NV32F100x 参考手册

V1.14

官网二维码

导言

本参考手册针对应用开发，提供关于如何使用 NV32F100x 系列控制器的存储器和外设的详细信息。关于 NV32F100x 的尾缀命名规则请参考 NV32F100x 用户手册。
NV32F100x 系列拥有不同的存储器容量、封装和外设配置。
关于订货编号、电气和物理性能参数，请参考 NV32F100x 用户手册。
关于芯片内部 Flash 编程、擦除和加密操作请参考 NV32F100x Flash 编程手册。
关于 Cortex M0+内核的具体信息，请参考 Cortex M0+技术参考手册。

相关文档

- Cortex M0+技术参考手册，下载链接网址：
http://infocenter.arm.com/help/topic/com.arm.doc.ddi0484c/DDI0484C_cortex_m0p_r0p1_trm.pdf
- Cortex M0+用户手册，下载链接网址：
http://infocenter.arm.com/help/topic/com.arm.doc.dui0662b/DUI0662B_cortex_m0p_r0p1_dgug.pdf

下述文档可以在 Navota 网站下载(http://www.navota.com)：

- NV32F100x 用户手册。
- NV32F100x 参考手册。
- NV32F100x Flash 编程手册。
- NV32F100x PDK 开发包。

推荐开发者参阅学习《ARM Cortex-M0 权威指南》，作者：ARM 公司设计专家 Joseph Yiu。

缩写词

缩写词	英文全称	中文对照
ACMP	Analog Comparators	模拟比较器
AWIC	Asynchronous Wakeup Interrupt Controller	异步唤醒中断控制器
AHB	Advanced High Performance Bus	高级高性能总线
APB	Advanced Peripheral Bus	高级外设总线
BOS	Bit Operation Storage	位操作存储
CAN	Controller Area Network	控制器局域网总线
CRC	Cyclic Redundancy Check	循环冗余码校验
CRBS	Crossbar Switch	交叉开关
DAP	Debug Access Port	调试访问端口
DMA	Direct Memory Access	直接内存存取
ETM	Enhance Timer	增强型定时器
EFM	Embedded Flash Memory	嵌入式 Flash 存储器
FLL	Frequency-Locked Loop	锁频环
FMC	Flash Memory Controller	Flash 存储器控制器
GPIO	General Purpose Input/Output	通用输入/输出
HMI	Human-machine Interfaces	人机界面
ICS	Internal Clock Source	内部时钟源
KBI	Keyboard Interrupt	键盘中断
LOC	Loss-of-Clock	时钟丢失
LPO	Low-Power Oscillator	低功率振荡器
LVD	Low-voltage Detect	低电压检测
LVW	Low-voltage Warning	低电压警报
MCM	Miscellaneous Control Module	杂项控制模块
NVIC	Nested Vectored Interrupt Controller	嵌套向量中断控制器
PBB	Peripheral Bus Bridge	外设总线桥
PIT	Periodic Interrupt Timers	周期中断定时器
PMC	Power Management Controllers	电源管理控制器
PWM	Pulse Width Modulation	脉冲宽度调制
PWT	Pulse Width Timer	脉冲宽度定时器
RTC	Real Time Clock	实时时钟
SIM	System Integration Module	系统集成模块
SPI	Serial Peripheral Interfaces	串行外设接口
SW	Software Reset	软复位
SWD	Serial Wire Debug	串口线调试
SysTick	System Tick Timer	系统节拍定时器
UART	Universal Asynchronous Receiver/Transmitter	通用异步收发器
WDOG	Watchdog	看门狗

第 1 章 系统架构和内核模块

■ 1.1 简介

本章对基于 ARM Cortex-M0+内核的 NV32F1xx 系列 MCU 系统架构进行概要说明，其中包括 M0+内核架构、M0+内核配置、NV32F1xx 系统架构、存储映射、嵌套向量中断控制、异步唤醒中断、复位引导、时钟分配、电源管理、下载调试等内容。

■ 1.2 ARM Cortex-M0+ 处理器简介

Cortex-M0 处理器是基于冯·诺依曼架构(单总线接口），即指令和数据共享同一总线，总线为两级流水，32 位的精简指令集(RISC)，其运算能力可以达到 0.9 DMIPS/MHz。该内核架构简单，方便存储器管理，系统能效高，并且具有良好的电磁干扰特性(EMI)。

ARM Cortex-M0+核是基于 32 位 ARMv6 内核架构，支持 Thumb-2 ISA 子集，指令集 100%向下兼容 Cortex-M0 内核指令，同时向上兼容 Cortex-M3/M4 内核指令。ARM Cortex-M0+的改进包括 1 个 ARMv6Thumb-2 DSP，该 DSP 采用 ARMv6-A/R 配置架构，提供 32 位指令，具有 SIMD（单指令多数据）DSP 类型的乘法/加法和饱和算法，支持单周期 32x32 乘法器。NV32F100x 系列 MCU 采用的 Cortex-M0+内核版本是 r0p0。

为了更多关于 ARM Cortex-M0+内核信息请访问 http://www.arm.com 网站。

■ 1.2.1 ARM Cortex-M0+ 内核架构

ARM Cortex-M0+内核架构包括唤醒中断控制器(WIC)、嵌套向量中断控制器（NVIC)、处理器内核、调试子系统、内部总线系统、AHB-Lite 总线接口单元和 JTAG/SWD 调试接口等模块。如下图所示：

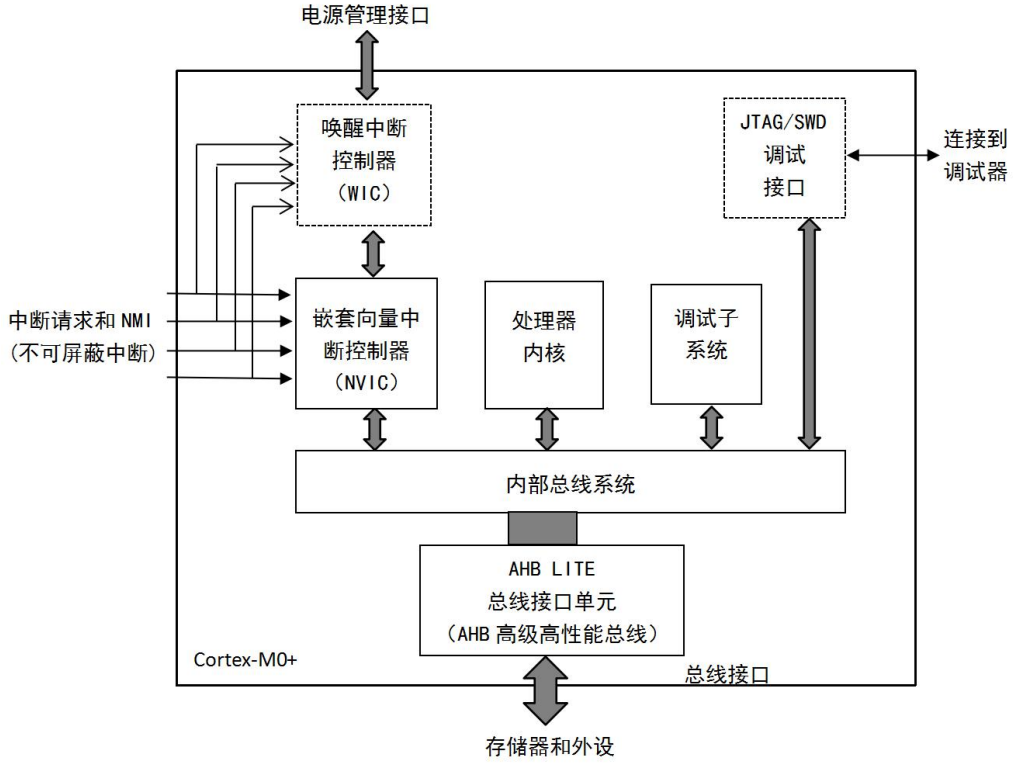


图 1-1 ARM Cortex-M0+ 内核架构

嵌套向量中断控制器（NVIC）可以处理最多 32 个中断请求和一个不可屏蔽中断（$\overline{\text{NMI}}$）输入。NVIC 需要比较正在执行中断和处于请求状态中断的优先级，然后自动执行高优先级中断。如果要处理一个中断，NVIC 会和处理器进行通信，通知处理器执行正确的中断处理。

唤醒中断控制器（WIC）为可选的单元，在低功耗应用中，在关闭了处理器大部分模块后，微控制器会进入待机状态。此时，WIC 可以在 NVIC 和处理器处于休眠的情况下，执行中断屏蔽功能。当 WIC 检测到一个中断时，会通知电源管理部分给系统上电，让 NVIC 和处理器内核执行剩下的中断处理。

调试子系统包括多个功能模块，以处理调试控制、程序断点和数据监视点（data watchpoint）。当调试事件发生时，处理器内核会被置于暂停状态，这时开发人员可以检查当前的处理器状态。

JTAG（联合测试行动小组）和 SWD（串行线调试）提供了通向总线系统和调试功能的入口。JTAG 是通用的 5 针通信协议，一般用作测试。串行线协议为新扩展的，只需两根线就可以了（时钟线和数据线），而且可以实现 JTAG 相同的调试功能。

内部总线系统、处理器内核的数据通路以及 AHB-Lite 总线接口都是 32 位宽的。AHB-Lite 是片上总线协议,已应用于多款 ARM 处理器。这种总线协议在“高级微控制器总线架构说明”(Advanced Microcontroller Bus Architecture specification）中有所体现。AMBA 是 ARM 开发的总线架构，已经广泛应用于 IC 设计领域。

■ 1.2.2 ARM Cortex-M0+ 配置选项

表 1-1 ARM Cortex-M0+ 内核配置选项

特性	Cortex-M0+ 配置选项	NV32F100x 配置
中断数量	0-32 个	32 个
数据字节排序	低位字节排序 或 高位字节排序	低位字节排序
SysTick Timer	有效 or 无效	有效
监视点数	0，1，2	2
硬件断点数	0，1，2，3，4	2
停止模式调试	有效（CPU 完全停止，寄存器可以读写）or 无效	有效
乘法器	快速（单周期） or 小型（面积和功耗低）	快速（单周期）
单周期 I/O 口	有效 or 无效	有效
唤醒中断 WIC	有效（CPU 在休眠状态下掉电）or 无效	无效
向量表偏移寄存器	有效 or 无效	有效
特权指令	有效 or 无效	有效
内存保护单元	无效 or 8 个单元	8 个单元
复位所有寄存器	有效 or 无效	无效
取指位宽	16 位 or 32 位	32 位
架构时钟门控	实施架构时钟门控	有效
DAP 分控端口支持	支持 AHB 调试端口访问	有效
DAP ROM 表基本内容	DAP ROM 表	0xF000_2003
调试端口协议	JTAG or SWD	SWD

■ 1.2.3 NV32F1xx 系统架构

下图为 NV32F1xx 系列 MCU 的系统架构：

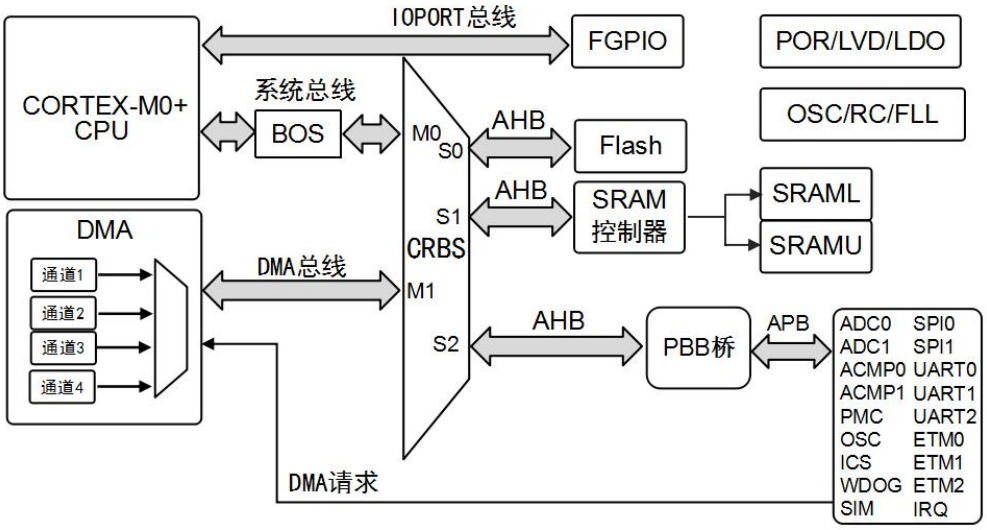


图 1-2 NV32F1xx 系统架构图

Cortex-M0+内核经过系统总线、位操作存储模块 BOS、交叉开关 CRBS 和 AHB 总线访问 Flash 和 SRAM 存储器，再经过 PBB 外设总线桥和 APB 外设总线访问各种外设模块；还可以通过 I/O 端口总线（IOPORT），对 FGPIO 进行单周期加载和存储。

交叉开关 CRBS 负责 Cortex-M0+内核系统总线和 DMA 总线之间的访问仲裁，仲裁利用轮换算法。AHB-Lite Bus 外设通过交叉开关 CRBS 与系统总线相连，允许 DMA 访问。PBB 桥负责 AHB-Lite 总线和 APB 总线间连接，通过 APB 外设总线，Cortext-M0+内核和 DMA 可以访问各种外设模块。

4 通道通用 DMA 可以管理存储器到存储器，外设到存储器和存储器到外设的直接访问。DMA 支持环形缓冲区的管理，在控制器达到缓冲区的末尾时不再需要用户代码的干预。每个通道连接到专用硬件 DMA 请求，支持软件对每个通道的触发。由软件完成 DMA 的配置，源和目标之间传输的数据量都是独立的。DMA 可以用于主要的外设： SPI，UART，ADC，Timer 模块。当前版本无 DMA 模块，请参阅用户手册。

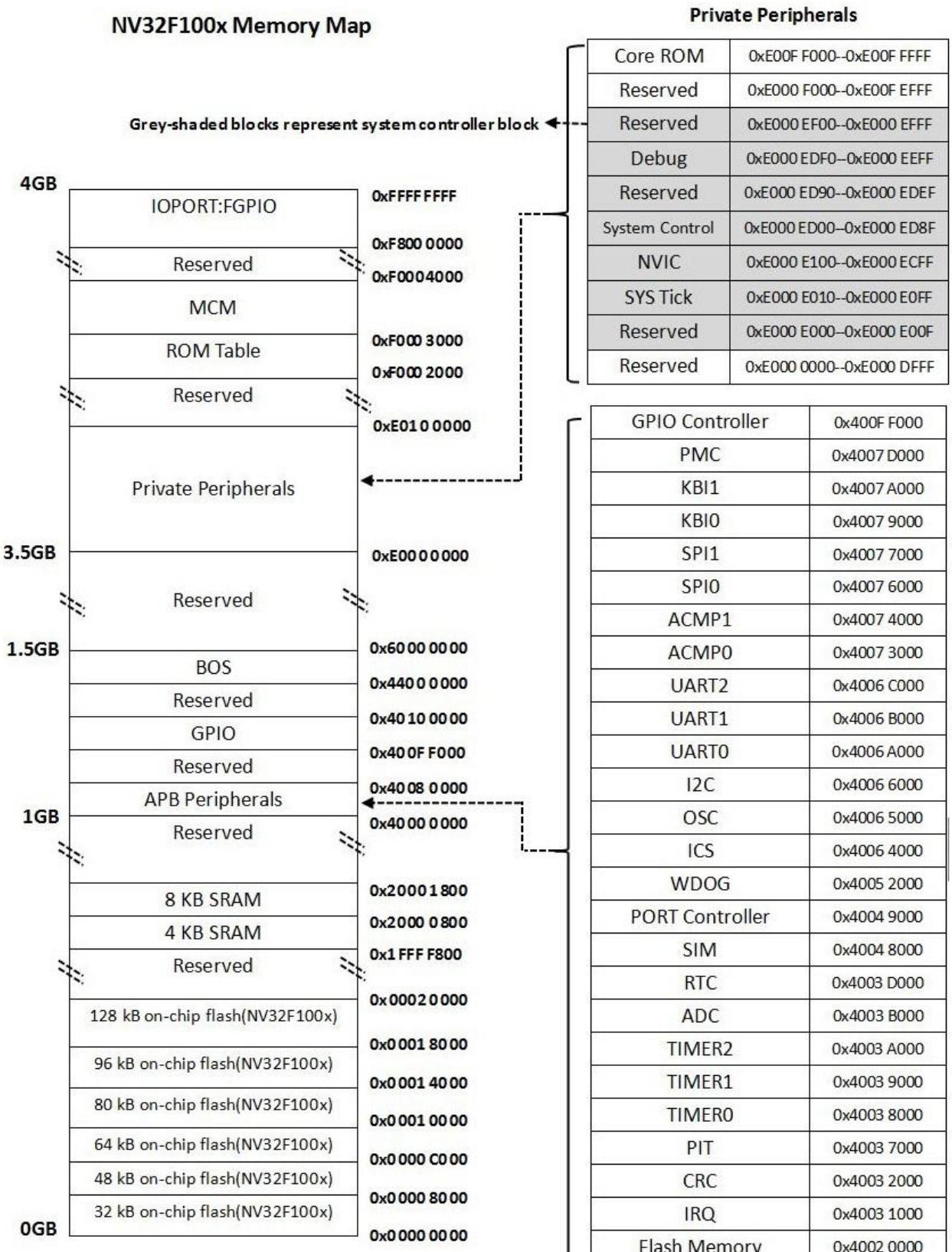
■ 1.3 存储器映射

■ 1.3.1 简介

程序存储器、数据存储器、寄存器和输入输出端口被组织在同一个 4GB 的地址空间内。

数据字节以低位优先的格式存放在存储器中，一个字里的最低地址字节被认为是该字的最低有效字节，而最高地址字节是最高有效字节。

■ 1.3.2 存储器地址映射


NV32F100x Memory Map
Grey-shaded blocks represent system controller block
4GB
IOPORT:FGPIO
0xFFFF FFFF
0xF800 0000
Reserved
0xF000 4000
MCM
ROM Table
0xF000 3000
0xF000 2000
Reserved
0xE010 0000
Private Peripherals
3.5GB
0xE000 0000
Reserved
1.5GB
0x6000 0000
BOS
0x4400 0000
Reserved
0x4010 0000
GPIO
0x400F F000
Reserved
0x4008 0000
APB Peripherals
1GB
0x4000 0000
Reserved
0x2000 1800
8 KB SRAM
0x2000 0800
4 KB SRAM
0x1FFF F800
Reserved
0x0002 0000
128 kB on-chip flash(NV32F100x)
0x0001 8000
96 kB on-chip flash(NV32F100x)
0x0001 4000
80 kB on-chip flash(NV32F100x)
0x0001 0000
64 kB on-chip flash(NV32F100x)
0x0000 C000
48 kB on-chip flash(NV32F100x)
0x0000 8000
32 kB on-chip flash(NV32F100x)
0GB
0x0000 0000
Private Peripherals
Core ROM 0xE00F F000--0xE00F FFFF
Reserved 0xE000 F000--0xE00F EFFF
Reserved 0xE000 EF00--0xE000 EFFF
Debug 0xE000 EDF0--0xE000 EEFF
Reserved 0xE000 ED90--0xE000 EDEF
System Control 0xE000 ED00--0xE000 ED8F
NVIC 0xE000 E100--0xE000 ECFF
SYS Tick 0xE000 E010--0xE000 E0FF
Reserved 0xE000 E000--0xE000 E00F
Reserved 0xE000 0000--0xE000 DFFF
GPIO Controller 0x400F F000
PMC 0x4007 D000
KBI1 0x4007 A000
KBI0 0x4007 9000
SPI1 0x4007 7000
SPI0 0x4007 6000
ACMP1 0x4007 4000
ACMP0 0x4007 3000
UART2 0x4006 C000
UART1 0x4006 B000
UART0 0x4006 A000
I2C 0x4006 6000
OSC 0x4006 5000
ICS 0x4006 4000
WDOG 0x4005 2000
PORT Controller 0x4004 9000
SIM 0x4004 8000
RTC 0x4003 D000
ADC 0x4003 B000
TIMER2 0x4003 A000
TIMER1 0x4003 9000
TIMER0 0x4003 8000
PIT 0x4003 7000
CRC 0x4003 2000
IRQ 0x4003 1000
Flash Memory 0x4002 0000
GPIO Controller 0x4000 F000

■ 1.4 嵌套向量中断控制器(NVIC)配置

■ 1.4.1 NVIC 简介

嵌套向量中断控制器(NVIC)集成可重定位的向量表，支持许多外部中断、一个不可屏蔽中断(NMI)和优先级。NVIC 用等效系统和简化编程性能代替影子寄存器。NVIC 包括特定中断响应函数的地址，该地址通过指令端口提取，允许并行寄存器堆叠与查找，前 16 个入口分配至 ARM 内部源，其余入口则映射至 MCU 定义的中断。关于更详细的 NVIC 信息请参阅 ARM 公司相关文档。

■ 1.4.2 中断优先级

该 NV32F100x 系列 MCU 支持 4 个中断优先级，在 NVIC 中，IPR 寄存器中的各个源都包含 2 位，例如：IPR0。如下所示：

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	IRQ3								IRQ2								IRQ1								IRQ0							
写																																

■ 1.4.3 不可屏蔽中断 NMI

对 NVIC 的不可屏蔽中断请求由外部 NMI信号控制。NMI信号进行多路复用所在的引脚必须配置NMI功能，以生成不可屏蔽中断请求。

注：在不使用此功能时，建议布板时 PB4 外接 4.7K 上拉电阻

■ 1.4.4 中断通道分配

下表中定义了中断向量分配。

- 向量编号 – 中断处于响应状态时，存储在堆栈上的值。
- IRQ 编号 – 非内核中断源编号，即向量编号减 16。

IRQ 编号用于 ARM 的 NVIC 文档中。

表 1-2 中断向量分配

向量	IRQ	NVIC IPR 编号	中断源	说明	地址
ARM 内核系统处理程序向量					
0	–	–	ARM 内核	初始堆栈指针	0x0000_0000
1	–	–	ARM 内核	初始程序计数	0x0000_0004
2	–	–	ARM 内核	不可屏蔽中断(NMI)	0x0000_0008
3	–	–	ARM 内核	硬故障	0x0000_000C
4	–	–	–	–	0x0000_0010
5	–	–	–	–	0x0000_0014
6	–	–	–	–	0x0000_0018
7	–	–	–	–	0x0000_001C

8	–	–	–	–	0x0000_0020
9	–	–	–	–	0x0000_0024
10	–	–	–	–	0x0000_0028
11	–	–	ARM 内核	管理程序调用(SVCall)	0x0000_002C
12	–	–	–	–	0x0000_0030
13	–	–	–	–	0x0000_0034
14	–	–	ARM 内核	可挂起的系统服务请求	0x0000_0038
15	–	–	ARM 内核	系统节拍定时器(SysTick)	0x0000_003C
非内核向量					
16	0	0	–	–	0x0000_0040
17	1	0	–	–	0x0000_0044
18	2	0	–	–	0x0000_0048
19	3	0	–	–	0x0000_004C
20	4	1	–	–	0x0000_0050
21	5	1	–	–	0x0000_0054
22	6	1	PMC	低电压警告	0x0000_0058
23	7	1	IRQ	外部中断	0x0000_005C
24	8	2	I2C	所有源的单个中断向量	0x0000_0060
25	9	2	–	–	0x0000_0064
26	10	2	SPI0	所有源的单个中断向量	0x0000_0068
27	11	2	SPI1	所有源的单个中断向量	0x0000_006C
28	12	3	UART0	状态和错误	0x0000_0070
29	13	3	UART1	状态和错误	0x0000_0074
30	14	3	UART2	状态和错误	0x0000_0078
31	15	3	ADC0	ADC 转换完成中断	0x0000_007C
32	16	4	ACMP0	模拟比较器 0 中断	0x0000_0080
33	17	4	ETimer0	所有源的单个中断向量	0x0000_0084
34	18	4	ETimer1	所有源的单个中断向量	0x0000_0088
35	19	4	ETimer2	所有源的单个中断向量	0x0000_008C
36	20	5	RTC	RTC 溢出中断	0x0000_0090
37	21	5	ACMP1	模拟比较器 1 中断	0x0000_0094
38	22	5	PIT_CH0	PIT CH0 溢出中断	0x0000_0098
39	23	5	PIT_CH1	PIT CH1 溢出中断	0x0000_009C
40	24	6	KBI0(32bit)	键盘中断 0（32 位）	0x0000_00A0
41	25	6	KBI1(32bit)	键盘中断 1（32 位）	0x0000_00A4
42	26	6	–	–	0x0000_00A8
43	27	6	ICS	时钟失锁	0x0000_00AC

44	28	7	WDOG	看门狗超时中断	0x0000_00B0
45	29	7	-	-	0x0000_00B4
46	30	7	-	-	0x0000_00B8
47	31	7	-	-	0x0000_00BC

■ 1.4.5 配置中断向量

假设您需要配置 SPI0 中断；下表是中断优先级中 SPI0 行的信息。

表 1-3. SPI0 中断向量分配

向量	IRQ	NVIC IPR 编号	中断源	说明	地址
26	10	2	SPI0	所有源的单个中断向量	0x0000_0068

您将用于配置中断的 NVIC 寄存器为：NVICIPR2

- 要确定这些特定寄存器中特定 IRQ 的字段位置：
  NVICIPR2 字段起始位置= 8 *（IRQ mod 4）+ 6 = 22。
- 由于 NVICIPR 字段为 2 位宽（4 个优先级），NVICIPR2 字段范围为位 22－23。因此，字段位置 NVICIPR2[23:22]用于配置 SPI0 中断。

■ 1.5 异步唤醒中断控制器(AWIC)

异步唤醒中断控制器(AWIC)的主要功能是检测停止模式下的异步唤醒事件，并向时钟控制逻辑发送信号以恢复系统时钟。时钟重启后，NVIC 观察处于挂起状态的中断，并执行正常中断或事件处理。

表 1-4 AWIC 停止唤醒源

唤醒源	说明
可用系统复位	LPO 是其时钟源时的$\overline{\text{RESET}}$引脚
低电压警告	电源管理控制器
IRQ	IRQ 引脚
引脚中断	KBI-任何使能的引脚中断都能唤醒系统
ADC	ADC 在使用内部时钟源时可在停止模式下运行
ACMP	正常中断
I2C	地址匹配唤醒
SPI	SPI 从机模式中断
UART	UART_RX 引脚处的 UART 有效边沿检测
RTC	警告中断
不可屏蔽中断	$\overline{\text{NMI}}$中断

■ 1.6 复位引导

■ 1.6.1 简介

该 MCU 支持下列复位源：

表 1-5 复位源

复位源	说明
POR 复位	● 上电复位(POR)
系统复位	● 外部引脚复位(PIN) ● 低压检测(LVD) ● 看门狗定时器(WDOG) ● ICS 时钟丢失(LOC)复位 ● 停止模式应答错误 （SACKERR) ● 软件复位(SW) ● 死锁复位(LOCKUP) ● MDM DAP 系统复位

每个系统复位源在系统复位状态和 ID 寄存器(SIM_SRSID)中都有相应的状态位。MCU 可在功能模式下退出和复位，而 CPU 在功能模式下可能正在执行代码（默认）或处于调试暂停状态。MCU 有多种引导选项可进行配置。

■ 1.6.2 复位

本节讨论基本复位机制和复位源。某些能导致复位的模块可以通过配置引起中断来代替复位。更多信息，请参考独立的外设章节。

■ 1.6.3 上电复位

MCU 初次上电或电源电压下降到低于上电复位电压(VPOR)时，POR 电路将导致 POR 复位条件。

随着电源电压上升，低压检测电路（LVD）会将 MCU 保持在复位状态，直到电源电压上升高于 LVD 低阈值(VLVDL),MCU 才可以正常启动。系统复位状态和 ID 寄存器 (SIM_SRSID)的 POR 和 LVD 字段 SIM_SRSID[POR]和 SIM_SRSID[LVD]在 POR 之后置位。

■ 1.6.4 系统复位源

MCU 提供一种从已知的初始条件开始处理复位的方法。系统复位开始时,片上稳压器处于完全稳压状态,系统时钟来源于内部基准时钟。处理器退出复位状态后，执行下列操作：

- 从向量表偏移 0 处读取 SP(SP_main)初始值
- 从向量表偏移 4 处读取程序计数器(PC)初始值
- 链路寄存器(LR)设置为 0xFFFF_FFFF

片上外设模块禁用且非模拟 I/O 引脚初始配置为禁用 （SWD_DIO/SWD_CLK、$\overline{\text{NMI}}$和$\overline{\text{RESET}}$引脚可根据系统选项寄存器(SIM_SOPT)的设置在系统复位后使能的情况除外。具有模拟功能的引脚在复位后默认为模拟功能。

● 1.6.4.1 外部引脚复位(RESET)

该引脚具有内部上拉电阻，$\overline{\text{RESET}}$ 的电平变为有效值可将器件从任意模式唤醒。POR 复位后，PA5 默认功能为$\overline{\text{RESET}}$。必须对 SIM_SOPT[RSTPE]进行编程以使能其他功能。该字段清零后，此引脚可用作 PA5 或其他替代功能。

● 1.6.4.2 复位引脚滤波器

$\overline{\text{RESET}}$/IRQ 引脚滤波器支持用 1kHz LPO 时钟和总线时钟进行滤波。它可用作简单的低通滤波器，从而过滤由$\overline{\text{RESET}}$/IRQ 引脚产生的任何毛刺。

毛刺宽度阈值可根据端口滤波寄存器(PORT_IOFLT)任意调节，设置端口滤波寄存器(PORT_IOFLT)范围在 1~4096 BUSCLK（或 1~128 LPOCLK）之间。通过该寄存器配置的毛刺滤波器可替代板载外部模拟滤波器，并且能极大地提高 EMC 性能。设置端口滤波寄存器(PORT_IOFLT)可配置整个端口的滤波器。

● 1.6.4.3 低压检测(LVD)

该器件集成了一个可避免低压条件的系统，以便在电源电压发生变化期间保护存储器内容和控制 MCU 系统状态。该系统由上电复位(POR)电路和LVD电路组成,LVD具有用户可选的跳闸电压,可以是高电平(VLVDH)或低电平(VLVDL)。

PMC_SPMSC1[LVDE]置位且 PMC_SPMSC2[LVDV]选定跳闸电压后，LVD 电路使能。除非 PMC_SPMSC1[LVDSE]置位或处于串行线调试(SWD)模式，否则 LVD 一进入停止模式就会被禁用。若 PMC_SPMSC1[LVDSE]和 PMC_SPMSC1[LVDE]均置位，在 LVD 使能的情况下，停止模式下的电流消耗将更高。

● 1.6.4.4 看门狗定时器

看门狗(WDOG)通过预期的，定期的软件通讯来对系统操作进行监控。此通讯一般被称为刷新看门狗（喂狗）。如果此周期性刷新没有出现，则看门狗将发送系统复位。WDOG 复位会使 SIM_SRSID[WDOG]置位。

● 1.6.4.5 ICS 时钟丢失(LOC)

该芯片上的 ICS 支持带复位功能的外部参考时钟监控器。

在 FBE 或 FEE 模式下，如将 1 写入 ICS_C4[CME]，则时钟监控器使能。若外部参考频率下降低于某个特定频率（比如：floc_high 或 floc_low，具体取决于 OSC_CR[RANGE]），则 MCU 将复位。SIM_SRSID[LOC]将置位以指出错误。

在 FBILP 模式下，FLL 不启用，因此即便将 1 写入 C4[CME]，外部参考时钟监视器也不会运行。外部基准时钟监控器将 FLL 用作内部参考时钟。FLL 必须在 ICS_C4[CME]置位前能够工作。

● 1.6.4.6 停止模式应答错误(SACKERR)

在内核试图进入停止模式时，在 1025 个 1kHz LPO 时钟周期内，仍然有外设没有响应停止模式，则会产生该复位。

如果发生错误条件，模块对进入停止模式可能不会作出应答。该错误可能是由模块的外部时钟输入故障引起的。

● 1.6.4.7 软件复位(SW)

通过设置 NVIC 应用中断及复位控制寄存器中的 SYSRESETREQ 字段可以强制使设备产生软件复位。（有关寄存器字段的完整说明，尤其是 VECTKEY 字段的要求，请参见 ARM 的 NVIC 文档）。设置 SYSRESETREQ 可产生软件复位请求。此次复位对除调试模块以外的所有主要模块强制执行系统复位。

● 1.6.4.8 死锁复位(LOCKUP)

LOCKUP 可立即表明内核软件出现严重出错。这是由于激活处理器内置系统状态保护硬件后出现无法恢复的异常情况而导致内核锁定的结果。

LOCKUP 条件可导致系统复位，也可导致 SIM_SRSID[LOCKUP]置位。

● 1.6.4.9 MDM-AP 系统复位请求

设置 MDM-AP 控制寄存器中的“系统复位请求”字段产生系统复位。这是通过 SWD 接口进行复位的主要方法。在清除此位前系统一直保持复位状态。

设置 MDM-AP 控制寄存器中的“内核保持复位”字段，将内核保持在复位状态，而芯片其他功能退出复

位状态。

■ 1.6.5 MCU 复位

MCU 产生各种复位以复位不同模块。

● 1.6.5.1 仅 POR 复位

“仅 POR 复位”仅由 POR 复位源产生。它复位 PMC 和 RTC。

“仅 POR 复位”还会产生所有其他的复位类型。

● 1.6.5.2 芯片 POR

“芯片 POR”由 POR 和 LVD 复位源产生。它复位“复位引脚滤波器”寄存器和部分 SIM 和 ICS 的寄存器。

“芯片 POR”还能导致芯片复位（包括“早期芯片复位”）产生。

● 1.6.5.3 早期芯片复位

“早期芯片复位”在所有复位源上都产生。它仅复位 Flash 存储器模块和 ARM 平台。它在 Flash 存储器开始初始化之前撤除（“早于”芯片复位撤除）。

● 1.6.5.4 芯片复位

“芯片复位”在所有复位源上都产生，并且仅在 RESET 引脚撤除后才撤除。它复位剩余的模块（未由其他复位类型复位的模块）。

■ 1.6.6 启动引导

本小节介绍引导顺序，包括来源和选项。

引导过程自动加载某些配置信息，如存在工厂编程 Flash 位置中的时钟调整值。

● 1.6.6.1 引导源

CM0+内核加入对可配置向量表偏移寄存器(VTOR)的支持，可重新定位中断向量表。该器件支持从内部 Flash 和 RAM 引导。（VTOR：指的是 ARMv6-M 架构参考手册中的向量表偏移寄存器。）

该器件支持从复位向量位于地址 0x0（初始 SP_main）、0x4（初始 PC）的内部 Flash 引导，以及从中断向量表重定位至 RAM 的 RAM 引导。

● 1.6.6.2 引导序列

上电时，片上稳压器将系统保持在 POR 状态，直到输入电源电压高于 POR 阈值。系统持续处于该静态，直到内部稳压电源达到 LVD 所决定的安全操作电压。复位控制器逻辑随后执行以下序列以退出复位。

1. 内部逻辑基础上保持系统复位，$\overline{\text{RESET}}$ 引脚被驱动至输出低电平（约 4.2μs），并且 ICS 按默认配置使能。

2. 释放$\overline{\text{RESET}}$引脚。如果$\overline{\text{RESET}}$引脚的电平继续为有效值（表示$\overline{\text{RESET}}$引脚上升时间缓慢或外部驱动低电平），则系统继续保持在复位状态。一旦检测到$\overline{\text{RESET}}$引脚处于高电平，内核时钟使能且系统解除复位状态。

3. NVM 开始内部初始化。Flash 控制器解除复位状态并开始执行初始化操作，而内核在 Flash 初始化完成之前依然保持停止状态。

4. Flash 初始化完成(16μs)后，内核设置堆栈、程序计数器(PC)和链路寄存器(LR)。

处理器从向量表偏移 0 读取 SP (SP_main)初始值。内核从向量表偏移 4 读取 PC 初始值。LR 设 0xFFFF_FFFF，CPU 在 PC 位置处开始执行。

后续系统复位遵循相同的复位流程。

■ 1.7 时钟分配

■ 1.7.1 简介

本章介绍该器件的时钟架构以及时钟概述，并提供术语部分。

Cortex M0+位于同步内核平台上，可对处理器和总线主机、Flash 和外设时钟进行独立配置。ICS 模块将用于主系统时钟生成。ICS 模块选择时钟源（内部基准、外部晶体或外部时钟信号）产生系统时钟源。

■ 1.7.2 编程模型

系统时钟源的选择和多路复用通过 ICS 模块控制和配置。该系统的时钟分频器设置和模块时钟选通通过 SIM 模块配置。有关寄存器和位的详细说明，请参见各章节相关部分。

■ 1.7.3 器件时钟连接示意图

该器件包含下列片上时钟源：

- 内部时钟源(ICS)模块：向外设提供总线时钟和其他参考时钟的主时钟源发生器
- 系统振荡器(OSC)模块：向内部时钟源(ICS)、实时计数器时钟模块(RTC)和其他 MCU 子系统提供参考时钟的系统振荡器
- 低功耗振荡器(LPO)模块：向 RTC 和看门狗(WDOG)提供 1kHz 基准时钟的片上低功耗振荡器

下图显示的是来自 ICS 模块和 OSC 模块的时钟如何分配给微控制器的其他功能单元，微控制器中的某些模块具有可选时钟输入。下列系统振荡器 、ICS 和 SIM 模块的寄存器控制多路复用器、分频器和时钟脉冲门，如下图所示：

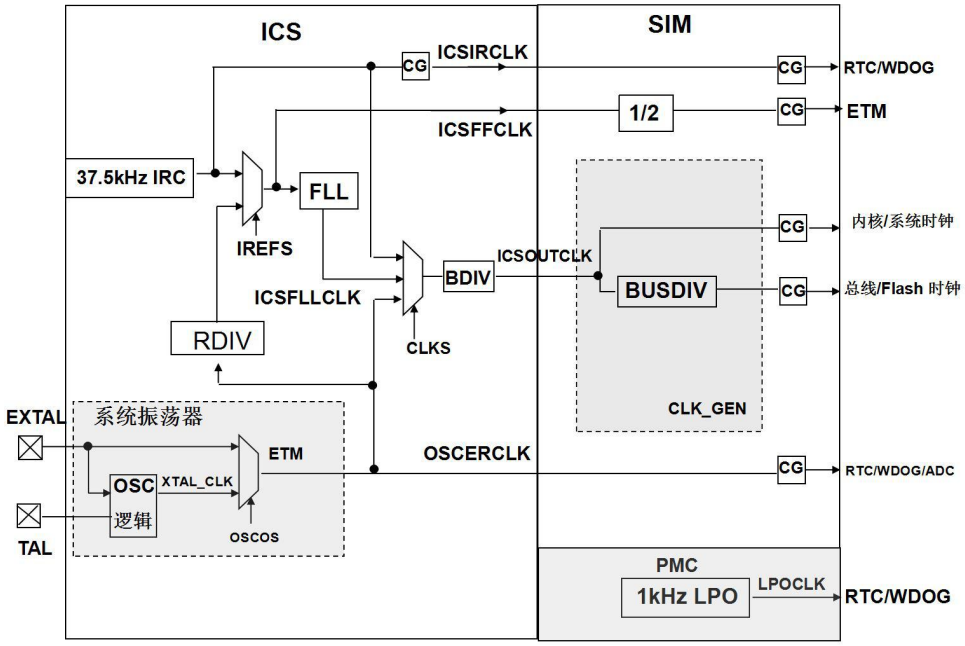


图 1-3　时钟示意图

表 1-6 控制多路复用器、分频器和时钟脉冲门的寄存器

	OSC	ICS	SIM
多路复用器	OSC_CR	ICS_C1	SIM_SOPT
分频器	-	ICS_C2	SIM_CLKDIV
时钟脉冲门	OSC_CR	ICS_C1	SIM_SCGC

■ 1.7.4 时钟定义

下表说明上图的时钟：

表 1-7 时钟定义

时钟名称	说明
内核时钟	由 DIV1 分频的 ICSOUTCLK，为 ARM Cortex-M0+内核计时，它是 CPU HCLK。
平台时钟	由 DIV1 分频的 ICSOUTCLK，为交叉开关和 NVIC 计时，它是自由运行的 FCLK。
系统时钟	由 DIV1 分频的 ICSOUTCLK，直接为总线主机计时。
总线时钟	由 DIV2 分频的系统时钟，为总线从机和外设计时。
Flash 时钟	由 DIV2 分频的系统时钟，为 Flash 存储器模块计时，它与该器件中的总线时钟相同。
定时器时钟	由 DIV3 分频的 ICSOUTCLK，为 ETM 计时。
调试时钟	调试逻辑时钟。就该器件而言，它来源于平台时钟。
SWD 时钟	DAP 接口时钟。SWD 时钟通常由外部调试器驱动，且完全与内核时钟和平台时钟异步。
ICSIRCLK	内部 32kHz IRC 基准时钟的 ICS 输出，ICSIRCLK 可作为 RTC 或 WDOG 模块的时钟源。
ICSOUTCLK	IRC、ICSFLLCLK 或 ICS 外部基准时钟的 ICS 输出，是内核、系统、总线和 Flash 时钟的时钟源。
ICSFLLCLK	FLL 的输出，FLL 将频率锁定为 1280 乘以内部或外部基准频率。
ICSFFCLK	固定频率时钟的 ICS 输出。ICSFFCLK 可作为 ETM 模块的时钟源。ICSFFCLK 的频率由 ICS 的设置决定。
OSCCLK	内部振荡器的系统振荡器输出，或由 EXTAL 直接提供时钟源。用作 ICS 外部基准时钟。
OSCERCLK	由 OSCCLK 提供时钟源的系统振荡器输出，可作为 RTC、WDOG 或 ADC 模块的时钟源。
LPOCLK	PMC 1kHz 输出，LPOCLK 可作为 RTC 或 WDOG 模块的时钟源。

● 1.7.4.1 器件时钟汇总

下表提供有关片上时钟的更多信息。

表 1-8 时钟汇总

时钟名称	运行模式频率	时钟源	时钟禁用条件
内核时钟	最高 48MHz	ICSOUTCLK 时钟分频器	在等待和停止模式下
平台时钟	最高 48MHz	ICSOUTCLK 时钟分频器	在停止模式下
系统时钟	最高 48MHz	ICSOUTCLK 时钟分频器	在停止模式下
定时器时钟	最高 48MHz	ICSOUTCLK 时钟分频器	在停止模式下
总线时钟	最高 48MHz	ICSOUTCLK 时钟分频器	在停止模式下
调试时钟	最高 48MHz	来自平台时钟	调试未使能
SWD 时钟	最高 48MHz	SWD_CLK 引脚	外部时钟输入，因此将不会被禁用
Flash 时钟	最高 48MHz	ICSOUTCLK 时钟分频器	在停止模式下
内部基准时钟 (ICSIRCLK)	31.25-39.0625kH IRC	IRC	ICS_C1[IRCLKEN]=0 或在停止模式下且 ICS_C1[IREFSTEN]=0
外部基准时钟 (OSCERCLK)	DC 高达 24MHz(旁路) 31.25-39.0625kHz 或 4-24MHz(晶体)	系统 OSC	OSC_CR[OSCEN]=0,或在停止模式下且 OSC_CR[OSCSTEN]=0
FLL 输出时钟 (ICSFLLCLK)	40-50MHz 建议不超过 48M	系统 OSC/IRC	在停止模式下/FLL 未使能

ICS 固定频率时钟（ICSFFCLK）	31.25-39.0625kHz	系统 OSC/IRC	在停止模式下
LPOCLK	1kHz	PMC	在所有电源模式下都可用

● 1.7.4.2 时钟分配

下图是时钟分配示意图：

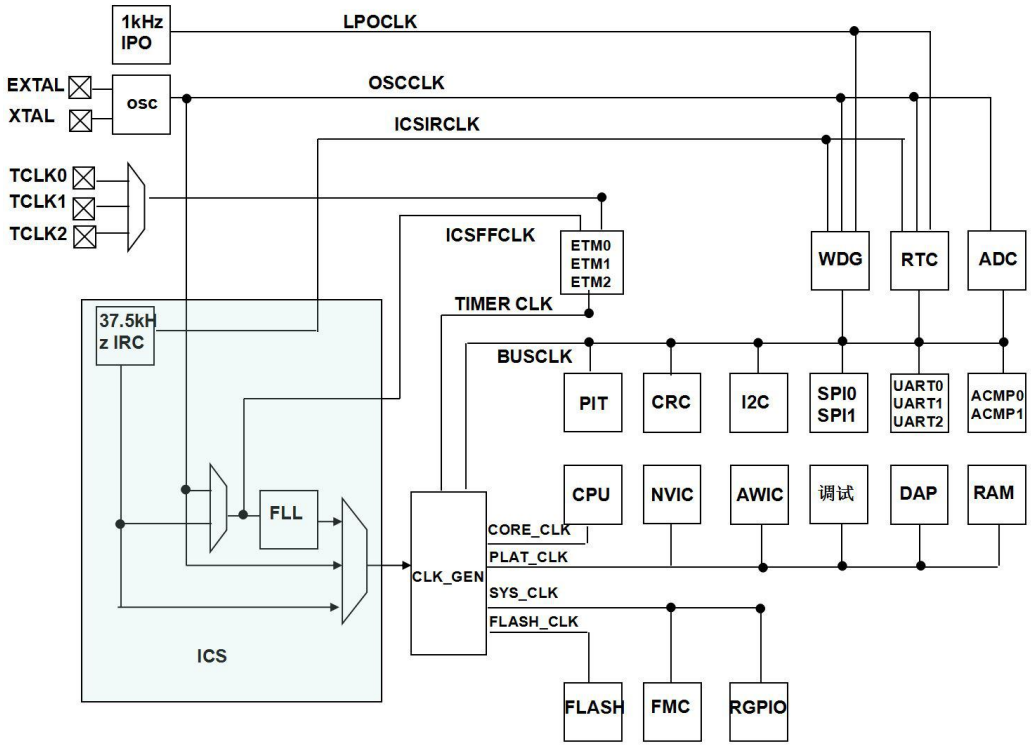


图 1-4 时钟分配示意图

■ 1.7.5 内部时钟源

该器件的内部时钟源如下：（使用 FEI 建议不超过 48M，内部 RC 出厂可校准）

- 片上 RC 振荡器，范围为 31.25 – 39.0625 kHz，作为 FLL 输入的基准
- 片上内部 1kHz 振荡器，作为 RTC 和 WDOG 的低频低功耗源，符合特定用例要求

下表显示的是该器件的频率可用性：

表 1-9 基于内部参考的可行的的 ICS 总线频率

基准		ICSOUTCLK
FEI(高范围)	BDIV=0	40MHz~50MHz[1]
	BDIV=1	20MHz~25MHz
	BDIV=2	10MHz~12.5MHz
	BDIV=4	5MHz~6.25MHz
	BDIV=8	2.5MHz~3.125MHz
	BDIV=16	1.25MHz~1.5625MHz
	BDIV=32	625kHz~781.25kHz
	BDIV=64	312.5kHz~390.625kHz
	BDIV=128	156.25kHz~195.3125kHz

1. 仔细配置 SIM_CLKDIV 和 BDIV，确保避免任何高于 48MHz 的时钟频率。

■ 1.7.6 外部时钟源

该器件支持下列两个外部时钟源：

- 外部方波输入时钟，DC-24MHz
- 外部晶振或谐振器
- 小范围：31.25 – 39.0625 kHz
- 大范围：4 – 24 MHz

注：外部方波输入时钟仅在 OSC 模块设置为外部时钟模式下使用（旁路）。凭借外部方波时钟源，用户可使用 FLL 被禁用的 FBE 模式来实现更低功耗或精密时钟源。

下表显示的是该器件采用 OSC 时钟作为时钟源时的可用频率。OSC 外部时钟模式未显示。

表 1-10 可用的 OSC 频率

ICS 配置	外部基准	RDIV
FBE	31.25kHz~39.0625kHz	–
	4MHz~24MHz	–
FEE[1]	31.25kHz~39.0625kHz	RDIV=1
	62.5kHz~78.125kHz	RDIV=2
	125kHz~56.25kHz	RDIV=4
	250kHz~312.5kHz	RDIV=8
	500kHz~625kHz	RDIV=16
	1MHz~1.25MHz	RDIV=32
	2MHz~2.5MHz	RDIV=64
	4MHz~5MHz	RDIV=128
	8MHz~10MHz	RDIV=256
	16MHz~20MHz	RDIV=512

1. 在 FEE 模式下，FLL 输出频率 = OSC/RDIV*1280。仔细选择 OSC 和 RDIV，确保将 FLL 输出频率保持在限值内。

■ 1.7.7 时钟选通

每个模块的时钟都可通过配置系统时钟选通控制寄存器(SIM_SCGC)单独地实现开/关的选通。进行模块初始化之前，配置系统时钟选通控制寄存器(SIM_SCGC)中相应位以使能时钟。关闭时钟前，务必禁用该模块。

任何通过总线访问时钟关闭的外设都会产生错误而终止。

■ 1.7.8 模块时钟

下表汇总与各模块相关的时钟。

表 1-11 模块时钟

模块	总线接口时钟	内部时钟	I/O 接口时钟
内核模块			
ARM Cortex-M0+ 内核	平台时钟	内核时钟	–
NVIC	平台时钟	–	–

DAP	平台时钟	-	SWD_CLK
系统模块			
端口控制	总线时钟	-	-
交叉开关	平台时钟	-	-
外设联接器	系统时钟	总线时钟	-
PMC、SIM	总线时钟	LPOCLK	-
MCM	平台时钟	-	-
CRC	总线时钟	-	-
看门狗定时器	总线时钟	总线时钟 LPOCLK/ICSIRCLK /OSCERCLK	-
时钟			
ICS	总线时钟	ICSOUTCLK/ICSFLLCLK/ICSIRCLK/OSCERCLK	-
OSC	总线时钟	OSCERCLK	-
存储器和存储器接口			
Flash 控制器	系统时钟	-	-
Flash 存储器	Flash 时钟	-	-
SRAM	平台时钟	-	-
模拟模块			
ADC	总线时钟	总线时钟 OSCERCLK/ADACK	-
ACMP0	总线时钟	-	-
ACMP1	总线时钟	-	-
定时器			
PIT	总线时钟	-	-
ETM0	定时器时钟	定时器时钟 ICSFFCLK	TCLK0/1/2
ETM1	定时器时钟	定时器时钟 ICSFFCLK	TCLK0/1/2
ETM2	定时器时钟	定时器时钟 ICSFFCLK	TCLK0/1/2
RTC	总线时钟	总线时钟 LPOCLK、ICSIRCLK、OSCERCLK	RTC_CLKOUT
通信接口			
SPI0	总线时钟	-	SPI0_SCK
SPI1	总线时钟	-	SPI1_SCK
I2C	总线时钟	-	I2C_SCL
UART0	总线时钟	-	-
UART1	总线时钟	-	-
UART2	总线时钟	-	-
人机接口			
GPIO	系统时钟	-	-
KBI0	总线时钟	-	-
KBI1	总线时钟	-	-

■ 1.8 电源管理

■ 1.8.1 简介

本章介绍多种芯片电源模式以及各模块在这些模式下的功能。

■ 1.8.2 功耗模式

电源管理控制器(PMC)为用户提供多种功耗选项。支持各种工作模式，来允许用户针对所需的功能优化功耗。该器件支持运行、待机和停止模式，无论是不同的功耗水平还是功能要求，客户都能轻松使用。所有模式下都能保持 I/O 状态。

- 运行模式—CPU 时钟可在全速状态下运行，内部电源处于全稳压状态。
- 待机模式—关断 CPU 以降低功耗；此时系统时钟和总线时钟依然运行，保持完全稳压状态。
- 停止模式—可选 LVD 使能，稳压器处于待机状态。

三种工作模式为：运行、待机和停止。WFI 指令激活芯片的待机和停止模式。

表 1-12 芯片功率模式

功耗模式	说明	内核模式	一般恢复方法
正常运行	允许芯片工作在最高性能下。复位后的默认状态；片上稳压器开启。	运行	-
通过 WFI 实现正常待机	允许外设工作，同时让内核处于睡眠模式，可降低功耗。NVIC 依然监测中断；继续提供外设时钟。	睡眠	中断
通过 WFI 实现正常停止	将芯片置于静态。最低功耗模式保持全部寄存器值，并保留 LVD 保护功能(可选)。NVIC 禁用；使用 AWIC 从中断唤醒；外设时钟停止。	深度睡眠	中断

芯片通过 WFI 指令进入等待和停止模式。处理器通过中断退出低功耗模式。

注：WFE 指令可能会对进入低功耗模式产生副作用，但这并非其预期用途。有关 WFE 指令的更多信息，请参见 ARM 文档。

下表说明了该芯片处于各低功耗模式时每个模块的功能。表中显示了标准特性以及某些例外情况。

表 1-13 低功耗模式下的模块操作

模块	运行	等待	停止
内核模块			
CPU	开启	待机	待机
NVIC	开启	开启	待机
系统模块			
PMC	完全稳压	完全稳压	宽松稳压
WDOG	开启	开启	可选开启
LVD	开启	开启	可选开启
CRC	开启	开启	待机
时钟模块			
ICS	开启	开启	可选开启
OSC	开启	开启	可选开启
LPO	开启	开启	始终开启

存储器			
FLASH	开启	开启	待机
SRAM	开启	待机[1]	待机
定时器模块			
ETM	开启	开启	待机
PIT	开启	开启	待机
RTC	开启	开启	可选开启
模拟			
ADC	开启	开启	可选开启
ACMP	开启	开启	可选开启
通信接口			
UART	开启	开启	待机[2]
SPI	开启	开启	待机[3]
IIC	开启	开启	待机[4]
人机接口			
KBI	开启	开启	待机[5]
IRQ	开启	开启	待机[5]
I/O	开启	开启	状态保持

1. SRAM 使能信号保持在低电平时，禁用内部时钟信号并屏蔽地址和数据输入；芯片中的 RAM 时钟可在等待模式下处于激活状态。
2. 支持停止模式下的边沿唤醒
3. 支持停止模式下的从机模式接收和唤醒
4. 支持停止模式下的地址匹配唤醒
5. 支持停止模式下的引脚中断唤醒

■ 1.9 调试

■ 1.9.1 简介

该器件基于 ARM CoreSight 架构进行调试，配置为在引脚分配和其他可用资源所允许的最大限度内提供最大的灵活性。

外部调试器通过调试接口访问寄存器和存储器。调试接口包括基本运行/停止控制加上 2 个断点和 2 个观察点。

该器件仅支持一个调试器接口，即串行调试(SWD)。

■ 1.9.2 调试端口引脚说明

上电复位(POR)后，调试端口引脚默认为其为 SWD 功能。

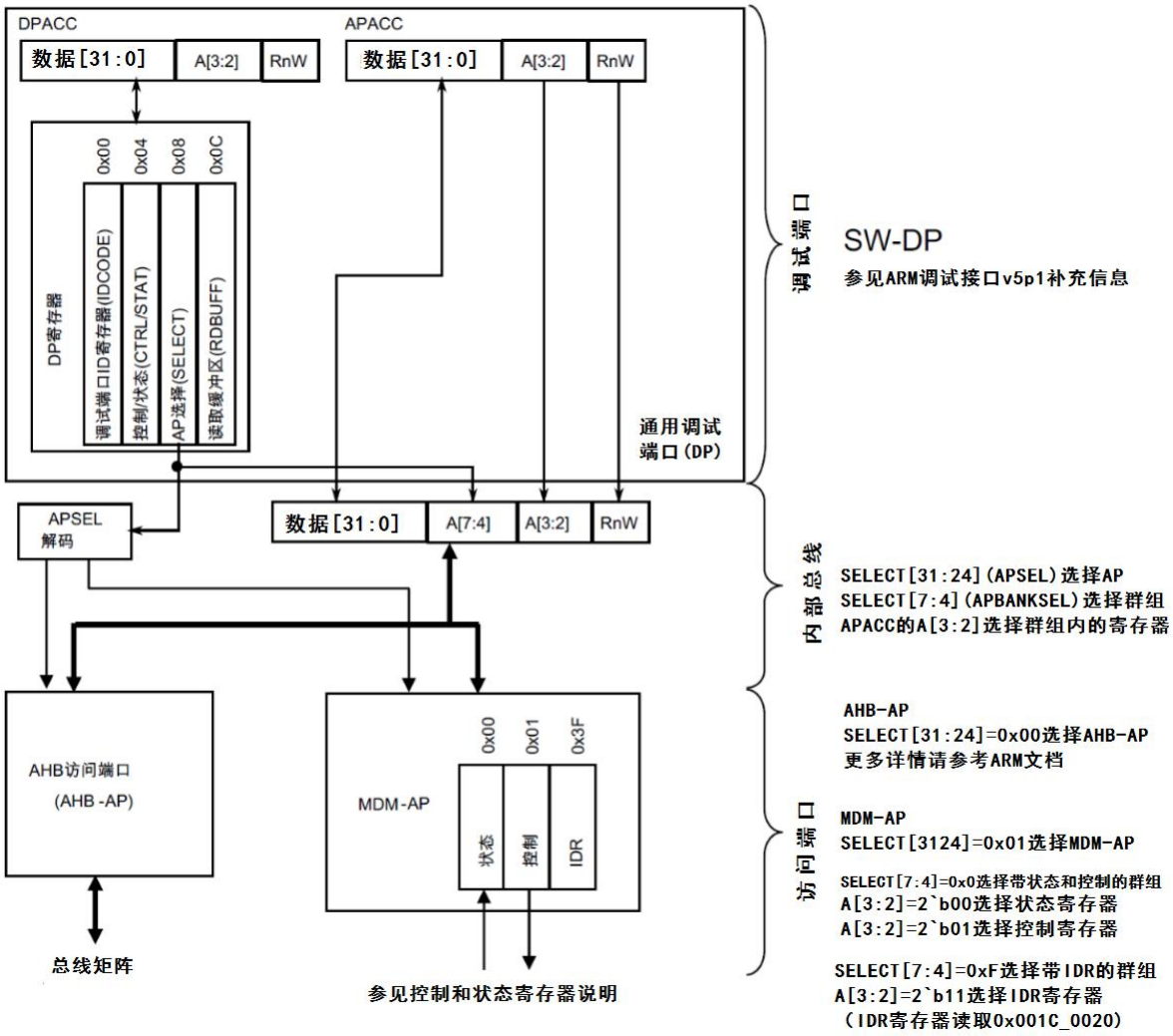


图 1-5 MDM AP 寻址

表 1-14 串行线调试引脚说明

引脚名称	类型	说明
SWD_CLK	输入	串行线时钟。该引脚在串行线调试模式下作为调试逻辑的时钟。
SWD_DIO	输入/输出	串行线调试数据输入/输出。外部调试工具通过 SWD_DIO 引脚进行通信和器件控制。该引脚在内部上拉。

注：该器件不支持片上下拉；SWD_CLK 引脚仅支持由 PAPE0 控制的上拉，完全支持 SWD 协议需要外部下拉电阻。

■ 1.9.3 SWD 状态和控制寄存器

通过 ARM 调试访问端口(DAP)，调试器可访问以寄存器形式在 DAP 总线上实施的状态和控制元件。这类寄存器针对低功耗模式恢复和典型运行一控制场景提供其他控制和状态。凭借状态寄存器位，调试器无需通过交叉开关发起总线操作即能获得内核的已更新状态，从而确保调试期间更少的干扰。

该器件上包括了其他调试模块(MDM)，其中包含 DAP 控制和状态寄存器。值得注意的是，这些 DAP 控制

和状态寄存器并非在系统存储器映射中进行地址映射，并且只能通过使用 SWD 的调试访问端口进行访问。调试访问端口的 MDM-AP 的可访问寄存器如下表所示：

表 1-15 MDM-AP 寄存器汇总

地址	寄存器	说明
0x0100_0000	状态寄存器	参见 MDM-AP 状态寄存器
0x0100_0004	控制寄存器	参见 MDM-AP 控制寄存器
0x0100_00FC	IDR 寄存器	只读识别寄存器，始终以 0x001C_0020 读取

● 1.9.3.1 MDM-AP 状态寄存器

表 1-16 MDM-AP 状态寄存器分配

位	名称	说明
0	Flash 整体擦除应答	Flash 整体擦除应答字段在 POR 复位后清零。通过对 MDM AP 控制寄存器的“正在执行 Flash 整体擦除”字段执行写操作而发出整体擦除命令时该字段也会清零。在 Flash 控制逻辑开始了整体擦除操作后，将设置该 flash 整体擦除应答。
1	Flash 就绪	表示 Flash 存储器已完成初始化，调试器可配置，即使调试器继续保持系统在复位状态。 0 Flash 正在初始化 1 Flash 就绪
2	系统安全性	表示安全状态。处于安全状态时，调试器无法访问系统总线或任何存储器映射外设。该字段表示器件锁定且无法进行系统总线访问。 注：该位在 Flash 就绪位置位前无效。 0 器件未处于安全状态。 1 器件处于安全状态。
3	系统复位	表示系统复位状态。 0 器件未处于复位状态。 1 器件处于复位状态。
4	保留	-
5-15	保留供将来使用	读取始终为 0。
16	内核暂停	表示内核已进入调试暂停模式 0 内核未暂停 1 内核已暂停
17	内核 SLEEPDEEP	SLEEPDEEP=1 表示内核已进入停止模式。
18	内核 SLEEPING	SLEEPING=1 表示内核已进入等待模式。
19-31	保留供将来使用	始终读取 0。

● 1.9.3.2 MDM-AP 控制寄存器

表 1-17 MDM-AP 控制寄存器分配

位	名称	安全[1]	说明
0	正在执行 Flash 整体擦除	是	置位以导致整体擦除。在整体擦除完成后由硬件清除。
1	调试禁用	否	置位禁用调试。清零允许调试操作。置位时，它覆盖 DHCSR 中的 C_DEBUGEN 字段[2]并强制禁用调试逻辑。
2	调试请求	否	置位强制暂停内核。 若内核处于停止或等待模式，该字段可用于唤醒内核并转变到暂停状态。
3	系统复位请求	是	置位强制进行系统复位。系统在该字段清零前都保持在复位状态。该位置位后，RESET 引脚不反映系统复位的状态并且不保持低电平。
4	内核保持	否	配置字段，可控制系统复位序列结束系统内核操作。 0 正常运行-在系统复位序列结束时，解除内核与系统的其他部分的复位。 1 挂起运行-在复位序列结束后将内核保持在复位状态。一旦系统进入挂起状态，该控制位清零可立即解除内核的复位状态并且 CPU 开始工作。
5-31	保留	否	-

1. 安全模式下可以用的命令　2. DHCSR 值的是 ARMv6-M 架构参考手册中的 “调试暂停控制和状态寄存器”。

■ 1.9.4 调试复位

调试系统将收到以下复位源：

- 系统 POR 复位

相反，调试系统能够使用以下机制产生系统复位：

- DAP 控制寄存器中的系统复位，允许调试器保持系统在复位状态。
- 将 1 写入 NVIC 应用中断和复位控制寄存器中的 SYSRESETREQ 字段
- DAP 控制寄存器中的系统复位，允许调试器保持内核在复位状态。

■ 1.9.5 低功耗模式下的调试

在调试模式保持为静态或掉电状态的低功耗模式下，调试器无法在低功耗模式时间段内采集任何调试数据。

- 如果调试器保持为静态，那么只要退出低功耗模式且系统返回到某个具有主动调试的状态，调试器端口就会恢复所有功能。
- 如果调试器逻辑处于掉电状态，那么调试器在恢复是就会复位并且在退出低功耗模式时必须重新配置。

主动调试将防止芯片进入低功耗模式。如果芯片已处于低功耗模式，那么来自 MDM-AP 控制寄存器的调试请求将从低功耗模式唤醒该芯片。

■ 1.9.6 调试和安全性

使能 Flash 安全性后，调试端口的功能将受限，以防对安全数据的不当利用。在安全状态下，调试器仍然可以访问状态寄存器，并且可以确定该器件当前的安全状态。如果器件处于安全状态，那么调试器仅可以执行整体擦除操作。

第 2 章 系统模块

■ 2.1 简介

本章对基于 ARM Cortex-M0+内核的 NV32F100x 系列 MCU 系统相关的模块做了详细说明，其中包括 SIM 系统集成、PMC 电源管理、MCM、CRC 循环冗余校验、BOS 位操作等模块内容。

■ 2.2 SIM 系统集成模块

■ 2.2.1 简介

系统集成模块(SIM)提供系统控制和芯片配置寄存器。

■ 2.2.2 特性

SIM 模块的特性如下：

- 复位状态和器件 ID 信息
- 系统互连配置和特殊引脚的启用
- 引脚重映射控制
- 系统时钟选通控制和时钟分频

■ 2.2.3 存储器映射和寄存器说明

SIM 模块包含许多字段，可用来为不同的模块时钟选择时钟源和分频器。

表 2-1 SIM 存储器映射

绝对地址 (16 进制)	寄存器名称	偏移量	位宽	权限	复位值
4004_8000	系统复位状态和 ID 寄存器 (SIM_SRSID)	00h	32	R	参见章节
4004_8004	系统选项寄存器 (SIM_SOPT)	04h	32	R/W	参见章节
4004_8008	引脚选择寄存器 (SIM_PINSEL)	08h	32	R/W	00h
4004_800C	系统时钟选通控制寄存器 (SIM_SCGC)	0Ch	32	R/W	00h
4004_8010	通用唯一标识符低位寄存器 (SIM_UUIDL)	10h	32	R	未定义
4004_8014	通用唯一标识符中低位寄存器 (SIM_UUIDML)	14h	32	R	未定义
4004_8018	通用唯一标识符中高位寄存器 (SIM_UUIDMH)	18h	32	R	未定义
4004_801C	时钟分频器寄存器 (SIM_BUSDIV)	1Ch	32	R/W	参见章节

● 2.2.3.1 系统复位状态和 ID 寄存器 (SIM_SRSID)

地址: 4004_8000h 基准 + 0h 偏移 = 4004_8000h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	FAMID				SUBFAMID				RevID				PINID			
写																
复位	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1
LVD	0	0	0	0	0	0	1	0	0	0	0	0	0	1	1	1

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0		SACKERR	0	MDMAP	SW	LOCKUP	0	POR	PIN	WDOG	0		LOC	LVD	0
写																
复位	0	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0
LVD	0	0	0	0	0	0	0	0	U*	0	0	0	0	0	1	0

注: **U*** = LVD 复位值不确认，除 POR 上电复位为 1，其他复位该位为 0。

表 2-2 SIM_SRSID 字段描述

位	描述
31-28 FAMID	NV32F100x 系列 ID 0000 NV32F100x 系列。 其他 保留。
27-24 SUBFAMID	NV32F100x 子系列 ID 0100 A 子系列 0110 B 子系列 其他 保留
23-20 RevID	器件版本号
19-16 PINID	器件引脚 ID 0000 8 引脚 0001 16 引脚 0010 20 引脚（带晶振版和无晶振版） 0011 24 引脚（需要订做） 0100 32 引脚 0101 44 引脚 0110 48 引脚 0111 64 引脚 其他 保留。 注:除 32，48，64 外的其他管脚需要预订，关闭不用管脚以降低功耗，默认为 64 引脚。
15-14 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

13 SACKERR	停止模式应答错误复位 指示在尝试进入停止模式时，因 IIC 未能在约一秒时间内对此作出应答而引起复位。 0 复位不是因为外设未能对尝试进入停止模式作出应答而引起。 1 复位是因为外设未能对尝试进入停止模式作出应答而引起。
12 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
11 MDMAP	MDM-AP 系统复位请求 指示 MDM-AP 控制寄存器中系统复位请求字段的主机调试器系统设置已引起复位。 0 复位不是由系统复位请求位的主机调试器系统设置引起。 1 复位是由系统复位请求位的主机调试器系统设置引起。
10 SW	软件指示 ARM 内核中应用程序中断和复位控制寄存器的 SYSRESETREQ 位的软件设置已引起复位。 0 复位不是由 SYSRESETREQ 位的软件设置引起。 1 复位是由 SYSRESETREQ 位的软件设置引起。
9 LOCKUP	内核锁定 指示 LOCKUP 事件的 ARM 内核指示已引起复位。 0 复位不是由内核 LOCKUP 事件引起。 1 复位是由内核 LOCKUP 事件引起。
8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 POR	上电复位 通过上电检测逻辑引起复位。当内部供电电压斜升时，低压复位(LVR)状态字段也在此时置位，指示已因内部供电电压低于 LVR 阈值而发生复位。 注: 该位在 POR 时复位为 1，在 LVR 时复位为不定值，在任何其他情况下复位为 0。 0 复位不是由 POR 引起。 1 POR 引起复位。
6 PIN	外部复位引脚 通过外部复位引脚的有效低电平引起复位。 0 复位不是由外部复位引脚引起。 1 复位来源于外部复位引脚。
5 WDOG	看门狗(WDOG) 通过 WDOG 定时器超时引起复位。该复位源可通过设置 WDOG_CS1[EN] = 0 加以阻断。 0 复位不是由 WDOG 超时引起。 1 复位是由 WDOG 超时引起。
4-3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2 LOC	内部时钟源模块复位 通过 ICS 模块复位引起复位。 0 复位不是由 ICS 模块引起。 1 复位是由 ICS 模块引起。
1 LVD	低压检测 如果 PMC_SPMSC1[LVDRE]在运行模式下置位或 PMC_SPMSC1[LVDRE]和 PMC_SPMSC1[LVDSE]

	在停止模式下均置位，并且供电电压下降低于 LVD 跳变电压，那么将发生 LVD 复位。该字段也会因 POR 而置位。 注：该字段在 POR 和 LVR 时复位为 1，在其他复位时复位为 0。 0 复位不是由 LVD 跳变或 POR 引起。 1 复位是由 LVD 跳变或 POR 引起。
0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

● 2.2.3.2 系统选项寄存器(SIM_SOPT)

地址：4004_8000h 基准 + 4h 偏移 = 4004_8004h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	DELAY								DLYACT	0		FLASHDP	CLKOE	BUSREF		
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
POR/LVD	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	TXDME	0	RXDFE	RXDCE	ACIC	RTCC	ADHWT		0				SWDE	RSTPE	NMIE	0
写		ETMSYNC														
复位	0	0	0	0	0	0	0	0	0	0	0	0	1	u*	u*	0
POR/LVD	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0

注：1.RSTPE 和 NMIE 在每次复位时只能写入一次。 2.u* = 不受复位影响。

表 2-3 SIM_SOPT 字段描述

位	描述
31 – 24 DELAY	ETM2 触发延迟 指定将 1 写入 ADHWT 时从 ETM2 初始或匹配触发到 ADC 硬件触发的延迟。该 8 位模数值允许 0 到 255 的延迟，具体取决于 BUSREF 时钟设置。这是一个一次性计数器，当触发到达时开始计数，当计数器值达到所定义的模数值时停止计数。
23 DLYACT	ETM2 触发延迟有效 该只读字段指定有关 ETM2 初始或匹配延迟是否有效的状态。该字段在 ETM2 触发到达且延迟计数器正在计数时置位，否则，该字段清零。 0 延迟无效。 1 延迟有效。

22－21 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
20 FLASHDP	Flash Deep Sleep 使能控制 0 Flash 在 STOP 模式下不进入 Deep Sleep 1 Flash 在 STOP 模式下进入 Deep Sleep
19 CLKOE	总线时钟输出使能 0 总线时钟输出在 PH2 上禁用。 1 总线时钟输出在 PH2 上使能。
18－16 BUSREF	总线时钟输出选择 通过可选预分频器使能总线时钟输出。 000 总线 001 总线 2 分频 010 总线 4 分频 011 总线 8 分频 100 总线 16 分频 101 总线 32 分频 110 总线 64 分频 111 总线 128 分频
15 TXDME	UART0_TX 调制选择 使能由 ETM0 通道 0 调制的 UART0_TX 输出。 0 UART0_TX 输出直接连接到引出线。 1 UART0_TX 输出在映射到引出线前由 ETM0 通道 0 调制。
14 ETMSYNC	ETM2 同步选择 将 1 写入该字段时生成 ETM2 模块的 PWM 同步触发。 0 未触发任何同步。 1 生成 ETM2 模块的 PWM 同步触发。
13 RXDFE	UART0 RxD 滤波器选择 使能 UART0 RxD 输入由 ACMP 滤波。该功能使能时，任何具有 ACMP 输入标记的信号都可被视作 UART0。 0 RXD0 输入信号直接连接到 UART0 模块。 1 RXD0 输入信号由 ACMP0 滤波，然后注入 UART0。
12 RXDCE	UART0_RX 捕捉选择 使能 UART0_RX 由 ETM0 通道 1 捕捉。 0 UART0_RX 输入信号仅连接到 UART0 模块。 1 UART0_RX 输入信号连接到 UART0 模块和 ETM0 通道 1。
11 ACIC	模拟比较器至输入捕捉使能 将 ACMP0 的输出连接到 ETM1 输入通道 0。 0 ACMP0 输出未连接到 ETM1 输入通道 0。 1 ACMP0 输出连接到 ETM1 输入通道 0。
10 RTCC	实时计数器捕捉 允许实时计数器(RTC)溢出由 ETM1 通道 1 捕捉。 0 RTC 溢出未连接到 ETM1 输入通道 1。 1 RTC 溢出连接到 ETM1 输入通道 1。

9－8 ADHWT	ADC 硬件触发源 选择 ADC 硬件触发源，所有触发源都是在上升沿开始 ADC 转换。 00 用作 ADC 硬件触发的 RTC 溢出 01 用作 ADC 硬件触发的 PIT 通道溢出 10 带 8 位可编程计数器延迟的 ETM2 初始触发 11 带 8 位可编程计数器延迟的 ETM2 匹配触发
7-4 保留	此字段为保留字段。
3 SWDE	单线调试端口引脚使能 使能 PA4/ACMP0_OUT/SWD_DIO 引脚用作 SWD_DIO，使能 PC4/RTC_CLKOUT/ETM1_CH0/ACMP0_IN2/SWD_CLK 引脚用作 SWD_CLK。清零时，两个引脚用作 PA4 和 PC4。该引脚在任何 MCU 复位之后默认用作 SWD_DIO 和 SWD_CLK。 0 PA4/ACMP0_OUT/SWD_DIO 作为 PA4 或 ACMP0_OUT 功能，PC4/RTC_CLKOUT/ETM1_CH0/ACMP0_IN2/SWD_CLK 用作 PC4、RTC_CLKOUT、ETM1_CH0、OR ACMP0_IN2 功能。 1 PA4/ACMP0_OUT/SWD_DIO 用作 SWD_DIO 功能，PC4/RTC_CLKOUT/ETM1_CH0/ACMP0_IN2/SWD_CLK 用作 SWD_CLK 功能。
2 RSTPE	RESET 引脚使能 在任何复位后都可对该一次性写入字段进行写操作。RSTPE 置位时，PA5/IRQ/TCLK0/RESET 引脚用作RESET。清零时，该引脚用作备用功能之一。该引脚在 MCU POR 之后默认用作RESET。其他复位不会影响该字段。RSTPE 置位时，RESET上的内部上拉器件使能。 0 PA5/IRQ/TCLK0/RESET引脚用作 PA5/IRQ/TCLK0。 1 PA5/IRQ/TCLK0/RESET引脚用作RESET。
1 NMIE	NMI引脚使能（建议 NMI 脚即 PB4 在未做功能时，外接 4.7K 上拉电阻） 在任何复位后都可对该一次性写入字段进行写操作。NMIE 置位时，PB4/ETM2_CH4/SPI0_MISO/ACMP1_IN2/NMI引脚用作NMI。清零时，该引脚用作其他功能之一。该引脚在 MCU POR 之后默认用作NMI。其他复位不会影响该位。NMIE 置位时，NMI上的内部上拉器件使能。 0 PB4/ETM2_CH4/SPI0_MISO/ACMP1_IN2/NMI 引脚用作 PB4、ETM2_CH4、SPI0_MISO 或 ACMP1_IN2。 1 PB4/ETM2_CH4/SPI0_MISO/ACMP1_IN2/NMI引脚用作NMI。
0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

● 2.2.3.3 引脚选择寄存器(SIM_PINSEL)

地址：4004_8000h 基准 + 8h 偏移 = 4004_8008h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	ETM2PS3	ETM2PS2	ETM2PS1	ETM2PS0	ETM1PS1	ETM1PS0	ETM0PS1	ETM0PS0	UART0PS	SPI0PS	I2C0PS	RTCPS	0			
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 2-4 SIM_PINSEL 字段描述

位	描述
31-16	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
15 ETM2PS3	ETM2_CH3 端口引脚选择 选择 ETM2[3]通道输出分配。 0 ETM2_CH3 通道输出映射到 PC3 上。 1 ETM2_CH3 通道输出映射到 PD1 上。
14 ETM2PS2	ETM2_CH2 端口引脚选择 选择 ETM2[2]通道输出分配。 0 ETM2_CH2]通道输出映射到 PC2 上。 1 ETM2_CH2 通道输出映射到 PD0 上。
13 ETM2PS1	ETM2_CH1 端口引脚选择 选择 ETM2_CH1 通道输出分配。 0 ETM2_CH1 通道输出映射到 PC1 上。 1 ETM2_CH1 通道输出映射到 PH1 上。
12 ETM2PS0	ETM2_CH0 端口引脚选择 选择 ETM2_CH0 通道输出分配。 0 ETM2_CH0 通道输出映射到 PC0 上。 1 ETM2_CH0 通道输出映射到 PH0 上。
11 ETM1PS1	ETM1_CH1 端口引脚选择 选择 ETM1_CH1 通道引脚分配。 0 ETM1_CH1 通道映射到 PC5 上。 1 ETM1_CH1 通道映射到 PE7 上。
10 ETM1PS0	ETM1_CH0 端口引脚选择 选择 ETM1_CH0 通道引脚分配。 0 ETM1_CH0 通道映射到 PC4 上。 1 ETM1_CH0 通道映射到 PH2 上。
9 ETM0PS1	ETM0_CH1 端口引脚选择 选择 ETM0_CH1 通道引脚分配。 0 ETM0_CH1 通道映射到 PA1 上。 1 ETM0_CH1 通道映射到 PB3 上。
8 ETM0PS0	ETM0_CH0 端口引脚选择 选择 ETM0_CH0 通道引脚分配。 0 ETM0_CH0 通道映射到 PA0 上。 1 ETM0_CH0 通道映射到 PB2 上。

7 UART0PS	UART0 引脚选择 选择 UART0 引脚分配。 0 UART0_RX 和 UART0_TX 映射到 PB0 和 PB1 上。 1 UART0_RX 和 UART0_TX 映射到 PA2 和 PA3 上。
6 SPI0PS	SPI0 引脚选择 选择 SPI0 引脚分配。 0 SPI0_SCK、SPI0_MOSI、SPI0_MISO 和 SPI0_PCS 映射到 PB2、PB3、PB4 和 PB5 上。 1 SPI0_SCK、SPI0_MOSI、SPI0_MISO 和 SPI0_PCS 映射到 PE0、PE1、PE2 和 PE3 上。
5 I2C0PS	I2C0 端口引脚选择 选择 I2C0 端口引脚。 0 I2C0_SCL 和 I2C0_SDA 分别映射到 PA3 和 PA2 上。 1 I2C0_SCL 和 I2C0_SDA 分别映射到 PB7 和 PB6 上。
4 RTCPS	RTC0 引脚选择 选择 RTC0 端口引脚。 0 RTC0 映射到 PC4 上。 1 RTC0 映射到 PC5 上。
3-0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

● 2.2.3.4 系统时钟选通控制寄存器 (SIM_SCGC)

地址: 4004_8000h 基准 + Ch 偏移 = 4004_800Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	ACMP1	ACMP0	ADC	0	IRQ	0	KBI1	KBI0	0	UART2	UART1	UART0	SPI1	SPI0	I2C	0
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0	0	SWD	FLASH	0	CRC	0		ETM2	ETM1	ETM0	0			PIT	RTC
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 2-5 SIM_SCGC 字段描述

位	描述
31 ACMP1	ACMP1 时钟选通控制 控制 ACMP1 模块的时钟选通。 0 ACMP1 模块的总线时钟禁用。 1 ACMP1 模块的总线时钟使能。

30 ACMP0	ACMP0 时钟选通控制 控制 ACMP0 模块的时钟选通。 0 ACMP0 模块的总线时钟禁用。 1 ACMP0 模块的总线时钟使能。
29 ADC	ADC 时钟选通控制 控制 ADC 模块的时钟选通。 0 ADC 模块的总线时钟禁用。 1 ADC 模块的总线时钟使能。
28 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
27 IRQ	IRQ 时钟选通控制 控制 IRQ 模块的时钟选通。 0 IRQ 模块的总线时钟禁用。 1 IRQ 模块的总线时钟使能。
26 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
25 KBI1	KBI1 时钟选通控制 控制 KBI1 模块的时钟选通。 0 KBI1 模块的总线时钟禁用。 1 KBI1 模块的总线时钟使能。
24 KBI0	KBI0 时钟选通控制 控制 KBI0 模块的时钟选通。 0 KBI0 模块的总线时钟禁用。 1 KBI0 模块的总线时钟使能。
23 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
22 UART2	UART2 时钟选通控制 控制 UART2 模块的时钟选通。 0 UART2 模块的总线时钟禁用。 1 UART2 模块的总线时钟使能。
21 UART1	UART1 时钟选通控制 控制 UART1 模块的时钟选通。 0 UART1 模块的总线时钟禁用。 1 UART1 模块的总线时钟使能。
20 UART0	UART0 时钟选通控制 控制 UART0 模块的时钟选通。 0 UART0 模块的总线时钟禁用。 1 UART0 模块的总线时钟使能。

19 SPI1	SPI1 时钟选通控制 控制 SPI1 模块的时钟选通。 0 SPI1 模块的总线时钟禁用。 1 SPI1 模块的总线时钟使能。
18 SPI0	SPI0 时钟选通控制 控制 SPI0 模块的时钟选通。 0 SPI0 模块的总线时钟禁用。 1 SPI0 模块的总线时钟使能。
17 I2C	I2C 时钟选通控制 控制 I2C 模块的时钟选通。 0 I2C 模块的总线时钟禁用。 1 I2C 模块的总线时钟使能。
16-14 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
13 SWD	SWD（单线调试器）时钟选通控制 控制 SWD 模块的时钟选通。 0 SWD 模块的总线时钟禁用。 1 SWD 模块的总线时钟使能。
12 FLASH	Flash 时钟选通控制 控制 Flash 模块的时钟选通。 0 Flash 模块的总线时钟禁用。 1 Flash 模块的总线时钟使能。
11 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
10 CRC	CRC 时钟选通控制 控制 CRC 模块的时钟选通。 0 CRC 模块的总线时钟禁用。 1 CRC 模块的总线时钟使能。
9－8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 ETM2	ETM2 时钟选通控制 控制 ETM2 模块的时钟选通。 0 ETM2 模块的总线时钟禁用。 1 ETM2 模块的总线时钟使能。
6 ETM1	ETM1 时钟选通控制 控制 ETM1 模块的时钟选通。 0 ETM1 模块的总线时钟禁用。 1 ETM1 模块的总线时钟使能。
5 ETM0	ETM0 时钟选通控制 控制 ETM0 模块的时钟选通。 0 ETM0 模块的总线时钟禁用。

	1 ETMO 模块的总线时钟使能。
4 - 2 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
1 PIT	PIT 时钟选通控制 控制 PIT 模块的时钟选通。 0 PIT 模块的总线时钟禁用。 1 PIT 模块的总线时钟使能。
0 RTC	RTC 时钟选通控制 控制 RTC 模块的时钟选通。 0 RTC 模块的总线时钟禁用。 1 RTC 模块的总线时钟使能。

● 2.2.3.5 通用唯一标识符低位寄存器 (SIM_UUIDL)

只读 SIM_UUIDL 寄存器包含一系列数字，用以识别该系列中的不同器件。

地址: 4004_8000h 基准 + 10h 偏移 = 4004_8010h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	ID[31:0]																															
复位	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

注: * = 复位时未定义。

表 2-6 SIM_UUIDL 字段描述

字段	描述	复位值
ID[31:0]	通用唯一标识符	*

● 2.2.3.6 通用唯一标识符中位寄存器 (SIM_UUIDM)

只读 SIM_UUIDM 寄存器包含一组数字，用于标识系列中的唯一器件。

地址: 4004_8000h 基准 + 14h 偏移 = 4004_8014h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	ID[63:32]																															
复位	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

注:* = 复位时未定义。

表 2-7 SIM_UUIDM 字段描述

字段	描述	复位值
ID[63:32]	通用唯一标识符	*

● 2.2.3.7 通用唯一标识符高位寄存器 (SIM_UUIDH)

只读 SIM_UUIDH 寄存器包含一系列数字，用以识别该系列中的不同器件。

地址: 4004_8000h 基准 + 18h 偏移 = 4004_8018h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	ID[95:64]																															
复位	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

注:* = 复位时未定义。

表 2-8 SIM_UUIDH 字段描述

字段	描述	复位值
ID[95:64]	通用唯一标识符	*

■ 2.2.3.8 总线时钟分频器寄存器 (SIM_BUSDIV)

此寄存器为总线时钟设置分频值。如果用户想使用48MHz核心时钟，需先设置ICS_C2[BDIV]=0x000，分频为1倍。

地址: 4004_8000h 基准 + 1Ch 偏移 = 4004_801Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0															BUSDIV
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	u*
POR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

注: u = 不受复位影响。

表 2-9 SIM_BUSDIV 字段描述

位	描述
31 – 1 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
0 BUSDIV	总线时钟分频器 为总线时钟设置分频器值 0 总线时钟频率与 ICSOUTCLK 相同。 1 总线时钟频率是 ICSOUTCLK/2。

■ 2.3 PMC 电源管理模块

■ 2.3.1 简介

本章介绍芯片低功耗模式下各模块的功能以及电源管理控制器模块的操作。

■ 2.3.2 低电压检测(LVD)系统

该设备包含一个防御低电压影响的系统，以便在供电电压变动时保护存储器中的内容并控制 MCU 系统的状态。此系统包含一个上电复位(POR)电路，和具有可供用户选择跳闸电压（高电压(VLVDH)或低电压(VLVDL)）的 LVD 电路。SPMSC1[LVDE]置位且 SPMSC2[LVDV]选择跳闸电压后，即可启用 LVD 电路。除非 SPMSC1[LVDSE]置位，否则在进入停止模式时 LVD 会被禁用。如果 SPMSC1[LVDSE]和 SPMSC1[LVDE]均已置位，则启用 LVD 系统时，电流消耗量将高于停止模式下的电流消耗量。

下图呈现的是低电压检测(LVD)系统的结构框图。

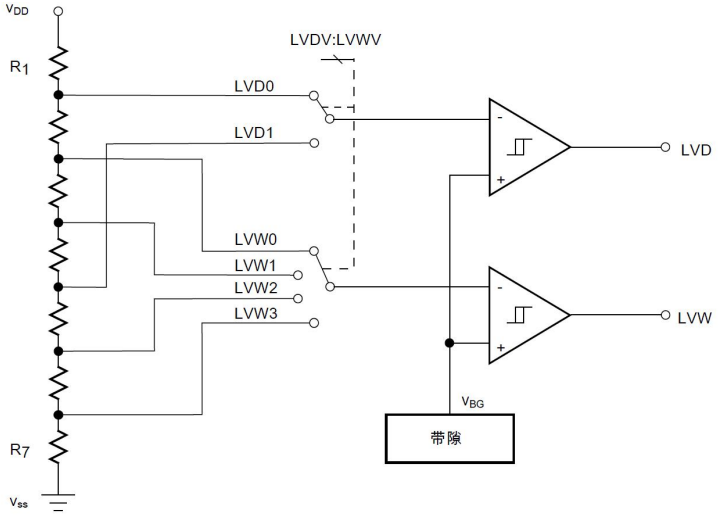


图 2-1. 低电压检测(LVD)结构框图

● 2.3.2.1 上电复位(POR)操作

MCU 初次上电或电源电压下降低于 V_{POR} 电平时，POR 电路将导致复位。随着电源电压上升，LVD 电路会将芯片保持在复位状态，直到电源上升高于 VLVDL 电平。SIM_SRSID[POR]和 SIM_SRSID[LVD]在 POR 之后都将置位。

● 2.3.2.2 LVD 复位操作

通过将 SPMSC1[LVDRE]置 1，LVD 可配置为一检测到低压条件就生成复位信号。LVD 复位后，LVD 系统会将 MCU 保持在复位状态，直到电源电压上升高于 LVDV 电平。SIM_SRSID[LVD]在 LVD 复位或 POR 之后置位。

● 2.3.2.3 停止模式下的 LVD 使能

电源电压下降低于 LVD 电压时，LVD 系统生成复位信号。CPU 执行“停止”指令时，如果 LVD 在停止（SPMSC1[LVDE]和 SPMSC1[LVDSE]均置 1）模式下启用，那么电压调节器在停止模式下保持有效状态。

● 2.3.2.4 低压警报(LVW)

LVD 系统具有低压警报标志，可指示电源电压正在接近 LVW 电压。检测到低压条件且 LVD 电路已配置中

断操作（SPMSC1[LVDE]置位，SPMSC1[LVWIE]置位）时，SPMSC1[LVWF]将置位，并且将发生 LVW 中断。每次 LVDV 配置时，LVW 都有四个用户可选跳闸电压。跳闸电压根据 SPMSC2[LVWV]进行选择。

■ 2.3.3 带隙基准源

该器件集成了一个连接至 ADC 通道的片上带隙基准源(≈1.1 V)。即使工作电压正在下降，带隙基准电压也不会下降低于全工作电压。该基准电压用作精确测量的理想基准电压。

■ 2.3.4 存储器映射和寄存器说明

表 2-10 PMC 存储器映射

绝对地址（16 进制）	寄存器名称	偏移量	位宽	权限	复位值
4007_D000h	系统电源管理状态和控制 1 寄存器(PMC_SPMSC1)	00h	8	R/W	1Ch
4007_D001h	系统电源管理状态和控制 2 寄存器(PMC_SPMSC2)	01h	8	R/W	00h

● 2.3.4.1 系统电源管理状态和控制寄存器 1(PMC_SPMSC1)

该高端页面寄存器包含状态位和控制位，支持低压检测功能，可使带隙基准电压供 ADC 模块使用。即使所需设置与复位设置相同，该寄存器也必须在用户复位初始化程序运行期间写入数据以设置所需的控制。

地址：4007_D000h 基准 + 0h 偏移 = 4007_D000h

位	7	6	5	4	3	2	1	0
读	LVWF	0	LVWIE	LVDRE	LVDSE	LVDE	0	BGBE
写		LVWACK						
复位	0	0	0	1	1	1	0	0

表 2-11 PMC_SPMSC1 字段描述

字段	描述
7 LVWF	低压警报标志 指示低压警报状态。 注：如果 Vsupply 转换低于跳变点或在复位位后 Vsupply 已低于 VLVW，那么 LVWF 将置位。上电复位后，LVWF 可能为 1；因此，为了使用 LVW 中断功能，在使能 LVWIE 前，LVWF 必须首先通过写 LVWACK 清零。 0 不存在低压警报。 1 存在或曾经存在低压警报。
6 LVWACK	低压警报应答 若 LVWF=1，则已发生低压条件。为了应答该低压警报，向 LVWACK 写入 1，如果低压警报不再发生，那么就会自动清零 LVWF。
5 LVWIE	低压警报中断使能 使能 LVWF 的硬件中断请求。 0 禁用硬件中断(使用轮询)。 1 LVWF=1 时请求硬件中断。

4 LVDRE	低压检测复位使能 该一次性写入位可使能 LVD 事件以产生硬件复位（假设 LVDE = 1） 注：复位后，该字段仅可可写入一次，其他写操作会被忽略。 若 LVDRE = 0，则使用用 LVW 监控状态，因为没有标志变为有效。 0 LVD 事件不产生硬件复位 1 发生使能低压检测事件时强制进行 MCU 复位
3 LVDSE	低压检测停止使能 假设 LVDE = 1,则该读/写字段可确定低压检测功能在 MCU 处于停止模式时是否工作。 0 低压检测在停止模式期间禁用。 1 低压检测在停止模式期间使能。
2 LVDE	低压检测使能 该一次性写入位使能低压检测逻辑并认证该寄存器中其他字段的操作。 注：复位后，该字段仅可可写入一次，其他写操作会被忽略。 0 LVD 逻辑禁用。 1 LVD 逻辑使能。
1 保留	该字段为保留字段。 此只读字段为保留字段且值始终为 0。
0 BGBE	带隙缓冲区使能 使能某个内部缓冲区以便带隙基准电压源可供 ADC 模块在其某个内部通道或选作 ACMP 基准的带隙上使用。 0 带隙缓冲区禁用。 1 带隙缓冲区使能。

● 2.3.4.2 系统电源管理状态和控制寄存器 2(PMC_SPMSC2)

该寄存器用于报告低压警报功能的状态，以及配置 MCU 的停止模式特性。即使所需设置与复位设置相同，该寄存器也应当在用户复位初始化程序运行期间写入数据以设置所需的控制。

地址：4007_D000h 基准 + 1h 偏移 = 4007_D001h

位	7	6	5	4	3	2	1	0
读	0	LVDV	LVWV		0			
写								
复位	0	0	0	0	0	0	0	0

表 2-12 PMC_SPMSC2 字段描述

字段	描述
7 保留	该字段为保留字段。 此只读字段为保留字段且值始终为 0。
6 LVDV	低压检测电压选择 该一次性写入选择低压检测(LVD)跳变点设置。 0 选择低电平跳变点(VLVD=VLVDL)。 1 选择高电平跳变点(VLVD=VLVDH)。
5-4 LVWV	低压警报电压选择 选择低压警报(LVW)跳变点电压。更多详情请参见数据手册。

	00 选择低电平跳变点(VLVW=VLVW1)。 01 选择中间电平 1 跳变点(VLVW=VLVW2)。 10 选择中间电平 2 跳变点(VLVW=VLVW3)。 11 选择高电平跳变点(VLVW=VLVW4)。
3-0 保留	该字段为保留字段。 读取始终为 0。

■ 2.4 MCM 杂项管理模块

■ 2.4.1 简介

本章介绍芯片其他系统控制功能。

■ 2.4.2 特性

该模块包含下列特性：

- 程序可见的有关平台配置的信息。

■ 2.4.3 存储器映射和寄存器说明

本节介绍了使用字节地址的寄存器,这些寄存器仅可在监控器模式下写入。

表 2-13 MCM 存储器映射

绝对地址 (16 进制)	寄存器名称	偏移量	位宽	权限	复位值
F000_3008h	交叉开关(CRBS)从机配置(MCM_PLASC)	08h	16	R	0007h
F000_300Ah	交叉开关(CRBS)主机配置(MCM_PLAMC)	0Ah	16	R	0001h

● 2.4.3.1 交叉开关(CRBS)从机配置(MCM_PLASC)

PLASC 是一个 16 位只读寄存器，用于识别是否存在至设备交叉开关的总线从连接。

地址: F000_3000h 基准 + 8h 偏移 = F000_3008h

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0								ASC							
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 2-14 MCM_PLASC 字段描述

字段	描述
15-8 保留	该字段为保留字段。 只读字段读取值始终为 0.
7-0 ASC	ASC 字段中的每个位表示交叉开关的从输入端口是否存在对应连接。 0 不存在至 CRBS 输入端口 n 的总线从连接。

	1 存在至 CRBS 输入端口 n 的总线从连接。

● 2.4.3.2 交叉开关(CRBS)主机配置(MCM_PLAMC)

PLASC 是一个 16 位只读寄存器，用于识别是否存在至设备交叉开关的总线主连接。

地址: F000_3000h 基准 + Ah 偏移 = F000_300Ah

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0								AMC							
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 2-15 MCM_PLAMC 字段描述

字段	描述
15-8 保留	该字段为保留字段。 只读字段读取值始终为 0.
7-0 AMC	AMC 字段中的每个位表示交叉开关的主输入端口是否存在对应连接。 0 不存在至 CRBS 输入端口 n 的总线主连接。 1 存在至 CRBS 输入端口 n 的总线主连接。

■ 2.5 CRC 循环冗余校验模块

■ 2.5.1 简介

循环冗余校验(CRC)模块生成 16/32 位 CRC 码以便进行误差检测，CRC 模块提供实施 16 位或 32 位 CRC 标准所需的可编程多项式、WAS 和其他参数。对于 32 位数据，每次都计算该 16/32 位代码。

注:MCU 进去低功耗模式后，该模块时钟禁用，任何进行中的 CRC 计算都会停止。

■ 2.5.2 结构框图

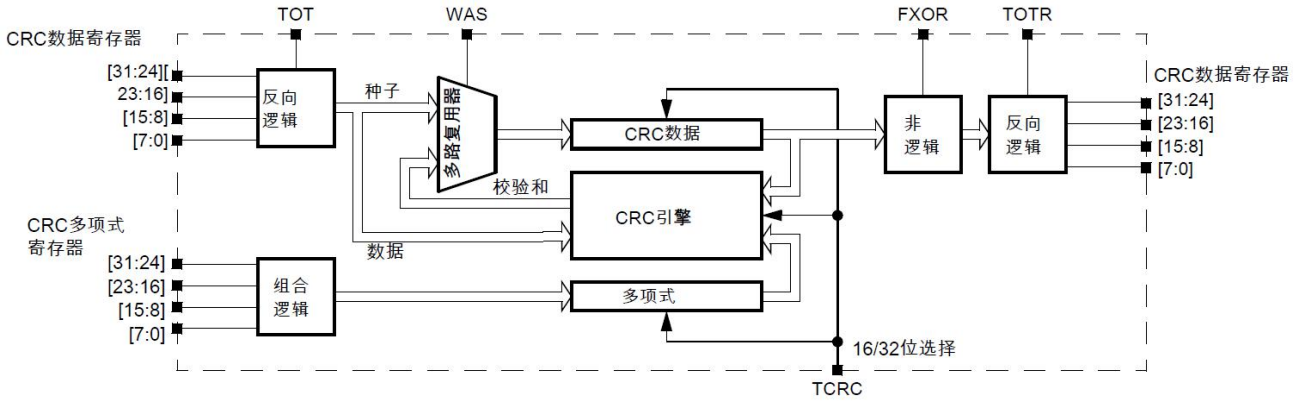


图 2-2 CRC 结构框图

■ 2.5.3 特性

CRC 模块包含下列特性：

- 使用 16 位或 32 位可编程移位寄存器的硬件 CRC 生成器电路

- 可编程初始种子值和多项式
- 逐位或逐字节转置输入数据或输出数据（CRC 结果）。某些 CRC 标准要求提供该选项。以 8 位读取操作访问 CRC 数据寄存器时，无法执行逐字节转置操作。这种情况下，用户软件必须执行逐字节转置操作。（转置操作建议在系统时钟<48M 的情况下使用）
- 提供最终 CRC 结果反转选项
- 32 位 CPU 寄存器编程接口

■ 2.5.4 存储器映射和寄存器说明

本节介绍了使用字节地址的寄存器,这些寄存器仅可在监控器模式下写入。

表 2-16 CRC 存储器映射

绝对地址 (16 进制)	寄存器名称	偏移量	位宽	权限	复位值
0x4003_2000	CRC 数据寄存器 (CRC_DATA)	00h	32	R/W	FFFF_FFFFh
0x4003_2004	CRC 多项式寄存器 (CRC_GPOLY)	04h	32	R/W	0000_1021h
0x4003_2008	CRC 控制寄存器 (CRC_CTRL)	08h	32	R/W	0000_0000h

● 2.5.4.1 CRC 数据寄存器(CRC_DATA)

CRC 数据寄存器包含种子、数据以及校验和的数值。如果 CTRL[WAS]置位，则对数据寄存器进行的任何写操作都被视为种子值。如果 CTRL[WAS]清零，则对数据寄存器进行的任何写操作都被视为用于一般 CRC 计算的数据。

在 16 位 CRC 模式下，不使用 HU 字段和 HL 字段来设定种子值，对这些字段进行读操作会返回不确定的值。在 32 位 CRC 模式下，所有字段都用于种子值的编程。

进行数据数值编程时，如果所有字节都是连续的，那么可一次写入 8 位、16 位或 32 位的数值；首先写入的是 MSB 数据数值。

写入所有数据数值后，可从该数据寄存器中读取 CRC 结果。在 16 位 CRC 模式下，LU 字段和 LL 字段提供 CRC 结果。在 32 位 CRC 模式下，所有字段均包含此结果。如果已配置 CRC 模块，那么随时对该寄存器进行读操作都会返回中间的 CRC 值。

地址: 4003_2000h 基准 + 0h 偏移 = 4003_2000h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	HU								HL								LU								LL							
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

表 2-17 CRC_DATA 字段描述

位	描述
31 - 24 HU	CRC 高字段高位字节 在 16 位 CRC 模式（CTRL[TCRC]为 0）下，该字段并不用于种子值的编程。在 32 位 CRC 模式（CTRL[TCRC]为 1）下，当 CTRL[WAS]为 1 时，写入该字段的值是种子值的一部分。当 CTRL[WAS]

	为 0 时，写入该字段的数据用于在 16 位 CRC 模式和 32 位 CRC 模式下生成 CRC 校验和。
23-16 HL	CRC 高字段低位字节 在 16 位 CRC 模式（CTRL[TCRC]为 0）下，该字段并不用于设定种子值。在 32 位 CRC 模式（CTRL[TCRC] 为1)下,当 CTRL[WAS]为 1 时,写入该字段的值是种子值的一部分。当 CTRL[WAS]为 0 时，写入该字段的数据用于在 16 位 CRC 模式和 32 位 CRC 模式下生成 CRC 校验和。
15 - 8 LU	CRC 低字段高位字节 当 CTRL[WAS]为 1 时，写入该字段的数值是种子值的一部分。当 CTRL[WAS]为 0 时，写入该字段的数据用于生成 CRC 校验和。
7-0 LL	CRC 低字段低位字节 当 CTRL[WAS]为 1 时，写入该字段的数值是种子值的一部分。当 CTRL[WAS]为 0 时，写入该字段的数据用于生成 CRC 校验和。

● 2.5.4.2 CRC 多项式寄存器(CRC_GPOLY)

该寄存器含有 CRC 计算所需的多项式值。HIGH 字段含有 CRC 多项式的高 16 位，仅在 32 位 CRC 模式下使用。在 16 位 CRC 模式下会忽略对 HIGH 字段的写操作。LOW 字段包含在 16 位 CRC 模式和 32 位 CRC 模式下都使用的 CRC 多项式的低 16 位。

地址: 4003_2000h 基准 + 4h 偏移 = 4003_2004h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	HIGH																LOW															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1

表 2-18 CRC_GPOLY 字段描述

位	描述
31 - 16 HIGH	多项式高半字 32 位 CRC 模式下可读写(CTRL[TCRC]为 1),该字段在 16 位 CRC 模式下不可写(CTRL[TCRC]为 0)。
15-0 LOW	多项式低半字 在 32 位 CRC 模式和 16 位 CRC 模式下都可读写。

● 2.5.4.3 CRC 控制存器(CRC_CTRL)

该寄存器控制 CRC 模块的配置和操作。开始进行新的 CRC 计算前，相应位必须置位。初始化新的 CRC 计算的方法是：使 CTRL[WAS]的电平变为有效，然后将种子写入 CRC 数据寄存器。

地址: 4003_2000h 基准 + 8h 偏移 = 4003_2008h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	TOT		TOTR			FXOR	WAS	TCRC	0							
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0								0							
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 2-19 CRC_CTRL 字段描述

位	描述
31 – 30 TOT	写入的转置类型 定义写入 CRC 数据寄存器的数据的转置配置。有关可用的转置选项，请参见转置特性说明。 00 无转置。 01 字节中的位转置；字节不转置。 10 字节中的位和字节均转置。 11 仅字节转置；字节中的位不转置。
29 – 28 TOTR	读取的转置类型 识别从 CRC 数据寄存器读取的数值的转置配置。有关可用的转置选项，请参见转置特性说明。 00 无转置。 01 字节中的位转置；字节不转置。 10 字节中的位和字节均转置。 11 仅字节转置；字节中的位不转置。
27 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
26 FXOR	CRC 数据寄存器的补充读取 某些 CRC 协议要求最终校验和与 0xFFFFFFFF 或 0xFFFF 进行异或运算。使该位的电平变为有效可使能已读取数据的即时补充。 0 读取时不执行异或运算。 1 反转或补充 CRC 数据寄存器的读取值。
25 WAS	作为种子写入 CRC 数据寄存器 电平变为有效后，写入 CRC 数据寄存器的值被视为种子值。电平变为无效后，写入 CRC 数据寄存器的值用作 CRC 计算中的数据。 0 写入 CRC 数据寄存器的是数据值。 1 写入 CRC 数据寄存器的是种子值。
24 TCRC	CRC 协议宽度。 0 16 位 CRC 协议。 1 32 位 CRC 协议。

23-0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

■ 2.5.5 功能说明

● 2.5.5.1 CRC 初始化/重新初始化

要使能 CRC 计算，用户必须在适用的寄存器内对 CRC_CTRL[WAS]、CRC_GPOLY、转置所必需的参数以及 CRC 结果反转进行编程。使 CRC_CTRL[WAS]的电平变为有效可实现将种子值编入 CRC_DATA 寄存器。完成 CRC 计算后，使 CRC_CTRL[WAS]的电平再次变为有效并进行种子的编程，且无论其值是全新数值还是之前使用过的种子值，都重新初始化 CRC 模块以便进行新的 CRC 计算。所有其他参数在进行种子值以及后续数据值的编程之前都必须设置。低功耗模式下，CRC 模块不可用。

● 2.5.5.2 CRC 计算

在 16 位 CRC 模式和 32 位 CRC 模式下，如果所有字节都是连续的，那么一次可进行 8 位 16 位或 32 位数据值的编程。非连续字节可能导致 CRC 计算错误。

❖ 2.5.5.2.1 16 位 CRC

如需计算 16 位 CRC：

1. 清零 CRC_CTRL[TCRC]以使能 16 位 CRC 模式。
2. 按 CRC 计算要求对转置进行编程，并在 CTRL 寄存器中补全选项位。更多详情，请参见转置特性和 CRC 结果补码 。
3. 将 16 位多项式写入 CRC_GPOLY[LOW]字段。CRC_GPOLY[HIGH]字段在 16 位 CRC 模式下不可用。
4. 置位 CRC_CTRL[WAS]以进行种子值的编程。
5. 将 16 位种子写入 CRC_DATA[LU:LL]。CRC_DATA[HU:HL]未使用。
6. 清零 CRC_CTRL[WAS]以开始写入数据值。
7. 将数据值写入 CRC_DATA[HU:HL:LU:LL]。每次执行数据值写入操作便计算 CRC，并将 CRC 中间值结果存回至 CRC_DATA[LU:LL]。
8. 完成所有数值的写入操作后，从 CRC_DATA[LU:LL]读取最终的 CRC 结果。

转置和补充操作在读取或写入数值的同时执行。更多详情，请参见转置特性 和 CRC 结果补码 。

❖ 2.5.5.2.2 32 位 CRC

如需计算 32 位 CRC：

1. 置位 CRC_CTRL[TCRC]以使能 32 位 CRC 模式。
2. 按 CRC 计算要求对转置进行编程，并在 CTRL 寄存器中补全选项位。更多详情，请参见转置特性和 CRC 结果补码 。
3. 将 32 位多项式写入 CRC_GPOLY[HIGH:LOW]。
4. 置位 CRC_CTRL[WAS]以进行种子值的编程。
5. 将 32 位种子写入 CRC_DATA[HU:HL:LU:LL]。
6. 清零 CRC_CTRL[WAS]以开始写入数据值。
7. 将数据值写入 CRC_DATA[HU:HL:LU:LL]。每次执行数据值写入操作便计算 CRC，并将 CRC 中间值结果存

回至 CRC_DATA[LU:LL]。

8. 完成所有数值的写入操作后，从 CRC_DATA[LU:LL]读取最终的 CRC 结果。

转置和补充操作在读取或写入数值的同时执行。更多详情，请参见转置特性和 CRC 结果补码 。

● 2.5.5.3 转置特性

默认情况下，转置特性未使能。然而，某些 CRC 标准要求对输入数据和/或最终校验和进行转置。用户软件可根据 CRC 标准需要选择单独配置每个转置操作。数据在读写的同时进行转置。某些协议计算 CRC 时对数据流采用低字节序格式。在这种情况下，转置特性非常有用，可以翻转位。该转置选项是 CRC 模块支持的功能类型之一。

❖ 2.5.5.3.1 转置类型

CRC 模块提供可翻转位和/或字节的多种转置功能类型，以便根据采用的 CRC 计算方法分别使用 CTRL[TOT]或 CTRL[TOTR]字段写入输入数据和读取 CRC 结果。

下列转置功能类型可用于对 CRC 数据寄存器进行读写操作：

1. CTRL[TOT]或 CTRL[TOTR]为 00。

   不发生转置。

2. CTRL[TOT]或 CTRL[TOTR]为 01。

   字节中的位转置，而字节不转置。

   reg[31:0]变为{reg[24:31], reg[16:23], reg[8:15], reg[0:7]}

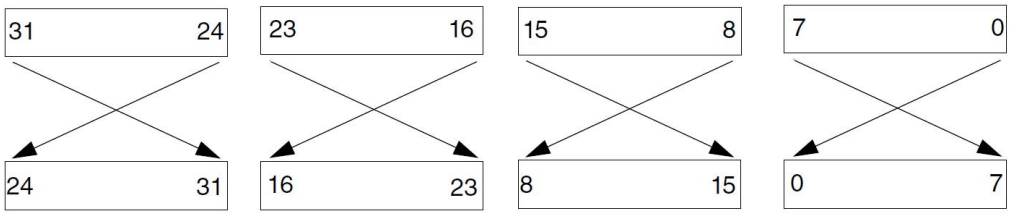


图 2-3 转置类型 01

3. CTRL[TOT]或 CTRL[TOTR]为 10。

   字节中的位和字节均转置。

   reg[31:0]变为 = {reg[0:7], reg[8:15],reg[16:23], reg[24:31]}

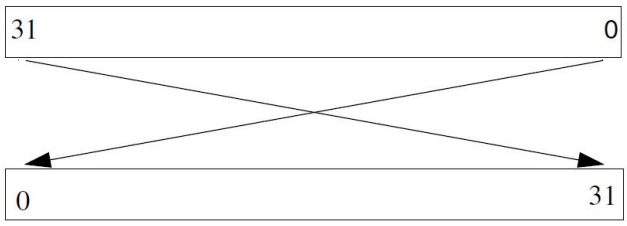


图 2-4 转置类型 10

4. CTRL[TOT]或 CTRL[TOTR]为 11。

   字节转置，但位不转置。

   reg[31:0]变为{reg[7:0], reg[15:8], reg[23:16], reg[31:24]}

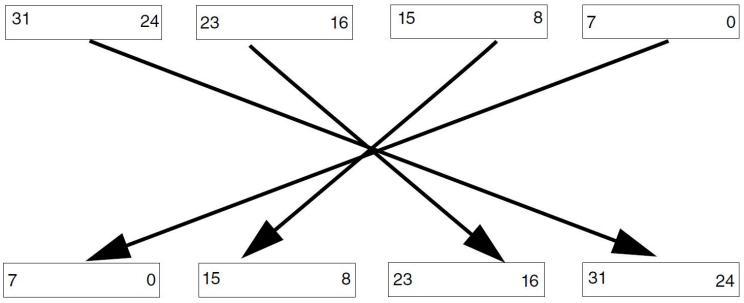


图 2-5 转置类型 11

注:为了对 CRC 数据寄存器进行 8 位或 16 位的写访问， 对不使用的字节该数据转置为零（将 32 位作为一个整体），但对于有效字节仅计算 CRC。读取 CRC 数据寄存器中的 16 位 CRC 结果并使用转置选项 10 和 11 时，转置后的结果数值位于 CRC[HU:HL]字段。读取 16 位 CRC 结果时，用户软件必须将该情形考虑在内，因此优先读取 32 位。

● 2.5.5.4 结果补码

CTRL[FXOR]置位后，对校验和进行补码。CRC 结果补码功能是在每次读 CRC 数据寄存器时将输出存储数据寄存器中校验和数值的补码。CTRL[FXOR]清零后，对 CRC 数据寄存器进行读操作会访问原始校验和数值。

■ 2.6 BOS 位操作存储模块

■ 2.6.1 简介

位操作存储模块 BOS(Bit Operation Storage)针对基于 Cortex-M0+的微控制器中对外设地址空间的读取-修改-写入存储器基元操作提供硬件支持。这种架构能力亦称为“位操作存储”，因其定义了一种机制，可为针对存储器映射外设的加载和存储操作提供位操作运算，而非仅仅读取数据值，或将其写入寻址存储器位置。在 BOS 定义中，“位操作存储”经编码后进入外设地址，用于存储器参考。该引擎模块能够降低总线的占有率和 CPU 执行时间，也可以节省代码空间，以达到降低系统能耗的效果。

通过组合 Cortex-M 指令集架构（v6M，v7M）的基本加载和存储指令，加上 BOS 提供的位操作存储命令，这种实施方法针对这类超低端微控制器具有可靠高效的读取-修改-写入能力。由该内核平台功能所定义的架构能力可在外设寄存器中实现 n 位字段操作，并与内置的 C 语言标准 I/O 硬件寻址相一致。就大多数 BOS 命令而言，单个内核读写总线周期转换为一个基元读取-修改-写入，即一个透明的“先读后写”总线序列。仅处理器内核产生的系统总线事务提供 BOS 位操作存储参考，其目标为标准 512 KB 外设地址空间，基地址为 0x4000_0000,SRAM_U 空间的基地址为 0x2000_0000。位操作存储嵌入地址位[28:19],在地址 0x4400_0000 – 0x5FFF_FFFF 处创建 448 MB 空间，用于 PBB（Peripherial Bus Bridge)；在地址 0x2400_0000 – 0x3FFF_FFFF 处创建 448 MB 空间，用于 SRAM_U；这些位略去了发送至外设总线控制器的实际地址，BOS 用它们来定义并控制其操作。

■ 2.6.2 特性

BOS 的主要特性包括：

- 针对选定地址空间进行的位操作存储的轻量级实施
- 将附加访问语义符号编码到参考地址中
- 位于处理器内核和开关主端口之间

- 符合 AHB 系统总线协议的两级流水线设计
- 可将非位操作存储访问组合传输至从设备总线控制器
- 可将来自处理器内核的位操作存储加载和存储转换为基元读取-修改-写入
- 位操作存储加载支持无符号位字段提取、加载和{置位，清零} 1 位操作
- 位操作存储支持位字段插入、逻辑“与”、“或”和“异或”操作
- 支持字节、半字和字大小的位操作存储
- 支持在 AHB 输出总线上执行最少的信号切换，以降低功耗

■ 2.6.3 结构框图

下图是该类超低端微控制器的处理器内核和平台的一般结构框图:

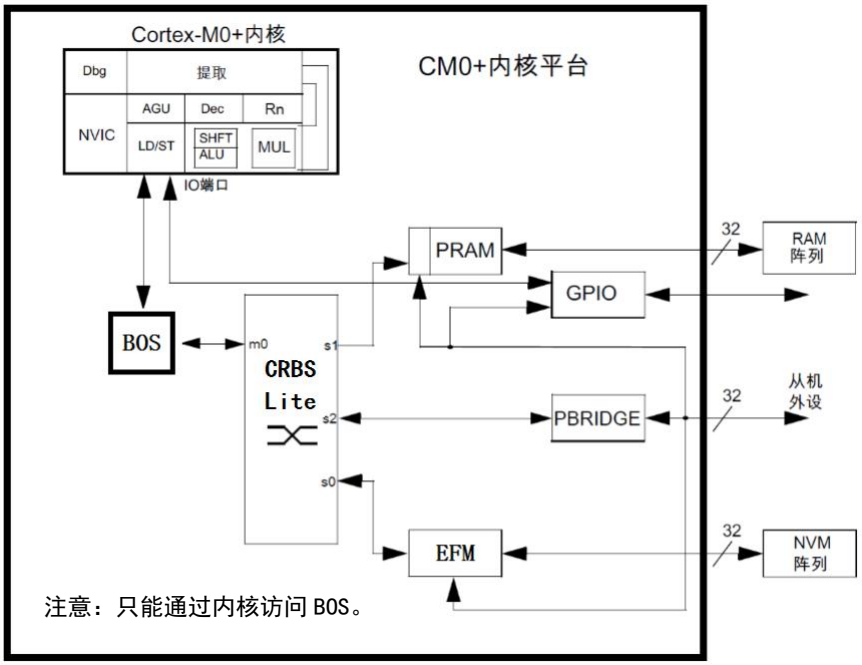


图 2-6 Cortex-M0+内核平台结构框图

如该结构框图所示，BOS 模块与交叉开关上的主端口连接，从而使其支持对 SRAM_U（图中所示的平台 RAM(PRAM)）和外设联接器(PBRIDGE)控制器进行的基元读取-修改-写入操作。BOS 硬件的微架构采用二级流水线设计，符合 AMBA、AHB 系统总线接口协议。PBRIDGE 模块将 AHB 系统总线协议转换为 APB 协议，用于连接的从外设。

■ 2.6.4 操作模式

BOS 模块不支持任何特殊操作模式。作为位于交叉主 AHB 系统总线端口上的内存映射器件，BOS 严格按照内存地址作出响应，以便访问 SRAM_U 和外设联接器总线控制器。所有 BOS 模块的相关功能都位于内核平台时钟域内，包括它与交叉主端口、SRAM_U 和 PBRIDGE 总线控制器的连接。

■ 2.6.5 存储器映射和寄存器说明

BOS 模块提供存储器映射功能，且不包含任何编程模型寄存器。BOS 真正支持的功能集详见功能说明 。

外设地址空间占据 512KB 区域：基地址为 0x4000_0000 的 512KB 加上基地址为 0x400F_F000 的 4KB 空间，用于 GPIO 访问；位操作存储地址空间映射至位于 0x4400_0000 – 0x5FFF_FFFF 的 448 MB 区域。与 SRAM_U 有关的位操作存储地址空间为映射在 0x2400_0000 – 0x3FFF_FFFF 的 448MB 区域。

■ 2.6.6 功能说明

此处信息详细说明 BOS 支持的特殊功能。如前文所述，Cortex-M 指令集架构（v6M，v7M）的基本加载和存储指令组合加上 BOS 提供的位操作存储概念，这种实施方法针对这类超低端微控制器具有可靠高效的读取-修改-写入能力。由该内核平台功能所定义的架构能力可在外设寄存器和 RAM 中实现 n 位字段操作，并与内置的 C 语言标准 I/O 硬件寻址相一致。就大多数 BOS 命令而言，单个内核读写总线周期转换为一个基元读取-修改-写入，即一个透明的“先读后写”总线序列。

应当首先执行位操作存储，然后执行位操作存储加载。

● 2.6.6.1 BOS 位操作存储

BOS 位操作存储支持的功能包括三个逻辑运算符（AND、OR、XOR）以及一个位字段插入。对于所有这些操作而言，BOS 可将单个位操作存储 AHB 存储事务转换为一个双周期基元读取-修改-写入序列，其中读取-修改组合操作在首个 AHB 数据阶段执行，然后写入操作在第二个 AHB 数据阶段执行。显示外设位字段插入位操作存储的一般时序示意图如下所示：

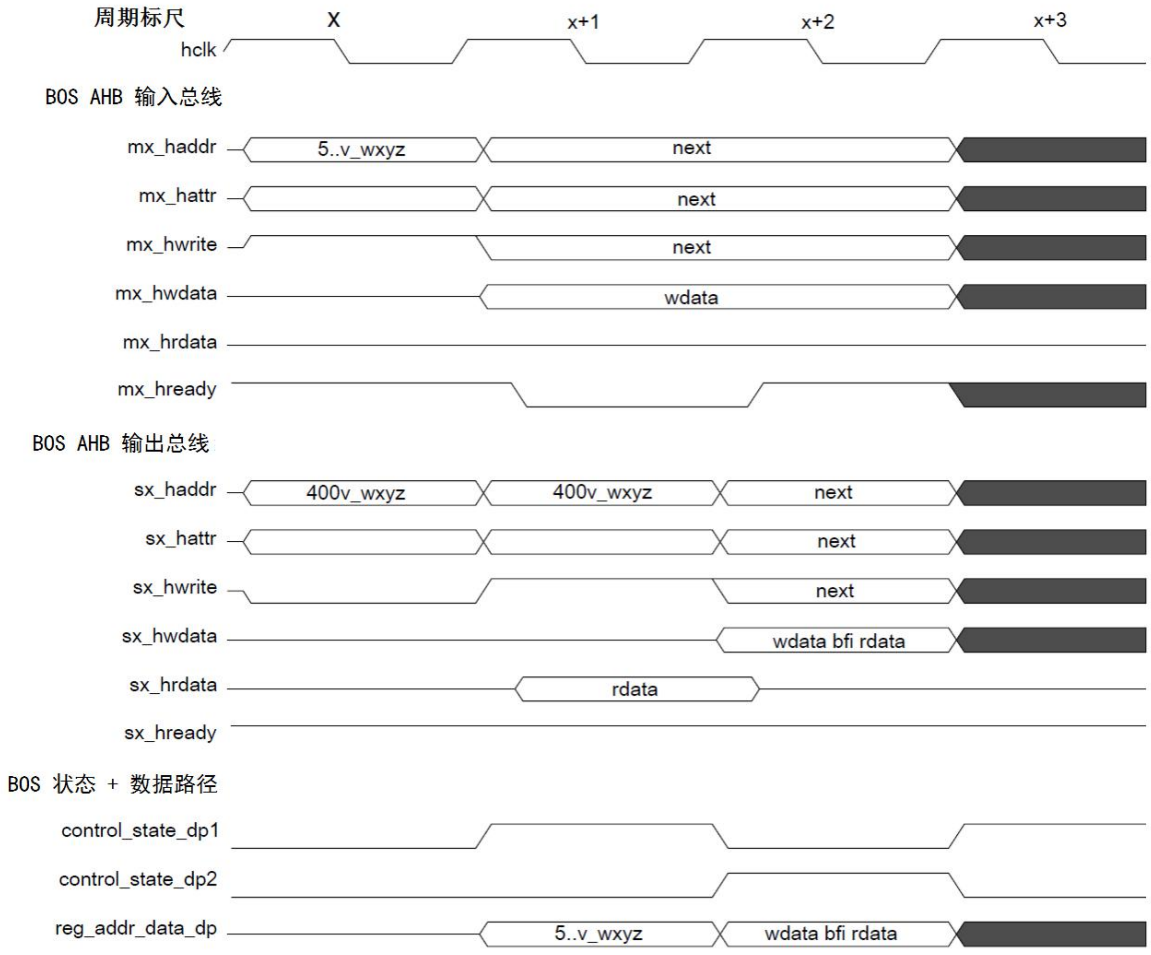


图 2-7 位操作存储：位字段插入时序示意图

所有位操作存储遵循上图中所示的相同执行模板，双周期读取-修改-写入操作：
1. 周期 x，首个 AHB 地址阶段：根据实际存储器地址（位操作存储已移除）将从输入总线的写入转换为输出总线上的读操作，然后在寄存器中捕捉。
2. 周期 x+1，第二个 AHB 地址阶段：写访问，并输出已寄存的（但却是实际的)存储器地址
3. 周期 x+1，首个 AHB 数据阶段：使用输入总线上的写入数据和位操作存储定义的功能对存储器读取数据进行修改，然后在数据寄存器中捕捉；输入总线周期停止。
4. 周期 x+2，第二个 AHB 数据阶段：已寄存的写入数据用作输出写入数据总线上的数据源。
注:由从设备插入的任何等待状态只通过 BOS 就传回主设备输入总线，从而逐周期停止 AHB 事务。

❖ 2.6.6.1.1 位操作存储逻辑与(AND)

该命令执行参考存储器位置的基元读取-修改-写入
1. 首先，读取该位置；
2. 然后，通过使用来自系统总线周期的写数据操作数执行逻辑与(AND)操作对其进行修改；
3. 最后，AND 操作结果写回参考存储器位置。
数据尺寸由写入操作定义，可以是字节（8 位）、半字（16 位）或字（32 位）。进行字节和半字传输时，内核执行所需的写入数据通道复制操作。

图 2-8 位操作存储地址：逻辑与(AND)

参见上图，其中 addr[30:29] = 01 (SRAM_U)，addr[30:29] = 10（外设），addr[28:26]= 001（指定 AND 操作），mem_addr[19:0]指定空间偏移地址（对于 SRAM_U，基地址为 0x2000_0000；对于外设，基地址为 0x4000_0000）。"-"表示该地址位为“无关位”。位操作存储 AND 写操作采用下列伪代码定义：

```
ioand<sz>(accessAddress, wdata) // decorated store AND
tmp = mem[accessAddress & 0xE00FFFFF, size] // memory read
tmp = tmp & wdata // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp // memory write
```

其中，操作数大小<sz>定义为 b（字节，8 位）、h（半字，16 位）和 w（字，32 位）。
本文档通篇采用此标记法。在周期定义表中，标记 AHB_ap 和 AHB_dp 表示 BOSAHB 事务的地址和数据阶段。逐周期 BOS 操作详情见下表。

表 2-20 位操作存储的周期定义：逻辑与(AND)

流水线级	周期		
	X	X+1	X+2
BOSAHB_ap	存储器的转发地址；解码位操作存储；将 master_wt 转换为 slave_rd；捕捉地址，属性	将捕捉到的地址以及属性循环输入到作为 slave_wt 的存储器中	<下一个>
BOSAHB_dp	<上一个>	执行存储器读取;在寄存器中形成形成(rdata & wdata)并捕捉目标数据	执行写操作，将寄存器中的数据发送至存储器

❖ 2.6.6.1.2 位操作存储逻辑或(OR)

该命令执行参考存储器位置的基本单元读取-修改-写入

1. 首先，读取该位置。
2. 然后，通过使用来自系统总线周期的写数据操作数执行逻辑或(OR)操作对其进行修改。
3. 最后，OR 操作结果写回参考存储器位置。

数据尺寸由写入操作定义，可以是字节（8 位）、半字（16 位）或字（32 位）。进行字节和半字传输时，内核执行所需的写入数据通道复制操作。

图 2-9 位操作存储地址存储：逻辑或(OR)

参见上图，其中 addr[30:29] = 01 (SRAM_U)，addr[30:29] = 10（外设），addr[28:26]= 010（指定 OR 操作），mem_addr[19:0]指定空间偏移地址（对于 SRAM_U，基地址为 0x2000_0000；对于外设，基地址为 0x4000_0000）。"-"表示该地址位为“无关位”。位操作存储 OR 写操作采用下列伪代码定义：

```
ioor<sz>(accessAddress, wdata) // decorated store OR
tmp = mem[accessAddress & 0xE00FFFFF, size] // memory read
tmp = tmp | wdata // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp // memory write
```

逐周期 BOS 操作详情见下表。

表 2-21 位操作存储的周期定义：逻辑或(OR)

流水线级	周期		
	X	X+1	X+2
BOSAHB_ap	存储器的转发地址；解码位操作	将捕捉到的地址以及属性循	<下一个>

	存储；将 master_wt 转换为 slave_rd；捕捉地址，属性	环输入到作为 slave_wt 的存储器中	
BOSAHB_dp	<上一个>	执行存储器读取；在寄存器中形成(rdata | wdata)并捕捉目标数据	执行写操作,将寄存器中的数据发送至存储器

❖ 2.6.6.1.3 位操作存储逻辑异或(XOR)

该命令执行参考存储器位置的基元读取-修改-写入

1. 首先，读取该位置。
2. 然后，通过使用来自系统总线周期的写数据操作数执行逻辑异或(XOR)操作对其进行修改。
3. 最后，XOR 操作结果写回参考存储器位置。

数据尺寸由写入操作定义，可以是字节（8 位）、半字（16 位）或字（32 位）。进行字节和半字传输时，内核执行所需的写入数据通道复制操作。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ioxorb	0	*	*	0	1	1	-	-	-	-	-	-	mem-addr																			
ioxorh	0	*	*	0	1	1	-	-	-	-	-	-	mem-addr																		0	
ioxorw	0	*	*	0	1	1	-	-	-	-	-	-	mem-addr																	0	0	

图 2-10 位操作存储地址存储：逻辑异或(XOR)

参见上图 ，其中 addr[30:29] = 01(SRAM_U)，addr[30:29] =10（外设），addr[28:26]= 011（指定 XOR 操作），mem_addr[19:0]指定外设空间偏移地址（对于 SRAM_U，基地址为 0x2000_0000；对于外设，基地址为 0x4000_0000）。"-"表示该地址位为“无关位”。位操作存储 XOR 写操作采用下列伪代码定义：

```
ioxor<sz>(accessAddress, wdata) // decorated store XOR
tmp = mem[accessAddress & 0xE00FFFFF, size] // memory read
tmp = tmp ^ wdata // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp // memory write
```

周期 BOS 操作详情见下表。

表 2-22 位操作存储的周期定义：逻辑 XOR

流水线级	周期		
	X	X+1	X+2
BOSAHB_ap	存储器的转发地址；解码位操作存储；将 master_wt 转换为 slave_rd；捕捉地址，属性	将捕捉到的地址以及属性循环输入到作为 slave_wt 的存储器中	<下一个>

BOSAHB_dp	<上一个>	执行存储器读取；在寄存器中形成(rdata ^ wdata)并捕捉目标数据	执行写操作，将寄存器中的数据发送至存储器

❖ 2.6.6.1.4 位操作存储位字段插入(BFI)

该命令使用一个基元读取-修改-写入序列，将由最低有效位(b)和位字段宽度(w+1)定义的写数据操作数中所含的位字段插入由与存储指令有关的访问尺寸定义的存储器“容器”中。

数据尺寸由写入操作定义，可以是字节（8 位）、半字（16 位）或字（32 位）。

注:就字大小的操作而言，最大位字段宽度为 16 位。进行字节和半字传输时，内核执行所需的写入数据通道复制操作。

BFI 操作可用于将单个位插入外设。就本例而言，w 字段置 0，表示位字段宽度为 1。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
lobfib	0	*	*	1	-	-	b	B	b	-	w	w	w	mem_addr																		
lobfih	0	*	*	1	-	b	b	b	b	w	w	w	w	mem_addr																		0
lobfiw	0	*	*	1	b	B	b	b	b	w	w	w	w	mem_addr																	0	0

图 2-11 位操作存储地址：位字段插入

其中，addr[30:29] = 01 (SRAM_U)，addr[30:29] =10（外设），addr[28] = 1（表示 BFI 操作），addr[27:23] = "b"（LSB 标识符），addr[22:19] = "w"（位字段宽度减 1 标识符），addr[18:0]指定外设空间偏移地址（对于 SRAM_U，基地址为 0x2000_0000；对于外设，基地址为 0x4000_0000）。"-"表示该地址位为“无关位”。注意，与其他位操作存储保存操作不同，BFI 使用 addr[19]作为"w"指示符中的最低有效位，而非地址位。

位操作存储 BFI 写操作采用下列伪代码定义：

```
iobfi<sz>(accessAddress, wdata) // decorated bit field insert
tmp = mem[accessAddress & 0xE007FFFF, size] // memory read
mask = ((1 << (w+1)) - 1) << b // generate bit mask
tmp = tmp & ~mask // modify
| wdata & mask
mem[accessAddress & 0xE007FFFF, size] = tmp // memory write
```

与存储指令有关的写数据操作数(wdata)包含待插入的位字段。它必须在右对齐的容器内正确排列，即字节操作为最低 8 位内，半字操作为最低 16 位内，字操作为整个 32 位内。

为便于说明，举例如下：向 8 位存储器容器中插入 3 位字段"xyz"，其初始设置为"abcd_efgh"。无论何种情况，w 均为 2，表示位字段宽度为 3。

```
if b = 0 and the decorated store (strb) Rt register[7:0] = ----_-xyz,
then destination is "abcd_exyz"
if b = 1 and the decorated store (strb) Rt register[7:0] = ----_xyz-,
then destination is "abcd_xyzh"
if b = 2 and the decorated store (strb) Rt register[7:0] = ---x_yz--,
then destination is "abcx_yzgh"
if b = 3 and the decorated store (strb) Rt register[7:0] = --xy_z---,
then destination is "abxy_zfgh"
if b = 4 and the decorated store (strb) Rt register[7:0] = -xyz_----,
then destination is "axyz_efgh"
if b = 5 and the decorated store (strb) Rt register[7:0] = xyz-_----,
then destination is "xyzd_efgh"
if b = 6 and the decorated store (strb) Rt register[7:0] = yz--_----,
then destination is "yzcd_efgh"
if b = 7 and the decorated store (strb) Rt register[7:0] = z---_----,
then destination is "zbcd_efgh"
```

该例中，需注意若起始位的位置加上字段宽度超出存储器尺寸，则仅向目标存储器位置插入一部分位字段源。若状态不同，假设(b + w+1) > container_width，则实际只有低阶"container_width - b"位被插入。

逐周期 BOS 操作详情见下表。

表 2-23 位操作存储的周期定义：位字段插入

流水线级	周期		
	X	X+1	X+2
BOS AHB_ap	存储器的转发地址；解码位操作存储；将 master_wt 转换为 slave_rd；捕捉地址，属性	将捕捉到的地址以及属性循环输入到作为 slave_wt 的存储器中	<下一个>
BOS AHB_dp	<上一个>	执行存储器读取；形成位屏蔽；在寄存器中形成按位数据 ((mask) ? wdata : rdata)) 并捕捉目标数据	执行写操作，将寄存器中的数据发送至存储器

● 2.6.6.2 BOS 位操作存储加载

BOS 位操作存储加载支持的功能包括：2 个加载-{置位，清零}单位操作以及无符号位字段提取。对于 2 个加载-{置位，清零}操作，BOS 可将单个位操作存储 AHB 加载事务转换为一个双周期基元读取-修改-写入序列，其中读取-修改组合操作在首个 AHB 数据阶段执行，然后写入操作在第二个 AHB 数据阶段执行，此时初始读取数据已返回至处理器内核。对于无符号位字段提取，位操作存储加载事务将在 BOS 中停止一个周期——因为数据字段已被提取——然后在第二个 AHB 数据阶段进行排列并返回至处理器。这是唯一一个非基元读取-修改-写入操作的位操作存储事务，因为它只是一次简单的数据读取。

显示外设加载-置位 1 位操作的位操作存储加载的一般时序示意图如下所示。

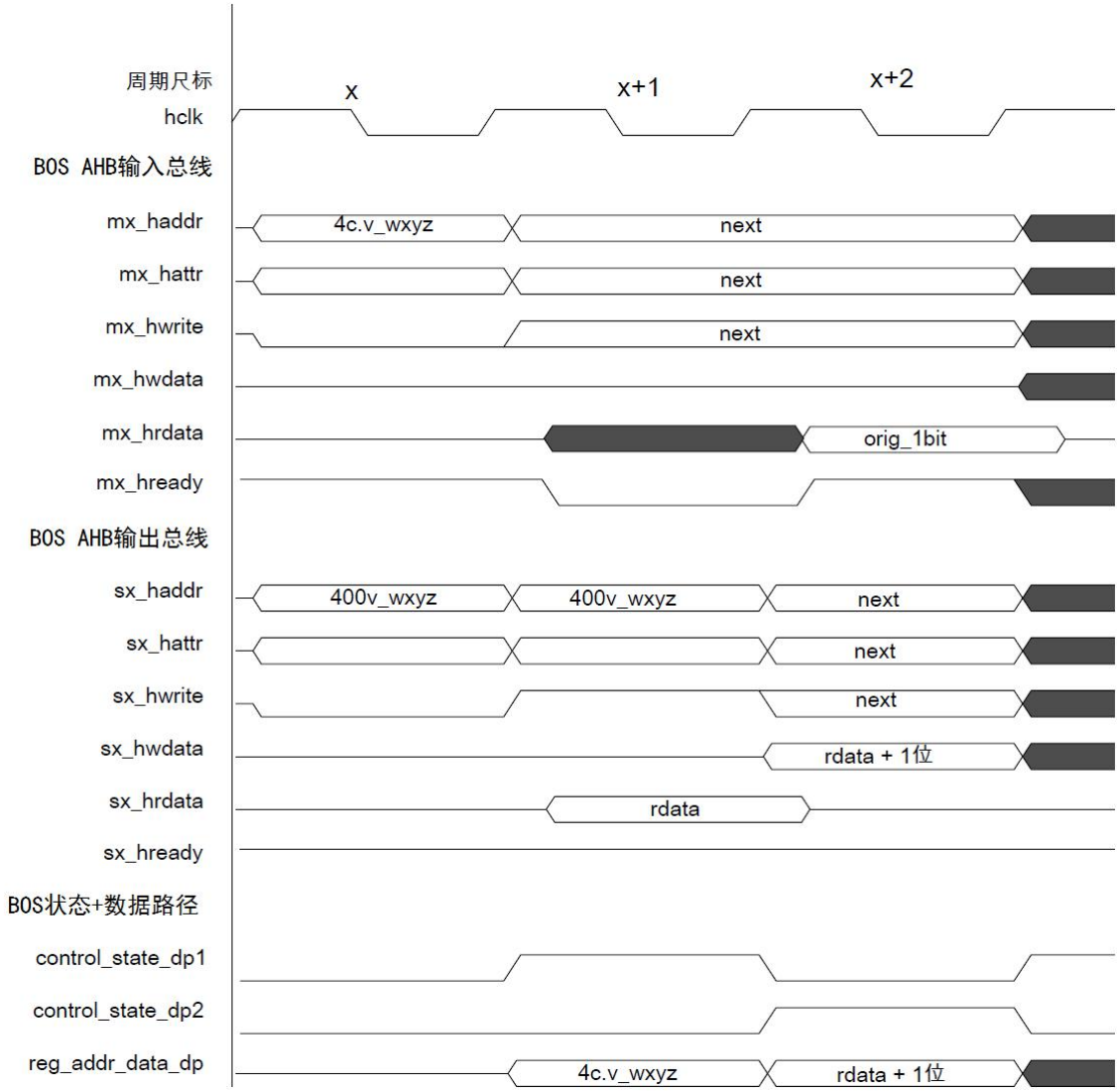


图 2-12 位操作存储加载：加载-置位 1 位字段插入时序示意图

位操作存储加载-{置位，清零}1 位操作遵循上图所示的执行模板：双周期读取-修改-写入操作：

1. 周期 x，首个 AHB 地址阶段：根据实际存储器地址（位操作存储已移除）将从输入总线的读取转换为输出总线上的读操作，然后在寄存器中捕捉。
2. 周期 x+1，第二个 AHB 地址阶段：写访问，并输出已寄存的（但却是实际的）存储器地址。
3. 周期 x+1，首个 AHB 数据阶段：在寄存器中捕捉“初始”1 位存储器读取数据，根据位操作存储（采用寄存器中捕捉到的已修改数据）定义的功能进行 1 位字段的置位或清零；输入总线周期停止。
4. 周期 x+2，第二个 AHB 数据阶段：将选定的初始 1 位数据右对齐、零填充，然后驱动至输入读取数据总线上，而已寄存的写入数据用作输出写入数据总线上的数据源。

注：由从设备插入的任何等待状态只通过 BOS 就传回主设备输入总线，从而逐周期停止 AHB 事务。显示无符号外设位字段操作的位操作存储加载的一般时序示意图如下图所示。

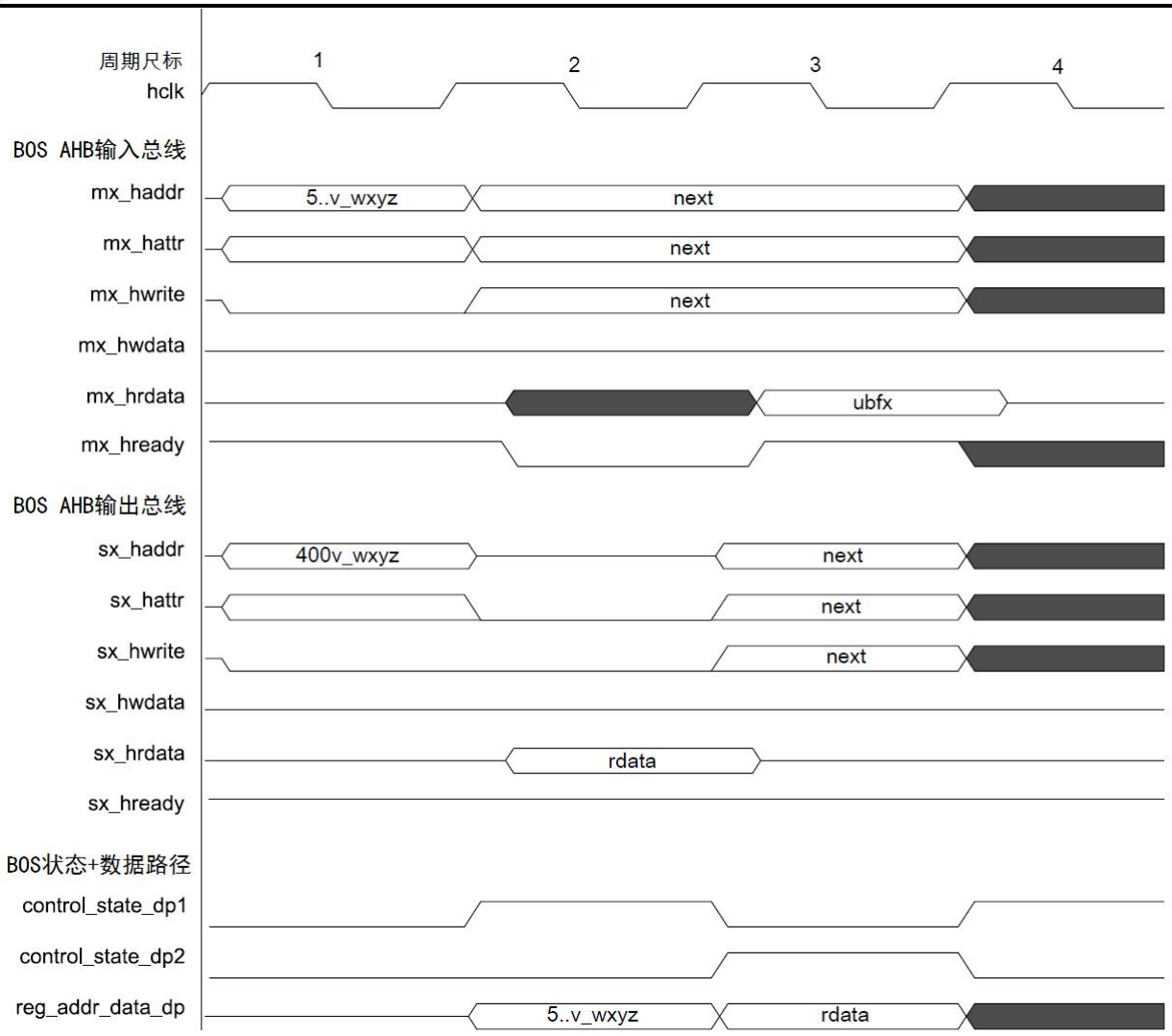


图 2-13 位操作存储加载：无符号位字段插入时序示意图

位操作存储无符号位字段提取遵循与上图中双周期读取操作相同的执行模板：

• 周期 x，首个 AHB 地址阶段：根据实际存储器地址（位操作存储已移除）将从输入总线的读取转换为输出总线上的读操作，然后在寄存器中捕捉。

• 周期 x+1，第二个 AHB 地址阶段：空闲周期

• 周期 x+1，第一个 AHB 数据阶段：根据起始位的位置和字段宽度生成一个位屏蔽；屏蔽位与存储器读取数据之间执行 AND 运算以隔离位字段；在数据寄存器中捕捉结果数据；输入总线周期停止

• 周期 x+2，第二个 AHB 数据阶段：将已寄存数据逻辑右对齐并驱动至输入读取数据总线

注:由从设备插入的任何等待状态只通过 BOS 就传回主设备输入总线，从而逐周期停止 AHB 事务。

❖ 2.6.6.2.1 位操作存储加载：1 位加载-清零（LAC1）

该命令将 LSB 位(b)定义的 1 位字段加载至内核通用目标寄存器(Rt)，并在执行基元读取-修改-写入序列后清零存储器空间中的位。从存储器地址中提取的 1 位数据字段在返回至内核的操作数中右对齐和零填充。

数据尺寸由读取操作指定，可以是字节（8 位）、半字（16 位）或字（32 位）。

	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0l ac 1b	0	*	*	0	1	0	-	-	b	b	b	-	mem_addr																			
0l ac 1h	0	*	*	0	1	0	-	b	b	b	B	-	mem_addr																		0	
0l ac 1w	0	*	*	0	1	0	b	b	b	b	B	-	mem_addr																	0	0	

图 2-14 位操作存储加载地址：1 位加载一清零

参见图 2-10 ，其中：addr[30:29] = 01 (SRAM_U)，addr[30:29] = 10（ 外设），addr[28:26]= 010（指定 1 位加载一清零操作），addr[25:21] = "b"（位标识符），mem_addr[19:0]指定空间偏移地址（对于 SRAM_U，基地址为 0x2000_0000；对于外设，基地址为 0x4000_0000）。"-"表示该地址位为“无关位”。位操作存储 1 位加载一清零读操作采用下列伪代码定义：

```
rdata = iolac1<sz>(accessAddress) // decorated load-and-clear 1
tmp = mem[accessAddress & 0xE00FFFFF, size] // memory read
mask = 1 << b // generate bit mask
rdata = (tmp & mask) >> b // read data returned to core
tmp = tmp & ~mask // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp // memory write
```

逐周期 BOS 操作详情见下表。

表 2-24 位操作存储加载的周期定义：1 位加载一清零

流水线级	周期		
	X	X+1	X+2
BOSAHB_ap	存储器的转发地址；解码位操作存储；捕捉地址，属性	将捕捉到的地址以及属性循环输入到作为 slave_wt 的存储器中	<下一个>
BOSAHB_dp	<上一个>	执行存储器读取；形成位屏蔽；从 rdata 提取位；在寄存器中形成 (rdata & ~mask)并捕捉目标数据	将已提取位返回至主设备；执行写操作，从而将已寄存的数据发送到存储器

❖ 2.6.6.2.2 位操作存储加载：1 位加载-置位(LAS1)

该命令将 LSB 位(b)定义的 1 位字段加载至内核通用目标寄存器(Rt)，并在执行基元读取-修改-写入序列后置位存储器空间中的位。从存储器地址中提取的 1 位数据字段在返回至内核的操作数中右对齐和零填充。

数据尺寸由读取操作指定，可以是字节（8 位）、半字（16 位）或字（32 位）。

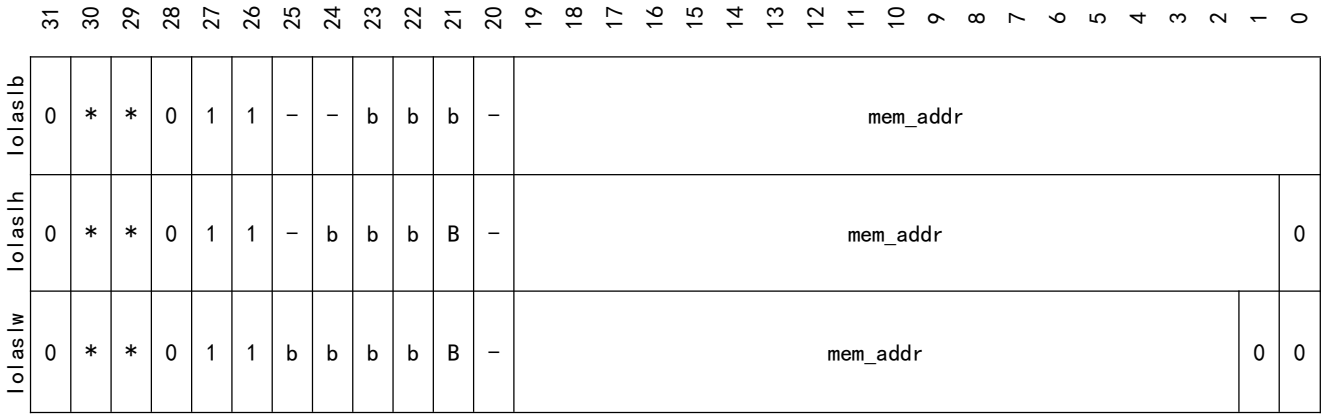


图 2-15 位操作存储加载地址：1 位加载－置位

其中，addr[30:29] = 01 (SRAM_U)，addr[30:29] = 10（外设），addr[28:26] = 011（指定 1 位加载－置位操作），addr[25:21] = "b"（位标识符），mem_addr[19:0]指定空间偏移地址（对于 SRAM_U，基地址为 0x2000_0000；对于外设，基地址为 0x4000_0000）。"-"表示该地址位为“无关位”。

位操作存储 1 位加载－置位读操作采用下列伪代码定义：

```
rdata = iolas1<sz>(accessAddress) // decorated load-and-set 1
tmp = mem[accessAddress & 0xE00FFFFF, size] // memory read
mask = 1 << b // generate bit mask
rdata = (tmp & mask) >> b // read data returned to core
tmp = tmp | mask // modify
mem[accessAddress & 0xE00FFFFF, size] = tmp // memory write
```

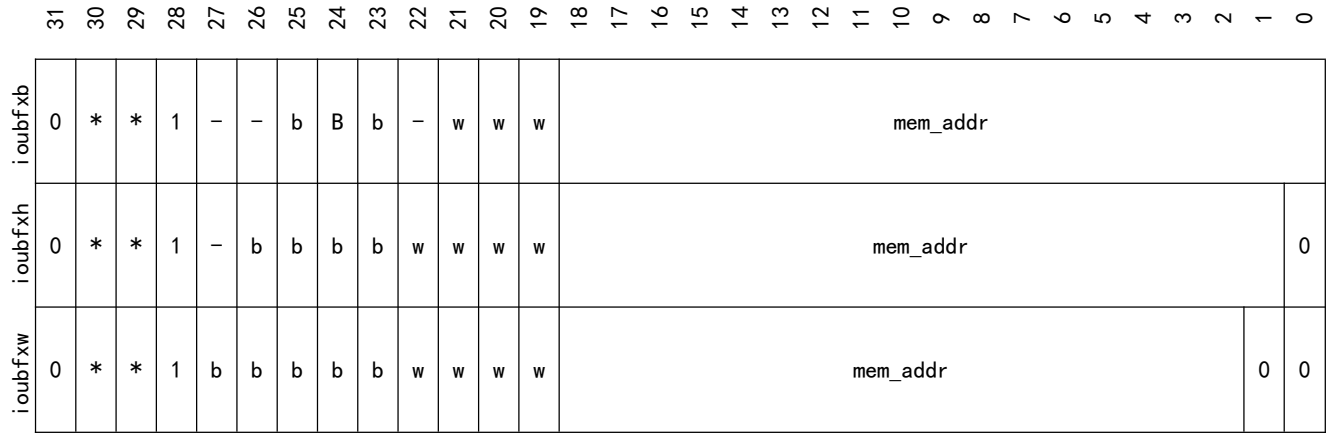
逐周期 BOS 操作详情见下表。

表 2-25 位操作存储加载的周期定义：1 位加载－置位

流水线级	周期		
	X	X+1	X+2
BOSAHB_ap	存储器的转发地址；解码位操作存储；捕捉地址，属性	将捕捉到的地址以及属性循环输入到作为 slave_wt 的存储器中	<下一个>
BOSAHB_dp	<上一个>	执行存储器读取；形成位屏蔽；从 rdata 提取位；在寄存器中形成 (rdata | mask)并捕捉目标数据	将已提取位返回至主设备；执行写操作，将已寄存的数据发送到存储器

❖ 2.6.6.2.3 位操作存储加载无符号位字段提取（UBFX）

该命令使用一个双周期读取序列，从与加载指令有关的访问尺寸所定义的存储器“容器”中提取 LSB 位(b)定义的位字节和位字段宽度(w+1)。从存储器地址中提取的位字段在返回至内核的操作数中右对齐和零填充。如前文所述，这是唯一一个不执行存储器写入的位操作存储，即 UBFX 仅执行读取操作。数据尺寸由写入操作定义，可以是字节（8 位）、半字（16 位）或字（32 位）。注意，就字大小的操作而言，最大位字段宽度为 16 位。



	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ioubfxb	0	*	*	1	-	-	b	B	b	-	w	w	w	mem_addr																		
ioubfxh	0	*	*	1	-	b	b	b	b	w	w	w	w	mem_addr																		0
ioubfxw	0	*	*	1	b	b	b	b	b	w	w	w	w	mem_addr																	0	0

图 2-16 位操作存储加载地址：无符号位字段提取

建议使用 UBFX 操作提取单个位。就本例而言，w 字段置 0，表示位字段宽度为 1。参见上图，其中：addr[30:29] = 01 (SRAM_U)，addr[30:29] = 10（外设），addr[28]= 1（指定无符号位字段提取操作），addr[27:23] = "b"（LSB 标识符），addr[22:19] ="w"（位字段宽度减 1 标识符），mem_addr[18:0]指定空间偏移地址（对于 SRAM_U，基地址为 0x2000_0000；对于外设，基地址为 0x4000_0000。"-"表示该地址位为“无关位”。

注:与其他位操作存储加载操作不同，UBFX 使用 addr[19]作为"w"指示器中的最低有效位，而非地址位。

位操作存储无符号位字段提取读操作采用下列伪代码定义：

```
rdata = ioubfx<sz>(accessAddress) // unsigned bit field extract
tmp = mem[accessAddress & 0xE007FFFF, size] // memory read
mask = ((1 << (w+1)) - 1) << b // generate bit mask
rdata = (tmp & mask) >> b // read data returned to core
```

与 BFI 操作相同，若起始位的位置加上字段宽度超出存储器容器尺寸时，仅从目标存储器位置提取一部分位字段源。若状态不同，假设(b + w+1) > container_width，则实际只有低阶"container_width - b"位被提取。逐周期 BOS 操作详情见下表。

表 2-26 位操作存储加载的周期定义：无符号位字段提取

流水线级	周期		
	X	X+1	X+2
BOS AHB_ap	存储器的转发地址；解码位操作存储；捕捉地址，属性	空闲 AHB 地址阶段	<下一个>
BOS AHB_dp	<上一个>	执行存储器读取;形成位执行存储器读取；形成位屏蔽；在寄存器中形成(rdata &mask)并捕捉目标数据	已寄存的数据逻辑右移；将对齐的 rdata 返回至主机

● 2.6.6.3 位操作存储地址和 GPIO 访问的更多详情

如前文所述,外设地址空间占用 516KB 区域:基地址为 0x4000_0000 的 512 KB 加上基地址为 0x400F_F000

的 4 KB 空间，用于 GPIO 访问。该存储器布局兼容同一系列，提供 129 个地址“插槽”，每个大小为 4KB。

GPIO 地址空间由硬件通过乘法映射：它在“标准”系统地址 0x400F_F000 处出现，并位于与地址 0x4000_F000 对应的地址插槽中。位操作存储加载和数据存储使情况稍显复杂，需访问 GPIO。如前文所述，对地址[19]的使用取决于位操作存储运算；对于 AND、OR、XOR、LAC1 和 LAS1 而言，该位用作真正地址位；而对于 BFI 和 UBFX，该位定义“w”位字段指示器的最低有效位。因此，非位操作存储的 GPIO 基准和位操作存储的 AND、OR、XOR、LAC1、LAS1 运算可使用标准 0x400F_F000 基地址，而位操作存储的 BF 和 UBFX 运算则必须使用替代的 0x4000_F000 基地址。另一种实施可以只将 0x400F_F000 用作基地址，以便进行全部非位操作存储 GPIO 访问，以及将 0x4000_F000 用作基地址，以便进行全部位操作存储访问。两种实施都由硬件提供支持。

表 2-27 位操作存储外设和 GPIO 地址详情

外部地址空间	说明
0x4000_0000 – 0x4007_FFFF	非位操作存储（正常）外设访问
0x4008_0000 – 0x400F_EFFF	非法地址；尝试的基准中止且错误已终止
0x400F_F000 – 0x400F_FFFF	使用标准地址的非位操作存储（正常）GPIO 访问
0x4010_0000 – 0x43FF_FFFF	非法地址；尝试的基准中止且错误已终止
0x4400_0000 – 0x4FFF_FFFF	位操作存储 AND、OR、XOR、LAC1、LAS1 以外设和基地址为 0x4000_F000 或 0x400F_F000 的 GPIO 为基准
0x5000_0000 – 0x5FFF_FFFF	位操作存储 BFI、UBFX 以外设和基地址仅为 0x4000_F000 的 GPIO 为基准

■ 2.6.7 应用信息

在本节中，以带 C 语言表达式操作数的 GNU 汇编器宏为例介绍执行位操作存储运算所需的指令。

本节专门介绍 BOS.h 文件的一部分，该文件定义的是位操作存储的汇编语言表达式：AND、OR 和 XOR。BFI 可比较函数和位操作存储加载更复杂，在完整的 BOS 头文件中可用。这些宏使用功能说明中介绍的相同的函数名。

```
#define IOANDW(ADDR,WDATA) \
__asm("ldr r3, =(1<<26);" \
"orr r3, %[addr];" \
"mov r2, %[wdata];" \
"str r2, [r3];" \
:: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
#define IOANDH(ADDR,WDATA) \
__asm("ldr r3, =(1<<26);" \
"orr r3, %[addr];" \
"mov r2, %[wdata];" \
"strh r2, [r3];" \
:: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
#define IOANDB(ADDR,WDATA) \
__asm("ldr r3, =(1<<26);" \
"orr r3, %[addr];" \
"mov r2, %[wdata];" \
```

```
"strb r2, [r3];" \
:: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
#define IOORW(ADDR,WDATA) \
__asm("ldr r3, =(1<<27);" \
"orr r3, %[addr];" \
"mov r2, %[wdata];" \
"str r2, [r3];" \
:: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
#define IOORH(ADDR,WDATA) \
__asm("ldr r3, =(1<<27);" \
"orr r3, %[addr];" \
"mov r2, %[wdata];" \
"strh r2, [r3];" \
:: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
#define IOORB(ADDR,WDATA) \
__asm("ldr r3, =(1<<27);" \
"orr r3, %[addr];" \
"mov r2, %[wdata];" \
"strb r2, [r3];" \
:: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
#define IOXORW(ADDR,WDATA) \
__asm("ldr r3, =(3<<26);" \
"orr r3, %[addr];" \
"mov r2, %[wdata];" \
"str r2, [r3];" \
:: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
#define IOXORH(ADDR,WDATA) \
    __asm("ldr r3, =(3<<26);" \
"orr r3, %[addr];" \
"mov r2, %[wdata];" \
"strh r2, [r3];" \
:: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
#define IOXORB(ADDR,WDATA) \
__asm("ldr r3, =(3<<26);" \
"orr r3, %[addr];" \
"mov r2, %[wdata];" \
"strb r2, [r3];" \
    :: [addr] "r" (ADDR), [wdata] "r" (WDATA) : "r2", "r3");
```

第 3 章 时钟模块

■ 3.1 简介

本章对基于 ARM Cortex-M0+内核的 NV32F100x 系列 MCU 时钟相关的模块做了详细说明，其中包括振荡器 OSC、内部时钟源 ICS 等模块内容。

■ 3.2 振荡器 OSC 模块

■ 3.2.1 简介

OSC 模块为 MCU 提供时钟源。OSC 模块可与外部晶振或谐振器一起为 MCU 生成用作参考时钟或总线时钟的时钟。

■ 3.2.2 特性

OSC 模块的主要特性有：

- 支持 32kHz 晶振（低范围模式）
- 支持 4－48 MHz 晶振和谐振器（高范围模式）
- 自动增益控制(AGC)，可使用低功耗模式优化两个频率范围内的功耗（低增益模式）
- 两种频率范围内均具有高增益选项：32 kHz，4－48 MHz
- 电压和频率滤波功能，确保时钟频率和稳定性
- 支持通过 ICS 使能

■ 3.2.3 结构框图

请参见下图 OSC 模块结构框图。

OSC 模块使用晶振或谐振器生成三个振荡器时钟滤波信号(XTL_CLK)。XTL_CLK 可在停止模式下工作，因为它们来自始终通电的硬件模块。OSCOS 决定 OSC_OUT 是来自内部振荡器(XTL_CLK)还是直接来自 EXTAL 引脚上设定的外部时钟。OSCOS 信号允许 XTAL 引脚用作 I/O 或测试时钟。

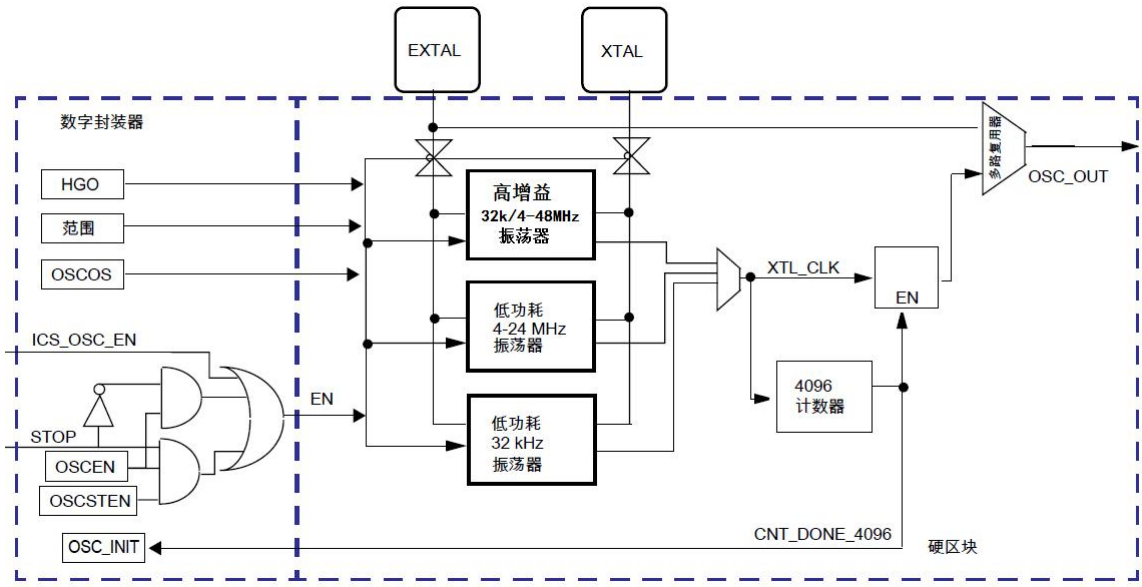


图 3-1 OSC 模块结构框图

下表介绍用户可访问的 OSC 模块信号。参见芯片级规格，了解外部引脚上实际连接了哪些信号。

表 3-1 OSC 信号说明

信号	说明	I/O
EXTAL	外部时钟/振荡器输入	模拟输入
XTAL	振荡器输出	模拟输出

■ 3.2.4 结构框图外部晶振/谐振器连接

晶振/谐振器频率参考的连接如图 3.2 和图 3.3 所示。在使用低频、低功耗模式时，唯一的外部组件就是晶振或谐振器本身。在其他振荡器模式下，需要使用负载电容器(C x ,C y)和反馈电阻器(RF)。另外，高增益模式下还要使用串联电阻器(RS)。数据表中列出了建议的组件值。

表 3-2 外部晶振/振荡器连接

振荡器模式	连接
低频、高增益	Connection2
低频、低功耗	Connection1
低频、高增益(4-20MHz)	Connection2
低频、低功耗(4-20MHz)	Connection2

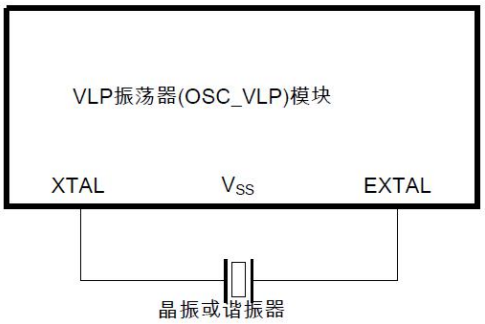


图 3-2 晶振/谐振器连接 - 连接 1

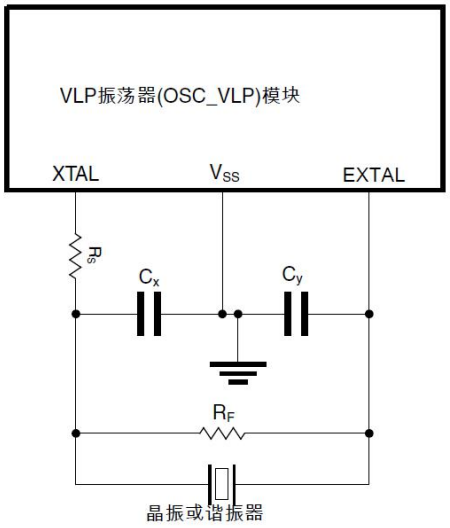


图 3-3 晶振/谐振器连接 - 连接 2

■ 3.2.5 外部时钟连接

在外部时钟模式下(OSC_CR[OSCOS] = 0)，引脚可如下图所示连接。

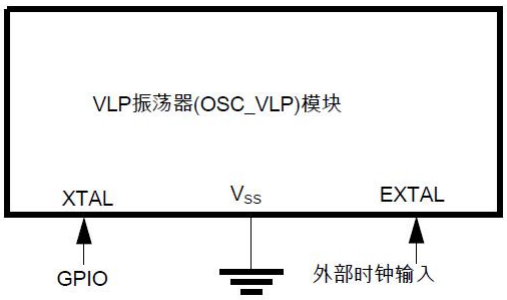


图 3-4 外部时钟连接

■ 3.2.6 存储器映射

表 3-3 OSC 存储器映射

绝对地址(16 进制)	寄存器名称	偏移量	位宽	权限	复位值
0x4006_5000h	OSC 控制寄存器 (OSC_CR)	00h	8	R/W	00h

● 3.2.6.1 OSC 控制寄存器(OSC_CR)

地址: 4006_5000h 基准 + 0h 偏移 = 4006_5000h

位	7	6	5	4	3	2	1	0
读	OSCEN	0	OSCSTEN	OSCOS	0	RANGE	HGO	OSCINIT
写								
复位	0	0	0	0	0	0	0	0

表 3-4 OSC_CR 字段描述

位	描述
7 OSCEN	OSC 使能 使能 OSC 模块。OSC 模块也可通过 ICS 模块使能。 0 OSC 模块禁用。 1 OSC 模块使能。
6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5 OSCSTEN	停止模式下的 OSC 使能 控制 OSC 时钟在 MCU 进入停止模式且 OSCEN 置位时是否保持使能。如果 ICS 请求 OSC 使能，则 OSCSTEN 无效。 0 OSC 时钟在停止模式下禁用。 1 OSC 时钟在停止模式下保持使能。
4 OSCOS	OSC 输出选择 选择 OSC 模块的输出时钟。 0 选择来自 EXTAL 引脚的外部时钟源。

	1 选择振荡器时钟源。
3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2 RANGE	频率范围选择 选择 OSC 模块的频率范围。 0 32 kHz 的低频范围。 1 4－48MHz 的高频范围。
1 HGO	高增益振荡器选择 控制 OSC 工作模式。 0 低功耗模式 1 高增益模式
0 OSCINIT	OSC 初始化 该字段在振荡器初始化周期完成后置位。 0 振荡器初始化未完成。 1 振荡器初始化已完成。

■ 3.2.7 功能说明

● 3.2.7.1 模块状态(OSC_CR)

OSC 模块有三种状态。状态图可参见图 3-5。本节说明各状态以及相互之间的转换。

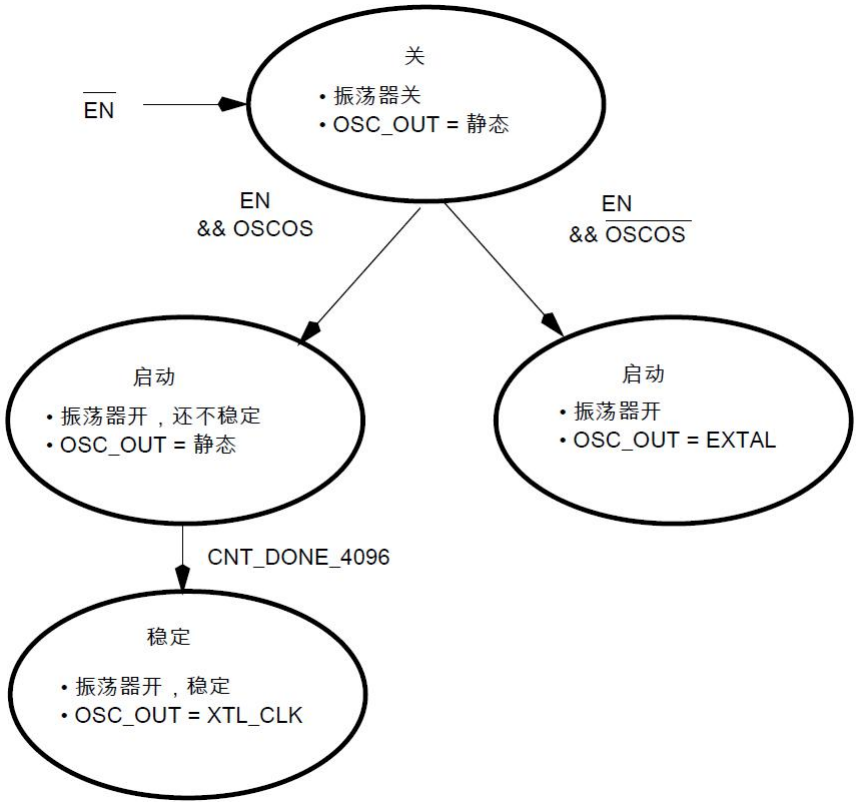


图 3-5 OSC 模块状态图

EN 由 OSC_CR[OSCEN]、停止、OSC_CR[OSCSTEN]和外部请求(ICS_OSC_EN)所确定。详情可参见下表。

表 3-5 EN 状态

EN	ICS_OSC_EN	OSC_CR[OSCEN]	OSC_CR[OSCSTEN]	停止
1	1	–	–	–
1	0	1	–	0
1	0	1	1	1
0	0	1	0	1
0	0	0	–	–

❖ 3.2.7.1.1 关断态

只要 EN 信号被否定，便进入关态。进入该状态后，XTL_CLK 和 OSC_OUT 为静态。该状态下，EXTAL 和 XTAL 引脚亦从所有其他振荡器电路去耦。OSC 模块电路配置为消耗最小的电流。

❖ 3.2.7.1.2 振荡器启动

只要振荡器首次使能，振荡器便进入启动状态（EN 转换为高电平），并且 OSC_CR[OSCOS]为高电平。该状态下，OSC 模块使能，开始振荡但尚未稳定。当振荡幅度变得足够大，可以通过输入缓冲区时，XTL_CLK 开始为计数器提供时钟源。当计数器看到 XTL_CLK 的 4096 个周期时，振荡器可视作稳定，并且 XTL_CLK 传输至输出时钟 OSC_OUT。

❖ 3.2.7.1.3 振荡器稳定

只要振荡器使能，振荡器便进入稳定状态（EN 高电平），OSC_CR[OSCOS]为高电平，并且计数器能看到 4096 个 XTL_CLK 周期（CNT_DONE_4096 为高电平）。在此状态下，OSC 模块在 OSC_OUT 上产生稳定的输出时钟。其频率由所用的外部组件确定。

❖ 3.2.7.1.4 外部时钟模式

使能振荡器（EN 高电平）且 OSC_CR[OSCOS]为低电平时，进入外部时钟状态。此状态下，OSC 模块设置为将来自 EXTAL 的时钟缓冲（有迟滞）至 OSC_OUT。其频率由提供的外部时钟所确定。

● 3.2.7.2 模块模式

振荡器是一个皮尔斯型振荡，支持外部晶振或谐振器，可在下表所示之频率范围内工作，这些模式假定 EN = 1OSC_CR[OSCOS] =1。

表 3-6 振荡器模式

范围	HGO	模式	频率范围
0	1	低频率、高增益	f_{lo}(min)到最高
0	0	低频率、低功耗(VLP)	f_{lo}(max)
1	1	高频率模式1、高增益	f_{hi}(min)到最高
1	0	高频率模式1、低功耗	f_{hi}(max)

❖ 3.2.7.2.1 低频率、高增益模式

在低频率、高增益模式下（OSC_CR[RANGE] = 0、OSC_CR[HGO] = 1），振荡器使用简单的逆变器类放大器。增益设置为可获得轨到轨振荡幅度。此模式下的振荡器输入缓冲区为单端式。它为电压滤波提供低通频率过滤和迟滞功能，并将输出转换为逻辑电平。

❖ 3.2.7.2.2 低频率、低功耗模式

在低频率、低功耗模式下（OSC_CR[RANGE] = 0、OSC_CR[HGO] = 0），振荡器使用增益控制环路来最大程度地降低功耗。随着振荡幅度的增加，放大器电流下降。这种状况将持续，直到所需的幅度到达稳态。此模式下的振荡器输入缓冲区为单端式。它为电压滤波提供低通频率过滤和迟滞功能，并将输出转换为逻辑电平。在该模式下，放大器输入、增益控制输入和输入缓冲区输入均为容性耦合，以提升漏电容差（对 EXTAL 的直流电平不敏感）。

该模式下，集成了除谐振器本身以外的所有外部元器件，包括：负载电容和反馈电阻，可偏置 EXTAL。

❖ 3.2.7.2.3 高频率、高增益模式

在高频率、高增益模式下（OSC_CR[RANGE] = 1、OSC_CR[HGO] = 1），振荡器使用简单的逆变器类放大器。增益设置为可获得轨到轨振荡幅度。此模式下的振荡器输入缓冲区为单端式。它为电压滤波提供低通频率过滤和迟滞功能，并将输出转换为逻辑电平。

❖ 3.2.7.2.4 高频率、低功耗模式

在高频率、低功耗模式下（OSC_CR[RANGE]=1、OSC_CR[HGO]=0），振荡器使用增益控制环路来最大程度地降低功耗。随着振荡幅度的增加，放大器电流下降。这种状况将持续，直到所需的幅度到达稳态。
此模式下的振荡器输入缓冲区为差分式。它为电压滤波提供低通频率过滤和迟滞功能，并将输出转换为逻辑电平。

❖ 3.2.7.2.5 计数器

振荡器输出时钟(OSC_OUT)将被门控关闭，直到计数器检测到 4096 个输入时钟周期(XTL_CLK)。一旦 4096 个周期完成后， 计数器会将 XTL_CLK 传送到 OSC_OUT 上。此计数超时可用于确保输出时钟的稳定性。

❖ 3.2.7.2.6 参考时钟的引脚要求

在振荡器模式下，OSC 模块需要同时使用 EXTAL 和 XTAL 引脚来生成输出时钟，但在外部时钟模式下，只需使用 EXTAL 引脚。只要满足数据表中列出的规格，便可将 EXTAL 和 XTAL 引脚用于 I/O 或测试时钟用途。

■ 3.3 内部时钟源 ICS 模块

■ 3.3.1 简介

内部时钟源(ICS)模块为 MCU 提供时钟源选择。该模块包含一个锁频环(FLL)作为时钟源，该锁频环可以采用内部或外部的参考时钟。该模块可提供该 FLL 时钟或者内部或外部参考时钟作为 MCU 系统时钟的时钟源。此外还提供可用于控制低功耗振荡器(OSC)模块的信号。这些信号配置并使能 OSC 模块以生成其外部

晶体/谐振器时钟(OSC_OUT)，供外设模块使用以及用作 ICS 外部参考时钟源。ICS 外部参考时钟可以是 OSC 提供的外部晶体/谐振器(OSC_OUT)，也可以是另一个外部时钟源。选定的 ICS 时钟源通过一个精简的总线分频器(BDIV)，以便产生较低的最终输出时钟频率。

■ 3.3.2 特性

内部时钟源具有以下特性：

- FLL 是可调整精度的；内置 FLL，倍频 1280
- 内部时钟和外部时钟源都可以作为 FLL 的参考输入
- 针对外部时钟提供了基准分频器
- 外部时钟需要先调整到 26K～40K 之间
- 内部时钟源有 9 位有效修调位
- 内部和外部时钟源都可以作为 MCU 的时钟参考源
- 无论选择哪个时钟作为时钟源，都可以对其进行分频
  - ❖ 针对时钟分频器提供 3 位选择
  - ❖ 可用分频比为：1、2、4、8、16、32、64、128（OSC_CR[RANGE]=0）；32、64、128、256、512、1024（OSC_CR[RANGE]=0）
- FLL 内部启用模式在退出复位后被自动选中
- 可选数字控制振荡器(DCO)优化的频率范围
- FLL 锁定检测器和外部时钟监控器
  - ❖ 具有中断功能的 FLL 锁定检测器
  - ❖ 具有复位功能的外部参考时钟监视器
- 数字控制振荡器针对 40-48MHz 频率范围进行了优化

■ 3.3.3 结构框图

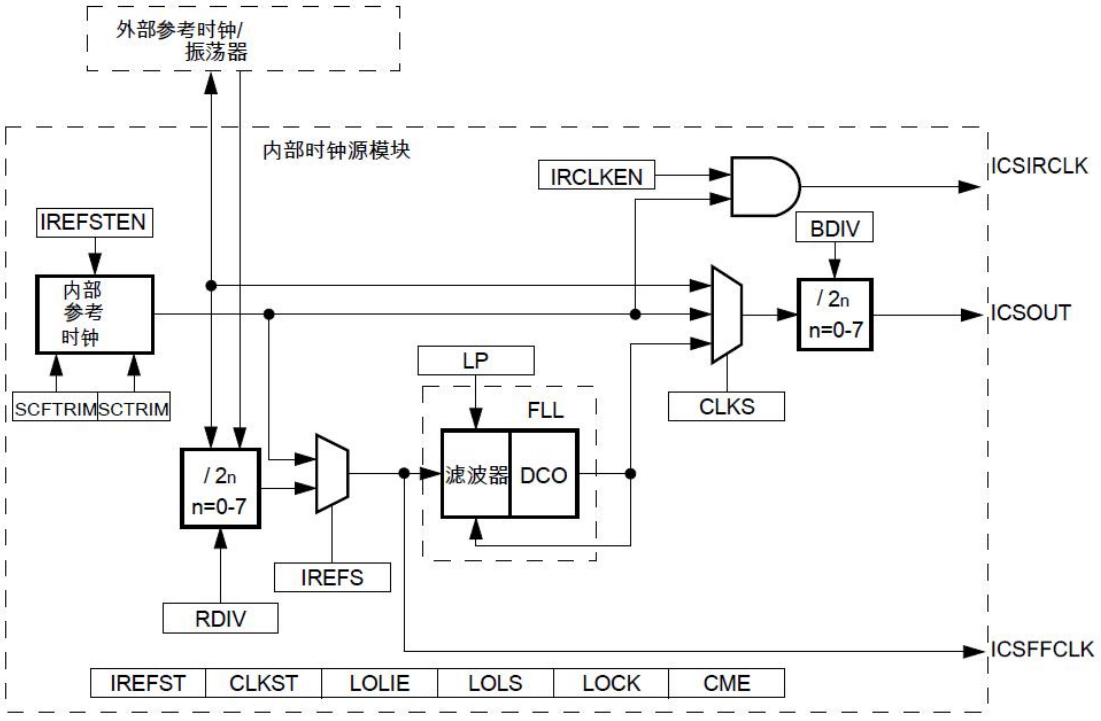


图 3-6 ICS 结构框图

■ 3.3.4 工作模式

ICS 有七种工作模式：FEI、FEE、FBI、FBILP、FBE、FBELP 和 STOP。下表分别做了说明:

表 3-7 ICS 工作模式

编号	模式	说明
1	FLL 内部启用(FEI)	默认模式，在该模式下，ICS 提供一个来源于 FLL 的时钟，FLL 时钟由内部参考时钟控制。
2	FLL 外部使用(FEE)模式	该模式下，ICS 提供一个来源于 FLL 的时钟，该 FLL 时钟由外部参考时钟控制。
3	FLL 内部旁路(FBI)	FLL 使能并通过内部参考时钟来控制，但处于旁路状态。ICS 提供从内部参考时钟分配而来的时钟。
4	FLL 内部旁路低功耗(FBILP)	FLL 禁用并旁路，ICS 提供由内部参考时钟分频而来的时钟。
5	FLL 外部旁路的低功耗(FBELP)	FLL 禁用并旁路，ICS 提供由外部参考时钟分频而来的时钟。
6	FLL 外部旁路(FBE)	FLL 使能并通过外部参考时钟来控制，但处于旁路状态。ICS 提供从外部参考时钟源分频而来的时钟。
7	停止 STOP	FLL 禁用,内部或 ICS 外部基准时钟源(OSC_OUT)是使能还是禁用可以选型。ICS 不提供任何 MCU 时钟源。

■ 3.3.5 存储器映射和寄存器说明

表 3-8 ICS 存储器映射

绝对地址(16 进制)	寄存器名称	偏移量	位宽	权限	复位值
4006_4000	内部时钟源控制寄存器 0（ICS_C1）	00h	8	R/W	04h
4006_4001	内部时钟源控制寄存器 1（ICS_C2）	01h	8	R/W	20h
4006_4002	内部时钟源控制寄存器 2（ICS_C3）	02h	8	R/W	未定义
4006_4003	内部时钟源控制寄存器 3（ICS_C4）	03h	8	R/W	参见本节
4006_4004	内部时钟源状态寄存器（ICS_S）	04h	8	R	10h

● 3.3.5.1 ICS 控制寄存器 1（ICS_C1）

此寄存器控制内部时钟源的参考时钟选择，参考时钟的分频倍数以及内部时钟源的工作模式。

地址: 4006_4000h 基准 + 0h 偏移 = 4006_4000h

位	7	6	5	4	3	2	1	0
读写	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
复位	0	0	0	0	0	0	0	0

表 3-9 ICS_C1 字段描述

位	描述
7-6	输出时钟源选择

CLKS	00 FLL 的输出 01 内部时钟源 10 外部时钟源 11 保留，默认值为 00。
5-3 RDIV	参考时钟分频系数，参考时钟的分频结果必须是 26k～40k

	OSC_CR[RANGE]=0	OSC_CR[RANGE]=1
000	1	32
001	2	64
010	4	128
011	8	256
100	16	512
101	32	1024
110	64	2048
111	128	保留

2 IREFS	FLL 参考时钟源选择 0 选择外部参考时钟源 1 选择内部参考时钟源
1 IRCLKEN	内部 IRC 时钟源使能 0 内部 IRC 不工作 1 内部 IRC 工作
0 IREFSTEN	内部参考时钟源 STOP 模式下是否有效 0 在 STOP 模式下内部时钟源不工作 1 如果 IRCLKEN 有效，或者在 FEI,FBI,FBILP 进入 STOP 模式，在 STOP 模式下内部时钟源保持使能

● 3.3.5.2 ICS 控制寄存器 2（ICS_C2）

ICS_C2 用来控制总线时钟频率。

地址：4006_4000h（基址）+ 1h（偏移量）= 4006_4001h

位	7	6	5	4	3	2	1	0
读	BDIV			LP	0			
写								
复位	0	0	1	0	0	0	0	0

表 3-10 ICS_C2 字段描述

位	描述
7-5 BDIV	内部时钟源分频参数 000 对选中的时钟源做 1 分频 001 对选中的时钟源做 2 分频（复位初始值） 010 对选中的时钟源做 4 分频

	011 对选中的时钟源做 8 分频 100 对选中的时钟源做 16 分频 101 对选中的时钟源做 32 分频 110 对选中的时钟源做 64 分频 111 对选中的时钟源做 128 分频
4 LP	低功耗选择 0 在 bypass 模式 FLL 不会被禁止 1 除非 debug 模式，FLL 将会禁止在 bypass 模式。
3-0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

● 3.3.5.3 ICS 控制寄存器 3（ICS_C3）

ICS_C3 主要用来控制内置振荡器的频率。（TRIM 值出厂可校准，默认 IRC-37.5K,倍频后 48M）

地址：4006_4000h（基址）+ 2h（偏移量）= 4006_4002h

位	7	6	5	4	3	2	1	0
读 写	SCTRIM							
复位	*	*	*	*	*	*	*	*

* 没有初始值

表 3-11 ICS_C3 字段描述

位	描述
7～0 SCTRIM	通过增加内部振荡器周期来控制，增加这个值，将会增加内置振荡器的频率。还有另一位寄存器在 ICS_C4[SCTRIM]

● 3.3.5.4 ICS 控制寄存器 4（ICS_C4）

地址：4006_4000h（基址）+3h（偏移量）= 4006_4003h

位	7	6	5	4	3	2	1	0
读 写	LOLIE	0	CME	0				SCFTRIM
复位	0	0	0	0	0	0	0	*

* 没有初始值

表 3-12 ADC_SC4 字段描述

位	描述
7 LOLIE	Loss of lock 中断使能 0 Loss of lock 时不触发中断 1 Loss of lock 触发中断
6 保留	这个字段被保留。 这个只读域被保留，并且总是具有值 0。

5 CME	时钟模拟使能，当发现一个外部时钟源 Loss，产生一个 Reset。 0 CME 禁止 1 当 LOL 外部时钟时，产生一个 Reset
4～1 保留	这个字段被保留。 这个只读域被保留，并且总是具有值 0。
0 SCFTRIM	通过增加内部振荡器周期来控制，增加这个值，将会增加内置振荡器的频率。

● 3.3.5.5 ICS 状态寄存器（ICS_S）

ICS_S 内部时钟源状态寄存器

地址：4006_4000h（基址）+ 4h（偏移量）= 4006_4004h

位	7	6	5	4	3	2	1	0
读	LOLS	LOCK	0	IREFST	CLKST		0	
写								
复位	0	0	0	1	0	0	0	0

表 3-13 ICS_S 字段描述

位	描述
7 LOLS	Loss of Lock 状态 当 FLL 没锁定的时候，LOLS 将置 1，当 reset 或者对该位写 1 时，LOLS 清 0
6 LOCK	锁定状态，表明 FLL 是否锁定，如果和 FLL 相关的寄存器重新设置，该位将会被清 0 0 FLL 没有锁定 1 FLL 是锁定的
5 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
4 IREFST	参考时钟的状态 0 参考时钟为外部时钟 1 参考时钟为内部时钟
3～2 CLKST	时钟模式状态，选择哪个时钟作为时钟输出 00 选择 FLL 时钟 01 跳过 FLL，直接采用内部时钟（31k～39K，出厂校准默认 37.5K） 10 跳过 FLL，直接采用外部时钟 11 保留值
1～0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

■ 3.3.6 功能说明

● 3.3.6.1 ICS 控制寄存器 1（ICS_C1）

状态图中显示了 ICS 的 7 种状态，介绍如下，箭头表示各状态间允许的切换。

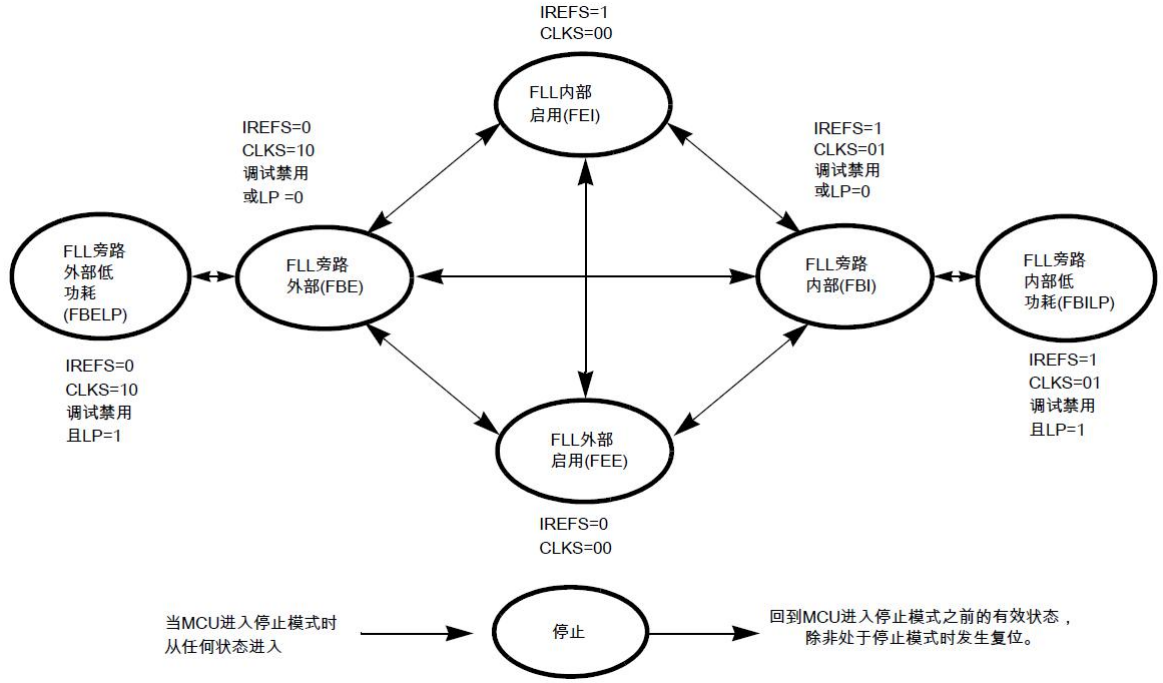


图 3-7 时钟切换模式

● 3.3.6.2 FLL 内部启用（FEI）

FLL 内部启用(FEI)模式是默认工作模式，当出现以下所有条件时进入该模式：

- ❖ 00b 写入 ICS_C1[CLKS]。
- ❖ 1b 写入 ICS_C1[IREFS]。

在“FLL 内部启用”模式下，ICSOUT 时钟从 FLL 时钟获得，FLL 时钟由内部参考时钟控制。FLL 环路将频率锁定到内部参考频率的 1280 倍。内部参考时钟源已启用。

● 3.3.6.3 FLL 外部启用（FEE）

出现以下所有条件时，进入 FLL 外部启用(FEE)模式：

- ❖ 00b 写入 ICS_C1[CLKS]。
- ❖ 0b 写入 ICS_C1[IREFS]。
- ❖ 对 ICS_C1[RDIV]和 OSC_CR[RANGE]进行写操作可将外部参考时钟分频到 31.25kHz 至 39.0625kHz 范围内。

在“FLL 外部启用”模式下，ICSOUT 时钟从 FLL 时钟获得，FLL 时钟由外部参考时钟源控制。FLL 环路将 FLL 频率锁定到外部基准频率（由 ICS_C1[RDIV]和 OSC_CR[RANGE]选择）的 1280 倍。外部参考时钟源已启用。

● 3.3.6.4 FLL 内部旁路（FBI）

出现以下所有条件时，进入 FLL 内部旁路(FBI)模式：

- ❖ 01b 写入 ICS_C1[CLKS]。
- ❖ 1b 写入 ICS_C1[IREFS]。

在“FLL 内部旁路”模式下，ICSOUT 时钟从内部参考时钟获得。FLL 时钟由内部参考时钟控制，FLL 循

环将 FLL 频率锁定为内部参考频率的 1280 倍。内部参考时钟源已启用。

● 3.3.6.5 FLL 内部旁路低功耗（FBILP）

出现以下所有条件时，进入 FLL 内部旁路低功耗(FBILP)模式：

- ❖ 01b 写入 ICS_C1[CLKS]。
- ❖ 1b 写入 ICS_C1[IREFS]。

在“FLL 内部旁路低功耗”模式下，ICSOUT 时钟从内部参考时钟获得，FLL 禁用。内部参考时钟源已启用。

● 3.3.6.6 FLL 外部旁路（FBE）

出现以下所有条件时，进入 FLL 外部旁路(FBE)模式：

- ❖ 10b 写入 ICS_C1[CLKS]。
- ❖ 0b 写入 ICS_C1[IREFS]。
- ❖ 对 ICS_C1[RDIV]和 OSC_CR[RANGE]进行写操作可将外部参考时钟分频到 31.25kHz 至 39.0625kHz 范围内。

在 FLL 外部旁通模式下，ICSOUT 时钟由外部参考时钟源分频而来。FLL 时钟由外部参考时钟控制，FLL 环路将 FLL 频率锁定到外部参考频率（由 ICS_C1[RDIV]和 OSC_CR[RANGE]选择）的 1280 倍。

● 3.3.6.7 外部旁路低功耗（FBELP）

出现以下所有条件时，进入 FLL 外部旁路低功耗(FBELP)模式：

- ❖ 10b 写入 ICS_C1[CLKS]。
- ❖ 0b 写入 ICS_C1[IREFS]。

在“FLL 外部旁路低功耗”模式下，ICSOUT 时钟从外部参考时钟源获得，FLL 禁用。外部参考时钟源已启用。

● 3.3.6.8 FLL 停止模式（STOP）

只要 MCU 进入 STOP 状态，就会进入停止模式。在该模式下，所有 ICS 时钟信号都保持静态，但例外情况如下：

出现以下所有条件时，ICSIRCLK 在停止模式下无效：

- ❖ 1b 写入 ICS_C1[IRCLKEN]。
- ❖ 1b 写入 ICS_C1[IREFSTEN]。

注意：DCO 频率从预停止值变为其复位值，FLL 需要在频率稳定前重新获取锁定。时序敏感的操作在执行前，必须等待 FLL 获取时间 tAcquire。

■ 3.3.7 模式切换

ICS_C1[IREFS]可以随时更改，但实际切换到新选择的时钟是由 ICS_S[IREFST]指示。在 FLL 内部启用(FEI）和 FLL 外部启用(FEE)模式之间切换时，FLL 在切换完成后再次开始锁定。

ICS_C1[CLKS]也可以随时更改，但实际切换到新选择的时钟是由 ICS_S[CLKST]指示。如果新选择的时钟不可用，那么仍会选择之前的时钟。

注意：从 FEE、FBE 或 FBELP 切换到 FEI 模式时，建议等待 IREFST 切换完成，然后再更改 ICS_C1[CLKS]。

■ 3.3.8 总线分频器

ICS_C2[BDIV]可以随时更改，并且实际转换到新频率会立即发生。

■ 3.3.9 低功耗字段用途

ICS_C2 寄存器中的低功耗(LP)字段在 FLL 不使用时可将其禁用从而降低功耗。然而，在某些应用中，可能需要使能 FLL 并锁定以实现最高精度，然后再切换到 FLL 启用模式。为此，将 0b 写入 ICS_C2[LP]。

■ 3.3.10 内部基准时钟

ICS_C1[IRCLKEN]置位时，内部基准时钟信号显示为 ICSIRCLK，它可以用作附加时钟源。要重新设定 ICSIRCLK 频率，请将新值写入 ICS_C3[SCTRIM]位来调整内部基准时钟的周期：

- 写入较大的值会降低 ICSIRCLK 频率。
- 将较小的值写入 ICS_C3 寄存器会提高 ICSIRCLK 频率。

ICS 处于 FLL 内部启用(FEI)、FLL 内部旁路(FBI)或 FLL 内部旁路低功耗(FBILP)模式时，TRIM 位会影响 ICSOUT 频率。

在调整 ICSIRCLK 之前，写入较低的总线分频(ICS_C2[BDIV])系数可能导致 ICSOUT 频率超过最大芯片级频率，违反芯片级时钟的时序规范。

如果 ICS_C1[IREFSTEN]置位且 1b 写入 ICS_C1[IRCLKEN]，那么内部基准时钟在停止模式下继续运行，以便能在退出停止模式时快速恢复。

所有 MCU 器件都利用保留存储器位置中的调整值进行出厂编程。该值在任何复位初始化期间上传到 ICS_C3 寄存器和 ICS_C4[SCFTRIM]。为了提高精度，在应用中调整内部振荡器并相应地设置 ICS_C4[SCFTRIM]。

■ 3.3.11 固定频率时钟

ICS 提供经过分频的 FLL 基准时钟作为 ICSFFCLK，以便用作附加时钟源。ICSFFCLK 频率不得大于 ICSOUT 频率的 1/4，否则无效。由于这一要求，在旁路模式下，ICSFFCLK 仅对满足以下条件（ICS_C2[BDIV]、ICS_C1[RDIV]分频比和 OSC_CR[RANGE]值）的外部旁路模式（FBE 和 FBELP）有效：

如果 OSC_CR[RANGE]为高电平，

- ICS_C2[BDIV] = 000，ICS_C2[RDIV] ≥ 010
- ICS_C2[BDIV] = 001（2 分频），ICS_C2[RDIV] ≥ 011
- ICS_C2[BDIV] = 010（4 分频），ICS_C2[RDIV] ≥ 100
- ICS_C2[BDIV] = 011（8 分频），ICS_C2[RDIV] ≥ 101

■ 3.3.12 FLL 锁定和时钟监控器

● 3.3.12.1 FLL 时钟锁定

在 FBE 和 FEE 模式下，锁定检测器源使用外部基准作为基准。当检测到 FLL 从锁定变为解锁时，ICS_S[LOLS]置位。如果 ICS_C4[LOLIE]置位，则会产生中断。ICS_S[LOLS]在置位时通过复位或写入逻辑 1

清零。将逻辑 0 写入 ICS_S[LOLS]无效。

在 FBI 和 FEI 模式下，锁定检测器源使用内部基准作为基准。当检测到 FLL 从锁定变为解锁时，ICS_S[LOLS]置位。如果 ICS_C4[LOLIE]置位，则会产生中断。ICS_S[LOLS]在置位时通过复位或写入逻辑 1 清零。将逻辑 0 写入 ICS_S[LOLS]无效。

在 FBELP 和 FBILP 模式下，FLL 不开启，因此锁定检测功能不适用。

● 3.3.12.2 外部基准时钟监控器

在 FBE、FEE 或 FBELP 模式下，如将 1 写入 ICS_C4[CME]，则时钟监控器使能。如果外部基准下降低于某个特定频率，那么 MCU 将复位。SIM_SRSID[LOC]将置位以指示错误。

■ 3.3.13 初始化/应用信息

本节通过代码示例为用户介绍如何初始化和配置 ICS 模块。软件示例采用 C 语言实施。

● 3.3.13.1 初始化 FEI 模式

以下代码段显示如何将 ICS 设置为 FEI 模式。

示例: 3.3.13.1 FEI 模式初始化例程

```
/* the following code segment demonstrates setting ICS to FEI mode generating 36MHz bus*/
ICS_C2 = 0x00; /*BDIV=0, no prescalar
ICS_C1 = 0x04; /* internal reference clock to FLL */
ICS_C3 = TRIM_VALUE_35.15625KHZ; /* FLL output 36MHz */
/* the following code segment demonstrates setting ICS to FEI mode generating 4.5MHz bus*/
ICS_C2 = 0x60; /*BDIV=3, prescalar = 8 */
ICS_C1 = 0x04; /* internal reference clock to FLL */
ICS_C3 = TRIM_VALUE_35.15625KHZ; /* FLL output 4.5MHz */
```

● 3.3.13.2 初始化 FBI 模式

以下代码段显示如何将 ICS 设置为 FBI 模式。

示例: 3.3.13.2 FBI 模式初始化例程

```
/* the following code segment demonstrates setting ICS to FBI mode generating 32768Hz
bus*/
ICS_C2 = 0x00; /*BDIV=0, no prescalar
ICS_C1 = 0x44;
ICS_C3 = TRIM_VALUE_32K768HZ;
```

● 3.3.13.3 初始化 FEE 模式

以下代码段显示如何将 ICS 设置为 FEE 模式。

示例: 3.3.13.3 FEE 模式初始化例程

```
/* the following code segment demonstrates setting ICS to FEE mode generating 32.768MHZ bus*/
/* supposing external 32.768K crystal is installed */
```

```
OSC_CR = 0xB0; /* enable oscillator with low power mode */
ICS_C2 = 0x00; /* BDIV=0, no prescalar/
while ((OSC_CR & OSC_CR_OSCINIT_MASK) == 0); /* waiting until oscillator is ready */
ICS_C1 = 0x80; /* external clock reference (32.768kHz) to FLL, RDIV = 0, external prescalar
= 0 */
```

● 3.3.13.4 初始化 FBE 模式

以下代码段显示如何将 ICS 设置为 FBE 模式。

示例: 3.3.13.4 FBE 模式初始化例程

```
/* the following code segment demonstrates setting ICS to FBE mode generating 20MHZ bus*/
/* supposing external 20MHZ crystal is installed */
OSC_CR = 0xA6; /* enable clock in high range, high gain mode */
ICS_C2 = 0x00; /* BDIV=0, no prescalar*/
ICS_C1 = 0x80; /* external clock reference (20MHZ) to FLL output */
```

第 4 章 存储器模块

■ 4.1 简介

本章对基于 ARM Cortex-M0+内核的 NV32F100x 系列 MCU 存储器相关的模块做了详细说明，其中包括 Flash、SRAM 等模块内容。

■ 4.2 Flash 存储器模块

■ 4.2.1 简介

Flash 存储器非常适合单一电源供电系统应用，允许现场在线编程，无需任何外部高压电源即可进行编程或擦除操作。Flash 模块包含一个存储器控制器，用于执行命令以修改 Flash 内容。与存储器控制器连接的用户接口包含 Flash 通用命令对象(EFM)寄存器，通过填入执行命令，全局地址，数据等参数对该寄存器进行写入操作。存储器控制器必须先完成某个命令的执行，然后才能以新的命令对 EFM 寄存器进行写入操作。

警告：在编程之前，Flash 字节或长字必须处于已擦除状态。不允许对 Flash 中已经编程（写 0）的区域重复编程（写 0）。

按字节对 Flash 存储器进行读取操作。读取访问时间是字节的一个总线周期。对于 Flash 存储器，已擦除的位读取为 1，已编程的位读取为 0。

■ 4.2.2 特性

Flash 存储器包含以下特性：

- 带有校验功能的自动化编程，擦除算法
- 快速扇区擦除和长字程序操作
- 灵活的保护方案以防止意外的编程和擦除操作
- 无需外部高电压电源即可进行 Flash 存储器的编程和擦除操作
- 编程完成时可产生中断
- 利用加密机制防止未经授权访问 Flash 存储器

■ 4.2.3 功能说明

● 4.2.3.1 操作模式

嵌入式存储器模块（EFM）提供正常的用户操作模式。操作模式取决于模块级的输入，同时会影响 EFM_STAT 寄存器和 EFM_TIM0/EFM_TIM1 寄存器。

- 等待模式

如果 MCU 进入等待模式，EFM 存储器模块不受影响。EFM 模块可通过 CCIF 中断从等待模式中恢复 MCU。

- 停止模式

如果在 MCU 请求停止模式时 EFM 模块命令处于活动状态（(CCIF = 0)），则 Flash 控制器会在允许 MCU 进入停止模式之前完成当前的 Flash 操作。

● 4.2.3.2 Flash 存储器映射

NV32F100x 包含一个 64/128KB Flash 存储器。该 Flash 数据块分为 256 个 512 字节的扇区，此外还包括十个非易失性寄存器扇区（NVR）（512 Bytes/个），和八个冗余扇区（512Bytes/个），扇区擦除操作将删除扇区内所有的字节（512 Bytes）。

表 4-1 NV32F100x Flash 存储器大小

型号	全局地址	大小	说明
NV32F100A NV32F100B NV32F100C	0x0000_0000-0x0000_FFFF	64KB	Flash 数据块包含 Flash 配置字段。
NV32F100D NV32F100E NV32F100F	0x0000_0000-0x0001_FFFF	128KB	Flash 数据块包含 Flash 配置字段。

● 4.2.3.3 系统复位之后 Flash 初始化

系统复位后,Flash 模块会执行初始化过程,此序列将为Flash数据块配置参数。如果在执行任何Flash命令的同时发生复位，则此命令立即中止。对于正在编程的字或正在擦除的扇区/区块，其状态无法得到保证。

● 4.2.3.4 Flash 命令操作

Flash 命令操作用于修改 Flash 存储器内容。

编程 Flash 包含以下三个步骤：

1. 为 Flash 存储器模块配置时钟。
2. 使用命令写入序列设置 Flash 命令参数和启动执行程序。
3. 根据 MCU 功能模式和 MCU 安全状态执行校验的 Flash 命令。

■ 4.2.4 存储器映射和寄存器说明

Flash 模块的寄存器能够以 32 位、16 位（对齐到[31:16]数据或[15:0]数据）或 8 位存取。如果是可写的寄存器，在 Flash 命令执行期间将会禁止写入访问。写保留地址位置没有影响并读取返回 0。

表 4-2 Flash 存储器映射

寄存器绝对地址(16 进制)	寄存器名称	偏移量	位宽	权限	复位值
0x4002_0000h	Flash 配置寄存器(EFM_CR)	0h	32	R/W	0x1400_0007
0x4002_0004h	Flash 安全读取寄存器 0	4h	32	R/W	*
0x4002_0008h	Flash 安全读取寄存器 1	8h	32	R/W	*
0x4002_000Ch	Flash 安全读取寄存器 2	Ch	32	R/W	*
0x4002_0010h	Flash 定时寄存器 0（EFM_TIM0）	10h	32	R/W	00h
0x4002_0014h	Flash 定时寄存器 1（EFM_TIM1）	14h	32	R/W	00h
0x4002_0018h	Flash 状态寄存器（EFM_STAT）	18h	8	R/W	C2h
0x4002_001Ah	Flash 命令寄存器（EFM_CMD）	1Ah	8	R/W	00h

● 4.2.4.1 Flash 配置寄存器（EFM_CR）

EFM 配置寄存器（EFM_CR）是不分组的用于配置和控制 EFM 阵列和总线接口单元的操作（BIU）。

地址：0x4002_0000h(基址) + 00h(偏移量) = 4002_0000h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0		DIVSEL[5:0]						DONEIE	0					VREAD0	VREAD1
写																
复位	0	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0	CEBATS[2:0]			0	OEBATS[2:0]			0				WS[3:0]			
写																
复位	0	0	0	1	0	0	0	1	0	0	0	0	0	1	1	1

表 4-3 EFM_CR 字段描述

位	名称	描述
31-30	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
29-24	DIVSEL{5:0]	时钟分频选择 EFM 分时钟频率= ipg_clk(总线时钟)频率/(DIVSEL [5:0]+1)。DIVSEL [5:0]=0 时， EFM 分时钟频率=ipg_clk 频率。最低可以分频到 1MHz 频率。
23	DONEIE	命令完成中断使能位 Flash 物理块操作命令执行结束时，是否使能中断。 0 中断使能无效 1 中断使能有效
22-18	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
17	VREAD0	VREAD0 Flash 功能使能 0 VREAD0 Flash 功能无效 1 VREAD0 Flash 功能有效
16	VREAD1	VREAD1 Flash 功能使能 0 VREAD1 Flash 功能无效 1 VREAD1 Flash 功能有效
15	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
14-12	CEBATS[2:0]	CEB 信号使能选择 0 选择无效 1 选择有效
11	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
10-8	OEBATS[2:0]	OEB 信号使能选择 0 选择无效

		1 选择有效
7-4	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
3-0	WS[3:0]	等待状态位 Flash 地址匹配时，RWS 场决定等待状态的次数。

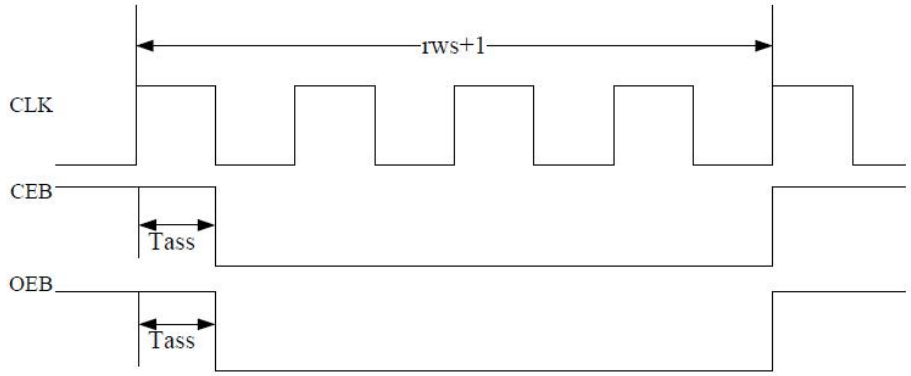


图 4-1 CEB/OEB 有效时序图

表 4-4 断言和否定时间 (unit : 系统周期)

CEB[2:0]/OEB[2:0]	Assert Timer (Tass)
000	0
001	1/2
010	2/2
011	3/2
100	4/2
101	5/2
110	6/2
111	7/2

● 4.2.4.2 EFM 安全读取寄存器 0（EFM_SEC0）

EFM 安全读取寄存器 0（EFM_SEC0）用于存储第一和第二 RDN 信息。

地址：0x4002_0000h(基址) + 04h(偏移量) = 4002_0004h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0							RDN1EN	RDN1INFO[7:0]							
写																
复位	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0							RDN2EN	RDN2INFO[7:0]							
写																
复位	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

*注：复位值由 NVR5 阵列中的 RDN 信息决定。

表 4-5 EFM_SEC0 字段描述

位	名称	描述
31-25	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
24	RDN1EN	RDN1 第一冗余扇区是否用于修复坏扇区 0 RDN1 修复功能未启用 1 RDN1 修复功能启用
23-16	RDN1INFO[7:0]	这 8 位信息显示需要 RDN1 修复的坏扇区地址。
15-9	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
8	RDN2EN	RDN2 第二冗余扇区是否用于修复坏扇区 0 RDN2 修复功能未启用 1 RDN2 修复功能启用
7-0	RDN2INFO[7:0]	这 8 位信息显示需要 RDN2 修复的坏扇区地址。

● 4.2.4.3 EFM 安全读取寄存器 1（EFM_SEC1）

EFM 安全读取寄存器 1（EFM_SEC1）用于存储第三和第四 RDN 信息。

地址：0x4002_0000h(基址) + 08h(偏移量) = 4002_0008h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0							RDN3EN	RDN3INFO[7:0]							
写																
复位	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0							RDN4EN	RDN4INFO[7:0]							
写																
复位	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

*注：复位值由 NVR5 阵列中的 RDN 信息决定。

表 4-6 EFM_SEC1 字段描述

位	名称	描述
31-25	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
24	RDN3EN	RDN3 第三冗余扇区是否用于修复坏扇区 0 RDN3 修复功能未启用 1 RDN3 修复功能启用

23-16	RDN3INFO[7:0]	这 8 位信息显示需要 RDN3 修复的坏扇区地址。
15-9	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
8	RDN4EN	RDN4 第四冗余扇区是否用于修复坏扇区 0 RDN4 修复功能未启用 1 RDN4 修复功能启用
7-0	RDN4INFO[7:0]	这 8 位信息显示需要 RDN4 修复的坏扇区地址。

● 4.2.4.4 EFM 安全读取寄存器 2（EFM_SEC2）

EFM 安全读取寄存器 2（EFM_SEC2）用于存储第五 RDN 信息。

地址：0x4002_0000h(基址) + 0Ch(偏移量) = 4002_000Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0							RDN5EN	RDN5INFO[7:0]							
写																
复位	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0															
写																
复位	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*

*注：复位值由 NVR5 阵列中的 RDN 信息决定。

表 4-7 EFM_SEC2 字段描述

位	名称	描述
31-25	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
24	RDN5EN	RDN5 第三冗余扇区是否用于修复坏扇区 0 RDN5 修复功能未启用 1 RDN5 修复功能启用
23-16	RDN5INFO[7:0]	这 8 位信息显示需要 RDN1 修复的坏扇区地址。
15-0	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

● 4.2.4.5 EFM 定时寄存器 0（EFM_TIM0）

EFM 定时寄存器 0（EFM_TIM0）用于配置编程和擦除算法中定时事件的定时参数。

地址：0x4002_0000h(基址) + 10h(偏移量) = 4002_0010h

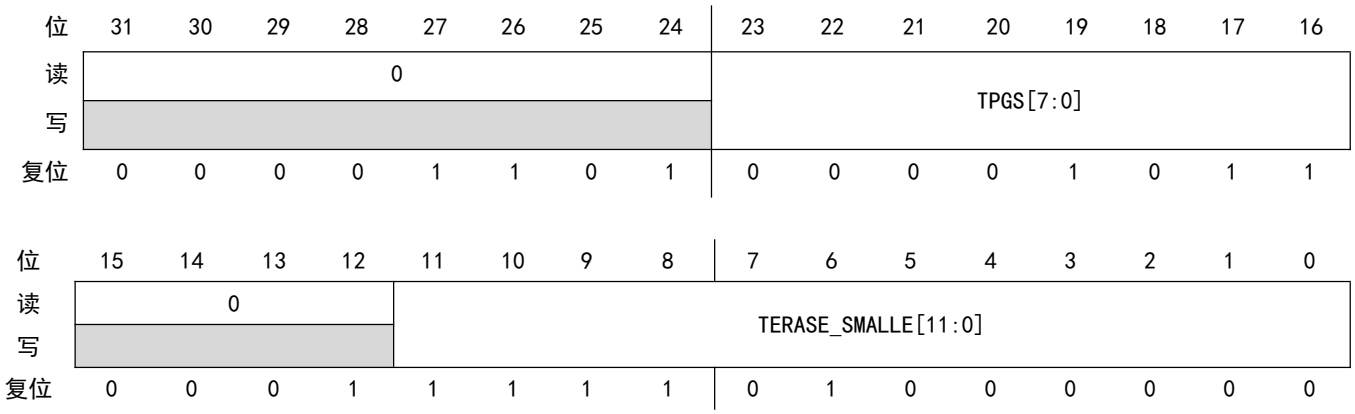


表 4-8 EFM_TIM0 字段描述

位	名称	描述
31-24	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
23-16	TPGS[7:0]	TPGS 周期计数 配置等待周期满足 WEB 低到 PROG 低的建立时间。WEB 低到 PROG2 高的建立时间应该在 2.5～3.2μs. TPGS 计数周期基于 ipg_clk. TPGS 可以设置为： TPGS =（2.5～3.2μs）/ipg_clk 周期
15-12	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
11-0	TERASE_SMALLE[11:0]	扇区/块擦除 Terase 周期计数。 配置等待周期满足扇区擦除时间。扇区擦除时间应该 4～5ms. 所以 TERASE_SMALL 可以这样设置为：TERASE_SMALL = (4～5ms)/The EFM 分频时钟周期（EFM divide clock period.）

● 4.2.4.6 EFM 定时寄存器 1（EFM_TIM1）

EFM 定时寄存器 1（EFM_TIM1）用于配置编程和擦除算法中定时事件的定时参数。

地址：0x4002_0000h(基址) + 14h(偏移量) = 4002_0014h

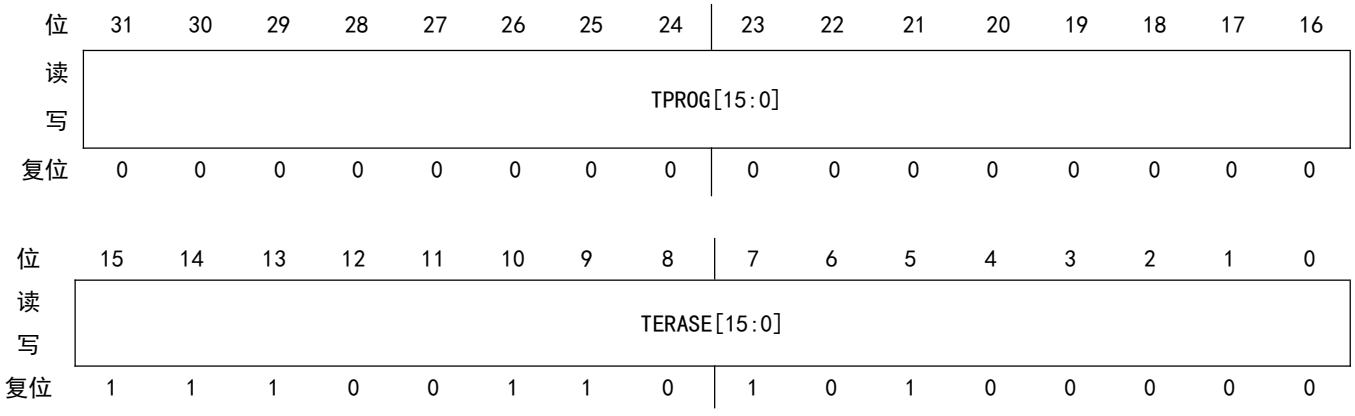


表 4-9 EFM_TIM1 字段描述

位	名称	描述

31-16	TPROG[15:0]	Tprog 周期计数 配置等待周期以满足字节编程时间。编程时间应在 6～7.5μs。TPROG 计数周期基于 ipg_clk。所以 TPROG 可以设置为： TPROG = (6～7.5μs)/ipg_clk 周期
15-0	TERASE[15:0]	Terase 周期计数 配置等待周期满足芯片擦除时间。最短的芯片擦除时间是 30ms，最长的芯片擦除时间是 40ms。所以 TERASE 可以设置为: TERASE= (30ms～40ms)/The EFM 分频时钟周期

● 4.2.4.7 EFM 状态寄存器（EFM_STAT）

地址：0x4002_0000h(基址) + 18h(偏移量) = 4002_0018h

位	7	6	5	4	3	2	1	0
读	0	DONE	CCIF	ACCERR	0			
写								
复位	0	0	1	0	0	0	1	0

表 4-10 EFM_STAT 字段描述

位	描述
7 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
6 DONE	命令完成中断标志 DONE 标志表示，没有命令操作闪存物理块。完成命令后 DONE 自动置位(set)。写入 1，DONE 被清零。如果 EFM_CR 寄存器中的 DONEIE 位被置 1，DONE 位能触发中断请求。 1 所有命令已完成 0 无效
5 CCIF	命令完成指示标志 CCIF 标志表示没有命令运行操作闪存物理块。CCIF 基于命令的开始和结束自动置位和清零，写入 CCIF 无效。 1 所有的命令已完成 0 命令运行中
4 ACCERR	访问错误标志 显示非法访问 EFM 阵列或由错误程序或擦除顺序所致的寄存器。写入 1， ACCERR 清零。写入 0 到 ACCERR 无效。 1 访问错误 0 无故障
3-0 保留	此字段为保留字段。 此只读字段为保留字段，读取值为 0。

● 4.2.4.8 EFM 命令寄存器（EFM_CMD）

地址：0x4002_0000h(基址) + 1Ah(偏移量) = 4002_001Ah

位	7	6	5	4	3	2	1	0
读	0	CMD6	CMD5	CMD4	CMD3	CMD2	CMD1	CMD0
写								
复位	0	0	0	0	0	0	0	0

表 4-11 EFM_CMD 字段描述

位	名称	描述
7	保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
6-0	CMD[6:0]	Flash 操作命令 0x20 编程　　只能为 32bit 编程 0x40 块擦除　　擦除 1 块 Flash(512 字节) 0x41 整片擦除　　擦除整个 Flash 主要存储区

■ 4.2.5 编程和擦除功能说明

● 4.2.5.1 设置 EFM_TIM0/1 寄存器

在启动任何编程或擦除命令之前，EFMTIM0/1 必须被写入，以设置在程序和擦除操作中的相应参数。

● 4.2.5.2 编程、擦除和验证序列

这三步命令的写顺序必须严格遵循。这三个步骤中，中间写入 EFM 模块是不被允许的。命令写入序列为：

1 读 CCIF 位，如果 CCIF 位被置位，地址，数据和命令都准备好开始新命令序列。

2 写入待编程的字节/半字/字到 EFM 阵列的位置。位置和数据将被存储在内部缓冲区。所有地址位是有效的程序命令。用于验证和擦除命令的写入数据的值被忽略。对于芯片擦除或验证，地址可以在 EFM 阵列的任何位置。对于扇区擦除，地址位[8:0]被忽略。

3 写入有效命令到 EFMCMD 启动命令。当命令启动时，CCIF 将自动被清除。

当命令执行完成后，FLASH 状态机将被设置成 CCIF 和 DONE 标志，表示该地址、数据和命令准备开始新命令序列。在开始另一个命令写入序列之前，DONE 标志应该被清零。

通过在 EFMSTAT 寄存器标志 ACCERR，FLASH 状态机将标志写入序列命令的错误。在开始另一个命令写入序列前，ACCERR 标志必须被清零。

● 4.2.5.3 Flash 非法操作

在命令写入序列，如果有以下任何非法操作，ACCERR 标志将被设置。

1 当命令序列忙(CCIF = 0)时，EFM 的读写将失效并被设置为 ACCERR；

2 当命令序列忙(CCIF = 0)时，写入新命令到 EFMCMD 将失效并被设置为 ACCERR。命令寄存器将不被更新。

3 当不被允许时，程序或扇区擦除 NVR 将失效，并被设置为 ACCERR；

4 当使用密码且输入密码不正确时，程序或扇区擦除 NVR 将失效并被设置为 ACCERR；

5 试着编程或扇区擦除 NVR11/NVR12 将失效并被设置成 ACCERR。

注：如果使用禁用(disable)NVR 编程/擦除功能或编程/擦除 NVR5。

■ 4.2.6 使用示例程序

● 4.2.6.1 初始化 Flash

```
uint16_t Flash_Init(void)
{
    uint16_t err   = FLASH_ERR_SUCCESS;
    uint8_t clkDIV = BUS_CLK_HZ/1000000L - 1;
    uint8_t Tpgs   =  (285 *(BUS_CLK_HZ/100))/1000000L;
    uint16_t Tprog =  (675 *(BUS_CLK_HZ/100))/1000000L;

    EFMCR=(clkDIV<<24) + 0x00000001; //divide to 1M hz
    EFMTIM0=(Tpgs<<16)  + 0x00001194;
    EFMTIM1=(Tprog<<16) + 0x000088B8;

  return(err);
}
```

对 3 个擦写相关的寄存器进行初始化，保证擦写 flash 的时序的正确性。

● 4.2.6.2 Flash 的擦写

先查看是否还有命令在执行，然后发送命令，然后等待命令结束。

```
 int EraseChip (void) {
     unsigned long  tmp;
     tmp = FTMRH->EFMCMD;
     FTMRH->EFMCMD =0x00005000; //清除命令
 if((tmp&(0x00002000)) ==0x00002000) //确认没有命令在运行
   {
   FTMRH->EFMCMD= 0x41000000;//0x41 发送擦除整个芯片的命令，0x40 为发送擦除 sector 的命令
   }
   while(1){
     tmp=  FTMRH->EFMCMD;
     if ((tmp &0x41006000) == 0x41006000) break;  //确认命令运行结束
   }
```

■ 4.3 SRAM 模块

■ 4.3.1 简介

NV32F100x系列型号片上内置了8K SRAM，分为两个范围：2K分配为SRAM_L（0x1FFF_F800-0x1FFF_FFFF）；6K分配为SRAM_U区（0x2000_0000-0x2000_17FF）。

■ 4.3.2 存储器映射

RAM 存储器映射如下图所示：

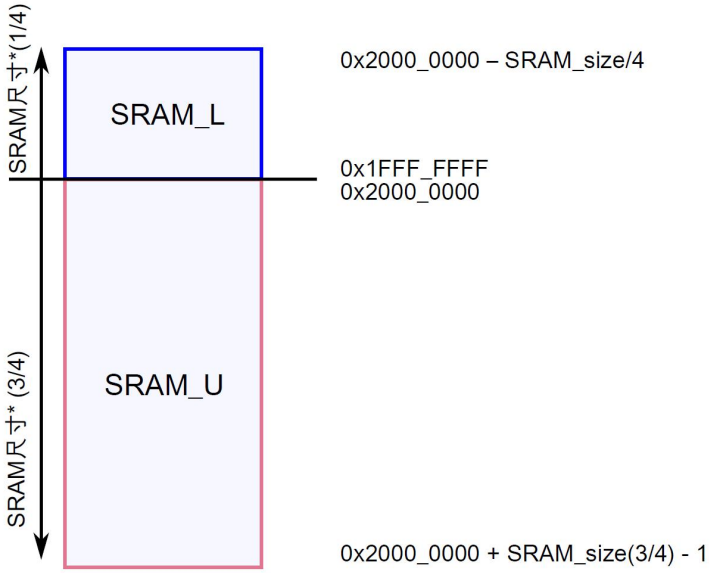


图4-2 RAM存储器映射图

■ 4.3.3 SRAM 位操作

SRAM_U范围通过两种方式支持该器件上的位操作：

- 位带区别名
- 位操作存储(BOS)

该位带区别名中的32位写操作与对SRAM_U区中目标位进行的读-修改-写操作的作用相同，只需一个周期时间，位带区别名仅支持简单置位或清零操作。通过BOS模块引擎对外设和SRAM_U地址空间可进行更复杂的位操作(AND、OR、XOR等)，通过将Cortex-M指令集架构中的基本负载和存储指令与BOS提供的位操作存储指令相结合，生成的存储操作将为此类超低端控制器提供强大和高效的“读取-修改-写入”的功能。关于BOS详细信息请参考BOS章节。

第 5 章 模拟模块

■ 5.1 简介

本章对基于 ARM Cortex-M0+内核的 NV32F100x 系列 MCU 模拟相关的模块做了详细说明，其中包括数模转换器 ADC、模拟比较器模块 ACMP 等模块内容。

■ 5.2 模数转换器(ADC)

■ 5.2.1 简介

12位模数转换器(ADC)是一种特别设计用于集成微控制器片上系统操作的逐次逼近型ADC。

■ 5.2.2 结构框图

下图是 ADC 模块的结构框图：

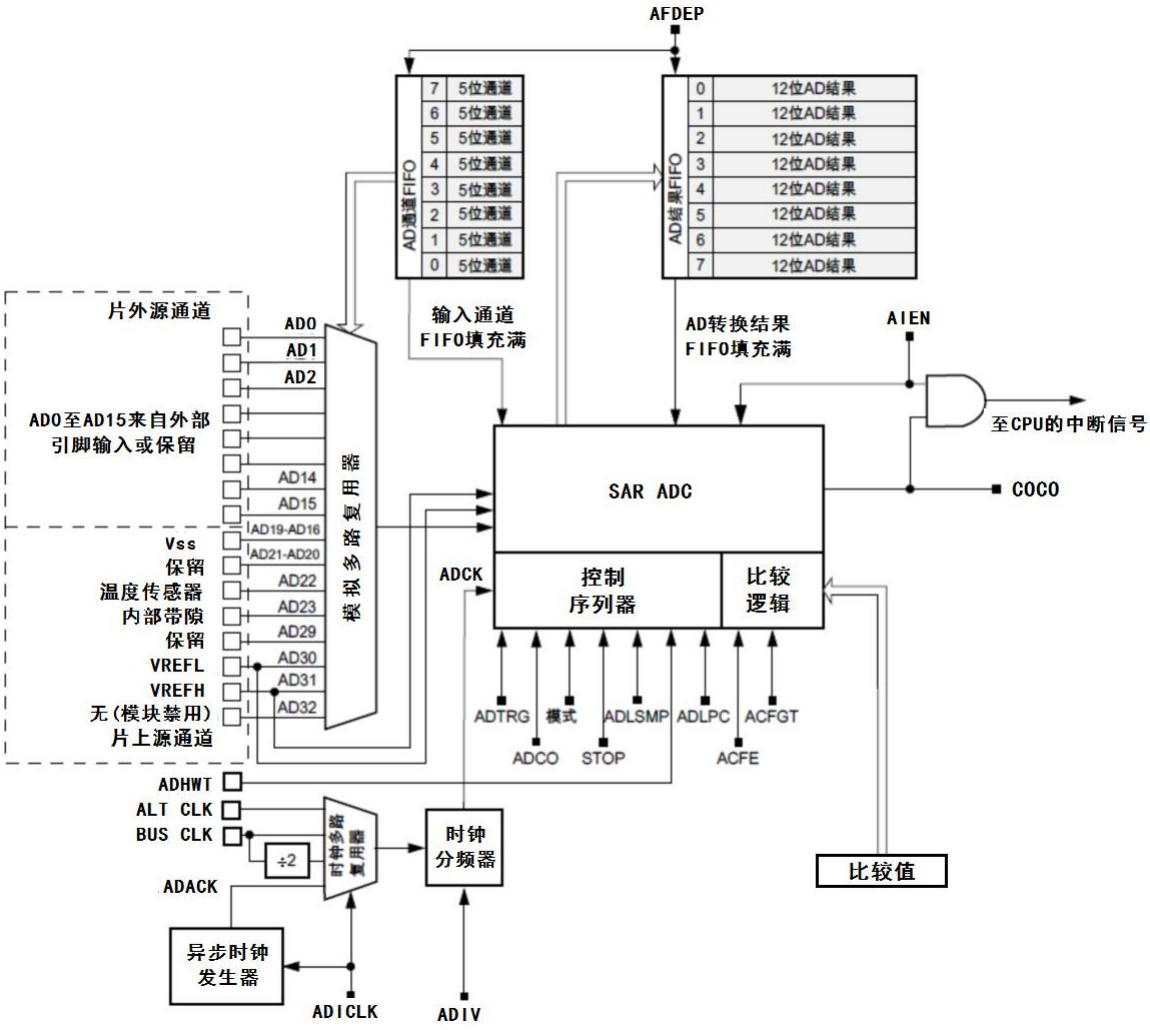


图 5-1 ADC 结构框图

■ 5.2.3 特性

ADC模块特性包括：

- 采用8位、10位或12位分辨率的线性逐次逼近算法
- 多达16个外部模拟输入、外部引脚输入以及5个内部模拟输入，包括内部带隙、温度传感器和参考电压
- 8位、10位或12位的右对齐无符号格式输出
- 单次或连续转换（单次转换后自动返回到空闲状态）
- 最多支持8个结果FIFO，可选择FIFO深度
- 可配置采样时间和转换速度/功耗
- 转换完成标志和中断
- 可从最多4个时钟源选择输入时钟
- 在等待或停止模式下工作，可降低操作噪声
- 异步时钟源，可降低操作噪声
- 可选择的异步硬件转换触发信号
- 自动与设定值进行比较（小于、大于或等于），根据结果产生中断

■ 5.2.4 外部信号说明

ADC 模块支持多达 24 个单独的模拟输入。它还要求四个电源/参考/接地连接。

表 5-1 信号属性

名称	功能
AD23-AD0	模拟通道输入
VREFH	高参考电压
VREFL	低参考电压
VDDA	模拟电源
VSSA	模拟接地

● 5.2.4.1 模拟电源(VDDA)

ADC 模拟部分将VDDA 用作其电源连接。在某些封装中，VDDA内部连接到VDD。如果可以外部连接，则将VDDA引脚连接到与VDD相同的电压电势。可能必须进行外部滤波确保VDDA干净以便获得良好的结果。

● 5.2.4.2 模拟接地(VSSA)

ADC 模拟部分将VSSA 用作其电源连接。在某些封装中，VSSA内部连接到VSS。如果可以外部连接，则将VSSA引脚连接到与VSS相同的电压电势。

● 5.2.4.3 高参考电压(VREFH)

VREFH 是转换器的高参考电压。在某些封装中， VREFH 内部连接到VDDA。如果可从外部连接，可将VREFH连接到与VDDA相同的电势，或由介于数据表中指定的最小VDDA与VDDA电势之间的外部电压源驱动（VREFH不能超过VDDA）。

● 5.2.4.4 低参考电压(VREFL)

VREFL是转换器的低参考电压。在某些封装中，VREFL内部连接到VSSA。如果可以外部连接，则将VREFL引脚连接到与VSSA相同的电势。

● 5.2.4.5 模拟通道输入(ADx)

ADC模块支持多达24个单独的模拟输入。通过ADCH通道选择位选择某个输入进行转换。

■ 5.2.5 存储器映射和寄存器说明

表 5-2 ADC 存储器映射

绝对地址 (16 进制)	寄存器名称	偏移量	位宽	权限	复位值
4003_B000h	状态和控制寄存器 1（ADC_SC1）	00h	32	R/W	0000_001Fh
4003_B004h	状态和控制寄存器 2（ADC_SC2）	04h	32	R/W	0000_0008h
4003_B008h	状态和控制寄存器 3（ADC_SC3）	08h	32	R/W	0000_0000h
4003_B00Ch	状态和控制寄存器 4（ADC_SC4）	0Ch	32	R/W	0000_0000h
4003_B010h	转换结果寄存器（ADC_R）	10h	32	R	0000_0000h
4003_B014h	比较值寄存器（ADC_CV）	14h	32	R/W	0000_0000h
4003_B018h	引脚控制 1 寄存器（ADC_APCTL1）	18h	32	R/W	0000_0000h

● 5.2.5.1 状态和控制寄存器 1(ADC_SC1)

本节介绍ADC状态和控制寄存器(ADC_SC1)的功能。对ADC_SC1进行写操作会中止当前转换并在ADCH位非全1时启动新的转换。

FIFO使能时，通过ADCH对模拟输入通道FIFO进行写操作。模拟输入通道队列必须连续写入ADCH。生成的FIFO顺序与写入模拟输入通道的顺序相同。当输入通道FIFO中填充的内容达到ADC_SC4[AFDEP]指示的深度时，ADC开始转换。在输入通道FIFO有效时将0x1F写入这些位就会使FIFO复位并停止转换。

地址: 4003_B000h 基准 + 0h 偏移 = 4003_B000h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0								COCO	AIEN	ADCO	ADCH				
写																
复位	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1

表 5-3 ADC_SC1 字段描述

字段	描述
31-8 保留	此字段为保留字段 此只读字段为保留字段且值始终为 0
7 COCO	转换完成标志 转换完成标志。COCO 标志是一个只读位，在每次转换完成时置位，此时比较功能禁用（ADC_SC2[ACFE]=0）。如果比较功能使能（ADC_SC2[ACFE]=1），则仅当比较结果为真时，COCO 标志才会在转换完成时置位。当使能 FIFO 功能时(ADC_SC4[AFDEP]>0),COCO 标志在 FIFO 转换全部完成时置位。对 ADC_SC1 进行写操作或对 ADC_R 进行读操作时，该位清零。 0 转换未完成 1 转换已完成
6 AIEN	中断使能 AIEN 使能转换完成中断。如果在 COCO 置位的同时 AIEN 为高电平，那么会产生中断。 0 转换完成中断禁用 1 转换完成中断使能
5 ADCO	连续转换使能 ADCO 使能连续转换。 0 选择软件触发操作时，对 ADC_SC1 进行写操作后启动一次转换；或选择硬件触发操作时，ADHWT 的电平变为有效值后启动一次转换。使能 FIFO 功能时（AFDEP>0），就会触发一组转换。 1 选择软件触发操作时，对 ADC_SC1 进行写操作后启动连续转换。选择硬件触发操作时，由 ADHWT 事件启动连续转换。FIFO 功能使能时（AFDEP>0），循环触发一组转换。
ADCH	输入通道选择 ADCH 位形成一个 5 位字段，用于选择一个输入通道。 00000-01111 AD0-AD15 10000-10011 Vss 10100-10101 保留位 10110 温度传感器 10111 带隙 11000-11100 保留位 11101 VREFH 11110 VREFL 11111 模块已禁用 注：在 FIFO 模式下复位 FIFO

● 5.2.5.2 状态和控制寄存器 2(ADC_SC2)

ADC_SC2 寄存器控制ADC模块的比较功能、转换触发和转换有效状态。

地址: 4003_B000h 基准 + 4h 偏移 = 4003_B004h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								ADACT	ADTRG	ACFE	ACFGT	FEMPTY	FFULL	REFSEL	
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0

表 5-4 ADC_SC2 字段描述

字段	描述
31-8 保留	此字段为保留字段 此只读字段为保留字段且值始终为 0
7 ADACT	转换有效 指示转换正在进行。ADACT 在转换启动时置位，在转换完成或中止时清零。 0 转换未在进行 1 转换正在进行
6 ADTRG	转换触发选择 选择用于启动转换的触发类型。有两类触发可选择：软件触发和硬件触发。选择软件触发时，对 ADC_SC1 进行写操作后启动转换。选择硬件触发时，ADHWT 输入的电平变为有效值后启动转换。 0 选择软件触发 1 选择硬件触发
5 ACFE	比较功能使能 0 比较功能禁用 1 比较功能使能
4 ACFGT	比较功能大于使能 将比较功能配置为在受监控的输入转换结果大于或等于比较值时触发。比较功能默认为在受监控的输入转换结果小于比较值时触发。 0 输入小于比较电平时，比较触发。 1 输入大于或等于比较电平时，比较触发。
3 FEMPTY	结果 FIFO 空 0 指示 ADC 结果 FIFO 至少有一个有效新数据。 1 指示 ADC 结果 FIFO 没有有效新数据。
2 FFULL	结果 FIFO 满 0 指示 ADC 结果 FIFO 未满，下一转换数据仍可存储在 FIFO 中。 1 指示 ADC 结果 FIFO 未满，如果不读取 FIFO，下一转换数据将覆盖旧数据。
1-0 REFSEL	基准电压源选择 选择用于转换的基准电压源。 00 默认基准电压引脚对（VREFH/VREFL）。 01 模拟供电引脚对（VDDA/VSSA）。 10 保留位。 11 保留位-选择默认基准电压（VREFH/VREFL）引脚对。

● 5.2.5.3 状态和控制寄存器 3(ADC_SC3)

ADC_SC3 选择操作模式、时钟源、时钟分频，并配置低功耗或长采样时间。

地址: 4003_B000h 基准 + 8h 偏移 = 4003_B008h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
R	0															
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0								ADLPC	ADIV		ADLSMP	MODE		ADICLK	
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 5-5 ADC_SC3 字段描述

字段	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
7 ADLPC	低功耗配置 ADLPC 控制逐次渐进转换器的速度和功耗配置。在无需更高采样率时，可以优化功耗。 0 高速配置 1 低功耗配置：通过降低最大时钟速度来降低功耗。
6-5 ADIV	时钟分频选择 ADIV 选择 ADC 用于生成内部时钟 ADCK 的分频比。 00 分频比=1，时钟频率 = 输入时钟。 01 分频比=2，时钟速率 = 输入时钟/2。 10 分频比=3，时钟速率 = 输入时钟/4。 11 分频比=4，时钟速率 = 输入时钟/8。
4 ADLSMP	长采样时间配置 ADLSMP 选择长或短采样时间。调整采样周期可以使高阻抗输入的采样更精确或使低阻抗输入的转换速度更快。当连续转换使能且无需高转换速率时，使用长采样时间还能降低整体功耗。 0 短采样时间。 1 长采样时间。
3-2 MODE	转换模式选择 MODE 位用于选择 12 位、10 位或 8 位操作。 00 8 位转换（N=8） 01 10 位转换（N=10） 10 12 位转换（N=12） 11 保留
1-0 ADICLK	输入时钟选择 ADICLK 位选择用于生成内部时钟 ADCK 的输入时钟源。

	00 总线时钟 01 总线时钟 2 分频 10 备用时钟（ALTCLK） 11 异步时钟（ADACK）

● 5.2.5.4 状态和控制寄存器 4(ADC_SC4)

该寄存器控制ADC模块的FIFO扫描模式、FIFO 比较功能和FIFO深度选择。

地址: 4003_B000h 基准 + Ch 偏移 = 4003_B00Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	19	17	16
R	0															
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R	0									ASCANE	ACFSEL	0		AFDEP		
W																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 5-6 ADC_SC4 字段描述

字段	描述
31 – 7 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
6 ASCANE	FIFO扫描模式使能 FIFO在使能时总是使用第一个空FIFO通道。该位置位且FIFO功能使能时，ADC将重复使用该第一个FIFO通道作为转换通道，直至结果FIFO填满。在连续模式(ADCO = 1)下，当COCO置位时，ADC将利用相同通道开始下一转换。 0 FIFO扫描模式禁用。 1 FIFO 扫描模式使能。
5 ACFSEL	比较功能选择 当FIFO功能使能时(AFDEP > 0)，比较功能选择“或”或“与”运算。当该字段清零时，ADC将对所有比较触发作“或”运算，如果至少有一个比较触发有效，就会置位COCO。当该字段置位时，ADC将对所有比较触发作“与”运算，如果所有比较触发都有效，就会置位COCO。 0 对所有比较触发作“或”运算。 1 对所有比较触发作“与”运算。
4 – 3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2-0 AFDEP	FIFO深度 FIFO深度用于使能FIFO功能和设置FIFO的深度。AFDEP清零时，FIFO禁用。AFDEP设置为非零值时，FIFO功能使能，深度由AFDEP位指示。FIFO功能使能时，FIFO模式必须访问ADC_SC1[ADCH]和ADC_R。当模拟通道FIFO位于AFDEP位指示的水平时，ADC启动转换。当转换集合完成且结果FIFO位于AFDEP位指示的水平时，COCO位置位。

	000 FIFO 禁用。 001 2 级FIFO 使能。 010 3 级FIFO 使能。 011 4 级FIFO 使能。 100 5 级FIFO 使能。 101 6 级FIFO 使能。 110 7 级FIFO 使能。 111 8 级 FIFO 使能。

● 5.2.5.5 转换结果寄存器(ADC_R)

在12位工作模式下，ADC_R包含12位转换的12位结果。在10位模式下，ADC_R包含10位转换的10位结果。在8位模式下，ADC_R包含8位转换的8位结果。除非使能了自动比较且不满足比较条件，ADC_R会在每次转换时更新。当FIFO使能时，结果FIFO通过ADC_R读取。当输入通道FIFO中填充的内容达到AFDEP指示的深度时，ADC转换即告完成。FIFO可以按照模拟输入通道ADCH设置的顺序通过ADC_R连续读取ADC转换的结果。如果MODE位改变，ADC_R中的所有数据都会变为无效。

地址: 4003_B000h 基准 + 10h 偏移 = 4003_B010h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0																ADR															
写																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 5-7 ADC_R 字段描述

字段	描述
31 - 12 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
11-0 ADR	转换结果

● 5.2.5.6 比较值寄存器(ADC_CV)

该寄存器保存比较值。12位模式下的转换完成后，位ADCV11:ADCV0与12位结果进行比较。

地址: 4003_B000h 基准 + 14h 偏移 = 4003_B014h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0																CV															
写																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 5-8 ADC_CV 字段描述

字段	描述

31-12 保留	此字段为保留字段。 此只读字段为保留字段且值始终为0。
11-0 CV	转换结果 [11:0]

● 5.2.5.7 引脚控制寄存器 1(ADC_APCTL1)

该引脚控制寄存器禁用用作模拟输入的MCU引脚的I/O端口控制。APCTL1用于控制与ADC模块的通道0-31相关的引脚。

地址: 4003_B000h 基准 + 18h 偏移 = 4003_B018h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0																ADPC															
写																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 5-9 ADC_APCTL1 字段描述

字段	描述
31-16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
15-0 ADPC	ADC引脚控制 ADPCx控制与通道ADx相关的引脚。 0 ADx引脚I/O 控制使能。 1 ADx 引脚 I/O 控制禁用。

■ 5.2.6 功能说明

ADC模块在复位期间或ADC_SC1[ADCH]位均为高电平时禁用。该模块在转换完成后以及另一个转换还未发起前处于空闲状态。空闲时，该模块处于其功耗最低状态。

ADC可在任何软件可选的通道上进行模数转换。在12位模式下，逐次逼近算法可将选定的通道电压转换成12位数字结果。在10位模式下，逐次逼近算法可将选定的通道电压转换成10位数字结果。在8位模式下，逐次逼近算法可将选定的通道电压转换成8位数字结果。

转换完成后，结果置于数据寄存器(ADC_R)中。在10位模式下，结果四舍五入为10位并置于数据寄存器(ADC_R)中。在8位模式下，结果四舍五入为8位并置于数据寄存器ADC_R中。转换完成标志(ADC_SC1[COCO])随后置位并且在转换完成中断已使能(ADC_SC1[AIEN] = 1)情况下会生成一个中断。

ADC模块能够自动将转换结果与其比较寄存器的内容作比较。比较功能在ADC_SC2[ACFE]位置位的情况下使能，而且能在任何转换模式和配置下工作。

● 5.2.6.1 时钟选择和分频控制

ADC模块的时钟源可以从四个时钟源中选择。然后用一个可配置的值将该时钟源分频，生成转换器的输入时钟(ADCK)。利用ADC_SC3[ADICLK]位从下列时钟源选择时钟：

- ❖ 总线时钟2分频：总线时钟速率较高时，最多可以对总线时钟进行16分频。
- ❖ ALTCLK，也就是备用时钟OSC_OUT
- ❖ 异步时钟(ADACK)：该时钟从ADC模块内部的时钟源生成。选择该时钟作为时钟源时，当MCU处于等待或停止模式时，它仍然有效，转换仍可进行，而且噪声很低。

无论选择哪个时钟，其频率必须在ADCK的指定频率范围内。如果可用时钟太慢，ADC将不能以额定性能工作。如果可用时钟太快，必须将其分频到合适的频率。该分频器由ADC_SC3[ADIV]位指定，可执行1、2、4或8分频。

在FIFO模式，当ADC_SC4[ASCANE]=0，总线不能慢于ADC转换时钟(ADCK)除以2. 在FIFO模式ADC_SC2[ACFE]= 1，总线时钟不能慢于ADC转换时钟(ADCK)除以2。

● 5.2.6.2 输入选择和引脚控制

引脚控制寄存器 （ ADC_APCTL1)禁用用作模拟输入的引脚的I/O端口控制。 当引脚控制寄存器位置位时，将强制使相关的MCU引脚处于以下状态：

- ❖ 输出缓冲强制进入高阻抗状态。
- ❖ 输入缓冲禁用。对于其输入缓冲被禁用的任何引脚，对I/O 端口进行读操作会返回0。
- ❖ 上拉禁用。

● 5.2.6.3 硬件触发器

ADC模块具有一个可选异步硬件转换触发器ADHWT，在ADC_SC2[ADTRG]位置位后会被启用。

如果具有ADHWT源并且已启用硬件触发器(ADC_SC2[ADTRG] = 1)，则会在ADHWT的上升沿发起转换。如果出现上升沿时转换正在进行中，则该上升沿会被忽略。在连续转换配置中，仅观察最开始启动连续转换的上升沿。硬件触发器功能可在任何转换模式及配置下运行。

● 5.2.6.4 转换控制

转换可在12位模式、10位模式或8位模式下进行，具体取决于ADC_SC3[MODE]位。转换可由软件或硬件触发发起。此外，可将ADC模块配置为低功耗操作、长采样时间、连续转换以及自动将转换结果与软件确定的比较值作比较。

❖ 5.2.6.4.1 发起转换

以下情形时会发起转换：

- 已选择软件触发操作的情况下，对 ADC_SC1 执行一次写操作或在 FIFO 模式下一组写操作（并非所有 ADCH 位都为 1）。
- 已选择硬件触发操作的情况下发生硬件触发器(ADHWT)事件。
- 连续转换使能的情况下结果被传输到数据寄存器。

如果连续转换使能，则在当前转换完成后会自动发起新转换。在软件触发操作中，连续转换在写入 ADC_SC1 后开始，并一直继续直到中止。在硬件触发操作中，连续转换在硬件触发事件后开始，并一直继续直到中止。

❖ 5.2.6.4.2 完成转换

转换结果传输到数据结果寄存器ADC_R时即完成转换。这通过ADC_SC1[COCO]置位而指示。如果DC_SC1[AIEN]在ADC_SC1[COCO]置位时为高电平，则会生成中断。

❖ 5.2.6.4.3 中止转换

正在进行中的任何转换在下列情况中都会中止：

- 发生对ADC_SC1的写操作。
  - ❖ 如果ADC_SC1[ADCH]非全1且ADC_SC4[AFDEP]为全0，将中止当前转换并启动新的转换。
  - ❖ 如果ADC_SC4[AFDEP]非全0，将中止当前转换和其余转换，并且不会启动新的转换。
  - ❖ 在ADC_SC4[AFDEP]位指定的层再次执行FIFO时，将会发起新转换。
- 发生对ADC_SC2、ADC_SC3、ADC_SC4、ADC_CV的写操作。这表明工作模式已发生改变，因此当前和其余转换（当ADC_SC4[AFDEP]非全0时）无效。
- MCU复位。
- MCU进入停止模式，ADACK未使能。

转换中止后，数据寄存器ADC_R的内容保持不变。然而，这些内容仍然是完成最后一次成功转换之后传输的值。如果转换因复位而中止，ADC_R会恢复到复位状态。

❖ 5.2.6.4.4 功率控制

ADC模块在启动转换前一直保持在其空闲状态。如果选择ADACK作为转换时钟源，则还会使能ADACK时钟产生器。模块为活动状态时，其功耗可通过置位ADC_SC3[ADLPC]来降低。这会降低f_{ADCK}的最大值。

❖ 5.2.6.4.5 采样时间和总转换时间

总转换时间取决于采样时间（由ADC_SC3[ADLSMP]决定）、MCU总线频率、转换模式（8位、10位或12位）及转换时钟频率(f_{ADCK})。该模块激活后，便开始对输入进行采样。ADC_SC3[ADLSMP]选择短（3.5ADCK周期）或长（23.5ADCK周期）采样时间。采样完成时，转换器与输入通道隔离开来，执行逐次逼近算法以确定模拟信号的数字值。转换算法完成时，转换结果转移到ADC_R。

如果总线频率低于f_{ADCK} 频率，使能短采样时间时(ADC_SC3[ADLSMP] = 0)，就无法保证连续转换的精确采样时间。如果总线频率低于f_{ADCK} 频率的1/11，使能长采样时间时(ADC_SC3[ADLSMP] = 1)，就无法保证连续转换的精确采样时间。

下表总结了不同条件下的最长总转换时间。

表 5-10 总转换时间与控制条件

转换类型	ADICLK	ADLSMP	最长总转换时间
8 位模式下,单次转换模式或者连续转换模式下的第一次转换	0x，10	0	20 个 ADCK 周期 + 5 个总线时钟周期
10位或12位模式下，单次转换模式或者连续转换模式下的第一次转换	0x，10	0	23 个 ADCK 周期 + 5 个总线时钟周期
8 位模式下,单次转换模式或者连续转换模式下的第一次转换	0x，10	1	40 个 ADCK 周期 + 5 个总线时钟周期
10位或12位模式下，单次转换模式或者连续转换模式下的第一次转换	0x，10	1	43 个 ADCK 周期 + 5 个总线时钟周期
8位模式下，单次转换模式或者连续转换模式下的第一次转换	11	0	5 μs + 20 个 ADCK + 5 个总线时钟周期
10位或12位模式下，单次转换模式或者连续转	11	0	5 μs + 23 个 ADCK + 5 个总线时钟周期

换模式下的第一次转换			
8位模式下，单次转换模式或者连续转换模式下的第一次转换	11	1	5 μs + 40 个 ADCK + 5 个总线时钟周期
10位或12位模式下，单次转换模式或者连续转换模式下的第一次转换	11	1	5 μs + 43 个 ADCK + 5 个总线时钟周期
8位模式下，连续转换模式下的后续转换；$f_{BUS} > f_{ADCK}$	xx	0	17 个 ADCK 周期
10位或12位模式下，连续转换模式下的后续转换；$f_{BUS} > f_{ADCK}$	xx	0	20 个 ADCK 周期
8位模式下，连续转换模式下的后续转换；$f_{BUS} > f_{ADCK}/11$	xx	1	37 个 ADCK 周期
10位或12位模式下，连续转换模式下的后续转换；$f_{BUS} > f_{ADCK}/11$	xx	1	40 个 ADCK 周期

最长总转换时间由选定的时钟源和分频比决定。时钟源可通过ADC_SC3[ADICLK]位选择，分频比通过ADC_SC3[ADIV]位指定。例如，在10位模式下，选择总线时钟作为输入时钟源，选定的输入时钟分频比为1，总线频率为8MHz，则单次转换的转换时间通过下式计算：

$$\text{转换时间} = \frac{23 ADCKCyc}{8MHz/1} + \frac{5busCyc}{8MHz} = 3.5us$$

8MHz 时的总线周期数为：

$$\text{总线周期} = 3.5us \times 8MHz = 28$$

注：为了满足ADC规格要求，ADCK 频率必须介于f_{ADCK}最小值和f_{ADCK}最大值之间。

● 5.2.6.5 自动比较功能

比较功能经配置后可用于检查上限或下限。对输入进行采样和转换后，结果与比较值(ADC_CV)的补值相加。与上限(ADC_SC2[ACFGT] = 1)比较时，如果结果大于或等于比较值，则ADC_SC1[COCO]置位。与下限(ADC_SC2[ACFGT]=0)比较时，如果结果小于比较值，则ADC_SC1[COCO]置位。转换结果与比较值的补值相加所得到的值被传输到ADC_R。

比较功能使能，且完成转换后，如果比较条件不成立，则ADC_SC1[COCO]不置位且无任何数据传输到结果寄存器。如果ADC中断使能(ADC_SC1[AIEN] = 1)，则在ADC_SC1[COCO]置位后会生成某个ADC中断。

比较功能和FIFO使能，且完成所有转换后，如果ADC_SC4[ACFSEL]为低电平时所有比较条件均不成立；或ADC_SC4[ACFSEL]为高电平时并非所有比较条件都成立，则ADC_SC1[COCO]不置位。无论比较条件成立或不成立，只要FIFO使能，比较数据就会传输到结果寄存器中。

注：(1) 在MCU处于等待或停止模式时，比较功能可监测通道上的电压。在比较条件得到满足时，ADC中断会唤醒MCU。

(2) 比较功能在FIFO使能时无法在连续转换模式下工作

● 5.2.6.6 FIFO 操作

ADC 模块支持 FIFO 操作以最大程度地减少 CPU 中断，从而降低 CPU 处理 ADC 中断服务例程的负荷。该模块包含两个 FIFO，分别缓存模拟输入通道和模拟结果。FIFO 功能在 ADC_SC4[AFDEP]位置位为非零值时使能。FIFO 深度由这些位指示。FIFO 最多支持 8 级缓冲区。

FIFO功能使能时，模拟输入通道FIFO通过ADC_SC1[ADCH]位进行访问。必须按顺序将模拟通道写入该FIFO。如果通道FIFO的填充水平低于ADC_SC4[AFDEP]位指示的水平，无论设置的是软件触发还是硬件触发，ADC都不会开始转换。对ADC_SC1[ADCH]进行读操作将读取当前活动通道值。对ADC_SC1[ADCH]进行写操作将重新填充通道FIFO以开始新的转换。当前转换和任何未开始的转换都会中止。在所有转换完成后或ADC处于空闲状态时对ADC_SC1执行写访问。

FIFO功能使能时，FIFO的结果通过ADC_R寄存器进行访问。必须通过上述两个寄存器并且按照与模拟输入通道FIFO相同的顺序读取结果，这样才能获得正确结果。在FIFO模式下，对ADC_R的读操作应等到所有转换都完成之后进行。无论设置的是软件触发还是硬件触发，只有模拟输入通道FIFO指示的所有转换都完成之后，ADC_SC1[COCO]位才会置位。当FIFO转换完成且ADC_SC1[COCO]位置位时，如果ADC_SC1[AIEN]置位，则将向CPU提交中断请求。

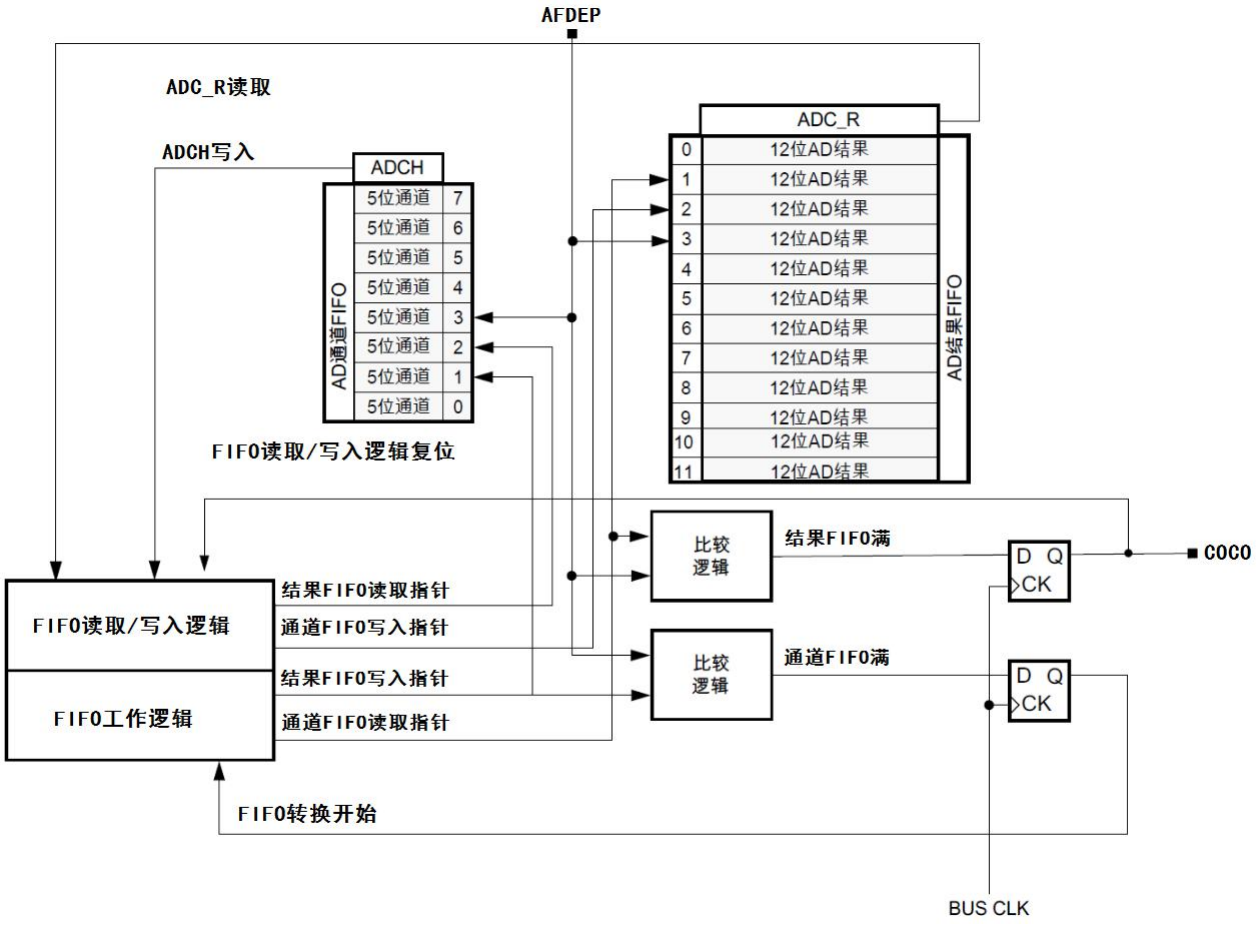


图 5-2 ADC FIFO 结构

如果使能软件触发,则完成一个转换且将其结果存储在结果FIFO中之后,就会立即从模拟输入通道FIFO中提取下一个模拟通道。当模拟输入通道 FIFO 中设置的所有转换都完成时，ADC_SC1[COCO]位置位，如果ADC_SC1[AIEN]位置位，则将向 CPU 提交中断请求。

如果使能硬件触发模式（ADC_SC2[ADTRG]= 1），则只有当前转换完成，其结果已存储到结果FIFO，并且将下一个硬件触发脉冲馈入ADC模块之后，才会从模拟输入通道FIFO中提取下一个模拟通道。模拟输入通道FIFO中设置的所有转换都完成后，如果ADC_SC1[AIEN]位置位，则ADC_SC1[COCO]位会置位且中断请求会

提交给CPU。

在单次转换模式下（ADC_SC1[ADCO]位清零），当ADC_SC1[COCO]位置位时，ADC停止转换，直至再次达到通道FIFO的填充要求或出现新的硬件触发脉冲。

FIFO还提供扫描模式来简化输入通道FIFO的重复工作。在FIFO模式下，当ADC_SC4[ASCANE]位置位时，无论输入通道FIFO为何值，FIFO都会使用第一个通道。一旦第一个通道写入输入通道值，ADC转换便开始在FIFO模式下工作。对输入通道FIFO执行连续的写操作会覆盖该FIFO中的第一个通道元素。在扫描FIFO模式下，当结果FIFO中填充的内容达到ADC_SC4[AFDEP]位指示的深度时，ADC_SC1[COCO]位置位。

在连续转换模式（ADC_SC1[ADCO]位置位）下，当所有转换完成时，ADC立即开始下一转换。ADC模块将从模拟输入通道FIFO的开始处提取模拟输入通道。

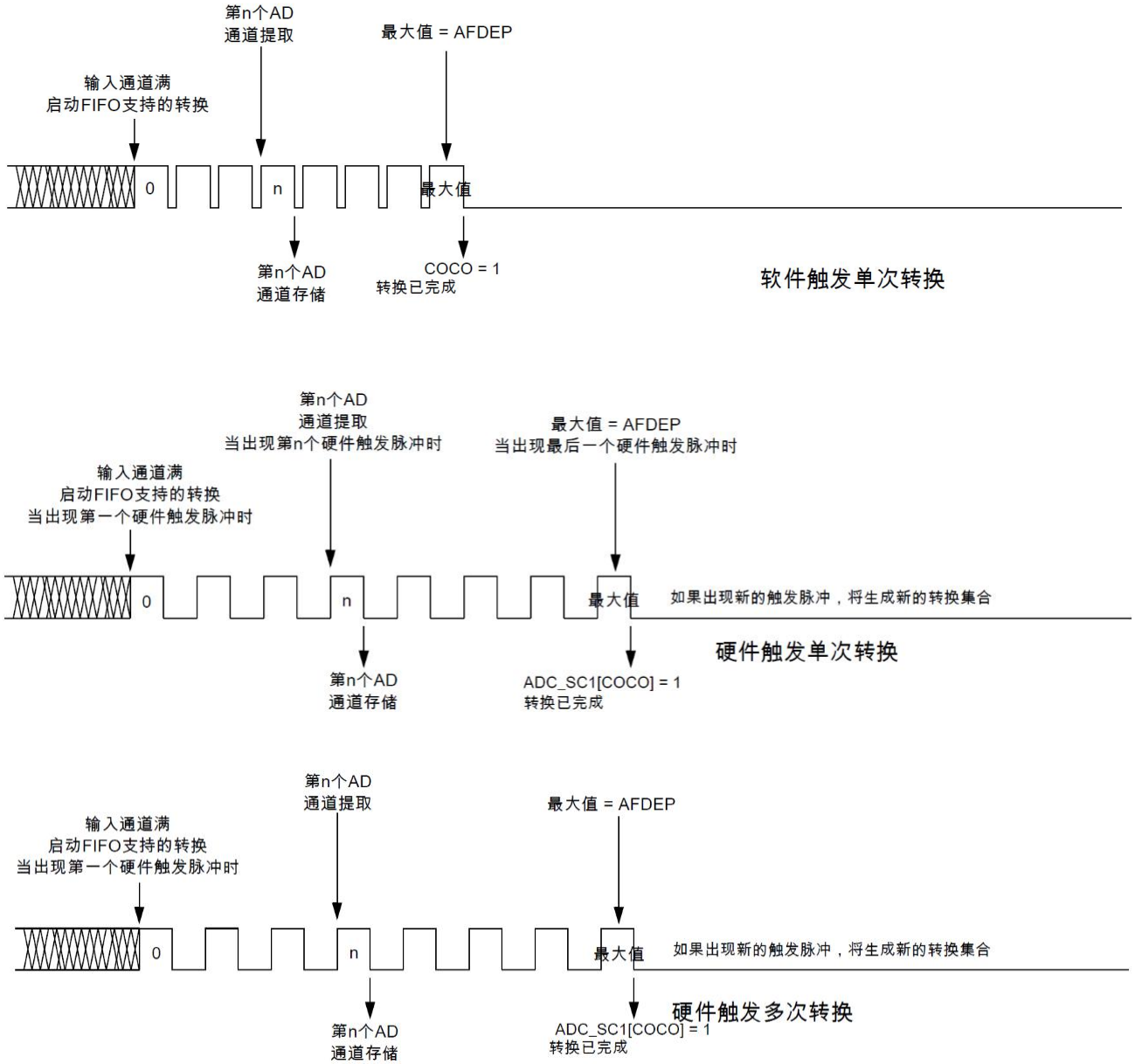


图5-3 ADC FIFO转换序列

● 5.2.6.7 MCU 等待模式下的操作

等待模式是低功耗的待机模式，通过该模式可迅速恢复，因为时钟源保持有效。如果转换在MCU进入等待模式时正在进行，则它会一直继续，直到完成。MCU因硬件触发而处于等待模式或连续转换已使能的情况下，可发起转换。

在等待模式下，可选择总线时钟、总线时钟2分频、ALTCLK和ADACK作为转换时钟源。

ADC中断已使能(ADC_SC1[AIEN] = 1)的情况下，转换完成事件会使ADC_SC1[COCO]置位，并生成ADC中断将MCU从等待模式中唤醒。

● 5.2.6.8 MCU 停止模式操作

停止模式是低功耗待机模式，在此模式下，MCU上的大多数甚至全部时钟源都被禁用。

❖ 5.2.6.8.1 采用停止模式并禁用 ADACK

如果没有选择异步时钟ADACK作为转换时钟，则执行STOP指令会中止当前转换并使ADC处于空闲状态。ADC_R的内容不受停止模式的影响。退出停止模式后，需要一次软件或硬件触发来使转换继续。

❖ 5.2.6.8.2 ADACK 使能情况下的停止模式

如果选择ADACK作为转换时钟，则ADC在停止模式期间会继续运行。为保证ADC的运行，MCU的电压调节器必须在停止模式期间保持有效。

如果转换在MCU 进入停止模式时正在进行，则它会一直继续，直到完成。MCU因硬件触发而处于停止模式或连续转换已使能的情况下，可发起转换。

ADC中断已使能(ADC_SC1[AIEN] = 1)的情况下，转换完成事件会使ADC_SC1[COCO]置位，并生成ADC中断将MCU从停止模式中唤醒。在FIFO模式下，ADC无法彻底完成转换操作，也无法将MCU从停止模式中唤醒。

注：ADC模块可将系统从低功耗停止模式中唤醒，并让MCU开始消耗运行级别的电流，但不生成系统级中断。为阻止这种情况的发生，在进入停止模式并继续执行ADC转换后，必须清除数据传输阻塞机制。

■ 5.2.7 初始化信息

本节将举例说明ADC模块的初始化和配置步骤。您可以将模块配置为8位、10位或12位分辨率、单一或连续转换以及轮询或中断方法，还有其他许多选项可供选择。有关本例中使用的信息，请参见ADC_SC3寄存器。

注：十六进制值的前缀是0x，二进制值的前缀是%，十进制值无前缀字符。

● 5.2.7.1 ADC 模块初始化示例

在使用ADC模块完成转换前，首先必须对它进行初始化。下面介绍ADC模块的初始化方法。

❖ 5.2.7.1.1 初始化序列

常见的初始化序列如下：

1. 更新配置寄存器(ADC_SC3)，以选择输入时钟源以及生成内部时钟ADCK所使用的分频比。此寄存器还可用于选择采样时间和低功耗配置

2. 更新状态和控制寄存器2(ADC_SC2)，以选择硬件或软件转换触发、比较功能选项（如果启用）。
3. 更新状态和控制寄存器1(ADC_SC1)，以选择转换是连续进行还是仅完成一次，以及是启用还是禁用转换完成中断。还可在此处选择执行转换的输入通道。

❖ 5.2.7.1.2 伪码示例

在本例中，ADC模块设置为启用中断，并以低功率、长采样时间在输入通道1 上执行单一的10位转换，其中内部的ADCK时钟是总线时钟的1分频。

示例：常规ADC初始化例程

```
void ADC_init(void)
{
       /* The following code segment demonstrates how to initialize ADC by low-power mode, Long sample time, bus frequency, software triggered from AD1 external pin without FIFO enabled
      */
      ADC_APCTL1 = ADC_APCTL1_ADPC1_MASK;
      ADC_SC3 = ADC_SC3_ADLPC_MASK | ADC_SC3_ADLSMP_MASK | ADC_SC3_MODE0_MASK;
      ADC_SC2 = 0x00;
      ADC_SC1 = ADC_SC1_AIEN_MASK | ADC_SC1_ADCH0_MASK;
}
```

● 5.2.7.2 ADC FIFO模块初始化示例

在使用ADC模块执行FIFO转换前，首先必须执行初始化过程。常见序列如下：
1. 更新配置寄存器(ADC_SC3)，以选择输入时钟源以及生成内部时钟ADCK所使用的分频比。此寄存器还可用于选择采样时间和低功耗配置。
2. 更新配置寄存器(ADC_SC4)，以选择FIFO扫描模式、FIFO比较功能选择（OR或AND 函数）和FIFO深度。
3. 更新状态和控制寄存器2(ADC_SC2)，以选择硬件或软件转换触发、比较功能选项（如果启用）。
4. 更新状态和控制寄存器1(ADC_SC1)，以选择转换是连续进行还是仅完成一次，以及是启用还是禁用转换完成中断。还可在此处选择执行转换的输入通道。

❖ 5.2.7.2.1 伪代码示例

本例中，ADC模块设置如下：使能中断，以低功耗和长采样时间对输入通道1、3、5、7 执行单次硬件触发的10位4级FIFO转换。内部ADCK时钟从总线时钟获得（1分频）。

示例：FIFO ADC 初始化例程

```
void ADC_init(void)
{
   /* The following code segment demonstrates how to initialize ADC by low-power mode, long sample time, bus frequency, hardware triggered from AD1, AD3, AD5, and AD7 external pins with 4-level FIFO enabled */
   ADC_APCTL1 = ADC_APCTL1_ADPC6_MASK | ADC_APCTL1_ADPC5_MASK | ADC_APCTL1_ADPC3_MASK |
   ADC_APCTL1_ADPC1_MASK;
   ADC_SC3 = ADC_SC3_ADLPC_MASK | ADC_SC3_ADLSMP_MASK | ADC_SC3_MODE1_MASK;
   // setting hardware trigger
```

```
    ADC_SC2 = ADC_SC2_ADTRG_MASK ;
    //4-Level FIFO
    ADC_SC4 = ADC_SC4_AFDEP1_MASK | ADC_SC4_AFDEP0_MASK;
    // dummy the 1st channel
    ADC_SC1 = ADC_SC1_ADCH0_MASK;
    // dummy the 2nd channel
    ADC_SC1 = ADC_SC1_ADCH1_MASK | ADC_SC1_ADCH0_MASK;
    // dummy the 3rd channel
    ADC_SC1 = ADC_SC1_ADCH2_MASK | ADC_SC1_ADCH0_MASK;
    // dummy the 4th channel and ADC starts conversion
    ADC_SC1 = ADC_SC1_AIEN_MASK | ADC_SC1_ADCH2_MASK | ADC_SC1_ADCH1_MASK | ADC_SC1_ADCH0_MASK;
}
```

示例: FIFO ADC中断服务例程

```
unsigned short buffer[4];
interrupt VectorNumber_Vadc void ADC_isr(void)
{
    /* The following code segment demonstrates read AD result FIFO */
    // read conversion result of channel 1 and COCO bit is cleared
    buffer[0] = ADC_R;
    // read conversion result of channel 3
    buffer[1] = ADC_R;
    // read conversion result of channel 5
    buffer[2] = ADC_R;
    // read conversion result of channel 7buffer[3] = ADC_R;
}
```

注：ADC_R是16位ADC结果寄存器，由ADC_RH和ADC_RL合并得来

■ 5.2.8 应用信息

本节包含在应用中使用ADC模块的信息。ADC已集成到微控制器中，可用在需要A/D转换器的嵌入式控制应用中。

● 5.2.8.1 外部引脚和布线

下面几节介绍与ADC模块相关的外部引脚以及如何使用它们才能达到最佳效果。

❖ 5.2.8.1.1 模拟电源引脚

ADC模块上的模拟电源和接地电源（VDDA 和VSSA）在某些器件上可用作单独的引脚。在某些器件上，VSSA与MCU数字VSS共用同一引脚。在其他器件上，VSSA和VDDA共用MCU数字供电引脚。在这类情况下，为了确保模拟电源与数字电源间保持一定的距离，需要为这类引脚提供各自单独的焊盘。

如果用在单独的引脚上，VDDA和VSSA必须连接到与其对应的MCU数字电源（VDD和VSS）相同的电压上，并且布线时必须非常小心，以最大限度提高抗噪性并绕过与封装靠得很近的电容器。

如果分别使用单独的模拟电源和数字电源，则两者之间的接地连接必须位于VSSA引脚上。这应该是两个

电源之间唯一的接地连接（如果可能）。VSSA引脚是良好的单点接地位置。

❖ 5.2.8.1.2 模拟参考引脚

除模拟电源之外，ADC模块还具有用于两个参考电压输入的连接。 高参考电压为VREFH，可在某些器件上作为VDDA被相同的引脚共用。低参考电压为VREFL，可在某些器件上作为VSSA被相同的引脚共用。

如果在单独的引脚上可用，则可将VREFH连接到与VDDA相同的电势，或由介于数据表中指定的最小VDDA与VDDA电势之间的外部电压源驱动（VREFH不能超过VDDA）。如果在单独的引脚上可用，则必须将VREFL连接到与VSSA相同的电势。必须小心安排VREFH和VREFL以便实现最大抗噪性，而且旁路电容必须尽可能靠近封装放置。

需要在每个逐次逼近步骤将电荷供应给电容阵列的交流电流（采用电流尖脉冲的形式）通过VREFH和VREFL环路消耗。满足这一电流要求的最佳外部元件为具有良好高频特性的0.1μF 电容。该电容在VREFH和VREFL之间连接，且必须尽可能地靠近封装引脚放置。该路径中不建议采用电阻，因为电流会引起电压降，这可能导致转换错误。 该路径中的电感必须为最小值（仅为寄生）。

❖ 5.2.8.1.3 模拟输入引脚

外部模拟输入通常由MCU器件上的数字I/O引脚共用。引脚I/O控制通过设置某个引脚控制寄存器的相应控制位而禁用。无需相关引脚控制寄存器位置位便可在输入上进行转换。将某个引脚用作模拟输入时，建议引脚控制寄存器位始终置位。这能避免争用问题，因为输出缓冲区处于其高阻抗状态且上拉电阻禁用。另外，输入缓冲区在其输入不处于VDD或VSS时会消耗直流电流。应当针对用作模拟输入的所有引脚设置引脚控制寄存器位以便实现最低工作电流。

经验数据表明模拟输入上的电容在存在噪声或电源阻抗高时刻提高性能。使用具有良好高频特性的0.01μF电容就足够了。在某些情况下并不一定要使用这类电阻，但是一旦使用，则必须将其尽可能地靠近封装引脚放置并将VSSA作为参考电压。

为确保转换正确完成，输入电压必须介于VREFH和VREFL之间。如果输入电压等于或大于VREFH，转换器电路会将信号转换为0xFFF（满刻度12位表示法）、0x3FF（满刻度10 位表示法）或0xFF（满刻度8位表示法）。如果输入电压等于或小于VREFL，则转换器电路会将其转换为0x000。在VREFH和VREFL之间的输入电压属于直线线性转换。采用电容进行充电时，存在与VREFL有关的短时电流。ADC_SC3[ADLSMP]为低电平时，对该输入进行采样的时间为ADCK源的3.5个周期，ADC_SC3[ADLSMP]为高电平时，进行采样的时间为23.5个周期。

为了尽量减小电流注入造成的准确度损失，转换过程中不能对与模拟输入引脚相邻的引脚进行转换。

● 5.2.8.2 误差来源

A/D 转换存在几个误差来源 这些会在下面的章节中进行讨论。

❖ 5.2.8.2.1 采样误差

为正确转换，输入的采样时间必须长到能实现正确的精度。假设最大输入电阻约为7kΩ、输入电容约为5.5pF，则在最小采样窗口（在最大ADCK 频率8MHz时为3.5个周期）内可将采样误差控制到1/4最低有效位内（12位分辨率），同时还假设外部模拟源(RAS) 的电阻始终小于2kΩ。通过设置ADC_SC3[ADLSMP]（以将采样窗口增加到23.5个周期），或减小ADCK频率以增加采样时间，可提高源电阻或采样精度。

❖ 5.2.8.2.2 引脚漏电误差

如果外部模拟源电阻(RAS)偏高，I/O 引脚上的漏电流可能会导致转换出错。如果应用程序无法接受这一误差，则使RAS 低于VDDA / (2N*I_{LEAK})以便使之小于1/4最低有效位漏电流误差（N = 8（8位），N = 10（10 位），N = 12（12 位）模式）。

❖ 5.2.8.2.3 噪声性误差

采样或转换过程中出现的系统噪音可能会影响转换精度。只有满足下列条件，ADC精度数才能按规定得到保证：

- VREFH到VREFL之间有一个0.1μF的低ESR电容器。
- VDDA 到VSSA 之间有一个0.1μF的低ESR电容器。
- 如果主电源使用了电感隔离，则VDDA到VSSA之间还要放一个1μF的电容器。
- VSSA(和VREFL，如果连接) 连接到接地层中某安静点处的VSS。
- 在发起（硬件触发转换）ADC转换前，或发起（硬件或软件触发转换）ADC转换后立即让MCU在等待或停止模式下运行。
- 对于软件触发的转换，对ADC_SC1 执行写操作后立即使用停止指令。
- 对于停止模式操作，选择ADACK 作为时钟源。停止模式下的操作可减少VDD噪音，但也会因停止恢复而增多有效转换时间。
- 转换期间，MCU上无I/O切换、输入或输出。

在某些情况中，外部系统活动会引起辐射或传导噪音排放或使过大的VDD噪音耦合到ADC中。在这类情况下，或者当无法将MCU置于等待或停止模式时，或者无法暂停I/O活动时，建议的这些动作可能会降低噪音对精度的影响：

- 在VREFL或VSSA的选定输入通道上放一个0.01μF的电容器 (CAS)（此举可改善噪音问题，但会影响基于外部模拟源阻抗的采样率）。
- 将模拟输入接连转换多次并除以结果总数得出结果的平均值。需要四个样本来消除1LSB的影响，一次性错误。
- 通过偏离异步时钟(ADACK)进行操作并取平均值来降低同步噪声的影响。与ADCK同步的噪音的平均值无法算出。

❖ 5.2.8.2.4 代码宽度和量化误差

ADC将理想的直线式传递函数量化为4096步（12 位模式下）。每一步在理想情况下都具有相同的高度（1个代码）和宽度。宽度定义为某个代码的转变点与下一个代码的转变点之间的距离差。定义为1LSB的N位转换器的理想代码宽度（在本例中，N可以是8、10或12）是：

$$1lsb = (V_{REFH} - V_{REFL}) / 2^N$$

结果的数字化导致存在固有的量化误差。对于8位或10位转换，该代码在电压位于各转变点之间的中点时转变，直线式传递函数在各转变点由实际的传递函数精确表示。因此，量化误差在8位模式或10位模式下将为± 1/2lsb。然而，首次(0x000)转换的代码宽度只有1/2lsb，最后次（0xFF或0x3FF）转换的代码宽度为1.5lsb。对于12位转换，代码只有在具有完整代码宽度时才会进行转变，因此量化误差为-1lsb到0lsb，每个振幅的代码宽度为1lsb。

❖ 5.2.8.2.5 线性误差

ADC还还体现出多种形式的非线性。虽然已尽可能减少这类误差，但系统必须意识到误差的存在，因为误差会影响整体精度。这些误差为：

- 零刻度误差(EZS)（有时也称为偏移量）— 该误差是指第一个转换的实际代码宽度与理想代码宽度（8位或10位模式下为1/2lsb；12位模式下为1lsb）之间的差值。如果第一个转换为0x001，则使用实际的0x001代码宽度与其理想宽度(1lsb)之间的差值。
- 满刻度误差(EFS) — 该误差是指最后一个转换的实际代码宽度与理想代码宽度（8位或10位模式下为1.5lsb；12位模式下为1lsb）之间的差值。如果最后一个转换为0x3FE，则使用实际的0x3FE代码宽度与其理想宽度(1lsb)之间的差值。
- 差分非线性(DNL) — 该误差是指所有转换的实际代码宽度与理想代码宽度之间的最差情况差值。
- 积分非线性(INL) — 该误差是指DNL运行和的绝对值所达到的最大值。更简而言之，它对所有代码来说，是给定代码的实际转换电压与其对应理想转换电压的最差情况差值。
- 未调整总误差(TUE) — 该误差是指实际传输函数与理想直线传递函数之间的差值,包括所有形式的误差。

❖ 5.2.8.2.6 代码抖动、非单调性和失码

模数转换器易受三种特殊形式错误的影响。它们是代码抖动、非单调性和失码。

代码抖动发生在给定的输入电压在某些点经过重复采样后转换为两个值中的某一个值时。理想情况下，当输入电压小于但无限接近转变电压时，转换器会产生更低代码（反之亦然）。然而，对于接近转变电压的各种输入电压，即使少量的系统噪声也可能会使转换器在两个代码间变得不确定。在8位或10位模式下，此范围一般约为±1/2lsb；在12位模式下，则约为2lsb，随噪声的增大而增大。

通过对输入重复采样并求结果的平均值可以减少出现此错误的几率。此外，噪声性误差中介绍的方法也可以减少出现此错误的几率。

除了代码抖动外，当转换器为较高的输入电压转换到较低的代码时，会出现非单调性问题。失码是指不会为任何输入值进行转换的那些值。在8位或10位模式下，ADC保证是单调的并且无失码。

■ 5.3 模拟比较器（ACMP）

■ 5.3.1 简介

模拟比较器模块(ACMP)提供用于比较两个模拟输入电压的电路。适用于在整个供电电压范围内操作（轨到轨操作）。模拟多路复用器提供一个用于从四个通道中选择模拟输入信号的电路。一个信号由6位DAC提供。多路复用器电路适用于在整个供电电压范围内操作。6位DAC是一个64分接头的梯形电阻网络，可为需要电压参考的应用提供可选的电压参考。此64分接头的梯形电阻网络将电源参考Vin分成64个电压电平。6位数字信号输入可选择从Vin至Vin/64的输出电压电平。可从两个电压源选择Vin。

■ 5.3.2 特性

ACMP特性包括：

- 可在2.7V至5.5V的整个电源电压范围上操作
- 片上6位分辨率DAC，可从VDD或内部带隙中选择基准电压。

- 可配置迟滞
- 可在比较器输出的上升沿、下降沿、两个上升沿或两个下降沿时选择中断
- 可选择翻转比较器输出
- 最多4个可选择比较器输入
- 可在停止模式下操作

■ 5.3.3 结构框图

下图是 ACMP 模块的结构框图。

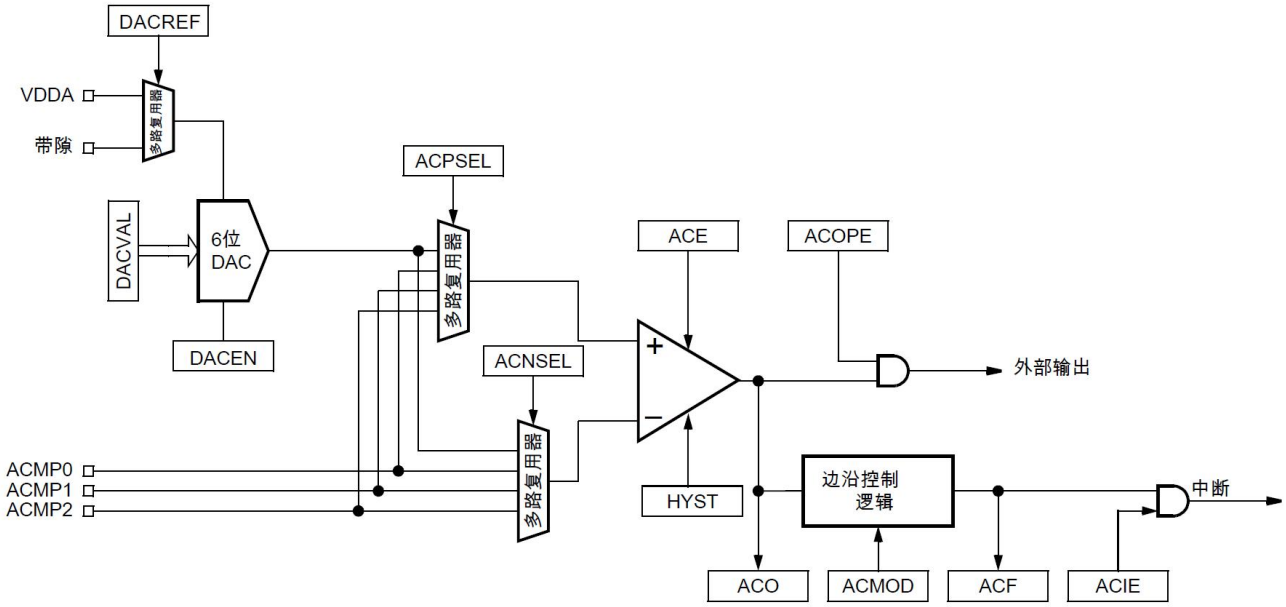


图 5-4 ACMP 结构框图

■ 5.3.4 操作模式

本节定义等待模式、停止模式和后台调试模式下的 ACMP 操作。

● 5.3.4.1 等待模式下的操作

在等待模式下，如果使能，ACMP将继续工作。如果使能，中断可唤醒MCU。

● 5.3.4.2 停止模式下的操作

如果已使能，ACMP（包括DAC 和CMP）会在停止模式下继续工作。如果ACMP_CS[ACIE]置位，则可生成ACMP中断将MCU从停止模式中唤醒。如果通过中断退出停止模式，那么在ACMP保留进入停止模式前的设置。如果通过复位退出停止模式，则ACMP会进入其复位状态。由于DAC消耗额外的电能，因此如果DAC输出不用作ACMP的基准输入，那么用户必须关闭DAC以省电。

● 5.3.4.3 调试模式下的操作

当MCU处于调试模式时，ACMP继续正常工作。

■ 5.3.5 外部信号说明

ACMP的输出也可以映射到外部引脚。当该输出映射到外部引脚时，ACMP_CS[ACOPE]控制该引脚来使能/禁用ACMP输出功能。

■ 5.3.6 存储器映射和寄存器说明

表 5-11 ACMP 存储器映射

绝对地址（十六进制）	寄存器名称	位宽	访问	复位值
4007_3000	ACMP0 控制和状态寄存器(ACMP0_CS)	8	R/W	00h
4007_3001	ACMP0 控制寄存器 0(ACMP0_C0)	8	R/W	00h
4007_3002	ACMP0 控制寄存器 1(ACMP0_C1)	8	R/W	00h
4007_3003	ACMP0 控制寄存器 2(ACMP0_C2)	8	R/W	00h
4007_4000	ACMP1 控制和状态寄存器(ACMP1_CS)	8	R/W	00h
4007_4001	ACMP1 控制寄存器 0(ACMP1_C0)	8	R/W	00h
4007_4002	ACMP1 控制寄存器 1(ACMP1_C1)	8	R/W	00h
4007_4003	ACMP1 控制寄存器 2(ACMP1_C2)	8	R/W	00h

● 5.3.6.1 ACMP 控制和状态寄存器（ACMPx_CS）

地址：基址 + 00h 偏移 ： ACMP0_CS = 4007_3000h / ACMP1_CS = 4007_4000h

位	7	6	5	4	3	2	1	0
读					ACO			
写	ACE	HYST	ACF	ACIE		ACOPE	ACMOD	
复位	0	0	0	0	0	0	0	0

表 5-12 ACMPx_CS 字段描述

字段	描述
7 ACE	模拟比较器使能控制位 0 ACMP禁用。 1 ACMP 使能。
6 HYST	模拟比较器迟滞选择控制位 0 20mV。 1 30mV。
5 ACF	ACMP中断标志位 当ACMP输出有一个由ACMOD定义的有效边沿时，由硬件同步置位。该位的置位滞后于ACMP0至总线时钟。将0写入ACF位可将该位清零。将1写入该位无效。
4 ACIE	ACMP中断使能 使能ACMP CPU中断。 0 禁用ACMP中断。

	1 使能 ACMP 中断。
3 ACO	ACMP输出 读取 ACO 将返回模拟比较器输出的当前值。ACO 复位到 0，并且在 ACMP 禁用(ACE = 0)时读作 0。
2 ACOPE	ACMP输出引脚使能 ACOPE使能焊盘逻辑，以便能将输出置于外部引脚上。 0 无法将ACMP输出置于外部引脚上。 1 可将 ACMP 输出置于外部引脚上。
1-0 ACMOD	ACMP MOD 确定中断触发器的触发模式。 00 ACMP 中断在输出下降沿发生。 01 ACMP 中断在输出上升沿发生。 10 ACMP 中断在输出下降沿发生。 11 ACMP 中断在输出下降沿或上升沿发生。

● 5.3.6.2 ACMP 控制寄存器 0（ACMPx_C0）

地址：基址 + 01h 偏移 ： ACMP0_C0 = 4007_3001h / ACMP1_C0 = 4007_4001h

位	7	6	5	4	3	2	1	0
读	0		ACPSEL		0		ACNSEL	
写								
复位	0	0	0	0	0	0	0	0

表 5-13 ACMPx_C0 字段描述

字段	描述
7 - 6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
5 - 4 ACPSEL	ACMP 正输入选择 00 外部基准0 01 外部基准 1 10 外部基准2 11 DAC 输出
3 - 2 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
1-0 ACNSEL	ACMP 负输入选择 00 外部基准0 01 外部基准1 10 外部基准2 11 DAC 输出

● 5.3.6.3 ACMP 控制寄存器 1 (ACMPx_C1)

地址：基址 + 02h 偏移 ： ACMP0_C1 = 4007_3002h / ACMP1_C1 = 4007_4002h

位	7	6	5	4	3	2	1	0

读 写	DACEN	DACREF	DACVAL					
复位	0	0	0	0	0	0	0	0

表 5-14 ACMPx_C1 字段描述

字段	描述
7 DACEN	DAC使能 使能6位DAC的输出。 0 DAC禁用。 1 DAC 使能。
6 DACREF	DAC基准选择 0 DAC选择带隙作为基准。 1 DAC 选择 VDDA 作为基准。
5-0 DACVAL	DAC输出电平选择 使用以下公式选择输出电压:Voutput= (Vin/64)x(DACVAL[5:0]+1)。Voutput 范围为 Vin/64 至 Vin，步进为 Vin/64。

● 5.3.6.4 ACMP 控制寄存器 2（ACMPx_C2）

地址：基址 + 03h 偏移 ： ACMP0_C2 = 4007_3003h / ACMP1_C2 = 4007_4003h

位	7	6	5	4	3	2	1	0
读	0						ACIPE	
写								
复位	0	0	0	0	0	0	0	0

表 5-15 ACMPx_C2 字段描述

字段	描述
7－3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2-0 ACIPE	ACMP 输入引脚使能 该3位字段控制对应的ACMP外部引脚是否能由模拟输入驱动。 0 不允许对应的外部模拟输入。 1 允许对应的外部模拟输入。

■ 5.3.7 功能说明

ACMP模块就功能而言由两部分组成：数模转换器(DAC)和比较器(CMP)。DAC包括一个64级DAC（数模转换器）和相关的控制逻辑。通过置位ACMP_C1[DACREF]，DAC可选择VDD或片上带隙基准源这两个参考输入的其中之一作为DAC输入Vin。DAC使能后，将ACMP_C1[DACVAL]中设置的数据转换为步进式模拟输出，然后馈入ACMP作为内部参考输入。该步进式模拟输出同时映射到模块之外。输出电压范围为Vin/64 到Vin。步进大小为Vin/64。

ACMP可以实现正输入与负输入的模拟比较，然后提供一个数字输出和相关的中断。ACMP的正负输入均可从四个通用输入中选择：来自DAC输出的三个外部参考输入和一个内部参考输入。ACMP的正输入通过

ACMP_C0[ACPSEL]选择，负输入通过ACMP_C0[ACNSEL]选择。用适当的值配置ACMPC0便可比较8个输入中的任意一对。

通过置位ACMP_CS[ACE]使能ACMP之后，比较结果以数字输出呈现。只要ACMP_CS[ACMOD]中定义的有效边沿出现，ACMP_CS[ACF]的电平就会变为有效值。如果ACMP_CS[ACIE]置位，那么就会发生ACMP CPU中断。有效边沿由ACMP_CS[ACMOD]定义。当ACMP_CS[ACMOD] = 00b或10b时，只有ACMP输出的下降沿有效。当ACMP_CS[ACMOD] = 01b时，只有ACMP输出的上升沿有效。当ACMP_CS[ACMOD] = 11b 时，ACMP输出的上升沿和下降沿均有效。

ACMP输出由总线时钟同步以生成ACMP_CS[ACO]，以便CPU能读取比较结果。在stop3模式下，即使ACMP的输出发生改变，但ACMP0无法及时被更新。因总线时钟可用而唤醒CPU时，可同步输出和更新ACMP_CS[ACO]。ACMP_CS[ACO]根据比较结果而改变，因此它可以用作一个跟踪标志，连续指示输入的电压变化。

如果选择芯片外部的参考输入作为ACMP的输入，那么必须置位对应的ACMP_C2[ACIPE]位以使能来自焊盘接口的输入。如果需要将ACMP的输出置于外部引脚上，那么必须置位ACMP_CS[ACOPE]位以使能焊盘逻辑的ACMP引脚功能。

■ 5.3.8 ACMP 的设置和操作

ACMP两部分（DAC 和CMP）的设置和操作可以独立进行。但是，如果DAC用作CMP的输入，那么在使能ACMP之前必须对DAC进行配置。

由于输入切换可能会引起ACMP输入的问题，因此用户应当在使能ACMP之前完成输入选择，并且在使能ACMP之后不得更改输入选择，以免产生未预期输出。同样，更改ACMP_C1[DACVAL]之后，DAC会有一定的设置延迟，因此用户应当在使能DAC之前完成ACMP_C1[DACVAL]的设置。

■ 5.3.9 复位

复位期间，会在默认模式下配置ACMP。CMP和DAC都被禁用。

■ 5.3.10 中断

如果在出现由ACMP_CS[ACMOD]定义的有效边沿时总线时钟可用，那么ACMP_CS[ACF]的电平就会变为有效值。如果ACMP_CS[ACIE]置位，那么会发生ACMP中断事件。ACMP_CS[ACF]位的电平将一直保持有效值，直到ACMP中断被软件清除。在停止模式下，ACMP输出的有效边沿会生成一个异步中断，它可将MCU从停止模式中唤醒。将0写入ACMP_CS[ACF]位可清除该中断。

第 6 章 定时器模块

■ 6.1 简介

本章对基于 ARM Cortex-M0+内核的 NV32F100x 系列 MCU 定时器相关的模块做了详细说明，其中包括看门狗定时器 WDOG、增强型定时器、周期性中断定时器 PIT、实时计数器 RTC 等模块内容。

■ 6.2 看门狗定时器（WDOG）模块

■ 6.2.1 简介

看门狗定时器(WDOG)模块是一个可供系统使用的独立定时器。它提供安全的特性，可确保软件按计划执行，且 CPU 不会陷入死循环中或者执行错误的代码。若 WDOG 模块在规定时间内未得到服务（刷新），它会复位 MCU。

■ 6.2.2 特性

WDOG 模块特性包括：

- 独立于总线时钟的可配置时钟源
  - ❖ 内部 32kHz RC 振荡器
  - ❖ 内部 1kHz RC 振荡器
  - ❖ 外部时钟源
- 可编程的超时周期
  - ❖ 可编程的 16 位超时值
  - ❖ 需要更长超时周期时，可选的固定 256 倍时钟预分频器
- 可实现计数器刷新的鲁棒性写入序列
  - ❖ 在 16 个总线时钟内，先写 0x02A6 后写 0x80B4 的刷新序列
- 可选的窗口模式刷新模式
  - ❖ 可编程的 16 位窗口值
  - ❖ 提供程序流程快于预期的鲁棒性检查
  - ❖ 试图提前刷新会触发复位。
- 允许后期诊断处理的可选超时中断
  - ❖ CPU 中断请求，产生相应的中断向量，并可执行相应的中断服务例程（ISR）
  - ❖ 在中断产生 128 个总线时钟后，强制复位。
- 看门狗配置位为复位后一次写入位，确保无法误更改看门狗的配置。
- 用于解锁一次写入配置位的鲁棒性写入序列
  - ❖ 在 16 个总线时钟内，先写入 0x20C5 后写入 0x28D9 的解锁序列，允许一次写入配置位的更新
  - ❖ 软件必须在解锁后以及 WDOG 关闭解锁窗口前的 128 个总线时钟内执行更新的操作。

■ 6.2.3 结构框图

下图为 WDOG 模块的结构框图。

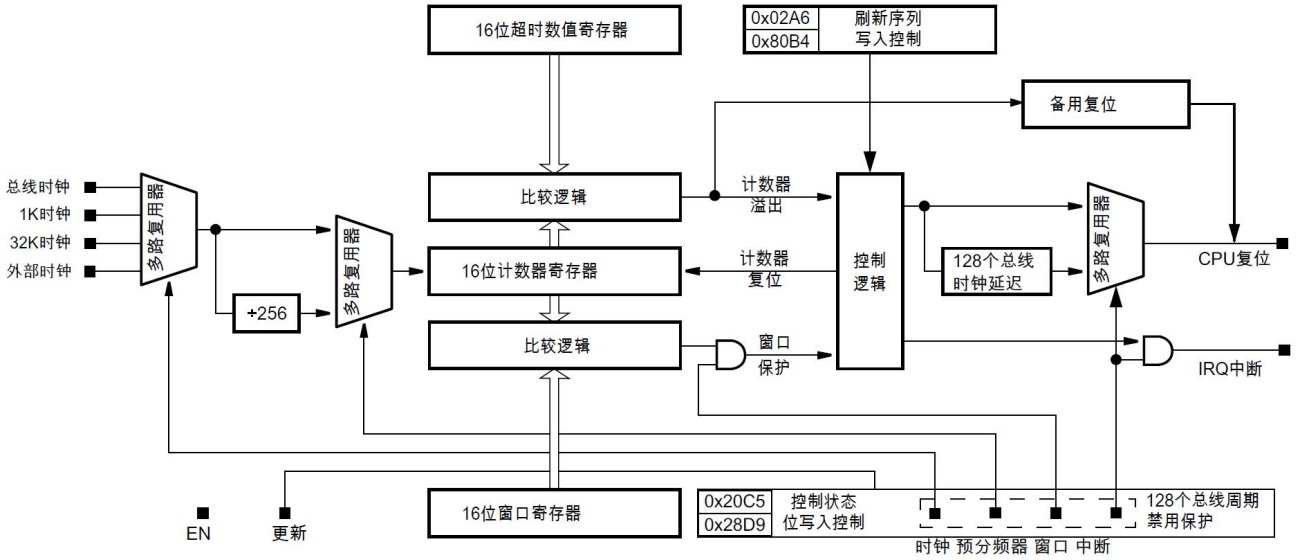


图 6-1. WDOG 结构框图

■ 6.2.4 存储器映射和寄存器说明

注：如果使用半字访问 WDOG_CNT、WDOG_TOVAL 和 WDOG_WIN，则必须遵循“低字节：高字节”的转置 16 位字节格式。因此首选 8 位读/写操作。

表 6-1 WDOG 存储器映射

绝对地址(16 进制)	寄存器名称	偏移量	位宽	访问	复位值
0x4005_2000	看门狗控制和状态寄存器 1（WDOG_CS1）	0h	8	R/W	80h
0x4005_2001	看门狗控制和状态寄存器 2（WDOG_CS2）	1h	8	R/W	01h
0x4005_2002	看门狗计数寄存器:高位（WDOG_CNTH）	2h	8	R	00h
0x4005_2003	看门狗计数寄存器:低位（WDOG_CNTL）	3h	8	R	00h
0x4005_2004	看门狗超时值寄存器:高位（WDOG_TOVALH）	4h	8	R/W	00h
0x4005_2005	看门狗超时值寄存器:低位（WDOG_TOVALL）	5h	8	R/W	04h
0x4005_2006	看门狗窗口寄存器：高位（WDOG_WINH）	6h	8	R/W	00h
0x4005_2007	看门狗窗口寄存器：低位（WDOG_WINL）	7h	8	R/W	00h

● 6.2.4.1 看门狗控制和状态寄存器 1（WDOG_CS1）

本节说明看门狗控制和状态寄存器 1 的功能。

注：TST 仅在 POR 复位时才清零（0:0）。任何其他复位都不会影响该字段的数值。

地址：4005_2000h 基准 + 0h 偏移 = 4005_2000h

位	7	6	5	4	3	2	1	0
读 写	EN	INT	UPDATE	TST		DBG	WAIT	STOP
复位值	1	0	0	0	0	0	0	0

表 6-2 WDOG_CS1 字段描述

位	描述
7 EN	看门狗使能 该一次写入位使能看门狗计数器以开始计数。 0 看门狗禁用 1 看门狗使能
6 INT	看门狗中断 该一次写入位将看门狗配置为在发生复位触发事件（超时或者看门狗非法写入）时后再强制进行复位。128 个总线时钟延迟之后发生强制复位。 0 看门狗中断禁用，看门狗复位无延迟。 1 看门狗中断使能，看门狗复位有 128 个总线时钟延迟。
5 UPDATE	允许更新 该一次写入位使系统无需复位即能重新配置看门狗。 0 不允许更新。完成初始配置后。除非强制复位，否则无法修改看门狗配置。 1 允许更新。执行解锁写入序列后，软件可以在 128 个总线时钟内修改看门狗配置寄存器。
4-3 TST	看门狗测试 使能快速测试模式。测试模式允许软件检查计数器的所有位，来证明看门狗工作正常。参见看门狗快速测试部分。 该一次写入字段仅在 POR 时才清零（0:0）。任何其他复位都不会影响该字段的数值。 00 看门狗测试模式禁用。 01 看门狗用户模式使能（看门狗测试模式禁用）。测试看门狗后，软件应当使用该设置，使看门狗在用户模式下正常工作。 10 看门狗测试模式使能，仅使用低位字节。WDOG_CNTL 与 WDOG_TOVALL 进行比较。 11 看门狗测试模式使能，仅使用高位字节。WDOG_CNTH 与 WDOG_TOVALH 进行比较。
2 DBG	调试模式使能 该一次写入位在芯片处于调试模式时使能看门狗的操作。 0 看门狗在芯片调试模式下禁用。 1 看门狗在芯片调试模式下使能。
1 WAIT	等待模式使能 该一次写入位在芯片处于等待模式时使能看门狗的操作。 0 看门狗在芯片等待模式下禁用。 1 看门狗在芯片等待模式下使能。
0 STOP	停止模式使能 该一次写入位在芯片处于停止模式时使能看门狗的操作。 0 看门狗在芯片停止模式下禁用。 1 看门狗在芯片停止模式下使能。

● 6.2.4.2 看门狗控制和状态寄存器 2（WDOG_CS2）

本节说明看门狗控制和状态寄存器 2 的功能。

地址：4005_2000h 基准+ 1h 偏移 = 4005_2001h

位	7	6	5	4	3	2	1	0
读	WIN	FLG	0	PRES	0		CLK	
写		w1c						
复位值	0	0	0	0	0	0	0	1

注：w1c 表示写 1 该位就会清零，写 0 无效。

表 6-3 WDOG_CS2 字段描述

位	描述
7 WIN	看门狗窗口工作模式 该一次写入位使能窗口模式。参见窗口模式部分。 0 窗口模式禁用 1 窗口模式使能
6 FLG	看门狗中断标志 当 INT 在控制和状态寄存器 1 中置位后，该位可表示中断的产生。写入 1 将其清零。 0 未发生中断。 1 发生一次中断。
5 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
4 PRES	看门狗预分频器 该一次写入位使能看门狗计数器参考时钟的 256 倍预分频器。（功能框图显示的该时钟分频器。） 0 256 预分频器禁用。 1 256 预分频器使能。
3-2 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
1-0 CLK	看门狗时钟 该一次写入字段表示看门狗计数器的时钟源。参见时钟源部分。 00 总线时钟。 01 1khz 内部低功耗振荡器（LPOCLK）。 10 32khz 内部振荡器（ICSIRCLK）。 11 外部时钟源。

● 6.2.4.3 看门狗计数器寄存器：高位（WDOG_CNTH）

本节说明看门狗计数器寄存器：高位（CNTH）和低位（CNTL）。

看门狗计数器寄存器 CNTH 和 CNTL 提供对看门狗计数器数值的访问。软件可在任意时刻对计数器寄存器进行读操作。

软件无法直接写入看门狗计数器；然而，针对这两个寄存器的两个写入序列具有特殊功能：

1. 刷新序列可将看门狗计数器复位至 0x0000。参见刷新看门狗部分。

2. 解锁序列使看门狗能在不强制复位（WDOG_CS1[UPDATE] = 1 时）的情况下得到重新配置。参见代码示例：重新配置看门狗部分。

注：对这些寄存器的所有其他写操作均无效，并且会强制复位。

地址：4005_2000h 基准 + 2h 偏移 = 4005_2002h

位	7	6	5	4	3	2	1	0
读	CNTHIGH							
写								
复位值	0	0	0	0	0	0	0	0

表 6-4 WDOG_CNTH 字段描述

位	名称	描述	复位值
7-0	CNTHIGH	看门狗计数器的高位字节	0

● 6.2.4.4 看门狗计数器寄存器：低位（WDOG_CNTL）

参见 WDOG_CNTH 寄存器说明。

地址：4005_2000h 基准 + 3h 偏移 = 4005_2003h

位	7	6	5	4	3	2	1	0
读	CNTLOW							
写								
复位值	0	0	0	0	0	0	0	0

表 6-5 WDOG_CNTH 字段描述

位	名称	描述	复位值
7-0	CNTLOW	看门狗计数器的低位字节	0

● 6.2.4.5 看门狗超时值寄存器：高位（WDOG_TOVALH）

本节说明看门狗超时值寄存器：高位(WDOG_TOVALH)和低位(WDOG_TOVALL)。WDOG_TOVALH 和 WDOG_TOVALL 包含 16 位数值，用于设置看门狗的超时周期。

看门狗计数器（WDOG_CNTH 和 WDOG_CNTL）与超时数值（WDOG_TOVALH 和 WDOG_TOVALL）连续作比较。若计数器达到超时值，则看门狗会强制进行复位。

注：不要将 0 写入看门狗超时数值寄存器，否则看门狗会始终生成复位。

地址：4005_2000h 基准 + 4h 偏移 = 4005_2004h

位	7	6	5	4	3	2	1	0
读	TOVALHIGH							
写								
复位值	0	0	0	0	0	1	0	0

表 6-6 WDOG_TOVALH 字段描述

位	名称	W/R 值	描述	复位值
7-0	TOVALHIGH		超时值的高位字节	0

● 6.2.4.6 看门狗超时值寄存器：低位（WDOG_TOVALL）

参见 WDOG_TOVALH 寄存器说明。

地址：4005_2000h 基准 + 5h 偏移 = 4005_2005h

位	7	6	5	4	3	2	1	0
读	TOVALLOW							
写								
复位值	0	0	0	0	0	1	0	0

表 6-7 WDOG_TOVALL 字段描述

位	名称	描述
7-0	TOVALLOW	超时值的低位字节

● 6.2.4.7 看门狗窗口寄存器：高位（WDOG_WINH）

本节说明看门狗窗口寄存器：高位(WDOG_WINH)和低位(WDOG_WINL)。窗口模式使能后（WDOG_CS2[WIN]置位），WDOG_WINH 和 WDOG_WINL 决定刷新序列被视为有效的最早时间。参见看门狗刷新机制部分。

WDOG_WINH 和 WDOG_WINL 必须小于 WDOG_TOVALH 和 WDOG_TOVALL。

地址：4005_2000h 基准 + 6h 偏移 = 4005_2006h

位	7	6	5	4	3	2	1	0
读	WINHIGH							
写								
复位值	0	0	0	0	0	0	0	0

表 6-8 WDOG_WINH 字段描述

位	名称	描述	复位值
7-0	WINHIGH	看门狗窗口的高位字节	0

● 6.2.4.8 看门狗窗口寄存器：低位（WDOG_WINL）

说明参见 WDOG_WINH 寄存器说明。

地址：4005_2000h 基准 + 7h 偏移 = 4005_2007h

位	7	6	5	4	3	2	1	0
读	WINLOW							
写								
复位值	0	0	0	0	0	0	0	0

表 6-9 WDOG_WINL 字段描述

位	名称	描述
7-0	WINLOW	看门狗窗口的低位字节

■ 6.2.5 功能说明

WDOG 模块提供故障安全机制，确保系统在出现故障时（比如CPU 时钟停摆或软件代码出现跑飞）能够复位至已知状态。看门狗计数器采用选定的时钟源工作，并判断是否得到定期的更新服务。若非如此，它便会复位系统。

超时周期、窗口模式和时钟源均可以编程配置，但必须在复位后的128 个总线时钟内进行第一次配置。

● 6.2.5.1 看门狗刷新机制

若看门狗计数器未得到刷新，则看门狗会复位MCU。鲁棒性刷新机制使跑飞代码不太可能刷新看门狗。要刷新看门狗计数器，软件必须在超时周期结束前执行刷新写入序列。此外，如果使用了窗口模式，那么只有在时间值在WDOG_WINH和WDOG_WINL之后，软件才能开始执行刷新序列。参见下图：

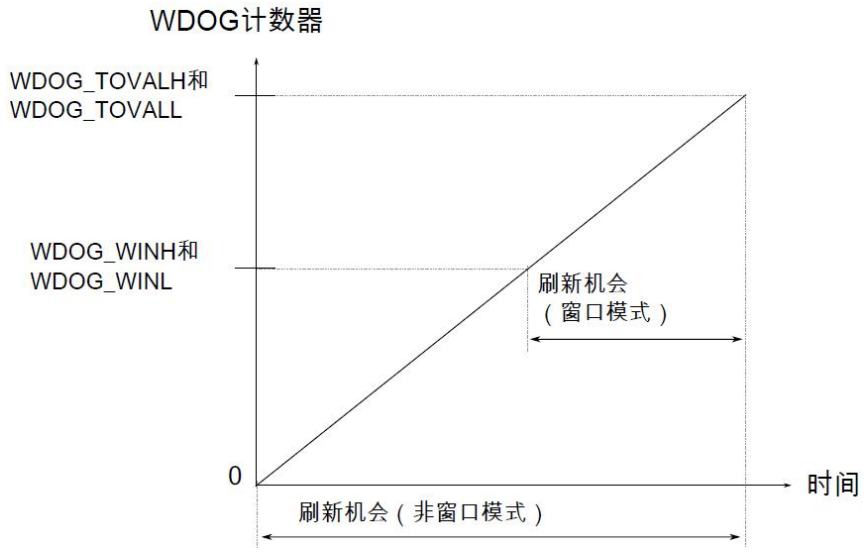


图 6-2 看门狗计数器刷新机会

❖ 6.2.5.1.1 窗口模式

软件完成其主控制处理比预期更快，则表明可能存在系统问题。根据用户应用的具体要求，可以使用WDOG 的窗口模式，在提前刷新看门狗的时候，强制系统复位。

窗口模式使能时，必须在计数器达到最小窗口时间值后刷新看门狗；否则，看门狗会复位MCU。最小窗口时间值在WDOG_WINH:L寄存器中指定。置位CS1[WIN]会使能窗口模式。

❖ 6.2.5.1.2 刷新看门狗

刷新写入序列为：先将 0x02A6写入WDOG_CNTH和WDOG_CNTL寄存器，然后再将0x80B4写入。写入0x80B4的操作必须在写入0x02A6后的16个总线时钟内执行；否则，看门狗会复位MCU。

注：在开始执行刷新序列前，先禁用全局中断。否则，如果产生中断，写入四个字节所需的时间就会长于16个总线时钟，那么中断实际上就能使刷新序列失效。在刷新序列完成后重新使能全局中断。

❖ 6.2.5.1.3 示例代码：刷新看门狗

下列代码段显示WDOG模块的刷新写入序列。
注：下列示例代码的组合情况为：8位WDOG_CNTH和WDOG_CNTL组合为16位WDOG_CNT；8位WDOG_TOVALH 和WDOG_TOVALL组合为16位WDOG_TOVAL；WDOG_WINH和WDOG_WINL组合为WDOG_WIN，并采用16位访问。

```
/* Refresh watchdog */
for (;;) // main loop
{
...
DisableInterrupts; // disable global interrupt
WDOG_CNT = 0x02A6; // write the 1st refresh word
WDOG_CNT = 0x80B4; // write the 2nd refresh word to refresh counter
EnableInterrupts; // enable global interrupt
...
}
```

● 6.2.5.2 配置看门狗

所有看门狗控制位、超时值和窗口值都只能在复位后一次写入。这表示，除非复位，否则它们的内容在写操作后无法更改。这就为配置看门狗提供了可靠的机制，确保软件跑飞情况无法误禁用或误修改已配置过的看门狗。

用户首先配置窗口和超时值，然后再配置其他控制位并确保CS1[UPDATE]也置0，从而实现上述功能。该新配置仅在复位后一次写入除WDOG_CNTH:L外的所有寄存器后生效。否则，WDOG默认使用复位值。如果未使用窗口模式（CS2[WIN]为0），那么不需要将数值写入WDOG_WINH:L来使新配置生效。

❖ 6.2.5.2.1 重新配置看门狗

某些情况下（比如支持bootloader功能时），用户可能想要在不先强制复位的情况下重新配置或禁用看门狗。这时可以通过复位后进行看门狗初始配置时将CS1[UPDATE]置1，从而使用户可采用执行解锁序列的方法，在任意时刻重新配置看门狗。（相反，如果CS1[UPDATE]保持0，重新配置看门狗的唯一方法是发起复位。）解锁序列与刷新序列类似，但使用不同的数值。

❖ 6.2.5.2.2 解锁看门狗

解锁序列是指在配置完看门狗之后，在16个总线时钟内先对WDOG_CNTH:L寄存器写入0x20C5，然后再写入0x28D9。完成解锁序列后，用户必须在128个总线时钟内重新配置看门狗。
注：由于重新配置看门狗需要128个总线时钟，因此在执行STOP或WAIT指令前、重新配置看门狗之后必须插入一些延迟。这样可确保看门狗的新配置在MCU进入低功耗模式之前生效。否则，可能无法将MCU从低功耗模式中唤醒。

❖ 6.2.5.2.3 代码示例：重新配置看门狗

下面的代码段显示了WDOG模块的重新配置示例。

```
/* Initialize watchdog with ~1-kHz clock source, ~1s time-out */
```

```
DisableInterrupts; // disable global interrupt
WDOG_CNT = 0x20C5; // write the 1st unlock word
WDOG_CNT = 0x28D9; // write the 2nd unlock word
WDOG_TOVAL = 1000; // setting timeout value
WDOG_CS2 = WDOG_CS2_CLK_MASK; // setting 1-kHz clock source
WDOG_CS1 = WDOG_CS1_EN_MASK; // enable counter running
EnableInterrupts; // enable global interrupt
```

● 6.2.5.3 时钟源

看门狗计数器有四个时钟源选项，可以通过对CS2[CLK]进行编程来选择：

- 总线时钟
- 运行频率约为1 kHz 的内部低功耗振荡器(LPO)（为默认源）
- 内部32kHz 时钟
- 外部时钟

这些选项允许软件选择独立于总线时钟的时钟源，以满足安全要求更严格的应用需要。如果总线时钟由于某些原因而暂停，那么使用非总线时钟的时钟源可确保看门狗计数器继续运行；参见备用复位 。

用于所有时钟源的可选固定预分频器可以实现较长的超时周期。CS2[PRES]置位后，时钟源先进行256倍预分频，随后对看门狗计数器进行计时。

下表总结了可用的不同看门狗超时周期。

参考时钟	预分频器	看门狗超时可用性
内部约 1 kHz （LPO）	直通式	约 1 ms – 65.5 s[1]
	÷256	约 256 ms – 16,777 s
内部 约 32 kHz	直通式	约31.25 μs – 2.048 s
	÷256	约 8 ms – 524.3 s
1 MHz（来自总线或外部）	直通式	1 μs – 65.54 ms
	÷256	256 μs – 6.777 s
20 MHz（来自总线或外部）	直通式	50 ns – 3.277 ms
	÷256	12.8 μs – 838.8 ms

1. 复位后的默认超时值约为 4ms。

● 6.2.5.4 使用中断延迟复位

中断使能(CS1[INT] = 1)时，看门狗在发生复位触发事件（如计数器超时或无效刷新尝试）时首先生成中断请求。看门狗延迟 128 个总线时钟的强制复位，从而允许中断服务例程(ISR)执行任务，如分析堆栈以调试代码。

● 6.2.5.5 备用复位

注：非总线时钟的时钟源必须用作计数器的参考时钟；否则，备用复位功能不可用。

备用复位功能是一种安全保护特性，在 WDOG 主逻辑丢失其时钟（总线时钟）而无法再监控计数器时，

独立生成复位。如果看门狗计数器连续两次溢出（没有强制复位），那么备用复位功能生效并生成复位。

● 6.2.5.6 调试模式及低功耗模式下的功能

默认情况下，看门狗在后台激活模式、等待模式或停止模式下不工作。但在以下情况下，看门狗在上述模式下会正常运行：

- 对于后台激活模式，置位 CS1[DBG]。（通过这种方式，即使 CPU 状态由调试模块保持，看门狗也可在后台激活模式下工作。）
- 对于等待模式，置位 CS1[WAIT]。
- 对于停止模式，置位 CS1[STOP]。

注：(1)即使 CS1[STOP]已置位，看门狗也无法在停止模式下生成中断，并且不会从停止模式唤醒 MCU。它在停止模式期间可生成复位。

(2)对于后台激活模式和停止模式，除了上述配置外，必须使用总线时钟以外的其他时钟源作为计数器的参考时钟；否则，看门狗无法运行。

● 6.2.5.7 看门狗快速测试

在注重安全的应用中，执行应用程序代码之前，用户需要测试看门狗是否按预期进行工作并复位 MCU。对 16 位计数器进行完全测试的方法是使其运行至溢出值，这会花费相对较长的时间（64k 个时钟周期）。

为有助于最大限度降低复位后应用程序代码的启动延迟，看门狗提供了一种测试方法，即通过将计数器分成字节宽度的各级，来更快速地测试看门狗。低位和高位字节独立运行，并针对超时值寄存器的对应字节执行超时测试。（为了全面测试计数器的高位字节，测试特性通过低位字节的第 8 位将信号馈送到输入时钟，从而确保低位字节到高位字节的溢出连接经过测试。）

使用该测试特性可将测试时间缩短至 512 个时钟（不包括用户配置和复位向量提取等开销）。为进一步加快测试速度，可以使用速度更快的时钟（如总线时钟）作为计数器参考。

上电复位时，系统复位寄存器中的 POR 位置位，指示用户应当执行 WDOG 快速测试。

✧ 6.2.5.3.1 测试计数器的每一个字节

测试流程遵照如下步骤：

1. 看门狗配置时间周期内，在 WDOG_TOVALH 和 WDOG_TOVALL 寄存器中写入期盼的数值。
2. 选择待测试的计数器的字节：对于低位字节，使用 WDOG_CS1[TST] = 10b；对于高位字节，使用 WDOG_CS1[TST] = 11b。
3. 等待看门狗超时。或者在闲置循环中，使 RAM 位置增量，以便用作后续比较的并行软件计数器。由于 RAM 不受看门狗复位影响，因此看门狗计数器的超时周期能与软件计数器作比较，以验证超时周期是否如预期般发生。
4. 看门狗计数器超时并强制复位。
5. 确认系统复位寄存器中的 WDOG 标志已置位，指示看门狗导致复位。（POR 标志位保持清零状态。）
6. 确认 WDOG_CS1[TST]显示测试（10b 或 11b）已执行。

若得到确认，计数功能和比较功能对选定的字节起作用。重复该流程，选择步骤 2 中的其他字节。

注：WDOG_CS1[TST]仅通过 POR 清零，并且不受其他复位影响。

✧ 6.2.5.3.2 进入用户模式

顺利完成对看门狗计数器低位和高位字节的测试后，用户可将 WDOG_CS1[TST]配置为 01b，以指示看门狗准备用于应用用户模式。因此，如果再次发生复位，软件可将复位触发视为软件跑飞或故障应用代码导致的真正看门狗复位。

作为一个持续进行的测试，在使用默认的 1kHz 时钟源时，软件可定期对 WDOG_CNTH 和 WDOG_CNTL 寄存器进行读操作以确保计数器处于递增状态。

■ 6.3 Enhance Timer 模块(ETM)

■ 6.3.1 简介

NV32系列的Enhance Timer模块(ETM)是一种两通道至六通道的多功能定时器，而NV32F100x系列型号的单个ETM模块最多有6个通道，支持输入捕捉、输出比较、PWM信号输出、定时中断、脉冲加减计数、脉冲周期宽度测量等功能。ETM的时间基准来自一个值可读取的16位计数器，可用作无符号计数器，也可作为有符号的补码计数器。

■ 6.3.2 ETM 的基本理念

NV32F100x 的 ETM 基于一个简单的定时器，共有 ETM0，ETM1，ETM2 三个独立的 ETM 模块。其中 ETM0 和 ETM1 各只有 2 个通道，而 ETM2 有 6 个通道，ETM2 可用于电机或舵机的 PWM 输出。如果同时使能 3 个 ETM 模块，最多可以 10 个通道有效，但 NV32F100x 的 3 个 ETM 模块都不具备正交解码功能。ETM 在功能方面实现了延伸，可更好满足电机控制、数字照明解决方案及功率转换的需求。其增强的主要功能如下：

- 带符号的向上计数器
- 支持硬件死区插入
- 故障控制输入
- 增强型触发功能
- 初始化和极性控制

已通过一组专用的寄存器添加了电机控制和功率转换特性，所有全新特性均默认为关断。硬件死区插入、极性控制、故障控制以及输出转换和屏蔽等全新特性，可极大地减轻执行软件的负担，其中每一项特性通常都由一组寄存器来控制。

ETM 输入触发源可从比较器、ADC 或其他子模块来自动启动定时器功能。这些触发源在子模块集成期间可以各种方式链接，所以需要使用适用于当前 ETM 配置的触发源选项。

假设每个 ETM 中的初始化、输入时钟、初始和最终计数值都相同，则通过同步多个 ETM 模块（计数器一致增加），可以得到计数器更大的定时器。

所有主要的用户访问寄存器都经过缓冲，以缓解当前运行软件的负载。根据用户定义的数据，通过触发源各可用选项可以确定需要更新的寄存器。

■ 6.3.3 特性

ETM 特性包括：

- ETM 时钟源是可选的
  - 时钟源可以是系统时钟、固定频率时钟或外部时钟（ETMx_CLK 引脚接入）
  - 固定频率时钟是一种附加时钟输入，允许在系统时钟以外选择一个片上时钟源

- 选择外部时钟可将 ETM 时钟连接到芯片级输入引脚，从而能够将 ETM 计数器与片外时钟源同步
- 预分频器（1、2、4、8、16、32、64 或 128 分频）
- 16 位计数器
  - 可以是自由运行的计数器，也可以是采用初始和最终值的计数器
  - 计数可以是向上计数或者先向上后向下计数
- 每个通道均可配置为输入捕捉、输出比较或边沿对齐 PWM 模式
- 在输入捕捉模式下：
  - 捕捉可以在上升沿、下降沿或两个边沿同时发生
  - 可为某些通道选择一个输入滤波器
- 在输出比较模式下，可以对输出信号采取置位、清除或匹配切换操作
- 所有通道均可配置为中心对齐 PWM 模式
- 每对通道均可组合起来生成一个 PWM 信号，并且独立控制 PWM 信号的两个边沿
- ETM 通道可采用具有同等输出或互补输出的成对工作方式，或者作为独立的通道独立输出信号
- 死区插入可用于每一对互补通道
- 生成匹配触发器
- 软件控制 PWM 输出
- 对于全局故障控制最多有 4 个故障输入
- 每个通道的极性都是可配置的
- 按通道生成中断
- 当计数器溢出时生成中断
- 检测到故障条件时生成中断
- 同步加载写缓冲 ETM 寄存器
- 关键寄存器的写入保护
- 输入捕捉测试可检测固定为零和固定为一的状况
- 双边沿捕捉可以用于脉冲和周期信号的宽度测量

■ 6.3.4 操作模式

当 MCU 处于有效调试模式时，ETM 会临时挂起所有计数，直到 MCU 返回正常用户工作模式。在停止模式下，所有 ETM 输入时钟都将停止，因此，在时钟恢复前，ETM 会一直有效地禁用。在等待模式下，ETM 继续正常工作。如果 ETM 无需生成实时参考或提供将 MCU 从等待模式唤醒所需的中断源，则可在进入等待模式之前通过禁用 ETM 功能省电。

■ 6.3.5 结构框图

ETM 为每个通道使用一个输入/输出(I/O)，CHn（ETM 通道(n)）中的 n 为通道编号(0－5)。
下图 6-3 展示了 ETM 结构。ETM 模块的核心组件是一个初始值和终值可编程的 16 为计数器，该计数器可向上或者先向上后向下计数。

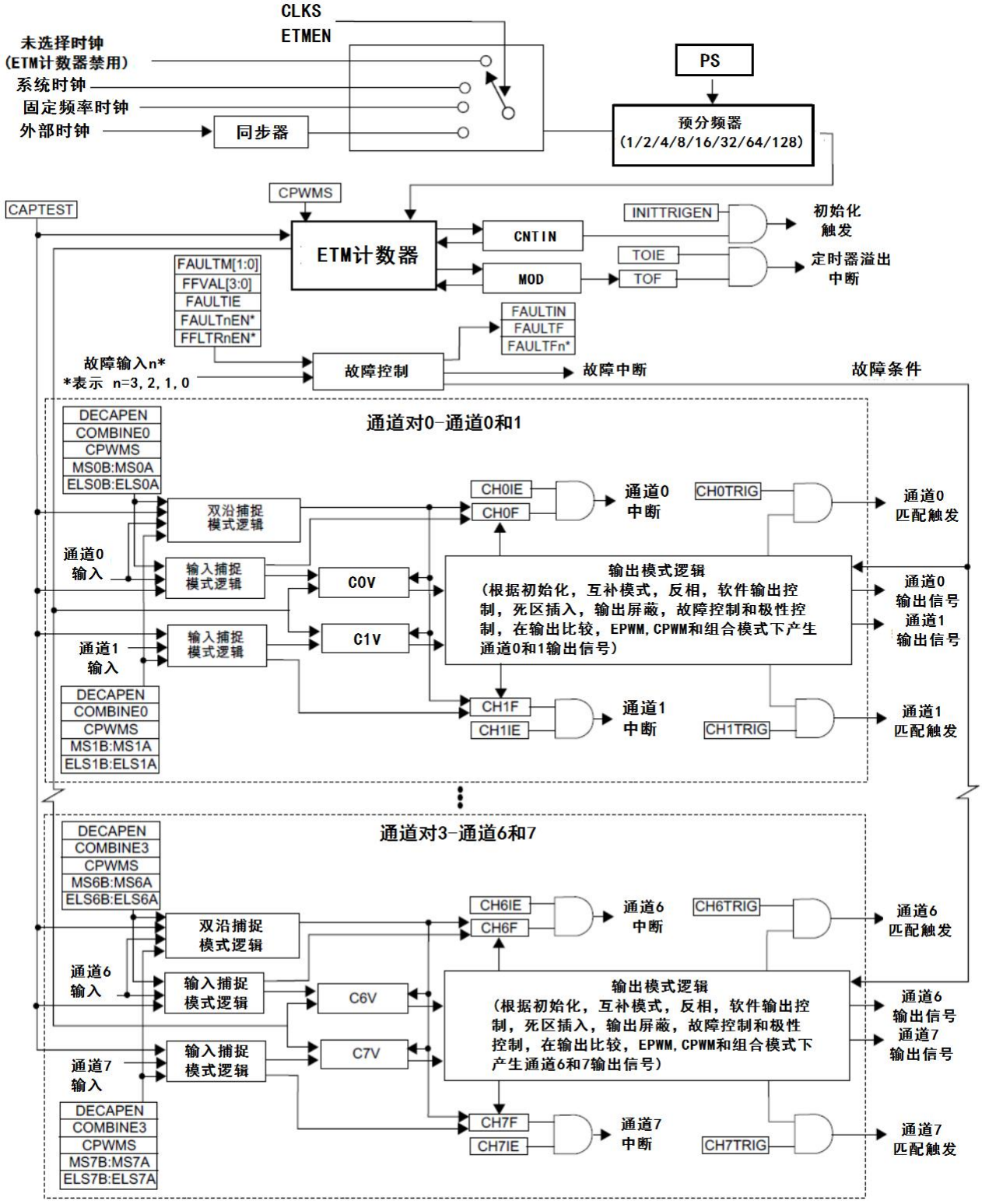


图 6-3 ETM 结构框图

■ 6.3.6 ETM 信号说明

表 6-11 显示了用户可访问的ETM信号。

表 6-11 ETM 信号说明

信号	说明	I/O	功能
EXTCLK	外部时钟。可选择 ETM 外部时钟用来驱动 ETM 计数器。	I	如果通过 SC 寄存器中的 CLKS[1:0]位选择了外部时钟输入信号，则该信号可用作 ETM 计数器时钟。该时钟信号不得超过系统时钟频率的 1/4。选择了外部时钟后，还会用到 ETM 计数器预分频器的选择和设置功能。
CHn	ETM 通道(n)，其中 n 可以是 5-0	I/O	每个 ETM 通道均可配置为输入或输出。每个通道（无论是输入还是输出）关联的方向是根据分配给该通道的模式选择的。
FAULTj	故障输入(j)，其中 j 可以是 3-0	I	故障输入信号用于控制 CHn 通道输出状态。如果检测到故障，FAULTj 信号变为有效，且通道输出进入安全状态。故障逻辑的行为方式由 MODE 寄存器中的 FAULTM[1:0]控制位和 COMBINEm 寄存器中的 FAULTEN 位定义。注意，由于 FAULTM[1:0]和 FAULTEN 控制位是为每一对通道定义的，因此每个 FAULTj 输入都有可能有选择地影响所有通道。因为有多个 FAULTj 输入（ETM2 模块最多有 2 个），所以这些输入中的每一个都会由 FLTCTRL 寄存器中的 FAULTjEN 位激活。

■ 6.3.7 存储器映射和寄存器说明

● 6.3.7.1 存储器映射

本节从较高层面概要介绍了ETM寄存器以及这些寄存器的映射方式。
ETM中不可用功能的寄存器和比特仍然保留在存储器映射中以及复位值中，但不具备有效功能。
注：ETMEN = 0时，请勿在CNTIN寄存器到PWMLOAD寄存器这一区域内进行写入操作。

● 6.3.7.2 寄存器说明

表 6-12 ETM 存储器映射

绝对地址 (16 进制)	寄存器名称	偏移量	位宽	权限	复位值
4003_8000	状态和控制寄存器(ETM0_SC)	00h	32	R/W	0000_0000h
4003_8004	计数器寄存器(ETM0_CNT)	04h	32	R/W	0000_0000h
4003_8008	模数寄存器(ETM0_MOD)	08h	32	R/W	0000_0000h
4003_800C	通道(n)状态和控制寄存器(ETM0_C0SC)	0Ch	32	R/W	0000_0000h
4003_8010	通道(n)值寄存器(ETM0_C0V)	10h	32	R/W	0000_0000h
4003_8014	通道(n)状态和控制寄存器(ETM0_C1SC)	14h	32	R/W	0000_0000h
4003_8018	通道(n)值寄存器(ETM0_C1V)	18h	32	R/W	0000_0000h
4003_9000	状态和控制寄存器(ETM1_SC)	00h	32	R/W	0000_0000h
4003_9004	计数器寄存器(ETM1_CNT)	04h	32	R/W	0000_0000h

4003_9008	模数寄存器(ETM1_MOD)	08h	32	R/W	0000_0000h
4003_900C	通道(n)状态和控制寄存器(ETM1_C0SC)	0Ch	32	R/W	0000_0000h
4003_9010	通道(n)值寄存器(ETM1_C0V)	10h	32	R/W	0000_0000h
4003_9014	通道(n)状态和控制寄存器(ETM1_C1SC)	14h	32	R/W	0000_0000h
4003_9018	通道(n)值寄存器(ETM1_C1V)	18h	32	R/W	0000_0000h
4003_A000	状态和控制寄存器(ETM2_SC)	00h	32	R/W	0000_0000h
4003_A004	计数器寄存器(ETM2_CNT)	04h	32	R/W	0000_0000h
4003_A008	模数寄存器(ETM2_MOD)	08h	32	R/W	0000_0000h
4003_A00C	通道(n)状态和控制寄存器(ETM2_C0SC)	0Ch	32	R/W	0000_0000h
4003_A010	通道(n)值寄存器(ETM2_C0V)	10h	32	R/W	0000_0000h
4003_A014	通道(n)状态和控制寄存器(ETM2_C1SC)	14h	32	R/W	0000_0000h
4003_A018	通道(n)值寄存器(ETM2_C1V)	18h	32	R/W	0000_0000h
4003_A01C	通道(n)状态和控制寄存器(ETM2_C2SC)	1Ch	32	R/W	0000_0000h
4003_A020	通道(n)值寄存器(ETM2_C2V)	20h	32	R/W	0000_0000h
4003_A024	通道(n)状态和控制寄存器(ETM2_C3SC)	24h	32	R/W	0000_0000h
4003_A028	通道(n)值寄存器(ETM2_C3V)	28h	32	R/W	0000_0000h
4003_A02C	通道(n)状态和控制寄存器(ETM2_C4SC)	2Ch	32	R/W	0000_0000h
4003_A030	通道(n)值寄存器(ETM2_C4V)	30h	32	R/W	0000_0000h
4003_A034	通道(n)状态和控制寄存器(ETM2_C5SC)	34h	32	R/W	0000_0000h
4003_A038	通道(n)值寄存器(ETM2_C5V)	38h	32	R/W	0000_0000h
4003_A04C	计数器初始值寄存器(ETM2_CNTIN)	4Ch	32	R/W	0000_0000h
4003_A050	捕捉和比较状态寄存器(ETM2_STATUS)	50h	32	R/W	0000_0000h
4003_A054	特性模式选.寄存器(ETM2_MODE)	54h	32	R/W	0000_0004h
4003_A058	同步寄存器(ETM2_SYNC)	58h	32	R/W	0000_0000h
4003_A05C	通道输出的初始状态寄存器(ETM2_OUTINIT)	5Ch	32	R/W	0000_0000h
4003_A060	输出屏蔽寄存器(ETM2_OUTMASK)	60h	32	R/W	0000_0000h
4003_A064	已连接通道功能寄存器(ETM2_COMBINE)	64h	32	R/W	0000_0000h
4003_A068	死区时间插入控制寄存器(ETM2_DEADETIME)	68h	32	R/W	0000_0000h
4003_A06C	ETM 外部触发寄存器(ETM2_EXTTRIG)	6Ch	32	R/W	0000_0000h
4003_A070	通道极性寄存器(ETM2_POL)	70h	32	R/W	0000_0000h
4003_A074	故障模式状态寄存器(ETM2_FMS)	74h	32	R/W	0000_0000h
4003_A078	输入捕捉滤波器控制寄存器(ETM2_FILTER)	78h	32	R/W	0000_0000h
4003_A07C	故障控制寄存器(ETM2_FLTCTRL)	7Ch	32	R/W	0000_0000h
4003_A084	配置寄存器(ETM2_CONF)	84h	32	R/W	0000_0000h
4003_A088	ETM 故障输入极性寄存器(ETM2_FLTPOL)	88h	32	R/W	0000_0000h
4003_A08C	同步配置寄存器(ETM2_SYNCONF)	8Ch	32	R/W	0000_0000h
4003_A090	ETM 反相控制寄存器(ETM2_INVCTRL)	90h	32	R/W	0000_0000h
4003_A094	ETM 软件输出控制寄存器(ETM2_SWOCTRL)	94h	32	R/W	0000_0000h
4003_A098	ETM PWM 装载寄存器(ETM2_PWMLOAD)	98h	32	R/W	0000_0000h

● 6.3.7.3 状态和控制寄存器 (ETMx_SC)

状态和控制寄存器含有溢出状态标志和控制位，用来配置中断使能、ETM、时钟源和预分频系数。这些控制与模块内的全部通道有关。

地址：基址 + 0h 偏移 ：ETM0_SC = 4003_8000h / ETM1_SC = 4003_9000h / ETM2_SC = 4003_A000h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0								TOF	TOIE	CPWMS	CLKS		PS		
写									0							
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-13 ETMx_SC 字段描述

位	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
7 TOF	定时器溢出标志 当 ETM 计数器达到 MOD 寄存器中的值.，通过硬件置位。在 TOF 置位的情况下读取 SC 寄存器，然后向 TOF 位写入 0，可清零 TOF 位。将 1 写入 TOF 无效。 若在读取和写入操作之间发生另一次 ETM 溢出，则写操作无效；因此，TOF 保持置位状态，表示发生了溢出。在此情况下，TOF 中断请求不会因为之前的 TOF 清零序列而丢失。 0 ETM 计数器未溢出。 1 ETM 计数器已溢出。
6 TOIE	定时器溢出中断使能 使能 ETM 溢出中断 0 禁用 TOF 中断。使用软件轮询。 1 使能 TOF 中断。TOF 等于 1 时，产生中断。
5 CPWMS	中心对齐 PWM 选择 选择 CPWM 模式。该模式配置 ETM 在先增后减计数模式中的操作。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 ETM 计数器可工作在向上计数模式中。 1 ETM 计数器可工作在先增后减计数模式中。

4-3 CLKS	时钟源选择 从三个 ETM 计数器时钟源选择。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 00 未选定时钟，该设置生效后可禁用 ETM 计数器。 01 系统时钟 10 固定频率时钟 11 外部时钟
2-0 PS	预分频系数选择 从 8 个分频系数中选择，用于 CLKS 所选择的时钟源。新的预分频系数将会在新数值更新至寄存器位后，于下一个系统时钟周期影响时钟源。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。

000	1 分频
001	2 分频
010	4 分频
011	8 分频
100	16 分频
101	32 分频
110	64 分频
111	128 分频

● 6.3.7.4 计数器寄存器 (ETMx_CNT)

CNT 寄存器含有ETM计数器值。
复位操作可清零CNT寄存器。向COUNT写入任何值都将用初始值CNTIN更新计数器。
处于调试状态时，ETM 计数器处于冻结状态。这是您可能读取的值。

地址: 基址 + 4h 偏移 ：ETM0_CNT = 4003_8004h / ETM1_CNT = 4003_9004h / ETM2_CNT = 4003_A004h

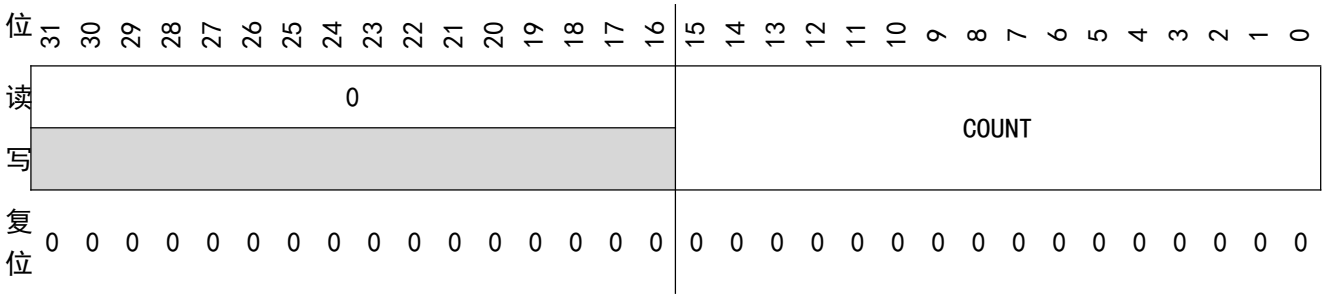


表 6-14 ETMx_CNT 字段描述

位	描述
31-16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
15-0 COUNT	计数器值

● 6.3.7.5 模数寄存器 (ETMx_MOD)

模数寄存器含有 ETM 计数器的模数值。ETM 计数器到达模数值后，溢出标志(TOF)在下一个时钟变成置位状态，且 ETM 计数器的下一个数值取决于选定的计数方式；参见计数器 。

写入 MOD 寄存器可将数值锁存到缓冲区中。将根据通过写缓存更新的寄存器用 MOD 寄存器的写缓冲器值更新此 MOD 寄存器。

如果 ETMEN = 0，可通过写入到 SC 寄存器（无论是否处于调试状态）手动复位此写入一致性机制。在写入到 MOD 寄存器之前，通过写 CNT 以初始化 ETM 计数器，从而避免对计数器首次发生溢出的时间造成混淆。

地址: 基址 + 8h 偏移 ：ETM0_MOD = 4003_8008h / ETM1_MOD = 4003_9008h / ETM2_MOD = 4003_A008h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	保 留																MOD															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-15 ETMx_MOD 字段描述

位	描述
31-16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
15-0 MOD	模数值

● 6.3.7.6 通道(n)状态和控制寄存器 (ETMx_CnSC)

CnSC 含有通道中断状态标志和控制位，用来配置中断使能、通道和引脚功能。

表 6-16 模式、边沿和电平选择

DECAPEN	COMBINE	CPWMS	MSnB:MsnA	ELSnB:ELSnA	模式	配置
X	X	X	XX	0	引脚未用于 ETM—通道引脚变回通用 I/O 或其他外设控制用途	
0	0	0	0	1	输入捕捉	仅在上升沿捕捉
				10		仅在下降沿捕捉
				11		在上升沿或下降沿捕捉
			1	1	输出比较	匹配时切换输出
				10		匹配时清零输出
				11		匹配时置位输出

			1X	10	边沿对齐 PWM	高电平真脉冲（匹配时清零输出）
				X1		低真脉冲（匹配时置位输出）
		1	XX	10	中心对齐 PWM	高真脉冲（向上匹配时清零输出）
				X1		低真脉冲（向上匹配时置位输出）
	1	0	XX	10	组合 PWM	高真脉冲（通道（n）匹配时置位，通道（n+1）匹配时清零）
				X1		低真脉冲（通道（n）匹配时清零，通道（n+1）匹配时置位）
1	0	0	X0	参见下表	双沿捕捉	单次捕捉模式
			X1			持续捕捉模式

表 6-17 双沿捕捉模式——边沿极性选择

ELSnB	ELSnA	通道端口使能	已检测边沿
0	0	禁用	无边沿
0	1	使能	上升沿
1	0	使能	下降沿
1	1	使能	上升沿和下降沿

表 6-18 绝对地址如下表所示（16 进制）

绝对地址	寄存器名称	绝对地址	寄存器名称	绝对地址	寄存器名称
4003_800C	ETM0_C0SC	4003_900C	ETM1_C0SC	4003_A00C	ETM2_C0SC
4003_8014	ETM0_C1SC	4003_9014	ETM1_C1SC	4003_A014	ETM2_C1SC
4003_801C	ETM2_C2SC	4003_902C	ETM2_C4SC		
4003_8024	ETM2_C3SC	4003_9034	ETM2_C5SC		

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0								CHF	CHIE	MSB	MSA	ELSB	ELSA	0	0
写									0							
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-19 ETMx_CnSC 字段描述

位	描述

31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
7 CHF	通道标志 通道上有事件发生时，通过硬件置位，在 CHnF 置位的情况下读取 CSC 寄存器，然后向 CHF 位写入 0，可清零 CHF。将 1 写入 CHF 无效。若在读取和写入操作之间发生另一次事件，则写操作无效；因此，CHF 保持置位状态，表示发生件发生了一次事。在此情况下，CHF 中断请求不会因为之的 CHF 清零序列而丢失。 0 未发生通道事件。 1 发生通道事件。
6 CHIE	通道中断使能。 使能通道中断。 0 禁用通道中断。使用软件轮询。 1 使能通道中断。
5 MSB	通道模式选择 用来在通道逻辑中进一步选择。其功能取决于通道模式。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。
4 MSA	通道模式选择 用来在通道逻辑中进一步选择。其功能取决于通道模式。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。
3 ELSB	边沿或电平选择 ELSB 和 ELSA 功能取决于通道模式。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。
2 ELSA	边沿或电平选择 ELSB 和 ELSA 功能取决于通道模式。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。
1-0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为0.

● 6.3.7.7 通道（n）值寄存器（ETMx_CnV）

这些寄存器含有捕捉的 ETM 的计数器值，用于输入模式；或者含有匹配值，用于输出模式。在输入捕捉、捕捉测试和双沿捕捉模式下，任何 CnV 寄存器的写操作均被忽略。在输出模式下，写入 CnV 寄存器可将数值锁存到缓冲区中。将根据通过写缓存更新的寄存器用 CnV 寄存器的写入缓冲器值更新此 CnV 寄存器。如果 ETMEN=0， 可通过写入到 CnSC 寄存器（无论是否处于调试状态）手动复位此写入一致性机制。

表 6-20 绝对地址如下表所示（16 进制）

绝对地址	寄存器名称	绝对地址	寄存器名称	绝对地址	寄存器名称
4003_8010	ETM0_C0V	4003_9010	ETM1_C0V	4003_A010	ETM2_C0V
4003_8018	ETM0_C1V	4003_9018	ETM1_C1V	4003_A018	ETM2_C1V
4003_A020	ETM2_C2V	4003_A030	ETM2_C4V		
4003_A028	ETM2_C3V	4003_A038	ETM2_C5V		

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0																VAL															
写																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-21 ETMx_CnV 字段描述

位	描述
31-16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
15-0 VAL	通道值 输入模式下捕捉的 ETM 计数器值或用于输出模式的匹配值

● 6.3.7.8 计数器初始值寄存器 (ETM2_CNTIN)

计数器初始值寄存器包含用于 ETM 计数器的初始数值。写入 CNTIN 寄存器可将数值锁存到缓冲区中。CNTIN 寄存器根据通过写缓存更新的寄存器 所示，更新为其写入缓冲区的内容。

初始选定 ETM 时钟时，通过向 CLKS 位写入非零值，会让 ETM 计数器以数值 0x0000 起始。为了避免发生这种情况，在首次写入以选择 ETM 时钟前，向 CNTIN 寄存器写入新数值，然后向 CNT 寄存器写入任意值可初始化 ETM 计数器。

地址：基址 + 4Ch 偏移 ：ETM2_CNTIN = 4003_A04Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	保留																INIT															
写																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-22 ETM2_CNTIN 字段描述

位	描述
31-16 保留	此字段为保留字段。
15-0 INIT	ETM 计数器初始值

● 6.3.7.9 捕捉和比较状态寄存器 (ETM2_STATUS)

状态寄存器含有 CnSC 中状态标志位 CHnF 的副本，可用于各 ETM 通道，为软件提供方便。

状态寄存器中每一个 CHnF 位都是 CnSC 中 CHnF 位的镜像。对状态寄存器进行一次读取，即可检查所有 CHnF 位。读取状态寄存器，然后向其写入 0x00，即可清零所有 CHnF 位。

通道上发生事件时，硬件将置位各通道标志。在 CHnF 置位的情况下，读取状态寄存器，然后向 CHnF 位写入 0，可清零 CHnF。将 1 写入 CHnF 无效。

若在读取和写入操作之间发生另一次事件，则写操作无效；因此，CHnF 保持置位状态，表示发生了一次事件。在此情况下，CHnF 中断请求不会因为之前的 CHnF 清零序列而丢失。

注：应当仅在组合模式下使用状态寄存器

地址：基址+50h 偏移：ETM2_STATUS = 4003_A050h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0										CH5F	CH4F	CH3F	CH2F	CH1F	CH0F
写											0	0	0	0	0	0
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-23 ETM2_STATUS 字段描述

位	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
5-0 CHnF	通道 n 标志 （n = 5 到 0） 0 未发生通道事件。 1 发生通道事件。

● 6.3.7.10 特性模式选择寄存器 (ETM2_MODE)

该寄存器含有 ETM 特有的全局使能位，以及用来配置的控制位

- 故障控制模式和中断
- 捕捉测试模式
- PWM 同步
- 写保护
- 通道输出初始化

这些控制与模块内的全部通道有关。

地址: 基址+54h 偏移 ： ETM2_MODE = 4003_A054h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0								FAULTIE	FAULTM		CAPTEST	PWMSYNC	WPDIS	INIT	ETMEN
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-24 ETM2_MODE 字段描述

位	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
7 FAULTIE	故障中断使能 ETM 检测到故障，且 ETM 故障控制使能时，使能中断生成。 0 故障控制中断禁用。 1 故障控制中断使能。
6-5 FAULTM	故障控制模式 定义 ETM 故障控制模式。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。
	00 所有通道禁用故障控制。 01 仅针对偶数通道（通道 0、2 和 4）使能故障控制，所选模 1 式为手动故障清零。 10 针对所有通道使能故障控制，所选模式为手动故障清零。 11 针对所有道具使能故障控制，所选模式为自动故障清零。
4 CAPTEST	捕捉测试模式使能 使能捕捉测试模式 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 捕捉测试模式禁用。 1 捕捉测试模式使能。
3 PWMSYNC	PWM 同步模式 选择可用于 MOD、CnV、OUTMASK 和 ETM 计数器同步的触发。参见 PWM 同步。当 SYNCMODE 为 0 时，PWMSYNC 位进行同步配置。 0 无限制。选择可用于 MOD、CnV、OUTMASK 和 ETM 计数器同步的软件和硬件触发。 1 软件触发只能用于 MOD 和 CnV 同步，硬件触发只能用于 OUTMASK 和 ETM 计数器同步。
2 WPDIS	写保护禁用 写保护使能时（WPDIS=0），无法向写保护位写入内容。写保护禁用后（WPDIS=1），可以写入写保护位。WPDIS 位与 WPEN 位相反。向 WPEN 写入 1 可清零 WPDIS。WPDIS 置位：WPEN 位读取为 1，然后向 WPDIS 写入 1。向 WPDIS 中写入 0 无效。 0 写保护使能。 1 写保护禁用。
1 INIT	初始化通道输出 INIT 位写入 1 后，通道输出根据 OUTINIT 寄存器中的对应位状态进行初始化。将 0 写入 INIT 位无效。 INIT 位始终读取为 0。
0 ETMEN	ETM 使能 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 ETM 专用寄存器无效。 1 ETM 专用寄存器有效。

● 6.3.7.11 同步寄存器（ETM2_SYNC）

该寄存器配置 PWM 同步性能。

同步事件可采用 MOD、CV 和 OUTMASK 寄存器的写入缓冲区数值和 ETM 计数器初始化数值执行同步更新。

注：当 SYNCMODE =0 时，同时使用软件触发（SWSYNC 位）和硬件触发（TRIG0、TRIG1 和 TRIG2 位）可能会产生冲突。同一时间应当仅使用硬件或软件触发（而非两者同时使用），否则可能产生无法预测的结果。选择加载点（CNTMAX 和 CNTMIN 位）以便在全部使能通道中同时提供 MOD、CNTIN 和 CnV 寄存器更新。使用加载点选择配合 SYNCMODE =0 和硬件触发选择（TRIG0、TRIG1 或 TRIG2 位）可能产生不可预测的结果。同步事件选择还取决于 PWMSYNC（MODE 寄存器）和 SYNCMODE（SYNCONF 寄存器）位。参见 PWM 同步。

地址：基址+58h 偏移 ：ETM2_SYNC = 4003_A058h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0								SWSYNC	TRIG2	TRIG1	TRIG0	SYNCHOM	REINT	CNTMAX	CNTMIN
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-25 ETM2_SYNC 字段描述

位	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
7 SWSYNC	PWM 同步软件触发 选择软件触发作为 PWM 同步触发。向 SWSYNC 位写入 1 时，发生软件触发。 0 软件触发未选定。 1 软件触发已选定。
6 TRIG2	PWM 同步硬件触发 2 使能 PWM 同步的硬件触发 2 在触发 2 输入信号中检测到上升沿时，产生硬件触发 2。 0 触发禁用 1 触发使能
5 TRIG1	PWM 同步硬件触发 1 使能 PWM 同步的硬件触发 1 在触发 1 输入信号中检测到上升沿时，产生硬件触发 1。 0 触发禁用 1 触发使能
4 TRIG0	PWM 同步硬件触发 0 使能 PWM 同步的硬件触发 0 在触发 0 输入信号中检测到上升沿时，产生硬件触发 0。 0 触发禁用 1 触发使能

3 SYNCHOM	输出屏蔽同步 选择 OUTMASK 寄存器何时以其缓冲区的数值更新。 0 OUTMASK 寄存器将在所有系统时钟的上升沿以其缓冲区的数值进行更新。 1 OUTMASK 寄存器仅通过 PWM 同步以其缓冲区值进行更新
2 REINIT	通过同步对 ETM 计数器进行重新初始化（ETM 计数器同步） 检测到用于同步的特定触发时，确定 ETM 计数器是否重新初始化。当 SYNCMODE 为 0 时，REINIT 位进行同步配置。 0 ETM 计数器继续正常计数 1 当检测到特定的触发时，ETM 计数器以其初始值更新。
1 CNTMAX	最大加载点使能 选择 PWM 同步的最大加载点，参见边界周期和加载点。若 CNTMAX 为 1，则选定的加载点为 ETM 计数器达到其最大值（MOD 寄存器）的时刻。 0 最大加载点禁用 1 最大加载点使能
0 CNTMIN	最小加载点使能 选择 PWM 同步的最小加载点，参见边界周期和加载点。若 CNTMIN 为 1，则选定的加载点为 ETM 计数器达到其最小值（CNTMIN 寄存器）的时刻。 0 最小加载点禁用 1 最小加载点使能

● 6.3.7.12 通道输出的初始状态寄存器(ETM2_OUTINIT)

地址:基址+5Ch 偏移：ETM2_OUTINIT = 4003_A05Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0										CH5OI	CH4OI	CH3OI	CH2OI	CH1OI	CH0OI
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-26 ETM2_OUTINIT 字段描述

位	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
5-0 CHnOI	通道 n 输出初始值 (n = 5 到 0) 进行初始化时，选择强制进入通道输出的数值。 0 初始值为 0。

	1 初始值为 1。

● 6.3.7.13 输出屏蔽寄存器(ETM2_OUTMASK)

该寄存器为每个 ETM 通道提供屏蔽。通道屏蔽确定其输出是否进行响应；也就是说，当发生匹配时，它是否屏蔽。该特性可用于 BLDC 控制；这类应用在特定时间内，电机上存在 PWM 信号，以提供电子换向。对 OUTMASK 寄存器采取的任意写操作都会将数值保存在其写入缓冲区中。寄存器根据 PWM 同步所示，更新为其写入缓冲区的内容。

地址:基址+60h 偏移：ETM2_OUTMASK = 4003_A060h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0										CH5OM	CH4OM	CH3OM	CH2OM	CH1OM	CH0OM
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-27 ETM2_OUTMASK 字段描述

位	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
5-0 CHnOM	定义通道 n 输出为屏蔽或未屏蔽 (n = 5 到 0) 0 通道输出未屏蔽，它连续正常工作。 1 通道输出已屏蔽，它被强制处于不活跃状态。

● 6.3.7.14 已连接通道功能寄存器(ETM2_COMBINE)

该寄存器包含控制位，可配置故障控制、同步、死区时间插入、双沿捕捉模式、互补和组合模式。用于通道对(n)和(n+1)，其中 n 等于 0、2 和 4。

地址:基址+ 64h 偏移：ETM2_COMBINE = 4003_A064h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0									FAULTEN2	SNCEN2	DTEN2	DECAP2	DECAPEN2	COMP2	COMBINE2
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0	FAULTEN1	SYNCEN1	DTEN1	DECAP1	DECAPEN1	COMP1	COMBINE1	0	FAULTEN0	SYNCEN0	DTEN0	DECAP0	DECAPEN0	COMP0	COMBINE0
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-28 ETM2_COMBINE 字段描述

位	描述
31-23 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
22 FAULTEN2	故障控制使能(n=4) 使能通道(n)和(n+1)中的故障控制。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 禁用该通道对中的故障控制 1 使能该通道对中的故障控制
21 SYNCEN2	同步使能(n=4) 使能寄存器 C(n)V 和 C(n+1)V 的 PWM 同步。 0 禁用该通道(n)和(n+1)中的 PWM 同步。 1 使能该通道(n)和(n+1)中的 PWM 同步。
20 DTEN2	死区时间使能(n=4) 使能通道(n)和(n+1)中的死区时间插入。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 禁用该通道对中的死区时间插入 1 使能该通道对中的死区时间插入。
19 DECAP2	双沿捕捉模式捕捉(n=4) 使能根据通道(n)输入事件捕捉 ETM 计数器值和双沿捕捉位配置。 该字段仅在 ETMEN=1 且 DECAPEN=1 时适用。 若选定双沿捕捉—单次模式，且完成通道(n+1)事件捕捉，则 DECAP 位由硬件自动清零。 0 双沿捕捉未激活。 1 双沿捕捉已激活。
18 DECAPEN2	双沿捕捉模式使能(n=4) 使能通道(n)和(n+1)的双沿捕捉模式。该位根据表 6—17，重新在双沿捕捉模式下配置 MSnA、ELSnB:ELSnA 和 ELS(n+1)B:ELS(n+1)A 位的功能。该字段仅在 ETMEN=1 时适用。 该字段为写保护。仅当 MODE[WPDIS]=1 时，不能写操作。 0 该通道对中的双沿捕捉模式禁用。 1 该通道对中的双沿捕捉模式使能。
17 COMP2	通道(n)互补(n=4) 使能组合通道的互补模式，在互补模式下，通道(n+1)输出与通道(n)输出相反。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 通道(n+1)输出与通道(n)输出相同。 1 通道(n+1)输出与通道(n)输出互补。

16 COMBINE2	组合通道(n=4) 使能通道(n)和(n+1)的组合功能 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 通道(n)和(n+1)独立。 1 通道(n)和(n+1)组合。
15 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
14 FAULTEN1	故障控制使能(n=2) 使能通道(n)和(n+1)中的故障控制。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 禁用该通道对中的故障控制 1 使能该通道对中的故障控制
13 SYNCEN1	同步使能(n=2) 使能寄存器 C(n)V 和 C(n+1)V 的 PWM 同步。 0 禁用该通道(n)和(n+1)中的死区时间插入。 1 使能该通道(n)和(n+1)中的死区时间插入。
12 DTEN1	死区时间使能(n=2) 使能通道(n)和(n+1)中的死区时间插入。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 禁用该通道对中的死区时间插入 1 使能该通道对中的死区时间插入。
11 DECAP1	双沿捕捉模式捕捉(n=2) 使能根据通道(n)输入事件捕捉 ETM 计数器值和双沿捕捉位配置。 该字段仅在 ETMEN=1 且 DECAPEN=1 时适用。 若选定双沿捕捉—单次模式，且完成通道(n+1)事件捕捉，则 DECAP 位由硬件自动清零。 0 双沿捕捉未激活。 1 双沿捕捉已激活。
10 DECAPEN1	双沿捕捉模式使能(n=2) 使能通道(n)和(n+1)的双沿捕捉模式。该位根据表 6—17，重新在双沿捕捉模式下配置 MSnA、ELSnB:ELSnA 和 ELS(n+1)B:ELS(n+1)A 位的功能。该字段仅在 ETMEN=1 时适用。 该字段为写保护。仅当 MODE[WPDIS]=1 时，不能写操作。 0 该通道对中的双沿捕捉模式禁用。 1 该通道对中的双沿捕捉模式使能。
9 COMP1	通道(n)互补(n=2) 使能组合通道的互补模式，在互补模式下，通道(n+1)输出与通道(n)输出相反。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 通道(n+1)输出与通道(n)输出相同。 1 通道(n+1)输出与通道(n)输出互补。
8 COMBINE1	组合通道(n=2) 使能通道(n)和(n+1)的组合功能 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 通道(n)和(n+1)独立。 1 通道(n)和(n+1)组合。

7 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
6 FAULTEN0	故障控制使能(n=0) 使能通道(n)和(n+1)中的故障控制。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 禁用该通道对中的故障控制 1 使能该通道对中的故障控制
5 SYNCEN0	同步使能(n=0) 使能寄存器 C(n)V 和 C(n+1)V 的 PWM 同步。 0 禁用该通道(n)和(n+1)中的 PWM 同步。 1 使能该通道(n)和(n+1)中的 PWM 同步。
4 DTEN0	死区时间使能(n=4) 使能通道(n)和(n+1)中的死区时间插入。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 禁用该通道对中的死区时间插入 1 使能该通道对中的死区时间插入。
3 DECAP0	双沿捕捉模式捕捉(n=0) 使能根据通道(n)输入事件捕捉 ETM 计数器值和双沿捕捉位配置。 该字段仅在 ETMEN=1 且 DECAPEN=1 时适用。 若选定双沿捕捉—单次模式，且完成通道(n+1)事件捕捉，则 DECAP 位由硬件自动清零。 0 双沿捕捉未激活。 1 双沿捕捉已激活。
2 DECAPEN0	双沿捕捉模式使能(n=0) 使能通道(n)和(n+1)的双沿捕捉模式。该位根据表 6—17，重新在双沿捕捉模式下配置 MSnA、ELSnB:ELSnA 和 ELS(n+1)B:ELS(n+1)A 位的功能。 该字段为写保护。仅当 MODE[WPDIS]=1 时，不能写操作。 0 该通道对中的双沿捕捉模式禁用。 1 该通道对中的双沿捕捉模式使能。
1 COMP0	通道(n)互补(n=0) 使能组合通道的互补模式，在互补模式下，通道(n+1)输出与通道(n)输出相反。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 通道(n+1)输出与通道(n)输出相同。 1 通道(n+1)输出与通道(n)输出互补。
0 COMBINE0	组合通道(n=0) 使能通道(n)和(n+1)的组合功能 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 通道(n)和(n+1)独立。 1 通道(n)和(n+1)组合。

● 6.3.7.15 死区时间插入控制寄存器 (ETM2_DEADETME)

该寄存器选择死区时间预分频器系数和死区时间值。对于死区时间插入而言，所有 ETM 通道均使用此时钟预分频器和该死区时间值。

地址：基址 + 68h 偏移：ETM2_DEADETME = 4003_A068h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0																								DTPS		DTVAL						
写																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-29 ETM2_DEADETME 字段描述

位	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
7-6 DTPS	死区时间预分频器值 选择系统时钟的分频系数。死区时间计数器使用此预分频时钟。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0x 系统时钟 1 分频。 10 系统时钟 4 分频 11 系统时钟 16 分频。
5-0 DTVAL	死区时间值 选择死区时间插入值，用于死区时间计数器。死区时间计数器由系统时钟的一部分提供时钟源。 参考 DTPS 说明。 死区时间插入数值=(DTPS*DTVAL)。 DTVAL 选择插入死区时间计数数目，如下所示： 若 DTVAL 等于 0，则不插入计数数目。 若 DTVAL 等于 1，则插入 1 个计数数目。 若 DTVAL 等于 2，则插入 2 个计数数目。 以此类推，最高可达 63 个计数数目。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。

● 6.3.7.16 ETM 外部触发寄存器 (ETM2_EXTTRIG)

该寄存器：

- 指示何时产生通道触发
- 当 ETM 计数器等于其初始值时，使能触发生成
- 产生通道触发时，选择所用的通道

可选择多个通道，以便在一个 PWM 周期内产生多个触发。通道 6 和 7 未用来产生通道触发。

地址：基址+6Ch 偏移：ETM2_EXTTRIG = 4003_A06Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	保留															
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	保留								TRIGF	INITIRIGEN	CH1TRIG	CH0TRIG	CH5TRIG	CH4TRIG	CH3TRIG	CH2TRIG
写									0							
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-30 ETM2_EXTTRIG 字段描述

位	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
7 TRIGF	通道触发标志 生成通道触发时，由硬件置位。在 TRIGF 置位的情况下通过读取 EXTTRIG，然后向 TRIGF 写入 0，可清零 TRIGF。将 1 写入 TRIGF 无效。 如果在完成清零序列之前生成另一个通道触发，则序列复位，从而 TRIGF 在完成针对先前 TRIGF 的清序列之后将保持置位状态。 0 未生成通道触发。 1 生成一个通道触发。
6 INITTRIGEN	初始化触发使能 当 ETM 计数器等于 CNTIN 寄存器时，使能触发生成。 0 禁用初始化触发生成。 1 使能初始化触发生成。
5 CH1TRIG	通道 1 触发使能 当 ETM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成 1 使能通道触发生成。
4 CH0TRIG	通道 0 触发使能 当 ETM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成 1 使能通道触发生成。
3 CH5TRIG	通道 5 触发使能 当 ETM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成 1 使能通道触发生成。
2 CH4TRIG	通道 4 触发使能 当 ETM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成 1 使能通道触发生成。

1 CH3TRIG	通道 3 触发使能 当 ETM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成 1 使能通道触发生成。
0 CH2TRIG	通道 2 触发使能 当 ETM 计数器等于 CnV 寄存器时，使能通道触发生成。 0 禁用通道触发生成 1 使能通道触发生成。

● 6.3.7.17 通道极性寄存器（ETM2_POL）

该寄存器定义 ETM 通道的输出极性。注：当故障控制使能且检测到故障条件时，通道输出的安全值就是通道的无效状态。也就是说，通道安全值是其 POL 位的值。

地址：基址 + 70h 偏移：ETM2_POL= 4003_A070h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	保留															
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写	保留										POL5	POL4	POL3	POL2	POL1	POL0
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-31 ETM2_POL 字段描述

位	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
5-0 POLn	通道 n 极性 （n= 5 到 0） 定义通道输出的极性。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 通道极性为高电平有效。 1 通道极性为低电平有效。

● 6.3.7.18 故障模式状态寄存器(ETM2_FMS)

该寄存器含有故障检测标志位、写保护使能位和使能故障输入的逻辑 OR。

地址：基址 +74h 偏移：ETM2_FMS= 4003_A074h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	保留															

写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0								FAULTF	WPEN	FAULTIN	0	FAULTF3	FAULTF2	FAULTF1	FAULTF0
写									0				0	0	0	0
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-32 ETM2_FMS 字段描述

位	描述
31-8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
7 FAULTF	故障检测标志 代表单个 FAULTFj 位的逻辑 OR，其中：j=3、2、1、0。在 FAULTF 置位的情况下，当已使能的故障输入端不存在故障条件时,读取 FMS 寄存器,然后向 FAULTF 写入 0 可清零 FAULTF。将 1 写入 FAULTF 无效。 如果在清零序列结束前检测到已使能故障输入的另一个故障条件，则序列复位，因此 FAULTF 将在完成针对上一个故障条件的清零序列之后保持置位状态。当 FAULTFj 位独立清零时，FAULTF 位亦清零。 0 未检测到故障条件。 1 检测到故障条件。
6 WPEN	写保护使能 WPEN 位与 WPDIS 位相反。将 1 写入 WPEN 时，可将其置位。WPEN 清零：WPEN 位读取为 1，然后向 WPDIS 写入 1. 向 WPEN 中写入 0 无效。 0 写保护禁用。可写入写保护位。 1 写保护使能。不可写入写保护位。
5 FAULTIN	故障输入 使能故障控制时，它代表滤波（假设滤波器使能）之后已使能故障输入的逻辑 OR。 0 已使能故障输入的逻辑 OR 为 0。 1 已使能故障输入的逻辑 OR 为 1。
4 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
3-0 FAULTFn	故障检测标志 n（n=3 到 0） 故障控制使能时由硬件置位，对应的故障输入使能，且在故障输入端检测到故障条件。当对应的故障输入无故障条件时，在 FAULTFn 置位的情况下读取 FMS 寄存器，然后将 0 写入 FAULTFn 即可清零 FAULTFn。将 1 写入 FAULTFn 无效。FAULTF 位清零时，也会清零 FAULTFn 位。如果在清零序列结束前检测到对应故障输入端存在另一个故障条件，则序列复位，因此 FAULTFn 将在完成针对上一个故障条件的清零序列之后保持置位状态。 0 故障输入端未检测到故障条件。 1 故障输入端检测到故障条件。

● 6.3.7.19 输入捕捉滤波器控制寄存器 (ETM2_FILTER)

该寄存器选择用于通道输入的滤波器值。通道 4、5、6、7 没有输入滤波器。
注：写入 FILTER 寄存器的内容即刻生效，且必须在通道 0、1、2 和 3 为非输入模式时才能完成。如果无法做到，则可能错过有效信号。

地址：基址+78h 偏移：ETM2_FILTER= 4003_A078h

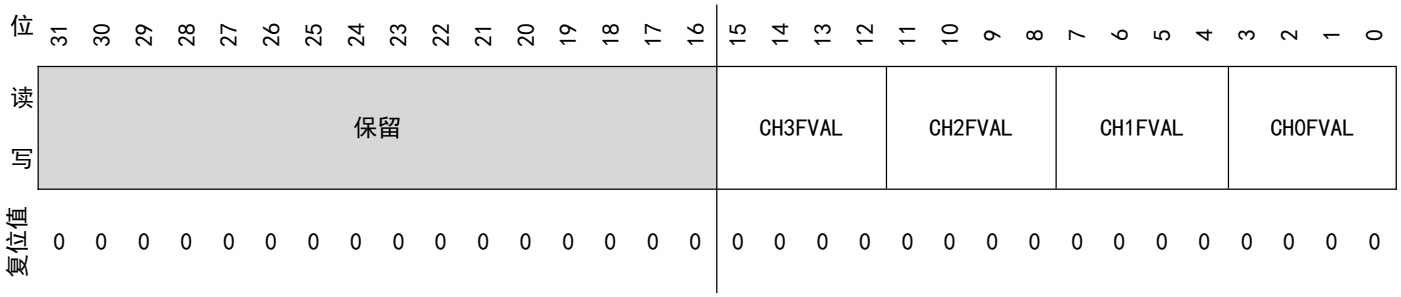


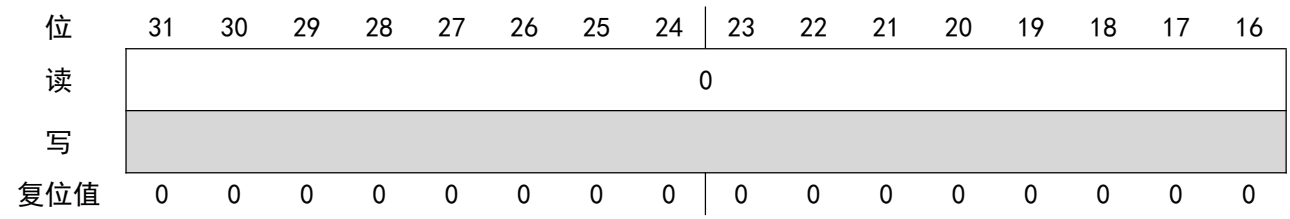
表 6-33 ETM2_FILTER 字段描述

位	描述
31-16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
15-12 CH3FVAL	通道 3 输入滤波器，选择用于通道输入的滤波器值。 0 滤波器禁用。 1 滤波器使能，设值滤波值。
11-8 CH2FVAL	通道 2 输入滤波器，选择用于通道输入的滤波器值。 0 滤波器禁用。 1 滤波器使能，设值滤波值。
7-4 CH1FVAL	通道 1 输入滤波器，选择用于通道输入的滤波器值。 0 滤波器禁用。 1 滤波器使能，设值滤波值。
3-0 CH0FVAL	通道 0 输入滤波器，选择用于通道输入的滤波器值。 0 滤波器禁用。 1 滤波器使能，设值滤波值。

● 6.3.7.20 故障控制寄存器(ETM2_FLTCTRL)

该寄存器选择故障输入的滤波器值，使能故障输入和故障输入滤波器。

地址：基址+7Ch 偏移：ETM2_FLTCTRL= 4003_A07Ch



位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0				FFVAL				FFLTR3EN	FFLTR2EN	FFLTR1EN	FFLTR0EN	FAULT3EN	FAULT2EN	FAULT1EN	FAULT0EN
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-34 ETM2_FLTCTRL 字段描述

位	描述
31-12 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
11-8 FFVAL	故障输入滤波器 选择用于故障输入的滤波器值。 数值为 0 时，故障滤波器禁用。 注：写操作即时生效，且必须在禁用故障控制或全部故障输入时完成。否则会导致故障检测失效。
7-4 FFLTRnEN	故障输入 n 滤波器使能 （n=3 到 0） 使能故障输入的滤波器。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 禁用故障输入滤波器。 1 使能故障输入滤波器。
3-0 FAULTnEN	故障输入 n 使能 （n = 3 到 0） 使能故障输入。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 禁用故障输入。 1 使能故障输入。

● 6.3.7.21 配置寄存器(ETM2_CONF)

该寄存器选择应当在 TOF 位置前发生的 ETM 计数器溢出次数、ETM 在调试模式下的特性、外部全局时基的使用以及全局时基信号生成。

地址：基址 + 84h 偏移：ETM2_CONF= 4006_A084h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0					GTBEOUT	GTBEEN	0	BDMMODE		0	NUMTOF				
写																

复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-35 ETM2_CONF 字段描述

位	描述
31-11 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
10 GTBEOUT	全局时基输出 使能到其他 ETM 的全局时基信号生成。 0 禁用全局时基信号生成。 1 使能全局时基信号生成。
9 GTBEEN	全局时基使能 配置 ETM，以使用另一个 ETM 产生的外部全局时基信号。 0 禁用外部全局时基。 1 使能外部全局时基。
8 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
7-6 BDMMODE	调试模式 选择调试模式下的 ETM 特性。参见调试模式。
5 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
4-0 NUMTOF	TOF 频率 选择计数器溢出数量与 TOF 位置次数之比。 NUMTOF=0：TOF 位针对每一位计数器溢出进行置位。 NUMTOF=1：TOF 位针对每一位计数器溢出进行置位，但不会对下一个溢出置位。 NUMTOF=2：TOF 位针对每一位计数器溢出进行置位，但不对之后的两个溢出置位。 NUMTOF=3：TOF 位针对每一位计数器溢出进行置位，但不对之后的三个溢出置位。 以此类推，此模式最多可延续到 31 次。

● 6.3.7.22 ETM 故障输入极性寄存器(ETM2_FLTPOL)

该寄存器定义故障输入极性。

地址：基址+88h 偏移：ETM2_FLTPOL = 4006_A088h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0												FLT3POL	FLT2POL	FLT1POL	FLT0POL
写																

复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-36 ETM2_FLTPOL 字段描述

位	描述
31-4 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
3-0 FLTnPOL	故障输入 n 极性 （n= 3 到 0） 定义故障输入的极性。 该字段为写保护。仅当 MODE[WPDIS]=1 时，才能写操作。 0 故障输入极性为高电平有效。故障输入端的 1 表示故障。 1 故障输入极性为低电平有效。故障输入端的 0 表示故障。

● 6.3.7.23 同步配置寄存器(ETM2_SYNCONF)

当检测到硬件触发 j 时，若 ETM 清零 TRIGj 位(j=0,1,2)，该寄存器 PWM 同步配置、SWOCTRL、INVCTRL 和 CNTIN 寄存器进行同步。

地址：基址 + 8Ch 偏移：ETM2_SYNCONF = 4006_A08Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0											HWSOC	HWINVC	HWOM	HWWFBUF	HWRSTCNT
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0			SWSOC	SWMSOC	SWOM	SWWRBUF	SWRSTCNT	SYNCODE	0	SWOC	INVC	0	CNTNC	0	HWTRIGMODE
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-37 ETM2_SYNCONF 字段描述

位	描述
31-21 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
20 HWSOC	软件输出控制同步通过硬件触发激活。 0 硬件触发不激活 SWOCTRL 寄存器同步。 1 硬件触发激活 SWOCTRL 寄存器同步。
19 HWINVC	反相控制同步通过硬件触发激活。 0 硬件触发不激活 INVCTRL 寄存器同步。 1 硬件触发激活 INVCTRL 寄存器同步。

18 HWOM	输出屏蔽同步通过硬件触发激活。 0 硬件触发不激活 OUTMASK 寄存器同步。 1 硬件触发激活 OUTMASK 寄存器同步。
17 HWWRBUF	MOD、CNTIN 和 CV 寄存器同步通过硬件触发激活。 0 硬件触发不激活 MOD、CNTIN 和 CV 寄存器同步。 1 硬件触发激活 MOD、CNTIN 和 CV 寄存器同步。
16 HWRSTCNT	ETM 计数器同步通过硬件触发激活 0 硬件触发不激活 ETM 计数器同步。 1 硬件触发激活 ETM 计数器同步。
15-13 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
12 SWSOC	软件输出控制同步通过软件触发激活。 0 软件触发不激活 SWOCTRL 寄存器同步。 1 软件触发激活 SWOCTRL 寄存器同步。
11 SWINVC	反相控制同步通过软件触发激活。 0 软件触发不激活 INVCTRL 寄存器同步。 1 软件触发激活 INVCTRL 寄存器同步。
10 SWOM	输出屏蔽同步通过软件触发激活 0 软件触发不激活 OUTMASK 寄存器同步。 1 软件触发激活 OUTMASK 寄存器同步。
9 SWWRBUF	MOD、CNTIN 和 CV 寄存器同步通过软件触发激活。 0 软件触发不激活 MOD、CNTIN 和 CV 寄存器同步。 1 软件触发激活 MOD、CNTIN 和 CV 寄存器同步。
8 SWRSTCNT	ETM 计数器同步通过软件触发激活。 0 软件触发不激活 ETM 计数器同步。 1 软件触发激活 ETM 计数器同步。
7 SYNCMODE	同步模式 选择 PWM 同步模式 0 选择传统 PWM 同步。 1 选择增强 PWM 同步。
6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
5 SWOC	SWOCTRL 寄存器同步 0 SWOCTRL 寄存器将在所有系统时钟的上升沿以其缓冲区的数值进行更新。 1 SWOCTRL 寄存器通过 PWM 同步以其缓冲区值进行更新。
4 INVC	INVCTRL 寄存器同步 0 INVCTRL 寄存器将在所有系统时钟的上升沿以其缓冲区的数值进行更新。 1 INVCTRL 寄存器通过 PWM 同步以其缓冲区值进行更新。
3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.

2 CNTINC	CNTIN 寄存器同步 0 CNTIN 寄存器将在所有系统时钟的上升沿以其缓冲区的数值进行更新。 1 CNTIN 寄存器通过 PWM 同步以其缓冲区值进行更新。
1 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
0 HWTRIGMODE	硬件触发模式 0 检测到硬件触发 j 时(j=0、1、2)。ETM 清零 TRIGj 位。 1 检测到硬件触发 j 时(j=0、1、2)。ETM 不清零 TRIGj 位。

● 6.3.7.24 ETM 反向控制寄存器(ETM2_INVCTRL)

该寄存器控制通道(n)输出何时变为通道(n+1)输出，以及通道(n+1)输出何时变为通道(n)输出。每个 INVmEN 位通过 INVCTRL 寄存器同步进行更新。

地址：基址+ 90h 偏移：ETM2_INVCTRL = 4006_A090h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0												INV3EN	INV2EN	INV1EN	INV0EN
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-38 ETM2_FLTPOL 字段描述

位	描述
31-4 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
3-0 INVnEN	通道 n 反相使能 (n=3 到 0) 0 禁用反相 1 使能反相

● 6.3.7.25 ETM 软件输出控制寄存器(ETM2_SWOCTRL)

该寄存器使能通道(n)输出的软件控制，并定义强制进入通道(n)输出的数值：
CHnOC 位通过软件使能对应通道(n)输出的控制。
CHnOC 位选择强制进入对应通道(n)输出的数值。
该寄存器具有写缓冲区。该字段通过 SWOCTRL 寄存器同步进行更新。

地址：基址+94h 偏移：ETM2_SWOCTRL = 4006_A094h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读			CH5OCV	CH4OCV	CH3OCV	CH2OCV	CH1OCV	CH0OCV			CH5OC	CH4OC	CH3OC	CH2OC	CH1OC	CH0OC
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-39 ETM2_FLTPOL 字段描述

位	描述
31-14 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
13-8 CHnOCV	通道 n 软件输出控制值 （n= 5 到 0） 0 软件输出控制强制 0 进入通道输出。 1 软件输出控制强制 1 进入通道输出。
7-6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
5-0 CHnOC	通道 n 软件输出控制使能 （n= 5 到 0） 0 通道输出不受软件输出控制的影响。 1 通道输出受软件输出控制的影响。

● 6.3.7.26 ETM PWM 装载寄存器(ETM2_PWMLOAD)

当 ETM 计数器从 MOD 寄存器值变更至其下一个值，或者发生通道（j）匹配时，使能 MOD、CNTIN、$C_{(n)}V$ 和 $C_{(n+1)}V$ 寄存器的加载，数值为他们写入缓冲区数值。当 ETM 计数器=$C_{(j)}V$ 时，通道（j）发生匹配。

地址：基址+ 98 偏移：ETM2_PWMLOAD = 4006_A098h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0						LDOK	0			CH5SEL	CH4SEL	CH3SEL	CH2SEL	CH1SEL	CH0SEL
写																

复位值	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-40 ETM2_PWMLOAD 字段描述

位	描述
31-10 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
9 LDOK	加载使能 使能 MOD、CNTIN 和 CV 寄存器的加载，数值为它们的写缓冲区内容。 0 禁用更新值加载。 1 使能更新值加载。
8-6 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0.
5-0 CHnSEL	通道 n 选择 （n=5 到 0） 0 匹配过程中不包括通道。 1 匹配过程中包括通道。

■ 6.3.8 功能说明

下图展示了本文档中用来表示计数器和信号生成的各种用法。
ETM 计数为向上计数，通道（n)处于先高后低 EPWM 模式。
PS[2:0]=001， CNTIN=0x0000， MOD=0x0004 ，Cnv=0x0002

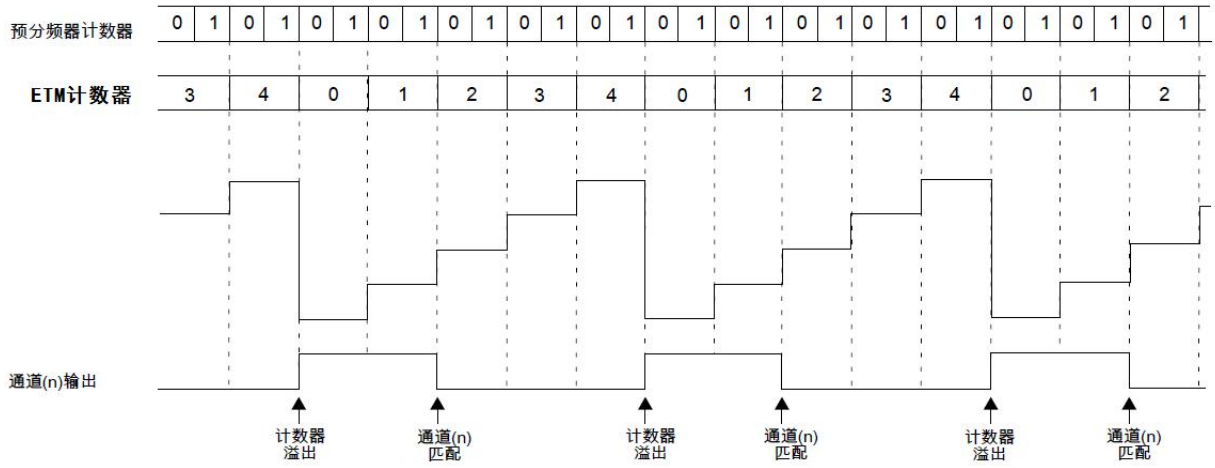


图 6-4 所用的用法

● 6.3.8.1 时钟源

ETM 模块只有 1 个时钟域：系统时钟。

❖ 6.3.8.1.1 计数器时钟源

SC 寄存器中的 CLKS[1:0]位可以从三种可能的时钟源中为 ETM 计数器选择一种，或者禁用 ETM 计数器。发生任意 MCU 复位后，CLKS[1:0] = 0:0，这意味着未选择任何时钟源。

可在任意时间对 CLKS[1:0]位进行读取或写入操作。通过向 CLKS[1:0]位写入 0:0 禁用 ETM 计数器不会影响 ETM 计数器值或其他寄存器。固定频率时钟 ICSFFCLK 是一种可以用于 ETM 计数器的备选时钟源，允许在系统时钟或外部时钟以外另选一种时钟源。这种时钟输入源集成在芯片内部。有关更多信息，请参见具体的芯片文档。由于 ETM 硬件的实施限制，固定频率时钟的频率不得超过系统时钟频率的 1/2。

外部时钟会经过一个由系统时钟计时的同步器，以确保计数器转换与系统时钟转换恰当的同步。因此，为符合奈奎斯特准则，并考虑到时钟抖动，外部时钟源的频率不得超过系统时钟频率的 1/4。

● 6.3.8.2 预分频器

所选的计数器时钟源会经过一个预分频器，该预分频器是一个 7 位计数器。由 PS[2:0]位选择预分频器的值。下图给出了预分频器计数器和 ETM 计数器的示例。

ETM 计数为向上计数。

PS[2:0]=001， CNTIN=0x0000 ，MOD=0x0003

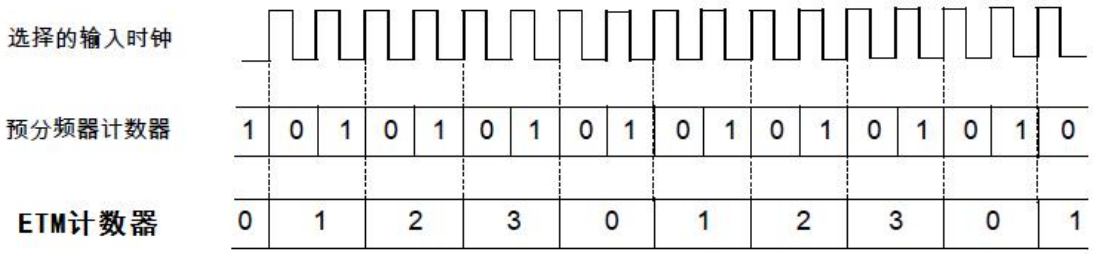


图 6-5 预分频器计数器示例

● 6.3.8.3 计数器

ETM 有一个 16 位计数器，通道可以选择用于输入输出模式。ETM 计数器时钟是由选定的输入时钟经过预分频得到的。

ETM 计时器有这些工作模式：

- 向上计数
- 向上-向下计数

❖ 6.3.8.3.1 向上计数

在以下情形中会选择向上计数：

- CPWMS = 0

CNTIN 定义计数的起始值，MOD 定义计数的最终值，参见下图：CNTIN 的值加载到 ETM 计数器中，计数器的值递增，直至达到 MOD 的值，此时计数器将重新加载 CNTIN 的值。

采用向上计数时的 ETM 周期为(MOD – CNTIN + 0x0001) × ETM 计数器时钟的周期。

ETM 计数器从 MOD 变为 CNTIN 时，TOF 位将置位。

ETM 计数为向上计数。

CNTIN=0xFFFC(在二的补码中等于-4) MOD=0x0004

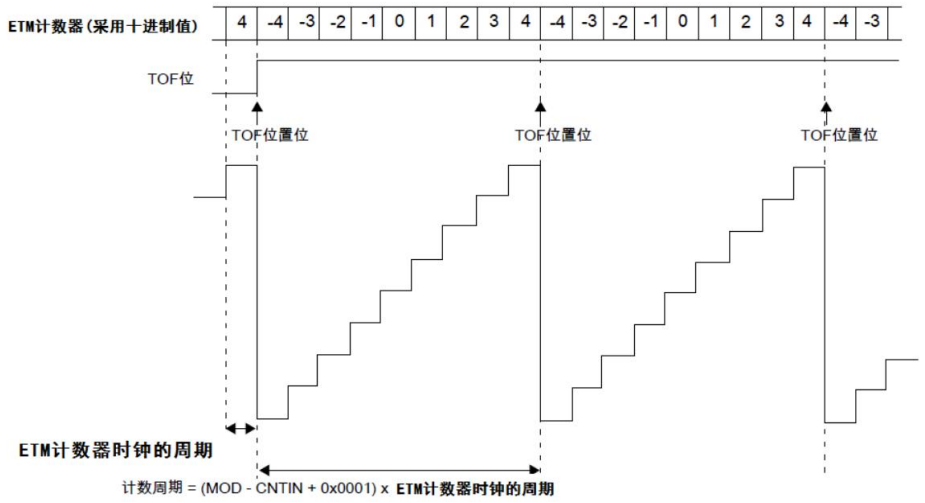


图 6-6 ETM 向上有符号计数示例

表 6-41 基于 CNTIN 值的 ETM 计数

满足条件	结果
CNTIN=0x0000	ETM 计数为向上无符号计数。参见下图：
CNTIN[15]=1	ETM 计数器的初始值是一个二补码负数，因此 ETM 计数为向上有符号计数。
CNTIN[15]=0 且 CNTIN ≠ 0x0000	ETM 计数器的初始值是一个正数，因此 ETM 计数为向上无符号计数。

ETM 计数为向上计数。

CNTIN=0x0000 MOD=0x0004

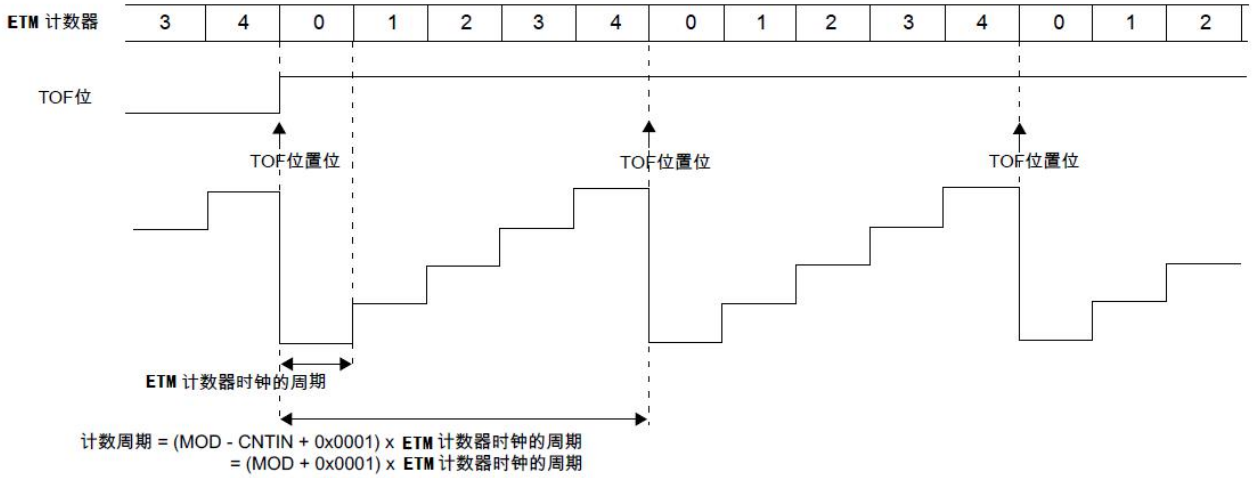


图 6-7 CNTIN=0x0000 时的 ETM 向上计数示例

注：

- 无论是无符号计数还是有符号计数，ETM 工作只有在 CNTIN 寄存器的值小于 MOD 寄存器的值时才有效。软件负责确保 CNTIN 和 MOD 寄存器中的值满足此要求。CNTIN 和 MOD 如有任何不满足此要求的值，将导致意外现象发生。
- MOD = CNTIN 是一个冗余条件。这种情况下，ETM 计数器始终等于 MOD，将在 ETM 计数器时钟的每一个上升沿置位 TOF 位。
- MOD = 0x0000、CNTIN = 0x0000（例如，复位后）且 ETMEN = 1 时，ETM 计数器将在 0x0000

保持停止，直到向 MOD 或 CNTIN 寄存器中写入非零值。

- 不建议将 CNTIN 设置为大于 MOD 的值，因为这种非寻常设置可能导致 ETM 的操作难以理解。然而，对于这种配置没有任何限制，下图给出了示例。

ETM 计数为向上计数。

CNTIN=0x0015　MOD=0x0005

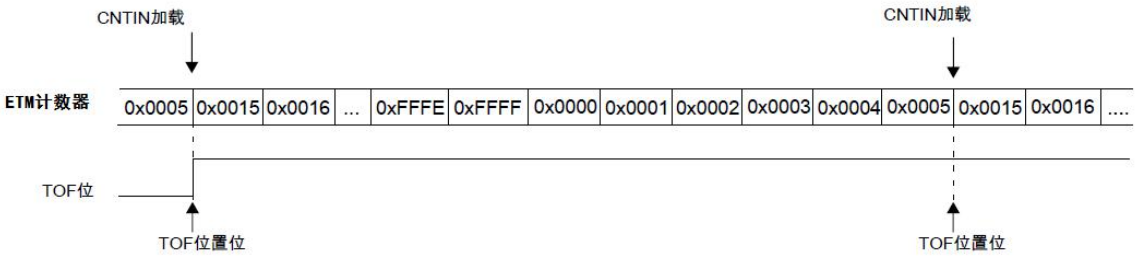


图 6-8 CNTIN 值大于 MOD 值时的 ETM 向上计数示例

❖ 6.3.8.3.2 向上-向下计数

向上-向下计数情况：

• CPWMS = 1

CNTIN 定义计数的起始值，MOD 定义计数的最终值。CNTIN 的值加载到 ETM 计数器中，计数器的值一直增加，直至达到 MOD 的值，紧接着计数器的值一直减少，直至回到 CNTIN 的值，然后计数器将重新开始自上而下计数。

采用向上-向下计数时的 ETM 周期为 2 × (MOD - CNTIN)×ETM 计数器时钟的周期。

ETM 计数器从 MOD 更改为(MOD - 1)时，TOF 位将被置位。如果(CNTIN = 0x0000)，ETM 计数为无符号向上-向下计数。参见下图：

ETM 计数为向上-向下计数。

CNTIN=0x0000, MOD=0x0004

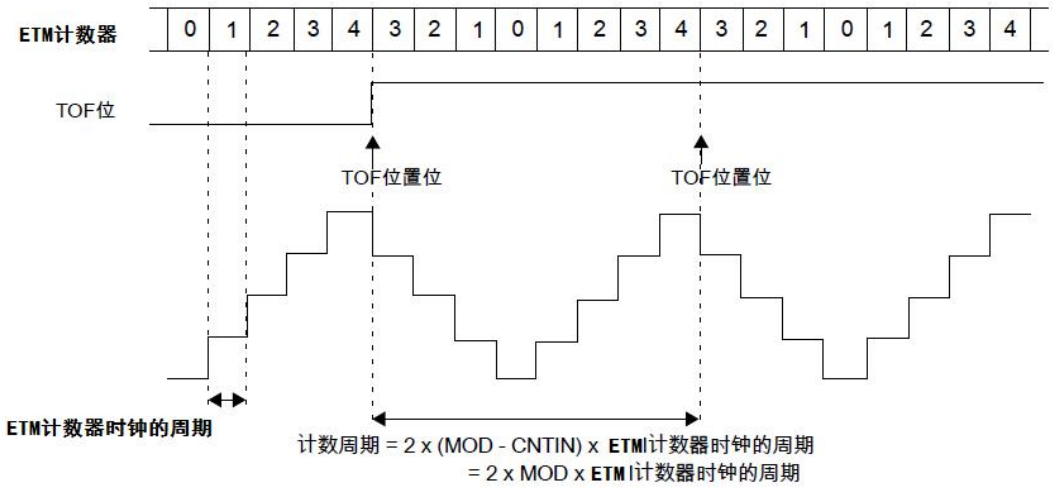


图 6-9 CNTIN=0x0000 时的 ETM 向上-向下计数示例

注:推荐在 CNTIN = 0x0000 的条件下使用向上-向下计数

❖ 6.3.8.3.3 自由运行计数器

如果(ETMEN =0)且（MOD=0x0000 或 MOD = 0xFFFF），则 ETM 计数器为自由运行计数器。这种情况下，ETM 计数器从 0x0000 到 0xFFFF 自由运行，并且当 ETM 计数器从 0xFFFF 更改为 0x0000 时，TOF 位将置位。参见下图：

ETMEN=0, MOD=0x0000

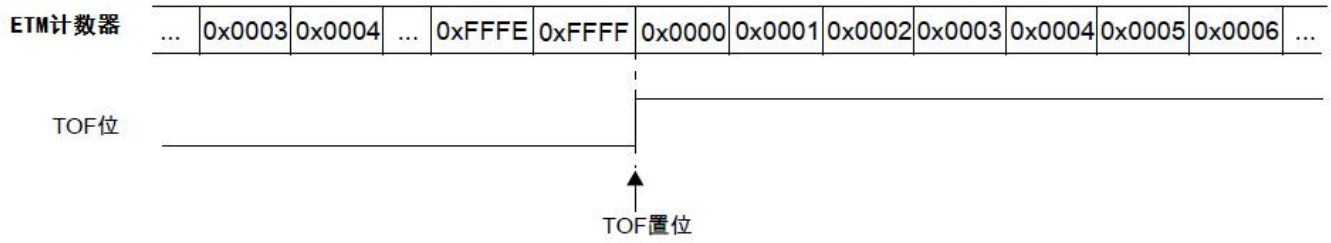


图 6-10 ETM 计数器自由运行时的示例

ETM 计数器在以下情形中也是自由运行计数器 ：

- ETMEN = 1
- CPWMS = 0
- CNTIN = 0x0000，且
- MOD = 0xFFFF

❖ 6.3.8.3.4 计数器复位

以下任一情形都会导致 ETM 计数器复位为 CNTIN 寄存器中的值，并且通道输出重置为其初始值，但处于输出比较模式的通道例外。

- 对 CNT 采取的任何写入操作。
- ETM 计数器同步 。

❖ 6.3.8.3.5 TOF 置位时

NUMTOF[4:0]位定义了 TOF 置位之前 ETM 计数器溢出应该发生的次数。如果 NUMTOF[4:0] = 0x00，则在每次 ETM 计数器溢出时置位 TOF 位。

对 NUMTOF[4:0]位采取写入操作之后，再通过对 ETM 计数器的 CNT 进行写入操作来初始化 ETM 计数器，这样可避免混淆何时发生第一次计数器溢出。

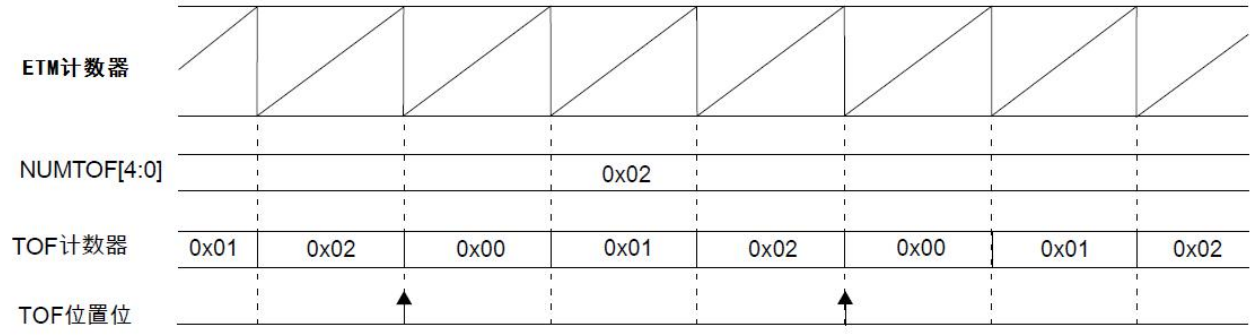


图 6-11 NUMTOF=0x02 条件下的周期性 TOF

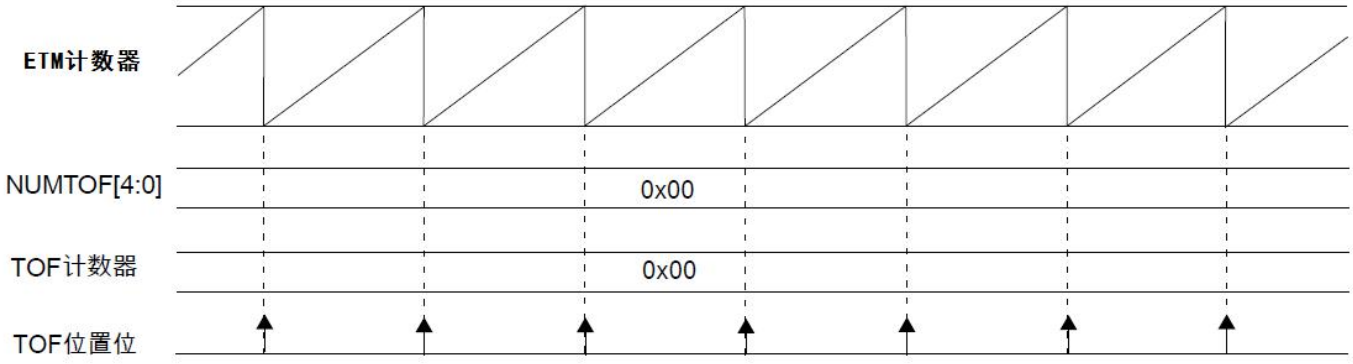


图 6-12 NUMTOF=0x00 条件下的周期性 TOF

● 6.3.8.4 输入捕捉模式

在以下情形中会选择输入捕捉模式：

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0
- MSnB:MSnA = 0:0，且
- ELSnB:ELSnA ≠ 0:0

通道输入中发生所选边沿时，ETM 计数器的当前值将会捕捉到 CnV 寄存器中，同时还会将 CHnF 位置位并生成通道中断（如果已通过 CHnIE =1 使能）。参见下图：

如果某个通道已配置为输入捕捉模式，则 ETMxCHn 引脚为边沿触发的输入。ELSnB:ELSnA 控制位可确定是哪个边沿（上升沿或下降沿）触发输入捕捉事件。注意，可以被正确检测到输入信号的最大频率为系统时钟的四分之一，这是满足信号采样的奈奎斯特准则所必需的条件。

在输入捕捉模式下，将忽略对 CnV 寄存器的写入操作。

在调试模式下，输入捕捉功能仍能按配置正常工作。发生所选边沿事件时，因调试而冻结的 ETM 计数器值被捕捉到 CnV 寄存器中且 CHnF 位被置位。

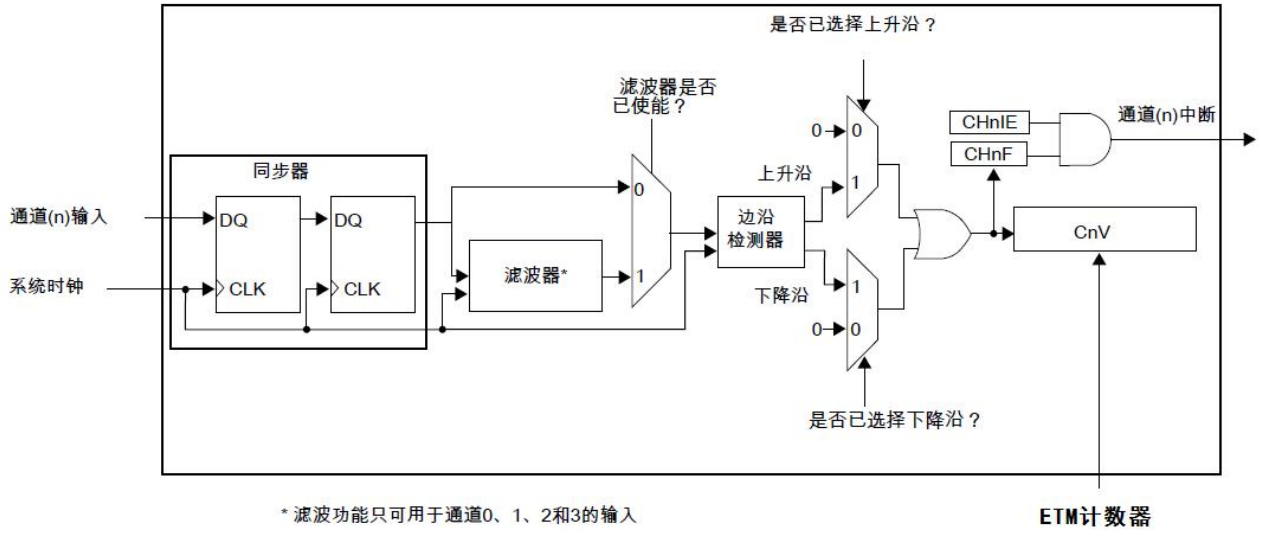


图 6-13 输入捕捉模式

如果通道输入中没有使能滤波器，则输入信号总是会被延迟 3 个系统时钟的上升沿，也就是说两个上

升沿到同步器，再加一个上升沿到边沿检测器。换言之，将在通道输入中发生有效边沿后，CHnF 位在系统时钟的第三个上升沿上置位。

注：只能在 CNTIN = 0x0000 的情况下使用输入捕捉模式。

❖ 6.3.8.4.1 输入捕捉模式的滤波器

滤波器功能只可用于通道 0、1、2 和 3。

首先，由系统时钟同步输入信号。同步之后，输入信号进入滤波器区块。参见下图：

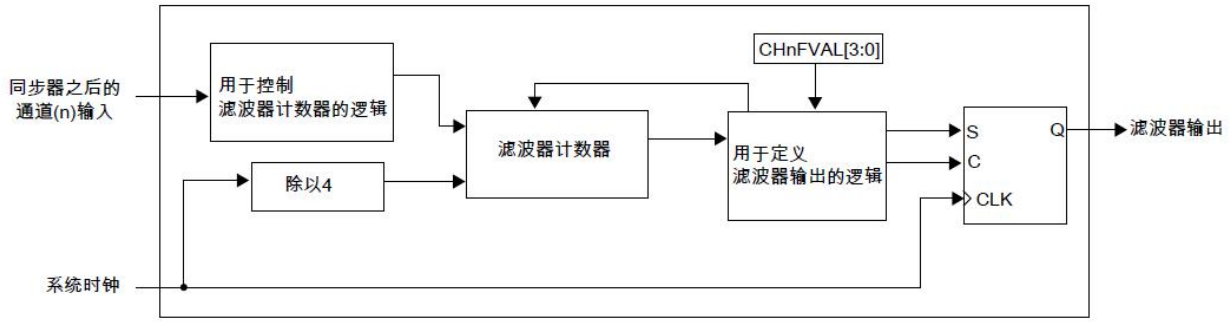


图 6-14 通道输入滤波器

当输入信号中发生状态更改时，计数器将复位并开始向上计数。只要新状态在输入端保持稳定，计数器就会继续增加数值。当计数器等于 CHnFVAL[3:0]时，输入信号的状态更改得以验证。然后就会作为脉冲边沿传输到边沿检测器。

如果在得以验证之前相对边沿出现在输入信号上，计数器就会复位。下一次输入转换时，计数器再次开始计数。任何比 CHnFVAL[3:0]（x 4 个系统时钟）所选的最小值短的脉冲都会被认作是毛刺，不会传递到边沿检测器上。下图为输入滤波器的时序图。

CHnFVAL[3:0]位为零时，将禁用滤波器功能。这种情况下，输入信号将按系统时钟的 3 个上升沿延迟。如果(CHnFVAL[3:0] ≠ 0000)，则输入信号按最小脉宽（CHnFVAL[3:0] × 4 个系统时钟）延迟，外加系统时钟的 4 个上升沿：两个上升沿到同步器，一个上升沿到滤波器输出，另外一个到边沿检测器。换言之，有效边沿出现在通道输入端之后，CHnF 将在(4+4xCHnFVAL[3:0])个系统时钟周期后被置位。通道输入滤波器中计数器的时钟是系统时钟的 4 分频。

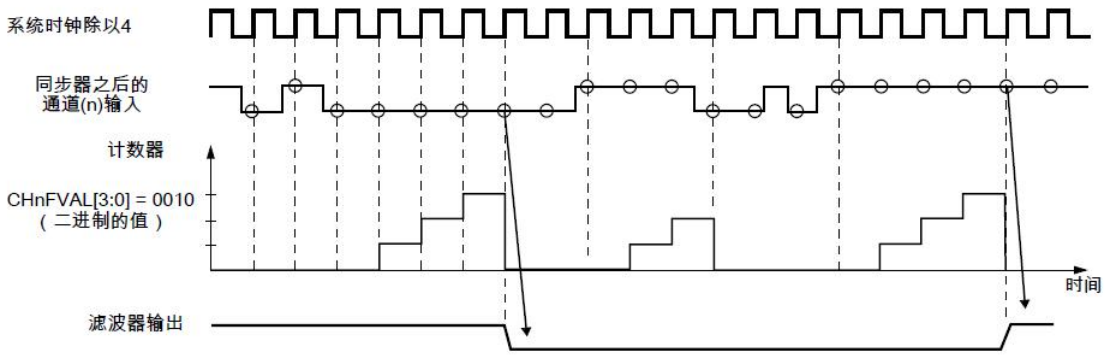


图 6-15 通道输入滤波器示例

● 6.3.8.5 输出比较模式

在以下情形中会选择输出比较模式：

- DECAPEN = 0

- COMBINE = 0
- CPWMS = 0，且
- MSnB:MSnA = 0:1

在输出比较模式下，ETM 可生成具有可编程位置、极性、持续时间和频率的定时脉冲。当计数器与某个输出比较通道的 CnV 寄存器中的值匹配时，可以设置、清除或翻转通道(n)输出。

当某个通道初次配置为翻转模式时，将保持通道输出先前的值，直到发生第一个输出比较事件。

通道(n)匹配（ETM 计数器 = CnV）时，CHnF 位将置位并生成通道(n)中断（条件是 CHnIE = 1）。

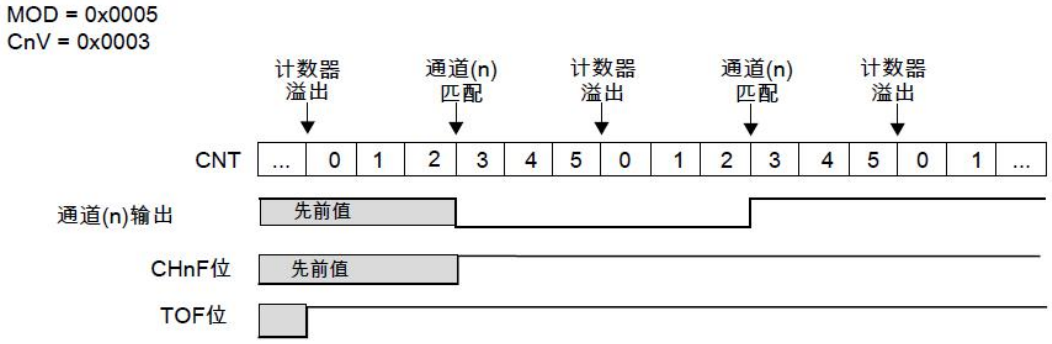


图 6-16 匹配翻转通道输出时的输出比较模式示例

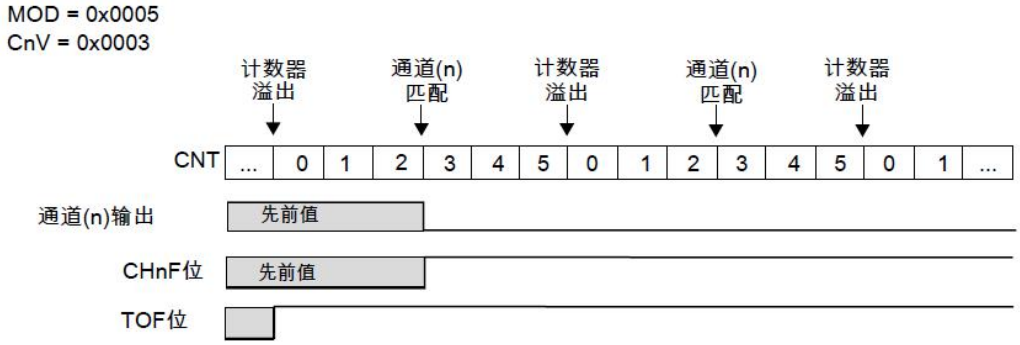


图 6-17 匹配清空通道输出时的输出比较模式示例

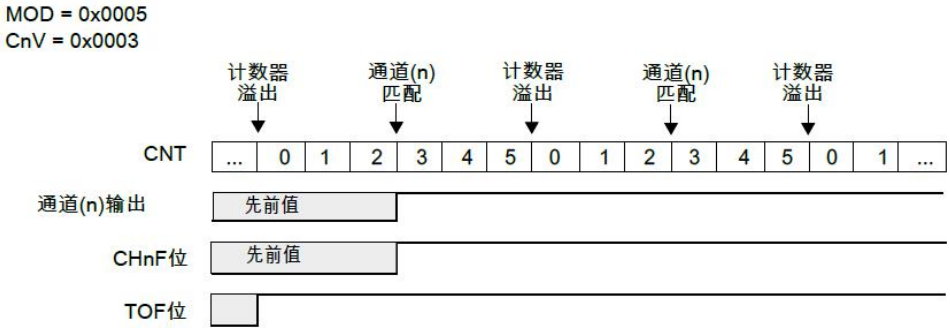


图 6-18 匹配置位通道输出时的输出比较模式示例

如果计数器达到 CnV 寄存器中的值时满足(ELSnB:ELSnA = 0:0)条件，则 CHnF 位置位并生成通道(n)中断（如果 CHnIE = 1），但是，通道(n)输出不由 ETM 修改和控制。

注：只能在 CNTIN = 0x0000 的情况下使用输出比较模式。

● 6.3.8.6 边沿对齐 PWM(EPWM)模式

在以下情形中会选择边沿对齐模式：

- DECAPEN = 0
- COMBINE = 0
- CPWMS = 0，以及
- MSnB = 1

EPWM 周期取决于(MOD - CNTIN + 0x0001)，脉宽（占空比）取决于(CnV-CNTIN)。

通道(n)匹配（ETM 计数器 =CnV）（即脉宽结束）时，CHnF 位将置位并生成通道(n)中断（条件是 CHnIE = 1）。

这种类型的 PWM 信号称为边沿对齐，因为所有 PWM 信号的前沿都与周期的开始对齐，这对 ETM 内的所有通道都一样。

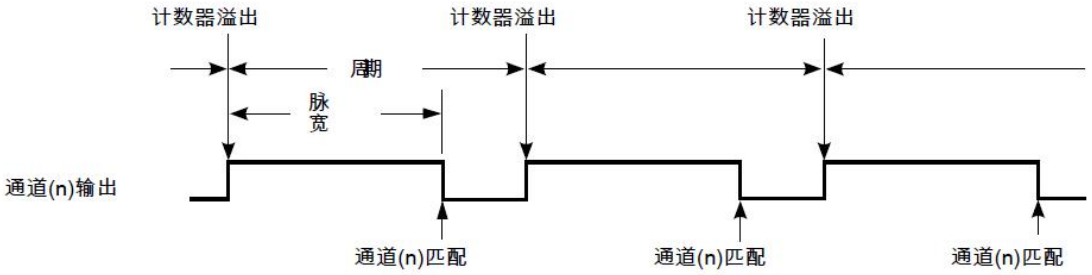


图 6-19 ELSnB:ELSnA=1:0 条件下的 EPWM 周期和脉宽

如果计数器达到 CnV 寄存器中的值时满足(ELSnB:ELSnA = 0:0)条件，则 CHnF 位置位并生成通道(n)中断（如果 CHnIE = 1），但是，通道(n)输出不由 ETM 控制。如果(ELSnB:ELSnA = 1:0)，那么，通道(n)输出会在 CNTIN 寄存器值加载到 ETM 计数器时强制为高电平，在通道(n)匹配（ETM 计数器 = CnV）的情况下强制为低电平。参见下图:

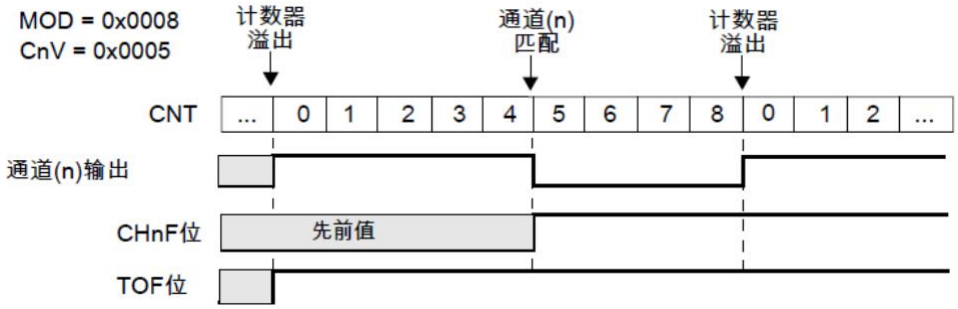


图 6-20 ELSnB:ELSnA=1:0 条件下的 EPWM 信号

如果(ELSnB:ELSnA = X:1)，那么，通道(n)输出会在 CNTIN 寄存器值加载到 ETM 计数器时强制为低电平，在通道(n)匹配（ETM 计数器 = CnV）的情况下强制为高电平。参见下图:

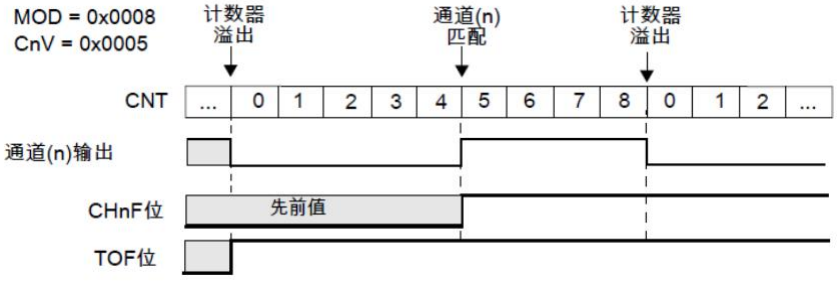


图 6-21 ELSnB:ELSnA=X:1 条件下的 EPWM 信号

如果(CnV = 0x0000)，则通道(n)输出为 0%占空比 EPWM 信号，而且 CHnF 位不置位，即便存在通道(n)

匹配也依然如此。如果(CnV > MOD)，则通道(n)输出为 100%占空比 EPWM 信号，而且 CHnF 位不置位，即便存在通道(n)匹配也依然如此。因此，要获得 100%占空比 EPWM 信号，MOD 必须小于 0xFFFF。

注：只能在 CNTIN = 0x0000 的情况下使用 EPWM 模式。

● 6.3.8.7 中心对齐 PWM(CPWM）模式

在以下情形中会选择中心对齐模式：

- DECAPEN = 0
- COMBINE = 0，且
- CPWMS = 1

CPWM 脉宽（占空比）取决于 2×(CnV - CNTIN)，周期取决于 2×(MOD - CNTIN)。参见下图：MOD 必须保持在 0x0001 到 0x7FFF 范围内，因为此范围以外的值会产生不明确的结果。在 CPWM 模式下，ETM 计数器在达到 MOD 之前会一直向上计数，然后一直向下计数，直至达到 CNTIN。

当 ETM 向下计数（脉宽开始）和 ETM 向上计数（脉宽结束）时，会在通道(n)匹配（ETM 计数器 =CnV）情况下置位 CHnF 位并生成通道(n)中断。

这种类型的 PWM 信号称为中心对齐，因为所有通道的脉宽中心都与 CNTIN 的值对齐。其他通道模式与自上而下的计数器不兼容(CPWMS = 1)。因此，当(CPWMS = 1)时，必须在 CPWM 模式下使用所有 ETM 通道。

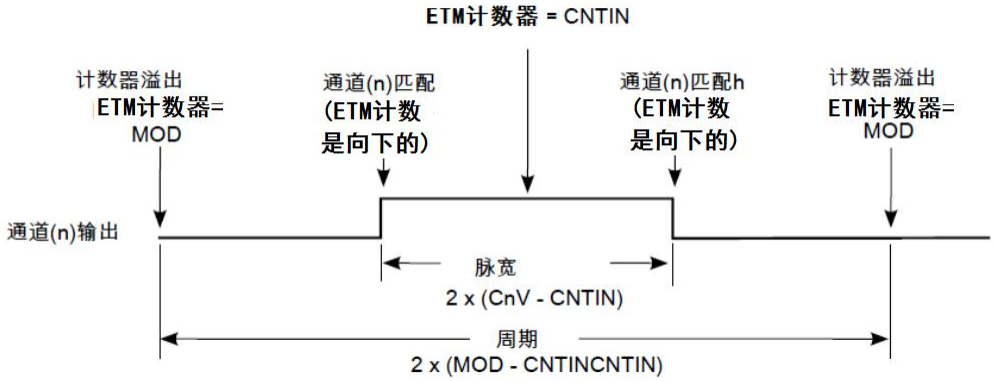


图 6-22 ELSnB:ELSnA=1:0 条件下的 CPWM 周期和脉宽

如果 ETM 计数器达到 CnV 寄存器中的值时满足(ELSnB:ELSnA = 0:0)条件,则 CHnF 位置位并生成通道(n)中断（如果 CHnIE = 1），但是，通道(n)输出不由 ETM 控制。

如果(ELSnB:ELSnA = 1:0)，那么，通道(n)输出会在向下计数时与通道(n)匹配（ ETM 计数器 = CnV）的情况下强制为高电平，在向上计数时与通道(n)匹配的情况下强制为低电平。参见下图：

图 6-23 ELSnB:ELSnA=1:0 条件下的 CPWM 信号

如果(ELSnB:ELSnA = X:1)，那么，通道(n)输出会在向下计数时与通道(n)匹配（ ETM 计数器 = CnV）的情况下强制为低电平，在向上计数时与通道(n)匹配的情况下强制为高电平。参见下图：

图 6-24 ELSnB:ELSnA=X:1 条件下的 CPWM 信号

如果(CnV = 0x0000)或 CnV 为负值，即(CnV[15] = 1)，则通道(n)输出为 0%占空因数 CPWM 信号，而且 CHnF 位不置位，即便存在通道(n)匹配也依然如此。

如果 CnV 为正值，即(CnV[15] = 0)、 (CnV ≥ MOD)且(MOD ≠ 0x0000)，则通道(n)输出为 100%占空因数的 CPWM 信号，而且 CHnF 位不置位，即便存在通道(n)匹配也依然如此。这就意味着，MOD 设置的周期可用范围是 0x0001 到 0x7FFE，如果不需要生成 100%占空因数 CPWM 信号，则为 0x7FFF。这并非一项重大限制，因为生成的周期比正常应用所需的周期要长得多。

ETM 计数器为自由运行计数器时，不得使用 CPWM 模式。

注 只能在 CNTIN = 0x0000 的情况下使用 CPWM 模式。

● 6.3.8.8 组合模式

在以下情形中会选择组合模式：

- ETMEN = 1
- DECAPEN = 0
- COMBINE = 1，且
- CPWMS = 0

在组合模式下，一个偶数通道(n)和相邻的奇数通道(n+1)组合起来生成一个 PWM 信号供通道(n)输出。

在组合模式下，PWM 周期取决于(MOD-CNTIN + 0x0001)，PWM 脉宽（占空比）取决于(|C(n+1)V-C(n)V|)。

通道(n)匹配（ETM 计数器 =C(n)V）时，CHnF 位将置位并生成通道(n)中断（条件是 CHnIE =1）。通道(n+1)匹配（ETM 计数器 = C(n+1)V）时，CH(n+1)F 位将置位并生成通道(n+1)中断（条件是 CH(n+1)IE = 1)。

如果(ELSnB:ELSnA=1:0)，则在周期开始（ETM 计数器= CNTIN）、通道(n+1)匹配（ETM 计数器= C(n+1)V）时强制通道(n)输出为低电平。在通道(n)匹配（ETM 计数器=C(n)V 时强制为高电平。参见下图：

如果(ELSnB:ELSnA = X:1)，则在周期开始（ETM 计数器=CNTIN）、通道(n+1)匹配（ETM 计数器= C(n+1)V）时强制通道(n)输出为高电平。在通道(n)匹配（ETM 计数器=C(n)V）时强制为低电平。参见下图：

在组合模式下生成通道(n)和(n+1)输出时不使用 ELS(n+1)B 和 ELS(n+1)A 位。但是，如果(ELSnB:ELSnA = 0:0)，则通道(n)输出不由 ETM 控制；如果(ELS(n+1)B:ELS(n+1)A=0:0)，则通道(n+1)输出不由 ETM 控制。

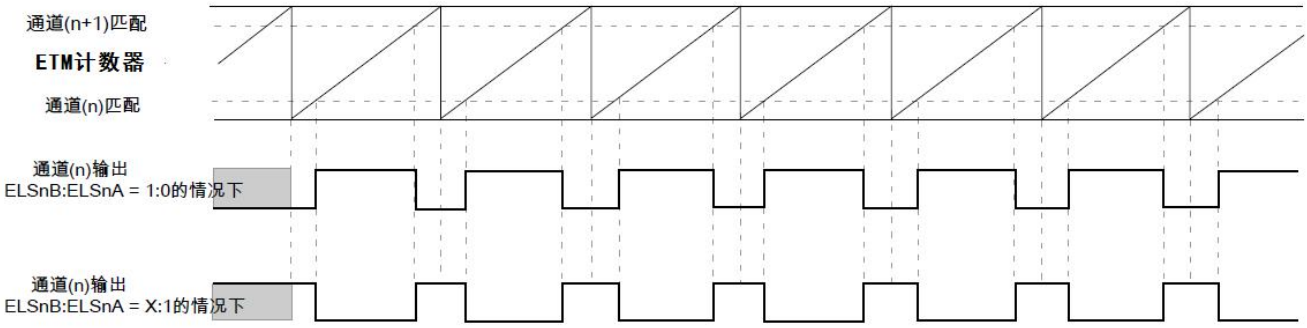


图 6-25 组合模式

下图展示了利用组合模式生成 PWM 信号的过程。

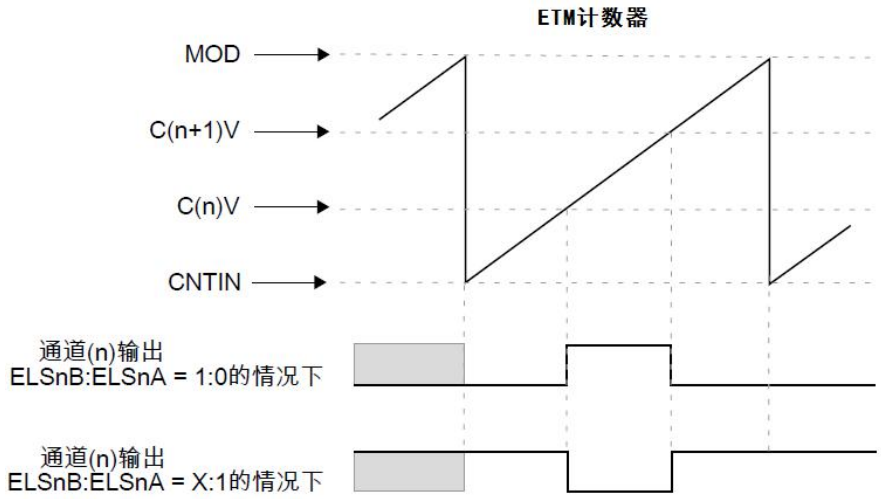


图 6-26 （CNTIN<C(n)V < MOD)、(CNTIN<C(n+1)V < MOD)且(C(n)V < C(n+1)V)条件下的通道(n)输出

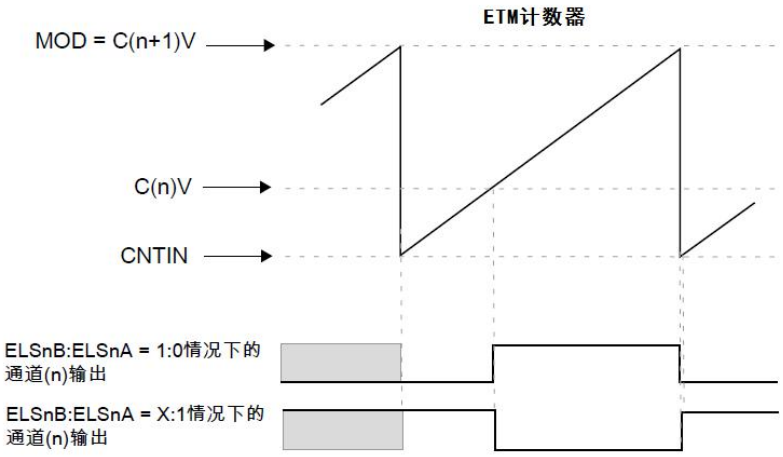


图 6-27 （CNTIN<C(n)V < MOD)且(C(n+1)V = MOD)条件下的通道(n)输出

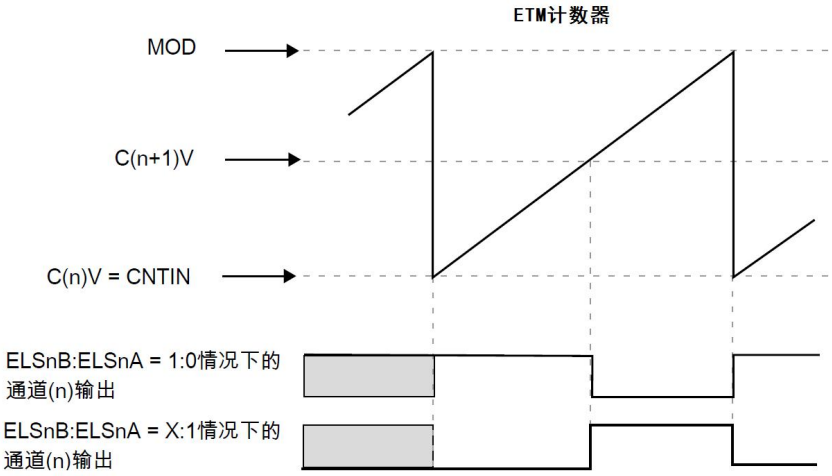


图 6-28 （C(n)V = CNTIN)且(CNTIN < C(n+1)V < MOD)条件下的通道(n)输出

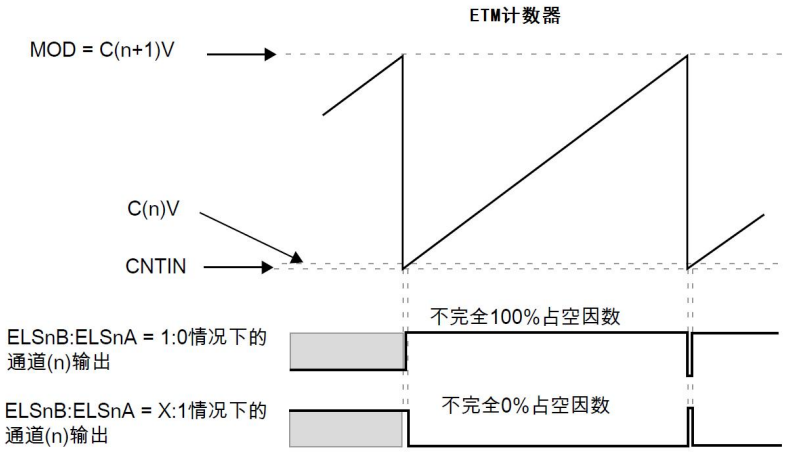


图 6-29 （CNTIN < C(n)V < MOD)、(C(n)V 约等于 CNTIN)且(C(n+1)V = MOD)条件下的通道(n)输出

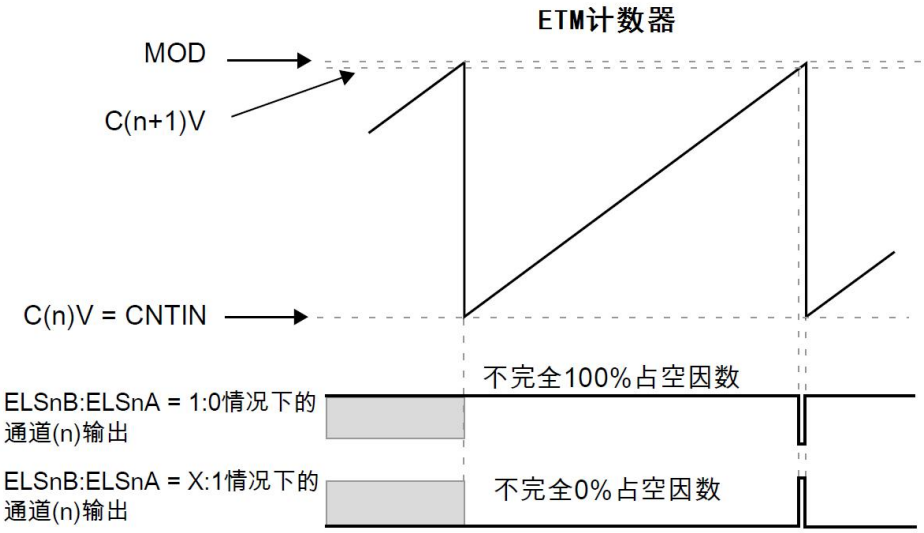


图 6-30 （C(n)V=CNTIN)、(CNTIN<C(n+1)V<MOD)且(C(n+1)V 约等于 MOD)条件下的通道(n)输出

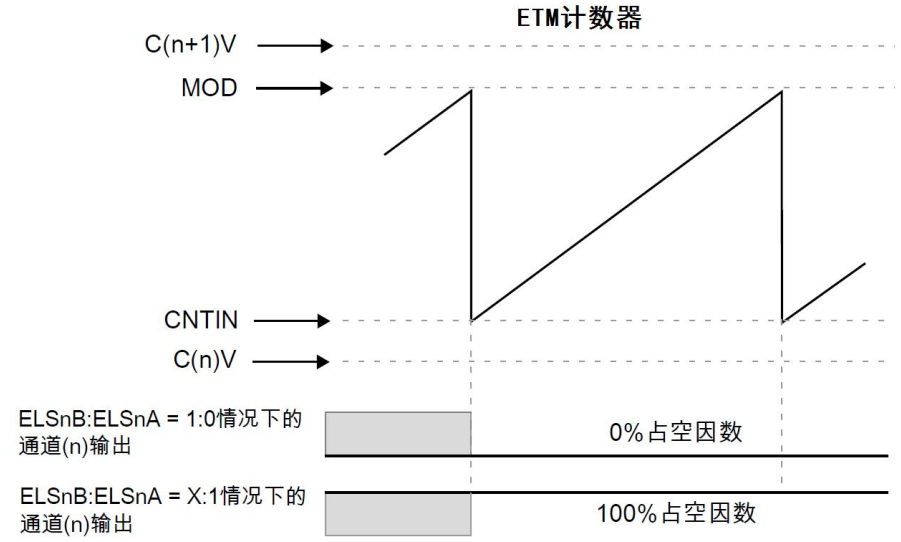


图 6-31　C(n)V 和 C(n+1)V 不在 CNTIN 和 MOD 之间条件下的通道(n)输出

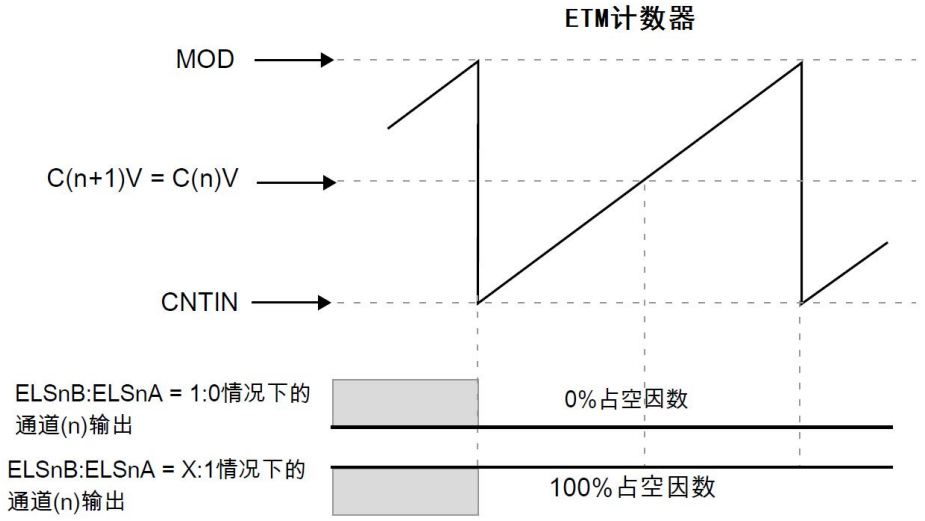


图 6-32　(CNTIN<C(n)V < MOD)、(CNTIN < C(n+1)V < MOD)且(C(n)V=C(n+1)V)条件下的通道(n)输出

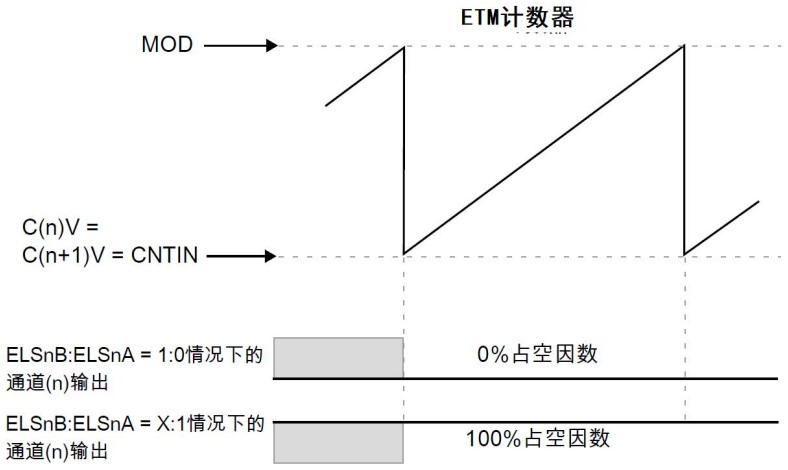


图 6-33　(C(n)V=C(n+1)V=CNTIN)条件下的通道(n)输出

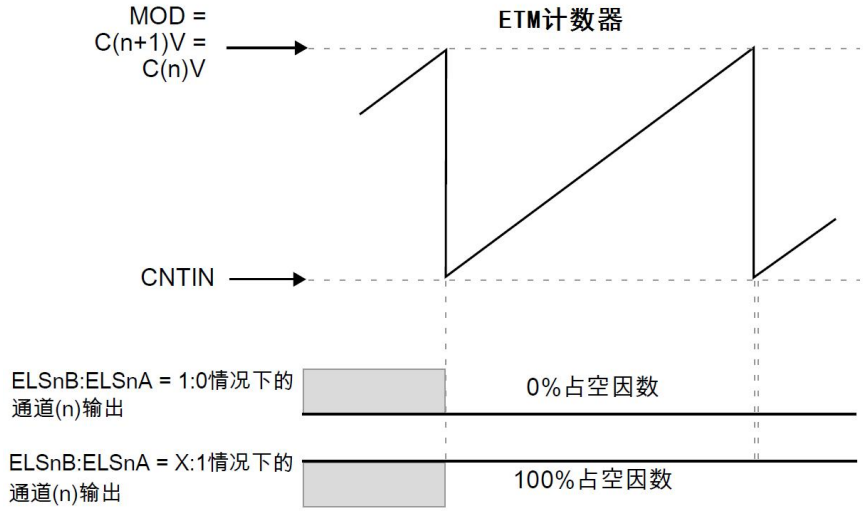


图 6-34 （C(n)V=C(n+1)V=MOD)条件下的通道(n)输出

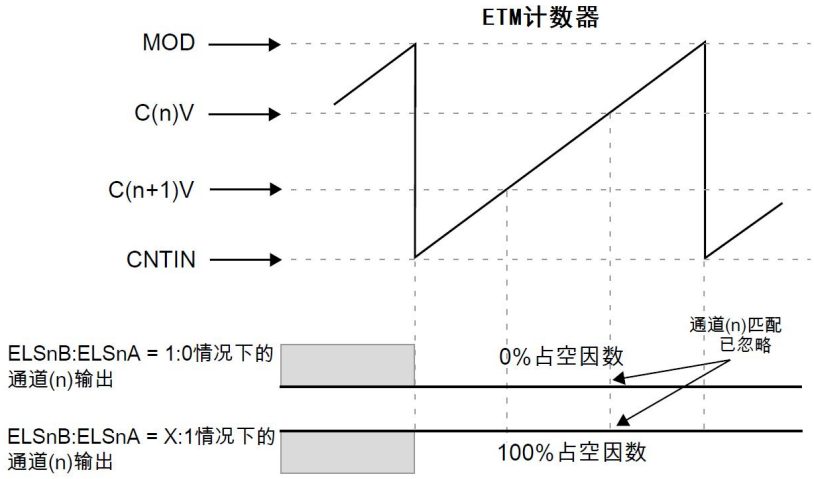


图 6-35 （CNTIN<C(n)V < MOD)、（CNTIN < C(n+1)V < MOD)且(C(n)V > C(n+1)V)条件下的通道(n)输出

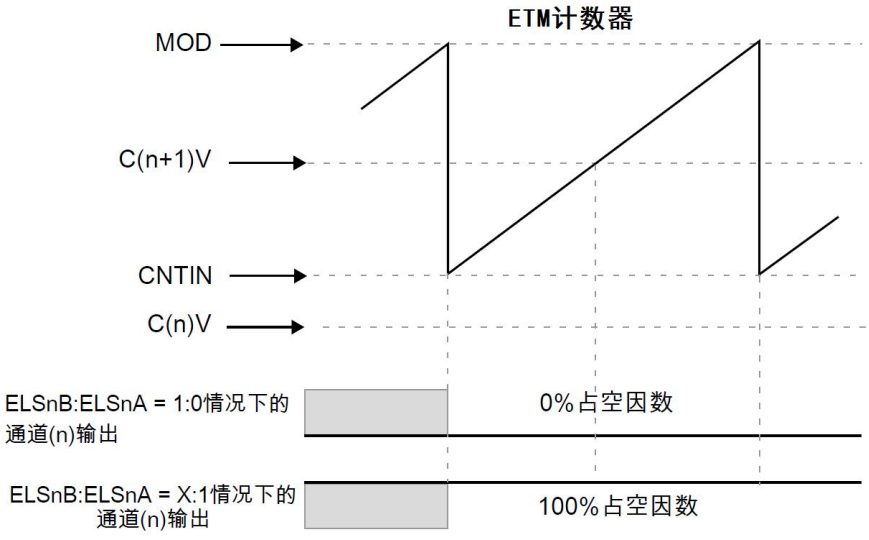


图 6-36 （C(n)V < CNTIN)且（CNTIN < C(n+1)V < MOD)条件下的通道(n)输出

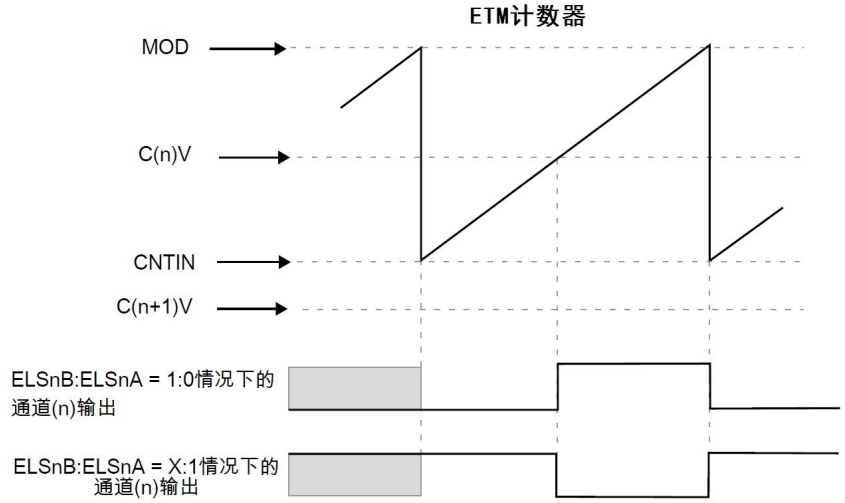


图 6-37 （C(n+1)V < CNTIN)且（CNTIN < C(n)V < MOD)条件下的通道(n)输出

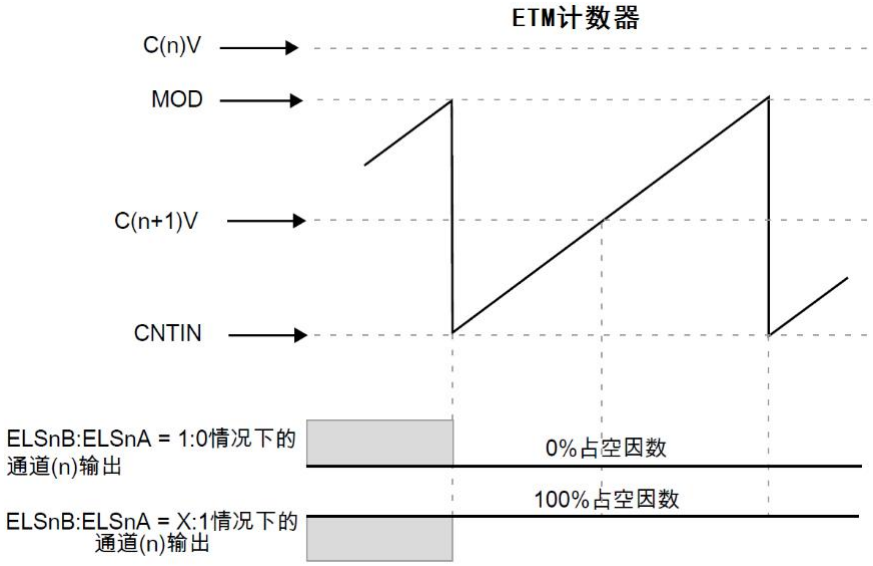


图 6-38 （C(n)V > MOD)且（CNTIN < C(n+1)V < MOD)条件下的通道(n)输出

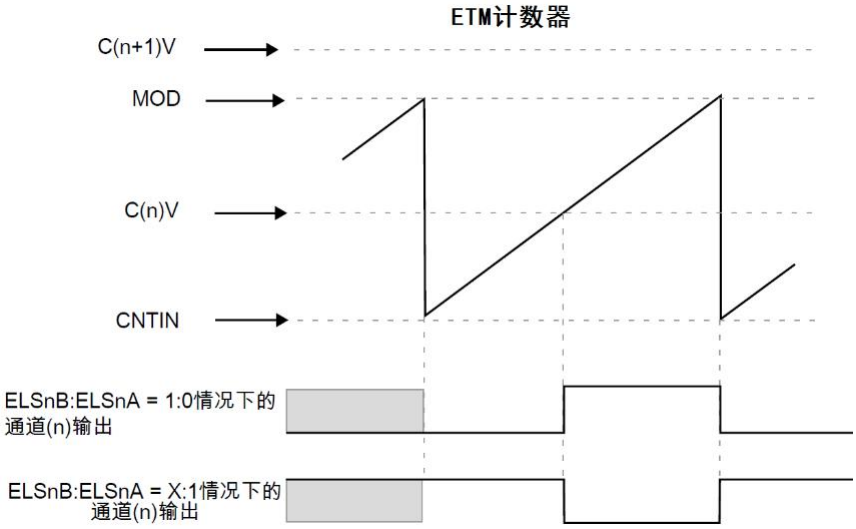


图 6-39 （C(n+1)V > MOD)且（CNTIN < C(n)V < MOD)条件下的通道(n)输出

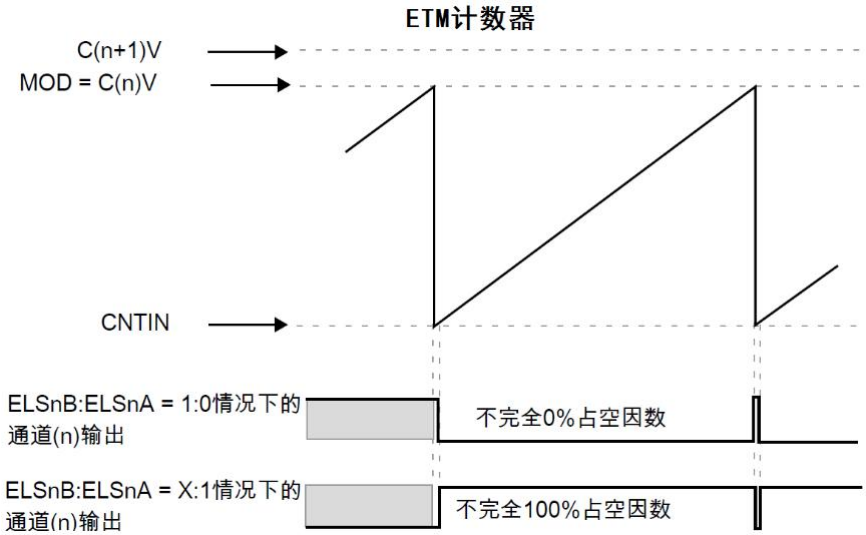


图 6-40 （C(n+1)V ＞ MOD)且（CNTIN ＜ C(n)V = MOD)条件下的通道(n)输出

❖ 6.3.8.8.1 不对称 PWM

在组合模式下，通道 n 发生匹配（即 ETM 计数器 = C(n)V）控制 PWM 信号的第一边沿，通道 n+1 发生匹配（即 ETM 计数器 = C(n+1)V）控制 PWM 信号的第二边沿，两者无关。因此，组合模式允许生成不对称的 PWM 信号。

● 6.3.8.9 互补模式

选择互补模式的配置如下：

- ETMEN = 1
- DECAPEN = 0
- COMBINE = 0
- CPWMS = 1，且
- COMP = 1

在互补模式下，通道(n+1)输出与通道(n)输出电平相反。

通道(n+1)输出与通道(n)输出相同的配置如下：

- ETMEN = 1
- DECAPEN = 0
- COMBINE = 1
- CPWMS = 0，且
- COMP = 0

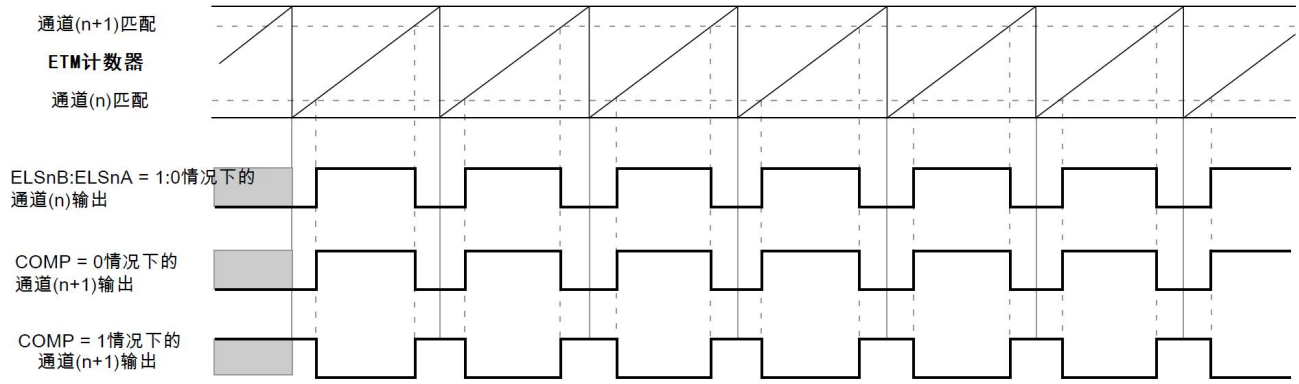


图 6-41 (ELSnB:ELSnA = 1:0)条件下互补模式中的通道(n+1)输出

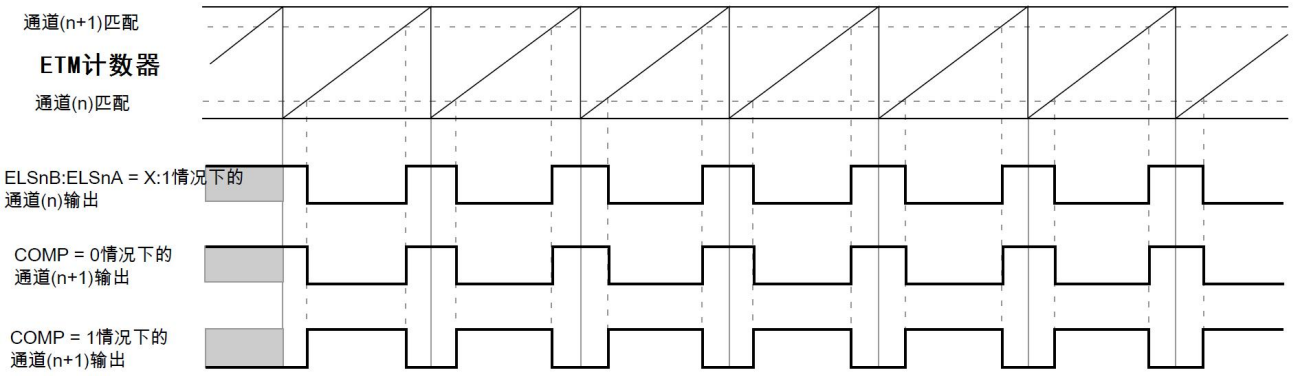


图 6-42 (ELSnB:ELSnA = X:1)条件下互补模式中的通道(n+1)输出

- **6.3.8.10 通过写缓存更新的寄存器**

❖ 6.3.8.10.1 CNTIN 寄存器更新

下表介绍了何时更新 CNTIN 寄存器：

表 6-42 CNTIN 寄存器更新

满足条件	然后更新 CNTIN 寄存器
CLKS[1:0]=0:0	对 CNTIN 寄存器采取写入操作时，与 ETMEN 位无关
ETMEN=0 或 CNTINC=0	对 CNTIN 寄存器采取写入之后的下一个系统时钟周期
ETMEN=1,SYNCMODE=1 且 CNTINC=1	参考 CNTIN 寄存器同步 章节

❖ 6.3.8.10.2 MOD 寄存器更新

下表介绍了何时更新 MOD 寄存器：

表 6-43 MOD 寄存器更新

满足条件	然后更新 MOD 寄存器
CLKS[1:0] = 0:0	对 MOD 寄存器采取写入操作时，与 ETMEN 无关
CLKS[1:0] ≠ 0:0 且 ETMEN = 0	根据 CPWMS 位，也就是： • 如果选择的模式不是 CPWM，则 MOD 寄存器在对 MOD 寄存器采取写入操作且 ETM 计数器从 MOD 更改为 CNTIN 之后更新。如果 ETM 计数器处于自由运行计数器模式，则此更新在 ETM 计数器从 0xFFFF 更改为 0x0000 时发生。 • 如果选择的模式为 CPWM，则 MOD 寄存器在对 MOD 寄存器采取写入操作且 ETM 计数器从 MOD 更改为(MOD - 0x0001)之后更新。
CLKS[1:0] ≠ 0:0 且 ETMEN = 1	参考 MOD 寄存器同步章节

❖ 6.3.8.10.3 CnV 寄存器更新

下表介绍了何时更新 CnV 寄存器：

表 6-44 CnV 寄存器更新

满足条件	然后更新 CnV 寄存器
CLKS[1:0] = 0:0	对 CnV 寄存器采取写入操作时，与 ETMEN 位无关

CLKS[1:0] ≠ 0:0 且 ETMEN = 0	根据选择的模式，即： • 如果选择的模式为输出比较，则 CnV 寄存器在下一次 ETM 计数器更改、预分频器计数结束时、对 CnV 寄存器采取写入操作之后更新。 • 如果选择的模式为 EPWM，则 CnV 寄存器在对 CnV 寄存器采取写入操作且 ETM 计数器从 MOD 更改为 CNTIN 之后更新。如果 ETM 计数器处于自由运行计数器模式，则此更新在 ETM 计数器从 0xFFFF 更改为 0x0000 时发生。 • 如果选择的模式为 CPWM，则 CnV 寄存器在对 CnV 寄存器采取写入操作且 ETM 计数器从 MOD 更改为(MOD - 0x0001)之后更新。
CLKS[1:0] ≠ 0:0 且 ETMEN = 1	根据选择的模式，即： • 如果选择的模式为输出比较，则 CnV 寄存器根据 SYNCEN 位更新。如果(SYNCEN = 0)，则在下一次更改 ETM 计数器、预分频器计数结束时对 CnV 寄存器采取写入操作。如果(SYNCEN = 1)，则根据 C(n)V 和 C(n+1)V 寄存器同步 更新 CnV 寄存器。 • 如果选择的模式不是输出比较且(SYNCEN = 1)，则根据 C(n)V 和 C(n+1)V 寄存器同步 更新 CnV 寄存器。

● 6.3.8.11 PWM 同步

通过 PWM 同步将有机会以 MOD、CNTIN、CnV、OUTMASK、INVCTRL 和 SWOCTRL 寄存器各自的缓存值对这些寄存器进行更新，并强制 ETM 计数器为 CNTIN 寄存器值。

注 • PWM 同步只能用于组合模式。

• 传统 PWM 同步(SYNCMODE = 0)是增强型 PWM 同步(SYNCMODE = 1)的子集。因此，只能使用增强型 PWM 同步。

❖ 6.3.8.11.1 硬件触发

TRIGn = 1（其中 n = 0、 1 或 2，各自对应于每一个输入信号）时，将使能 ETM 模块的三个硬件触发信号输入。硬件触发输入 n 由系统时钟同步。在使能的硬件触发输入上检测到上升沿时，将启动采用硬件触发的 PWM 同步。

如果(HWTRIGMODE = 0)，则当 0 被写入或者触发 n 事件被检测到时，对应的 TRIGn 会被清零。

这种情况下，如果使能了两个或更多硬件触发（例如， TRIG0 和 TRIG1 = 1），但只发生了触发 1 事件，则只清零 TRIG1 位。如果在发生触发 n 事件的同时还通过写操作使 TRIGn 位置位，则会启动同步，但 TRIGn 位仍会因写操作而保持置位。

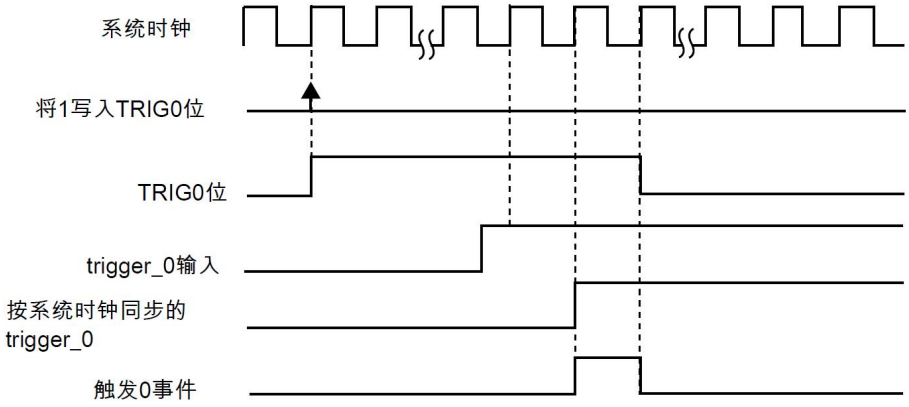


图 6-43 HWTRIGMODE=0 时的硬件触发事件

如果 HWTRIGMODE = 1，则 TRIGn 位只在向其写 0 的情况下清零。所有硬件触发都具有相同的行为方式。

注　只有在采用增强型 PWM 同步的情况下(SYNCMODE = 1)，才能使 HWTRIGMODE 位为 1。

❖ 6.3.8.11.2 软件触发

向 SYNC[SWSYNC]位写入 1 时，将发生软件触发事件。向 SWSYNC 位写入 0 时即清零此位，由软件事件启动的 PWM 同步完成时，也会清零此位。

如果在上一个软件触发事件发起的 PWM 同步结束的同时（通过向 SWSYNC 位再次写入 1）再次发生软件触发事件，将开始新的 PWM 同步，且 SWSYNC 位保持等于 1。

如果 SYNCMODE = 0，则 SWSYNC 位也会根据 PWMSYNC 和 REINIT 位由 ETM 清零。这种情况下，如果(PWMSYNC = 1)或（PWMSYNC = 0 且 REINIT = 0），SWSYNC 位将在软件触发事件发生后于所选的下一个加载点清零； 参见边界周期和加载点 和下图。如果(PWMSYNC =0)且(REINIT=1)，SWSYNC 位将在软件触发事件发生时清零。

如果 SYNCMODE = 1，则 SWSYNC 位也会根据 SWRSTCNT 位由 ETM 清零。如果 SWRSTCNT = 0，SWSYNC 位将在软件触发事件发生后于所选的下一个加载点清零； 参见下图：如果 SWRSTCNT = 1，SWSYNC 位将在软件触发事件发生时清零。

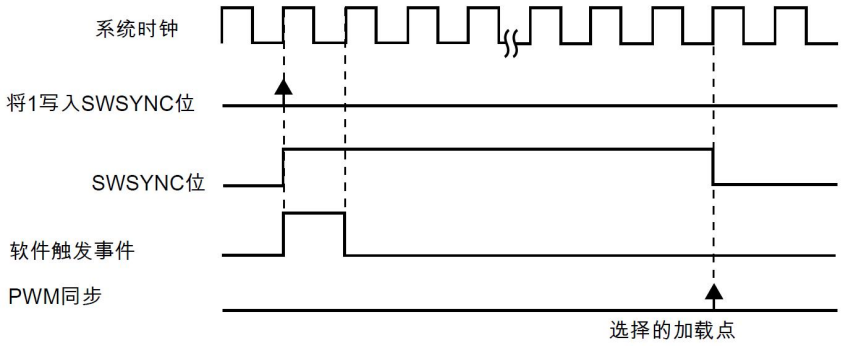


图 6-44 软件触发事件

❖ 6.3.8.11.3 边界周期和加载点

边界周期定义对寄存器 MOD、CNTIN 和 C(n)V 的加载点很重要。

在向上计数模式下，边界周期定义为计数器变为其初始值(CNTIN)的时候。如果是在向上-向下计数 模

式下，边界周期则定义为计数器从向下计数变为向上计数的时候以及从向上计数变为向下计数的时候。

下表显示了寄存器的边界周期和加载点。在向上计数模式中，如果有一个 CNTMIN 或 CTMAX 位为 1，则使能加载点。在自上而下计数模式中，如图所示，由 CNTMIN 和 CNTMAX 位选择加载点。这些加载点对寄存器更新而言是一个安全之处，从而可在 PWM 波形生成过程中实现顺畅的过渡。

在这两种计数模式中，如果 CNTMIN 和 CNTMAX 都不是 1，则边界周期不用作寄存器更新的加载点。有关详细信息，请参见以下各节中的寄存器同步说明。

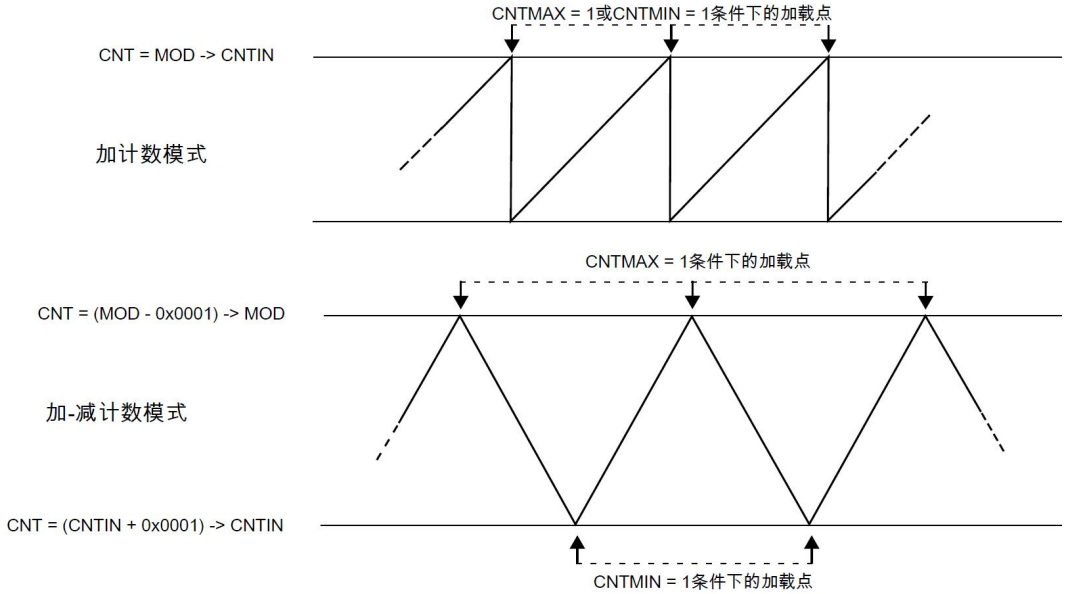


图 6-45 边界周期和加载点

❖ 6.3.8.11.4 MOD 寄存器同步

MOD 寄存器同步将以其缓存值对 MOD 寄存器进行更新。如果(ETMEN = 1)，将使能这种同步。

MOD 寄存器同步可通过增强型 PWM 同步(SYNCMODE = 1)或传统 PWM 同步(SYNCMODE = 0)完成。但是，更希望只通过增强型 PWM 同步来同步 MOD 寄存器。

采用增强型 PWM 同步时，依据下述流程图，MOD 寄存器同步取决于 SWWRBUF、SWRSTCNT、HWWRBUF 和 HWRSTCNT 位：

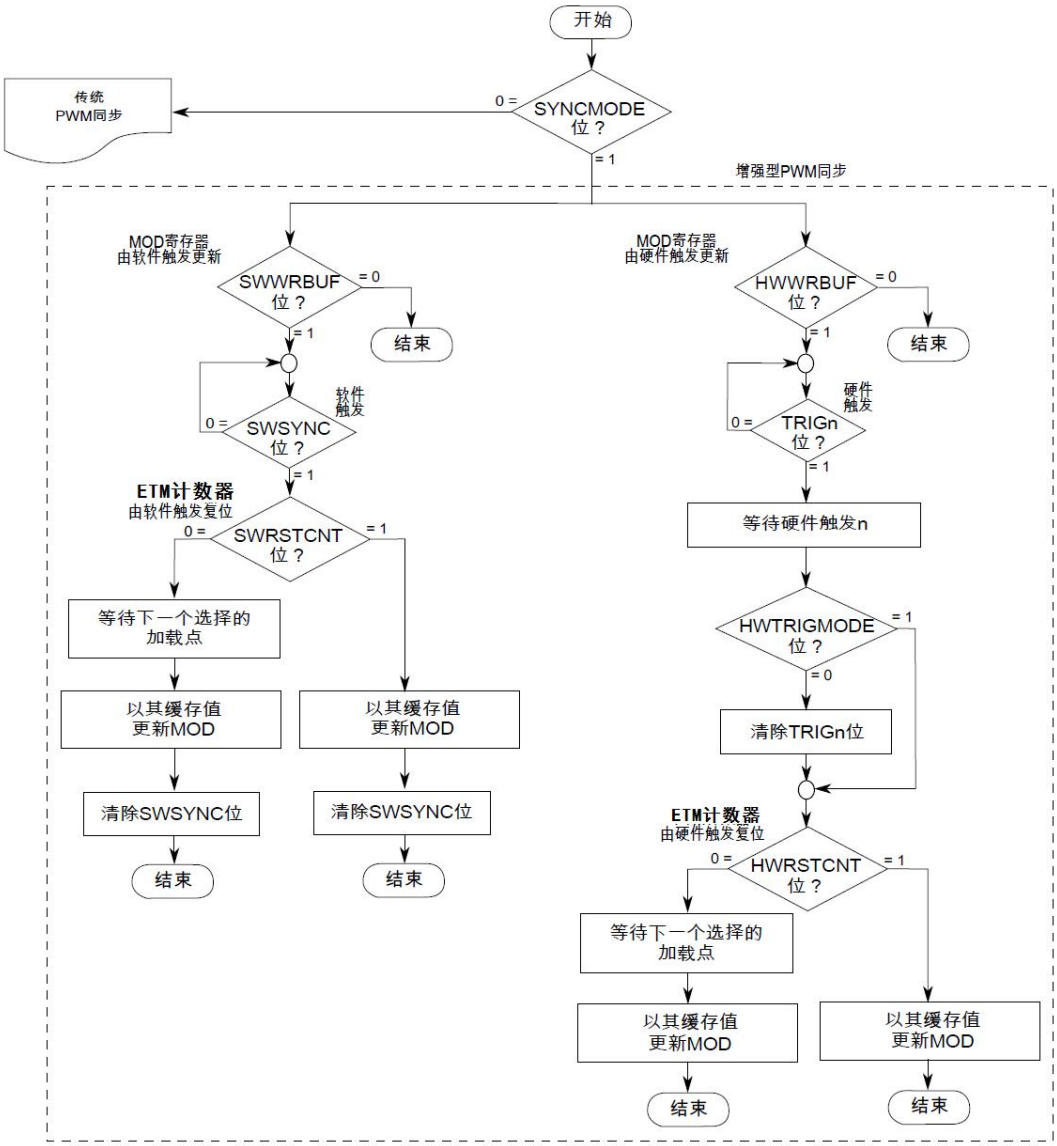


图 6-46 MOD 寄存器同步流程图

采用传统 PWM 同步时，依据下述说明，MOD 寄存器同步取决于 PWMSYNC 和 REINIT 位。

如果(SYNCMODE = 0)、 (PWMSYNC = 0)且(REINIT = 0)，则在发生所使能的触发事件之后、在选择的下一个加载点进行此同步。如果触发事件为软件触发，则在选择的下一个加载点清除 SWSYNC 位。如果触发事件为硬件触发，则根据硬件触发清除触发器使能位(TRIGn)。以下为采用软件和硬件触发的示例。

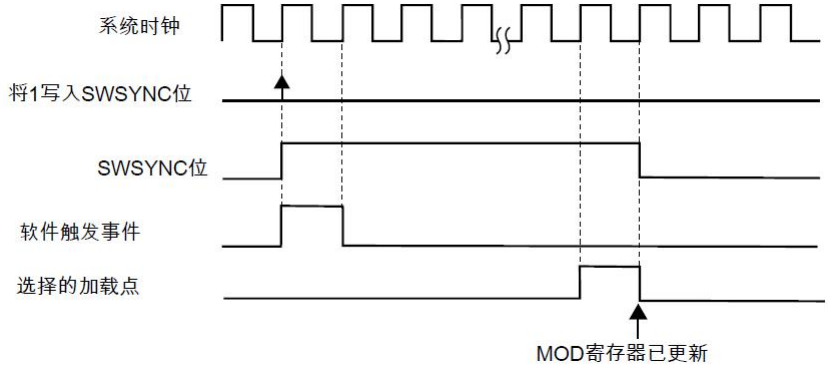


图 6-47 （SYNCMODE=0)、（PWMSYNC=0）、（REINIT=0)且使用软件触发时的 MOD 同步

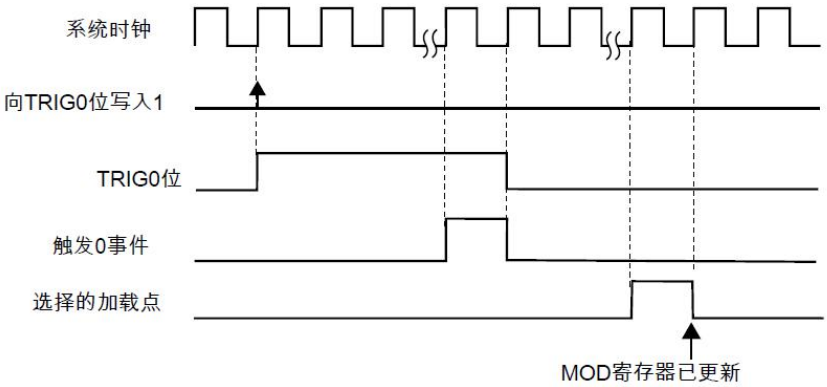


图 6-48 （SYNCMODE=0）、（HWTRIGMODE=0）、（PWMSYNC=0）、（REINIT=0）且使用硬件触发时的 MOD 同步

如果（SYNCMODE = 0）、（PWMSYNC = 0）且（REINIT = 1），则在下一次发生所使能的触发事件时进行此同步。如果触发事件为软件触发，则根据以下示例清除 SWSYNC 位。如果触发事件为硬件触发，则根据硬件触发 清除 TRIGn 位。以下为采用软件和硬件触发的示例。

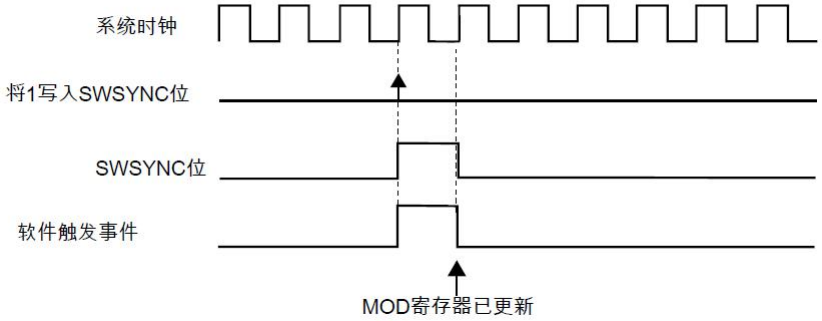


图 6-49 （SYNCMODE=0）、（PWMSYNC=0）、（REINIT=1）且使用软件触发时的 MOD 同步

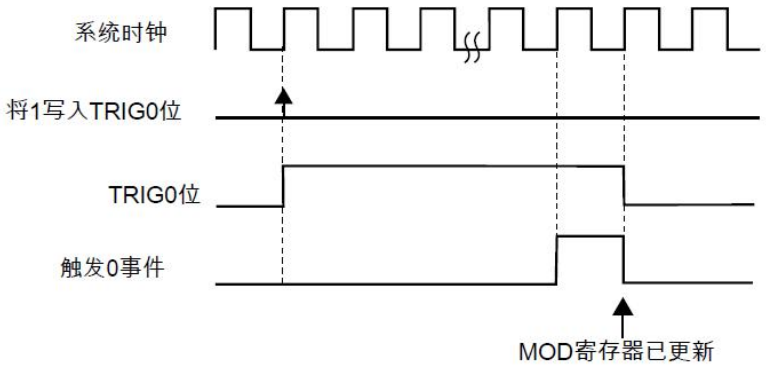


图 6-50 （SYNCMODE=0）、（HWTRIGMODE=0）、（PWMSYNC=0）、（REINIT=1）且使用硬件触发时的 MOD 同步

如果（SYNCMODE = 0）且（PWMSYNC = 1），则在发生软件触发事件之后、在选择的下一个加载点进行此同步。在选择的下一个加载点清除 SWSYNC 位：

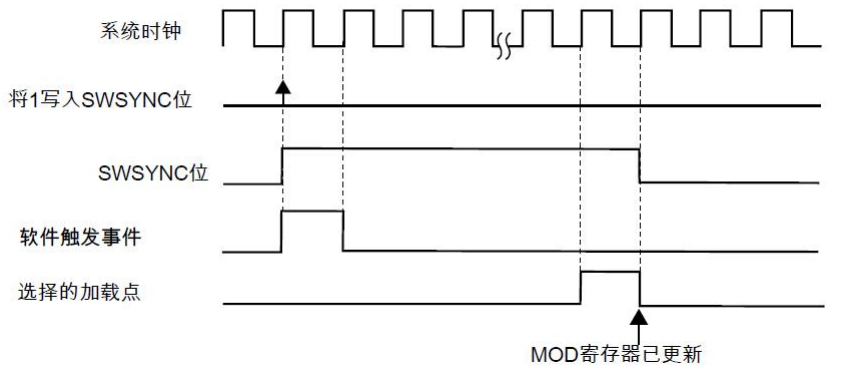


图 6-51 （SYNCMODE=0)且（PWMSYNC=1）时的 MOD 同步

❖ 6.3.8.11.5 CNTIN 寄存器同步

CNTIN 寄存器同步将以其缓存值对 CNTIN 寄存器进行更新。

如果(ETMEN = 1)、(SYNCMODE = 1)且(CNTINC = 1)，将使能这种同步。只能通过增强型 PWM 同步(SYNCMODE = 1)完成 CNTIN 寄存器同步。同步机制与通过增强型 PWM 同步完成的 MOD 寄存器同步相同；参见 MOD 寄存器同步 。

❖ 6.3.8.11.6 C(n)V 和 C(n+1)V 寄存器同步

C(n)V 和 C(n+1)V 寄存器同步用其缓存值对 C(n)V 和 C(n+1)V 寄存器进行更新。

如果(ETMEN = 1)且(SYNCEN = 1)，则会使能该同步。同步机制与 MOD 寄存器同步相同。但是，希望只通过增强型 PWM 同步(SYNCMODE = 1)完成 C(n)V 和 C(n+1)V 寄存器的同步。

❖ 6.3.8.11.7 OUTMASK 寄存器同步

OUTMASK 寄存器同步将以其缓存值对 OUTMASK 寄存器进行更新。

可通过增强型 PWM 同步（ SYNCHOM = 1 且 SYNCMODE = 1）或传统 PWM 同步（ SYNCHOM = 1 且 SYNCMODE = 0）在系统时钟的每个上升沿更新 OUTMASK 寄存器 OM(m):SYNCHOM = 0。但是，希望只通过增强型 PWM 同步来同步 OUTMASK 寄存器。

采用增强型 PWM 同步时，OUTMASK 寄存器同步取决于 SWOM 和 HWOM 位。参见以下流程图：

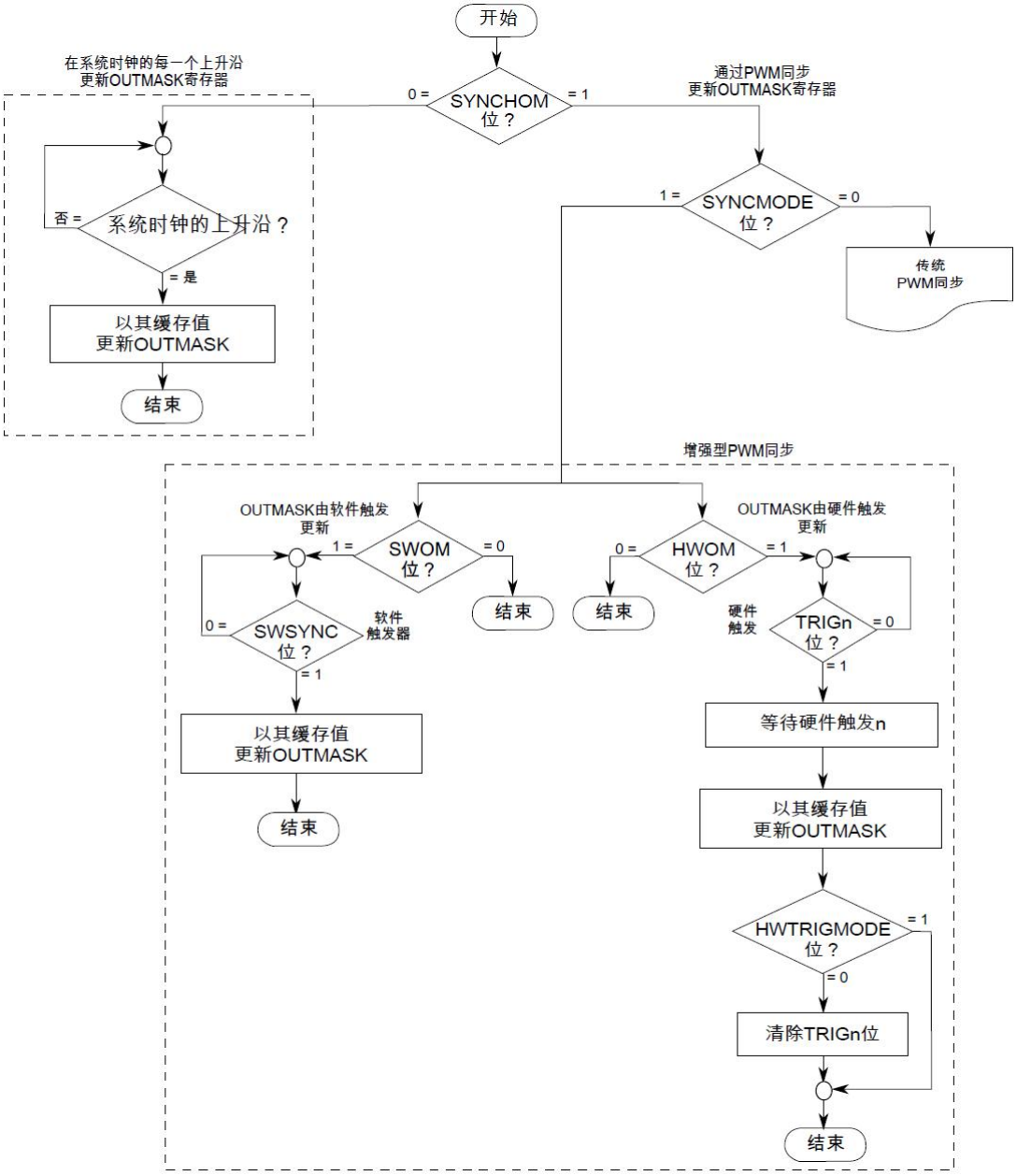


图 6-52 OUTMASK 寄存器同步流程图

采用传统 PWM 同步时，依据下述说明，OUTMASK 寄存器同步取决于 PWMSYNC 位。

如果(SYNCMODE = 0)、(SYNCHOM = 1)且(PWMSYNC = 0)，则在下一次发生所使能的触发事件时进行此同步。如果触发事件为软件触发，则在选择的下一个加载点清除 SWSYNC 位。如果触发事件为硬件触发，则根据硬件触发清除 TRIGn 位。以下为采用软件和硬件触发的示例。

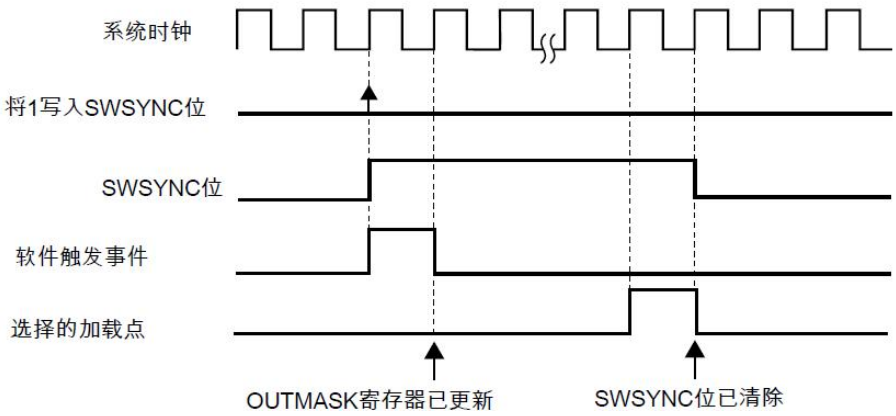


图 6-53 （SYNCMODE=0）、（SYNCHOM=1）、（PWMSYNC=0）且使用软件触发时的 OUTMASK 同步

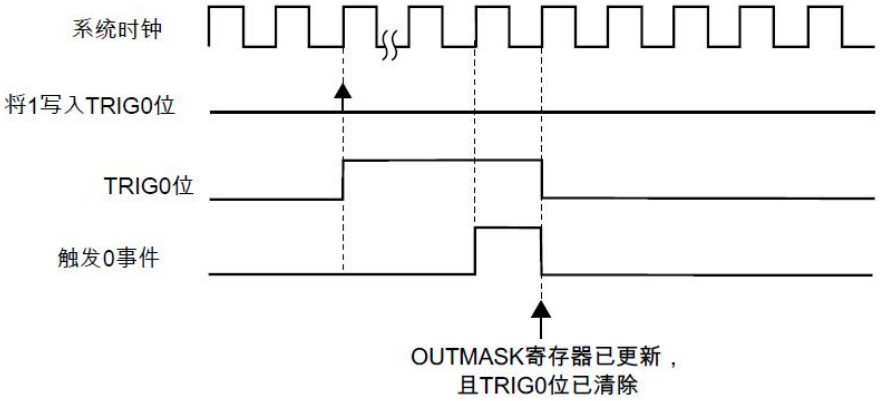


图 6-54 （SYNCMODE=0）、（HWTRIGMODE=0）（SYNCHOM=1）、（PWMSYNC=0）且使用硬件触发时的 OUTMASK 同步

如果(SYNCMODE = 0)、(SYNCHOM = 1)且(PWMSYNC = 1)，则在下一次发生所使能的硬件触发事件时进行此同步。根据硬件触发清除 TRIGn 位。以下是采用硬件触发的示例。

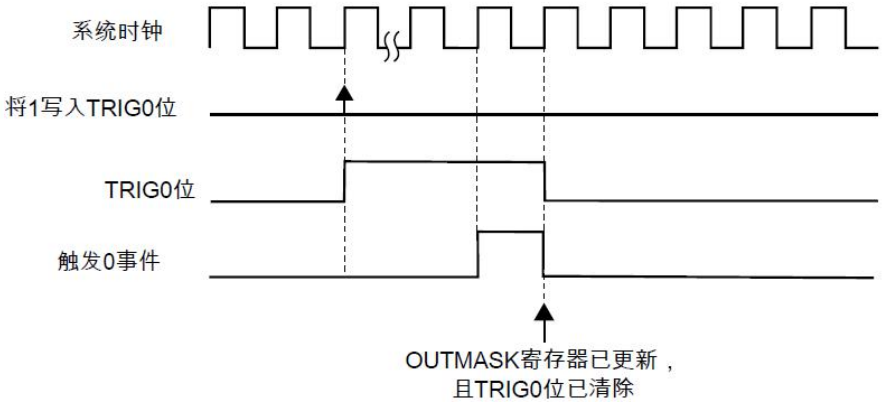


图 6-55 （SYNCMODE=0）、（HWTRIGMODE=0）（SYNCHOM=1）、（PWMSYNC=1）且使用硬件触发时的 OUTMASK 同步

❖ 6.3.8.11.8 INVCTRL 寄存器同步

INVCTRL 寄存器同步将通过其缓存值对 INVCTRL 寄存器进行更新。

INVCTRL 寄存器可在系统时钟的每个上升沿(INVC = 0)更新，或者通过增强型 PWM 同步（INVC = 1 且

SYNCMODE = 1）根据以下流程图更新。

采用增强型 PWM 同步时， INVCTRL 寄存器同步取决于 SWINVC 和 HWINVC 位。

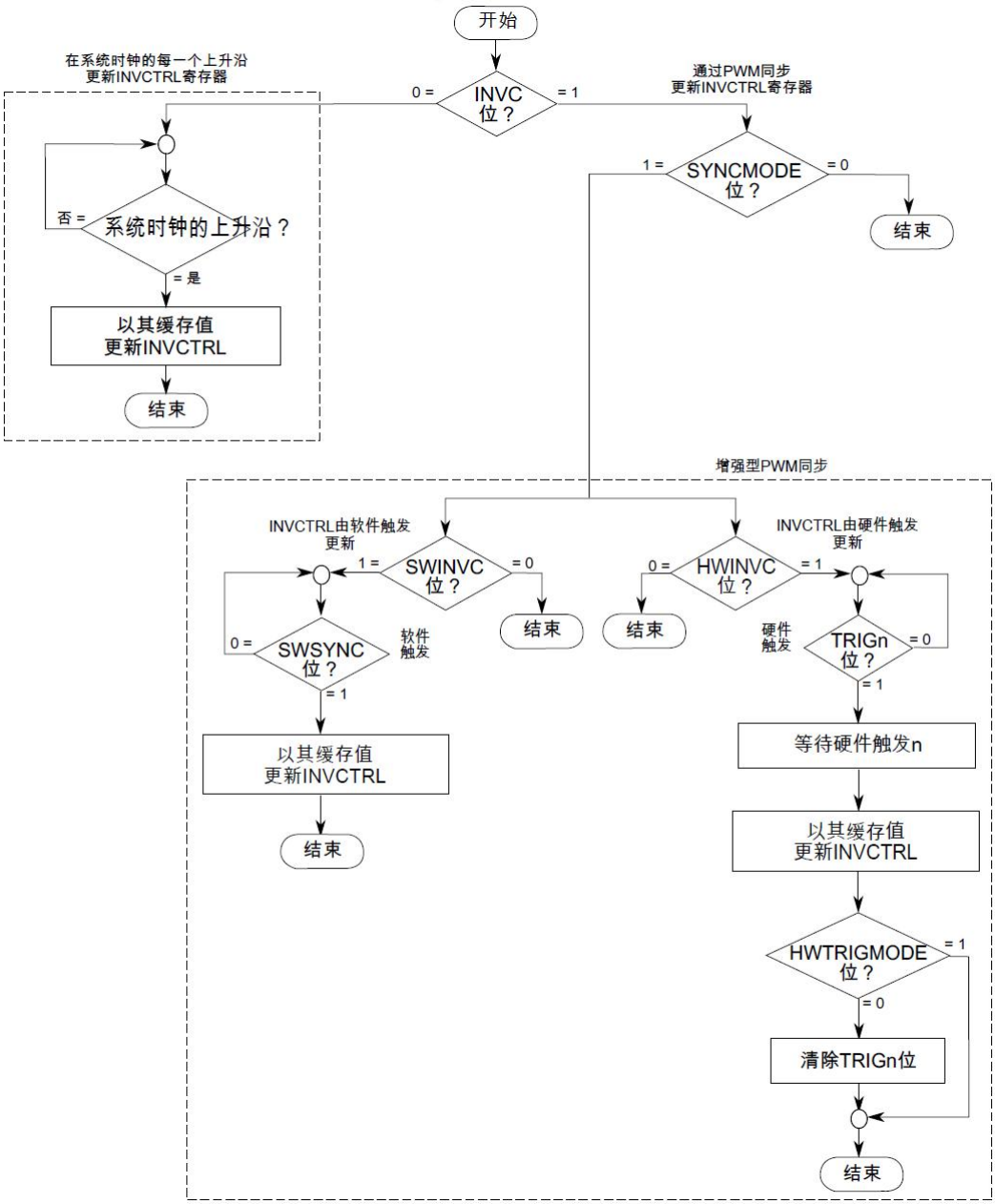


图 6-56 INVCTRL 寄存器同步流程图

❖ 6.3.8.11.9 SWOCTRL 寄存器同步

SWOCTRL 寄存器同步将以其缓存值对 SWOCTRL 寄存器进行更新。SWOCTRL 寄存器可在系统时钟的每个上升沿(SWOC = 0)更新，或者通过增强型 PWM 同步（ SWOC = 1 且 SYNCMODE = 1）根据以下流程图更新。

采用增强型 PWM 同步时，SWOCTRL 寄存器同步取决于 SWSOC 和 HWSOC 位。

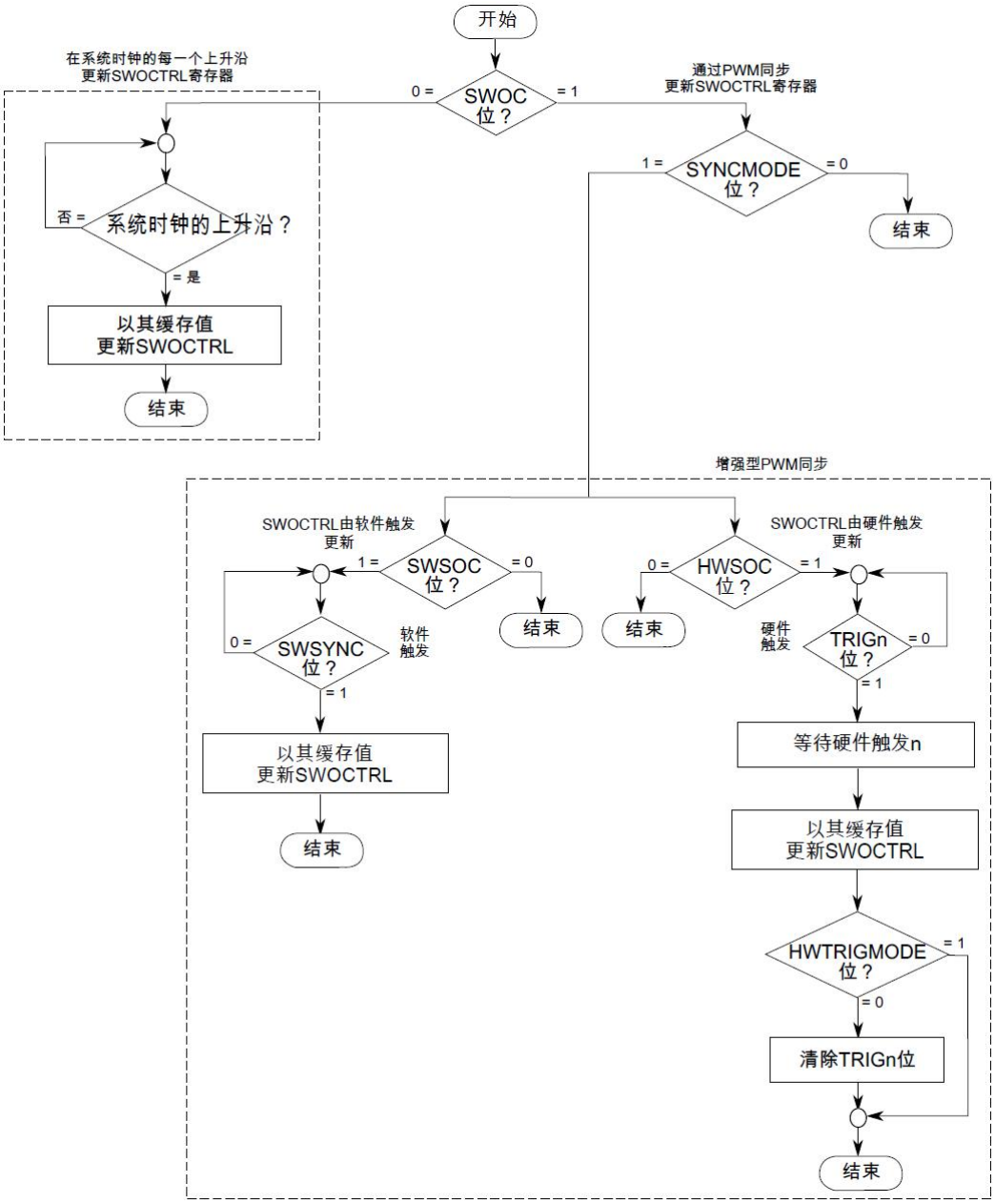


图 6-57 SWOCTRL 寄存器同步流程图

❖ 6.3.8.11.10 ETM 计数器同步

ETM 计数器同步是一种机制，通过这种机制 ETM 可以在 PWM 周期中的特定点重新开始生成 PWM。通道输出强制为各自的初始值（处于输出比较模式的通道则例外），ETM 计数器强制为 CNTIN 寄存器定义的初始计数值。

下图展示了 ETM 计数器同步情况。注意，同步事件发生后，通道(n)被设置为其初始值，通道(n+1)则不会被设置为其初始值，这是由此图中的特定时序造成的，在此图中死区插入阻止了该通道输出转换为 1。如果没有选择死区插入，则通道(n+1)在同步事件发生后会立即转换为逻辑值 1

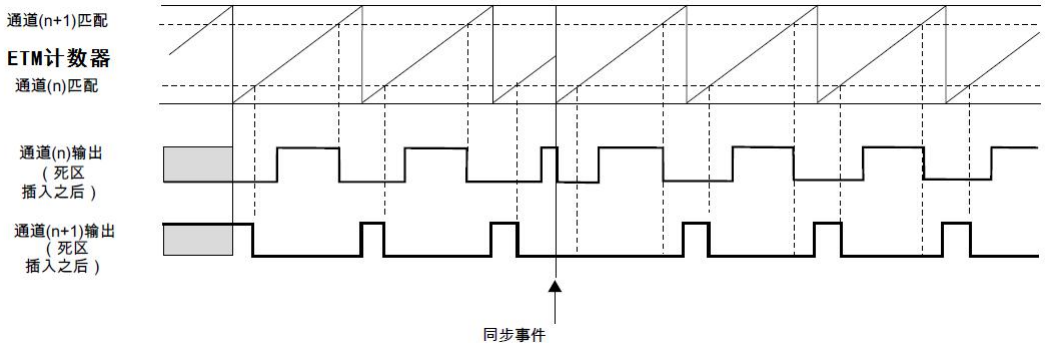


图 6-58 ETM 计数器同步

ETM 计数器同步可通过增强型 PWM 同步(SYNCMODE=1)或传统 PWM 同步(SYNCMODE= 0)完成。但是，只能通过增强型 PWM 同步来同步 ETM 计数器。采用增强型 PWM 同步时，依据下述流程图，ETM 计数器同步取决于 SWRSTCN 和 HWRSTCNT 位。

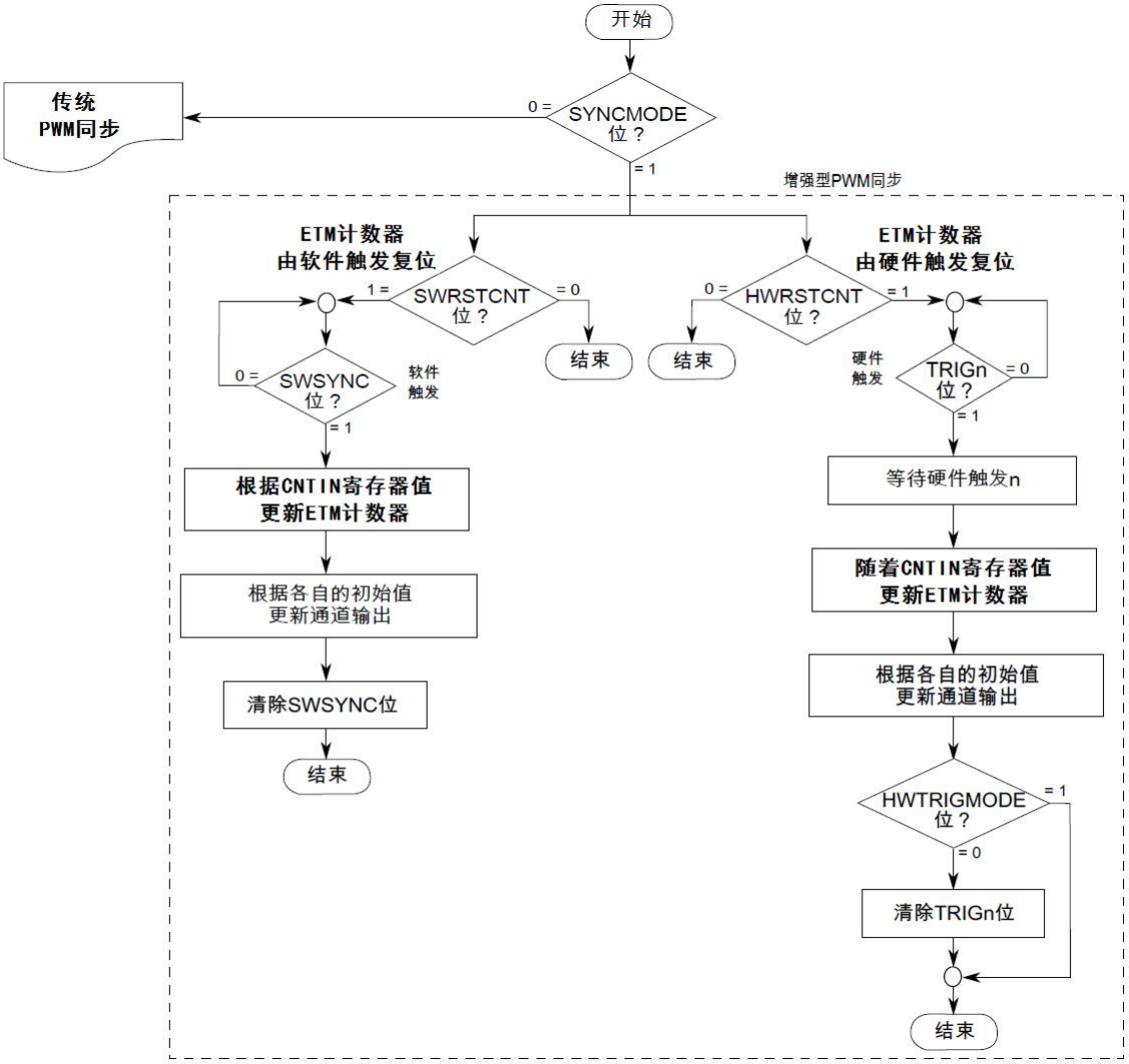


图 6-59 ETM 计数器同步流程图

采用传统 PWM 同步时，依据下述说明，ETM 计数器同步取决于 REINIT 和 PWMSYNC 位。

如果(SYNCMODE = 0)、 (REINIT = 1)且(PWMSYNC = 0)，则在下一次触发事件使能时进行此同步。如

果触发事件为软件触发，则根据以下示例清零 SWSYNC 位。

如果触发事件为硬件触发，则根据硬件触发清零 TRIGn 位。以下为采用软件和硬件触发的示例。

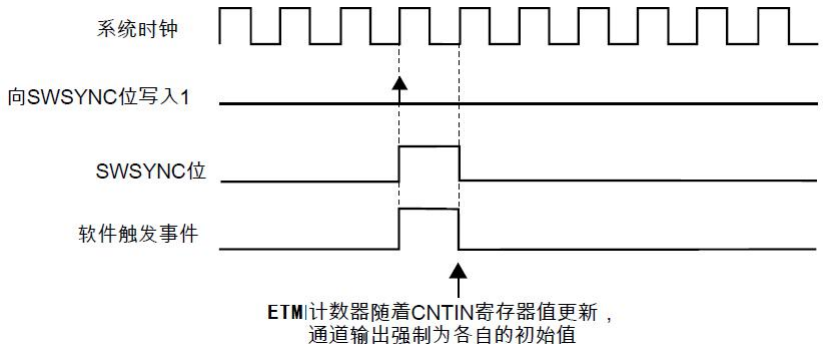


图 6-60 （SYNCMODE=0）、（REINIT=1）、（PWMSYNC=0）且使用软件触发时的 ETM 计数器同步

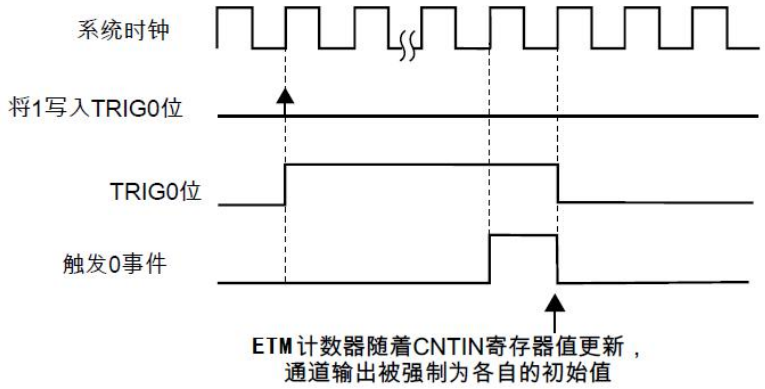


图 6-61 （SYNCMODE=0）、（HWTRIGMODE=0）、（REINIT=1）、（PWMSYNC=0）且使用硬件触发时的 ETM 计数器同步

如果(SYNCMODE = 0)、(REINIT = 1)且(PWMSYNC = 1)，则在下一次硬件触发使能时进行此同步。根据硬件触发 清零 TRIGn 位。

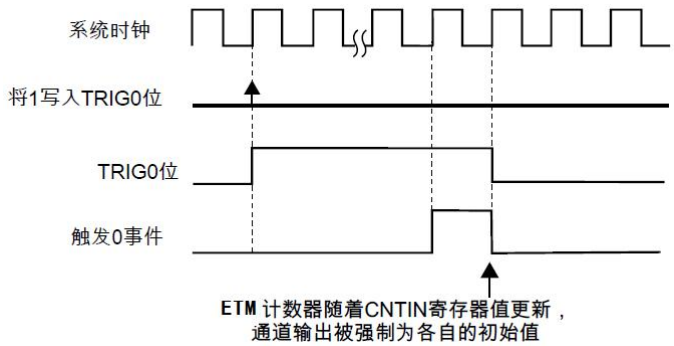


图 6-62 （SYNCMODE=0）、（HWTRIGMODE=0）、（REINIT=1）、（PWMSYNC=1）且使用硬件触发时的 ETM 计数器同步

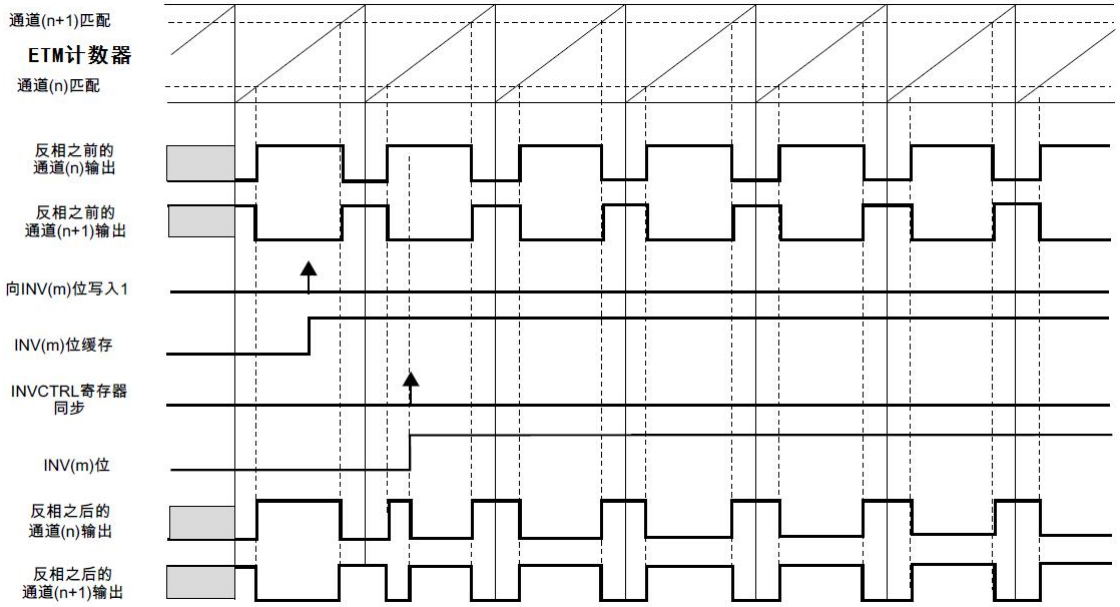
● 6.3.8.12 反相

反相功能在通道(n)和通道(n+1)输出之间交换信号。在以下情形中将选择反相操作：

- ETMEN = 1
- DECAPEN = 0
- COMBINE = 1
- COMP = 1
- CPWMS = 0，且
- INVm = 1（其中 m 代表一对通道）

INVCTRL 寄存器中的 INVm 位将根据 INVCTRL 寄存器同步以其缓冲值更新。

在高电平有效(ELSnB:ELSnA = 1:0)组合模式下，通道(n)输出在周期开始（ ETM 计数器 = CNTIN）时被强制为低电平，在通道(n)匹配时被强制为高电平，在通道(n+1)匹配时被强制为低电平。如果选择了反相，通道(n)输出的行为方式将变更为：在 PWM 周期开始时被强制为高电平，在通道(n)匹配时被强制为低电平，在通道(n+1)匹配时被强制为高电平。参见下图：



注释：INV(m)位选.通道对 n)和(n+1)的反相。

图 6-63 高电平有效(ELSnB:ELSnA=1:0)组合模式下反相之后的通道(n)和(n+1)输出

注意，应当考虑 ELSnB:ELSnA 位值，因为它们定义了通道输出的有效状态。在低电平有效(ELSnB:ELSnA = X:1)组合模式下，通道(n)输出在周期开始时被强制为高电平，在通道(n)匹配时被强制为低电平，在通道(n+1)匹配时被强制为高电平。选择反相后，通道(n)和(n+1)呈现的波形将如下图所示。

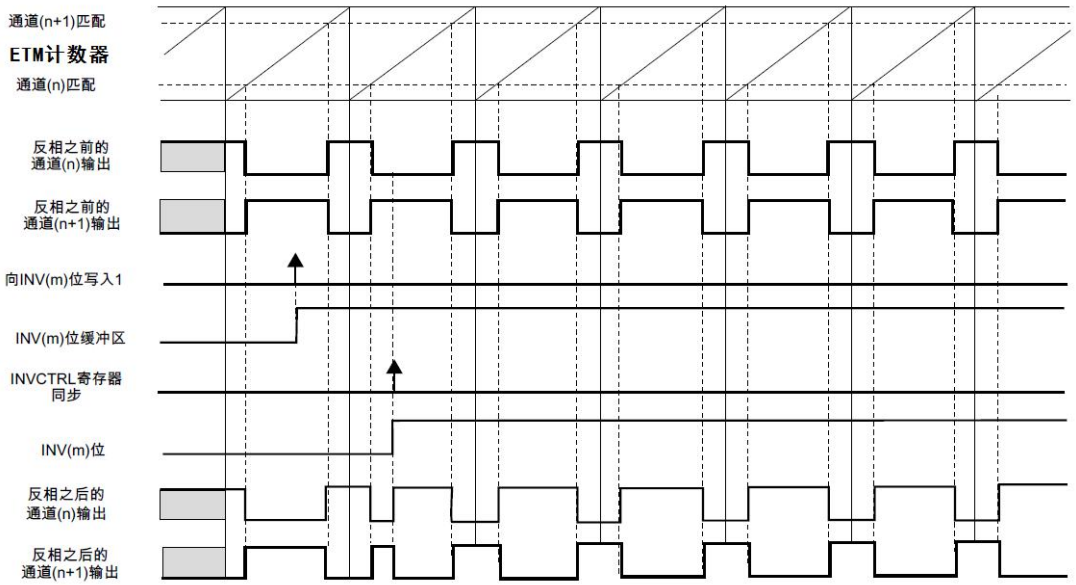


图 6-64 低电平有效(ELSnB:ELSnA=X:1)组合模式下反相之后的通道(n)和(n+1)输出

注释：INV(m)位选.通道对 n)和(n+1)的反相。反相特性只能用于组合模式。

● 6.3.8.13 软件输出控制

软件输出控制可在 ETM 生成过程中于特定时间点强制通道输出软件定义的值。

在以下情形中会选择软件输出控制：

- ETMEN = 1
- DECAPEN = 0
- COMBINE = 1
- CPWMS = 0，且
- CHnOC = 1

CHnOC 位使能针对特定通道输出的软件输出控制， CHnOCV 选择强制此通道输出的值。

SWOCTRL 寄存器中的 CHnOC 和 CHnOCV 位均根据 SWOCTRL 寄存器同步缓存，并以各自的缓冲区值更新。

下图展示了采用软件输出控制时的通道(n)和(n+1)输出信号。本例中，通道(n)和(n+1)设置为组合与互补模式。注:INV(m)位选.通道对 n)和(n+1)的反相。

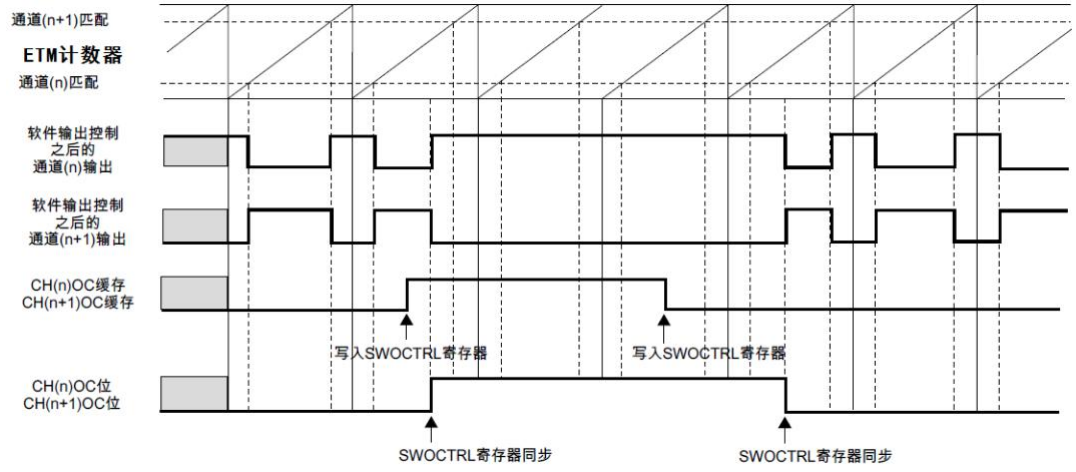


图 6-65 组合和互补模式下的软件输出控制示例

COMP 位为 0 时，软件输出控制强制通道(n)和(n+1)输出以下值。

表 6-45 (COMP=0)时的软件输出控制行为方式

CH(n) OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	通道(n)输出	通道(n+1) 输出
0	0	X	X	没有被 SWOC 修改	没有被 SWOC 修改
1	1	0	0	强制为 0	强制为 0
1	1	0	1	强制为 0	强制为 1
1	1	1	0	强制为 1	强制为 0
1	1	1	1	强制为 1	强制为 1

COMP 位为 1 时，软件输出控制强制通道(n)和(n+1)输出以下值。

表 6-46 (COMP=1)时的软件输出控制行为方式

CH(n) OC	CH(n+1)OC	CH(n)OCV	CH(n+1)OCV	通道(n)输出	通道(n+1) 输出
0	0	X	X	没有被 SWOC 修改	没有被 SWOC 修改
1	1	0	0	强制为 0	强制为 0
1	1	0	1	强制为 0	强制为 1
1	1	1	0	强制为 1	强制为 0
1	1	1	1	强制为 1	强制为 0

注

软件输出控制特性只能用于组合模式。

CH(n)OC 和 CH(n+1)OC 位应该相等。

使能软件输出控制（即 CH(n)OC = 1 和/或 CH(n+1)OC =1）时，不得修改 COMP 位。

软件输出控制的行为方式与处于禁用或使能状态的

ETM 计数器相同（参见状态和控制寄存器中的 CLKS 字段说明）。

● 6.3.8.14 死区插入

当(DTEN = 1)且 （ DTVAL[5:0]为非零）时，将使能死区插入。

DEADTIME 寄存器定义可用于所有 ETM 通道的死区延迟。DTPS[1:0]位定义系统时钟的预分频器，DTVAL[5:0]位则定义死区模数，即死区预分频器时钟的数量。

死区延迟插入可确保不会有两个互补信号（通道(n)和(n+1)）同时驱动有效状态。

如果 POL(n)=0、 POL(n+1)=0 且死区处于使能状态，那么在出现通道(n)匹配（ ETM 计数器 =C(n)V）情况时，通道(n)输出将保持低值，直到通道(n)输出置位、死区延迟结束时。与此类似，在出现通道(n+1)匹配（ ETM 计数器 = C(n+1)V）情况时，通道(n+1)输出将保持低值，直到通道(n+1)输出置位、死区延迟结束时。参见下图：

如果 POL(n)=1、 POL(n+1)=1 且死区处于使能状态，那么在出现通道(n)匹配（ ETM 计数器 =C(n)V）情况时，通道(n)输出将保持高值，直到通道(n)输出清除、死区延迟结束时。与此类似，在出现通道(n+1)匹配（ ETM 计数器 = C(n+1)V）情况时，通道(n+1)输出将保持高值，直到通道(n+1)输出清除、死区延迟结束时。

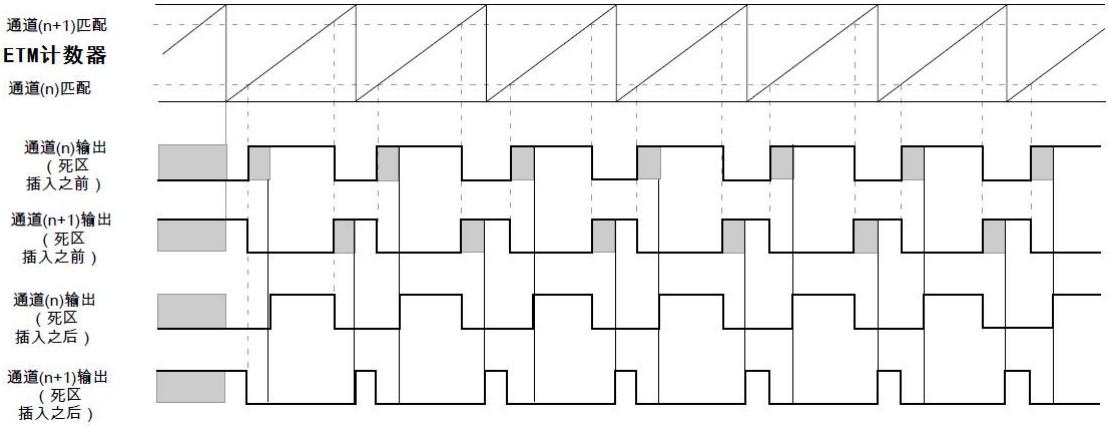


图 6-66 ELSnB:ELSnA=1:0、POL(n)=0 且 POL(n+1)=0 条件下的死区插入

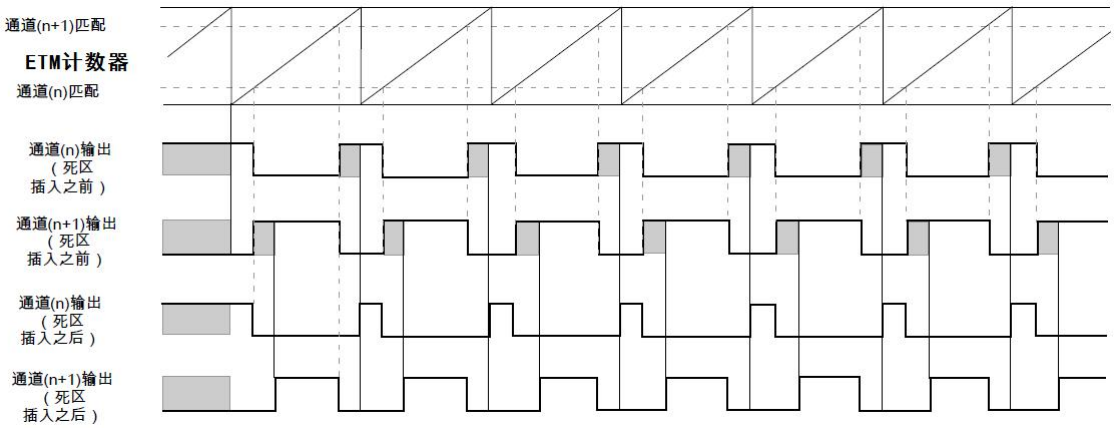


图 6-67 ELSnB:ELSnA=X:1、POL(n)=0 且 POL(n+1)=0 条件下的死区插入

注： 死区特性用于组合模式与互补模式。

❖ 6.3.8.14.1 死区插入个别案例

如果（ PS[2:0]已清零）、（ DTPS[1:0] = 0:0 或 DTPS[1:0] = 0:1）：

- 且死区延迟大于或等于通道(n)占空比（ (C(n+1)V－C(n)V)×系统时钟），则通道(n)输出始终为无效值（POL(n)位值）。
- 且死区延迟大于或等于通道(n+1)占空比（ (MOD－CNTIN + 1－(C(n+1)V－C(n)V))×系统时钟），则通道(n+1)输出始终为无效值（ POL(n+1)位值）。

尽管大多数情况下死区延迟无法与通道(n)和(n+1)占空比相比较，但下面这些图还是以示例说明了死区延迟可与占空比相比较的情况。

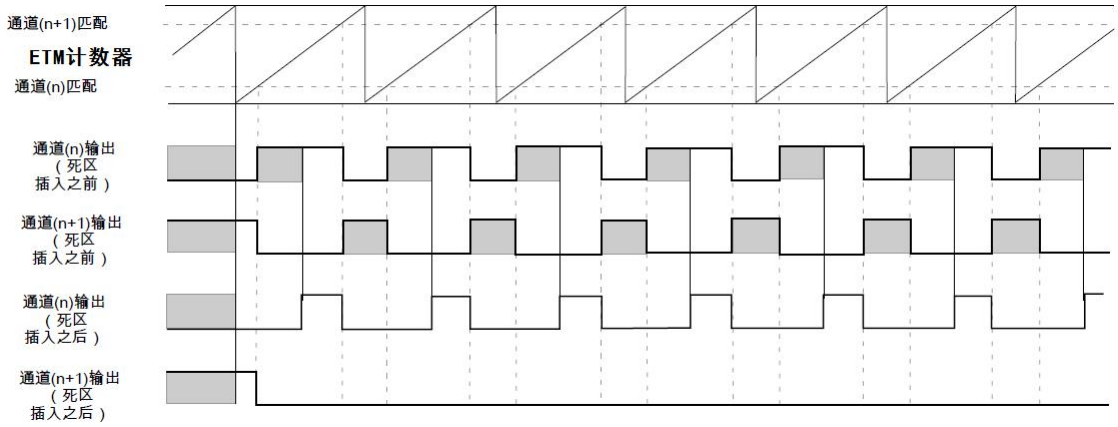


图 6-68 死区延迟可与通道(n+1)占空比相比较时的死区插入示例(ELSnB:ELSnA=1:0、POL(n)=0 且 POL(n+1)=0)

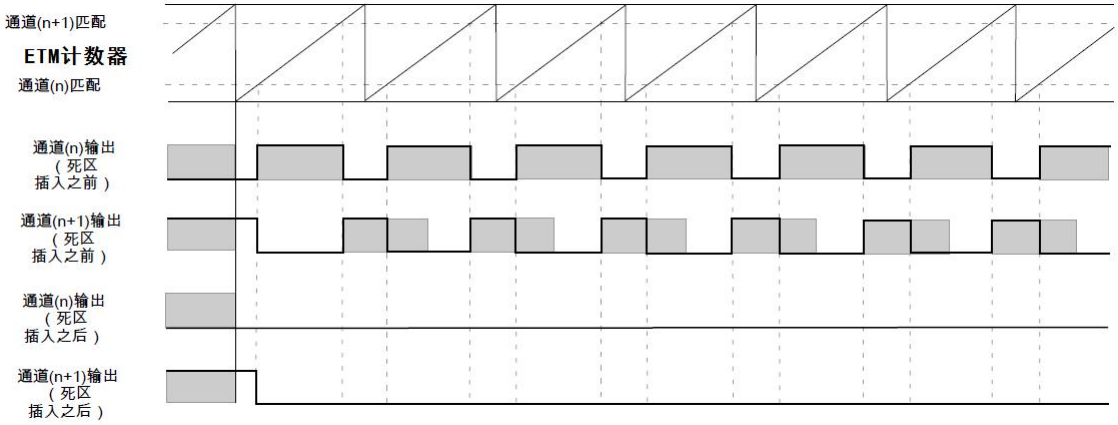


图 6-69 死区延迟可与通道 n 及(n+1)占空比相比较时的死区插入示例(ELSnB:ELSnA=1:0、POL(n)=0 且 POL(n+1)=0)

● 6.3.8.15 输出屏蔽

输出屏蔽可用于通过软件强制通道输出为各自的无效状态。例如，用于控制 BLDC 电机。

对 OUTMASK 寄存器采取的任意写入操作都会更新其写入缓冲区。OUTMASK 寄存器通过 PWM 同步以其缓冲区值进行更新；参见 OUTMASK 寄存器同步 。

如果 CHnOM = 1，则通道(n)输出强制为通道的无效状态（ POLn 位值）。如果 CHnOM = 0，则通道(n)输出不受输出屏蔽影响。参见下图：

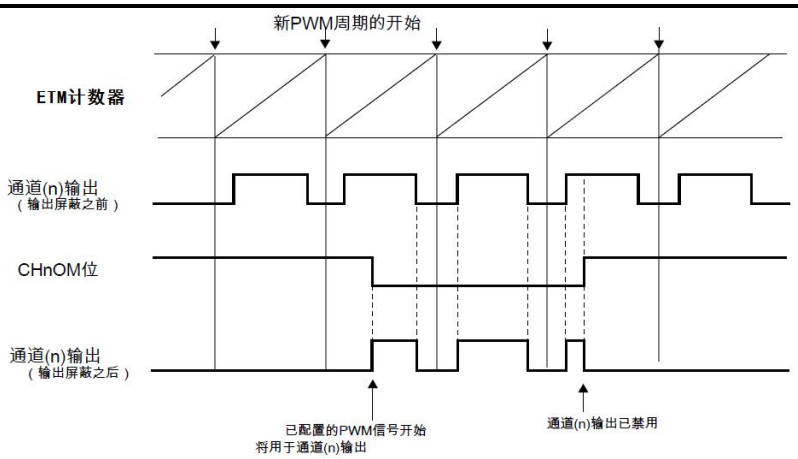


图 6-70 POLn=0 条件下的输出屏蔽

下表展示了极性控制之前的输出屏蔽结果。

表 6-47 极性控制之前的通道(n)输出屏蔽结果

CHnOM	输出屏蔽输入	输出屏蔽结果
0	无效状态	无效状态
	有效状态	有效状态
1	无效状态	无效状态
	有效状态	

注:输出屏蔽特性只能用于组合模式。

● 6.3.8.16 故障控制

如果(ETMEN = 1)且(FAULTM[1:0] ≠ 0:0)，则使能故障控制。

ETM 最多可拥有四个故障输入。FAULTnEN 位（其中 n = 0、 1、 2、 3）使能故障输入 n，FFLTRnEN 位使能故障输入 n 滤波器。FFVAL[3:0]位选择已使能的每个故障输入中处于使能状态的滤波器的值。

首先，每个故障输入信号由系统时钟同步；参见下图中的同步器时钟。同步之后，故障输入 n 信号进入滤波器模块。当故障输入 n 信号中发生状态更改时，5 位计数器将复位并开始向上计数。只要新状态在故障输入 n 中保持稳定，计数器就会继续增加数值。如果 5 位计数器溢出，即计数器超出 FFVAL[3:0]位的值，就表明新故障输入 n 值得以验证。然后就会作为脉冲边沿传输到边沿检测器。

如果在验证（计数器溢出）之前相反的边沿出现在故障输入 n 信号上，计数器就会复位。下一次输入转换时，计数器再次开始计数。任何比 FFVAL[3:0]位（x 系统时钟）所选的最小值短的脉冲都会被认作是毛刺，不会传递到边沿检测器上。

FFVAL[3:0]位为零或 FAULTnEN =0 时，将禁用故障输入 n 滤波器。这种情况下，故障输入 n 信号将按系统时钟的 2 个上升沿延迟，而且在故障输入 n 中出现上升沿之后 FAULTFn 位在系统时钟的第 3 个上升沿上置位。

如果 FFVAL[3:0] ≠ 0000 且 FAULTnEN=1，则故障输入 n 信号按系统时钟的(3 + FFVAL[3:0])个上升沿延迟，也就是在故障输入 n 中出现上升沿之后 FAULTFn 位在系统时钟的(4 + FFVAL[3:0])个上升沿上置位。

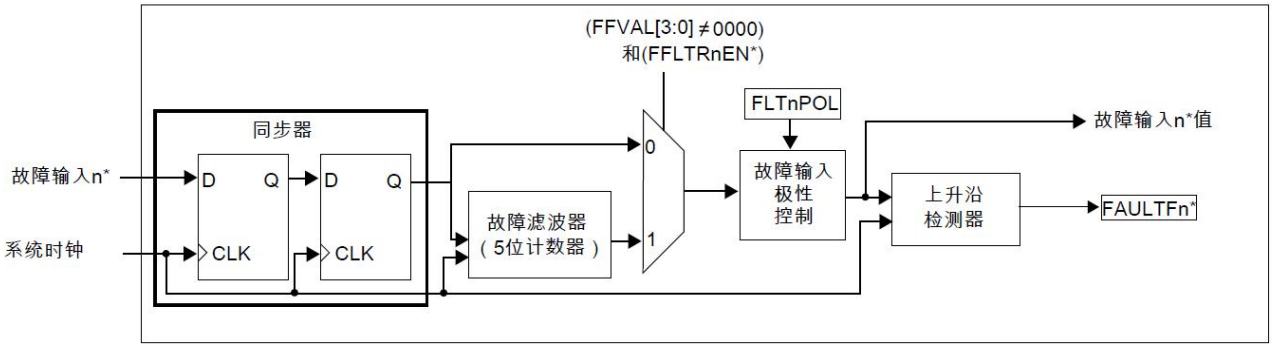


图 6-71 故障输入 n 控制功能框图

如果故障控制和故障输入 n 已使能，且在故障输入 n 信号上检测到上升沿，则表明已出现故障状况且 FAULTFn 位已置位。FAULTF 位是 FAULTFn[3:0]位的逻辑或(OR)。参见下图：

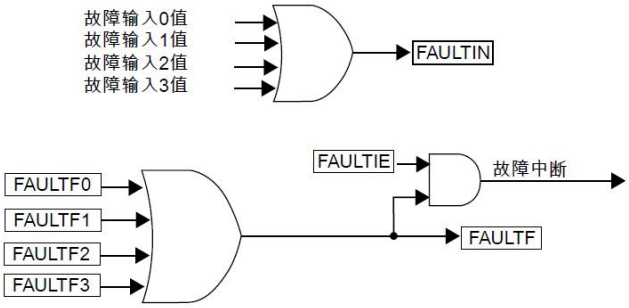


图 6-72 FAULTF 和 FAULTIN 位以及故障中断

如果已使能故障控制(FAULTM[1:0]≠0:0)、已出现故障状况且(FAULTEN=1)，则输出强制为各自的安全值：

- 通道(n)输出采用 POL(n)的值
- 通道(n+1)采用 POL(n+1)的值

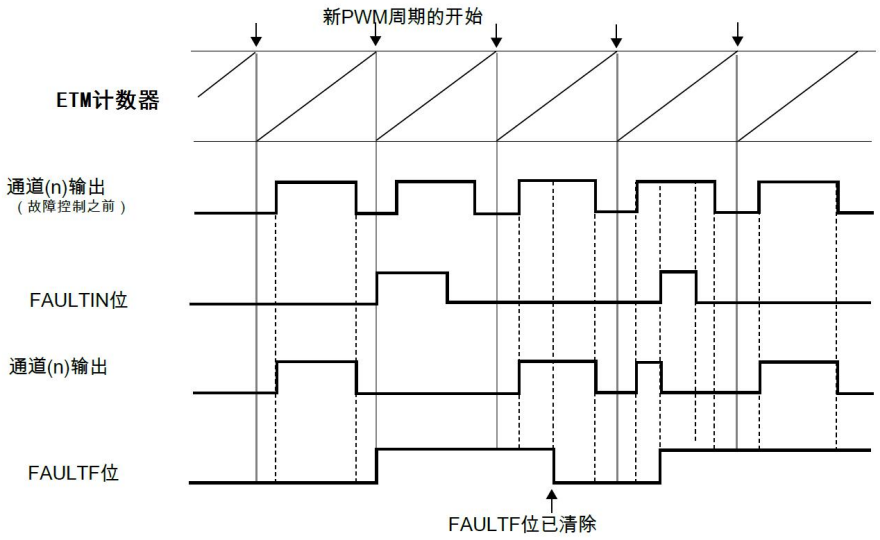
当(FAULTF = 1)且(FAULTIE = 1)时，生成故障中断。此中断请求一直保持置位状态，直到出现以下情形：

- 软件通过读取 FAULTF 位为 1 并向其写入 0 来清除 FAULTF 位
- 软件清除 FAULTIE 位
- 发生复位

注： 只能在组合模式下使用故障控制。

❖ 6.3.8.16.1 自动故障清除

如果选择了自动故障清除(FAULTM[1:0] = 1:1)，则当故障输入信号(FAULTIN)恢复为零，新的 PWM 周期开始时，被故障控制禁用的通道输出将再次进入使能状态。参见下图：

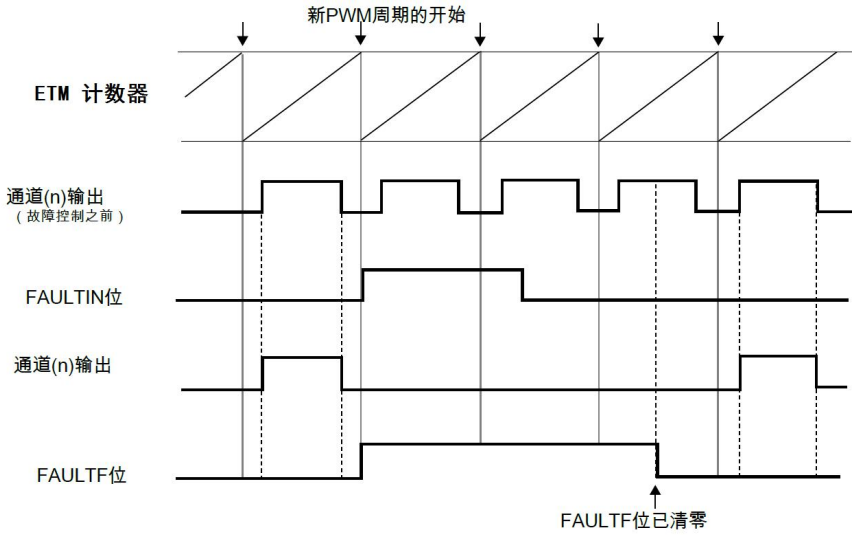


注释：通道(n)输出在故障控制之后，，手动清除故障，且 POLn = 0。

图 6-73 采用自动故障清除的故障控制

❖ 6.3.8.16.2 手动故障清除

如果选择了手动故障清除（ FAULTM[1:0] = 0:1 或 1:0），则当 FAULTF 位清除，新的 ETM 周期开始时，被故障控制禁用的通道输出将再次进入使能状态。参见下图：



注释：通道(n)输出在故障控制之后，，手动清除故障，且 POLn = 0。

图 6-74 采用手动故障清除的故障控制

❖ 6.3.8.16.3 故障输入极性控制

FLTjPOL 位选择故障输入 j 极性，其中 j = 0、 1、 2、 3：

- 如果 FLTjPOL = 0，则故障 j 输入极性高，因此故障输入 j 处的逻辑 1 表示故障。
- 如果 FLTjPOL = 1，则故障 j 输入极性低，因此故障输入 j 处的逻辑 0 表示故障。

● 6.3.8.17 极性控制

POLn 位选择通道(n)输出极性：

- 如果 POLn = 0，则通道(n)输出极性为高，因此逻辑 1 为有效状态，逻辑 0 为无效状态。
- 如果 POLn = 1，则通道(n)输出极性为低，因此逻辑 0 为有效状态，逻辑 1 为无效状态。

注： 只能在组合模式下使用极性控制。

● 6.3.8.18 初始化

向 INIT 位写入 1 时，初始化将强制通道(n)输出为 CHnOI 位的值。初始化取决于 COMP 和 DTEN 位。下表展示了 COMP 和 DTEN 位为零时初始化条件下通道(n)和(n+1)输出的值。

表 6-48 (COMP=0 且 DTEN=0)时初始化结果

CH(n) OI	CH(n+1)OI	通道(n)输出	通道(n+1) 输出
0	0	强制为 0	强制为 0
0	1	强制为 0	强制为 1
1	0	强制为 1	强制为 0
1	1	强制为 1	强制为 1

下表展示了(COMP=1)或(DTEN=1)时由初始化强制通道(n)和(n+1)输出值。

表 6-49 (COMP=1 或 DTEN=1)时初始化行为方式

CH(n) OI	CH(n+1)OI	通道(n)输出	通道(n+1) 输出
0	X	强制为 0	强制为 1
0	X	强制为 1	强制为 0

注：初始化特性只能在组合模式下与禁用的 ETM 计数器配合使用。参见状态和控制寄存器中关于 CLKS 字段的说明。

● 6.3.8.19 特殊优先级

下图展示了生成通道(n)和(n+1)输出信号时所用特性的优先级。

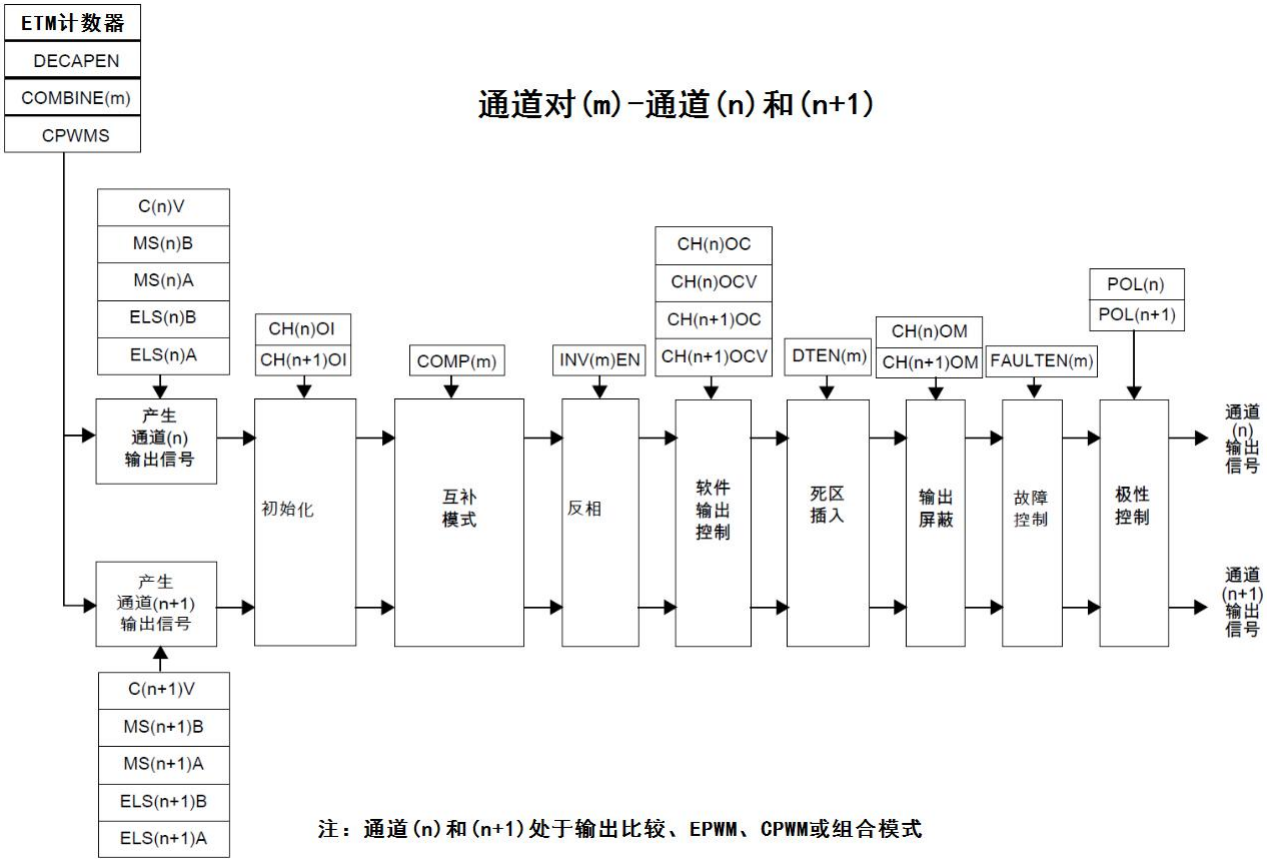


图 6-75 生成通道(n)和(n+1)输出信号时所用特性的优先级

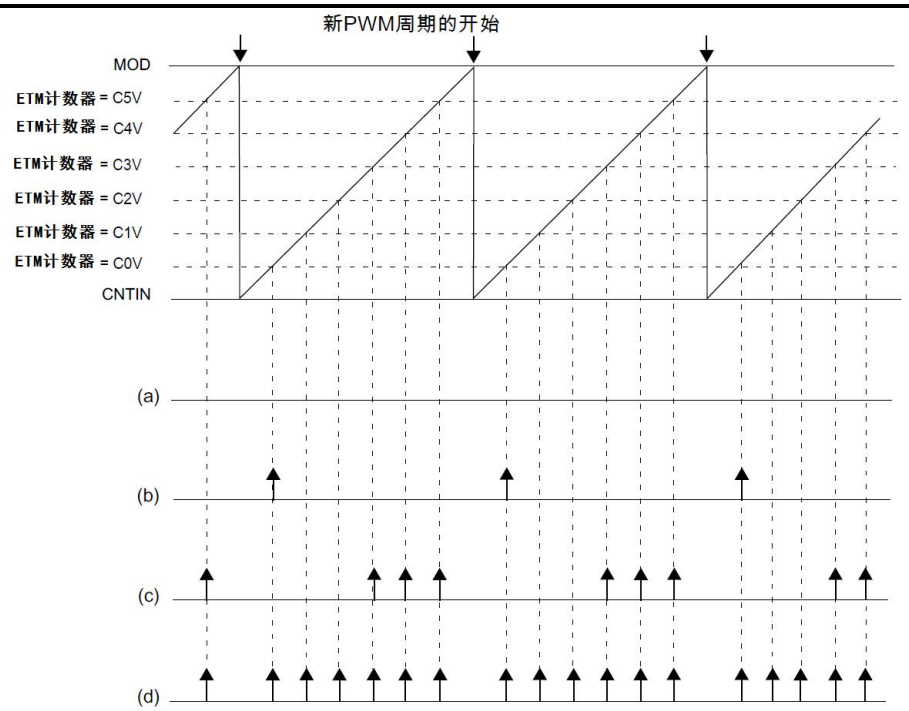
注 初始化特性不得与反相和软件输出控制特性一起使用。

● 6.3.8.20 通道触发器输出

如果 CHjTRIG = 1（其中 j = 0、 1、 2、 3、 4 或 5），则当通道(j)发生匹配事件（ETM 计数器=C(j)V），ETM 会产生触发信号。

通道触发器输出提供了一种可用于片上模块的触发信号。

ETM 能够在一个 PWM 周期内生成多个触发器。因为每个触发信号都是由一个特定通道生成的，所以要实现此功能需要有多个通道。下图描述了这种行为方式。



注释

(a) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0

(b) CH0TRIG = 1, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 0, CH4TRIG = 0, CH5TRIG = 0

(c) CH0TRIG = 0, CH1TRIG = 0, CH2TRIG = 0, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1

(d) CH0TRIG = 1, CH1TRIG = 1, CH2TRIG = 1, CH3TRIG = 1, CH4TRIG = 1, CH5TRIG = 1

图 6-76 通道匹配触发器

注 通道匹配触发器只能用于组合模式。

● 6.3.8.21 初始化触发

如果 INITTRIGEN = 1，那么在以下情形中 ETM 计数器更新为 CNTIN 寄存器值时，ETM 将生成触发。

- ETM 计数器通过所选计数模式自动更新为 CNTIN 寄存器值。
- 对 CNT 寄存器采取写入操作时。
- 存在 ETM 计数器同步时。
- 如果(CNT = CNTIN)、(CLKS[1:0] = 0:0)，并且向 CLKS[1:0]位写入了不为零的值。

以下各图展示了这些情形。

CNTIN=0x0000,MOD=0x000F,CPWMS=0

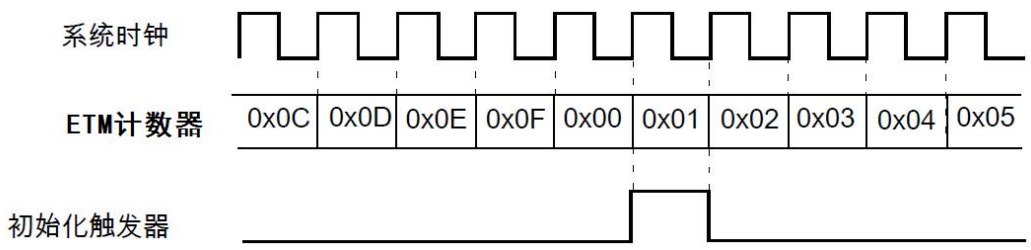


图 6-77 ETM 计数达到 CNTIN 寄存器值时生成初始化触发

CNTIN=0x0000,MOD=0x000F,CPWMS=0

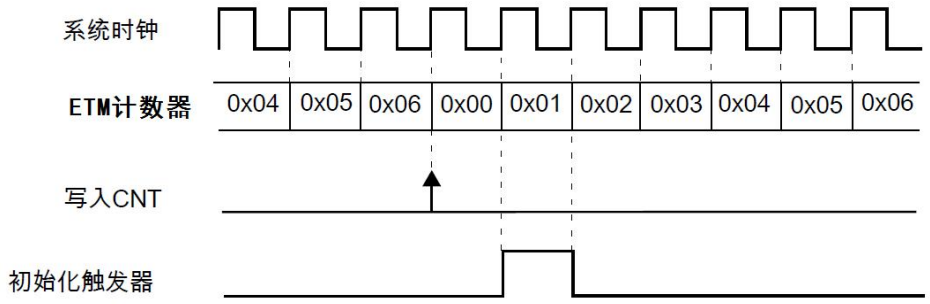


图 6-78 对 CNT 寄存器采取写入操作时生成初始化触发

CNTIN=0x0000,MOD=0x000F,CPWMS=0

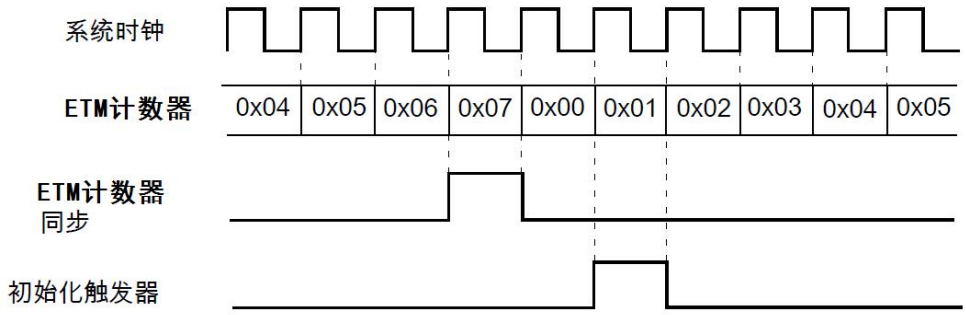


图 6-79 存在 ETM 计数器同步时生成初始化触发

CNTIN=0x0000,MOD=0x000F,CPWMS=0

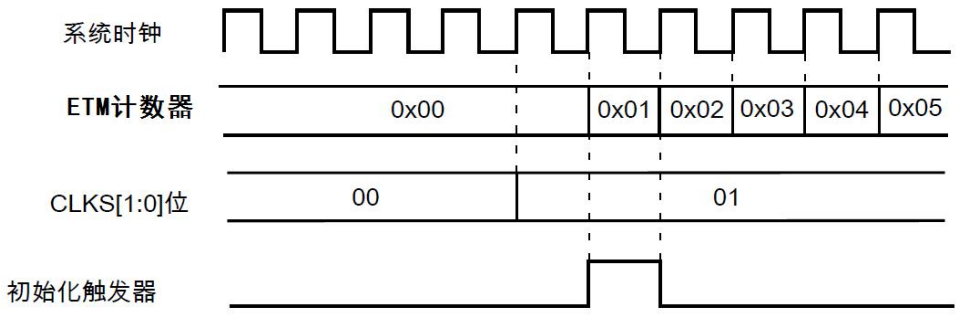


图 6-80 (CNT=CNTIN)、(CLKS[1:0]=0:0)且向 CLKS[1:0]位写入了不为零值时生成初始化触发

初始化触发输出提供了一种可用于片上模块的触发信号。

注:初始化触发只能用于组合模式。

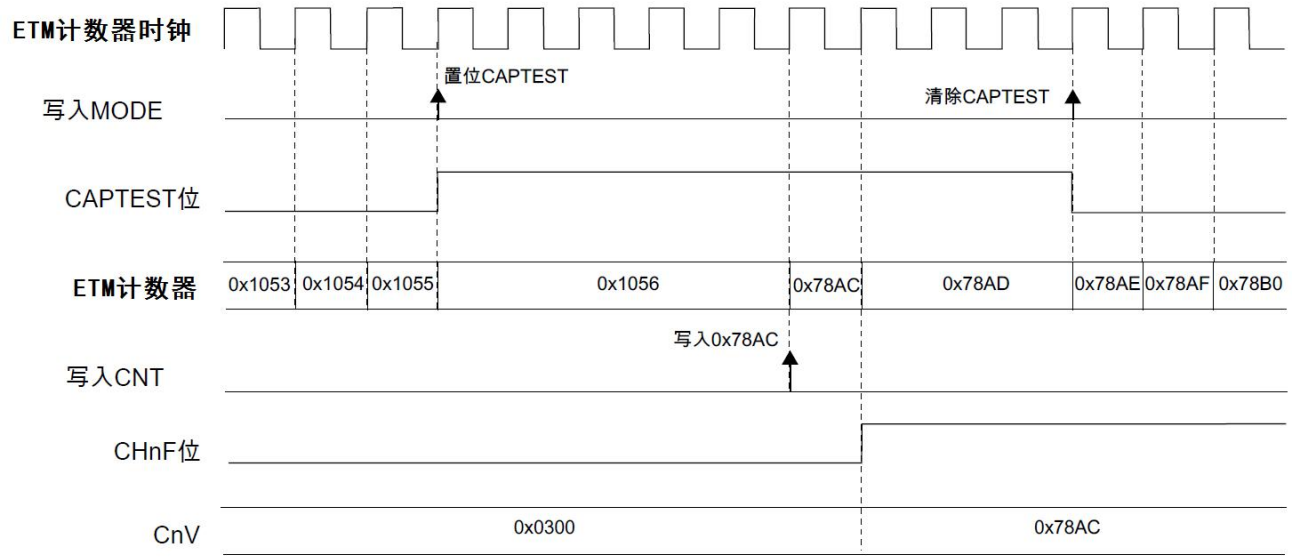
● 6.3.8.22 捕获测试模式

捕获测试模式允许测试 CnV 寄存器、 ETM 计数器以及 ETM 计数器与 CnV 寄存器之间的互连逻辑。

在这种测试模式下，必须为输入捕捉模式 配置所有通道，且 ETM 计数器必须配置为向上计数 。

使能捕获测试模式(CAPTEST = 1)后，ETM 计数器将冻结，对 CNT 寄存器进行的任何写入操作都会直接更新 ETM 计数器；参见下图：写入后，所有 CnV 寄存器都会更新为 CNT 寄存器中的值，同时 CHnF 位会置位。因此，ETM 计数器会根据其配置更新其下一个值。其下一个值取决于 CNTIN、MOD 以及写入 ETM 计数器的值。

接下来的 CnV 寄存器读取操作会返回写入 ETM 计数器的值，接下来的 CNT 寄存器读取操作则返回 ETM 计数器的下一个值。



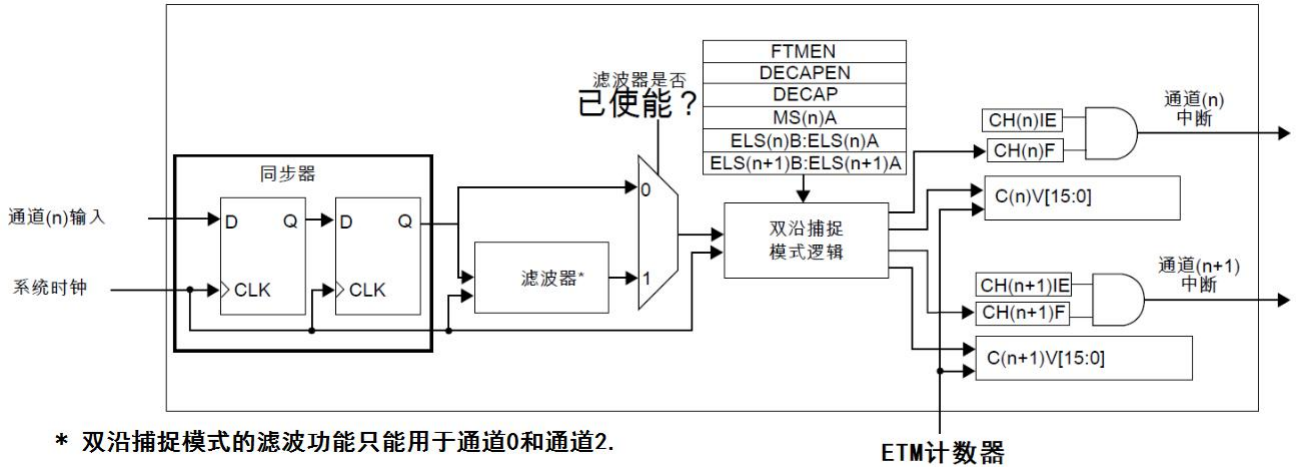
注：

- ETM 计数器配置：(ETMEN = 1)，(CAPTEST = 1)，(CPWMS = 0)，(CNTIN = 0x0000)，(MOD = 0xFFFF)
- ETM 通道 n 配置：输入捕捉模式 - (DECAPEN = 0)，(COMBINE = 0)，(MSnB:MSnA = 0:0)

图 6-81 捕捉测试模式

● 6.3.8.23 双沿捕获模式

如果 ETMEN = 1 且 DECAPEN = 1，将选择双沿捕捉模式。此模式允许在一对通道中通道(n)的输入端测量脉宽或信号周期。在此模式下，当 n 为 0 或 2 时，通道(n)滤波器可处于有效状态。



*** 双沿捕捉模式的滤波功能只能用于通道0和通道2.**

图 6-82 双沿捕捉模式结构框图

MS(n)A 位定义双沿捕捉模式是单次还是持续。

ELS(n)B:ELS(n)A 位选择由通道(n)捕捉的边沿，ELS(n+1)B:ELS(n+1)A 位选择由通道(n+1)捕捉的边沿。如果 ELS(n)B:ELS(n)A 和 ELS(n+1)B:ELS(n+1)A 位选择同一边沿，则为周期测量。如果这两个位选择不同边沿，则为脉宽测量。

在双沿捕捉模式下，只使用通道(n)输入，忽略通道(n+1)输入。

如果在通道(n)输入端检测到由通道(n)位定义的边沿，则 CH(n)F 位置位并生成通道(n)中断（条件是 CH(n)IE = 1）。如果在通道(n)输入端检测到由通道(n+1)位定义的边沿且(CH(n)F = 1)，则 CH(n+1)F 位

置位并生成通道(n+1)中断（条件是 CH(n+1)IE = 1）。

C(n)V 寄存器存储了在通道(n)输入端检测到通道(n)所选边沿时 ETM 计数器的值。

C(n+1)V 寄存器存储了在通道(n)输入端检测到通道(n+1)所选边沿时 ETM 计数器的值。

在此模式下，有一个一致性机制可确保读取 C(n)V 和 C(n+1)V 寄存器时保持一致的数据。这个唯一的要求就是必须在 C(n+1)V 之前读取 C(n)V。

注:

- CH(n)F、 CH(n)IE、 MS(n)A、 ELS(n)B 和 ELS(n)A 位是与通道(n)有关的位。
- CH(n+1)F、 CH(n+1)IE、 MS(n+1)A、 ELS(n+1)B 和 ELS(n+1)A 位是通道(n+1)位。
- 必须在 ELS(n)B:ELS(n)A = 0:1 或 1:0、ELS(n+1)B:ELS(n+1)A = 0:1 或 1:0 且 ETM 计数器为自由运行计数器情况下使用双沿捕捉模式。

❖ 6.3.8.23.1 单次捕捉模式

当(ETMEN = 1)、 (DECAPEN = 1)且(MS(n)A = 0)时选择单次捕捉模式。在这种捕捉模式下，只捕捉通道(n)输入上的一对边沿。ELS(n)B:ELS(n)A 位选择要捕捉的第一边沿，ELS(n+1)B:ELS(n+1)A 位选择要捕捉的第二边沿。

DECAP 位置位时将使能边沿捕捉。对于单次捕捉模式下的每一次新测量，必须首先清除 CH(n)F 和 CH(n+1)位，然后 DECAP 位必须置位。

在此模式下，将在捕捉通道(n+1)所选的边沿后由 ETM 自动清除 DECAP 位。因此，DECAP 位置位后，就会开始单次捕捉。清除此位后，两个边沿都会捕捉，捕捉的值也可用于在 C(n)V 和 C(n+1)V 寄存器中读取。

与此类似，CH(n+1)F 位置位后，两个边沿都会捕捉，捕捉的值也可用于在 C(n)V 和 C(n+1)V 寄存器中读取。

❖ 6.3.8.23.2 持续捕捉模式

当(ETMEN = 1)、 (DECAPEN = 1)且(MS(n)A = 1)时选择持续捕捉模式。在这种捕捉模式下，将持续捕捉通道(n)输入上的边沿。ELS(n)B:ELS(n)A 位选择要捕捉的初始沿，ELS(n+1)B:ELS(n+1)A 位选择要捕捉的最终沿。

DECAP 位置位时将使能边沿捕捉。初次使用时，首先必须清空 CH(n)F 和 CH(n+1)F 位，然后必须使 DECAP 位置位，以便开始持续测量。

CH(n+1)F 位置位后，两个边沿都会被捕捉，捕捉的值也可在 C(n)V 和 C(n+1)V 寄存器中读取。即便已清空 DECAP 位，最新捕捉到的值也仍然可用于这些寄存器。

在此模式下，可以只清空 CH(n+1)F 位。因此，CH(n+1)F 位再次置位后，最新捕捉到的值将可用在 C(n)V 和 C(n+1)V 寄存器中。

对于双沿捕捉－持续模式下的一系列新测量，清空 CH(n)F 和 CH(n+1)F 位便可开始新的测量。

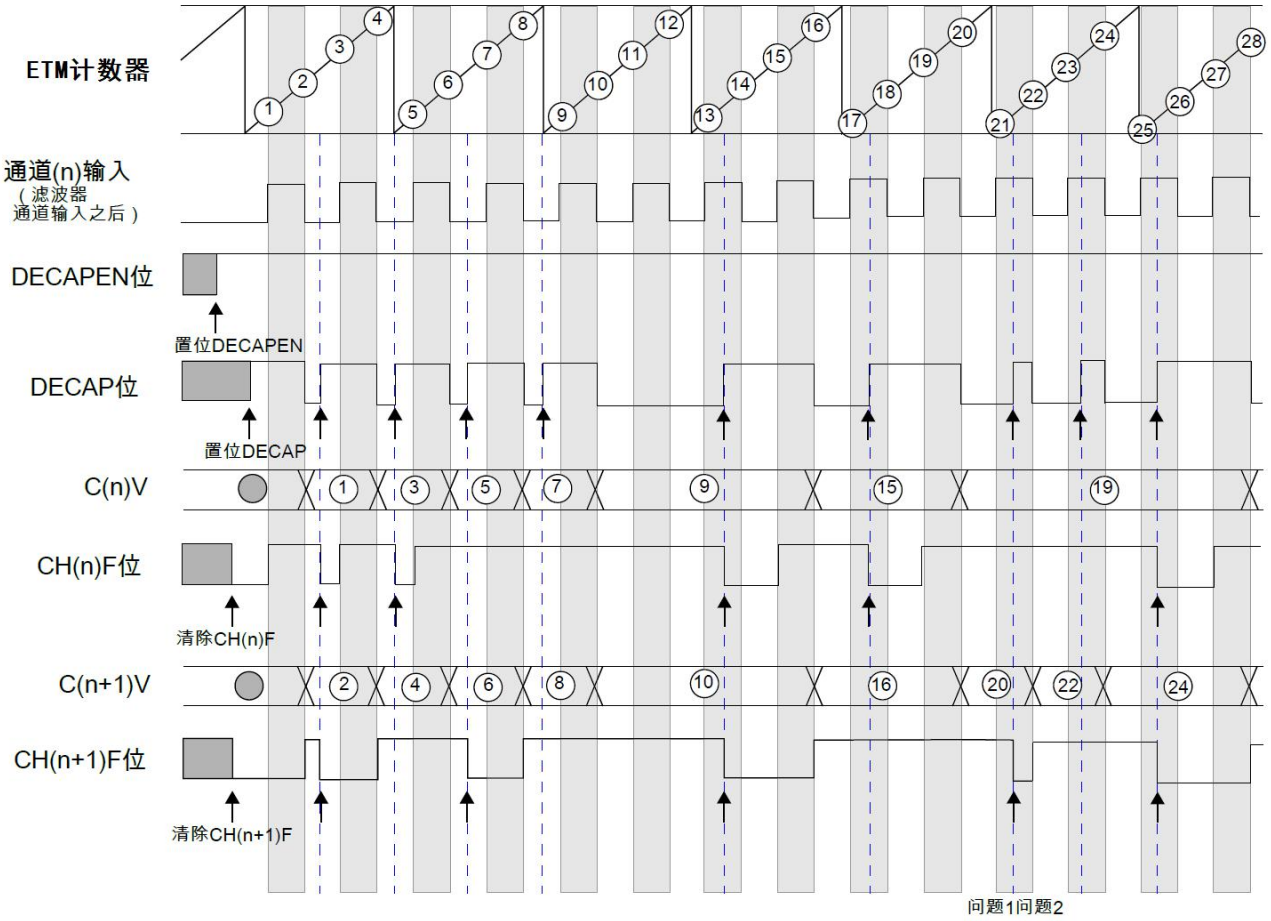
❖ 6.3.8.23.3 脉宽测量

如果通道(n)配置为捕捉上升沿(ELS(n)B:ELS(n)A = 0:1)，通道(n+1)配置为捕捉下降沿(ELS(n+1)B:ELS(n+1)A = 1:0)，则测量正极性脉宽。如果通道(n)配置为捕捉下降沿(ELS(n)B:ELS(n)A = 1:0)，通道(n+1)配置为捕捉上升沿(ELS(n+1)B:ELS(n+1)A = 0:1)，则测量负极性脉宽。

可在单次捕捉模式 或持续捕捉模式下进行脉宽测量。

下图给出了双沿捕捉 － 单次模式的示例，用于测量正极性脉宽。 DECAPEN 位选择双沿捕捉模式，因此保持置位。DECAP 位置位，以使能下一个正极性脉宽的测量。检测到此脉冲的第一个边沿（也就是由 ELS(n)B:ELS(n)A 位选择的边沿）时，将置位 CH(n)F 位。检测到此脉冲的第二个边沿（也就是由 ELS(n+1)B:ELS(n+1)A 位选择的边沿）时，将置位 CH(n+1)F 位并清除 DECAP 位。DECAP 和 CH(n+1)F 位指出

何时捕捉到脉冲的两个边沿，以及 C(n)V 和 C(n+1)V 寄存器何时准备好进行读取。

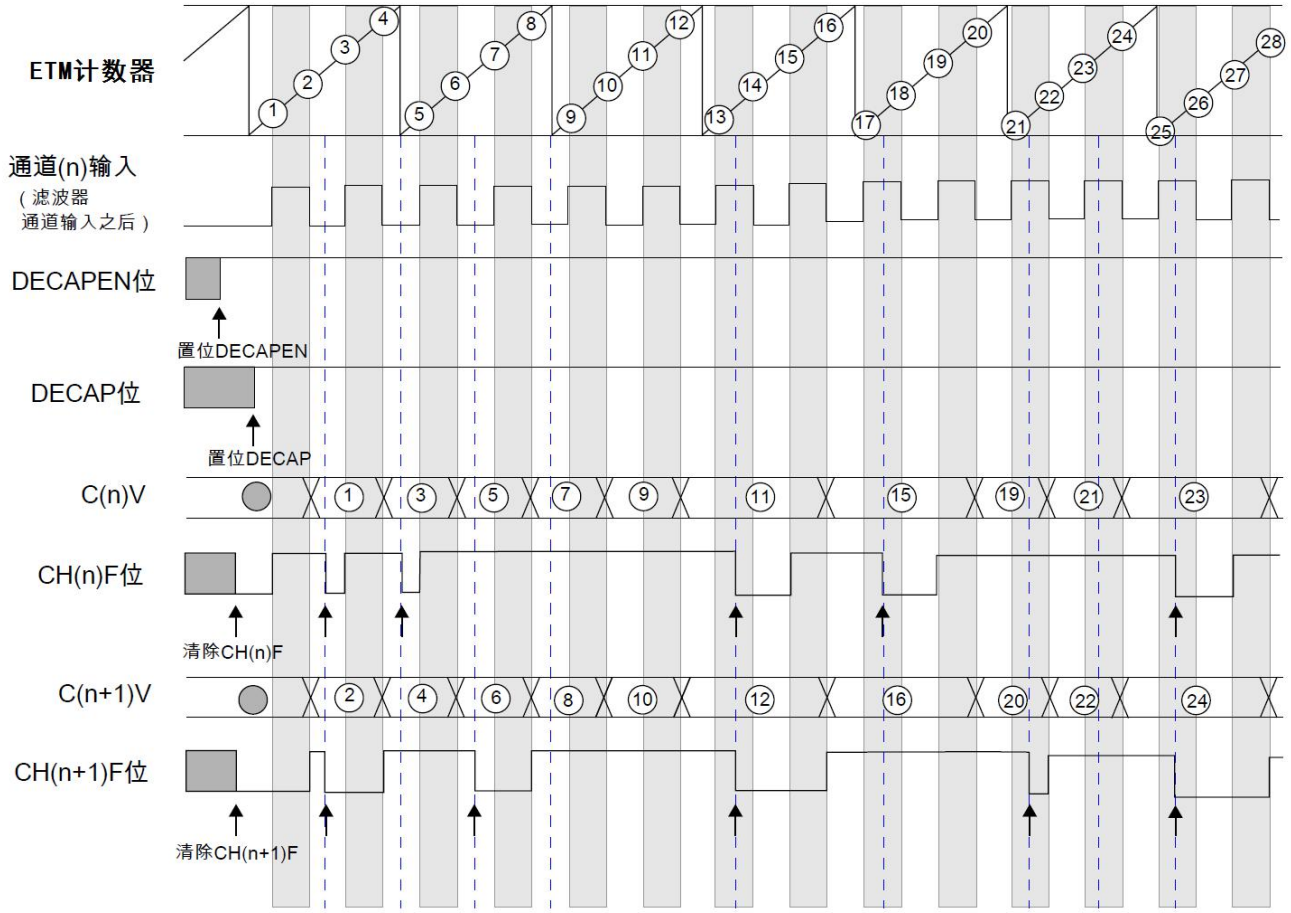


注释 置位 DECAPEN、置位 DECAP、清除 CH(n)F 和清除 CH(n+1)这些命令均由用户发出。

- 问题 1：通道(n)输入 = 1，置位 DECAP，不清除 CH(n)F，清除 CH(n+1)F。
- 问题 2：通道(n)输入 = 1，置位 DECAP，不清除 CH(n)F，不清除 CH(n+1)F。

图 6-83 用于正极性脉宽测量的双沿捕捉-单次模式

下图给出了双沿捕捉 - 持续模式的示例，用于测量正极性脉宽。 DECAPEN 位选择双沿捕捉模式，因此保持置位。DECAP 位置位时，将进行已配置的测量。检测到正极性脉冲的第一个边沿（也就是由 ELS(n)B:ELS(n)A 位选择的边沿）时，将置位 CH(n)F 位。检测到此脉冲的第二个边沿（也就是由 ELS(n+1)B:ELS(n+1)A 位选择的边沿）时，将置位 CH(n+1)F 位。CH(n+1)F 位指出何时捕捉到脉冲的两个边沿，以及 C(n)V 和 C(n+1)V 寄存器何时准备好进行读取。



注释 置位 DECAPEN、置位 DECAP、清除 CH(n)F 和清除 CH(n+1)这些命令均由用户发出。

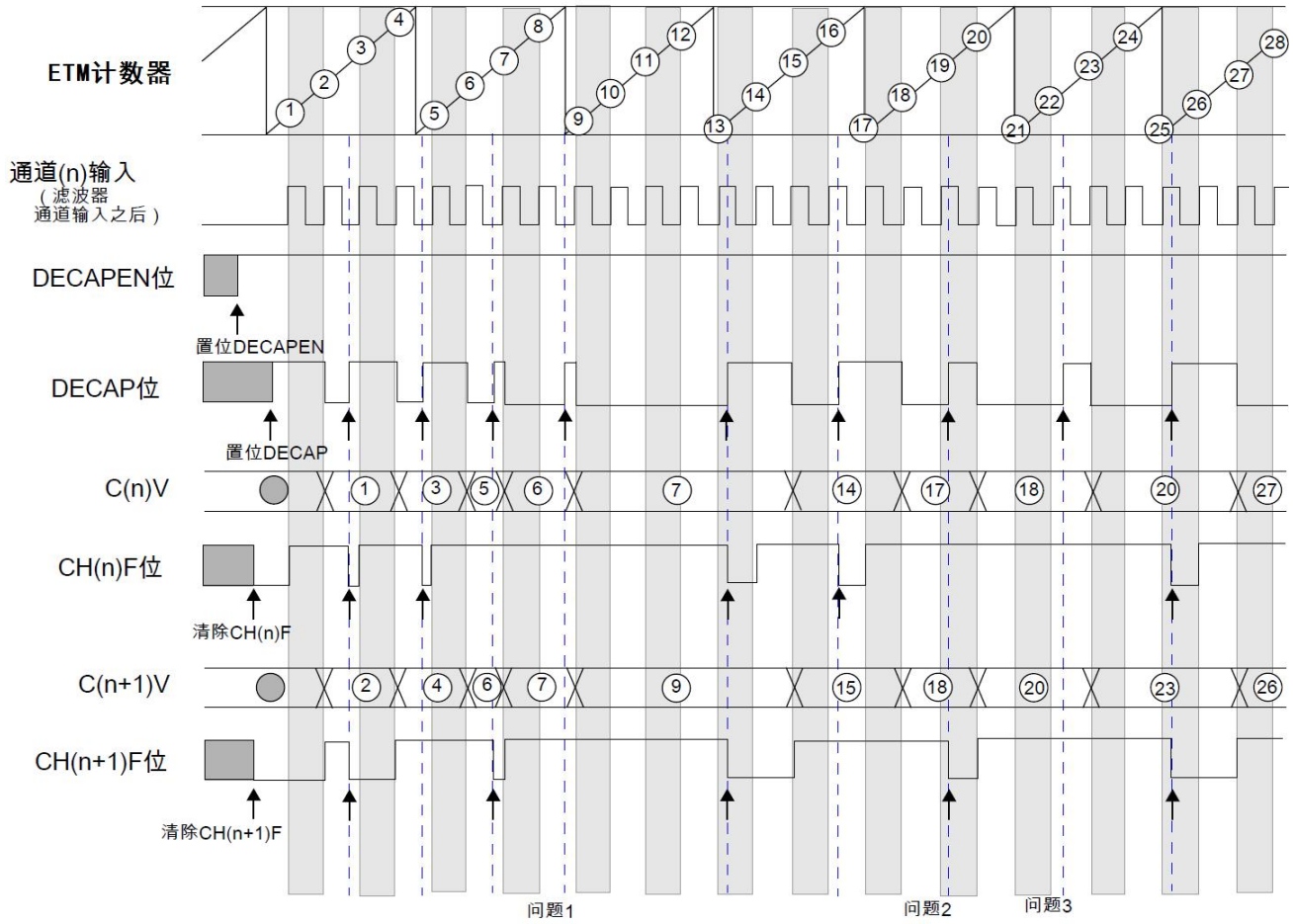
图 6-84 用于正极性脉宽测量的双沿捕捉-持续模式

❖ 6.3.8.23.4 周期测量

如果通道(n)和(n+1)配置为捕捉相同极性的连续边沿，则测量通道(n)输入信号的周期。如果通道(n)和(n+1)配置为捕捉上升沿（ ELS(n)B:ELS(n)A = 0:1 且 ELS(n+1)B:ELS(n+1)A = 0:1），则测量两个连续上升沿之间的周期。如果通道(n)和(n+1)配置为捕捉下降沿（ ELS(n)B:ELS(n)A = 1:0 且 ELS(n+1)B:ELS(n+1)A = 1:0），则测量两个连续下降沿之间的周期。

可在单次捕捉模式 或持续捕捉模式下进行周期测量。

下图给出了双沿捕捉－单次模式的示例，用于测量两个连续上升沿之间的周期。DECAPEN 位选择双沿捕捉模式，因此保持置位。DECAP 位置位，以使能下一周期的测量。检测到第一个上升沿（也就是由 ELS(n)B:ELS(n)A 位选择的边沿）时，将置位 CH(n)F 位。检测到第二个上升沿（也就是由 ELS(n+1)B:ELS(n+1)A 位选择的边沿）时，将置位 CH(n+1)F 位并清空 DECAP 位。DECAP 和 CH(n+1)F 位指出何时捕捉到所选的两个边沿，以及 C(n)V 和 C(n+1)V 寄存器何时准备好进行读取。

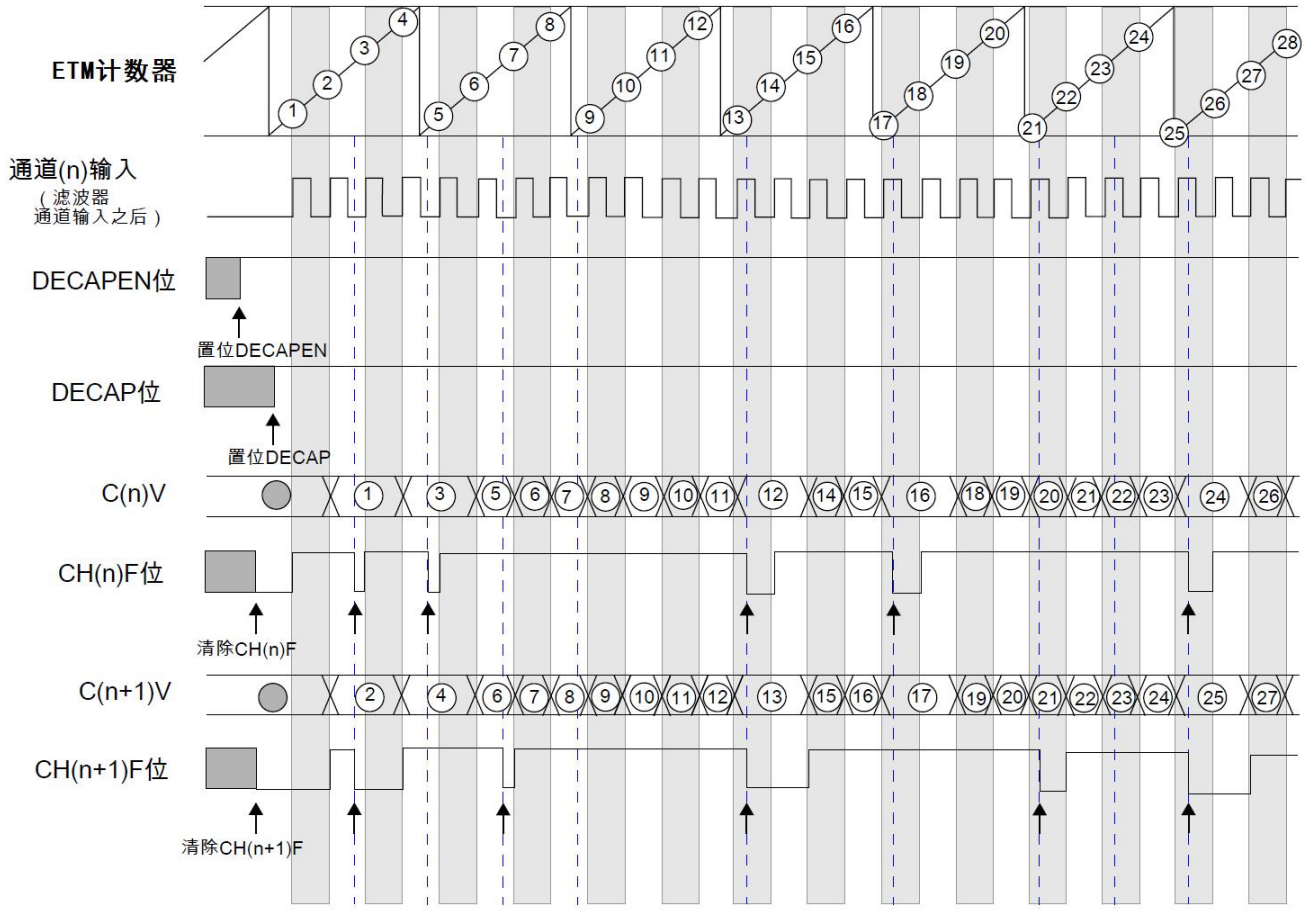


注释- 置位 DECAPEN、置位 DECAP、清除 CH(n)F 和清除 CH(n+1)这些命令均由用户发出。

- 问题 1：通道(n)输入 = 0 ，置位 DECAP，不清除 CH(n)F，不清除 CH(n+1)F。
- 问题 2：通道(n)输入 = 1，置位 DECAP，不清除 CH(n)F，清除 CH(n+1)F。
- 问题 3：通道(n)输入 = 1，置位 DECAP，不清除 CH(n)F，不清除 CH(n+1)F。

图 6-85 用于测量两个连续上升沿之间周期的双沿捕捉 - 单次模式

下图给出了双沿捕捉 - 持续模式的示例，用于测量两个连续上升沿之间的周期。DECAPEN 位选择双沿捕捉模式，因此保持置位。DECAP 位置位时，将进行已配置的测量。检测到第一个上升沿（也就是由 ELS(n)B:ELS(n)A 位选择的边沿）时，将置位 CH(n)F 位。检测到第二个上升沿（也就是由 ELS(n+1)B:ELS(n+1)A 位选择的边沿）时，将置位 CH(n+1)F 位。CH(n+1)F 位指出何时捕捉到周期的两个边沿，以及 C(n)V 和 C(n+1)V 寄存器何时准备好进行读取。



注释- 置位 DECAPEN、置位 DECAP、清除 CH(n)F 和清除 CH(n+1)这些命令均由用户发出。

图 6-86 用于测量两个连续上升沿之间周期的双沿捕捉-持续模式

❖ 6.3.8.23.5 读出一致性机制

对于在 C(n)V 和 C(n+1)V 寄存器中捕捉的 ETM 计数器值，双沿捕捉模式会实施一种读取一致性机制。下图对读取一致性机制进行了说明。此例中，通道(n)和(n+1)处于双沿捕捉 - 持续模式，以便进行正极脉宽测量。因此，通道(n)配置为在通道(n)输入信号中存在上升沿时捕捉 ETM 计数器值，通道(n+1)配置为在通道(n)输入信号中存在下降沿时捕捉 ETM 计数器值。

通道(n)输入信号中存在上升沿时，ETM 计数器值将捕捉到通道(n)捕捉缓存内。通道(n)输入信号中存在下降沿时，通道(n)捕捉缓存值将传输到 C(n)V 寄存器中。上一个上升沿发生时， C(n)V 寄存器拥有 ETM 计数器值；最后一个上升沿发生时，通道(n)捕捉缓存拥有 ETM 计数器值。

通道(n)输入信号中存在下降沿时， ETM 计数器值将捕捉到通道(n+1)捕捉缓存内。读取 C(n)V 寄存器后，通道(n+1)捕捉缓存值将传输到 C(n+1)V 寄存器中。

下图中，发生事件 1 时读取 C(n)V 将返回 ETM 计数器值，发生事件 2 时读取 C(n+1)V 将返回 ETM 计数器值。

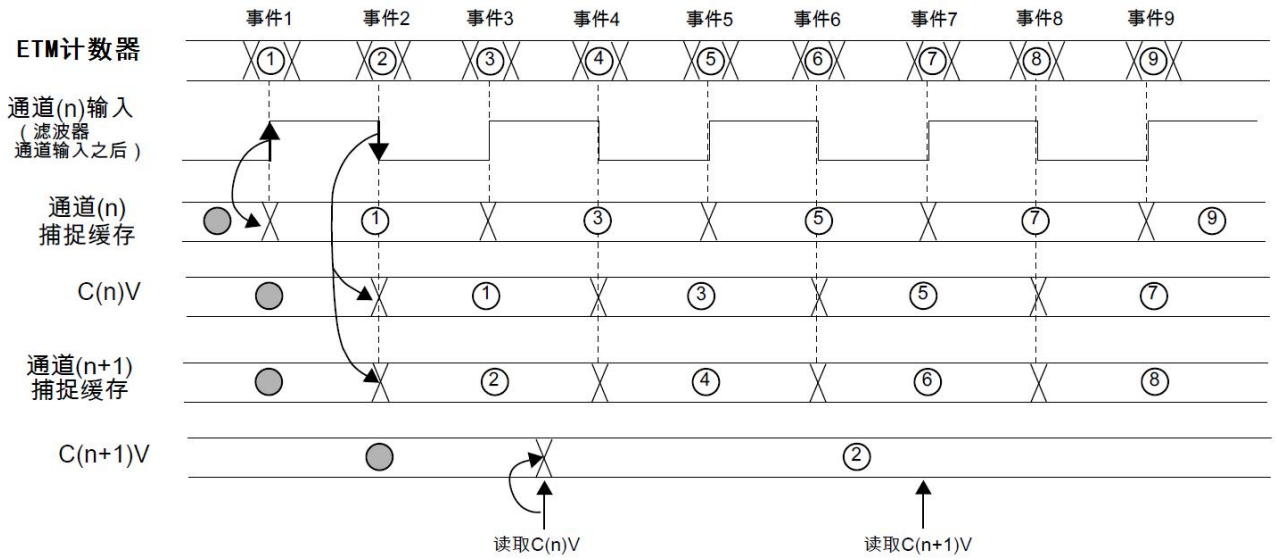


图 6-87 双沿捕捉模式读取一致性机制

要让读取一致性机制正常工作，在单次双沿捕捉和持续双沿捕捉模式下，必须先读取 C(n)V 寄存器，然后再读取 C(n+1)V 寄存器。

● 6.3.8.24 调试模式

芯片处于调试模式时，BDMMODE[1:0]位将根据下表选择 ETM 计数器、CH(n)F 位、通道输出以及对 MOD、CNTIN 和 C(n)V 寄存器的写入操作的行为方式。

表 6-50 芯片处于调试模式时的 ETM 行为方式

BDMMODE	ETM 计数器	CH(n)F 位	ETM 通道输出	对 MOD、CNTIN 和 C(n)V 寄存器的写入操作
00	已停止	可以设置	正常工作模式	对这些寄存器的写入操作会绕过寄存器缓冲区
01	已停止	未设置	通道输出会根据 POLn 位强制采取各自的安全值	对这些寄存器的写入操作会绕过寄存器缓冲区
10	已停止	未设置	芯片进入调试模式时，通道输出会被冻结	对这些寄存器的写入操作会绕过寄存器缓冲区
11	正常工作模式	可以设置	正常工作模式	正常工作模式

如果 BDMMODE[1:0] = 2’b00，通道输出将保持芯片进入调试模式时的值，因为 ETM 计数器已停止。但是，以下情形会在这种调试模式下修改通道输出。

• 向 CNT 寄存器写入任意值；参见计数器复位。这种情况下，ETM 计数器会根据 CNTIN 寄存器值更新，并且通道输出更新为初始值 – 设置为输出比较模式的通道则例外。

• ETM 计数器由 PWM 同步模式复位；参见 ETM 计数器同步 。这种情况下，ETM 计数器会根据 CNTIN 寄存器值更新，并且通道输出更新为初始值 – 处于输出比较模式的通道则例外。

• 在通道输出初始化过程中，向 INIT 位写入值 1 时，通道(n)输出将强制为 CH(n)OI 位值。参见初始化 。

注 BDMMODE[1:0] = 2’b00 不得与故障控制一起使用。即便已使能故障控制且存在故障条件，通道输出值也仍然会按上文所述更新。

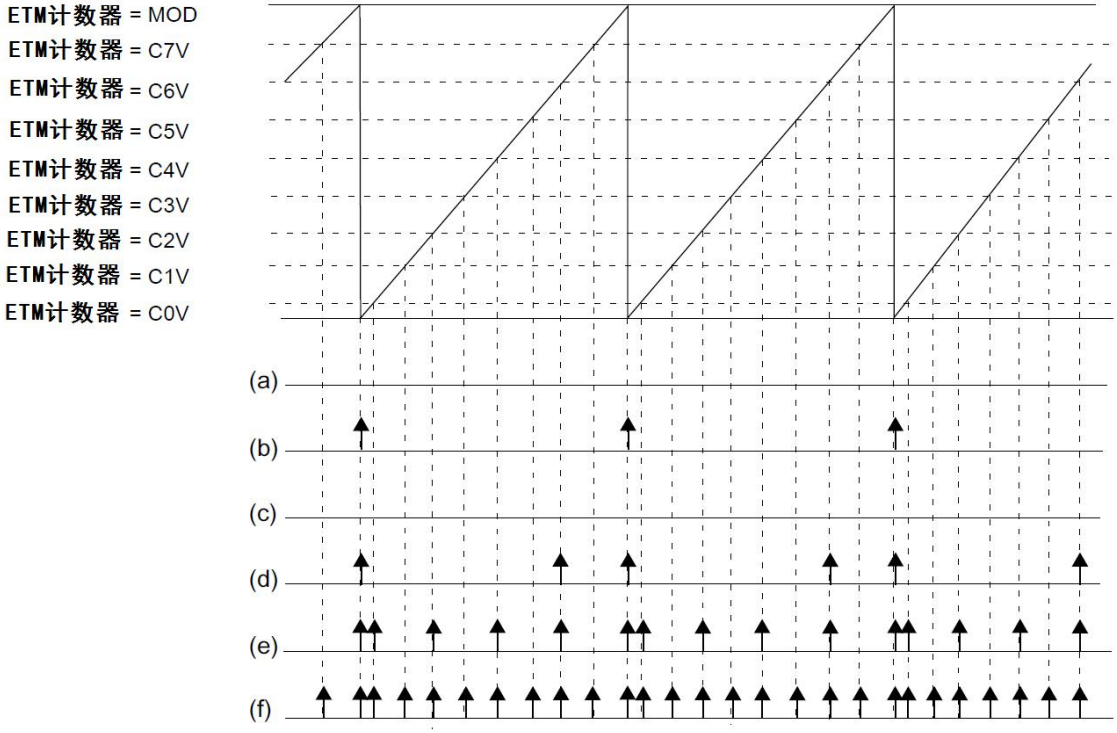
● 6.3.8.25 中间加载

PWMLOAD 寄存器允许在定义的重载点根据寄存器缓冲区的内容更新 MOD、CNTIN 和 C(n)V 寄存器。这种情况下，不需要使用 PWM 同步。对于中间加载，有多种可能的加载点：

表 6-51 使能可能的加载点时间

加载点	使能
ETM 计数器从 MOD 值变为 CNTIN 值时	始终
通道(j)匹配(ETM 计数器=C(j)V)时	CHjSEL=1 时

下图给出了一些有关已使能加载点的示例。



注释

(a) LDOK = 0, CHOSEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0

(b) LDOK = 1, CHOSEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0

(c) LDOK = 0, CHOSEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 1, CH4SEL = 0, CH5SEL = 0, CH6SEL = 0, CH7SEL = 0

(d) LDOK = 1, CHOSEL = 0, CH1SEL = 0, CH2SEL = 0, CH3SEL = 0, CH4SEL = 0, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0

(e) LDOK = 1, CHOSEL = 1, CH1SEL = 0, CH2SEL = 1, CH3SEL = 0, CH4SEL = 1, CH5SEL = 0, CH6SEL = 1, CH7SEL = 0

(f) LDOK = 1, CHOSEL = 1, CH1SEL = 1, CH2SEL = 1, CH3SEL = 1, CH4SEL = 1, CH5SEL = 1, CH6SEL = 1, CH7SEL = 1

图 6-88 中间加载的加载点

使能加载点之后，必须对要发生的加载设置 LDOK 位。这种情况下，加载将根据以下条件在下一个使能的加载点发生：

表 6-52 让加载在下一个使能的加载点发生的条件

向以下寄存器写入了新的值时	结果
MOD 寄存器	MOD 寄存器将以其写入缓冲区值更新。
CNTIN 寄存器，且 CNTINC = 1	CNTIN 寄存器将以其写入缓冲区值更新。

C(n)V 寄存器，且 SYNCENm = 1 - 其中 m 表示通道对(n) 和(n+1)	C(n)V 寄存器将以其写入缓冲区值更新。
C(n+1)V 寄存器，且 SYNCENm = 1 - 其中 m 表示通道对(n) 和(n+1)	C(n+1)V 寄存器将以其写入缓冲区值更新。

注:

如果 ELSjB 和 ELSjA 位不为零，则根据配置的输出模式生成通道(j)输出信号。如果 ELSjB 和 ELSjA 位为零，则生成的信号在通道(j)输出中不可用。

如果 CHjIE = 1，将在发生通道(j)匹配时生成通道(j)中断。

在中断加载，通道输出和 ETM 计数器均不改变。软件必须及时在一个安全点设置中间加载。

中间加载特性只能用于组合模式。

● 6.3.8.26 全局时基（GTB)

全局时基(GTB)是一项 ETM 功能，允许同步一个芯片上的多个 ETM 模块。下图以示例说明了如何利用 GTB 特性同步两个 ETM 模块。在本例中，ETM A 和 B 通道运作起来就好像只使用了一个 ETM 模块，这就是全局时基。

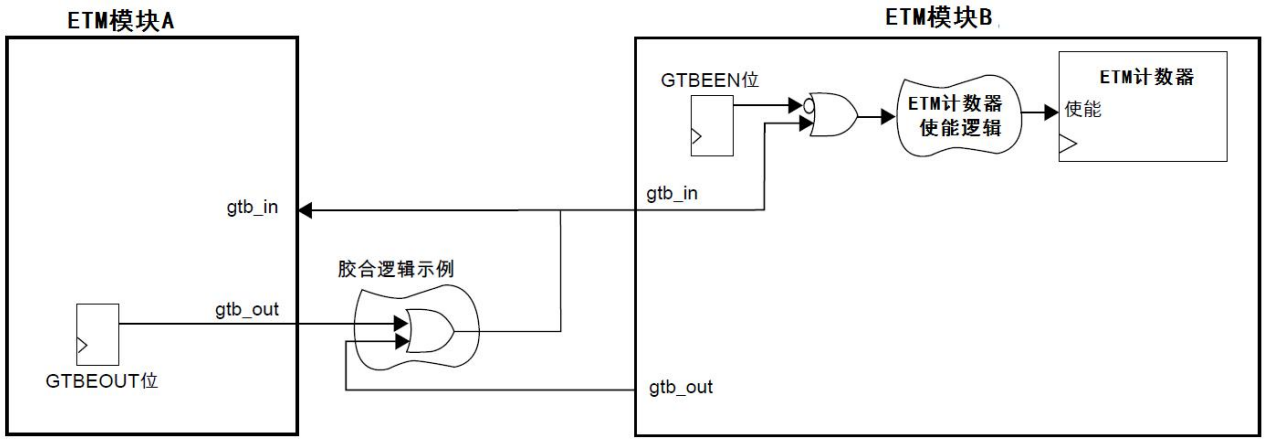


图 6-89 全局时基(GTB)功能框图

GTB 功能可通过 CONF 寄存器中的 GTBEEN 和 GTBEOUT 位以及输入信号 gtb_in 和输出信号 gtb_out 实现。GTBEEN 位让 gtb_in 能够控制 ETM 计数器使能信号：

- 如果 GTBEEN = 0，每一个 ETM 模块都将根据各自的配置模式独立工作。
- 如果 GTBEEN = 1，则只在 gtb_in 为 1 的情况下使能 ETM 计数器更新。

在上图所述的配置中，如果其中某个 ETM 模块的至少一个 gtb_out 信号为 1，则使能 ETM 模块 A 和 B 的 ETM 计数器。对于 gtb_in 和 gtb_out 信号的互连，有多种可能的配置，图中的示例胶合逻辑即体现了这一点。注意，这些配置是独立于芯片的，在 ETM 模块外部实施。有关芯片的具体实施，请参见芯片配置详情。

注 ：

- 为了使用 GTB 信号同步不同 ETM 模块的 ETM 计数器，每个 ETM 模块的配置都应该确保其 ETM 计数器在 gtb_in 信号为 1 时立刻开始计数。
- GTB 特性不提供 ETM 计数器的持续同步，这就意味着在 ETM 工作期间 ETM 计数器可能丢失同步。GTB 特性只允许 ETM 计数器同时开始它们的工作。

❖ 6.3.8.26.1 使能全局时基(GTB)

要使能 GTB 特性，请对每个参与的 ETM 模块执行以下步骤：

1. 停止 ETM 计数器：向 SC[CLKS]写入 00b。
2. 将 ETM 编程为预期配置。ETM 计数器模式需要在所有参与模块之间保持一致。
3. 向 CONF[GTBEEN]写入 1，同时向 CONF[GTBEOUT]写入 0。
4. 在 SC[CLKS]中选择预期的 ETM 计数器时钟源。时钟源需要在所有参与模块之间保持一致。
5. 复位 ETM 计数器：向 CNT 寄存器中写入任意值。

要在上图所述配置中启动 GTB 特性，请向用作时基的 ETM 模块中的 CONF[GTBEOUT]写入 1。

■ 6.3.9 复位概述

无论何时，只要有芯片复位，ETM 就会复位。

ETM 何时从复位状态中退出：

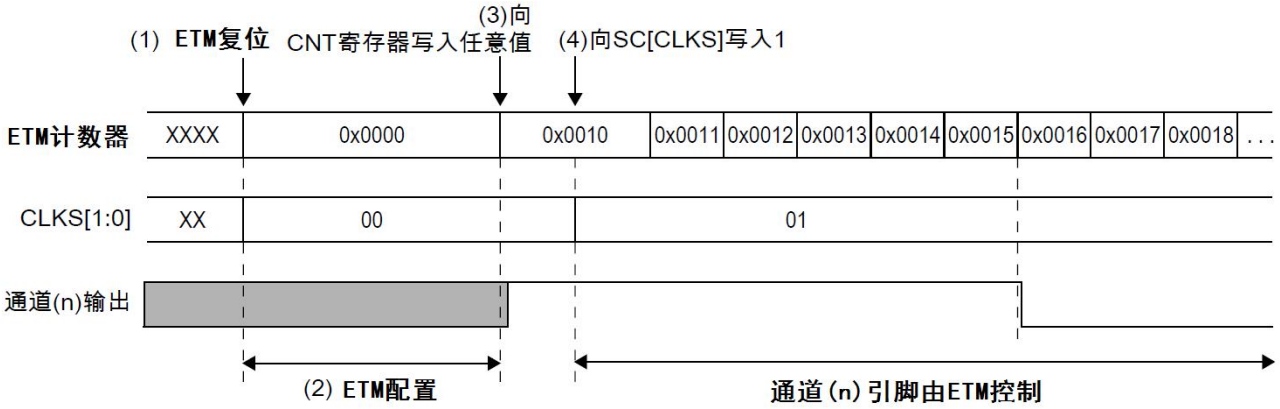
- ETM 计数器和预分频器计数器为零，且已停止(CLKS[1:0] = 00b)；
- 定时器溢出中断为零，参见定时器溢出中断 ；
- 通道中断为零，参见通道(n)中断 ；
- 故障中断为零，参见故障中断 ；
- 通道处于输入捕捉模式，参见输入捕捉模式 ；
- 通道输出为零；
- 通道引脚不由 ETM 控制(ELS(n)B:ELS(n)A = 0:0)（参见 CnSC 寄存器说明中的表格）。

下图展示了 ETM 在复位之后的行为方式。复位时（项目 1），ETM 计数器将会禁用（参见状态和控制寄存器中的 CLKS 字段说明），其值将更新为零，而且引脚不由 ETM 控制（参见 CnSC 寄存器说明中的表格）。

复位之后，应该对 ETM 进行配置（项目 2）。有必要根据通道模式来定义 ETM 计数器模式、ETM 计数限制（MOD 和 CNTIN 寄存器值）、通道模式和的 CnV 寄存器值。

因此，建议向 CNT 寄存器中写入任意值（项目 3）。采取这种写入操作之后，ETM 计数器会根据 CNTIN 寄存器值更新，通道输出会根据其初始值更新（处于输出比较模式的通道则例外）（ 计数器复位 ）。

下一步是通过 CLKS[1:0]位选择 ETM 计数器时钟（项目 4）。有必要强调的一点是，CLKS[1:0]位不为零时，引脚将只由 ETM 控制（参见 CnSC 寄存器说明中的表格）。

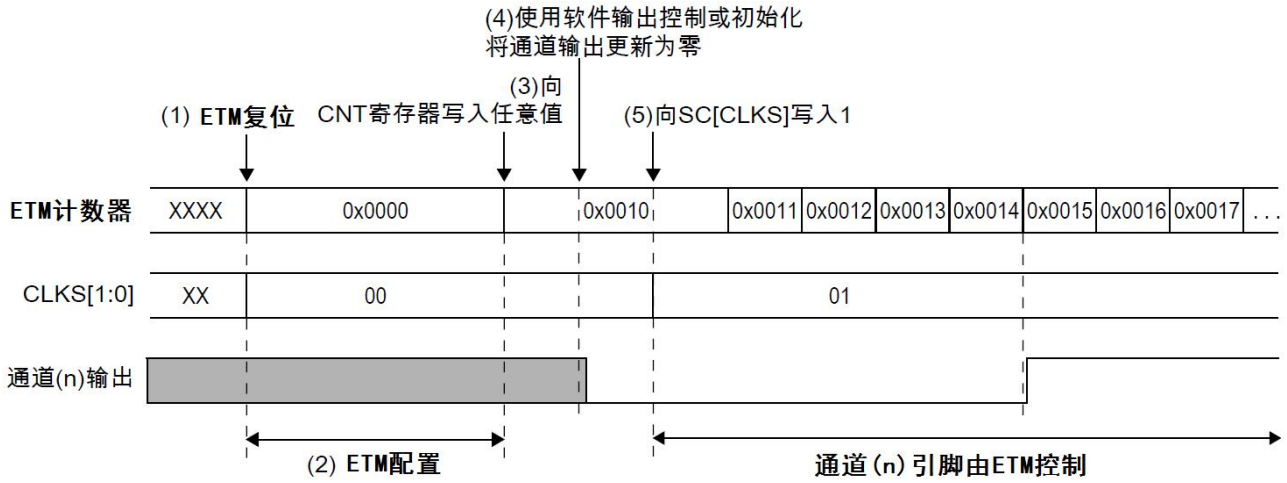


注释：

- CNTIN = 0x0010
- 通道(n)处于组合模式，CNTIN < C(n)V < C(n+1)V < MOD
- C(n)V = 0x0015

图 6-90 通道(n) 处于组合模式时复位之后的 ETM 行为方式

下图以示例说明了当通道(n)处于输出比较模式时，如果存在匹配，则切换通道(n)输出。在输出比较模式下，对 CNT 寄存器采取写入操作后（项目 3），通道输出不会更新为其初始值。这种情况下，利用软件输出控制（ 软件输出控制 ）或初始化（ 初始化 ）将通道输出更新为所选值（项目 4）。



注释：

- CNTIN = 0x0010
- 通道(n)处于输出比较模式，存在匹配时，将切换通道(n)输出
- C(n)V = 0x0014

图 6-91 通道(n）处于输出比较模式时复位之后的 ETM 行为方式

■ 6.3.10 ETM 中断

● 6.3.10.1 定时器溢出中断

(TOIE=1)且（TOF=1）时，将生成定时器溢出中断。

● 6.3.10.2 通道(n)中断

(CHnIE=1)且（CHnF=1）时，将生成通道(n)中断。

● 6.3.10.3 故障中断

当(FAULTIE=1)且（FAULTF=1）时，将生成故障中断。

■ 6.4 周期性中断定时器（PIT）

■ 6.4.1 简介

PIT 模块是一组可用于发起中断和触发的定时器。

■ 6.4.2 结构框图

下图是 PIT 模块的结构框图

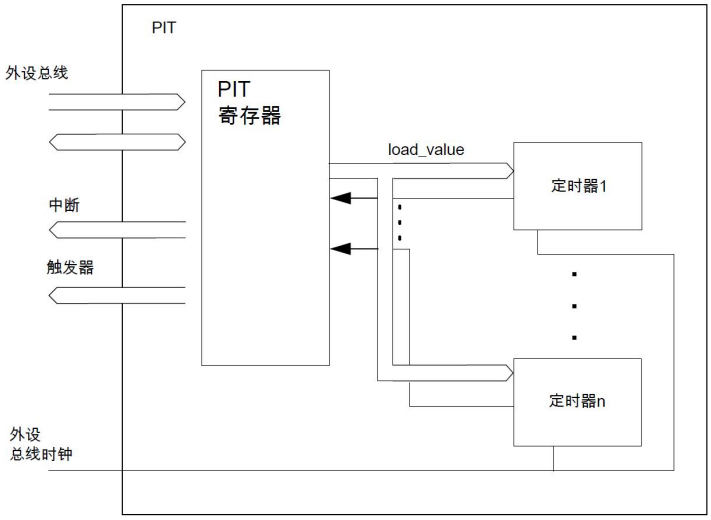


图 6-92 PIT 结构框图

注：有关该 MCU 使用的 PIT 通道数，请参见芯片配置详细信息。

■ 6.4.3 特性

该模块具有如下主要特性：

- 定时器能够生成触发脉冲
- 定时器能够生成中断
- 可屏蔽中断
- 每个定时器都具有独立的超时周期

■ 6.4.4 信号说明

PIT 模块没有外部引脚。

■ 6.4.5 存储器映射和寄存器说明

本节详细说明 PIT 模块中的可访问的所有寄存器。

- 保留寄存器将读取为 0，写操作将无效。
- 有关 MCU 使用的 PIT 通道数，请参考芯片配置详细信息。

表 6-53 PIT 存储器映射

绝对地址（16 进制）	寄存器名称	位宽	访问	复位值
4003_7000	PIT 模块控制寄存器(PIT_MCR)	32	R/W	0000_0006h
4003_7100	定时器加载值寄存器 0(PIT_LDVAL0)	32	R/W	0000_0000h
4003_7104	当前定时器值寄存器 0(PIT_CVAL0)	32	R	0000_0000h
4003_7108	定时器控制寄存器 0(PIT_TCTRL0)	32	R/W	0000_0000h
4003_710C	定时器标志控制寄存器 0(PIT_TFLG0)	32	R/W	0000_0000h
4003_7110	定时器加载值寄存器 1(PIT_LDVAL1)	32	R/W	0000_0000h
4003_7114	当前定时器值寄存器 1(PIT_CVAL1)	32	R	0000_0000h
4003_7118	定时器控制寄存器 1(PIT_TCTRL1)	32	R/W	0000_0000h
4003_711C	定时器标志控制寄存器 1(PIT_TFLG1)	32	R/W	0000_0000h

● 6.4.5.1 PIT 模块控制寄存器（PIT_MCR）

该寄存器在 PIT 进入调试模式时使能或禁用 PIT 定时器时钟以及控制定时器。

地址：4003_7000h + 0h 偏移= 4003_7000h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0													保留	MDIS	FRZ
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-54 PIT_MCR 字段描述

字段	描述
31-3 保留	此字段为保留字段 此只读字段为保留字段且值始终为 0.
2 保留	保留
1 MDIS	模块禁用 -（PIT 部分） 禁用标准定时器。必须在执行任何其他设置之前使能该字段。 0 标准 PIT 定时器的时钟使能。 1 标准 PIT 定时器的时钟禁用。
0 FRZ	冻结 当器件进入调试模式时，允许停止定时器。 0 定时器在调试模式下继续运行。 1 定时器在调试模式下停止运行。

● 6.4.5.2 定时器加载值寄存器（PIT_LDVALn）

这些寄存器选择定时器中断的超时周期。

地址：PIT_LDVAL0 = 4003_7100h / PIT_LDVAL1 = 4003_7110h

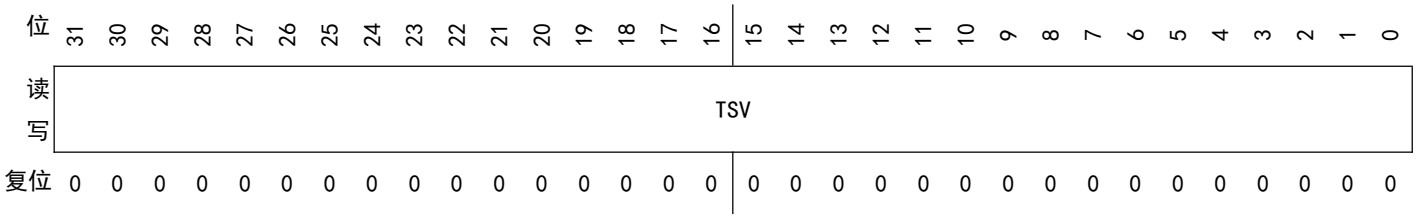


表 6-55 PIT_LDVALn 字段描述

字段	描述
31-0 TSV	定时器起始值 设置定时器起始值。定时器将倒计时至 0，然后生成一次中断并再次加载该寄存器值。将新值写入寄存器不会重启定时器，定时器到期后会加载新值。要中指当前周期并用新值开始一个定时器周期，必须先禁用该定时器然后再将其使能。

● 6.4.5.3 当前定时器加载值寄存器（PIT_CVALn）

这些寄存器指示当前定时器位置。

地址：PIT_CVAL0 = 4003_7104h / PIT_CVAL11 = 4003_7114h

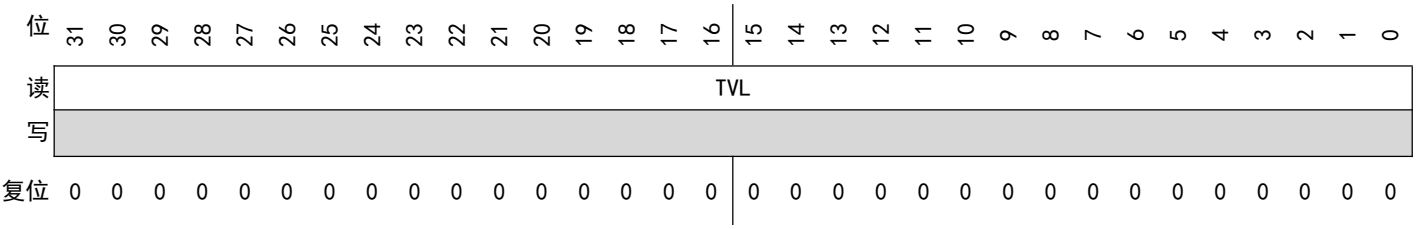


表 6-56 PIT_CVALn 字段描述

字段	描述
31-0 TVL	当前定时器值 代表当前定时器值（若定时器已经使能）： 注： 1.若定时器已禁用，请勿使用该字段，因为其值不可靠。 2.定时器使用向下计数器。如果 MCR[FRZ]置位，定时器在调试模式下会被冻结。

● 6.4.5.4 定时器控制寄存器（PIT_TCTRLn）

这些寄存器包括各定时器的控制位。

地址：PIT_TCTRL0 = 4003_7108h / PIT_TCTRL1 = 4003_7118h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0													CHN	TIE	TEN
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-57 PIT_TCTRLn 字段描述

字段	描述
31-3 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
2 CHN	链模式 激活时，定时器 n-1 需先到期，定时器 n 才能递减 1. 不能链接定时器 0. 0 定时器不链接 1 定时器链接到前一定时器。例如，对于通道 1，如该字段置位，则定时器 1 链接到定时器 0。
1 TIE	定时器中断使能 某个中断挂起或 TFLGn[TIF]置位时，使能该中断将立即引起中断事件。为避免这种情况，必须先清零相关的 TFLGn[TIF]。 0 定时器 n 的中断请求禁用 1 只要 TIF 置位，就会请求中断
0 TEN	定时器使能 使能或禁用定时器 0 定时器 n 禁用 1 定时器 n 使能

● 6.4.5.5 定时器标志寄存器（PIT_TFLGn）

这些寄存器占有 PIT 中断标志。

地址：PIT_TFLG0 = 4003_710Ch / PIT_TFLG1 = 4003_711Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0															TIF
写																w1c
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-58 PIT_TFLGn 字段描述

字段	描述
31-1 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
0 TIF	定时器中断标志。注：w1c 表示写 1 该位就会清零，写 0 无效。 在定时器周期结束时置 1。将 1 写入该标志可将其清零。写入 0 则无效。若使能或 TCTRLn[TIE]=1，TIF 将引发中断请求。 0 超时尚未发生 1 超时已经发生

■ 6.4.6 功能说明

本节说明该模块操作。每个定时器都可用于生成触发脉冲和中断。每个中断都可用于单独的中断线。

● 6.4.6.1 一般操作

每个引脚的逻辑状态可通过端口数据输入寄存器来指明，假设已将该引脚配置用于执行数字功能且对应的端口控制和中断模块已使能。

✧ 6.4.6.1.1 定时器

定时器在使能时定期生成触发脉冲。定时器加载 LDVAL 寄存器指定的起始值，倒数至 0，然后再次加载相应的起始值。每次定时器达到 0 时，它就会生成一个触发脉冲并置位中断标志。

所有中断都可通过设置 TCTRLn[TIE]来使能或屏蔽。新的中断只能在清除上一个中断之后生成。

需要时，定时器的当前计数值可通过 CVAL 寄存器读取。通过 TCTRLn[TEN]先禁用定时器再将其使能可重启计数周期。参见下图：

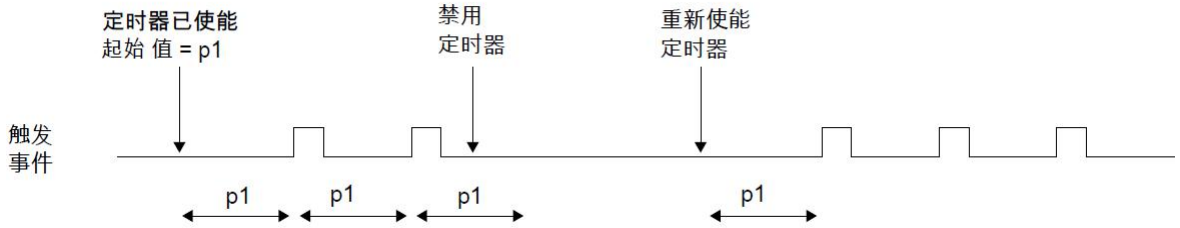


图 6-93 停止和启动定时器

先禁用定时器，设置新加载值，然后再次使能定时器，可以修改运行中的定时器的计数周期。参见下图:

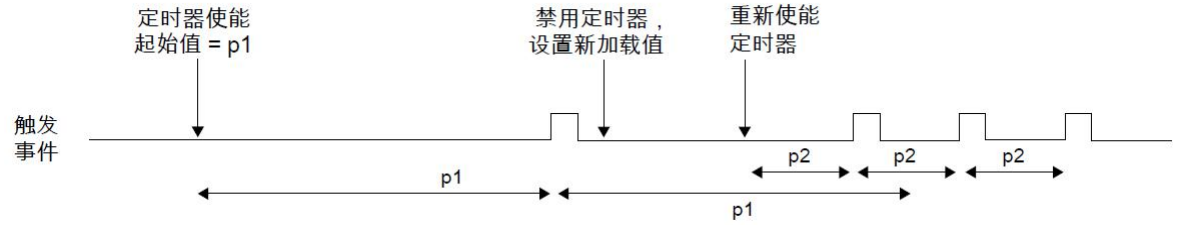


图 6-94 修改运行中的定时器的周期

还可以在不重启定时器的情况下更改计数周期，方法是将新的加载值写入 LDVAL。该值将在下一个触发事件之后加载。参见下图：

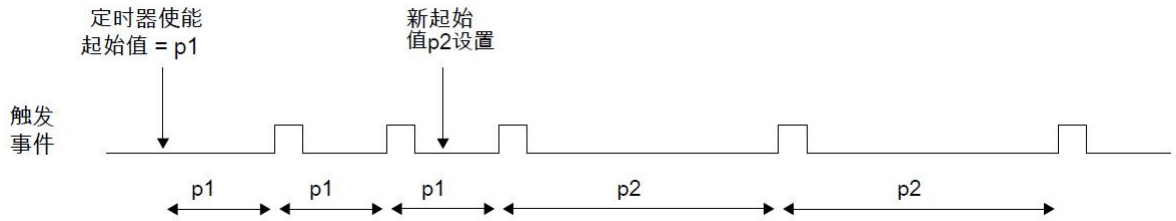


图 6-95 动态设置新加载值

✧ 6.4.6.1.2 调试模式

在调试模式下，定时器将根据 MCR[FRZ]决定是否冻结。其目的是协助软件开发，使开发人员能够暂停处理器，调查系统的当前状态，如定时器值等，然后继续运行。

● 6.4.6.2 中断

所有定时器都支持中断生成。相关的向量地址和优先级，请参见 MCU 规范。

定时器中断可通过置位 TCTRLn[TIE]来使能。相关定时器发生超时时，TFLGn[TIF]置位为 1；将 1 写入对应的 TFLGn[TIF]可将其清零。

● 6.4.6.3 链接定时器

如果某个定时器使链模式处于使能状态，那么只有在上一个定时器溢出后，它才会开始计时。因此，如果定时器 n-1 已倒数至 0，定时器 n 的值将递减 1。这样就能将某些定时器链接起来形成更长的定时器。第一个定时器（定时器 0）不能链接至任何其他定时器。

■ 6.4.7 初始化和应用信息

在配置示例中：

- 系统时钟配置为 FEE 模式，使用外部 10M 晶振，PIT 时钟的频率为 50MHz。
- 定时器 0 每隔 5.12ms 生成一个中断。
- 定时器 1 每隔 30ms 生成一个触发事件。

PIT 模块必须通过将 0 写入 MCR[MDIS]来激活。

50MHz 时钟频率相当于 20ns 时钟周期。定时器 0 需要每隔 5.12ms/20ns =256,000 个周期触发一次，定时器 1 每隔 30ms/20ns = 1,500,000 个周期触发一次。LDVAL 寄存器触发器的值计算如下：

LDVAL 触发器 = （周期 / 时钟周期）-1

这意味着必须分别将 0x0003E7FF 和 0x0016E35F 写入 LDVAL0 和 LDVAL1。定时器 0 的中断通过置位 TCTRL0[TIE]来使能。定时器通过将 1 写入 TCTRL0[TEN]来启动。定时器 1 只能用于触发。因此，定时器 1 通过将 1 写入 TCTRL1[TEN]来启动。TCTRL1[TIE]保持为 0。

下面的示例代码与上述设置一致：

```
// turn on PIT
PIT_MCR = 0x00;
// Timer 0
```

```
PIT_LDVAL0 = 0x0003E7FF; // setup timer 0 for 256000 cycles
PIT_TCTRL0 = TIE; // enable Timer 0 interrupts
PIT_TCTRL0 |= TEN; // start Timer 0
// Timer 1
PIT_LDVAL1 = 0x0016E35F; // setup timer 1 for 1500000 cycles
PIT_TCTRL1 |= TEN; // start Timer 1
```

■ 6.4.8 链接定时器配置示例

在配置示例中：

- 系统时钟配置为 FEE 模式，使用外部 10M 晶振，PIT 时钟的频率为 25 MHz。
- 定时器 0 和定时器 1 可用。
- 每隔半小时应发生一次中断。

PIT 模块需要通过将 0 写入 MCR[MDIS]来激活。

25MHz 时钟频率相当于 40 ns 的时钟周期，因此 PIT 需要计数 450 亿个周期，这不是单个定时器能够做到的。因此，将定时器 0 设置为每 18s（18 亿个周期）触发一次。定时器 1 链接至定时器 0，经过编程后触发 25 次。

LDVAL 寄存器触发器的值等于周期数减 1,因此 LDVAL0 接收值 0x6B49D1FF,LDVAL1 接收值 0x00000018。定时器 0 的中断通过置位 TCTRL0[TIE]来使能，链模式通过置位 TCTRL1[CHN]来激活，定时器通过将 1 写入 TCTRL1[TEN]来启动。TCTRL0[TEN]需要置位，TCTRL0[CHN]和 TCTRL0[TIE]需要清零。

下面的示例代码与上述设置一致：

```
// turn on PIT
PIT_MCR = 0x00;
// Timer 1
PIT_LDVAL1 = 0x00000018; // setup Timer 1 for 25 counts
PIT_TCTRL1 = TIE; // enable Timer 1 interrupt
PIT_TCTRL1 |= CHN; // chain Timer 1 to Timer 0
PIT_TCTRL1 |= TEN; // start Timer 1
// Timer 0
PIT_LDVAL0 = 0x6B49D1FF; // setup Timer 0 for 1800 000 000 cycles
PIT_TCTRL0 = TEN; // start Timer 0
```

■ 6.5 实时计数器（RTC）

■ 6.5.1 简介

实时计数器(RTC)由一个 16 位计数器、一个 16 位比较器、若干个基于二进制和基于十进制的预分频器、三个时钟源、一个可编程周期性中断和一个可编程外部切换脉冲输出组成。此模块可用于计时、日历或任何任务调度功能。它还能充当循环唤醒，将器件从低功耗模式、停止模式和等待模式中唤醒而无需外部组件。

■ 6.5.2 特性

RTC 模块特性包括：

- 16 位向上计数器
  - 16 位模数匹配限制
  - 软件可控制的周期性匹配中断
- 可编程 16 位预分频器可由软件选择时钟源
  - OSC 32.768KHz （标称值）
  - LPO（约 1kHz）
  - 总线时钟
  - 内部参考时钟(32kHz)

● 6.5.2.1 工作模式

本节说明 RTC 在停止、等待和后台调试模式下的操作。

❖ 6.5.2.1.1 等待模式

如果 RTC 在执行 WAIT 指令前已使能，那么在等待模式下会继续运行。因此，如果实时中断使能，就可利用 RTC 使 MCU 离开等待模式。为实现最低电流消耗，如果 RTC 在等待模式期间无需用作中断源 ，那么必须通过软件使其停止。

❖ 6.5.2.1.2 停止模式

如果在执行 STOP 指令之前已使能 RTC，则此 RTC 能够继续在停止模式下运行。因此，如果已使能实时中断，则此 RTC 可用于使 MCU 退出停止模式，而无需外部组件。

● 6.5.3 结构框图

下图显示的是 RTC 模块的结构框图。

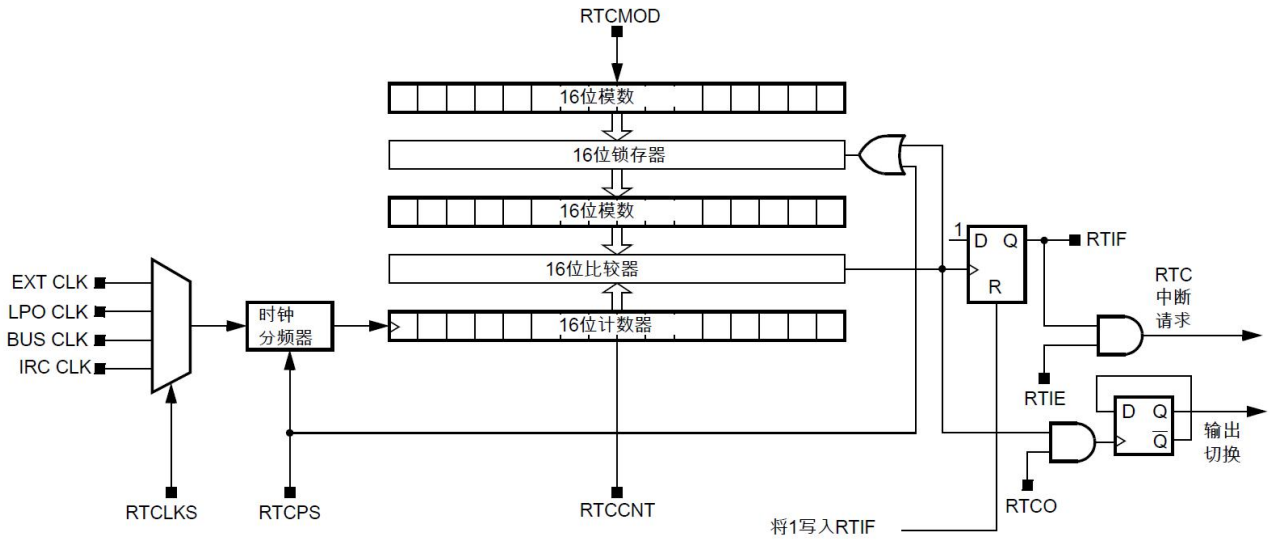


图 6-96 实时时钟（RTC)结构框图

■ 6.5.4 外部信号说明

RTCO 是 RTC 的输出。MCU 复位后，RTC_SC[RTCO]置位为高电平。计数器溢出时，输出切换。

■ 6.5.5 存储器映射和寄存器说明

RTC 包括状态和控制寄存器、16 位计数器寄存器和 16 位模数寄存器。

表 6-59 RTC 存储器映射

绝对地址（16 进制）	寄存器名称	偏移量	位宽	访问	复位值
4003_D000	RTC 状态和控制寄存器(RTC_SC)	00h	32	R/W	0000_0000h
4003_D004	RTC 数模寄存器(RTC_MOD)	04h	32	R/W	0000_0000h
4003_D008	RTC 计数器寄存器(RTC_CNT)	08h	32	R	0000_0000h

● 6.5.5.1 RTC 状态和控制寄存器(RTC_SC)

RTC_SC 包括实时中断状态标志(RTIF)和切换输出使能位(RTCO)。

地址：4003_D000h 基址 + 0h 偏移 = 4003_D000h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	RTCLKS		0			RTCPS			RTIF	RTIE	0	RTCO	0			
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-59 RTC_SC 字段描述

字段	描述
31-16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
15-14 RTCLKS	实时时钟源选择 该读/写字段选择 RTC 预分频器的时钟源输入。更改时钟源会将预分频器和 RTCCNT 计数器清零。复位会将 RTCLKS 清除为 00。 00 外部时钟源 01 实时时钟源为 1kHz（LPOCLK）。 10 内部参考时钟（ICSIRCLK）。 11 总线时钟。
13-11 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
10-8 RTCPS	实时时钟预分频器选择 该读/写字段为时钟源选择基于二进制或基于十进制的分频值。更改预分频器值会将预分频

	器和 RTCCNT 计数器清零。复位会将 RTCPS 清除为 000。 000 关闭。 001 如果 RTCLKS=x0,它为 1;如果 RTCLKS=x1,它为 128。 010 如果 RTCLKS=x0,它为 2;如果 RTCLKS=x1,它为 256。 011 如果 RTCLKS=x0,它为 4;如果 RTCLKS=x1,它为 512。 100 如果 RTCLKS=x0,它为 8;如果 RTCLKS=x1,它为 1024。 101 如果 RTCLKS=x0,它为 16;如果 RTCLKS=x1,它为 2048。 110 如果 RTCLKS=x0,它为 32;如果 RTCLKS=x1,它为 100。 111 如果 RTCLKS=x0,它为 64;如果 RTCLKS=x1,它为 1000。
7 RTIF	实时中断标志 该状态位指示 RTC 计数器寄存器已达到 RTC 模数寄存器中的值。写入逻辑 0 无效。写入逻辑 1 会将该位清零并清除实时中断请求。复位会将 RTIF 清除为 0。 0 RTC 计数器未达到 RTC 模数寄存器中的值。 1 RTC 计数器已达到 RTC 模数寄存器中的值。
6 RTIE	实时中断使能 该读/写位使能实时中断。如果 RTIE 置位，那么在 RTIF 置位时会生成中断。复位会将 RTIE 清除为 0。 0 实时中断请求禁用。使用软件轮询。 1 实时中断请求使能。
5 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
4 RTCO	实时计数器输出 该读/写位使能实时计数器把切换输出到引脚上。如果该位置位，那么在 RTC 计数器溢出时，将切换 RTCO 至引脚。 0 实时计数器输出禁用。 1 实时计数器输出使能。
3-0 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。

● 6.5.5.2 RTC 数模寄存器(RTC_MOD)

RTC_MOD 指示 16 位模数值。

地址：4003_D000h 基址+ 4h 偏移 = 4003_D004h

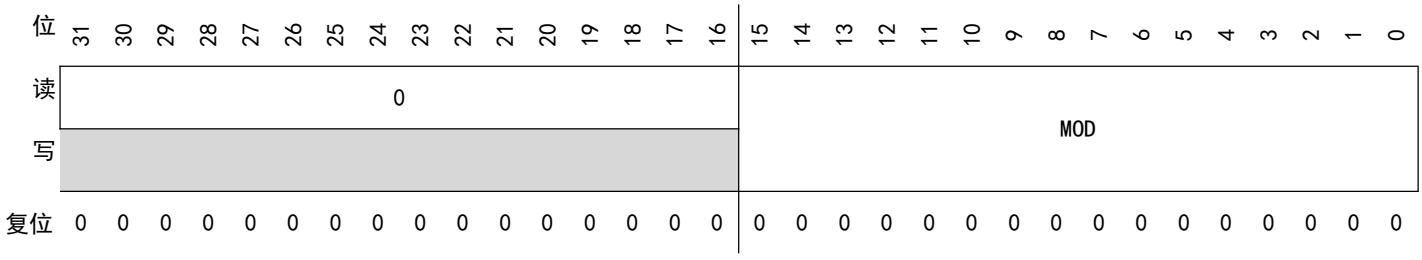


表 6-60 RTC_MOD 字段描述

字段	描述
31-16	此字段为保留字段。

保留	此只读字段为保留字段且值始终为 0。
15-0 MOD	RTC 模数 该读/写字段包括模数值，在比较结果匹配时将计数值复位到 0x0000，以及置位 SC[RTIF]状态字段。值 0x0000 使 SC[RTIF]在预分频器输出的每个上升沿都置位。复位会将该模数设置为 0x0000。

● 6.5.5.3 RTC 计数寄存器(RTC_CNT)

RTC_CNT 指示 16 位计数器的当前 RTC 计数的只读值。

地址：4003_D000h 基址 + 8h 偏移 = 4003_D008h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0																CNT															
写																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 6-61 RTC_CNT 字段描述

字段	描述
31-16 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
15-0 CNT	RTC 计数 该只读字段包括 16 位计数器的当前值，先读取 CNT[7:0]，再读取 CNT[15:8]。写操作对该寄存器无效。复位或将其他值写入 SC[RTCLKS]和 SC[RTCPS]会将该计数清除为 0x00000。

■ 6.5.6 功能说明

RTC 由一个带 16 位模数寄存器的主 16 位上行计数器、一个时钟源选择器以及一个预分频器模块（可选择基于二进制或基于十进制的值）组成。该模块还包含用于引脚分配的软件可选中断逻辑和和切换逻辑。

MCU 复位后，计数器停止并复位到 0x0000，模数寄存器设置到 0x0000，预分频器关闭。外部振荡器时钟被选作默认时钟源。要启动预分频器，需将非 0 值写入预分频器选择字段(RTC_SC[RTCPS])。

时钟源可通过软件选择：外部振荡器(OSC)、片上低功耗振荡器(LPO)、32-kHz 内部基准时钟和总线时钟。RTC 时钟选择字段(RTC_SC[RTCLKS])用于选择预分频器所需的时钟源。如果将不同的值写入 RTC_SC[RTCLKS]，预分频器和 CNT 计数器会复位到 0x00。

RTC_SC[RTCPS]和 RTC_SC[RTCLKS]选择所需的分频值。如果将不同的值写入 RTC_SC[RTCPS]，预分频器和 RTCCNT 计数器会复位到 0x00。下表列出了不同的预分频器周期值。

表 6-62 预分频器周期

RTCPS	32768 Hz OSC 时钟源预分频器周期 (RTCLKS=00)	LPO 时钟(1kHz)源预分频器周期 (RTCLKS=01)	内部参考时钟 (32.768kHz)源预分频器周期	总线时钟(8MHz)源预分频器周期 (RTCLKS=11)

			(RTCLKS=10)	
000	关闭	关闭	关闭	关闭
001	30.5176μs	128ms	30.5176μs	16μs
010	61.0351μs	256ms	61.0351μs	32μs
011	122.0703μs	512ms	122.0703μs	64μs
100	244.1406μs	1024ms	244.1406μs	128μs
101	488.28125μs	2048ms	488.28125s	256μs
110	976.5625μs	100ms	976.5625μs	12.5μs
111	1.9531ms	1s	1.9531ms	125μs

RTC 模数寄存器(RTC_MOD)允许将比较值设置为 0x0000 到 0xFFFF 之间的任意值。计数器处于激活状态时，会以所选速率递增，直到计数值与模数值匹配为止。

当这些值匹配时，计数器复位到 0x0000 并继续计数。只要出现匹配情况，实时中断标志(RTC_SC[RTIF])就会置位。该标志在模数值变为 0x0000 时置位。写入 RTC_MOD 的模数值处于锁存状态，直到 RTC 计数器溢出或选择的 RTC_SC[RTCPS]为非 0 值。

只要 RTC_SC[RTIF]置位，RTC 便允许生成一个中断。要使能实时中断，需将实时中断使能字段(RTC_SC[RTIE])置位。将 1 写入 RTC_SC[RTIF]可清零 RTC_SC[RTIF]。

通过电平翻转，RTC 还允许输出到外部管脚。必须置位 RTC_SC[RTCO]以使能外部管脚翻转。该功能有效时，电平取决于计数器溢出时管脚的前一状态。

● 6.5.6.1 RTC 操作示例

本节举例说明了计数器达到模数寄存器中匹配值时的 RTC 操作。

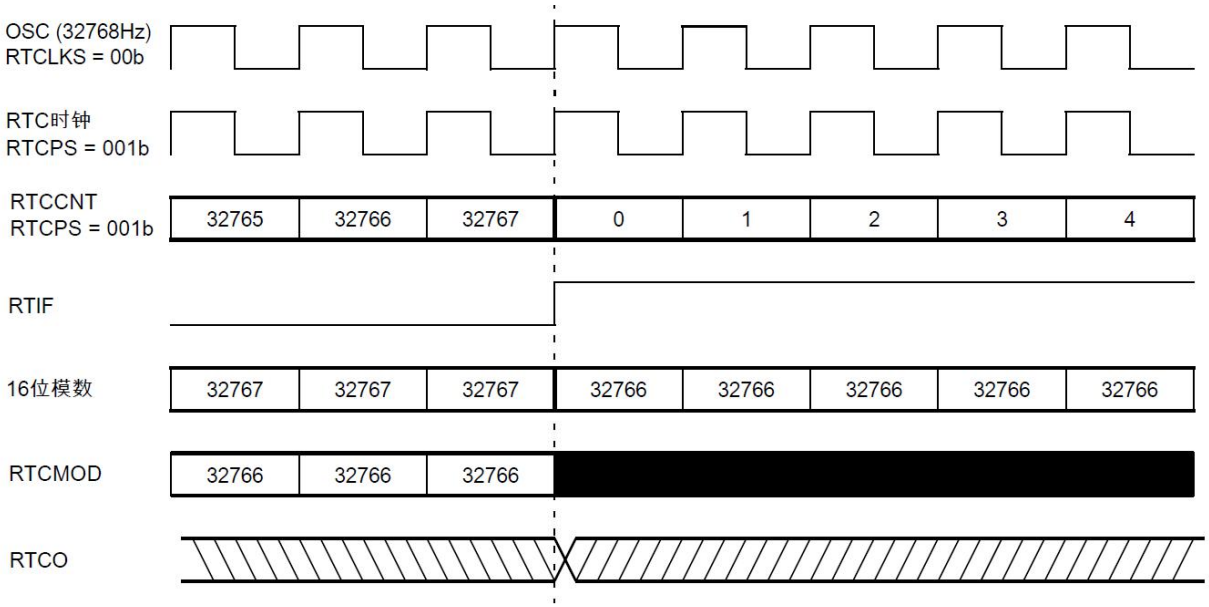


图 6-97 RTC 计数器溢出示例

在上例中，选择的是外部时钟源。预分频器设置为 RTC_SC[RTCPS] = 001b 或直通。当 RTC_MOD 寄存器中的模数值设置为 32766 时，16 位比较器使用的实际模数值是 32767。当计数器 RTC_CNT 达到模数值 32767 时，计数器溢出到 0x00 并继续计数。模数值通过提取 RTC_MOD 寄存器中的值来更新。当计数器值从 0x7FFF

变为 0x0000 时，实时中断标志 RTC_SC[RTIF]置位。RTC_SC[RTCO]在 RTC_SC[RTIF]置位时也翻转。

■ 6.5.7 初始化和应用信息

本节给出代码示例旨在就如何进行 RTC 模块的初始化和配置向用户提供基本指引。软件示例采用 C 语言实施。下述示例显示使用 OSC 时钟源实现可能的最低功耗时，如何用 RTC 实施计时。

示例: **6.5.7.1 RTC ISR** 中实施的软件日历

```
/* Initialize the elapsed time counters */
Seconds = 0;
Minutes = 0;
Hours = 0;
Days=0;
/* Configure RTC to interrupt every 1 second from OSC (32.768KHz) clock source */
RTC_MOD = 511; // overflow every 32 times
RTC_SC = RTC_SC_RTCPS_MASK; // external 32768 clock selected with 1/64 predivider.
RTC_SC = RTC_SC_RTIF_MASK | RTC_SC_RTIE_MASK; // interrupt cleared and enabled
/*****************************************************************
Function Name : RTC_ISR   Notes : Interrupt service routine for RTC module.
*****************************************************************/
void RTC_ISR(void)
{
/* Clears the interrupt flag, RTIF, and interrupt request */
RTC_SC |= RTC_SC_RTIF_MASK;
/* RTC interrupts every 1 Second */
Seconds++;
/* 60 seconds in a minute */
if (Seconds > 59)
{
Minutes++;
Seconds = 0;
}
/* 60 minutes in an hour */
if (Minutes > 59)
{
Hours++;
Minutes = 0;
}
/* 24 hours in a day */
if (Hours > 23)
{
Days ++;
Hours = 0;
}}
```

第 7 章 通信接口模块

■ 7.1 简介

本章对基于 ARM Cortex-M0+内核的 NV32F100x 系列 MCU 通信接口相关的模块做了详细说明，其中包括串口外设接口 SPI、I2C 总线、通用异步收发器 UART 等模块内容。

■ 7.2 串口外设接口 SPI 模块

■ 7.2.1 简介

SPI(Serial Peripheral Interface--串行外设接口)总线系统是一种同步串行外设接口，它可以使 MCU 与各种外围设备以串行方式进行通信以交换信息。SPI 有三个寄存器分别为：控制寄存器 SPCR，状态寄存器 SPSR，数据寄存器 SPDR。外围设备包括 FLASH、RAM、网络控制器、LCD 显示驱动器、A/D 转换器和 MCU 等。SPI 总线系统可直接与各个厂家生产的多种标准外围器件直接接口，该接口一般使用 4 条线：串行时钟线（SPSCK）、主机输入/从机输出数据线 MISO、主机输出/从机输入数据线 MOSI 和低电平有效的从机选择线 NSS。

在主机模式下，该 SPI 模块运行在一个波特率可达总线时钟的 1/2，从机模式下，波特率为总线时钟的 1/4。软件可以查询状态标志，或 SPI 操作可以作为中断驱动。

■ 7.2.2 特性

SPI 包含以下特性：

- 主机模式或从机模式操作
- 全双工或单线双向模式
- 可编程传输波特率
- 双缓冲区发送和接收数据寄存器
- 串行时钟相位和极性选项
- 从机选择输出
- 带 CPU 中断功能的模式错误标志
- 在等待模式控制 SPI 操作
- 可选 MSB 优先或 LSB 优先变换
- 接收数据缓冲硬件匹配功能

■ 7.2.3 操作模式

SPI 包含三个模式：

- 运行模式 （这是基础的模式操作）
- 等待模式

SPI 操作如果在等待模式下初始化为低功耗模式，控制该操作的 SPISWAI 位设在 SPIx_C2 寄存器。在等待模式下，如果 C2[SPISWAI]置 1，SPI 的运作就像在运行模式。如果 C2[SPISWAI]置 1 时，SPI 将进入低功耗状态，此时 SPI 时钟关闭。如果 SPI 配置为主机，任何正在进行的传输将停止，但会在 CPU 进入运行模式后恢复。如果 SPI 被配置为一个从机，接收和发送字节继续，如此从机都会保持同步到主机。

- 停止模式

为了降低功耗，SPI 处于停止模式，其中外围总线时钟已停止，但内部逻辑状态被保留。如果 SPI 配置为主机，正在进行的任何传输将停止，但会在 CPU 进入运行模式后恢复。如果 SPI 配置为数据的从机，接收和发送继续，如此从机都会保持同步到主机。

SPI 在停止模式下完全禁用，其中外设总线时钟停止和内部逻辑状态不会被保留。当 CPU 从这个停止模式唤醒，所有的 SPI 寄存器内容复位。

■ 7.2.4 结构框图

当 SPI 配置为主机时，时钟输出连接到 SPSCK 引脚，移位寄存器输出连接到 MOSI，移位寄存器输入从 MISO 管脚接入。当 SPI 配置为从机时，NSS 引脚为低电平从机选择有效，其他信号和主机模式相反。

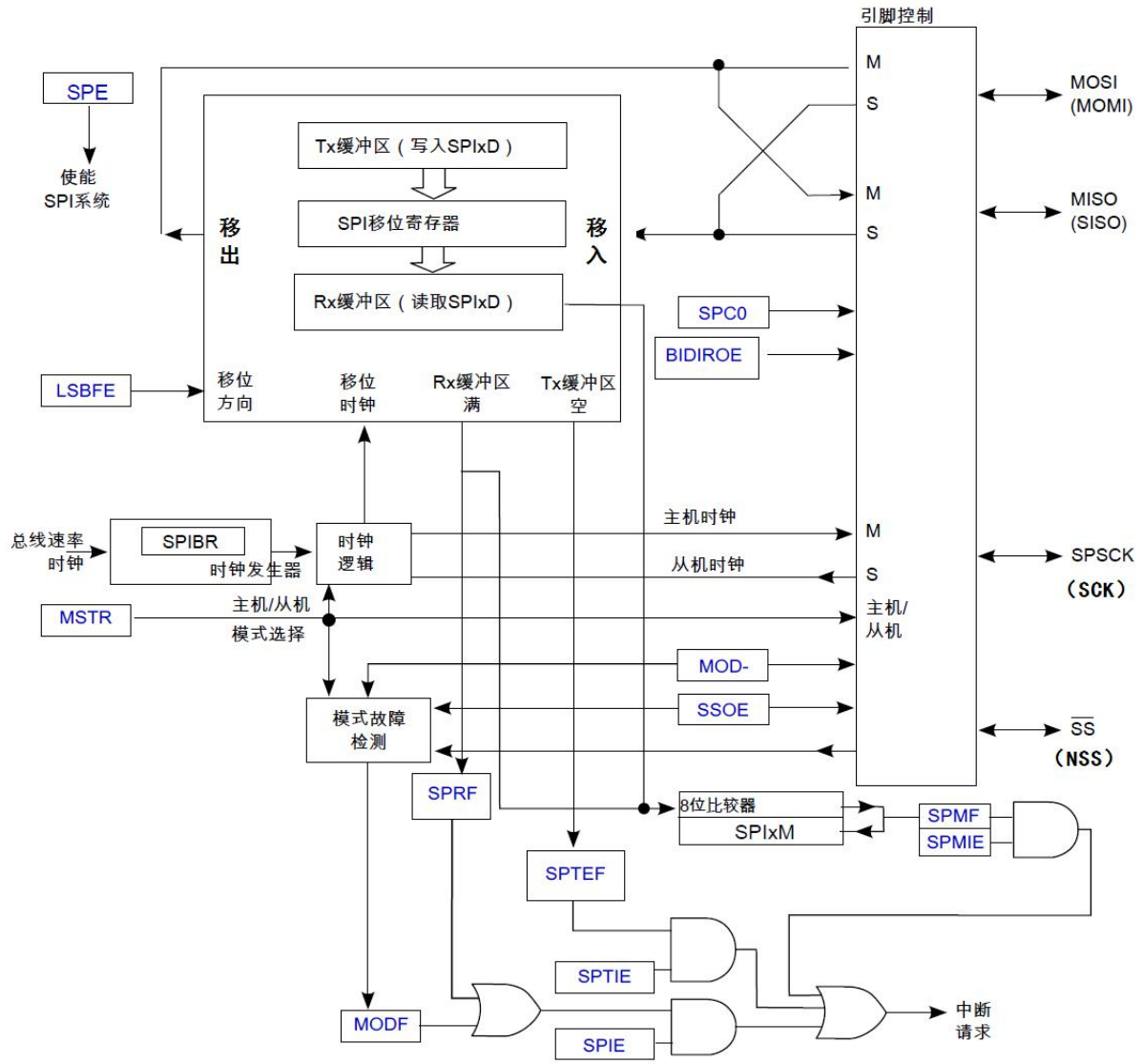


图 7-1 SPI 结构框图

■ 7.2.5 引脚配置

每个 SPI 模块有两种引脚配置，需要对 SIM 模块的相关寄存器进行配置，具体请参考 SIM 寄存器说明。

■ 7.2.6 信号描述

表 7-1 SPI 的信号描述

信号	描述	类型	引脚
MOSI	SPI 总线主机输入/从机输出。	I/O	PD1/PB3/PE1
SPSCK	SPI 串行时钟。	I/O	PD0/PB2/PE0
PCS0（NSS）	SPI 从机选择，低电平有效，参见 MODFEN 和 SSOE 字段详细说明。	I/O	PD3/PE3/PB5
MISO	SPI 总线主机输出/从机输入。	I/O	PD2/PE2/PB4

■ 7.2.7 存储器映射和寄存器说明

SPI 有 8 位的寄存器去选择 SPI 选项，去控制波特率，反馈 SPI 状态，保持 SPI 数据匹配值和收发数据。

表 7-2 存储器映射

绝对地址（16 进制）	寄存器名称	偏移量	位宽	权限	复位值
4007_*000h	SPI 控制寄存器 1（SPIx_C1）	0h	8	R/W	04h
4007_*001h	SPI 控制寄存器 2（SPIx_C2）	1h	8	R/W	00h
4007_*002h	SPI 波特率寄存器（SPIx_BR）	2h	8	R/W	00h
4007_*003h	SPI 状态寄存器（SPIx_S）	3h	8	R	20h
4007_*005h	SPI 数据寄存器（SPIx_D）	5h	8	R/W	00h
4007_*007h	SPI 匹配寄存器（SPIx_M）	7h	8	R/W	00h

注：SPI0 *=6, x=0 / SPI1 *=7 x=1

● 7.2.7.1 SPI 控制寄存器 1（SPIx_C1）

该读写寄存器包含 SPI 控制使能，中断使能，和配置选项。

地址：基准 + 0h 偏移 ：SPI0_C1 = 4007_6000h / SPI1_C1= 4007_7000h

位	7	6	5	4	3	2	1	0
读 写	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
复位值	0	0	0	0	0	1	0	0

表 7-3 SPIx_C1 字段描述

位	描述
7 SPIE	SPI 中断使能：对于 SPRF 和 MODF 对于 SPI 接收数据缓冲区满标志（SPRF）和模式错误标志（MODF）的中断使能。 0 从 SPRF 和 MODF 的中断被禁止——使用轮询 1 当 SPRF 和 MODF 为 1 时硬件中断请求

6 SPE	SPI 系统使能 启用 SPI 系统和将 SPI 端口引脚应用于 SPI 系统功能。如果 SPE 被清除，SPI 被禁止，被迫进入空闲状态，并且在 S 寄存器中的所有状态位都被复位。 0 SPI 系统被闲置 1 SPI 系统被启用
5 SPTIE	SPI 发送中断使能 这是一个发送数据缓存区为空的中断使能位。中断发生时，SPI 发送缓冲区为空（SPTEF 置 1）。 0 从 SPTEF 的中断被禁止（使用轮询） 1 当 SPTEF 为 1，有硬件中断请求
4 MSTR	主/从机选择 选择主机或从机模式 0 SPI 模块配置为从机模式 1 SPI 模块配置为主机模式
3 CPOL	时钟极性 选择一个倒置或不倒置 SPI 时钟。为了 SPI 模块之间的数据能相互传输 SPI 模块必须具有相同的 CPOL 值。 该位在主 SPI 设备或一个从 SPI 设备的串行时钟中有效放置了反相器。 0 高有效 SPI 时钟（空闲低） 1 低有效 SPI 时钟（空闲高）
2 CPHA	时钟相位 选择两种时钟格式给不同类型的同步串行外围设备。 0 SPSCK 第一边沿在数据传送的第一次循环的中间发生。 1 SPSCK 第一边沿在数据传送的第一次循环的起始发生。
1 SSOE	从机选择输出使能 此位与在 C2 寄存器的模式故障使能（MODFEN）字段和主/从（MSTR）控制位结合使用，以决定 SS 引脚的功能。 0 当 C2[MODFEN]为 0：在主机模式中，SS 引脚的功能是通用引脚。在从机模式中，SS 引脚是从机选择输入。当 C2[MODFEN]为 1：在主机模式中，SS 引脚的功能是模式错误的 SS 信号输出。在从机模式中，SS 引脚是从机选择输入。 1 当 C2[MODFEN]为 0：在主机模式中，SS 引脚的功能是通用引脚。在从机模式中，SS 引脚是从机选择输入。当 C2[MODFEN]为 1：在主机模式中，SS 引脚的功能是自动的 SS 信号输出。在从机模式中，SS 引脚是从机选择输入。
0 LSBFE	LSB 优先（移位方向） 该位不影响 MSB 和 LSB 在数据寄存器中的位置。数据寄存器的读取和写入总是有 MSB 的第 7 位。 0 最高有效位的串行数据传输开始 1 最低有效位的串行数据传输开始

● 7.2.7.2 SPI 控制寄存器 2（SPIx_C2）

该读写寄存器用于控制 SPI 系统的可选特征。

地址: 基准 + 1h 偏移 ：SPI0_C2 = 4007_6001h / SPI1_C2= 4007_7001h

位	7	6	5	4	3	2	1	0
读写	SPMIE	保留		MODFEN	BIDIROE	保留	SPISWAI	SPC0
复位值	0	0	0	0	0	0	0	0

表 7-4 SPI 控制寄存器 2 字段描述

位	描述
7 SPMIE	SPI 匹配中断使能 这是一个 SPI 接收数据缓冲硬件匹配功能的中断使能位。 SPMF 中断被禁止（使用轮询） 当 SOMF 为 1，有硬件中断请求
6-5 保留	该位保留 该只读位保留且有且仅有值 0
4 MODFEN	主机模式错误功能使能 当 SPI 配置为从机模式，该位没有意义或影响。 （SS 引脚是从机选择输入。）在主控模式下，该位决定 SS 引脚如何被使用。 0 模式错误功能关闭 1 模式错误功能开启
3 BIDIROE	双向模式输出使能 当双向模式被启用，因为 SPI 引脚控制（SPC0）设置为 1，BIDIROE 决定 SPI 数据输出驱动是否使能到单线双向 SPI I/O 引脚。SPI 使用 MOSI（MOMI）或 MISO（SISO）引脚取决于是否 SPI 被配置为主机或从机,分别作为 SPI 数据 I/O 引脚。当 SPC0 为 0,BIDIROE 没有意义或影响。 0 输出驱动器禁用，因此 SPI 数据 I/ O 引脚作为输入 1 SPI I/ O 引脚使能为输出
2 保留	该位保留 该只读位保留且有且仅有值 0
1 SPISWAI	SPI 在等待模式下停止 当设备处于等待模式下该位用于降低功耗。 0 SPI 时钟在等待模式下一直运行 1 SPI 时钟在等待模式下关闭
0 SPC0	SPI 引脚控制 使能双向引脚配置 0 SPI 使用独立引脚用于数据输入和数据输出（引脚模式是正常的）。 在主机模式：MISO 是主机输入和 MOSI 是主机输出。 在从机模式：MISO 是从机输出和 MOSI 是从机输入。 1 SPI 配置为单线双向操作（引脚模式是双向的）。 在主机模式：MISO 不使用 SPI; 当 BIDIROE 为 0，MOSI 是主机输入或当 BIDIROE 为 1，MOSI 为主机 I/O。 在从机模式：当 BIDIROE 为 0，MISO 是从机输入或当 BIDIROE 为 1，MISO 从机 I／O，;MOSI 不被 SPI 使用。

● 7.2.7.3 SPI 波特率寄存器（SPIx_BR）

使用该寄存器设置 SPI 主机的分频器和波特率因子。该寄存器可以在任何时间读出或写入。
波特率除数方程如下（除了在那些保留组合 SPI 波特率分频表）。

$$BaudRateDivisor=(SPPR+1)\times 2^{(SPR+1)}$$

波特率可以用下面的公式计算：
波特率=总线时钟/ BaudRateDivisor

地址: 基准 + 2h 偏移 ：SPI0_BR = 4007_6002h / SPI1_BR= 4007_7002h

位	7	6	5	4	3	2	1	0
读	0	SPPR[2:0]			SPR[3:0]			
写								
复位值	0	0	0	0	0	0	0	0

表 7-5 SPIx_BR 字段描述

位	描述	
7 保留	该位保留	
	该只读位保留并有且仅有值 0	
6-4 SPPR[2:0]	SPI 波特率分频因子 这 3 位字段选择八个因子为 SPI 波特率分频数之一。该分频器的输入是总线时钟速率（BUSCLK）。该分频器的输出驱动 SPI 波特率分频器的输入。	
	000	波特率分频因子为 1
	001	波特率分频因子为 2
	010	波特率分频因子为 3
	011	波特率分频因子为 4
	100	波特率分频因子为 5
	101	波特率分频因子为 6
	110	波特率分频因子为 7
	111	波特率分频因子为 8
3-0 SPR[3:0]	SPI 波特率因子 这 4 位字段选择九个除数之一为 SPI 波特率因子。	
	0000	波特率因子为 2
	0001	波特率因子为 4
	0010	波特率因子为 8
	0011	波特率因子为 16
	0100	波特率因子为 32
	0101	波特率因子为 64
	0110	波特率因子为 128
	0111	波特率因子为 256
	1000	波特率因子为 512
	其他	保留

- **7.2.7.4 SPI 状态寄存器（SPIx_S）**

该寄存器是只读寄存器。写没有任何意义。

地址: 基准 + 3h 偏移 ：SPI0_S = 4007_6003h / SPI1_S= 4007_7003h

位	7	6	5	4	3	2	1	0
读	SPRF	SPMF	SPTEF	MODF	0			
写								
复位值	0	0	1	0	0	0	0	0

表 7-6 SPIx_S 字段描述

位	描述
7 SPRF	SPI 读数据缓冲区满标志 SPRF 在 SPI 完成传送后置 1，以指示接收的数据可从 SPI 数据（D）寄存器读出。 SPRF 置 1 后，通过先读取该标志位，然后再读取数据寄存器的方式来清零。 0 接收数据缓冲区无数据可用 1 接收数据缓冲区有数据可用
6 SPMF	SPI 匹配标志 当在接收数据缓冲区的值与 M 寄存器的值相匹配时，SPMF 在 SPRF 为 1 后置 1。 SPMF 置 1 后，通过先读取该标志位，然后再向该位写 1 的方式来清零。 0 接收数据缓冲区里的值与 M 寄存器里的值不匹配 1 接收数据缓冲区里的值与 M 寄存器里的值匹配
5 SPTEF	SPI 发送数据缓冲区空标志 当发送数据缓冲区为空时，此位置 1。SPTEF 清零是通过读到 S 寄存器 SPTEF 置 1，然后写一个数据值到 D 寄存器。在写数据到 D 寄存器之前，必须读到 SPTEF 为 1 的 S 寄存器;否则，则 D 的写入被忽略。SPTEF 自动置 1 当所有数据从发送缓冲器传输到发送移位寄存器中。对于闲置 SPI，写入到 D 的数据被立即发送到移位寄存器中，使 SPTEF 在两个总线周期间置 1，允许第二组排队的数据进入发送缓冲器。在移位寄存器中的数据传输完成后，发送缓冲器中排队发送的数据自动移动到移位器,并且 SPTEF 被置 1 指示发送数据缓冲区为空可写入新的数据。如果没有新数据写入发送缓冲区，SPTEF 只是保持置 1 的状态并且没有数据从缓冲区到移位器移动。 如果传输没有停止，最后一个发送数据会被再次送出。 0 SPI 发送数据缓冲区不为空 1 SPI 发送数据缓冲区位空
4 MODF	主机模式错误标志 MODF 被置 1，如果 SPI 配置为主机那么从机选择输入变为低电平，指示一些其它 SPI 设备也被配置为主机。只有当 C1[MSTR]为 1，C2[MODFEN]为 1，和 C1[SSOE]为 0 时 SS 引脚作为模式故障输入;否则,MODF 不会被置 1。 MODF 为通过读置 1 的 MODF,然后写 SPI 控制寄存器 1(C1)清除。 0 没有模式故障 1 检测到模式故障
3-0 保留	该字段保留 该只读域保留并有且仅有值 0

● 7.2.7.5 SPI 数据寄存器（SPIx_D）

该寄存器是输入和输出寄存器，用于 SPI 数据收发。对该寄存器写意味着写数据到发送数据缓冲区，允许数据被排列并发送。

当 SPI 配置为主机，排列在发送数据缓冲区的数据会在先前传输已完成之后立即发送。

在 S 寄存器的 SPTEF 位用于表示发送数据缓冲区准备接受新的数据。写 SPI 之前 S 寄存器必须被读取到 SPTEF 为 1;否则，写操作将被忽略。

当 S[SPRF]置 1 后，到下一次传送结束前的任何时候均可从 SPI 数据寄存器读取接收到的数据。在新的传输结束前向数据接收缓冲区读取数据的失败将会导致接收溢出，并且新传送的数据丢失。新的数据将丢失因为接收缓冲区仍然保存先前的数据，还没有准备好接受新的数据。没有任何指示可以表示接收数据溢出，因此应用系统的设计者必须确保以前的数据已经从接收缓冲器读出在新的传送过程开始之前。

地址: 基准 + 5h 偏移 ：SPI0_D = 4007_6005h / SPI1_D= 4007_7005h

位	7	6	5	4	3	2	1	0
读 写	Bit[7:0]							
复位值	0	0	0	0	0	0	0	0

表 7-7 SPIx_D 字段描述

位	名称	描述
7-0	Bit	数据（低字节）

● 7.2.7.6 SPI 匹配寄存器（SPIx_M）

该寄存器包含硬件比较值。

地址: 基准 + 7h 偏移 ：SPI0_M = 4007_6007h / SPI1_M = 4007_7007h

位	7	6	5	4	3	2	1	0
读 写	Bit[7:0]							
复位值	0	0	0	0	0	0	0	0

表 7-8 SPIx_M 字段描述

位	名称	描述
7-0	Bit	硬件比较值（低字节）

■ 7.2.8 功能描述

● 7.2.8.1 功能概述

SPI 系统通过设置 SPI 控制寄存器 1 中的 SPI 使能（SPE）位启用。当 C1[SPE]置 1，四个关联的 SPI 端口引脚都应用于 SPI 功能

如：

- 从选择（SS）
- 串行时钟（SPSCK）
- 主输出/从输入（MOSI）
- 主输入/从输出（MISO）

一个 SPI 传输是通过读取 SPI 状态寄存器中 SPTEF 位为 1 的状态来启动在主 SPI 设备，然后将数据写入发送数据缓冲区（写入 SPIxD）。当传输完成后，接收到的数据被转移到接收数据缓冲区。SPIxD 寄存器用作 SPI 接收数据缓冲区（读操作）和 SPI 传送数据缓冲区（写操作）。

SPI 控制寄存器 1（SPIx_C1）的时钟相位控制位（CPHA）和时钟极性控制位（CPOL）会选择四个可能的时钟格式之一应用于 SPI 系统。该 CPOL 位简单地选择非反转或反转时钟。 C1[CPHA]是用来通过在 SPSCK 边沿奇采样数据或在 SPSCK 边沿偶采样数据来容纳两个根本不同的协议。

该 SPI 可以被配置为工作在主机模式或从机模式。当 SPI 控制寄存器 1 中的 MSTR 位被置 1 时，主机模式被选择;当 C1[MSTR]被清零，从机模式被选择。

● 7.2.8.2 主机模式

SPI 在 C1[MSTR]置位时以主机模式工作。只有主机 SPI 模块能发起传输。使传输开始的方法是：在 S[SPTEF] = 1 时先对 SPIx_S 寄存器进行读操作，然后对主机 SPI 数据寄存器进行写操作。如果移位寄存器为空，字节会立即转移到移位寄存器。 数据在串行时钟的控制下开始移出到 MOSI 引脚上。

- SPSCK
  - SPI 波特率寄存器中的波特率选择位 SPR3、SPR2、SPR1、SPR0 和波特率预选择位 SPPR2、SPPR1、SPPR0 共同控制波特率发生器并决定传输速度。SPSCK 引脚是 SPI 时钟输出。通过 SPSCK 引脚，主机的波特率发生器控制从机外设的移位寄存器。

- MOSI、MISO 引脚
  - 在主机模式下，串行数据输出引脚(MOSI)和串行数据输入引脚(MISO)的功能由 SPC0 和 BIDIROE 控制位确定。

- $\overline{SS}$ 引脚
  - 如果 C2[MODFEN]和 C1[SSOE]均置位，那么 SS 引脚配置为从机选择输出。SS 输出在每次传输期间变为低电平，SPI 处于空闲状态时变为高电平。如果 C2[MODFEN]置位且 C1[SSOE]被清零，则 SS 引脚配置为用于检测模式故障错误的输入。如果 SS 输入变为低电平，则表示存在模式故障错误（另一个主机试图驱动 MOSI 和 SPSCK 线路）。在该情况下，SPI 通过清零 C1[MSTR]立即切换到从机模式，并且还禁用从机输出缓冲区 MISO（双向模式下为 SISO）。结果，所有输出都禁用，SPSCK、MOSI 和 MISO 均为输入。如果在发生模式故障时有传输在进行，那么该传输会中止并且 SPI 会强制进入空闲状态。该模式故障错误会置位 SPI 状态寄存器 (SPIx_S)中的模式故障(MODF)标志。如果 S[MODF]置位时 SPI 中断使能位(SPIE)置位，则随后还会请求 SPI 中断序列。对主机中的 SPI 数据寄存器进行写操作时，会有半个 SPSCK 周期的延迟。延迟之后，SPSCK 在主机中启动。传输操作的其余部分略有不同，具体取决于 SPI 控制寄存器 1 中 SPI 时钟相位位 CPHA 指定的时钟格式（参见 SPI 时钟格式 ）。

注:在主机模式下,更改 C1[CPOL]、C1[CPHA]、C1[SSOE]、C1[LSBFE]、C2[MODFEN]、C2[SPC0]、C2[BIDIROE]（C2[SPC0]置位时）、 SPPR2-SPPR0 和 SPR3-SPR0 会中止正在进行的传输并强制使 SPI 进入空闲状态。远程从机无法检测到该情况，因此主机必须确保远程从机的状态被设置回空闲状态。

● 7.2.8.3 从机模式

SPI 在 SPI 控制寄存器 1 中的 MSTR 位被清零时以从机模式工作。

- SPSCK

在从机模式下，SPSCK 是来自主机的 SPI 时钟输入。

• MISO、MOSI 引脚

在从机模式下，串行数据输出引脚(MISO)和串行数据输入引脚(MOSI)的功能由 SPI 控制寄存器 2 中的 SPC0 位和 BIDIROE 位确定。

• SS 引脚

SS 引脚是从机选择输入。在发生数据传输之前，从机 SPI 的 SS 引脚必须为低电平。SS 在传输完成之前必须一直保持低电平。 如果 SS 变为高电平，SPI 会强制进入空闲状态。

SS 输入还控制串行数据输出引脚。如果 SS 为高电平（未选中），则串行数据输出引脚为高阻抗。如果 SS 为低电平，则 SPI 数据寄存器中的首位会从串行数据输出引脚驱动输出。 此外，如果未选中从机（SS 为高电平），那么就会忽略 SPSCK 输入并且 SPI 移位寄存器不会发生内部移位。

尽管 SPI 能够进行双工操作，但某些 SPI 外设在从机模式下只能接收 SPI 数据。这些更简单的器件没有串行数据输出引脚。

注:使用具有双工能力的外设时，切勿同时使能如下的两个接收器：二者的串行输出驱动同一系统从机的串行数据输出线路。

只要不是一个以上的从器件驱动系统从机的串行数据输出线路，多个从机就能从一个主机接收同一数传输，不过主机不会从所有接收从机接收到返回信息。

如果 SPI 控制寄存器 1 中的 CPHA 位处于清零状态，那么 SPSCK 输入上的奇数边沿会使串行数据输入引脚处的数据进入锁存状态。偶数边沿使之前从串行数据输入引脚锁存的值移入 SPI 移位寄存器的最低有效位或最高有效位，具体情况取决于 LSBFE 位。

如果 C1[CPHA]置位，那么 SPSCK 输入上的偶数边沿会使串行数据输入引脚上的数据进入锁存状态。奇数边沿使之前从串行数据输入引脚锁存的值移入 SPI 移位寄存器的最低有效位或最高有效位，具体情况取决于 C1[LSBFE]。如果 C1[CPHA]置位，第一个边沿用于将第一数据位移到串行数据输出引脚上。当 C1[CPHA]处于清零状态且 SS 输入为低电平（已选择从机）时，SPI 数据的首位从串行数据输出引脚驱动输出。移出第 8 位数据后，认为传输完成，接收数据转移到 SPI 数据寄存器。为了指示传输完成，SPI 状态寄存器中的 SPRF 标志置位。

注:在从机模式下，更改 C2[BIDIROE]（C2[SPC0]置位时）、C1[CPOL]、C1[CPHA]、C1[SSOE]、C1[LSBFE]、C2[MODFEN]和 C2[SPC0]位会破坏正在进行的传输，必须避免此类情况。

● 7.2.8.4 SPI 时钟格式

为了支持不同制造商的各种同步串行外设，SPI 系统的控制寄存器 1 有一个时钟极性(CPOL)位和一个时钟相位(CPHA)控制位，用于选择四种时钟格式中的一种进行数据传输。C1[CPOL]可视情况选择插入一个与时钟串联的反相器。C1[CPHA]选择时钟与数据之间的两种不同时钟相位关系中的一种。

下图所示为 CPHA = 1 时的时钟格式。图顶部显示的是供参考的 8 位时间,其中,位 1 开始于第一个 SPSCK 边沿，位 8 结束于第八个 SPSCK 边沿后的半个 SPSCK 周期。最高有效位优先和最低有效位优先行显示 SPI 数据位的顺序，它取决于 LSBFE 的设置。图中显示了 SPSCK 极性的两种形态，但对于特定传输，只有一种波形适用，具体取决于 C1[CPOL]中的值。SAMPLE IN 波形适用于从机的 MOSI 输入或主机的 MISO 输入。MOSI 波形适用于主机的 MOSI 输出引脚，MISO 波形适用于从机的 MISO 输出。$\overline{SS}$ OUT 波形适用于主机的从机选择输出（前提是 C2[MODFEN]和 C1[SSOE] = 1）。主机 SS 输出在传输开始前的半个 SPSCK 周期变为低电平有效，在传输的第八位时间结束时变回高电平。$\overline{SS}$ IN 波形适用于从机的从机选择输入。

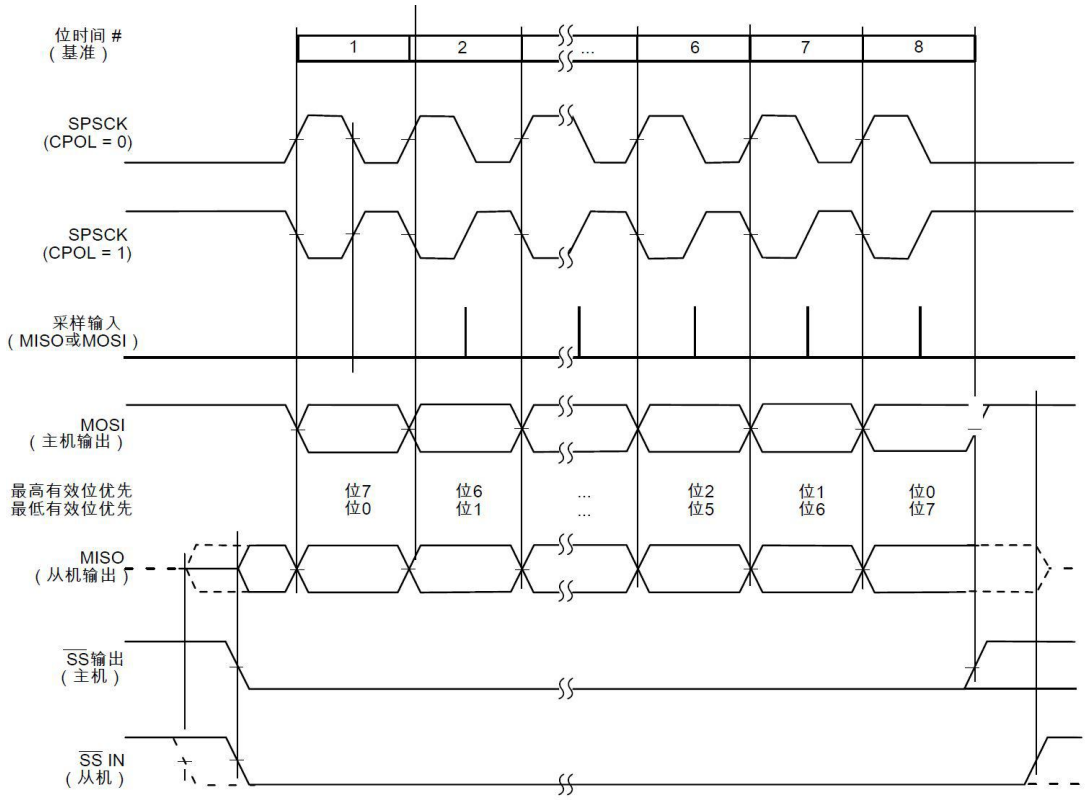


图 7-2 SPI 时钟格式(CPHA=1)

当 C1[CPHA]=1 时，从机在 SS 变为低电平有效时开始驱动其 MISO 输出，但数据要等到第一个 SPSCK 边沿后才定义。第一个 SPSCK 边沿将数据的第一位从移位器移出到主机的 MOSI 输出和从机的 MISO 输出上。在下一个 SPSCK 边沿，主机和从机分别对 MISO 和 MOSI 输入上的数据位值进行采样。在第三个 SPSCK 边沿，SPI 移位器移动一个位位置，以便移入刚刚采样的位值，并将第二数据位值从移位器的另一端移出到主机的 MOSI 输出和从机的 MISO 输出。

当 C1[CPHA] = 1 时，从机的 SS 输入在两次传输之间无需变为高电平无效状态。采用该时钟格式时，可能会发生背靠背传输，如下所述：

1. 一个传输正在进行。
2. 一个新数据字节在正在进行的传输完成之前被写入发送缓冲区。
3. 正在进行的传输完成后，新的已就绪数据会被立即发送。

在这两次连续传输之间，无需插入暂停；$\overline{SS}$引脚保持低电平。

下图所示为 C1[CPHA] = 0 时的时钟格式。图顶部显示的是供参考的 8 位时间，其中，位 1 在从机被选中时（SS 输入变为低电平）开始，位 8 结束于最后一个 SPSCK 边沿。最高有效位优先和最低有效位优先行显示 SPI 数据位的顺序，它取决于 LSBFE 的设置。图中显示了 SPSCK 极性的两种形态，但对于特定传输，只有一种波形适用，具体取决于 CPOL 中的值。SAMPLE IN 波形适用于从机的 MOSI 输入或主机的 MISO 输入。MOSI 波形适用于主机的 MOSI 输出引脚，MISO 波形适用于从机的 MISO 输出。SS OUT 波形适用于主机的从机选择输出（前提是 C2[MODFEN]和 C1[SSOE] = 1）。主机 SS 输出在传输的第一位时间开始时变为低电平有效，在传输的第八位时间结束后的半个 SPSCK 周期变回高电平。SS IN 波形适用于从机的从机选择输入。

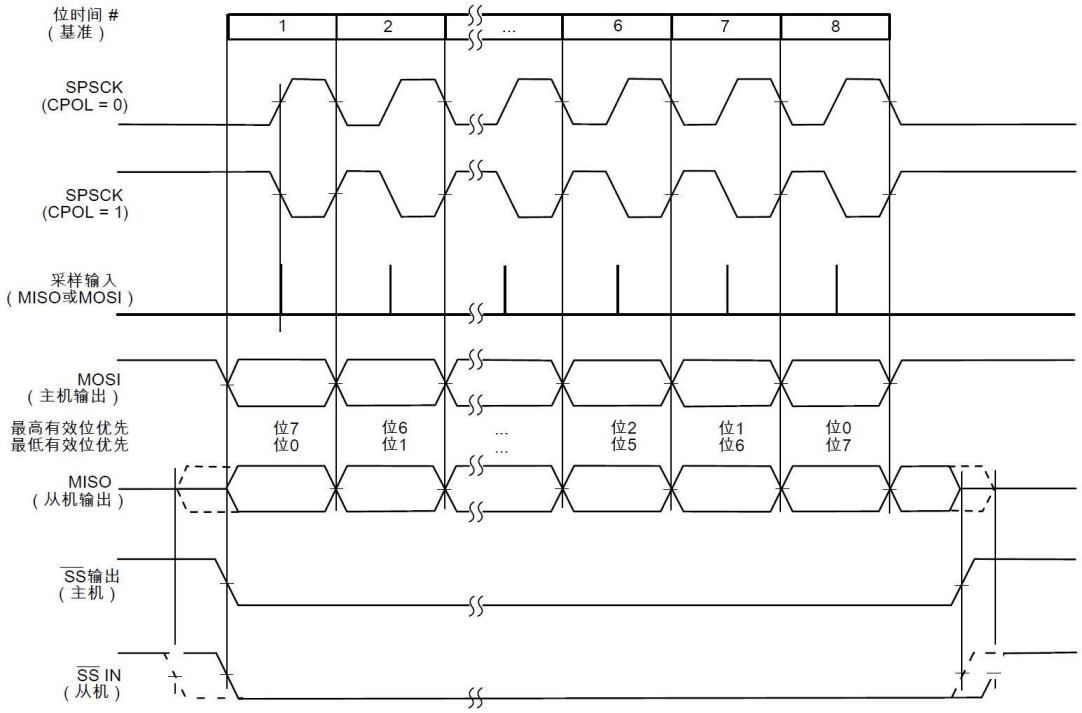


图 7-3 SPI 时钟格式(CPHA=0)

当 C1[CPHA] = 0 时，从机在 SS 变为低电平有效时开始用第一数据位值（最高有效位或最低有效位，具体取决于 LSBFE）驱动其 MISO 输出。在第一个 SPSCK 边沿，主机和从机分别对 MISO 和 MOSI 输入上的数据位值进行采样。在第二个 SPSCK 边沿，SPI 移位器移动一个位位置，以便移入刚刚采样的位值，并将第二数据位值从移位器的另一端移出到主机的 MOSI 输出和从机的 MISO 输出。当 C1[CPHA] =0 时，从机的 SS 输入在两次传输之间必须变为高电平无效状态。

● 7.2.8.5 SPI 波特率生成

如下图所示，SPI 波特率发生器的时钟源是总线时钟。三个预分频位(SPPR2:SPPR1:SPPR0)选择预分频因子 1、2、3、4、5、6、7 或 8。三个速率选择位(SPR3:SPR2:SPR1:SPR0)对预分频器级的输出进行 2、4、8、16、32、64、128、256 或 512 分频，以获得内部 SPI 主机模式比特率时钟。

波特率发生器仅当 SPI 处于主机模式且发生串行传输时才被激活。其他情况，分频器禁用以降低 I_{DD} 电流。

波特率因子等式如下所示（“SPI 波特率因子”表中的保留组合除外）。

$$BaudRateDivisor = (SPPR + 1) \times 2^{(SPR + 1)}$$

波特率可用下列等式进行计算：

$$BaudRate = BusClock / BaudRateDivisor$$

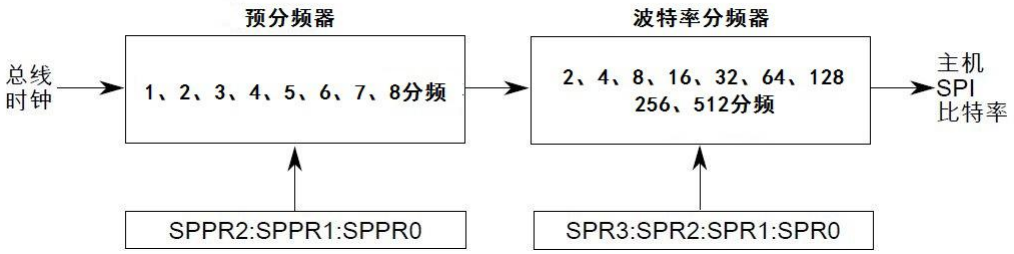


图 7-4 SPI 波特率生成

● 7.2.8.6 SPI 特殊功能

以下各节说明 SPI 模块的其他特殊功能。

■ 7.2.8.6.1 NSS 输出

SS输出在传输期间自动将SS引脚驱动为低电平以选择外部器件，在空闲期间将SS引脚驱动为高电平以取消选择外部器件。选择SS输出时，SS 输出引脚连接到外部器件的SS 输入引脚。

在 SPI 正常操作期间，SS输出仅在主机模式下可用；要使能该功能，请按照 C1[SSOE]的说明将 C1[SSOE]和 C2[MODFEN]的电平变为有效值。使能 SS 输出时，模式故障功能禁用。

注：在多主机系统中使用SS输出功能时应小心，因为无法利用模式故障功能来检测主机之间的系统错误

■ 7.2.8.6.2 双向模式（MOMI 或 SISO）

选择双向模式是在 SPI 控制寄存器 2 中的 SPC0 位置位时（参见下表）。在该模式下，SPI 仅使用一个串行数据引脚与一个或多个外部器件接合。C1[MSTR]决定使用哪个引脚。MOSI 引脚成为串行数据 I/O（MOMI）引脚（适用于主机模式），MISO 引脚成为串行数据 I/O（SISO）引脚（适用于从机模式）。SPI 不使用主机模式下的 MISO 引脚和从机模式下的 MOSI 引脚。

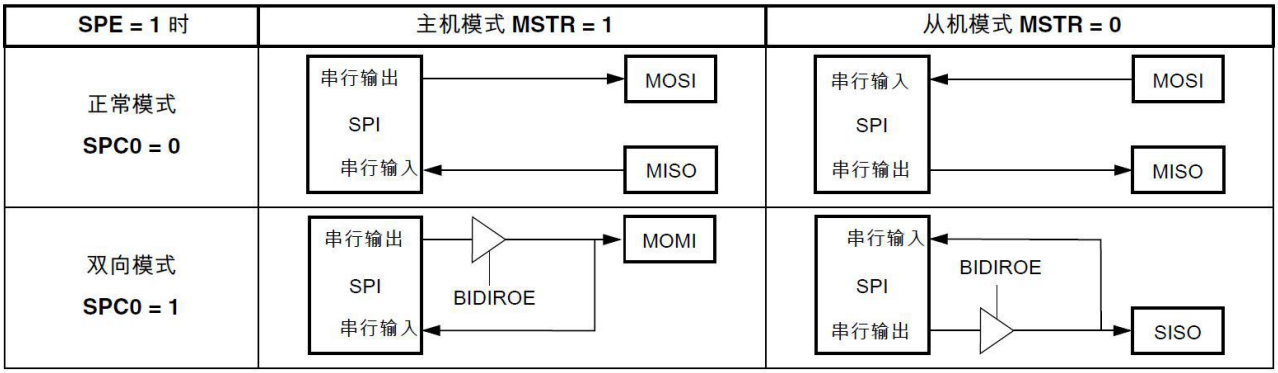


图 7-5 正常模式和双向模式

各串行 I/O 引脚的方向取决于 C2[BIDIROE]。如果该引脚配置为输出，则移位寄存器的串行数据从该引脚驱动输出。该引脚也是移位寄存器的串行输入。SPSCK 对于主机模式是输出，对于从机模式是输入。SS对于主机模式是输入或输出，但对于从机模式始终是输入。双向模式不影响 SPSCK 功能和SS功能。

注：处于双向主机模式且模式故障特性使能时，MISO 和 MOSI 这两个数据引脚都可供 SPI 占用，不过 MOSI 常用于双向模式下的传输且 SPI 不使用 MISO。 如果发生模式故障，SPI 会自动切换至从机模式。在该情况下，MISO 变为由 SPI 占用且 SPI 不使用 MOSI。如果 MISO 引脚另作他用，则应考虑这种情形。

● 7.2.8.7 错误条件

SPI 模块有一个错误条件：模式故障错误。

■ 7.2.8.7.1 模式故障错误

如果 SPI 配置为主机的同时$\overline{SS}$输入变为低电平，那么表示存在系统错误（有一个以上的主机可能在同时尝试驱动 MOSI 和 SPSCK 线路）。 这种情况在正常工作中是不允许的，在 C2[MODFEN]置位的情况下，它会使 SPI 状态寄存器中的 MODF 位自动置位。

在 SPI 处于主机模式且 C2[MODFEN]被清零的特殊情况下，SPI 不使用$\overline{SS}$引脚。此时，模式故障错误功能禁用，MODF 保持被清零状态。如果 SPI 系统配置为从机，那么$\overline{SS}$引脚是专用输入引脚。 模式故障错误在从机模式下不会发生。

发生模式故障错误时，SPI 会切换到从机模式，但从机输出缓冲区禁用的情况除外。因此，SPSCK、MISO 和 MOSI 引脚会强制变为高阻抗输入，以免与另一输出驱动器发生任何可能的冲突。正在进行的传输会中止并且 SPI 会强制进入空闲状态。

对于在主机模式下配置的 SPI 系统，如果在双向模式下发生模式故障错误，那么 MOMI（双向模式下的 MOSI）的输出使能会被清零（若之前已置位）。对于在从机模式下配置的 SPI 系统，双向模式下不会发生任何模式故障错误。

将模式故障标志自动清零的方法是：先对 SPI 状态寄存器（MODF 置位时）进行读操作，再对 SPI 控制寄存器 1 进行写操作。将模式故障标志清零后，SPI 会再次变为正常主机或从机。

● 7.2.8.8 低功耗模式选项

本节说明低功耗模式选项。

■ 7.2.8.8.1 运行模式下的 SPI

在运行模式下，当 SPI 控制寄存器 1 中的 SPI 系统使能(SPE)位处于清零状态时，SPI 系统处于低功耗禁用状态。SPI 寄存器仍处于可访问状态，但该模块内核的时钟被禁用。

■ 7.2.8.8.2 等待模式下的 SPI

SPI 在等待模式下的操作取决于 SPI 控制寄存器 2 中 SPISWAI 位的状态。

- 如果 C2[SPISWAI]处于清零状态，则当 CPU 处于等待模式时，SP 正常工作。
- 如果 C2[SPISWAI]置位，当 CPU 处于等待模式时，SPI 时钟发生器停止工作，SPI 模块进入降耗状态。
  - 如果 C2[SPISWAI]置位且 SPI 针对主机进行配置，那么任何正在进行的传输和接收都会在进入等待模式时停止。SPI 退出等待模式时，传输和接收恢复。
  - 如果 C2[SPISWAI]置位且 SPI 配置为从机，那么任何正在进行的传输和接收都会继续进行（如果主机继续驱动 SPSCK）。这就使从机保持与主机和 SPSCK 的同步。

    如果主机在从机处于等待模式的同时发送数据，那么从机会继续发送与等待模式开始时的工作模式一致的数据(也就是说,如果从机当前正在向主机发送 SPIx_D,则它会继续发送该字节。 否则，如果从机当前正在发送从主机接收到的最后数据字节，则它会继续发送先前从主机接收的各数据字节）。

注:从机处于等待模式或停止模式（外设总线时钟停止但内部逻辑状态得以保留）时，如果预期主机会提供数据，则必须小心处理。即使移位寄存器继续工作，但 SPI 的其余部分也会被关断（也就是说，SPRF 中断要等到退出停止或等待模式后才会生成）。此外，移位寄存器的数据要等到从机 SPI 退出等待或停止

模式后才复制到 SPIx_D 寄存器中。只有在传输期间进入或退出等待模式时，才会生成 SPRF 标志和 SPIx_D 复制。 如果从机在空闲模式下进入等待模式且在空闲模式下退出等待模式，那么 SPRF 和 SPIx_D 复制都不会发生。

■ 7.2.8.8.3 停止模式下的 SPI

停止模式（外设总线时钟停止但内部逻辑状态得以保留）下的操作取决于 SPI 系统。停止模式与 C2[SPISWAI]无关。 一进入此类停止模式，SPI 模块时钟就会被禁用（保持在高电平或低电平）。

• 如果 SPI 在 CPU 进入停止模式时处于主机模式且正在交换数据，那么传输在 CPU 退出停止模式之前会一直被冻结。退出停止模式后，与外部 SPI 的数据交换正常进行。

• 在从机模式下，SPI 保持与主机同步。

在停止模式（外设总线时钟停止且不保留内部逻辑状态）下，SPI 完全禁用。退出此类停止模式后，所有寄存器都被复位到其默认值，并且必须重新初始化 SPI 模块。

● 7.2.8.9 复位

寄存器和信号的复位值在“存储器映像”和“寄存器说明”内容中均有介绍，其详细介绍了寄存器及其位字段。

• 如果复位后在从机模式下发生数据传送且未写入 SPIx_D，则此次传送包含“无用资料”或是在复位前最后一次从主机接收的数据。

• 复位后从 SPIx_D 读取始终会归零。

● 7.2.8.10 中断

SPI 仅在使能时（SPIx_C1 寄存器中的 SPE 位置位）发起中断请求。 以下是关于 SPI 如何发出请求以及 MCU 应如何应答该请求的描述。中断向量偏移和中断优先权和芯片有关。

与 SPI 系统相关的有四个标志位、三个中断屏蔽位和一个中断矢量。SPI 中断使能屏蔽(SPIE)可从 SPI 接收器完整标志 (SPRF) 和模式故障标志(MODF)处使能中断。SPI 传输中断使能屏蔽(SPTIE)可从 SPI 传送缓冲器空标志(SPTEF)处使能中断。SPI 匹配中断使能屏蔽位(SPIMIE)可从 SPI 匹配标志(SPMF)处使能中断。其中某个标志位置位且相关中断屏蔽位也置位时，会向 CPU 发送一个硬件中断请求。如果中断屏蔽位已清除，则软件可对相关标志位进行轮询，而非使用中断。SPI 中断服务例程(ISR)应检查标志位以确定引起中断的事件。服务例程在从 ISR 返回之前（通常接近 ISR 开始时）还应当清除标志位。

■ 7.2.8.10.1 MODF

模式故障(MODF)在主机检测到$\overline{SS}$引脚出错时发生。主机 SPI 必须针对 MODF 特性进行配置(参见 C1[SSOE]位的说明）。一旦 MODF 置位，当前传输就会中止，SPIx_C1 寄存器中的主机(MSTR)位复位到 0。

MODF 中断在状态寄存器的 MODF 标志中反映。清除该标志也会清除该中断。该中断在 MODF 标志置位期间保持有效状态。MODF 有一个自动清除流程，具体请参见“SPI 状态寄存器”章节中的说明。

■ 7.2.8.10.2 SPRF

SPIF 在新数据已被接收并复制到 SPI 接收数据缓冲区时发生。SPRF 置位后，对其进行修改之前不会进

行清零。SPRF 有一套自动清零流程，这在“SPI 状态寄存器”一节有详细说明。如果下一次传输结束之前 SPRF 仍未得到修正（即在另一次传输过程中 SPRF 仍有效），则后续传输将被忽略而且不会有新数据复制到数据寄存器。

■ 7.2.8.10.3 SPTEF

SPTEF 在 SPI 传送缓冲区准备接受新数据时发生。

SPTEF 置位后，对其进行修改之前不会进行清零。SPTEF 有一套自动清零流程，这在 SPI 状态寄存器中有详细说明。

■ 7.2.8.10.4 SPMF

SPMF 在接收数据缓冲区中的数据等于 SPI 匹配寄存器中的数据时发生。

■ 7.2.8.10.5 低功耗模式下的异步中断

CPU 处于等待模式或停止模式且 SPI 模块接收传输时，SPI 模块可生成一个异步中断将 CPU 从低功耗模式中唤醒。该模块仅在以下条件全部得到满足时生成异步中断：

1. C1[SPIE]置 1。

2. CPU 处于等待模式（C2[SPISWAI]在该情况下必须为 1）或处于停止模式（外部总线时钟停止但内部逻辑状态得以保留）。

3. SPI 模块处于从机模式。

4. 收到的传输结束。

当中断唤醒 CPU 且外设总线时钟再次处于活动状态时，此 SPI 模块会将接收到的数据从移位器中复制到数据寄存器中并生成标志信号。在唤醒阶段，如果从主机连续不断地传输，则会破坏首次接收到的数据。

■ 7.2.9 初始化/应用信息

本节讨论如何初始化和使用 SPI 的例子。

● 7.2.9.1 初始化序列

在 SPI 模块可用于通信之前，必须执行如下的初始化程序：

1. 更新控制寄存器 1 (SPIx_C1)以使能 SPI 并控制中断使能。该寄存器还可将 SPI 设置为主机或从机，确定时钟相位和极性，以及配置 SPI 主要选项。

2. 更新控制寄存器 2 (SPIx_C2)以使能 SPI 匹配中断特性等其他 SPI 功能、主机模式故障功能和双向模式输出，以及控制和其他可选特性。

3. 更新波特率寄存器(SPIx_BR)以设置 SPI 主机的预分频器和比特率因子。

4. 用要与接收数据寄存器作比较的值更新硬件匹配寄存器(SPIx_M)，以便在硬件匹配中断使能时触发中断。

5. 在主机中，在 S[SPTEF] = 1 时对 SPIx_S 进行读操作，然后写入发送数据寄存器(SPIx_D)以开始传输。

● 7.2.9.2 伪代码示例

在此示例中，建立适合主机模式的 SPI 模块，且只使能硬件匹配中断。SPI 在以 2 分频总线时钟的最大波特率运行。时钟相位和极性针对高电平有效 SPI 时钟而设置，其中 SPSCK 上的第一条边沿在首个数据传输周期开始时出现。

SPIx_C1 = 0x54(%01010100)			
位 7	SPIE	= 0	禁用接收和模式故障中断
位 6	SPE	= 1	启用 SPI 系统
位 5	SPTIE	= 0	禁用 SPI 传送中断
位 4	MSTR	= 1	将 SPI 模块设置为 SPI 主器件
位 3	CPOL	= 0	将 SPI 时钟配置为高电平有效
位 2	CPHA	= 1	SPSCK 上的第一条边沿在首个数据传输周期开始时出现
位 1	SSOE	= 0	使能模式故障时确定 SS 引脚功能
位 0	LSBFE	= 0	SPI 串行数据传输从最高有效位开始

SPIx_C2 = 0x80(%10000000)			
位 7	SPIE	= 1	SPI 硬件匹配中断已使能
位 6		= 0	未执行
位 5		= 0	保留
位 4	MODFEN	= 0	禁用模式故障功能
位 3	BIDIROE	= 0	SPI 数据 I/O 引脚用作输入
位 2		= 0	保留
位 1	SPISWAI	= 0	SPI 时钟在等待模式下运行
位 0	SPC0	= 0	使用适合数据输入和输出的单独引脚

SPIx_BR = 0x00(%00000000)		
位 7	= 0	保留
位 6:4	= 000	将预分频器除数设置为 1
位 3:0	= 0000	将波特率除数设置为 2

SPIx_S = 0x00(%00000000)			
位 7	SPRF	= 0	标志在接收数据缓冲区满时置位
位 6	SPMF	= 0	标志在 SPIx_M = 接收数据缓冲区时置位
位 5	SPTEF	= 0	标志在传送数据缓冲区满时置位
位 4	MODF	= 0	适合主机模式的模式故障标志
位 3:0		= 0	保留

SPIx_M = 0xXX
占用硬件匹配缓冲区的位 0 到位 7

SPIx_D = 0xxx
占用由传送缓冲区传送并由接收缓冲区接收数据的位 0 到位 7

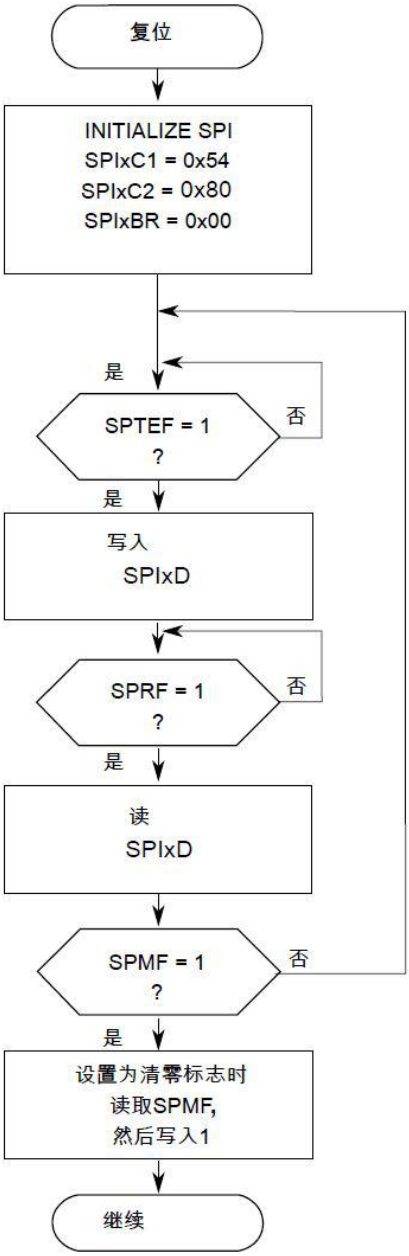


图 7-6 SPI 主器件的初始化流程图示例

■ 7.3 I2C 总线模块

■ 7.3.1 简介

I^2C（也有简写为 IIC、I2C 的，为了和演示程序兼容，本文档后面都简写为 I2C）是 Inter-Integrated circuit 的简写，它可以在多个芯片之间提供一种通讯方式。

NV32 的 I2C 的设计目标是在最大总线负载下能达到 100kbps 的速率。通过减小总线负载，可以提高 I2C 设备的操作速度，最高达到时钟的二十分之一。通讯的最长距离和可连接的设备数量，受限于 400pF 的最大总线负载要求。NV32 的 I2C 模块也符合版本 2.0 的 SMBus 总线规范。

本文档还采用如下的专用术语：

SCL：I2C 总线的时钟信号线

SDA：I2C 总线的数据线

START ：I2C 的起始(START)位。当 SCL 为高期间，SDA 由高到低的跳变被定义为 I2C 传输的起始位。

STOP：I2C 的停止（STOP）位。当 SCL 为高期间，SDA 由低到高的跳变被定义为 I2C 传输的停止位。START 和 STOP 是一种电平跳变序列，而不是一个电平信号。

RE-START：I2C 中的重复起始位（Repeated START）。通常的 I2C 通讯是以 START 开始，以 STOP 结束，而 RE-START 是在一次 I2C 通讯从 START 开始后，STOP 之前，又发起了一次 START。

ACK 和 NACK：I2C 中的响应和不响应。I2C 通讯中，接收数据的设备用低电平通知对方数据正确接收，即 ACK；用高电平通知对方数据接收错误或者结束数据传输，即 NACK。

置位：设置为 1 的代名词。

清零：设置为 0 的代名词。

■ 7.3.2 特性

NV32 的 I2C 模块具有以下特性：

- 兼容 I2C 总线规范
- 支持多主机操作
- 支持串行时钟频率 64 值可软件编程
- 可软件选择的应答位
- 中断驱动的逐字节数据传输模式
- 从主设备切换为从设备的自动模式切换时产生仲裁丢失中断
- 广播地址识别中断
- 产生和检测 START、STOP
- 产生和检测 RE-START
- 产生和检测 ACK/NACK
- 检测总线忙状态（空闲状态）
- 识别通用广播
- 支持十比特地址扩展
- 支持 Version2 的 SMBus 总线规范
- 输入毛刺滤波宽度可编程
- 低功耗模式下从地址匹配时可唤醒
- 支持从地址范围选择
- 支持运行模式、等待模式、停止模式等三种不同的低功耗模式

■ 7.3.3 操作模式

NV32 支持运行模式、等待模式、停止模式等三种不同的低功耗模式，在这些模式下，NV32 的 I2C 模块的操作情况如下：

运行模式：这是 NV32 的基本操作模式。在运行模式下需要降低功耗，需要关闭整个 I2C 模块的功能。

等待模式：NC32 的内核处于等待模式时，I2C 模块依然处于工作状态，同时可提供唤醒中断。

停止模式：在内核处于停止模式时，除非使能了地址匹配模式，否则 I2C 模块也处于休眠状态以降低功耗。内核停止指令并不影响 I2C 模块的寄存器状态。

■ 7.3.4 结构框图

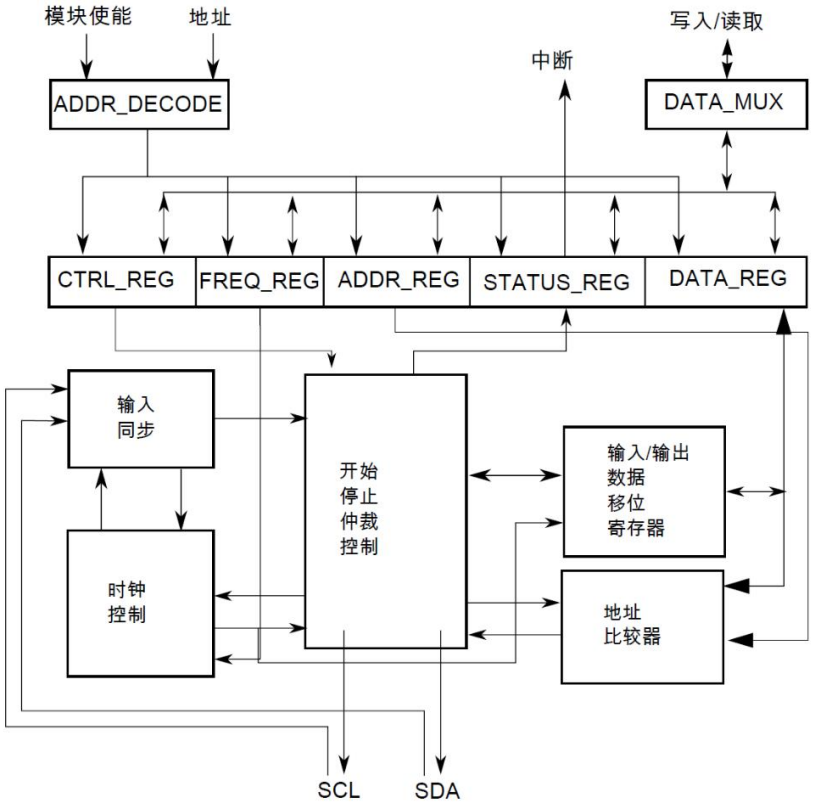


图 7-7 I2C 结构框图

■ 7.3.5 引脚说明

I2C 模块有两种引脚配置，需要对 SIM 模块的相关寄存器进行配置，具体请参考 SIM 寄存器说明。

■ 7.3.6 I2C 信号描述

I2C 的信号组成如下表所示：

表 7-9 I2C 信号描述

信号	描述	类型	引脚
SCL	I2C 系统的双向串行时钟线	I/O	PA3/PB7
SDA	I2C 系统的双向串行数据线	I/O	PA2/PB6

■ 7.3.7 存储器和寄存器说明

本节详细描述了所有的 I2C 寄存器访问的最终用户。

表 7-10 I2C 寄存器映射

绝对地址 (16 进制)	寄存器名称	偏移量	位宽	权限	复位值
4006_6000h	I2C 地址寄存器 1（I2C0_A1）	0h	8	R/W	00h
4006_6001h	I2C 分频寄存器（I2C0_F）	1h	8	R/W	00h
4006_6002h	I2C 控制寄存器 1（I2C0_C1）	2h	8	R/W	00h
4006_6003h	I2C 状态寄存器（I2C0_S）	3h	8	R/W	80h
4006_6004h	I2C 收发数据寄存器（I2C0_D）	4h	8	R/W	00h
4006_6005h	I2C 控制寄存器 2（I2C0_C2）	5h	8	R/W	00h
4006_6006h	I2C 可编程输入毛刺滤波寄存器（I2C0_FLT）	6h	8	R/W	00h
4006_6007h	I2C 地址范围寄存器（I2C0_RA）	7h	8	R/W	00h
4006_6008h	I2C SMBus 控制和状态寄存器（I2C0_SMB）	8h	8	R/W	00h
4006_6009h	I2C 地址寄存器 2（I2C0_A2）	9h	8	R/W	C2h
4006_600Ah	I2C 低电平超时寄存器(高字节)（I2C0_SLTH）	Ah	8	R/W	00h
4006_600Bh	I2C 低电平超时寄存器(低字节)（I2C0_SLTL）	Bh	8	R/W	00h

● 7.3.7.1 I2C 地址寄存器 1（I2Cx_A1）

此寄存器包含 I2C 模块的从机地址。

地址：4006_6000h（基址）+ 0h（偏移量）= 4006_6000h

位	7	6	5	4	3	2	1	0
读 写	AD[7:1]							0
复位值	0	0	0	0	0	0	0	0

表 7-11 I2Cx_A1 字段描述

位	描述
7-1 AD[7:1]	地址 在 I2C 模块设置为从机时，该字段为从机基本地址，该字段用于 7 位地址方案以及 10 位地址方案中的七个低位
0	这个字段被保留。 该字段为保留字段。为只读寄存器，输出值为 0。。

● 7.3.7.2 I2C 分频寄存器（I2Cx_F）

用于 I2C 波特率的产生。

地址：4006_6000h（基址）+ 1h（偏移量）= 4006_6001h

位	7	6	5	4	3	2	1	0
读 写	MULT				ICR			
复位值	0	0	0	0	0	0	0	0

表 7-12 I2Cx_F 字段描述

位	描述
7-6 MULT	MULT 位定义了乘数因子 MUL。这个因子与 SCL 分频器相结合后生成 I2C 波特率 00 mul=1， 01 mul=2 ,10 mul=4 ， 11 保留
5-0 ICR	时钟速率 对 I2C 模块进行预分频，选择比特率。此字段和 MULT 字段决定了 I2C 波特率、SDA 保持时间、SCL 开始保持时间和 SCL 停止保持时间。有关每项 ICR 设置对应值，请参见 I2C 分频器保持值。 SCL 分频数与 mul 乘数因子相乘的结果决定 I2C 波特率。 I2C baud rate = bus speed (Hz)/(mul × SCL divider) SDA 保持时间是从 SCL（I2C 时钟）的下降沿开始直至 SDA（I2C 数据）发生变化的延迟。 SDA hold time = bus period (s) × mul × SDA hold value SCL 开始保持时间是从 SCL 为高（启动状态）的 SDA（I2C 数据）的下降沿开始到 SCL（I2C 时钟）的下降沿的延迟。 SCL start hold time = bus period (s) × mul × SCL start hold value 在 SCL 停止保持时间是从 SCL（I2C 时钟）的上升沿开始到 SCL 为高（停止条件）的 SDA 的上升沿（I2C 数据）之间的延迟。 SCL stop hold time = bus period (s) × mul × SCL stop hold value

I2C 分频设置与保持时间值列表

MULT	ICR	保持时间(Hold Times) us		
		SDA	SCL Start	SCL Stop
2h	00h	3.500	3.000	5.500
1h	07h	2.500	4.000	5.250
1h	0Bh	2.250	4.000	5.250
0h	14h	2.125	4.250	5.125
0h	18h	1.125	4.750	5.125

● 7.3.7.3 I2C 控制寄存器 1（I2Cx_C1）

控制 I2C 部分功能的打开与关闭。

地址: 4006_6000h（基址）+ 2h（偏移量）= 4006_6002h

位	7	6	5	4	3	2	1	0
读	IICEN	IICIE	MST	TX	TXAK	0	WUEN	0
写						RSTA		
复位值	0	0	0	0	0	0	0	0

表 7-13 I2Cx_C1 字段描述

位	描述
7 IICEN	I2C 使能 使能 I2C 模块操作 0 关闭使能 1 打开使能

6 IICIE	I2C 中断使能 使能 I2C 模块操作 0 关闭使能 1 打开使能
5 MST	主机模式选择 配置 MST 从 0 到 1 时，自动产生 START，然后 I2C 进入主机模式； 配置 MST 从 1 到 0 时，自动产生 STOP，然后 I2C 进入从机模式； 0 设置 I2C 模块为从机模式 1 设置 I2C 模块为主机模式
4 TX	发送模式选择 选择主机和从机的传输方向。在主机模式下，该位设置必须和传输方向一致。因此，对于地址周期，该位始终置 1。当作为从机时该位必须由软件根据状态寄存器的 SRW 位进行设置 0 接收模式 1 发送模式
3 TXAK	发送 ACK 使能 当 I2C 被设置成接收器时，指定驱动数据会在数据确认周期内发送给到 SDA。状态寄存器的 FACK 位的值会影响 NACK/ACK 产生。 注：在 TXAK 被写入前，SCL 一直保持为低电平。 0 向接下来的字节（如果 FACK 已清零）或当期接收字节（如果 FACK 已置位）的总线上发送应答信号。 1 不向接下来的字节（如果 FACK 已清零）或当期接收字节（如果 FACK 已置位）的总线上发送应答信号。
2 RSTA	重复起始位 在主机模式下写 1 到该位产生一个重复起始条件并送给当前主机。但该位值始终读为零。在错误的时间打开循环会导致仲裁丢失。
1 WUEN	唤醒使能 当 I2C 模块进行从地址匹配并且没有外设总线运行时可将 MCU 从低功耗模式下唤醒。 0 正常操作。在低功耗模式下地址匹配无唤醒中断产生。 1 打开在低功耗模式下的唤醒使能
0 保留	该位保留 这个只读域被保留，并且总是具有值 0。

● 7.3.7.4 I2C 状态寄存器（I2Cx_S）

地址：4006_6000h（基址）+ 3h（偏移量）= 4006_6003h

位	7	6	5	4	3	2	1	0
读	TCF	IAAS	BUSY	ARBL	RAM	SRW	IICIF	RXAK
写				w1c			w1c	
复位值	1	0	0	0	0	0	0	0

表 7-14 I2Cx_B 字段描述

位	描述

7 TCF	传输完成标志 当完成一个字节的数据传输和对字节传输做出应答后，该位置 1。该位仅在 I2C 模块即时传输的时候有效。在传输模式中，通过接收模式中对 I2C 数据寄存器的读和发送模式中对 I2C 数据寄存器的写完成 TCF 位的清零。 0 传输正在进行中 1 传输完成
6 IAAS	编址为从机 此位通过下列条件之一置位： • 调用地址与 A1 寄存中已经编程的从机原地址匹配,或者与 RA 寄存中的范围地址匹配(必须设定一个不为零的值) • GCAEN 位被置 1 后收到一个通用调用。 • SIICAEN 被置 1，并且调用地址与第二个已编程的从机地址匹配 • ALERTEN 被置 1，并且接收到收到了 SMBus 报警响应地址 • RMEN 被置 1 后并接收到一个地址，它的值是在 A1 寄存器的值和 RA 寄存器的值之间。 该位在 ACK 位之前设置。 CPU 必须检查 SRW 位，并相应地设置 TX / RX。给 C1 寄存器编写任意值会给该位清零。 0 未编址 1 编址位从机
5 BUSY	总线繁忙标志位 表明总线的状态，无论从机或主机模式。当检测到一个 START 信号该位被设置，当检测到一个 STOP 信号该位清零。 0 总线空闲 1 总线繁忙
4 ARBL	仲裁丢失 注：w1c 表示写 1 该位就会清零，写 0 无效。 当仲裁程序丢失该位会被硬件设置 1。ARBL 位必须由软件置 1 清零 0 标准总线操作 1 仲裁丢失
3 RAM	范围地址匹配 该位会在以下任意条件下置 1： • 收到与 RA 寄存器的地址匹配的任何非零呼叫地址。 • RMEN 位被设置并且呼叫地址在 A1 和 RA 寄存器值的范围之内。 0 无寻址 1 从机寻址
2 SRW	从机读/写标志位 当 I2C 模块作为从机时，SRW 表示呼叫地址发送到主机的 R/W 命令值。 0 从机接收，主机写给从机 1 从机发送，主机向从机读取
1 IICIF	中断标志位 有 I2C 中断产生时此位置 1。由软件写 1 才可以将该位清零，比如在中断例程中。下面任意一个事件都会将此位置 1： • FACK 为 0 的情况下，完成一个字节传输，包括 ACK / NACK 位。在处于接收模式该位被设置的情况下，通过写 0 或 1 发给 TXAK 的方式将 ACK 和 NACK 传给总线。

	• FACK 为 0 的情况下，完成一个字节的传输，但不包括 ACK / NACK 位。 • 从机地址与调用地址的匹配主要包括从机地址、范围从机地址、提醒响应地址、第二地址或通用调用地址 • 仲裁丢失 • 在 SMBus 模式下，除了 SCL 和 SDA 高电平超时的任何超时事件。 • 如果在输入毛刺滤波器寄存器中的 SSIE 位为 1，I2C 总线停止或启动检测。 注：要清除 I2C 总线停止或启动检测中断：在中断服务程序中，首先清除输入毛刺滤波器寄存器中的 STOPF 或 STARTF 位，通过写 1 到该位，即可清除 IICIF 位。如果此顺序相反，对 IICIF 位再次声明。 0 无中断产生 1 有中断产生
0 RXAK	接收 ACK 标志位 0 在总线上完成一个字节的传输后，ACK 信号被接收。 1 未检测到 ACK 信号

● 7.3.7.5 I2C 收发数据寄存器（I2Cx_D）

数据寄存器主要用于存储接收的数据或准备需要发送的数据。

地址：4006_6000h（基址）+ 4h（偏移量）= 4006_6004h

位	7	6	5	4	3	2	1	0
读 写	DATA							
复位值	0	0	0	0	0	0	0	0

表 7-15 I2Cx_D 字段描述

字段	描述
7—0 DATA	数据 在主机发送模式下，当数据被写入该寄存器，数据传输开始，最高位首先发送。在主机接收模式下，读该寄存器会启动接收下一个字节数据。即:写寄存器发送，读寄存器接收。 注意：需要把 I2C 模块从主机接收模式退出时，应该在读 I2Cx_D 寄存器前切换 I2C 模式，这样可以避免不小心触发主机接收模式下的数据接收过程。 在从模式下，地址匹配完成后也可以实现相同的功能，（即“写该寄存器触发字节发送，读该寄存器触发字节接收”）。 在传输开始前 C1[TX]位必须正确反映主从模式传输所需方向。例如，I2C 模块配置为主机发送模式，但是主机却收到“接收”命令，这时读取数据寄存器就不会启动接收。 当 I2C 模块配置为主接收或从机接收模式，读取数据寄存器会返回最后接收到的字节。数据寄存器并不反映 I2C 发送的每个字节，不允许软件通过读寄存器方式来验证某个字节是否正确的写入到数据寄存器。 在 I2C 模块被设置为主机发送模式时，在产生 START 之后，或者产生 RE-START 之后，写入寄存器 I2Cx_D 的第一个字节将用作地址字节，因此其值必须是： 高 7 位为呼叫地址（从机设计的基本地址）； 最低位为需要的 R/W 位（之后向从机写数据时该位为 0，从从机读数据时该位为 1）。

● 7.3.7.6 I2C 控制寄存器 2（I2Cx_C2）

地址：4006_6000h（基址）+ 5h（偏移量）= 4006_6005h

位	7	6	5	4	3	2	1	0
读 写	GCAEN	ADEXT	0	SBRC	RMEN	AD[10:8]		
复位值	0	0	0	0	0	0	0	0

表 7-16 I2Cx_C2 字段描述

字段	描述
7 GCAEN	通用地址启动使能 0 使能关闭 1 打开使能
6 ADEXT	地址扩展 该位控制从机地址的位宽 0 7 位地址方案 1 10 位地址方案
5 保留	该位保留 该只读位保留并一直有值 0
4 SBRC	从机波特率控制位 为了让时钟延长在非常快的 I2C 模式下运行，必须把独立从机模式波特率设置在最高频率。举一个快模式的例子，当主机传输 40kbit / s 的数据时，但是从机只能捕捉 10 kbit / s 的主服务器数据。 0 从机波特率跟随主机波特率变化，时钟延长可能会出现 1 从机波特率独立于主波特率
3 RMEN	范围地址匹配使能 该位控制从机地址在 A1 和 RA 寄存器之间的值匹配。当该位置 1 时，从机地址匹配值大于 A1 寄存器和小于等于 RA 寄存器 0 范围匹配模式关闭。 1 范围匹配模式打开。从机地址在 A1 与 RA 寄存器之间匹配
2-0 AD[10:8]	从机地址 该位是 10 位地址方案中的高三位。该位只有当 ADEXT 位置 1 时有效。

● 7.3.7.7 I2C 可编程输入毛刺滤波寄存器（I2Cx_FLT）

地址：4006_6000h（基址）+ 6h（偏移量）= 4006_6006h

位	7	6	5	4	3	2	1	0
读 写	SHEN	STOPF w1c	SSIE	STARTF w1c	FLT			
复位值	0	0	0	0	0	0	0	0

表 7-17 I2Cx_FLT 字段描述

字段	描述
7 SHEN	暂缓数据使能位 设置该位暂缓数据传输或接收时进入停止模式。 下面的方案说明了隔离功能： 1.I2C 模块被配置为基本传输设置，而 SHEN 位设置为 1。 2.传输开始。 3.MCU 通知 I2C 模块进入 STOP 模式。 4.完成当前正在传输字节(包括地址和数据字节)。 5.I2C 从机或主机确认字节传输完成，并响应 CPU 进入停止模式的请求。 6.接收 I2C 模块进入停止模式的请求后，MCU 确定是否关断 I2C 模块的时钟。 如果 SHEN 位被置 1，且 I2C 模块处于闲置或禁用状态时，只要 MCU 输入停止模式信号，I2C 模块会立即做出响应，并进入停止模式。 如果 SHEN 位为 0，则进入停止模式而被暂缓的数据后续发送或接收操作将无法继续完成。要恢复 MCU 退出停止模式后的数据传输，软件必须通过重新发送从机地址的方式重新初始化传输。 如果 I2C 控制寄存器的 IICIE 位置 1，MCU 进入停止模式前，系统软件将获得由 I2C 状态寄存器的 TCF 位触发中断后，MCU 从停止模式唤醒。 0 暂缓数据关闭 1 暂缓数据使能
6 STOPF	I2C 总线 STOP 检测标志位 注：w1c 表示写 1 该位就会清零，写 0 无效。 当 I2C 总线 STOP 状态被检测到，该位会被硬件置 1.该位必须通过写 1 方式清零。 0 I2C 无 STOP 发生 1 检测到 I2C 总线 STOP
5 SSIE	I2C 总线 START 和 STOP 状态检测中断使能位 注意：清除 SSIE 中断的方式：在中断服务程序中，首先通过向 STOPF、STARTF 写 1 的方式给 STOPF、STARTF 清零，然后清除 IICIF 寄存器。如果不是按照这个顺序，而是先清除 IICIF 寄存器，再通过向 STOPF、STARTF 写 1 的方式给 STOPF、STARTF 清零，则 IICIF 将会被再次置位。 0 START 和 STOP 状态检测中断关闭 1 START 和 STOP 状态检测中断使能
4 STARTF	I2C 总线 START 检测标志 当 I2C 总线 START 被检测到该位硬件置 1。该位必须通过写 1 方式清零。
3-0 FLT	I2C 可编程过滤因子 该位控制过滤宽度，以总线时钟周期为单位，不大于 FLT 个单位的脉冲会被当作毛刺处理，从而被该滤波器过滤掉。 0 无过滤 n ：取值从 1 到 15 ，表示内部滤波器将滤除小于或等于 n 个 I2C 模块时钟周期的脉冲信号。

● 7.3.7.8 I2C 地址范围寄存器（I2Cx_RA）

地址：4006_6000h（基址）+ 7h（偏移量）= 4006_6007h

位	7	6	5	4	3	2	1	0

读 写	RAD							0
复位值	0	0	0	0	0	0	0	0

表 7-18 I2Cx_RA 字段描述

字段	描述
7-1 RAD	范围从机地址 此字段包含 I2C 模块使用的从机地址。该字段被用在 7 位地址方案。任何非零值写给该位都能打开该寄存器。该寄存器的使用类似于在 A1 寄存器，但此外，该寄存器可以考虑范围匹配模式中的最大边界。
0 保留	该位保留 该位只读并有且仅有值 1

● 7.3.7.9 I2C SMBus 控制和状态寄存器（I2Cx_SMB）

地址: 4006_6000h（基址)+ 8h（偏移量）= 4006_6008h

位	7	6	5	4	3	2	1	0
读 写	FACK	ALERTEN	SIICAEN	TCKSEL	SLTF w1c	SHTF1	SHTF2 w1c	SHTF2IE
复位值	0	0	0	0	0	0	0	0

注意:当 SCL 和 SDA 信号被保持高电平的长度时间大于所述高超时周期，SHTF1 标志位置 1。在到达该阈值之前，当系统检测这些信号被拉高的时间，主机便可认为总线是空闲的。然而，SHTF1 位会在总线空闲状态下置 1。当 TCKSEL 位被置 1，没有必要监视 SHTF1 位，因为对于匹配 SMBus，总线的速度太快。w1c 表示写 1 该位就会清零，写 0 无效。

表 7-19 I2Cx_SMB 字段描述

字段	描述
7 FACK	快速 NACK/ACK 使能 为 SMBus 的分组差错校验,则 CPU 必须能够根据其结果发出一个 ACK 或 NACK 的接收数据字节。 0 一个 ACK 或 NACK 会被发送到将要接收的字节上 1 在接收一个字节后写 0 到 TXAK 位上代表 ACK。在接受一个字节后写 1 到 TXAK 位上代表 NACK。
6 ALERTEN	SMBus 报警响应地址使能 启用或禁用 SMBus 报警响应地址匹配。 注：在主机响应使用报警响应地址的设备时，则必须使用软件把设备的地址加载在总线上。警报协议在 SMBus 规范描述。 0 关闭 SMBus 报警响应地址匹配 1 开启 SMBus 报警响应地址匹配
5 SIICAEN	第二 I2C 地址启用 打开或关闭 SMBus 设备的错误地址 0 关闭 I2C 地址寄存器 2 的匹配 1 开启 I2C 地址寄存器 2 的匹配

4 TCKSEL	超时计数器时钟选择 选择时钟资源给超时计数器 0 超时计数器以总时钟 1/64 的频率计数 1 超时计数器以总时钟的频率计数
3 SLTF	SCL 低超时标志 注：w1c 表示写 1 该位就会清零，写 0 无效。 这个位在 SLT 寄存器（由 SLTH 和 SLTL 寄存器组成）被装入一个非零设置值时置 1（LoValue）和 SCL 低超时发生。软件通过写 1 来清除该位。 注：低超时功能在 SLT 寄存器为零的情况下是关闭的。 0 无低超时发生 1 有低超时发生
2 SHTF1	SCL 高超时标志位 1 这个只读位会在 SCL 与 SDA 的高电平保持时间超过 clock*LoValue/512 的时候置 1，此时总线空闲。该位会自动清零。 0 无 SCL 高超时和 SDA 低超时发生 1 有 SCL 高超时和 SDA 低超时发生
1 SHTF2	SCL 高超时标志位 2 注：w1c 表示写 1 该位就会清零，写 0 无效。 这个只读位会在 SCL 与 SDA 的低电平保持时间超过 clock*LoValue/512 的时候置 1。该位需要软件写 1 到它的方式清零。 0 无 SCL 高超时和 SDA 低超时发生 1 有 SCL 高超时和 SDA 低超时发生
0 SHTF2IE	SHTF2 中断标志使能 SCL 高超时和 SDA 低超时中断使能 0 SHTF2 中断关闭 1 SHTF2 中断打开

● 7.3.7.10 I2C 地址寄存器 2（I2Cx_A2）

地址：4006_6000h（基址）+ 9h（偏移量）= 4006_6009h

位	7	6	5	4	3	2	1	0
读	SAD							0
写								
复位值	1	1	0	0	0	0	1	0

表 7-20 I2Cx_A2 字段描述

描述	描述
7—1 SAD	SMBus 地址 该位包含了 SMBus 使用的从机地址。该位域用于设备错误地址或其他关联地址。
0 保留	该位保留 该只读位保留并且有且仅有值 0

● 7.3.7.11 I2C SCL 低位超时寄存器高电平（I2Cx_SLTH）

地址：4006_6000h（基址）+ Ah（偏移量）= 4006_600Ah

位	7	6	5	4	3	2	1	0
读写	SSLT[15:8]							
复位值	0	0	0	0	0	0	0	0

表 7-21 I2Cx_SLTH 字段描述

字段	名称	描述
7-0	SSLT[15:8]	SCL 最高有效位的低超时的值决定了 SCL 低超时的周期

● 7.3.7.12 I2C SCL 地位超时寄存器低电平（I2Cx_SLTL）

地址: 4006_6000h（基址)+ Bh（偏移量）= 4006_600Bh

位	7	6	5	4	3	2	1	0
读写	SSLT[7:0]							
复位值	0	0	0	0	0	0	0	0

表 7-22 I2Cx_SLTL 字段描述

字段	名称	描述	复位值
7—0	SSLT[7:0]	SCL 最低有效位的低超时值决定了 SCL 低超时的周期	0

■ 7.3.8 功能说明

本节全面说明了 I2C 模块的功能。

● 7.3.8.1 I2C 协议

I2C 总线系统使用串行数据线（SDA）和串行时钟线（SCL）进行数据传输。与其相连的所有设备必须具有漏极开路或集电极开路输出。逻辑和功能上实行两条线并加有外部上拉电阻。这些电阻值取决于系统。通常情况下，通信的标准实例是由四个部分组成：

1. 启动信号
2. 从地址传输
3. 数据传输
4. 停止信号

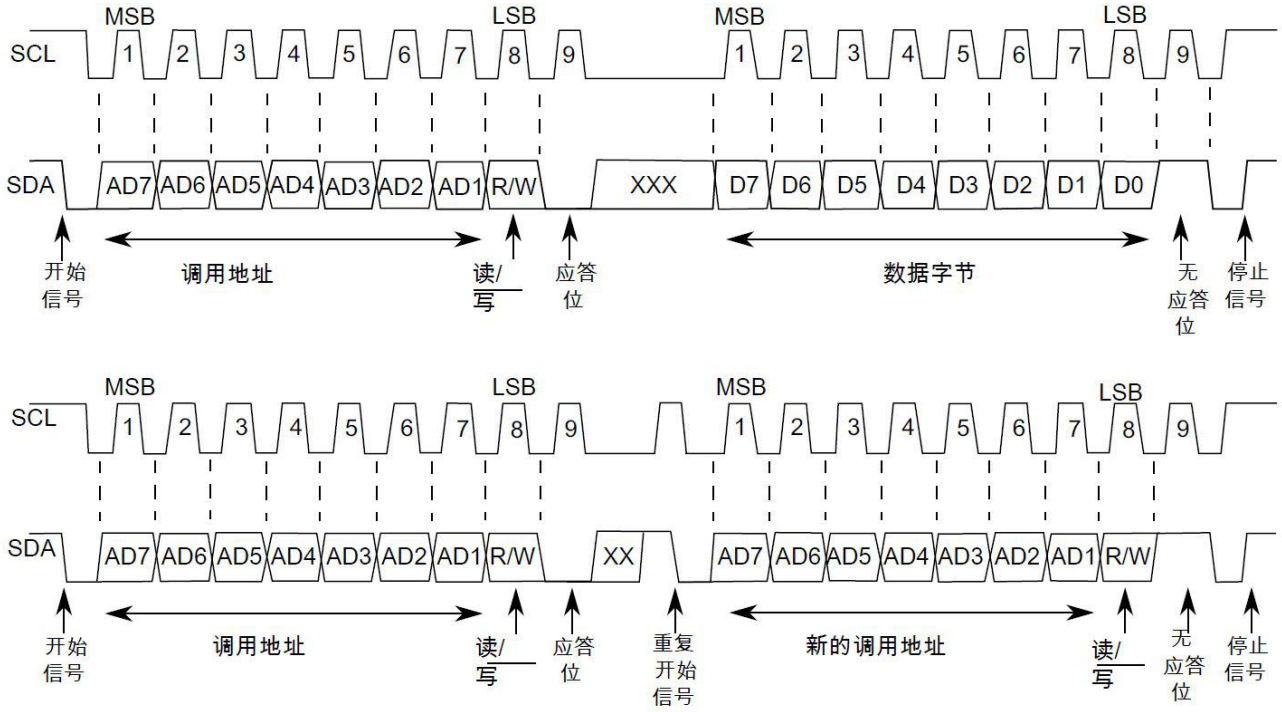


图 7-8 I2C 总线发送信号时序图

● 7.3.8.1.1 START 信号

当无主器件支配总线时（SCL 和 SDA 均为高电平），总线空闲。当总线空闲时，主器件可通过发送 START 信号启动通信。START 信号定义为在 SCL 为高电平的同时 SDA 发生高电平到低电平转换。此信号表示新数据传输的开始（每次数据传输可能包含多个字节的数据），所有从器件都会离开空闲状态。

● 7.3.8.1.2 从器件地址传输

紧接着 START 信号之后，数据传输的第一个字节是主器件发送的从器件地址。该地址包括一个 7 位调用地址和紧接着的一个 R/W 位。R/W 位告知从器件数据传输的方向。

- 1 = 读传输：从器件向主器件发送数据
- 0 = 写传输：主器件向从器件发送数据

只有地址与主机发送的调用地址一致的从器件才会响应，发送一个应答位。从机通过在第 9 个时钟周期拉低 SDA 来发送应答位。

系统中任意两个从器件的地址不能相同。如果 I2C 模块是主器件，则它不得发送一个与其自己的从地址相同的地址。I2C 模块不能既是主器件，同时又是从器件。然而，若仲裁在一个地址周期中丢失，I2C 模块将变回从器件模式，即使被另一个主器件寻址，它也能正常工作。

● 7.3.8.1.3 数据传输

成功实现从器件寻址时，便可按照调用主器件发送的 R/W 位所指定的方向，开始逐字节的数据传输。所有遵循一个地址周期的传输都被称为数据传输，哪怕是携带从器件子地址信息的传输。每个数据字节为 8 位长。数据只能在 SCL 为低电平时改变。在 SCL 为高电平时，数据必须保持稳定。对每个数据位，SCL 有一个时钟脉冲；按照 MSB 优先顺序传输。每个数据字节之后是第 9 位（应答位），它由接收器件在第 9 个时

钟脉冲拉低 SDA 而发出。总之，一个完整的数据传输需要 9 个时钟脉冲。

如果从器件接收器未在第 9 位应答主器件，从器件必须不改变 SDA，让其保持高电平。主器件将未能应答情况解释为数据传输不成功。

一个数据字节传输完毕后，如果主器件接收器未应答从器件发送器，从器件将把该情况解释为数据传输结束，从而释放 SDA 线。

无论是从器件还是主器件未能应答，数据传输都会中止，并且主器件执行以下两个操作之一：

- 产生 STOP 信号，释放总线。
- 产生重复 START 信号，开始新的调用。

● 7.3.8.1.4 停止信号

主机可通过生成停止信号释放总线的方式终止通信。停止信号定义为 SCL 的电平变为有效值时，SDA 从低电平转为高电平。

即使从机已生成了应答信号，主机仍可生成停止信号，此时从机必须释放总线。

● 7.3.8.1.5 重复开始信号

主机可能会在生成开始信号后，随即生成调用命令，而不会先生成停止信号。此操作称为重复开始。主机使用重复开始的方式并在不同的模式（传送/接收模式）下与另一台从机或同一台从机进行通信，而无需释放总线。

● 7.3.8.1.6 仲裁程序

I2C 总线是真正的多主器件总线，支持连接多个主器件。

若有两个或更多主器件同时试图控制总线，则通过一个时钟同步程序来确定总线时钟。总线时钟的低电平周期等于主器件中最长的时钟低电平周期，高电平周期等于最短的高电平周期。

竞争主器件的相对极性由数据仲裁程序决定。如果一个总线主器件发送逻辑电平 1，而另一个主器件发送逻辑电平 0，则前者出局。出局的主器件立即切换到从器件接收模式，并且停止驱动 SDA 输出。这种情况下，从主器件到从器件的模式转换不产生 STOP 条件。与此同时，硬件设置一个状态位以指示其出局。

● 7.3.8.1.7 同步时钟

由于线“与”逻辑是在 SCL 上执行，因此 SCL 上的高到低转换会影响总线上的所有器件。器件开始计数其低电平周期，某个器件的时钟变为低电平后，该器件将使 SCL 保持低电平，直至时钟达到高电平状态。然而，如果另一器件的时钟仍在低电平周期内，该器件时钟的低到高变化可能不会改变 SCL 的状态。因此，同步时钟 SCL 保持低电平的时间取决于低电平周期最长的器件。低电平周期较短的器件会在该时间内进入高电平等待状态，参见下图。当所有相关的器件都已计数完其低电平周期时，同步时钟 SCL 被释放并变为高电平。此后，器件时钟与 SCL 状态之间即无差异，所有器件开始计数其高电平周期。第一个计数完高电平周期的器件再次将 SCL 拉低。

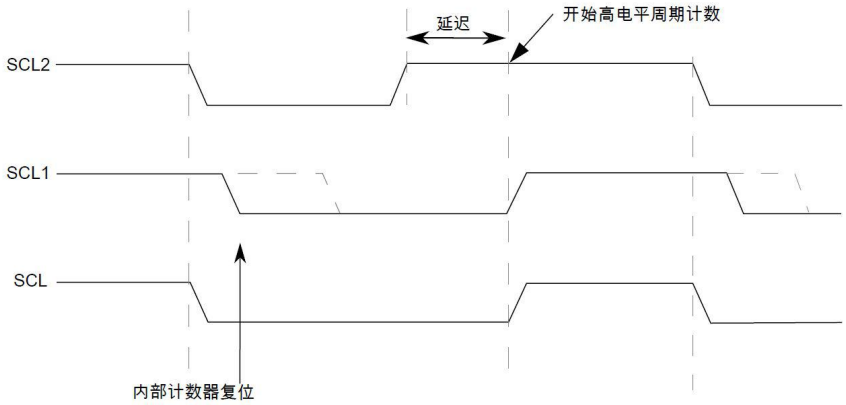


图 7-9 I2C 时钟同步

- **7.3.8.1.8 信号交换**

时钟同步机制可用作数据传输中的信号交换。完成单字节传输（9 位）之后，从器件可能会使 SCL 保持低电平。这种情况下，它会停止总线时钟，使主时钟强制进入等待状态，直至该从器件解除 SCL。

- **7.3.8.1.9 时钟拉伸**

从机可使用时钟同步机制降低传输比特率。主机将 SCL 驱动为低电平之后，从机可将 SCL 驱动为低电平，在经过所需的周期后将其释放。如果从机的 SCL 低电平周期大于主机的 SCL 低电平周期，则会拉伸产生的 SCL 总线信号的低电平周期。换言之，将增加 SCL 总线信号的低电平周期，使其与从机的 SCL 低电平周期相同。

- **7.3.8.1.10 I2C 分频器和保持值**

注： 对于某些器件，在某些情况下，当 ICR 的值在 00h 到 0Fh 这一范围时，SCL 分频器的值可能存在±2 或±4 的差异。下表突出显示了这些可能存在差异的 SCL 分频器值。有关您的器件实际的 SCL 分频器值，请参见关于 I2C 模块的芯片特定详情。

表 7-23 I2C 分频器和保持值

ICR 16 进制	SCL 分频值	SDA 保持值	SCL 保持(开始)值	SCL 保持(停止)值		ICR 16 进制	SCL 分频值	SDA 保持值	SCL 保持(开始)值	SCL 保持(停止)值
00	20	7	6	11		20	160	17	78	81
01	22	7	7	12		21	192	17	94	97
02	24	8	8	13		22	224	33	110	113
03	26	8	9	14		23	256	33	126	129
04	28	9	10	15		24	288	49	142	145
05	30	9	11	16		25	320	49	158	161
06	34	10	13	18		26	384	65	190	193
07	40	10	16	21		27	480	65	238	241
08	28	7	10	15		28	320	33	158	161
09	32	7	12	17		29	384	33	190	193

0A	36	9	14	19		2A	448	65	222	225
0B	40	9	16	21		2B	512	65	254	257
0C	44	11	18	23		2C	576	97	286	289
0D	48	11	20	25		2D	640	97	318	321
0E	56	13	24	29		2E	768	129	382	385
0F	68	13	30	35		2F	960	129	478	481
10	48	9	18	25		30	640	65	318	321
11	56	9	22	29		31	768	65	382	385
12	64	13	26	33		32	896	129	446	449
13	72	13	30	37		33	1024	129	510	513
14	80	17	34	41		34	1152	193	574	577
15	88	17	38	45		35	1280	193	638	641
16	104	21	46	53		36	1536	257	766	769
17	128	21	58	65		37	1920	257	958	961
18	80	9	38	41		38	1280	129	638	641
19	96	9	46	49		39	1536	129	766	769
1A	112	17	54	57		3A	1792	257	894	897
1B	128	17	62	65		3B	2048	257	1022	1025
1C	144	25	70	73		3C	2304	385	1150	1153
1D	160	25	78	81		3D	2560	385	1278	1281
1E	192	33	94	97		3E	3072	513	1534	1537
1F	240	33	118	121		3F	3840	513	1918	1921

● 7.3.8.2 10 位地址

在 10 位编址中，将对第一个地址字节的前 5 个位使用 0x11110。在包含 10 位编址的传输中，可以有各种组合形式的读/写格式。

● 7.3.8.2.1 主传送器对从接收器进行寻址

传输方向未更改。当 START 条件后跟 10 位地址时，各从机会将从地址 (11110XX)第一个字节的前 7 位与其自己的地址作比较并测试第八位（R/W 方向位）是否为 0。可能会有一个以上的器件出现匹配情况并生成应答(A1)。每个出现匹配情况的从机都会将从地址第二个字节的 8 位与其自己的地址作比较，但仅有一个从机出现匹配情况并生成应答(A2)。匹配的从机仍由主机寻址，直至它收到 STOP 条件(P)或后跟不同从地址的重复 START 条件(Sr)。

表 7-24 主传送器对具有 10 位地址的从接收器进行寻址

S	从器件地址前 7 位 11110+AD10+AD9	R/$\overline{W}$ 0	A1	从器件地址第二个字节 AD[8：1]	A2	数据	A	...	数据	A/A	P

主传送器发送 10 位地址的第一个字节后，从接收器会发现 I2C 中断。对于此中断，用户软件必须确保忽略数据寄存器的内容，不将其视为有效数据。

● 7.3.8.2.2 主器件接收器对从器件传送器进行寻址

第二个 R/$\overline{W}$ 位之后，传输方向改变。截止应答位 A2（包括该位）的程序与针对主器件传送器寻址从器件接收器所述的程序相同。重复 START 条件(Sr)之后，匹配的从器件记得它曾被寻址过。该从器件随后检查 Sr 之后的从器件地址的第一个字节的前 7 位是否与 START 条件(S)之后的情形相同，并测试第 8 位(R/$\overline{W}$)是否为 1。如果一致，从器件就会认为它已被寻址为发送器，并产生应答 A3。该从器件发送器保持定址状态，直至收到 STOP 条件(P)或后跟其他从器件地址的重复 START 条件(Sr)。重复 START 条件(Sr)之后，所有其他从器件也会将从器件地址的第一个字节的前 7 位与其自己的地址相比较，并测试第 8 位(R/$\overline{W}$) 。然而，它们中的任何一个器件都不会被寻址，因为 R/$\overline{W}$ = 1（对于 10 位器件），或者 11110XX 从器件地址（对于 7 位器件）不匹配。

表 7-25 主器件接收器用 10 位地址寻址从器件发送器

S	从器件地址前 7 位 11110+AD10 +AD9	R/$\overline{W}$ 0	A1	从器件地址第二个字节 AD[8：1]	A2	Sr	从器件地址前 7 位 11110+AD10 +AD9	R/$\overline{W}$ 1	A3	数据	A	...	数据	A	P

主器件接收器发送 10 位地址的第一个字节之后，从器件发送器收到一个 I2C 中断。对于此中断，用户软件必须确保忽略数据寄存器的内容，不将其当作有效数据对待。

● 7.3.8.3 地址匹配

对于所有接收到的地址，都能够以 7 位或 10 位地址格式请求。

• 地址寄存器 1（包含 I2C 第一从机地址）中的 AD[7:1]始终参与地址匹配过程。它提供一个 7 位地址。

• 如果 ADEXT 位置位，控制寄存器 2 中的 AD[10:8]将参与地址匹配过程。它将 I2C 第一从机地址扩展为一个 10 位地址。影响地址匹配的其他条件包括：

- 如果 GCAEN 位置位，通用广播地址将参与地址匹配过程。
- 如果 ALERTEN 位置位，警告地址将参与地址匹配过程。
- 如果 SIICAEN 位置位，地址寄存器 2 将参与地址匹配过程。
- 如果 RMEN 位置位，当范围地址寄存器编程为非零值时，地址寄存器 1（不包含）和范围地址寄存器（包含）的值范围内的任何地址都将参与地址匹配过程。范围地址寄存器值必须编程为大于地址寄存器 1 的值。

I2C 模块响应这些地址中的任一地址时，将充当从机接收器，并且 IAAS 位将在地址周期结束后置位。软件必须在第一个字节传输后读取数据寄存器，以确定地址已匹配。

● 7.3.8.4 系统管理总线规范

SMBus 提供系统和电源管理相关任务的控制总线。系统可使用 SMBus 向设备发送消息和从设备接收消息，而不是跳变单个控制线路。取消单个控制线路可减少引脚数量。接受消息可确保未来的可扩展性。使用系统管理总线，设备可实现以下功能：提供制造商信息、告知系统其型号/设备编号、保存挂起事件的状态、报告不同类型的错误、接受控制参数、返回其状态。

● 7.3.8.4.1 超时

根据 $T_{TIMEOUT.MIN}$ 参数，主机或从机可决定是缺陷器件将时钟无限期地保持在低电平，还是主机有意驱动器件脱离总线。从器件检测到任意单个时钟保持在低电平的时间长于 $T_{TIMEOUT.MIN}$ 时，必须释放总线（停

止驱动总线并使 SCL 和 SDA 悬空为高电平）。已检测到此条件的器件必须复位其通信并且能在 $T_{TIMEOUT.MAX}$ 的时间范围内接收新的 START 条件。

SMBus 定义 35ms 的时钟低电平超时 $T_{TIMEOUT}$，将 $T_{LOW:SEXT}$ 指定为从器件的累积时钟低电平延长时间，将 $T_{LOW:MEXT}$ 指定为主器件的累积时钟低电平延长时间。

● 7.3.8.4.1.1 SCL 低电平超时

如果 SCL 线路通过总线上的从器件保持为低电平，则无法开展进一步通信。此外，主器件无法将 SCL 线路强制变为高电平以纠正该错误状况。为了解决这个问题，SMBus 协议要求参与传输的器件必须检测时钟周期保持低电平的时间是否超过某一超时值。检测到超时状况的器件必须复位通信。当 I2C 模块是有效主器件时，若它检测到 SMBCLK 低电平时间已超过 $T_{TIMEOUT.MIN}$ 的值，它必须在传输过程的当前数据字节内或之后产生停止条件。当 I2C 模块是从器件时，若它检测到 $T_{TIMEOUT.MIN}$ 状况，它将复位通信，然后便能接收新的起始(START)条件。

● 7.3.8.4.1.2 SCL 高电平超时

I2C 模块确定 SMBCLK 和 SMBDAT 信号已处于高电平状态至少 $T_{HIGH:MAX}$ 时间后，它假设总线处于闲置状态。

当总线上出现 START 条件，并且 STOP 条件未出现前，高电平超时会出现。满足下列任一条件时，任何检测到该情况的主机都可假设总线处于空闲状态。

- SHTF1 上升。
- BUSY 位处于高电平状态，SHTF1 也处于高电平状态。

当有一段时间 SMBDAT 信号处于低电平状态且 SMBCLK 信号处于高电平状态时，会发生另一种超时。必须在软件中定义该时间周期。SHTF2 在已达到时间限制的情况下作为标志。该标志也是一个中断资源，因此它会触发 IICIF。

● 7.3.8.4.1.3 CSMBCLK TIMEOUT MEXT 和 CSMBCLK TIMEOUT SEXT

下图说明超时间隔 $T_{LOW:SEXT}$ 和 $T_{LOW:MEXT}$ 的定义。在主机模式下，在一个字节之内，I2C 模块时钟周期的累计延长时间不得超过 $T_{LOW:MEXT}$；各字节定义为 START 至 ACK、ACK 至 ACK 或 ACK 至 STOP。发生 CSMBCLK TIMEOUT MEXT 时，SMBus MEXT 上升，同时会触发 SLTF。

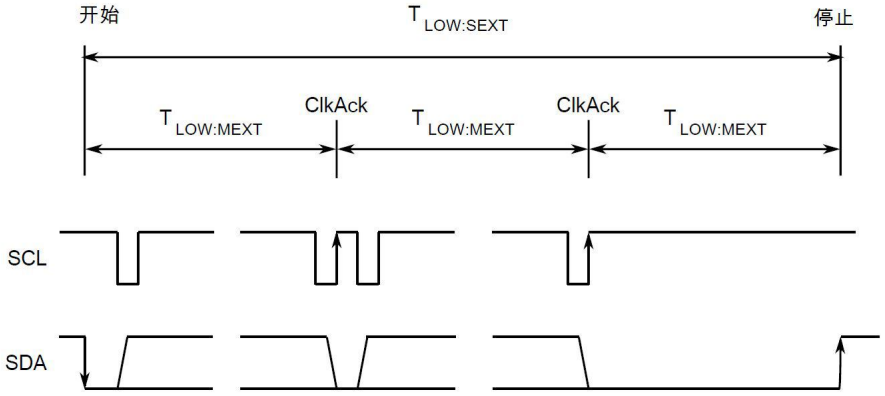


图 7-10 超时测量间隔

主机可中止正在进行的任何违反 $T_{LOW:SEXT}$ 或 $T_{TIMEOUT.MIN}$ 要求的从机的处理。要中止处理，主机应在当前字节传输结束时发送 STOP 条件。作为从机，在从初始 START 到 STOP 的任何消息传输期间，I2C 模块时钟周期的累计延长时间不得超过 $T_{LOW:SEXT}$。发生 CSMBCLK TIMEOUT SEXT 时，SEXT 上升，同时会触发 SLTF。

注:CSMBCLK TIMEOUT SEXT 和 CSMBCLK TIMEOUT MEXT 是在第二步中实现的可选功能。

● 7.3.8.4.2 FAST ACK 和 NACK

为了提高可靠性和通信的稳定性，SMBus 设备是否实现数据包差错校验(PEC)是可选的，但对于参与地址解析协议(ARP)过程的器件，PEC 则是必需的，并且仅在该过程中需要。PEC 是一个基于所有消息字节的 CRC-8 错误校验字节。PEC 由提供最后一个数据字节的器件追加到消息中。如果 PEC 存在但不正确，接收器将发送 NACK，否则发送 ACK。为了通过软件计算 CRC-8，在接收到第 8 个 SCL（第 8 位）之后，如果该字节是数据字节，此模块将使 SCL 线保持低电平。这样，软件就能确定应当向总线发送 ACK 还是 NACK，即置位还是清除 TXAK 位（如果快速 ACK/NACK 使能位 FACK 已使能）。

SMBus 要求设备始终应答其自己的地址，作为检测总线上可移除设备（如电池或插接站）存在与否的机制。除了指示从设备繁忙条件以外，SMBus 还使用 NACK 机制来表示接收到无效命令或数据。由于这种条件可能出现在最后一个字节时，因此 SMBus 设备必须有能力在传输每个字节之后及完成处理之前产生不应答信号。因为 SMBus 不提供任何其他重新发送信令，这一要求非常重要。使用 NACK 信令中的这一差别对 SMBus 端口的具体实现有影响，尤其是对于 SMBus 主机和 SBS 器件等处理关键系统数据的设备。

注:在主机接收从机传送模式的最后一个字节中，主机必须向总线发送一个 NACK，因此在传送最后一个字节之前必须关闭 FACK。

● 7.3.8.5 复位

复位后，I2C 模块禁止使用。I2C 模块无法引发内核复位。

● 7.3.8.6 中断

假定 IICIE 位置位，当发生此处表格中的任一事件时，I2C 模块都会生成中断。中断由（I2C 状态寄存器的）IICIF 位驱动，通过（I2C 控制寄存器 1）IICIE 位屏蔽。在中断例程中，必须通过向 IICIF 位写入 1 的方式由软件清零 IICIF 位。SMBus 超时中断由 SLTF 驱动并通过 IICIE 位屏蔽。在中断例程中，必须通过向 SLTF 位写入 1 的方式由软件清零 SLTF 位。 可通过对状态寄存器进行读操作确定中断类型。
注：在主机接收模式下，最后一个字节传输之前，FACK 位必须设为零。

表 7-26 中断汇总

中断源	状态	标志	本地使能
完成 1 个字节的传输	TCF	IICIF	IICIE
已接收的调用地址匹配	IAAS	IICIF	IICIE
仲裁丢失	ARBL	IICIF	IICIE
I2C 总线停止检测	STOPF	IICIF	IICIE & SSIE
I2C 总线起始位检测	STARTF	IICIF	IICIE & SSIE
SMBus SCL 低电平超时	SLTF	IICIF	IICIE
SMBus SCL 高电平 SDA 低电平超时	SHTF2	IICIF	IICIE 和 SHTF2IE
从停止或等待模式唤醒	IAAS	IICIF	IICIE 和 WUEN

● 7.3.8.6.1 字节传输中断

传输完成标志(TCF)位在第九个时钟的下降沿处置位，以表示字节和应答传输完成。启用 FACK 后，TCF 在第八个时钟的下降沿处置位 ，以表示字节完成。

● 7.3.8.6.2 地址检测中断

当调用地址与所设置的从器件地址（I2C 地址寄存器）一致时，或者当 GCAEN 位置位且收到广播时，状态寄存器的 IAAS 位置位。如果 IICIE 位置位，则 CPU 中断。CPU 必须检查 SRW 位，并相应地设置其 Tx 模式。

● 7.3.8.6.3 停止检测中断

当在 I2C 总线上检测到停止状态时，STOPF 位置 1。如果 IICIE 位和 STOPIE 位均置 1，则会中断 CPU。

● 7.3.8.6.4 退出低功耗/停止模式

在低功耗模式（等待和停止）下，从器件接收输入检测电路和地址匹配功能仍然有效。如果中断未被屏蔽，与从器件地址或广播地址匹配的异步输入将使 CPU 离开低功耗/停止模式。因此，TCF 和 IAAS 均可触发该中断。

● 7.3.8.6.5 仲裁丢失中断

I2C 是真正的多主机总线，支持连接多个主机。如果两个或两个以上主机要同时控制总线，则进行竞争的主机的相对极性由数据仲裁程序决定。I2C 模块在其丢失数据仲裁进程且状态寄存器的 ARBL 位置位时会使仲裁-丢失中断的电平变为有效。

仲裁在下列情形时丢失：

1. 在地址或数据传送周期内，主机驱动为高电平时以低电平采样 SDA。
2. 在应答位的数据接收周期内，主机驱动为高电平时以低电平采样 SDA。
3. 总线繁忙时尝试采用 START 周期。
4. 从机模式请求重复的 START 周期。
5. 主机未请求该周期时检测到 STOP 条件。

必须通过将 1 写入 ARBL 位由软件将此位清零。

● 7.3.8.6.6 SMBus 中的超时中断

当 IICIE 位置位时，一旦检测到上述任一超时条件，I2C 模块就会将超时中断的电平变为有效值（输出 SLTF 和 SHTF2），但有一个例外。SCL 高电平和 SDA 高电平 TIMEOUT 机制不得用来影响超时中断输出，因为该超时指示总线空闲条件。当与 SCL 高电平和 SDA 高电平 TIMEOUT 匹配时，SHTF1 上升，并自动下降以指示总线状态。SHTF2 的超时周期与 SHTF1 相同，比 SLTF 要短，因此增加了另一个控制位 SHTF2IE 来使能或禁用它。

● 7.3.8.7 可编程输入毛刺过滤器

I2C 毛刺过滤器已添加到传统 I2C 模块的外部，而非 I2C 封装内。该过滤器可吸附 I2C 时钟和 I2C 模块

数据线路上的毛刺。

可根据（半）I2C 模块时钟周期数指定待吸收毛刺的宽度。提供单个可编程输入毛刺过滤器控制寄存器。实际上，I2C 模块通常会忽略数据线路上的任意下降-上升-下降或上升-下降-上升转换（在该寄存器编程的时钟周期数范围内）。程序员必须指定毛刺的大小（根据 I2C 模块时钟周期），以便过滤器吸收毛刺，并且阻止毛刺通过。

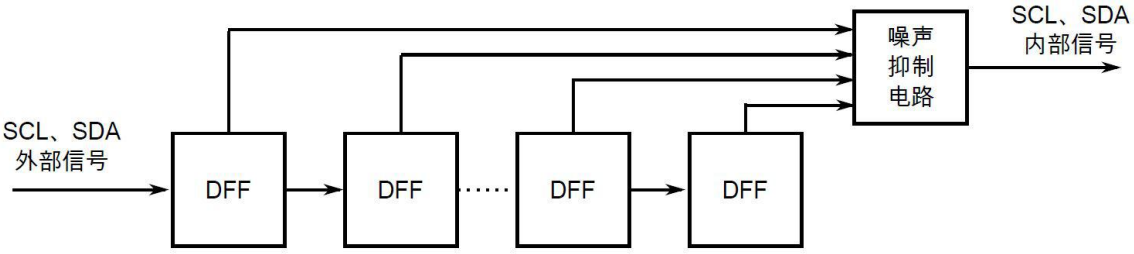


图 7-11 可编程输入毛刺过滤器示意图

● 7.3.8.8 地址匹配唤醒

当 I2C 模块处于从接收模式下时，如果发生第一地址、范围地址或通用调用地址匹配情况，则 MCU 将从低功耗模式（无外围总线在此模式下运行）中唤醒。

设置地址匹配 IAAS 位后，将在地址匹配结束时发送中断以唤醒内核。

注：唤醒过程中，如果某个外部主机继续向从机发送数据，则停止模式下的波特率必须低于 50 kbit/s。为避免在停止模式下出现速度较低的波特率，主机可在固件中增加一个短时延迟，直至唤醒过程完成，然后再发送数据。SMBus 模式不支持地址匹配引起的唤醒。

■ 7.3.9 初始化/应用信息

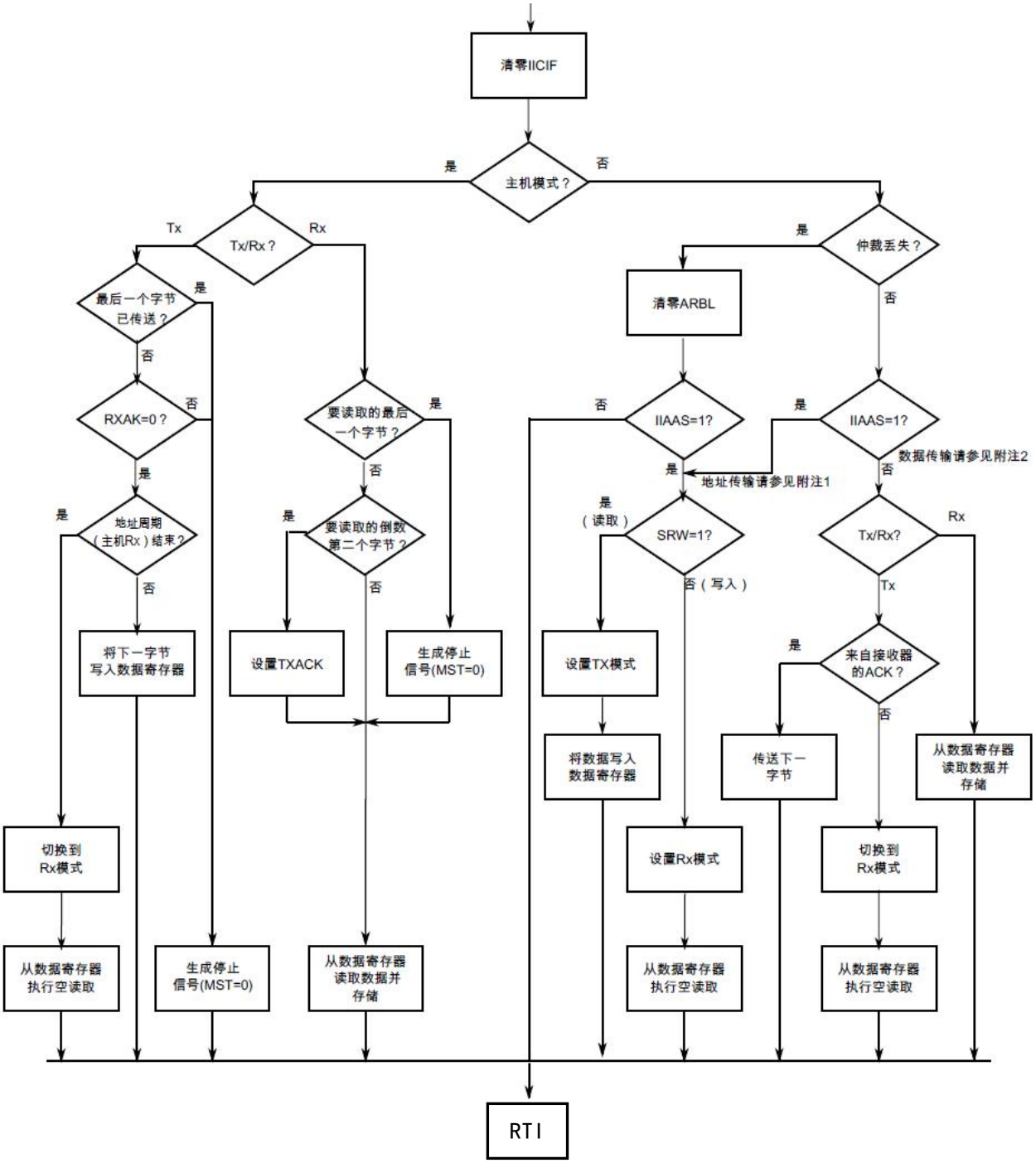
（一）模块初始化（从机）

1. 写入：控制寄存器 2
   - 使能或禁用通用调用命令
   - 选择 10 位或 7 位编址模式
2. 写入：地址寄存器 1，以置位从机地址
3. 写入：控制寄存器 1，以使能 I2C 模块和中断
4. 初始化发送数据的 RMA 变量（IICEN = 1 和 IICIE = 1）
5. 初始化用于达到下图所示例程的 RAM 变量

（二）模块初始化（主机）

1. 写入：分频器寄存器，以置位 I2C 波特率（参见 ICR 说明中的示例）
2. 写入：控制寄存器 1，以使能 I2C 模块和中断
3. 初始化发送数据的 RMA 变量（IICEN = 1 和 IICIE = 1）
4. 初始化用于实现下图所示例程的 RAM 变量
5. 写入：控制寄存器 1，以使能 TX
6. 写入：控制寄存器 1，以使能 MST（主机模式）
7. 写入：数据寄存器和目标从机地址（该字节的 LSB 确定是主机接收还是发送通信）

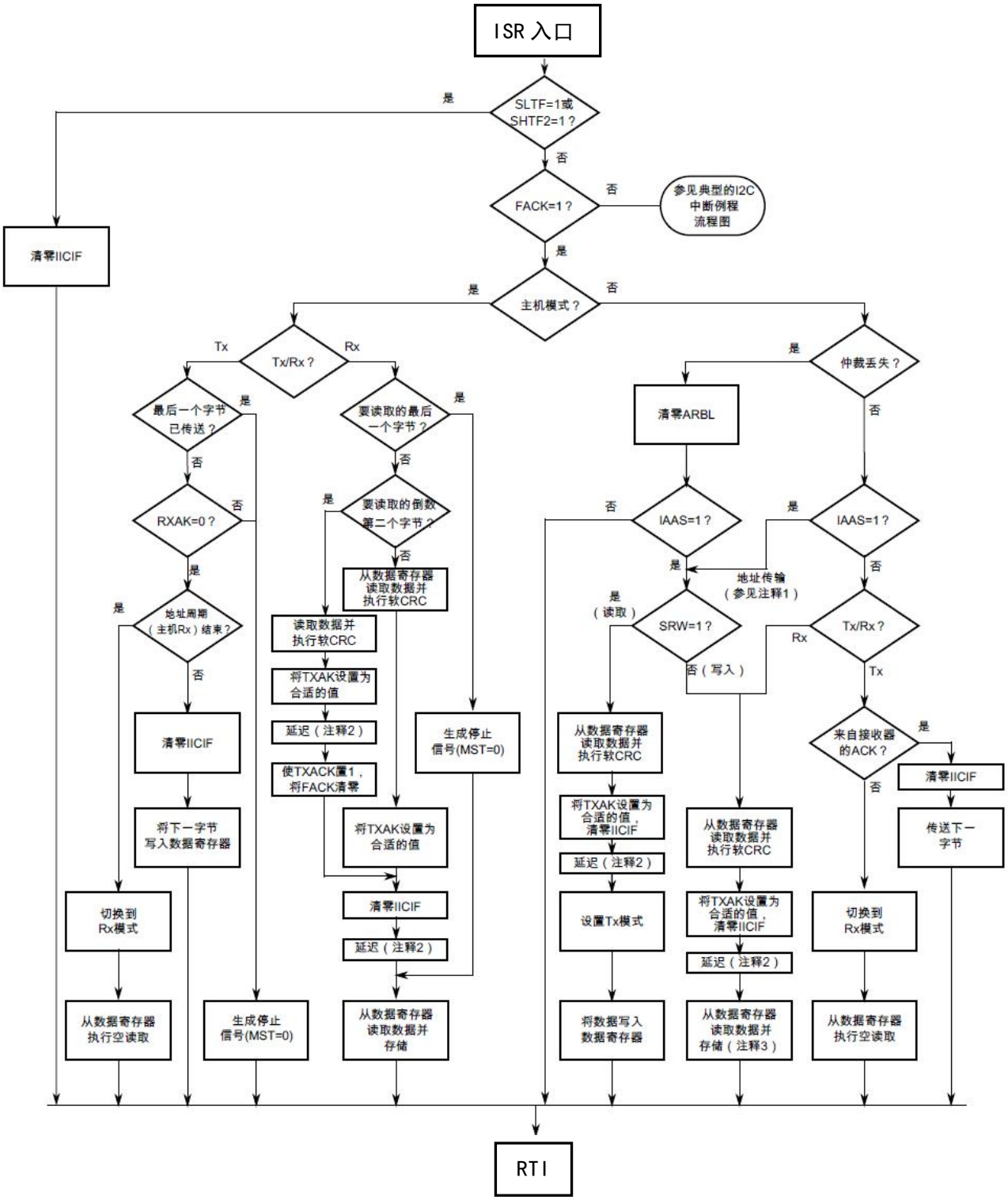
下图例程显示了主机和从机 I2C 操作。对于从机操作，包含正确地址的 I2C 传入消息开始 I2C 通信。对于主机操作，通信初始化必须通过写入数据寄存器的方式完成。



注释：1. 如果已使能通用调用，则检查以确定接收到的地址是否是通用调用地址(0x00)。如果接收到的地址是通用调用地址，则通用调用必须有用户软件处理。

2. 如果 10 位寻址进行的是从机寻址，则在扩展地址的首个字节后，该从机会遭遇中断。对于此中断，必须确保忽略数据寄存器的内容，且不将其当作有效数据传输。

图 7-12 典型 I2C 中断例程



注释：1. 如果已使能通用调用或 SIICAEN，则进行检查以确定接收到的地址是通用调用地址(0x00)或 SMBus 器件默认地址。无论哪一种，都必须由用户软件来处理。

2. 在接收模式下，在第一个和第二个数据读取之前，可能需要一个位时间的延迟，以等待第 9 个 SCL 周期可能的最长时间周期(最差的情况下）。

3. 这是一个空读取，是为了复位 SMBus 接收器状态机。

图 7-13 典型 I2C SMBus 中断例程

■ 7.4 通用异步收发器 UART 模块

■ 7.4.1 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，是电脑硬件的一部分。将资料由串行通信与并行通信间作传输转换，作为并行输入成为串行输出的芯片，通常集成于其他通讯接口的连结上。

UART 模块有两种引脚配置，需要对 SIM 模块的相关寄存器进行配置，具体请参考 SIM 寄存器说明。

■ 7.4.2 操作模式

- 8 位或 9 位数据模式
- 停止模式操作
- 循环模式
- 单线模式

■ 7.4.3 结构框图

下图是 UART 传送器结构框图：

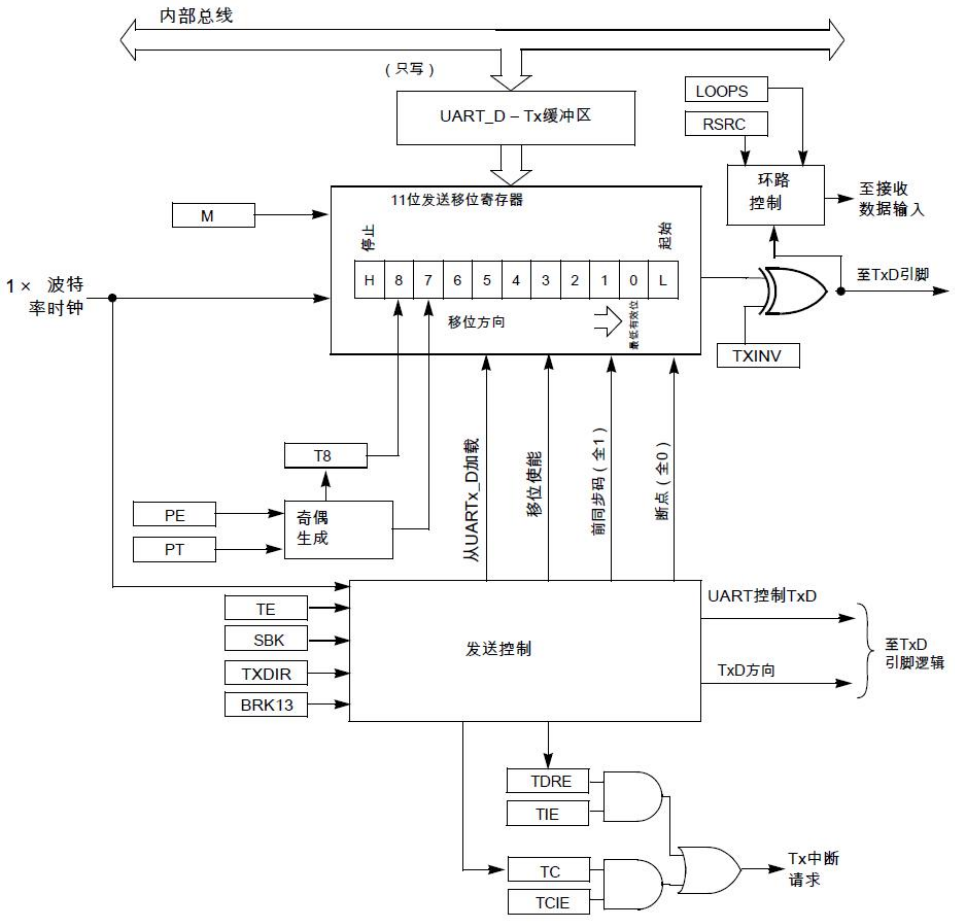


图 7-14 UART 传送器结构框图

下图是 UART 接收器结构框图：

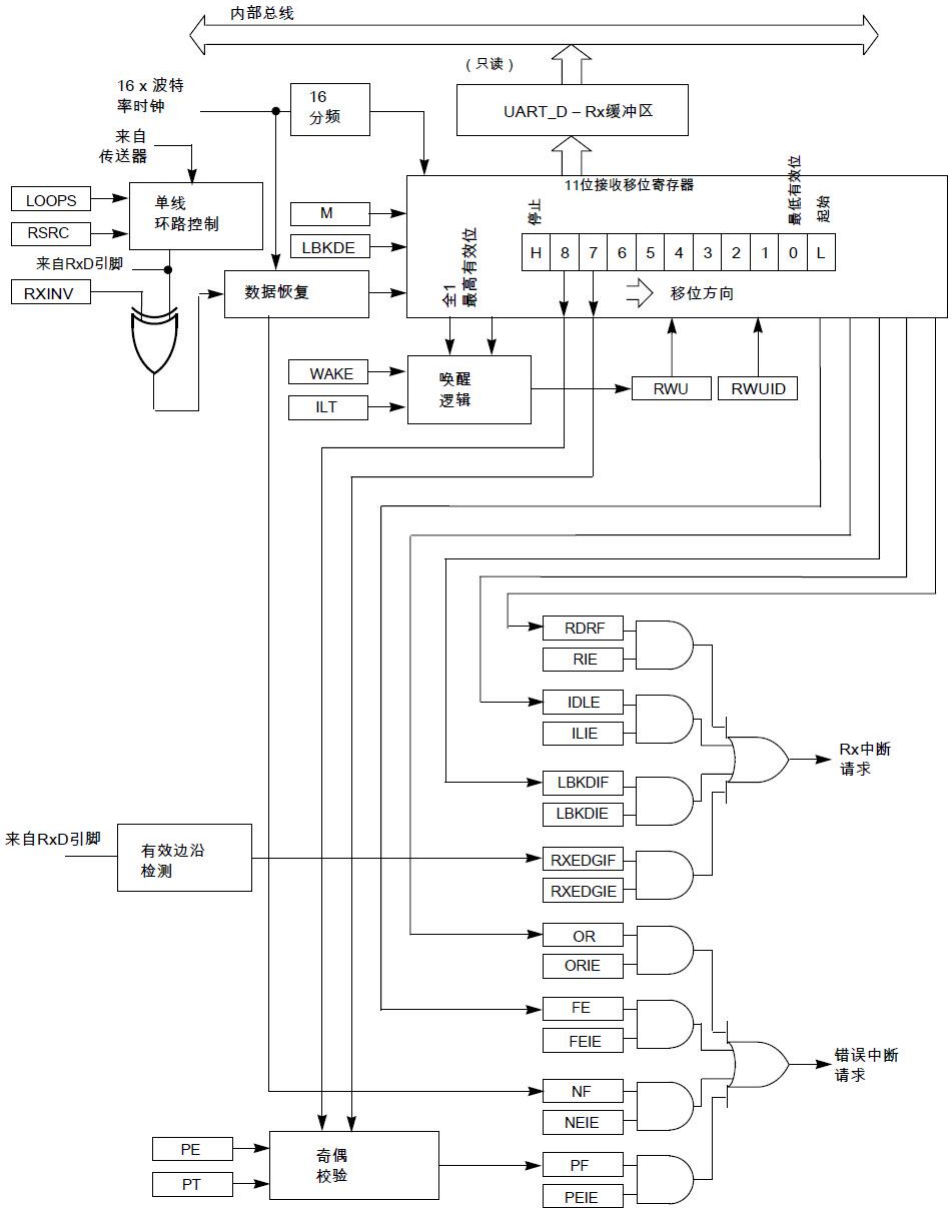


图 7-15 UART 接收器结构框图

■ 7.4.4 特性

UART 模块的特性：

- 全双工，标准的不归零（NRZ）格式
- 双缓冲发射器和接收器，具有单独的使能
- 可编程波特率（13 位模因子）
- 中断驱动或轮询操作的条件：
  - 发送数据寄存器空，发送完成
  - 接收数据寄存器满
  - 接收溢出，奇偶校验错误，帧错误和噪声误差
  - 空闲接收器检测

- 在接收引脚出现有效边沿
- 支持 LIN 的间隔检测

- 硬件奇偶产生和校验
- 可编程的 8 位或 9 位字符长度
- 可编程 1 位或 2 位停止位
- 唤醒接收器通过空闲线或地址标记
- 可选的 13 位间隔字符代/11 位间隔字符检测
- 可选择变送器输出极性

■ 7.4.5 引脚说明

UART 模块有两种引脚配置，需要对 SIM 模块的相关寄存器进行配置，具体请参考 SIM 寄存器说明。

■ 7.4.6 存储器映射和寄存器说明

串口有 8 位的寄存器去控制波特率，控制串口选项，反馈串口状态和进行数据的收发。请参见直接页寄存器总结本文档或记忆章绝对地址分配所有 UART 寄存器。本节涉及到寄存器和它们的控制位。

表 7-23 寄存器映射

绝对地址 (16 进制)	寄存器名称	偏移量	位宽	权限	复位值
4006_*000h	UART 波特率寄存器：高（UARTx_BDH）	0h	8	R/W	00h
4006_*001h	UART 波特率寄存器：低（UARTx_BDL）	1h	8	R/W	04h
4006_*002h	UART 控制寄存器 1（UARTx_C1）	2h	8	R/W	00h
4006_*003h	UART 控制寄存器 2（UARTx_C2）	3h	8	R/W	00h
4006_*004h	UART 状态寄存器 1（UARTx_S1）	4h	8	R	C0h
4006_*005h	UART 状态寄存器 2（UARTx_S2）	5h	8	R/W	00h
4006_*006h	UART 控制寄存器 3（UARTx_C3）	6h	8	R/W	00h
4006_*007h	UART 数据寄存器（UARTx_D）	7h	8	R/W	00h

注：UART0 *= A x=0 / UART1 *= B x=1 / UART2 *= C x=2

● 7.4.6.1 串口波特率寄存器：高位（UARTx_BDH）

该寄存器，结合 UART_BDL，控制预分频因子来产生 UART 波特率。想要更新 13 位波特率设置[SBR12：SBR0]，首先要把新值的一半写进 UART_BDH 来缓冲高位，然后写 UART_BDL。该 UART_BDH 工作值不会改变，直到 UART_BDL 被写入。

地址：基址+ 0h 偏移 ：UART0_BDH= 4006_A000h \ UART1_BDH=4006_B000h \ UART2_BDH=4006_C000h

位	7	6	5	4	3	2	1	0
读写	LBKDIE	RXEDGIE	SBNS	SBR				
复位值	0	0	0	0	0	0	0	0

表 7-24 UARTx_BDH 字段描述

位	描述
7 LBKDIE	LIN 中止检测中断使能（对于 LBKDIF） 0 UART_S2[LBKDIF]位被禁止的硬件中断（使用轮询）。 1 当 UART_S2[LBKDIF]标志为 1 的硬件中断请求
6 RXEDGIE	RXD 输入有效边中断使能（对于 RXEDGIF） 0 UART_S2[RXEDGIF]位被禁止的硬件中断（使用轮询）。 1 UART_S2[RXEDGIF]标志为 1 的硬件中断请求。
5 SBNS	停止位位宽选择 0 一位停止位 1 两位停止位
4-0 SBR	波特率模数因子 SBR[12：0]被统称为 BR，可以为 UART 波特率发生器设置模块因子。当 BR 被清零时，UART 波特率发生器被禁止以减少电源电流。当 BR 为 1 – 8191 的 UART 波特率等于 BUSCLK/（16×BR）。

● 7.4.6.2 串口波特率寄存器：低位（UARTx_BDL）

该寄存器，结合 UART_BDH，控制预分频因子来产生 UART 波特率。想要更新 13 位波特率设置[SBR12：SBR0]，首先要把新值的一半写进 UART_BDH 来缓冲高位，然后写 UART_BDL。该 UART_BDH 工作值不会改变，直到 UART_BDL 被写入。

地址：基址+ 1h 偏移 ：UART0_BDL= 4006_A001h \ UART1_BDL=4006_B001h \ UART2_BDL=4006_C001h

位	7	6	5	4	3	2	1	0
读 写	SBR							
复位值	0	0	0	0	0	1	0	0

表 7-25 UARTx_BDL 字段描述

位	描述
7-0 SBR	波特率模数因子 SBR[12：0]被统称为 BR，可以为 UART 波特率发生器设置模块因子。当 BR 被清零时，UART 波特率发生器被禁止以减少电源电流。当 BR 为 1 – 8191 的 UART 波特率等于 BUSCLK/(16×BR)。

● 7.4.6.3 UART 控制寄存器 1（UARTx_C1）

该读/写寄存器控制 UART 系统的各种可选功能。

地址：基址+ 2h 偏移 ：UART0_C1= 4006_A002h \ UART1_C1=4006_B002h \ UART2_C1=4006_C002h

位	7	6	5	4	3	2	1	0
读 写	LOOPS	UARTSWAI	RSRC	M	WAKE	ILT	PE	PT
复位值	0	0	0	0	0	0	0	0

表 7-26 UARTx_C1 字段描述

位	描述
7 LOOPS	循环模式选择 循环模式和正常模式之间选择。当 LOOPS 被设置时，发送器输出连接到接收器的输入。 0 正常操作 – RXD 和 TXD 使用不同的引脚 1 循环模式或单线模式下发送器输出连接到接收器的输入并且 RXD 引脚不被使用
6 UARTSWAI	在等待模式下串口停止 0 串口时钟在等待模式下是一直运行的所以串口可以作为一种中断资源去唤醒 CPU 1 串口时钟在 CPU 处于等待模式的情况下是关闭的
5 RSRC	接收器资源选择 该位在 LOOPS 位置 1 时毫无意义和影响。当 LOOPS 位置 1，接收器的内部输入会连接到 TxD 引脚上，RSRC 位决定了此连接是否也连接到发送器的输出上。 0 假如 LOOPS 位置 1，RSRC 被清除，启用内部循环模式并且 UART 不使用 RxD 引脚。 1 单线模式下 TxD 引脚连接到发送器的输出和接收器的输出上
4 M	9 位或 8 位模式选择 0 正常启动+8 位数据位+停止 1 接收器和发送器使用 9 位字符启动+8 位数据位+第九位数据位+停止
3 WAKE	接收器唤醒方法选择 0 空闲线唤醒 1 地址标志唤醒
2 ILT	空闲线类型选择 对此位置 1,确保停止位和字符末尾的逻辑 1 位不计入在 10 或 11 位逻辑需要空闲线检测的逻辑高电平上。 0 在起始位后启用空闲字符位计数 1 在停止位后启用空闲字符位计数
1 PE	奇偶校验使能 使能奇偶校验和检测，当奇偶校验打开，数据的最高有效位，第 8 和第 9 数据位都被当作奇偶校验位。 0 无硬件奇偶产生或检测 1 奇偶校验使能打开
0 PT	奇偶校验类型 当 PE 位置 1 时才能选择奇校验和偶校验。奇校验指数据中的 1 包括校验位是否为奇数。偶校验指数据中的 1 包括校验位是否为偶数。 0 偶校验 1 奇校验

● 7.4.6.4 UART 控制寄存器 2（UARTx_C2）

该寄存器可以在任意时刻被读写。

地址: 基址+ 3h 偏移 ：UART0_C2= 4006_A003h \ UART1_C2=4006_B003h \ UART2_C2=4006_C003h

位	7	6	5	4	3	2	1	0
读 写	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
复位值	0	0	0	0	0	0	0	0

表 7-27 UARTx_C2 字段描述

位	描述
7 TIE	对于 TDRE 位的发送中断使能 0 对于 TDRE 位的硬件中断禁止，使用轮询。 1 TDRE 标志为 1 的硬件中断请求时。
6 TCIE	传输完成中断使能 0 对于 TC 位的硬件中断禁止，使用轮询。 1 TDRE 标志为 1 的硬件中断请求时。
5 RIE	接收器中断使能 0 对于 RDRF 位硬件中断禁止，使用轮询。 1 TDRE 标志为 1 的硬件中断请求时。
4 ILIE	空闲线中断使能 0 硬件中断禁止，使用轮询。 1 硬件中断请求时，当 IDLE 标志为 1。
3 TE	发送器使能 TE 位必须置 1 才能使用 UART 发送器。当 TE 位置 1，串口会将 TxD 引脚作为串口系统的输出当串口被初始化位单线操作模式，XDIR 位控制着单个 UART 交互通信线方向（TXD 引脚）。 TE 还可以通过清除 TE 然后设置 TE 的方式对传输过程中的空闲字符进行排列。 在允许 TxD 引脚恢复到一个通用 I/ O 引脚进行发送前，当 0 写入 TE，发射机一直保持着对 TXD 引脚的控制，直到所有数据，空闲队列，或间隔字符流完成发送。 0 发送器关闭 1 发送器打开
2 RE	接收器使能 当 UART 接收器关时，RxD 引脚恢复为通用端口 I/ O 引脚。如果 LOOPS 位置 1 那么 RxD 引脚恢复为通用端口 I/ O 引脚，即使 RE 位置 1。 0 接收关闭 1 接收打开
1 RWU	接收器唤醒控制 该位可以写入 1 放置 UART 接收器在备用状态，等待自动硬件检测唤醒条件。WAKE=0，空闲线唤醒，或者在一个字符最高有效数据位为逻辑 1。WAKE=1，地址标志唤醒。应用软件置位 RWU，通常，所选择的硬件条件自动清零 RWU 位。 0 正常 UART 接收器操作 1 UART 接收器在待机状态，等待唤醒条件。
0 SBK	发送间隔 写一个 1 然后一个 0 到 SBK 发送数据流中的间隔字符。其他间隔字符为 10 或 11 或 12，13 或 14 或 15 个字符，如果 BRK13 位= 1，逻辑 0 的位周期将被排列，只要 SBK 位置 1。根据设

	定的时间和在发送过程中清零 SBK，在软件清零 SBK 前第二间隔符可能会被排列。 0 正常发送器操作 1 队列间隔字符发送

● 7.4.6.5 UART 状态寄存器 1（UARTx_S1）

该寄存器有 8 个只读标志。对其写毫无影响。特定的软件编程并不能对其清零和置 1。

地址：基址+ 4h 偏移 ：UART0_S1= 4006_A004h \ UART1_S1=4006_B004h \ UART2_S1=4006_C004h

位	7	6	5	4	3	2	1	0
读	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
写								
复位值	1	1	0	0	0	0	0	0

表 7-28 UARTx_S1 字段描述

位	描述
7 TDRE	发送寄存器空标志 TDRE 被设置了复位值。当从发送数据缓冲区发送数据值传送到发送移位器，留下空间中的缓冲区的新字符。要清除 TDRE，需要读取到 TDRE 位为 1 ，然后写数据到 UART 数据寄存器（UART_D）。 0 发送寄存器满。 1 发送寄存器空。
6 TC	发送完成标志 TC 被设置了复位值。当 TDRE 位置 1 无数据，前导符和间隔字符被发送。 通过读取 TC 置 1 的 UART_S1 寄存器的方式来给 TC 清零然后执行以下任一操作： • 向 UART 数据寄存器写数据来发送数据 • 通过将 TE 位由 0 向 1 变换来排列前导符 • 通过向 UART_C2[SBK]位置 1 来排列间隔字符 0 发射机忙（发送数据，前导符或休息）。 1 发送器空闲（发送活动完成）。
5 RDRF	接收数据寄存器满标志 当从一个字符从接收转换器传输到接收数据寄存器（UART_D），RDRF 置 1。要清零 RDRF,要先读取 RDRF 位置 1 的 UART_S1 寄存器 ,然后读去 UART 数据寄存器(UART_D)。 0 接收数据寄存器空 1 接收数据寄存器满
4 IDIE	空闲线标志 当 UART 接收线路空闲一段完整的字符时间后，IDIE 位置 1。当 ILT 被清零时，接收器在起始位结束后计数空闲位。如果接收的字节全为 1，这些位周期和停止位周期可通过整个字节的逻辑高电平来计数，10 位或 11 位周期取决于 M 控制位，需要接收器来检测空闲线。当 ILT 被置 1，接收器不启动空闲位计数直至停止位结束。停止位和任何逻辑高电平在前面的字符的结束时间内不计入满字符周期的逻辑高电平。 为了清零 IDLE，需要读取 IDLE 置 1 的 UART_S1 寄存器，然后读 UART 数据寄存器

	（UART_D）。在 IDLE 位被清零后，它才能被重新设置直到一个新的字符已被接收和 RDRF 置 1。IDLE 只设置一次，即使接收线闲置较长时间。 0 无空闲线检测 1 有空闲线检测
3 OR	接收器过载标志 OR 位置 1，当一个新的串行数据准备要传送到接收数据寄存器（缓冲器），但先前接收到的数据并不是从 UART_D 寄存器中读出。在这种情况下，新的字符，和所有的相关错误信息，将会丢失，因为没有空间来将其移动到 UART_D。要清零 OR，先读取 UART_S1 寄存器 OR 位置 1 的状态，然后读 UART 数据寄存器（UART_D）。 0 无过载 1 接收到过载（新串口数据丢失）
2 NF	噪声标志 在接收器中使用先进的采样技术使用了七个样品中的起始位和三个在每个数据位样本和停止位。如果任何这些样品不同于的其余样品在帧中的任何位周期，则标志 NF 置 1 同时 RDRF 会因为数据置 1。要 NF 清零，读 UART_S1 然后读 UART 数据寄存器（UART_D）。 0 没有检测到噪声 1 检测到噪声
1 FE	帧错误标志 FE 被置 1 在同一时间接收器检测到 RDRF 为逻辑 0，其中的停止位被预置。这表明接收器没有正确对齐字符帧。要清零 FE，读 FE 置 1 的 UART_S1 寄存器，然后读 UART 数据寄存器（UART_D）。 0 没有检测到帧错误。这并不保证帧一定正确 1 检测到帧错误
0 PF	奇偶检验错误标志 PF 置 1 在同一时间 RDRF 决定奇偶校验被启用时（PE = 1），所接收到的奇偶校验值与预期奇偶校验值不一致。要清除 PF，读 UART_S1 寄存器然后读取 UART 数据寄存器（UART_D）。 0 奇偶校验未检测到错误 1 奇偶校验检测到错误

● 7.4.6.6 UART 状态寄存器 2（UARTx_S2）

当使用在 LIN 系统内部的振荡器，有必要提高检测阈值位时间的间隔。在最坏的情况下时序并且 LIN 允许的条件下，如果从机比主机运行快 14%的情况下，0×00 的数据字符可能表现为 10.26 位时间长。这将引发正常间隔检测电路旨在检测 10 位间断的计划。当 LBKDE 位设置，帧错误从 10 位抑制，间隔检测阈值更改为 11 位，防止一个为 0x00 的数据字符作为 LIN 断开符号的错误检测。

地址：基址+ 5h 偏移 ：UART0_S2= 4006_A005h \ UART1_S2=4006_B005h \ UART2_S2=4006_C005h

位	7	6	5	4	3	2	1	0
读	LBKDIF	RXEDGIF	0	RXINV	RWUIR	BRK13	LBKDE	RAF
写								
复位值	0	0	0	0	0	0	0	0

表 7-29 UARTx_S2 字段描述

位	描述
7 LBKDIF	LIN 终止检测中断标志 当 LIN 间隔检测电路被启用并检测到 LIN 间隔字符是 LBKDIF 置 1。 LBKDIF 通过写 1 清零。 0 未检测到 LIN 间隔字符。 1 检测到 LIN 间隔字符。
6 RXEDGIF	RxD 引脚的有效边沿中断标志 RXEDGIF 置 1，当一个有效边沿，下降 RXINV=0，上升 RXINV=1，都在 RxD 引脚产生。 RXEDGIF 通过写 1 清零。 0 在接收引脚没有有效边沿产生 1 在接收引脚有有效边沿产生
5 保留	该位保留 该只读位保留并有且仅有值 0。
4 RXINV	接收数据反转 设置此位反转所接收的数据输入的极性 注：置 1 给 RXINV 将 RxD 输入设置在所有情况下反转：数据位，起始位和停止位，间隔，和空闲。 0 接收数据未反转 1 接收数据反转
3 RWUID	接收唤醒空闲检测 RWUID 控制着空闲字节是否唤醒接收器设置 IDIE 位。 0 在接收待命状态（RWU=1），没有检测到空闲字符 IDIE 位不置 1 1 在接收待命状态（RWU=1），检测到空闲字符 IDLE 位置 1
2 BRK13	间隔字符长度 BRK13 选择更长的发送间隔字符长度。该位的状态不会影响帧误差检测。 0 间隔字符按 10 位时间长度发送（如果 M =0，SBNS= 0）或 11（如果 M =1，SBNS=0 或 M= 0，SBNS= 1）或 12（如果 M =1，SBNS= 1）。 1 间隔字符按 13 位时间长度发送（如果 M =0，SBNS= 0）或 14（如果 M =1，SBNS=0 或 M= 0，SBNS= 1）或 15（如果 M =1，SBNS= 1）。
1 LBKDE	LIN 间隔检测使能 LBKDE 控制着间隔检测。当 LBKDE 置 1，帧错误（FE）和接收数据寄存器满（RDRF）标志设置将被阻止。 0 间隔检测使能关闭 1 间隔检测使能打开
0 RAF	有效接收器标志 当 UART 接收器检测到有效起始位 RAF 置 1，当接收器检测空闲线的时候 RAF 自动清零。这个状态位可以用来检查是否一个 UART 字符在 MCU 进入停止模式之前被接收。 0 UART 接收器空闲等待起始位。 1 UART 接收器有效（RxD 输入不空闲）。

● 7.4.6.7 UART 控制寄存器 3（UARTx_C3）

地址：基址+ 6h 偏移 ：UART0_C3= 4006_A006h \ UART1_C3=4006_B006h \ UART2_C3=4006_C006h

位	7	6	5	4	3	2	1	0
读	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
写								
复位值	0	0	0	0	0	0	0	0

表 7-30 UARTx_C3 字段描述

位	描述
7 R8	接收器第九位数据位 当 UART 被配置为 9 位数据,R8 可以被认为是作为第九位接收数据并处于 UART_D 寄存器中缓冲的数据的最高位的左边。当读 9 位数据，要在读 UART_D 之前读 R8 因为读取 UART_D 可以完成序列清零，这样可以让 R8 和 UART_D 被新的数据覆盖。
6 T8	发送器第九位数据位 当 UART 配置为 9 位数据（M= 1），T8 也可以认为是第九位发送数据并处于 UART_D 寄存器中的数据的最高位的左边。当写 9 位数据时，在 UART_D 被写入后整个 9 位值被转移到 UART 移位寄存器，如果需要从之前的值发生改变，在 UART_D 被写入之前。如果 T8 不需要为新的数据值变动，例如它被用于产生标记或空间奇偶校验，那么 UART_D 不需要每次被写入。
5 TXDIR	TxD 引脚在单线模式下的传输方向 当 UART 被配置为单线半双工操作模式（LOOPS=RSRC=1），该位决定了 TxD 引脚的数据传输方向。 0 在单线模式下 TxD 为输入 1 在单线模式下 TxD 为输出
4 TXINV	发送数据反转 设置此位反转所发送的数据的输出的极性。 注：置 1 给 TXINV 将 TxD 输出设置在所有情况下反转：数据位，开始位和停止位，间隔和空闲。 0 发送数据未反转 1 发送数据反转
3 ORIE	过载中断使能 该位控制着过载标志（OR）去产生硬件中断请求。 0 OR 中断关闭：使用轮询。 1 当 OR 置 1 有硬件中断请求
2 NEIE	噪声错误中断使能 该位控制着噪声标志去产生硬件中断请求 0 NF 中断关闭：使用轮询 1 当 NF 置 1 时有硬件中断请求
1 FEIE	帧错误中断使能 该位控制着帧错误标志去产生硬件中断请求 0 FE 中断关闭：使用轮询 1 当 FE 置 1 有硬件中断请求
0	奇偶校验中断使能

PEIE	该位打开了奇偶校验标志去产生硬件中断请求 0 PF 中断关闭：使用轮询 1 当 PF 置 1 有硬件中断请求

● 7.4.6.8 UART 数据寄存器（UARTx_D）

这个寄存器实际上是两个独立的寄存器。读取返回的只读内容接收数据缓冲器和写入到只写发送数据缓冲区。读取和写入该寄存器也参与了自动清除标志的机制。

地址: 基址+ 7h 偏移 ：UART0_D= 4006_A007h \ UART1_D=4006_B007h \ UART2_D=4006_C007h

位	7	6	5	4	3	2	1	0
读 写	R7T7	R6T6	R5T5	R4T4	R3T3	R2T2	R1T1	R0T0
复位值	0	0	0	0	0	0	0	0

表 7-31 UARTx_D 字段描述

位	名称	描述
7	R7T7	读接收器缓冲数据第 7 位或写发送器缓冲数据第 7 位
6	R6T6	读接收器缓冲数据第 6 位或写发送器缓冲数据第 6 位
5	R5T5	读接收器缓冲数据第 5 位或写发送器缓冲数据第 5 位
4	R4T4	读接收器缓冲数据第 4 位或写发送器缓冲数据第 4 位
3	R3T3	读接收器缓冲数据第 3 位或写发送器缓冲数据第 3 位
2	R2T2	读接收器缓冲数据第 2 位或写发送器缓冲数据第 2 位
1	R1T1	读接收器缓冲数据第 1 位或写发送器缓冲数据第 1 位
0	R0T0	读接收器缓冲数据第 0 位或写发送器缓冲数据第 0 位

■ 7.4.7 功能描述

UART 允许全双工异步的 NRZ 串行通信 MCU 和远程设备包括其它 MCU。该 UART 包括波特率发生器，发射器，和接收器模块。发射器和接收器操作独立，但它们使用相同的波特率发生器。在正常操作期间，单片机监视 UART 的状态，写入要发送的数据，并处理接收到的数据。各个模块的描述如下

● 7.4.7.1 波特率产生模块

UART 通信所需要的发射器和接收器，这通常获得波特率价格从独立的时钟源，使用相同的波特率。在允许偏差此波特频率取决于接收机如何同步到细节执行起始位。

该 MCU 重新同步位边界在每个高到低转换。在最坏的情况下，不存在这样的转变在全 10 位或 11 位或 12 bit time 字符框所以在波特率的不匹配都会积累在整个字符时间。对于 UART 系统，其总线频率有一个晶振得到，所允许的波特率速率失配是为 8 位数据格式±4.5%和大约±4%为 9 位数据格式。虽然波特率分频因子的设置并不总是产生完全符合标准的波特率，通常可以在一个很小的百分比误差，这是可以接受的可靠的通信。

如下图所示，UART 波特率发生器的时钟源是总线时钟。

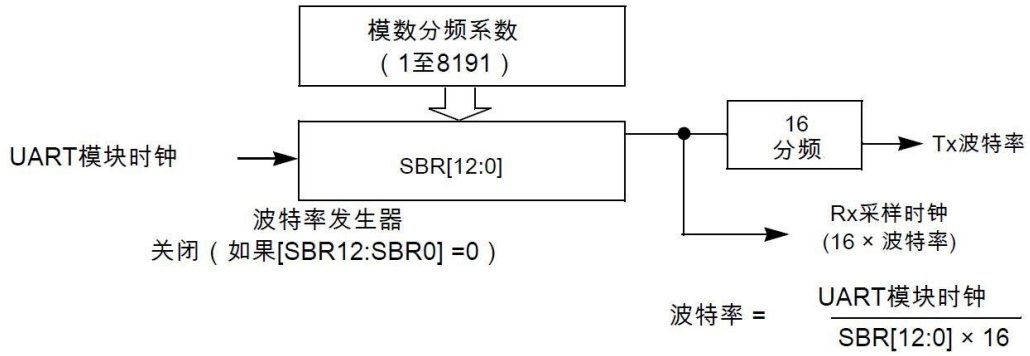


图 7-16 UART 波特率生成

● 7.4.7.2 发送器功能描述

本节描述的整体框图 UART 发送器，以及发送终止的特殊功能和空闲字符。发送器输出（TXD）空闲状态默认是逻辑高，UART_C3[TXINV]复位后清零。变送器输出的反转由 UART_C3[TXINV]设定。变送器通过设置 UART_C2 TE 位启用。发送序列的前导字符是一个空闲状态的满字符帧。发射机保持空闲状态，直到数据在发送数据缓冲区可用。程序存储数据到通过写至 UART 的数据寄存器（UART_D）发送数据缓冲区。

UART 传送器的中心元件是传送移位寄存器，它有 10、11 或 12 位长，具体取决于 UART_C1[M]控制位和 UART_BDH[SBNS]位的设置。本节其余部分假设 UART_C1[M]被清零，UART_BDH[SBNS]也被清零，选择常规 8 位数据模式。在 8 位数据模式下，移位寄存器占用一个起始位、八个数据位和一个停止位。当传送移位寄存器可载入新 UART 字符时，在传送数据寄存器中等待的值会被传输到移位寄存器，与波特率时钟同步，并且传送数据寄存器空(UART_S1[TDRE])状态标志置位，指示可将另一个字符写入 UART_D 处的传送数据缓冲区。务必先对 UART_S1 进行读操作，再对 UART_D 进行写操作，以便能够传送数据。

TxD 引脚移出一个停止位后，如果传送数据缓冲区中没有新字符在等待，那么传送器会置位传送完成标志并进入空闲模式，且 TxD 为高电平，等待传送更多字符。将 0 写入 UART_C2[TE]不会立即解除该引脚的状态使其变为通用 I/O 引脚。必须首先完成任何正在进行的传送活动。这包括正在传送的数据字符、排队的空闲字符和断点字符。

■ 7.4.7.2.1 发送断点和排队空闲字符

UART_C2[SBK]发送断点字符，其原始用途是引起旧式电传打字接收机的注意。断点字符是具有完整字符时间的逻辑 0，共 10 位时间，包括起始位和停止位。通过置位 UART_S2[BRK13]可使能 13 位时间的更长断点。通常情况下，程序会等待 UART_S1[TDRE]置位以指示消息的最后一个字符已被移入发送移位器，然后向 UART_C2[SBK]写入 1，再写入 0。该操作将一个断点字符加入发送队列，一旦移位器可用就会发送。当排队的断点字符移入移位器时（与波特率时钟同步），如果 UART_C2[SBK]仍然为 1，则队列中会再增加一个断点字符。如果接收器件是相同协议的 UART，那么所有八个数据位中接收到的断点字符都为 0，并且会发生帧传输错误(UART_S1[FE] = 1)。

使用空闲线路唤醒时，消息之间需要具有完整字符时间的空闲（逻辑 1）以唤醒任何处于睡眠状态的接收器。通常，程序会等待 UART_S1[TDRE]置位以指示消息的最后一个字符已被移入发送移位器，然后向 UART_C2[TE]位写入 0，再写入 1。该操作将一个空闲字符加入发送队列，一旦移位器可用就会发送。清除 UART_C2[TE]时，只要移位器中的字符未完成发送，UART 发送器就不会真正解除对 TxD 引脚的控制。如果存在这样一种可能性：清除 UART_C2[TE]的同时移位器正在完成发送，那么设置通用 I/O 控制，这样与 TxD 共用的引脚就

是一个驱动逻辑 1 的输出。如此一来，即使在向 UART_C2[TE]先写入 0 再写入 1 期间 UART 失去对端口引脚的控制，TxD 线路看起来也是正常的空闲线路。

断点字符的长度受 UART_S2[BRK13]和 UART_C1[M]影响，如下表所示。

表 7-32 断点字符长度

BRK13	M	SBNS	断点字符长度
0	0	0	10 位时间
0	0	1	11 位时间
0	1	0	11 位时间
0	1	1	12 位时间
1	0	0	13 位时间
1	0	1	14 位时间
1	1	0	14 位时间
1	1	1	15 位时间

● 7.4.7.3 接收器功能描述

在本章中，接收器框图是整个接收功能的指南描述。接着，数据采样技术用于重建接收数据是更详细地描述。最后，接收器唤醒功能的两种变化解释。接收机输入端通过设置 UART_S2[RXINV]反转。该接收器由 UART_C2[RE]位设置使能。字符帧包括逻辑 0 的起始位，八个（或 9）数据位（LSB 在先），和一个（或两个）逻辑 1 的停止位。对于本次讨论的其余部分，UART 配置为正常的 8 位数据模式进行配置。

接收移位器接收到停止位后，如果接收数据寄存器未满，那么数据字符会被转移到接收数据寄存器，然后接收数据寄存器满(UART_S1[RDRF])状态标志置位。如果 UART_S1[RDRF]已置位指示接收数据寄存器(缓冲区）已满，那么溢出(OR)状态标志置位且新数据丢失。由于 UART 接收器采用双缓冲区，因此程序在 UART_S1[RDRF]置位后以及必须读取接收数据缓冲区中数据以免接收器溢出之前有一个完整的字符时间。

当程序检测到接收数据寄存器已满(UART_S1[RDRF] = 1)时，它可以通过读取 UART_D 从接收数据寄存器获得数据。UART_S1[RDRF]标志由一个两步骤序列自动清除，管理接收数据的用户程序在运行过程中一般会满足该序列。有关标志清除的更多信息，请参见中断和状态标志 。

■ 7.4.7.3.1 数据采样技术

UART 接收器利用 16 倍波特率时钟进行采样。过采样比是固定值 16。接收器启动后，以 16 倍的波特率采集逻辑电平样本，寻找 RxD 串行数据输入引脚上的下降沿。下降沿的定义是连续三个逻辑 1 样本后出现一个逻辑 0 样本。16 倍波特率时钟将位时间分割为 16 段，标记为 UART_D[RT1]至 UART_D[RT16]。找到一个下降沿时，在 UART_D[RT3]、UART_D[RT5]和 UART_D[RT7]处再采集三个样本，确保它是真正的起始位，而不只是噪声。如果这三个样本中至少有两个为 0，那么接收器就会认定它与接收字符同步。接收器随后在 UART_D[RT8]、UART_D[RT9]和 UART_D[RT10]处对每个位时间进行采样，包括起始位和停止位，以确定该位的逻辑电平。某一位的逻辑电平是指该位时间内采集的大多数样本电平。对于起始位，如果在 UART_D[RT3]、UART_D[RT5]和 UART_D[RT7]处采集的样本中至少有两个为 0，那么即使在 UART_D[RT8]、UART_D[RT9]和 UART_D[RT10]处采集的样本中有一个或全部是 1，也认为该位为 0。如果某个字符帧中的任何样本在任何位时间（包括起始位和停止位）都与该位的逻辑电平不一致，那么当已接收到的字符转移到接收数据缓冲区时，噪声标志(UART_S1[NF])置位。

下降沿检测逻辑持续寻找下降沿。检测到一个下降沿时，采样时钟便与位时间重新同步。在有噪声或波特率不匹配的情况下，这可以提高接收器的可靠性。它不能改善最差情况分析，因为某些字符在字符帧

中的任何地方都没有多余的下降沿。发生帧传输错误时，如果已接收到的字符不是断点字符，那么寻找下降沿的采样逻辑将用三个逻辑 1 样本填充，这样就几乎能立即检测到新起始位。发生帧传输错误时，禁止接收器接收任何新字符，直到帧传输错误标志被清除为止。接收移位寄存器继续工作，但如果 UART_S1[FE]仍然置位，就无法将完整字符转移到接收数据缓冲区。

■ 7.4.7.3.2 接收器唤醒操作

接收器唤醒是一种硬件机制，使 UART 接收器能忽略用于其他 UART 接收器的消息中的字符。在此类系统中，各接收器都会评估每条消息的首字符，一旦确定其用于其他接收器，就会将逻辑 1 写入接收器唤醒控制字段(UART_C2[RWU])。当 UART_C2[RWU]置位时，禁止与接收器相关的状态标志（UART_S2[RWUID]置位时空闲位 IDLE 除外）置位，从而消除软件处理不重要消息字符的开销。消息结束时，或下一消息开始时，所有接收器都自动将 UART_C2[RWU]强制为 0，因此所有接收器都会及时醒来，以便检查下一消息的首字符。

（一）空闲线路唤醒

唤醒位被清除时，接收器配置为空闲线路唤醒。在该模式下，当接收器检测到完整字符时间的空闲线路电平时，UART_C2[RWU]被自动清除。UART_C1[M]控制字段选择 8 位或 9 位数据模式，UART_BDH[SBNS]选择 1 个或 2 个停止位，从而确定需要多少位时间的空闲才能构成完整字符时间：10 位、11 位或 12 位时间（包括起始位和停止位）。

当 UARTI_C2[RWU]为 1 且 UART_S2[RWUID]为 0 时，唤醒接收器的空闲条件不会置位 UART_S1[IDLE]。接收器唤醒后，等待下一消息的第一个数据字符，由后者置位 UART_S1[RDRF]并成生中断（若使能）。当 UART_S2[RWUID]为 1 时，任何空闲条件都会置位 UART_S1[IDLE]标志并生成中断（若使能），无论 UART_C2[RWU]是 0 还是 1。

空闲线路类型(UART_C1[ILT])控制位选择两种空闲线路检测方式中的一种。当 UART_C1[ILT]被清除时，空闲位计数器在起始位之后开始计数，因此字符结束时的停止位和任何逻辑 1 都会计数到空闲的完整字符时间中。当 UART_C1[ILT]置位时，空闲位计数器要等到停止位时间之后才开始计数，因此空闲检测不受上一消息的最后一个字符中的数据影响。

（二）地址标记唤醒

设置唤醒后，会针对地址标记唤醒配置接收器。在该模式中，如果 UART_BDH[SBNS] = 1，则当接收器检测到已接收字符的最高有效位（清除 UART 时为第八位，UART _C1[M]置位时为第九位）中有一到两个逻辑 1 时，将自动清除 UART_C2[RWU]。

地址标记唤醒允许消息包含闲置字符，但需要保留最高有效位以用于地址帧中。如果 UART_BDH[SBNS] = 1，则地址帧最高有效位中的这一到两个逻辑 1 会在接收到停止位之前清除 UART_C2[RWU]位并设置 UART_S1[RDRF]标志。在这种情况下，即使接收器在具有最高有效位集的字符的大部分时间里都处于睡眠状态，还是会收到此字符。

● 7.4.7.4 中断和状态标志

UART 系统具有一个中断向量，它有三种不同的中断类型。一个中断类型与发送器相关，用于 UART_S1[TDRE]和 UART_S1[TC]事件。另一个中断类型与接收器相关，用于 RDRF、IDLE、RXEDGIF 和 LBKDIF 事件。第三个类型用于 OR、NF、FE 和 PF 错误条件。这十个中断源各自都可以通过本地中断使能掩码加以屏蔽。当清除本地掩码以禁用硬件中断请求生成时，可通过软件轮询这些标志。

UART 发送器具有两个状态标志，它们可以选择生成硬件中断请求。发送数据寄存器空(UART_S1[TDRE])指示发送数据缓冲区中有空间将另一个发送字符写入 UART_D。如果发送中断使能(UART_C2[TIE])位置位，

则当 UART_S1[TDRE]置位时，就会请求硬件中断。发送完成(UART_S1[TC])指示发送器已完成所有数据、前同步码和断点字符的发送，现处于空闲状态，且 TxD 处于无效电平。该标志常用于带调制解调器的系统，以便确定何时可以安全关闭调制解调器。如果发送完成中断使能(UART_C2[TCIE])位置位，则当 UART_S1[TC]置位时，就会请求硬件中断。如果清除对应的 UART_C2[TIE]或 UART_C2[TCIE]本地中断掩码，那么就可利用软件轮询代替硬件中断来监控 UART_S1[TDRE]和 UART_S1[TC]状态标志。

当程序检测到接收数据寄存器已满(UART_S1[RDRF]=1)时，它可以通过读取 UART_D 从接收数据寄存器获得数据。在 UART_S1[RDRF]置位的情况下读取 UART_S1，然后读取 UART_D，即可清除 UART_S1[RDRF]标志。

使用轮询时，这一序列在用户程序的正常流程中会自然得到满足。如果使用硬件中断，则必须在中断服务例程(ISR)中读取 UART_S1。通常情况下，这总是在 ISR 中完成以检查有无接收错误，因此会自动满足该序列。

IDLE 状态标志包含一个逻辑，其在 RxD 线路长时间空闲时可防止该标志重复置位。在 UART_S1[IDLE]置位的情况下读取 UART_S1，然后读取 UART_D，即可清除 IDLE。清除 UART_S1[IDLE]之后，要再次将其置位，则必须等到接收器接收到至少一个新字符且已置位 UART_S1[RDRF]之后。如果在引起 UART_S1[RDRF]置位的接收字符中检测到相关错误，则错误标志——噪声标志(UART_S1[NF])、帧传输错误(UART_S1[FE])和奇偶校验错误标志(UART_S1[PF])将与 UART_S1[RDRF]同时置位。这些标志在溢出情况下不会置位。

当一个新字符准备好从接收移位器转移到接收数据缓冲区时，如果 UART_S1[RDRF]已经置位，则溢出(UART_S1[OR])标志将置位，数据以及任何相关的 NF、FE、PF 条件都会丢失。

无论何时，RxD 串行数据输入引脚上的有效边沿都会使 UART_S2[RXEDGIF]标志置位。UART_S2[RXEDGIF]标志可通过向其写入 1 来清除。该功能要求接收器使能(UART_C2[RE] = 1)。

● 7.4.7.5 波特率公差

传送器件工作时的波特率可能低于或高于接收器工作时的波特率。

累积的位时间偏差可导致三个停止位数据样本（RT8、RT9 和 RT10）之一超出实际停止位之外。如果 RT8、RT9 和 RT10 样本的逻辑值不全相同，就会产生噪声误差。如果接收器时钟发生偏差，即大部分 RT8、RT9 和 RT10 停止位样本值为逻辑零，则将产生成帧错误。当接收器对传入的帧进行采样时，它在此帧的任何有效上升沿上重新与 RT 时钟进行同步。在帧中进行重新同步将纠正传送位时间与接收位时间之间的偏差。

■ 7.4.7.5.1 慢速数据公差

图 7-12 显示了慢速接收帧最多可以有多大的对齐误差而不引起噪声错误或帧传输错误。慢速停止位开始于 RT8，而非 RT1，但会及时到达，以便在 RT8、RT9 和 RT10 进行停止位数据采样。

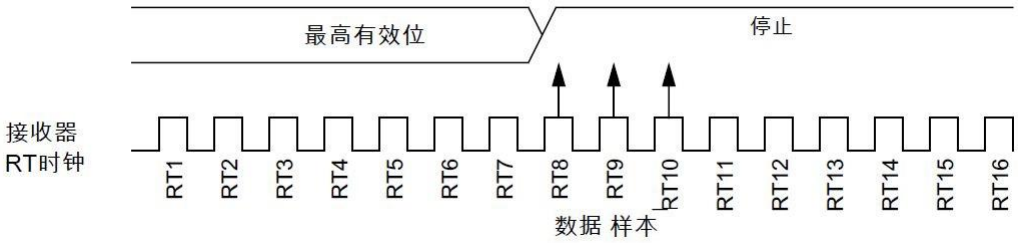


图 7-17 慢速数据

对于 8 位数据和 1 停止位的字符，停止位的数据采样需要接收器 9 位时间 x 16 RT 周期 + 10 RT 周期 = 154RT 周期。

对于图 7-12 所示的未对齐字符，当传送器计数到 9 位时间 x 16 RT 周期 + 3RT 周期 = 147RT 周期时，

接收器计数到 154 RT 周期。

在无误差的情况下，慢速 8 位数据和 1 停止位的字符的接收器计数与传送器计数之间的最大百分比差异为：((154 - 147) / 154) x 100 = 4.54%

对于 9 位数据或 2 停止位的字符，停止位的数据采样需要接收器 10 位时间 x 16RT 周期 + 10RT 周期 = 170RT 周期。

对于图 7-12 所示的未对齐字符，当传送器计数到 10 位时间 x 16RT 周期 +3RT 周期 =163RT 周期时，接收器计数到 170RT 周期。

在无误差的情况下，慢速 9 位或 2 停止位的字符的接收器计数与传送器计数之间的最大百分比差异为：((170 - 163) / 170) X 100 = 4.12%

对于 9 位数据和 2 停止位的字符，停止位的数据采样需要接收器 11 位时间 x 16 RT 周期 + 10 RT 周期 = 186RT 周期。

对于图 7-12 所示的未对齐字符，当传送器计数到 11 位时间 x 16 RT 周期 + 3RT 周期 = 179RT 周期时，接收器计数到 186RT 周期。在无误差的情况下，慢速 9 位和 2 停止位的字符的接收器计数与传送器计数之间的最大百分比差异为：((186 - 179) / 186) X 100 = 3.76%

■ 7.4.7.5.2 快速数据公差

图 7-13 显示了快速接收帧允许的最大对齐误差。快速停止位结束于 RT10,而非 RT16,不过仍然在 RT8、RT9 和 RT10 进行采样。

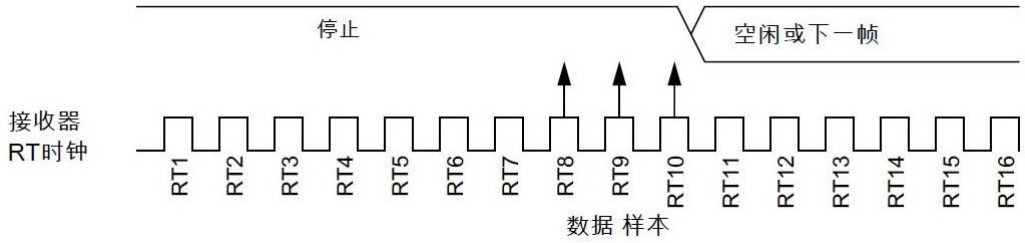


图 7-18 快速数据

对于 8 位数据和 1 停止位的字符，停止位的数据采样需要接收器 9 位时间 x 16RT 周期 + 10RT 周期 = 154RT 周期。

对于图 7-13 所示的未对齐字符，当传送器件计数到 10 位时间 x 16RT 周期 = 160RT 周期时，接收器计数到 154RT 周期。

一个快速 8 位和 1 停止位的无错误字符的接收器计数与传送器计数之间的最大百分比差异为：((154 - 160) / 154) x 100 = 3.90%

对于 9 位数据或 2 停止位的字符，停止位的数据采样需要接收器 10 位时间 x 16RT 周期 +10RT 周期 = 170RT 周期。

对于所示的未对齐字符，当传送器件计数到 11 位时间 x 16RT 周期 = 176RT 周期时，接收器计数到 170RT 周期。

一个快速 9 位或 2 停止位的无错误字符的接收器计数与传送器计数之间的最大百分比差异为：((170 - 176) / 170) x 100 = 3.53%

对于 9 位数据和 2 停止位的字符，停止位的数据采样需要接收器 11 位时间 x 16RT 周期 +10RT 周期 = 186RT 周期。

对于所示的未对齐字符，当传送器件计数到 12 位时间 x 16RT 周期 =193RT 周期时，接收器计数到 186 RT 周期。

一个快速 9 位和 2 停止位的无错误字符的接收器计数与传送器计数之间的最大百分比差异为：

((186 - 192) / 186) x 100 = 3.23%

● 7.4.7.6 其他 UART 功能

以下各节说明了 UART 其他功能。

■ 7.4.7.6.1 8 位和 9 位数据模式

通过设置 UART_C1[M]，可将 UART 系统、发送器和接收器配置为以 9 位数据模式工作。在 9 位模式下，UART 数据寄存器的最高有效位左边有一个第九数据位。对于发送数据缓冲区，该位存储在 UART_C3 的 T8 中。对于接收器，第九位保持在 UART_C3[R8]中。

为了一致地写入发送数据缓冲区，应先写入 UART_C3[T8]，再写入 UART_D。如果作为新字符第九位的待发送位值与前一字符的该位值相同，则无需再次写入 UART_C3[T8]。当数据从发送数据缓冲区转移到发送移位器时，UART_C3[T8]中的值在数据从 UART_D 转移到移位器的同时复制。

9 位数据模式通常配合奇偶校验使用，以支持 8 位数据并将第九位用于奇偶校验，或者配合地址标记唤醒使用，第九数据位用作唤醒位。在自定义协议中，第九位也可用作软件控制标记。

■ 7.4.7.6.2 停止模式操作

在所有停止模式中，UART 模块的时钟都会暂停。任何 UART 模块寄存器在停止模式下都不会受到影响。

接收输入有效边沿检测电路在停止模式下仍然有效。中断未被屏蔽(UART_BDH[RXEDGIE] = 1)的情况下，接收输入上的有效边沿会使 CPU 退出停止模式。

由于时钟暂停，只有在停止模式下，UART 模块才能在退出停止模式时恢复工作。当 UART 模块正在发送或接收字符（包括前同步码、断点和普通数据）时，软件必须确保它不进入停止模式，也就是说，要进入停止模式，必须满足以下全部条件：UART_S1[TC] =1、UART_S1[TDRE] = 1 且 UART_S2[RAF] = 0。

■ 7.4.7.6.3 循环模式

UART_C1[LOOPS]置位后，相同寄存器中的 UART_C1[RSRC]位选择循环模式(UART_C1[RSRC] = 0)或单线模式(UART_C1[RSRC] = 1)。循环模式有时用于检查软件（与外部系统中的连接无关）以帮助隔离系统问题。在此模式下，从发送器到接收器的内部回环连接会导致接收器接收由发送器发出的字符。

■ 7.4.7.6.4 单线操作

当UART_C1[LOOPS]置位时,UART_C1[RSRC]选择环路模式(UART_C1[RSRC] =0)或单线模式(UART_C1[RSRC] = 1)。单线模式采用半双工串行连接。接收器内部连接到发送器输出和 TxD 引脚。RxD 引脚不使用，恢复为通用端口 I/O 引脚。

在单线模式下，UART_C3[TXDIR]位控制 TxD 引脚上串行数据的方向。当 UART_C3[TXDIR]被清除时，TxD 引脚是 UART 接收器的输入，发送器暂时与 TxD 引脚断开，以便外部器件能向接收器发送串行数据。当 UART_C3[TXDIR]置位时，TxD 引脚是由发送器驱动的输出。在单线模式下，发送器输出内部连接到接收器输入且 UART 不使用 RxD 引脚，因此它恢复为通用端口 I/O 引脚。

第 8 章 人机接口模块

■ 8.1 简介

本章对基于 ARM Cortex-M0+内核的 NV32F100x 系列 MCU 人机接口相关的模块做了详细说明，其中包括管脚信号分配、端口控制 PORT、通用输入/输出 GPIO、键盘中断 KBI、外部中断 IRQ 等模块内容。

■ 8.2 管脚信号分配模块

■ 8.2.1 简介

为了优化小型封装的功能，引脚通过信号多路复用技术实现多种可用功能。本节说明该器件的哪些信号在哪个外部引脚上多路复用。

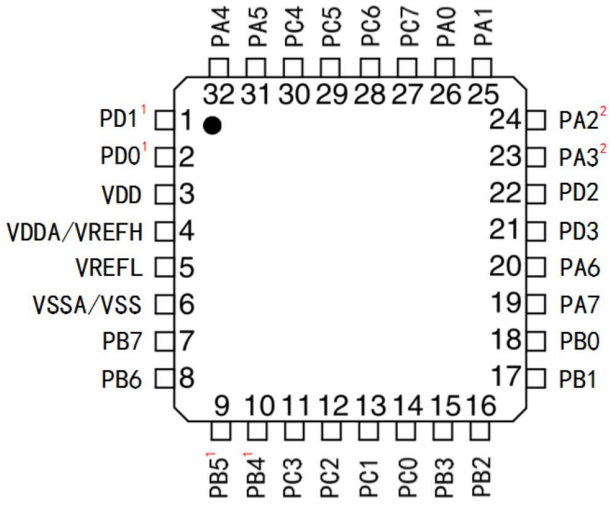
引脚选择寄存器 0(SIM_PINSEL0)和引脚选择寄存器 1(SIM_PINSEL1)控制该外部引脚上有哪些信号。有关特定多路复用引脚控制操作的详细信息，请参考该寄存器的相关内容。

■ 8.2.2 引脚分配

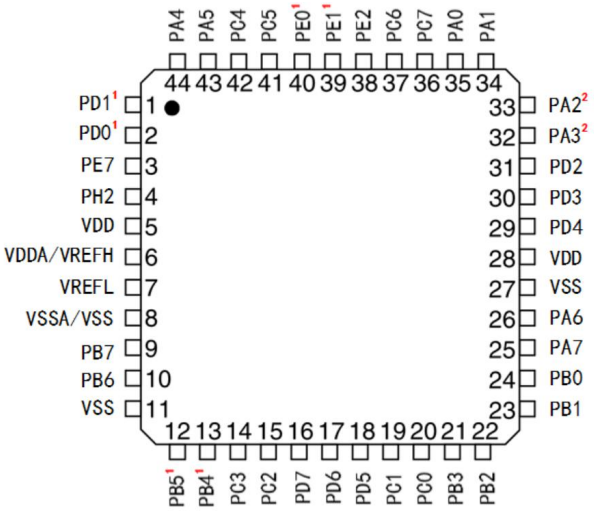
● 8.2.2.1 信号多路复用和引脚分配

下表显示的是各引脚上的信号以及这些引脚在本文档支持的器件上的位置。“端口控制模块”负责选择每个引脚上的 ALT 功能。

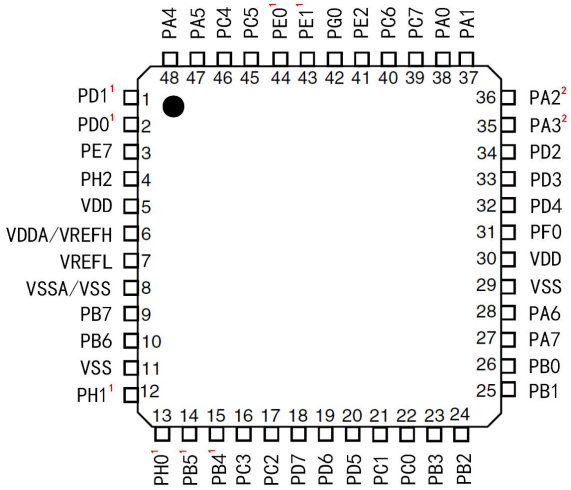
注：VSS 与 VSSA 在内部相连。VREFH 与 VDDA 在 64 引脚封装中内部相连。PB4、PB5、PD0、PD1、PE0、PE1、PH0 和 PH1 用作输出时为大电流驱动引脚。PA2 和 PA3 在用作输出时为有效的开漏引脚。



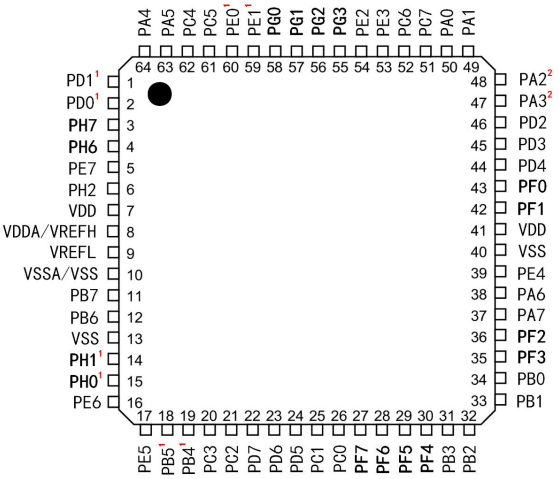
LQFP32 封装



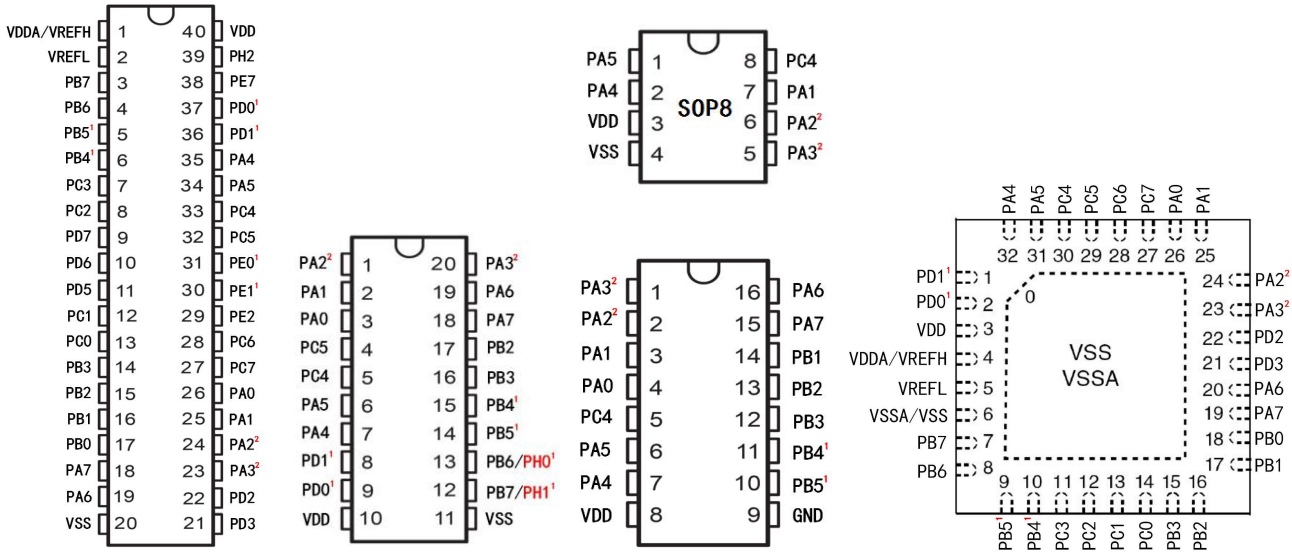
LQFP44 封装



LQFP48 封装



PQFP64/LQFP64 封装



注释： 1. 大电流管脚 2. 开漏管脚

图 8-1 引脚分配

表 8-1 管脚分配和名称

管脚数编号								优先级 低 ---> 高				
64	48	44	40	32	20	16	8	管脚名	功能 1	功能 2	功能 3	功能 4
1	1	1	36	1	8	–	–	PD1[1]	KBI1_P1	ETM2_CH3	SPI1_MOSI	–
2	2	2	37	2	9	–	–	PD0[1]	KBI1_P0	ETM2_CH2	SPI1_SCK	–
3	–	–	–	–	–	–	–	PH7	–	–	–	–
4	–	–	–	–	–	–	–	PH6	–	–	–	–
5	3	3	38	–	–	–	–	PE7	–	ETM2_CLK	–	ETM1_CH1
6	4	4	39	–	–	–	–	PH2	–	BUSOUT	–	ETM1_CH0
7	5	5	40	3	10	8	3	–	–	–	–	VDD
8	6	6	1	4	10	8	3	–	–	–	VDDA	VREFH[4]

管脚数编号								优先级 低 ---> 高				
64	48	44	40	32	20	16	8	管脚名	功能 1	功能 2	功能 3	功能 4
9	7	7	2	5	11	9	4	–	–	–	–	VREFL
10	8	8	–	6	11	9	4	–	–	–	VSSA	VSS[3]
11	9	9	3	7	12	–	–	PB7	–	I2C0_SCL	–	EXTAL
12	10	10	4	8	13	–	–	PB6	–	I2C0_SDA	–	XTAL
13	11	11	–	–	–	–	–	–	–	–	–	VSS
14	12	–	–	–	12[5]	–	–	PH1[1]	–	ETM2_CH1	–	–
15	13	–	–	–	13[5]	–	–	PH0[1]	–	ETM2_CH0	–	–
16	–	–	–	–	–	–	–	PE6	–	–	–	–
17	–	–	–	–	–	–	–	PE5	–	–	–	–
18	14	12	5	9	14	10	–	PB5[1]	ETM2_CH5	SPI0_PCS0	ACMP1_OUT	–
19	15	13	6	10	15	11	–	PB4[1, 6]	ETM2_CH4	SPI0_MISO	$\overline{\text{NMI}}$	ACMP1_IN2
20	16	14	7	11	–	–	–	PC3	ETM2_CH3	–	–	ADC0_SE11
21	17	15	8	12	–	–	–	PC2	ETM2_CH2	–	–	ADC0_SE10
22	18	16	9	–	–	–	–	PD7	KBI1_P7	UART2_TX	–	–
23	19	17	10	–	–	–	–	PD6	KBI1_P6	UART2_RX	–	–
24	20	18	11	–	–	–	–	PD5	KBI1_P5	–	–	–
25	21	19	12	13	–	–	–	PC1	–	ETM2_CH1	–	ADC0_SE9
26	22	20	13	14	–	–	–	PC0	–	ETM2_CH0	–	ADC0_SE8
27	–	–	–	–	–	–	–	PF7	–	–	–	ADC0_SE15
28	–	–	–	–	–	–	–	PF6	–	–	–	ADC0_SE14
29	–	–	–	–	–	–	–	PF5	–	–	–	ADC0_SE13
30	–	–	–	–	–	–	–	PF4	–	–	–	ADC0_SE12
31	23	21	14	15	16	12	–	PB3	KBI0_P7	SPI0_MOSI	ETM0_CH1	ADC0_SE7
32	24	22	15	16	17	13	–	PB2	KBI0_P6	SPI0_SCK	ETM0_CH0	ADC0_SE6
33	25	23	16	17	–	14	–	PB1	KBI0_P5	UART0_TX	–	ADC0_SE5
34	26	24	17	18	–	–	–	PB0	KBI0_P4	UART0_RX	–	ADC0_SE4
35	–	–	–	–	–	–	–	PF3	–	–	–	–
36	–	–	–	–	–	–	–	PF2	–	–	–	–
37	27	25	18	19	18	15	–	PA7	–	ETM2_FLT2	ACMP1_IN1	ADC0_SE3
38	28	26	19	20	19	16	–	PA6	–	ETM2_FLT1	ACMP1_IN0	ADC0_SE2
39	–	–	–	–	–	–	–	PE4	–	–	–	–
40	29	27	20	–	–	–	–	–	–	–	–	VSS
41	30	28	–	–	–	–	–	–	–	–	–	VDD
42	–	–	–	–	–	–	–	PF1	–	–	–	–
43	31	–	–	–	–	–	–	PF0	–	–	–	–
44	32	29	–	–	–	–	–	PD4	KBI1_P4	–	–	–
45	33	30	21	21	–	–	–	PD3	KBI1_P3	SPI1_PCS0	–	–
46	34	31	22	22	–	–	–	PD2	KBI1_P2	SPI1_MISO	–	–
47	35	32	23	23	20	1	5	PA3[2]	KBI0_P3	UART0_TX	I2C0_SCL	–

管脚数编号								优先级 低 ---> 高				
64	48	44	40	32	20	16	8	管脚名	功能 1	功能 2	功能 3	功能 4
48	36	33	24	24	1	2	6	PA2[2]	KBI0_P2	UART0_RX	I2C0_SDA	-
49	37	34	25	25	2	3	7	PA1	KBI0_P1	ETM0_CH1	ACMP0_IN1	ADC0_SE1
50	38	35	26	26	3	4	-	PA0	KBI0_P0	ETM0_CH0	ACMP0_IN0	ADC0_SE0
51	39	36	27	27	-	-	-	PC7	-	UART1_TX	-	-
52	40	37	28	28	-	-	-	PC6	-	UART1_RX	-	-
53	-	-	-	-	-	-	-	PE3	-	SPI0_PCS0	-	-
54	41	38	29	-	-	-	-	PE2	-	SPI0_MISO	-	-
55	-	-	-	-	-	-	-	PG3	-	-	-	-
56	-	-	-	-	-	-	-	PG2	-	-	-	-
57	-	-	-	-	-	-	-	PG1	-	-	-	-
58	42	-	-	-	-	-	-	PG0	-	-	-	-
59	43	39	30	-	-	-	-	PE1[1]	-	SPI0_MOSI	-	-
60	44	40	31	-	-	-	-	PE0[1]	-	SPI0_SCK	ETM1_CLK	-
61	45	41	32	29	4	-	-	PC5	-	ETM1_CH1	-	RTC0
62	46	42	33	30	5	5	8	PC4	RTC0	ETM1_CH0	ACMP0_IN2	SWD_CLK
63	47	43	34	31	6	6	1	PA5	IRQ	ETM0_CLK	-	$\overline{\text{RESET}}$
64	48	44	35	32	7	7	2	PA4	-	ACMP0_OUT	-	SWD_DIO

注释： 1. 做输出管脚时为大电流管脚 2. 做输出管脚时为开漏状态 3. VSSA 和 VSS 芯片内部是连接的

4. VERFH 和 VDDA 芯片内部是连接的 5. TSSOP20 无外置晶振封装形式，编号 NV32F100xT20**B**(x=D,E,F)

6. PA4,PA5,PB4,PC4 复位后，默认功能不是 GPIO，如用 GPIO 功能需配置相关寄存器

■ 8.2.3 模块信号描述

下面的章节描述芯片级信号名称与模块章节中使用的信号名称关联，还简要介绍信号功能和方向。

表 8-2 信号说明

芯片信号名称	模块名称	说明	I/O
SWD_DIO	内核模块	串行线调试数据输入/输出。外部调试工具通过 SWD_DIO 引脚进行通信和器件控制。该引脚在内部上拉。	I/O
SWD_CLK	内核模块	串行线时钟。该引脚在串行线调试模式下作为调试逻辑的时钟。[1]	I
$\overline{\text{NMI}}$	系统模块	非屏蔽中断 注意：如果相应引脚选择$\overline{\text{NMI}}$功能，那么将$\overline{\text{NMI}}$信号驱动至低电平会强制生成非屏蔽中断。	I/O
$\overline{\text{RESET}}$	系统模块	复位双向信号	I/O
VDD	电源	MCU 电源	I
VSS	电源	MCU 接地	I
EXTAL	OSC 模块	外部时钟/振荡器输入	I
XTAL	OSC 模块	振荡器输出	O
ADC0_SEn	ADC	模拟通道输入	I
VDD/VREFH	模拟电源	模拟电源/基准电压源高电平	I
VSS/VREFL	模拟地	模拟电源地/基准电压源低电平	I

ACMPn_INn	比较器	模拟电压输入	I
ACMPn_OUT	比较器	比较器输出	O
ETMn_CLK	定时器	ETM 外部时钟	I
ETMn_CH[1:0]	定时器	ETM 通道	I/O
ETMn_FLT[2:1]	定时器	ETM 故障输入，需要使能 ETM2_FLTCTRL 寄存器的故障输入位	I
RTC_CLKOUT	RTC	RTC 时钟输出	O
SPIn_MISO	SPI	主机数据输入，从机数据输出	I/O
SPIn_MISI	SPI	主机数据输出，从机数据输入	I/O
SPIn_SCK	SPI	SPI 串行时钟	I/O
SPIn_PCS	SPI	从机选择	I/O
I2C_SCL	I2C	I2C 系统的双向串行时钟线路	I/O
I2C_SDA	I2C	I2C 系统的双向串行数据线路	I/O
UARTn_TX	UART	发送数据	I/O
UARTn_RX	UART	接收数据	I
Px[7：0]	GPIO	GPIO	I/O
KBIx_Pn	KBI	键盘中断引脚,n 可以是 0~31	I/O
IRQ	IRQ	IRQ 输入	I/O

注：1.该器件不支持片上下拉；SWD_CLK 引脚仅支持有 PE0 控制的上拉，完全支持 SWD 协议需要外部下拉电阻。

■ 8.3 端口控制（PORT）模块

■ 8.3.1 简介

该器件具有八组 I/O 端口，包括多达 57 个通用 I/O 引脚。并非所有器件都提供全部引脚。很多 I/O 引脚都共用片上外设功能。外设模块的优先级高于 I/O，因此当启用外设时，会禁用相关的 I/O 功能。复位后，共用的外设功能禁用，因此这些引脚通过并行 I/O 控制，但默认关联至 SWD_DIO、SWD_CLK、NMI 和 RESET 功能的 PA4、PA5、PB4 和 PC4 除外。所有并行 I/O 都配置为高阻抗状态(Hi-Z)。每个引脚的引脚控制功能配置如下：

- 输入禁用(GPIOx_PIDR[PID] = 1)，
- 输出禁用(GPIOx_PDDR[PDD] = 0)，
- 内部上拉禁用(PORT_PUE(L/H)[PxPEn] = 0)。

此外，支持高驱动强度能力的并行 I/O 在复位后禁用(HDRVE = 0x00)。

以下三张图显示每个 I/O 引脚的结构。

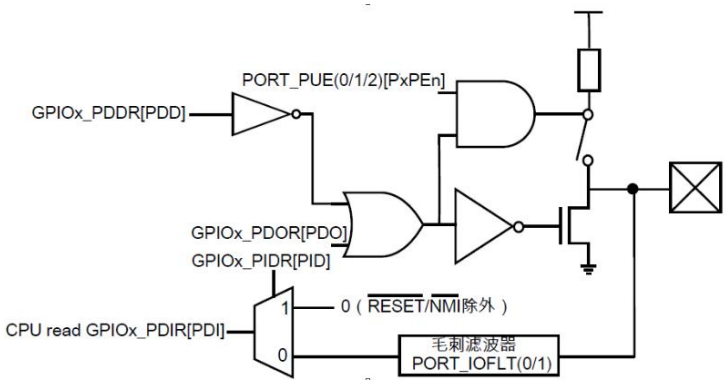


图 8-2 普通 I/O 结构

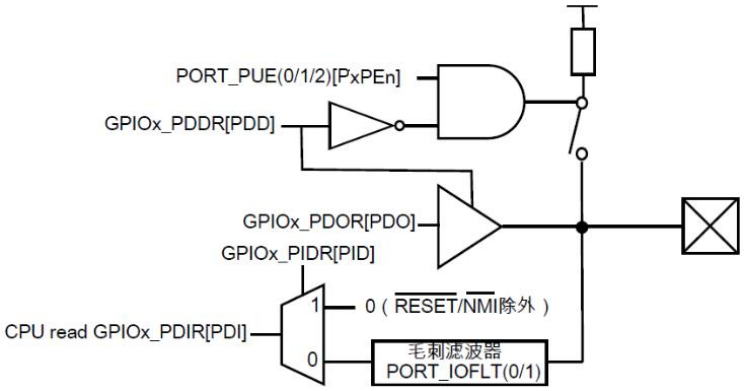


图 8-3 SDA(PA2)/SCL（PA3)结构

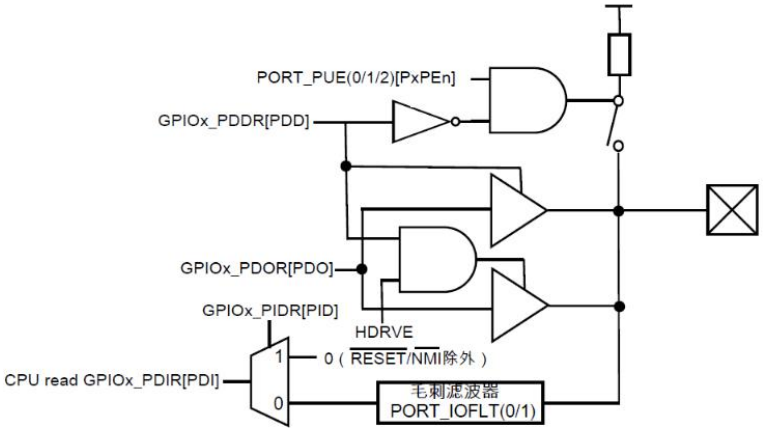


图 8-4 高强度驱动 I/O 结构

■ 8.3.2 端口数据和数据方向

并行 I/O 的读取和写入通过端口数据寄存器(GPIOx_PDIR/PDOR)实现。方向（输入或输出）通过输入禁用寄存器(GPIOx_PIDR)和数据方向寄存器(GPIOx_PDDR)进行控制。

复位后，所有并行 I/O 均默认处于高阻态。必须配置端口数据方向寄存器(GPIOx_PDDR)或输入禁用寄存器(GPIOx_PIDR)中的相应位以便进行输出或输入操作。每个端口引脚都具有一个输入禁用位和一个输出使能位。当 GPIOx_PIDR[PID]=0 时，从读取 GPIOx_PDIR 将返回对应引脚的输入值；当 GPIOx_PIDR[PID]=1 时，读取 GPIOx_PDIR[PDI]将返回 0，但$\overline{\text{RESET}}$/$\overline{\text{NMI}}$除外。

注：当对应的引脚用作输入功能时必须将 GPIOx_PDDR 清零，以防冲突。如果同时设置 GPIOx_PDDR 和 GPIOx_PIDR 位，则读取 GPIOx_PDIR 将始终读取引脚状态。

当外设模块或系统功能控制端口引脚时，即使外设系统对实际引脚方向具有覆盖控制能力，数据方向寄存器位也仍然控制针对端口数据寄存器的读操作所返回的内容。

■ 8.3.3 内部上拉使能

通过设置其中某个上拉使能寄存器(PORT_PUE(L/H))中的相应位，便可针对每个端口引脚使能内部上拉器件。如果此引脚配置为通过并行 I/O 控制逻辑或任何共享外设功能输出，那么无论对应的上拉使能寄存器位的状态如何，都将禁用此内部上拉器件。若引脚由模拟功能控制，内部上拉器件同样会被禁用。

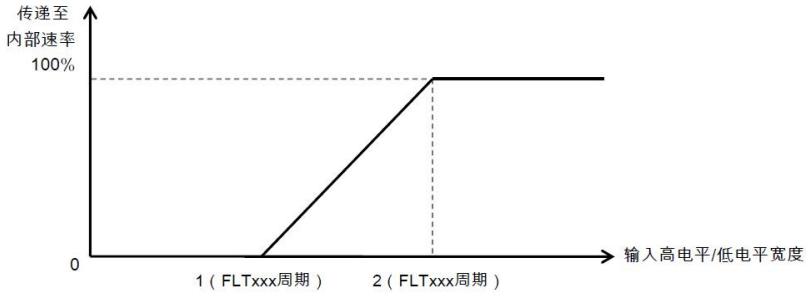
注：如果为使用“SDA(PA2/PB6)和 SCL(PA3/PB7)”引脚配置 I2C，并且采用的是内部上拉器件而非外部上拉器件，那么当这些引脚配置为输出时，内部上拉器件仍保持当前设置，不过在输出值较低时会被自

动禁用以省电。

■ 8.3.4 输入毛刺滤波器设置

滤波器应用于每个按数字输入配置的端口引脚。它可用作简单的低通滤波器，对 GPIO、IRQ、$\overline{\text{RESET}}$、$\overline{\text{NMI}}$ 和 KBI 引脚产生的任何毛刺进行滤波。毛刺宽度阈值可在 1~4096 BUSCLK （或 1~128 LPOCLK）的区间设置 PORT_IOFLT[FCTDIVn]轻松调节。该可配置毛刺滤波器可替代板载外部模拟滤波器，并且能极大地提高 EMC 性能，因为任何毛刺都不会被错误地采样或忽略。设置寄存器 PORT_IOFLT 可配置整个端口或外设输入的滤波器。例如，设置 PORT_IOFLT0[FLTA]将会影响所有 PAn 引脚。

毛刺宽度比所选时钟周期短的毛刺会被滤除；但毛刺宽度比所选时钟周期小两倍以上的毛刺将不会被滤除，并且会进入内部电路。



注释：FLTxxx 是寄存器 PORT_IOFLT 中的内容。

图 8-5 输入毛刺滤波器

■ 8.3.5 高电流驱动

输出高灌电流/拉电流驱动可通过设置 HDRVE 寄存器中的相应位使能。这些引脚可用作输出和输入；引脚作为输出时，输出的是高灌电流/拉电流。

- 若引脚通过并行 I/O 控制逻辑配置为输入，则高电流驱动功能禁用。
- 配置为任何共用外设功能时，高电流驱动功能依然可以作用于这些引脚，但仅当这些引脚配置为输出时才有效。

■ 8.3.6 停止模式下的引脚特性

在停止模式下，保持所有 I/O 状态，因为内部逻辑电路保持上电状态。只要一恢复，正常的 I/O 功能就可供用户使用。

■ 8.3.7 端口数据寄存器映射和定义

表 8-3 PORT 存储器映射

绝对地址	寄存器名称	偏移量	位宽	访问	复位值
0x4004_9000	端口滤波寄存器(PORT_IOFLT)	00h	32	R/W	00C0_0000h
0x4004_9004	端口上拉使能滤波寄存器L(PORT_PUEL)	04h	32	R/W	0010_0000h
0x4004_9008	端口上拉使能滤波寄存器H(PORT_PUEH)	08h	32	R/W	0000_0000h
0x4004_900C	端口高强度驱动寄存器(PORT_HDRVE)	0Ch	32	R/W	0000_0000h

● 8.3.7.1 端口滤波寄存器(PORT_IOFLT)

该寄存器设置输入引脚的滤波器。配置高电平/低电平毛刺宽度阈值。持续时间比所选时钟周期更短的毛刺将被滤除；持续时间比所选时钟周期长两倍以上的毛刺将不会被滤除，并且会进入内部电路。

地址: 4004_9000h 基准 + 0h 偏移 = 4004_9000h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	FLTDIV3			FLTDIV2			FLTDIV1		FLTNMI		FLTKBI1		FLTKBI0		FLTRST	
复位	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写	FLTH		FLTG		FLTF		FLTE		FLTD		FITC		FLTB		FLTA	
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-4 PORT_IOFLT 字段描述

位	名称	描述
31-29	FLTDIV3	滤波器分组 3 端口滤波器分组 3 000 LPOCLK 001 LPOCLK/2 010 LPOCLK/4 011 LPOCLK/8 100 LPOCLK/16 101 LPOCLK/32 110 LPOCLK/64 111 LPOCLK/128
28-26	FLTDIV2	滤波器分组 2 端口滤波器分组 2 000 BUSCLK/32 001 BUSCLK/64 010 BUSCLK/128 011 BUSCLK/256 100 BUSCLK/512 101 BUSCLK/1024 110 BUSCLK/2048 111 BUSCLK/4096

25-24	FLTDIV1	滤波器分组 1 端口滤波器分组 1 000 BUSCLK/2 001 BUSCLK/4 010 BUSCLK/8 011 BUSCLK/16
23-22	FLTNMI	针对$\overline{\text{NMI}}$输入的滤波器选择 00 无滤波 01 选择 FLTDIV1,在停止模式下会自动切换到 FLTDIV3。 10 选择 FLTDIV2,在停止模式下会自动切换到 FLTDIV3。 11 FLTDIV3
21-20	FLTKB1	针对 KBI1 输入的滤波选择 00 无滤波 01 选择 FLTDIV1,在停止模式下会自动切换到 FLTDIV3。 10 选择 FLTDIV2,在停止模式下会自动切换到 FLTDIV3。 11 FLTDIV3
19-18	FLTKB0	针对 KBI0 输入的滤波选择 00 无滤波 01 选择 FLTDIV1,在停止模式下会自动切换到 FLTDIV3。 10 选择 FLTDIV2,在停止模式下会自动切换到 FLTDIV3。 11 FLTDIV3
17-16	FLTRST	针对$\overline{\text{RESET}}$/IRQ 输入的滤波选择 00 无滤波 01 选择 FLTDIV1,在停止模式下会自动切换到 FLTDIV3。 10 选择 FLTDIV2,在停止模式下会自动切换到 FLTDIV3。 11 FLTDIV3
15-14	FLTH	针对 PH 输入的滤波选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
13-12	FLTG	针对 PG 输入的滤波选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
11-10	FLTF	针对 PF 输入的滤波选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3

9-8	FLTE	针对 PE 输入的滤波选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
7-6	FLTD	针对 PD 输入的滤波选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
5-4	FLTC	针对 PC 输入的滤波选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
3-2	FLTB	针对 PB 输入的滤波选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3
1-0	FLTA	针对 PA 输入的滤波选择 00 BUSCLK 01 FLTDIV1 10 FLTDIV2 11 FLTDIV3

● 8.3.7.2 端口上拉使能寄存器 L(PORT_PUEL)

地址: 4004_9000h 基准 + 4h 偏移 = 4004_9004h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读 写	PDPE7	PDPE6	PDPE5	PDPE4	PDPE3	PDPE2	PDPE1	PDPE0	PCPE7	PCPE6	PCPE5	PCPE4	PCPE3	PCPE2	PCPE1	PCPE0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读 写	PBPE7	PBPE6	PBPE5	PBPE4	PBPE3	PBPE2	PBPE1	PBPE0	PAPE7	PAPE6	PAPE5	PAPE4	PAPE3	PAPE2	PAPE1	PAPE0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-5 PORT_PUEL 字段描述

位	名称	描述

31-24	PDPEn (n=7:0)	端口 D 位 n 上拉使能 该控制字段确定内部上拉器件是否针对相关的 PD 引脚使能。对于配置为输出或高阻态的端口 D 引脚，该字段无效。 0 端口 D 位 n 上拉禁用。 1 端口 D 位 n 上拉使能。
23-16	PCPEn (n=7:0)	端口 C 位 n 上拉使能 该控制字段确定内部上拉器件是否针对相关的 PC 引脚使能。对于配置为输出或高阻态的端口 C 引脚，该字段无效。 0 端口 C 位 n 上拉禁用。 1 端口 C 位 n 上拉使能。
15-8	PBPEn (n=7:0)	端口 B 位 n 上拉使能 该控制字段确定内部上拉器件是否针对相关的 PB 引脚使能。对于配置为输出或高阻态的端口 B 引脚，该字段无效。 0 端口 B 位 n 上拉禁用。 1 端口 B 位 n 上拉使能。
7-0	PAPEn (n=7:0)	端口 A 位 n 上拉使能 该控制字段确定内部上拉器件是否针对相关的 PA 引脚使能。对于配置为输出或高阻态的端口 A 引脚，该字段无效。 0 端口 A 位 n 上拉禁用。 1 端口 A 位 n 上拉使能。

● 8.3.7.3 端口上拉使能寄存器 H(PORT_PUEH)

地址: 4004_9000h 基准 + 8h 偏移 = 4004_9008h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读写	PHPE7	PHPE6	0			PHPE2	PHPE1	PHPE0	0				PGPE3	PGPE2	PGPE1	PGPE0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	PFPE7	PFPE6	PFPE5	PFPE4	PFPE3	PFPE2	PFPE1	PFPE0	PEPE7	PEPE6	PEPE5	PEPE4	PEPE3	PEPE2	PEPE1	PEPE0
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-6 PORT_PUEH 字段描述

位	名称	描述
31-30 26-24	PHPEn (n=7, 6, 2, 1, 0)	端口 H 位 n 上拉使能 该控制字段确定内部上拉器件是否针对相关的 PH 引脚使能。对于配置为输出或高阻态的端口 H 引脚，该字段无效。 0 端口 H 位 n 上拉禁用。 1 端口 H 位 n 上拉使能。

29-27	保留	保留字段。 只读且始终读取为 0.
23-20	保留	保留字段。 只读且始终读取为 0.
19-16	PGPEn (n=3:0)	端口 G 位 n 上拉使能 该控制字段确定内部上拉器件是否针对相关的 PG 引脚使能。对于配置为输出或高阻态的端口 G 引脚，该字段无效。 0 端口 G 位 n 上拉禁用。 1 端口 G 位 n 上拉使能。
15-8	PFPEn (n=7:0)	端口 F 位 n 上拉使能 该控制字段确定内部上拉器件是否针对相关的 PF 引脚使能。对于配置为输出或高阻态的端口 F 引脚，该字段无效。 0 端口 F 位 n 上拉禁用。 1 端口 F 位 n 上拉使能。
7-0	PEPEn (n=7:0)	端口 E 位 n 上拉使能 该控制字段确定内部上拉器件是否针对相关的 PE 引脚使能。对于配置为输出或高阻态的端口 E 引脚，该字段无效。 0 端口 E 位 n 上拉禁用。 1 端口 E 位 n 上拉使能。

● 8.3.7.4 端口高强度驱动使能寄存器(PORT_HDRVE)

地址: 4004_9000h 基准 + Ch 偏移 = 4004_900Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
读	0															
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

位	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0								PH1	PH0	PE1	PE0	PD1	PD0	PB5	PB4
写																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-7 PORT_HDRVE 字段描述

位	名称	描述
31-8	保留	保留字段。 只读且始终读取为 0.
7-6	PHn (n=1:0)	PHn 端口高电流驱动能力 该读/写字段使能 PHn 的高电流驱动能力。 0 PHn 端口禁用高电流驱动。 1 端口使能高电流驱动。

5-4	PEn (n=1:0)	PEn 端口高电流驱动能力 该读/写字段使能 PEn 的高电流驱动能力。 0 PEn 端口禁用高电流驱动。 1 PEn 端口使能高电流驱动。
3-2	PDn (n=1:0)	PDn 端口高电流驱动能力 该读/写字段使能 PDn 的高电流驱动能力。 0 PDn 端口禁用高电流驱动。 1 PDn 端口使能高电流驱动。
1-0	PBn (n=5:4)	PBn 端口高电流驱动能力 该读/写字段使能 PBn 的高电流驱动能力。 0 PBn 端口禁用高电流驱动。 1 PBn 端口使能高电流驱动。

■ 8.4 通用输入/输出（GPIO）模块

■ 8.4.1 简介

通用输入和输出(GPIO)模块可通过外设总线访问，还能通过零等待状态接口(IOPORT)与处理器内核通信，实现最高的引脚性能。GPIO 寄存器支持 8 位、16 位或 32 位访问。当引脚配置为用于 GPIO 功能时，GPIO 数据方向和输出数据寄存器控制每个引脚的方向和输出数据。假设引脚对应的端口控制和中断模块已使能，则当引脚配置为用于任意数字功能时，GPIO 输入数据寄存器显示每个引脚上的逻辑值。通过为每个端口输出数据寄存器增加针对只写寄存器的置位、清零和切换操作，便可有效支持通用输出的位操作。

■ 8.4.2 特性

GPIO 模块的特性包括：

- 端口数据输入寄存器适用于所有数字引脚多路复用模式
- 端口数据输入寄存器带对应的置位/清零/切换寄存器
- 端口数据方向寄存器
- 通过 IOPORT 实现对 GPIO 寄存器的零等待状态访问

注：GPIO 模块由系统对 GPIO 寄存器的零等待状态访问

■ 8.4.3 操作模式

下表介绍了 GPIO 模块的不同操作模式及其在这些模式下的行为:

表 8-8 操作模式

操作模式	说明
运行	GPIO 模式正常运行。
等待	GPIO 模式正常运行。
停止	GPIO 模式已禁用。
调试	GPIO 模式正常运行。

■ 8.4.4 GPIO 信号说明

表 8-9 GPIO 信号说明

GPIO 信号说明	说明	I/O
PA7-PA0	通用输入/输出	I/O
PB7-PB0	通用输入/输出	I/O
PC7-PC0	通用输入/输出	I/O
PD7-PD0	通用输入/输出	I/O
PE7-PE0	通用输入/输出	I/O
PF7-PF0	通用输入/输出	I/O
PG7-PG0	通用输入/输出	I/O
PH7-PH0	通用输入/输出	I/O

注：并非每个端口中的所有引脚都实施在每个器件上。有关器件的可用 GPIO 端口数量，请参见信号多路复用的相关章节。

● 8.4.4.1 GPIO 信号详细说明

表 8-10 GPIO 信号详细说明

信号	I/O	说明	
PA7-PA0	I/O	通用输入/输出	
PB7-PB0		状态含义	电平有效：该引脚为 1。 电平无效：该引脚为 0。
PC7-PC0			
PD7-PD0		时序	有效：输出时，该信号发生在系统时钟的上升沿。就输入而言，它可能在任何时刻发生，且输入变为有效的时间与系统时钟可能相异。
PE7-PE0			
PF7-PF0			
PG7-PG0			无效：输出时，该信号发生在系统时钟的上升沿。就输入而言，它可能在任何时刻发生，且输入变为有效的时间与系统时钟可能相异。
PH7-PH0			

■ 8.4.5 GPIO 存储器映射和寄存器说明

所有GPIO寄存器均可通过基地址0x400F_F000或0x4000_F000进行访问。建议将0x400F_F000用作GPIO模块的基地址，且本章所述的寄存器存储器映射同样基于基地址0x400F_F000。

在有效存储器映射外对 GPIO 存储器空间进行任何读写访问都将导致总线错误。

● 8.4.5.1 GPIO/FGPIO 寄存器位分配

在该器件中，所有 8 位端口引脚都映射到 32 位 GPIO/FGPIO 寄存器，如表中所示：

表 8-11 GPIOA/FGPIOA 寄存器位分配

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
引脚	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0

表 8-12 GPIOB/FGPIOB 寄存器位分配

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
引脚	PH7	PH6	保留	保留	保留	PH2	PH1	PH0	保留	保留	保留	保留	PG3	PG2	PG1	PG0	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0

表 8-13 GPIO 存储器映射

绝对地址（16 进制）	寄存器名称	偏移量	位宽	访问	复位值
0x400F_F000	端口数据输出寄存器(GPIOA_PDOR)	00h	32	R/W	0000_0000h
0x400F_F004	端口置位输出寄存器(GPIOA_PSOR)	04h	32	W(读为 0)	0000_0000h
0x400F_F008	端口清零输出寄存器(GPIOA_PCOR)	08h	32	W(读为 0)	0000_0000h
0x400F_F00C	端口切换输出寄存器(GPIOA_PTOR)	0Ch	32	W(读为 0)	0000_0000h
0x400F_F010	端口数据输入寄存器(GPIOA_PDIR)	10h	32	R	0000_0000h
0x400F_F014	端口数据方向寄存器(GPIOA_PDDR)	14h	32	R/W	0000_0000h
0x400F_F018	端口输入禁用寄存器(GPIOA_PIDR)	18h	32	R/W	ffff_ffffh
0x400F_F040	端口数据输出寄存器(GPIOB_PDOR)	40h	32	R/W	0000_0000h
0x400F_F044	端口置位输出寄存器(GPIOB_PSOR)	44h	32	W(读为 0)	0000_0000h
0x400F_F048	端口清零输出寄存器(GPIOB_PCOR)	48h	32	W(读为 0)	0000_0000h
0x400F_F04C	端口切换输出寄存器(GPIOB_PTOR)	4Ch	32	W(读为 0)	0000_0000h
0x400F_F050	端口数据输入寄存器(GPIOB_PDIR)	50h	32	R	0000_0000h
0x400F_F054	端口数据方向寄存器(GPIOB_PDDR)	54h	32	R/W	0000_0000h
0x400F_F058	端口输入禁用寄存器(GPIOB_PIDR)	58h	32	R/W	ffff_ffffh

● 8.4.5.2 端口数据输出寄存器（GPIOx_PDOR）

该寄存器配置在各通用输出引脚上驱动的逻辑电平。

注：请勿修改与选定封装中未提供的引脚有关的引脚配置寄存器。封装中未提供的所有未连接引脚将默认处于禁用状态，以实现最低功耗。

地址：基址 + 偏移：GPIOA_PDOR = 0x400F_F000h / GPIOB_PDOR = 0x400F_F040h

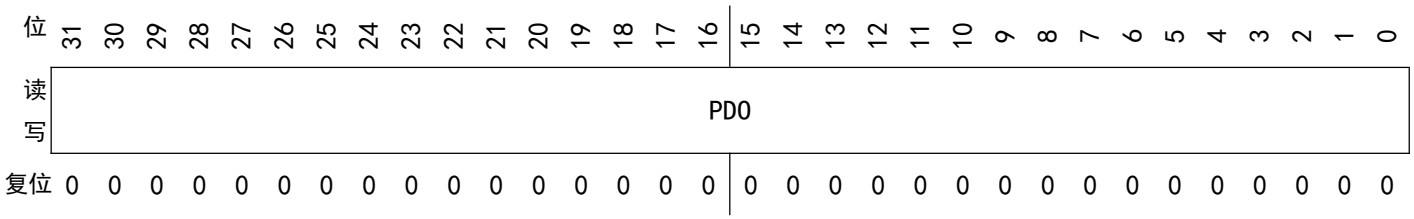


表 8-14 GPIOx_PDOR 字段描述

字段	描述
PDO	端口数据输出

	读取未连接引脚的寄存器位将返回未定义值 0 假设引脚配置为通过输出，则在引脚上驱动逻辑电平 0。 1 假设引脚配置为通过输出，则在引脚上驱动逻辑电平 1。

● 8.4.5.3 端口置位输出寄存器（GPIOx_PSOR）

该寄存器配置是否置位 PDOR 的字段。

地址：基址 + 偏移：GPIOA_PSOR = 0x400F_F004h / GPIOB_PSOR = 0x400F_F044h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0																															
写	PTSO																															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-15 GPIOx_PSOR 字段描述

字段	描述
PTSO	数据置位输出 如下所示对该寄存器进行写操作将更新 PDOR 中对应位的值： 0 PDORn 中的对应位不发生变化。 1 PDORn 中的对应位置位为逻辑电平 1。

● 8.4.5.4 端口清零输出寄存器（GPIOx_PCOR）

该寄存器配置是否清零 PDOR 的字段。

地址：基址 + 偏移：GPIOA_PCOR = 0x400F_F008h / GPIOB_PCOR = 0x400F_F048h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0																															
写	PTCO																															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-16 GPIOx_PCOR 字段描述

字段	描述
PTCO	端口清零输出 如下所示对该寄存器进行写操作将更新 PDOR 中对应位的值： 0 PDORn 中的对应位不发生变化。 1 PDORn 中的对应位清零为逻辑电平 0。

● 8.4.5.5 端口切换输出寄存器（GPIOx_PTOR）

地址：基址 + 偏移：GPIOA_PTOR = 0x400F_F00Ch / GPIOB_PTOR = 0x400F_F04Ch

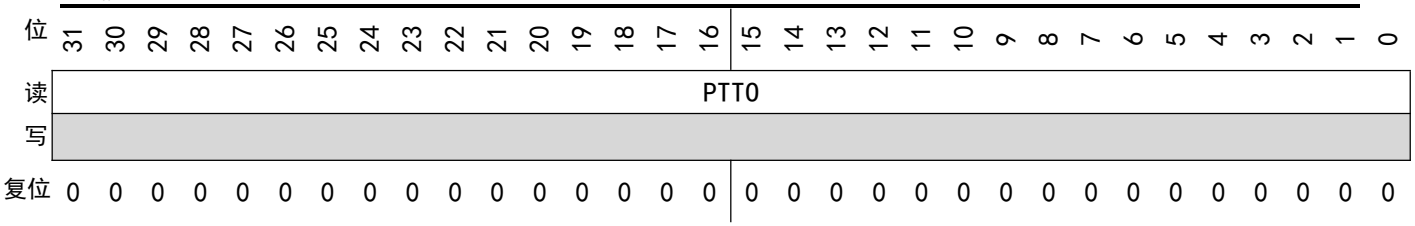


表 8-17 GPIOx_PTOR 字段描述

字段	描述
PTTO	端口切换输出 如下所示对该寄存器进行写操作将更新 PDOR 中对应位的值： 0 PDORn 中的对应位不发生变化。 1 PDORn 中的对应位置位为现有逻辑状态的反相电平。

● 8.4.5.6 端口数据输入寄存器（GPIOx_PDIR）

地址：基址 + 偏移：GPIOA_PDIR = 0x400F_F010h / GPIOB_PDIR = 0x400F_F050h

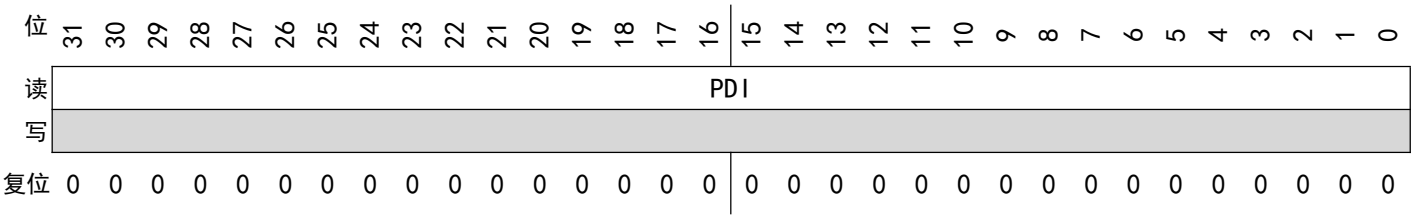


表 8-18 GPIOx_PDIR 字段描述

字段	描述
PDI	端口数据输入 在特定器件的未实施引脚读取 0。未配置为用于数字功能的引脚读取 0。如果禁用端口控制和中断模块，那么 PDIR 中的对应位不更新。 0 引脚逻辑电平为逻辑 0，或者未配置为供数字功能使用。 1 引脚逻辑电平为逻辑 1。

● 8.4.5.7 端口数据方向寄存器（GPIOx_PDDR）

PDDR 将各个端口引脚配置为输入或输出。

地址：基址 + 偏移：GPIOA_PDDR = 0x400F_F014h / GPIOB_PDDR = 0x400F_F054h

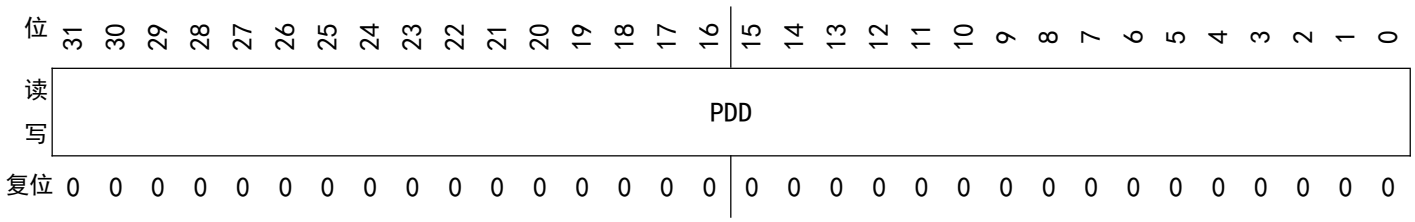


表 8-19 GPIOx_PDDR 字段描述

字段	描述
PDD	端口数据方向 将各个端口引脚配置为输入或输出: 0 引脚配置为通用输入，用于 GPIO 功能。若在 GPIOx_PIDR 寄存器中禁用端口输入，则引脚将为高阻态。 1 引脚配置为通用输出，用于 GPIO 功能。

● 8.4.5.8 端口输入禁用寄存器（GPIOx_PIDR）

地址：基址 + 偏移：GPIOA_PIDR = 0x400F_F018h / GPIOB_PIDR = 0x400F_F058h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	PID																															
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

表 8-20 GPIOx_PIDR 字段描述

字段	描述
PID	端口输入禁用 0 假设引脚配置为用于任意数字功能，则该引脚配置为通用输入。 1 引脚未配置为通用输入。对应的端口数据输入寄存器位将读取零。

■ 8.4.6 FGPIO 存储器映射和寄存器说明

GPIO 寄存器也可另外命名，指向 Cortex-M0+上地址 0xF800_0000 处的 IOPORT 接口。

通过 IOPORT 接口执行的访问与任何指令的获取保持同步，因此可在一个周期内完成。该另外命名的快速 GPIO 存储器映射称为 FGPIO。

对超出有效存储器映射范围的 FGPIO 存储器空间进行任何读写访问都会导致总线错误。所有寄存器访问完成时均具有零等待状态，完成时具有一个等待状态的错误访问除外。

● 8.4.6.1 GPIO/FGPIO 寄存器位分配

在该器件中，所有 8 位端口引脚都映射到 32 位 GPIO/FGPIO 寄存器，如表中所示：

表 8-21 GPIOA/FGPIOA 寄存器位分配

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
引脚	PD7	PD6	PD5	PD4	PD3	PD2	PD1	PD0	PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0	PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0	PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0

表 8-22 GPIOB/FGPIOB 寄存器位分配

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
引脚	PH7	PH6	保留	保留	保留	PH2	PH1	PH0	保留	保留	保留	保留	PG3	PG2	PG1	PG0	PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0	PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0

表 8-23 FGPIO 存储器映射

绝对地址（16 进制）	寄存器名称	偏移量	位宽	访问	复位值
0xF800_0000	端口数据输出寄存器(FGPIOA_PDOR)	00h	32	R/W	0000_0000h
0xF800_0004	端口置位输出寄存器(FGPIOA_PSOR)	04h	32	W(读为 0)	0000_0000h
0xF800_0008	端口清零输出寄存器(FGPIOA_PCOR)	08h	32	W(读为 0)	0000_0000h
0xF800_000C	端口切换输出寄存器(FGPIOA_PTOR)	0Ch	32	W(读为 0)	0000_0000h
0xF800_0010	端口数据输入寄存器(FGPIOA_PDIR)	10h	32	R	0000_0000h
0xF800_0014	端口数据方向寄存器(FGPIOA_PDDR)	14h	32	R/W	0000_0000h
0xF800_0018	端口输入禁用寄存器(FGPIOA_PIDR)	18h	32	R/W	ffff_ffffh
0xF800_0040	端口数据输出寄存器(FGPIOB_PDOR)	40h	32	R/W	0000_0000h
0xF800_0044	端口置位输出寄存器(FGPIOB_PSOR)	44h	32	W(读为 0)	0000_0000h
0xF800_0048	端口清零输出寄存器(FGPIOB_PCOR)	48h	32	W(读为 0)	0000_0000h
0xF800_004C	端口切换输出寄存器(FGPIOB_PTOR)	4Ch	32	W(读为 0)	0000_0000h
0xF800_0050	端口数据输入寄存器(FGPIOB_PDIR)	50h	32	R	0000_0000h
0xF800_0054	端口数据方向寄存器(FGPIOB_PDDR)	54h	32	R/W	0000_0000h
0xF800_0058	端口输入禁用寄存器(FGPIOB_PIDR)	58h	32	R/W	ffff_ffffh

● 8.4.6.2 端口数据输出寄存器（FGPIOx_PDOR）

该寄存器配置在各通用输出引脚上驱动的逻辑电平。

地址：基址 + 偏移：FGPIOA_PDOR = 0xF800_0000h / FGPIOB_PDOR = 0xF800_0040h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	PDO																															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-24 FGPIOx_PDOR 字段描述

字段	描述
PDO	端口数据输出 读取未连接引脚的寄存器位将返回未定义值 0 假设引脚配置为通过输出，则在引脚上驱动逻辑电平 0。 1 假设引脚配置为通过输出，则在引脚上驱动逻辑电平 1。

● 8.4.6.3 端口置位输出寄存器（FGPIOx_PSOR）

该寄存器配置是否置位 PDOR 的字段。

地址：基址 + 偏移：FGPIOA_PSOR = 0xF800_0004h / FGPIOB_PSOR = 0xF800_0044h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0																															
写	PTSO																															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-25 FGPIOx_PSOR 字段描述

字段	描述
PTSO	数据置位输出 如下所示对该寄存器进行写操作将更新 PDOR 中对应位的值： 0 PDORn 中的对应位不发生变化。 1 PDORn 中的对应位置位为逻辑电平 1。

● 8.4.6.4 端口清零输出寄存器（FGPIOx_PCOR）

该寄存器配置是否清零 PDOR 的字段。

地址：基址 + 偏移：FGPIOA_PCOR = 0xF800_0008h / FGPIOB_PCOR = 0xF800_0048h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0																															
写	PTCO																															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-26 FGPIOx_PCOR 字段描述

字段	描述
PTCO	端口清零输出 如下所示对该寄存器进行写操作将更新 PDOR 中对应位的值： 0 PDORn 中的对应位不发生变化。 1 PDORn 中的对应位清零为逻辑电平 0。

● 8.4.6.5 端口切换输出寄存器（FGPIOx_PTOR）

地址：基址 + 偏移：FGPIOA_PTOR = 0xF800_000Ch / FGPIOB_PTOR = 0xF800_004Ch

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	0																															
写	PTTO																															
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-27 FGPIOx_PTOR 字段描述

字段	描述
PTTO	端口切换输出 如下所示对该寄存器进行写操作将更新 PDOR 中对应位的值： 0 PDORn 中的对应位不发生变化。 1 PDORn 中的对应位置位为现有逻辑状态的反相电平。

● 8.4.6.6 端口数据输入寄存器（FGPIOx_PDIR）

注：请勿修改与选定封装中未提供的引脚有关的引脚配置寄存器。封装中未提供的所有未连接引脚将默认处于禁用状态，以实现最低功耗。

地址：基址 + 偏移：FGPIOA_PDIR = 0xF800_0010h / FGPIOB_PDIR = 0xF800_0050h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	PDI																															
写																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-28 FGPIOx_PDIR 字段描述

字段	描述
PDI	端口数据输入 在特定器件的未实施引脚读取 0。未配置为用于数字功能的引脚读取 0。如果禁用端口控制和中断模块，那么 PDIR 中的对应位不更新。 0 引脚逻辑电平为逻辑 0，或者未配置为供数字功能使用。 1 引脚逻辑电平为逻辑 1。

● 8.4.6.7 端口数据方向寄存器（FGPIOx_PDDR）

PDDR 将各个端口引脚配置为输入或输出。

地址：基址 + 偏移：FGPIOA_PDDR = 0xF800_0014h/FGPIOB_PDDR = 0xF800_0054h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读	PDD																															
写																																
复位	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

表 8-29 FGPIOx_PDDR 字段描述

字段	描述
PDD	端口数据方向

	将各个端口引脚配置为输入或输出: 0 引脚配置为通用输入，用于 GPIO 功能。若在 FGPIOx_PIDR 寄存器中禁用端口输入，则引脚将为高阻态。 1 引脚配置为通用输出，用于 GPIO 功能。

● 8.4.6.8 端口输入禁用寄存器（FGPIOx_PIDR）

地址：基址 + 偏移：FGPIOA_PIDR = 0xF800_0018h/FGPIOB_PIDR = 0xF800_0058h

位	31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
读写	PID																															
复位	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

表 8-30 FGPIOx_PIDR 字段描述

字段	描述
PID	端口输入禁用 0 假设引脚配置为用于任意数字功能，则该引脚配置为通用输入。 1 引脚未配置为通用输入。对应的端口数据输入寄存器位将读取零。

■ 8.4.7 功能说明

● 8.4.7.1 通用输入

每个引脚的逻辑状态可通过端口数据输入寄存器来指明，假设已将该引脚配置用于执行数字功能且对应的端口控制和中断模块已使能。

● 8.4.7.2 通用输出

可通过端口数据输出寄存器和端口数据方向寄存器来控制每个引脚的逻辑状态，假设引脚已配置为执行 GPIO 功能。下表介绍了要配置为输入/输出的引脚的各种情形。

表 8-31 输入/输出引脚的各种情形

条件	结果
引脚已配置为执行 GPIO 功能，并且已清除对应的端口数据方向寄存器位。	引脚已配置为输入。
引脚已配置为执行 GPIO 功能，并且已设置对应的端口数据方向寄存器位。	引脚已配置为输出，并且引脚的逻辑状态与对应的端口数据输出寄存器相同。

为了对通用输出进行高效的位处理，须提供引脚数据置位、引脚数据清零和引脚数据切换寄存器，以便仅通过一次寄存器写访问置位、清零或切换一个端口内的一个或多个输出。

无需启用对应的端口控制和中断模块来更新端口数据方向寄存器和端口数据输出寄存器（包括置位/清零/切换寄存器）的状态。

● 8.4.7.3 IOPORT

GPIO 寄存器也可另外命名，指向 Cortex-M0+上地址 0xF800_0000 处的 IOPORT 接口。通过 IOPORT 接口执行的访问与任何指令的获取保持同步，因此可在一个周期内完成。

■ 8.5 键盘中断（KBI）

■ 8.5.1 特性

键盘中断 KBI 模块特性：

- 最多 32 个具有单个使能位的键盘中断引脚
- 每个键盘中断引脚可编程为：
  - ❖ 仅下降沿触发
  - ❖ 仅上升沿触发
  - ❖ 上升沿和高电平都触发
  - ❖ 下降沿和低电平都触发

■ 8.5.2 操作模式

本节定义以下模式中的 KBI 操作:

- 等待模式
- 停止模式
- 后台调试模式

● 8.5.2.1 等待模式下的 KBI

执行等待指令会将 MCU 置于等待模式。如果需要，则在执行等待指令之前应先使能 KBI 中断(KBI_SC[KIBE]=1)。使 KBI 在 MCU 处于等待模式时能够继续工作。如果已使能 KBI 中断(KBI_SC[KIBE]=1)，则已使能的 KBI 引脚 KBI_PE[KBIPEn]=1)可使 MCU 退出等待模式。

● 8.5.2.2 停止模式下的 KBI

执行等待指令会将 MCU 置于停止模式（停止模式使能有效时），在该模式下 KBI 可异步工作。执行停止指令前必须先使能 KBI 中断(KBI_SC[KIBE]=1)。使 KBI 在 MCU 处于停止模式时能够继续工作。如果已使能 KBI 中断（KBI_SC[KIBE]=1)，则已使能的 KBI 引脚 KBI_PE[KBIPEn]=1)可使 MCU 退出停止模式。

● 8.5.2.3 结构框图

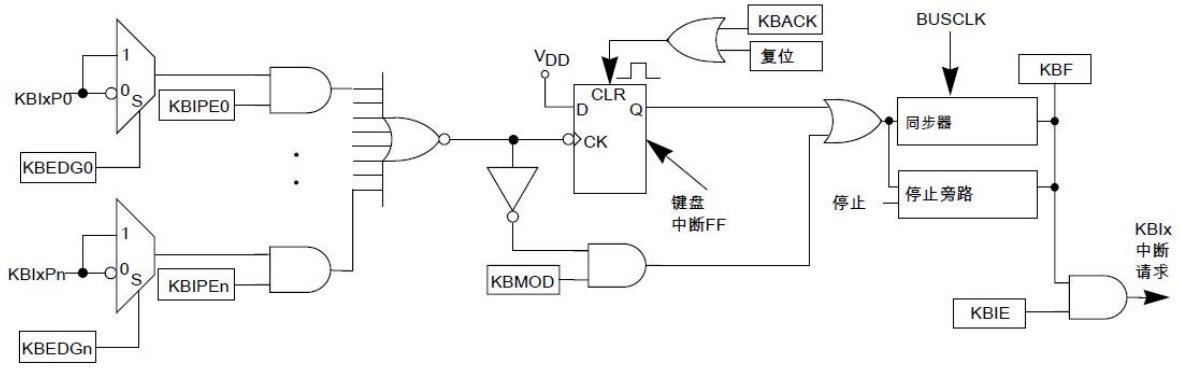


图 8-6 KBI 结构框图

■ 8.5.3 外部信号说明

KBI 输入引脚可用于检测下降沿或同时检测下降沿和低电平中断请求。 KBI 输入引脚还可用于检测上升沿或同时检测上升沿和高电平中断请求。

KBI 的型号属性如下表所示：

表 8-32 外部信号说明

信号	功能	I/O
KBIxPn	键盘中断引脚	I

■ 8.5.4 存储器映射和寄存器说明

表 8-33 KBI 存储器映射

绝对地址（16 进制）	寄存器名称	偏移量	位宽	权限	复位值
4007_9000	KBI0 状态和控制寄存器（KBI0_SC）	00h	8	R/W	00h
4007_9001	KBI0 引脚使能寄存器（KBI0_PE）	01h	8	R/W	00h
4007_9002	KBI0 边沿选择寄存器（KBI0_ES）	02h	8	R/W	00h
4007_A000	KBI1 状态和控制寄存器（KBI1_SC）	00h	8	R/W	00h
4007_A001	KBI1 引脚使能寄存器（KBI1_PE）	01h	8	R/W	00h
4007_A002	KBI1 边沿选择寄存器（KBI1_ES）	02h	8	R/W	00h

● 8.5.4.1 KBI 状态和控制寄存器(KBIx_SC)

KBIx_SC 包含用于配置 KBI 的状态标志和控制位。

地址：基址 + 00h 偏移 = KBI0_SC：0x4007_9000h，KBI1_SC：0x4007_A000h

位	7	6	5	4	3	2	1	0
读	0				KBF		KBIE	KBMOD
写						KBACK		
复位	0	0	0	0	0	0	0	0

表 8-34 KBIx_SC 字段描述

字段	描述
7-4 保留	此字段为保留字段 此只读字段为保留字段且值始终为 0。
3 KBF	KBI 中断标志 表示检测到 KBI 中断请求。写操作对 KBF 无效。 0 未检测到 KBI 中断请求。 1 检测到 KBI 中断请求。
2 KBACK	KBI 应答 在 KBACK 中写入 1 是标志清零机制的一部分。
1 KBIE	KBI 中断使能 确定 KBI 中断是否已使能： 0 KBI 中断未使能。 1 KBI 中断已使能。
0 KBMOD	KBI 检测模式 KBMOD(与 ES[KBEDG]寄存器）一起控制 KBI 中断引脚的检测模式： 0 键盘仅检测边沿。 1 键盘检测边沿和电平。

● 8.5.4.2 引脚使能寄存器（KBIx_PE）

KBI_PE 包含引脚使能控制位。

地址：基址 + 01h 偏移 = KBI0_PE：0x4007_9001h，KBI1_PE：0x4007_A001h

位	7	6	5	4	3	2	1	0
读 写	KBIPE							
复位	0	0	0	0	0	0	0	0

表 8-35 KBIx_PE 字段描述

字段	描述
KBIPE	KBI 引脚使能 每个 KBIPEn 位使能对应 KBI 中断引脚 0 引脚未使能为 KBI 中断。 1 引脚已使能为 KBI 中断。

● 8.5.4.3 KBI 边沿选择寄存器(KBIx_ES)

KBI_ES 包含边沿选择控制位。

地址：基址 + 02h 偏移 = KBI0_ES：0x4007_9002h，KBI1_ES：0x4007_A002h

位	7	6	5	4	3	2	1	0
读 写	KBEDG							
复位	0	0	0	0	0	0	0	0

表 8-36 KBIx_ES 字段描述

字段	描述
KBEDG	KBI 边沿选择 每个 KBEDGn 位选择对应引脚的下降沿/低电平或上升沿/高电平功能 0 下降沿/低电平。 1 上升沿/高电平。

■ 8.5.5 功能说明

该片上外设模块被称为键盘中断模块，因为其最初设计用于简化键盘开关行列式矩阵的连接和使用。但是，这些输入可以作为有用的额外的外部中断的输入，成为一个很好的将 MCU 从停止或等待低功耗模式唤醒的外部方式。

KBI 模块使最多八个引脚能够用作其他中断源。在 KBIx_PE[KBIPEn]位中单独写入将使能或禁用各 KBI 引脚。各 KBI 引脚可在 KBIx_SC[KBMOD]位上配置为基于边沿触发或基于边沿和电平触发。边沿触发可通过软件编程为下降沿出发或上升沿触发；电平可配置为低电平触发或高电平触发。边沿极性或边沿和电平敏感度可使用 KBIx_ES[KBEDGn]位选择。

● 8.5.5.1 边沿触发

同步逻辑用于检测边沿。当已使能的键盘中断(KBIx_PE[KBIPEn]=1)输入信号在一个总线周期中被采集为逻辑 1（无效电平）并且在下一个周期中被采集为逻辑 0（有效电平），则会检测到下降沿。当该输入信号在一个总线周期中被采集为逻辑 0（无效电平）并且在下一个周期中被采集为逻辑 1（有效电平），则会检测到上升沿。

检测到第一个边沿之前，所有已使能的键盘中断输入信号必须处于无效逻辑电平。检测到任何边沿之后，所有已使能的键盘中断输入信号必须返回至无效电平，然后才能检测到任何新边沿。

已使能 KBI 引脚上的有效边沿将置位 KBIx_SC[KBF]。如果 KBIx_SC[KBIE]已置位，则 MCU 中会出现中断请求。在 KBIx_SC[KBACK]中写入 1 将会清零 KBIx_SC[KBF]。

● 8.5.5.2 边沿和电平触发

已使能 KBI 上的有效边沿或电平将置位 KBIx_SC[KBF]。如果 KBIx_SC[KBIE]已置位，则 MCU 中会出现中断请求。如果所有已使能的键盘输入处于其无效电平，则在 KBIx_SC[KBACK]中写入 1 将会清零 KBIx_SC[KBF]。当尝试通过在 KBIx_SC[KBACK]中写入 1 来清零 KBIx_SC[KBF]时，如果任何已使能的 KBI 引脚电平变为有效值，则 KBIx_SC[KBF]将保持置位。

● 8.5.5.3 KBI 上位电阻器

各 KBI 引脚（如已由 KBIx_PE 使能）可通过相关 I/O 端口的拉使能寄存器进行配置，请参见“并行输入

/输出”部分，以使用内部上拉电阻，或不使用电阻器。

如果内部上拉电阻已针对使能的 KBI 引脚使能，则相关 I/O 端口的拉选择寄存器（参见“I/O 端口”一章）可用于选择内部上拉电阻。

● 8.5.5.4 KBI 初始化

首次使能键盘中断时，可能会获得错误的键盘中断标志。要防止键盘初始化过程中的错误中断请求，用户应进行如下操作：

1. 通过清除 KBIx_SC[KBIE]屏蔽键盘中断。
2. 通过设置适当的 KBIx_ES[KBEDGn]位置位而使能 KBI 极性。
3. 使用内部上拉电阻之前，先配置 PORT_PUE0 和 PORT_PUE1 中的关联位。
4. 通过设置适当的 KBIx_PE[KBIPEn]位置位而使能 KBI 引脚。
5. 在 KBIx_SC[KBACK]中进行写操作将清除任何错误中断。
6. 使 KBIx_SC[KBIE]置位以使能中断。

■ 8.6 中断（IRQ）

■ 8.6.1 特性

IRQ 模块特性包括：

- IRQ 中断控制位
- 可编程中断由边沿或边沿和电平触发
- 自动中断应答
- 内部上拉器件

施加在外部中断请求(IRQ)引脚上的低电平可锁存 CPU 中断请求。下图显示的是 IRQ 模块结构：

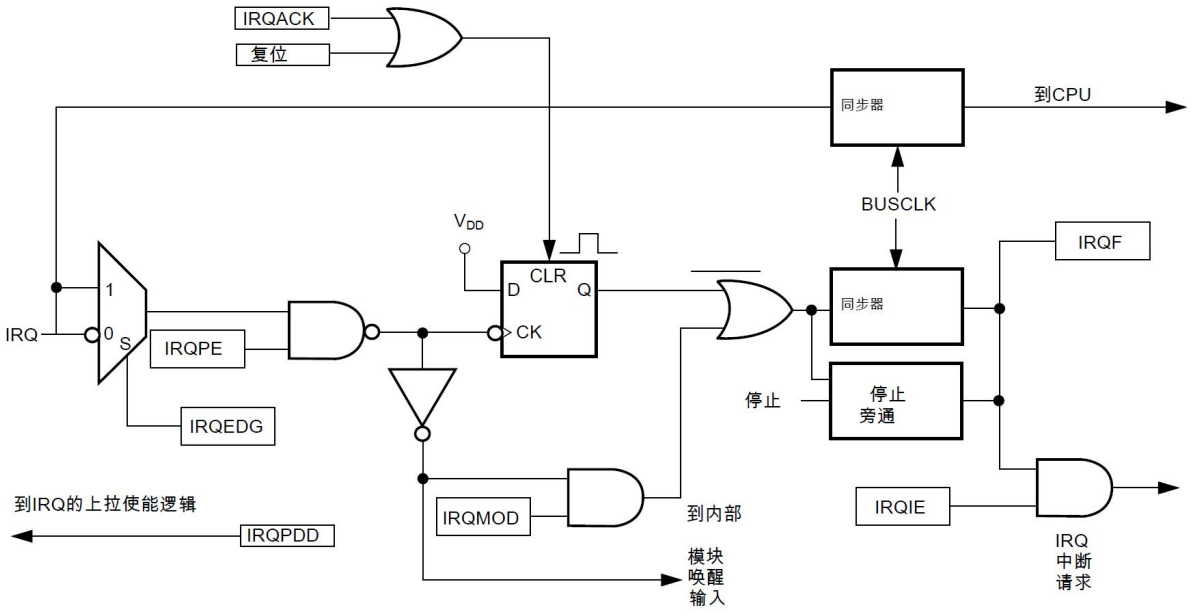


图 8-7 IRQ 模块结构框图

外部中断由 IRQSC 状态和控制寄存器管理。使能 IRQ 功能后，同步逻辑监控引脚，以便确定是仅边沿事件还是边沿和电平事件。MCU 处于停止模式且系统时钟被关断时，就会使用单独的异步路径，从而 IRQ（若使能）能唤醒 MCU。

● 8.6.1.1 引脚配置选项

IRQ 引脚使能（IRQSC[IRQPE]）控制字段必须置 1，IRQ 引脚才能用作 IRQ 输入。用户可以选择检测的边沿或电平的极性（IRQEDG），引脚是只检测边沿还是检测边沿和电平（IRQMOD），或者事件是触发中断还是仅置位 IRQSC[IRQF] 标志，再通过软件轮询。

使能时，IRQ 引脚默认使用内部上拉器件（IRQSC[IRQPDD] = 0）。若用户使用外部上拉或下拉器件，则可将 IRQSC[IRQPDD]置 1 以关断内部器件。

在 IRQ 引脚配置为 IRQ 输入时，BIH 和 BIL 指令可用于检测该引脚上的电平。

注：

(1)该引脚不含钳位至 VDD 的钳位二极管，且不得输入超过 V_{DD}。内部上拉 IRQ 引脚上测得的电压可能低至 V_{DD}－0.7V。连接该引脚的内部栅极一路上拉至 V_{DD}。

(2)使能 IRQ 引脚以供使用后，IRQSC[IRQF]将置位，并且在使能中断前必须清零。针对 3V 系统中的下降沿和电平灵敏度配置引脚时，必须至少等待标志清零以及中断使能之间的周期时间长度。

● 8.6.1.2 边沿和电平灵敏度

IRQS[IRQMOD]控制字段重新配置检测逻辑，以便它能检测边沿事件和引脚电平。在该检测模式下，如果 IRQ 引脚从无效电平变为有效电平，则在检测到边沿时 IRQSC[IRQF]状态标志置位；但只要 IRQ 引脚保持在有效电平，该标志就会继续置位并且无法清零。

■ 8.6.2 中断引脚请求寄存器

表 8-37 IRQ 存储器映射

绝对地址（16 进制）	寄存器名称	偏移量	位宽	访问	复位值
4003_1000	中断引脚请求状态和控制寄存器（IRQ_SC）	0h	8	R/W	00h

● 8.6.2.1 中断引脚请求状态和控制寄存器（IRQ_SC）

该直接页面寄存器包括状态位和控制位，用于配置 IRQ 功能、报告状态和应答 IRQ 事件。

地址：4003_1000h 基准 + 0h 偏移 = 4003_1000h

位	7	6	5	4	3	2	1	0
读	0	IRQPDD	IRQEDG	IRQPE	IRQF		IRQIE	IRQMOD
写						IRQACK		
复位值	0	0	0	0	0	0	0	0

表 8-38 IRQ_SC 字段描述

位	描述
7 保留	此字段为保留字段。 此只读字段为保留字段且值始终为 0。
6 IRQPDD	中断请求(IRQ)上拉器件禁用 当 IRQ 引脚使能时(IRQPE = 1)，该读/写控制位用于禁用内部上拉器件，从而允许使用外部器件。 0 若 IRQPE = 1，则 IRQ 上拉器件使能。 1 若 IRQPE = 1，则 IRQ 上拉器件禁用。
5 IRQEDG	中断请求(IRQ)边沿选择 该读/写控制字段用于选择使 IRQF 置位的 IRQ 引脚上的边沿极性或电平。IRQMOD 控制字段确定 IRQ 引脚是对边沿和电平都敏感还是只对边沿敏感。当 IRQ 引脚使能用作 IRQ 输入并配置为检测上升沿时，可选上拉电阻禁用。 0 IRQ 对下降沿或下降沿/低电平敏感。 1 IRQ 对上升沿或上升沿/高电平敏感。
4 IRQPE	IRQ 引脚使能 该读/写控制字段使能 IRQ 引脚功能。该字段置位后，IRQ 引脚可用作中断请求。 0 IRQ 引脚功能禁用。 1 IRQ 引脚功能使能。
3 IRQF	IRQ 标志 该只读状态字段指示何时发生中断请求事件。 0 无 IRQ 请求 1 检测到 IRQ 事件。
2 IRQACK	IRQ 应答 该只写字段用于应答中断请求事件（写入 1 可清零 IRQF）。写入 0 则无意义或无效。读取操作始终返回 0。如果选择边沿和电平检测(IRQMOD = 1)，则无法清零 IRQF，同时 IRQ 引脚保持在其有效电平。
1 IRQIE	IRQ 中断使能 该读/写控制字段确定 IRQ 事件是否生成中断请求。 0 IRQF 置位时禁止发送中断请求（使用轮询） 。 1 只要 IRQF = 1 就发送中断请求。
0 IRQMOD	IRQ 检测模式 该读/写控制字段选择仅边沿检测或边沿和电平检测。 0 仅在下降沿/上升沿检测 IRQ 事件。 1 在下降沿/上升沿和低电平/高电平检测 IRQ 事件。

版本历史

版本	发布日期	更新内容
V1.0	2016.05.18	初次发布
V1.1	2016.07.18	更新 6.3 ETM 模块
V1.11	2016.08.23	第 7 章增加功能描述和第 8 章管脚定义更新
V1.12	2016.10.08	第 1.7 节图 1-3 和图 1-4 第 8 章 311 页 8.5.1 特性
V1.13	2016.12.11	P290、P291 图 8-1,表 8-1
V1.14	2017.01.18	SIM_SOPT NIME 控制位说明、ADC_SC1 寄存器说明

注意事项 Notice

本公司不建议将该系列芯片超规范参数使用，或应用于易对生命安全、人体健康、财产安全、环境保护等方面造成危害的产品中，未经本公司安全确认或由于客户设计缺陷造成的损失，客户自行承担风险。本说明书版本更新不做另行通知，想了解最新以及更多信息请访问官方网站：www.navota.com，所有解释权归纳瓦特公司所有。

The company is not recommended specification parameters of super of the series of chips, or used in the safety of life, health, property safety, protecting the environment caused by the harm of the product in, not by the company safety confirmation or due to losses caused by defects in the customer design, the customer is at your own risk.

The manual version updates do not notice, want to know the latest and more information please visit the official website: www.navota.com.