成都翌创微电子有限公司
Chengdu ET Microelectronics Co., Ltd.

ET6002

32 位 Arm® Cortex®-M7 微控制器（MCU/DSP)

专注实时控制，铸造安全可靠中国芯

用户手册

文档版本　1.0.4

发布日期　2025-03-31

商标声明

ETMCU和其他翌创微商标均为成都翌创微电子有限公司的商标。

本文档提及的其他所有商标或注册商标，由各自的所有人拥有。

注意

您购买的产品、服务或特性等应受成都翌创微电子有限公司商业合同和条款的约束，本文档中描述的全部或部分产品、服务或特性可能不在您的购买或使用范围之内。除非合同另有约定，成都翌创微电子有限公司对本文档内容不做任何明示或默示的声明或保证。

由于产品版本升级或其他原因，本文档内容会不定期进行更新。除非另有约定，本文档仅作为使用指导，本文档中的所有陈述、信息和建议不构成任何明示或暗示的担保。

成都翌创微电子有限公司

地址：四川省成都市高新区高朋大道 3 号东方希望大厦 9 楼 901 号
电话：028-83350520
网址：https://www.etmcu.com
邮箱：support@etmcu.com

目录

1 芯片概述

1.1 功能描述

ET6002 是一款功能丰富的高性能数模混合微控制器 (MCU/DSP)，内部集成 1 个 32 位高性能 ARM Cortex-M7 内核，工作时钟频率最高可达 300 MHz；内置 96 KB 片内 SRAM、512 KB 嵌入式程序 Flash (PFLASH)、128 KB 嵌入式数据 Flash (DFLASH) 和 2KB 的用户 OTP。

ET6002 集成了丰富的高性能模拟模块，包括 3 个独立的 12-bit ADC (最多 34 路模拟采样通道，采样速率最高可达 4 MSPS)，2 路独立的 12-bit Buffered DAC，4 路模拟窗口比较器 (ACMP)，也可以作为 8 路独立比较器使用。

ET6002 还集成了超高精度 PWM 模块 (SRPWM)，32-bit 增强型定时器 (ETIMER)，正交编码模块 (QEP)，Σ-Δ 滤波器模块 (SDFM)，64-bit 超级定时器 (STIMER)，硬件加速器 CORDIC、硬件 CRC、对称加解密 AES、FMAC 滤波算法加速模块，DMA 控制器以及 CAN FD、UART、I2C、SPI 等通信接口和外设资源。

ET6002 功能框图如下图所示。

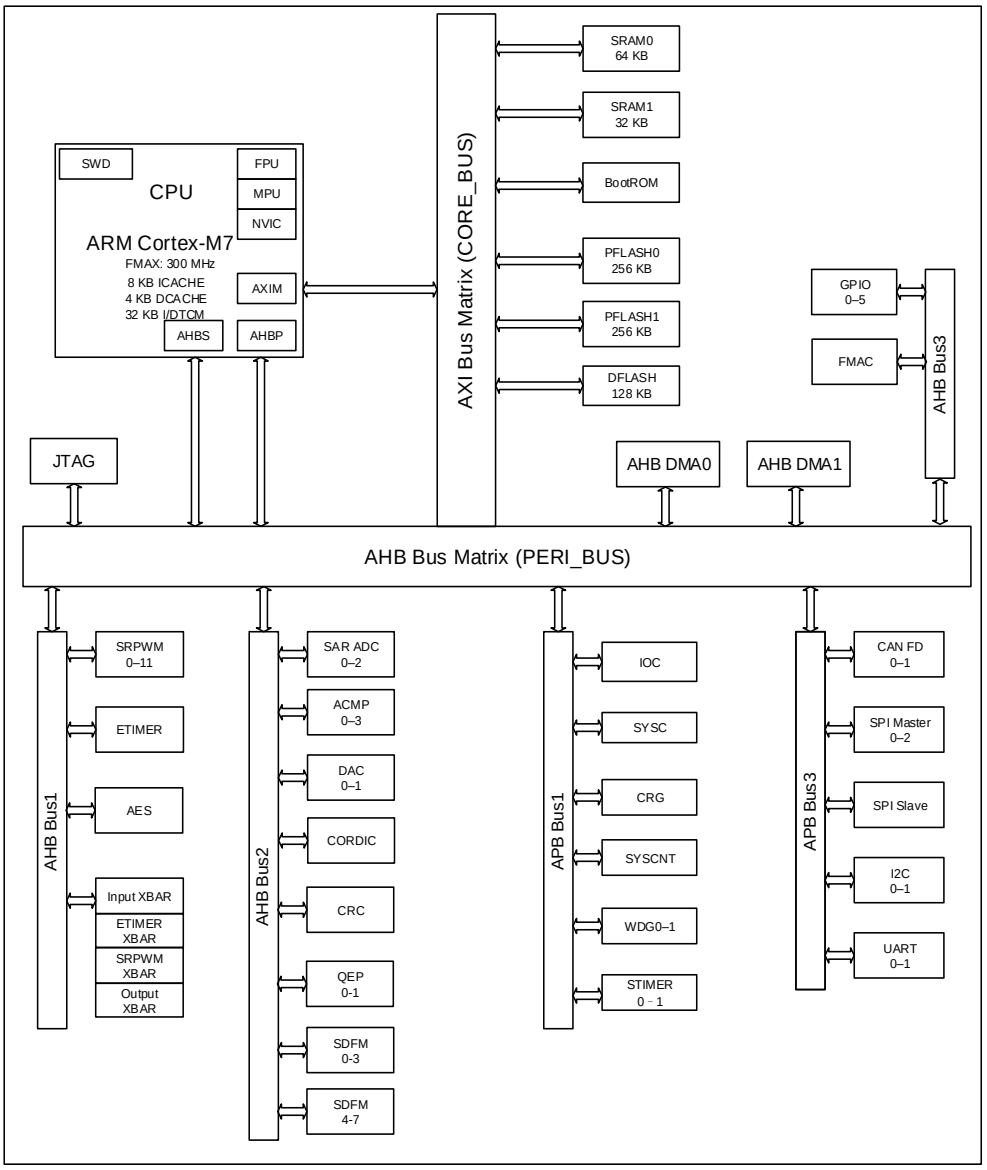


图1-1 ET6002 芯片功能框图

说明

APB Bus2 用于 Flash 和 SRAM 控制器等配置使用，上图未体现。

2 芯片特性

ET6002 是一款功能强大的高性能数模混合微控制器 (MCU/DSP)，支持特性如下：

- 处理器
  - ➢ 内置 32 位 ARM Cortex-M7 内核
    - 最高主频 300 MHz
    - 支持单精度浮点运算单元 (FPU)
    - 支持 DSP 指令 (乘加、移位)
    - 内存保护单元 (MPU) 支持多达 16 个内存区域 (Region) 的保护
    - 自带 8 KB 指令缓存 (ICACHE) 和 4 KB 数据缓存 (DCACHE)，支持 ECC 纠一检二保护功能
    - 自带 32 KB 指令内存 (ITCM) 和 32 KB 数据内存 (DTCM)，支持 ECC 纠一检二保护功能
    - 支持中断管理 (NVIC)
- 片内存储：640KB Flash+96KB SRAM
  - 内置高达 512 KB 的程序 Flash (PFLASH) (可配置为 2 个 256 KB Bank，以支持 256 KB OTA 烧写)，支持 ECC 纠一检二保护功能
  - 内置 128 KB 数据 Flash (DFLASH)，支持 ECC 纠一检二保护功能
  - 内置 2KB OTP
  - 提供高达 96 KB 的片内 SRAM，支持 ECC 纠一检二保护功能
- 时钟，复位
  - 支持芯片上电复位 (Power-on Reset, POR)
  - 内置 25 MHz 出厂校准振荡器
  - 支持 25 MHz 外部晶振输入或单端时钟输入
  - 内置锁相环 (PLL) 时钟
  - 支持芯片欠压复位 (Brown-out Reset, BOR)
  - 支持时钟丢失检测电路
- 供电
  - 支持 3.3 V 单电源供电
  - 内置 LDO 模块
  - 内置温度传感器
  - 支持欠压、过温检测与保护
- 系统外设
  - 内置 2 个 8 通道直接存储器访问 (DMA) 控制器
  - 最多 84 个独立可编程多路复用 GPIO 管脚
  - 内置 2 路窗口化看门狗模块 (Watchdog)
  - 2 个 64-bit 超级定时器 (STIMER)
  - 1 个 64-bit 系统定时器 (SYS_CNT)
- 信息安全
  - 安全启动
  - JTAG 权限访问
  - 对接加解密算法 AES128、AES256

- 存储安全保护
- 通用硬件加速器
  - 三角函数加速器 CORDIC，24 比特精度定点运算
  - 硬件 CRC
  - 对称加解密 AES128、AES256
  - FMAC 滤波算法加速
- 通信接口
  - 2 个 CAN FD 接口
  - 2 个 UART 接口
  - 2 个 I2C 接口，兼容 PMBus1.1
  - 3 个 SPI Master 接口
  - 1 个 SPI Slave 接口
- 模拟系统
  - 3 个 12-bit ADC
    - ✓ 最多 34 个外部通道
    - ✓ 采样速率最高可达 4 MSPS
    - ✓ 精度：
      - INL: −4~4LSB
      - DNL: −1~4LSB
    - ✓ 支持基于序列调度的顺序采样，单次采样，过采样，序列 Pattern 可配置
    - ✓ 支持 Watchdog 功能
    - ✓ 支持输出结果的硬件 IIR、非滑动平均滤波、增益控制、Offset 控制处理
    - ✓ 支持 Blanking 控制
  - 2 路 12-bit Buffered DAC
    - ✓ 精度：
      - INL: ±5 LSB
      - DNL: ±2 LSB
    - ✓ 刷新速率：≥ 1 MSPS
  - 4 对内置 12-bit DAC 的模拟窗口比较器 (ACMP)
    - ✓ 也可作为 8 路独立比较器使用
    - ✓ 支持 Blanking 控制
    - ✓ 支持对输出信号滤波
  - 温度传感器 (Tsensor)
- 增强型控制外设
  - 增强型定时器 (ETIMER)
    - ✓ 14 路 ETIMER，每路同时支持 CAP 和 PWM 功能，两个功能共一个时基计数器
    - ✓ 各路 PWM 可同步或独立运行
    - ✓ 14 路 PWM 功能可配置为互补 PWM 波，支持死区保护
    - ✓ 各路支持封波保护
  - 超高精度脉宽调制模块 (SRPWM)
    - ✓ 12 通道，24 路 PWM 输出，每个通道可输出一对互补或独立的 PWM 发波信号
    - ✓ 其中 8 路 PWM 支持高精度调整，分辨率为 156 ps
    - ✓ 互补 PWM 支持死区保护
    - ✓ 各发波通道间支持主从同步或独立运行
    - ✓ 各通道支持封波保护
  - 正交编码模块 (QEP)
    - ✓ 2 个 QEP 模块
    - ✓ 支持正交格式、方向脉冲计数格式、CW/CCW 格式输入
  - Σ-Δ 滤波器模块 (SDFM)
    - ✓ 8 个标准滤波 SDFM 模块
- IP 协同、端口灵活管理
  - XBAR 管理，支持 CLU、Input XBAR、Output XBAR、SRPWM XBAR、ETIMER XBAR
- 调试接口
  - 2 线 SWD 调试接口
- 工作温度
  - 工作结温：−40°C~125°C
- 封装

- 144 引脚 LQFP 封装
- 128 引脚 LQFP 封装
- 100 引脚 LQFP 封装
- 80 引脚 LQFP 封装

3 引脚定义

3.1 引脚框图

图 3-1 展示了 F60020PZ2I 采用 LQFP144 封装的引脚分配，图 3-2 展示了 F60020PY2I 采用 LQFP128 封装的引脚分配，图 3-3 展示了 F60020PV2I 采用 LQFP100 封装的引脚分配，图 3-4 展示了 F60020PY2I 采用 LQFP80 封装的引脚分配。

除 SWCLK 和 SWDIO 外，GPIO 引脚上仅显示 GPIO 功能，完整的 GPIO 复用关系请参考 3.4 "引脚复用"章节；AFE 引脚仅显示 SAR ADC 和 DAC 引脚功能，完整的 AFE 引脚复用功能请参考 3.2 "引脚属性"章节。

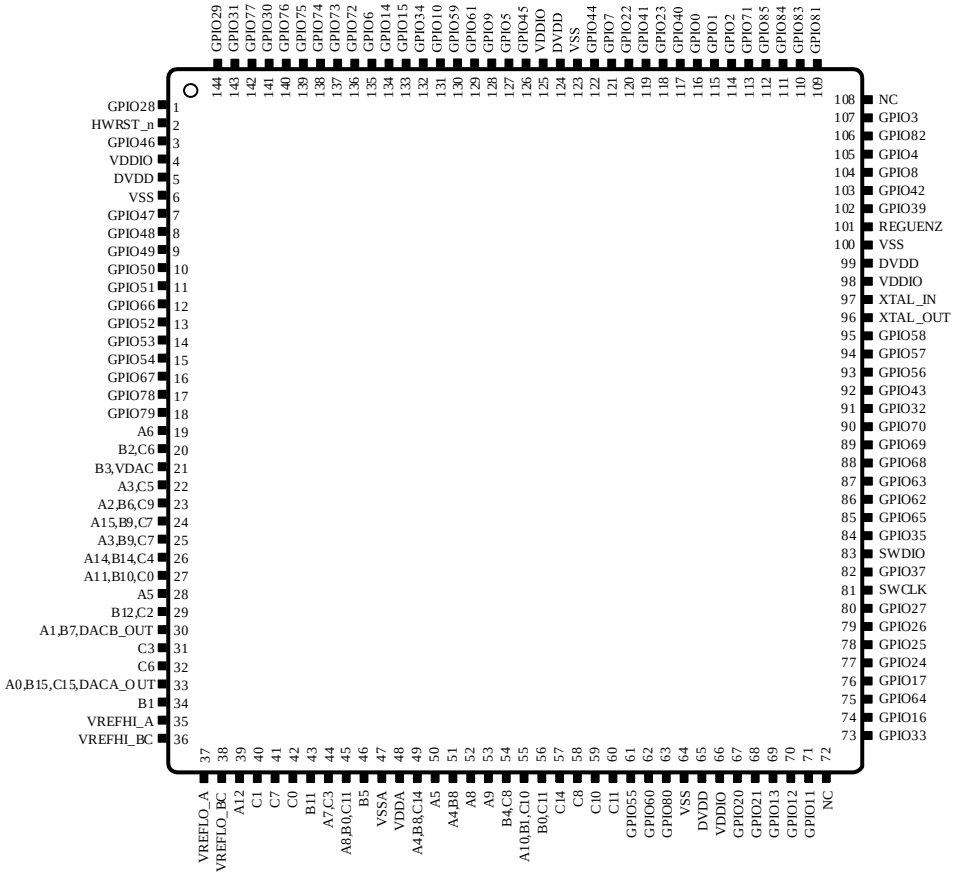


图3-1 F60020PZ2I 采用 LQFP144 封装的引脚分配 (顶视图)

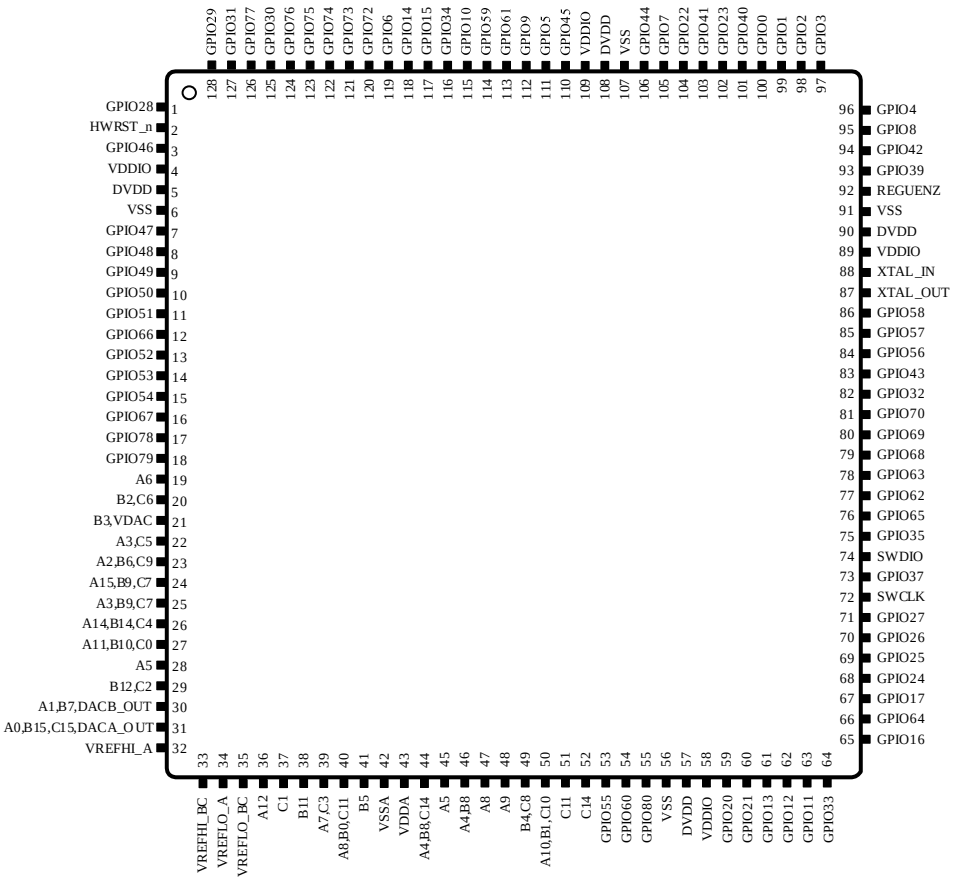


图3-2 F60020PY2I 采用 LQFP128 封装的引脚分配 (顶视图)

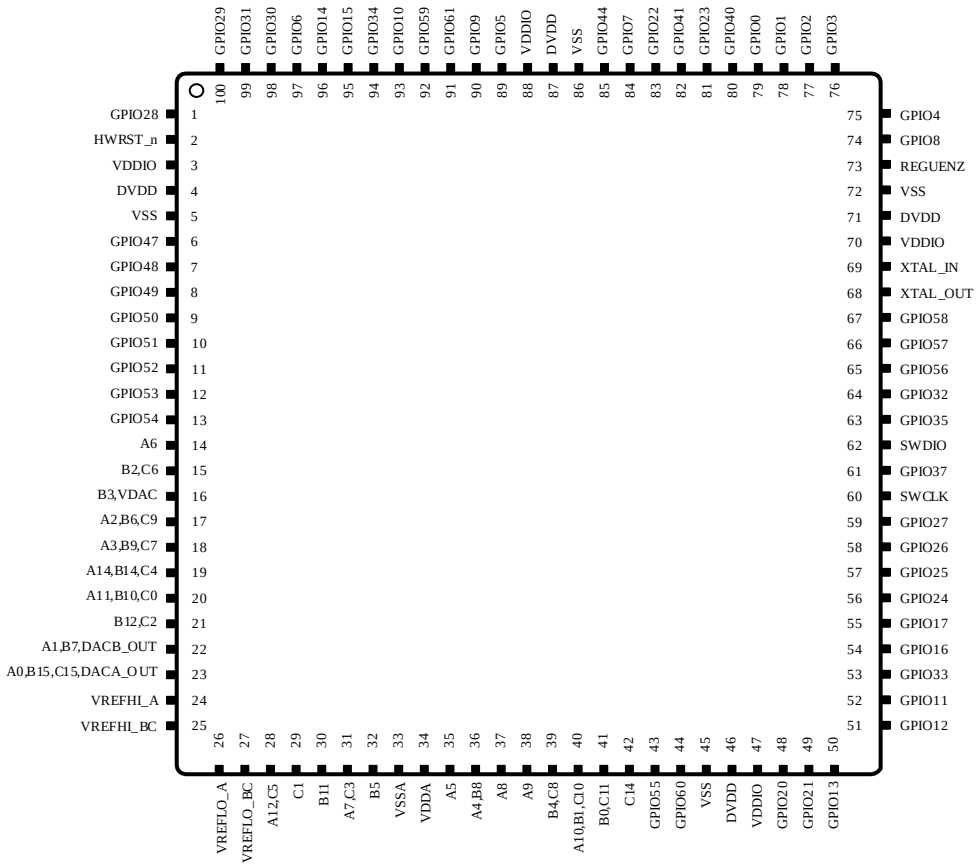


图3-3 F60020PV2I 采用 LQFP100 封装的引脚分配 (顶视图)

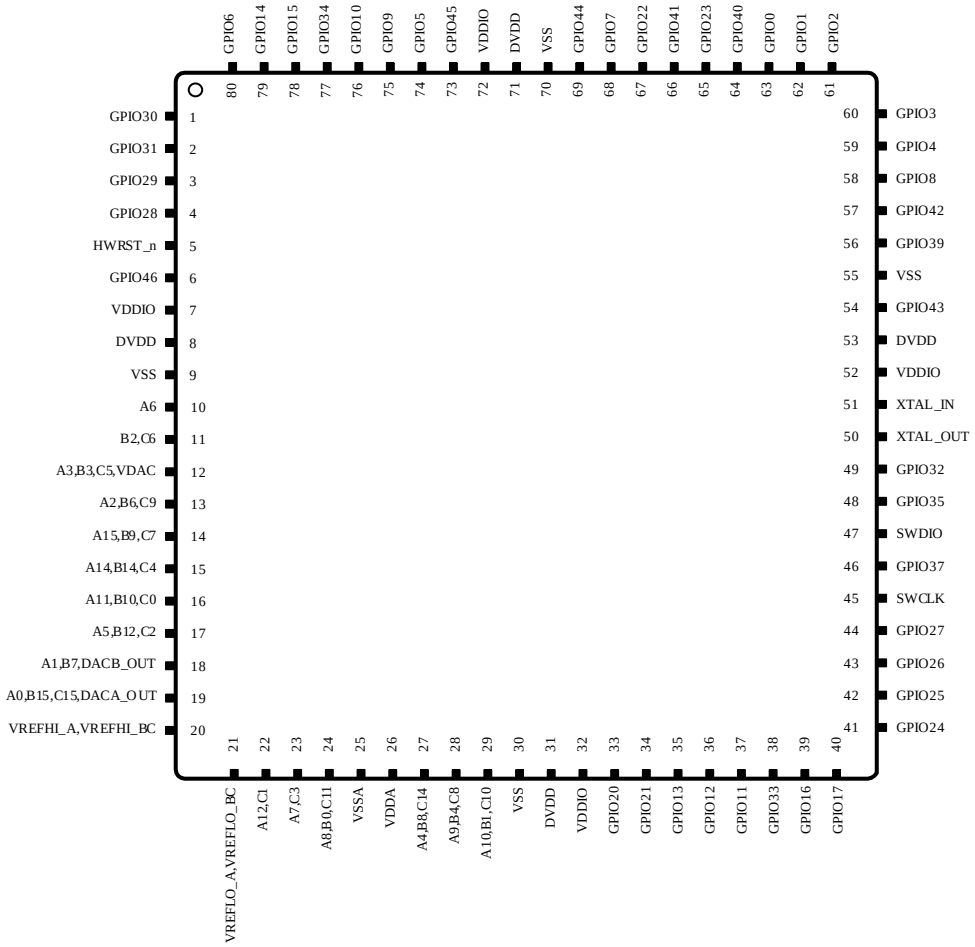


图3-4 F60020PE2I 采用 LQFP80 封装的引脚分配 (顶视图)

3.2 引脚属性

表3-1 ET6002 引脚属性

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
模拟						
19	23	31	33	SAR_A0	I	ADC-A 输入 0
				SAR_B15	I	ADC-B 输入 15
				SAR_C15	I	ADC-C 输入 15
				DACA_OUT	I	缓冲 DAC-A 输出
				CMP3_HP2	I	ACMP-3 高电平比较器正输入 2
				CMP3_LP2	I	ACMP-3 低电平比较器正输入 2
18	22	30	30	SAR_A1	I	ADC-A 输入 1
				SAR_B7	I	ADC-B 输入 7
				DACB_OUT	I	缓冲 DAC-B 输出
				CMP1_HP4	I	ACMP-1 高电平比较器正输入 4
				CMP1_LP4	I	ACMP-1 低电平比较器正输入 4
13	17	23	23	SAR_A2	I	ADC-A 输入 2
				SAR_B6	I	ADC-B 输入 6
				SAR_C9	I	ADC-C 输入 9
				CMP1_HP0	I	ACMP-1 高电平比较器正输入 0
				CMP1_LP0	I	ACMP-1 低电平比较器正输入 0
-	18	25	25	SAR_A3	I	ADC-A 输入 3
				SAR_B9	I	ADC-B 输入 9
				SAR_C7	I	ADC-C 输入 7

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				CMP3_HP5	I	ACMP-3 高电平比较器正输入 5
				CMP3_LP5	I	ACMP-3 低电平比较器正输入 5
-	36	46	51	SAR_A4	I	ADC-A 输入 4
				SAR_B8	I	ADC-B 输入 8
				CMP2_HP0	I	ACMP-2 高电平比较器正输入 0
				CMP2_LP0	I	ACMP-2 低电平比较器正输入 0
-	35	45	50	SAR_A5	I	ADC-A 输入 5
				CMP2_HP5	I	ACMP-2 高电平比较器正输入 5
				CMP2_LP5	I	ACMP-2 低电平比较器正输入 5
10	14	19	19	SAR_A6	I	ADC-A 输入 6
				CMP1_HP2	I	ACMP-1 高电平比较器正输入 2
				CMP1_LP2	I	ACMP-1 低电平比较器正输入 2
23	31	39	44	SAR_A7	I	ADC-A 输入 7
				SAR_C3	I	ADC-C 输入 3
				CMP4_HP1	I	ACMP-4 高电平比较器正输入 1
				CMP4_LP1	I	ACMP-4 低电平比较器正输入 1
				CMP4_HN1	I	ACMP-4 高电平比较器负输入 1
				CMP4_LN1	I	ACMP-4 低电平比较器负输入 1
-	37	47	52	SAR_A8	I	ADC-A 输入 8
				CMP4_HP4	I	ACMP-4 高电平比较器正输入 4
				CMP4_LP4	I	ACMP-4 低电平比较器正输入 4
28	38	48	53	SAR_A9	I	ADC-A 输入 9
				CMP2_HP2	I	ACMP-2 高电平比较器正输入 2
				CMP2_LP2	I	ACMP-2 低电平比较器正输入 2

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
29	40	50	55	SAR_A10	I	ADC-A 输入 10
				SAR_B1	I	ADC-B 输入 1
				SAR_C10	I	ADC-C 输入 10
				CMP2_HP3	I	ACMP-2 高电平比较器正输入 3
				CMP2_LP3	I	ACMP-2 低电平比较器正输入 3
				CMP2_HN0	I	ACMP-2 高电平比较器负输入 0
				CMP2_LN0	I	ACMP-2 低电平比较器负输入 0
16	20	27	27	SAR_A11	I	ADC-A 输入 11
				SAR_B10	I	ADC-B 输入 10
				SAR_C0	I	ADC-C 输入 0
				CMP1_HP1	I	ACMP-1 高电平比较器正输入 1
				CMP1_LP1	I	ACMP-1 低电平比较器正输入 1
				CMP1_HN1	I	ACMP-1 高电平比较器负输入 1
				CMP1_LN1	I	ACMP-1 低电平比较器负输入 1
-	41	51	56	SAR_B0	I	ADC-B 输入 0
				SAR_C11	I	ADC-C 输入 11
				CMP2_HP4	I	ACMP-2 高电平比较器正输入 4
				CMP2_LP4	I	ACMP-2 低电平比较器正输入 4
-	-	-	34	SAR_B1	I	ADC-B 输入 1
11	15	20	20	SAR_B2	I	ADC-B 输入 2
				SAR_C6	I	ADC-C 输入 6
				CMP3_HP0	I	ACMP-3 高电平比较器正输入 0
				CMP3_LP0	I	ACMP-3 低电平比较器正输入 0
12	16	21	21	SAR_B3	I	ADC-B 输入 3

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				VDAC	I	片上 DAC 的可选外部基准电压
				CMP3_HP3	I	ACMP-3 高电平比较器正输入 3
				CMP3_LP3	I	ACMP-3 低电平比较器正输入 3
				CMP3_HN0	I	ACMP-3 高电平比较器负输入 0
				CMP3_LN0	I	ACMP-3 低电平比较器负输入 0
28	39	49	54	SAR_B4	I	ADC-B 输入 4
				SAR_C8	I	ADC-C 输入 8
				CMP4_HP0	I	ACMP-4 高电平比较器正输入 0
				CMP4_LP0	I	ACMP-4 低电平比较器正输入 0
	32	41	46	SAR_B5	I	ADC-B 输入 5
				CMP1_HP5	I	ACMP-1 高电平比较器正输入 5
				CMP1_LP5	I	ACMP-1 低电平比较器正输入 5
12	-	22	22	SAR_A3	I	ADC-A 输入 3
				SAR_C5	I	ADC-C 输入 5
14	-	24	24	SAR_A15	I	ADC-A 输入 15
				SAR_B9	I	ADC-B 输入 9
				SAR_C7	I	ADC-C 输入 7
				CMP1_HP3	I	ACMP-1 高电平比较器正输入 3
				CMP1_LP3	I	ACMP-1 低电平比较器正输入 3
				CMP1_HN0	I	ACMP-1 高电平比较器负输入 0
				CMP1_LN0	I	ACMP-1 低电平比较器负输入 0
17	-	28	28	SAR_A5	I	ADC-A 输入 5
24	-	40	45	SAR_A8	I	ADC-A 输入 8
				SAR_B0	I	ADC-B 输入 0

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				SAR_C11	I	ADC-C 输入 11
				CMP2_HP4	I	ACMP-2 高电平比较器正输入 4
				CMP4_HP4	I	ACMP-4 高电平比较器正输入 4
				CMP2_LP4	I	ACMP-2 低电平比较器正输入 4
				CMP4_LP4	I	ACMP-4 低电平比较器正输入 4
27	-	44	49	SAR_A4	I	ADC-A 输入 4
				SAR_B8	I	ADC-B 输入 8
				SAR_C14	I	ADC-C 输入 14
				CMP2_HP0	I	ACMP-2 高电平比较器正输入 0
				CMP4_HP3	I	ACMP-4 高电平比较器正输入 3
				CMP2_LP0	I	ACMP-2 低电平比较器正输入 0
				CMP4_LP3	I	ACMP-4 低电平比较器正输入 3
				CMP4_HN0	I	ACMP-4 高电平比较器负输入 0
				CMP4_LN0	I	ACMP-4 低电平比较器负输入 0
-	30	38	43	SAR_B11	I	ADC-B 输入 11
				CMP4_HP5	I	ACMP-4 高电平比较器正输入 5
				CMP4_LP5	I	ACMP-4 低电平比较器正输入 5
-	-	-	42	SAR_C0	I	ADC-C 输入 0
22	29	37	40	SAR_C1	I	ADC-C 输入 1
				CMP4_HP2	I	ACMP-4 高电平比较器正输入 2
				CMP4_LP2	I	ACMP-4 低电平比较器正输入 2
17	21	29	29	SAR_B12	I	ADC-B 输入 12
				SAR_C2	I	ADC-C 输入 2
				CMP3_HP1	I	ACMP-3 高电平比较器正输入 1

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				CMP3_LP1	I	ACMP-3 低电平比较器正输入 1
				CMP3_HN1	I	ACMP-3 高电平比较器负输入 1
				CMP3_LN1	I	ACMP-3 低电平比较器负输入 1
-	-	-	31	SAR_C3	I	ADC-C 输入 3
15	19	26	26	SAR_A14	I	ADC-A 输入 14
				SAR_B14	I	ADC-B 输入 14
				SAR_C4	I	ADC-C 输入 4
				CMP3_HP4	I	ACMP-3 高电平比较器正输入 4
				CMP3_LP4	I	ACMP-3 低电平比较器正输入 4
22	28	36	39	SAR_A12	I	ADC-A 输入 12
				CMP2_HP1	I	ACMP-2 高电平比较器正输入 1
				CMP2_LP1	I	ACMP-2 低电平比较器正输入 1
				CMP2_HN1	I	ACMP-2 高电平比较器负输入 1
				CMP2_LN1	I	ACMP-2 低电平比较器负输入 1
-	-	-	32	SAR_C6	I	ADC-C 输入 6
-	-	-	41	SAR_C7	I	ADC-C 输入 7
-	-	-	58	SAR_C8	I	ADC-C 输入 8
	42	52	57	SAR_C14	I	ADC-C 输入 14
				CMP4_HP3	I	ACMP-4 高电平比较器正输入 3
				CMP4_LP3	I	ACMP-4 低电平比较器正输入 3
				CMP4_HN0	I	ACMP-4 高电平比较器负输入 0
				CMP4_LN0	I	ACMP-4 低电平比较器负输入 0
-	-	-	59	SAR_C10	I	ADC-C 输入 10
-	-	-	60	SAR_C11	I	ADC-C 输入 11

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
20	24	32	35	VREFHI_A	I	ADC-A 高基准电压。在外部基准模式下，从外部驱动这个引脚上的高基准电压。在内部基准模式下，电压由器件驱动到该引脚。在任一模式下，在此引脚上放置至少一个 2.2 μF 电容器。此电容器应放置 VREFHI_A 和 VREFLO_A 引脚之间尽可能靠近器件的位置。
21	26	34	37	VREFLO_A	I	ADC-A 低基准电压
20	25	33	36	VREFHI_BC	I	ADC-B/C 高基准电压。在外部基准模式下，从外部驱动这个引脚上的高基准电压。在内部基准模式下，电压由器件驱动到该引脚。在任一模式下，在此引脚上放置至少一个 2.2 μF 电容器。此电容器应放置 VREFHI_BC 和 VREFLO_BC 引脚之间尽可能靠近器件的位置。
21	27	35	38	VREFLO_BC	I	ADC-B/C 低基准电压
GPIO						
4	1	1	1	GPIO28	I/O	通用输入/输出 28
				UART0_RX	I	UART0 接收
				SRPWM6_A	O	SRPWM6A 输出
				UART1_TX	O	UART1 输出
				OUTXBAR4	O	输出 X-BAR 输出 4
				EQEP0_A	I	EQEP0 输入 A

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				SDFM1_DATA3	I	SDFM1 通道 3 数据输入
				EQEP1_STROBE	I	EQEP1 选通
				SPIM1_CLK	O	SPIM1 时钟
				I2C1_SDA	I/OD	I2C1 开漏双向数据
6	-	3	3	GPIO46	I/O	通用输入/输出 46
				UART1_RX	I	UART1 接收
				I2C1_CTL_n	I/OD	I2C1 控制信号-从器件输入/主器件输出（低有效）
				CANFD1_TX	O	CAN FD1 发送
				I2C0_SDA	I/OD	I2C0 开漏双向数据
				SDFM1_CLK4	I	SDFM1 通道 4 时钟输入
-	6	7	7	GPIO47	I/O	通用输入/输出 47
				CANFD1_RX	I	CAN FD1 接收
				I2C0_SCL	I/OD	I2C0 开漏双向时钟
				SDFM1_DATA4	I	SDFM1 通道 4 数据输入
-	7	8	8	GPIO48	I/O	通用输入/输出 48
				OUTXBAR2	O	输出 X-BAR 输出 2
				CANFD0_TX	O	CAN FD0 发送
				UART0_TX	O	UART0 输出
				SDFM0_DATA1	I	SDFM0 通道 1 数据输入
				I2C0_SDA	I/OD	I2C0 开漏双向数据
-	8	9	9	GPIO49	I/O	通用输入/输出 49
				OUTXBAR3	O	输出 X-BAR 输出 3
				SPIS0_CLK	I	SPIS0 时钟

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				CANFD0_RX	I	CAN FD0 接收
				SPIM0_CLK	O	SPIM0 时钟
				UART0_RX	I	UART0 接收
				SDFM0_CLK1	I	SDFM0 通道 1 时钟输入
				SDFM1_DATA1	I	SDFM1 通道 1 数据输入
-	9	10	10	GPIO50	I/O	通用输入/输出 50
				EQEP0_A	I	EQEP0 输入 A
				UART0_RX	I	UART0 接收
				CANFD1_TX	O	CAN FD1 发送
				SPIM1_MOSI	O	SPIM1 主器件输出，从器件输入
				SDFM0_DATA2	I	SDFM0 通道 2 数据输入
				I2C1_SDA	I/OD	I2C1 开漏双向数据
				SDFM1_DATA2	I	SDFM1 通道 2 数据输入
-	10	11	11	GPIO51	I/O	通用输入/输出 51
				EQEP0_B	I	EQEP0 输入 B
				UART0_TX	O	UART0 输出
				CANFD1_RX	I	CAN FD1 接收
				SPIM1_MISO	I	SPIM1_主器件输入，从器件输出备份
				SDFM0_CLK2	I	SDFM0 通道 2 时钟输入
				I2C1_SCL	I/OD	I2C1 开漏双向时钟
				SDFM1_DATA3	I	SDFM1 通道 3 数据输入
-	-	12	12	GPIO66	I/O	通用输入/输出 66
				SPIM0_CS0_n	O	SPIM0 从器件片选 0（低有效）

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				OUTXBAR9	O	输出 X-BAR 输出 9
-	11	13	13	GPIO52	I/O	通用输入/输出 52
				EQEP0_STROBE	I	EQEP0 选通
				SPIM1_CLK	O	SPIM1 时钟
				SDFM0_DATA3	I	SDFM0 通道 3 数据输入
				SYNCOUT	O	ETIMER 同步信号输出
				SDFM1_DATA4	I	SDFM1 通道 4 数据输入
-	12	14	14	GPIO53	I/O	通用输入/输出 53
				EQEP0_INDEX	I	EQEP0 索引
				I2C1_CTL_n	I/OD	I2C1 控制信号-从器件输入/主器件输出（低有效）
				SPIM1_MOSI	O	SPIM1 主器件输出，从器件输入
				SPIM1_CS0_n	O	SPIM1 从器件片选 0（低有效）
				SDFM0_CLK3	I	SDFM0 通道 3 时钟输入
				CANFD0_RX	I	CAN FD0 接收
				SDFM0_CLK1	I	SDFM0 通道 1 时钟输入
-	13	15	15	GPIO54	I/O	通用输入/输出 54
				SPIM0_MOSI	O	SPIM0 主器件输出，从器件输入
				I2C1_ALERT_n	I/OD	I2C1 控制信号-从器件输入/主器件输出（低有效）
				SPIM1_MISO	I	SPIM1_主器件输入，从器件输出备份
				EQEP1_A	I	EQEP1 输入 A
				OUTXBAR1	O	输出 X-BAR 输出 1

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				SDFM0_DATA4	I	SDFM0 通道 4 数据输入
				SDFM0_CLK2	I	SDFM0 通道 2 时钟输入
-	-	16	16	GPIO67	I/O	通用输入/输出 67
				SPIM1_CS0_n	O	SPIM1 从器件片选 0（低有效）
				CANFD1_TX	O	CAN FD1 发送
				I2C1_CTL_n	I/OD	I2C1 控制信号-从器件输入/主器件输出（低有效）
				OUTXBAR12	O	输出 X-BAR 输出 12
-	-	17	17	GPIO78	I/O	通用输入/输出 78
				SPIM1_CS1_n	O	SPIM1 从器件片选 1（低有效）
				CANFD1_RX	I	CAN FD1 接收
				I2C1_ALERT_n	I/OD	I2C1 控制信号-从器件输入/主器件输出（低有效）
				OUTXBAR10	O	输出 X-BAR 输出 10
-	-	18	18	GPIO79	I/O	通用输入/输出 79
				SPIM1_CS2_n	O	SPIM1 从器件片选 2（低有效）
				SPIM0_CS1_n	O	SPIM0 从器件片选 1（低有效）
				ETIMER10	O	通用定时器输出 10
				OUTXBAR11	O	输出 X-BAR 输出 11
-	43	53	61	GPIO55	I/O	通用输入/输出 55
				SPIM0_MISO	I	SPIM0_主器件输入，从器件输出备份
				SRPWM9_A	O	SRPWM9A 输出
				EQEP1_B	I	EQEP1 输入 B

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				OUTXBAR2	O	输出 X-BAR 输出 2
				SDFM0_CLK4	I	SDFM0 通道 4 时钟输入
				SDFM0_CLK3	I	SDFM0 通道 3 时钟输入
-	44	54	62	GPIO60	I/O	通用输入/输出 60
				CANFD1_TX	O	CAN FD1 发送
				SRPWM9_B	O	SRPWM9B 输出
				OUTXBAR2	O	输出 X-BAR 输出 2
				SPIM1_MOSI	O	SPIM1 主器件输出，从器件输入
				SDFM1_DATA3	I	SDFM1 通道 3 数据输入
				SDFM1_CLK4	I	SDFM1 通道 4 时钟输入
-	-	55	63	GPIO80	I/O	通用输入/输出 80
				SRPWM8_B	O	SRPWM8B 输出
				I2C1_ALERT_n	I/OD	I2C1 控制信号-从器件输入/主器件输出（低有效）
33	48	59	67	GPIO20	I/O	通用输入/输出 20
				EQEP0_A	I	EQEP0 输入 A
				TESTPIN0	O	测试引脚 0
				SRPWM10_A	O	SRPWM10A 输出
				SPIM1_MOSI	O	SPIM1 主器件输出，从器件输入
				SDFM0_DATA3	I	SDFM0 通道 3 数据输入
				CANFD1_TX	O	CAN FD1 发送
				ECAP10	I	增强型捕获输入 10
34	49	60	68	GPIO21	I/O	通用输入/输出 21
				EQEP0_B	I	EQEP0 输入 B

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				TESTPIN1	O	测试引脚 1
				SRPWM10_B	O	SRPWM10B 输出
				SPIM1_MISO	I	SPIM1_主器件输入，从器件输出备份
				SDFM0_CLK3	I	SDFM0 通道 3 时钟输入
				CANFD1_RX	I	CAN FD1 接收
				ECAP11	I	增强型捕获输入 11
35	50	61	69	GPIO13	I/O	通用输入/输出 13
				SRPWM6_B	O	SRPWM6B 输出
				CANFD1_TX	O	CAN FD1 发送
				SRPWM11_A	O	SRPWM11A 输出
				EQEP0_INDEX	I	EQEP0 索引
				UART1_RX	I	UART1 接收
				I2C0_ALERT_n	I/OD	I2C0 控制信号-从器件输入/主器件输出（低有效）
				ECAP12	I	增强型捕获输入 12
				SPIM0_MISO	I	SPIM0_主器件输入，从器件输出备份
				CANFD0_TX	O	CAN FD0 发送
36	51	62	70	GPIO12	I/O	通用输入/输出 12
				SRPWM6_A	O	SRPWM6A 输出
				CANFD1_RX	I	CAN FD1 接收
				SRPWM11_B	O	SRPWM11B 输出
				EQEP0_STROBE	I	EQEP0 选通

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				UART1_TX	O	UART1 输出
				I2C0_CTL_n	I/OD	I2C0 控制信号-从器件输入/主器件输出（低有效）
				ECAP13	I	增强型捕获输入 13
				SPIM0_CLK	O	SPIM0 时钟
				CANFD0_RX	I	CAN FD0 接收
37	52	63	71	GPIO11	I/O	通用输入/输出 11
				SRPWM5_B	O	SRPWM5B 输出
				OUTXBAR6	O	输出 X-BAR 输出 6
				ETIMER0	O	通用定时器输出 0
				EQEP0_B	I	EQEP0 输入 B
				UART1_RX	I	UART1 接收
				SPIM0_CS0_n	O	SPIM0 从器件片选 0（低有效）
				EQEP1_A	I	EQEP1 输入 A
				SPIM0_MOSI	O	SPIM0 主器件输出，从器件输入
38	53	64	73	GPIO33	I/O	通用输入/输出 33
				I2C0_SCL	I/OD	I2C0 开漏双向时钟
				SPIM1_CS0_n	O	SPIM1 从器件片选 0（低有效）
				ETIMER1	O	通用定时器输出 1
				OUTXBAR3	O	输出 X-BAR 输出 3
				SDFM0_CLK2	I	SDFM0 通道 2 时钟输入
				CANFD0_RX	I	CAN FD0 接收
				EQEP1_B	I	EQEP1 输入 B
				SDFM0_CLK1	I	SDFM0 通道 1 时钟输入

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
39	54	65	74	GPIO16	I/O	通用输入/输出 16
				SPIM0_MOSI	O	SPIM0 主器件输出，从器件输入
				OUTXBAR6	O	输出 X-BAR 输出 6
				ETIMER2	O	通用定时器输出 2
				SRPWM4_A	O	SRPWM4A 输出
				UART0_TX	O	UART0 输出
				SDFM0_DATA1	I	SDFM0 通道 1 数据输入
				EQEP0_STROBE	I	EQEP0 选通
				I2C0_SCL	I/OD	I2C0 开漏双向时钟
				EQEP1_B	I	EQEP1 输入 B
				SPIM1_MISO	I	SPIM1_主器件输入，从器件输出备份
-	-	66	75	GPIO64	I/O	通用输入/输出 64
				SRPWM8_A	O	SRPWM8A 输出
				I2C1_CTL_n	I/OD	I2C1 控制信号-从器件输入/主器件输出（低有效）
40	55	67	76	GPIO17	I/O	通用输入/输出 17
				SPIM0_MISO	I	SPIM0_主器件输入，从器件输出备份
				OUTXBAR7	O	输出 X-BAR 输出 7
				ETIMER6	O	通用定时器输出 6
				SRPWM4_B	O	SRPWM4B 输出
				UART0_RX	I	UART0 接收
				SDFM0_CLK1	I	SDFM0 通道 1 时钟输入

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				EQEP0_INDEX	I	EQEP0 索引
				I2C0_SDA	I/OD	I2C0 开漏双向数据
				CANFD0_TX	O	CAN FD0 发送
41	56	68	77	GPIO24	I/O	通用输入/输出 24
				OUTXBAR0	O	输出 X-BAR 输出 0
				EQEP1_A	I	EQEP1 输入 A
				SRPWM7_A	O	SRPWM7A 输出
				SPIM1_MOSI	O	SPIM1 主器件输出，从器件输入
				SDFM1_DATA1	I	SDFM1 通道 1 数据输入
				I2C0_SCL	I/OD	I2C0 开漏双向时钟
				UART0_TX	O	UART0 输出
42	57	69	78	GPIO25	I/O	通用输入/输出 25
				OUTXBAR1	O	输出 X-BAR 输出 1
				EQEP1_B	I	EQEP1 输入 B
				ETIMER3	O	通用定时器输出 3
				EQEP0_A	I	EQEP0 输入 A
				SPIM1_MISO	I	SPIM1_主器件输入，从器件输出备份
				SDFM1_CLK1	I	SDFM1 通道 1 时钟输入
				I2C0_SDA	I/OD	I2C0 开漏双向数据
				UART0_RX	I	UART0 接收
43	58	70	79	GPIO26	I/O	通用输入/输出 26
				OUTXBAR2	O	输出 X-BAR 输出 2
				EQEP1_INDEX	I	EQEP1 索引

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				TESTPIN2	O	测试引脚 2
				ETIMER4	O	通用定时器输出 4
				OUTXBAR2	O	输出 X-BAR 输出 2
				SPIM1_CLK	O	SPIM1 时钟
				SDFM1_DATA2	I	SDFM1 通道 2 数据输入
				I2C0_CTL_n	I/OD	I2C0 控制信号-从器件输入/主器件输出（低有效）
				I2C0_SDA	I/OD	I2C0 开漏双向数据
44	59	71	80	GPIO27	I/O	通用输入/输出 27
				OUTXBAR3	O	输出 X-BAR 输出 3
				EQEP1_STROBE	I	EQEP1 选通
				TESTPIN3	O	测试引脚 3
				ETIMER5	O	通用定时器输出 5
				OUTXBAR3	O	输出 X-BAR 输出 3
				SPIM1_CS0_n	O	SPIM1 从器件片选 0（低有效）
				SDFM1_CLK2	I	SDFM1 通道 2 时钟输入
				I2C0_ALERT_n	I/OD	I2C0 控制信号-从器件输入/主器件输出（低有效）
				I2C0_SCL	I/OD	I2C0 开漏双向时钟
45	60	72	81	SWCLK	O	SWD 时钟
				GPIO18	I/O	通用输入/输出 18
46	61	73	82	GPIO37	I/O	通用输入/输出 37
				OUTXBAR1	O	输出 X-BAR 输出 1
				I2C0_SCL	I/OD	I2C0 开漏双向时钟

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				ECAP0	I	增强型捕获输入 0
				UART0_TX	O	UART0 输出
				CANFD0_TX	O	CAN FD0 发送
				EQEP0_B	I	EQEP0 输入 B
				I2C0_ALERT_n	I/OD	I2C0 控制信号-从器件输入/主器件输出（低有效）
47	62	74	83	SWDIO	I/O	SWD 数据
				GPIO19	I/O	通用输入/输出 19
48	63	75	84	GPIO35	I/O	通用输入/输出 35
				UART0_RX	I	UART0 接收
				I2C0_SDA	I/OD	I2C0 开漏双向数据
				ECAP1	I	增强型捕获输入 1
				CANFD0_RX	I	CAN FD0 接收
				I2C0_SCL	I/OD	I2C0 开漏双向时钟
				EQEP0_A	I	EQEP0 输入 A
				I2C0_CTL_n	I/OD	I2C0 控制信号-从器件输入/主器件输出（低有效）
				SRPWM4_B	O	SRPWM4B 输出
				SDFM1_CLK1	I	SDFM1 通道 1 时钟输入
-	-	76	85	GPIO65	I/O	通用输入/输出 65
				EQEP0_B	I	EQEP0 输入 B
				ETIMER8	O	通用定时器输出 8
				OUTXBAR12	O	输出 X-BAR 输出 12
-	-	77	86	GPIO62	I/O	通用输入/输出 62

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				EQEP0_INDEX	I	EQEP0 索引
				ETIMER9	O	通用定时器输出 9
				OUTXBAR10	O	输出 X-BAR 输出 10
-	-	78	87	GPIO63	I/O	通用输入/输出 63
				SPIM2_CLK	O	SPIM2 时钟
-	-	79	88	GPIO68	I/O	通用输入/输出 68
				SPIM2_MOSI	O	SPIM2 主器件输出，从器件输入
				SPIM1_CS3_n	O	SPIM1 从器件片选 3（低有效）
				ETIMER11	O	通用定时器输出 11
-	-	80	89	GPIO69	I/O	通用输入/输出 69
				SPIM2_MISO	I	SPIM2_主器件输入，从器件输出备份
				SPIM0_CS2_n	O	SPIM0 从器件片选 2（低有效）
				ETIMER12	O	通用定时器输出 12
-	-	81	90	GPIO70	I/O	通用输入/输出 70
				SPIM2_CS0_n	O	SPIM2 从器件片选 0（低有效）
				SPIM0_CS3_n	O	SPIM0 从器件片选 3（低有效）
				ETIMER13	O	通用定时器输出 13
				OUTXBAR13	O	输出 X-BAR 输出 13
49	64	82	91	GPIO32	I/O	通用输入/输出 32
				I2C0_SDA	I/OD	I2C0 开漏双向数据
				SPIM1_CLK	O	SPIM1 时钟
				SRPWM7_B	O	SRPWM7B 输出
				SDFM0_DATA2	I	SDFM0 通道 2 数据输入

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				CANFD0_TX	O	CAN FD0 发送
54	-	83	92	GPIO43	I/O	通用输入/输出 43
				SPIM2_CLK	O	SPIM2 时钟
				ECAP2	I	增强型捕获输入 2
				OUTXBAR8	O	输出 X-BAR 输出 8
				I2C0_SCL	I/OD	I2C0 开漏双向时钟
				EQEP0_INDEX	I	EQEP0 索引
-	65	84	93	GPIO56	I/O	通用输入/输出 56
				SPIM0_CLK	O	SPIM0 时钟
				CANFD1_TX	O	CAN FD1 发送
				ECAP3	I	增强型捕获输入 3
				EQEP1_STROBE	I	EQEP1 选通
				UART1_TX	O	UART1 输出
				SDFM1_DATA1	I	SDFM1 通道 1 数据输入
				SPIM1_MOSI	O	SPIM1 主器件输出，从器件输入
				I2C0_SDA	I/OD	I2C0 开漏双向数据
				EQEP0_A	I	EQEP0 输入 A
				SDFM0_CLK4	I	SDFM0 通道 4 时钟输入
-	66	85	94	GPIO57	I/O	通用输入/输出 57
				SPIM0_CS0_n	O	SPIM0 从器件片选 0（低有效）
				CANFD1_RX	I	CAN FD1 接收
				ECAP4	I	增强型捕获输入 4
				EQEP1_INDEX	I	EQEP1 索引
				UART1_RX	I	UART1 接收

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				SDFM1_CLK1	I	SDFM1 通道 1 时钟输入
				SPIM1_MISO	I	SPIM1_主器件输入，从器件输出备份
				I2C0_SCL	I/OD	I2C0 开漏双向时钟
				EQEP0_B	I	EQEP0 输入 B
-	67	86	95	GPIO58	I/O	通用输入/输出 58
				ECAP5	I	增强型捕获输入 5
				OUTXBAR0	O	输出 X-BAR 输出 0
				SPIM1_CLK	O	SPIM1 时钟
				SDFM1_DATA2	I	SDFM1 通道 2 数据输入
				CANFD0_TX	O	CAN FD0 发送
				EQEP0_STROBE	I	EQEP0 选通
				SDFM1_CLK2	I	SDFM1 通道 2 时钟输入
56	-	93	102	GPIO39	I/O	通用输入/输出 39
				SPIM2_MOSI	O	SPIM2 主器件输出，从器件输入
				ECAP6	I	增强型捕获输入 6
				EQEP0_INDEX	I	EQEP0 索引
57	-	94	103	GPIO42	I/O	通用输入/输出 42
				SPIM2_MISO	I	SPIM2_主器件输入，从器件输出备份
				EQEP0_A	I	EQEP0 输入 A
				ETIMER7	O	通用定时器输出 7
				OUTXBAR9	O	输出 X-BAR 输出 9
				I2C0_SDA	I/OD	I2C0 开漏双向数据

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
58	74	95	104	GPIO8	I/O	通用输入/输出 8
				SRPWM4_A	O	SRPWM4A 输出
				CANFD0_TX	O	CAN FD0 发送
				EQEP0_STROBE	I	EQEP0 选通
				UART0_TX	O	UART0 输出
				SPIM0_MOSI	O	SPIM0 主器件输出，从器件输入
				I2C0_SCL	I/OD	I2C0 开漏双向时钟
59	75	96	105	GPIO4	I/O	通用输入/输出 4
				SRPWM2_A	O	SRPWM2A 输出
				CANFD1_TX	O	CAN FD1 发送
				CANFD0_RX	I	CAN FD0 接收
				OUTXBAR2	O	输出 X-BAR 输出 2
				CANFD0_TX	O	CAN FD0 发送
				SPIM1_CLK	O	SPIM1 时钟
				EQEP1_STROBE	I	EQEP1 选通
-	-	-	106	GPIO82	I/O	通用输入/输出 82
				EQEP0_STROBE	I	EQEP0 选通
				ECAP7	I	增强型捕获输入 7
60	76	97	107	GPIO3	I/O	通用输入/输出 3
				SRPWM1_B	O	SRPWM1B 输出
				OUTXBAR1	O	输出 X-BAR 输出 1
				CANFD1_TX	O	CAN FD1 发送
				OUTXBAR1	O	输出 X-BAR 输出 1
				SPIM0_CLK	O	SPIM0 时钟

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				UART0_RX	I	UART0 接收
				I2C1_SCL	I/OD	I2C1 开漏双向时钟
				CANFD0_RX	I	CAN FD0 接收
-	-	-	109	GPIO81	I/O	通用输入/输出 81
				SPIM2_CLK	O	SPIM2 时钟
				OUTXBAR8	O	输出 X-BAR 输出 8
				ECAP8	I	增强型捕获输入 8
-	-	-	110	GPIO83	I/O	通用输入/输出 83
				EQEP0_STROBE	I	EQEP0 选通
				ECAP10	I	增强型捕获输入 10
				ETIMER10	O	通用定时器输出 10
				OUTXBAR11	O	输出 X-BAR 输出 11
-	-	-	111	GPIO84	I/O	通用输入/输出 84
				EQEP0_A	I	EQEP0 输入 A
				ECAP11	I	增强型捕获输入 11
				ETIMER11	O	通用定时器输出 11
-	-	-	112	GPIO85	I/O	通用输入/输出 85
				EQEP0_B	I	EQEP0 输入 B
				ECAP12	I	增强型捕获输入 12
				ETIMER12	O	通用定时器输出 12
-	-	-	113	GPIO71	I/O	通用输入/输出 71
				EQEP0_INDEX	I	EQEP0 索引
				ECAP13	I	增强型捕获输入 13
				ETIMER13	O	通用定时器输出 13

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
61	77	98	114	GPIO2	I/O	通用输入/输出 2
				SRPWM1_A	O	SRPWM1A 输出
				CANFD1_RX	I	CAN FD1 接收
				OUTXBAR0	O	输出 X-BAR 输出 0
				SPIM0_MOSI	O	SPIM0 主器件输出，从器件输入
				UART0_TX	O	UART0 输出
				I2C1_SDA	I/OD	I2C1 开漏双向数据
				CANFD0_TX	O	CAN FD0 发送
62	78	99	115	GPIO1	I/O	通用输入/输出 1
				SRPWM0_B	O	SRPWM0B 输出
				SPIS0_CLK	I	SPIS0 时钟
				SPIM0_CLK	O	SPIM0 时钟
				I2C0_SCL	I/OD	I2C0 开漏双向时钟
				SPIM0_MISO	I	SPIM0_主器件输入，从器件输出备份
				CANFD1_TX	O	CAN FD1 发送
63	79	100	116	GPIO0	I/O	通用输入/输出 0
				SRPWM0_A	O	SRPWM0A 输出
				SPIS0_MOSI	I	SPIS0 主器件输出，从器件输入
				SPIM0_MOSI	O	SPIM0 主器件输出，从器件输入
				I2C0_SDA	I/OD	I2C0 开漏双向数据
				SPIM0_CS0_n	O	SPIM0 从器件片选 0（低有效）
				CANFD1_RX	I	CAN FD1 接收
				EQEP0_INDEX	I	EQEP0 索引

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
64	80	101	117	GPIO40	I/O	通用输入/输出 40
				SPIM1_MOSI	O	SPIM1 主器件输出，从器件输入
				SPIS0_MISO	O	SPIS0_主器件输入，从器件输出备份
				SPIM0_MISO	I	SPIM0_主器件输入，从器件输出备份
				SRPWM1_B	O	SRPWM1B 输出
				I2C0_SDA	I/OD	I2C0 开漏双向数据
				UART1_TX	O	UART1 输出
				EQEP0_A	I	EQEP0 输入 A
65	81	102	118	GPIO23	I/O	通用输入/输出 23
				EQEP0_INDEX	I	EQEP0 索引
				SPIS0_CS0_n	O	SPIS0 从器件片选 0（低有效）
				UART1_RX	I	UART1 接收
				SPIM0_CS0_n	O	SPIM0 从器件片选 0（低有效）
				SPIM1_CS0_n	O	SPIM1 从器件片选 0（低有效）
				SDFM0_CLK4	I	SDFM0 通道 4 时钟输入
				SRPWM3_B	O	SRPWM3B 输出
66	82	103	119	GPIO41	I/O	通用输入/输出 41
				SPIM2_CLK	O	SPIM2 时钟
				SRPWM1_A	O	SRPWM1A 输出
				I2C0_SCL	I/OD	I2C0 开漏双向时钟
				UART1_RX	I	UART1 接收
				EQEP0_B	I	EQEP0 输入 B

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				SPIM1_MISO	I	SPIM1_主器件输入，从器件输出备份
67	83	104	120	GPIO22	I/O	通用输入/输出 22
				EQEP0_STROBE	I	EQEP0 选通
				SPIM1_CS3_n	O	SPIM1 从器件片选 3（低有效）
				UART1_TX	O	UART1 输出
				SPIM2_MOSI	O	SPIM2 主器件输出，从器件输入
				SPIM1_CLK	O	SPIM1 时钟
				SDFM0_DATA4	I	SDFM0 通道 4 数据输入
				SRPWM3_A	O	SRPWM3A 输出
68	84	105	121	GPIO7	I/O	通用输入/输出 7
				SRPWM3_B	O	SRPWM3B 输出
				SPIM0_CS2_n	O	SPIM0 从器件片选 2（低有效）
				OUTXBAR4	O	输出 X-BAR 输出 4
				SPIM2_MISO	I	SPIM2_主器件输入，从器件输出备份
				EQEP0_B	I	EQEP0 输入 B
				SPIM1_MOSI	O	SPIM1 主器件输出，从器件输入
69	85	106	122	GPIO44	I/O	通用输入/输出 44
				SPIM0_CS3_n	O	SPIM0 从器件片选 3（低有效）
				OUTXBAR6	O	输出 X-BAR 输出 6
				SPIM2_CS0_n	O	SPIM2 从器件片选 0（低有效）
				EQEP0_A	I	EQEP0 输入 A
				I2C0_SDA	I/OD	I2C0 开漏双向数据

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				I2C0_CTL_n	I/OD	I2C0 控制信号-从器件输入/主器件输出（低有效）
73	-	110	126	GPIO45	I/O	通用输入/输出 45
				I2C1_ALERT_n	I/OD	I2C1 控制信号-从器件输入/主器件输出（低有效）
				OUTXBAR9	O	输出 X-BAR 输出 9
				ECAP9	I	增强型捕获输入 9
				I2C0_ALERT_n	I/OD	I2C0 控制信号-从器件输入/主器件输出（低有效）
				SDFM1_CLK3	I	SDFM1 通道 3 时钟输入
74	89	111	127	GPIO5	I/O	通用输入/输出 5
				SRPWM2_B	O	SRPWM2B 输出
				OUTXBAR2	O	输出 X-BAR 输出 2
				SRPWM0_A	O	SRPWM0A 输出
				CANFD1_RX	I	CAN FD1 接收
				CANFD0_RX	I	CAN FD0 接收
				SPIM0_CS0_n	O	SPIM0 从器件片选 0（低有效）
75	90	112	128	GPIO9	I/O	通用输入/输出 9
				SRPWM4_B	O	SRPWM4B 输出
				UART1_TX	O	UART1 输出
				OUTXBAR5	O	输出 X-BAR 输出 5
				SRPWM0_B	O	SRPWM0B 输出
				EQEP0_INDEX	I	EQEP0 索引
				UART0_RX	I	UART0 接收

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				SPIM0_CLK	O	SPIM0 时钟
				I2C1_SCL	I/OD	I2C1 开漏双向时钟
-	91	113	129	GPIO61	I/O	通用输入/输出 61
				CANFD1_RX	I	CAN FD1 接收
				SRPWM1_A	O	SRPWM1A 输出
				OUTXBAR3	O	输出 X-BAR 输出 3
				SPIM1_MISO	I	SPIM1_主器件输入，从器件输出备份
				SDFM1_CLK3	I	SDFM1 通道 3 时钟输入
				CANFD0_RX	I	CAN FD0 接收
-	92	114	130	GPIO59	I/O	通用输入/输出 59
				SRPWM1_B	O	SRPWM1B 输出
				OUTXBAR1	O	输出 X-BAR 输出 1
				SPIM1_CS0_n	O	SPIM1 从器件片选 0（低有效）
				SDFM1_CLK2	I	SDFM1 通道 2 时钟输入
				CANFD0_RX	I	CAN FD0 接收
				EQEP0_INDEX	I	EQEP0 索引
				SDFM1_CLK3	I	SDFM1 通道 3 时钟输入
76	93	115	131	GPIO10	I/O	通用输入/输出 10
				SRPWM5_A	O	SRPWM5A 输出
				SRPWM2_A	O	SRPWM2A 输出
				EQEP0_A	I	EQEP0 输入 A
				UART1_TX	O	UART1 输出
				SPIM0_MISO	I	SPIM0_主器件输入，从器件输出备

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
						份
				I2C0_SDA	I/OD	I2C0 开漏双向数据
77	94	116	132	GPIO34	I/O	通用输入/输出 34
				OUTXBAR0	O	输出 X-BAR 输出 0
				SRPWM2_B	O	SRPWM2B 输出
				I2C0_SDA	I/OD	I2C0 开漏双向数据
				I2C1_SDA	I/OD	I2C1 开漏双向数据
78	95	117	133	GPIO15	I/O	通用输入/输出 15
				SRPWM7_B	O	SRPWM7B 输出
				UART1_RX	I	UART1 接收
				SRPWM3_A	O	SRPWM3A 输出
				I2C1_SCL	I/OD	I2C1 开漏双向时钟
				OUTXBAR3	O	输出 X-BAR 输出 3
				SPIM1_CS0_n	O	SPIM1 从器件片选 0（低有效）
				EQEP1_B	I	EQEP1 输入 B
				SRPWM2_B	O	SRPWM2B 输出
79	96	118	134	GPIO14	I/O	通用输入/输出 14
				SRPWM7_A	O	SRPWM7A 输出
				UART1_TX	O	UART1 输出
				SRPWM3_B	O	SRPWM3B 输出
				I2C1_SDA	I/OD	I2C1 开漏双向数据
				OUTXBAR2	O	输出 X-BAR 输出 2
				SPIM1_CLK	O	SPIM1 时钟
				EQEP1_A	I	EQEP1 输入 A

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				SRPWM2_A	O	SRPWM2A 输出
80	97	119	135	GPIO6	I/O	通用输入/输出 6
				SRPWM3_A	O	SRPWM3A 输出
				OUTXBAR3	O	输出 X-BAR 输出 3
				SYNCOUT	O	ETIMER 同步信号输出
				SRPWM4_A	O	SRPWM4A 输出
				EQEP0_A	I	EQEP0 输入 A
				SPIM1_MISO	I	SPIM1_主器件输入，从器件输出备份
-	-	120	136	GPIO72	I/O	通用输入/输出 72
				SRPWM6_A	O	SRPWM6A 输出
-	-	121	137	GPIO73	I/O	通用输入/输出 73
				SRPWM7_A	O	SRPWM7A 输出
-	-	122	138	GPIO74	I/O	通用输入/输出 74
				SRPWM6_B	O	SRPWM6B 输出
-	-	123	139	GPIO75	I/O	通用输入/输出 75
				SRPWM7_B	O	SRPWM7B 输出
-	-	124	140	GPIO76	I/O	通用输入/输出 76
				SPIM0_MOSI	O	SPIM0 主器件输出，从器件输入
				CANFD0_TX	O	CAN FD0 发送
1	98	125	141	GPIO30	I/O	通用输入/输出 30
				CANFD0_RX	I	CAN FD0 接收
				SPIM1_MOSI	O	SPIM1 主器件输出，从器件输入
				SRPWM4_B	O	SRPWM4B 输出

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				OUTXBAR6	O	输出 X-BAR 输出 6
				EQEP0_STROBE	I	EQEP0 选通
				SDFM1_DATA4	I	SDFM1 通道 4 数据输入
				CANFD1_RX	I	CAN FD1 接收
				SRPWM0_A	O	SRPWM0A 输出
-	-	126	142	GPIO77	I/O	通用输入/输出 77
				SPIM0_MISO	I	SPIM0_主器件输入，从器件输出备份
				CANFD0_RX	I	CAN FD0 接收
2	99	127	143	GPIO31	I/O	通用输入/输出 31
				CANFD0_TX	O	CAN FD0 发送
				SPIM1_MISO	I	SPIM1_主器件输入，从器件输出备份
				SRPWM5_A	O	SRPWM5A 输出
				OUTXBAR7	O	输出 X-BAR 输出 7
				EQEP0_INDEX	I	EQEP0 索引
				SDFM1_CLK4	I	SDFM1 通道 4 时钟输入
				CANFD1_TX	O	CAN FD1 发送
				SRPWM0_B	O	SRPWM0B 输出
3	100	128	144	GPIO29	I/O	通用输入/输出 29
				UART0_TX	O	UART0 输出
				UART1_RX	I	UART1 接收
				SRPWM6_B	O	SRPWM6B 输出
				SRPWM5_B	O	SRPWM5B 输出

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
				OUTXBAR5	O	输出 X-BAR 输出 5
				EQEP0_B	I	EQEP0 输入 B
				SDFM1_CLK3	I	SDFM1 通道 3 时钟输入
				EQEP1_INDEX	I	EQEP1 索引
				SPIM1_CS0_n	O	SPIM1 从器件片选 0（低有效）
				I2C1_SCL	I/OD	I2C1 开漏双向时钟
时钟、JTAG 和复位						
50	68	87	96	XTAL_OUT	O	晶体振荡器输出
51	69	88	97	XTAL_IN	I	晶体振荡器或单端时钟输入，为了使用此振荡器，必须将一个石英晶体电路连接至 XTAL_IN 和 XTAL_OUT，并且预留补偿电容。此引脚也可用于馈入单端 3.3 V 电平时钟。
5	2	2	2	HWRST_n	I	SoC 外部复位信号（低有效）
45	60	72	81	SWCLK	I	SWD 时钟输入
47	62	74	83	SWDIO	I/O	SWD 数据信号输入输出
电源和接地						
-	73	92	101	REGUENZ	I	VDD 外部电源选择，低电平使用内部电源，高电平使用外部电源
8、31、53、71	4、46、71、87	5、57、90、108	5、65、99、124	VDD	P	1.1 V 数字逻辑电源引脚。建议在每个 VDD 引脚附近放置一个去耦电容，其最小总电容约为 20 µF。当不使用内部电压调整器时，去耦电容的确切值应由系统电压调节方案确

LQFP80	LQFP100	LQFP128	LQFP144	信号名称	引脚类型	说明
						定。
26	34	43	48	VDDA	P	3.3 V 模拟电源引脚。在每个引脚上放置一个最小 2.2 μF 的去耦电容到 VSSA。
7、32、52、72	3、47、70、88	4、58、89、109	4、66、98、125	VDDIO	P	3.3 V 数字 I/O 电源引脚。在每个引脚上放置一个最小 0.1 μF 的去耦电容。
9、30、55、70	5、45、72、86	6、56、91、107	6、64、100、123	VSS	G	数字地
25	33	42	47	VSSA	G	模拟地
-	-	-	72	NC	-	NC(1)，悬空处理
-	-	-	108	NC	-	NC(1)，悬空处理

说明

(1) 上电时必须保持上拉，阻值推荐使用 4.7 kΩ 或 10 kΩ，待 POR 解复位锁存 GPIO 状态后，可以设置为输入或输出模式。

(2) 默认状态配置详见芯片启动模式章节，阻值推荐使用 4.7 kΩ 或 10 kΩ，待 POR 解复位锁存 GPIO 状态后，可以设置为输入或输出模式。

(3) 表中 NC 代表 Not Connected，是空脚。

3.3 信号描述

3.3.1 模拟信号

表3-2 模拟信号

信号名称	引脚类型	说明	LQFP144	LQFP128	LQFP100	LQFP80
SAR_A0	I	ADC-A 输入 0	33	31	23	19
SAR_A1	I	ADC-A 输入 1	30	30	22	18
SAR_A2	I	ADC-A 输入 2	23	23	17	13
SAR_A3	I	ADC-A 输入 3	25	25	18	-
SAR_A3	I	ADC-A 输入 3	22	22	-	12
SAR_A4	I	ADC-A 输入 4	51	46	36	-
SAR_A4	I	ADC-A 输入 4	49	44	-	27
SAR_A5	I	ADC-A 输入 5	50	45	35	-
SAR_A5	I	ADC-A 输入 5	28	28	-	17
SAR_A6	I	ADC-A 输入 6	19	19	14	10
SAR_A7	I	ADC-A 输入 7	44	39	31	23
SAR_A8	I	ADC-A 输入 8	52	47	37	-
SAR_A8	I	ADC-A 输入 8	45	40	-	24
SAR_A9	I	ADC-A 输入 9	53	48	38	28
SAR_A10	I	ADC-A 输入 10	55	50	40	29
SAR_A11	I	ADC-A 输入 11	27	27	20	16
SAR_A12	I	ADC-A 输入 12	39	36	28	22
SAR_A14	I	ADC-A 输入 14	26	26	19	15

信号名称	引脚类型	说明	LQFP144	LQFP128	LQFP100	LQFP80
SAR_A15	I	ADC-A 输入 15	24	24	-	14
SAR_B0	I	ADC-B 输入 0	56	51	41	-
SAR_B0	I	ADC-B 输入 0	45	40	-	24
SAR_B1	I	ADC-B 输入 1	55	50	40	29
SAR_B1	I	ADC-B 输入 1	34	-	-	-
SAR_B2	I	ADC-B 输入 2	20	20	15	11
SAR_B3	I	ADC-B 输入 3	21	21	16	12
SAR_B4	I	ADC-B 输入 4	54	49	39	28
SAR_B5	I	ADC-B 输入 5	46	41	32	-
SAR_B6	I	ADC-B 输入 6	23	23	17	13
SAR_B7	I	ADC-B 输入 7	30	30	22	18
SAR_B8	I	ADC-B 输入 8	51	46	36	-
SAR_B8	I	ADC-B 输入 8	49	44	-	27
SAR_B9	I	ADC-B 输入 9	25	25	18	-
SAR_B9	I	ADC-B 输入 9	24	24	-	14
SAR_B10	I	ADC-B 输入 10	27	27	20	16
SAR_B11	I	ADC-B 输入 11	43	38	30	-
SAR_B12	I	ADC-B 输入 12	29	29	21	17
SAR_B14	I	ADC-B 输入 14	26	26	19	15
SAR_B15	I	ADC-B 输入 15	33	31	23	19
SAR_C0	I	ADC-C 输入 0	27	27	20	16
SAR_C0	I	ADC-C 输入 0	42	-	-	-
SAR_C1	I	ADC-C 输入 1	40	37	29	22
SAR_C2	I	ADC-C 输入 2	29	29	21	17

信号名称	引脚类型	说明	LQFP144	LQFP128	LQFP100	LQFP80
SAR_C3	I	ADC-C 输入 3	44	39	31	23
SAR_C3	I	ADC-C 输入 3	31	-	-	-
SAR_C4	I	ADC-C 输入 4	26	26	19	15
SAR_C5	I	ADC-C 输入 5	22	22	-	12
SAR_C6	I	ADC-C 输入 6	20	20	15	11
SAR_C6	I	ADC-C 输入 6	32	-	-	-
SAR_C7	I	ADC-C 输入 7	25	25	18	-
SAR_C7	I	ADC-C 输入 7	24	24	-	14
SAR_C7	I	ADC-C 输入 7	41	-	-	-
SAR_C8	I	ADC-C 输入 8	54	49	39	28
SAR_C8	I	ADC-C 输入 8	58	-	-	-
SAR_C9	I	ADC-C 输入 9	23	23	17	13
SAR_C10	I	ADC-C 输入 10	55	50	40	29
SAR_C10	I	ADC-C 输入 10	59	-	-	-
SAR_C11	I	ADC-C 输入 11	56	51	41	-
SAR_C11	I	ADC-C 输入 11	45	40	-	24
SAR_C11	I	ADC-C 输入 11	60	-	-	-
SAR_C14	I	ADC-C 输入 14	49	44	-	27
SAR_C14	I	ADC-C 输入 14	57	52	42	-
SAR_C15	I	ADC-C 输入 15	33	31	23	19
CMP1_HN0	I	CMP1 高电平比较器负输入 0	24	24	-	14
CMP1_HN1	I	CMP1 高电平比较器负输入 1	27	27	20	16
CMP1_HP0	I	CMP1 高电平比较器正输入 0	23	23	17	13
CMP1_HP1	I	CMP1 高电平比较器正输入 1	27	27	20	16

信号名称	引脚类型	说明	LQFP144	LQFP128	LQFP100	LQFP80
CMP1_HP2	I	CMP1 高电平比较器正输入 2	19	19	14	10
CMP1_HP3	I	CMP1 高电平比较器正输入 3	24	24	-	14
CMP1_HP4	I	CMP1 高电平比较器正输入 4	30	30	22	18
CMP1_HP5	I	CMP1 高电平比较器正输入 5	46	41	32	-
CMP1_LN0	I	CMP1 低电平比较器负输入 0	24	24	-	14
CMP1_LN1	I	CMP1 低电平比较器负输入 1	27	27	20	16
CMP1_LP0	I	CMP1 低电平比较器正输入 0	23	23	17	13
CMP1_LP1	I	CMP1 低电平比较器正输入 1	27	27	20	16
CMP1_LP2	I	CMP1 低电平比较器正输入 2	19	19	14	10
CMP1_LP3	I	CMP1 低电平比较器正输入 3	24	24	-	14
CMP1_LP4	I	CMP1 低电平比较器正输入 4	30	30	22	18
CMP1_LP5	I	CMP1 低电平比较器正输入 5	46	41	32	-
CMP2_HN0	I	CMP2 高电平比较器负输入 0	55	50	40	29
CMP2_HN1	I	CMP2 高电平比较器负输入 1	39	36	28	22
CMP2_HP0	I	CMP2 高电平比较器正输入 0	51	46	36	-
CMP2_HP0	I	CMP2 高电平比较器正输入 0	49	44	-	27
CMP2_HP1	I	CMP2 高电平比较器正输入 1	39	36	28	22
CMP2_HP2	I	CMP2 高电平比较器正输入 2	53	48	38	28
CMP2_HP3	I	CMP2 高电平比较器正输入 3	55	50	40	29
CMP2_HP4	I	CMP2 高电平比较器正输入 4	56	51	41	-
CMP2_HP4	I	CMP2 高电平比较器正输入 4	45	40	-	24
CMP2_HP5	I	CMP2 高电平比较器正输入 5	50	45	35	-
CMP2_LN0	I	CMP2 低电平比较器负输入 0	55	50	40	29
CMP2_LN1	I	CMP2 低电平比较器负输入 1	39	36	28	22

信号名称	引脚类型	说明	LQFP144	LQFP128	LQFP100	LQFP80
CMP2_LP0	I	CMP2 低电平比较器正输入 0	51	46	36	-
CMP2_LP0	I	CMP2 低电平比较器正输入 0	49	44	-	27
CMP2_LP1	I	CMP2 低电平比较器正输入 1	39	36	28	22
CMP2_LP2	I	CMP2 低电平比较器正输入 2	53	48	38	28
CMP2_LP3	I	CMP2 低电平比较器正输入 3	55	50	40	29
CMP2_LP4	I	CMP2 低电平比较器正输入 4	56	51	41	-
CMP2_LP4	I	CMP2 低电平比较器正输入 4	45	40	-	24
CMP2_LP5	I	CMP2 低电平比较器正输入 5	50	45	35	-
CMP3_HN0	I	CMP3 高电平比较器负输入 0	21	21	16	12
CMP3_HN1	I	CMP3 高电平比较器负输入 1	29	29	21	17
CMP3_HP0	I	CMP3 高电平比较器正输入 0	20	20	15	11
CMP3_HP1	I	CMP3 高电平比较器正输入 1	29	29	21	17
CMP3_HP2	I	CMP3 高电平比较器正输入 2	33	31	23	19
CMP3_HP3	I	CMP3 高电平比较器正输入 3	21	21	16	12
CMP3_HP4	I	CMP3 高电平比较器正输入 4	26	26	19	15
CMP3_HP5	I	CMP3 高电平比较器正输入 5	25	25	18	-
CMP3_LN0	I	CMP3 低电平比较器负输入 0	21	21	16	12
CMP3_LN1	I	CMP3 低电平比较器负输入 1	29	29	21	17
CMP3_LP0	I	CMP3 低电平比较器正输入 0	20	20	15	11
CMP3_LP1	I	CMP3 低电平比较器正输入 1	29	29	21	17
CMP3_LP2	I	CMP3 低电平比较器正输入 2	33	31	23	19
CMP3_LP3	I	CMP3 低电平比较器正输入 3	21	21	16	12
CMP3_LP4	I	CMP3 低电平比较器正输入 4	26	26	19	15
CMP3_LP5	I	CMP3 低电平比较器正输入 5	25	25	18	-

信号名称	引脚类型	说明	LQFP144	LQFP128	LQFP100	LQFP80
CMP4_HN0	I	CMP4 高电平比较器负输入 0	49	44	-	27
CMP4_HN0	I	CMP4 高电平比较器负输入 0	57	52	42	-
CMP4_HN1	I	CMP4 高电平比较器负输入 1	44	39	31	23
CMP4_HP0	I	CMP4 高电平比较器正输入 0	54	49	39	28
CMP4_HP1	I	CMP4 高电平比较器正输入 1	44	39	31	23
CMP4_HP2	I	CMP4 高电平比较器正输入 2	40	37	29	22
CMP4_HP3	I	CMP4 高电平比较器正输入 3	49	44	-	27
CMP4_HP3	I	CMP4 高电平比较器正输入 3	57	52	42	-
CMP4_HP4	I	CMP4 高电平比较器正输入 4	45	40	-	24
CMP4_HP4	I	CMP4 高电平比较器正输入 4	52	47	37	-
CMP4_HP5	I	CMP4 高电平比较器正输入 5	43	38	30	-
CMP4_LN0	I	CMP4 低电平比较器负输入 0	49	44	-	27
CMP4_LN0	I	CMP4 低电平比较器负输入 0	57	52	42	-
CMP4_LN1	I	CMP4 低电平比较器负输入 1	44	39	31	23
CMP4_LP0	I	CMP4 低电平比较器正输入 0	54	49	39	28
CMP4_LP1	I	CMP4 低电平比较器正输入 1	44	39	31	23
CMP4_LP2	I	CMP4 低电平比较器正输入 2	40	37	29	22
CMP4_LP3	I	CMP4 低电平比较器正输入 3	49	44	-	27
CMP4_LP3	I	CMP4 低电平比较器正输入 3	57	52	42	-
CMP4_LP4	I	CMP4 低电平比较器正输入 4	45	40	-	24
CMP4_LP4	I	CMP4 低电平比较器正输入 4	52	47	37	-
CMP4_LP5	I	CMP4 低电平比较器正输入 5	43	38	30	-
DACA_OUT	I	缓冲 DAC-A 输出	33	31	23	19
DACB_OUT	I	缓冲 DAC-B 输出	30	30	22	18

信号名称	引脚类型	说明	LQFP144	LQFP128	LQFP100	LQFP80
VDAC	I	片上 DAC 的可选外部基准电压	21	21	16	12
VREFHI_A	I	ADC-A 高基准电压。在外部基准模式下，从外部驱动这个引脚上的高基准电压。在内部基准模式下，电压由器件驱动到该引脚。在任一模式下，在此引脚上放置至少一个 2.2 µF 电容器。此电容器应放置 VREFHI_A 和 VREFLO_A 引脚之间尽可能靠近器件的位置。	35	32	24	20
VREFHI_BC	I	ADC-B/C 高基准电压。在外部基准模式下，从外部驱动这个引脚上的高基准电压。在内部基准模式下，电压由器件驱动到该引脚。在任一模式下，在此引脚上放置至少一个 2.2 µF 电容器。此电容器应放置 VREFHI_BC 和 VREFLO_BC 引脚之间尽可能靠近器件的位置。	36	33	25	20
VREFLO_A	I	ADC-A 低基准电压	37	34	26	21
VREFLO_BC	I	ADC-B/C 低基准电压	38	35	27	21

3.3.2 数字信号

表3-3 数字信号

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
CANFD0_RX	I	CAN FD0 接收	GPIO49,GPIO53,GPIO12,GPIO	36,38,48,59,60,7	8,12,51,53,63,75,	9,14,62,64,75,9	9,14,70,73,84,

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
			33,GPIO35,GPIO4,GPIO3,GPIO5,GPIO61,GPIO59,GPIO30,GPIO77	4,1	76,89,91,92,98	6,97,111,113,114,125,126	105,107,127,129,130,141,142
CANFD0_TX	O	CAN FD0 发送	GPIO48,GPIO13,GPIO17,GPIO37,GPIO32,GPIO58,GPIO8,GPIO4,GPIO2,GPIO76,GPIO31	35,40,46,49,58,59,61,2	7,50,55,61,64,67,74,75,77,99	8,61,67,73,82,86,95,96,98,124,127	8,69,76,82,91,95,104,105,114,140,143
CANFD1_RX	I	CAN FD1 接收	GPIO47,GPIO51,GPIO78,GPIO21,GPIO12,GPIO57,GPIO2,GPIO0,GPIO5,GPIO61,GPIO30	34,36,61,63,74,1	6,10,49,51,66,77,79,89,91,98	7,11,17,60,62,85,98,100,111,113,125	7,11,17,68,70,94,114,116,127,129,141
CANFD1_TX	O	CAN FD1 发送	GPIO46,GPIO50,GPIO67,GPIO60,GPIO20,GPIO13,GPIO56,GPIO4,GPIO3,GPIO1,GPIO31	6,33,35,59,60,62,2	9,44,48,50,65,75,76,78,99	3,10,16,54,59,61,84,96,97,99,127	3,10,16,62,67,69,93,105,107,115,143
ECAP0	I	增强型捕获输入 0	GPIO37	46	61	73	82
ECAP1	I	增强型捕获输入 1	GPIO35	48	63	75	84
ECAP10	I	增强型捕获输入 10	GPIO20,GPIO83	33	48	59	67,110
ECAP11	I	增强型捕获输入 11	GPIO21,GPIO84	34	49	60	68,111
ECAP12	I	增强型捕获输入 12	GPIO13,GPIO85	35	50	61	69,112
ECAP13	I	增强型捕获输入 13	GPIO12,GPIO71	36	51	62	70,113
ECAP2	I	增强型捕获输	GPIO43	54		83	92

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
		入 2					
ECAP3	I	增强型捕获输入 3	GPIO56		65	84	93
ECAP4	I	增强型捕获输入 4	GPIO57		66	85	94
ECAP5	I	增强型捕获输入 5	GPIO58		67	86	95
ECAP6	I	增强型捕获输入 6	GPIO39	56		93	102
ECAP7	I	增强型捕获输入 7	GPIO82				106
ECAP8	I	增强型捕获输入 8	GPIO81				109
ECAP9	I	增强型捕获输入 9	GPIO45	73		110	126
EQEP0_A	I	EQEP0 输入 A	GPIO28,GPIO50,GPIO20,GPIO25,GPIO35,GPIO56,GPIO42,GPIO84,GPIO40,GPIO44,GPIO10,GPIO6	4,33,42,48,57,64,69,76,80	1,9,48,57,63,65,80,85,93,97	1,10,59,69,75,84,94,101,106,115,119	1,10,67,78,84,93,103,111,117,122,131,135
EQEP0_B	I	EQEP0 输入 B	GPIO51,GPIO21,GPIO11,GPIO37,GPIO65,GPIO57,GPIO85,GPIO41,GPIO7,GPIO29	34,37,46,66,68,3	10,49,52,61,66,82,84,100	11,60,63,73,76,85,103,105,128	11,68,71,82,85,94,112,119,121,144
EQEP0_INDEX	I	EQEP0 索引	GPIO53,GPIO13,GPIO17,GPIO62,GPIO43,GPIO39,GPIO71,GPIO0,GPIO23,GPIO9,GPIO59,	35,40,54,56,63,65,75,2	12,50,55,79,81,90,92,99	14,61,67,77,83,93,100,102,112,114,127	14,69,76,86,92,102,113,116,118,128,130,14

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
			GPIO31				3
EQEP0_STROBE	I	EQEP0 选通	GPIO52,GPIO12,GPIO16,GPIO58,GPIO8,GPIO82,GPIO83,GPIO22,GPIO30	36,39,58,67,1	11,51,54,67,74,83,98	13,62,65,86,95,104,125	13,70,74,95,104,106,110,120,141
EQEP1_A	I	EQEP1 输入 A	GPIO54,GPIO11,GPIO24,GPIO14	37,41,79	13,52,56,96	15,63,68,118	15,71,77,134
EQEP1_B	I	EQEP1 输入 B	GPIO55,GPIO33,GPIO16,GPIO25,GPIO15	38,39,42,78	43,53,54,57,95	53,64,65,69,117	61,73,74,78,133
EQEP1_INDEX	I	EQEP1 索引	GPIO26,GPIO57,GPIO29	43,3	58,66,100	70,85,128	79,94,144
EQEP1_STROBE	I	EQEP1 选通	GPIO28,GPIO27,GPIO56,GPIO4	4,44,59	1,59,65,75	1,71,84,96	1,80,93,105
ETIMER0	O	通用定时器输出 0	GPIO11	37	52	63	71
ETIMER1	O	通用定时器输出 1	GPIO33	38	53	64	73
ETIMER10	O	通用定时器输出 10	GPIO79,GPIO83			18	18,110
ETIMER11	O	通用定时器输出 11	GPIO68,GPIO84			79	88,111
ETIMER12	O	通用定时器输出 12	GPIO69,GPIO85			80	89,112
ETIMER13	O	通用定时器输出 13	GPIO70,GPIO71			81	90,113
ETIMER2	O	通用定时器输出 2	GPIO16	39	54	65	74

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
ETIMER3	O	通用定时器输出 3	GPIO25	42	57	69	78
ETIMER4	O	通用定时器输出 4	GPIO26	43	58	70	79
ETIMER5	O	通用定时器输出 5	GPIO27	44	59	71	80
ETIMER6	O	通用定时器输出 6	GPIO17	40	55	67	76
ETIMER7	O	通用定时器输出 7	GPIO42	57		94	103
ETIMER8	O	通用定时器输出 8	GPIO65			76	85
ETIMER9	O	通用定时器输出 9	GPIO62			77	86
GPIO0	I/O	通用输入/输出 0	GPIO0	63	79	100	116
GPIO1	I/O	通用输入/输出 1	GPIO1	62	78	99	115
GPIO2	I/O	通用输入/输出 2	GPIO2	61	77	98	114
GPIO3	I/O	通用输入/输出 3	GPIO3	60	76	97	107
GPIO4	I/O	通用输入/输出 4	GPIO4	59	75	96	105
GPIO5	I/O	通用输入/输出	GPIO5	74	89	111	127

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
		5					
GPIO6	I/O	通用输入/输出 6	GPIO6	80	97	119	135
GPIO7	I/O	通用输入/输出 7	GPIO7	68	84	105	121
GPIO8	I/O	通用输入/输出 8	GPIO8	58	74	95	104
GPIO9	I/O	通用输入/输出 9	GPIO9	75	90	112	128
GPIO10	I/O	通用输入/输出 10	GPIO10	76	93	115	131
GPIO11	I/O	通用输入/输出 11	GPIO11	37	52	63	71
GPIO12	I/O	通用输入/输出 12	GPIO12	36	51	62	70
GPIO13	I/O	通用输入/输出 13	GPIO13	35	50	61	69
GPIO14	I/O	通用输入/输出 14	GPIO14	79	96	118	134
GPIO15	I/O	通用输入/输出 15	GPIO15	78	95	117	133
GPIO16	I/O	通用输入/输出 16	GPIO16	39	54	65	74
GPIO17	I/O	通用输入/输出 17	GPIO17	40	55	67	76

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
GPIO18	I/O	通用输入/输出 18	SWCLK	45	60	72	81
GPIO19	I/O	通用输入/输出 19	SWDIO	47	62	74	83
GPIO20	I/O	通用输入/输出 20	GPIO20	33	48	59	67
GPIO21	I/O	通用输入/输出 21	GPIO21	34	49	60	68
GPIO22	I/O	通用输入/输出 22	GPIO22	67	83	104	120
GPIO23	I/O	通用输入/输出 23	GPIO23	65	81	102	118
GPIO24	I/O	通用输入/输出 24	GPIO24	41	56	68	77
GPIO25	I/O	通用输入/输出 25	GPIO25	42	57	69	78
GPIO26	I/O	通用输入/输出 26	GPIO26	43	58	70	79
GPIO27	I/O	通用输入/输出 27	GPIO27	44	59	71	80
GPIO28	I/O	通用输入/输出 28	GPIO28	4	1	1	1
GPIO29	I/O	通用输入/输出 29	GPIO29	3	100	128	144
GPIO30	I/O	通用输入/输出	GPIO30	1	98	125	141

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
		30					
GPIO31	I/O	通用输入/输出 31	GPIO31	2	99	127	143
GPIO32	I/O	通用输入/输出 32	GPIO32	49	64	82	91
GPIO33	I/O	通用输入/输出 33	GPIO33	38	53	64	73
GPIO34	I/O	通用输入/输出 34	GPIO34	77	94	116	132
GPIO35	I/O	通用输入/输出 35	GPIO35	48	63	75	84
GPIO37	I/O	通用输入/输出 37	GPIO37	46	61	73	82
GPIO39	I/O	通用输入/输出 39	GPIO39	56		93	102
GPIO40	I/O	通用输入/输出 40	GPIO40	64	80	101	117
GPIO41	I/O	通用输入/输出 41	GPIO41	66	82	103	119
GPIO42	I/O	通用输入/输出 42	GPIO42	57		94	103
GPIO43	I/O	通用输入/输出 43	GPIO43	54		83	92
GPIO44	I/O	通用输入/输出 44	GPIO44	69	85	106	122

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
GPIO45	I/O	通用输入/输出 45	GPIO45	73		110	126
GPIO46	I/O	通用输入/输出 46	GPIO46	6		3	3
GPIO47	I/O	通用输入/输出 47	GPIO47		6	7	7
GPIO48	I/O	通用输入/输出 48	GPIO48		7	8	8
GPIO49	I/O	通用输入/输出 49	GPIO49		8	9	9
GPIO50	I/O	通用输入/输出 50	GPIO50		9	10	10
GPIO51	I/O	通用输入/输出 51	GPIO51		10	11	11
GPIO52	I/O	通用输入/输出 52	GPIO52		11	13	13
GPIO53	I/O	通用输入/输出 53	GPIO53		12	14	14
GPIO54	I/O	通用输入/输出 54	GPIO54		13	15	15
GPIO55	I/O	通用输入/输出 55	GPIO55		43	53	61
GPIO56	I/O	通用输入/输出 56	GPIO56		65	84	93
GPIO57	I/O	通用输入/输出	GPIO57		66	85	94

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
		57					
GPIO58	I/O	通用输入/输出 58	GPIO58		67	86	95
GPIO59	I/O	通用输入/输出 59	GPIO59		92	114	130
GPIO60	I/O	通用输入/输出 60	GPIO60		44	54	62
GPIO61	I/O	通用输入/输出 61	GPIO61		91	113	129
GPIO62	I/O	通用输入/输出 62	GPIO62			77	86
GPIO63	I/O	通用输入/输出 63	GPIO63			78	87
GPIO64	I/O	通用输入/输出 64	GPIO64			66	75
GPIO65	I/O	通用输入/输出 65	GPIO65			76	85
GPIO66	I/O	通用输入/输出 66	GPIO66			12	12
GPIO67	I/O	通用输入/输出 67	GPIO67			16	16
GPIO68	I/O	通用输入/输出 68	GPIO68			79	88
GPIO69	I/O	通用输入/输出 69	GPIO69			80	89

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
GPIO70	I/O	通用输入/输出 70	GPIO70			81	90
GPIO71	I/O	通用输入/输出 71	GPIO71				113
GPIO72	I/O	通用输入/输出 72	GPIO72			120	136
GPIO73	I/O	通用输入/输出 73	GPIO73			121	137
GPIO74	I/O	通用输入/输出 74	GPIO74			122	138
GPIO75	I/O	通用输入/输出 75	GPIO75			123	139
GPIO76	I/O	通用输入/输出 76	GPIO76			124	140
GPIO77	I/O	通用输入/输出 77	GPIO77			126	142
GPIO78	I/O	通用输入/输出 78	GPIO78			17	17
GPIO79	I/O	通用输入/输出 79	GPIO79			18	18
GPIO80	I/O	通用输入/输出 80	GPIO80			55	63
GPIO81	I/O	通用输入/输出 81	GPIO81				109
GPIO82	I/O	通用输入/输出	GPIO82				106

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
		82					
GPIO83	I/O	通用输入/输出 83	GPIO83				110
GPIO84	I/O	通用输入/输出 84	GPIO84				111
GPIO85	I/O	通用输入/输出 85	GPIO85				112
I2C0_ALERT_n	I/OD	I2C0 控制信号 - 从器件输入/主器件输出（低有效）	GPIO13,GPIO27,GPIO37,GPIO45	35,44,46,73	50,59,61	61,71,73,110	69,80,82,126
I2C0_CTL_n	I/OD	I2C0 控制信号 - 从器件输入/主器件输出（低有效）	GPIO12,GPIO26,GPIO35,GPIO44	36,43,48,69	51,58,63,85	62,70,75,106	70,79,84,122
I2C0_SCL	I/OD	I2C0 开漏双向时钟	GPIO47,GPIO33,GPIO16,GPIO24,GPIO27,GPIO37,GPIO35,GPIO43,GPIO57,GPIO8,GPIO1,GPIO41	38,39,41,44,46,48,54,58,62,66	6,53,54,56,59,61,63,66,74,78,82	7,64,65,68,71,73,75,83,85,95,99,103	7,73,74,77,80,82,84,92,94,104,115,119
I2C0_SDA	I/OD	I2C0 开漏双向数据	GPIO46,GPIO48,GPIO17,GPIO25,GPIO26,GPIO35,GPIO32,GPIO56,GPIO42,GPIO0,GPIO40,GPIO44,GPIO10,GPIO34	6,40,42,43,48,49,57,63,64,69,76,77	7,55,57,58,63,64,65,79,80,85,93,94	3,8,67,69,70,75,82,84,94,100,101,106,115,116	3,8,76,78,79,84,91,93,103,116,117,122,131,132
I2C1_ALERT_n	I/OD	I2C1 控制信号 - 从器件输入/	GPIO54,GPIO78,GPIO80,GPIO45	73	13	15,17,55,110	15,17,63,126

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
		主器件输出（低有效）					
I2C1_CTL_n	I/OD	I2C1 控制信号 - 从器件输入/主器件输出（低有效）	GPIO46,GPIO53,GPIO67,GPIO64	6	12	3,14,16,66	3,14,16,75
I2C1_SCL	I/OD	I2C1 开漏双向时钟	GPIO51,GPIO3,GPIO9,GPIO15,GPIO29	60,75,78,3	10,76,90,95,100	11,97,112,117,128	11,107,128,133,144
I2C1_SDA	I/OD	I2C1 开漏双向数据	GPIO28,GPIO50,GPIO2,GPIO34,GPIO14	4,61,77,79	1,9,77,94,96	1,10,98,116,118	1,10,114,132,134
OUTXBAR0	O	输出 X-BAR 输出 0	GPIO24,GPIO58,GPIO2,GPIO34	41,61,77	56,67,77,94	68,86,98,116	77,95,114,132
OUTXBAR1	O	输出 X-BAR 输出 1	GPIO54,GPIO25,GPIO37,GPIO3,GPIO3,GPIO59	42,46,60,60	13,57,61,76,76,92	15,69,73,97,97,114	15,78,82,107,107,130
OUTXBAR10	O	输出 X-BAR 输出 10	GPIO78,GPIO62			17,77	17,86
OUTXBAR11	O	输出 X-BAR 输出 11	GPIO79,GPIO83			18	18,110
OUTXBAR12	O	输出 X-BAR 输出 12	GPIO67,GPIO65			16,76	16,85
OUTXBAR13	O	输出 X-BAR 输出 13	GPIO70			81	90
OUTXBAR2	O	输出 X-BAR 输出 2	GPIO48,GPIO55,GPIO60,GPIO26,GPIO26,GPIO4,GPIO5,GPIO14	43,43,59,74,79	7,43,44,58,58,75,89,96	8,53,54,70,70,96,111,118	8,61,62,79,79,105,127,134

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
OUTXBAR3	O	输出 X-BAR 输出 3	GPIO49,GPIO33,GPIO27,GPIO27,GPIO61,GPIO15,GPIO6	38,44,44,78,80	8,53,59,59,91,95,97	9,64,71,71,113,117,119	9,73,80,80,129,133,135
OUTXBAR4	O	输出 X-BAR 输出 4	GPIO28,GPIO7	4,68	1,84	1,105	1,121
OUTXBAR5	O	输出 X-BAR 输出 5	GPIO9,GPIO29	75,3	90,100	112,128	128,144
OUTXBAR6	O	输出 X-BAR 输出 6	GPIO11,GPIO16,GPIO44,GPIO30	37,39,69,1	52,54,85,98	63,65,106,125	71,74,122,141
OUTXBAR7	O	输出 X-BAR 输出 7	GPIO17,GPIO31	40,2	55,99	67,127	76,143
OUTXBAR8	O	输出 X-BAR 输出 8	GPIO43,GPIO81	54		83	92,109
OUTXBAR9	O	输出 X-BAR 输出 9	GPIO66,GPIO42,GPIO45	57,73		12,94,110	12,103,126
SDFM0_CLK1	I	SDFM0 通道 1 时钟输入	GPIO49,GPIO53,GPIO33,GPIO17	38,40	8,12,53,55	9,14,64,67	9,14,73,76
SDFM0_CLK2	I	SDFM0 通道 2 时钟输入	GPIO51,GPIO54,GPIO33	38	10,13,53	11,15,64	11,15,73
SDFM0_CLK3	I	SDFM0 通道 3 时钟输入	GPIO53,GPIO55,GPIO21	34	12,43,49	14,53,60	14,61,68
SDFM0_CLK4	I	SDFM0 通道 4 时钟输入	GPIO55,GPIO56,GPIO23	65	43,65,81	53,84,102	61,93,118
SDFM0_DATA1	I	SDFM0 通道 1 数据输入	GPIO48,GPIO16	39	7,54	8,65	8,74
SDFM0_DATA2	I	SDFM0 通道 2	GPIO50,GPIO32	49	9,64	10,82	10,91

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
		数据输入					
SDFM0_DATA3	I	SDFM0 通道 3 数据输入	GPIO52,GPIO20	33	11,48	13,59	13,67
SDFM0_DATA4	I	SDFM0 通道 4 数据输入	GPIO54, GPIO22	67	13,83	15,104	15,120
SDFM1_CLK1	I	SDFM1 通道 1 时钟输入	GPIO25,GPIO35,GPIO57	42,48	57,63,66	69,75,85	78,84,94
SDFM1_CLK2	I	SDFM1 通道 2 时钟输入	GPIO27,GPIO58,GPIO59	44	59,67,92	71,86,114	80,95,130
SDFM1_CLK3	I	SDFM1 通道 3 时钟输入	GPIO45,GPIO61,GPIO59,GPIO29	73,3	91,92,100	110,113,114,128	126,129,130,144
SDFM1_CLK4	I	SDFM1 通道 4 时钟输入	GPIO46,GPIO60,GPIO31	6,2	44,99	3,54,127	3,62,143
SDFM1_DATA1	I	SDFM1 通道 1 数据输入	GPIO49,GPIO24,GPIO56	41	8,56,65	9,68,84	9,77,93
SDFM1_DATA2	I	SDFM1 通道 2 数据输入	GPIO50,GPIO26,GPIO58	43	9,58,67	10,70,86	10,79,95
SDFM1_DATA3	I	SDFM1 通道 3 数据输入	GPIO28,GPIO51,GPIO60	4	1,10,44	1,11,54	1,11,62
SDFM1_DATA4	I	SDFM1 通道 4 数据输入	GPIO47,GPIO52,GPIO30	1	6,11,98	7,13,125	7,13,141
SPIM0_CLK	O	SPIM0 时钟	GPIO49,GPIO12,GPIO56,GPIO3,GPIO1,GPIO9	36,60,62,75	8,51,65,76,78,90	9,62,84,97,99,112	9,70,93,107,115,128
SPIM0_CS0_n	O	SPIM0 从器件片选 0（低有	GPIO66,GPIO11,GPIO57,GPIO0,GPIO23,GPIO5	37,63,65,74	52,66,79,81,89	12,63,85,100,102,111	12,71,94,116,118,127

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
		效）					
SPIM0_CS1_n	O	SPIM0 从器件片选 1（低有效）	GPIO79			18	18
SPIM0_CS2_n	O	SPIM0 从器件片选 2（低有效）	GPIO69,GPIO7	68	84	80,105	89,121
SPIM0_CS3_n	O	SPIM0 从器件片选 3（低有效）	GPIO70,GPIO44	69	85	81,106	90,122
SPIM0_MISO	I	SPIM0_主器件输入，从器件输出备份	GPIO55,GPIO13,GPIO17,GPIO1,GPIO40,GPIO10,GPIO77	35,40,62,64,76	43,50,55,78,80,93	53,61,67,99,101,115,126	61,69,76,115,117,131,142
SPIM0_MOSI	O	SPIM0 主器件输出，从器件输入	GPIO54,GPIO11,GPIO16,GPIO8,GPIO2,GPIO0,GPIO76	37,39,58,61,63	13,52,54,74,77,79	15,63,65,95,98,100,124	15,71,74,104,114,116,140
SPIM1_CLK	O	SPIM1 时钟	GPIO28,GPIO52,GPIO26,GPIO32,GPIO58,GPIO4,GPIO22,GPIO14	4,43,49,59,67,79	1,11,58,64,67,75,83,96	1,13,70,82,86,96,104,118	1,13,79,91,95,105,120,134
SPIM1_CS0_n	O	SPIM1 从器件片选 0（低有效）	GPIO53,GPIO67,GPIO33,GPIO27,GPIO23,GPIO59,GPIO15,GPIO29	38,44,65,78,3	12,53,59,81,92,95,100	14,16,64,71,102,114,117,128	14,16,73,80,118,130,133,144
SPIM1_CS1_n	O	SPIM1 从器件片选 1（低有效）	GPIO78			17	17

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
SPIM1_CS2_n	O	SPIM1 从器件片选 2（低有效）	GPIO79			18	18
SPIM1_CS3_n	O	SPIM1 从器件片选 3（低有效）	GPIO68, GPIO22	67	83	79,104	88,120
SPIM1_MISO	I	SPIM1_主器件输入，从器件输出备份	GPIO51,GPIO54,GPIO21,GPIO16,GPIO25,GPIO57,GPIO41,GPIO61,GPIO6,GPIO31	34,39,42,66,80,2	10,13,49,54,57,66,82,91,97,99	11,15,60,65,69,85,103,113,119,127	11,15,68,74,78,94,119,129,135,143
SPIM1_MOSI	O	SPIM1 主器件输出，从器件输入	GPIO50,GPIO53,GPIO60,GPIO20,GPIO24,GPIO56,GPIO40,GPIO7,GPIO30	33,41,64,68,1	9,12,44,48,56,65,80,84,98	10,14,54,59,68,84,101,105,125	10,14,62,67,77,93,117,121,141
SPIM2_CLK	O	SPIM2 时钟	GPIO63,GPIO43,GPIO81,GPIO41	54,66	82	78,83,103	87,92,109,119
SPIM2_CS0_n	O	SPIM2 从器件片选 0（低有效）	GPIO70,GPIO44	69	85	81,106	90,122
SPIM2_MISO	I	SPIM2_主器件输入，从器件输出备份	GPIO69,GPIO42,GPIO7	57,68	84	80,94,105	89,103,121
SPIM2_MOSI	O	SPIM2 主器件输出，从器件输入	GPIO68,GPIO39, GPIO22	56,67	83	79,93,104	88,102,120
SPIS0_CLK	I	SPIS0 时钟	GPIO49,GPIO1	62	8,78	9,99	9,115
SPIS0_CS0_n	O	SPIS0 从器件	GPIO23	65	81	102	118

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
		片选 0（低有效）					
SPIS0_MISO	O	SPIS0_主器件输入，从器件输出备份	GPIO40	64	80	101	117
SPIS0_MOSI	I	SPIS0 主器件输出，从器件输入	GPIO0	63	79	100	116
SRPWM0_A	O	SRPWM0A 输出	GPIO0,GPIO5,GPIO30	63,74,1	79,89,98	100,111,125	116,127,141
SRPWM0_B	O	SRPWM0B 输出	GPIO1,GPIO9,GPIO31	62,75,2	78,90,99	99,112,127	115,128,143
SRPWM1_A	O	SRPWM1A 输出	GPIO2,GPIO41,GPIO61	61,66	77,82,91	98,103,113	114,119,129
SRPWM1_B	O	SRPWM1B 输出	GPIO3,GPIO40,GPIO59	60,64	76,80,92	97,101,114	107,117,130
SRPWM10_A	O	SRPWM10A 输出	GPIO20	33	48	59	67
SRPWM10_B	O	SRPWM10B 输出	GPIO21	34	49	60	68
SRPWM11_A	O	SRPWM11A 输出	GPIO13	35	50	61	69
SRPWM11_B	O	SRPWM11B 输出	GPIO12	36	51	62	70
SRPWM2_A	O	SRPWM2A 输	GPIO4,GPIO10,GPIO14	59,76,79	75,93,96	96,115,118	105,131,134

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
		出					
SRPWM2_B	O	SRPWM2B 输出	GPIO5,GPIO34,GPIO15	74,77,78	89,94,95	111,116,117	127,132,133
SRPWM3_A	O	SRPWM3A 输出	GPIO22,GPIO15,GPIO6	67,78,80	83,95,97	104,117,119	120,133,135
SRPWM3_B	O	SRPWM3B 输出	GPIO23,GPIO7,GPIO14	65,68,79	81,84,96	102,105,118	118,121,134
SRPWM4_A	O	SRPWM4A 输出	GPIO16,GPIO8,GPIO6	39,58,80	54,74,97	65,95,119	74,104,135
SRPWM4_B	O	SRPWM4B 输出	GPIO17,GPIO35,GPIO9,GPIO30	40,48,75,1	55,63,90,98	67,75,112,125	76,84,128,141
SRPWM5_A	O	SRPWM5A 输出	GPIO10,GPIO31	76,2	93,99	115,127	131,143
SRPWM5_B	O	SRPWM5B 输出	GPIO11,GPIO29	37,3	52,100	63,128	71,144
SRPWM6_A	O	SRPWM6A 输出	GPIO28,GPIO12,GPIO72	4,36	1,51	1,62,120	1,70,136
SRPWM6_B	O	SRPWM6B 输出	GPIO13,GPIO74,GPIO29	35,3	50,100	61,122,128	69,138,144
SRPWM7_A	O	SRPWM7A 输出	GPIO24,GPIO14,GPIO73	41,79	56,96	68,118,121	77,134,137
SRPWM7_B	O	SRPWM7B 输出	GPIO32,GPIO15,GPIO75	49,78	64,95	82,117,123	91,133,139
SRPWM8_A	O	SRPWM8A 输出	GPIO64			66	75

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
SRPWM8_B	O	SRPWM8B 输出	GPIO80			55	63
SRPWM9_A	O	SRPWM9A 输出	GPIO55		43	53	61
SRPWM9_B	O	SRPWM9B 输出	GPIO60		44	54	62
SWCLK	O	SWD 时钟	SWCLK	45	60	72	81
SWDIO	I/O	SWD 数据	SWDIO	47	62	74	83
SYNCOUT	O	ETIMER 同步信号输出	GPIO52,GPIO6	80	11,97	13,119	13,135
TESTPIN0	O	测试引脚 0	GPIO20	33	48	59	67
TESTPIN1	O	测试引脚 1	GPIO21	34	49	60	68
TESTPIN2	O	测试引脚 2	GPIO26	43	58	70	79
TESTPIN3	O	测试引脚 3	GPIO27	44	59	71	80
UART0_RX	I	UART0 接收	GPIO28,GPIO49,GPIO50,GPIO17,GPIO25,GPIO35,GPIO3,GPIO9	4,40,42,48,60,75	1,8,9,55,57,63,76,90	1,9,10,67,69,75,97,112	1,9,10,76,78,84,107,128
UART0_TX	O	UART0 输出	GPIO48,GPIO51,GPIO16,GPIO24,GPIO37,GPIO8,GPIO2,GPIO29	39,41,46,58,61,3	7,10,54,56,61,74,77,100	8,11,65,68,73,95,98,128	8,11,74,77,82,104,114,144
UART1_RX	I	UART1 接收	GPIO46,GPIO13,GPIO11,GPIO57,GPIO23,GPIO41,GPIO15,GPIO29	6,35,37,65,66,78,3	50,52,66,81,82,95,100	3,61,63,85,102,103,117,128	3,69,71,94,118,119,133,144
UART1_TX	O	UART1 输出	GPIO28,GPIO12,GPIO56,GPIO40,	4,36,64,67,75,76,79	1,51,65,80,83,90,93,96	1,62,84,101,104,112,115,118	1,70,93,117,120,128,131,134

信号名称	引脚类型	说明	通用输入/输出 (GPIO)	LQFP80	LQFP100	LQFP128	LQFP144
			GPIO22,GPIO9,GPIO10,GPIO14				

3.3.3 电源和接地

表3-4 电源和接地

信号名称	引脚类型	说明	LQFP144	LQFP128	LQFP100	LQFP80
REGUENZ(2)	I	DVDD 外部电源选择，低电平使用内部电源，高电平使用外部电源。	103	94	73	-
DVDD	P	1.1 V 数字逻辑电源引脚。建议在每个 DVDD 引脚附近放置一个去耦电容，其最小总电容约为 20 μF。当不使用内部电压调整器时，去耦电容的确切值应由系统电压调节方案确定。	5、65、99、124	5、57、90、108	4、46、71、87	8、31、53、71
VDDA	P	3.3 V 模拟电源引脚。在每个引脚上放置一个最小 2.2 μF 的去耦电容到 VSSA。	48	43	34	26
VDDIO	P	3.3 V 数字 I/O 电源引脚。在每个引脚上放置一个最小 0.1 μF 的去耦电容。	4、66、98、125	4、58、89、109	3、47、70、88	7、32、52、72
VSS	G	数字地	6、64、100、123	6、56、91、107	5、45、72、86	9、30、55、70
VSSA	G	模拟地	47	42	33	25
NC		NC(1)，浮空处理	72	-	-	-
NC		NC(1)，浮空处理	108	-	-	-

说明

(1) 表中 NC 代表 Not Connected，是空脚。

(2) LQFP80 封装 REGUENZ 芯片内部默认接地，DVDD 不支持外部供电。

3.3.4 时钟、JTAG 和复位

表3-5 时钟、JTAG 和复位

信号名称	引脚类型	说明	LQFP144	LQFP128	LQFP100	LQFP80
XTAL_OUT	O	晶体振荡器输出	96	87	68	50
XTAL_IN	I	晶体振荡器或单端时钟输入，为了使用此振荡器，必须将一个石英晶体电路连接至 XTAL_IN 和 XTAL_OUT，并且预留补偿电容。此引脚也可用于馈入单端 3.3V 电平时钟。	97	88	69	51
SWCLK	I	SWD 时钟输入	81	72	60	45
SWDIO	I/O	SWD 数据信号输入输出	83	74	62	47
HWRST_n	I	SoC 外部复位信号（低有效）	2	2	2	5

3.4 引脚复用

下表列出了 ET6002 系列 GPIO 引脚复用关系，每个 GPIO 引脚的默认模式都是 GPIO 功能，但是 GPIO18 和 GPIO19 除外，这两个引脚的默认关系分别是 SWCLK 和 SWDIO。可以通过 IOMUX 寄存器配置把 SWCLK 和 SWDIO 配置为 GPIO18 和 GPIO19。未显示的 GPIO MODE 和带有“-”的行和列是保留的 GPIO 多路复用设置。ET6002 系列不存在 GPIO36 和 GPIO38。

表3-6 GPIO 多路复用引脚

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	Mode9	Mode10	Mode11	Mode13	Mode14
GPIO0	SRPW	SPIS0_M	-	SPIM0_M	-	I2C0_SD	SPIM0_C	-	CANFD1	-	EQEP0_I	-

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	Mode9	Mode10	Mode11	Mode13	Mode14
	M0_A	OSI		OSI		A	S0_n		_RX		NDEX	
GPIO1	SRPWM0_B	SPIS0_CLK	-	SPIM0_CLK	-	I2C0_SCL	SPIM0_MISO	-	CANFD1_TX	-	-	-
GPIO2	SRPWM1_A	-	-	CANFD1_RX	OUTXBAR0	-	SPIM0_MOSI	UART0_TX	-	I2C1_SDA	-	CANFD0_TX
GPIO3	SRPWM1_B	OUTXBAR1	-	CANFD1_TX	OUTXBAR1	-	SPIM0_CLK	UART0_RX	-	I2C1_SCL	-	CANFD0_RX
GPIO4	SRPWM2_A	-	CANFD1_TX	CANFD0_RX	OUTXBAR2	CANFD0_TX	SPIM1_CLK	EQEP1_STROBE	-	-	-	-
GPIO5	SRPWM2_B	-	OUTXBAR2	SRPWM0_A	CANFD1_RX	CANFD0_RX	SPIM0_CS0_n	-	-	-	-	-
GPIO6	SRPWM3_A	OUTXBAR3	SYNCOUT	SRPWM4_A	EQEP0_A	-	SPIM1_MISO	-	-	-	-	-
GPIO7	SRPWM3_B	SPIM0_CS2_n	OUTXBAR4	SPIM2_MISO	EQEP0_B	-	SPIM1_MOSI	-	-	-	-	-
GPIO8	SRPWM4_A	-	-	CANFD0_TX	EQEP0_STROBE	UART0_TX	SPIM0_MOSI	I2C0_SCL	-	-	-	-
GPIO9	SRPWM4_B	UART1_TX	OUTXBAR5	SRPWM0_B	EQEP0_INDEX	UART0_RX	SPIM0_CLK	-	-	-	-	I2C1_SCL
GPIO10	SRPWM5_A	-	-	SRPWM2_A	EQEP0_A	UART1_TX	SPIM0_MISO	I2C0_SDA	-	-	-	-
GPIO11	SRPWM5_B	-	OUTXBAR6	ETIMER0	EQEP0_B	UART1_RX	SPIM0_CS0_n	-	-	EQEP1_A	SPIM0_MOSI	-
GPIO12	SRPWM6_A	-	CANFD1_RX	SRPWM11_B	EQEP0_STROBE	UART1_TX	I2C0_CTL_n	-	ECAP13	SPIM0_CLK	CANFD0_RX	-

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	Mode9	Mode10	Mode11	Mode13	Mode14
GPIO13	SRPWM6_B	-	CANFD1_TX	SRPWM11_A	EQEP0_INDEX	UART1_RX	I2C0_ALERT_n	-	ECAP12	SPIM0_MISO	CANFD0_TX	-
GPIO14	SRPWM7_A	UART1_TX	-	SRPWM3_B	I2C1_SDA	OUTXBAR2	-	SPIM1_CLK	EQEP1_A	-	SRPWM2_A	-
GPIO15	SRPWM7_B	UART1_RX	-	SRPWM3_A	I2C1_SCL	OUTXBAR3	-	SPIM1_CS0_n	EQEP1_B	-	SRPWM2_B	-
GPIO16	SPIM0_MOSI	-	OUTXBAR6	ETIMER2	SRPWM4_A	UART0_TX	SDFM0_DATA1	EQEP0_STROBE	I2C0_SCL	-	EQEP1_B	SPIM1_MISO
GPIO17	SPIM0_MISO	-	OUTXBAR7	ETIMER6	SRPWM4_B	UART0_RX	SDFM0_CLK1	EQEP0_INDEX	I2C0_SDA	CANFD0_TX	-	-
SWCLK	GPIO18	-	-	-	-	-	-	-	-	-	-	-
SWDIO	GPIO19	-	-	-	-	-	-	-	-	-	-	-
GPIO20	EQEP0_A	-	TESTPIN0	SRPWM10_A	-	SPIM1_MOSI	SDFM0_DATA3	CANFD1_TX	ECAP10	-	-	-
GPIO21	EQEP0_B	-	TESTPIN1	SRPWM10_B	-	SPIM1_MISO	SDFM0_CLK3	CANFD1_RX	ECAP11	-	-	-
GPIO22	EQEP0_STROBE	SPIM1_CS3_n	UART1_TX	SPIM2_MOSI	-	SPIM1_CLK	SDFM0_DATA4	-	-	-	-	SRPWM3_A
GPIO23	EQEP0_INDEX	SPIS0_CS0_n	UART1_RX	SPIM0_CS0_n	-	SPIM1_CS0_n	SDFM0_CLK4	-	-	-	-	SRPWM3_B
GPIO24	OUTXBAR0	EQEP1_A	-	-	SRPWM7_A	SPIM1_MOSI	SDFM1_DATA1	-	I2C0_SCL	UART0_TX	-	-

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	Mode9	Mode10	Mode11	Mode13	Mode14
GPIO25	OUTXBAR1	EQEP1_B	-	ETIMER3	EQEP0_A	SPIM1_MISO	SDFM1_CLK1	-	I2C0_SDA	UART0_RX	-	-
GPIO26	OUTXBAR2	EQEP1_INDEX	TESTPIN2	ETIMER4	OUTXBAR2	SPIM1_CLK	SDFM1_DATA2	-	I2C0_CTL_n	I2C0_SDA	-	-
GPIO27	OUTXBAR3	EQEP1_STROBE	TESTPIN3	ETIMER5	OUTXBAR3	SPIM1_CS0_n	SDFM1_CLK2	-	I2C0_ALERT_n	I2C0_SCL	-	-
GPIO28	UART0_RX	-	SRPWM6_A	UART1_TX	OUTXBAR4	EQEP0_A	SDFM1_DATA3	EQEP1_STROBE	-	SPIM1_CLK	-	I2C1_SDA
GPIO29	UART0_TX	UART1_RX	SRPWM6_B	SRPWM5_B	OUTXBAR5	EQEP0_B	SDFM1_CLK3	EQEP1_INDEX	-	SPIM1_CS0_n	-	I2C1_SCL
GPIO30	CANFD0_RX	-	SPIM1_MOSI	SRPWM4_B	OUTXBAR6	EQEP0_STROBE	SDFM1_DATA4	-	CANFD1_RX	SRPWM0_A	-	-
GPIO31	CANFD0_TX	-	SPIM1_MISO	SRPWM5_A	OUTXBAR7	EQEP0_INDEX	SDFM1_CLK4	-	CANFD1_TX	SRPWM0_B	-	-
GPIO32	I2C0_SDA	-	SPIM1_CLK	-	SRPWM7_B	-	SDFM0_DATA2	-	CANFD0_TX	-	-	-
GPIO33	I2C0_SCL	-	SPIM1_CS0_n	ETIMER1	OUTXBAR3	-	SDFM0_CLK2	-	CANFD0_RX	EQEP1_B	-	SDFM0_CLK1
GPIO34	OUTXBAR0	-	-	SRPWM2_B	-	I2C0_SDA	-	-	-	-	-	I2C1_SDA
GPIO35	UART0_RX	-	I2C0_SDA	ECAP1	CANFD0_RX	I2C0_SCL	-	EQEP0_A	I2C0_CTL_n	SRPWM4_B	SDFM1_CLK1	-
GPIO37	OUTXBAR1	-	I2C0_SCL	ECAP0	UART0_TX	CANFD0_TX	-	EQEP0_B	I2C0_ALERT_n	-	-	-
GPIO3	-	-	SPIM2_	ECAP6	-	-	-	-	-	-	-	EQEP0_I

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	Mode9	Mode10	Mode11	Mode13	Mode14
9			MOSI									NDEX
GPIO40	SPIM1_MOSI	SPIS0_MISO	-	SPIM0_MISO	SRPWM1_B	I2C0_SDA	-	UART1_TX	EQEP0_A	-	-	-
GPIO41	-	-	-	SPIM2_CLK	SRPWM1_A	I2C0_SCL	-	UART1_RX	EQEP0_B	-	-	SPIM1_MISO
GPIO42	-	-	SPIM2_MISO	EQEP0_A	-	ETIMER7	OUTXBAR9	I2C0_SDA	-	-	-	-
GPIO43	-	-	SPIM2_CLK	ECAP2	-	-	OUTXBAR8	I2C0_SCL	EQEP0_INDEX	-	-	-
GPIO44	-	SPIM0_CS3_n	OUTXBAR6	SPIM2_CS0_n	EQEP0_A	I2C0_SDA	-	I2C0_CTL_n	-	-	-	-
GPIO45	-	-	I2C1_ALERT_n	OUTXBAR9	ECAP9	-	-	I2C0_ALERT_n	-	-	SDFM1_CLK3	-
GPIO46	-	-	-	UART1_RX	I2C1_CTL_n	CANFD1_TX	-	I2C0_SDA	-	-	SDFM1_CLK4	-
GPIO47	-	-	-	-	CANFD1_RX	-	-	I2C0_SCL	-	-	SDFM1_DATA4	-
GPIO48	OUTXBAR2	-	CANFD0_TX	-	-	UART0_TX	SDFM0_DATA1	I2C0_SDA	-	-	-	-
GPIO49	OUTXBAR3	SPIS0_CLK	CANFD0_RX	SPIM0_CLK	-	UART0_RX	SDFM0_CLK1	-	-	-	SDFM1_DATA1	-
GPIO50	EQEP0_A	-	-	UART0_RX	CANFD1_TX	SPIM1_MOSI	SDFM0_DATA2	I2C1_SDA	-	-	SDFM1_DATA2	-
GPIO51	EQEP0_B	-	-	UART0_TX	CANFD1_RX	SPIM1_MISO	SDFM0_CLK2	I2C1_SCL	-	-	SDFM1_DATA3	-

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	Mode9	Mode10	Mode11	Mode13	Mode14
GPIO52	EQEP0_STROBE	-	-	SPIM1_CLK	-	-	SDFM0_DATA3	SYNCOUT	-	-	SDFM1_DATA4	-
GPIO53	EQEP0_INDEX	-	I2C1_CTL_n	SPIM1_MOSI	-	SPIM1_CS0_n	SDFM0_CLK3	-	CANFD0_RX	-	SDFM0_CLK1	-
GPIO54	SPIM0_MOSI	-	I2C1_ALERT_n	SPIM1_MISO	EQEP1_A	OUTXBAR1	SDFM0_DATA4	-	-	-	SDFM0_CLK2	-
GPIO55	SPIM0_MISO	-	-	SRPWM9_A	EQEP1_B	OUTXBAR2	SDFM0_CLK4	-	-	-	SDFM0_CLK3	-
GPIO56	SPIM0_CLK	-	CANFD1_TX	ECAP3	EQEP1_STROBE	UART1_TX	SDFM1_DATA1	SPIM1_MOSI	I2C0_SDA	EQEP0_A	SDFM0_CLK4	-
GPIO57	SPIM0_CS0_n	-	CANFD1_RX	ECAP4	EQEP1_INDEX	UART1_RX	SDFM1_CLK1	SPIM1_MISO	I2C0_SCL	EQEP0_B	-	-
GPIO58	-	-	-	ECAP5	OUTXBAR0	SPIM1_CLK	SDFM1_DATA2	-	CANFD0_TX	EQEP0_STROBE	SDFM1_CLK2	-
GPIO59	-	-	-	SRPWM1_B	OUTXBAR1	SPIM1_CS0_n	SDFM1_CLK2	-	CANFD0_RX	EQEP0_INDEX	SDFM1_CLK3	-
GPIO60	-	-	CANFD1_TX	SRPWM9_B	OUTXBAR2	SPIM1_MOSI	SDFM1_DATA3	-	-	-	SDFM1_CLK4	-
GPIO61	-	-	CANFD1_RX	SRPWM1_A	OUTXBAR3	SPIM1_MISO	SDFM1_CLK3	-	-	-	-	CANFD0_RX
GPIO62	-	-	-	EQEP0_INDEX	-	ETIMER9	OUTXBAR10	-	-	-	-	-
GPIO63	-	-	-	SPIM2_CLK	-	-	-	-	-	-	-	-

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	Mode9	Mode10	Mode11	Mode13	Mode14
GPIO64	-	-	-	SRPWM8_A	I2C1_CTL_n	-	-	-	-	-	-	-
GPIO65	-	-	-	EQEP0_B	-	ETIMER8	OUTXBAR12	-	-	-	-	-
GPIO66	-	-	-	SPIM0_CS0_n	-	-	OUTXBAR9	-	-	-	-	-
GPIO67	-	-	-	SPIM1_CS0_n	CANFD1_TX	I2C1_CTL_n	OUTXBAR12	-	-	-	-	-
GPIO68	-	-	-	SPIM2_MOSI	SPIM1_CS3_n	ETIMER11	-	-	-	-	-	-
GPIO69	-	-	-	SPIM2_MISO	SPIM0_CS2_n	ETIMER12	-	-	-	-	-	-
GPIO70	-	-	-	SPIM2_CS0_n	SPIM0_CS3_n	ETIMER13	OUTXBAR13	-	-	-	-	-
GPIO71	-	-	-	EQEP0_INDEX	ECAP13	ETIMER13	-	-	-	-	-	-
GPIO72	-	-	-	SRPWM6_A	-	-	-	-	-	-	-	-
GPIO73	-	-	-	SRPWM7_A	-	-	-	-	-	-	-	-
GPIO74	-	-	-	SRPWM6_B	-	-	-	-	-	-	-	-
GPIO75	-	-	-	SRPWM7_B	-	-	-	-	-	-	-	-
GPIO7	-	-	-	SPIM0_M	CANFD0_	-	-	-	-	-	-	-

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	Mode9	Mode10	Mode11	Mode13	Mode14
6				OSI	TX							
GPIO77	-	-	-	SPIM0_MISO	CANFD0_RX	-	-	-	-	-	-	-
GPIO78	-	-	-	SPIM1_CS1_n	CANFD1_RX	I2C1_ALERT_n	OUTXBAR10	-	-	-	-	-
GPIO79	-	-	-	SPIM1_CS2_n	SPIM0_CS1_n	ETIMER10	OUTXBAR11	-	-	-	-	-
GPIO80	-	-	-	SRPWM8_B	I2C1_ALERT_n	-	-	-	-	-	-	-
GPIO81	-	-	SPIM2_CLK	OUTXBAR8	ECAP8	-	-	-	-	-	-	-
GPIO82	-	-	-	EQEP0_STROBE	ECAP7	-	-	-	-	-	-	-
GPIO83	-	-	-	EQEP0_STROBE	ECAP10	ETIMER10	OUTXBAR11	-	-	-	-	-
GPIO84	-	-	-	EQEP0_A	ECAP11	ETIMER11	-	-	-	-	-	-
GPIO85	-	-	-	EQEP0_B	ECAP12	ETIMER12	-	-	-	-	-	-

3.5 引脚配置

3.5.1 简介

IOC (IOMUX Controller) 是对 ET6002 数字 GPIO Pin 复用的控制模块，IOC 为每个 Pin 都提供了相关 Pullup, Pulldown, Input-Enable，斯密特输入触发使能，驱动强度等特性，但具体需要根据每个 Pin 的复用关系决定上述配置参数是否存在。

3.5.2 结构框图

IOC 框图如下所示：

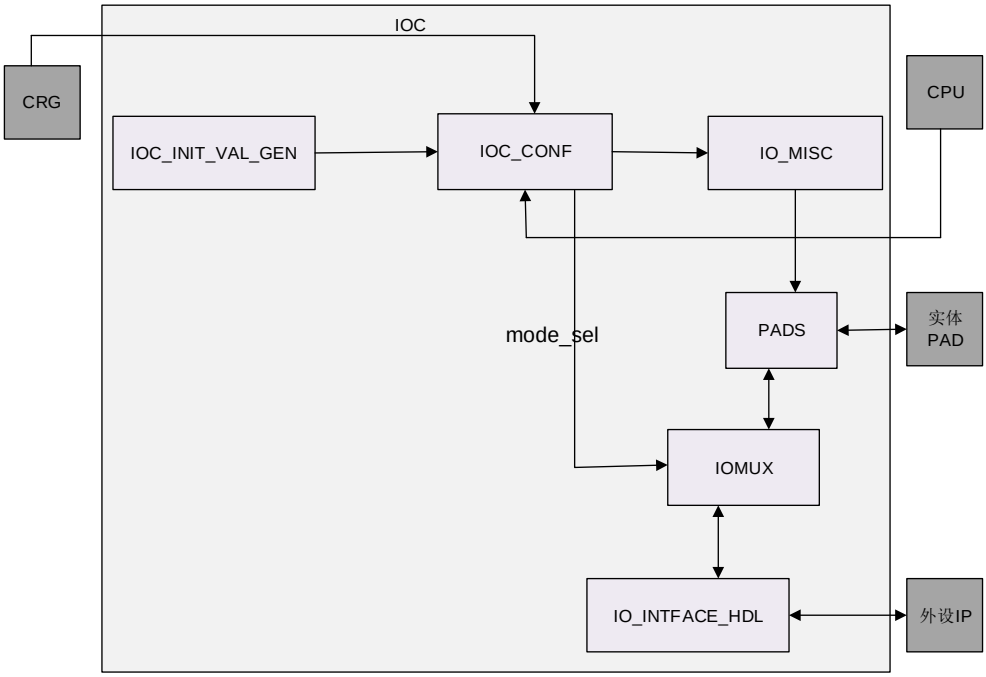


图3-5 IOC 框图

3.5.3 主要特性

IOC 模块具有以下特性:

- 对于 Pin 脚为输入模式下，提供 Pullup 和 Pulldown 控制；
- 对于 Pin 脚为输入模式，提供 Input Enable 控制 Pin 脚信号传递到芯片 Core 逻辑内部；
- 对于 Pin 脚为输出模式下，提供 4 个等级驱动能力；
- 针对 Pin 脚为复位信号输入，提供斯密特触发使能控制，使得复位信号上升沿和下降沿的电平变化 (transition) 时间变短；
- 如果 Pin 脚存在复用关系 (参见**《ET6002-数据手册》**)，提供 4bit mode 输入；

- 支持输入输出的备份功能，输入方向可从多个 Pin 脚进行输入，输出方向可将待发送信息映射到多个 Pin 脚上；可参见**《ET6002-数据手册》**中 **4.4"引脚复用"**章节获取各功能的备份数量及 Pin 管脚信息；

Pin 脚复用关系不支持动态配置，当进行配置时，必须要确保对应的 Pin 功能不在使用状态，否则可能导致相关功能异常；

3.5.4 功能描述

下面的功能描述以 GPIO32 作为例子进行解释。GPIO32 脚的模式复用关系如下：

Mode0	Mode1	Mode2	Mode3	Mode4	Mode5	Mode6	Mode7	Mode9	Mode10	Mode11	Mode13	Mode14
GPIO32	I2C0_SDA_BK1	-	SPIM1_CLK_BK1	-	SRPWM7_B_BK1	-	SDFM0_DATA2_BK1	-	CANFD0_TX_BK1	-	-	-

以 GPIO32 Pin 复用为例，对于 GPIO32 Pin 存在如下复用关系；

- 在模式 0 下为 GPIO32 使用，可以作为输入或者输出使用，输入输出方向由 GPIO5 IP 输出控制；
- 在模式 1 下为 I2C0_SDA 信号使用，注意该只是其中的一路备份。输入输出方向由 I2C0 控制；
- 在模式 3 固定为 SPIM1 的时钟输入信号使用；
- 在模式 5 固定为 SRPWM7_B 的输出信号，由 SRPWM IP 控制；
- 在模式 7 固定为 SDFM0 的数据输入；
- 在模式 10 固定为 CAN FD 的 TX 输出信号；

3.5.5 未使用管脚的建议

若芯片某些数字管脚在特定的应用场景下未被使用，为了防止引脚浮空造成芯片内部接收到一个不确定状态，建议芯片内部进行对应的上下拉配置，该上下拉配置应与《**ET6002-数据手册**》中章节 **4.6“未使用管脚的处理方式”**保持一致。

3.5.6 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 IOC 章节。

4 电源管理单元 (PMU)

芯片的电源分为三个部分：VDDIO，DVDD 和 VDDA。

- VDDIO 给芯片 I/O、PMU 和时钟模块供电。
- DVDD 可通过 PMU 内部 LDO 或者外灌电源给芯片的数字系统供电。
- VDDA 给 ADC，DAC，ACMP 和 Tsensor 供电。

电源框架图如下图所示：

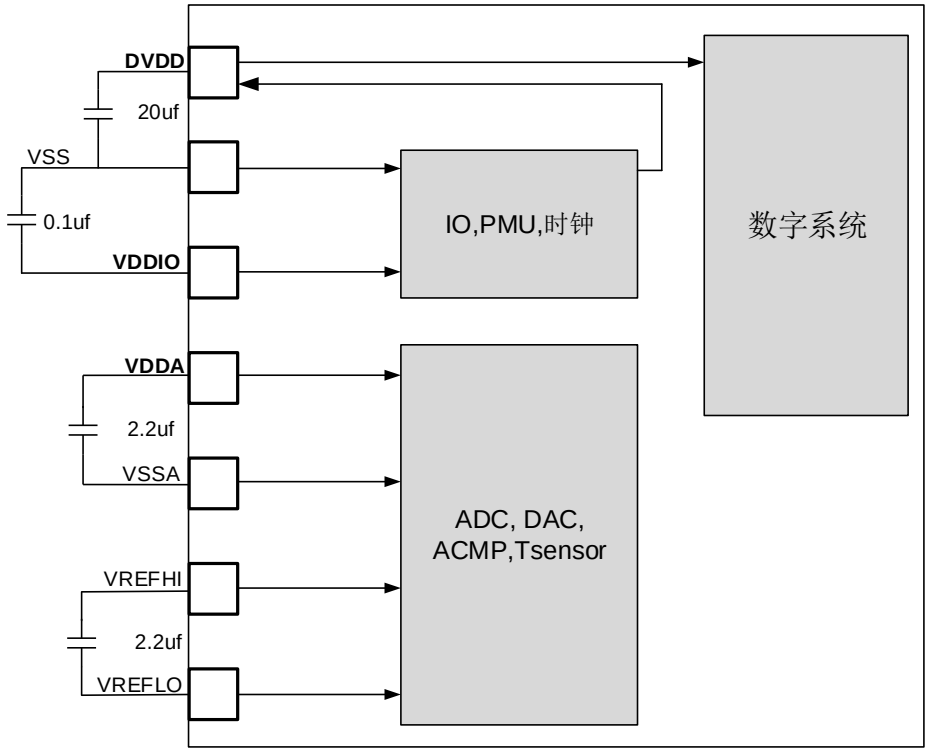


图4-1 电源框架图

VDDIO 和 VDDA 的工作电压要求介于 2.97 V 至 3.63 V 之间，即 3.3 V±10%。

4.2 内部低压差线性稳压器 (LDO)

VDDIO 电源给 PMU 内部 LDO 供电，该 LDO 用于产生 1.1 V (DVDD) 电压。在使用 PMU 内部 LDO 供电时需要将 REGUENZ 管脚下拉到 VSS，并且 LDO 的输出管脚需要添加负载电容以保证 LDO 的稳定性。

4.3 电源监控

芯片内部对 VDDIO 电压和 DVDD 电压进行了电压监控，包括上电复位检测，欠压检测和过压检测。除上电复位检测外，其余检测电路需要配置寄存器使能模块才能正常工作。

4.3.1 上电复位检测

针对 VDDIO 和 DVDD 电压的上电复位检测描述如下：

VDDIO 电压：

- 当 VDDIO 电压上电高于 2.8 V，延迟 1.2 ms（典型值）输出高电平。
- 当 VDDIO 电压下电低于 2.7 V，输出低电平。

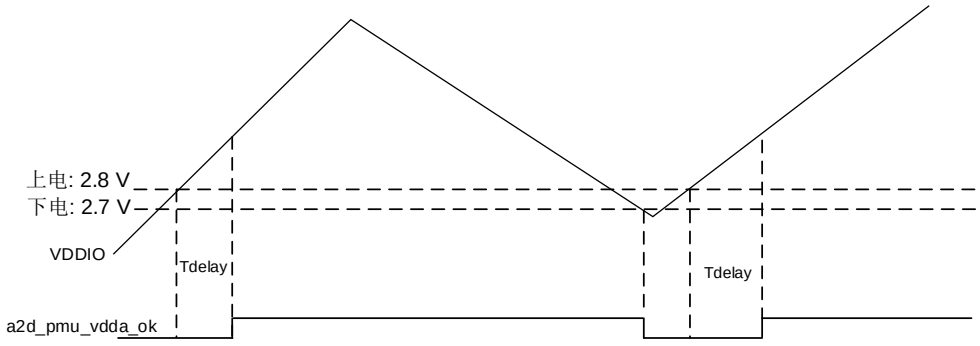


图4-2 VDDIO 电压上电复位检测

a2d_pmu_vdda_ok 寄存器状态可通过 9.2.7 "状态及历史记录上报"章节查看。

DVDD 电压：

- 当 DVDD 电压上电高于 0.8 V，延迟 1.2 ms (典型值) 输出高电平。
- 当 DVDD 电压下电低于 0.7 V，输出低电平。

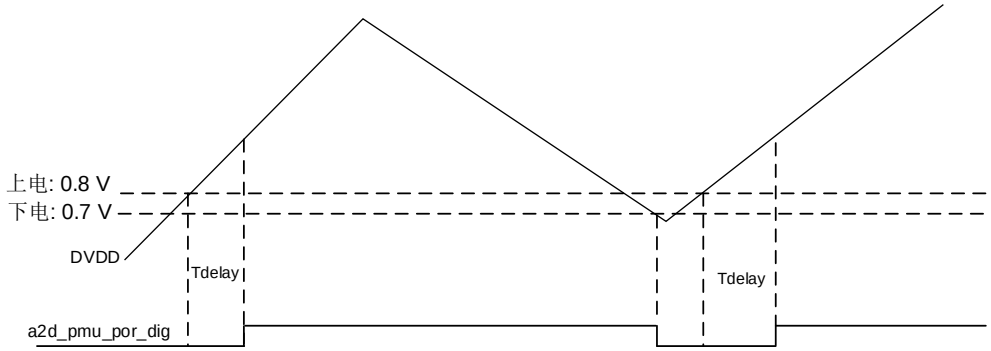


图4-3 DVDD 电压上电复位检测

a2d_pmu_por_dig 寄存器状态可通过 9.2.7 "状态及历史记录上报"章节查看。

4.3.2 欠压检测

下面两章节介绍 VDDIO 电压和 DVDD 电压的欠压检测。

欠压检测包括欠压指示和欠压消除两部分。

4.3.2.1 欠压指示

VDDIO 电压和 DVDD 电压欠压指示阈值说明如下：

VDDIO 电压：

当 VDDIO 电压在经过判决阈值 2.89 V 时，欠压产生。

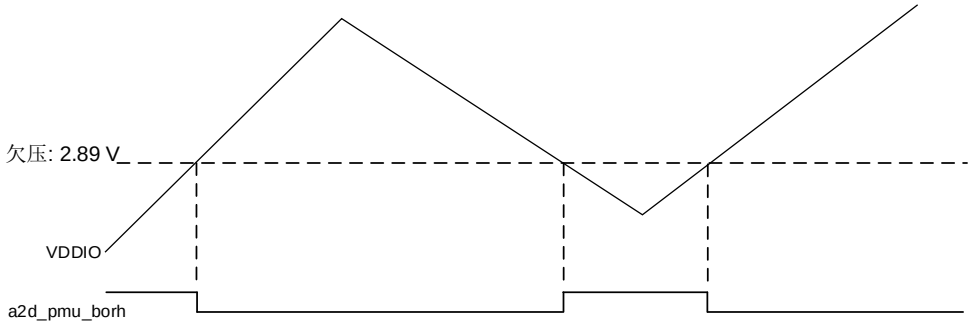


图4-4 VDDIO 电压欠压检测

a2d_pmu_borh 寄存器状态可通过 9.2.7 "状态及历史记录上报"章节查看。

DVDD 电压：当 DVDD 电压在经过判决阈值 0.89 V 时，欠压产生。

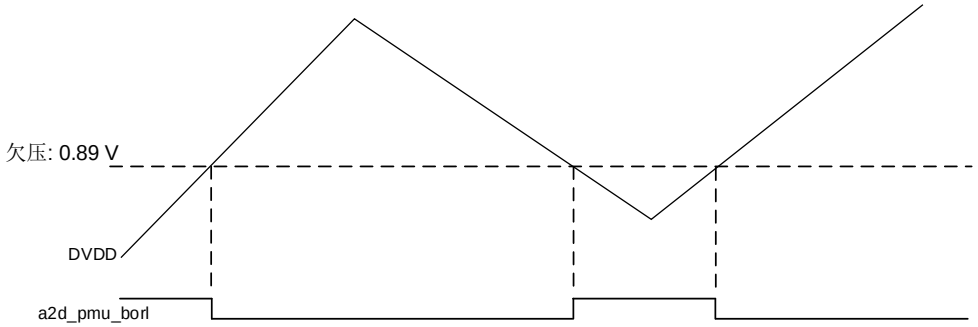


图4-5 DVDD 电压欠压检测

a2d_pmu_borl 寄存器状态可通过 9.2.7 "状态及历史记录上报"章节查看。

4.3.2.2 欠压消除

VDDIO 电压和 DVDD 电压的欠压消除阈值说明如下：

VDDIO 电压：

当 VDDIO 电压在经过判决阈值 2.99 V 时，欠压消除。

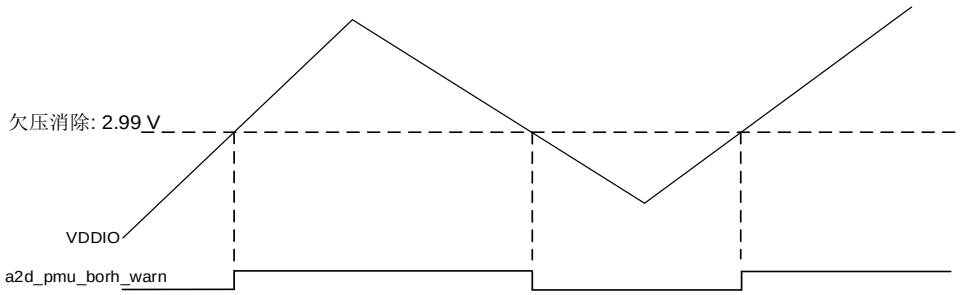


图4-6 VDDIO 欠压消除

a2d_pmu_borh_warn 寄存器状态可通过 9.2.7 "状态及历史记录上报"章节查看。

DVDD 电压：

当 DVDD 电压在经过判决阈值 0.99 V 时，欠压消除。

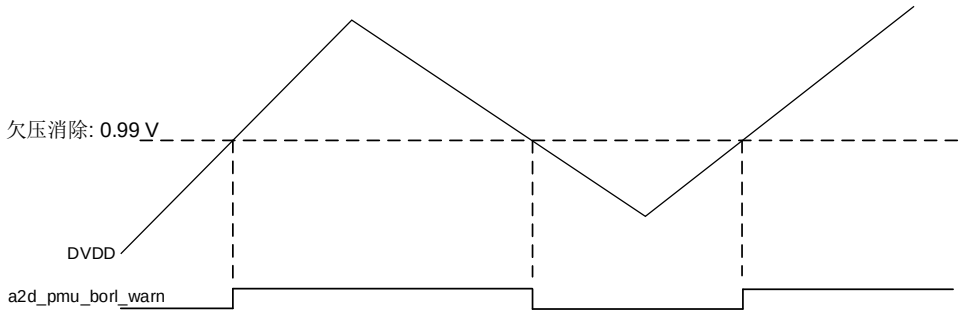


图4-7 DVDD 欠压消除

a2d_pmu_borl_warn 寄存器状态可通过 9.2.7 "状态及历史记录上报"章节查看。

4.3.3 过压检测

VDDIO 电压和 DVDD 电压的过电压检测如下：

VDDIO 电压：

当 VDDIO 电压在经过判决阈值 3.6 V 时，过压产生。

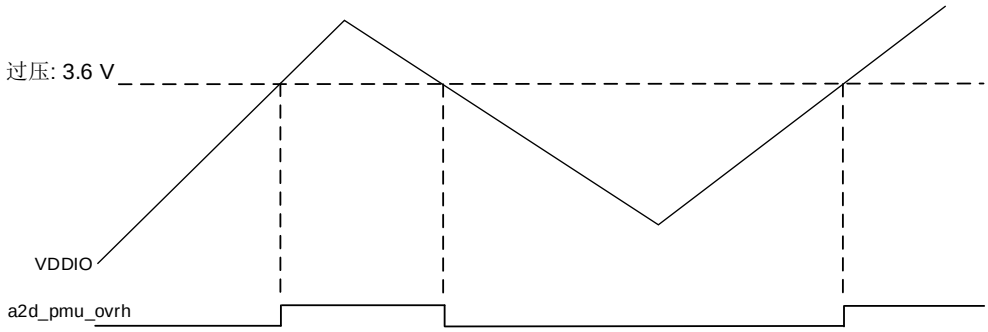


图4-8 VDDIO 过压检测

a2d_pmu_ovrh 寄存器状态可通过 9.2.7 "状态及历史记录上报"章节查看。

DVDD 电压：

当 DVDD 电压在经过判决阈值 1.26 V 时，过压产生。

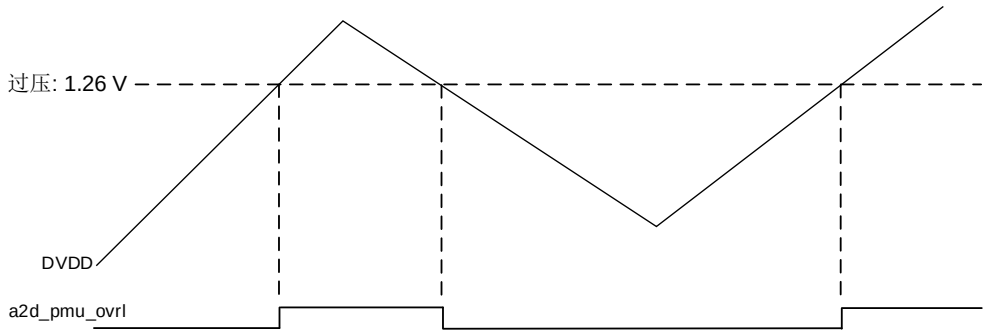


图4-9 DVDD 过压检测

a2d_pmu_ovrl 寄存器状态可通过 9.2.7 "状态及历史记录上报"章节查看。

5 复位与时钟

复位与时钟模块管理整个微控制器的时钟和复位信号的生成。

5.1 复位

共有三种类型的复位，分别为系统复位、电源复位、模块级软件复位。

表5-1 复位域影响范围

复位源	M7 Core	外设	IOC (1)	SYSCTRL (2)
POR 复位	Yes	Yes	Yes	全部
外部管脚复位	Yes	Yes	Yes	部分
看门狗复位	Yes	Yes	No	部分
全局软复位	Yes	Yes	No	部分
CPU 请求全局复位	Yes	Yes	No	部分
CPU 请求核复位	Yes	No	No	No

说明

(1) IOC 复位域默认只受 POR 复位和外部管脚复位影响，但是可以通过 CFG_SYS2IOC 寄存器配置为受 POR 复位和外部管脚复位及系统复位影响。

(2) SYSCTRL 的部分寄存器只受 POR 复位影响，具体见 SYSCTRL 的寄存器描述。

注意

软件在使用 CPU 请求核复位的复位源时，需要确保 AXI 总线空闲(FLASH/SRAM 读写操作)。并建议同时开启看门狗及看门狗复位，执行复位操作的代码运行在 ITCM 中。

5.1.2 系统复位

系统复位将复位除 Power-on 复位域的其他所有寄存器，当发生以下任一事件时，将产生一个系统复位：

- HWRST_n 引脚上的低电平 (外部管脚复位)
- 2 个看门狗请求复位 (WDG 复位)
- 1 个 CPU 请求复位 (CPU 请求全局软复位)
- 全局软件复位 (内部软件复位)

可通过查看 SYSCTRL 模块中的复位状态寄存器 (RST_REC) 来识别复位事件的来源。

5.1.2.1 引脚复位

芯片具有一个独立的复位引脚 HWRST_n，默认上拉，拉低该引脚将引起系统复位。

图 5-1 展示了 RC 复位参考电路。

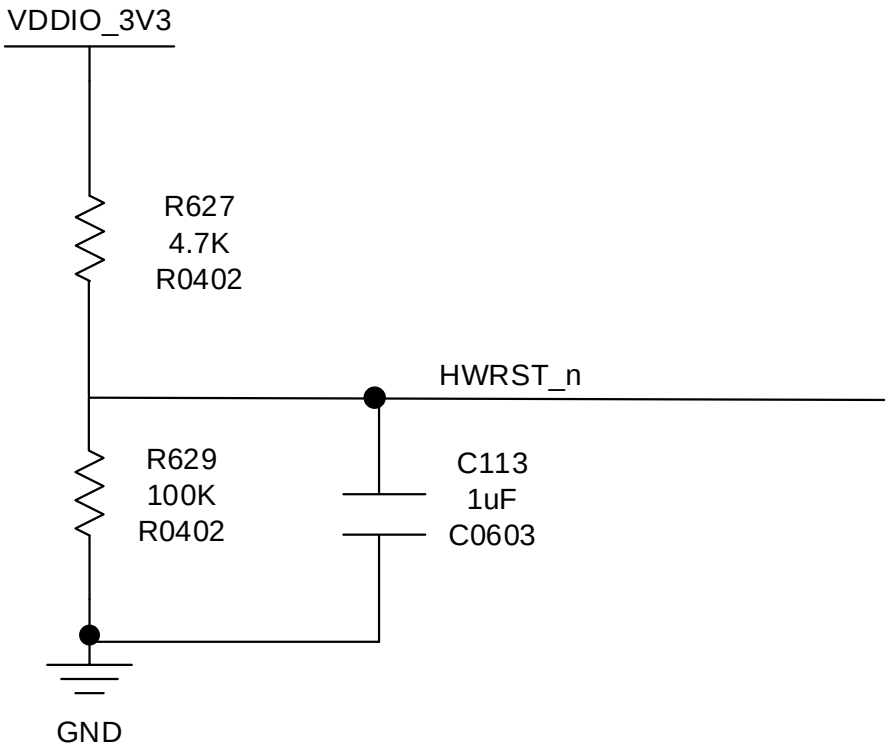


图5-1 RC 复位参考电路

说明

芯片上电启动时，需要引脚提供一个复位有效的低电平，在 3.3 V 上电稳定后，**HWRST_n** 保持低电平持续时间大于 1 us。如上图所示，可在芯片 **HWRST_n** 引脚采用 RC (R627、C113)复位电路实现。R627 推荐值 4.7 K，C113 推荐值 1 uF。

5.1.2.2 看门狗请求复位

芯片二个看门狗可独立发起复位请求 (高电平有效)，将引起系统复位。

5.1.2.3 CPU 请求复位

1 个 CPU 可独立发起复位请求 (高电平有效)。

- CPU 复位请求有效时：会产生全局复位 (将 SYSCTRL 请求模式寄存器 (**CFG_SYSC_CPU0**) 配置为 **0**)。
- CPU 复位请求无效时：不会产生全局复位 (将 SYSCTRL 请求模式寄存器 (**CFG_SYSC_CPU0**) 配置为 **1**),只会复位 M7 核本身。

说明

SYSCTRL 控制器 CFG_SYSC_CPU0 寄存器默认值为 0.

5.1.2.4 全局软件复位

全局软件复位可通过向 SYSCTRL 模块的全局软件复位寄存器 (**CFG_SYSC2CRG**) 写入固定值 (**0X45670123**) 来产生，同时将引起系统复位。

5.1.3 电源复位

5.1.3.1 上电复位

芯片内部有一个完整的上电复位 (POR) 电路，当供电电压达到系统指定的限定电压时，系统将解除上电复位。电源复位将复位所有寄存器。

5.1.4 模块级软件复位

芯片支持为外设模块提供模块级软件复位控制，POR 上电复位后，默认只有一部分的外设处于解复位状态，用户若要使用这些外设，需要先解复位这些模块。具体参见**《ET6002-用户手册》**中 CRG 模块寄存器 **CFG_CRG_SRST0/1/2/3** 的描述。

说明

除 (CAN、UART、I2C、SPI、CORDIC、CRC、AES、QEP) 外，其余模块的模块级软件复位仅支持在线解复位，不支持在线复位。

5.2 时钟

芯片有以下不同的时钟源：

- HSE 振荡器时钟
- HSI 振荡器时钟
- PLL 时钟

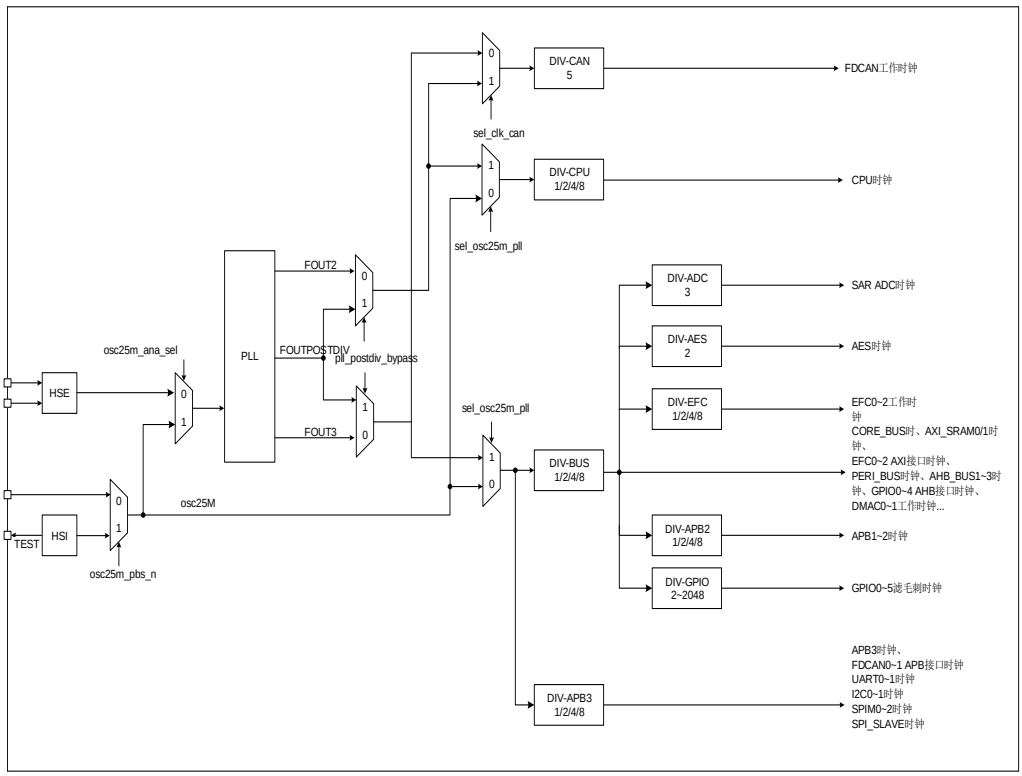


图5-2 时钟电路简图

为了高度的灵活性，用户可选择使用外部晶振方案或者使用内部 RC 振荡器的无晶振方案。通过 PLL 的合理配置，既可满足系统的最高时钟频率，也可为需要特定时钟的外设保证合适的频率。

可通过配置 CRG 模块内的多个分频器，分别实现 PLL、CPU、AXI、AHB、APB 等时钟的频率调整，其中 APB_BUS3 域固定时钟频率为 100 MHz。

5.2.1 HSE 振荡器

HSE 框图如图 5-3 所示。

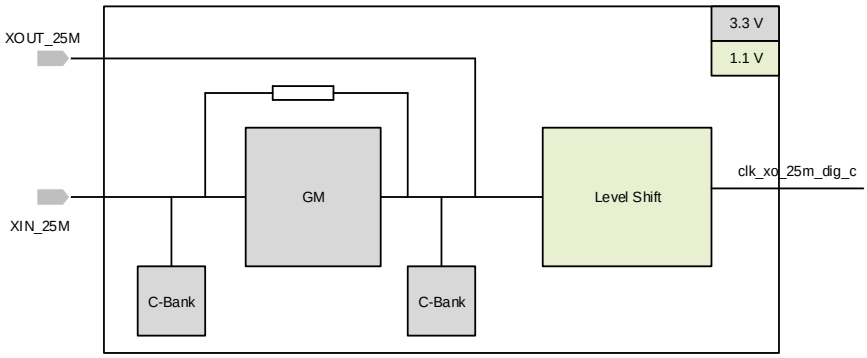


图5-3 HSE 结构框图

HSE 采用外部高精度时钟源，该时钟源可直接供片内数字电路使用，同时可作为 PLL 参考时钟输入，为 PLL 提供高精度/低抖动的参考时钟。

25M 晶体驱动电路支持 12.5M~37.5M 晶体输入模式，同时支持 TCXO 单端输入模式。

HSE 的特点是精度非常高。

在芯片上电完成后，HSE 默认处理关闭状态，需要通过配置 SYSCTRL 中的寄存器 (**CMU_XO_25M.d2a_xo_25m_pd_c**) 来控制 HSE 的开启或者关闭。

外部晶振谐振器 Y1 和负载电容 C1/C2 必须尽可能地靠近 HSE 的引脚，以尽量减小输出失真和起振稳定时间。外部负载电容 C1/C2 推荐值 10 pF，HSE 内部参考电容配置 (**CMU_XO_25M.d2a_xo_25m_ctrim_c**) 采用默认配置 0x0。

5.2.2 HSI 振荡器

HSI 框图如图 5-4 所示。

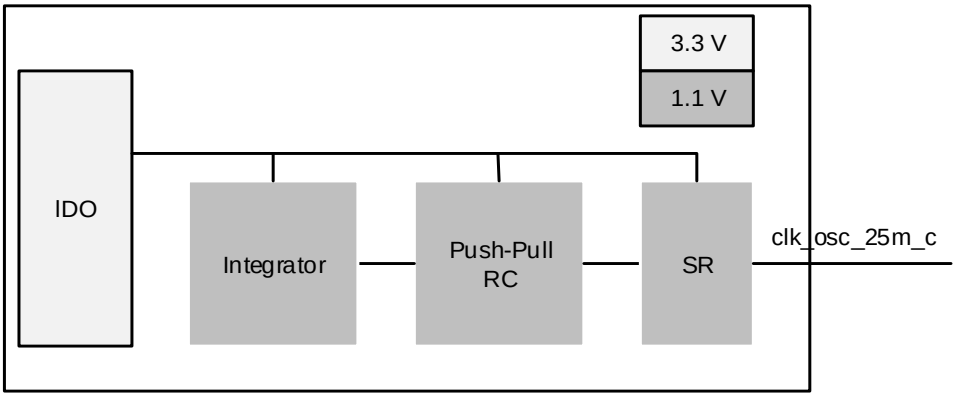


图5-4 HSI 结构框图

HSI 时钟信号是由内部高频 RC 振荡器 (25 MHz) 生成，HSI 可直接用作为系统时钟，也可作为 PLL 参考时钟输入。

芯片内部的高频 RC 振荡器的优点是方案成本更低 (无晶振方案，无需使用外部组件)。此外，其启动速度也要比 HSE 晶振快，但是其精度不及外部晶振或陶瓷谐振器。

在芯片上电完成后，HSI 振荡器默认为使能状态，无法通过配置将其关闭。

5.2.3 PLL

芯片内置 1 个 PLL，为 CPU、总线、以及外设提供时钟。

PLL 框图如图 5-5 所示。

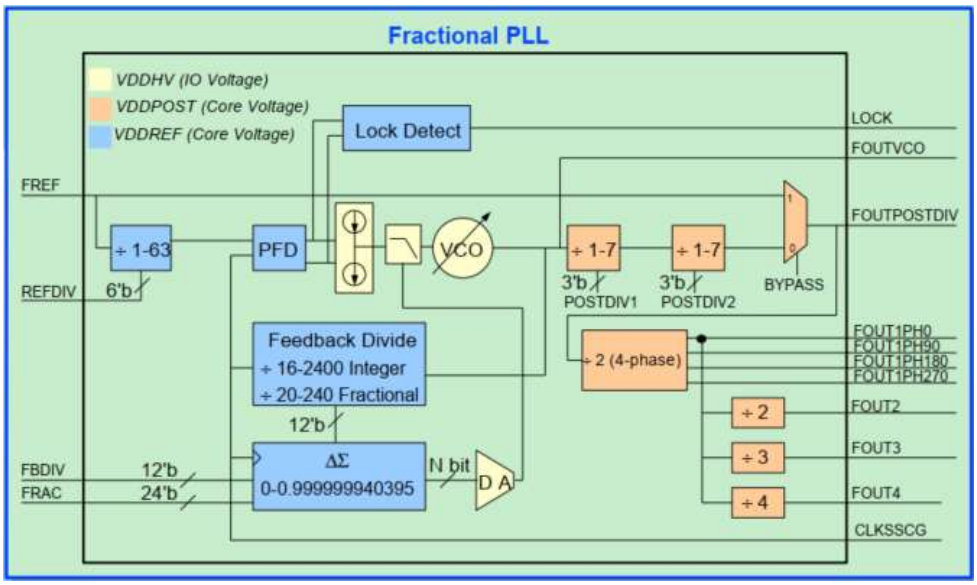


图5-5 PLL 结构框图

由于在 PLL 使能后，PLL 配置参数不建议动态修改，所以要求先完成 PLL 参数的配置，然后再使能。如需要在 PLL 使能后调整其配置参数，需要先将系统的时钟源从 PLL 切换到一个稳定的时钟源上 (例如 HSI)，然后关闭 PLL，完成 PLL 参数的修改后，再使能 PLL。

PLL 的参考时钟输入选择，通过 CRG 模块的寄存器 (**CFG_PLLREF_CFG.cfg_crg_pll_ref_sel**)进行选择配置。PLL 的参考时钟源如下：

- 'b0: HSE
- 'b1: HSI

PLL 输入参考频率范围 1 MHz~800 MHz，输出范围 12 MHz~2400 MHz。

可通过 CRG 模块寄存器 (**CFG_PLL_CFG_CFG0**) 进行配置如下公式中的相关 PLL 参数，然后运用此公式计算出具体的 PLL 输出时钟频率：

PLL_CLK_OUT = FREF/REFDIV*FBDIV/POSTDIV1/POSTDIV2

5.2.4 系统时钟 (SYSCLK)

在系统复位后，默认的系统时钟为 HSI，若需要切换时钟，需要在目标时钟源已准备就绪 (时钟在启动完成或 PLL 锁相后)，才可从一个时钟源切换到另一个。

以下时钟间切换均支持动态无毛刺切换：

1. 从 HSI 时钟切换到 PLL 时钟 (正常 Boot 流程)
2. 从 PLL 时钟切换到 HSI 时钟 (系统复位/Sleep 模式)

说明

切换时钟时，需注意分频比和 Flash 时序参数配置和切换顺序。

系统上电启动时钟切换顺序：

1. 系统上电后默认工作于 HSI 时钟域，在该时钟域下完成 NVR_CFG 硬件读取并配置 eFlash 内部寄存器，然后解复位 CPU；

2. 通过软件将 CRG 模块的寄存器 (**CFG_KEY_CTR.sw_access_key**) 配置为 **0x7879a8a9**，对 CRG 模块的配置寄存器解锁；
3. 通过 CPU Boot 程序配置 PLL 参数，等待 PLL 锁定后，软件配置 CRG 模块的寄存器 (**CFG_PLLCLK_CFG0.cfg_crg_sel_osc25m_pll**)、CRG 模块内的多个分频器系数 (**CFG_PLLCLK_CFG0，CFG_PLLCLK_CFG1，CFG_PLLCLK_CFG2，CFG_PLLCLK_CFG3**)，将系统时钟 HSI 切换到 PLL 时钟域；
4. 通过软件配置 CRG 模块的寄存器 (**CFG_ GTEN0.cfg_crg_osc25m_gten**) 对 HSI 进行门控；
5. 通过读取 HSI 所对应的 Flash NVR 里的 trim 值对其进行 trim，等待 trim 完成后打开 (**CFG_GTEN0.cfg_crg_osc25m_gten**) 时钟门控；
6. 通过软件将 CRG 模块的寄存器 (**CFG_KEY_CTR.sw_access_key**) 配置为非 **0x7879a8a9** 的值，对 CRG 模块的配置寄存器上锁；

芯片推荐运行场景下的各时钟关系如下表所示。

表5-2 芯片典型时钟 profile

子系统	IP/模块	High Performance Profile1			Sleep Profile		
		频率 (MHz)	时钟源	说明	频率 (MHz)	时钟源	说明
Clock Source	OSC_25M	25	/	On	25	/	On
	XO_25M	25	/	On	25	/	On
	PLL_FOUT2	300	/	On	300	/	Off
	PLL_FOUT3	200	/	On	200	/	Off
Core	M7 Core	300	PLL_FOUT2		25	OSC_25M	
Core	CORE_BUS (AXI_SRAM0/1)	200	PLL_FOUT3		25	OSC_25M	
Core	EFC0/1/2 AXI interface	200	PLL_FOUT3	EFC AXI clock	25	OSC_25M	
	EFC0/1/2 Flash ctrl	200	PLL_FOUT3	Flash controller clock	25	OSC_25M	
	EFC0/1/2 Flash device	50	PLL_FOUT3	Flash device clock	25	OSC_25M	
PERI	PERI_BUS (AHB_BUS2, AHB_BUS3, AHB_DMAC0/1)	200	PLL_FOUT3		25	OSC_25M	
	AHB_BUS1 (SRPWM, ETIMER, XBAR)	200	PLL_FOUT3		25	OSC_25M	
	AHB_BUS2	200	PLL_FOUT3		25	OSC_25M	
	SAR-ADC AHB interface	200	PLL_FOUT3		25	OSC_25M	
	SAR-ADC core	66.6	PLL_FOUT3		25	OSC_25M	
	AES	100	PLL_FOUT3		25	OSC_25M	
	APB_BUS1 (STIMER, IOC)	100	PLL_FOUT3		25	OSC_25M	

子系统	IP/模块	High Performance Profile1			Sleep Profile		
		频率 (MHz)	时钟源	说明	频率 (MHz)	时钟源	说明
	APB_BUS2	100	PLL_FOUT3		25	OSC_25M	
	APB_BUS3 (UART, I2C, SPIM, SPIS)	100	PLL_FOUT3		/		off
	CAN FD	40	PLL_FOUT3		/		off

5.2.4.1 分频器配置约束

BUS 时钟下的分频器系数配置受如下 LCM (最小公倍数) 和 Update (更新使能) 控制。

- CLK_BUS：CFG_PLLCLK_CFG0.cfg_crg_div_bus_lcm/cfg_crg_div_pll_update

说明

LCM 为其所控制的分频器系数之最小公倍数，Update 为其所控制的分频器系数之更新使能。

用户需要在 Update 不使能时，配置好相应的分频器系数和最小公倍数，然后使能 Update，延迟一段时间 (分频器输入时钟计数到配置的 LCM 最大值) 后关闭 Update，以防止对分频器的误配置操作。

LCM/Update 控制的分频器如下：

- CLK_BUS: DIV_APB、DIV_EFC、DIV_ADC

5.2.4.2 奇数分频

只有 DIV_CAN、DIV_ADC 两个分频器支持奇数分频时 50/50 duty cycle，其余分频器只支持偶数分频时 50/50 duty cycle。

5.2.4.3 时钟门控

芯片支持为外设模块提供模块级时钟门控，具体参见 CRG 模块寄存器 **CFG_GTEN0/1/2/3** 描述。

5.2.5 时钟安全系统 (CSS)

为了增强系统的可靠性，防止因时钟失效造成系统出错甚至死机的严重后果，芯片内部增加了一个时钟安全监控 (CSS) 模块。CSS 模块可监控的时钟关系如下：

1. 支持 HSI 时钟检测 PLL_FOUT3/PLL_FOUT2 时钟有无告警上报，分别通过 SYSCTRL 模块的寄存器 (DIG_STS. pll1_los_sts)、(DIG_STS. pll0_los_sts) 和寄存器 (DIG_REC. pll1_los_rec)、(DIG_REC. pll0_los_rec) 上报丢失状态和丢失历史记录；通过 CRG 模块的寄存器 (CFG_PLL1_CNTRPT.cfg_crg_pll1_drift_cnt)、(CFG_PLL0_CNTRPT.cfg_crg_pll0_drift_cnt) 上报检测周期内计数值，软件通过计数值计算频偏；
2. 支持 HSI 时钟检测 HSE 时钟有无告警上报，分别通过 SYSCTRL 模块的寄存器 (DIG_STS.xo25m_los_sts) 和寄存器 (DIG_REC.xo25m_los_rec) 上报丢失状态和丢失历史记录；通过 CRG 模块的寄存器 (CFG_XO25M_CNTRPT.cfg_crg_xo25m_drift_cnt) 上报检测周期内计数值，软件通过计数值计算频偏；
3. 支持 HSE 时钟检测 HSI 时钟有无告警上报，分别通过 SYSCTRL 模块的寄存器 (DIG_STS.osc25m_los_sts) 和寄存器 (DIG_REC.osc25m_los_rec) 上报丢失状态和丢失历史记录；通过 CRG 模块的寄存器 (CFG_OSC25M_CNTRPT.cfg_crg_osc25m_drift_cnt) 上报检测周期内计数值，软件

通过计数值计算频偏；

时钟检测功能可通过 CRG 模块寄存器 (**CFG_CLK_TEST_EN**) 来控制开启或者关闭(所有时钟检测同时开启或者关闭)。

可通过配置 CRG 模块的如下寄存器来控制每对时钟检测的窗口宽度和丢失判决阈值：

- CFG_OSC25M_TEST0
- CFG_OSC25M_TEST1
- CFG_XO25M_TEST0
- CFG_XO25M_TEST1
- CFG_CRG_PLL0FOUT_TEST0
- CFG_CRG_PLL0FOUT_TEST1
- CFG_CRG_PLL1FOUT_TEST0
- CFG_CRG_PLL1FOUT_TEST1

用户通过时钟状态寄存器获取到各时钟状态，当发生时钟异常时，用户可根据具体应用选择相应处理方式。

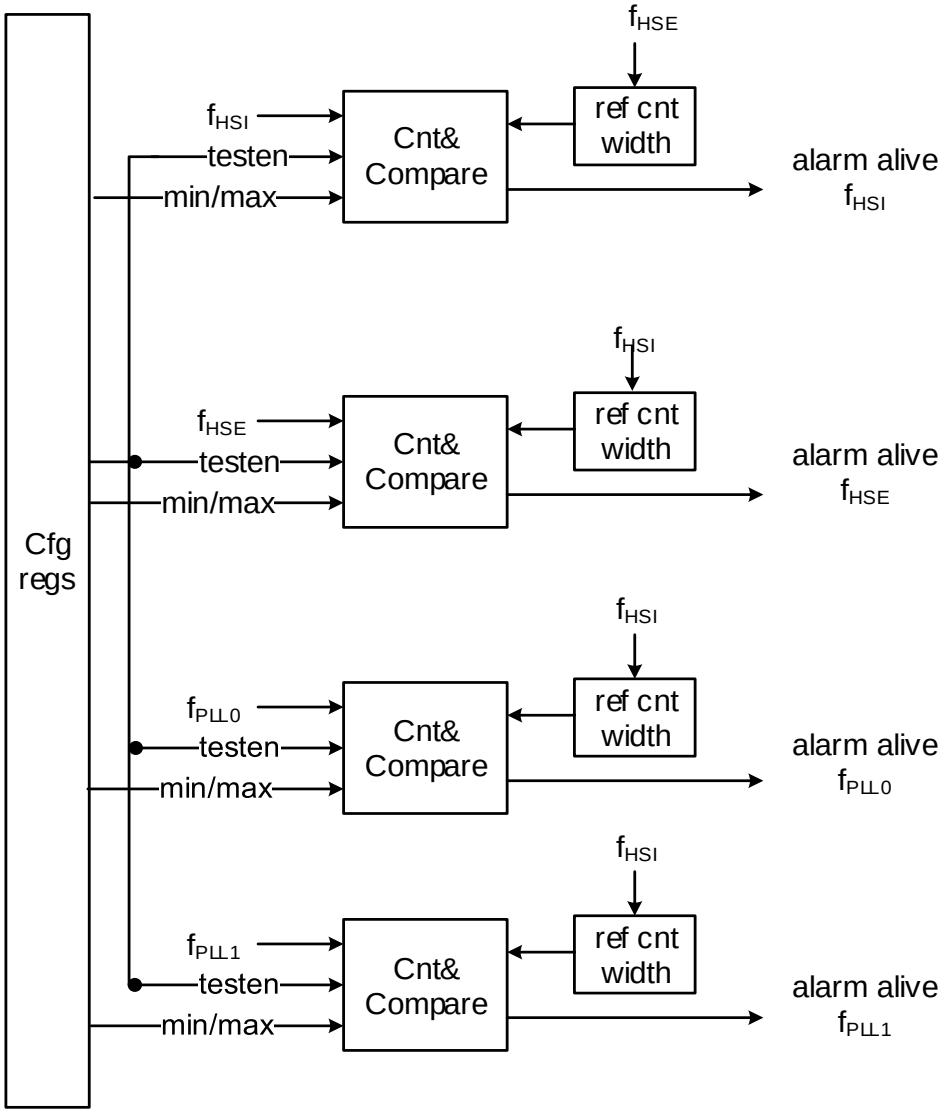


图5-6 时钟检测简图

5.2.6 看门狗时钟

通过配置 CRG 模块寄存器 (**CFG_PLLCLK_CFG3.cfg_crg_wdg0/1_sel**)，二个看门狗可独立选择工作时钟源 (HSI/CLK_APB_BUS1)。

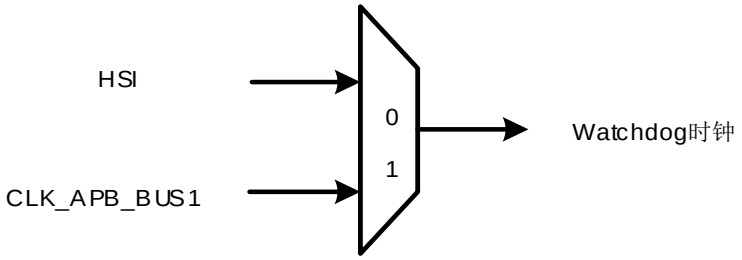


图5-7 看门狗时钟选择

5.2.7 时钟输出功能

芯片上电完成后，HSI 时钟默认通过 GPIO51 输出，无需用户配置。

芯片共有四个测试引脚，用户可通过配置 SYSCTRL 模块中测试引脚输出选择寄存器(**CFG_TESTPIN_SEL0/1**)，选择需要输出的时钟源。四个测试引脚可独立配置选择，可选择输出的时钟源如下：

- PLL_FOUT3/16
- PLL_FOUT2/16
- HSI
- HSE
- CLK_CPU/8
- CLK_BUS/8
- CLK_APB/8
- CLK_APB3/8
- CLK_CAN/8
- CLK_SYSCNT/8
- CLK_WDG0/8
- CLK_WDG1/8

对于四个测试引脚，必须将相应的 GPIO 端口 (GPIO20 /GPIO21 /GPIO26 /GPIO27)的功能复用设置为正确的复用模式。I/O 输出时钟频率建议不超过 75 MHz。

5.3 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 CRG 章节。

6 存储器和总线架构

6.1 系统架构

如图 6-1 所示，系统通过多层 AXI 和 AHB 总线矩阵，以及 AHB-to-APB 桥，AHB 和 APB 总线实现总线主设备与总线从设备的互连。

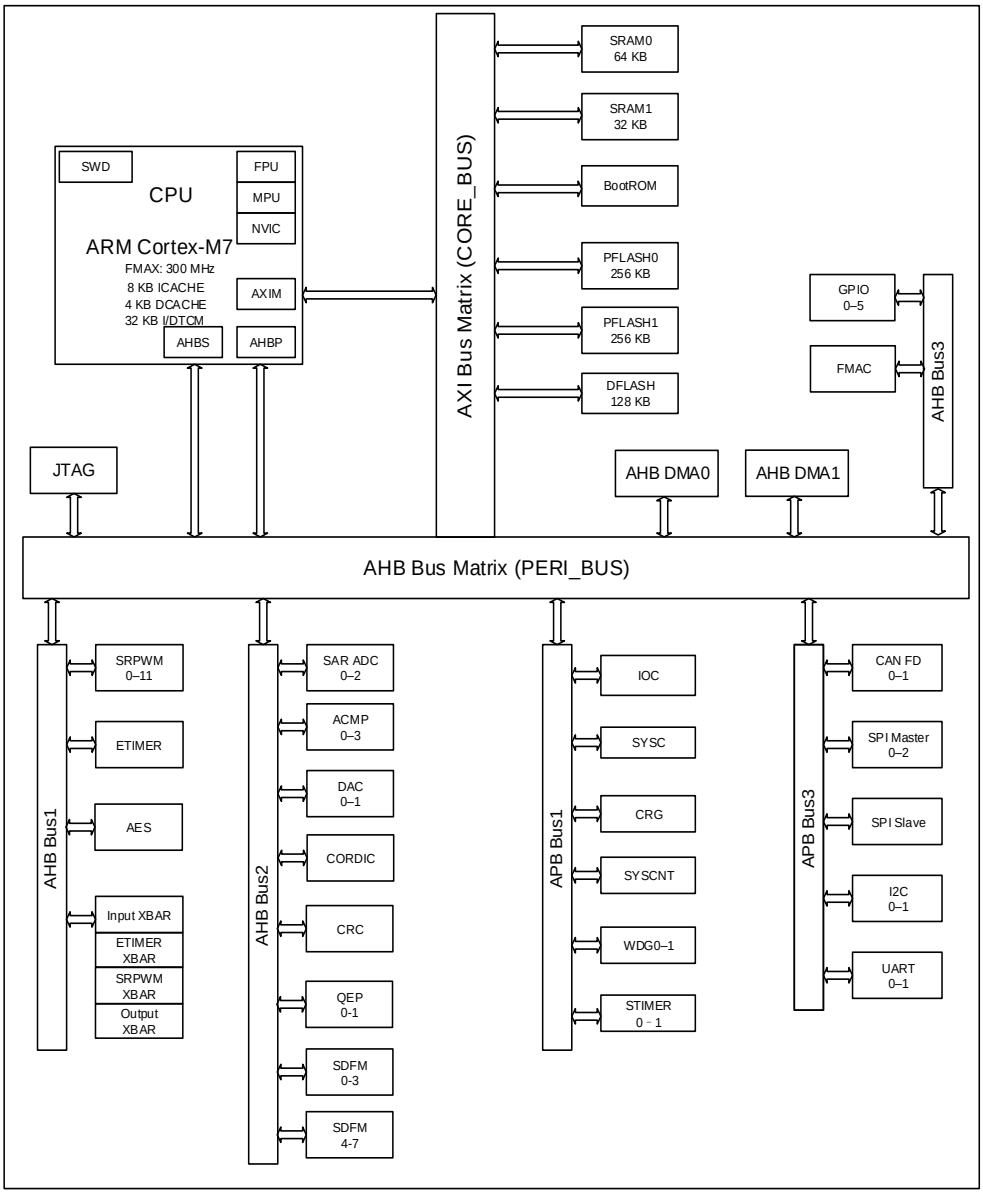


图6-1 ET6002 系统架构框图

6.1.1 设备互连

表 6-1 列出了每个总线主设备与总线从设备之间的互连关系。

表6-1 总线主设备与从设备的互连

外设	CPU	DMA(1)
ITCM_CPU	Y	Y
DTCM_CPU	Y	Y
SRAM0	Y	Y
SRAM1	Y	Y
BootROM	N(2)	N(2)
PFLASH0	Y	Y
PFLASH1	Y	Y
DFLASH	Y	Y
AHB DMA0 配置	Y	Y
AHB DMA1 配置	Y	Y
GPIOA	Y	Y
GPIOB	Y	Y
GPIOC	Y	Y
GPIOD	Y	Y
GPIOE	Y	Y
GPIOF	Y	Y
FMAC	Y	Y
SRPWM1	Y	Y
SRPWM2	Y	Y
SRPWM3	Y	Y
SRPWM4	Y	Y
SRPWM5	Y	Y
SRPWM6	Y	Y
SRPWM7	Y	Y
SRPWM8	Y	Y
SRPWM9	Y	Y
SRPWM10	Y	Y
SRPWM11	Y	Y
ETIMER	Y	Y
AES	Y	Y
Input XBAR	Y	Y
SRPWM XBAR	Y	Y
ETIMER XBAR	Y	Y
Output XBAR	Y	Y
SAR-ADC0 配置	Y	Y
SAR-ADC1 配置	Y	Y
SAR-ADC2 配置	Y	Y

外设	CPU	DMA(1)
DAC0 配置	Y	Y
DAC1 配置	Y	Y
Dual-ACMP0~3 配置	Y	Y
CORDIC	Y	Y
CRC	Y	Y
QEP0	Y	Y
QEP1	Y	Y
SDFM0-3	Y	Y
SDFM4-7	Y	Y
IOC	Y	Y
SYSC	Y	Y
CRG	Y	Y
SYSCNT	Y	Y
WDG0	Y	Y
WDG1	Y	Y
STIMER0	Y	Y
STIMER1	Y	Y
CPU 配置	Y	Y
EFC0_CFG (PFLASH0 控制器)	Y	Y
EFC1_CFG (PFLASH1 控制器)	Y	Y
EFC2_CFG (DFLASH 控制器)	Y	Y
CORESC_CFG (处理器子系统配置)	Y	Y
SRAM0 配置	Y	Y
SRAM1 配置	Y	Y
CAN FD0	Y	Y
CAN FD1	Y	Y
SPI_Master0	Y	Y
SPI_Master1	Y	Y
SPI_Master2	Y	Y
SPI_Slave	Y	Y
UART0	Y	Y
UART1	Y	Y
I2C0	Y	Y
I2C1	Y	Y

说明

(1) DMA 同时代表 AHB DMA0、AHB DMA1。

(2) CPU 和 DMA 可否访问 BootROM 由 SYSC 寄存器控制。在 BootROM 代码执行完成前，如 CPU 检测到 SWD 信号，将 BootROM 代码挂起，此时 SYSC 寄存器未被配置，CPU 和 DMA 可对 BootROM 访问。当 BootROM 启动完成后，SYSC 寄存器被设置为禁止 CPU 和 DMA 对 BootROM 访问。

6.1.2 总线矩阵

- **AXI 总线矩阵 (CORE_BUS)**

  CORE_BUS 总线包含 2 个总线主设备端口，7 个总线从设备端口，为从多个主设备到多个从设备的并发访问提供保证和仲裁，可实现高速外设的高效同步运行。

  CORE_BUS 总线内部实现地址译码，访问通路路由与仲裁，AXI/AHB 协议转换，总线数据位宽转换，同异步处理等。

  对总线上未分配的地址空间的访问，返回错误响应。

  仲裁采用轮询调度算法。

  DTCM 和 ITCM (数据和指令紧密耦合 RAM) 通过 CPU 子系统内部专用 TCM 总线直接连接到 Cortex-M7 内核。ITCM 由 Cortex-M7 以 CPU 时钟速度 (零等待周期) 访问。

- **AHB 总线矩阵 (PERI_BUS)**

  PERI_BUS 总线包含 5 个总线主设备端口，6 个总线从设备端口，为从多个主设备到多个从设备的并发访问提供保证和仲裁，可实现高速外设的高效同步运行。

  仲裁采用循环调度算法。

- **AHB Bus**

  用于把 AHB 外设连接到 PERI_BUS 总线上。

- **APB Bus**

  用于把 APB 外设连接到 PERI_BUS 总线上。

说明

AHB DMA 通过 PERI_BUS 访问 CORE_BUS 中 Flash 空间时，Flash 数据访问时长会超过总线设置的超时时长，需要通过系统控制器关闭 PERI2CORE 的超时监测功能；

6.2 存储器映射

程序存储器、数据存储器、寄存器和输入输出 (I/O) 接口统一编址在 4G 字节的线性地址空间里。

数据字节在存储器中以小端格式存放，小端指的是数据的低位保存在内存的低地址中，而数据的高位保存在内存的高地址中。

未分配给片上存储器和外设的所有存储区域均视为“保留区”。

图 6-2 展示了基本的存储器架构，遵循 Cortex®-M7 的内存要求。

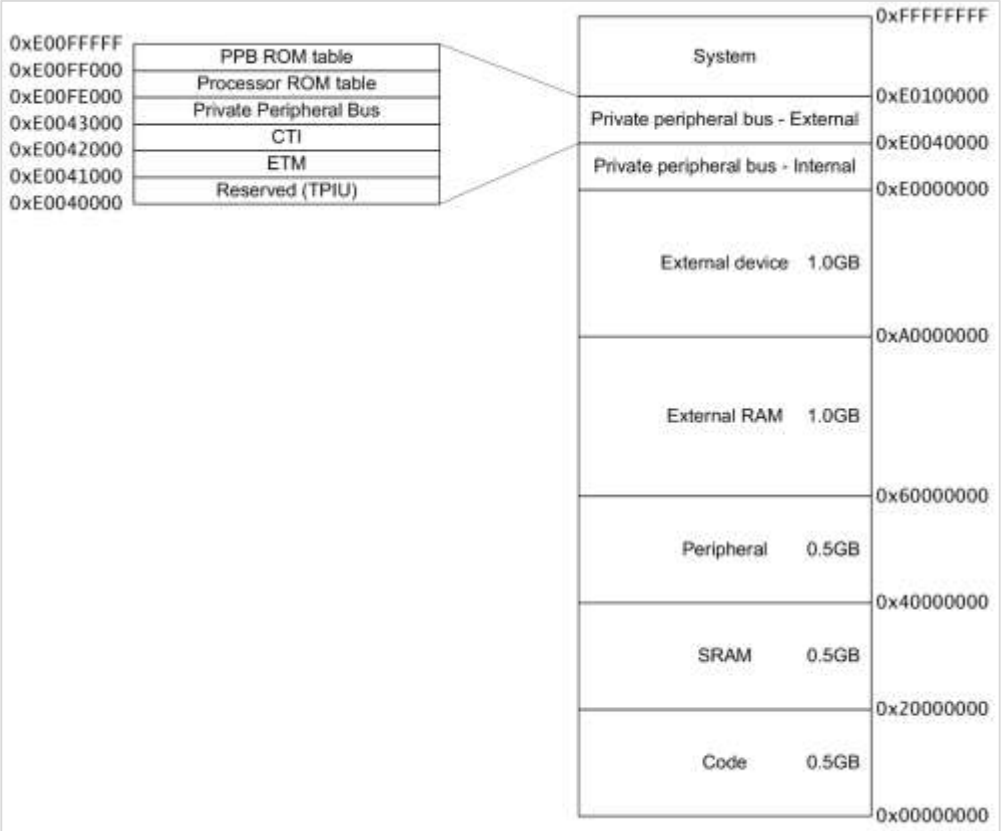


图6-2 Cortex®-M7 存储映射图

6.2.1 CPU 地址映射

表 6-2 提供了 CPU 视角系统中各个外设模块的地址边界划分。

表6-2 CPU 地址映射

CPU 地址映射			
外设模块	**空间大小**	**起始地址**	**结尾地址**
ITCM_M7	32 KB	0x0000_0000	0x0000_7FFF
PFLASH0	256 KB	0x0800_0000	0x0803_FFFF
PFLASH1	256 KB	0x0804_0000	0x0807_FFFF
DFLASH	128 KB	0x0808_0000	0x0809_FFFF
DTCM_M7	32 KB	0x2000_0000	0x2000_7FFF
AXI_SRAM0	64 KB	0x2040_0000	0x2040_FFFF
AXI_SRAM1	32 KB	0x2041_0000	0x2041_7FFF
ROM	32 KB	0x2045_0000	0x2045_7FFF
SRPWM0	4 KB	0x4000_0000	0x4000_0FFF
SRPWM1	4 KB	0x4000_1000	0x4000_1FFF
SRPWM2	4 KB	0x4000_2000	0x4000_2FFF
SRPWM3	4 KB	0x4000_3000	0x4000_3FFF
SRPWM4	4 KB	0x4000_4000	0x4000_4FFF
SRPWM5	4 KB	0x4000_5000	0x4000_5FFF

CPU 地址映射			
外设模块	空间大小	起始地址	结尾地址
SRPWM6	4 KB	0x4000_6000	0x4000_6FFF
SRPWM7	4 KB	0x4000_7000	0x4000_7FFF
SRPWM8	4 KB	0x4000_8000	0x4000_8FFF
SRPWM9	4 KB	0x4000_9000	0x4000_9FFF
SRPWM10	4 KB	0x4000_A000	0x4000_AFFF
SRPWM11	4 KB	0x4000_B000	0x4000_BFFF
SRPWM_Com	4 KB	0x4000_C000	0x4000_CFFF
XBAR	4 KB	0x4000_D000	0x4000_DFFF
AES	4 KB	0x4000_E000	0x4000_EFFF
SRPWM_BC	4 KB	0x4000_F000	0x4000_FFFF
ETIMER	4 KB	0x4001_C000	0x4001_CFFF
DAC0	4 KB	0x4010_0000	0x4010_0FFF
DAC1	4 KB	0x4010_1000	0x4010_1FFF
CRC	4 KB	0x4010_2000	0x4010_2FFF
Dual-ACMP0	4 KB	0x4010_4000	0x4010_4FFF
Dual-ACMP1	4 KB	0x4010_5000	0x4010_5FFF
Dual-ACMP2	4 KB	0x4010_6000	0x4010_6FFF
Dual-ACMP3	4 KB	0x4010_7000	0x4010_7FFF
QEP0	4 KB	0x4010_8000	0x4010_8FFF
QEP1	4 KB	0x4010_9000	0x4010_9FFF
SDFM0	4 KB	0x4010_A000	0x4010_AFFF
SDFM1	4 KB	0x4010_B000	0x4010_BFFF
SARC0	4 KB	0x4010_C000	0x4010_CFFF
SARC1	4 KB	0x4010_D000	0x4010_DFFF
SARC2	4 KB	0x4010_E000	0x4010_EFFF
CORDIC	4 KB	0x4010_F000	0x4010_FFFF
AHB_DMA0	4 KB	0x4020_1000	0x4020_1FFF
AHB_DMA1	4 KB	0x4020_2000	0x4020_2FFF
GPIOA	4 KB	0x4020_6000	0x4020_6FFF
GPIOB	4 KB	0x4020_7000	0x4020_7FFF
GPIOC	4 KB	0x4020_8000	0x4020_8FFF
GPIOD	4 KB	0x4020_9000	0x4020_9FFF
GPIOE	4 KB	0x4020_A000	0x4020_AFFF
GPIOF	4 KB	0x4020_B000	0x4020_BFFF
FMAC	4 KB	0x4020_C000	0x4020_CFFF
SYSC	4 KB	0x4040_0000	0x4040_0FFF
WDT0	4 KB	0x4040_2000	0x4040_2FFF
WDT1	4 KB	0x4040_3000	0x4040_3FFF
STIMER64_0	4 KB	0x4040_6000	0x4040_6FFF
STIMER64_1	4 KB	0x4040_7000	0x4040_7FFF
CRG	4 KB	0x4040_C000	0x4040_CFFF

CPU 地址映射			
外设模块	空间大小	起始地址	结尾地址
SYSCNT_RO	4 KB	0x4040_D000	0x4040_DFFF
SYSCNT_CTRL	4 KB	0x4040_E000	0x4040_EFFF
IOC	4 KB	0x4040_F000	0x4040_FFFF
UART0	4 KB	0x4050_0000	0x4050_0FFF
UART1	4 KB	0x4050_1000	0x4050_1FFF
I2C0	4 KB	0x4050_4000	0x4050_4FFF
I2C1	4 KB	0x4050_5000	0x4050_5FFF
SPI_Master0	4 KB	0x4050_8000	0x4050_8FFF
SPI_Master1	4 KB	0x4050_9000	0x4050_9FFF
SPI_Master2	4 KB	0x4050_A000	0x4050_AFFF
SPI_Slave0	4 KB	0x4050_C000	0x4050_CFFF
CAN FD0	12 KB	0x4051_8000	0x4051_AFFF
CAN FD1	12 KB	0x4052_0000	0x4052_2FFF
CPU_CFG	4 KB	0x4060_0000	0x4060_0FFF
EFC0_CFG	4 KB	0x4060_2000	0x4060_2FFF
EFC1_CFG	4 KB	0x4060_3000	0x4060_3FFF
EFC2_CFG	4 KB	0x4060_4000	0x4060_4FFF
SRAM0_CFG	4 KB	0x4060_8000	0x4060_8FFF
SRAM1_CFG	4 KB	0x4060_9000	0x4060_9FFF
M7_0_Private_PERI	64 KB	0xE000_0000	0xE000_FFFF

6.2.2 DMA 地址映射

表 6-3 提供了 DMA 视角系统中各个外设模块的地址边界划分

表6-3 DMA 地址映射

DMA 地址映射			
外设模块	空间大小	起始地址	结尾地址
ITCM_M7	32 KB	0x2800_0000	0x2800_7FFF
PFLASH0	256 KB	0x0800_0000	0x0803_FFFF
PFLASH1	256 KB	0x0804_0000	0x0807_FFFF
DFLASH	128 KB	0x0808_0000	0x0809_FFFF
DTCM_M7	32 KB	0x2840_0000	0x2840_7FFF
AXI_SRAM0	64 KB	0x2040_0000	0x2040_FFFF
AXI_SRAM1	32 KB	0x2041_0000	0x2041_7FFF
ROM	32 KB	0x2045_0000	0x2045_7FFF
SRPWM0	4 KB	0x4000_0000	0x4000_0FFF
SRPWM1	4 KB	0x4000_1000	0x4000_1FFF
SRPWM2	4 KB	0x4000_2000	0x4000_2FFF
SRPWM3	4 KB	0x4000_3000	0x4000_3FFF
SRPWM4	4 KB	0x4000_4000	0x4000_4FFF

DMA 地址映射			
外设模块	空间大小	起始地址	结尾地址
SRPWM5	4 KB	0x4000_5000	0x4000_5FFF
SRPWM6	4 KB	0x4000_6000	0x4000_6FFF
SRPWM7	4 KB	0x4000_7000	0x4000_7FFF
SRPWM8	4 KB	0x4000_8000	0x4000_8FFF
SRPWM9	4 KB	0x4000_9000	0x4000_9FFF
SRPWM10	4 KB	0x4000_A000	0x4000_AFFF
SRPWM11	4 KB	0x4000_B000	0x4000_BFFF
SRPWM_Com	4 KB	0x4000_C000	0x4000_CFFF
XBAR	4 KB	0x4000_D000	0x4000_DFFF
AES	4 KB	0x4000_E000	0x4000_EFFF
SRPWM_BC	4 KB	0x4000_F000	0x4000_FFFF
ETIMER	4 KB	0x4001_C000	0x4001_CFFF
DAC0	4 KB	0x4010_0000	0x4010_0FFF
DAC1	4 KB	0x4010_1000	0x4010_1FFF
CRC	4 KB	0x4010_2000	0x4010_2FFF
Dual-ACMP0	4 KB	0x4010_4000	0x4010_4FFF
Dual-ACMP1	4 KB	0x4010_5000	0x4010_5FFF
Dual-ACMP2	4 KB	0x4010_6000	0x4010_6FFF
Dual-ACMP3	4 KB	0x4010_7000	0x4010_7FFF
QEP0	4 KB	0x4010_8000	0x4010_8FFF
QEP1	4 KB	0x4010_9000	0x4010_9FFF
SDFM0	4 KB	0x4010_A000	0x4010_AFFF
SDFM1	4 KB	0x4010_B000	0x4010_BFFF
SARC0	4 KB	0x4010_C000	0x4010_CFFF
SARC1	4 KB	0x4010_D000	0x4010_DFFF
SARC2	4 KB	0x4010_E000	0x4010_EFFF
CORDIC	4 KB	0x4010_F000	0x4010_FFFF
AHB_DMA0	4 KB	0x4020_1000	0x4020_1FFF
AHB_DMA1	4 KB	0x4020_2000	0x4020_2FFF
GPIOA	4 KB	0x4020_6000	0x4020_6FFF
GPIOB	4 KB	0x4020_7000	0x4020_7FFF
GPIOC	4 KB	0x4020_8000	0x4020_8FFF
GPIOD	4 KB	0x4020_9000	0x4020_9FFF
GPIOE	4 KB	0x4020_A000	0x4020_AFFF
GPIOF	4 KB	0x4020_B000	0x4020_BFFF
FMAC	4 KB	0x4020_C000	0x4020_CFFF
SYSC	4 KB	0x4040_0000	0x4040_0FFF
WDT0	4 KB	0x4040_2000	0x4040_2FFF
WDT1	4 KB	0x4040_3000	0x4040_3FFF
STIMER64_0	4 KB	0x4040_6000	0x4040_6FFF
STIMER64_1	4 KB	0x4040_7000	0x4040_7FFF

DMA 地址映射			
外设模块	空间大小	起始地址	结尾地址
CRG	4 KB	0x4040_C000	0x4040_CFFF
SYSCNT_RO	4 KB	0x4040_D000	0x4040_DFFF
SYSCNT_CTRL	4 KB	0x4040_E000	0x4040_EFFF
IOC	4 KB	0x4040_F000	0x4040_FFFF
UART0	4 KB	0x4050_0000	0x4050_0FFF
UART1	4 KB	0x4050_1000	0x4050_1FFF
I2C0	4 KB	0x4050_4000	0x4050_4FFF
I2C1	4 KB	0x4050_5000	0x4050_5FFF
SPI_Master0	4 KB	0x4050_8000	0x4050_8FFF
SPI_Master1	4 KB	0x4050_9000	0x4050_9FFF
SPI_Master2	4 KB	0x4050_A000	0x4050_AFFF
SPI_Slave0	4 KB	0x4050_C000	0x4050_CFFF
CAN FD0	12 KB	0x4051_8000	0x4051_AFFF
CAN FD1	12 KB	0x4052_0000	0x4052_2FFF
CPU_CFG	4 KB	0x4060_0000	0x4060_0FFF
EFC0_CFG	4 KB	0x4060_2000	0x4060_2FFF
EFC1_CFG	4 KB	0x4060_3000	0x4060_3FFF
EFC2_CFG	4 KB	0x4060_4000	0x4060_4FFF
SRAM0_CFG	4 KB	0x4060_8000	0x4060_8FFF
SRAM1_CFG	4 KB	0x4060_9000	0x4060_9FFF
M7_0_Private_PERI	64 KB	N/A	N/A

说明

DMA 代表总线主设备 AXI DMA、AHB DMA0、AHB DMA1。

6.2.3 嵌入式 Flash 存储器

ET6002 集成了一个片上嵌入式程序 Flash 存储器 (PFLASH) 和一个片上嵌入式数据 Flash 存储器 (DFLASH)。PFLASH 用于存放程序代码，空间容量最大可以到 512 KB，可配置为双 Bank 256 KB，以支持 OTA。DFLASH 用于存放数据，空间容量大小为 128 KB。

PFLASH 主要特性：

- 存储容量最高可达 512 KB；
- 高达 512 个扇区，每个扇区 1024 字节；
- ECC (Error Code Correction)支持纠一检二：64 位 Flash 字增加 8 个 ECC 位；
- 支持双字、字、半字和字节读操作；
- 按 64 位进行 Flash 编程；
- 支持双字、字、半字和字节编程操作；
- 支持扇区擦除、片擦除；

- 增强安全功能：Flash Main 写保护功能 (Main WRP)；
- 双存储区构成支持同时操作：可在两个存储区中并行执行两个读取/编程/擦除操作；
- 存储区交换：每个存储区的 Flash Main 地址映射可进行交换；
- 支持总线预取功能，预取深度为 128 字节；
- 支持 Flash 低功耗模式；

DFLASH 主要特性：

- Flash Main 存储容量最高可达 128 KB；
- Flash NVR 存储容量为 6 KB；
- 高达 128 个扇区，每个扇区 1024 字节；
- ECC (Error Code Correction) 支持纠一检二：64 位 Flash 字增加 8 个 ECC 位；
- 支持双字、字、半字和字节读操作；
- 按 64 位进行 Flash 编程；
- 支持双字、字、半字和字节编程操作；
- 支持扇区擦除、片擦除；
- 增强安全功能：
  - Flash NVR 读保护功能 (RDP)；
  - Flash NVR 写保护功能 (NVR WRP)；
  - Flash Main 写保护功能 (Main WRP)；
- 支持总线预取功能，预取深度为 128 字节；
- 支持 Flash 低功耗模式；

6.2.4 片上 SRAM

ET6002 集成了多个 SRAM，用于存放程序代码和数据。

- 多达 96 KB 的系统 SRAM
- CPU 32 KB 的指令 TCM SRAM
- CPU 32 KB 的数据 TCM SRAM

TCM SRAM 专用于 Cortex®-M7，DMA 通过 Cortex®-M7 CPU 的 AHBS 接口从总线访问。

系统 SRAM 可按字节、半字 (16 位)、全字 (32 位) 或双字 (64 位) 访问。

系统 SRAM 分为 2 块，SRAM0 大小为 64 KB，SRAM1 大小为 32 KB，支持总线主设备并行访问。

对于系统 SRAM 和 TCM SRAM，ET6002 都实现了 ECC 校验特性，用以增强数据安全。ECC 功能可以修正单比特错误和检测两比特错误。当使能该特性时，一旦数据出现 1-bit 纠错或者 2-bit 检测错误，就会产生中断通知 CPU。

6.3 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 AXI SRAM 章节。

7 启动引导配置

7.1 简介

本章节主要介绍了用于系统上电引导的 BootROM 功能，启动流程、系统引导模式，以及安全启动相关内容。

芯片内部内置了一块 ROM 区域，起始地址为 0x20450000，大小 32 KB。该区域固化了 BootROM 启动引导程序。在芯片每次解复位后，都会执行 BootROM 程序，根据外部 BOOT 管脚和选项字启动配置决定后续的执行流程。

BootROM 功能特性介绍如表 7-1 所示。

表7-1 BootROM 特性

特性	CPU
初始化	硬件初始化，Trim 加载
启动模式选择	BOOT 管脚，Flash 选项字 NBOOT1
启动模式支持	• Flash 启动 • SRAM 启动 • 自举启动
自举协议支持接口	UART CAN

7.2 启动流程

芯片每次冷热复位时的标准 BootROM 启动流程：

步骤 1 CPU 解复位后，禁止所有中断源，并初始化看门狗。

步骤 2 获取系统复位原因，判断是否为冷启动。

步骤 3 如果为冷启动，加载 Trim，配置时钟，从 OSC 时钟切换到 PLL 时钟；如果为热启动，配置时钟，从 OSC 时钟切换到 PLL 时钟。

步骤 4 读取 OTP 信息，进行芯片校准配置。

步骤 5 根据启动模式判断，进入相应的启动模式。

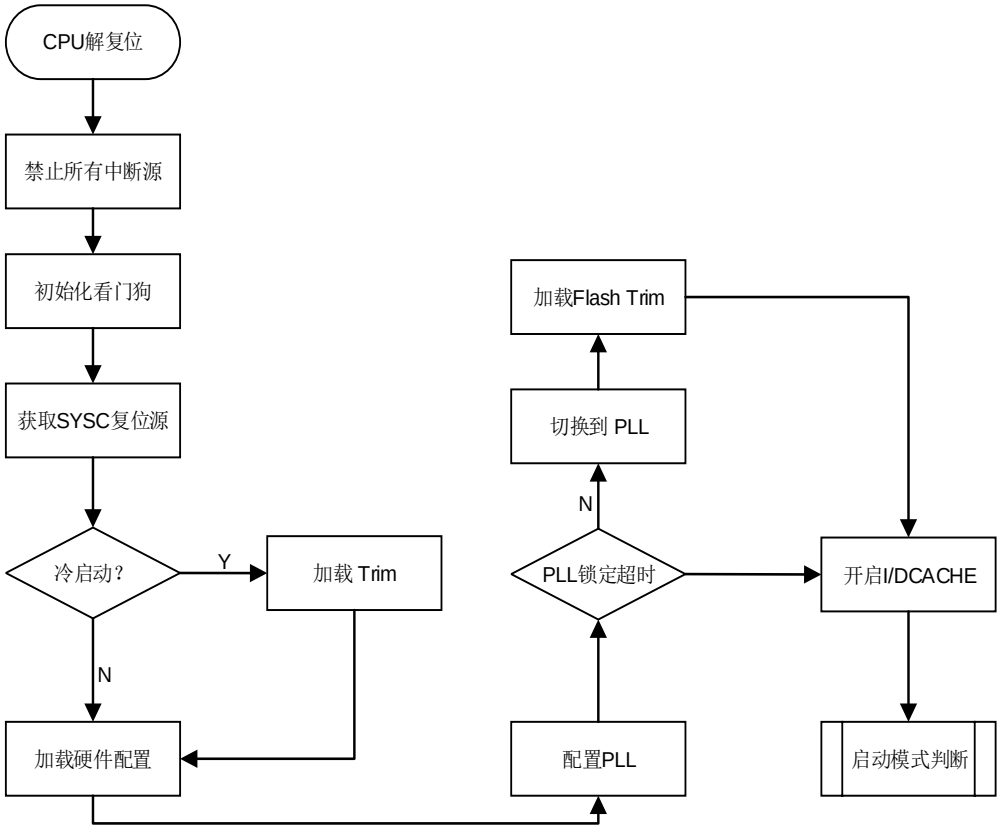


图7-1 BootROM 启动流程图

7.2.1 时钟初始化

在芯片启动过程中，BootROM 会初始化系统时钟，尝试将系统时钟从内置 OSC25M 时钟切换 PLL 时钟，如果切换成功后续会持续工作在 PLL 时钟，否则将依旧使用 OSC25M 引导应用程序，然后由应用程序切换系统时钟到 PLL 时钟。

用户可通过 CRG 模块寄存器 **CFG_PLLCLK_CFG0** 的 bit0 来判断当前系统工作在 PLL 时钟还是 OSC25M 时钟。

7.3 启动模式

本章节主要描述了 BootROM 的启动模式，功能以及如何选择启动模式。启动模式的选择由 BOOT 管脚 (GPIO24) 和 Flash 选项字 **nBOOT1** 组合共同决定。

- GPIO24 为 IO 管脚，默认外接上拉电阻，为高电平。
- **nSWBOOT1** 和 **nBOOT1** 为 Flash 选项字，出厂默认值为**'1'**。

  nSWBOOT1 为 **nBOOT1** 的使能开关：
  - 当 **nSWBOOT1** 为**'0'**时，使能 **nBOOT1**；
  - 当 **nSWBOOT1** 为**'1'**时，**nBOOT1** 值的改变不会影响启动模式的选择。

说明

选项字的设置修改，请参考 7.4 "选项字配置"章节。

启动模式的介绍，请参考章节 7.3.2 ~7.3.5 "自举启动模式"。

表7-2 BootROM 启动模式选择

GPIO24 [1]	nBOOT1	nSWBOOT1	启动模式
1	X[2]	1	正常启动 (从内置 Flash 启动)
1	0	0	安全启动 (从内置 Flash 启动)
1	1	0	从内置 SRAM 启动
0	X	1	ROM 自举启动
0	X	0	

说明

[1] 在上电锁存过程中，额外需要保持 GPIO32 为高电平。

[2] X 表示具体值是 0 或 1 都可行。

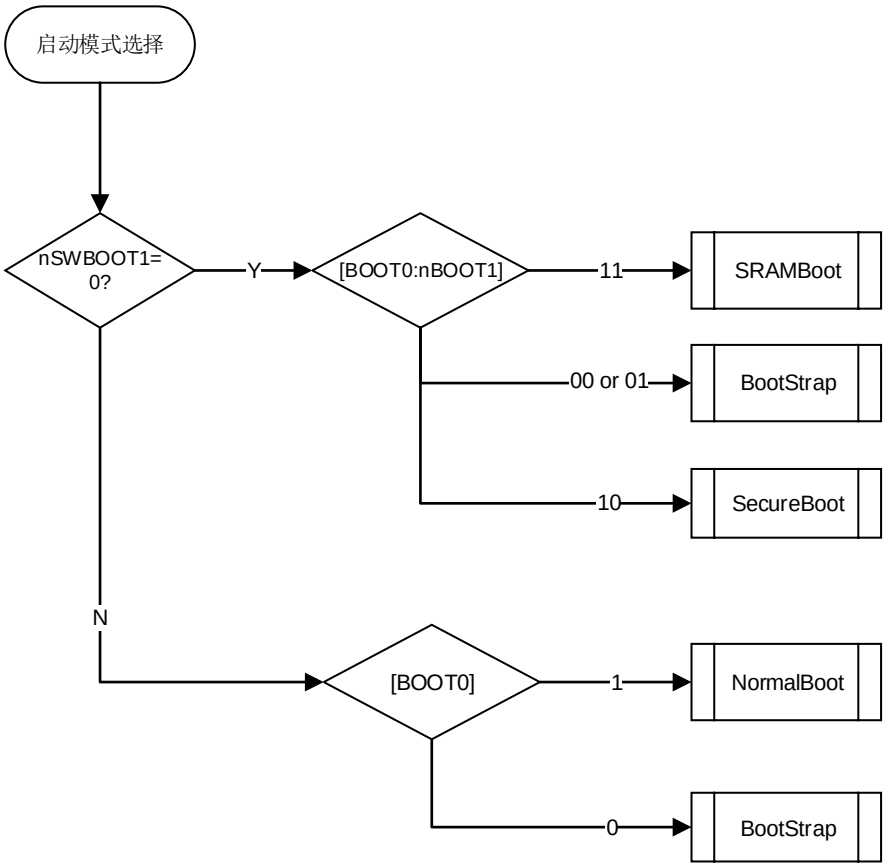


图7-2 启动模式判断

7.3.1 启动模式配置

芯片出厂默认的启动模式为正常启动模式。

若要改变默认的启动模式，请按照以下步骤执行：

步骤 1 确定目标启动模式。

如需要启动引导应用程序，则选择正常启动或者安全启动；如需要烧录应用程

序，则选择自举启动；如需要调试应用程序，则选择 SRAM 启动。

步骤 2 从当前启动模式切换到目标启动模式 (改变 IO 电平或者修改 Flash 选项字)。

步骤 3 如果目标启动模式为自举启动，则根据 CAN 或 UART 接口硬件设计所用的 IO 管脚，修改 BOOTDEF 关键字设置。

步骤 4 执行复位动作。

如果修改 GPIO24，则需要芯片 POR 复位或 HARDRST 复位，修改才生效，并且复位过程中保持 GPIO32 为高电平。

如果修改 Flash 选项字，则只需要系统复位即可生效。

步骤 5 进入目标启动模式。

7.3.2 正常启动模式

正常启动模式为默认的启动方式，该模式用于引导加载已烧录到 PFLASH 的应用程序。该模式下，引导应用程序前会根据选项字 nCRCEN、nOTAMODE 组合判断加载 A/B 分区的镜像文件。

- 若 nCRCEN 未使能，默认引导启动地址 0x08000000 起始的镜像；
- 若 nCRCEN 使能，会对 A 分区的镜像进行 CRC 校验;
  - nOTAMODE 不生效的情况下：
    - 若校验失败，则跳转到自举启动模式;
    - 若校验成功，则直接启动 A 分区镜像；
  - nOTAMODE 生效的情况下：

    若 A 分区镜像校验成功，则直接加载 A 分区镜像；

    若 A 分区镜像校验失败，则继续对 B 分区镜像进行校验；
    - 如果 B 分区镜像校验成功，则加载 B 分区镜像；
    - 如果 B 分区镜像校验失败，则跳转到自举启动模式；

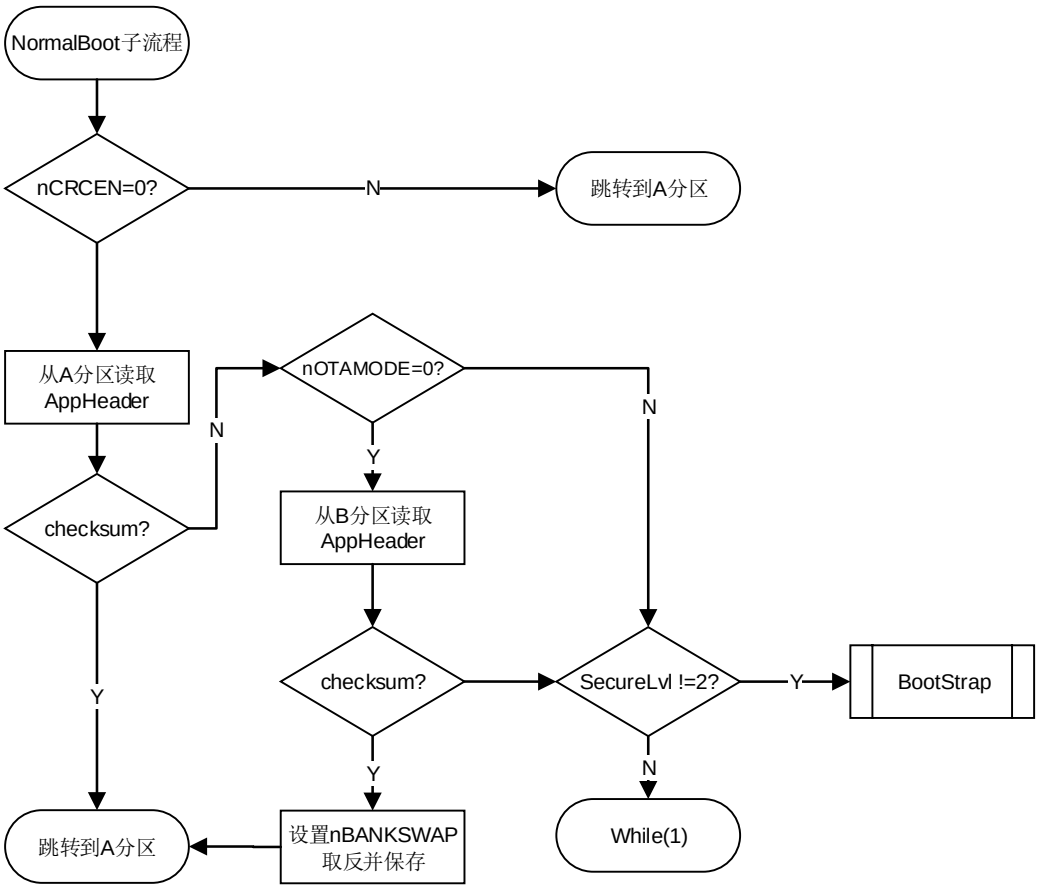
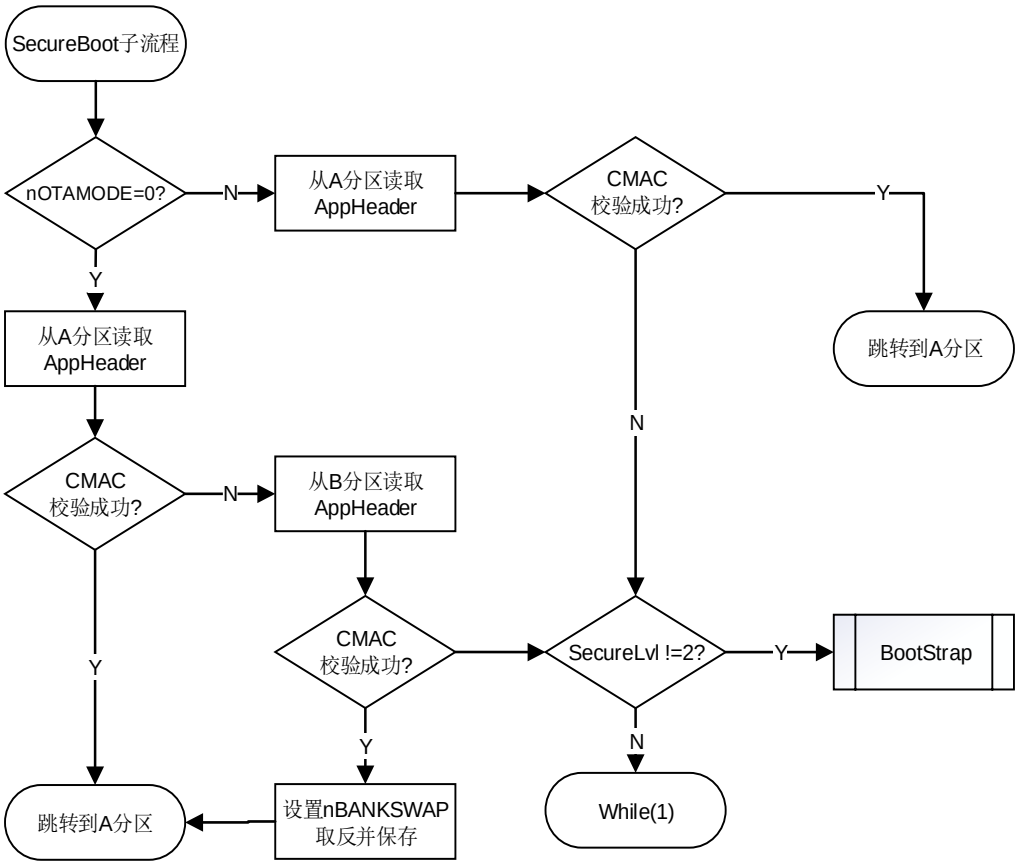


图7-3 正常启动流程图

7.3.3 安全启动模式

安全启动模式与正常启动模式流程基本相同，区别是 nCRCEN 不再生效，启动时强制对固件进行 CMAC 校验。



7.3.4 SRAM 启动模式

SRAM 启动模式为一种方便应用程序调试的启动方式，该模式下，应用程序相比 Flash 烧录要快很多，适合程序调试修改运行的场景。该模式的启动地址为 0x20450000，由 BootROM 引导启动，启动流程如下：

1. 用户修改启动模式后复位，使系统进入 SRAM 启动模式。
2. 用户使用仿真器通过 SWD 接口连接 CPU。
3. 将需要引导的程序烧写到 SRAM。
4. 通过调试软件或程序将 SRAM 程序的启动地址写入 SYSC 寄存器 BLBX0（修改该寄存器，需要先解锁 SYSC 寄存器锁 **CFG_SYSC_NV**）。
5. 用户执行系统复位或软复位。
6. 系统再次进入 SRAM 启动模式，根据步骤 4 写入的启动地址自动加载执行 SRAM 烧录的程序，并清除已写入的启动地址。

说明

写入的启动地址必须在 SRAM 地址范围内，并且是 4 字节对齐。

7.3.5 自举启动模式

进入自举启动模式后，BooROM 会按照选项字 BOOTDEF 定义的管脚初始化 UART0 和 CAN0 接口，并等待上位机握手连接。连接成功后，用户可通过上位机对芯片执行以下操作：

- 用户选项字的读取和配置
- PFLASH 的擦除操作
- DFLASH 的读写保护设置
- 固件的烧录和运行
- 安全等级的读取和设置
- 安全模式下密钥的烧录和验证
- 安全模式下固件的烧录和签名

进入自举启动模式后，UART0 和 CAN0 接口同时进行握手等待，任一接口连接成功后，另一接口则无法连接。切换接口需要对芯片进行复位操作。

7.3.5.1 自举接口参数说明

7.3.5.1.1 UART0

表7-3 UART 参数

波特率	初始 115200 bps，握手连接后可修改
数据位	8bit
校验位	Even
停止位	1

可支持的波特率 (bps)：

- 115200
- 128000
- 230400
- 256000
- 468000
- 512000
- 921600
- 1024000
- 2000000
- 3000000

7.3.5.1.2 CAN0

表7-4 CAN 参数

波特率	初始 125 kbps，握手连接后可修改
CAN/CAN FD	CAN
ID Type	STANDARD ID
Filter Type	Mask
Filter ID	0x111

可支持的波特率 (kbps)：

- 125
- 250
- 500
- 1000

7.3.5.2 自举协议支持命令

表7-5 自举协议命令

命令	说明
CMD_GET_COMMAND	获取支持的命令列表
CMD_GET_VERSION	获取系统软件版本
CMD_GET_ID	获取芯片版本
CMD_SPEED	获取和设置接口速率
CMD_READ_MEMORY	读取内存类设备数据
CMD_WRITE_MEMORY	写入内存类设备数据
CMD_GO	指定地址运行程序
CMD_SPECIAL	扩展指令
CMD_ERASE_MEMORY	Flash 擦除
CMD_VERIFY_FW	固件校验

7.3.5.3 固件格式说明

在正常启动模式下当选项字 nCRCEN 使能或者在安全启动模式下，烧录的镜像必须包含 Header Info。Header Info 包含了固件的版本信息，CRC 校验，CMAC 信息等。该信息可由上位机工具根据导入的 Raw Bin 自动生成。

图7-4 固件格式

7.4 选项字配置

芯片利用 Flash 的非易失性 Sector 为用户提供了一块可供重复配置启动相关参数的区域，称之为选项字区域。

该区域被映射到虚拟地址 0x1fff7800 开始，用户可通过支持自举协议的上位机进行选项字的读取和配置。

表7-6 用户选项字布局

映射地址	选项字	有效位	说明
0x1fff7800	nSWBOOT1	bit[0]	nBOOT1 使能开关
0x1fff7808	nBOOT1	bit[0]	启动模式决定关键字
0x1fff7810	nOTAMODE	bit[63:0]	OTA 升级使能开关
0x1fff7818	nBANKSWAP	bit[63:0]	Flash 双 Bank 切换使能开关
0x1fff7820	nCRCEN	bit[63:0]	App 引导 CRC 校验使能开关
0x1fff7828	BOOTDEF	bit[31:0]	自举接口 IO 管脚用户定义值
0x1fff7838	DFlash WRP	Bit[31:0]	DFLASH 扇区读写保护使能开关

7.4.1 选项字说明

除选项字 BOOTDEF 外，其余选项字在其值为全 0 时有效，具体有效位参考表 7-6。所有的选项字配置修改后，需要芯片复位 (任意复位源) 后生效。

7.4.1.1 nSWBOOT1

nSWBOOT1 为选项字 nBOOT1 的使能开关，只有当 nSWBOOT1 值为 **0** 时，nBOOT1 值才会影响和决定启动模式的改变。

7.4.1.2 nBOOT1

nBOOT1 为启动模式配置字，如表 7-2 启动模式选择所示，nBOOT1 和管脚 GPIO24 共同决定启动模式选择。

7.4.1.3 nCRCEN

nCRCEN 选项字用于控制 BootROM 引导应用程序前，是否对其进行 CRC 校验，如果使能该选项字，只有应用程序校验成功后，才会被加载引导。

7.4.1.4 nOTAMODE

nOTAMODE 选项字用于控制启动引导加载应用程序时，当 A 分区镜像损坏后，是否切换到 B 分区。该选项字与 nCRCEN、nBANKSWAP 配合一起使用。

7.4.1.5 nBANKSWAP

nBANKSWAP 选项字用于用户手动控制 A/B 分区 (A/B 分区的定义请参考表 7-7) 的切换，当用户修改选项字并复位芯片后，A/B 分区对应的物理分区进行互换。同时当开启 nOTAMODE 使能时，BootROM 也会自动修改该选项字的值以进行 A/B 分区交换。具体的交换流程请参考 7.3.2 "正常启动模式"章节。

表7-7 A/B 分区的定义

虚拟分区	映射地址	物理 Bank (nBANKSWAP=1)	物理 Bank (nBANKSWAP=0)
A 分区	0x08000000	Bank0	Bank1
B 分区	0x08040000	Bank1	Bank0

7.4.1.6 BOOTDEF

BOOTDEF 选项字用于配置自举接口 CAN0 和 UART0 的 IO 管脚分布，该选项字共 32bit，高 16bit 为 BOOTDEF 值的取反。BOOTDEF 低 8bit 表示 UART0 的 IOMUX 选项，高 8bit 表示 CAN 的 IOMUX 选项，每个接口最多支持 8 组 IOMUX 配置。

表7-8 选项字 BOOTDEF 布局

[31~16]	[15~8]	[7~0]
~BOOTDEF	BOOTDEF(MSB)	BOOTDEF(LSB)
BOOTDEF 值取反	CAN 接口定义	UART 接口定义

每个接口的 IO 管脚分布如下表所示：

表7-9 选项字 BOOTDEF IO 管脚定义对照表

BOOTDEF (MSB)	CANFD0_RX	CANFD0_TX	BOOTDEF (LSB)	UART0_RX	UART0_TX
0 (default)	GPIO4 (mode4)	GPIO8 (mode4)	0 (default)	GPIO3 (mode9)	GPIO2 (mode9)
1	GPIO49 (mode3)	GPIO32 (mode10)	1	GPIO50 (mode4)	GPIO24 (mode11)
2	GPIO77 (mode5)	GPIO48 (mode3)	2	GPIO28 (mode1)	GPIO48 (mode1)
3	GPIO53 (mode10)	GPIO76 (mode5)	3	GPIO49 (mode6)	GPIO51 (mode4)
4	GPIO3 (mode14)	GPIO4 (mode6)	4	GPIO9 (mode6)	GPIO2 (mode9)
5	GPIO5 (mode6)	GPIO2 (mode14)	5	GPIO17 (mode6)	GPIO29 (mode1)
6	GPIO61 (mode14)	GPIO31 (mode1)	6	GPIO25 (mode11)	GPIO16 (mode6)
7	GPIO59 (mode10)	GPIO13 (mode13)	7	GPIO35 (mode1)	GPIO37 (mode5)

7.4.1.7 DFLASH WRP

DFLASH WRP 选项字用于设置 DFLASH Main 空间的读和写保护使能。其中 Main 空间按 16 KB 粒度进行设置，每 1bit 控制 16 KB 地址空间。

表7-10 DFLASH WRP 选项字布局

[31~16]	[15~8]	[7~0]

0xFFFF	dflash_main_wrp_n	dflash_main_rdp_n

7.5 安全启动

7.5.1 安全等级

根据用户对系统资源访问的权限不同，将权限类别划分为 3 个不同的安全等级，分别为：

- Secure Level 0
- Secure Level 1

系统默认工作在 Secure Level 0 下，Secure Level 值可通过上位机工具在自举启动模式下修改。**该值为 OTP 属性，只能从低等级设置为高等级一次。**

不同安全等级下，系统资源访问权限如下：

安全等级 / 访问资源	0	1
SRAM 启动	允许	禁止
SWD 接口	允许	禁止
用户选项字写入	允许	禁止
SecureLevel 写入	允许	禁止
密钥写入	允许	禁止
安全启动模式切换其他模式	允许	需鉴权
自举模式命令支持	全部支持	支持受限
启动镜像失败后是否支持跳转到自举启动	Yes	No

修改 Secure Level 需慎重：

- **Secure Level 值只能从低等级设置为高等级 1 次，且不可逆。**
- **提升 Secure Level 等级前，请确保已经烧录了密钥。**
- **SWD 接口禁止为永久禁止，不可恢复。**
- **SWD 鉴权需要额外的 GPIO 提供鉴权接口。**

7.5.2 密钥烧录与鉴权

安全启动需要用户烧录自定义的密钥来对应用固件进行加签操作，同时 BootROM 启动时会对固件进行验签操作。密钥的烧录与验证需要在 BootROM 自举启动模式下，配合上位机工具进行。

- 密钥最多可重复烧录 8 次；
- 密钥的第 2 次烧录开始，需要提供上一次烧录的 key 进行验证；
- 密钥验证错误次数达到 50 次后，安全等级提升为 Secure Level 1；

- 密钥的存储区对用户不可以见，不可直接访问；

7.5.3 固件烧录与鉴权

安全启动加载镜像前，会对镜像的 CMAC 进行校验，只有校验通过才会加载镜像运行。

- 安全启动模式下，用户需要利用上位机对固件进行加签操作；
- 安全启动模式下，若未烧录密钥，则 BootROM 不会加载镜像启动；

7.5.4 恢复出厂设置

当 Secure Level 设置为 1，系统资源访问受限的情况下，用户希望恢复到出厂设置，可通过进入自举启动模式，利用上位机软件对系统进行恢复出厂设置。

- 执行恢复出厂设置命令需要用户提供密钥进行鉴权；
- 恢复出厂设置后，PFLASH 和 DFLASH 的 Main 区域全片擦除，选项字恢复出厂设置；

7.6 UUID

芯片内置一组唯一的设备序列号，长度为 128bit。该序列号包含了芯片生产版本信息及随机数。用户可通过访问 SYSC 寄存器组 CHIP_DIE[4]获取该序列号。

8 中断和事件

8.1 简介

Cortex-M7 功能框图如图 8-1 所示。

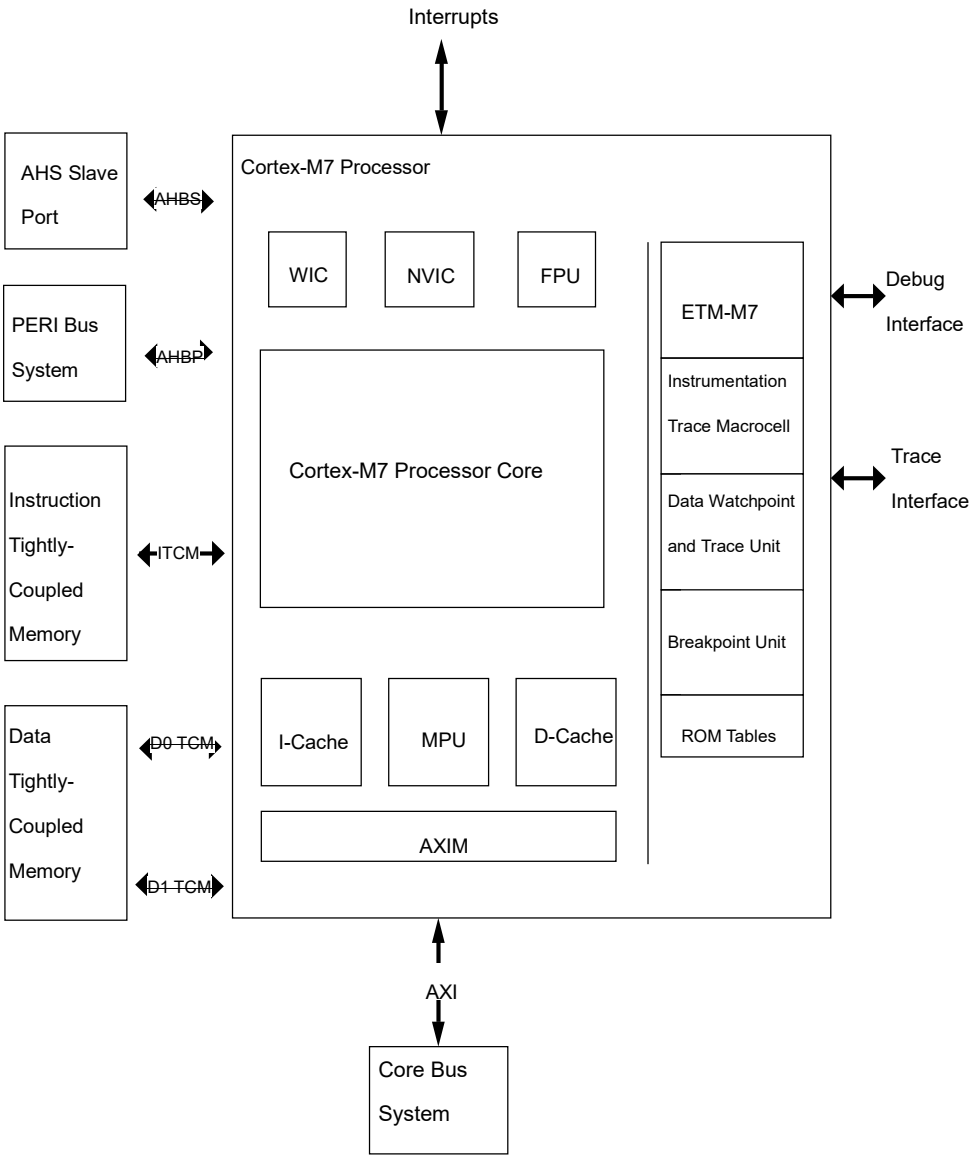


图8-1 Cortex-M7 功能框图

Cortex-M7 Core 内部自带不可屏蔽中断控制器 (NMI)，嵌入式中断控制器 (NVIC) 和唤醒中断控制器 (WIC)。

NMI 中断在 CPU_CFG 模块里可配置，能够产生只有 WDG0、WDG1 两种 NMI 中断源

的超时中断；

NVIC 与内核紧密集成，实现低延迟中断处理。功能包括：

- 支持多达 240 个外部中断，中断优先级从 8 到 256 可配。
- 动态重新确定中断的优先级。
- 支持中断优先级分组，允许选择抢占式中断优先级和非抢占式中断优先级。
- 支持中断嵌套和中断的延迟到达，可实现背靠背中断处理，而无需在中断之间进行状态保存和恢复的开销。

有关 NVIC 寄存器可访问性及其使用限制的详细信息，请参阅 ARM®v7-M 体系结构参考手册。

WIC 提供超低功耗睡眠模式支持，支持 Sleeping 和 Deepsleep 两种模式，由 CPU Core 的 SCR 寄存器定义。

唤醒中断控制器 (WIC)，这使处理器和 NVIC 能够进入非常低功耗的睡眠模式，使 WIC 能够识别中断并确定其优先级。使用 WIC 时，必须在系统控制寄存器中启用 Deepsleep。处理器充分执行等待中断 (WFI)、等待事件 (WFE) 和发送事件 (SEV) 指令。此外，处理器还支持使用 SLEEPONEXIT，这会导致处理器内核在从异常处理程序返回到线程模式时进入睡眠模式。有关详细信息，请参阅 ARM®v7-M 体系结构参考手册。

8.2 结构框图

中断结构框图如图 8-2 所示。

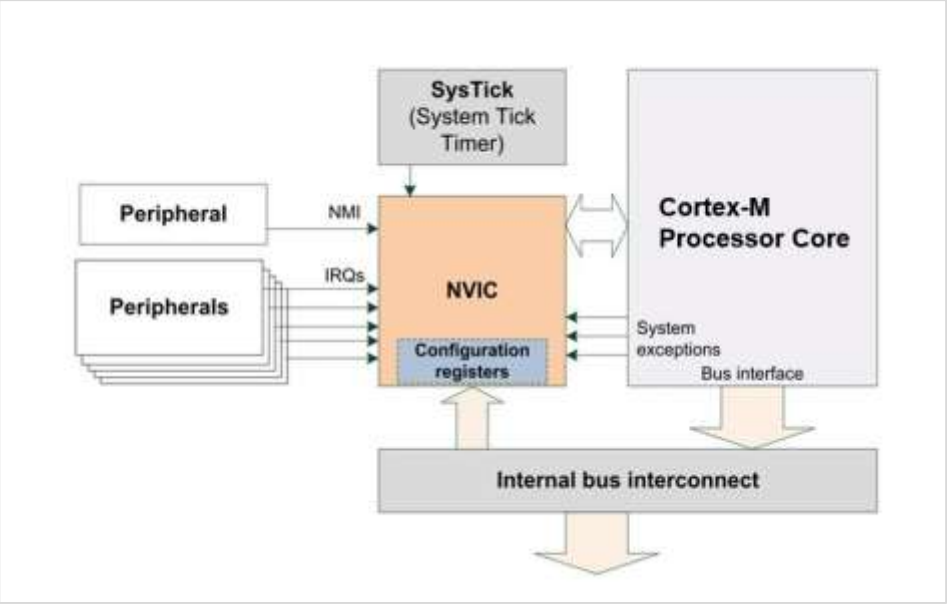


图8-2 中断结构框图

8.3 主要特性

NVIC 支持多达 240 个中断，每个中断具有多达 256 个优先级。可以改动态中断的优先级。NVIC 和处理器核心接口紧密相连耦合，实现低延迟中断处理和高效处理延迟到达的中

断。NVIC 保持对堆叠或嵌套中断的关系，以启用中断的 tail-chain 处理。

只能从特权模式完全访问 NVIC，使用字传输来访问 NVIC 中的寄存器，但可能会导致中断进入用户模式下的挂起状态 (如果启用配置和控制寄存器)。任何其他用户模式访问会导致总线故障。

NVIC 支持电平和脉冲中断，保持电平中断被置位，直到它由访问设备的 ISR 清除。脉冲中断使用时，必须确保脉冲在处理器时钟 FCLK 的上升沿被异步处理后采样。

对于电平中断，如果在从中断例程返回之前未取消置位信号，中断再次进入挂起状态并重新激活。这对于 FIFO 和基于缓冲区的设备特别有用，因为它确保它们通过单个 ISR 或重复 ISR 耗尽数据调用，无需额外工作。

可以在 ISR 期间重新置位脉冲中断，以便中断可以处于挂起状态并且同时激活。如果在中断仍处于挂起状态时另一个脉冲到达，则当前中断保持挂起状态，ISR 仅运行一次。

8.4 NMI 中断源

CPU 的 NMI 中断可以通过 CPU_CFG 模块的寄存器 **CFG_M7_NMI_SEL** 进行控制，注意配置该寄存器需要先配置 CPU_CFG 模块 **CFG_M7_KEY_CTRL.sw_access_key** 寄存器配置为 **0x7879a8a9**。

配置参数 **cfg_m7_nmi_mask[3:0]**实际只有低 2 位有效，bit1 对应 WDG1 的超时中断，bit0 对应 WDG0 的超时中断，1 表示屏蔽，0 表示允许上报。

1.11 CFG_M7_NMI_SEL				CFG_M7_NMI_SEL		Reg.	0x2C
offset		external		default	0x0000000f		
{lock=CFG_M7_KEY_CTRL.sw_access_key!=32'h7879a8a9}							

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

bits	name	s/w	h/w	default	description
11:8	nmi_src_rpt	ro	wo	0x0	当前中断屏蔽前的四路 NMI 中断源状态信号
7:4	nmi_src_masked_rpt	ro	wo	0x0	当前中断屏蔽后的四路 NMI 中断源状态信号
3:0	cfg_m7_nmi_mask	rw	ro	0xf	M7 的 NMI 中断源屏蔽控制信号; **sw_access_key** 设置为 **0x7879a8a9**，才能更新该寄存器;

8.5 NVIC IRQ 表

No.	Interrupt Source	Entry Address (Hexadecimal)	Interrupt Name	Description
0	CPU	0	ThreadMode	Thread Mode
1	CPU	4	Reserve	Reserve
2	CPU	8	NMI	NMI
3	CPU	C	HardFault	Hard Fault
4	CPU	10	MemManage	Mem Manage
5	CPU	14	BusFault	Bus Fault

No.	Interrupt Source	Entry Address (Hexadecimal)	Interrupt Name	Description
6	CPU	18	UsageFault	Usage Fault
7	CPU	1C	Reserve	Reserve
8	CPU	20	Reserve	Reserve
9	CPU	24	Reserve	Reserve
10	CPU	28	Reserve	Reserve
11	CPU	2C	SVCall	SV Call
12	CPU	30	DebugMonitor	Debug Monitor
13	CPU	34	Reserve	Reserve
14	CPU	38	PendSV	Pend SV
15	CPU	3C	SysTick	SysTick
16	Reserve	40	Reserve	Reserve
17	HDMAC0	44	hdmac0_int_flag[0]	AHB DMAC0 int_flag[0]: logical OR of all interrupts of type IntTfr
18	HDMAC0	48	hdmac0_int_flag[1]	AHB DMAC0 int_flag[1]: logical OR of all interrupts of type IntBlock
19	HDMAC0	4C	hdmac0_int_flag[2]	AHB DMAC0 int_flag[2]: logical OR of all interrupts of type IntSrcTran
20	HDMAC0	50	hdmac0_int_flag[3]	AHB DMAC0 int_flag[3]: logical OR of all interrupts of type IntDstTran
21	HDMAC0	54	hdmac0_int_flag[4]	AHB DMAC0 int_flag[4]: logical OR of all interrupts of type IntErr
22	HDMAC0	58	hdmac0_int_combined	AHB DMAC0 int_combined: logical OR of all individual interrupts
23	HDMAC1	5C	hdmac1_int_flag[0]	AHB DMAC1 int_flag[0]: logical OR of all interrupts of type IntTfr
24	HDMAC1	60	hdmac1_int_flag[1]	AHB DMAC1 int_flag[1]: logical OR of all interrupts of type IntBlock
25	HDMAC1	64	hdmac1_int_flag[2]	AHB DMAC1 int_flag[2]: logical OR of all interrupts of type IntSrcTran
26	HDMAC1	68	hdmac1_int_flag[3]	AHB DMAC1 int_flag[3]: logical OR of all interrupts of type IntDstTran

No.	Interrupt Source	Entry Address (Hexadecimal)	Interrupt Name	Description
27	HDMAC1	6C	hdmac1_int_flag[4]	AHB DMAC1 int_flag[4]: logical OR of all interrupts of type IntErr
28	HDMAC1	70	hdmac1_int_combined	AHB DMAC1 int_combined: logical OR of all individual interrupts
29	WDG0	74	wdg0_intr	WDG0 intr
30	WDG1	78	wdg1_intr	WDG1 intr
31	UART0	7C	uart0_intr	UART0 intr
32	UART1	80	uart1_intr	UART1 intr
33	I2C0	84	i2c0_ic_intr	I2C0 ic_intr
34	I2C1	88	i2c1_ic_intr	I2C1 ic_intr
35	SPI_MASTER0	8C	spim0_ssi_intr	SPI Master0 ssi_intr
36	SPI_MASTER1	90	spim1_ssi_intr	SPI Master1 ssi_intr
37	SPI_MASTER2	94	spim2_ssi_intr	SPI Master2 ssi_intr
38	SPI_SLAVE0	98	spis0_ssi_intr	SPI Slave0 ssi_intr
39	CAN0	9C	can0_int0	CAN0 can_int0
40	CAN0	A0	can0_int1	CAN0 can_int1
41	CAN0	A4	can0_dmu_int	CAN0 can_dmu_int
42	CAN1	A8	can1_int0	CAN1 can_int0
43	CAN1	AC	can1_int1	CAN1 can_int1
44	CAN1	B0	can1_dmu_int	CAN1 can_dmu_int
45	CAN	B4	mttcan_ecc_int	CAN MSGRAM ECC int
46	GPIOA	B8	gpioa_combint	GPIOA COMBINT
47	GPIOB	BC	gpiob_combint	GPIOB COMBINT
48	GPIOC	C0	gpioc_combint	GPIOC COMBINT
49	GPIOD	C4	gpiod_combint	GPIOD COMBINT
50	GPIOE	C8	gpioe_combint	GPIOE COMBINT
51	GPIOF	CC	gpiof_combint	GPIOF COMBINT
52	PBUS1_TIMEOUT	D0	pbus1_timeout	apb_bus1 timeout flag
53	PBUS2_TIMEOUT	D4	pbus2_timeout	apb_bus2 timeout flag
54	HBUS1_TIMEOUT	D8	hbus1_timeout	ahb_bus1 timeout flag
55	HBUS2_TIMEOUT	DC	hbus2_timeout	ahb_bus2 timeout flag
56	HBUS3_TIMEOUT	E0	hbus3_timeout	ahb_bus3 timeout flag
57	GPIO35	E4	gpio35_gpioint	GPIO35_GPIOINT
58	GPIO43	E8	gpio43_gpioint	GPIO43_GPIOINT

No.	Interrupt Source	Entry Address (Hexadecimal)	Interrupt Name	Description
59	GPIO56	EC	gpio56_gpioint	GPIO56_GPIOINT
60	GPIO57	F0	gpio57_gpioint	GPIO57_GPIOINT
61	GPIO58	F4	gpio58_gpioint	GPIO58_GPIOINT
62	GPIO39	F8	gpio39_gpioint	GPIO39_GPIOINT
63	GPIO42	FC	gpio42_gpioint	GPIO42_GPIOINT
64	GPIO65	100	gpio65_gpioint	GPIO65_GPIOINT
65	GPIO62	104	gpio62_gpioint	GPIO62_GPIOINT
66	GPIO83	108	gpio83_gpioint	GPIO83_GPIOINT
67	GPIO84	10C	gpio84_gpioint	GPIO84_GPIOINT
68	GPIO85	110	gpio85_gpioint	GPIO85_GPIOINT
69	GPIO71	114	gpio71_gpioint	GPIO71_GPIOINT
70	GPIO82	118	gpio82_gpioint	GPIO82_GPIOINT
71	GPIO81	11C	gpio81_gpioint	GPIO81_GPIOINT
72	GPIO45	120	gpio45_gpioint	GPIO45_GPIOINT
73	EFC0	124	efc0_flash_int	EFC0 efc_int
74	EFC1	128	efc1_flash_int	EFC1 efc_int
75	EFC2	12C	efc2_flash_int	EFC2 efc_int
76	PBUS3_TIMEOUT	130	pbus3_timeout	apb_bus3 timeout flag
77	AXI2SRAM0	134	sram0_ecc_serr_int	AXI SRAM0 1bit ECC ERROR Flag int
78	AXI2SRAM0	138	sram0_ecc_merr_int	AXI SRAM0 2bit ECC ERROR Flag int
79	AXI2SRAM1	13C	sram1_ecc_serr_int	AXI SRAM1 1bit ECC ERROR Flag int
80	AXI2SRAM1	140	sram1_ecc_merr_int	AXI SRAM1 2bit ECC ERROR Flag int
81	CORE2PERI_TMON	144	core2peri_timeout	CORE2PERI AHB TIMEOUT FLAG int
82	PERI2CORE_TMON	148	peri2core_timeout	PERI2CORE AHB TIMEOUT FLAG int
83	reserve	14C	reserve	Reserve
84	reserve	150	reserve	Reserve
85	reserve	154	reserve	Reserve
86	CPU0_M7_MISC	158	reserve	M7 WRAP misc error int
87	CMPC0	15C	cmpc_intr[0]	CMPC0 interrupt
88	CMPC1	160	cmpc_intr[1]	CMPC1 interrupt
89	CMPC2	164	cmpc_intr[2]	CMPC2 interrupt
90	CMPC3	168	cmpc_intr[3]	CMPC3 interrupt
91	DACC0	16C	dacc_intr[0]	DACC0 interrupt

No.	Interrupt Source	Entry Address (Hexadecimal)	Interrupt Name	Description
92	DACC1	170	dacc_intr[1]	DACC1 interrupt
93	SARC0	174	sarc_intr[0]	SARC0 interrupt
94	SARC1	178	sarc_intr[1]	SARC1 interrupt
95	SARC2	17C	sarc_intr[2]	SARC2 interrupt
96	EQEP0	180	eqep_intr[0]	EQEP0 interrupt
97	EQEP1	184	eqep_intr[1]	EQEP1 interrupt
98	SDFM0	188	sdfm_err_intr[0]	SDFM0 interrupt
99	SDFM0	18C	sdfm_dr_intr[0]	SDFM1 interrupt
100	SDFM0	190	sdfm_dr_intr[1]	SDFM2 interrupt
101	SDFM0	194	sdfm_dr_intr[2]	SDFM3 interrupt
102	SDFM0	198	sdfm_dr_intr[3]	SDFM4 interrupt
103	SDFM1	19C	sdfm_err_intr[1]	SDFM5 interrupt0
104	SDFM1	1A0	sdfm_dr_intr[4]	SDFM6 interrupt
105	SDFM1	1A4	sdfm_dr_intr[5]	SDFM7 interrupt
106	SDFM1	1A8	sdfm_dr_intr[6]	SDFM8 interrupt
107	SDFM1	1AC	sdfm_dr_intr[7]	SDFM9 interrupt
108	CRC	1B0	crc_intr	CRC interrupt
109	CORDIC	1B4	cordic_intr	CORDIC interrupt
110	FMAC	1B8	fmac_intr	FMAC interrupt
111	STM0	1BC	stm_intr[0]	STMER0 interrupt
112	STM1	1C0	stm_intr[1]	STMER1 interrupt
113	AES	1C4	aes_int	AES interrupt
114	ETIM	1C8	etim_intr[0]	ETIMER0 interrupt
115	ETIM	1CC	etim_intr[1]	ETIMER1 interrupt
116	ETIM	1D0	etim_intr[2]	ETIMER2 interrupt
117	ETIM	1D4	etim_intr[3]	ETIMER3 interrupt
118	ETIM	1D8	etim_intr[4]	ETIMER4 interrupt
119	ETIM	1DC	etim_intr[5]	ETIMER5 interrupt
120	ETIM	1E0	etim_intr[6]	ETIMER6 interrupt
121	ETIM	1E4	etim_intr[7]	ETIMER7 interrupt
122	ETIM	1E8	etim_intr[8]	ETIMER8 interrupt
123	ETIM	1EC	etim_intr[9]	ETIMER9 interrupt
124	ETIM	1F0	etim_intr[10]	ETIMER10 interrupt
125	ETIM	1F4	etim_intr[11]	ETIMER11 interrupt
126	ETIM	1F8	etim_intr[12]	ETIMER12 interrupt
127	ETIM	1FC	etim_intr[13]	ETIMER13 interrupt
128	MEPWM	200	epwm_epwmcom_intr[0]	SRPWM0 interrupt
129	MEPWM	204	epwm_epwmcom_intr[1]	SRPWM1 interrupt
130	MEPWM	208	epwm_epwmcom_in	SRPWM2 interrupt

No.	Interrupt Source	Entry Address (Hexadecimal)	Interrupt Name	Description
			tr[2]	
131	MEPWM	20C	epwm_epwmcom_intr[3]	SRPWM3 interrupt
132	MEPWM	210	epwm_epwmcom_intr[4]	SRPWM4 interrupt
133	MEPWM	214	epwm_epwmcom_intr[5]	SRPWM5 interrupt
134	MEPWM	218	epwm_epwmcom_intr[6]	SRPWM6 interrupt
135	MEPWM	21C	epwm_epwmcom_intr[7]	SRPWM7 interrupt
136	MEPWM	220	epwm_epwmcom_intr[8]	SRPWM8 interrupt
137	MEPWM	224	epwm_epwmcom_intr[9]	SRPWM9 interrupt
138	MEPWM	228	epwm_epwmcom_intr[10]	SRPWM10 interrupt
139	MEPWM	22C	epwm_epwmcom_intr[11]	SRPWM11 interrupt
140	SYSC	230	sysc_intr	SYSC interrupt
141	XBAR	234	xbar_intr[0]	XBAR0 interrupt
142	XBAR	238	xbar_intr[1]	XBAR1 interrupt
143	XBAR	23C	xbar_intr[2]	XBAR2 interrupt
144	XBAR	240	xbar_intr[3]	XBAR3 interrupt
145	XBAR	244	xbar_intr[4]	XBAR4 interrup

8.6 睡眠唤醒模式

CPU 可以进入低功耗睡眠模式，包括 Sleep 和 Deep-sleep 两种模式。两种模式需要通过 M7 Core 的 SCR 寄存器进行配置选择。退出低功耗睡眠模式支持 NVIC 唤醒和 WIC 唤醒。在 WIC 唤醒模式时，NVIC 中的中断都可以作为唤醒中断，另外支持 NMI，RXFE 作为 WIC 唤醒中断源。睡眠唤醒中断模式配置，参见 CPU_CFG 模块的 **CFG_M7_WIC_MODE** 寄存器说明。

1.8 CFG_M7_WIC_MODE				CFG_M7_WIC_MODE	Reg.	0x24
offset		external		default		0x00000000

{lock=CFG_M7_KEY_CTRL.sw_access_key!=32'h7879a8a9}

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

bits	name	s/w	h/w	default	description
31	cfg_m7_wicenreq	rw	ro	0x0	**sw_access_key** 设置为 **0x7879a8a9**，才能更新该寄存器;

					• 1 表示当前 Sleep 过程采用 WIC 唤醒； • 0 表示当前 Sleep 采用 NVIC 唤醒；
8	m7_wicenack_rpt	ro	wo	0x0	当前 M7 WICENACK 状态上报
4	m7_wakeup_rpt	ro	wo	0x0	当前 M7 wakeup 状态上报
3:0	m7_wicsense_merge_rpt	ro	wo	0x0	当前 WIC 唤醒的来源说明

8.6.1 SCR 寄存器定义

Bit	Name	Function
[31:5]	-	Reserved
[4]	SEVONPEND	Send Event on Pending bit: • 0: Only enabled interrupts or events can wake up the processor, and disabled interrupts are excluded. • 1: Enabled events and all interrupts, including disabled interrupts, can wake up the processor.
[3]	-	Reserved.
[2]	SLEEPDEEP	Controls whether the processor uses Sleep or Deep-sleep as its low-power mode: • 0: Sleep • 1: Deep-sleep
[1]	SLEEPONEXIT	Indicates Sleep-on-Exit when returning from Handler mode to Thread mode: • 0: Do not enter **Sleep** mode when returning to **Thread** mode. • 1: Enter **Sleep** or **Deep-sleep** mode, on return from an ISR. Setting this bit to **1** enables an interrupt driven application to avoid returning to an empty main application.
[0]	-	Reserved

8.7 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 M7 章节。

9 系统配置控制器 (SYSC)

9.1 简介

SYSC 模块主要功能包含两个主要部分：

1. 实现需要 POR 复位相关的逻辑，以及芯片其它部分的控制逻辑。
2. DMA_MUX 汇集 HAC 的 DMA 触发源以及 SoC 的 DMA 请求。

9.2 主要功能特性

主要功能如下：

1. 特定寄存器的写保护功能；
2. Boot Mode 的锁存；
3. 全局软复位的产生；
4. DAC 保持功能指示信号产生；
5. ECC 错误记录与清零；
6. 实现部分模拟器件的配置输出；
7. 实现各 Timeout、中断、ECC、时钟丢失等事件的状态及历史记录上报；
8. DMA_MUX；

9.2.1 特定寄存器的写保护

1. 特定寄存器包括：
   - CFG_SYSC_BLBX_0~15 (16 个黑匣子寄存器)
   - CFG_SYSC2CRG
   - CFG_SYSC2DACC
   - CFG_CPU_BTADDR
   - CFG_CPU_WAIT
   - CFG_SYSC_CPU0
   - CFG_SYS_SCNT
2. 当 32bit 锁寄存器 (CFG_SYSC_NV) 上电复位值为 0x0 时，以上特定寄存器默认上锁；
3. 当锁寄存器配置为 **0x12345678** 时，可解锁对特定寄存器的写操作，此时可执行写操作。

4. 特定寄存器配置完成后，当配置锁寄存器为非 **0x12345678**，可上锁以上特定寄存器，此时这些特定寄存器不允许执行写操作 (读操作不影响)，以防止电气干扰等原因造成的意外行为；

9.2.2 Boot Mode 锁存

在 POR 复位和 HWRST_n 撤销后，第 16 个时钟上升沿，对 GPIO24 进行锁存。硬件自动读取 OTP 信息 nSWBOOT1 和 nBOOT1，与 GPIO24 锁存后的值一起产生启动模式；硬件根据启动模式将 CPU 的启动地址锁存到 **CFG_CPU0_BTADDR** 寄存器。

9.2.3 全局软复位的产生

1. 32bit 复位控制寄存器 (**CFG_SYSC2CRG**)，默认值为 0x0。当配置为 **0x45670123** 时，SYSC 模块拉低全局软复位信号 (使能全局软复位)，CRG 随即执行芯片全局软复位；
2. CRG 执行芯片全局软复位后，该复位控制寄存器 (**CFG_SYSC2CRG**)又被复位成 default 值 (0x0)，此时 SYSC 模块拉高全局软复位信号 (不使能全局软复位)，指示完成一次全局软复位。

9.2.4 DAC 保持功能指示信号产生

以下是保持 DAC 输出的两种情况：

- 情况 1：CRG 送过来的 DAC 硬件保持信号，早于系统复位 16 个时钟周期，在软复位前根据 DAC 硬件保持信号及屏蔽配置 (**CFG_SYSC2DACC. hw_hold_mask**) 决定是否保持 DAC 的输出；
- 情况 2：直接通过配置寄存器 (**CFG_SYSC2DACC.sw_hold_en**) 来实现软复位期间保持 DAC 的输出；

9.2.5 ECC 错误记录与清零

各自模块 ECC 使能的情况下，对应的 ECC 处理：

1. 当 ECC 错误信息到来时，错误标记拉高，软件通过寄存器 (**ECC_INFO_SB_REC/ECC_INFO_MB_REC**) 查看；软件确认需要清除该记录时，配置清零寄存器 (**CFG_ECCCLR**) 对应 bit，硬件自动在 4 个 clk 后将清零寄存器 (**CFG_ECCCLR**) 对应 bit 拉低；
2. 当 ECC 错误信息到来时，硬件锁存 ECC 错误地址 (在软件清零前如果有多个 ECC 错误时，锁存第一个错误地址)，软件通过寄存器 (**ECC_ADDR_SB_REC~ECC_ADDR_SB_REC_15/ECC_ADDR_MB_REC~ ECC_ADDR_MB_REC_15**) 查看。

9.2.6 实现部分模拟器件的配置输出

SYSC 模块部分寄存器的功能是对模拟器件的控制。

具体寄存器以 AFE、CMU、PMU 为前缀。

9.2.7 状态及历史记录上报

SYSC 实现芯片历史记录和实时状态的上报功能。

- 历史记录包括：STIMER 紧急 counter 锁存记录、总线 timeout 状态记录、复位记录、CPU 死锁记录、ECC 错误标志和错误地址记录、时钟丢失记录、PLL 失锁记录、过压欠压记录、过温过流记录。
- 实时状态上报包括：CPU 的 WFI 状态、时钟及 PLL 状态、欠压过压告警状态、过温过流状态。

9.2.8 DMA_MUX

DMA_MUX 模块可以根据软件配置将 20 个 SoC 的 DMA 请求和 48 个 HAC DMA 请求做选择操作，分配到 8 个 AHB DMAC0 通道以及 8 个 AHB DMAC1 通道。上述共 16 个 DMA 通道的 DMA 请求选择均可支持软件独立配置，互不相关。HAC 寄存器 DMA 读写请求可支持一次触发对应多个 Burst 和 Single 的 DMA 读写流程。

9.3 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 SYSC 章节。

10 程序存储 Flash (PFLASH0/1)

10.1 简介

Flash 控制器可管理任何主设备通过 AXI 接口对 Flash 进行读取、编程操作，并实施读取和编程/擦除保护机制。

Flash 接口可管理 CPU 通过 APB 接口对 Flash Main 空间进行擦除。

10.2 结构框图

PFLASH 框图如下：

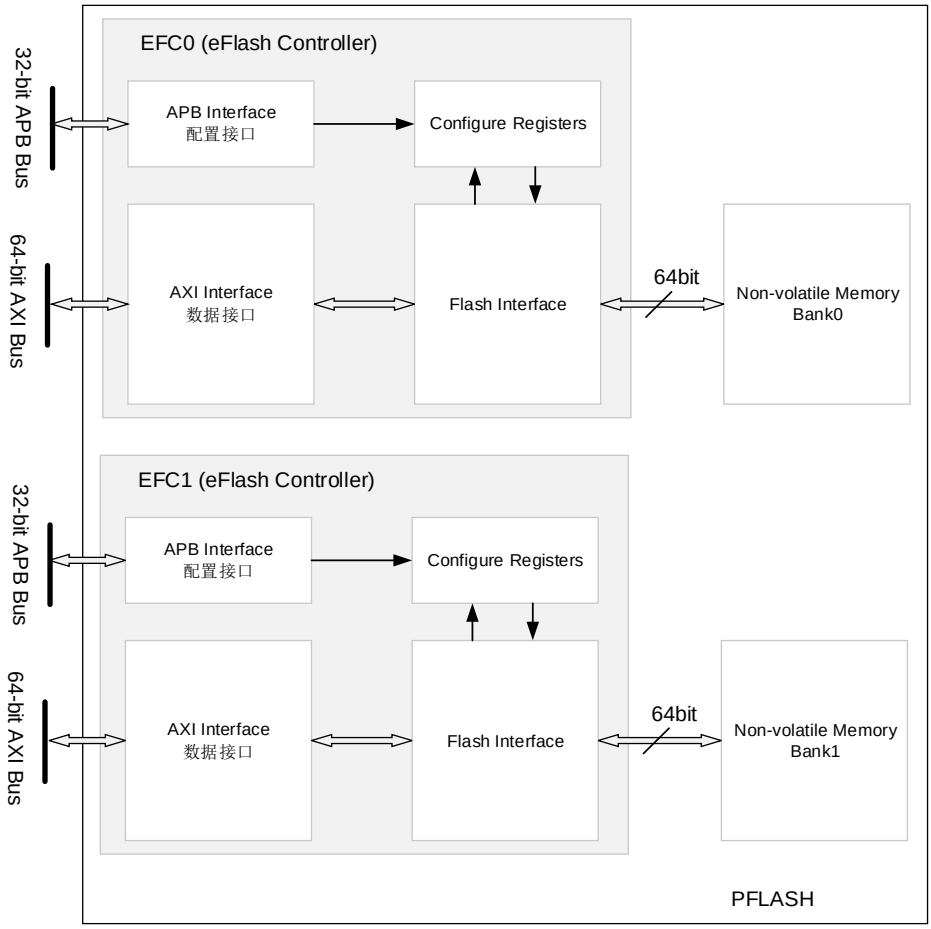


图10-1 PFLASH 框图

10.3 主要特性

PFLASH 主要特性如下：

- 存储容量最高可达 512 KB；
- 高达 512 个扇区，每个扇区 1024 字节；
- ECC (Error Code Correction) 支持纠一检二：64 位 Flash 字 8 个 ECC 位；
- 支持双字、字、半字和字节读操作；
- 按 64 位进行 Flash 编程；
- 支持双字、字、半字和字节编程操作；
- 支持扇区擦除、片擦除；
- 增强安全功能：Flash Main 写、擦除保护功能 (Main WRP)，全片安全擦除保护功能 (Secure Erase)；
- 双存储区构成支持同时操作：可在两个存储区中并行执行两个读取/编程/擦除操作；
- 存储区交换：每个存储区的 Flash Main 地址映射可进行交换；
- 支持总线预取功能，预取深度为 128 字节；
- 支持 Flash 低功耗模式;

10.4 功能描述

10.4.1 预取功能

控制器中有一个 16 深度 x 64bit 的预取 Buffer，可以提升软件的读取效率。这个预取效率提升程度取决于软件的分支数，因为该预取 Buffer 的设计是基于软件线性执行来处理的。

准静态下打开和关闭预取 Buffer 功能：

1. 打开预取 Buffer 功能，需要设备上电后由软件开启；
2. 关闭预取 Buffer 功能，需要保证没有 Flash 操作后，通过系统软复位后方可生效；

10.4.2 Flash 操作

执行任何 Flash 操作 (含读/编程/擦除) 时，都需要先配置好 Flash 时钟，且根据 Flash 的工作时钟频率配置对应的 Timing 参数，如果参数配置不正确，可能导致读/编程/擦除操作结果不正确。

如果在 Flash 进行编程/擦除操作期间发生器件复位，无法确定 Flash 中的数据内容。

在对 Flash 进行编程/擦除操作期间，总线上的 Flash 读取动作会被暂停。只有完成 Flash 编程/擦除操作后，才会正确处理读操作。

从 Flash 接口看，所有操作均为串行执行，上一组操作完成后才会开始下一组操作。

10.4.2.1 Flash 解锁

PFLASH 中用于解锁寄存器的 Key 值为：

- **REG_KEY1 = 0x01234567**
- **REG_KEY2 = 0x89ABCDEF**

◆ **Flash 的寄存器控制解锁**

复位后，Flash 控制器的寄存器都不允许执行写操作 (读操作不需要解锁)，以防因电气干扰等原因出现对 Flash 的意外操作。

- 对 Flash 的擦除操作是通过写寄存器实现的，因此需要先对寄存器进行解锁。
- 对 Flash 的擦除、编程操作前，需要将 **cfg_efc_write_en** 配置为 **1**，因此需要先对寄存器进行解锁；

  总之，当需要执行 Flash 编程/擦除操作时，必须先执行解锁，再发起相应的操作。

  如下描述 Flash 控制寄存器的解锁和加锁：

➢ 解锁方式：

向 **CFG_EFC_REG_KEY1** 寄存器写入 **REG_KEY1**，再向 **CFG_EFC_REG_KEY2** 寄存器写入 **REG_KEY2**，成功解锁后，**CFG_EFC_WPFLG** 寄存器中 **cfg_efc_reg_wrprot_flg** 会自动清零，这时即可对 Flash 进行编程/擦除操作。

➢ 加锁方式：

完成所需的编程/擦除操作后，通过软件将 **CFG_EFC_WPFLG** 寄存器中 **cfg_efc_reg_wrprot_flg** 设置为 **1**，将重新加锁寄存器，实现加锁功能。下次再进行编程/擦除操作前，需要重新进行解锁操作。

说明

- 进行编程/擦除操作前，必须对 Flash 进行解锁，否则将导致编程/擦除操作无效。
- 解锁过程必须严格按照正确的解锁顺序进行，否则将导致解锁失败。如果解锁失败，即使按照正常解锁顺序再进行一次解锁，同样也不会解锁成功，除非芯片进行系统复位。
- 建议在完成目标操作后，对 Flash 进行加锁，避免意外操作。
- 对 Flash Main 空间的读操作，不需要解锁，可直接读取。

10.4.2.2 Flash Timing 配置

PFLASH 没有单独的 Timing 配置，共用 DFLASH 的 Timing 参数。

表10-1 Flash 颗粒工作参数

处理内容花费时间	最小值	典型值	最大值	单位
64-bit 编程时间	36	40	50	us
扇区擦除时间	8	9	20	ms
片擦除时间	8	9	20	ms

10.4.2.3 Flash 擦除

Flash 擦除操作可针对扇区擦除或整片擦除执行。

Flash 寄存器解锁后，将 **cfg_efc_write_en** 配置为 **1**，通过写寄存器可以对 Flash 进行擦除。

➢ **扇区擦除**

扇区擦除的具体步骤如下：

1. 检查 **CFG_EFC_IND_STS** 寄存器的 **cfg_efc_indirect_sts** 是否为 **0** (正在操作)，如果是，则等待上一组操作完成。
2. 设置 **CFG_EFC_IND_CMD** 寄存器：
   (1) **cfg_efc_indcmd_subtype** 设置为 3 (表示 Main)；
   (2) **cfg_efc_indcmd_type** 设置为 1 (表示 sec 擦除) 或 5 (表示 retry 擦除)；
   (3) **cfg_efc_indcmd_sel** 设置为目标扇区地址 (8byte 为单位)；
3. 检查 **CFG_EFC_IND_STS** 寄存器的 **cfg_efc_indirect_sts** 是否为 **0** (表示正在操作)，如果是，则等待操作完成。

说明

如果上一组操作未完成，发起一次扇区擦除，则本次操作会被屏蔽。

Retry 擦除和 sector 擦除耗时差别如下：

Sector 擦除	每次耗时 9 ms 左右	完全擦除需耗时 9 ms 左右。
Retry 擦除	每次耗时 0.9 ms 左右，retry 次数可能 1~20 次(硬件自动执行判断)	完全擦除需耗时 0.9 ms~18 ms。

➢ **整片擦除**

整片擦除的具体步骤如下：

1. 检查 **CFG_EFC_IND_STS** 寄存器 **cfg_efc_indirect_sts** 是否为 **0** (正在操作)，如果是，则等待上一组操作完成。
2. 设置 **CFG_EFC_IND_CMD** 寄存器：
   (1) **cfg_efc_indcmd_subtype** 设置为 **3** (表示 Main) 或 **4** (表示 Main+RDN)；
   (2) **cfg_efc_indcmd_type** 设置为 **2** (表示整片擦除)；
   (3) **cfg_efc_indcmd_sel** 设置为 **0**；
3. 检查 **CFG_EFC_IND_STS** 寄存器 **cfg_efc_indirect_sts** 是否为 **0** (表示正在操作)，如果是，则等待操作完成。

说明

如果上一组操作未完成，发起一次整片擦除，则本次操作会被屏蔽。

10.4.2.4 Flash 编程

Flash 寄存器解锁后，将 **cfg_efc_write_en** 配置为 **1**，AXI 总线上各 Master (CPU/DMA 等) 即可对 Flash 进行编程操作。

说明

把 Flash 单元从逻辑“1”编程为逻辑“0”时，可以无需执行擦除操作即可进行写操作，这样可以减少 Flash 擦写次数。但把 Flash 单元从逻辑“0”写为逻辑“1”时，则必须先执行 Flash 擦除操作。

Flash 擦除操作从 APB 接口输入，Flash 编程操作从 AXI 接口输入。接口处命令可同时发起，但 Flash 会先根据优先级仲裁，然后根据仲裁优先级串行执行。

Flash 的编程，必须调用 SDK 中对应的编程函数才可正常编程。

10.4.2.5 Flash 读取

不需要额外的动作，AXI 总线上各 Master (CPU/DMA 等) 即可对 Flash 进行读取操作。

10.4.2.6 Flash ECC 校验

由于 Flash 存储数据受电子干扰或其他因素导致数据错误，会导致系统工作异常。为了提高系统稳定性及芯片可靠性，对 Flash 加入 ECC 校验功能，采用 8bit 纠 64bit 的 ECC 策略，可以实现 64bit 纠 1bit，检查 2bit。

该功能通过寄存器 **CFG_EFC_ECC** 中 **cfg_efc_main_ecc_en** 配置打开或关闭。

在对 Flash 发起编程操作时，ECC 功能模块会自动对 64bit 数据进行 ECC 计算，生成 ECC 校验码，同时将编程数据和 ECC 校验码写入 Flash 中。

当进行读操作时，将数据和 ECC 校验码同时读回，再进行 ECC 校验。如果发生 1bit 错误，会自动将该 bit 纠正，同时产生 ECC 1bit 错误中断上报 (如果对应中断使能打开)；如果发生 2bit 错误，无法纠正数据错误，会产生 ECC 2bit 错误中断上报 (如果对应中断使能打开)。

10.4.2.7 Flash 低功耗

当软件判断 Flash 不再使用时，可配置 Flash 为 Deep Power-Down (DPD) 模式 (低功耗模式)，同时关闭 Flash 控制器的时钟。

Flash 低功耗配置的具体步骤如下：

1. 检查 **CFG_EFC_IND_STS** 寄存器 **cfg_efc_indirect_sts** 是否为 **0** (表示正在操作)，如果是，则等待上一组操作完成。
2. 判断 AXI 总线对 Flash 的读取/编程操作已完成，且后续不会再有读取/编程操作。
3. 读取寄存器 **CFG_EFC_DBG_STS**，判断是否为 0x100 (表示空闲状态)，如果不是，则等待 Flash 操作完全空闲。
4. 配置 SYSC 寄存器 **CFG_EFC_DPD**，将不使用的 Flash 对应 bit 配置为 1，让其进入低功耗模式。
5. 检查 **CFG_EFC_DEBUG** 寄存器 **cfg_efc_fctrl_pwrst_rpt[1:0]**是否为 2’b11 (表示 DPD 状态)，如果不是，则等待进入该状态。
6. 配置 CRG 寄存器 **CFG_CRG_GTEN0** 寄存器 **cfg_crg_efc_clk_req[2:0]**中不使用的 Flash 对应 bit 配置为 0，将对应 Flash 控制器的时钟关断。

7. 判断 Flash 需要重新打开。
8. 配置 CRG 寄存器 **CFG_CRG_GTEN0** 寄存器 **cfg_crg_efc_clk_req[2:0]**中待使用的 Flash 对应 bit 配置为 1，将对应 Flash 控制器的时钟打开。
9. 配置 SYSC 寄存器 **CFG_EFC_DPD**，将不使用的 Flash 对应 bit 配置为 0，让其从低功耗模式进入工作模式。
10. 检查 **CFG_EFC_DEBUG** 寄存器 **cfg_efc_fctrl_pwrst_rpt[1:0]**是否为 2'b10 (表示工作状态)，如果不是，则等待进入该状态。

检查 Flash 是否成功进入工作状态。如果是，Flash 可正常工作。

10.4.2.8 Flash 中断和事件

以下是 Flash 中断和事件产生的错误标志:

- **擦除完成 (erase_done)**

  Flash Main 空间的扇区擦除或片擦除完成时，上报中断；

- **Retry 擦除完成 (retry_done)**

  Flash Main 空间的 sector retry 擦除完成时，上报中断；

- **编程顺序错误 (pgs_err)**

  当编程顺序不正确时，该信号置 1。满足以下条件时，即表示编程顺序不正确：

  1. 数据总线发出了写请求，但 **cfg_efc_write_en** 并没有置 1；
  2. 不一致错误标记还未清零，又发出了新的编程/擦除操作；
  3. 数据接口 Flash Main ECC 2bit 错误标记还未清零，又发出了新的编程/擦除操作；

  该状态通过将对应的错误指示清零信号置为 1 而清零；

  该状态必须清零才能执行新的编程/擦除操作，否则会产生编程顺序错误，下一次编程/擦除操作也会被终止 (配置接口和数据接口相同)。

- **选通错误 (strb_err)**

  当数据接口连续 2 次以上向同一个地址编程同一字节时，该信号置 1。当信号置 1 后，编程动作不会终止，应用程序可忽略该错误，继续执行当前编程操作，并可以继续执行新的编程/擦除操作；

  该状态通过将对应的错误指示清零信号置为 1 而清零；

  该状态不须清零也可以执行新的编程/擦除操作；

- **不一致错误 (inc_err)**

  当配置接口上一组命令还未执行完成，配置接口再次发出新的命令，就发生不一致错误，并且新的命令不会被执行；

  该状态通过将对应的错误指示清零信号置为 1 而清零；

  对于配置接口，该状态必须清零才能执行新的编程/擦除操作，否则会产生编程顺序错误，下一次编程/擦除操作也会被终止；

  数据接口不受该错误影响；

- **Main ECC 1bit 错误 (main_ecc1bit_err)**

  数据接口读取 Main 时，发生 ECC 1bit 错误，并纠错，该信号置 1。当信号置 1 后，返回的读取数据正确，应用程序可忽略该错误，继续执行当前读操作，以及下一步操作 (不需要对当前操作进行 retry 处理)；

  该状态通过将 ECC 错误指示清零信号置为 1 而清零；

  该状态不须清零也可以执行新的读/编程/擦除操作；

- **Main ECC 2bit 错误 (main_ecc2bit_err)**

  数据接口读取 Main 时，发生 ECC 2bit 错误，该信号置 1。当信号置 1 后，返回的读取数据不正确，应用程序无法忽略该错误，需要对当前操作进行 retry 处理或其他动作；

  发生此错误时上报中断，总线数据返回 0；

  该状态通过将 ECC 错误指示清零信号置为 1 而清零；

  对于配置接口，该状态必须清零才能执行新的擦除操作，否则会产生编程顺序错误，下一次擦除操作也会被终止；

  对于数据接口，该状态必须清零才能执行新的编程/擦除操作，否则会产生编程顺序错误，下一次编程/擦除操作也会被终止；在该错误未清零前，新的读操作也不再执行；

- **Boot Timeout 错误 (boot_timeout)**

  Boot 过程中，如果 Flash 初始化超时，发生 Boot Timeout 错误。

- **Flash 门控时 APB 访问错误 (gating_apberr)**

  当 Flash 在低功耗状态时，配置接口上还有数据交互未完成或来了新的总线命令，发生访问错误；此时上报中断，总线数据返回 0；

  该状态不须清零也可以执行新的读/编程/擦除操作；

- **Flash 门控时 AXI 访问错误 (gating_axierr)**

  当 Flash 在低功耗状态时，数据接口上还有数据交互未完成或来了新的总线命令，总线保护模块将总线未完成的交互模拟完成，避免总线挂死；

  此时上报中断，总线数据返回 0；

  该状态不须清零也可以执行新的读/编程/擦除操作；

- **数据位宽访问错误 (dw_err)**

  当对 Flash 操作的 ECC 功能打开时，AXI 总线上出现非 64bit 操作，因为 Flash 特性无法再次计算 ECC 值并写入 Flash，则上报数据位宽访问错误。

10.4.3 Flash 空间

配置 SYSC 寄存器 **cfg_sysc_otamode/cfg_sysc_bankswap**：

- DFLASH OB 空间 addr=='7170 (byte 为单位) 中：

  bit[7:0] = sysc_otamode；(单 bit 域段，用 8bit 表示)

- DFLASH OB 空间 addr=='7171 (byte 为单位) 中：

bit[7:0] = sysc_bankswap；(单 bit 域段，用 8bit 表示)

说明

在 BootROM 中，会自动将 NVR 空间的这两个配置读取后，配置到 SYSC 寄存器中，软件可查看。

软件需要对场景发生改变时 (如在线升级)，先写 NVR 空间中这两个配置数值，再软件复位系统，BootROM 启动后即完成场景的切换。

Flash 空间根据以上两个配置，有以下几种使用场景：

cfg_sysc_otamode	cfg_sysc_bankswap	Flash Memory	start_addr	end_addr	场景描述
0	0	EFC0	0x0800_0000	0x0803_FFFF	总共用作 512 KB 空间
		EFC1	0x0804_0000	0x0807_FFFF	
0	1	EFC0	0x0804_0000	0x0807_FFFF	总共用作 512 KB 空间，两块 PFLASH 空间地址交换
		EFC1	0x0800_0000	0x0803_FFFF	
1	0	EFC0	0x0800_0000	0x0803_FFFF	在线升级模式，当前使用 EFC0，升级空间为 EFC1。
		EFC1	0x0804_0000	0x0807_FFFF	
1	1	EFC0	0x0800_0000	0x0803_FFFF	在线升级模式，当前使用 EFC1，升级空间为 EFC0。
		EFC1	0x0804_0000	0x0807_FFFF	

10.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 PFLASH 章节。

11 数据存储 Flash (DFLASH)

11.1 简介

Flash 控制器可管理任何主设备通过 AXI 接口对 Flash 进行读取、编程操作，并实施读取和编程/擦除保护机制。

Flash 接口可管理 CPU 通过 APB 接口对 Flash Main 空间进行擦除。

11.2 结构框图

DFLASH 框图如下：

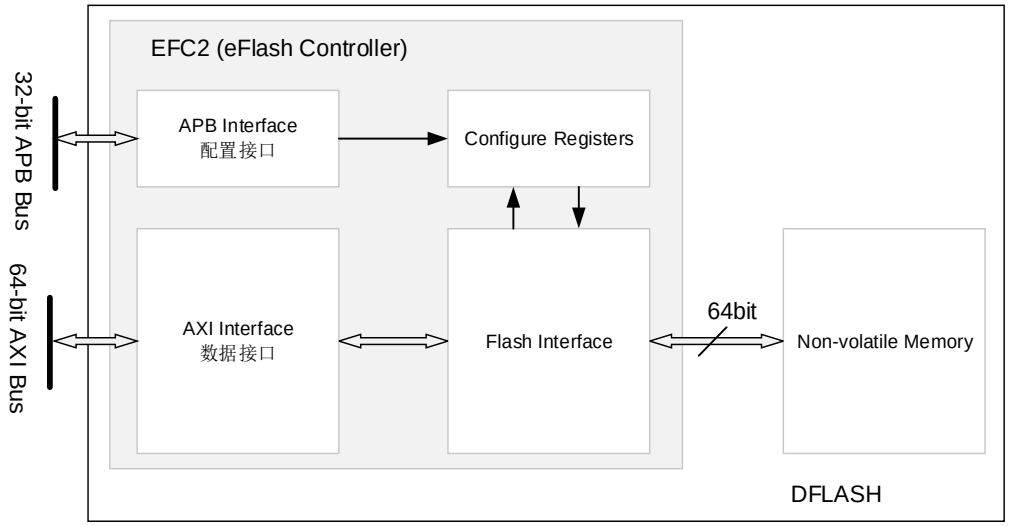


图11-1 DFLASH 框图

11.3 主要特性

DFLASH 主要特性如下：

- Flash Main 存储容量最高可达 128 KB；
- 高达 128 个扇区，每个扇区 1024 字节；
- ECC (Error Code Correction) 支持纠一检二：64 位 Flash 字 8 个 ECC 位；
- 支持双字、字、半字和字节读操作；
- 按 64 位进行 Flash 编程；
- 支持双字、字、半字和字节编程操作；
- 支持扇区擦除、片擦除；

- 增强安全功能：
  - Flash Main 读保护功能 (Main RDP)；
  - Flash Main 写/擦除保护功能 (Main WRP)；
  - 全片安全擦除保护功能 (Secure Erase)；
- 支持总线预取功能，预取深度为 128 字节；
- 支持 Flash 低功耗模式；

11.4 功能描述

11.4.1 预取功能

控制器中有一个 16 深度 x 64bit 的预取 Buffer，可以提升数据的读取效率。这个预取效率提升程度取决于数据的连续性，因为该预取 Buffer 的设计是基于连续读取来处理的。

准静态下打开和关闭预取 Buffer 功能：

- 打开预取 Buffer 功能，需要设备上电后由软件开启。
- 关闭预取 Buffer 功能，需要保证没有 Flash 操作后，通过系统软复位后方可生效。

11.4.2 Flash 操作

执行任何 Flash 操作 (含读/编程/擦除) 时，都需要先配置好 Flash 的时钟，且根据 Flash 的工作时钟频率配置对应的 timing 参数，如果参数配置不正确，可能导致读/编程/擦除操作结果不正确。

如果在 Flash 进行编程/擦除操作期间发生器件复位，无法保证 Flash 中的内容。

在对 Flash 进行编程/擦除操作期间，总线上的 Flash 读取动作会被 hold 住。只有完成 Flash 编程/擦除操作后，才会正确处理读操作。

从 Flash 接口看，所有操作均为串行执行，上一组操作完成后才会开始下一组操作；

11.4.2.1 Flash 解锁

DFLASH 中用于解锁寄存器的 Key 值为：

REG_KEY1 = 0x0123CDEF

REG_KEY2 = 0x89AB4567

NVR_KEY1 = 0xCDEF0123

NVR_KEY2 = 0x456789AB

- **Flash 的寄存器控制解锁**

  复位后，Flash 控制器的寄存器都不允许执行写操作 (读操作不需要解锁)，以防因电气干扰等原因出现对 Flash 的意外操作。

  - 对 Flash NVR 的读/擦除/编程操作是通过写寄存器实现的，因此需要先对寄存器进行解锁。

- 对 Flash Main 的擦除操作为写寄存器的间接访问，因此需要先对寄存器进行解锁；然后将需要操作的 sector 对应的 unlock 标记更新为 1 (寄存器为 **CFG_EFC_UNLOCK**)。
- 对 Flash Main 的编程操作前，需要将 **cfg_efc_write_en** 配置为 **1**，然后将需要操作的 sector 对应的 unlock 标记更新为 1 (寄存器为 **CFG_EFC_UNLOCK**)，因此需要先对寄存器进行解锁。

总之，当需要执行 Flash 操作时，必须先执行解锁，再发起相应的操作。

如下描述 Flash 控制寄存器的解锁和加锁：

➢ 解锁方式：

向 **CFG_EFC_REG_KEY1** 寄存器写入 **REG_KEY1**，再向 **CFG_EFC_REG_KEY2** 寄存器写入 **REG_KEY2**，成功解锁后，**CFG_EFC_WPFLG**寄存器中**cfg_efc_reg_wrprot_flg**会自动清零，这时即可对Flash进行编程/擦除操作；

➢ 加锁方式：

完成所需的编程/擦除操作后，通过软件将**CFG_EFC_WPFLG**寄存器中**cfg_efc_reg_wrprot_flg**设置为**1**，将重新加锁寄存器，实现加锁功能。下次再进行编程/擦除操作前，需要重新进行解锁操作。

说明

- 进行编程/擦除操作前，必须进行解锁，否则将导致编程/擦除操作无效。
- 解锁过程必须严格按照正确的解锁顺序，否则将导致解锁失败。如果解锁失败，即使按照正常解锁顺序再进行一次解锁，同样也不会解锁成功，只有对芯片系统复位才能重新解锁。
- 建议在完成目标操作后，对 Flash 进行加锁，避免意外操作。
- 对 Flash Main 空间的读操作，不需要解锁，可直接读取。

● **Flash 的 NVR 空间控制解锁**

对 Flash NVR 的读/编程/擦除操作，需要写入特定的密钥进行操作解锁，避免意外的读/编程/擦除保护产生。

➢ 解锁方式：

寄存器解锁后，向**CFG_EFC_NVR_KEY1**寄存器写入**NVR_KEY1**，再向**CFG_EFC_NVR_KEY2** 寄存器写入 **NVR_KEY2**，成功解锁后，**CFG_EFC_WPFLG**寄存器中**cfg_efc_nvr_wrprot_flg**会自动清零，这时即可对Flash NVR空间进行读/编程/擦除操作；

➢ 加锁方式：

完成所需的编程/擦除操作后，通过软件将**CFG_EFC_WPFLG**寄存器中**cfg_efc_nvr_wrprot_flg**设置为**1**，将重新加锁寄存器，实现加锁功能。下次再进行读/编程/擦除操作前，需要重新进行解锁操作。

说明

- 对 Flash NVR 的读/编程操作为写寄存器的间接访问，在该操作前需要先进行寄存器解锁。
- 进行 Flash NVR 空间读/编程/擦除操作前，必须进行解锁，否则将导致读/编程/擦除操作无

效。

- 解锁过程必须严格按照正确的解锁顺序，否则将导致解锁失败。如果解锁失败，即使按照正常解锁顺序再进行一次解锁，同样也不会解锁成功，只有对芯片系统复位才能重新解锁。

建议在完成目标操作后，对 Flash 进行加锁，避免意外操作。

11.4.2.2 Flash Timing 配置

表 11-1 列出了 Flash 操作相关参数。

表11-1 Flash 操作相关参数

Flash 操作时间	最小值	典型值	最大值	单位
64-bit 编程时间	36	40	50	us
扇区擦除时间	8	9	20	ms
片擦除时间	8	9	20	ms

11.4.2.3 Flash 擦除

Flash Main Sector 写保护解除后，再解锁 Flash 寄存器，Flash 擦除操作可针对扇区擦除或整片擦除执行。

➢ **扇区擦除**

扇区擦除的具体步骤如下：

1. 检查 **CFG_EFC_IND_STS** 寄存器 **cfg_efc_indirect_sts** 是否为 **0** (表示正在操作)，如果是，则等待上一组操作完成。
2. 设置 **CFG_EFC_IND_CMD** 寄存器：
   (1) **cfg_efc_indcmd_subtype** 设置为 1 (表示 NVR) 或 3 (表示 Main)；
   (2) **cfg_efc_indcmd_type** 设置为 1 (表示 sec 擦除) 或 5 (表示 retry 擦除)；
   (3) **cfg_efc_indcmd_sel** 设置为目标扇区地址 (8byte 为单位)；
3. 检查 **CFG_EFC_IND_STS** 寄存器 **cfg_efc_indirect_sts** 是否为 **0** (表示正在操作)，如果是，则等待操作完成。

说明

如果上一组操作未完成，发起一次扇区擦除，则本次操作会被屏蔽。

Retry 擦除和 sector 擦除耗时差别如下：

Sector 擦除	每次耗时 9 ms 左右	完全擦除需耗时 9 ms 左右。
Retry 擦除	每次耗时 0.9 ms 左右，retry 次数可能 1~20 次 (硬件自动执行判断)	完全擦除需耗时 0.9 ms~18 ms。

➢ **整片擦除**

整片擦除的具体步骤如下：

1. 检查 **CFG_EFC_IND_STS** 寄存器 **cfg_efc_indirect_sts** 是否为 **0** (表示正在操作)，

如果是，则等待上一组操作完成。

2. 设置 **CFG_EFC_IND_CMD** 寄存器：

   (1) **cfg_efc_indcmd_subtype** 设置为 3 (表示 Main) 或 4 (表示 Main+RDN)；

   (2) **cfg_efc_indcmd_type** 设置为 2 (表示 chip 擦除)；

   (3) **cfg_efc_indcmd_sel** 设置为 0；

3. 检查 **CFG_EFC_IND_STS** 寄存器 **cfg_efc_indirect_sts** 是否为 **0** (表示正在操作)，如果是，则等待操作完成。

说明

如果上一组操作未完成，发起一次整片擦除，则本次操作会被屏蔽

11.4.2.4 Flash 编程

Flash Main sector 写保护解除后，AXI 总线上各 Master (CPU/DMA 等) 即可对 Flash 进行编程操作。

说明

把 Flash 单元从逻辑“1”编程为逻辑“0”时，可以无需执行擦除操作即可进行写操作，这样可以减少 Flash 擦写次数。但把 Flash 单元从逻辑“0”写为逻辑“1”时，则必须先执行 Flash 擦除操作。

Flash 擦除操作从 APB 接口输入，Flash 编程操作从 AXI 接口输入。接口处命令可同时发起，但 Flash 会先根据优先级仲裁，然后根据仲裁优先级串行执行。

Flash 的编程，必须调用 SDK 中对应的编程函数才可正常编程。

11.4.2.5 Flash 读取

Flash Main sector 读保护解除后，AXI 总线上各 Master (CPU/DMA 等) 即可对 Flash 进行读操作。

11.4.2.6 Flash ECC 校验

由于 Flash 存储数据受电子干扰或其他因素导致数据错误，会导致系统工作异常。为了提高系统稳定性及芯片可靠性，对 Flash 加入 ECC 校验功能，采用 8bit 纠 64bit 的 ECC 策略，可以实现 64bit 纠 1bit，检查 2bit 错误。

该功能通过寄存器 **CFG_EFC_ECC** 中 **cfg_efc_main_ecc_en** 配置打开或关闭。

在对 Flash 发起编程操作时，ECC 功能模块会自动对 64bit 数据进行 ECC 计算，生成 ECC 校验码，同时将编程数据和 ECC 校验码写入 Flash 中；

当进行读操作时，将数据和 ECC 校验码同时读回，再进行 ECC 校验。如果发生 1bit 错误，会自动将该 bit 纠正，同时产生 ECC 1bit 错误中断上报 (如果对应中断使能打开)；如果发生 2bit 错误，无法纠正数据错误，会产生 ECC 2bit 错误中断上报 (如果对应中断使能打开)。

11.4.2.7 Flash 低功耗

当软件判断 Flash 不再使用时，可配置 Flash 为 Deep Power-Down (DPD) 模式 (低功耗模式)，同时关闭 Flash 控制器的时钟。

Flash 低功耗配置的具体步骤如下：

1. 检查 **CFG_EFC_IND_STS** 寄存器 **cfg_efc_indirect_sts** 是否为 **0** (表示正在操作)，如果是，则等待上一组操作完成。
2. 判断 AXI 总线对 Flash 的读取/编程操作已完成，且后续不会再有读取/编程操作。
3. 读取寄存器 **CFG_EFC_DBG_STS**，判断是否为 0x100 (表示空闲状态)，如果不是，则等待 Flash 操作完全空闲。
4. 配置 SYSC 寄存**器 CFG_EFC_DPD**，将不使用的 Flash 对应 bit 配置为 1，让其进入低功耗模式。
5. 检查 **CFG_EFC_DEBUG** 寄存器 **cfg_efc_fctrl_pwrst_rpt[1:0]**是否为 2'b11 (表示 DPD 状态)，如果不是，则等待进入该状态。
6. 配置 CRG 寄存器 **CFG_CRG_GTEN0** 寄存器 **cfg_crg_efc_clk_req[2:0]**中不使用的 Flash 对应 bit 配置为 0，将对应 Flash 控制器的时钟关断。
7. 判断 Flash 需要重新打开。
8. 配置 CRG 寄存器 **CFG_CRG_GTEN0** 寄存器 **cfg_crg_efc_clk_req[2:0]**中待使用的 Flash 对应 bit 配置为 1，将对应 Flash 控制器的时钟打开。
9. 配置 SYSC 寄存器 **CFG_EFC_DPD**，将不使用的 Flash 对应 bit 配置为 0，让其从低功耗模式进入工作模式。
10. 检查 **CFG_EFC_DEBUG** 寄存器 **cfg_efc_fctrl_pwrst_rpt[1:0]**是否为 2'b10 (表示工作状态)，如果不是，则等待进入该状态。
11. 检查 Flash 是否成功进入工作状态。如果是，Flash 可正常工作。

11.4.2.8 Flash 中断和事件

以下是 Flash 中断和事件产生的错误标志:

- **擦除完成 (erase_done)**

  Flash Main 空间的 sector 擦除或片擦除完成时，上报中断；

- **Retry 擦除完成 (retry_done)**

  Flash Main 空间的 sector retry 擦除完成时，上报中断；

- **NVR 编程完成 (nvr_wrdone)**

  Flash NVR 空间的编程完成时，上报中断；

- **写保护错误 (wrp_err)**

  当配置接口/数据接口尝试对受保护区域 (**CFG_EFC_UNLOCK** 配置为 **0** 的 sector) 进行编程/擦除操作时，该信号置 1。当信号置 1 后，编程/擦除动作终止，不会对数据产生任何改变；

该状态通过将对应的错误指示清零信号置为 1 而清零；

该状态必须清零才能执行新的编程/擦除操作，否则会产生编程顺序错误，下一次编程/擦除操作也会被终止 (配置接口和数据接口相同)。

- **编程顺序错误 (pgs_err)**

  当编程顺序不正确时，该信号置 1。满足以下条件时，即表示编程顺序不正确：

  1. 数据总线发出了写请求，但 **cfg_efc_write_en** 并没有置 1；
  2. 写保护错误标记还未清零，又发出了新的编程/擦除操作；
  3. 不一致错误标记还未清零，又发出了新的编程/擦除操作；
  4. 配置接口 Flash NVR ECC 2bit 错误标记还未清零，又发出了新的编程/擦除操作；
  5. 数据接口 Flash Main ECC 2bit 错误标记还未清零，又发出了新的编程/擦除操作；

  该状态通过将对应的错误指示清零信号置为 1 而清零；

  该状态必须清零才能执行新的编程/擦除操作，否则会产生编程顺序错误，下一次编程/擦除操作也会被终止 (配置接口和数据接口相同)。

- **选通错误 (strb_err)**

  当数据接口连续 2 次以上向同一个地址编程同一字节时，该信号置 1。当信号置 1 后，编程动作不会终止，应用程序可忽略该错误，继续执行当前编程操作，并可以继续执行新的编程/擦除操作；

  该状态通过将对应的错误指示清零信号置为 1 而清零；

  该状态不须清零也可以执行新的编程/擦除操作；

- **不一致错误 (inc_err)**

  当配置接口上一组命令还未执行完成，配置接口再次发出新的命令，就发生不一致错误，并且新的命令不会被执行；

  该状态通过将对应的错误指示清零信号置为 1 而清零；

  对于配置接口，该状态必须清零才能执行新的编程/擦除操作，否则会产生编程顺序 错误，下一次编程/擦除操作也会被终止；

  数据接口不受该错误影响；

- **NVR ECC1bit 错误 (nvr_ecc1bit_err)**

  配置接口读取 NVR 时，发生 ECC 1bit 错误，并纠错，该信号置 1。当信号置 1 后，读取返回数据正确，应用程序可忽略该错误，继续执行当前读操作，以及下一步操作(不需要对当前地址进行 retry 处理)；

  该状态通过将 ECC 错误指示清零信号置为 1 而清零；

  该状态不须清零也可以执行新的读/编程/擦除操作；

- **NVR ECC2bit 错误 (nvr_ecc2bit_err)**

  配置接口读取 NVR 时，发生 ECC 2bit 错误，该信号置 1。当信号置 1 后，读取返回数据不正确，应用程序无法忽略该错误，需要对当前地址进行 retry 处理或其他动作；

该状态通过将 ECC 错误指示清零信号置为 1 而清零；

对于配置接口，该状态必须清零才可以执行新的读/编程/擦除操作；

数据接口不受该错误影响；

- **Main ECC 1bit 错误 (main_ecc1bit_err)**

  数据接口读取 Main 时，发生 ECC 1bit 错误，并纠错，该信号置 1。当信号置 1 后，读取返回数据正确，应用程序可忽略该错误，继续执行当前读操作，以及下一步操作 (不需要对当前操作进行 retry 处理)；

  该状态通过将 ECC 错误指示清零信号置为 1 而清零；

  该状态不须清零也可以执行新的读/编程/擦除操作；

- **Main ECC 2bit 错误 (main_ecc2bit_err)**

  数据接口读取 Main 时，发生 ECC 2bit 错误，该信号置 1。当信号置 1 后，读取返回数据不正确，应用程序无法忽略该错误，需要对当前操作进行 retry 处理或其他动作；

  发生此错误时上报中断，总线数据返回 0；

  该状态通过将 ECC 错误指示清零信号置为 1 而清零；

  对于配置接口，该状态必须清零才能执行新的擦除操作，否则会产生编程顺序 错误，下一次擦除操作也会被终止；

  对于数据接口，该状态必须清零才能执行新的编程/擦除操作，否则会产生编程顺序 错误，下一次编程/擦除操作也会被终止；在该错误未清零前，新的读操作也不再执行；

- **Boot Timeout 错误 (boot_timeout)**

  Boot 过程中，如果 Flash 初始化超时，发生 Boot Timeout 错误。

- **门控 APB 访问错误 (gating_apberr)**

  当 Flash 在低功耗状态时，APB 总线上还有数据交互未完成或来了新的总线命令；发生访问错误；

  此时上报中断，总线数据返回 0；

  该状态不须清零也可以执行新的读/编程/擦除操作；

- **门控 AXI 访问错误 (gating_axierr)**

  当 Flash 在低功耗状态时，AXI 总线上还有数据交互未完成或来了新的总线命令，总线保护模块将总线未完成的交互模拟完成，避免总线挂死；

  此时上报中断，总线数据返回 0；

  该状态不须清零也可以执行新的读/编程/擦除操作；

- **数据位宽访问错误 (dw_err)**

  当对 Flash 操作的 ECC 功能打开时，AXI 总线上出现非 64bit 操作，因为 Flash 特性无法再次计算 ECC 值并写入 Flash，则上报数据位宽访问错误。

11.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 DFLASH 章节。

12 SPI Master

12.1 简介

串行外设接口 (SPI) 可使用特定同步协议与外部器件进行半双工、全双工和单工同步串行通信。

一个 SPI 总线只支持一个 Master，但可以有一个或多个 Slave。在 SPI 应用架构中，SPI Master 作为主机，负责输出时钟和片选信号给 Slave 器件。

以 SPI Master0 IP 为例，1 主 1 从的应用架构如下图所示。

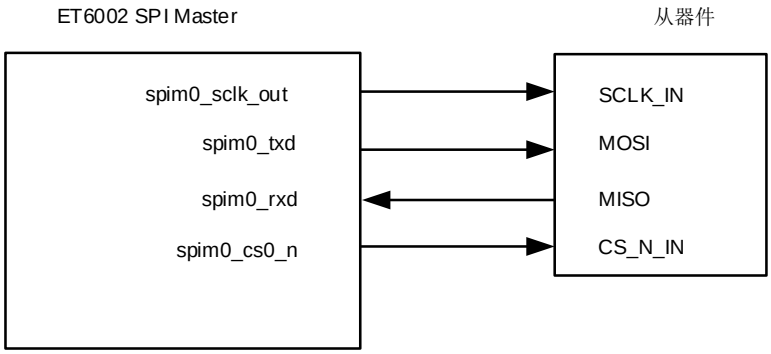


图12-1 一主一从 SPI 应用结构

以 SPI Master0 IP 为例，1 主多从的应用结构如下图所示:

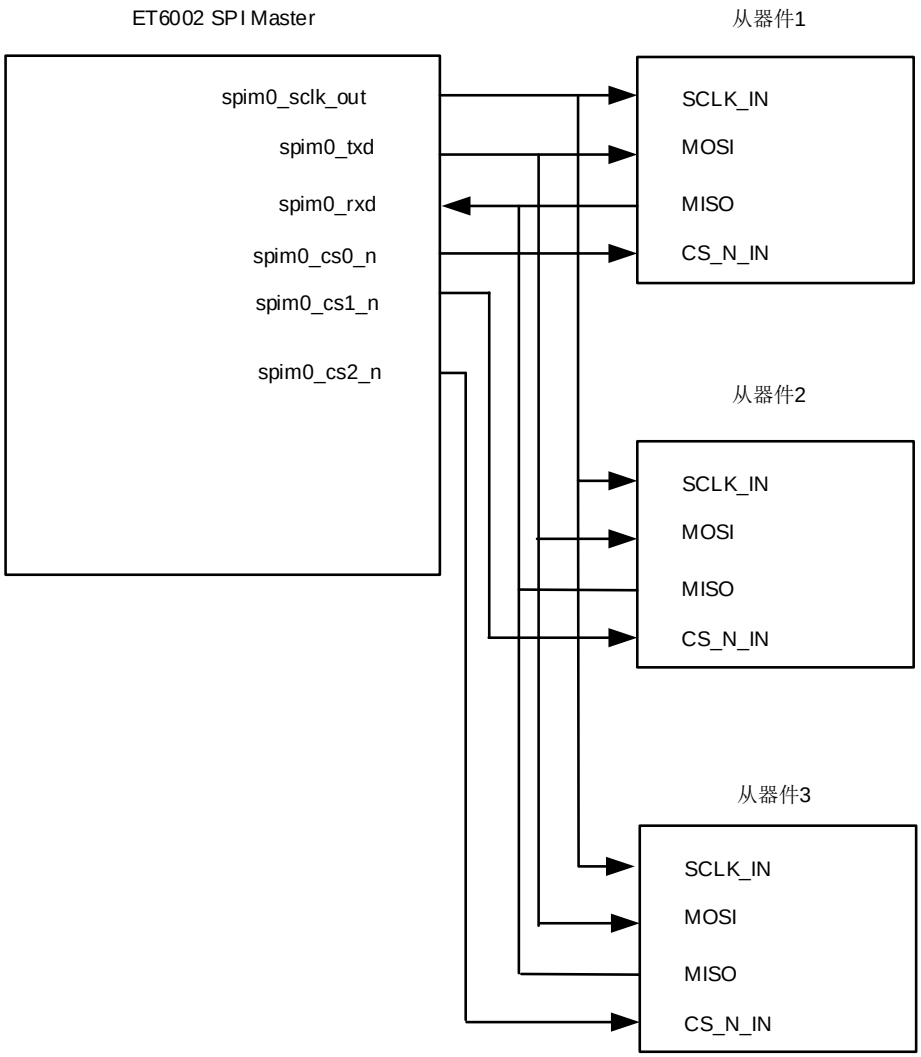


图12-2 1 主多从 SPI 应用结构

12.2 结构框图

SPI Master 框图如下：

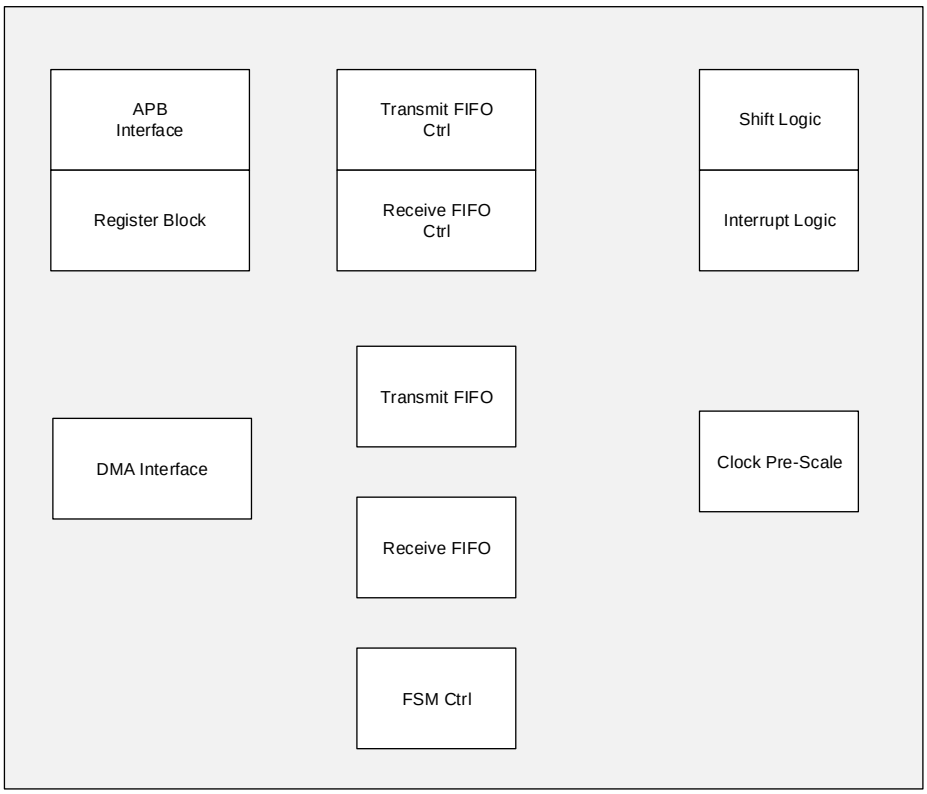


图12-3 SPI Master 框图

12.3 主要特性

ET6002 提供 3 组 1 线 SPI Master 接口，只能作为 SPI Master 使用，不能作为 SPI Slave 使用。

- 支持 16bit 内偶数整数分频系数配置，对应 SPI 输出时钟为 ssi_clk (100 MHz)/dividor；
- SPI Master 输出 SCLK 频率范围最高为 25 MHz；
- SPI Master 最高写时钟和最高读时钟均为 25MHz，SPI Master IP 核支持接收端采样延时 (RX_SAMPLE_DELAY)，Delay 颗粒度基于 SSI_CLK 时钟频率 (100 MHz)；
- SPI Master 支持的协议：
  - Motorola SPI 协议
  - TI 的 Synchronous Serial Protocol (SSP) 协议
  - National Semiconductor Microwire 协议
- 支持最大 16-bit 传输数据帧，数据帧长度可配置；
- SPI Master 支持 4 路 CS 信号输出；
- 支持硬件 DMA 握手请求；
- 支持基于内部接收和发送 FIFO 数据水线触发中断，其中 SPI Master0 接收和发送 FIFO 深度为 32，SPI Master1、2 接收和发送 FIFO 深度为 16，FIFO 宽度均为 16bit；
- 支持 TX-RX loopback 测试；
- SPI 协议模式下支持通过软件配置输出时钟的有效相位和极性；

12.4 功能描述

12.4.1 RX_SAMPLE_Delay 特性

RXD 采样延时时序图如图 12-4 所示。

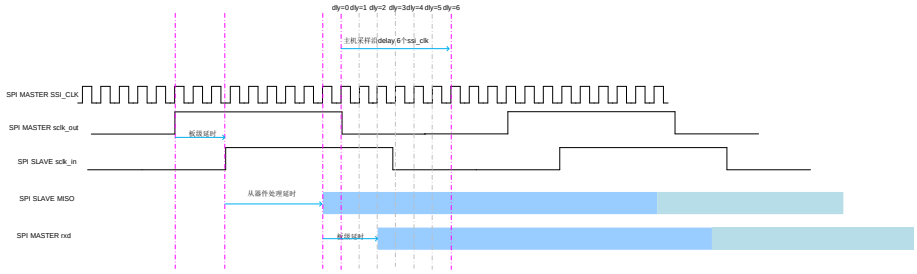


图12-4 RX Sample Delay 时序图

当主机要采样从机的 MISO 数据时，主机输出的 sclk_out 经过板级延时到达从机的 sclk_in 端口，原定于 sclk_in 上升沿输出的 MISO 数据由于从机内部逻辑处理延时又发生了 delay；

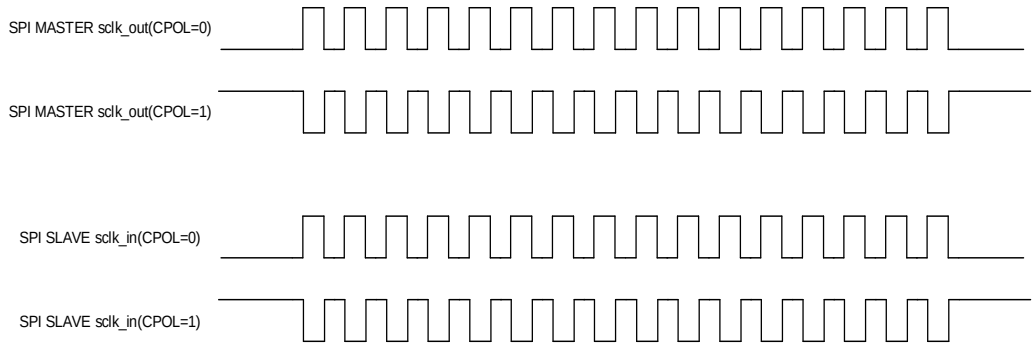
由从机输出的 MISO 经过板级延时到达主机的 RXD 输入端口；上述时序图如果主机基于默认的 sclk_out 下降沿位置采样主机的 RXD 端口数据，此时主机采样结果是错误的。**RX_SAMPLE_DELAY** 特性提供了基于主机的 **SPI Master ssi_clk** 的延时颗粒度，延迟 RXD 采样的 sclk_out 下降沿位置，上述时序图是将 **RX_SAMPLE_DELAY** 参数设置为 **6**，采样沿 delay 了 6 个 ssi_clk，此时主机就可以正确采样到从机的 MISO 数据。

12.4.2 Motorola SPI 标准特性说明

在 Motorola Serial Peripheral Interface (SPI) 标准模式下，SPI 帧格式支持 4~32bit 长度可配置。

12.4.2.1 sclk_out 时钟默认电平

当 SPI Master 没有传输 SPI 数据帧时，sclk_out 默认输出电平状态可以通过 CPOL 配置参数控制，此时要求主机和从机设备的 CPOL 设置要一致，否则会引起通信功能错误。



12.4.2.2 sclk_out 时钟默认驱动沿设置特性

sclk_out 时钟默认驱动沿可以通过 SPI Master SCPH 参数进行设置，此时从机的 SCPH 参数也需要和主机设置一致，否则会导致通信功能错误。

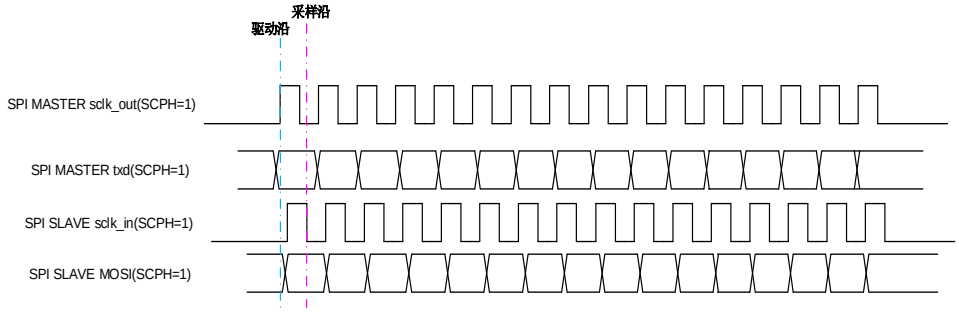


图12-5 主机在上升沿驱动 TXD，从机在下降沿采样 MOSI

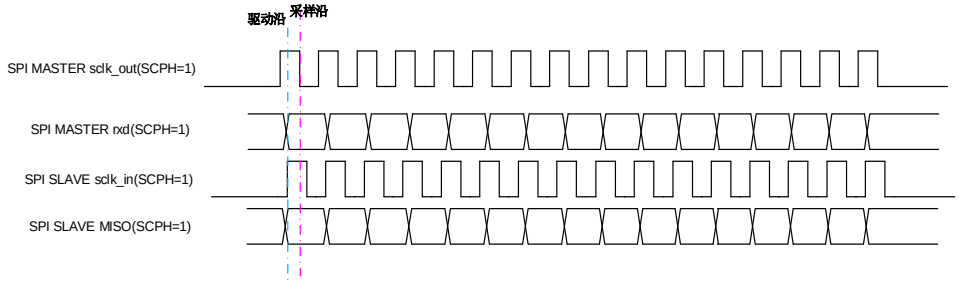


图12-6 从机在上升沿驱动 MISO，主机在下降沿采样 RXD

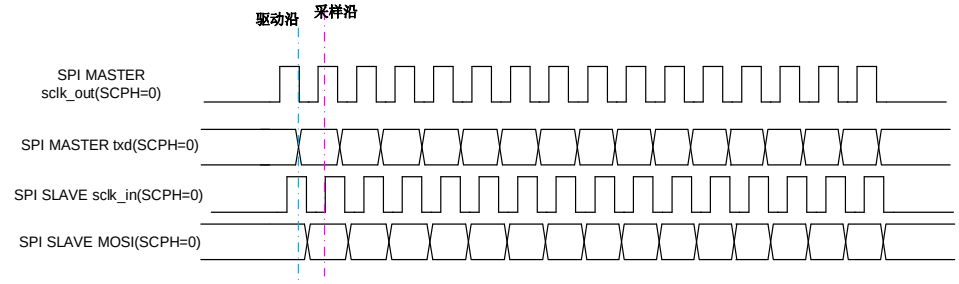


图12-7 主机在下降沿驱动 TXD，从机在上升沿采样 MOSI

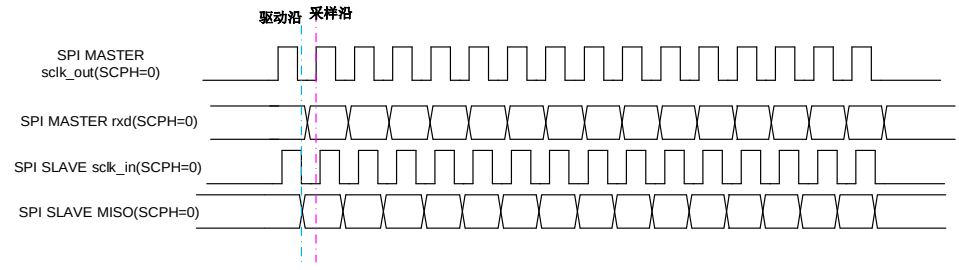


图12-8 从机在下降沿驱动 MISO，主机在上升沿采样 RXD

12.4.2.3 片选信号 Toggle 特性

该特性决定在 SPI Master 发送完一个 SPI 数据帧后，输出给从机的片选信号是否需要发生 toggle。该特性只在 SCPH 参数=0 模式下生效，可以通过 SSTE 配置参数控制。

SSTE = 0，时序如下：

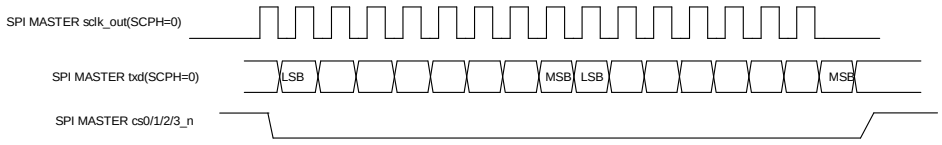


图12-9 主机输出的 SPI 帧之间，从机片选信号不发生 toggle

SSI_SCPH0_SSTOGGLE = 1，时序如下：

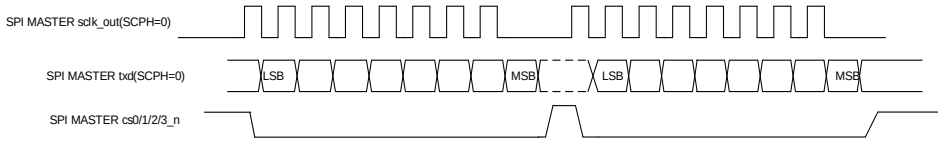
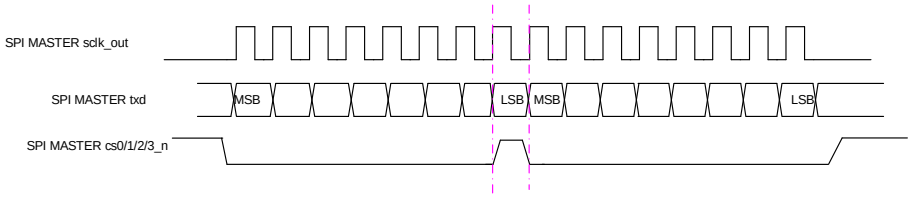


图12-10 主机输出的 SPI 帧之间，从机片选信号发生 toggle

12.4.3 Texas Instruments SSP 标准特性说明

在 TI Synchronous Serial Protocol (TI SSP) 标准模式下，SSP 帧格式支持 4~32bit 长度可配置。



在 SSP 模式下，串行数据是先发送 MSB，最后是 LSB。在 LSB 发送时，SPI Master 输出的片选信号会同步 toggle 一个 sclk_out 周期，指示是当前帧的最后一个 LSB。

要让 SPI Master IP 工作在 SSP 模式下，需要配置 **SYSC_CFG** 的 **CFG_SPI_SSP_MODE_CTRL** 寄存器。

1.252 CFG_SPI_SSP_MODE_CTRL		CFG_SPI_SSP_MODE_CTRL		Reg.	0x16C
offset		external		default	0x00000007

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

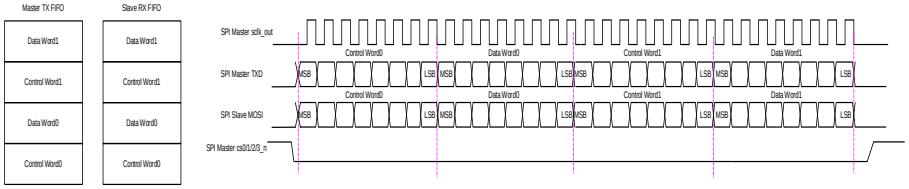
Bits	Name	S/W	H/W	Default	Description
2	sysc_spim0	rw	ro	0x1	SPIM0 SSP 模式开关

	_ssp_mode _disable				• 'b0：SSP 模式打开； • 'b1：SSP 模式关闭；
1	sysc_spim1 _ssp_mode _disable	rw	ro	0x1	SPIM1 SSP 模式开关 • 'b0：SSP 模式打开； • 'b1：SSP 模式关闭；
0	sysc_spim2 _ssp_mode _disable	rw	ro	0x1	SPIM2 SSP 模式开关 • 'b0：SSP 模式打开； • 'b1：SSP 模式关闭；

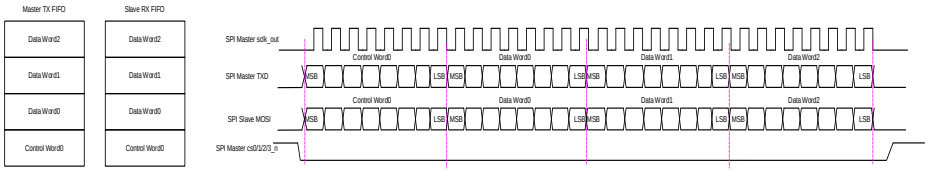
12.4.4 National Semiconductor Microwire 标准特性说明

在 NS Microwire 标准模式下，Control Word 帧支持 1~16bit 长度可配置，Data Word 帧支持 4~32bit 长度可配置。

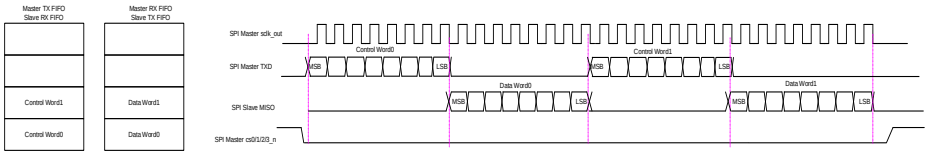
(1) SPI Master 串行发送控制帧+数据帧，SPI Slave 接收帧。



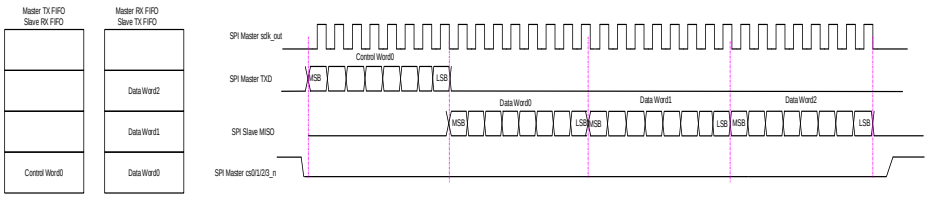
(2) SPI Master 发送 1 控制帧+多个数据帧，SPI Slave 接收帧。



(3) SPI Master 发送 1 个控制帧后，SPI Slave 发送 1 个数据帧，然后 SPI Master 接收数据帧。



(4) SPI Master 发送 1 个控制帧后，SPI Slave 发送多个数据帧，SPI Master 接收多个数据帧。



12.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 SPI Master 章节。

13 SPI Slave

13.1 简介

串行外设接口 (SPI) 可使用特定同步协议与外部器件进行半双工、全双工和单工同步串行通信。一个 SPI 总线只支持一个 Master，但可以有一个或多个 Slave。在 SPI 应用架构中，SPI Slave 作为从机，接收主机输出时钟和片选信号。

以 SPI Slave0 IP 为例，1 主 1 从的应用架构如下图所示:

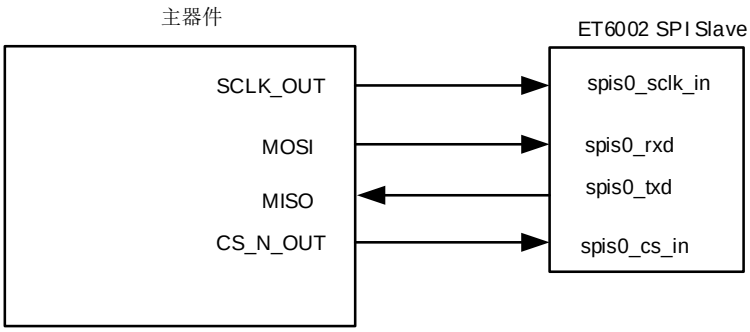


图13-1 一主一从 SPI 应用结构

以 SPI Slave0 IP 为例，1 主多从的应用结构如下图所示:

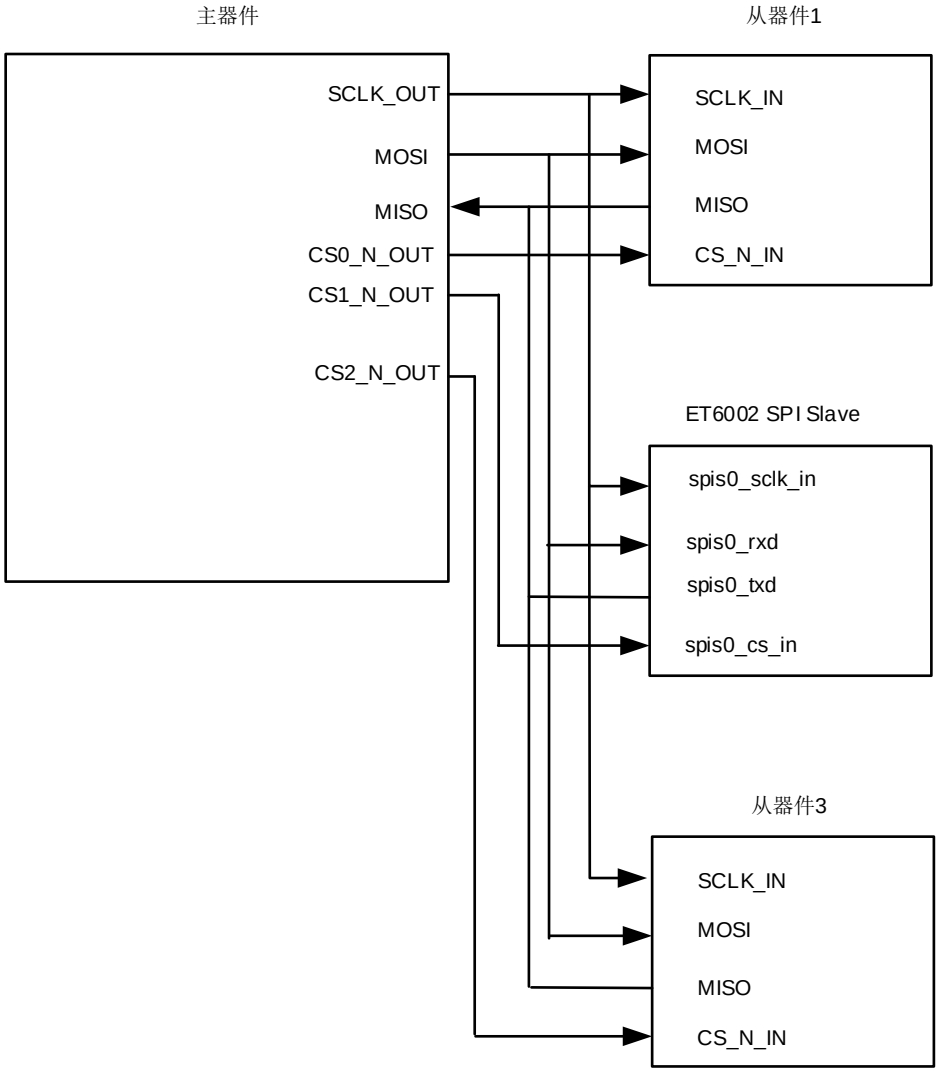


图13-2 1 主多从 SPI 应用结构

13.2 结构框图

SPI Slave 框图如下：

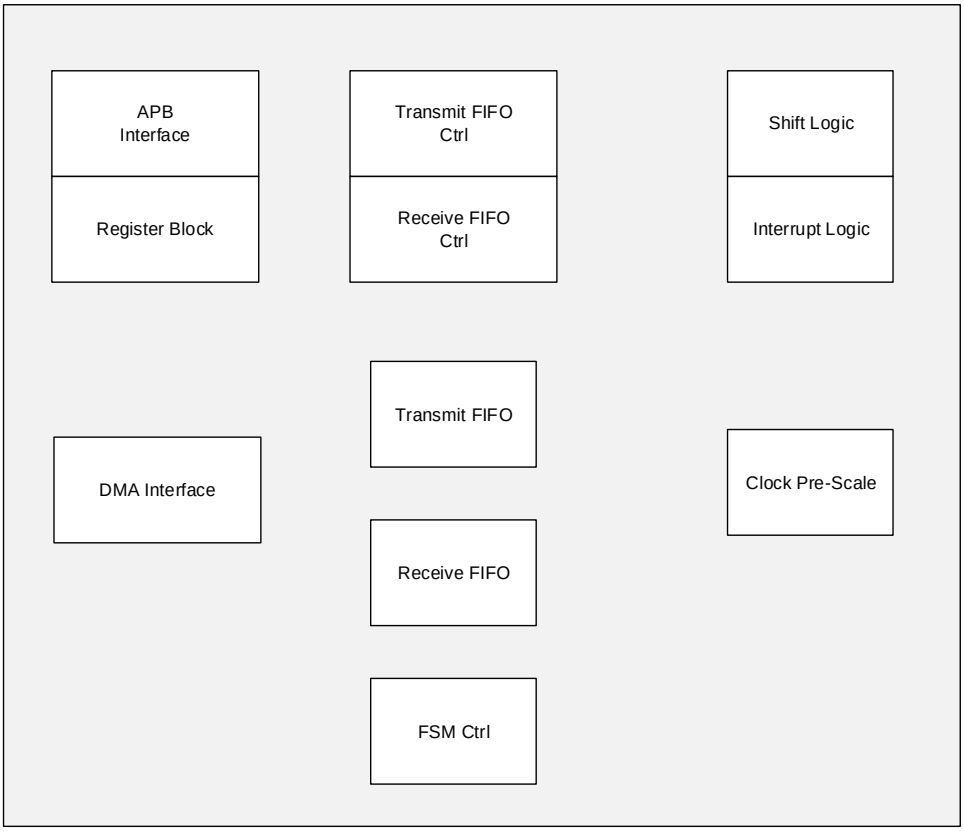


图13-3 SPI Slave 框图

13.3 主要特性

ET6002 提供 1 组 1 线 SPI Slave 接口，只能作为 SPI Slave 使用，不能作为 SPI Master 使用。

- SPI Slave 输入 SCLK 频率范围最高为 12.5 MHz；
- SPI Slave 最高写时钟为 12.5 MHz，SPI 最高读时钟为 12.5 MHz；
- SPI Slave 支持的协议：
  - Motorola SPI 协议
  - TI 的 Synchronous Serial Protocol (SSP) 协议
  - National Semiconductor Microwire 协议
- 支持最大 16-bit 传输数据帧，数据帧长度可配置；
- SPI Slave 支持 1 路 CS 信号输入；
- 支持硬件 DMA 握手请求；
- 支持基于内部接收和发送 FIFO 数据水线触发中断，接收和发送 FIFO 深度为 16，FIFO 宽度为 16bit；
- 支持 TX-RX loopback 测试；
- SPI 协议模式下支持通过软件配置输出时钟的有效相位和极性；

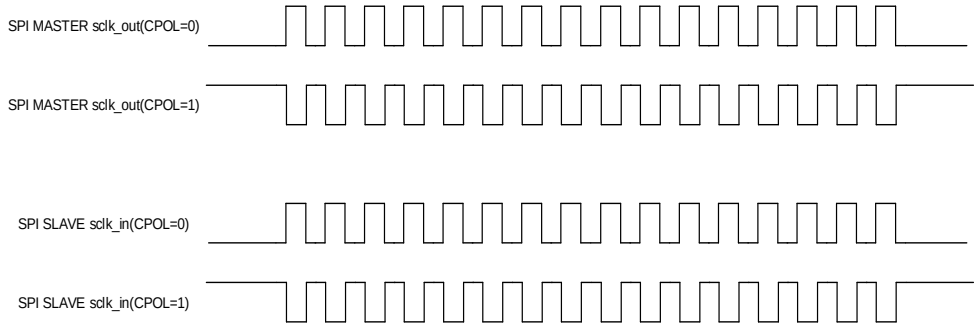
13.4 功能描述

13.4.1 Motorola SPI 标准特性说明

在 Motorola Serial Peripheral Interface (SPI) 标准模式下，SPI 帧格式支持 4~16bit 长度参数可配置。

13.4.1.1 sclk_out 时钟默认电平

当 SPI Master 没有传输 SPI 数据帧，sclk_out 默认输出电平状态可以通过 CPOL 配置参数控制，此时要求主机和从机设备的 CPOL 设置要一致，否则会引起通信功能错误。



13.4.1.2 sclk_out 时钟默认驱动沿设置特性

sclk_out 时钟默认驱动沿可以通过 SPI Master SCPH 参数进行设置，此时从机的 SCPH 参数也需要和主机设置一致，否则会导致通信功能错误。

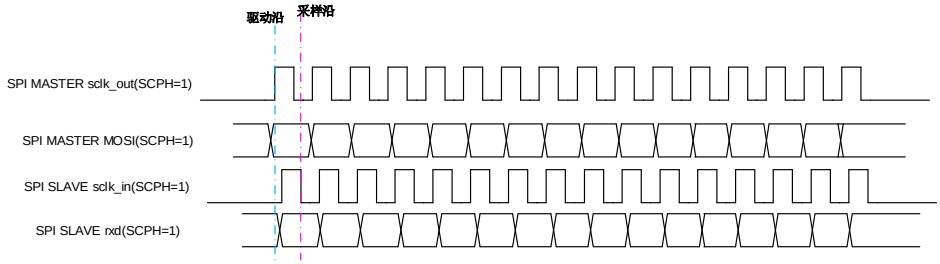


图13-4 主机在上升沿驱动 MOSI，从机在下降沿采样 RXD

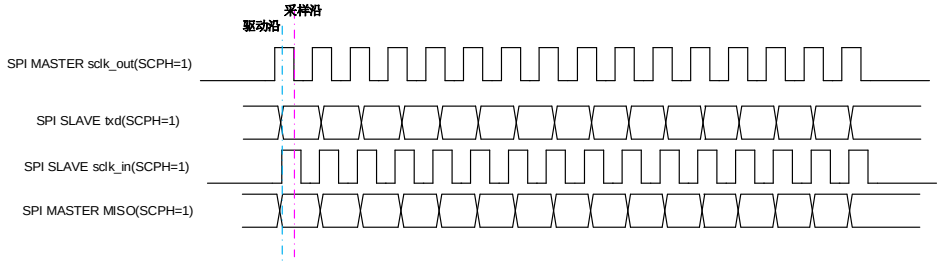


图13-5 从机在上升沿驱动 MISO，主机在下降沿采样 TXD

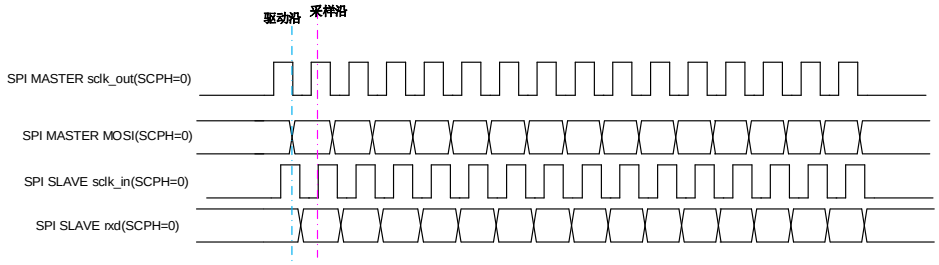


图13-6 主机在下降沿驱动 MOSI，从机在上升沿采样 RXD

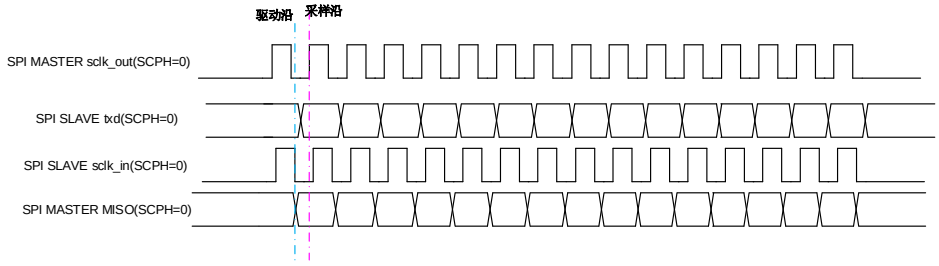


图13-7 从机在下降沿驱动 TXD，主机在上升沿采样 MISO

13.4.1.3 片选信号 Toggle 特性

该特性决定在 SPI Master 发送完一个 SPI 数据帧后,输出给从机的片选信号是否需要发生 toggle，该特性只在 SCPH 参数=0 模式下生效，可以通过 SSTE 配置参数控制。

SSTE = 0，时序如下：

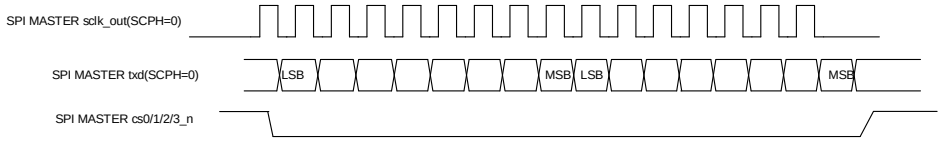


图13-8 主机输出的 SPI 帧之间，从机片选信号不发生 toggle

SSI_SCPH0_SSTOGGLE = 1，时序如下：

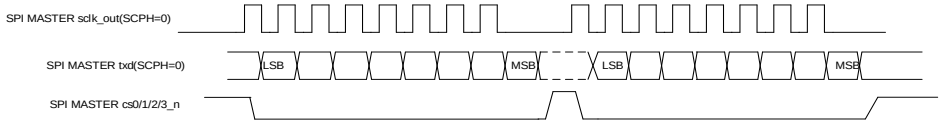
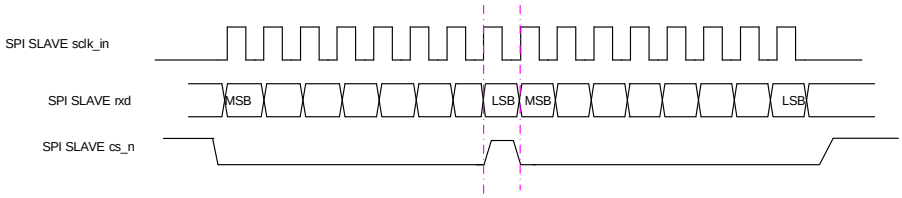


图13-9 主机输出的 SPI 帧之间，从机片选信号发生 toggle

13.4.2 Texas Instruments SSP 标准特性说明

在 TI Synchronous Serial Protocol (TI SSP) 标准模式下，SSP 帧格式支持 4~32bit 长度参数可配置。

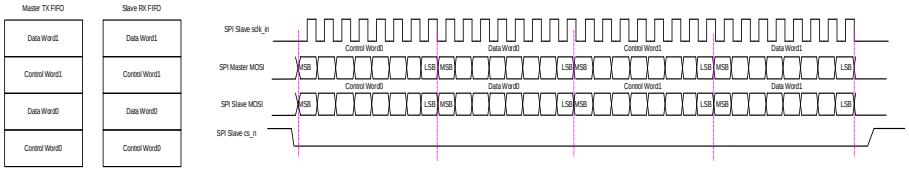


在 SSP 模式下，串行数据是先发送 MSB，最后是 LSB，在 LSB 发送时，需要 SPI Master 输出的片选信号同步 toggle 一个 sclk_out 周期，指示是当前帧的最后一个 LSB。

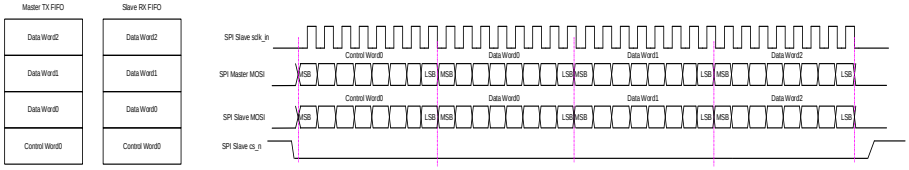
13.4.3 National Semiconductor Microwire 标准特性说明

在 NS Microwire 标准模式下，Control Word 帧长度支持 1~16bit 长度参数可配置，Data Word 帧长度支持 4~32bit 长度参数可配置。

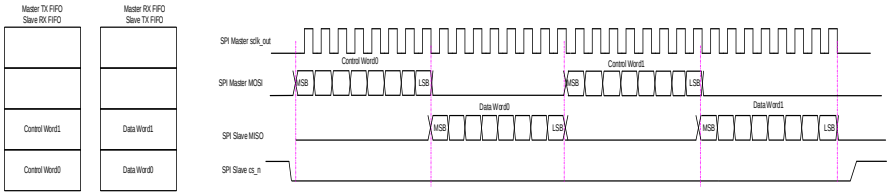
(1) SPI Master 串行发送控制帧+数据帧，SPI Slave 接收帧。



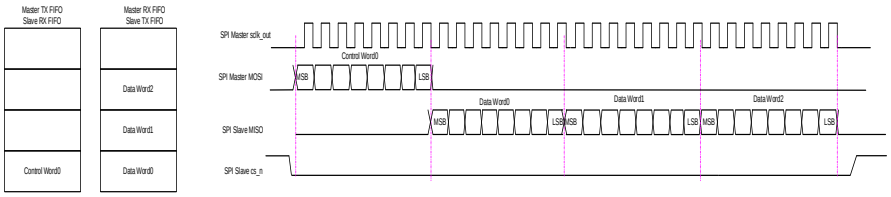
(2) SPI Master 发送 1 控制帧+多个数据帧，SPI Slave 接收帧。



(3) SPI Master 发送 1 个控制帧后，SPI Slave 发送 1 个数据帧，然后 SPI Master 接收数据帧。



(4) SPI Master 发送 1 个控制帧后，SPI Slave 发送多个数据帧，SPI Master 接收多个数据帧。



13.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 SPI Slave 章节。

14 通用 I/O (GPIO)

14.1 简介

GPIO 提供具有 16 位 I/O 接口，在 ET6002 提供了 6 组 GPIO IP，理论可同时提供 84 根 GPIO 信号输入或者输出使用。

每个 I/O 都有独立的输出值寄存器和输出方向控制 bit，通过 set 和 clr 操作进行配置，不会影响已用通道的配置。

I/O 端口作为输入时，I/O 状态变化时会产生中断，中断触发源支持电平信号或者边沿变化。

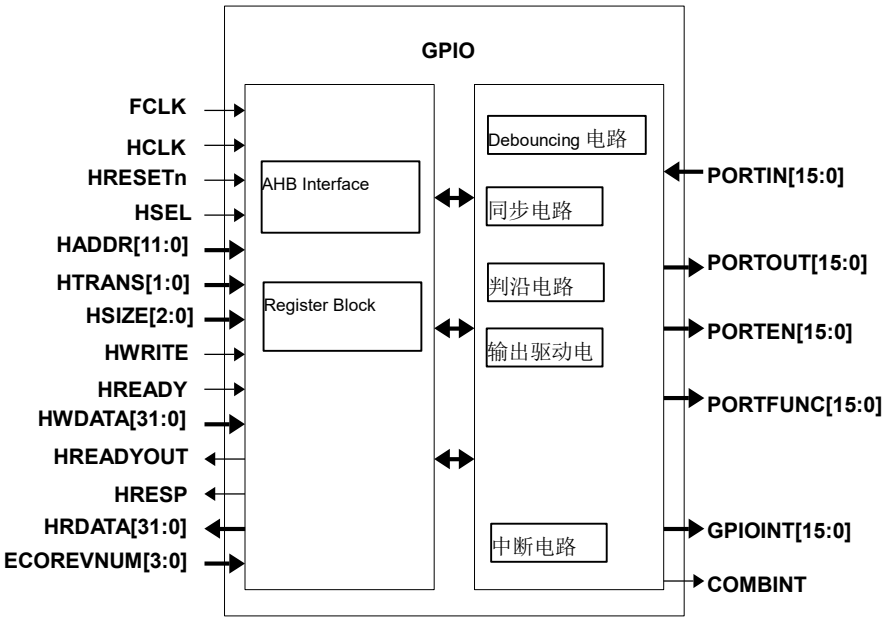
允许通过读寄存器方式来读取 I/O 端口电平的数字信号。

一个 GPIO IP 提供 16 个 I/O 通道，每路 I/O 通道单独输出 1bit 中断。一个 GPIO IP 送出共 16bit 中断，高有效，与 PCLK 同步；以及 16 个通道合并后的 1bit 中断。

GPIO IP IO 与其他 IP 输出的 I/O 信号经芯片的 IOMUX 模块复用配置后，被选输出信号再与芯片 Pin 连接。

GPIO4 直接发送 16bit 中断以及合并后的中断给 CPU 作为中断源，GPIO0~3、5 只发送合并后的中断给 CPU。

14.2 结构框图



14.3 主要特性

- 提供可编程中断生成功能。三个寄存器对此进行控制，每个寄存器都有单独的设置地址和清除地址；
- 屏蔽访问功能允许在单个传输中读取或写入单个位或多个位。这样可以避免基于软件的**读取-修改-写入**操作；
- 默认为输入模式，输出驱动电路不使能。每个 Port 可以独立设置输入输出方向；
- 使用地址值的位屏蔽支持；
- 通过为控制寄存器提供单独的设置地址和清除地址来实现线程安全操作；
- 支持外部去毛刺滤波时钟输入，每个 GPIO 管脚可以独立设置；支持高、低电平毛刺滤波，可滤除毛刺宽度受外部输入的去毛刺滤波时钟周期决定；
- 使用双触发器对经去毛刺滤波电路处理后的输入信号进行采样，以避免亚稳态问题；

14.4 功能描述

14.4.1 去毛刺滤波功能

可根据实际使用场景，判断毛刺为高电平毛刺还是低电平毛刺。如果要求双沿去毛刺，则通过配置 **DEBOUNCE_MODE** 和 **BOTHEDGE_DB_MODE** 使能相应的模式；如果需要滤除高电平毛刺，需要配置 **GPIO_IN_POLARITY** 寄存器。

DEBOUNCE_MODE[i],BOTHEDGE_DB_MODE[i]:

2'b00: GPIOx 输入无滤毛刺功能

2'b01: GPIOx 输入无滤毛刺功能

2'b10: GPIOx 输入高脉冲或者低脉冲滤毛刺功能

2'b11: GPIOx 输入双沿滤毛刺功能

14.4.2 输入信号中断产生模式

GPIO 提供可编程中断生成功能，受三个寄存器控制，每个寄存器有单独的 set 和 clear 地址。可基于这三个寄存器配置 I/O 引脚的每个 bit 位，从而生成中断。如表 14-1 所示。

表14-1 中断产生

Interrupt Enable[n]	Interrupt Polarity[n]	Interrupt Type[n]	Interrupt Feature
0	-	-	Disabled
1	0	0	低电平产生中断
1	0	1	下降沿产生中断
1	1	0	高电平产生中断
1	1	1	上升沿产生中断

触发中断后，将设置 INTSTATUS 寄存器中的相应位。这也会导致 GPIOINT[15：0]信号

的相应位被置位。因此，组合中断信号 COMBINT 也被置位。可以使用中断处理程序清除中断状态，该处理程序将 **1** 写入 **INTCLEAR** 寄存器的相应位，该地址与 **INTSTATUS** 寄存器相同。

14.4.3 屏蔽访问

屏蔽访问功能允许在单个传输中读取或写入单个位或多个位。这样可以避免基于软件的读取-修改-写入操作。通过屏蔽访问操作，16 位 I/O 分为两半，即下字节和高字节。位掩码地址空间定义为两个数组，每个数组包含 256 个字。

例如，要在单个操作中将位[1:0]设置为 1 并清除位[7:6]，可以对下字节掩码访问地址空间执行写入。所需的位掩码是 **0xC3**，可以将操作编写为 **MASKLOWBYTE[0xC3] = 0x03**。

图14-1 屏蔽访问 1

同样，要更新 GPIO 端口上部 8 位中的某些位，可以使用 MASKHIGHBYTE 数组。

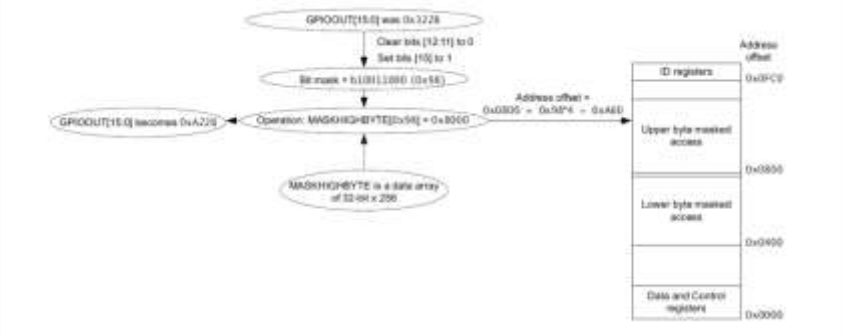


图14-2 屏蔽访问 2

14.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 GPIO 章节。

15 超级定时器 (STIMER)

15.1 简介

STIMER 超级定时器由 2 个 64 位的独立计数器组成，可单独工作。计数器的时钟可通过预分频器进行预分频。支持自动重载计数周期寄存器。自动重载寄存器可通过软件进行读写。即使在计数器运行时也可执行读写操作。

STIMER 超级定时器根据配置的 4 个比较门限，产生 ACMP0~3 共 4 个比较事件，并且还可以触发 4 个比较中断，中断类型可通过寄存器查询。

15.2 结构框图

STIMER 结构框图如下：

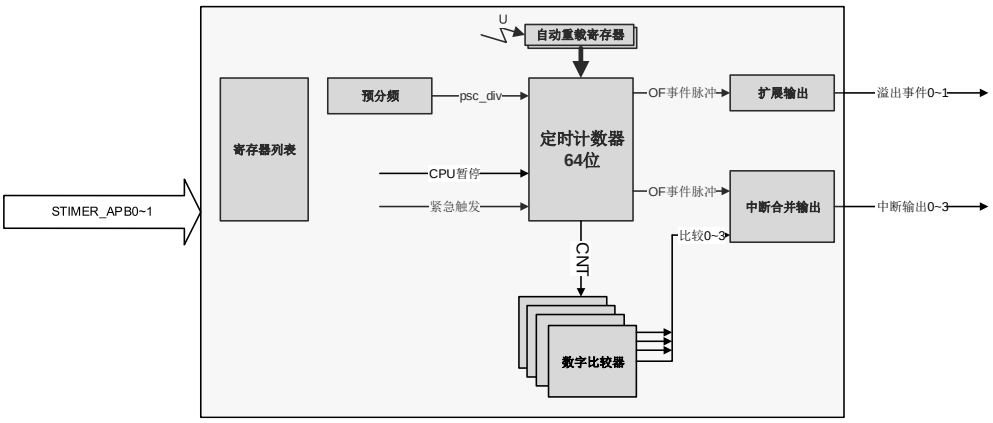


图15-1 STIMER 结构框图

15.3 主要特性

STIMER 超级定时器主要有以下特性：

- STIMER0/STIMER1 为 64 位的递增超级定时器。
- STIMER0~1 具有 16bit 的预分频器，分频系数为 1~65536，对 APB 总线时钟进行分频。
- STIMER0~1 分别支持模块使能，不使能时，不会输出中断和事件。
- STIMER0~1 分别支持自动重载寄存器，使计数最大值有预加载功能。
- STIMER0~1 分别支持定时计数器软件清 0 功能。
- 支持两种工作模式：
  - One-shot 单次计数模式：内部计数器计到最大值后返回 0。

- Free-run 循环计数模式：内部计数器计到最大值后返回 0，并重新加载计数最大值后继续计数。

- 当计数器计数到软件配置的最大值后产生溢出事件和中断。
- 支持 CPU debug 调试模式 (需要使能)，进入调试模式时，STIMER 暂停计数。
- 支持芯片紧急故障触发处理，每个 STIMER 支持 2 个触发源。
- 支持 4 个比较门限配置，当计数到配置门限值时，产生 ACMP0~3 比较事件，并可以触发中断。

15.4 功能描述

15.4.1 预分频

STIMER0~1 支持时钟预分频，预分频是一个 16bit 计数器，当这个计数器计数到寄存器配置的预分频系数 **cfg_stm_pscl_r** 时，产生一个 APB 时钟周期的分频脉冲 psc_div。

预分频计数只有在通过设置 **cfg_stm_ton_r** 使能 STIMER 模块时才能开始分频计数，并且预分频系数 **cfg_stm_pscl_r** 是静态参数，只有在模块使能关闭后才能修改。

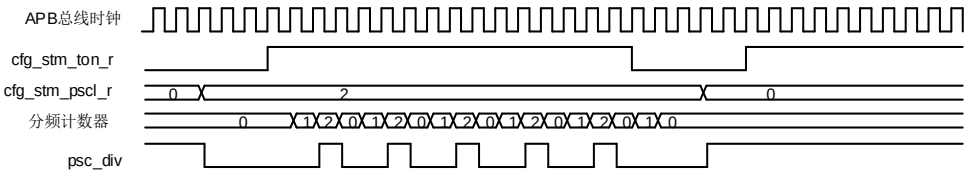


图15-2 STIMER 预分频计数

预分频系数是 1~65536，代表是 1~65536 分频，设置预分频参数 **cfg_stm_pscl_r** 时候，其取值需要在预分频系数基础上减 1。比如，如果是 1 分频 (不分频)，只需要在 **cfg_stm_ton_r** 为 **0** 时，配置 **cfg_stm_pscl_r** 为 **0**；如果是最大 65536 分频，同样配置 **cfg_stm_pscl_r** 为 65535。

预分频的公式如下：

Fdiv = Fapb/(Rpsc + 1)

- Fdiv 是信号 **psc_div** 分频频率；
- Fapb 是 APB 总线时钟频率；
- Rpsc 是寄存器 **cfg_stm_pscl_r** 配置的减 1 后的分频值；

psc_div 在不分频时为常高电平。

15.4.2 定时计数器

STIMER 超级定时器只支持递增模式计数，在定时器模块通过设置 **cfg_stm_ton_r** 开启

后，定时器内部计数器在分频使能 **psc_div** 为高电平时，累加计数。

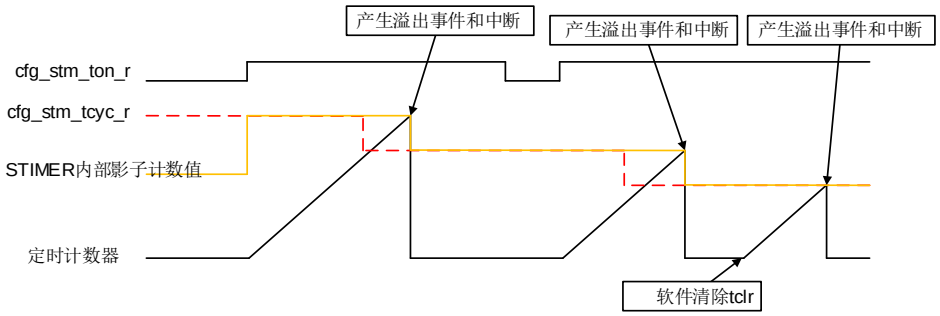
自动重载寄存器是定时计数器模块的计数最大值预加载寄存器，防止定时器在计数过程中，由于修改计数最大值寄存器 **cfg_stm_tcyc_r** 后，导致计数发生错误。只有在计数器返回 0 或者软件配置的周期计数值加载后 (需要打开影子加载模式)，计数最大值寄存器 **cfg_stm_tcyc_r** 才能写入计数器内部中。

定时计数器软件清 0 寄存器 **cfg_stm_tclr_r**，用于置位内部计数器为 0，写自清 0 寄存器 **cfg_stm_tclr_r**，会产生一个时钟周期脉冲，只有在 **pre_div** 和 **cfg_stm_tclr_r** 同时有效时，才能清除计数器；如果 **pre_div** 为低但是 **cfg_stm_tclr_r** 为高时，底层逻辑会保持 cfg_stm_tclr_r 信号为高电平，直到 **pre_div** 转为高电平。

软件配置暂停使能信号寄存器 **cfg_stm_thlt_r**，为高电平时会立即暂停计数器计数。当软件计数暂停和周期计数 Overflow 事件同时发生时，Overflow 事件优先级高，此时 STIMER 会翻转计数到 0 后，再执行计数暂停操作。

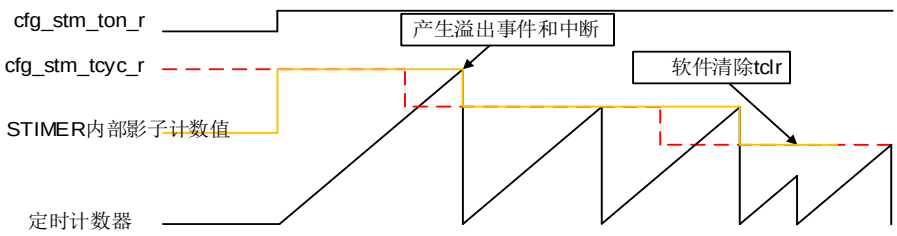
支持两种工作模式：

- One-shot 单次计数模式：内部计数器计到最大值后返回 0，产生溢出事件和中断，并且停止计数。如果需要重新计数，则需要先关闭 STIMER 模块开关后重新打开，或者在 STIMER 模块使能开启的状态下配置计数软件清 0，此时会重新开启一次计数。
- Free-run 循环计数模式：内部计数器计到最大值后返回 0，并重新加载计数最大值后继续计数，每次计数器到最大值后产生溢出事件和中断。



1. 为了图示方便，预分频系数为1，不分频。
2. 模块使能从0变为1时，开始单次计数，计数到配置的最大值后产生溢出事件和中断，返回0后停止计数。
3. 如果需要重新开始新的一次计数，则需要把模块使能关闭后重新开启，或者配置软件清除寄存器cfg_stm_tclr_r。

图15-3 单次计数示意图



1. 为了图示方便，预分频系数为1，不分频。
2. 模块使能从0变为1时，开始循环计数，计数到配置的最大值后产生溢出事件和中断，返回0后把计数器最大值cfg_stm_tcyc_r配置到内部计数器中，重新开始计数。

图15-4 循环计数示意图

15.4.3 调试模式

在进入 CPU debug 调试模式时，需要配置 **cfg_stm_cpuhalt_mode_r** 寄存器。

- 为 0 时：自动配置 Timer 为暂停计数；
- 为 1 时：CPU 不进入调试模式。

CPU debug 调试模式可软件配置是否开启，上电默认为开启 Timer Halt 暂停功能。当计数暂停和周期计数 Overflow 事件同时来临时，Overflow 事件优先级高，此时 STIMER 会翻转计数到 0 后，再执行暂停操作。

该操作可避免在 Overflow 时刻使能 timer 暂停功能时，由于 timer 被冻结而不停地产生中断和触发事件。

15.4.4 事件与中断

STIMER 超级定时器的计数器在计数溢出时会产生溢出事件和溢出中断，ACMP0~3 门限比较事件和中断。

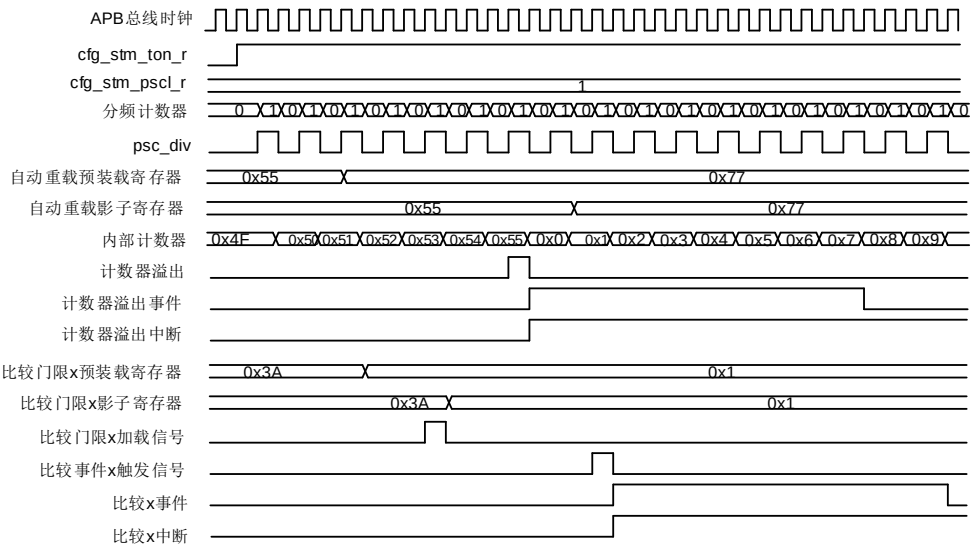


图15-5 STIMER 事件和中断

- 定时器溢出事件：在内部计数器计数到最大值后，会产生 16 个 APB 时钟周期的脉冲信号到外部模块，作为触发事件包括但不限于：XBAR、SAR ADC、BDAC、ACMP、GPIO 等。
- 定时器比较事件：当定时器内部计数器计数到配置的 4 个比较门限值时，会产生 4 个比较事件，事件为 16 个 APB 时钟周期的脉冲信号，输出到外部 SARC 模块，作

为 ADC 的触发源。

- 定时器溢出中断：在内部计数器计数到最大值后，会产生电平中断并保持，直到中断被软件清除。
- 定时器比较中断：当定时器内部计数器计数到配置的 4 个比较门限值时，会产生 4 个比较中断，该中断为电平中断。

STIMER 的溢出中断和 4 个比较中断合并成一个中断送给 SoC，如果需要知道中断的类型，可以通过中断寄存器查询。

15.4.5 紧急故障触发

当芯片发生紧急故障时，锁定当前 STIMER 值，并送往系统控制器 (SYSC) 的非易失寄存器 (仅可以通过上电复位进行复位)。

芯片紧急故障触发源 0 和芯片紧急故障触发源 1 为 XBAR 模块输出，在 STIMER 中取上升沿，把计数器模块的计数值锁存到非易失寄存器。

15.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 STIMER 章节。

16 增强型定时器 (ETIMER)

16.1 简介

增强型定时器 (ETIMER) 包含 14 个独立的通道 (ETIMER_Core)，每个通道包含一个 32 位自动重载递增时基计数器 (内部 PWM 和 CAP 共用该计数器)，该计数器由可编程的 6 位预分频器驱动。同时支持 14 路通用 PWM 功能和 14 路通用 CAP 功能，且均可以独立配置。

所有通道之间彼此独立，不共享任何资源，并且所有通道具有相同的功能 (一路 PWM 功能和一路 CAP 功能)。

该定时器可用于多种用途，包括最基本的定时功能 (基础计时)，测量输入信号 (输入捕获)，生成输出 PWM 波形 (输出比较)。

16.2 结构框图

增强型定时器 (ETIMER) 主要包含如下处理单元：

- 14 个独立的通道 (ETIMER_Core)
- 32 位递增自动重载时基计数器 (ETIMER_CNT)
- PWM 发波控制 (PWM)
- 输入捕获控制 (CAP)
- 配置寄存器、DMA 和中断处理 (ETIMER_CMN)
- PWM 发波互补模式输出处理 (ETIMER_PWM_OUT)

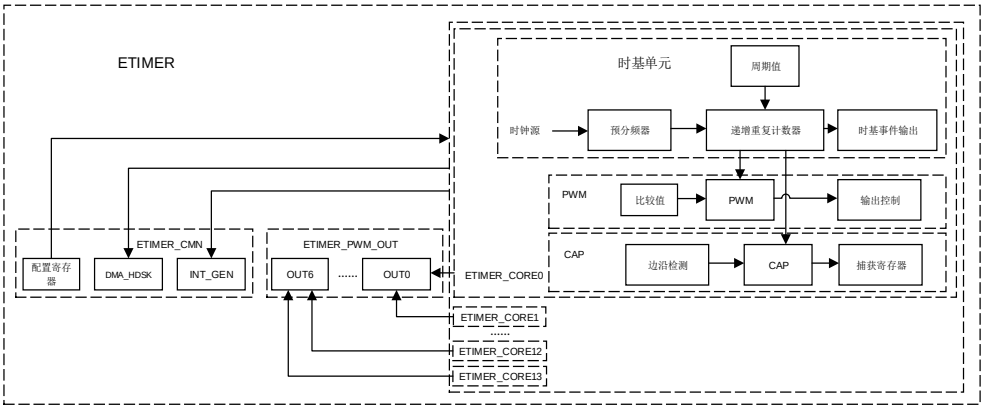


图16-1 ETIMER 结构框图

16.3 主要特性

增强型定时器 (ETIMER) 的主要特性如下：

- 32 位递增自动重载时基计数器，在 0 到周期值之间递增循环计数，最小分辨率 5 ns；
- 6 位可编程预分频器，用于对时基计数器的时钟进行分频，分频系数介于 1 到 64 之间；
- 所有时间戳寄存器均支持立即和影子两种加载模式；
- 多达 14 个独立通道 (ETIMER_Core)，每个通道可同时支持 1 路 PWM 功能和 1 路 CAP 功能；
- PWM 支持占空比和周期可调，支持死区配置；
- 14 路通用 PWM 可两两组合为互补模式发波：0 和 1、2 和 3、4 和 5、6 和 7、8 和 9、10 和 11、12 和 13 通道配对互补输出，支持 SWAP 和极性可配置；
- PWM 发波支持 COMPA 和 COMPA+COMPB 两种模式：
  - COMPA 模式，如果“Timer 值＜COMPA”，则 PWM 输出为 1，否则为 0；
  - COMPA+COMPB 模式，如果“COMPB＞Timer 值≥COMPA”，则 PWM 输出为 1，否则为 0；

说明

COMPA 和 **COMPB** 代表预设的比较值。

- PWM 支持 Fault 模式封波，支持 CBC 和 One-shot 两种封波类型，且 Fault 模式下输出极性可配置；
- 通用 CAP 功能包括：
  - 上升沿捕获
  - 下降沿捕获
  - 正脉冲宽度捕获
  - 负脉冲宽度捕获
  - 周期捕获
- CAP 捕获触发支持 One-shot 和 Continuous 两种模式；
- 支持对 CAP 输入信号进行滤波处理；
- 支持 CAP 捕获数据通过中断/DMA 进行传输请求；
- 支持四个时间戳事件输出：
  - 计数到 0 时间戳事件
  - 计数到周期值时间戳事件
  - 计数到 COMPA 时间戳事件
  - 计数到 EVT 时间戳事件
- 支持 CAP 捕获完成事件输出；
- 支持 14 个 ETIMER 通道独立产生中断请求，中断源如下：
  - PWM 在 One-shot 模式下，Fault 模式有效
  - PWM 在 CBC 模式下，Fault 模式有效
  - Fault 模式未退出，再次收到有效 Fault 源信号

  - COMPA 和 COMPB 配置错误，COMPA≥COMPB
  - 计数到 COMPA 溢出事件
  - 计数到 COMPB 溢出事件
  - 计数到配置周期事件
  - 计数器最大值溢出事件
  - 捕获 CEVT1、CEVT2、CEVT3、CEVT4 事件
- 支持 COMPA≥COMPB 配置错误告警；
- 支持四个独立的 DMA 通道，CAP 捕获完成标志触发 DMA 请求；
- 支持 ETIMER 模块各通道相应功能可独立开启或者关闭：
  - ETIMER 模块使能 (CFG_ETIM_CMN0.cfg_etim_en)
  - 计数器使能 (CFG_ETIM_CMN0.cfg_etim_timer_en)
  - PWM 使能 (CFG_ETIM_CMN1.cfg_etim_pwm_en)
  - CAP 使能 (CFG_ETIM_CMN1.cfg_etim_cap_en)
  - Filter 使能 (CFG_ETIM_CMN2.cfg_etim_cap_filter_en)
  - 事件输出使能 (CFG_ETIM_CMN3.cfg_etim_event_en)

16.4 功能描述

增强型定时器主要包含 14 个通道处理单元、PWM 输出处理单元、事件产生单元、中断处理单元和 DMA 处理单元。其中每个通道处理单元包含一个计数器单元、一个 PWM 发波处理单元和一个 CAP 捕获处理单元。每个通道中，PWM 和 CAP 处理单元共用一个计数器单元。14 路 PWM 可两两组合 (固定组合) 为互补模式发波，支持 SWAP 和极性可配置。下图主要描述了 ETIMER 模块单通道内部组成。

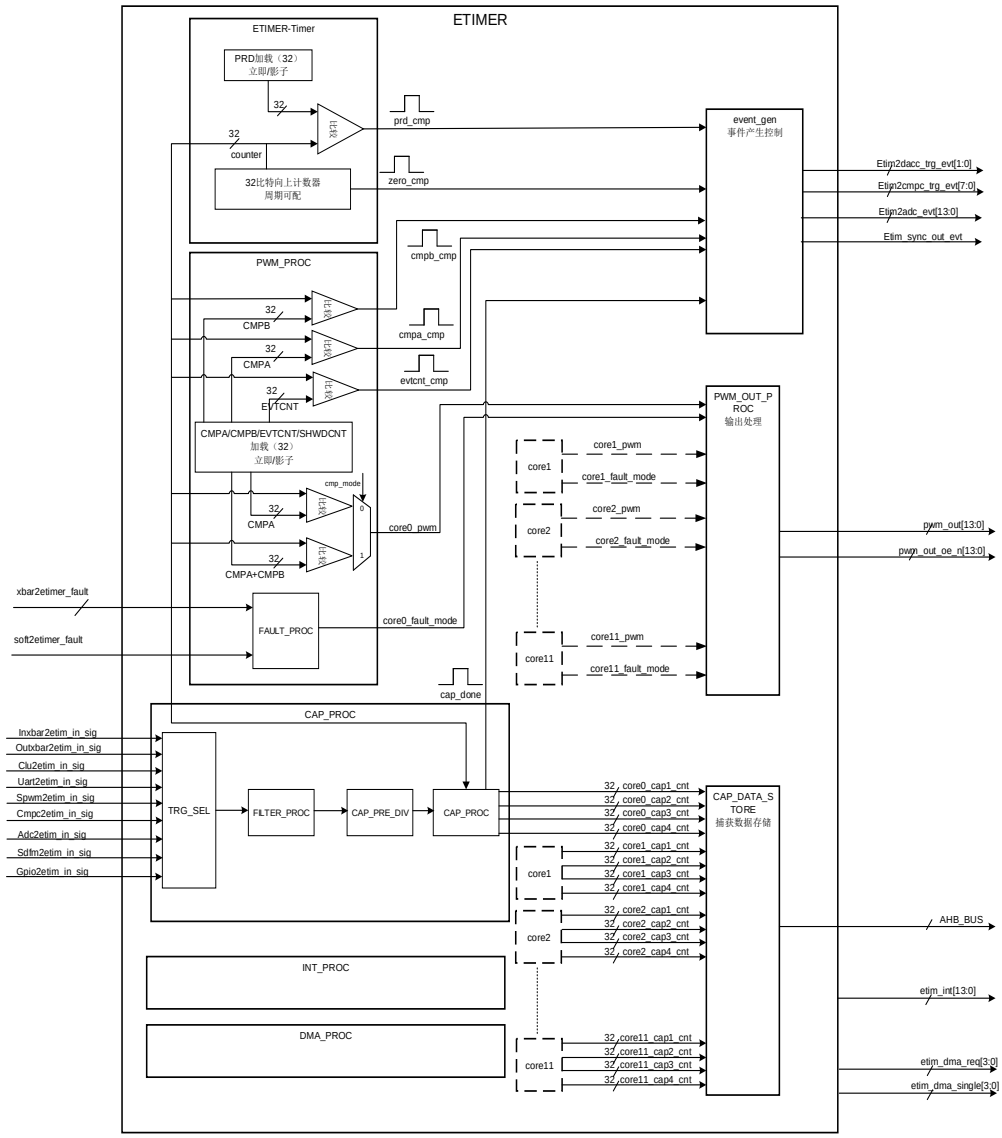


图16-2 ETIMER 实现原理图

16.4.1 时基单元

增强型定时器的每个通道包含一个独立的 32 位时基计数器及其相关的自动重载寄存器，计数器采用递增方式计数。计数器的时钟可通过预分频器进行分频。

自动重载计数器、周期值 (计数最大值) 寄存器、预分频寄存器可通过软件进行读写，即使在计数器运行中也可以执行读写操作。

时基单元包含：

- 自动重载计数器 (ETIMER_CNT)
- 周期值寄存器 (ETIMER_TCYC)
- 预分频寄存器 (ETIMER_DIV_FAC)

其中周期值寄存器 (ETIMER_TCYC) 是预装载的。预装载寄存器的内容既可以直接传送到影子寄存器立即生效 (立即加载)，也可以在每次发生更新事件时更新到影子寄存器 (影

子加载)，这取决于更新模式控制寄存器配置 (立即加载 **PWM_CTRL.tcld** 和影子加载 PWM_CTRL.shdw_ug)。

周期值寄存器影子加载时,事件更新为计数器计数到 0 的时刻。

周期值寄存器立即加载时，在影子寄存器更新时刻：

- 如果当前计数值小于新加载的周期值，则更新周期值后计数器继续计数；
- 如果当前计数值大于新加载的周期值，则更新周期值后计数器从 0 开始重新计数；

计数器由输入时钟源经过预分频器分频后提供的时钟驱动，且仅当控制寄存器 (**CFG_ETIM_CMN0**) 中 ETIMER 模块使能 (**cfg_etim_en**) 和计数器使能 (**cfg_etim_timer_en**) 同时置 1 时，才会启动计数器计数。

16.4.1.1 预分频器说明

预分频器可对计数器时钟频率进行分频，分频系数介于 1 到 64 之间。通过配置预分频系数寄存器 (**ETIMER_ DIV_FAC**) 设定预分频的分频系数。计数器计数频率公式如下：

$$f_{CLK\_CNT}=\frac{f_{CLK\_SRC}}{DIV\_FAC+1}$$

该寄存器具有预加载缓存功能，因此可以对预分频器实现实时修改。预分频器只支持立即加载，即软件配置成功后，新的预分频系数立即生效。

下面图示说明预分频比实时变化时计数器的行为：

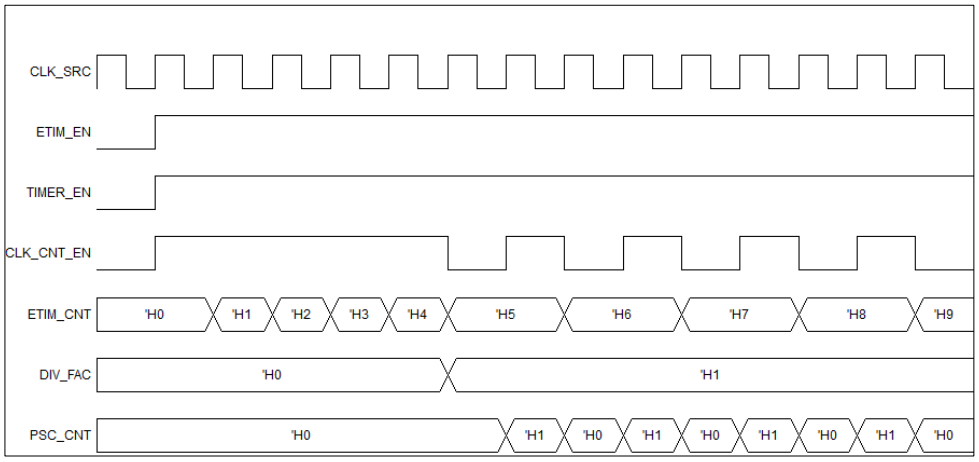


图16-3 实时修改预分频系数从 0 变为 1 的时序图

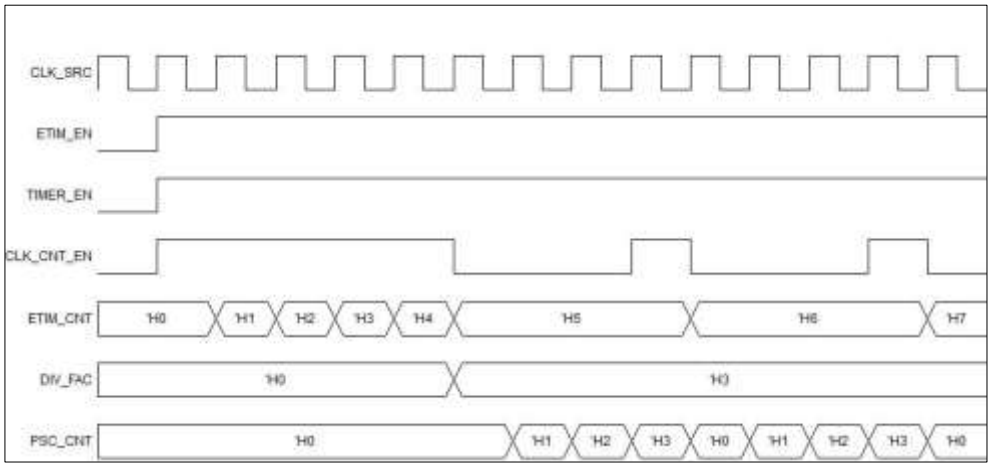


图16-4 实时修改预分频系数从 0 变为 3 的时序图

16.4.2 计数器模式

增强型定时器仅支持递增计数模式，且仅支持连续计数模式。当 ETIMER 模块使能 (**cfg_etim_en**) 和计数器使能 (**cfg_etim_timer_en**) 同时置 1 时，计数器从 0 开始递增计数到周期值。一个计数周期完成后，自动从 0 开始重新计数。

下面将通过图示举例说明当周期值配置为 **0x48** 时，不同时钟频率下计数器的计数行为。

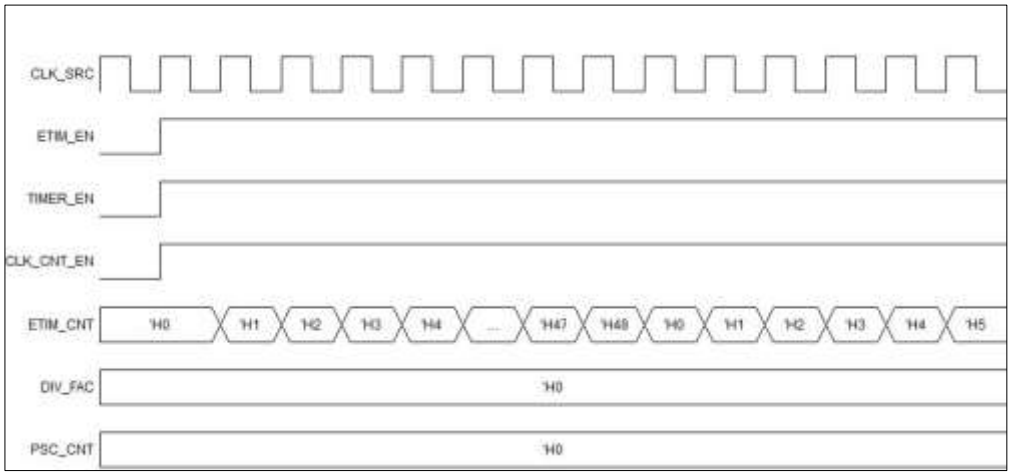


图16-5 计数器时序图，1 分频

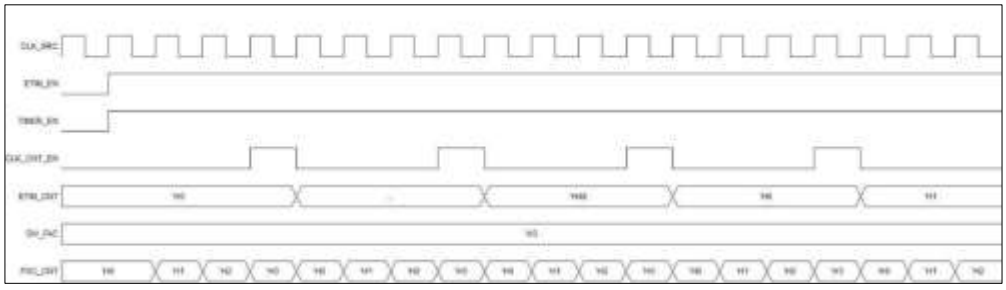


图16-6 计数器时序图，4 分频

16.4.3 时钟选择

增强型定时器 (ETIMER) 的计数器时钟源来自系统 AHB_BUS1 总线时钟，通过预分频寄存器 (**ETIMER_DIV_FAC**) 的分频产生计数使能。

16.4.4 捕获/比较通道

增强型定时器 (ETIMER) 包含 14 个独立通道，每个通道包含一计数器、一个输出比较和一个输入捕获。

输出比较是利用计数器计数，并与预先配置的比较值 (COMPA 和 COMPB) 进行比较，根据计数值与比较值的大小关系，输出对应的比较电平。输出比较支持 Fault 管理。输出比较的电平和 Fault 处理标志进入 PWM_OUT_PROC 模块，进行互补、交换和极性控制后，最终输出 14 路 PWM。

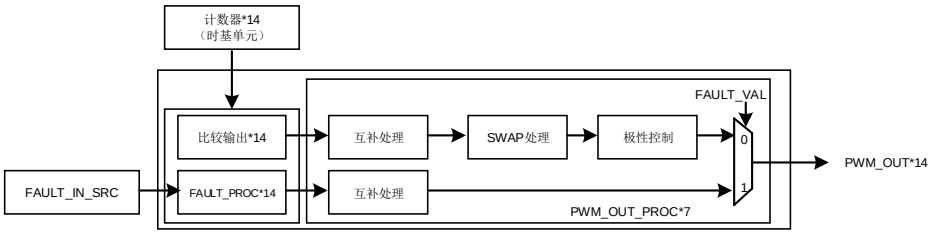


图16-7 输出比较示意图

输入捕获，首先对输入信号和输入触发源进行选择处理，然后对输入信号进行滤波处理，然后通过捕获类型来生成检测边沿。在触发信号到来时，利用计数器对检测边沿进行捕获，获取对应的捕获值。

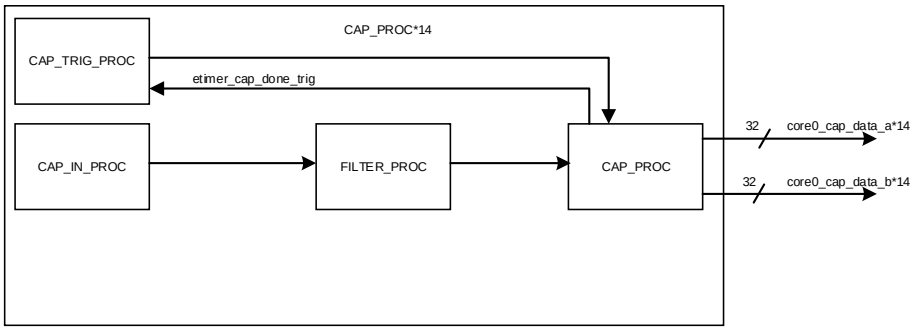


图16-8 输入捕获示意图

在比较模式下，预装载寄存器的内容将复制到影子寄存器中，然后将影子寄存器的内容与计数器进行比较。

在捕获模式下，捕获实际发生在影子寄存器中，然后将影子寄存器的内容复制到预装载寄存器中上报。

16.4.5 输出比较

当 ETIMER 模块使能(**CFG_ETIM_CMN0.cfg_etim_en**)和 PWM 使能(**CFG_ETIM_CMN1.cfg_etim_pwm_en**)都配置为 1 时，启动输出比较功能。

16.4.5.1 比较输出

比较输出需要 3 个预装载寄存器：COMPA、COMPB 和 SHWD，3 个寄存器都支持立即加载和影子加载。

COMPA 和 COMPB 的影子加载事件产生时刻为计数器计数值与 SHWD 寄存器配置值相等时刻；SHWD 的影子加载事件产生时刻为计数器计数到 0 的时刻。

比较输出支持 COMPA 和 COMPA+COMPB 两种模式：

- COMPA 模式，如果“Timer 值＜COMPA”，则 PWM 输出为 1，否则为 0；
- COMPA+COMPB 模式，如果“COMPB＞Timer 值≥COMPA”，则 PWM 输出为 1，否则为 0；

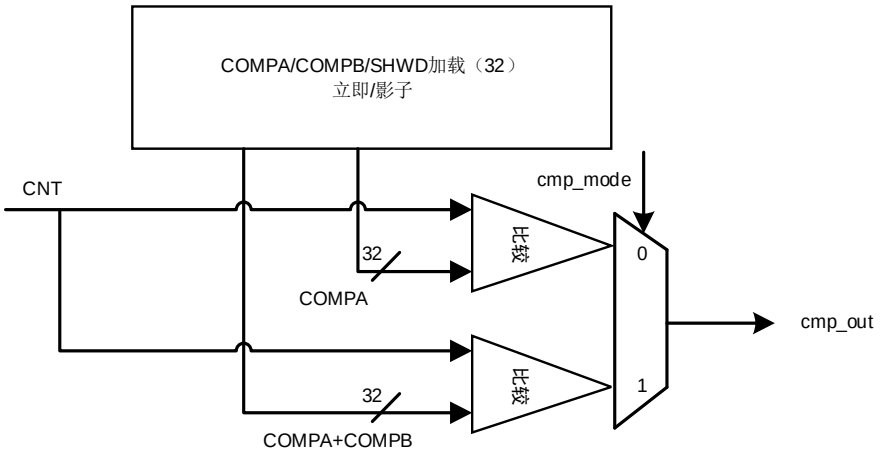


图16-9 单个通道比较输出示意

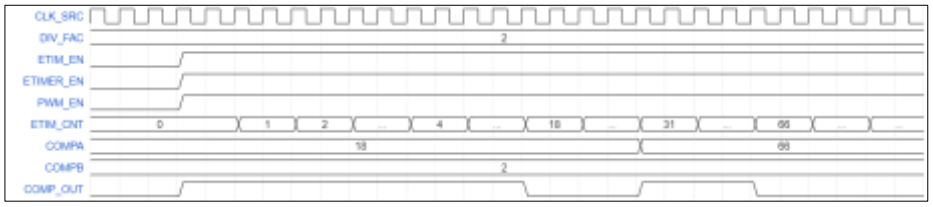


图16-10 COMPA 模式时序示意

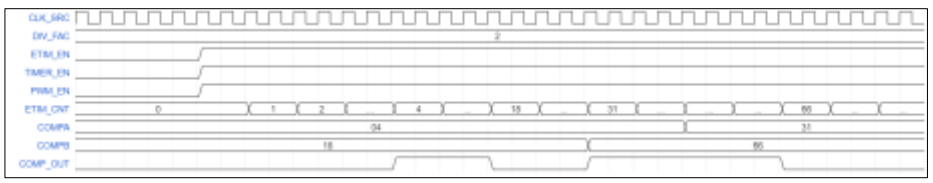


图16-11 COMPA+COMPB 模式时序示意图

16.4.5.2 Fault 处理

输出比较支持 Fault 模式封波，支持 CBC 和 One-shot 两种封波类型，且 Fault 模式下输出极性可配置。Fault 模式有效时，输出 Fault 模式输出极性配置值，而不再输出比较结果。

CBC 模式下，当计数到周期值时，如果 Fault 事件消失，则退出 Fault 模式，否则 Fault 模式进入下一个计数周期。根据 Fault 模式有效的高电平产生对应的告警和中断。

One-shot 模式下，当计数到周期值时，如果 Fault 事件消失，且软件在该周期内配置 FAULT 清除指示，则退出 Fault 模式，否则 Fault 模式将进入下一个计数周期。进入下一个计数周期后，仍需要在该周期内软件配置 Fault 清除指示后，在计数到周期值时才退出 Fault 模式，以此类推。

CBC 模式和 One-shot 模式下，都是由 Fault 模式有效的高电平产生对应的告警和中断。

Fault 模式保持有效期间，再次输入 Fault 事件，则产生 Fault 模式溢出中断。

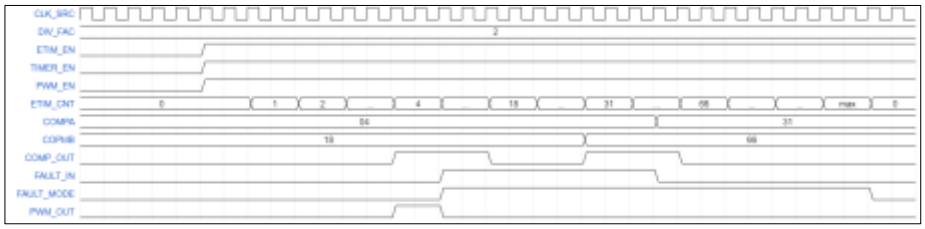


图16-12 Fault 模式 (CBC) 下输出 0

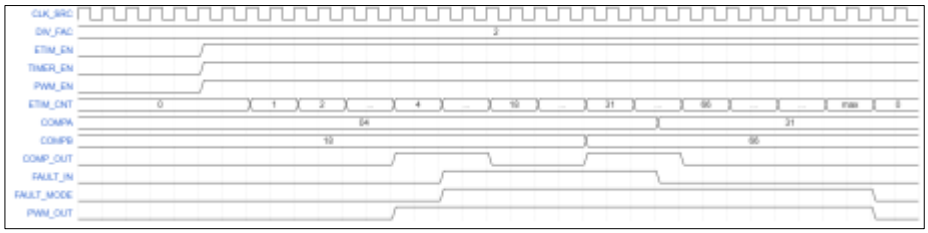


图16-13 Fault 模式 (CBC) 下输出 1

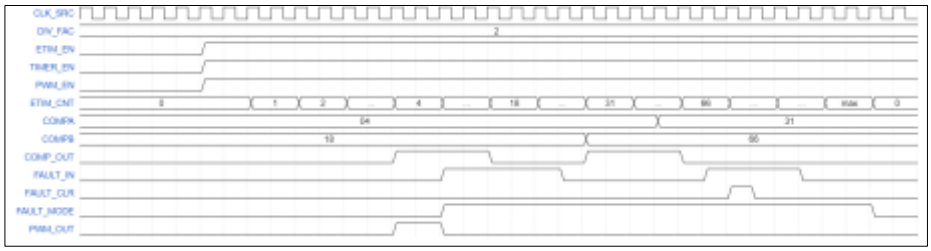


图16-14 Fault 模式 (One-shot) 下输出 0

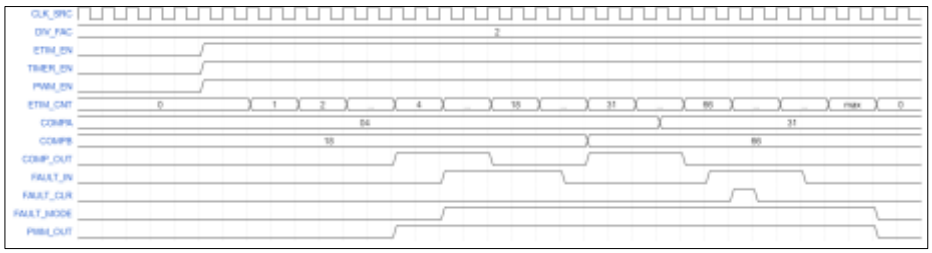


图16-15 Fault 模式 (One-shot) 下输出 1

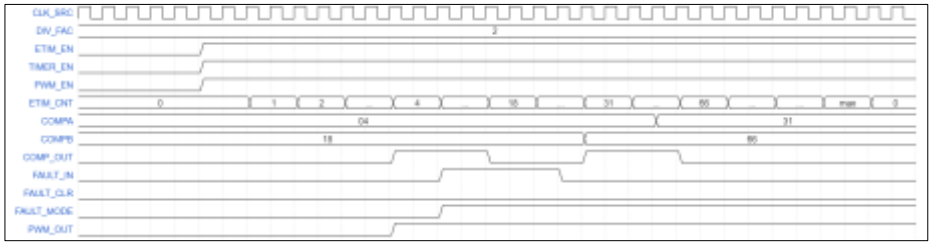


图16-16 Fault 模式 (One-shot) 下输出 1，无 FAULT_CLR

16.4.5.3 PWM 输出控制

14 个输出比较通道输出的 PWM，进入输出控制模块，进行互补、交换和极性控制后，最终输出 14 路 PWM 波形。Fault 模式下的输出优先级最高，且 Fault 模式也可以进行两两互补处理。

互补分组为固定分组：

- 第 0 组 (通道 0/1)；
- 第 1 组 (通道 2/3)；
- 第 2 组 (通道 4/5)；
- 第 3 组 (通道 6/7)；
- 第 4 组 (通道 8/9)；
- 第 5 组 (通道 10/11)；
- 第 6 组 (通道 12/13)；

Fault 模式时，对于奇数通道，根据互补模式选择偶数通道中输出的 Fault 模式有效信号，或者奇数通道的 Fault 模式有效信号，从而输出 Fault 模式下的输出极性配置。奇偶通道 Fault 模式下输出极性是独立配置的，不受互补模式影响。

非 Fault 模式时，对输出处理流程如下：

1. 奇数通道，根据互补模式选择偶数通道中比较输出的 COMP_OUT 信号取反，或者奇数通道的 COMP_OUT 信号；
2. 偶数通道，无论是否互补模式，都选择自己的 COMP_OUT 信号；
3. 根据 SWAP 配置，选择是否对奇数和偶数通道互补处理后的输出进行相互交换；
4. 然后根据输出取反使能配置，选择是否对上面交换处理后的输出进行取反后输出；

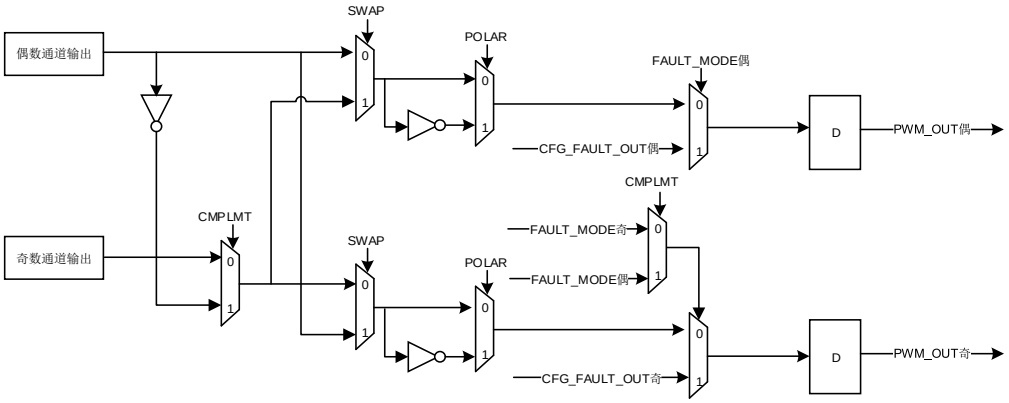


图16-17 第 x 组输出处理结构图

16.4.6 输入捕获

16.4.6.1 选择处理

- 支持对输入捕获信号进行 CAP 捕获，输入信号源如下：
  - 外部 GPIO 信号
  - 模拟比较器输出信号
  - Input XBAR、Output XBAR 输出信号
  - CLU 输出信号
  - SRPWM 输出事件信号
  - SDFM 输出事件信号
- 通用 CAP 捕获功能包括：
  - 上升沿捕获
  - 下降沿捕获
  - 正脉冲宽度捕获
  - 负脉冲宽度捕获
  - 周期捕获

捕获触发源支持 One-shot 与 Continuous 模式可配置。

16.4.6.2 输入捕获信号预分频

根据输入捕获信号预分频配置寄存器 (**CAP_CTRL. cap_pre_div**) 设置 2~62 分频，寄存器配置为 0 时，对输入信号 bypass 不分频处理。寄存器配置为 1 时，对输入信号 2 分频，寄存器配置为 31 时，对输入信号 62 分频。

输入信号 2 分频时序图 (配置 **CAP_CTRL. cap_pre_div** 为 **1**)。

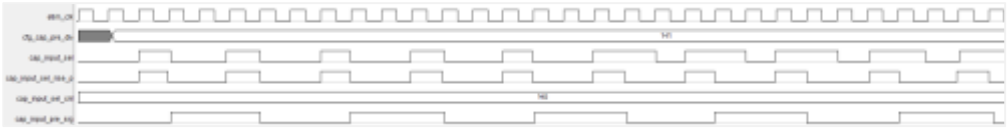


图16-18 经过滤波后的输入信号 2 分频

输入信号 4 分频时序图 (配置 **CAP_CTRL. cap_pre_div** 为 **2**)。

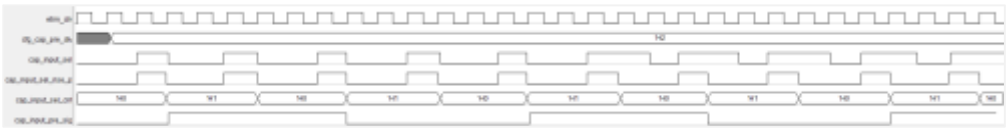


图16-19 经过滤波后的输入信号 4 分频

输入信号 6 分频时序图 (配置 **CAP_CTRL. cap_pre_div** 为 **3**)。

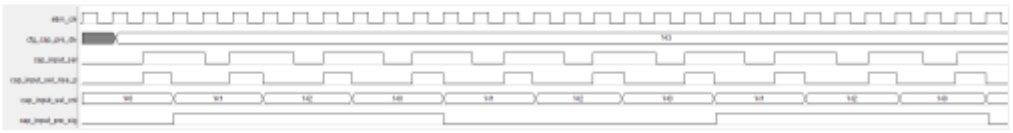


图16-20 经过滤波后的输入信号 6 分频

16.4.6.3 滤波处理

当 ETIMER 模块使能 (**CFG_ETIM_CMN0.etim_en**) 和 Filter 使能 (**CFG_ETIM_CMN2. cap_filter_en**) 都配置为 1 时，启动滤波功能。

滤波处理原理：

N 为 8bit 滤波宽度配置值，当连续 (N+1) 个 0 或者 1 输入时，滤波器输出值才对输入值进行采样。否则，维持上次值不变，滤波器输出值上电默认为 0。

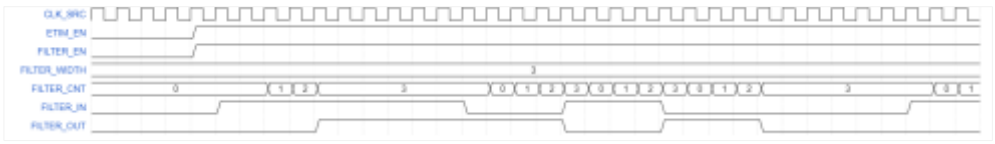


图16-21 滤波处理时序示意

16.4.6.4 捕获处理

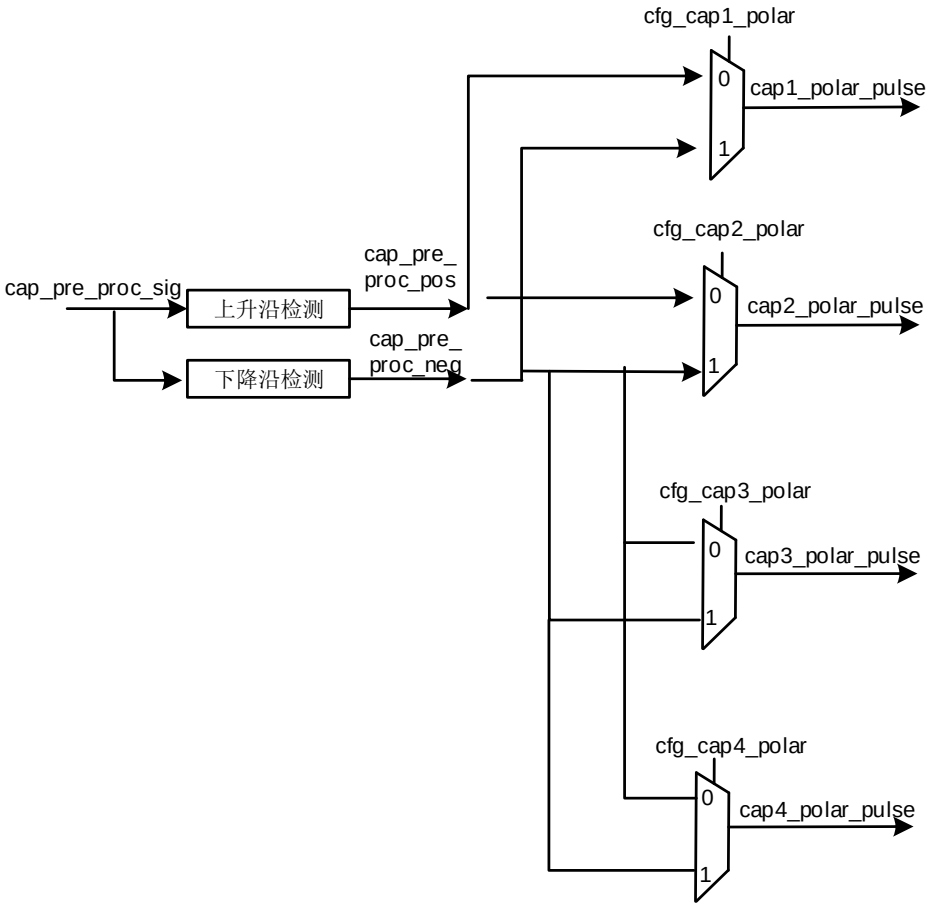
当 ETIMER 模块使能 (**CFG_ETIM_CMN0.cfg_etim_en**) 和 CAP 使能 (**CFG_ETIM_CMN1.cfg_etim_cap_en**) 都配置为 1 时，启动捕获功能。

CAP 捕获功能支持以下类型捕获：

- 上升沿捕获
- 下降沿捕获

- 正脉冲宽度捕获
- 负脉冲宽度捕获
- 周期捕获

经过预分频后的信号，CAP1，CAP2，CAP3，CAP4 做极性选择：



采用绝对时间戳模式下 CEVT1，CEVT2，CEVT3，CEVT4 4 个捕获事件下捕获时序图：
相关设置方式如下：

1. 极性设置：CAP1 设置为上升沿捕获，CAP2 设置为下降沿捕获，CAP3 设置为上升沿捕获，CAP4 设置为下降沿捕获。
2. 设置捕获方式: Continous 连续捕获
3. 时间戳计数方式：绝对捕获：**CAP_CTRL.cap1_clr_cnt**，**CAP_CTRL.cap2_clr_cnt** ，**CAP_CTRL.cap3_clr_cnt**，**CAP_CTRL.cap4_clr_cnt** 配置为不清零。
4. 循环方式：STOP_WRAP 设置为 3，MOD4 状态计数按照 MOD4_S0->MOD4_S1->MOD4_S2->MOD4_S3->MOD4_S0->MOD4_S1->MOD4_S2->MOD4_S3

时序图处理如下：

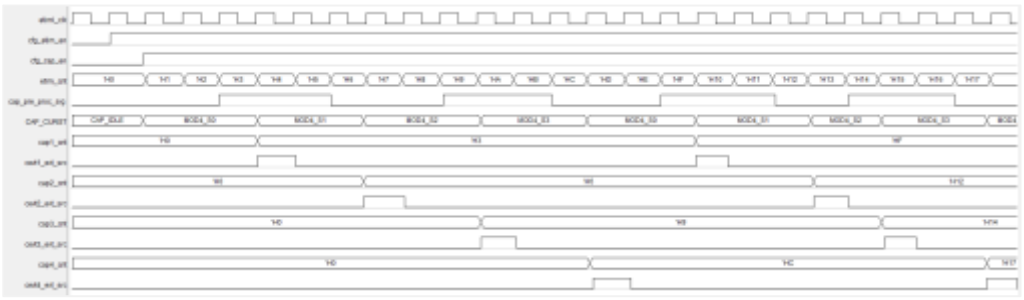


图16-22 绝对时间戳连续模式

采用相对时间戳模式下 CEVT1，CEVT2，CEVT3，CEVT4 4 个捕获事件下捕获时序图：相关设置方式如下：

1. 极性设置：CAP1 设置为上升沿捕获，CAP2 设置为下降沿捕获，CAP3 设置为上升沿捕获，CAP4 设置为下降沿捕获。
2. 设置捕获方式: 单次捕获 one -shot
3. 时间戳计数方式：相对捕获：**CAP_CTRL.cap1_clr_cnt**，**CAP_CTRL.cap2_clr_cnt** ，**CAP_CTRL.cap3_clr_cnt**，**CAP_CTRL.cap4_clr_cnt** 配置为清零。
4. 循环方式：STOP_WRAP 设置为 3，MOD4 状态计数按照 MOD4_S0->MOD4_S1-MOD4_S2->MOD4_S3

时序图处理如下：

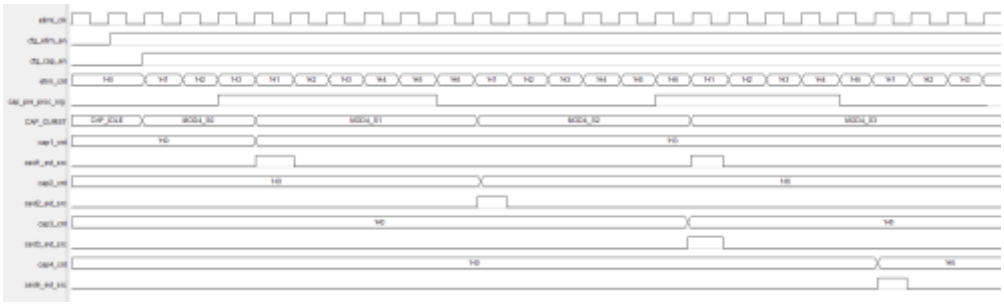


图16-23 相对事件单次模式

CAP MOD4 控制：

1. 状态机默认在 IDLE 初始状态，配置 **CFG_ETIM_EN1.cap_en** 为 1，MOD4 计数器进入 MOD4_S0。
2. 配置 **CAP_CTRL.cap_stop_wrap** 为 0，MOD4 计数一直处于 MOD4_S0。
3. 如果当前 **CAP_CTRL.cap_stop_wrap** 不为 0,并且检测到 CAP1 捕获事件 cap1_polar_pulse 时，MO4 计数器进入 MOD4_S1；当进去 MO4_S1 后，如果此时 **CAP_CTRL.cap_stop_wrap** 为 1，配置为连续模式时，将会进入 MOD4_S0，以便实现 MOD4_S0->MOD4_S1->MOD4_S0->MOD4_S1 的循环；如果配置为单次模式 MOD4 计数器将会停止，处于 MOD4_S1 状态。
4. 其余几个状态类推；通过 MOD4 计数器 STOP_WRAP 功能实现连续和单次模式。

- 单次模式下 STOP，可以执行如下几种情况：
  - ✓ MOD4_S0；
  - ✓ MOD4_S0->MOD4_S1；
  - ✓ MOD4_S0->MOD4_S1->MOD4_S2；
  - ✓ MOD4_S0->MOD4_S1->MOD4_S2->MOD4_S3；
- 连续模式下 WRAP，可以执行如下几种情况：
  - ✓ MOD4_S0->MOD4_S0；
  - ✓ MOD4_S0 -> MOD4_S1->MOD4_S0->MOD4_S1；
  - ✓ MOD4_S0->MOD4_S1->MOD4_S2->MOD4_S0->MOD4_S1->MOD4_S2
  - ✓ MOD4_S0->MOD4_S1->MOD4_S2->MOD4_S3->MOD4_S0->MOD4_S1->MOD4_S2->MOD4_S3；

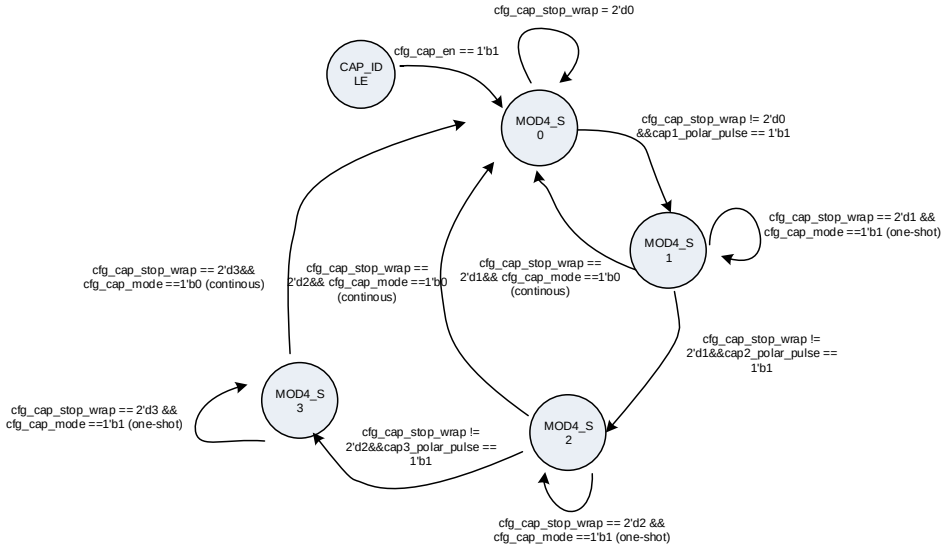


图16-24 MOD4 计数状态机

16.4.7 定时器同步

增强型定时器中的 14 个通道使用独立的计数器，用于产生 PWM 波和对输入信号的捕获。为了实现 14 个通道的计数器同步开启或者关闭，有两种办法，第一是将 ETIMER 的模块使能和计数器的使能控制放在同一个配置寄存器（**CFG_ETIM_CMN0**）中进行控制。

- ETIMER 模块使能（CFG_ETIM_CMN0.cfg_etim_en）
- 计数器使能（CFG_ETIM_CMN0.cfg_etim_timer_en）

第二是通过 SYNC 同步信号将 14 个通道的内部计数器同步。SYNC 同步信号来自如下几个模块：

- SRPWM：外部同步信号，200 MHz 单周期脉冲；
- Input XBAR：外部同步信号，200 MHz 单周期脉冲；
- ETIMER：内部同步信号，200 MHz 单周期脉冲；

说明

Input XBAR 本身不输出同步源给 ETIMER，此处的 Input XBAR 的同步源来自 Input XBAR 的输入捕获源低 6 位。

来源	类别
SRPWM	12
ETIMER	14
Input XBAR	6
合计	32

输入同步信号选择：

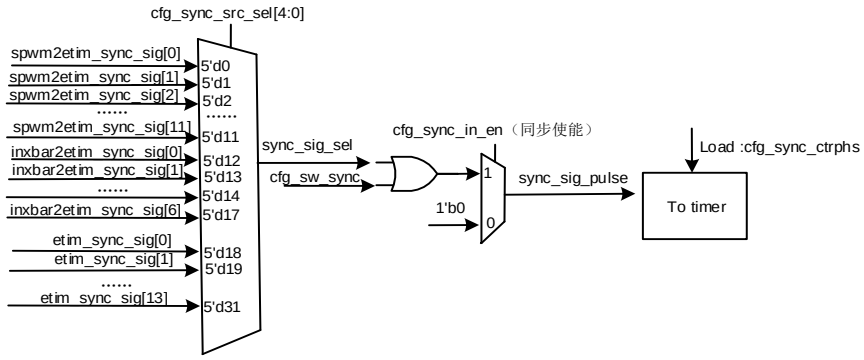


图16-25 同步信号输入选择图

16.4.8 事件与中断

16.4.8.1 事件

当 ETIMER 模块使能 (**CFG_ETIM_CMN0.cfg_etim_en**) 和事件输出使能 (**CFG_ETIM_CMN3.cfg_etim_event_en**) 都配置为 1，启动事件输出功能。事件输出信号宽度为 16 个 ETIMER 模块输入时钟 (预分频前) 周期。

ETIMER 模块支持以下 13 个时间戳事件：

- 计数到 0 时间戳事件
- 计数到周期值时间戳事件
- 计数到 CMPA 时间戳事件
- 计数到 CMPB 时间戳事件
- 计数到 EVT 时间戳事件
- 触发 ADC 采样事件
- 触发 DACC 发波事件
- 触发 CMPC 内部 DAC 发波事件
- 捕获 CEVT1、CEVT2、CEVT3、CEVT4 事件
- 同步输出事件

其中事件预装载寄存器 EVT 支持立即加载和影子加载，EVT 寄存器影子加载事件产生

时刻为计数器计数值与 SHWD 寄存器配置值相等时刻。

16.4.8.2 中断

ETIMER 模块 14 通道可独立产生中断输出，共 14 个中断输出，每个通道的中断源相同。

单个通道中断源如下：

- PWM 在 One-shot 模式下，Fault 模式有效
- PWM 在 CBC 模式下，Fault 模式有效
- Fault 模式未退出，再次收到有效 Fault 源信号
- COMPA 和 COMPB 配置错误，COMPA≥COMPB
- 计数到 COMPA 溢出事件
- 计数到 COMPB 溢出事件
- 计数到配置周期事件
- 计数器最大值溢出事件
- 捕获 CEVT1、CEVT2、CEVT3、CEVT4 事件

16.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 ETIMER 章节。

17 Σ-Δ 滤波器模块 (SDFM)

17.1 简介

Σ-Δ 滤波器模块 (SDFM) 是一种四通道数字滤波器，专为电机控制应用中的电流测量和解析器位置解码而设计。每个输入通道可以接收一个独立的 delta-sigma(Δ-Σ)调制器比特流。比特流由四个单独可编程的数字抽取滤波器处理。该滤波器集包括一个主滤波器 (数据滤波器)和一个快速比较器 (第二滤波器)，快速比较器用于过流和欠流监测的即时数字阈值比较，以及过零检测。

17.2 结构框图

SDFM 结构框图如下：

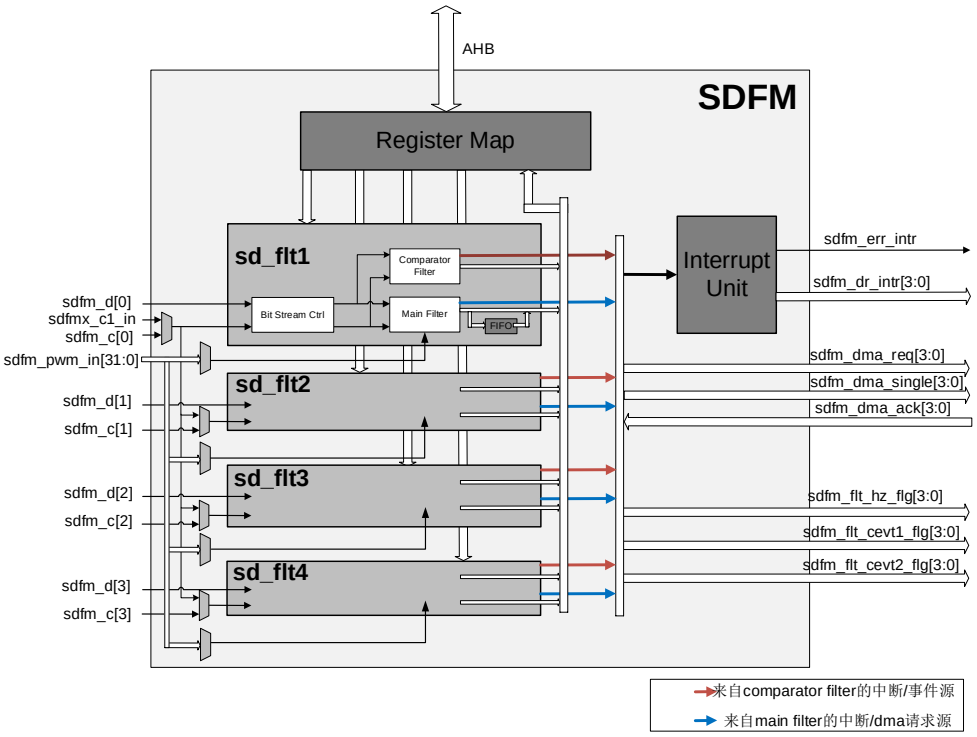


图17-1 SDFM 结构框图

17.3 主要特性

- 每个 SDFM 模块 8 个外部引脚
  - 每个 SDFM 模块有 4 个 Σ-Δ 数据输入引脚 (sdfm-d[x]，其中 x=0 至 3)；
  - 每个 SDFM 模块有 4 个 Σ-Δ 时钟输入引脚 (sdfm-c[x]，其中 x=0 至 3)；

- 支持一种调制器时钟模式：
  - 调制器时钟速率等于调制器数据速率；
- 每个 SDFM 模块有 4 个独立的可配置第二滤波器 (比较器) 单元：
  - 提供 4 个不同的滤波器类型选择 (Sinc1/Sinc2/SincFast/Sinc3) 选项
  - 能够检测高值条件，低值条件和阈值越过条件
    - 2 个独立的更高阈值比较器 (被用作检查高值条件)
    - 2 个独立的更低阈值比较器 (被用作检查低值条件)
    - 1 个独立的阈值越过比较器 (被用 ETIMER 测量占空比/频率)
  - 比较器滤波器单元的 OSR 值 (COSR) 可编程为 1 至 32
- 每个 SDFM 模块有 4 个独立的可配置主滤波器 (数据滤波器) 单元：
  - 提供 4 个不同的滤波器类型选择 (Sinc1/Sinc2/SincFast/Sinc3) 选项
  - 数据滤波器单元的 OSR 值 (DOSR) 可编程为 1 至 256
  - 能够启用或禁用独立的滤波器模块 (或全部)
  - 能够使用主滤波器使能位或 PWM 同步信号 (sdfm_pwm_in) 来同步 SDFM 模块的所有 4 个独立滤波器
- 数据滤波器输出可以用 16 位或 32 位表示 (有效位为 26 位)。
- 数据滤波器单元具有可编程模式 FIFO 来减少中断开销。该 FIFO 具有以下特性：
  - 主滤波器 (数据滤波器) 具有一个 16 深度 x26 位 FIFO；
  - FIFO 可在达到可编程数量的数据就绪事件后中断 CPU；
  - FIFO 等待同步功能：能够忽略数据就绪事件，直至接收到 PWM 同步信号一旦接收到 PWM 同步信号，就会在每个数据就绪事件时填充 FIFO；
  - 数据滤波器输出可以用 16 位或 32 位表示；
- 每个数据滤波器通道可以单独配置 sdfm_pwm_in 中的 1 位为 PWM 的同步信号源
- 可使用 PWM 的输出作为外部 Σ-Δ 模拟调制器的时钟
- sdfm-c[x]和 sdfm-d[x]需要在模块内部做滤波处理
- 能够使用一个滤波器通道时钟 (sdfm-c[0]) 为其他滤波器时钟通道提供时钟
- 在发生比较器滤波器事件时可以使用可配置的数字滤波器来清除杂散噪声引起的比较器事件

17.4 功能描述

每个 SDFM 模块有 4 个独立的滤波器模块。这些滤波器模块是相同的，可以独立配置。

每个单独的滤波器模块有以下单元：

- 比特流控制单元；
- 主 (数据) 滤波器单元；
- 比较 (第二) 滤波器单元与 5 个独立的比较器；

SDFM 模块框图如图 17-1 所示。

图 17-2 所示的每个滤波器模块都有 1 个主 (数据) 滤波器和 1 个辅助 (比较器) 滤波器对，它们接收相同的比特流。

除了输入比特流，主滤波器和第二滤波器都是完全相互独立的。每个滤波器模块都可以独立配置。因此，在 SDFM 模块中，总共有 4 个主滤波器和四个辅助滤波器。

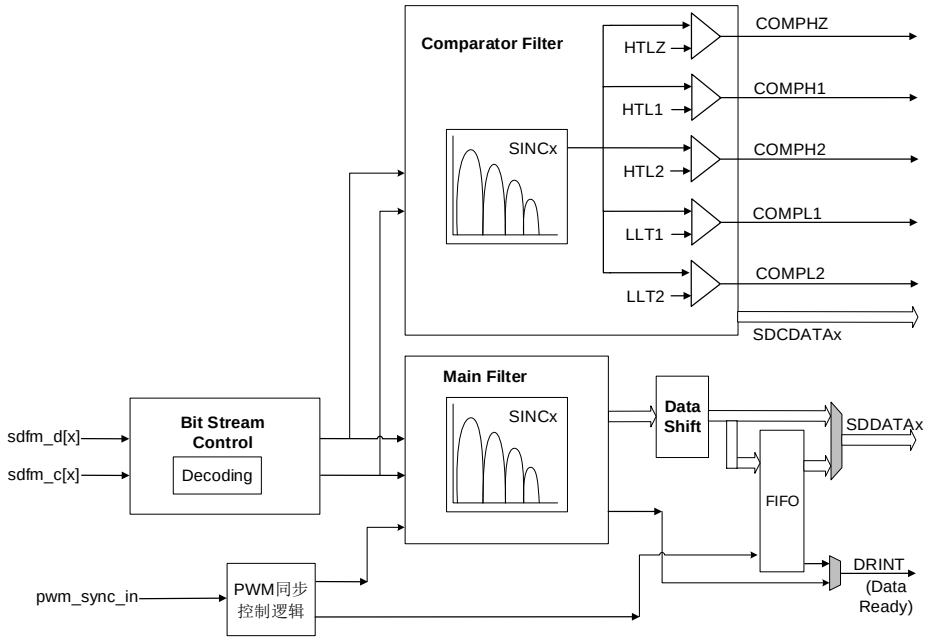


图17-2 滤波器模块的框图

17.4.2 输入同步特性

GPIO 输入为 ASYNC，请确保检查 SDFM 电气数据和时序 (使用 ASYNC) 要求是否满足，并注意以下警告信息。

⚠ 警告

SDFM 时钟输入 (sdfm_c[x]) 直接为 SDFM 模块 sdfm_d[x]的伴随时钟。这些输入上的任何毛刺或振铃噪声都可能破坏 SDFM 模块的操作。应对这些信号采取特别预防措施，以确保信号干净、无噪声，符合 SDFM 时序要求。

建议采取预防措施，如由于时钟驱动器的任何阻抗不匹配而导致振铃的串联终止，以及与其他噪声信号的走线间隔。

⚠ 注意

脉冲噪声源 (例如电磁干扰、串扰等)，有可能破坏 sdfm_c[x]和 sdfm_d[x]位流，导致 SDFM 滤波后的数据损坏。模块内部使用 sdfm_clk 同步 sdfm_c[x]和 sdfm_d[x]信号以减轻这种脉冲噪声影响。

17.4.3 比特流控制单元

比特流控制单元接收 Σ-Δ 调制数据和 Σ-Δ 调制时钟。所接收的调制数据被捕获并传递到数据滤波单元和比较器单元。该单元可以接收调制器时钟速率和调制器数据速率相同，如图 17-3 所示。

⚠ 注意

为了达到最大值，sigma-delta 调制器必须在绝对最大正或负满量程下工作，这超出了大多数 sigma-delta 调制器推荐的 80%的满量程范围。

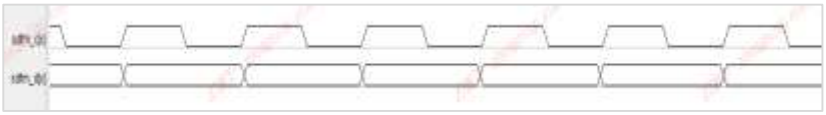


图17-3 调制器时钟和数据

17.4.4 SDFM 调制器时钟

在系统中，可以使用 PWM 来生成调制器时钟。假设所有的 **sdfm_c[x]**在板上的线路延迟相同，可以使用一个调制器时钟来驱动多个滤波器，从而减少用于 SDFM 的引脚数量。**sdfm_c[0]**可以被配置送到其他滤波器通道。可以通过配置 **cfg_sdclksel** 寄存器来选择各个滤波器通道的调制器时钟。

17.4.5 Sinc 滤波器

SDFM 中可用的比较器滤波器和数据滤波器都以 SincN 滤波器为核心。SincN 滤波器本质上是一个低通滤波器，它通过数字滤波和抽取将输入比特流转换为数字数据。

这个滤波后的数字数据表示给 Σ-Δ 调制器的模拟输入。简化的 SincN 架构由级联的积分器和微分器组成，由下采样器分隔，如图 17-4 所示。N 阶 Sinc 滤波器的 z 传递函数如图 17-5 所示。

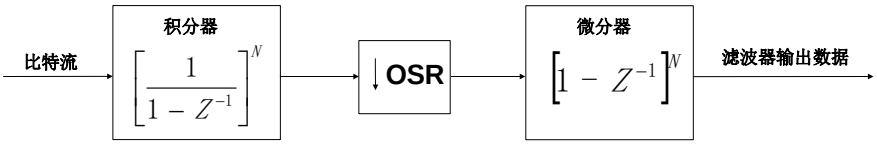


图17-4 Sinc 滤波器的简化框图

$$H(Z) = \left[\frac{1 - Z^{-OSR}}{1 - Z^{-1}}\right]^N$$

N： 滤波器的阶数

OSR：过采样率

图17-5 N 阶 Sinc 滤波器的传递函数

Sinc 滤波器的有效分辨率(ENOB)取决于滤波器类型、OSR 和 Σ-Δ 调制器频率。通常，对于给定的滤波器类型，更高的 OSR 可以实现更高的分辨率或 ENOB；然而，代价是增加了滤波器延迟。需要折中考虑最佳速度与分辨率，以选择合适的 Σ-Δ 调制器。参考相应的 Σ-Δ 调制器数据表来确定给定 Sinc 滤波器配置的有效分辨率。图 17-6 显示了当 OSR = 32 并且 Σ-Δ 调制器频率为 10 MHz 时，不同滤波器结构的频率响应。

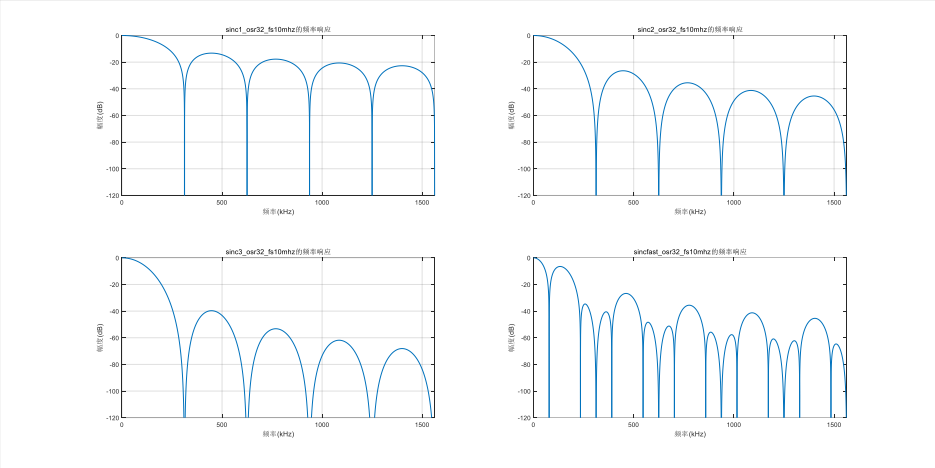


图17-6 不同 Sinc 滤波器的频率响应

不同 Sinc 滤波器的阶数被显示在表 17-1。

表17-1 Sinc 滤波器的阶数

滤波器类型	Sinc 滤波器阶数
Sinc1	1
Sinc2	2
Sinc3	3
SincFast	3

17.4.5.1 Sinc 滤波器的数据速率和延迟

Sinc 滤波器的数据速率(滤波器吞吐量)以采样/秒表示，计算公式如下:

$$\text{Sinc 滤波器的数据速率}=\frac{\text{调制器的数据速率}}{\text{OSR}} \quad (17\text{-}1)$$

Sinc 滤波器的延迟 (以秒为单位) 定义为 Sinc 滤波器在初始化时产生正确滤波输出所需的时间。

对于给定的滤波器类型，延迟由以下公式计算:

$$\text{Sinc 滤波器的延迟}=\frac{\text{Sinc 滤波器的阶数}}{\text{Sinc 滤波器的数据速率}} \quad (17\text{-}2)$$

例如：

Sinc 滤波器的类型：Sinc3；

调制器数据速率：10 MHz；

OSR：256；

Sinc 滤波器的数据速率：10 MHz/256=39.1K samples/sec；

Sinc 滤波器的延迟：76.8 us；

Sinc 滤波器的类型：Sinc2；

调制器数据速率：10 MHz；

OSR：256；

Sinc 滤波器的数据速率：10 MHz/256=39.1K samples/sec；

Sinc 滤波器的延迟：51.2 us；

17.4.5.2 主(数据)滤波器单元

数据滤波器是一个可配置的 Sinc 滤波器，支持以下滤波器类型:Sinc1, Sinc2, Sinc3 和 SincFast。数据滤波器 OSR (DOSR) 设置可以配置为 1 到 256，独立于比较器滤波器。数据滤波器的有效分辨率 (ENOB) 取决于数据滤波器类型、DOSR 和 Σ-Δ 调制器频率。数据滤波器输出以 26 位带符号整数表示，采用二进制补码格式。该滤波器单元将低输入信号转换为“-1”，将高输入信号转换为“1”。结果计算为数据滤波器的输出提供正负值。表 17-2 显示了数据滤波器使用不同 OSR 可以存储的不同满刻度值。

表17-2 不同 Sinc 滤波器在不同 DOSR 值下滤波器输出的峰值

DOSR	Sinc1	Sinc2	Sinc3	SincFast
x	x	x^2	x^3	$2x^2$
4	-4~3	-16~15	-64~63	-32~31
8	-8~7	-64~63	-512~511	-128~127
16	-16~15	-256~255	-4096~4095	-512~511
32	-32~31	-1024~1023	-32768~32767	-2048~2047
64	-64~63	-4096~4095	-262144~262143	-8192~8191
128	-128~127	-16384~16383	-2097152~2097151	-32768~32767
256	-256~255	-65536~65535	-16777216~16777215	-131072~131071

17.4.5.2.1 32 位或 16 位数据滤波器输出表示

数据滤波器的输出可以表示为 32 bit 或者 16 bit 格式。

32 bit 数据滤波器表示：

- 当 cfg_dr=1，数据滤波器输出以 32 位格式表示。在此配置中，移位控制位的写入对数据滤波器的输出没有任何影响。

16 bit 数据滤波器表示：

- 默认情况下，cfg_dr=0，数据滤波器输出以 16 位格式表示。用户需要配置移位控制位 (cfg_sh 寄存器) 以控制滤波器输出的 26 位数据中哪 16 位有效数据输出。

  例如，对数据滤波器进行如下配置:

  - 过滤器类型 = Sinc3；
  - OSR = 128；

- cfg_dr = 0；

数据滤波器的 26 位带符号输出值取值范围为-16777216~16777215。但是，16 位带符号输出只支持从-32,768 到 32,767 的值。因此，需要将移位控制位 (cfg_sh 寄存器) 配置为 7，以正确表示 16 位格式的数据输出。不同 OSR 和滤波器类型的移位控制位配置如表 17-3 所示。

表17-3 移位控制位配置设置

OSR	Sinc1	Sinc2	Sinc3	SincFast
1~31	0	0	0	0
32~40	0	0	1	0
41~50	0	0	2	0
51~63	0	0	3	0
64~80	0	0	4	0
81~101	0	0	5	0
102~127	0	0	6	0
128~161	0	0	7	1
162~181	0	0	8	1
182~203	0	1	8	2
204~255	0	1	9	2
256	0	2	10	3

说明

移位控制位配置不正确将导致得到错误的 16 位数据滤波器输出。

17.4.5.2.2 Data FIFO

每个主 (数据) 滤波器通道都有一个 16-level 深度，26-bit FIFO。

FIFO 可以通过配置“水线”来减少中断开销。

默认情况下，FIFO 操作被失能，可以通过配置 FIFO 使能寄存器来使能。当 FIFO 被使能时，来自数据滤波器的每个数据就绪事件都会填充 FIFO，FIFO 的状态通过对应的状态寄存器显示。

- **设置 FIFO 在接收可编程数量的数据后中断:**
  - 启用 SDFM FIFO；
  - 启用 SDFM FIFO 中断；
  - 配置 FIFO “水线”；
  - 配置 SDFM 数据就绪中断源选择 FIFO 数据就绪中断 (设置 cfg_drintsel=1)，当 FIFO 中接收数据的个数大于 FIFO “水线”时，产生数据就绪中断；
- **等待同步功能:**

  FIFO 等待同步功能可用于忽略来自数据滤波器的数据就绪事件，直到 PWM 同步事件到来。

  默认情况下，等待同步功能处于关闭状态，该特性可以通过设置 cfg_wtsyncen=1 来启用。

- **失能等待同步功能时:**

  每个数据就绪事件都会填充 FIFO，直到 FIFO 满。

- **当等待同步功能使能时:**

  在 FIFO 接收到 PWM 同步事件之前，数据就绪事件不会填充 FIFO。在 PWM 同步事件到来时，FIFO 设置 wtsynflg_rpt=1 并且来自主滤波器的数据就绪事件开始填充 FIFO，直到 FIFO 满。wtsynflg_rpt 可以自动或手动清除。

  当 wtsynflg_rpt=0 时，FIFO 内容被冻结，直到下一个 PWM 同步事件到来之后数据就绪事件才填充 FIFO。

- **wtsynflg_rpt 自动清除模式:**

  默认情况下，开启该模式。当 cfg_wtsclren=1 时，wtsynflg_rpt 在 FIFO 数据就绪中断时自动清除。

- **wtsynflg_rpt 手动清除模式:**

  当 cfg_wtsclren=0 时，设置 cfg_wtsynclr=1 可用于手动清除 wtsynflg_rpt。

- **清除 FIFO 内容:**

  FIFO 内容可以通过以下任何一种方法清除:

  - 失能 FIFO 清除 FIFO 内容。
  - 失能主滤波器清除 FIFO 内容。
  - FIFO 内容也可以在接收 PWM 同步事件时自动清除。默认情况下，该特性未使能，可通过配置 cfg_ffsynclren=1 使能。只有启用了等待同步功能(cfg_wtsyncen=1)，才能启用该功能。

- **FIFO 调试访问行为:**

  对 fifo_data 寄存器的调试访问不影响 FIFO 写指针。在 CPU 访问 fifo_data 寄存器时，FIFO 读指针将前进到 FIFO 中的下一个可用地址。

17.4.5.2.3 PWM 同步事件

主 (数据) 滤波器可以根据 PWM 同步事件进行同步。来自 PWM 模块的 sdfm_pwm_in 信号用于复位 DOSR 计数器。该特性在默认情况下是失能的，可以通过设置 cfg_sdsyncen=1 来使能。每个主滤波器可以通过配置 cfg_syncsel 寄存器来选择 12 个 sdfm_pwm_in 信号中的任意一个。

注意:

请确保每个 PWM 定时周期只生成一个 PWM 同步事件。在 PWM 上计数或下计数模式下可以自动确保只生成一个 PWM 同步事件。但是，如果使用上下计数模式，应保证每个 PWM 周期只产生一个 PWM 同步事件；否则，每个 PWM 周期提供两个脉冲将破坏 SDFM 主滤波器的定时。

由于 Sinc 滤波器的固有架构 (Sinc1, Sinc2, Sinc3, SincFast)，前几个滤波器输出数据是不正确的。在以下条件下，错误数据个数如表 17-4 所示:

- 首次使能并配置 Sinc filter 时。
- 当 Sinc 滤波器被使能，并在操作过程中重新使能或重新配置。
- 当数据滤波器通过 PWM 同步事件进行同步时。

表17-4 滤波器输出的不正确数据个数

滤波器类型	滤波被使能和配置后输出的不正确数据个数
Sinc1	没有不正确的数据
Sinc2	第一个数据不正确
Sinc3	开始两个数据不正确
SincFast	开始两个数据不正确

说明

SDFM 比较滤波器的中断只有在提供足够的稳定时间后才能启用，以确保比较滤波器不会在这些不正确的样本上跳闸。这个足够的延迟等于比较滤波器的延迟和 5 个 sdfm_c[x]伴随时钟周期。

17.4.5.3 比较 (第二) 滤波器单元

大多数控制系统需要在电流或电压越界时通过跳闸 PWM 来保护系统。比较(第二)滤波器的主要目的是允许用户以快速的稳定时间监视输入条件。这允许用户跳闸 PWM，以保护系统免受潜在的损害。

注意：比较 (第二) 滤波器不能用 PWM 同步事件同步。

比较器滤波器是一个可配置的 Sinc 滤波器，支持以下滤波器类型:Sinc1、Sinc2、Sinc3 和 SincFast。比较器 OSR (COSR) 设置可以配置为 1 到 32，并且独立于数据滤波器。比较器滤波器的有效分辨率 (ENOB) 取决于比较器滤波器类型、COSR 和 Σ-Δ 调制器频率。默认情况下，比较器滤波器是失能的，通过配置对应的使能寄存器使能。比较器滤波器输出以 16 位无符号格式表示。该滤波器单元将低输入信号转换为“0”，将高输入信号转换为“1”，比较滤波器的计算结果输出正值。表 17-5 显示了比较器滤波器可以使用不同的 OSR 存储的不同满量程值。

表17-5 不同 Sinc 滤波器在不同 OSR 值下滤波器输出的范围

OSR	Sinc1	Sinc2	Sinc3	SincFast
x	0~x	0~x^2	0~x^3	0~$2x^2$
4	0~4	0~16	0~64	0~32
8	0~8	0~64	0~512	0~128
16	0~16	0~256	0~4096	0~512
32	0~32	0~1024	0~32768	0~2048

比较器滤波器的输出是内存映射的，可以在 **sdcdata_rpt** 寄存器中读取，每次在 COSR 个 **sdfm_c[x]**周期被更新。比较滤波器的输出连接到数字比较器。

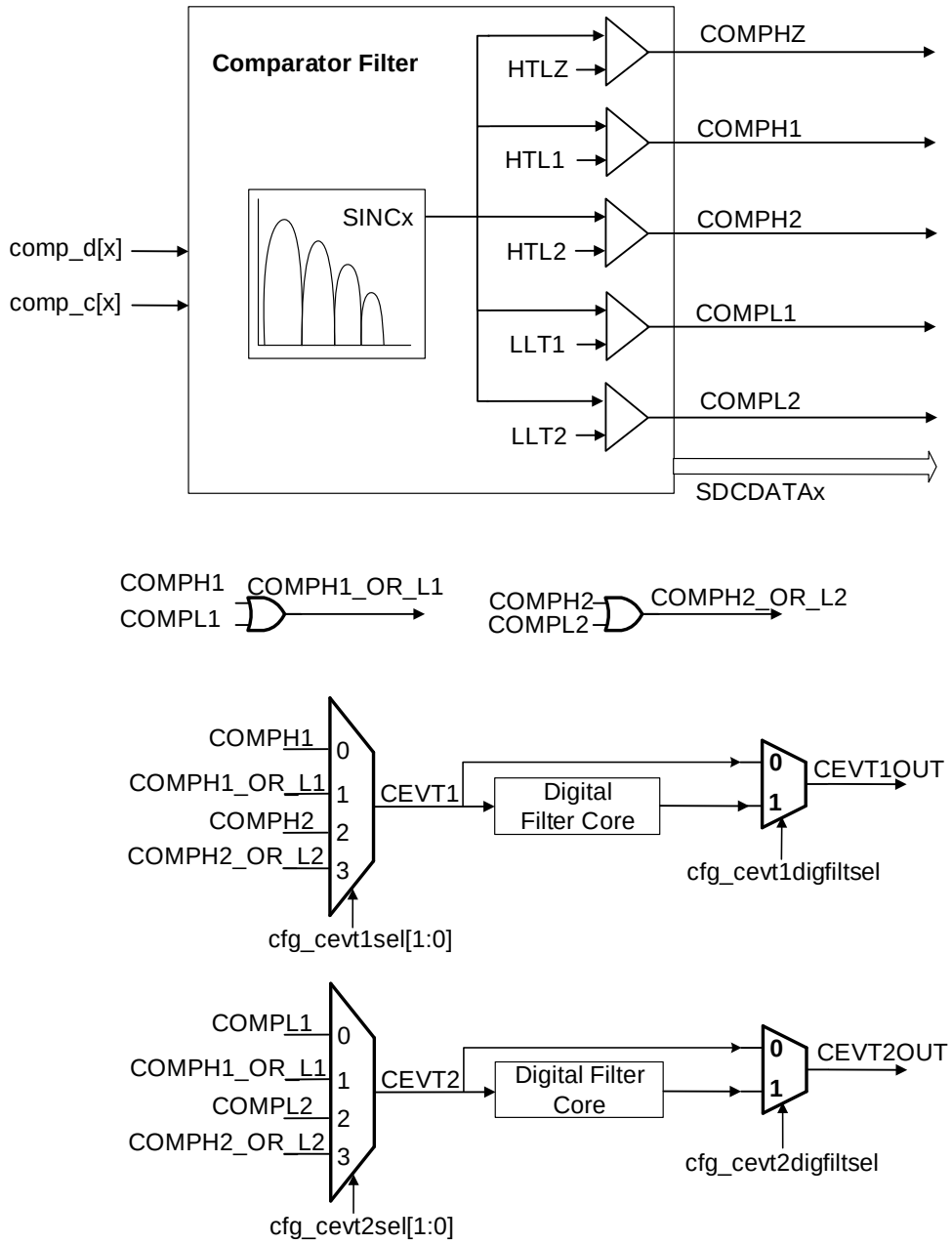


图17-7 比较器滤波器结构

17.4.5.3.1 高阈值 (HLT) 比较器

- 高阈值比较器可用于检测超值情况。
- 当比较器数据>=高阈值寄存器时，生成高阈值事件。
- 除了 COMPHZ 之外，高阈值比较器事件可以配置为触发以下事件：CPU 中断，PWM 跳闸。
- 高阈值比较器事件可以与 ETIMER 一起使用，以测量阈值跨越的频率/占空比
- 该设备有三个高阈值比较器:

高阈值 1 (HLT1) 比较器:

- 当比较器数据>=高阈值 1 寄存器 (cfg_hlt_val) 时，生成 COMPH1 事件。

- COMPH1 事件连接到 CEVT1 和 CEVT2。

高阈值 1 (HLT2) 比较器:

- 当比较器数据>=高阈值 2 寄存器 (cfg_hlt2_val) 时，生成 COMPH2 事件。
- COMPH2 事件连接到 CEVT1 和 CEVT2。

高阈值 (HTLZ) 比较器:

- 当比较器数据>=高阈值 (HTLZ) 寄存器 (cfg_hltz_val)，它可以生成高阈值 (HTLZ) 事件 (COMPHZ) 并设置相应的状态标志。但是，不能将此事件配置为生成 SDFM 中断。高阈值 (HTLZ) 比较器可以与 ETIMER 一起使用，以测量阈值越过事件的频率/占空比。

  例：为了测量过零事件的频率/占空比，用户需要配置 **cfg_hltz_val=0x4000** 并使能高电平高阈值 (HTLZ) 事件，然后将高阈值 (HTLZ) 事件连接到 ETIMER 进行频率/占空比测量。

17.4.5.3.2 低阈值 (LLT) 比较器

低阈值比较器可用于检测低值状态。

- 当比较器数据<=低阈值寄存器时，产生低阈值事件。
- 低阈值比较器事件可以配置为触发以下事件：CPU 中断，PWM 跳闸。
- 低阈值比较器事件可以与 ETIMER 一起使用，以测量阈值越过事件的频率/占空比
- 该设备有两个低阈值比较器:

低阈值 1 (LLT1) 比较器:

- 当比较器数据<=低阈值 1 寄存器 (cfg_llt_val) 时，生成 COMPL1 事件。
- COMPL1 事件连接到 CEVT1 和 CEVT2。

低阈值 1 (LLT2) 比较器:

- 当比较器数据<=低阈值 2 寄存器 (cfg_llt2_val) 时，生成 COMPL2 事件。
- COMPL2 事件连接到 CEVT1 和 CEVT2。

17.4.5.3.3 数字滤波器

数字滤波器首先将输入数据采样到 (cfg_sampwin+1) 个 FIFO 样本窗口上，然后滤波器再解析出样本窗口的大多数数值，其中大多数由阈值 (cfg_thresh) 值定义。如果大多数阈值未满足，则滤波器输出保持不变。

为了正常运行，cfg_thresh 的值必须大于 cfg_sampwin/2。

预分频功能 (cfg_clkprescale) 确定滤波器的采样率，其中滤波器 FIFO 每隔 cfg_clkprescale 个系统时钟捕获一个样本。来自 FIFO 的旧数据将被丢弃。

数字滤波器的概念模型如图 17-8 所示。

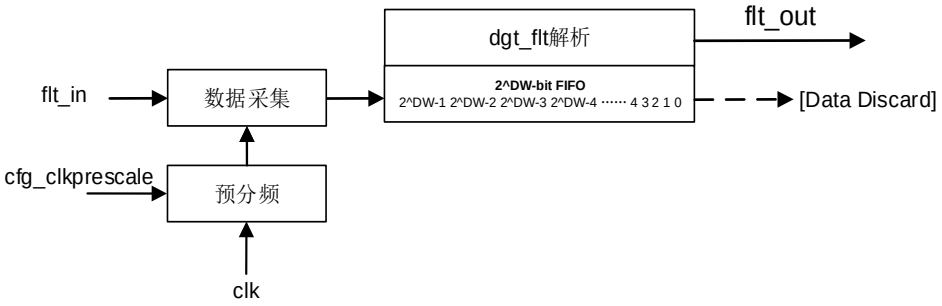


图17-8 数字滤波器的概念模型

滤波器实现的的等效 c 代码如下所示：

```
if(flt_out == 0) {
      if(Num_1s_in_SAMPWIN >= cfg_thresh){
            flt_out = 1;
      }
}
else {
      if(Num_0s_in_SAMPWIN >= cfg_thresh){
            flt_out = 0;
      }
}
```

数字滤波器的初始化顺序

为了确保 dgt_flt 的正常运行，建议按照以下初始化顺序进行操作:

1. 配置数字滤波器参数：
   - 设置 cfg_sampwin，确定 FIFO 窗口中要监测的样本数量。
   - 设置 cfg_thresh，确定大多数数值所需的阈值。
   - 设置 cfg_clkprescale，确定数字滤波器时钟预分频值。
2. 通过设置 cfg_filinit 来初始化数字 FIFO 窗口中的样本值。
   - 当 cfg_filinit=0 时，FIFO 中的样本值初始化为 0；
   - 当 cfg_filinit=1 时，FIFO 中的样本值初始化为 cfg_sampwin 个 flt_in 的采样值；

17.4.6 SDFM 滤波器的理论输出值

下面的公式可以用来推导出一个比较器滤波器和数据滤波器的理论输出值。

$$\text{比特流中 1 的密度}=\frac{\text{输入电压}+\text{Vclipping}}{2\times\text{Vclipping}} \quad (17\text{-}3)$$

其中：

- Vclipping=调制器的最大差分电压输入范围；

- 输入电压=施加到调制器的差分输入电压；

比较滤波器的输出 (理论上) = 比特流中 1 的密度×比较滤波器的最大输出 (滤波器类型，COSR) (17-4)

$$\text{主滤波器的输出 (理论上)} = \left\{\frac{\text{abs（输入电压）}}{\text{Vclipping}}\right\} \times \text{主滤波器的最大输出 (滤波器类型，DOSR)} \quad (17\text{-}5)$$

$$\text{主滤波器的输出 32 位 (理论上)} = \begin{cases} \text{主滤波器输出 如果输入电压为+ve V} \\ \text{2 的补码} \quad \text{如果输入电压为-ve V} \end{cases} \quad (17\text{-}6)$$

滤波器的输出 16 位 (理论上) = 主滤波器的输出 32 位(理论上)≫移位值 (滤波器类型，DOSR) (17-7)

例，当使用 AMC1306x25 调制器时：

AMC1306x25	Vclipping = 输入电压 (AINP-AINN)	320 mV 100 mV
SDFM 滤波器设置	滤波器类型 = COSR = DOSR =	3 32 100

比特流中 1 的密度	用等式 (17-3)	0.65625
比较滤波器输出 滤波器类型：Sinc3 COSR：32	用等式 (17-4)	21504
主滤波器输出 (32 位) 滤波器类型：Sinc3 COSR：100	用等式 (17-5) 和 (17-6)	312500
主滤波器输出 (16 位) 滤波器类型：Sinc3 COSR：100 (正确的移位值为 5)	用等式 (17-7)	9765

17.4.7 中断

每个 SDFM 有 5 个 CPU 中断，分别为 SDFM 错误中断 (sdfm_err_intr) 和每个主滤波器模块的数据就绪中断 (sdfm_dr_intr[3:0])。

17.4.7.1 错误中断 (sdfm_err_intr) 源

SDFM 的错误中断 (sdfm_err_intr) 来源于 4 个通道，每个通道 6 个，包括比较器事件 1 (CEVT1)、比较器事件 2 (CEVT2)、调制器故障 (MF) 事件、FIFO 溢出 (SDFFOVF) 事件、FIFO 写错误事件、FIFO 读错误事件。

sdfm_err_intr 中断可以被这 24 个事件中的任何一个触发。

1. **比较器事件 1 (CEVT1)**

来自四个比较器滤波器模块中的任何一个的 CEVT1 事件都可以触发 CPU 中断。该事件可以配置为错误中断 (sdfm_err_intr)，需设置以下配置:

- 使能主中断 (cfg_mie=1)；
- 使能对应比较滤波器的 CEVT1 中断；

在 CEVT1 事件中，CEVT1 的状态位也会被设置。在中断源不再活动时设置其对应的清除位，可以清除该状态位。

2. **比较器事件 2 (CEVT2)**

来自四个比较器滤波器模块中的任何一个的 CEVT2 事件都可以触发 CPU 中断。该事件可以配置为错误中断 (sdfm_err_intr)，需设置以下配置:

- 使能主中断 (cfg_mie=1)；
- 使能对应比较滤波器的 CEVT2 中断；

在 CEVT2 事件中，CEVT2 的状态位也会被设置。在中断源不再活动时设置其对应的清除位，可以清除该状态位。

3. **调制器故障 (MF) 事件**

sdfm_c[x]丢失时产生调制器故障(MF)。如果 sdfm_c[x]在 192 个 sdfm_clk 系统时钟（200MHz）周期不翻转，则认为调制器时钟缺失。来自四个滤波器模块中的任何一个的 MF 事件都可以触发 CPU 中断。该事件可以配置为错误中断 (sdfm_err_intr)，需设置以下配置:

- 使能主中断 (cfg_mie=1)；
- 使能调制器时钟故障中断源；

在调制器故障(MF)事件中，调制器故障(MF)的状态位也会被设置。在中断源不再活动时设置其对应的清除位，可以清除该状态位。

4. **FIFO 溢出 (SDFFOVF) 事件**

FIFO 中可用滤波器数据的数量可以通过 FIFO 状态寄存器查询。如果 FIFO 接收到的数据个数大于 FIFO 最大深度 (16)，则产生 SDFFOVF 事件。来自四个滤波器模块中的任何一个的 SDFFOVF 事件都可以触发 CPU 中断。该事件可以配置为错误中断 (sdfm_err_intr)，需设置以下配置:

- 使能 SDFM FIFO；
- 使能主中断 (cfg_mie=1)；
- 使能 SDFM FIFO 溢出中断；

在 FIFO 溢出事件中，所有后续的主 (数据) 滤波器数据都将丢失，不存储在 FIFO 中。

在 FIFO 溢出事件中，FIFO 溢出的状态位也会被设置。在中断源不再活动时设置其对应的清除位，可以清除该状态位。

17.4.7.2 数据就绪中断 (sdfm_dr_intr)源

数据就绪中断 (sdfm_dr_intr) 源的结构如图 17-9 所示。每个 sdfm_dr_intr[x]中断源由相应通道的主 (数据) 滤波器产生的数据应答 (AF) 事件或者 FIFO 数据就绪 (SDFFINT) 事

件触发。

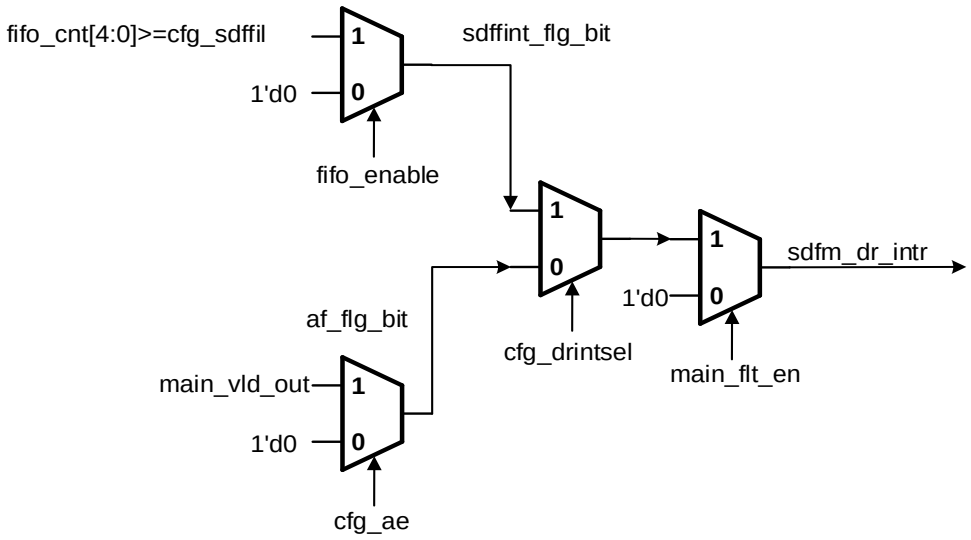


图17-9 SDFM 数据就绪中断 (dfm_dr_intr)

1. **数据应答 (AF) 事件**

   当主滤波器产生新的滤波器数据时，将生成 AF 事件。来自每个主滤波器的 AF 事件可以生成它们自己的数据就绪中断。该事件可以配置为 sdfm_dr_intr[x]中断，需设置以下配置:

   - 使能单独的滤波器中断；
   - 选择数据就绪中断源 AF (cfg_drintsel=0)；

   在 AF 事件中，AF 状态位也会被设置。在中断源不再活动时设置其对应的清除位，可以清除该状态位。

2. **FIFO 数据就绪 (SDFFINT) 事件**

   每当满足 sdffst_rpt >= cfg_sdffil 条件时，就会生成 FIFO 数据就绪事件。每个滤波器的 FIFO 数据就绪事件可以生成它们自己的数据就绪中断。该事件可以配置为 sdfm_dr_intr[x]中断，需设置以下配置:

   - 使能 SDFM FIFO；
   - 使能单独的滤波器中断；
   - 选择 FIFO 数据就绪中断源 AF (cfg_drintsel=1)；
   - 如何选择数据就绪中断 (sdfm_dr_intr[x]) 的输出如表 17-6 所示。

表17-6 SDFM 数据就绪中断输出选择

cfg_drintsel	滤波器中断使能	FIFO 使能	sdfm_dr_intr[x]
0	0	x	0
0	1	x	AF
1	0	x	0
1	x	0	0
1	1	1	SDFFINT

17.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 SDFM 章节。

18 正交编码器脉冲 (QEP)

18.1 简介

Quadrature Encoder Pulse (QEP) 正交编码器脉冲配合外部正交码盘，通过检测电机编码输出信号，获取电机转速、位置、方向等信号，用于高性能的位置及速度控制系统。

18.2 结构框图

QEP 主要包含如下处理单元：

- 输入选择及模式适配处理单元 (PROC)；
- 用于 M 法测转速时，指定时基单元 (UTIME)；
- 位置计数器控制单元 (PCCU)；
- 用于 T 法测转速的捕获单元 (CAP)；
- 检测失速单元 (WDG)；

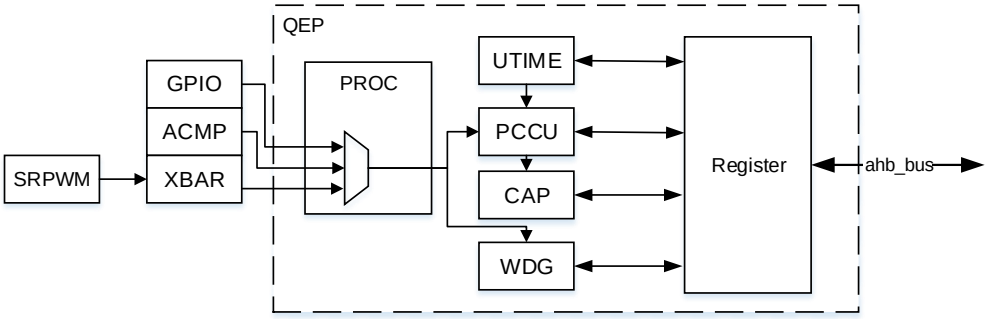


图18-1 QEP 结构框图

18.3 主要特性

QEP 模块主要特性如下：

- 兼容 CW/CCW 格式信号、方向脉冲格式信号输入；
- 支持独立的 T 法测速，单位距离可通过软件编程；
- 支持独立的 M 法测速，单位时间可通过软件编程；
- 支持失速检测；
- 支持 A、B、INDEX 相输入源来自芯片内部 SRPWM、ACMP 模块，完成芯片内部自测；
- 除了编码器 A、B、IINDEX 相信号，还支持 STROBE 相信号做位置指示；该 STROBE 信号同样可来自 SRPWM 模块；

- 支持内部计数器的多种复位模式、初始化模式、锁存模式；
- 内置位置计数比较功能，比较匹配后送出可扩展宽度电平于 INDEX 或 STROBE 引脚；

18.4 功能描述

18.4.1 管脚信号滤波功能

针对 A/B/INDEX/STROBE 管脚输入信号，支持滤波功能，滤波原理为信号的电平宽度大于预设时钟周期值时才可体现变化。4 个管脚信号的滤波配置共用一套。

18.4.2 CW/CCW、方向脉冲模式兼容

- QEP 模块兼容 CW/CCW 模式输入。将 CW/CCW 信号格式转换为方向脉冲模式。并且对 CW/CCW 模式下的默认值也做了 0\1 两种情况兼容。完成 CW/CCW 信号对内部位置计数值的控制；
- 兼容方向脉冲模式输入。兼容 A/B 相任意一路为方向指示信号，另一路作为脉冲信号；

18.4.3 T 法测速

T 法测速原理为预设单位距离，测量该单位距离所需时长即可得到转速。

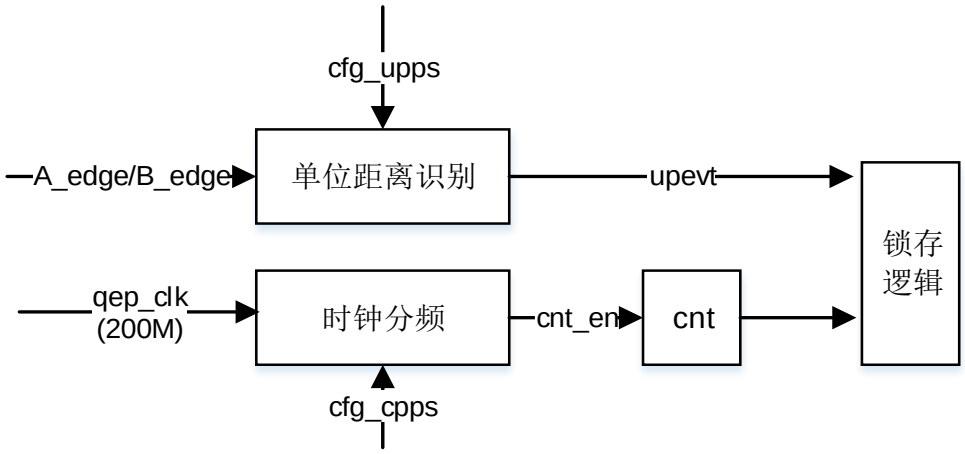


图18-2 T 法示意图

- 可通过配置 cfg_upps 作为单位距离对应的脉冲数的设定；
  - 单位距离对应的脉冲数为 2^{cfg_upps}；
- 可通过 cfg_cpps 完成内部 cnt 的分频计数功能，延长 cnt 可计数范围；
  - cnt_en 为 qep_clk 的 2cfg_cpps 分频。即 2cfg_cpps 个时钟完成一次 cnt 加一动作；

- upevt 事件同时会锁存并清除 cnt 值，开始新一轮计时；

18.4.4 M 法测速

M 法测速原理为预设单位时间，测量该单位时间内所前进距离即可得到转速。

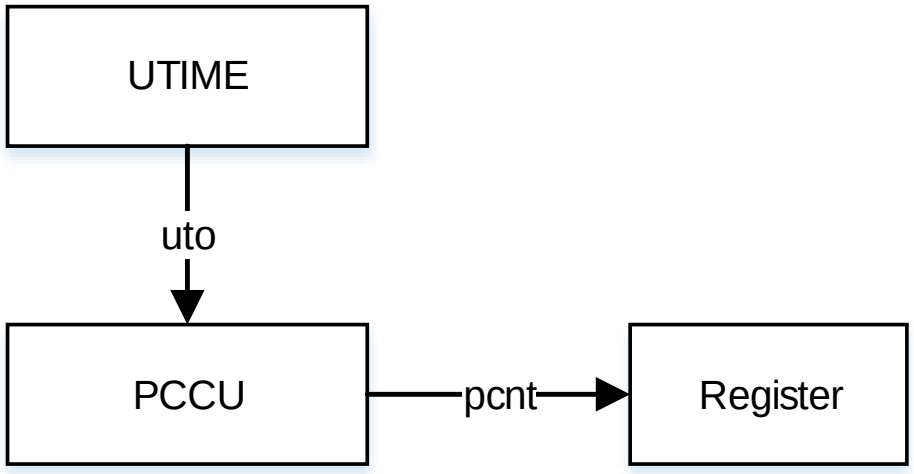


图18-3 M 法示意图

- UTIME 作为时基单元，可通过配置 cfg2ut_uprd 指定单位时间给出 uto 标志；
- PCCU 接收到 uto 信息后将对应的 pcnt 锁存；

uto 中断后，获取两次 uto 锁存值作差即可得到单位时间走过的距离。进一步即可得到对应转速；

18.4.5 失速检测

内置一看门狗检测用以检测失速，该看门狗计时值基于 QEP 模块时钟的 64 分频。当输入的 A、B 相信号超时未进行翻转，使得该狗计时值超过预设值，则中断告警。

18.4.6 计数模式

QEP 内部计数器拥有以下计数模式：

- **正交时钟模式：**输入 A，B 为相差 90°的信号，硬件通过相位差判断计数方向，计数脉冲为 A/B 的上下沿；
- **方向计数模式：**输入信号 A 的上下沿代表了计数脉冲，B 信号代表了计数方向，高为正方向，低为反方向；
- **增计数模式：**固定方向为 1 (正向)，计数脉冲为交换后 A 信号上升沿或双沿；
- **减计数模式：**固定方向为 0 (反向)，计数脉冲为交换后 A 信号上升沿或双沿；

18.4.7 复位模式

内部位置计数值拥有多种复位模式：

- **第一次 INDEX 复位**：此模式下位置计数值只在第一次 INDEX 为高时的 A/B 边沿信号进行复位，此后的 INDEX 不触发复位；
- **INDEX 复位**：此模式下第一次复位发生在 INDEX 为高时的 A/B 边沿处，并记录此时的方向信息，及边沿类别 (A 上升沿、A 下降沿、B 上升沿、B 下降沿)；此后的复位发生在 INDEX 为高，若此时方向与记录方向相同，则在相同的边沿信号进行复位，若不同则在相反的边沿信号复位;
- **超时复位**：位置计数值在超时中断给出时进行复位；
- **最值复位**：注意以上三种模式都存在该复位，只要位置计数值超过预设最值，则复位；

18.4.8 初始化模式

内部位置计数值支持多种初始化模式有：

- INDEX 触发
- STROBE 触发
- 软件触发

1. 位置计数值的 INDEX 初始化可配模式；
   - 在 INDEX 上升沿初始化；
   - 在 INDEX 下降沿初始化；
   - INDEX 不影响初始化；
2. 位置计数值的 STROBE 初始化可配模式：
   - 在 STROBE 上升沿初始化；
   - 正向 STROBE 上升沿初始化或反向 STROBE 下降沿初始化；
   - STROBE 不影响初始化；
3. 软件写 1 到一特定寄存器 (**cfg2pccu_swi**) 的写 1 动作触发初始化，该寄存器不会自动清零；

18.4.9 锁存模式

针对不同事件，可将当前的位置计数值锁存到对目的寄存器；

1. 支持位置计数值 INDEX 来临时锁存到相应寄存器 (PCNTILAT) 模式：
   - INDEX 上升沿锁存；
   - INDEX 下降沿锁存；
   - 标记时刻锁存；
   - 不锁存；

说明

(1) 注意当位置计数值的复位模式为 INDEX 复位，则直接为标记时刻锁存，忽略该配置；

(2) 标记时刻定义第一次 INDEX 为高时的 A/B 的边沿处，并记录此时的方向信息，及边沿类别 (A 上升沿、A 下降沿、B 上升沿、B 下降沿)；此后在 INDEX 为高电平期间，若方向与记录方向相同，则称相同的边沿信号处为标记时刻，若不同则记相反的边沿信号为标记时刻;

2. 支持位置计数值的 STROBE 锁存到相应寄存器 (PCNTSLAT) 模式；

- STROBE 上升沿锁存；
- 正向时，STROBE 的上升沿锁存；或反向时，STROBE 的下降沿；

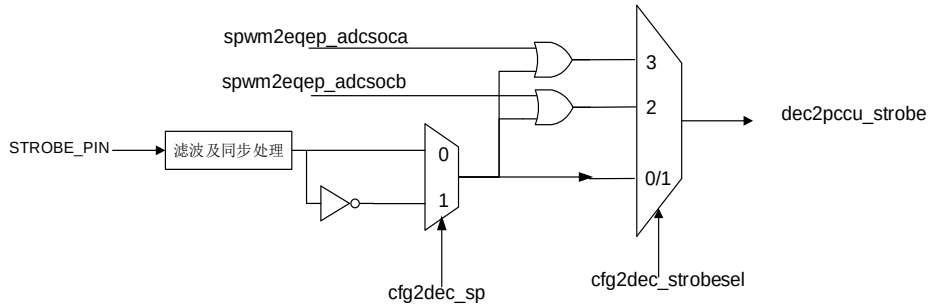
18.4.10 输入选择

输入的 A/B/INDEX 信号来源除了芯片管脚，还可通过内部其它模块给入：

序号	A	B	INDEX
0	Device Pin A for QEPA	Device Pin B for QEPB	Device Pin INDEX for INDEX
1	cmpc_ctripl[0]	cmpc_ctripl[0]	cmpc_ctripl[0]
2	cmpc_ctripl[1]	cmpc_ctripl[1]	cmpc_ctripl[1]
3	cmpc_ctripl[2]	cmpc_ctripl[2]	cmpc_ctripl[2]
4	cmpc_ctripl[3]	cmpc_ctripl[3]	cmpc_ctripl[3]
5	cmpc_ctriph[0]	cmpc_ctriph[0]	cmpc_ctriph[0]
6	cmpc_ctriph[1]	cmpc_ctriph[1]	cmpc_ctriph[1]
7	cmpc_ctriph[2]	cmpc_ctriph[2]	cmpc_ctriph[2]
8	cmpc_ctriph[3]	cmpc_ctriph[3]	cmpc_ctriph[3]
9	xbar2spwm_fault[0]	xbar2spwm_fault[0]	xbar2spwm_fault[0]
10	xbar2spwm_fault[1]	xbar2spwm_fault[1]	xbar2spwm_fault[1]
11	xbar2spwm_fault[2]	xbar2spwm_fault[2]	xbar2spwm_fault[2]
12	xbar2spwm_fault[3]	xbar2spwm_fault[3]	xbar2spwm_fault[3]
13	xbar2spwm_fault[4]	xbar2spwm_fault[4]	xbar2spwm_fault[4]
14	xbar2spwm_fault[5]	xbar2spwm_fault[5]	xbar2spwm_fault[5]
15	xbar2spwm_fault[6]	xbar2spwm_fault[6]	xbar2spwm_fault[6]

针对 INDEX 输入，还提供了 STROBE 门控 INDEX 的功能，仅当 STROBE 为高期间，内部模块才可识别到 INDEX 信号；

STROBE 信号的来源同样可来自 SRPWM 模块，其逻辑如下：



18.4.11 比较输出

QEP 内部支持对位置计数值进行比对，并将比较结果进行输出到 INDEX 管脚或 STROBE 管脚。

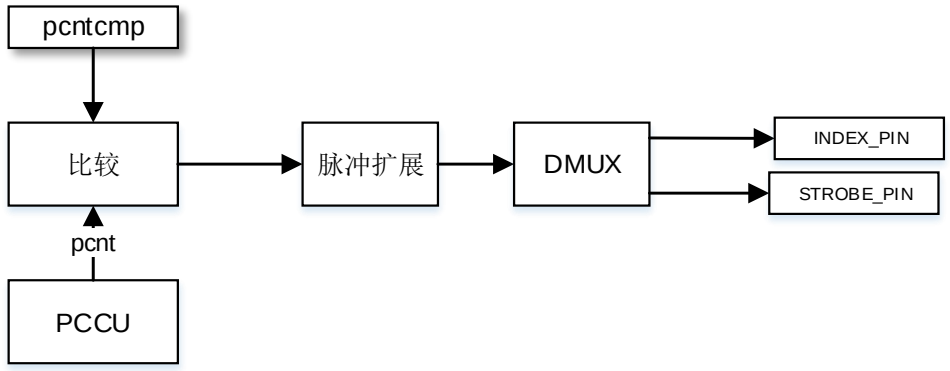


图18-4 QEP 比较结果输出

- 比较值的设定支持影子加载，可设置以下两种事件作为影子模式下更新条件；
  - 位置计数值为 0 时更新比较值；
  - 上一次位置比较匹配成功更新比较值；
- 该比较结果进行扩展后可通过 INDEX 管脚或 STROBE 管脚进行输出，告知对应状态；

18.4.12 事件与中断

QEP 模块除上述比较输出外，无其它事件输出。

QEP 的中断信息如下：

- UTO (Unit timeout)，计时单元超时触发中断；
- IEL (Index event latch)，index 事件锁存成功，触发中断；
- SEL (Strobe event latch)，strobe 事件锁存成功，触发中断；
- PCM (Position compare match)，位置比较单元匹配上触发中断；
- PCR (Position compare ready)，位置比较单元影子模式下完成更新，触发中断；

- PCO (Position counter overflow)，位置计数值 overflow，达到最大值，并还要进行递增，且此时没有其它模式进行复位位置值的复位操作，则触发中断；
- PCU (Position counter underflow)，位置计数值 underflow，达到 0 值，并还要进行减法处理，且没有其它模式的复位操作，则触发中断；
- WTO (Watchdog timeout)，失速检测看门狗超时触发中断；
- QDC (Quadrature Direction change)，计数方向变化，触发中断；
- PHE (Phase Error)，正交计数模式下 A 相、B 相同时出现跳变沿，触发中断；
- PCE (Position counter error)，位置计数错误：index 复位模式下，锁存值不等于预设值中断；
- QMAE，适配器检测错误上报，A，B 相信号同时为有效电平或 A，B 同时跳变；

18.4.13 状态记录

QEP 模块提供了丰富状态信息，以便应用需要；

- upevt 记录：该状态记录代表硬件完成一次单位距离的运动，需软件清除；
- fidf 记录：该状态记录模块复位解除后，第一次 INDEX 为高电平时的方向信息；
- qdf 记录：该状态反应了当前硬件控制位置计数方向；
- qdlf 记录：该状态每次 INDEX 存在时的方向信息；
- coef 记录：单位距离设定过大，使得内部计时值溢出状态，需软件清除；
- cdef 记录：测量单位距离期间，存在方向改变状态，需软件清除；
- fimf 记录：该状态反应硬件认为的第一次 INDEX 为高电平已来过，需软件清除；
- pcef 记录：INDEX 复位模式下，记录位置计数值的正确性，每次 INDEX 来临时更新；

18.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 QEP 章节。

19 高精度脉宽调制器 (SRPWM)

19.1 简介

SRPWM 模块包含 12 通道 PWM 发波子模块，其中有 4 通道 (0~3 通道，共 8 个支路) 支持高精度 PWM 发波，其余 8 通道 (4~11 通道，共 16 个支路) 仅支持标准精度 PWM 发波。

上述 PWM 通道中的每一个通道均可输出一对发波信号 (可配置为互补或独立发波)，共计输出 24 路 PWM 波。可用于电机控制和电源控制 (PFC、LLC、OBC……) 等场合。每个通道独占一个时基模块，12 个通道的 PWM 波周期可独立配置，24 路 PWM 波的相位、占空比及死区均可独立配置。其中 0~3 通道的周期、占空比及死区支持高精度调整，调整精度可达 156 ps (200 MHz 工作时钟)。12 个通道间可任意配置为主从同步通道或是独立运行通道。各通道均支持外部异常信号、内部比较器输出信号，以及内部故障/告警/复位/看门狗等多种信号触发封波保护。

19.2 结构框图

SRPWM 模块框图如下：

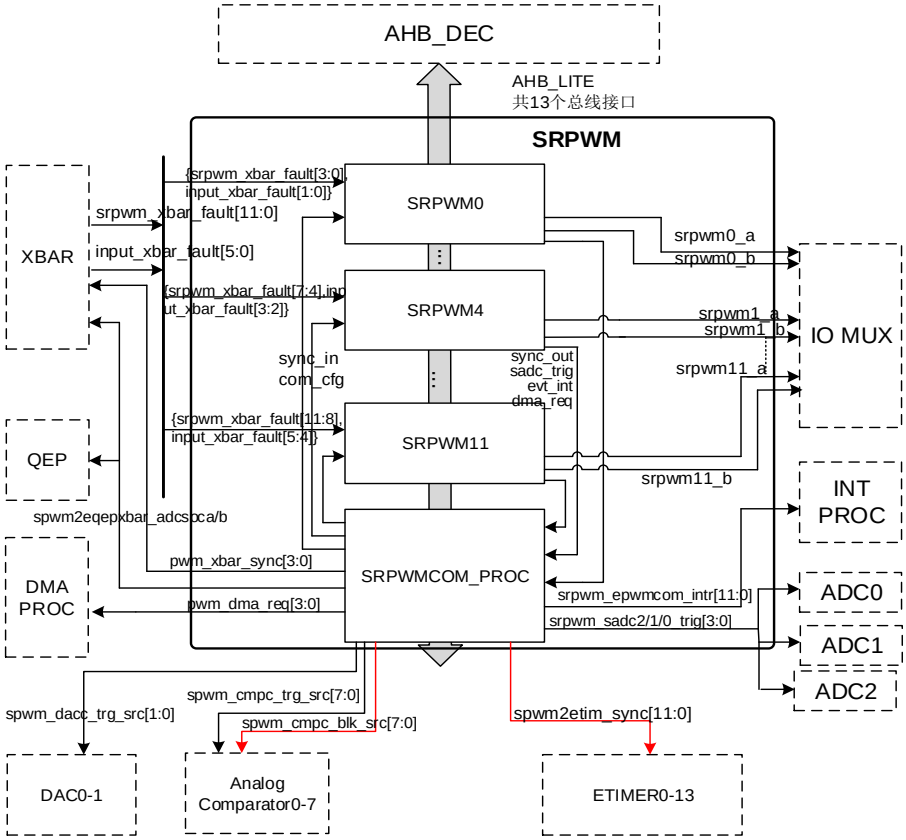


图19-1 SRPWM 模块整体结构图

19.3 主要特性

主要特性如下：

- 可输出 12 通道共 24 路 PWM 波信号
  - 每个通道输出两路 PWM 波；
  - 每个通道内的两路 PWM 波既可以配置为互补关系，也可以配置为相位和占空比完全独立的关系；
- 12 个 PWM 通道均配备了独立的定时单元，用于当前通道的发波信号周期、占空比及相位控制
  - 12 个通道的每一个通道均支持响应 2-bit 同步输入信号，同时自身可以产生 2-bit 同步输出信号；
  - 12 个通道的 12 个定时单元既可以配置为独立运行，互不相关，也可以配置为主从关系 (通过 2-bit 同步输入信号进行同步)；
- SRPWM 各通道间支持级联同步方式或一对多星型同步方式
  - 支持任意两个及两个以上定时单元配置为主从单元，其余定时单元为独立的定时单元；
  - 支持同步信号以级联方式传递或者是以一对多的星型方式传递；

  - 任意两个及两个以上定时单元配置为主从单元时，可选择以下任一种同步方式：
    - ✓ 同步于指定通道的定时单元；
    - ✓ 同步于芯片外部 (通过 XBAR) 输入的同步触发信号；
    - ✓ 同步于芯片内其他模块 (通过 XBAR) 输入的 Fault 信号；
- 0~3 通道 PWM 波信号支持周期、占空比及死区做高精度调整，精度可达到 156 ps
  - 4 个通道的 PWM 信号的周期、占空比及死区可独立进行高精度调整；
  - 高精度调整功能可旁路；
- 支持丰富的比较事件
  - 每个通道均配置 4 路主比较器和 4 路从比较器，产生定时计数值比较事件；
  - 比较事件可用于发波信号占空比调整；
  - 比较事件可作为各种同步触发事件；
- 支持丰富的发波触发事件
  - 每个通道内部支持 22 种事件可配置触发 PWM 发波；
  - 每个通道内部 2 路 PWM 波的发波触发事件可独立配置；
- 支持标准精度和高精度死区调整 (0~3 通道支持高精度死区调整)
  - 标准精度死区调整的精度为 SRPWM 工作时钟周期；
  - 高精度死区调整的精度可达到 156 ps；
- 支持 Burst 发波模式
- 前级保护模块支持多种保护封波形式
  - 支持多种保护封波触发信号：来自 XBAR 的外部输入信号，来自芯片内部比较器的比较结果，CPU 故障、看门狗信号、Flash 错误、温度告警；
  - 支持多种封波模式：CBC 封波模式、无约束封波模式、One-shot 封波模式；
  - 支持多种封波动作：置 0、置 1、置高阻、取反；
- 后级保护模块支持快速封波以及通道内和通道间发波信号的互斥判断，可结合互斥结果和外部故障信号产生相应的保护动作
- 支持输出 4 路 DMA
- 支持输出 12 路中断信号
- 为每一个 ADC Core 提供多达 12 路采样触发信号
- 支持全部通道或若干通道并发同时配置
- 支持全部通道或若干通道的关键参数寄存器 (周期、占空比、死区、发波响应等) 同时立即生效或同步到周期开始生效
- 支持谷值开关功能

19.4 功能描述

19.4.1 SRPWM 关键接口信号

表19-1 SRPWM 与芯片内部其他模块的交互信号

关键接口信号名称	信号类型	说明
clk_srpwm	输入	SRPWM 模块的工作时钟，来自于 PLL0 路径，典型配置值 200 MHz
rst_srpwm_n	输入	SRPWM 模块的复位信号，低有效
xbar_srpwm_fault[17:0]	输入	来自 XBAR 的 17-bit 信号，可作为同步、封波保护触发源
srpwm_dma_req [3:0]	输出	4 个 DMA 通道的 DMA 请求
dma_srpwm_ack [3:0]	输入	4 个 DMA 通道的 ACK 应答
srpwm_srpwmcom_intr [11:0]	输出	12 个 SRPWM 通道送出的中断信号
srpwm_iomux_pwma[11:0]	输出	SRPWM 模块送往 IOMUX 模块的 12 通道发波信号 a 支路
srpwm_iomux_pwmb[11:0]	输出	SRPWM 模块送往 IOMUX 模块的 12 通道发波信号 b 支路
srpwm_iomux_pwma_oen	输出	SRPWM 模块送往 IOMUX 模块的 12 通道发波信号 a 支路 OE 开关，低电平输出发波信号，高电平将相应发波管脚置为高阻态
srpwm_iomux_pwmb_oen	输出	SRPWM 模块送往 IOMUX 模块的 12 通道发波信号 b 支路 OE 开关，低电平输出发波信号，高电平将相应发波管脚置为高阻态
srpwm_xbar_sync[3:0]	输出	SRPWM 模块送往 XBAR 的 4-bit 同步信号，可经由 XBAR 送往芯片外部，用于芯片外部器件的同步信号
srpwm_dacc_trg_src[1:0]	输出	送往 DAC 的触发信号
srpwm_cmpc_trg_src[6:0]	输出	送往 7 个比较器中 DAC 的触发信号
spwm_cmpc_trg_src[7:0]	输出	送给 8 个 ACMP 模块作为其内部 DAC 模块的同步加载信号
spwm_cmpc_blk_src[7:0]	输出	送给 8 个 ACMP 模块作为其消隐触发信号
spwm2eqepxb_adcsoca	输出	送给 QEP 模块的 a 支路触发信号，同时也送到 XBAR 模块
spwm2eqepxb_adcsocb	输出	送给 QEP 模块的 b 支路触发信号，同时也送到 XBAR 模块
srpwm2etm_trig_sig[11:0]	输出	送往 12 个 ETIMER 通道的触发信号，每个比特送到 1 个通道
srpwm_sadc0_trig[3:0]	输出	送往 3 个 ADC 模块的第一组 4-bit 触发信号，对应于 ADC 寄存器 VC_CTL 描述中 srpwm_sadc0_trig[3:0]信号
srpwm_sadc1_trig[3:0]	输出	送往 3 个 ADC 模块的第二组 4-bit 触发信号，

关键接口信号名称	信号类型	说明
		对应于 ADC 寄存器 VC_CTL 描述中 srpwm_sadc1_trig[3:0]信号
srpwm_sadc2_trig[3:0]	输出	送往 3 个 ADC 模块的第三组 4-bit 触发信号，对应于 ADC 寄存器 VC_CTL 描述中 srpwm_sadc2_trig[3:0]信号

19.4.2 SRPWM 模块术语说明

19.4.2.1 公共寄存器

用于同时配置 12 个通道公共参数的寄存器，在 SRPWM 章节中需引用公共寄存器时会特别标明公共寄存器字样。

19.4.2.2 通道寄存器

指每个 SRPWM 通道对应的专用寄存器，每个通道对应的寄存器定义一致，只是地址空间不同。在 SRPWM 章节中引用寄存器时，如未特别说明为公共寄存器，则默认为通道寄存器。

19.4.2.3 寄存器引用说明

在 SRPWM 章节中以寄存器名[寄存器域]的方式做寄存器引用，比如“寄存器 **CFG_CMP_CTL[cfg_cmp_md]**”，是指寄存器中寄存器名为 CFG_CMP_CTL 的 bit0 (cfg_cmp_md 位于该寄存器的 bit0 位置)。

19.4.2.4 寄存器组的引用说明

某些参数是在一组连续地址的寄存器中做配置，比如 **CFG_PG_CTLA** 寄存器组对应的是偏移地址为 0x80，0x84……的一系列寄存器，在当前章节中引用到这类寄存器组时，以 _GRP 后缀代表，如 **CFG_PG_CTLA_GRP**。

19.4.2.5 Mask_OR 选择方式

以 Mask_OR 方式进行选择，表示在一系列事件中选定多个事件，选定的事件按或方式组合产生触发信号，触发相关逻辑进行动作。在进行 Mask_OR 方式选择时，相应的配置寄存器包含多个比特位，比特位个数与目标事件集合的数目一致。每个比特位对应一个事件，当某个比特位置 1 时，则选定对应的事件。一个 Mask_OR 方式选择的例子见图 19-12。

19.4.3 事件说明

事件是指模块外部输入信号产生高电平脉冲或者是模块内部某类条件满足产生高电平脉冲信号，脉冲宽度可包含 1 个或多个时钟周期。在 SRPWM 模块中包含如下事件：

- 软件强制事件 (sft_evt)：由公共寄存器 **CFG_SRPWM_SFT_EVT** 配置产生，该寄存器共 12 位对应 SRPWM 通道 0~11，最高位对应通道 11，软件对相关比特写 1 值，将会在其对应的通道上产生一个脉冲事件；
- 发波 a 支路上升沿事件 (pwma_pos)：SRPWM 通道内 a 支路做初级保护处理之前的 PWM 波脉冲信号上升沿事件；
- 发波 a 支路下降沿事件 (pwma_neg)：SRPWM 通道内 a 支路做初级保护处理之前的 PWM 波脉冲信号下降沿事件；
- 发波 b 支路上升沿事件 (pwmb_pos)：SRPWM 通道内 b 支路做初级保护处理之前的 PWM 波脉冲信号上升沿事件；
- 发波 b 支路下降沿事件 (pwmb_neg)：SRPWM 通道内 b 支路做初级保护处理之前的 PWM 波脉冲信号下降沿事件；
- 突发结束事件 (burst_end)：SRPWM 通道内突发发波处理结束标志，详见突发发波处理章节；
- 同步输入事件 (sync_in[1:0])：输入 SRPWM 通道的同步脉冲事件，共有两比特；
- Fault 上升沿事件 (fpp_fault_pos[3:0])：SRPWM 接收来自 XBAR 的 6-bit 信号 (XBAR 共向 SRPWM 模块送来 18-bit 信号，每个通道分配 6 个比特)，其中低 4 比特信号的上升沿对应产生 4 比特上升沿事件 (详见 19.4.8 "Fault 信号处理 (FAULT_PROC)")；
- Fault 实时事件 (fpp_fault_real[5:0])：输入每个通道的 6 比特 fault 信号，在通道内经过 Fault 信号处理后输出 6 比特的实时信号 (详见 19.4.8 "Fault 信号处理 (FAULT_PROC)")；
- Fault 锁存事件 (fpp_fault_latch[5:0])：输入每个通道的 6 比特 Fault 信号，在通道内经过 Fault 信号处理后输出 6 比特的锁存信号 (详见 19.4.8 "Fault 信号处理 (FAULT_PROC)")；
- 主比较器向上计数事件 (cmp_up_evt[3:0])：4 比特事件，从最低位至最高位依次当定时计数器向上计数值等于 (或大于等于，视寄存器 **CFG_CMP_CTL[cfg_cmp_md]** 的配置而定) 寄存器 **CFG_CMP_CMPA[cfg_cmp_cmpa]** 、 **CFG_CMP_CMPB[cfg_cmp_cmpb]** 、 **CFG_CMP_CMPC[cfg_cmp_cmpc]**、**CFG_CMP_CMPD[cfg_cmp_cmpd]**的高 16 位配置值时所产生的脉冲事件；
- 主比较器向下计数事件 (cmp_dw_evt[3:0])：4 比特事件，从最低位至最高位依次当定时计数器向下计数值等于 (或小于等于，视寄存器 **CFG_CMP_CTL[cfg_cmp_md]** 的配置而定) 寄存器 **CFG_CMP_CMPA[cfg_cmp_cmpa]** 、 **CFG_CMP_CMPB[cfg_cmp_cmpb]** 、 **CFG_CMP_CMPC[cfg_cmp_cmpc]**、**CFG_CMP_CMPD[cfg_cmp_cmpd]**的高 16 位配置值时所产生的脉冲事件；
- 从比较器向上计数事件 (scmp_up_evt[3:0])：4 比特事件，从最低位至最高位依次当定时计数器向上计数值等于寄存器 **CFG_CMP_SCMPA[cfg_cmp_scmpa]**、**CFG_CMP_SCMPB[cfg_cmp_scmpb]** 、 **CFG_CMP_SCMPC[cfg_cmp_scmpc]** 、**CFG_CMP_SCMPD[cfg_cmp_scmpd]**的 16 位配置值时所产生的脉冲事件；
- 从比较器向下计数事件 (scmp_dw_evt[3:0])：4 比特事件，从最低位至最高位依次

当定时计数器向下计数值等于寄存器 **CFG_CMP_SCMPA[cfg_cmp_scmpa]**、**CFG_CMP_SCMPB[cfg_cmp_scmpb]**、**CFG_CMP_SCMPC[cfg_cmp_scmpc]**、**CFG_CMP_SCMPD[cfg_cmp_scmpd]**的 16 位配置值时所产生的脉冲事件；

- 周期事件 (prd_evt)：定时计数器的值大于或等于寄存器 **CFG_TB_PRD[cfg_tb_prd]** 高 16 位配置值的当前活跃值时，产生的脉冲事件；
- 过零事件 (zero_evt)：定时计数器向上计数时，当其值等于 0，产生过零事件；
- 时基同步信号 0 (timer_0sync)：是从时基同步信号集合 timer_evt 中选择输出的同步信号，通过配置寄存器 **CFG_TB_SYNC[cfg_tb_sync0_msel]**，从 timer_evt 事件集合中选择得到的一个同步事件；
- 时基同步信号 1 (timer_1sync)：是从时基同步信号集合 timer_evt 中选择输出的同步信号，通过配置寄存器 **CFG_TB_SYNC[cfg_tb_sync1_msel]**，从 timer_evt 事件集合中选择得到的一个同步事件。

19.4.4 事件集合说明

事件集合是由多个事件组成的集合，用于各种同步信号产生、触发信号产生以及动作控制场合。

19.4.4.1 所有事件集合 (all_evt)

所有事件集合为 30-bit 的事件集合，主要用于产生通道的同步输出信号 syncout0 和 syncout1、XBAR 输入信号在通道内做故障处理时的 Blank 区域产生、封波保护的时候 CBC 和 One-shot 方式的时刻控制，从高到低依次对应下述事件：

{sft_evt, pwma_pos, pwma_neg, pwmb_pos, pwmb_neg, burst_over, sync_in[1:0], fpp_fault_pos[3:0], cmp_dw_evt[3:0], cmp_up_evt[3:0], scmp_dw_evt[3:0], scmp_up_evt[3:0], prd_evt, zero_evt}

19.4.4.2 同步加载事件集合 1 (ld01_evt)

同步加载事件集合 1 为 14-bit 的事件集合，主要用于周期、相位、主比较器值、从比较器值以及 Burst 参数的影子加载，从高到低依次对应下述事件：

{sft_evt, burst_over, sync_in[1:0], fpp_fault_pos[3:0], scmp_dw_evt[1:0], scmp_up_evt[1:0], prd_evt, zero_evt}

19.4.4.3 同步加载事件集合 2 (ld23_evt)

同步加载事件集合 2 为 14-bit 的事件集合，主要用于 PWM 波产生动作参数和死区参数的影子加载，从高到低依次对应事件：

{sft_evt, burst_over, sync_in[1:0], fpp_fault_pos[3:0], scmp_dw_evt[3:2], scmp_up_evt[3:2], prd_evt, zero_evt }

19.4.4.4 时基同步信号集合 (timer_evt)

时基同步信号集合为7-bit事件集合，主要用于时基同步信号timer_0sync和timer_1sync，从高到低依次对应事件：**{sft_evt, sync_in[1:0], fpp_fault_pos[3:0]}**

19.4.4.5 输出事件可选同步集合 (sync_all)

输出事件可选同步集合为 24-bit 事件集合，主要用于 SRPWM 通道产生 DMA 请求信号、中断信号及 ADC trigger 信号，从高到低依次对应事件：**{timer_1sync, timer_0sync, pwmb_pos, pwmb_neg, pwma_pos, pwma_neg, scmp_evt[7:0], cmp_evt[7:0], prd_evt, zero_evt}**

其中：

timer_1sync 和 timer_0sync 是从时基同步信号集合 timer_evt 中选择输出的同步信号；

- timer_0sync 是通过配置寄存器 **CFG_TB_SYNC[cfg_tb_sync0_msel]**，从 timer_evt 事件集合中选择得到的一个同步事件，
- timer_1sync 是通过配置寄存器 **CFG_TB_SYNC[cfg_tb_sync1_msel]**，从 timer_evt 事件集合中选择得到的一个同步事件。

19.4.4.6 Burst 同步事件集合 (burst_evt)

Burst 同步事件集合为 6-bit 事件集合，用于 Burst 模式的开始和结束同步事件选择，从高到低依次对应事件：**{scmp_evt[5:4], scmp_evt[1:0], prd_evt,zero_evt}**

19.4.4.7 动作矩阵事件集合 (pg_evt)

动作矩阵事件集合为 22-bit 的事件集合，可用于产生 PWM 发波动作，从高到低依次对应事件：

{prd_evt, scmp_dw_evt[0], scmp_dw_evt[1], scmp_dw_evt[2], scmp_dw_evt[3], cmp_dw_evt[0], cmp_dw_evt[1], cmp_dw_evt[2], cmp_dw_evt[3], zero_evt, scmp_up_evt[0], scmp_up_evt[1], scmp_up_evt[2], scmp_up_evt[3], cmp_up_evt[0], cmp_up_evt[1], cmp_up_evt[2], cmp_up_evt[3], fpp_fault_pos[3:0]}

19.4.4.8 PG 同步事件集合 (pgsft_evt)

PG 同步事件集合为 10-bit 的事件集合，可用于软件强制产生发波动作的起始和结束同步触发，从高到低依次对应事件：**{pwma_pos, pwma_neg, pwmb_pos, pwmb_neg, scmp_evt[5:4], scmp_evt[1:0], prd_evt, zero_evt}**

19.4.5 时钟

SRPWM 模块的工作时钟和总线时钟均为来自 PLL0 路径上的同一个时钟，其典型配置值为 200 MHz。另外，只有当 SRPWM 模块工作于 200 MHz 时钟时，才可使能高精度调整

功能，否则需旁路高精度调整功能。

19.4.6 通道使能

SRPWM 模块中，各个 PWM 通道的工作使能生效时刻与公共寄存器 **CFG_SRPWM_EN** 和寄存器 **CFG_TB_CTL[cfg_tb_sync_en]**配置有关。

当第 x 个通道中 **CFG_TB_CTL[cfg_tb_sync_en]** 配置为 **'b0**，则公共寄存器 **CFG_SRPWM_EN[cfg_srpwm_en[x]]**置为**'b1** 时，该通道使能生效，通道逻辑正常工作。

当 x 通道的 **CFG_TB_CTL[cfg_tb_sync_en]** 配置为 **'b1**，则 **CFG_SRPWM_EN[cfg_srpwm_en[x]]**置为**'b1** 时后，仍需等待等时基同步信号 0 (timer_0sync) 生效后，通道才正式使能，开始正常工作，timer_0sync 真正生效为 1 之前，除了寄存器配置及影子加载功能正常运行外，其余逻辑功能均不起作用。关于 timer_0sync 的详细描述见 19.4.9.2 "PWM 时基模块的同步"小节。

说明

通道使能应在其余参数配置完成后再置'b1。

19.4.7 SRPWM 参数的加载

19.4.7.1 参数的加载模式

在 SRPWM 模块中，七种类型的关键参数 (定时器周期值、定时器相位值、主比较器比较值、从比较器比较值、PWM 动作矩阵值、死区调整值、Burst 发波参数，具体参数寄存器名称详见后继各章节说明) 支持立即加载、通道内影子加载以及多通道间同步加载功能。上述每种类型参数的加载方式可独立配置，互不关联。除上述七种类型参数外，其余参数只支持立即加载模式：配置到寄存器后立即生效，成为活跃值。

19.4.7.2 关键参数的加载

19.4.7.2.1 关键参数的立即加载模式

在前述七种关键参数中，当其对应的同步加载使能 (**cfg_tb_prd_ldsync_en/cfg_tb_ph_ldsync_en/cfg_cmp_ldsync_en……**) 置为 **b'0**，则相应的参数的加载为立即加载模式，软件配置到寄存器后立即生效，成为活跃值。当相关参数的同步加载使能置 1，则相应参数的加载为通道内影子加载或多通道同步加载模式。

19.4.7.2.2 关键参数的通道内影子加载模式

相关参数启用通道内影子加载模式的条件是该参数对应的同步加载使能 (**cfg_tb_prd_ldsync_en/cfg_tb_ph_ldsync_en/cfg_cmp_ldsync_en……**) 置为'b1，且寄存器 **CFG_PSTP_MULTICFG [cfg_shd_multicfg_en]**对应该参数的比特位置为'b0。寄存器 **CFG_PSTP_MULTICFG [cfg_shd_multicfg_en]**各个比特位对应的参数为：

- cfg_shd_multicfg_en[0]: timer 计数器周期参数；
- cfg_shd_multicfg_en[1]: timer 计数器初相参数；

- cfg_shd_multicfg_en[2]: CMP 参数；
- cfg_shd_multicfg_en[3]: SCMP 参数；
- cfg_shd_multicfg_en[4]: PG 矩阵参数；
- cfg_shd_multicfg_en[5]: 死区参数；
- cfg_shd_multicfg_en[6]: Burst 参数；

启用通道内影子加载的参数需等到其对应的同步加载事件（由相应的寄存器配置**cfg_tb_prd_ldsync_msel/cfg_tb_ph_ldsync_msel/cfg_cmp_ldsync_msel……**从同步加载事件集合 1/2 中以 Mask_OR 方式选择）生效后，参数才能够成为活跃值生效。

19.4.7.2.3 关键参数的通道间同步加载模式

通道间同步加载模式支持指定多个通道，这些通道在接收到配置指令后，才开始等待并响应其选择的同步加载触发事件。相关参数启用通道间同步加载模式的条件是该参数对应的同步加载使能（**cfg_tb_prd_ldsync_en/cfg_tb_ph_ldsync_en/cfg_cmp_ldsync_en……**）置为**'b1**，且寄存器 **CFG_PSTP_MULTICFG[cfg_shd_multicfg_en]**对应该参数的比特位也置为**'b1**。

在通道间同步加载模式中，同样需要选择同步加载触发事件(**cfg_tb_prd_ldsync_msel/cfg_tb_ph_ldsync_msel/cfg_cmp_ldsync_msel……**)。与通道内影子加载模式不同的是，要使软件配置的关键参数成为活跃值，首先需要向支持通道间同步加载的所有通道同时写入同步配置指令。通过向公共寄存器 **CFG_SHD_MULTICFG_UPT** (共 12 比特，对应 12 个通道，最低比特对应通道 0) 中多个比特位同时写一次**'b1** 即可实现配置指令的统一下发。该配置指令下发后，相关通道的相关参数（**cfg_shd_multicfg_en** 中置为**'b1** 的比特位所对应的参数）才会等待并响应一次同步加载事件，将软件配置参数变为活跃值。需要说明的是，向公共寄存器 **CFG_SHD_MULTICFG_UPTSRPWM** 写一次**'b1**，就只支持一次同步加载事件触发，完成该次触发后，其后的同步加载触发事件均被忽略，直至向公共寄存器 **CFG_SHD_MULTICFG_UPTSRPWM** 的相关位再写一次**'b1**，才会再响应一次同步加载事件。

在通道间同步加载模式中，通过配置寄存器 **CFG_PSTP_MULTICFG[cfg_shd_multicfg_align_en]**，可以控制同步加载事件的响应时机。如将该寄存器的相关位配置**'b1**，同步配置指令下发后，仍需等到 PWM 周期开始（即 zero_evt 生效）后再准备响应同步加载事件，如果该寄存器相关位配置**'b0**，则同步配置指令下发后，即准备响应同步加载事件。

⚠ 注意

在上述七种关键参数中，若某些关键参数启用了通道间同步加载方式，对于其余未参与通道间同步加载的参数，如果其配置值在完成初始化以后，不会在 SRPWM 工作过程再做更新，则注意将这些参数的影子加载使能置 **0**（影子加载使能是指 **cfg_tb_prd_ldsync_en/cfg_tb_ph_ldsync_en/cfg_cmp_ldsync_en.....**);

当有一个或若干个关键参数启用了通道间同步加载方式，在公共寄存器下发同步配置指令（向 **CFG_SHD_MULTICFG_UPTSRPWM** 写'b1）之前，每一种使能了影子加载的参数（无论其是否启用了通道间同步加载）均需做一次软件配置寄存器动作，否则响应一次同步配置指令后，使能了通道间同步加载的参数在后继工作过程中会按通道内影子加载方式完成加载，无视同步配置指令下发，直至所有使能了影子加载的参数类型均发生过至少一次软件配置动作后，才会再出现一次同步配置指令的响应动作。举例如下：假设仅计数器周期参数和计数初相参数使能了通道

间同步加载（**cfg_shd_multicfg_en=0x03**），其相应的影子加载也使能（**CFG_TB_PRD_LDSYNC[cfg_tb_prd_ldsync_en]='b1**，**CFG_TB_PH_LDSYNC[cfg_tb_ph_ldsync_en]='b1**）；CMP 参数虽未使能通道间同步加载，但使能了影子加载（**CFG_CPM_LDSYNC[cfg_cmp_ldsync_en]='b1**）。某个 PWM 周期内希望进行通道间同步加载计数器周期，但初始相位不做更新，所有 CMP 参数也不做更新。在向 **CFG_SHD_MULTICFG_UPTSRPWM** 写'b1 下发同步配置指令前，应向寄存器 **CFG_TB_PRD**、**CFG_TB_0PHASE** 和任一个 CMP 配置值寄存器（比如 **CFG_CMP_CMPD**）写入配置值，否则在响应当次同步配置指令下发后，后继的同步配置指令均会被忽略，计数周期和初始相位均按通道内影子加载方式配置，直至 **CFG_TB_PRD**、**CFG_TB_0PHASE** 和任一个 CMP 配置值寄存器均发生过软件配置动作，才会再响应一次通道间同步配置指令。

19.4.7.3 多通道参数的并发配置

SYSC 模块寄存器 **CFG_SPWM_BC_MAP[cfg_sysc2spwm_bc_map_info]**的 12 位比特对应 12 个 SRPWM 通道 (LSB 对应 SRPWM0)，将其中的若干比特位置**'b1**，则这些置 1 比特位对应的通道将不再响应对其自身地址空间的访问，而统一响应对 0x4000_F000~0x4000_FFFF 地址空间的访问。在该地址空间中寄存器的偏移地址定义与通道寄存器的偏移地址一致。通过对该地址空间的地址写参数，可将参数同时下发到上述 SRPWM 通道中。

19.4.8 Fault 信号处理 (FAULT_PROC)

19.4.8.1 外部输入 Fault 信号的分配

外部 (指 SRPWM 模块外部) 输入 SRPWM 模块的 xbar_fault 信号源自两方面的信号：

- 芯片外部通过 GPIO 接口送入的故障信号/同步信号/触发信号；
- 芯片内部其余模块送来的故障、告警及复位等指示信号 (比较器 ACMP 模块的比较结果、ADC 看门狗信号、Flash 出错告警、CPU 告警、温度告警等等，详见下图)

其中通过 GPIO 输入的故障信号，经过 XBAR 的 INPUT_XBAR 子模块处理后产生 INXBAR_OUT 信号的低 6 比特（INXBAR_OUT [5:0]）送到 SRPWM 模块 (参见图 25-2 和图 25-3)。另外芯片内部故障信号和 CMPC 比较信号以及 INXBAR_OUT[15:0]在 XBAR 模块的 PWMXBAR 子模块块中处理后产生 12 比特的 PFXBAR_OUT[11:0]信号也送到 SRPWM 模块 (参见图 25-6 和图 25-3)。其中 INXBAR_OUT [1:0]和 PFXBAR_OUT[3:0]组成 6 比特 fault 信号 (INXBAR_OUT [1:0]置于 6 比特 fault 信号的最低两位) 送到 SRPWM 的 0~3 通道。类似的，INXBAR_OUT [3:2]和 PFXBAR_OUT[7:4]组成 6 比特 fault 信号 (INXBAR_OUT [3:2]置于 6 比特 fault 信号的最低两位) 送到 SRPWM 的 4~7 通道；INXBAR_OUT [5:4]和 PFXBAR_OUT[11:8]组成 6 比特 fault 信号送到 SRPWM 的 8~11 通道。如下图所示：

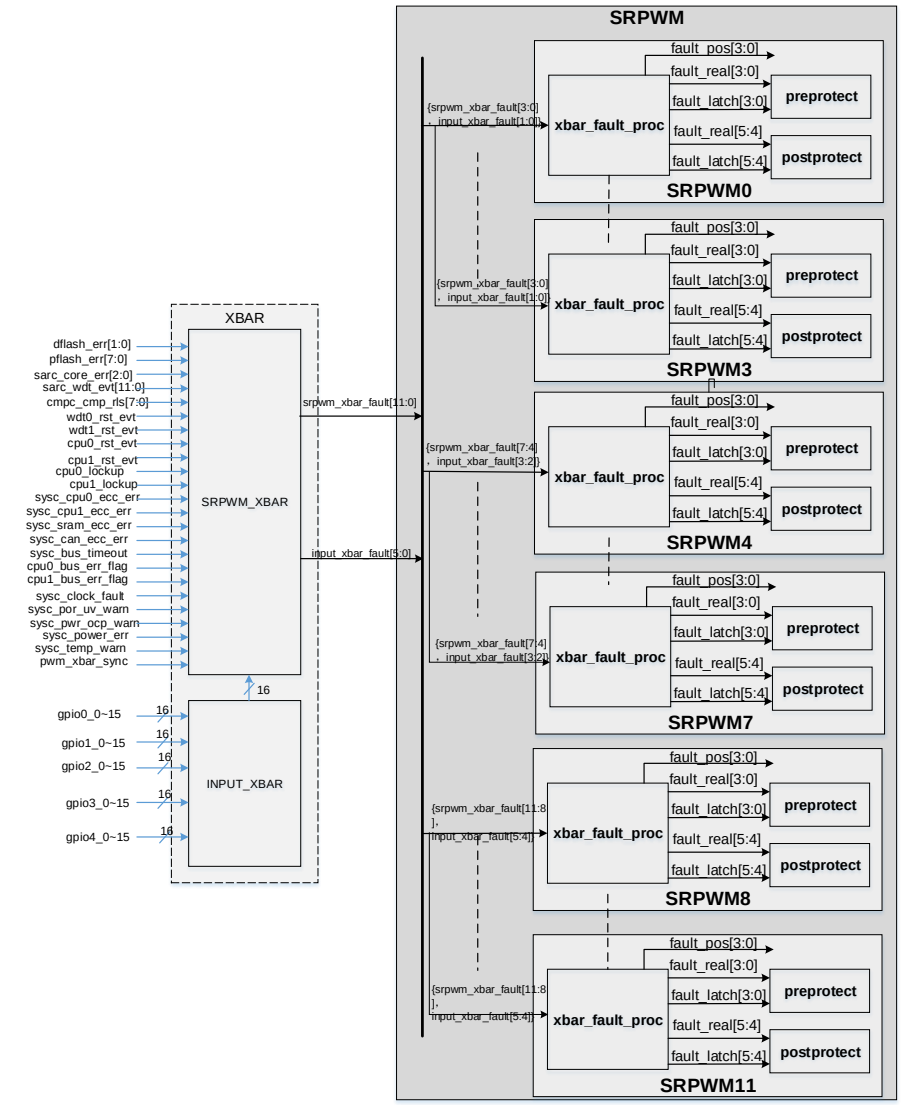


图19-2 Fault 信号在 SRPWM 模块内的分配及处理

说明

当外部信号需要从 GPIO 经 XBAR 送入 SRPWM 模块，需特别注意 GPIO 与其他功能的复用关系，保证欲使用的 GPIO 管脚未被其他功能占用；

GPIO 输入以 4 个为一组 (GPIO0~3, 4~7, 8~11, 12~15)，在 XBAR 模块中首先进行四选一操作，因此如果希望两路或更多路 GPIO 信号都能送到 SRPWM 模块，则需注意将这些信号分配到不同的 GPIO 分组中。

19.4.8.2 Fault 信号处理的功能

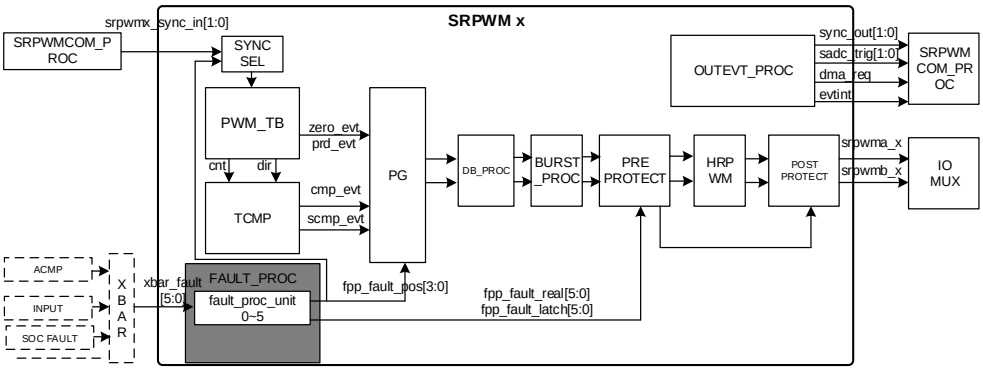


图19-3 FAULT_PROC 模块在 SRPWM 通道内的位置

在单个 SRPWM 通道内，来自 XBAR 的 6-bit 输入信号需进行预处理，包括上、下沿选择，Blanking 操作、锁存处理等，处理后产生 Fault 上升沿事件 (fpp_fault_pos)、Fault 实时事件 (fpp_fault_real) 以及 Fault 锁存事件 (fpp_fault_latch)。其中 fpp_fault_pos 事件可用于定时器同步、参数影子加载以及触发 PWM 发波动作，fpp_fault_real 及 fpp_fault_latch 可用于封波保护以及中断产生。

19.4.8.3 Fault 信号处理流程

在通道内，6-bit 的 Fault 信号并行在 6 个 Fault 处理单元中独立进行处理，互不相关，如下图所示：

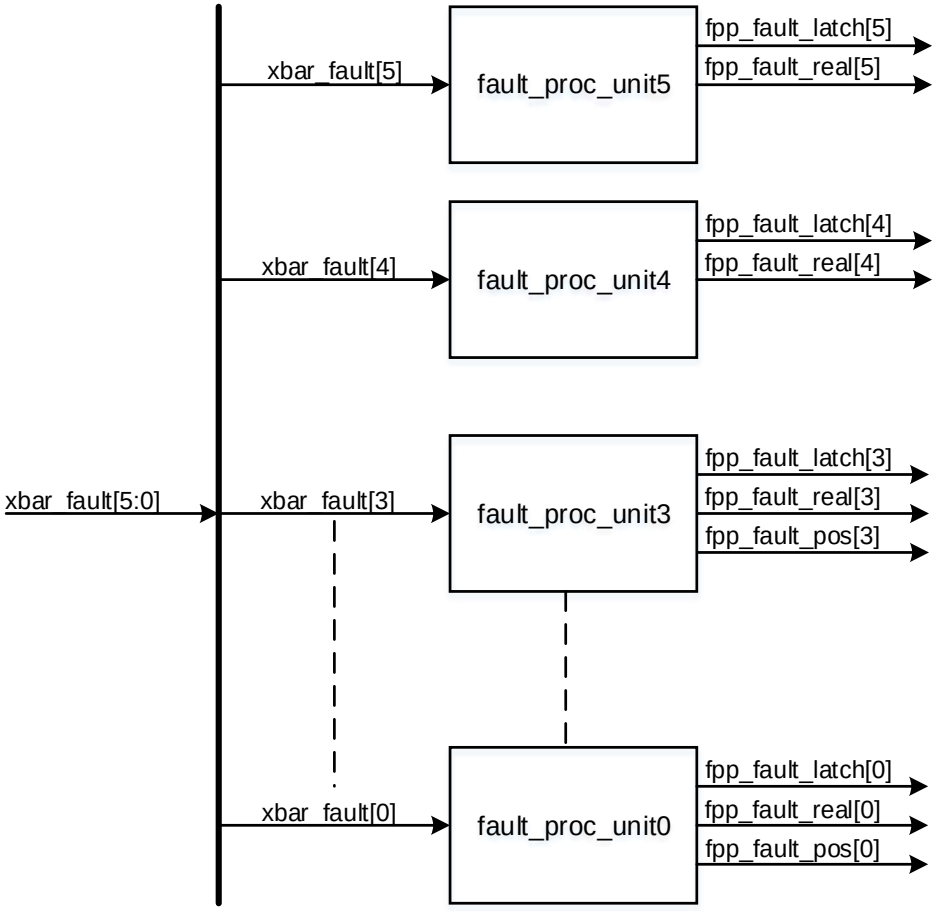


图19-4 通道内 XBAR 输入信号并行处理

如上图所示，6 比特 xbar_fault 信号在 6 个 fault_proc_unit 单元中分别产生 6 比特的实时 Fault 信号 (fpp_fault_real[5:0]) 和 6 比特的锁存 fault 信号 (fpp_fault_latch[5:0])。6 比特 xbar_fault 信号的低 4 比特在 4 个 fault_proc_unit 单元中产生 4 比特的 fpp_fault_pos[3:0]。

fpp_fault_real[5:0]和 fpp_fault_latch[5:0]信号作为实时 Fault 事件和锁存 Fault 事件可用做前级保护触发源、次级保护触发源以及中断触发源。fpp_fault_pos[3:0]可作为时基同步信号触发源、all_evt 事件集的触发源、同步加载事件触发源、产生 PWM 发波信号的动作矩阵事件触发源。

上述 3 种 Fault 信号在单个 fault_proc_unit 单元中的产生流程如下图所示：

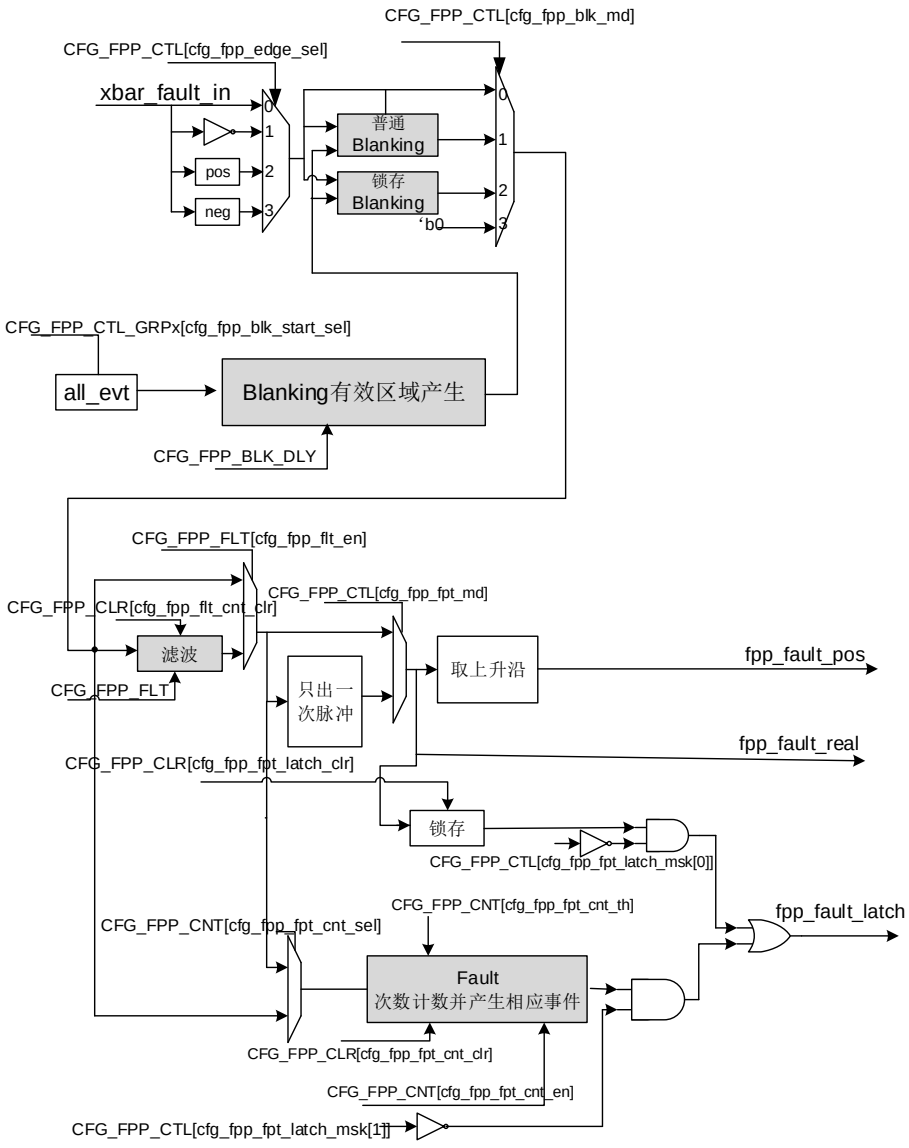


图19-5 单个 fault_proc_unit 单元内信号的处理流程

上图中，Blanking 有效区域产生是指产生一个数据处理有效区域，只处理该区域内出现的 Fault 信号，在有效区域时段之外的 Fault 信号将被抛弃，不做处理。Blanking 区域的开始时刻由寄存器配置 **CFG_FPP_CTL_GRPx[cfg_fpp_blk_start_sel]**在所有事件集合 (all_evt) 中选择 (GRPx 的 **x** 对应 6 比特 Fault 信号的 6 个配置地址，x=0 对应 Fault 信号最低比特的配置地址)。Blanking 区域产生结束时刻由寄存器配置 **CFG_FPP_CTL_GRPx[cfg_fpp_blk_end_sel]**在所有事件集合 (all_evt) 中选择。

说明

如果要选择以 Blanking 方式处理数据 (**CFG_FPP_CTL[cfg_fpp_blk_md]=1** 或 **2**)，则 **CFG_FPP_CTL_GRPx[cfg_fpp_blk_dly_en]**必须配置为 **1**。

上图中普通 Blanking 是指只选择处理有效区域以内出现的 Fault 信号，有效区域外的信号忽略，如下图：

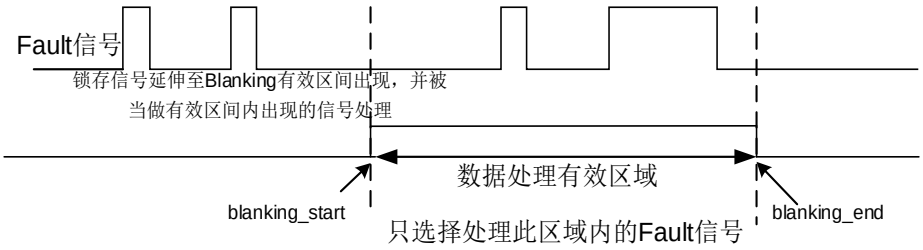


图19-6 Blanking 有效区域选择示意

锁存 Blanking 是指在数据有效区域产生之前如果出现过一次或多次 Fault 脉冲信号，则将会锁存第一次 Fault 信号，直至 Blanking 有效区域出现之后，将此锁存信号当做在 Blanking 区域中出现的 Fault 信号来处理，并随后解除此 Fault 信号的锁存，如下图所示：

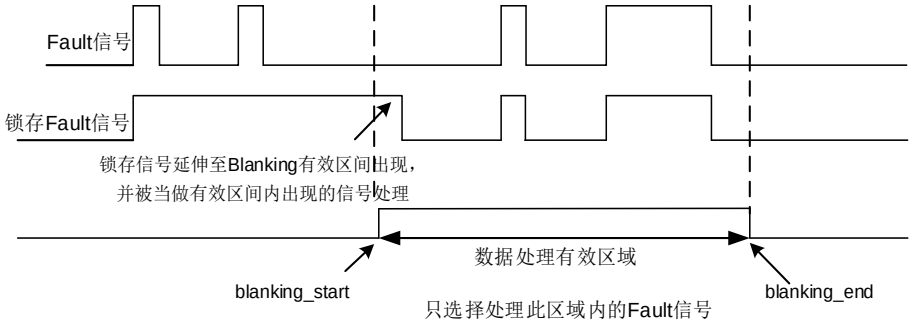


图19-7 锁存 Blanking 处理示意

如果 **CFG_FPP_FLT[cfg_fpp_flt_en]**配置值为 **0**，则滤波逻辑被旁路，滤波输入直通滤波输出。如果 **CFG_FPP_FLT[cfg_fpp_flt_en]** 配置为 **1**，则图 19-5 中的滤波逻辑将对其输入信号进行滤波处理，将第一个脉冲宽度小于滤波长度 **CFG_FPP_FLT[cfg_fpp_flt_len]**配置值的 Fault 脉冲信号滤除掉。需要注意的是此滤波处理逻辑只滤除第一个脉冲宽度小于配置值的 Fault 脉冲信号，随后出现并与该 Fault 信号间隔超过 **CFG_FPP_FLT[cfg_fpp_flt_len]** 配置值的 fault 脉冲信号均不受滤波处理逻辑影响，直通而过。如果此后向寄存器 **CFG_FPP_CLR[cfg_fpp_flt_cnt_clr]**写 **1**，则会产生清零脉冲，将滤波逻辑恢复到初始态，滤波逻辑可再次滤除其后第一个脉冲宽度小于滤波长度的脉冲信号。

图 19-5 中，只出一次脉冲的功能是指当 **CFG_FPP_CTL[cfg_fpp_fpt_md]**配置为**'b1**，且启用 Blanking 模式后，在 Blanking 有效区域内，只选择第一个 Fault 脉冲事件做后继处理，其余脉冲事件抛弃。如果 **CFG_FPP_CTL[cfg_fpp_fpt_md]**配置为**'b0** 或者未启用 Blanking 模式则不会有只出一次脉冲的限制。

图 19-5 中，Fault 次数计数是指每来一个 Fault 上升沿，Fault 计数逻辑计数一次，计到 65535 后保持该值不变，直至向寄存器 **CFG_FPP_CLR[cfg_fpp_fpt_cnt_clr]**写入 **1**，产生清零脉冲，对该计数器清零。

19.4.8.4 谷值开关功能

谷值开关技术可以在开关管电压达到谷底附近打开下个周期的 PWM 波，实现开关损耗的最小化。对于不同的应用场景的和条件，可能会选择不同的位置的谷点开启 PWM 波。谷值开关模块与比较器 (ACMP) 协同合作，检测电压谷点，控制 PWM 波的开启时机。

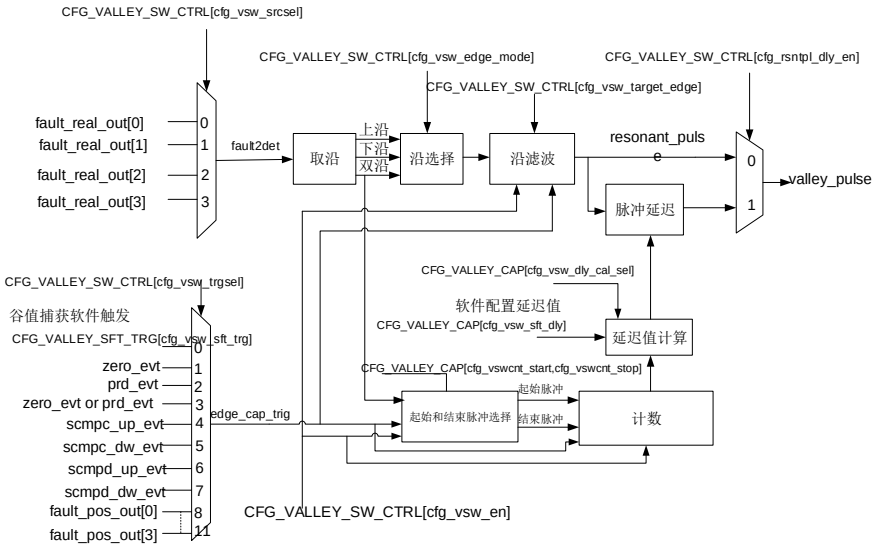


图19-8 谷值开关示意

谷值开关功能的工作参数配置说明如下：

1. CFG_VALLEY_SW_CTRL[cfg_vsw_en]为谷值开关使能配置，置'b1，使能谷值开关功能；
2. 配置 CFG_VALLEY_SW_CTRL [cfg_vsw_srcsel]，从 4 比特的故障信号 (来自 0~3 fault_proc_unit 单元的 fpp_fault_real 信号，如图 19-5 所示) 中选择一个信号作为检测对象；
3. 配置 CFG_VALLEY_SW_CTRL[cfg_vsw_edge_mode]，选择跳变沿检测对象的上沿、下沿或上/下沿作为统计对象；
4. 配置 CFG_VALLEY_SW_CTRL [[cfg_vsw_trgsel]，从过零事件、周期事件、过零和周期事件或、向上计数/向下计数的 SCMPC 事件、向上计数/向下计数的 SCMPD 事件、4 比特故障信号以及谷值捕获软件触发事件中选择一个事件作为谷值跳变次数统计及跳变间隔计数的触发事件，其中谷值捕获软件触发事件来自于对 CFG_VALLEY_SFT_TRG[cfg_vsw_sft_trg]寄存器写 1 操作而产生的脉冲；
5. 配置 CFG_VALLEY_SW_CTRL[cfg_vsw_target_edge]，选择在检测到第几个跳变沿的时候产生一个指示脉冲 (上图中 resonant_pulse)；
6. 配置 CFG_VALLEY_CAP[cfg_vswcnt_start] 和 CFG_VALLEY_CAP[cfg_vswcnt_stop]，分别选择跳变间隔计数的起始和结束触发脉冲位置 (即第几个触发脉冲开始，第几个触发脉冲结束)；
7. 配置 CFG_VALLEY_CAP[cfg_vsw_sft_dly] 值，并配置

CFG_VALLEY_CAP[cfg_vsw_dly_cal_sel]，选择计算指示脉冲的延迟值方式：直接用软件配置值、软件配置值加上跳变间隔计数值、软件配置值加上跳变间隔计数值除以 2/除以 4/除以 16；

19.4.8.5 故障事件锁存 TB 计数值及计数方向

SRPWM 模块的任一通道均支持从 6 个 fault_proc_unit 输出的 fpp_fault_real 事件和 fpp_fault_latch 事件中选择一个或多个 (做或运算)，作为锁存 TB 计数值的锁存信号，当所选事件生效时，锁存 TB 计数值和计数方向并上报。详细配置选择参见寄存器 **CFG_CAP_TBCNT_CTRL** 、 **CFG_CAP_PRT_CLR** 、 **CAP_TBCNT_RPT** 、 **CAP_TBCNT_FLG_RPT** 的说明。

19.4.9 PWM 时基模块 (PWM_TB)

19.4.9.1 PWM 时基模块的功能

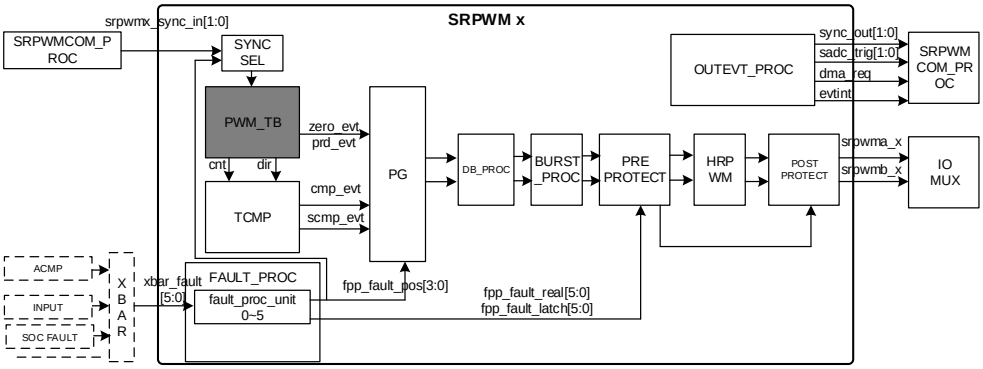


图19-9 PWM 时基模块在 SRPWM 通道内的位置

每个 SRPWM 通道均配备了一个 PWM 时基模块，每个时基模块中包含一个 PWM 定时计数器，该定时计数器可对 PWM 发波周期进行标准精度计数 (计数精度与 SRPWM 模块工作时钟周期一致) 以及增强型高精度计数 (计数精度可达 156.25 ps)。

标准精度计数为 22-bit 计数，用于过零事件及周期事件的产生，控制 PWM 发波信号的周期，并为后继定时比较器逻辑提供比较输入，进而控制 PWM 发波信号的占空比。同时标准精度计数值还为通道内各种触发事件提供时间基准。标准精度计数支持仅向上计数以及向上向下计数两种模式。

下面两幅图中给出了向上计数模式和向上向下计数模式中定时计数器的计数行为。图中两种计数模式的 **CFG_TB_PRD[cfg_tb_prd[26:5]]** 均配置为 **6**。从图中可见，当 **CFG_TB_PRD[cfg_tb_prd[26:5]]=N (N>1)**，对于向上计数模式，PWM 波周期即为 N 个时钟周期，对于向上向下计数模式，PWM 波周期为 2*N 个时钟周期。

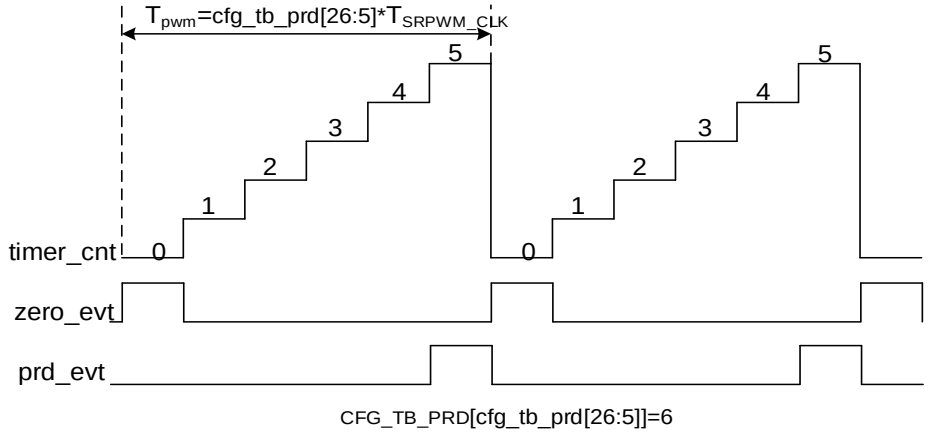


图19-10 向上计数模式

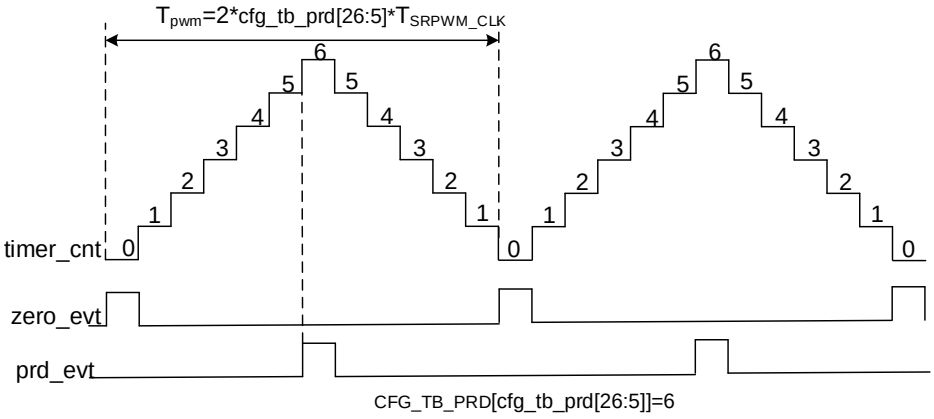


图19-11 向上向下计数模式

CFG_TB_PRD[cfg_tb_prd_hr_msb]对应高精度周期的最高位，在双向计数模式下，需调整的高精度周期值大于等于一个 SRPWM 工作时钟周期时，该寄存器位配置为**'b1**，否则置'b0。

基于标准精度定时计数器可产生过零事件 (zero_evt，对应通道使能生效后计数方向向上且计数值为 0 的时刻) 和周期事件 (prd_evt，计数值大于等于寄存器**CFG_TB_PRD[cfg_tb_prd[20:5]]**的活跃值)。

增强型高精度计数在 SRPWM 模块工作时钟为 200 MHz 且公共寄存器**CFG_SRPWM_HR_EN** 中该通道对应的比特位置为**'b1** (启用高精度模式) 时工作。

说明

当 SRPWM 模块工作时钟不等于 200 MHz 时，需将 **CFG_SRPWM_HR_EN** 中该通道对应的比特位置为**'b0**。

19.4.9.2 PWM 时基模块的同步

在指定事件 (通过软件配置) 的触发下，将定时计数器的计数值置为软件配置的相位值，可实现通道间定时器的同步。PWM 定时器标准精度计数器的初相值可在时基同步信号 0 (timer_0sync) 和时基同步信号 1 (timer_1sync) 触发下完成实时配置。

timer_0sync 由寄存器 **CFG_TB_SYNC[cfg_tb_sync0_msel]**从时基同步事件集合中选择一个或多个触发事件进行 Mask_OR 运算后产生，如下图所示：

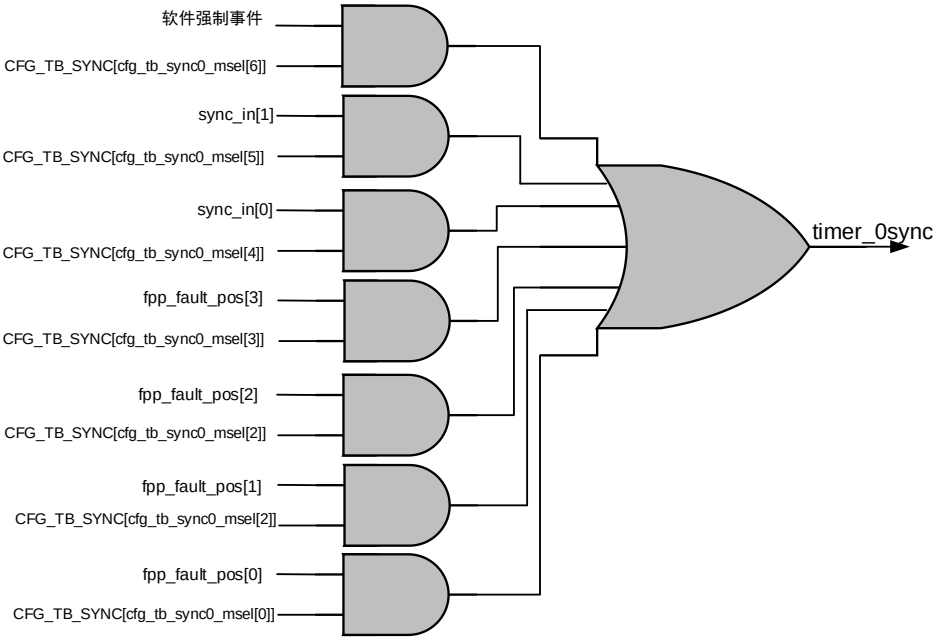


图19-12 按 Mask_OR 方式进行选择的时基同步信号 0

timer_1sync 由寄存器 **CFG_TB_SYNC[cfg_tb_sync1_msel]**从时基同步事件集合中选择一个或多个触发事件进行 Mask_OR 运算后产生。

timer_0sync 的优先级高于 timer_1sync，即当两者同时生效时，只响应 timer_0sync。

通道使能生效 (详见 19.4.6 "通道使能"小节) 后，如果 timer_0sync 和 timer_1sync 触发均未生效，则标准精度计数器默认从 0 值开始计数。当 timer_0sync 生效，标准精度计数器的计数值置为寄存器 **CFG_TB_0PHASE[cfg_tb_0phase]**对应的活跃值并从该值开始计数，**CFG_TB_0PHASE[cfg_tb_0phase]**的活跃值见 19.4.9.3 小节的说明。当 timer_1sync 生效，标准精度计数器的计数值置为寄存器 **CFG_TB_1PHASE[cfg_tb_1phase]**的配置值，即直接使用软件配置的 **CFG_TB_1PHASE[cfg_tb_1phase]**值。

timer_0sync 和 timer_1sync 还可触发标准精度计数器在 UPDOWN 模式下计数方向的同步加载，加载值由寄存器 **CFG_TB_CTL[cfg_tb_0dir]** (对应 timer_0sync 生效) 和 **CFG_TB_CTL[cfg_tb_1dir]** (对应 timer_1sync 生效)决定。

图 19-12 中的 syncin[1:0]信号来自于其它 SRPWM 通道或者是当前通道的同步信号，是实现通道间同步关键触发信号。首先，12 个通道通过其寄存器

CFG_SRPWM_SYNCOUT[cfg_srpwm_syncout0/1_sel]的配置，可以从所有事件集合(all_evt)中选择一个事件脉冲作为同步信号输出，每个通道可以输出 2 个比特的 sync_out[1:0]同步信号，12 个 SRPWM 通道输出 syncout[1:0]信号汇集后进行交叉互联得到 12 对 sync_in[1:0]同步信号送回 12 个通道，如下图：

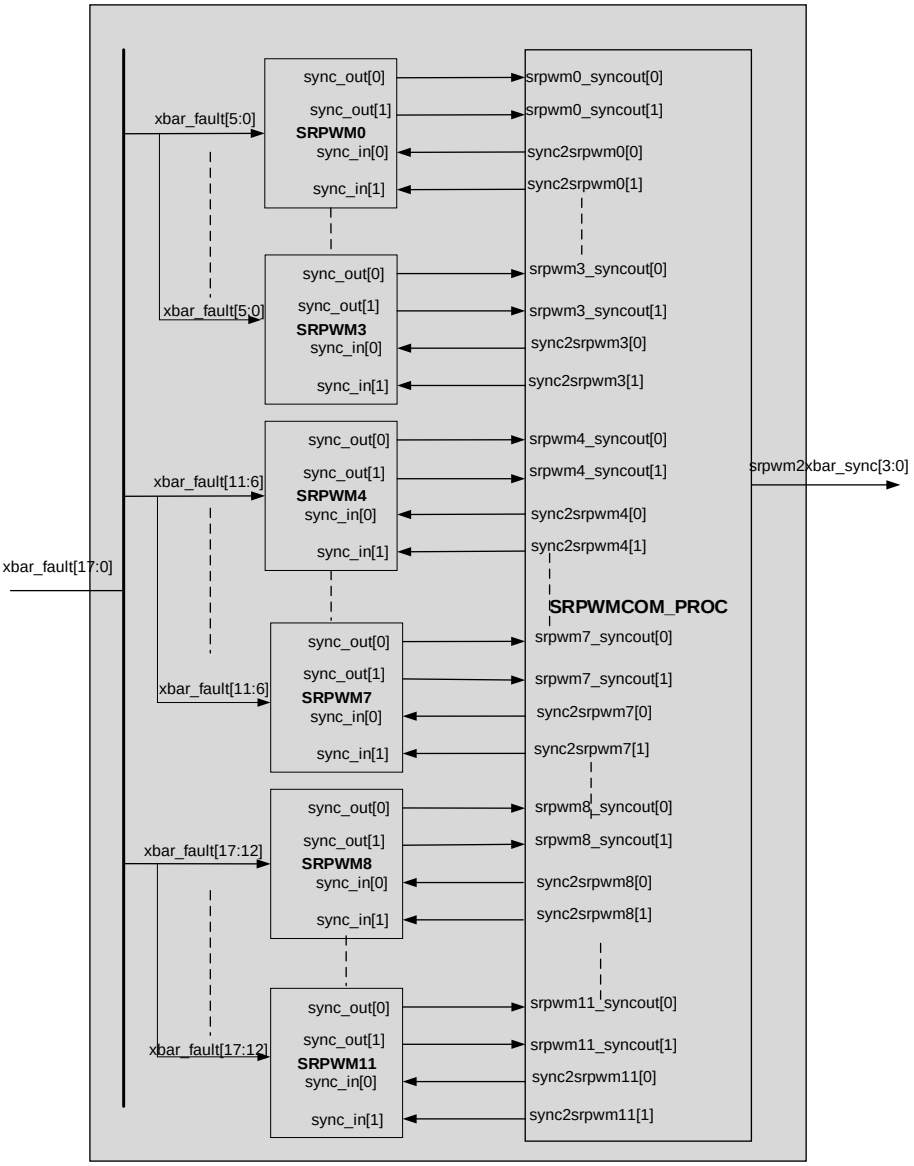


图19-13 各通道同步信号与 SRPWMCOM 模块的交互

12 个通道输出的 sync_out[1:0]信号在 SRPWMCOM_PROC 模块中进行交叉互联处理，如下图所示。通过配置公共寄存器组 **CFG_MSRPWM_SYNC_SEL**，可灵活实现所有通道同步于同一个通道，或者是将 12 个通道分成若干组，每个组内的所有通道指定同步于某个通道：

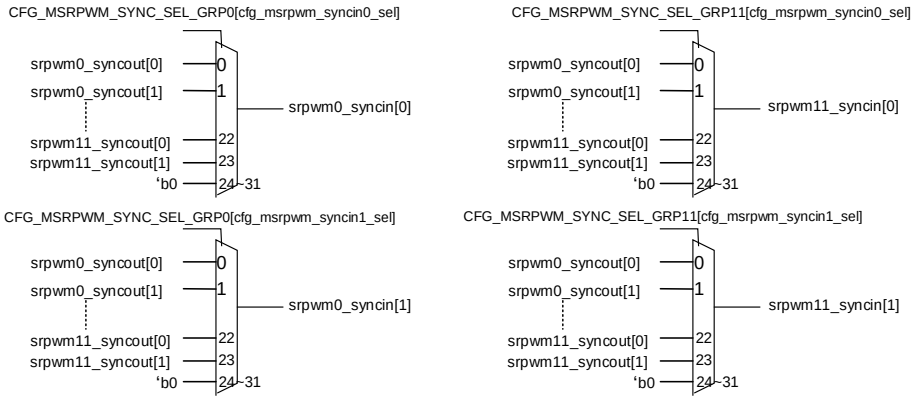


图19-14 通道间同步信号的交叉互联

另外，通过公共寄存器组 **CFG_MSRPWM_XBAR_SYNC_SEL** 的配置，可从 12 个通道的 syncout[1:0]中选择出 4 个比特送往 XBAR，通过 XBAR 的配置，送到芯片外部：

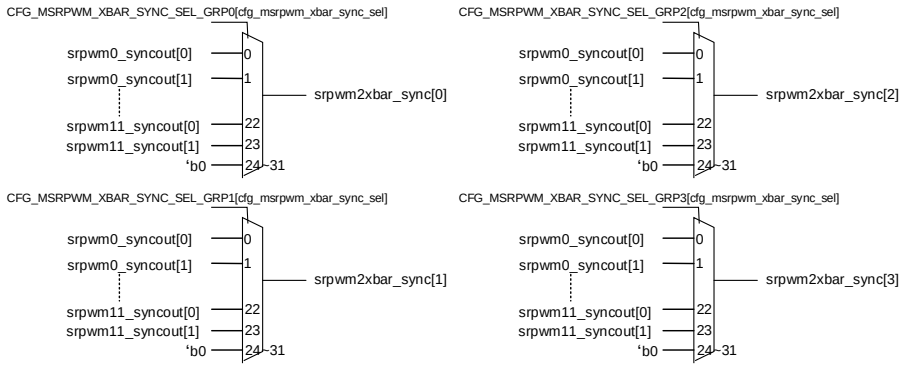


图19-15 送往 XBAR 的同步比特信号选择

软件配置通道间同步有个例子：

- 方式一：通过配置公共寄存器 **CFG_SRPWM_SFT_EVT** 可在多个通道中同时产生软件强制事件，选择软件强制事件产生 timer_0sync 或 timer_1sync，触发定时计数器初始相位的更新。
- 方式二：需做同步的多个通道均选择同一个 fpp_fault_pos (源自 XBAR 输入的 Fault 信号) 信号作为产生 timer0_sync 或 timer_1sync 的触发信号，触发定时计数器初始相位的更新。如果多个通道不能选择同一 fpp_fault_pos 信号，则选择通道 x 使用指定的 fpp_fault_pos 信号产生 sync_out，然后通过公共寄存器配置，将需做同步的各个通道的 sync_in 选择为通道 x 的 sync_out (包括通道 x 的 sync_in)。各个通道选择以 sync_in 产生 timer_0sync 或 timer_1sync，触发定时计数器初始相位更新。
- 方式三：假设各个同步通道间希望以θ度相位差同步，则可以级联方式实现通道同步。包括主通道在内的各个通道均选择 cmp_evt 作为 sync_out 的触发信号，且产生 cmp_evt 的比较配置值均应设置为对应角度**θ**，各通道的 sync_out 和 sync_in 按级

联方式配置。这种同步场景下**需注意：从 x 通道的 sync_out 触发源有效至其级联通道完成初始值加载有 3 个 SRPWM 时钟周期的延迟，因此需补偿此时延差值，可通过两种方式补偿：**

- 将主通道寄存器 **CFG_PG_CTL[cfg_pg_prior_itv]**的配置值设置为 N，N=k+3×m，m 为主通道后面级联的从通道个数，k 为级联通道中最后一级通道的 **CFG_PG_CTL[cfg_pg_prior_itv]**配置值。主通道级联的下一级通道的 **CFG_PG_CTL[cfg_pg_prior_itv]**的配置值设置为**N=k+3×(m-1)**…注意需保证触发 PWM 波翻转的有效事件间距应大于 **CFG_PG_CTL[cfg_pg_prior_itv]**配置值。**CFG_PG_CTL[cfg_pg_prior_itv]**对 PWM 波产生的影响详见 19.4.11.3 "PWM 波发生器的动作事件优先级"小节的说明。
- 软件应用程序中对这 3 个时钟周期进行补偿，比如在配置 PWM 波动作的 CMP 值时，将主通道至各级级联通道的 CMP 值依次向后调节3×m，3×(m-1)，3×(m-2)…个时钟周期。

19.4.9.3 PWM 时基模块参数的同步加载

PWM 时基模块中的 PWM 定时器的周期值 (来自于 **CFG_TB_PRD[cfg_tb_prd[20:0]]**)、初始相位值 0 (来自于 **CFG_TB_0PHASE[cfg_tb_0phase]**) 均支持立即加载、通道内影子加载及通道间同步加载模式，关于上述加载方式详见 19.4.7 "SRPWM 参数的加载"小节的说明。

说明

1. **CFG_TB_0PHASE[cfg_tb_0phase]**支持影子加载，如启用影子加载，完成参数配置后，需先等触发事件触发成为活跃值，再等待 timer_0sync 同步信号生效，配置参数才真正加载到定时计数器中；
2. **CFG_TB_1PHASE[cfg_tb_1phase]**的配置值只支持立即加载模式，完成参数配置后，只需等待 timer_1sync 同步信号生效，配置参数即可加载到定时计数器中；
3. 当启用时基同步信号 0 加载时基模块参数时，**CFG_TB_0PHASE[cfg_tb_0phase]**和 **CFG_TB_CTL[cfg_tb_0dir]**不能同时配置为 0 值，如软件实时计算出现二者同时为 0 的情况，软件应加保护：满足此条件时，强制将 **CFG_TB_CTL[cfg_tb_0dir]**置为 1；
4. 当启用时基同步信号 1 加载时基模块参数时，**CFG_TB_1PHASE[cfg_tb_1phase]**和 **CFG_TB_CTL[cfg_tb_1dir]**不能同时配置为 0 值，如软件实时计算出现二者同时为 0 的情况，软件应加保护：满足此条件时，强制将 **CFG_TB_CTL[cfg_tb_1dir]**置为 1。

19.4.9.4 SRPWM 波周期参数的计算

在 SRPWM 模块中，PWM 波的目标周期值可通过如下公式转换成寄存器 **CFG_TB_PRD[cfg_tb_prd]**的参数值：

$$\text{CFG\_TB\_PRD[cfg\_tb\_prd[20:5]]}=\begin{cases}\text{Floor}(\frac{T_{prd}}{T_{clk}}), & \text{CFG\_TB\_CTL[cfg\_tb\_md]='b1}\\ \text{Floor}\left(\frac{T_{prd}}{2\times T_{clk}}\right), & \text{CFG\_TB\_CTL[cfg\_tb\_md]='b0}\end{cases}$$

$$CFG\_TB\_PRD[cfg\_tb\_prd[4:0]] = \begin{cases} ROUND\left(\frac{T_{prd}-Floor(\frac{T_{prd}}{T_{clk}})\times T_{clk}}{T_{Hres}}\right), & CFG\_TB\_CTL[cfg\_tb\_md]='b1 \\ MOD\left(ROUND\left(\frac{T_{prd}-Floor\left(\frac{T_{prd}}{2\times T_{clk}}\right)\times 2\times T_{clk}}{T_{Hres}}\right), 32\right) & CFG\_TB\_CTL[cfg\_tb\_md]='b0 \end{cases}$$

$$CFG\_TB\_PRD[cfg\_tb\_prd\_hr\_msb] = Floor\left(ROUND\left(\frac{T_{prd}-Floor\left(\frac{T_{prd}}{2\times T_{clk}}\right)\times 2\times T_{clk}}{T_{Hres}}\right)/32\right)$$

上述三个式子中：

- T_{prd}为 PWM 波的目标周期值，单位为秒；
- T_{clk}为 SRPWM 模块的工作时钟周期值，单位为秒；
- T_{Hres}为高精度分辨率，取为156.25×10^{-12}秒；
- Floor 表示向下取证；
- ROUND 表示四舍五入取整；
- MOD(X,M) 表示 X÷M 的余数；

说明

当公共寄存器 **CFG_SRPWM_HR_EN[cfg_srpwm_hr_en[x]]**的配置值 (**x** 为当前通道号，从 0 开始计) 为 0 时，当前通道不支持高精度周期调整，**CFG_TB_PRD[cfgtbprd[4:0]]**的配置值将被忽略。

19.4.10 定时计数比较器 (TCMP)

19.4.10.1 定时计数比较器的功能

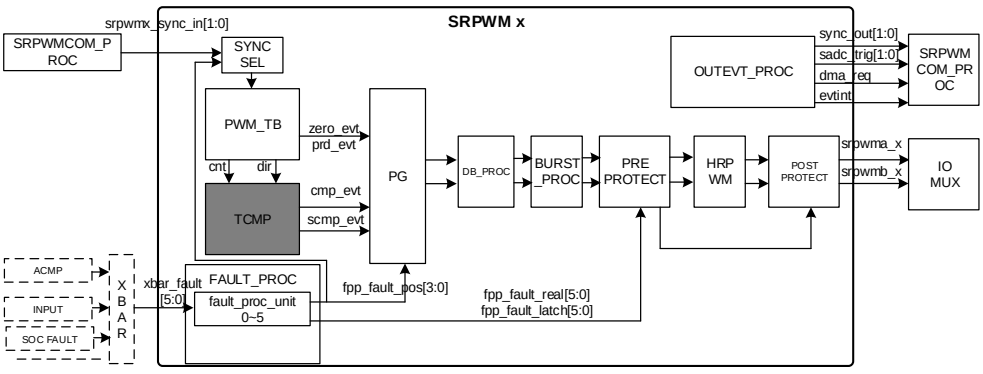


图19-16 定时计数比较器模块在 SRPWM 通道内的位置

定时计数比较器 (在 SRPWM 章节内简称定时比较器) 接收 PWM 定时器产生的 22 比特标准精度定时计数值，与软件配置的比较值进行比较，产生比较事件脉冲，用于除时基同步信号集合之外的所有事件集合中。当比较事件脉冲用于 PWM 波发生器中作为动作矩阵的触发源，可调整 PWM 波的占空比。当前定时计数比较器模块支持标准精度占空比的调整，高精度占空比的调整在高精度调整单元中实现。

每个 SRPWM 通道内均配备有 4 个主定时比较器 (CMP_A、CMP_B、CMP_C、CMP_D)

和 4 个从定时比较器 (SCMP_A、SCMP_B、SCMP_C、SCMP_D)，分别产生 4 个向上方向主比较事件 (cmp_up_evt[3:0])、4 个向下方向主比较事件 cmp_dw_evt[3:0]、4 个向上方向从比较事件 (scmp_up_evt[3:0]) 和 4 个向下方向从比较事件 (scmp_dw_evt[3:0])。

19.4.10.2 比较事件的产生

4 个主比较器的比较值 **CFG_CMP_CMPA/B/C/D[cfg_cmp_cmpa/b/c/d[26:5]]**对应产生 cmp_up_evt[3:0]或 cmp_dw_evt[3:0] (其中 **CFG_CMP_CMPA[cfg_cmp_cmpa[26:5]]**产生 cmp_up_evt[0]或 cmp_dw_evt[0])。

4 个从比较器的比较值 **CFG_CMP_SCMPA/B/C/D[cfg_cmp_scmpa/b/c/d[26:5]]**对应产生 scmp_up_evt[3:0]或 scmp_dw_evt[3:0] (其中 **CFG_SCMP_CMPA[cfg_cmp_scmpa[26:5]]** 产生 scmp_up_evt[0]或 scmp_dw_evt[0])。

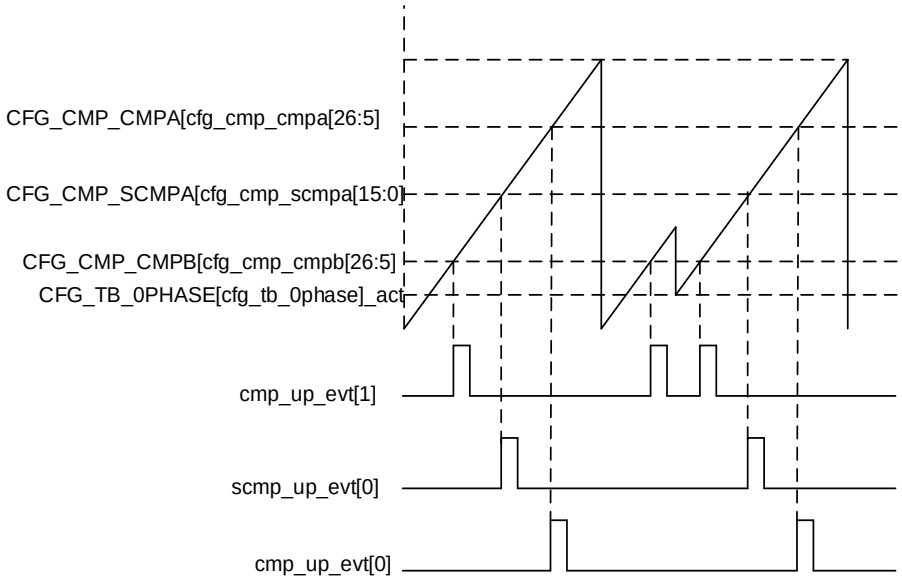


图19-17 向上计数比较事件产生

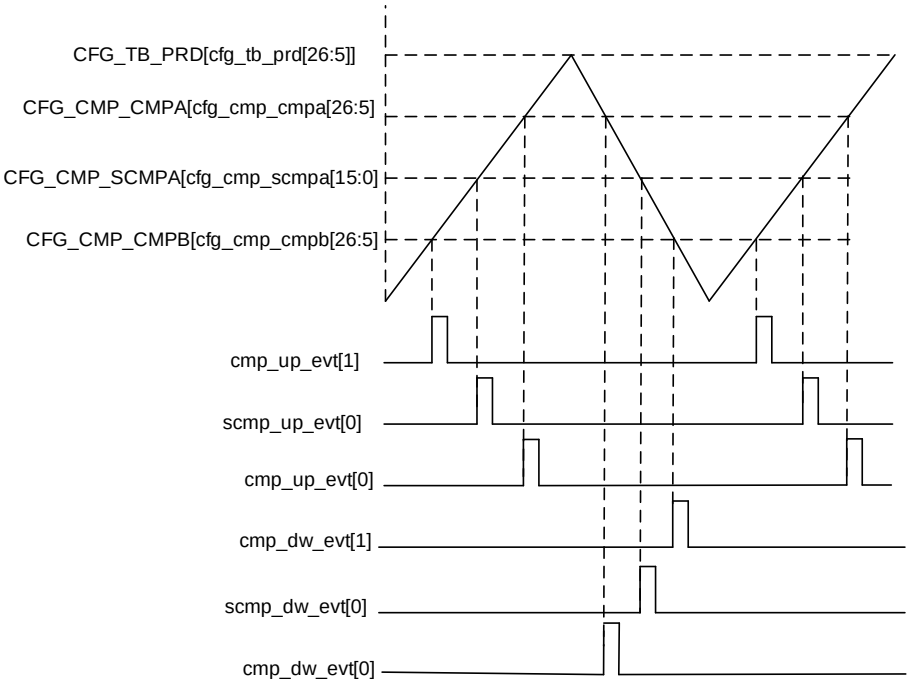


图19-18 向上向下计数模式比较事件的产生

主、从比较器的区别为：从比较器仅支持定时计数值等于比较配置值时产生比较事件脉冲，而主比较器既可配置为仅在计数值等于比较配置值时产生比较事件，也可配置为向上计数的计数值大于等于比较配置值或是向下计数的计数值小于等于比较配置值时产生比较事件。(由 **CFG_CMP_CTL[cfg_cmp_md]**控制)。

说明

定时计数值大于等于 (或小于等于) 比较配置值而产生的比较事件脉冲与定时计数值等于比较配置值而产生的比较事件脉冲宽度是一样，都是单时钟周期脉冲。

比较配置值可以在工作过程中实时更新，且更新时刻可以同步于指定的触发事件。在下图中，定时计数器配置为向上计数器，主比较器比较条件配置为大于等于比较配置值，图中给出了不同更新场景下的比较事件脉冲产生情况：

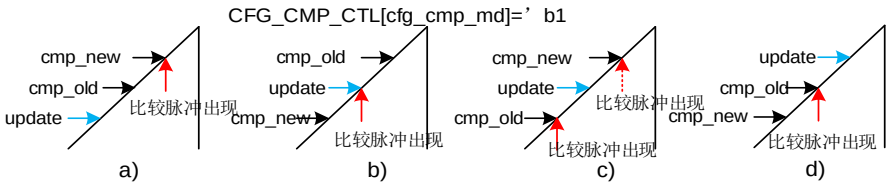


图19-19 配置为大于等于条件下的几种场景

上图中 cmp_old 和 cmp_new 分别对应比较配置值的老值和新值，update 是指新配置值更新时刻，垂直方向的箭头是指产生比较事件脉冲出现的时刻。

下图与上图更新场景相同，但比较条件配置为等于时，场景 **b)**在当前 PWM 周期内无法产生向上比较脉冲，因为场景 **b)**中新配置值更新时刻，定时计数值已大于比较配置值。类似的情形也适用于向下计数方向。

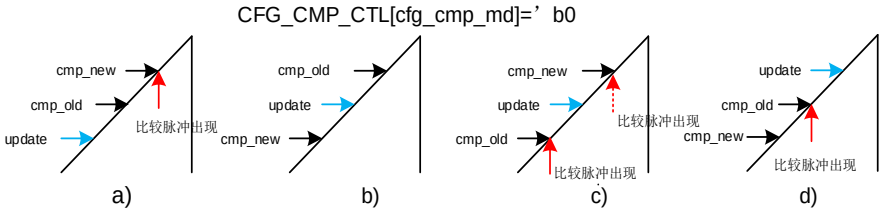


图19-20 配置为等于条件下的几种场景

由于 4 个从比较器只支持等于条件产生比较事件，因此 **CFG_CMP_CTL[cfg_cmp_md]** 的配置不影响从比较事件的产生。

另外，通过配置寄存器 **CFG_CMP_CTL[cfg_cmp_onetime]**，可以控制在向上或向下方向上是否仅产生一次比较脉冲。如果 **CFG_CMP_CTL[cfg_cmp_onetime]**配置为**'b1**，上面两幅图中场景 **c)**的垂直虚线对应的时刻不会产生比较事件脉冲。这个寄存器配置对于主比较器和从比较器均适用。

此外还需注意，在图 19-19 中，场景 **d)**更新后的新比较值小于更新时刻对应的定时计数值，比较条件满足，但更新时刻不会再产生一次比较事件脉冲，因为更新前后定时计数值一直大于等于配置比较值，该条件满足的情况并未发生变化。

19.4.10.3 比较器参数的同步加载

主比较器的比较值(对应寄存器 **CFG_CMP_CMPA**、**CFG_CMP_CMPB**、**CFG_CMP_CMPC**、**CFG_CMP_CMPD**)和从比较器的比较值(对应寄存器**CFG_CMP_SCMPA**、**CFG_CMP_SCMPB**、**CFG_CMP_SCMPC**、**CFG_CMP_SCMPD**)支持立即加载、通道内影子加载及通道间同步加载模式，关于上述加载方式详见 19.4.7 "SRPWM 参数的加载"小节的说明。

19.4.11 PWM 波发生器 (PG)

19.4.11.1 PWM 波发生器的功能

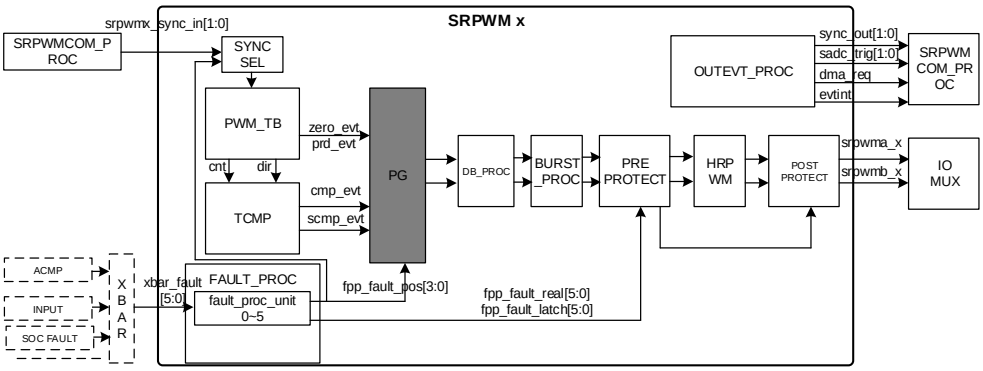


图19-21 PWM 波发生器在 SRPWM 通道内的位置

PWM 波发生器可响应动作矩阵事件集合 (pg_evt) 的 22 个事件，根据寄存器组 **CFG_PG_CTLA_GRPx** (对应 a 支路) 和 **CFG_PG_CTLB_GRPx** (b 支路) 的动作配置值，在相应事件出现时，产生相应动作。

22 个动作矩阵事件包括：

- {prd_evt, scmp_dw_evt[0], scmp_dw_evt[1], scmp_dw_evt[2], scmp_dw_evt[3], cmp_evt_down[0], cmp_evt_down[1], cmp_evt_down[2], cmp_evt_down[3], zero_evt, scmp_up_evt[0], scmp_up_evt[1], scmp_up_evt[2], scmp_up_evt[3], cmp_up_evt[0], cmp_up_evt[1], cmp_up_evt[2], cmp_up_evt[3], fpp_fault_pos[3:0]}

寄存器组 **CFG_PG_CTLA_GRPx** 中偏移地址 0x80 对应事件 fpp_fault_pos[0]、偏移地址 0x84 对应事件 fpp_fault_pos[1]……寄存器组 **CFG_PG_CTLB_GRPx** 中偏移地址 0xe0 对应事件 fpp_fault_pos[0]、偏移地址 0xe4 对应事件 fpp_fault_pos[1]……

对于上述每一个事件，有 4 个动作可供配置选择：

- 'd0：不动作；
- 'd1：PWM 波信号置 0；
- 'd2：PWM 波信号置 1；
- 'd3：PWM 波信号取反;

在一个 SRPWM 通道内支持产生 a/b 两支路的 PWM 波信号，a/b 两路的动作矩阵配置完全独立，互不相干。

19.4.11.2 PWM 波发生器的软件强制动作

在 SRPWM 通道的 PWM 波发生器中，除了可以响应前述的 22 个事件产生 PWM 波，还可响应软件配置的强制事件产生 PWM 波。软件强制分为单次模式和连续模式：

- 公共寄存器 **CFG_PG_OTSFT[cfg_pg_otsfta]**和 **CFG_PG_OTSFT[cfg_pg_otsftb]** 分别对 a 支路和 b 支路 PWM 波信号进行一次强制动作。

- 公共寄存器 **CFG_PG_CSFT[cfg_pg_csfta]**和 **CFG_PG_OTSFT[cfg_pg_csftb]**分别对 a 支路和 b 支路 PWM 波信号进行连续的强制动作。

当进行单次强制动作时，其操作及相应流程如下：

- 软件向公共寄存器 **CFG_PG_OTSFT[cfg_pg_otsfta/b[x]]**写入一次**'b1**；
- 硬件逻辑响应该写入行为，产生单次触发脉冲；
- 等到 start 事件有效，会执行相应动作(动作内容由寄存器 **CFG_PG_CTL_SFT[cfg_pg_ctl_sfta/b]**配置的软件强制内容决定)，之后等 End 事件有效，该次软件强制动作结束；
- Start 事件和 End 事件分别由寄存器 **CFG_PG_SFT_SYNCA/B[cfg_pg_sftstart_sync_msela/b]** 和 **CFG_PG_SFT_SYNCA/B[cfg_pg_sftend_sync_msela/b]**，按 Mask_OR 方式从 PG 同步事件集合 (pgsft_evt) 中选定；
- 由于软件写一次只产生一个脉冲，因此动作持续的时间是从 Start 事件到 End 事件的间隔时间。软件强制动作结束后恢复响应前述 22 个事件的动作。

当进行连续强制动作时，其操作及相应流程如下：

- 软件向公共寄存器 **CFG_PG_CSFT[cfg_pg_csfta/b[x]]**写入一次**'b1**，产生持续的高电平事件，直至软件向该寄存器写入**'b0** 该高电平事件才终止；
- 如果寄存器 **CFG_PG_CTL[cfg_pg_sftstart_mda/b]** 置为 **'b0** ，**CFG_PG_CSFT[cfg_pg_csfta/b[x]]**写入 **1** 后立即开始强制动作 (动作内容由寄存器 **CFG_PG_CTL_SFT[cfg_pg_ctl_sfta/b]**决定)，否则需等待 Start 事件生效后才开始执行相应动作；
- 如果寄存器 **CFG_PG_CTL[cfg_pg_sftend_mda/b]** 置为 **'b0** ，**CFG_PG_CSFT[cfg_pg_csfta/b[x]]**写入 **0** 后立即结束强制动作 (动作内容由寄存器 **CFG_PG_CTL_SFT[cfg_pg_ctl_sfta/b]**决定)，否则需等待 End 事件生效后才开始执行相应动作；
- start 事件和 end 事件分别由该通道的寄存器 **CFG_PG_SFT_SYNCA/B[cfg_pg_sftstart_sync_msela/b]** 和 **CFG_PG_SFT_SYNCA/B[cfg_pg_sftend_sync_msela/b]**，按 Mask_OR 方式从 PG 同步事件集合 (pgsft_evt) 中选定；

说明

软件强制动作的优先级高于前述 22 个事件，因此发生软件强制事件期间，PWM 波发生器不再响应前述 22 个事件定义的动作，直至软件强制事件结束后，恢复响应前述 22 个事件动作。

19.4.11.3 PWM 波发生器的动作事件优先级

PWM 波发生器支持动作优先级可配置：在一个可配时间内 (4-bit) 如果多个事件同时触发，只响应优先级高的事件：

- 向上向下模式的向上阶段：
  - 软件强制事件 > fpp_fault_pos[0]> fpp_fault_pos[1]> fpp_fault_pos[2]> fpp_fault_pos[3]> cmp_up_evt[3] (CMPD) > cmp_up_evt[2] (CMPC) >

cmp_up_evt[1] (CMPB) > cmp_up_evt[0] (CMPA) > scmp_up_evt[3] (SCMPD) > scmp_up_evt[2] (SCMPC) > scmp_up_evt[1] (SCMPB) > scmp_up_evt[0] (SCMPA) > zero_evt

- 向上向下模式的向下阶段：
  - 软件强制事件 > fpp_fault_pos[0] > fpp_fault_pos[1] > fpp_fault_pos[2] > fpp_fault_pos[3] > cmp_dw_evt[3] (CMPD) > cmp_dw_evt[2](CMPC) > cmp_dw_evt[1] (CMPB) > cmp_dw_evt[0] (CMPA) > scmp_dw_evt[3] (SCMPD) > scmp_dw_evt[2] (SCMPC) > scmp_dw_evt[1] (SCMPB) > scmp_dw_evt[0] (SCMPA) > prd_evt
- 向上模式：
  - 软件强制事件 > fpp_fault_pos[0] > fpp_fault_pos[1] > fpp_fault_pos[2] > fpp_fault_pos[3] > prd_evt > cmp_up_evt[3] (CMPD) > cmp_up_evt[2] (CMPC) > cmp_up_evt[1] (CMPB) > cmp_up_evt[0] (CMPA) > scmp_up_evt[3] (SCMPD) > scmp_up_evt[2] (SCMPC) > scmp_up_evt[1] (SCMPB) > scmp_up_evt[0] (SCMPA) > zero_evt

寄存器 **CFG_PG_CTL[cfg_pg_prior_itv]**配置优先级判断间隔，其配置值对动作响应的影响为：

- 当其配置为’d0，表示无滤波，当多个事件同时生效时，按照上述优先级关系，响应优先级高的事件，当多个事件先后生效时，依次按照各个事件的动作矩阵配置完成相应动作；
- 当其配置为 N，表示滤 N 个 cycle：当多个事件同时生效时，按照上述优先级关系，响应优先级高的事件，当多个事件先后生效时，且先后时间间隔小于等于 N，则对这些事件进行优先级判断，只响应优先级最高的事件，**此时需注意，动作响应 (最高优先级事件) 的时间会比无滤波配置时延迟 N 个 SRPWM 时钟周期**。

在工作过程中，参与以上优先级排列和判断的事件，其对应的动作应配置为三种选项之一：置 1/置 0/取反，动作配置为“不动作”的事件默认优先级最低，不参与以上优先级判断。

19.4.11.4 PWM 波产生例子

下图给出了向上向下计数模式下，产生不同占空比 PWM 波的例子：

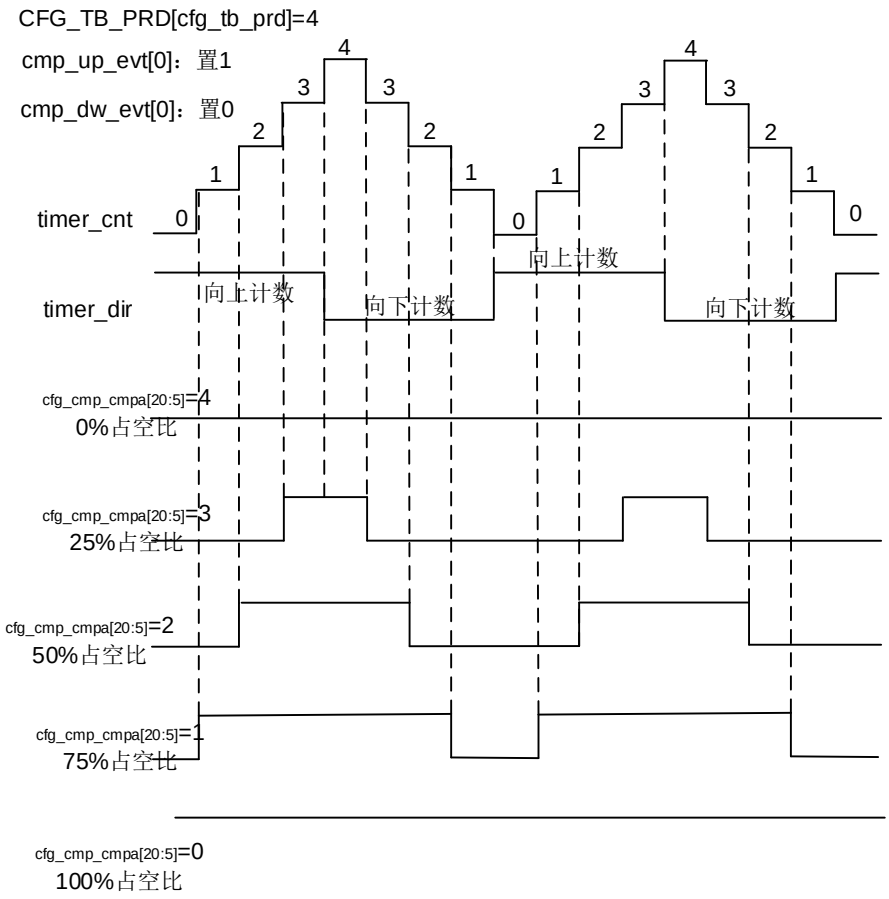


图19-22 向上向下计数各种占空比波形产生

下图给出了向上计数模式下，利用比较器事件、周期和过零事件产生 PWM 波的例子。

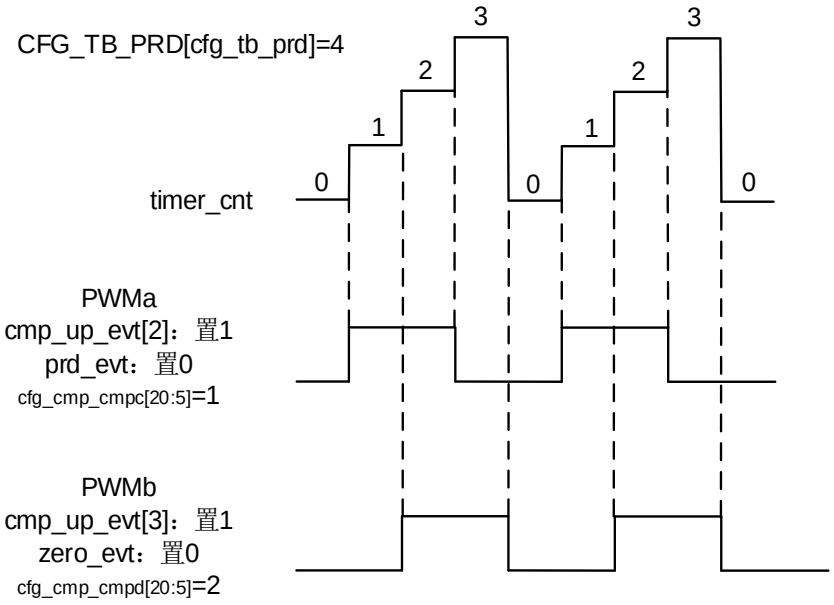


图19-23 向上计数 PWM 波形产生

19.4.11.5 PWM 波发生器参数的同步加载

控制 PWM 波产生的寄存器组 **CFG_PG_CTLA_GRP** 和 **CFG_PG_CTLB_GRP** 支持立即加载、通道内影子加载及通道间同步加载模式，关于上述加载方式详见 19.4.7 "SRPWM 参数的加载"小节的说明。

19.4.12 死区调整 (DB_PROC)

19.4.12.1 死区调整的功能

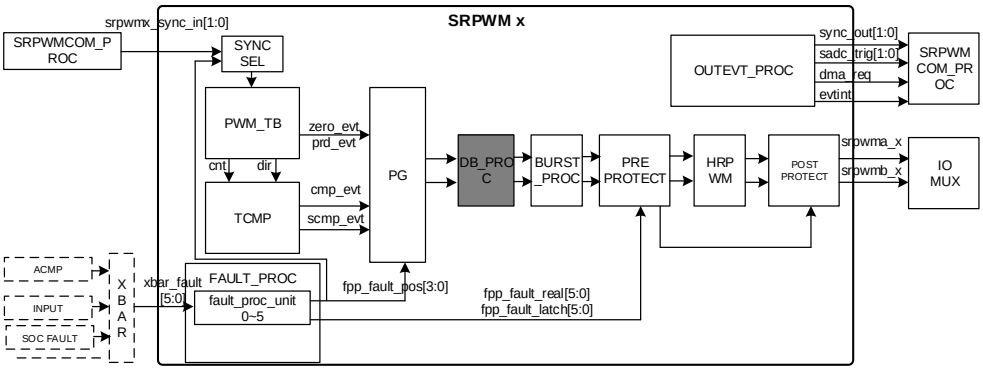


图19-24 死区调整模块在 SRPWM 通道内的位置

数字电源控制中，一般要求上下管的开关具有一定死区，在电平翻转时插入一定时间间

隔避免上下管同时导通。虽然可以通过配置不同的 CMPa/b/c/d 比较时刻以实现死区调整，但是这种配置既不灵活，也不方便，所以需要通过专门的死区模块调整 a/b 支路上升沿/下降沿位置，使得 a/b 通道的 PWM 波不会同时生效。

19.4.12.2 死区调整流程

PWM 波发生器产生的 a、b 两支路发波信号在死区调整模块中经过下图所示的流程：

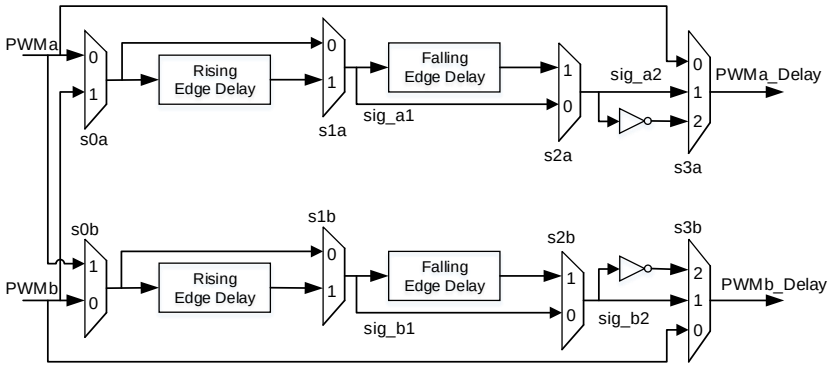


图19-25 死区调整的信号处理流程

上图中 sxa/b 是指寄存器 **CFG_DB_SEL[cfg_db_sxa/b]**的配置值，Rising Edge Delay 由寄存器 **CFG_DB_A/BPOS_DLY** 配置，Falling Edge Delay 由寄存器 **CFG_DB_A/BNEG_DLY** 配置。

上图中第一级选择和最后一级选择前的取反操作，可实现死区输出信号的互补功能。

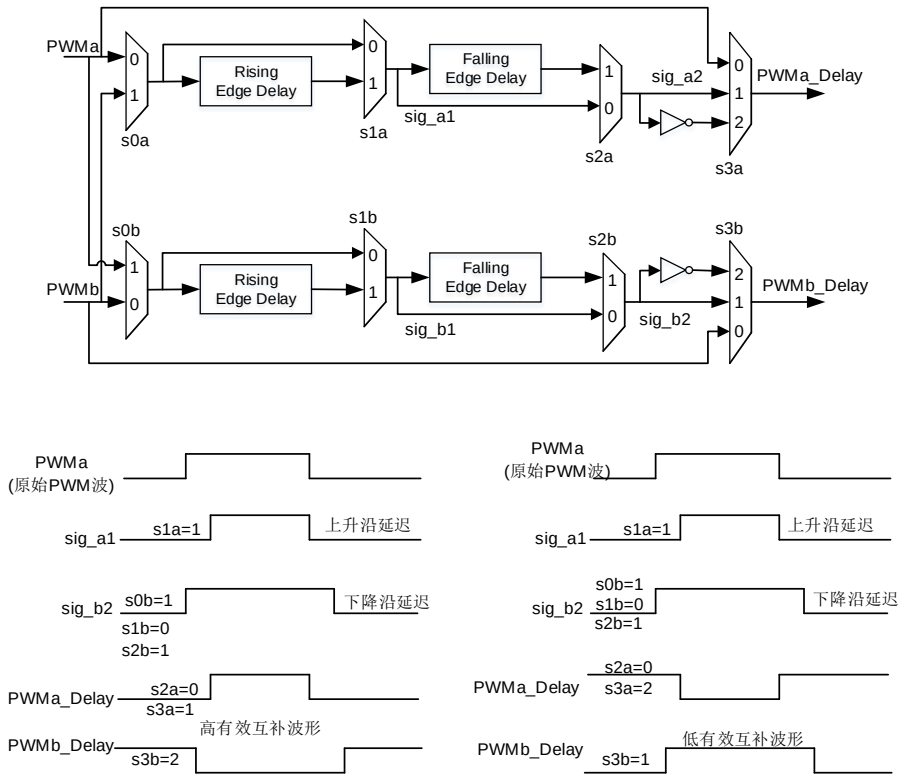


图19-26 上/下沿死区调整效果示意

当前模块可实现标准精度死区调整，高精度死区调整在高精度模块中实现。

19.4.12.3 死区调整参数的同步加载

死区上下沿调整参数（对应寄存器 **CFG_DB_A/BPOS_DLY** 和 **CFG_DB_A/BNEG_DLY**）支持立即加载，单通道影子加载，多通道同步加载，参见 19.4.7 "SRPWM 参数的加载"小节的描述。

19.4.13 Burst 发波 (BURST_PROC)

19.4.13.1 Burst 发波功能

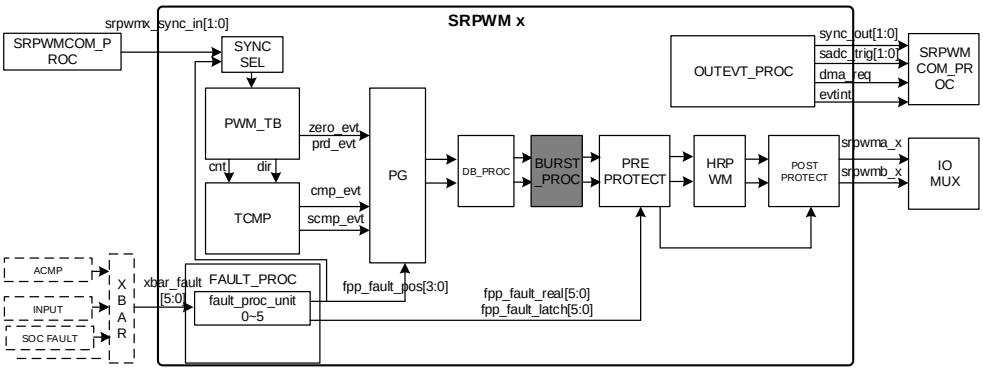


图19-27 Burst 发波模块在 SRPWM 通道内的位置

死区调整输出的 PWM 波信号可进一步进行突发发波处理 (Burst 发波处理)，设置重复的突发周期 (每个突发周期包含若干个 PWM 波周期)，在每一个突发周期内指定发送若干个周期的 PWM 波，其余时间则不发波。通过设置 PWM 波周期发出高频 PWM 波，结合这种突发处理方式，可以达到类似于斩波的效果。

19.4.13.2 Burst 发波配置

Burst 发波功能可以通过配置相关寄存器启用或禁止 (参见 **CFG_BURST_CRL** 寄存器说明)。如启用 Burst 发波模式，需配置 Burst 发波的周期 M，单位为 PWM 波周期，即一个 Burst 周期包含 M 个 PWM 波周期，其次还需配置有效发波个数值 N，指定在一个 Burst 周期中，有 N 个连续周期会发出 PWM 波，其余 M-N 个周期不会对外输出 PWM 波。另外还可以通过寄存器 **CFG_BURST_CRL** 配置，指定这 N 个发波周期位于 Burst 周期的起始时间段还是在 Burst 周期的结束时间段。Burst 模式的起始和结束都需要与事件同步，通过配置寄存器 **CFG_BURST_SYNC_SEL**，可以从 Burst 同步事件集合 (burst_evt) 以 Mask_OR 方式选择一个或多个事件，作为 Burst 工作所需的同步触发事件。Burst 模式是在 **CFG_BURST_CRL[cfg_burst_en]**置为**'b1** 后，该同步触发事件生效才正式生效。Burst 模式的结束通常是由于外部条件发生了变化，比如电压低于某个阈值，需要结束 Burst 模式，此时需要由软件将 **CFG_BURST_CRL[cfg_burst_en]**置为**'b0**，待走完整个 Burst 周期 (包含 M 个 PWM 发波周期)，在前述同步触发事件的触发下结束。

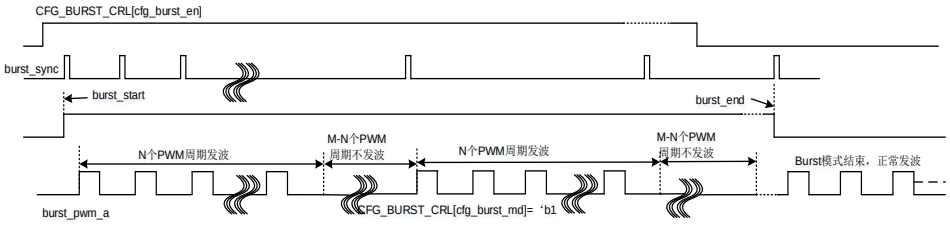


图19-28 Burst 周期起始先发波

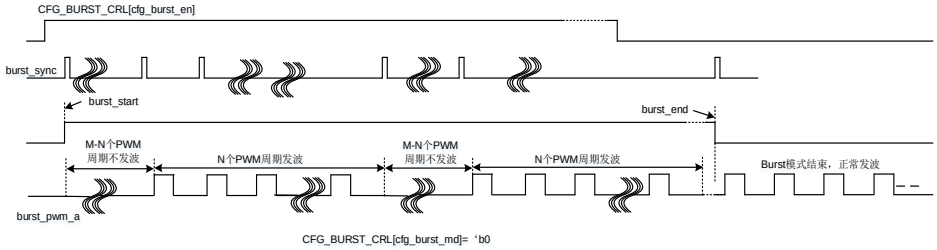


图19-29 Burst 周期尾段发波

19.4.13.3 Burst 参数的同步加载

寄存器 **CFG_BURST_MN** 支持支持支持立即加载、通道内影子加载及通道间同步加载模式，关于上述加载方式详见 19.4.7 "SRPWM 参数的加载"小节的说明。

说明

强烈建议 **CFG_BURST_LDSYNC[cfg_burst_ldysnc_msel]** 选定的同步信号和 **CFG_BURST_SYNC_SEL[cfg_burst_sync_msel]**选定的 Burst 起始同步信号为相同的信号。

19.4.14 前级保护 (PREPROTECT)

19.4.14.1 前级保护的功能

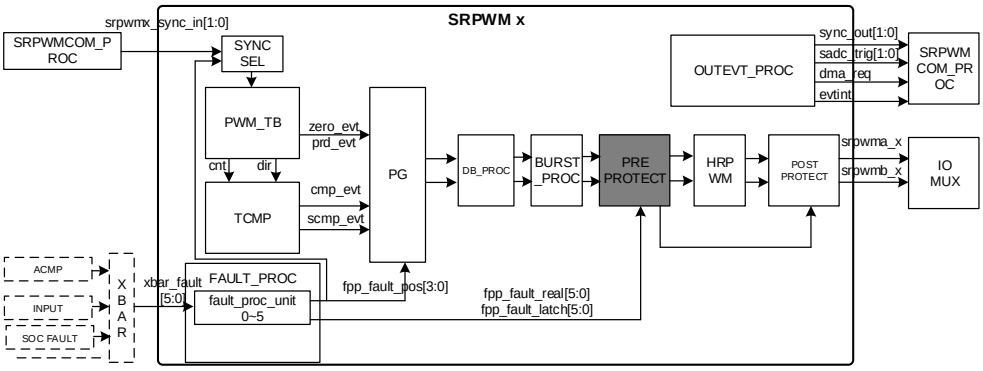


图19-30 前级保护模块在 SRPWM 通道内的位置

SRPWM 模块包含了前级保护和次级保护两级保护逻辑，可以在外部电路发生故障或芯片内部出现告警时将输出的 PWM 波信号置于预置的安全状态。其中前级保护基于 SRPWM 模块工作时钟驱动，次级保护的保护行为无需时钟驱动。

前级保护的保护触发信号来自于 Fault 信号处理的 6 比特结果 fpp_fault_real[5:0]和 fpp_fault_latch[5:0]，fpp_fault_real 和 fpp_fault_latch 的产生流程详见 19.4.8 "Fault 信号处理(FAULT_PROC)"小节描述。这些触发信号的来源是芯片 GPIO 输入的故障信号、芯片内部 ACMP 结果、芯片内部的各类告警信号，如下图：

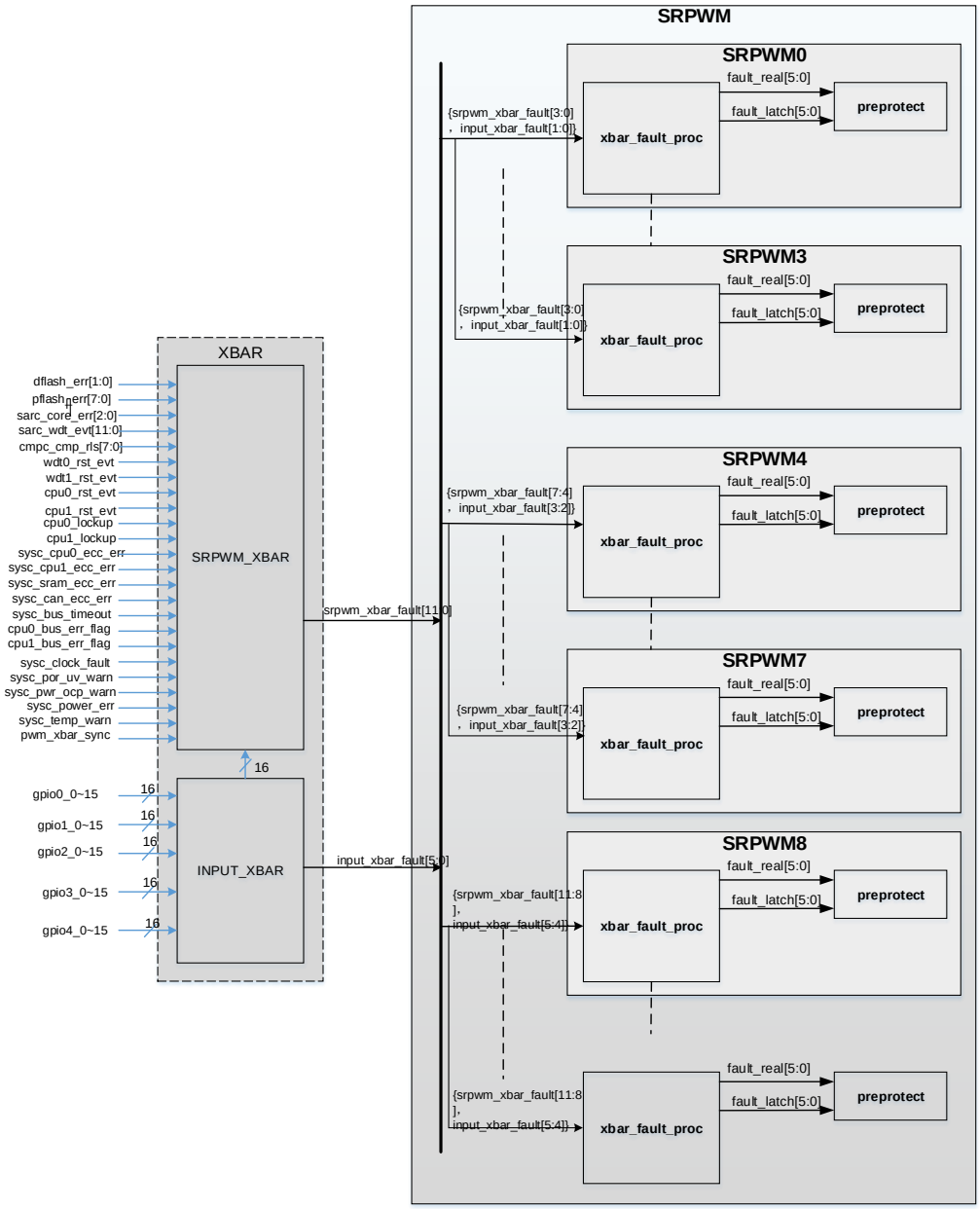


图19-31 前级及次级保护模块的故障触发信号来源

前级保护生效时可通过寄存器预先配置，选择将 PWM 波信号置于以下四种状态：

- 目标 PWM 信号对应的输出管脚强制置为高阻态；
- PWM 输出强制置 0；
- PWM 输出强制置 1；
- PWM 输出强制取反 (相较于保护生效时刻的状态取反)。

前级保护有三种保护模式：

- CBC (逐周期) 保护模式：故障触发信号有效，立即将 PWM 波信号置于前述四种预置状态之一，故障触发信号消除，保护动作并不立即撤销，需等待同步事件触发后再撤销；
- OS (单次) 保护模式：通过寄存器 **CFG_PREP_SYNC_MD** 配置，可选择故障信号

有效后，立即开始保护动作或是等待同步事件触发后才开始保护，故障信号消除后，立即撤销保护动作或等待同步事件触发后才撤销保护；

- UNRES (无约束) 保护模式：故障信号有效，立即将 PWM 波信号置于前述四种预置状态之一，故障触发信号消除，立即撤销保护动作。

针对上述三种模式，均可独立配置其对应的保护动作 (高阻、置 0、置 1 或取反)。

在每个 SRPWM 通道中包含 a、b 两支路 PWM 波信号，保护模式和保护动作在 a、b 支路上独立配置，互不关联。

另外，强制置 0 和强制置 1 动作均支持生效延后功能，当寄存器 **CFG_PREP_SETA/B_DLY、CFG_PREP_CLRA/B_DLY** 的配置值不为 0 时，动作的生效时刻相应延后 (动作结束时刻不受影响)，详见相应寄存器说明。

当寄存器 CFG_EPWM_FAST_MASK[cfg_fpp_fast_mask]配置为'b0 时，支持快速封波模式，封波动作不经过任何时钟打拍逻辑，延时最短。建议在应用中采用快速封波模式。

注意：

由于上述寄存器的配置可独立进行，因此存在同时启用 CBC、OS 和 UNRES 模式的可能，此时各模式对应的动作可做相同配置也可做不同配置，当不同模式的动作事件同时出现时，各个动作的优先级从高到低依次为：置高阻、置 0、置 1、取反。当不同工作模式的同一种动作对应的触发事件同时出现，保护动作的开始与最早开始的动作事件对齐，保护动作的结束与最晚结束的动作事件对齐，下图给出了一个极端例子，示出当同时启用了 OS 模式和 UNRES 模式，且两种模式的保护动作都选为置 0 的保护场景。

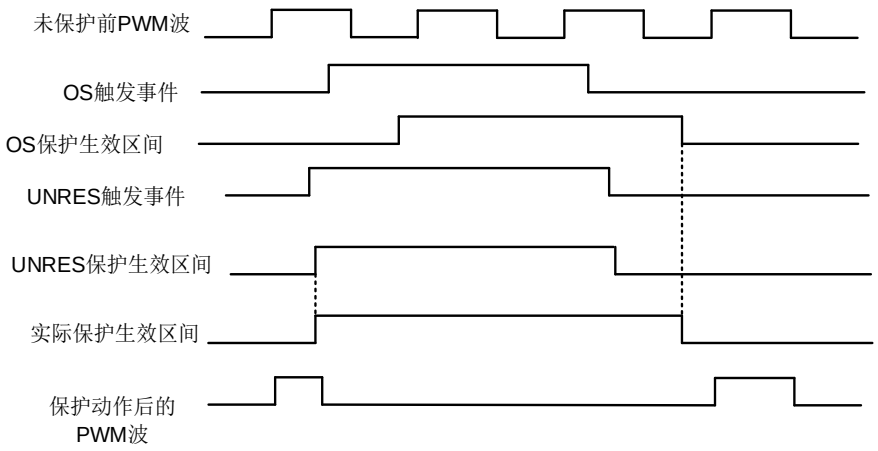


图19-32 OS 模式和 UNRES 模式同时启用保护

19.4.14.2 CBC 保护模式

通过配置寄存器 **CFG_PREP_CBC_MSEL**，在触发事件{cfg_prep_cbc_sft，fpp_fault_latch，fpp_fault_real}中以 Mask_OR 方式选择 1 个或多个事件，可启用 CBC 保护模式。cfg_prep_cbc_sft 事件是由软件往公共寄存器 **CFG_PREP_CBC_SFT[cfg_prep_cbca/b_sft]**写入**'b1** 产生脉冲得到 (每写一次'b1，产生一个脉冲) 的软件强制事件。

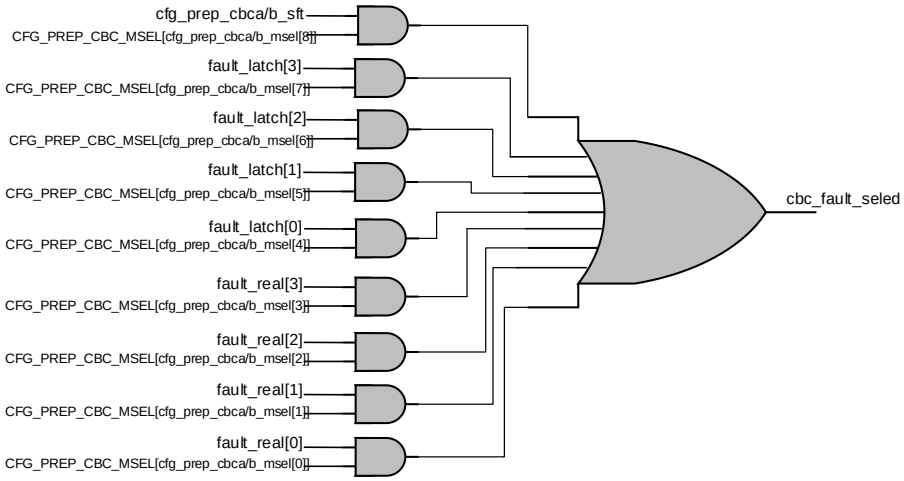


图19-33 CBC 保护模式的 Fault 信号以 Mask_OR 方式选择

在 CBC 保护模式下，当选定的保护触发事件 (上图中 cbc_fault_seled) 生效，保护动作 (由寄存器 **CFG_PREP_PROT_ST** 定义) 起作用，当选定的故障事件消除，需等待同步触发事件 (由寄存器 **CFG_PREP_SYNC_SEL** 选择) 生效后，保护动作方可解除。

19.4.14.3 OS 保护模式

通过配置寄存器 **CFG_PREP_OSC_MSEL**，在触发事件{cfg_prep_os_sft，fpp_fault_latch，fpp_fault_real}中以 Mask_OR 方式选择 1 个或多个事件，可启用 OS 保护模式。cfg_prep_os_sft 事件源自软件往公共寄存器 **CFG_PREP_OS_SFT[cfg_prep_osa/b_sft]**写入**'b1**，强制产生触发事件高电平。要消除此强制事件，需由软件往该寄存器相应位置写入**'b0**，这点与 cfg_prep_cbc_sft 事件不同，需注意。

在 OS 保护模式中，保护触发事件生效 (或消除) 后，相应的保护动作是立即生效 (或立即消除)，需视寄存器**CFG_PREP_SYNC_MD** 的配置情况而定。该寄存器相应位置为'b1，则保护触发事件生效 (或消除) 后，仍需等待同步事件 (**CFG_PREP_SYNC_SEL** 配置) 满足，才开始进入 (或结束) 保护动作。

19.4.14.4 UNRES 保护模式

UNRES 保护模式中，当故障事件生效，立即进入保护动作，当故障事件消除，保护动作的结束时间需视寄存器 **CFG_PREP_UNRES_MIN_TIME** 的配置而定。该寄存器定义了 UNRES 保护状态的最小持续时间，即使故障事件消失，保护动作的持续时间需大于等于 **CFG_PREP_UNRES_MIN_TIME** 的配置值，方可结束保护动作。

19.4.15 高精度调整单元 (HRPWM)

19.4.15.1 高精度调整单元的功能

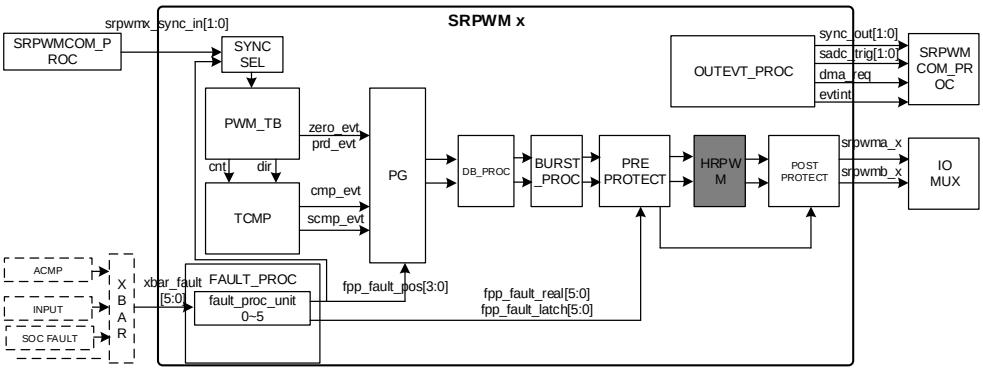


图19-34 高精度调整单元在 SRPWM 通道内的位置

高精度调整单元接收 PWM 定时器产生的高精度周期参数，结合软件配置的高精度定时计数比较值、高精度死区上下沿调整值，对经过前级保护处理的 PWM 波信号进行高精度延时调整，实现对 PWM 周期、占空比以及死区的微调，调整精度可达 156 ps。

周期、占空比以及死区的高精度调整可以联合同时进行，即支持 PWM 波的周期、占空比及死区同时进行调整。

19.4.15.2 高精度调整单元的参数

19.4.15.2.1 高精度周期调整参数

产生实时周期参数所需的配置值计算详见 19.4.9.4 "SRPWM 波周期参数的计算"小节。

19.4.15.2.2 高精度占空比调整参数

高精度占空比的调整包括压缩高脉冲宽度或扩展高脉冲宽度。

根据寄存器 **CFG_DB_SEL[cfg_db_s0a]**的取值不同，a 支路的高电平脉冲压缩和扩展调整参数可分别从不同的寄存器配置获得：

- **CFG_DB_SEL[cfg_db_s0a]='b0**：a 支路 PWM 波高电平压缩时间由寄存器 **CFG_CMP_CMPA[cfg_cmp_cmpa[4:0]]**提供，a 支路 PWM 波高电平扩展调整参数由寄存器 **CFG_CMP_CMPB[cfg_cmp_cmpb[4:0]]**提供，LSB 对应 156.25 ps；
- **CFG_DB_SEL[cfg_db_s0a]='b1**：a 支路 PWM 波高电平压缩时间由寄存器 **CFG_CMP_CMPC[cfg_cmp_cmpc[4:0]]**提供，a 支路 PWM 波高电平扩展时间由寄存器 **CFG_CMP_CMPD[cfg_cmp_cmpd[4:0]]**提供，LSB 对应 156.25 ps。

根据寄存器 **CFG_DB_SEL[cfg_db_s0b]**的取值不同，b 支路的高电平脉冲压缩和扩展调整参数可分别从不同的寄存器配置获得：

- **CFG_DB_SEL[cfg_db_s0b]='b0**：b 支路 PWM 波高电平压缩时间由寄存器 **CFG_CMP_CMPC[cfg_cmp_cmpc[4:0]]**提供，b 支路 PWM 波高电平扩展时间由

寄存器 **CFG_CMP_CMPD[cfg_cmp_cmpd[4:0]]**提供，LSB 对应 156.25 ps；

- **CFG_DB_SEL[cfg_db_s0b]='b1**：b 支路 PWM 波高电平压缩时间由寄存器 **CFG_CMP_CMPA[cfg_cmp_cmpa[4:0]]**提供，b 支路 PWM 波高电平扩展时间由寄存器 **CFG_CMP_CMPB[cfg_cmp_cmpb[4:0]]**提供，LSB 对应 156.25 ps。

如要进行高精度占空比调整，建议用 CMPa/b/c/d 作为相应的置 1 或置 0 触发事件。

高精度占空比调整参数的计算公式如下：

$$\text{Duty\_adj\_value}=\text{ROUND}(\frac{T_{duty\_adj}}{T_{Hres}})$$

上式中Duty_adj_value为高电平脉冲压缩或扩展的配置值，T_{duty_adj}为高电平脉冲压缩或扩展的时间值，单位为秒，$T_{Hres}=156.25\times10^{-12}$秒。

19.4.15.2.3 高精度死区调整参数

死区的高精度调整涉及上升沿的延迟和下降沿的延迟。

- a 支路死区上、下沿高精度调整参数获取如下：
  - 当 **CFG_DB_SEL[cfg_db_s3a]**的值为**'d0**，意味着不做死区调整，a 支路上、下沿高精度调整参数均会被强制置零，与软件配置值无关；
  - 当 **CFG_DB_SEL[cfg_db_s3a]**的值大于**'d0**，如果 **CFG_DB_SEL[cfg_db_s1a]**的值为 **'b1**，a 支路上升沿调整参数由寄存器 **CFG_DB_APOS_DLY[cfg_db_apos_dly[4:0]]** 给出，如果 **CFG_DB_SEL[cfg_db_s1a]**的值为**'b0**，意味着不做死区上升沿调整，a 支路上升沿高精度调整参数均会被强制置零，与软件配置值无关；
  - 当 **CFG_DB_SEL[cfg_db_s3a]**的值大于**'d0**，如 **CFG_DB_SEL[cfg_db_s2a]**的值为 'b1，a 支路下降沿调整参数由寄存器 **CFG_DB_APOS_DLY[cfg_db_aneg_dly[4:0]]** 给出，如 **CFG_DB_SEL[cfg_db_s2a]**的值为**'b0**，意味着不做死区下降沿调整，a 支路下降沿高精度调整参数均会被强制置零，与软件配置值无关。
- b 支路死区上、下沿高精度调整参数的获取与 a 支路类似。

  软件配置的高精度死区调整参数的计算公式如下：

$$\text{DB\_adj\_value}=\text{ROUND}(\frac{T_{DB\_adj}}{T_{Hres}})$$

  上式中DB_adj_value为死区上升沿或下降沿做高精度延迟调整的配置值，T_{DB_adj}为死区上升沿或下降沿调整的时间值，单位为秒，$T_{Hres}=156.25\times10^{-12}$秒。

19.4.15.3 PWM 波的滤波

在高精度调整单元中，可通过寄存器 CFG_HRPWM_FILTER 控制是否启用 PWM 波滤波。启用滤波功能后，PWM 波信号将会受到如下影响：

- **CFG_HRPWM_FILTER[cfg_hrpwm_filter_len]**的配置值为**'d0**，则输入输出的脉冲宽度不变，但输入输出之间有 1 个 SRPWM 工作时钟周期的延迟 (如果不启用滤波功能，输入输出完全一致，没有延迟)；

- **CFG_HRPWM_FILTER[cfg_hrpwm_filter_len]**的配置值为 **N** (N>0，且 N 的配置值应保证不会出现跨越 PWM 计数周期进行滤波的情况出现)，如果 PWM 波信号的高电平脉冲宽度小于等于 N，则该高电平脉冲将会被滤除，如果 PWM 波信号的高电平脉冲宽度大于 N，滤波器输出的 PWM 波信号高电平脉冲宽度将减少 N 个 SRPWM 工作时钟周期，且滤波输出脉冲上升沿比滤波输入脉冲的上升沿延迟 N+1 个 SRPWM 工作时钟周期。

19.4.15.4 高精度调整功能的旁路

公共寄存器 **CFG_SRPWM_HR_EN [cfg_srpwm_hr_en]**控制高精度周期的调整使能，对高精度占空比及高精度死区调整功能没有影响。寄存器 **CFG_HRPWM_BPS[cfg_hrpwm_bps]**置**'b1** 可旁路整个高精度调整过程 (完全旁路周期、占空比及死区的高精度调整)，直接将 PWM 滤波输出作为 PWM 波输出信号。

注意：

- 如果 **CFG_HRPWM_BPS[cfg_hrpwm_bps]**置**'b0**，但周期高精度调整、占空比调整参数及死区调整均置为'd0，PWM 波信号也不会进行高精度调整。
- 如果 **CFG_HRPWM_BPS[cfg_hrpwm_bps]**置为**'b1**，注意将当前通道对应公共寄存器 **CFG_SRPWM_HR_EN [cfg_srpwm_hr_en]**位应置为**'b0**，否则可能会引起波形错误抖动 (如果 **CFG_TB_PRD[cfg_tb_prd[4:0]]**的配置值大于 **0**)。

19.4.15.5 SRPWM 高精度应用约束

当启用 SRPWM 模块的高精度功能，对应用场景的配置约束如下：

1. 当只做高精度周期调整时，最小脉冲宽 (包括高电平和低电平) 为 25 ns；当同时做高精度周期调整和高精度占空比调整，或是同时做高精度周期调整和高精度死区调整，最小脉冲宽 (包括高电平和电平) 为 30ns；当同时做高精度周期调整、高精度占空比调整以及高精度死区调整，最小脉冲冲宽 (包括高电平和电平) 为 35 ns；
2. PWM 波开关周期应与 TB_TIMER 的计数周期相等；
3. 在双向计数模式下，CMPa/b/c/d 的配置值不可等于 TB 计数周期值。需要用到计数周期值作为比较值的时候可用 prd_evt 事件替代。

19.4.16 次级保护 (POSTPROTECT)

19.4.16.1 次级保护的功能

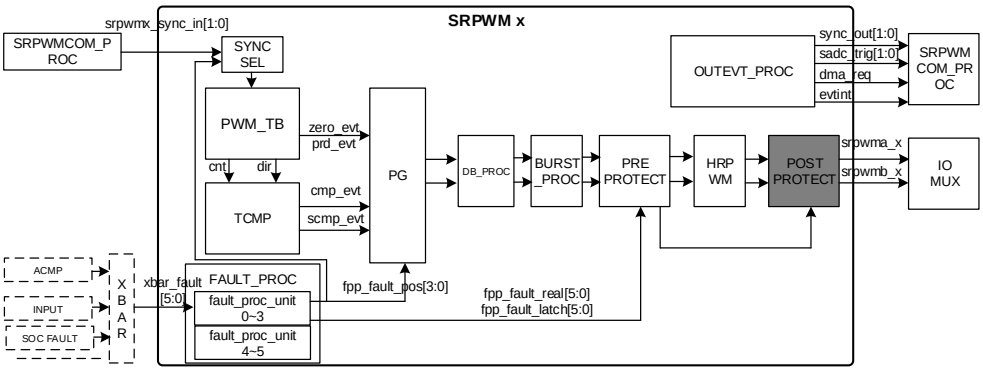


图19-35 次级保护模块在 SRPWM 通道内的位置

次级保护功能可以在不需要时钟驱动的条件下对 PWM 波信号的快速故障保护。另外次级保护功能中还可对 a、b 支路进行互斥判断，并综合通道间互斥判断结果 (来自 SRPWMCOM 模块，详见 19.4.18.5 "通道间信号互斥判断"小节的说明)，在互斥条件满足时可选择对 PWM 波信号做保护处理，将 PWM 波置于预先配置的安全状态下 (也可选择不做保护动作)。

19.4.16.2 次级保护的流程

次级保护的流程如下图：

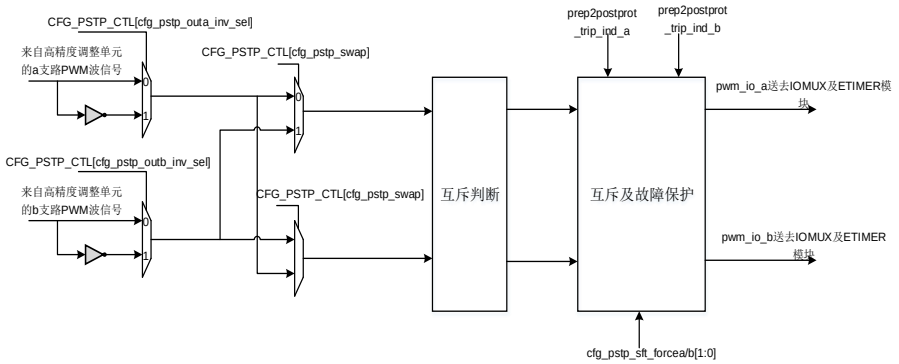


图19-36 次级保护处理流程

来自高精度调整单元的 a、b 支路 PWM 波信号在次级保护模块里可以选择是否取反，是否进行 a、b 路交换，完成上述选择后再进行 a、b 支路的互斥判断以及互斥后和故障发生后的保护动作。通过配置寄存器 **CFG_PSTP_CTL** 可以控制 a、b 支路交换是否同步进行，如果该寄存器的值配置为**'b1**，则 **CFG_PSTP_CTL[cfg_pstp_swap]**的值变化以后还需等待过零事件出现才开始响应交换状态的变化。

a、b 支路互斥成立的条件可以通过寄存器 **CFG_PSTP_CTL** 配置，是否需要进行互斥判断通过寄存器 **CFG_PSTP_CTL[cfg_pstp_exc_en]**配置。a、b 支路的互斥判断结果和通道间互斥判断的结果 (通道间互斥判断详见 19.4.18.5 "通道间信号互斥判断"小节的描述) 做或后可以作为互斥保护条件，如果互斥保护条件成立，则根据寄存器 **CFG_PSTP_CTL[cfg_pstp_exc_forcea/b]**的配置，分别将 a、b 支路置于安全状态，可选的保护动作包括：

- CFG_PSTP_CTL[cfg_pstp_exc_forcea/b]='d0：不动作；
- CFG_PSTP_CTL[cfg_pstp_exc_forcea/b]='d1：强制置高电平；
- CFG_PSTP_CTL[cfg_pstp_exc_forcea/b]='d2：强制置低电平；
- CFG_PSTP_CTL[cfg_pstp_exc_forcea/b]='d3：输出高阻；

另外，在 PREPROTECT 模块章节提到的快速封波保护动作最终是在次级保护模块中完成，PREPROTECT 模块送来的保护动作指示可将 PWM 波信号置高阻/置 0/置 1/取反。

另外，还可以通过公共控制寄存器 **CFG_PSTP_SFT_07FORCE** 和 **CFG_PSTP_SFT_811FORCE** 对各个通道进行后保护的强制配置动作，软件强制配置对应的动作为：

- 软件配置值='d0：不动作；
- 软件配置值='d1：强制置高电平；
- 软件配置值='d2：强制置低电平；
- 软件配置值='d3：输出高阻；

软件配置强制动作的优先级最高，故障事件的优先级次之，互斥判断结果的优先级最低。

次级保护动作不依赖于工作时钟进行，所有保护动作均是动作触发条件成立则立即进行保护动作。

19.4.17 通道输出事件 (OUTEVT_PROC)

19.4.17.1 输出事件的功能

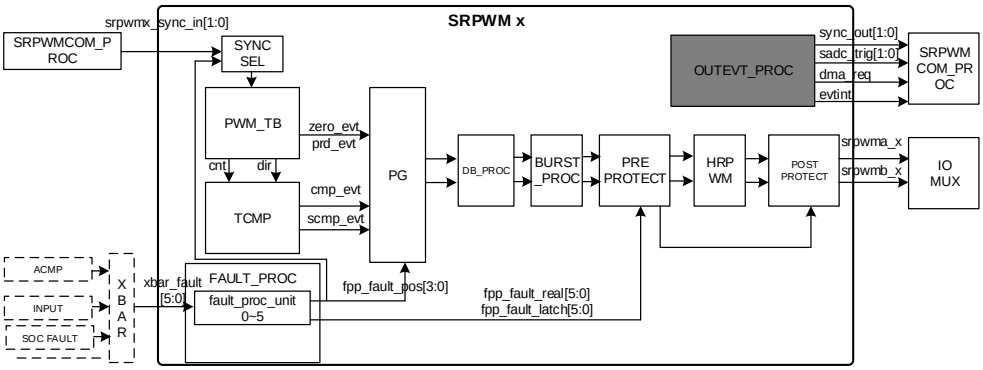


图19-37 OUTEVT_PROC 模块在 SRPWM 通道内的位置

通道内的输出事件处理包括产生通道间同步所需的同步比特 (sync_out[1:0])、ADC 模块所需触发信号 (srpwm_adc_tri[1:0])、DMA 请求信号、中断信号、DAC 同步触发信号、

ACMP 同步触发及 Blanking 触发信号。其中关于 sync_out[1:0]的产生在 19.4.9.2 "PWM 时基模块的同步"小节中已有描述，在当前章节不做赘述。

19.4.17.2 ADC 触发信号的产生

SRPWM 模块送往 ADC 的触发信号从输出事件可选同步集合 (sync_all) 以 Mask_OR 方式选择，支持分频和倍频处理，如下图所示：

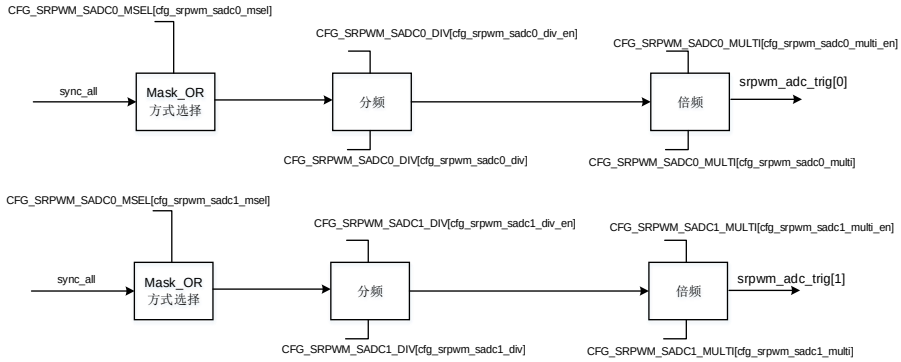


图19-38 ADC 触发信号产生流程

上图中，对输出事件可选同步集合 (sync_all) 的 Mask_OR 选择如下图：

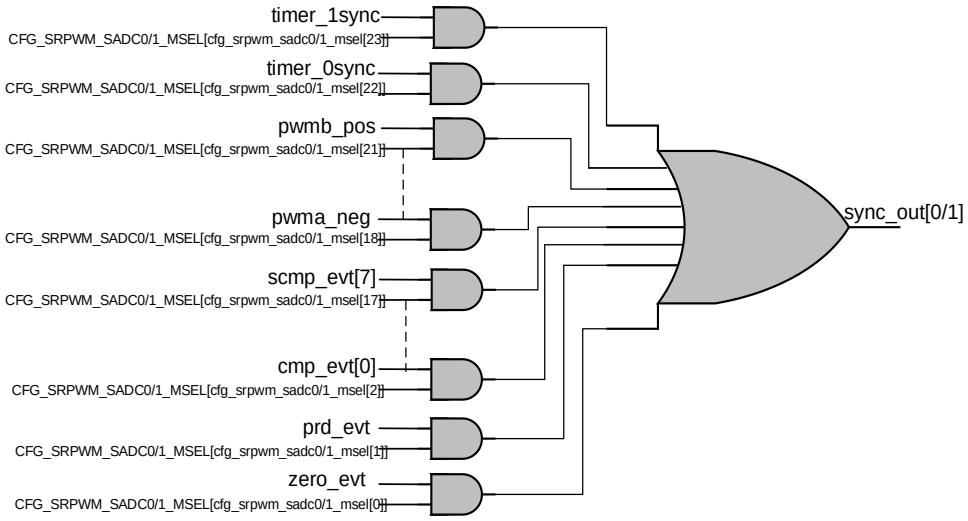


图19-39 对 sync_all 的 Mask_OR 方式选择

上图中 timer_0sync 和 timer_1sync 是时基同步信号，其产生参见图 19-12。

在图 19-38 中，分频功能实现触发脉冲信号的分频，当分频比 (寄存器 **CFG_SRPWM_SADC0/1_DIV**) 置为**'d0**，或者分频使能置为**'b0**，分频输入直通到分频输出，不做分频处理，当分频使能置为**'b1** 且分频比配置为 N (N>0)，则每输入 N+1 个脉冲才会选通 1 个脉冲 (选通 N+1 个脉冲中的最后一个)。

图 19-38 中的倍频功能实现的效果：如果开启倍频使能，且倍频倍数设置为 N（N>0），

则在相邻两次触发事件之间，每 N+1 个 PWM 工作时钟周期就会输出 1 个触发脉冲。每次到来一次触发事件，会把这个计数器重新清零后继续做倍频计数。一个对倍频功能使用的例子是：设定 PRD 事件为触发事件，可以利用此处的倍频功能在一个 PWM 周期内多次触发 ADC 采样求平均。另外，当启用倍频功能，而倍频倍数 N 值置为 0，则会输出一个常高电平信号。

12 个通道共产生 24bit 的 ADC 触发信号，这些信号需在 SRPWMCOM_PROC 模块中做汇聚处理后才最终送往 ADC 模块，详见 19.4.18.3 "ADC 触发信号的汇集及分配"的描述。

19.4.17.3 DMA 请求信号的产生

每个 SRPWM 通道支持产生一比特的 DMA 请求信号，DMA 请求信号的产生流程与 ADC 触发信号类似，如下图：

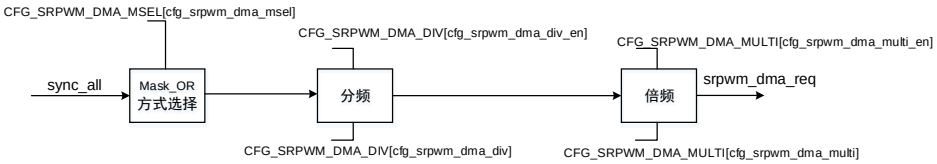


图19-40 DMA 请求信号产生流程

19.4.17.4 中断信号的产生

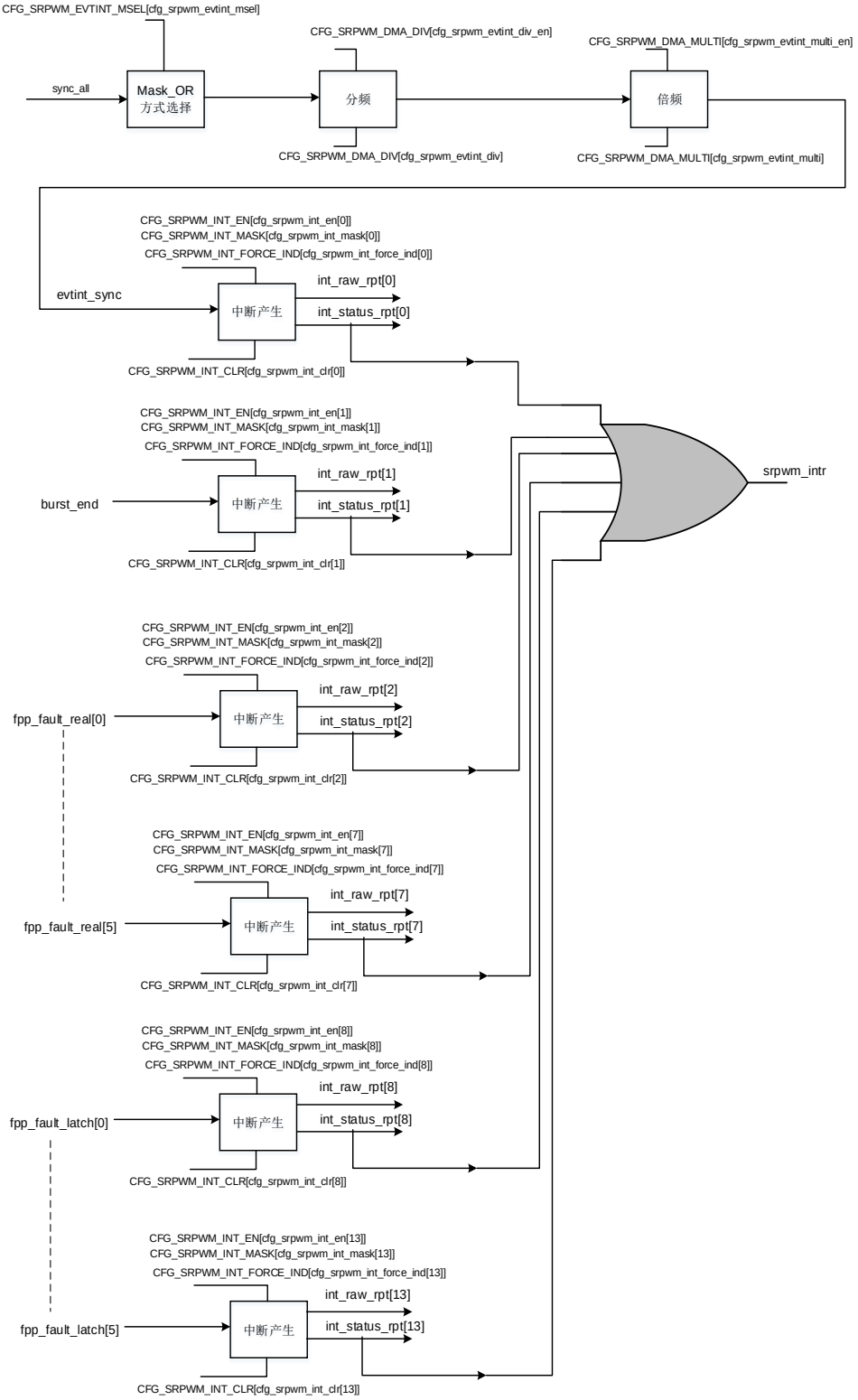


图19-41 中断信号的产生

每个 SRPWM 通道可产生 1bit 的中断信号，上图中的 srpwm_int 信号即为一个通道内产生的中断信号，12 个通道共产生 12 比特的中断信号，送往芯片中的中断处理单元。上图中 int_ram_rpt[13:0]是上报寄存器的中断原始状态（未做中断屏蔽处理前的中断状态），int_status_rpt[13:0]是上报寄存器的中断最终状态（经过中断屏蔽处理后的中断状态）。

19.4.17.5 DAC 和 ACMP 同步触发信号的产生

通过配置 **CFG_CMPC_BDAC_TRG[cfg_cmpc_sync_sel]**、**CFG_CMPC_BDAC_TRG[cfg_cmpc_blk_sel]**、**CFG_CMPC_BDAC_TRG[cfg_bdac_sync_sel]**，可从输出事件可选同步集合中分别选择一个事件作为ACMP的同步装载触发信号、ACMP中Blanking功能的触发信号、DAC的同步装载触发信号。

19.4.18 公共控制模块（SRPWCOM_PROC）

19.4.18.1 公共控制模块的主要功能

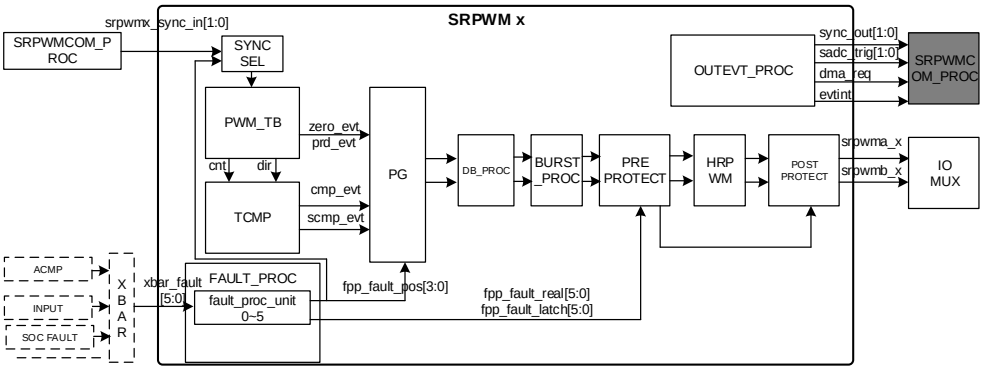


图19-42 SRPWMCOM_PROC 模块与 SRPWM 通道的关系

SRPWMCOM_PROC 模块的主要功能包括：

- 汇集 12 个 SRPWM 通道输出的 24 比特同步信号（sync_out），进行交叉矩阵互联选择，分别向每个通道输出两支路同步信号；
- 汇集 12 个 SRPWM 通道输出的 24 比特同步信号（sync_out），选择输出 4 比特同步信号到 XBAR 模块；
- 汇集 12 个 SRPWM 通道的输出的 12 比特 DAC 同步触发信号，选择输出 2 比特触发信号分别送到两个 DAC 模块；
- 汇集 12 个 SRPWM 通道的输出的 12 比特 ACMP 同步触发信号，选择输出 8 比特同步触发信号分别送到 8 个 ACMP 模块；
- 汇集 12 个 SRPWM 通道的输出的 12 比特 ACMP Blanking 触发信号，选择输出 8 比特 Blanking 触发信号分别送到 8 个 ACMP 模块；
- 将 12 个 SRPWM 通道的 12 比特同步信号 sync_out[0]送往 ETIMER 模块作为其触发信号；

- 汇集 12 个 SRPWM 通道的 24-bit ADC trigger 信号，并选择出 12 比特信号输出到 3 个 ADC 模块；
- 汇集 12 个 SRPWM 通道的 DMA 请求信号，选择输出 4 比特 DMA 请求信号到 DMA 处理逻辑；
- 通道间信号互斥判断；
- 锁存上报 12 个通道的定时计数器值；

19.4.18.2 同步触发信号在 SRPWMCOM_PROC 模块中的处理

SRPWMCOM_PROC 模块对通道间同步信号的分配及输出同步信号到 XBAR 模块详见 19.4.9.2 "PWM 时基模块的同步"的描述。SRPWMCOM_PROC 模块输出触发信号到 DAC 模块、ACMP 模块的选择详见下列 **SRPWMCOM_PROC** 寄存器的描述：

- CFG_MSRPWM_DACC_SYNC_SEL[cfg_msrpwm_dacc_sync_sel]
- CFG_MSRPWM_CMPC_SYNC_SEL[cfg_msrpwm_cmpc_sync_sel]

SRPWMCOM_PROC 模块将 12 个 SRPWM 通道的 12 比特同步信号 0 (sync_out[0]) 一一对应送往 ETIMER 模块的通道 0~通道 11 作为其触发信号 (通道 0 的 sync_out[0]送至 ETIMER0，通道 11 的 sync_out[0]送至 ETIMER11)。

19.4.18.3 ADC 触发信号的汇集及分配

如 19.4.17.2 "ADC 触发信号的产生"小节所述，每个 SRPWM 通道可输出 2 比特的 ADC 触发信号，SRPWMCOM_PROC 共汇集 12 个 SRPWM 通道共 24 比特 ADC 触发信号，根据软件配置选择输出 12 比特触发信号送往 3 个 ADC 模块，如下图所示：

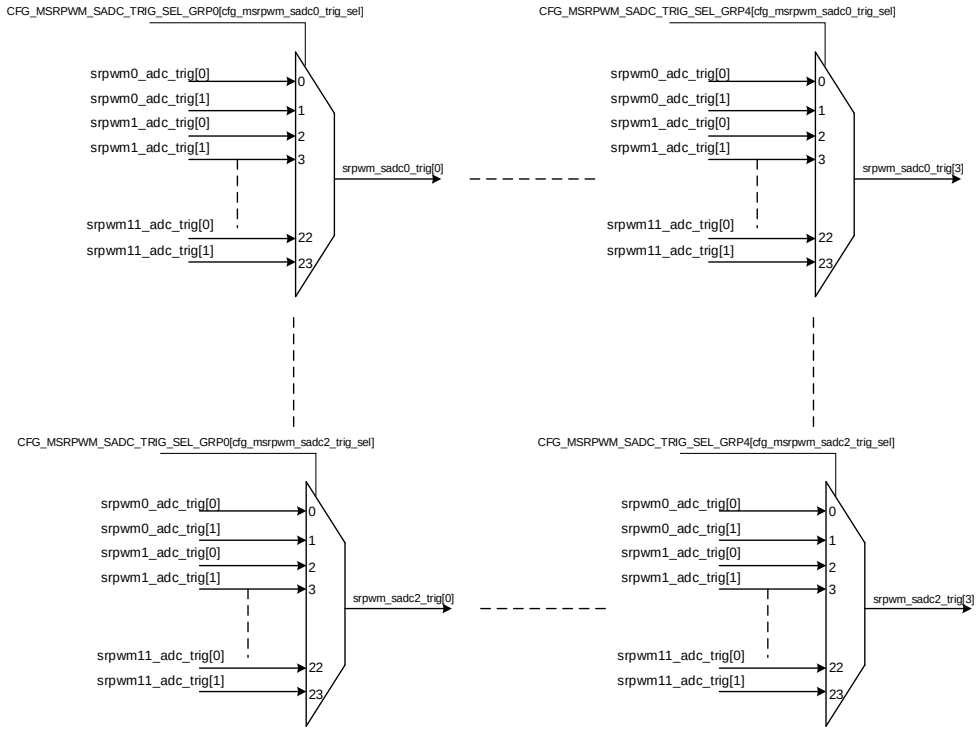


图19-43 ADC 触发信号输出选择

上图中产生的 srpwm_sadc0_trig[3:0]、srpwm_sadc1_trig[3:0]、srpwm_sadc2_trig[3:0]组合成 12 比特触发信号同时送到三个 ADC 模块作为 ADC 采样触发源及 Blanking 触发源，需注意上述三组信号在 3 个 ADC 模块中作为触发源时，其排列顺序随 ADC 模块不同而不同，详见 ADC 寄存器 **CFG_SARC_BLK_CTL0[cfg_sarc_blanking]** 和 **VC_CTL[cfg_sarc_vc_trig_sel]**的描述。

19.4.18.4 DMA 请求信号及中断信号输出

12 个 SRPWM 通道产生的 12bit 中断信号直接送出 SRPWM 模块，送往中断处理逻辑。

SRPWM_PROC 模块汇集 12 个通道产生的 12 比特 DMA 请求信号 (DMA 请求信号的产生详见 19.4.17.3 "DMA 请求信号的产生"小节)，通过公共寄存器 **CFG_MSRPWM_DMA_SEL** 的配置，最后选择输出 4 比特 DMA 请求信号，如下图所示：

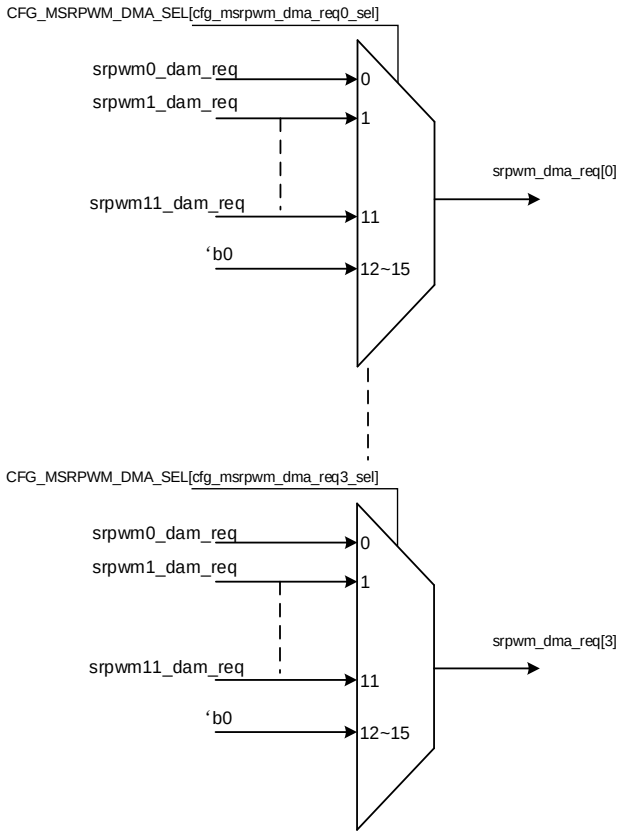


图19-44 DMA 请求输出选择

当公共寄存器 **CFG_MSRPWM_DMA_EN [cfg_msrpwm_dma_en [x]]**置为**'b0**，则第 **x** 个 DMA 输出不生效，保持低电平。

19.4.18.5 通道间信号互斥判断

SRPWMCOM_PROC 模块汇集 12 个 SRPWM 通道输出的 12 对 PWM 发波信号 (每个通道包含 a/b 支路)，根据软件配置，重新组合成 12 组信号，如下图所示：

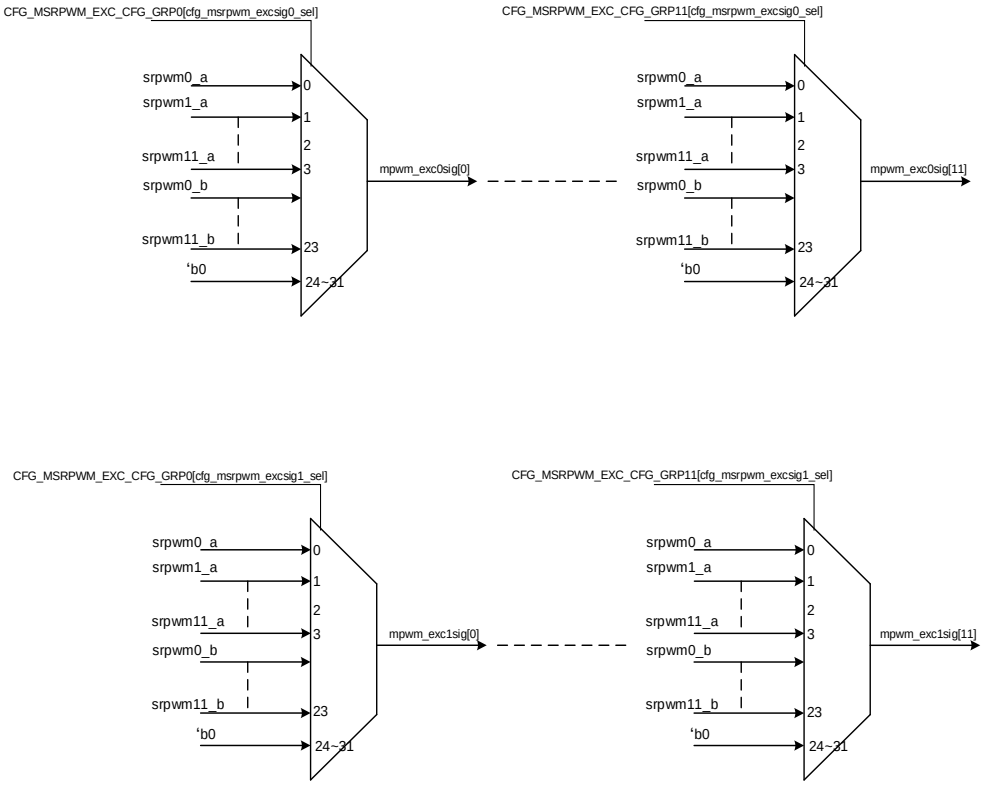


图19-45 通道间互斥判断的信号分组

根据寄存器组**CFG_MSRPWM_EXC_CFG**的配置，重新组合出12对mpwm_exc0/1_sig，每一对信号做互斥判断后可得到12个互斥结果，根据寄存器组**CFG_MSRPWM_EXC_CFG[cfg_msrpwm_exc_mask]**的配置，可选择屏蔽0~11个互斥结果，将未屏蔽的互斥判断结果做或运算，形成互斥标志。寄存器组**CFG_MSRPWM_EXC_CFG**一共包含12个寄存器，上述互斥判断处理就相应的产生12比特互斥标志分别送往12个SRPWM通道。这些通道间的互斥标志可与通道内的互斥判断组合得到每一个通道的互斥检测结果。

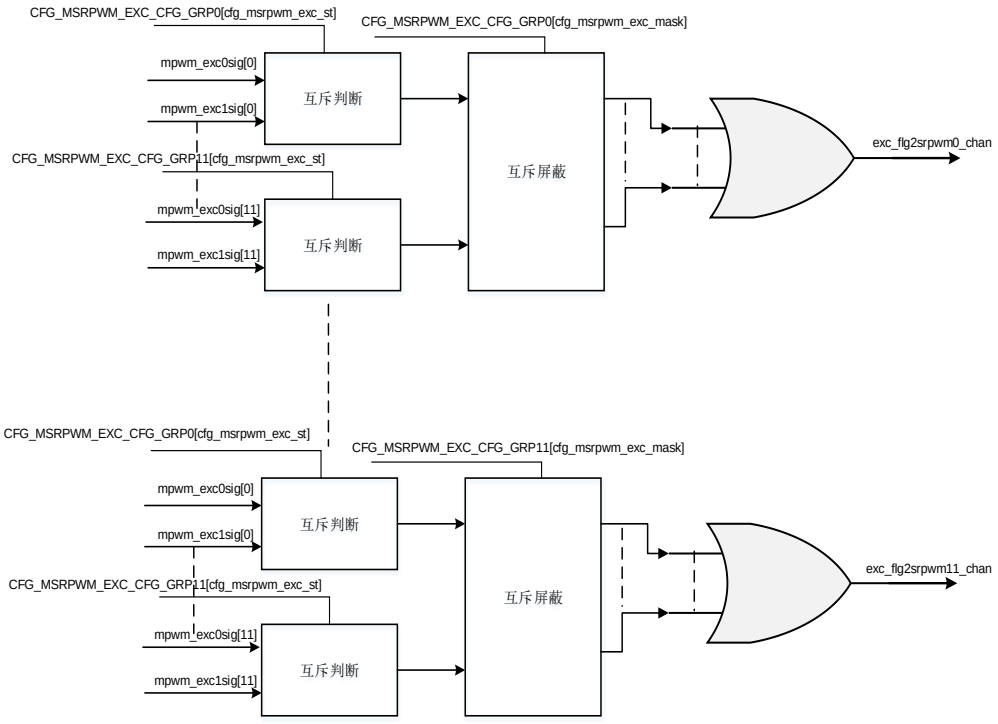


图19-46 通道间互斥判断

19.4.18.6 各通道定时计数值的锁存上报

SRPWMCOM_PROC 对各个通道定时计数值的锁存配置，详见公共寄存器 **CFG_MSRPWM_TBCNT_LCSIG_SEL** 和 **CFG_MSRPWM_TBCNT_LC_EN** 的描述。

19.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 SRPWM 和 SRPWMCOM 章节。

20 AHB DMA 控制器

20.1 简介

AHB DMA 用于在外设与存储器之间以及存储器与存储器之间提供高速数据传输，可以在无需任何 CPU 操作的情况下通过 DMA 快速移动数据。这样节省的 CPU 资源可供其它操作使用。

AHB DMA 控制器总共 2 个，每个有 8 个逻辑通道，8 对硬件握手接口，每一个 DMA 控制器都用于管理一个或多个外设的存储器访问请求。

20.2 结构框图

AHB DMA 控制器框图如下：

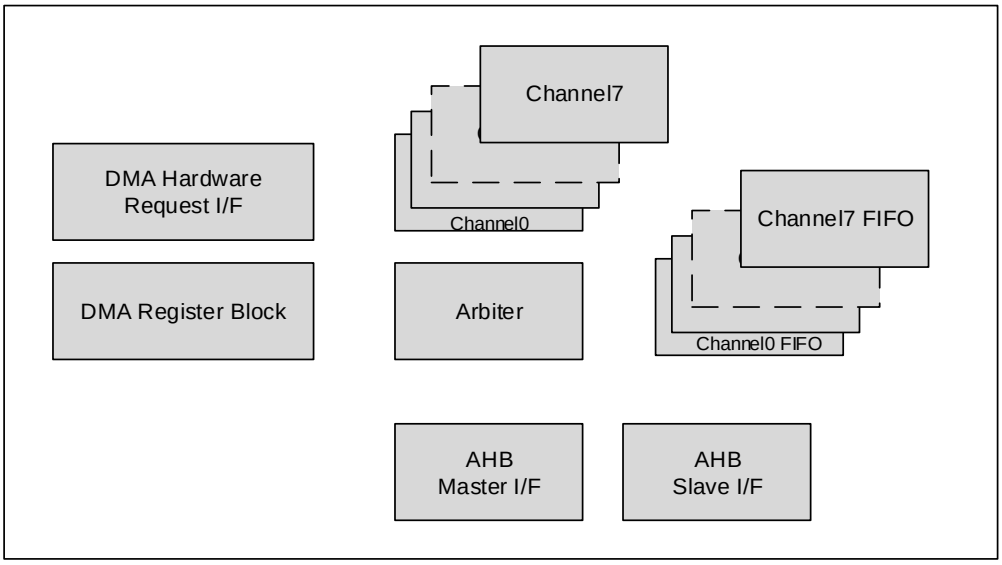


图20-1 AHB DMA 控制器框图

20.3 主要特性

AHB DMA 控制器特性如下：

- 内部含有 8 个逻辑通道，每个通道支持优先级独立配置;
- Channel1~2 FIFO 深度为 64 Bytes，其余 3~8 Channel FIFO 深度为 16 Bytes;
- DMA 包含 8 组 DMA 硬件握手信号，同时包含 single 和 burst;
- Multi-block 传输支持下列 4 种方式：
  - Auto-reloading

  - No change in the address
  - Contiguous
  - Linked list
- 只支持 DMA 作为流控的主控方；不支持外设作为主控;
- 支持内存到内存，内存到外设，外设到内存，外设到外设的数据搬运;

20.4 功能描述

AHB DMA 与外设的硬件握手关系可以通过 DMA_MUX 进行配置，具体参考 DMA_MUX 的功能描述。

20.4.1 单块传输模式配置流程

- 读取 ChEnReg 寄存器，选择未使用的通道；
- 通过写入中断清除寄存器 (ClearTfr、ClearBlock、ClearSrcTran、ClearDstTran 和 ClearErr)，从上一个 DMA 传输中清除通道上的任何挂起中断。读取中断原始状态和中断状态寄存器确认所有中断均已清除；
- 在通道的 SAR 寄存器中写入起始源地址，在通道的 DAR 寄存器中写入起始目标地址；
- 对通道的 CTL 和 CFG 寄存器进行配置；
- 配置 DmaCfgReg 寄存器为 1；
- 软件通过将 1 写入寄存器 ChEnReg 中的相应位位置来启用通道；
- 源和目标请求 Single 或 Burst DMA 事件来传输数据块，AHB DMA 在 Block 中的每笔传输 (Burst 和 Single) 完成时确认；
- 软件等待 Block 传输完成中断，也可以轮询寄存器 RawBlock 中的块传输完成指示位 (RAW 指示对应通道位为 1)；

20.4.2 基于 Linked List 的多块传输模式配置流程

- 读取 **ChEnReg** 寄存器，选择未使用的通道；
- 通过写入中断清除寄存器 (**ClearTfr**、**ClearBlock**、**ClearSrcTran**、**ClearDstTran** 和 **ClearErr**)，从上一个 DMA 传输中清除通道上的任何挂起中断。读取中断原始状态和中断状态寄存器确认所有中断均已清除；
- 对通道的 CTL 和 CFG 进行编程，设置相关 DMA 配置参数，要设置 SRC_LLP_EN 或者 DST_LLP_EN；
- 软件将第一个链表项的基址写入 LLP 寄存器中；
- 软件在系统内存中创建一个或多个链表项。软件必须按 LLP 结构，提前创建整个链表项；
- 软件通过将 1 写入 ChEnReg 寄存器中的相应位位置来使能对应的通道；

- 源和目标请求 Single 或 Burst DMA 事件来传输数据块，AHB DMA 在 Block 中的每笔传输 (Burst 和 Single) 完成时确认；
- 软件等待 DMA 传输完成中断，也可以轮询寄存器 **RawTfr** 中的传输完成指示位 (RAW 指示对应通道位为 1)；

20.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 AHB DMAC 章节。

21 模数转换器 (ADC)

21.1 简介

芯片包含 3 个 12 位的逐次逼近型模数转换器 (SAR ADC)。3 个 ADC 具有各自的控制电路，基于触发源选择，可以在同步模式下运行，也可以在非同步模式下运行。

每个 ADC 具有多达 16 个复用通道。各种不同通道的 A/D 转换可以单次、连续或者不连续采样模式下进行。ADC 的采样结果存储在多个 16 位的寄存器中。

ADC 映射到系统 AHB 总线，从而可实现快速数据处理。

每个 ADC 具有模拟看门狗功能，允许应用检测输入电压的采样值是否超过了用户自定义的阈值上限或下限。

每个 ADC 内置硬件滤波器，支持多种滤波类型，可提高模拟性能，同时还能减轻 CPU 进行相关计算的负担。

21.2 结构框图

每个 ADC 主要包含如下单元：

- 模拟采样单元
- 数字控制单元
- 采样虚拟通道单元
- 采样结果校准单元
- 采样结果预处理单元
- 采样结果滤波单元
- 采样结果缓存单元

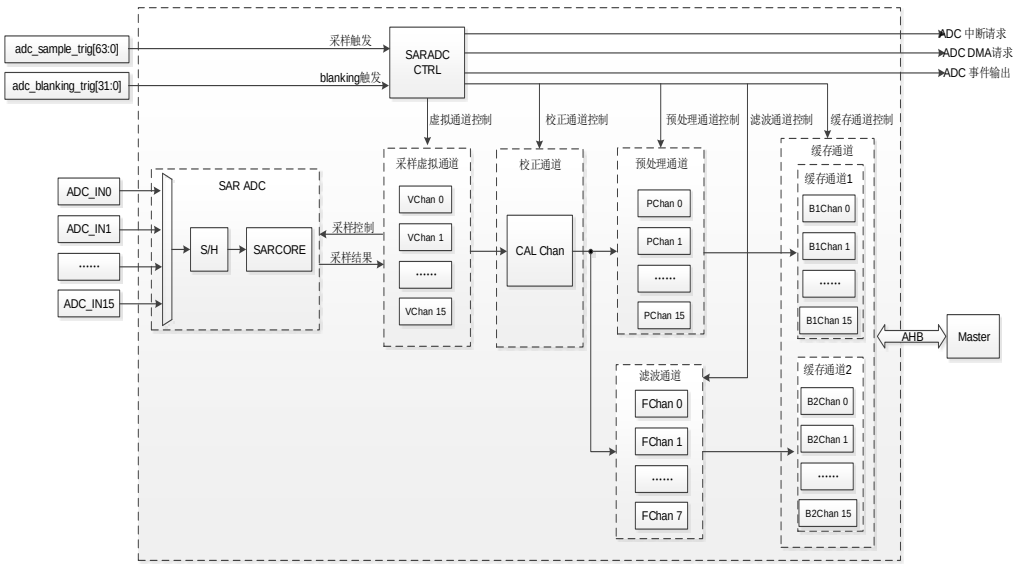


图21-1 ADC 结构框图

21.3 主要特性

主要特性如下：

- 高性能特性
  - 支持多达 3 个独立 ADC；
  - 支持 12 位分辨率；
  - 每个 ADC 支持最高采样率 4.1 MSPS；
  - 每个 ADC 支持 16 个采样通道；
  - 每个 ADC 支持独立设置各通道的采样时间；
  - 支持采样通道优先级灵活配置；
  - 每个 ADC 支持 4 路 DMA 搬运数据；
  - 每个路 ADC 支持多种中断源产生中断请求；
  - 每个 ADC 支持独立的 Power-Gating 功能，以便节省功耗；
  - 每个 ADC 独立挂接到 AHB 总线，便于采样结果高效读取；
- ADC 校准
  - 每个 ADC 支持软件启动校准功能，实现 ADC 采样校准；
- 采样转换
  - 每个 ADC 支持 16 个虚拟采样通道 VC，每个虚拟通道独立控制一次采样；
  - 16 个虚拟采样通道 VC 支持根据优先级进行调度；
  - 每个 ADC 支持基于 VC 调度的顺序采样、单次采样、过采样；
  - 每个 ADC 支持多种采样触发源；
  - 支持多路 ADC 基于触发源选择进行同步采样，同一时刻采样不同的电压信号；
  - 支持多路 ADC 基于触发源选择进行冗余采样，同一时刻采样相同的电压信号；

- 每个 ADC 支持 1 个 Blanking 事件功能，Blanking 有效窗口内，只允许 VC0 触发正常响应。若 Blanking 有效窗口内出现其他 VC 触发，等窗口结束后按优先级进行响应；
- 每个 ADC 支持多种 Blanking 触发源，Blanking 生效时刻及有效窗口长度可配置；
- 每个 ADC 支持多个 VC 同时触发时响应方式可选择，低优先级 VC 丢弃或者排队响应；
- 每个 ADC 采样触发支持单次触发和连续触发 2 种触发模式；
- 每个 ADC 支持基于虚拟通道上报 Trigger-to-Sample 的延迟时间上报；
- 每个 ADC 支持基于虚拟通道产生 EOC (End-Of-Conversion) 事件；

- 采样结果预处理
  - 每个 ADC 支持 16 个预处理通道 PC，PC 与 VC 一一对应；
  - 每个 ADC 支持 16 组增益补偿，与 PC 一一对应；
  - 每个 ADC 支持 16 组偏置补偿，与 PC 一一对应；
  - 每个 ADC 支持 16 个模拟看门狗，与 PC 一一对应，支持模拟电压上下门限检测；
  - 每个模拟看门狗支持检测结果告警事件通过配置选择输出；
- 采样结果滤波
  - 每个 ADC 支持 8 个滤波通道 FC，16 个 VC 可通过软件配置选择 FC 进行相应滤波处理；
  - 每个滤波通道基于软件配置可独立支持 IIR 和非滑动平均 2 种滤波类型；
- 采样结果缓存
  - 每个 ADC 支持 2 组缓存通道 BC1 和 BC2，每组缓存通道 16 个结果寄存器，与 VC 一一对应；
  - 支持通过 FIFO 模式进行采样结果读取；

21.4 功能描述

21.4.1 ADC 引脚和内部信号

表21-1 ADC 内部输入/输出信号

内部信号名称	信号类型	说明
adc_hclk	输入	ADC 总线时钟
adc_clk	输入	ADC 采样时钟
adc_sample_trig[63:0]	输入	ADC 采样触发源
adc_blanking_trig[31:0]	输入	Blanking 事件功能触发源
adc_int	输出	ADC 中断请求输出
adc_dma_req[3:0]	输出	4 个 DMA 通道的 req 请求

内部信号名称	信号类型	说明
adc_dma_single[3:0]	输出	4 个 DMA 通道的 single 请求
adc_dma_ack[3:0]	输入	4 个 DMA 通道的 ACK 应答
adc_eoc_evt[1:0]	输出	16 个 VC 的 EOC 事件，配置选择 2 个事件输出
adc_etim_awdt _evt[3:0]	输出	16 个 VC 的模拟看门狗告警事件，配置选择 4 个事件输出到 ETIMER
adc_xbar_awdt _evt[3:0]	输出	16 个 VC 的模拟看门狗告警事件，配置选择 4 个事件输出到 XBAR

表21-2 ADC 输入/输出引脚

引脚名称	信号类型	说明
SAR_ADC_AIN[15:0]	外部模拟信号输入	每个 ADC 多达 16 个模拟输入通道
VREFHI	参考基准输入	ADC 参考基准电压输入
VERFLO	参考基准输入	ADC 参考基准电压输入
VDD	电源	1.1 V 电源
VSS	地	1.1 V 地
VDDA	模拟电源	模拟 3.3 V 电源
VSSA	模拟地	模拟 3.3 V 地

21.4.2 时钟

ADC 采用双时钟架构，ADC 时钟与 AHB 时钟同步异频，且保持整数倍关系。

3 个 ADC 的输入时钟相同。adc_hclk 时钟与系统 AHB 总线时钟同频同步。adc_clk 最高支持频率为 66.67 MHz。

通常情况下，ADC 时钟与 AHB 时钟之间需要保持一定的比例：$F_{adc\_hclk}>=2*F_{adc\_clk}$

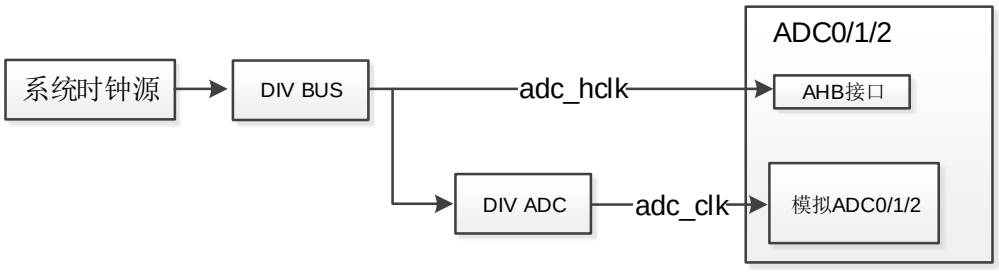


图21-2 ADC 时钟方案

21.4.3 ADC 接口时序

21.4.3.1 采样时间

单次 ADC 采样转换中，最小采样窗口长度为 2.5UI，单次转换窗口长度固定为 13UI，ADC 采样结果输出打拍占用 0.5UI，故单次 ADC 采样转换最小时间窗口为 16UI。

这样 ADC 在 50 MHz 工作时钟下的最高采样率为 3.125 MSPS，在 66.67 MHz 工作时钟

下的最高采样率为 4.125 MSPS。ADC 工作时钟最高频率为 66.67 MHz。

用户可通过配置寄存器 **CFG_SRAC_VC_CTL.VC_CTL.cfg_sarc_vc_spltime** 进行扩展采样，每个 VC 通道扩展采样参数可独立配置，扩展采样参数支持的配置范围为 0~256 个 ADCCLK 时钟周期。在 ADC 工作时钟频率不变的情况下，扩展采样参数配置越大，ADC 采样率越低。

21.4.3.2 接口时序

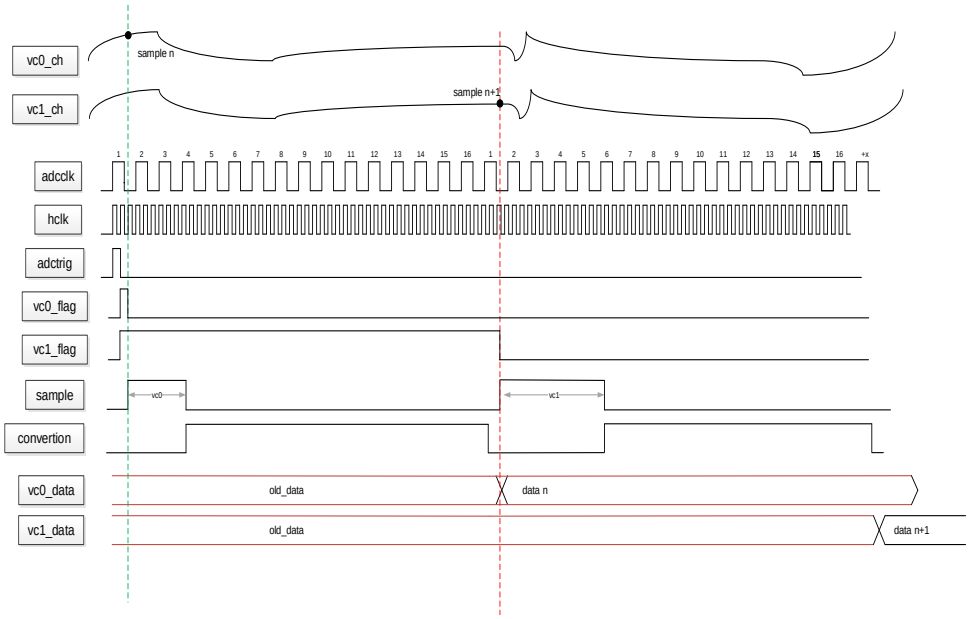


图21-3 ADC 接口时序示意

21.4.4 ADC 采样虚拟通道

每个 ADC 包含 16 个采样虚拟通道 VC，每个 VC 可独立控制 ADC 采样。16 个 VC 独立配置，配置域如下：

- VC 使能配置 (CFG_SARC_VCEN_CTL. cfg_sarc_vc_en)
- VC 优先级配置 (VC_CTL. cfg_sarc_vc_priority)
- VC 采样结果滤波功能配置
- VC 采样触发模式及触发源配置
- VC 采样通道类型及通道选择配置
- VC 采样时间长度配置

21.4.4.1 VC 优先级

每个 ADC 的 16 个 VC 可配置采样优先级，共 4 组绝对优先级 0/1/2/3，优先级序号越小优先级越高。在同一优先级组内，VC 编号越小优先级越高。

VC 优先级主要处理如下事件：

- 当多个 VC 同时被有效触发时，根据寄存器 **CFG_SARC_ADC_CTL0.**

cfg_sarc_adc_event_conflict 配置选择以下 2 种处理方式之一：

- 低优先级的 VC 都被丢弃，只响应优先级最高的 1 个 VC 进行采样；
- 所有被触发的 VC 均按各自的优先级进行排队采样；

- 根据 VC 的优先级配置进行 ADC 采样顺序调度；

21.4.4.2 VC 采样结果滤波选择

每个 VC 可独立配置选择是否对该 VC 的采样结果进行滤波处理，以及选择相应的滤波通道。

21.4.4.3 VC 触发

16 个 VC 触发源可独立配置选择，触发源来自 IPTest、SRPWM、ETIMER、STIMER、XBAR、ADC、ACMP 等模块。

3 个 ADC 的采样触发源如下：

Sample 触发源	位域	SARC0/1/2
SRPWM	[11:0]	{epwm_sadc2_trig[3:0], epwm_sadc1_trig[3:0], epwm_sadc0_trig[3:0]}
ETIMER	[25:12]	etim2adc _evt[13:0]
ACMP	[33:26]	{cmpc_ctripl[3], cmpc_ctriph[3], cmpc_ctripl[2], cmpc_ctriph[2], cmpc_ctripl[1], cmpc_ctriph[1], cmpc_ctripl[0], cmpc_ctriph[0]}
STIMER	[41:34]	{stm1_oc_exp[3:0], stm0_oc_exp[3:0]}
XBAR	[45:42]	xbar2sarc_cludata[3:0]
ADC	[51:46]	{sarc2_eoc2spl_trig [1:0], sarc1_eoc2spl_trig [1:0], sarc0_eoc2spl_trig[1:0]}
INPUTXBAR	52	inputxbar_data[4]
IPTest	53	iptest_adc_start
RESERVED	[63:54]	10'd0

16 个 VC 可独立配置触发模式：One-shot 模式和 Continuous 模式。

- One-shot 模式下，VC 使能一次只允许该 VC 响应一次触发采样，之后需要软件关闭该 VC 使能，重新配置相应参数后打开使能，该 VC 才能响应下一次触发采样。即一次配置只响应一次触发采样。
- Continuous 模式下，只要 VC 的使能处于打开状态，则可以一直响应触发采样。即一次配置可响应多次触发采样。

21.4.4.4 软件直接触发采样

在 ADC 控制器使能打开的前提下，无论虚拟 VC 通道是否配置了相应的采样触发源，且无论相应的采样触发源是否产生，软件通过对相应寄存器置位，可以触发启动对应虚拟通道进行采样转换。软件向配置寄存器 **CFG_SARC_VC_FORCE** 写 **1** 来实现启动采样转换，该寄存器自清零 (软件写 1 产生脉冲信号)。

寄存器 **CFG_SARC_VC_FORCE** 有 16 个有效 bit 位，分别对应 16 个虚拟 VC 通道，指示对应虚拟 VC 通道通过软件启动转换开始标志。该寄存器相应 bit 位写 **1** 会强制将寄存器 **CFG_SARC_VC_FLAG** 的对应位置 **1**，用于软件控制启动转换。该位写 **0** 无效，该位软件读操作返回 0。

在同一时钟 cycle，如果软件对寄存器 CFG_SARC_VC_FORCE 某位置 **1**，同时硬件对寄存器 **CFG_SARC_VC_FLAG** 的相同位置 **0**，则软件置位的优先级高，即寄存器 **CFG_SARC_VC_FLAG** 的对应位响应软件置 **1**，此时寄存器 **CFG_SARC_VC_OVF** (虚拟通道启动转换溢出标志) 的对应 bit 位不受影响。

例如软件配置寄存器 **CFG_SARC_VC_FORCE** 为 0x000F，在 ADC 控制器使能打开的前提下，则寄存器 **CFG_SARC_VC_FLAG** 中 VC0、VC1、VC2、VC3 对应位置 1，即该 4 个虚拟通道被软件强制触发，然后根据对应优先级进行排队转换。

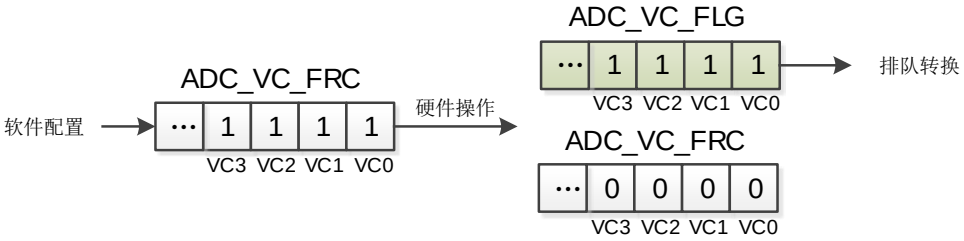


图21-4 软件直接触发采样示意

21.4.4.5 采样通道

每个 ADC 基于 VC 配置寄存器 **VC_CTL.cfg_sarc_vc_se_mux** 可选择相应的模拟采样通道 (ADC_IN0~ADC_IN15)，每个 VC 都可以独立配置选择相应的模拟采样通道。

21.4.4.6 VC 采样时间长度

通过配置 ADC 寄存器 **VC_CTL.cfg_sarc_vc_spltime**，可选择是否增加 ADC 采样时间长度，以及增加的长度值。每个 ADC 的 16 个 VC 可独立配置该功能。该配置域位宽 9bit，支持 0-256 个 ADC 时钟周期数配置范围。

ADC 一次采样转换的最小时间为 16 个 ADC 时钟周期，如果寄存器 **VC_CTL.cfg_sarc_vc_spltime** 配置值为 3，则此时 ADC 一次采样转换时间为 19 个 ADC 时钟周期。

21.4.4.7 Blanking 事件功能

每个 ADC 包含一个 Blanking 管理模块，支持一个 Blanking 事件，可以对采样触发源进行延迟 Blanking 操作，Blanking 功能可屏蔽。每个 ADC 中仅 VC0 支持 Blanking 管理，Blanking 功能由 Blanking 触发源触发启动。

Blanking 支持 3 类触发源：PITIMER，ETIMER，SRPWM。

3 个 ADC 的 Blanking 触发源如下：

Blanking 触发源	位域	SARC0/1/2
SRPWM	[11:0]	{epwm_sadc2_trig[3:0], epwm_sadc1_trig[3:0],

Blanking 触发源	位域	SARC0/1/2
		epwm_sadc0_trig[3:0]}
ETIMER	[25:12]	etim2adc _evt[13:0]
RESERVED	[31:26]	6'd0

Blanking 支持延迟触发，延迟触发时间可配置，且对所有触发源统一配置，延迟单位为 AHB 时钟周期。

Blanking 有效窗口长度可配置，长度单位为 AHB 时钟周期。有效 Blanking 窗口内 VC0 的触发 (专用周期触发源) 可以正常响应，窗口内若出现其他触发信号，待 Blanking 窗口结束后，按优先级排队处理。

若 Blanking 窗口未结束，其他触发信号 (非专用周期触发) 出现重复触发，上报告警，并忽略该重复触发。

Blanking 机制的基本原理如下图：

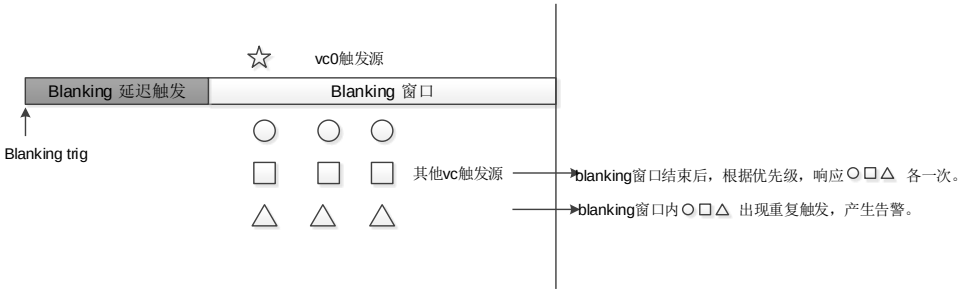


图21-5 Blanking 原理示意

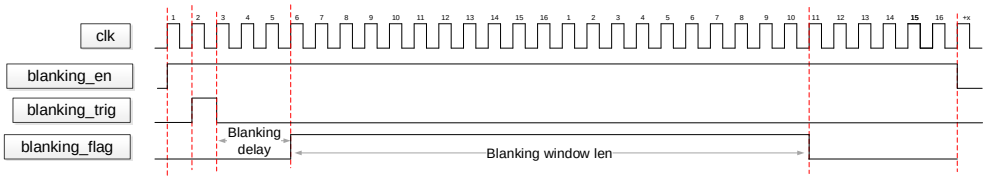


图21-6 Blanking 时序

21.4.4.8 ADC 同步模式

同步采样：多个 ADC Core 同时采样不同信号源。

冗余采样：多个 ADC Core 同时采样相同信号源。

由于 3 个 ADC 的触发源选择独立，故在实现 ADC 同步模式时，需要软件保证将参与同步模式的 ADC 配置为相同的采样触发信号源。

ADC 同步模式下，软件在启动同步采样转换时，要确保当前参与同步采样的 ADC Core 都处于 Idle 状态。ADC 同步模式下，ADC Core 的采样保持时间、触发模式 (单次或连续) 需要保持一致。

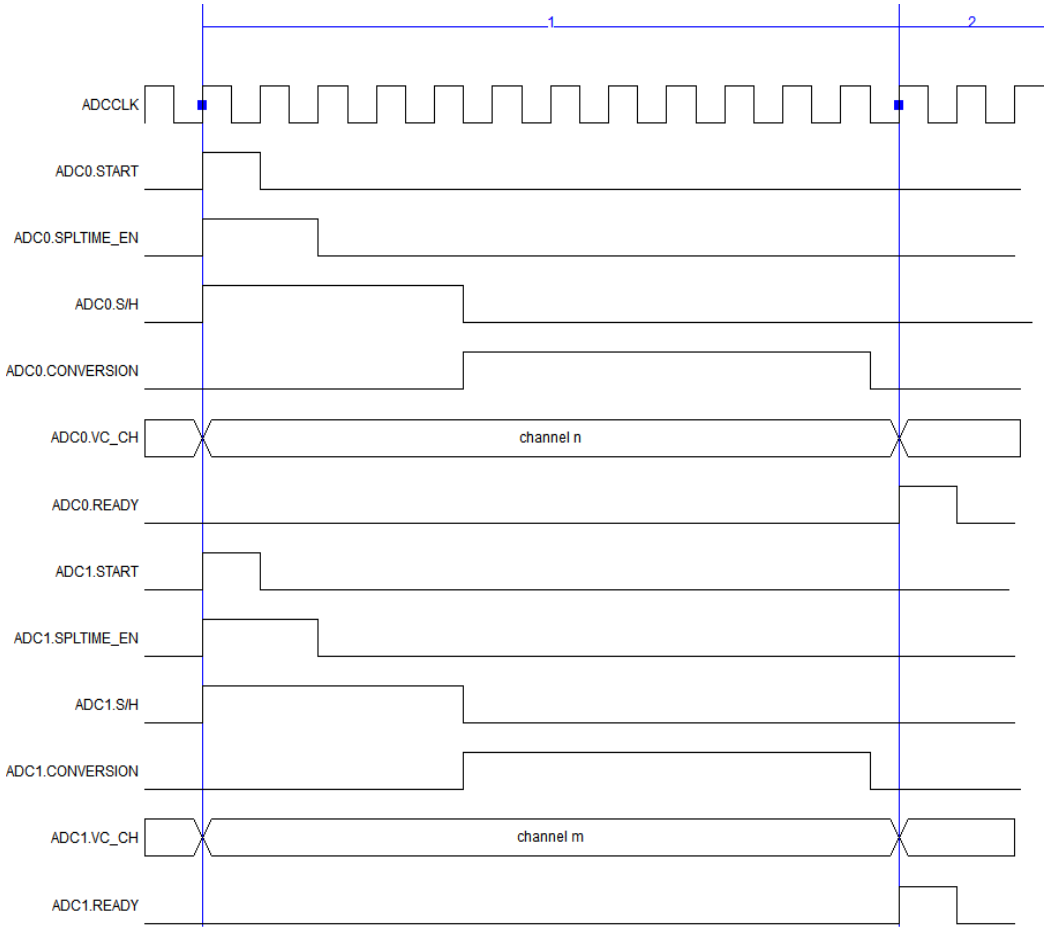


图21-7 同步采样：多个 ADC Core 同时采样不同信号

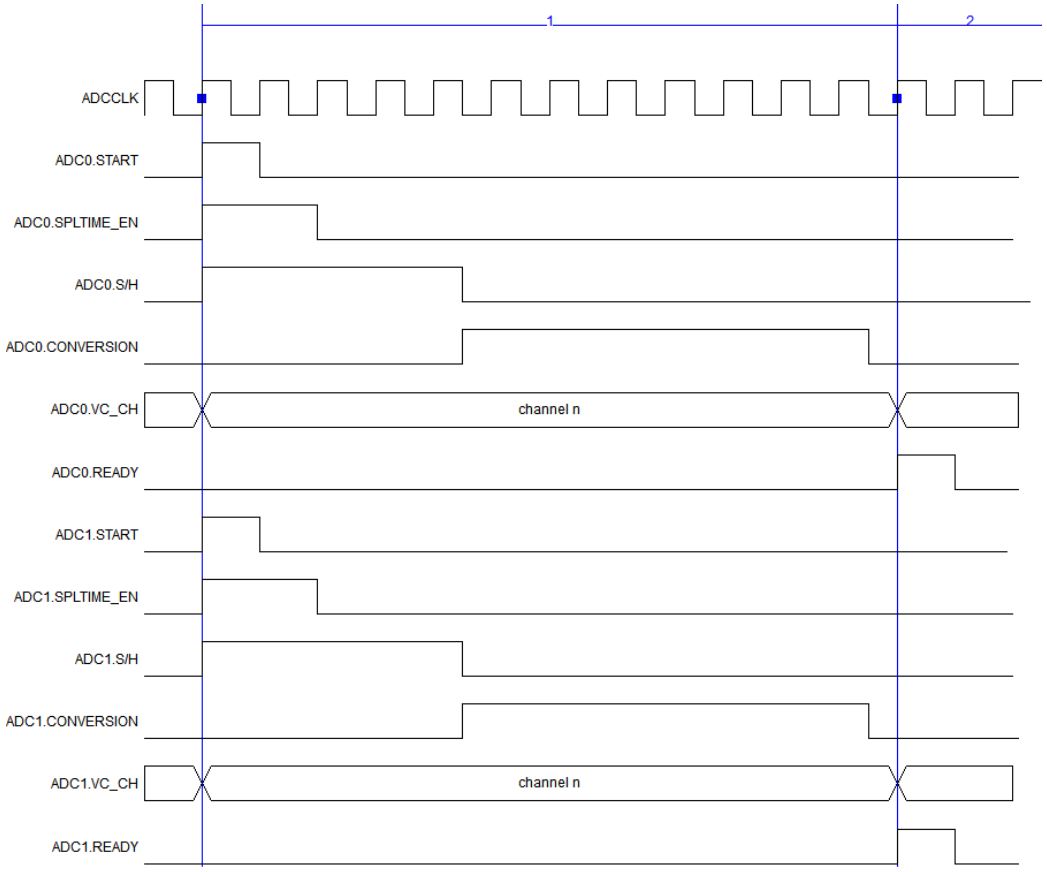


图21-8 冗余采样：多个 ADC Core 同时采样相同信号

21.4.5 采样延时时间捕获

ADC 控制器支持基于 VC 通道进行 Trigger-to-Sample 延迟计算，即有效触发时刻到采样开始时刻的延迟时间，并上报该延迟时间 (HCLK 周期计数值)。

每个 VC 通道独立上报该延迟时间值。软件可通过状态寄存器 **T2S_STAMP_REQ/DLY** 进行相应时间值的读取。

ADC 控制器包含一个基于 HCLK 的 12bit free-run 内部计数器，用于 Trigger-to-Sample 延迟计算，最大可计 4096 个 HCLK 时钟周期，该计数器的值软件可通过寄存器 **T2S_FR_CNT** 进行实时读取。

延迟时间包括两个值：

REQ-STAMP：有效采样触发信号到达时刻的时间戳。在触发信号到达时刻锁存内部 free-run 计数器的值。

DLY-STAMP：Trigger-to-Sample 的延时时间值。使用采样开始时刻 (S/H 窗口上升沿) 内部 free-run 计数器的值，减去采样触发信号到达时刻的时间戳 REQ-STAMP。

说明

内部 free-run 计数器的有效计数量程为 4096，当 Trigger-to-Sample 的延时时间超过 4096 时，该 DLY-STAMP 上报值可能是错误值，应用时要避免出现该情况。

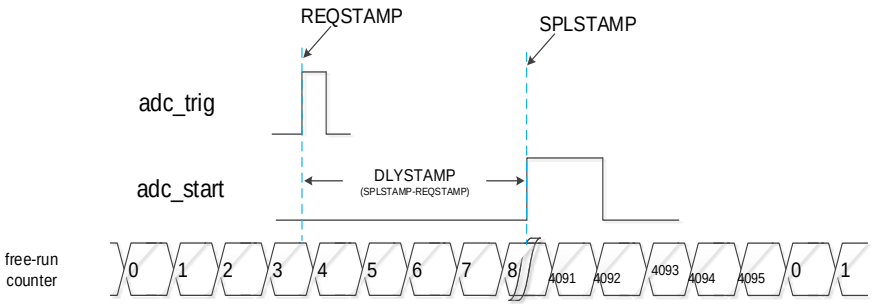


图21-9 Trigger-to-Sample 延时计算示意

21.4.6 EOC

每个 VC 通道都包含一个对应的 EOC (End-of-Conversion) 信号，该 EOC 信号可用于产生中断、DMA 请求、ADC 采样触发。

EOC 脉冲信号可通过配置 **EOC_CTL.cfg_eoc_mod** 寄存器从如下两个位置进行选择，所有 VC 通道统一配置，默认选择 S/H 窗口结束时刻：

- S/H 窗口结束时刻
- 转换结束时刻

EOC 信号选择时刻到中断产生的上沿延时可配置，16bit HCLK 时钟计数值；且所有 VC 通道统一配置。如果当前 EOC 脉冲信号到来时，上一个 EOC 脉冲信号的延时处理没有完成，则在该时刻输出上一个 EOC 的脉冲信号，同时当前 EOC 脉冲信号进入延时模块进行延时处理。

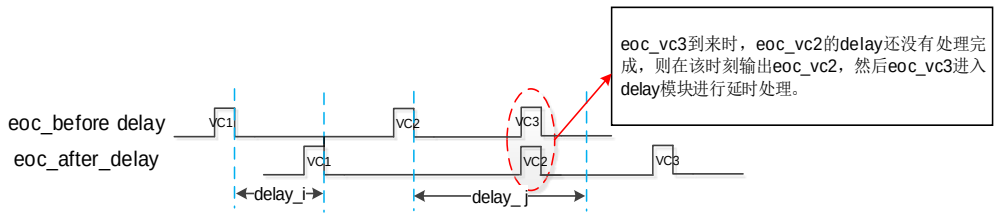


图21-10 EOC 延时处理示意

16 个 VC 通道对应产生 16 个 EOC 脉冲信号，该 16 个 EOC 脉冲信号全部可作为中断源或者 DMA 请求源。其中，通过配置 **EOC_CTL.cfg_eoc2spl_trig0/1_sel** 寄存器从 16 个 EOC 脉冲信号中选择 2 个作为 ADC 的采样触发信号源。

当 EOC 的延时处理出现冲突时，即当前 EOC 脉冲信号到来时，上一个 EOC 脉冲信号的延时处理没有完成，将上报延时冲突告警，软件可通过状态寄存器 EOC_STS 查询冲突状态，上报冲突告警时同时上报延时未处理完成的 VC 通道编号。延时冲突告警状态通过软件读操作进行清除。

21.4.7 ADC 采样结果处理

采样结果从模拟 ADC 输出 (12bit 无符号数)，先将 12bit 无符号数转成 12bit 有符号数

s(12,0)，然后将整数和小数位各扩展 2bit 得到 s(16,2)，以此数据形式经过数字校准 dig_cal、预处理 pre_process 和滤波处理 filter，最终得到 s(16,2)有符号数，此数据通过软件配置进行定点格式转换，然后将转换后的数据存入对应的缓存通道。

由于采样控制是基于 VC，所以采样结果处理的每个阶段都应该根据相应的 VC 编号及配置参数进行。

如果相应处理阶段 bypass，则结果数据在经过该阶段时直接输出，不作处理。

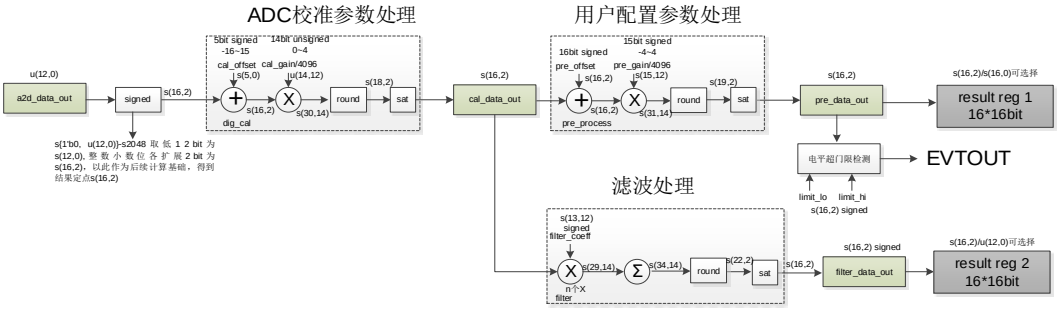


图21-11 ADC 采样结果处理流程

21.4.8 ADC 校准通道

在很多应用中，由于非理想因素存在，会引入增益误差 (gain)、直流失配 (os)。

每个 ADC 均提供校准功能，包含模拟域直流失配补偿 (**CFG_SARC_CAL_CTL.cfg_sarc_d2a_adc_os_se_tune**)，数字域增益误差补偿 (**CFG_SARC_CAL_GAIN.cfg_sarc_cal_se_gain**) 和直流失配补偿 (**CFG_SARC_CAL_OS.cfg_sarc_cal_se_os**)。

校准补偿参数默认出厂设置，也支持用户配置修改。在 ADC 采样结果处理的校准参数处理阶段，硬件会自动将采样结果与相应校准参数进行加/减/乘运算操作，然后将运算后的结果进行定点化处理后送入后级处理模块。

基于 SAR ADC 不同摆幅，数字域校准补偿参数存在两个配置值，需由客户根据单板及使用场景，确定具体使用哪一个 (从 Flash 中加载到上述相应的补偿寄存器中)。

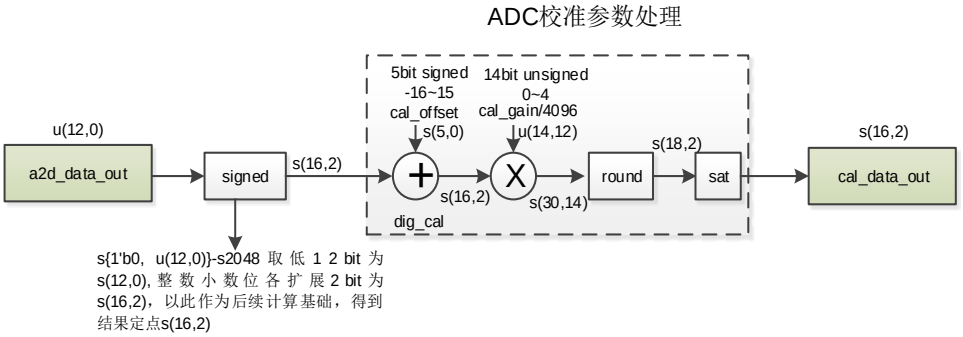


图21-12 ADC 数字域校准框图

其中，Offset 的加法器和 Gain 的乘法器均可以 Bypass。Bypass 后，校准通道的输出等于输入。

21.4.9 预处理通道

每个 ADC 包含 16 个预处理通道 PC，16 个 PC 与 16 个 VC 一一对应。每个 VC 的采样结果均需要进行其对应的 PC 进行处理。

21.4.9.1 预处理 Offset/Gain

每个 PC 可独立配置预处理 Offset/Gain 参数 (该预处理 Offset/Gain 是用户需要，与校准 Offset/Gain 没有关系)，Offset 加法器和 Gain 乘法器均可独立 Bypass。采样结果预处理流程如下图：

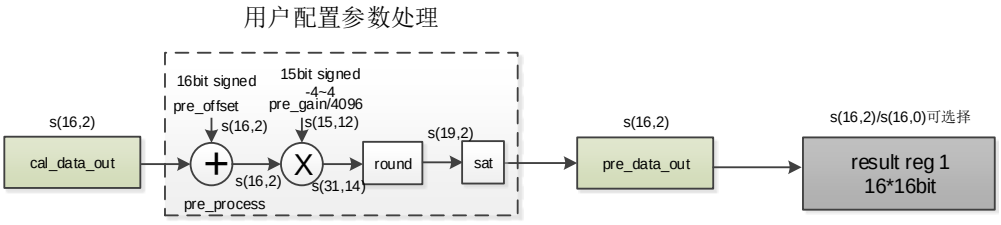


图21-13 采样结果预处理流程

21.4.9.2 模拟看门狗

16 个 PC 支持独立的模拟看门狗功能。模拟看门狗可以检测每个 VC 的采样结果是否保持在用户配置的电压上下门限范围内。超门限检测的对象为用户预处理补偿后的采样结果。

检测包括超上门限和超下门限检测，每个用户预处理通道的采样结果独立配置上下门限值、独立检测。

21.4.9.2.1 上下门限检测

每个 PC 可独立配置上下门限阈值，且上门限和下门限独立检测。

当某个 VC 的采样结果经过对应预处理 Offset/Gain 处理后，进入看门狗检测。

当采样结果大于上门限阈值时，输出采样结果上溢告警。

当采样结果小于下门限阈值时，输出采样结果下溢告警。

上下门限检测输出使用 CBC 或者 One-shot 两种模式 (软件配置选择)。

CBC 模式下，根据每个新的采样结果是否超门限控制是否输出告警，同时软件也可以进行清除；

One-shot 模式下，一旦产生了超门限告警，需要软件进行清除。

每个超门限检测通道输出 1bit 事件，上下门限告警通过 MUX-OR 的方式输出。同时单独上报超上下门限的实时状态。

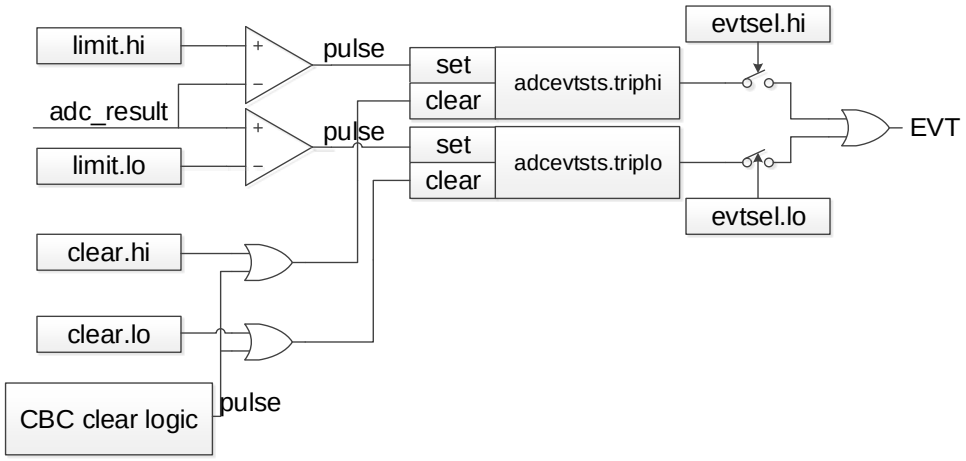


图21-14 单通道超门限检测原理框图

CBC 和 One-shot 两种模式下，超门限检测检测告警产生示意图如下：

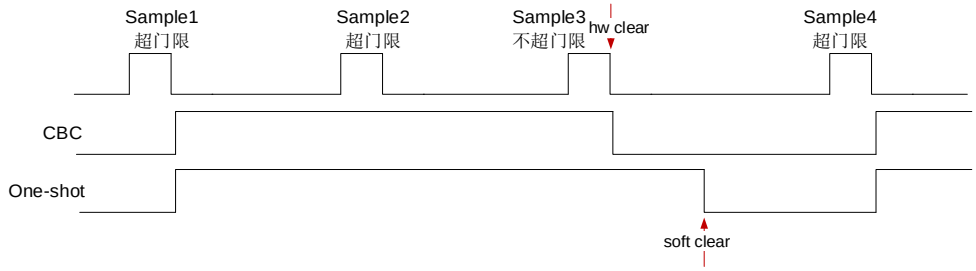


图21-15 单通道超门限结果

21.4.9.2.2 检测结果

- **检测结果上报状态寄存器**

  16 个 PC 通道的超门限检测结果，独立上报状态寄存器。16bit 的超上门限检测结果 **WDT_STS.wdt_hi_sts[15:0]** 和 16bit 超下门限检测结果 **WDT_STS.wdt_lo_sts[15:0]**。***_sts[15:0]**分别表示 PC 通道 15~PC 通道 0 的超门限检测结果。

- **检测结果产生中断源**

  16 个 PC 通道的超门限检测结果，独立产生中断源，一共 32 个中断源。超上门限中断源 16bit 和超下门限中断源 16bit。

- **检测结果产生 DMA 请求源**

  16 个 PC 通道的超门限检测结果，按通道产生 DMA 请求源，一共 16 个 DMA 请求源，每个 PC 通道一个请求源。单个通道的超上门限结果和超下门限结果相或后产生该通道的 DMA 请求源 (**CFG_WDT_EVT_EN** 寄存器配置对应 PC 通道上下门限检测告警使能配置为有效且有对应的检测告警产生，才会生成相应的 DAM 请求)。

- **检测结果输出告警事件**

  16 个 PC 通道的超门限检测结果，一共产生 16 个告警事件，每个 PC 通道一个告

警事件。单个通道的超上门限结果和超下门限结果相或后产生该通道的告警事件(**CFG_WDT_EVT_EN** 寄存器配置对应 PC 通道上下门限检测告警使能配置为有效且有对应的检测告警产生，才会生成相应的告警事件)。

通过 **CFG_WDT_EVT_SEL** 寄存器可从 16 个 PC 的告警事件中，独立配置选择 4 个事件输出到 XBAR 模块和 ETIMER 模块。

21.4.10 滤波通道

每个 ADC 包含 8 个滤波通道 FC，VC 与 FC 的对应关系通过软件配置实现。每个 VC 可任意选择 8 个 FC 中的 1 个进行采样结果滤波处理。8 个滤波通道独立支持 IIR 和非滑动平均 2 种滤波类型配置选择。8 个滤波通道的滤波参数均可独立配置。滤波通道的输入数据为 ADC 采样值经过 ADC 校准通道处理后的结果。

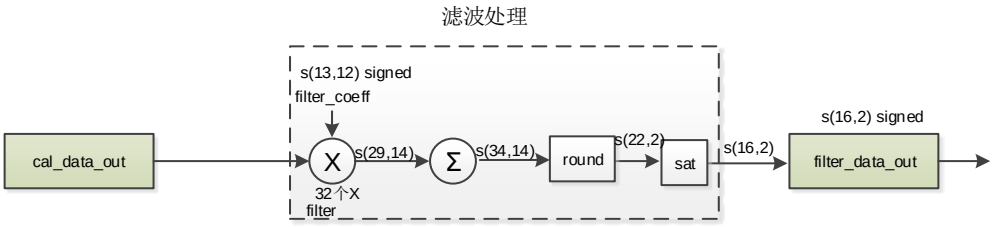


图21-16 滤波通道处理流程

21.4.10.1 IIR

ADC 中的 IIR 滤波器采用一阶低通滤波。

$$Y(n) = \alpha X(n) + (1-\alpha)Y(n-1)$$

式中：

- α 为滤波系数，取值范围为：0≤α≤1
- X(n)为本次采样值
- Y(n−1)为上次滤波输出值
- Y(n)为本次滤波输出值

21.4.10.2 非滑动平均

“非滑动平均”滤波，根据用户配置的滤波次数，将 2^n 个转换结果进行累加，然后通过将累加右移 n 位，得到非滑动平均的滤波结果。其原理示意图如下：

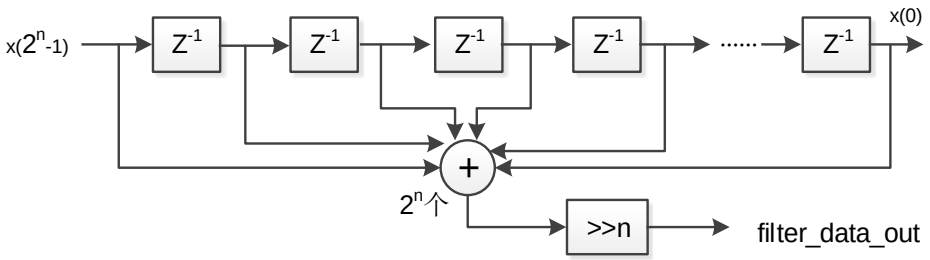


图21-17 非滑动平均滤波原理

其中，n 取值范围为 4bit 可配置，即最大滤波次数为 2^{15}。

21.4.11 缓存通道

每个 ADC 包含 2 组采样结果缓存通道 BC1 和 BC2，每组缓存通道包含 16 个独立寄存器以存储 16 个 VC 的最终采样结果，每个结果寄存器与 VC 一一对应。2 组 BC 通道内结果寄存器地址空间连续排布，便于 DMA 进行数据读取。

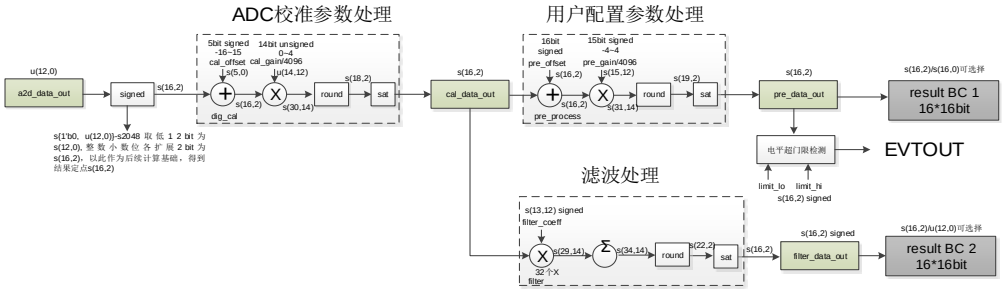


图21-18 采样结果缓存通道示意

当有新的采样数据写入缓存通道时，数据有效标志 val 置高，如果此时软件或者 DMA 未读取同通道前一次采样数据，将产生 ovf 告警信号上报。val 标志和 ovf 告警都是由软件或者 DMA 进行读清。每个采样结果缓存寄存器都有独立的 val 和 ovf 指示。

21.4.11.1 缓存通道 BC1

缓存通道 BC1 存储采样值经校准处理到用户预处理后的采样结果。

支持 2 种数据格式的采样结果，数据格式软件可通过 **CFG_RESULT_CTL.cfg_pre_res_mod** 进行配置选择：

- s(16,2)，一共 16bit 数据，有符号数，其中低 2bit 为小数，且该格式数据值包含-2048 的偏置；
- s(16,0)，一共 16bit 数据，有符号数，无小数位，且该格式数据值不包含-2048 的偏置；

21.4.11.2 缓存通道 BC2

缓存通道 BC2 存储采样值经校准处理到滤波处理后的采样结果。

支持 2 种数据格式的采样结果，数据格式软件可通过 **CFG_RESULT_CTL.cfg_fil_res_mod** 进行配置选择：

- s(16,2)，一共 16bit 数据，有符号数，其中低 2bit 为小数，且该格式数据值包含-2048 的偏置；
- u(12,0)，一共 12bit 数据，无符号数，无小数位，且该格式数据值不包含-2048 的偏置；

21.4.11.3 缓存通道的 FIFO

缓存通道 BC1 和 BC2 分别支持 FIFO 模式读取采样结果，每个 BC 对应一个独立的 FIFO 模式读地址。软件通过配置 bitmap 的方式从 BC 内的 16 个结果寄存器中选择相应的寄存器映射到 FIFO 模式读地址，BC1 和 BC2 的 FIFO 模式工作参数均独立配置。

缓存通道 FIFO 模式应用特性：

- bitmap 由配置寄存器**CFG_SARC_VCEN_CTL** 和**CFG_FIFO_VC_SEL** 共同决定，即 **bitmap[15:0] = *_vc_en[15:0] & *_fifo_vc_sel[15:0]**。*_fifo_vc_sel** 默认为全 1，即默认所有使能的 VC 通道均映射到 FIFO 模式。
- 在初始化或者更新 bitmap 相关参数配置后，都需要对寄存器 **CFG_FIFO_CTRL.*_bitmap_reload** 进行写 1 操作，才能完成相应的初始化或者参数更新。
- 当应用 FIFO 模式时，软件必须保证 VC 通道的采样顺序必须与 bitmap 配置中 VC 通道编号从小到大的顺序相同。
- 软件可通过寄存器 **CFG_FIFO_CTRL.*_fifo_depth** 配置对应 FIFO 的深度，FIFO 最大深度为 16，且默认配置深度为 16。
- 软件或 DMA 通过 FIFO 模式读取采样结果时，数据格式为**{vc_num[3:0], ovf, val, data[15:0]}**。
- 软件可通过状态寄存器***_FIFO_STS** 查询 FIFO 状态，包括空，满，空溢出，满溢出，FIFO 中数据个数。

下面以 vc0/vc1/vc2/vc3 映射到 FIFO 模式为例简单说明：

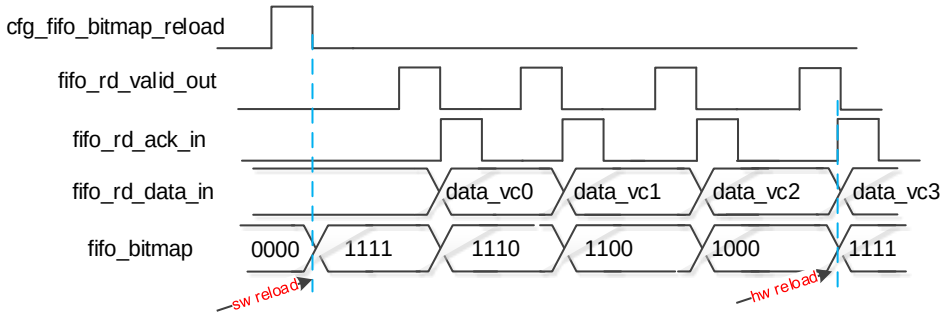


图21-19 FIFO 模式应用示意

bitmap 初始化通过 reload 进行加载，该示例的有效 bitmap 为 16'h000f。

软件或者 DMA 通过 FIFO 模式读地址进行采样结果读取，每次读取 bitmap 中最低位为 1 的 bit 对应的 VC 通道采样结果。

硬件对 bitmap 进行管理，每次读取后，对 bitmap 上的对应位进行清 0 操作；在最后一次读取后，对 bitmap 进行硬件初始化操作。

21.4.12 中断

每个 ADC 输出 1bit 中断信号，包含如下中断源：

- EOC 脉冲信号，基于 VC 通道，16 个中断源
- 模拟看门狗告警
  - 采样结果超过上门限阈值，基于 VC 通道，16 个中断源
  - 采样结果超过下门限阈值，基于 VC 通道，16 个中断源
- 采样结果进入缓存通道
  - 缓存通道 BC1，基于 VC 通道，16 个中断源
  - 缓存通道 BC2，基于 VC 通道，16 个中断源

当中断产生时，上一中断还未响应 (中断未被清除)，此时产生中断溢出告警指示。

中断处理按下面的方式统一处理：

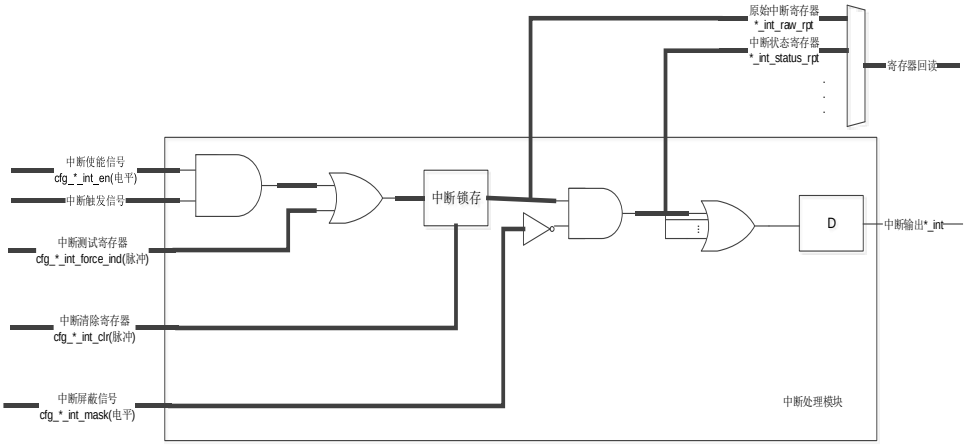


图21-20 中断处理

21.4.13 DMA

每个 ADC 支持 4 个 DMA 请求通道，4 个通道的请求信号可独立从以下请求源选择：

- EOC 脉冲信号，基于 VC 通道，16 个中断源
- 模拟看门狗告警
  - 采样结果超过上门限阈值告警与采样结果超过下门限阈值告警相或，基于 VC 通道，16 个中断源
- 采样结果进入缓存通道
  - 缓存通道 BC1，基于 VC 通道，16 个中断源
  - 缓存通道 BC2，基于 VC 通道，16 个中断源
- 缓存通道 BC1 对应的 FIFO 模式中，FIFO 的非空信号
- 缓存通道 BC2 对应的 FIFO 模式中，FIFO 的非空信号

ADC 的 DMA 通道选择请求源的方式如下图。

从 EOC 脉冲信号 16bit、超门限检测看门狗告警 16bit、预处理后采样结果指示 16bit、滤波后采样结果指示 16bit，共 64bit 的请求源中通过 **cfg_sarc_dma_ch*_sel[5:0]**配置选择 1bit 请求信号 A。

从两个 FIFO 的非空信号中通过 **cfg_sarc_fifo_dma_ch*_sel** 配置选择一个 FIFO 的请求

信号 B。

然后从请求信号 A 和请求信号 B 中通过 **cfg_sarc_fifo_dma_ch*_en** 配置选择一个信号作为 ADC 的 DMA 请求输出。

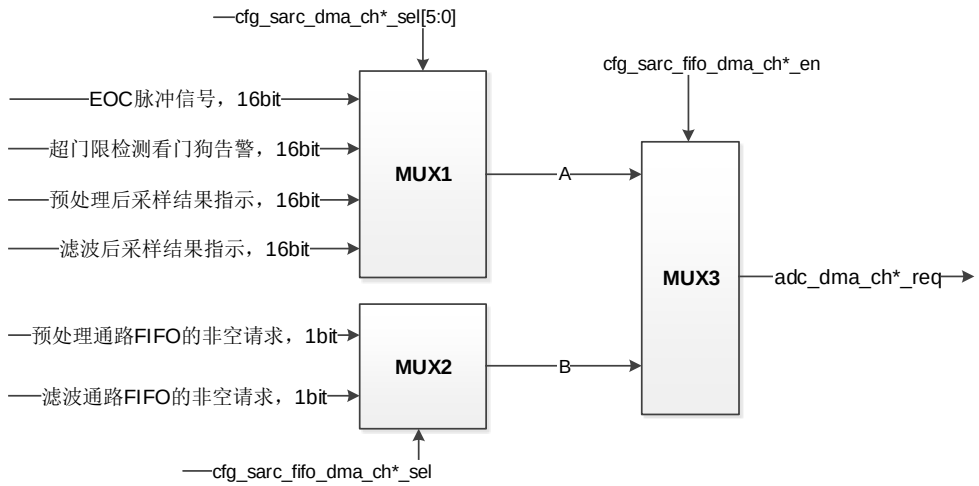


图21-21 DMA 请求选择方式示意

21.4.14 状态和告警上报

- VC 采样溢出 (16 个 VC 独立上报状态)

  16 个 bit 对应 16 个 VC 状态，为 1 表示对应 VC 正在排队时又收到有效触发；
- 模拟 ADC 工作状态
  - Idle：指示 ADC 空闲，没有采样任务；
  - Busy：指示 ADC 正在进行采样；
  - 通道标识：当 ADC 处于 Idle 时，指示前一个采样的 VC 号；当 ADC 处于 Busy 时，指示当前正在采样的 VC 号；
- VC 工作状态 (16 个 VC 独立上报状态)
  - Idle：对应 VC 处理空闲状态，没有被有效触发；
  - Busy：对应 VC 正在进行采样；
  - Pending：对应 VC 被有效触发，但因为优先级较低而处于排队状态；
- VC 排队状态
  - 16 个 bit 对应 16 个 VC 状态，为 1 表示对应 VC 被有效触发后处于排队状态，为 0 表示对应 VC 没有被触发或者正在进行采样；
- 采样结果溢出
  - 当有新的采样数据写入缓存通道时，软件或者 DMA 未读取前一次采样数据，产生告警；

21.4.15 约束说明

1. ADC 数字控制器不能早于模拟 ADC 解复位；

2. 系统时钟频率不能低于 ADC 工作时钟频率；
3. 外部输入的采样触发源脉冲需要为正脉冲，且脉冲宽度要大于 1 个系统时钟周期(因为是同步处理)；
4. ADC 校准时，要求关闭模拟 ADC 使能，待校准流程完成后再打开模拟 ADC 使能；
5. 对虚拟通道 VC 的配置，需要将对应的 VC_EN 关闭后进行配置，配置完成后再将 VC_EN 打开；
6. 模拟 ADC 的 EN 应该最先打开，然后是数字控制器的 EN 信号，然后是数字控制器内的其他 EN 信号 (比如 VC_EN 等)；

21.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 ADC 章节。

22 数模转换器 (DAC)

22.1 简介

DAC 用于将 12bit 的数字值通过模拟器件转换成模拟电压，输出电压值为 0~Vref，包含 2 个独立通道，每个通道的触发源和控制寄存器等相互独立。DAC 支持立即加载和影子加载模式，可以产生静态波、方波、三角波、正弦波等波形，既可以通过配置 DAC 发数寄存器产生，又可以通过 WAG 外设产生。DAC 输出由电压缓冲器驱动。

22.2 结构框图

DAC 内部结构图如下：

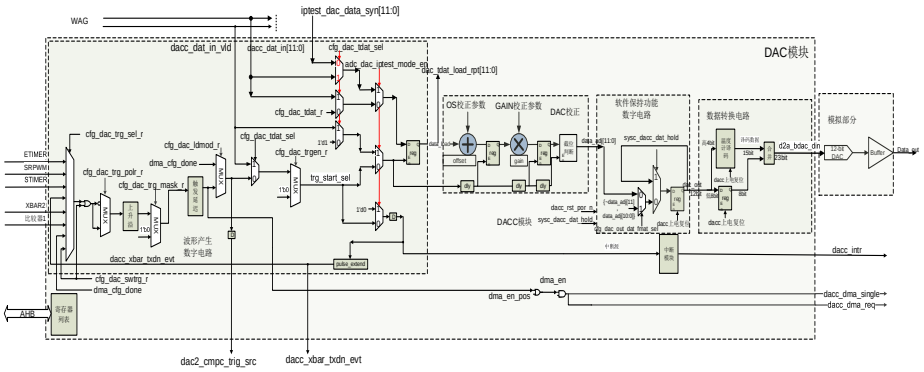


图22-1 DAC 内部结构图

22.3 主要特性

主要特性如下：

- 支持 2 个独立通道，有不同的输出电压和中断；
- 支持配置为静态输出特定值；
- 支持产生静态波、方波、三角波、正弦波等波形，既可以通过配置 DAC 发数寄存器产生，又可以通过 WAG 外设产生；
- 支持事件触发影子加载模式和软件配置寄存器立即加载模式；
- 支持通过 CPU 和 DMA 握手配置发波数据；
- 支持对 2 个通道的 DAC 上电校准；
- 支持软件配置输出保持功能；
- DAC 的输出支持补码和偏移码选择；
- DAC 的输出支持数字配置反向选择；

- 可配置选择外灌基准或者内部带隙基准电压；

22.4 功能描述

22.4.1 DAC 引脚和内部信号

表22-1 DAC 模块接口说明

信号	输入输出	说明
dac0_hclk	Input	DAC0 外设时钟，最大 200 MHz AHB 总线时钟
dac1_hclk	Input	DAC1 外设时钟，最大 200 MHz AHB 总线时钟
stim_dacc0_trg_src	Input	STIMER 到 DAC0 的触发源
stim_dacc1_trg_src	Input	STIMER 到 DAC1 的触发源
etim_dacc0_trg_src	Input	ETIMER 到 DAC0 的触发源
etim_dacc1_trg_src	Input	ETIMER 到 DAC1 的触发源
spwm_dacc0_trg_src	Input	SRPWM 到 DAC0 的触发源
spwm_dacc1_trg_src	Input	SRPWM 到 DAC1 的触发源
sysc_dac0_dat_hold	Input	SYSCTRL 模块输出的 DAC0 外部保持功能信号
sysc_dac1_dat_hold	Input	SYSCTRL 模块输出的 DAC1 外部保持功能信号
d2a_bdac0_din	Output	DAC0 数字端输出到模拟端的数字值
d2a_bdac1_din	Output	DAC1 数字端输出到模拟端的数字值
dac0_intr	Output	DAC0 输出中断
dac1_intr	Output	DAC1 输出中断
dac0_txdn_evt	Output	DAC0 输出发送完成事件
dac1_txdn_evt	Output	DAC1 输出发送完成事件

22.4.2 DAC 通道使能

在 SYSC 系统控制器的非易失性寄存器 **AFE_DAC.d2a_bdac0_en** 和 **AFE_DAC.d2a_bdac1_en**，分别代表 DAC 的通道 0 和通道 1 的使能信号。当对应的寄存器配置为 1 时，才能送出 DAC 静态波形。

22.4.3 DAC 转换输出模式

DAC 双通道只支持一种直接输出静态波形模式，可通过触发源触发或立即加载的方式发送波形。

- 立即加载时，加载模式选择寄存器 **CFG_DAC_TRG_SET.cfg_dac_ldmod** 配置为 **0**，软件配置 DAC 发送数据寄存器 **CFG_DAC_TDAT.cfg_dac_tdat** 时，会立即输出 DAC 电压值。并且产生发波结束事件 dac_txdn_evt。

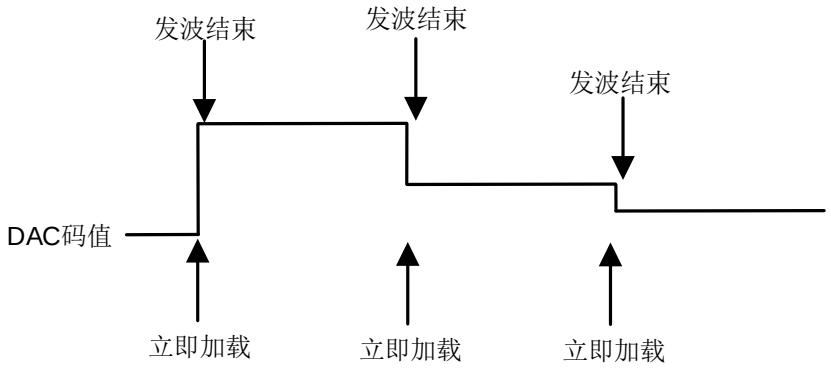


图22-2 立即加载 DAC 输出

- 触发源触发加载时，加载模式选择寄存器 **CFG_DAC_TRG_SET.cfg_dac_ldmod** 配置为 1，软件配置 DAC 发送数据寄存器 **CFG_DAC_TDAT.cfg_dac_tdat**，此时 DAC 不会输出，待触发源触发时，会更新 DAC 输出电压值。

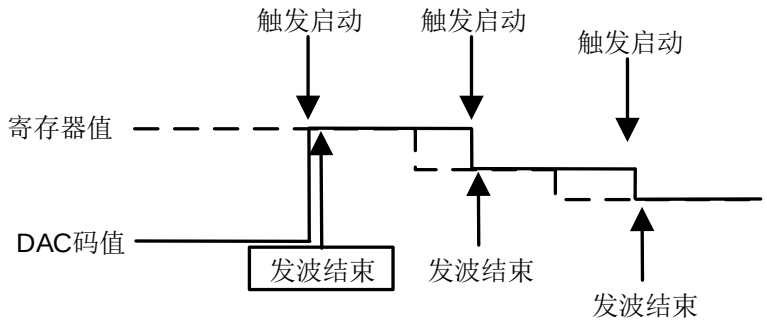


图22-3 触发源触发加载 DAC 输出

22.4.4 DAC 输出电压

理想的 DAC 电压输出参考下列公式：

DACOUT=DACVALA_IN*Vref/4095

其中：

- DACOUT 为模拟 DAC IP 的输出；
- DACVALA_IN 为数字 DAC IP 的输入，输入范围为 0~4095 的无符号数；
- Vref 为模拟 DAC IP 的参考电压；

22.4.5 DAC 触发事件选择

DAC 有两个通道，各有 7 个触发源输入，这 7 个触发源在在每个通道的含义一致。DAC

触发源有三种类型的触发方式，通过**CFG_DAC_TRG_SEL.cfg_dac_trg_sel**寄存器来选择：

- 内部触发：

  包含 SW_TRG 软件配置触发源输入、DMA_TRG DMA 加载完成触发源输入、DAC 发送数据完成触发源输入。

  SW_TRG 软件配置触发源是配置寄存器 **CFG_DAC_TRG_SET.cfg_dac_swtrg** 来触发。

  DMA_TRG DMA 加载完成触发源在 DMA 完成 DAC 发数寄存器**CFG_DAC_TDAT.cfg_dac_tdat** 配置之后，由底层逻辑产生。

  DAC 发送数据完成触发源由 DAC 内部自己的发波结束事件产生。

- 外部触发：

  包含 STIM_TRG、ETIM_TRG、SRPWM_TRG 共 3 个外部触发源。

  - STIM_TRG 为 STIMER 溢出事件循环触发源，DAC0 STIM_TRG 由 STIMER0 的溢出事件触发，DAC1 STIM_TRG 由 STIMER1 的溢出事件触发。
  - ETIM_TRG 为 ETIMER PWM 比较事件触发源，DAC0 由 ETIMER0 的 PWM 比较事件触发；DAC1 由 ETIMER1 的 PWM 比较事件触发。
  - SRPWM_TRG 为 SRPWM 送给 DAC 同步事件触发源，2bit 位宽。其中 DAC0 的 SRPWM_TRG 触发源为同步事件低 bit，DAC1 的 SRPWM_TRG 触发源为同步事件高 bit。

- 静默：

  NO_TRG 为静默，此时触发源固定接 0，不会产生触发事件。

TRG Source	DAC_TRG_SEL	说明
SW_TRG	000	软件配置触发源输入
STIM_TRG	001	STIMER 溢出事件循环触发源输入
ETIM_TRG	010	ETIMER 事件触发源输入
SRPWM_TRG	011	SRPWM 事件触发源输入
DAC_TRG	100	DAC 发波完成触发源输入
NO_TRG	101/110	固定接 0，没有触发输入
DMA_TRG	111	DMA 加载完成触发源输入

22.4.6 触发源信号切换

由于触发源选择信号 **CFG_DAC_TRG_SEL.cfg_dac_trg_sel** 是一个动态切换信号，需要在切换时将触发源屏蔽寄存器 **CFG_DAC_TRG_SET.cfg_dac_trg_mask** 寄存器控制为 1，屏蔽触发源输出为 0，避免在切换中，出现异常情况导致下游电路误操作。

触发启动源的切换过程中，建议软件配置触发源屏蔽寄存器为屏蔽状态和解除屏蔽状态之间的时间间隔必须大于 1 us，以满足模拟 DAC 数据速率为 1 M 的要求。如果在切换时不满足，有可能在切换后的第一个 DAC 数据异常，但是后面会恢复正常。

22.4.7 DMA 请求

DAC 两个通道分别有独立的 DMA 握手使能寄存器 **CFG_DAC_HDSK_EN.cfg_dac_hdsk_en**，默认是关闭状态。

在使用 DMA 请求去配置 DMA 发波数据时，每次 DMA 通过 AHB 总线只配置一次 **CFG_DAC_TDAT.cfg_dac_tdat** 数据，在下次触发源触发时发送到模拟 DAC。

建议在使用 DMA 请求发波时，触发源只选择 STIM_TRG、ETIM_TRG 和 SRPWM_TRG，而且需要软件保证每次触发源触发时，触发的时间间隔必须大于 1 us，以满足模拟 DAC 数据速率为 1 M 的要求。

22.4.8 DAC 矫正

22.4.8.1 校正原理

DAC 的非理想因素会引起 DAC 的直流失配 (DC offset,简称为 Offset) 和增益误差 (Gain error)。在本设计的校正方案中包含 Offset 校正和 Gain 校正。

由于 DAC 的失配表征为模拟信号的大小，为提取出模拟量，需要借助 ADC 对模拟值量化。因此在本校正方案实施前需要优先完成 ADC 的校正。ADC 的校正参考 21.4.8 ADC 校。

ADC 完成校正后，需要配相关寄存器把 DAC 连接过去，如下图：

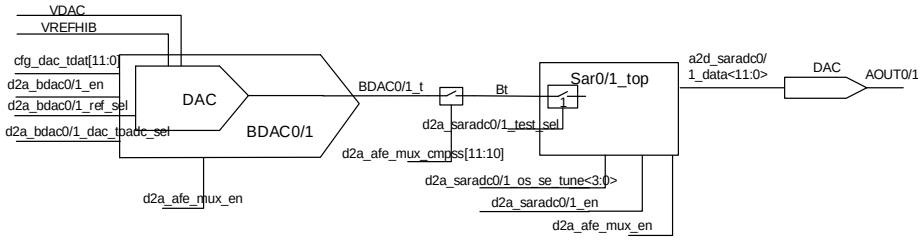


图22-4 DAC 校准连接关系图

1. 在 DAC 校正模式时，DAC0 的输出连接到 ADC1(sar1)，DAC1 的输出连接 ADC1 (sar1)。
2. 首先配置 SYSC 寄存器，DAC1 和 DAC0 的 DAC 测试控制位 **AFE_DAC.d2a_bdac1_dac_toadc_sel** 和 **AFE_DAC.d2a_bdac0_dac_toadc_sel** 的寄存器为 1，选择为测试模式，把 DAC 输出接到 ADC 输入。
3. 配置 SYSC 寄存器，DAC1 和 DAC0 的 dac reference 源选择为内部 2.5v ref buffer 输入，相应配置 **AFE_DAC.d2a_bdac1_ref_sel** 和为 **AFE_DAC.d2a_bdac0_ref_sel1** (内部供电 ref 为 2.5 V；为 0 的时候则为外部供电)。
4. 配置 SAR ADC 寄存器 **CFG_SARC_CAL_CTL.cfg_sarc_adc_test_sel** 为 1，控制 AFE 数模接口 d2a_saradc0_test_sel 和 d2a_saradc1_test_sel 的寄存器为 1，打开 ADC0/1 的测试模式，使 ADC0/1 测试内部 DAC 的输出，用于 DAC 校正。
5. 配置 SYSC 寄存器 **AFE_MUX.d2a_afe_mux_cmpss<15:0>**的第 11bit 和第 10bit 位，

使 **d2a_afe_mux_cmpss <11>**和 **d2a_afe_mux_cmpss <10>**为 1，其他为 0。

6. bit11 为 1：BDAC0 测试节点 BDAC0_t 连到 ADC0 的 test 输入 Bt (BDAC test，见上图)。bit10 为 1：BDAC1 测试节点 BDAC1_t 连到 ADC1 的 test 输入 Bt。
7. 配置 SYSC 寄存器 **AFE_DAC.d2a_bdac1_en** 和 **AFE_DAC. d2a_bdac0_en** 为 1，使能 BDAC1 和 BDAC0。
8. 配置 SYSC 寄存器 **AFE_MUX.cfg_d2a_afe_mux_en**，使能 AFE test 功能。
9. 配置 SAR ADC 寄存器 **CFG_SARC_ADC_CTL0.cfg_sarc_adc_en** 为 1，使能 ADC Core。此时 **d2a_saradc1_en** 和 **d2a_saradc0_en** (使能端) 为 1；

在 DAC 校正时，首先需要完成 DAC offset 校正，随后完成 Gain 校正，如下图。

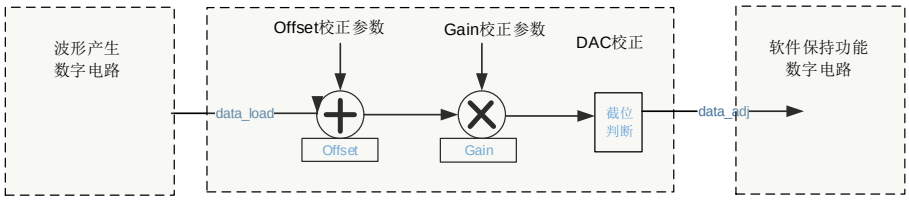


图22-5 DAC 校正

- data_load：由波形产生数字电路产生的静态数据输出。12bit 无符号数取值范围为 0~4095 (2^12-1)。
- Offset 校正参数：软件经过 DAC offset 校正后，配置下来的直流失配调整值，13bit 有符号数，8 位整数位，四舍五入保留 4 位小数位。
- Gain 校正参数：软件经过 DAC gain 校正后，配置下来的增益误差值，11bit 无符号数，整数位是 1 位，小数位是 10 位。
- 截位判断：经过 DAC offset 校正和 DAC gain 校正后，得到的结果整数位和小数位被扩展，需要截位后去掉小数位，剩下的有符号整数也需要判断，将数值饱和在 0~4095 之间，使其以 12bit 无符号整数输出。具体操作如图：

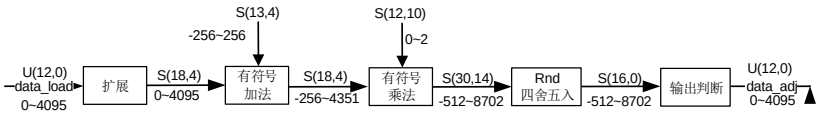


图22-6 校正符号位判断

data_load 为 12bit 无符号数据 u(12,0)和有符号 s(13,4)配置值 OS 校正参数相加，需要将 data_load 添加符号位和小数位，u(12,0)转变为有符号数 s(17,4)，此时 data_load 添加的符号位为 0。为了避免相加时数据上溢，需要将 data_load 扩展一位整数，变为 s(18,4)，符号位扩展后所表示的范围还是 0~4095。

data_load 和 Offset 校正参数相加后，与增益误差值 Gain 校正参数相乘，s(18,4)*s(12,10)=s(30,14)，小数位不输出，截掉小数位后数据变为 s(16,0)，此时可以判断截掉后的数据，如果小于 0 则饱和到 0，如果大于 4095，则饱和到 4095 输出给软件保持功能数字电路。

data_adj：校正后的数据输出，无符号 12bit 数据。

22.4.8.2 Offset 校正

首先配置模拟 MUX 使 DAC 输出至 ADC (ADC 配置为正常转换模式)。

1. 配置 DAC 输入 code 为 2047。读取 ADC 的转换值，为了滤除噪声，ADC 连续转换 8/16/32 次，四舍五入取平均值计为 code1。
2. 配置 DAC 的输入值为 code1，读取 ADC 的转换值，ADC 连续转换 8/16/32 次，四舍五入取平均值计为 code2。
3. 提取 DAC 的 offset 值，os=code2-code1，8 位整数位，四舍五入保留 4 位小数位。

22.4.8.3 Gain 校正

首先完成 DAC offset 校正，并配置模拟 MUX 使 DAC 输出至 ADC (ADC 配置为正常转换模式)。

1. 配置 DAC 的输入 code_dac2=1023。读取 ADC 的转换码字，ADC 连续转换 8/16/32 可配置，四舍五入取平均值计为 code1。
2. 配置 DAC 的输入 code_dac2=3071。读取 ADC 的转换码字，ADC 连续转换 8/16/32 可配置，四舍五入取平均值计为 code2。
3. 计算并提取 DAC 的增益误差 gain=(code2-code1)/2048。四舍五入保留 10 位小数。

22.4.9 输出保持

DAC 需要支持软件配置输出保持模式，便于低功耗或其他模式保持 DAC 输入值。当配置为输出保持模式时，输出当前配置值。保持模式输出寄存器只被 POR 复位。

sysc_dac0/1_dat_hold 为 SYSCTRL 模块输出的外部保持功能选择信号，当 DAC 模块需要输出保持时，**sysc_dacc_dat_hold** 从低电平配置为高电平，将校正模块输出 data_adj 的值锁存起来输出到模拟部分。DAC 的输出可以通过配置寄存器 **cfg_dac_out_dat_fmat_sel** 选择补码和偏移码。

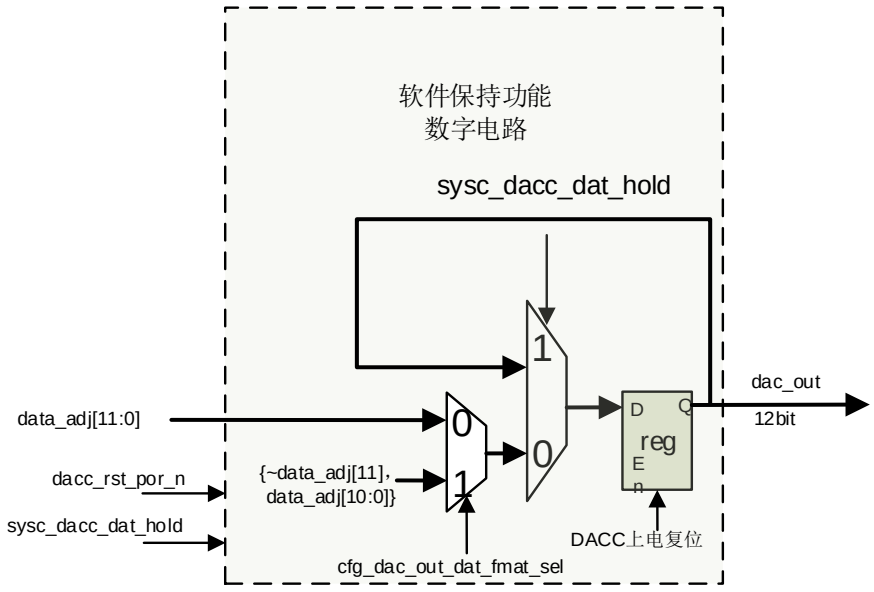


图22-7 DAC 输出保持功能

22.4.10 数据转换电路

对 DAC 12bit 数据处理，产生模拟 DAC IP 需要的数据格式，方便模拟 DAC IP 处理，具体要求如下：

1. 12bit 为无符号数，定义分为高 4bit 和低 8bit 位。其中高 4bit 位采用温度计译码方式译码，低 8bit 不做变换，如下表格：

表22-2 温度计译码

高 4Bit	译码后的数据
0000	15'b000_0000_0000_0000
0001	15'b000_0000_0000_0001
0010	15'b000_0000_0000_0011
0011	15'b000_0000_0000_0111
……..	…….
1110	15'b011_1111_1111_1111
1111	15'b111_1111_1111_1111

2. 高 4bit 译码后的数据和原始低 8bit 数据，打拍对齐，同步送给模拟 DAC。bit 转换关系如下图。

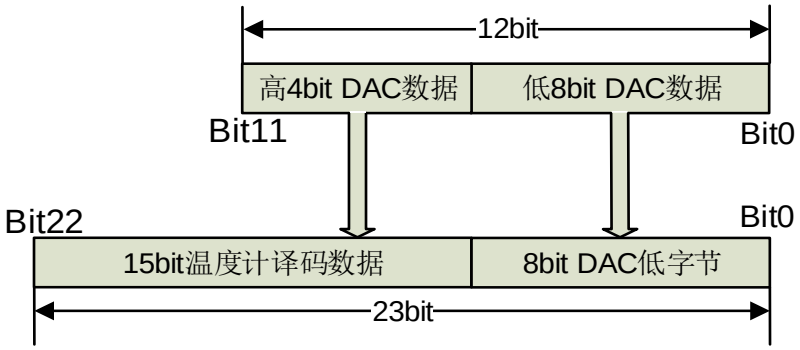


图22-8 bit 转换关系图

输出的 23bit 中，高 15bit 为 DAC 12bit 中高 4bit 的温度计译码后输出结果，23bit 的低 8bit 为 DAC 12bit 中低 8bit 打拍后，与 15bit 对齐后输出。

3. DAC 的输出支持数字配置反向选择。如下图所示。

输出无反向	
din<11:0>	d2a_bdac_din<23:0>
0000 0000 0000	000 0000 0000 0000 0000 0000
0000 0000 0001	000 0000 0000 0000 0000 0001
...	.....
0001 1010 1010	000 0000 0000 0001 1010 1010
0001 1010 1011	000 0000 0000 0001 1010 1011
...	.....
0100 1001 1010	000 0000 0000 1111 1001 1010
0100 1001 1011	000 0000 0000 1111 1001 1011
...	.....
1000 1000 1100	000 0000 1111 1111 1000 1100
1000 1000 1101	000 0000 1111 1111 1000 1101
...	.....
1010 0111 0000	000 0011 1111 1111 0111 0000
1010 0111 0001	000 0011 1111 1111 0111 0001
...	.....
1111 1111 1110	111 1111 1111 1111 1111 1110
1111 1111 1111	111 1111 1111 1111 1111 1111

输出有反向	
din<11:0>	d2a_bdac_din<23:0>
0000 0000 0000	111 1111 1111 1111 1111 1111
0000 0000 0001	111 1111 1111 1111 1111 1110
...	.....
0001 1010 1010	011 1111 1111 1111 0101 0101
0001 1010 1011	011 1111 1111 1111 0101 0100
...	.....
0100 1001 1010	000 0111 1111 1111 0110 0101
0100 1001 1011	000 0111 1111 1111 0110 0100
...	.....
1000 1000 1100	000 0000 0111 1111 0111 0011
1000 1000 1101	000 0000 0111 1111 0111 0010
...	.....
1010 0111 0000	000 0000 0001 1111 1000 1111
1010 0111 0001	000 0000 0001 1111 1000 1110
...	.....
1111 1111 1110	000 0000 0000 0000 0000 0001
1111 1111 1111	000 0000 0000 0000 0000 0000

图22-9 DAC 的输出反向选择

22.4.11 中断

每一路 DAC 都能产生独立的中断 dacc_intr，为 DAC 数据发送完成时的中断上报；该中断为电平输出。

22.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 DAC 章节。

23 模拟比较器 (ACMP)

23.1 简介

芯片内置 4 个 Dual-ACMP 比较器模块，每个 Dual-ACMP 比较器模块包含 2 个 ACMP (分别称为高电平 ACMP 和低电平 ACMP)。每个高电平 ACMP 包括一个下斜坡产生器，只能产生下斜坡波形，输入到高电平 ACMP 内部的 DAC。每个 ACMP 包含 1 个内部 DAC，1 个 CMP 和对应的 CMP 数字模块。

每个 ACMP 都有独立的配置寄存器，具体功能如下：

- 含有 1 个内部 DAC，用于静态电压配置；
- 含有 1 个 CMP，用于比较输出；
- 模拟比较器具有失调校正功能；
- 模拟比较器内置 DAC 基准可选择内部带隙基准产生或者外灌输入；

23.2 结构框图

ACMP 结构框图如下：

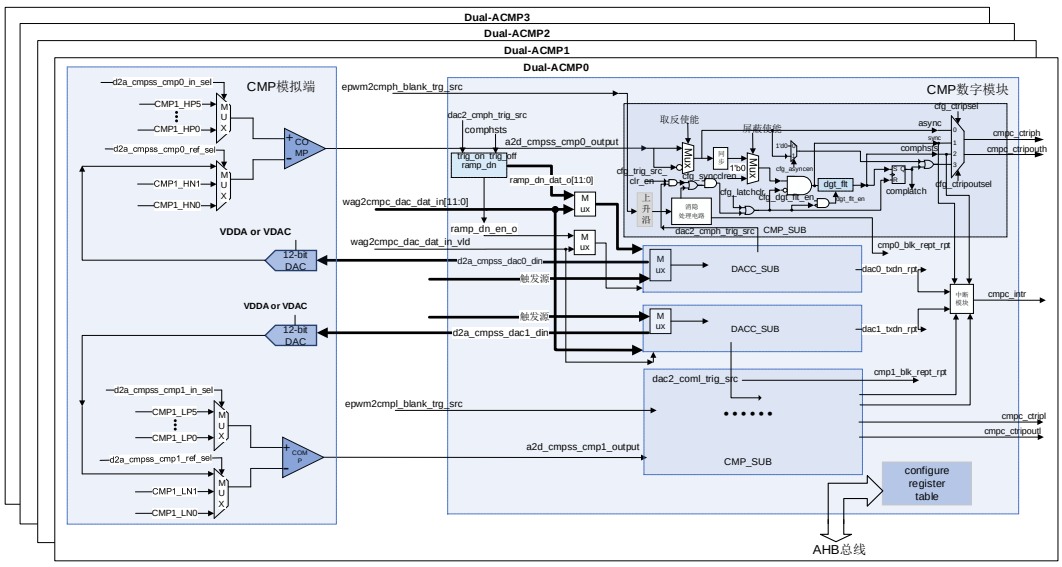


图23-1 比较器结构框图

23.3 主要特性

主要特性如下：

- 可选的参考端和比较端的模拟输入；

- Dual-ACMP 可以配置 2 个 DAC、模拟比较器和 CMP 数字模块成对工作；
- 高电平 ACMP 支持产生下斜坡波形输入到内部的 DAC；
- 支持对模拟比较器输出进行 Blanking 处理；
- 支持对模拟比较器输出进行数字滤波处理；

23.4 功能描述

23.4.1 主要输入输出信号

表23-1 ACMP 输入信号

编号	输入信号	描述
1	CMPx_HPx	比较器子系统高电平比较器正输入
2	CMPx_HNx	比较器子系统高电平比较器负输入
3	CMPx_LPx	比较器子系统低电平比较器正输入
4	CMPx_LNx	比较器子系统低电平比较器负输入
5	DACx	DAC 缓冲输出
6	stim_cmpx_trg_src	STIMER 到 ACMP 的触发源
7	etim_cmpx_trg_src	ETIMER 到 ACMP 的触发源
8	spwm_cmpx_trg_src	SRPWM 到 ACMP 的触发源
9	epwm2cmph_blank_trg_src[x]	SRPWM 到高电平 ACMP 的 Blanking 触发源
10	epwm2cmpl_blank_trg_src[x]	SRPWM 到低电平 ACMP 的 Blanking 触发源

表23-2 ACMP 输出信号

编号	输出信号	描述
1	cmpc_ctriph[3:0]	输出给 EQEP、XBAR、ETIMER、SARC
2	cmpc_ctripl[3:0]	输出给 EQEP、XBAR、ETIMER、SARC
3	cmpc_ctripouth[3:0]	输出给 XBAR
4	cmpc_ctripoutl[3:0]	输出给 XBAR

23.4.2 输出消隐

模拟 DAC 电平转换时，会在触发时刻点产生不稳定状态，所以需要 Blanking，把不稳定状态过滤掉。Blanking 源是内部 DAC 的影子触发源，这时需要配置 DAC 触发源的 Delay，需要把模拟 DAC 产生的不稳定状态包含在消隐脉冲里面。

消隐处理电路会根据软件配置的消隐脉冲宽度寄存器拓展消隐脉冲宽度。产生的消隐脉冲可旁路。时序图如下：

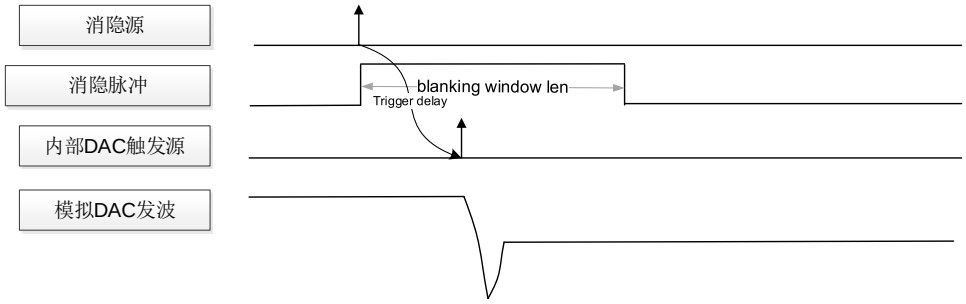


图23-2 消隐时序图

由软件配置消隐使能信号，若使能后，消隐源触发消隐脉冲，消隐脉冲可以根据软件配置的消隐脉冲宽度寄存器拓展消隐脉冲宽度；如果不使能，消隐源触发不会产生消隐脉冲。

23.4.3 数字滤波器 (dgt_flt)

数字滤波器首先将输入数据采样到 (cfg_sampwin+1) 个 FIFO 样本窗口上，然后滤波器再解析出样本窗口的大多数数值，其中大多数由阈值 (cfg_thresh) 定义。如果大多数阈值未满足，则滤波器输出保持不变。

为了正常运行，cfg_thresh 的值必须大于 cfg_sampwin/2。

预分频功能 (cfg_clkprescale) 确定滤波器的采样率，其中滤波器 FIFO 每隔 cfg_clkprescale 个系统时钟捕获一个样本。来自 FIFO 的旧数据将被丢弃。

数字滤波器的概念模型如图 23-3 所示。

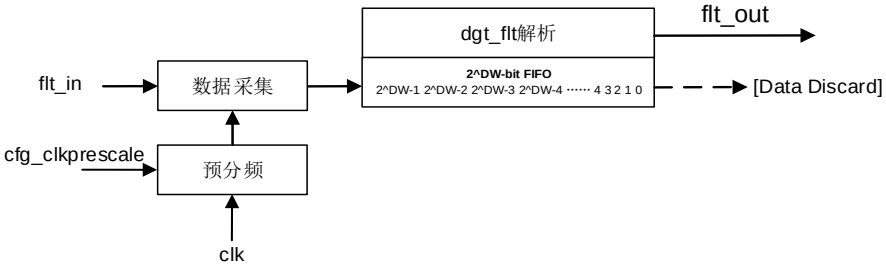


图23-3 数字滤波器的概念模型

滤波器实现的的等效 c 代码如下所示：

```
if(flt_out == 0) {
      if(Num_1s_in_SAMPWIN >= cfg_thresh){
            flt_out = 1;
      }
}
else {
      if(Num_0s_in_SAMPWIN >= cfg_thresh){
            flt_out = 0;
      }
```

```
}
```

数字滤波器的初始化顺序

为了确保 dgt_flt 的正常运行，建议按照以下初始化顺序进行操作：

1. 配置数字滤波器参数：
   - 设置 cfg_sampwin，确定 FIFO 窗口中要监测的样本数量。
   - 设置 cfg_thresh，确定大多数数值所需的阈值。
   - 设置 cfg_clkprescale，确定数字滤波器时钟预分频值。
2. 通过设置 cfg_filinit 来初始化数字 FIFO 窗口中的样本值。
   - 当 cfg_filinit=0 时，FIFO 中的样本值初始化为 0；
   - 当 cfg_filinit=1 时，FIFO 中的样本值初始化为 cfg_sampwin 个 flt_in 的采样值；

23.4.4 下斜坡产生器（ramp_dn）

当选择下斜坡产生器时，为内部 DAC 产生一个 12 位的下降斜坡输入。采用 rampsts 递减计数器（16 位计数器）的高 12 位作为 DAC 的输入，rampsts 递减计数器的低 4 位可以作为下降斜坡速率的预分频，下斜坡递减步进值寄存器可配。

通过设置寄存器 h_dacsource=1 来使能下斜坡产生器。当选择 h_dacsource=1 时，rampsts 的值从 cfg_rampmaxrefs 寄存器加载，并且保持静态，直到接收到 dac2_cmph_trig_src 信号。在接收到 dac2_cmph_trig_src 信号后，在随后的每个 ramp_vld (递减计数器的 valid 信号，对 cmp 数字模块的工作时钟进行预分频) 为高时中从 rampsts 中减去下斜坡递减步进值。

可以设置 cfg_rampdlyas 递减延时寄存器，在 dac2_cmph_trig_src 到来之后，延时 cfg_rampdlyas 个 ramp_vld，rampsts 递减计数器才开始递减，如果 rampsts 的值达到零，rampsts 寄存器将保持静态为零，直到接收到下一个 dac2_cmph_trig_src。

下斜坡产生器的示意框图如下图所示。

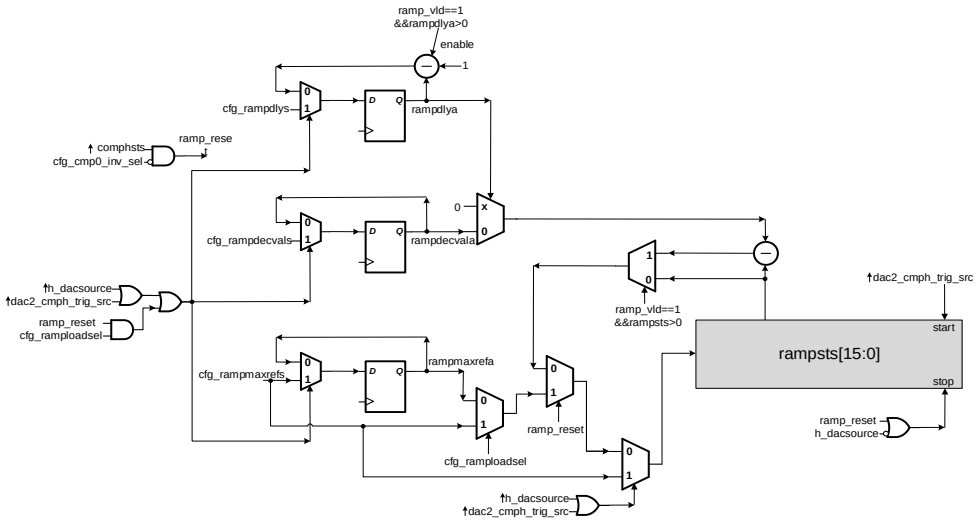


图23-4 下斜坡产生器示意框图

下斜坡产生器在 h_dacsource、dac2_cmph_trig_src 和 comphsts 的每个上升沿进行状态更

新。

- 在 h_dacsource 的上升沿：rampmaxrefa、rampdecvala 和 rampdlya 装载了它们的影子寄存器。rampsts 加载 cfg_rampmaxrefs。
- 在 dac2_cmph_trig_src 的上升沿：rampmaxrefa、rampdecvala 和 ramplya 装载了它们的影子寄存器。rampsts 加载 cfg_rampmaxrefs，并在 rampdlya 计数器达到零时开始递减。
- 在 comphsts 的上升沿：
  - 在 ramploadsel = 1 时 comphsts 的上升沿：rampmaxrefa, rampdecvala 和 rampdlya 加载了它们的影子寄存器。rampsts 装载 cfg_rampmaxrefs 并停止递减。
  - 在 ramploadsel = 0 时 comphsts 的上升沿：rampsts 加载 rampmaxrefa 并停止递减。

下图显示了下斜坡产生器中 rampsts 与启动和停止触发信号的时序关系。

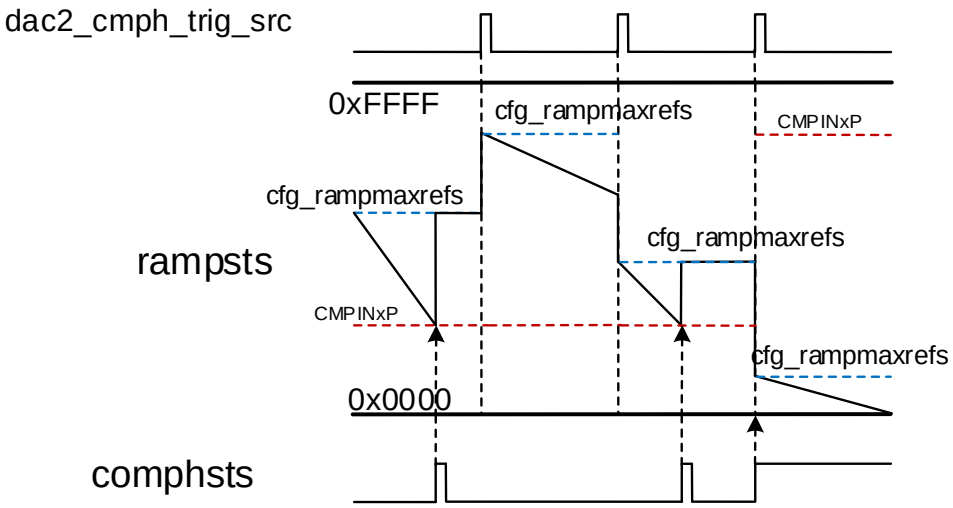


图23-5 rampsts 与启动和停止触发信号的时序关系

由于下斜坡产生器在 dac2_cmph_trig_src 和 comphsts 的每个上升沿进行状态更新，因此当这两个事件同时发生或非常接近时，下面的不同情形的行为是确定的。

- Case 1：comphsts 上升沿事件比 dac2_cmph_trig_src 上升沿事件提前发生一个或多个周期。

  当 comphsts 上升沿事件发生时，rampsts 停止递减。在 dac2_cmph_trig_src 上升沿事件，当 rampdlya 递减到 0 时，rampsts 开始递减。

- Case 2：comphsts 上升沿事件与 dac2_cmph_trig_src 上升沿事件同时发生。

  dac2_cmph_trig_src 上升沿事件优先，当 rampdlya 递减到 0 时，rampsts 开始递减。comphsts 上升沿事件被忽略，不停止 rampsts。

- Case 3：在 dac2_cmph_trig_src 上升沿事件发生一个或多个周期之后，且 rampdlya 递减到 0 之前，comphsts 上升沿事件发生。

  当 rampdlya 递减到 0 时，rampsts 不递减。

- Case 4：在 dac2_cmph_trig_src 上升沿事件发生后，且 rampdlya 递减到 0 时，comphsts

上升沿事件发生。当 rampdlya 递减到 0 时，rampsts 不递减。

23.4.5 中断

每一个 ACMP 都能产生独立的中断 cmpc_intr，可支持 DAC 发送完成中断、数字滤波前输出中断和数字滤波后输出中断；该中断为电平输出。

23.4.6 约束说明

1、 ctripouthsel、ctripoutlsel、ctriphsel、ctriplsel 这几个寄存器需要在 cmpc 工作之前配置位非 0 值，芯片不支持异步路径输出；

2、 asynchen 和 asynclen 寄存器保持默认值 0，芯片不支持异步路径；

23.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 ACMP 章节。

24 温度传感器 (Tsensor)

24.1 简介

温度传感器可用于测量器件结温。

温度传感器通过与 ADC 的内部连接进行采样，并通过软件转换为温度。

在对温度传感器进行采样时，ADC 必须满足表 24-1 中的采集时间要求。

24.2 温度传感器参数

温度传感器电气参数如表 24-1 所示。

表24-1 温度传感器电气特征

参数		最小值	典型值	最大值	单位
$T_{startup}$	启动时间				μs
Avg_Slope	平均斜率	2.456		2.521	mV/°C
$V_{30°C}$	30°C 时的电压	757.2	759.6	762.9	mV
t_{SH}	ADC 采样保持时间	200			ns

24.3 结构框图

温度传感器框图如图 24-1 所示。

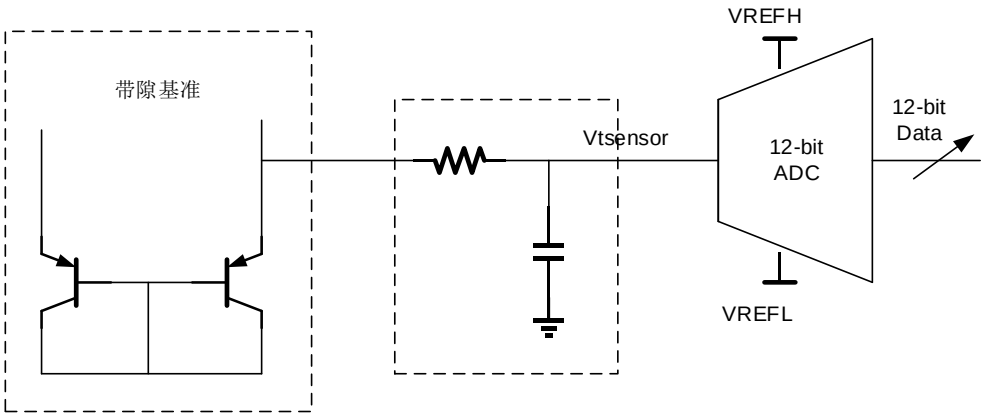


图24-1 温度传感器框图

24.4 主要特性

主要特性如下：

- 基准电压由片内高精度带隙基准产生；
- 通过片内 ADC 实现数字码字读取功能；
- 出厂前通过温度电压曲线微调，使用时可以通过出厂微调值进行温度校准；

25 XBAR

25.1 简介

XBAR 模块主要功能，是为芯片的输入管脚、输出管脚和内部模块之间提供灵活的连接关系，这些连接关系可通过配置进行选择，也可通过内部集成的 CLU (Configurable Logic Unit) 对部分连接关系进行逻辑运算，以提供芯片的灵活性和可用性。

25.2 结构框图

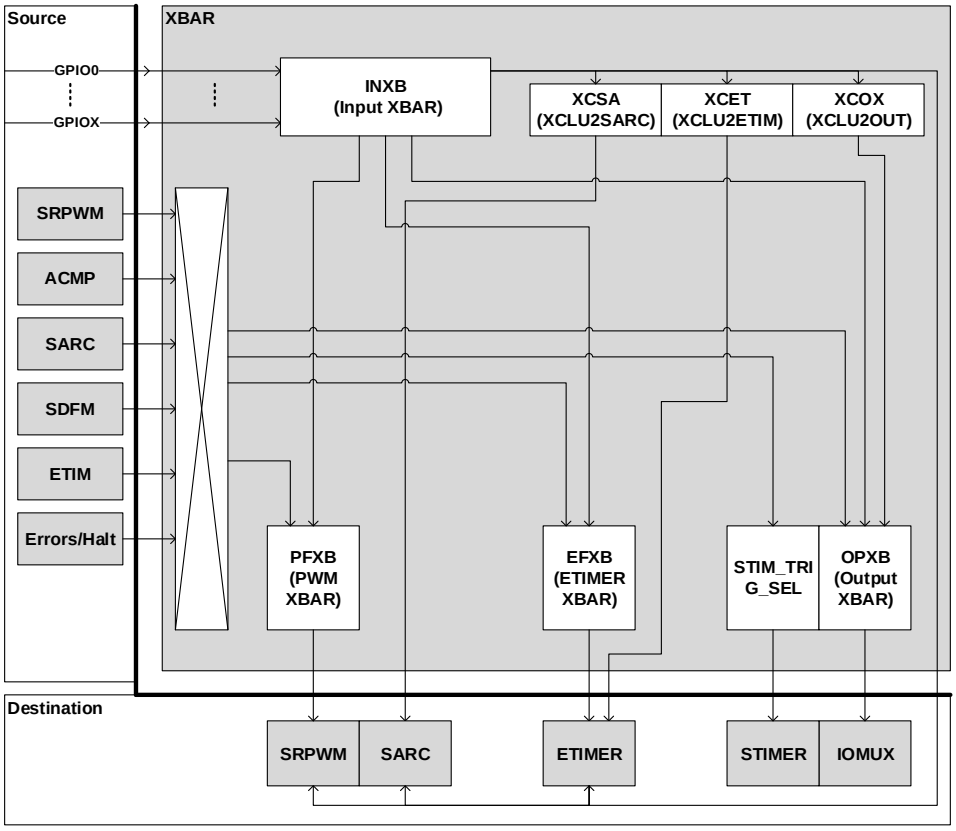


图25-1 XBAR 模块框图

25.3 功能特性

XBAR 支持对 GPIO 输入信号和来自内部模块的信号进行选择、逻辑计算等处理，处理后的信号可送往 SRPWM、SARC、ETIMER、STIMER 和 I/O，同时作为后级模块的信号输入、触发事件或故障处理信号。

XBAR 主要由以下几个模块组成：

- INXB (Input XBAR)，输出送往各内部模块；
- PFXB (PWM XBAR)，输出送往 SRPWM；
- STIM_TRIG_SEL，对紧急触发源进行选择后输出送往 STIMER；
- EFXB (ETIMER XBAR)，输出送往 ETIMER；
- OPXB (Output XBAR)，输出送往 IOMUX；
- CLU 模块，连接到 Input XBAR，输出作为 SARC、ETIMER 和 OPXB 输入信号来源。

除此之外，XBAR 还支持故障信号选择合并处理，配置写保护和中断上报等功能。

25.3.1 故障信号处理

XBAR 支持故障信号输入，并对故障信号进行合并，合并后的信号可作为 STIMER 紧急故障触发源和其他 XBAR 内部模块的 Fault 输入。通过配置 cfg_fault_en 对应 bit 使能实现故障信号选择合并，通过配置 **cfg_stm*_ergtrig_0en** 对应 bit 使能实现 STIMER 紧急触发源选择合并。

表25-1 XBAR 故障信号列表

故障信号	索引	信号说明	故障合并信号	紧急触发源合并信号
pflash_ecc_err	17	PFLASH ECC 错误指示	选择合并为 FLASH_ERR	选择合并为 STIMER 紧急触发源
pflash_bus_err	16	PFLASH 总线错误指示		
dflash_ecc_err	15	DFLASH ECC 错误指示		
dflash_bus_err	14	DFLASH 总线错误指示		
wdt0_rst	13	看门狗计数器 0 复位指示	选择合并为 SYS_ERR	
wdt1_rst	12	看门狗计数器 1 复位指示		
cpu_rst_rec	11	CPU 复位指示		
cpu_lockup	10	CPU 死锁指示		
cpu_ecc_err	9	CPU ECC 错误指示		
sram_ecc_err	8	SRAM ECC 错误指示		
can_ecc_err	7	CAN ECC 错误指示		
bus_timeout	6	总线超时指示		
cpu_bus_err	5	CPU 总线错误指示		
clock_fault	4	时钟故障告警		
por_uv_warn	3	欠压告警	选择合并为 PT_ERR	
pwr_ocp_warn	2	过流告警		
power_err	1	供电异常告警		
temp_warn	0	过温告警		

25.3.2 配置写保护

XBAR 支持通过配置一个 16bit Key (cfg_xbar_key) 实现寄存器写保护，支持写保护的寄存器包括：Fault 使能配置、STIMER 紧急触发源配置、CLU 相关配置、滤波窗口、滤波使能配置、输入输出极性配置、MUX-OR 配置、输入输出使能配置、输出选择配置、软件信

号源配置、输出脉宽扩展配置、输出模式选择配置。仅当该 Key 值配置为 **0x5a5a** 时可配置相关寄存器。

25.3.3 中断上报

XBAR 支持 5 个中断上报，其中 Input XBAR 支持 4 个中断独立输出，3 个 CLU 模块的中断源合并为 1 个中断输出，对每个中断源，可配置中断使能、中断屏蔽、中断清除、强制中断，可查询原始中断和中断状态。

25.3.4 Input XBAR

Input XBAR 输入均来自 GPIO，84 个 GPIO 信号经过极性选择，信号同步，接口滤波后，进行信号输出选择，每个输出信号可选择任意一路 GPIO 或多路 GPIO 的逻辑或作为输出，同时支持对输出信号进行锁存和取沿，输出使能控制和输出极性选择。

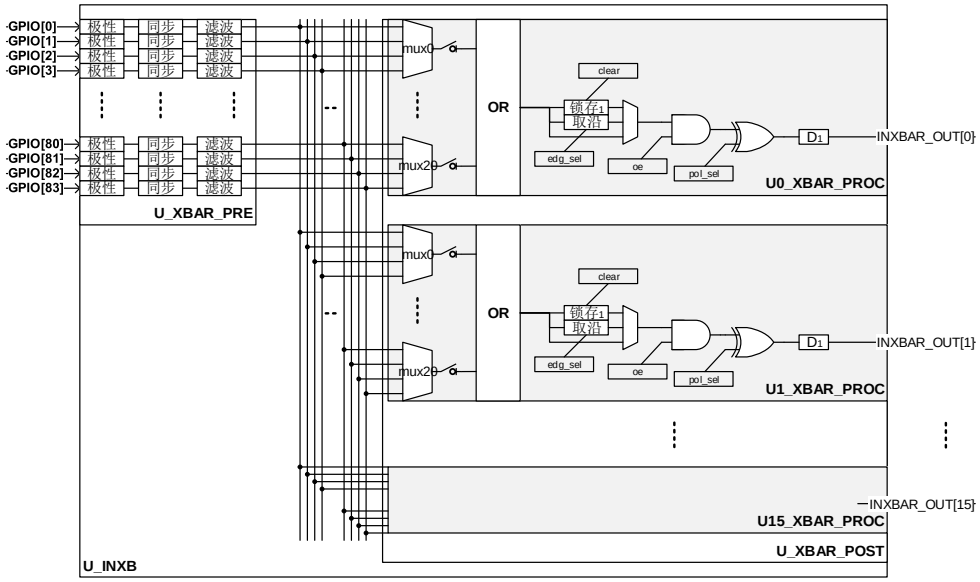


图25-2 Input XBAR 模块框图

Input XBAR 支持对输入信号进行预处理，包含极性选择，时钟域同步和接口滤波功能。

其中，极性选择通过配置 **cfg_inxb*inpol** 寄存器来实现输入信号取反。接口滤波功能将小于寄存器配置值宽度的输入脉冲信号在输入接口进行过滤，通过 **cfg_inxb_flt*en** 配置对应输入信号的滤波使能，通过 **cfg_inxb_flt_wid***寄存器配置滤波窗口宽度，滤波窗口支持 0~1023 个 XBAR 工作时钟周期。

输入信号经过 Input XBAR 预处理后，会基于输出处理通道进行选择，每个处理通道可以选择 84 个 GPIO 输入信号中的其中一个输入信号或多个输入信号的或逻辑组合作为输出。

说明

选择器由 21 个 4 选一选通器、21 个与开关和或逻辑门组成，因此无法选择同一个 4 选一选通器对应的输入信号进行或逻辑组合。在该处理过程中，上述 4 选一选通器通过 **cfg_inxb*mux** 寄存器配置，与开关通过 **cfg_inxb*enb** 寄存器配置。

输入信号经过 Input XBAR 选择器后，进入输出通道，Input XBAR 包含 16 个输出通道，

每个输出通道输出 1bit 信号，直接送往后级 SRPWM、ETIMER、SARC 等模块，或通过内部其他 XBAR 和 CLU 模块后送往后级模块。信号互联路径如下图所示。

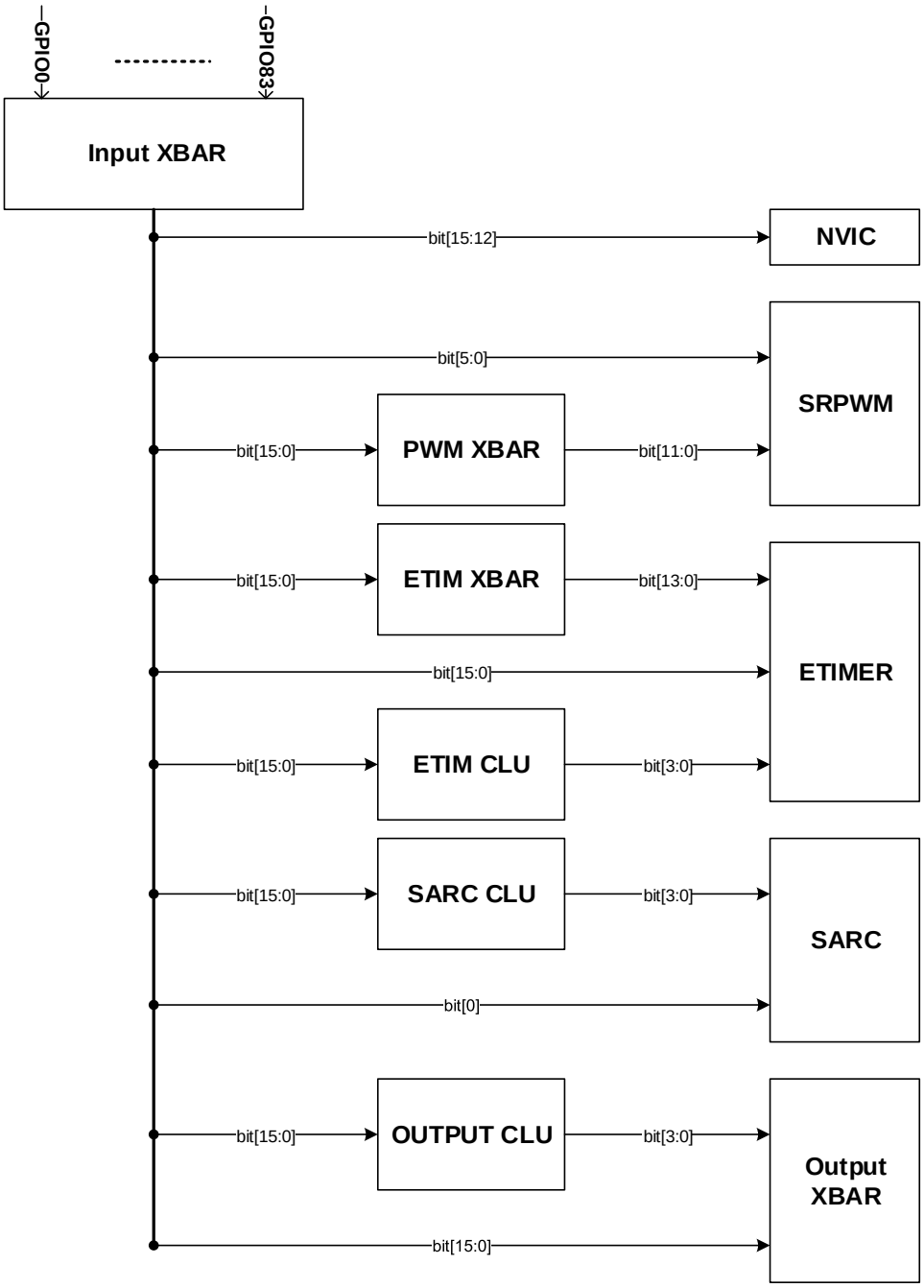


图25-3 Input XBAR 输出信号互联关系图

每个输出通道包含高电平锁存，取沿，输出开关和输出极性配置。通过配置 **cfg_inxb_dsel** 寄存器来选择输出锁存、取沿或原始信号输出，通过配置 **cfg_inxb_lat_clr** 寄存器来对高电平锁存后的寄存器进行清除，通过配置 **cfg_inxb_oe** 输出使能寄存器对输出信号进行控制，通过配置 **cfg_inxb_outpol** 寄存器来调整输出信号极性。**cfg_inxb_oe 和 cfg_inxb_outpol** 寄存器结合可实现软件控制 Input XBAR 输出电平。

25.3.5 CLU

XBAR 中包含三组 CLU 处理模块，根据连接关系不同，分为 XCSA、XCET 和 XCOX，三个模块独立配置，实现完全一致。主要完成简单逻辑组合实现和选择，以及中断上报功能。具体实现结构如下图所示。

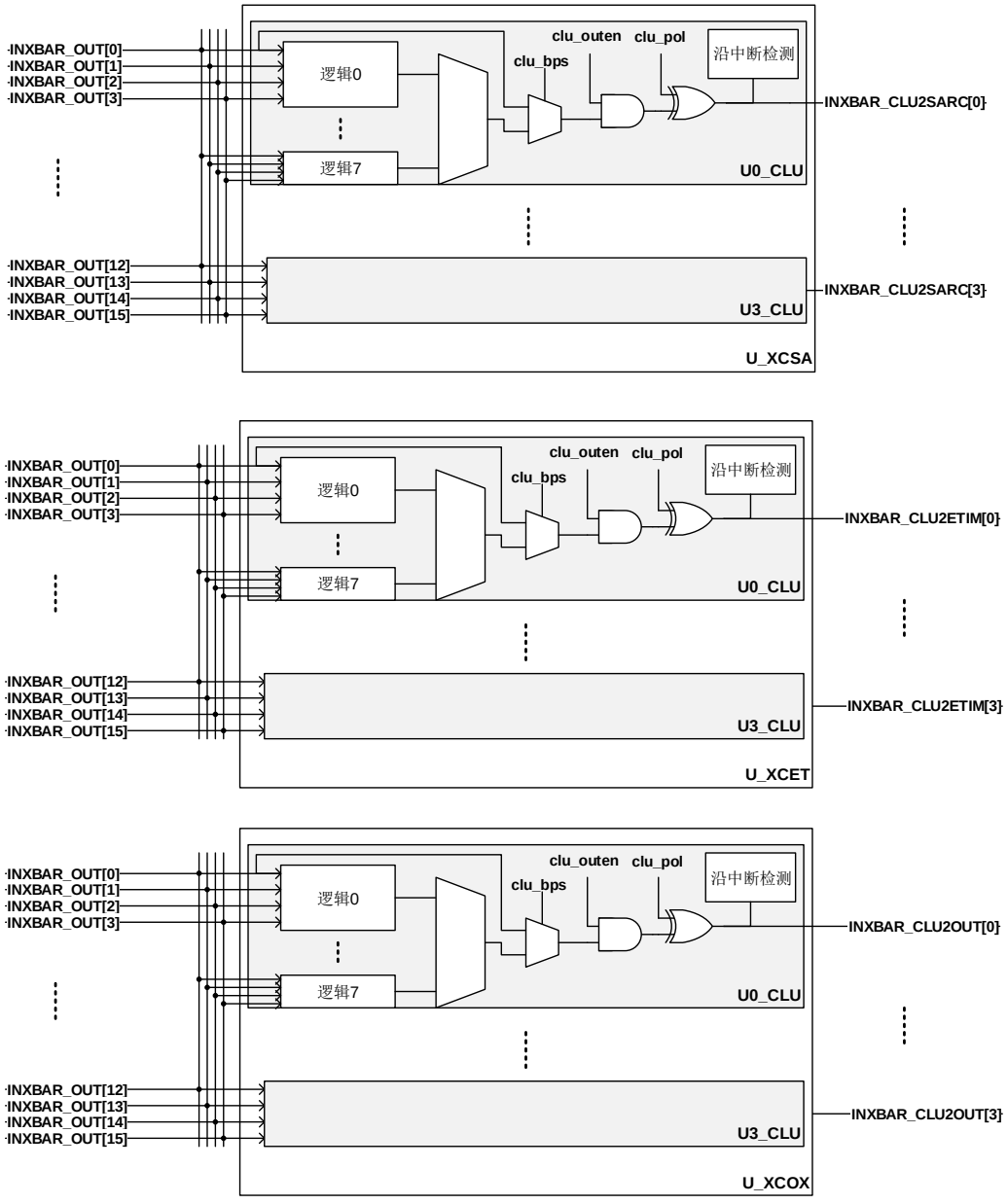


图25-4 XCSA、XCET 和 XCOX 模块框图

CLU 实现对 Input XBAR 信号的逻辑组合，其输入为 Input XBAR 模块 16bit 输出，输出分别为 4bit 信号逻辑组合信号，通过各自内置 4 个 4 输入 1 输出的 CLU 模块实现。XCSA、XCET 和 XCOX 模块输出分别送往 SAXB、EIXB 和 OPXB 模块作为后者输入。

CLU 逻辑处理模块支持 8 种逻辑处理组合，通过 **cfg_xc*_mode** 寄存器进行配置，包括：

- 逻辑 0：AND-OR
- 逻辑 1：OR-XOR
- 逻辑 2：4 输入 AND
- 逻辑 3：S-R 锁存器
- 逻辑 4：带置 1 和复位功能的 D 触发器
- 逻辑 5：带复位功能的 D 触发器
- 逻辑 6：带复位功能的 J-K 触发器
- 逻辑 7：带置 1 和复位功能的透明锁存器

其中，D 触发器、J-K 触发器和锁存器的相关逻辑采用基于工作时钟的寄存器时序逻辑进行实现。

CLU 模块支持输出旁路，通过 **cfg_xc*_lcbps** 寄存器进行配置，在旁路模式下，CLU 固定选择输入 4bit 分组中的最低位输出。

CLU 模块支持通过 **cfg_xc*_lcen** 寄存器配置输出使能，通过 **cfg_xc*_lcpol** 寄存器配置输出极性，二者结合可实现软件配置输出电平。

CLU 模块支持对输出信号进行信号沿检测，并产生中断事件脉冲信号。可通过 **cfg_xc*_intedg_sel** 寄存器配置，检测信号上升沿，下降沿，或同时检测上升沿和下降沿。

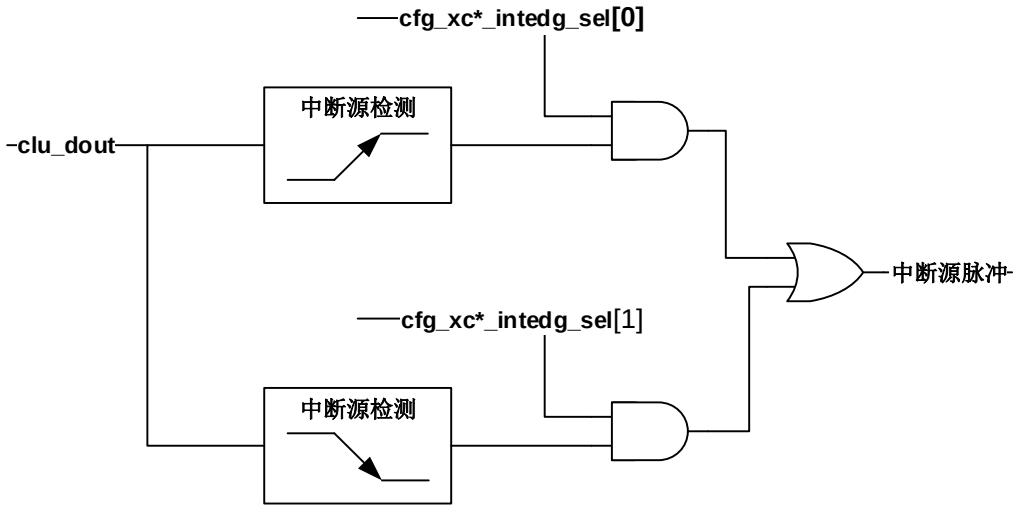


图25-5 CLU 模块中断源产生

25.3.6 PWM XBAR

PWM XBAR (PFXB) 处理方式与 Input XBAR 类似，区别在于输入信号直接进入选择器处理，以及输入和输出的信号数量和连接关系不同。

PWM XBAR 输出 12bit，选择后分别连接到 12 个 SRPWM 通道。

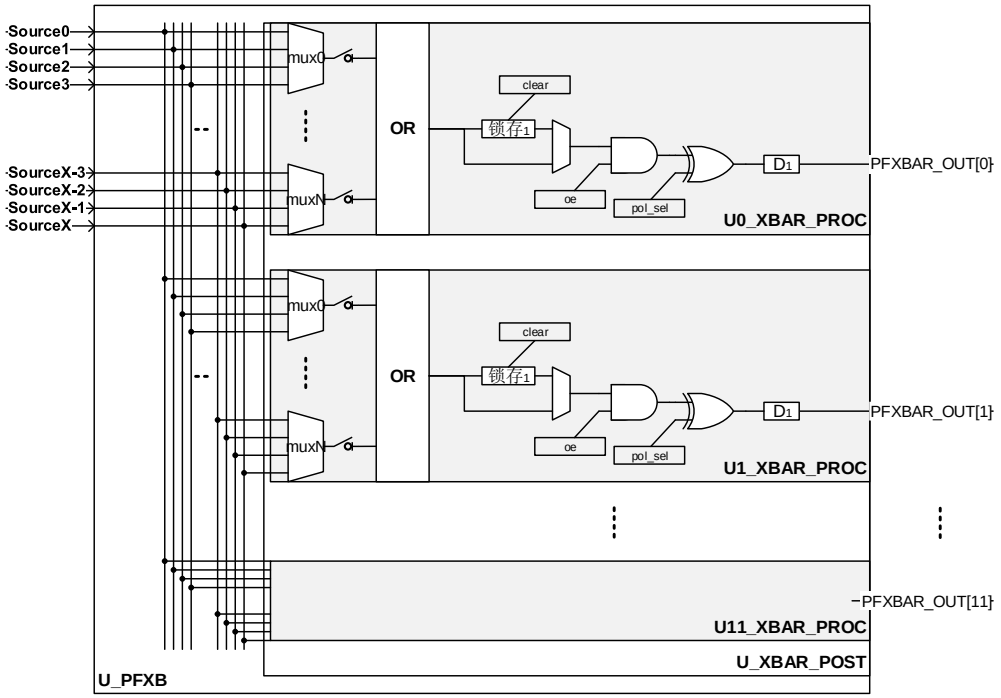


图25-6 PWM XBAR 模块框图

PWM XBAR 信号源选择关系如下表所示。

表25-2 PWM XBAR 信号源选择列表

MUX	0	1	2	3
0	CMP_EVT0	CMP_EVT0_OR_EVT1	ADCA_EVT0	Reserved
1	CMP_EVT1	INPUTXBAR0	ADCA_EVT1	ETIMOUT0
2	CMP_EVT2	CMP_EVT2_OR_EVT3	ADCA_EVT2	Reserved
3	CMP_EVT3	INPUTXBAR1	ADCA_EVT3	ETIMOUT1
4	CMP_EVT4	CMP_EVT4_OR_EVT5	ADCB_EVT0	Reserved
5	CMP_EVT5	INPUTXBAR2	ADCB_EVT1	ETIMOUT2
6	CMP_EVT6	CMP_EVT6_OR_EVT7	ADCB_EVT2	Reserved
7	CMP_EVT7	INPUTXBAR3	ADCB_EVT3	ETIMOUT3
8	Reserved	Reserved	ADCC_EVT0	Reserved
9	Reserved	INPUTXBAR4	ADCC_EVT1	ETIMOUT4
10	Reserved	ADCSOCAO	ADCC_EVT2	Reserved
11	Reserved	INPUTXBAR5	ADCC_EVT3	Reserved
12	ERRORSTS	Reserved	FLASH_ERR	Reserved
13	EXTSYNCOUT	INPUTXBAR6	CPU_HALT	Reserved
14	Reserved	ADCSOCBO	SYS_ERR	Reserved
15	Reserved	INPUTXBAR7	PT_ERR	ETIMOUT5
16	SD0FLT0_EVT0	SD0FLT0_EVT0_OR_EVT1	Reserved	Reserved
17	SD0FLT0_EVT1	INPUTXBAR8	Reserved	ETIMOUT6
18	SD0FLT1_EVT0	SD0FLT1_EVT0_OR_EVT1	Reserved	Reserved
19	SD0FLT1_EVT1	INPUTXBAR9	Reserved	ETIMOUT7

MUX	0	1	2	3
20	SD0FLT2_EVT0	SD0FLT2_EVT0_OR_EVT1	Reserved	Reserved
21	SD0FLT2_EVT1	INPUTXBAR10	Reserved	ETIMOUT8
22	SD0FLT3_EVT0	SD0FLT3_EVT0_OR_EVT1	Reserved	Reserved
23	SD0FLT3_EVT1	INPUTXBAR11	Reserved	ETIMOUT9
24	SD1FLT0_EVT0	SD1FLT0_EVT0_OR_EVT1	SRPWM_XBAR_SYNC0	Reserved
25	SD1FLT0_EVT1	INPUTXBAR12	Reserved	ETIMOUT10
26	SD1FLT1_EVT0	SD1FLT1_EVT0_OR_EVT1	SRPWM_XBAR_SYNC1	Reserved
27	SD1FLT1_EVT1	INPUTXBAR13	Reserved	ETIMOUT11
28	SD1FLT2_EVT0	SD1FLT2_EVT0_OR_EVT1	SRPWM_XBAR_SYNC2	Reserved
29	SD1FLT2_EVT1	INPUTXBAR14	Reserved	ETIMOUT12
30	SD1FLT3_EVT0	SD1FLT3_EVT0_OR_EVT1	SRPWM_XBAR_SYNC3	Reserved
31	SD1FLT3_EVT1	INPUTXBAR15	ERRORSTS	ETIMOUT13

25.3.7 ETIMER XBAR

ETIMER XBAR (ETXB) 处理方式与 PFXB 类似，区别仅在于信号选择不同。

ETIMER XBAR 信号源选择关系如下表所示。

表25-3 ETIMER XBAR 信号源选择列表

MUX	0	1	2	3
0	CMP_EVT0	CMP_EVT0_OR_EVT1	ADCA_EVT0	Reserved
1	CMP_EVT1	INPUTXBAR0	ADCA_EVT1	Reserved
2	CMP_EVT2	CMP_EVT2_OR_EVT3	ADCA_EVT2	Reserved
3	CMP_EVT3	INPUTXBAR1	ADCA_EVT3	Reserved
4	CMP_EVT4	CMP_EVT4_OR_EVT5	ADCB_EVT0	Reserved
5	CMP_EVT5	INPUTXBAR2	ADCB_EVT1	Reserved
6	CMP_EVT6	CMP_EVT6_OR_EVT7	ADCB_EVT2	Reserved
7	CMP_EVT7	INPUTXBAR3	ADCB_EVT3	Reserved
8	Reserved	Reserved	ADCC_EVT0	Reserved
9	Reserved	INPUTXBAR4	ADCC_EVT1	Reserved
10	Reserved	Reserved	ADCC_EVT2	Reserved
11	Reserved	INPUTXBAR5	ADCC_EVT3	Reserved
12	ERRORSTS	Reserved	FLASH_ERR	Reserved
13	EXTSYNCOUT	INPUTXBAR6	CPU_HALT	Reserved

MUX	0	1	2	3
14	Reserved	CFG_ETXB_SWx	SYS_ERR	Reserved
15	Reserved	INPUTXBAR7	PT_ERR	Reserved
16	SD0FLT0_EVT0	SD0FLT0_EVT0_OR_EVT1	Reserved	Reserved
17	SD0FLT0_EVT1	INPUTXBAR8	Reserved	Reserved
18	SD0FLT1_EVT0	SD0FLT1_EVT0_OR_EVT1	Reserved	Reserved
19	SD0FLT1_EVT1	INPUTXBAR9	Reserved	Reserved
20	SD0FLT2_EVT0	SD0FLT2_EVT0_OR_EVT1	Reserved	Reserved
21	SD0FLT2_EVT1	INPUTXBAR10	Reserved	Reserved
22	SD0FLT3_EVT0	SD0FLT3_EVT0_OR_EVT1	Reserved	Reserved
23	SD0FLT3_EVT1	INPUTXBAR11	Reserved	Reserved
24	SD1FLT0_EVT0	SD1FLT0_EVT0_OR_EVT1	Reserved	Reserved
25	SD1FLT0_EVT1	INPUTXBAR12	Reserved	Reserved
26	SD1FLT1_EVT0	SD1FLT1_EVT0_OR_EVT1	Reserved	Reserved
27	SD1FLT1_EVT1	INPUTXBAR13	Reserved	Reserved
28	SD1FLT2_EVT0	SD1FLT2_EVT0_OR_EVT1	Reserved	Reserved
29	SD1FLT2_EVT1	INPUTXBAR14	Reserved	Reserved
30	SD1FLT3_EVT0	SD1FLT3_EVT0_OR_EVT1	CFG_ETXB_SWx	Reserved
31	SD1FLT3_EVT1	INPUTXBAR15	ERRORSTS	Reserved

ETIMER XBAR 输出 14bit，分别对应 14 个 ETIMER 输出通道。

ETIMER XBAR 支持软件可配置 14bit CFG_ETXB_SWx 寄存器 **(cfg_etxb_sw)**，分别对应 14 个 ETIMER 通道。

25.3.8 Output XBAR

Output XBAR (OPXB) 处理方式与 PFXB 类似，区别仅在于信号选择不同。

Output XBAR 信号源选择关系如下表所示。

表25-4 Output XBAR 信号源选择列表

MUX	0	1	2	3
0	CMP_OUT0	CMP_OUT0_OR_OUT1	ADCA_EVT0	Reserved
1	CMP_OUT1	INPUTXBAR0	ADCA_EVT1	ETIMOUT0
2	CMP_OUT2	CMP_OUT2_OR_OUT3	ADCA_EVT2	Reserved
3	CMP_OUT3	INPUTXBAR1	ADCA_EVT3	ETIMOUT1
4	CMP_OUT4	CMP_OUT4_OR_OUT5	ADCB_EVT0	Reserved
5	CMP_OUT5	INPUTXBAR2	ADCB_EVT1	ETIMOUT2
6	CMP_OUT6	CMP_OUT6_OR_OUT7	ADCB_EVT2	Reserved
7	CMP_OUT7	INPUTXBAR3	ADCB_EVT3	ETIMOUT3
8	Reserved	CFG_OPXB_SWx	ADCC_EVT0	Reserved
9	Reserved	INPUTXBAR4	ADCC_EVT1	ETIMOUT4
10	Reserved	ADCSOCAO	ADCC_EVT2	Reserved

MUX	0	1	2	3
11	Reserved	INPUTXBAR5	ADCC_EVT3	Reserved
12	ERRORSTS	CFG_OPXB_SWx	FLASH_ERR	Reserved
13	EXTSYNCOUT	INPUTXBAR6	CPU_HALT	Reserved
14	Reserved	ADCSOCBO	SYS_ERR	Reserved
15	Reserved	INPUTXBAR7	PT_ERR	ETIMOUT5
16	SD0FLT0_EVT0	SD0FLT0_EVT0_OR_EVT1	STM0_OC0	Reserved
17	SD0FLT0_EVT1	INPUTXBAR8	STM0_OC1	ETIMOUT6
18	SD0FLT1_EVT0	SD0FLT1_EVT0_OR_EVT1	STM0_OC2	Reserved
19	SD0FLT1_EVT1	INPUTXBAR9	STM0_OC3	ETIMOUT7
20	SD0FLT2_EVT0	SD0FLT2_EVT0_OR_EVT1	STM1_OC0	Reserved
21	SD0FLT2_EVT1	INPUTXBAR10	STM1_OC1	ETIMOUT8
22	SD0FLT3_EVT0	SD0FLT3_EVT0_OR_EVT1	STM1_OC2	Reserved
23	SD0FLT3_EVT1	INPUTXBAR11	STM1_OC3	ETIMOUT9
24	SD1FLT0_EVT0	SD1FLT0_EVT0_OR_EVT1	SRPWM_XBAR_SYNC0	INPUTXBAR_CLU2OUT[0]
25	SD1FLT0_EVT1	INPUTXBAR12	Reserved	ETIMOUT10
26	SD1FLT1_EVT0	SD1FLT1_EVT0_OR_EVT1	SRPWM_XBAR_SYNC1	INPUTXBAR_CLU2OUT[1]
27	SD1FLT1_EVT1	INPUTXBAR13	Reserved	ETIMOUT11
28	SD1FLT2_EVT0	SD1FLT2_EVT0_OR_EVT1	SRPWM_XBAR_SYNC2	INPUTXBAR_CLU2OUT[2]
29	SD1FLT2_EVT1	INPUTXBAR14	Reserved	ETIMOUT12
30	SD1FLT3_EVT0	SD1FLT3_EVT0_OR_EVT1	SRPWM_XBAR_SYNC3	INPUTXBAR_CLU2OUT[3]
31	SD1FLT3_EVT1	INPUTXBAR15	ERRORSTS	ETIMOUT13

Output XBAR 输出 14bit，分别对应 14 个 Output XBAR 输出通道。

Output XBAR 支持软件可配置 14bit CFG_OPXB_SWx 寄存器 **(cfg_opxb_sw)**，分别对应 14 个 Output XBAR 通道。

Output XBAR 支持输出模式选择，通过 **cfg_opxb_oemod** 寄存器配置，选择到 ETIMOUTx 信号时，输出对应的 OEN 信号，也可配置固定输出模式，和固定输出三态模式。

Output XBAR 还支持输出信号展宽，固定展宽 16 拍，是否展宽可通过 **cfg_opxb_exp_en** 寄存器配置。

25.4 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 XBAR 章节。

26 看门狗 (Watchdog)

26.1 简介

ET6002 内部包含两个 Watchdog，分别用于 CPU、系统看门狗，支持窗口、非窗口 2 种模式。

26.2 结构框图

Watchdog 框图如下。

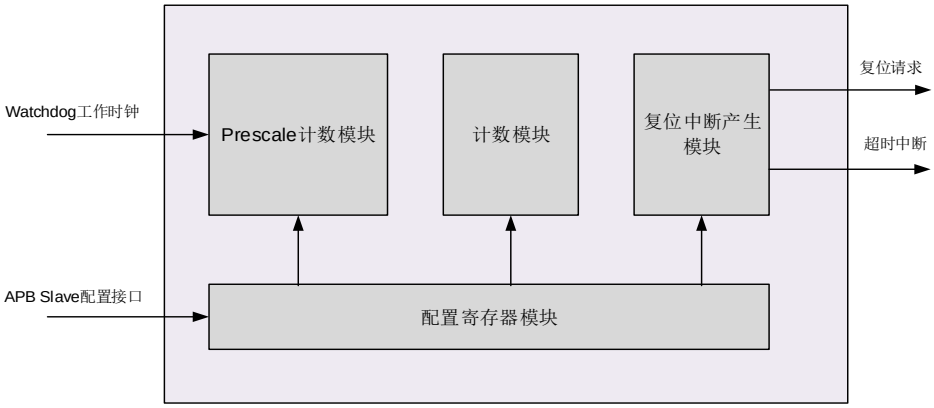


图26-1 Watchdog 框图

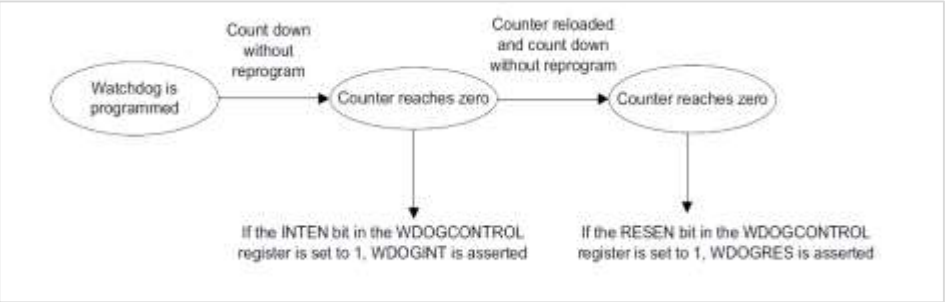
26.3 主要特性

- Watchdog0/1 输入计数时钟：启动时是基于内置 25 MHz 时钟进行计数，后续可由软件修改为基于 100 MHz 时钟，支持 CRG 给的 clk_en 计数使能，可配置为 1-7 分频。
- Watchdog0/1 如果发生软件喂狗超时，可以触发系统复位操作；
- Watchdog0/1 内部计数位宽为 32bit；
- 每个 Watchdog 都支持根据 CPU 是否进入仿真器调试状态进行暂停计数；
- Watchdog0/1 超时可触发 NMI 中断；

26.4 功能描述

26.4.1 中断和复位产生机制

Watchdog 可以工作在直接产生复位信号，或者先产生超时中断，再产生复位信号。



26.4.2 Debug 模式下停止计数

CPU 处于 Debug 状态时，通过 SYSC 中的寄存器，可以控制 Watchdog0，Watchdog1 是否暂停计数；**cfg_sysc_wdg1_halt_mode_en，cfg_sysc_wdg0_halt_mode_en** 对应的信号为 1 时，当 CPU0 或者 CPU1 连接仿真器进入 Debug 状态时，停止计数。

1.19 CFG_WDT_HALT_MOD_EN		CFG_WDT_HALT_MOD_EN		Reg.	0x118
offset		external		default	0x00000000

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

Bits	Name	S/W	H/W	Default	Description
1	cfg_sysc_wdg1_halt_mode_en	rw	ro	0x0	看门狗 1 halt 模式使能
0	cfg_sysc_wdg0_halt_mode_en	rw	ro	0x0	看门狗 0 halt 模式使能

26.4.3 CPU NMI 中断源控制

NMI 中断在 CPU_CFG 模块里可以实现可配置，能够产生 NMI 中断源只有 WDG0，WDG1 的超时中断。

CPU 的 NMI 中断可以通过 CPU_CFG 模块的寄存器 **CFG_M7_NMI_SEL** 进行控制，注意配置该寄存器需要先配置 CPU_CFG 模块 **CFG_M7_KEY_CTRL.sw_access_key** 寄存器配置为 **0x7879a8a9**。

配置参数 **cfg_m7_nmi_mask[3:0]**实际只有低 2 位有效，bit1 对应 WDG1 的超时中断，bit0 对应 WDG0 的超时中断，1 表示屏蔽，0 表示允许上报。

1.11 CFG_M7_NMI_SEL		CFG_M7_NMI_SEL		Reg.	0x28
offset		external		default	0x0000000f
{lock=CFG_M7_KEY_CTRL.sw_access_key!=32'h7879a8a9}					

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

Bits	Name	S/W	H/W	Default	Description
11:8	nmi_src_rpt	ro	wo	0x0	当前屏蔽前的四路 NMI 中断源状态信号
7:4	nmi_src_masked_rpt	ro	wo	0x0	当前屏蔽后的四路 NMI 中断源状态信号
3:0	cfg_m7_nmi_mask	rw	ro	0xf	**sw_access_key** 设置为 **0x7879a8a9**，才能更新该寄存器 M7 的 NMI 中断源屏蔽控制信号

26.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 Watchdog 章节。

27 加解密模块（AES）

27.1 简介

加解密模块采用了 AES 硬件加速器来实现数据的加解密操作。AES 是一个对称密码模块，意味着加密密钥和解密密钥是完全相同的，在 ET6002 中支持 128/256 bit 密钥。

27.2 结构框图

ET6002 AES 交互模块主要包含以下功能单元:

- 系统控制单元（SYSC)；
- DMA 请求信号选择处理单元（DMA_MUX)；
- 中断控制器单元（INTC)；

AES 整体结构框图如下：

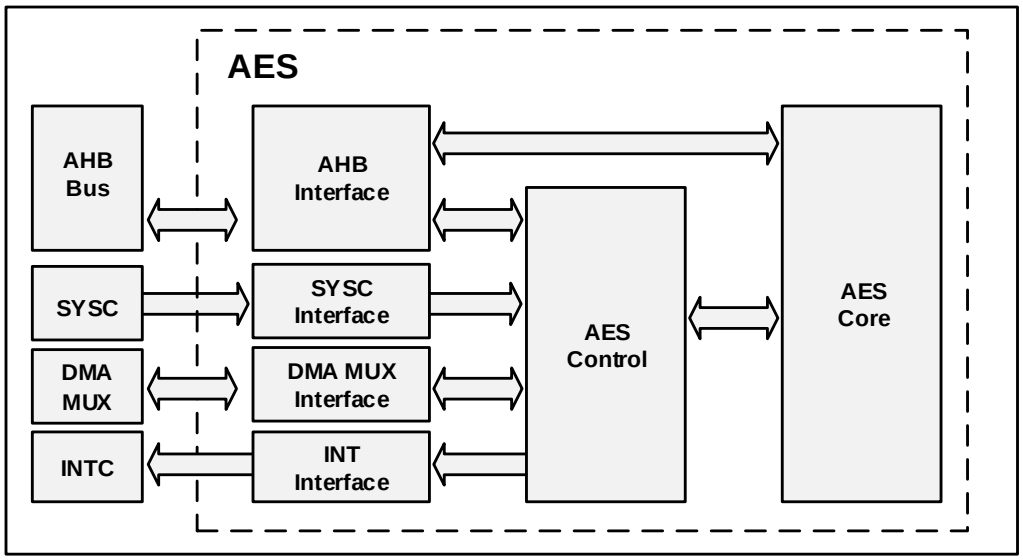


图27-1 AES 整体结构框图

27.3 主要特性

AES 硬件加速器，支持特性如下：

- 支持 128，256 bit 密钥长度；
- 支持 ECB、CBC、CTR 加解密模式；
- 支持 CBC 模式下 128 bit 初始向量，CTR 模式下 96 bit 初始向量和 32 bit 初始计数器值配置；

- 支持 Zeros，PKCS#7 数据填充；
- 支持轮询、中断、DMA 方式完成数据的传入和传出；
- 支持硬件使能控制 AES 工作状态；
- 内置轮密钥调度器，动态生成下一轮轮密钥；

27.4 功能描述

27.4.1 AES 内部信号

表27-1 AES 内部输入/输出信号

内部信号名称	信号类型	说明
aes_hclk	输入	AES 总线时钟
aes_ work_hard_en	输入	外部输入 AES 模块硬件使能信号，高电平时有效
aes_int	输出	AES 中断请求输出
aes_wr_dma_busy	输入	AES DMA 写通道 busy 返回信号
aes_rd_dma_busy	输入	AES DMA 读通道 busy 返回信号
aes_dma_ack[1:0]	输入	2 个 DMA 通道的 ACK 应答
aes_dma_req[1:0]	输出	2 个 DMA 通道的 req 请求
aes_dma_single[1:0]	输出	2 个 DMA 通道的 single 请求

27.4.2 AES 加解密核心

AES 根据寄存器配置对数据进行加密、解密操作，其核心由以下部分组成。

27.4.2.1 轮密钥调度器

轮密钥调度器主要用于生成加解密过程中的子轮密钥。在每一轮的迭代过程中，调度器会根据输入的密钥，动态扩展生成当前轮次的子轮密钥，与数据并行处理，用于与数据的轮密钥加处理。不仅减少了轮密钥的存储器需求，同时也避免了子轮密钥的残留，提升安全性。

27.4.2.2 加密核心块

加密核心块使用了符合 FIPS PUB 197 中定义的 AES 算法，实现对数据的加密操作。加密核心块对输入的数据按照 128/256 bit 密钥位宽迭代 10/14 轮，每轮迭代涉及字节代换，行移位，列混合，轮密钥加等操作。其中字节代换操作涉及 S-Box 盒的使用，轮密钥加则是从轮密钥调度器中获得响应的轮密钥。

27.4.2.3 解密核心块

解密核心块是加密核心块的反向操作过程，迭代轮数与加密轮数保持一致。进行轮密钥加，逆行移位，逆字节代换，逆列混合等操作，其中逆字节代换涉及逆 S-Box 盒的使用，同时，解密核心块的轮密钥加操作也是逆向的，所以解密核心块第一轮迭代过程中使用的轮密

钥是加密核心块迭代过程中最后一轮的子轮密钥，以此类推。

27.4.2.4 参数反馈/链接逻辑

不同于 ECB 加密模式，CBC 加密模式下，需要将当前数据块的密文反馈作为下一数据块的初始向量进行轮密钥加操作，由此将整个加密数据链接起来，不会出现相同明文块加密后出现相同的密文块情况。CTR 模式下，需要将当前数据块的计数器值加一作为下一数据块的计数器值，链接整个数据的加密操作。

27.4.3 AES 工作模式

27.4.3.1 ECB 模式

如图 27-2 所示，显示了 ECB 模式下，加解密的流程示意图。初始密钥会输入到轮密钥调度器中。数据则会根据加解密模式，进入加密核心或者解密核心，在进行 10/14 轮迭代后，获得密文、明文。

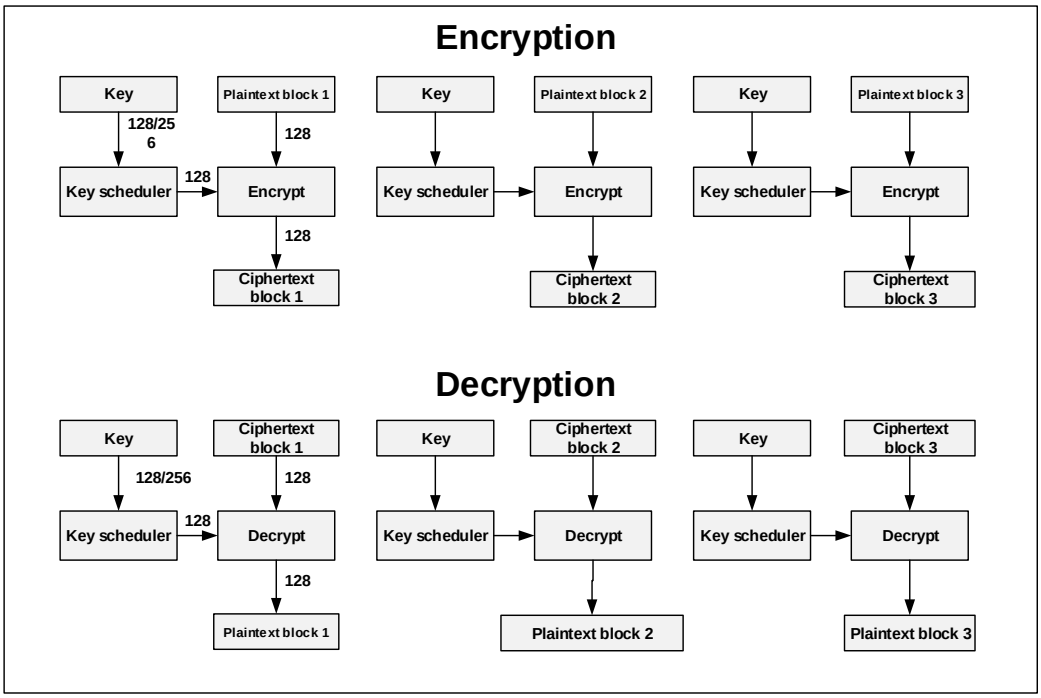


图27-2 ECB 模式加解密流程示意图

27.4.3.2 CBC 模式

如图 27-3 所示，显示了 CBC 模式下，加解密的流程示意图。相比于 ECB 模式，CBC 增添了 128bit 的初始向量输入，在明文块进入加密核心块之前，会与初始向量进行轮密钥加操作，除了第一个明文块是与软件配置的初始向量进行轮密钥加操作外，后续的数据块都是将前一个数据块的密文作为初始向量进行轮密钥加。

解密过程则是加密过程的反向操作，同时需要将当前的密文块进行缓存，用作下一个密文块的初始向量。

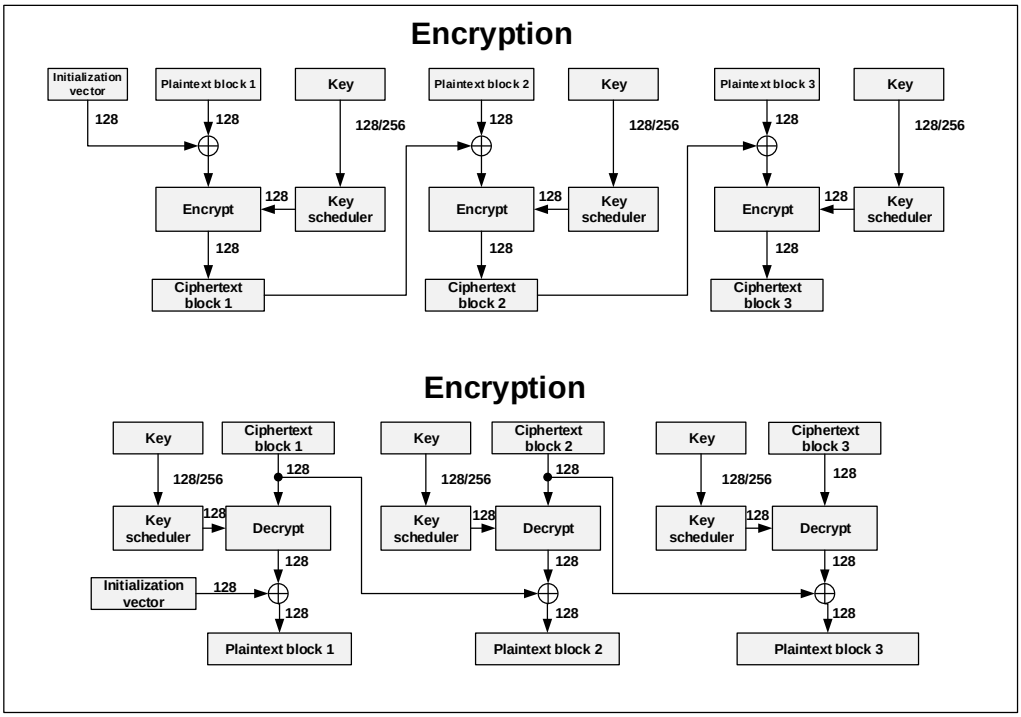


图27-3 CBC 模式加解密流程示意图

27.4.3.3 CTR 模式

如图 27-4 所示，显示了 CTR 模式下，加解密的流程示意图。相比于 ECB 模式，CTR 增添了 96 bit 的初始向量，以及 32 bit 的计数器值输入，拼接成 128 bit 数据。但是 CTR 加解密模式都是将拼接成的 128 bit 数据输入到加密核心块处理，然后将得到的数据与待加密/解密的数据块进行轮密钥加操作。在整个加解密过程中，计数器值随着数据块递增。

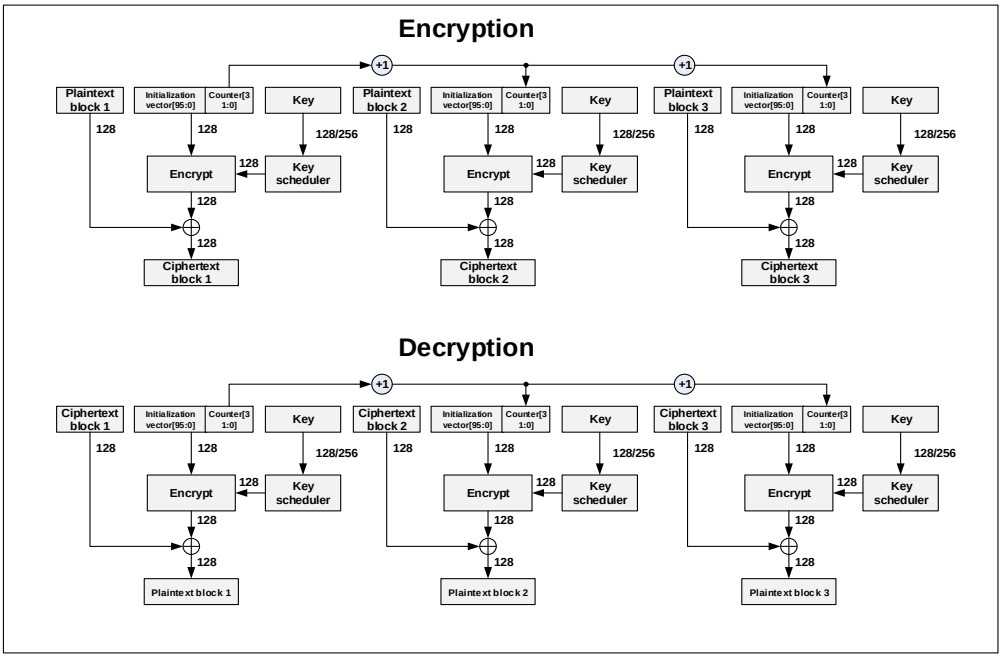


图27-4 CTR 模式加解密流程示意图

27.4.4 AES 数据填充

在 AES 内部支持 Zeros 和 PKCS#7 两种数据填充模式，仅在 ECB 和 CBC 模式下需要进行填充，CTR 模式下，是对初始向量和计数器值的加密，不需要进行填充。

27.4.4.1 Zeros 填充模式

在 AES 加密过程中，会将数据划分成 128 bit (16 bytes) 的数据块进行处理，当最后数据不满足 16 bytes 时，AES 会将数据进行填充成完整的数据块，再进行加密操作。因为输入的数据是字节对齐的，所以不满足 16 bytes 时，缺少的也是整数倍的字节块。Zeros 填充模式就是将缺少的字节块全部填充 **0x00**，然后进行加密操作。如图 27-5 所示，当数据块差 6 个字节为完整的数据块时，AES 会在数据后面填充 6 个字节的 **0x00**。而当数据为完整 16 bytes 数据块时，AES 会多产生一个完整 16 bytes 的数据块，每个字节全是 **0x00**，进行加密。

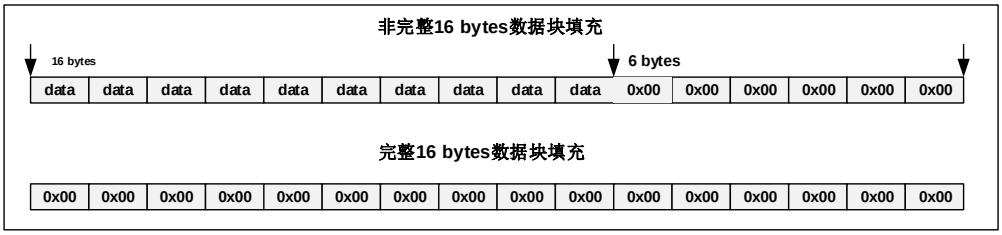


图27-5 Zeros 填充示意图

27.4.4.2 PKCS#7 填充模式

类似于 Zeros 填充模式，PKCS#7 填充模式是将缺少的字节块全部填充缺少的字节数，然后进行加密操作。如图 27-6 所示，当数据块差 6 个字节为完整的数据块时，AES 会在数据后面填充 6 个字节的 **0x06**。而当数据为完整 16 bytes 数据块时，AES 会多产生一个完整 16 bytes 的数据块，每个字节全是 **0x10**，进行加密。

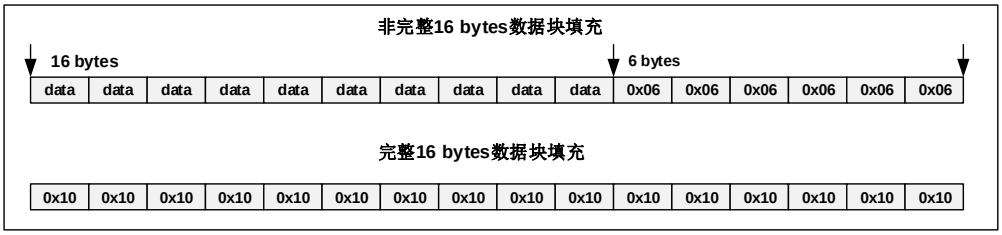


图27-6 Pkcs#7 填充示意图

27.4.5 AES 数据输入/输出方式

AES 支持轮询、中断、DMA 三种方式完成对数据的传输，在此之前需要完成对 AES 的初始化配置。

1. 对模块使能，密钥大小，加解密方向，填充模式进行配置；
2. 对需要加解密的数据长度进行配置；
3. 配置加解密的模式；

4. 配置加解密密钥，初始向量；
5. 配置 CTR 加解密的初始计数器值，仅 CTR 模式进行配置 (必须)；
6. 中断使能配置；
7. DMA 请求使能配置 (DMA 场景下，非 DMA 场景不用配置)；
8. 响应数据输入请求；

完成以上配置过后，开始对 AES 的数据输入/输出请求进行处理。

27.4.5.1 轮询模式

软件定时访问上报寄存器状态，进行 AES 的数据输入/输出请求作出处理：

- 通过 **RPT_AES_STS.din_flg** 寄存器对 AES 的输入数据请求进行处理，当寄存器拉高时有效；
- 通过 **RPT_AES_STS.dout_flg** 寄存器对 AES 的输出数据请求进行处理，当寄存器拉高时有效；

27.4.5.2 中断模式

软件通过中断请求来应答 AES 的数据输入/输出请求，需要在初始化配置时，打开 AES 数据输入/输出请求中断使能，当 AES 发出中断请求过后，软件访问中断寄存器状态，完成数据输入/输出：

- 通过访问 **RPT_AES_INT_RAW.int_raw[5]**寄存器状态，对 AES 的输入数据请求进行处理，当寄存器拉高时有效；
- 通过访问 **RPT_AES_INT_RAW.int_raw[6]**寄存器状态，对 AES 的输出数据请求进行处理，当寄存器拉高时有效；

说明

中断模式将对每一个数据块产生输入中断请求，输出中断请求，当对大量数据进行加解密操作时，中断请求数量也会随着增加。

27.4.5.3 DMA 模式

AES 支持读写两个请求 DMA 通道，使用 DMA 模式时，初始化配置将中断模式的输入/输出请求中断使能关闭，打开 AES 的 DMA 请求使能配置。AES 数据输入/输出数据支持 BURST 传输，类型为 INCR 4。

AES 内部产生数据输入/输出 DMA 请求信号过后，输出到 DMA_MUX 模块，再由 DMA 控制器响应请求：

- aes_dma_req[0]，AES 数据输入 DMA 请求，高有效；
- aes_dma_req[1]，AES 数据输出 DMA 请求，高有效；

27.4.6 AES 事件和中断

AES 的中断产生主要包含以下事件：

- 在 DMA 模式下，上一数据块的加密/解密输出数据在下一数据块加密/解密数据到来时，数据还未被读走，上报中断；
- 待加密/解密数据数据块被输入到 AES 模块，而密钥没有配置完成的情况下，上报中断；
- CBC，CTR 模式下，待加密/解密数据数据块被输入到 AES 模块，而初始向量没有配置完成的情况下，上报中断；
- AES 整个加密/解密事件完成，上报中断；
- 数据传输在中断模式下，AES 内部产生输入数据请求，上报中断；
- 数据传输在中断模式下，AES 内部产生输出数据请求，上报中断；

27.4.7 AES 约束

AES 主要包含以下约束：

- 待加解密数据大小 **cfg_aes_data_size**$_x$ 和工作模式 **cfg_aes_work_mode** 优先密钥和初始向量配置，当数据长度小于等于 2^{32} **(cfg_aes_data_size0[31:0])** 字节时，也需要对 **cfg_aes_data_size1[3:0]**进行 4’d0 写入；
- AES 完成初始化配置过后，才能响应数据输入请求；
- 密钥 **cfg_aes_key**，初始向量 **cfg_aes_iv**，计数器值 **cfg_aes_block_cnt** 为静态参数，不能动态修改；
- AES 工作过程中不能拉低工作使能，一旦拉低，AES 内部会将轮密钥信息进行清除，终止事件；
- 支持 2^{36}-32 字节及以下大小数据加解密，超过字节需要拆分为多次事件；
- AES 数据输入/输出数据在 DMA 模式下，支持 Burst 传输，类型为 INCR 4；

27.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 AES 章节。

28 循环冗余校验模块 (CRC)

28.1 简介

CRC 模块用于 CRC 的计算与控制。CRC (Cyclic Redundancy Check) 循环冗余校验，主要用来检测或校验数据传输错误或者存储完整性，在数字电路中根据循环移位和异或完成。

本模块支持的 CRC 多项式类型有 Ethernet CRC32、CRC32P4、CCITT CRC16、SAE J1850 CRC8。CRC 采用单通道串行计算，多项式可软件配置，支持字节交换，输入数据字节内反转；支持输出数据反转，异或；支持数据校验，支持中断上报等功能。

28.2 结构框图

CRC 功能框图如下：

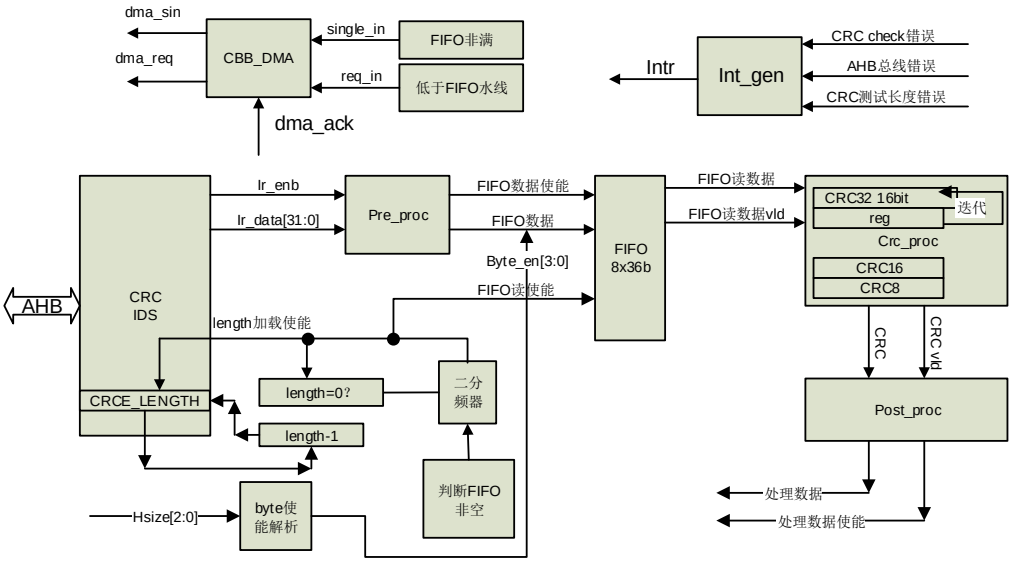


图28-1 CRC 功能框图

28.3 主要特性

CRC 主要特性如下：

- 支持任意字节的 CRC 计算和校验；
- 支持 CRC8、CRC16 和 CRC32，并且多项式可任意配置；
- 支持对输入数据字节交换、大小端转换、字节内反转；
- 支持 CRC 计算值按照寄存器配置是否反转，是否按照异或处理；

28.4 功能描述

28.4.1 输入数据处理

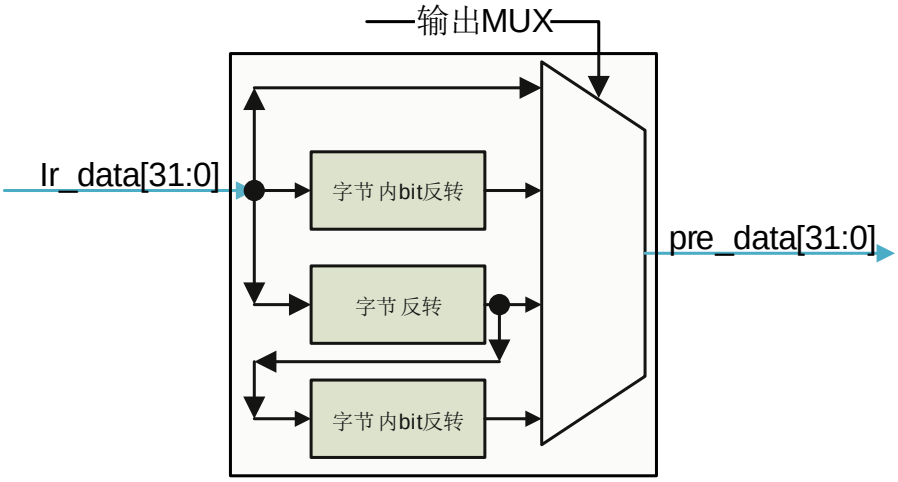


图28-2 数据前处理逻辑

将配置数据 ir_data，按照配置寄存器中字节交换和字节内反转的使能方式，两两组合一共有 4 种处理方式，统一按照 32 位进行处理。

由于 CRC32,CRC16,CRC8 的有效数据位不同，根据软件配置的 CRC 类型，从前处理得到的 32bit 数据中提取有效位，得到的数据送给 FIFO 缓存单元。FIFO 非空后数据读出来送往 CRC 计算单元。

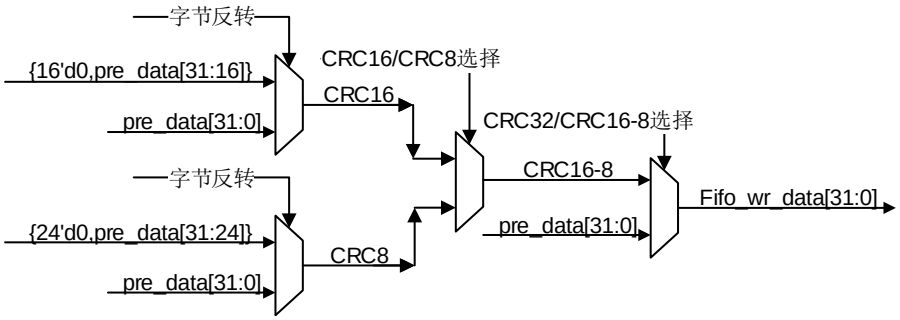


图28-3 根据配置的 CRC 类型处理有效输入数据

28.4.2 输出数据处理

根据软件配置 CRC 多项式类型得到的 CRC 计算值 crc_cal[31:0]，按照配置寄存器中位反转和异或选择的方式，两两组合一共有 4 种处理方式，统一按照 32 位进行处理。

字节内反转通过调用 bit_invert 模块 t；由于异或选择是与 0x00000000 和 0xffffffff 进行异或，直接对 crc_cal[31:0]进行取反即可。既进行按位反转又进行异或 (先处理按位反转，再处理异或)，将 crc_refout 再次调用 bit_invert 得到后处理数据送往寄存器。见下图所示：

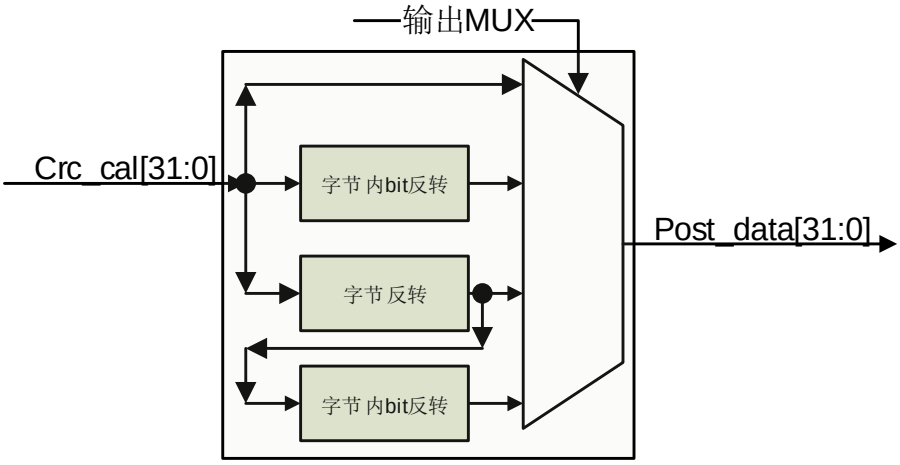


图28-4 输出数据反转，异或处理

由于 CRC32,CRC16,CRC8 的有效数据位不同，根据软件配置的 CRC 类型，从统一处理的 post_data [31:0]中提取有效位，得到 res_cal[31:0]送给 ids RES 寄存器。

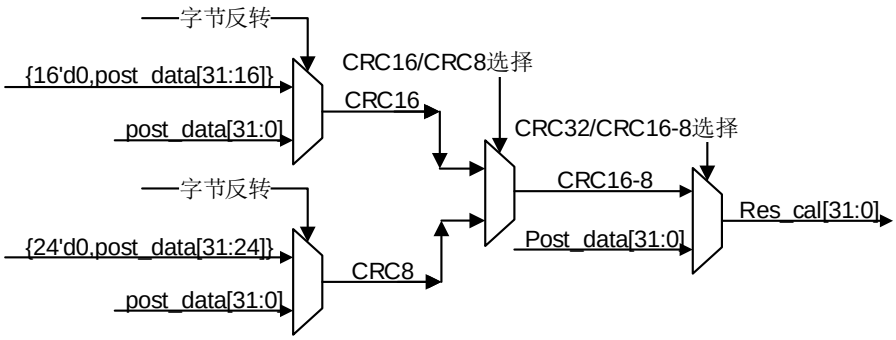


图28-5 根据配置的 CRC 类型处理有效输出数据

28.4.3 状态上报

- 总线上报错误状态 crc_bef_int_src

  当 CRC 类型配置为 CRC32 时，通过读取 AHB 总线的数据传输位 crc_hsize，当 crc_hsize 为 3'b000 时 (数据传输位为 8 位) 或者当 crc_hsize 为 3'b001 时 (数据传输为 16 位)，上报总线错误

  当 CRC 类型配置为 CRC16 时，通过读取 AHB 总线的数据传输位 crc_hsize，当 crc_hsize 为 3'b000 时 (数据传输位为 8 位)，上报总线错误错误。

  当 CRC 类型配置为 CRC8 时，由于 AHB 总线至少为传输 1 个字节，不存在输入数据的位宽小于 8 的情况，不上报总线错误。

- 匹配错误 crc_cmf_int_src

  当 crc_check_en 使能后，CRC block 计算完成时，若检测到 res_cal 与 crc_check 不相等时，crc_cmf_int 为 1，其余情况 crc_cmf_status 为 0。

- 长度错误 crc_lef_int_src

当 length 寄存器为 0 时，此时还有数据要进行 CRC 计算，会上报中断。

28.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 CRC 章节。

29 坐标旋转数字计算模块 (CORDIC)

29.1 简介

坐标旋转数字计算模块 (Coordinate Rotation Digital Computer) 提供了在电机控制、计量、信号处理和许多其他应用中经常使用的特定数学函数 (正弦、余弦、双曲正弦、双曲余弦、反正切 (atan/atan2)、反双曲正切、模值、平方根、自然对数函数) 的硬件加速。与软件实现相比，CORDIC 加速了这些函数的计算，使得可以使用更低的操作频率，或者释放处理器周期以执行其他任务。

29.2 结构框图

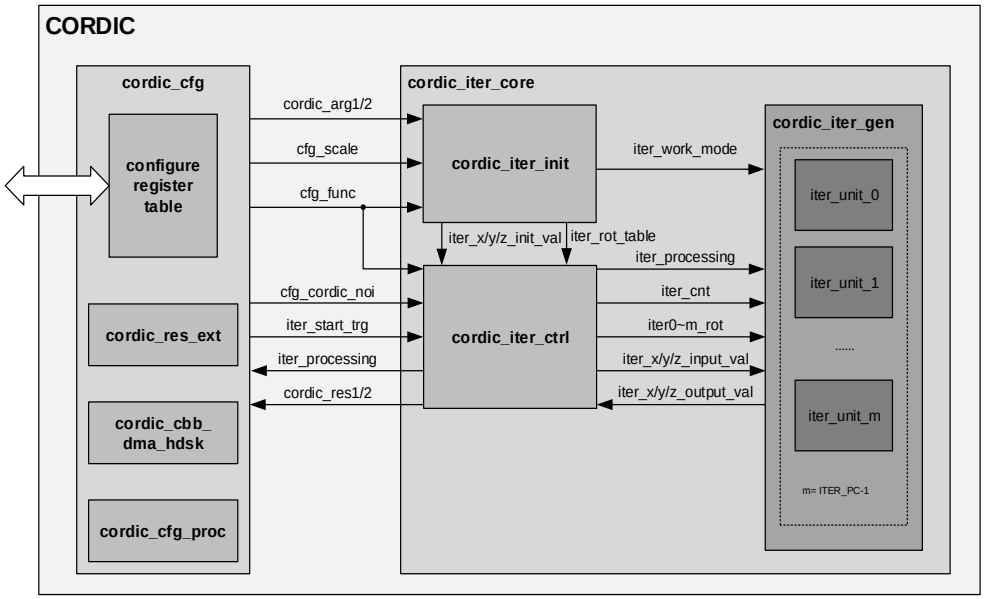


图29-1 CORDIC 模块整体结构图

29.3 主要特性

主要特性如下：

- 24bit 的 CORDIC 旋转引擎；
- 支持圆坐标系和双曲坐标系模式；
- 支持旋转和向量模式；
- 支持以下函数：正弦、余弦、双曲正弦、双曲余弦、反正切 (atan/atan2)、反双曲正

切、模值、平方根、自然对数；

- 支持可配置迭代精度；
- 每个时钟周期迭代 4 次；
- 支持 16 位和 32 位定点输入和输出格式；
- 支持轮询和中断上报；
- 支持零延迟模式，结果可以在准备好后立即读取，无需轮询或中断；
- 支持 DMA 读写；

29.4 功能描述

29.4.1 CORDIC 运算数据结构

CORDIC 计算采用定点带符号整数数据格式，输入输出采用 s(32,31) 或者 s(16,15) 数据格式。

在 s(32,31) 格式中，数字由一个符号位和 31 个小数位 (二进制) 表示；数值范围是 $[-1(0x80000000),1-2^{-31}(0x7FFFFFFF)]$；精度是$2^{-31}$(大约$5\times10^{-10}$)。

表29-1 s(32,31)数据格式

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
S	C_{30}	C_{29}	C_{28}	C_{27}	C_{26}	C_{25}	C_{24}	C_{23}	C_{22}	C_{21}	C_{20}	C_{19}	C_{18}	C_{17}	C_{16}
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C_{15}	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

在 s(16,15)格式中，数字由一个符号位和 15 个小数位 (二进制) 表示；数值范围是 $[-1(0x8000),1-2^{-15}(0x7FFF)]$；精度是$2^{-15}$(大约$3\times10^{-5}$)。

表29-2 s(16,15)数据格式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

这种格式的优点是可以将两个输入参数打包到一个 32 位寄存器中写入，并且可以在一个 32 位寄存器的读操作中获取两个结果；但精度降低到2^{-15}(大约3×10^{-5})。

29.4.2 CORDIC 运算比例系数

由 29.4.1 "CORDIC 运算数据结构"可知，CORDIC 运算要求输入数值在[-1,1](s(32,31)用 0x7FFFFFFF 表示+1, s(16,15)用 0x7FFF 表示+1)区间内，因此当函数输入在区间范围外，需要配置 CFG_CORDIC_CTRL0 寄存器对应的比例系数 cfg_cordic_scale==n，先对输入值 x 做运算$ARG=x\cdot2^{-n}$，使输入落入[-1,1]区间，同时 CORDIC 运算逻辑也会进行相同的移位处理(实际上是将定点数的格式进行变化，增加整数位，减少小数位)。CORDIC 计算完成后输出结果根据值域，同样进行了缩放，$RES=y\cdot2^{-m}$，需要乘以2^{-m}得到正确的结果。

29.4.3 CORDIC 输入变量

参见 29.4.6 "CORDIC 计算函数"章节表 29-3 的不同计算函数，除了双曲反正切、自然对数和平方根函数只需要一个输入变量，其余函数需要两个变量。

因此对于需要两个变量的函数，变量输入配置如下：

1. CFG_CORDIC_CTRL0 对应的输入格式选择寄存器 **cfg_cordic_argsize** 配置为 **1'b0** 时，输入格式为 16bit，此时高 16bit 为变量 2、低 16bit 为变量 1，配置一次 32bit CFG_CORDIC_CTRL2 的输入变量配置寄存器 **cfg_cordic_arg** 后开始迭代计算；
2. CFG_CORDIC_CTRL0 对应的输入格式选择寄存器 **cfg_cordic_argsize** 配置为 **1'b1** 时，输入格式为 32bit，CFG_CORDIC_CTRL0 对应的 CORDIC 输入变量个数选择寄存器 **cfg_cordic_nargs** 配置为 **1'b1**，输入变量数为 2，此时配置两次 32bit CFG_CORDIC_CTRL2 的输入变量配置寄存器 **cfg_cordic_arg**，依次配置变量 1、变量 2 后开始迭代计算；
3. CFG_CORDIC_CTRL0 对应的输入格式选择寄存器 **cfg_cordic_argsize** 配置为 **1'b1** 时，输入格式为 32bit，CFG_CORDIC_CTRL0 对应的 CORDIC 输入变量个数选择寄存器 **cfg_cordic_nargs** 配置为 **1'b0**，输入变量数为 1，此时配置 CFG_CORDIC_CTRL0 对应的 CORDIC 输入值选择寄存器 **cfg_cordic_arg_sel**，1'b0 对应变量 1，1'b1 对应变量 2，此时配置一次 32bit CFG_CORDIC_CTRL2 的输入变量配置寄存器 **cfg_cordic_arg** 后开始迭代计算；（注意对应的另外一个变量需要参考第二种配置在初始化时先进行一次配置，此后另外一个变量保持不变）。

对于需要一个变量的函数，变量输入配置如下：

1. CFG_CORDIC_CTRL0 对应的输入格式选择寄存器 **cfg_cordic_argsize** 配置为 **1'b0** 时，输入格式为 16bit，此时高 16bit 不关注、低 16bit 为变量 1，配置一次 32bit CFG_CORDIC_CTRL2 的输入变量配置寄存器 **cfg_cordic_arg** 后开始迭代计算；
2. CFG_CORDIC_CTRL0 对应的输入格式选择寄存器 **cfg_cordic_argsize** 配置为 **1'b1** 时，输入格式为 32bit，CFG_CORDIC_CTRL0 对应的 CORDIC 输入变量个数选择寄存器 **cfg_cordic_nargs** 配置为 **1'b0**，输入变量数为 1，配置 CFG_CORDIC_CTRL0 寄存器对应的 CORDIC 输入值选择寄存器 **cfg_cordic_arg_sel**，1'b0 对应变量 1，配置一次 32bit CFG_CORDIC_CTRL2 的输入变量配置寄存器 **cfg_cordic_arg** 后开始迭代计算。

29.4.4 CORDIC 输出结果

参见 29.4.6 "CORDIC 计算函数"章节表 29-3 的不同计算函数，除了反正切、双曲反正切、自然对数和平方根函数产生一个输出结果，其余函数产生两个输出结果。

因此对于产生两个输出结果的函数，结果读取配置和步骤如下：

1. CFG_CORDIC_CTRL0 对应的 CORDIC 输出结果格式选择寄存器 **cfg_cordic_ressize** 配置为 **1'b1** 时，输出结果格式为 16bit，CORDIC_RES1_RPT 的 cordic_res1_rpt 寄存器输出两个结果，高 16bit 为结果 2、低 16bit 为结果 1，此时读取 cordic_res1_rpt 即结束当次迭代；
2. CFG_CORDIC_CTRL0 对应的 CORDIC 输出结果格式选择寄存器

cfg_cordic_ressize 配置为 **1'b0** 时，输出结果格式为 32bit，CFG_CORDIC_CTRL10 对应的 CORDIC 输出结果读取个数选择寄存器 **cfg_cordic_nress** 配置为 **1'b1** 时，CORDIC_RES1_RPT 的 cordic_res1_rpt 寄存器输出结果 1，CORDIC_RES2_RPT 的 **cordic_res2_rpt** 寄存器输出结果 2，此时需要读取 cordic_res1_rpt 和 cordic_res2_rpt 才能结束当次迭代；

3. CFG_CORDIC_CTRL0 对应的 CORDIC 输出结果格式选择寄存器 **cfg_cordic_ressize** 配置为 **1'b0** 时，输出结果格式为 32bit，CFG_CORDIC_CTRL10 对应的 CORDIC 输出结果读取个数选择寄存器 **cfg_cordic_nress** 配置为 **1'b0** 时，CORDIC_RES1_RPT 的 **cordic_res1_rpt** 寄存器输出结果 1，此时读取 cordic_res1_rpt 即结束当次迭代；

对于产生一个输出结果的函数，结果读取配置和步骤如下：

1. CFG_CORDIC_CTRL0 对应的 CORDIC 输出结果格式选择寄存器 **cfg_cordic_ressize** 配置为 **1'b1** 时，输出结果格式为 16bit，CORDIC_RES1_RPT 的 **cordic_res1_rpt** 寄存器输出两个结果，高 16bit 为全 0、低 16bit 为结果 1，此时读取 cordic_res1_rpt 即结束当次迭代；
2. CFG_CORDIC_CTRL0 对应的 CORDIC 输出结果格式选择寄存器 **cfg_cordic_ressize** 配置为 **1'b0** 时，输出结果格式为 32bit，CFG_CORDIC_CTRL10 对应的 CORDIC 输出结果读取个数选择寄存器 **cfg_cordic_nress** 配置为 **1'b1** 时，CORDIC_RES1_RPT 的 **cordic_res1_rpt** 寄存器输出结果 1，CORDIC_RES2_RPT 的 **cordic_res2_rpt** 寄存器输出全 0，此时需要读取 cordic_res1_rpt 和 cordic_res2_rpt 才能结束当次迭代 (不推荐，会降低迭代速度)；
3. CFG_CORDIC_CTRL0 对应的 CORDIC 输出结果格式选择寄存器 **cfg_cordic_ressize** 配置为 **1'b0** 时，输出结果格式为 32bit，CFG_CORDIC_CTRL10 对应的 CORDIC 输出结果读取个数选择寄存器 **cfg_cordic_nress** 配置为 1'b0 时，CORDIC_RES1_RPT 的 **cordic_res1_rpt** 寄存器输出结果 1，此时读取 cordic_res1_rpt 即结束当次迭代；

29.4.5 CORIDC 迭代次数

配置 CFG_CORDIC_CTRL0 对应的 CORDIC 迭代周期数选择寄存器 **cfg_cordic_noi**，迭代周期数为配置值+1，迭代次数为迭代周期数 x4，当配置值大于 5 时，自动按照 6 次迭代周期进行迭代计算，迭代次数为 24 次。

29.4.6 CORDIC 计算函数

表29-3 CORDIC 支持函数列表

函数 (Function)	变量 1 (ARG1)	变量 2 (ARG2)	结果 1 (RES1)	结果 2 (RES2)
余弦 (Cosine)	角度θ	幅度$\frac{m}{k^{(1)}}$	$m\cdot\cos\theta$	$m\cdot\sin\theta$
正弦 (Sine)	角度θ	幅度$\frac{m}{k}$	$m\cdot\sin\theta$	$m\cdot\cos\theta$

函数 (Function)	变量 1 (ARG1)	变量 2 (ARG2)	结果 1 (RES1)	结果 2 (RES2)
相位 (Phase)	$\frac{x}{k}$	$\frac{y}{k}$	atan2(y,x)	$\sqrt{x^2+y^2}$
幅度 (Modulus)	$\frac{x}{k}$	$\frac{y}{k}$	$\sqrt{x^2+y^2}$	atan2(y,x)
反正切(Arctangent)	$\frac{x}{k}$	$\frac{1}{k}$	$\tan^{-1}(x)$	none
双曲余弦(Hyperbolic cosine)	x	$\frac{1}{kh^{(2)}}$	cosh x	sinh x
双曲正弦(Hyperbolic sine)	x	$\frac{1}{kh^{(2)}}$	sinh x	cosh x
双曲反正切 (Hyperbolic arctangent)	x	none	$\tanh^{-1}$ x	none
自然对数 (Natural logarithm)	x	none	ln x	none
平方根(Square root)	x	none	$kh\sqrt{x}$	none

(1) 圆周坐标旋转伸缩因子 k，只和迭代次数相关，计算公式如下：

$$k=\prod_l \frac{1}{\cos\theta^{(i)}}=\prod_l\left(\sqrt{1+2^{(-2i)}}\right),\ l = \text{number of iterations}$$

(2) 双曲坐标旋转伸缩因子 kh，同样只和迭代次数相关，计算公式如下：

$$kh=\prod_l \frac{1}{\cosh\theta^{(i)}}=\prod_{i=1}^{l}\left(\sqrt{1-2^{(-2i)}}\right),i=1,2,3,4,4,5,\ldots,\text{repeat when } i=4,13,\ l = \text{number of iterations}$$

(3) 以上函数值域由于算法特性，可能会超出值域范围，属于正常误差。

表29-4 余弦（Cosine）函数

参数	描述	值域
变量 1	角度θ，单位为π*弧度	[-1,1]
变量 2	幅度$\frac{m}{k^{(2)}}$	$\left[0,\frac{1}{k}\right]$
结果 1	$m\cdot\cos\theta$	[-1,1]
结果 2	$m\cdot\sin\theta$	[-1,1]
比例系数	N/A	0

(1) 输入变量 1 需要预处理，将弧度值除以π得到，作为输出结果时，需要乘以π得到弧度值。

(2) 输入变量 2 需要预处理，将幅度除以 k（圆周坐标旋转伸缩因子）得到。

表29-5 正弦（Sine）函数

参数	描述	值域
变量 1	角度θ，单位为π*弧度	[-1,1]
变量 2	幅度$\frac{m}{k}$	$\left[0,\frac{1}{k}\right]$
结果 1	$m\cdot\sin\theta$	[-1,1]
结果 2	$m\cdot\cos\theta$	[-1,1]
比例系数	N/A	0

表29-6 相位（Phase）函数

参数	描述	值域
变量 1	$\frac{x}{k}$坐标值	$\left[-\frac{1}{k},\frac{1}{k}\right]$
变量 2	$\frac{y}{k}$坐标值	$\left[-\frac{1}{k},\frac{1}{k}\right]$
结果 1	相位角度θ，单位为π*弧度	[-1,1]
结果 2	幅度 m	[-1,1]
比例系数	NA	0

(1) 输入变量 1、2 需要预处理，将 x、y 坐标除以 k（圆周坐标旋转伸缩因子）得到。

(2) 输出结果 2（幅度 m），由$\sqrt{x^2+y^2}$计算得到，如果$\sqrt{x^2+y^2}>1$，幅度 m 将饱和截位到 1。

表29-7 幅度（Modulus）函数

参数	描述	值域
变量 1	$\frac{x}{k}$坐标值	$\left[-\frac{1}{k},\frac{1}{k}\right]$
变量 2	$\frac{y}{k}$坐标值	$\left[-\frac{1}{k},\frac{1}{k}\right]$
结果 1	幅度 m	[-1,1]
结果 2	相位角度θ，单位为π*弧度	[-1,1]
比例系数	NA	0

表29-8 反正切(Arctangent)函数

参数	描述	值域
变量 1	$\frac{x}{k}\cdot 2^{-n}$	$\left[-\frac{1}{k},\frac{1}{k}\right]$
变量 2	$\frac{1}{k}\cdot 2^{-n}$	-
结果 1	$2^{-n}\cdot\tan^{-1}x$，单位为π*弧度	[-1,1]

参数	描述	值域
结果 2	N/A	-
比例系数	n	[0,7]

(1) 输入变量 1、2 需要预处理，将 x、1 除以 k（圆周坐标旋转伸缩因子）得到。

(2) 当输入变量或者输出结果超过 1 时，需要通过比例系数2^{-n}将输入输出值缩放到[-1,1]之间，比例系数每个函数各自定义其使用范围。

(3) 对于反正切函数，最大输入值$x=\tan\theta=128$，可得最大输出角度$\theta\approx89.552°$，绝对值大于 128 的输入，需要根据反正切函数的性质间接求得函数值。

已知反正切函数的计算公式：

$$\tan^{-1}A+\tan^{-1}B=\tan^{-1}\frac{A+B}{1-AB}$$

当$x>128$，令$B=1$，$\frac{A+B}{1-AB}=x$，可得$A=\frac{x-1}{x+1}\in(0,1)$，因此将$\frac{x-1}{x+1}$带入反正切函数，并将结果加上$\tan^{-1}1=\frac{\pi}{4}$，即可得到$\tan^{-1}x$的结果，即$\tan^{-1}x=\tan^{-1}\frac{x-1}{x+1}+\frac{\pi}{4}$。

当$x<-128$，令$B=-1$，$\frac{A+B}{1-AB}=x$，可得$A=\frac{1+x}{1-x}\in(-1,0)$，因此将$\frac{1+x}{1-x}$带入反正切函数，并将结果加上$\tan^{-1}(-1)=-\frac{\pi}{4}$，即可得到$\tan^{-1}x$的结果，即$\tan^{-1}x=\tan^{-1}\frac{1+x}{1-x}-\frac{\pi}{4}$。

表29-9 双曲余弦（Hyperbolic cosine）函数

参数	描述	值域
变量 1	$x\cdot2^{-n}$	[-0.559,0.559]
变量 2	$\frac{1}{kh}\cdot2^{-n}$	-
结果 1	$2^{-n}\cdot\cosh x$	[0.5,0.846]
结果 2	$2^{-n}\cdot\sinh x$	[-0.683,0.683]
比例系数	n	1

(1) 输入变量 2 需要预处理，将 1 除以 kh（双曲坐标旋转伸缩因子）得到。

表29-10 双曲正弦（Hyperbolic sine）函数

参数	描述	值域
变量 1	$x\cdot2^{-n}$	[-0.559,0.559]
变量 2	$\frac{1}{kh}\cdot2^{-n}$	-
结果 1	$2^{-n}\cdot\sinh x$	[−0.683,0.683]
结果 2	$2^{-n}\cdot\cosh x$	[0.5,0.846]
比例系数	n	1

(1) 对于双曲余弦/正弦函数，由于$\sum_{i=1}^{\infty}\tanh^{-1}2^{-i}=1.1181$，并且$\cosh x\geq 1$，因此将比例系数固定为 1，输入变量 1 的定义域为[-0.559,0.559]。

(2) 对于x∈[-1.1181,1.1181]之外的函数值，可以根据以下公式间接求得：

① 两角和差公式

$$\sinh(x+y)=\sinh x\cosh y+\cosh x\sinh y$$

$$\sinh(x-y)=\sinh x\cosh y-\cosh x\sinh y$$

$$\cosh(x+y)=\cosh x\cosh y+\sinh x\sinh y$$

$$\cosh(x-y)=\cosh x\cosh y-\sinh x\sinh y$$

② 二倍角公式

$$\sinh 2x=2\sinh x\cosh x$$

$$\cosh 2x=\cosh^2 x+\sinh^2 x=2\cosh^2 x-1=2\sinh^2 x+1$$

③ 三倍角公式

$$\sinh 3x=3\sinh x+4\sinh^3 x$$

$$\cosh 3x=4\cosh^3 x-3\cosh x$$

表29-11 双曲反正切（Hyperbolic arctangent）函数

参数	描述	值域
变量 1	$x\cdot 2^{-n}$	[-0.403,0.403]
变量 2	NA	-
结果 1	$2^{-n}\cdot\tanh^{-1}x$	[-0.559,0.559]
结果 2	NA	-
比例系数	n	1

(1) 对于双曲反正切函数，同样由于$\sum_{i=1}^{\infty}\tanh^{-1}2^{-i}=1.1181$，因此将比例系数固定为 1，输出结果 1 的值域为[-0.559,0.559]，并且双曲反正切函数为单调递增函数，得到输入变量 1 的定义域为[-0.403,0.403]。

(2) 对于x∈(-1,-0.8069),(0.8069,1)区间的函数值，需要根据双曲反正切函数的性质间接求得函数值。

已知双曲反正切函数的计算公式：

$$\tanh^{-1}A+\tanh^{-1}B=\tanh^{-1}\frac{A+B}{1+AB}$$

当$x\in(0.8069,1)$，令$B=\frac{1}{2}$，$\frac{A+B}{1+AB}=x$，可得$A=\frac{2x-1}{2-x}\in\left(0,\frac{1}{2}\right)$，因此将$\frac{2x-1}{2-x}$带入反正切函数，并将结果加上$\tanh^{-1}\frac{1}{2}$，即可得到$\tanh^{-1}x$的结果，即$\tanh^{-1}x=\tanh^{-1}\frac{2x-1}{2-x}+\tanh^{-1}\frac{1}{2}$。

当$x\in(-1,-0.8069)$，令$B=-\frac{1}{2}$，$\frac{A+B}{1+AB}=x$，可得$A=\frac{2x+1}{x+2}\in\left(-\frac{1}{2},0\right)$，因此将$\frac{2x+1}{x+2}$带入反正切

函数，并将结果加上 $\tanh^{-1}\left(-\frac{1}{2}\right)$，即可得到 $\tanh^{-1}x$ 的结果，即 $\tanh^{-1}x=\tanh^{-1}\frac{2x+1}{x+2}+\tanh^{-1}\left(-\frac{1}{2}\right)$。

表29-12 自然对数（Natural logarithm）函数

参数	描述	值域
变量 1	$x\cdot 2^{-n}$	[0.054,0.875]
变量 2	N/A	-
结果 1	$2^{-(n+1)}\cdot\ln x$	[-0.559,0.137]
结果 2	N/A	-
比例系数	n	[1,4]

(1) 自然对数函数由 $\ln x=2\tanh^{-1}\left(\frac{x-1}{x+1}\right)$ 间接求得，根据双曲反正切函数的定义域可得 $\left|\frac{x-1}{x+1}\right|\leq 0.8069$，从而可确定 $x\in[0.1069,9.3573]$，因此可得到比例系数以及对应值域。

对于 $x\in(0,\ 0.1069)$，$(9.3573,\ +\infty)$ 区间内的函数值，可以根据以下公式间接求得：

$$\ln xy=\ln x+\ln y$$

$$\ln\frac{x}{y}=\ln x-\ln y$$

$$\ln x^{y}=y\ln x$$

$$\ln\sqrt[y]{x}=\frac{\ln x}{y}$$

$$\ln x=-\ln\frac{1}{x}$$

表29-13 自然对数比例系数及对应变量和结果值域

n	x 值域	ARG1 值域	RES1 值域
1	0.107≤x<1	0.0535≤ARG1<0.5	-0.559≤RES1<0
2	1≤x<3	0.25≤ARG1<0.75	0≤RES1<0.137
3	3≤x<7	0.375≤ARG1<0.875	0.069≤RES1<0.122
4	7≤x≤9.35	0.4375≤ARG1≤0.584	0.061≤RES1≤0.070

表29-14 平方根（Square root）函数

参数	描述	值域
变量 1	$x\cdot 2^{-n}$	[0.027,0.875]
变量 2	N/A	-
结果 1	$2^{-n}\cdot kh\cdot\sqrt{x}$	[0.164kh,0.866kh]

参数	描述	值域
结果 2	N/A	-
比例系数	n	[0,2]

(1) 输出结果需要除以 kh (双曲坐标伸缩因子) 和2^{-n}比例系数得到平方根结果。

(2) 平方根函数由双曲线坐标系，向量模式的输出$x=K^*\sqrt{x^2-y^2}$间接求得，$\sqrt{x}=\sqrt{\left(x+\frac{1}{4}\right)^2-\left(x-\frac{1}{4}\right)^2}$，带入$\left|\frac{x-\frac{1}{4}}{x+\frac{1}{4}}\right|\leq 0.8069$，从而可确定$x\in[0.027,2.341]$，因此可得到比例系数以及对应值域。

对于$x\in(0，0.0.027)，(2.341，+\infty)$区间内的函数值，可以根据以下公式间接求得：

$$\sqrt{xy}=\sqrt{x}\sqrt{y}$$

$$\sqrt{\frac{x}{y}}=\frac{\sqrt{x}}{\sqrt{y}}$$

表29-15 平方根比例系数及对应值域

n	x 值域	ARG1 值域	RES1 值域
0	0.027≤x<0.75	0.027≤ARG1<0.75	0.164kh≤RES1<0.866kh
1	0.75≤x<1.75	0.375≤ARG1<0.875	0.433kh≤RES1<0.661kh
2	1.75≤x≤2.341	0.4375≤ARG1<0.585	0.330kh≤RES1<0.383kh

29.4.7 CORDIC 结果上报模式

配置 CFG_CORDIC_CTRL1 对应的 CORDIC 结果上报模式选择寄存器 **cfg_cordic_work_mode** 如下：

- 2'b00:中断模式
- 2'b01:轮询模式
- 2'b10:零开销模式
- 2'b11:DMA 搬运模式

当配置为中断模式时，需要根据 29.4.4 "CORDIC 输出结果"章节配置，读取对应的结果，并且将上报的中断写清；

当配置为轮询模式、零开销和 DMA 搬运模式时，需要根据 29.4.4 "CORDIC 输出结果"章节配置，读取对应的结果；

当配置为零开销模式时，可以不用读取 CORDIC_STATUS_RPT 的 CORDIC 计算完成结果上报中断信号状态上报寄存器 **rrdy_flg_rpt**，立即发起对于结果的读取，该模式下会在当前迭代结束后返回当前迭代的结果，其余三种模式需要首先读取 CORDIC_STATUS_RPT 的 CORDIC 计算完成结果上报中断信号状态上报寄存器 **rrdy_flg_rpt**，等待当前迭代结束后发起对于结果的读取，并且读取后需要对 CORDIC_STATUS_RPT 的 CORDIC 计算完成结果上报中断信号状态上报寄存器 **rrdy_flg_rpt** 进行写 1 清，之后才能开始下一次迭代。

29.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 CORDIC 章节。

30 滤波器数学加速器 (FMAC)

30.1 简介

FMAC 对向量执行算术运算。它包括一个乘法/累加 (MAC) 单元，以及地址生成逻辑，这使得它可以索引保存在本地内存中的向量元素。

该单元支持输入输出环形缓冲器，以实现数字滤波器。可实现 FIR 滤波器，IIR 滤波器，相关运算和向量的点积运算。

FMAC 可以从 CPU 中释放与滤波器有关的运算，让 CPU 去执行其他任务。在许多情况下，与软件实现相比，它可以加速此类计算，从而加快时间关键任务的速度。

30.2 结构框图

FMAC 结构框图如下：

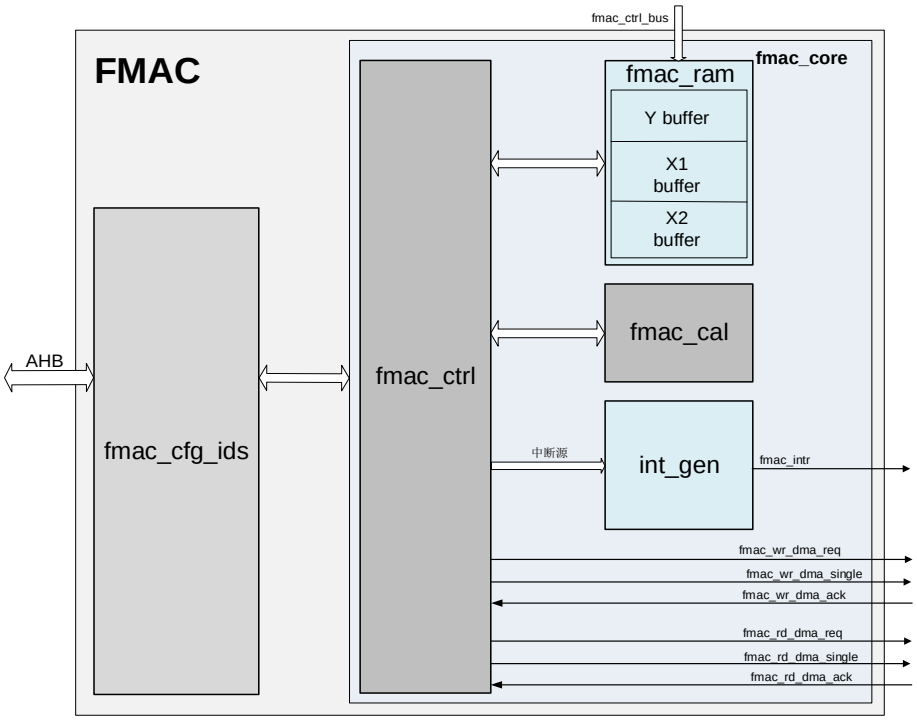


图30-1 FMAC 结构框图

30.3 主要特性

主要特性如下：

- 16 位输入输出数据

- 256 x 16 位本地内存
- 最多可以在内存中定义三个区域用于数据缓存 (两个输入，一个输出)，由可编程的基地址指针和相关的大小寄存器定义
- 输入和输出采样缓冲区为环形 buffer
- 可编程输入输出 buffer“水线”以减少中断开销
- 滤波器功能：FIR，IIR (直接 1 型)
- 支持 DMA 读写通道

30.4 功能描述

该外设是围绕一个定点乘法器和累加器 (MAC) 构建而成的。MAC 可以从内存中读取两个 16 位有符号输入值，将它们相乘并累加。内存中输入值的地址通过一组指针确定，这些指针可以由内部控制逻辑加载、递增、循环或重置。

为了计算点积，处理器或 DMA 控制器将两个输入向量加载到本地内存中，选择并启动所需要的操作。每对输入向量元素都从内存中提取，相乘并累加。当所有向量元素都处理完毕后，累加器的内容就存储在本地内存中，处理器或 DMA 再从本地内存中读出运算结果。

有限脉冲响应 (FIR) 滤波器操作 (也称为卷积) 包括重复计算系数向量和输入样本向量的点积，一次点积计算完之后将输入样本向量移动一个样本延迟，丢弃掉最先被计算的样本，并在每次重复计算中添加一个新样本。

无限脉冲响应 (IIR) 滤波器运算是将反馈系数与之前的输出样本进行卷积，再加上 FIR 卷积的结果。

30.4.1 本地内存和缓冲区

该外设包含一个 256×16 位读写内存，用于本地存储：

- 输入值 (输入样本向量和系数向量) 存储在两个缓冲区 X1 和 X2 中
- 输出值 (运算结果) 存储在另一个缓冲区 Y 中
- 缓冲区的位置和大小指定如下：
  - x1_base: X1 缓冲区的基址
  - x2_base: X2 缓冲区的基址
  - y_base: Y 缓冲区的基址
  - x1_buf_size: X1 缓冲区的地址大小
  - x2_buf_size: X2 缓冲区的地址大小
  - y_buf_size: Y 缓冲区的地址大小

CPU (或 DMA 控制器) 可以初始化每个缓冲区的内容。输入值被传输到写指针指示的目标缓冲区中的位置。每次写操作完成后，写指针都会递增。当写指针到达所分配的缓冲区空间的末尾时，它将绕回到基地址。

30.4.2 缓冲区配置

必须在 X1、X2 和 Y 缓冲区配置寄存器中配置缓冲区大小和基地址。基地址可以在内存中的任何位置选择，前提是所有缓冲区都符合内存地址范围 (0x00 到 0xFF)，也就是说，基址+缓冲区大小必须小于 256。

缓冲区的大小和位置没有限制 (它们可以重叠)，对于滤波器功能，建议不要重叠缓冲区，因为如果操作不当可能会导致错误的行为。

设置循环缓冲区的大小时，可以在缓冲区的大小上增加一个可选的余量 (d)。此外，还可以设置水线，以调节 CPU 或 DMA 活动。余量的取值和水线的选择应根据应用需求进行。为了获得最大吞吐量，输入缓冲区不应该空，因此余量应该大于或者等于水线，允许任何中断或 DMA 延迟。如果输入数据速率小于运算速度，则可以允许缓冲区清空，等待下一个数据写入。

30.4.3 输入缓冲区

X1 和 X2 缓冲区用于存储输入到 MAC 的数据。每次乘法都从 X1 缓冲区和 X2 缓冲区中取一个值并将它们相乘。控制逻辑中的指针为每个值生成读地址偏移量 (相对于缓冲区基地址)。

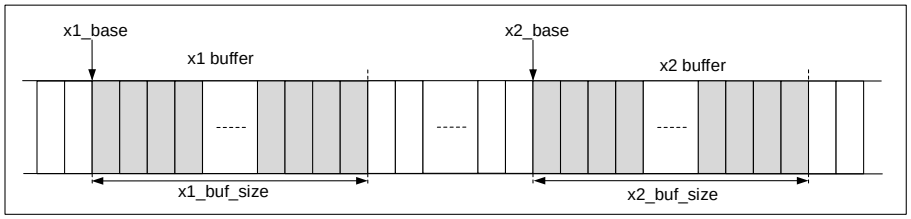


图30-2 输入缓冲区

X1 缓冲区可以用作循环缓冲区，在这种情况下，只要有空间，新数据就能连续传输到输入缓冲区。预加载这个缓冲区对于数字滤波器来说是可选的，因为如果在运算开始时在缓冲区没有输入样本，它会被标记为空，这会触发 CPU 或 DMA 加载新的样本，直到有足够的样本才开始运算。

X2 缓冲区只能在向量模式下使用，并且需要预加载。对于滤波器功能，X2 缓冲区用于存储滤波器系数。

当 X1 缓冲区作为循环缓冲区使用时，分配给缓冲区的空间 (x1_buf_size) 通常应该大于当前计算中使用的元素数量，这样缓冲区中有新的值可用。图 30-3 演示了滤波器运算时循环缓冲区中数据的分布。在计算输出样本 y[n]时，该运算使用一组 N+1 个输入样本，x[n-N]到 x[n]。完成后，单元开始计算 y[n+1]，使用输入样本集合 x[n- N+1]到 x[n+1]。最先输入的样本 x[n- N]从运算向量中移除，一个新的样本 x[n+1]被添加到运算向量中。如果缓冲区中没有样本 x[n+1]，缓冲区将被标记为空，运算将暂停，直到缓冲区中有新的样本再继续运算。

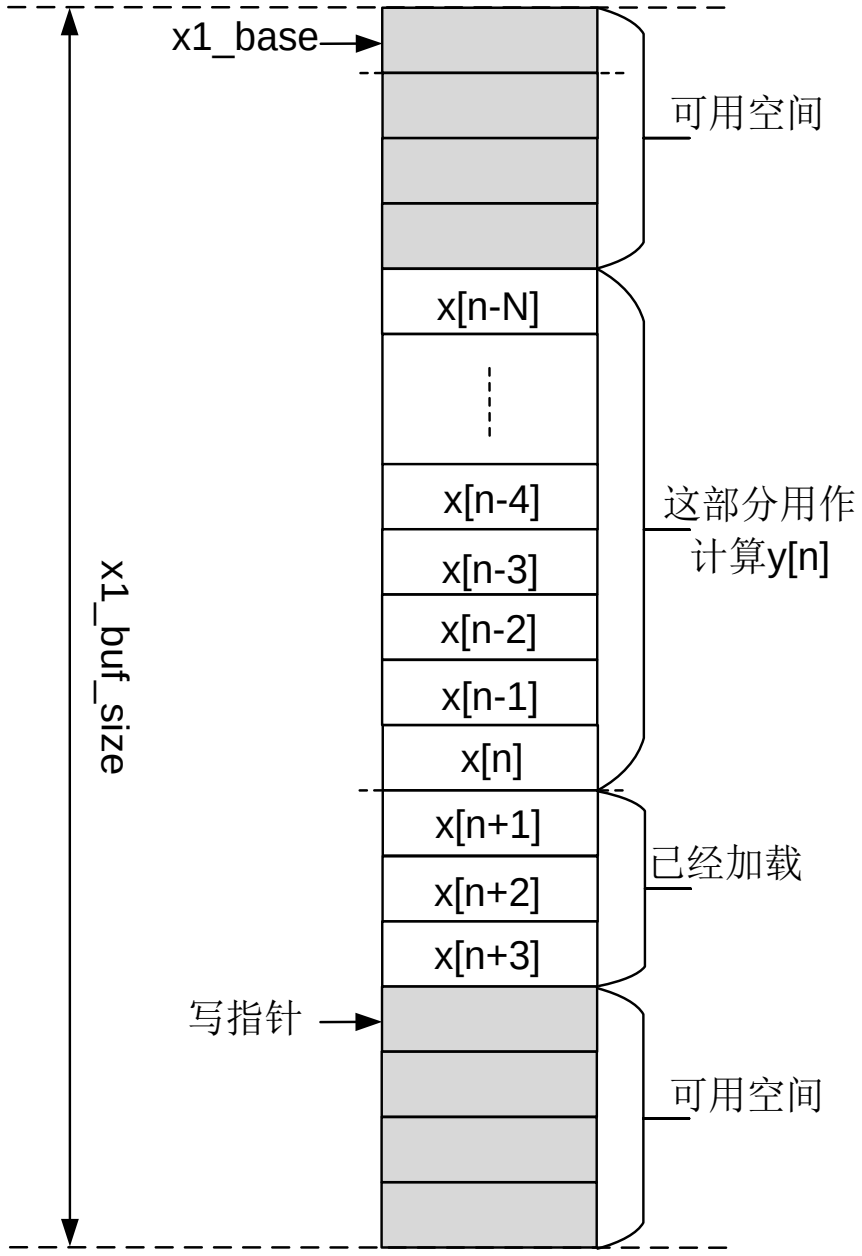


图30-3 循环输入缓冲区

注意：

如果样本的输入是由定时器或其他外设 (如 ADC) 控制的，缓冲区会定期变空，因为滤波器处理每个新样本的速度比样本的输入速度快。这是滤波器操作的一个基本特征。

如果缓冲区中的空闲空间数量小于写 X1 缓冲区水线，则 X1 缓冲区被标记为满。只要未设置满标志，就会生成中断请求 (如果启用)，以便为缓冲区请求更多数据。水线允许在一个中断下传输多个数据，而不会溢出。如果发生溢出，则设置上溢错误标志并忽略写入的数据。写指针在发生溢出时不会递增。

图 30-4 说明了滤波运算期间 X1 缓冲区的操作过程。这个例子显示了一个水线设置为 4 的 8 抽头 FIR 滤波器。

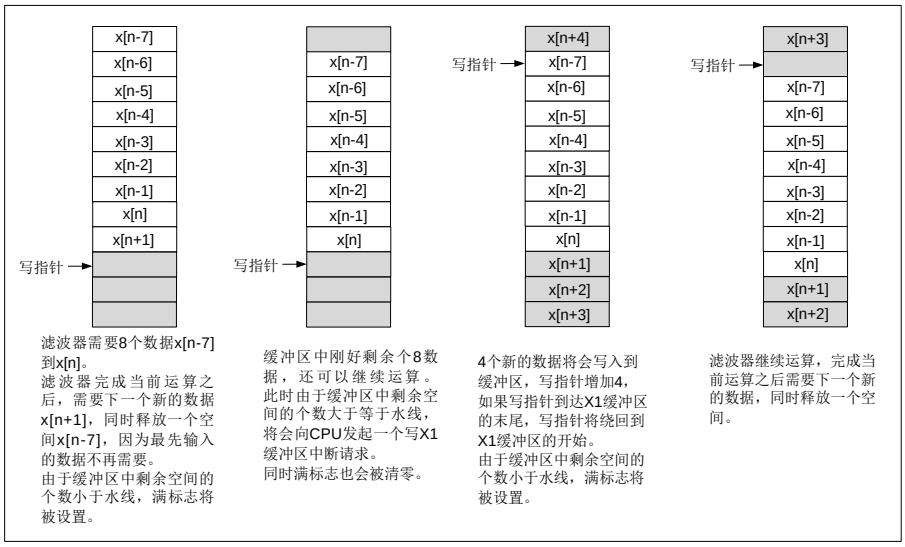


图30-4 循环输入缓冲区的操作过程

30.4.4 输出缓冲区

Y(输出) 缓冲区用于存储运算结果。每个新的输出值都存储在缓冲区中，直到被处理器或 DMA 控制器读取为止。通过读数据寄存器进行读访问，从读指针指示的地址提取读数据，该指针在每次读取后递增，并在到达分配的 Y 缓冲区空间的末尾时绕回到缓冲区的基地址。

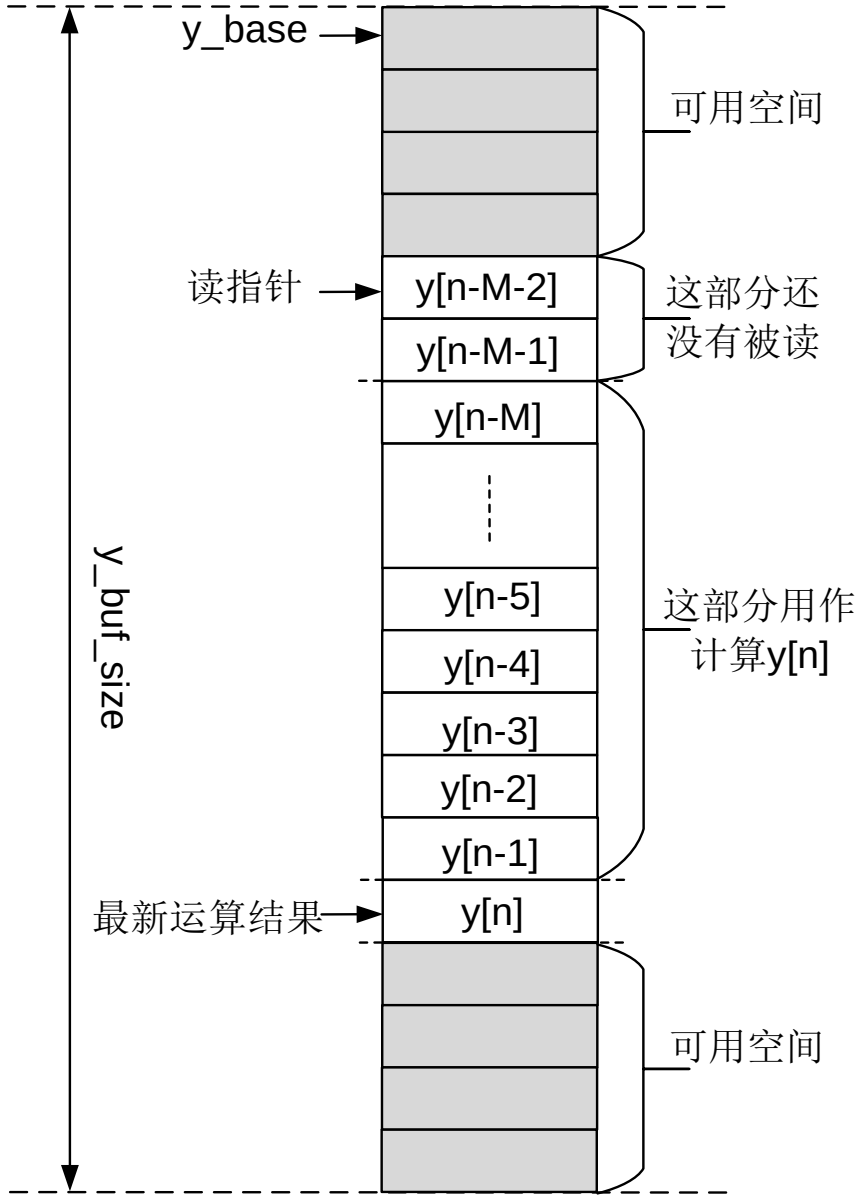


图30-5 循环输出缓冲区

Y 缓冲器也可以作为循环缓冲器使用。如果下一个运算结果需要写入的地址与读指针(未读样本) 所指示的地址相同并且 Y 缓冲区中未读数据的个数大于 0，则缓冲区将被标记为已满，并暂停运算，直到样本被读为止。

在 IIR 滤波器的情况下，Y 缓冲区用于存储 M 个前输出样本的集合，Y [n-M]到 Y [n-1]，用于计算下一个输出样本 Y [n]。每当一个新的样本被添加到集合中，最先进入集合的样本 y[n-M]就会移除。

如果缓冲区中未读数据的数量小于读 Y 缓冲区水线，则 Y 缓冲区被标记为空。只要没有设置空标志，就会生成中断或 DMA 请求 (如果启用)，请求从缓冲区读取运算结果。水线允许在一个中断下传输多个数据，而不会发生下溢。如果确实发生下溢，则设置下溢错误标志。在这种情况下，读指针不递增，读操作返回一个错误的结果。

图 30-6 说明了滤波运算期间 Y 缓冲区的操作过程。这个例子显示了一个水线设置为 4 的 7 抽头 IIR 滤波器。

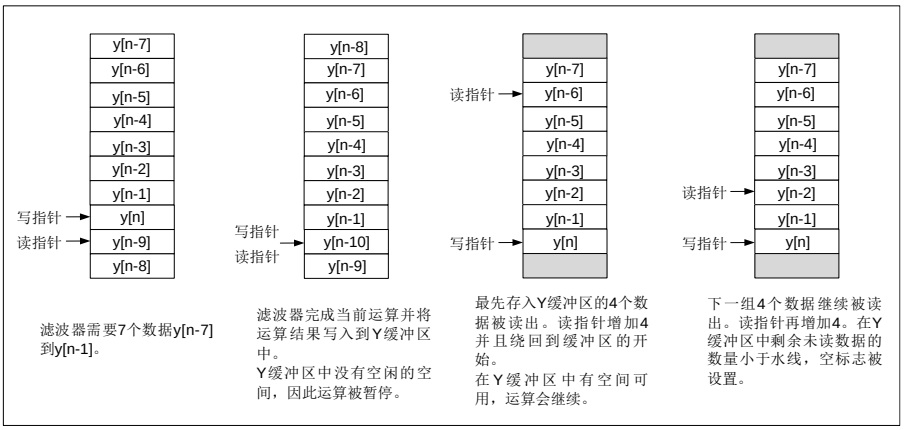


图30-6 循环输出缓冲区的操作过程

30.4.5 初始化功能

下面的功能初始化 FMAC 单元，需要在 PARAM 寄存器的 cfg_func 位域中设置适当的值，并通过设置 cfg_enable 位域触发。cfg_p 和 cfg_q 位域还必须包含每个功能的适当参数值，位域 cfg_r 未使用。当初始化功能完成后，硬件会自动将位域 cfg_enable 清零。

在初始化期间，建议禁用 DMA 请求和中断。数据到 FMAC 内存的传输可以通过软件或内存/外设到内存的 DMA 传输完成。

30.4.5.1 加载 X1 缓冲区

该功能从 x1_base 地址开始，通过 **fifo_data** 寄存器将 N 个输入数据加载到 X1 缓冲区中。当该初始化完成时，内部写指针指向地址 x1_base+N。

参数：

- 参数 **cfg_p** 包含要加载到 X1 缓冲区中的输入数据的数量 N
- 参数 **cfg_q** 和 **cfg_r** 未被使用。

当对 **fifo_data** 寄存器执行了 N 次写操作时，初始化功能就完成了。

30.4.5.2 加载 X2 缓冲区

该功能从 x2_base 地址开始，通过 fifo_data 寄存器将 N+M 个输入数据加载到 X2 缓冲区中。

该功能可用于预加载向量或滤波器系数到 X2 缓冲区中。在 IIR 的情况下，N 个前馈和 M 个反馈系数被连接并加载到 X2 缓冲区中。系数的总数等于 N+M。对于 FIR，没有反馈系数，所以 M = 0。

参数：

- 参数 **cfg_p** 包含要从地址 x2_base 开始加载到 X2 缓冲区中的数据的数量 N。

- 参数 **cfg_q** 包含要从地址 x2_base+N 开始加载到 X2 缓冲区中的数据的数量 M。
- 参数 **cfg_r** 未被使用。

当对 **fifo_data** 寄存器执行了 N+M 次写操作时，初始化功能就完成了。

30.4.5.3 加载 Y 缓冲区

该功能从 y_base 地址开始，通过 fifo_data 寄存器将 N 个输入数据加载到 Y 缓冲区中。当该初始化完成时，内部写指针指向地址 y_base+N。

该功能可用于预加载 IIR 滤波器的反馈存储元素。

参数：

- 参数 **cfg_p** 包含要加载到 Y 缓冲区中的数据的数量 N。
- 参数 **cfg_q** 和 **cfg_r** 未被使用。

当对 **fifo_data** 寄存器执行了 N 次写操作时，初始化功能就完成了。

30.4.6 滤波器功能

FMAC 单元支持以下滤波器功能。这些功能是通过将相应的值写入 PARAM 寄存器的 cfg_func 位域中并设置 cfg_enable 位来触发的。cfg_p，cfg_q 和 cfg_r 位域包含每个滤波器功能所需要的参数值。滤波功能持续运行，直到 cfg_enable 位被软件清零。

30.4.6.1 FIR 滤波器 (Convolution)

Y=B*X (34-1)

$y_n=2^R\times\sum_{k=0}^{N} b_k x_{n-k}$ (34-2)

该功能执行长度为 N+1 的向量 B 与不定长度的向量 X 的卷积。Y 中每次增加的元素y_n都是用点积来计算的: y_n=B*X_n，其中X_n=[x_{n-N},⋯,x_n]由 N+1 个 X 中的元素组成。

该功能对应于有限脉冲响应 (FIR) 滤波器，其中向量 B 包含滤波器系数，向量 X 包含输入数据。

FIR 滤波器的结构如下图所示。

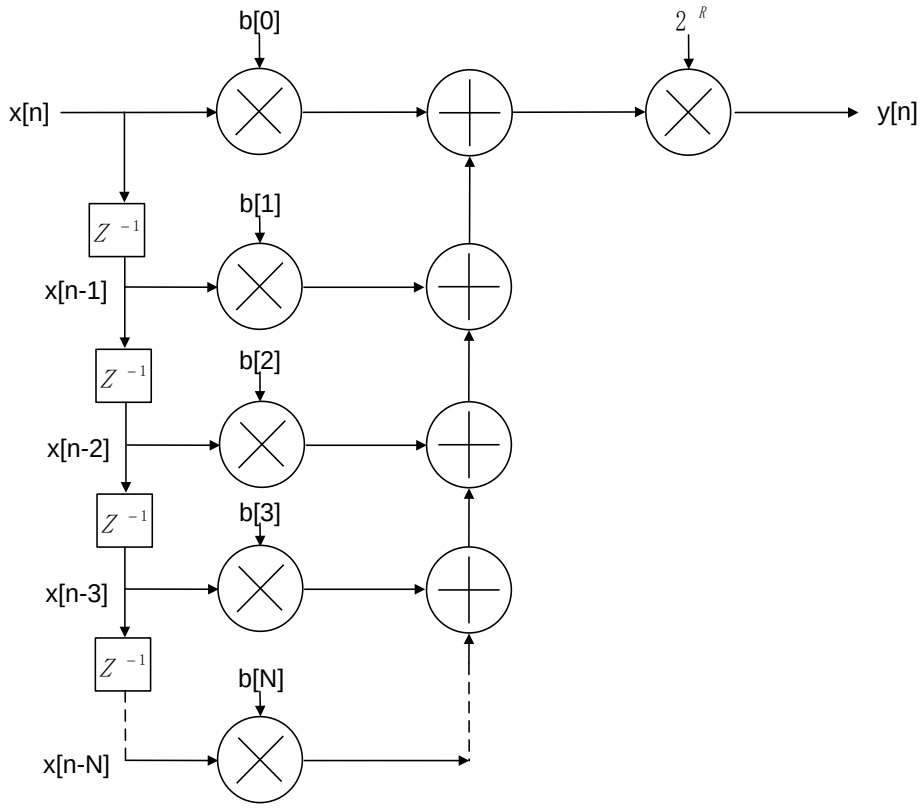


图30-7 FIR 滤波器的结构

输入:

- X1 缓冲区包含向量 X 的元素，它是一个长度为 N+1+d 的循环缓冲区。
- X2 缓冲区包含向量 B 的元素，它是一个长度为 N+1 的固定缓冲区。

输出:

- Y 缓冲区包含输出值y_n，它是一个长度为 d 的循环缓冲区。

参数:

- 参数 **cfg_p** 包含长度为 N+1 的系数向量 B，范围为[2:127]。
- 参数 **cfg_r** 包含应用于累加器输出的增益值，输出到 Y 缓冲区的值乘以 2^R，其中 R 的范围为[0:7]。
- 参数 cfg_q 未被使用。

当软件拉低 PARAM 寄存器中的 cfg_enable 位时，此功能完成。

30.4.6.2 IIR 滤波器 (直接 1 型)

$Y=B*X+A*Y'$ (34-3)

$y_n=2^R\times\left(\sum_{k=0}^{N} b_k x_{n-k}+\sum_{k=1}^{M} a_k y_{n-k}\right)$ (34-4)

这个功能实现了一个无限脉冲响应(IIR)滤波器。滤波器输出向量 Y 是长度为 N+1 的系数向量 B 和不确定长度的向量 X 的卷积，加上延迟输出向量Y'与长度为 M 的第二个系数向量 A 的卷积。Y 中每次新增的元素：$y_n=B*X_n+A*Y_{n-1}$，其中$X_n=[x_{n-N},\cdots,x_n]$由 N+1 个 X 中

的元素组成，$Y_{n-1}=[y_{n-M},\cdots,y_{n-1}]$由 M 个 Y 中的元素组成。

IIR 滤波器 (直接 1 型) 的结构如下图所示。

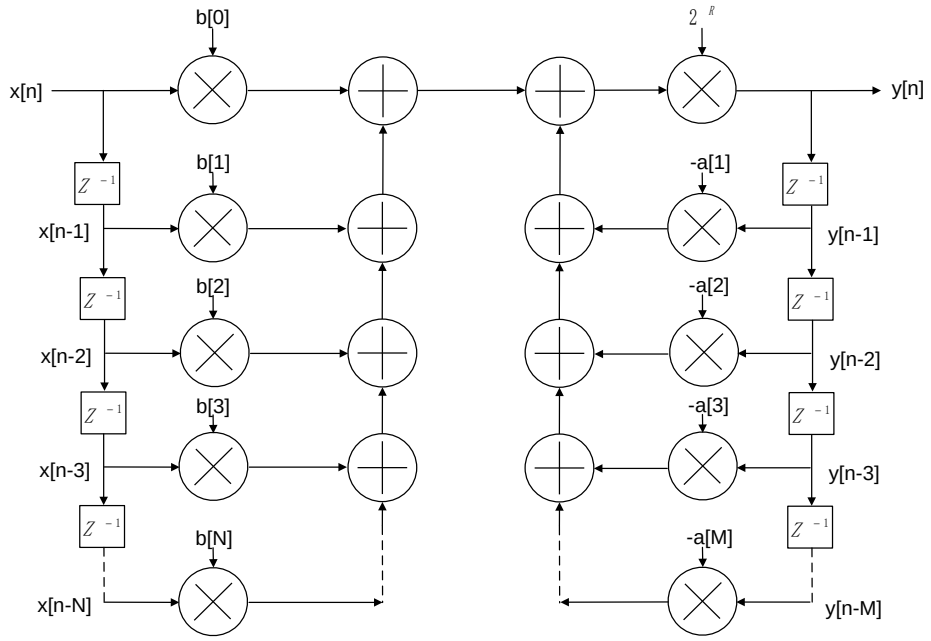


图30-8 IIR 滤波器 (直接 1 型) 的结构

输入:

- X1 缓冲区包含向量 X 的元素，它是一个长度为 N+1+d 的循环缓冲区。
- X2 缓冲区包含系数向量 B 和 A 的元素 $(b_0,b_1,b_2...,b_N,a_1,a_2,...,a_M)$，它是一个长度为 M+N+1 的固定缓冲区。

输出:

- Y 缓冲区包含输出值y_n，它是一个长度为 M+d 的循环缓冲区。

参数:

- 参数 **cfg_p** 包含长度为 N+1 的系数向量 B，范围为[2:64]。
- 参数 **cfg_q** 包含长度为 M 的系数向量 A，范围为[1:63]。
- 参数 **cfg_r** 包含应用于累加器输出的增益值，输出到 Y 缓冲区的值乘以 2^R，其中 R 的范围为[0:7]。

当软件拉低 PARAM 寄存器中的 cfg_enable 位时，此功能完成。

30.4.7 定点数表示

FMAC 以定点有符号整数格式进行运算。输入和输出数据的格式为 s(16,15)。

在 s(16,15)格式中，数字总共 16 位，由一个符号/整数位和 15 个小数位 (二进制小数位)

表示，数值范围是-1 (0x8000)到 1–2^-15 (0x7FFF)。

累加器有 26 位，其中 22 位是小数，4 位是整数/符号位，表示为 s(26,22)，支持的数值范围为-8 (0x2000000)到+7.99999976 (0x1FFFFFF)。cfg_r 的范围为[0:7]，故可编程增益从 0dB 到 42dB，步长为 6dB，可应用于累加器的输出。

注意：

如果超过数值范围，累加器的内容可以通过 CR 寄存器中的 cfg_clipen 位选择 wrap 或者 saturate 的处理方式。当选择 wrap 的处理方式时，大于+7.99999976 或小于-8 的部分和将 wrap，如果后续的累加结果可以撤销 wrap，则这是无害的。然而，如果累加器的内容发生了溢出，SR 寄存器中的 sat_rpt 标志将会被设置，如果使能了溢出中断，则会产生中断请求，这有助于滤波器的调试。

在应用可编程增益后，累加器输出的数据可以通过 CR 寄存器中的 cfg_acc_out_wrap_sel 位选择 saturate 或者 wrap 的处理方式。

30.4.8 用 FMAC 实现 FIR 滤波器

FMAC 支持长度为 N 的 FIR 滤波器，其中 N 是抽头或相关系数的数量。对于长度为 N 的 FIR 滤波器，本地内存的最小要求是 2N+1。

- N 个系数
- N 个输入数据
- 1 个输出数据

由于本地内存大小为 256，所以 N 的最大值为 127。

如果需要提高吞吐量，可以为输入和输出数据缓冲区分别分配少量额外空间 d1 和 d2，以确保滤波器既不会因为等待新的输入数据被写入而暂停，也不会因为等待输出数据被读取而暂停。在这种情况下，本地内存的需求为 2N+d1+d2。

缓冲区的配置如下:

- x1_buf_size = N+d1;
- x2_buf_size = N;
- y_buf_size = d2 (如果没有额外的空间被需要可以设置为 1)；

缓冲区基址可以分配到内存中的任何位置，但 X2 缓冲区不能与其他缓冲区重叠，否则系数将被覆盖。例如:

- x2_base = 0;
- x1_base = N;
- y_base = 2N+d1；

但是若内存空间有限，X1 和 Y 缓冲区可以重叠，这样每个输出样本都会取代最原始的不再需要的输入样本：

- x2_base = 0;
- x1_base = N;
- y_base = N

在本例中，y_buf_size = x1_buf_size = N+d1，以便缓冲区保持同步。

注意：

X1 缓冲区的水线必须配置为小于或等于 $\log_2$(d1)的值，否则在写入 N 个输入样本之前，缓冲区就被标记为满，并且不再请求更多的样本。类似地，Y 缓冲区的水线也必须小于或等于 $\log_2$(d2)。

滤波器系数必须使用加载 X2 缓冲区功能预先加载到 X2 缓冲区中。X1 缓冲区可以选择预加载任意数量的样本，最大可达 N+d1。预加载 Y 缓冲区没有意义，因为对于 FIR 滤波器，没有反馈路径。

在配置和初始化缓冲区之后，应该根据应用需求设置向 FMAC 内存写入数据和读取数据的方式。

支持三种方式：

- 轮询：不产生 DMA 请求或中断请求。软件必须在向 X1 缓冲区写入数据之前检查 X1 缓冲区的满标志值是否为低，或者在从 Y 缓冲区读取数据之前检查 Y 缓冲区空标志值是否低。
- 中断：当 X1 缓冲区的满标志值为低时，产生写数据中断请求；当 Y 缓冲区空标志值为低时，产生读数据中断请求；
- DMA：当 X1 缓冲区的满标志值为低时，产生 DMA 写数据请求；当 Y 缓冲区空标志值为低时，产生 DMA 读数据请求。

读写数据可以使用不同的方法。但是，对于同一个操作 (写入或者读取)，不建议同时使用中断和 DMA 请求。读写数据的有效组合方式如下表所示。

表30-1 读写数据的有效组合方式

写数据中断使能	读数据中断使能	DMA 写数据使能	DMA 读数据使能	写数据	读数据
0	0	0	0	轮询	轮询
0	1	0	0	轮询	中断
1	0	0	0	中断	轮询
1	1	0	0	中断	中断
0	0	0	1	轮询	DMA
0	0	1	0	DMA	轮询
0	0	1	1	DMA	DMA
0	1	1	0	DMA	中断
1	0	0	1	中断	DMA

FMAC 通过以下配置流程来启动 FIR 滤波器：

- 配置 CR 寄存器，主要配置 cfg_clr 位域复位内部逻辑，其他位域 (dma 使能和 clipen 选择等) 根据需要配置；
- 清中断 (避免上次操作结束时未清中断，导致中断上报寄存器中的状态错误)；
- 根据滤波器的阶数配置 X1_BUF 寄存器，包括 X1 缓冲区的基地址 (x1_base)，X1 缓冲区大小 (x1_buf_size)，和 X1 缓冲区水线(x1_full_wm)；
- 配置 PARAM 寄存器，加载 X1 缓冲区；
- 根据滤波器的阶数配置 X2_BUF 寄存器，包括 X2 缓冲区的基地址 (x2_base)，X2 缓冲区大小 (x2_buf_size)；

- 配置 PARAM 寄存器，加载 X2 缓冲区；
- 根据滤波器的阶数配置 Y_BUF 寄存器，包括 Y 缓冲区的基地址 (y_base)，Y 缓冲区大小 (y_buf_size)，和 Y 缓冲区水线 (y_empty_wm)；
- 根据需求选择是否需要配置中断或者 DMA 使能；
- 配置 PARAM 寄存器，启动 fir 运算；

如果在 X1 缓冲区中预装的值小于或等于N+d-$2^{x1\_full\_wm}$，X1 缓冲区满标志保持低位。如果写数据中断使能，那么将会请求写数据中断，请求处理器通过 **fifo_data** 寄存器向 X1 缓冲区中写入$2^{x1_full_wm}$个输入数据。中断会一直被声明，直到 X1 缓冲区满标志被拉高。中断服务程序应该在每次向 **fifo_data** 寄存器写入个$2^{x1_full_wm}$个数据后检查满标志，如果为低则继续写入下一组输入数据，直到满标志变高，然后清中断。类似地，如果 DMA 写数据使能，DMA 写通道请求也会产生，直到满标志拉高。

当至少有 N 个输入数据被写入 X1 缓冲区 (包括预加载的数据) 时，滤波器计算第一个输出数据。

当 Y 缓冲区写入$2^{y_empty_wm}$个输出数据时，Y 缓冲区的空标志被拉低。如果读数据中断使能，则将会请求读数据中断，请求处理器通过 fifo_data 寄存器从 Y 缓冲区中读取$2^{y_empty_wm}$个输出数据，直到空标志拉高为止。中断服务程序应该在每次从 **fifo_data** 寄存器读取$2^{y_empty_wm}$个数据之后检查空标志，如果为低则继续读取下一组数据，直到空标志变高，然后清中断。如果如果 DMA 读数据使能，则会生成 DMA 读通道请求，直到空标志拉高为止。

滤波器继续以这种方式工作，直到软件拉低 cfg_enable 时暂停。

30.4.9 用 FMAC 实现 IIR 滤波器

FMAC 支持长度为 N 的 IIR 滤波器，其中 N 是前馈抽头或系数的数量。反馈系数的数量 M 可以是 1 到 N-1 之间的任意值。只能实现直接形式 1，因此其他形式的滤波器需要转换。

具有 N 个前馈系数和 M 个反馈系数的 IIR 滤波器的最小内存要求为 2N+2M+1:

- N+M 个系数
- N 个输入数据
- M+1 个输出数据

如果 M = N-1，则可以实现的最大滤波器长度为 N = 64。

对于 IIR，为了提高吞吐量，输入和输出缓冲区大小应分别分配少量额外空间 d1 和 d2，使总内存需求 2M+2N+d1+d2。

缓冲区必须配置如下:

- x1_buf_size = N+d1;
- x2_buf_size = N+M;
- y_buf_size = M+d2;

缓冲区基址可以分配到任何地方，但不能重叠。下面给出了一个配置示例:

- x2_base = 0;
- x1_base = N+M;

- y_base = 2N+M+d1;

注意：

X1 缓冲区的水线应该小于或等于 $\log_2(d1)$，否则在写入 N 个输入样本之前，缓冲区就会被标记为满，并且不再请求写入更多的数据。类似地，Y 缓冲区的水线也必须小于或等于 $\log_2(d2)$。

滤波器系数 (N 前馈，M 反馈) 必须使用加载 X2 缓冲区功能预先加载到 X2 缓冲区中。X1 缓冲区可以预装任意数量的数据，最大可达 N+d1。Y 缓冲区也可以预装任意数量的数据，最大为 M，实现初始化反馈延迟线的效果。

IIR 的配置与 FIR 类似，在加载 X2 缓冲区时 q 参数不为零 (FIR 的 q 参数为 0)，多了一个加载 Y 缓冲区的配置流程。FMAC 通过以下配置流程来启动 IIR 滤波器：

- 配置 CR 寄存器，主要配置 cfg_clr 位域复位内部逻辑，其他位域 (DMA 使能和 clipen 选择等) 根据需要配置；
- 清中断 (避免上次操作结束时未清中断，导致中断上报寄存器中的状态错误)；
- 根据滤波器的阶数配置 X1_BUF 寄存器，包括 X1 缓冲区的基地址 (x1_base)，X1 缓冲区大小 (x1_buf_size)，和 X1 缓冲区水线 (x1_full_wm)；
- 配置 PARAM 寄存器，加载 X1 缓冲区；
- 根据滤波器的阶数配置 X2_BUF 寄存器，包括 X2 缓冲区的基地址 (x2_base)，X2 缓冲区大小 (x2_buf_size)；
- 配置 PARAM 寄存器，加载 X2 缓冲区；
- 根据滤波器的阶数配置 Y_BUF 寄存器，包括 Y 缓冲区的基地址 (y_base)，Y 缓冲区大小 (y_buf_size)，和 Y 缓冲区水线 (y_empty_wm)；
- 配置 PARAM 寄存器，加载 Y 缓冲区；
- 根据需求选择是否需要配置中断或者 DMA 使能；
- 配置 PARAM 寄存器，启动 IIR 运算；

如果在 X1 缓冲区中预装的值小于或等于$N+d-2^{x1_full_wm}$，X1 缓冲区满标志保持低位。如果写数据中断使能，那么将会请求写数据中断，请求处理器通过 fifo_data 寄存器向 X1 缓冲区中写入$2^{x1_full_wm}$个输入数据。中断会一直被申请，直到 X1 缓冲区满标志被拉高。中断服务程序应该在每次向 fifo_data 寄存器写入个$2^{x1_full_wm}$个数据后检查满标志，如果为低则继续写入下一组输入数据，直到满标志变高，然后清中断。类似地，如果 DMA 写数据使能，DMA 写通道请求也会产生，直到满标志拉高。

当至少有 N 个样本被写入 X1 缓冲区 (包括任何预加载的样本) 时，滤波器计算第一个输出数据。第一个数据是使用 X1 缓冲区中的前 N 个数据和 Y 缓冲区中的前 M 个数据进行计算的。第一个输出数据被写入到 Y 缓冲区的 y_base+M 位置。

当 Y 缓冲区写入$2^{y_empty_wm}$个输出数据时，Y 缓冲区的空标志被拉低。如果读数据中断使能，则将会请求读数据中断，请求处理器通过 fifo_data 寄存器从 Y 缓冲区中读取$2^{y_empty_wm}$个输出数据，直到空标志拉高为止。中断服务程序应该在每次从 fifo_data 寄存器读取$2^{y_empty_wm}$个数据之后检查空标志，如果为低则继续读取下一组数据，直到空标志变高，然后清中断。如果如果 DMA 读数据使能，则会生成 DMA 读通道请求，直到空标志拉高为止。

滤波器继续以这种方式工作，直到软件拉低 cfg_enable 时暂停。

30.4.10 滤波器初始化例子

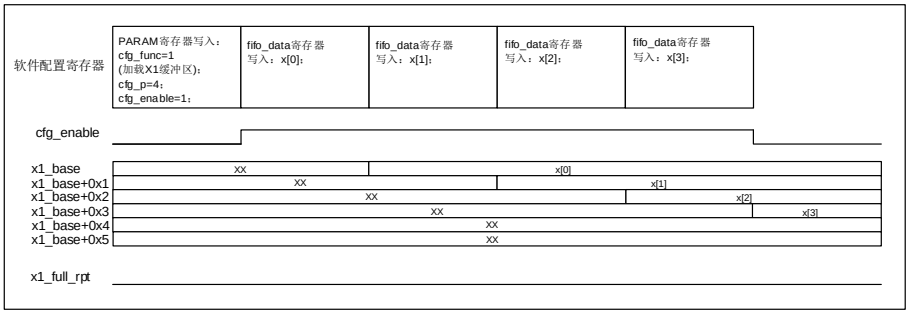


图30-9 X1 缓冲区初始化

上图展示了一个 X1 缓冲区预加载的例子，加载 4 个输入数据 (cfg_p＝4)，缓冲区大小为 6 (x1_buf_size＝6)。通过 PARAM 寄存器中 cfg_enable 位域来启动初始化，通过 fifo_data 寄存器从 x1_base 地址开始向本地内存写入 4 个数据。在写完第四个数据之后，内部写指针指向 x1_base+0x4，cfg_enable 位被清零。

30.4.11 滤波器运算例子

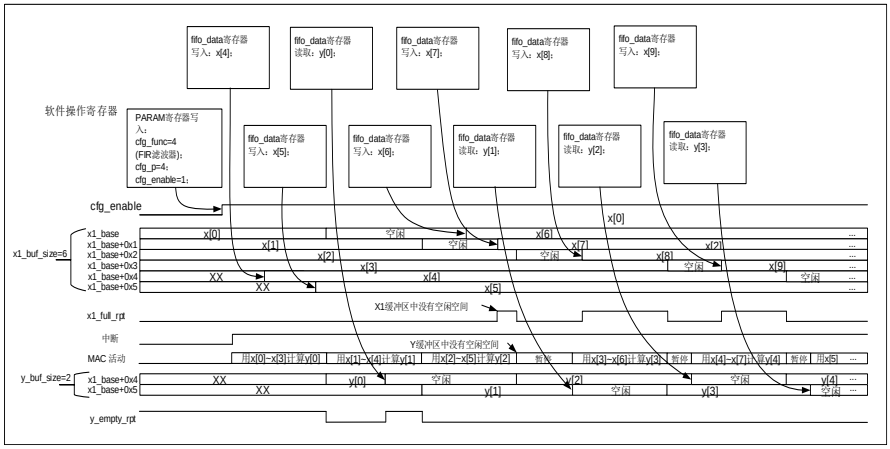


图30-10 滤波器运算过程

上图显示了滤波器的运算过程。滤波器有四个抽头 (cfg_p=4)。X1 缓冲区大小为 6，Y 缓冲区大小为 2，X1 缓冲区水线和 Y 缓冲区水线都设置为 1。在启动滤波器之前，X1 缓冲区已经预加载了四个样本 x[0]~x[3]，如图 30-9 所示。因此，在使能 cfg_enable 之后，滤波器立即开始计算第一个输出数据 y[0]。因为 X1 缓冲区中有两个未初始化的空间，故 X1 缓冲区满标志为低，同时中断也会被声明，以请求新数据。处理器通过 fifo_data 寄存器将两个新的数据 (x[4]和 x[5]) 写入到 X1 缓冲区中的空位置。

在此期间，FMAC 计算完第一个输出数据 y[0]，并将其写入 Y 缓冲区，导致 Y 缓冲区的空标志拉低；同时 X1 缓冲区中的 x[0]数据不再需要，将会被丢弃，从而释放掉它在内存中的空间 (x1_base 地址)。由于 x[1]~x[4]都在 X1 缓冲区中，故 FMAC 可以继续运算第二个输出数据 y[1]。

在处理器向 X1 缓冲区写入 x[5]之后，由于 Y 缓冲区的空标志为低，所以中断仍然是有效的。处理器通过 fifo_data 寄存器从 Y 缓冲区中读取 y[0]，释放它在 Y 缓冲区中的空间。现在输出缓冲区中没有可读数据，因为 y[1]仍在计算中，所以 Y 缓冲区的空标志拉高。尽管如此，中断仍然是有效的，因为 X1 缓冲区中有空闲空间，处理器接下来将会向 X1 缓冲区写入 x[6]，以此类推。

30.4.12 滤波器设计技巧

FMAC 结构对数字滤波器的设计施加了一些限制，详细说明如下：

- 直接 2 型或转置型的实现效率不高，这些类型的滤波器应该转换为直接 1 型。
- 级联滤波器必须组合成一个单独的级，或者用分离的滤波器实现。在后一种情况下，可以将多组过滤器系数预加载到内存中，每级一组，并且只更改 x2_base 地址以选择使用哪一组。
- 对于 IIR 设计，使用直接 1 型可能导致累加器中出现较大的正或负部分和，例如，如果输入上出现较大的步进，或者某些滤波器系数的绝对值大于 1。由于累加器被限制为 26 位，它在不 wrapping (改变符号) 的情况下可以处理的最大值是 0x1FFFFFF (最大正数) 或 0x2000000 (最大负数)。这分别对应于 s(26,22)定点格式中的+7.99999976 和-8。如果在累加结束之前“撤销”了 wrapping，则 wrapping 不代表有问题。在输出缓冲区中预加载合适的值可以避免这种情况。
- IIR 滤波器具有前馈(分子)系数$[b_0,b_1,\cdots,b_{N-1}]$，反馈(分母)系数$[1,a_1,\cdots,a_M]$。许多 IIR 滤波器要求某些分母系数的绝对值大于 1，以实现急剧滚降的频率响应，通过将分母系数乘以因子2^{-R}来实现，使得$2^{-R}\cdot[1,a_1,\cdots,a_M]$都小于 1。然而，必须在累加器的输出端应用$2^R$的逆增益来补偿缩放，这对信噪比有不利影响。

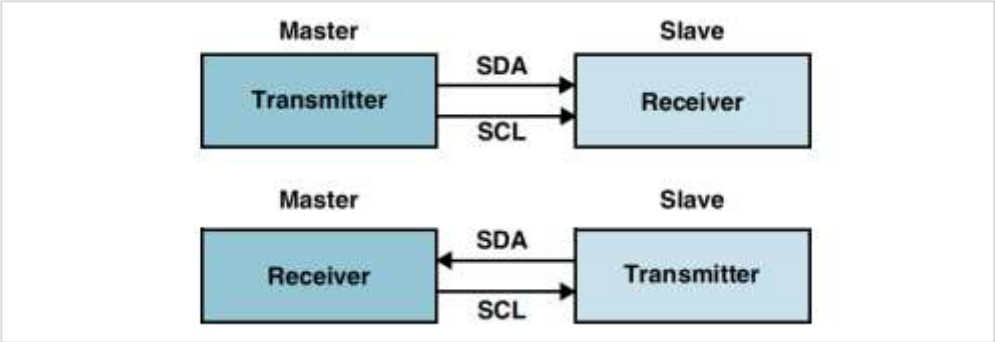
30.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 FMAC 章节。

31 I2C

31.1 简介

ET6002 内部集成 2 个 I2C 控制器接口 IP；提供了 ET6002 与串行 I2C 总线间的通信。I2C 提供多主模式功能，可以控制所有 I2C 总线特定的序列、协议、仲裁和时序，还与 SMBus (系统管理总线) 和 PMBus (电源管理总线) 兼容。



31.2 结构框图

I2C 结构框图如下：

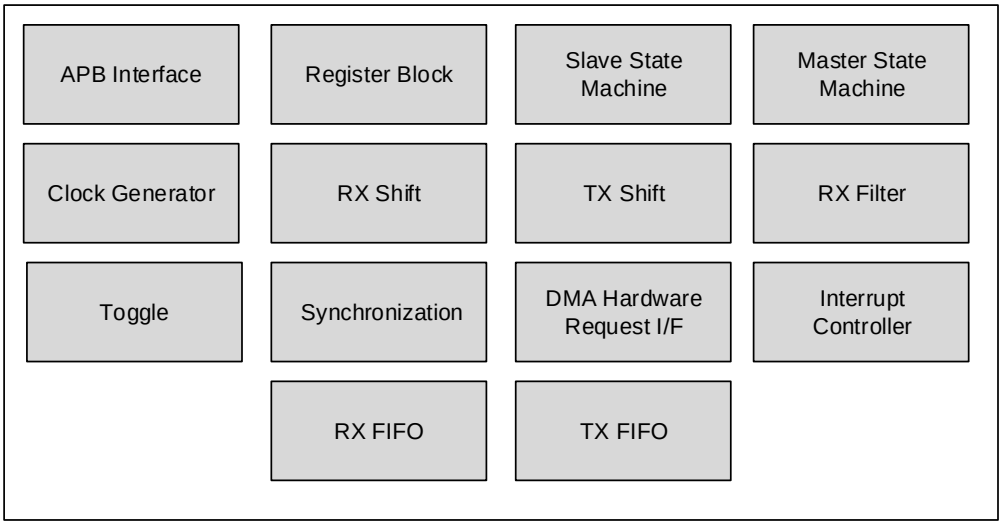


图31-1 I2C 结构框图

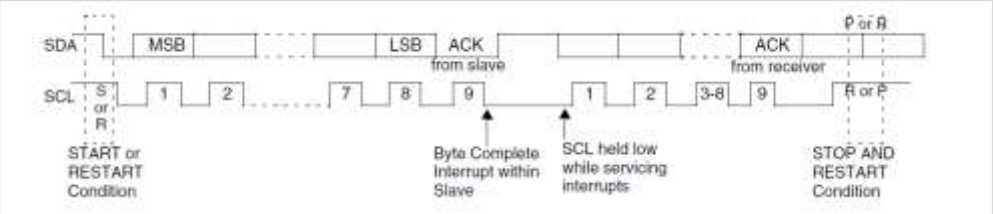
31.3 主要特性

主要特性如下：

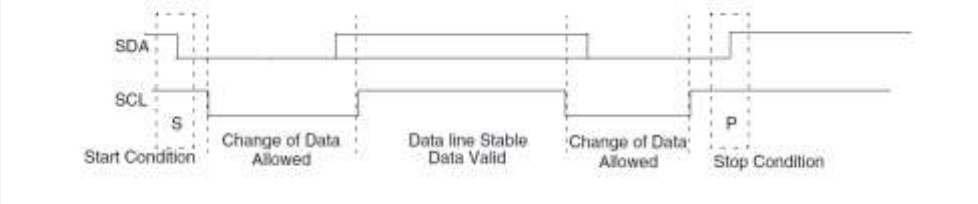
- 支持三种 I2C 接口速率：Standard (100 kbit/s)，Fast (400 kbit/s)，Fast Plus (1 Mbit/s)；
- 支持 SMBus V3.0 和 PMBus V1.2 协议标准；
- 支持 I2C 器件 Master 模式和 Slave 模式软件可配置；
- 支持 7 位和 10 位寻址；
- 支持 SCL，SDA 输入毛刺滤波模式，毛刺滤除宽度可以编程设置；
- 支持基于内部接收和发送 FIFO 数据水线触发中断，接收和发送 FIFO 深度为 16，FIFO 宽度为 8bit；
- 可以在所有速度模式下处理 bit 和 byte waiting；
- 支持 SDA 保持时间可编程；
- 支持总线清零 (Bus Clear) 特性；
- 支持硬件 DMA 握手请求；

31.4 功能描述

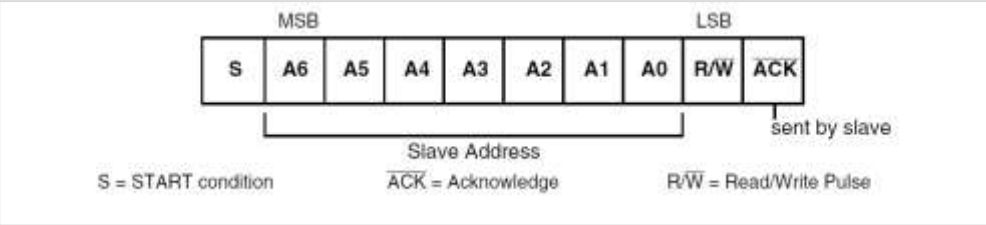
31.4.1 I2C 帧格式

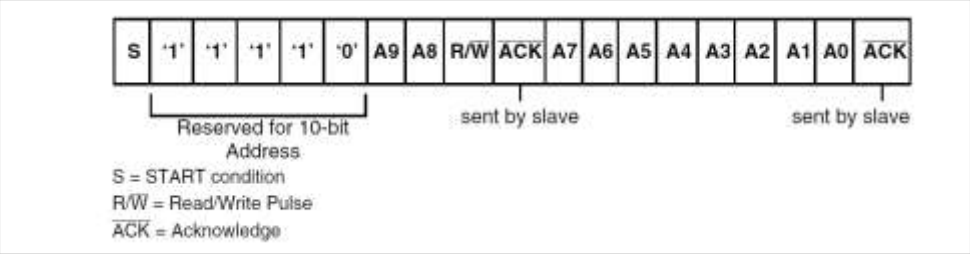


Start 位和 Stop 位示意

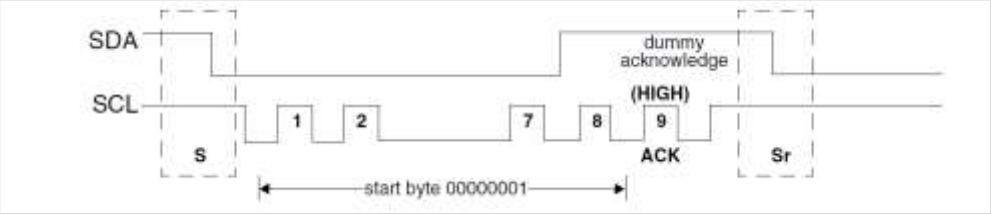


31.4.2 7-Bit 和 10-Bit 地址排布

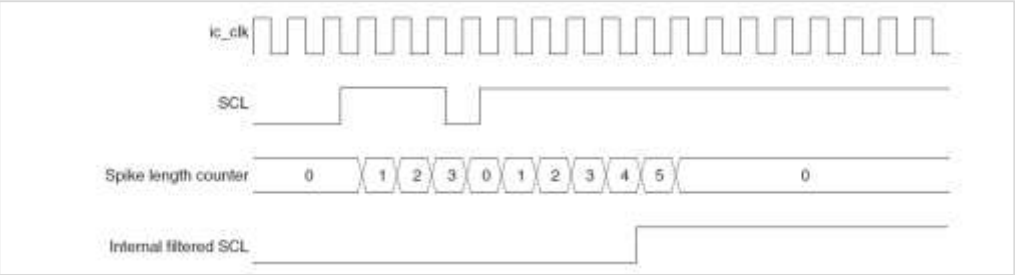




31.4.3 基于 Start Byte 的通信时序



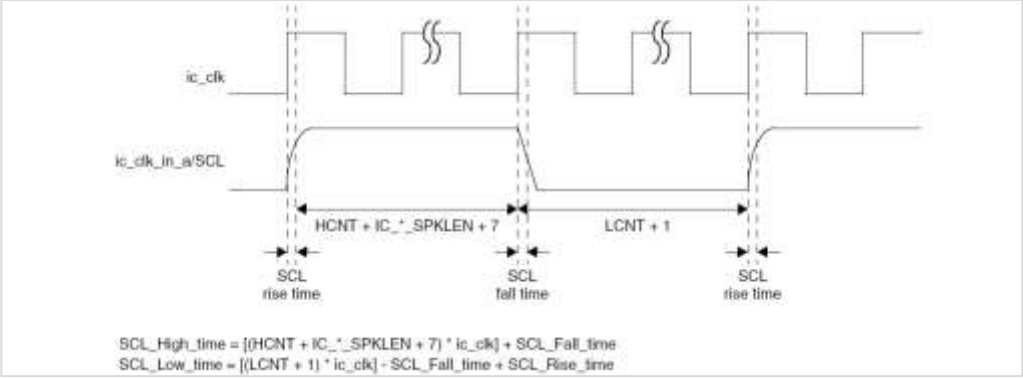
31.4.4 SCL，SDA 输入毛刺滤波功能



FS，HS 两种速率模式下有独立的 IC_FS_SPKLEN，IC_HS_SPKLEN 设置 SCL，SDA 输入滤除毛刺的宽度，不区分 SCL，SDA。

31.4.5 SCL 工作速率设置

SS，FS，HS 三种速率模式下有独立的 SCL HCNT，LCNT 设置高低电平持续时间 ic_clk = 100 MHz，下图中 SCL_Fall_time，SCL_Rise_time 可以忽略。



31.4.6 作为 I2C 从设备使用

通过将“0”写入 IC_ENABLE 寄存器的位 0 来禁用 I2C。

写入 **IC_SAR** 寄存器 (位 9:0) 以设置 Slave 地址。

写入 **IC_CON** 寄存器以指定支持的寻址方式 (7 位或 10 位寻址方式)，**IC_SLAVE_DISABLE=0，MASTER_MODE=1**。

在 **IC_ENABLE** 寄存器的 ENABLE=1 来启用 I2C。

31.4.7 作为 I2C 主设备使用

通过将“0”写入 IC_ENABLE 寄存器的位 0 来禁用 I2C。

配置 IC_CON 寄存器，设置 **IC_SLAVE_DISABLE = 1，IC_RESTART_EN = 1，IC_10BITADDR_MASTER** (7 位或 10 位寻址方式)，**IC_10BITADDR_SLAVE** (7 位或 10 位寻址方式)，**IC_MASTER_MODE = 1**。

设置相关速率模式下的 IC_XXX_SCL_HCNT，IC_XXX_SCL_LCNT 设置 SCL 信号持续高，低电平的时间 (单位为 100 MHz 时钟的个数)。

设置 IC_INTR_MASK 使能相关中断。

设置 IC_RX_TL，IC_TX_TL RXFIFO 和 TXFIFO 水线。

在 IC_ENABLE 寄存器的 ENABLE=1 来启用 I2C。

将待发送的数据写入 TX FIFO，I2C 将数据通过 I2C 总线发送给从设备。

31.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 I2C 章节。

32 UART

32.1 简介

ET6002 提供 2 组 UART 接口，定义为 UART0~1。支持 RS485 以及 2 线普通串口以及带流控的串口模式。

32.2 结构框图

UART 结构框图如下：

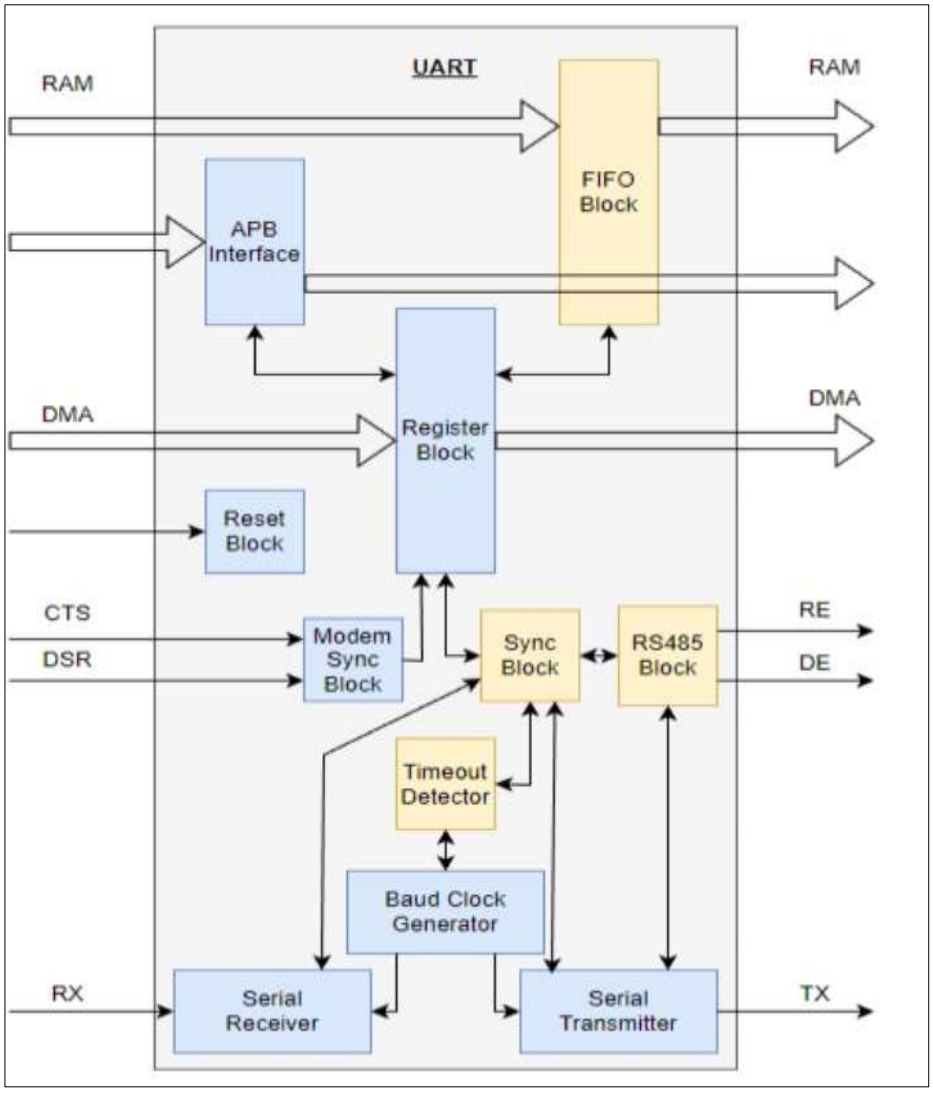


图32-1 UART 结构框图

32.3 主要特性

主要特性如下：

- 支持 UART 最高波特率≥1 Mbaud；
- 支持 9 位串行数据；
- 错误起始位检测，支持检测中断上报；
- 输入参考时钟为 100M，支持可编程波特率发生器，可对 UART 输入的 sclk 进行固定 16 分频后，再进行 16bit 整数位分频以及 5bit 小数位分频，以支持可编程分数波特率；
- 可以配置为软件可编程 RS485 模式，支持多点 RS485 接口。如果未使能此模式，则仅 UART (RS232 标准) 串行数据格式通信；
- 支持基于内部接收和发送 FIFO 数据水线触发中断，TX，RX FIFO 各是 16Bytes；
- 支持 FIFO Test 模式 (Master 向 TX FIFO 写发送数据，从 RX FIFO 中读接收数据)，即 TX–>RX Loopback 测试；
- 兼容 ISO 16750 标准的自动流量控制模式来提高系统效率并减少软件负载；
- 支持硬件 DMA 握手请求；

32.4 功能描述

32.4.1 波特率配置参数计算方法

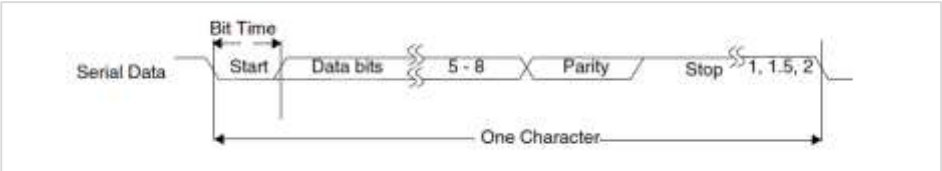
以 UART1 工作在 115200 波特率为例，UART1 输入的参考时钟为 100 MHz。

分频参数 = 100*1000000/(16*115200) = 54.2534722

0.2534722*32=8，则 UART 的整数分频参数为 54 (DLL=54,DLH=0)，小数分频参数为 8 (DLF=8)。

32.4.2 UART 串行帧格式

软件可编程，数据支持 5~8 bit 长度，支持 STOP 位设置 1, 1.5, 2 bits。



32.4.3 RS485 模式串行帧格式

32.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 UART 章节。

33 CAN FD

33.1 简介

CAN FD 是 CAN 总线的升级版本，向下兼容 CAN1.0，CAN2.0 协议版本。

CAN FD 在 CAN 协议的基础上，通过支持新的帧格式，提高了 CAN 总线的通信速率，由 CAN2.0 的 1 Mbps 提升到 CAN FD 的 8 Mbps。

33.2 结构框图

CAN FD 结构框图如下：

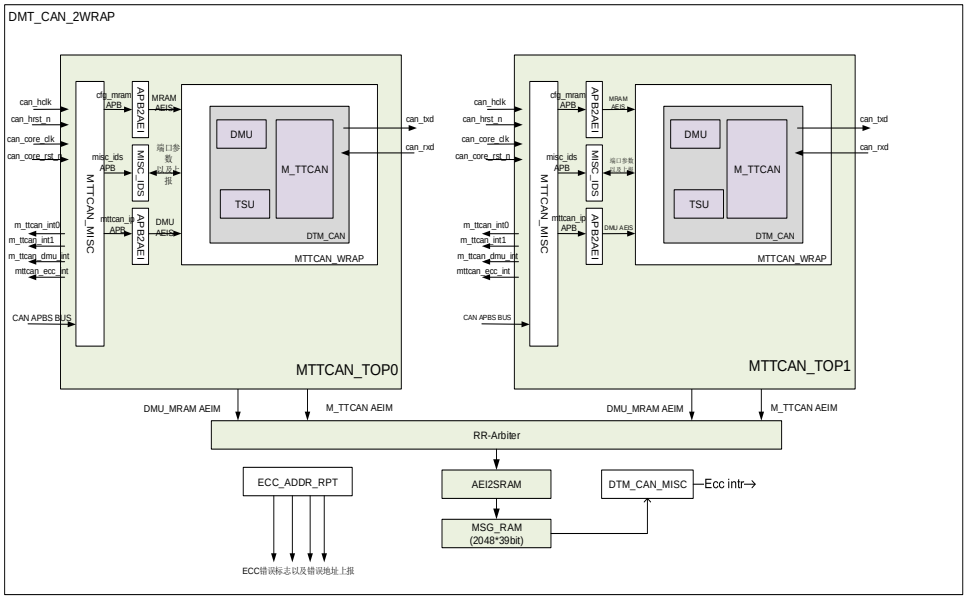


图33-1 CAN FD 结构框图

33.3 主要特性

主要特性如下：

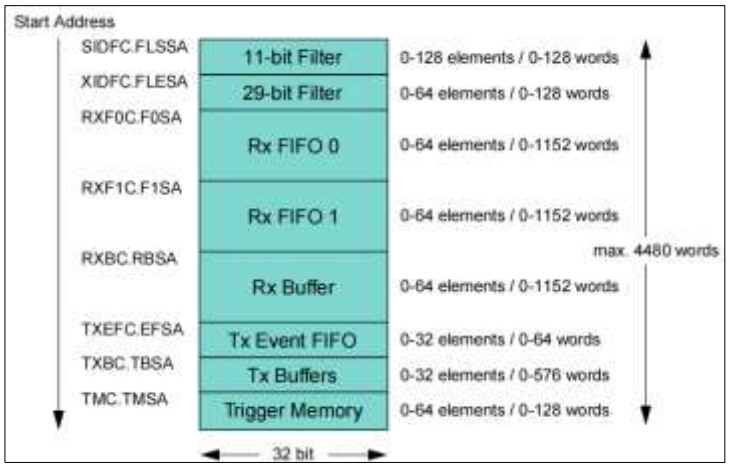
- 遵从 ISO 11898-1:2015，ISO 11898-4 标准；
- CAN FD 模式下最大支持 64-Byte 帧格式；
- 每路 CAN 设备只能作为 1 个 Node 器件接入 CAN 总线网络使用；
- 硬件支持 TTCAN Level1，Level2 协议；
- 支持基于同步事件触发的通信；
- CAN 通信过程错误记录；

- 支持 SAE J1939 BOSCH 标准；
- 支持优化的接收帧数据滤波机制；
- 2 个 CAN IP 共享消息 RAM 大小为 8 KB，支持 SECDED (单 bit 纠错，两 bit 检错) 的 ECC 保护机制，支持访问错误地址上报；
- 内部含有 2 个深度可配置的接收 FIFO；
- 支持区分接收过程中的高优先级数据帧消息；
- 支持最大 64 元素 (Element) 的接收缓存；
- 支持最大 32 元素 (Element) 的发送缓存；
- 支持 1 个深度可配置的发送 FIFO；
- 支持可配置的发送队列机制；
- 支持 1 个深度可配置的事件发送 FIFO；

33.4 功能描述

33.4.1 软件可配置的 MSG RAM

CAN 顶层集成了一块 2K word (8KB) 的共享 MSG RAM，这块 RAM 可以按如下结构由软件划分各个部分的存储空间。所以各个空间总的大小不能超过 2K word (8KB)。



33.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 FDCAN 和 FDCAN_MISC 章节。

34 调试接口

34.1 简介

ET6002 存在 1 套 SWD 调试接口，用于 CPU 的硬件调试，互不影响。

SWD 接口遵从 ARM Debug Interface Architecture Specification V5.2 标准。

34.2 结构框图

标准中定义 SWD 接口实际只有两根线，分别为 SWCLK，SWDIO；

其中:

- SWCLK (ET6002 Pin 脚名为 CPU_SWJ_SWCLKTCK) 是一个单向信号，由仿真器提供；
- SWDIO (ET6002 Pin 脚名为 CPU_SWJ_TMS) 是双向信号，由仿真器和 ET6002 分时驱动。

ET6002 和仿真器连接关系示意图：

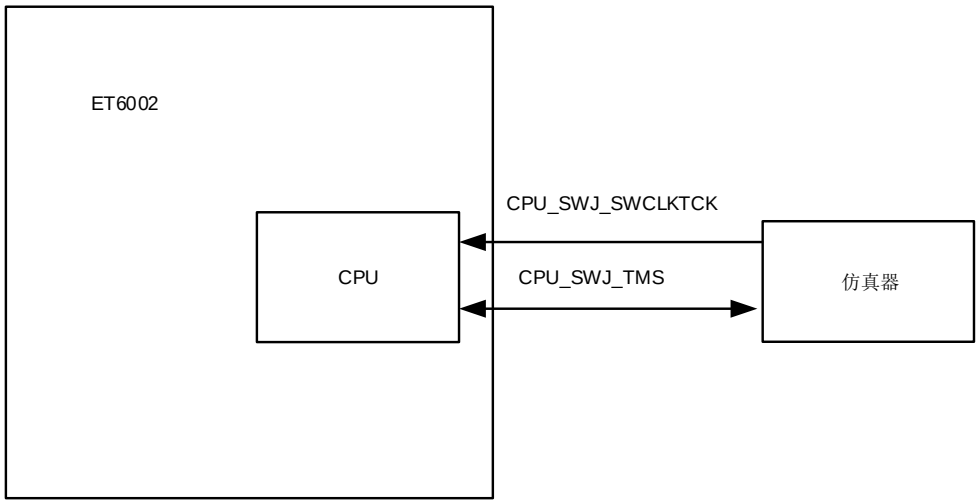


图34-1 ET6002 和仿真器连接关系示意图

34.3 主要特性

主要特性如下：

- 支持 SWD 通过 CPU 的调试口经 CPU 访问总线接口方式访问内部 IP 寄存器；
- 支持 SWD 通过 DP 的方式访问 DP 内部寄存器；
- 支持 SWD 写操作；

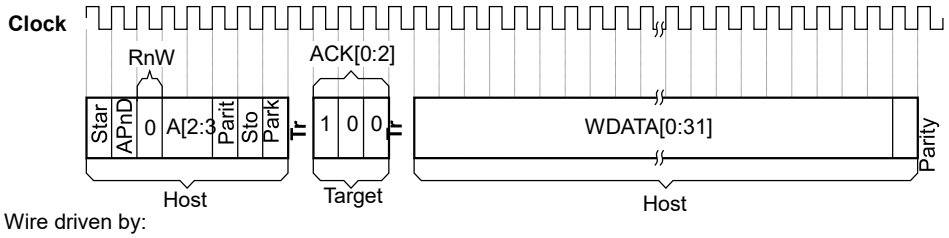
- 支持 SWD 读操作；

34.4 功能描述

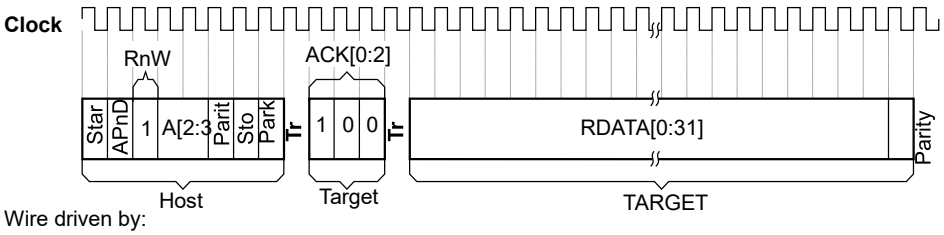
SWD 协议介绍：

符号	含义
Start	START 位，固定为 1
APnDP	AP，DP 访问操作指示，1 表示 AP 操作，0 表示 DP 操作
RnW	读写操作指示，1 表示读操作，0 表示写操作
A[2:3]	A[3:2]对应 DP 或者 AP 的目标寄存器地址，低位在前
Parity	奇偶校验位
Stop	STOP 位，固定为 0
Park	Park 位，固定为 1
Trn	Trunaround 位，HOST 和 DEVICE 切换
ACK[0:2]	Device 对命令的 ACK 应答，低位在前 ACK[0:2]编码含义 0b100: OK 0b010: WAIT 0b001: FAULT
WDATA[0:31]	HOST 向 DEVICE 的写数据，低位在前
RDATA[0:31]	DEVICE 向 HOST 的读数据，低位在前

- **SWD 接口写操作时序**



- **SWD 接口读操作时序**



35 WAG

35.1 简介

WAG（Wave Generator）模块通过软件配置产生各种特定波形，经过 DAC 输出。

35.2 结构框图

WAG 结构框图如下：

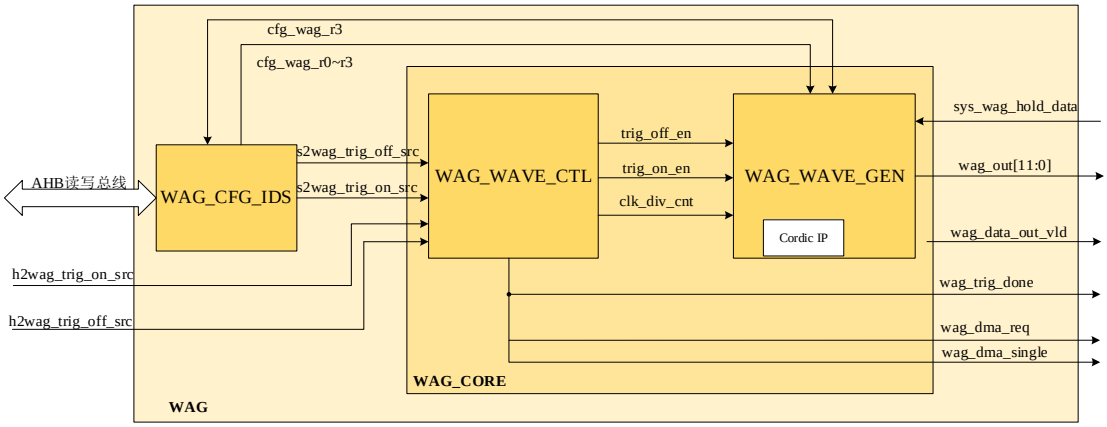


图35-1 WAG 结构框图

35.3 主要特性

主要特性如下：

- 支波方波
- 支持递增和递减波形（RAMP 波）
- 支持三角波和锯齿波
- 支持正弦波

35.4 功能描述

35.4.1 方波

预设 2 个输出码值，通过触发启动信号改变翻转 DAC 输出电平。

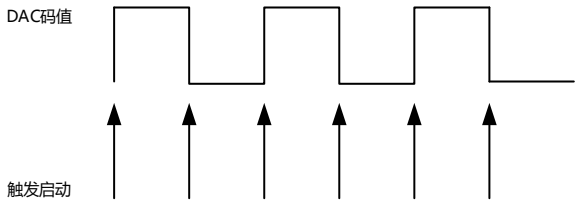


图35-2 非连续方波形示意图

方波除支持单次触发模式外，还支持连续输出模式，触发一次后无需后续触发即可连续输出对应波形，直到触发终止信号到达，方波回到下门限停止翻转。

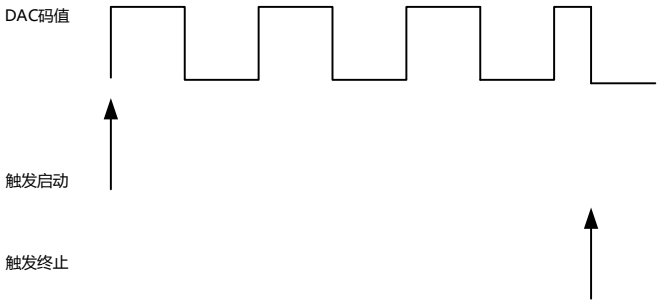


图35-3 连续方波形示意图

35.4.2 递增和递减波形（RAMP 波）

触发启动信号有效，加载相关配置，从下/上门限启动，每个计数时钟累加/减上/下行步进，直到达到上/下门限后保持（如累加/减超过 DAC 接口位宽，则进行饱和操作，下同），计数期间，如有新的触发启动有效，则按新的加载参数运行，期间如触发终止信号有效，回到当前下/上门限等待下一次触发启动。

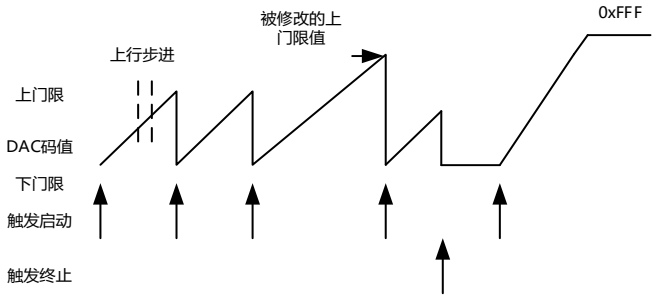


图35-4 递增波形示意图

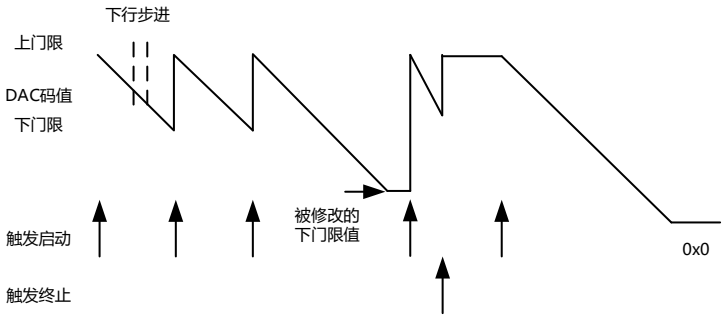


图35-5 递减波形示意图

35.4.3 三角波

触发启动信号有效，加载相关配置，从下/上门限启动，每个计数时钟累加/减上/下行步进，直到达到上/下门限后开始以每个计数时钟累减/加下/上行步进，直到达到下/上门限后保持。期间如收到新的触发启动，处理与递增递减波形处理类似。

期间如果收到触发终止信号，需要等待一个上下行周期结束后才停止：

（1）触发终止信号的优先级大于触发启动优先级。在触发终止信号未被响应前，触发启动信号均被忽略。

（2）当到达下限之后，触发终止信号被响应的同时，触发启动信号到来，此时触发启动信号会被响应。

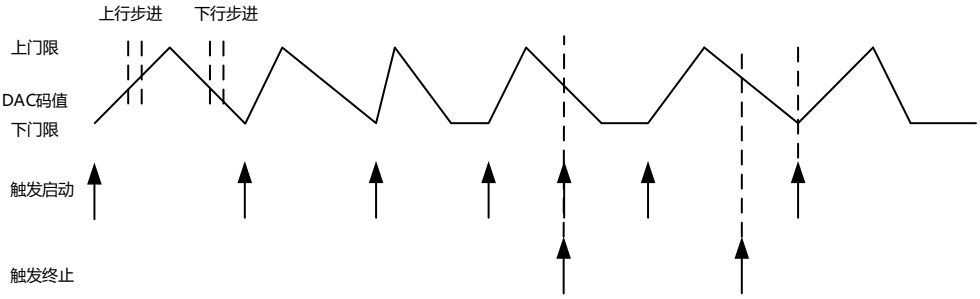


图35-6 非连续翻转三角波示意图

35.4.4 锯齿波

三角波形通过设置上/下步进，可得到锯齿波。如：上步进不做改变，下步进改为 ceil（（上门限-下门限）/下门限）*下门限，则触发启动信号有效时，从下门限启动，每个计数时钟按照上步进进行累加，到达上门限之后，下一个计数时钟减下步进，回到下门限后保持。

注：ceil 表示向上取整

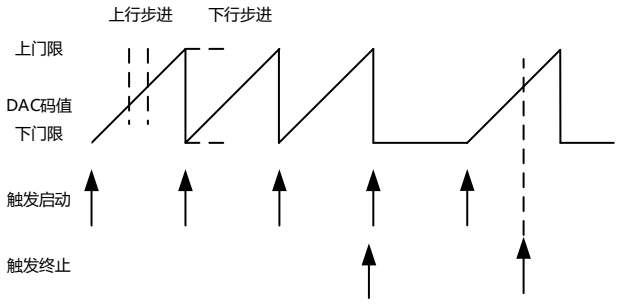


图35-7 非连续翻转锯齿波示意图

三角波和锯齿波除支持单次触发模式外，还支持连续输出模式，触发一次后无需后续触发即可连续输出对应波形。当幅值经过一个上下行周期回到初始门限后，自动启动下一个周期的波形产生，波形参数保持不变，直到触发终止回归初始门限。

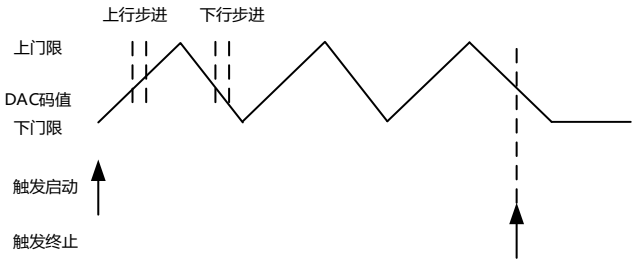


图35-8 连续翻转三角波示意图

35.4.5 正弦波

仅支持类似三角波和锯齿波的连续输出模式。

正弦波通过配置的初相、频率控制字（相位累加值）和输出幅值来产生，然后叠加配置的直流幅值（仅支持 1/2N 幅度调整，N=0,…,12）后输出。正弦波频率f_{sin}与系统时钟f_{sys}、分频比、频率控制字 FCW 和输出位宽 N 的关系如下：

$$f_{sin} = \frac{f_{sys}.FCW}{(cfg\_wag\_clk\_div\_para + 1).2^N}$$

波形在触发启动后开始输出，触发终止后输出配置直流电平幅值。采用 Cordic 算法 IP 产生正弦波。

Cordic IP 的正弦函数输入输出变量如表 1-1 所示。

表35-1 正弦（Sin）函数

参数	描述	值域
输入变量 1	角度θ，单位为π*弧度(1)	[−1,1]
输入变量 2	幅度 m/k	[0,1]
输出结果 1	m*sinθ(2)	[−1,1]
输出结果 2	m*cosθ	[−1,1]
比例系数	NA	0

注释 1：该输入变量需要预处理，将弧度值除以 π 得到，作为输出结果时，需要乘以 π 得到弧度值。

注释 2：k 为 Cordic IP 计算后产生的缩放因子，与计算精度相关，约为 1.6。

注释 3：WAG 调用 Cordic IP 数据格式 s(16,15)数据格式，精度为 2^(-15)（大约 3×10^(-5)）。

表35-2 s(16,15)数据格式

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S	C_{14}	C_{13}	C_{12}	C_{11}	C_{10}	C_9	C_8	C_7	C_6	C_5	C_4	C_3	C_2	C_1	C_0

35.5 寄存器定义

寄存器列表和详细描述，参见附件《ET6002-用户手册_寄存器定义》中的 WAG 章节。

修订记录

修订记录累积了本文档每次修订的具体信息。

序号	版本	修改日期	修改说明
1	1.0.0	2024/09/06	初始发布
2	1.0.1	2025/01/21	根据软件的反馈刷新对应章节; 删除 sarc 中的 fir 滤波器相关的描述； 增加 ACMP 中异步路径相关的约束； 增加 WAG 章节的用户手册；
3	1.0.2	2025/03/06	19.4.9.3 PWM 时基模块参数的同步加载章节，说明事项中添加 3、4 两条，约束启用时基同步信号 0 时，**CFG_TB_0PHASE[cfg_tb_0phase]** 和 **CFG_TB_CTL[cfg_tb_0dir]**不能同时配置为 0 值；启用时基同步信号 1 时，**CFG_TB_1PHASE[cfg_tb_1phase]** 和 **CFG_TB_CTL[cfg_tb_1dir]**不能同时配置为 0 值。另外对说明 1、2 的描述做调整，使其更明晰。
4	1.0.3	2025/03/25	19.4.7.2.3 关键参数的通道间同步加载模式章节，修改了注意事项的第二段描述，使其更明晰，无歧义。
5	1.0.4	2025/03/31	5.1 复位章节增加复位源说明