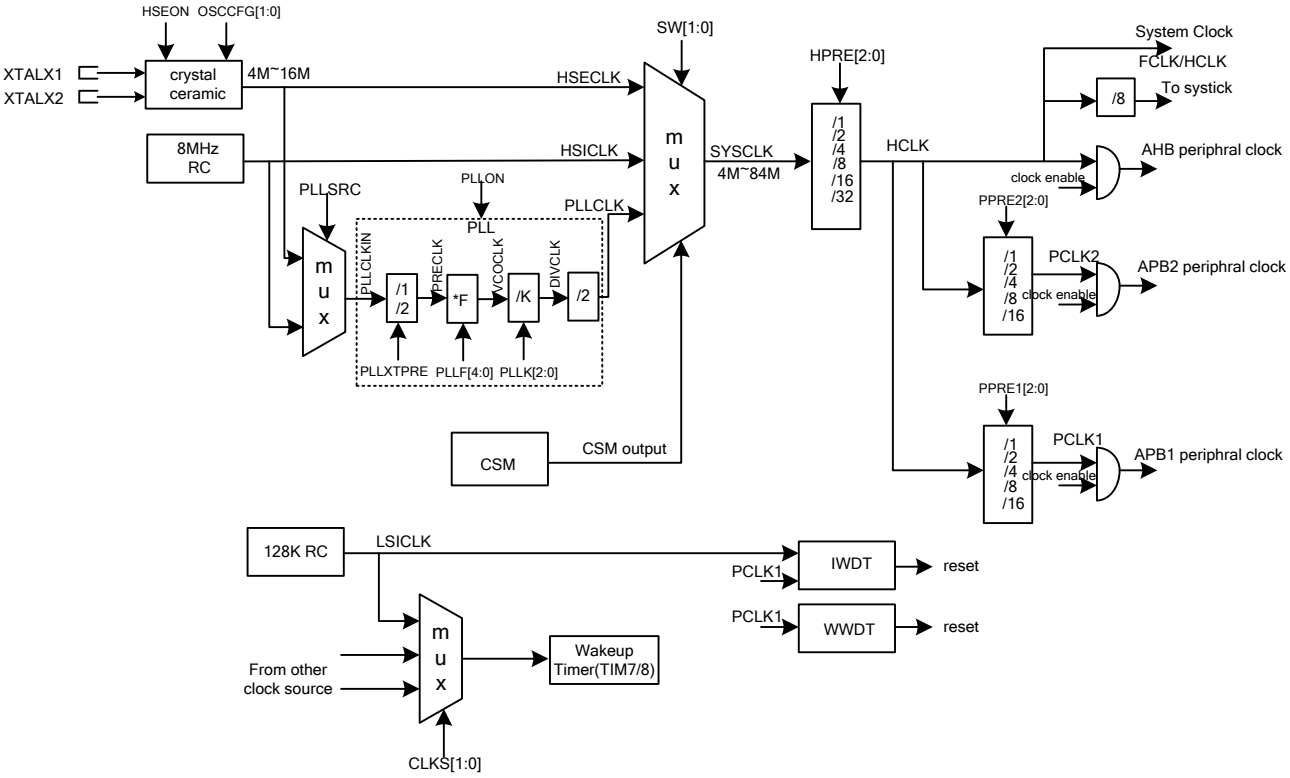


图10-1　　SH32F284 的时钟结构框图

用户可通过多个预分频器配置AHB、APB2和APB1的频率。

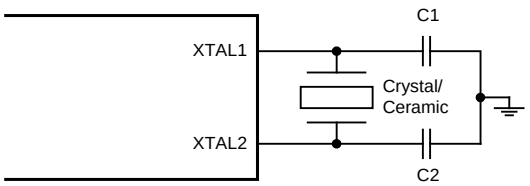
- HCLK：AHB总线时钟，最高频率84MHz，包括Cortex-M3内核、内存等都由HCLK驱动。
- PCLK2：APB2总线时钟，最高可设置为HCLK的二分之一，APB2总线上的外设都由PCLK2驱动。
- PCLK1：APB1总线时钟，最高可设置为HCLK的四分之一，APB1总线上的外设都由PCLK1驱动。

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 - 16MHz

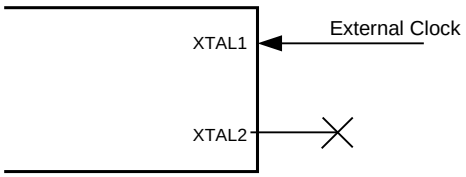


4~16Mz外部振荡器（包含晶体谐振器和陶瓷谐振器）可为系统提供更为精确的主时钟。在时钟控制寄存器HSERDY@RCC_CR位用来指示高速外部振荡器是否稳定。在启动时，直到这一位被硬件置'1'，时钟才被释放出来。如果在时钟中断寄存器RCC_CIENR中允许产生中断，将会产生相应中断。

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 – 16MHz



在这个模式里，必须提供外部时钟。它的频率最高可达16MHz。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，即启动预二分频。

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%(常温下)，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]位来进一步微调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：

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

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

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

N：由预分频因子PLLXTPRE决定，取1或2，对于HSI时钟源，一般预分频因子N取2。

F：由倍频因子PLLF指定，取值范围15～46；

K：由分频因子PLLK指定，取值范围1～8；

对PLL模块而言，PLLCLKIN输入时钟范围4MHz～16MHz，PRECLK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。最终PLLCLK时钟要求满足4MHz～84MHz范围。N、F、K参数的配置必须满足以上时钟范围要求（注）。

注：PRECLK、VCOCLK、PLLCLK都要求设置在允许范围内，超范围的PLL参数配置无法保证时钟系统稳定性。

以设置84MHz系统主频为例：

（1） 选择8MHz HSI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；

（2） 选择8MHz HSE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。

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时钟不是一个精确的时钟源，频率范围在40KHz～160KHz。

LSI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 ready或PLL ready)，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。

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，SH32F284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（频率低于正常值或高于正常值），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；

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，该中断连接至NMI不可屏蔽中断（由IEN_CSM@SYSCFG.SAFR控制位使能）。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上电是常开的，但可以通过用户代码选项“OP_CSM”关闭。CSM开启时只有在HSE ready或PLL lock后监控才有效。

发生CSM后，除了HSECSMF和PLLCSMF被置位外，其他RCC寄存器控制和状态位如HSEON[注]、HSERDY、HSERDYIE、HSERDYIF、PLLON、PLLRDY、PLLRDYIE、PLLRDYIF都可能会发生变化，需要用户程序重新初始化RCC模块。

注：PLLSRC=1时无法关闭HSEON，因此关闭HSEON要求先把PLLSRC复位。一旦CSM中断产生，引起NMI被不断执行，直到CSM中断挂起位被清除。因此，在NMI的处理程序中必须通过设置时钟中断寄存器RCC@CICLR中的CSMC位来清除CSM中断（将会同时清除2个异常标志）。

下面代码演示了时钟失效与恢复的过程。

CSM可以配置为产生NMI中断，中断服务程序执行时钟恢复判断和切换操作。

```
Void NMIException(void)
{
    if (产生了CSM中断)
    {
            （1）系统时钟已经自动切换到HSI；
            （2）RCC相关控制和状态位已经自动复位；
            （3）用户可以在此添加应用相关的保护代码；
            /* 下面为HSE/PLL恢复做准备，一旦恢复将查询到ready标志（也可用中断方式）*/
            （4）用户手动打开HSE和PLL；
            （5）程序等待HSE ready；
            （6）如果超时，说明HSE尚未恢复，可以继续等待，程序仍然运行在HSI时钟下；
            （7）HSE ready，继续等待PLL ready；
            （8）如果超时，说明PLL尚未恢复，可以继续等待，程序仍然运行在HSI时钟下；
            （9）PLL ready，系统时钟切换到PLL运行；
            （10）清除CSM中断挂起位；
    }
}
```

10.2.7 系统节拍定时器（SysTick）

SysTick是一个24位倒计数定时器，在Cortex-M3内核中实现。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Cortex-M3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，也可使用外部参考时钟，其节拍周期可由“重装载值寄存器”设定。具体寄存器定义请参考ARM官方手册”Cortex-M3 Technical Reference Manual - Revision r2p1”。（汉译版本为“Cortex-M3技术参考手册”）

SH32F284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150000，在HCLK为80MHz频率下，产生15ms的时间基准。

10.2.8 HSI 微调功能（HSITRIM）

SH32F284出厂时HSI已校准到一个很高的精度（常温0.2%），但在全温度范围，HSI精度会有所下降（不同的温度范围精度不同，具体参考“电气特性”），此时用户可以再使用3bit控制位HSITRIM[2:0]对HSI进行微调，以适当提升精度。

SH32F284附加了一个小功能，假如系统刚好又有一个外部时钟基准[注]，可以使用该外部时钟基准去在线校准HSI时钟，方法是使用HSI去计量4096分频后的外部时钟基准，结果存放在TRIMREF[12:0]@RCC_HSICAL中，由用户根据该结果再去微调HSITRIM[2:0]，从而达到补偿温度变化的目的。该功能由TRIMRUN@RCC_HSICAL开启。

计算公式为：HSI当前频率 = (TRIMREF[12:0]+1)/4096 * 基准时钟频率

比如外部时钟基准是8.000MHz，如果得到TRIMREF[12:0]=4086，表示HSI当前频率为7.982MHz，那么要校准到8.000MHz，需要+0.22%，查表最接近的档位为+0.25%。

如果系统时钟使用HSI+PLL，做HSI微调之后需要重新打开PLL。

10.3 寄存器

RCC 模块寄存器列表 (基地址:0x0x4004 4400)

地址	寄存器名	说明
0x4004 4400	CR	时钟控制寄存器
0x4004 4404	CFGR	时钟配置寄存器
0x4004 4408	CIENR	时钟中断使能寄存器
0x4004 440C	CISTR	时钟中断状态寄存器
0x4004 4410	CICLR	时钟中断标志清除寄存器
0x4004 4414	AHBRSTR	AHB外设复位寄存器
0x4004 4418	APB2RSTR	APB2外设复位寄存器
0x4004 441C	APB1RSTR	APB1外设复位寄存器
0x4004 4420	AHBENR	AHB外设时钟使能寄存器
0x4004 4424	APB2ENR	APB2外设时钟使能寄存器
0x4004 4428	APB1ENR	APB1外设时钟使能寄存器
0x4004 442C	RSTSTR	RESET状态寄存器
0x4004 4430	RSTCLR	清除RESET状态寄存器
0x4004 4434	HSICAL	HSI振荡器校准寄存器
0x4004 4438	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								PLLRDY	PLLON	HSERDY	HSEON	SWS[1:0]		SW[1:0]	
-								*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 ~ 8	保留	-
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4-16MHz时钟周期清零 0：外部4-16MHz时钟没有就绪 1：外部4-16MHz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												PLLXTPRE	PLLSRC	PLLF[5:0]	
-												*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]			PPRE2[2:0]			PPRE1[2:0]			HPRE[2:0]		
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	**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	保留	-

16 ~ 12	**PLLF[4:0]**	**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。 00000：F=15 00001：F=16 00010：F=17 ... ... 11101：F=44 11110：F=45 11111：F=46 注意：F、K的设置必须满足PLLCLK输出范围要求。SH32F284要求输出范围4M～84M，具体见“PLL”章节。
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 注意：F、K的设置必须满足PLLCLK输出范围要求。SH32F284要求输出范围4M～84M，具体见“PLL”章节。
8 ~ 6	**PPRE2[2:0]**	**高速APB预分频(APB2) (APB high-speed prescaler (APB2))** 由软件置’1’或清’0’来控制高速APB2时钟(PCLK2)的预分频系数。 0xx：HCLK不分频 100：HCLK 2分频 101：HCLK 4分频 110：HCLK 8分频 111：HCLK 16分频 警告：软件必须保证APB2时钟频率不超过60MHz。
5 ~ 3	**PPRE1[2:0]**	**低速APB预分频(APB1) (APB low-speed prescaler (APB1))** 由软件置’1’或清’0’来控制低速APB1时钟(PCLK1)的预分频系数。 0xx：HCLK不分频 100：HCLK 2分频 101：HCLK 4分频 110：HCLK 8分频 111：HCLK 16分频 警告：软件必须保证APB1时钟频率不超过30MHz。
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4-16MHz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4-16MHz振荡器产生的时钟就绪中断； 1：外部4-16MHz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	GPTRST	保留	GPT2RST	GPT1RST	GPT0RST	CRCRST	MACPRST	DMARST	SYSCFGRST	ADC3RST	保留	ADC1RST	保留	IODATRST	IOCFGRST
-	*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 ~ 15	保留	-
14	**GPTRST**	**GPT模块外围控制复位** 由软件置'1'，硬件清'0' 0：无影响 1：GPT复位
13	保留	-
12	**GPT2RST**	**GPT2复位 (Gernal PWM Timer 2 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：GPT2复位
11	**GPT1RST**	**GPT1复位 (Gernal PWM Timer 1 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：GPT1复位
10	**GPT0RST**	**GPT0复位 (Gernal PWM Timer 0 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：GPT0复位
9	**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
8	**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
7	**DMARST**	**DMA模块复位 (DMA reset)** 由软件置'1'，硬件清'0' 0：无影响 1：DMA模块复位

6	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。
5	ADC3RST	**ADC3复位 (ADC3 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：ADC3复位
4	保留	-
3	ADC1RST	**ADC1复位 (ADC1 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：ADC1复位
2	保留	-
1	IODATRST	**通用I/O端口数据区复位 (GPIO data block reset)** 由软件置'1'，硬件清'0' 0：无影响 1：通用I/O端口数据区复位
0	IOCFGRST	**通用I/O端口配置区复位 (GPIO configuration 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 APB2 外设复位寄存器 (RCC_APB2RSTR)

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
保留															MCM1RST
-															*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	MCM1RST	**MCM1模块复位 (MCM1 module reset)** 由软件置'1'，硬件清'0' 0：无影响 1：MCM1模块复位

10.3.8 APB1 外设复位寄存器 (RCC_APB1RSTR)

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		AMOCRST	WWDTRST	TWI1RST	保留	保留	SPI1RST	UART3RST	UART2RST	UART1RST	QEIRST	TIM8RST	TIM7RST	TIM6RST	TIM5RST
-		*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 ~ 14	保留	-
13	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
12	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
11	TWI1RST	**TWI1复位 (TWI 1 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：TWI1复位
10	保留	-
9	保留	-
8	SPI1RST	**SPI1复位 (SPI 1 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：SPI1复位
7	UART3RST	**UART3复位 (UART3 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：UART3复位
6	UART2RST	**UART2复位 (UART 2 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：UART2复位
5	UART1RST	**UART1复位 (UART 1 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：UART1复位
4	QEIRST	**QEI复位 (QEI reset)** 由软件置'1'，硬件清'0' 0：无影响

		1：QEI复位
3	**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
2	**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7复位
1	**TIM6RST**	**定时器6复位 (Timer 6 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：定时器6复位
0	**TIM5RST**	**定时器5复位 (Timer 5 reset)** 由软件置'1'，硬件清'0' 0：无影响 1：定时器5复位

10.3.9 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	GPTEN	保留	GPT2EN	GPT1EN	GPT0EN	CRCEN	MACPEN	DMAEN	SYSCFGEN	ADC3EN	保留	ADC1EN	保留	IODATEN	IOCFGEN
-	*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 ~ 15	保留	-
14	**GPTEN**	**GPT模块外围控制时钟使能** 由软件置'1'或清'0' 0：GPT时钟关闭 1：GPT时钟开启 注：要使用GPTx模块，需先开启GPT时钟
13	保留	-
12	**GPT2EN**	**GPT2时钟使能 (GPT 2 clock enable)** 由软件置'1'或清'0' 0：GPT2时钟关闭 1：GPT2时钟开启
11	**GPT1EN**	**GPT1时钟使能 (GPT 1 clock enable)** 由软件置'1'或清'0' 0：GPT1时钟关闭

		1：GPT1时钟开启
10	**GPT0EN**	**GPT0时钟使能 (GPT 0 clock enable)** 由软件置'1'或清'0' 0：GPT0时钟关闭 1：GPT0时钟开启
9	**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
8	**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
7	**DMAEN**	**DMA模块时钟使能 (DMA clock enable)** 由软件置'1'或清'0' 0：DMA模块时钟关闭 1：DMA模块时钟开启
6	**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。
5	**ADC3EN**	**ADC3时钟使能 (ADC3 clock enable)** 由软件置'1'或清'0' 0：ADC3时钟关闭 1：ADC3时钟开启
4	**保留**	**-**
3	**ADC1EN**	**ADC1时钟使能 (ADC1 clock enable)** 由软件置'1'或清'0' 0：ADC1时钟关闭 1：ADC1时钟开启
2	**保留**	**-**
1	**IODATEN**	**通用I/O端口数据区时钟使能 (GPIO data block clock enable)** 由软件置'1'或清'0' 0：通用I/O端口数据区时钟关闭 1：通用I/O端口数据区时钟开启
0	**IOCFGEN**	**通用I/O端口配置区时钟使能 (GPIO configurature block clock enable)** 由软件置'1'或清'0' 0：通用I/O端口配置区时钟关闭 1：通用I/O端口配置区时钟开启

10.3.10 APB2 外设时钟使能寄存器 (RCC_APB2ENR)

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															MCM1EN
-															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	MCM1EN	**MCM1模块时钟使能 (MCM1 module clock enable)** 由软件置'1'或清'0' 0：MCM1模块时钟关闭 1：MCM1模块时钟开启

10.3.11 APB1 外设时钟使能寄存器 (RCC_APB1ENR)

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															
-															
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		AMOCEN	WWDTEN	TWI1EN	保留	保留	SPI1EN	UART3EN	UART2EN	UART1EN	QEIEN	TIM8EN	TIM7EN	TIM6EN	TIM5EN
-		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 ~ 14	保留	-
13	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
12	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
11	TWI1EN	**TWI1时钟使能 (TWI1 clock enable)** 由软件置'1'或清'0' 0：TWI1时钟关闭 1：TWI1时钟开启
10	保留	-
9	保留	-
8	SPI1EN	**SPI1时钟使能 (SPI1 clock enable)** 由软件置'1'或清'0' 0：SPI1时钟关闭

		1：SPI1时钟开启
7	**UART3EN**	**UART3时钟使能 (UART3clock enable)** 由软件置'1'或清'0' 0：UART3时钟关闭 1：UART3时钟开启
6	**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
5	**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
4	**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
3	**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
2	**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
1	**TIM6EN**	**定时器6时钟使能 (Timer 6 clock enable)** 由软件置'1'或清'0' 0：定时器6时钟关闭 1：定时器6时钟开启
0	**TIM5EN**	**定时器5时钟使能 (Timer 5 clock enable)** 由软件置'1'或清'0' 0：定时器5时钟关闭 1：定时器5时钟开启

10.3.12 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									保留	WWDTRSTF	IWDTRSTF	SWRSTF	PORSTF	LVRSTF	PINRSTF
-									*RO*	*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 ~ 7	保留	-
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

10.3.13 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									保留	WWDTRSTFC	IWDTRSTFC	SWRSTFC	PORSTFC	LVRSTFC	PINRSTFC
-									WO	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 ~ 7	保留	-
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

10.3.14 HSI 振荡器校准寄存器 (RCC_HSICAL)

偏移地址: 0x0034

复位值: 0xXXXX XX00

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	0	0	0	0	0	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%

10.3.15 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

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_CSR标志。

独立看门狗工作时钟是LSI，默认溢出时间约1秒，由于LSI的范围较宽，用户程序需确保足够的裕量。

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时为最迟喂狗时刻，可申请WWDT中断。

SH32F284中WWDT的窗口下边界固定为0。在计数器到达0xFF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PCLK1。

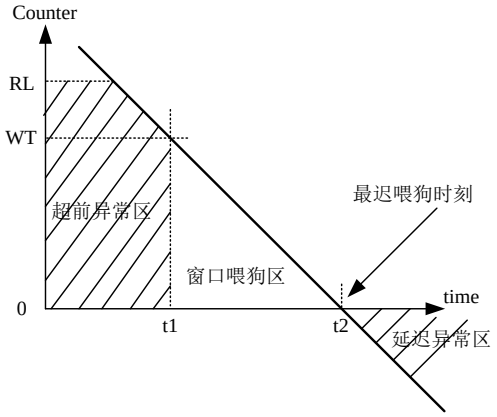


图11-1 窗口看门狗示意图

应用WWDT时，先设置WWDT的窗口重载值WWDTRLR，窗口值WWDTWTR，然后开启WWDT，计数器开始运行。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。

当微控制器进入调试模式时(Cortex-M3核心停止)，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 寄存器

IWDT 模块寄存器列表 (基地址:0x0x4000 4C00)

地址	寄存器名	说明
0x4000 4C00	CR	独立看门狗控制寄存器
0x4000 4C04	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
IWDTON	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。

WWDT 模块寄存器列表 (基地址:0x0x4000 5000)

地址	寄存器名	说明
0x4000 5000	CR	窗口看门狗控制寄存器
0x4000 5004	SR	窗口看门狗状态寄存器
0x4000 5008	CLR	窗口看门狗喂狗寄存器
0x4000 500C	WTR	窗口看门狗窗口寄存器

11.3.3 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=1×256（默认） 001: 预分频因子=2×256 010: 预分频因子=4×256 011: 预分频因子=8×256 100: 预分频因子=16×256 101: 预分频因子=32×256 110: 预分频因子=64×256 111: 预分频因子=128×256
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。

11.3.4 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。

11.3.5 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。

11.3.6 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，CRC_INIT的低16位有效，当计算CRC-8时，CRC_INIT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。
- 支持DMA (CRC_DR可作为DMA的目标地址)
- 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

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。

12.1.3.1 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			**读取**		
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			**读取**		
	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			**读取**		
	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’
	8位写入			**读取**		
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			**读取**		
	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			**读取**		
	COMPLW	RBITW	RBYTEW	COMPLR	RBITR	RBYTER
CRC_8	B0 = ~B0	b[0..7]=b[7..0]	B0=B3’	B0 = ~B0	b[0..7]=b[7..0]	忽略

	B1 = ~B1 B2 = ~B2 B3 = ~B3	b[8..15]=b[15..8] b[16..23]=b[23..16] b[24..31]=b[31..24]	B3=B0' B1=B2' B2=B1'			
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'

12.1.3.2 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:0x0x4004 5800)

地址	寄存器名	说明
0x4004 5800	**DR**	**CRC数据寄存器**
0x4004 5804	**CR**	**CRC控制寄存器**
0x4004 5808	**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]**	**1. 写入数据并触发CRC计算** **2. 读取时返回上一次CRC计算的结果** **3. 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) 01: CRC-16 (生成多项式为0x8005) 10: CRC-CITT (生成多项式为0x1021) 11: CRC-8 (生成多项式为0x7)
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。 MODE=0: 32位有效 MODE=1: 低16位有效 MODE=2: 低16位有效 MODE=3: 低8位有效

12.2 SRAM 检测模块（RAMBIST）

12.2.1 SRAM 检测模块简介

为满足IEC60730对RAM区检测的要求，SH32F284提供了硬件实现的RAM检测模块。此模块支持March-C及March-X算法。并支持检测与用户程序并行执行，与传统RAM检测的软件实现相比，极大提高了检测效率及降低了软件的复杂度

SRAM检测模块功能描述

1. 支持March-C算法和March-X算法。可以检测到SAF,TF,CFin,CFid,BF,SCF等错误
2. 支持自动数据备份及恢复，需要待测试RAM的高地址区域作为RAM的数据备份区。在检测过程中，此区域用于保存需要检测RAM(CRAM或SRAM)的数据，等检测完成后再恢复到原RAM。在检测过程中不允许用户程序访问此备份区域。数据备份区的大小由BLKSZ@RAMBIST_CFG 寄存器决定。例如BLKSZ为1代表检测大小为32Bytes,所以备份区的大小也为32Bytes.
3. 支持检测Code RAM 及 Data RAM
4. 支持数据备份区的自检功能。
5. 支持此检测运行过程中用户程序访问除数据备份区域之外的RAM。用户程序的访问优先于模块的检测，检测中无需中断程序的正常执行。
6. 检测过程中发现错误后会立即置ERR@RAMBIST_CSR位，然后恢复备份数据并结束检测.
7. 检测过程中，不允许程序修改此模块的寄存器但允许程序读取这些寄存器。
8. 无论是否在进行RAM检测, 都不允许对写保护的CRAM进行写操作。

12.2.1.1 March-C 算法描述

Step1: 全写 0。

Step2: 从头到尾按Bit读0判断,然后写1

Step3: 从头到尾按Bit读1判断,然后写0

Step4: 从尾到头按Bit读0判断,然后写1

Step5: 从尾到头按Bit读1判断,然后写0

Step6: 读取后判断全0。

12.2.1.2 March-X 算法描述

March-X是March-C算法的子集。

Step1: 全写 0。

Step2: 从头到尾按Bit读0判断，然后写1

Step3: 从尾到头按Bit读1判断，然后写0

Step4: 读取后判断全0。

12.2.1.3 SRAM 检测模块操作步骤

软件操作步骤：

1. 备份区域自检
   a) 设置RAMTYP@RAMBIST_ADDR设定备份区域位置(C-RAM 或 D-RAM)
   b) 设置BLKSZ@RAMBIST_CFG设定备份区域大小
   c) 设置SEL@ RAMBIST_CFG选择所用的算法(March-C 或 March-X)
   d) 清ERR@ RAMBIST_CSR。清除错误标志。
   e) 设置MOD@ RAMBIST_CSR,设定自检模式。
   f) 往RUN@ RAMBIST_CSR写入0x59A6,启动模块运行。
   g) 检测BYS@ RAMBIST_CSR,判断是否检测结束
   h) 检测ERR@ RAMBIST_CSR,判断是否出错。
2. 普通RAM区域检测
   a) 设置RAMTYP@RAMBIST_ADDR设定检测位置(C-RAM 或 D-RAM)
   b) 设置BLKSZ@RAMBIST_CFG设定检测区域大小
   c) 设置SEL@ RAMBIST_CFG选择所用的算法(March-C 或 March-X)
   d) 清MOD@ RAMBIST_CSR,设定普通检测模式
   e) 设置ADDR@ RAMBIST_ADDR设定检测地址位置

f) 清ERR@ RAMBIST_CSR。清除错误标志
g) 往RUN@ RAMBIST_CSR写入0x59A6,启动模块运行
h) 检测BYS@RAMBIST_CSR,判断是否检测结束
i) 检测BYS@RAMBIST_CSR,判断是否检测结束
j) 循环 e ~ i 直到RAM的所有区块(备份区除外)检查完毕

12.2.2 RAMBIST 寄存器

RAMBIST 模块寄存器列表 (基地址:0x0x4004 5C00)

地址	寄存器名	说明
0x4004 5C00	ADDR	SRAM自检地址寄存器
0x4004 5C04	CFG	SRAM自检配置寄存器
0x4004 5C08	CSR	SRAM自检控制寄存器

12.2.2.1 SRAM 自检地址寄存器 (RAMBIST_ADDR)

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		RAMTYP[1:0]		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		RAMADR[13:0]													
-		RW													
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 30	保留	-
29 ~ 28	RAMTYP[1:0]	**选择之间区域CRAM及DRAM.** 00:无效 01:选择CRAM区 10:选择DRAM区 11:无效
27 ~ 14	保留	-
13 ~ 0	RAMADR[13:0]	**设置要检查RAM区域的地址[A13:A0].** 待检测区域的地址必须根据BLKSZ的大小对齐，否则会由硬件自动按BLKSZ的大小舍弃低位地址。 例: 检测区域的BLKSZ为 1 (表示32字节对齐),所以合法的RAMADDR为 00 RAMTYP[1:0] 0000 00xx xxxx xxx0 0000 如果输入00 RAMTYP[1:0] 0000 00xx xxxx xxx0 0101,在实际检查时也是从00 RAMTYP[1:0] 0000 00xx xxxx xxx0 0000

12.2.2.2 SRAM 自检配置寄存器 (RAMBIST_CFG)

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
SEL	保留												BLKSZ[2:0]		
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	SEL	选择检测算法 0：March-C 1：March-X
14 ~ 3	保留	-
2 ~ 0	BLKSZ[2:0]	测试区域大小 0: 16Bytes 1: 32Bytes 2: 64Bytes 3: 128Bytes 4: 256Bytes 5: 512Bytes 6: 1024Bytes 7: 2048Bytes

12.2.2.3 SRAM 自检控制寄存器 (RAMBIST_CSR)

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													ERR	BSY	MOD
-													RW	RO	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
RUN[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 ~ 19	保留	-
18	ERR	模块出错标志，软件清0.与RUN位同时进行. 0：没有检测到错误 1：检测到有错误 (包括:数据备份校验错误,RAM检测失败, 数据恢复校验错误, RAMTYP无效)
17	BSY	模块运行标志位，由模块自动置位及清除 0：表示没有检测或检测已完成 1：表示正在检测
16	MOD	选择检测模式 0: 检测由寄存器RAMADDR及SZBK@SYSCFG_RAMCFG指定的RAM区域 1: 检测数据备份区域. 数据备份区域大小BLKSZ@SYSCFG_RAMCFG寄存器决定,位置在SRAM的高位地址. 在此模式下RAMADDR的设置被忽略.
15 ~ 0	RUN[15:0]	启动检测，启动后BSY Flag自动置'1' 写0x59A6：启动检测，读取永远为0 其他：不启动检测

13. 通用功能 I/O (GPIO)

13.1 简介

SH32F284提供最多58个通用I/O口，分成5组端口（A～E），每组最多包含16个输入输出引脚。每组I/O端口都有独立的配置寄存器和数据寄存器（注）。

注：由于封装的限制，每组的端口数量是不定的，具体参考"引脚定义"，此处对具体引脚不再单列，统一按一组进行说明。

除PE组个别I/O口外，SH32F284的每一个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，或者与调试接口复用。

所有I/O都支持上拉和下拉、推挽和开漏输出，I/O口输入和输出驱动能力多档可调，个别I/O口提供120mA的灌电流能力，以满足某些应用中直驱LED或小功率器件的要求。

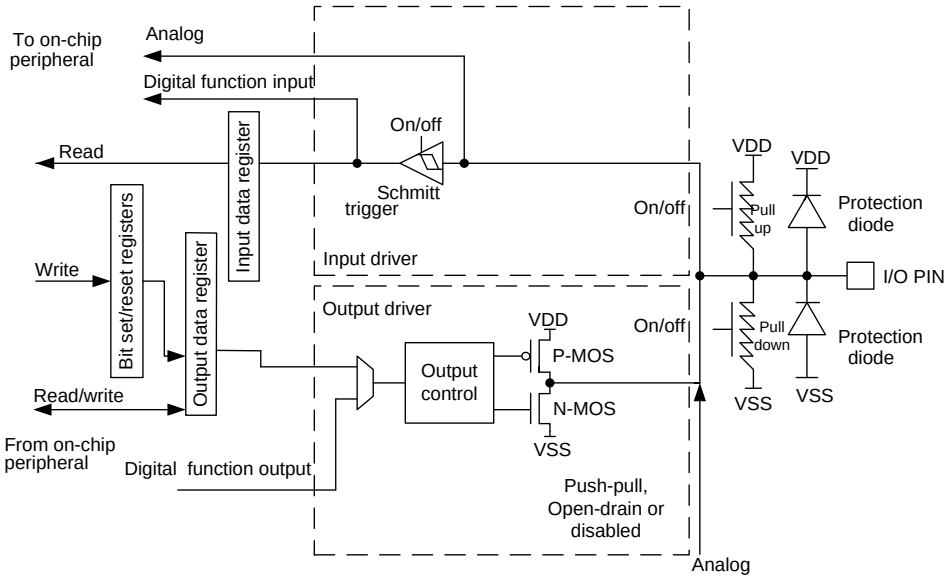


图13-1　　I/O 端口基本结构框图

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/下拉
- 输出管脚可配置为推挽或开漏，驱动能力多档可调
- 基于寄存器的按位置位/复位控制
- 端口寄存器支持按字节操作
- 所有端口都可用作外部中断，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
- I/O口复用为数字外设功能时允许读引脚输入状态
- I/O口支持复用功能重映射
- 支持端口配置锁定功能

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、ODRVR、PUPDR）完成的，用于配置I/O的输入输出模式（方向）、输出推挽/开漏方式、输出驱动档位、端口上下拉属性。

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	ODRVR[1:0]	I/O属性	
GPIO	AF0	0	0	x	GPIO输入	注1
			1	x	无意义	注2
		1	0	0/1/2/3	GPIO推挽输出	
			1		GPIO开漏输出	
数字外设	AF1～AF13	x	x	x	数字输入	注3
			0	0/1/2/3	数字推挽输出	
			1		数字开漏输出	
模拟外设	AF14	x	x	x	模拟输入输出	
全关	AF15	x	x	x	输入输出全关	注4

表中x表示不影响。

注1：启动完成后，默认是输入浮空（高阻）状态。

注2：此选项禁止使用。

注3：数字外设的输入输出是与外设绑定的，即启用某个数字外设功能后，用户仅需配置OTYPE、ODRVR和PUPDR寄存器，无需配置MODER寄存器。

注4：全关状态会完全切断输入输出通道。

除了上表中4种基本的端口功能外，还有一种系统接口（system port），包含调试接口SWJ和外部振荡器接口XTAL共7个端口。系统接口也可以复用为GPIO或数字外设。但在作为系统接口使用时上述配置寄存器无效，系统接口在SYSCFG中有进一步说明。

上下拉控制寄存器和端口功能是完全独立的，即除了GPIO输入外，GPIO输出、数字外设、模拟外设、全关时上下拉都可设置。

PUPDR[1:0]		端口功能	
0	0	GPIO/数字外设/模拟外设/全关	无上下拉
0	1		上拉
1	0		下拉
1	1		上下拉(注)

注：上拉和下拉可以同时打开，此时I/O口上可以获得近似VDD/2电平，，除非有必要，否则不建议这么做。

13.3.1 I/O 引脚的默认配置

在复位期间和复位之后，I/O口功能配置为“GPIO浮空输入”功能，除了SWJ接口。

系统复位后，SWJ作为默认功能被选中，为了保证正确进入调试模式，保持SWJ输入引脚非浮空是有必要的，为此复位后对应的SWJ引脚状态设置如下：

- JTRST：输入上拉
- JTDI：输入上拉
- JTMS/SWDIO：输入上拉
- JTCK/SWCLK：输入下拉（注）
- JTDO/SWO：输出状态

一旦JTAG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。

与SWJ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）。

13.3.2 I/O 控制寄存器

每个I/O口都有4个32位寄存器用于配置工作模式：MODER，OTYPER， ODRVR， PUPDR。

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(当输出0时，只有N-MOS被打开)；

ODRVR：设置输出驱动能力，分为4档：普通档、弱驱动档、极弱驱动档、超强驱动档（仅少量引脚有此能力）；

PUPDR：设置内部弱上拉和弱下拉，在输入和输出时都能设置上下拉寄存器相应位开启和关闭。

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，ODRVR，PUPDR，AFRL和AFRH。

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

所有端口都可以通过外部中断配置寄存器（EXTI_CFG1/ EXTI_CFG2，在EXTI模块中）来选择连接到外部中断输入线上。为了使用外部中断线，端口需要配置成输入模式。

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

SH32F284定义了8个TTL电平输入口，分别为PA2，PA3，PA8，PA11，PB0，PB1，PB4，PB5，由TTL接收开关TTLEN_x (x=A~E)@GPIO_CFG对应开启。

13.3.8 I/O 复用功能

寄存器AFRL和AFRH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16输入的I/O多路复用选择器用于选择需要的复用功能，即同一时刻每个端口只允许一个外设连接。

16个多路选择通道（AF0～AF15）分成4类：

- AF0：GPIO基本功能（默认）；
- AF1～AF13：数字外设复用功能；
- AF14：模拟外设复用功能；
- AF15：输入输出全关；

GPIO基本通道：AF0

这是I/O口上电和复位后的默认配置，所有I/O口上电都配置为GPIO输入浮空状态，除了JTAG接口。

JTAG接口引脚只有通过SWJCFG@SYSCFG_SAFR关闭SWJ-DP接口后才能作为AF0功能使用。

数字外设复用通道：AF1～AF13

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)、上下拉(PUPDR)、输出驱动(ODRVR)、通道选择(AFRL/AFRH)参数需用户软件配置。

请参考“第4节引脚定义”的“复用功能映射表”，以获取详细的引脚复用映射。

模拟外设复用通道：AF14

此通道用于片上模拟外设，输入输出方向、输出类型和输出驱动配置无意义，但上下拉是允许的。

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。

输入输出全关复用通道：AF15

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，但允许上下拉。

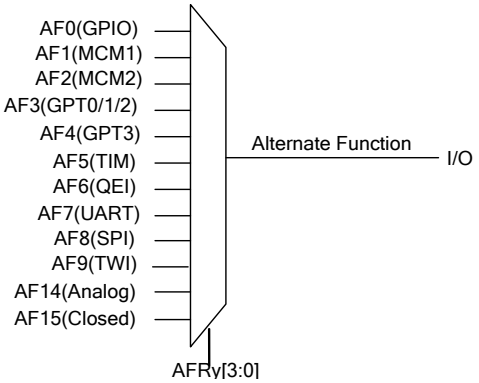


图13-2 I/O 复用功能多路选择框图

SH32F284支持引脚重映射，即同一外设可映射到不同I/O口上[注]，具体映射可对照“复用功能映射表”。

13.3.9 GPIO 输入功能

当I/O端口配置为GPIO输入时：

- 输入有效
- 输出无效（注）
- 根据上拉下拉配置的不同，上拉和下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

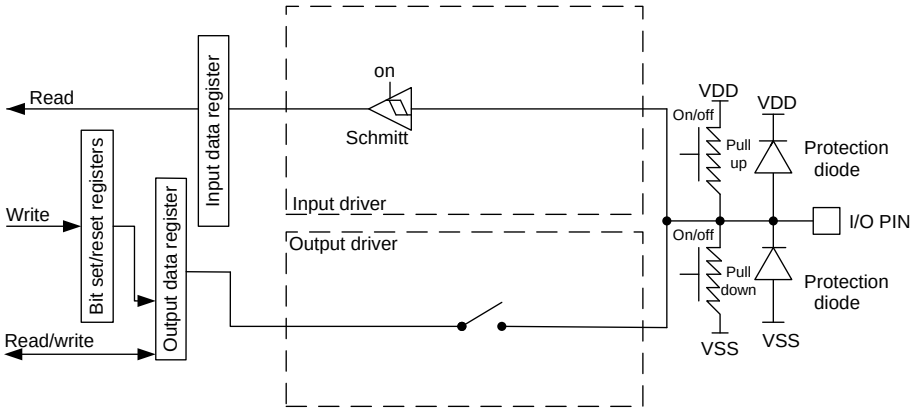


图13-3　　I/O 端口位的 GPIO 输入配置框图

13.3.10 GPIO 输出功能

当I/O端口被配置为GPIO输出时：

- 输出有效
  —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。
- 输入有效
- 根据上拉下拉配置的不同，上拉和下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状态注
- 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。

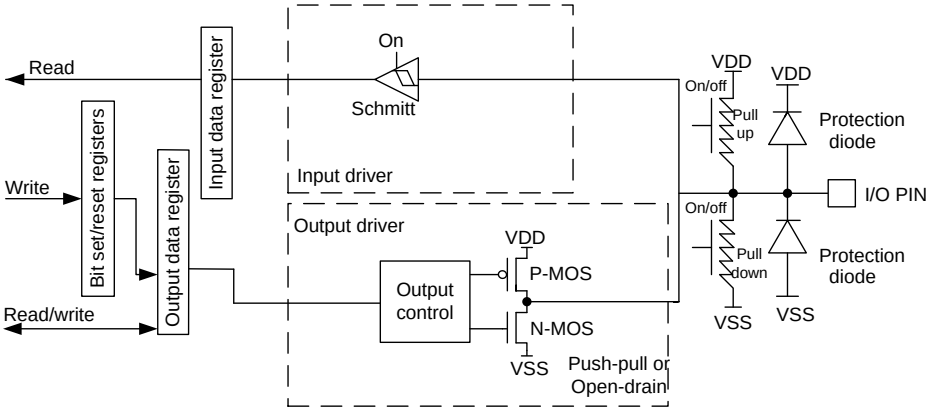


图13-4　　I/O 端口位的 GPIO 输出配置框图

13.3.11 数字复用功能

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缓冲区被关闭
- 作为数字复用输出时输出缓冲器被打开（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
- 根据上拉下拉配置的不同，上拉和下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：

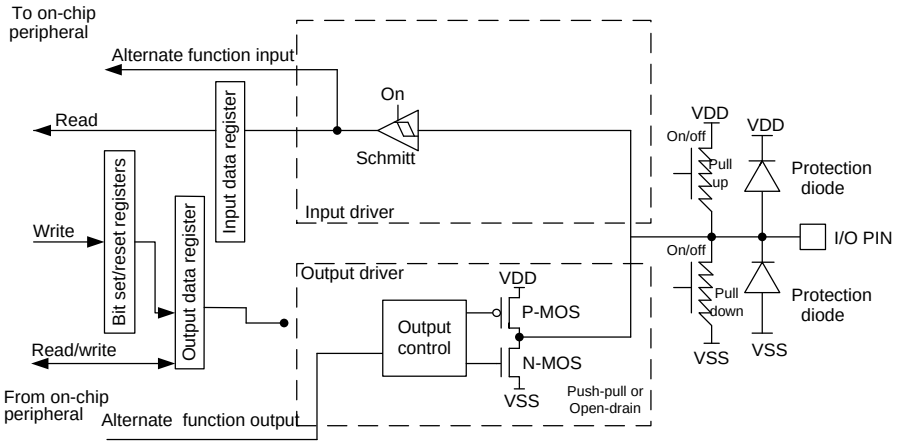


图13-5　　I/O 端口位的数字复用功能配置框图

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，此时输出缓冲器被关闭，输出通道呈现高阻状态。当然用户可通过配置上拉或下拉给出一个确定电平。

13.3.12 模拟复用功能

当I/O端口被配置为模拟复用配置时：

- 输出缓冲器被禁止；
- 自动关闭施密特触发输入，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'；
- 允许开启弱上拉和下拉；

下图示出了I/O端口位的模拟复用配置：

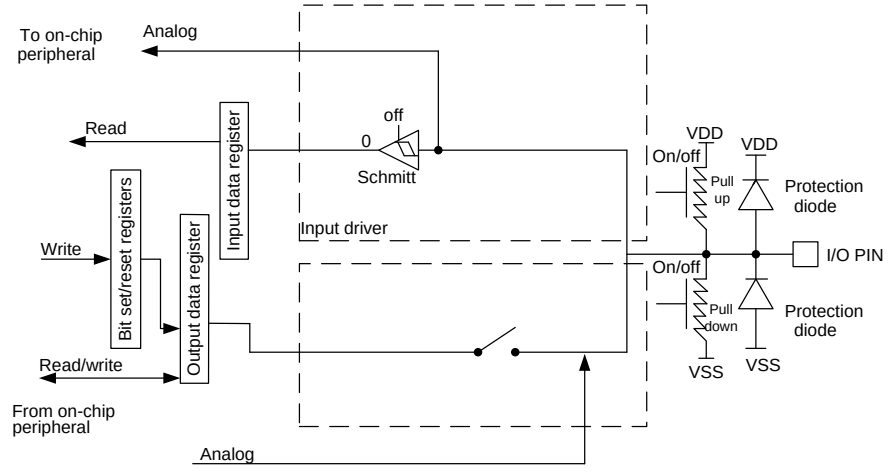


图13-6　　I/O 端口位的模拟复用功能配置框图

13.3.13 全关功能

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，除JTAG外的所有I/O口都处于全关状态，复位完成后进入默认状态（AF0输入悬空）。

13.4 寄存器

GPIO_CFG 模块寄存器列表 (基地址:0x0x4004 0000)

地址	寄存器名	说明
0x4004 0000 + 0x20*x	OTYPER	端口x输出方式配置寄存器 (端口A～E对应x:0~4)
0x4004 0004 + 0x20*x	ODRVR	端口x输出驱动能力配置寄存器 (端口A～E对应x:0~4)
0x4004 0008 + 0x20*x	PUPDR	端口x上下拉配置寄存器 (端口A～E对应x:0~4)
0x4004 000C + 0x20*x	AFRL	端口x复用功能寄存器低位 (端口A～E对应x:0~4)
0x4004 0010 + 0x20*x	AFRH	端口x复用功能寄存器高位 (端口A～E对应x:0~4)
0x4004 0014 + 0x20*x	TTLEN	端口x部分引脚TTL电平选择寄存器 (端口A～E对应x:0~4)
0x4004 0018 + 0x20*x	LCKR	端口x配置锁定寄存器 (端口A～E对应x:0~4)

13.4.1 端口 x 输出方式配置寄存器 (GPIOx_CFG_OTYPER)

偏移地址: 0x0000 + 0x20*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
OTy(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	**OTy(y=15~0)**	**端口x输出方式配置 (Port x output type configuration) 由软件配置** 0：output push-pull（reset state） 1：output open-drain

13.4.2 端口 x 输出驱动能力配置寄存器 (GPIOx_CFG_ODRVR)

偏移地址: 0x0004 + 0x20*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
ODRVR15[1:0]		ODRVR14[1:0]		ODRVR13[1:0]		ODRVR12[1:0]		ODRVR11[1:0]		ODRVR10[1:0]		ODRVR9[1:0]		ODRVR8[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
ODRVR7[1:0]		ODRVR6[1:0]		ODRVR5[1:0]		ODRVR4[1:0]		ODRVR3[1:0]		ODRVR2[1:0]		ODRVR1[1:0]		ODRVR0[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	**ODRVR15[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动

		11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
29 ~ 28	**ODRVR14[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
27 ~ 26	**ODRVR13[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
25 ~ 24	**ODRVR12[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
23 ~ 22	**ODRVR11[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
21 ~ 20	**ODRVR10[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
19 ~ 18	**ODRVR9[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
17 ~ 16	**ODRVR8[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
15 ~ 14	**ODRVR7[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动

		10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
13 ~ 12	**ODRVR6[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
11 ~ 10	**ODRVR5[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
9 ~ 8	**ODRVR4[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
7 ~ 6	**ODRVR3[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
5 ~ 4	**ODRVR2[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
3 ~ 2	**ODRVR1[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。
1 ~ 0	**ODRVR0[1:0]**	**端口x输出驱动能力配置 (Port x output driver configuration) 由软件配置** 00：普通状态（复位值） 01：弱驱动 10：极弱驱动 11：超强驱动 注：超强驱动仅部分引脚具备，具体见“电气特性”说明。如某个不具备此功能的引脚误配置了超强驱动，则呈现复位值配置。

13.4.3 端口 x 上下拉配置寄存器 (GPIOx_CFG_PUPDR)

偏移地址: 0x0008 + 0x20*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
PUPDR15[1:0]		PUPDR14[1:0]		PUPDR13[1:0]		PUPDR12[1:0]		PUPDR11[1:0]		PUPDR10[1:0]		PUPDR9[1:0]		PUPDR8[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
PUPDR7[1:0]		PUPDR6[1:0]		PUPDR5[1:0]		PUPDR4[1:0]		PUPDR3[1:0]		PUPDR2[1:0]		PUPDR1[1:0]		PUPDR0[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	**PUPDR15[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
29 ~ 28	**PUPDR14[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
27 ~ 26	**PUPDR13[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
25 ~ 24	**PUPDR12[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
23 ~ 22	**PUPDR11[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
21 ~ 20	**PUPDR10[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
19 ~ 18	**PUPDR9[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉

		注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
17 ~ 16	**PUPDR8[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
15 ~ 14	**PUPDR7[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
13 ~ 12	**PUPDR6[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
11 ~ 10	**PUPDR5[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
9 ~ 8	**PUPDR4[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
7 ~ 6	**PUPDR3[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
5 ~ 4	**PUPDR2[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
3 ~ 2	**PUPDR1[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。
1 ~ 0	**PUPDR0[1:0]**	**端口x上下拉配置 (Port x pull-up pull-down configuration) 由软件配置** 00：No pull-up，no pull-down（reset state） 01：Pull-up 10：Pull-down 11：同时上拉下拉 注：允许同时上下拉，个别端口可以获得理想的中间电平，具体见“电气特性”。

13.4.4 端口 x 复用功能寄存器低位 (GPIOx_CFG_AFRL)

偏移地址: 0x000C + 0x20*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
AFR7[3:0]				AFR6[3:0]				AFR5[3:0]				AFR4[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
AFR3[3:0]				AFR2[3:0]				AFR1[3:0]				AFR0[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	**AFR7[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR7功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
27 ~ 24	**AFR6[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR6功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
23 ~ 20	**AFR5[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR5功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5

		0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
19 ~ 16	**AFR4[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR4功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
15 ~ 12	**AFR3[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR3功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
11 ~ 8	**AFR2[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR2功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12

		1101: AF13 1110: AF14 1111: AF15
7 ~ 4	**AFR1[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR1功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
3 ~ 0	**AFR0[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR0功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15

13.4.5 端口 x 复用功能寄存器高位 (GPIOx_CFG_AFRH)

偏移地址: 0x0010 + 0x20*x

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
AFR15[3:0]				**AFR14[3:0]**				**AFR13[3:0]**				**AFR12[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
AFR11[3:0]				**AFR10[3:0]**				**AFR9[3:0]**				**AFR8[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	**AFR15[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR15功能选择： 0000: AF0 0001: AF1 0010: AF2

		0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
27 ~ 24	**AFR14[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR14功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
23 ~ 20	**AFR13[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR13功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
19 ~ 16	**AFR12[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR12功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9

		1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
15 ~ 12	**AFR11[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR11功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
11 ~ 8	**AFR10[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR10功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
7 ~ 4	**AFR9[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置** AFR9功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15
3 ~ 0	**AFR8[3:0]**	**端口x复用功能选择位 (Port x alternate function selection) 由软件配置**

		AFR8功能选择： 0000: AF0 0001: AF1 0010: AF2 0011: AF3 0100: AF4 0101: AF5 0110: AF6 0111: AF7 1000: AF8 1001: AF9 1010: AF10 1011: AF11 1100: AF12 1101: AF13 1110: AF14 1111: AF15

13.4.6 端口 x 部分引脚 TTL 电平选择寄存器 (GPIOx_CFG_TTLEN)

偏移地址: 0x0014 + 0x20*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				PA11	保留		PA8	保留				PA3	PA2	保留	
-				*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	PA11（注）	**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 ~ 9	保留	-
8	PA8	**PA8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。
7 ~ 4	保留	-
3	PA3	**PA3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。
2	PA2	**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 ~ 0	保留	-

注：TTL输入属性不是每个端口都具备，具体参考SYSCFG章节“TTL输入选择”，此处针对PA口进行说明。

13.4.7 端口 x 配置锁定寄存器 (GPIOx_CFG_LCKR)

偏移地址: 0x0018 + 0x20*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
LCKy(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	**LOCK[15:0]**	**LCK控制位操作解锁位 (Port x reset bit)** 这些位是只写，读取这些位返回0x0000。 0x5AA5：解锁 其他：锁定
15 ~ 0	**LCKy(y=15~0)**	**端口x锁定位(Port x lock bit)** 这些位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要） 0: 端口位配置未锁定 1: 端口位配置锁定

GPIO_IOD 模块寄存器列表 (基地址:0x0x4004 0400)

地址	寄存器名	说明
0x4004 0400 + 0x20*x	MODER	端口x模式寄存器 (x:0~4)
0x4004 0404 + 0x20*x	IDR	端口x输入数据寄存器 (x:0~4)
0x4004 0408 + 0x20*x	ODR	端口x输出数据寄存器 (x:0~4)
0x4004 040C + 0x20*x	BSRR	端口x位清除/置位寄存器 (x:0~4)

13.4.8 端口 x 模式寄存器 (GPIOx_IOD_MODER)

偏移地址: 0x0000 + 0x20*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
MODERy(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	**MODERy(y=15~0)**	**端口x模式配置 (Port x mode configuration) 由软件配置** 0：input mode（reset state） 1：output mode

13.4.9 端口 x 输入数据寄存器 (GPIOx_IOD_IDR)

偏移地址: 0x0004 + 0x20*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。

13.4.10 端口 x 输出数据寄存器 (GPIOx_IOD_ODR)

偏移地址: 0x0008 + 0x20*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。

13.4.11 端口 x 位清除/置位寄存器 (GPIOx_IOD_BSRR)

偏移地址: 0x000C + 0x20*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IDRy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ODRy位 注：如果同时设置同一个Port口的同一位BRy和BSy，则BSy优先级高。

14. 系统配置模块(SYSCFG)

此模块用于系统层面的相关配置，包括PWR配置、SAFR特殊复用功能定义、CRAM lock定义、DBG控制，该模块挂在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 PWR 配置

此处仅是BOD的配置和开关控制，包括PWRCR和PWRSR两个寄存器，具体参见“9.2掉电检测”章节。

14.1.2 SAFR 特殊复用功能定义

14.1.2.1 NMI 异常源开关

NMI为不可屏蔽中断，SH32F284提供了三种NMI异常源：时钟异常CSM、掉电检测BOD、外部中断（只取了EXTI 0），每个信号源都有独立的NMI使能控制。

NMI使能后不影响原异常源的中断通路，比如BOD和EXTI0本身的中断通路也是有效的，当然NMI的优先级更高。

14.1.2.2 SWJ 和 XTAL 引脚复用功能定义

SWJ接口包含四线JTAG接口和两线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[注]。

建议常用两线SWD接口，可额外释放3个I/O口。

注：如果SWJ接口全部配置为GPIO口，断电后如需再次连接调试器需在SWD接口模式下短按NRST引脚通过复位进入。

注意：SWJ接口在上电时进入调试模式，具有默认的上下拉属性，复用为GPIO输出时需注意上电时的电平影响。但复用为GPIO输入是没问题的。

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。

SWJ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、上下拉等配置。

14.1.3 CRAM 锁定选项

一旦CRAM设置为锁定，将变为只读，此时对CRAM进行写入将触发总线Fault。

CRAM锁定以2K为扇区，通过CRAMLCKy(y=0～7)@SYSCFG_CRAMLOCK开启。具体参见”CRAM”章节。

14.1.4 调试接口寄存器

DBGCR为调试接口寄存器，此处定义了调试模式下各主要外设模块的工作状态，以及进入低功耗模式的系统状态，提供了调试开关选项，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:0x0x4004 4000)

地址	寄存器名	说明
0x4004 4000	PWRCR	电源控制寄存器
0x4004 4004	PWRSR	电源状态寄存器
0x4004 4008	SAFR	系统配置寄存器
0x4004 400C	CRAMLOCK	CRAM扇区锁定控制寄存器
0x4004 4010	DBGCR	调试接口控制寄存器

14.2.1 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								BODEN	BODIE	BODMD[1:0]		VBOD[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 ~ 8	保留	-
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

14.2.2 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

14.2.3 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	SWJCFG[2:0]			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 ~ 2	SWJCFG[2:0]	**串行线JTAG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SWJ和跟踪复用功能的I/O口。SWJ(串行线JTAG)支持JTAG或SWD访问Cortex M3的调试端口。系统复位后的默认状态是启用SWJ但没有跟踪功能，这种状态下可以通过JTMS/JTCK脚上的特定信号选择JTAG或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，仅在SWD接口有效。 000：完全SWJ(JTAG + SWD)，SWJ调试口无法复用为普通I/O，JTAG，SWD调试接口均可使用。 001：完全SWJ(JTAG + SWD)，JTRST可复用为普通I/O；JTAG，SWD调试接口均可使用。

		010：关闭JTAG，只用SWD（包含SWO），JTRST，JTDI可复用为普通I/O；JTAG调试接口无法使用，只能使用SWD调试接口。 011：关闭JTAG，启用SWD（不包含SWO），JTRST，JTDI，JTDO可复用为普通I/O；JTAG调试接口无法使用，只能使用SWD调试接口。 1XX：关闭JTAG，关闭SWD；JTRST，JTDI，JTDO可复用为普通I/O，但JTMS（SWDIO），JTCK（SWCLK）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，此时JTAG调试接口无法使用，SWD调试接口仍可使用。 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JTAG接口无法使用，但可以通过SWD接口恢复，同时需配合reset信号。
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；

14.2.4 CRAM 扇区锁定控制寄存器 (SYSCFG_CRAMLOCK)

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								CRAMLCKy(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 ~ 16	**LOCK[15:0]**	**CRAMLCKy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 ~ 8	**保留**	**-**
7 ~ 0	**CRAMLCKy(y=7~0)**	**CRAM sector 7锁定位(CRAM sector 7 lock bit)** 这些位可读可写，写入这些位必须要满足LOCK解锁条件。（读取不需要） 0: CRAM sector y未锁定 1: CRAM sector y锁定

14.2.5 调试接口控制寄存器 (SYSCFG_DBGCR)

偏移地址: 0x0010

复位值: 0x0000 0080

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_GPT	DBG_WWDT	DBG_IWDT	DBG_DMA	保留					DBG_STOP	DBG_SLEEP
RW	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	-					*RW*	*RW*
0	0	0	0	0	0	0	0	1	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TIM5-TIM8模块停止工作** 0：TIM5-TIM8模块停止工作 1：TIM5-TIM8模块仍然正常工作
10	**DBG_GPT**	**当内核进入调试状态时GPT停止工作** 0：GPT停止工作 1：GPT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_DMA**	**当内核进入调试状态时DMA停止工作** 0：DMA停止工作 1：DMA仍然正常工作
6 ~ 2	**保留**	**-**
1	**DBG_STOP**	**调试停机模式。** 0：在停机模式时，系统时钟保持，由内部RC振荡器提供系统时钟(包括HCLK和FCLK)。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：在停机模式时，系统时钟关闭，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。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。
0	**DBG_SLEEP**	**调试睡眠模式** 0：在睡眠模式时，系统时钟保持，FCLK和HCLK时钟都由原先配置好的系统时钟提供。 1：在睡眠模式时，系统时钟关闭，FCLK由原先已配置好的系统时钟提供，HCLK则关闭。由于睡眠模式不会复位已配置好的时钟系统，因此从睡眠模式退出时，软件不需要重新配置时钟系统。

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Cortex-M3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 Cortex-M3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
- SH32F284支持 43 个可屏蔽中断(不包含16个Cortex-M3的中断)。
- 16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，CM3 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。CM3的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

CM3 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”Cortex-M3 Technical Reference Manual - Revision r2p1”。（汉译版本为“Cortex-M3 技术参考手册”）

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

SH32F284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可以选择PA0、PB0、PC0、PD0、PE0作为输入。 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个Cortex-M3内核标准异常源 + 43个硬件中断源

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	RESEET	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（本例中仅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)	MPU 不匹配，包括违反访问规范以及不匹配。是同步的。

				即使 MPU 被禁止或不存在，也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	-	0x0048	-	
3	-	0x004C	-	
4	RCC	0x0050	可调整(2)	PLL及外部时钟就绪中断
5	EXTI0	0x0054	可调整(2)	EXTI引脚0中断
6	EXTI1	0x0058	可调整(2)	EXTI引脚1中断
7	EXTI2	0x005C	可调整(2)	EXTI引脚2中断
8	EXTI3	0x0060	可调整(2)	EXTI引脚3中断
9	EXTI4	0x0064	可调整(2)	EXTI引脚4中断
10	DMA_CH0	0x0068	可调整(2)	DMA通道0中断
11	DMA_CH1	0x006C	可调整(2)	DMA通道1中断
12	DMA_CH2_7	0x0070	可调整(2)	DMA通道2~7中断
13	MCM1_FLT	0x0074	可调整(2)	MCM1模块PWM故障中断
14	MCM1_PWM	0x0078	可调整(2)	MCM1模块PWM中断
15	-	0x007C	-	-
16	-	0x0080	-	-
17	ADC1	0x0084	可调整(2)	ADC1中断
18	-	0x0088	-	-
19	ADC3	0x008C	可调整(2)	ADC3中断
20	-	0x0090	-	
21	-	0x0094	-	
22	GPT0_GTCIV	0x0098	可调整(2)	GPT0计数器上溢/下溢中断
23	GPT0_GTCIN	0x009C	可调整(2)	GPT0输入捕捉/比较匹配，死区时间错误，电平异常中断
24	GPT1_GTCIV	0x00A0	可调整(2)	GPT1计数器上溢/下溢中断
25	GPT1_GTCIN	0x00A4	可调整(2)	GPT1输入捕捉/比较匹配，死区时间错误，电平异常中断
26	GPT2_GTCIV	0x00A8	可调整(2)	GPT2计数器上溢/下溢中断
27	GPT2_GTCIN	0x00AC	可调整(2)	GPT2输入捕捉/比较匹配，死区时间错误，电平异常中断
28	-	0x00B0	-	-
29	-	0x00B4	-	-
30	GPT_POE	0x00B8	可调整(2)	GPT输入引脚中断
31	EXTI9_5	0x00BC	可调整(2)	EXTI引脚5~9中断
32	CMP1	0x00C0	可调整(2)	比较器1中断

33	CMP2	0x00C4	可调整(2)	比较器2中断
34	CMP3	0x00C8	可调整(2)	比较器3中断
35	QEI	0x00CC	可调整(2)	正交编码器接口(QEI)中断
36	TWI	0x00D0	可调整(2)	双线串行接口中断。
37	-	0x00D4	-	
38	SPI1	0x00D8	可调整(2)	SPI1 中断
39	-	0x00DC	-	-
40	UART1	0x00E0	可调整(2)	UART1中断
41	UART2	0x00E4	可调整(2)	UART2中断
42	UART3	0x00E8	可调整(2)	UART3中断
43	-	0x00EC	-	
44	EXTI15_10	0x00F0	可调整(2)	EXTI引脚15~10中断
45	-	0x00F4		
46	-	0x00F8		
47	-	0x00FC		
48	-	0x0100		
49	TIM5	0x0104	可调整(2)	定时器5中断
50	TIM6	0x0108	可调整(2)	定时器6中断
51	TIM7	0x10C	可调整(2)	定时器7中断
52	TIM8	0x0110	可调整(2)	定时器8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"Cortex-M3 Technical Reference Manual - Revision r2p1"。（汉译版本为"Cortex-M3技术参考手册"）

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 (33~64 个中断)
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	0x400249F0
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"Cortex-M3 Technical Reference Manual - Revision r2p1"。（汉译版本为"Cortex-M3技术参考手册"）

15.2 外部中断/事件控制器（EXTI）

外部中断/事件控制器由16个产生事件/中断请求的电平和边沿检测器组成，这16条线都可配置到GPIO口。每个输入线可以独立地配置输入类型(电平或边沿)和对应的触发事件(高低电平、上升沿下降沿或双沿)。每个输入线都可以独立地被使能。挂起寄存器保持着状态线的中断请求。

15.2.1 主要特性

- 每个中断/事件都有独立的使能控制
- 每个中断线都有专用的状态位
- 支持多达16个软件触发中断请求
- 支持作为DMA触发源
- 输入去抖动滤波参数可调

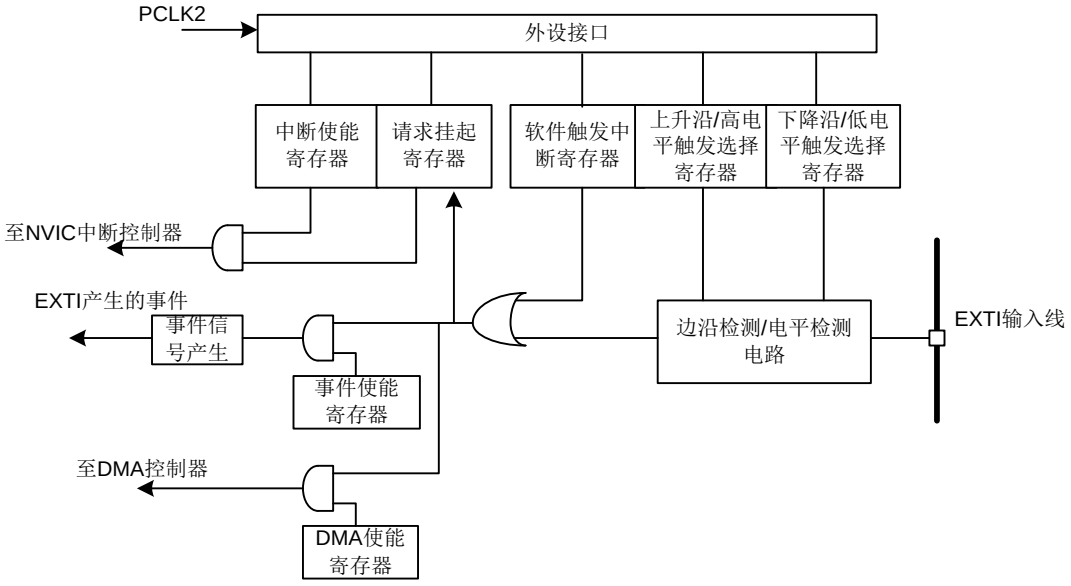


图15-1　　外部中断/事件控制器框图

15.2.2 唤醒事件管理

SH32F284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Cortex-M3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，因为对应事件线的中断挂起位（EXTI挂起寄存器中）也会被置位，软件需清除相应外设的挂起位。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。

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/电平检测设置2个触发寄存器，同时在中断使能寄存器的相应位写'1'允许中断请求。当外部中断线上发生了期待的边沿/电平时，将产生一个中断请求，对应的挂起位也随之被置'1'。在挂起寄存器的清除位写'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/电平检测设置2个触发寄存器，同时在事件使能寄存器的相应位写'1'允许事件请求。当事件线上发生了需要的边沿/电平时，将产生一个事件请求脉冲，同时中断通路对应的挂起位（PR）也会被置'1'。

SH32F284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，在正常运行时使用PCLK2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。在停机模式下由于PCLK2关闭，EXTI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，不建议再开启预分频处理。

EXTI支持作为DMA触发源，EXTI外部事件发生时，可选择送出DMA请求，触发DMA传输，DMA请求在DMA控制器响应并应答后释放。EXTI的DMA使能信号具体见“DMA使能寄存器”。

EXTI0能引起NMI中断，由IEN_EXTI0@SYSCFG_SAFR控制位使能。

硬件中断选择

通过下面的过程来配置16个输入线做为中断源：

- 配置16个中断线的使能位(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中断通道的使能和使能位，使得16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16个线路为事件源

- 配置16个事件线的使能位(EXTI_EMR)
- 配置事件线的触发选择位(EXTI_RTSR和EXTI_FTSR)

软件触发中断的选择

16个线路可以被配置成软件触发中断线。下面是产生软件中断的过程：

- 配置16个中断使能位(EXTI_IMR)
- 设置软件触发中断寄存器的请求位(EXTI_SWIER)

15.2.4 外部中断/事件线路映像

GPIO以下图的方式连接到16个外部中断/事件线上：

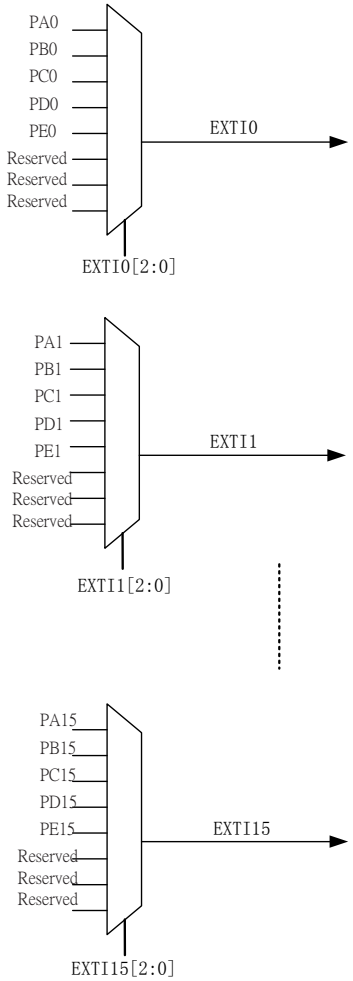


图15-2　　外部中断通用 I/O 映像

15.3 寄存器

EXTI 模块寄存器列表 (基地址:0x0x4002 1000)

地址	寄存器名	说明
0x4002 1000	IMR	中断使能寄存器
0x4002 1004	EMR	事件使能寄存器
0x4002 1008	TMSR	触发模式选择寄存器
0x4002 100C	RTSR	上升沿/高电平触发模式选择寄存器
0x4002 1010	FTSR	下降沿/低电平触发模式选择寄存器
0x4002 1014	SWIER	软件触发中断寄存器
0x4002 1018	PR	挂起寄存器
0x4002 101C	CFGL	外部中断配置低寄存器
0x4002 1020	CFGH	外部中断配置高寄存器
0x4002 1024	SAMPL	EXTI采样控制低寄存器
0x4002 1028	SAMPH	EXTI采样控制高寄存器
0x4002 102C	DMR	DMA使能寄存器

15.3.1 中断使能寄存器 (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
IMRy(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	IMRy(y=15~0)	线y上的中断使能(Interrupt Mask on line y) 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 事件使能寄存器 (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
EMRy(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	EMRy(y=15~0)	线y上的事件使能(Event Mask on line y) 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
TMRy(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	**TMRy(y=15~0)**	**线y上的触发模式选择位(Trigger Mode selection)** 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
RTRy(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	**RTRy(y=15~0)**	**线y上的上升沿/高电平触发事件配置位 (Rising/high level trigger event configuration bit of line y)** 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
FTRy(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	FTRy(y=15~0)	**线y上的下降沿/低电平触发事件配置位 (Falling/low level trigger event configuration bit of line y)** 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 软件触发中断寄存器 (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
SWIERy(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	保留	-
15 ~ 0	SWIERy(y=15~0)	**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EXTI_IMR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
PRCy(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
PRy(y=15~0)															
RO															
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 16	PRCy(y=15~0)	**挂起清除位** 0：无效 1：清除
15 ~ 0	PRy(y=15~0)	**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_CFGL)

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	EXTI7[2:0]			保留	EXTI6[2:0]			保留	EXTI5[2:0]			保留	EXTI4[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	EXTI3[2:0]			保留	EXTI2[2:0]			保留	EXTI1[2:0]			保留	EXTI0[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	EXTI7[2:0]	**EXTI 7引脚配置** 000: PA[7] 引脚 001: PB[7] 引脚 010: PC[7] 引脚 011: PD[7] 引脚 100: PE[7] 引脚 其他: 保留
27	保留	-
26 ~ 24	EXTI6[2:0]	**EXTI 6引脚配置** 000: PA[6] 引脚 001: PB[6] 引脚 010: PC[6] 引脚 011: PD[6] 引脚 100: PE[6] 引脚 其他: 保留
23	保留	-
22 ~ 20	EXTI5[2:0]	**EXTI 5引脚配置** 000: PA[5] 引脚 001: PB[5] 引脚 010: PC[5] 引脚 011: PD[5] 引脚 100: PE[5] 引脚 其他: 保留
19	保留	-
18 ~ 16	EXTI4[2:0]	**EXTI 4引脚配置** 000: PA[4] 引脚 001: PB[4] 引脚 010: PC[4] 引脚 011: PD[4] 引脚 100: PE[4] 引脚 其他: 保留
15	保留	-
14 ~ 12	EXTI3[2:0]	**EXTI 3引脚配置** 000: PA[3] 引脚 001: PB[3] 引脚 010: PC[3] 引脚 011: PD[3] 引脚 100: PE[3] 引脚

		其他: 保留
11	保留	-
10 ~ 8	EXTI2[2:0]	**EXTI 2引脚配置** 000: PA[2] 引脚 001: PB[2] 引脚 010: PC[2] 引脚 011: PD[2] 引脚 100: PE[2] 引脚 其他: 保留
7	保留	-
6 ~ 4	EXTI1[2:0]	**EXTI 1引脚配置** 000: PA[1] 引脚 001: PB[1] 引脚 010: PC[1] 引脚 011: PD[1] 引脚 100: PE[1] 引脚 其他: 保留
3	保留	-
2 ~ 0	EXTI0[2:0]	**EXTI 0引脚配置** 000: PA[0] 引脚 001: PB[0] 引脚 010: PC[0] 引脚 011: PD[0] 引脚 100: PE[0] 引脚 其他: 保留

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	EXTI15[2:0]			保留	EXTI14[2:0]			保留	EXTI13[2:0]			保留	EXTI12[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	EXTI11[2:0]			保留	EXTI10[2:0]			保留	EXTI9[2:0]			保留	EXTI8[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	EXTI15[2:0]	**EXTI 15引脚配置** 000: PA[15] 引脚 001: PB[15] 引脚 010: PC[15] 引脚 011: PD[15] 引脚 100: PE[15] 引脚 其他: 保留
27	保留	-
26 ~ 24	EXTI14[2:0]	**EXTI 14引脚配置** 000: PA[14] 引脚 001: PB[14] 引脚 010: PC[14] 引脚 011: PD[14] 引脚

		100: PE[14] 引脚 其他: 保留
23	**保留**	**-**
22 ~ 20	**EXTI13[2:0]**	**EXTI 13引脚配置** 000: PA[13] 引脚 001: PB[13] 引脚 010: PC[13] 引脚 011: PD[13] 引脚 100: PE[13] 引脚 其他: 保留
19	**保留**	**-**
18 ~ 16	**EXTI12[2:0]**	**EXTI 12引脚配置** 000: PA[12] 引脚 001: PB[12] 引脚 010: PC[12] 引脚 011: PD[12] 引脚 100: PE[12] 引脚 其他: 保留
15	**保留**	**-**
14 ~ 12	**EXTI11[2:0]**	**EXTI 11引脚配置** 000: PA[11] 引脚 001: PB[11] 引脚 010: PC[11] 引脚 011: PD[11] 引脚 100: PE[11] 引脚 其他: 保留
11	**保留**	**-**
10 ~ 8	**EXTI10[2:0]**	**EXTI 10引脚配置** 000: PA[10] 引脚 001: PB[10] 引脚 010: PC[10] 引脚 011: PD[10] 引脚 100: PE[10] 引脚 其他: 保留
7	**保留**	**-**
6 ~ 4	**EXTI9[2:0]**	**EXTI 9引脚配置** 000: PA[9] 引脚 001: PB[9] 引脚 010: PC[9] 引脚 011: PD[9] 引脚 100: PE[9] 引脚 其他: 保留
3	**保留**	**-**
2 ~ 0	**EXTI8[2:0]**	**EXTI 8引脚配置** 000: PA[8] 引脚 001: PB[8] 引脚 010: PC[8] 引脚 011: PD[8] 引脚 100: PE[8] 引脚 其他: 保留

15.3.10 EXTI 采样控制低寄存器 (EXTI_SAMPL)

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（Prescaler select）** 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（sample number）** 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（Prescaler select）** 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（sample number）** 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（Prescaler select）** 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（sample number）** 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（Prescaler select）** 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（sample number）** 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（Prescaler select）** 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（sample number）** 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（Prescaler select）** 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（sample number）** 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（Prescaler select）** 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（sample number）** 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（Prescaler select）** 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（sample number）** 00：1 01：2 10：3 11：4

15.3.11 EXTI 采样控制高寄存器 (EXTI_SAMPH)

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
PS15[1:0]		SN15[1:0]		PS14[1:0]		SN14[1:0]		PS13[1:0]		SN13[1:0]		PS12[1:0]		SN12[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
PS11[1:0]		SN11[1:0]		PS10[1:0]		SN10[1:0]		PS9[1:0]		SN9[1:0]		PS8[1:0]		SN8[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	**PS15[1:0]**	**EXTI15采样时钟预分频比选择位（Prescaler select）** 00：1/1 01：1/4 10：1/16 11：1/128
29 ~ 28	**SN15[1:0]**	**EXTI15连续采样次数选择位（sample number）** 00：1 01：2 10：3 11：4
27 ~ 26	**PS14[1:0]**	**EXTI14采样时钟预分频比选择位（Prescaler select）** 00：1/1 01：1/4 10：1/16 11：1/128
25 ~ 24	**SN14[1:0]**	**EXTI14连续采样次数选择位（sample number）** 00：1 01：2 10：3 11：4
23 ~ 22	**PS13[1:0]**	**EXTI13采样时钟预分频比选择位（Prescaler select）** 00：1/1 01：1/4 10：1/16 11：1/128
21 ~ 20	**SN13[1:0]**	**EXTI13连续采样次数选择位（sample number）** 00：1 01：2 10：3 11：4
19 ~ 18	**PS12[1:0]**	**EXTI12采样时钟预分频比选择位（Prescaler select）** 00：1/1 01：1/4 10：1/16 11：1/128
17 ~ 16	**SN12[1:0]**	**EXTI12连续采样次数选择位（sample number）** 00：1 01：2 10：3 11：4
15 ~ 14	**PS11[1:0]**	**EXTI11采样时钟预分频比选择位（Prescaler select）** 00：1/1 01：1/4 10：1/16 11：1/128
13 ~ 12	**SN11[1:0]**	**EXTI11连续采样次数选择位（sample number）** 00：1 01：2 10：3 11：4
11 ~ 10	**PS10[1:0]**	**EXTI10采样时钟预分频比选择位（Prescaler select）** 00：1/1 01：1/4

		10：1/16 11：1/128
9 ~ 8	**SN10[1:0]**	**EXTI10连续采样次数选择位（sample number）** 00：1 01：2 10：3 11：4
7 ~ 6	**PS9[1:0]**	**EXTI9采样时钟预分频比选择位（Prescaler select）** 00：1/1 01：1/4 10：1/16 11：1/128
5 ~ 4	**SN9[1:0]**	**EXTI9连续采样次数选择位（sample number）** 00：1 01：2 10：3 11：4
3 ~ 2	**PS8[1:0]**	**EXTI8采样时钟预分频比选择位（Prescaler select）** 00：1/1 01：1/4 10：1/16 11：1/128
1 ~ 0	**SN8[1:0]**	**EXTI8连续采样次数选择位（sample number）** 00：1 01：2 10：3 11：4

15.3.12 DMA 使能寄存器 (EXTI_DMR)

偏移地址: 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
DMRy(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	**DMRy(y=15~0)**	**线y上的DMA使能(DMA Mask on line y)** 0: 屏蔽来自线y上的DMA请求 1: 开放来自线y上的DMA请求

16. DMA 控制器

16.1 简介

直接存储器存取(DMA)用来提供在外设和存储器之间或者存储器和存储器之间的高速数据传输。无须CPU干预，数据可以通过DMA快速地移动，这就节省了CPU的资源来做其它操作。DMA控制器有8个通道，每个通道专门用来管理来自于一个或多个外设对存储器访问的请求。还有一个仲裁器来协调CPU与DMA请求的优先权。

注：对于一个双向数据传输应用，需要2个DMA通道分别完成发送和接收。

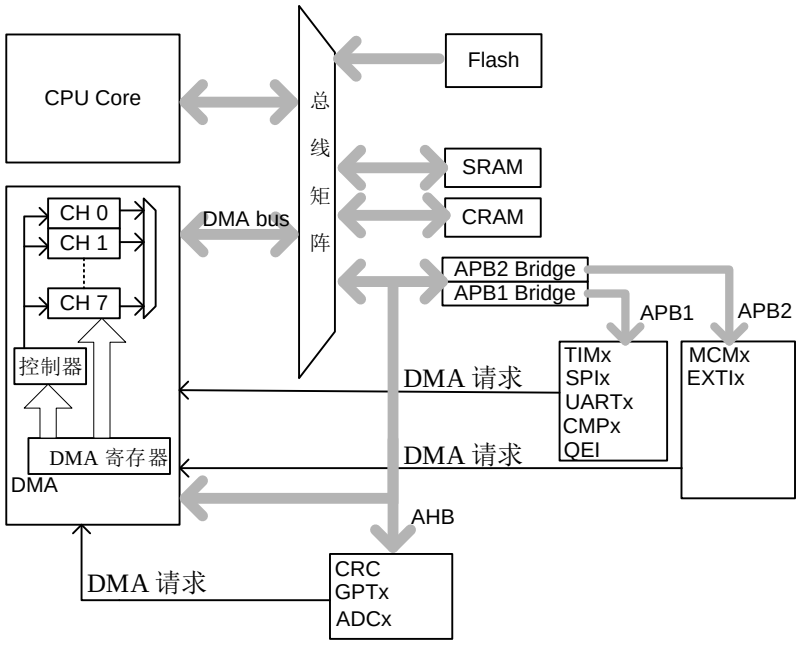


图16-1 DMA 控制器逻辑框图

16.2 主要特性

- 8个独立的可配置的通道，每个通道可支持一个单向传输
- 每个通道都直接连接专用的硬件DMA请求，每个通道都同样支持软件触发
- 支持One-Shot和To-End两种触发方式，支持软件触发方式
- 基本传输单位是突发（burst），突发长度可设，当burst长度是1时，相当于single传输
- 在同一个DMA模块上，多个请求间的优先权可以通过软件编程设置(共有四级：很高、高、中等和低)，同一优先权的访问顺序还会进行轮换处理
- 数据源和目标数据区的传输方式、传输位宽、循环方式、循环数量、指针增量都可独立配置
- 每个通道都有4个事件标志(半传输、传输完成、突发传输结束、传输出错)，这4个事件任一个都可产生中断请求
- DMA源和目标都支持存储器和外设，即支持存储器到存储器、外设到存储器、存储器到外设、外设到外设之间的传输
- SRAM、CRAM、APB1、APB2和AHB外设均可作为访问的源和目标，Flash只能作为源不能作为目标
- 可编程的数据传输数目，最大为512个突发（burst），支持自动重载
- 每个突发之间可强制释放DMA一段时间以保证CPU效率

注：通过 DMA 向 Flash 写入时没有效果，不会产生 DMA error，也不会产生 flash write error。

16.3 功能描述

DMA控制器和CPU核心共享系统数据总线，执行直接存储器数据传输。当CPU和DMA同时访问相同的目标(RAM或外设)时，DMA请求会暂停CPU访问系统总线达若干个周期，总线仲裁器执行循环调度，以保证CPU至少可以得到一半的系统总线带宽。而在突发传输模式下，DMA暂停CPU的时间还要更长。

16.3.1 DMA 处理

在发生一个事件后，外设向DMA控制器发送一个请求信号。DMA控制器根据通道的优先权处理请求。当DMA控制器开始访问发出请求的外设时，DMA控制器立即发送给它一个应答信号。当从DMA控制器得到应答信号时，外设立即释放它的请求。一旦外设释放了这个请求，DMA控制器同时撤销应答信号。如果有更多的请求时，外设可以启动下一个周期。总之，每次DMA传送由3个操作组成：

- 从源地址寄存器指示的存储器地址取数据，第一个源地址是DMA_SARx寄存器指定的外设基地址或存储器单元。
- 存数据到目标地址寄存器指示的目标存储器地址，第一个目标地址是DMA_DARx寄存器指定的外设基地址或存储器单元。
- 执行一次DMA_CPKTx寄存器的递增操作，该寄存器包含已完成的突发数目。

16.3.2 仲裁器

仲裁器有2个基本功能：

- 负责协调CPU与DMA控制器对共享总线的占用；
- 负责协调多个DMA通道的优先级处理；

16.3.3 DMA 通道

每个通道都可以在有固定地址的源存储空间和目标存储空间之间执行DMA传输。总共8个DMA通道，每个通道可连接多个外设，但每次只能选择一个外设，通过DMA_CCRx寄存器的STRMSEL[2:0]字段进行选择。

16.3.3.1 DMA 传输数量和自动重载

DMA传输数量寄存器DMA_NPKTx定义的是DMA传输的突发个数，表示NPKT+1个突发传输数量。最小设置0表示1个突发，最大设置511表示512个突发。DMA_CPKTx是DMA传输计数器，每次传输后计数器加1，直到计数到DMA_NPKTx时，根据DMA的自动重载功能设置，DMA将停止传输或者重载后继续传输。

当RELOAD@DMA_CCRx为0时自动重载关闭，CPKTx在计数到达NPKTx后将停止传输，并自动清除DMA使能位，此时计数器DMA_CPKTx保持不变，如果使能DMA传输完成中断，则申请DMA中断。DMA_CPKTx在下一次DMA使能位开启时自动清0。

当RELOAD@DMA_CCRx为1时自动重载开启，CPKTx在计数到达NPKTx后自动清0，DMA使能位保持开启，DMA传输源和目标地址恢复为初始设定值，DMA传输重新开始（不间断）。如果开启DMA传输完成中断，同样会申请DMA中断。在用户关闭RELOAD控制位后，DMA会等到当前这一轮重载传输完成后才会停止传输。

注意：

在设置To-end时，开启自动重载后不会自动开始传输，需要硬件或软件重新触发一下。

当DMA_NPKTx=0时，DMA使能打开，触发信号引起1次传输后使能关闭，DMA_CPKTx为0，表示1次突发传输。

DMA_CPKTx初始化为0，在DMA使能关闭时该初始化值并无意义，在DMA开启传输后以CPKT+1表示实际发生的突发传输数量。

DMA_NPKTx和DMA_CPKTx的计数单位都是突发（Burst）。每个突发包含1～16个DMA数据，具体见DMA_CCRx@DMA_BURSTLEN位设置。

根据DMA_CCRx@SIZE设置的不同，每个DMA数据可能是1、2、4个字节。

DMA传输数量寄存器只能在DMA关闭时设置，在开启DMA通道时，该寄存器变为只读。

16.3.3.2 可编程数据位宽

源端和目标端的传输数据位宽可以通过DMA_CCRx寄存器中的SIZE字段编程。根据DMA_CCRx@SIZE设置的不同，每个DMA数据可能是1、2、4个字节。

16.3.3.3 指针修改方式

通过设置DMA_CCRx寄存器中的SPTYP和DPTYP标志位，源端和目标端的内部当前地址指针（无法读出）在每次传输后可以有选择地完成自动修改。DMA支持指针INC、DEC、FIX和WRAP方式，其中WRAP方式用于循环模式。指针修改方式全部由软件设定。

当设置为INC方式时，下一个要传输的地址将是前一个地址加上增量值，增量值取决于所选的数据宽度不同而可取1、2或4三种情况。

DMA_SARx/DMA_DARx存放的是源端/目标端的初始指针，并且在DMA传输过程中保持不变，软件不能改变和读出当前正在传输的地址（它在内部的当前源端/目标端地址寄存器中）。

16.3.3.4 循环模式

循环模式用于处理循环缓冲区和连续的数据传输(如ADC的连续转换模式)。

循环周期固定为突发长度，循环传输完成后其地址指针会自动恢复成循环开始时的初值，无需软件更改指针。

当设置SPTYP/DPTYP=WRAP方式时就开启对应的循环模式。

下表给出了DMA循环模式的各种设置。

表16-1 DMA 循环方式设置表

循环方式	设置	功能说明
非循环	SPTYP=00/01/10，DPTYP=00/01/10	非循环模式。需要软件进行指针操作。
单循环	源端循环： SPTYP=11，DPTYP=00/01/10 目标端循环： SPTYP=00/01/10，DPTYP=11	单循环模式。比如接收多通道ADC采样数据，在ADC数据寄存器端按照通道数设置循环长度，但数据缓冲区端指针还需要软件进行处理。
双循环	SPTYP=11，DPTYP=11	双循环模式。针对某些特定的DMA传输，可以自动在源和目标端进行指针操作，不需要软件介入。

16.3.3.5 软件触发方式

DMA通道的操作可以在没有外设请求的情况下进行，这种操作就是软件触发方式。

软件触发方式一般用于存储器到存储器之间的数据搬移，但并不局限于此。当设置了DMA_CCRx寄存器中的SWTRG位之后，将启动一次软件触发，其效果与脉冲触发相同。

在做存储器到存储器数据搬移时，需注意Flash只能作为源被读取，无法写入。

16.3.3.6 单次传输与突发传输

单次传输是突发传输的特例，突发长度为1即为单次传输。

单次传输是DMA的基本传输方式，DMA仲裁策略能够保证CPU至少获得一半的运行时间，从而在保证CPU运行时间与解放CPU干预外设负担之间达成较为理想的平衡，有助于提升系统运行效率。

突发传输（指突发长度大于1）适用于快速小块数据搬移，具体见“仲裁器”部分描述。

16.3.3.7 One-shot 触发与 To-end 触发

One-shot触发每次只进行一次DMA突发传输，To-end触发后将把DMA_NPKTx寄存器中的全部DMA传输数量的突发传输完成后才停止。

To-end触发时前后突发之间同样存在IDLE时隙供CPU运行。

16.3.3.8 BUSY 信号

DMA通道在传输时会置出busy信号，用户查询各通道的busy信号可确定当前哪一个通道在传输。BUSY信号在One-Shot时只指示一次突发传输，在To-End时需要持续到传输完成。

16.3.3.9 Burst Idle 设置

DMA在传输完成一个突发后，释放总线给CPU执行。如果有多个DMA通道开启，则需要等待当前通道的Burst Idle结束后才能执行下一个DMA通道，即CPU的执行时间不会被排队的DMA抢占，这种设计保证了CPU的执行效率。

16.3.3.10 通道配置过程

下面是配置DMA通道x的过程(x代表通道号，采用非循环单次传输单次触发模式说明)：

（1）在DMA_SARx寄存器中设置源数据区的首地址。

（2）在DMA_DARx寄存器中设置目标数据区的首地址。

（3）在DMA_NPKTx寄存器中设置要传输的数据量。

（4）在DMA_CCRx寄存器中设置DMA通道映像表STRMSEL[2:0]，每个DMA通道只能选择一个外设。

（5）在DMA_CCRx寄存器中设置突发长度、触发方式、指针修改方式、数据位宽、通道优先级、中断使能。

（6）设置DMA_CCRx寄存器的ENABLE位，启动该通道。

一旦启动了DMA通道，它即可响应连到该通道上的内存或外设的DMA请求。

当传输一半的数据后，半传输标志(HTIF)被置‘1’，当设置了允许半传输中断位(HTIE)时，将产生一个中断请求。在数据传输结束后，传输完成标志(TCIF)被置‘1’，当设置了允许传输完成中断位(TCIE)时，将产生一个中断请求。

注：有关外设的DMA请求使能，可以通过设置相应外设寄存器中的控制位（在对应外设章节中），被独立地开启或关闭，如果配置并开启了对应的DMA通道，而相应的外设的DMA请求未开启，则硬件不做处理，DMA传输也不会发生。

16.3.4 源和目标数据传输格式

当SSIZE和DSIZE不相同时，DMA模块按照下表格式进行数据传输。

表16-2 DMA 源和目标数据传输格式对照表（当 SPTYP=DPTYP=00）

源端宽度	目标宽度	传输数目	源：地址/数据	目标：地址/数据
8	**8**	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3
8	16	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	0x0 / 00B0 0x2 / 00B1 0x4 / 00B2 0x6 / 00B3
8	32	4	0x0 / B0 0x1 / B1 0x2 / B2 0x3 / B3	0x0 / 000000B0 0x4 / 000000B1 0x8 / 000000B2 0xC / 000000B3
16	8	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	0x0 / B0 0x1 / B2 0x2 / B4 0x3 / B6
16	**16**	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6
16	32	4	0x0 / B1B0 0x2 / B3B2 0x4 / B5B4 0x6 / B7B6	0x0 / 0000B1B0 0x4 / 0000B3B2 0x8 / 0000B5B4 0xC / 0000B7B6
32	8	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	0x0 / B0 0x1 / B4 0x2 / B8 0x3 / BC
32	16	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	0x0 / B1B0 0x2 / B5B4 0x4 / B9B8 0x6 / BDBC
32	**32**	4	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC	0x0 / B3B2B1B0 0x4 / B7B6B5B4 0x8 / BBBAB9B8 0xC / BFBEBDBC

16.3.5 错误管理

读写一个保留的地址区域，将会产生DMA传输错误，具体包括

（1） CRAM区域（CRAM设置为read only时）进行写入

（2） 对Cortex-M3私有外设区进行读写

（3） 在DMA读写时发生的其它总线错误

当在DMA读写操作时发生DMA传输错误，硬件会自动地清除发生错误的通道所对应的通道配置寄存器(DMA_CCRx)的EN位，该通道操作被停止（注：需等待当前Burst传输完成）。此时，在DMA_IFR寄存器中对应该通道的传输错误中断标志位(TEIF)将被置位，如果在DMA_CCRx寄存器中设置了传输错误中断允许位，则将产生中断。

16.3.6 DMA 中断

每个DMA通道都有4种传输事件，当传输事件发生时可以产生中断。即突发传输结束(BE)、半传输(HT)、传输完成(TC)和传输错误(TE)。DMA配置寄存器CCRx 中设置相关控制位可使能相关中断事件。如果BE、HT、TC 或TE 四个事件中有任意一个事件发生时，将产生中断请求。

在中断向量表中，DMA模块占用3个入口，分别为DMA_CH1、DMA_CH2、DMA_CH3～8。

表16-3　　DMA 中断请求

中断事件	事件标志位	使能控制位
突发传输结束	BEIF	BEIE
传输过半	HTIF	HTIE
传输完成	TCIF	TCIE
传输错误	TEIF	TEIE

16.3.7 DMA 请求映像

DMA控制器从外设MCMx、GPTx、QEIx、ADCx、UARTx、SPIx、CMPx、EXTIx、TIMx产生的请求，通过输入源选择器（STRMSEL[2:0]）输入到仲裁器，这意味着每一个DMA通道同时只能有一个请求有效。参见下图的DMA请求映像。

表16-4　　各个通道的 DMA 请求一览

DMA输入源选择 STRMSEL[2:0]	DMA Channel 0	DMA Channel 1	DMA Channel 2	DMA Channel 3	DMA Channel 4	DMA Channel 5	DMA Channel 6	DMA Channel 7
stream0	UART2_TX	UART2_RX	UART1_TX	UART1_RX	UART1_TX	UART1_RX	UART3_RX	UART3_TX
stream1	SPI1_TX	SPI1_RX	-	-	UART2_TX	UART2_RX	SPI1_TX	SPI1_RX
stream2	EXTI0/1	EXTI2/3	EXTI4/5	EXTI6/7	EXTI8/9	EXTI10/11	EXTI12/13	EXTI14/15
stream3	ADC1	-	ADC3	ADC1	ADC1	-	ADC3	-
stream4	MCM1	-	MCM1	-	MCM1	-	MCM1	-
stream5	GPT0	GPT1	GPT2	-	GPT0	GPT1	GPT2	-
stream6	CMP1	QEI	GPT	CMP2	QEI	-	CMP3	GPT（同步）
stream7	TIM5	TIM6	TIM7	TIM8	TIM5	TIM6	TIM7	TIM8

16.4 寄存器

DMA 模块寄存器列表 (基地址:0x0x4004 4800)

地址	寄存器名	说明
0x4004 4800	IFSR	DMA中断状态寄存器
0x4004 4804	IFCR	DMA中断标志清除寄存器
0x4004 4808	CSR	DMA控制状态寄存器
0x4004 4810	CCR0	DMA通道0配置寄存器
0x4004 4814	NPKT0	DMA通道0传输数量寄存器
0x4004 4818	CPKT0	DMA通道0传输计数寄存器
0x4004 481C	SAR0	DMA通道0源地址寄存器
0x4004 4820	DAR0	DMA通道0目标地址寄存器
0x4004 4830	CCR1	DMA通道1配置寄存器
0x4004 4834	NPKT1	DMA通道1传输数量寄存器
0x4004 4838	CPKT1	DMA通道1传输计数寄存器
0x4004 483C	SAR1	DMA通道1源地址寄存器
0x4004 4840	DAR1	DMA通道1目标地址寄存器
0x4004 4850	CCR2	DMA通道2配置寄存器
0x4004 4854	NPKT2	DMA通道2传输数量寄存器
0x4004 4858	CPKT2	DMA通道2传输计数寄存器
0x4004 485C	SAR2	DMA通道2源地址寄存器
0x4004 4860	DAR2	DMA通道2目标地址寄存器
0x4004 4870	CCR3	DMA通道3配置寄存器
0x4004 4874	NPKT3	DMA通道3传输数量寄存器
0x4004 4878	CPKT3	DMA通道3传输计数寄存器
0x4004 487C	SAR3	DMA通道3源地址寄存器
0x4004 4880	DAR3	DMA通道3目标地址寄存器
0x4004 4890	CCR4	DMA通道4配置寄存器
0x4004 4894	NPKT4	DMA通道4传输数量寄存器
0x4004 4898	CPKT4	DMA通道4传输计数寄存器
0x4004 489C	SAR4	DMA通道4源地址寄存器
0x4004 48A0	DAR4	DMA通道4目标地址寄存器
0x4004 48B0	CCR5	DMA通道5配置寄存器
0x4004 48B4	NPKT5	DMA通道5传输数量寄存器
0x4004 48B8	CPKT5	DMA通道5传输计数寄存器
0x4004 48BC	SAR5	DMA通道5源地址寄存器
0x4004 48C0	DAR5	DMA通道5目标地址寄存器
0x4004 48D0	CCR6	DMA通道6配置寄存器
0x4004 48D4	NPKT6	DMA通道6传输数量寄存器
0x4004 48D8	CPKT6	DMA通道6传输计数寄存器
0x4004 48DC	SAR6	DMA通道6源地址寄存器
0x4004 48E0	DAR6	DMA通道6目标地址寄存器
0x4004 48F0	CCR7	DMA通道7配置寄存器
0x4004 48F4	NPKT7	DMA通道7传输数量寄存器
0x4004 48F8	CPKT7	DMA通道7传输计数寄存器
0x4004 48FC	SAR7	DMA通道7源地址寄存器
0x4004 4900	DAR7	DMA通道7目标地址寄存器

16.4.1 DMA 中断状态寄存器 (DMA_IFSR)

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
TEIFy(y=7~0)								HTIFy(y=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
TCIFy(y=7~0)								BEIFy(y=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	**TEIFy(y=7~0)**	**通道y的传输错误标志，硬件置位，在DMA_IFCR寄存器的相应位写入'1'可以清除对应标志。** 0：在通道y没有传输错误（TE）事件 1：在通道y产生了传输错误（TE）事件
23 ~ 16	**HTIFy(y=7~0)**	**通道y的半传输标志，硬件置位，在DMA_IFCR寄存器的相应位写入'1'可以清除对应标志。** 0：在通道y没有半传输（HT）事件 1：在通道y产生了半传输（HT）事件
15 ~ 8	**TCIFy(y=7~0)**	**通道y的传输完成标志，硬件置位，在DMA_IFCR寄存器的相应位写入'1'可以清除对应标志。** 0：在通道y没有传输完成（TC）事件 1：在通道y产生了传输完成（TC）事件
7 ~ 0	**BEIFy(y=7~0)**	**通道y的块传输完成中断标志，硬件置位，在DMA_IFCR寄存器的相应位写入'1'可以清除对应标志。** 0：在通道y没有块传输完成（BE）事件 1：在通道y产生了块传输完成（BE）事件

16.4.2 DMA 中断标志清除寄存器 (DMA_IFCR)

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
CTEIFy(y=7~0)								CHTIFy(y=7~0)							
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
CTCIFy(y=7~0)								CBEIFy(y=7~0)							
WO								WO							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 24	**CTEIFy(y=7~0)**	**清除通道y的传输错误标志，软件清除** 0：不起作用 1：清除DMA_ISR寄存器中的对应的TEIF标志
23 ~ 16	**CHTIFy(y=7~0)**	**清除通道y的半传输标志，软件清除** 0：不起作用 1：清除DMA_ISR寄存器中的对应的HTIF标志
15 ~ 8	**CTCIFy(y=7~0)**	**清除通道y的传输完成标志，软件清除** 0：不起作用 1：清除DMA_ISR寄存器中的对应的TCIF标志
7 ~ 0	**CBEIFy(y=7~0)**	**清除通道y的块传输完成中断标志，软件清除** 0：不起作用 1：清除DMA_ISR寄存器中的对应的BEIF标志

16.4.3 DMA 控制状态寄存器 (DMA_CSR)

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
RELOADy(y=7~0)								保留				BURSTIDLE[3:0]			
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
DBUSYy(y=7~0)								SWTRGy(y=7~0)							
RO								RW1s							
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 24	**RELOADy(y=7~0)**	**DMA通道y的重载控制位，该位由软件设置和清除** 0：不自动重载 1：自动重载
23 ~ 20	**保留**	**-**
19 ~ 16	**BURSTIDLE[3:0]**	**DMA突发传输后释放周期设置位，该位由软件设置和清除** 0000：释放1个周期 0001：释放2个周期 0010：释放3个周期 0011：释放4个周期 0100：释放5个周期 0101：释放6个周期 0110：释放7个周期 0111：释放8个周期 1000：释放9个周期 1001：释放10个周期 1010：释放12个周期 1011：释放16个周期 1100：释放20个周期 1101：释放24个周期 1110：释放28个周期 1111：释放32个周期
15 ~ 8	**DBUSYy(y=7~0)**	**DMA通道y的状态，由硬件设置与清除，包含软件硬件触发状态** 0：idle 1：busy
7 ~ 0	**SWTRGy(y=7~0)**	**DMA通道y的软件触发，该位由软件设置，硬件清除** 0：无软件触发，可读0，写0无意义 1：置位软件触发，在DMA响应后自动清0

16.4.4 DMA 通道 0 配置寄存器 (DMA_CCRn)(n=0..7)

偏移地址: 0x0010
:0x0030
:0x0050
:0x0070
:0x0090
:0x00B0
:0x00D0
:0x00F0

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
保留								TRGMODE	STRMSEL[2:0]			BURSTLEN[3:0]			
-								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
PL[1:0]		SSIZE[1:0]		DSIZE[1:0]		SPTYP[1:0]		DPTYP[1:0]		保留	TEIE	BEIE	HTIE	TCIE	EN
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 ~ 24	保留	-
23	TRGMODE	DMA触发模式选择位，该位由软件设置和清除 0：one-shot触发方式 1：to-end触发方式
22 ~ 20	STRMSEL[2:0]	DMA通道输入源选择控制位，这些位由软件设置和清除 000：选择stream0输入 001：选择stream1输入 ..... 111：选择stream7输入
19 ~ 16	BURSTLEN[3:0]	DMA突发长度设置位，该位由软件设置和清除 0000：每个burst包含1个数据（即single传输） 0001：每个burst包含2个数据 … 1111：每个burst包含16个数据 注：根据SIZE设置的不同，1个数据传输可能包含1字节/2字节/4字节。
15 ~ 14	PL[1:0]	通道优先级，这些位由软件设置和清除 00：低 01：中 10：高 11：最高
13 ~ 12	SSIZE[1:0]	源端数据宽度，这些位由软件设置和清除 00：8位 01：16位 10：32位 11：保留
11 ~ 10	DSIZE[1:0]	目标端数据宽度，这些位由软件设置和清除 00：8位 01：16位 10：32位 11：保留
9 ~ 8	SPTYP[1:0]	源端指针修改方式，该位由软件设置和清除 00：递增 01：递减

		10：固定 11：循环（递增循环）
7 ~ 6	**DPTYP[1:0]**	**目标端指针修改方式，该位由软件设置和清除** 00：递增 01：递减 10：固定 11：循环（递增循环）
5	**保留**	**-**
4	**TEIE**	**允许传输错误中断，该位由软件设置和清除** 0：禁止TE中断 1：允许TE中断
3	**BEIE**	**允许burst传输结束中断，该位由软件设置和清除** 0：禁止BE中断 1：允许BE中断
2	**HTIE**	**允许半传输中断，该位由软件设置和清除** 0：禁止HT中断 1：允许HT中断
1	**TCIE**	**允许传输完成中断，该位由软件设置和清除** 0：禁止TC中断 1：允许TC中断
0	**EN**	**通道使能控制位，由软件设置与清除** 0：通道关闭 1：通道开启

16.4.5 DMA 通道 0 传输数量寄存器 (DMA_NPKTn)(n=0..7)

偏移地址: 0x0014
:0x0054
:0x0074
:0x0094
:0x00B4
:0x00D4
:0x00F4

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					NPKT[10:0]										
-					*RW*										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 11	**保留**	**-**
10 ~ 0	**NPKT[10:0]**	**通道n数据传输数量（burst数）** 数据传输数量为NPKT，设置范围0～511，表示传输burst数量从1至512个。这个寄存器只能在通道不工作(DMA_CCRx的EN=0)时写入。通道开启后该寄存器变为只读。

16.4.6 DMA 通道 0 传输计数寄存器 (DMA_CPKTn)(n=0..7)

偏移地址: 0x0018
:0x0038
:0x0058
:0x0078

:0x0098
:0x00B8
:0x00D8
:0x00F8

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					CPKT[10:0]										
-					*RO*										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 11	保留	-
10 ~ 0	**CPKT[10:0]**	**通道n数据传输计数（burst数）** 该寄存器用于指示DMA传输数量，只读。开启DMA通道后，该计数器自动清0，并根据DMA传输进行计数，当计数到NPKT数量时，停止DMA传输，清除DMA通道使能位，并保持计数值不变。 该寄存器设置范围0～511，表示传输burst数量从1至512个。

16.4.7 DMA 通道 0 源地址寄存器 (DMA_SARn)(n=0..7)

偏移地址: 0x001C
:0x003C
:0x005C
:0x007C
:0x009C
:0x00BC
:0x00DC
:0x00FC

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
SAR[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
SAR[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	**SAR[31:0]**	**DMA源地址寄存器 (Source address register)** 源数据寄存器的基地址 当SSIZE='01'(16位)，不使用SAR[0]位。操作自动地与半字地址对齐。 当SSIZE='10'(32位)，不使用SAR[1:0]位。操作自动地与字地址对齐。 当开启通道(DMA_CCRx的EN=1)时不能写该寄存器

16.4.8 DMA 通道 0 目标地址寄存器 (DMA_DARn)(n=0..7)

偏移地址: 0x0020
:0x0040
:0x0060
:0x0080
:0x00A0
:0x00C0

:0x00E0
:0x0100
复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
DAR[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
DAR[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	**DAR[31:0]**	**DMA目标地址寄存器 Destination address register)** 目标数据寄存器的基地址 当DSIZE='01'(16位)，不使用DAR[0]位。操作自动地与半字地址对齐。 当DSIZE='10'(32位)，不使用DAR[1:0]位。操作自动地与字地址对齐。 当开启通道(DMA_CCRx的EN=1)时不能写该寄存器

16.4.9 DMA 通道 1 传输数量寄存器 (DMA_NPKT1)

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
保留					NPKT[10:0]										
-					RW										
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	符号	说明
31 ~ 11	保留	-
10 ~ 0	**NPKT[10:0]**	**DMA通道1传输计数寄存器** 数据传输数量为NPKT，设置范围0～511，表示传输burst数量从1至512个。这个寄存器只能在通道不工作(DMA_CCRx的EN=0)时写入。通道开启后该寄存器变为只读。

17. 电机控制模块（Motor Control Module）

17.1 简介

SH32F284内建一个电机控制PWM模块（MCM1）。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17-1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。

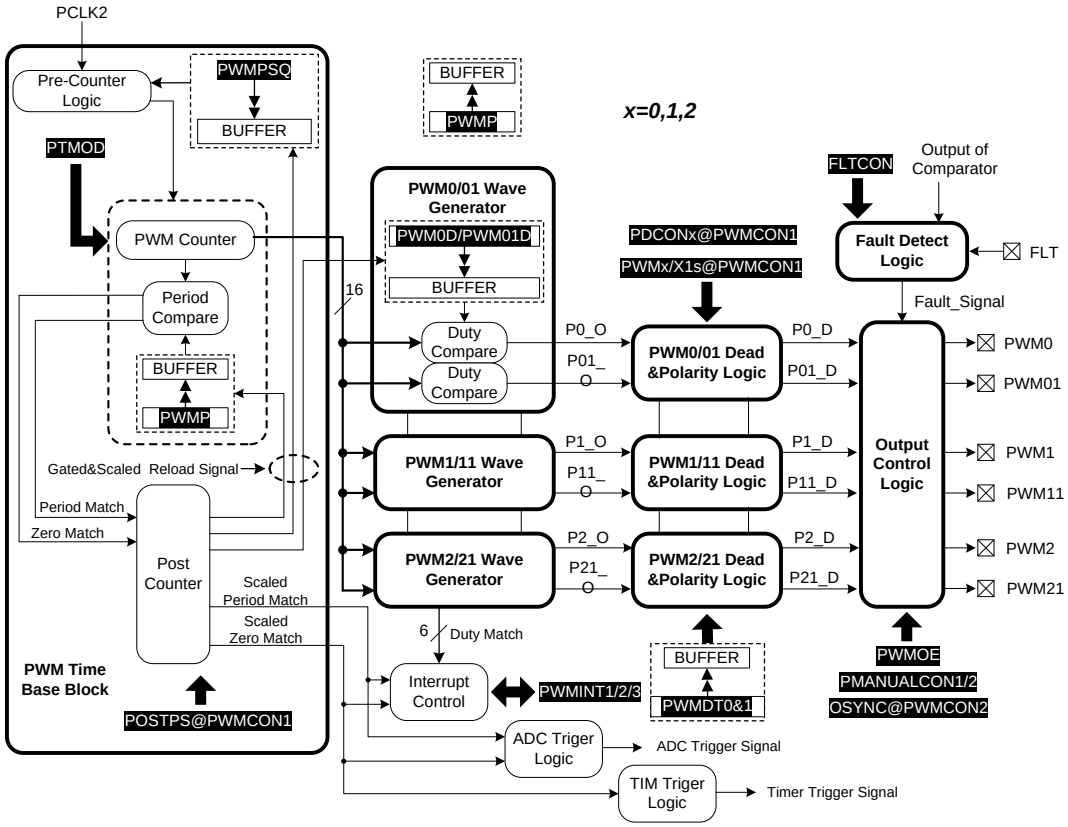


图17-1　　PWM 模块简图

17.2 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）

17.3 功能描述

17.3.1 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17-2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17-2给出了时基模块的逻辑简图。

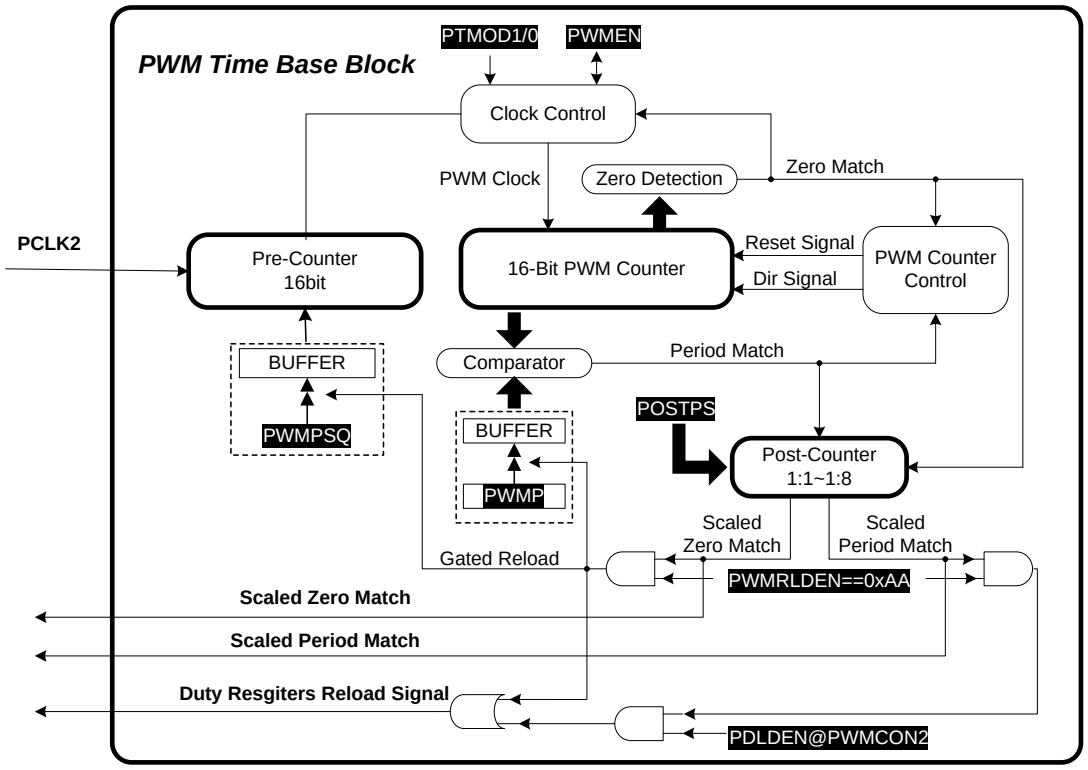


图17-2　　PWM 时基逻辑简图

17.3.1.2　PWM 工作时钟

PWM时钟（PWM Clock）由APB2时钟（PCLK2）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PCLK2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从0置'1'时，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。

17.3.1.3　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。

17.3.1.4　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PCLK2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17-3所示。

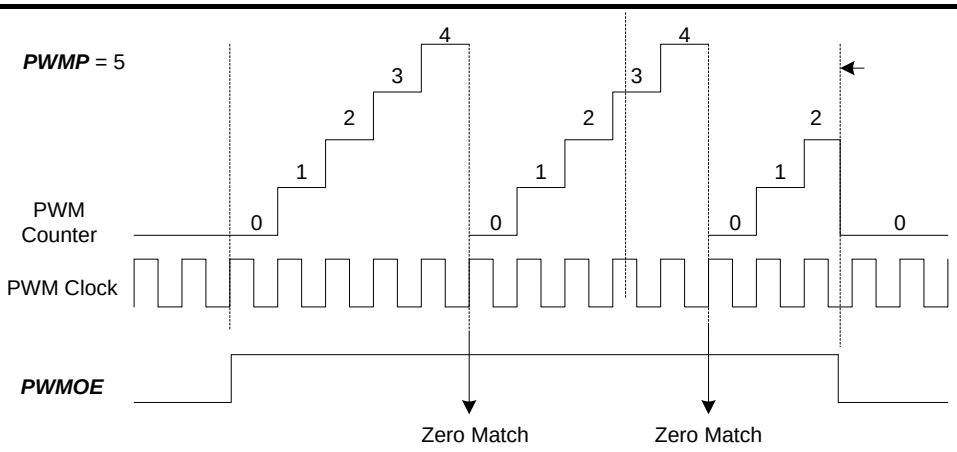


图17-3　　边沿对齐计数模式

17.3.1.5　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PCLK2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17-4所示。

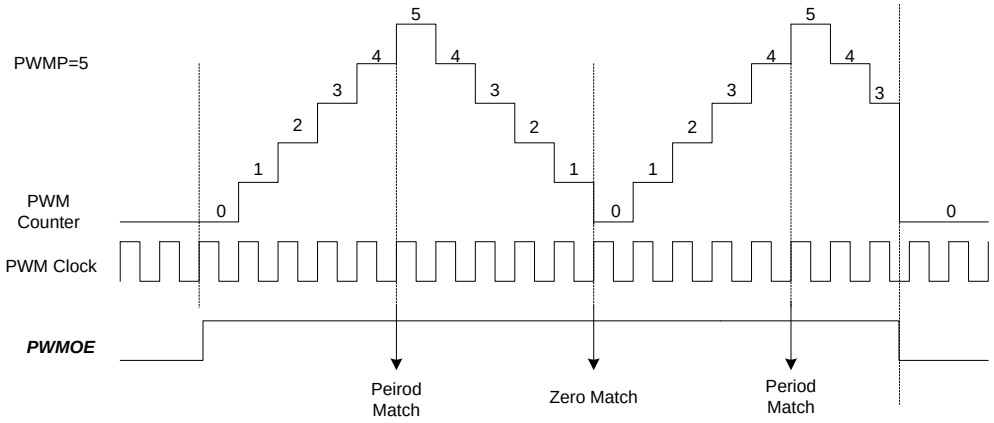


图17-4　　中心对齐计数模式

17.3.1.6　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PCLK2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17-5所示。

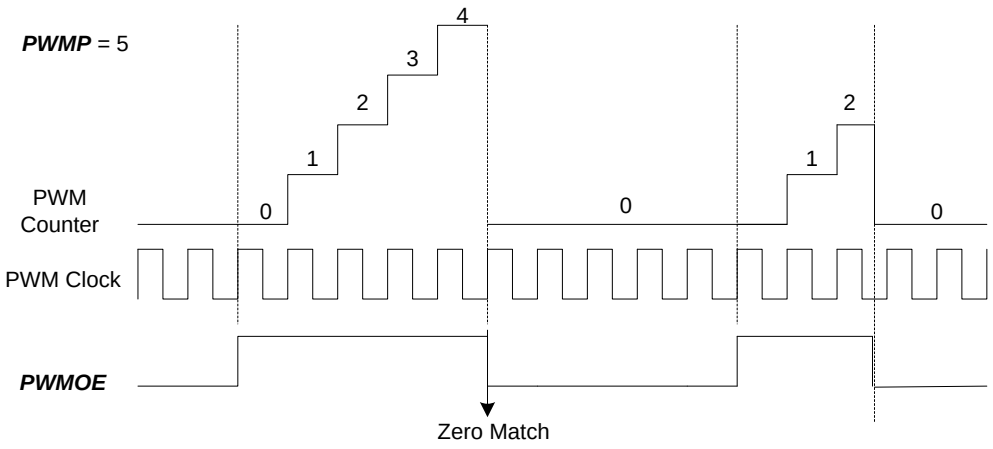


图17-5　　单次计数模式

17.3.1.7　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17-2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17-1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。

17.3.1.8　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’。

17.3.1.9　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17-2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17-2所示。PCLK2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17-2所示。PCLK2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：这3个寄存器的重载时刻不可配置，只能发生在归零匹配。

17.3.2 PWM0/1/2 波形发生模块

PWM波形发生模块指图17-1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17-1所示。

波形发生器的逻辑简图如图17-6所示。

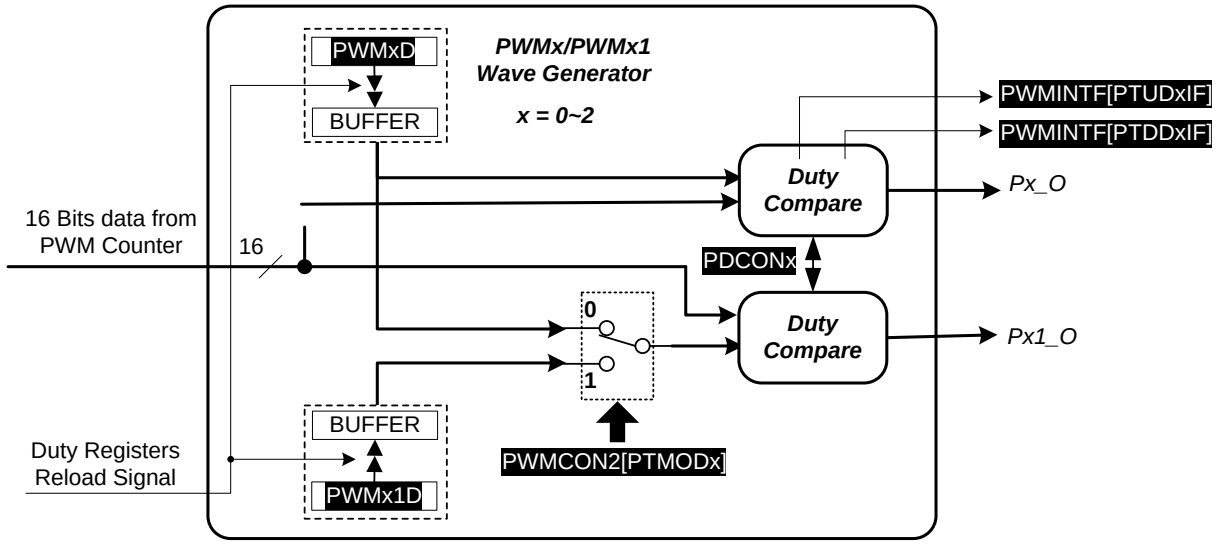


图17-6　　波形发生器逻辑简图

17.3.2.1　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17-1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。

17.3.2.2　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17.3.3节“PWM死区与极性控制模块”）。（x = 0 - 2）

17.3.2.3　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17.3.3节“PWM死区与极性控制模块”）。（x = 0 - 2）

17.3.2.4　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）。

17.3.2.5 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17-7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。

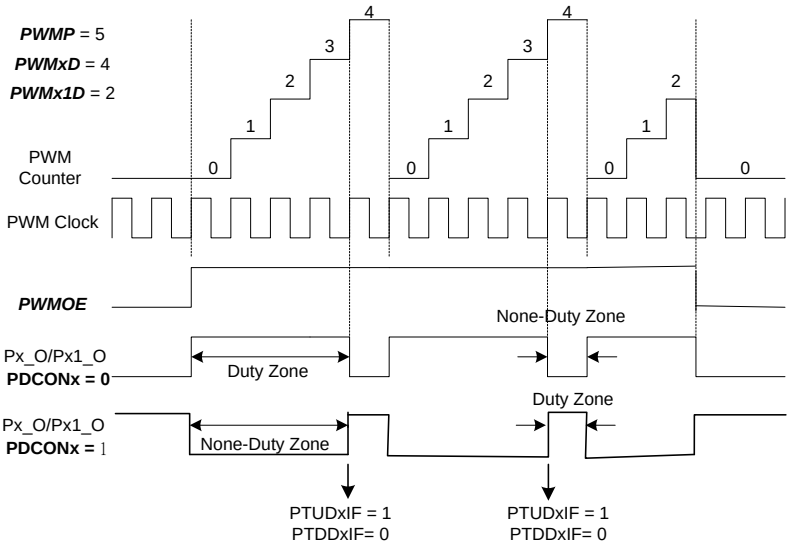


图17-7　　边沿对齐计数、互补输出模式下的 PWM 原始波形

在独立输出模式下，PWM原始输出波形Px_O和Px1_O如图17-8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。

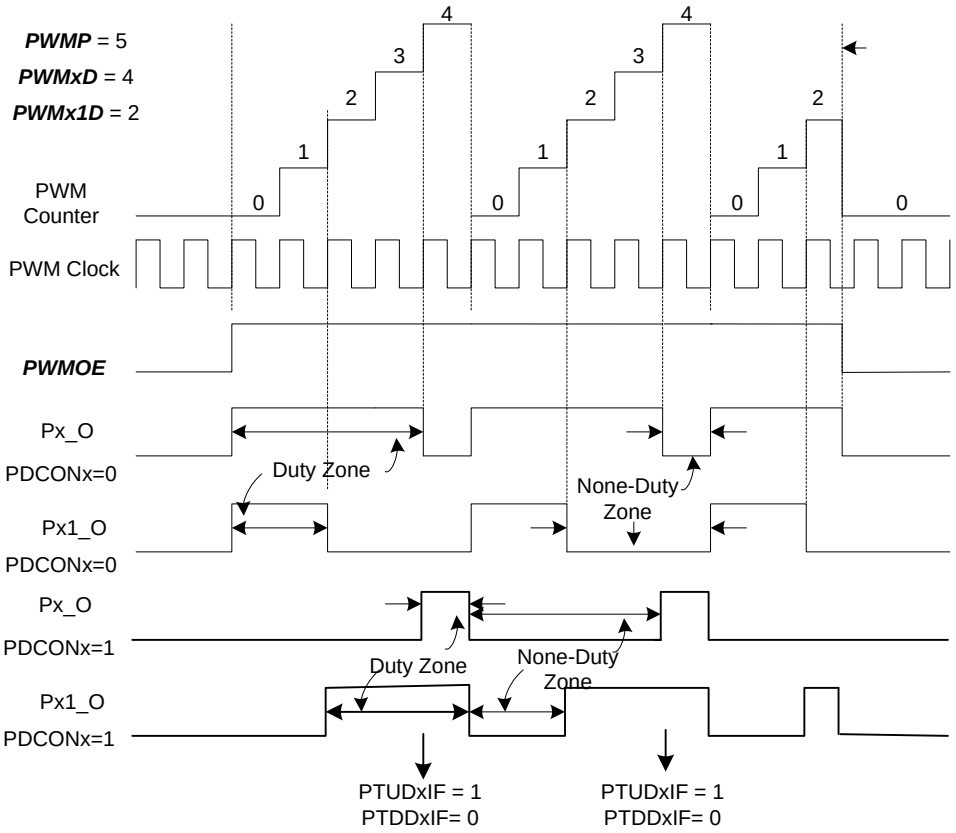


图17-8　　边沿对齐计数、独立输出模式下的 PWM 原始波形

17.3.2.6　　中心对齐计数模式下的 PWM0/1/2 原始波形

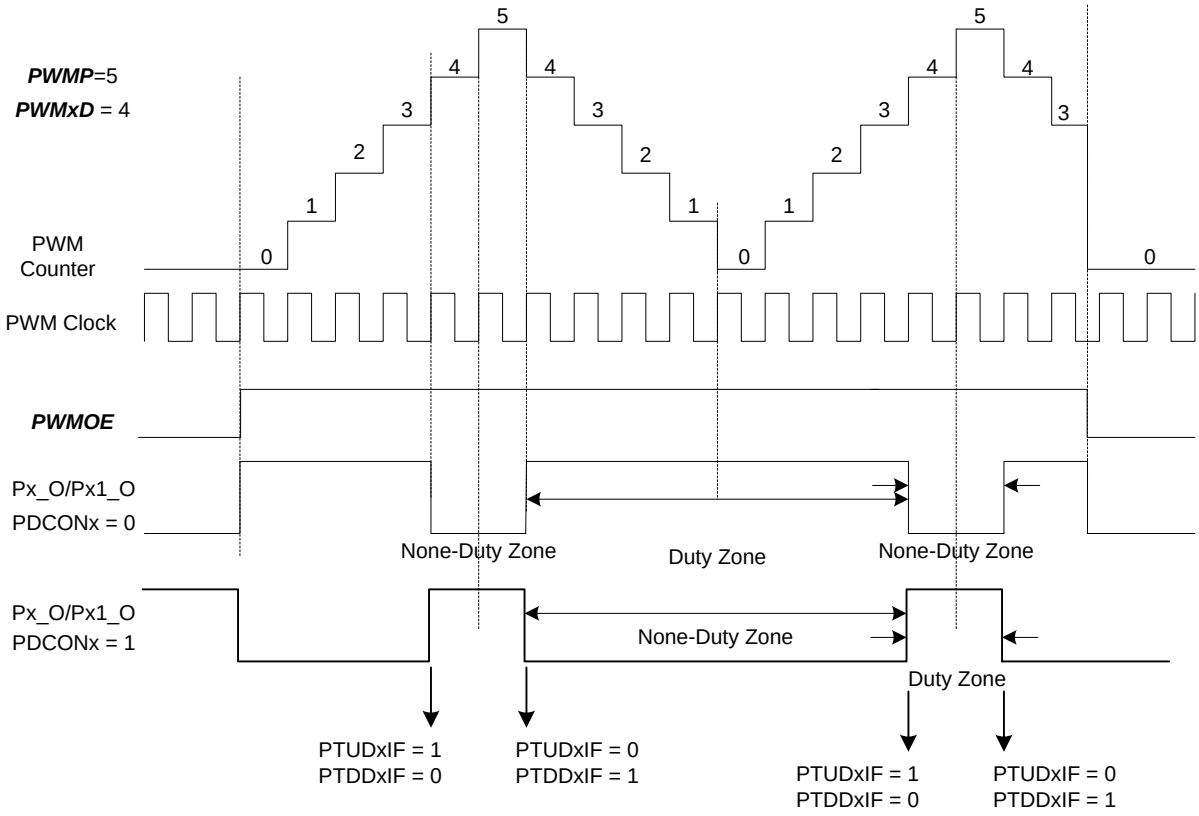


图17-9　　中心对齐计数、互补输出模式下的 PWM 原始波形（对称波形）

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17-9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。

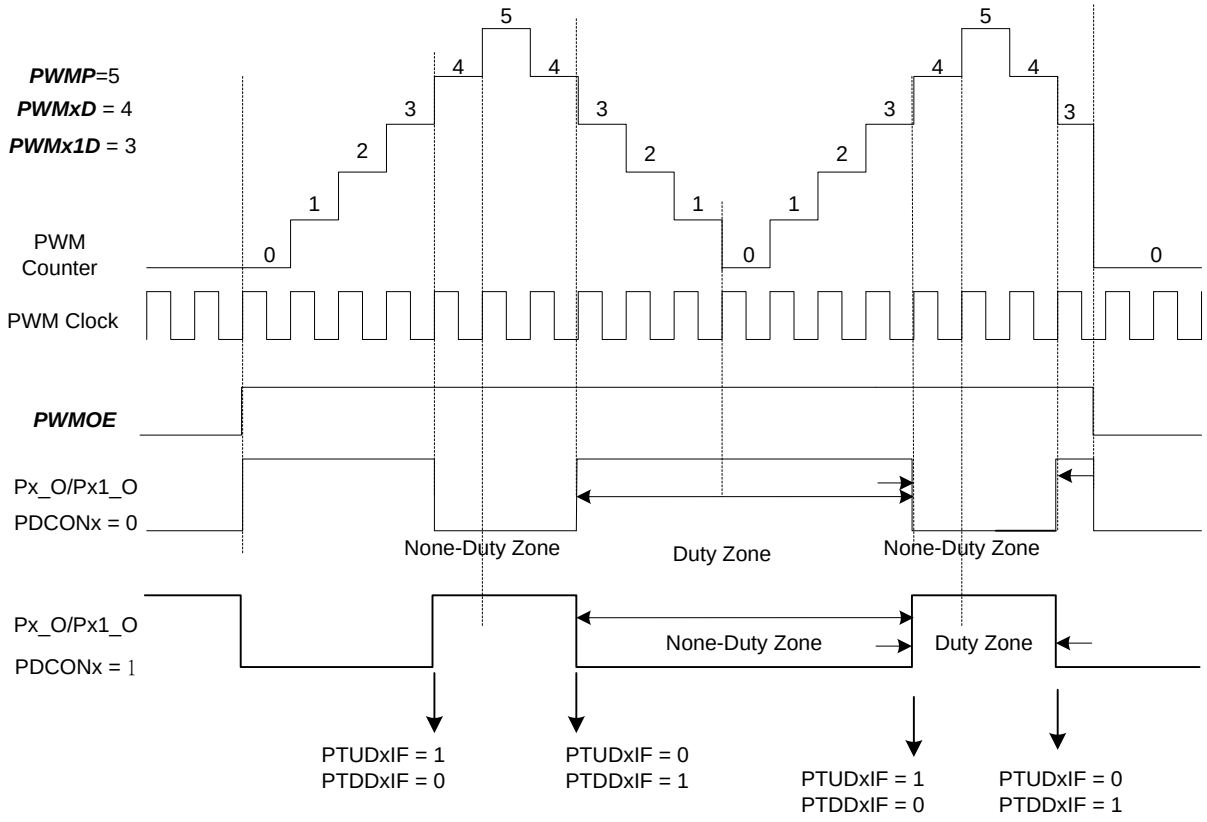


图17-10　　中心对齐计数、互补输出模式下的 PWM 原始波形（非对称波形）

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17-10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。

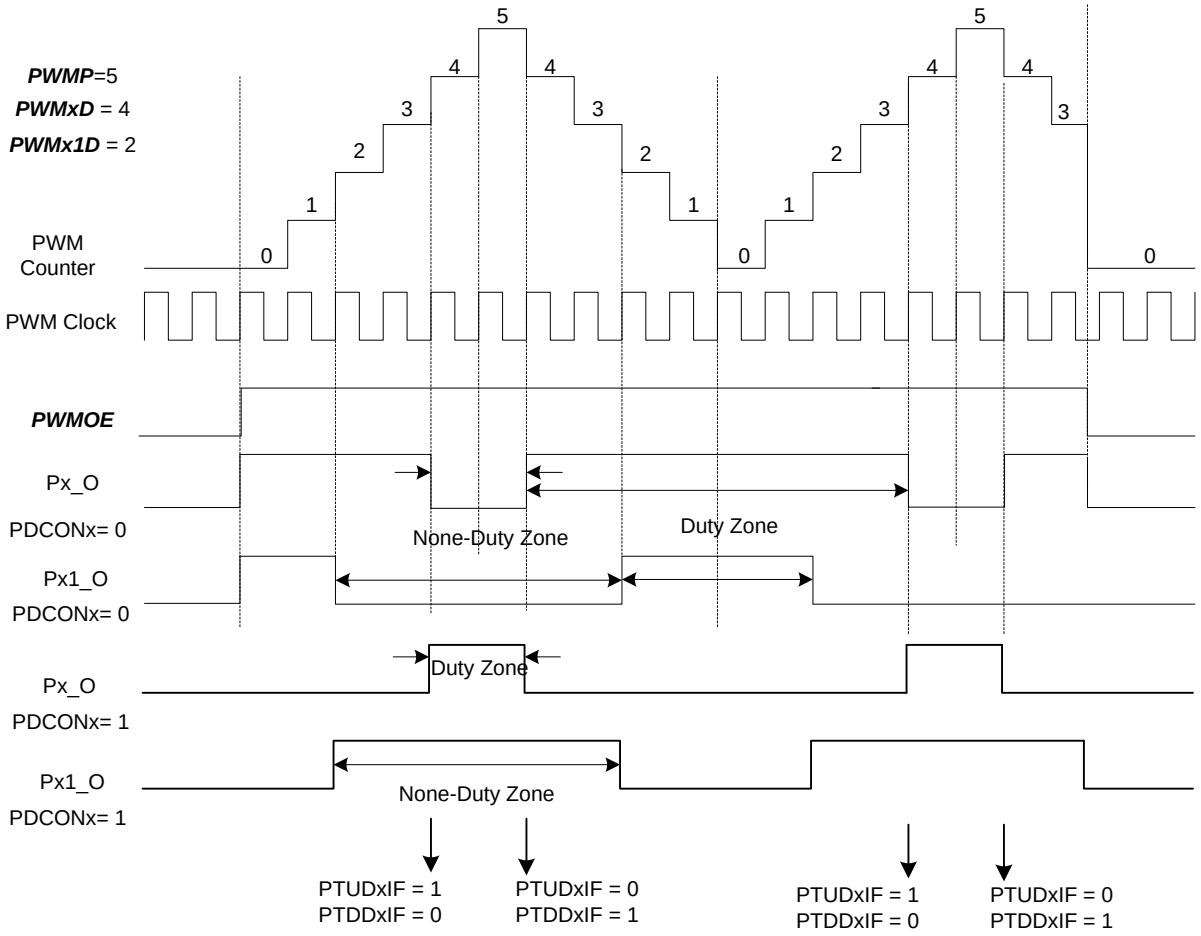


图17-11　　中心对齐计数、独立输出模式下的 PWM 原始波形

独立输出模式下，PWM原始输出波形Px_O和Px1_O如图17-11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。

17.3.2.7　　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17-12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。

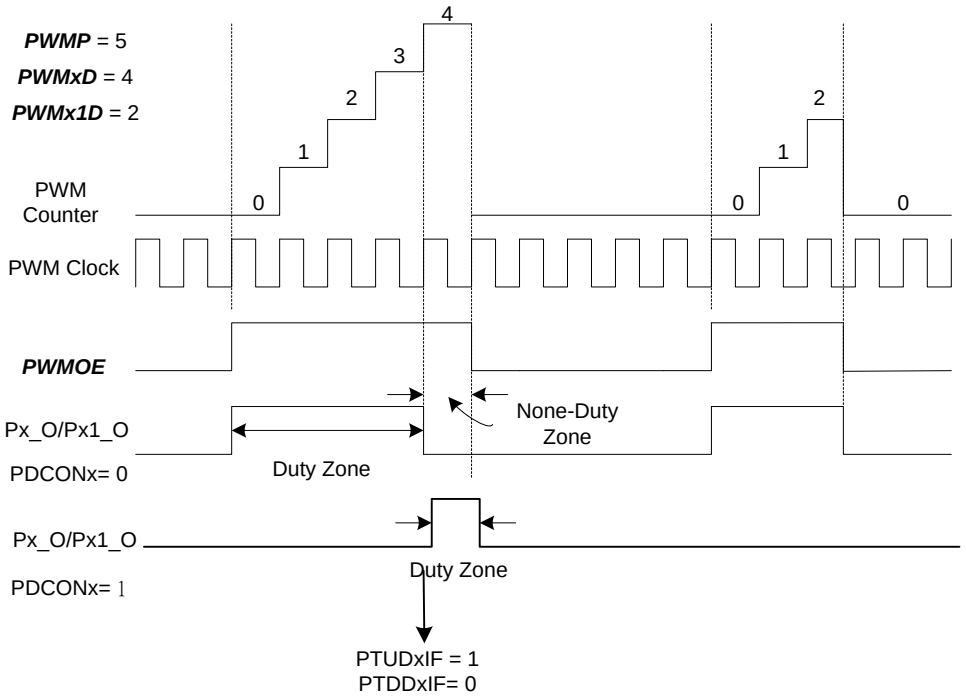


图17-12　　单次计数、互补输出模式下的 PWM 原始波形

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。

17.3.2.8　　占空比定义及占空比寄存器的自动重载

如图17-6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17-2和图17-6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17-6所示。

由于存在缓冲寄存器，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位为1，则分频后的周期匹配信号也会将硬件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。

17.3.2.9　　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17-7/图17-9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17-9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’。

17.3.2.10 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，同时缓冲运行功能不变。

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。

注意：当DILDEN@PWMCON2 =1时，占空比寄存器也受PWMCON2寄存器的PDLDEN和ZDLDEN位控制，同时死区寄存器和PDCONx寄存器的更新方式还是受这两位控制。

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：

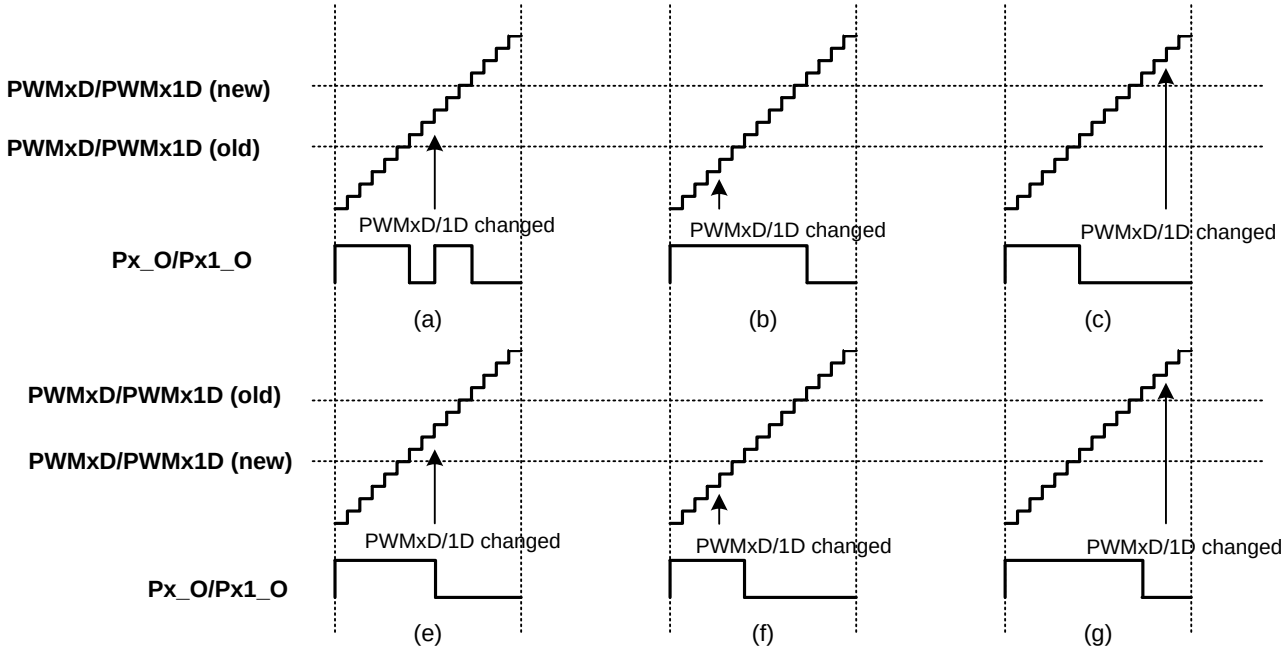


图17-13 递增计数过程的 PWMxD 和 PWMx1D 寄存器更新(x=0~2)

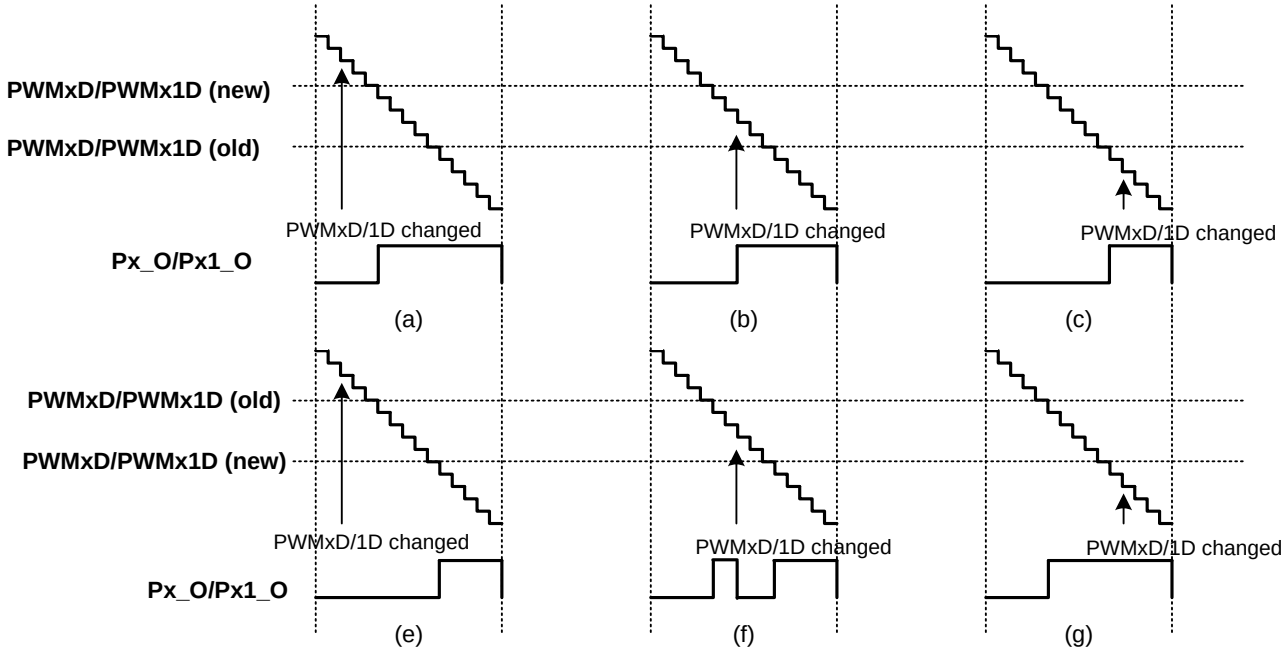


图17-14　　递减计数过程的 PWMxD 和 PWMx1D 寄存器更新(x=0~2)

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。

17.3.2.11　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，同时缓冲运行功能不变。

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。

17.3.3　PWM 死区与极性控制模块

如图17-1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17-15所示。

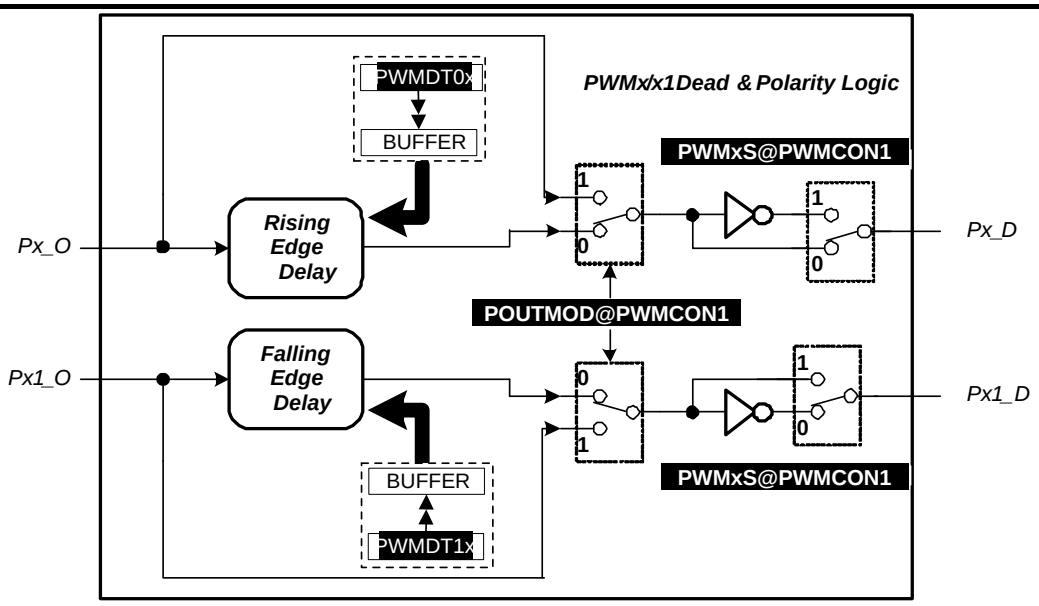


图17-15　死区与极性控制逻辑简图

17.3.3.2　PWM 死区的实现

如图17-15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。

17.3.3.3　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17-7 - 图17-11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17-7 - 图17-11所示的PWM原始波形Px_O/Px1_O和图17-15可以清晰的反映死区与极性的实现。

17.3.3.4　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，有自己的缓冲寄存器（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。

17.3.3.5　不同极性设置下带死区的 PWM 波形

图17-16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。

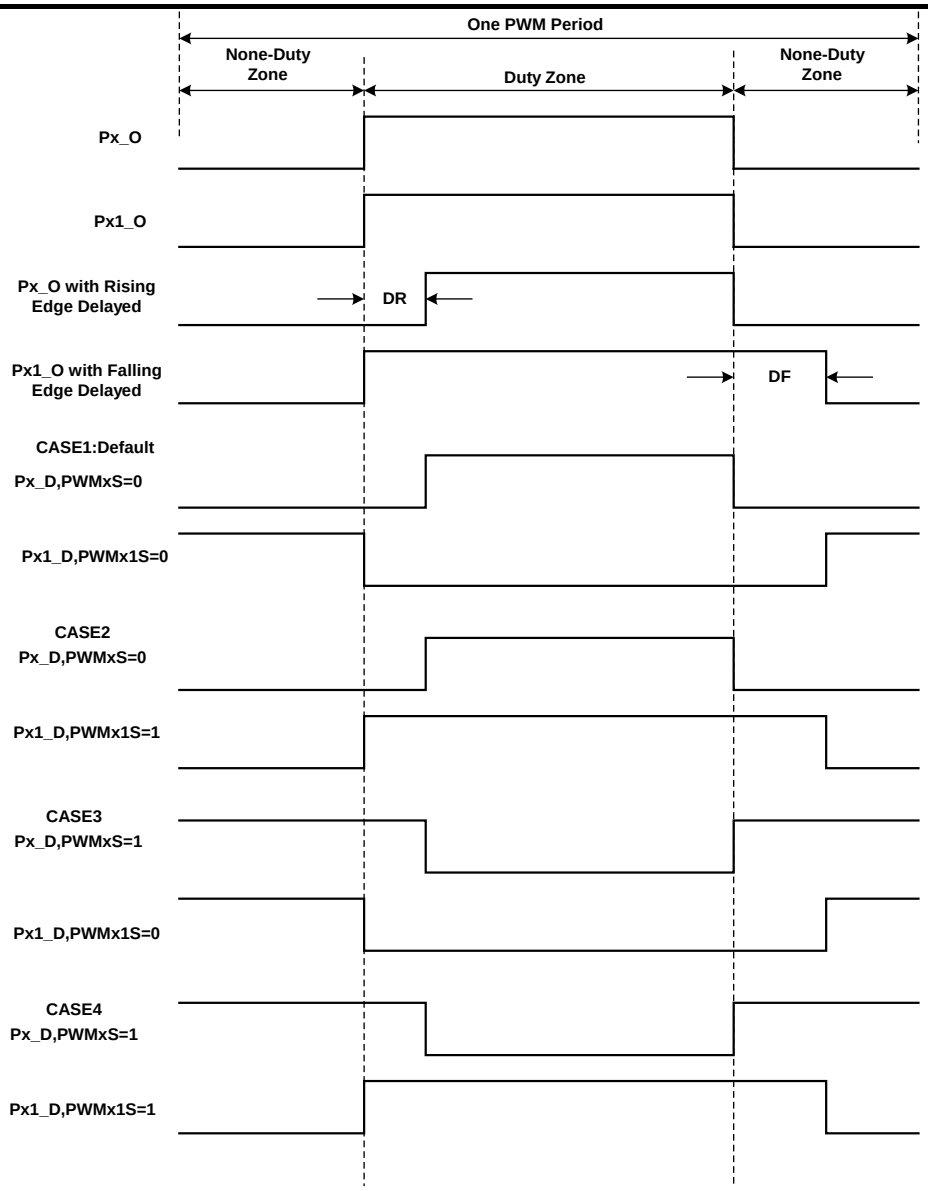


图17-16　　边沿对齐计数模式

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。

17.3.3.6　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17-17所示。

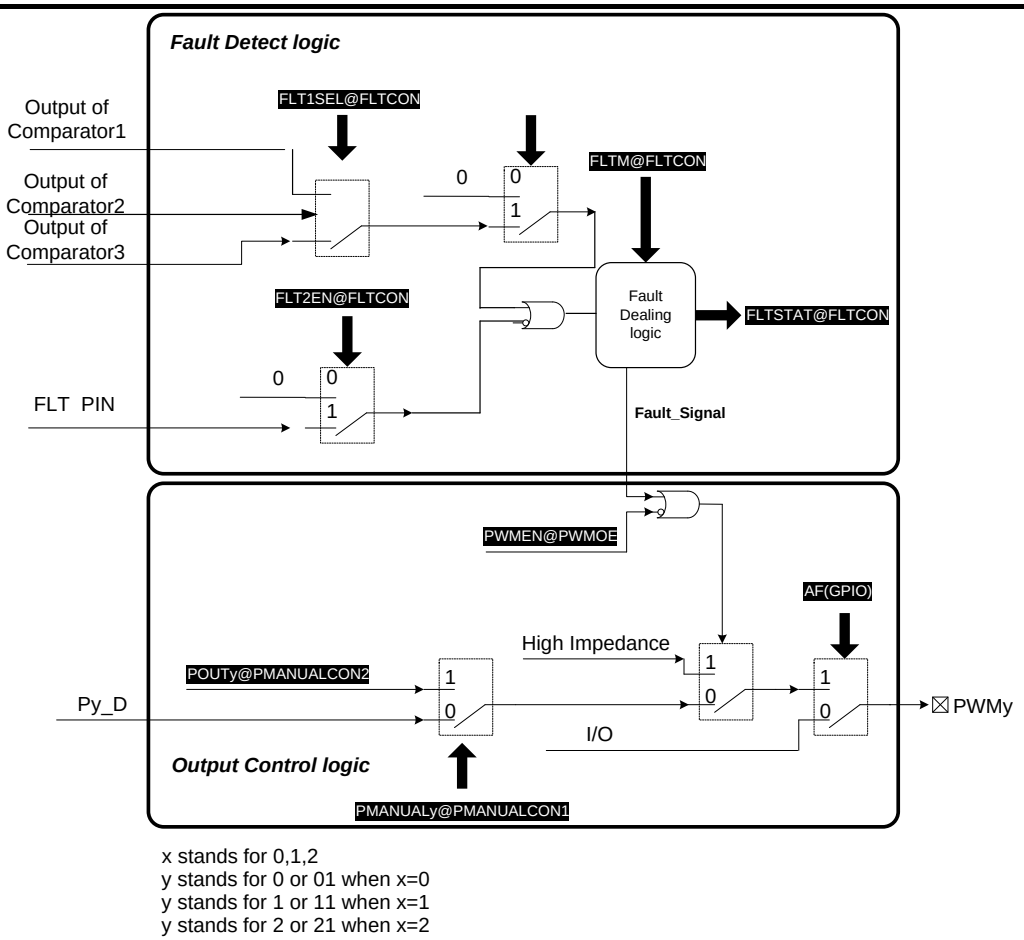


图17-17　　输出控制与故障检测逻辑简图

17.3.4　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FLT1EN位或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17-17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。

17.3.4.1　故障信号输入源的选择

可以选择比较器（1/2/3）的输出或FLT引脚输入作为故障检测输入信号，如图17-17所示。无论故障信号来自比较器1/2/3输出或FLT，高电平为有效故障信号，即比较器输出由低变高时将自动切断PWM的输出。

如图17-17所示，若FLT1EN = 1，则比较器1或者2或者3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），则六路PWM引脚PWMx/PWMx1立即进入无效驱动状态（x = 0 - 2）。同理，若FLT2EN = 1，则FLT引脚输入作为故障检测输入信号，一旦FLT引脚输入变高（FLT2S = 0时），并保持滤波时间，则六路PWM引脚PWMx/PWMx1立即进入无效驱动状态（x = 0 - 2）。如果FLT2EN = 0，则禁止FLT引脚的故障检测功能，FLT引脚作为普通IO使用（但比较器的故障检测功能不受影响）。

当使用PWM模块来控制图17-18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）。

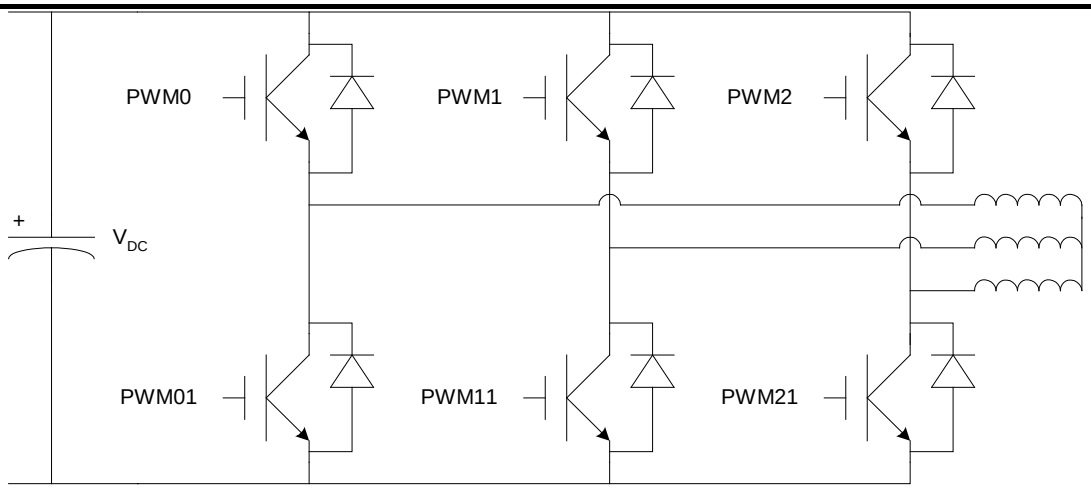


图17-18　　PWM 控制逆变桥示例图

17.3.4.2　　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 = 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，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。

FLT1EN = 1时，可选比较器1、2、3的输出信号作为故障检测输入信号使用，当比较器输出由0变1时触发保护，此时比较器1、2、3的输出滤波时间和算法分别在寄存器CMP1CON、CMP2CON2和CMP2CON3中设置。（详见运算放大器和比较器模块章节）

17.3.4.3　　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FLTCON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FLTCON.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FLTCON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。

17.3.4.4 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。

17.3.5 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。

17.3.5.1 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。

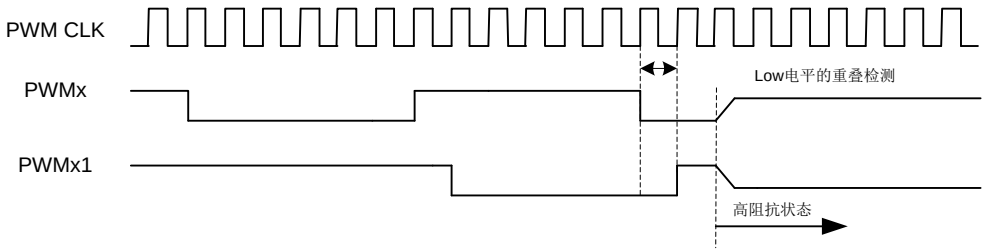


图17-19 PWM 输出电平比较

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。

17.3.5.2 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。

17.3.5.3 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。

17.3.5.4 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。

17.3.6 PWM 输出控制模块

17.3.6.1 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17-17所示。

17.3.6.2 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17-17所示。

更改PMANUALCON1和PMANUALCON2寄存器时，PWMx/PWMx1引脚输出的改变与PCLK2同步，输出状态改变立即生效。

17.3.6.3 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17-17所示。

17.3.7 事件触发功能

PWM 事件可以触发 ADC、寄存器的自动重载、中断和 DMA 的启动。

17.3.7.1 PWM0/1/2 计数器触发 ADC

若寄存器PWMCON中的CMPx =1 （x=0-3），则在PWM计数期间当计数器值与PWMCMPx （x=0-3）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位和ADSTRS[6:0]相应位被允许，将会启动一次序列转换）；CMPx = 2 （x=0-3），则仅在PWM增计数期间当计数器值与PWMCMPx （x=0-3）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位和ADSTRS[6:0]相应位被允许，将会启动一次序列转换）；CMPx = 3（x=0-3），则仅在PWM减计数期间当计数器值与PWMCMPx（x=0-3）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位和ADSTRS[6:0]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。

17.3.7.2 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CMP1/CMP2的重载也受归零信号或周期匹配信号控制。

17.3.7.3 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）。

17.3.7.4 DMA 的启动

能通过各个MCM的中断源启动DMA。但是忽视对应状态标志为“1”时的DMA启动请求，因此为了重新产生DMA启动请求，必须将对应的状态标志置“0”。

注：启动DMA与是否发生中断没有关系，DMA只需要中断源的事件信号进行触发。

17.3.8 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（缓冲完之后），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：

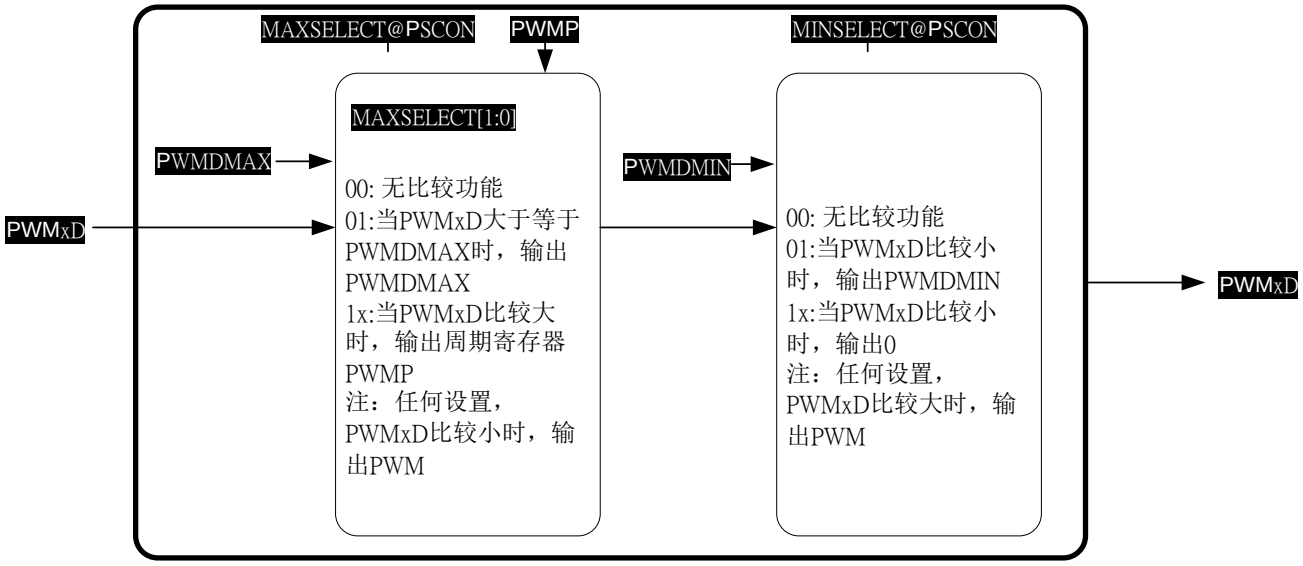


图17-20　　饱和功能图

17.3.9　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，无法实现两相电流值采样。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：

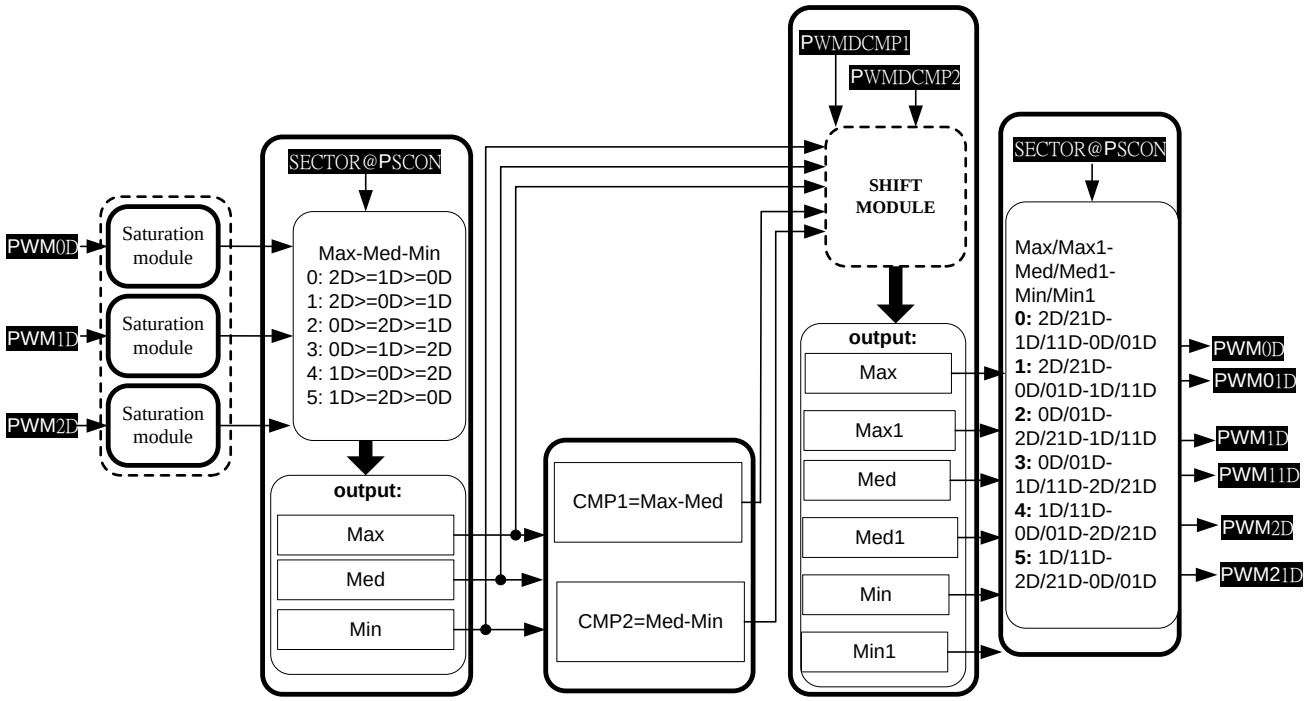


图17-21　　PWM 相移功能图

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、中间值和最小值占空比，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。

17.3.9.2　　象限选择模块

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SECTOR@PSCON来判断PWM三相占空比大小，对应关系如图17-23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。

17.3.9.3 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：

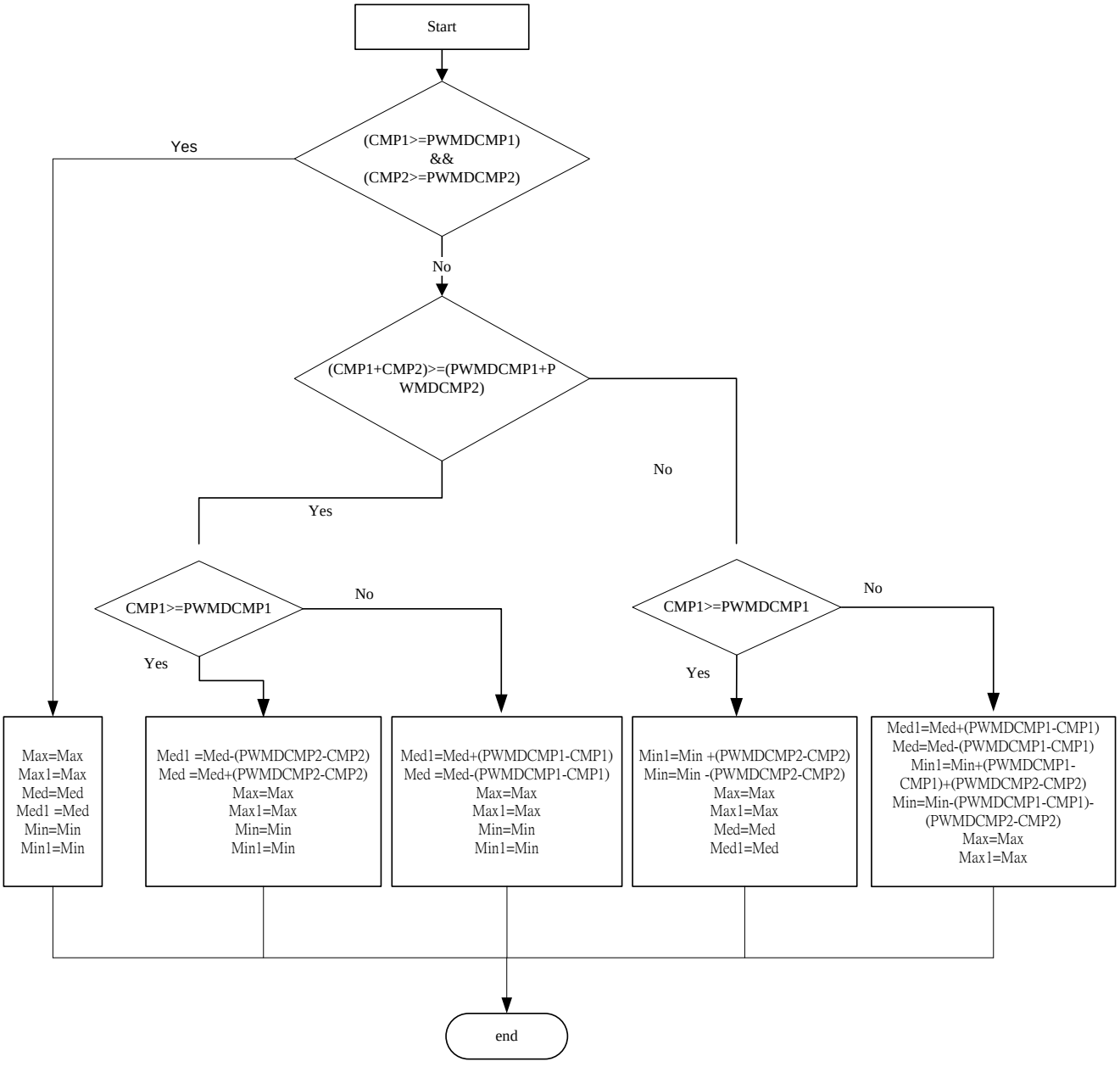


图17-22　　PWM 相移实现图

17.3.9.4 饱和相移应用示例

设定：APB2的时钟PCLK2频率为60Mhz，MCM不分频，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@PSCON=01b，MinSelect@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PDMaxBuffer = 4500，PDMumBuffer = 4200，PDMinBuffer = 500，CMP1BUFFER = 300，CMP2BUFFER = 3800。经过移位模块后，PDMaxBuffer = 4500，PDMax1Buffer = 4500，PDMumBuffer = 4000，PDMum1Buffer = 4400，PDMinBuffer = 500，PDMin1Buffer = 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：

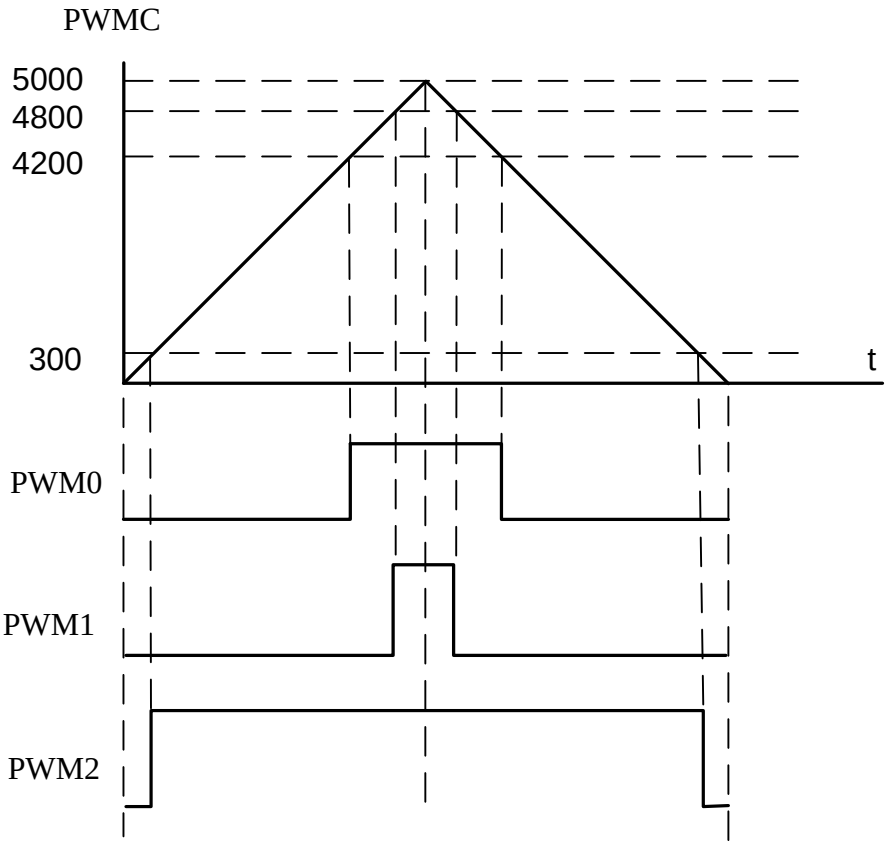


图17-23　PWM 饱和相移前波形

经过饱和相移后波形如下：

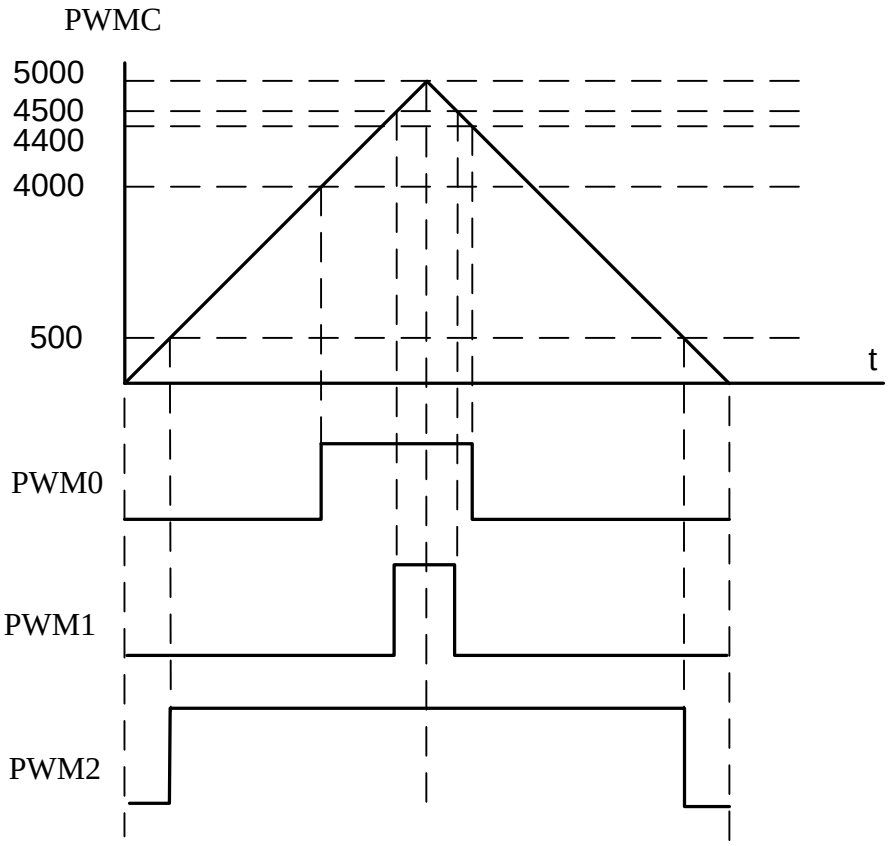


图17-24　　PWM 饱和相移后波形

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。

17.3.10 PWM 逐波限流功能

17.3.10.1　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OLSGx@POSCR(x=0,1,2,01,11,21)来确定。例如OLSG0@POSCR = 1，则当发生逐波限流时，PWM0引脚输出低电平。限流保护时间也通过SCTIME@SCxCON[3:0](x=1,2,3)可以设置。

注：逐波限流功能与故障检测模块可以同时开启。

17.3.10.2　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。SC1保护PWM0，保护时间设定为2400个PLCK2(40us)；SC2保护PWM1，保护时间设定为3600个PLCK2(60us)；SC3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。

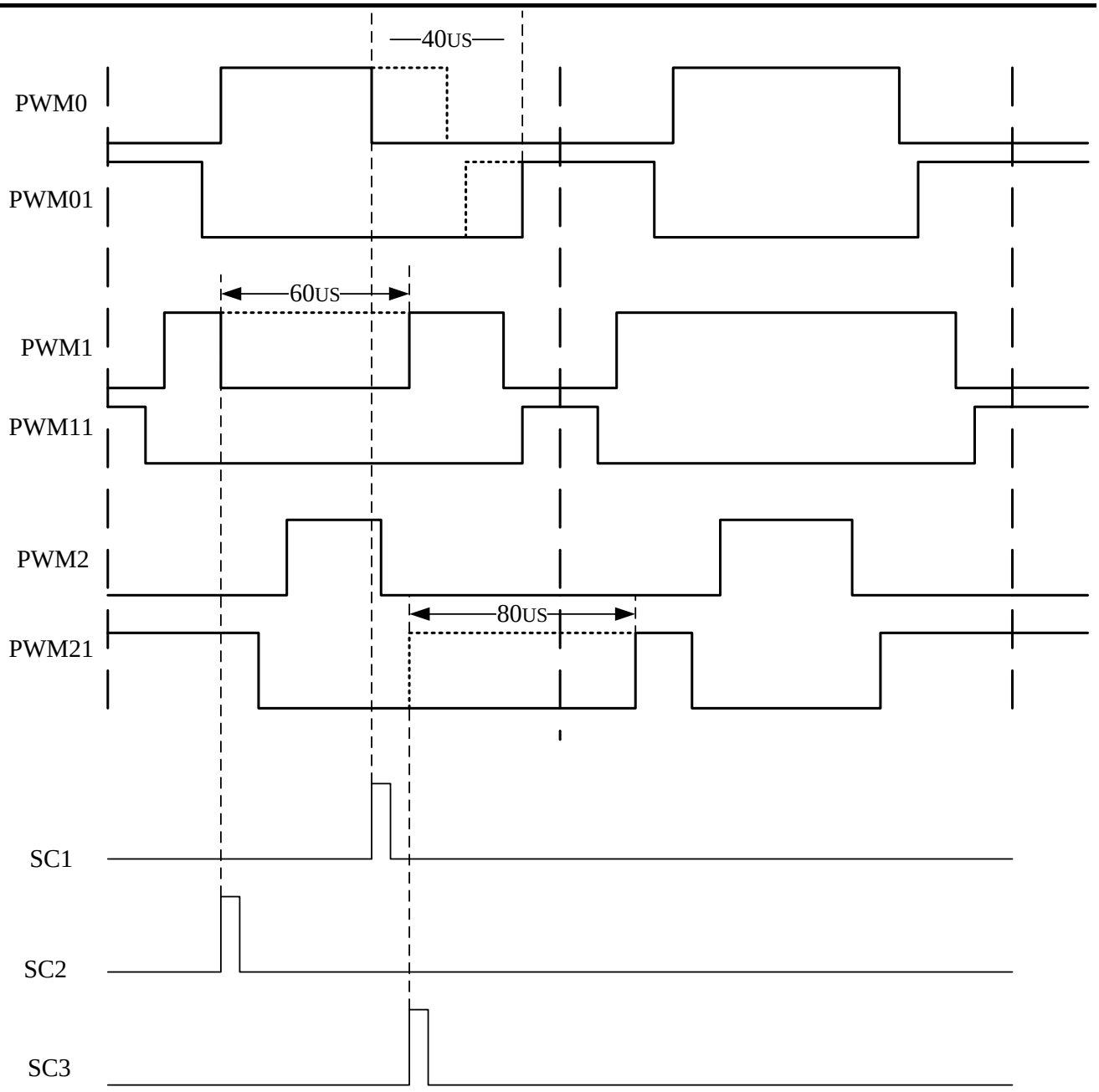


图17-25　　逐波限流应用示例

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。

17.3.11 PWM 寄存器的写保护

17.3.11.1　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冲寄存器，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时才允许软件修改，否则修改无效。

注：寄存器的重载时间点如下：

1、 带BUFFER且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、 带BUFFER且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、 带BUFFER且重载点为归零、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)。

17.3.11.2 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。

17.3.12 注意事项

17.3.12.1 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17-28 - 图17-30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。

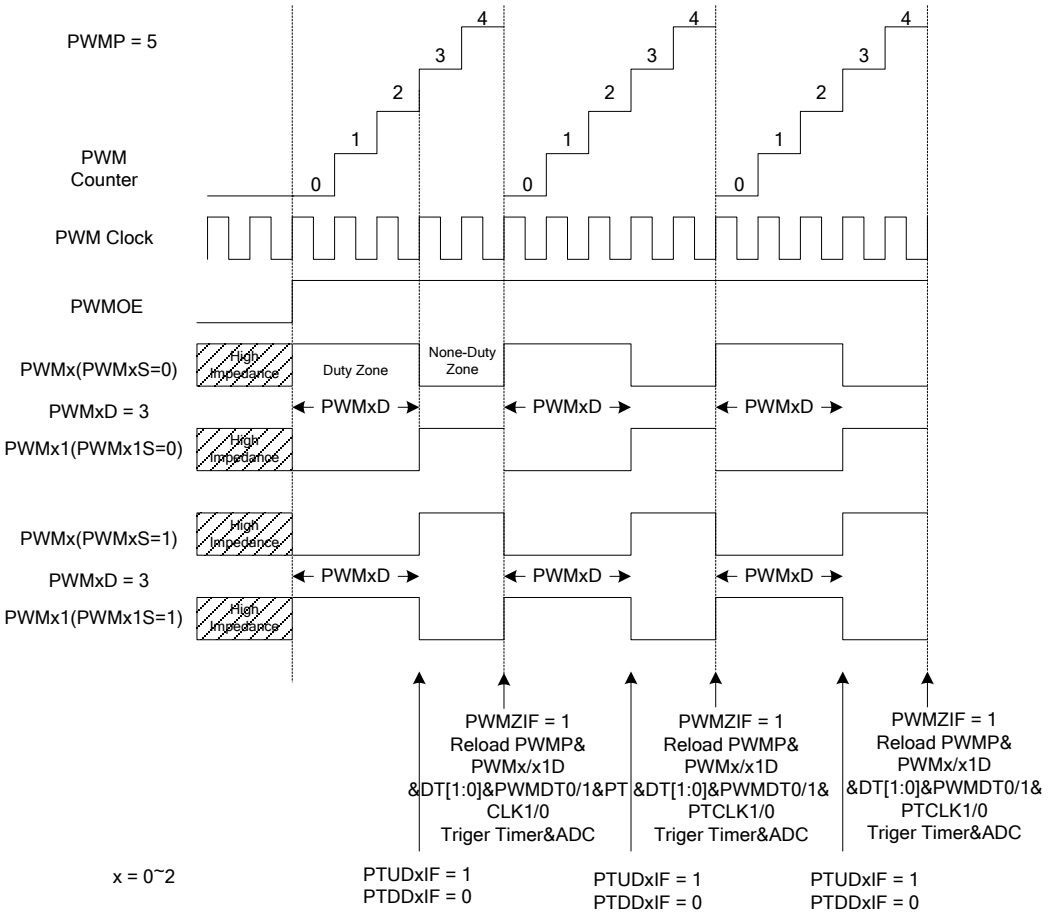


图17-26　　边沿对齐计数、互补输出模式

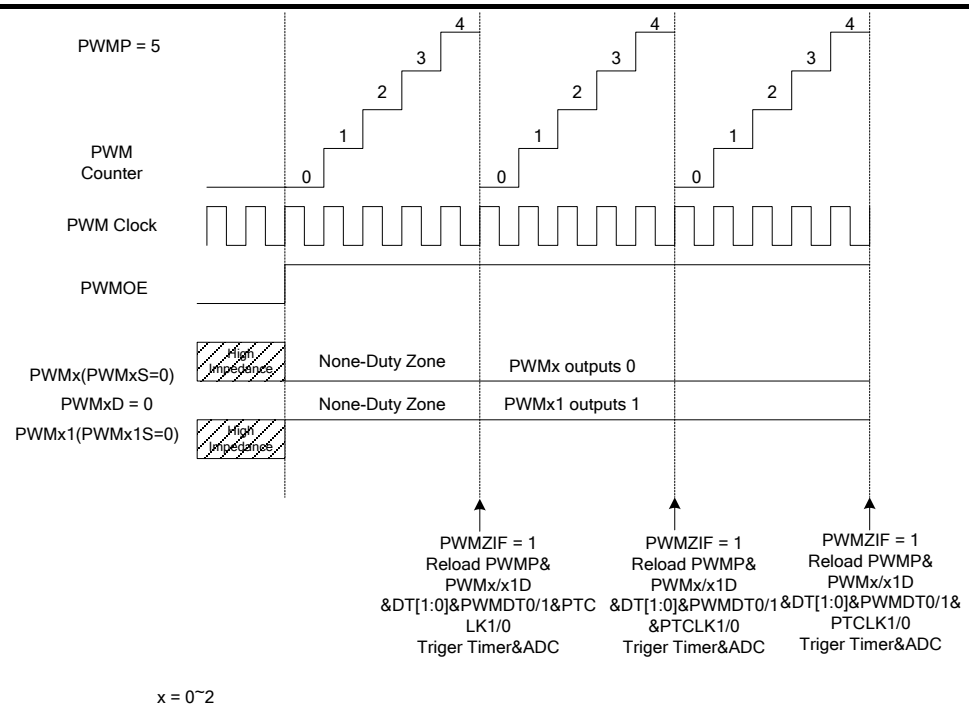


图17-27　　边沿对齐计数、互补输出模式，占空比 = 0

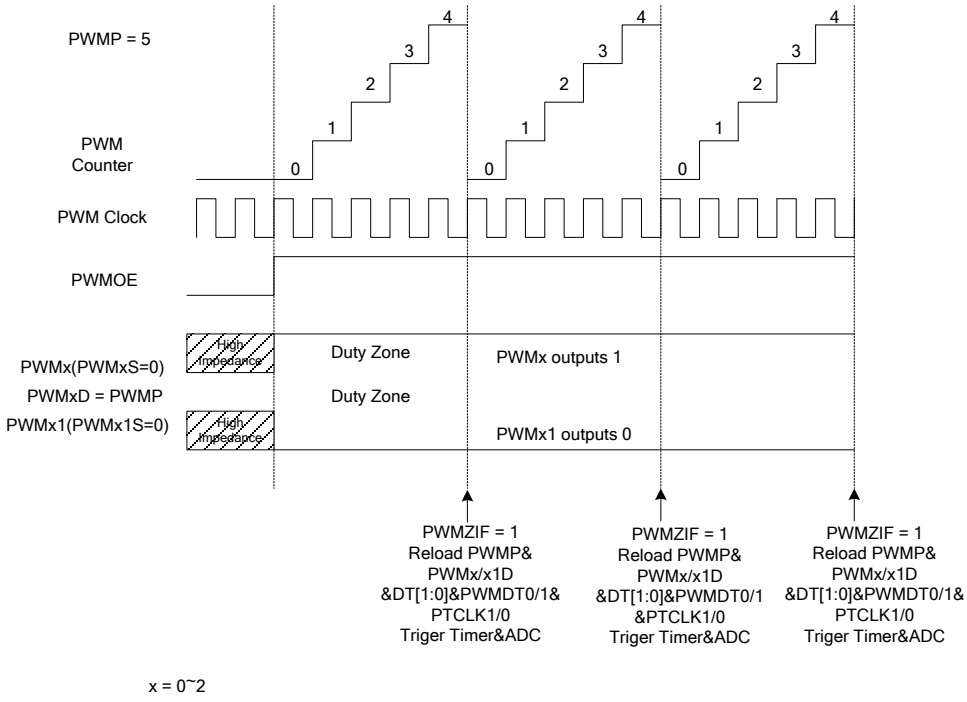


图17-28　　边沿对齐计数、互补输出模式，占空比 = 周期寄存器

17.3.12.2　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17-31 - 图17-33所示。

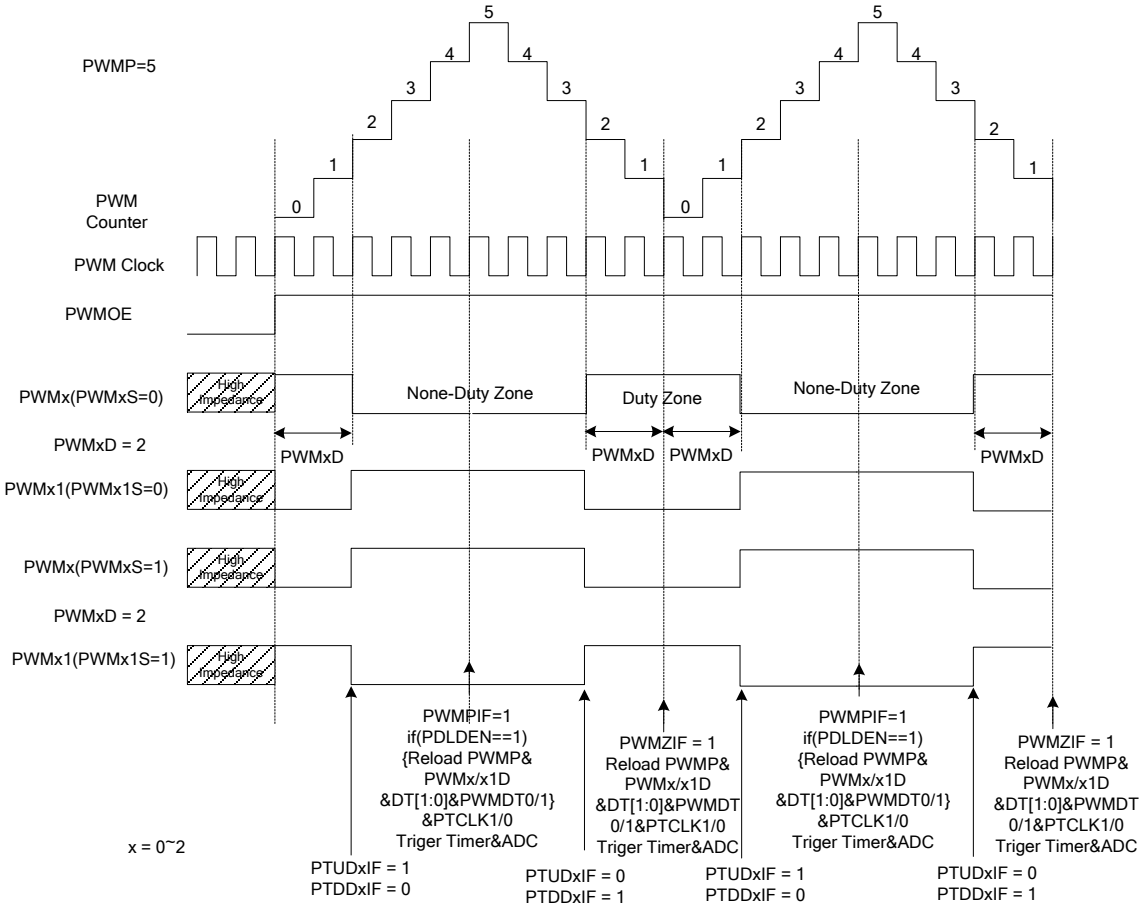


图17-29　　中心对齐计数、互补输出模式

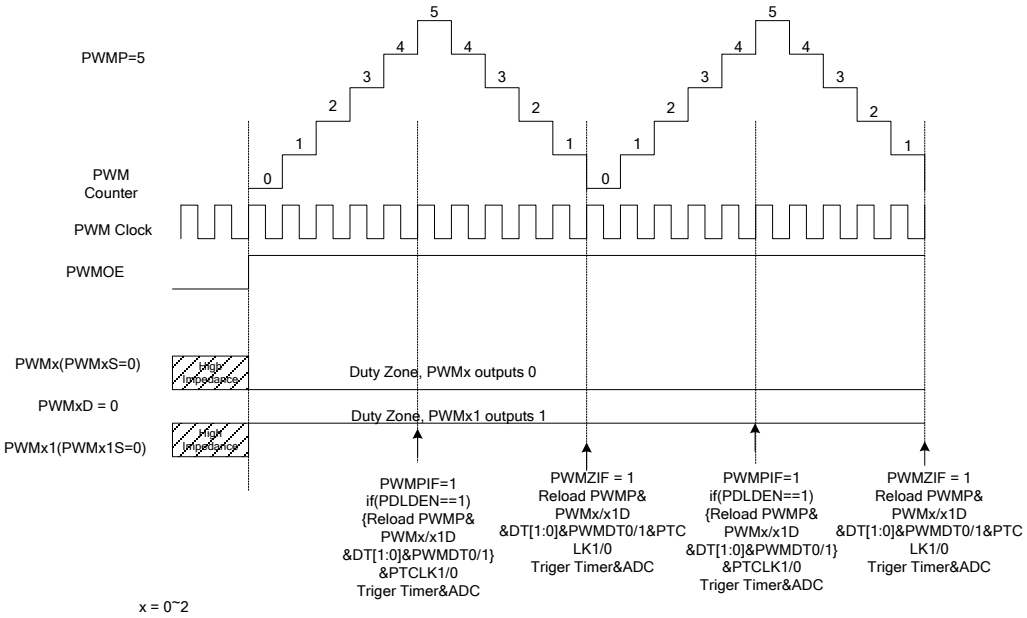


图17-30　　中心对齐计数、互补输出模式、占空比 = 0

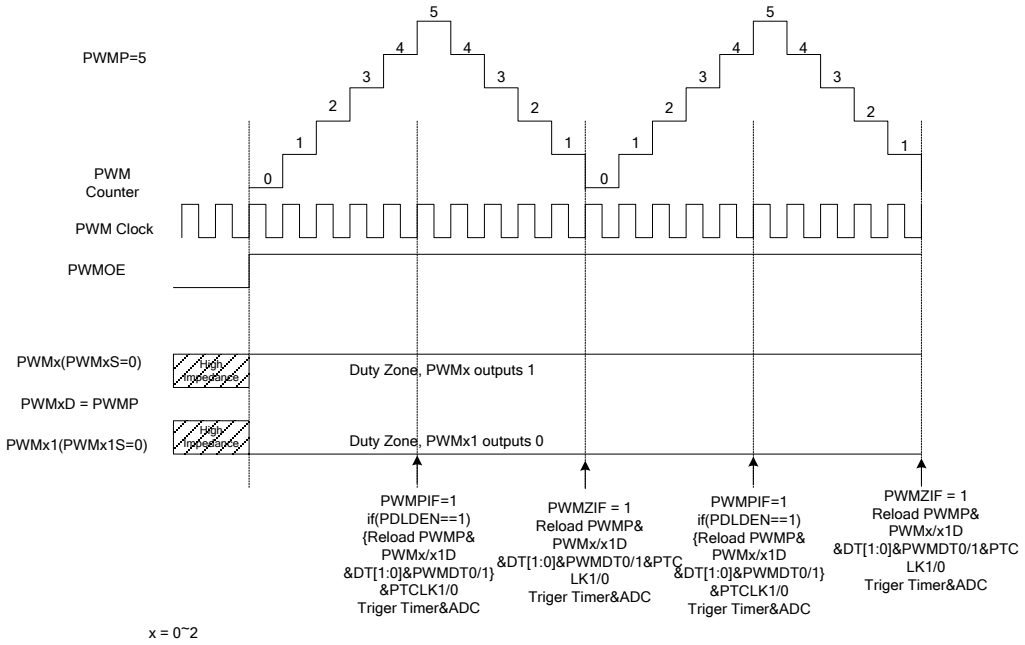


图17-31　　中心对齐计数、互补输出模式、占空比 = 周期

17.3.12.3　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。

17.3.12.4　仿真状态的PWM 输出

SH32F284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。

17.3.12.5　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。

17.3.12.6　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。

17.4 寄存器

MCM 模块寄存器列表 (基地址:0x0x4002 0000)

地址	寄存器名	说明
0x4002 0000	PWMOE	PWM 输出使能寄存器
0x4002 0004	PWMCON1	PWM 模块控制寄存器 1
0x4002 0008	PWMCON2	PWM 模块控制寄存器 2
0x4002 000C	PWMP	PWM 周期寄存器
0x4002 0010	PWMC	PWM 计数寄存器
0x4002 0014	PWMPSQ	PWM 时钟预分频寄存器
0x4002 0018	PWM0D	PWM0 占空比寄存器
0x4002 001C	PWM1D	PWM1 占空比寄存器
0x4002 0020	PWM2D	PWM2 占空比寄存器
0x4002 0024	PWM01D	PWM01 占空比寄存器
0x4002 0028	PWM11D	PWM11 占空比寄存器
0x4002 002C	PWM21D	PWM21 占空比寄存器
0x4002 0030	PWMCMP1	事件触发比较寄存器 1
0x4002 0034	PWMCMP2	事件触发比较寄存器 2
0x4002 0038	PWMCMP3	事件触发比较寄存器 3
0x4002 003C	PWMCMP4	事件触发比较寄存器 4
0x4002 0040	PWMDT00	PWM 通道 0 上升沿死区控制寄存器
0x4002 0044	PWMDT01	PWM 通道 0 下降沿死区控制寄存器
0x4002 0048	PWMDT10	PWM 通道 1 上升沿死区控制寄存器
0x4002 004C	PWMDT11	PWM 通道 1 下降沿死区控制寄存器
0x4002 0050	PWMDT20	PWM 通道 2 上升沿死区控制寄存器
0x4002 0054	PWMDT21	PWM 通道 2 下降沿死区控制寄存器
0x4002 0058	PMANUALCON1	PWM0/1/2 手动输出设置寄存器 1
0x4002 005C	PMANUALCON2	PWM0/1/2 手动输出设置寄存器 2
0x4002 0060	FLTCON	PWM0/1/2 故障检测保护寄存器
0x4002 0064	POSCR	PWM0/1/2 输出电平保护控制寄存器
0x4002 0068	POSTDCR	PWM0/1/2 振荡器停止检测保护控制位
0x4002 006C	PWMDMAEN	PWMDMA 使能控制寄存器
0x4002 0070	PWMINTEN	PWM 中断使能控制寄存器
0x4002 0074	PWMINTF	PWM 中断标志和清除寄存器
0x4002 0078	PWMRLDEN	寄存器修改和重载控制寄存器
0x4002 007C	PSCON	PWM 饱和/相移控制寄存器
0x4002 0080	PWMDMAX	PWM 占空比上限比较寄存器
0x4002 0084	PWMDMIN	PWM 占空比下限比较寄存器
0x4002 0088	PWMDCMP1	PWM 占空比移相最小采样间隔寄存器 1
0x4002 008C	PWMDCMP2	PWM 占空比移相最小采样间隔寄存器 2
0x4002 0090	SC1CON	逐波限流引脚 1 功能选择寄存器
0x4002 0094	SC2CON	逐波限流引脚 2 功能选择寄存器
0x4002 0098	SC3CON	逐波限流引脚 3 功能选择寄存器
0x4002 009C	FLTWEN	寄存器修改和重载保护控制寄存器

17.4.1 PWM 输出使能寄存器 (MCMx_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

17.4.2 PWM 模块控制寄存器 1 (MCMx_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

17.4.3 PWM 模块控制寄存器 2 (MCMx_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
MCMSYNEN	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*		*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	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：无论寄存器如何设置，周期匹配时不会重载 PWM 周期寄存器、后分频、预分频寄存器
11	**ZDLDEN**	**PWM 计数器归零匹配时重载占空比死区寄存器使能位** 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：不管此值是否设定，归 0 匹配时都会自动重载周期寄存器、后分频、预分频寄存器，不受任何寄存器控制
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 （ADC 的 ADON 和 PWMTRGEN 位置‘1’的情况下）
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）（ADC 的 ADON 和 PWMTRGEN 位置‘1’的情况下）
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 （ADC 的 ADON 和 PWMTRGEN 位置‘1’的情况下）
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）（ADC 的 ADON 和 PWMTRGEN 位置‘1’的情况下）

17.4.4 PWM 周期寄存器 (MCMx_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

17.4.5 PWM 计数寄存器 (MCMx_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 在模块停止时被硬件清零一次，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

17.4.6 PWM 时钟预分频寄存器 (MCMx_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(PCLK2/(PSQ[15:0]+1))

17.4.7 PWM0 占空比寄存器 (MCMx_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

17.4.8 PWM01 占空比寄存器 (MCMx_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
PWMn1D[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

17.4.9 事件触发比较寄存器 1 (MCMx_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

17.4.10 PWM 通道 0 上升沿死区控制寄存器 (MCMx_PWMDTn0)(n=0..2)

偏移地址: 0x0040
:0x0048
: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
PWMDTn0[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	**PWMDTn0[15:0]**	**PWM 通道 n 上升沿死区控制寄存器**

17.4.11 PWM 通道 0 下降沿死区控制寄存器 (MCMx_PWMDTn1)(n=0..2)

偏移地址: 0x0044
:0x004C
: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
PWMDTn1[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	**PWMDTn1[15:0]**	**PWM 通道 n 下降沿死区控制寄存器**

17.4.12 PWM0/1/2 手动输出设置寄存器 1 (MCMx_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

17.4.13 PWM0/1/2 手动输出设置寄存器 2 (MCMx_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。

17.4.14 PWM0/1/2 故障检测保护寄存器 (MCMx_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															
-															
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
FOUT1[1:0]		FOUT0[1:0]		保留	FLT1EN	FLT1SEL[1:0]		FLT2DEB[3:0]				FLT2EN	FLT2S	FLTM	FLTSTAT
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 ~ 14	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
13 ~ 12	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
11	保留	-
10	FLT1EN	**故障检测 1 功能控制位（注 1）** 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 1 或比较器 2
9 ~ 8	FLT1SEL[1:0]	**故障检测 1 输入源选择控制位** 00：比较器 1 滤波后的输出作为 PWM 故障检测 1 输入源，高电平有效 01：比较器 2 滤波后的输出作为 PWM 故障检测 1 输入源，高电平有效 10：比较器 3 滤波后的输出作为 PWM 故障检测 1 输入源，高电平有效 11：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。 注 2：滤波说明：用内部时钟采样输入信号，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	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	**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，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	**FLTSTAT**	**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*

注 1：故障检测输入 1 和 2 可同时打开，任一信号有效都会关闭 PWM0/1/2 模块输出。

17.4.15 PWM0/1/2 输出电平保护控制寄存器 (MCMx_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								OLSEN	保留	OLSG21	OLSG11	OLSG01	OLSG2	OLSG1	OLSG0
-								*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

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

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
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	-

17.4.17 PWMDMA 使能控制寄存器 (MCMx_PWMDMAEN)

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															
-															
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								PWMPDE	PWMZDE	PTDD2DE	PTUD2DE	PTDD1DE	PTUD1DE	PTDD0DE	PTUD0DE
-								*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	PWMPDE	PWM 时基周期匹配 DMA 触发使能位（只在中心对齐模式下有效） 0：禁止 PWM 时基周期匹配 DMA 1：允许 PWM 时基周期匹配 DMA
6	PWMZDE	PWM 时基归零 DMA 触发使能位 0：禁止 PWM 时基归零 DMA 1：允许 PWM 时基归零 DMA
5	PTDD2DE	PWM 时基计数器减计数时与占空比寄存器 PWM2D/PWM21D 匹配时 DMA 触发使能位（只在中心对齐模式下有效） 0：禁止 PWM 计数器减计数时与 PWM2D/PWM21D 匹配时触发 DMA 1：允许 PWM 计数器减计数时与 PWM2D/PWM21D 匹配时触发 DMA 注：在中心对称模式下，为 PWM2D；在中心对齐非对称模式下，为 PWM21D
4	PTUD2DE	PWM 时基计数器增计数时与占空比寄存器 PWM2D 匹配时 DMA 触发使能位 0：禁止 PWM 计数器增计数时与 PWM2D 匹配时触发 DMA 1：允许 PWM 计数器增计数时与 PWM2D 匹配时触发 DMA
3	PTDD1DE	PWM 时基计数器减计数时与占空比寄存器 PWM1D/PWM11D 匹配时 DMA 触发使能位（只在中心对齐模式下有效）

		0：禁止 PWM 计数器减计数时与 PWM1D/PWM11D 匹配时触发 DMA 1：允许 PWM 计数器减计数时与 PWM1D/PWM11D 匹配时触发 DMA 注：在中心对称模式下，为 PWM1D；在中心对齐非对称模式下，为 PWM11D
2	**PTUD1DE**	**PWM 时基计数器增计数时与占空比寄存器 PWM1D 匹配时 DMA 触发使能位** 0：禁止 PWM 计数器增计数时与 PWM1D 匹配时触发 DMA 1：允许 PWM 计数器增计数时与 PWM1D 匹配时触发 DMA
1	**PTDD0DE**	**PWM 时基计数器减计数时与占空比寄存器 PWM0D/PWM01D 匹配时 DMA 触发使能位（只在中心对齐模式下有效）** 0：禁止 PWM 计数器减计数时与 PWM0D/PWM01D 匹配时触发 DMA 1：允许 PWM 计数器减计数时与 PWM0D/PWM01D 匹配时触发 DMA 注：在中心对称模式下，为 PWM0D；在中心对齐非对称模式下，为 PWM01D
0	**PTUD0DE**	**PWM 时基计数器增计数时与占空比寄存器 PWM0D 匹配时 DMA 触发使能位** 0：禁止 PWM 计数器增计数时与 PWM0D 匹配时触发 DMA 1：允许 PWM 计数器增计数时与 PWM0D 匹配时触发 DMA

17.4.18 PWM 中断使能控制寄存器 (MCMx_PWMINTEN)

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					OSTDIE	OIE	FLTIE	PWMPIE	PWMZIE	PTDD2IE	PTUD2IE	PTDD1IE	PTUD1IE	PTDD0IE	PTUD0IE
-					*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 ~ 11	保留	-
10	**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
9	**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
8	**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
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

17.4.19 PWM 中断标志和清除寄存器 (MCMx_PWMINTF)

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		SC3STATC	SC2STATC	SC1STATC	OSTDFC	OSFC	FLTIFC	PWMPIFC	PWMZIFC	PTDD2IFC	PTUD2IFC	PTDD1IFC	PTUD1IFC	PTDD0IFC	PTUD0IFC
-		*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
保留		SC3STAT	SC2STAT	SC1STAT	OSTDF	OSF	FLTIF	PWMPIF	PWMZIF	PTDD2IF	PTUD2IF	PTDD1IF	PTUD1IF	PTDD0IF	PTUD0IF
-		*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 ~ 30	保留	-
29	**SC3STATC**	**SC3 故障检测标志位清除位** 0：无效 1：清除
28	**SC2STATC**	**SC2 故障检测标志位清除位** 0：无效 1：清除
27	**SC1STATC**	**SC1 故障检测标志位清除位** 0：无效 1：清除
26	**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
25	**OSFC**	**输出短路标志位清除位** 0：无效 1：清除
24	**FLTIFC**	**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 ~ 14	**保留**	-
13	**SC3STAT**	**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
12	**SC2STAT**	**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
11	**SC1STAT**	**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
10	**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
9	**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）
8	**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
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 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 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

		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 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 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 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。

17.4.20 寄存器修改和重载控制寄存器 (MCMx_PWMRLDEN)

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
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：允许模块带缓冲的寄存器的重载 *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、POSTDCR 和 FLTWEN 外，其它寄存器的修改只有当 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冲寄存器，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。*

17.4.21 PWM 饱和/相移控制寄存器 (MCMx_PSCON)

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							STATRUN	MINSELECT[1:0]		MAXSELECT[1:0]		SECTOR[2:0]			SHIFTRUN
-							RW1s	RW		RW		RW			RW1s
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	**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 1：共 6 个占空比寄存器 注 2：软件置 1，运算完成后硬件清零 注 3：硬件单个 MCM 时钟周期完成
7 ~ 6	**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
5 ~ 4	**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
3 ~ 1	**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 MUM 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 注意：设定 110 和 111 时无效
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 1：软件置 1，运算完成后硬件清零 注 2：硬件单个 MCM 时钟周期完成

17.4.22 PWM 占空比上限比较寄存器 (MCMx_PWMDMAX)

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
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

17.4.23 PWM 占空比下限比较寄存器 (MCMx_PWMDMIN)

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
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

17.4.24 PWM 占空比移相最小采样间隔寄存器 1 (MCMx_PWMDCMPn)(n=1..2)

偏移地址: 0x0088
:0x008C
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

17.4.25 逐波限流引脚 1 功能选择寄存器 (MCMx_SCnCON)(n=1..2)

偏移地址: 0x0090
:0x0094
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	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 ~ 13	保留	-
12	SCEN	**故障检测 1 功能控制位（注 1）** 0：SCn 引脚逐波限流功能禁止 1：SCn 引脚逐波限流功能打开
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 0001：30*Pclk2 0010：60*Pclk2 0011：90*Pclk2 0100：120*Pclk2 0101：180*Pclk2 0110：240*Pclk2 0111：360*Pclk2 1000：480*Pclk2 1001：600*Pclk2 1010：720*Pclk2 1011：840*Pclk2 1100：960*Pclk2 1101：1200*Pclk2 1110：1440*Pclk2 1111：1920*Pclk2 注 1：上述滤波常数时间不是精确值，仅供参考。 注 2：滤波说明：用内部时钟采样输入信号，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 ~ 0	SCTIME[3:0]	**保护时间选择位：** 0000：600*Pclk2 0001：900*Pclk2 0010：1200*Pclk2 0011：1500*Pclk2 0100：1800*Pclk2 0101：2100*Pclk2 0110：2400*Pclk2 0111：3000*Pclk2 1000：3600*Pclk2 1001：4800*Pclk2 1010：6000*Pclk2 1011：12000*Pclk2 1100：24000*Pclk2 1101：36000*Pclk2

		1110：48000*Pclk2 1111：60000*Pclk2

17.4.26 逐波限流引脚 3 功能选择寄存器 (MCMx_SC3CON)

偏移地址: 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	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 ~ 13	保留	-
12	SCEN	**故障检测 1 功能控制位（注 1）** 0：SC3 引脚逐波限流功能禁止 1：SC3 引脚逐波限流功能打开
11	SCS	**SC3 有效电平选择位** 0：逐波限流 SC3 输入高电平有效 1：逐波限流 SC3 输入低电平有效
10	SCPWM2EN	**PWM2 保护使能位：** 0：SC3 不保护 PWM2 1：SC3 保护 PWM2
9	SCPWM1EN	**PWM1 保护使能位：** 0：SC3 不保护 PWM1 1：SC3 保护 PWM1
8	SCPWM0EN	**PWM0 保护使能位：** 0：SC3 不保护 PWM0 1：SC3 保护 PWM0
7 ~ 4	SCDEB[3:0]	**逐波限流 SC3 输入滤波参数选择** 0000：0 0001：30*Pclk2 0010：60*Pclk2 0011：90*Pclk2 0100：120*Pclk2 0101：180*Pclk2 0110：240*Pclk2 0111：360*Pclk2 1000：480*Pclk2 1001：600*Pclk2 1010：720*Pclk2 1011：840*Pclk2 1100：960*Pclk2 1101：1200*Pclk2 1110：1440*Pclk2 1111：1920*Pclk2 注 1：上述滤波常数时间不是精确值，仅供参考。 注 2：滤波说明：用内部时钟采样输入信号，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 ~ 0	**SCTIME[3:0]**	**保护时间选择位：** 0000：600*Pclk2 0001：900*Pclk2 0010：1200*Pclk2 0011：1500*Pclk2 0100：1800*Pclk2 0101：2100*Pclk2 0110：2400*Pclk2 0111：3000*Pclk2 1000：3600*Pclk2 1001：4800*Pclk2 1010：6000*Pclk2 1011：12000*Pclk2 1100：24000*Pclk2 1101：36000*Pclk2 1110：48000*Pclk2 1111：60000*Pclk2

17.4.27 寄存器修改和重载保护控制寄存器 (MCMx_FLTWEN)

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
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，这些寄存器的修改只有当 FLTWEN = 0x33CC 时才允许软件修改，否则修改无效。

18. 通用 PWM 定时器（General PWM Timer）

18.1 简介

SH32F284内置三个高级16位定时器（GPT0～GPT2），两个16位定时器可级联成一个32位定时器，每个定时器最大能以AHB时钟（HCLK）运行。可实现3个定时器的同步启动和同步清除。GPT带输入捕捉、输出比较、PWM输出功能，每个GPT可生成2路（或1对）互补输出的PWM信号，3个GPT可实现三相电机驱动。

18.2 主要特性

- 3个16位定时器
- 每个定时器能够进行递增计数（锯齿形）、递减计数（锯齿形）、递增/ 递减计数（三角形）
- 每个定时器可以选择独立的时钟源
- 每个定时器有2个输入/ 输出引脚
- 每个定时器有2 个用于输出比较/ 输入捕捉的寄存器
- 3个缓冲寄存器，分别对应各通道的2 组输出比较/ 输入捕捉寄存器
- 在输出比较运行时，能分别在波峰/ 波谷进行缓冲运行，并且生成左右不对称的PWM 波形
- 同步运行模式（支持同时或者以任意的时序进行相位移位）
- 能在PWM 运行时生成死区时间
- 组合3 个定时器，能生成带死区时间的三相PWM 波形
- 能通过外部/ 内部触发开始、清除或者停止计数
- 内部触发源有软件和比较匹配
- 能够级联运行
- 输出I/O口可灵活配置

18.3 功能描述

○：能

—：不能

项目		GPT0	GPT1	GPT2
计数时钟（GPTCLK）		HCLK，GPT1	HCLK	HCLK
时钟源分频		GPTCLK/(PSQ+1)	GPTCLK/(PSQ+1)	GPTCLK/(PSQ+1)
输出比较/ 输入捕捉寄存器（GTCCR）		GTCCRA GTCCRB	GTCCRA GTCCRB	GTCCRA GTCCRB
比较/缓冲寄存器		GTCCRC GTCCRD GTCCRE GTCCRF	GTCCRC GTCCRD GTCCRE GTCCRF	GTCCRC GTCCRD GTCCRE GTCCRF
周期设定寄存器		GTPR	GTPR	GTPR
周期设定缓冲寄存器		GTPBR GTPDBR	GTPBR GTPDBR	GTPBR GTPDBR
输出引脚		GTIOC0A GTIOC0B	GTIOC1A GTIOC1B	GTIOC2A GTIOC2B
输入引脚		GTIOC0A GTIOC0B	GTIOC1A GTIOC1B	GTIOC2A GTIOC2B
比较匹配输出	低电平输出	○	○	○
	高电平输出	○	○	○
	交替输出	○	○	○
输入捕捉功能		○	○	○
同步运行		○	○	○
相位移位开始		○	○	○
死区时间自动附加功能		○	○	○
PWM模式		○	○	○
缓冲运行		○	○	○
单触发运行		○	○	○
A/D转换开始触发		GTADTRA、GTADTRB的比较匹配	GTADTRA、GTADTRB的比较匹配	GTADTRA、GTADTRB的比较匹配

中断源	6个中断源： • GTCCRA比较匹配/输入捕捉（GTCIA0） • GTCCRB比较匹配/输入捕捉（GTCIB0） • 死区时间错误（GTICIC0） • 电平异常保护标志（GTOSC0） • GTCNT上溢（GTPR比较匹配）/GTCNT下溢（GTCIV0）	6个中断源： • GTCCRA比较匹配/输入捕捉（GTCIA1） • GTCCRB比较匹配/输入捕捉（GTCIB1） •死区时间错误（GTICIC1） • 电平异常保护标志（GTOSC1） •GTCNT上溢（GTPR比较匹配）/GTCNT下溢（GTCIV1）	6个中断源： • GTCCRA比较匹配/输入捕捉（GTCIA2） • GTCCRB比较匹配/输入捕捉（GTCIB2） •死区时间错误（GTICIC2） • 电平异常保护标志（GTOSC2） •GTCNT上溢（GTPR比较匹配）/GTCNT下溢（GTCIV2）
中断减少功能	减少GTCNT上溢（GTPR比较匹配）/GTCNT下溢（GTCIV0）中断（有联动其它中断和A/D转换开始请求的功能）	减少GTCNT上溢（GTPR比较匹配）/GTCNT下溢（GTCIV1）中断（有联动其它中断和A/D转换开始请求的功能）	减少GTCNT上溢（GTPR比较匹配）/GTCNT下溢（GTCIV2）中断（有联动其它中断和A/D转换开始请求的功能）

GPT的输入/输出引脚：

通道	引脚名	输入/输出	功能
GPT	GTETRG	输入	外部触发的输入引脚
	GTPOE	输入	保护GPT输出，如果发生异常，GPT输出高阻态
GPT0	GTIOC0A	输入/输出	GTCCRA的输入捕捉的输入引脚/输出比较的输出引脚/PWM输出引脚
	GTIOC0B	输入/输出	GTCCRB的输入捕捉的输入引脚/输出比较的输出引脚/PWM输出引脚
GPT1	GTIOC1A	输入/输出	GTCCRA的输入捕捉的输入引脚/输出比较的输出引脚/PWM输出引脚
	GTIOC1B	输入/输出	GTCCRB的输入捕捉的输入引脚/输出比较的输出引脚/PWM输出引脚
GPT2	GTIOC2A	输入/输出	GTCCRA的输入捕捉的输入引脚/输出比较的输出引脚/PWM输出引脚
	GTIOC2B	输入/输出	GTCCRB的输入捕捉的输入引脚/输出比较的输出引脚/PWM输出引脚

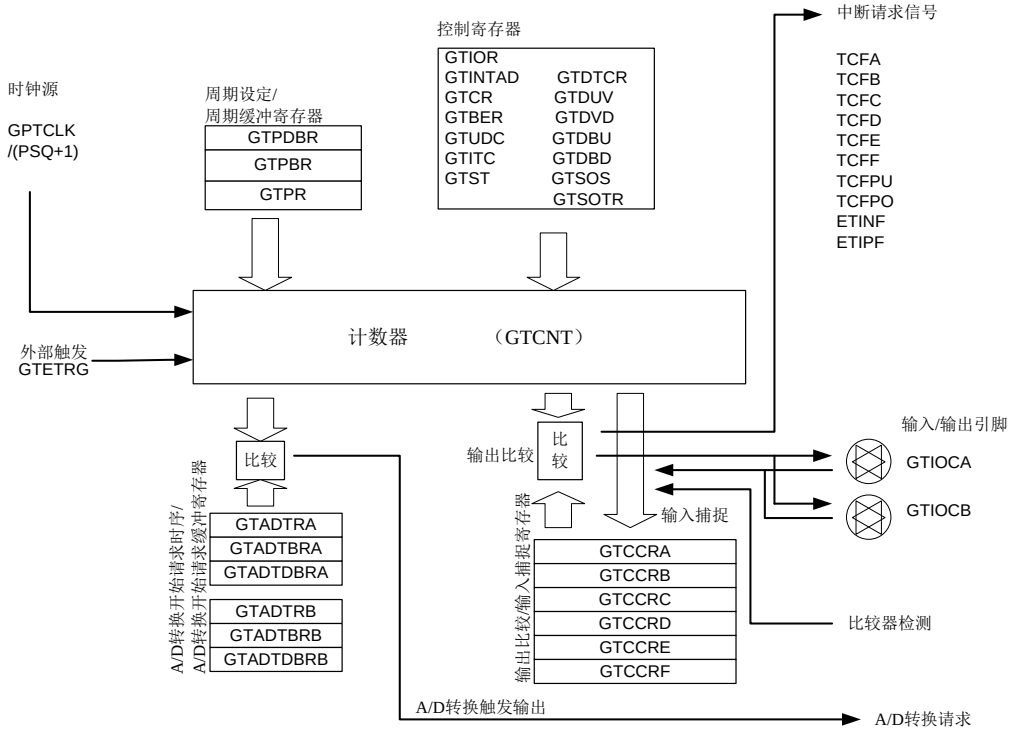


图18-1 GPT 模块简图

18.3.1 定时器的运行

18.3.1.1 周期递增定时器运行

如果将GTSTR寄存器的对应CST位置“1”，各通道的计数器就开始递增计数。当GTCNT计数器的值和GTPR寄存器的值相同（上溢）时，TCFPO@GPTx_GTST标志置“1”。此时，如果GTINTPR[0]@GPTx_ GTINTAD 位为“1”，就请求GTCIV中断。在发生上溢后，GTCNT计数器从“0000h”开始继续递增计数。在此模式下的工作如图18-2所示。

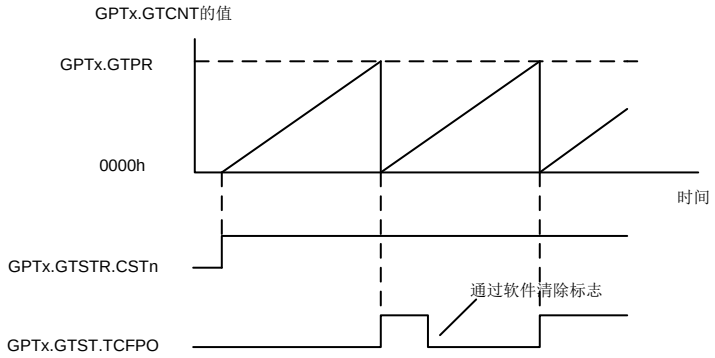


图18-2 周期递增运行示例

18.3.1.2 周期递减定时器运行

各模块的定时器能通过设定GTUDC寄存器进行递减计数。如果GTCNT计数器的值为“0”（下溢），TCFPU@GPTx_GTST标志被置为“1”。此时，如果GTINTPR[1]@GPTx_GTINTAD位为“1”，就请求GTCIV中断。在发生下溢后，GTCNT计数器从GTPR寄存器的值开始继续递减计数。在此模式下的工作如图18-3所示。

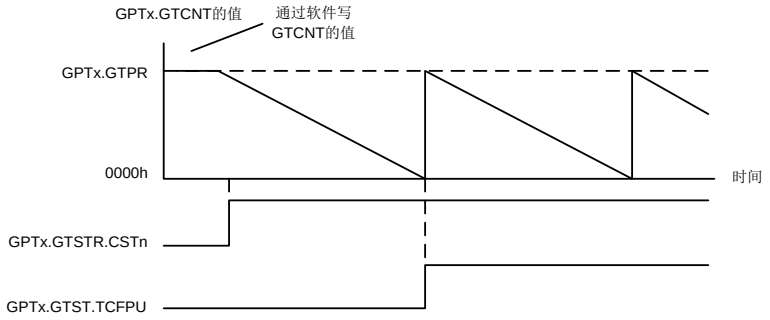


图18-3　　周期递减运行示例

18.3.2　比较匹配输出

如果GPTx_GTCNT计数器的值与GPTx_GTCCRA寄存器、GPTx_GTCCRB寄存器的值相同，就能分别从GTIOCxA引脚和GTIOCxB引脚输出Low电平、High电平、或者进行交替输出。如果GPTx_GTCNT计数器的值到达周期结束时刻（周期寄存器GPTx_GTPR），也能从GTIOCxA引脚和GTIOCxB引脚输出Low电平、High电平或者进行交替输出。

周期结束时刻如下所示。

- 递增计数的情况：当GPTx_GTCNT=GPTx_GTPR时（上溢）
- 递减计数的情况：当GPTx_GTCNT=0时（下溢）
- 递增/递减计数的情况：当GPTx_GTCNT=0时

18.3.2.1　LOW 电平/HIGH 电平输出

通过与GTCCRA寄存器、GTCCRB寄存器的比较匹配进行的Low电平输出/High电平输出的运行例子如图18-4所示。

在此例子中假设GPTx进行递增计数，并且通过与GPTx_GTCCRA寄存器的比较匹配从GTIOCxA引脚输出High电平，通过与GPTx_GTCCRB寄存器的比较匹配从GTIOCxB引脚输出Low电平。如果设定的电平和引脚的电平相同，引脚的电平就不变。

设置运行举例：

GTIOA[5:0]@GPTx_GTIOR的设定：初始输出为“0”、通过比较匹配变为“1”、在周期结束时保持输出

GTIOB[5:0]@GPTx_GTIOR的设定：初始输出为“1”、通过比较匹配变为“0”、在周期结束时保持输出

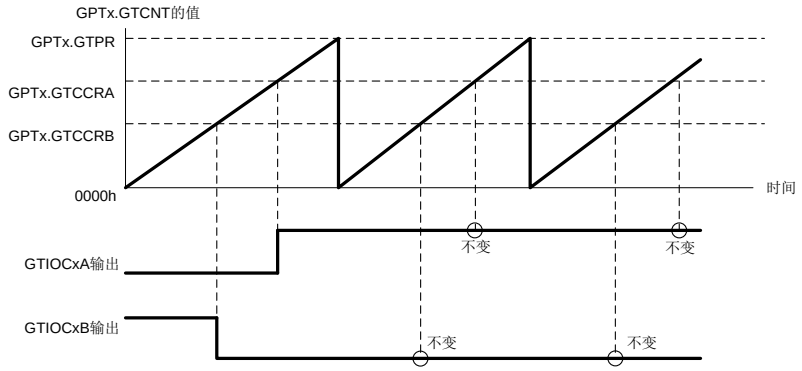


图18-4　　LOW 电平/HIGH 电平输出设置运行举例

18.3.2.2　交替输出

通过与GTCCRA寄存器、GTCCRB寄存器的比较匹配进行交替输出运行例子如图18-5、图18-6所示。

在图18-5的例子中假设GPTx进行递增计数，并且通过与GPTx_GTCCRA寄存器、GPTx_GTCCRB寄存器的比较匹配从GTIOCxA引脚和GTIOCxB引脚进行交替输出。

在图18-6的例子中假设GPTx进行递增计数，并且通过与GPTx_GTCCRA寄存器的比较匹配从GTIOCxA引脚进行交替输出，在周期结束时从GTIOCxB引脚进行交替输出。

设置运行举例1：

GTIOA[5:0]@GPTx_GTIOR的设定：初始输出为”1”、通过比较匹配进行交替输出、在周期结束时保持输出

GTIOB[5:0]@GPTx_GTIOR的设定：初始输出为”0”、通过比较匹配进行交替输出、在周期结束时保持输出

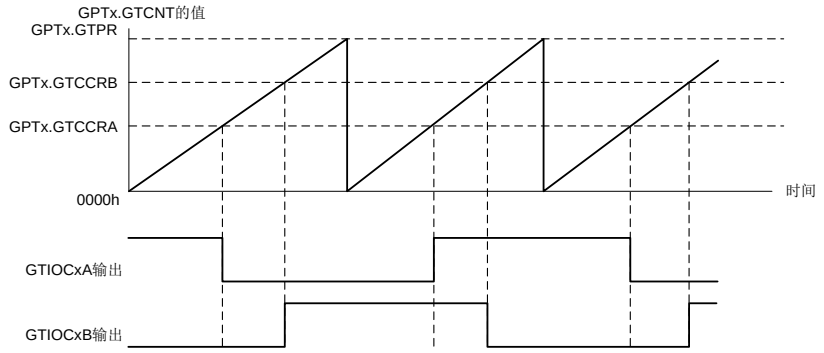


图18-5　　交替输出设置运行举例 1

设置运行举例2：

GTIOA[5:0]@GPTx_GTIOR的设定：初始输出为’1’、通过比较匹配进行交替输出、在周期结束时保持输出

GTIOB[5:0]@GPTx_GTIOR的设定：初始输出为“0”、通过比较匹配保持输出、在周期结束时进行交替输出

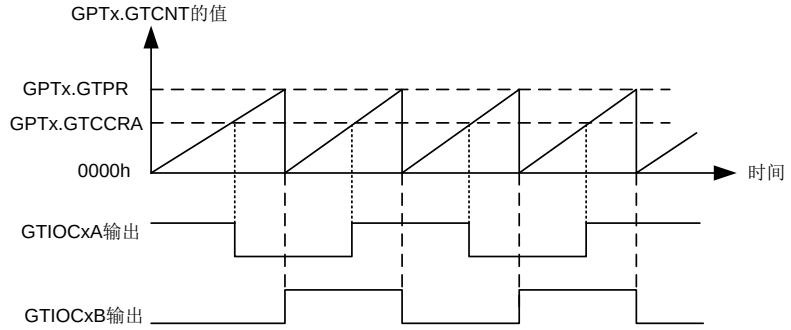


图18-6　　交替输出设置运行举例 2

18.3.3　输入捕捉功能

能检测GTIOCxA引脚和GTIOCxB引脚的边沿，并且将GPTx_.GTCNT计数器的值分别传送到GPTx_.GTCCRA寄存器和GPTx_GTCCRB寄存器。从上升沿、下降沿和双边沿中选择检测边沿。GPT0~2的输入捕捉引脚可以选择GTIOCxA/B，也可以选择串口通信的RXD1/RXD2/RXD3。选择方式见寄存器的表1~表4。

输入捕捉功能的运行例子如图18-7所示。

在此例子中假设GPTx进行递增计数，并且在GTIOCxA引脚的双边沿进行输入捕捉，在GTIOCxB引脚的上升沿进行输入捕捉。

设置运行举例：

GTIOA[5:0]@GPTx_GTIOR的设定：在双边沿进行输入捕捉

GTIOB[5:0]@GPTx_GTIOR的设定：在上升沿进行输入捕捉

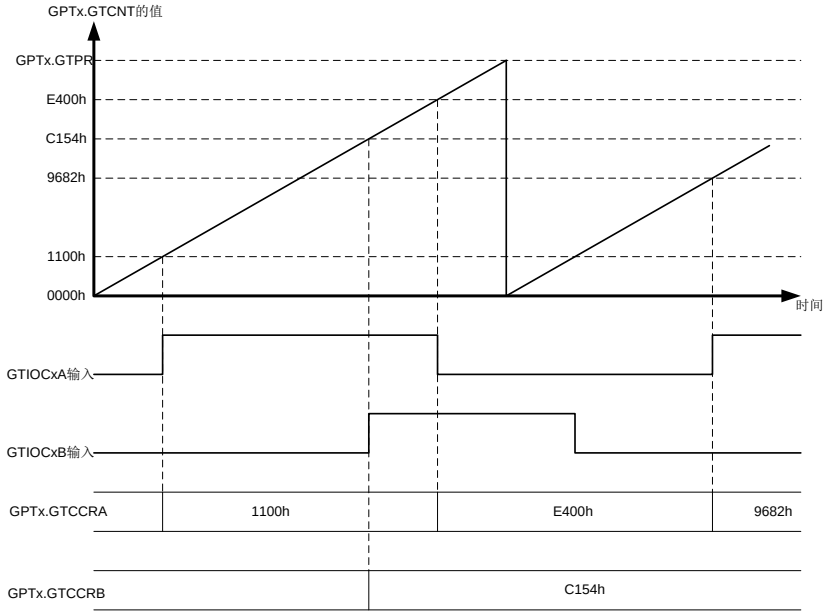


图18-7　　输入捕捉的设置运行举例

18.3.4　输入信号滤波功能

6个定时器输入引脚、1个外部触发引脚和1个保护引脚的输入信号都可以通过内部滤波电路保证检测的可靠性。

GTDEB[3:0]位段来调整对此信号的滤波时间。GTDEB[3:0] = 0时，无滤波作用。GTDEB[3:0] = 1 - 15时，可以将滤波时间设置为30~1920*HCLK 15档，输入信号至少保持GTDEB[3:0]定义的时间长度，检测模块才认为检测输入信号有效，从而触发相应事件。

18.3.5　级联功能

级联运行是将2个模块的16位定时器组合为32位定时器的功能。

通过GTCR的TPSC[2:0]位将GPT0的定时器时钟设定为通过GPT1_GTCNT的上溢/下溢进行计数，实现级联运行的功能。

级联的两种组合：

组合	高16位	低16位
通道A和通道B	GPT0	GPT1

注1：级联之后，GPT0_GTCNT的值被自动映射到GPT1_GTCNT的高16位，所以读取只需要读取GPT1_GTCNT的32位寄存器。GPT1_GTCNT的高16位寄存器为只读属性，如果想在计数之前写高16位计数值，需要写GPT0_GTCNT。

注2：级联时，GPT0和GPT1的使能位都要开启。

18.3.5.1　定时器级联功能

级联之后高16位定时器只能选择GPT1作为时钟输入源。

定时器级联运行的例子：

设定GPT0的定时器的时钟源为GPT1的溢出，计数如下图：

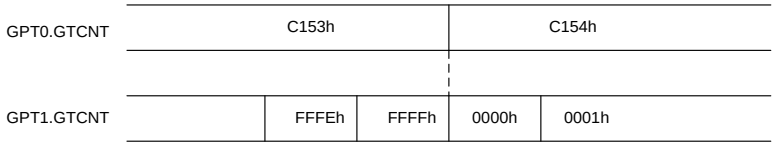


图18-8　　定时器级联运行示例

18.3.5.2　输入捕获级联功能

GPT模块能够通过两个16位定时器级联成一个32位捕捉定时器，同时两个定时器的4个输入捕捉口通过设定都可以做为32位定时器的输入捕捉口。同时捕捉结果会放在一个32位寄存器中，例如GPT0和GPT1级联时，GPT0的定时寄存器被映射到GPT1的定时寄存器的高位上，当读取时，可以直接读取GPT1_GTCNT的32bit值。

注：当处于级联模式下，每组级联的输入捕捉口的输入信号都可以做为清除源。

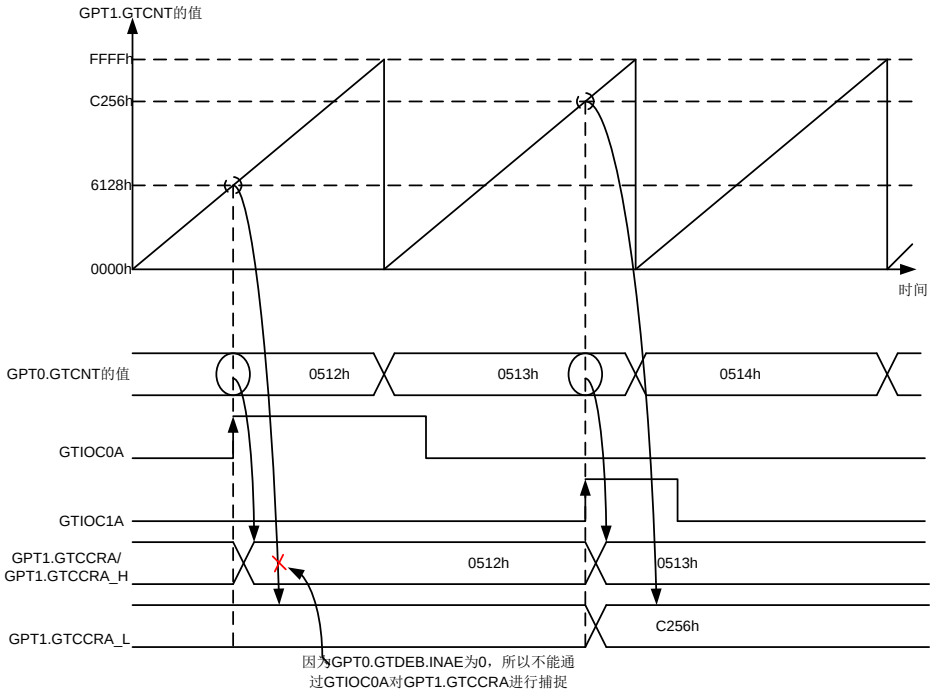


图18-9　　输入捕捉级联运行示例 1

输入捕捉级联示例1的寄存器设定：在将GPT0_GTCNT和GPT1_GTCNT进行级联，并且通过将位INAE@GPT1_GTDEB置’1’，使GTIOC1A引脚追加到GPT0_GTCCRA输入捕捉条件的运行如图18-9所示。在此例子中，将位GTIOA[5:0]@GPT0_GTIOR设定为在（GTIOC0A的）上升沿进行输入捕捉，将位GTIOA[5:0]@GPT1_GTIOR设定为在（GTIOC1A的）上升沿进行输入捕捉。此时，GTIOC0A和GTIOC1A的上升沿被设定为GPT0_GTCCRA的输入捕捉条件，GTIOC1A的上升沿被设定为GPT1_GTCCRA的输入捕捉条件。

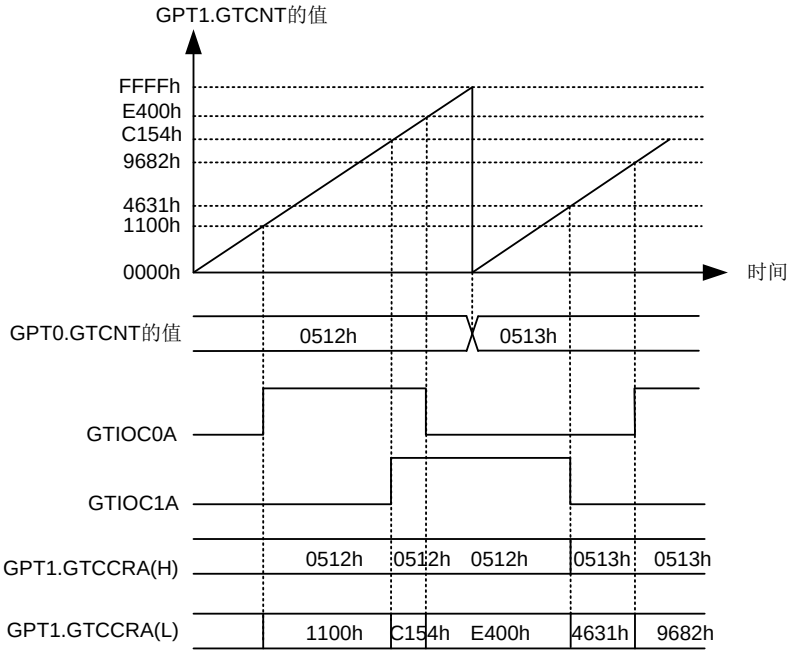


图18-10　　输入捕捉级联运行示例 2

输入捕捉级联示例2的寄存器设定：将GPT0_GTCNT和GPT1_GTCNT进行级联，并且通过将位INAE@GPT0_GTDEB使GTIOC0A引脚追加到GPT1_GTCCRA输入捕捉条件。位INAE@GPT1_GTDEB置’1’，使GTIOC1A引脚追加到GPT0_GTCCRA输入捕捉条件的运行如图18-10所示。在此例子中，将GPT0_GTIOR和GPT1_GTIOR的GTIOA[5:0]位都设定为在双边沿进行输入捕捉。

18.3.6 缓冲运行功能

能通过设定GTBER寄存器进行以下的缓冲运行。

• 进行GTCCRA寄存器和GTCCRC寄存器、GTCCRD寄存器组合的缓冲运行

• 进行GTCCRB寄存器和GTCCRE寄存器、GTCCRF寄存器组合的缓冲运行

• 进行GTPR寄存器和GTPBR寄存器、GTPDBR寄存器组合的缓冲运行

• 进行GTADTRA寄存器和GTADTBRA寄存器、GTADTDBRA寄存器组合的缓冲运行

• 进行GTADTRB寄存器和GTADTBRB寄存器、GTADTDBRB寄存器组合的缓冲运行或者，能通过设定GTDTCR寄存器进行以下的缓冲运行。

• 进行GTDVU寄存器和GTDBU寄存器组合的缓冲运行

• 进行GTDVD寄存器和GTDBD寄存器组合的缓冲运行

18.3.6.1 GTPR 寄存器的缓冲运行

GTPBR寄存器作为GTPR寄存器的缓冲寄存器运行，GTPDBR寄存器作为GTPBR寄存器的缓冲寄存器（GTPR寄存器的双缓冲寄存器）运行。

锯齿波时的缓冲传送时序为发生上溢（递增计数）或者下溢（递减计数）时，三角波时的缓冲传送时序为波谷。

GTPR寄存器进行双缓冲运行时，必须将位PR[1:0]@GPTx_GTBER设定为“10b”或者“11b”，GTPR寄存器进行单缓冲运行时，必须将位PR[1:0]@GPTx_GTBER设定为“01b”，GTPR寄存器不进行缓冲运行时，必须将位PR[1:0]@GPTx_GTBER设定为“00b”。

GTPR寄存器的缓冲运行例子如图18-11～图18-13。

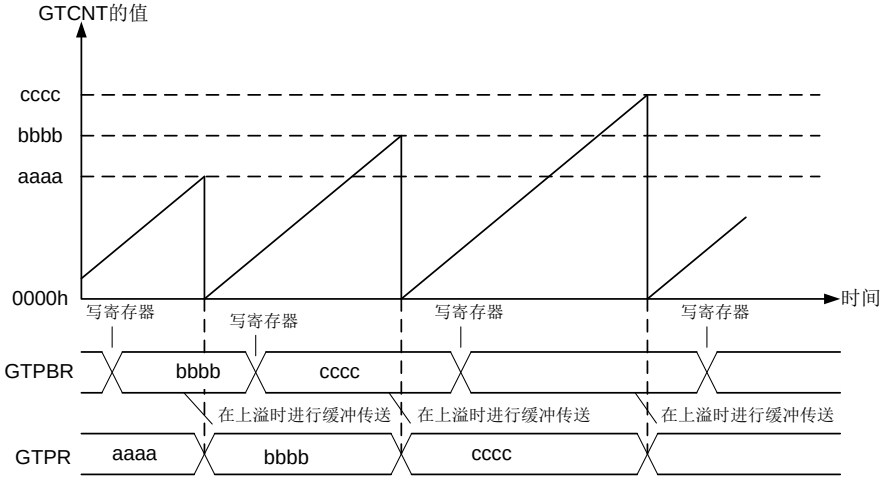


图18-11 GTPR 寄存器的缓冲运行例子（在锯齿波进行递增计数的情况）

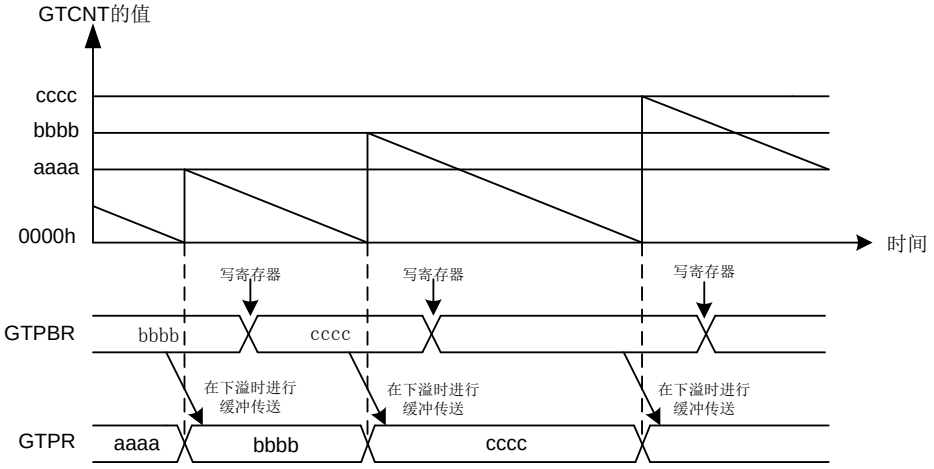


图18-12　　GTPR 寄存器的缓冲运行例子（在锯齿波进行递减计数的情况）

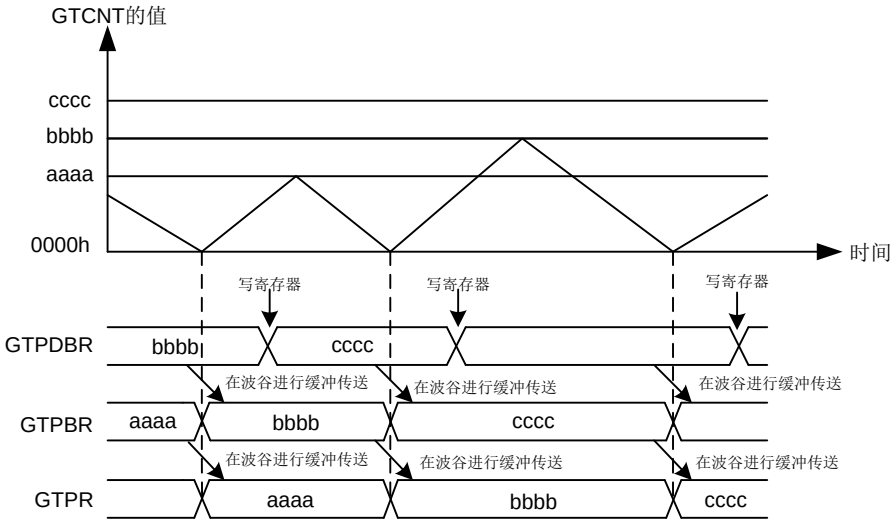


图18-13　　GTPR 寄存器的缓冲运行例子（三角波的情况）

18.3.6.2　　GTCCRA 寄存器和GTCCRB 寄存器的缓冲运行

GTCCRC 寄存器作为GTCCRA 寄存器的缓冲寄存器运行，GTCCRD 寄存器作为GTCCRC 寄存器的缓冲寄存器（GTCCRA 寄存器的双缓冲寄存器）运行。同样。GTCCRE 寄存器作为GTCCRB 寄存器的缓冲寄存器运行，GTCCRF 寄存器作为GTCCRE 寄存器的缓冲寄存器（GTCCRB 寄存器的双缓冲寄存器）运行。

GTCCRA 寄存器和GTCCRB 寄存器进行双缓冲运行时，必须分别将位CCRA[1:0]@GPTx_GTBER 和位CCRB[1:0]@GPTx_GTBER设定为“10b”或者“11b”，GTCCRA 寄存器和GTCCRB 寄存器进行单缓冲运行时，必须将CCRA[1:0]位和CCRB[1:0] 位设定为“01b”，GTCCRA 寄存器和GTCCRB 寄存器不进行冲运行时，必须将CCRA[1:0] 位和CCRB[1:0] 位设定为“00b”。

GTCCRA 寄存器和GTCCRB 寄存器作为输出比较寄存器运行的情况

锯齿波时的缓冲传送时序为发生上溢（递增计数）或者下溢（递减计数）时，三角波时的缓冲传送时序为波谷。

GTCCRA寄存器和GTCCRB寄存器的缓冲运行例子如图18-14～图18-16。

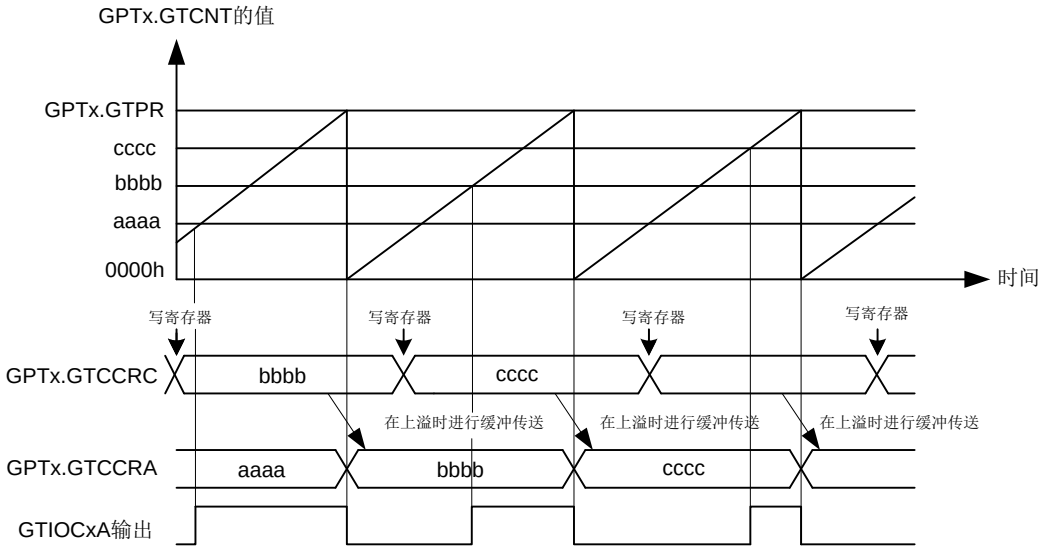


图18-14　　GTCCRA 寄存器的缓冲运行例子

（输出比较、在锯齿波进行递增计数、在 GTCCRA 比较匹配时输出 High 电平、在周期结束时输出 Low 电平的情况）

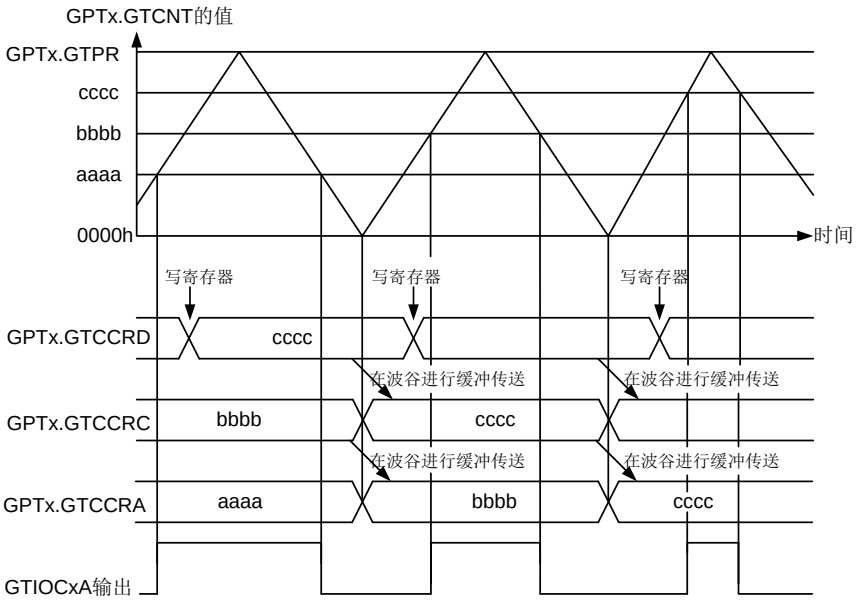


图18-15　　GTCCRA 寄存器的双缓冲运行例子

（输出比较、三角波、在波谷进行缓冲传送、在 GTCCRA 比较匹配时进行交替输出、在周期结束时保持输出的情况）

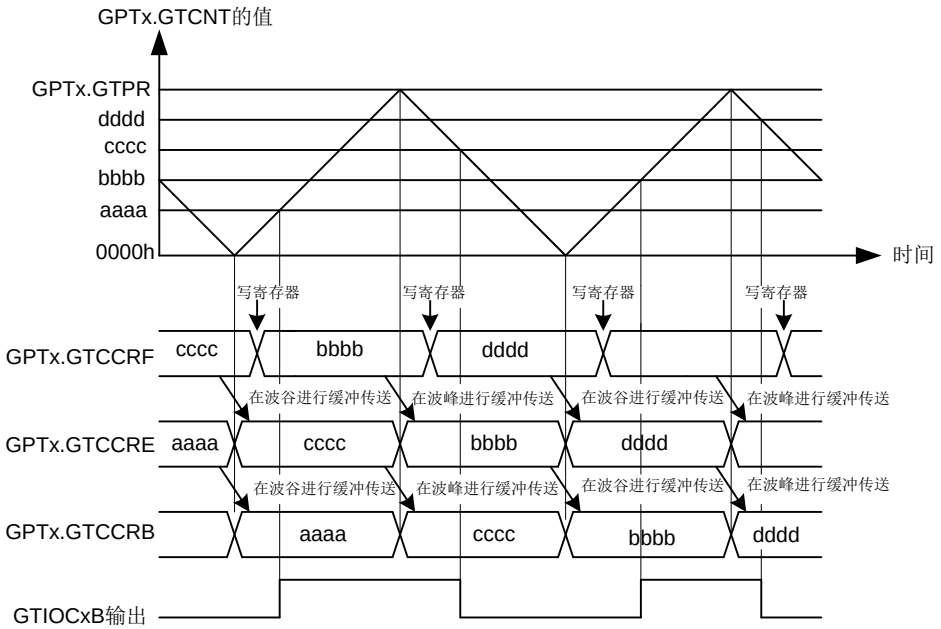


图18-16　　GTCCRB 寄存器的双缓冲运行例子

（输出比较、三角波、在波峰和波谷都进行缓冲传送、在 GTCCRB 比较匹配时进行交替输出、在周期结束时保持输出的情况）

GTCCRA寄存器和GTCCRB寄存器作为输入捕捉寄存器运行的情况

缓冲传送的时序为发生输入捕捉时。如果发生输入捕捉，就在将GTCNT计数器的值传送到GTCCRA寄存器和GTCCRB寄存器的同时，将保存在GTCCRA寄存器和GTCCRB寄存器的值传送到缓冲寄存器。

GTCCRA寄存器和GTCCRB寄存器的缓冲运行例子如图18-17、图18-18。

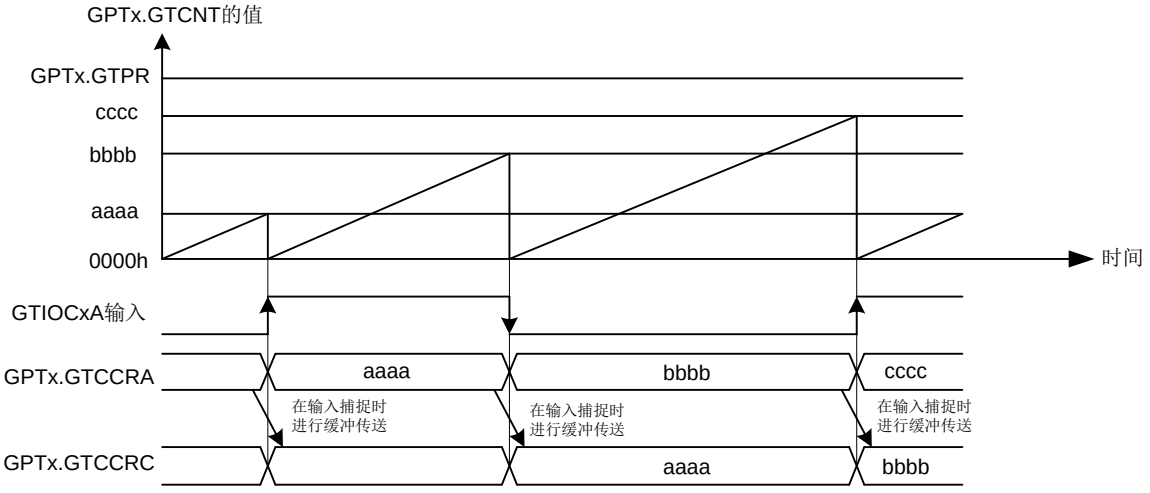


图18-17　　GTCCRA 寄存器的缓冲运行例子

（在锯齿波进行递增计数、在 GTIOCxA 输入的双边沿进行输入捕捉、在 GTCCRA 寄存器的输入捕捉时进行 GTCNT 计数器清除的情况）

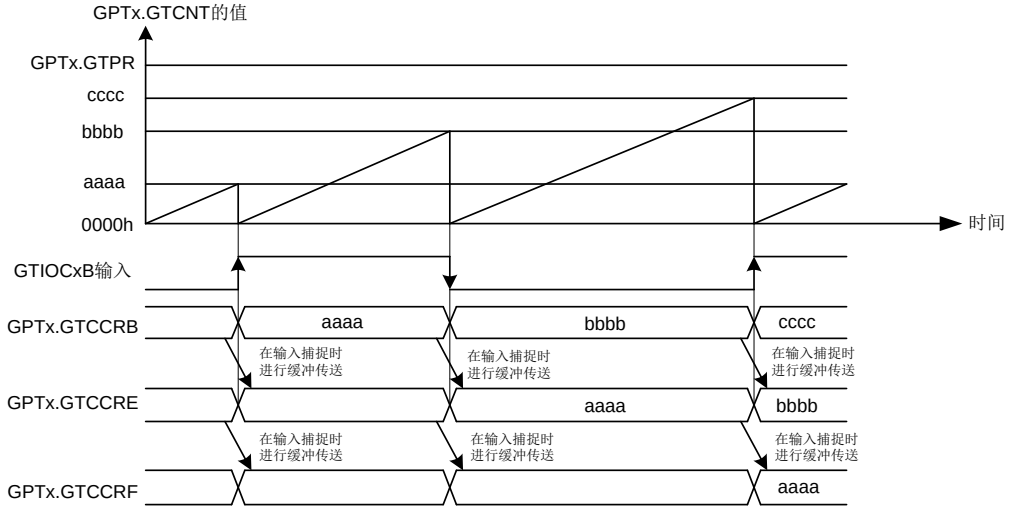


图18-18　　GTCCRB 寄存器的双缓冲运行例子

（在锯齿波进行递增计数、在 GTIOCxB 输入的双边沿进行输入捕捉、在 GTCCRB 寄存器的输入捕捉时进行 GTCNT 计数器清除的情况）

18.3.6.3　GTADTRA 寄存器和GTADTRB 寄存器的缓冲运行

GTADTBRA寄存器作为GTADTRA寄存器的缓冲寄存器运行，GTADTDBRA寄存器作为GTADTBRA寄存器的缓冲寄存器（GTADTRA寄存器的双缓冲寄存器）运行。同样，GTADTBRB寄存器作为GTADTRB寄存器的缓冲寄存器运行，GTADTDBRB寄存器作为GTADTBRB寄存器的缓冲寄存器（GTADTRB寄存器的双缓冲寄存器）运行。

GTADTRA 寄存器和GTADTRB 寄存器进行双缓冲运行时，必须分别将位ADTDA@GPTx_GTBER 和位ADTDB@GPTx_GTBER设定为”1”，GTADTRA寄存器和GTADTRB寄存器进行单缓冲运行时，必须将ADTDA位和ADTDB位设定为”0”，GTADTRA寄存器和GTADTRB寄存器不进行缓冲运行时，必须分别将位ADTTA[1:0]@GPTx_GTBER 和位ADTTB[1:0]@GPTx_GTBER设定为”00”。

能通过ADTTA[1:0]位设定缓冲传送的时序。锯齿波时的缓冲时序为发生上溢（递增计数）或者下溢（递减计数）时，三角波时的缓冲传送时序在ADTTA[1:0]位是”01”时为波峰、“10”时为波谷，“11”时为波谷和波峰。

GTADTRA寄存器和GTADTRB寄存器的缓冲运行例子如图18-19～图18-21。

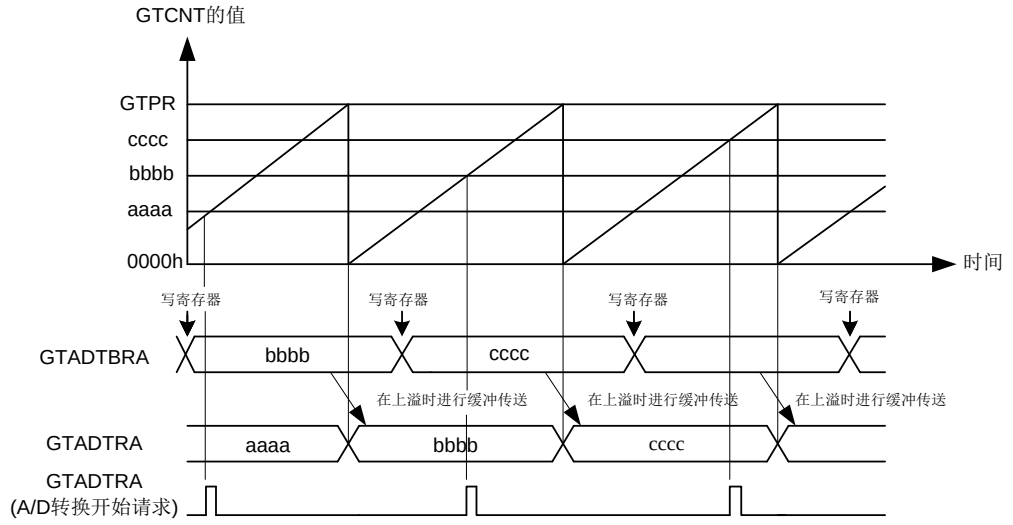


图18-19　　GTADTRA 寄存器的缓冲运行例子

（在锯齿波进行递增计数、在递增计数时产生 A/D 转换开始请求的情况）

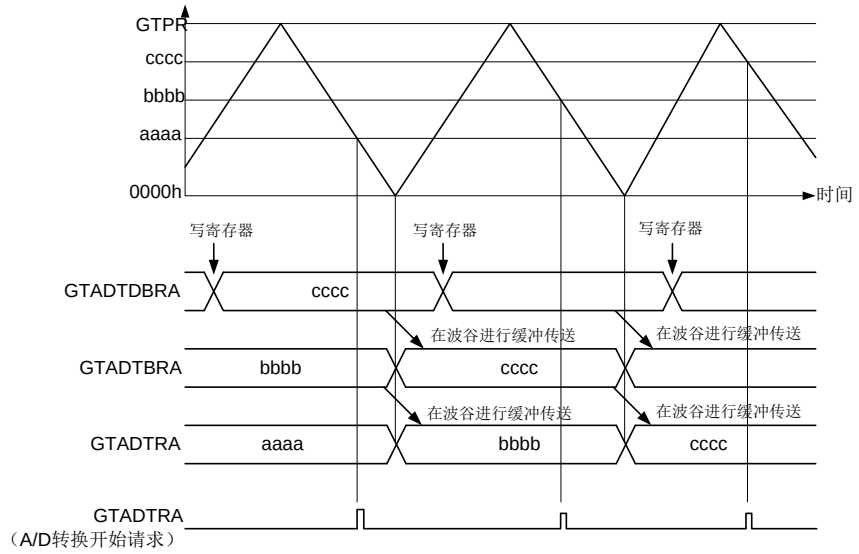


图18-20　GTADTRA 寄存器的双缓冲运行例子（三角波、在波谷进行缓冲传送、在递减计数时产生 A/D 转换请求的情况）

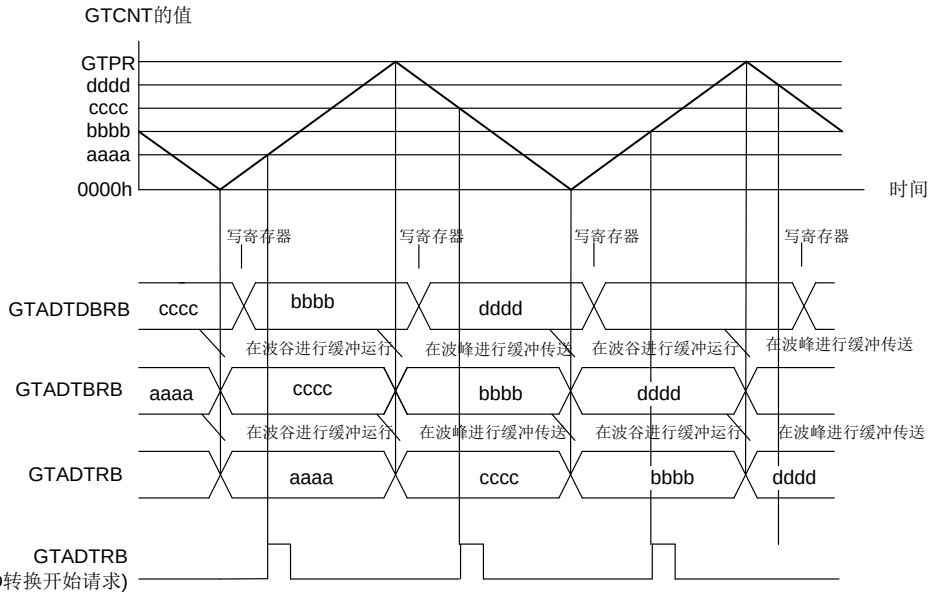


图18-21　GTADTRB 寄存器的双缓冲运行例子（三角波、在波谷和波峰都进行缓冲传送、在递增计数和递减计数时都产生 A/D 转换请求的情况）

18.3.7　PWM 输出运行模式

能通过GPTx_GTCNT计数器和GPTx_GTCCRA寄存器、GPTx_GTCCRB寄存器的比较匹配，对GTIOCxA引脚和GTIOCxB引脚输出PWM波形。能给全部通道独立设定运行模式，也能进行通道间的同步运行。

能通过设定GTDTCR寄存器、GTDVU寄存器和GTDVD寄存器，将用于带死区时间的反相波形的比较匹配值自动设定到GTCCRB寄存器。

18.3.7.1　锯齿波 PWM 模式

锯齿波PWM模式是将周期设定到GPTx_GTPR寄存器，使GPTx_GTCNT计数器进行锯齿波（半波）运行，并且通过GPTx_GTCCRA寄存器、GPTx_GTCCRB寄存器的比较匹配将PWM波形输出到GTIOCxA引脚和GTIOCxB引脚的模式。通过GTIOR寄存器将引脚的输出值设定为在比较匹配时进行Low电平输出/High电平输出/交替输出，在周期结束时进行Low电平输出/High电平输出/交替输出。

锯齿波PWM模式的运行例子如图18-22。

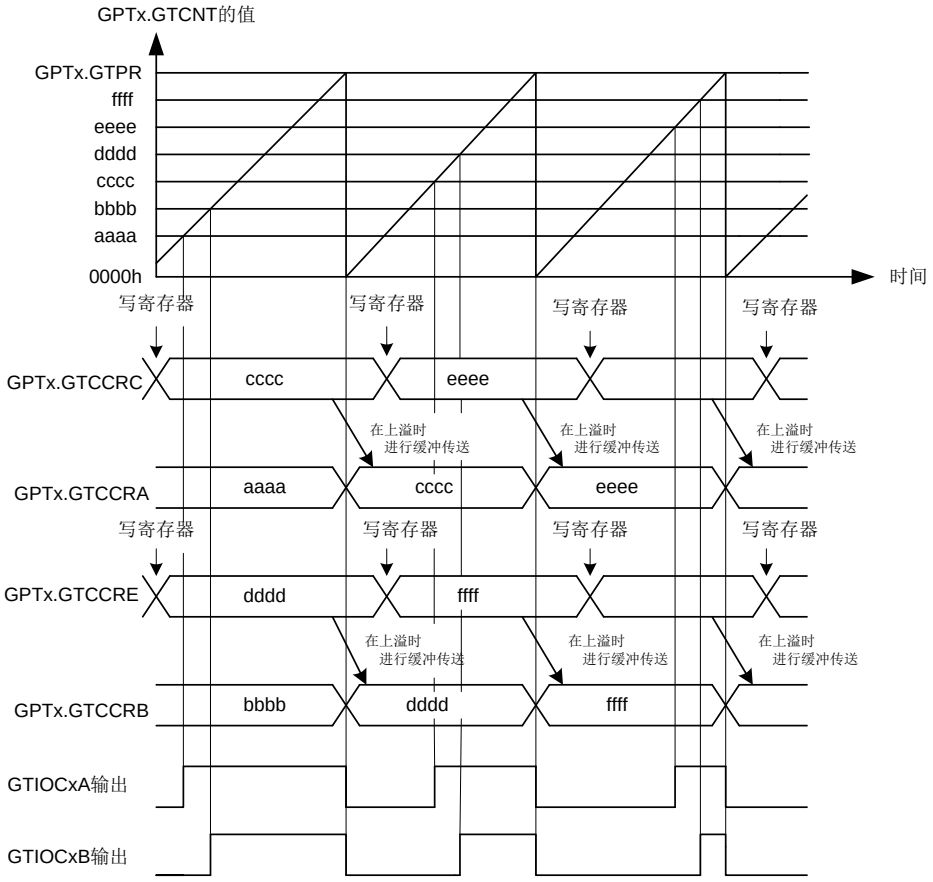


图18-22　锯齿波 PWM 模式的运行例子（递增计数、缓冲运行、在 GTCCRA/B 比较匹配时进行 High 电平输出、在周期结束时进行 Low 电平输出的情况）

18.3.7.2　锯齿波单触发脉冲模式

锯齿波单触发脉冲模式是将周期设定到GPTx_GTPR寄存器，使GPTx_GTCNT计数器进行锯齿波（半波）运行，并且固定为缓冲运行，通过GPTx_GTCCRA寄存器、GPTx_GTCCRB寄存器的比较匹配，将PWM波形输出到GTIOCxA引脚和GTIOCxB引脚的模式。锯齿波单触发脉冲模式的缓冲运行与通常的缓冲运行不同，在周期结束时，进行从GTCCRC寄存器到GTCCRA寄存器、从GTCCRE寄存器到GTCCRB寄存器、从GTCCRD寄存器到暂存器A、从GTCCRF寄存器到暂存器B的缓冲传送。而且在GTCCRA寄存器的比较匹配时进行从暂存器A到GTCCRA寄存器的缓冲传送，在GTCCRB寄存器的比较匹配时进行从暂存器B到GTCCRB寄存器的缓冲传送。能通过GTIOR寄存器将引脚的输出值设定为在比较匹配时进行Low电平输出/High电平输出/交替输出，在周期结束时进行Low电平输出/High电平输出/交替输出。

通过设定GTDTCR寄存器、GTDVU寄存器和GTDVD寄存器，将用于带死区时间的反相波形的比较匹配值自动设定到GTCCRB寄存器。

锯齿波单触发脉冲模式的运行例子如图18-23。

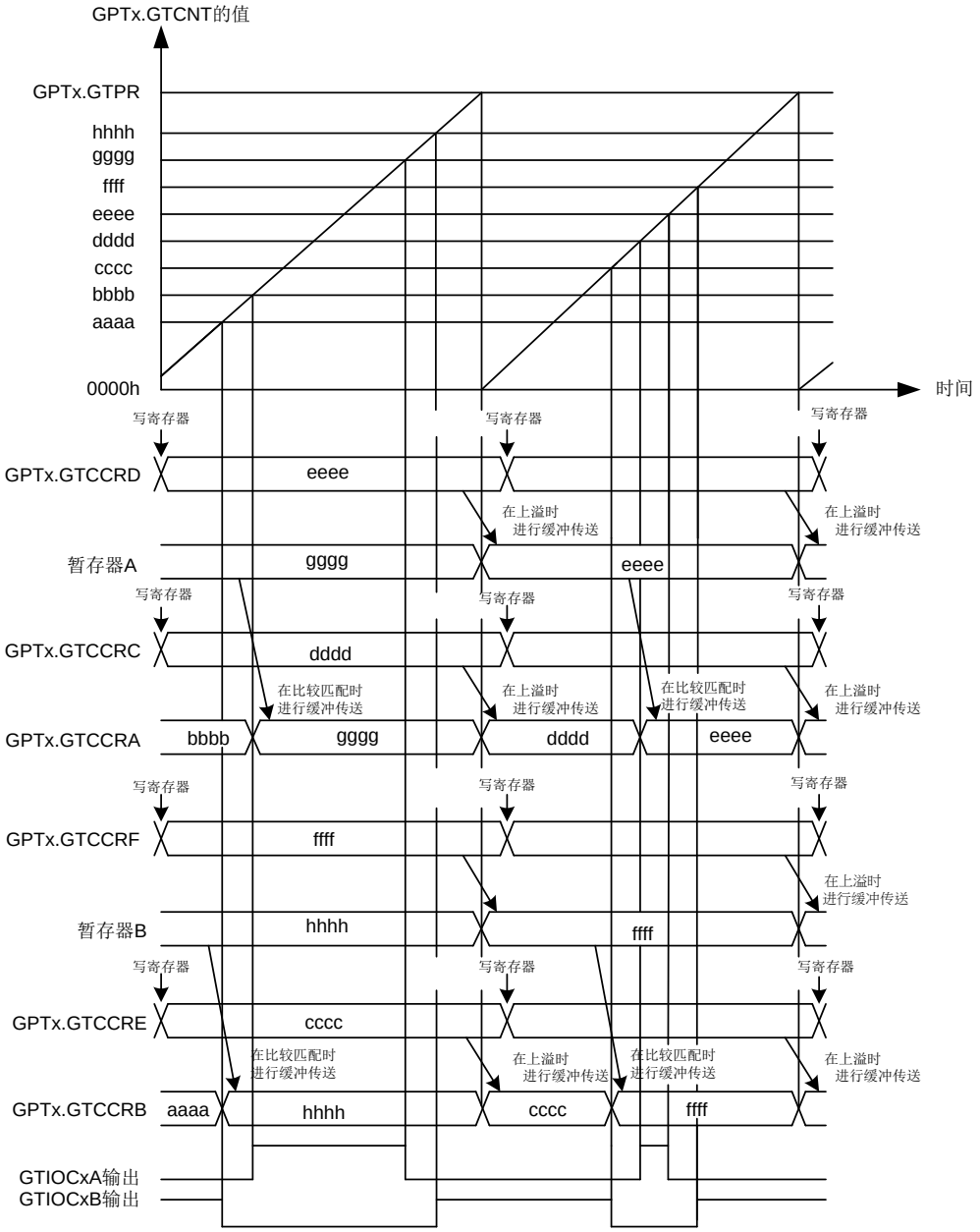


图18-23　锯齿波单触发脉冲模式的运行例子（递增计数、在开始计数时 GTIOCxA 引脚为 Low 电平输出 /GTIOCxB 引脚为 High 电平输出、在 GTCCRA/B 比较匹配时进行交替输出、在周期结束保持输出的情况）

注：在未开启前，暂存器的初值等于相应的缓冲值。

18.3.7.3　三角波 PWM 模式 1（在波谷进行 16 位传送）

三角波PWM模式1是将周期设定到GPTx_GTPR寄存器，使GPTx_GTCNT进行三角波（全波）运行，并且通过GPTx_GTCCRA寄存器、GPTx_GTCCRB寄存器的比较匹配，将PWM波形输出到GTIOCxA引脚和GTIOCxB引脚的模式。缓冲运行的时序为波谷。能通过GTIOR寄存器将引脚的输出值设定为在比较匹配时进行Low电平输出/High电平输出/交替输出，在周期结束时进行Low电平输出/HIgh电平输出/交替输出。

能通过设定GTDTCR寄存器、GTDVU寄存器和GTDVD寄存器，将用于带死区时间的反相波形的比较匹配值自动设定到GTCCRB寄存器。

三角波PWM模式1的运行例子如图18-24。

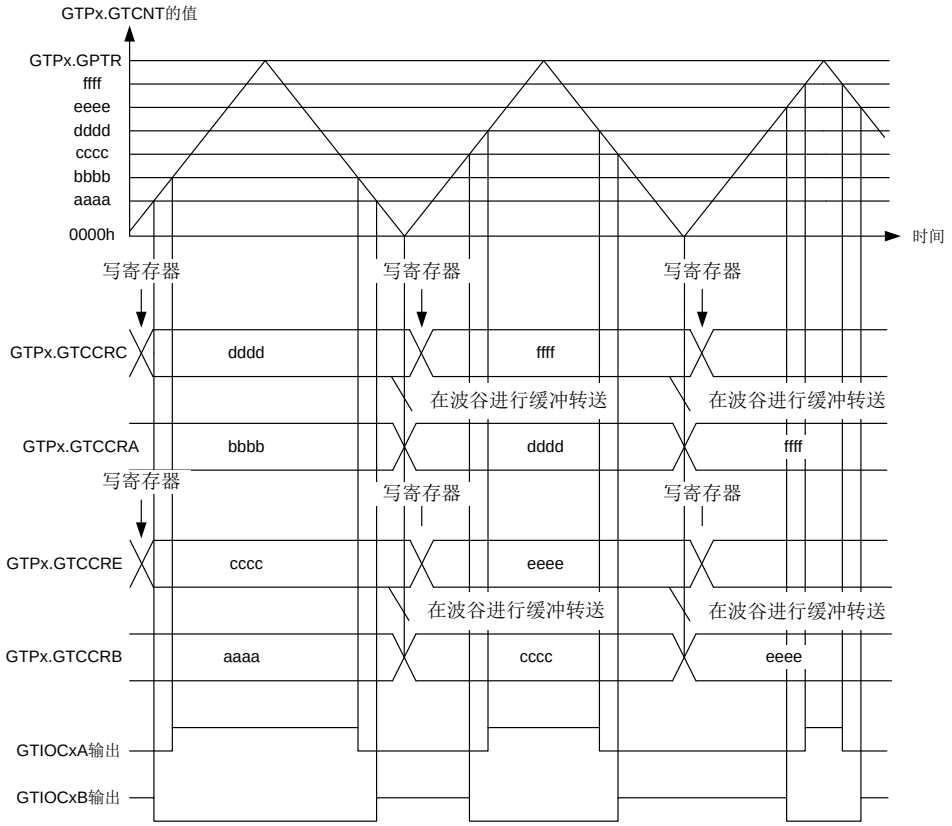


图18-24　三角波 PWM 模式 1 的运行例子（缓冲运行、在开始计数时 GTIOCxA 引脚为 Low 电平输出 /GTIOCxB 引脚为 High 电平输出、在 GTCCRA/B 比较匹配时进行交替输出、在周期结束时保持输出的情况）

18.3.7.4　三角波PWM 模式2（在波峰和波谷进行16 位传送）

三角波PWM模式2与三角波PWM模式1一样，是将周期设定到GPTx_GTPR寄存器，使GPTx_GTCNT进行三角波（全波）运行，并且通过GPTx_GTCCRA寄存器、GPTx_GTCCRB寄存器的比较匹配，将PWM波形输出到GTIOCxA引脚和GTIOCxB引脚的模式。但是，缓冲运行的时序为波峰和波谷。能通过GTIOR寄存器将引脚的输出值设定为在比较匹配时进行Low电平输出/High电平输出/交替输出，在周期结束时进行Low电平输出/High电平输出/交替输出。

通过设定GTDTCR寄存器、GTDVU寄存器和GTDVD寄存器，将用于带死区时间的反相波形的比较匹配值自动设定到GTCCRB寄存器。

三角波PWM模式2的运行例子如图18-25。

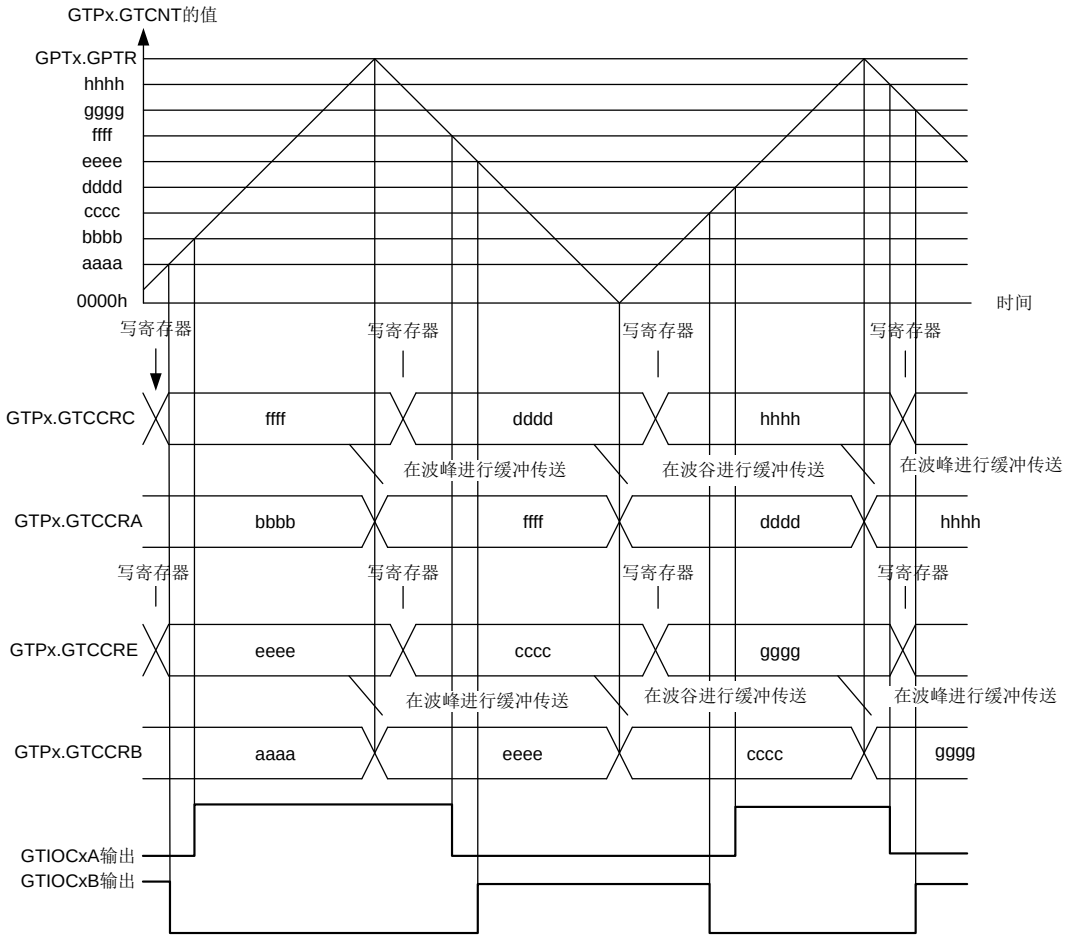


图18-25　三角波 PWM 模式 2 的运行例子（缓冲运行、在开始计数时 GTIOCxA 引脚为 Low 电平输出 /GTIOCxB 引脚 High 高电平输出、在 GTCCRA/B 比较匹配时进行交替输出、在周期结束时保持输出的情况）

18.3.7.5　三角波PWM 模式 3（在波谷进行 32 位传送）

三角波PWM模式3是将周期设定到GPTx_GTPR寄存器，使GPTn.GTCNT进行三角波（全波）运行，并且固定为缓冲运行，通过GPTx_GTCCRA寄存器、GPTx_GTCCRB寄存器的比较匹配，将PWM波形输出到GTIOCxA引脚和GTIOCxB引脚的模式。三角波PWM模式3的缓冲运行与通常的缓冲运行不同，在波谷进行从GTCCRC寄存器到GTCCRA寄存器、从GTCCRE寄存器到GTCCRB寄存器、从GTCCRD寄存器到暂存器A、从GTCCRF寄存器到暂存器B的缓冲传送。而且在波峰进行从暂存器A到GTCCRA寄存器、从暂存器B到GTCCRB寄存器的缓冲传送。通过GTIOR寄存器将引脚的输出值设定为在比较匹配时进行Low电平输出/High电平输出/交替输出，在周期结束时进行Low电平输出/High电平输出/交替输出。

通过设定GTDTCR寄存器、GTDVU寄存器和GTDVD寄存器，将用于带死区时间的反相波形的比较匹配值自动设定到GTCCRB寄存器。

三角波PWM模式3的运行例子如图18-26。

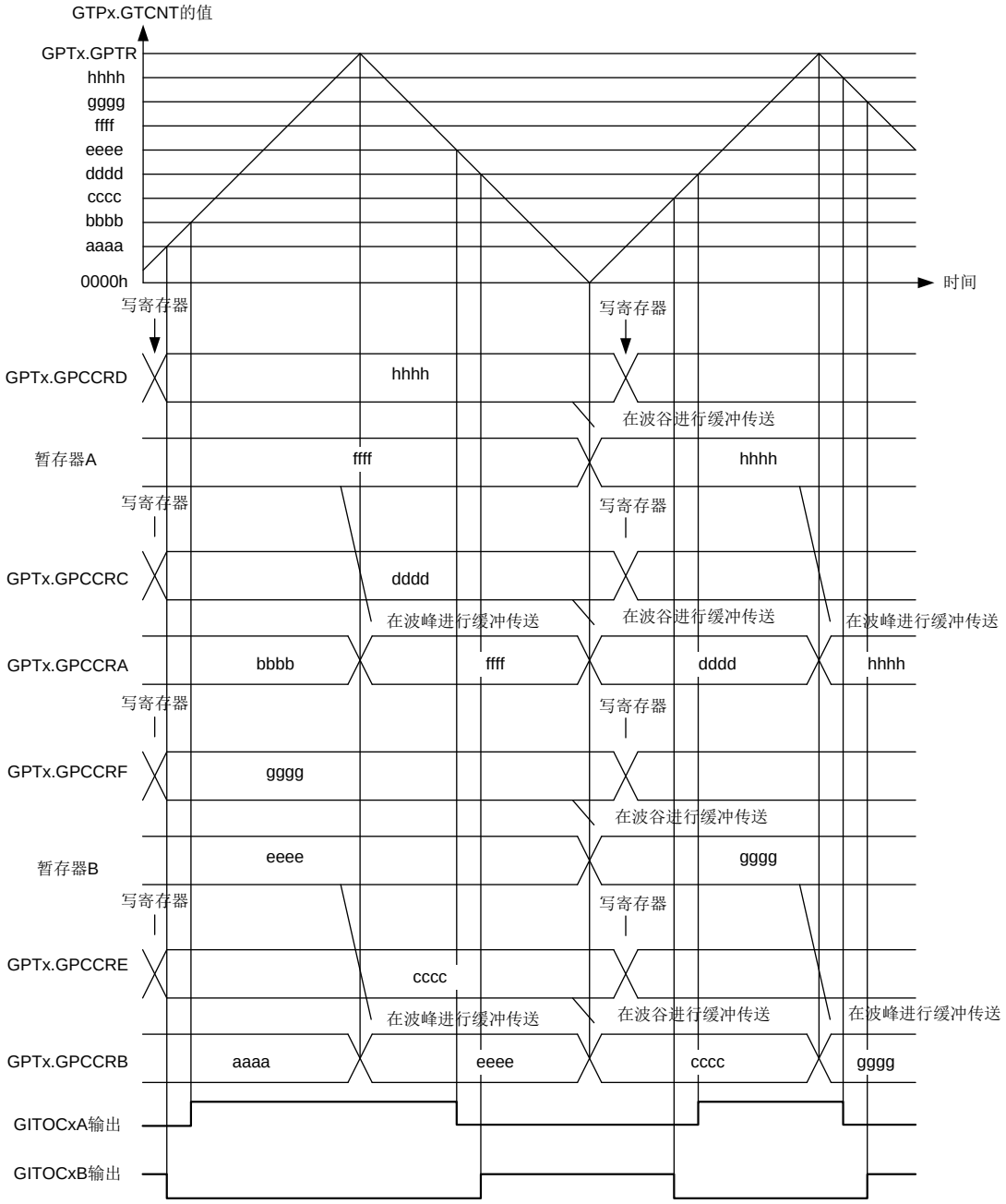


图18-26　三角波 PWM 模式 3 的运行例子（在开始计数时 GTIOCxA 引脚为 Low 电平输出 /GTIOCxB 引脚为 High 电平输出、在 GTCCRA/B 比较匹配时进行交替输出、在周期结束时保持输出的情况）

18.3.7.6　三角波 PWM 模式 4（在波峰进行 16 位传送）

三角波PWM模式4是将周期设定到GPTx_GTPR寄存器，使GPTx_GTCNT进行三角波（全波）运行，并且通过GPTx_GTCCRA寄存器、GPTx_GTCCRB寄存器的比较匹配，将PWM波形输出到GTIOCxA引脚和GTIOCxB引脚的模式。缓冲运行的时序为波峰。通过GTIOR寄存器将引脚的输出值设定为在比较匹配时进行Low电平输出/High电平输出/交替输出，在周期结束时进行Low电平输出/HIgh电平输出/交替输出。

通过设定GTDTCR寄存器、GTDVU寄存器和GTDVD寄存器，将用于带死区时间的反相波形的比较匹配值自动设定到GTCCRB寄存器。

18.3.8 死区时间自动设定

通过设定GTDTCR寄存器，从正相波形的比较匹配值（GTCCRA寄存器的值）和死区时间值（GTDVU寄存器和GTDVD寄存器的值）生成带死区时间的反相波形的比较匹配值，并且自动设定到GTCCRB寄存器。

死区时间自动设定功能可以用于锯齿波单触发脉冲模式和全部的三角波PWM模式。

可以在前侧和后侧独立设定死区时间，也可以在前侧和后侧使用共同的的死区时间。通过GTDVU寄存器设定对应于反相波形前侧转换时序的死区时间，通过GTDVD寄存器设定对应于反相波形后侧转换时序的死区时间。

另外，GTDBU寄存器可以用作GTDVU寄存器的缓冲寄存器。同样，GTDBD寄存器可以用作GTDVD寄存器的缓冲寄存器。缓冲传送时序为周期结束（锯齿波时GTCNT为发生上溢（递增计数）或者下溢（递减计数）时，三角波时为波谷）时。

在使用死区时间自动设定功能时，禁止写GTCCRB寄存器，也禁止设定超出周期的死区时间。可以通过读GTCCRB寄存器的值确认死区时间的自动设定值。

死区时间自动设定功能的运行例子如图18-27～图18-29。

注1：死区时间自动设定会造成GTCCRB寄存器的值超出(0~GTPR)范围，所以如果自动生成的GTCCRB小于0时，GTCCRB为0，如果自动生成的GTCCRB大于GTPR时，GTCCRB为GTPR。

注2：以上设定会出现没有死区的情况，所以在使用时要特别注意GTCCRA的范围。

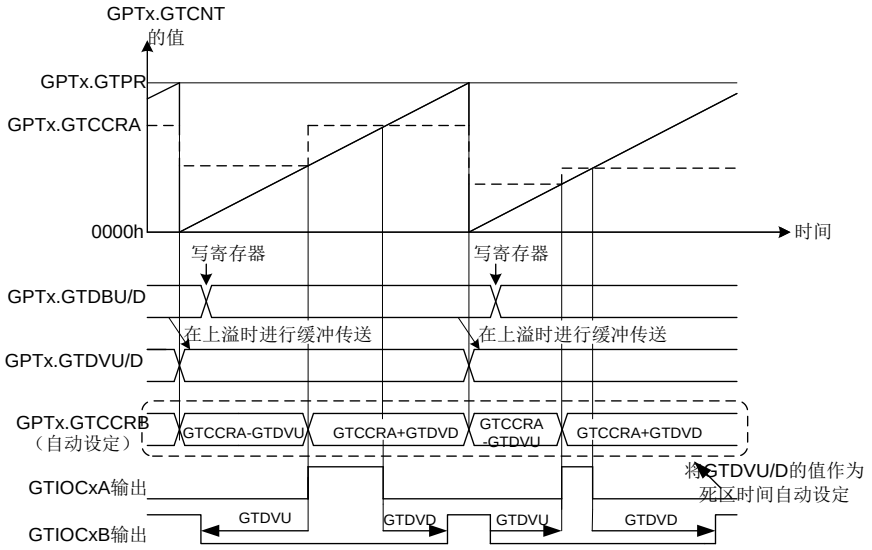


图18-27　死区时间自动设定功能的运行例子

（锯齿波单触发脉冲模式、 GTDVU/D 寄存器进行缓冲运行、有效电平为 High 电平的情况）

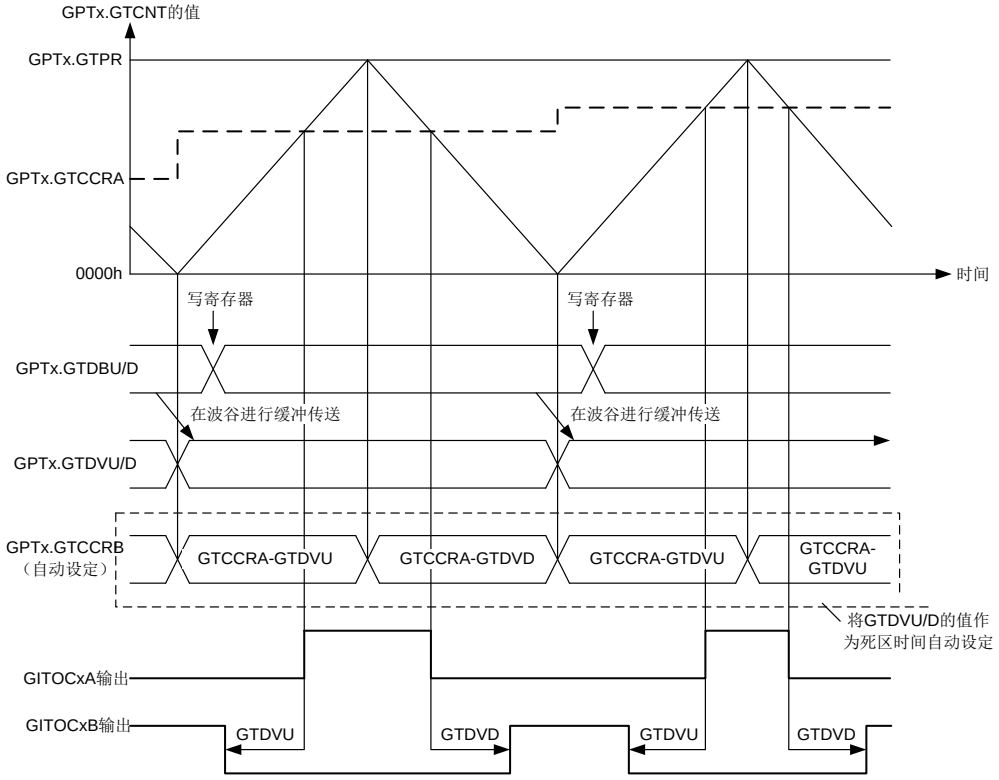


图18-28　　带死区时间自动设定功能的运行例子

（三角波 PWM 模式 1 、 GTDVU/D 寄存器进行缓冲运行、有效电平为 High 电平的情况）

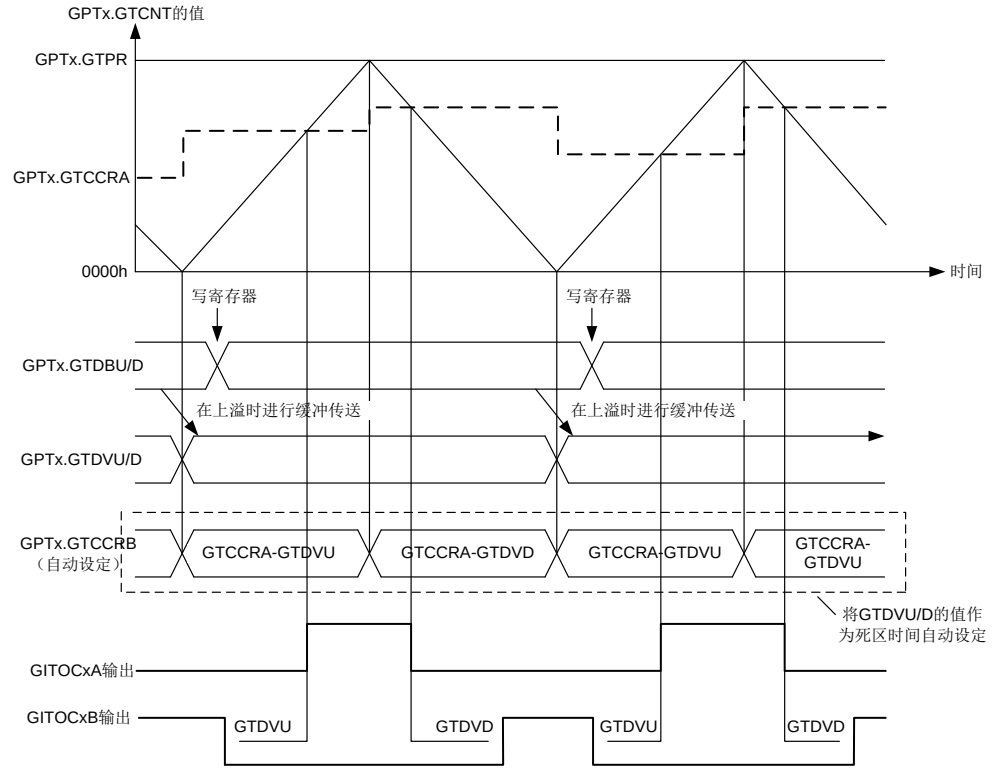


图18-29　　死区时间自动设定功能的运行例子

（三角波 PWM 模式 2 和 3、 GTDVU/D 寄存器进行缓冲运行、有效电平为 High 电平的情况）

18.3.9 计数方向转换功能

通过更改位UD@GPTx_GTUDC的值，可以转换GTCNT计数器的计数方向。

为锯齿波的情况时，如果在计数时更改UD位的值，就在发生上溢（在递增计数时更改）或者下溢（递减计数时更改）时转换计数方向。如果在停止计数并且位UDF@GPTx_GTUDC为“0”的状态下更改UD位的值，计数开始后不反映此值，但是在发生上溢或者下溢时转换计数方向。如果在停止计数时将UDF位置“1”，就从计数开始后反映此时的UD位的值。

为三角波的情况时，即使在计数时更改UD位的值也不转换计数方向。同样，即使在停止计数并且UDF位为“0”的状态下更改UD位的值也不反映此值。如果在停止计数时将UDF位置’1’，就从计数开始后反映此时的UD位的值。

在锯齿波的计数过程中转换计数方向时，如果是递增计数，就将递增计数开始后的GTPR寄存器值反映到计数周期，如果是递减计数，就将递减计数开始前的GTPR寄存器值反映到计数周期。

计数方向转换功能的运行例子如图18-30所示

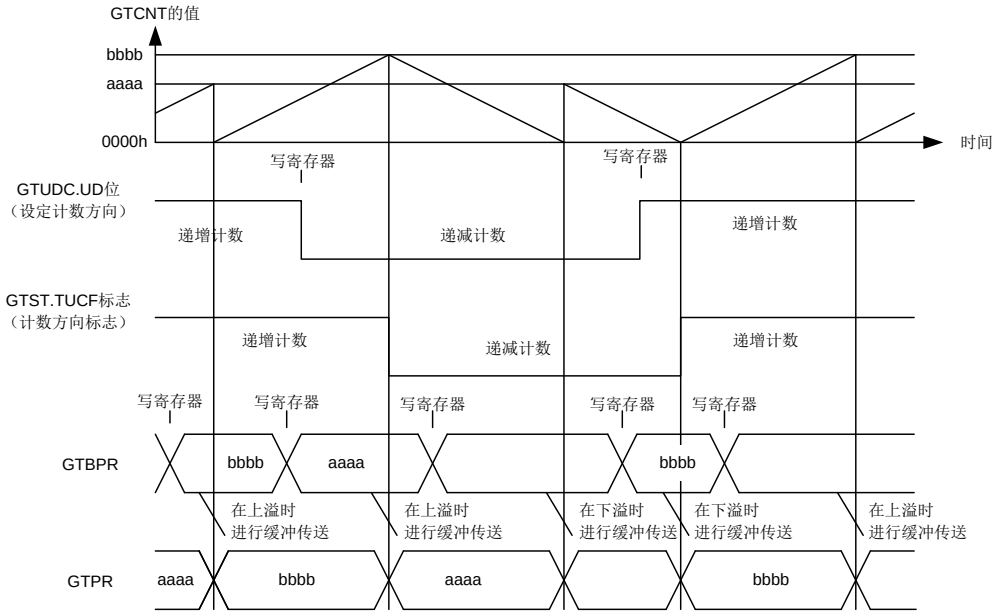


图18-30　　计数方向转换功能的运行例子（缓冲运行时）

18.3.10 硬件启动/停止、清除运行

通过SH32F284内部的硬件源GTETRG输入引脚可以对GTCNT计数器的启动、停止以及清除进行控制。另外，也可以通过GTCCRA寄存器和GTCCRB寄存器的输入捕捉进行计数器清除。

18.3.10.1　硬件启动

通过硬件源可以控制GTCNT计数器的启动，通过位CSHW@GPTx_GTHCR指定硬件源的转换边沿并且允许计数。

通过硬件源启动计数的运行例子如图18-31所示。

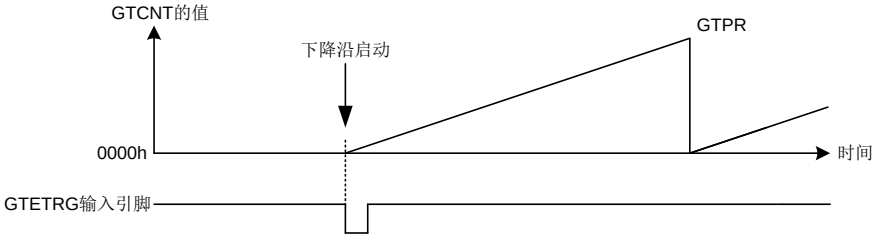


图18-31　　通过硬件源启动计数的运行例子

18.3.10.2　硬件停止运行

通过硬件源可以控制GTCNT计数器的停止。仅支持GTETRG硬件源。通过位CPHW@GPTx_GTHCR指定硬件源的转换边沿并且允许停止计数。

通过硬件源开始或者停止计数的运行例子如图18-32所示。在此例子中，外部输入触发GTETRG在High电平区间进行计数。

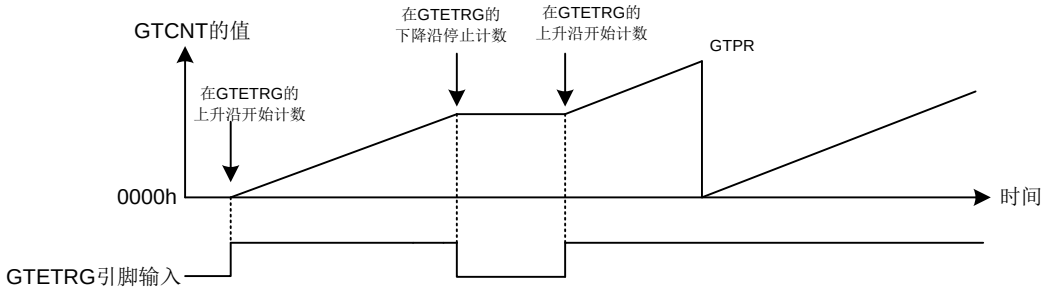


图18-32　通过硬件源开始或者停止计数的运行例子
（在 GTETRG 引脚输入的上升沿开始计数，在 GTETRG 引脚输入的下降沿停止计数）

18.3.10.3　硬件清除运行

通过硬件源可以控制GTCNT计数器的清除。通过位CCHW@GPTx_GTHCR指定硬件源的转换边沿并且允许清除计数器。

另外，通过设定位CCLR[1:0]@GPTx_GTCR，GTCCRA寄存器和GTCCRB寄存器的输入捕捉也可以进行计数器清除。

即使发生硬件源清除计数器和通过软件源清除计数器，也不发生GTCIV中断（上溢/下溢中断）。

通过硬件源进行计数器清除的运行例子如图18-33、图18-34所示。在此例子中，通过GTIOCxA引脚输入的双边沿开始计数，并且通过GTIOCxB引脚输入的双边沿停止或者清除计数的例子。

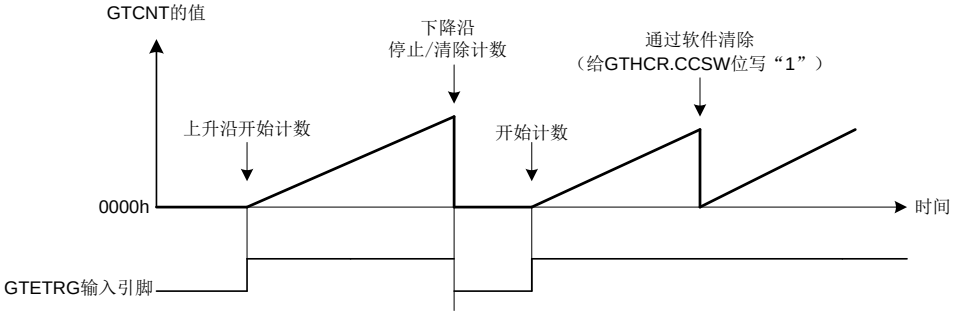


图18-33　通过硬件源进行计数器清除的运行例子
（在锯齿波进行递增计数、通过 GTETGR 引脚输入上升沿开始计数、通过 GTETGR 引脚输入下降沿沿停止和清除计数）

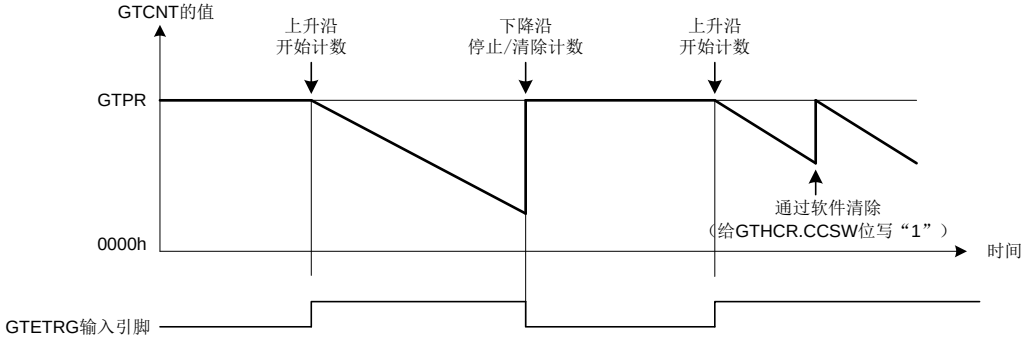


图18-34　通过硬件源进行计数器清除的运行例子
（在锯齿波进行递减计数、通过 GTETRG 引脚输入上升沿开始计数、通过 GTETRG 引脚输入下降沿沿停止和清除计数）

通过硬件进行计数器清除和GTCIV中断的关系如图18-35所示。

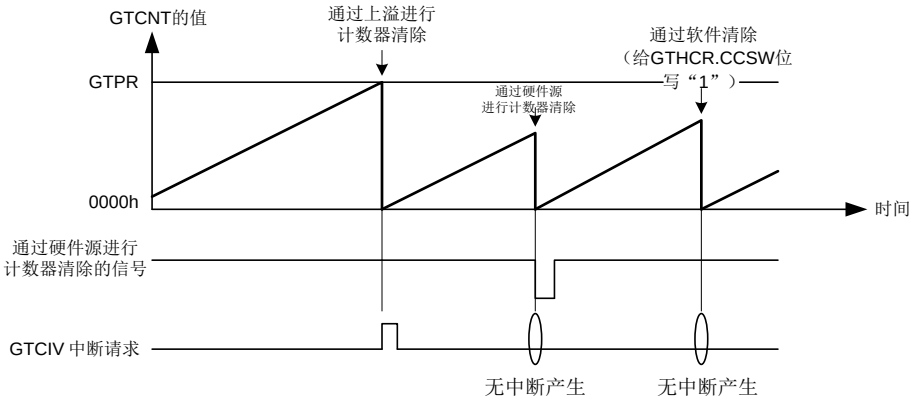


图18-35　通过硬件进行计数器清除和 GTCIV 中断的关系

18.3.11 同步运行

可以进行通道间的同步运行（同步清除、同步启动）。

18.3.11.1　同步清除运行

不同模块计数器可以实现同步清除。将需要进行同步清除的模块的位CCLR[1:0]@GPTx_GTCR置“11b”，然后通过位SYNC[1:0]@GPT_GTHCR设定通过哪个模块的清除源进行同步清除。

同步清除的运行例子如图18-36所示。在此例子中，通过GPT0_GTCNT的清除源（上溢）同步清除GPT1_GTCNT和GPT2_GTCNT的例子。

不发生已被同步清除的模块的同步清除（同步清除不被传播）。

已被同步清除的模块的同步清除运行例子如图18-37所示。在此例子中，通过GPT0_GTCNT的清除源（上溢）同步清除GPT1_GTCNT，通过GPT1_GTCNT的清除源（上溢）同步清除GPT2_GTCNT的例子。通过GPT0_GTCNT的清除源（上溢）同步清除的GPT1_GTCNT的同步清除不传播到GPT2_GTCNT。

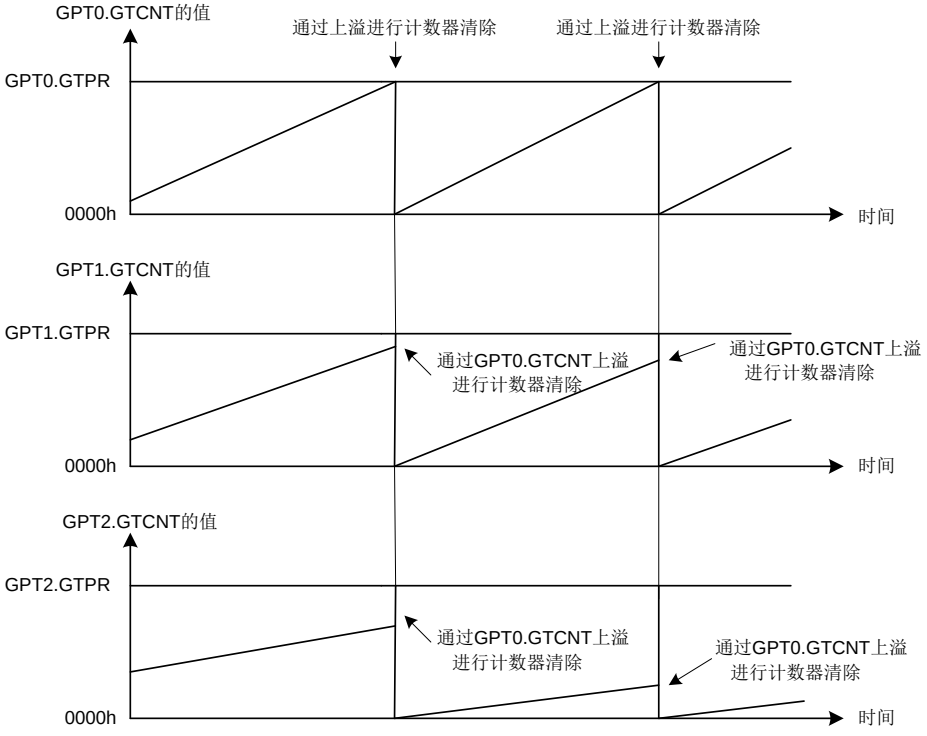


图18-36　同步清除的运行例子（通过 GPT0_GTCNT 的清除源同步清除 GPT1_GTCNT 和 GPT2_GTCNT）

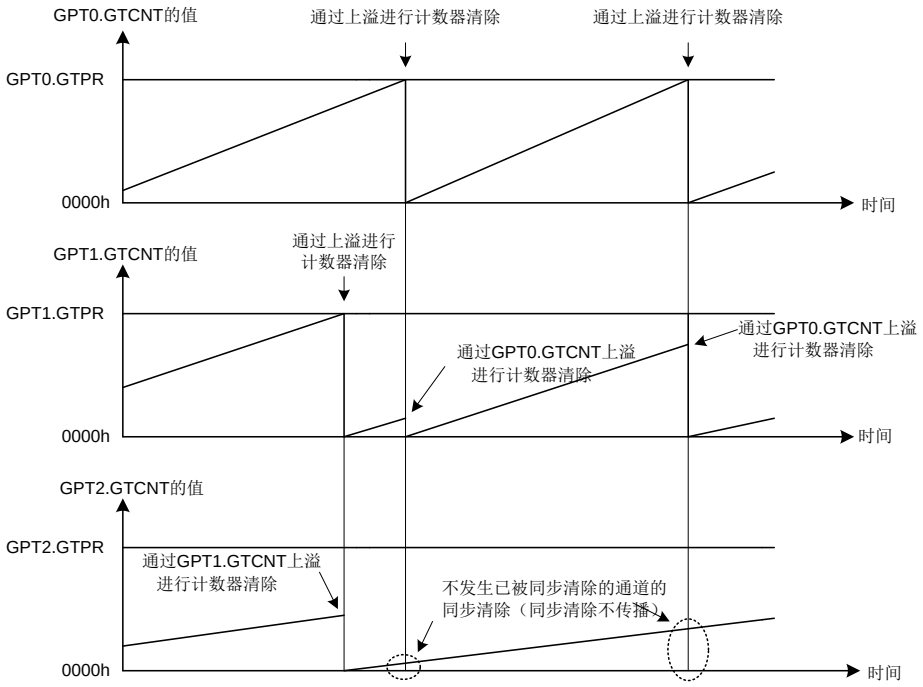


图18-37　同步清除的运行例子
（通过 GPT0_GTCNT 的清除源同步清除 GPT1_GTCNT、通过 GPT1_GTCNT 的清除源同步清除 GPT2_GTCNT）

18.3.11.2　软件同时启动

通过同时将位CSTn@GPT_GTSTR（n=0～2）置’1’，同时开始各模块的计数运行。

通过软件同时启动的运行例子如图18-38所示。

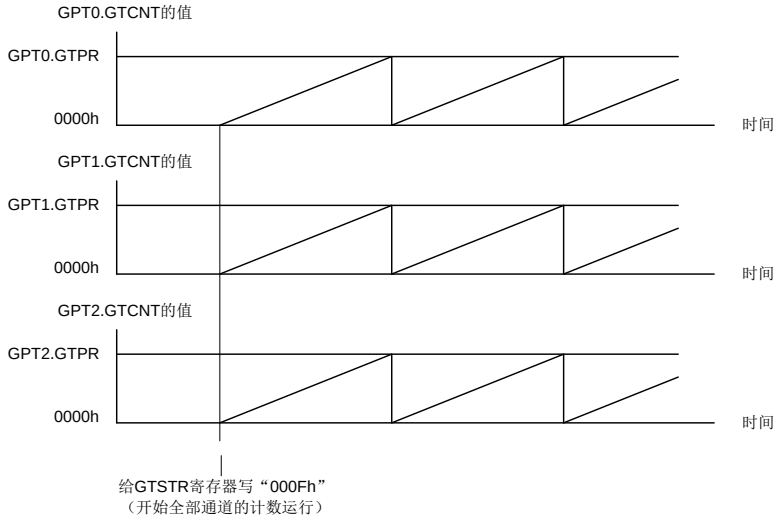


图18-38　通过软件同时启动的运行例子（计数周期（ GTPR 的值）相同时）

18.3.11.3　软件相位启动

在计数开始前，事先设定各模块的GTCNT值，同时将位CSTn@GPT_GTSTR（n=0～2）置’1’，可以在各模块间进行具有相位差的计数运行。

通过软件进行相位启动的运行例子如图18-39所示。

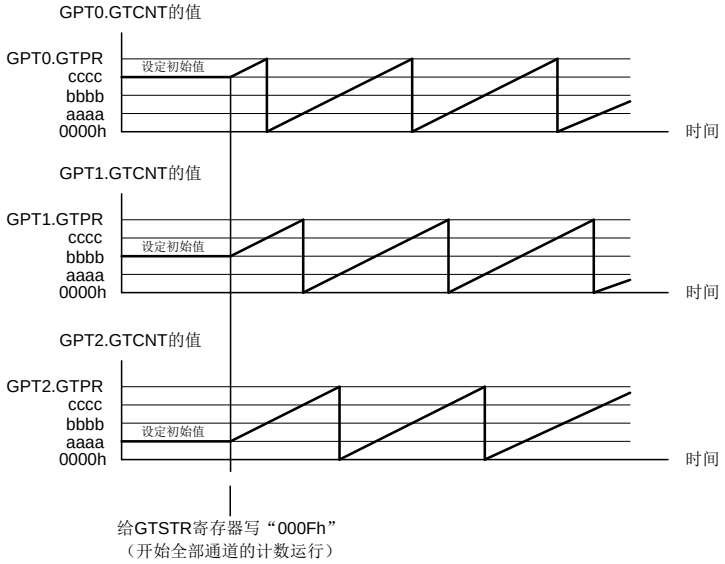


图18-39　通过软件进行相位启动的运行例子（计数周期（ GTPR 的值）相同时）

18.3.11.4　硬件源同时启动

通过SH32F284内部的硬件源GTETRG输入引脚，可以同时启动各模块的计数运行。

通过硬件源同步启动的运行例子如图18-40所示。这是通过外部输入引脚GTETRG开始全部模块的计数运行的例子。

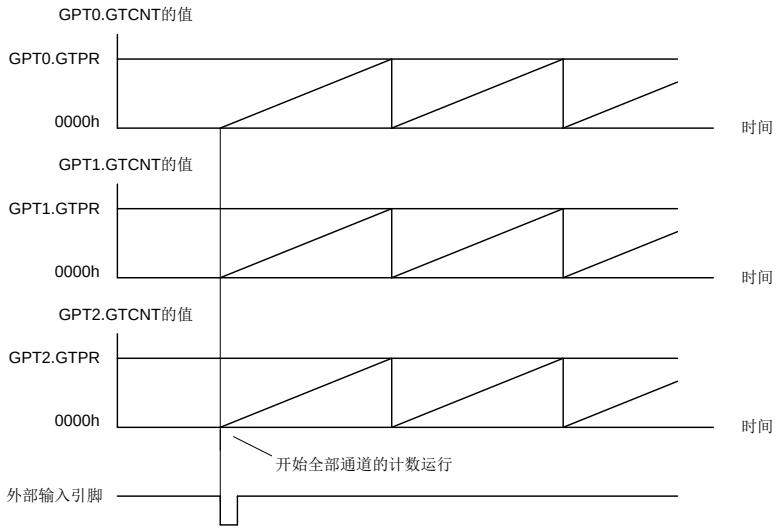


图18-40　通过硬件源同步启动的运行例子（计数周期（ GTPR 的值）相同时）

18.3.11.5　通过硬件源相位启动

通过SH32F284内部的硬件源GTETRG输入引脚，可以在各模块间开始进行具有相位差的计数运行。

通过硬件源进行相位启动的运行例子如图18-41所示。GPT0_GTCNT和GPT1_GTCNT通过GTIOC2A和GTIOC2B的内部输出（输出比较）开始计数运行的例子。

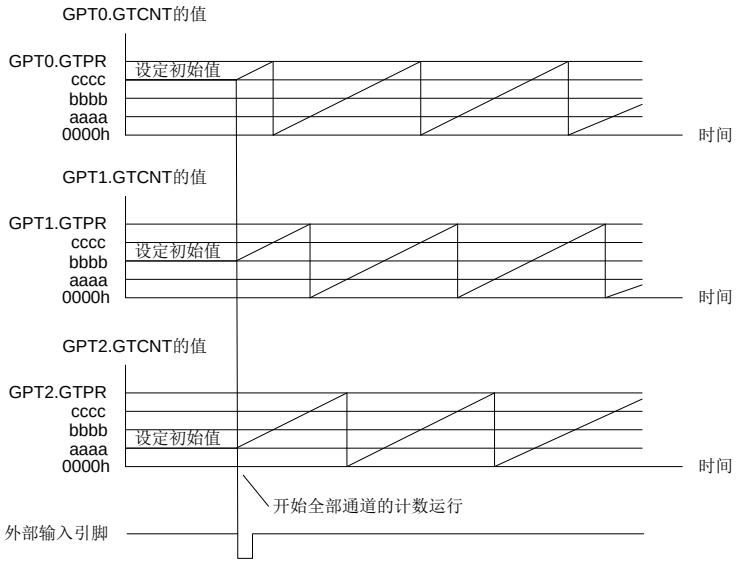


图18-41　　通过硬件源进行相位启动的运行例子（计数周期（ GTPR 的值）相同时）

18.3.12 PWM 输出运行实例

18.3.12.1　同步 PWM 运行

通过模块间的同步运行可以输出最多3个模块的6相联动PWM波形。

图18-42是在锯齿波PWM模式中让全部的模块进行同步运行，并且输出6相PWM波形的例子。在此例子中如下设定GTIOCxA输出，初始输出为Low电平输出、通过GTCCRA寄存器的比较匹配进行High电平输出、在周期结束时进行Low电平输出；并且如下设定GTIOCxB输出，初始输出为Low电平输出、通过GTCCRB寄存器的比较匹配进行High电平输出、在周期结束时进行Low电平输出。

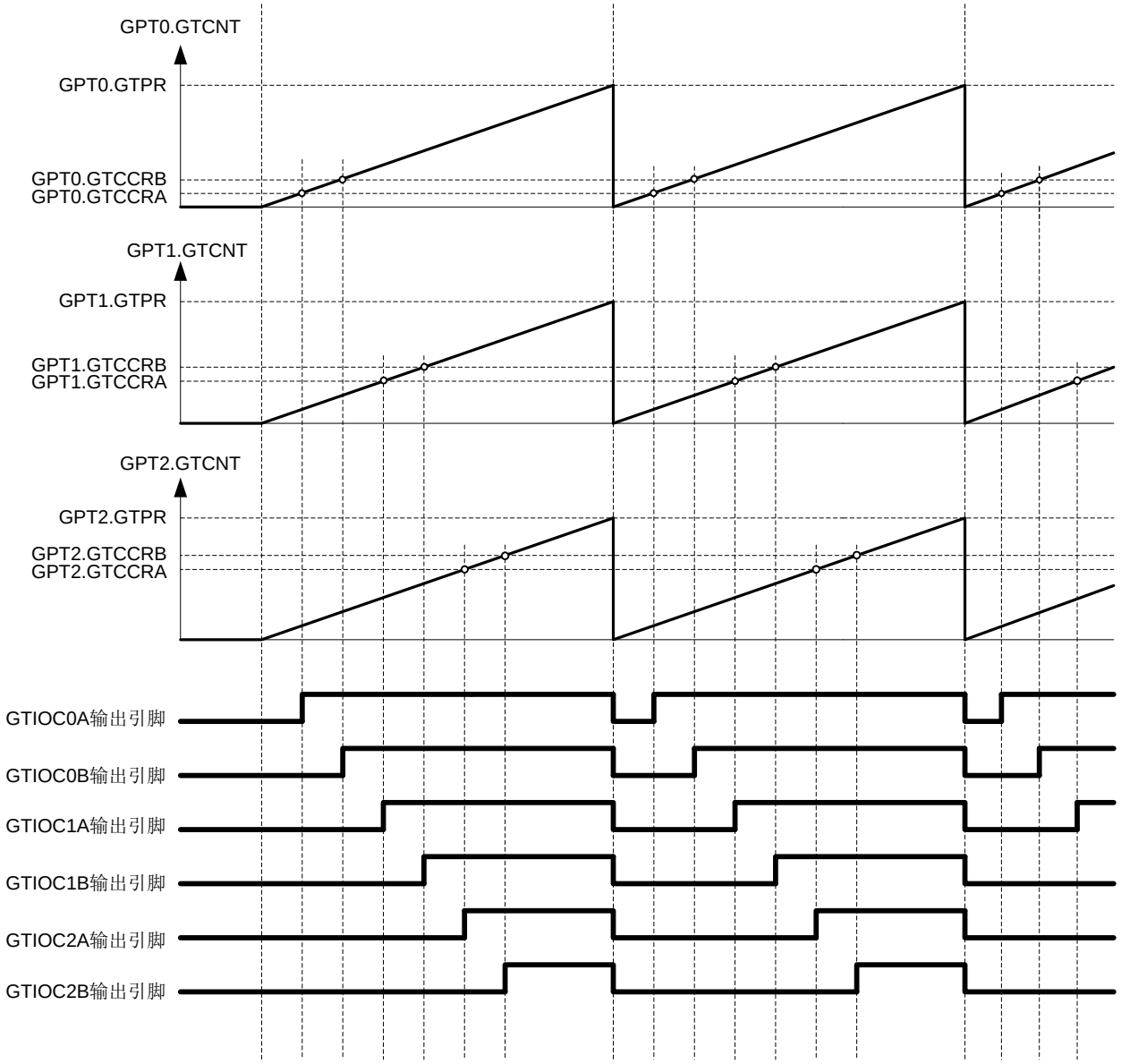


图18-42　同步 PWM 输出的例子

18.3.12.2　锯齿波 3 相互补 PWM 输出

图18-43是在锯齿波PWM模式中让3个模块进行同步运行，并且输出3相互补PWM波形的例子。在此例子中如下设定GTIOCxA输出，初始输出为Low电平输出、通过GTCCRA寄存器的比较匹配进行High电平输出、在周期结束时进行Low电平输出；并且如下设定GTIOCxB输出，初始输出为High电平输出、通过GTCCRB寄存器的比较匹配进行Low电平输出、在周期结束时进行High电平输出。

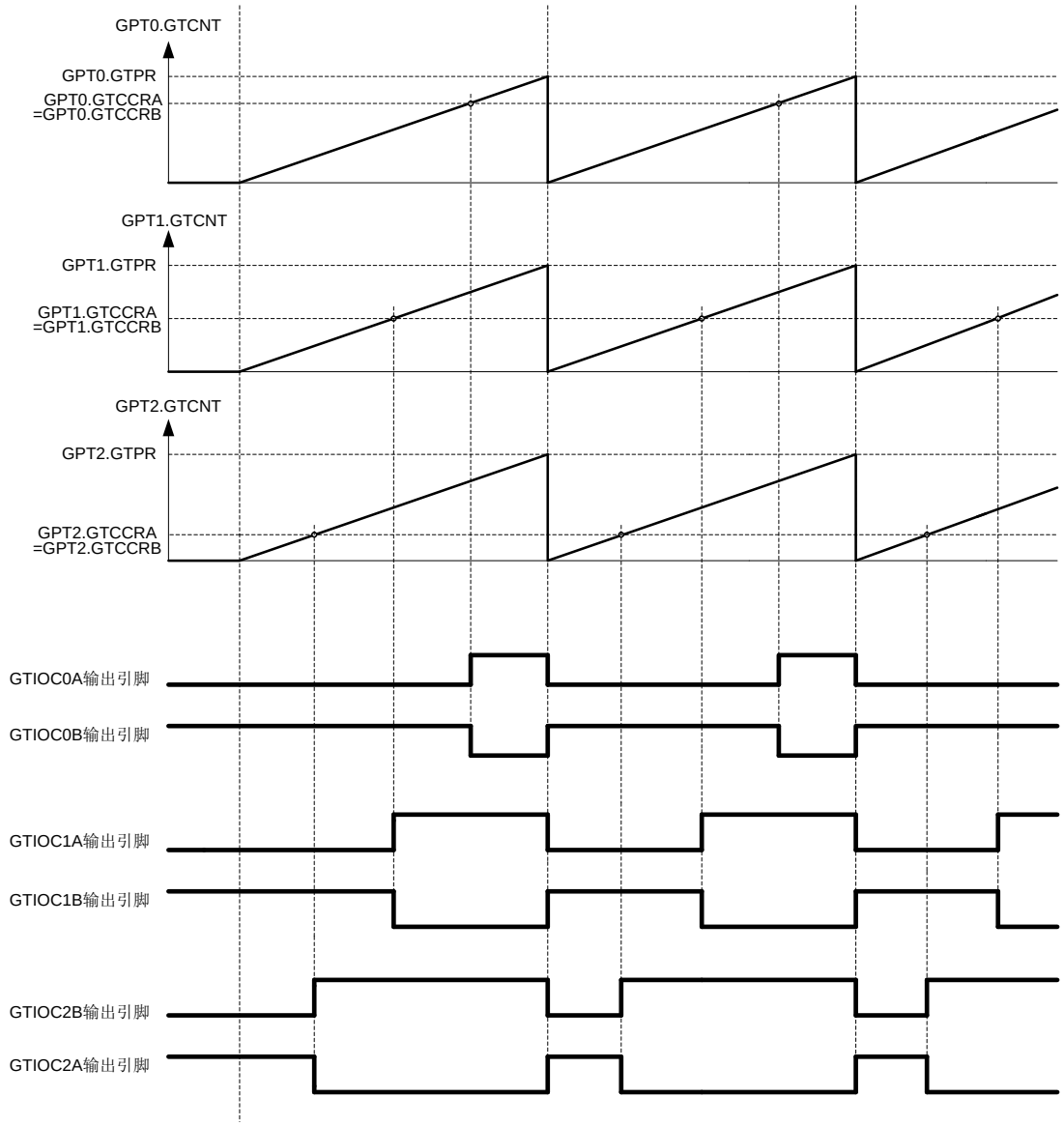


图18-43　　锯齿波 3 相互补 PWM 输出的例子

18.3.12.3　锯齿波 3 相互补 PWM 输出（自动设定死区时间）

图18-44是在自动设定了死区时间的锯齿波单触发脉冲模式中让3个模块进行同步运行，并且输出3相互补PWM波形的例子。在此例子中如下设定GTIOCxA输出，初始输出为Low电平输出、通过GTCCRA寄存器的比较匹配进行交替输出、在周期结束时保持输出；并且如下设定GTIOCxB输出，初始输出为High电平输出、通过GTCCRB寄存器的比较匹配进行交替输出、在周期结束时保持输出。

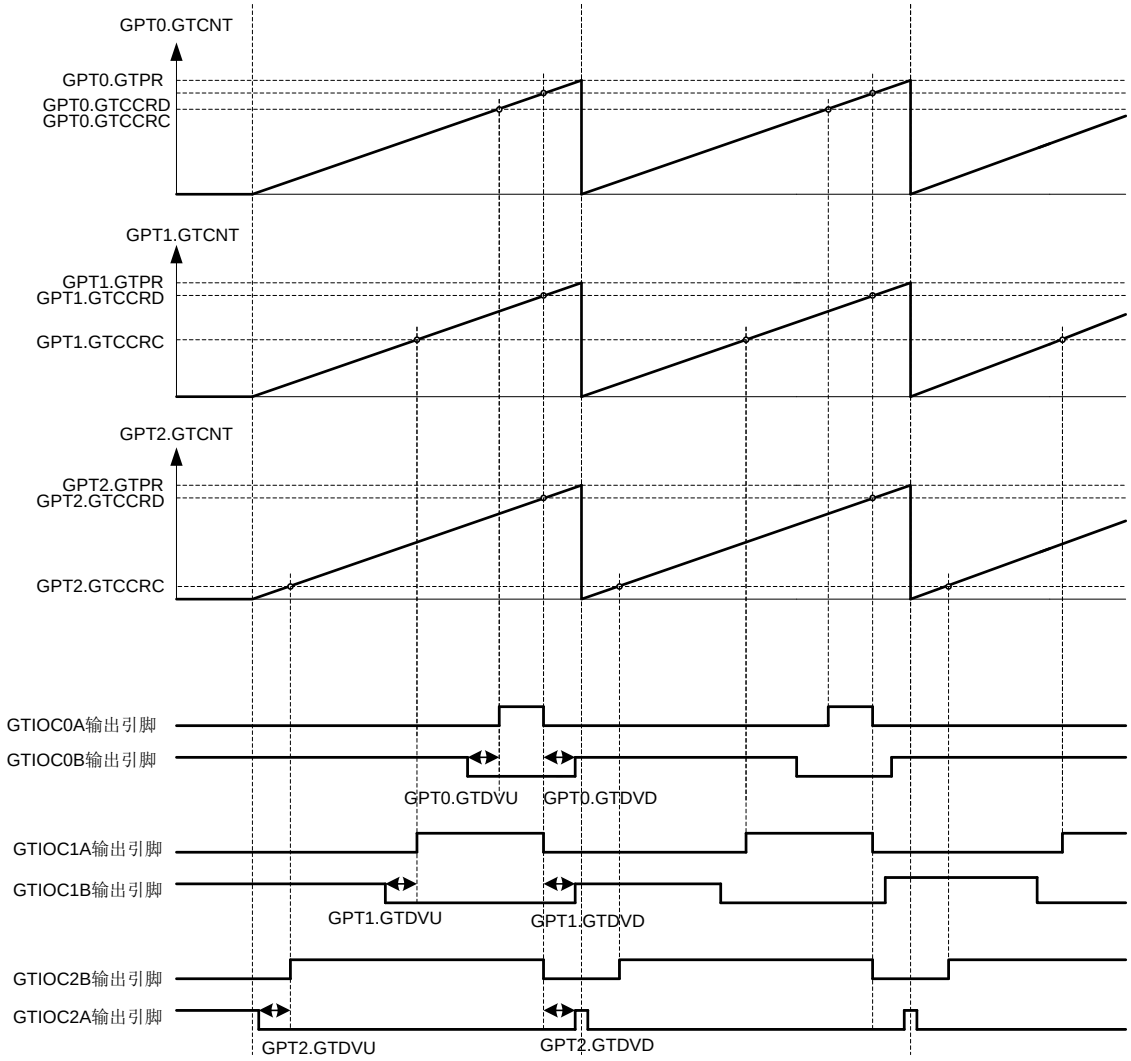


图18-44　　锯齿波 3 相互补 PWM 输出的例子（自动设定死区时间）

18.3.12.4　三角波 3 相互补 PWM 输出

图18-45是在三角波PWM模式1中让3个模块进行同步运行，并且输出3相互补PWM波形的例子。在此例子中如下设定GTIOCxA输出，初始输出为Low电平输出、通过GTCCRA寄存器的比较匹配进行交替输出、在周期结束时保持输出；并且如下设定GTIOCxB输出，初始输出为High电平输出、通过GTCCRB寄存器的比较匹配进行交替输出、在周期结束时保持输出。

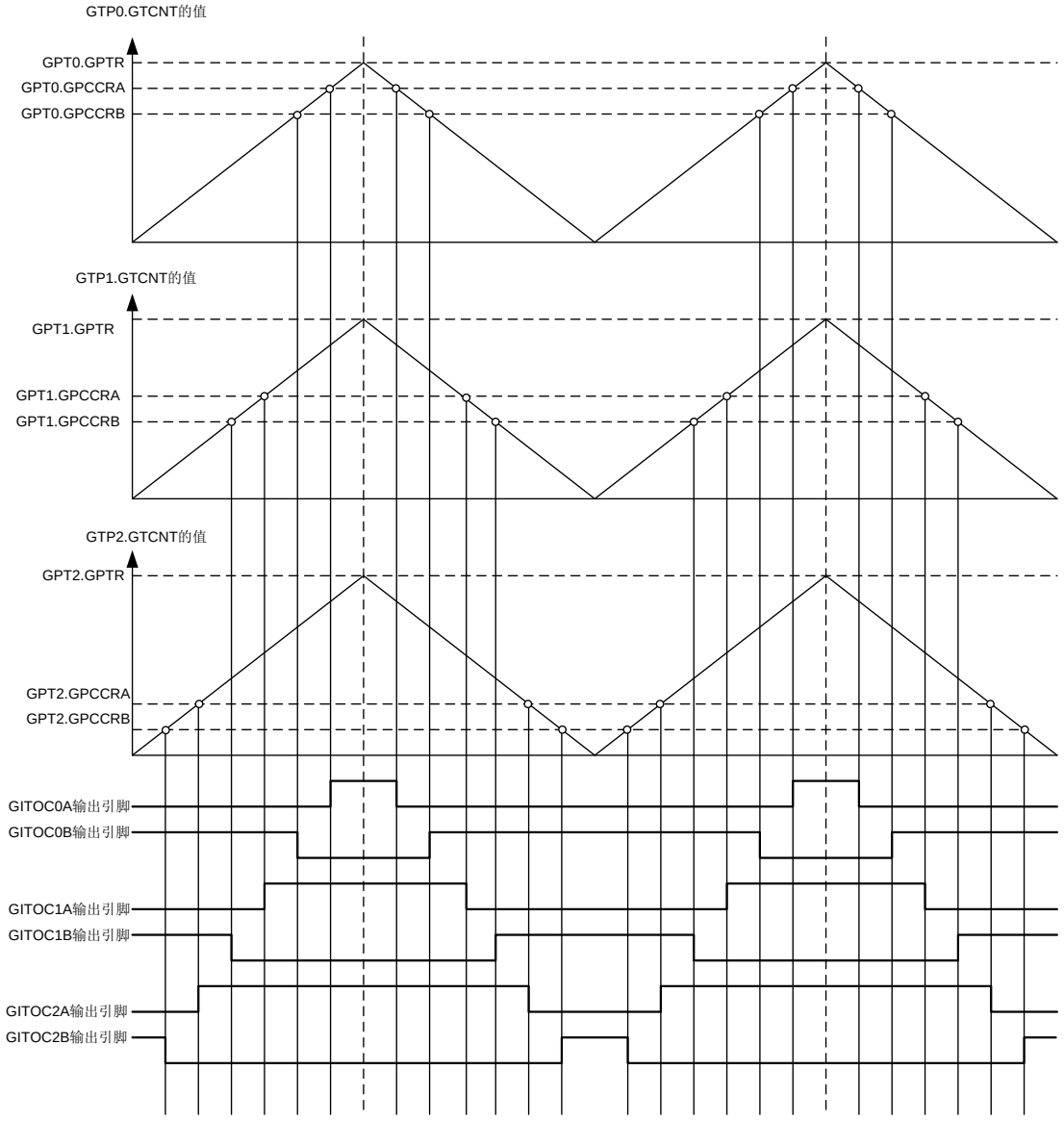


图18-45　　三角波 3 相互补 PWM 输出的例子

18.3.12.5　三角波 3 相互补 PWM 输出（自动设定死区时间）

图18-46是在自动设定了死区时间的三角波PWM模式1中让3个模块进行同步运行，并且输出3相互补PWM波形的例子。在此例子中如下设定GTIOCxA输出，初始输出为Low电平输出、通过GTCCRA寄存器的比较匹配进行交替输出、在周期结束时保持输出；并且如下设定GTIOCxB输出，初始输出为High电平输出、通过GTCCRB寄存器的比较匹配进行交替输出、在周期结束时保持输出。

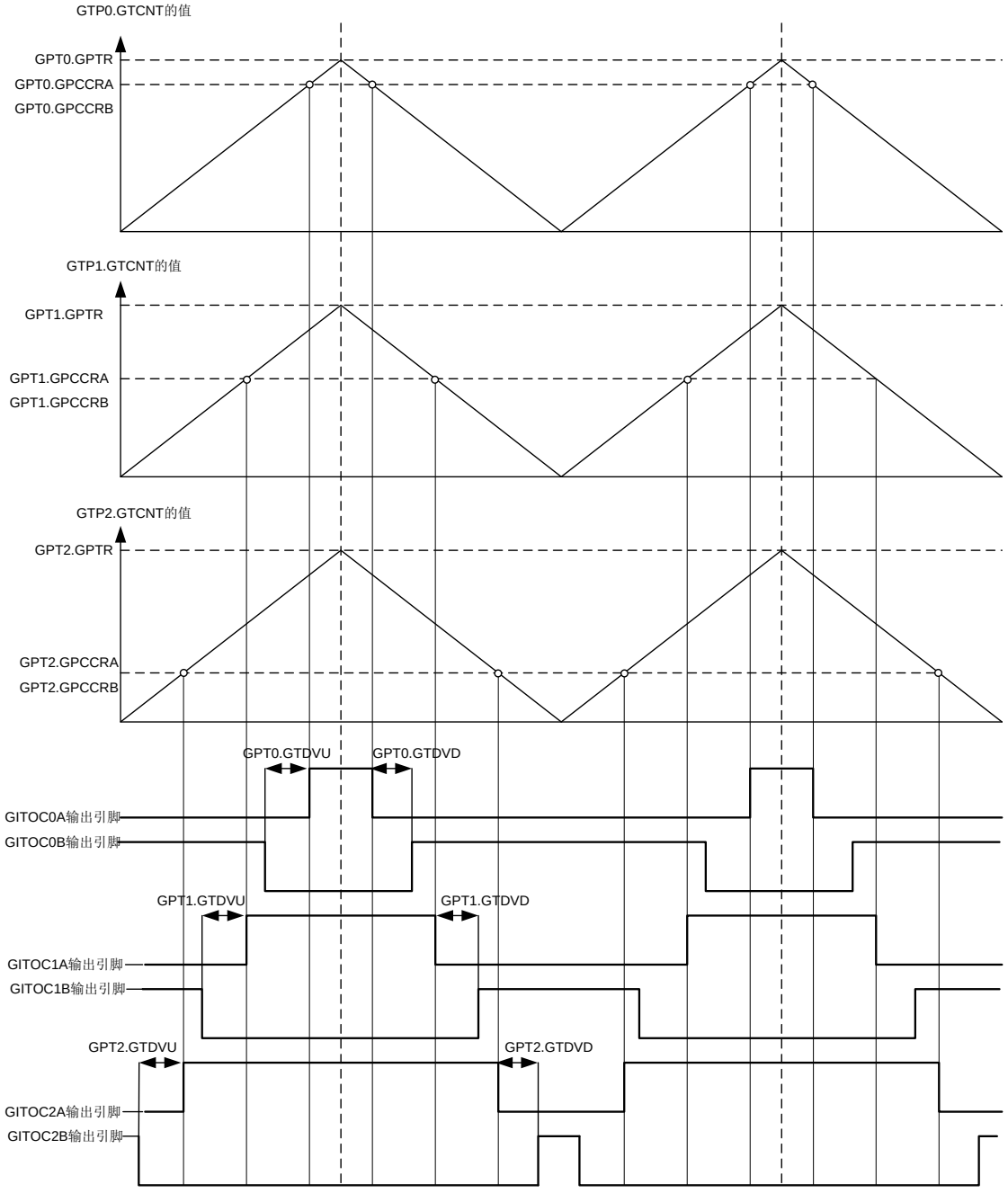


图18-46　三角波 3 相互补 PWM 输出的例子（自动设定死区时间）

18.3.12.6　非对称三角波 3 相互补 PWM 输出（自动设定死区时间）

图18-47是在自动设定了死区时间的三角波PWM模式3中让3个模块进行同步运行，并且输出3相互补PWM波形的例子。在此例子中如下设定GTIOCxA输出，初始输出为Low电平输出、通过GTCCRA寄存器的比较匹配进行交替输出、在周期结束时保持输出；并且如下设定GTIOCxB输出，初始输出为High电平输出、通过GTCCRB寄存器的比较匹配进行交替输出、在周期结束时保持输出。

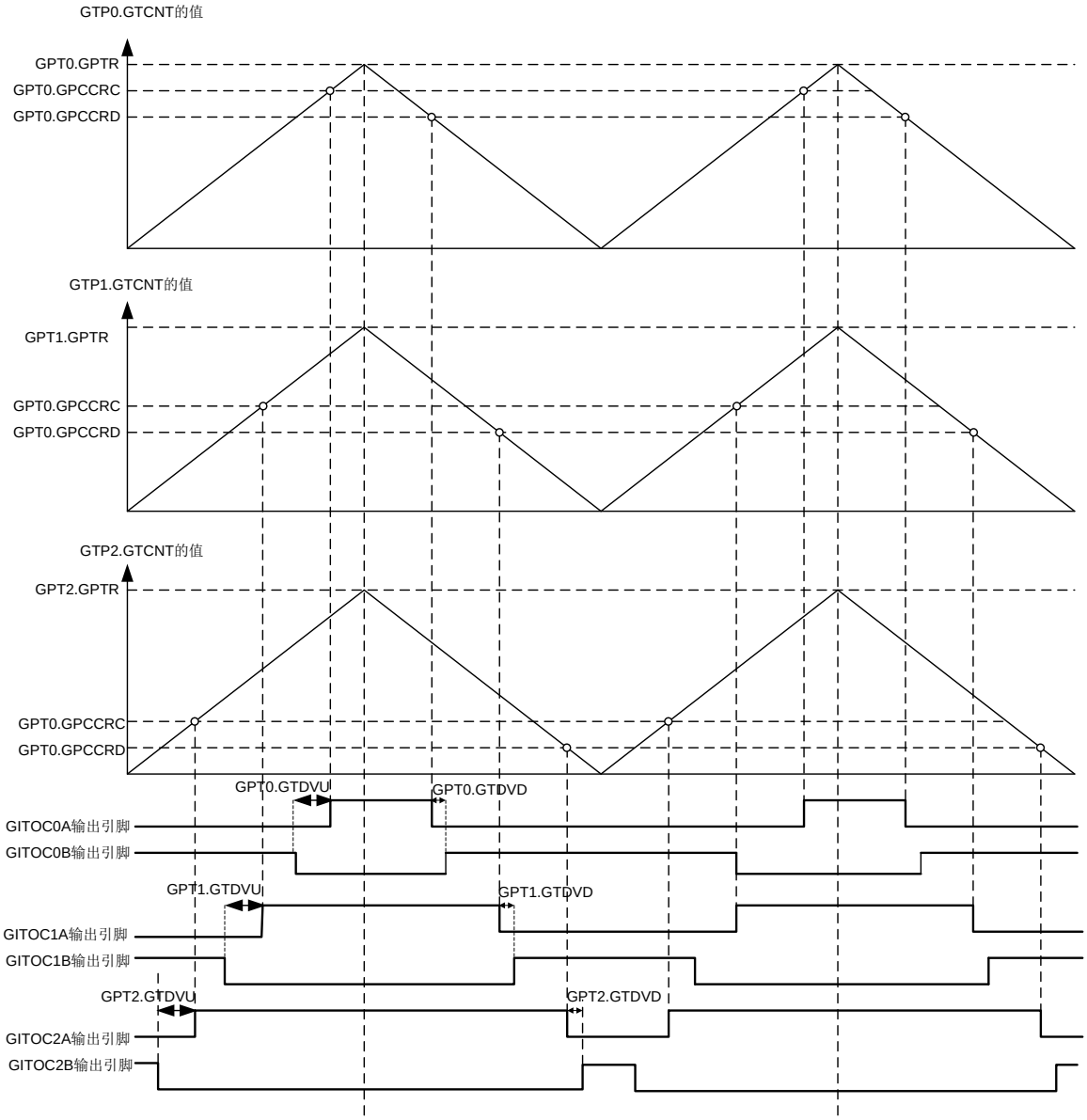


图18-47　　非对称三角波形 3 相互补 PWM 输出的例子（自动设定死区时间）

18.3.13 中断源

18.3.13.1 中断源和优先级

GPT的中断源有2种，分别是GTCCR寄存器的输入捕捉/比较匹配、GTCNT计数器的上溢（GTPR寄存器的比较匹配）/下溢。各中断源分别具有专用的状态标志和产生中断请求的控制位，并且能独立允许或者禁止中断请求的产生。

如果产生中断源，GTST寄存器对应的状态标志就变为“1”。此时，如果GTINTAD寄存器对应的中断请求允许/禁止位为“1”，就请求中断。

通道	中断源	中断名称	中断事件	中断标志位
总体	POE	GTPOE	外部输入引脚GTPOE中断	POEF
		GTETIN	外部输入引脚的下降沿中断	ETINF
		GTETIP	外部输入引脚的上升沿中断	ETIPF
0	GTCIV0	GTCIPO0	GPT0.GTCNT 的上溢（GPT0.GTPR 的比较匹配）	TCFPO
		GTCIPU0	GPT0.GTCNT 的下溢	TCFPU
	GTCIN0	GTCIA0	GPT0.GTCCRA 的输入捕捉/ 比较匹配	TCFA
		GTCIB0	GPT0.GTCCRB 的输入捕捉/ 比较匹配	TCFB
		GTDT0	死区时间错误	DTEF
		GTOSC0	电平异常保护标志	OSF
1	GTCIV1	GTCIPO1	GPT1.GTCNT 的上溢（GPT1.GTPR 的比较匹配）	TCFPO
		GTCIPU1	GPT1.GTCNT 的下溢	TCFPU
	GTCIN1	GTCIA1	GPT1.GTCCRA 的输入捕捉/ 比较匹配	TCFA
		GTCIB1	GPT1.GTCCRB 的输入捕捉/ 比较匹配	TCFB
		GTDT1	死区时间错误	DTEF
		GTOSC1	电平异常保护标志	OSF
2	GTCIV2	GTCIPO2	GPT2.GTCNT 的上溢（GPT2.GTPR 的比较匹配）	TCFPO
		GTCIPU2	GPT2.GTCNT 的下溢	TCFPU
	GTCIN2	GTCIA2	GPT2.GTCCRA 的输入捕捉/ 比较匹配	TCFA
		GTCIB2	GPT2.GTCCRB 的输入捕捉/ 比较匹配	TCFB
		GTDT2	死区时间错误	DTEF
		GTOSC2	电平异常保护标志	OSF

注：整个GPT模块有一个总的中断向量，每个GPT还有两个中断向量，详细的中断入口分配和优先级请查看”中断” 章节。

18.3.13.2 输入捕捉/比较匹配中断

如果发生各模块的GTCCR寄存器的输入捕捉/比较匹配，GTST寄存器的对应状态标志就变为“1”。此时，如果GTINTAD寄存器的对应中断允许位为“1”，就请求中断。

18.3.13.3 上溢/下溢周期中断

可以通过设定各通道的GTPR寄存器，发生决定中断间隔的周期中断。

在锯齿波并且进行递增计数的情况下，如果在GTCNT计数器的值和GTPR寄存器的值相同时（上溢）标志位TCFPO@GPTx_GTST为“1”；在锯齿波并且进行递减计数的情况下，如果GTCNT计数器的值为“0”（下溢），标志位TCFPU@GPTx_GTST就变为“1”。此时，如果将位GTINTPR[1:0]@GPTx_GTINTAD设定为“01b”、“10b”或者“11b”，就请求GTCIV中断。

在三角波的情况下，当GTCNT计数器的值和GTPR寄存器的值相同（波峰），TCFPO标志就变为“1”，当GTCNT计数器的值为“0”（波谷）时，TCFPU标志就变为“1”。此时，如果将GTINTPR[1:0]位的对应位设定为“01b”、“10b”或者“11b”，就请求GTCIV中断。

18.3.13.4 死区时间错误中断

当设置为自动设定死区时间时，如果自动附加后的定时器输出交替点时刻超出定时器周期，标志位DTEF@GPTx_GTST就变为“1”。此时，如果位EINT@GPTx_GTINTAD为“1”，就请求中断。

18.3.14 DMA 启动

通过各模块的中断可以启动DMA。通过设置寄存器GPT_GTETINT和GPTx_GTDMA开启DMA触发功能。

18.3.15 中断、AD 转换开始请求的减少功能

通过设定GTITC寄存器可以减少GTCNT计数器的上溢（GTPR寄存器的比较匹配）/下溢中断（GTCIV）。联动GTCIV中断减少功能能减少其它中断和A/D转换开始请求。但是，死区时间错误中断不能联动GTCIV中断减少功能。中断减少后，对应的状态标志的变化也减少，并且在状态标志为“1”的期间保持中断减少功能的运行。

在三角波中对波峰和波谷进行计数并且减少波峰和波谷时，如果将减少次数设定为奇数，根据减少计数器的开始时序，产生只在波峰或者只在波谷的GTCIV中断请求。只在波峰或者只在波谷的中断请求取决于计数开始时计数递增或者递减方向。

同样，在锯齿波中一边转换计数方向一边对上溢和下溢进行计数并且减少上溢和下溢时，有可能不产生只在发生上溢或者只在发生下溢时的GTCIV中断。在锯齿波中一边转换计数方向一边对上溢和下溢进行计数并且减少上溢和下溢时，如果使用只在发生上溢或者只在发生下溢时产生的GTCIV中断，就必须在充分检测中断状态后再使用。

必须在暂时解除中断联动功能（IVTC[1:0]@GPTx_GTITC=00b）后，才能更改减少次数。

减少功能的运行例子如图18-48～图18-53所示。

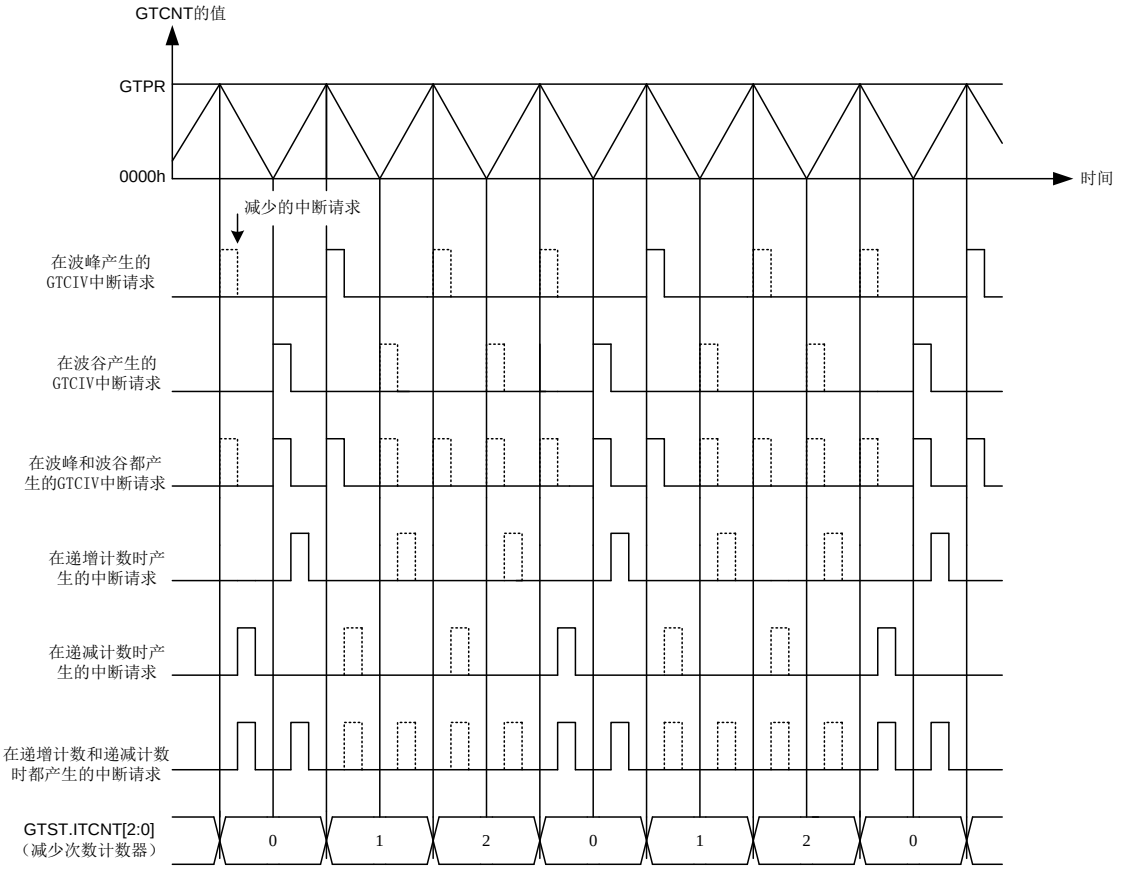


图18-48 中断减少功能的运行例子（三角波、对波峰进行计数并且减少波峰、减少次数为 2 次）

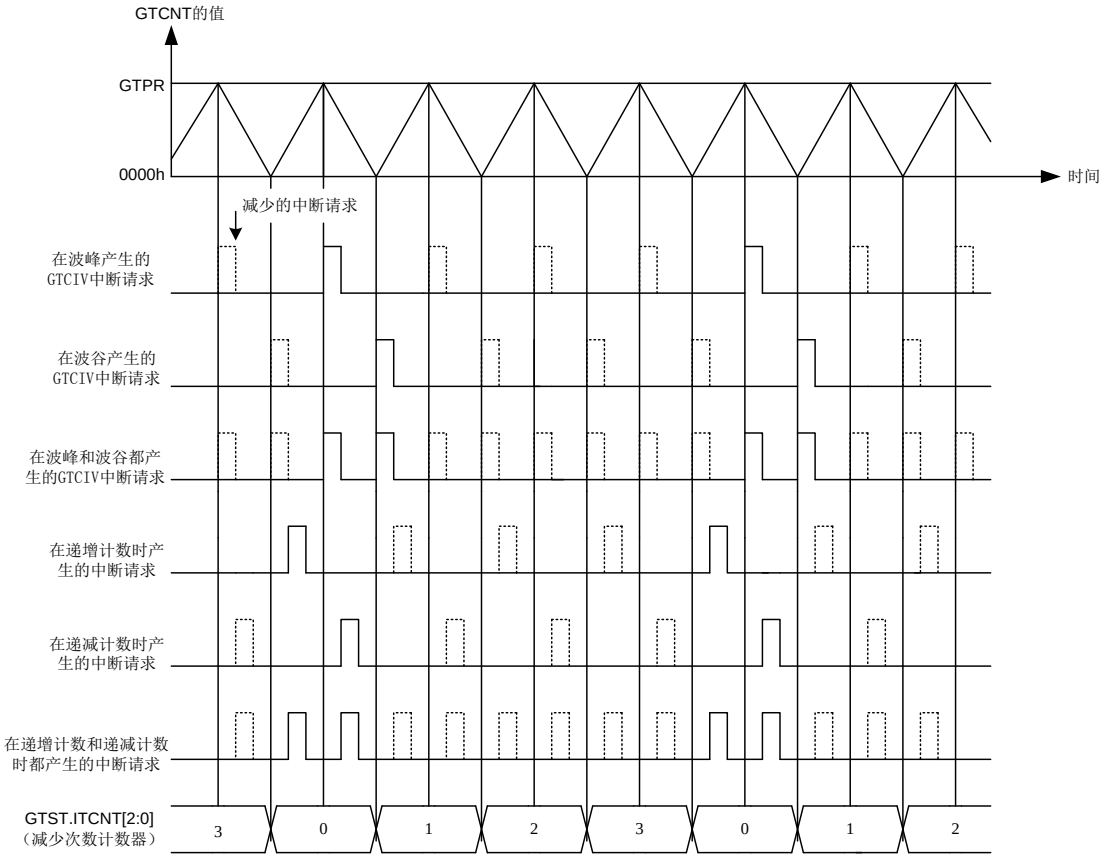


图18-49　中断减少功能的运行例子（三角波、对波谷进行计数并且减少波谷、减少次数为 3 次）

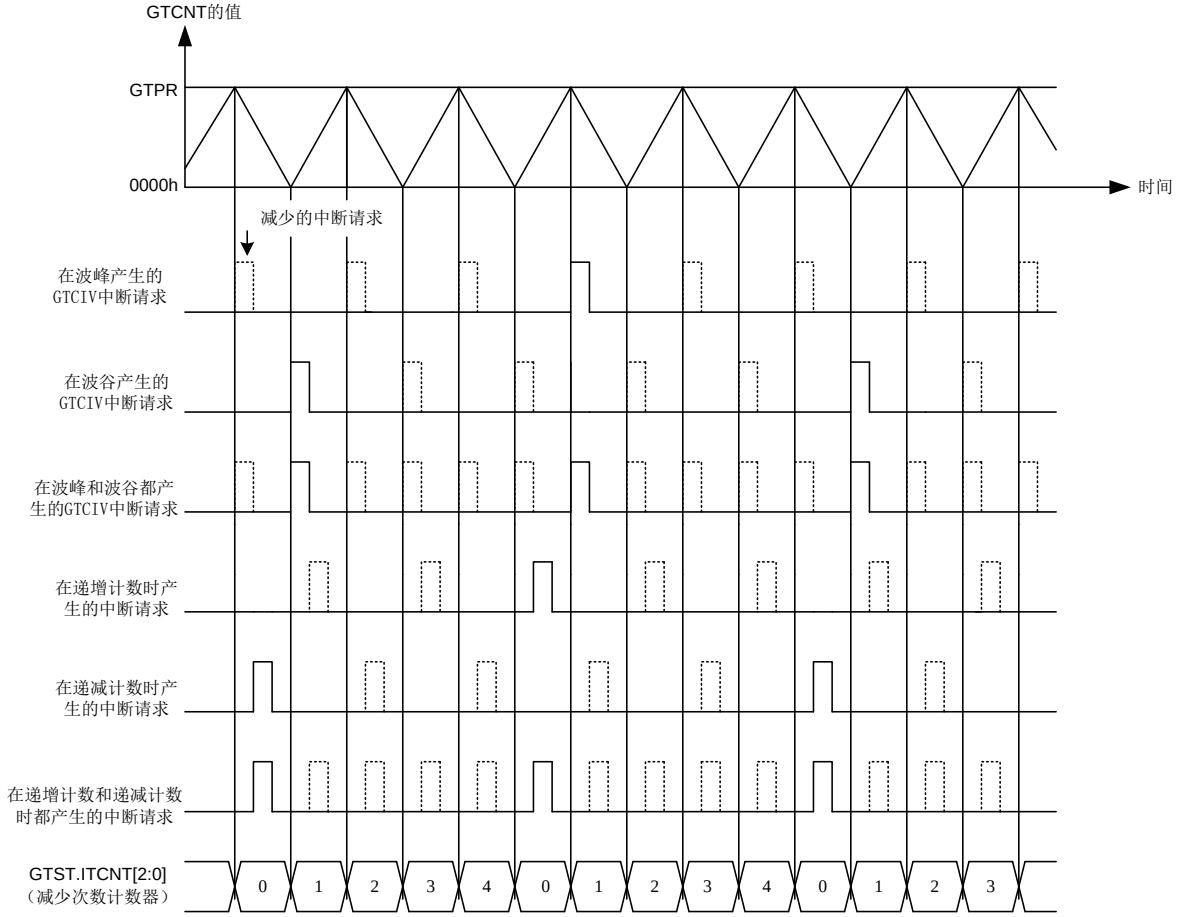


图18-50　　中断减少功能的运行例子
（三角波、对波谷和波峰进行计数并且减少波谷和波峰、减少次数为 4 次）

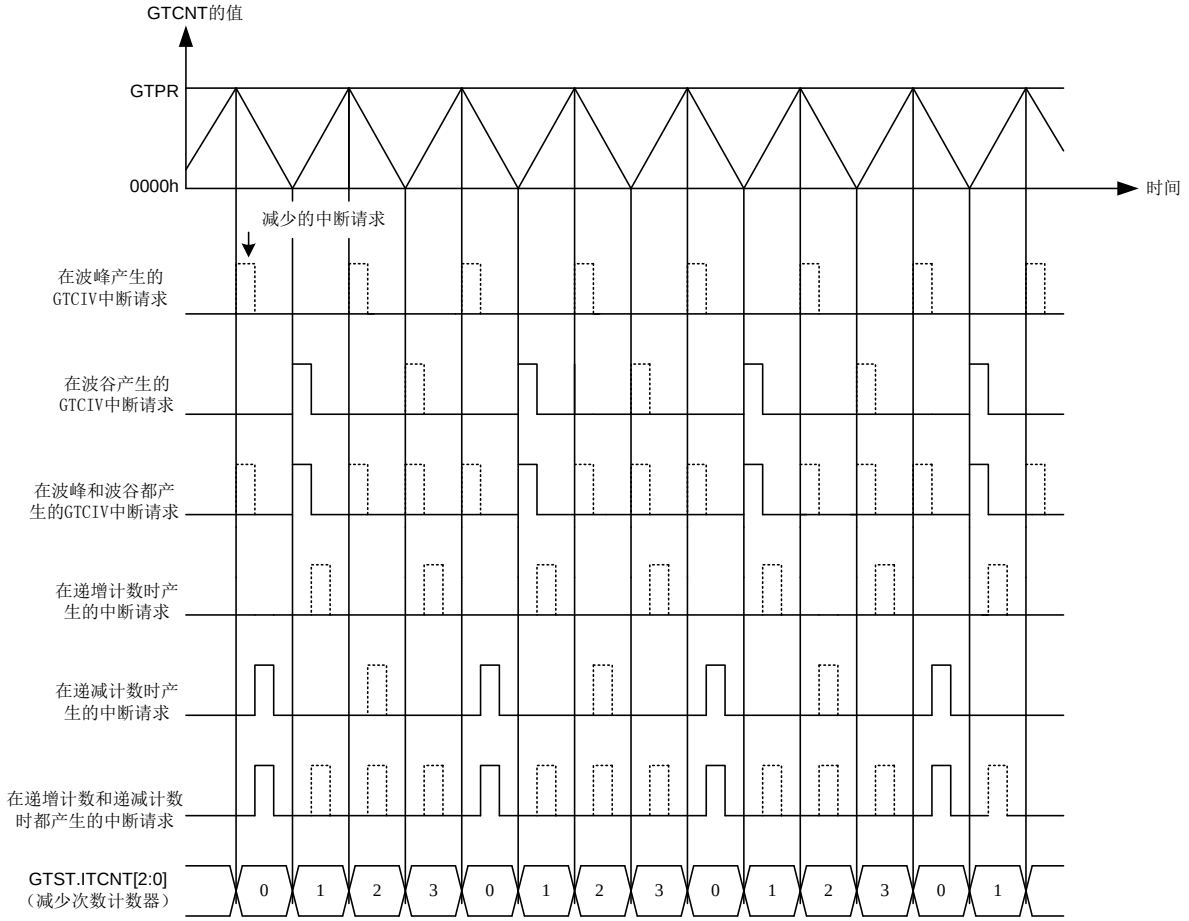


图18-51　　中断减少功能的运行例子

（三角波、对波谷和波峰进行计数并且减少波谷和波峰、减少次数为 3 次、在初始计数为递增计数时开始减少）

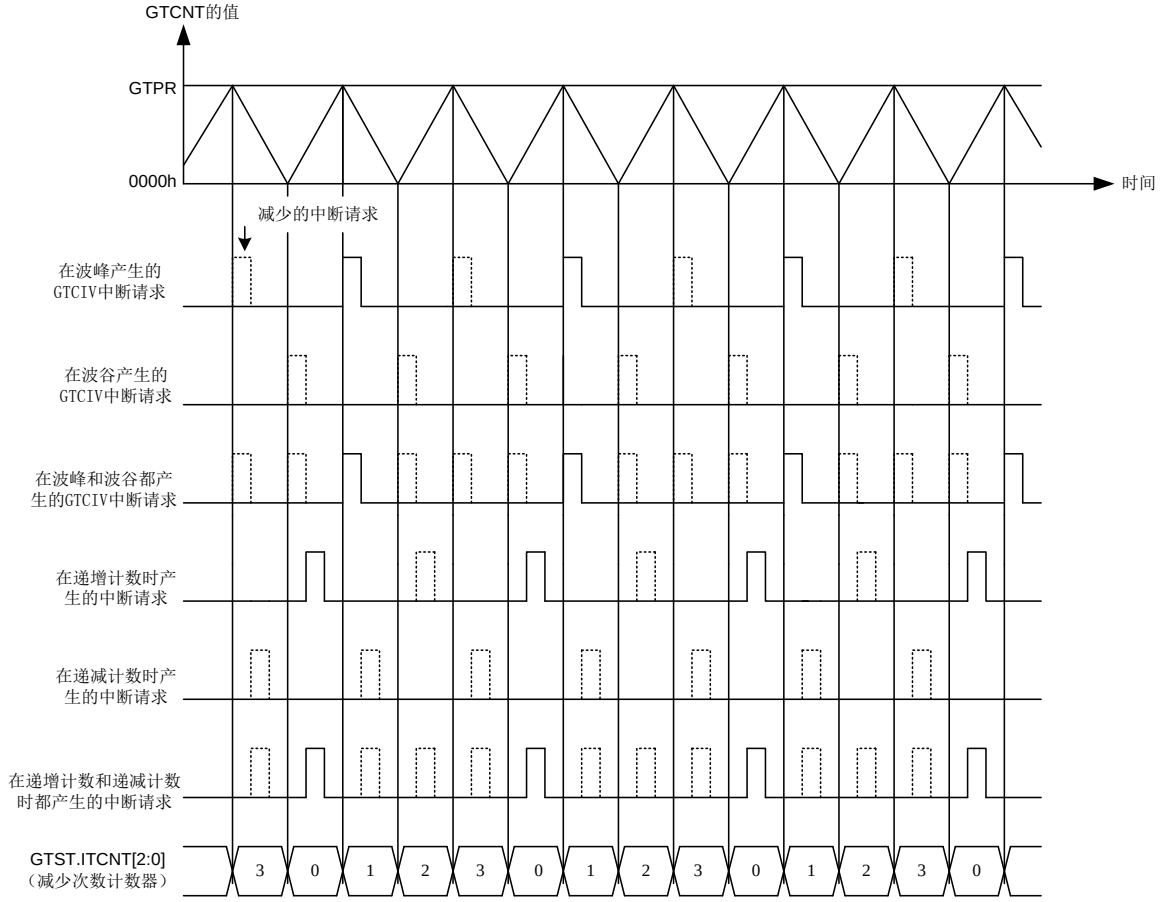


图18-52　　中断减少功能的运行例子

（三角波、对波谷和波峰进行计数并且减少波谷和波峰、减少次数为 3 次、在初始计数为递减计数时开始减少）

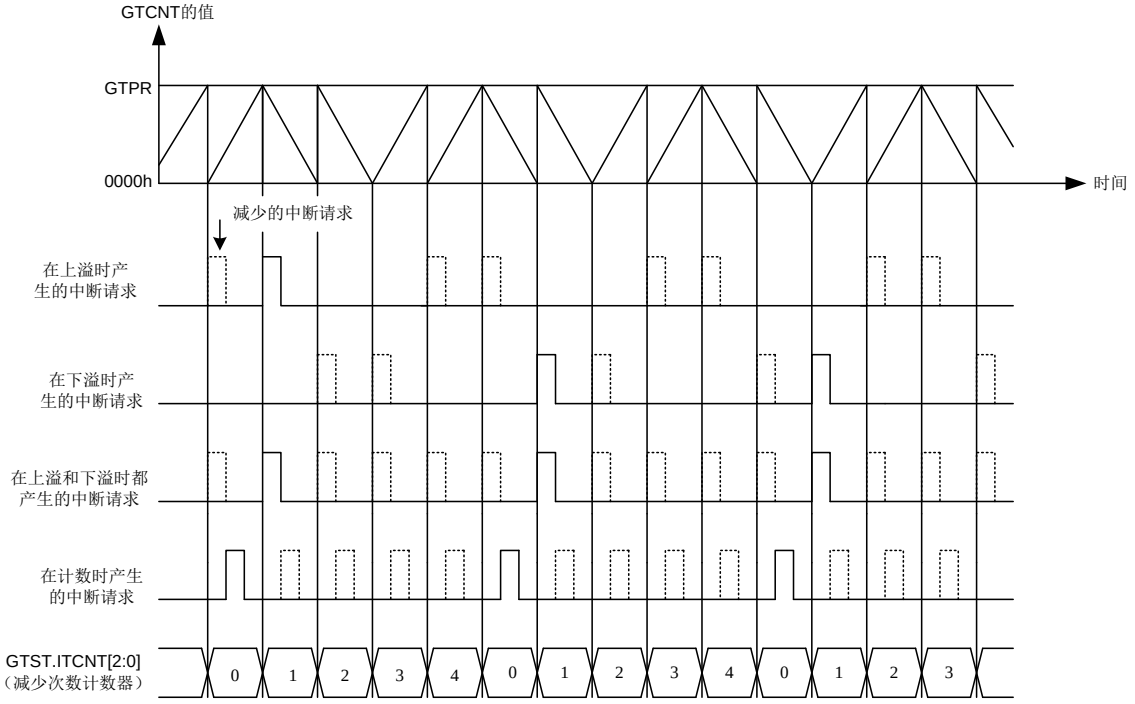


图18-53　　中断减少功能的运行例子

（锯齿波、一边转换计数方向一边对上溢和下溢进行计数并且减少上溢和下溢、减少次数为 4 次）

18.3.16 AD 转换开始请求

通过GTCNT计数器和GTADTRA寄存器、GTADTRB寄存器的比较匹配可以产生A/D转换开始请求。递增计数、递减计数、递增/递减计数均可以产生A/D转换开始请求。

A/D转换开始请求的运行例子如图18-54。

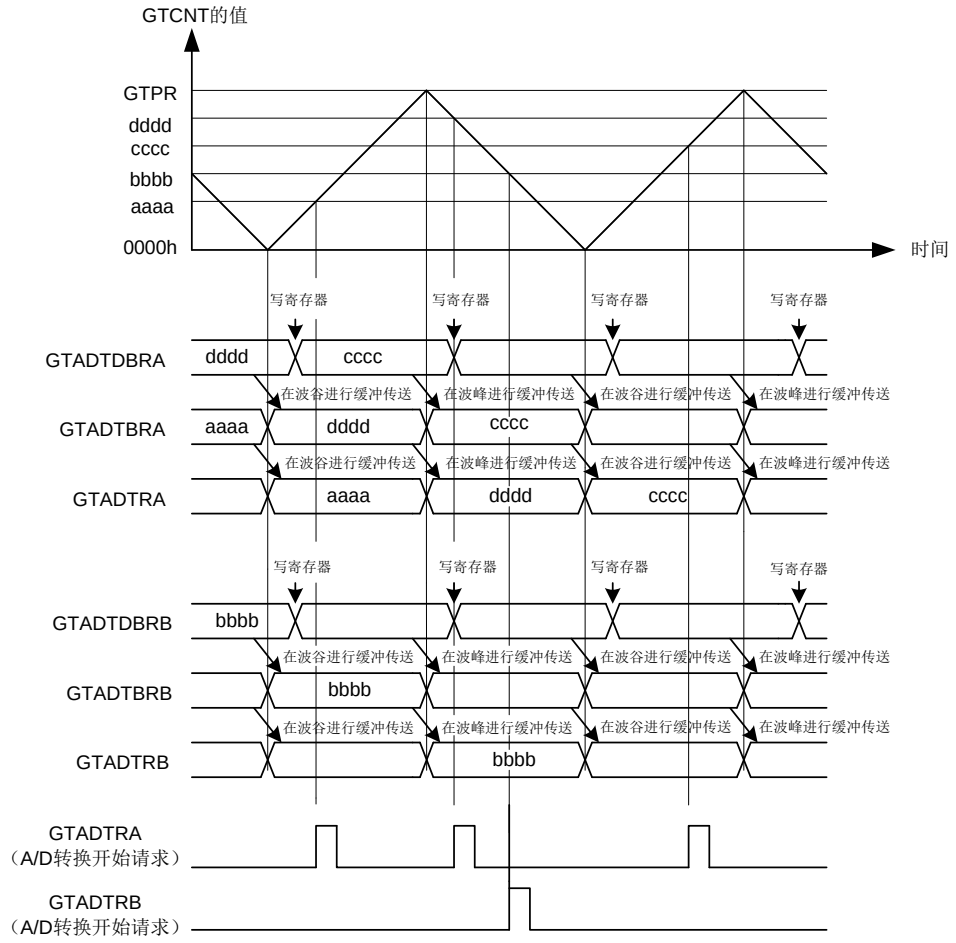


图18-54　A/D 转换开始请求的运行例子

（三角波、双缓冲运行、在波谷和波峰进行缓冲传送、 GTADTRA0 在递增计数和递减计数时都产生 A/D 转换开始请求，GTADTRB0 在递减计数时产生 A/D 转换开始请求）

18.3.17 保护功能

18.3.17.1　寄存器的写保护

为了防止误写寄存器，整个GPT模块中（包含3个GPT共有的寄存器），除了GPTx_GTST，其它寄存器都要通过设定位WP[7:0]@GPTx_GTWP给每个模块设定禁止写寄存器。

另外整个GPT共有的寄存器GTSTR、GTETINT和GTPOECR也有写入限制条件，只有在GTPRWEN = 0x33CC时才允许软件修改，否则修改无效。

18.3.17.2　缓冲运行的禁止

当缓冲寄存器的写操作赶不上缓冲传送时序时，通过设定GTBDR寄存器可以禁止缓冲运行。如果在写缓冲寄存器前将GTBDR寄存器的对应位置“1”（禁止缓冲运行），在全部缓冲寄存器的写操作结束后将GTBDR寄存器的对应位置“0”（允许缓冲运行），即使在写缓冲寄存器的过程中产生缓冲传送条件，也将暂时禁止缓冲传送。

缓冲运行的禁止例子如图18-55所示。

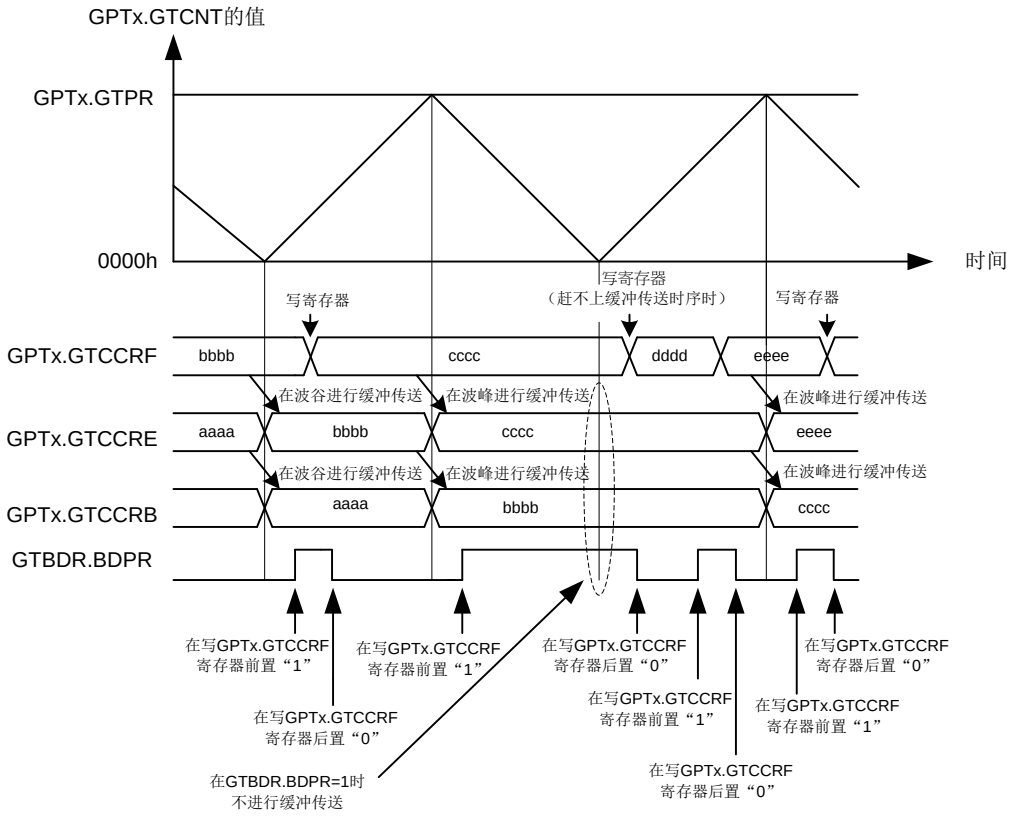


图18-55　缓冲运行的禁止例子（三角波、双缓冲运行、在波谷和波峰都进行缓冲传送）

18.3.17.3　通过外部输入引脚进行 GTIOC 引脚输出的高阻抗控制

作为系统发生异常时的保护措施，通过GTPOE引脚，可以进行3个GPT的GTIOA/B引脚输出的高阻抗控制。

注1：没有使能位，把I/O设置作为使能位。

18.3.17.4　通过比较器 1 进行 GTIOC 引脚输出的高阻抗控制

当HIZ@GPTx_GTCR置为1，如果比较器1的C1IF@AMOC_CMPINTF标志位被置起，则对应GPT的GTIOC引脚输出被置为高阻抗。

注：比较器1的标志位设置请见AMOC章节。

18.3.17.5　GPT 输出电平异常保护

首先设定GPT保护的有效电平，在GPT输出使能寄存器的允许位CSTn@GPT_GTSTR(n=0~2) 开启后，当上下桥同时输出OLSGA@GPTx_GTOSCR和OLSGB@GPTx_GTOSCR设定的有效电平时，置起异常标志位，同时GPT输出高阻态。如下图，设定有效电平都为低电平，输出保护后输出高阻态。

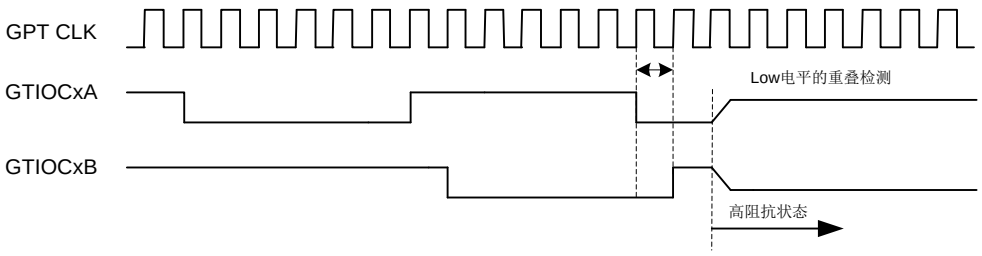


图18-56　PWM 输出电平比较

18.3.17.6　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。

注意：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。

18.3.17.7 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。

18.3.18 引脚重映射功能

GPT0~2的6个引脚和GTPOE引脚都可以重映射到不同的IO口，可实现在不改变硬件的条件下，使用不同模块（GPT/MCM）来驱动电机，

详细的重映射说明，见IO复用章节。

18.3.19 输出引脚的初始化方法

18.3.19.1 复位后的引脚设定

GPT的寄存器在复位时被初始化。设定GTIOR寄存器，并且在将GPT功能输出到外部引脚后，使能GPT模块开始计数。

设置运行举例：

GTIOA[5:0]@GPTx_GTIOR的设定：初始输出为低电平、在周期结束时保持输出、在比较匹配时交替输出

GTIOB[5:0]@GPTx_GTIOR的设定：初始输出为高电平、在周期结束时保持输出、在比较匹配时交替输出

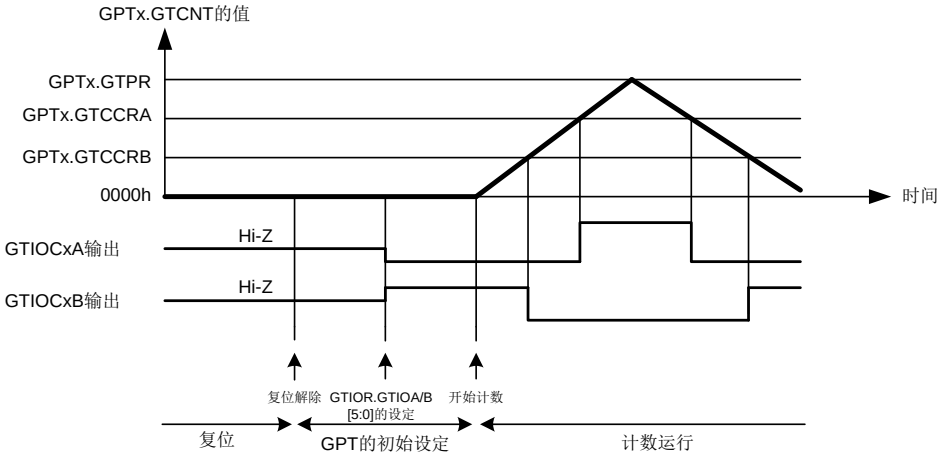


图18-57 复位后的引脚设定例子

18.3.19.2 复位后的引脚设定

从GPT的运行过程中发生异常等开始到引脚初始化前的引脚处理，有以下方法：

1. 将GTIOR寄存器的OAHLD位和OBHLD位置“1”，在停止计数时保持输出。
2. 将GTIOR寄存器的OAHLD位和OBHLD位置“0”，给GTIOR寄存器的OADFLT位和OBDFLT位设定任意的输出值，在停止计数时输出任意的值。
3. 使用POE功能，将输出置为高阻抗。

在自动设定死区时间时，必须在停止计数后将位TDE@GPTx_GTDTCR置“0”。

在停止计数时，只有因GPT以外的外部源变化的寄存器发生变化，其它寄存器不变。在重新开始计数后，持续运行。

在停止计数时，只有在将各寄存器初始化后才能重新开始计数。

18.3.20 注意事项

18.3.20.1 复位后的引脚设定模块停止功能的设定

能通过模块停止控制寄存器设定禁止或者运行GPT的运行。在设定为初始值时，GPT停止运行。能通过解除模块停止状态存取寄存器。详细内容请参照“低功耗功能”。

18.3.20.2 比较匹配运行时的GTCCRn 寄存器的设定（n=A、B、C、D、E、F）

GPT的寄存器在复位时被初始化。设定GTIOR寄存器，并且在将GPT功能输出到外部引脚后，开始计数。

■ 在三角波PWM模式中自动设定死区时间的情况

必须将GTCCRA寄存器设定在GTDVU＜GTCCRA、GTDVD＜GTCCRA、GTCCRA＜GTPR的范围内。

如果在计数运行时设定GTCCRA=0或者GTCCRA≥GTPR，输出保护功能就运行。

必须在将GTCCRA寄存器设定在0＜GTCCRA＜GTPR的范围内的状态下，开始计数。如果在设定在0＜GTCCRA＜GTPR的外围外的状态下开始计数，输出保护功能就无法正常运行。

详细内容请参照“18.3.18.4 GTIOC引脚输出的输出保护功能”。

■ 在三角波PWM模式中不自动设定死区时间的情况

必须将GTCCRA寄存器设定在0＜GTCCRA＜GTPR的范围内。如果设定为GTCCRA=0或者GTCCRA=GTPR，只有在GTCCRA=0或者GTCCRA=GTPR成立时在周期内发生比较匹配。另外，如果设定为GTCCRA＞GTPR，就不发生比较匹配。

同样，必须将GTCCRB寄存器设定在0＜GTCCRB＜GTPR的范围内。如果设定为GTCCRB=0或者GTCCRB=GTPR，只有在GTCCRB=0或者GTCCRB=GTPR成立时在周期内发生比较匹配。另外，如果设定为GTCCRB＞GTPR，就不发生比较匹配。

■ **在锯齿波PWM模式中自动设定死区时间的情况**

GTCCRC寄存器和GTCCRD寄存器必须满足以下的限制。否则有可能无法获得确保了死区时间的正常输出波形。

递增计数时：GTCCRC＜GTCCRD、GTCCRC＞GTDVU、GTCCRD＜GTPR－GTDVD

递减计数时：GTCCRC＞GTCCRD、GTCCRC＜GTPR－GTDVU、GTCCRD＞GTDVD

同样，GTCCRE寄存器和GTCCRF寄存器必须满足以下的限制。否则有可能无法获得确保了死区时间的正常输出波形。

递增计数时：GTCCRE＜GTCCRF、GTCCRE＞GTDVU、GTCCRF＜GTPR－GTDVD

递减计数时：GTCCRE＞GTCCRF、GTCCRE＜GTPR－GTDVU、GTCCRF＞GTDVD

■ **在锯齿波单触发脉冲模式中自动设定死区时间的情况**

GTCCRC寄存器和GTCCRD寄存器必须满足以下的限制。否则比较匹配就不发生2次，也无法获得脉冲输出。

递增计数时：0＜GTCCRC＜GTCCRD＜GTPR

递减计数时：GTPR＞GTCCRC＞GTCCRD＞0

同样，GTCCRE寄存器和GTCCRF寄存器必须满足以下的限制。否则比较匹配就不发生2次，也无法获得脉冲输出。

递增计数时：0＜GTCCRE＜GTCCRF＜GTPR

递减计数时：GTPR＞GTCCRE＞GTCCRF＞0

■ **在锯齿波PWM模式的情况**

必须将GTCCRA寄存器设定在0＜GTCCRA＜GTPR的范围内。如果设定为GTCCRA=0或者GTCCRA=GTPR，只有在GTCCRA=0或者GTCCRA=GTPR成立时在周期内发生比较匹配。另外，如果设定为GTCCRA＞GTPR，就不发生比较匹配。

同样，必须将GTCCRB寄存器设定在0＜GTCCRB＜GTPR的范围内。如果设定为GTCCRB=0或者GTCCRB=GTPR，只有在GTCCRB=0或者GTCCRB=GTPR成立时在周期内发生比较匹配。另外，如果设定为GTCCRB＞GTPR，就不发生比较匹配。

18.3.20.3 安全停止定时器的方法

当通过写GTSTR寄存器来停止定时器与GPT的比较匹配中断时序发生竞争时，有可能在写GTSTR寄存器后发生中断。

如果按照以下的步骤停止定时器，就能安全的使定时器停止运行，并且在停止后不发生比较匹配中断。

（1）通过GPT的中断输出设定寄存器（GTINTAD）禁止中断请求。

（2）将GTSTR寄存器的CSTn位置“0”。

18.4 寄存器

GPT 模块寄存器列表 (基地址:0x0x4004 2000)

地址	寄存器名	说明
0x4004 2000	GTSTR	GPT软件启动寄存器
0x4004 2004	GTETINT	GPT外部触发输入中断寄存器
0x4004 2008	GTPOECR	GPT端口输出允许控制寄存器
0x4004 201C	GTPRWEN	GPT寄存器修改和重载控制寄存器
0x4004 2020	GTINTF	GPT中断标志和清除寄存器

18.4.1 GPT 软件启动寄存器 (GPT_GTSTR)

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												CST3	CST2	CST1	CST0
-												*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 ~ 3	保留	-
2	CST2	**GPT2.GTCNT 计数开始位** 0：停止计数 1：开始计数
1	CST1	**GPT1.GTCNT 计数开始位** 0：停止计数 1：开始计数
0	CST0	**GPT0.GTCNT 计数开始位** 0：停止计数 1：开始计数

注：也能通过GTHCR寄存器设定由硬件源引起的计数器的计数/停止计数。如果通过硬件源启动计数，此位就自动变为“1”，如果通过硬件源停止计数，此位就自动变为“0”。

18.4.2 GPT 外部触发输入中断寄存器 (GPT_GTETINT)

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								PS[1:0]		SN[1:0]		ETNDMA	ETPDMA	ETINEN	ETIPEN
-								*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 ~ 6	PS[1:0]	外部引脚GTETRG输入采样时钟预分频比选择位（Prescaler select） 00：1/1 01：1/4 10：1/16 11：1/128
5 ~ 4	SN[1:0]	外部引脚GTETRG输入连续采样次数选择位（sample number） 00：1 01：2 10：3 11：4
3	ETNDMA	外部触发输入下降沿的DMA请求允许位 0：禁止DMA请求 1：允许DMA请求
2	ETPDMA	外部触发输入上升沿的DMA请求允许位 0：禁止DMA请求 1：允许DMA请求
1	ETINEN	外部触发输入下降沿的中断请求允许位 0：禁止中断请求 1：允许中断请求
0	ETIPEN	外部触发输入上升沿的中断请求允许位 0：禁止中断请求 1：允许中断请求

18.4.3 GPT 端口输出允许控制寄存器 (GPT_GTPOECR)

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								POEIE	保留	POEM[1:0]		保留	GPT2ABZE0	GPT1ABZE0	GPT0ABZE0
-								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	POEIE	GTPOE允许中断请求位 0：禁止中断请求 1：允许中断请求
6	保留	-
5 ~ 4	POEM[1:0]	GTPOE引脚滤波选择位 00：GTPOE输入的Low电平，保护发生。 01：在每HCLK/8 时钟对GTPOE输入的Low电平进行16 次采样并且全部为Low电平时，保护发生。

		10：在每HCLK/16时钟对GTPOE输入的Low电平进行16 次采样并且全部为Low电平时，保护发生。 11：在每HCLK/128 时钟对GTPOE输入的Low电平进行16 次采样并且全部为Low电平时，保护发生。
3	**保留**	**-**
2	**GPT2ABZE0**	**POE对GTIOC2A/B保护的使能位** 0：关闭保护功能 1：开启保护功能
1	**GPT1ABZE0**	**POE对GTIOC1A/B保护的使能位** 0：关闭保护功能 1：开启保护功能
0	**GPT0ABZE0**	**POE对GTIOC0A/B保护的使能位** 0：关闭保护功能 1：开启保护功能

18.4.4 GPT 寄存器修改和重载控制寄存器 (GPT_GTPRWEN)

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
GTPRWEN[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	**GTPRWEN[15:0]**	**寄存器修改和重载控制寄存器** 0x33CC：允许软件对保护模块寄存器的修改 注：保护寄存器包括GTSTR、GTETINT和GTPOECR，这些寄存器的修改有限制条件，只有当**GTPRWEN** = 0x33CC时才允许软件修改，否则修改无效

18.4.5 GPT 中断标志和清除寄存器 (GPT_GTINTF)

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													**POEIFC**	**ETINFC**	**ETIPFC**
-													*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													**POEIF**	**ETINF**	**ETIPF**
-													*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	**POEIFC**	**GTPOE中断标志清除位** 0：无效 1：清除
17	**ETINFC**	**外部触发输入下降沿的中断请求标志清除位** 0：无效 1：清除
16	**ETIPFC**	**外部触发输入上升沿的中断请求标志清除位** 0：无效 1：清除
15 ~ 3	保留	-
2	**POEIF**	**GTPOE中断标志位** 0：GTPOE无高阻抗请求 1：GTPOE有高阻抗请求
1	**ETINF**	**外部触发输入下降沿的中断请求标志** 0：无中断请求 1：有中断请求
0	**ETIPF**	**外部触发输入上升沿的中断请求标志** 0：无中断请求 1：有中断请求

GPT 模块寄存器列表 (基地址:0x0x4004 2400)

地址	寄存器名	说明
0x4004 2400	GTHCR	通用PWM 定时器的硬启动、停止和清除控制寄存器
0x4004 2404	GTDEB	通用PWM定时器的输入滤波寄存器
0x4004 2408	GTWP	通用PWM 定时器的写保护寄存器
0x4004 240C	GTBDR	通用PWM 定时器的缓冲运行禁止寄存器
0x4004 2410	GTIOR	通用PWM 定时器的I/O 控制寄存器
0x4004 2414	GTINTAD	通用PWM定时器的中断输出设定寄存器
0x4004 2418	GTDMA	通用PWM定时器的中断输出设定寄存器
0x4004 241C	GTCR	通用PWM定时器的控制寄存器
0x4004 2420	GTPSQ	GPT时钟预分频寄存器
0x4004 2424	GTBER	用PWM定时器的缓冲允许寄存器
0x4004 2428	GTUDC	通用PWM定时器的计数方向寄存器
0x4004 242C	GTITC	通用PWM定时器的中断、A/D转换开始请求减少设定寄存器
0x4004 2430	GTST	通用PWM定时器的状态寄存器
0x4004 2434	GTCNT	通用PWM 定时器的计数器
0x4004 2438	GTCCRA	通用PWM定时器的比较捕捉寄存器A
0x4004 243C	GTCCRB	通用PWM定时器的比较捕捉寄存器B
0x4004 2440	GTCCRC	通用PWM定时器的比较捕捉寄存器C
0x4004 2444	GTCCRD	通用PWM定时器的比较捕捉寄存器D
0x4004 2448	GTCCRE	通用PWM定时器的比较捕捉寄存器E
0x4004 244C	GTCCRF	通用PWM定时器的比较捕捉寄存器F
0x4004 2450	GTPR	通用PWM定时器的周期设定寄存器
0x4004 2454	GTPBR	通用PWM定时器的周期设定缓冲寄存器

0x4004 2458	GTPDBR	通用PWM定时器的周期设定双缓冲寄存器
0x4004 245C	GTADTRA	A/D转换开始请求时序寄存器A
0x4004 2460	GTADTRB	A/D转换开始请求时序寄存器B
0x4004 2464	GTADTBRA	A/D 转换开始请求时序缓冲寄存器A
0x4004 2468	GTADTBRB	A/D 转换开始请求时序缓冲寄存器B
0x4004 246C	GTADTDBRA	A/D 转换开始请求时序双缓冲寄存器A
0x4004 2470	GTADTDBRB	A/D 转换开始请求时序双缓冲寄存器B
0x4004 2474	GTDTCR	通用PWM 定时器的死区时间控制寄存器
0x4004 2478	GTDVU	通用PWM 定时器的死区时间值寄存器U
0x4004 247C	GTDVD	通用PWM 定时器的死区时间值寄存器D
0x4004 2480	GTDBU	通用PWM 定时器的死区时间缓冲寄存器U
0x4004 2484	GTDBD	通用PWM 定时器的死区时间缓冲寄存器D
0x4004 2488	GTOSCR	通用PWM定时器的电平保护控制寄存器

18.4.6 通用 PWM 定时器的硬启动、停止和清除控制寄存器 (GPTx_GTHCR)

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							CCSW	SYNC[1:0]		CCHW[1:0]		CPHW[1:0]		CSHW[1:0]	
-							*WO*	*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	CCSW	**GPTx.GTCNT 计数器清除位** 注：写“1”后自动恢复“0”，读取值为“0”。
7 ~ 6	SYNC[1:0]	**GPTx.GTCNT 的同步计数器清除源选择位** 00：通过GPT0 的清除源清除GPTx.GTCNT 01：通过GPT1 的清除源同步清除GPTx.GTCNT 10：通过GPT2 的清除源同步清除GPTx.GTCNT 11：保留 注1：必须在GPTx_GTCNT（0=0～2）计数器停止计数的状态下进行写操作。 注2：如果设定位SYNC[1:0]@GPTx_GTHCR，必须事先将位CCLR[1:0]@GPTx_GTCR置“11b”（通过正在进行同步清除/同步运行的其他计数器的清除源清除）。
5 ~ 4	CCHW[1:0]	**GPTx.GTCNT 硬件源GTETRG计数器清除** 00：未通过硬件源清除计数器 01：在硬件源的上升沿清除计数器 10：在硬件源的下降沿清除计数器 11：在硬件源的双边沿清除计数器 注：当设定为通过硬件源清除GPTx_GTCNT计数器时，不管GPTx_GTCNT计数器正在计数（CST@GPT_GTSTR = 1）还是停止计数（CTS=0），都通过硬件源进行计数器清除。
3 ~ 2	CPHW[1:0]	**GPTx.GTCNT 硬件源GTETRG计数停止位：**

		00：未通过硬件源停止计数 01：在硬件源的上升沿停止计数 10：在硬件源的下降沿停止计数 11：在硬件源的双边沿停止计数 注：如果通过硬件源停止计数，GTSTR寄存器的对应位就自动变为“0”。
1 ~ 0	**CSHW[1:0]**	**GPTx.GTCNT 硬件源GTETRG计数开始位：** 00：未通过硬件源开始计数 01：在硬件源的上升沿开始计数 10：在硬件源的下降沿开始计数 11：在硬件源的双边沿开始计数 注：如果通过硬件源启动计数，GTSTR寄存器的对应位就自动变为“1”。

注：如果同时发生由硬件源引起的GPTx.GTCNT计数器的计数和停止计数，就优先停止。

18.4.7 通用 PWM 定时器的输入滤波寄存器 (GPTx_GTDEB)

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						INBE	INAE	保留				DEB[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 ~ 10	保留	-
9	**INBE**	**通道B输入捕捉级联允许位** 0：不将GPTx_GTCCRB引脚追加到级联后32位另外一个定时器的输入捕捉条件 1：将GPTx_GTCCRB引脚追加到级联后32位另外一个定时器的输入捕捉条件 注1：当级联时，GPTx_GTCCRB对整个32位定时器都起作用 注2：当不级联时，GPTx_GTCCRB可以作为另外一个定时器的输入条件，所以本身定时器可以设定为比较输出，另外一个作为输入捕捉的定时器增加了一个输入引脚 注3：GPT0和GPT1是一对
8	**INAE**	**通道A输入捕捉级联允许位** 0：不将GPTx_GTCCRA引脚追加到级联后32位另外一个定时器的输入捕捉条件 1：将GPTx_GTCCRA引脚追加到级联后32位另外一个定时器的输入捕捉条件 注1：当级联时，GPTx_GTCCRA对整个32位定时器都起作用 注2：当不级联时，GPTx_GTCCRA可以作为另外一个定时器的输入条件，所以本身定时器可以设定为比较输出，另外一个作为输入捕捉的定时器增加了一个输入引脚
7 ~ 4	保留	-
3 ~ 0	**DEB[3:0]**	**GPTx输入信号滤波选择位**

		0000：无滤波 0001：滤波时间常数为30*HCLK(0.5us) 0010：滤波时间常数为60*HCLK(1us) 0011：滤波时间常数为120*HCLK(2us) 0100：滤波时间常数为180*HCLK(3us) 0101：滤波时间常数为240*HCLK(4us) 0110：滤波时间常数为300*HCLK(5us) 0111：滤波时间常数为360*HCLK(6us) 1000：滤波时间常数为480*HCLK(8us) 1001：滤波时间常数为600*HCLK(10us) 1010：滤波时间常数为720*HCLK(12us) 1011：滤波时间常数为960*HCLK(16us) 1100：滤波时间常数为1200*HCLK(20us) 1101：滤波时间常数为1440*HCLK(24us) 1110：滤波时间常数为1680*HCLK(28us) 1111：滤波时间常数为1920*HCLK(32us) 注1：上述滤波常数时间不是精确值，仅供参考。 注2：滤波说明：用内部时钟采样输入信号，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。

18.4.8 通用 PWM 定时器的写保护寄存器 (GPTx_GTWP)

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								GTWP[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	GTWP[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 注1：除了GPTx_GTST外的其他所有寄存器 注2：只有当WP[7:0] = 0x55时才允许软件修改，否则修改无效

18.4.9 通用 PWM 定时器的缓冲运行禁止寄存器 (GPTx_GTBDR)

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												BDDV	BDADTR	BDPR	BDCCR
-												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 ~ 4	保留	-
3	BDDV	GTDV 缓冲运行禁止位 0：允许缓冲运行 1：禁止缓冲运行
2	BDADTR	GTADTR 缓冲运行禁止位 0：允许缓冲运行 1：禁止缓冲运行
1	BDPR	GTPR 缓冲运行禁止位 0：允许缓冲运行 1：禁止缓冲运行
0	BDCCR	GTCCR 缓冲运行禁止位 0：允许缓冲运行 1：禁止缓冲运行

注：GTBDR寄存器是一次性设定允许或者禁止各通道的缓冲运行的寄存器。如果没有通过GTBER寄存器将缓冲运行置为有效，即使将GTBDR寄存器的各位置"0"（允许缓冲运行），也不进行缓冲运行。

18.4.10 通用 PWM 定时器的 I/O 控制寄存器 (GPTx_GTIOR)

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
OBHLD	OBDFLT	GTIOB[5:0]						OAHLD	OADFLT	GTIOA[5:0]					
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 ~ 16	保留	-
15	OBHLD	GTIOCxB 引脚计数开始/ 停止时的输出保持位 0：计数开始/ 停止时的GTIOCxB 引脚的输出电平取取决于寄存器的设定值。 在计数开始时，输出GTIOR寄存器的bit4指定的值 在计数停止时，输出OBDFLT位指定的值 在计数停止时更改OBDFLT位的值的情况下，立即反映到输出 1：保持计数开始/ 停止时的GTIOCxB 引脚的输出电平
14	OBDFLT	GTIOCxB 引脚计数停止时的输出值位 0：在停止计数时从GTIOCxB 引脚输出Low 电平 1：在停止计数时从GTIOCxB 引脚输出High 电平

13 ~ 8	**GTIOB[5:0]**	**GTIOCxB 引脚功能选择位** 请参照下文表1、2、3的设定
7	**OAHLD**	**GTIOCxA 引脚计数开始/ 停止时的输出保持位** 0：计数开始/ 停止时的GTIOCxA 引脚的输出电平取取决于寄存器的设定值。 在计数开始时，输出GTIOR寄存器的bit4指定的值 在计数停止时，输出OADFLT位指定的值 在计数停止时更改OADFLT位的值的情况下，立即反映到输出 1：保持计数开始/ 停止时的GTIOCxA 引脚的输出电平
6	**OADFLT**	**GTIOCxA 引脚计数停止时的输出值位** 0：在停止计数时从GTIOCxA 引脚输出Low 电平 1：在停止计数时从GTIOCxA 引脚输出High 电平
5 ~ 0	**GTIOA[5:0]**	**GTIOCxA 引脚功能选择位** 请参照下文表1、2、3的设定

表1：GPT0~GPT2作为比较匹配输出时GTIOA[5:0] 位（GTIOB[5:0] 位）的设定

GTIOA/B[5:0]						功能				
b5	b4	b3	b2	b1	b0	b5	b4	b3	b2	b1~b0
0	0	0	0	0	0	比较匹配	初始输出为LOW电平	在周期结束时保持输出		通过GPTn.GTCCRA/B 的比较匹配保持输出
0	0	0	0	0	1					通过GPTn.GTCCRA/B 的比较匹配输出Low 电平
0	0	0	0	1	0					通过GPTn.GTCCRA/B 的比较匹配输出High 电平
0	0	0	0	1	1					通过GPTn.GTCCRA/B 的比较匹配进行交替输出
0	0	0	1	0	0			在周期结束时输出LOW电平		通过GPTn.GTCCRA/B 的比较匹配保持输出
0	0	0	1	0	1					通过GPTn.GTCCRA/B 的比较匹配输出Low 电平
0	0	0	1	1	0					通过GPTn.GTCCRA/B 的比较匹配输出High 电平
0	0	0	1	1	1					通过GPTn.GTCCRA/B 的比较匹配进行交替输出
0	0	1	0	0	0			在周期结束时输出HIGH电平		通过GPTn.GTCCRA/B 的比较匹配保持输出
0	0	1	0	0	1					通过GPTn.GTCCRA/B 的比较匹配输出Low 电平
0	0	1	0	1	0					通过GPTn.GTCCRA/B 的比较匹配输出High 电平
0	0	1	0	1	1					通过GPTn.GTCCRA/B 的比较匹配进行交替输出
0	0	1	1	0	0			在周期结束时交替输出		通过GPTn.GTCCRA/B 的比较匹配保持输出
0	0	1	1	0	1					通过GPTn.GTCCRA/B 的比较匹配输出Low 电平
0	0	1	1	1	0					通过GPTn.GTCCRA/B 的比较匹配输出High 电平
0	0	1	1	1	1					通过GPTn.GTCCRA/B 的比较匹配进行交替输出
0	1	0	0	0	0		初始输出为HIGH电平	在周期结束时保持输出		通过GPTn.GTCCRA/B 的比较匹配保持输出
0	1	0	0	0	1					通过GPTn.GTCCRA/B 的比较匹配输出Low 电平
0	1	0	0	1	0					通过GPTn.GTCCRA/B 的比较匹配输出High 电平
0	1	0	0	1	1					通过GPTn.GTCCRA/B 的比较

									匹配进行交替输出
0	1	0	1	0	0			在周期结束时输出LOW电平	通过GPTn.GTCCRA/B 的比较匹配保持输出
0	1	0	1	0	1				通过GPTn.GTCCRA/B 的比较匹配输出Low 电平
0	1	0	1	1	0				通过GPTn.GTCCRA/B 的比较匹配输出High 电平
0	1	0	1	1	1				通过GPTn.GTCCRA/B 的比较匹配进行交替输出
0	1	1	0	0	0			在周期结束时输出HIGH电平	通过GPTn.GTCCRA/B 的比较匹配保持输出
0	1	1	0	0	1				通过GPTn.GTCCRA/B 的比较匹配输出Low 电平
0	1	1	0	1	0				通过GPTn.GTCCRA/B 的比较匹配输出High 电平
0	1	1	0	1	1				通过GPTn.GTCCRA/B 的比较匹配进行交替输出
0	1	1	1	0	0			在周期结束时交替输出	通过GPTn.GTCCRA/B 的比较匹配保持输出
0	1	1	1	0	1				通过GPTn.GTCCRA/B 的比较匹配输出Low 电平
0	1	1	1	1	0				通过GPTn.GTCCRA/B 的比较匹配输出High 电平
0	1	1	1	1	1				通过GPTn.GTCCRA/B 的比较匹配进行交替输出

表2： GPT0~GPT2作为输入捕捉时GTIOA[5:0] 位的设定

GTIOA[5:0]						功能				
b5	b4	b3	b2	b1	b0	b5	b4	b3	b2	b1~b0
1	x	0	0	0	0	输入捕捉	don’t care	GTIOCxA作为捕捉输入口		在上升沿输入捕捉
1	x	0	0	0	1					在下降沿输入捕捉
1	x	0	0	1	0					在双边沿输入捕捉
1	x	0	0	1	1					
1	x	0	1	0	0			RXD1作为是捕捉输入口		在上升沿输入捕捉
1	x	0	1	0	1					在下降沿输入捕捉
1	x	0	1	1	0					在双边沿输入捕捉
1	x	0	1	1	1					
1	x	1	0	0	0			RXD2作为是捕捉输入口		在上升沿输入捕捉
1	x	1	0	0	1					在下降沿输入捕捉
1	x	1	0	1	0					在双边沿输入捕捉
1	x	1	0	1	1					
1	x	1	1	0	0			RXD3作为是捕捉输入口		在上升沿输入捕捉
1	x	1	1	0	1					在下降沿输入捕捉
1	x	1	1	1	0					在双边沿输入捕捉
1	x	1	1	1	1					

表3： GPT0~GPT2作为输入捕捉时GTIOB[5:0] 位的设定

GTIOB[5:0]						功能				
b5	b4	b3	b2	b1	b0	b5	b4	b3	b2	b1~b0
1	x	x	x	0	0	输入捕捉	don’t care			在上升沿输入捕捉

1	x	x	x	0	1			在下降沿输入捕捉
1	x	x	x	1	0			在双边沿输入捕捉
1	x	x	x	1	1			

注1：x：don't care

注2：周期结束在锯齿波时表示上溢（递增计数时的GTCNT=GTPR）或者下溢（递减计数时的GTCNT=0），在三角波时表示波谷（GTCNT=0）。

注3：假如在时间点上有重合，则按照逻辑先后顺序叠加发生。例如，三角波模式，占空比比较匹配时交替输出，周期结束时输出低电平，占空比的值刚好为0，则占空比下降沿匹配、周期结束和占空比上升沿匹配都发生在同一时刻，则输出波形逻辑为：先发生下降沿匹配（交替输出）->再发生周期结束（输出低电平）->最后发生上升沿匹配（交替输出），所以最后输出波形为输出高电平。

18.4.11 通用 PWM 定时器的中断输出设定寄存器 (GPTx_GTINTAD)

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				ADTRBDEN	ADTRBUEN	ADTRADEN	ADTRAUEN	保留		OINT	EINT	GTINTPR[1:0]		GTINTB	GTINTA
-				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	ADTRBDEN	GTADTRB 比较匹配（递减计数） A/D 转换开始请求允许位 0：禁止A/D 转换开始请求 1：允许A/D 转换开始请求
10	ADTRBUEN	GTADTRB 比较匹配（递增计数） A/D 转换开始请求允许位 0：禁止A/D 转换开始请求 1：允许A/D 转换开始请求
9	ADTRADEN	GTADTRA 比较匹配（递减计数） A/D 转换开始请求允许位 0：禁止A/D 转换开始请求 1：允许A/D 转换开始请求
8	ADTRAUEN	GTADTRA 比较匹配（递增计数） A/D 转换开始请求允许位 0：禁止A/D 转换开始请求 1：允许A/D 转换开始请求
7 ~ 6	保留	-
5	OINT	输出短路的中断允许位 0：禁止中断请求 1：允许中断请求
4	EINT	死区时间错误中断允许位 0：禁止中断请求 1：允许中断请求
3 ~ 2	GTINTPR[1:0]	GTPR 比较匹配中断允许位 00：禁止中断请求 01：锯齿波时在上溢、三角波时在波峰允许中断请求

		10：锯齿波时在下溢、三角波时在波谷允许中断请求 11：锯齿波时在上溢和下溢、三角波时在波峰和波谷都允许中断请求
1	**GTINTB**	**GTCCRB 比较匹配/ 输入捕捉中断允许位** 0：禁止中断请求 1：允许中断请求
0	**GTINTA**	**GTCCRA 比较匹配/ 输入捕捉中断允许位** 0：禁止中断请求 1：允许中断请求

18.4.12 通用 PWM 定时器的中断输出设定寄存器 (GPTx_GTDMA)

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
保留												PR[1:0]		TB	TA
-												*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 ~ 4	保留	-
3 ~ 2	**PR[1:0]**	**GTPR 比较匹配DMA允许位** 00：禁止DMA请求 01：锯齿波时在上溢、三角波时在波峰允许DMA请求 10：锯齿波时在下溢、三角波时在波谷允许DMA请求 11：锯齿波时在上溢和下溢、三角波时在波峰和波谷都允许DMA请求
1	**TB**	**GTCCRB 比较匹配/ 输入捕捉DMA允许位** 0：禁止DMA请求 1：允许DMA请求
0	**TA**	**GTCCRA 比较匹配/ 输入捕捉DMA允许位** 0：禁止DMA请求 1：允许DMA请求

18.4.13 通用 PWM 定时器的控制寄存器 (GPTx_GTCR)

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		CCLR[1:0]		保留		HIZ	TPSC	保留						MD[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 ~ 14	保留	-
13 ~ 12	CCLR[1:0]	计数器清除源选择位 00：不能设定下述清除源 01：通过GTCCRA 寄存器的输入捕捉清除 10：通过GTCCRB 寄存器的输入捕捉清除 11：通过正在进行同步清除/ 同步运行的其他计数器清除源清除 注1：如果在锯齿波的情况下选择同步清除，此时的同步清除与通过自身的上溢/ 下溢进行的清除相同，进行引脚输出/ 缓冲运行，但是，上溢标志/ 下溢标志不被置位；如果在三角波的情况下选择同步清除，此时的同步清除只进行计数器的清除，计数器的值变为“0”，但是，不作为“ 波谷” 处理。 注2：通过计数器清除源选择“01b”、“10b” 或者“11b” 时，不管GTCNT 计数器正在计数（CSTn@GPT_GTSTR=1）(n=0~3)，还是停止计数（CSTn=0）（n=0 ～3），都通过计数器清除源清除计数器。
11 ~ 10	保留	-
9	HIZ	比较器CMP1保护GPTx的使能位 0：禁止CMP1保护GPTx 1：使能CMP1保护GPTx 注：此位置起后，当CMP1的C1IF@AMOC_CMPINTF为1时，GPTx输出高阻态
8	TPSC	定时器的时钟源选择位： 0：HCLK 1：为GPT1的上溢/下溢进行计数 注：GPT0的这位有效，GPT1的这位保留
7 ~ 3	保留	-
2 ~ 0	MD[2:0]	模式选择位 000：锯齿波PWM 模式（能单/ 双缓冲） 001：锯齿波单触发脉冲模式（固定为缓冲运行） 010：不能设定 011：不能设定 100：三角波PWM 模式1（在波谷以16 位传送）（能单/ 双缓冲） 101：三角波PWM 模式2（在波峰/ 波谷以16 位传送）（能单/ 双缓冲） 110：三角波PWM 模式3（在波谷以32 位传送）（固定为缓冲运行） 111：三角波PWM 模式4（在波峰以16位传送）（能单/双缓冲）

注：必须在GTCNT 计数器停止计数的状态下设定GTCR 寄存器。

18.4.14 GPT 时钟预分频寄存器 (GPTx_GTPSQ)

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
GTPSQ[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	GTPSQ[15:0]	GPT时钟为(GPTCLK/(PSQ[15:0]+1)) 注：在定时器运行时不允许被修改

18.4.15 用 PWM 定时器的缓冲允许寄存器 (GPTx_GTBER)

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	ADTDB	ADTTB[1:0]		保留	ADTDA	ADTTA[1:0]		保留	CCRSWT	PR[1:0]		CCRB[1:0]		CCRA[1:0]	
-	*RW*	*RW*		-	*RW*	*RW*		-	*WO*	*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	ADTDB	GTADTRB 双缓冲运行位 0：进行单缓冲运行（GTADTBRB=>GTADTRB） 1：进行双缓冲运行（GTADTDBRB=>GTADTBRB=>GTADTRB）
13 ~ 12	ADTTB[1:0]	GTADTRB 缓冲传送时序选择位 三角波的情况： 00：不传送 01：在波峰进行传送 10：在波谷进行传送 11：在波峰/ 波谷进行传送 锯齿波的情况： 00：不传送 其他：在上溢（递增计数）、下溢（递减计数）时进行传送
11	保留	-
10	ADTDA	GTADTRA 双缓冲运行位 0：进行单缓冲运行（GTADTBRA=>GTADTRA） 1：进行双缓冲运行（GTADTDBRA=>GTADTBRA=>GTADTRA）
9 ~ 8	ADTTA[1:0]	GTADTRA 缓冲传送时序选择位 三角波的情况： 00：不传送 01：在波峰进行传送 10：在波谷进行传送 11：在波峰/ 波谷进行传送 锯齿波的情况： 00：不传送 其他：在上溢（递增计数）、下溢（递减计数）时进行传送
7	保留	-
6	CCRSWT	GTCCRA、GTCCRB 强制缓冲运行位

		如果写“1”，就强制进行GTCCRA、GTCCRB 的缓冲传送。写“1”后自动回复“0”。 读取值为“0”。 注：只在停止计数并且设定为比较匹配运行时有效。
5 ~ 4	**PR[1:0]**	**GTPR 缓冲运行位** 00：不缓冲运行 01：进行单缓冲运行（GTPBR=>GTPR） 1x：进行双缓冲运行（GTPDBR=>GTPBR=>GTPR）
3 ~ 2	**CCRB[1:0]**	**GTCCRB 缓冲运行位** 00：不缓冲运行 01：进行单缓冲运行（GTCCRB=>GTCCRE） 1x：进行双缓冲运行（GTCCRB=>GTCCRE=>GTCCRF）
1 ~ 0	**CCRA[1:0]**	**GTCCRA 缓冲运行位** 00：不进行缓冲运行 01：进行单缓冲运行（GTCCRA=>GTCCRC） 1x：进行双缓冲运行（GTCCRA=>GTCCRC=>GTCCRD）

注1：必须在GTCNT计数器停止计数的状态下设定GTBER寄存器。

注2. 在锯齿波单触发脉冲模式或者三角波PWM模式3（在波谷以32位传送）的情况下，固定为缓冲运行。

18.4.16 通用 PWM 定时器的计数方向寄存器 (GPTx_GTUDC)

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															
-															
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														UDF	UD
-														*RW*	*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	**UDF**	**计数方向强制设定位** 0：不强制设定 1：强制设定 此位将GTCNT计数器开始运行时的计数方向强制设定为UD位的值。计数时只能给此位写“0”。
0	**UD**	**计数方向设定位** 0：GTCNT 计数器递增计数 1：GTCNT 计数器递减计数

注：

- *锯齿波的情况：*

如果在递增计数时将UD位置“1”，就在发生上溢（GTCNT=GTPR）时转换计数方向。

如果在递减计数时将UD位置“0”，就在发生下溢（GTCNT=0）时转换计数方向。

如果在停止计数时UDF位为“0”的状态下，将UD位从“0”更改为“1”，初始计数运行就为递增计数，并且在发生上溢（GTCNT=GTPR）时转换计数方向。

如果在停止计数时UDF位为“0”的状态下，将UD位从“1”更改为“0”，初始计数运行就为递减计数，并且在发生下溢（GTCNT=0）时转换计数方向。

如果在停止计数时将UDF位置“1”，此时的UD位的值就反映到开始计数后的计数方向。

■ *三角波的情况：*

即使在计数时更改UD位的值，也不反映到计数方向。

即使在停止计数时UDF位为“0”的状态下更改UD位的值，也不反映到开始计数后的计数方向。

如果在停止计数时将UDF位置“1”，此时的UD位的值反映到开始计数后的计数方向。

18.4.17 通用 PWM 定时器的中断、A/D 转换开始请求减少设定寄存器 (GPTx_GTITC)

偏移地址: 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
保留	ADTBL	保留	ADTAL	保留	IVTT[2:0]			保留		IVTC[1:0]		保留		ITLB	ITLA
-	*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	ADTBL	**GTADTRB 的A/D 转换开始请求联动位** 0：不联动GTCIV 中断减少功能 1：联动GTCIV 中断减少功能
13	保留	-
12	ADTAL	**GTADTRA 的A/D 转换开始请求联动位** 0：不联动GTCIV 中断减少功能 1：联动GTCIV 中断减少功能
11	保留	-
10 ~ 8	IVTT[2:0]	**GTCIV 中断减少次数选择位** 000：不减少 001：减少次数为1 次 010：减少次数为2 次 011：减少次数为3 次 100：减少次数为4 次 101：减少次数为5 次 110：减少次数为6 次 111：减少次数为7 次 注：必须在将IVTC[1:0]位置“00b”后才能更改IVTT[2:0]位。
7 ~ 6	保留	-
5 ~ 4	IVTC[1:0]	**GTCIV 中断减少功能选择位** 00：不联动 01：对锯齿波时的上溢、三角波时的波峰进行计数后减少 10：对锯齿波时的下溢、三角波时的波谷进行计数后减少 11：对锯齿波时的上溢/ 下溢、三角波波峰/ 波谷都进行计数后减少
3 ~ 2	保留	-
1	ITLB	**GTCCRB 比较匹配/ 输入捕捉中断联动位** 0：不联动GTCIV 中断减少功能

		1：联动GTCIV 中断减少功能
0	**ITLA**	**GTCCRA 比较匹配/ 输入捕捉中断联动位** 0：不联动GTCIV 中断减少功能 1：联动GTCIV 中断减少功能

注：死区时间错误中断不能联动GTCIV中断减少功能。当设定为中断减少功能时，状态标志的变化也减少。

18.4.18 通用 PWM 定时器的状态寄存器 (GPTx_GTST)

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										OSFC	DTEFC	TCFPUC	TCFPOC	TCFBC	TCFAC
-										*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
TUCF	保留				ITCNT[2:0]			保留		OSF	DTEF	TCFPU	TCFPO	TCFB	TCFA
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 ~ 22	保留	-
21	**OSFC**	**输出短路标志位清除位** 0：无效 1：清除
20	**DTEFC**	**死区时间错误标志清除位** 0：无效 1：清除
19	**TCFPUC**	**下溢标志清除位** 0：无效 1：清除
18	**TCFPOC**	**上溢标志清除位** 0：无效 1：清除
17	**TCFBC**	**输入捕捉/ 比较匹配B标志清除位** 0：无效 1：清除
16	**TCFAC**	**输入捕捉/ 比较匹配A标志清除位** 0：无效 1：清除
15	**TUCF**	**计数方向标志** 0：GPTx.GTCNT 计数器递增计数 1：GPTx.GTCNT 计数器递减计数
14 ~ 11	保留	-
10 ~ 8	**ITCNT[2:0]**	**GTCIV 中断减少次数计数器** 在使用GTCIV中断减少功能（将位IVTC[1:0]@GPTx_GTITC设定为不为“00b”）时，每产生1次GTCIV中断源就递增1。 为“0”的条件: ■ 未使用GTCIV中断减少功能时（IVTC[1:0]位为“00b”、IVTT[2:0]位为

		“000b”时） ■ GTCIV中断减少次数相同时（通过IVTT[2:0]位设定的减少次数与ITCNT[2:0]位的值相同时
7 ~ 6	**保留**	-
5	**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）
4	**DTEF**	**死区时间错误标志** 0：未发生死区时间错误 1：发生死区时间错误 这是表示自动附加死区时间后的定时器输出交替点超出定时器周期的标志。 如果自动附加死区时间后的定时器输出交替点回复到定时器周期内，此位就变为“0”。DTEF标志是只读标志（即使写“0”也不变为“0”）。 如果允许由DTEF标志引起的中断（EINT@GPTx_GTINTAD=1），每当DTEF标志从“0”变为“1”时，就发生GTCIC中断。 为“1”的条件: ■ 自动附加死区时间后的定时器输出交替点超出定时器周期时 为“0”的条件: ■ 自动附加死区时间后的定时器输出交替点在定时器周期内时
3	**TCFPU**	**下溢标志** 0：未发生下溢（递减计数时GTCNT=0）或者波谷 1：发生下溢或者波谷
2	**TCFPO**	**上溢标志** 0：未发生上溢（递增计数时GTCNT=GTPR）或者波峰 1：发生上溢或者波峰
1	**TCFB**	**输入捕捉/ 比较匹配B标志** 0：未发生GTCCRB 寄存器的输入捕捉/ 比较匹配 1：发生GTCCRB 寄存器的输入捕捉/ 比较匹配 ■ 在GTCCRB寄存器用作比较匹配寄存器的情况下，GTCNT=GTCCRB时 ■ 在GTCCRB寄存器用作输入捕捉寄存器的情况下，通过输入捕捉信号将GTCNT计数器的值传送到
0	**TCFA**	**输入捕捉/ 比较匹配A标志** 0：未发生GTCCRA 寄存器的输入捕捉/ 比较匹配 1：发生GTCCRA 寄存器的输入捕捉/ 比较匹配 ■ 在GTCCRA寄存器用作比较匹配寄存器的情况下，GTCNT=GTCCRA时 ■ 在GTCCRA寄存器用作输入捕捉寄存器的情况下，通过输入捕捉信号将GTCNT计数器的值传送到

注：GTCCRC~F不进行比较匹配的条件如下

- *MD[2:0]@GTCR =001b（锯齿波单触发脉冲模式）*
- *MD[2:0]@GTCR =110b（三角波PWM模式3）*
- *CCRA[1:0]@ GTBER =01b、10b、11b（GTCCRC~F寄存器缓冲运行）*

18.4.19 通用 PWM 定时器的计数器 (GPTx_GTCNT)

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
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GPT1有效**
15 ~ 0	**TCNTL[15:0]**	**定时器计数值低16位：全部GPT有效**

注1：GTCNT计数器是16位可读写计数器。各通道有1个（共计4个）GTCNT计数器。不能在计数时进行写操作，只能在停止计数时进行写操作。禁止以8位为单位存取GTCNT计数器，必须以16位为单位进行存取。

注2：当写入GTCNT初值大于GTPR时，GTCNT会根据初始计数方向递增或者递减计数，如果为递增，则从初值直到计到0xFFFF后回到0x0000。

注3：级联时，GPT0的定时寄存器被映射到GPT1的定时寄存器的高位上，当读取时，可以直接读取GPT1.GTCNT的32bit值。

18.4.20 通用 PWM 定时器的比较捕捉寄存器 A (GPTx_GTCCRA)

偏移地址: 0x0038

复位值: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
GTCCRAH[15:0]															
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
GTCCRAL[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	**GTCCRAH[15:0]**	**GTCCRA寄存器高16位：** 当级联时，低位GPT的高16位寄存器为只读寄存器，与高位GPT的低16位读到的值一样。例如GPT0和GPT1级联时，GPT1_GTCCRAH只能读，与GPT0_GTCCRAL的值一样。 当非级联时，此16位永远为0 注：只有级联时的GPT1有效
15 ~ 0	**GTCCRAL[15:0]**	**GTCCRA寄存器低16位：** 注：全部GPT有效

18.4.21 通用 PWM 定时器的比较捕捉寄存器 B (GPTx_GTCCRB)

偏移地址: 0x003C

复位值: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
GTCCRBH[15:0]															
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
GTCCRBL[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	**GTCCRBH[15:0]**	**GTCCRB寄存器高16位：** 当级联时，低位GPT的高16位寄存器为只读寄存器，与高位GPT的低16位读到的值一样。例如GPT0和GPT1级联时，GPT1_GTCCRBH只能读，与GPT0_GTCCRBL的值一样。 当非级联时，此16位永远为0 注：只有级联时的GPT1有效
15 ~ 0	**GTCCRBL[15:0]**	**GTCCRB寄存器低16位：** 注：全部GPT有效

18.4.22 通用 PWM 定时器的比较捕捉寄存器 C (GPTx_GTCCRC)

偏移地址: 0x0040

复位值: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
GTCCRCH[15:0]															
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
GTCCRCL[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	**GTCCRCH[15:0]**	**GTCCRC寄存器高16位：** 当级联时，低位GPT的高16位寄存器为只读寄存器，与高位GPT的低16位读到的值一样。例如GPT0和GPT1级联时，GPT1_GTCCRCH只能读，与GPT0_GTCCRCL的值一样。 当非级联时，此16位永远为0 注：只有级联时的GPT1有效
15 ~ 0	**GTCCRCL[15:0]**	**GTCCRC寄存器低16位：** 注：全部GPT有效

18.4.23 通用 PWM 定时器的比较捕捉寄存器 D (GPTx_GTCCRD)

偏移地址: 0x0044

复位值: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
GTCCRDH[15:0]															
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
GTCCRDL[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	**GTCCRDH[15:0]**	**GTCCRD寄存器高16位：** 当级联时，低位GPT的高16位寄存器为只读寄存器，与高位GPT的低16位读到的值一样。例如GPT0和GPT1级联时，GPT1_GTCCRDH只能读，与GPT0_GTCCRDL的值一样。 当非级联时，此16位永远为0 注：只有级联时的GPT1有效
15 ~ 0	**GTCCRDL[15:0]**	**GTCCRD寄存器低16位：** 注：全部GPT有效

18.4.24 通用 PWM 定时器的比较捕捉寄存器 E (GPTx_GTCCRE)

偏移地址: 0x0048

复位值: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
GTCCREH[15:0]															
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
GTCCREL[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	**GTCCREH[15:0]**	**GTCCRE寄存器高16位：** 当级联时，低位GPT的高16位寄存器为只读寄存器，与高位GPT的低16位读到的值一样。例如GPT0和GPT1级联时，GPT1_GTCCREH只能读，与GPT0_GTCCREL的值一样。 当非级联时，此16位永远为0 注：只有级联时的GPT1有效
15 ~ 0	**GTCCREL[15:0]**	**GTCCRE寄存器低16位：** 注：全部GPT有效

18.4.25 通用 PWM 定时器的比较捕捉寄存器 F (GPTx_GTCCRF)

偏移地址: 0x004C

复位值: 0x0000 FFFF

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
GTCCRFH[15:0]															
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
GTCCRFL[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	**GTCCRFH[15:0]**	**GTCCRF寄存器高16位：** 当级联时，低位GPT的高16位寄存器为只读寄存器，与高位GPT的低16位读到的值一样。例如GPT0和GPT1级联时，GPT1_GTCCRFH只能读，与GPT0_GTCCRFL的值一样。 当非级联时，此16位永远为0 注：只有级联时的GPT1有效
15 ~ 0	**GTCCRFL[15:0]**	**GTCCRF寄存器低16位：** 注：全部GPT有效

18.4.26 通用 PWM 定时器的周期设定寄存器 (GPTx_GTPR)

偏移地址: 0x0050

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
GTPR[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	保留	-
15 ~ 0	**GTPR[15:0]**	**通用PWM定时器的周期设定寄存器：** GTPR寄存器是16位可读写寄存器。用于设定GTCNT计数器的最大计数值。各通道有1个（共计4个）GTPR寄存器。 在锯齿波时，计数周期为GTPR值×1，在三角波时，计数周期为GTPR值×2。

18.4.27 通用 PWM 定时器的周期设定缓冲寄存器 (GPTx_GTPBR)

偏移地址: 0x0054

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
GTPBR[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	保留	-
15 ~ 0	**GTPBR[15:0]**	**通用PWM定时器的周期设定缓冲寄存器：** GTPBR寄存器是16位可读写寄存器。用作GTPR寄存器的缓冲寄存器。

18.4.28 通用 PWM 定时器的周期设定双缓冲寄存器 (GPTx_GTPDBR)

偏移地址: 0x0058

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
GTPDBR[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	保留	-
15 ~ 0	**GTPDBR[15:0]**	**通用PWM定时器的周期设定双缓冲寄存器：** GTPDBR寄存器是16位可读写寄存器。用作GTPBR寄存器的缓冲寄存器（GTPR寄存器的双缓冲寄存器）。

18.4.29 A/D 转换开始请求时序寄存器 A (GPTx_GTADTRA)

偏移地址: 0x005C

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
GTADTRA[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	保留	-
15 ~ 0	GTADTRA[15:0]	**A/D 转换开始请求时序寄存器：** 用于设定A/D转换开始请求的时序。当GTADTRA寄存器的值与GTCNT计数器的值相同时，产生A/D转换开始请求。禁止以8位为单位存取GTADTR寄存器，必须以16位为单位进行存取。

18.4.30 A/D 转换开始请求时序寄存器 B (GPTx_GTADTRB)

偏移地址: 0x0060

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
GTADTRB[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	保留	-
15 ~ 0	GTADTRB[15:0]	**A/D 转换开始请求时序寄存器：** 用于设定A/D转换开始请求的时序。当GTADTRB寄存器的值与GTCNT计数器的值相同时，产生A/D转换开始请求。禁止以8位为单位存取GTADTR寄存器，必须以16位为单位进行存取。

18.4.31 A/D 转换开始请求时序缓冲寄存器 A (GPTx_GTADTBRA)

偏移地址: 0x0064

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
GTADTBRA[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	保留	-
15 ~ 0	GTADTBRA[15:0]	**A/D 转换开始请求时序缓冲寄存器：** GTADTBRA寄存器是16位可读写寄存器。用作GTADTRA寄存器的缓冲寄存器。禁止以8位为单位存取GTADTBR寄存器，必须以16位为单位进行存取。

18.4.32 A/D 转换开始请求时序缓冲寄存器 B (GPTx_GTADTBRB)

偏移地址: 0x0068

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
GTADTBRB[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	保留	-
15 ~ 0	GTADTBRB[15:0]	**A/D 转换开始请求时序缓冲寄存器：** GTADTBRB寄存器是16位可读写寄存器。用作GTADTRB寄存器的缓冲寄存器。禁止以8位为单位存取GTADTBR寄存器，必须以16位为单位进行存取。

18.4.33 A/D 转换开始请求时序双缓冲寄存器 A (GPTx_GTADTDBRA)

偏移地址: 0x006C

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
GTADTDBRA[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	保留	-
15 ~ 0	GTADTDBRA[15:0]	通用PWM定时器的周期设定双缓冲寄存器： GTADTDBRA寄存器是16位可读写寄存器。用作GTADTBRA寄存器的缓冲寄存器（GTADTR的双缓冲寄存器）。不能以8位为单位存取GTADTDBR寄存器，必须以16位为单位进行存期。

18.4.34 A/D 转换开始请求时序双缓冲寄存器 B (GPTx_GTADTDBRB)

偏移地址: 0x0070

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
GTADTDBRB[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	保留	-
15 ~ 0	GTADTDBRB[15:0]	通用PWM定时器的周期设定双缓冲寄存器： GTADTDBRB寄存器是16位可读写寄存器。用作GTADTBRB寄存器的缓冲寄存器（GTADTR的双缓冲寄存器）。不能以8位为单位存取GTADTDBR寄存器，必须以16位为单位进行存期。

18.4.35 通用 PWM 定时器的死区时间控制寄存器 (GPTx_GTDTCR)

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							TDFER	保留		TDBDE	TDBUE	保留			TDE
-							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	TDFER	GTDVD 设定位 0：个别设定GTDVU 寄存器、GTDVD 寄存器 1：将GTDVU 寄存器的写入值自动设定到GTDVD寄存器

7 ~ 6	保留	-
5	TDBDE	**GTDVD 缓冲运行运行位** 0：禁止GTDVD 寄存器的缓冲运行 1：允许GTDVD 寄存器的缓冲运行 此位允许GTDVD寄存器和GTDBD寄存器组合的缓冲运行。缓冲传送时序在三角波时为波谷，在锯齿波时为上溢或者下溢。同时将TDFER位置“1”时，优先TDFER位的设定。
4	TDBUE	**GTDVU 缓冲运行允许位** 0：禁止GTDVU 寄存器的缓冲运行 1：允许GTDVU 寄存器的缓冲运行 缓冲传送时序在三角波时为波谷，在锯齿波时为上溢或者下溢。
3 ~ 1	保留	-
0	TDE	**反相波形设定位** 0：不使用GTDVU 寄存器、GTDVD 寄存器，而个别设定GTCCRB 寄存器 1：使用GTDVU 寄存器、GTDVD 寄存器，将带死区时间的反相波形的比较匹配值自动设定到GTCCRB寄存器。 此位设定是否使用GTDVU寄存器和GTDVD寄存器。当使用GTDVU寄存器和GTDVD寄存器时，将由正相波形的比较匹配值（GTCRRA寄存器）和死区时间值（GTDVU寄存器和GTDVD寄存器）算出的带死区时间的反相波形的比较匹配值自动设定到GTCCRB寄存器。 在锯齿波PWM模式中忽视TDE位的设定，也不自动设定此位。 自动设定的GTCCRB寄存器的上限值/下限值如下所示。当算出的GTCCRB寄存器值在上限值/下限值的范围外时，就给GTCCRB寄存器设定上限值/下限值，并且GTST.DTEF标志变为“1”。 三角波的情况上限值：GTPR－1　下限值：递增计数时为“1”，递减计数时为“0” 锯齿波单触发脉冲模式的情况上限值：GTPR　下限值：0

18.4.36 通用 PWM 定时器的死区时间值寄存器 U (GPTx_GTDVU)

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
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
GTDVU[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	GTDVU[15:0]	**通用PWM定时器的死区时间值寄存器：** GTDVU寄存器是16位可读写寄存器。用于设定生成带死区时间的PWM波形的死区时间。 禁止设定超出周期的死区时间值。能通过读GTCCRB寄存器读设定值。在使用GTDV寄存器时禁止写GTCCRB寄存器。如果将死区时间值设定为“0”，就输出无死区时间的波形。 禁止以8位为单位存取GTDV寄存器，必须以16位为单位进行存取。

18.4.37 通用 PWM 定时器的死区时间值寄存器 D (GPTx_GTDVD)

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
GTDVD[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	**GTDVD[15:0]**	**通用PWM定时器的死区时间值寄存器：** GTDVD寄存器是16位可读写寄存器。用于设定生成带死区时间的PWM波形的死区时间。 禁止设定超出周期的死区时间值。能通过读GTCCRB寄存器读设定值。在使用GTDV寄存器时禁止写GTCCRB寄存器。如果将死区时间值设定为“0”，就输出无死区时间的波形。 禁止以8位为单位存取GTDV寄存器，必须以16位为单位进行存取。

18.4.38 通用 PWM 定时器的死区时间缓冲寄存器 U (GPTx_GTDBU)

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
GTDBU[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	**GTDBU[15:0]**	**通用PWM定时器的死区时间缓冲寄存器：** GTDBU寄存器是16位可读写寄存器。用作GTDVU寄存器的缓冲寄存器。

18.4.39 通用 PWM 定时器的死区时间缓冲寄存器 D (GPTx_GTDBD)

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
GTDBD[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	GTDBD[15:0]	通用PWM定时器的死区时间缓冲寄存器： GTDBD寄存器是16位可读写寄存器。用作GTDVD寄存器的缓冲寄存器。

18.4.40 通用 PWM 定时器的电平保护控制寄存器 (GPTx_GTOSCR)

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
保留								OLSEN	保留					OLSGB	OLSGA
-								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 ~ 2	保留	-
1	OLSGB	输出短路保护中GTIOCxB的有效电平设定位 0：Low 电平有效 1：High 电平有效
0	OLSGA	输出短路保护中GTIOCxA的有效电平设定位 0：Low 电平有效 1：High 电平有效

19. 基本定时器（Basic Timer）

19.1 简介

SH32F284内置4个基本的16 位定时器（TIM5～TIM8），可级联成两个32位的定时器。4个TIMER都挂在APB1总线上，可配置以PCLK1时钟运行，有两个定时器可以在低频RC（128K的LSICLK）时钟下运行。

19.2 主要特性

- 4个16位定时器
- 每个定时器能够进行自动重载递增计数
- 每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
- 4个16位定时器能够级联成2个32位定时器

19.3 功能描述

SH32F284有4个基本定时器，每个定时器有一个16位计数器TIMx_TCNT（x=5~8），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=5~8）。TIMx_CR（x=5~8）寄存器的IE位置1使能定时器中断。

下图为定时器框图，其中TIM7和TIM8可以唤醒低功耗。

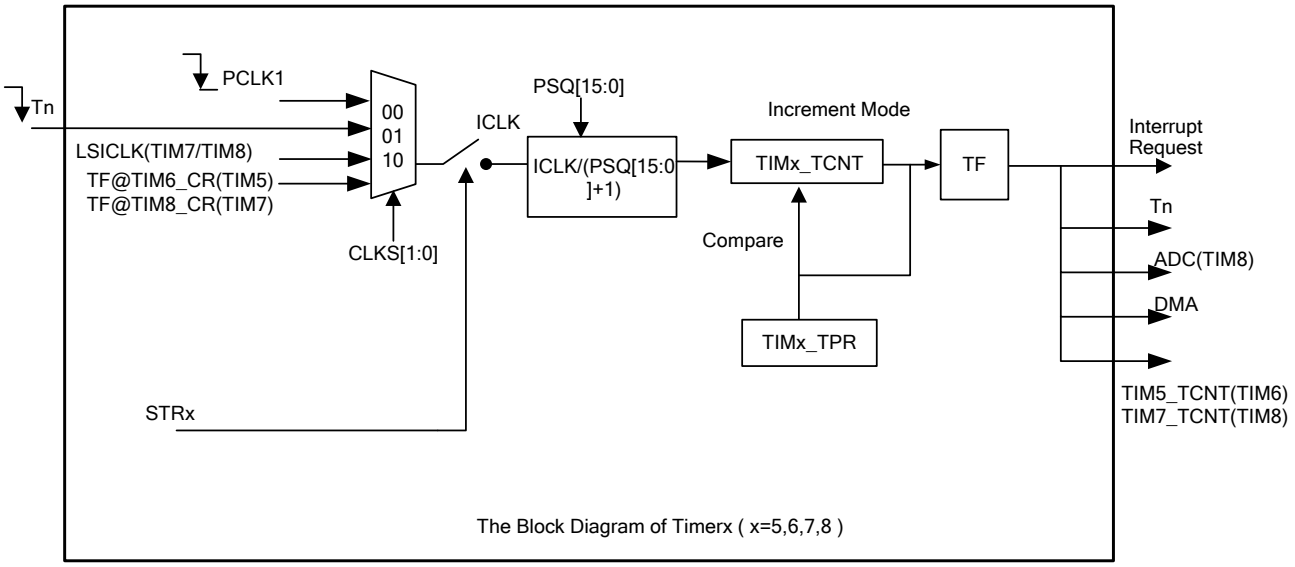


图19-1　　定时器框图

注1：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。

如果CLKS[1:0]@TIMx_CR为00，定时器x工作在普通模式下。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。如果CLKS[1:0]@TIM7_CR为10，定时器x可以工作在低功耗模式下，且支持开启级联功能。

○：能　　—：不能

PCLK1为APB1时钟，ICLK为时钟源

项目	TIM5	TIM6	TIM7	TIM8
时钟源（ICLK）	PCLK1 T5 TIM6	PCLK1 T6	PCLK1 LSICLK T7 TIM8	PCLK1 LSICLK T8
时钟源分频	ICLK/(PSQ[15:0]+1)	ICLK/(PSQ[15:0]+1)	ICLK/(PSQ[15:0]+1)	ICLK/(PSQ[15:0]+1)
周期设定寄存器	TIM5.TPR	TIM6.TPR	TIM7.TPR	TIM8.TPR

输入引脚	T5	T6	T7	T8
输出引脚	T5	T6	T7	T8
级联运行	○	○	○	○
DMA启动	全部的中断源			
A/D转换开始触发	—	—	—	•TCNT溢出 （**TIM8.TF**）
中断源	• TCNT溢出 （**TIM5.TF**）	• TCNT溢出 （**TIM6.TF**）	• TCNT溢出 （**TIM7.TF**）	• TCNT溢出 （**TIM8.TF**）

19.3.1 计数器/定时器运行

开始计时后，定时器开始递增计数，当计数值计到周期寄存器TIMx.TPR（x=5~8）的值后溢出，系统置起定时器溢出标志TFx，计数器TIMx_TCNT（x=5~8）被清零。如果定时器x中断被允许，将会产生一个中断。

举例：设定定时周期寄存器TIMx_TPR = 36，TIMx_TCNT = 0（默认为0），开启定时，此时计数器TIMx_TCNT从0计数到36后产生溢出信号，即定时器周期为37个时钟，时序图如下图所示：

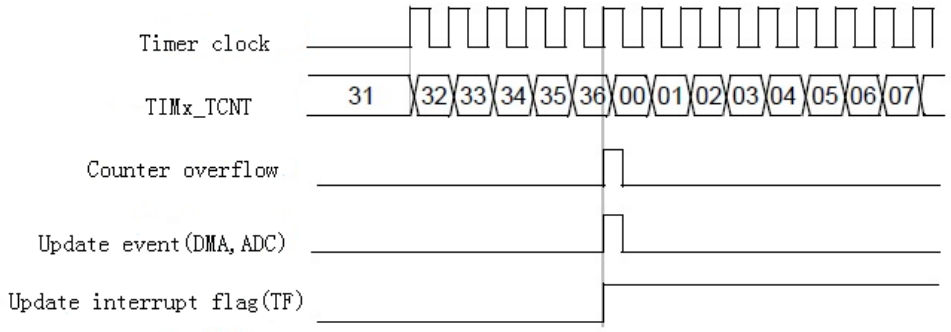


图19-2　　定时器工作时序图

CLKS[1:0]@TIMx_CR位选择定时器的时钟源：可以选择PCLK1、定时器x（x=5~8）输入引脚（Tx）的下降沿、低频内部RC（LSICLK）和另外一个TIMER的溢出标志位。其中只有TIM5和TIM7可以分别选择TIM6和TIM8的溢出标志位作为时钟源（级联），只有TIM7和TIM8可以选择低频内部RC（LSICLK）。

STR@TIMx_CR置1时不复位定时器，这意味着如果STR@TIMx_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，应该设定定时器计数器的初始值。

在定时器模式下，可配置TC@TIMx_CR位使定时器x溢出时Tx脚输出电平自动翻转，此配置下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递增计数到与TIMx_TPR相等后归零时刻。

注4：预分频寄存器在定时器停止与运行时都能修改。

*注5：Tx输入的最快频率1/2*PCLK1。*

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。

19.3.2 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CASCEN@TIM5/7_CR设为1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5和定时器6	TIM5	TIM6
定时器7和定时器8	TIM7	TIM8

计数器级联运行的例子，设定TIM6的时钟源为外部引脚T6口下降沿，计数如下图：

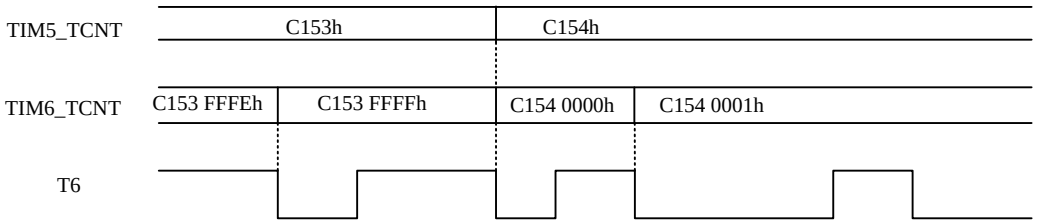


图19-3　　级联运行示例图

注1：级联之后，只需要配置TIM6/TIM8的值，不需要配置TIM5/7。TIM5_TCNT的值被自动映像到TIM6_TCNT的高16位，所以读取只需要读取TIM6的32位寄存器。TIM7与TIM8也一样。TIM6_TCNT和TIM8_TCNT的高16位寄存器为只读属性，如果想在计数之前写高16位计数值，需要写TIM5_TCNT和TIM7_TCNT。

注2：级联时，开启低位TIM(TIM6和TIM8)的STR@CR位可以同时开始两个Timer。

19.4 寄存器

TIM 模块寄存器列表 (基地址:0x0x4000 0400)

地址	寄存器名	说明
0x4000 0400	CR	TIM 控制寄存器
0x4000 0404	TCNT	TIM 计数寄存器
0x4000 0408	TPR	TIM 周期寄存器
0x4000 040C	PSQ	TIM 预分频寄存器
0x4000 0410	TIMINTF	TIMx 中断标志和清除寄存器

19.4.1 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				TC	ETEN	TRIGEN	保留	IE	CLKS[1:0]		OPM	保留		STR
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=5,7，只在级联的高位 TIMER 有效)** 0：禁止 TIM5/TIM6 或者 TIM7/TIM8 级联功能 1：使能 TIM5/TIM6 或者 TIM7/TIM8 级联功能 注：只有在级联的两个 TIMER 都停止计数时才能修改此位
14 ~ 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=5,6,7,8)口。
9	ETEN	**TIMx 溢出事件使能位** 0：禁止定时器溢出唤醒 CM3 内核 1：允许定时器溢出唤醒 CM3 内核 注：级联之后为 32 位溢出时才允许唤醒
8	TRIGEN	**TIMx 溢出触发外部模块使能位** 0：定时器 x 溢出禁止触发其他模块 1：定时器 x 溢出允许触发其他模块 注 1：其他模块有 DMA、ADC 注 2：所有模块都能触发 DMA，但是只有 TIM8 能够触发 ADC
7	保留	-
6	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
5 ~ 4	CLKS[1:0]	**TIMx 时钟源（ICLK）选择位** 00：内部 APB1 时钟（PCLK1） 01：Tx(x=5,6,7,8)口输入，自动上拉 10：内部低频 RC 振荡器（LSICLK）（TIM7 和 TIM8 才有此位，TIM5 和 TIM6 无此位） 11：保留 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

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=5,6,7,8)** 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

19.4.2 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 TIM6 和 TIM8 位有效** 注：只有级联时有效，例如 TIM5.TCNTL 与 TIM6.TCNTH 的读写值都一样。
15 ~ 0	**TCNTL[15:0]**	**定时器计数值低 16 位：全部 TIM 有效**

注：此寄存器在定时器停止时，被自动清零，且只清一次。

19.4.3 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 TIM6 和 TIM8 位有效** 注：只有级联时有效，例如 TIM5.TCNTL 与 TIM6.TCNTH 的读写值都一样。
15 ~ 0	**TPRL[15:0]**	**定时器自动重载寄存器** 注：低 16 位，全部 TIM 有效

19.4.4 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 TIM6 和 TIM8 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))

19.4.5 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

20. 数学协处理单元（Math Co-Processor）

20.1 简介

SH32F284的数学协处理单元（MACP）包含一个32位CORDIC运算单元（简称为CORDIC）、一个扩展的硬件IQ格式 32位除法器（简称为IQDIV）、以及一个电机专用的SVPWM增强引擎（简称为SVPWM）。

CORDIC单元可实现基于圆函数的旋转与向量运算，尤其可快速完成高精度的正余弦运算。

IQDIV除法器可用硬件快速完成32位以内的有/无符号定点除法运算，较之CPU软件实现的IQ除法运算有极大速度提升。

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。

20.2 寄存器列表

MACP 模块寄存器列表 (基地址:0x0x4004 5400)

地址	寄存器名	说明
0x4004 5400	CORDCSR0	CORDIC控制状态寄存器组0
0x4004 5404	OPRDX0	CORDIC运算操作数X寄存器0
0x4004 5408	OPRDY0	CORDIC运算操作数Y寄存器0
0x4004 540C	OPRDZ0	CORDIC运算操作数Z寄存器0
0x4004 5410	CORDCSR1	CORDIC控制状态寄存器组1
0x4004 5414	OPRDX1	CORDIC运算操作数X寄存器1
0x4004 5418	OPRDY1	CORDIC运算操作数Y寄存器1
0x4004 541C	OPRDZ1	CORDIC运算操作数Z寄存器1
0x4004 5500	IQDIVCSR0	IQ格式除法控制状态寄存器0
0x4004 5504	IQDIVDND0	IQ格式除法被除数寄存器0
0x4004 5508	IQDIVSOR0	IQ格式除法除数寄存器0
0x4004 550C	IQDIVRLT0	IQ格式除法结果寄存器0
0x4004 5510	IQDIVCSR1	IQ格式除法控制状态寄存器 1
0x4004 5514	IQDIVDND1	IQ格式除法被除数寄存器1
0x4004 5518	IQDIVSOR1	IQ格式除法除数寄存器1
0x4004 551C	IQDIVRLT1	IQ格式除法结果寄存器1
0x4004 5600	SVCON	SVPWM运算启动标志寄存器
0x4004 5604	SVUALPHA	SVPWM输入Ualpha值寄存器
0x4004 5608	SVUBETA	SVPWM输入Ubeta值寄存器
0x4004 560C	SVIQN	SVPWM数据格式配置寄存器
0x4004 5610	SVTA	SVPWM输出Ta值寄存器
0x4004 5614	SVTB	SVPWM输出Tb值寄存器
0x4004 5618	SVTC	SVPWM输出Tc值寄存器
0x4004 561C	SVSECTOR	SVPWM象限寄存器

20.3 CORDIC 运算单元

20.3.1 主要特性

- 基于CORDIC算法（圆函数）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，可支持圆函数的全范围取值
- 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

20.3.2 功能描述

20.3.2.1 CORDIC 计算单元

SH32F284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）**，可用来解圆函数（三角函数）等。

一般CORDIC算法对应如下CORDIC方程：

$$x_{i+1} = x_i - d_i \cdot y_i \cdot 2^{-i}$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i}$$

$$z_{i+1} = z_i - d_i \cdot e_i$$

旋转模式下 $d_i = sign\ (z_i), z_i \to 0$

向量模式下 $d_i = sign\ (y_i), y_i \to 0$

$$e_i = \arctan(2^{-i})$$

20.3.2.2 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，用来计算圆函数(三角函数)[注]，可通过CORDCON控制寄存器设置旋转或向量模式。

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：即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。

20.3.2.3 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。

20.3.2.4　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。

20.3.2.5　操作模式和对应结果数据

CORDIC方程如下，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

圆函数 $e_i = \arctan(2^{-i})$	**旋转模式**	**向量模式**
	$d_i = sign(z_i), z_i \to 0$	$d_i = sign(y_i), y_i \to 0$
	$X_{result} = K[X\cos(Z) - Y\sin(Z)]$ $Y_{result} = K[X\sin(Z) + Y\cos(Z)]$ $Z_{result} = 0$ 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

20.3.3 CORDIC 寄存器

CMDSU模块所使用寄存器如下：

功能	名称	寄存器描述
操作控制寄存器	CORDCR0	用于设定组0的CORDIC运算的参数，启动，运行标志等。
	CORDCR1	用于设定组1的CORDIC运算的参数，启动，运行标志等。
操作数寄存器	OPRDX0	组0的操作数寄存器，运算前存放CORDIC运算X操作数；运算后存放CORDIC运算X结果
	OPRDY0	组0的操作数寄存器，运算前存放CORDIC运算Y操作数；运算后存放CORDIC运算Y结果
	OPRDZ0	组0的操作数寄存器，运算前存放CORDIC运算Z操作数；运算后存放CORDIC运算Z结果（一般为0，保留此寄存器）
	OPRDX1	组1的操作数寄存器，运算前存放CORDIC运算X操作数；运算后存放CORDIC运算X结果
	OPRDY1	组1的操作数寄存器，运算前存放CORDIC运算Y操作数；运算后存放CORDIC运算Y结果
	OPRDZ1	组1的操作数寄存器，运算前存放CORDIC运算Z操作数；运算后存放CORDIC运算Z结果（一般为0，保留此寄存器）

20.3.3.1 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 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								MODE	FORMAT[1:0]		保留	XYMRS	KADJ	OVF	RUN
-								*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 ~ 8	保留	-
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。

20.3.3.2 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

20.3.3.3 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

20.3.3.4 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

20.4 硬件 IQ 除法单元

SH32F284除了Cortex-M3内核具有硬件32位除法器外，更扩展了专用的硬件IQ除法单元，用于加快高精度定点除法运算。

20.4.1 IQ 格式除法

IQ格式是用整数定标的方法确定小数，从而使浮点运算转换为定点运算，是电机控制中的常用方法。本运算单元直接提供IQ格式的除法，提高除法运算的运行效率。

- 支持两组独立的寄存器，可以同时启动。
- 支持Q0~31格式，根据需要的精度选择合适的Q格式
- 支持有符号运算及无符号运算
- 支持对运算结果自动作四舍五入或截尾处理
- 支持对运算结果的自动饱和处理。

进行IQ格式除法时，MACP_IQDIVDND用于存放除法操作的被除数；MACP_IQDIVSOR用于存放除法操作的除数，Q@MACP_IQDIVCR用于存放除法的格式(即结果的精度，范围为0~31)，结果为一个32位的商存放于MACP_ IQDIVRLT 中。

此除法单元共有两组寄存器，两组寄存器独立设置并启动，但实际处理是串行的，即同时只能处理一组计算配置，另一组必须等上一组处理完后才会执行。假设组0启动时正在计算组1，则组0不会马上执行，而是等待组1处理完后再执行。在此期间组0的RUN状态一直为1。反之如果组1启动时正在计算组0，则组1不会马上执行，而是等待组0处理完后再执行。在此期间组1的RUN状态一直为1。

- 写除数到寄存器MACP_IQDIVSOR，写被除数到寄存器MACP_IQDIVDND, 数据格式到Q@MACP_IQDIVCR。
- 设置寄存器SIGN@MACP_IQDIVCR = 0选择无符号除法操作，操作数用原码表示；SIGN@MACP_IQDIVCR = 1选择有符号除法操作，操作数用补码表示，最高位表示符号位；写1到位RUN@MACP_IQDIVCR启动运算,此位在运行期间保持为1直到运算结束,软件只能写1不能写0(写0忽略)。SAT@MACP_IQDIVCR位为饱和位(即当运算溢出或除0后，由硬件自动将结果设为饱和值且将此Flag置为1)，使用中由软件负责清0。
- 设置操作参数与启动可在一条指令中完成。
- 由于除法操作需要多个系统周期执行时间（17个周期）.启动除法后，RUN@MACP_IQDIVCR保持为1表示未完成，直到结果寄存器有效后自动变为0，所以软件可以利用此位作为运算是否结束的判断标志。
- 当硬件完成操作后，会自动将RUN@MACP_IQDIVCR清0。此时，可以从相应寄存器中读取结果。
- 读取操作不会改变寄存器中值。在RUN@MACP_IQDIVCR为1时任何对寄存器的写入动作都无效，由硬件保证。
- 运算时由于输入的被除数是32位，所以要将输入的32位被除数作为64位除法的高32位，低32位补0。然后除以32位除数，得到64位的商。然后根据指定的数据格式Q@MACP_IQDIVCR对结果进行处理。
- 对于有符号除法运算，输入的32位数是有符号数(包括被除数及除数)，以补码表示，结果也应是带符号数
- 对于无符号除法运算，输入的32位数是无符号数(包括被除数及除数)，以原码表示，结果也应是无符号数
- 如果结果溢出就置SAT@MACP_IQDIVCR为1。RUN@MACP_IQDIVCR位清0，结果为饱和值。溢出情形见下表。
- 如果除数为0就置SAT@MACP_IQDIVCR为1。RUN@MACP_IQDIVCR位清0，结果为饱和值。
- CMOD@MACP_IQDIVCR用作指定在选取结果时是否作四舍五入。如果为1，会直接丢弃多余的尾数，如果为0需要作四舍五入以提高数据的精度。

20.4.2 注意事项

(1) 在硬件执行除法操作没有结束时读取运算结果（即读取寄存器MACP_IQDIVRLT），会得到不可预知的结果。但是，读取动作不会影响硬件继续运算并得到正确结果（当硬件完成运算后，仍然可以读取到正确的结果）。

(2）如果除法操作中除数为0，则不会执行除法操作，操作数寄存器保持原有值不变。硬件自动将RUN@MACP_IQDIVCR位清0并置位SAT@MACP_IQDIVCR标志，其余位保持不变，不会影响用户程序继续运行。SAT@MACP_IQDIVCR异常标志位由软件清0。

(3) 在一组运算未完成时（RUN@MACP_IQDIVCR = 1），任何对于该组IQ寄存器的的写入操作都是无效的，另一组不受影响。例如：组0的RUN Flag为1时，对组0寄存器的任何写操作都无效。但是可以正常读写组1的寄存器。

(4)两组寄存器在使用时是完全独立的，但是执行时还是顺序执行的，即必须等一组计算完后才进行另一组计算，这个等待由硬件完成，无需软件参与。

20.4.3 IQDIV 寄存器

20.4.3.1 IQ 格式除法控制状态寄存器 0 (MACP_IQDIVCSRn)(n=0..1)

偏移地址: 0x0100

:0x0110

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							CMOD	Q[4:0]					SAT	SIGN	RUN
-							*RW*	*RW*					*RW*	*RW*	*RW1s*
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	CMOD	**结果处理方式选择位** 0:小数做四舍五入处理（默认为高精度的四舍五入） 1:小数直接舍弃
7 ~ 3	Q[4:0]	**设置Q格式，最大为31**
2	SAT	**饱和标志** 如果结果为饱和值就置此位。 当遇到除0，符号位溢出，32位结果溢出后，结果寄存器就设为饱和值且此位设为1，否则为0。 无符号数除法时饱和值为0xFFFFFFFF 当有符号数除法时看结果的符号位（被除数或除数只有一个是负数时，结果为负），如果是负数饱和值为0x80000000,如果是正数则饱和值为0x7FFFFFFF。
1	SIGN	**选择有符号除法** 0：无符号除法(操作数为原码) 1：有符号除法(操作数为补码)
0	RUN	**IQDIV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IQDIV运算，运算过程中标志为1,运算完成后标志由硬件清0

20.4.3.2 IQ 格式除法被除数寄存器 0 (MACP_IQDIVDNDn)(n=0..1)

偏移地址: 0x0104

:0x0114

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
IQDIVDND[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
IQDIVDND[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	IQDIVDND[31:0]	**被除数，32位，当有符号数除法时需要补码，无符号除法时需要原码**

20.4.3.3 IQ 格式除法除数寄存器 0 (MACP_IQDIVSORn)(n=0..1)

偏移地址: 0x0108

:0x0118

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
IQDIVSOR[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
IQDIVSOR[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	**IQDIVSOR[31:0]**	**除数，32位，当有符号数除法时需要补码，无符号除法时需要原码**

20.4.3.4 IQ 格式除法结果寄存器 0 (MACP_IQDIVRLTn)(n=0..1)

偏移地址: 0x010C

:0x011C

复位值: 0x0000 0000

b31	b30	b29	b28	b27	b26	b25	b24	b23	b22	b21	b20	b19	b18	b17	b16
IQDIVRLT[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
IQDIVRLT[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	**IQDIVRLT[31:0]**	**结果，32位，当有符号数除法时需要补码，无符号除法时需要原码**

20.5 电机专用 SVPWM 增强引擎

SH32F284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。

20.5.1 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。

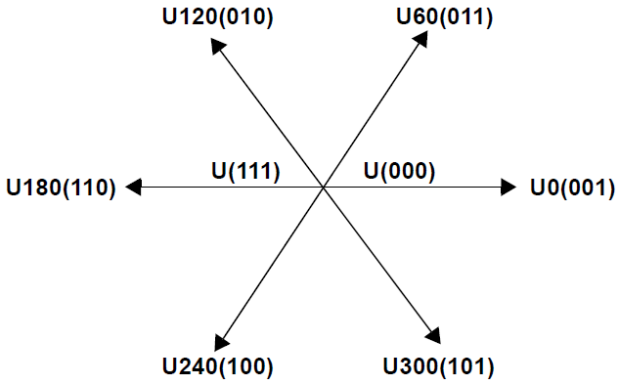


图20-1　　空间矢量调制

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)

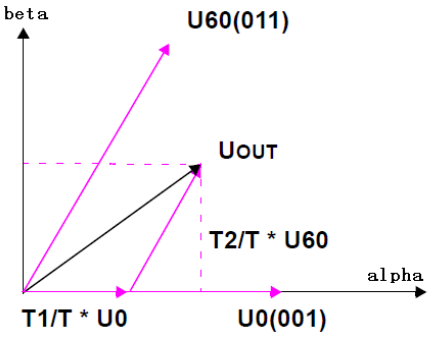


图20-2　　平均空间矢量调制

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。

20.5.2 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：

$$\begin{cases} V_{r1\beta}=U \\ V_{r2\beta\alpha}=(-U\ +\sqrt{3}*U\ )/2 \\ V_{r3\beta\alpha}=(-U\ -\sqrt{3}*U\ )/2 \end{cases}$$

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：

$$\begin{cases} X=U_{\beta} \\ Y=(U_{\beta\alpha}+\sqrt{3}*U\ )/2 \\ Z=(U_{\beta\alpha}-\sqrt{3}*U\ )/2 \end{cases}$$

对于不同扇区T1,T2值，有如下对应关系：

表20-1　　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。

20.5.3　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。

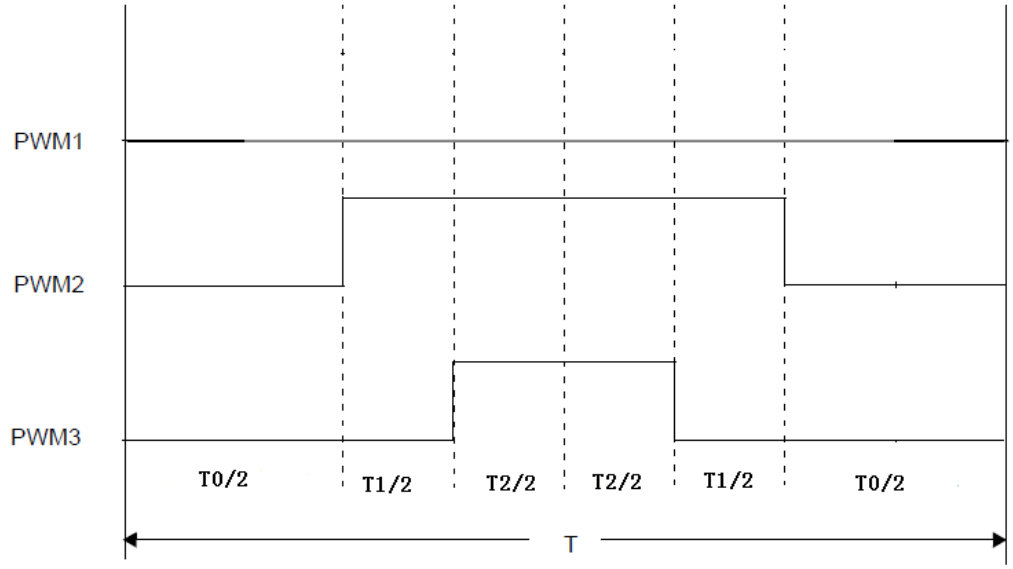


图20-3　　周期 T 内的 PWM

SH32F284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。

20.5.4　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。

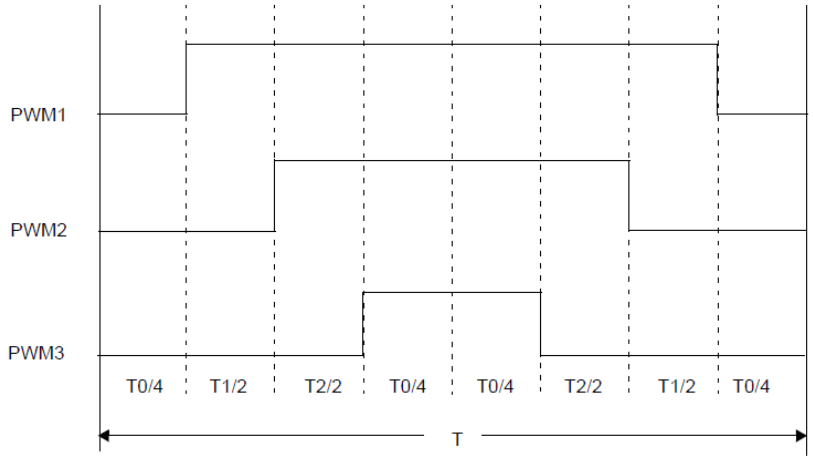


图20-4　　周期 T 内的 PWM

SH32F284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。

20.5.5　SVPWM 寄存器

20.5.5.1　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

20.5.5.2　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**

20.5.5.3 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**

20.5.5.4 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
保留							SVTYP	SVIQN[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 ~ 9	保留	-
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**

20.5.5.5 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**

20.5.5.6 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**

20.5.5.7 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**

20.5.5.8 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**

21. 模数转换器（Analog to Digital Converter）

21.1 简介

SH32F284包含两个单端型12位高速逐次逼近型模数转换器（ADC1、ADC3），模块图如图21-1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端口的输入或者内建VREF作为基准电压。

ADC1最多有11路模拟输入通道（ADC1.AN0 –ADC1.AN7（注），OPxOUT(x=2,3)，VREF）；ADC3最多有12路模拟输入通道（ADC3.AN0 –ADC3.AN7，OPxOUT(x=1, 3)，VREF，TPS）。2个ADC都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 - 7）中，每转换一次序列，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。

注：不同封装的产品ADC输入引脚略有差异，ANx编号不一定连续，有的ADC输入通道没有对应外部引脚，具体请参考"引脚定义"，此处对具体通道不再单列，统一按照最多AN0～AN7标注。。

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。

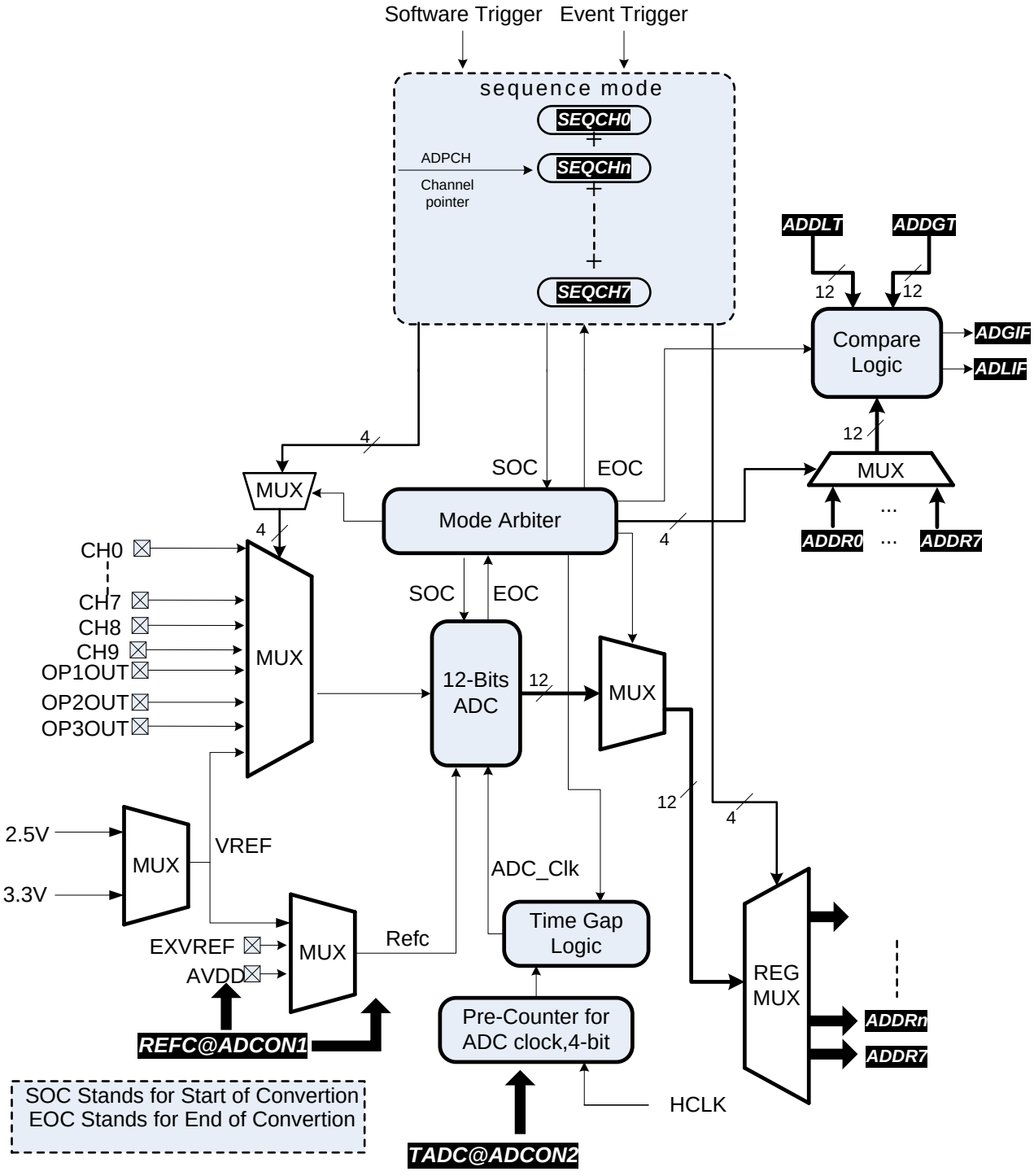


图21-1　　ADC 系统框图

21.2 主要特性

- SH32F284内置2个逐次逼近方式的12位A/D转换器
- 参考电压可选内建VREF，外接VREF或AVDD，并且内建VREF可以输出给其它模块，例如OP、比较器使用。
- 内建参考电压Vref为3.3V和2.5V两种可选
- 16路模拟输入口、3个OP输出口、一个内建参考电压、一个温度传感器，共21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8个通道，转换结果保存在8个结果寄存器中
- 存在三种转换方式：序列转换方式、间断转换方式和连续转换方式
- 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、三个GPT定时器、一个普通定时器(TIM8)和两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
- 三种转换方式都带比较功能
- 带DMA功能

21.3 功能描述

21.3.1 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[2:0])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ADCON2中的ADCIF位将由硬件置起，此时若ADCON2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[2:0] = 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。

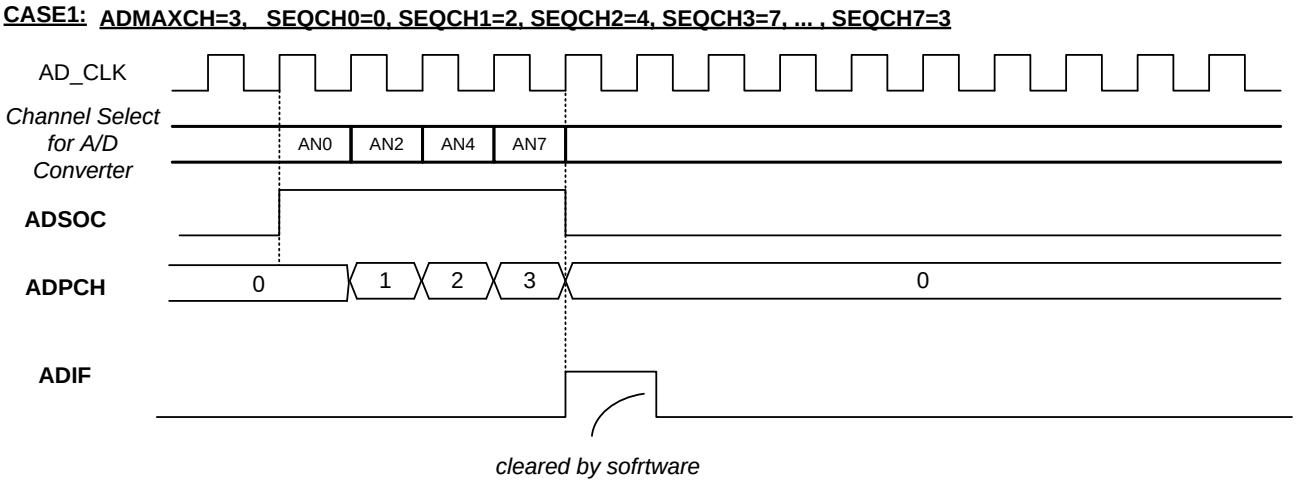


图21-2　　单次转换方式波形示例图

21.3.2 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[2:0])设定值时，本次序列转换结束。ADC采样通道序号最大为7，因此当转换通道指针寄存器ADPCH超过7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，SEQCH6=1，SEQCH7=0，当ADMAXCH=3时，若当前ADPCH=6，在间断转换方式下，SEQCH6，SEQCH7，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ADDR6，ADDR7，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21-3所示：

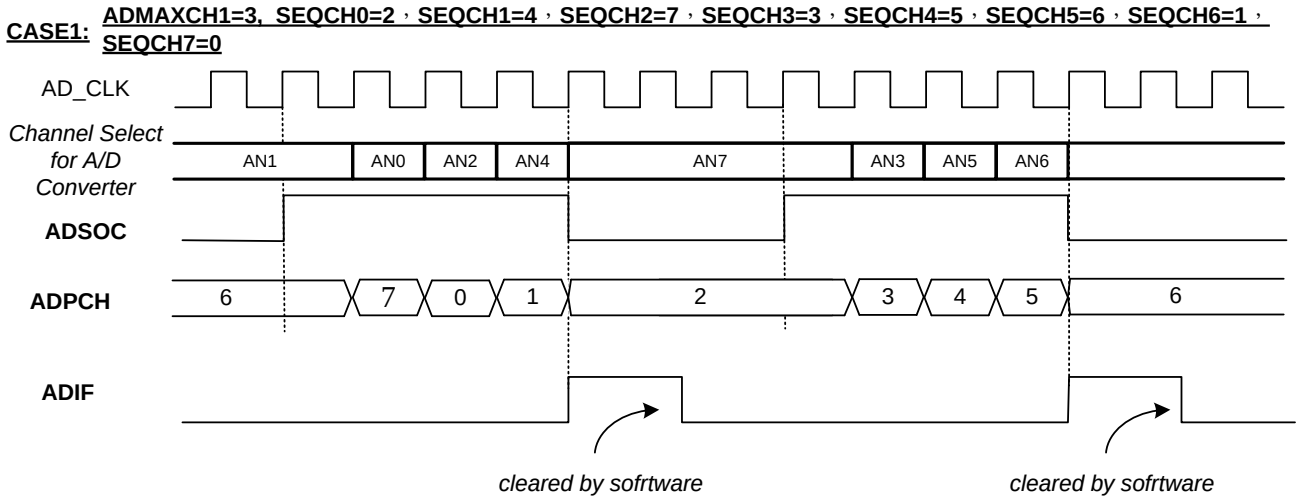


图21-3　　单序列顺序采样模式，间断转换方式的波形示例图

注1：ADPCH只能在ADC不采样（ADSOC等于0）时，能够被软件写入，并且只有在间断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。

21.3.3　连续转换方式

ADCTU[1:0] = 1x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21-4所示：

注意：ADPCH的值在采样结束时被清零。

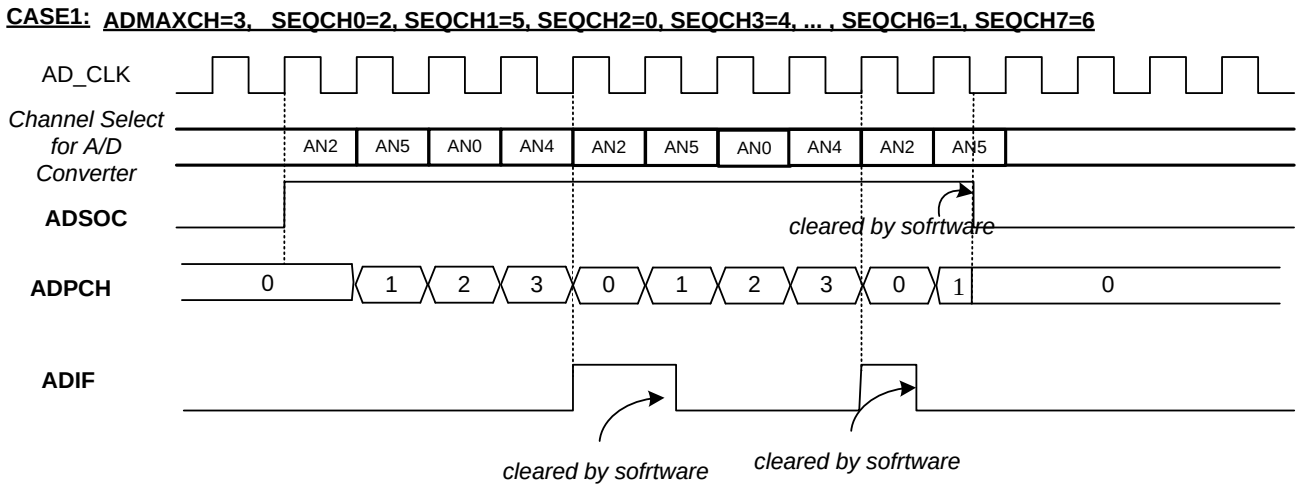


图21-4　　连续转换方式波形示例图

21.3.4　其它功能描述

21.3.4.1　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 - 7），寄存器SEQCHx有8组，所以序列转换时，一次最多可转换8路通道。而每次转换的通道数量由寄存器ADCON2中的ADMAXCH[2:0]的值决定。例：ADMAXCH[2:0] = 0，即单通道转换，将SEQCH0中存放的通道进行转换；ADMAXCH[2:0] = 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 - 7）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[2:0] = 2，SEQCH0 = 8，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_AFRH/L设成AF14）。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。

21.3.4.2　序列转换模式ADC结果寄存器ADDRx（x = 0 - 7）

序列转换的结果依次储存在结果寄存器ADDRx（x = 0 - 7）中，结果寄存器为只读寄存器，为右对齐的方式。例如当ADC的基准电压选择AVDD是，正的采样结果最大值0x0FFF代表(4095/4096)*AVDD，如下图。一个序列转换完毕，结果寄存器ADDRx中的值也更新了一次。

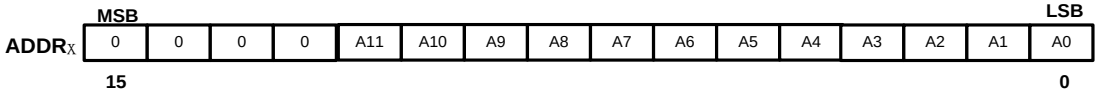


图21-5　　结果寄存器示例图

21.3.4.3　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[3:0] = 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。

21.3.4.4　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ADCON2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21.3.5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5种方式，外部引脚下降沿触发启动、MCM1模块触发信号触发启动、GPT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。具体通过设置ADSTRS@ADCx->ADCON1位、CMPx@MCM1/2->PWMCON2、ADTRxxEN@GPTx->GTINTAD和TRIGEN@TIM8->CR。

硬件触发信号优先于软件触发信号。当软件已经处于序列转换中时，硬件触发会终止先前的序列转换而重新启动一次序列转换，转换值将会覆盖先前的结果。

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，则前一次转换终止，重新开始一次转换。

注3：在转换过程中，即ADSOC为1时，对ADCON2、SEQCHSEL和ADGAPON的写入都将视为无效操作。另外，只有在ADSOC=0并且ADCTU@ADCON1=1时，ADPCH才能被修改。

21.3.4.5 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。

21.3.5 比较功能

21.3.5.1 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。以图21-6的配置为例，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：*在写入比较值ADDGT和ADDLT时，应注意结果寄存器的存储方式是左对齐还是右对齐，数据写入格式应和结果寄存器的存储格式一致。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。*

21.3.5.2 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21-6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21-6的右半部分所示。

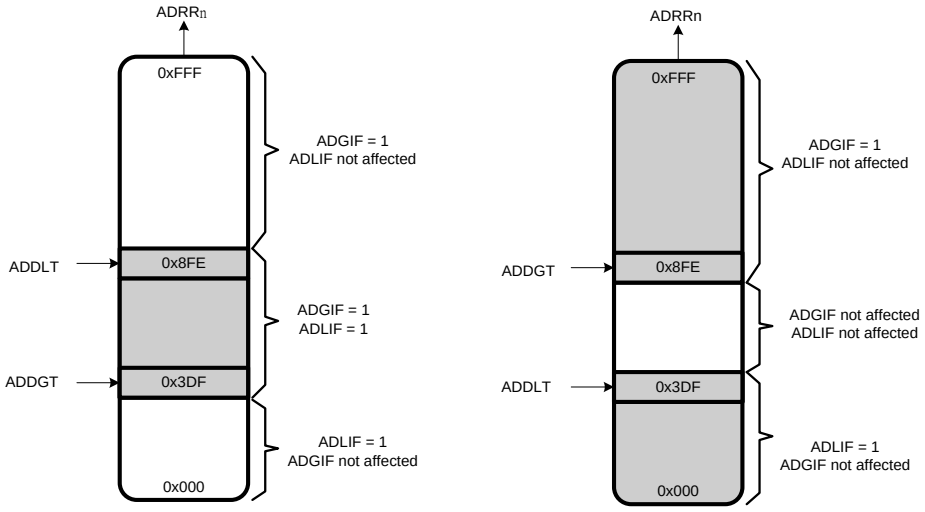


图21-6 比较过程图

21.3.5.3 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。

21.3.6 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}。对于10BIT模式AD转换每个通道的AD转换时间固定为13 * t_{AD}。因此每个通道的总共转换时间 = t_{SAMP} + 13 * t_{AD}。

ADC时钟最快设置成40MHz。

21.3.7 ADC 模块参考电压的设置

可以选择芯片AV_{DD}、内建V_{REF}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AV_{DD}。令寄存器ADCON1中的REFC[1:0]位置1将使ADC模块使用V_{REF}引脚上的电压作为参考电压。

21.3.8 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。

21.3.9 DMA 请求

能够通过DMA存储ADC采样结果，在所有通道采样结束后，触发DMA功能，实现连续转换而不需要软件干预。可以通过ADDE@ADNCON1位来选择是否开启此功能。

21.3.10 AD 转换时对传感器输出阻抗的要求

如图21-7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SH32F284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（SH32F284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：

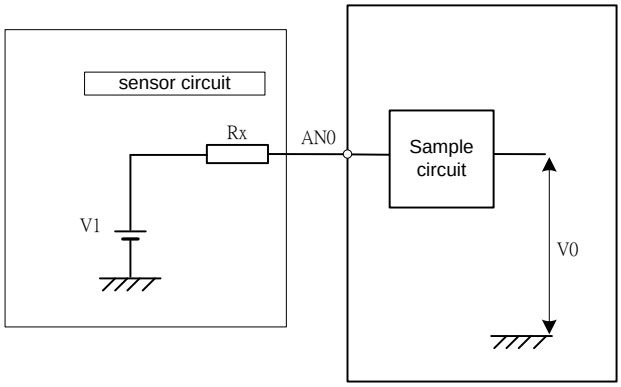


图21-7 输出阻抗

$$Rx \le -\frac{T}{5\times10^{-12} * Ln\frac{V1-V0}{V1}} - 0.8\times10^{3}$$

上式也可表示为：

$$Rx \le -\frac{T}{5\times10^{-12} * Ln\frac{N}{M}} - 0.8\times10^{3}$$

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。

21.3.11 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。

ADC转换开启后，不能更改通道参数，包括TGAP、GAPENx、TS、ADMAXCH、TADC、ADPCH、SEQCHx等这些核心通道参数，这是由硬件保证的。

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OPUT1、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。

注1：在单次和间断转换过程中，即ADSOC为1时，对除了ADCON1.ADON、ADCON2、ADDGT、ADDLT以外的所有ADC寄存器的写入都将视为无效操作。

注2：在连续转换过程中，即ADSOC为1时，对除了ADCON1.ADON、ADCON2、ADCMPCON、ADDGT、ADDLT以外的所有ADC寄存器的写入都将视为无效操作。

21.4 寄存器

ADC 模块寄存器列表 (基地址:0x0x4004 1000)

地址	寄存器名	说明
0x4004 1000	ADCON1	ADC 控制寄存器 2
0x4004 1004	ADCON2	ADC 控制寄存器 1
0x4004 1008	ADPCH	ADC 采样转换通道指针寄存器
0x4004 100C	ADDR0	ADC 结果寄存器 0
0x4004 1010	ADDR1	ADC 结果寄存器 1
0x4004 1014	ADDR2	ADC 结果寄存器 2
0x4004 1018	ADDR3	ADC 结果寄存器 3
0x4004 101C	ADDR4	ADC 结果寄存器 4
0x4004 1020	ADDR5	ADC 结果寄存器 5
0x4004 1024	ADDR6	ADC 结果寄存器 6
0x4004 1028	ADDR7	ADC 结果寄存器 7
0x4004 102C	ADCMPCON	ADC 比较控制寄存器
0x4004 1030	ADDGT	ADC 上限比较寄存器
0x4004 1034	ADDLT	ADC 下限比较寄存器
0x4004 1038	SEQCHSEL	ADC 通道选择寄存器
0x4004 103C	ADGAPON	ADC 通道 GAP 控制位
0x4004 1040	ADINTF	ADC 中断标志和清除寄存器

21.4.1 ADC 控制寄存器 2 (ADCx_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
ADON	ADSTRS[6:0]							ADIE	ADDE	REFC[1:0]		MOD	ADCTU[1:0]		ADSOC
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	**ADON**	**ADC 允许位** 0：禁止 ADCx 模块 1：允许 ADCx 模块
14 ~ 8	**ADSTRS[6:0]**	**ADC 触发源选择位：** 为 ADC 的通道序列启动转换使能位。 见表 21.1
7	**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
6	**ADDE**	**通道序列转换结束 DMA 触发使能位：** 0：禁止 ADC 转换结束触发 DMA 1：使能 ADC 转换结束触发 DMA
5 ~ 4	**REFC[1:0]**	**基准电压选位** 00：选择 AVDD 为基准电压 01：选择外部 VREF 端口输入为基准电压 10：选择内建的 VREF 为基准电压 11：保留

		注:因为内建 VREF 和外部 VREF 共用引脚,3 个 ADC 不能既能选择内建 VREF 又选外接 VREF
3	**MOD**	**ADC 工作模式选择位：** 0：12bit 1：10bit
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：保留 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

21.4.2 ADC 控制寄存器 1 (ADCx_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													TGAP [2:0]		
-													*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				TS [3:0]				保留	ADMAXCH[2:0]			TADC[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 ~ 19	**保留**	**-**
18 ~ 16	**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
15 ~ 12	**保留**	**-**
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 21.3.10; 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；10bit 模式一个通道的总共采样转换时间 = 13tAD + 采样时间
7	保留	-
6 ~ 4	**ADMAXCH[2:0]**	**ADC 通道序列的总长度设置位（设置值 0~7）：** 一次 AD 转换过程的模拟通道总数为 ADMAXCH+1
3 ~ 0	**TADC[3:0]**	**ADC 时钟周期选择** 0000：ADC 时钟周期 tAD = 1 tHCLK 0001：ADC 时钟周期 tAD = 2 tHCLK 0010：ADC 时钟周期 tAD = 3 tHCLK 0011：ADC 时钟周期 tAD = 4 tHCLK 0100：ADC 时钟周期 tAD = 5 tHCLK 0101：ADC 时钟周期 tAD = 6 tHCLK 0110：ADC 时钟周期 tAD = 8 tHCLK 0111：ADC 时钟周期 tAD = 12 tHCLK 1000：ADC 时钟周期 tAD = 16 tHCLK 1001：ADC 时钟周期 tAD = 24 tHCLK 1010：ADC 时钟周期 tAD = 32 tHCLK 1011：ADC 时钟周期 tAD = 48 tHCLK 1100：ADC 时钟周期 tAD = 64 tHCLK 1101：ADC 时钟周期 tAD = 128 tHCLK 1110：ADC 时钟周期 tAD = 256 tHCLK 1111：ADC 时钟周期 tAD = 320 tHCLK 注：tHCLK 为 HCLK 的时钟周期

21.4.3 ADC 采样转换通道指针寄存器 (ADCx_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[2:0]		
-													*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 ~ 0	**ADPCH[2:0]**	**ADC 采样转换通道指针寄存器** 000：ADC 当前采样通道为 SEQCH0 001：ADC 当前采样通道为 SEQCH1 010：ADC 当前采样通道为 SEQCH2 011：ADC 当前采样通道为 SEQCH3 100：ADC 当前采样通道为 SEQCH4 101：ADC 当前采样通道为 SEQCH5 110：ADC 当前采样通道为 SEQCH6 111：ADC 当前采样通道为 SEQCH7 注：ADPCH 只能在 ADC 不采样（ADSOC=0）且 ADCTU@ADCON1=1 时，才能被软件写入。

21.4.4 ADC 结果寄存器 0 (ADCx_ADDRn)(n=0..7)

偏移地址: 0x000C

:0x0010

:0x0014
:0x0018
:0x001C
:0x0020
:0x0024
:0x0028

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
ADDRn[15:0]															
RO															
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	**ADDRn[15:0]**	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

21.4.5 ADC 比较控制寄存器 (ADCx_ADCMPCON)

偏移地址: 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
保留								ADGIE	ADGDE	ADLIE	ADLDE	保留	CSEL[2:0]		
-								*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	**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	**ADGDE**	**上限比较 DMA 使能位** 0：禁止上限比较触发 DMA 1：使能上限比较触发 DMA
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	**ADLDE**	**下限比较 DMA 使能位** 0：禁止下限比较 DMA 1：使能下限比较 DMA
3	保留	-
2 ~ 0	**CSEL[2:0]**	**用于比较的结果寄存器选择位段** 0：选择 ADDR0 中的值与 ADDGT、ADDLT 比较 1：选择 ADDR1 中的值与 ADDGT、ADDLT 比较 2：选择 ADDR2 中的值与 ADDGT、ADDLT 比较 3：选择 ADDR3 中的值与 ADDGT、ADDLT 比较

		4：选择 ADDR4 中的值与 ADDGT、ADDLT 比较 5：选择 ADDR5 中的值与 ADDGT、ADDLT 比较 6：选择 ADDR6 中的值与 ADDGT、ADDLT 比较 7：选择 ADDR7 中的值与 ADDGT、ADDLT 比较

21.4.6 ADC 上限比较寄存器 (ADCx_ADDGT)

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
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。

21.4.7 ADC 下限比较寄存器 (ADCx_ADDLT)

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
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。

21.4.8 ADC 通道选择寄存器 (ADCx_SEQCHSEL)

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
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(ADC1 无此通道) 1001 - AN9(ADC1 无此通道) 1010 - Vref 1011 - OP1OUT(ADC1 无此通道) 1100 - OP2OUT(ADC3 无此通道) 1101 - OP3OUT 1110 - TPS(温度传感器)(ADC1 无此通道) 1111 - 保留
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(ADC1 无此通道) 1001 - AN9(ADC1 无此通道) 1010 - Vref 1011 - OP1OUT(ADC1 无此通道) 1100 - OP2OUT(ADC3 无此通道) 1101 - OP3OUT 1110 - TPS(温度传感器)(ADC1 无此通道) 1111 - 保留
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(ADC1 无此通道) 1001 - AN9(ADC1 无此通道) 1010 - Vref 1011 - OP1OUT(ADC1 无此通道) 1100 - OP2OUT(ADC3 无此通道) 1101 - OP3OUT 1110 - TPS(温度传感器)(ADC1 无此通道) 1111 - 保留
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(ADC1 无此通道) 1001 - AN9(ADC1 无此通道) 1010 - Vref 1011 - OP1OUT(ADC1 无此通道) 1100 - OP2OUT(ADC3 无此通道) 1101 - OP3OUT 1110 - TPS(温度传感器)(ADC1 无此通道) 1111 - 保留
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(ADC1 无此通道) 1001 - AN9(ADC1 无此通道) 1010 - Vref 1011 - OP1OUT(ADC1 无此通道) 1100 - OP2OUT(ADC3 无此通道) 1101 - OP3OUT 1110 - TPS(温度传感器)(ADC1 无此通道) 1111 - 保留
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(ADC1 无此通道) 1001 - AN9(ADC1 无此通道) 1010 - Vref 1011 - OP1OUT(ADC1 无此通道) 1100 - OP2OUT(ADC3 无此通道) 1101 - OP3OUT 1110 - TPS(温度传感器)(ADC1 无此通道) 1111 - 保留
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(ADC1 无此通道) 1001 - AN9(ADC1 无此通道) 1010 - Vref 1011 - OP1OUT(ADC1 无此通道) 1100 - OP2OUT(ADC3 无此通道) 1101 - OP3OUT 1110 - TPS(温度传感器)(ADC1 无此通道) 1111 - 保留
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(ADC1 无此通道) 1001 - AN9(ADC1 无此通道) 1010 - Vref 1011 - OP1OUT(ADC1 无此通道) 1100 - OP2OUT(ADC3 无此通道) 1101 - OP3OUT 1110 - TPS(温度传感器)(ADC1 无此通道) 1111 - 保留

21.4.9 ADC 通道 GAP 控制位 (ADCx_ADGAPON)

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
保留								GAPENy(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	**GAPENy(y=7~0)**	**ADC 通道 y 的 GAP 时间使能位：** 0：禁止 1：使能

21.4.10 ADC 中断标志和清除寄存器 (ADCx_ADINTF)

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													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

表21-1 ADC 触发源选择表

ADSTRS[6:0]					发生源	AD开始源	开始条件
-	-	-	-	-	软件	ADSOC	软件启动
000	x	x	x	1	外部引脚	ADTRG1	触发输入引脚1
	x	1	x	x		ADTRG3	触发输入引脚3
001	x	x	x	x	定时器8	BTIMTRG	定时器8溢出
010	x	x	x	1	MCM1	MCM1TRG1	MCM1. PWMCMP1比较匹配

	x	x	1	x		MCM1TRG2	MCM1. PWMCMP2比较匹配
	x	1	x	x		MCM1TRG3	MCM1. PWMCMP3比较匹配
	1	x	x	x		MCM1TRG4	MCM1. PWMCMP4比较匹配
100	x	x	x	1	GPT0	GTADTRA0N	GPT0.GTADTRA 的比较匹配
	x	x	1	x		GTADTRB0N	GPT0.GTADTRB 的比较匹配
101	x	x	x	1	GPT1	GTADTRA1N	GPT1.GTADTRA 的比较匹配
	x	x	1	x		GTADTRB1N	GPT1.GTADTRB 的比较匹配
110	x	x	x	1	GPT2	GTADTRA2N	GPT2.GTADTRA 的比较匹配
	x	x	1	x		GTADTRB2N	GPT2.GTADTRB 的比较匹配

注1：外部触发引脚（ADTRGx，x=1、3）固定为下降沿启动ADC，触发信号至少要持续2个HCLK时钟才有效。

22. 正交编码器接口模块（QEI）

22.1 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：

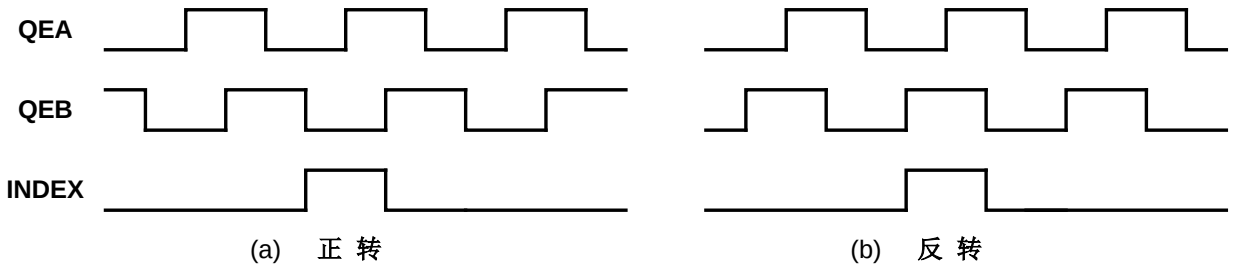


图22-1 增量式旋转编码器输出信号

下图提供了逻辑框图：

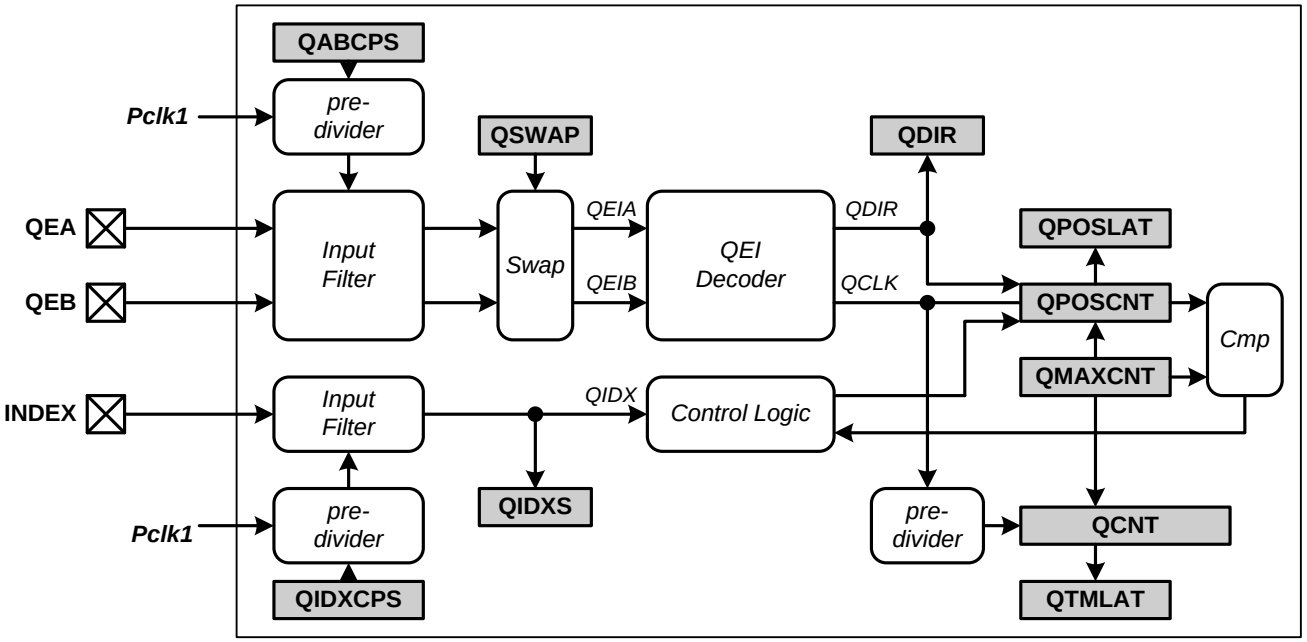


图22-2 正交编码器接口模块 QEI 逻辑框图

22.2 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

22.3 功能描述

22.3.1 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APB1时钟分频（PCLK1）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APB1时钟分频（PCLK1）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。

22.3.2 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。

22.3.2.1 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。

22.3.2.2 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。

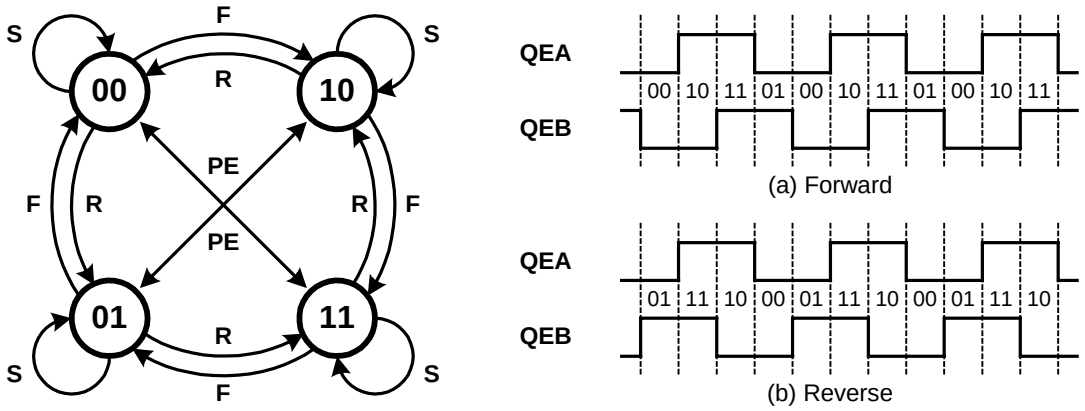


图22-3　　QEA/QEB 状态转换图

22.3.3　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。

22.3.3.1　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：

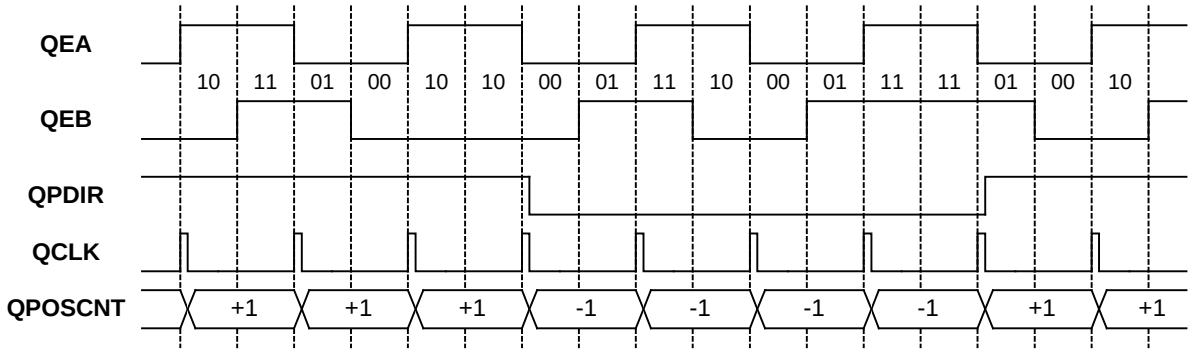


图22-4　　x2 计数模式下 QPOSCNT 的工作时序

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：

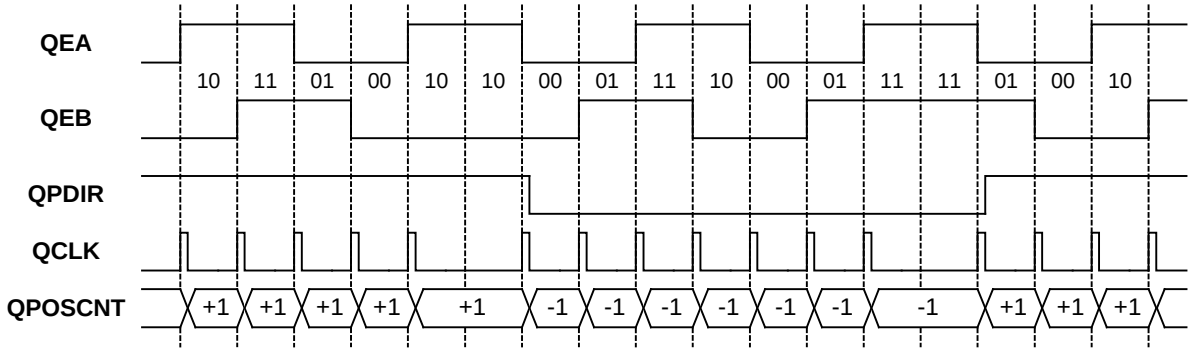


图22-5　　x4 计数模式下 QPOSCNT 的工作时序

22.3.3.2　　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）：

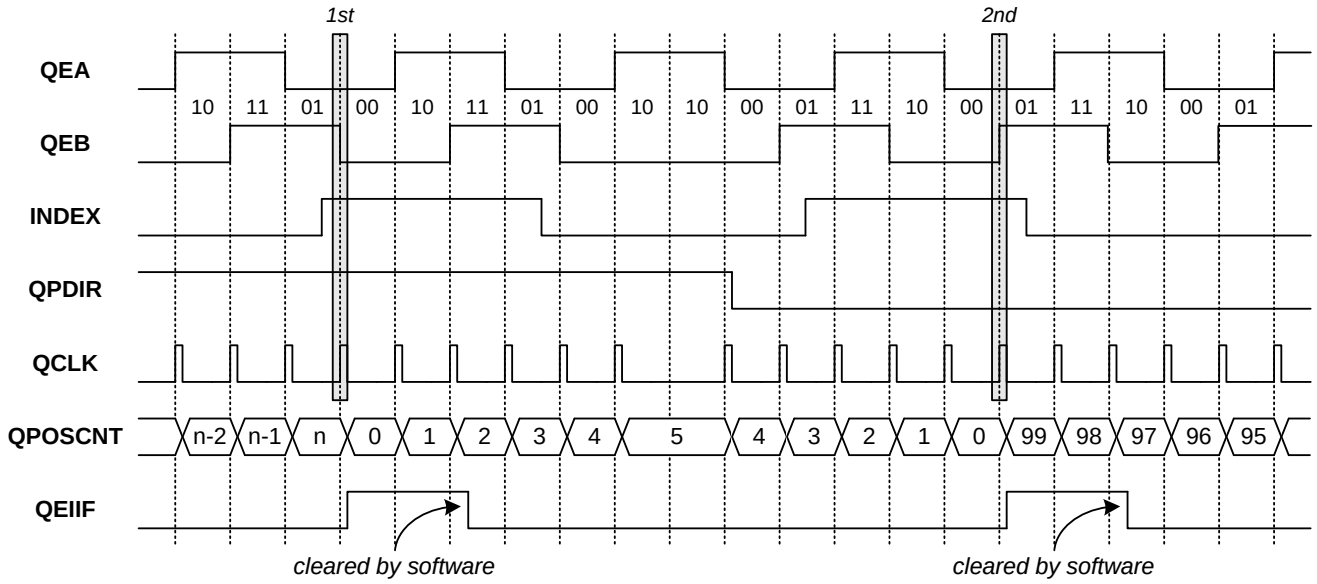


图22-6　　索引脉冲 INDEX 复位方式的工作时序(x4 计数模式)

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：

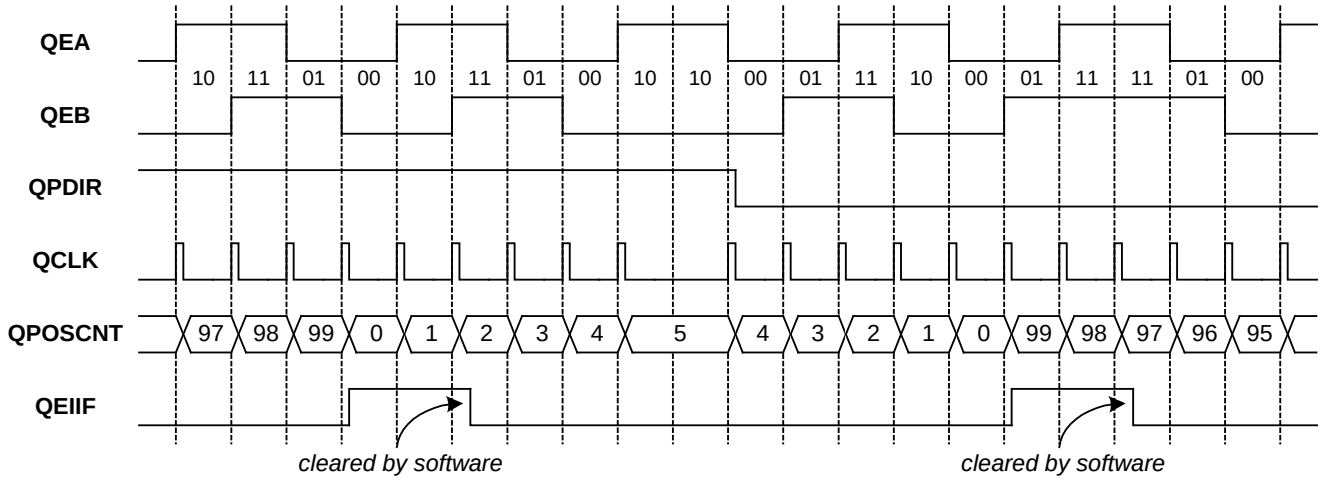


图22-7　　自动循环复位方式的工作时序(x4 计数模式)

22.3.4　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。

22.3.4.1　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）：

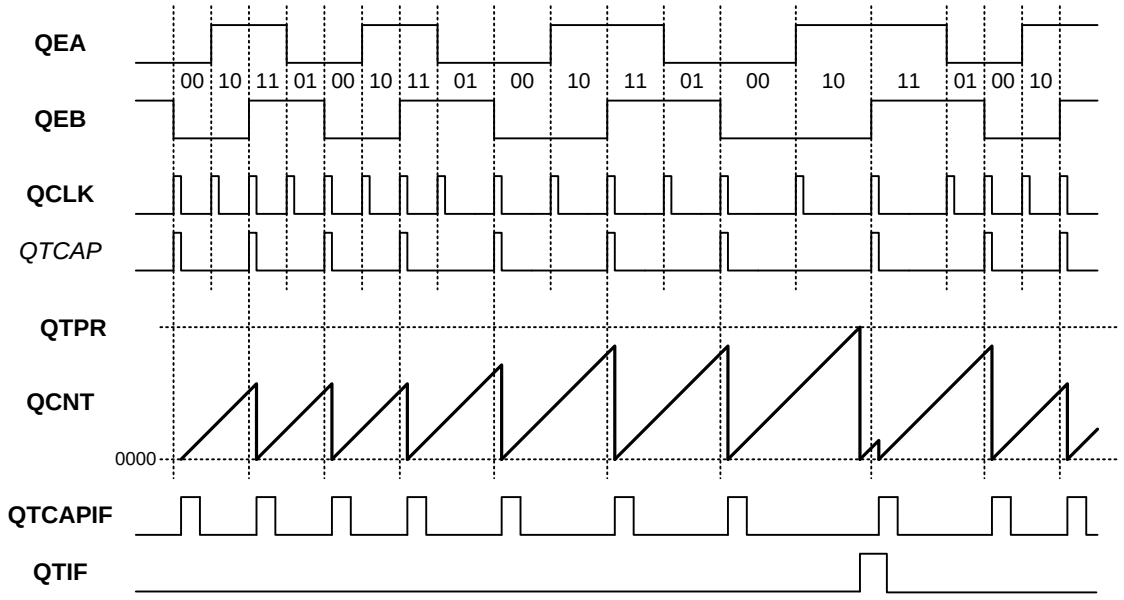


图22-8　　低转速测速应用方案中关键信号示意图(QTEPS=0001)

22.3.4.2　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。

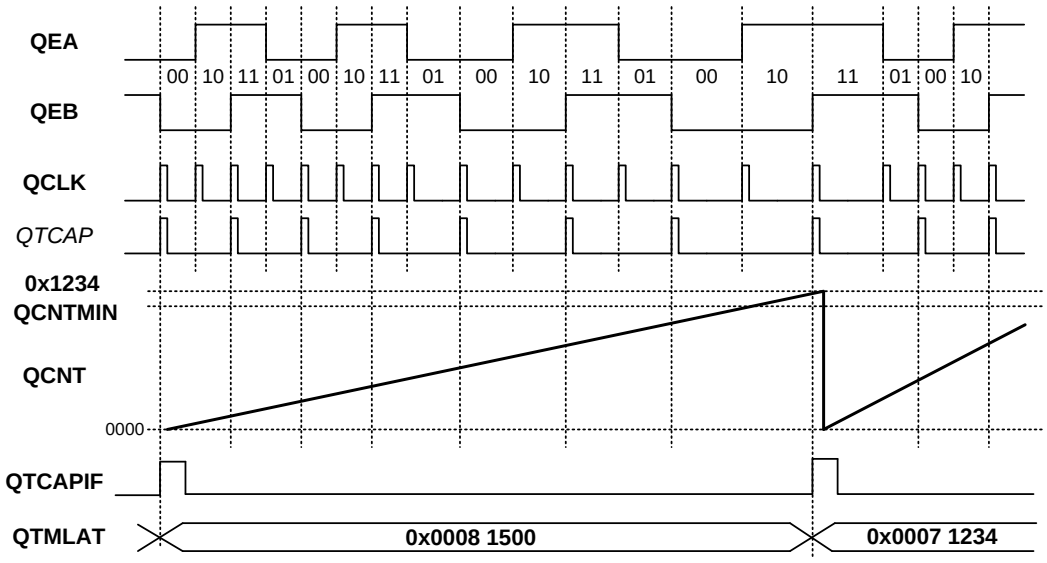


图22-9　　高速测速应用方案中关键信号示意图(QTEPS=0001)

22.3.5　QEI 模块的中断

22.3.5.1　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。

22.3.5.2　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。

22.3.5.3　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。

22.3.5.4　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。

22.3.5.5　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。

22.4 寄存器

QEI 模块寄存器列表 (基地址:0x0x4000 1C00)

地址	寄存器名	说明
0x4000 1C00	QEICON	QEI控制寄存器
0x4000 1C04	QFLTCON	QEI输入引脚滤波控制寄存器
0x4000 1C08	QPOSCNT	QEI位置计数寄存器
0x4000 1C0C	QPOSMAX	QEI最大计数值寄存器
0x4000 1C10	QTMLAT	QEI计数值锁存寄存器
0x4000 1C14	QCNT	QEI计数寄存器
0x4000 1C18	QCNTMIN	QEI计数器在捕捉时最小计数值
0x4000 1C1C	QTPR	QEI计数器周期寄存器
0x4000 1C20	QTPSQ	QEI计数器在捕捉时最小计数值
0x4000 1C24	QEIINT	QEI中断和状态寄存器
0x4000 1C28	QTINTF	QEI中断标志和清除寄存器

22.4.1 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：保留
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

22.4.2 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 PCLK1 001：1/2 PCLK1 010：1/4 PCLK1 011：1/16 PCLK1 100：1/32 PCLK1 101：1/64 PCLK1

		110：1/128 PCLK1 111：1/256 PCLK1
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 PCLK1 001：1/2 PCLK1 010：1/4 PCLK1 011：1/16 PCLK1 100：1/32 PCLK1 101：1/64 PCLK1 110：1/128 PCLK1 111：1/256 PCLK1

22.4.3 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**

22.4.4 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

22.4.5 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**

22.4.6 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

22.4.7 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。

22.4.8 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**

22.4.9 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]**	计数器时钟预分频值

22.4.10 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						QTDE	QTCAPDE	QEIDE	QCEDE	QPEDE	QTIE	QTCAPIE	QEIIE	QCEIE	QPEIE
-						*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 ~ 10	保留	-
9	QTDE	**Qtimer溢出触发DMA使能位：** 0：禁止溢出事件发生时触发DMA 1：允许溢出事件发生时触发DMA
8	QTCAPDE	**QTimer捕捉事件触发DMA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DMA 1：允许捕捉事件发生时触发DMA
7	QEIDE	**QEI功能触发DMA使能位：** 0：禁止QEI触发DMA中断 1：使能QEI触发DMA中断
6	QCEDE	**INDEX信号复位位置计数器模式下，错误检测触发DMA使能位：** 0：禁止错误检测触发DMA 1：使能错误检测触发DMA
5	QPEDE	**编码器输入相位错误触发DMA使能位，QEA/QEB信号同相或反相时视为相位错误：** 0：禁止相位错误触发DMA 1：使能相位错误触发DMA
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

22.4.11 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

23. 片上模拟功能模块（Analog Module On Chip）

23.1 简介

SH32F284片上集成了3个高速运算放大器（OP1～OP3）、3个比较器（CP1～CP3）和一个温度传感器（TPS）。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。

23.2 主要特性

- 集成3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
- 比较器为多输入比较器
- 比较器输出结果滤波后可以用作MCM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
- 内置温度传感器，可软件校准
- 温度传感器可作为ADC输入通道，信号可通过内部独立放大（固定2倍增益）

23.3 运算放大器

23.3.1 运放典型应用图

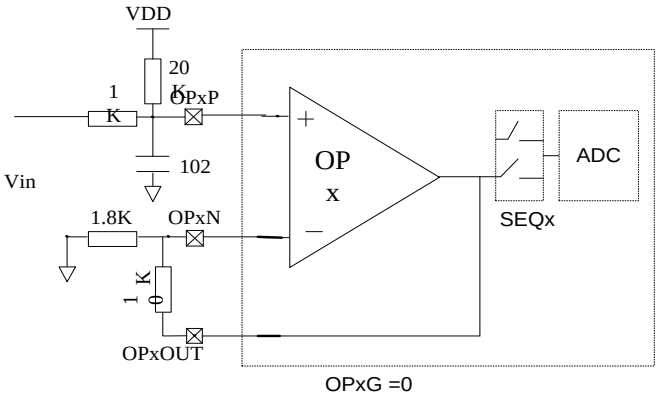


图23-1　　运放典型应用图

注意：运放使用时，要求相应的I/O口先切换到AF14（analog），再打开OPxEN，即先配置好I/O通道，再打开运放。

23.4 比较器

23.4.1 比较器用法

SH32F284内置3个比较器，比较器1、比较器2和比较器3。3个比较器的输出都可以作为MCM模块的故障输入检测信号，**而只有比较器1可以作为3个GPT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。3个比较器都有内置可编程的施密特窗口，同时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。

注：

1．比较器的上电默认状态为关闭。

2．比较器的功能和放大器的功能可以同时实现。

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。

23.4.2 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：

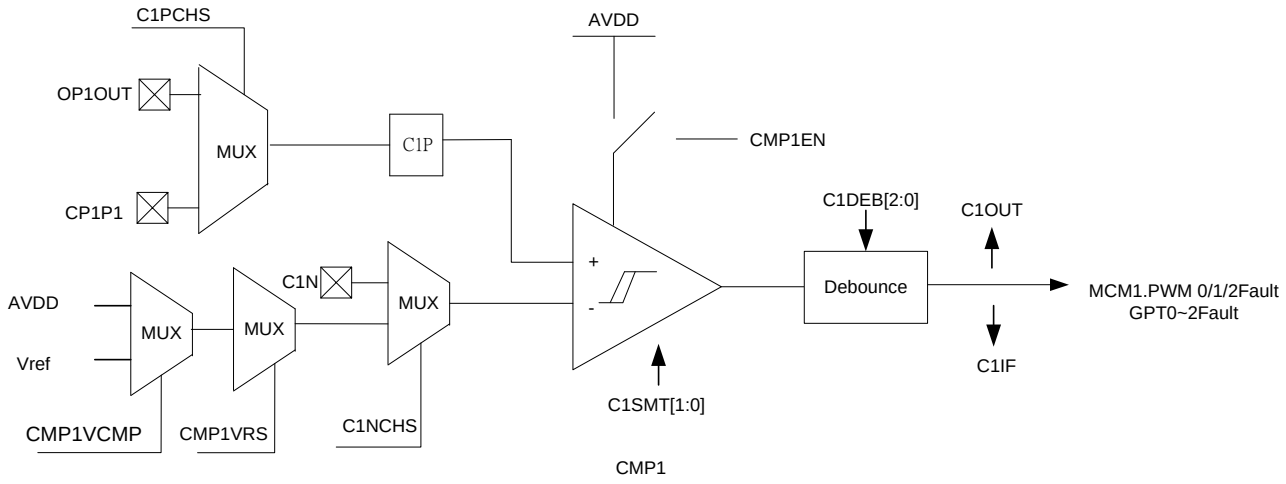


图23-2　　多通道输入模拟比较器 1

注意：仅比较器1支持GPT0～2故障保护。

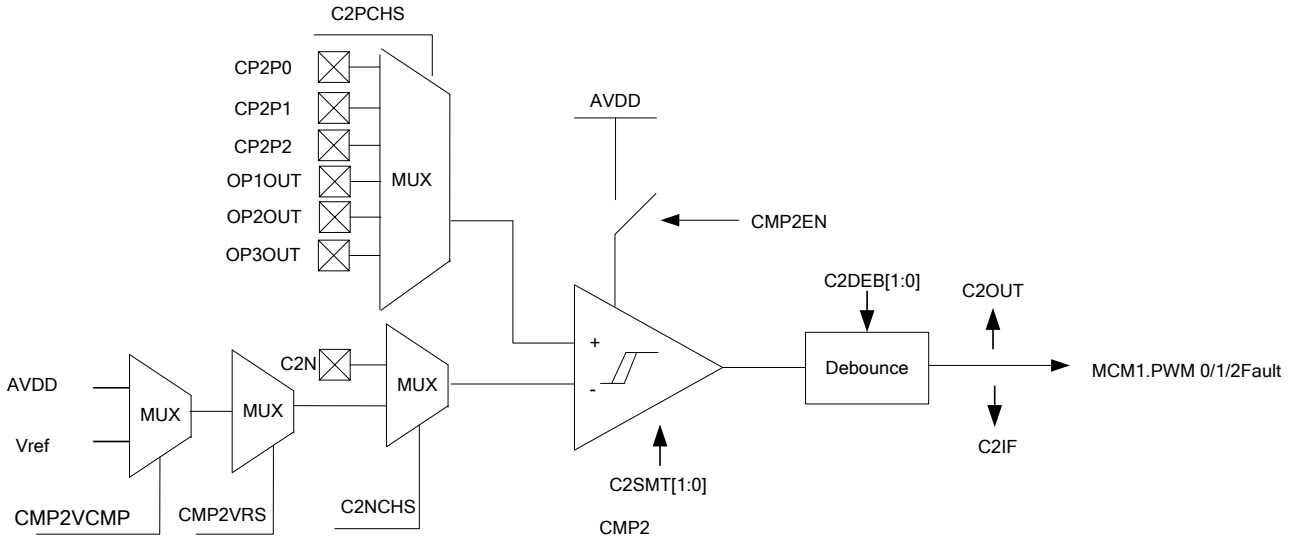


图23-3　　多通道输入模拟比较器 2

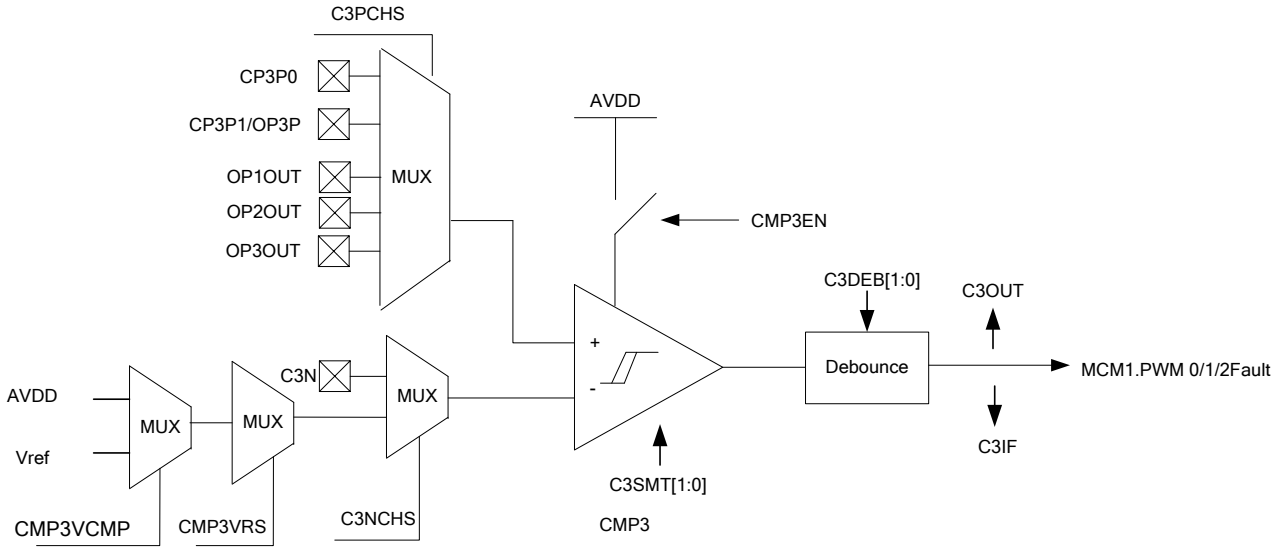


图23-4　　多通道输入模拟比较器 3

比较器1/2/3可以通过设置寄存器CxSMT[1:0](x=1-3)设置施密特比较器窗口（迟滞比较器），如下图所示：

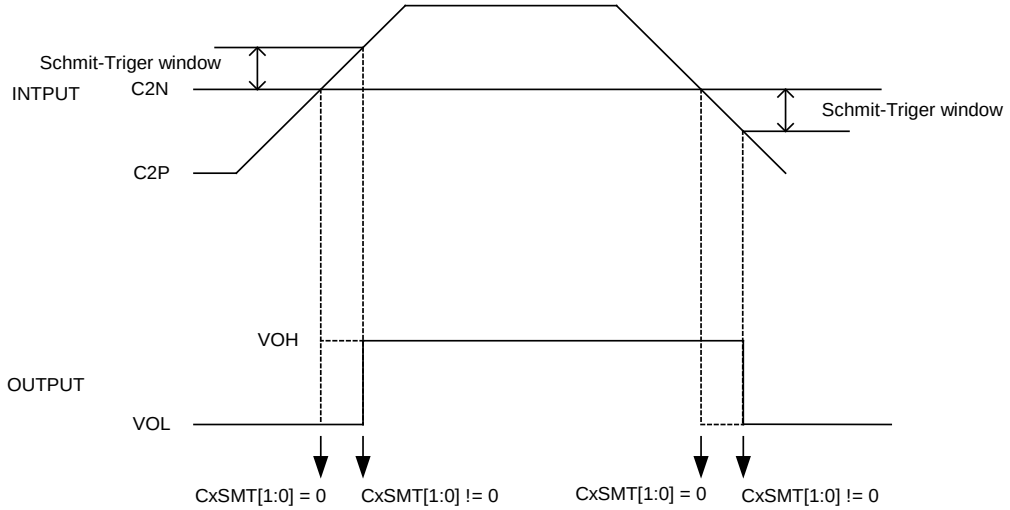


图23-5　　施密特比较器窗口

23.5 温度传感器

23.5.1 功能介绍

SH32F284包含一个温度传感器转换器，可通过TPSEN位控制，温度传感器由独立的通道送入ADC3进行采样，转换为数字值。

为了更准确读取温度传感器数值，建议同时开启内部运放的CHOP功能，即在每次转换时同时切换温度传感器与内部运放的CHOP的开启与关闭，在CHOP开启/关闭状态下分别得到温度传感器AD转换数值，取平均后即可得到更准确的温度转换结果。

温度 T = ((V25 – VTPS) / Tslope）+ 25

V25 为25度时温度传感器输出电压

VTPS 为温度传感器输出电压

Tslope 为温度传感器斜率

23.5.2 温度传感器校准

SH32F284每颗芯片出厂后都校准温度传感器，将实时AD转换温度与校准时存储AD值相减后加上校准时温度值则可以得到真实的温度。

校准时温度传感器实际温度值存于0FFF F070H – 0FFF F071H，校准时SH32F284温度传感器TPSCHOP关闭和开启时AD值分别存于0FFF F072H – 0FFF F075H。校准时实际温度保留1位小数，按Q4格式存放（即储存值=温度×2^4，16进制）。采样AD值存放格式为12bit模式下右对齐值。

程序读出示例：

```
unsigned short Temp， TempAD0， TempAD1
Temp = ReadAddr[0x0FFFF070];      //校准时温度值，16bits，Q4格式
TempAD0 = ReadAddr [0x0FFFF072]; //校准时存储的AD值 CHOP=0，16bits，12位AD采样值
TempAD1 = ReadAddr [0x0FFFF074]; //校准时存储的AD值 CHOP=1，16bits，12位AD采样值
```

温度传感器输出电压随温度线性变化，但不同芯片的温度特性曲线会有不同，尽管提供了上述校准方法，但实际温度仍然会受不同芯片的不同温度特性差异，以及芯片本身功耗差异造成温升差异等因素影响，因此温度传感器更适合检测温度的变化，而不是测量绝对温度。如需要测量精确的绝对温度，应该使用专用的外置温度传感器。

23.6 寄存器

AMOC 模块寄存器列表 (基地址:0x0x4000 5800)

地址	寄存器名	说明
0x4000 5800	CMP1CON	比较器1控制寄存器
0x4000 5804	CMP2CON	比较器2控制寄存器
0x4000 5808	CMP3CON	比较器3控制寄存器
0x4000 580C	CMPINTF	比较器中断标志和清除寄存器
0x4000 5810	OPCON	放大器控制寄存器1
0x4000 5814	AVREFCON	模拟模块内建Vref控制寄存器
0x4000 5818	TPSCON	温度传感器控制寄存器

23.6.1 比较器 1 控制寄存器 (AMOC_CMP1CON)

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												CMP1VCMP	CMP1VRS[2:0]		
-												*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
C1DE	C1IES[1:0]		保留	C1OUTEN	C1OUT	C1SMT[1:0]		CMP1EN	C1NCHS	保留	C1PCHS[1:0]		C1DEB[2:0]		
RW	*RW*		-	*RW*	*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 ~ 20	保留	-
19	**CMP1VCMP**	**比较器1反相端参考电压源Vcmp选择位：** 0：AVDD 1：内建Vref
18 ~ 16	**CMP1VRS[2:0]**	**比较器1反相端参考电压选择位** 000：Vcmp 001：1×Vcmp / 8 010：2×Vcmp / 8 011：3×Vcmp / 8 100：4×Vcmp / 8 101：5×Vcmp / 8 110：6×Vcmp / 8 111：7×Vcmp / 8
15	**C1DE**	**比较器1触发DMA使能位：** 0：当有中断标志位置起时，不触发DMA 1：当有中断标志位置起时，触发DMA 注：中断标志位的置起方式由C1IES[1:0]决定
14 ~ 13	**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2输出由高变低时和由低变高时都触发中断标志

12	保留	-
11	**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
10	**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
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	**CMP1EN**	**比较器1使能控制位** 0：比较器1关闭 1：比较器1开启
6	**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CMP1VRS作为比较器1反相输入端 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
5	保留	-
4 ~ 3	**C1PCHS[1:0]**	**比较器1同相输入端选择位** 00：选择OP1OUT作为比较器1同相输入端（OP1OUT需配置为AF14） 01：选择C1P1作为比较器1的同相输入端 10：保留 11：保留
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：滤波说明：用内部时钟采样输入信号，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。

23.6.2 比较器 2 控制寄存器 (AMOC_CMP2CON)

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												CMP2VCMP	CMP2VRS[2:0]		
-												*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
C2DE	C2IES[1:0]		保留	C2OUTEN	C2OUT	C2SMT[1:0]		CMP2EN	C2NCHS	C2PCHS[2:0]			C2DEB[2:0]		
RW	*RW*		*-*	*RW*	*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 ~ 20	保留	-
19	**CMP2VCMP**	**比较器2反相端参考电压源Vcmp选择位：** 0：AVDD 1：内建Vref
18 ~ 16	**CMP2VRS[2:0]**	**比较器2反相端参考电压选择位** 000：Vcmp 001：1×Vcmp / 8 010：2×Vcmp / 8 011：3×Vcmp / 8 100：4×Vcmp / 8 101：5×Vcmp / 8 110：6×Vcmp / 8 111：7×Vcmp / 8
15	**C2DE**	**比较器2触发DMA使能位：** 0：当有中断标志位置起时，不触发DMA 1：当有中断标志位置起时，触发DMA 注：中断标志位的置起方式由C2IES[1:0]决定
14 ~ 13	**C2IES[1:0]**	**比较器2中断模式选择位** 00：不触发中断标志 01：下降沿触发，比较器2输出由高变低时触发中断标志 10：上升沿触发，比较器2输出由低变高时触发中断标志 11：双沿触发，比较器2输出由高变低时和由低变高时都触发中断标志
12	保留	-
11	**C2OUTEN**	**比较器2I/O输出口使能** 0：不通过比较器2的CP2OUT引脚输出，CP2OUT为普通IO 1：通过比较器2的CP2OUT引脚输出 注意：I/O输出是带滤波功能的，滤波参数为C2DEB[2:0]的设定值
10	**C2OUT**	**比较器2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2输出低 1：比较器2输出高
9 ~ 8	**C2SMT[1:0]**	**比较器2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	**CMP2EN**	**比较器2使能控制位** 0：比较器2关闭 1：比较器2开启
6	**C2NCHS**	**比较器2反相输入端选择位** 0：选择C2N作为比较器2反相输入端

		1：选择内部参考CMP2VRS作为比较器2反相输入端，CP2N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
5 ~ 3	**C2PCHS[2:0]**	**比较器2同相输入端选择位** 000：选择C2P0作为比较器2同相输入端 001：选择C2P1作为比较器2同相输入端 010：选择C2P2作为比较器2同相输入端 011：选择OP1OUT作为比较器2的同相输入端 100：选择OP2OUT作为比较器2的同相输入端 101：选择OP3OUT作为比较器2的同相输入端 11x：保留
2 ~ 0	**C2DEB[2:0]**	**比较器2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：滤波说明：用内部时钟采样输入信号，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。

23.6.3 比较器 3 控制寄存器 (AMOC_CMP3CON)

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												CMP3VCMP	CMP3VRS[2:0]		
-												*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
C3DE	C3IES[1:0]		保留	C3OUTEN	C3OUT	C3SMT[1:0]		CMP3EN	C3NCHS	C3PCHS[2:0]			C3DEB[2:0]		
RW	*RW*		-	*RW*	*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 ~ 20	保留	-
19	**CMP3VCMP**	**比较器3反相端参考电压源Vcmp选择位：** 0：AVDD 1：内建Vref
18 ~ 16	**CMP3VRS[2:0]**	**比较器3反相端参考电压选择位** 000：Vcmp 001：1×Vcmp / 8 010：2×Vcmp / 8

		011：3×Vcmp / 8 100：4×Vcmp / 8 101：5×Vcmp / 8 110：6×Vcmp / 8 111：7×Vcmp / 8
15	**C3DE**	**比较器3触发DMA使能位：** 0：当有中断标志位置起时，不触发DMA 1：当有中断标志位置起时，触发DMA 注：中断标志位的置起方式由C3IES[1:0]决定
14 ~ 13	**C3IES[1:0]**	**比较器3中断模式选择位** 00：不触发中断标志 01：下降沿触发，比较器3输出由高变低时触发中断标志 10：上升沿触发，比较器3输出由低变高时触发中断标志 11：双沿触发，比较器3输出由高变低时和由低变高时都触发中断标志
12	**保留**	**-**
11	**C3OUTEN**	**比较器3I/O输出口使能** 0：不通过比较器3的CP3OUT引脚输出，CP3OUT为普通IO 1：通过比较器3的CP3OUT引脚输出 注意：I/O输出是带滤波功能的，滤波参数为C3DEB[2:0]的设定值
10	**C3OUT**	**比较器3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3输出低 1：比较器3输出高
9 ~ 8	**C3SMT[1:0]**	**比较器3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	**CMP3EN**	**比较器3使能控制位** 0：比较器3关闭 1：比较器3开启
6	**C3NCHS**	**比较器3反相输入端选择位** 0：选择C3N作为比较器3反相输入端 1：选择内部参考CMP3VRS作为比较器1反相输入端 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
5 ~ 3	**C3PCHS[2:0]**	**比较器3同相输入端选择位** 000：选择C3P0作为比较器3同相输入端 001：选择C3P1作为比较器3同相输入端 010：选择OP1OUT作为比较器3的同相输入端 011：选择OP2OUT作为比较器3的同相输入端 100：选择OP3OUT作为比较器3的同相输入端 其他：保留
2 ~ 0	**C3DEB[2:0]**	**比较器3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：滤波说明：用内部时钟采样输入信号，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。

23.6.4 比较器中断标志和清除寄存器 (AMOC_CMPINTF)

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													C3IFC	C2IFC	C1IFC
-													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													C3IF	C2IF	C1IF
-													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	C3IFC	比较器3输出上升沿中断标志位清除位 0：无效 1：清除
17	C2IFC	比较器2输出上升沿中断标志位清除位 0：无效 1：清除
16	C1IFC	比较器1输出上升沿中断标志位清除位 0：无效 1：清除
15 ~ 3	保留	-
2	C3IF	比较器3输出上升沿中断标志位（经过滤波器以后） 0：比较器3输出没有产生中断请求 1：比较器3输出产生中断请求
1	C2IF	比较器2输出上升沿中断标志位（经过滤波器以后） 0：比较器2输出没有产生中断 1：比较器2输出产生中断请求
0	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输出发生中断请求

23.6.5 放大器控制寄存器 1 (AMOC_OPCON)

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													OP3EN	OP2EN	OP1EN
-													*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	**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 注意：运算放大器开启时，相应的输入输出引脚I/O需提前配置为模拟引脚！
1	**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 注意：运算放大器开启时，相应的输入输出引脚I/O需提前配置为模拟引脚！
0	**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 注意：运算放大器开启时，相应的输入输出引脚I/O需提前配置为模拟引脚！

23.6.6 模拟模块内建 Vref 控制寄存器 (AMOC_AVREFCON)

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														VREFSEL	VREFEN
-														*RW*	*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	**VREFSEL**	**内建电压输出选择位** 0：选择2.5V为内建VREF 1：选择3.3V为内建VREF
0	**VREFEN**	**内建电压开启位**

		0：停止内建VREF 1：开启内建VREF 注：基准源开启需要建立时间，最长不超过100us，因此当需要用到VREF时，需要提前100us将此标志位置1

23.6.7 温度传感器控制寄存器 (AMOC_TPSCON)

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
保留														TPSCHOP	TPSEN
-														*RW*	*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	**TPSCHOP**	**温度传感器CHOP开关** 0：温度传感器CHOP功能关闭 1：温度传感器CHOP功能开启
0	**TPSEN**	**温度传感器开关** 0：关闭温度传感器 1：开启温度传感器 注：温度传感器从关闭到开启，有一个建立过程，从TPSEN置为1开始，至少等待100us再开始AD转换读取转换结果

24. 串行通信接口

24.1 通用异步收发器（UART）

24.1.1 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。SH32F284包含3个UART（UART1～UART3）。

24.1.2 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
- 支持 DMA 通信

24.1.3 功能描述

24.1.3.1 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 引脚上移入 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24-1 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	PCLK1/（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	PCLK1/（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 支持与外部设备的同步通信。在 RXD 引脚上收发串行数据，TXD 引脚发送移位时钟。SH32F284 提供 TXD 引脚上的移位时钟，因此这种方式是串行通信的半双工方式。在这个方式中，每帧收发 8 位，低位先接收或发送。

通过置 SM2 位@UARTx_CR 为 0 或 1，波特率固定为 PCLK1 时钟的 1/48 或 1/12。当 SM2 位等于 0 时，串行端口以 PCLK1 时钟的 1/48 运行。当 SM2 位等于 1 时，串行端口以 PCLK1 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。

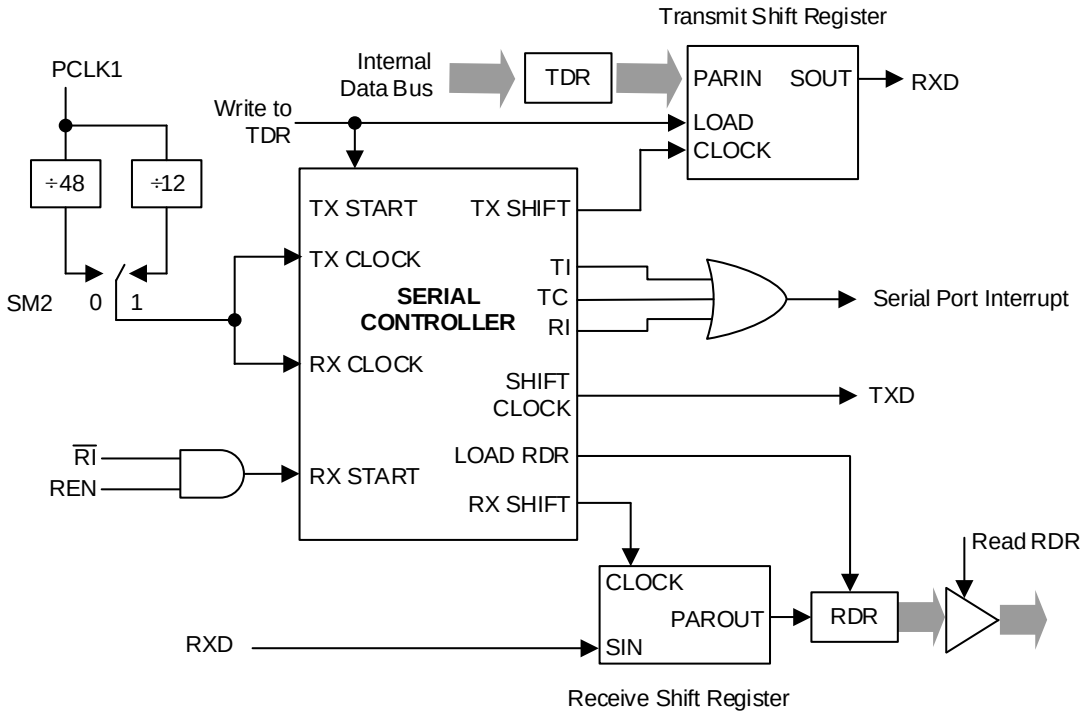
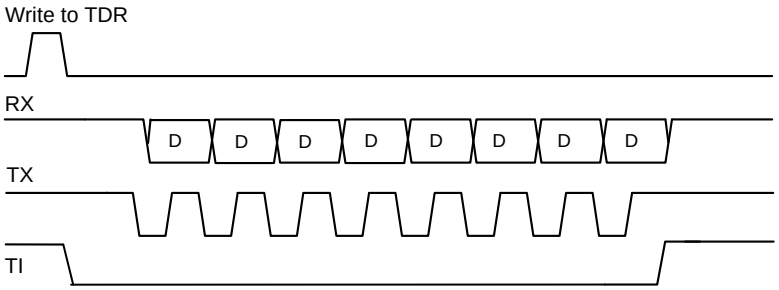


图24-1　　方式 0 功能块框图

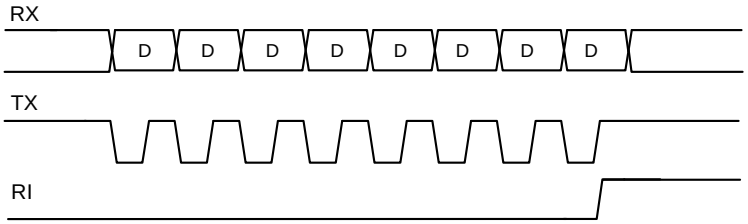
在使能了 TEN@UARTx_CR 后,UARTx_TDR 寄存器为空，写操作可以被执行。写入数据下一个 PCLK1 时钟 TX 控制块开始发送。数据转换发生在移位时钟的下降沿，移位寄存器的内容逐次从左往右移位，空位置 0。当移位寄存器中的所有 8 位都发送后，TX 控制模块停止发送操作，然后在下一个 PCLK1 时钟的上升沿将数据从 UARTx_TDR 寄存器搬移置 SHIFT 寄存器，此时，UARTx_TDR 又为空，TI @UARTx_FR 硬件置位。TI 为发送中断标志，实时反映发送寄存器状态，置位表示发送寄存器为空。由于 UART 发送有一级缓冲区，所以首次发送可以连续写入两 byte 发送数据。



Send Timing of Mode 0

图24-2　　方式 0 发送时序图

REN@UARTx_CR 置 1 和 RI@UARTx_FR 写 1 清零初始化接收。下一个 PCLK1 时钟启动接收，在移位时钟的上升沿锁存数据，接收转换寄存器的内容逐次向左移位。当所有 8 位数据都移到移位寄存器中后，RX 控制停止接收，在下一个 PCLK1 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 1 清零 RI）。RI 为接收中断标志，实时反映接收寄存器状态，置位表示接收寄存器为满。由于 UART 接收有一级缓冲区，所以使能接收后首次接收会连续发送 16 个时钟，此时需要及时将 RDR 内的数据取走（即要在第二个数据进入 RDR 之前取走第一个数据），否则接收流程会挂起并置起 RXOV 标志，需要重新使能 REN(先写 0 关闭再写 1 开启)才能恢复后续接收。



Receive Timing of Mode 0

图24-3　　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_CR 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：

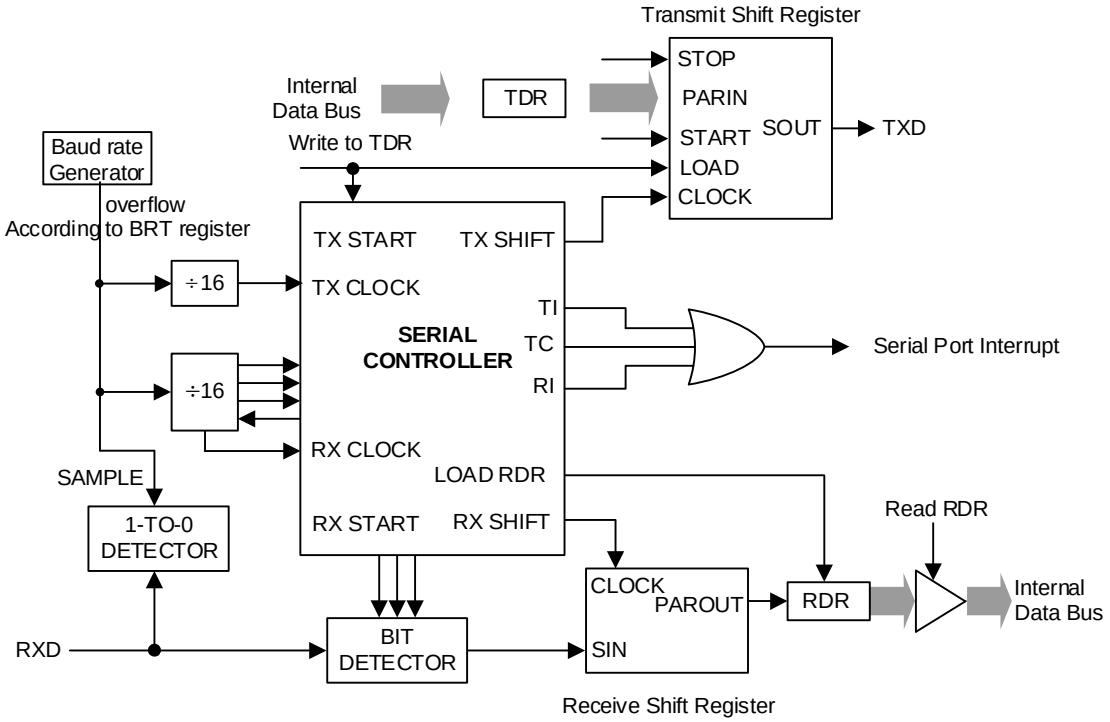
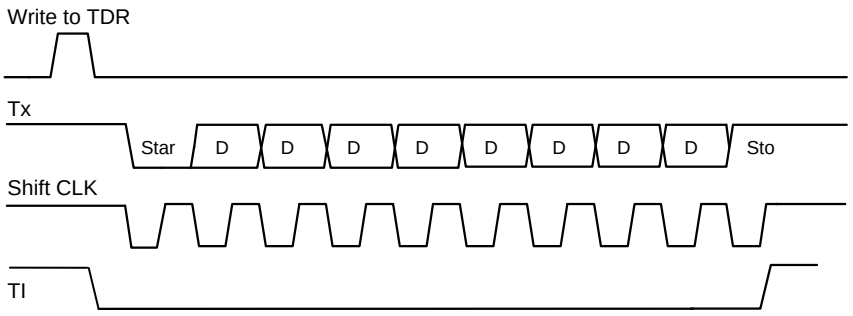


图24-4　　方式 1 功能块框图

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 PCLK1 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24-5　　方式 1 发送时序图(STOP=1BIT)

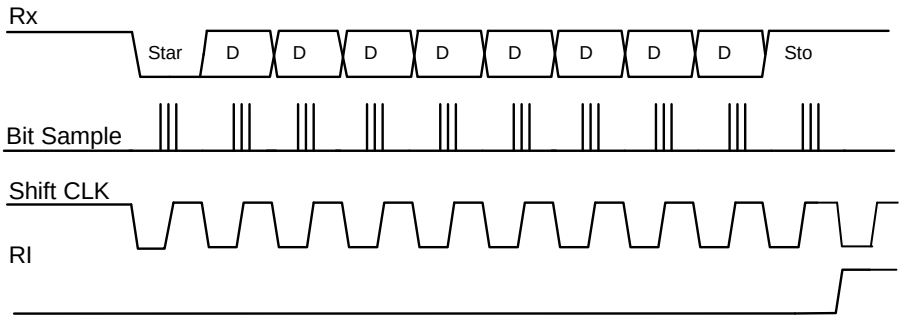
只有 REN 置位时才允许接收。当 RXD 引脚检测到下降沿时串行口开始接收串行数据。为此，UART 对 RXD 不断采样，采样速率为波特率的 16 倍。当检测下降沿时，16 分频计数器立即复位，这有助于 16 分频计数器与 RXD 引脚上的串行数据位同步。16 分频计数器把每一位的时间分为 16 个状态，在第 7、8、9 状态时，位检测器对 RXD 端的电平进行采样。为抑制噪声，在这 3 个状态采样中至少有 2 次采样值一致数据才被接收。如果所接收的第一位不是 0，说明这位不是一帧数据的起始位，该位被忽略，接收电路被复位，等待 RXD 引脚上另一个下降沿的到来。若起始位有效，则移入移位寄存器，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 UARTx_RDR 和 RB8 中，RI 置位，但必须满足下列条件：

（1） RI = 0

（2）SM2 = 0 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。用户应及时将 UARTx_RDR 内数据取走（或用软件写 1 清零 RI），如果 UARTx_RDR 内数据未取走，且移位寄存器内数据接收满，RXOV 标志将会被置起，后续数据将会丢失[注]。

注：发生 RXOV 后数据有丢失，需要重新接收。



Receive Timing of Mode 1

图24-6　　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_CR）可以写 0 或 1，该位的设置需要在写入相应 UARTx_TDR 之前完成。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。若使用软件读取 RB8 位需要在读取 UARTx_RDR 寄存器之前读取。通过 SMOD@UARTx_CR 位选择波特率为 PCLK1 的 1/256 或 1/128。功能块框图如下所示：

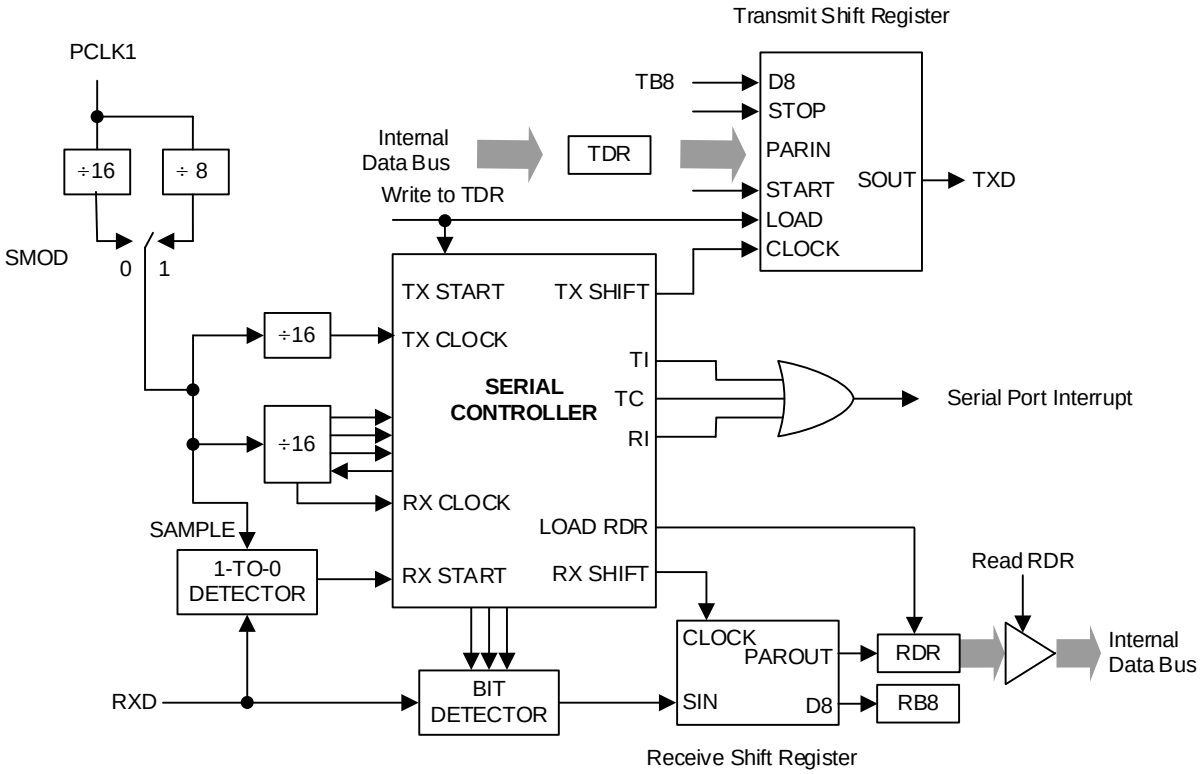
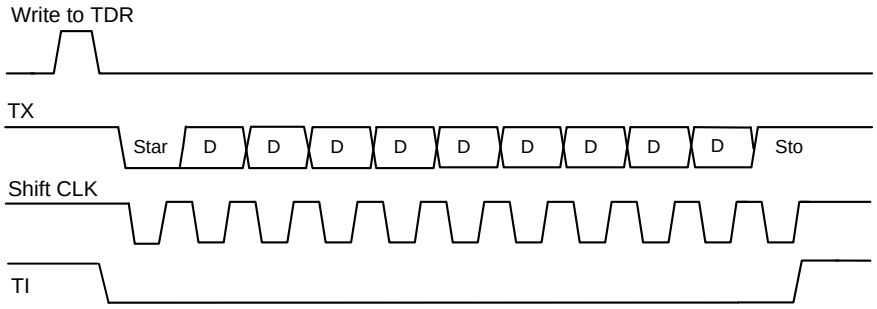


图24-7　　方式 2 功能块框图

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 PCLK1 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24-8　　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：

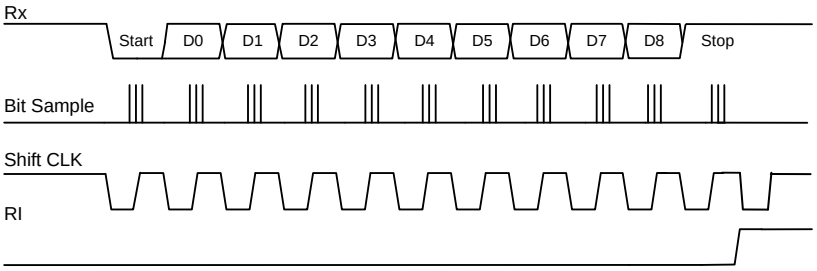
（1）RI = 0

（2）SM2 = 0（用户自定义第 9 位模式）

或者，SM2 = 1，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。在停止位结束后，接收器重新寻找 RXD 引脚上的下降沿开始新的一字节数据的接收。用户应及时将 UARTx_RDR 内数据取走（或用软件写 1 清零 RI），如果 UARTx_RDR 内数据未取走，且移位寄存器内数据接收满，RXOV 标志将会被置起，后续数据将会丢失。



Receive Timing of Mode 2

图24-9　　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。

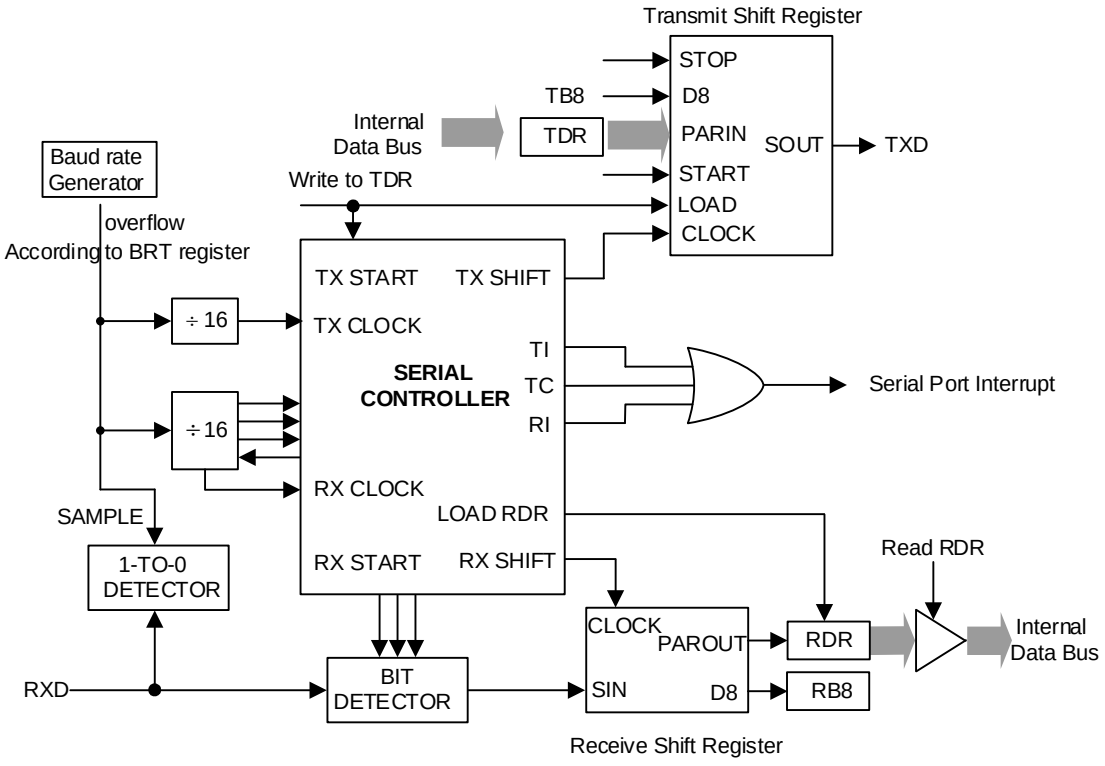


图24-10　　方式 3 功能块框图

关于 TI,TC 标志的说明

TI@UARTx_FR为发送中断标志位。若UARTx_TDR为空，TI将被置位。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里，第二个数据才被写入到UARTx_TDR里。所以在第二个数据写入到UARTx_TDR内后，TI标志位才会清零。（如图24-11所示）

TC@UARTx_FR为连续帧发送结束标志位。该位通过以下操作序列来实现清零：step1，读UARTx_FR寄存器; step2，写UARTx_TDR寄存器。对TCC@UARTx_FR写1也可以清零TC位。当连续发送的一帧数据的最后一字节数据发送结束后（停止位发送结束后），并且检测到TI@UARTx_FR为高的情形下，TC标志位置起。（如图24-11所示）

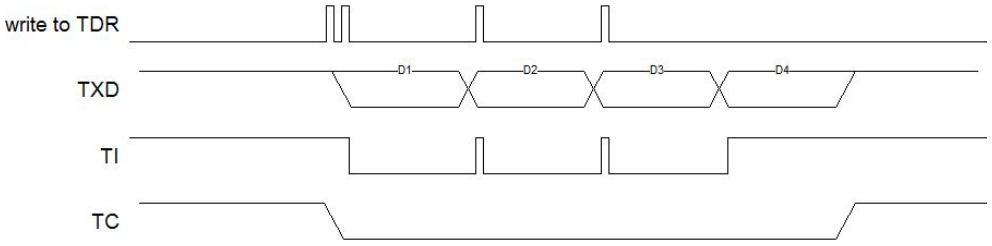


图24-11　　TC 工作时序图

24.1.3.2　　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。

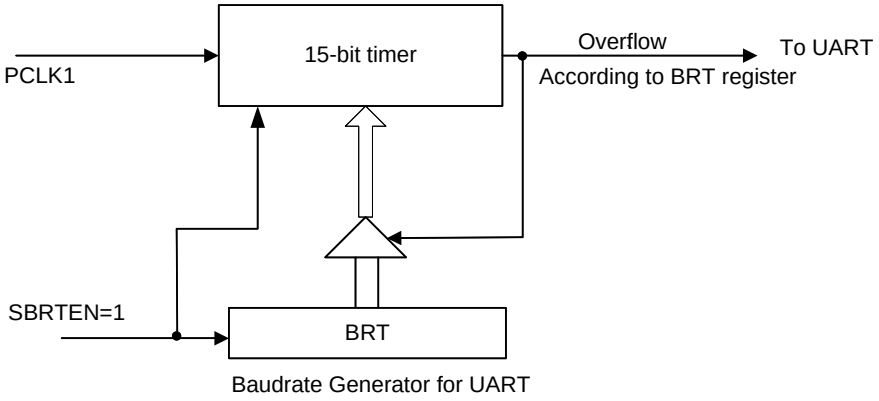


图24-12　　波特率发生器功能块框图

由图得到，波特率发生器的溢出率为

$$\text{SBRT overflow rate} = \frac{\text{PCLK1}}{\text{BRT}}$$

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 APB1 时钟的 1/48 或 1/12，由 SM2 位决定。当 SM2 为 0 时，串行端口在 PCLK1 时钟的 1/48 下运行。当 SM2 为 1 时，串行端口在 PCLK1 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 PCLK1 时钟，公式如下：

$$\text{BaudRate} = \frac{\text{PCLK1}}{16\times(\text{SBRT}+1)+\text{BFINE}}$$

例如：PCLK1 = 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 PCLK1 时钟的 1/256 或 1/128，由 SMOD 位@UARTx_CR 中决定。当 SMOD 位为 0 时，UART 以 PCLK1 时钟的 1/256 运行。当 SMOD 位为 1 时，UART 以 PCLK1 时钟的 1/128 运行。

$$\text{BaudRate} = 2^{\text{SMOD}}\times\frac{\text{PCLK1}}{256}$$

24.1.3.3 多机通讯

自动（硬件）地址识别

在方式 2 和方式 3 中，SM2 置位（进入多机通讯模式），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匹配 UART 的从机地址，UART 将开始接收后续数据字节。否则，继续等待地址匹配。

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24-2 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。

24.1.3.4 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

如果在 UARTx_RDR 中的数据未被读取之前，又有新的数据存入 UARTx_RDR，那么接收溢出位 RXOV@UARTx_FR 置位。如果发生了接收溢出，UARTx_RDR 中未被取走的数据将丢失。该位的清零通过对 RXOVC@UARTx_FR 写 1 来实现。

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。

24.1.3.5 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，若奇偶校验允许位 PCE@UARTx_CR 为 1，第九位 TB9 将用为奇偶校验位（优先级高于多机通讯模式和用户自定义第九位模式），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。

24.1.3.6 波特率检测

波特率检测模式由 GTIOA@GPTx_GTIOR 使能。该功能使 RXD 引脚与 CAPTURE 模块（本例中由 GPT 实现捕捉功能）连接，可供 RXD 用于波特率检测，注意在进入波特率检测模式时，RXD 信号将被内部直连到 CAPTURE，直到退出波特率检测模式。

24.1.3.7 LIN（局域互联网）总线模式

SH32F284 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。

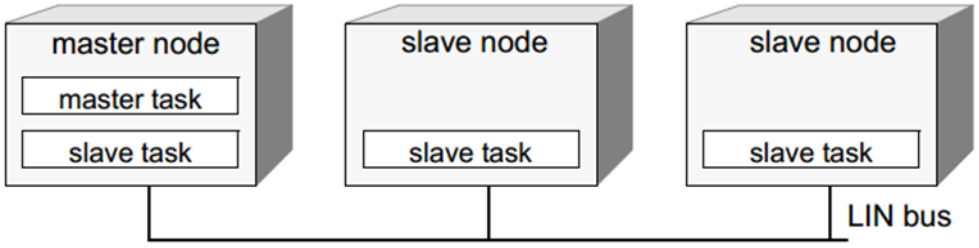


图24-13 LIN 总线模式

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。

24.1.3.8 利用 DMA 通信

用户可选择通过使用 DMA 操作 UART 的发送和或接收。DMA 模式由 UARTx_CR 寄存器中的 DMAT 和 DMAR 模式选择位分别决定发送与接受是否采用 DMA 总线进行通讯。选择 DMA 模式后，建议将 UART 的相应中断使能位关闭，因为 DMA 模式下 DMA 请求无需通过 CPU 参与便可以自行处理数据流传输。

UART 发送

在 DMA 模式中，当发送数据缓存 UARTx_TDR 数据发送完毕为空的情况下，串口发送的 DMA 请求就会生效，以请求 DMA 将后续数据传输到发送数据缓存 UARTx_TDR 内数据。串口发送 DMA 请求由 DMA 控制器清除。

UART 接收

在 DMA 模式中，当接收数据缓存 UARTx_RDR 数据更新完毕为满的情况下，串口接收的 DMA 请求就会生效，以请求 DMA 将接收数据缓存 UARTx_RDR 内数据取走。串口接收 DMA 请求由 DMA 控制器清除。

24.1.3.9 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。

采用 DMA 通道时，UART 的读写请求信号直接传送至 DMA 控制器，中断信号由 DMA 控制器负责清 0；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。

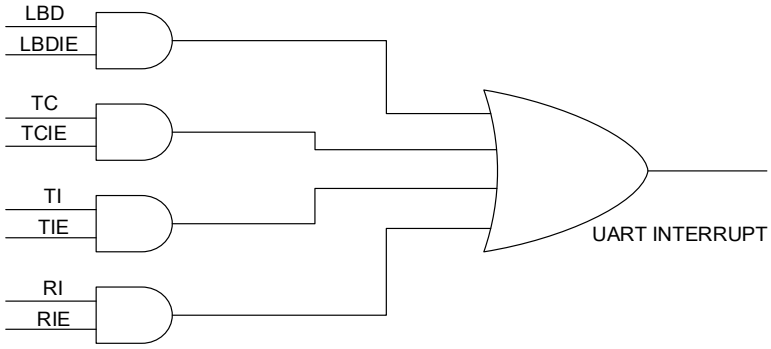


图24-14　　UART 中断控制图

24.1.4　UART 寄存器

UART 模块寄存器列表 (基地址:0x0x4000 2000)

地址	寄存器名	说明
0x4000 2000	FR	UART中断标志寄存器
0x4000 2008	TDR	UART发送数据寄存器
0x4000 200C	RDR	UART接收数据寄存器
0x4000 2010	ADDR	UART地址配置寄存器
0x4000 2014	BRT	UART波特率配置寄存器
0x4000 2018	CR	UART控制寄存器

24.1.4.1　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，也可以先读FR再写TDR的操作序列亦可清零
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，也可以也可以写TDR的操作序列清零
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

24.1.4.2 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
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								TDR[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	TDR[7:0]	UART的发送数据寄存器

24.1.4.3 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
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								RDR[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	RDR[7:0]	UART的接收数据寄存器

24.1.4.4 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

24.1.4.5 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

24.1.4.6 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											DMAT	DMAR	TEN	REN	LINEN
-											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]		SM2	PCE	PS	TB8	RB8	LBDL	LBDIE	TCIE	TIE	RIE	SMOD	SBRTEN	STOP
RW	RW		RW	RW	RW	RW	RO	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 ~ 21	保留	-
20	DMAT	UART发送DMA通道使能位 0：发送数据不通过DMA传输 1：发送数据通过DMA传输
19	DMAR	UART接收DMA通道使能位 0：接收数据不通过DMA传输 1：接收数据通过DMA传输
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	**SM2**	**多机处理通讯允许位** 0：在方式0下，波特率是PCLK1时钟的1/48 在方式1下，禁止停止位确认检验，任何停止位都会置位RI 在方式2和3下，任何字节都会置位RI 1：在方式0下，波特率是PCLK1时钟的1/12 在方式1下，允许停止位确认检验，只有有效的停止位(1)才能置位RI 在方式2和3下，只有地址字节（第9位=1）才能置位RI
11	**PCE**	**奇偶校验使能位** 0:禁止奇偶校验 1:使能奇偶校验
10	**PS**	**奇偶校验选择位** 0:奇校验 1:偶校验
9	**TB8**	**在UART的方式2和3下发送的第9位**
8	**RB8**	**在UART的方式2和3下接收的第9位**
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	**SMOD**	**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

24.2 串行外设接口（SPI）

24.2.1 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。SH32F284 包含 1 个 SPI（SPI1）。

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。

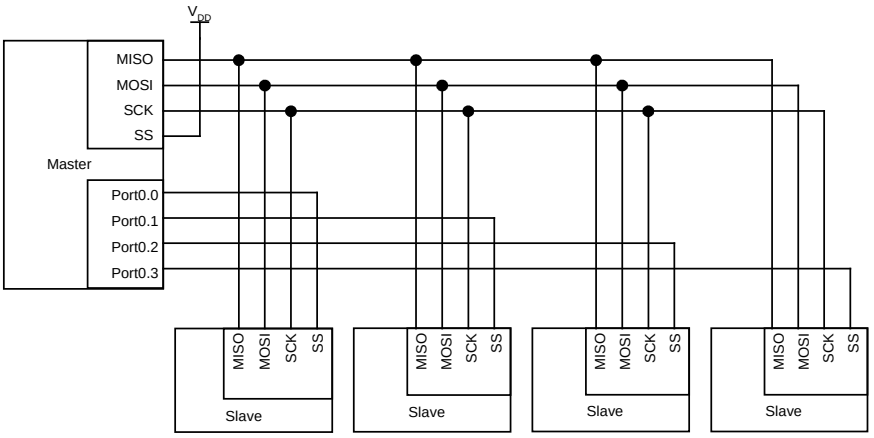


图24-15　　SPI 总线网络图

24.2.2 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 可选
- 支持 DMA 通信
- 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

24.2.3 功能描述

24.2.3.1　　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。

24.2.3.2 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。

24.2.3.3 原理描述

下图所示是 SPI 模块的详细结构。

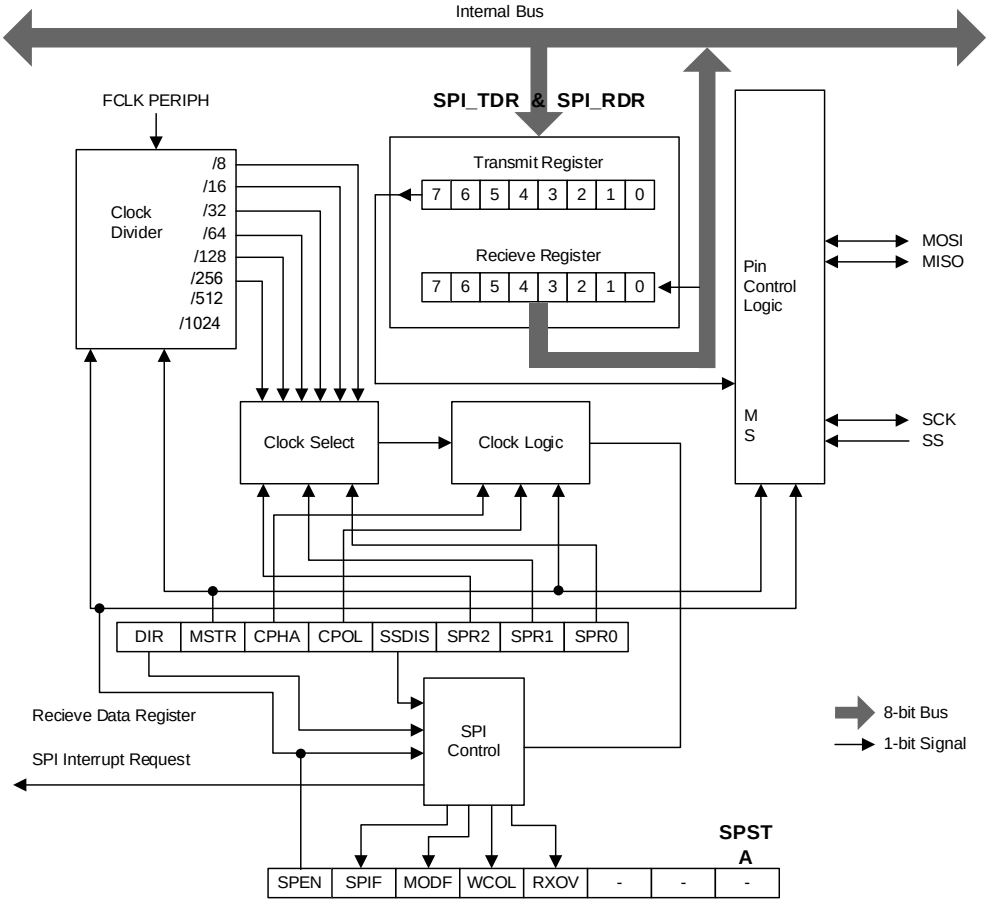


图24-16 SPI 模块框图

24.2.3.4 工作模式

SPI 可配置为主模式或从模式中的一种。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。

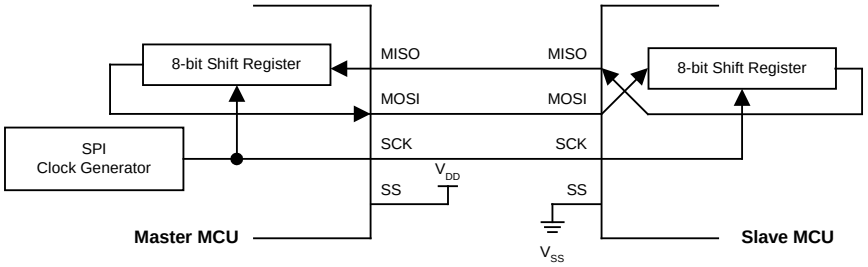


图24-17　　全双工主从互联图（CPHA=1）

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，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 内有数据，那么 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。

24.2.3.5　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。

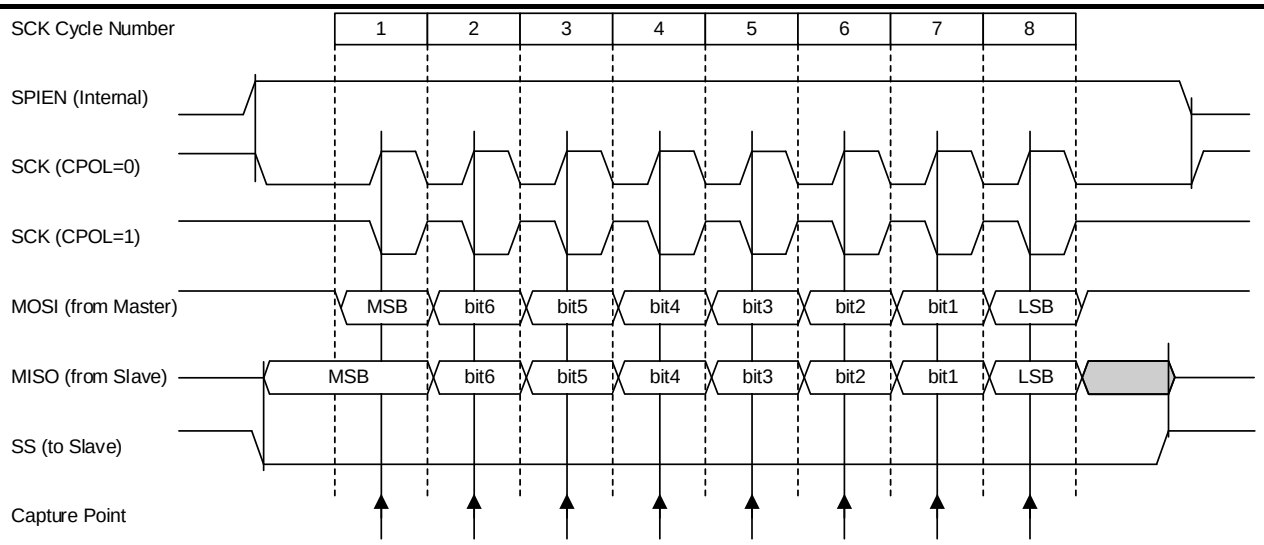


图24-18　　数据传送形式（CPHA = 0）图

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，同时，使能该功能后还可以使用DMA方式来传输数据。

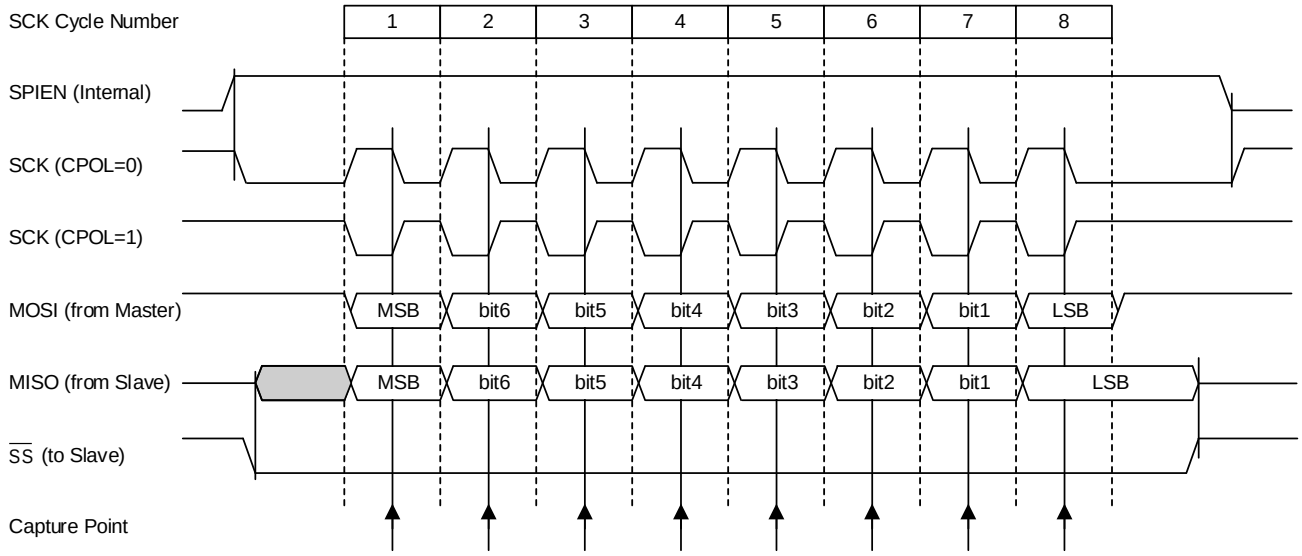


图24-19　　数据发送形式（CPHA = 1）图

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。

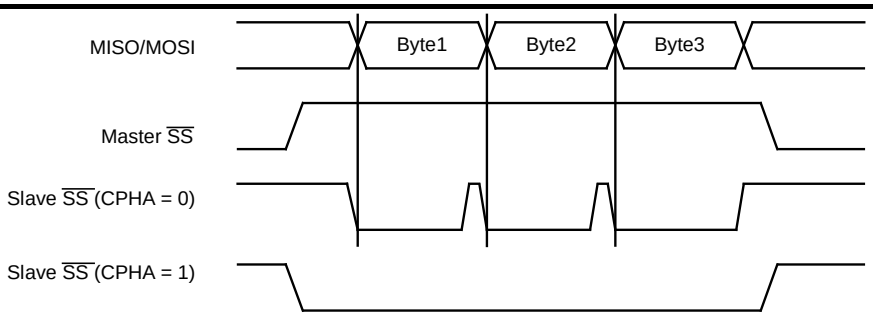


图24-20　CPHA/NSS 时序图

注意：当 SPI 用作从设备模式，且 SPIx_CR 寄存器的 CPOL 位清 0，SCK 端口必须在数据传输前打开上拉电阻。

24.2.3.6　数据字 8、16 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 位或 16 位。采用 16 位字长时，移位寄存器与缓存寄存器均被扩展为 16 位模式，对 SPIx_TDR 与 SPIx_RDR 的读写均为 16 位操作。时序图如下：

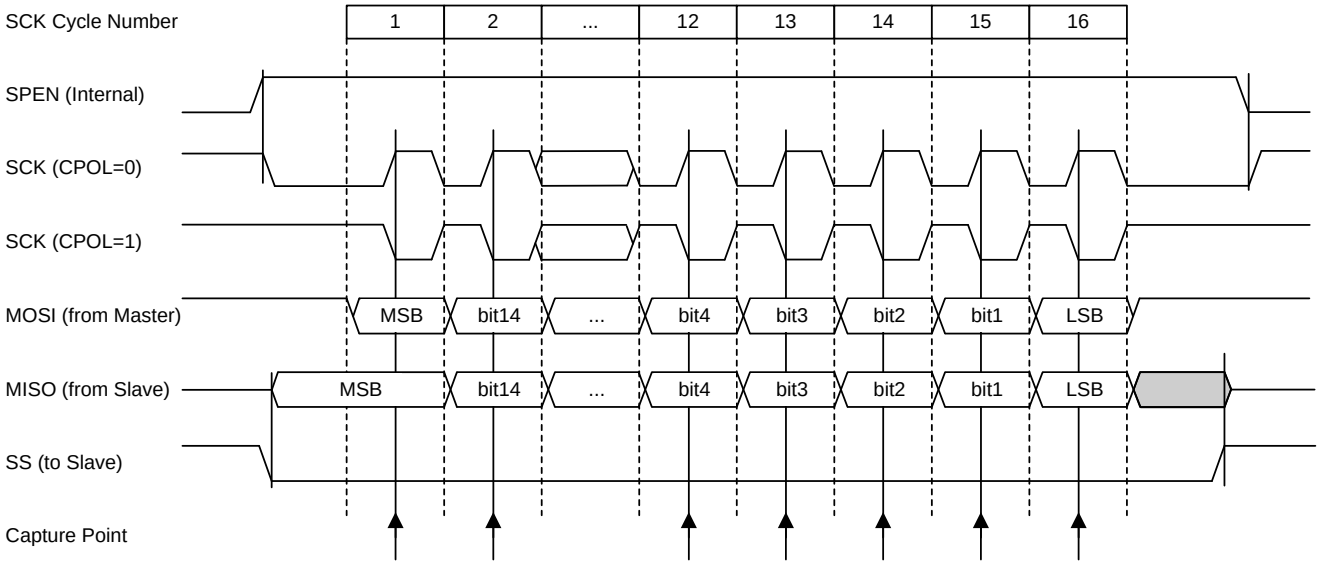


图24-21　数据传送形式（CPHA = 0）

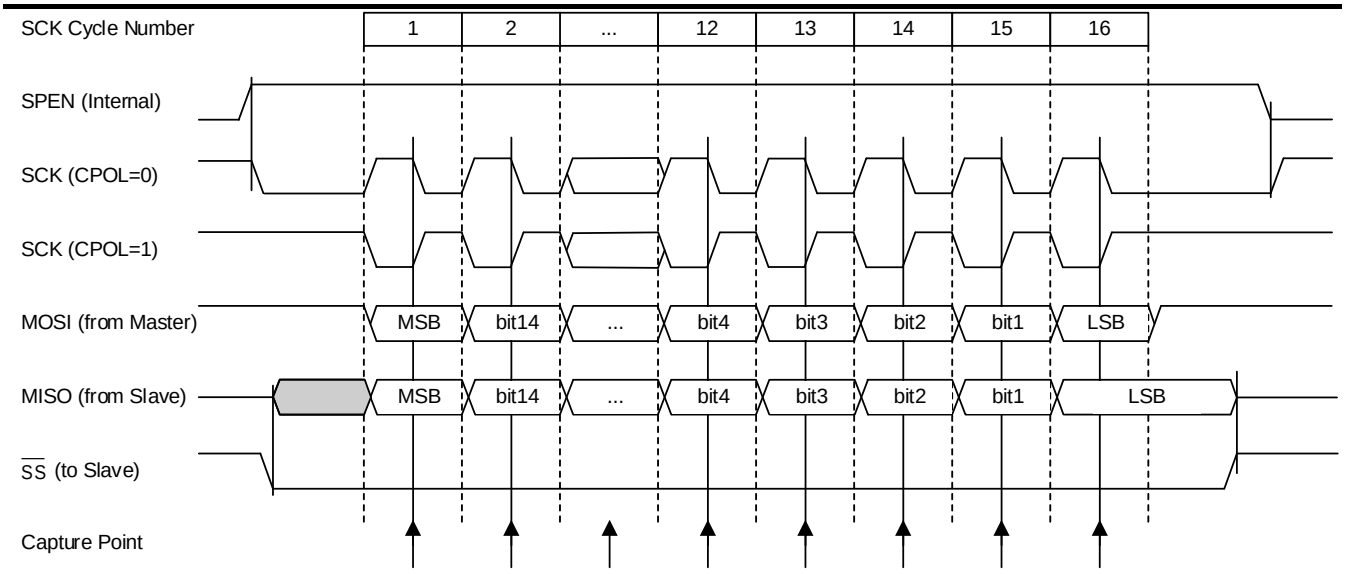


图24-22 数据发送形式（CPHA = 1）

24.2.3.7 利用DMA 通信

为了达到最大通信速度，需要及时往 SPI 发送缓冲器填数据，同样接收缓冲器中的数据也必须及时读走以防止溢出。为了方便高速率的数据传输，SPI 采用了一种采用简单的请求/应答的 DMA 机制。

当 SPIx_CR 寄存器上的对应使能位 DMAT、DMAR@SPIx_CR 被设置时，SPI 模块可以发出 DMA 传输请求。发送缓冲器和接收缓冲器有各自的 DMA 请求。选择 DMA 模式后，建议将 SPI 的相应中断使能位关闭，因为 DMA 模式下 DMA 请求无需通过 CPU 参与便可以自行处理数据流传输。

SPI 发送

发送时，在每次 SPTI 被置 1 时发出 DMA 请求，DMA 控制器则写数据至 SPIx_TDR 寄存器，SPTI 标志因此而被清 0。

SPI 接收

接收时，在每次 SPRI 被置 1 时发出 DMA 请求，DMA 控制器则从 SPIx_RDT 寄存器读出数据，SPRI 标志因此而被清 0。

当只使用 SPI 发送数据时，只需使能 SPI 的发送 DMA 通道。此时，若因为没有读取收到的数据，RXOV 被置为'1'(软件不必理会这个标志)。

24.2.3.8 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。

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。

如果设备需要做主机传输数据，需要先查看 SS 信号电平是否为高电平，若为高电平则允许重新使能主机展开通讯。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。

24.2.3.9 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；

使能 DMA 通道时，SPI 的 DMA 请求信号在完成一字节传输后（发送时为发送寄存器为空，接收时为接收寄存器为满）产生，直接传送至 DMA 控制器，请求信号由 DMA 控制器负责清 0；

模式故障标志，MODF：该位被置 1 表示$\overline{SS}$引脚上的电平与SPI多主机模式发生总线冲突。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。

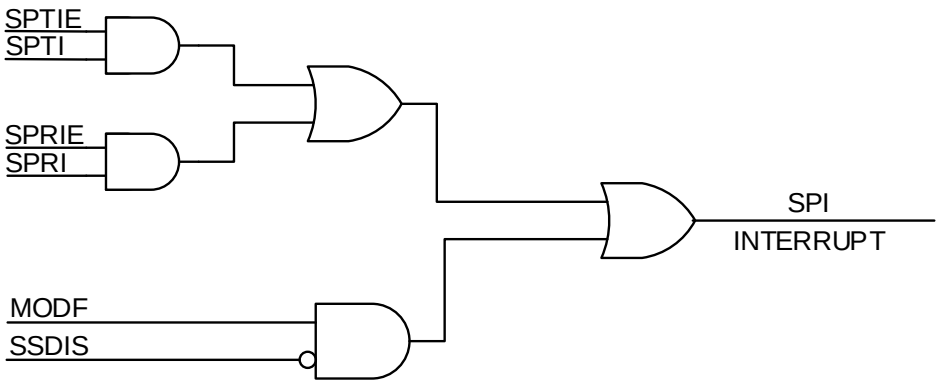


图24-23　　SPI 中断控制图

24.2.4　SPI 寄存器

SPI 模块寄存器列表 (基地址:0x0x4000 2C00)

地址	寄存器名	说明
0x4000 2C00	FR	SPI中断标志寄存器
0x4000 2C04	TDR	SPI发送数据寄存器
0x4000 2C08	RDR	SPI接收数据寄存器
0x4000 2C0C	CR	SPI控制寄存器

24.2.4.1　SPI 中断标志寄存器 (SPIx_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	保留	SPTI	SPRI
-										*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**	**SPI一字节数据传送中断标志清除位** 0：无效 1：清除
16	**SPRIC**	**SPI一字节数据传送中断标志清除位** 0：无效 1：清除
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	**保留**	**-**
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一字节数据传送中断标志位** 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

24.2.4.2 SPI 发送数据寄存器 (SPIx_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
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
TDR[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	**TDR[15:0]**	**发送缓存寄存器** 当SPDATL=1,为16位SPI发送缓存寄存器 当SPDATL=0,为8位SPI发送缓存寄存器

24.2.4.3 SPI 接收数据寄存器 (SPIx_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
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
RDR[15:0]															
RO															
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	**RDR[15:0]**	**接收缓存寄存器** 当SPDATL=1,为16位SPI接收缓存寄存器 当SPDATL=0,为8位SPI接收缓存寄存器

24.2.4.4 SPI 控制寄存器 (SPIx_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															
-															
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	SPDMAT	SPDMAR	SPTIE	SPRIE	SPDATL	DIR	MSTR	CPHA	CPOL	SSDIS	SPR[3:0]			
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	**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	**SPDMAT**	**SPI数据发送DMA通道使能** 0：不使用DMA来传输SPI的发送数据 1：使用DMA来传输SPI的发送数据
12	**SPDMAR**	**SPI数据接收DMA通道使能** 0：不使用DMA来传输SPI的接收数据 1：使用DMA来传输SPI的接收数据
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	**SPDATL**	**数据字长度选择位** 0：每次发送8bit数据 1：每次发送16bit数据
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

24.3 两线串行接口（TWI）

24.3.1 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。

SH32F284 包含 1 个 TWI（TWI1）。

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。

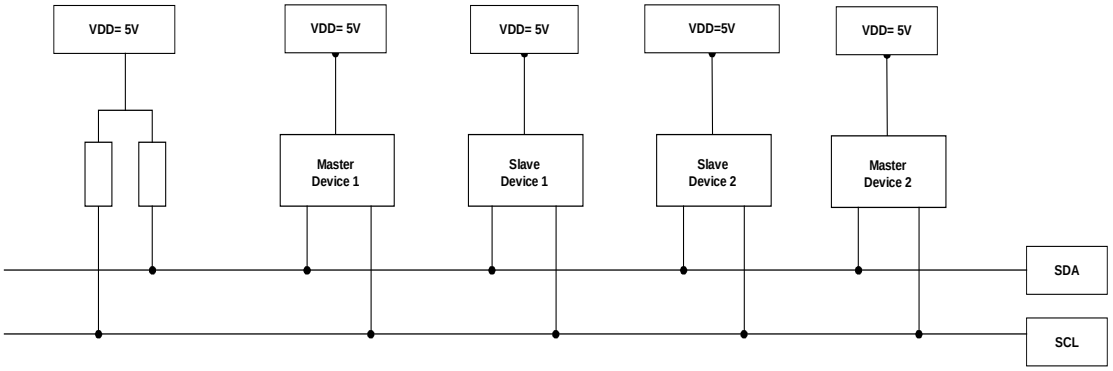


图24-24　　典型 TWI 总线应用框图

24.3.2 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

24.3.3 功能描述

24.3.3.1　　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。

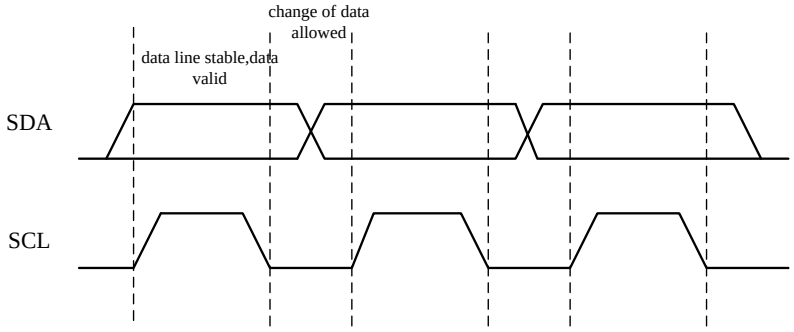


图24-25　　TWI 总线的基本位传输特性

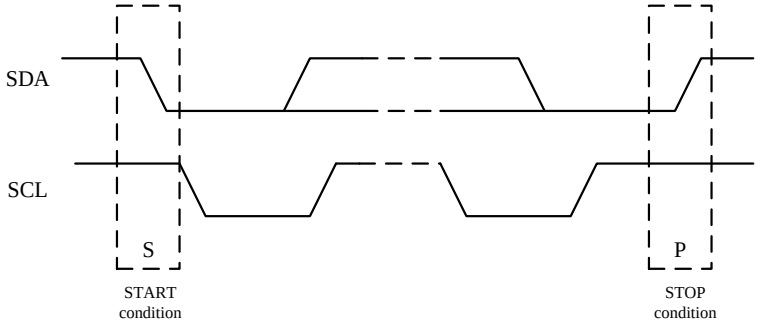


图24-26　　TWI 总线的起始条件（起始信号）和停止条件（停止信号）

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。

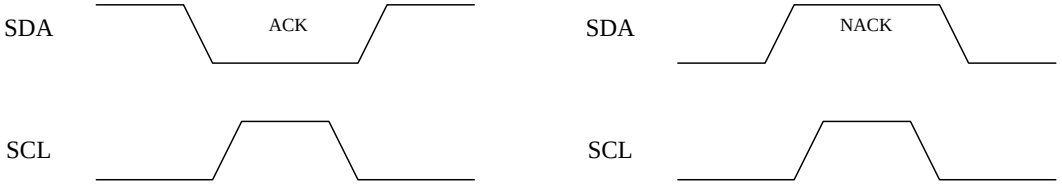


图24-27　　ACK 和 NACK 信号

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，SH32F284 拉低 SDA 信号线。中断标志位置起期间，SH32F284 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。

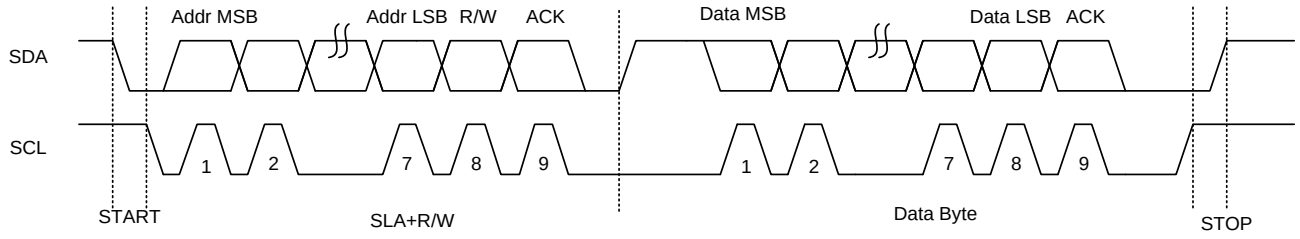


图24-28　一次典型的 TWI 数据传输时序图

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。

24.3.3.2　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。

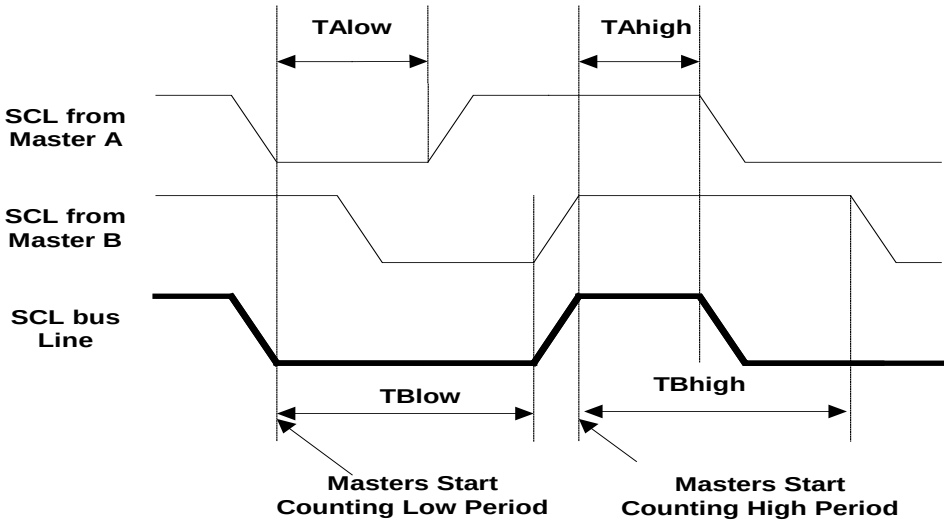


图24-29　时钟同步时序图

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。

24.3.3.3　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：

（1）主机 1 发送重复起始条件，主机 2 发送数据

（2）主机 1 发送停止条件，主机 2 发送数据

（3）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。

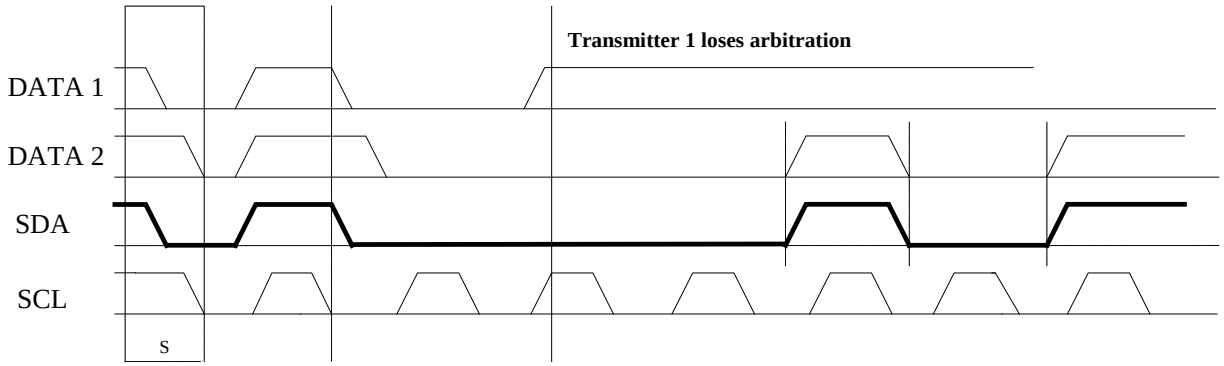


图24-30　　数据仲裁时序图

24.3.3.4　　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。

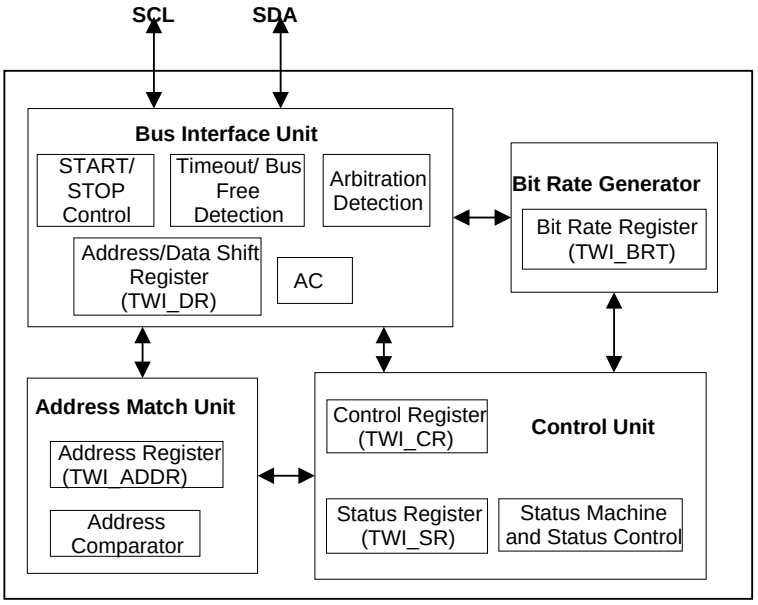


图24-31　　TWI 模块结构图

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 SH32F284 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。

SH32F284 在传输数据/地址时，必须在 SCL 由低跳高前维持数据稳定。

SH32F284 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。

SH32F284 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。SH32F284 当前通讯终止，该状态与普通 00H 状态处理一致。

SH32F284 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。SH32F284 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。SH32F284 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。

SH32F284 在当前通讯终止后，不会再参与当前传输。SH32F284 若作为主机存在，请开启 EFREE 功能，防止进入逻辑死区。

SH32F284 规定总线维持高电平超过 TFREE =Tsys*TWI_HOC*256（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）。SH32F284 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。SH32F284 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)。

注：TWI 模块工作在 30MHz 时，EFREE=1 时，TFREE 最大 255×256/30000=2.176ms，最小 256/30000=0.00853ms。
当 EFREE=0 时，TFREE 固定 25000/30000=0.83ms。

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。

注：TWI 模块工作在 30MHz 时，TOUT 最大 200000/30000=6.67ms，最小 25000/30000=0.83ms。

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。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。

24.3.3.5 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：

- 主机发送模式
- 主机接收模式
- 从机接收模式
- 从机发送模式

有关这四种模式下文会有详细介绍。

SH32F284 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 TWI 通讯中断（TWINT）外，有时还需要开启总线超时（TOUT）和 SCL 高电平超时（TFREE）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“TWI 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
Ā ：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。

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。

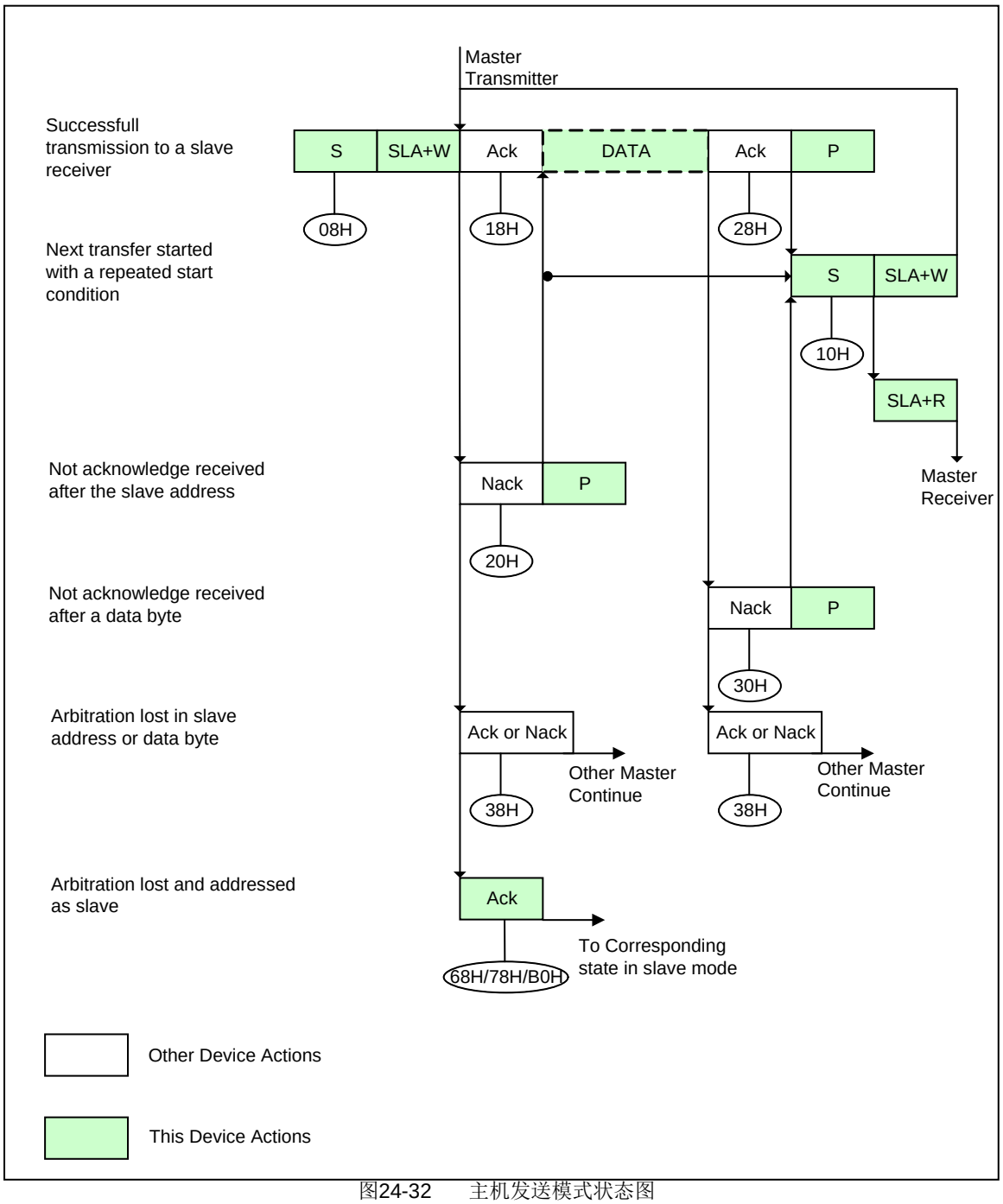


图24-32　　主机发送模式状态图

表24-3　　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。

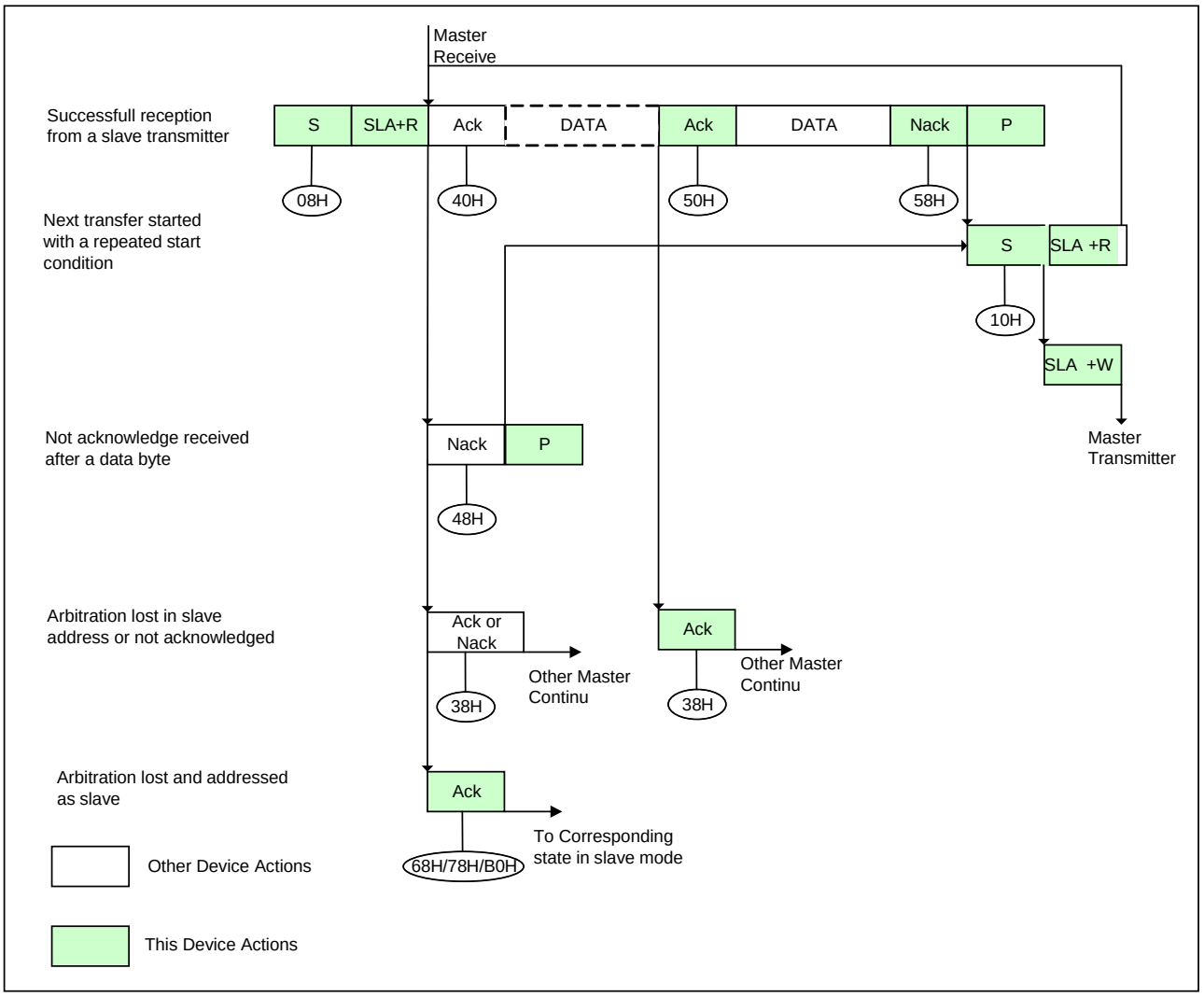


图24-33　　主机接收模式状态图

表24-4　　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 TWIADR 为 SH32F284 准备相应的地址。如果 GC 置位，SH32F284 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，SH32F284 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。

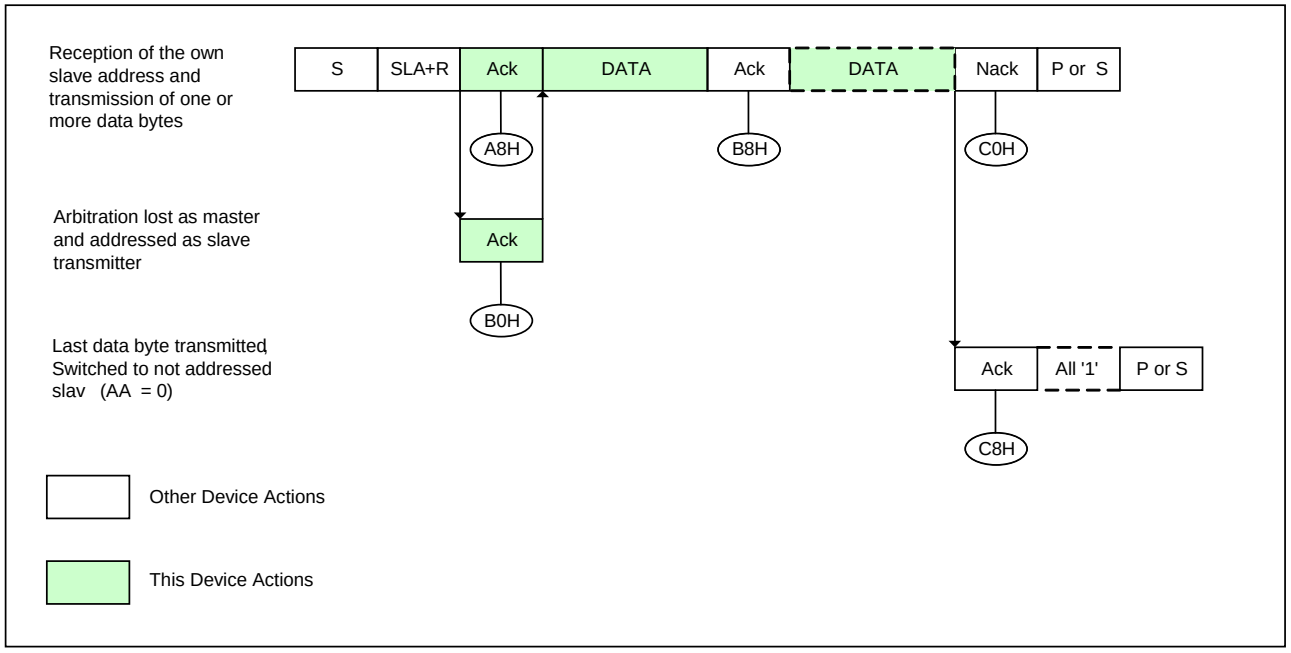


图24-34　　从机发送模式状态图

表24-5　　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 SH32F284 准备相应的地址。如果 GC 置位，SH32F284 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，SH32F284 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，SH32F284 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 恢复对自己地址的相应。可以通过 AA=0 暂时将 SH32F284 从总线隔离。

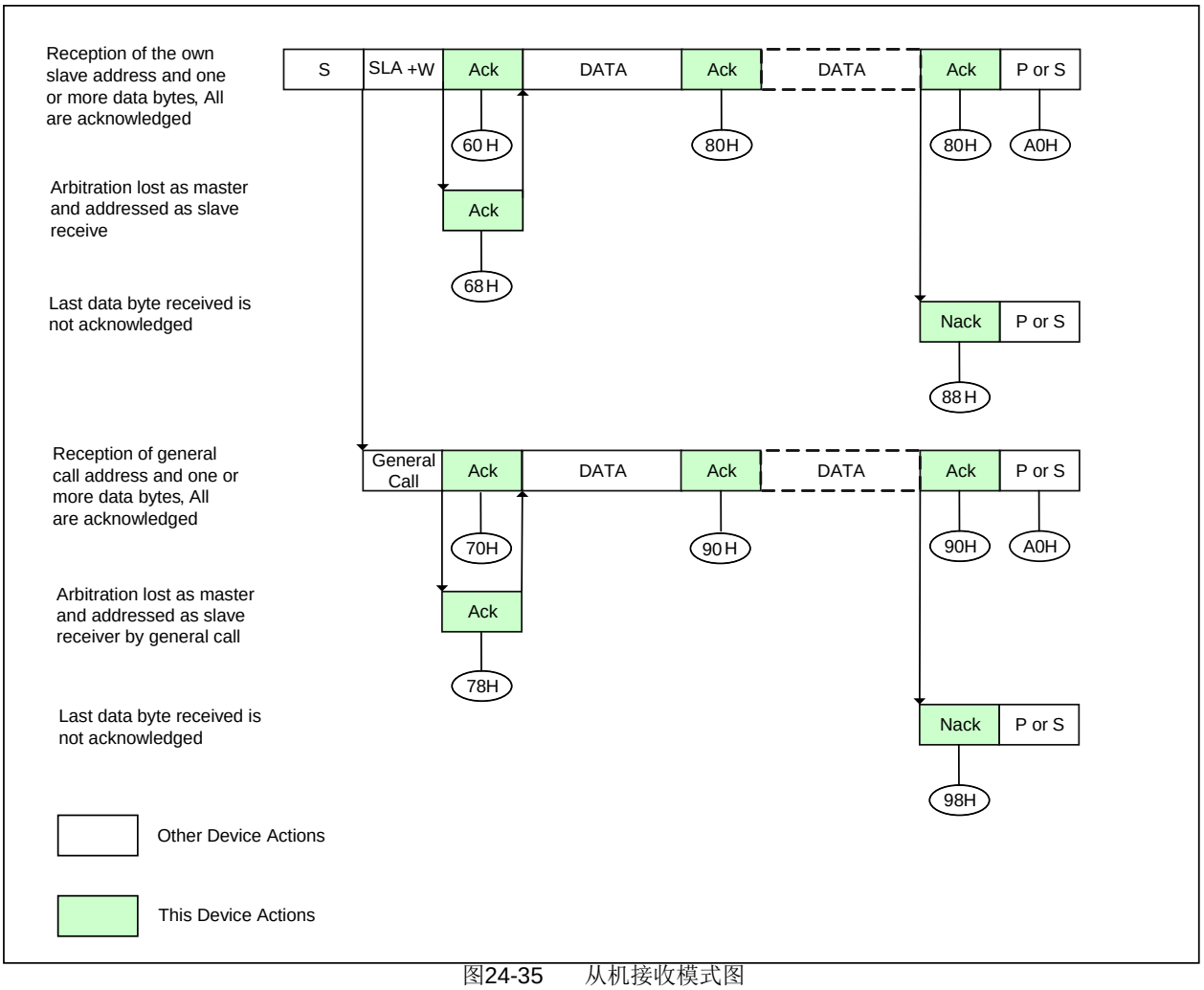


图24-35　　从机接收模式图

表24-6　　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	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，SH32F284 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24-7　　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

24.3.3.6　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。

24.3.3.7　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，三个标志任一个置位都会引起TWI中断，三者共享中断向量，标志必须由软件清0。

TOUT：总线超时标志。当时钟线（SCL）低电平超过N*Ttwi时发生总线超时（事件）。N值由CNT[1:0]@TWI_CR寄存器位设定。总线超时检测功能是自动开启的，只要TWI工作该功能就有效（无法关闭）。但总线超时（事件）要引起TWI中断，首先需要产生TOUT标志，而TOUT标志的产生又受ETOT控制，ETOT使能，TOUT才能产生，此时如果TWI总中断使能（TWINTE=1）就会引起TWI中断。反之ETOT未使能，或TWINTE未使能，都不会产生TWI中断。

TFREE：SCL高电平超时标志。当时钟线（SCL）高电平超过HOC *256*Ttwi时发生SCL高电平超时（事件）。HOC值由HOC[7:0]@TWI_HOC寄存器位设定。SCL高电平超时检测功能是自动开启的，只要TWI工作该功能就有效（无法关闭）。但SCL高电平超时（事件）要引起TWI中断，首先需要产生TFREE标志，而TFREE标志的产生又受EFREE控制，EFREE使能，TFREE才能产生，此时如果TWI总中断使能（TWINTE=1）就会引起TWI中断。反之EFREE未使能，或TWINTE未使能，都不会产生TWI中断。

TWINT：TWI通讯中断标志，当TWI模块使能时即开启。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总中断使能位，当TWINTE使能，并且ETOT和EFREE都开启时，TOUT/TFREE/TWINT任一个（中断）事件发生都会引起TWI中断。如果关闭总中断使能TWINTE，则应用软件只能通过查询方式进行TWI通信处理。

24.3.3.8　TWI通信服务程序开发

SH32F284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：

（1）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。

（2）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。

（3）支持4种TWI操作模式的多种状态服务程序，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）。

24.3.4 TWI 寄存器

TWI 模块寄存器列表 (基地址:0x0x4000 3400)

地址	寄存器名	说明
0x4000 3400	FR	TWI 中断标志寄存器
0x4000 3404	STAT	TWI 状态寄存器
0x4000 3408	HOC	TWI 高电平超时检测配置寄存器
0x4000 340C	BRT	TWI 比特率配置寄存器
0x4000 3410	DR	TWI 数据寄存器
0x4000 3414	ADDR	TWI 地址配置寄存器
0x4000 3418	CR	TWI 控制寄存器

24.3.4.1 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：无超时标志发生 1：当 TWI 总线低电平超过 N*Tsys 时置位。N 值由 CNT@TWI_CR 寄存器位决定
2	TFREE	SCL 高电平超时标志位 0：无超时标志发生 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：无通讯标志产生 1：单字节数据传送完毕，由硬件置位

24.3.4.2 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	-

24.3.4.3 TWI 高电平超时检测配置寄存器 (TWI_HOC)

偏移地址: 0x0008
复位值: 0x0000 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								HOC[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	HOC[7:0]	TWI 的 SCL 总线高电平超时检测配置位

24.3.4.4 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

24.3.4.5 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

24.3.4.6 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

24.3.4.7 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
TWIEN	保留				TWINTE	ETOT	EFREE	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	**总线超时标志使能位** 0：禁止 TWI 总线超时产生 TOUT 标志（TOUT 中断也无法产生） 1：允许 TWI 总线超时产生 TOUT 标志（允许 TOUT 产生中断）
8	EFREE	**SCL 高电平超时标志使能位** 0：禁止 SCL 高电平超时产生 TFREE 标志（TFREE 中断也无法产生） 1：允许 SCL 高电平超时产生 TFREE 标志（允许 TFREE 产生中断）
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）

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 SH32F284 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)。

26. 调试接口

26.1 主要特性

SH32F284 使用 Cortex™-M3 内核，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。

SH32F284 支持 SWJ 调试接口，当 SH32F284 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 Cortex™-M3 内核提供集成的片上调试功能。它由以下部分组成：

- SWJ-DP：串行/JTAG 调试端口
- AHP-AP： AHB 访问端口
- ITM：执行跟踪单元
- FPB：闪存指令断点
- DWT：数据触发
- TPUI：跟踪单元接口(仅较大封装的芯片支持)

有关调试接口的更多信息请参考ARM官方手册"Cortex-M3 Technical Reference Manual - Revision r2p1"。（汉译版本为"Cortex-M3技术参考手册"），以及ARM开发工具集技术参考手册。

26.2 SWJ 调试端口(serial wire and JTAG)

SH32F284 内核集成了串行/JTAG 调试接口(SWJ-DP)。这是标准的 ARM CoreSight 调试接口，包括 JTAG-DP 接口(5 个引脚)和 SW-DP 接口(2 个引脚)。

- JTAG 调试接口(JTAG-DP)为 AHP-AP 模块提供 5 针的 JTAG 接口。
- 串行调试接口(SW-DP)为 AHP-AP 模块提供 2 针(时钟＋数据)接口。

在 SWJ-DP 接口中， SW-DP 接口的 2 个引脚和 JTAG 接口的 5 个引脚中的一些是复用的。

26.2.1 SWJ 调试端口引脚

SH32F284 的 5 个普通 I/O 口可用作 SWJ-DP 接口引脚。这些引脚在所有的封装里都存在。

表26-1 SWJ 调试端口引脚

SWJ-DP 端口引脚名称	JTAG 调试接口		SW 调试接口		引脚分配
	类型	描述	类型	调试功能	
JTRST	输入	JTAG 模块复位	-	-	PA2
JTDI	输入	JTAG 数据输入	-	-	PA3
JTMS/SWDIO	输入	JTAG 模式选择	输入/输出	串行数据输入/输出	PA4
JTCK/SWCLK	输入	JTAG 时钟	输入	串行时钟	PA5
JTDO/SWO	输出	JTAG 数据输出	-	-	PA6

26.2.2 JTAG 脚上的内部上拉和下拉

请参考"GPIO"章节相关介绍。

26.2.3 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 SWJ-DP 的所有 5 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，SH32F284 微控制器可以把 SWJ 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(JTRST, JTMS,J TDI)，带下拉的输入(JTCK)，和输出(JTDO)状态，并持续一段时间，直到用户代码释放这些引脚。

当这些引脚仍被配置成调试接口时(JTAG 或者 SW)，修改相应的普通 I/O 口配置寄存器是无效的。

表26-2　　SWJ_DP 引脚分配

SWJCFG[2:0]	配置为调试专用的引脚	SWJ 接口的 I/O 口分配					JTAG 调试接口是否可用	SW 调试接口 是否可用
		PA5/JTMS /SWDIO	PA6/JTCK /SWCLK	PA7/JTDO /SWO	PA4/JTDI	PA3/JTRST		
000	所有的 SWJ 引脚 (JTAG-DP + SW-DP) 复位状态	调试专用	调试专用	调试专用	调试专用	调试专用	可用	可用
001	所有的 SWJ 引脚 (JTAG-DP + SW-DP) 除了JTRST引脚	调试专用	调试专用	调试专用	调试专用	用户可用	可用	可用
010	JTAG-DP 接口禁止， SW-DP 接口允许（含 SWO）	调试专用	调试专用	调试专用	用户可用	用户可用	不可用	可用
011	JTAG-DP 接口禁止， SW-DP 接口允许（不含 SWO）	调试专用	调试专用	用户可用	用户可用	用户可用	不可用	可用
1XX	JTAG-DP 接口禁止， SW-DP 接口禁止	用户可用 （**Note1**）	用户可用 （**Note1**）	用户可用	用户可用	用户可用	不可用	可用
注：该模式下，JTMS，JTCK 在正常运行状态下可复用为普通 I/O，但在调试模式状态下仍强制作为调试接口。								

26.3 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、GPT、TIM5-8、UART、SPI、TWI、DMA、IWDT、WDT 的时钟控制

26.3.1 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。

同样的，SLEEP 模式也有 DBG_SLEEP 进行 HCLK 和 FCLK 的配置。

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。

26.3.2 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、GPT、TIM5-8、UART、SPI、TWI、DMA、IWDT、WWDT 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。

26.3.3 调试 MCU 配置寄存器

SYSCFG 模块寄存器列表 (基地址:0x0x4004 4000)

地址	寄存器名	说明
0x4004 4008	SAFR	系统配置寄存器
0x4004 4010	DBGCR	调试接口控制寄存器

26.3.3.1 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	SWJCFG[2:0]			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 ~ 2	**SWJCFG[2:0]**	**串行线JTAG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SWJ和跟踪复用功能的I/O口。SWJ(串行线JTAG)支持JTAG或SWD访问Cortex M3的调试端口。系统复位后的默认状态是启用SWJ但没有跟踪功能，这种状态下可以通过JTMS/JTCK脚上的特定信号选择JTAG或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，仅在SWD接口有效。 000：完全SWJ(JTAG + SWD)，SWJ调试口无法复用为普通I/O，JTAG，SWD调试接口均可使用。 001：完全SWJ(JTAG + SWD)，JTRST可复用为普通I/O；JTAG，SWD调试接口均可使用。 010：关闭JTAG，只用SWD（包含SWO），JTRST，JTDI可复用为普通I/O；JTAG调试接口无法使用，只能使用SWD调试接口。 011：关闭JTAG，启用SWD（不包含SWO），JTRST，JTDI，JTDO可复用为普通I/O；JTAG调试接口无法使用，只能使用SWD调试接口。 1XX：关闭JTAG，关闭SWD；JTRST，JTDI，JTDO可复用为普通I/O，但JTMS（SWDIO），JTCK（SWCLK）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，此时JTAG调试接口无法使用，SWD调试接口仍可使用。 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JTAG接口无法使用，但可以通过SWD接口恢复，同时需配合reset信号。
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；

26.3.3.2 调试接口控制寄存器 (SYSCFG_DBGCR)

偏移地址: 0x0010

复位值: 0x0000 0080

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_GPT	DBG_WWDT	DBG_IWDT	DBG_DMA	保留					DBG_STOP	DBG_SLEEP
RW	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	*RW*	-					*RW*	*RW*
0	0	0	0	0	0	0	0	1	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TIM5-TIM8模块停止工作** 0：TIM5-TIM8模块停止工作 1：TIM5-TIM8模块仍然正常工作
10	**DBG_GPT**	**当内核进入调试状态时GPT停止工作** 0：GPT停止工作 1：GPT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_DMA**	**当内核进入调试状态时DMA停止工作** 0：DMA停止工作 1：DMA仍然正常工作
6 ~ 2	**保留**	-
1	**DBG_STOP**	**调试停机模式。** 0：在停机模式时，系统时钟保持，由内部RC振荡器提供系统时钟(包括HCLK和FCLK)。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：在停机模式时，系统时钟关闭，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。
0	**DBG_SLEEP**	**调试睡眠模式** 0：在睡眠模式时，系统时钟保持，FCLK和HCLK时钟都由原先配置好的系统时钟提供。 1：在睡眠模式时，系统时钟关闭，FCLK由原先已配置好的系统时钟提供，HCLK则关闭。由于睡眠模式不会复位已配置好的时钟系统，因此从睡眠模式退出时，软件不需要重新配置时钟系统。

27. 用户代码选项

OP_LVREN：

0：禁止低电压复位功能（默认）
1：允许低电压复位功能

OP_LVRLE：

00：低电压复位设定电压为4.1V（默认）
01：低电压复位设定电压为3.7V
1x：低电压复位设定电压为2.5V

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_HSE16M：

0:晶振高频模式（16M）关闭（默认）
1: 晶振高频模式（16M）开启

OP_CSM：

0：上电后CSM模块即打开（默认）
1：CSM模块保持关闭

OP_MCM1PIN：

MCM1模块 PWM波形输出顺序：

引脚 / 选项	PB13	PB12	PB11	PB10	PB9	PB8
00（默认）	PWM0	PWM1	PWM2	PWM01	PWM11	PWM21
01	PWM21	PWM11	PWM01	PWM2	PWM1	PWM0
10	PWM0	PWM01	PWM1	PWM11	PWM2	PWM21
11	PWM21	PWM2	PWM11	PWM1	PWM01	PWM0

28. 电气特性

极限参数*

数字供电电压. . . . . -0.3V to +6.5V
模拟供电电压. . . . . -0.3V to +6.5V
数字输入/输出电压. . . . . GND-0.3V to VDD+0.3V
模拟输入电压. . . . . AGND-0.3V to AVDD +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, AVDD = VDD, GND = AGND = 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
数字工作电压	V_{DD}	2.4	5.0	5.5	V	4MHz ≤ f_{sys} ≤84MHz
模拟工作电压	AV_{DD}	VDD-0.1	VDD	VDD+0.1	V	
工作电流	I_{OP1}	-	50	60	mA	f_{sys} = **84MHz**，PLL开，V_{DD}= AV_{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	13	mA	f_{sys} = 8MHz，PLL关，V_{DD}= AV_{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
待机电流 (睡眠模式)	I_{SB1}	-	5	8	mA	f_{sys} = 8MHz，PLL关，V_{DD}= AV_{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}	-	15	-	μA	HSE关闭，PLL关，V_{DD}= AV_{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/3.7V/2.5V
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}		SWDIO、SWCLK、NRST引脚 V_{DD} = 2.4 – 5.5V
输入高电压2	V_{IH2}	0.85 X V_{DD}	-	V_{DD}		SWDIO、SWCLK 、NRST引脚 V_{DD} = 2.4 – 5.5V
输入低电压3	V_{IL3}	GND	-	0.8	V	仅PA2，PA3，PA8，PA11，PB0，PB1，PB4，PB5（输入高低压窗口0.4V） V_{DD} = 4.5 - 5.5V，TTL输入功能打开（Note4）
		GND	-	0.15 X V_{DD}	V	仅PA2，PA3，PA8，PA11，PB0，PB1，PB4，PB5（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PA2，PA3，PA8，PA11，PB0，PB1，PB4，

						PB5（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	PA2，PA3，PA8，PA11，PB0，PB1，PB4，PB5（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/下拉电阻1*	R_{PH1}	-	30	-	kΩ	V_{DD} =5.0V, V_{IN}=GND
上拉/下拉电阻2*	R_{PH2}	-	10	-	kΩ	仅PB0～PB4 此电阻精度较高，且不随V_{DD}变化的影响
输出高电压	V_{OH}	V_{DD} – 0.7	-	-	V	I/O端口, I_{OH} = -8mA, V_{DD} = 5.0V， *GPIOx_ODRIVER = 00b (Note5)*
输出高电压1	V_{OH1}	V_{DD} - 0.7			V	I/O端口, I_{OH} = -4mA, V_{DD} = 5.0V， *GPIOx_ODRIVER = 01b*
输出高电压2	V_{OH2}	V_{DD} - 0.7			V	I/O端口, I_{OH} = -2.5mA, V_{DD} = 5.0V， *GPIOx_ODRIVER = 10b*
输出高电压3	V_{OH3}	V_{DD} - 0.7	-	-	V	I/O端口, I_{OH} = -1mA, V_{DD} = 5.0V， *GPIOx_ODRIVER = 11b*
输出低电压	V_{OL}	-	-	GND + 0.6	V	I/O 端口, I_{OL} = 15mA, V_{DD} = 5.0V， *GPIOx_ODRIVER = 00b*
输出低电压1	V_{OL1}	-	-	GND + 0.6	V	I/O 端口, I_{OL} = 7mA, V_{DD} = 5.0V， *GPIOx_ODRIVER = 01b*
输出低电压2	V_{OL2}	-	-	GND + 0.6	V	I/O 端口, I_{OL} = 3.5mA, V_{DD} = 5.0V， *GPIOx_ODRIVER = 10b*
输出低电压3	V_{OL3}	-	-	GND + 1.5	V	I/O端口，I_{OL} = 120mA，V_{DD} = 5.0V， *GPIOx_ODRIVER = 11b*， 仅PA7～PA12、PE1、PE2

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200mA。流过GND的最大电流值须小于250mA。

3. 设计保证，实际测试时未包含温度。.

4. TTL输入功能参考SYSCFG模块的SYSCFG_SAFR寄存器定义。

5. 驱动能力选择参考GPIO模块的GPIOx_ODRIVER寄存器定义。

高速12BIT模/数转换器电气特性1（1LSB = AV_{DD}/4096）

转换速率最高2MSPS。（AV_{DD} = 2.7V～5.5V，AGND = 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 21.3.10"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，AV_{DD}= 5.0V
微分非线性误差	D_{LE}	-	±0.5	±2	LSB	AV_{DD} = 5.0V，V_{REF} = AV_{DD}，ADC CLK = 40MHz

积分非线性误差	I_{LE}	-	±1	±3	LSB	AV_{DD} = 5.0V，V_{REF} = AV_{DD}，ADC CLK = 40MHz
满刻度误差	E_F	-	±3	±5	LSB	AV_{DD} = 5.0V，V_{REF} = AV_{DD}，ADC CLK = 40MHz
偏移量误差	E_Z	-	±3	±7	LSB	AV_{DD} = 5.0V，V_{REF} = AV_{DD}，ADC CLK = 40MHz
总绝对误差	E_{AD}	-	-	±8	LSB	AV_{DD} = 5.0V，V_{REF} = AV_{DD}，ADC CLK = 40MHz
ADC工作时钟	f_{ADC}	0.03	-	40	MHz	AV_{DD} = 5.0V，V_{REF} = AV_{DD}
			-	30		AV_{DD} = 3.3V，V_{REF} = AV_{DD}
ADC采样时间	t_{SAMP}	0.125	-	-	μs	AV_{DD} = 5.0V，V_{REF} = AV_{DD}
		0.2	-	-		AV_{DD} = 3.3V，V_{REF} = AV_{DD}
ADC转换速率	F_{CON}	-	-	2	MSPS	AV_{DD} = 5.0V，V_{REF} = AV_{DD}
		-	-	1.4		AV_{DD} = 3.3V，V_{REF} = AV_{DD}
总转换时间	T_{CON}	16	-	159	t_{AD}	（1 t_{AD}～ 16 t_{AD}）+t_{GAP} + t_{COMP}

温度传感器特性（AV_{DD} = 2.7 - 5.5V，AGND = 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
工作温度*	T	-40		150	℃	
校正后绝对误差	T_L		±3	±5	℃	AV_{DD} = 5V, T_A = -40 ~ +150°C
温度传感器斜率	T_{Slope}		5.153		mV/℃	
25度时输出电压	V_{25}		1.5		V	

注意：

1. ""温度在105°C以上由设计保证，实际不测试。*

内部电压基准源特性（AVDD = 2.7 - 5.5V，AGND = 0V，TA = +25°C，除非另有说明），VREF引脚接4.7uF陶瓷电容

参数	符号	最小值	典型值	最大值	单位	条件
基准源输出精度	V_O	2.49	2.5	2.51	V	AV_{DD}= 5.0V，开启2.5V参考电压输出，T_A = 25°C
		3.29	3.30	3.31	V	AV_{DD}= 5.0V，开启3.30V参考电压输出，T_A = 25°C
负载调整率	ΔV_{O1}	-	0.05	0.2	%	AV_{DD}= 5.0V，V_O = 2.5V or 3.3V，T_A = 25°C I_{LOAD}= 0 ~ 2mA
输入电压调整率	ΔV_{O2}	-	0.05	0.1	%	AV_{DD}= 3.0~ 5.5V，V_O = 2.5V，T_A = 25°C ILOAD=1mA
		-	0.05	0.1	%	AV_{DD}=3.8~ 5.5V，V_O = 3.3V，T_A = 25°C ILOAD=1mA
参考源输出温度系数	TCV_O	-	50	150	ppm/°C	AV_{DD}=5.0V，V_O = 2.5V or 3.3V，T_A = -40 ~ +105°C*

注意：

1. ""温度在105°C以上由设计保证，实际不测试。*

比较器电气特性（AV_{DD} = 2.7 - 5.5V，AGND = 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	AV_{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	AV_{DD} = 5V，CMPxN = 1V，CMPxP = 0V Step to 2V
比较器施密特窗口1	V_{SMT1}	-	0	-	mv	AV_{DD} = 5V，无施密特窗口，下降沿窗口

比较器施密特窗口2	V_{SMT2}	-	10	16	mv	AV_{DD} = 5V，施密特窗口10mV，下降沿窗口
比较器施密特窗口3	V_{SMT3}	-	20	30	mv	AV_{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	AV_{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}	-	-	1%		AV_{DD} = 5V

放大器电气特性（AV_{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

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}	10	-	-	μs	低电平有效
复位引脚上拉电阻	R_{RPH}	-	30	-	kΩ	V_{DD} = 5.0V，V_{IN} = GND
频率精度 (RC)	|ΔF|/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）
	|ΔF|/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）
	|ΔF|/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	-	16	MHz	
PLL输入时钟Duty	D_{PLL_IN}	40	-	60	%	
PLL输出时钟频率	f_{PLL_OUT}			120	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}	3.6	3.7	3.8	V	LVR2使能
LVR电压3	V_{LVR3}	2.4	2.5	2.6	V	LVR3使能
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	

外设模块功耗（VDD = 2.4 - 5.5V，GND = 0V，TA = +25°C，除非另有说明）

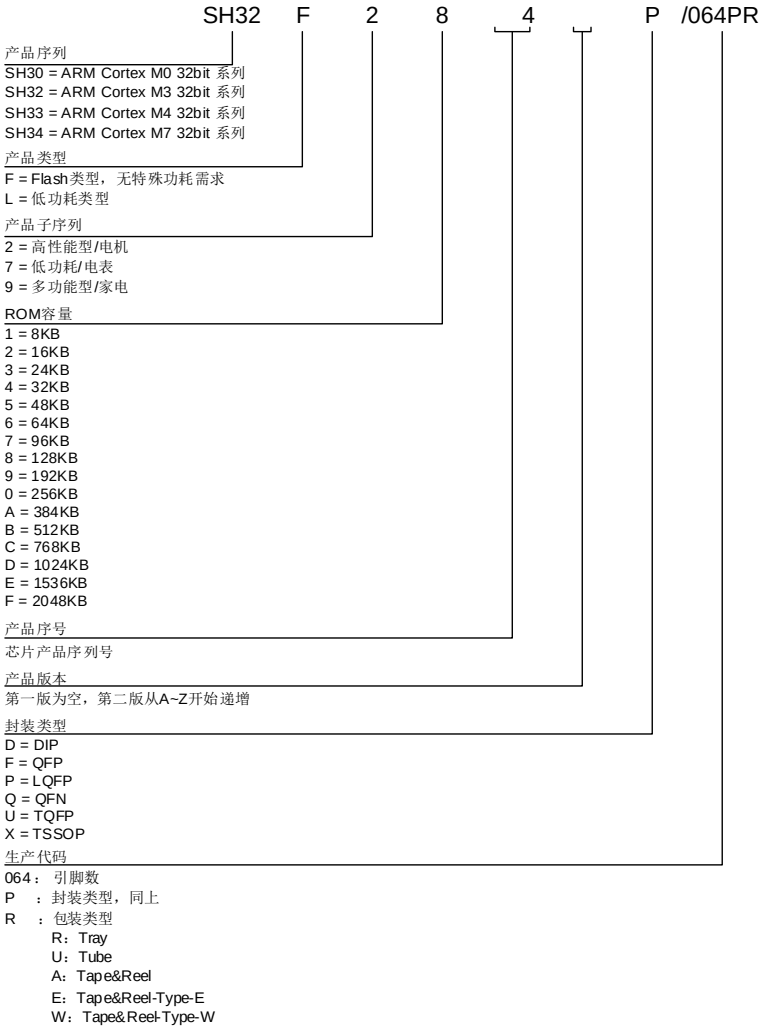
总线	模块	典型电流（主频84MHz）	典型电流（主频8MHz）	单位	条件
AHB（AHB时钟取主频时钟）	GPIO	0.76	0.05	mA	整个GPIO模块
	ADC	1.04	0.09		单个ADC电流
	DMA	5.42	0.51		整个DMA模块
	MACP	4.86	0.46		整个MACP模块
	CRC	0.35	0.02		
	GPT	1.89	0.17		单个GPT电流
APB2（APB2时钟取AHB二分频）	MCM	1.67	0.15	mA	单个MCM电流
APB1（APB1时钟取AHB四分频）	TIM	0.16	0.01	mA	单个TIM电流
	QEI	0.17	0.01		
	UART	0.15	0.01		单个UART电流
	SPI	0.16	0.01		单个SPI电流
	TWI	0.12	0.01		
	WWDT	0.08	0.01		
	AMOC	0.11	0.01		整个AMOC模块

注意：

1. 以上功耗仅包含模块时钟开启，模块处于ready状态，实际模块工作功耗还与外围电路参数有关，此处不测试。

29. 订购信息

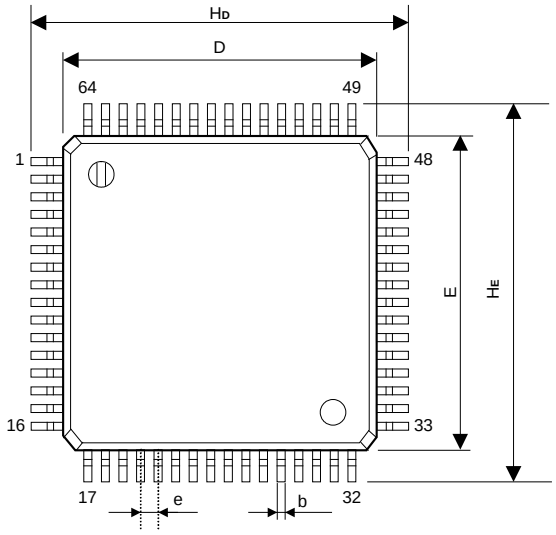
产品编号	封装
SH32F284P/064PR	LQFP64
SH32F284U/048UR	TQFP48

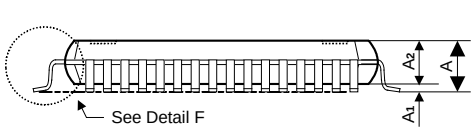


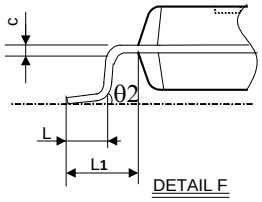
30. 封装信息

LQFP64（10×10）外形尺寸

单位: 英寸/毫米







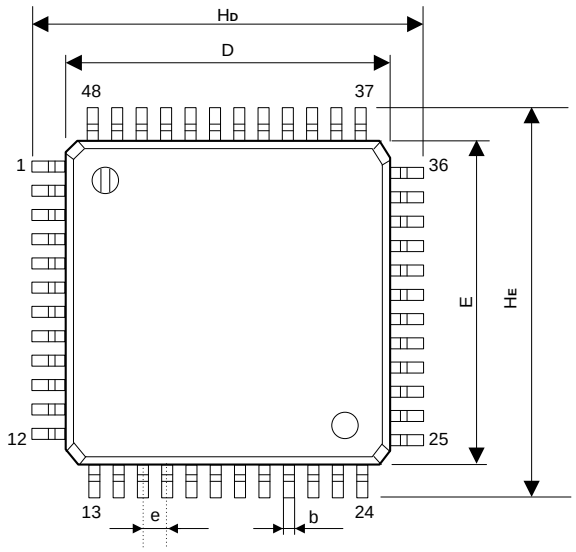
Symbol	Dimensions in inches		Dimensions in mm	
	MIN	MAX	MIN	MAX
A	---	0.063	---	1.6
A1	0.002	0.006	0.05	0.15
A2	0.053	0.057	1.35	1.45
D	0.388	0.400	9.85	10.15
E	0.388	0.400	9.85	10.15
H_D	0.465	0.480	11.8	12.2
H_E	0.465	0.480	11.8	12.2
b	0.007	0.011	0.17	0.27
e	0.016BSC		0.50BSC	
c	0.004	0.008	0.09	0.20
L	0.018	0.030	0.45	0.75
L1	0.033	0.045	0.850	1.150
θ2	0°	10°	0°	10°

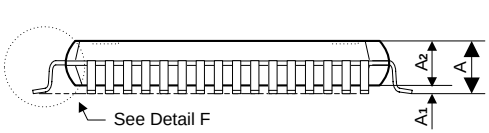
注意:

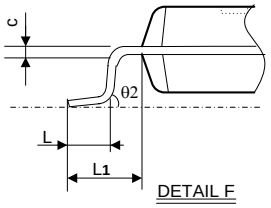
1. 封装尺寸不包括模的毛边凸起或门毛刺。
2. 如无特殊规定，容差为±0.1毫米。
3. 共面性：0.1毫米。
4. 控制尺寸为毫米。

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°

31. 规格更改记录

版本	记录
1.0	初始版本

32. 目录