XMC1300

工业应用微控制器系列

XMC1000 家族

ARM® Cortex™-M0
32 位处理器核

参考手册
V1.0 2013-03

微控制器

2013-03 版
英飞凌科技股份有限公司出版
81726 慕尼黑，德国

法律声明

本文档中的信息在任何情况下都不应被视为一种条件或特性的保证。英飞凌科技公司在此声明，对于本文档给出的任何实例或提示、任何典型值和 / 或任何与器件应用相关的信息，英飞凌科技股份公司也不承担任何保证和责任，包括但不限于，保证不侵犯任何第三方的知识产权。

信息

有关技术方面的进一步信息、交货条款以及条件和价格，请联系最近的英飞凌科技公司办事处 (**www.infineon.com**)。

Warnings

由于技术需要，元件可能包含危险物质。如有问题请联系最近的英飞凌科技公司办事处。如果有理由预料英飞凌科技公司元件的故障会导致生命支持设备或系统的故障，亦或影响其安全性或效果，则只有在得到英飞凌科技股份公司书面批准的情况下，才能在这类设备或系统中使用这些元件。生命支持设备器件或系统的目的是植入人体或支持和 / 或维护以及维持和 / 或保护人的生命。如果这些设备出现故障，可以有理由假设，用户或其他人可能受到危害。

XMC1300

工业应用微控制器系列

XMC1000 家族

ARM® Cortex™-M0
32 位处理器核

参考手册
V1.0 2013-03

微控制器

XMC1300 参考手册

修订历史：V1.0 2013-03

先前的版本：
无

页	主题

商标

C166™、 TriCore™ 和 DAVE™ 是英飞凌科技股份有限公司的商标。

ARM®、 ARM Powered® 和 AMBA® 是 ARM 有限公司的注册商标。

Cortex™、 CoreSight™、 ETM™、 Embedded Trace Macrocell™ 和 Embedded Trace Buffer™ 是 ARM 有限公司的商标。

我们期待您的指正
如果您发现本文档有错误、不明确或遗漏之处，敬请反馈和指正，以帮助我们不断改善本文档的质量。请将您的建议（包括需修改处在本文档中的对应位置）发送至：
mcdocu.comments@infineon.com

目录

目录

目录

目录

目录

目录

目录

关于本文档

本参考手册是为嵌入式硬件和软件开发人员编写的，为读者提供有关 XMC1300 系列各功能单元行为及其交互作用的详细描述。

本手册描述 XMC1300 微控制器系列的超集器件的功能。有关某一特定 XMC1300 衍生品 (衍生器件) 的可用功能 (特性)，请参见相应的数据手册。为简单起见，本手册对各种不同器件型号都使用集合术语 XMC1300。

XMC1000 家族用户文档

这组用户文档包括：

- **参考手册**
  - 描述超集器件的功能。
- **数据手册**
  - 列出衍生器件的完整订购信息、可用功能和电气特性。
- **勘误表**
  - 列出与相关参考手册或数据手册所给出的技术规格的偏差之处。勘误表针对超集器件提供。

注意：　请查阅本组文档的所有部分，以对您使用的器件有综合的认识。

与应用相关的指导由**用户指南**和**应用笔记**提供。

请访问 **http://www.infineon.com/xmc1000** ，以获得这些文档的最新版本。

相关文档

以下是相关的参考文档：

- ARM® Cortex M0
  - 技术参考手册
  - 用户指南、参考材料
- ARM®v6-M 体系结构参考手册
- AMBA® 3 AHB-Lite 协议规范
- AMBA® 3 APB 协议规范

版权声明

文本约定

本文档使用下述命名规则：

- XMC1300 的功能单元用简单易懂的大写字母给出。例如：“USIC 单元支持 …”。
- 使用负逻辑的引脚用上划线表示。例如：“WAIT 输入有 …”。

- 寄存器中的位域和位一般表示为 “ 模块 _ 寄存器名 . 位域 ” 或 “ 模块 _ 寄存器名 . 位 ”。例如：“USIC0_PCR.MCLK 位使能 …”。大多数寄存器名包含一个模块前缀，一个下划线字符 “_” 将前缀与实际的寄存器名分开。(例如 “USIC0_PCR”，其中 “USIC0” 是模块名前缀， “PCR” 是内核寄存器的名称)。在描述外设模块内核的章节中，引用寄存器主要使用其内核寄存器名。外设模块的实现部分主要使用带模块前缀的实际寄存器名称。
- 用于描述处理单元组或寄存器组的变量以大写和小写混合的形式出现。例如，寄存器名 “MOFCRn” 指多个 “MOFCR” 寄存器，n 为变量。在第一次使用该寄存器的表述时总是会给出变量的范围（例如，“n = 0-31”），在文档的其他部分也会根据需要重复给出该范围。
- 默认的数制为十进制。十六进制常数以下标字母 “H” 为后缀，例如 100_H。二进制常数以下标字母 “B” 为后缀，例如 111_B。
- 在文档正文中，当以集合形式定义寄存器域、组寄存器位或组引脚的范围时，将它们表示为 “NAME[A:B]”，这种表示方式定义组的范围是从 B 到 A。单独的位、信号或引脚以 “NAME[C]” 的形式表示，其中变量 C 的范围在正文中给出。例如：CFG[2:0] 和 SRPN[0]。
- 单位缩写如下：
  - **MHz** = 兆赫
  - **µs** = 微秒
  - **kBaud, kbit** = 1000 字符 / 比特每秒
  - **MBaud, Mbit** = 1,000,000 字符 / 比特每秒
  - **Kbyte, KB** = 1024 字节存储器
  - **Mbyte**, MB= 1048576 字节存储器
    一般来说，前缀 k 将单位乘以 1000，而前缀 K 将单位乘以 1024。因此，单位 Kbyte 将其前面的表述乘以 1024。单位 kBaud 将其前面的表述乘以 1000。前缀 M 换算为 1,000,000 或 1048576。例如，1 Kbyte 为 1024 字节，1 Mbyte 为 1024 ×1024 字节，1 kBaud/kbit 为 1000 字符 / 比特每秒，1 MBaud/Mbit 为 1000000 字符 / 比特每秒， 1 MHz 为 1,000,000 Hz。
- 数据格式的位数定义如下：
  - **字节** = 8 位
  - **半字** = 16 位
  - **字** = 32 位
  - **双字** = 64 位

位功能术语

在阐述寄存器位或位域的表中，使用下述约定指示访问类型。

表 1　位功能术语

位功能	描述
rw	位或位域可读可写。
rwh	同 rw，但位或位域也可被硬件置位或复位。在发生硬件和软件写冲突的情况下，如果没有特别说明，则软件写操作优先。
r	位或位域只能被读取 (只读)。
w	位或位域只能被写入 (只写)。读该寄存器将总是返回一个默认值。
rh	位或位域可被硬件修改 (读硬件，典型例子：状态标志)。读该位或位域返回该位或位域的当前状态。写该位或位域无效。

寄存器访问方式

对寄存器和存储器单元的读和写访问有时是受限的。在存储器和寄存器访问表中，使用下面的术语。

表 2　寄存器访问方式

符号	描述
PV (SV), U	允许在特权 （管理）模式下访问。 ***注：　ARM® Cortex M0 处理器不支持不同特权级别。XMC1000 家族只支持特权 (管理) 模式。符号 “U” 和符号 “PV” 可用于表示允许在该模式下访问。***
BP	表示只能在位保护被禁止时访问该寄存器。见存储器组织一章中对位保护机制的详细描述。
32	只允许对该寄存器 / 地址范围进行 32 位字访问。
NC	无变化，表示寄存器未发生改变。
BE	表示访问该地址范围会产生总线错误。
nBE	表示访问该地址范围不会产生总线错误。

保留位

寄存器位域被定义为**保留（Reserved）**或 **0** 表示没有用于实现具体功能，这样的位域具有以下行为。

- 读这些位域返回 0。
- 如果这些位域被定义为 r 或 rh，则应被写入 0。
- 如果这些位域被定义为 rw，必须被写入 0。

这些位域被保留。这些位域的详细说明见寄存器描述部分。

缩略语和首字母缩写词

下面是本文档中使用的首字母缩写词和术语：

ACMP	模拟比较器
AHB	先进的高性能总线
AMBA	先进的微控制器总线架构
ANACTRL	模拟控制单元
APB	先进的外设总线
ASC	异步串行通道
BCCU	亮度和色彩控制单元
BMI	引导模式索引
CMSIS	Cortex 微控制器软件接口标准
CPU	中央处理单元
CCU4	捕获比较单元 4
CCU8	捕获比较单元 8
CRC	循环冗余码
DCO	数控振荡器
ECC	纠错码
ERU	事件请求单元
EVR	嵌入式稳压器
FPU	浮点单元
GPIO	通用输入 / 输出
HMI	人机接口
IIC	集成电路间总线 (亦称 I2C)
IIS	集成电路间音频接口
I/O	输入 / 输出
JTAG	联合测试行动组 = IEEE1149.1
LED	发光二极管
LEDTS	LED 和触摸感应控制单元
MSB	最高有效位
NC	未连接
NMI	不可屏蔽中断
NVIC	嵌套向量中断控制器

ORC	超量程比较器
PAU	外设访问单元
POSIF	位置接口
PRNG	伪随机数发生器
ROM	只读存储器
RAM	随机存取存储器
RTC	实时时钟
SCU	系统控制单元
SFR	特殊功能寄存器
SHS	采样保持定序器
SPI	串行外设接口
SRAM	静态 RAM
SR	服务请求
SSC	同步串行通道
SSW	启动软件
TSE	温度传感器
UART	通用异步收发器
USIC	通用串行接口通道
VADC	通用模 / 数转换器
WDT	看门狗定时器

引言

1 引言

XMC1300 系列属于 XMC1000 工业微控制器家族，该家族基于 ARM Cortex-M0 处理器核。XMC1300 系列器件针对电机控制、电源转换和 LED 照明应用而优化设计。

嵌入式控制应用的复杂性和计算能力需求与日俱增，要求微控制器具有强大的 CPU 性能、集成的外设功能和快速的开发环境，既缩短上市时间，又不影响成本效益。XMC1300 微控制器的体系结构继承了英飞凌微控制器家族长期以来建立起来的硬件和软件理念。

1.1 概述

XMC1300 系列器件将 Cortex-M0 核的扩展功能和性能与强大的片上外设子系统和片上存储器单元整合在一起。下面是 XMC1300 系列器件所具备的关键特性：

CPU 子系统

- CPU 核
  - 高性能 32 位 Cortex-M0 CPU
  - 可执行 16 位 Thumb 指令集中的大多数指令
  - 32 位 Thumb2 指令集的子集
  - 保持 32 位性能的高代码密度
  - 单周期 32 位硬件乘法器
  - 支持操作系统的系统定时器 (SysTick)
  - 超低功耗
- 嵌套矢量中断控制器 (NVIC)
- 事件请求单元 (ERU)，用于外部和内部服务请求的可编程处理
- MATH 协处理器 (MATH)，包含一个可计算三角函数的 CORDIC 单元和一个除法单元

片上存储器

- 8 KB 片上 ROM
- 16 KB 片上高速 SRAM
- 高达 200 KB 的片上 Flash 程序和数据存储器

通信外设

- 两个通用串行接口通道 (USIC)，可作为 UART、2 位 SPI、4 位 SPI、IIC、IIS 和 LIN 接口使用

模拟前端外设

- A/D 转换器，最多可有 12 个通道，包含 2 个采样保持电路和一个增益可调节的快速 12 位模 / 数转换器
- 多达 8 通道的超量程比较器 (ORC)
- 多达 3 个快速模拟比较器 (ACMP)

工业控制外设

- 捕获 / 比较单元 4 (CCU4)，用作通用定时器
- 捕获 / 比较单元 8 (CCU8)，用于电机控制和电源转换
- 位置接口 (POSIF)，用于连接霍尔传感器和正交编码器以及电机定位
- 亮度和色彩控制单元 (BCCU)，用于 LED 色彩控制和调光应用

系统控制

- 窗式看门狗定时器 (WDT)，针对安全敏感型应用
- 实时时钟模块 (RTC)，支持告警功能
- 系统控制单元 (SCU)，用于系统配置和控制
- 伪随机数发生器 (PRNG)，能快速提供随机数据

具有单独位控制能力的输入 / 输出线

- 三态输入方式
- 推挽或漏极开路输出方式
- 可配置的焊盘滞回电压

调试系统

- 通过标准 ARM 串行线调试 (SWD) 接口或单引脚调试 (SPD) 接口访问
- 断点单元 (BPU) 支持最多 4 个硬件断点
- 观察点单元 (DWT) 支持最多 2 个观察点

封装信息

- PG-TSSOP-38
- PG-TSSOP-16

注：　有关某一具体型号的可用封装方面的详细信息，请查阅数据手册。

1.1.1 功能框图

下面的框图示出了 XMC1300 系统内的功能块及其基本连接。

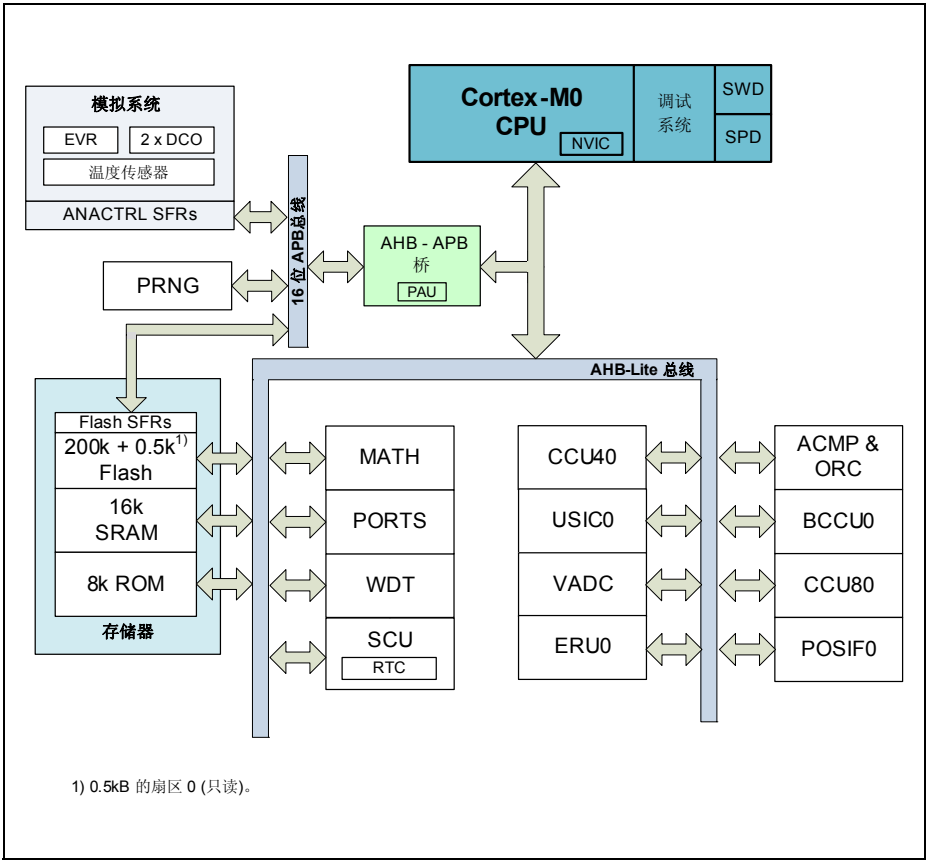


图 1-1 XMC1300 功能框图

1.2 核心处理单元

XMC1300 系统内核包含 CPU 和存储器接口功能块。

1.2.1 中央处理单元 (CPU)

ARM Cortex-M0 处理器基于面积和功耗高度优化的 32 位处理器核而构建，采用具有 3 级流水线的冯 · 诺依曼体系结构。该处理器不仅通过小而强大的指令集和广泛优化的设计提供非凡的能效，而且还提供了包括单周期乘法器在内的高端处理硬件。

该处理器的指令集基于 16 位 Thumb 指令集，并包含了 Thumb-2 技术。该指令集提供了现代 32 位体系结构应有的优异性能，并且有比 8 位和 16 位微控制器更高的代码密度。

1.2.2 可编程多优先级中断系统 (NVIC)

XMC1300 提供多个单独的中断节点，每个节点可以有 4 个中断优先级。大多数中断源都被连接到一个专用的中断节点。在有些情况下，多源中断节点被合并到一起，以有效利用系统资源。这些节点可以被多个源请求激活，并可由中断子节点控制寄存器控制。

1.2.3 数学协处理器 (MATH)

数学协处理器(MATH)模块由两个独立的单元组成，支持CPU完成数学密集型计算：除法器单元 (DIV) 执行有符号和无符号 32 位除法运算， CORDIC (坐标旋转数字计算机) 协处理器执行三角函数、线性或双曲函数计算。

1.3 系统单元

XMC1300 控制器提供了一系列围绕 CPU 而设计的系统资源。

1.3.1 存储器

8 KB ROM 存储器用于执行启动代码和存储异常向量表。该 ROM 包含系统的基本初始化序列代码， CPU 在复位解除后立即从该 ROM 执行程序。引导加载程序 (BSL) 和用户子程序也存储在 ROM 中。

最多可达 200 KB 的片上 Flash 存储器用于存储代码和常量数据。其动态纠错功能为所有读访问提供很高的读数据安全性。

16 KB 的片上代码 RAM (SRAM) 用于存储用户代码或数据以及系统变量 （例如系统堆栈）。由于CPU通过AHB访问该SRAM，所以可为CPU代码执行提供零等待的访问时间。

1.3.2 看门狗定时器 (WDT)

窗式看门狗定时器的主要用途是改善系统的完整性。如果主程序因某种故障条件未能定期维护看门狗，则 WDT 会触发系统复位或其他纠错动作，例如中断。其目的是将系统从无反应状态带回到正常工作状态。

1.3.3 实时时钟 (RTC)

实时时钟 (RTC) 是跟踪当前时间的时钟。几乎在任何一个需要维持精确时间并以数字格式进行时钟显示的电子设备和计算机系统中都有 RTC。

1.3.4 系统控制单元 (SCU)

系统控制单元 (SCU) 处理所有系统任务，但与调试相关的任务除外。所有功能都密切联系在一起，所以它们可以很方便地由一个单元来处理，这个单元就是 SCU。SCU 包含电源控制单元 (PCU)、复位控制单元、时钟控制单元 (CCU) 和杂项控制单元 (GCU)。

CCU用64MHz的 DCO1振荡器产生主时钟(MCLK)和快速外设时钟(PCLK) 。PCU有一个用于产生内核电压的嵌入式稳压器 (EVR)。 PCU 还提供电压监视检测器，以便在紧急情况下 （例如掉电）保护系统性能。

1.3.5 伪随机位发生器 (PRNG)

伪随机位发生器 (PRNG) 可快速产生随机数据。

1.4 外设单元

XMC1300 提供一组支持工业应用的片上外设。

通用串行接口通道 (USIC)

USIC 是一个灵活的接口模块，它能处理多种串行通信协议，如 ASC、LIN、SSC、I2C、I2S。一个 USIC 模块包含两个独立的、可以并行使用的通信通道。集成的 FIFO 允许缓存要发送和已接收的数据，对一些实时应用情况起到缓冲作用。USIC 有多个片选信号，允许在同一通道与多个器件通信。

模 / 数转换器 (VADC)

通用模 / 数转换器模块包含一个按逐次逼近原理 (SAR) 工作的独立内核。其分辨率是从 8 位到 12 位可编程的。

ADC 内核提供一个通用的状态机，允许复杂的测量序列。多个内核可以被同步，数据转换可以完全在后台进行。可以对多个触发事件划分优先级，这样可实现对时间关键型信号的精确测量。转换结果的缓存和处理功能可避免数据丢失并保证一致性。自检机制可用于进行似真性检查。

VADC 的基本结构支持清晰的软件架构，在这样的架构中软件任务可能只需读取有效结果，而不需要关心启动转换。

模拟比较器 (ACMP) 和超量程比较器 (ORC)

模拟比较器用于比较两个模拟输入的电压，其数字输出指示哪一个输入电压较高。其中一个输入既可以是内部参考电压，也可以来自外部引脚。比较器有低功耗模式，可帮助降低总功耗。

片内集成的超量程比较器用于对 VADC 的模拟输入引脚进行过压监视。

温度传感器 (TSE)

温度传感器产生一个直接指示晶片温度的测量结果，并能在温度测量值超过所选上限/下限阈值时产生中断请求。

捕获 / 比较单元 4 (CCU4)

CCU4 外设对需要使用通用定时器进行信号监视 / 调理以及产生脉冲宽度调制 (PWM) 信号的系统来说是一个重要部件。电源的电子控制系统，类似开关模式电源或不间断电源，可以很容易地使用 CCU4 外设的功能来实现。

CCU4 内部的模块化结构使其成为一个软件友好的系统，可以快速进行代码开发和在应用之间移植。

捕获 / 比较单元 8 (CCU8)

CCU8 外设在需要复杂脉冲宽度调制 (PWM) 信号（控制互补的高边和低边开关）、多相位控制或输出奇偶校验的应用中起重要作用。 CCU8 专为最先进的电机控制、多相位和多电平电源电子系统而优化。

CCU8 内部的模块化结构使其成为一个软件友好的系统，可以快速进行代码开发和在应用之间移植。

位置接口单元 (POSIF)

POSIF 单元是一个灵活而强大的部件，它针对使用旋转编码器或霍尔传感器作为反馈回路的电机控制系统。该模块的配置方案面向非常广泛的电机控制应用需求。

该接口面向高性能运动和位置监测，为工业和汽车电机应用构建简单和复杂的控制反馈回路提供了可能。

亮度和色彩控制单元 (BCCU)

BCCU 是针对 LED 照明应用的调光控制外设，可以控制多个 LED 通道。BCCU 为每个通道提供一个单比特 sigma-delta 位流来控制亮度。通过使用专用的调光引擎，可以按照某一指数曲线来渐变调节亮度，以使在人眼看来更为自然。该模块通过调节所选通道的相对强度实现色彩控制，使用线性游走方案实现色彩的平滑改变。该模块还支持大功率多通道 LED 灯具，通过选择性地 “ 打包 （packing）” 控制位流来提供设定的输出接通时间。

通用 I/O 端口

端口为所有标准数字 I/O 提供一个通用和非常灵活的软件和硬件接口。每个端口单元都有单独的接口用于通用 I/O 操作，还为片内外设提供连接，并对焊盘特性进行控制。

1.5 调试单元

片上调试系统基于 ARM Cortex-M0™ 调试系统，提供 XMC1300 中内建的广泛的调试和仿真特性。因此，可以在目标系统环境下调试运行在 XMC1300 上的用户软件。

调试单元由外部调试工具通过调试接口控制。调试器通过一组专用寄存器控制调试单元，这组专用寄存器可通过调试接口访问。另外，调试接口还可被 CPU 控制，例如被监控程序控制。

片内硬件或软件能触发多个断点。调试单元还支持单步调试以及任意指令注入和对整个内部地址空间的读 / 写访问。断点触发信号的响应可以是 CPU 停机、一次监控程序调用或一次数据传送。

为提高性能，可以通过调试接口得到在一个观察点 （见前面的描述）传送的数据。

CPU 子系统

2 中央处理单元 (CPU)

XMC1300 基于 ARM Cortex-M0 处理器。这是一款针对广泛的嵌入式应用而设计的入门级 32 位 ARM Cortex 处理器 。该 CPU 给用户带来了极大的益处，包括：

- 易于学习和编程的简单架构
- 超低功耗，允许高能效工作
- 极佳的代码密度
- 确定性的高性能中断处理
- 与 Cortex-M 处理器家族向上兼容

ARM 参考文档

下面的文档可通过 **http://infocenter.arm.com** 访问。

[1] Cortex™-M0 Devices, Generic User Guide (ARM DUI 0467B)

[2] ARMv6-M Architecture Reference Manual (ARM DDI 0419)

[3] Cortex Microcontroller Software Interface Standard (CMSIS)

ARM 参考图形

[4] **http://www.arm.com**

2.1 概述

Cortex-M0 处理器基于面积和功耗高度优化的、具有 3 级流水线的冯诺依曼体系结构 32 位处理器核而构建。该处理器通过小而强大的指令集和全面优化的设计获得了优异的能效，同时提供高端处理硬件，包括一个单周期乘法器。

Cortex-M0 处理器采用 ARMv6-M 架构。该架构基于 16 位 Thumb® 指令集，并引入 Thumb-2 技术。Cortex-M0 指令集提供现代 32 位架构所能期望的非凡性能，同时具有比其他 8 位和 16 位微控制器更高的代码密度。

Cortex-M0 处理器紧密地集成了一个可配置的 NVIC，实现了业内领先的中断性能。该 NVIC 提供 4 个中断优先级。处理器核与 NVIC 的紧密集成使中断服务程序 (ISR) 得以快速执行，显著减小了中断延迟。这是通过寄存器硬件堆栈以及多加载和多存储操作的放弃和重启能力实现的。中断处理程序不需要任何汇编包装代码，从而消除了 ISR 的代码开销。尾链优化也在很大程度上减小了从一个 ISR 切换到另一个 ISR 的开销。

为了优化低功耗设计，NVIC 还与休眠模式紧密结合，其中深度休眠模式可使整个器件快速下电。

2.1.1 特性

CPU 提供下述功能：

- Thumb 指令集将高代码密度与 32 位性能相结合
- 适合低功耗应用的集成休眠模式

- 快速程序执行能力允许较慢的处理器时钟或增加休眠模式时间
- 单周期 32 位硬件乘法器
- 适合时间关键型应用的高性能中断处理
- 扩展的调试能力：
  – 串行线调试和单引脚调试减少了调试所需引脚数。

2.1.2 框图

Cortex-M0 的核心部件包括：

处理器核

CPU 提供大多数的 16 位 Thumb 指令和 32 位 Thumb2 指令集的子集。

嵌套向量中断控制器

NVIC 是一个嵌入式中断控制器，支持低延迟中断处理。

调试方案

XMC1300 实现了完整的硬件调试解决方案。

- 单引脚调试 (SPD) 或 2 脚串行线调试 (SWD)
- 多种硬件断点和观察点选项

调试方案提供了对处理器和存储器的高度控制和可见性，即使对小封装器件也不例外。

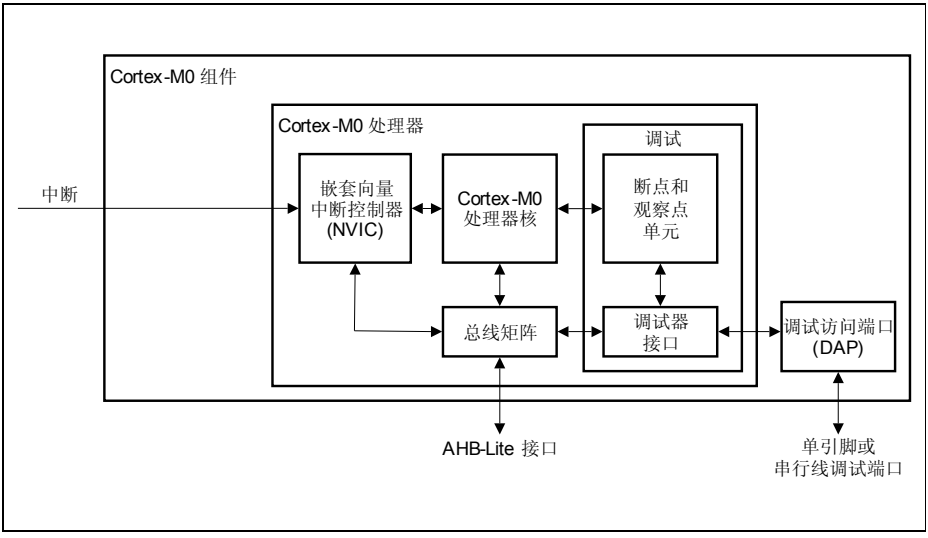


图 2-1 Cortex-M0 框图

系统级接口

Cortex-M0 处理器提供了单一的系统级接口，该接口使用 AMBA® 技术，能提供高速、低延迟的存储器访问。

2.2 程序员模型

本节描述 Cortex-M0 的程序员模型。除了对每个内核寄存器进行描述外，还包含有关处理器模式和堆栈的信息。

2.2.1 处理器模式

处理器模式有：

- **线程模式**
  用于执行应用软件。处理器在复位结束后即进入线程模式。
- **异常处理模式**
  用于处理异常。处理器在结束所有异常处理后返回到线程模式。

2.2.2 堆栈

处理器使用满递减堆栈。这意味着堆栈指针保持最后入栈的数据项在存储器中的地址。当处理器向堆栈中压入一个新数据项时，它先将堆栈指针减 1，然后将该数据项写入新的存储器单元。处理器实现了两个堆栈，即主堆栈和进程堆栈。每个堆栈都有一个指针，分别保存在不同的寄存器中，见 **堆栈指针**。

在线程模式，控制寄存器控制处理器使用主堆栈还是进程堆栈，见 **控制寄存器**。在异常处理模式，处理器总是使用主堆栈。处理器操作的选项为：

表 2-1 处理器模式、执行和堆栈使用选项一览表

处理器模式	所执行的代码	使用的堆栈
线程模式	应用软件	主堆栈或进程堆栈[1)]
异常处理模式	异常处理程序	主堆栈

1) 见 **控制寄存器**。

2.2.3 内核寄存器

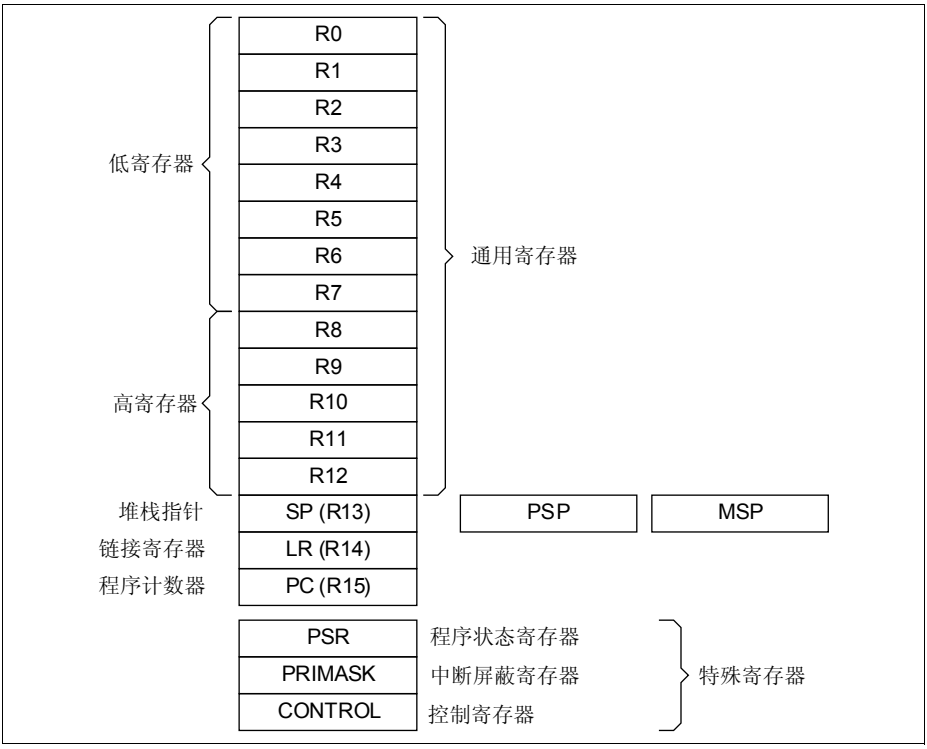


图 2-2 内核寄存器

处理器内核寄存器有：

表 2-2 内核寄存器一览表

名称	类型[1)]	复位值	描述
R0-R12	rw	未知	通用寄存器，页 2-5
MSP	rw	见描述	堆栈指针，页 2-5
PSP	rw	未知	堆栈指针，页 2-5
LR	rw	未知	链接寄存器，页 2-6
PC	rw	见描述	程序计数器，页 2-7
PSR	rw	未知	程序状态寄存器，页 2-7
APSR	rw	未知	应用程序状态寄存器，页 2-8

表 2-2 内核寄存器一览表 (续表)

名称	类型[1)]	复位值	描述
IPSR	r	00000000_H	中断程序状态寄存器，页 2-8
EPSR 优先级	r	未知	执行程序状态寄存器，页 2-10
PRIMASK	rw	00000000_H	优先级屏蔽寄存器，页 2-11
CONTROL	rw	00000000_H	控制寄存器，页 2-11

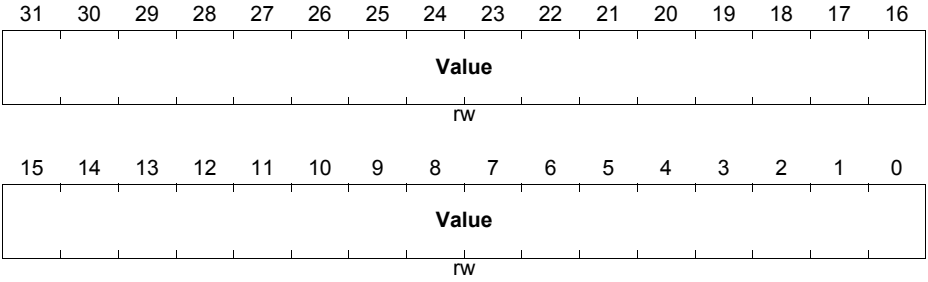
1) 描述在线程模式和异常处理模式下程序执行期间的访问类型。调试访问可能与此不同。

通用寄存器

R0-R12 是用于数据操作的 32 位通用寄存器。

注： 有关内核寄存器编程方面的信息，请参见 ARMv6-M Architecture Reference Manual [2]。

Rx (x=0-12)
通用寄存器 Rx **复位值 : $XXXXXXXX_H$**



域	位	类型	描述
Value	[31:0]	rw	**寄存器内容**

堆栈指针

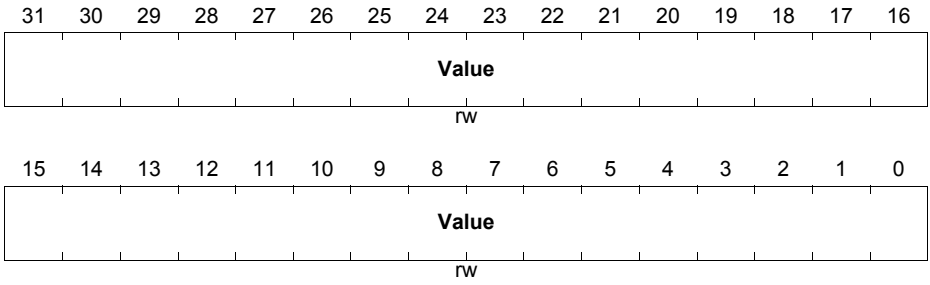
堆栈指针 (SP) 为寄存器 R13。在线程模式，控制寄存器的位 [1] 指示要使用的堆栈指针：

- 0 = 主堆栈指针 (MSP)。这是复位值。
- 1 = 进程堆栈指针 (PSP)。

在复位时，处理器将地址 00000000_H 中的值装入 MSP。

注： 有关内核寄存器编程方面的信息，请参见 ARMv6-M Architecture Reference Manual [2]。

SP
堆栈指针 **复位值：00000000_H**



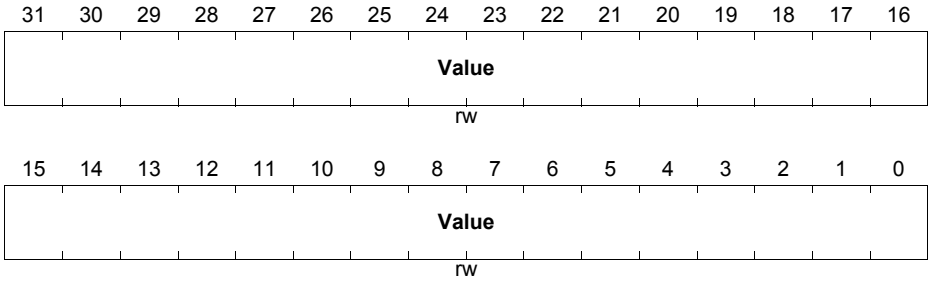
域	位	类型	描述
Value	[31:0]	rw	**寄存器内容**

链接寄存器

链接寄存器 (LR) 为寄存器 R14。它保存子程序、函数调用和异常的返回信息。复位时，LR 的值不确定。

注： 有关内核寄存器编程方面的信息，请参见 ARMv6-M Architecture Reference Manual [2]。

LR
链接寄存器 **复位值：$XXXXXXXX_H$**



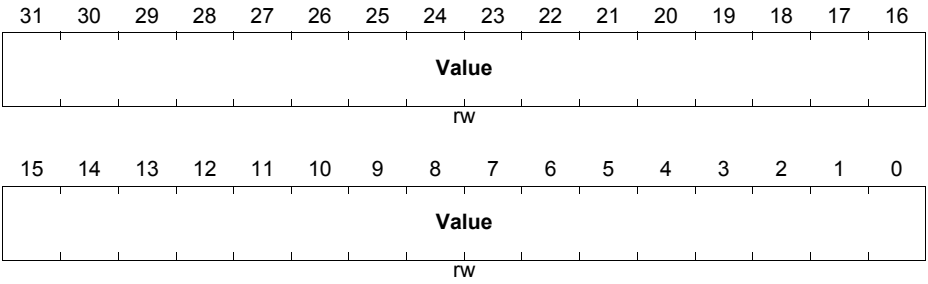
域	位	类型	描述
Value	[31:0]	rw	**寄存器内容**

程序计数器

程序计数器 (PC) 为寄存器 R15，它含有当前程序地址。复位时，处理器将位于地址 00000004_H 的复位向量值装入 PC。复位时该值的位 [0] 被装入到 EPSR 的 T 位，而且必须为 1。

注： 有关内核寄存器编程方面的信息，请参见 ARMv6-M Architecture Reference Manual [2]。

PC
程序计数器 **复位值 : 00000004_H**



域	位	类型	描述
Value	[31:0]	rw	**寄存器内容**

程序状态寄存器

程序状态寄存器 (PSR) 组合了：

- 应用程序状态寄存器 (APSR)
- 中断程序状态寄存器 (IPSR)
- 执行程序状态寄存器 (EPSR)

这些寄存器的位域在 32 位 PSR 中是互斥的。

可以单独访问这些寄存器中的某一个，也可以将任意两个或全部三个寄存器组合在一起访问。访问这些寄存器时使用寄存器名作为 MSR 或 MRS 指令的参数。例如：

- 在 MRS 指令中使用 PSR 将读所有这些寄存器。
- 在 MSR 指令中使用 APSR 将写 APSR N, Z, C 和 V 位

PSR 的组合和属性是：

表 2-3 PSR 寄存器组合

寄存器	类型	组合
PSR	rw[1)2)]	APSR, EPSR 和 IPSR
IEPSR	r	EPSR 和 IPSR
IAPSR	rw[1)]	APSR 和 IPSR
EAPSR	rw[2)]	APSR 和 EPSR

1) 处理器忽略对 IPSR 位的写操作。
2) 读 EPSR 位返回 0 值，处理器忽略对这些位的写操作。

应用程序状态寄存器

APSR 含有反映前一条指令执行情况的条件标志的当前状态 。其属性见表 2-2 的寄存器一览表。

APSR
应用程序状态寄存器 **复位值 : $XXXXXXXX_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
N	**Z**	**C**	**V**	**0**											
rw	rw	rw	rw	rw											

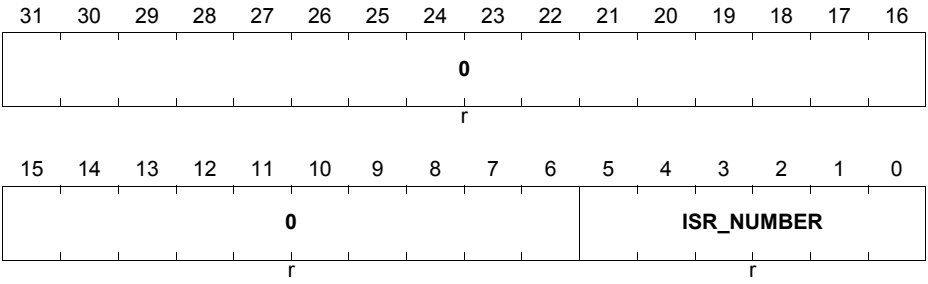
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
rw															

域	位	类型	描述
N	31	rw	负标志
Z	30	rw	零标志
C	29	rw	进位或借位标志
V	28	rw	溢出标志
0	[27:0]	r	保留

中断程序状态寄存器

IPSR 包含当前中断服务程序 (ISR) 的异常类型号。其属性见表 2-2 的寄存器一览表。

IPSR
中断程序状态寄存器 **复位值：00000000_H**



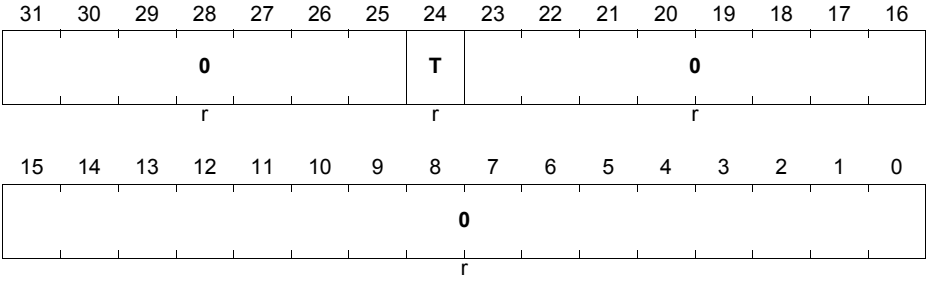
域	位	类型	描述
0	[31:6]	r	**保留**
ISR_NUMBER	[5:0]	r	**当前异常的编号** 0_D 线程模式 1_D 保留 2_D 保留 3_D 硬故障 4_D 保留 5_D 保留 6_D 保留 7_D 保留 8_D 保留 9_D 保留 10_D 保留 11_D SVCall 12_D 保留 13_D 保留 14_D PendSV 15_D SysTick 16_D IRQ0 ... 47_D IRQ31 48_D-63_D 保留 更详细的信息见 **2.5.2 节** 异常类型。

执行程序状态寄存器

EPSR 包含 Thumb 状态位。其属性见表 2-2 的寄存器一览表。

在应用软件中使用 MSR 指令直接读 EPSR 将总是返回零值。在应用软件中使用 MSR 指令写 EPSR 的操作被忽略。故障处理程序可通过检查堆叠在 PSR 中的 EPSR 值来确定故障原因。见 2.5.6 节 异常进入和返回。

EPSR
执行程序状态寄存器 **复位值：$XXXXXXXX_H$**



域	位	类型	描述
0	[31:25]	r	**保留**
T	24	r	**Thumb 状态位** 见下面对 Thumb 状态的描述。
0	[23:0]	r	**保留**

可中断 - 可重开始指令

如果在执行 LDM、STM、PUSH、POP 指令期间发生中断，则处理器放弃执行当前指令。在中断服务结束后，处理器从开始处重新执行该指令。

Thumb 状态

Cortex-M0 仅支持在 Thumb 状态执行指令。下面的指令或位域可将 T 位清 0：

- BLX、 BX 和 POP{PC} 指令
- 在异常返回时从堆叠的 xPSR 值恢复
- 异常进入时异常向量值的位 [0] 。

当 T 位为 0 时试图执行指令会导致硬故障或死锁。更详细的信息见 2.6.1 节 死锁。

异常屏蔽寄存器

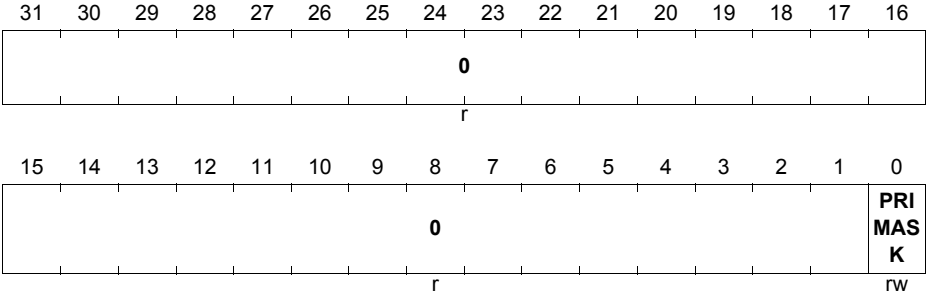
异常屏蔽寄存器禁止处理器对异常进行处理。可以禁止那些对时间关键型任务或要求原子性的代码序列可能有影响的异常。

异常的禁止或重新使能可以用 MSR 和 MRS 指令实现，也可以由 CPS 指令通过改变 PRIMASK 或 FAULTMASK 寄存器的值实现。

优先级屏蔽寄存器

PRIMASK 寄存器可避免具有可编程优先级的所有中断都被激活 。该寄存器的属性见 **表 2-2** 的寄存器一览表。

PRIMASK
优先级屏蔽寄存器 **复位值 : 00000000_H**



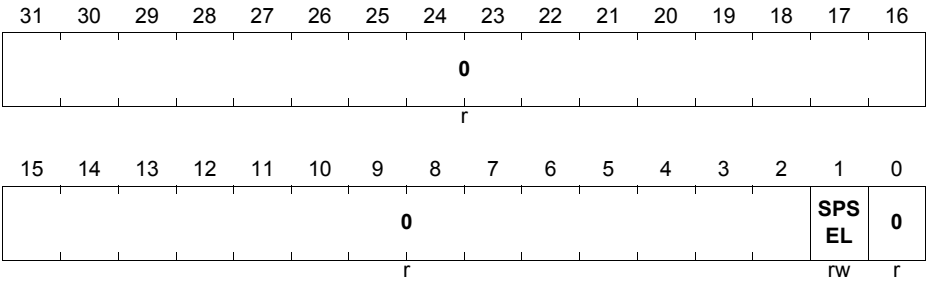
域	位	类型	描述
0	[31:1]	r	**保留**
PRIMASK	0	rw	**优先级屏蔽** 0_B 无效。 1_B 防止具有可编程优先级的所有中断都被激活 。

控制寄存器

控制寄存器控制处理器处于线程模式时使用的堆栈。该寄存器的属性见**表 2-2** 的寄存器一览表。

CONTROL
控制寄存器

复位值：00000000_H



域	位	类型	描述
0	[31:2]	r	**保留**
SPSEL	1	rw	**活动堆栈指针** 该位定义当前使用的堆栈。在异常处理模式，该位的读出值为 0，对该位的写操作被忽略。 0_B MSP 为当前堆栈指针 1_B PSP 为当前堆栈指针
0	0	r	**保留**

异常处理模式总是使用 MSP，因此处理器在该模式下忽略对控制寄存器的活动堆栈指针位的显性写操作。异常进入和返回机制自动更新控制寄存器。

在操作系统（OS）环境下，建议在线程模式运行的线程使用进程堆栈，系统内核及异常处理程序使用主堆栈。

默认情况下，线程模式使用 MSP。若要将线程模式下使用的堆栈指针切换到 PSP，需要使用 MSR 指令将活动堆栈指针位设置为 1。

注：　在改变堆栈指针时，软件必须在 MSR 指令之后立即使用一条 ISB 指令。这样方可保证 ISB 指令之后的指令使用新堆栈指针执行。

2.2.4　异常和中断

Cortex-M0 处理器支持中断和系统异常。处理器和 NVIC 处理所有异常并将其分为不同优先级。中断或异常会改变正常的软件控制流。处理器使用异常处理模式处理除复位以外的所有异常。更详细的信息见 **2.5.6.1 节** 异常进入和 **2.5.6.2 节** 异常返回。

NVIC 寄存器控制对中断的处理。更详细的信息见中断系统一章。

2.2.5 数据类型

该处理器：

- 支持以下数据类型：
  – 32 位字
  – 16 位半字
  – 8 位字节
- 按小端模式管理所有数据存储器访问。更详细的信息见 **2.3.1 节** 存储器区、类型和属性。

2.2.6 Cortex 微控制器软件接口标准

对于 Cortex-M0 微控制器系统， Cortex 微控制器软件接口标准 (CMSIS) **[3]** 定义了：

- 一种统一的方法来：
  – 访问外设寄存器
  – 定义异常向量
- 名称：
  – 内核外设寄存器
  – 内核异常向量
- RTOS 内核的器件无关接口。

CMSIS 包含对 Cortex-M0 处理器内核外设的地址定义和数据结构。

CMSIS 使开发者可重用模板代码，并可将来自不同中间件供应商的 CMSIS 兼容软件模块组合在一起，从而简化了软件开发。软件供应商可以扩充 CMSIS，以包含其外设定义和针对这些外设的访问函数。

本文档包含 CMSIS 定义的寄存器名称，并给出对访问处理器核和内核外设的 CMSIS 函数的简要描述。

注： 本文档使用 CMSIS 定义的寄存器简称。在有些情况下，它们可能与在其他文档中使用的架构简称不同。

下述各节给出有关 CMSIS 的更详细信息：

- **2.7.3 节** 电源管理编程提示
- **2.2.7 节** CMSIS 函数
- 用 CMSIS 访问 CPU 寄存器，位于中断系统一章
- NVIC 编程提示 ，位于中断系统一章

有关 CMSIS 的更多信息，请参见 **http://www.onarm.com/cmsis**。

2.2.7 CMSIS 函数

ISO/IEC C 代码不能直接访问某些 Cortex-M0 指令。本节描述由 CMSIS 提供的、能产生这些指令的内部函数和可能由 C 编译器提供的内部函数。如果所用的 C 编译器不提供合适的内部函数，可以使用在线汇编器访问相应的指令。

CMSIS 提供以下内部函数来产生 ISO/IEC C 不能直接访问的指令。

表 2-4 产生某些 Cortex-M0 指令的 CMSIS 函数

指令	CMSIS 内部函数
CPSIE i	void __enable_irq (void)
CPSID i	void __disable_irq (void)
ISB	void __ISB (void)
DSB	void __DSB (void)
DMB	void __DMB (void)
NOP	void __NOP (void)
REV	uint32_t __REV (uint32_t int value)
REV16	uint32_t __REV16 (uint32_t int value)
REVSH	uint32_t __REVSH (uint32_t int value)
WFE	void __WFE (void)
WFI	void __WFI (void)

CMSIS 还提供一些使用 MRS 和 MSR 指令来访问特殊寄存器的函数。

表 2-5 访问特殊寄存器的 CMSIS 函数

特殊寄存器	访问类型	CMSIS 函数
PRIMASK	读	uint32_t __get_PRIMASK (void)
	写	void __set_PRIMASK (uint32_t value)
CONTROL	读	uint32_t __get_CONTROL (void)
	写	void __set_CONTROL (uint32_t value)
MSP	读	uint32_t __get_MSP (void)
	写	void __set_MSP (uint32_t TopOfMainStack)
PSP	读	uint32_t __get_PSP (void)
	写	void __set_PSP (uint32_t TopOfMainStack)

2.3 存储器模型

本节描述处理器的存储器映射和存储器访问行为。处理器有固定的默认存储器映射，可提供高达 4GB 的可寻址存储器。存储器映射情况如图 2-3 所示。

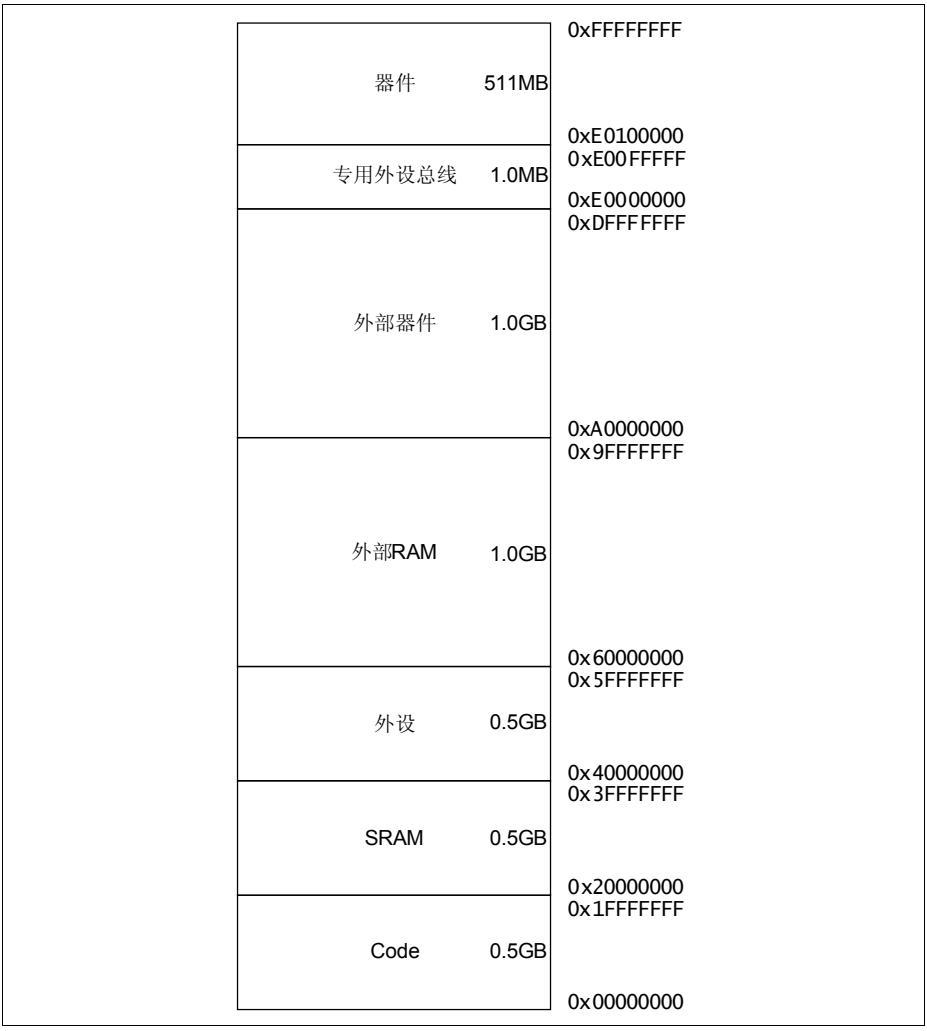


图 2-3　　存储器映射

处理器为内核外设寄存器保留专用外设总线 (PPB) 地址范围的区域，详见 **2.8.1 节** 关于专用外设。

2.3.1 存储器区、类型和属性

存储器映射被划分成多个区。每个区有限定的存储器类型，某些区有附加的存储器属性。存储器类型和属性决定了访问这个区的行为。

存储器类型有：

正常 处理器可对事务重新排序以提高效率，或执行推测读。

器件 处理器保持相对于其他针对器件或强秩序存储器的事务的事务顺序。

强秩序 处理器保持相对于所有其他事务的事务顺序。

对器件和强秩序存储器的不同访问顺序要求意味着存储器系统可以缓存一个对器件存储器的写操作，但绝对不会缓存一个对强秩序存储器的写操作。

附加的存储器属性包括：

永不执行 (XN) 表示处理器阻止指令访问。在执行一条从存储器的 XN 区取出的指令时产生硬故障 （HardFault）异常。

2.3.2 存储器访问次序

对于大多数由显性存储器访问指令引起的存储器访问而言，存储器系统不能保证这些访问的完成顺序与程序的指令顺序一致，因为任何重新排序都不影响指令序列的行为。一般来说，如果正确的程序执行取决于两次按程序顺序完成的存储器访问，则软件必须在这两个存储器访问指令之间插入一条存储器屏障指令，见 **2.3.4 节** 存储器访问的软件次序。

然而，存储器系统不保证对器件和强秩序存储器的某些访问的顺序。对于两个存储器访问指令 A1 和 A2，如果在程序顺序上 A1 先于 A2，则由这两条指令导致的存储器访问的顺序如 **图 2-4** 所示。

A1 \ A2	正常访问	器件访问	强秩序访问
正常访问	-	-	-
器件访问	-	<	<
强秩序访问	-	<	<

图 2-4 存储器系统访问次序

图中：

- “-” 表示存储器系统不保证访问顺序。
- “<” 表示这两次访问遵循程序顺序，即 A1 总是先于 A2 执行。

2.3.3 存储器访问行为

对存储器映射中的每个区的访问行为如下：

表 2-6 存储器访问行为

地址范围	存储器区	存储器类型[1)]	XN[1)]	描述
0x00000000-0x1FFFFFFF	代码	正常	-	程序代码的可执行区。可以在该区存放数据。
0x20000000-0x3FFFFFFF	SRAM	正常	-	数据的可执行区。可以在该区存放代码。
0x40000000-0x5FFFFFFF	外设	器件	XN	外设区。
0x60000000-0x9FFFFFFF	外部 RAM	正常	-	数据的可执行区。
0xA0000000-0xDFFFFFFF	外部器件	器件	XN	外部器件存储器。
0xE0000000-0xE00FFFFF	专用外设总线	强秩序	XN	该区包括 NVIC、系统定时器和系统控制块。在该区只能使用字访问。
0xE0100000-0xFFFFFFFF	器件	器件	XN	厂商专用。

1) 更详细的信息见 2.3.1 节 存储器区、类型和属性 。

代码区、 SRAM 区和外部 RAM 区都可存储程序。

2.3.4 存储器访问的软件顺序

程序流中指令的顺序不能总是保证对应的存储器事务的顺序。这是因为：

- 为了提高效率，处理器可以改变某些存储器访问的顺序，前提是这样做不影响指令序列的行为。
- 存储器映射中的存储器或器件有不同的等待状态。
- 有些存储器访问是被缓存的或是推测性的。

2.3.2 节 存储器访问次序一节描述了存储器系统保证存储器访问顺序的情况。如果存储器访问的顺序很关键，软件必须包含存储器屏障指令，以强制存储器访问按顺序进行。处理器提供下面的存储器屏障指令：

DMB 数据存储器屏障 (DMB) 指令保证未完成的存储器事务在其后的存储器事务之前结束。

DSB 数据同步屏障 (DSB) 指令保证未完成的存储器事务在其后的指令执行之前结束。

ISB 指令同步屏障 (ISB) 保证所有已完成的存储器事务的结果可被随后的指令承认。

2.3.5 存储器的端格式

处理器将存储器视为从零开始向上编号的字节的线性集合。例如，字节 0-3 保存第一个存储字，字节 4-7 保存第二个存储字。**2.3.5.1 节** 描述数据字是如何在存储器中存储的。

2.3.5.1 小端格式

在小端格式，处理器将一个字的最低有效字节 (lsbyte) 存储在编号最小的字节，而最高有效字节 (msbyte) 存储在编号最大的字节。**图 2-5** 给出了小端格式的一个例子。

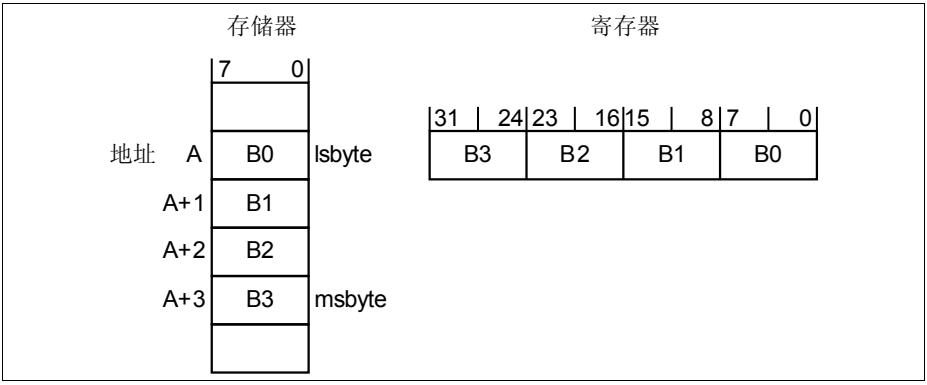


图 2-5 小端格式 (示例)

2.4 指令集

表 2-7 列出了所支持的 Cortex-M0 指令。有关指令和操作数的详细信息，请参见 Cortex™-M0 Devices, Generic User Guide **[1]**。

表 2-7 Cortex-M0

助记符	操作数	简要描述	标志
ADCS	{Rd,} Rn, Rm	带进位加法	N,Z,C,V
ADD{S}	{Rd,} Rn, <Rm|#imm>	加法	N,Z,C,V
ADR	Rd, label	PC 相对地址到寄存器	-
ANDS	{Rd,} Rn, Rm	按位与	N,Z
ASRS	{Rd,} Rm, <Rs|#imm>	算数右移	N,Z,C
B{cc}	label	转移 { 有条件 }	-
BICS	{Rd,} Rn, Rm	位清零	N,Z
BKPT	#imm	断点	-
BL	label	带链接转移	-
BLX	Rm	带链接间接转移	-
BX	Rm	间接转移	-
CMN	Rn, Rm	反值比较	N,Z,C,V
CMP	Rn, <Rm|#imm>	比较	N,Z,C,V
CPSID	i	改变处理器状态，禁止中断	-
CPSIE	i	改变处理器状态，允许中断	-
DMB	-	数据存储屏障	-
DSB	-	数据同步屏障	-
EORS	{Rd,} Rn, Rm	异或	N,Z
ISB	-	指令同步屏障	-
LDM	Rn{!}, reglist	加载多个寄存器，后增 1	-
LDR	Rt, label	从 PC 相对地址加载寄存器	-
LDR	Rt, [Rn, <Rm|#imm>]	加载字到寄存器	-
LDRB	Rt, [Rn, <Rm|#imm>]	加载字节到寄存器	-
LDRH	Rt, [Rn, <Rm|#imm>]	加载半字到寄存器	-
LDRSB	Rt, [Rn, <Rm|#imm>]	加载有符号字节到寄存器	-

表 2-7 Cortex-M0 (续表)

助记符	操作数	简要描述	标志
LDRSH	Rt, [Rn, <Rm|#imm>]	加载有符号半字到寄存器	-
LSLS	{Rd,} Rn, <Rs|#imm>	逻辑左移	N,Z,C
LSRS	{Rd,} Rn, <Rs|#imm>	逻辑右移	N,Z,C
MOV{S}	Rd, Rm	传送	N,Z
MRS	Rd, spec_reg	从特殊寄存器传送到通用寄存器	-
MSR	spec_reg, Rm	从通用寄存器传送到特殊寄存器	N,Z,C,V
MULS	Rd, Rn, Rm	乘法， 32 位结果	N,Z
MVNS	Rd, Rm	按位取反	N,Z
NOP	-	空操作	-
ORRS	{Rd,} Rn, Rm	逻辑或	N,Z
POP	reglist	寄存器出栈	-
PUSH	reglist	寄存器入栈	-
REV	Rd, Rm	反转字的字节顺序	-
REV16	Rd, Rm	反转每个半字的字节顺序	-
REVSH	Rd, Rm	反转有符号半字的字节顺序	-
RORS	{Rd,} Rn, Rs	循环右移	N,Z,C
RSBS	{Rd,} Rn, #0	反向减法	N,Z,C,V
SBCS	{Rd,} Rn, Rm	带借位减法	N,Z,C,V
STM	Rn!, reglist	存储多个寄存器，后增 1	-
STR	Rt, [Rn, <Rm|#imm>]	按字存储寄存器	-
STRB	Rt, [Rn, <Rm|#imm>]	按字节存储寄存器	-
STRH	Rt, [Rn, <Rm|#imm>]	按半字存储寄存器	-
SUB{S}	{Rd,} Rn, <Rm|#imm>	减法	N,Z,C,V
SVC	#imm	管理程序调用	-

表 2-7 Cortex-M0 (续表)

助记符	操作数	简要描述	标志
SXTB	Rd, Rm	带符号字节扩展	-
SXTH	Rd, Rm	带符号半字扩展	-
TST	Rn, Rm	基于逻辑与的测试	N,Z
UXTB	Rd, Rm	无符号字节扩展	-
UXTH	Rd, Rm	无符号半字扩展	-
WFE	-	等待事件	-
WFI	-	等待中断	-

2.4.1 内部函数

ISO/IEC C代码不能直接访问某些Cortex-M0指令。能产生这些指令的内部函数由CMSIS提供，也可能由 C 编译器提供。对这些函数的描述见 **2.2.7 节**。

2.5 异常模型

本节描述异常模型。包括：

- 异常状态 (**2.5.1 节**)
- 异常类型 (**2.5.2 节**)
- 异常处理程序 (**2.5.3 节**)
- 向量表 (**2.5.4 节**)
- 异常优先级 (**2.5.5 节**)
- 异常进入和返回 (**2.5.6 节**)

2.5.1 异常状态

每个异常都有以下状态：

不活动 异常不活动，也不处于挂起状态。

挂起 异常正在等待处理器服务。
来自外设或软件的中断请求可将相应中断的状态改变为挂起状态。

活动 异常得到处理器服务，但服务尚未结束。

注： 一个异常处理程序可中断另一异常处理程序的执行。在这种情况下，两个异常都处于活动状态。

活动并挂起 异常正在被处理器服务，又产生一个来自相同异常源的挂起异常。

2.5.2 异常类型

异常类型见表 2-8 中的描述。

表 2-8 异常类型

异常类型	描述
复位	复位由上电或热复位操作引起。异常模型将复位作为一种特殊形式的异常。当复位有效时，处理器可在一条指令的任何位置停止运行。当复位解除时，处理器从向量表中的复位入口地址开始执行程序。处理器以线程模式重新执行程序。
硬故障（HardFault）	硬故障是一种在正常和异常处理期间因发生错误而产生的异常。硬故障有固定的优先级 -1，意味着其优先级高于任何具有可编程优先级的异常。
SVCall	管理程序调用 (SVC) 是由 SVC 指令触发的一种异常。在 OS 环境下，应用程序可以使用 SVC 指令访问 OS 内核函数和驱动程序。
PendSV	PendSV 是一种中断驱动的系统级服务请求。在 OS 环境下，当没有其他异常处于活动状态时，使用 PendSV 进行上下文切换。
SysTick	SysTick 异常是系统定时器计数到零时产生的异常。软件也可以产生 SysTick 异常。在 OS 环境下，处理器可以使用该异常作为系统节拍。
中断 (IRQ)	中断，或称 IRQ，是由外设或软件请求产生的异常。所有中断都不与指令执行同步。在系统中，外设使用中断与处理器通信。

表 2-9 不同异常类型的性质

异常号[1]	IRQ 号[1]	异常类型	优先级	向量地址或偏移量[2]	激活方式
1	-	复位	-3，最高	0x00000004	异步
2	-	保留	-	-	-
3	-13	硬故障	-1	0x0000000C	同步
4-10	-	保留	-	-	-
11	-5	SVCall	可编程[3]	0x0000002C	同步
12-13	-	保留	-	-	-
14	-2	PendSV	可编程[3]	0x00000038	异步
15	-1	SysTick	可编程[3]	0x0000003C	异步
16 及以上	0 及以上	中断 (IRQ)	可编程[3]	0x00000040 及以上[4]	异步

1) 为了简化软件层，CMSIS 仅使用 IRQ 号，因此除中断以外的所有异常都使用负值。IPSR 返回异常号，见 中断程序状态寄存器。
2) 更详细的信息见 2.5.4 节 向量表。
3) 见中断系统一章中的中断优先级寄存器。
4) 以 4 为步长增加。

对于异步异常 （复位除外），处理器在异常被触发到处理器进入异常处理程序之前这一段时间还可以执行一些指令。

软件可以禁止表 2-9 中具有可编程优先级的异常，见中断系统一章中的中断清除 - 使能寄存器。

有关硬故障的更详细信息，见 2.6 节 故障处理。

2.5.3 异常处理程序

处理器在处理异常时使用：

中断服务程序 (ISR)	中断 IRQ0 ~ IRQ31 是由 ISR 处理的异常。
故障处理程序	硬故障是由故障处理程序处理的唯一异常。
系统处理程序	PendSV、 SVCall、 SysTick 和硬故障都是由系统处理程序处理的系统异常。

2.5.4 向量表

向量表包含堆栈指针的复位值和所有异常处理程序的起始地址，也称为异常向量。图 2-6 示出了向量表中异常向量的次序。每个向量的最低有效位必须为 1，表示异常处理程序使用 Thumb 代码编写。

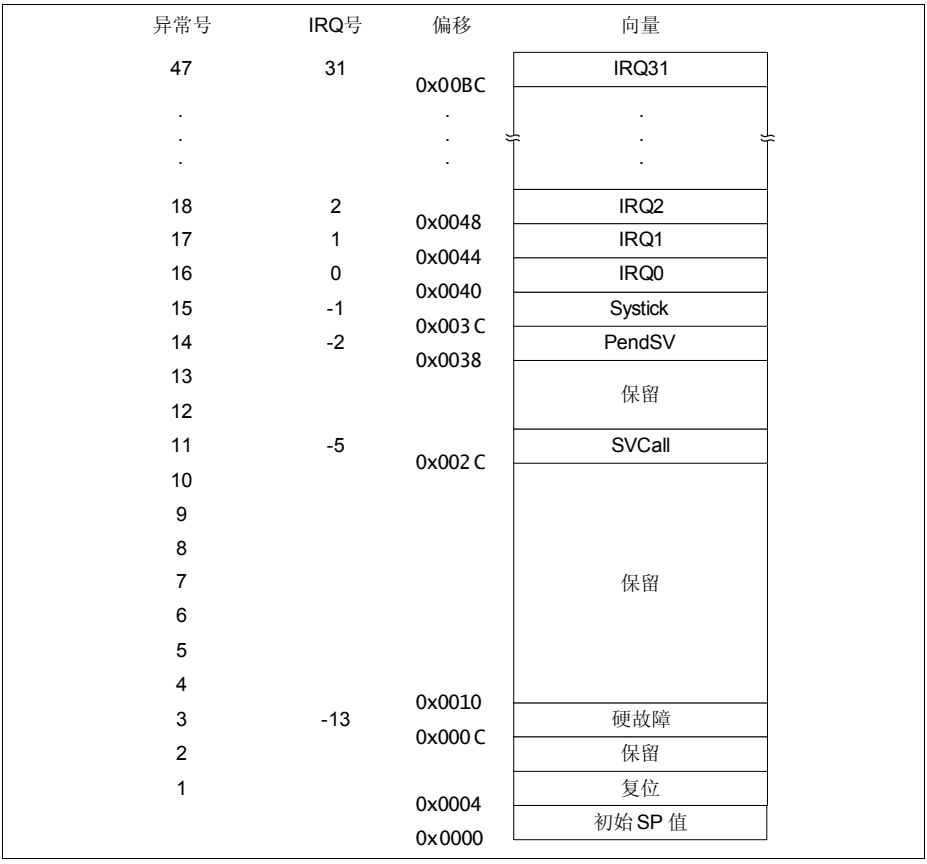


图 2-6　　向量表

向量表固定从地址 0x00000000 开始。

2.5.4.1　向量表重映射

在 XMC1300 中，向量表位于 ROM 内。因此，要根据**表 2-10** 所示的映射关系将向量表重新映射到 SRAM 中。用户应用程序使用这些存储单元作为实际的异常和中断处理程序的入口点。这是通过将这些处理程序的代码存放在该位置，或在该位置放置一条转移到处理程序的指令来实现的。

例如，在发生因 IRQ0 引起的异常进入时，处理器从向量表读取处理程序的中间起始地址 $2000'0040_H$ (在 ROM 中固定) 并从该地址开始执行。如果因存储器大小的限制，实际的处理程序位于另外一个地址，则地址 $2000'0040_H$ 应触发一条加载和一条转移指令，以跳转到这个新地址单元。

注： 如果所有向量都要用到，用户应用需保留 SRAM 地址 2000’000C$_H$ - 2000’00BF$_H$，为重新映射向量表所用。

表 2-10 重映射的向量表

异常号	IRQ 号	向量	默认向量地址	重映射的向量地址
-	-	初始 SP 值	0000’0000$_H$	1000’1000$_H$
1	-	复位	0000’0004$_H$	1000’1004$_H$ 1)
3	-13	硬故障	0000’000C$_H$	2000’000C$_H$
11	-5	SVCall	0000’002C$_H$	2000’002C$_H$
14	-2	PendSV	0000’0038$_H$	2000’0038$_H$
15	-1	SysTick	0000’003C$_H$	2000’003C$_H$
16-47	0-31	IRQn (n=0-31)	0000’0040$_H$ + (n*4)	2000’0040$_H$ + (n*4)

1) 重映射的复位向量地址是指用户模式下启动软件在退出启动过程之际要跳往的地址 (Flash 存储器起始地址)。

2.5.5 异常的优先级

表 2-9 列出了所有异常及其相关联的优先级，其中：

- 优先级值越低表示优先级越高
- 除复位和硬故障以外的所有异常都具有可编程优先级。

如果软件不配置任何优先级，则具有可编程优先级的所有异常的优先级都是 0。有关配置异常优先级的信息见：

- 系统处理程序优先级寄存器 **SHPR2**、 **SHPR3**。
- 中断系统一章中的中断优先级寄存器。

注： 可编程优先级值的范围是 0-192，步长为 64。这意味着具有固定负优先级值的复位和硬故障异常总是具有比任何其他异常更高的优先级。

例如，给 IRQ[0] 分配一个较高的优先级值，给 IRQ[1] 分配一个较低的优先级值，这表示 IRQ[1] 比 IRQ[0] 有更高的优先级。如果 IRQ[1] 和 IRQ[0] 都有效，则 IRQ[1] 在 IRQ[0] 之前被处理。

如果多个挂起的异常具有相同的优先级，则具有最小异常号的挂起异常优先被处理。例如，如果 IRQ[0] 和 IRQ[1] 都处于挂起状态，且二者具有相同的优先级，则 IRQ[0] 在 IRQ[1] 之前被处理。

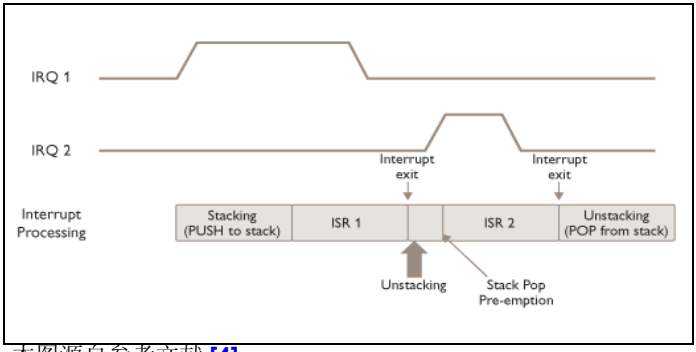
当处理器正在执行一个异常处理程序时，如果发生了一个具有更高优先级的异常，则正在执行的异常处理程序被抢占。如果在该异常被处理期间，发生了一个具有相同优先级的异常，无论其异常号为何，该异常处理程序都不会被抢占。但是新中断的状态变为挂起。

2.5.6 异常进入和返回

异常处理可以用下面的术语描述：

抢占

当处理器正在执行一个异常处理程序时，一个具有比正被处理的异常更高优先级的异常可以抢占该异常处理程序。
当一个异常抢占另一个异常时，这些异常被称为嵌套异常。更详细的信息见 **2.5.6.1 节** 异常进入。



本图源自参考文献 **[4]**。

返回

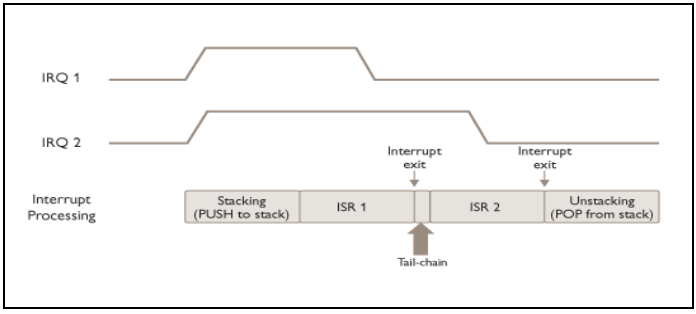
异常返回发生在异常处理程序结束之后，并且：

- 没有具有足够高优先级的挂起异常需要处理。
- 已结束的异常处理程序当时不在处理晚到异常。

处理器进行出栈操作并将处理器状态恢复到中断发生之前的状态。更详细的信息见 **2.5.6.2 节** 异常返回。

尾链 （Tail-chaining）

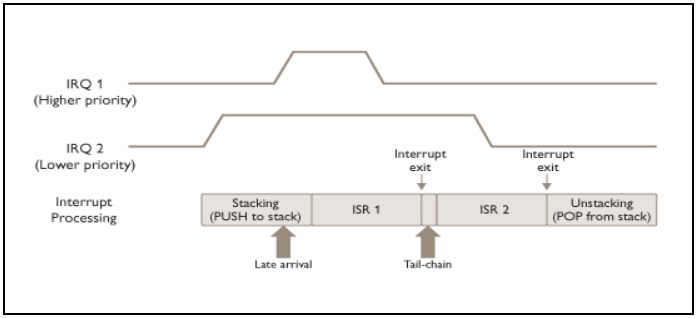
该机制能加速异常服务。在一个异常处理程序结束时，如果有一个挂起异常符合异常进入的要求，则出栈操作被跳过，系统控制权转给新异常处理程序。



本图源自参考文献 [4]。

晚到 （Late-arriving）

该机制加速抢占过程。如果在前一个异常的状态保存期间发生了一个具有更高优先级的异常，处理器会转去处理优先级较高的异常，并开始获取该异常的向量。状态保存过程不受后来的这个异常的影响，因为要保存的状态对这两个异常来说是相同的。在从该晚到异常的异常处理程序返回时，正常的尾链规则同样适用。



本图源自参考文献 [4]。

2.5.6.1 异常进入

存在一个具有足够高优先级的挂起异常，并且满足下述条件之一，则会发生异常进入。

- 处理器处于线程模式
- 新异常比正被处理的异常优先级高。在这种情况下，新异常会抢占原来的异常。

当一个异常抢占另一个异常时，这两个异常是嵌套的。

足够高的优先级意指该异常具有比由异常屏蔽寄存器设置的任何极限值更高的优先级，见 异常屏蔽寄存器。优先级比该异常低的异常进入挂起状态，但不会被处理器处理。

当处理器开始处理一个异常时，处理器将信息压入到当前堆栈，除非该异常是一个尾链或晚到异常。该操作被称为入栈， 8 个数据字的结构被称为栈帧。栈帧包含的信息如图 2-7 所示。

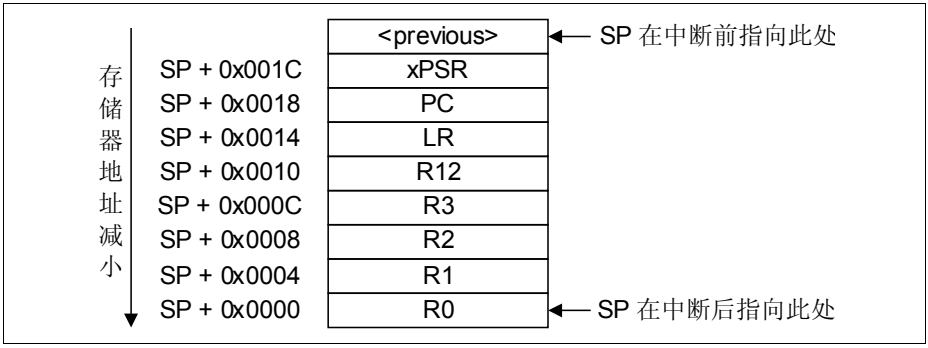


图 2-7　　异常栈帧

在入栈操作之后，堆栈指针指示位于栈帧的最低地址。栈帧被对齐到双字地址。

栈帧包含返回地址，这是被中断的程序的下一条指令的地址。该值在异常返回时被恢复到 PC 中，以使被中断的程序从该地址恢复执行。

处理器执行一次向量获取操作，即从向量表读取异常处理程序的起始地址。入栈操作结束后，处理器开始执行异常处理程序。与此同时，处理器写一个 EXC_RETURN 值到 LR。该值指示哪一个堆栈指针对应该栈帧，以及在异常进入发生前处理器处于什么工作模式。

如果在异常进入期间没有发生具有更高优先级的异常，则处理器开始执行异常处理程序，并且自动将相应的挂起中断的状态改变为活动状态。

如果在异常进入期间发生了另一个具有更高优先级的异常，则处理器开始执行该异常的异常处理程序，并且不改变早先那个异常的挂起状态。这种情况就是晚到中断。

2.5.6.2　异常返回

当处理器处于异常处理模式并且执行下面一条试图将 PC 设置为一个 EXC_RETURN 值的指令时，发生异常返回：

- 加载 PC 的 POP 指令；
- 使用任一寄存器的 BX 指令。

在异常进入时，处理器将一个 EXC_RETURN 值保存到 LR。异常机制依赖该值检测处理器何时执行完异常处理程序。EXC_RETURN 值的位 [31:4] 被置 1。当该值被加载到 PC 时，处理器检测到异常处理完成，便开始异常返回过程。EXC_RETURN 值的位 [3:0] 指

示所需要的返回堆栈和处理器模式。表 2-11 列出了 EXC_RETURN 值，以及对异常返回行为的描述。

表 2-11 异常返回行为

EXC_RETURN[31:0]	描述
0xFFFFFFF1	返回到异常处理模式。 返回时从主堆栈获取状态。 返回后使用 MSP 执行。
0xFFFFFFF9	返回到线程模式。 返回时从 MSP 获取状态。 返回后使用 MSP 执行。
0xFFFFFFFD	返回到线程模式。 返回时从 PSP 获取状态。 返回后使用 PSP 执行。
所有其他值	保留。

2.6 故障处理

故障是异常的一个子集，见 2.5 节 异常模型。所有故障都导致硬故障（HardFault）异常或引起死锁 （如果故障发生在硬故障处理期间）。下面是全部故障：

- 在等于或高于 SVCall 的优先级执行一条 SVC 指令
- 在没有连接调试器的情况下执行一条 BKPT 指令
- 在执行加载或存储时系统发生总线错误
- 从一个 XN 存储器地址执行一条指令
- 从一个存储单元执行一条指令，系统因此产生总线故障
- 取异常向量时系统发生总线错误
- 执行一条未定义的指令
- 由于 T 位之前被清 0 而在非 Thumb 状态执行了一条指令
- 试图加载或存储到一个未对齐的地址

注： 只有复位能抢占具有固定优先级的硬故障处理程序。一个硬故障能抢占除复位和另一硬故障以外的任何异常。

2.6.1 死锁

如果在执行硬故障处理程序时又发生了一个故障，则处理器进入死锁状态。如果在一个异常返回使用 MSP 执行 PSR 出栈操作时系统产生了总线错误，处理器也同样进入死锁状态。当处理器处于死锁状态时，它不执行任何指令。处理器会一直保持在死锁状态，直到发生下面的一种情况：

- 处理器被复位
- 处理器被调试器停止

2.7 电源管理

Cortex-M0 处理器的休眠模式降低功耗。有两种休眠模式：

- 休眠模式
- 深度休眠模式

SCR 的 SLEEPDEEP 位选择使用哪一种休眠模式，见系统控制寄存器 **SCR**。

本节描述进入休眠模式的机制以及从休眠模式唤醒的条件。

2.7.1 进入休眠模式

本节描述软件将处理器置于休眠模式所能使用的机制。

系统可能产生假性唤醒事件。例如，一次调试操作即可唤醒处理器。因此，软件必须能在发生这种事件后使处理器返回到休眠模式。程序可以使用一个空闲循环将处理器重新置回休眠模式。

等待中断

等待中断指令 WFI 会使处理器立即进入休眠模式。当处理器执行一条 WFI 指令时，它会立即停止执行指令并进入休眠模式。

等待事件

等待事件指令 WFE 会使处理器进入休眠模式，但取决于一个单比特事件寄存器的值。当处理器执行一条 WFE 指令时，它要检查该事件寄存器的值：

0 处理器停止执行指令并进入休眠模式。
1 处理器将该事件寄存器清 0，继续执行指令而不进入休眠模式。

如果该事件寄存器的值为 1，表示处理器在执行 WFE 指令后绝不能进入休眠模式。一般来说，这是因为有一个外部事件生效。软件不能直接访问该寄存器。

退出时休眠

如果 SCR 的 SLEEPONEXIT 位被置 1，当处理器结束执行异常处理程序并返回线程模式时，会立即进入休眠模式。该机制仅用于处理器在有中断发生时才运行的应用中。

2.7.2 从休眠模式唤醒

处理器从休眠模式被唤醒的条件取决于使其进入休眠模式的机制。

从 WFI 或退出时休眠唤醒

下面的事件是 WFI 唤醒事件：

- 复位事件
- 调试事件，如果调试被使能
- 具有能抢占任何当前活动异常的优先级的异常，如果 PRIMASK 被设置为 0。

注： 如果 PRIMASK 被置 1，具有比当前异常更高优先级的中断或异常会将处理器唤醒。中断处理程序要等到处理器将 PRIMASK 清 0 后才会被执行。有关 PRIMASK 的更详细信息见异常屏蔽寄存器。

从 WFE 唤醒

以下事件是 WFE 唤醒事件：

- 复位事件
- 具有能导致异常进入的足够高优先级的异常或中断
- 进入挂起状态的异常或中断，如果 SEVONPEND 被置 1 (见 **SCR**)
- 调试事件，如果调试被使能

注： XMC1300 不支持外部事件。然而， SEV 指令会置位事件寄存器 。

2.7.3 电源管理编程提示

ISO/IEC C代码不能直接生成WFI和WFE指令。CMSIS提供的下述函数能生成这些指令：

```
void __WFE(void)  // 等待事件
void __WFI(void)  // 等待中断
```

2.8 专用外设

下面各节是关于 ARM Cortex-M0 内核外设的参考文档。

2.8.1 关于专用外设

专用外设总线 (PPB) 的地址映射为：

表 2-12 内核外设寄存器区

地址	内核外设	描述
0xE000E008-0xE000E00F	系统控制块	见 **2.8.2 节** 和 **2.9.1 节**
0xE000E010-0xE000E01F	系统定时器	见 **2.8.3 节**和 **2.9.2 节**
0xE000E100-0xE000E4EF	嵌套向量中断控制器	见中断系统一章
0xE000ED00-0xE000ED3F	系统控制块	见 **2.8.2 节** 和 **2.9.1 节**
0xE000EF00-0xE000EF03	嵌套向量中断控制器	见中断系统一章

2.8.2 系统控制块

系统控制块 (SCB) 提供系统实现方面的信息和进行系统控制，包括配置、控制和系统异常报告。

2.8.2.1 系统控制块使用提示

要确保软件使用对齐的 32 位字访问系统控制块的所有寄存器。

2.8.3 系统定时器 SysTick

处理器有一个 24 位的系统定时器 SysTick。该定时器自重载值开始向下计数到零，再重新加载，即在下一时钟周期回绕到 SYST_RVR 寄存器中的重载值，然后在随后的时钟周期继续向下计数。

注： 当处理器因调试而停止执行时，计数器不进行减 1 计数。

2.8.3.1 SysTick 使用提示

中断控制器时钟更新 SysTick 的计数值。当选择处理器时钟并因为低功耗模式而停止该时钟信号时，SysTick 计数器停止计数。在选用外部时钟时，该时钟在低功耗模式继续运行， SysTick 可用作唤醒源。

要确保软件使用字对齐的操作访问 SysTick 寄存器。

如果 SysTick 计数器重载值和当前值在复位时不确定，则对 SysTick 计数器正确的初始化过程为：

1. 对重载值编程。
2. 清除当前值。
3. 对控制和状态寄存器编程。

2.9 PPB 寄存器

CPU 专用外设寄存器的基地址为 $E000E000_H$。

表 2-13 寄存器一览表

简称	描述	偏移地址	访问方式		描述见
			读	写	
系统控制空间 (SCS)					
CPUID	CPUID 基本寄存器	$D00_H$	PV, 32	PV, 32	页 2-33
ICSR	中断控制和状态寄存器	$D04_H$	PV, 32	PV, 32	页 2-35

表 2-13 寄存器一览表 (续表)

简称	描述	偏移地址	访问方式		描述见
			读	写	
AIRCR	应用中断和复位控制寄存器	$D0C_H$	PV, 32	PV, 32	页 2-37
SCR	系统控制寄存器	$D10_H$	PV, 32	PV, 32	页 2-38
CCR	配置和控制寄存器	$D14_H$	PV, 32	PV, 32	页 2-40
SHPR2	系统处理程序优先级寄存器 2	$D1C_H$	PV, 32	PV, 32	页 2-41
SHPR3	系统处理程序优先级寄存器 3	$D20_H$	PV, 32	PV, 32	页 2-42
SHCSR	系统处理程序控制和状态寄存器	$D24_H$	PV, 32	PV, 32	页 2-43
系统定时器 (SysTick)					
SYST_CSR	SysTick 控制和状态寄存器	010_H	PV, 32	PV, 32	页 2-43
SYST_RVR	SysTick 重载值寄存器	014_H	PV, 32	PV, 32	页 2-46
SYST_CVR	SysTick 当前值寄存器	018_H	PV, 32	PV, 32	页 2-47
SYST_CALIB	SysTick 校准值寄存器	$01C_H$	PV, 32	-	页 2-48

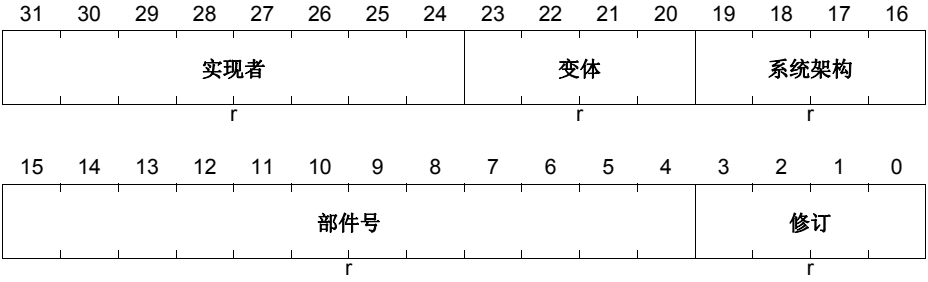
2.9.1 SCS 寄存器

CPUID

CPUID 寄存器包含处理器的部件号、版本和实现信息。

CPUID

CPUID 基本寄存器 **(E000ED00$_H$)** **复位值 : 410CC200$_H$**



域	位	类型	描述
修订	[3:0]	r	修订号 0_H Patch 0
部件号	[15:4]	r	处理器的部件号 $C20_H$ Cortex-M0
系统架构	[19:16]	r	系统架构 C_H ARMv6-M
变体	[23:20]	r	变体号 0_H Revision 0
实现者	[31:24]	r	实现者代码 41_H ARM

ICSR

ICSR：

- 提供：
  – PendSV 和 SysTick 异常的挂起置 1 和挂起清 0 位。
- 指示：
  – 正被处理的异常的异常号
  – 是否有被抢占的活动异常
  – 优先级最高的挂起异常的异常号
  – 是否有中断处于挂起状态。

ICSR
中断控制和状态寄存器 **(E000ED04$_H$)** **复位值：00000000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			PENDSVSET	PENDSVCLR	PENDSTSET	PENDSTCLR	0		ISRPENDING	0				VECTPENDING	
r			rw	w	rw	w	r		r	r				r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VECTPENDING				0						VECTACTIVE					
r				r						r					

域	位	类型	描述
VECTACTIVE[1)]	[5:0]	r	**活动异常的编号** 00_H 线程模式 非零值：当前活动异常的异常号。 *注：从该值减去 16 即可得到 CMSIS IRQ 号，该 IRQ 号用作中断清除 - 使能、置位 - 使能、清除 - 挂起、置位 - 挂起或优先级寄存器的索引，见 **中断程序状态寄存器**。*
0	[11:6]	r	**保留** 读出值为 0；应写入 0。
VECTPENDING	[17:12]	r	**挂起异常号** 指示优先级最高的被使能的挂起异常的异常号。 0_H 无挂起异常 非零值：优先级最高的被使能挂起异常的异常号。
0	[21:18]	r	**保留** 读出值为 0；应写入 0。

域	位	类型	描述
ISRPENDING	22	r	**中断挂起标志** 该位置 1 中断挂起标志，故障除外。 0_B 中断不处于挂起状态。 1_B 中断处于挂起状态。
0	[24:23]	r	**保留** 读出值为 0；应写入 0。
PENDSTCLR	25	w	**SysTick 异常挂起清除** 0_B 无效 1_B 清除 SysTick 异常的挂起状态。 该位是只写位，其读出值不确定。
PENDSTSET	26	rw	**SysTick 异常挂起置位** 0_D SysTick 异常未挂起。 1_D SysTick 异常挂起。 向该位写 0 无效。
PENDSVCLR	27	w	**PendSV 挂起清除** 该位清除一个挂起的 PendSV 异常。 0_B 不清除。 1_B 清除 PendSV 异常的挂起状态。
PENDSVSET	28	rw	**PendSV 挂起置位** 该位将一个 PendSV 异常的挂起状态置 1，或回读当前状态。 0_B PendSV 异常未挂起。 1_B PendSV 异常挂起。 *注：　向该位写 1 是将 PendSV 异常设置为挂起状态的唯一方法。* 软件向该位写 0 无效。
0	[31:29]	r	**保留** 读出值为 0；应写入 0。

1) 该值与 IPSR 位 [5:0] 的值相同，见 **中断程序状态寄存器**。

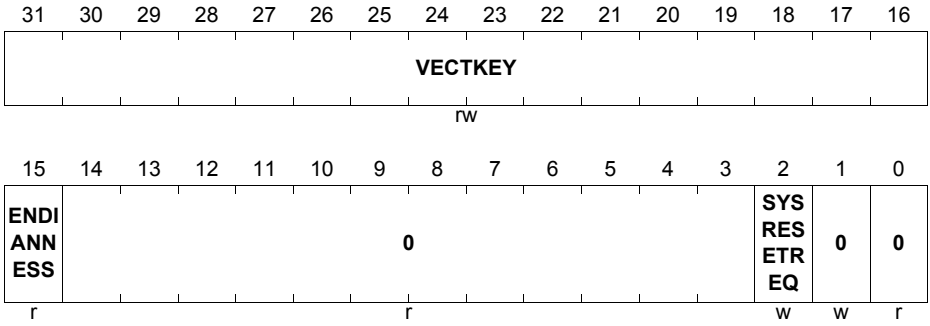
注：　如果发生下面的情况，结果是不可预知的：

1. *PENDSVSET 和 PENDSVCLR 位都被置 1。*
2. *PENDSTSET 和 PENDSTCLR 位都被置 1。*

AIRCR

AIRCR 寄存器提供数据访问的大小端状态和系统复位控制。要对该寄存器写入，必须向 VECTKEY 域写 0x5FA，否则处理器会忽略该写操作。

AIRCR
应用中断和复位控制寄存器　　(E000ED0C$_H$)　　复位值 : FA050000$_H$



域	位	类型	描述
0	0	r	**保留** 读出值为 0；应写入 0。
0	1	w	**保留** 必须写 0。
SYSRESETREQ	2	w	**系统复位请求** 0_B　无效。 1_B　请求一次系统级复位。 该位的读出值为 0。
0	[14:3]	r	**保留** 读出值为 0；应写入 0。
ENDIANNESS	15	r	**数据端格式** 0_B　小端
VECTKEY	[31:16]	rw	**寄存器密钥** 读出值不确定。 写入时，要向 VECTKEY 写 0x5FA，否则该写操作被忽略。

SCR

SCR 控制进入和退出低功耗状态的特性。

SCR
系统控制寄存器 **(E000ED10$_H$)** **复位值 : 00000000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											SEVONPEND	0	SLEEPDEEP	SLEEPONEXIT	0
r											rw	r	rw	rw	r

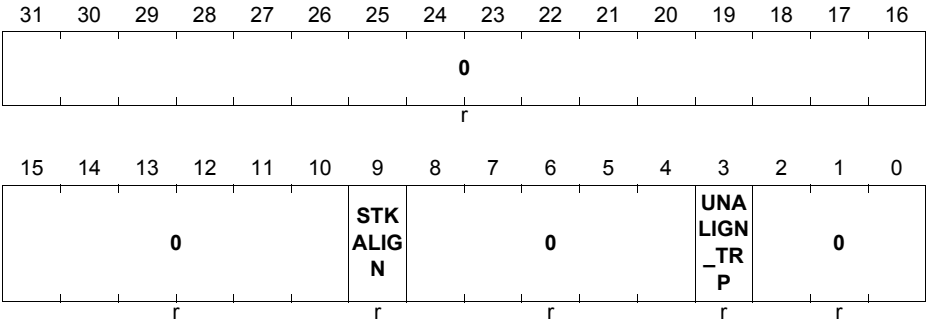
域	位	类型	描述
0	0	r	**保留** 读出值为 0；应写入 0。
SLEEPONEXIT	1	rw	**退出后休眠（Sleep-on-exit）** 该位指示从异常处理模式返回到线程模式后是否休眠。 0_B 返回线程模式时不休眠。 1_B 从一个 ISR 返回到线程模式时进入休眠或深度休眠。 将该位置 1 即可使一个应用变成中断驱动式应用，以避免返回到空主程序。
SLEEPDEEP	2	rw	**低功耗休眠模式** 该位控制处理器使用休眠或深度休眠作为其低功耗模式。 0_B 休眠 1_B 深度休眠
0	3	r	**保留** 读出值为 0；应写入 0。

域	位	类型	描述
SEVONPEND	4	rw	**挂起位置 1 时的事件发送控制** 0_B 只有被使能的中断或事件能唤醒处理器，被禁止的中断不能唤醒处理器。 1_B 被使能的事件和所有中断，包括被禁止的中断，都可以唤醒处理器。 当一个事件或中断进入挂起状态时，该事件信号将处理器从等待事件状态 （WFE）唤醒。如果处理器不在等待某个事件，则该事件被记录下来并且影响下一个 WFE。
0	[31:5]	r	**保留** 读出值为 0；应写入 0。

CCR

CCR 是一个只读寄存器，它指示 Cortex-M0 处理器行为的某些方面。

CCR
配置和控制寄存器 **(E000ED14$_H$)** **复位值 : 00000208$_H$**



域	位	类型	描述
0	[2:0]	r	**保留** 读出值为 0；应写入 0。
UNALIGN_TRP	3	r	**未对齐访问陷阱** 该位的读出值总是为 1，指示所有未对齐的访问都会产生硬故障。
0	[8:4]	r	**保留** 读出值为 0；应写入 0。
STKALIGN	9	r	**堆栈对齐** 该位的读出值总是为 1，指示异常进入时按 8 字节堆栈对齐。 在异常进入时，处理器使用堆叠的 PSR 的位 [9] 指示堆栈对齐情况。在从异常返回时，处理器使用该堆叠位恢复正确的堆栈对齐。
0	[31:10]	r	**保留** 读出值为 0；应写入 0。

系统处理程序优先级寄存器

SHPR2 和 SHPR3 寄存器为那些具有可编程优先级的异常处理程序设置优先级，其范围为 0 ~192。

SHPR2 和 SHPR3 需要按字访问。使用 CMSIS 访问系统异常优先级时，要使用下面的 CMSIS 函数：

- uint32_t NVIC_GetPriority(IRQn_Type IRQn)
- void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)

系统故障处理程序、每个异常处理程序的优先级域和寄存器如表 2-14 所示。

表 2-14　　系统故障处理程序优先级域

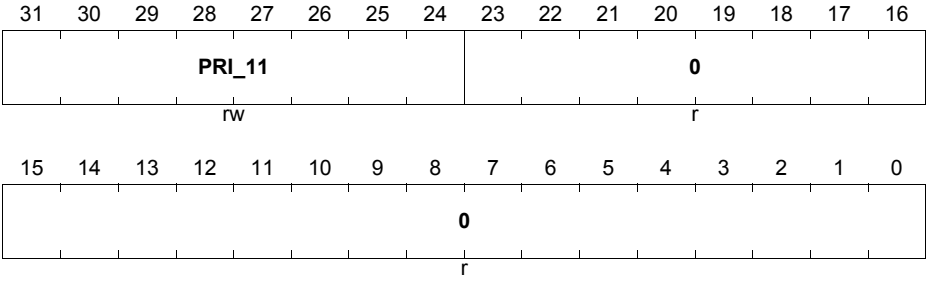
故障处理程序	域	寄存器描述
SVCall	PRI_11	系统处理程序优先级寄存器 2，见页 2-41
PendSV	PRI_14	系统处理程序优先级寄存器 3，见页 2-42
SysTick	PRI_15	

每个 PRI_N 域都是 8 位宽，但 XMC1300 只使用了每个域的位 [7:6]，位 [5:0] 的读出值为零，对这些位的写操作被忽略。

SHPR2

SHPR2 寄存器设置 SVCall 处理程序的优先级。

SHPR2
系统处理程序优先级寄存器 2　　(E000ED1C$_H$)　　复位值 : 00000000$_H$



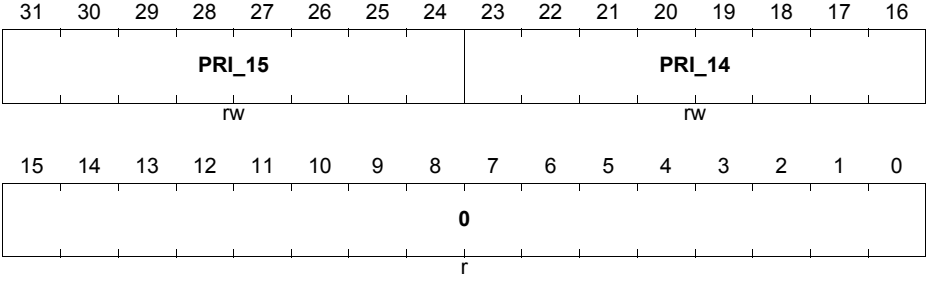
域	位	类型	描述
0	[23:0]	r	**保留** 读出值为 0；应写入 0。

域	位	类型	描述
PRI_11	[31:24]	rw	**系统处理程序 11 的优先级** SVCall。

SHPR3

SHPR3 寄存器设置 PendSV 和 SysTick 处理程序的优先级。

SHPR3
系统处理程序优先级寄存器 3 **(E000ED20$_H$)** **复位值 : 00000000$_H$**

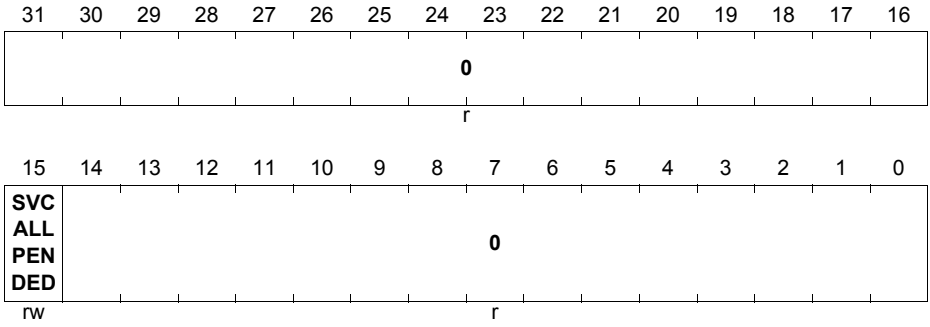


域	位	类型	描述
0	[15:0]	r	**保留** 读出值为 0；应写入 0。
PRI_14	[23:16]	rw	**系统处理程序 14 的优先级** PendSV。
PRI_15	[31:24]	rw	**系统处理程序 15 的优先级** SysTick 异常。

SHCSR

SHCSR 寄存器控制系统处理程序并提供其状态。

SHCSR
系统处理程序控制和状态寄存器　　(E000ED24$_H$)　　复位值：00000000$_H$



域	位	类型	描述
0	[14:0]	r	**保留** 读出值为 0；应写入 0。
SVCALLPENDED	15	rw	**SVCall 挂起位** 读时，该位反映挂起状态；写时，该位将挂起状态更新为写入值。 0_B　SVCall 不处于挂起状态。 1_B　SVCall 处于挂起状态[1]。
0	[31:16]	r	**保留** 读出值为 0；应写入 0。

1) 挂起状态位在异常发生时被置 1，在异常变为活动状态时被清 0。

2.9.2　SysTick 寄存器

SYST_CSR

SYST_CSR 寄存器使能 SysTick 定时器的功能。
读 SYST_CSR 时会将 COUNTFLAG 位清 0。

SYST_CSR
SysTick 控制和状态寄存器 **(E000E010$_H$)** **复位值：00000000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															COUNTFLAG
r															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													CLKSOURCE	TICKINT	ENABLE
r													rw	rw	rw

域	位	类型	描述
ENABLE	0	rw	**计数器使能** 该位使能 SysTick 计数器。 0_B 计数器被禁止。 1_B 计数器被使能。
TICKINT	1	rw	**SysTick 异常请求** 该位使能 SysTick 的异常请求。 0_B 向下计数到零时不产生 SysTick 异常请求。 1_B 向下计数到零时产生 SysTick 异常请求。 在软件中，COUNTFLAG 位可用于确定 SysTick 是否已计数到零。
CLKSOURCE	2	rw	**时钟源** 该位选择 SysTick 定时器的时钟源。 0_B 外部时钟[1)]。 1_B 处理器时钟。
0	[15:3]	r	**保留** 读出值为 0；应写入 0。
COUNTFLAG	16	rw	**计数器标志** 如果自上一次读该寄存器以来定时器已计数到零，该位返回 1。
0	[31:17]	r	**保留** 读出值为 0；应写入 0。

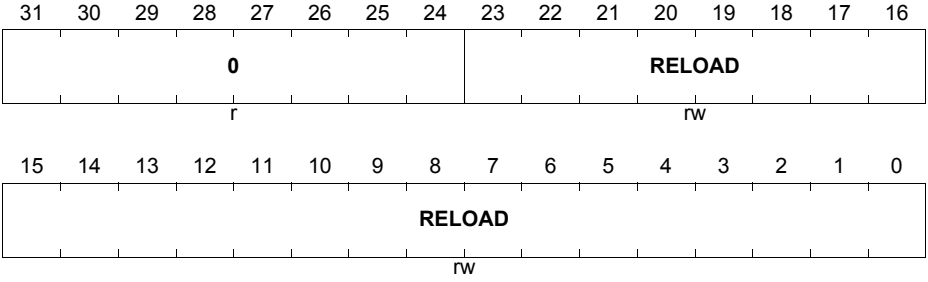
1) 在 XMC1300 中，外部时钟是指片内 32 kHz 待机时钟。

当 ENABLE 被置 1 时，计数器从 SYST_RVR 寄存器加载 RELOAD 值，然后向下计数。计数到 0 时，计数器将 COUNTFLAG 置 1，同时根据 TICKINT 的值决定产生或不产生 SysTick 异常请求。然后，计数器重新加载 RELOAD 值，并开始计数。

SYST_RVR

SYST_RVR 寄存器指定要加载到 SYST_CVR 寄存器的起始值。

SYST_RVR
SysTick 重载值寄存器 **(E000E014$_H$)** **复位值 : XXXXXXXX$_H$**



域	位	类型	描述
RELOAD	[23:0]	rw	**重载值** 该域设置要加载到 SYST_CVR 寄存器的值，该值在SysTick计数器被使能并且其计数值达到0时被加载。
0	[31:24]	r	**保留** 读出值为 0；应写入 0。

关于计算 RELOAD 值的注释

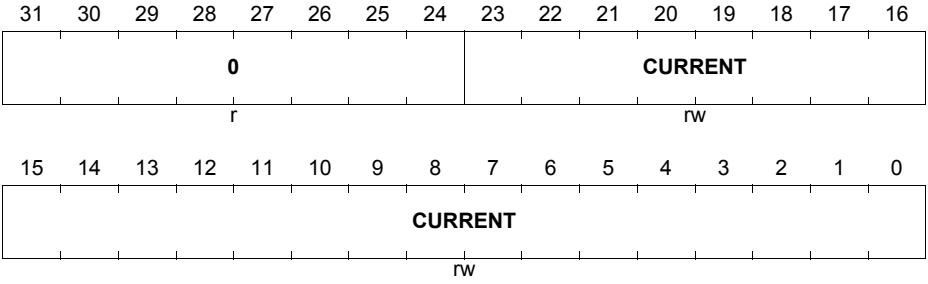
1. *RELOAD 值可以是 0x00000001-0x00FFFFFF 范围内的任何值。起始值可以为 0，但不起作用。因为 SysTick 异常请求和 COUNTFLAG 仅在从 1 计到 0 时被激活。*
2. *RELOAD 值根据其用途计算 。例如，要产生一个周期为 N 个处理器时钟的多拍定时器，需要使用的 RELOAD 值为 N-1。如果需要每 100 个时钟脉冲产生一次中断，应将 RELOAD 设置为 99。*

SYST_CVR

SYST_CVR 寄存器含有 SysTick 计数器的当前值。

写 SYST_CVR 时会将其清 0，并将 COUNTFLAG 状态位清 0。写该寄存器不会触发 SysTick 异常逻辑。读该寄存器返回其被访问时的值。

SYST_CVR
SysTick 当前值寄存器 **(E000E018$_H$)** **复位值 : XXXXXXXX$_H$**

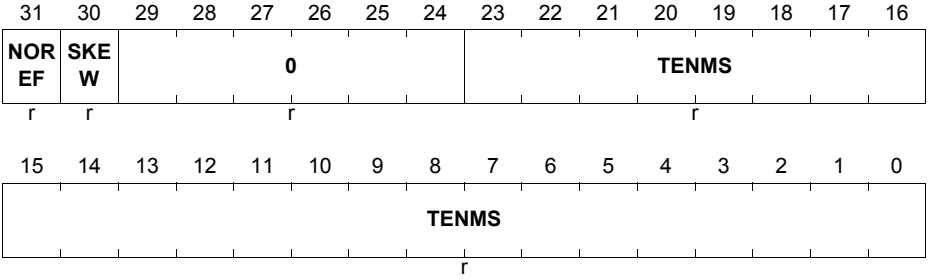


域	位	类型	描述
CURRENT	[23:0]	rw	**SysTick 计数器当前值** 读该寄存器时，这些位返回 SysTick 计数器的当前值。 写任何值都会将该域清 0，同时也将 SYST_CSR.COUNTFLAG 位清 0。
0	[31:24]	r	**保留** 读出值为 0；应写入 0。

SYST_CALIB

SYST_CALIB 寄存器指示 SysTick 的校准特性。

SYST_CALIB
SysTick 校准值寄存器 **(E000E01C$_H$)** **复位值 : 40000147$_H$**



位域	位	类型	描述
TENMS	[23:0]	r	**10 毫秒** 用于 10ms 定时的重载值易受系统时钟偏斜误差的影响。 TENMS 的缺省值是 0x000147。
0	[29:24]	r	**保留** 读出值为 0 ；应写入 0。
SKEW	30	r	**时钟偏斜** 该位的读出值为 1。它指示由于时钟频率的原因，10ms 的校准值是不精确的。
NOREF	31	r	**参考时钟** 该位的读出值为 0。它指示使用外部提供的参考时钟。

3 总线系统

XMC1300 中的单主总线系统由一个基于行业内 AMBA 3 AHB-Lite 协议标准的、针对存储器和高带宽片内外设的高性能系统总线和一个宽度较窄的、针对低带宽片内外设的 APB 总线组成。

3.1 总线接口

本章描述两种接口的功能特性。

- 存储器接口
- 外设接口

所有片内模块都采用小端数据格式实现。

存储器接口

片内存储器能在每个总线时钟周期接受一次传送请求。

存储器接口数据总线的宽度为32位。Flash存储器仅支持32位访问，而 SRAM允许 32位、16 位和 8 位写访问。对 SRAM 的读访问总是 32 位宽。

外设接口

AHB-Lite 总线上的每个从设备都支持 32 位访问。另外：

- USIC0 支持 8 位和 16 位访问
- MATH 协处理器支持对除法器寄存器的 16 位访问

APB 总线上的每个从设备都只支持 16 位访问。

注： 对存储器或外设从设备的非对齐存储器访问会导致硬故障异常。

4 服务请求处理

在 XMC1300 系统中，硬件脉冲被称为系统请求 (SR) 。系统请求是在有连接的片内资源之间发送触发 “ 消息 ” 的最快速的方式。

一个 SR 可产生下面任何一种请求：

- 中断
- 外设动作

本章描述可用的服务请求和选择及处理这些请求的不同方式。

表 4-1 缩略语

ERU	事件请求单元
NVIC	嵌套向量中断控制器
SR	服务请求

4.1 概述

高效的服务请求处理是以请求源与请求处理单元之间的互连为基础的。XMC1300 提供了固定和可编程的互连方案。

4.1.1 特性

系统为服务请求处理提供了以下特性：

- 服务请求与请求处理单元之间的连接矩阵
  - 固定连接
  - 使用 ERU 的可编程连接

4.1.2 原理框图

图 4-1 示出了请求源与请求处理单元之间的交互作用。

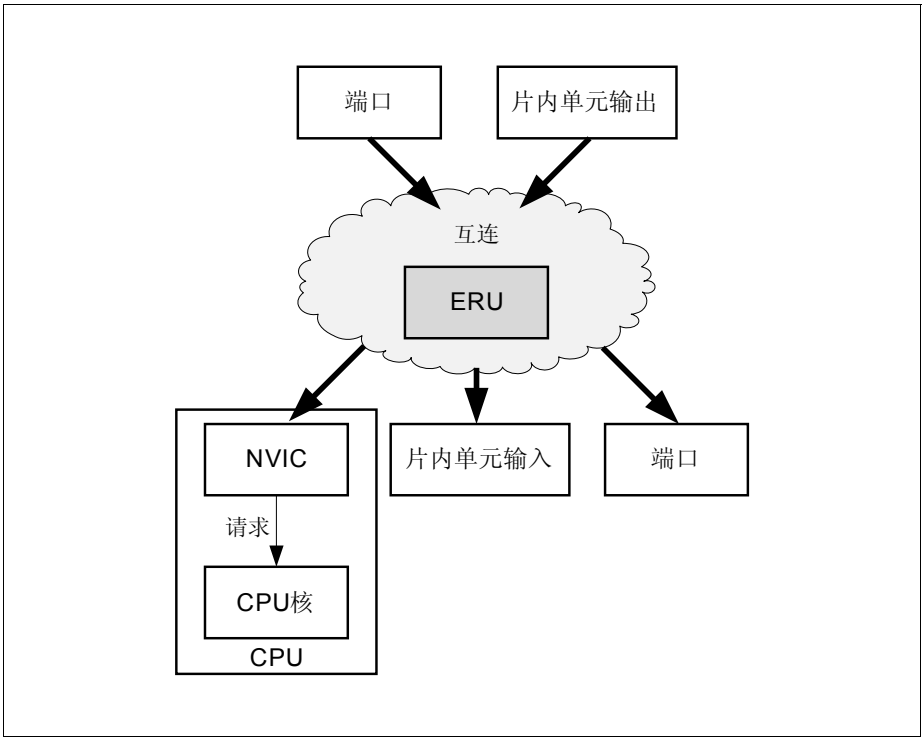


图 4-1　　服务请求处理原理框图

4.2　　服务请求分配

图 4-2 示出了如何并发分配服务请求的例子。为了支持并发分配到多个接收者，接收模块可以使能 / 禁止进入的请求。

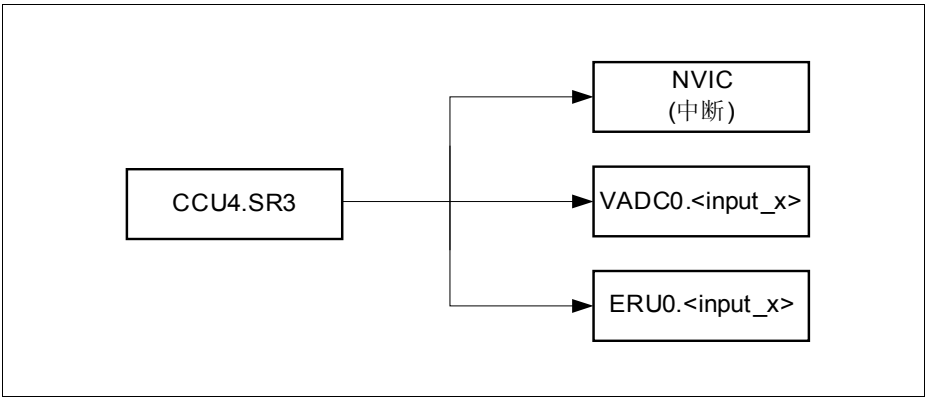


图 4-2　　服务请求分配示例

与服务请求分配相关的单元可被划分为：

- 嵌入式实时服务
- 中断服务

嵌入式实时服务

片内单元与 端口 （PORTS）之间的连接是实时应用，而且与芯片的封装相关。芯片引脚的相关连接和可用性可以在下面的章节中查到：

- 模块所在章节的 “ 互连 ” 部分
- “ 并行端口 ” 一章和数据手册中有关端口的内容
- “ 事件请求单元 ” 一章

中断服务

下面的表 4-2 给出了每个模块的服务请求数量，并说明了这些服务请求如何被分配给 NVIC 中断服务提供者。

在 XMC1300 中，服务请求的类型都是 “ 脉冲 ”。

表 4-2　　各模块的中断服务

模块	请求源	NVIC	类型
VADC	12	6	脉冲
CCU40	4	4	脉冲
CCU80	4	2	脉冲
POSIF0	2	2	脉冲
USIC0	6	6	脉冲

表 4-2 各模块的中断服务 (续表)

模块	请求源	NVIC	类型
BCCU0	1	1	脉冲
MATH	1	1	脉冲
SCU	3	3	脉冲
ERU0	4	4	脉冲
总计	37	29	-

5 中断子系统

XMC1300 中的中断子系统由嵌套向量中断控制器 (NVIC) 和各功能模块的中断产生模块组成。

注： CPU 异常模型描述见 CPU 一章。

5.1 嵌套向量中断控制器 (NVIC)

NVIC 是 Cortex M0 处理器单元的一个组成部分。因为与 CPU 紧密接合，所以它能提供最低的中断延迟和对晚到中断的高效处理。

5.1.1 特性

NVIC 具有以下特性：

- 32 个中断节点
- 每个中断节点有 4 个可编程优先级
- 支持尾链和晚到中断
- 软件产生中断

5.1.2 中断节点分配

表 5-1 按外设列出了服务请求源以及为其分配的 NVIC 中断节点。若要计算向量例程地址，请参考 CPU 一章中的向量表一节。

表 5-1 中断节点分配

服务请求	节点 ID	描述
SCU.SR0 - SCU.SR2	0...2	系统控制 SR0 是系统关键请求 SR1 是公共 SCU 请求 SR2 是比较器 (ACMPx 和 ORCx) 请求
ERU0.SR0 - ERU0.SR3	3...6	外部请求单元 0
MATH.SR0 - MATH.SR1	7	MATH
NC	8	保留
USIC0.SR0 - USIC0.SR5	9...14	通用串行接口通道 (模块 0)
VADC0.C0SR0 - VADC0.C0SR1	15...16	模 / 数转换器 (公用)
VADC0.G0SR0 - VADC0.G0SR1	17...18	模 / 数转换器 (组 0)

表 5-1 中断节点分配 (续表)

服务请求	节点 ID	描述
VADC0.G1SR0 - VADC0.G1SR1	19...20	模 / 数转换器 (组 1)
CCU40.SR0 - CCU40.SR3	21...24	捕获比较单元 4 (模块 0)
CCU80.SR0 - CCU80.SR1	25...26	捕获比较单元 8 (模块 0)
POSIF0.SR0 - POSIF0.SR1	27...28	位置接口 (模块 0)
NC	29...30	保留
BCCU0.SR0	31	亮度控制单元 (模块 0)

5.1.3 中断信号产生

在 XMC1300 中，所有外设都只支持脉冲中断产生。脉冲中断也被描述为边沿触发中断。

脉冲中断是在处理器时钟 (MCLK) 的上升沿同步采样的中断信号。为保证 NVIC 检测到中断，外设要将中断信号保持有效至少一个 MCLK 时钟周期，在此期间 NVIC 检测到中断脉冲并将该中断锁存。

当处理器进入 ISR 时，它会自动清除中断的挂起状态，详见中断的硬件和软件控制。

处理器在异常进入时自动将其状态入栈，在异常退出时自动将其状态出栈，无任何指令开销。该机制提供低延迟的的异常处理。

中断的硬件和软件控制

Cortex-M0 锁存所有中断。外设中断会因为下述原因之一而变成挂起状态：

- NVIC 检测到中断信号激活，但中断未被激活
- NVIC 检测到中断信号的上升沿
- 软件写对应的中断挂起置位寄存器位，见中断挂起置位寄存器 NVIC_ISPR

一个挂起中断会一直保持其挂起状态，直到发生下述条件之一：

- 处理器进入该中断的 ISR。这使中断的状态从挂起变为活动。然后：
  - NVIC 继续监视该中断信号。如果出现中断脉冲，则中断状态变为挂起并活动。在这种情况下，当处理器从 ISR 返回时，中断状态变为挂起，可能会导致处理器立即重新进入该中断的 ISR。
    如果在处理器执行 ISR 期间中断信号未产生脉冲，则中断的状态在处理器从 ISR 返回时变为非活动状态。
- 软件写对应的中断挂起清除寄存器位。
  - 如果中断原来为挂起状态，则变为非活动状态；如果中断原来为活动且挂起状态，则变为活动状态。

5.1.4 NVIC 设计提示

即使在被禁止的情况下，中断节点也可以进入挂起状态。禁止一个中断节点只能防止处理器从该节点进入中断处理。

NVIC 编程提示

软件使用 CPSIE i 和 CPSID i 指令使能和禁止中断。CMSIS 为这些指令提供下列内部函数：

```
void __disable_irq(void)  // 禁止中断
void __enable_irq(void)  // 使能中断
```

另外， CMSIS 还提供一些用于 NVIC 控制的函数，包括：

表 5-2 用于 NVIC 控制的 CMSIS 函数

CMSIS 中断控制函数	描述
void NVIC_EnableIRQ (IRQn_t IRQn)	使能 IRQn
void NVIC_DisableIRQ (IRQn_t IRQn)	禁止 IRQn
uint32_t NVIC_GetPendingIRQ (IRQn_t IRQn)	如果 IRQn 为挂起状态，返回真 (1)
void NVIC_SetPendingIRQ (IRQn_t IRQn)	将 IRQn 置为挂起状态
void NVIC_ClearPendingIRQ (IRQn_t IRQn)	清除 IRQn 的挂起状态
void NVIC_SetPriority (IRQn_t IRQn, uint32_t priority)	设置 IRQn 的优先级
uint32_t NVIC_GetPriority (IRQn_t IRQn)	读取 IRQn 的优先级
void NVIC_SystemReset (void)	复位系统

输入参数 IRQn 是 IRQ 编号。要获取有关这些函数的详细信息，请参见 CMSIS 文档。

5.1.5 使用 CMSIS 函数访问 CPU 寄存器

CMSIS 函数使不同 Cortex-M 处理器之间的软件移植成为可能。使用 CMSIS 访问 NVIC 寄存器时，要使用下列函数：

表 5-3 CMSIS 访问 NVIC 的函数

CMSIS 函数	描述
void NVIC_EnableIRQ (IRQn_Type IRQn)[1)]	使能一个中断或异常。
void NVIC_DisableIRQ (IRQn_Type IRQn)[1)]	禁止一个中断或异常。
void NVIC_SetPendingIRQ (IRQn_Type IRQn)[1)]	将中断或异常的挂起状态置 1。

表 5-3 CMSIS 访问 NVIC 的函数 (续表)

CMSIS 函数	描述
void NVIC_ClearPendingIRQ (IRQn_Type IRQn)[1)]	将中断或异常的挂起状态清 0。
uint32_t NVIC_GetPendingIRQ (IRQn_Type IRQn)[1)]	读取中断或异常的挂起状态。 如果挂起状态被置 1，该函数返回非零值。
void NVIC_SetPriority(IRQn_Type IRQn, uint32_t priority)[1)]	设置具有可编程优先级的中断或异常的优先级。
uint32_t NVIC_GetPriority(IRQn_Type IRQn)[1)]	读取具有可编程优先级的中断或异常的优先级。 该函数返回当前优先级。

1) 输入参数 IRQn 是 IRQ 的编号。

5.1.6 中断优先级

可以为中断节点分配四个优先级之一。优先级的步距为 64，其值从 0 到 192，由中断优先级寄存器 (IPRx) 中的 8 位优先级域定义。较高的优先级值对应较低的优先级，因此优先级值 0 对应最高的中断优先级。

由于每个 IPRx 寄存器中有 4 个优先级域，并且每个域对应一个中断节点，所以共需 8 个 IPRx 寄存器 (IPR0...IPR7)，如 图 5-1 所示。

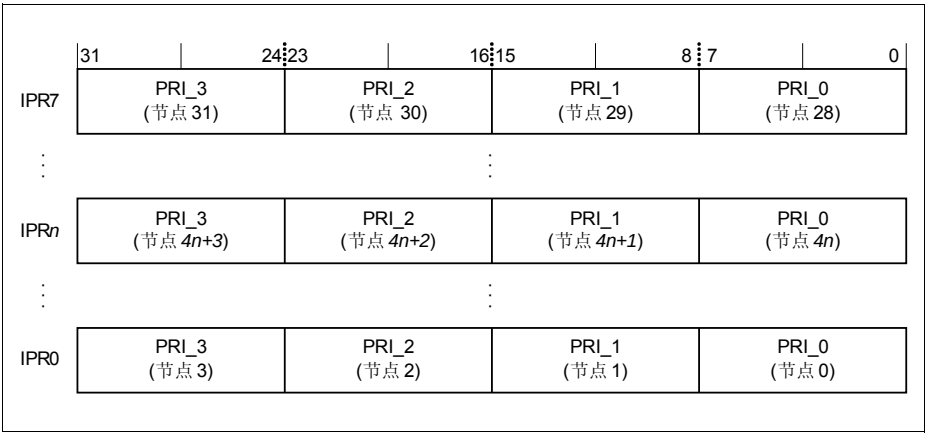


图 5-1 中断优先级寄存器

中断节点 m (0...31) 的 IPR 编号和字节偏移可用下面的方法获得：

- 对应的 IPR 编号 n 由下面的公式给出：
  n = m DIV 4
- 在该寄存器中，所需要的优先级域的字节偏移为 m MOD 4，其中：
  – 字节偏移 0 对应寄存器位 [7:0]
  – 字节偏移 1 对应寄存器位 [15:8]
  – 字节偏移 2 对应寄存器位 [23:16]
  – 字节偏移 3 对应寄存器位 [31:24]
- 例如，中断节点 21 的优先级域位于 IPR5.[15:8]，因为
  – n = 21 DIV 4 = 5
  – 字节偏移 = 21 MOD 4 = 1

注： IPRx 寄存器只能按字访问。

有关访问中断节点优先级配置的详细信息，请参见**表 5-2** 。该表提供了中断节点优先级的软件视图。

5.1.7 中断响应时间

中断响应时间被定义为从检测到产生的中断脉冲并由 NVIC 锁存该中断开始到执行中断处理程序的第一条指令的时间，该时间一般为 21 个 MCLK 时钟周期， 如 **图 5-2** 所示。

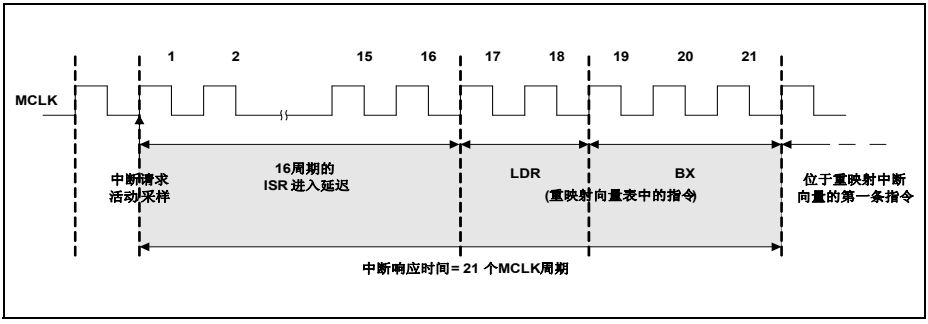


图 5-2 典型的中断响应时间

该中断响应时间基于下面的条件：

- 中断产生被使能
- 不发生中断抢占、晚到中断或尾链中断
- 不考虑因存储器等待状态产生的延迟

5.2 一般模块的中断结构

一个模块可能有多个中断源。每个中断源一般都有下述结构 (见 **图 5-3**)：

- 一个中断源状态标志
- 一个置 1 位，允许软件将状态标志置 1
- 一个清 0 位，允许软件将状态标志清 0

- 一个使能位，用于在发生硬件事件或状态标志置 1 位被置位 (即软件触发的中断) 时触发中断

注：　如果一个标志置 1 事件 (因为一个外设硬件事件) 与一个标志清 0 事件 (因为软件将清 0 位置 1) 在同一时钟周期发生，则置 1 比清 0 有更高的优先级。

注：　硬件事件将状态标志置 1 或软件写状态标志置 1 位都与中断产生是否被使能／禁止无关。类似地，中断产生与状态标志的优先级无关。

另外，有些模块的中断源可能比中断信号线多。因此，它们包含一个中断路由管理单元，能将中断源映射到中断信号线。

更详细的信息以及与上述一般结构不同的例外情况，请参见描述各模块的章节。本章最后给出所有 XMC1300 中断源的一览表。

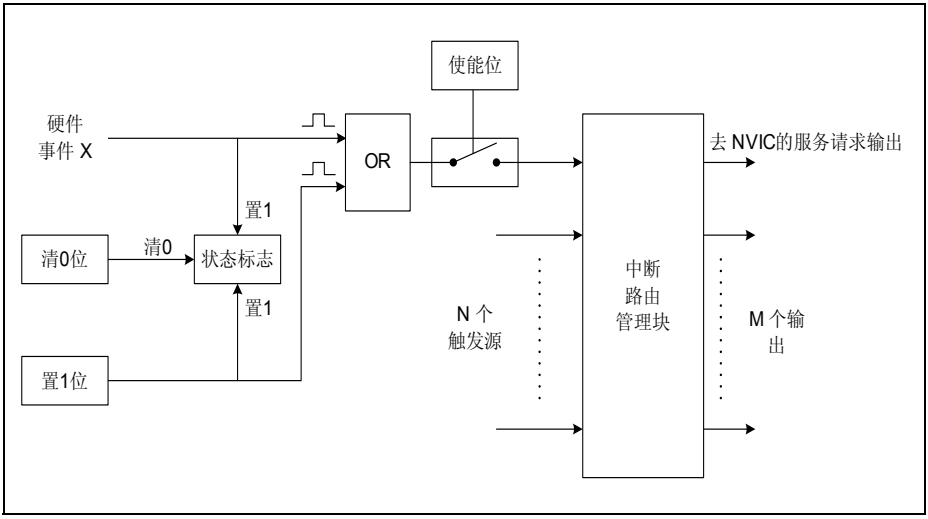


图 5-3　　典型的模块中断结构

要使一个模块的硬件事件能产生中断，软件必须：

- 通过 NVIC_ISER 寄存器使能 NVIC 中被分配给该模块的中断节点。
- 如果模块有一个中断路由管理单元，选择一个可用的服务请求输出，通过该输出可以向 NVIC 产生中断。这通常通过配置模块中的一个中断节点指针寄存器完成。
- 最后，将用于产生中断的模块硬件事件的中断使能位置 1。

5.3 寄存器

表 5-4 寄存器地址空间

模块	基地址	结束地址	备注
CPU PPB: 系统控制空间 (SCS)	$E000E000_H$	$E000EFFF_H$	

寄存器概览

绝对寄存器地址通过下面的加法公式计算：

模块基地址 + 偏移地址

表 5-5 寄存器一览表

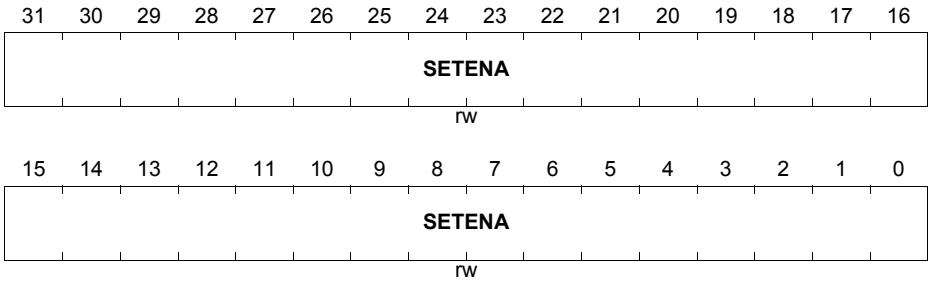
简称	描述	偏移地址	访问方式		描述见
			读	写	
嵌套向量中断控制器 (NVIC)					
NVIC_ISER	中断使能置位寄存器	100_H	U, PV	U, PV	页 5-7
NVIC_ICER	中断使能置零寄存器	180_H	U, PV	U, PV	页 5-9
NVIC_ISPR	中断挂起置位寄存器	200_H	U, PV	U, PV	页 5-10
NVIC_ICPR	中断挂起清零寄存器	280_H	U, PV	U, PV	页 5-11
NVIC_IPR0 - NVIC_IPR7	中断优先级寄存器	400_H-$41C_H$	U, PV	U, PV	页 5-12

5.3.1 NVIC 寄存器

NVIC_ISER

ISER 寄存器使能中断节点，并指示哪些中断节点被使能。

NVIC_ISER
中断使能置位寄存器 **($E000E100_H$)** **复位值：00000000_H**



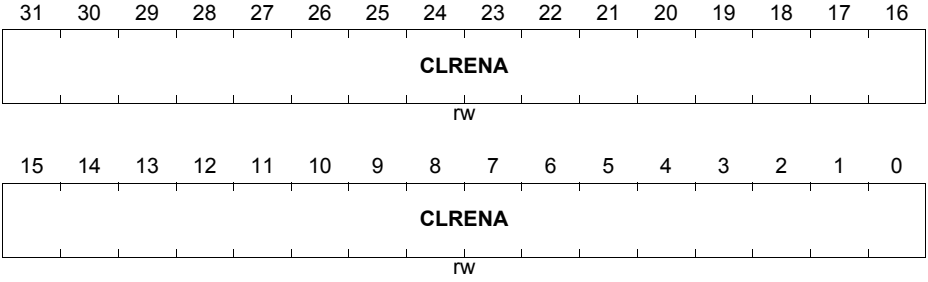
域	位	类型	描述
SETENA	[31:0]	rw	**中断节点使能置位** 0_B 读：中断节点被禁止。 写：无效。 1_B 读：中断节点被使能。 写：使能中断节点

如果一个挂起中断被使能，则 NVIC 会根据优先级激活该中断。如果一个中断未被使能，中断信号有效会将中断状态变为挂起，但 NVIC 绝不会激活该中断，无论其优先级如何。

NVIC_ICER

ICER 寄存器禁止中断节点，并指示哪些中断节点被使能。

NVIC_ICER
中断使能清零寄存器 **(E000E180$_H$)** **复位值 : 00000000$_H$**

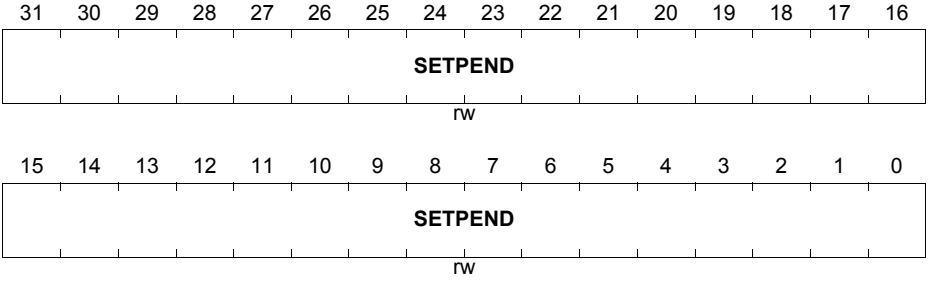


域	位	类型	描述
CLRENA	[31:0]	rw	**中断节点使能清零** 0_B 读：中断节点被禁止。 写：无效。 1_B 读：中断节点被使能。 写：禁止中断节点。

NVIC_ISPR

ISPR 寄存器强制中断节点进入挂起状态，并指示哪些中断节点处于挂起状态。

NVIC_ISPR
中断挂起置位寄存器 **(E000E200$_H$)** **复位值 : 00000000$_H$**



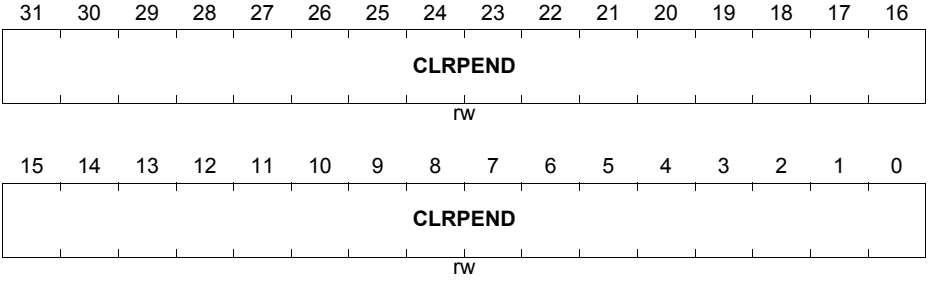
域	位	类型	描述
SETPEND	[31:0]	rw	**中断节点挂起置位** 0_B 读：中断节点未挂起。 写：无效。 1_B 读：中断节点被挂起。 写：将中断状态变为挂起。

注：　向一个 ISPR 位写 1 对应于：
- 对处于挂起状态的中断节点不起作用。
- 对被禁止的中断节点，将其状态设置为挂起。

NVIC_ICPR

ICPR 寄存器清除中断节点的挂起状态，并指示哪些中断节点处于挂起状态。

NVIC_ICPR
中断挂起清零寄存器 **(E000E280$_H$)** **复位值 : 00000000$_H$**



域	位	类型	描述
CLRPEND	[31:0]	rw	**中断节点挂起清零** 0_B 读：中断节点未挂起。 写：无效。 1_B 读：中断节点被挂起。 写：清除中断节点的挂起状态。

注： 向一个 ICPR 位写 1 不影响对应中断节点的活动状态。

NVIC_IPRx (x=0-7)

IPR0-IPR7 寄存器为每个中断节点提供一个 8 位的优先级域。每个寄存器保存 4 个优先级域。

每个优先级域保持一个优先级值，0-192。优先级值越低，对应中断节点的优先级越高。处理器只使用了每个域的位 [7:6]，位 [5:0] 的读出值为 0，对这些位的写操作被忽略。这意味着向一个优先级寄存器写入 255 时，实际保存到该寄存器的值为 192。

NVIC_IPRx (x=0-7)
中断优先级寄存器 x **($E000E400_H$ + 4*x)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PRI_3								PRI_2							
rw								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PRI_1								PRI_0							
rw								rw							

域	位	类型	描述
PRI_3	[31:24]	rw	**优先级，字节偏移 3**
PRI_2	[23:16]	rw	**优先级，字节偏移 2**
PRI_1	[15:8]	rw	**优先级，字节偏移 1**
PRI_0	[7:0]	rw	**优先级，字节偏移 0**

5.4 中断请求源概览

下面的几页给出了所有 XMC1300 中断源及其相关寄存器位的一览表。

表 5-6 中断源一览表

IRQ	中断节点	中断源	状态标志		中断使能		置位标志		清除标志		节点指针	
			寄存器	位	寄存器	位	寄存器	位	寄存器	位	寄存器	位
0	SCU.SR0	Flash double bit ECC[1)]	SCU_SRRAW	FLECC2I	SCU_SRMSK	FLECC2I	SCU_SRSET	FLECC2I	SCU_SRCLR	FLECC2I	-	-
			NVM_NVMSTATUS	ECC2READ	-	-	-	-	NVM_NVMPROG	RSTECC	-	-
		Flash operation complete	SCU_SRRAW	FLCMPLTI	NVM_NVMCONF	INT_ON	SCU_SRSET	FLCMPLTI	SCU_SRCLR	FLCMPLTI	-	-
		SRAM parity error	SCU_SRRAW	PESRAMI	SCU_SRMSK	PESRAMI	SCU_SRSET	PESRAMI	SCU_SRCLR	PESRAMI	-	-
		USIC RAM parity error	SCU_SRRAW	PEU0I	SCU_SRMSK	PEU0I	SCU_SRSET	PEU0I	SCU_SRCLR	PEU0I	-	-
		Loss of clock	SCU_SRRAW	LOCI	SCU_SRMSK	LOCI	SCU_SRSET	LOCI	SCU_SRCLR	LOCI	-	-

表 5-6　　中断源一览表 (续表)

IRQ	中断节点	中断源	状态标志		中断使能		置位标志		清除标志		节点指针	
			寄存器	位	寄存器	位	寄存器	位	寄存器	位	寄存器	位
1	SCU.SR1	Standby clock failure	SCU_SRRAW	SBYCLKFI	SCU_SRMSK	SBYCLKFI	SCU_SRSET	SBYCLKFI	SCU_SRCLR	SBYCLKFI	-	-
		VDDP pre-warning	SCU_SRRAW	VDDPI	SCU_SRMSK	VDDPI	SCU_SRSET	VDDPI	SCU_SRCLR	VDDPI	-	-
		VDDC drops below VDROP	SCU_SRRAW	VDROPI	SCU_SRMSK	VDROPI	SCU_SRSET	VDROPI	SCU_SRCLR	VDROPI	-	-
		VDDC rises above VCLIP	SCU_SRRAW	VCLIPI	SCU_SRMSK	VCLIPI	SCU_SRSET	VCLIPI	SCU_SRCLR	VCLIPI	-	-
		TSE done	SCU_SRRAW	TSE_DONE	SCU_SRMSK	TSE_DONE	SCU_SRSET	TSE_DONE	SCU_SRCLR	TSE_DONE	-	-
		TSE compare high	SCU_SRRAW	TSE_HIGH	SCU_SRMSK	TSE_HIGH	SCU_SRSET	TSE_HIGH	SCU_SRCLR	TSE_HIGH	-	-
		TSE compare low	SCU_SRRAW	TSE_LOW	SCU_SRMSK	TSE_LOW	SCU_SRSET	TSE_LOW	SCU_SRCLR	TSE_LOW	-	-
		WDT pre-warning	SCU_SRRAW	PRWARN	SCU_SRMSK	PRWARN	SCU_SRSET	PRWARN	SCU_SRCLR	PRWARN	-	-
		RTC periodic event	SCU_SRRAW	PI	-	-	SCU_SRSET	PI	SCU_SRCLR	PI	-	-
		RTC alarm	SCU_SRRAW	AI	-	-	SCU_SRSET	AI	SCU_SRCLR	AI	-	-
		RTC CTR Mirror Register updated	SCU_SRRAW	RTC_CTR	SCU_SRMSK	RTC_CTR	SCU_SRSET	RTC_CTR	SCU_SRCLR	RTC_CTR	-	-
		RTC ATIM0 Mirror Register updated	SCU_SRRAW	RTC_ATIM0	SCU_SRMSK	RTC_ATIM0	SCU_SRSET	RTC_ATIM0	SCU_SRCLR	RTC_ATIM0	-	-
		RTC ATIM1 Mirror Register updated	SCU_SRRAW	RTC_ATIM1	SCU_SRMSK	RTC_ATIM1	SCU_SRSET	RTC_ATIM1	SCU_SRCLR	RTC_ATIM1	-	-
		RTC TIM0 Mirror Register updated	SCU_SRRAW	RTC_TIM0	SCU_SRMSK	RTC_TIM0	SCU_SRSET	RTC_TIM0	SCU_SRCLR	RTC_TIM0	-	-
		RTC TIM1 Mirror Register updated	SCU_SRRAW	RTC_TIM1	SCU_SRMSK	RTC_TIM1	SCU_SRSET	RTC_TIM1	SCU_SRCLR	RTC_TIM1	-	-

表 5-6 中断源一览表 (续表)

IRQ	中断节点	中断源	状态标志		中断使能		置位标志		清除标志		节点指针	
			寄存器	位	寄存器	位	寄存器	位	寄存器	位	寄存器	位
2	SCU.SR2	Out of range comparator x event (x=0-7)	SCU_ SRRAW	ORCxI	SCU_ SRMSK	ORCxI	SCU_ SRSET	ORCxI	SCU_ SRCLR	ORCxI	-	-
		Analog comparator x event (x=0-2)	SCU_ SRRAW	ACMPxI	SCU_ SRMSK	ACMPxI	SCU_ SRSET	ACMPxI	SCU_ SRCLR	ACMPxI	-	-
3, 4, 5, 6	ERU0. SR[3:0]	ERU0_IOUTx (x=0-3)	See section on ERU0 for details.									
7	MATH. SR0	CORDIC end of calculation	MATH_ EVFR	CDEOC	MATH_ EVIER	CDEOCIEN	MATH_ EVFSR	CDEOCS	MATH_ EVFCR	CDEOCC	-	-
		CORDIC error	MATH_ EVFR	CDERR	MATH_ EVIER	CDERRIEN	MATH_ EVFSR	CDERRS	MATH_ EVFCR	CDERRS	-	-
	MATH. SR1	DIV end of calculation	MATH_ EVFR	DIVEOC	MATH_ EVIER	DIVEOCIEN	MATH_ EVFSR	DIVEOCS	MATH_ EVFCR	DIVEOCC	-	-
		DIV error	MATH_ EVFR	DIVERR	MATH_ EVIER	DIVERRIEN	MATH_ EVFSR	DIVERRS	MATH_ EVFCR	DIVERRC	-	-
8	Reserved											
9, 10, 11, 12, 13, 14	USIC0_SR[5:0]	USIC: Standard receive event	USIC0_ PSR	RIF	USIC0_ CCR	RIEN	-	-	USIC0_ PSCR	CRIF	USIC0_ INPR	RINP
		USIC: Receive start event	USIC0_ PSR	RSIF	USIC0_ CCR	RSIEN	-	-	USIC0_ PSCR	CRSIF	USIC0_ INPR	TBINP
		USIC: Alternate receive event	USIC0_ PSR	AIF	USIC0_ CCR	AIEN	-	-	USIC0_ PSCR	CAIF	USIC0_ INPR	AINP
		USIC: Transmit shift event	USIC0_ PSR	TSIF	USIC0_ CCR	TSIEN	-	-	USIC0_ PSCR	CTSIF	USIC0_ INPR	TSINP
		USIC: Transmit buffer event	USIC0_ PSR	TBIF	USIC0_ CCR	TBIEN	-	-	USIC0_ PSCR	CTBIF	USIC0_ INPR	TBINP
		USIC: Data lost event	USIC0_ PSR	DLIF	USIC0_ CCR	DLIEN	-	-	USIC0_ PSCR	CDLIF	USIC0_ INPR	PINP
		USIC: BRG event	USIC0_ PSR	BRGIF	USIC0_ CCR	BRGIEN	-	-	USIC0_ PSCR	CBRGIF	USIC0_ INPR	PINP

表 5-6　　中断源一览表 (续表)

IRQ	中断节点	中断源	状态标志		中断使能		置位标志		清除标志		节点指针	
			寄存器	位	寄存器	位	寄存器	位	寄存器	位	寄存器	位
9, 10, 11, 12, 13, 14	USIC0_SR[5:0]	USIC: Standard transmit buffer event	USIC0_TRBSR	STBI	USIC0_TBCTR	STBIEN	-	-	USIC0_TRBSCR	CSTBI	USIC0_TBCTR	STBINP
		USIC: Standard transmit buffer event	USIC0_TRBSR	STBT	USIC0_TBCTR	STBIEN	-	-	-	-	USIC0_TBCTR	STBINP
		USIC: Transmit Buffer error event	USIC0_TRBSR	TBERI	USIC0_TBCTR	TBERIEN	-	-	USIC0_TRBSCR	CTBERI	USIC0_TBCTR	ATBINP
		USIC: Standard receive buffer event	USIC0_TRBSR	SRBI	USIC0_RBCTR	SRBIEN	-	-	USIC0_TRBSCR	CSRBI	USIC0_RBCTR	SRBINP
		USIC: Standard receive buffer event	USIC0_TRBSR	SRBT	USIC0_RBCTR	SRBIEN	-	-	-	-	USIC0_RBCTR	SRBINP
		USIC: Alternate receive buffer event	USIC0_TRBSR	ARBI	USIC0_RBCTR	ARBIEN	-	-	USIC0_TRBSCR	CARBI	USIC0_RBCTR	ARBINP
		USIC: Receive buffer error event	USIC0_TRBSR	RBERI	USIC0_RBCTR	RBERIEN	-	-	USIC0_TRBSCR	CRBERI	USIC0_RBCTR	ARBINP
		ASC: Synchronisation break detected	USIC0_PSR	SBD	USIC0_PCR	SBDIEN	-	-	USIC0_PSCR	CSBD	USIC0_INPR	PINP
		ASC: Collision detected	USIC0_PSR	COL	USIC0_PCR	CDIEN	-	-	USIC0_PSCR	CCOL	USIC0_INPR	PINP
		ASC: Receiver noise detected	USIC0_PSR	RNS	USIC0_PCR	RNIEN	-	-	USIC0_PSCR	CRNS	USIC0_INPR	PINP
		ASC: Format error in stop bit 0	USIC0_PSR	FER0	USIC0_PCR	FEIEN	-	-	USIC0_PSCR	CFER0	USIC0_INPR	PINP
		ASC: Format error in stop bit 1	USIC0_PSR	FER1	USIC0_PCR	FEIEN	-	-	USIC0_PSCR	CFER1	USIC0_INPR	PINP
		ASC: Receive frame finished	USIC0_PSR	RFF	USIC0_PCR	FFIEN	-	-	USIC0_PSCR	CRFF	USIC0_INPR	PINP
		ASC: Transmit frame finished	USIC0_PSR	TFF	USIC0_PCR	FFIEN	-	-	USIC0_PSCR	CTFF	USIC0_INPR	PINP

表 5-6 中断源一览表 (续表)

IRQ	中断节点	中断源	状态标志		中断使能		置位标志		清除标志		节点指针	
			寄存器	位	寄存器	位	寄存器	位	寄存器	位	寄存器	位
9, 10, 11, 12, 13, 14	USIC0_SR[5:0]	SSC: MSLS event detected	USIC0_PSR	MSLSEV	USIC0_PCR	MSLSIEN	-	-	USIC0_PSCR	CMSLSEV	USIC0_INPR	PINP
		SSC: Parity error detected	USIC0_PSR	PAERR	USIC0_PCR	PARIEN	-	-	USIC0_PSCR	CPAERR	USIC0_INPR	PINP
		SSC: DX2T event detected	USIC0_PSR	DX2TEV	USIC0_PCR	DX2TIEN	-	-	USIC0_PSCR	CDX2TEV	USIC0_INPR	PINP
		IIC: Wrong TDF code detected	USIC0_PSR	WTDF	USIC0_PCR	ERRIEN	-	-	USIC0_PSCR	CWTDF	USIC0_INPR	PINP
		IIC: Start condition received	USIC0_PSR	SCR	USIC0_PCR	SCRIEN	-	-	USIC0_PSCR	CSCR	USIC0_INPR	PINP
		IIC: Repeated start condition received	USIC0_PSR	RSCR	USIC0_PCR	RSCRIEN	-	-	USIC0_PSCR	CRSCR	USIC0_INPR	PINP
		IIC: Stop condition received	USIC0_PSR	PCR	USIC0_PCR	PCRIEN	-	-	USIC0_PSCR	CPCR	USIC0_INPR	PINP
		IIC: NACK received	USIC0_PSR	NACK	USIC0_PCR	NACKIEN	-	-	USIC0_PSCR	CNACK	USIC0_INPR	PINP
		IIC: Arbitration lost	USIC0_PSR	ARL	USIC0_PCR	ARLIEN	-	-	USIC0_PSCR	CARL	USIC0_INPR	PINP
		IIC: Slave read request	USIC0_PSR	SRR	USIC0_PCR	SRRIEN	-	-	USIC0_PSCR	CSRR	USIC0_INPR	PINP
		IIC: Error detected	USIC0_PSR	ERR	USIC0_PCR	ERRIEN	-	-	USIC0_PSCR	CERR	USIC0_INPR	PINP
		IIC: ACK received	USIC0_PSR	ACK	USIC0_PCR	ACKIEN	-	-	USIC0_PSCR	CACK	USIC0_INPR	PINP
		IIS: DX2T event detected	USIC0_PSR	DX2TEV	USIC0_PCR	DX2TIEN	-	-	USIC0_PSCR	CDX2TEV	USIC0_INPR	PINP
		IIS: WA falling edge event	USIC0_PSR	WAFE	USIC0_PCR	WAFEIEN	-	-	USIC0_PSCR	CWAFE	USIC0_INPR	PINP
		IIS: WA rising edge event	USIC0_PSR	WARE	USIC0_PCR	WAREIEN	-	-	USIC0_PSCR	CWARE	USIC0_INPR	PINP
		IIS: WA generation end	USIC0_PSR	END	USIC0_PCR	ENDIEN	-	-	USIC0_PSCR	CEND	USIC0_INPR	PINP

表 5-6 中断源一览表 (续表)

IRQ	中断节点	中断源	状态标志		中断使能		置位标志		清除标志		节点指针	
			寄存器	位	寄存器	位	寄存器	位	寄存器	位	寄存器	位
15, 16,	VADC0_C0SR[1:0]	Source Event 0	VADC0_G xSEFLAG	SEV0	VADC0_G xQINR0	ENSI	VADC0_G xSEFLAG	SEV0	VADC0_G xSEFCLR	SEV0	VADC0_G xSEVNP	SEV0NP
17, 18, 19, 20		Source Event 1	VADC0_G xSEFLAG	SEV1	VADC0_G xASMR	ENSI	VADC0_G xSEFLAG	SEV1	VADC0_G xSEFCLR	SEV1	VADC0_G xSEVNP	SEV1NP
	VADC0_GxSR[1:0] (x=0-1)	Channel Event y (y=0-7)	VADC0_G xCEFLAG	CEVy	VADC0_G xCHCTRy	CHEVMODE	VADC0_G xCEFLAG	CEVy	VADC0_G xCEFCLR	CEVy	VADC0_G xCEVNP	CEVyINP
		Result Event y (y=0-7)	VADC0_G xREFLAG	REVy	VADC0_G xRCRy	SRGEN	VADC0_G xREFLAG	REVy	VADC0_G xREFCLR	REVy	VADC0_G xREVNP0	REVyNP
		Result Event y (y=8-15)	VADC0_G xREFLAG	REVy	VADC0_G xRCRy	SRGEN	VADC0_G xREFLAG	REVy	VADC0_G xREFCLR	REVy	VADC0_G xREVNP1	REVyNP
		Global Source Event	VADC0_G LOBEFLAG	SEVGLB	VADC0_BRSMR	ENSI	VADC0_G LOBEFLAG	SEVGLB	VADC0_G LOBEFLAG	SEVGLBCLR	VADC0_G LOBEVNP	REV0NP
		Global Result Event	VADC0_G LOBEFLAG	REVGLB	VADC0_G LOBERCR	SRGEN	VADC0_G LOBEFLAG	REVGLB	VADC0_G LOBEFLAG	REVGLBCLR	VADC0_G LOBEVNP	SEV0NP

表 5-6 中断源一览表 (续表)

IRQ	中断节点	中断源	状态标志		中断使能		置位标志		清除标志		节点指针	
			寄存器	位	寄存器	位	寄存器	位	寄存器	位	寄存器	位
21, 22, 23, 24	CCU40_SR[3:0]	Event 0 edge(s) information from event selector	CCU40_CC4yINTS	E0AS	CCU40_CC4yINTE	E0AE	CCU40_CC4ySWS	SE0A	CCU40_CC4ySWR	RE0A	CCU40_CC4ySRS	E0SR
		Event 1 edge(s) information from event selector	CCU40_CC4yINTS	E1AS	CCU40_CC4yINTE	E1AE	CCU40_CC4ySWS	SE1A	CCU40_CC4ySWR	RE1A	CCU40_CC4ySRS	E1SR
		Event 2 edge(s) information from event selector	CCU40_CC4yINTS	E2AS	CCU40_CC4yINTE	E2AE	CCU40_CC4ySWS	SE2A	CCU40_CC4ySWR	RE2A	CCU40_CC4ySRS	E2SR
		Period Match while counting up	CCU40_CC4yINTS	PMUS	CCU40_CC4yINTE	PME	CCU40_CC4ySWS	SPM	CCU40_CC4ySWR	RPM	CCU40_CC4ySRS	POSR
		Compare Match while counting up	CCU40_CC4yINTS	CMUS	CCU40_CC4yINTE	CMUE	CCU40_CC4ySWS	SCMU	CCU40_CC4ySWR	RCMU	CCU40_CC4ySRS	CMSR
		Compare Match while counting down	CCU40_CC4yINTS	CMDS	CCU40_CC4yINTE	CMDE	CCU40_CC4ySWS	SCMD	CCU40_CC4ySWR	RCMD	CCU40_CC4ySRS	CMSR
		One Match while counting down	CCU40_CC4yINTS	OMDS	CCU40_CC4yINTE	OME	CCU40_CC4ySWS	SOM	CCU40_CC4ySWR	ROM	CCU40_CC4ySRS	POSR
		Entering Trap State	CCU40_CC4yINTS	TRPF	CCU40_CC4yINTE	E2AE	CCU40_CC4ySWS	STRPF	CCU40_CC4ySWR	RTRPF	CCU40_CC4ySRS	E2SR

表 5-6 中断源一览表 (续表)

IRQ	中断节点	中断源	状态标志		中断使能		置位标志		清除标志		节点指针	
			寄存器	位	寄存器	位	寄存器	位	寄存器	位	寄存器	位
25, 26	CCU80_ SR[1:0]	Event 0 edge(s) information from event selector	CCU80_ CC8yINTS	E0AS	CCU80_ CC8yINTE	E0AE	CCU80_ CC8ySWS	SE0A	CCU80_ CC8ySWR	RE0A	CCU80_ CC8ySRS	E0SR
		Event 1 edge(s) information from event selector	CCU80_ CC8yINTS	E1AS	CCU80_ CC8yINTE	E1AE	CCU80_ CC8ySWS	SE1A	CCU80_ CC8ySWR	RE1A	CCU80_ CC8ySRS	E1SR
		Event 2 edge(s) information from event selector	CCU80_ CC8yINTS	E2AS	CCU80_ CC8yINTE	E2AE	CCU80_ CC8ySWS	SE2A	CCU80_ CC8ySWR	RE2A	CCU80_ CC8ySRS	E2SR
		Period Match while counting up	CCU80_ CC8yINTS	PMUS	CCU80_ CC8yINTE	PME	CCU80_ CC8ySWS	SPM	CCU80_ CC8ySWR	RPM	CCU80_ CC8ySRS	POSR
		Compare Match while counting up from compare channel 1	CCU80_ CC8yINTS	CMU1S	CCU80_ CC8yINTE	CMU1E	CCU80_ CC8ySWS	SCM1U	CCU80_ CC8ySWR	RCM1U	CCU80_ CC8ySRS	CM1SR
		Compare Match while counting down from compare channel 1	CCU80_ CC8yINTS	CMD1S	CCU80_ CC8yINTE	CMD1E	CCU80_ CC8ySWS	SCM1D	CCU80_ CC8ySWR	RCM1D	CCU80_ CC8ySRS	CM1SR
		Compare Match while counting up from compare channel 2	CCU80_ CC8yINTS	CMU2S	CCU80_ CC8yINTE	CMU2E	CCU80_ CC8ySWS	SCM2U	CCU80_ CC8ySWR	RCM2U	CCU80_ CC8ySRS	CM2SR
		Compare Match while counting down from compare channel 2	CCU80_ CC8yINTS	CMD2S	CCU80_ CC8yINTE	CMD2E	CCU80_ CC8ySWS	SCM2D	CCU80_ CC8ySWR	RCM2D	CCU80_ CC8ySRS	CM2SR
		One Match while counting down	CCU80_ CC8yINTS	OMDS	CCU80_ CC8yINTE	OME	CCU80_ CC8ySWS	SOM	CCU80_ CC8ySWR	ROM	CCU80_ CC8ySRS	POSR
		Entering Trap State	CCU80_ CC8yINTS	TRPF	CCU80_ CC8yINTE	E2AE	CCU80_ CC8ySWS	STRPF	CCU80_ CC8ySWR	RTRPF	CCU80_ CC8ySRS	E2SR

表 5-6　　中断源一览表 (续表)

IRQ	中断节点	中断源	状态标志		中断使能		置位标志		清除标志		节点指针	
			寄存器	位	寄存器	位	寄存器	位	寄存器	位	寄存器	位
27, 28	POSIF. SR[1:0]	Transition at Hall inputs	POSIF0_ PFLG	HIES	POSIF0_ PFLGE	EHIE	POSIF0_ SPFLG	SHIE	POSIF0_ RPFLG	RHIE	POSIF0_ PFLGE	HIESEL
		Occurrence of correct Hall event	POSIF0_ PFLG	CHES	POSIF0_ PFLGE	ECHE	POSIF0_ SPFLG	SCHE	POSIF0_ RPFLG	RCHE	POSIF0_ PFLGE	CHESEL
		Occurrence of wrong Hall event	POSIF0_ PFLG	WHES	POSIF0_ PFLGE	EWHE	POSIF0_ SPFLG	SWHE	POSIF0_ RPFLG	RWHE	POSIF0_ PFLGE	WHESEL
		Shadow transfer of MCM pattern	POSIF0_ PFLG	MSTS	POSIF0_ PFLGE	EMST	POSIF0_ SPFLG	SMST	POSIF0_ RPFLG	RMST	POSIF0_ PFLGE	MSTSEL
		Index event detection	POSIF0_ PFLG	INDXS	POSIF0_ PFLGE	EINDX	POSIF0_ SPFLG	SINDX	POSIF0_ RPFLG	RINDX	POSIF0_ PFLGE	INDSEL
		Phase detection error	POSIF0_ PFLG	ERRS	POSIF0_ PFLGE	EERR	POSIF0_ SPFLG	SERR	POSIF0_ RPFLG	RERR	POSIF0_ PFLGE	ERRSEL
		Quadrature clock generation	POSIF0_ PFLG	CNTS	POSIF0_ PFLGE	ECNT	POSIF0_ SPFLG	SCNT	POSIF0_ RPFLG	RCNT	POSIF0_ PFLGE	CNTSEL
		Period clock generation	POSIF0_ PFLG	PCLKS	POSIF0_ PFLGE	EPCLK	POSIF0_ SPFLG	SPCLK	POSIF0_ RPFLG	RPCLK	POSIF0_ PFLGE	PCLSEL
		Direction change	POSIF0_ PFLG	DIRS	POSIF0_ PFLGE	EDIR	POSIF0_ SPFLG	SDIR	POSIF0_ RPFLG	RDIR	POSIF0_ PFLGE	DIRSEL
29, 30	Reserved											
31	BCCU0. SR0	Trigger 0	BCCU0_ EVFR	T0F	BCCU0_ EVIER	T0IEN	BCCU0_ EVFSR	T0FS	BCCU0_ EVFCR	T0FC	-	-
		Trigger 1	BCCU0_ EVFR	T1F	BCCU0_ EVIER	T1IEN	BCCU0_ EVFSR	T1FS	BCCU0_ EVFCR	T1FC	-	-
		FIFO Full	BCCU0_ EVFR	FF	BCCU0_ EVIER	FIEN	BCCU0_ EVFSR	FFS	BCCU0_ EVFCR	FFC	-	-
		FIFO Empty	BCCU0_ EVFR	EF	BCCU0_ EVIER	EIEN	BCCU0_ EVFSR	EFS	BCCU0_ EVFCR	EFC	-	-
		Trap	BCCU0_ EVFR	TPF	BCCU0_ EVIER	TPIEN	BCCU0_ EVFSR	TPFS	BCCU0_ EVFCR	TPFC	-	-

1) Flash ECC double bit error has two status flags, each having its own clear bit. It is sufficient to use only one of the status flag and ignore the other.

6 事件请求单元 (ERU)

如在服务请求处理一章所述，XMC1300 使用事件请求单元 (ERU) 来支持服务请求处理的可编程互连。

6.1 特性

ERU 支持这些特性：

- 外部和内部服务请求的灵活处理
- 可编程为边沿和 / 或电平触发
- 每个通道可以有多个输入
- 可由多个输入组合触发
- 输入和输出门控

6.2 概述

事件请求单元 (ERU) 是一个通用的多输入事件检测和处理单元。

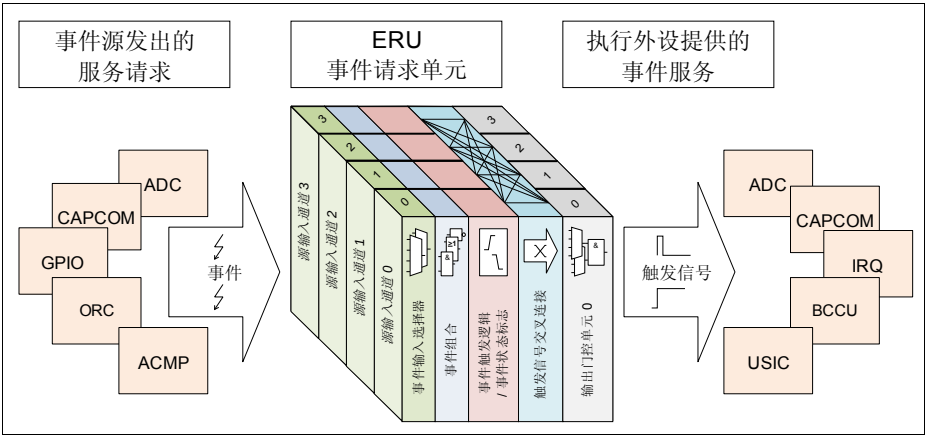


图 6-1 事件请求单元概览

每个 ERU 单元由下述功能块组成：

- 一个**事件请求选择 (ERS)** 单元。
  - 事件输入选择器允许从两个输入中选择一个。这两个输入中的每个输入都有一个具有 4 个可能信号的向量。
  - 事件组合允许两个输入信号进行逻辑组合，以产生一个公共触发信号。
- 每个输入通道有一个**事件触发逻辑 (ETL)**，允许定义导致触发事件的跃变方向（边沿选择，或由软件定义）并可存储事件状态。所选信号的输入电平在此被转换成事件。
- 触发信号**交叉连接矩阵**将事件和状态标志分配给输出通道。另外，还可使用来自其他模块的触发信号，并可将这些触发信号与本地触发信号进行逻辑组合。

- 一个**输出门控单元 (OGU)** 将触发事件和状态信息组合起来，并根据门控信号控制输出。

注：　一个输入上的一个事件可以影响多个输出，多个输入上的多个事件也可以组合起来去影响一个输出。

6.3　事件请求选择单元 (ERS)

每个输入通道 x (x = 0-3) 都有一个 ERSx 单元，它为相关联的 ETLx 单元选择输入。每个 ERSx 对两个信号 (Ax, Bx) 执行逻辑组合，为相关联的 ETLx 提供一个组合输出信号 ERSxO。输入 Ax 可以从输入向量 ERU_xA[3:0] 的 4 个选项中选择，并且可选择反相信号。输入 Bx (从向量 ERU_xB[3:0] 中选择) 的结构类似。

除了直接选择输入 Ax 或 Bx 或其反相信号外，还可以对所选择的两路输入进行逻辑组合，即逻辑与或者逻辑或。

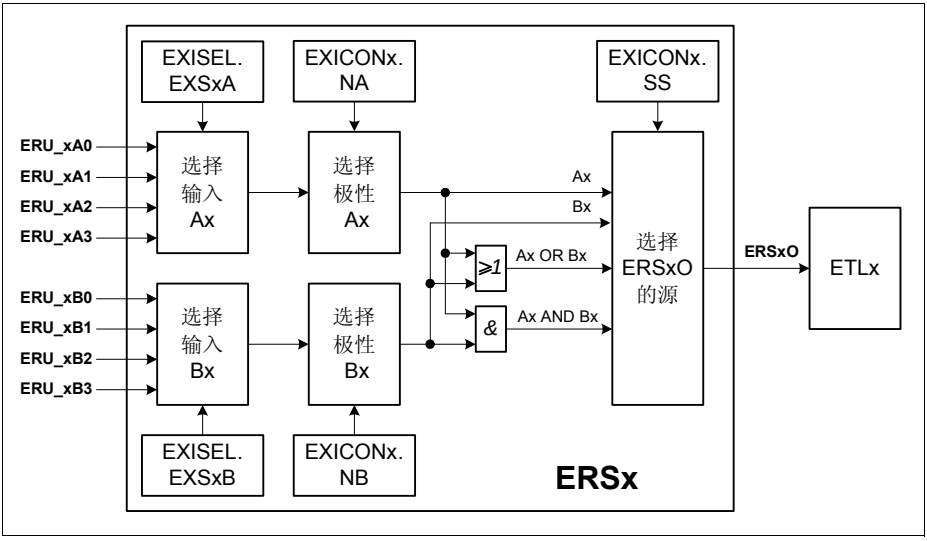


图 6-2　事件请求选择单元概览

ERS 单元由寄存器 **ERU0_EXISEL** (所有 4 个 ERSx 单元共用一个寄存器) 和寄存器 EXICONx (每个 ERSx 及相关联的 ETLx 单元使用一个寄存器，例如输入通道 0 的 **ERU0_EXICONx (x=0-3)**)。

6.4　事件触发逻辑 (ETLx)

每个输入通道x (x = 0-3)有一个事件触发逻辑ETLx。ETLx从输入ERSxO产生触发事件及相关的状态信息。每个 ETLx 都基于一个边沿检测块，可以单独使能对上升沿或下降沿的

检测。如果两个使能位都被置 1 (例如处理一个切换输入)，则两个边沿都会产生触发事件。

4 个 ETLx 单元中的每个单元都有一个相关联的 EXICONx 寄存器，它控制该 ETLx 的所有选项 (该寄存器还保持相关联的 ERSx 单元的控制位，例如 **ERU0_EXICONx (x=0-3)** 用于控制 ERS0 和 ETL0)。

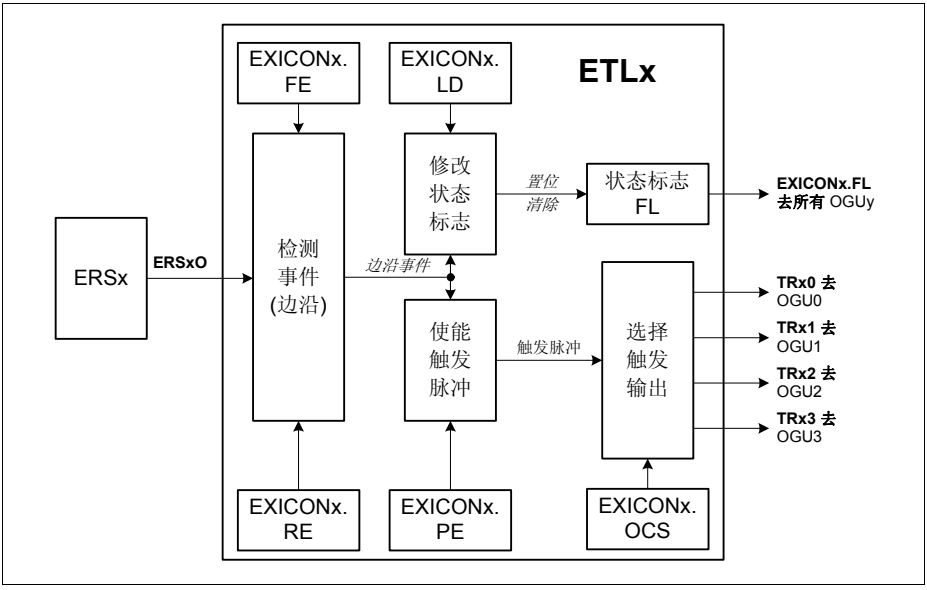


图 6-3　　事件触发逻辑概览

当检测到所选择的事件 (边沿) 时，状态标志 EXICONx.FL 被置位。该标志也可被软件修改 。该状态标志支持两种不同的工作模式。

它可被作为 “ 粘滞（sticky）” 标志使用，当检测到所期望的事件时由硬件置位，但必须由软件清除。在这种工作模式下，该标志指示事件已经发生，但并不指示输入的当前状态。

在第二种工作模式下，如果检测到 “ 相反的 ” 事件，该标志被自动清除。例如，如果只使能下降沿检测导致状态标志置位，则当检测到上升沿时该标志被自动清除。在这种模式下，状态标志可用于模式检测。因为在进行模式检测时，输入的当前状态很重要 （在该模式下，使能两个边沿检测没有意义）。

状态标志的输出被并行连接到位于后级的所有输出门控单元 (OGUy) (见**图 6-4**)，为**所有 OGUy 单元提供基于不同或相同状态标志的模式检测能力**。

除了修改状态标志之外，可以使能 (用位 EXICONx.PE) 并选择 ETLx 的一个触发脉冲输出，用于**触发一个 OGUy 单元的动作**。触发脉冲的目标 OGUy 由位域 EXICON.OCS 选择。

当检测到所选的边沿事件时，触发信号变成活动状态，与状态标志 EXICONx.FL 无关。

6.5 交叉连接矩阵

图 6-4 所示的连接矩阵将来自不同 ETLx 单元的触发信号 (TRxy) 和状态信号 (EXICONx.FL) 分发给 OGUy 单元。另外，它还接收外设触发信号，这些信号可与 ETLx 触发信号在 OGUy 单元中进行逻辑或组合。

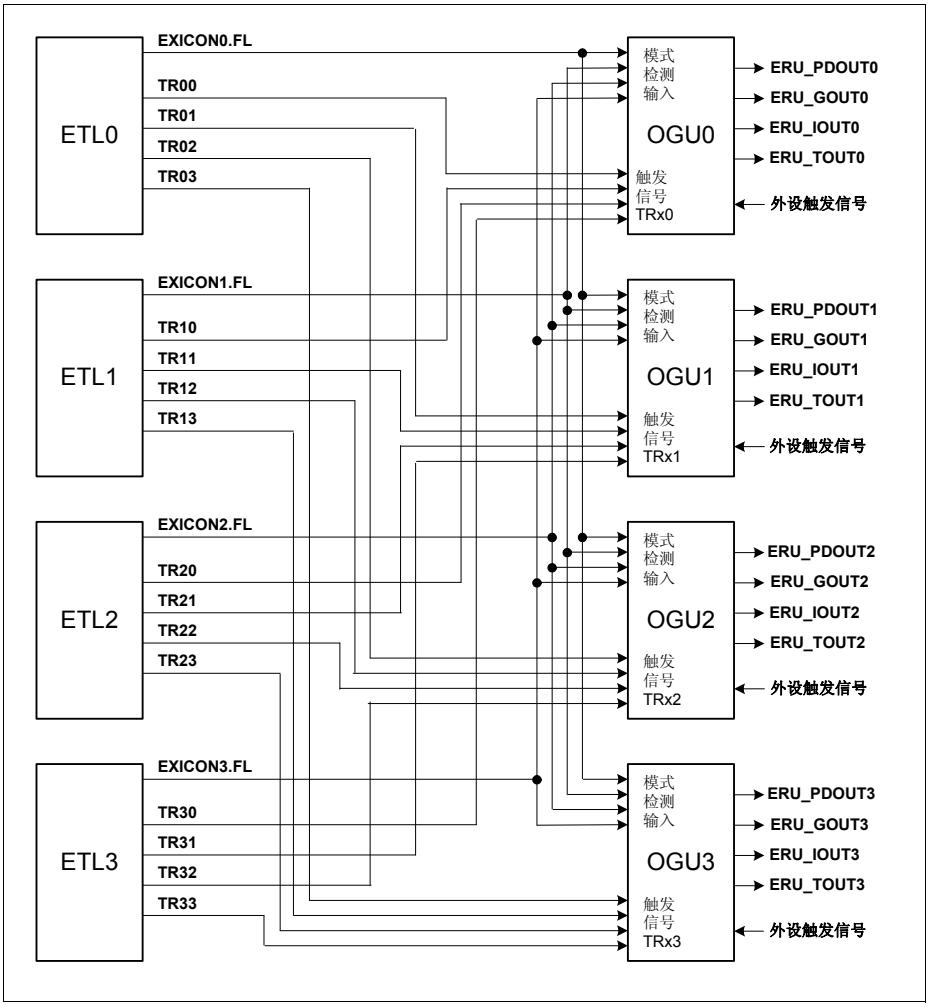


图 6-4 ERU 交叉连接矩阵

6.6 输出门控单元 (OGUy)

每个 OGUy (y = 0-3) 单元对来自输入通道的可用触发事件和状态标志进行组合，将结果提供给系统。**图 6-5** 示出了一个 OGUy 单元内的逻辑块。一个 OGUy 单元的所有功能都由其相关联的 EXOCONy 寄存器控制，例如 **ERU0_EXOCONx (x=0-3)** 控制 OGU0。一个 OGUy 单元的功能可被划分成两部分：

- **触发组合**：
  对来自输入通道的所有被使能并连接到 OGUy 的触发信号、一个选中的外设相关的触发事件和一个模式改变事件 （如果被使能）执行逻辑或运算。
- **模式检测**：
  输入通道的状态标志 EXICONx.FL 可被使能参加模式检测。当所有被使能的状态标志都置位时，检测到模式匹配。

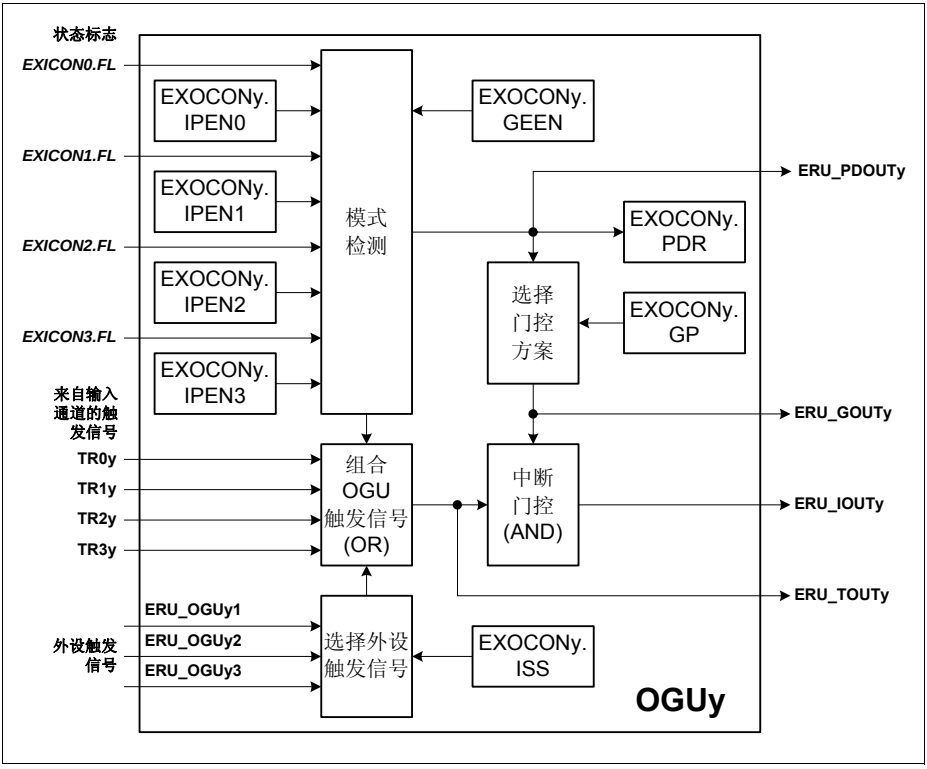


图 6-5 输出通道 y 的输出门控单元

每个 OGUy 单元产生 4 个输出信号分发给系统 (并非所有这些信号都必须使用)：

- **ERU_PDOUTy** 直接输出模式匹配信息，用作其他模块的门控信号 (模式匹配 = 1)。

- **ERU_GOUTy** 输出模式匹配或模式失配(模式匹配的反相信号)信息，或在软件控制下永久性输出 0 或 1，用作其它模块的门控信号。
- **ERU_TOUTy** 作为一个外设触发信号、一个模式检测结果改变事件或 ETLx 触发输出 TRxy 的组合输出，用于触发其它模块的动作。
- **ERU_IOUTy** 作为受控的触发输出(ERU_GOUTy和ERU_TOUTy的逻辑与)，用于触发服务请求 (例如，服务请求的产生可受门控信号控制，以允许在一个确定的时间窗口激活服务请求)。

触发组合

触发组合逻辑将不同的触发输入进行逻辑或组合，生成一个公共的触发信号 ERU_TOUTy。可能的触发输入有：

- **在输入通道**的每个 ETLx 单元内，触发输出 TRxy 都可被使能，并且触发事件可被连接到一个 OGUy 单元。
- 可在每个 OGUy 的三个**外设触发信号**中选择一个作为附加的触发源。这些外设触发信号由片内外设模块产生，例如捕获 / 比较单元或定时器单元。这种选择由位域 EXOCONy.ISS 完成。
- 在至少有一个**模式检测**输入被使能 (EXOCONy.IPENx) 并且检测到模式检测结果从模式匹配变为模式失配（或反之）的情况下，会产生一个触发事件，指示发生了一个模式检测结果事件 （如果由 ECOCONy.GEEN 使能）。

触发信号组合为给多个输入信号 （每个输入通道独立）或外设信号设置不同的触发条件提供了可能，也为将这些信号组合成单一输出提供了可能，例如用于产生一个服务请求或启动一次 ADC 转换。这种信号组合能力允许每个 OGU 从多个输入产生一个服务请求（多个请求源导致一个反应）。

这种选择由寄存器 **ERU0_EXOCONx (x=0-3)** 中的位域 ISS 定义。

模式检测

模式检测逻辑允许使用所有 ETLx 单元的状态标志的组合。每个状态标志都可以被单独包含在每个 OGUy 的模式检测逻辑中，也可以被排除在每个 OGUy 的模式检测逻辑之外，这可通过对控制位 EXOCONy.IPENx 编程来实现。模式检测块输出下述模式检测结果：

- **模式匹配** (EXOCONy.PDR = 1 且 ERU_PDOUTy = 1)：
  当包含在模式检测逻辑中的所有状态标志 FL 都为 1 时，指示模式匹配。
- **模式失配** (EXOCONy.PDR = 0 且 ERU_PDOUTy = 0)：
  当包含在模式检测逻辑中的状态标志 FL 至少有一个为 0 时，指示模式失配。

另外，如果模式检测结果从匹配变为失配或反之，模式检测逻辑能产生一个触发事件 (如果由 EXOCONy.GEEN = 1 使能)。模式检测结果改变事件与其他被使能的触发事件进行逻辑或运算后支持服务请求产生或触发其他模块功能 (例如 ADC)。当模式检测结果改变并且 EXOCONy.PDR 被更新后，会指示发生模式检测结果改变事件。

OGUy 中服务请求的产生基于触发信号 ERU_TOUTy，该信号可由模式检测结果 ERU_PDOUTy 控制（屏蔽）。这就允许在一个确定的时间窗口内自动和重复产生服务请求。该请求事件由触发信号组合逻辑产生，时间窗口信息（门控）由模式检测结果给定。

例如，在一个输入信号组合发生 (模式检测基于 ETLx 状态位) 期间，服务请求可以周期性地发出 (选择来自捕获 / 比较单元的外设触发输入) 。

可编程门控机制为适合应用需求提供了灵活性，并且允许在不同条件下产生服务请求 ERU_IOUTy：

- **模式匹配** (EXOCONy.GP = 10_B)：
  在模式检测结果为模式匹配期间，如果发生了触发事件，则会产生一次服务请求。
- **模式失配** (EXOCONy.GP = 11_B)：
  在模式检测结果为模式失配期间，如果发生了触发事件，则会产生一次服务请求。
- 与模式检测**无关** (EXOCONy.GP = 01_B)：
  在该模式，每次发生触发事件都会导致一次服务请求。模式检测输出可作为其他外设的门控信号使用，与触发组合无关 (ERU_TOUTy 和 ERU_PDOUTy 独立使用，发生触发事件时产生服务请求)。
- **无服务请求** (EXOCONy.GP = 00_B，默认设置)
  在该模式，发生触发事件并不会导致一次服务请求。模式检测输出可作为其他外设的门控信号使用，与触发组合无关 (ERU_TOUTy 和 ERU_PDOUTy 独立使用，发生触发事件时不产生服务请求)。

6.7 电源、复位和时钟

ERU 使用主时钟 MCLK 运行。只要 MCLK 保持运行，则 ERU 在所有工作模式下都消耗功率。

6.8 初始化和系统相关性

必须在源端和目的端一直使能服务请求。另外，还必须检查是否有必要将 ERU0 编程为处理和派送服务请求。

使能外设的 SRx 输出

- 必须有选择地使能外设的 SRx 输出。该过程取决于每个外设本身。详细信息请参阅相应外设所在章的 “ 服务请求产生 ” 一节。
- 可选择将 ERU0 编程为处理和派送该请求。

使能外部请求

- 所选端口必须被编程为输入。
- ERU0 必须被编程为处理和派送外部请求。

注： 外部服务请求输入的个数可能受所用封装的限制。

6.9 寄存器

表 6-1 寄存器地址空间

模块	基地址	结束地址	备注
ERU0	$4001\ 0600_H$	$4001\ 06FF_H$	

寄存器概览

绝对寄存器地址由下面的加法计算：

模块基地址 + 偏移地址

表 6-2 寄存器一览表

简称	描述	偏移地址	访问方式		描述见
			读	写	
EXISEL	ERU 外部输入控制选择	0000_H	U, PV	U, PV	页 6-8
EXICON0	ERU 外部输入控制选择	0010_H	U, PV	U, PV	页 6-10
EXICON1	ERU 外部输入控制选择	0014_H	U, PV	U, PV	页 6-10
EXICON2	ERU 外部输入控制选择	0018_H	U, PV	U, PV	页 6-10
EXICON3	ERU 外部输入控制选择	$001C_H$	U, PV	U, PV	页 6-10
EXOCON0	ERU 输出控制寄存器	0020_H	U, PV	U, PV	页 6-12
EXOCON1	ERU 输出控制寄存器	0024_H	U, PV	U, PV	页 6-12
EXOCON2	ERU 输出控制寄存器	0028_H	U, PV	U, PV	页 6-12
EXOCON3	ERU 输出控制寄存器	$002C_H$	U, PV	U, PV	页 6-12

6.9.1 ERU 寄存器

ERU0_EXISEL
事件输入选择 **(00_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXS3B		EXS3A		EXS2B		EXS2A		EXS1B		EXS1A		EXS0B		EXS0A	
rw		rw		rw		rw		rw		rw		rw		rw	

域	位	类型	描述
EXS0A	[1:0]	rw	**A0 事件源 (ERS0) 选择** 该位域定义为 A0 选择哪个输入。 00_B 选择输入 ERU_0A0 01_B 选择输入 ERU_0A1 10_B 选择输入 ERU_0A2 11_B 选择输入 ERU_0A3
EXS0B	[3:2]	rw	**B0 事件源 (ERS0) 选择** 该位域定义为 B0 选择哪个输入。 00_B 选择输入 ERU_0B0 01_B 选择输入 ERU_0B1 10_B 选择输入 ERU_0B2 11_B 选择输入 ERU_0B3
EXS1A	[5:4]	rw	**A1 事件源 (ERS1) 选择** 该位域定义为 A1 选择哪个输入。 00_B 选择输入 ERU_1A0 01_B 选择输入 ERU_1A1 10_B 选择输入 ERU_1A2 11_B 选择输入 ERU_1A3
EXS1B	[7:6]	rw	**B1 事件源 (ERS1) 选择** 该位域定义为 B1 选择哪个输入。 00_B 选择输入 ERU_1B0 01_B 选择输入 ERU_1B1 10_B 选择输入 ERU_1B2 11_B 选择输入 ERU_1B3
EXS2A	[9:8]	rw	**A2 事件源 (ERS2) 选择** 该位域定义为 A2 选择哪个输入。 00_B 选择输入 ERU_2A0 01_B 选择输入 ERU_2A1 10_B 选择输入 ERU_2A2 11_B 选择输入 ERU_2A3
EXS2B	[11:10]	rw	**B2 事件源 (ERS2) 选择** 该位域定义为 B2 选择哪个输入。 00_B 选择输入 ERU_2B0 01_B 选择输入 ERU_2B1 10_B 选择输入 ERU_2B2 11_B 选择输入 ERU_2B3

域	位	类型	描述
EXS3A	[13:12]	rw	**A3 事件源 (ERS3) 选择** 该位域定义为 A3 选择哪个输入。 00_B 选择输入 ERU_3A0 01_B 选择输入 ERU_3A1 10_B 选择输入 ERU_3A2 11_B 选择输入 ERU_3A3
EXS3B	[15:14]	rw	**B3 事件源 (ERS3) 选择** 该位域定义为 B3 选择哪个输入。 00_B 选择输入 ERU_3B0 01_B 选择输入 ERU_3B1 10_B 选择输入 ERU_3B2 11_B 选择输入 ERU_3B3
0	[31:16]	r	**保留** 读出值为 0；应写入 0。

ERU0_EXICONx (x=0-3)
事件输入控制 x **(10_H + 4*x)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**NB**	**NA**	**SS**		**FL**	**OCS**			**FE**	**RE**	**LD**	**PE**
r				rw	rw	rw		rwh	rw			rw	rw	rw	rw

域	位	类型	描述
PE	0	rw	**ETLx 输出触发脉冲使能** 当检测到所选择的边沿(状态标志FL的置位条件)时，若该位使能则会在 TRxy 产生一个输出触发脉冲。 0_B 禁止产生触发脉冲 1_B 允许产生触发脉冲

域	位	类型	描述
LD	1	rw	**ETLx 状态标志的重建电平检测** 该位选择是将FL用作“粘滞”位，还是用FL重建电平检测结果。 0_B 状态标志FL不被硬件清零，且被作为“粘滞”位使用。该标志一旦置位，便不受任何边沿的影响，直到其被软件清零。 1_B 状态标志 FL 对所期望的事件重建一次电平检测。该标志在发生一次上升沿 （如果 RE = 1）或下降沿 （如果 FE = 1）时自动变成置位状态。该标志在发生一次上升沿 （如果 RE = 0）或下降沿 （如果 FE = 0）时被自动清零。
RE	2	rw	**ETLx 上升沿检测使能** 该位使能 / 禁止上升沿事件作为边沿事件，作为状态标志 FL 的置位条件，或作为 TRxy 的可能触发脉冲。 0_B 上升沿不被视为边沿事件 1_B 上升沿被视为边沿事件
FE	3	rw	**ETLx 下降沿检测使能** 该位使能 / 禁止下降沿事件作为边沿事件，作为状态标志 FL 的置位条件，或作为 TRxy 的可能触发脉冲。 0_B 下降沿不被视为边沿事件 1_B 下降沿被视为边沿事件
OCS	[6:4]	rw	**ETLx 输出触发脉冲的输出通道选择** 该位域为一个被使能的触发脉冲 TRxy 定义目标输出通道 OGUy。 000_B 触发脉冲被发送到 OGU0 001_B 触发脉冲被发送到 OGU1 010_B 触发脉冲被发送到 OGU2 011_B 触发脉冲被发送到 OGU3 其他：保留，不能使用这些组合。
FL	7	rwh	**ETLx 的状态标志** 该位是状态标志，由边沿检测结果置位或清零。 0_B 未检测到被使能的边沿事件。 1_B 检测到被使能的边沿事件。

域	位	类型	描述
SS	[9:8]	rw	**ERSx 输入源选择** 该位域定义使用哪种逻辑组合作为 ERSxO。 00_B 输入 A，无额外组合 01_B 输入 B，无额外组合 10_B 输入 A OR 输入 B 11_B 输入 A AND 输入 B
NA	10	rw	**ERSx 输入 A 的反相选择** 该位选择输入 A 的极性。 0_B 输入 A 直接使用 1_B 输入 A 被反相
NB	11	rw	**IERSx 输入 B 的反相选择** 该位选择输入 B 的极性。 0_B 输入 B 直接使用 1_B 输入 B 被反相
0	[31:12]	r	**保留** 读出值为 0；应写入 0。

ERU0_EXOCONx (x=0-3)
事件输出触发控制 x **(20_H + 4*x)** **复位值 : 0000 0008$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IPEN 3	**IPEN 2**	**IPEN 1**	**IPEN 0**	**0**						**GP**		**PDR**	**GEE N**	**ISS**	
rw	rw	rw	rw	r						rw		rh	rw	rw	

域	位	类型	描述
ISS	[1:0]	rw	**内部触发源选择** 该位域定义哪个输入被选择为 OGUy 的外设触发输入。 00_B 外设触发功能被禁止 01_B 选择输入 ERU_OGUy1 10_B 选择输入 ERU_OGUy2 11_B 选择输入 ERU_OGUy3

域	位	类型	描述
GEEN	2	rw	**门控事件使能** 当模式检测结果从匹配变为失配或反之，位 GEEN 使能触发事件产生。 0_B 事件检测被禁止 1_B 事件检测被使能
PDR	3	rh	**模式检测结果标志** 该位指示模式检测结果。 0_B 检测到模式失配 1_B 检测到模式匹配
GP	[5:4]	rw	**模式检测结果的门控选择** 该位域定义服务请求产生的门控机制 (OGU 输出 ERU_PDOUTy 与 ERU_GOUTy 之间的关系)。 00_B ERU_GOUTy 总是被禁止，并且 ERU_IOUTy 不能被激活。 01_B ERU_GOUTy 总是被使能，并且 ERU_IOUTy 在 ERU_TOUTy 每次激活时变为激活状态。 10_B ERU_GOUTy 与 ERU_PDOUTy 相同，并且在检测到所期望的模式 (模式匹配 PDR = 1) 期间，ERU_IOUTy 在 ERU_TOUTy 激活时变为激活状态。 11_B ERU_GOUTy 与 ERU_PDOUTy 反相，并且在未检测到所期望的模式 (模式失配 PDR = 0) 期间，ERU_IOUTy 在 ERU_TOUTy 激活时变为激活状态。
IPENx (x = 0-3)	12+x	rw	**ETLx 模式检测使能** 位 IPENx 定义 ETLx 的触发事件状态标志 EXICONx.FL 是否参与 OGUy 的模式检测。 0_B 标志 EXICONx.FL 不参与模式检测。 1_B 标志 EXICONx.FL 参与模式检测。
0	[31:16], [11:6]	r	**保留** 读出值为 0 ； 应写入 0。

6.10 互连

本节描述 ERU0 模块在 XMC1300 系统内部是如何连接的。

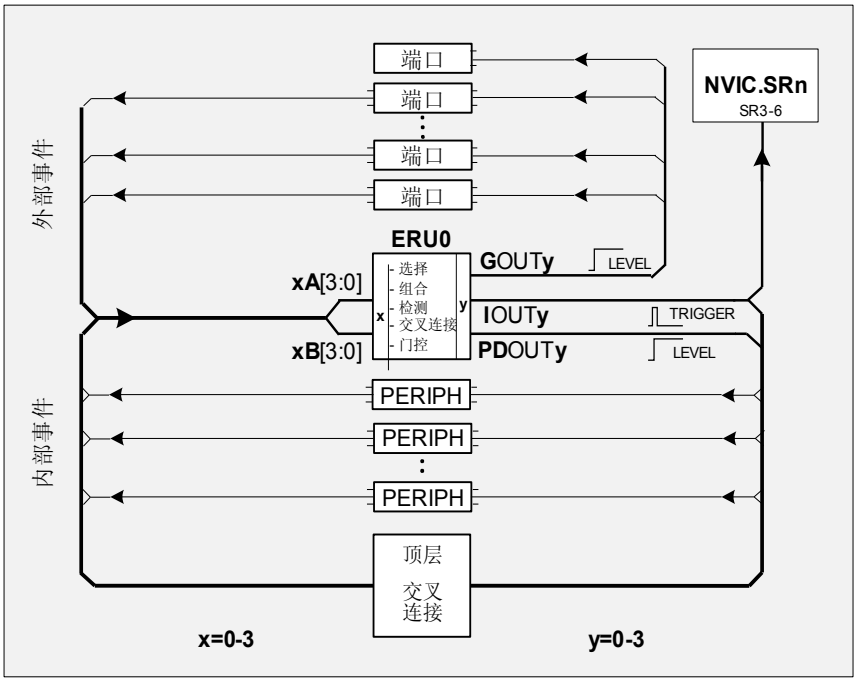


图 6-6 ERU 互连概览

6.10.1 ERU0 连接

下表列出了 ERU0 的连接情况。

表 6-3 ERU0 引脚连接

全局输入 / 输出	连接到	I/O	描述
ERU0.0A0	ACMP0.OUT	I	
ERU0.0A1	P2.4	I	
ERU0.0A2	ORC2.OUT	I	
ERU0.0A3	VADC0.G0BFLOUT0	I	来自 ADC 边界标志
ERU0.0B0	P2.0	I	
ERU0.0B1	P2.2	I	
ERU0.0B2	ORC0.OUT	I	

表 6-3 ERU0 引脚连接

全局输入 / 输出	连接到	I/O	描述
ERU0.0B3	VADC0.G1BFLOUT0	I	来自 ADC 边界标志
ERU0.1A0	ACMP1.OUT	I	
ERU0.1A1	P2.5	I	
ERU0.1A2	ORC3.OUT	I	
ERU0.1A3	VADC0.G0BFLOUT1	I	来自 ADC 边界标志
ERU0.1B0	P2.1	I	
ERU0.1B1	P2.3	I	
ERU0.1B2	ORC1.OUT	I	
ERU0.1B3	VADC0.G1BFLOUT1	I	来自 ADC 边界标志
ERU0.2A0	ACMP2.OUT	I	
ERU0.2A1	P2.6	I	
ERU0.2A2	ORC4.OUT	I	
ERU0.2A3	VADC0.G0BFLOUT2	I	来自 ADC 边界标志
ERU0.2B0	P2.10	I	
ERU0.2B1	P2.11	I	
ERU0.2B2	ORC2.OUT	I	
ERU0.2B3	VADC0.G1BFLOUT2	I	来自 ADC 边界标志
ERU0.3A0	ORC7.OUT	I	
ERU0.3A1	P2.7	I	
ERU0.3A2	ORC5.OUT	I	
ERU0.3A3	VADC0.G0BFLOUT3	I	来自 ADC 边界标志
ERU0.3B0	P2.9	I	
ERU0.3B1	P2.8	I	
ERU0.3B2	ORC6.OUT	I	
ERU0.3B3	VADC0.G1BFLOUT3	I	来自 ADC 边界标志
ERU0.OGU01	CCU40.SR0	I	
ERU0.OGU02	VADC0.C0SR2	I	
ERU0.OGU03	CCU80.SR2	I	
ERU0.OGU11	CCU40.SR1	I	
ERU0.OGU12	VADC0.C0SR2	I	

表 6-3 ERU0 引脚连接

全局输入 / 输出	连接到	I/O	描述
ERU0.OGU13	CCU80.SR2	I	
ERU0.OGU21	CCU40.SR2	I	
ERU0.OGU22	VADC0.C0SR3	I	
ERU0.OGU23	CCU80.SR3	I	
ERU0.OGU31	CCU40.SR3	I	
ERU0.OGU32	VADC0.C0SR3	I	
ERU0.OGU33	CCU80.SR3	I	
ERU0.PDOUT0	VADC0.BGREQGTO VADC0.G0REQGTO VADC0.G1REQGTO CCU40.IN1D CCU40.IN0J CCU80.IN0F CCU80.IN1L CCU80.IN2L CCU80.IN3L POSIF0.IN0D USIC0_CH1.HWIN0 P0.0 P2.11	O	
ERU0.GOUT0	P0.0 P2.11	O	
ERU0.TOUT0	未连接	O	
ERU0.IOUT0	NVIC.ERU0.SR0 SCU.RTC_extclk VADC0.BGREQTRM VADC0.G0REQTRM VADC0.G1REQTRM CCU40.CLKB CCU40.IN0K CCU80.CLKB CCU80.IN0G POSIF0.EWHEB BCCU0.TRAPINE	O	

表 6-3 ERU0 引脚连接

全局输入 / 输出	连接到	I/O	描述
ERU0.PDOUT1	VADC0.BGREQGTP VADC0.G0REQGTP VADC0.G1REQGTP CCU40.IN0D CCU40.IN1J CCU80.IN0L CCU80.IN1F POSIF0.IN1D USIC0_CH1.HWIN2 P0.1 P2.10	O	
ERU0.GOUT1	P0.1 P2.10	O	
ERU0.TOUT1	未连接	O	
ERU0.IOUT1	NVIC.ERU0.SR1 VADC0.BGREQTRN VADC0.G0REQTRN VADC0.G1REQTRN CCU40.CLKC CCU40.IN1K CCU80.CLKC CCU80.IN1G BCCU0.TRAPINF	O	
ERU0.PDOUT2	VADC0.BGREQGTK VADC0.G0REQGTK VADC0.G1REQGTK CCU40.IN3D CCU40.IN2J CCU80.IN2F POSIF0.IN2D P0.2 P2.1	O	
ERU0.GOUT2	P0.2 P2.1	O	
ERU0.TOUT2	未连接	O	

表 6-3 ERU0 引脚连接

全局输入 / 输出	连接到	I/O	描述
ERU0.IOUT2	NVIC.ERU0.SR2 VADC0.BGREQTRG VADC0.G0REQTRG VADC0.G1REQTRG CCU40.IN2K CCU80.IN2G POSIF0.MSETF BCCU0.TRAPING	O	
ERU0.PDOUT3	VADC0.BGREQGTL VADC0.G0REQGTL VADC0.G1REQGTL CCU40.IN3J CCU40.IN2D CCU80.IN3F P0.3 P2.0	O	
ERU0.GOUT3	P0.3 P2.0	O	
ERU0.TOUT3	未连接	O	
ERU0.IOUT3	NVIC.ERU0.SR3 VADC0.BGREQTRH VADC0.G0REQTRH VADC0.G1REQTRH CCU40.IN3K CCU80.IN3G POSIF0.EWHEC POSIF0.MSETG BCCU0.TRAPINH	O	

7 MATH 协处理器 （MATH）

7.1 概述

MATH 协处理器模块包括两个独立的子单元，来支持 CPU 完成数学密集型运算：一个是能完成有符号和无符号 32 位除法运算的除法器单元 (DIV)，另一个是能进行三角函数、线性函数或双曲函数运算的 CORDIC (坐标旋转数字计算机) 协处理器。

7.1.1 特性

MATH 协处理器有以下特性：

- 有操作数预处理和结果后处理能力的除法功能
- 能计算三角函数、双曲函数和线性函数的 CORDIC 协处理器
- 支持内核时钟与接口时钟 2:1 的比率，使执行速度更快
- 支持除法器单元和 CORDIC 协处理器单元之间的结果链接

7.1.2 框图

图 7-1 给出了 MATH 协处理单元的框图。

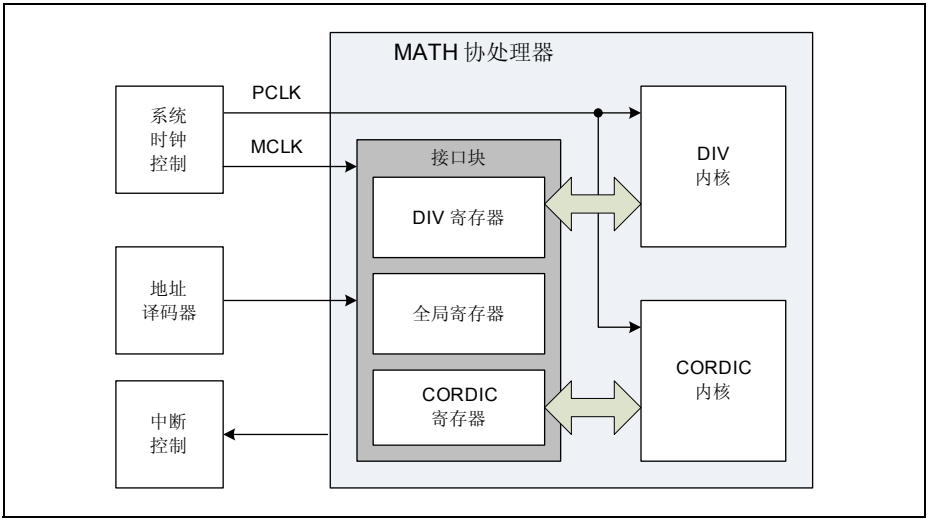


图 7-1 MATH 协处理器框图

7.2 除法器单元 (DIV)

7.2.1 特性

DIV 支持以下特性：

- 使用 35 个内核时钟周期的有符号 / 无符号 32 位除法操作
- 操作数预处理，可配置移位位数：
  - 被除数左移
  - 除数右移
- 结果后处理，可配置移位位数和移位方向

7.2.2 除法操作

DIV 支持舍尾除法运算，这也是 ISO C99 标准，并且是现代处理器的普遍选择。除法和舍尾除法的取模功能有下述关系：

If q = D div d

and r = D mod d

then D = q * d + r

and | r | < | d |

这里 “D” 是被除数，“d” 是除数，“q” 是商，“r” 是余数。舍尾除法把商向零方向舍入为整数，并且余数的符号总是与被除数的符号一致，即 sign (r) = sign (D)。

要使用 DIV 执行一次除法运算，首先要按下述操作配置该除法：

- 通过 DIVCON.USIGN 位选择有符号或者无符号除法
- 通过 DIVCON.STMODE 位选择启动模式

然后向 DVD 和 DVS 寄存器写入被除数和除数的值。根据启动模式不同，除法运算可以通过写 DVS 寄存器启动，也可以通过将启动位 DIVCON.ST 置 1 来启动。如果使用 ST 位启动，则该位在下一个内核时钟周期被自动清零 。除法运算启动后忙标志 DIVST.BSY 被置 1。

除法运算总是耗时 35 个内核时钟周期，一旦完成，即可在 QUOT 和 RMD 寄存器中得到可用的商和余数，并且 BSY 标志会被清零。计算结束事件将除法器事件标志 EVFR.DIVEOC 置 1，并且向 NVIC 触发一次中断请求 （如果通过 EVIER.DIVEOCIEN 位使能了该中断）。该标志只能通过软件对 EVFCR.DIVEOCC 位的写操作来清零。

图 7-2 给出了一次除法运算的时序图。

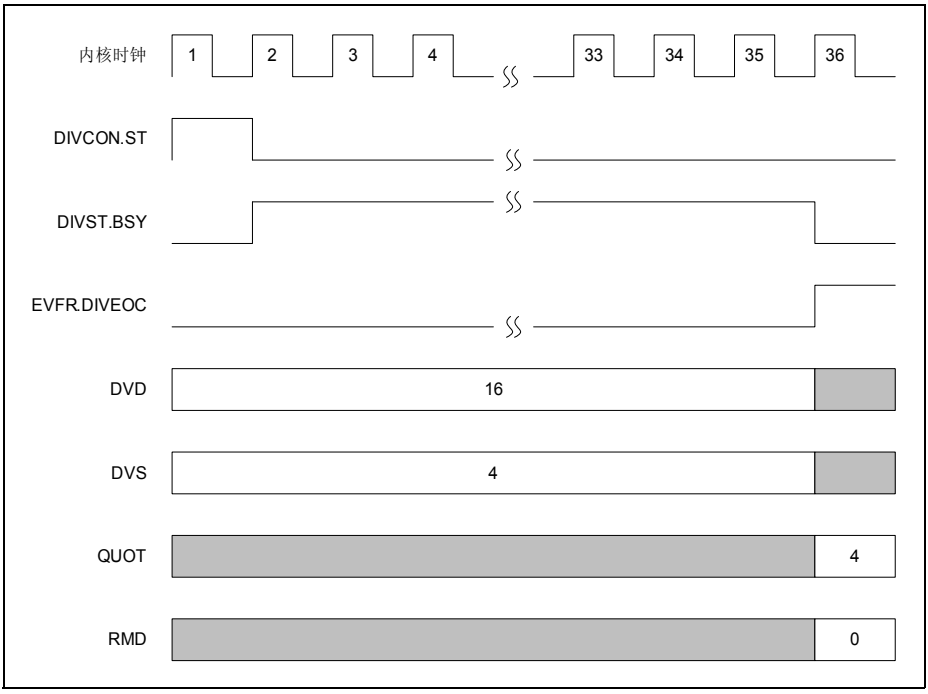


图 7-2　　除法运算的时序图

注：　在 BSY=1 期间，读 QUOT 和 RMD 寄存器会导致 DIV 在总线上插入等待状态，直到当前计算结束 (BSY=0)。这样就会确保对结果寄存器 QUOT 或者 RMD 的任何读访问都能返回一个有效的结果。然而，由于总线可能被锁定一定数量的内核时钟周期，中断延迟将会增加。

7.2.2.1　启动模式选择

启动除法运算的条件通过 DIVCON.STMODE 位选择：

- 当 STMODE = 0 时，通过对 DVS 寄存器的写操作来启动除法运算。在这种情况下，不需要软件对 ST 位执行置 1 操作。
- 当 STMODE = 1 时，通过将 ST 位置 1 来启动除法运算。

对于这两种启动方式，必须确保在启动运算之前 DIV 不执行任何计算，并且 DIVST.BSY 标志位为 0。否则，启动请求会被放弃，尽管 DVS 仍然会被新写入的值更新。在 BSY = 1 期间，对 DIVCON 寄存器的写访问被忽略。因此，建议应用程序在通过写 ST 位或写 DVS 寄存器的操作启动除法运算之前，查询 BSY 标志。

7.2.2.2 被零除错误

如果在 DVS 寄存器中的值等于 0 的情况下启动了除法运算，则 EVFR.DIVERR 标志会被置 1。同时，如果通过 EVIER.DIVERRIEN 使能了中断，将会向 NVIC 产生一个中断请求。除法运算仍会继续正常进行，并在 35 个内核时钟周期内完成，如图 7-3 所示。

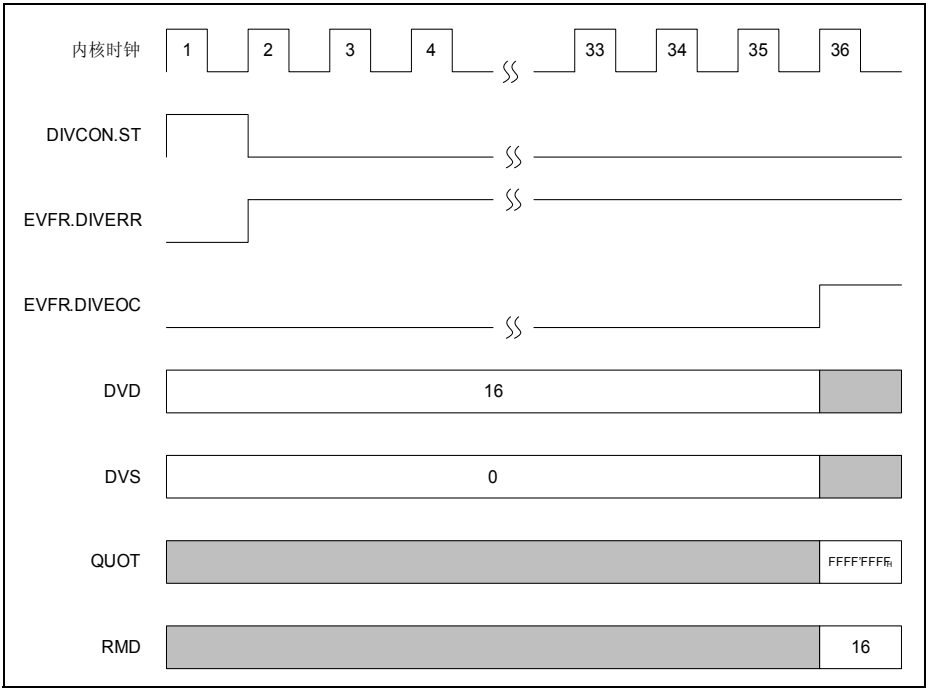


图 7-3 被零除错误时序图

如果 DIVCON.USIGN 为 0（有符号数运算），并且 DVD 为负值，则 QUOT 寄存器包含全 0，否则包含全 1。在任何一种情况下， RMD 寄存器都将包含被除数的值。只能通过对 EVFCR.DIVERRC 位的软件写操作来清除 EVFR.DIVERR 标志。

注： 如果结果后处理功能（见 7.2.3 节）被使能，商值（全 0 或全 1）仍会根据 DIVCON.QSCNT 位的值而移位。

DIV不检测溢出错误。用户必须确保带结果后处理和不带结果后处理的除法运算的结果位于 DIV 的范围之内。

7.2.3 操作数预处理 / 结果后处理

DIV 支持下述操作数预处理和结果后处理操作：

- 除法开始前被除数左移

- 除法开始前除数右移
- 除法结束后商值左移或右移

移位的位数由 DIVCON 寄存器中相应的 5 位移位计数位域决定。此外，对商值而言，移位方向由 DIVCON.QSDIR 位决定。

所有的移位都是算数移位。这意味着，如果左移，则在 LSB 位插入零，如果右移，零 (在无符号模式) 或者是符号位 (在有符号模式) 将会被插入到 MSB 位。

如果通过向移位计数位域写入一个非零值来使能了移位操作，移位操作将会在除法运算的开始和结束时被触发，并且不会消耗任何额外的内核时钟周期。DIV 事件标志在 35 个执行周期结束时产生。

在操作数预处理期间，只有 DVD 和 DVS 寄存器的输出被执行移位操作，而不是 DVD 和 DVS 寄存器本身。因此，这两个寄存器的内容保持不变。

7.3 CORDIC 协处理器

CORDIC 协处理器能计算三角函数、线性函数、双曲函数及相关函数。

7.3.1 概述

本文档描述 XMC1000 产品家族的 CORDIC 协处理器的特性。介绍 CORDIC 算法的理论背景不是本文档的目的。

对于三角函数、线性函数、双曲函数和相关函数的计算而言， CORDIC 算法是一种有用的收敛方法。该算法不仅可以在欧几里得平面执行矢量旋转，而且在线性平面和双曲平面也可以执行矢量旋转。

CORDIC 算法是一个迭代过程，截断误差是这个过程的固有性质。CORDIC 协处理器每次计算使用 27 次迭代，并且使用至少为 32 位的内核数据宽度，因而可获得较高的精度。使用该算法的主要优点在于比软件有更快的计算速度，并可获得高精度。

通用的 CORDIC 算法使用以下公式，因子 m 控制矢量旋转并选择三角函数、线性函数和双曲函数的角度：$x_{i+1} = x_i - m \cdot d_i \cdot y_i \cdot 2^{-i}$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i} \quad (7.1)$$

$$z_{i+1} = z_i - d_i \cdot e_i \quad (7.2)$$

其中：

m = 1 三角函数 (基本 CORDIC)， $e_i = \text{atan}(2^{-i})$

m = 0 线性函数， $e_i = 2^{-i}$

m = -1 双曲函数， $e_i = \text{atanh}(2^{-i})$

为清晰起见，本文档使用以下术语表示 CORDIC 数据：

- 结果数据：在 CORDIC 计算结束时 （BSY 位不再有效）的最终结果数据。
- 计算数据：来自 CORDIC 迭代的中间数据或上一次数据结果。

- 初始数据：第一次 CORDIC 迭代中使用的数据，通常是用户的初始化数据。

7.3.1.1 特性

CORDIC 协处理器的关键特性如下：

- 运算模式
  - 支持所有的CORDIC运算模式，用于解决圆函数（三角函数）、线性(乘-加,除-加)函数和双曲函数问题。
  - 集成的查找表 (LUT) 可用于所有的运算模式
- 圆向量模式：支持满范围 $[-2^{23},(2^{23}-1)]$ 内的初始 X 和 Y 数据值，来计算角度和幅值。
- 圆旋转模式：支持满范围 $[-2^{23},(2^{23}-1)]$ 内的初始 Z 数据值，代表范围 $[-\pi,((2^{23}-1)/2^{23})\pi]$ 内的角度，来求解三角学问题。
- 24 位可访问的数据宽度
  - 32 位内核数据宽度外加 2 个符号扩展位以及 X 和 Y 各一位溢出位
  - 29 位内核数据宽度外加 1 个符号扩展位和 Z 的 1 位溢出位
  - 一个 KEEP 位，用于在内核寄存器中为进行一次新计算保持最后一次结果值
- 每次运算进行27次迭代；从启动位 (ST)置1到计算结束标志置1需要62个内核时钟周期或更少，这不包括写和读数据字节消耗的时间。
- 补码数据处理
  - 唯一的例外：具有用户可选项的 X 结果数据 （对于无符号结果）。
- 接受的 X 和 Y 数据一般为整数或有理数； X 和 Y 必须是相同的数据格式。
- 截断误差
  - 由于低有效位的截断效应，一次 CORDIC 计算的结果可能返回一个近似值。
  - CORDIC 计算出的结果数据精度高，尤其是在圆模式下。
- 中断
  - 在一次计算完成时
  - 中断使能和中断标志

表 7-1 CORDIC 应用

用例	应用
角度计算	EPS, 电机控制
派克变换	电机控制
Y +XZ	数字滤波 , PI 控制回路

7.3.1.2 框图

图 7-4 给出了 CORDIC 协处理器内核的框图 。

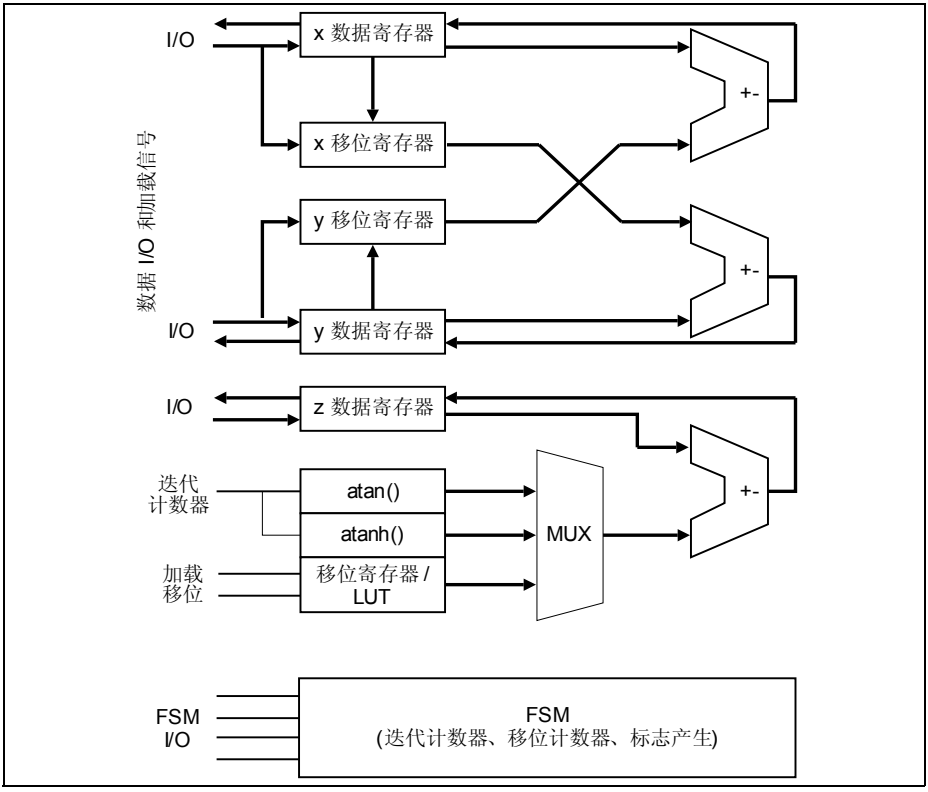


图 7-4 CORDIC 框图

7.3.2 功能概述

以下各小节描述 CORDIC 协处理器的功能。

7.3.2.1 服务请求处理的操作

不论在旋转模式还是在向量模式，服务请求处理都可以用来完成园函数（三角函数）、线性 (乘 – 加，除 – 加) 函数或双曲函数运算。用软件通过 **CON** 控制寄存器来选择模式。

内核数据寄存器的初始化是通过将寄存器 **STATC** 中相应的 **STATC**.KEEPx 位清零来使能的。如果 **CON**.ST_MODE = 1，向 **CON**.ST 位写 1 即启动一次新的计算。否则，在 **CON**.ST_MODE = 0 的默认情况下，在对寄存器 **CORDX** 进行写访问后会启动一次新的计算。每次计算包含固定的 27 次迭代。当一次运算正在进行时， **STATC**.BSY 位被置 1，表示处于忙状态。在一次计算结束时， **STATC**.BSY 位被硬件清零。

启动一次 CORDIC 计算的第一步 (如果相应的 KEEP 位没有被置 1) 是将初始数据从数据寄存器 CORDx 加载到内核数据寄存器。在计算期间，内核数据寄存器总是保持最后的中间数据。在计算结束时，结果数据被加载到结果寄存器 CORRx。

数据寄存器 CORDx 的功能有如映射寄存器，可以在不影响正在执行的计算的情况下被写入。只有当 **CON**.ST 位被有效置 1 时，或者是在 **CON**.ST_MODE = 0 情况下，对 X 数据寄存器 **CORDX** (如果相应的数据 KEEP 位没有被置 1) 的写访问之后，数据才被传送到内核数据寄存器。在计算结束时 (BSY 位不再有效)，可以从结果寄存器 CORRx 读取结果数据。需要在启动下一次计算之前读取结果数据。在 **STATC**.BSY 被置 1 期间 (内核仍在进行一次计算)，对结果寄存器 CORRx 的任何读访问都会导致内核发出一个总线等待，直到 **STATC**.BSY 被复位。这样可以确保对结果寄存器 CORRx 的任何一次读访问都会返回一个有效结果。

在每次计算结束时， **STATC**.BSY 返回到 0，指示 CORDIC 运算结束的 EVFR.CDEOC 位被置 1。如果通过 EVIER.CDEOCIEN = 1 使能了中断，则中断请求信号被激活。EVFR.CDEOC 标志必须由软件清零，这可通过设置 EVFCR.CDEOCC = 1 来实现。X、Y 和 Z 中的结果数据在内部被检查，在发生数据溢出的情况下， EVFR.CDERR 位被置 1。如果通过 EVIER.CDEOCIEN = 1 使能了中断，则中断请求信号将被激活。EVFR.CDERR 标志必须由软件清零，通过设置 EVFCR.CDERRC = 1 来实现。

在一次运算正在进行期间 (如果 **CON**.ST_MODE = 1)，在 **STATC**.BSY 被置 1 时置位 **CON**.ST 不起作用。要启动一次新的计算，必须等到 **STATC**.BSY 不再有效时重新置位 **CON**.ST。同样地，在一次运算正在进行期间 (通过 **STATC**.BSY 指示)，改变操作模式也不起作用。

在自动启动模式 (**CON**.ST_MODE = 0)，在 **STATC**.BSY 被置 1 期间发生的对 **CORDX** 的写访问不会启动一次新的计算，这与 **CON**.ST_MODE = 1 的情况类似。

7.3.2.2 规格化的结果数据

在所有操作模式下，服务请求处理 返回一个规格化的结果数据到 X 和 Y，如下面的方程所示：

$$\text{X 或 Y 结果数据} = \frac{\text{CORDIC 计算数据}}{\text{MPS}}$$

另一方面，根据所使用的 CORDIC 函数不同，对 Z 结果数据的解读是不同的：

对于线性函数，对 CORDIC 计算的 Z 数据没有额外的处理过程，它本身就被直接当作结果数据。可访问的 Z 结果数据是一个实数，表示为有符号的 4Q19。

对于三角函数和双曲函数，可访问的 Z 结果数据是一个规格化的整数值，其范围为 $[-2^{23},(2^{23}-1)]$，代表的角度范围为 $[-\pi,((2^{23}-1)/2^{23})\pi]$。服务请求处理要求将 Z 数据解读如下：

$$??Z???? = ??Z???(?????) \times \frac{8388608}{\pi}$$

$$??Z???(?????) = Z???? \times \frac{\pi}{8388608}$$

CORDIC 计算出的数据包含一个固有的由旋转或向量引起的增益系数 K。每个 CORDIC 函数的 K 值是不同的，如 **表 7-2** 所示。

表 7-2 CORDIC 函数固有的结果数据增益系数

函数	近似增益 K
三角函数	1.646760258121
双曲函数	0.828159360960
线性函数	1

7.3.3 服务请求处理工作模式

表 7-3 给出了服务请求处理工作模式的概览。在该表中，X、Y 和 Z 代表初始数据，X_{final}、Y_{final} 和 Z_{final} 代表当所有处理过程都结束并且 BSY 不再有效时的最终结果数据。

CORDIC 方程式为 ：

$$x_{i+1} = x_i - m \cdot d_i \cdot y_i \cdot 2^{-i} \quad (7.3)$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i} \quad (7.4)$$

$$z_{i+1} = z_i - d_i \cdot e_i \quad (7.5)$$

表 7-3 服务请求处理工作模式和相应的结果数据

函数	旋转模式	向量模式
	d_i = sign (z_i), $z_i \rightarrow 0$	d_i = -sign (y_i), $y_i \rightarrow 0$
圆函数 m = 1 $e_i = \text{atan}(2^{-i})$	X_{final} = K[X cos(Z) - Y sin(Z)] / MPS Y_{final} = K[Y cos(Z) + X sin(Z)] / MPS Z_{final} = 0 其中 K ≈ 1.646760258121	X_{final} = K sqrt(X^2+Y^2) / MPS Y_{final} = 0 Z_{final} = Z + atan(Y / X) 其中 K ≈ 1.646760258121
	为了计算 cos(Z) and sin(Z)，令 X = 1 / K, Y = 0。 定义域：由于有预处理逻辑，所以支持满范围的 X, Y 和 Z 。	为了计算向量的模 (sqrt(x^2+y^2))，令 X = x / K, Y = y / K。 定义域：由于有预处理和后处理逻辑，所以支持满范围的 X 和 Y。 用于计算 atan(Y / X), 令 Z = 0。 定义域：满范围的 X 和 Y， X = 0 除外。
	关系： tan(v) = sin(v) / cos(v)	关系： acos(w) = atan[sqrt(1-w^2) / w] asin(w) = atan[w / sqrt(1-w^2)]
线性函数 m = 0 $e_i = 2^{-i}$	X_{final} = X / MPS Y_{final} = [Y + X Z] / MPS Z_{final} = 0	X_{final} = X / MPS Y_{final} = 0 Z_{final} = Z + Y / X
	为了计算 $X \cdot Z$, 令 Y = 0。 定义域：|Z| £ 2。	为了计算比值 Y / X, 令 Z = 0。 定义域：|Y / X| ≤ 2, X > 0。

表 7-3　服务请求处理工作模式和相应的结果数据

函数	旋转模式	向量模式
双曲函数 m = -1 $e_i = \text{atanh}(2^{-i})$	X_{final} = k[X cosh(Z) + Y sinh(Z)] / MPS Y_{final} = k[Y cosh(Z) + X sinh(Z)] / MPS Z_{final} = 0 其中 k ≈ 0.828159360960	X_{final} = k sqrt(X^2-Y^2) / MPS Y_{final} = 0 Z_{final} = Z + atanh(Y / X) 其中 k ≈ 0.828159360960
	为了计算 cosh(Z) 、 sinh(Z) 和 e^Z, 令 X = 1 / k, Y = 0。 定义域：$\|Z\| \le 1.11$rad, Y = 0。	为了计算 sqrt(x^2-y^2), 令 X = x / k, Y = y / k. 定义域：$\|y\| < \|x\|$, $X > 0$。 为了计算 atanh(Y / X), 令 Z = 0。 定义域：$\|\text{atanh}(Y / X)\| \le 1.11$rad, $X > 0$。
	关系： tanh(v) = sinh(v) / cosh(v) e^v = sinh(v) + cosh(v) $w^t = e^{t\ \ln(w)}$	关系： ln(w) = 2 atanh[(w-1) / (w+1)] sqrt(w) = sqrt((w+0.25)^2-(w-0.25)^2) acosh(w) = ln[w+sqrt(1-w^2)] asinh(w) = ln[w+sqrt(1+w^2)]

用法说明

- 对于每种函数的计算，用户必须初始化 CORDIC 数据 (X, Y 和 Z)，数据必须是收敛域内有意义的初始值以确保结果收敛。**表 7-3** 中列出的 “ 定义域 ” 涵盖了所支持的 CORDIC 算法收敛域，而且排除了对函数没有意义的范围。欲了解关于所支持的收敛域的详细信息，请参见 **7.3.3.1 节**。关于结果数据的精度，请参见 **7.3.5 节**。
- 必须考虑函数的限制，例如，设置初始 X = 0 对 atan(Y / X) 是没有意义的。违反这些函数限制可能会产生紊乱的 CORDIC 结果数据。
- 所有的输入数据都作为补码处理。唯一的例外是用户可选择将 X 结果数据 （仅对 X）作为无符号数读取。
- 结果数据总是正值并且比初始值大的唯一一种情况是圆向量模式下的 X 结果数据（仅对 X）。因此用户可能希望将 MSB 位用作数据位而不是符号位。通过设置 X_USIGN = 1， X 结果数据可以作为无符号数处理。

对于圆函数和双曲函数，由于有相应的固定 LUT （查找表）， Z 数据总是被作为有符号整数 S28 (可按 S23 访问) 来处理。LUT 包含 atan(2^{-i}) 和 atanh(2^{-i}) 的定标整数值 (S28)，其中 i = 0, 1, 2, ..., 27 ； 这使整数值的范围为 [-2^{23},(2^{23}-1)]，表示的角度范围为 [-π,((2^{23}-1)/2^{23})π] 。因此 Z 数据被限定为 （不考虑收敛域）表示这些 CORDIC 函数的角度 [-π,((2^{23}-1)/2^{23})π]。任何计算出的 Z 值若超出该范围都将导致溢出错误。

- 对于线性函数，Z 数据总是被作为有符号小数 S4.24 处理 (可按 S4.19 访问，格式是有符号的 4Q19)。模拟的 LUT 实际上是一个按 1.23 格式保存数据的移位寄存器，该格

式给出 2^{-i} 的真值。因此，不管收敛域如何， Z 数据在逻辑上只对那些模小于 16 的数值才是有用的。溢出错误由 EVFR.CDERR 位指示。

- MPS的设置对Z数据没有影响。用户必须确保正确地初始化Z初始数据，以防止产生溢出和不正确的结果数据。
- 服务请求处理的设计保证在使用 MPS > 1 这样正确的用户设置时， X 和 Y 数据不会产生内部溢出，读取的结果数据是正确的。但是需要注意，在这种情况下， MPS 的设置值越大，则结果数据的分辨率越低，这是因为损失了 LSB 位。
- 在双曲旋转模式，由于初始 Y 数据必须被设置为 0，结果精度受到限制。换句话说，服务请求处理只用一次计算不能返回精确的 cosh(Z)+/-sinh(Z) 结果 。

7.3.3.1 收敛域

本文档的目的并非阐述有关 CORDIC 收敛的固有限制的理论，在旋转模式或向量模式，不同 CORDIC 函数的收敛域是不同的。

对于结果数据的收敛域，根据所用的工作模式不同，初始数据的值或模是有限制的，相应的数据格式也有限制。以下是关于 CORDIC 结果数据收敛的一般适用结论。

旋转模式： Z 数据必须收敛于 0。初始 Z 数据必须等于或小于 $\sum d_i \cdot e_i$，其中 e_i 总是随迭代次数 i 的增大而减小。换句话说，|Z| ≤ LUT 的和。对于圆函数，这意味着 |Z| ≤ 代表 1.74 弧度的整数值。对于线性函数，|Z| ≤ 2。对于双曲函数，|Z| ≤ 代表 1.11 弧度的整数值。

向量模式： Y 数据必须收敛于 0。 X 和 Y 的初始值被 Z 函数限定，而 Z 函数取决于对应的 LUT。对于圆函数，这意味着 |atan(Y / X)| ≤ 1.74 弧度。对于线性函数， |Y / X| ≤ 2。对于双曲函数，|atanh(Y / X)| ≤ 1.11 弧度。在向量模式，还要求 X > 0。

虽然服务请求处理的工作模式通常受这些收敛极限值的限定，但对圆旋转模式和圆向量模式还是有例外，这些模式可以用额外的预处理(和后处理)逻辑来支持更宽的输入范围。

圆旋转模式： 支持满范围 $[-2^{23},(2^{23}-1)]$ 的 Z 输入，代表的角度范围是 $[-\pi,((2^{23}-1)/2^{23})\pi]$。对 X 和 Y 初始输入没有限制，但要注意溢出，可以通过设置 MPS 克服溢出。

圆向量模式： 支持满范围 $[-2^{23},(2^{23}-1)]$ 的 X 和 Y 输入， Z 初始值应当满足 |Z| ≤ π /2 ，以防止可能的 Z 结果数据溢出。

注： 应当考虑函数的限制，例如结果数据的意义，被零除就是没有意义的。表 7-3 中包含的每种函数的 “ 定义域 ” 意在涵盖函数的 CORDIC 收敛域和有用范围。

注： 输入值可能位于收敛域内，但这并不能保证 CORDIC 结果数据有固定的精度级。有关服务请求处理精度的详细信息，请参见 7.3.5 节 。

7.3.3.2 溢出考虑

除了考虑收敛域之外，还必须考虑输入数据幅值的限制，以防止结果数据溢出。

在所有工作模式下，服务请求处理都以相同的方式处理数据溢出。用户可以通过正确设置 MPS 来避免 X 和 Y 数据溢出。MPS 值的选择在一定程度上要基于服务请求处理的工作模式和应用数据。

MPS 的设置对 Z 数据没有影响。对于圆函数和双曲函数，超出 $[-\pi,((2^{23}-1)/2^{23})\pi]$ 范围的任何 Z 值都不能被表示，而且会导致 Z 数据溢出错误。注意，内核数据 Z 的数值范围是 $[-\pi,((2^{23}-1)/2^{23})\pi]$ ，换算后的范围是 $[-2_{23},(2_{23}-1)]$，所以 Z 数据的读出值和写入值总是被这样规格化的。对于线性函数， Z 是一个实际值， Z 的模不能超过 4 个整数位。

7.3.4 服务请求处理数据格式

服务请求处理接受补码格式的 （初始） X、 Y 和 Z 输入 数据。结果数据也是补码格式。唯一的例外是圆向量模式下的 X 结果数据。X 结果数据有一个默认的补码数据格式，但是用户可以通过设置 **CON**.X_USIGN = 1 来选择将 X 结果 数据作为无符号值读取。该选项能防止潜在的 X 结果数据溢出 (结合 MPS 设置)，因为 MSB 位现在是一个数据位。注意，只有在圆向量工作模式时，设置 **CON**.X_USIGN = 1 才会有效，这样总能产生正的结果数据，而且比初始值大。

一般来说，输入数据 X 和 Y 可以是整数或有理数 （小数）。但是在任何计算中， X 和 Y 的数据格式必须相同。而且在小数的情况下， X 和 Y 必须具有相同的小数位数。

对于圆函数或双曲函数， Z 数据总是基于规格化因子被处理的整数。在线性函数的情况下，可访问的 Z 数据是一个实数，并且具有固定的输入和结果数据格式 S4.19 (有符号的 4Q19)，该格式有 19 个小数位。

有关数据规格化的详细信息，请参见 **7.3.2.2 节**。**表 7-4** 概括了初始和结果数据格式。

表 7-4 X、 Y 和 Z 数据格式一览表

函数	数据		模式	
			旋转	向量
圆函数	X	初始	24 位补码	
		结果	24 位补码	24 位补码 / 24 位无符号数 (**CON**.X_USIGN = 1)
	Y	初始	24 位补码	
		结果		
	Z	初始		
		结果		
线性函数	X	初始		
		结果		
	Y	初始		
		结果		
	Z	初始	S4.19	
		结果		

表 7-4　　X、 Y 和 Z 数据格式一览表

函数	数据		模式	
			旋转	向量
双曲线函数	X	初始	24 位补码	
		结果		
	Y	初始		
		结果		
	Z	初始		
		结果		

7.3.5　　服务请求处理的精度

每次 CORDIC 计算包含固定的 27 次 CORDIC 迭代，从迭代序号 0 开始。双曲函数在这方面的特殊之处是它从迭代序号 1 开始，按定义的步数重复迭代。可寻址的数据寄存器为 24 位宽，而用于计算的内核 X 和 Y 数据寄存器是 35 位宽 (32 个数据位加上 2 个扩展位和 1 个溢出位)，内核 Z 数据寄存器是 31 位宽 (29 个数据位加 1 个符号扩展位和 1 个溢出位)。有关 LUT 的数据格式的详细信息，请参见 **7.3.7.1 节** 和 **7.3.7.2 节**。

对于在指定定义域内 (见**表 7-3**) 的输入数据，服务请求处理的每次计算结果都确保收敛。但是由于每种工作模式下数据格式不同，所以精度是不固定的。精度是结果数据与所期望的来自一个高精度计算器的结果之间差值大小的一种测量。归一化偏差 (ND) 是一个通用术语，它指示结果数据与期望结果的偏差大小。，偏差值是将输入 / 结果数据作为整数来计算得到的。在数据是有理数的情况下，偏差的大小必须要加以阐释。例如，线性向量模式下的 Z 数据格式是 S4.19 —— 则 ND = 1 (01_B) 意味着与预期真实数据的之间的差值不大于 $|2^{-19} + 2^{-19}|$； ND = 2 (10_B) 表示差值不大于 $|2^{-18}+2^{-19}|$； ND = 3 (11_B) 表示差值不大于 $|2^{-19}+2^{-18}+2^{-19}|$； ND = 4 ($100_B$) 表示差值不超过 $|2^{-17}+2^{-19}|$，依此类推。差值中总是加上 2^{-19}，这是因为可能的截断误差为 2^{-19}。

表 7-5 列出了一次计算中归一化偏差的概率，这是在每种服务请求处理工作模式下用大约一百万个不同的输入集并基于规定的输入条件 （总是位于定义域内，可能还有附加条件）而获得的结果。

如果用有理数 (小数) 代替整数进行计算，则很容易提高每种模式的精度。这只适用于 X 和 Y 数据 (X 和 Y 必须总是具有相同的数据格式)，而 Z 的数据格式是固定的，由相应的 LUT （查找表）的定义决定。很明显，我们可以预期，对于给定的输入 X 和 Y （和 Z），计算结果总是会返回一个常数值，无论 X 和 Y 是整数还是有理数。唯一的区别对输入和结果数据的阐释，即没有小数位或有多少小数位。如果 X 和 Y 数据是整数而不是有理数，CORDIC 结果与预期数据的偏差不可能更小。因此，在可能的情况下，要指定 X 和 Y 为有理数，并谨慎选择小数点的位置 ，选择的依据是所用模式下的最大 ND 值。

表 7-5　一次计算的归一化偏差

模式	X 的归一化偏差	Y 或 Z 的归一化偏差
圆向量模式	输入条件：定义域和 $[(1.64676/2)\cdot sqrt(X^2+Y^2) \geq 600]$	
	0 : 97.8830% 1 : 2.1150% 2 : 0.0010% 3 : 0.0010% X ≤ 3 的 ND 值	0 : 69.6738% 1 : 29.8307% 2 : 0.4919% 3 : 0.0036% Z ≤ 3 的 ND 值
圆旋转模式	输入条件：定义域 (满范围的 X, Y 和 Z)	
	0 : 47.8958% 1 : 50.3470% 2 : 1.7561% 3 : 0.0006% 4 : 0.0003% 5 : 0.0002% X ≤ 5 的 ND 值	0 : 47.8773% 1 : 50.3626% 2 : 1.7438% 3 : 0.0103% 4 : 0.0052% w5 : 0.0008% Y ≤ 5 的 ND 值
线性向量模式	输入条件：定义域 (|Y / X| ≤ 2, X > 0)	
	0 : 99.5869% 1 : 0.4131% X ≤ 1 的 ND 值	0 : 47.6909% 1 : 52.2723% 2 : 0.0239% 3 : 0.0129% Z ≤ 3 的 ND 值
线性旋转模式	输入条件：定义域 (|Z| ≤ 2)	
	0 : 100% X ≤ 0 的 ND 值	0 : 48.2181% 1 : 50.7801% 2 : 0.9996% 3 : 0.0022% Y ≤ 3 的 ND 值
双曲向量模式	输入条件：定义域 (|Y| < |X|, X > 0, |atanh(Y / X)| ≤ 1.11rad)	
	0 : 98.1092% 1 : 1.8855% 2 : 0.0017% 3 : 0.0019% 4 : 0.0017% X ≤ 4 的 ND 值	0 : 47.7010% 1 : 51.2954% 2 : 0.9938% 3 : 0.0098% Z ≤ 3 的 ND 值

表 7-5　一次计算的归一化偏差

模式	X 的归一化偏差	Y 或 Z 的归一化偏差
双曲旋转模式	输入条件：定义域 (|Y| < |X|, X > 0, |atanh(Y / X)| ≤ 1.11rad)	
	0 : 47.6358% 1 : 51.1029% 2 : 1.2578% 3 : 0.0012% 4 : 0.0007% 5 : 0.0006% 6 : 0.0005% 7 : 0.0002% 8 : 0.0003% X ≤ 8 的 ND 值	0 : 47.5419% 1 : 51.2064% 2 : 1.2478% 3 : 0.0013% 4 : 0.0010% 5 : 0.0007% 6 : 0.0006% 7 : 0.0003% Y ≤ 7 的 ND 值

注：　对于多步计算的最终结果而言，上面所描述的每种模式的精度／偏差并不能得到保证。例如，如果一次运算包含两次 CORDIC 计算，第二次计算使用第一次计算的结果数据 (通过将对应的 KEEP 位置 1 来使能)。这是由近似值和误差的累积而造成的。

7.3.6　服务请求处理的性能

CORDIC 的计算时间随着计算精度的增加而线性增加。精度的增加是通过更多次数的迭代来获得的，这就要求增加数据参数的宽度。

服务请求处理使用桶式移位器实现数据移位。因为每次计算都需要固定的 27 次迭代，所以从启动计算到 EVFR.CDEOC 标志被置位那一刻的总时间大约是 62 个内核时钟周期。需要注意的是，EVFR.CDERR 标志在 EVFR.CDEOC 被置位之后一个内核时钟周期才有效。一次完整计算的时序也适用于那些包含额外数据处理的模式，也适用于包含重复迭代和一个额外的模式设置时钟周期的双曲模式。

注：　上面提到的时序不包括软件加载初始数据到6 个数据寄存器和从 6 个数据寄存器读取最终结果的时间。

7.3.7　服务请求处理查找表

本节为用户提供关于服务请求处理查找表的须知信息。

本节描述 CORDIC 协处理器内核使用的查找表。

7.3.7.1　反正切和双曲反正切查找表

反正切查找表 （atan LUT）和双曲反正切查找表 （atanh LUT）分别为 29 位和 30 位宽。反正切 LUT 的每一项被分成一个符号位 （MSB）和后面跟随的 28 位整数部分。对于双曲反正切 LUT，每一项有一个重复位（MSB），后面跟随一个符号位，然后是 28 位的整数部分。

LUT 内容是：

- 数据格式为 S29 的反正切 LUT，见表 7-6。

表 7-6 atan(2^{-i}) 的预计算定标值

迭代序号	十六进制格式的 atan(2^{-i}) 定标值	迭代次数	十六进制格式的 atan(2^{-i}) 定标值
i = 0	4000000	i = 14	145F
i = 1	25C80A4	i = 15	A30
i = 2	13F670B	i = 16	518
i = 3	A2223B	i = 17	28C
i = 4	5161A8	i = 18	146
i = 5	28BAFC	i = 19	A3
i = 6	145EC4	i = 20	51
i = 7	A2F8B	i = 21	29
i = 8	517CA	i = 22	14
i = 9	28BE6	i = 23	A
i = 10	145F3	i = 24	5
i = 11	A2FA	i = 25	3
i = 12	517D	i = 26	1
I = 13	28BE		

- 数据格式为 S29 的双曲反正切 LUT，见表 7-7。

表 7-7 atanh(2^{-i}) 的预计算定标值

迭代序号	十六进制格式的 atanh(2^{-i}) 定标值	迭代次数	十六进制格式的 atanh(2^{-i}) 定标值
i = 0	-	i = 14	145F
i = 1	2CC2F12	i = 15	A30
i = 2	14D01AC	i = 16	518
i = 3	A3D4E0	i = 17	28C
i = 4	5197FC	i = 18	146
i = 5	28C1C7	i = 19	A3
i = 6	145F9D	i = 20	51
i = 7	A2FA6	i = 21	29

表 7-7　　atanh(2^{-i}) 的预计算定标值

迭代序号	十六进制格式的 atanh(2^{-i}) 定标值	迭代次数	十六进制格式的 atanh(2^{-i}) 定标值
i = 8	517CE	i = 22	14
i = 9	28BE6	i = 23	A
i = 10	145F3	i = 24	5
i = 11	A2FA	i = 25	3
i = 12	517D	i = 26	1
I = 13	28BE		

Z 数据是实际角度的规格化表示。例如内部换算将 [-π,((2^{23}-1)/2^{23})π] 等价于 [-2^{28},(2^{28}-1)]。当被寻址时，最后的 5 个 LSB 位被截断， 24 位数据被传送到数据总线。从用户的角度看，角度 [-π,((2^{23}-1)/2^{23})π] 由范围 [-2^{23},(2^{23}-1)] 来代表。

7.3.7.2　线性函数的模拟查找表

线性函数的模拟 LUT 实际是一个移位寄存器。该模拟 LUT 有一个整数位 （MSB），后面跟随 23 位 1Q23 格式的小数位。

在线性函数中， Z 是一个实数，内部 的 Z 数据是格式为 4Q19 的有符号数。外部读取的数据是小数部分最后 5 位被截断后的结果，这就导致读取的数据是一个符号位后面跟随 4 位整数部分，最后是 19 位的小数部分。

7.4　全局函数

因为 DIV 和 CORDIC 都有自己的控制和数据寄存器组 ，所以它们可以各自独立工作。

然而，它们共用一个公共总线接口和一些全局函数。

7.4.1　结果链接

MATH 协处理器支持 DIV 和 CORDIC 之间的结果链接。

对于 DIV，这意味着每个操作数寄存器即 DVD 和 DVS，都可以被来自任何一个结果寄存器 (DIV 中的 QUOT 和 RMD ； CORDIC 中的 CORR[Z:X]) 的值更新。

对于 CORDIC，这意味着每个操作数寄存器 CORD[Z:X] 都可以被来自 DIV 结果寄存器 QUOT 或 RMD 的值更新。

在这两种情况下，都是通过 GLBCON 寄存器中的操作数寄存器结果链接位域 (xRC) 来选择用哪个寄存器的值更新操作数寄存器。

注：　要用来自相应的 CORR[Z:X] 寄存器的值更新 CORDIC CORD[Z:X] 寄存器，必须使用 STATC 寄存器中的 KEEP[Z:X] 位进行单独控制。

一旦 前面的 DIV 或 CORDIC 运算结束，数据就被更新，同时更新当前结果寄存器。图 7-5 给出了链接 DIV 商值到 CORDIC 的 X 操作数的例子。

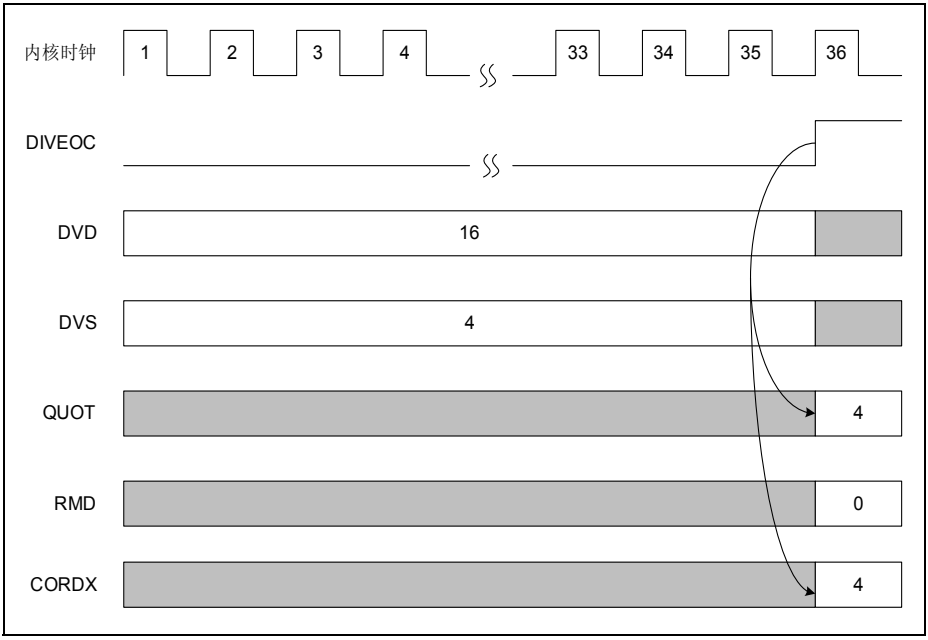


图 7-5　　CORDXRC = 01_B 时的除法操作

7.4.1.1　启动模式为 0 时的结果链接

当相应的启动模式为 0 时，应用软件可以用写 DVS 和 CORDX 操作数寄存器的方式启动 DIV 和 CORDIC 运算 (见 7.2.2.1 节 和 7.3.2 节)。结果链接引起的对这些寄存器的更新等价于硬件写操作，因此具有相同的效果。

为了避免潜在的死锁情况，应用程序在使用结果链接功能时，必须注意避免以下两种情况：

1. 在 DIVCON.STMODE = 0 时， DVS 被链接到 QUOT 或 RMD
2. DVS被链接到CORRx，CORDX被链接到 QUOT 或 RMD，而此时DIVCON.STMODE 和 CON.ST_MODE 都等于 0。

7.4.1.2　结果链接被使能时的忙标志处理

使用图 7-5 给出的例子，一个 DIV 结果被链接到 CORDIC 操作数 CORDX。如果 CON.ST_MODE= 0，启动 DIV 计算不仅会置位 DIV 忙标志，还会置位 CORDIC 的忙标志。即使 CORDIC 起初没有被激活，也会出现这种情况。也就是说，只有当 DIV 计算完

成，并且 QUOT 或 RMD 的值被写入到 CORDX 时， CORDIC 计算才会启动。类似地，在 DIV 计算完成后变为非活动状态的 DIV 不会立即清除忙标志，而是在 CORDIC 计算也完成后， DIV 和 CORDIC 的忙标志同时被清除。

上述规则在另一方向也适用，即如果一个 CORDIC 结果被链接到 DIV 的 DVS 操作数寄存器且 DIVCON.STMODE= 0 的情况。

在忙标志位被置位期间：

- 读 DIV 或 CORDIC 的 结果寄存器会导致在总线上插入等待状态。
- 在 DIV 或 CORDIC 处于活动状态期间，启动一次新的 DIV 或 CORDIC 计算或者向它们各自的控制寄存器 DIVCON 和 CON 进行写操作是无效的。
- 然而，如果 DIV 或 CORDIC 仍处于非活动状态，或刚从活动状态返回到非活动状态，启动一次新的 DIV 或 CORDIC 计算或者向它们各自的控制寄存器进行写操作是有效的。

在 DIV 和 CORDIC 的忙标志被置位期间，必须避免启动一次新的计算或向控制寄存器进行写操作，否则计算结果可能被破坏。

7.5 服务请求的产生

如果被 EVIER 寄存器中各自的中断使能位使能， 则 DIV 和 CORDIC 的错误及计算结束事件会向 NVIC 触发中断服务请求。这些事件由 EVFRT 寄存器中的事件标志指示。只能通过向 EVFCR 寄存器中的事件标志清零位写 1 来清除事件标志位。

向 EVFSR 寄存器中的事件标志置 1 位写 1 与计算结束事件具有相同的作用。

图 7-6 给出了 MATH 协处理器中的中断结构。

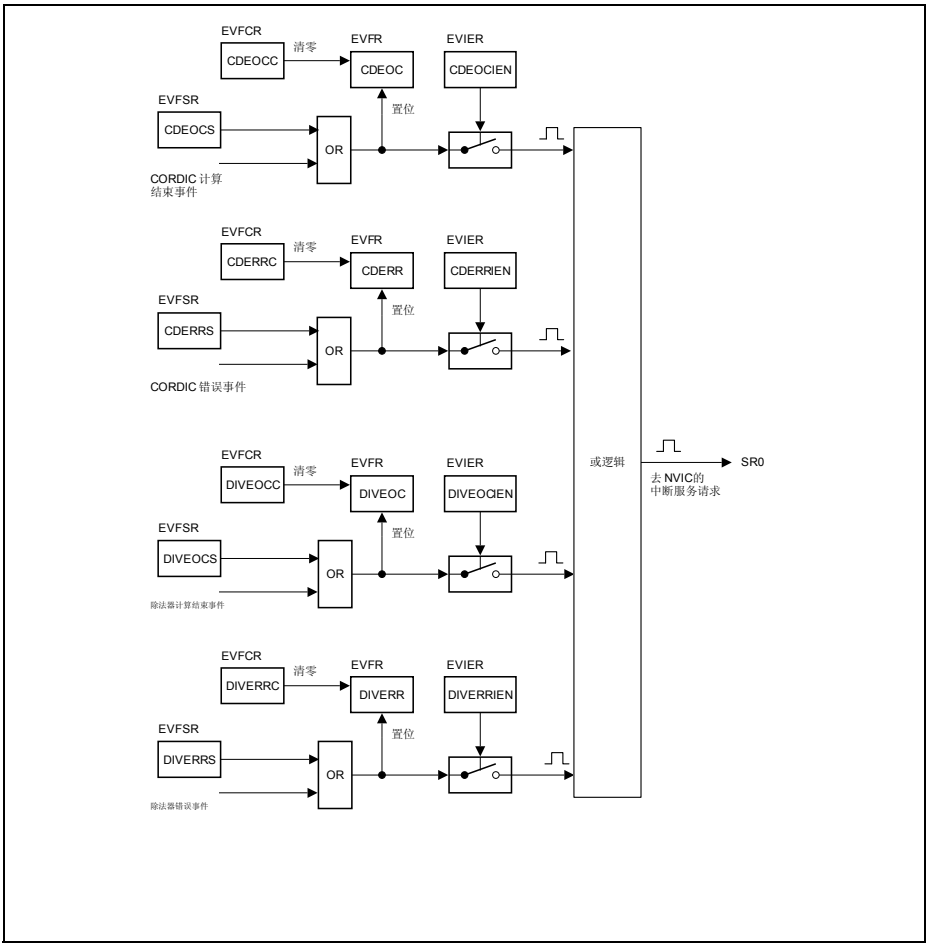


图 7-6　　中断结构

7.6　　调试行为

MATH 可以被配置为在 CPU 的程序执行被调试器停止时进入挂起模式 (由挂起信号有效来指示)。

有两种挂起模式，可以通过控制位域 GLBCON.SUSCFG 来选择挂起模式：

- 硬挂起模式
  - 内核时钟立即被关闭，因而停止所有计算
- 软挂起模式

- 任何一次活动的计算都被允许继续进行，在该计算结束后内核时钟才被关闭。

在内核时钟被关闭以后，所有的寄存器都变成只读状态。在该状态写寄存器无效。当挂起信号变为无效时，挂起模式结束，恢复正常操作。

对于寄存器位而言，挂起模式是非侵入性的。这意味着当进入或离开挂起模式时，硬件不能修改寄存器位。

注： 在 XMC1300 中，任何一次复位都会将 GLBCON.SUSCFG 位域复位到其默认值。如果在调试期间需要使用挂起功能，建议在用户初始化代码部分使能该功能。在对该位域编程之前，必须使能接口时钟。当按 SCU 一章中的 CCU(时钟选通控制) 一节所述来使能接口时钟时，需要特别谨慎。

7.7 电源、复位和时钟

MATH 协处理器位于内核电源域。该模块，包括所有寄存器，都会在系统复位时被复位到其默认状态。

MATH 协处理器需要两个输入时钟信号，一个用于内核时钟，另一个用于接口时钟。内核时钟与接口时钟的比值可以是 2:1 或 1:1，但它们必须始终互相同步。

这两个时钟在默认情况下都被禁止，可以通过 SCU_CGATCLR0 寄存器使能。使能和禁止这两个时钟可能引起负载变化，并出现如 SCU 一章中 CCU(时钟选通控制) 一节所述的时钟消隐。强烈建议在用户初始化代码中设置这两个时钟，以避免在运行时出现时钟消隐。

7.8 寄存器

寄存器概述

寄存器绝对地址用下面的加法来计算：

模块基地址 + 偏移地址

注： 表 7-9 中所示的 DIV 寄存器支持 16 位和 32 位总线访问。所有其他寄存器 (全局和 CORDIC) 仅支持 32 位访问。

表 7-8 寄存器地址空间

模块	基地址	结束地址	备注
MATH	$4003\ 0000_H$	$4003\ FFFF_H$	

表 7-9 寄存器一览表

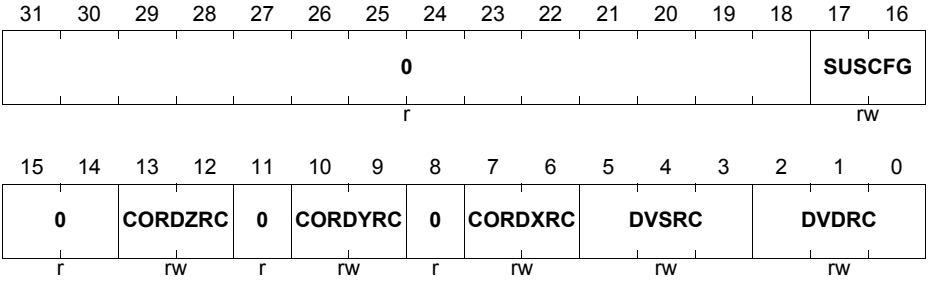
简称	描述	偏移地址	访问方式		描述见
			读	写	
全局寄存器					
Reserved	保留	0000_H	BE	BE	
GLBCON	全局控制寄存器	0004_H	U, PV	U, PV	页 7-24
ID	模块标识寄存器	0008_H	U, PV	BE	页 7-26
EVIER	事件中断使能寄存器	$000C_H$	U, PV	U, PV	页 7-27
EVFR	事件标志寄存器	0010_H	U, PV	U, PV	页 7-28
EVFSR	事件标志置位寄存器	0014_H	U, PV	U, PV	页 7-29
EVFCR	事件标志清除寄存器	0018_H	U, PV	U, PV	页 7-29
Reserved	保留	$001C_H$	BE	BE	
除法器寄存器					
DVD	被除数寄存器	0020_H	U, PV	U, PV	页 7-31
DVS	除数寄存器	0024_H	U, PV	U, PV	页 7-31
QUOT	商寄存器	0028_H	U, PV	BE	页 7-32
RMD	余数寄存器	$002C_H$	U, PV	BE	页 7-32
DIVST	除法器状态寄存器	0030_H	U, PV	BE	页 7-33
DIVCON	除法器控制寄存器	0034_H	U, PV	U, PV	页 7-34
Reserved	保留	0038_H - $003F_H$	BE	BE	
CORDIC 寄存器					
STATC	状态和数据控制寄存器	0040_H	U, PV	U, PV	页 7-35
CON	控制寄存器	0044_H	U, PV	U, PV	页 7-37
CORDX	X 数据寄存器	0048_H	U, PV	U, PV	页 7-38
CORDY	Y 数据寄存器	$004C_H$	U, PV	U, PV	页 7-39
CORDZ	Z 数据寄存器	0050_H	U, PV	U, PV	页 7-39
CORRX	X 结果寄存器	0054_H	U, PV	BE	页 7-39
CORRY	Y 结果寄存器	0058_H	U, PV	BE	页 7-40
CORRZ	Z 结果寄存器	$005C_H$	U, PV	BE	页 7-40

7.8.1 全局寄存器描述

GLBCON

GLBCON 寄存器包含 MATH 的全局控制位。

GLBCON
全局控制寄存器 **(0004_H)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
DVDRC	[2:0]	rw	**被除数寄存器结果链接** 当发生结果链接触发事件时， DIV 中的 DVD 寄存器将被所选择的结果寄存器的值更新。 000_B 未选择结果链接 001_B QUOT 寄存器是所选数据源 010_B RMD 寄存器是所选数据源 011_B CORRX 是所选数据源 100_B CORRY 是所选数据源 101_B CORRZ 是所选数据源 110_B 保留 111_B 保留

域	位	类型	描述
DVSRC	[5:3]	rw	**除数寄存器结果链接** 当发生结果链接触发事件时， DIV 中的 DVS 寄存器将被所选择的结果寄存器的值更新。 000_B 未选择结果链接 001_B QUOT 寄存器是所选数据源 010_B RMD 寄存器是所选数据源 011_B CORRX 是所选数据源 100_B CORRY 是所选数据源 101_B CORRZ 是所选数据源 110_B 保留 111_B 保留
CORDXRC	[7:6]	rw	**CORDX 寄存器结果链接** 当发生结果链接触发事件时，CORDIC 中的 CORDX 寄存器将被所选择的结果寄存器的值更新。 00_B 未选择结果链接 01_B QUOT 寄存器是所选数据源 10_B RMD 寄存器是所选数据源 11_B 保留
CORDYRC	[10:9]	rw	**CORDY 寄存器结果链接** 当发生结果链接触发事件时，CORDIC 中的 CORDY 寄存器将被所选择的结果寄存器的值更新。 00_B 未选择结果链接 01_B QUOT 寄存器是所选数据源 10_B RMD 寄存器是所选数据源 11_B 保留
CORDZRC	[13:12]	rw	**CORDZ 寄存器结果链接** 当发生结果链接触发事件时，CORDIC 中的 CORDZ 寄存器将被所选择的结果寄存器的值更新。 00_B 未选择结果链接 01_B QUOT 寄存器是所选的数据源 10_B RMD 寄存器是所选数据源 11_B 保留
SUSCFG	[17:16]	rw	**挂起模式配置** 该位决定当 CPU 暂停执行时， MATH 协处理器是否进入挂起模式。 00_B 永不进入挂起模式 01_B 当 CPU 暂停时，进入硬挂起模式。 10_B 当 CPU 暂停时，进入软挂起模式。 11_B 保留

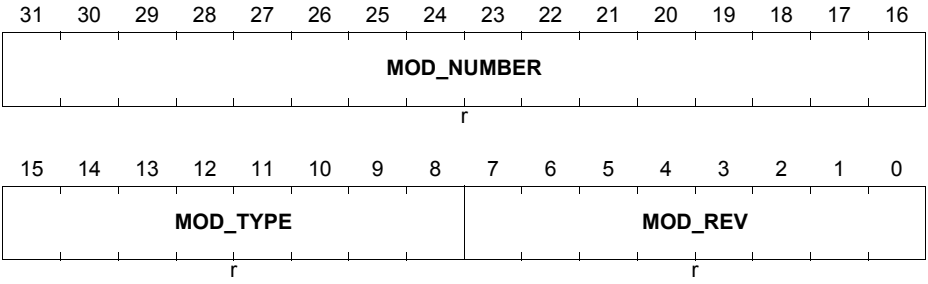
域	位	类型	描述
0	[31:18], [15:14], 11, 8	r	**保留** 读出值为 0 ； 应写入 0。

MATH_ID

MATH_ID 寄存器指示 MATH 协处理器的功能和设计阶段。

MATH_ID
模块标识寄存器

(0008$_H$) **复位值 : 00F2 C0XX$_H$**

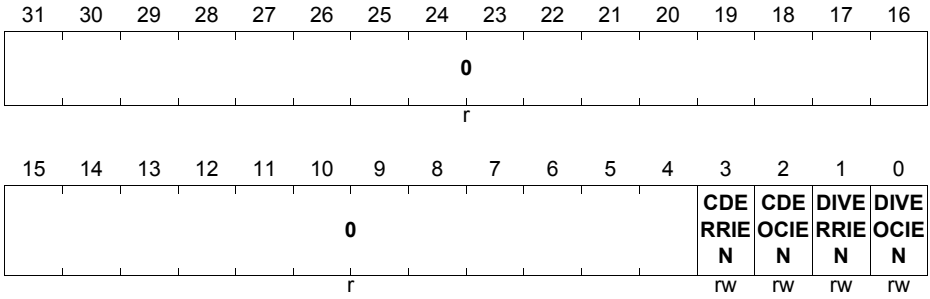


域	位	类型	描述
MOD_REV	[7:0]	r	**模块版本号** MOD_REV 定义版本号。模块的版本值从 01_H 开始 (第一个版本)。
MOD_TYPE	[15:8]	r	**模块类型** 该位域为 $C0_H$。它定义该模块为 32 位模块。
MOD_NUMBER	[31:16]	r	**模块编号值** 该位域定义模块标识号。

事件中断使能寄存器

该寄存器使能每个事件的中断产生。如果被使能，检测到事件后会通过服务请求输出产生中断脉冲。

EVIER
事件中断使能寄存器 **($000C_H$)** **复位值 : $0000\ 0000_H$**

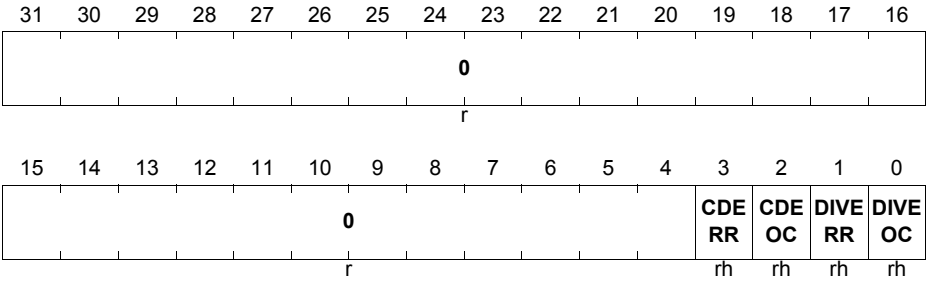


域	位	类型	描述
DIVEOCIEN	0	rw	**除法器计算结束中断使能** 0_B 禁止除法器计算结束中断。 1_B 使能除法器计算结束中断。
DIVERRIEN	1	rw	**除法器错误中断使能** 0_B 禁止除法器错误中断。 1_B 使能除法器错误中断。
CDEOCIEN	2	rw	**CORDIC 计算结束中断使能** 0_B 禁止 CORDIC 计算结束中断。 1_B 使能 CORDIC 计算结束中断。
CDERRIEN	3	rw	**CORDIC 错误中断使能** 0_B 禁止 CORDIC 错误中断。 1_B 使能 CORDIC 错误中断。
0	[31:4]	r	**保留** 读出值为 0；应写入 0。

事件标志寄存器

该寄存器包含每个事件的状态标志。如果置位，表示已检测到事件。

EVFR
事件标志寄存器 **(0010_H)** **复位值 : $0000\ 0000_H$**



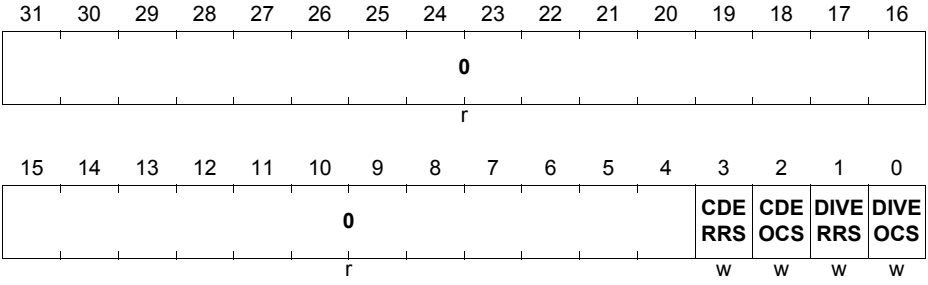
域	位	类型	描述
DIVEOC	0	rh	**除法器计算结束事件标志** 0_B 未检测到除法器计算结束事件。 1_B 检测到除法器计算结束事件。
DIVERR	1	rh	**除法器错误事件标志** 0_B 未检测到除法器错误事件。 1_B 检测到除法器错误事件。
CDEOC	2	rh	**CORDIC 计算结束事件标志** 0_B 未检测到 CORDIC 计算结束事件。 1_B 检测到 CORDIC 计算结束事件。
CDERR	3	rh	**CORDIC 错误事件标志** 0_B 未检测到 CORDIC 错误事件。 1_B 检测到 CORDIC 错误事件。
0	[31:4]	r	**保留** 读出值为 0；应写入 0。

事件标志置位寄存器

该寄存器允许应用程序置位EVFR寄存器中每个事件的状态标志，就像检测到事件一样。如果在此之前使能了 EVIER 寄存器中的事件中断，将会向 NVIC 产生一次中断请求。

向这些位写 0 无效，读这些位总是返回 0。

EVFSR
事件标志置位寄存器 **(0014_H)** **复位值 : 0000 0000$_H$**



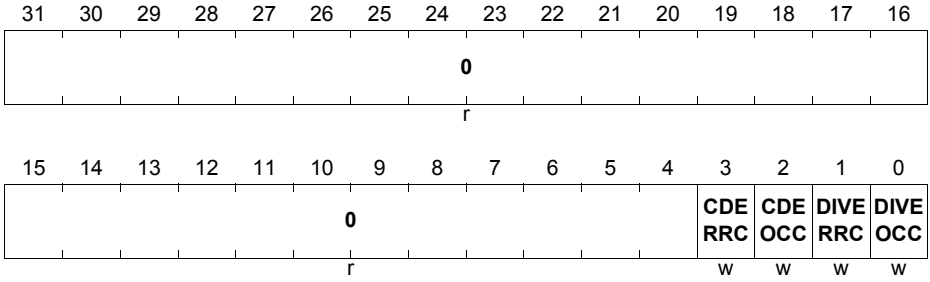
域	位	类型	描述
DIVEOCS	0	w	**除法器计算结束事件标志置位** 0_B 无效。 1_B 置位 EVFR 寄存器中的除法器计算结束事件标志。如果在 EVIER 中被使能，将会产生中断。
DIVERRS	1	w	**除法器错误事件标志置位** 0_B 无效。 1_B 置位 EVFR 寄存器中的除法器错误事件标志。如果在 EVIER 寄存器中被使能，将会产生中断。
CDEOCS	2	w	**CORDIC 事件标志置位** 0_B 无效。 1_B 置位 EVFR 寄存器中的 CORDIC 计算结束事件标志。如果在 EVIER 中被使能，将会产生中断。
CDERRS	3	w	**CORDIC 错误事件标志置位** 0_B 无效。 1_B 置位 EVFR 寄存器中的 CORDIC 错误事件标志。如果在 EVIER 寄存器中被使能，将会产生中断。

域	位	类型	描述
0	[31:4]	r	**保留** 读出值为 0；应写入 0。

事件标志清除寄存器

通过向事件标志清零寄存器相应的位写 1 来清零 EVFR 寄存器的事件标志。
向这些位写零无效，读这些位总是返回零。

EVFCR
事件标志清除寄存器 **(0018_H)** **复位值 : 0000 0000_H**



域	位	类型	描述
DIVEOCC	0	w	**除法器计算结束事件标志清除** 0_B 无效。 1_B 清除 EVFR 寄存器的除法器计算结束事件标志。
DIVERRC	1	w	**除法器错误事件标志清除** 0_B 无效。 1_B 清除 EVFR 寄存器的除法器错误事件标志。
CDEOCC	2	w	**CORDIC 计算结束事件标志清除** 0_B 无效。 1_B 清除 EVFR 寄存器的 CORDIC 计算结束事件标志。
CDERRC	3	w	**CORDIC 错误事件标志清除** 0_B 无效。 1_B 清除 EVFR 寄存器的 CORDIC 错误事件标志。
0	[31:4]	r	**保留** 读出值为 0；应写入 0。

7.8.2 除法器寄存器描述

DVD

DVD 寄存器用于存储除法运算的被除数。该寄存器可由软件写入，如果结果链接被使能，也可由硬件写入。

DVD
被除数寄存器 **(0020_H)** **复位值 : 0000 0000_H**

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
VAL																															
rwh																															

域	位	类型	描述
VAL	[31:0]	rwh	**被除数的值**

DVS

DVS 寄存器用于存储除法运算的除数。该寄存器可由软件写入，如果结果链接被使能，也可由硬件写入。

当 DIVCON.STMODE = 0 时，对 DVS 的写操作可以启动一次除法运算。

DVS
除数寄存器 **(0024_H)** **复位值 : 0000 0000_H**

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
VAL																															
rwh																															

域	位	类型	描述
VAL	[31:0]	rwh	**除数值**

QUOT

QUOT 寄存器用于存储除法运算的商。每当除法运算完成时，该寄存器被硬件自动更新。

注： 如果在BSY = 1 时读QUOT 寄存器，将会在总线上插入等待状态。仅当新的QUOT 值在当前计算结束后变为可用时，总线读事务才会完成。

QUOT
商寄存器 **(0028$_H$)** **复位值 : 0000 0000$_H$**

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
VAL																															
rh																															

域	位	类型	描述
VAL	[31:0]	rh	**商值**

RMD

RMD 寄存器用于存储除法操作的余数。当每次除法操作完成时，该寄存器被硬件自动更新。

注： 如果在 BSY = 1 时读 RMD 寄存器，将会在总线上插入等待状态。仅当新的 RMD 值在当前计算结束后变为可用时，总线读事务才会完成。

RMD
余数寄存器 **(002C$_H$)** **复位值 : 0000 0000$_H$**

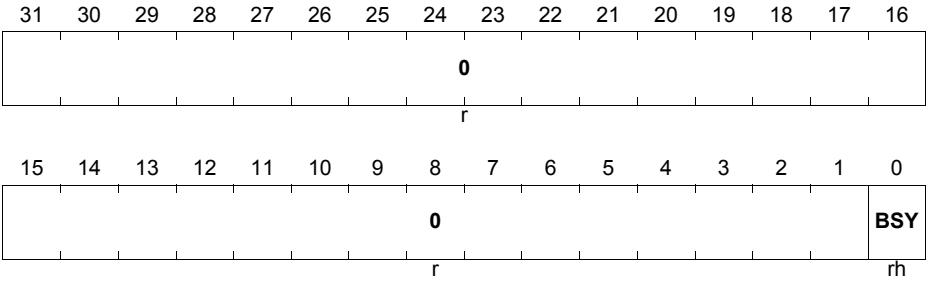
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VAL																															
rh																															

域	位	类型	描述
VAL	[31:0]	rh	**余数值**

DIVST

DIVST 寄存器包含 DIV 的状态标志。

DIVST
除法器状态寄存器 **(0030_H)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
BSY	0	rh	**忙指示** 0_B 除法器不在执行任何除法运算。 1_B 除法器仍在执行除法运算。
0	[31:1]	r	**保留** 读出值为 0；应写入 0。

DIVCON

DIVCON 寄存器包含 DIV 的控制位。

当 BSY=1 时，对 DIVCON 的写访问被忽略。在这种情况下不会产生总线错误。

注： 如果结果链接被使能，请参见 7.4.1.2 节 关于忙标志处理的描述。

DIVCON
除法器控制寄存器 **(0034$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			DVSSRC					0			DVDSLC				
r			rw					r			rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
QSDIR	0		QSCNT					0			DIVMODE		USIGN	STMODE	ST
rw	r		rw					r			rw		rw	rw	rwh

域	位	类型	描述
ST	0	rwh	**启动位** 0_B 无效。 1_B 当 STMODE=1_B 时启动除法运算。 该位在一个内核时钟周期后被硬件自动清零。
STMODE	1	rw	**启动模式** 选择除法运算的启动模式： 0_B 写 DVS 寄存器会自动启动一次计算。 1_B 通过将 ST 位置 1 来启动计算。 *注： 如果 BSY = 1，启动一次新的除法运算的请求将被忽略。*
USIGN	2	rw	**无符号除法使能** 0_B 选择有符号除法。 1_B 选择无符号除法。
QSDIR	15	rw	**商移位方向** 此位用于选择除法运算结束后商的移位方向： 0_B 左移。 1_B 右移。

域	位	类型	描述
QSCNT	[12:8]	rw	**商移位计数** 如果 QSCNT 不等于 0，它指示除法运算完成后对商进行移位的位数 。 如果 QSCNT=0，不会发生移位操作。
DVDSLC	[20:16]	rw	**被除数左移计数** 如果 DVDSLC 不等于 0，它指示除法运算开始前对被除数进行左移的位数。 如果 DVDSLC = 0，不会发生移位操作。
DVSSRC	[28:24]	rw	**除数右移计数** 如果 DVSSRC 不等于 0，它指示除法运算开始前对除数进行右移的位数。 如果 DVSSRC = 0，不会发生移位操作。
0	[31:29], [23:21], [14:13], [7:3]	r	**保留** 读出值为 0；应写 入 0。

7.8.3 CORDIC 寄存器描述

STATC

STATC 寄存器通常反映服务请求处理的状态。该寄存器还包含数据控制位。

STATC
CORDIC 状态和数据控制寄存器 (0040$_H$) 复位值 : 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								KEEPZ	KEEPY	KEEPX	0				BSY
r								rw	rw	rw	r				rh

域	位	类型	描述
BSY	0	rh	**忙指示** 当置位时，指示正在进行一次计算。该标志在 ST 被置位一个时钟周期后变为有效。在一次计算结束时，该位变为无效。
0	[4:1]	r	**保留**
KEEPX	5	rw	**最后一次的 X 结果作为新计算的初始数据** 如果置位，新的计算将会使用前一次计算的结果值作为初始数据。换句话说，在新的计算开始时，各自的内核数据寄存器不会被映射数据寄存器的内容覆盖。对于第一次计算，为了加载初始 X 数据，该位应被清零。 映射数据寄存器会继续保持最后写入的初始数据，直到发生下一下软件写操作，而与 KEEPx 位无关。 如果 KEEPx 位被置 1，即要进行一次多步计算，则相应的最终 x 结果数据的精度可能降低，并且不能保证，如 **7.3.5 节**所述。
KEEPY	6	rw	**最后一次的 Y 结果作为新计算的初始数据** < 见对 KEEPX 的描述 > 此外，当最后的 Y 结果收敛于 0 时，在向量模式下将该位置 1 可能没有意义。
KEEPZ	7	rw	**最后一次的 Z 结果作为新计算的初始数据** < 见对 KEEPX 的描述 > 此外 , 当最后的 Z 结果收敛于 0 时，在向量模式下将该位置 1 可能没有意义。
0	[31:8]	r	**保留**

CON

CON 寄存器允许对服务请求处理的全局控制。当 **STATC**.BSY 被置 1 时，对该寄存器的写操作无效。

*注： 如果使能了结果链接，请参见 **7.4.1.2 节**关于忙标志处理的描述。*

CON
CORDIC 控制寄存器 **(0044$_H$)** **复位值 : 0000 0062$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								MPS		X_USIGN	ST_MODE	ROTVEC	MODE		ST
r								rw		rw	rw	rw	rw		rwh

域	位	类型	描述
ST	0	rwh	**启动计算** 如果 ST_MODE = 1，置位 ST 即启动一次 CORDIC 计算。该位仅在 BSY 未被置 1 时有效。该位也可能在一次对 CON 寄存器写访问中随其他位一起被置 1。 该位在计算开始时被硬件清零。
MODE	[2:1]	rw	**工作模式** 00_B 线性模式 01_B 圆模式 (默认) 10_B 保留 11_B 双曲模式
ROTVEC	3	rw	**旋转向量选择** 0_B 向量模式 (默认) 1_B 旋转模式
ST_MODE	4	rw	**启动方式** 0_B 执行对X参数数据寄存器**CORDX**的写访问后自动启动计算 (默认)。 1_B 仅在 ST 位被置 1 后启动计算。

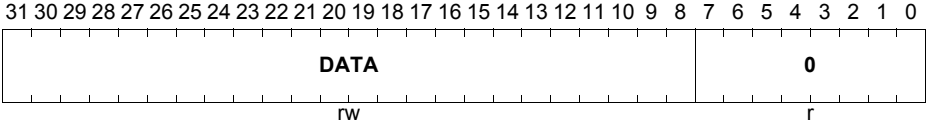
域	位	类型	描述
X_USIGN	5	rw	**圆向量模式下的 X 结果数据格式** 当读 X 结果数据时，X 数据有以下格式： 0_B 有符号，补码 1_B 无符号 (默认) 该位被置 1 时， X 结果数据的 MSB 位被作为数据位来处理，而不是符号位。 该位仅在圆向量模式下有效。在所有其他模式，X 总是被作为补码数据处理。 在圆向量模式， X_USIGN = 1 是有意义的，因为结果数据总是正值，并且总是大于初始值。
MPS	[7:6]	rw	**X 和 Y 大小预定标** 在一次计算的最后一次迭代之后，X 和 Y 的计算值除以该系数后得到最终结果。 正确地设置这些位对避免各个内核数据寄存器中的结果溢出是非常重要的。 00_B 除以 1 01_B 除以 2 (默认) 10_B 除以 4 11_B 保留，保持最后的 MPS 设置。
0	[31:8]	r	**保留**

CORDx

该数据寄存器是用于初始化 X, Y 和 Z 参数。

CORDX

CORDIC X 数据寄存器 **(0048_H)** **复位值 : 0000 0000_H**



域	位	类型	描述
0	[7:0]	r	**保留**
DATA	[31:8]	rw	**初始 X 参数数据** 在 **CON**.ST_MODE = 0 时写该寄存器将启动一次运算。

CORDY
CORDIC Y 数据寄存器 **(004C$_H$)** **复位值 : 0000 0000$_H$**

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
DATA																								0							
rw																								r							

域	位	类型	描述
0	[7:0]	r	保留
DATA	[31:8]	rw	初始 Y 参数数据

CORDZ
CORDIC Z 数据寄存器 **(0050$_H$)** **复位值 : 0000 0000$_H$**

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
DATA																								0							
rw																								r							

域	位	类型	描述
0	[7:0]	r	保留
DATA	[31:8]	rw	初始 Z 参数数据

CORRx

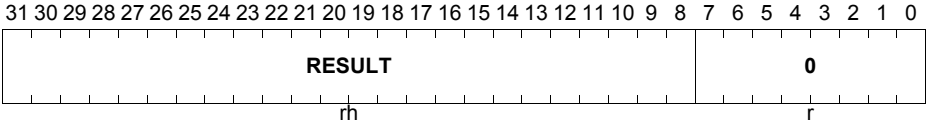
CORDIC 计算的结果数据将会被写入到各自的结果寄存器。在 **STATC**.BSY 被置 1 期间（计算仍在进行），对这些寄存器的任何读访问都会导致内核发起总线等待，直到 BSY 被复位。

CORRX
CORDIC X 结果寄存器 **(0054$_H$)** **复位值 : 0000 0000$_H$**

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
RESULT																								0							
rh																								r							

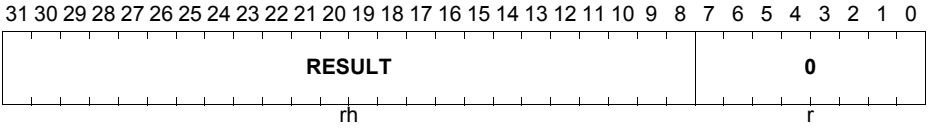
域	位	类型	描述
0	[7:0]	r	保留
RESULT	[31:8]	rh	**X 计算结果**

CORRY
CORDIC Y 结果寄存器 **(0058$_H$)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
0	[7:0]	r	保留
RESULT	[31:8]	rh	**Y 计算结果**

CORRZ
CORDIC Z 结果寄存器 **(005C$_H$)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
0	[7:0]	r	保留
RESULT	[31:8]	rh	**Z 计算结果**

7.9 互连

表 7-10 列出了模块的互联情况：

表 7-10 模块互连

输入 / 输出	I/O	连接到	描述
SR0	O	NVIC	**中断服务请求输出**
SR0	O	NVIC	中断服务请求输出。

片上存储器

8 存储器组织

本章描述系统的存储器组织、存储器访问和存储器保护策略。

参考文献

[1] Cortex®-M0 用户指南， ARM DUI 0497A (ID112109)

8.1 概述

XMC1300 的存储器映射遵循标准的 ARM Cortex-M0 系统存储器映射。

8.1.1 特性

存储器映射实现了以下特性：

- 与标准的 ARM Cortex-M0 CPU 兼容 **[1]**
- 整个 XMC1000 家族 完全兼容

8.2 存储器映射

表 8-1 详细定义了 XMC1300 的系统存储器映射，每个外设或存储实例都拥有其自己的地址空间。有关系统组件和外设的详细寄存器描述请参见本文档的相应章节。

注： 根据器件的型号不同，并非所有外设和存储器地址范围都是可用的。

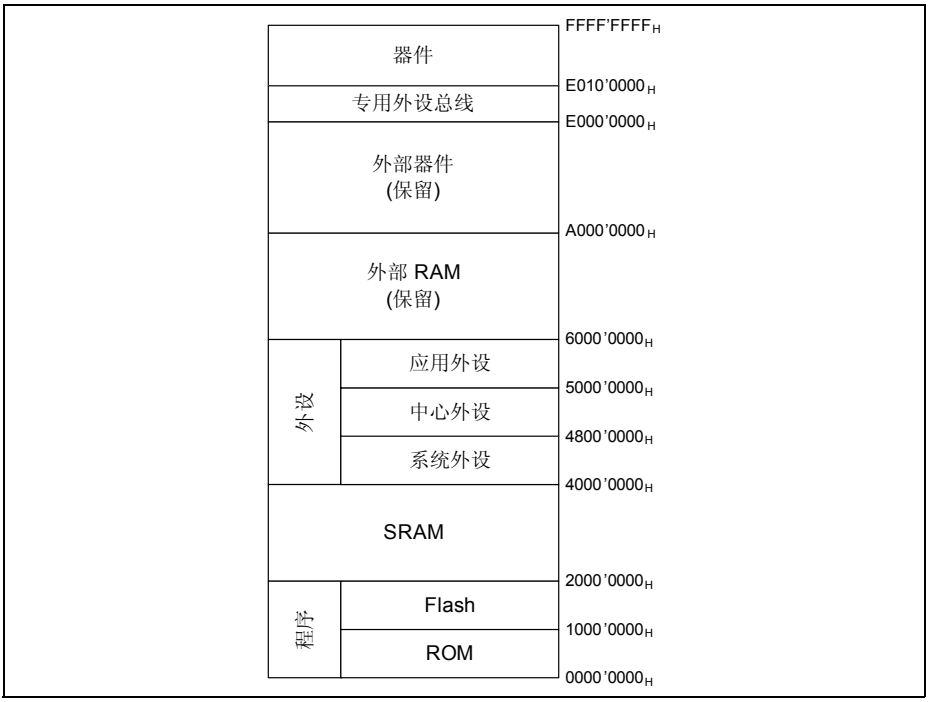


图 8-1 XMC1300 地址空间

表 8-1 存储器映射

地址空间	地址范围	描述	访问类型[1]	
			读	写
Code	00000000_H - $00000AFF_H$	ROM (用户可读)	U, PV	nBE
	$00000B00_H$ - $00001FFF_H$	ROM (用户不可读)	BE	BE
	00002000_H - $0FFFFFFF_H$	保留	BE	BE
	10000000_H - $10000DFF_H$	Flash 扇区 0 (用户不可读)	nBE	nBE
	$10000E00_H$ - $10000FFF_H$	Flash 扇区 0 (用户可读)	U, PV	nBE
	10001000_H - $10032FFF_H$	Flash (200 KB)	U, PV	U, PV
	10033000_H - $1FFFFFFF_H$	保留	BE	BE
SRAM[2]	20000000_H - $20000FFF_H$	SRAM 块 0	U, PV	U, PV
	20001000_H - $20001FFF_H$	SRAM 块 1	U, PV	U, PV
	20002000_H - $20002FFF_H$	SRAM 块 2	U, PV	U, PV
	20003000_H - $20003FFF_H$	SRAM 块 3	U, PV	U, PV
	20004000_H - $3FFFFFFF_H$	保留	BE	BE

表 8-1 存储器映射 (续表)

地址空间	地址范围	描述	访问类型[1]	
			读	写
系统外设	40000000_H - $400007FF_H$	存储器控制	U, PV	U, PV
	40000800_H - $4000FFFF_H$	保留	BE	BE
	40010000_H - $40010FFF_H$	SCU (包括 RTC)	U, PV	U, PV
	40011000_H - $4001107F_H$	ANACTRL	U, PV	U, PV
	40011080_H - $4001FFFF_H$	保留	BE	BE
	40020000_H - $4002001F_H$	WDT	U, PV	U, PV
	40020020_H - $4002FFFF_H$	保留	BE	BE
	40030000_H - $4003003F_H$	MATH 全局寄存器和 DIV	U, PV	U, PV
	40030040_H - $4003007F_H$	MATH CORDIC	U, PV	U, PV
	40030080_H - $4003FFFF_H$	保留	BE	BE

表 8-1 存储器映射 (续表)

地址空间	地址范围	描述	访问类型[1]	
			读	写
系统外设 (续)	40040000_H - $4004007F_H$	端口 0	U, PV	U, PV
	40040080_H - $400400FF_H$	保留	BE	BE
	40040100_H - $4004017F_H$	端口 1	U, PV	U, PV
	40040180_H - $400401FF_H$	保留	BE	BE
	40040200_H - $4004027F_H$	端口 2	U, PV	U, PV
	40040280_H - $4004FFFF_H$	保留	BE	BE
	40050000_H - $400500DF_H$	Flash 寄存器	U, PV	U, PV
	$400500E0_H$ - $47FFFFFF_H$	保留	BE	BE
中心外设	48000000_H - $480001FF_H$	USIC0 通道 0	U, PV	U, PV
	48000200_H - $480003FF_H$	USIC0 通道 1	U, PV	U, PV
	48000400_H - $480007FF_H$	USIC0 RAM	nBE	BE
	48000800_H - $4801FFFF_H$	保留	BE	BE
	48020000_H - $4802000F_H$	PRNG	U, PV	U, PV
	48020010_H - $4802FFFF_H$	保留	BE	BE

表 8-1 存储器映射 (续表)

地址空间	地址范围	描述	访问类型[1]	
			读	写
中心外设 (续)	48030000_H - $480303FF_H$	VADC0 通用寄存器和全局寄存器	U, PV	U, PV
	48030400_H - $480307FF_H$	VADC0 组 0	U, PV	U, PV
	48030800_H - $48030BFF_H$	VADC0 组 1	U, PV	U, PV
	$48030C00_H$ - $48033FFF_H$	保留	BE	BE
	48034000_H - $480341FF_H$	SHS0	U, PV	U, PV
	48034200_H - $4803FFFF_H$	保留	BE	BE
	48040000_H - $480401FF_H$	CCU40 CC40 和内核寄存器	U, PV	U, PV
	48040200_H - $480402FF_H$	CCU40 CC41	U, PV	U, PV
	48040300_H - $480403FF_H$	CCU40 CC42	U, PV	U, PV
	48040400_H - $480404FF_H$	CCU40 CC43	U, PV	U, PV
	48040500_H - $4FFFFFFF_H$	保留	BE	BE

表 8-1 存储器映射 (续表)

地址空间	地址范围	描述	访问类型[1]	
			读	写
应用外设	50000000_H - $500001FF_H$	CCU80 CC80 和内核寄存器	U, PV	U, PV
	50000200_H - $500002FF_H$	CCU80 CC81	U, PV	U, PV
	50000300_H - $500003FF_H$	CCU80 CC81	U, PV	U, PV
	50000400_H - $500004FF_H$	CCU80 CC83	U, PV	U, PV
	50000500_H - $5000FFFF_H$	保留	BE	BE
	50010000_H - $500101FF_H$	POSIF0	U, PV	U, PV
	50010200_H - $5002FFFF_H$	保留	BE	BE
	50030000_H - $500301FF_H$	BCCU0	U, PV	U, PV
	50030200_H - $57FFFFFF_H$	保留	BE	BE
外设	58000000_H - $5FFFFFFF_H$	保留	BE	BE
外部 SRAM	60000000_H - $9FFFFFFF_H$	保留	BE	BE
外部器件	$A0000000_H$ - $DFFFFFFF_H$	保留	BE	BE
专用外设总线	$E0000000_H$ - $E00FFFFF_H$	NVIC、系统定时器、系统控制块	U, PV	U, PV
厂商专用区 1	$E0100000_H$ - $EFFFFFFF_H$	保留	BE	BE
厂商专用区 2	$F0000000_H$ - $F0000FFF_H$	系统 ROM 表	U, PV	nBE
	$F0001000_H$ - $FFFFFFFF_H$	保留	BE	BE

1) 对于外设占用的地址范围，每个地址的访问类型都可能与表中所示不同。详见相应章节。
2) 在器件启动期间，从 $2000'0000_H$ 到 $2000'01FF_H$ 的地址范围会被启动软件盖写。

8.3 存储器访问

本节描述对 XMC1300 中不同类型存储器的访问。

8.3.1 Flash 存储器访问

XMC1300 提供了最多达 200 KB 的 Flash 存储器，用于指令代码或常量数据存储，其地址从 $1000'1000_H$ 开始。这不包括 Flash 扇区 0，该扇区用于存储系统信息，并且是只读类型。

有关 Flash 存储器访问的详细信息，参见 Flash 架构一章。

8.3.2 SRAM 访问

XMC1300 提供 16 KB 的 SRAM，起始地址为 $2000'0000_H$，可用于存储指令代码或常量数据以及系统变量，例如系统堆栈。

SRAM 支持 8 位、16 位 和 32 位写操作，并且为每 8 位写入的数据产生一个奇偶校验位。读操作会对 32 位的读数据进行奇偶错误检查。访问 SRAM 不需要等待状态。

16 KB 的 SRAM 逻辑上被分成 4 块，每块 4 KB。在运行期间，可以用外设特权访问机制使能或禁止对块 1、 2 和 3 的访问。详见 PAU 一章。

注： 从 $2000'0000_H$ 到 $2000'01FF_H$ 的地址范围在器件启动阶段会被启动软件盖写。因此，在引导加载程序向 SRAM 加载代码 / 数据期间，不应将这些地址作为目标地址，这些地址也不应当用于存储在系统复位或软件主复位之后应用程序仍然需要的关键数据。

8.3.3 ROM 访问

XMC1300 提供 8 KB 的 ROM, 它包含启动软件、向量表和用户例程。

对 ROM 的读访问不需要等待状态。

8.4 存储器保护策略

存储器保护要考虑以下两个方面：

1. 知识产权 (IP) 保护
2. 运行时的存储器访问保护

XMC1300 中可用的存储器保护措施列于 表 8-2 中。

表 8-2 存储器保护措施

保护的方面	保护措施	保护目标
IP 保护	封锁未授权的外部访问	Flash 存储器内容
存储器访问保护	位保护机制	特定的系统关键寄存器 / 位域
	外设特权访问控制	特定地址范围；每个地址范围可以被独立控制

8.4.1 知识产权 (IP) 保护

IP 保护是指防止未经授权从 Flash 存储器读取关键数据和用户 IP

8.4.1.1 封锁未经授权的外部访问

在 XMC1300 中，启动模式索引 (BMI) 用于控制启动选项，一旦 BMI 被编程为进入用户模式 (生产模式) ，在没有擦除整个用户 Flash(包括扇区 0) 的情况下，不允许再进入其他启动模式。

这样一来，加载和执行外部代码 （包括可能会读出 Flash 存储器内容的未授权代码）的引导选项将会被封锁，只有来自 Flash 存储器的用户代码才能被执行。

8.4.2 运行时的存储器访问保护

存储器访问保护是指防止在运行时对一个存储器地址空间的意外写访问。

8.4.2.1 位保护机制

位保护机制防止用 SCU 模块中的 PASSWD 寄存器对所选择的寄存器位 (即被保护的位) 进行直接的软件写操作。当位域 MODE 为 11_B 时，向位域 PASS 写 10011_B 将打开对所有被保护位的写访问；向位域 PASS 写 10101_B 将关闭对所有被保护位的写访问。在这两种情况下，即使 PASSWD 寄存器被写入 98_H 或 $A8_H$，位域 MODE 的值也不会改变。只有当位域 PASS 被写入 11000_B 时，位域 MODE 的值才可能被改变。例如，向 PASSWD 寄存器写入 $0000'00C0_H$ 将禁止位保护机制。

如果没有写入 “ 关闭访问 ” 的口令，访问会被打开最多 32 个 MCLK 周期。如果在 32 个 MCLK 周期结束之前重新写入 “ 打开访问 ” 口令，将会重新计数 32 个 MCLK 周期。

表 8-3 列出了 XMC1300 中的保护位。

表 8-3 受保护的寄存器位域一览表

寄存器	位域
SCU_CLKCR	FDIV, IDIV, PCLKSEL, RTCLKSEL
SCU_CGATSET0	所有位
SCU_CGATCLR0	所有位
SCU_ANAOFFSET	ADJL_OFFSET
VADC0_ACCPROT0	所有位
VADC0_ACCPROT1	所有位

在没有通过位域 PASS 打开访问的情况下，对所有被保护的寄存器 (除了 VADC0_ACCPROT[1:0] 之外) 的写操作都会被忽略，不会产生总线错误。用户应该回读寄存器的值，以确保写操作已经发生。

在没有通过位域 PASS 打开访问的情况下，写 VADC0_ACCPROT[1:0] 会触发一次硬故障。如果在访问被打开后立刻发生了一个中断，并且中断服务程序需要大于 32 个 MCLK 的执行时间，则对寄存器的写操作将会在访问被关闭之后发生，从而触发一次硬故障。为了避免这种情况，在写这两个寄存器之前应禁止中断。

SCU 一章中也描述了 PASSWD 寄存器。

SCU_PASSWD
口令寄存器 **(4001 0024$_H$)** **复位值 : 0000 0007$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								PASS					PROTS	MODE	
r								w					rh	rw	

域	位	类型	描述
MODE	[1:0]	rw	**位保护机制控制位** 00_B 位保护机制禁止 - 允许对受保护位的直接访问。 11_B 位保护机制使能 - 必须向位域PASS写入口令才能打开或关闭对保护位的访问。 (默认) 其他：位保护机制使能，类似于设置 MODE = 11_B。 这两位不能直接写入。要想在 11_B 和 00_B 之间改变其值，必须向位域 PASS 写 11000_B。只有这样，MODE 位域才会被写入并保存。
PROTS	2	rh	**位保护信号状态位** 该位指示保护状态。 0_B 软件能写所有被保护的位。 1_B 软件不能写任何被保护的位。
PASS	[7:3]	w	**口令位** 该位保护机制只能识别以下三个口令： 11000_B 使能对位域 MODE 的写操作。 10011_B 打开对所有被保护位的写访问 。 10101_B 关闭对所有被保护位的写访问。
0	[31:8]	r	**保留**

8.4.2.2 外设特权访问控制

所有的 CPU 访问都是特权访问。在 XMC1300 中，提供了一种称为外设特权访问控制的独立机制，该机制允许禁止一个外设的存储器地址空间，以封锁任何意外的写或读访问。必要时可重新使能该地址空间。

详见 PAU 一章。

9 Flash 架构

本章描述非易失性存储器 (NVM) 模块 。NVM 有以下特性：

- 按字读、按块写和按页 (256 字节) 擦除。
- 支持快速个性化 。
- 支持自动校验。
- 每页多达 50,000 个 擦除周期。
- 对于先前从未被编程过的存储单元，可在 25°C 保存数据至少十年。
- 片上产生 Flash 编程电压 。
- 可配置的擦除和写保护。
- 节电休眠模式。
- 无擦除信号量的递增写。
- 具有可用于改善良品率的冗余扇区。

9.1 定义

整篇文档中，术语 NVM (非易失性存储器) 和 Flash 被作为同义词使用，而不考虑这样一个事实：NVM 描述的是更宽泛的存储器类，而 Flash 只是一个特例。

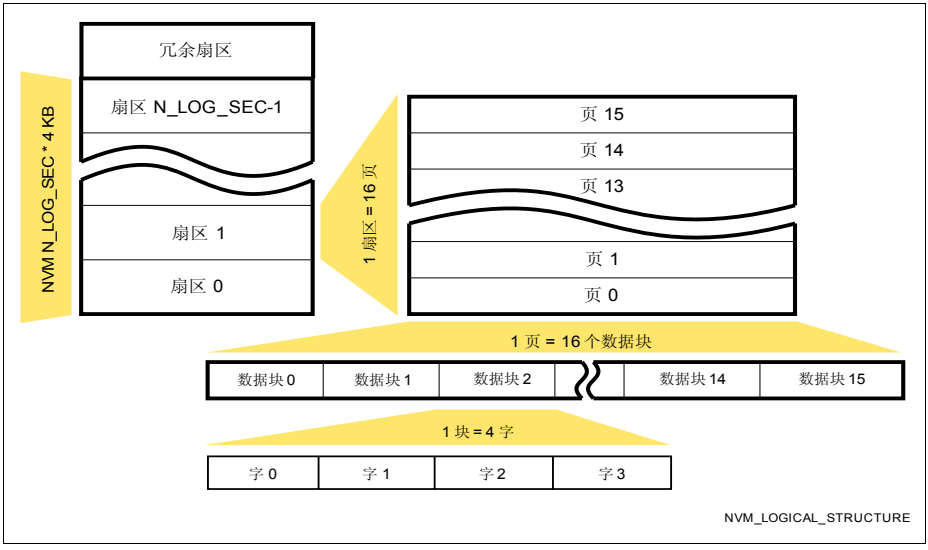


图 9-1 NVM 模块的逻辑结构

9.1.1 逻辑和物理状态

擦除

一个单元的被擦除状态是 ‘1’。强制 NVM 单元到该状态称为擦除。擦除操作的最小粒度是页 (见下文)。

写

一个单元的被写状态是 ‘0’。改变被擦除的单元到该状态称为写。写操作的最小粒度是块 (见下文)。

编程

擦除和写入的组合称为编程。编程通常意味着写先前擦除的页。

注： 写这个词也用于访问特殊功能寄存器，其含义取决于上下文。

9.1.2 数据划分

字

一个字由 32 位组成。字代表在一个访问周期内从 NVM 模块读或者向 NVM 模块写的数据尺寸。

块

一块由 4 个字组成 (128 位数据 , 扩展的 4 位校验位和 6 位 ECC)。块代表可以写入的最小的数据部分。

页

一页由 16 块组成。

扇区

一个扇区由 16 页组成。

9.1.3 地址类型

物理地址

CPU 系统的地址。

基地址或存储器基地址

NVM 模块存储器访问的下边界物理地址。

逻辑地址

NVM 模块内部的存储器地址偏移：如果存储器被寻址，物理地址减去基地址就是逻辑地址。

扇区地址

标明扇区的逻辑地址的模块特定部分，计算方法见 **9.2.1 节**。

页地址

逻辑地址的模块特定部分，计算方法见 **9.2.1 节**。

9.1.4 模块专用定义

下面的表格定义了本节中用到的 NVM 专用常量。

表 9-1 模块特殊定义

模块大小 [KB]	204	
常量名	**值**	**说明**
N_BLOCKS	16	每页的块数
N_PAGES	16	每扇区的页数
N_SECTORS	52	包括冗余在内的扇区数
N_LOG_SEC	51	不包括冗余的扇区数

9.2 模块组件

9.2.1 存储单元阵列

非易失性存储单元是按扇区组织的，扇区由页组成，页由块构成。
逻辑存储器地址 addr 有以下结构：

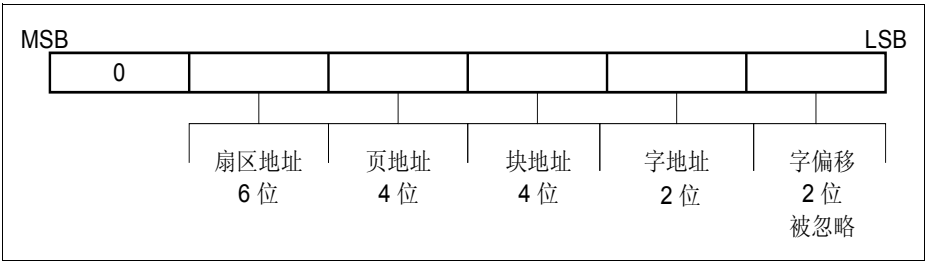


图 9-2 逻辑地址分解

- addr[1:0] = 字偏移，未定义和被忽略的存储器地址
- addr[3:2] = 字地址
- addr[7:4] = 块地址
- addr[11:8] = 页地址
- addr[17:12] = 扇区地址

存储器子系统总是访问整字，因此字偏移被 NVM 模块忽略。

9.2.1.1 页

每页由 16 个数据块组成，每个数据块有 138 位 (包括校验位和 ECC 位)。
页是 NVM 单元阵列中可被擦除的数据粒度。

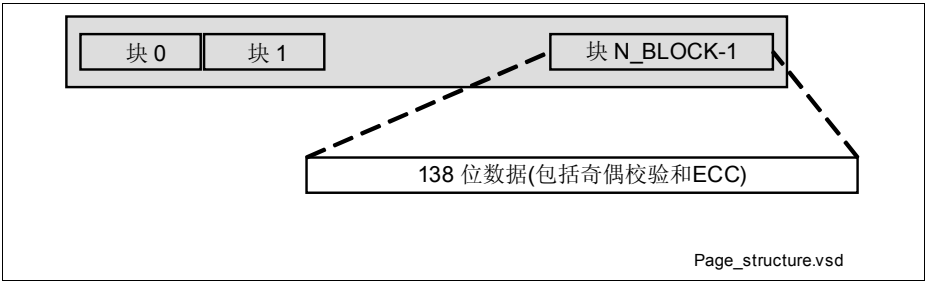


图 9-3 页结构

9.2.1.2 扇区

16 页组成一个扇区。
整个单元阵列由 52 个扇区构成。

9.3 功能描述

NVM 模块支持对存储器和特殊功能寄存器 (SFR) 的读和写访问。不支持对 SFR 的读 - 修改 - 写机制。

NVM 模块的主要任务是从存储阵列读取或向存储器阵列写入。

9.3.1 SFR 访问

在 NVM 模块的每一种模式下都可以读特殊功能寄存器。

当状态机处于忙状态时，不可以写寄存器。在这种情况中，写操作被冻结。其他例外见 **9.5.2 节**。

9.3.2 存储器读

可以按最小粒度，即字来读取 NVM 存储器，前提是该字位于模块的地址范围内。

如果要读的字不在 NVM 模块的存储器地址范围内，则该模块不会有有任何反应，一个不同的存储器模块可能会处理该访问。

当没有 FSM 过程 (擦除、写、校验、休眠或唤醒) 正在进行时，才可以进行存储器读访问。在 FSM 处于忙状态期间，存储器读访问被冻结，直到 FSM 重新变为空闲状态，然后读访问被执行。这样的冻结也会使 CPU 停止执行代码。

如果需要， NVM 会在存储器读期间自动插入等待状态 。

9.3.3 存储器写

从用户的观点看，数据可以直接写入存储器阵列。但在模块内部，是先缓存一块数据并且计算 ECC 位，然后启动一个有限状态机将 ECC 位和数据位写入存储器域。

写操作不支持环绕式处理，写一个块时必须总是从所选块的字 0 开始，然后自动按升序继续写到字 3。

由于写一个块必须按 4 字的写入步骤进行，所以必须遵循特殊的程序来传送完整的块数据到 NVM：写一个块时必须总是从被寻址块的字 0 开始。接下来的三次写入需要按升序来寻址同一块的其他三个字。如果这一规则未被遵守，已经提供的字将会被丢弃。如果在这种情况下最后提供的字地址是字 0，则该字已作为一次新块写过程的第一个字被接受。在该过程中间发生的读操作不会影响写数据的传送。在该过程期间，与被中断的写操作为同一个目标地址的读操作会从存储器阵列返回数据，即不会读回已经传送的写数据。

要写存储阵列时， NVM 模块必须被设置成一次性或连续写模式，该操作通过将 SFR **NVMPROG** 设置为相应的值来实现。现在可以将要写入的数据写入到所期望的地址。如果所寻址的块在此之前被擦除过，则不进行错误检查。由于写操作只能清除数据位而不能将任何数据位置 1，所以写一个未被擦除的块会导致数据被破坏。读这样的块将极有可能导致 ECC 错误。

当 NVM 不处于写模式时，对 NVM 的写访问不会触发异常或中断，数据会丢失。对 NVM 的写访问也用来触发其他操作，这些操作将在下面的小节中描述。类似地，当寻址受保护的扇区时，对 NVM 模块的写访问无效，数据丢失 (见 **9.3.6 节**)。

存储器写操作会执行自动校验，这取决于 **NVMPROG** 的设置 (见 **9.3.5 节**).

在一次性写的情况下，写模式自动结束。在连续写模式下，能继续进行对新地址的写操作，直到用户程序明确结束写模式。连续写操作的目标可以是整个存储器内的所有块，没有任何扇区或页边界限制。

9.3.4 存储器擦除

只能按整页进行物理擦除，即被寻址页的所有位都被物理上置成 '1'。

要擦除存储器域中的一页，必须将 NVM 模块设置为一次性或连续页擦除模式，这可通过将 SFR **NVMPROG** 设置为相应的值来实现。对一个存储单元的写访问指定待擦除页的地址，并触发擦除操作。

当 NVM 不处于页擦除模式时，对 NVM 的写访问不会触发异常或中断，这是因为它们还被用来触发其他操作，如本节所述。类似地，当寻址受保护的扇区时，也不会触发擦除操作 (见 **9.3.6 节**)。

在一次性页擦除情况中，页擦除模式自动结束。在连续页擦除模式，可以继续进行页擦除操作，直到应用程序明确停止页擦除模式。

9.3.5 校验

如 **9.3.3 节**所述，写入的数据能被自动校验。 NVM 自动将写入到单元阵列中的数据与仍然位于模块内部缓冲区的可用数据进行比较。这种比较会用写 hardread 和擦除 hardread 执行两次。这些 hardread 值与正常读的值相比由一定的余量，以确保数据被编程为写和擦除位应具有的值。校验结果可从 SFR **NVMSTATUS** 中读取。
也可以通过将 SFR **NVMPROG** 设置为相应值来启动独立的校验操作。在这种情况下，写入到存储器单元的数据与位于指定地址的存储器域的内容比较。这里的比较只执行一次，与之前所选的来自正常读、 hardread 写以及 hardread 擦除的读标准比较。

当 NVM 不处于校验模式时，对 NVM 的写访问不会触发异常或中断，因为写访问也用于触发其他操作，如本节所述。类似地，当寻址受保护的扇区时，对 NVM 的写访问无效 (见 **9.3.6 节**)。

在一次性校验的情况下，校验模式会自动停止。在连续校验模式，可以继续进行校验操作，直到应用程序明确结束校验模式。

注： 没有自动校验的连续写操作后面跟随两次独立校验：hardread 写和 hardread 读。这比具有连续自动校验的写操作更快。这是因为在第二种情况下，对于每次块写，对 hardread 值的写必须改变两次，而第一种情况下，这种改变对于整个写数据只执行两次。
换句话说，对于连续自动校验，校验所用的参考数据在 NVM 模块内直接可用，而对于独立校验，参考数据需要由 CPU 再次提供。

9.3.6 擦除保护和写保护

通过设置 **NVMCONF**.SECPROT，可以选择对一定数量的扇区进行保护，以防止任何修改，即可将从扇区 0 开始的一个扇区范围定义为受擦除保护和写保护。

9.4 冗余

为了提高产品的良品率， NVM 模块包含冗余功能。冗余对用户来说是透明的。

9.5 省电模式

NVM 模块支持下面提到的省电模式。这些模式在功耗和重启存储器的时间方面都不相同。

9.5.1 空闲模式

在充电泵充电完成并且复位之后，MCU 进入空闲模式。空闲模式的功耗低于存储器读和写访问期间的功耗。

NVM 模块的任何操作 (SFR 访问或存储器访问) 都可以没有延迟地从空闲模式启动。

9.5.2 休眠模式

进入和退出 NVM 休眠模式需要特殊的状态机序列，这需要一些时间。因此，除了 SFR 访问之外， NVM 模块不能在被从休眠模式唤醒后立即使用，这一点由 **NVMSTATUS**.BUSY 指示。

通过 WFE/WFI 指令进入 NVM 休眠模式，这时 Flash 下电被激活，见 SCU 一章。也可通过写 SFR **NVMCONF** 的相应位来触发 NVM 模块的休眠模式。

在 NVM 休眠模式，只有 SFR **NVMCONF** 是可读和可写的，所有其他 SFR 只能读。休眠模式下不能进行存储器访问。

9.6 纠错码 (ECC) 的质和实现

对于每个 128 位的数据块，纠错码 (ECC) 是一个纠错位和两个错误检测位。每个数据块有 4 个校验位，6 个 ECC 位，外加被保护的 128 位数据位。当写数据时，每一块的 ECC 位自动产生。

9.7 NVM SFRs

表 9-3 列出了所有的 NVM 特殊功能寄存器

表 9-2 寄存器地址空间

模块	基地址	结束地址	备注
NVM	$4005\ 0000_H$	$4005\ 00FF_H$	

表 9-3 寄存器一览表

寄存器简称	寄存器全称	偏移地址	页号
NVM SFRs, 寄存器描述			
NVMSTATUS	NVM 状态寄存器	0000_H	页 9-9
NVMPROG	NVM 编程控制寄存器	0004_H	页 9-11
NVMCONF	NVM 配置寄存器	0008_H	页 9-14

寄存器是按字寻址的。

在 NVM 模块处于忙状态期间，可以读 SFR。如果 SFR 的值取决于 NVM 序列的完成，则在读 SFR 之前，必须查询 **NVMSTATUS**.BUSY 为 0_B。否则，读 SFR 可能得到一个尚未更新的值。

9.7.1 寄存器描述

NVM 状态寄存器

NVMSTATUS
NVM 状态寄存器 **(0000_H)** **复位值 : 0002_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									WRPERR	ECC2READ	ECC1READ	VERR		SLEEP	BUSY
r									r	r	r	r		r	r

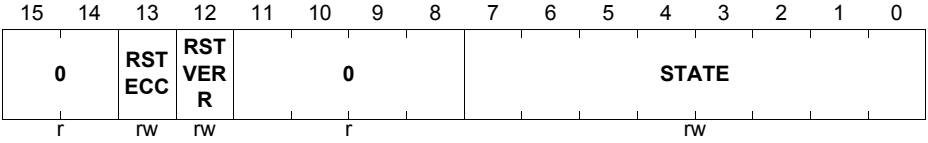
域	位	类型	描述
0	15:7	r	**保留** 读出值为 0。
WRPERR	6	r	**写协议错误** 该标志累积在最后一次四字写操作期间对写协议的违反情况 (对于写或校验)。当一个被触发的操作因为写保护而被忽略时，它也被置位。纠错协议在 **9.3.3 节**中定义。当写 **NVMPROG**.RSTECC 时，它被硬件复位。 0_B **WRPROTOK**，未发生写协议错误。 1_B **WRPROTFAIL**，至少检测到一次写协议错误。
ECC2READ	5	r	**ECC2 读**[1)] 该标志累积存储器读操作期间的 ECC 双位错误。当写 **NVMPROG**.RSTECC 时，它被硬件复位。 0_B **ECC2RDOK**，存储器读操作期间未发生 ECC 双位错误。 1_B **ECC2RDFAIL**，至少检测到一次 ECC 双位错误。
ECC1READ	4	r	**ECC1 读**[1)] 该标志累积最后一次存储器读操作期间的 ECC 一位错误。当写 **NVMPROG**.RSTECC 时，它被硬件复位。 0_B **ECC1RDOK**，未发生 ECC 一位错误。 1_B **ECC1RDFAIL**，至少检测到一次 ECC 一位错误。

域	位	类型	描述
VERR	3:2	r	**校验错误** 当写 **NVMPROG**.RSTVERR 时，该标志被硬件复位。 当进入写模式或只校验模式时，即 **NVMPROG**.ACTION.OPTYPE = 0001_B 或 **NVMPROG**.ACTION.VERIFY = 11_B 时，该标志也被复位。 该标志累积校验错误，即 VERR 每次的更新值都比当前值要大，直到写模式或只校验模式结束 (在一次性写操作中自动结束) 校验过程中错误位数的信息： 00_B **NOFAIL**，无错误位。 01_B **ONEFAIL**，在一个数据块中有一个错误位 。 10_B **TWOFAIL**，在两个不同的数据块中有两个错误位。 11_B **MOREFAIL**，在一个数据块中有两个或更多的错误位，或共有三个或更多的错误位。
SLEEP	1	r	**休眠模式** 0_B **READY**， NVM 不处于休眠模式，没有休眠或唤醒过程正在进行。 1_B **SLEEP**， NVM 处于休眠模式，或因休眠或唤醒过程而处于忙状态。
BUSY	0	r	**忙** 0_B **READY**， NVM 不忙。可以进行对存储器单元阵列的读访问和对寄存器的写访问。 1_B **BUSY**， NVM 忙。不能进行存储器读和寄存器写访问。

1) 如果从包含两个或更多奇偶校验错误的存储区域读一个数据块，ECC1 和 ECC2 错误标志可能同时被置位。

NVM 编程控制寄存器

NVMPROG
NVM 编程控制寄存器 **(0004_H)** **复位值 : 0000_H**



域	位	类型	描述
0	15:14	r	**保留** 读出值为 0； 应写入 0。
RSTECC	13	rw	**复位 ECC** 只能通过软件置位，由硬件自动复位。 0_B **NOP**，无动作。 1_B **RESET**，复位 **NVMSTATUS**.ECCxREAD 和 NVMSTATUS.WRPERR。
RSTVERR	12	rw	**复位校验错误** 只能通过软件置位，由硬件自动复位。 0_B **NOP**，无动作。 1_B **RESET**，复位 **NVMSTATUS**.VERR。
0	11:8	r	**保留** 读出值为 0； 应写入 0。

域	位	类型	描述
ACTION	7:0	rw	**ACTION: [VERIFY, ONE_SHOT, OPTYPE]** 该位域选择擦除、写或校验操作。见 **ACTION 的更多详情**。 ACTION 由三个位域拼接而成：ACTION[7:6] = VERIFY，ACTION[5:4] = ONE_SHOT 和 ACTION[3:0] = OPTYPE。 **OPTYPE** 定义如下操作： 0000_B: 空闲或只校验，这取决于 VERIFY 的设置； 0001_B: 写； 0010_B: 页擦除。 **ONE_SHOT** 是 OPTYPE 的一个参数，取值如下： 01_B: 一次 10_B: 连续。 在 01_B 的情况下，操作执行完后， ACTION 自动复位到空闲模式。 **VERIFY 定义** OPTYPE 的第二个参数： 01_B: 每次写操作后用 hardread 标准校验所写数据。 10_B: 无校验 , 11_B: 对阵列内容进行校验。 定义了以下操作，其他值当做 00_H 处理 00_H ，空闲状态，没有动作被触发。写 00_H 退出当前模式。 51_H ，启动带自动校验的一次性写操作。 91_H ，启动不带校验的一次性写操作。 61_H ，启动对每次写进行自动校验的连续写操作。 $A1_H$ ，启动不带校验的连续写操作。 92_H ，启动一次性页擦除操作。 $A2_H$ ，启动连续页擦除操作。 $D0_H$ ，启动一次性只校验：所写数据与阵列内容比较。 $E0_H$ ，启动连续只校验：所写数据与阵列内容比较。

ACTION 的更多详情

当对 NVM 地址范围执行一次写操作时，ACTION 所选择的操作被执行。在一次性操作的情况下， ACTION 被自动复位。

校验结果可以从 **NVMSTATUS**.VERR 读取。

当 ACTION 的当前值为 00_H 时，它才能被改变，否则 ACTION 要置为 00_H。一旦 ACTION 不处于闲置状态， ACTION 就可以被重新写入其当前值；任何其他值会导致 ACTION 复位。

只有写操作能被自动校验。页擦除操作不能使用自动校验，它们必须通过设置 ACTION.VERIFY = 10_B 来启动。

如果要校验当前擦除的块或页，必须用 ACTION.VERIFY = 11_B 来启动一个单独的序列。

校验操作要求提供一个完整块的数据并且包含 ECC 位。当写数据时， 每个块的 ECC 位自动生成。

通过 ACTION.VERIFY = 01_B 启动的校验操作伴随着 hardread 写和 hardread 擦除标准而自动执行。
通过 ACTION.VERIFY = 11_B 启动的校验操作伴随着单一 hardread 标准而执行。单一 hardread 标准在启动时通过 **NVMCONF**.HRLEV 定义。

只有在执行了地址（和块数据）传送之后，通过设置 ACTION 启动的一个序列才会置位 **NVMSTATUS**.BUSY。可以通过查询 NVMSTATUS.BUSY 或等待一个 NVM 中断的方式检测该序列的结束。

进入休眠模式会复位 ACTION。

NVM 配置寄存器

NVMCONF 是唯一一个在模块处于休眠模式时能写的 SFR。这对用 NVM_ON = 0_B 使能休眠模式进入和用 NVM_ON = 1_B 重新唤醒是必须的。

NVMCONF
NVM 配置寄存器 **(0008_H)** **复位值 : 9000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NVM_ON	INT_ON	0	1	SECPROT								0	HRELV		0
rw	rw	rw	rw	rw								r	rw		rw

域	位	类型	描述
NVM_ON	15	rw	**NVM 开** 当被清零时，不能从 NVM 执行任何软件代码，直到它被重新置位。也就是说，用来改变 NVM_ON 本身的软件代码不能位于 NVM 中，否则软件将会永远停止执行。 0_B **SLEEP**， NVM 被切换到或保持在休眠模式。 1_B **NORM**， NVM 被切换到或保持在正常模式。
INT_ON	14	rw	**中断开** 当被使能时，通过设置 **NVMPROG** 启动的一个序列 (写、擦除或 只校验序列) 如果完成，会产生 NVM 中断。这同样适用于唤醒序列。 0_B **INTOFF**，不产生 NVM 就绪中断。 1_B **INTON**，产生 NVM 就绪中断。
0	13	rw	**保留为将来使用** 必须写 0，以允许正确操作。
1	12	rw	**保留为将来使用** 必须写 1，以允许正确操作。
SECPROT	11:4	rw	**扇区保护[1)]** 该域定义写、擦除、校验被保护的扇区数，从物理扇区 0 开始。
0	3	r	**保留** 读出值为 0； 应当写入 0。

域	位	类型	描述
HRLEV	2:1	rw	**Hardread 标准**[2)] 通过 **NVMPROG**.ACTION.VERIFY = 11_B 为校验定义单一 hardread 标准： 00_B **NR**，正常读 01_B **HRW**， Hardread 写 10_B **HRE**， Hardread 擦除 11_B **RFU**，保留为将来使用
0	0	rw	**保留为将来使用** 必须写 0，以允许正确操作。

1) 对于 SECPROT > 0，SECPROT 定义被保护的扇区数。扇区 0 到 SECPROT-1 不能被写、擦除或校验。模块接受对所有被保护扇区的写操作，但这些写操作在内部被忽略。

2) HRLEV 定义由 NVMPROG.ACTION.VERIFY = 11_B 启动的独立校验序列的 hardread 标准。该 hardread 标准在校验序列结束之前一直使用。在此期间 HRLEV 不能改变。

9.8 序列示例

本节介绍几个底层编程实例。

在下述所有所有操作中，需要考虑由 **NVMCONF**.SECPROT 定义的保护。

9.8.1 存储器写

9.8.1.1 写单个块

该序列要求目标块已经被擦除。

另外还假设：**NVMPROG**.ACTION = 00_H。

1. 启动一次性写操作：
   根据是否对所写数据执行自动校验，分别写 **NVMPROG**.ACTION = 51_H，或 **NVMPROG**.ACTION = 91_H。
2. 写一个块的数据到物理地址。
3. 查询 **NVMSTATUS**.BUSY 标志，直到写序列结束，或等待 NVM 就绪中断 (如果被使能)。
4. 如果在第一步请求了对所写数据进行自动校验，则读 **NVMSTATUS**.VERR 可得到校验结果。

如果在第一步未请求对所写数据进行校验，则第四步也就不需要对 NVM SFR 进行读访问，因此第三步可省略。

对 NVM 模块的下一次访问或对一个 NVM SFR(不论哪一个访问在先) 的下一次写访问将会自动停止，直到第二步操作完成。

9.8.1.2 写多个块

该序列要求目标块已经被擦除。

另外还假设：**NVMPROG**.ACTION = 00_H.

1. 启动连续写操作：
   根据是否对数据执行自动校验，分别写 **NVMPROG**.ACTION = 61_H，或 **NVMPROG**.ACTION = $A1_H$。
2. 写一个块的数据到物理地址。
3. 查询 **NVMSTATUS**.BUSY 标志，直到写序列结束，或等待 NVM 就绪中断 (如果被使能)。
   可以选择省略第三步：对 NVM 模块的下一次访问或对一个 NVM SFR(不论哪一个在先) 的下一次写访问将会自动停止，直到第二步的操作完成。
4. 跳到第二步，除非所有数据都已写完。
5. 停止连续写操作：
   写 **NVMPROG**.ACTION = 00_H。
6. 如果在第一步请求了对写数据进行自动校验，则读 **NVMSTATUS**.VERR 可得到校验结果。

9.8.2 擦除存储器

9.8.2.1 擦除单个页

假设：**NVMPROG**.ACTION = 00_H。

1. 启动一次性页擦除操作：
   写 **NVMPROG**.ACTION = 92_H。
2. 写一个虚拟数据字到待擦除页的任意一个物理地址。
3. 查询 **NVMSTATUS**.BUSY 标志，直到页擦除序列完成，或等待 NVM 就绪中断 (如果被使能)。

可选择省略第三步：对 NVM 的下一次访问或对 NVM 一个 SFR(不论哪一个在先) 的下一次写访问将会自动停止，直到第二步的操作完成。

9.8.2.2 擦除多个页

假设：**NVMPROG**.ACTION = 00_H。

1. 启动一个连续页擦除操作：
   写 **NVMPROG**.ACTION = $A2_H$。
2. 写一个字的虚拟数据到被擦除的页的任意一个物理地址。
3. 轮询 **NVMSTATUS**.BUSY 标志，直到页擦除时序完成或等待 NVM 准备中断 (如果使能)。
4. 跳到第二阶段，除非所有页被擦除。

5. 停止连续页擦除操作：
   写 **NVMPROG**.ACTION = 00_H。

9.8.3 校验存储器

9.8.3.1 校验单个块

假设：**NVMPROG**.ACTION = 00_H。

1. 通过设置 NVMCONF.HRLEV 选择所希望的 hardread 标准 。
2. 启动一次性校验操作：
   写 **NVMPROG**.ACTION = $D0_H$。
3. 写一个块的参考数据到物理地址。
4. 查询 **NVMSTATUS**.BUSY 标志，直到校验序列完成，或等待 NVM 就绪中断 (如果被使能)。
5. 读 **NVMSTATUS**.VERR ，得到校验结果。

9.8.3.2 校验多个块

假设：**NVMPROG**.ACTION = 00_H。

1. 通过设置 NVMCONF.HRLEV 选择所希望的硬读标准。
2. 启动连续校验操作：
   写 **NVMPROG**.ACTION = $E0_H$。
3. 写一个块的参考数据到物理地址。
4. 查询 **NVMSTATUS**.BUSY 标志，直到校验序列完成，或等待 NVM 就绪中断 (如果被使能)。
   可选择省略第四步：对 NVM 的下一次访问或对一个 NVM SFR(不论哪一个在先) 的下一次写访问将会自动停止，直到第三步的操作完成。
5. 跳到第三步，除非所有块都被校验完。
6. 停止连续校验操作：
   写 **NVMPROG**.ACTION = 00H。
7. 读 **NVMSTATUS**.VERR ，得到校验结果。

9. 8. 4 写一个已写过的块

一般来说，写额外的位到一个已经写过的块是不可行的。因为已经写入的 ECC 位和校验位也需要更新，在大多数情况下需要擦除一些位，而这是不可能的。其结果将会产生一个 ECC 错误块。

出于特殊目的，重复更新特殊结构的数据是可能的。例如存储在掉电恢复软件实现中要使用的标识位。

从一个被擦除块开始，使用特殊结构的 ECC，重复增加两个新写位到块中任意两个字的相同位是可能的。其原理由**表 9-4** 表中的示范写方案展示，在该例中，一个块被更新了 32 次而未发生 ECC 错误，即数据在整个过程中都受 ECC 保护。

表 9-4 特殊结构数据块的增量更新

操作	块写数据	结果块内容 [1]
擦除块		FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFF
增加第 1 个值	FFFFFFFF FFFFFFFC FFFFFFFF FFFFFFFC	FFFFFFFF FFFFFFFC FFFFFFFF FFFFFFFC
增加第 2 个值	FFFFFFFF FFFFFFF3 FFFFFFFF FFFFFFF3	FFFFFFFF FFFFFFF0 FFFFFFFF FFFFFFF0
增加第 3 个值	FFFFFFFF FFFFFFCF FFFFFFFF FFFFFFCF	FFFFFFFF FFFFFFC0 FFFFFFFF FFFFFFC0
增加第 4 个值	FFFFFFFF FFFFFF3F FFFFFFFF FFFFFF3F	FFFFFFFF FFFFFF00 FFFFFFFF FFFFFF00
增加第 5 个值	FFFFFFFF FFFFFCFF FFFFFFFF FFFFFCFF	FFFFFFFF FFFFFC00 FFFFFFFF FFFFFC00
增加第 6 个值	FFFFFFFF FFFFF3FF FFFFFFFF FFFFF3FF	FFFFFFFF FFFFF000 FFFFFFFF FFFFF000
...	...	...
增加第 15 个值	FFFFFFFF CFFFFFFF FFFFFFFF CFFFFFFF	FFFFFFFF C0000000 FFFFFFFF C0000000
增加第 16 个值	FFFFFFFF 3FFFFFFF FFFFFFFF 3FFFFFFF	FFFFFFFF 00000000 FFFFFFFF 00000000
增加第 17 个值	FFFFFFFC FFFFFFFF FFFFFFFC FFFFFFFF	FFFFFFFC 00000000 FFFFFFFC 00000000
...	...	...
增加第 30 个值	F3FFFFFF FFFFFFFF F3FFFFFF FFFFFFFF	F0000000 00000000 F0000000 00000000
增加第 31 个值	CFFFFFFF FFFFFFFF CFFFFFFF FFFFFFFF	C0000000 00000000 C0000000 00000000
增加第 32 个值	3FFFFFFF FFFFFFFF 3FFFFFFF FFFFFFFF	00000000 00000000 00000000 00000000
增加到无效写入值 [2]	FFFFFFFF FFFFFFFF FFFFFFFF FFFFFFFE	同前，但发生 ECC2 错误

1) ECC- 清除和校验清除 (ECC 位和校验位都保持被擦除状态)，除非另外说明。

2) 该写数据可以在上述任何状态下写（除了完全擦除状态以外），总是会导致不变的数据内容，但现在产生了 ECC2 错误 (写入三个 ECC 位和一个校验位)。

其他写入值组合也是可能的，只要遵守 “ 在每两个字中增加两个相同的位 ” 这个基本条件。**表 9-4** 也显示了通过故意制造一个 ECC2 错误而造成数据无效也是可能的。

为了最小化写时间和最小化块的加载周期，如**表 9-4** 所示那样只写数据块的新位是很重要的。在原理上，重复写哪些已经过的位是可能的，但是这会增加不必要的写时间，而且还会增加加载周期。

9.8.5 休眠模式

假设：活动模式 (**NVMSTATUS**.SLEEP = 0_B) 和 **NVMSTATUS**.BUSY = 0_B。

进入休眠模式

1. 执行 WFE/WFI 或 **NVMCONF**.NVM_ON = 0_B。
2. NVMSTATUS.BUSY = 1B 和 NVMSTATUS.SLEEP = 1_B，直到进入休眠模式。
3. 在休眠模式期间，NVMSTATUS.BUSY = 0B 和 NVMSTATUS.SLEEP = 1_B 。

从休眠模式唤醒

1. 任何唤醒事件和 NVMCONF.NVM_ON = 1_B。
2. NVMSTATUS.BUSY = 1_B 和 NVMSTATUS.SLEEP = 1_B，直到进入活动模式。
3. 在活动模式期间，NVMSTATUS.BUSY = 0_B 和 NVMSTATUS.SLEEP = 0_B 。

在进入休眠模式的过程还没有完成时发生的唤醒事件并不缩短这一过程，而只是在休眠后直接启动唤醒模式过程。

9.8.6 时序

下面的状态图展示了所有可能序列的状态变迁和时序。

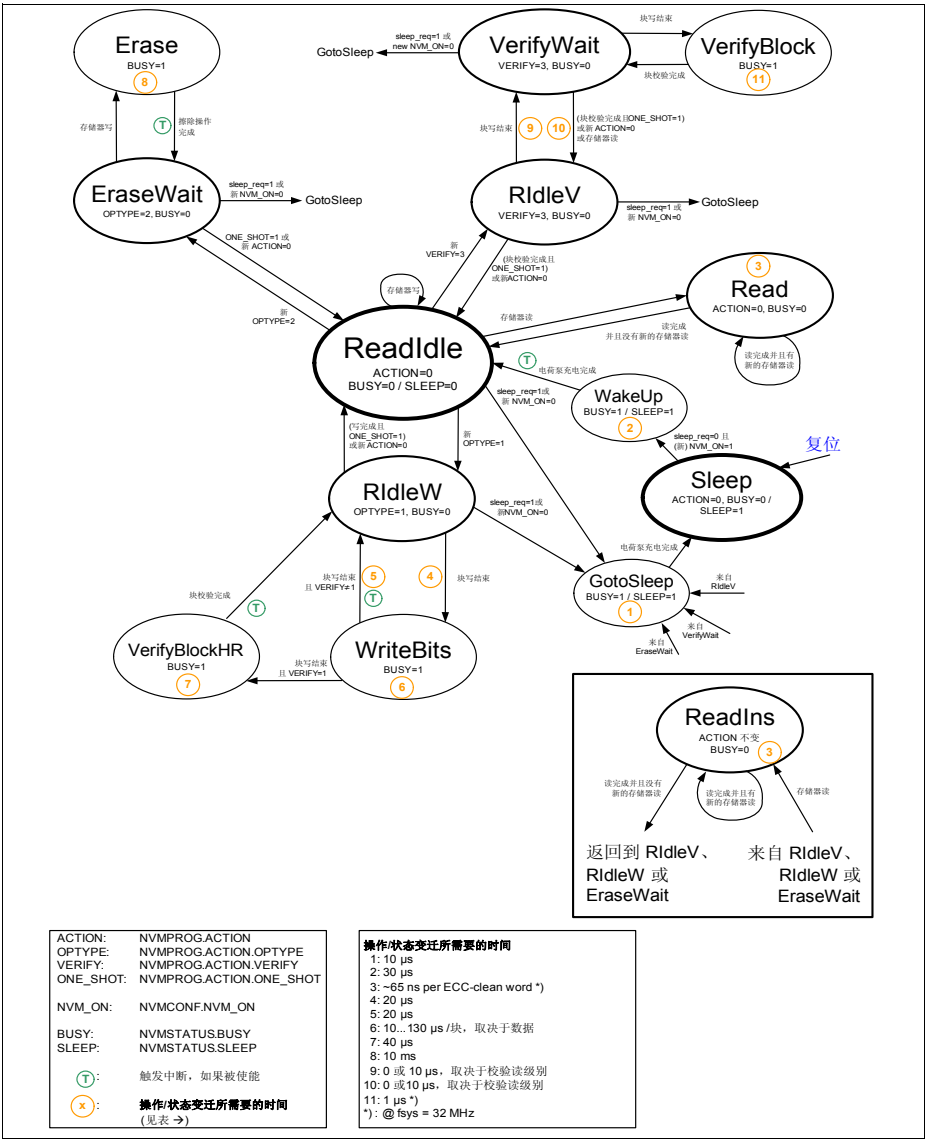


图 9-4 NVM 模块时序状态图

10 外设访问单元 (PAU)

外设访问单元 (PAU) 支持存储器和外设的集中访问控制。

10.1 特性

PAU 具有以下特性：

- 允许用户应用程序使能 / 禁止对外设寄存器的访问
- 当发生对被禁止的或未分配的地址单元的访问时，产生一个硬故障异常
- 提供外设可用性信息和存储器大小信息

10.2 外设特权访问控制

用户应用可以使用外设特权访问寄存器 PRIVDISn 来禁止对一个外设的访问。当与该外设相对应的 PDISx 位被置 1 时，映射到该外设的存储器地址空间变为无效。对这样的一个无效地址的访问会导致硬故障异常。

应用程序能清除这个相同的位来再次使能对外设的访问。

表 10-1 列出了外设和分配给它们的 PDISx 位。没有 PDISx 位的外设在任何时刻都可以访问。

表 10-1 外设可用性和特权访问控制

外设	地址组	AVAILn.AVAILx 位	PRIVDIS.PDISx 位
Flash	Flash SFR	-	PRIVDIS0.2
SRAM	RAM 块 1	AVAIL0.5	PRIVDIS0.5
	RAM 块 2	AVAIL0.6	PRIVDIS0.6
	RAM 块 3	AVAIL0.7	PRIVDIS0.7
WDT	WDT	-	PRIVDIS0.19
MATH	MATH 全局 SFR 和 DIV	AVAIL0.20	PRIVDIS0.20
	MATH CORDIC	AVAIL0.21	PRIVDIS0.21
Ports	端口 0	AVAIL0.22	PRIVDIS0.22
	端口 1	AVAIL0.23	PRIVDIS0.23
	端口 2	AVAIL0.24	PRIVDIS0.24
USIC0	USIC0_CH0	AVAIL1.0	PRIVDIS1.0
	USIC0_CH1	AVAIL1.1	PRIVDIS1.1
PRNG	PRNG	AVAIL1.4	-

表 10-1 外设可用性和特权访问控制

外设	地址组	AVAILn.AVAILx 位	PRIVDIS.PDISx 位
VADC0	VADC0 基本 SFR	AVAIL1.5	PRIVDIS1.5
	VADC0 组 0 SFR	AVAIL1.6	PRIVDIS1.6
	VADC0 组 1 SFR	AVAIL1.7	PRIVDIS1.7
SHS0	SHS0	AVAIL1.8	PRIVDIS1.8
CCU4	CC40 和 CCU40 内核 SFR	AVAIL1.9	PRIVDIS1.9
	CC41	AVAIL1.10	PRIVDIS1.10
	CC42	AVAIL1.11	PRIVDIS1.11
	CC43	AVAIL1.12	PRIVDIS1.12
CCU80	CC80 和 CCU80 内核 SFR	AVAIL2.0	PRIVDIS2.0
	CC81	AVAIL2.1	PRIVDIS2.1
	CC82	AVAIL2.2	PRIVDIS2.2
	CC83	AVAIL2.3	PRIVDIS2.3
POSIF0	POSIF0	AVAIL2.12	PRIVDIS2.12
BCCU0	BCCU0	AVAIL2.15	PRIVDIS2.15

10.3 外设可用性和存储器大小

外设的可用性和存储器大小是随产品型号不同而变化的。用户应用可以通过读 外设可用性寄存器 AVAILn 来检查一个具体型号的可用外设。参见 **表 10-1** 关于位的分配。类似地，存储器大小寄存器 (例如，针对 Flash 存储器的 FLSIZE 寄存器) 可用来检查可用存储器的大小。

10.4 PAU 寄存器

寄存器概述

绝对寄存器地址可以通过下面的加法计算：
模块基地址 + 偏移地址

表 10-2 寄存器地址空间

模块	基地址	结束地址	备注
PAU	$4000\ 0000_H$	$4000\ FFFF_H$	

表 10-3 寄存器概述

简称	描述	偏移地址	访问模式		描述见
			读	写	
保留	保留	0000_H - $003C_H$	nBE	nBE	
AVAIL0	外设可用性寄存器 0	0040_H	U, PV	BE	页 10-7
AVAIL1	外设可用性 寄存器 1	0044_H	U, PV	BE	页 10-8
AVAIL2	外设可用性寄存器 2	0048_H	U, PV	BE	页 10-10
保留	保留	$004C_H$ - $007C_H$	nBE	nBE	
PRIVDIS0	外设特权访问寄存器 0	0080_H	U, PV	U, PV	页 10-3
PRIVDIS1	外设特权访问寄存器 1	0084_H	U, PV	U, PV	页 10-5
PRIVDIS2	外设特权访问寄存器 2	0088_H	U, PV	U, PV	页 10-6
保留	保留	$008C_H$ - $03FC_H$	nBE	nBE	
ROMSIZE	ROM 大小寄存器	0400_H	U, PV	BE	页 10-11
FLSIZE	Flash 大小寄存器	0404_H	U, PV	BE	页 10-11
RAM0SIZE	RAM0 大小寄存器	0410_H	U, PV	BE	页 10-12

10.4.1 外设特权访问寄存器 (PRIVDISn)

PRIVDISn 寄存器中的位可以在运行期间使能和禁止对外设的访问。当一个外设被禁止时，对该外设的访问会产生总线错误。

PRIVDIS0
外设特权访问寄存器 0 **(0080_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							PDIS24	PDIS23	PDIS22	PDIS21	PDIS20	PDIS19	0		
r							rw	rw	rw	rw	rw	rw	r		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								PDIS7	PDIS6	PDIS5	0		PDIS2	0	
r								rw	rw	rw	r		rw	r	

域	位	类型	描述
PDIS2	2	rw	**Flash SFR 特权禁止标志** 0_B 可以访问 Flash SFR。 1_B 不能访问 Flash SFR。
PDIS5	5	rw	**RAM 块 1 特权禁止标志** 0_B 可以访问 RAM 块 1。 1_B 不能访问 RAM 块 1。
PDIS6	6	rw	**RAM 块 2 特权禁止标志** 0_B 可以访问 RAM 块 2。 1_B 不能访问 RAM 块 2。
PDIS7	7	rw	**RAM 块 3 特权禁止标志** 0_B 可以访问 RAM 块 3。 1_B 不能访问 RAM 块 3。
PDIS19	19	rw	**WDT 特权禁止标志** 0_B 可以访问 WDT。 1_B 不能访问 WDT。
PDIS20	20	rw	**MATH 全局 SFR 和除法器特权禁止标志** 0_B 可以访问 MATH 全局 SFR 和除法器。 1_B 不能访问 MATH 全局 SFR 和除法器。
PDIS21	21	rw	**MATH CORDIC 特权禁止标志** 0_B 可以访问 MATH CORDIC。 1_B 不能访问 MATH CORDIC。
PDIS22	22	rw	**端口 0 特权禁止标志** 0_B 可以访问端口 0 。 1_B 不能访问端口 0。
PDIS23	23	rw	**端口 1 特权禁止标志** 0_B 可以访问端口 1。 1_B 不能访问端口 1。
PDIS24	24	rw	**端口 2 特权禁止标志** 0_B 可以访问端口 2。 1_B 不能访问端口 2。
0	[31:25], [18:8], [4:3], [1:0]	r	**保留**

PRIVDIS1
外设特权访问寄存器 1 **(0084$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**PDIS 12**	**PDIS 11**	**PDIS 10**	**PDIS 9**	**PDIS 8**	**PDIS 7**	**PDIS 6**	**PDIS 5**	**0**			**PDIS 1**	**PDIS 0**
r			rw	rw	rw	rw	rw	rw	rw	rw	r			rw	rw

域	位	类型	描述
PDIS0	0	rw	**USIC0 通道 0 特权禁止标志** 0_B 可以访问 USIC0 通道 0。 1_B 不能访问 USIC0 道 0。
PDIS1	1	rw	**USIC0 通道 1 特权禁止标志** 0_B 可以访问 USIC0 通道 1。 1_B 不能访问 USIC0 通道 1。
PDIS5	5	rw	**VADC0 基本 SFR 特权外设禁止标志** 0_B 可以访问 VADC0 基本 SFR。 1_B 不能访问 VADC0 基本 SFR。
PDIS6	6	rw	**VADC0 组 0 SFR 特权禁止标志** 0_B 可以访问 VADC0 组 0 SFR。 1_B 不能访问 VADC0 组 0 SFR。
PDIS7	7	rw	**VADC0 组 1 SFR 特权禁止标志** 0_B 可以访问 VADC0 组 1 SFR。 1_B 不能访问 VADC0 组 1 SFR。
PDIS8	8	rw	**SHS0 特权禁止标志** 0_B 可以访问 SHS0。 1_B 不能访问 SHS0。
PDIS9	9	rw	**CC40 和 CCU40 内核 SFR 特权禁止标志** 0_B 可以访问 CC40 和 CCU40 内核 SFR。 1_B 不能访问 CC40 和 CCU40 内核 SFR。
PDIS10	10	rw	**CC41 特权禁止标志** 0_B 可以访问 CC41。 1_B 不能访问 CC41。

域	位	类型	描述
PDIS11	11	rw	**CC42 特权禁止标志** 0_B 可以访问 CC42。 1_B 不能访问 CC42。
PDIS12	12	rw	**CC43 特权禁止标志** 0_B 可以访问 CC43。 1_B 不能访问 CC43。
0	[31:13], [4:2]	r	**保留**

PRIVDIS2
外设特权访问寄存器 2 **(0088_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDIS 15	**0**		**PDIS 12**	**0**								**PDIS 3**	**PDIS 2**	**PDIS 1**	**PDIS 0**
rw	r		rw	r								rw	rw	rw	rw

名称	位	类型	描述
PDIS0	0	rw	**CC80 和 CCU80 内核 SFR 特权禁止标志** 0_B 可以访问 CC80 和 CCU80 内核 SFR。 1_B 不能访问 CC80 和 CCU80 内核 SFR。
PDIS1	1	rw	**CC81 特权禁止标志** 0_B 可以访问 CC81。 1_B 不能访问 CC81。
PDIS2	2	rw	**CC82 特权禁止标志** 0_B 可以访问 CC82。 1_B 不能访问 CC82。
PDIS3	3	rw	**CC83 特权禁止标志** 0_B 可以访问 CC83。 1_B 不能访问 CC83。

名称	位	类型	描述
PDIS12	12	rw	**POSIF0 特权禁止标志** 0_B 可以访问 POSIF0。 1_B 不能访问 POSIF0。
PDIS15	15	rw	**BCCU0 特权禁止标志** 0_B 可以访问 BCCU0。 1_B 不能访问 BCCU0。
0	[31:16], [14:13], [11:4]	r	**保留**

10.4.2 外设可用性寄存器 (AVAILn)

AVAILn 寄存器指示具体型号器件的可用外设。

注： AVAILn *寄存器的复位值显示所有可用外设的配置。实际值可能会因为产品型号不同而不同。*

AVAIL0
外设可用性寄存器 0 **(0040_H)** **复位值 : 01FF 00FF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							**AVAIL24**	**AVAIL23**	**AVAIL22**	**AVAIL21**	**AVAIL20**	**1**			
r							r	r	r	r	r	r			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								**AVAIL7**	**AVAIL6**	**AVAIL5**	**1**				
r								r	r	r	r				

域	位	类型	描述
AVAIL5	5	r	**RAM 块 1 可用性标志** 0_B RAM 块 1 不可用。 1_B RAM 块 1 可用。
AVAIL6	6	r	**RAM 块 2 可用性标志** 0_B RAM 块 2 不可用。 1_B RAM 块 2 可用。

域	位	类型	描述
AVAIL7	7	r	**RAM 块 3 可用性标志** 0_B RAM 块 3 不可用。 1_B RAM 块 3 可用。
AVAIL20	20	r	**MATH 全局 SFR 和除法器可用性标志** 0_B MATH 全局 SFR 和除法器不可用。 1_B MATH 全局 SFR 和除法器可用。
AVAIL21	21	r	**MATH CORDIC 可用性标志** 0_B MATH CORDIC 不可用。 1_B MATH CORDIC 可用。
AVAIL22	22	r	**端口 0 可用性标志** 0_B 端口 0 不可用。 1_B 端口 0 可用。
AVAIL23	23	r	**端口 1 可用性标志** 0_B 端口 1 不可用。 1_B 端口 1 可用。
AVAIL24	24	r	**端口 2 可用性标志** 0_B 端口 2 不可用。 1_B 端口 2 可用。
1	[19:16], [4:0]	r	**保留**
0	[31:25], [15:8]	r	**保留**

AVAIL1
外设可用性寄存器 1 **(0044_H)** **复位值 : 0000 1FF7$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**AVAIL12**	**AVAIL11**	**AVAIL10**	**AVAIL9**	**AVAIL8**	**AVAIL7**	**AVAIL6**	**AVAIL5**	**AVAIL4**	**0**	**1**	**AVAIL1**	**AVAIL0**
r			r	r	r	r	r	r	r	r	r	r	r	r	r

域	位	类型	描述
AVAIL0	0	r	**USIC0 通道 0 可用性标志** 0_B USIC0 通道 0 不可用。 1_B USIC0 通道 0 可用。
AVAIL1	1	r	**USIC0 通道 1 用性标志** 0_B USIC0 通道 1 不可用。 1_B USIC0 通道 1 可用。
AVAIL4	4	r	**PRNG 可用性标志** 0_B PRNG 不可用。 1_B PRNG 可用的。
AVAIL5	5	r	**VADC0 基本 SFR 可用性标志** 0_B VADC0 基本 SFR 不可用。 1_B VADC0 基本 SFR 可用。
AVAIL6	6	r	**VADC0 组 0 SFR 可用性标志** 0_B VADC0 组 0 SFR 不可用。 1_B VADC0 组 0 SFR 可用。
AVAIL7	7	r	**VADC0 组 1 SFR 可用性标志** 0_B VADC0 组 1 SFR 不可用。 1_B VADC0 组 1 SFR 可用。
AVAIL8	8	r	**SHS0 可用性标志** 0_B SHS0 不可用。 1_B SHS0 可用。
AVAIL9	9	r	**CC40 可用性标志** 0_B CC40 不可用。 1_B CC40 可用。
AVAIL10	10	r	**CC41 可用性标志** 0_B CC41 不可用。 1_B CC41 可用。
AVAIL11	11	r	**CC42 可用性标志** 0_B CC42 不可用。 1_B CC42 可用。
AVAIL12	12	r	**CC43 可用性标志** 0_B CC43 不可用。 1_B CC43 可用。
1	2	r	**保留**
0	[31:13], 3	r	**保留**

AVAIL2
外设可用性寄存器 2 **(0048_H)** **复位值 : 0000 900F$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AVAIL15	0		AVAIL12	0								AVAIL3	AVAIL2	AVAIL1	AVAIL0
r	r		r	r								r	r	r	r

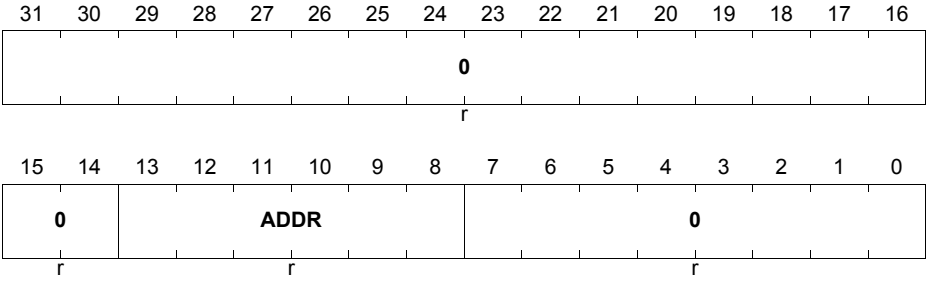
名称	位	类型	描述
AVAIL0	0	r	**CC80 和 CCU80 内核 SFR 可用性标志** 0_B CC80 和 CCU80 内核 SFR 不可用。 1_B CC80 和 CCU80 内核 SFR 可用。
AVAIL1	1	r	**CC81 可用性标志** 0_B CC81 不可用。 1_B CC81 可用。
AVAIL2	2	r	**CC82 可用性标志** 0_B CC82 不可用。 1_B CC82 可用。
AVAIL3	3	r	**CC83 可用性标志** 0_B CC83 不可用。 1_B CC83 可用。
AVAIL12	12	r	**POSIF0 可用性标志** 0_B POSIF0 不可用。 1_B POSIF0 可用。
AVAIL15	15	r	**BCCU0 可用性标志** 0_B BCCU0 不可用。 1_B BCCU0 可用。
0	[31:16], [14:13], [11:4]	r	保留

10.4.3 存储器大小寄存器

ROM、 SRAM 和 Flash 存储器都有存储器大小寄存器。这些寄存器用来指示器件中 ROM、 SRAM 和 Flash 的可用存储器大小。

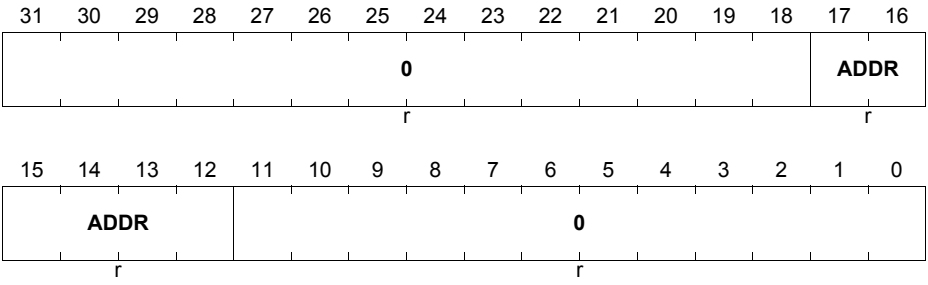
注： 这些大小寄存器的复位值显示超集器件的配置。实际值可能随产品型号的不同而不同。

ROMSIZE
ROM 大小寄存器 **(0400_H)** **复位值 : 0000 0B00$_H$**



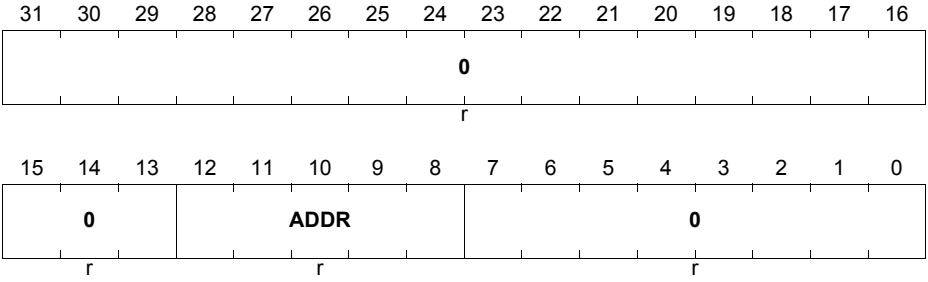
域	位	类型	描述
ADDR	[13:8]	r	**ROM 大小** 用户可读 ROM 的大小 （字节数）= ADDR * 256
0	[31:14], [7:0]	r	**保留**

FLSIZE
Flash 大小寄存器 **(0404_H)** **复位值 : 0003 3000$_H$**



域	位	类型	描述
ADDR	[17:12]	r	**Flash 大小** Flash (不包括 Flash 扇区 0) 大小（字节数）= (ADDR - 1) * 4
0	[31:18], [11:0]	r	**保留**

RAM0SIZE
RAM0 大小寄存器 **(0410$_H$)** **复位值 : 0000 1000$_H$**



域	位	类型	描述
ADDR	[12:8]	r	**RAM0 大小** RAM 块 0 的大小 （字节数）= ADDR * 256 对于总的 RAM 大小，必须将 RAM 块 1 到 3 考虑在内。
0	[31:13], [7:0]	r	**保留**

系统控制

11 窗式看门狗定时器 (WDT)

窗式看门狗定时器模块的用途是改善系统的完整性。如果主程序因某些故障条件而忽视了定期操作看门狗（也称为“踢狗”、“爱抚狗”、“喂狗”或“唤醒狗”），WDT 会触发系统复位或者其他纠正动作，例如中断。目的是将系统从无反应状态带回到正常工作状态。

参考文献

[1] Cortex-M0 User Guide, ARM DUI 0467B (ID081709)

11.1 概述

一次成功的 WDT 服务会导致在信号 wdt_service 上产生一个脉冲。该信号也作为一个备用功能输出被提供，可用于向一个外部看门狗表示系统还在正常运行。

WDT 定时器是一个从 0_H 向上计数的 32 位计数器。在计数器的值位于窗口边界内（即下边界值和上边界值之间）时，它可以被服务。正确的服务会导致计数器复位到 0_H。一次所谓的“差服务”尝试会导致产生系统复位请求。

定时器模块使用独立于总线时钟的 f_{WDT} 时钟运行。定时器的值在相应的 AHB 寄存器 **TIM** 内更新。该机制使得对来自总线读访问能立即反应。

11.1.1 特性

看门狗定时器 (WDT) 是一个独立的窗式看门狗定时器。

它有以下特性：

- 当没有及时得到服务或者得到错误方式的服务时会触发系统复位
- 服务被限制在刷新窗口的边界内
- 能使用独立的时钟运行
- 提供服务指示给一个外部引脚
- 能在停止模式下挂起
- 提供可选的复位前报警

表 11-1 应用特性

特性	用途 / 应用
在错误服务情况下系统复位	被触发后恢复系统的稳定运行，确保系统的完整性
服务被限制在定义的刷新窗口边界内	允许考虑最小和最大的软件定时
独立的时钟	确保 WDT 即使在系统发生时钟故障时也能计数
在外部引脚上的服务指示	对于双通道的看门狗解决方案，提供对系统完整性的附加外部控制

表 11-1　　应用特性 (续表)

特性	用途 / 应用
在停止模式下挂起	保证对生产代码的安全调试
报警	软件恢复。允许软件恢复程序采取纠正动作将系统从无反应状态带回到正常工作状态

11.1.2　框图

WDT 框图如 图 11-1 所示。

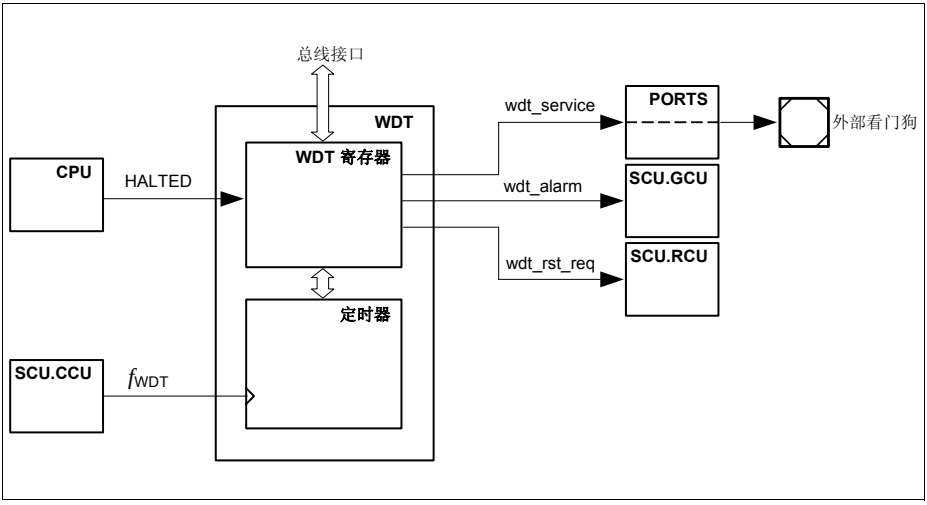


图 11-1　　看门狗定时器框图

11.2　超时模式

当计数值超过上边界时，WDT 会发生溢出，如果没有用 **CTR** 寄存器使能溢出事件预警，会立即导致产生一次复位请求，该请求通过信号 wdt_rst_req 进入 SCU 的 RCU。。执行一次成功的服务是在有效的服务窗口内向 WDT 的 **SRV** 寄存器写入一个被称为 “ 魔字 ” 的特定值，这会导致在信号 wdt_service 上产生一个脉冲并使定时器的计数器复位。

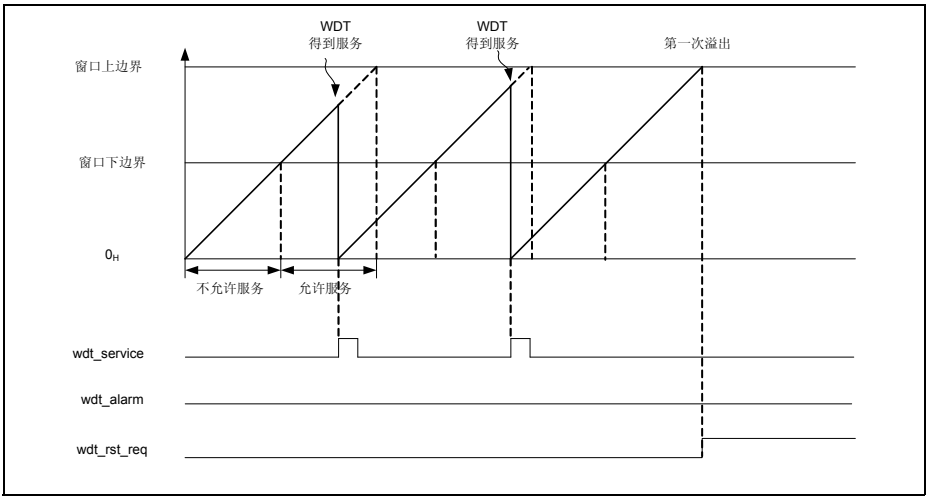


图 11-2　　无预警复位

在图 11-2 中描绘的示例场景展示了在有效时间窗口内 WDT 模块成功服务后产生的两个连续的服务脉冲。对于没有进行过服务的情况，在计数器的值超出窗口上限值后立即在 wdt_rst_req 上触发产生复位请求。

11.3　　预警模式

在预警模式，溢出事件的作用在使能和不使能预警功能的情况下是不同的。当预警被使能时，计数值第一次超过上限时触发输出报警信号 wdt_alarm。只能在下一次溢出时产生复位请求。报警状态通过寄存器 **WDTSTS** 指示，可通过寄存器 **WDTCLR** 清除。清除报警状态会使 WDT 回到正常状态。

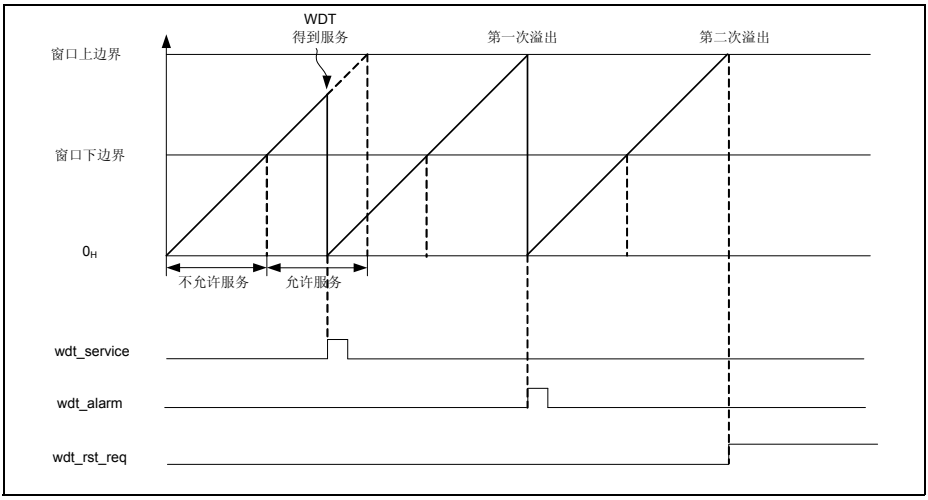


图 11-3　　预警后复位

图11-3中描绘的示例场景展示了在有效时间窗内WDT模块成功服务后产生的服务脉冲。第一次错过服务时，WDT 在 wdt_alarm 上产生报警脉冲。报警信号被作为中断请求连接到 SCU。在这个报警服务请求发生后和下一次溢出前，用户可以清除 WDT 的状态位，并给予一次正确的 WDT 服务。否则 WDT 将会在第二次错过服务时在 wdt_rstn 上产生复位请求。

11.4　　错误服务操作

一次错误的服务尝试会导致产生一次复位请求。一次错误的服务尝试可能是因为在窗口边界外提供服务或使用一个无效的魔字提供服务。

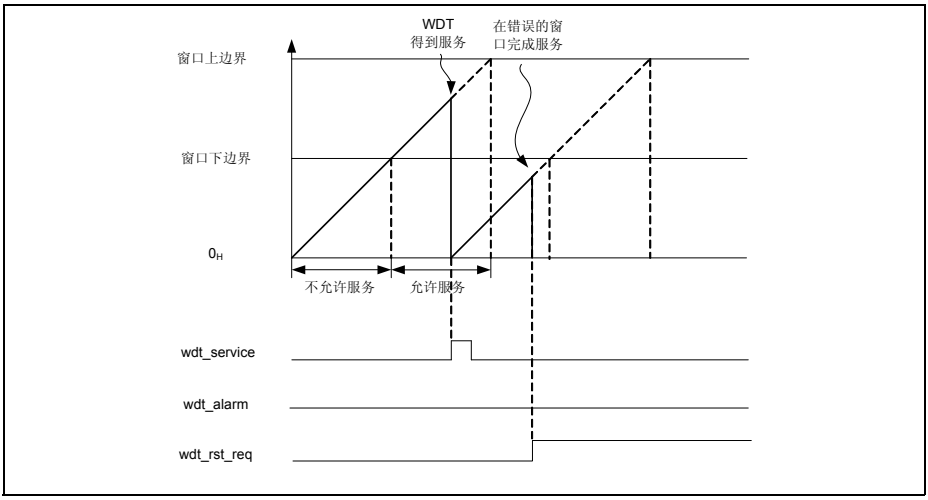


图 11-4　　在错误的窗口服务后复位

图 11-4 的例子示出了在有效服务窗口之外进行的服务。在计数器的值仍然低于窗口下边界时，试图服务 WDT 会导致在 wdt_rst_req 信号上立即产生复位请求。

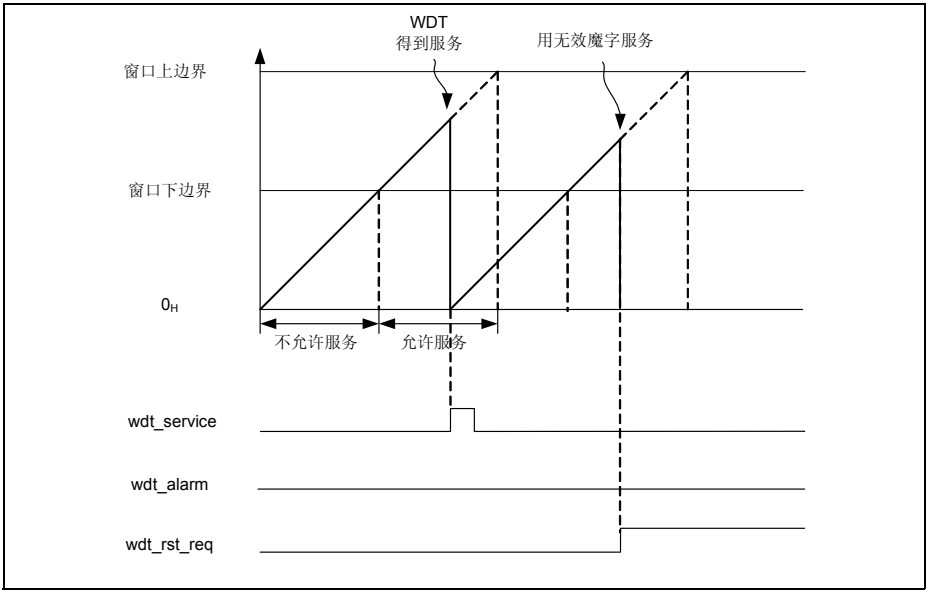


图 11-5　　使用错误的魔字服务后复位

图 11-5 的例子示出了在一个有效服务窗口内但却使用了一个无效魔字进行的服务。试图向 SRV 寄存器写入一个错误的字会导致在 wdt_rst_req 信号上立即产生复位请求。

11.5 服务请求处理

在预警模式被使能的情况下，当计数器第一次超过看门狗上限而溢出时， WDT 会通过 wdt_alarm 输出信号产生看门狗报警服务请求。报警服务请求由 SCU 处理。

服务请求可以通过 SCU 中的服务请求屏蔽寄存器分别禁止。

11.6 调试行为

当 CPU 进入 HALT 模式时， WDT 功能可以被挂起。WDT 调试功能受 CTR 寄存器中 DSP 位域的控制，其默认设置是被挂起。

11.7 电源、复位和时钟

WDT 模块是内核域的一部分，由电压 V_{DDC} 供电。

所有的 WDT 寄存器在系统复位期间被复位。

SCU/CCU 模块的复位状态寄存器 RSTSTAT 的一个粘滞位指示最后一次系统复位是否是被 WDT 模块触发的。该位在系统复位时不复位。

WDT 计数器的输入时钟由来自 SCU/CCU 模块的内部 32kHz 待机时钟提供，独立于 AHB 接口时钟。

WDT 模块的时钟默认是被禁止的，可以通过 SCU_CGATCLR0 寄存器使能。使能和禁止模块时钟可能引起负载的变化，并可能发生如 SCU 一章的 CCU （时钟门控制）一节所阐明的那种时钟消隐。强烈建议在用户初始化代码中设置模块的时钟，以避免在运行时出现时钟消隐。

11.8 初始化和控制序列

WDT 模块的编程模型假设了几个使用不同控制序列的场景。

注： 本章描述的一些场景需要在系统级上操作，这不在 WDT 模块描述的范围内，因此详细的信息请参见本文档的相关章节。

11.8.1 操作的初始化和启动

系统复位后需要进行完全的 WDT 模块初始化。

- 检查最后一次系统复位的原因，以确定电源状态
  - 读 SCU_RSTSTAT.RSTSTAT 寄存器位域，以确定最后一次系统复位的原因，可以用位 SCU_RSTCLR.RSCLR 清除这个位。
  - 根据最后一次系统的复位执行合适的操作。
- WDT 软件初始化序列

  - 用 SCU_CGATCLR0.WDT 寄存器位域使能 WDT 时钟。
  - 用 WDT_WLB 寄存器设置窗口下边界
  - 用 WDT_WUB 寄存器设置窗口上边界
  - 配置外部看门狗服务指示 (可选，请参见端口一章)
  - 用 SCU_SRMSK 寄存器在系统级上使能预警中断（可选，仅用于 WDT 预警模式）
- 软件启动序列
  - 选择模式 （超时或预警），并用 WDT_CTR 寄存器使能 WDT 模块
- 服务看门狗
  - 在有效时间窗口内写魔字到 WDT_SRV 寄存器

11.8.2 软件停止和恢复操作

可以使用软件序列在任何时刻停止或重新启动 WDT 模块，例如用于调试目的。

- 软件停止序列
  - 用 WDT_CTR 寄存器禁止 WDT 模块
- 执行任何用户操作
- 软件启动 （恢复）序列
  - 用 WDT_CTR 寄存器中使能 WDT 模块
- 服务看门狗
  - 在有效时间窗口内写魔字到 WDT_SRV 寄存器

11.8.3 进入休眠 / 深度睡眠和恢复操作

在休眠和深度睡眠状态期间， WDT 计数器时钟可以被配制为停止。在 CPU 休眠期间，如果 WDT 时钟配制为停止，则在这些模式下软件与 WDT 不需要直接交互，看门狗超时也不会发生。

- 休眠 / 深度睡眠模式的软件配置序列
  - 用 SCU_CGATx 寄存器配置看门狗的行为
- 进入休眠 / 深度睡眠模式的软件序列
  - 在 CPU 中选择休眠或深度睡眠模式 （详情请参阅 Cortex-M0 文档 **[1]**）
  - 进入所选模式 （详情请参阅 Cortex-M0 文档 **[1]**）
- 等待一个唤醒事件 （没有软件交互， CPU 停止）
- 恢复操作 （在发生唤醒事件时 CPU 时钟自动重启）
- 服务看门狗
  - 在有效时间窗口内写魔字到 WDT_SRV 寄存器

11.8.4 预警警报处理

在预警模式下，如果在有效时间窗口内未得到服务，WDT 会在请求系统复位前启动预警警报。在启动警报后和定时器的计数器值第二次超过上限之前，指示该警报的 WDT 状态寄存器必须被清除。在清除了警报状态后，必须在有效时间窗口内进行正常的看门狗服务。

- 警报事件
  – 异常例程 （服务请求）用 WDT_WDTCLR 寄存器清除 WDT_WDTSTAT 寄存器
- 服务看门狗
  – 在有效时间窗口内写魔字到 WDT_SRV 寄存器

11.9 WDT 寄存器

寄存器概述

绝对寄存器地址通过下面的加法计算：

模块基地址 + 偏移地址

表 11-2 寄存器地址空间

模块	基地址	结束地址	备注
WDT	$4002\ 0000_H$	$4002\ FFFF_H$	看门狗定时器寄存器

表 11-3 寄存器一览表

简称	描述	偏移地址	访问方式		描述见
			读	写	
WDT 内核寄存器					
ID	模块 ID 寄存器	00_H	U, PV	PV	页 11-9
CTR	控制寄存器	04_H	U, PV	PV	页 11-10
SRV	服务寄存器	08_H	BE	PV	页 11-11
TIM	定时器寄存器	$0C_H$	U, PV	BE	页 11-12
WLB	窗口下边界	10_H	U, PV	PV	页 11-12
WUB	窗口上边界	14_H	U, PV	PV	页 11-13
WDTSTS	看门狗状态寄存器	18_H	U, PV	PV	页 11-14
WDTCLR	看门狗状态清除寄存器	$1C_H$	U, PV	PV	页 11-14

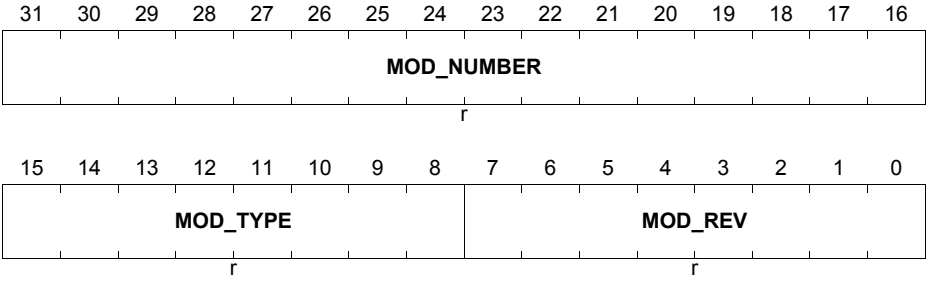
11.9.1 寄存器描述

ID

模块唯一 ID 寄存器。

ID
WDT 模块 ID 寄存器

(0000$_H$) 复位值 : 00AD C0XX$_H$

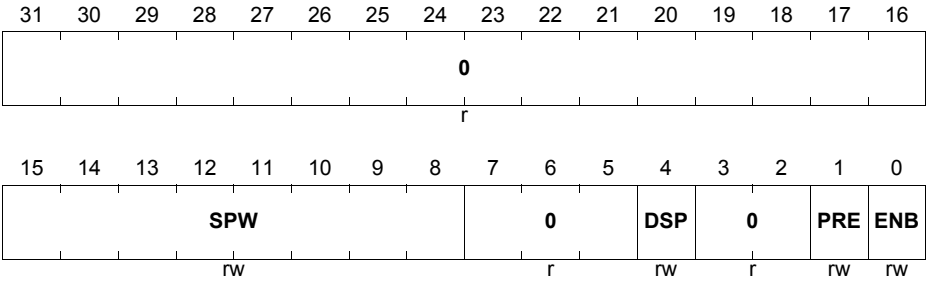


域	位	类型	描述
MOD_REV	[7:0]	r	**模块版本号** 指示实现的版本号，模块版本号从 01_H 开始（第一个版本）。
MOD_TYPE	[15:8]	r	**模块类别** 该位域固定为 $C0_H$。
MOD_NUMBER	[31:16]	r	**模块编号值** 该位域定义模块的标识号。

CTR

工作模式控制寄存器

CTR
WDT 控制寄存器 **(04_H)** **复位值 : 0000 0000_H**



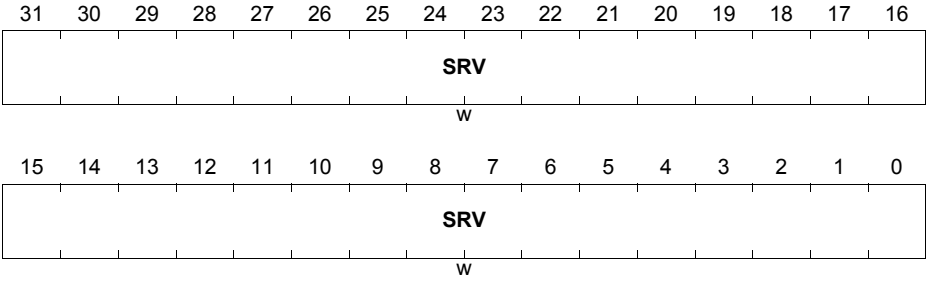
域	位	类型	描述
ENB	0	rw	**使能** 0_B 禁止看门狗定时器 1_B 使能看门狗定时器
PRE	1	rw	**预警** 0_B 禁止预警 1_B 使能预警
DSP	4	rw	**调试挂起** 0_B 在调试暂停模式期间停止看门狗定时器 1_B 在调试暂停模式期间不停止看门狗定时器
SPW	[15:8]	rw	**服务指示脉冲宽度** 服务指示脉冲的宽度 (SPW+1)，以 clk_wdt 周期为单位
0	[3:2], [7:5], [31:16]	r	**保留**

SRV

WDT 服务寄存器。软件必须在定时器值位于有效窗口边界内时写入魔字。在定时器值位于窗口边界内时写魔字将服务看门狗并导致用 0H 重新加载定时器。

在有效时间窗口内写不同于魔字的数据，或者虽然写入正确的魔字但却是在有效时间窗口之外，则不会执行看门狗服务，而是会立即产生一个系统复位请求。

SRV
WDT 服务寄存器 **(08_H)** **复位值 : 0000 0000$_H$**

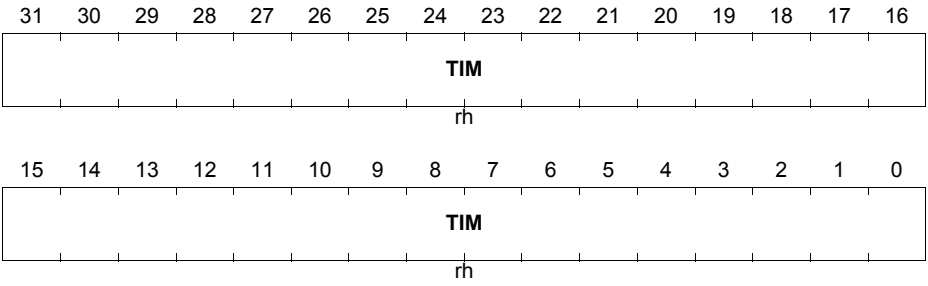


域	位	类型	描述
SRV	[31:0]	w	**服务** 当定时器的值位于窗口边界内时，写入魔字 $ABADCAFE_H$ 将会服务看门狗。

TIM

看门狗定时器寄存器的当前计数值。可以通过软件读取这个寄存器，以确定在 WDT 时间窗口内的当前位置。

TIM
WDT 定时器寄存器 **($0C_H$)** **复位值 : 0000 0000_H**

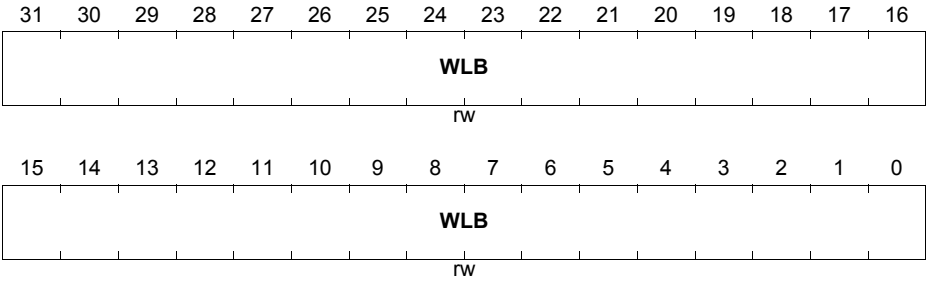


域	位	类型	描述
TIM	[31:0]	rh	**定时器值** 看门狗定时器的当前值

WLB

窗口下边界寄存器定义服务窗口的下边界。看门狗服务仅在窗口边界内有效。

WLB
WDT 窗口下边界寄存器 **(10_H)** **复位值 : 0000 0000_H**

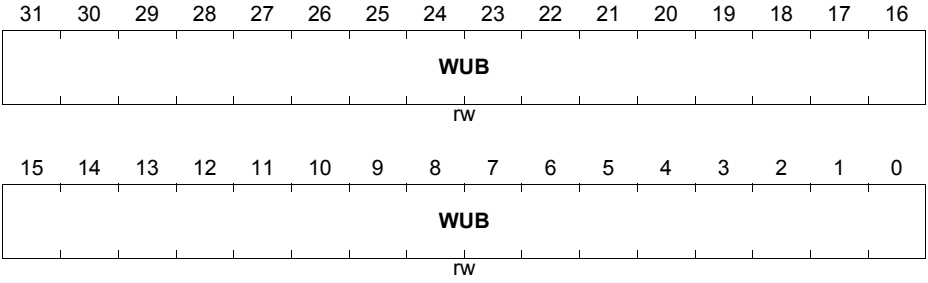


域	位	类型	描述
WLB	[31:0]	rw	**窗口下边界** 服务窗口的下边界 设置下边界为 0_H 将禁用窗口机制。

WUB

窗口上边界寄存器定义服务窗口的上边界。看门狗服务仅在窗口边界内有效。

WUB
WDT 窗口上边界寄存器 (14_H) 复位值 : FFFF FFFF$_H$

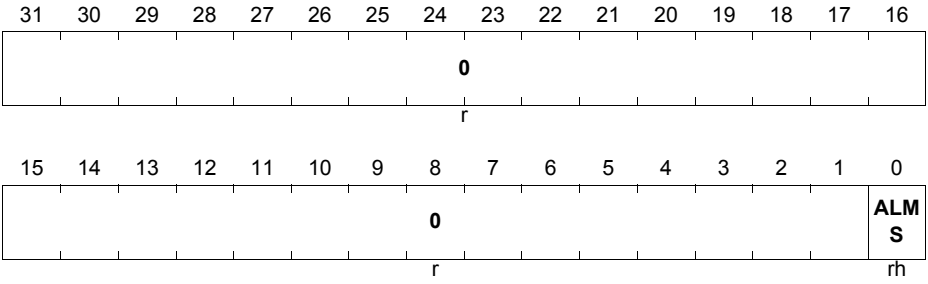


域	位	类型	描述
WUB	[31:0]	rw	**窗口上边界** 服务窗口的上边界 在没有预警使能的情况下，当定时器超过上限值时，WDT 会触发一次复位请求 在有预警使能的情况下，第一次超过上限值时会触发看门狗警报，第二次超过上限值时会触发一次系统复位。

WDTSTS

状态寄存器包含指示报警情况发生的粘滞位。

WDTSTS
WDT 状态寄存器 **(0018_H)** **复位值 : 00000000_H**

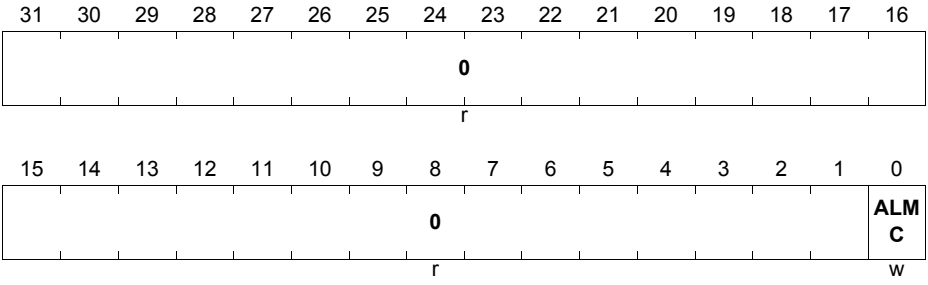


域	位	类型	描述
ALMS	0	rh	**预警警报** 1_B 发生了预警警报 0_B 未发生预警警报
0	[31:1]	r	**保留**

WDTCLR

状态寄存器包含粘滞位域指示报警情况的发生。

WDTCLR
WDT 清除寄存器 **($001C_H$)** **复位值 : 00000000_H**



域	位	类型	描述
ALMC	0	w	**预警警报** 1_B 清除预警警报 0_B 无动作
0	[31:1]	r	**保留**

11.10 互联

表 11-4 引脚表

输入 / 输出	I/O	连接到	描述
时钟和复位信号			
f_{WDT}	I	SCU.CCU	定时器时钟
定时器信号			
wdt_service	O	PORTS	给外部看门狗的服务指示
HALTED	I	CPU	在停止模式下调试。 在内核处于调试期间， HALTED 仍然保持有效。
服务请求连接			
wdt_alarm	O	SCU.GCU	预警警报
wdt_rst_req	O	SCU.RCU	复位请求

12 实时时钟 (RTC)

实时时钟 (RTC) 是一个跟踪当前时间的时钟。RTC 出现在计算机系统和几乎任何电子设备中，这些设备需要以数字格式保持供时钟显示用的精确时间。

12.1 概述

RTC 模块用独立的时、分、秒寄存器跟踪时间。日历寄存器跟踪日期、星期、月和年，还能自动纠正闰年[1)]。 RTC 时钟可以通过位 SCU_CLKCR.RTCCLKSEL 来选择。

RTC 模块中的定时器在休眠或深度休眠模式下可以继续工作。

12.1.1 特性

实时时钟模块 (RTC) 有以下特性：

- 使用下面的时钟保持实际时间
  - 32.768 kHz 外部时钟
  - 32.768 kHz 内部时钟
- 周期性的基于时间的中断
- 可编程的时间匹配报警中断
- 支持从休眠或深度休眠模式唤醒

表 12-1 应用特性

特性	用途 / 应用
精确的实际时间保持	减少了调时的需要
周期性的基于时间的中断	在精确定义的时间间隔执行计划任务
可编程的时间匹配报警中断	在精确定义的时间执行计划任务
支持从休眠或深度休眠模式唤醒	从休眠或深度休眠模式自动唤醒，用于系统状态控制和维护例行操作

12.1.2 框图

RTC 的框图如图 12-1 所示。

RTC 的主要构件是实现实时计数的时间计数器和 RTC 寄存器， RTC 寄存器包括用于时间计数器的多域寄存器和报警编程寄存器，有专用位域分别代表流逝的秒、分、时、天、星期、月和年值。

RTC 模块直接受 SCU 模块的控制，与 SCU 的其它子模块共享系统总线接口。

访问 RTC 寄存器是经串行接口通过更新后的寄存器镜像来进行的。

1) 当年可被 4 整除时，自动执行闰年校正。

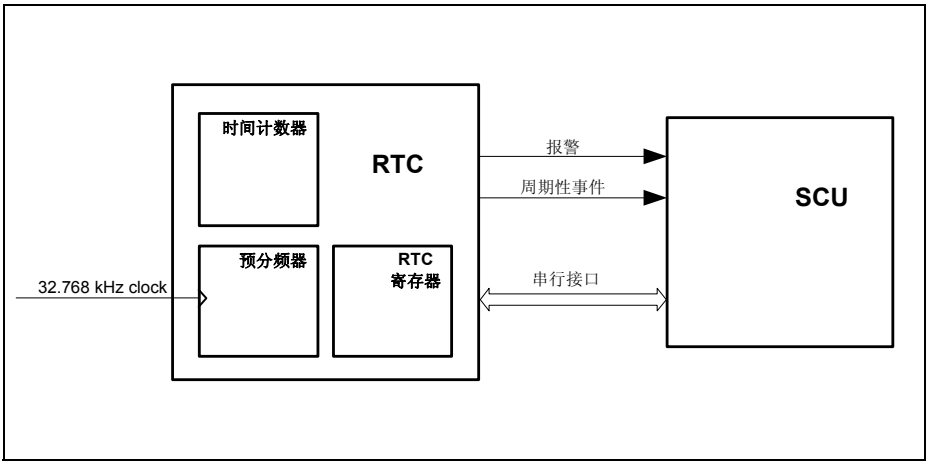


图 12-1　　实时时钟结构框图

12.2　　RTC 操作

RTC 定时器在不同域内对时间的秒、分、时、天、星期、月和年计数 (见 图 12-2)。软件通过 SCU 模块中的镜像寄存器经串行接口来编程和读取 RTC 计数器的独立位域。

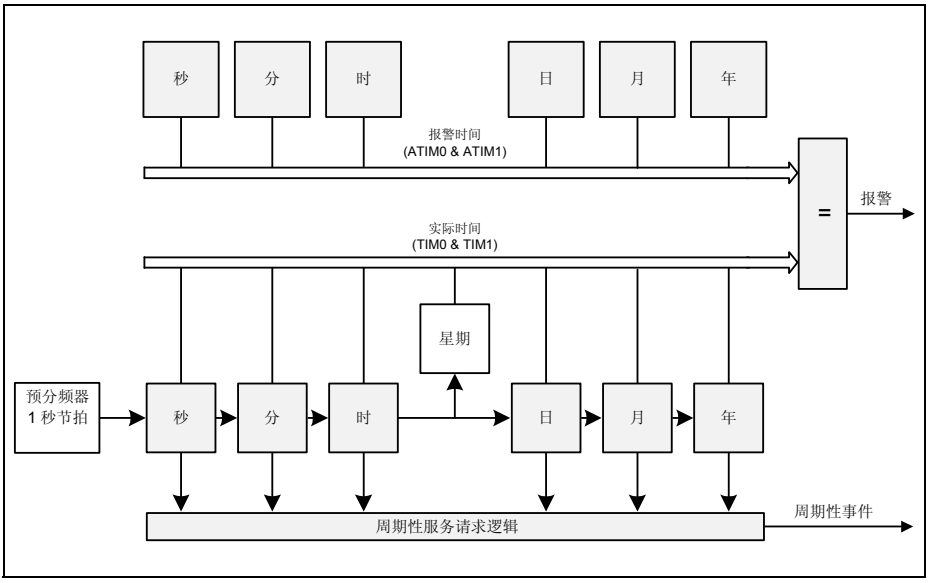


图 12-2　　RTC 时间计数器框图

发生的内部定时器事件保存在服务请求原始状态寄存器 **RAWSTAT** 中。状态寄存器 **RAWSTAT** 的值驱动输出的服务请求线警报和 periodic_event 事件。

12.3　寄存器访问操作

从编程模型的角度看 RTC 模块是 SCU 的一部分，它与 SCU 的其他子模块共享用于配置的寄存器地址空间。RTC 寄存器在 RTC 模块内实现，但在 SCU 内有其镜像。寄存器通过在 32kHz 时钟频率下运行的串行接口在两个时钟域里得到更新。

执行任何寄存器更新时都需要有一定的延时，这是因为通过串行接口与镜像寄存器交换数据需要时间。对内核域内的 RTC 寄存器的访问决不允许阻塞 SCU 模块的总线接口。有关寄存器镜像和串口通信处理的详细信息请参见 SCU 一章。

为了可靠地写定时器寄存器 **TIM0** 和 **TIM1**，寄存器 **TIM0** 必须在寄存器 **TIM1** 之前写入。

为了可靠读取定时器寄存器 **TIM0** 和 **TIM1**，寄存器 **TIM0** 必须在寄存器 **TIM1** 之前读取。在 **TIM0** 和 **TIM1** 的值被复制到内核域的镜像寄存器之前，每次读取 **TIM0** 时， **TIM1** 的值被保存在一个映射寄存器中。

12.4 服务请求处理

RTC 在发生下述情况下产生服务请求：

- 周期性的定时器事件
- 配置的报警条件

可以在内核域将这些服务请求作为常规服务请求或者作为从休眠或深度休眠模式的唤醒触发处理。

12.4.1 周期性服务请求

只要定时器计数器的非屏蔽域得到更新，就会产生周期性的定时器服务请求。这些位的屏蔽使用 **MSKSR** 寄存器执行。周期性的服务请求可以用 **MSKSR** 禁止。

12.4.2 定时器报警服务请求

当 **TIM0** 和 **TIM1** 位域的值与 **ATIM0**、 **ATIM1** 寄存器中所有对应位域的值匹配时，会触发报警中断。定时器报警服务请求可以用 **MSKSR** 寄存器使能 / 禁止。

12.5 调试行为

当 CPU 进入 HALT 模式时， RTC 的功能可被挂起。RTC 调试功能由 **CTR** 寄存器中的 SUS 位域控制。

注： 在 XMC1300 中，任何复位都会将位 CTR.SUS 复位到其默认值。因此，如果在调试期间需要挂起功能，建议在用户的初始化代码中使能调试挂起功能。在对寄存器编程前，必须使能模块时钟，并且在使能模块时钟时，要像 SCU 一章的 CCU (时钟门控控制) 一节描述的那样谨慎处理。

12.6 电源、复位和时钟

在休眠和深度休眠模式，可以将 RTC 编程为一直保持供电。在初次上电后，RTC 模块一直保持在复位状态，直到复位解除。

RTC 定时器使用一个内部或外部 32.768 kHz 的时钟运行，这可以通过 SCU/CCU 模块的 CLKCR.RTCCLKSEL 控制寄存器来选择，如 **图 12-3** 所示。将预分频值设置为 $7FFF_H$ 会导致每秒更新一次 RTC 定时器。

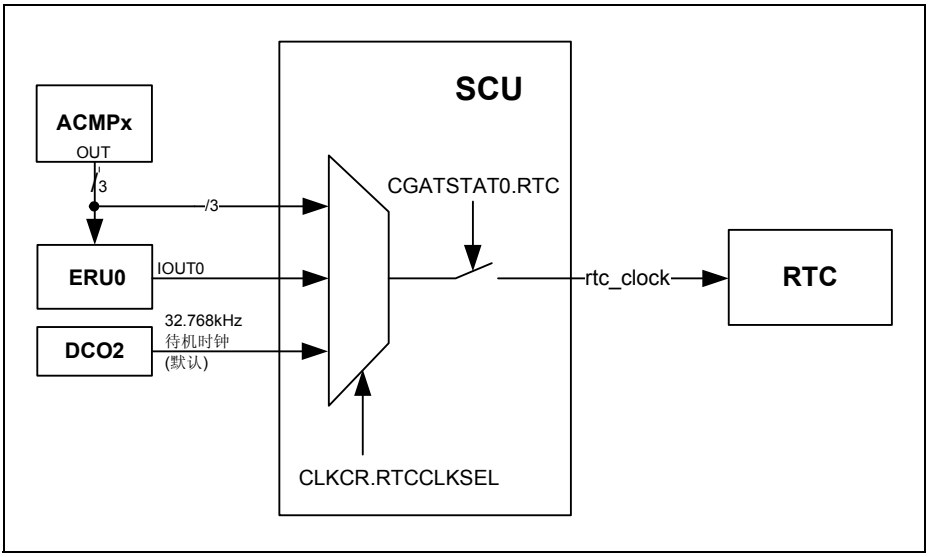


图 12-3　　RTC 时钟选择

RTC 模块时钟在默认情况下是被禁止的，可以通过 SCU_CGATCLR0 寄存器使能。

注：　在通过 CLKCR.RTCCLKSEL 改变 RTC 时钟源之前，RTC 时钟必须由 CGATSET0.RTC 位进行门控。

12.7　初始化和控制序列

RTC 模块的编程模型假设了几个使用不同控制序列的场景。

注：　本章中描述的某些场景需要在系统级操作，这不在 RTC 模块描述的范围，因此更详细的信息请参见本文档的相关章节。

12.7.1　操作的初始化和启动

复位后需要对 RTC 模块进行全面初始化。对 RTC 寄存器的访问通过专用镜像寄存器执行 (更详细的信息请参见 SCU 一章)

- 使能给 RTC 模块的时钟
  - 向 SCU_CGATCLR0.RTC 写 1
- 用当前时间对 RTC_TIM0 和 RTC_TIM1 寄存器编程
  - 检查SCU_MIRRSTS，以确保没有经串行接口到RTC_TIM0和RTC_TIM1寄存器的数据传送正在进行
  - 向 RTC_TIM0 和 RTC_TIM1 寄存器写入一个新值
- 使能 RTC 模块以启动计时
  - 向 RTC_CTR.ENB 写 1

注： 为确保经串行接口进行的数据传送成功，对于每次数据传送， RTC_TIM0 和 RTC_TIM1 只能被写一次。另外，写入这些寄存器的数据必须是 32 位宽。单独的位访问不会启动串行传送操作。

12.7.2 配置和使能周期性事件

RTC 周期性事件的配置需要编程，以便在相应位域的值发生改变时能产生中断请求。

- 在 RTC 模块中使能周期性定时器事件服务请求
  - 在 RTC_MSKSR 寄存器中设置各自的位域 (MPSE, MPMI, MPHO, MPDA, MPMO, MPYE)，以使能独立的周期性定时器事件。

12.7.3 配置和使能定时器事件

RTC 报警事件的配置需要编程，以便在 ATIM0 和 ATIM1 分别与 TIM0 和 TIM1 对应位域的值匹配时能产生中断请求。

- 将比较值编程到 RTC 模块中的 ATIM0 和 ATIM1 的独立位域
  - 检查 SCU_MIRRSTS，以确保没有经串行接口到 RTC_ATIM0 和 RTC_ATIM1 寄存器的数据传送正在进行。
  - 写 RTC_ATIM0 和 RTC_ATIM1 寄存器
- 使能用于 RTC 模块中定时器报警事件服务请求
  - 设置 RTC_MSKSR 寄存器的 MAI 位域，以使能独立的周期性定时器事件。

注： 为确保经串行接口进行的数据传送成功，对于每次数据传送， RTC_ATIM0 和 RTC_ATIM1 只能被写一次。另外，写入这些寄存器的数据必须是 32 位宽。单独的位访问不会启动串行传送操作。

12.8 RTC 寄存器

寄存器概述

绝对寄存器地址通过下面的加法计算：

模块基地址 + 偏移地址

表 12-2 寄存器地址空间

模块	基地址	结束地址	备注
RTC	$4001\ 0A00_H$	$4001\ 0AFF_H$	

表 12-3　　寄存器一览表

简称	描述	偏移地址	访问方式		描述见
			读	写	
RTC 内核寄存器					
ID	ID 寄存器	0000_H	U, PV	BE	页 12-7
CTR	控制寄存器	0004_H	U, PV	PV	页 12-8
RAWSTAT	原始服务请求寄存器	0008_H	U, PV	BE	页 12-9
STSSR	状态服务请求寄存器	$000C_H$	U, PV	BE	页 12-10
MSKSR	屏蔽服务请求寄存器	0010_H	U, PV	PV	页 12-11
CLRSR	清除服务请求寄存器	0014_H	U, PV	PV	页 12-12
ATIM0	报警时间寄存器 0	0018_H	U, PV	PV	页 12-13
ATIM1	报警时间寄存器 1	$001C_H$	U, PV	PV	页 12-14
TIM0	时间寄存器 0	0020_H	U, PV	PV	页 12-16
TIM1	时间寄存器 1	0024_H	U, PV	PV	页 12-17

12.8.1　寄存器描述

ID

RTC 模块的只读 ID 寄存器包含 RTC 模块的唯一标识码。

ID
RTC 模块 ID 寄存器　　(00_H)　　复位值 : 00A3 C0XX$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MOD_NUMBER															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOD_TYPE								MOD_REV							
r								r							

域	位	类型	描述
MOD_REV	[7:0]	r	**模块的版本号** 指示实现的版本号。模块版本值从 01_H(第一个版本) 开始。
MOD_TYPE	[15:8]	r	**模块类型** 该位域固定为 $C0_H$。
MOD_NUMBER	[31:16]	r	**模块编号值** 该位域定义模块的标识号

CTR

RTC 控制寄存器提供对模块工作模式的控制方法。

CTR
RTC 控制寄存器 **(04_H)** **复位值 : 7FFF 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DIV															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			0					0						SUS	ENB
r	r	r	r	r	r	r	r			r			r	rw	rw

域	位	类型	描述
ENB	0	rw	**RTC 模块使能** 0_B 禁止 RTC 模块 1_B 使能 RTC 模块
SUS	1	rw	**调试挂起控制** 0_B 在停止模式下调试时 RTC 不停止 1_B 在停止模式下调试时 RTC 停止
DIV	[31:16]	rw	**分频值** RTC 预分频器的重载值。时钟要除以 DIV+1。 $7FFF_H$ 是 RTC 在 32.768 kHz 晶体或外部时钟模式的默认值。

域	位	类型	描述
0	[15:2]	r	**保留** 读出值为 0；应写入 0

RAWSTAT

RTC 原始服务请求寄存器包含原状态信息，即在状态屏蔽生效前产生的服务请求或中断。该寄存器用于调试目的，但也能用于状态查询，不产生服务请求。

RAWSTAT
RTC 原始服务请求寄存器 **(08_H)** **复位值：0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							**RAI**	**0**	**RPYE**	**RPMO**	**0**	**RPDA**	**RPHO**	**RPMI**	**RPSE**
r							rh	r	rh	rh	r	rh	rh	rh	rh

域	位	类型	描述
RPSE	0	rh	**原始周期性秒服务请求** 在秒计数增 1 时置位
RPMI	1	rh	**原始周期性分服务请求** 在分计数增 1 时置位
RPHO	2	rh	**原始周期性时服务请求** 在小时计数增 1 时置位
RPDA	3	rh	**原始周期性日服务请求** 在天数增 1 时置位
RPMO	5	rh	**原始周期性月服务请求** 在月计数增 1 时置位
RPYE	6	rh	**原始周期性年服务请求** 在年计数增 1 时置位
RAI	8	rh	**报警服务请求** 在计数值与比较值匹配时置位
0	4, 7, [31:9]	r	**保留**

STSSR

RTC 服务请求状态寄存器包含在服务请求或中断产生时反映状态屏蔽效果的状态信息。软件需要访问该寄存器来确定事件的真实原因。

STSSR
RTC 服务请求状态寄存器 **($0C_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							SAI	0	SPYE	SPMO	0	SPDA	SPHO	SPMI	SPSE
r							rh	r	rh	rh	r	rh	rh	rh	rh

域	位	类型	描述
SPSE	0	rh	屏蔽后的周期性秒服务请求状态
SPMI	1	rh	屏蔽后的周期性分服务请求状态
SPHO	2	rh	屏蔽后的周期性时服务请求状态
SPDA	3	rh	屏蔽后的周期性日服务请求状态
SPMO	5	rh	屏蔽后的周期性月服务请求状态
SPYE	6	rh	屏蔽后的周期性年服务请求状态
SAI	8	rh	屏蔽后的报警服务请求状态
0	4, 7, [31:9]	r	保留

MSKSR

RTC 服务请求屏蔽寄存器包含服务请求或中断产生控制的屏蔽值。

MSKSR
RTC 服务请求屏蔽寄存器 **(10_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							MAI	0	MPYE	MPMO	0	MPDA	MPHO	MPMI	MPSE
r							rw	r	rw	rw	r	rw	rw	rw	rw

域	位	类型	描述
MPSE	0	rw	**周期性秒中断屏蔽** 0_B 禁止中断 1_B 使能中断
MPMI	1	rw	**周期性分中断屏蔽** 0_B 禁止中断 1_B 使能中断
MPHO	2	rw	**周期性时中断屏蔽** 0_B 禁止中断 1_B 使能中断
MPDA	3	rw	**周期性日中断屏蔽** 0_B 禁止中断 1_B 使能中断
MPMO	5	rw	**周期性月中断屏蔽** 0_B 禁止中断 1_B 使能中断
MPYE	6	rw	**周期性年中断屏蔽** 0_B 禁止中断 1_B 使能中断
MAI	8	rw	**报警中断屏蔽** 0_B 禁止中断 1_B 使能中断

域	位	类型	描述
0	4, 7, [31:9]	r	保留

CLRSR

RTC 清除服务请求寄存器的用途是清除 **RAWSTAT** 和 **STSSR** 寄存器的粘滞位。向某位写入 1 以便清除状态位。写 0 对置位或者复位没有影响。

CLRSR

RTC 清除服务请求寄存器 (14_H) 复位值 : 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							**RAI**	**0**	**RPYE**	**RPMO**	**0**	**RPDA**	**RPHO**	**RPMI**	**RPSE**
r							w	r	w	w	r	w	w	w	w

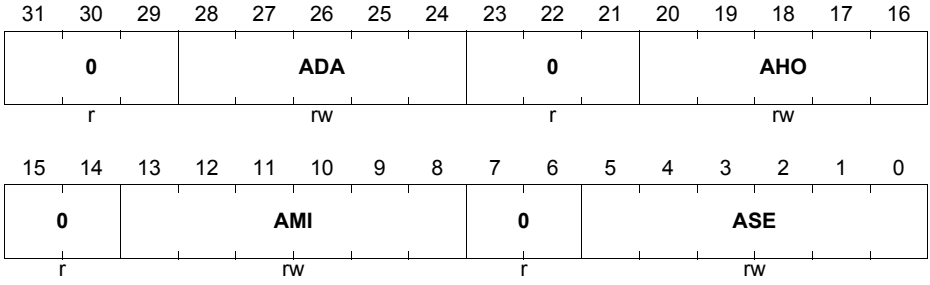
域	位	类型	描述
RPSE	0	w	**原始周期性秒中断清除** 0_B 无影响 1_B 清除状态位
RPMI	1	w	**原始周期性分中断清除** 0_B 无影响 1_B 清除状态位
RPHO	2	w	**原始周期性时中断清除** 0_B 无影响 1_B 清除状态位
RPDA	3	w	**原始周期性日中断清除** 0_B 无影响 1_B 清除状态位
RPMO	5	w	**原始周期性月中断清除** 0_B 无影响 1_B 清除状态位

域	位	类型	描述
RPYE	6	w	**原始周期性年中断清除** 0_B 无影响 1_B 清除状态位
RAI	8	w	**原始报警中断清除** 0_B 无影响 1_B 清除状态位
0	4, 7, [31:9]	r	**保留**

ATIM0

RTC 报警时间寄存器 0 的用途是对单一的报警时间编程，以便在期望的时间点与 **TIM0** 寄存器比较。ATM0 寄存器包含秒、分、时和日的位域部分。当试图向一个位域写入一个无效值时，例如超过最大值，则会用该位域的默认值对其编程，见对每个位域的描述。

ATIM0
RTC 报警时间寄存器 0 **(18_H)** **复位值 : 0000 0000$_H$**



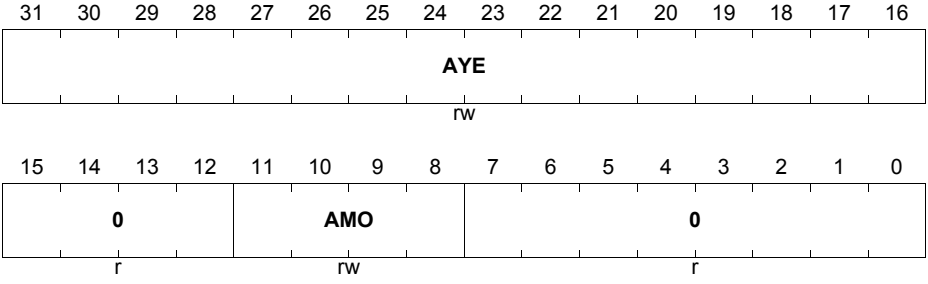
域	位	类型	描述
ASE	[5:0]	rw	**报警秒比较值** 秒定时器计数值与该值匹配时会触发报警秒中断。 设置值等于或大于 $3C_H$ 会导致将该域值设置为 0_H。
AMI	[13:8]	rw	**报警分比较值** 分定时器计数值与该值匹配时会触发报警分中断。 设置值等于或大于 $3C_H$ 会导致将该域值设置为 0_H。

域	位	类型	描述
AHO	[20:16]	rw	**报警时比较值** 小时定时器计数值与该值匹配时会触发报警时中断。 设置值等于或大于 18_H 会导致将该域值设置为 0_H。
ADA	[28:24]	rw	**报警日比较值** 日定时器计数值与该值匹配时会触发报警日中断。 设置值等于或大于 $1F_H$ 会导致将该域值设置为 0_H。
0	[7:6], [15:14], [23:21], [31:29]	r	**保留**

ATIM1

RTC 报警时间寄存器 1 服务的的用途是对单一的报警时间编程，以便在期望的时间点与 **TIM1** 寄存器比较。ATM1 寄存器包含月和年的位域部分。当试图向一个位域写入一个无效值时，例如超过最大值，则会用该位域的默认值对其编程，见对每个位域的描述。

ATIM1
RTC 报警时间寄存器 1 **($1C_H$)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
AMO	[11:8]	rw	**报警月比较值** 月定时器计数值与该值匹配时会触发报警月中断。 设置值等于或大于实际月计数天数时会导致将该域值设置为 0_H。
AYE	[31:16]	rw	**报警年比较值** 年定时器计数值与该值匹配时会触发报警年中断。

域	位	类型	描述
0	[7:0], [15:12]	r	保留

TIM0

RTC 时间寄存器 0 包含秒、分、时和日的当前时间值。各位域在与它们各自的含义相对应的时间间隔内得到更新。该寄存器需要在初次上电后被编程，以反映当前时间，然后连续计时。如果被使能，即使在休眠或深度休眠状态下也会继续计时。当试图向一个位域写入一个无效值时，例如超过最大值，则会用该位域的默认值对其编程，见对每个位域的描述。

TIM0
RTC 时间寄存器 0 **(20_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			DA					0			HO				
r			rwh					r			rwh				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			MI					0			SE				
r			rwh					r			rwh				

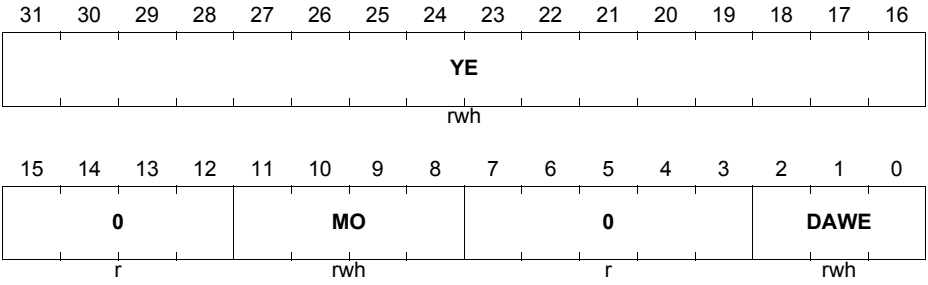
域	位	类型	描述
SE	[5:0]	rwh	**秒时间值** 设置值等于或大于 $3C_H$ 会导致将该域值设置为 0_H。 当 RTC 通过位 CTR.ENB 被禁止时，该值只能被写入。
MI	[13:8]	rwh	**分时间值** 设置值等于或大于 $3C_H$ 会导致将该域值设置为 0_H。 当 RTC 通过位 CTR.ENB 被禁止时，该值只能被写入。
HO	[20:16]	rwh	**时时间值** 设置值等于或大于 18_H 会导致将该域值设置为 0_H 当 RTC 通过位 CTR.ENB 被禁止时，该值只能被写入。
DA	[28:24]	rwh	**日时间值** 设置值等于或大于实际月计数的天数会导致将该域值设置为 0_H 当 RTC 通过位 CTR.ENB 被禁止时，该值只能被写入。 日计数器在月的第一天从 0 值开始。

域	位	类型	描述
0	[7:6], [15:14], [23:21], [31:29]	r	**保留**

TIM1

RTC 时间寄存器 1 包含星期、月和年的当前时间值。各位域在与它们各自的含义相对应的时间间隔内得到更新。该寄存器需要在初次上电后被编程，以反映当前时间，然后连续计时。如果被使能，即使在休眠或者深度休眠状态下也会继续计时。当试图向一个位域写入一个无效值时，例如超过最大值，则会用该位域的默认值对其编程，见对每个位域的描述

TIM1
RTC 时间寄存器 1 **(24_H)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
DAWE	[2:0]	rwh	**星期时间值** 设置值等于或大于 6_H 会导致将该域值设置为 0_H。 当 RTC 通过位 CTR.ENB 被禁止时，该值只能被写入。 日计数器在周的第一天从 0 值开始。
MO	[11:8]	rwh	**月时间值** 设置值等于或大于 C_H 会导致将该域值设置为 0_H。 当 RTC 通过位 CTR.ENB 被禁止时，该值只能被写入。 月计数器在年的第一个月从 0 值开始。
YE	[31:16]	rwh	**年时间值** 当 RTC 被禁止时，该值只能被写入。

域	位	类型	描述
0	[7:3], [15:12]	r	**保留**

12.9 互连

表 12-4 引脚连接

输入 / 输出	I/O	连接到	描述
时钟信号			
f_{RTC}	I	SCU.CCU	所选的 32.768 kHz 时钟
调试信号			
HALTED	I	CPU	表明处理器处于调试状态并停止
服务请求连接			
periodic_event	O	SCU.GCU	定时器周期性服务请求
alarm	O	SCU.GCU	报警服务请求

13 系统控制单元 (SCU)

SCU 是 SoC 电源、复位和时钟的管理者，另外还负责提供系统稳定性保护和其它辅助功能。

13.1 概述

本章中描述的 SCU 的功能组织在下面的几个小节内，每节介绍系统控制的不同方面：

- 杂项控制功能， GCU **13.2 节**
- 电源控制，PCU **13.3 节**
- 复位操作， RCU **13.4 节**
- 时钟控制， CCU **13.5 节**

13.1.1 特性

SCU 提供的下述特性用于监视和控制系统：

- 一般控制
  - 启动软件 (SSW) 和启动模式支持
  - 存储器内容保护
  - 中断处理
- 电源控制
  - 由 EVR 产生片上内核电源
  - 电源验证
  - 电源看门狗
  - 电压监视
  - 负载变化处理
- 复位控制
  - 由多种复位请求源产生复位
  - 系统复位产生
  - 复位后的复位源检查
- 时钟控制
  - 时钟产生
  - 时钟监控
  - 独立的外设时钟门控
  - 时钟消隐支持

13.1.2 框图

图 13-1 示出了下面的子单元：

- 电源控制单元 (PCU)
- 复位控制单元 (RCU)
- 时钟控制单元 (CCU)
- 综合控制单元 (GCU)

SCU 中的所有 SFR 都可以通过 AHB 和 16 位的 APB 总线接口访问，如 图 13-1 所示。APB 总线接口常用于访问称为 ANACTRL 寄存器的 SFR 组。这些寄存器用于配置系统中的模拟模块，即嵌入式稳压器 (EVR) 和数控振荡器 (DCO1 和 DCO2)。称为 SCU 寄存器的另一个 SFR 组可通过 AHB 总线接口访问。

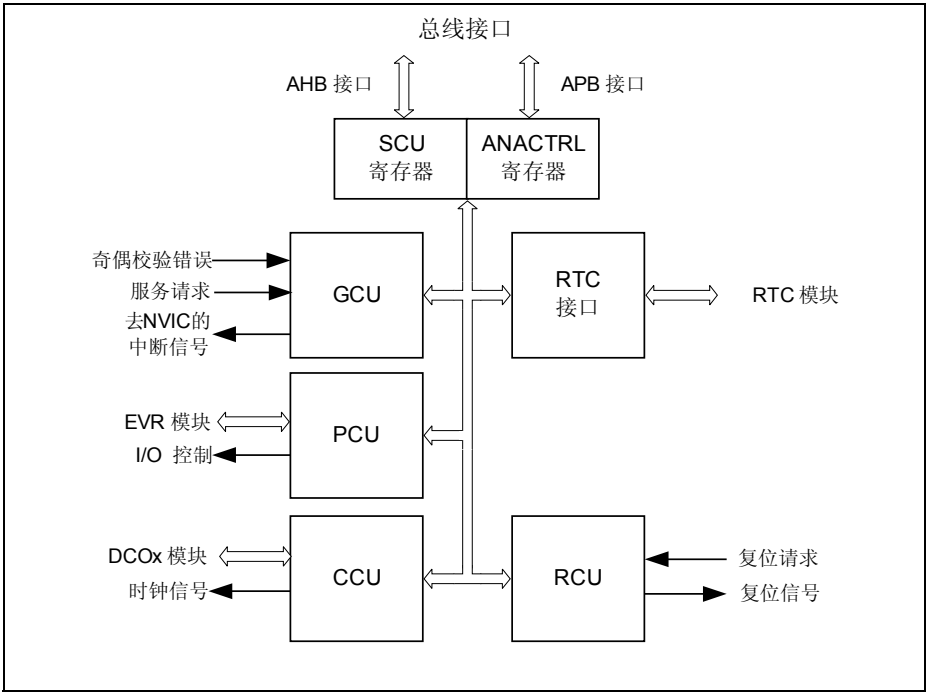


图 13-1 SCU 框图

综合控制单元接口

综合控制单元 GCU 有一个与每个片内 SRAM 和 Flash 的存储器验证逻辑连接的存储器故障接口，以接收类似于奇偶校验错误这样的存储器故障事件。

电源控制单元接口

电源控制单元 PCU 有一个与嵌入式稳压器 (EVR) 的接口和一个与 CCU 模块的接口。有关 PCU 相关信号的详细信息在 **13.3 节**中描述。

复位控制单元接口

复位控制单元 RCU 有一个与嵌入式稳压器 (EVR) 的接口。RCU 接收来自 EVR 的上电复位和掉电复位信息。复位请求可来自看门狗、CPU、GCU 和时钟控制单元 (CCU)。RCU

向芯片上内核电源域的所有其他单元提供复位信号。有关 RCU 相关信号的更多详细信息在 **13.4 节**中描述。

时钟控制单元接口

时钟控制单元 (CCU) 接收来自片上数字控制振荡器 (DCO) 的时钟源。 CCU 向片上所有其它单元提供时钟信号。

RTC 接口

对 RTC 模块的访问是经串行接口实现的。该接口通过与 RTC 模块寄存器之间的串行接口提供更新的镜像寄存器。经串行接口的镜像寄存器更新受 **MIRRSTS** 寄存器的控制。更新结束后还能通过 **SRRAW** 寄存器触发服务请求。寄存器镜像中寄存器的更新连续不断地进行，以尽可能快的速度即刻反映两边任何寄存器状态的变化。

RTC 模块的功能在单独的 RTC 一章描述。

13.2 杂项控制功能 (GCU)

系统控制执行系统管理功能，这些功能可通过 GCU 寄存器访问。包括各种辅助功能的综合系统控制在综合控制单元 (GCU) 中执行。

13.2.1 服务请求处理

表 13-1 列出的服务请求事件能够导致产生中断。参见 **SRMSK** 寄存器的描述。

中断结构如**图 13-2** 所示。中断请求或相应的中断置 1 位 (在寄存器 **SRSET** 中) 能在所选择的中断节点 x 触发中断产生。服务请求脉冲是从 **SRRAW** 寄存器中的中断标志独立产生的。中断标志可以由软件通过写寄存器 **SRCLR** 中的对应位清除。

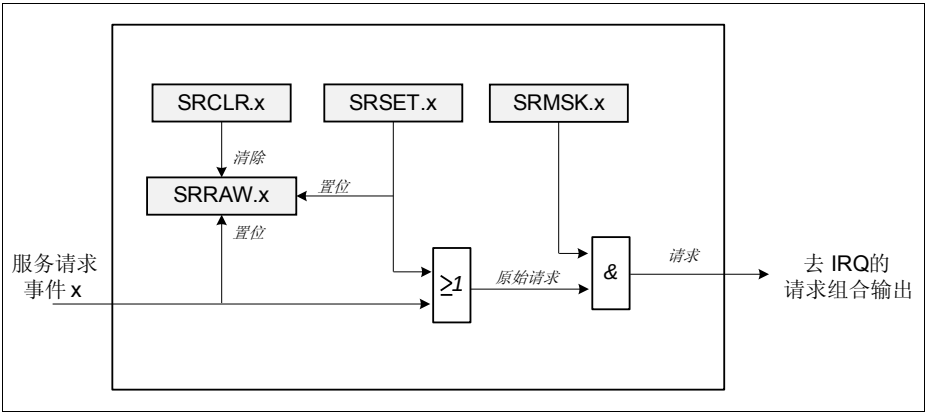


图 13-2 服务请求处理

寄存器 **SRRAW** 中的标志可通过软件写寄存器 **SRCLR** 中的对应位来清除 。所有的陷阱请求可以被组合到一根共同的线上，并连接到 NVIC 的一个常规中断节点。

注： 当对一个 SCU 服务请求进行服务时，要确保处理完所识别的请求后将所有相关的请求标志清除 。

13.2.1.1 服务请求源

SCU 支持在**表 13-1** 列出和在 **SRRAW**、**SRMSK**、**SRCLR** 、**SRSET** 寄存器中反映的服务请求源。触发这些服务请求的事件在各自的模块章节或者 SCU 一章的各节中描述。

表 13-1 服务请求

模块	服务请求名称	服务请求简称	SCU.SRx
NVM	Flash 双位 ECC 事件	FLECC2I	SR0
	Flash 操作完成事件	FLCMPLTI	
	16kB SRAM 奇偶校验错误事件	PESRAMI	
	USIC0 SRAM 奇偶校验错误事件	PEU0I	
SCU:CCU	时钟丢失事件	LOCI	
	待机时钟故障事件	SBYCLKFI	SR1
SCU:PCU	VDDP 预警事件	VDDPI	
	VDROP 事件	VDROPI	
	VCLIP 事件	VCLIPI	
SCU:GCU	温度传感器完成事件	TSE_DONE	
	温度传感器比较高事件	TSE_HIGH	
	温度传感器比较低事件	TSE_LOW	
WDT	WDT 预警	PRWARN	
RTC	RTC 周期性事件	PI	
	RTC 报警	AI	
	RTC CTR 镜像寄存器更新	RTC_CTR	
	RTC ATIM0 镜像寄存器更新	RTC_ATIM0	
	RTC ATIM1 镜像寄存器更新	RTC_ATIM1	
	RTC TIM0 镜像寄存器更新	RTC_TIM0	
	RTC TIM1 镜像寄存器更新	RTC_TIM1	
ORC	超量程比较器事件	ORCxI [x = 0 - 7]	SR2
ACMP	模拟比较器事件	ACMP0I, ACMP1I, ACMP2I	

13.2.2 SRAM 存储器内容保护

为了管理片上 SRAM 存储器的内容， SCU 提供了下面的机制：

所有的片上 SRAM 都通过奇偶校验提供内容保护。在执行写操作时，奇偶逻辑产生附加的奇偶位，这些奇偶位与每个数据字一起存储。读操作隐含了对先前存储的奇偶信息的检查。

奇偶检验错误的出现可以在 **SRRAW** 状态寄存器观察到。存储器错误是否触发中断可以通过 **SRMSK** 配置。当 **RSTCON**.SPERSTEN 或 **RSTCON**.U0PERSTEN 被置 1 时，奇偶检验错误还能触发系统复位。

可以用位 **PMTSR**.MTENS 单独使能对 16 kB SRAM 存储器的奇偶控制软件测试功能，例如用于支持在系统测试以满足 B 级（Class B）需求。一旦该位被置 1，在写操作时产生反转的奇偶位。当执行对该 SRAM 地址的读操作时，可以检测出奇偶检验错误。

注： 测试软件应位于不同的存储器空间。

13.2.3 ID 概要

本节描述 XMC1300 中的各种 ID。

模块标识

模块标识寄存器 指示每个外设的功能和设计步骤，寄存器 SCU_**ID** 用于 SCU 模块。

系统 ROM 表 ID

系统 ROM 表中的 PID 值在**表 13-2** 中定义。XMC1300 的系统 ROM 表位于 F000 0000$_H$。Cortex-M0 的 ROM 表在调试一章中描述。

表 13-2 XMC1300 系统 ROM 表的 PID 值

名称	偏移	参考	值
PID0	FE0$_H$	XMC1300 部件号 [7:0]	ED$_H$
PID1	FE4$_H$	位 [7:4] JEP106 ID 码 [3:0] 位 [3:0] XMC1300 部件号 [11:8]	11$_H$
PID2	FE8$_H$	位 [7:4] XMC1300 修订号 位 [3] == 1: JEDEC 分配的 ID 域 位 [2:0] JEP106 ID 码 [6:4]	1C$_H$
PID3	FEC$_H$	位 [7:4] 版本和小修订域 位 [3:0] 如果不为 0，指示一个客户修改的块	00$_H$
PID4	FD0$_H$	位 [7:4] 4KB 计数 位 [3:0] JEP106 连续码	00$_H$

芯片标识号

芯片标识号是一个 8 字长的编号。它包含寄存器 DBGROMID、IDCHIP、PAU_FLSIZE、PAU_RAM0SIZE 和 PAU_AVAILn(n=0-2) 各自的值。该编号用于很容易地识别器件的衍生型号信息，例如封装类型、温度范围、Flash 大小、 RAM 大小和外设的可用性。

13.3 电源管理单元 (PCU)

电源管理控制在电源控制单元 (PCU) 中执行。

13.3.1 功能描述

XMC1300 使用 1.8 - 5.5V (V_{DDP}) 的单一外部电源运行。主电源电压由电源看门狗管理。I/O 直接使用外部电源电压运行。内核电压 (V_{DDC}) 由片上嵌入式稳压器 (EVR) 产生。内核电压的安全电压范围由一个电源验证电路管理，该电路是 EVR 的一部分。

13.3.2 系统状态

系统有下述综合系统状态：

- 关闭
- 活动
- 休眠
- 深度休眠

图 13-3 示出了状态图和状态之间的转换。

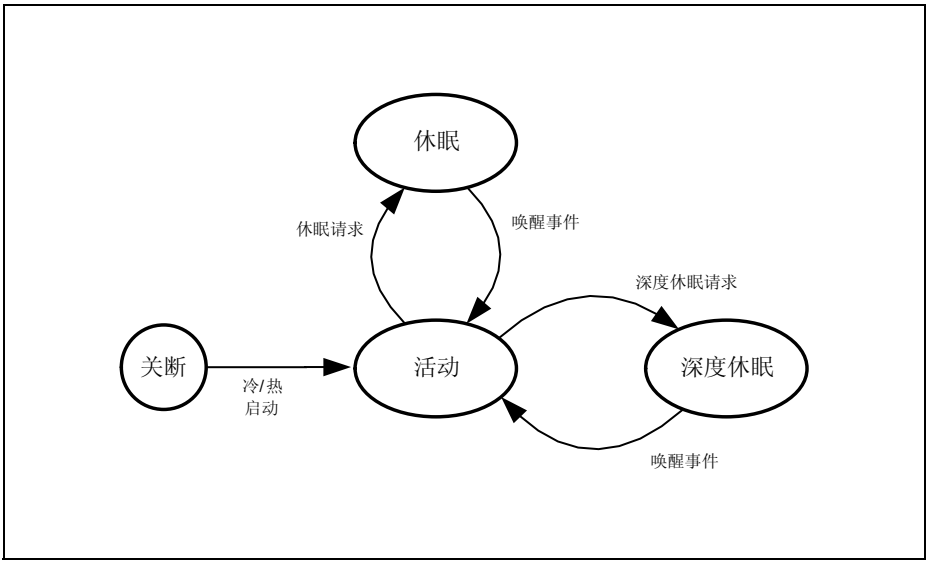


图 13-3　　系统状态图

活动状态

活动状态是正常工作状态。系统完全供电。CPU 通常使用一个高速时钟运行。系统时钟可以根据应用需求降低速度。可以通过关闭未用外设的时钟使其停止工作。

休眠状态

系统的休眠状态与 CPU 的休眠状态一致。通过 CPU 的 WFI 或 WFE 指令进入这一状态。在该状态下，CPU 的时钟被停止。为了省电，可以在进入休眠状态之前通过寄存器 **CGATSET0** 关闭那些不需要在休眠期间工作的外设的时钟。

可以在进入休眠状态前，通过位 NVMCONF.NVM_ON 在活动状态将 Flash 置于关闭模式，以进一步降低功耗。然而，在进入休眠状态前和从休眠状态唤醒后，用户代码必须在 SRAM 中执行。

为避免由于关闭 Flash 而将代码执行切换到 SRAM，可以使用寄存器 **PWRSVCR**。当 FPD 位被置 1 时，Flash 仅在设备进入休眠状态时被关闭。关闭操作在内核执行 WFI/WFE 指令后被执行。在检测到唤醒事件后，系统将恢复到先前的状态，即在 CPU 能够继续获取和执行代码前，Flash 重新变为可工作状态。在这种情况下，用户代码可以在 Flash 中执行而不需切换到 SRAM 中。使用该方法时唤醒时间变长因为 Flash 达到活动状态需要一定时间。

外设可以不受影响地继续运行，最后产生一个事件来唤醒 CPU。在用户调试模式 (UMD) 下 或者用户调试模式和 HAR (UMHAR)，调试 HALT 请求也能唤醒 CPU。任何相应配置过的中断都会通过 NVIC 或 M0 调试系统将 CPU 带回到运行状态。

深度休眠状态

深度休眠状态的进入机制与休眠状态相同，另外用户代码可以在系统控制寄存器中使能深度休眠状态。该状态类似于休眠状态，不同之处是， PCLK 和 MCLK 在深度休眠状态会被切换到一个慢的待机时钟，并且 DCO1 会被置于掉电模式。

模拟比较器也能在深度休眠状态下保持活动。可以将模拟比较器切换到低功耗模式以节省功耗。

同上节的解释一样，通过 NVMCONF.NVM_ON 或 PWRSVCR.FPD 关闭 Flash 同样适用于深度休眠模式。

继续运行的外设将使用慢速的待机时钟，能最后产生一个事件去唤醒 CPU。在用户调试模式 (UMD) 下或者用户调试模式和 HAR (UMHAR)，调试 HALT 请求也能唤醒 CPU。任何相应配置过的中断都会通过 NVIC 或 M0 调试系统将 CPU 带回到运行状态。在 CPU 被唤醒后，时钟系统恢复到先前的活动状态配置。活动的外设会用恢复后的时钟配置运行。

在深度休眠状态， SRAM 的内容被保持。

注： 建议在进入深度休眠模式前降低 PCLK 和 MCLK 的频率，以防止突然的负载变化引起掉电复位。

13.3.3 嵌入式稳压器 (EVR)

EVR 用外部电源电压 V_{DDP} 产生内核电压 V_{DDC}。EVR 为输入电压 V_{DDP} 提供了 2 个电源监视器。产生的内核电压 V_{DDC} 由电源验证电路 (PV) 监视。

13.3.4 上电复位

一旦 V_{DDP} 高于所规定的最低电平， EVR 即开始工作。当外部电压 V_{DDP} 和产生的电压 V_{DDC} 都高于复位阈值并达到标称值时， EVR 结束复位。

13.3.5 电源验证

电源验证电路监视内部的内核电源电压 V_{DDC}。它监视内核电压是否高于能保证安全工作的电压阈值 V_{DDCBO}。只要电压低于这个阈值电平，就会产生一个掉电复位。

13.3.6 电源电压监测

EVR 中有 2 个常用来监视 V_{DDP} 的检测器，即外部电压检测器 (VDEL) 和外部掉电探测器 (BDE) 。

VDEL 检测器将电源电压与一个预警阈值电压进行比较。阈值电平可以通过寄存器 ANAVDEL.VDEL_SELECT 编程。如果检测到电源电压低于这个阈值电压，则会触发一

个中断（如果被使能），并将 SRRAW 寄存器中的标志位 VDDP 置 1。标志位 VDESR.VDDPPW 指示检测器的输出情况。

BDE 检测器用于在电源电压 V_{DDP} 低于所定义的阈值时触发掉电复位。类似地， BDE 检测器也用于在上电阶段当 V_{DDP} 高于规定的阈值时确保正确启动。

数据手册定义了标称值和应用的滞回电压。

13.3.7 V_{DDC} 在负载变化期间的响应

在 XMC1300 中，当负载增加时内核电压电平 V_{DDC} 会低于典型阈值，当负载降低时内核电压电平 V_{DDC} 会高于典型阈值。VDROP 和 VCLIP 这两个检测器分别用于监视内核电压电平的下限和上限 (检测器的细节在下一部分描述)。当 VDDC 下降到低于 VDROP 阈值电压时，会产生一个 VDROP 事件。当 VDDC 上升到高于 VCLIP 阈值电压时，会产生一个 VCLIP 事件。如果通过 SRMSK 寄存器被使能，这些事件中的每一个都能够触发其专有中断，事件的状态可通过 SRRAW 寄存器监视。

在 XMC1300 中，下面的情形由于负载改变可能触发一个 VDROP/VCLIP 事件：

- 通过 CLKCR 寄存器改变 MCLK 和 PCLK 的频率
- 通过 CGATSET0/CGATCLR0 寄存器使能 / 关闭外设时钟

当发生突然的负载改变时 (< 4* 基本负载 或 > 0.25* 基本负载)(TBC)，不管负载是增加还是减少， EVR 都需要时间 (15 微秒) 来调整内核电压，使其回到稳定的标称电压。在这段时期里，要求保持当前的负载，不允许负载改变。 CRCLK 寄存器里中的状态位 VDDC2LOW 和 VDDC2HIGH 用于指示电压是否稳定。

注： 不建议负载增加到超过 4 倍的基本负载或者负载减小到低于 0.25 倍的基本负载。如果需要负载变化超过额定值，建议分步改变负载，每一步的负载变化都在限制值内。例如，从当前 1mA 的基本负载到最终 16mA 的负载至少需要两步。一步是从 1mA 到 4mA，接着另一步是从 4mA 到 16mA。

VDDC2LOW 和 VDDC2HIGH 状态位由一个 10 位计数器产生，该计数器使用 DCO1, 64MHz 时钟作为输入时钟。它实现计数 16 微秒 (默认)，这是在 VDROP 或 VCLIP 事件发生后获得稳定的 V_{DDC} 所需要的时间。该计数器的长度可以根据负载的变化量通过位域 CLKCR.CNTADJ 改变。在使用一个稳定时钟的情况下，负载变化越大，用户需要等待的时间越长。每个模块的电流消耗请参见数据手册。

使用上面的例子，在因编程某一配置而引起负载发生从 1mA 到 4mA 的改变后，用户可以查询 VDDC2LOW (CNTADJ=3FF$_H$) ，以确保调整 EVR 所需要的 16 毫秒 (最大) 。在 VDDC2LOW 被设置为 0 后，可以执行另一个从 4mA 到 16mA 的负载改变，每一步负载改变都要重复这一周期。

在 VDROP 事件期间，会发生时钟消隐，详细描述见**页 13-14 “ 时钟消隐 ”**。在 VCLIP 事件期间， CPU 时钟和外设时钟继续运行。

注： 当溢出事件发生时，如果 VDROP=1 (基于 CNTADJ 值的溢出时间比器件停留在 VDROP 事件里的时间更短)，则 10 位计数器会使用 CNTADJ 值自动重启。

注： 在深度休眠状态， VDROP 和 VCLIP 检测器被禁止。

13.3.8 Flash 功率控制

闪存模块可以被关闭以减小静态功耗。在休眠或深度休眠状态，Flash 模块是否被置于休眠状态取决于寄存器 **PWRSVCR** 的设置。用户必须评估减小的漏电流和更长的启动时间。此外，也可以在进入这些省电模式之前用 NVMCONF.NVM_ON 将 Flash 置于休眠状态。

13.4 复位控制单元 (RCU)

复位控制单元控制所有与复位有关的功能，包括：

- 对各种复位请求源的复位断言
- 复位后检测复位源
- 外设的选择性复位

13.4.1 功能描述

XMC1300 有以下系统复位类型：

- 主复位， $\overline{\text{MRESET}}$
- 系统复位， $\overline{\text{SYSRESET}}$

主复位 , $\overline{\text{MRESET}}$

主复位的触发源有：

- 上电复位 (PORST)
- V_{DDP} 或 V_{DDC} 欠压复位 (也称为掉电复位)
- 软件主复位，通过将位 RSTCON.MRSTEN 置 1

器件的完全复位由主复位执行。主复位在加电时由上电复位触发。当电源 V_{DDP} 上升并超过 V_{DDP} 和 V_{DDC} 电压阈值时，上电复位解除。当 V_{DDP} 电压或 V_{DDC} 电压下降到低于复位阈值时，上电复位 (也称为掉电复位) 重新被触发。另外，主复位可以通过将位 RSTCON.MRSTEN 置 1 来触发。

主复位的触发源也会触发系统复位 $\overline{\text{SYSRESET}}$。

系统复位 , $\overline{\text{SYSRESET}}$

系统复位的触发源有：

- 软件复位，通过 Cortex M0 的应用中断和复位控制寄存器 (AIRCR)
- 在 RCU 中被使能的来自 Cortex M0 的锁定信号
- 看门狗复位
- 被使能的存储器奇偶校验错误
- 被使能的 Flash ECC 双位错误
- 被使能的时钟丢失
- 主复位的触发源

系统复位影响几乎所有的逻辑。唯一的例外是 RCU 寄存器和有调试探针时的调试系统。复位被扩展到由实现要求所定义的长度。

在正常工作模式下，当调试探针不存在时，调试系统通过系统复位来复位，当调试探针存在时，系统复位不影响调试系统。

13.4.2 复位状态

EVR 为 RCU 提供上电复位的原因。可以在恢复运行后通过读寄存器 **RSTSTAT** 检查复位原因。该寄存器还指示引起系统复位触发的事件源。

RCU 的所有寄存器都只能由主复位进行复位，但 RSTSTAT 和 RSTCON 寄存器例外。RSTSTAT 寄存器只能由上电复位来复位， RSTCON 可以由任何复位类型复位。

*注： 强烈建议通过寄存器位 **RSTCLR**.RSCLR 来清除复位状态，以确保对下一次复位起因的清晰指示。*

表 13-3 列出了所有复位信号及其来源和对系统各部分的影响

表 13-3 复位一览表

模块 / 功能	上电复位	通过软件位的主复位	系统复位
CPU 内核	是	是	是
SCU	是	是	是，复位标志位除外
外设	是	是	是
调试系统	是	是	见脚注[1)2)]
端口控制	是	是	是
SRAM	受影响，不可靠	不受影响	不受影响
Flash	是	是	是
EVR	是	否[3)]	否
时钟系统	是	是	是

1) 调试探针不存在，调试系统将会复位。

2) 即使调试探针存在，每次复位后都禁止访问调试接口。更多细节请查看调试系统一章的热复位一节。

3) EVR 的电源不受影响，因此 EVR 的完全复位是不可能的。然而，它将部分受 ANACTRL 模块中复位的影响。

13.5 时钟控制单元 (CCU)

13.5.1 特性

时钟控制单元 CCU 具有以下功能：

- 专用的 RTC 和待机时钟
- 时钟监视
  - 振荡器看门狗
- 系统频率的宽范围频率缩放
- 独立的外设时钟门控

13.5.2 时钟系统和控制

图 13-4 示出了 XMC1300 中时钟系统的框图。它包含两个振荡器 (DCO1[1] 和 DCO2) 和一个时钟控制单元 (CCU)。DCO1 有一个运行在 64MHz 的时钟输出 dco1_dclk 。DCO2 用于产生运行在 32.768kHz 的待机时钟。主时钟 MCLK 和快速外设时钟 PCLK 由 dco1_dclk 产生。 PCLK 运行在与 MCLK 相同的频率或 MCLK 频率的两倍频率，可通过 CLKCR.PCLKSEL 选择。

图 13-4 示出了运行在 PCLK 域的外设列表。除 RTC 和 WDT 外，其余外设都使用 MCLK 作为时钟，与内核和总线系统的时钟相同。 RTC 和 WDT 运行在来自待机时钟的 32.768kHz 频率，待机时钟与 MCLK 和 PCLK 时钟不同步。

1) DCO1 时钟的输出精度可以通过校准来改善，这种校准基于温度传感器给出的晶片温度。详细描述见 13.5.4 节。

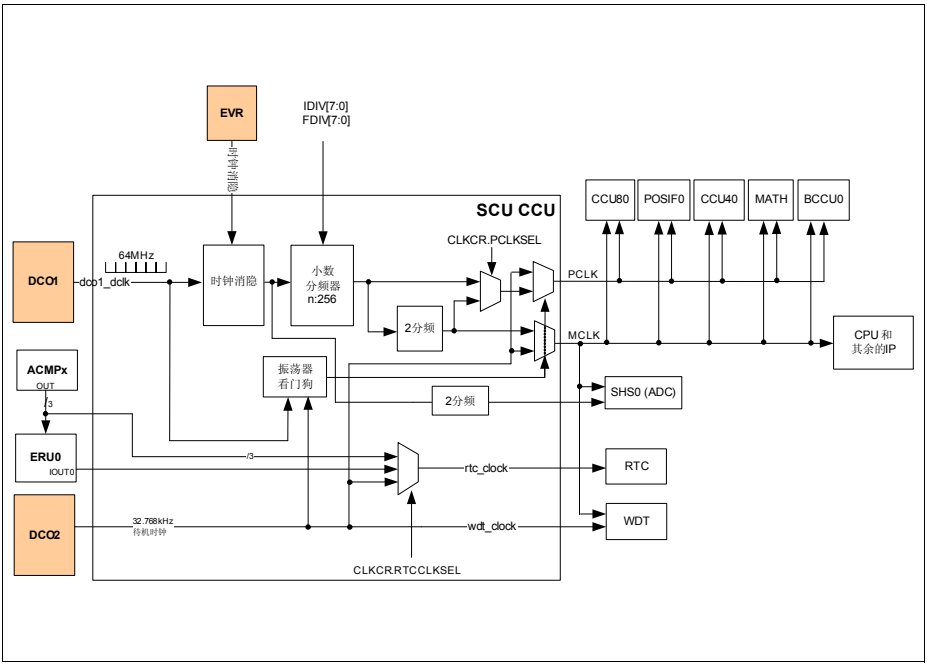


图 13-4　　时钟系统框图

注：　当发生时钟丢失事件时，SHS(ADC) 时钟不能切换到待机时钟源。

注：　RTC_extclk 在 RTC 一章进一步描述。

小数分频器

MCLK 和 PCLK 的频率可通过小数分频器编程。PCLK 的频率范围从 125kHz 到 64MHz，MCLK 的频率范围从 125kHz 到 32MHz。

下面的公式用于计算 MCLK 时钟频率。

(13.1)

$$\left(MCLK = \frac{dco\_dclk}{(2) \times \left(IDIV + \frac{FDIV}{256} \right)} \right) \text{for IDIV} > 0$$

当 CLKCR.PCLKSEL 被置 1 时，用下面的公式可以计算 PCLK 时钟频率，该频率是 MCLK 的两倍。

(13.2)

$$\left(PCLK = \frac{dco\_dclk}{\left(IDIV + \frac{FDIV}{256} \right)} \right) \text{for IDIV} > 0$$

IDIV 代表来自位域 CLKCR.IDIV 的无符号 8 位整数， FDIV/256 定义 CLKCR.FDIV 中小数分频器的选择。MCLK 和 PCLK 的改变能在 2 个时钟周期内完成。当改变 MCLK 时钟和 PCLK 时钟的频率时， 建议禁止所有中断，以防止任何可能导致 Flash 操作不成功的 Flash 访问。

注： 改变 MCLK 和 PCLK 的频率可能导致负载改变，负载变化能引起 时钟消隐发生 。更详细的信息参见 ***时钟消隐*** *和* ***V_{DDC} 在负载变化期间的响应****。*

时钟消隐

为了防止在负载突然增加的情况下发生欠压复位，使用了时钟消隐电路。它冻结去小数分频器的时钟输入，以调节负载的变化。时钟消隐时间较长时引起的抖动很小。

当检测到 V_{DDC} 低于 VDROP 阈值时，时钟消隐被激活。一旦 V_{DDC} 高于这个阈值，被使能的时钟重新恢复。该电压下降检测器是 EVR 的一部分，默认在任何复位后都被激活。

为了监视时钟消隐活动，用户可以使能中断，但只能在时钟恢复后进入 ISR。

除了使用时钟消隐功能来防止欠压复位外，还要求系统在 16 微秒 (最大) 的时间内保持当前负载，不允许有进一步的负载改变。负载改变期间内核电压行为的详细描述见页 13-9 “V_{DDC} 在负载变化期间的响应 ”中描述。

如在页 13-9 “V_{DDC} 在负载变化期间的响应 ”中所述，状态位 CLKCR.VDDC2LOW 用于向用户指示：内核电压 V_{DDC} 低于标称电压，EVR 正在对其进行调节。在此期间，时钟可能未运行在所选速度，建议在对当前负载继续进行任何大的改变前查询该位。

注： 建议在进入深度休眠状态前降低 PCLK 和 MCLK 的频率，以防止在进入该状态时发生突然的负载改变，从而引起欠压复位 。

13.5.2.1 振荡器看门狗

振荡器看门狗 (OWD) 使用待机时钟作为参考时钟来监视 DCO1 的时钟频率。可以通过 OSCCSR.OWDEN 禁止振荡器看门狗。通过置位 OSCCSR.OWDRES[1]，可以重新启动对 DCO1 时钟频率的检测。检测状态输出在几个备用时钟频率周期后才生效。当 OWD 被禁止时，检测状态会被复位，这时不可能进行检测。

如果在进入深度休眠模式前使能了 OWD，它将在进入深度休眠模式时被硬件自动禁止，在退出深度休眠状态后又会被重新使能。器件从深度休眠状态唤醒后，对 DCO1 时钟频率的检测在被复位后又重新启动。

13.5.2.2 时钟丢失检测和时钟恢复

在正常工作期间，当振荡器看门狗(OWD)检测到DCO1频率低于50MHz或高于68.5MHz时，说明发生了时钟丢失。在这种情况下，会产生一个中断 （如果被使能） 。同时，振荡器状态标志 OSCCSR.OSC2L 或 OSCCSR.OSC2H 被置 1，系统时钟将由 32.768kHz 的待机时钟提供。此时可以执行紧急程序来安全地关闭系统。当时钟丢失发生时，除了触发中断外，还可以触发系统复位，前提是通过 **RSTCON**.LOCRSTEN 使能了时钟丢失复位。

*注： 在时钟丢失事件期间，只有在 DCO2 仍然运行 (>0MHz) 的情况下才能切换到待机时钟。位 **SRRAW**.SBYCLKFI 指示待机时钟的故障状态。*

XMC1300 会一直保持在时钟丢失状态，直到下一次复位或成功执行了时钟恢复之后。时钟恢复可以通过重新启动时钟丢失检测 （将位 OSCCSR.OWDRES 置 1）来执行。当检测到一个高于 50MHz 且低于 68.5MHz 的稳定振荡器频率时，OSC2L 和 OSC2H 会被清 0， MCLK 将自动切换到 DCO1 时钟源。

13.5.2.3 备用时钟故障

当 OWD 检测到一个待机时钟故障，即待机时钟停止运行时 (~0kHz)，说明发生了待机时钟故障事件。位 SRRAW.SBYCLKFI 被置 1 ，并且会通过 SCU_SR1 服务请求触发一个中断 （如果该中断在寄存器 SRMSK 中被使能）。

1) 用于在重新启动检测之前清除状态位 SRRAW.LOCI 和 SRRAW.SBYCLKFI。

13.5.2.4 系统时钟的启动控制

当 XMC1300 在复位后启动时，系统频率由 DCO1 振荡器提供。复位后，CPU 运行在 8MHz 的默认频率，用户可以改变用于执行 SSW （启动软件）的频率，在 Flash 单元 1000 1010_H 定义这个频率 。更详细的信息参见引导和启动一章。

13.5.3 时钟门控控制

给外设的时钟可以被单独控制，可以通过寄存器 **CGATSET0** 停止系统某些部分的运行。在发生一次主复位后，只有内核、存储器、 SCU 和端口外设的时钟未被停止。其余外设的时钟默认被停止。复位后，用户可用 SSW 选择单独使能每个模块的时钟，这种选择在 Flash 单元 1000 1014_H 中定义。更详细的信息参见引导和启动一章。

在模块时钟使能或门控期间的负载变化

使能或者门控给外设的时钟能引起负载的改变，这可能导致时钟消隐发生。此外， 高于当前负载 4 倍以上的负载改变可能需要系统在 16 微秒 (最大) 内保持当前的负载，不允许有进一步的负载改变。更详细的信息见 **V_{DDC} 在负载变化期间的响应**。

在休眠和深度休眠模式下的模块时钟门控

建议使用寄存器**CGATSET0**去控制在休眠模式或深度休眠模式期间所不需要的模块的时钟。这些模块在进入休眠或深度休眠模式之前必须被禁止。另外， PCLK 和 MCLK 将被切换到一个慢速待机时钟， DCO1 在深度休模式会进入掉电模式。

13.5.4 基于温度校准 DCO

在 XMC1300 中，可以在运行时校准 DCO1 的时钟频率，以获得更高的精度。校准需要使用片内温度传感器测量温度，可以使用下面所示的公式获得一个偏移值：

OFFSET = tbd [steps]

在公式中有 2 个常数：DCO_ADJL_RT 和 DCO_ADJL_HT，这些值存放在如**表 13-4** 所示的 Flash 配置页。

表 13-4 Flash CS0 中的 DCO 校准数据

地址	长度	作用
		DCO 校准数据：
$1000'0F40_H$	2 B	DCO_ADJL_RT 在室温下测量的频率低调整值 (5 个最低位)
$1000'0F42_H$	2 B	DCO_ADJL_HT 在高温下测量的频率低调整值 (5 个最低位)

将偏移值输入到寄存器位域 **ANAOFFSET**.ADJL_OFFSET 以启动 DCO1 校准。这个位域是受保护的。DCO1 将用大约 5 微秒 (最大) 的时间使时钟调整到新频率。建议在待定 °C 的每一步调整 DCO1。

13.6 服务请求的产生

SCU 模块提供 3 个服务请求输出 SR[2:0]。SR0 用于系统关键请求，例如时钟丢失事件。SR1 用于公共 SCU 请求，例如 TSE 请求。SR2 用于 ACMP 和 ORC 请求。服务请求输出 SR[2:0] 连接到嵌套向量中断控制器 (NVIC) 中的中断节点。

更详细的信息参见 **13.2.1 节**。

13.7 调试行为

在使用外部调试探针执行调试活动时，SCU 模块基于不受来自 SCU 的 HALTED 信号的影响。

13.8 电源、复位和时钟

SCU 模块实现的功能包括受模块直接控制的各种功能以及通过专用接口的、在不同的电源域、时钟域和复位域实例化的功能。这些模块在功能上被认为是 SCU 的一部分，因此在这个意义上 SCU 也被认为是一个多域电路。

电源域 :

电源域被分成适当的电源分离区域。

- 内核域由 V_{DDC} 电压供电
- 焊盘域由 V_{DDP} 电压供电

时钟域 :

所有的跨域接口都实现了信号同步。

- 内部的 SCU 时钟是 MCLK，总是与 CPU 时钟相同
- RTC 和寄存器镜像接口时钟是从 DCO2 振荡器产生的 32.786kHz 时钟

复位域 :

所有复位都在内部与各自的时钟同步。

- 系统复位 ($\overline{\text{SYSRESET}}$) 可以复位 SCU 中的大多数逻辑，能够被多种源触发 (更详细的信息请参见**复位控制单元 (RCU)** 一节)
- 主复位 ($\overline{\text{MRESET}}$) 有助于产生系统复位，由内核域的上电序列触发。

13.9 寄存器

本节描述 SCU 的寄存器，其中一些位于 ANACTRL 模块。大多数寄存器由 $\overline{\text{SYSRESET}}$ 复位信号复位，但某些寄存器只能由上电复位来复位。SCU 寄存器可通过 AHB-lite 总线访问。ANACTRL 寄存器可通过 APB 总线访问。ANACTRL 寄存器的名称用 “ANA” 开始。

表 13-5 部分 SCU 寄存器的基地址

简称	描述	偏移地址[1)]
GCU 寄存器	综合控制单元的偏移地址	0000_H
PCU 寄存器	电源控制单元的偏移地址	0200_H
CCU 寄存器	时钟控制单元的偏移地址	0300_H
RCU 寄存器	复位控制单元的偏移地址	0400_H
RTC 寄存器	实时时钟模块的偏移地址	$0A00_H$
ANACTRL 寄存器	ANACTRL 寄存器的偏移地址	1000_H

1) 绝对寄存器地址计算如下：
模块基地址 + 子模块偏移地址 (如该列所示) + 寄存器偏移地址

下列对 SCU/ANACTRL SFR 的访问会导致一个 AHB/APB 错误响应：

- 对未定义地址的读或写访问
- 对只读寄存器的写访问
- 对启动保护寄存器的写访问

表 13-6 寄存器地址空间

模块	基地址	结束地址	备注
SCU	$4001\ 0000_H$	$4001\ FFFF_H$	系统控制单元寄存器

表 13-7 寄存器一览表

简称	描述	偏移地址[1]	访问方式		描述见
			读	写	
PCU 寄存器 (ANACTRL)					
ANAVDEL	电压检测器控制寄存器	1050_H	U, PV	U, PV	页 13-20
PCU 寄存器 (SCU)					
VDESR	电压检测器状态寄存器	0000_H	U, PV	U, PV	页 13-21
CCU 寄存器 (SCU)					
CLKCR	时钟控制	0000_H	U, PV	U, PV, BP	页 13-22
PWRSVCR	节电控制寄存器	0004_H	U, PV	U, PV	页 13-24
CGATSTAT0	外设 0 的时钟门控状态	0008_H	U, PV	U, PV	页 13-25
CGATSET0	外设 0 的时钟门控置位	$000C_H$	U, PV	U, PV, BP	页 13-26
CGATCLR0	外设 0 的时钟门控清除	0010_H	U, PV	U, PV, BP	页 13-28
OSCCSR	振荡器控制和状态寄存器	0014_H	U, PV	U, PV	页 13-29
CCU 寄存器 (ANACTRL)					
ANAOFFSET	DCO1 偏移寄存器	$106C_H$	U, PV	U, PV, BP	页 13-31
RCU 寄存器 (SCU)					
RSTSTAT	复位状态	0000_H	U, PV	BE	页 13-32
RSTSET	复位置位寄存器	0004_H	U, PV	U, PV	页 13-33
RSTCLR	复位清除寄存器	0008_H	U, PV	U, PV	页 13-33
RSTCON	复位控制寄存器	$000C_H$	U, PV	U, PV	页 13-35
GCU 寄存器 (SCU)					
ID	模块标识寄存器	0008_H	U, PV	BE	页 13-36
IDCHIP	芯片 ID	0004_H	U, PV	SP	页 13-37
DBGROMID	DBGROMID	0000_H	U, PV	SP	页 13-38
SSW0	SSW 支持寄存器	0014_H	U, PV	U, PV	页 13-39
CCUCON	CCUx 全局启动控制寄存器	0030_H	U, PV	U, PV	页 13-39
SRRAW	RAW 服务请求状态	0038_H	U, PV	BE	页 13-40

表 13-7 寄存器一览表 (cont'd)

简称	描述	偏移地址[1]	访问方式		描述见
			读	写	
SRMSK	服务请求屏蔽	$003C_H$	U, PV	U, PV	页 13-43
SRCLR	服务请求清除	0040_H	U, PV	U, PV	页 13-45
SRSET	服务请求置位	0044_H	U, PV	U, PV	页 13-48
PASSWD	位保护寄存器	0024_H	U, PV	U, PV	页 13-50
MIRRSTS	镜像更新状态寄存器	0048_H	U, PV	BE	页 13-51
PMTSR	奇偶校验存储器测试选择寄存器	0054_H	U, PV	U, PV	页 13-53

1) 绝对寄存器地址计算如下：
模块基地址 + 子模块偏移地址 + 偏移地址 (如该列所示)

13.9.1 PCU 寄存器 (ANACTRL)

ANAVDEL

电压检测器控制寄存器。

ANAVDEL
电压检测器控制寄存器 **(1050_H)** **复位值 :$001C_H$**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											VDEL_EN	VDEL_TIM_ADJ		VDEL_SELECT	
r											rw	rw		rw	

域	位	类型	描述
VDEL_SELECT	1:0	rw	**VDEL 范围选择** 这些位设置 VDDP 的范围。 00B 2.25V 01B 3.0V 10B 4.4V

域	位	类型	描述
VDEL_TIM_ADJ	3:2	rw	**VDEL 定时设置** 这些位控制 VDEL 的反应速度。 该值由特性决定。 00B 典型值 1µs - 最慢的响应时间 01B 典型值 500ns 10B 典型值 250ns 11B 无延迟 - 最快的响应时间。
VDEL_EN	4	rw	**VDEL 单元使能** 0B 禁止 VDEL 1B 使能 VDEL
0	15:5	r	**保留** 读出值为 0；应写入 0。

13.9.2 PCU 寄存器 (SCU)

VDESR

电压检测器状态寄存器。

VDESR

电压检测器状态寄存器 **(0200_H)** **复位值：0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														**VDDPPW**	**VCLIP**
r														rh	rh

域	位	类型	描述
VCLIP	0	rh	**VCLIP 指示** VCLIP 监视位。 0_B VCLIP 不活动 1_B VCLIP 活动

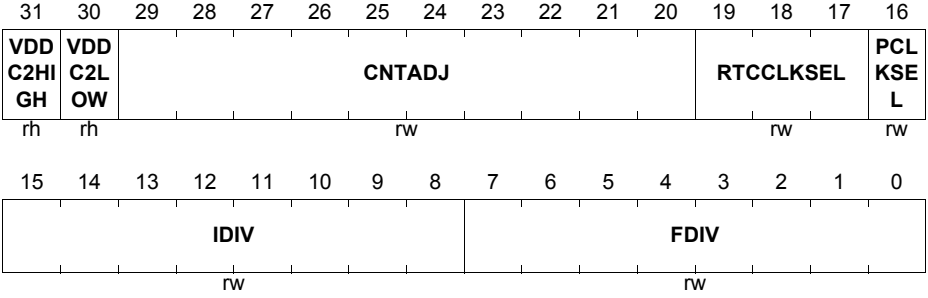
域	位	类型	描述
VDDPPW	1	rh	**VDDPPW 指示** 0_B VDDP 高于预警阈值 1_B VDDP 低于预警阈值
0	[31:2]	r	**保留** 读出值为 0；应写入 0。

13.9.3 CCU 寄存器 (SCU)

CLKCR

时钟控制寄存器。

CLKCR
时钟控制寄存器 **(0300_H)** **复位值 : 3FF0 0400_H**



域	位	类型	功能
FDIV	[7:0]	rw	**小数分频器选择** 选择小数分频器为 n/256，其中 n 是 FDIV 的值，范围是 0 到 255。 例如，向 FDIV 写入 0001_B 选择小数分频器为 1/256。 该位受在存储器组织一章描述的位保护机制保护。 *注： 如果 IDIV = 00_H，小数分频器不起作用。*

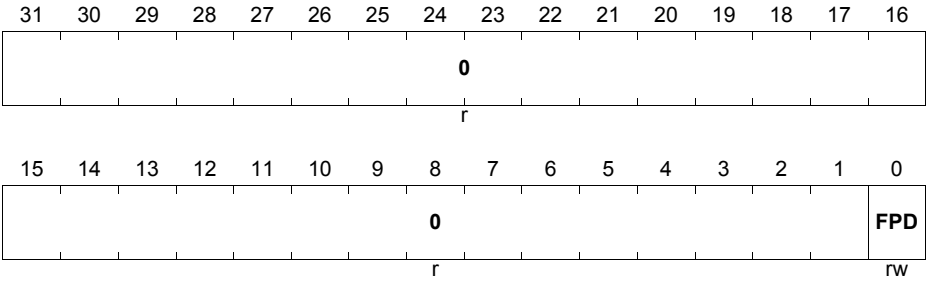
域	位	类型	功能
IDIV	[15:8]	rw	**分频器选择** 00_H 分频器被旁路。 01_H 1; MCLK = 32 MHz 02_H 2; MCLK = 16 MHz 03_H 3; MCLK = 10.67 MHz 04_H 4; MCLK = 8 MHz FE_H 254; MCLK = 126 kHz FF_H 255; MCLK = 125.5 kHz 该位受在存储器组织一章描述的位保护机制保护。
PCLKSEL	16	rw	**PCLK 时钟选择** 0_B PCLK = MCLK 1_B PCLK = 2 x MCLK 该位受在存储器组织一章描述的位保护机制保护。
RTCCLKSEL	[19:17]	rw	**RTC 时钟选择** 000_B 32.768kHz 备用时钟 001_B 来自 ERU0.IOUT0 的 32.768kHz 外部时钟 010_B 来自 ACMP0.OUT 的 32.768kHz 外部时钟 011_B 来自 ACMP1.OUT 的 32.768kHz 外部时钟 100_B 来自 ACMP2.OUT 的 32.768kHz 外部时钟 101_B 保留 110_B 保留 111_B 保留 该位受在存储器组织一章描述的位保护机制保护。
CNTADJ	[29:20]	rw	**计数器调整** 000_H 1 个 DCO1 时钟周期， 64MHz 时钟 001_H 2 个 DCO1 时钟周期， 64MHz 时钟 002_H 3 个 DCO1 时钟周期， 64MHz 时钟 003_H 4 个 DCO1 时钟周期， 64MHz 时钟 004_H 5 个 DCO1 时钟周期， 64MHz 时钟 $3FE_H$ 1023 个 DCO1 时钟周期， 64MHz 时钟 $3FF_H$ 1024 个 DCO1 时钟周期， 64MHz 时钟
VDDC2LOW	30	rh	**VDDC 过低** 0_B VDDC 不低，小数分频器输入时钟运行在目标频率 1_B VDDC 过低，小数分频器输入时钟未运行在目标频率
VDDC2HIGH	31	rh	**VDDC 过高** 0_B VDDC 不高 1_B VDDC 过高

PWRSVCR

定义在深度休眠模式或休眠模式期间一些系统行为方面的配置寄存器。原系统状态在从休眠模式或深度休眠模式唤醒时得到恢复。

PWRSVCR

节电控制寄存器 **(0304_H)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
FPD	0	rw	**Flash 下电** 0_B 无效 1_B 进入节电模式时 Flash 下电。当唤醒时，CPU 能够从 Flash 中获取代码。
0	[31:1]	r	**保留** 读出值为 0。

CGATSTAT0

XMC1300 外设的时钟门控状态。复位后，该寄存器中列出的所有外设都不运行。它们的模块时钟是受门控的。

该寄存器中的每一位都受存储器组织一章描述的位保护机制的保护。

CGATSTAT0
外设 0 时钟门控状态 **(0308_H)** **复位值 : 0000 07FF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					RTC	WDT	MATH	POSIF0	0		BCCU0	USIC0	CCU40	CCU80	VADC
r					r	r	r	r	r		r	r	r	r	r

域	位	类型	描述
VADC	0	r	**VADC 和 SHS 的门控状态** 0_B 门控无效 1_B 门控生效
CCU80	1	r	**CCU80 的门控状态** 0_B 门控无效 1_B 门控生效
CCU40	2	r	**CCU40 的门控状态** 0_B 门控无效 1_B 门控生效
USIC0	3	r	**USIC0 的门控状态** 0_B 门控无效 1_B 门控生效
BCCU0	4	r	**BCCU0 的门控状态** 0_B 门控无效 1_B 门控生效
POSIF0	7	r	**POSIF0 的门控状态** 0_B 门控无效 1_B 门控生效

域	位	类型	描述
MATH	8	r	**MATH 的门控状态** 0_B 门控无效 1_B 门控生效
WDT	9	r	**WDT 的门控状态** 0_B 门控无效 1_B 门控生效
RTC	10	r	**RTC 的门控状态** 0_B 门控无效 1_B 门控生效
0	[6:5], [31:11]	r	**保留**

CGATSET0

XMC1300 外设的时钟门控使能寄存器。向所选位写 1 将使能对应时钟的门控，写 0 无效。该寄存器中的每一位都受存储器组织一章描述的位保护机制的保护。

CGATSET0
外设 0 时钟门控置位 **($030C_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					**RTC**	**WDT**	**MATH**	**POSIF0**	**0**		**BCCU0**	**USIC0**	**CCU40**	**CCU80**	**VADC**
r					w	w	w	w	r		w	w	w	w	w

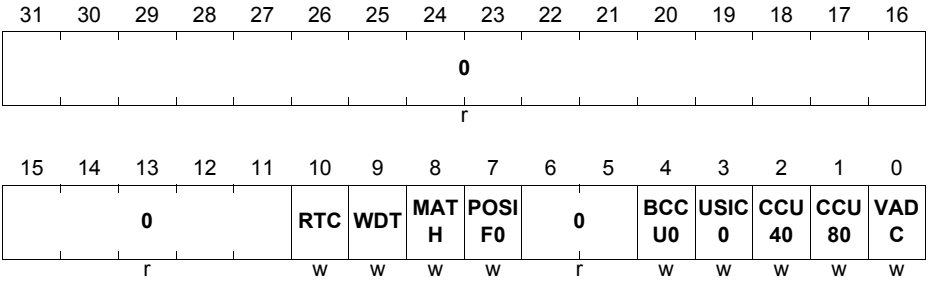
域	位	类型	描述
VADC	0	w	**VADC 和 SHS 门控置位** 0_B 无效 1_B 使能门控
CCU80	1	w	**CCU80 门控置位** 0_B 无效 1_B 使能门控

域	位	类型	描述
CCU40	2	w	**CCU40 门控置位** 0_B 无效 1_B 使能门控
USIC0	3	w	**USIC0 门控置位** 0_B 无效 1_B 使能门控
BCCU0	4	w	**BCCU0 门控置位** 0_B 无效 1_B 使能门控
POSIF0	7	w	**POSIF0 门控置位** 0_B 无效 1_B 使能门控
MATH	8	w	**MATH 门控置位** 0_B 无效 1_B 使能门控
WDT	9	w	**WDT 门控置位** 0_B 无效 1_B 使能门控
RTC	10	w	**RTC 门控置位** 0_B 无效 1_B 使能门控
0	[6:5], [31:11]	r	**保留**

CGATCLR0

XMC1300外设的时钟门控禁止寄存器，向所选位写1将禁止对应时钟的门控，写0无效。

CGATCLR0
外设 0 时钟门控清除 **(0310_H)** **复位值：$0000\ 0000_H$**



域	位	类型	描述
VADC	0	w	**VADC 和 SHS 门控清除** 0_B 无效 1_B 禁止门控
CCU80	1	w	**CCU80 门控清除** 0_B 无效 1_B 禁止门控
CCU40	2	w	**CCU40 门控清除** 0_B 无效 1_B 禁止门控
USIC0	3	w	**USIC0 门控清除** 0_B 无效 1_B 禁止门控
BCCU0	4	w	**BCCU0 门控清除** 0_B 无效 1_B 禁止门控
POSIF0	7	w	**POSIF0 门控清除** 0_B 无效 1_B 禁止门控
MATH	8	w	**MATH 门控清除** 0_B 无效 1_B 禁止门控

域	位	类型	描述
WDT	9	w	**WDT 门控清除** 0_B 无效 1_B 禁止门控
RTC	10	w	**RTC 门控清除** 0_B 无效 1_B 禁止门控
0	[6:5], [31:11]	r	**保留**

OSCCSR

振荡器控制和状态寄存器。

OSCCSR
振荡器控制和状态寄存器

(0314_H) **复位值 : 0000 000X$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0														**OWDEN**	**OWDRES**
r														rw	rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														**OSC2H**	**OSC2L**
r														rh	rh

域	位	类型	描述
OSC2L	0	rh	**振荡器有效低状态位** 该位指示 OSC 的频率输出是否可用，这可以用振荡器看门狗检查。 0_B OSC 频率可用 1_B OSC 频率不可用。频率过低。
OSC2H	1	rh	**振荡器有效高状态位** 该位指示 OSC 的频率输出是否可用，这可以用振荡器看门狗检查。 0_B OSC 频率可用 1_B OSC 频率不可用。频率过高。

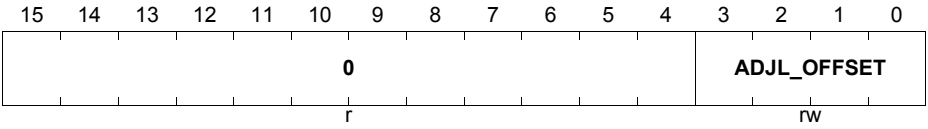
域	位	类型	描述
OWDRES	16	rwh	**振荡器看门狗复位** 将该位置 1 会重新启动振荡器检测。在 OWD 被复位后，该位被自动复位到 0，该过程因同步而需 2 个待机时钟周期完成。 0_B 振荡器看门狗不被清除并保持活动 1_B 振荡器看门狗被清除并重新启动。 OSC2L 和 OSC2H 标志将保持最终值，直到在 3 个待机时钟周期后被更新。
OWDEN	17	rw	**振荡器看门狗使能** 0_B 禁止振荡器看门狗 1_B 使能振荡器看门狗 *注： 在 OWD 被禁止时，OSC2H 和 OSC2L 会被清 0。*
0	[15:2], [31:18]	r	**保留** 读出值为 0。

13.9.4 CCU 寄存器 (ANACTRL)

ANAOFFSET

DCO1 偏移寄存器。

ANAOFFSET
DCO1 偏移寄存器 **(106C$_H$)** **复位值 : 0004$_H$**



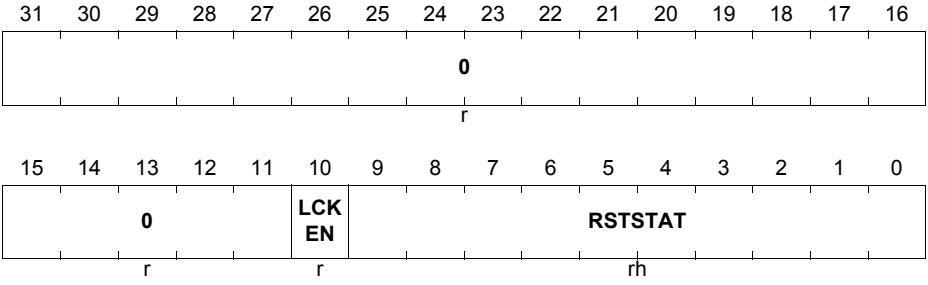
域	位	类型	描述
ADJL_OFFSET	3:0	rw	**ADJL 偏移寄存器** 可根据下述机制调节振荡器的频率。频率的步进响应取决于 DCO_ADJR。 该寄存器中的每一位都受存储器组织一章描述的位保护机制的保护。 0H - 4, 典型值 -1.2MHz (DCO_ADJR=0) 或 -1.8MHz (DCO_ADJR=1) 1H - 3, 典型值 -0.9MHz (DCO_ADJR=0) 或 -1.35MHz (DCO_ADJR=1) 4H 0, 默认 5H + 1, 典型值 +0.3MHz (DCO_ADJR=0) 或 +0.45MHz (DCO_ADJR=1) 8H + 4, 典型值 +1.2MHz (DCO_ADJR=0) 或 +1.8MHz (DCO_ADJR=1)
0	15:4	r	**保留** 读出值为 0；应写入 0。

13.9.5 RCU 寄存器 (SCU)

RSTSTAT

复位状态寄存器。系统启动后需要检查这个寄存器，以确定最后一次复位的原因。用户应在读该寄存器后应该将其清除，以确保在下一次复位发生时该寄存器是清除状态。该寄存器由上电复位来复位。

RSTSTAT
RCU 复位状态 **(0400_H)** **复位值：0000 0XXX$_H$**



域	位	类型	描述
RSTSTAT	[9:0]	rh	**复位状态信息** 提供最后一次复位的原因 0000000001_B 上电复位或欠压复位 $XXXXXXXX1X_B$ 通过位 RSTCON.MRSTEN 的主复位 $XXXXXXX1XX_B$ CPU 系统复位请求 $XXXXXX1XXX_B$ CPU 锁定复位 $XXXXX1XXXX_B$ Flash ECC 复位 $XXXX1XXXXX_B$ WDT 复位 $XXX1XXXXXX_B$ 时钟丢失复位 $XX1XXXXXXX_B$ 奇偶校验错误复位
LCKEN	10	r	**使能锁定状态** 0_B 通过锁定禁止复位 1_B 通过锁定使能复位
0	[31:11]	r	**保留**

RSTSET

系统中复位行为的选择性配置。写 1 将所选位置位，写 0 无效。

RSTSET
RCU 复位置位寄存器 **(0404_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					LCKEN	0									
r					w	r									

域	位	类型	描述
LCKEN	10	w	**使能锁定复位** 0_B 无效 1_B 当锁定生效时使能复位
0	[9:0], [31:11]	r	**保留**

RSTCLR

系统中复位行为的选择性配置。写 1 将清除所选位，写 0 无影响。

RSTCLR
RCU 复位清除寄存器 **(0408_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					LCKEN	0									RSCLR
r					w	r									w

域	位	类型	描述
RSCLR	0	w	**清除复位状态** 0_B 无效 1_B 清除域 **RSTSTAT.RSTSTAT**
LCKEN	10	w	**使能锁定复位** 0_B 无效 1_B 当锁定生效时，禁止复位
0	[9:1], [31:11]	r	**保留**

RSTCON

使能由关键事件触发的复位，可被任何复位类型复位。

RSTCON
RCU 复位控制寄存器 **(040C$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															MRSTEN
r															w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												U0PERSTEN	SPERSTEN	LOCRSTEN	ECCRSTEN
r												rw	rw	rw	rw

域	位	类型	描述
ECCRSTEN	0	rw	**使能 ECC 错误复位** 0_B 发生 ECC 双位错误时不复位 1_B 发生 ECC 双位错误时复位
LOCRSTEN	1	rw	**使能时钟丢失复位** 0_B 发生时钟丢失时不复位 1_B 发生时钟丢失时复位
SPERSTEN	2	rw	**使能 16kB SRAM 奇偶校验错误复位** 0_B 发生 SRAM 奇偶校验错误时不复位 1_B 发生 SRAM 奇偶校验错误时复位
U0PERSTEN	3	rw	**使能 USIC0 SRAM 奇偶校验错误复位** 0_B 发生 USIC0 存储器奇偶校验错误时不复位 1_B 发生 USIC0 存储器奇偶校验错误时复位
MRSTEN	16	w	**使能主复位** 0_B 无效 1_B 触发主复位
0	[15:4], [31:17]	r	**保留**

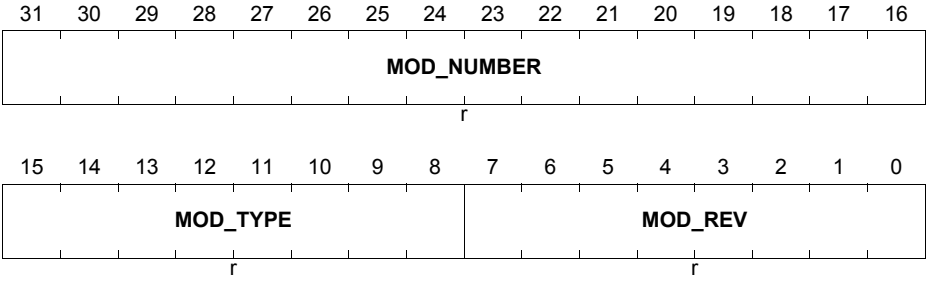
13.9.6 GCU 寄存器 (SCU)

ID

包含模块唯一 ID 的寄存器。

ID

SCU 模块 ID 寄存器 **(0008_H)** **复位值 : 00F1 C0XX$_H$**

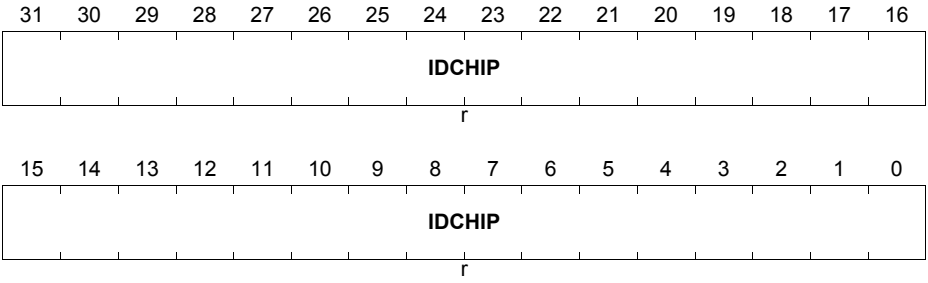


域	位	类型	描述
MOD_REV	[7:0]	r	**模块版本号** MOD_REV 定义了版本号。模块版本值从 01_H 开始 (第一个版本)。
MOD_TYPE	[15:8]	r	**模块类型** 该位域是 $C0_H$。它定义该模块是一个 32 位模块。
MOD_NUMBER	[31:16]	r	**模块编号值** 该位域定义模块标识号。

IDCHIP

包含 XMC 家族中芯片唯一 ID 的寄存器。该寄存器的值是 芯片标识号中描述的芯片标识号的一部分。

IDCHIP
芯片 ID 寄存器 **(0004_H)** **复位值 : $0000\ 0000_H$**

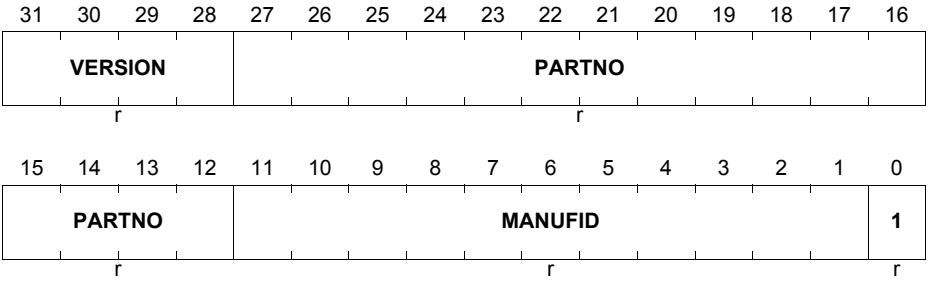


域	位	类型	描述
IDCHIP	[31:0]	r	**芯片 ID** $0001\ 3XXX_H$ XCM1300 $0001\ XXX2_H$ 温度：-40 - 85 °C $0001\ XXX3_H$ 温度：-40 - 105 °C $0001\ XX1X_H$ TSSOP38 引脚封装 $0001\ XX2X_H$ TSSOP28 引脚封装 $0001\ XX3X_H$ TSSOP16 引脚封装 其他值保留

DBGROMID

包含唯一厂商 ID、芯片的部件号和设计等级代码的寄存器。

DBGROMID
调试系统 ROM ID 寄存器 **(0000_H)** **复位值 : $101E\ D083_H$**



域	位	类型	描述
MANUFID	[11:1]	r	**厂商标识**
PARTNO	[27:12]	r	**部件号**
VERSION	[31:28]	r	**产品版本**
1	0	r	**保留** 读出值为 0；应写入 0。

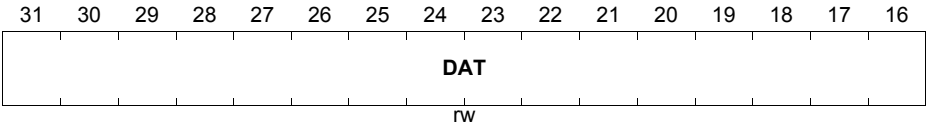
SSW0

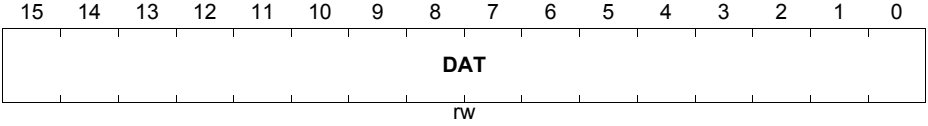
软件支持寄存器。

SSW0 用于改变 BMI 的值。

SSW0

SSW 寄存器 0 **(0014$_H$)** **复位值 : 0000 0000$_H$**





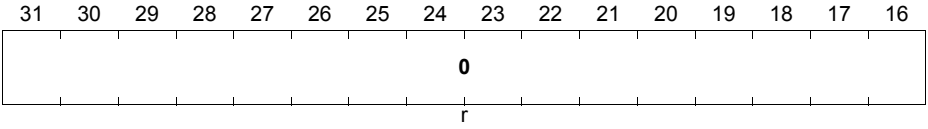
域	位	类型	描述
DAT	[31:0]	rw	**SSW 数据** *注： SSW 寄存器只能由主复位来复位。*

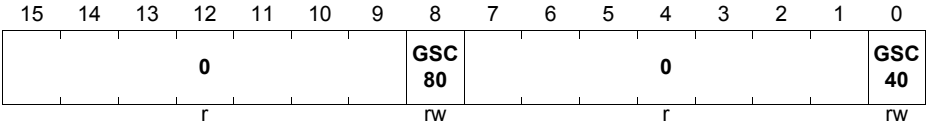
CCUCON

CAPCOM 模块控制寄存器。

CCUCON

CCU 控制寄存器 **(0030$_H$)** **复位值 : 0000 0000$_H$**





域	位	类型	描述
GSC40	0	rw	**全局启动控制 CCU40** 0_B 禁止 1_B 使能
GSC80	8	rw	**全局启动控制 CCU80** 0_B 禁止 1_B 使能
0	[7:1], [31:9]	r	**保留** 读出值为 0； 应写入 0。

SRRAW

未加屏蔽的服务请求状态。向 SRCLR 寄存器的某位写 1 将清除对应位，向 SRSET 的某位写 1 将对应位置位。写 0 无影响。

SRRAW

SCU 原始服务请求状态 **(0038_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSE_LOW	TSE_HIGH	TSE_DONE	RTC_TIM1	RTC_TIM0	RTC_ATIM1	RTC_ATIM0	RTC_CTR	0	SBYCLKFI	VCLIPI	FLCMPLTI	FLECC2I	PEU0I	PESRAMI	LOCI
rh	rh	rh	rh	rh	rh	rh	rh	r	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ORC7I	ORC6I	ORC5I	ORC4I	ORC3I	ORC2I	ORC1I	ORC0I	VDROPI	ACMP2I	ACMP1I	ACMP0I	VDDPI	AI	PI	PRWARN
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

域	位	类型	描述
PRWARN	0	rh	**屏蔽前的 WDT 预警事件状态** 0_B 事件尚未发生 1_B 事件已发生
PI	1	rh	**屏蔽前的 RTC 原始周期事件状态** 0_B 事件尚未发生 1_B 事件已发生
AI	2	rh	**屏蔽前的 RTC 原始报警事件状态** 0_B 事件尚未发生 1_B 事件已发生

域	位	类型	描述
VDDPI	3	rh	**屏蔽前的 VDDP 预警事件状态** 0_B 事件尚未发生 1_B 事件已发生
ACMP0I	4	rh	**屏蔽前的模拟比较器 0 事件状态** 0_B 事件尚未发生 1_B 事件已发生
ACMP1I	5	rh	**屏蔽前的模拟比较器 1 事件状态** 0_B 事件尚未发生 1_B 事件已发生
ACMP2I	6	rh	**屏蔽前的模拟比较器 2 事件状态** 0_B 事件尚未发生 1_B 事件已发生
VDROPI	7	rh	**屏蔽前的 VDROP 事件状态** 0_B 事件尚未发生 1_B 事件已发生
ORCxI (x=0-7)	x+8	rh	**屏蔽前的超量程比较器 X 事件状态** 0_B 事件尚未发生 1_B 事件已发生
LOCI	16	rh	**屏蔽前的时钟丢失事件状态** 0_B 事件尚未发生 1_B 事件已发生
PESRAMI	17	rh	**屏蔽前的 16kB SRAM 奇偶检验错误事件状态** 0_B 事件尚未发生 1_B 事件已发生
PEU0I	18	rh	**屏蔽前的 USIC0 SRAM 奇偶检验错误事件状态** 0_B 事件尚未发生 1_B 事件已发生
FLECC2I	19	rh	**屏蔽前的 Flash 双位 ECC 事件状态** 0_B 事件尚未发生 1_B 事件已发生
FLCMPLTI	20	rh	**屏蔽前的 Flash 操作完成事件状态** 0_B 事件尚未发生 1_B 事件已发生
VCLIPI	21	rh	**屏蔽前的 VCLIP 事件状态** 0_B 事件尚未发生 1_B 事件已发生

域	位	类型	描述
SBYCLKFI	22	rh	**屏蔽前的待机时钟故障事件状态** 0_B 未发生待机时钟故障 1_B 发生了待机时钟故障
RTC_CTR	24	rh	**屏蔽前的 RTC CTR 镜像寄存器更新状态** 0_B 未更新 1_B 更新已完成
RTC_ATIM0	25	rh	**屏蔽前的 RTC ATIM0 镜像寄存器更新状态** 0_B 未更新 1_B 更新已完成
RTC_ATIM1	26	rh	**屏蔽前的 RTC ATIM1 镜像寄存器更新状态** 0_B 未更新 1_B 更新已完成
RTC_TIM0	27	rh	**屏蔽前的 RTC TIM0 镜像寄存器更新** 0_B 未更新 1_B 更新已完成
RTC_TIM1	28	rh	**屏蔽前的 RTC TIM1 镜像寄存器更新状态** 0_B 未更新 1_B 更新已完成
TSE_DONE	29	rh	**屏蔽前的 TSE 测量完成事件状态** 0_B 事件未发生 1_B 事件已发生
TSE_HIGH	30	rh	**屏蔽前的 TSE 比较高温事件状态** 0_B 事件未发生 1_B 事件已发生
TSE_LOW	31	rh	**屏蔽前的 TSE 比较低温事件状态** 0_B 事件未发生 1_B 事件已发生
0	23	r	**保留**

SRMSK

服务请求屏蔽用于屏蔽 RAW 寄存器的输出。当某位被置 1 时，对应的事件发生时会触发中断或服务请求。

SRMSK

SCU 服务请求屏蔽 **(003C$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSE_LOW	TSE_HIGH	TSE_DONE	RTC_TIM1	RTC_TIM0	RTC_ATIM1	RTC_ATIM0	RTC_CTR	0	SBYCLKFI	VCLIPI	0	FLECC2I	PEU0I	PESRAMI	LOCI
rw	rw	rw	rw	rw	rw	rw	rw	r	rw	rw	r	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ORC7I	ORC6I	ORC5I	ORC4I	ORC3I	ORC2I	ORC1I	ORC0I	VDROPI	ACMP2I	ACMP1I	ACMP0I	VDDPI	0		PRWARN
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	r		rw

域	位	类型	描述
PRWARN	0	rw	**WDT 预警中断屏蔽** 0_B 禁止中断 1_B 使能中断
VDDPI	3	rw	**VDDP 预警中断屏蔽** 0_B 禁止中断 1_B 使能中断
ACMP0I	4	rw	**模拟比较器 0 中断屏蔽** 0_B 禁止中断 1_B 使能中断
ACMP1I	5	rw	**模拟比较器 1 中断屏蔽** 0_B 禁止中断 1_B 使能中断
ACMP2I	6	rw	**模拟比较器 2 中断屏蔽** 0_B 禁止中断 1_B 使能中断
VDROPI	7	rw	**VDROP 中断屏蔽** 0_B 禁止中断 1_B 使能中断

域	位	类型	描述
ORCxI (x=0-7)	x+8	rw	**超量程比较器 X 中断屏蔽** 0_B 禁止中断 1_B 使能中断
LOCI	16	rw	**时钟丢失中断屏蔽** 0_B 禁止中断 1_B 使能中断
PESRAMI	17	rw	**16kB SRAM 奇偶校验错误中断屏蔽** 0_B 禁止中断 1_B 使能中断
PEU0I	18	rw	**USIC0 SRAM 奇偶检验错误中断屏蔽** 0_B 禁止中断 1_B 使能中断
FLECC2I	19	rw	**Flash 双位 ECC 中断屏蔽** 0_B 禁止中断 1_B 使能中断
VCLIPI	21	rw	**VCLIP 中断屏蔽** 0_B 禁止中断 1_B 使能中断
SBYCLKFI	22	rw	**待机时钟故障中断屏蔽** 0_B 禁止中断 1_B 使能中断
RTC_CTR	24	rw	**RTC CTR 镜像寄存器更新屏蔽** 0_B 禁止中断 1_B 使能中断
RTC_ATIM0	25	rw	**RTC ATIM0 镜像寄存器更新屏蔽** 0_B 禁止中断 1_B 使能中断
RTC_ATIM1	26	rw	**RTC ATIM1 镜像寄存器更新屏蔽** 0_B 禁止中断 1_B 使能中断
RTC_TIM0	27	rw	**RTC TIM0 镜像寄存器更新屏蔽** 0_B 禁止中断 1_B 使能中断
RTC_TIM1	28	rw	**RTC TIM1 镜像寄存器更新屏蔽** 0_B 禁止中断 1_B 使能中断

域	位	类型	描述
TSE_DONE	29	rw	**TSE 测量完成中断屏蔽** 0_B 禁止中断 1_B 使能中断
TSE_HIGH	30	rw	**TSE 比较高温中断屏蔽** 0_B 禁止中断 1_B 使能中断
TSE_LOW	31	rw	**TSE 比较低温中断屏蔽** 0_B 禁止中断 1_B 使能中断
0	[2:1], 20, 23	r	**保留**

SRCLR

寄存器 SRRAW 的服务请求清除位。写 1 将清除对应位，写 0 无效。

SRCLR
SCU 服务请求清除 **(0040_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSE_LOW	TSE_HIGH	TSE_DONE	RTC_TIM1	RTC_TIM0	RTC_ATIM1	RTC_ATIM0	RTC_CTR	0	SBYCLKFI	VCLIPI	FLCMPLTI	FLECC2I	PEU0I	PESRAMI	LOCI
w	w	w	w	w	w	w	w	r	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ORC7I	ORC6I	ORC5I	ORC4I	ORC3I	ORC2I	ORC1I	ORC0I	VDROPI	ACMP2I	ACMP1I	ACMP0I	VDDPI	AI	PI	PRWARN
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

域	位	类型	描述
PRWARN	0	w	**WDT 预警中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
PI	1	w	**RTC 周期中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位

域	位	类型	描述
AI	2	w	**RTC 报警中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
VDDPI	3	w	**VDDP 预警中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
ACMP0I	4	w	**模拟比较器 0 中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
ACMP1I	5	w	**模拟比较器 1 中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
ACMP2I	6	w	**模拟比较器 2 中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
VDROPI	7	w	**VDROP 中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
ORCxI (x=0-7)	x+8	w	**超量程比较器 X 中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
LOCI	16	w	**时钟丢失中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
PESRAMI	17	w	**16kB SRAM 奇偶校验错误中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
PEU0I	18	w	**USIC0 SRAM 奇偶校验错误中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
FLECC2I	19	w	**Flash 双位 ECC 中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
FLCMPLTI	20	w	**Flash 操作完成中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位

域	位	类型	描述
VCLIPI	21	w	**VCLIP 中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
SBYCLKFI	22	w	**待机时钟故障中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
RTC_CTR	24	w	**RTC CTR 镜像寄存器更新清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
RTC_ATIM0	25	w	**RTC ATIM0 镜像寄存器更新清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
RTC_ATIM1	26	w	**RTC ATIM1 镜像寄存器更新清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
RTC_TIM0	27	w	**RTC TIM0 镜像寄存器更新清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
RTC_TIM1	28	w	**RTC TIM1 镜像寄存器更新清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
TSE_DONE	29	w	**TSE 测量完成中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
TSE_HIGH	30	w	**TSE 比较高温度中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
TSE_LOW	31	w	**TSE 比较低温度中断清除** 0_B 无效 1_B 清除原始状态寄存器中的状态位
0	23	r	**保留**

SRSET

寄存器 SRRAW 中的服务请求置 1 位。写 1 将对应位置 1，写 0 无效。

SRSET

SCU 服务请求置位 **(0044$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TSE_LOW	TSE_HIGH	TSE_DONE	RTC_TIM1	RTC_TIM0	RTC_ATIM1	RTC_ATIM0	RTC_CTR	0	SBYCLKFI	VCLIPI	FLCMPLTI	FLECC2I	PEU0I	PESRAMI	LOCI
w	w	w	w	w	w	w	w	r	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ORC7I	ORC6I	ORC5I	ORC4I	ORC3I	ORC2I	ORC1I	ORC0I	VDROPI	ACMP2I	ACMP1I	ACMP0I	VDDPI	AI	PI	PRWARN
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

域	位	类型	描述
PRWARN	0	w	**WDT 预警中断置位** 0_B 无效 1_B 将原始状态寄存器中的状态位置 1
PI	1	w	**RTC 周期中断置位** 0_B 无效 1_B 将原始状态寄存器中的状态位置 1
AI	2	w	**RTC 报警中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
VDDPI	3	w	**VDDP 预警中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
ACMP0I	4	w	**模拟比较器 0 中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
ACMP1I	5	w	**模拟比较器 1 中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
ACMP2I	6	w	**模拟比较器 2 中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1

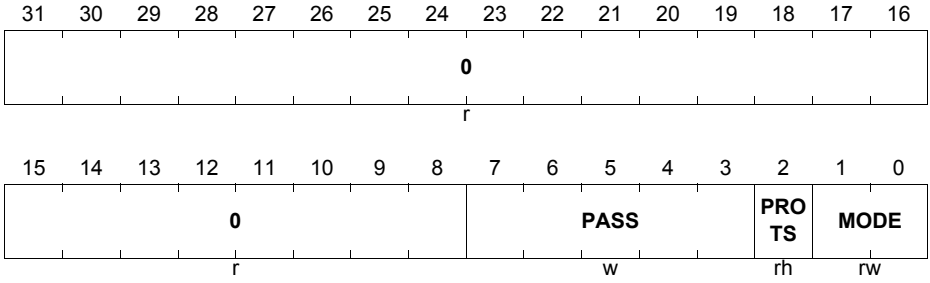
域	位	类型	描述
VDROPI	7	w	**VDROP 中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
ORCxI (x=0-7)	x+8	w	**超量程比较器 X 中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
LOCI	16	w	**时钟丢失中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
PESRAMI	17	w	**16kB SRAM 奇偶校验错误中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
PEU0I	18	w	**USIC0 SRAM 奇偶校验错误中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
FLECC2I	19	w	**Flash 双位 ECC 中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
FLCMPLTI	20	w	**Flash 操作完成中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
VCLIPI	21	w	**VCLIP 中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
SBYCLKFI	22	w	**待机时钟故障中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
RTC_CTR	24	w	**RTC CTR 镜像寄存器更新置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
RTC_ATIM0	25	w	**RTC ATIM0 镜像寄存器更新置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
RTC_ATIM1	26	w	**RTC ATIM1 镜像寄存器更新置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1

域	位	类型	描述
RTC_TIM0	27	w	**RTC TIM0 镜像寄存器更新置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
RTC_TIM1	28	w	**RTC TIM1 镜像寄存器更新置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
TSE_DONE	29	w	**TSE 测量完成中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
TSE_HIGH	30	w	**TSE 比较高温中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
TSE_LOW	31	w	**TSE 比较低温中断置位** 0_B 无效 1_B 将原始状态寄存器里设置中的状态位置 1
0	23	r	**保留**

PASSWD

PASSWD 寄存器用于控制位保护机制。

PASSWD
口令寄存器 **(0024_H)** **复位值 : 0000 0007_H**



域	位	类型	描述
MODE	[1:0]	rw	**位保护机制控制位** 00_B 禁止位保护机制 - 允许直接访问受保护位。 11_B 使能位保护机制 - 必须向位域PASS写入口令，才能打开或关闭对保护位的访问。 (默认) 其他：使能位保护机制，类似于设置 MODE = 11_B。 这两位不能直接写入。要在值 11_B 和 00_B 之间改变其值，必须向位域 PASS 写 11000_B。唯有如此，MODE 位域才能被写入。
PROTS	2	rh	**位保护信号状态位** 该位指示保护状态。 0_B 软件能写所有受保护的位。 1_B 软件不能写任何受保护的位。
PASS	[7:3]	w	**口令位** 该位保护机制只识别以下三个口令： 11000_B使能对位域 MODE 的写操作。 10011_B开放对所有受保护位的写访问。 10101_B关闭对所有受保护位的写访问。
0	[31:8]	r	**保留**

MIRRSTS

用于控制 SCU 和 RTC 之间通信的镜像状态寄存器。

MIRRSTS

镜像更新状态寄存器 **(0048_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											RTC_TIM1	RTC_TIM0	RTC_ATIM1	RTC_ATIM0	RTC_CTR
r											rh	rh	rh	rh	rh

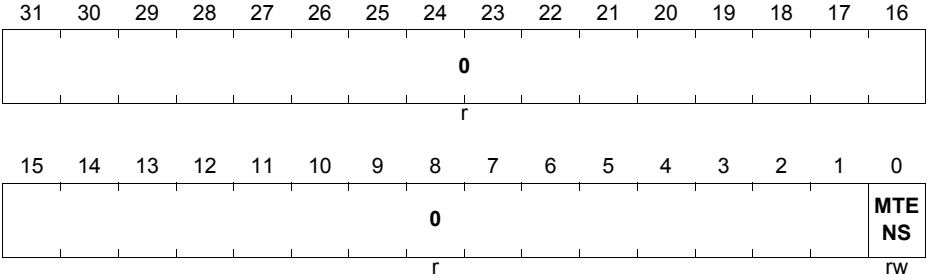
位域	位	类型	描述
RTC_CTR	0	rh	**RTC CTR 镜像寄存器更新状态** 0_B 不在更新 1_B 正在更新
RTC_ATIM0	1	rh	**RTC ATIM0 镜像寄存器更新状态** 0_B 不在更新 1_B 正在更新
RTC_ATIM1	2	rh	**RTC ATIM1 镜像寄存器更新状态** 0_B 不在更新 1_B 正在更新
RTC_TIM0	3	rh	**RTC TIM0 镜像寄存器更新状态** 0_B 不在更新 1_B 正在更新
RTC_TIM1	4	rh	**RTC TIM1 镜像寄存器更新状态** 0_B 不在更新 1_B 正在更新
0	[31:14]	r	**保留** 读出值为 0； 应写入 0。

PMTSR

该寄存器从一个存储器实例中选择奇偶校验测试输出。

PMTSR

奇偶校验存储器测试选择寄存器　　(0054$_H$)　　复位值 : 0000 0000$_H$



域	位	类型	描述
MTENS	0	rw	**16kB SRAM 的奇偶校验测试使能控制** 控制 16kB SRAM 的测试多路选择器 0_B　标准操作 1_B　在写操作期间产生一个反转的奇偶校验位
0	31:1]	r	**保留** 应写入 0。

14 伪随机数发生器

本章描述伪随机数发生器 (PRNG) 模块，包含以下部分：

14.1 简介

伪随机位发生器 (PRNG) 能快速产生随机数据。

14.2 工作模式描述

14.2.1 密钥加载模式

在 PRNG 能够使用前，必须由用户软件对其初始化。

PRNG 的密钥 (种子) k 是一个长度为 n 的位串 $k = (k_{n-1}, k_{n-2}, ..., k_2, k_1, k_0)$ 。尽管允许更短或更长的密钥，但建议使用 80 位的密钥长度。推荐使用芯片独有的种子值。

PRNG 的初始化包含 2 个基本阶段：

1. 密钥加载
2. 预热

通过将位 **PRNG_CTRL**.KLD 置 "1" 来初始化密钥加载。在密钥加载模式，**PRNG_WORD** 总是表现为一个 16 位的目的寄存器。密钥 $k = (W_{p-1}, ..., W_1, W_0)$ 的 p 部分字 W_i $(0 <= i < p)$ 按照 $W_0, W_1, ..., W_{p-1}$ 的顺序被连续写入 **PRNG_WORD**，其中 $W_i = (k_{15}, ..., k_1, k_0)$。一种实用的种子长度是 80 位 (即 5 个数据字，p=5)。因为部分密钥字的数据位按顺序连续加载到 PRNG 的内部状态，所以加载一个密钥字需要 16 个时钟周期。当加载正在进行时，**PRNG_CHK**.RDV 的标志被置 "0"。该标志为 "1" 表示可向 **PRNG_WORD** 写下一个部分密钥值。

在加载了完整的密钥后，**PRNG_CTRL**.KLD 的标志必须被置 "0"，以准备进行下面的预热阶段。该操作需要一个时钟周期。

预热阶段使密钥位彻底扩散。为达此目的，用户必须读取和丢弃来自寄存器 **PRNG_WORD** 的 64 个随机位。随机数据输出块必须被设置为 b = 8 或 16 位。这可以通过将 **PRNG_CTRL**.RDBS 域设置为相应值来实现。

标志 **PRNG_CHK**.RDV 置 "1" 表示可以从 **PRNG_WORD** 中读取下一个宽度为 b 的随机数据块。

如果因为任何原因，**PRNG_CTRL**.RDBS 被复位到其默认值 00_B，PRNG 必须再次被初始化 – 即必须再加载一次密钥和执行一次预热阶段。

14.2.2 流模式

标志 **PRNG_CHK**.RDV 置 1_B 表示可以从 **PRNG_WORD** 中读取下一个随机数据块。在读取一个字后，标志 **PRNG_CHK**.RDV 被硬件复位到 0，开始产生新的随机位。 PRNG 需要 17–18 个时钟周期来产生 16 个随机位，需要 9–10 个时钟周期产生 8 个随机位。从软件的观点来看，没必要查询 **PRNG_CHK**.RDV 标志。只要 **PRNG_CHK**.RDV 为 "0"，对 **PRNG_WORD** 的连续读访问会被硬件自动延迟。

输出数据块的宽度可通过设置 **PRNG_CTRL**.RDBS 的值来改变。该操作应在进入流模式之前完成。否则，如果在流模式期间改变 **PRNG_CTRL**.RDBS 的值，则新设置要等到下一个随机数产生周期开始才生效。硬件检查所选择的位数是否可用，当该条件为真时，标志 **PRNG_CHK**.RDV 被置 1。

在从 8 位工作模式切换到 16 位工作模式时，为避免读到一个重复的随机字节，应在切换到 16 位模式之前将在 8 位模式下产生的最后一个字节丢弃。

注： PRNG_WORD 应被当作 16 位寄存器访问，高 16 位 (一次 32 位访问的) 在写入时被忽略，在读取时返回 0，因此不包含随机数据。

14.2.3 随机比特流的刷新和重新启动

可以用一个新密钥刷新随机比特序列。在这种情况下，包含在最后内部状态中的熵没有被清除，而是新的密钥被混合到当前的 PRNG 状态。这一过程被称为刷新。

刷新是将另外的熵引入到随机比特序列产生的一种方法。如果不进行刷新，整个熵保持在初始密钥 (或者种子) 状态，该初始密钥是在第一次密钥加载期间用于初始化 PRNG 的密钥。此后，随机比特序列的产生是纯粹确定性的。无论何时执行刷新，这个确定性过程都会被打破。刷新意味着在数据生成时使用相同的密钥重新产生相同的随机结果是不可能的。

由于 PRNG 的内部状态不能被直接读取和设置，所以不能从任意给定的状态恢复一个序列。然而，基于某个确定初始密钥的随机比特序列可以继续。这意味着输出序列的一段 **s** 被保存，该段稍后被用作初始密钥。长度为 **n** 的 这个段 $\mathbf{s} = (s_{n-1}, s_{n-2}, ..., s_2, s_1, s_0)$ 应该是新的 – 即在应用中使用的最后几位之后产生。**s** 的长度至少应为 n = 80 位。这样以来，一个伪随机比特序列可以在系统复位后继续，而不需要新的密钥素材。该过程被称为重新启动。

注： 在多应用 / 多任务环境下，结合需求获得一个可再生的伪随机位序列可能需要一个状态保存和恢复功能。因此 OS 必须能在将控制交给应用 2 之前保存 PRNG 的状态，在将控制返回到应用 1 之前恢复 PRNG 的状态。这对 PRNG 模块是不可能的。

14.3 调试行为

当系统被调试器停止时，PRNG 不支持挂起模式。这意味着 PRNG 在调试停止期间继续它的操作。

14.4 PRNG 寄存器

伪随机数发生器的接口包括寄存器 PRNG_WORD、 PRNG_CHK 和 PRNG_CTRL。

表 14-1 寄存器地址空间

模块	基地址	结束地址	备注
PRNG	$4802\ 0000_H$	$4802\ 000F_H$	

表 14-2 寄存器一览表

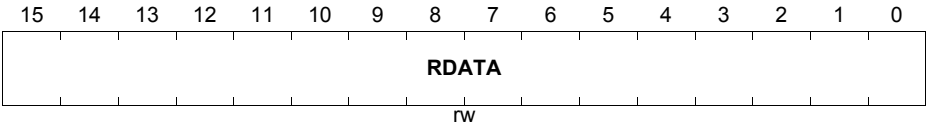
寄存器简称	寄存器全称	偏移地址	页码
PRNG 寄存器，数据 SFR			
PRNG_WORD	伪随机数发生器字寄存器	00_H	**页 14-4**
PRNG_CHK	伪随机数发生器状态检查寄存器	04_H	**页 14-5**
PRNG 寄存器，控制 SFR			
PRNG_CTRL	伪随机数发生器控制寄存器	$0C_H$	**页 14-6**

寄存器是按字寻址的。

14.4.1 数据 SFR

伪随机数发生器字寄存器

PRNG_WORD
伪随机数发生器字寄存器 **(00_H)** **复位值 : 0000_H**



域	位	类型	描述
RDATA	15:0	rw	**随机数** 随机位块或要加载的密钥。 在流模式，有效随机位的范围由 PRNG_CTRL.RDSB 给定的设置和标志 PRNG_CHK.RDV 的值定义。 在密钥加载模式，种子值 (密钥) 通过该寄存器写入。在这种情况下，密钥以 16 位为一组写入。

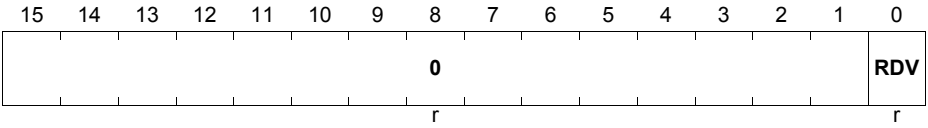
附加信息

Notes

1. *当 PRNG 运行在密钥加载模式时 (通过置位 PRNG_CTRL.KLD 来配置)，读 PRNG_WORD 会返回最后写入到该寄存器的值，而当 PRNG 运行在流模式时 (PRNG_CTRL.KLD = '0')，对该寄存器的写访问会被忽略。*
2. *对 SFR PRNG_WORD 的写访问：*
   强烈建议在 SFR 位 PRNG_CHK.RDV 指示寄存器 PRNG_WORD 准备好接收 (新) 数据 (将用作 PRNG 的种子。) 前一直等待。
3. *对 SFR PRNG_WORD 的读访问：*
   强烈建议在读 SFR PRNG_WORD 之前检查 SFR 位 PRNG_CHK.RDV 。

伪随机数发生器状态检查寄存器

PRNG_CHK
伪随机数发生器状态检查寄存器 (04_H) 复位值 : 0000_H



域	位	类型	描述
RDV	0	r	**随机数 / 密钥有效标志** 0_B **INV，** ，新的随机数据块尚未准备好被读取。当加载正在进行时，如在**页 14-1“ 密钥加载模式 ”** 中所述，该标志被置为 0_B。 1_B **VAL，** ，随机数据块有效。在密钥加载模式，该值表示可以向 **PRNG_WORD** 写下一个部分密钥字 。
0	15:1	r	**保留**

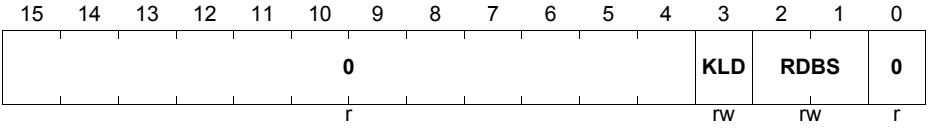
附加信息

*注： 如果从 **PRNG_WORD** 中读一个字或字节，但此时数据尚未准备好 (**PRNG_CHK**.RDV = 0_B)，则系统会暂停运行，直到数据准备好。*

14.4.2 控制 SFR

伪随机数发生器控制寄存器

PRNG_CTRL
伪随机数发生器控制寄存器 **($0C_H$)** **复位值 : 0000_H**



域	位	类型	描述
KLD	3	rw	**密钥加载操作模式** 0_B **STRM，** , 流模式 (默认) 1_B **KLD，** , 密钥加载模式
RDBS	2:1	rw	**随机数据块大小** 设置读访问的随机数据块大小 (密钥加载时总是使用 16 位) 00_B **RES，** , 复位状态 (未定义随机数据块的大小)[1)]，**PRNG_WORD** 的值不确定。 01_B **BYTE，** , 在 PRNG_WORD.RDATA[7:0] 中的 8 位 10_B **WORD，**, 在 PRNG_WORD.RDATA[15:0] 中的 16 位 11_B **RFU，** , 保留为将来使用，**PRNG_WORD** 的值不确定。
0	0	r	**保留**
0	15:4	r	**保留**

1) 一旦将 RDBS 设置为复位值 (00_B)，PRNG 在功能能被使用前必须完全初始化 。初始化需要先执行密钥加载，然后执行预热阶段。

附加信息

*注： 在密钥加载期间，建议不要改变 **PRNG_CTRL**.KLD 标志。否则，PRNG 位的分布会不均等。*

通信外设

15 通用串行接口通道 (USIC)

通用串行接口通道模块 (USIC) 是一个能处理多种串行通信协议的灵活的接口模块。一个 USIC 模块包含两个独立的通信通道，即 USICx_CH0 和 USICx_CH1，x 为通用 USIC 模块的编号 (例如， USIC 模块 0 的通道 0 用 USIC0_CH0 来表示)。用户可以在运行期间对模块编程，决定每个通信通道处理哪种协议和使用哪些引脚。

参考文献

更多的信息请参考下列文档：

[1] IIC Bus Specification (Philips Semiconductors v2.1)

[2] IIS Bus Specification (Philips Semiconductors June 5 1996 revision)

15.1 概述

本节是关于 USIC 特性的简介。

15.1.1 特性

每个 USIC 通道都可以被独立配置，以满足不同的应用需求。例如，运行过程中可以选择或更改协议，而不需要复位。 USIC 支持以下协议：

- **UART (ASC，异步串行通道)**
  - 模块能力：接收器 / 发送器的最大波特率为 f_{PB} / 4
  - 宽波特率范围，波特率可降至个位数
  - 每个数据帧的数据位数：1 到 63
  - MSB 在先或 LSB 在先
- **LIN** 硬件支持 (局部互联网络)
  - 基于 ASC 协议的数据传送
  - 可以通过内建的波特率发生器捕获事件进行波特率检测
  - 软件控制的校验和产生具有更高的灵活性
- **SSC/SPI** (带或不带从选通线的同步串行通道)
  - 支持标准 SPI 两位 SPI 和四位 SPI 格式
  - 模块能力：最大波特率为 f_{PB}/2，受环路延迟限制
  - 每个数据帧的数据位数可从 1 到 63，在有显性停止条件时可更多
  - 支持奇偶校验位生成
  - MSB 在先或 LSB 在先
- **IIC**(IC 间总线)
  - 应用的波特率在 100 kbit/s 至 400 kbit/s 之间
  - 支持 7 位和 10 位寻址
  - 完整的主机和从机能力
- **IIS** (娱乐音频总线)
  - 模块能力：最大波特率为 f_{PB} / 2

注： 在实际应用中，可以达到的实际波特率取决于器件的工作频率、数据表中描述的时序参数、PCB 上的信号延迟和对等器件的时序。

除了可以灵活选择通信协议之外，USIC 的结构设计旨在减轻系统负荷（CPU 负荷），这样就允许高效的数据处理。 USIC 的结构设计考虑了以下几个方面：

- **数据缓冲器能力**
  标准的缓冲器能力包括一个用于接收数据的双字缓冲器和一个用于发送数据的单字缓冲器。这就允许 CPU 有更长的反应时间 （例如中断延迟）。
- **额外的 FIFO 缓冲区能力**
  除了标准缓冲器能力之外，已接收的数据和要发送的数据可被缓存在一个 FIFO 缓冲区结构中。接收和发送 FIFO 缓冲区的大小是可以独立编程的。根据应用需要，可将总共 64 个数据字的缓冲区分配给一个 USIC 模块的接收和发送 FIFO 缓冲区。（该 USIC 模块的两个通道共享 64 个数据字的缓冲区）。
  除了 FIFO 缓冲区以外，还有一个旁路机制允许引入高优先级数据，而不需刷新 FIFO 缓冲区。
- **发送控制信息**
  对于每一个要发送的数据字，一个 5 位的发送控制信息被自动添加到一些发送参数中，如字长、帧长或 SPI 协议所使用的从选择控制。通过分析用户软件已经写完的待发送数据的地址，可自动生成发送控制信息（32个输入单元=2^5=5位发送控制信息）。该特性允许单独处理每个数据字。例如，存储在发送 FIFO 中的与这些数据字相关联的发送控制信息可自动修改从机选择输出，以选择不同的通信目标 （从器件），而无需 CPU 负荷。另外，它可以用来控制帧的长度。
- **灵活的帧长控制**
  在一个数据帧中，待发送的位数与数据字长度无关，并且可以用两种不同的方式来处理。第一种方式允许使用一个已知长度，可自动生成最多 63 位的帧；第二种方式支持更长的帧 （甚至是无限长）或动态控制长度的帧。
- **中断能力**
  对于每个 USIC 模块，根据应用的需要，每个 USIC 通道的事件都可被单独地发送到 6 个可用的服务请求输出 SR[5:0] 之一。此外，除了协议专用的事件，还支持特定的帧起始和结束指示。
- **灵活的接口连接**
  每个USIC通道为通信信号提供几种可能的输入和输出引脚连接选择。这就允许USIC信号到引脚的灵活分配，这种分配可以在不对器件复位的情况下改变。
- **输入信号调理**
  每个输入信号都经过一个可编程输入调理级的处理，该调理级具有可编程滤波和同步能力。
- **波特率发生器**
  每个 USIC 通道都包含其自己的波特率发生器。波特率的产生可以基于内部的模块时钟，也可以基于一个外部的频率输入。这种结构允许使用一个非内部产生的频率进行数据发送，例如用于对多个通信伙伴进行同步。
- **传送触发能力**
  在主模式下，数据传送可以由 USIC 模块外部产生的事件触发。例如：由一个输入引脚或定时器单元触发 （发送数据验证）。该功能允许时基相关的数据发送。

- **调试器支持**
  USIC 提供了读出接收数据的具体地址，不需要与 FIFO 缓冲区机制交互。该功能允许调试器访问接收数据而不破坏接收数据序列。

为了获得所需要的波特率，必须考虑模块能力和应用环境这两个准则。模块能力由模块的输入时钟频率定义，它是模块运行的基础。可达到的波特率一般受限于应用环境，尽管模块的能力可以高出很多（取决于模块时钟和代表一个数据位所需的时钟周期数）。在大多数情况下，驱动器延迟、信号传播时间或电磁干扰（EMI）等应用环境的原因限制了可达到的最大波特率。

注： *可达到的最大波特率还受限于所选择的附加功能（例如，数字滤波器、输入同步级、采样点调整、数据结构等等）。还需要注意额外的延迟，例如（内部或外部）传播延迟和驱动延迟（例如 ASC 模式下的冲突检测、IIC 等）。*

USIC 模块 / 通道结构的框图如图 15-1 所示。

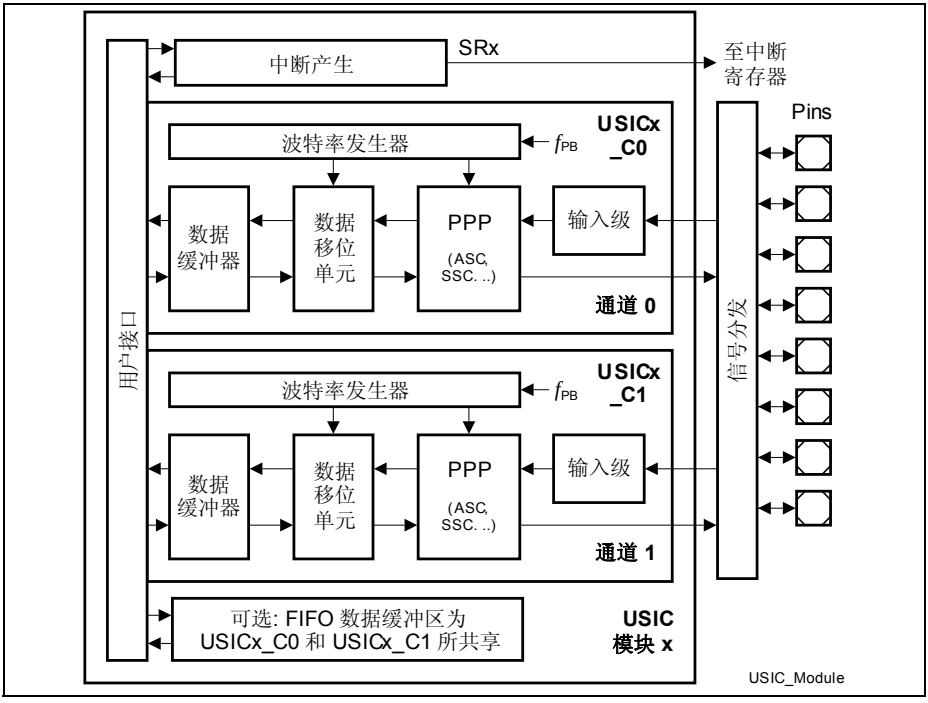


图 15-1　　USIC 模块 / 通道结构

15.2 操作 USIC

本节描述如何操作 USIC 通信通道。

15.2.1 USIC 结构简介

本节简要介绍 USIC 结构。

15.2.1.1 通道结构

USIC 模块包含两个独立的通信通道，结构如图 15-1 所示。

每个通道的数据移位单元和数据缓存机制都支持全双工数据传送。协议特定的行为由协议预处理器（PPP）处理。为了简化数据处理，每个 USIC 模块都有一个额外的 FIFO 数据缓冲区供选用，为每个通道存储发送和接收的数据。

由于通道控制与波特率产生是独立的，所以每个通信通道的通信协议、波特率和数据格式都可以独立编程。

15.2.1.2 输入级

对于每个协议，所使用的输入信号的数目取决于所选择的协议。每个输入信号由一个完成信号调理的输入级处理（称作 DXn，其中 n=0-5），如输入选择、极性控制或数字输入滤波。可以根据协议的内容将这些信号进行分类，见表 15-1。

标记为“可选”的输入对一个协议的标准功能来说是不需要的，但可用于增强功能。协议特定内容的描述在相关协议的章节中给出。对于外部频率输入请参见波特率发生器一节，而有关发送数据验证请参见数据处理一节。

表 15-1 不同协议的输入信号

选择的协议	移位数据输入 (由 DX0、DX3、DX4 和 DX5 处理)[1)]	移位时钟输入 (由 DX1 处理)	移位控制输入 (由 DX2 处理)
ASC, LIN	RXD	可选： 外部频率输入或 TXD 冲突检测	可选： 发送数据验证
标准 SSC, SPI (主机)	DIN0 (MRST, MISO)	可选： 外部频率输入或延迟补偿	可选： 发送数据验证或延迟补偿
标准 SSC, SPI (从机)	DIN0 (MTSR, MOSI)	SCLKIN	SELIN
2 位 SSC, SPI (主机)	DIN[1:0] (MRST[1:0], MISO[1:0])	可选： 外部频率输入或延迟补偿	可选： 发送数据验证或延迟补偿

表 15-1　不同协议的输入信号 (续表)

选择的协议	移位数据输入 (由 DX0、DX3、DX4 和 DX5 处理)[1]	移位时钟输入 (由 DX1 处理)	移位控制输入 (由 DX2 处理)
4 位 SSC, SPI (从机)	DIN[1:0] (MTSR[1:0], MOSI[1:0])	SCLKIN	SELIN
4 位 SSC, SPI (主机)	DIN[3:0] (MRST[3:0], MISO[3:0])	可选：外部频率输入或延迟补偿	可选：发送数据验证或延迟补偿
4 位 SSC, SPI (从机)	DIN[3:0] (MTSR[3:0], MOSI[3:0])	SCLKIN	SELIN
IIC	SDA	SCL	可选：发送数据验证
IIS (主机)	DIN0	可选：外部频率输入或延迟补偿	可选：发送数据验证或延迟补偿
IIS (从机)	DIN0	SCLKIN	WAIN

1) ASC、 IIC、 IIS 和标准 SSC 协议只使用 DX0 作为移位数据输入。

注：　为了灵活地将所需要的 USIC 输入功能分配到器件的端口引脚，每个输入级都可以在多种可能的输入位置中选择。
在互连一节列出了可用的 USIC 信号及其端口位置，见 ***页 15-193****。*

15.2.1.3　输出信号

对于每个协议，最多可以有 14 个协议相关的输出信号可用。实际使用的输出信号数量取决于所选择的协议。可以根据这些信号在协议中的含义对其进行分类，见 **表 15-2**。

标记为 “ 可选 ” 的输出对一个协议的标准功能来说是不需要的，但可用于增强功能。协议特定内容的描述在相关协议的章节中给出。为了使一个从器件与一个主器件同步，MCLKOUT 输出信号与移位时钟输出有一个稳定的频率关系（MCLKOUT 的频率可以高于 SCLKOUT）。如果一个特定的协议（例如在 SSC 从模式）不需要波特率发生器，则SCLKOUT和MCLKOUT信号可以作为具有50%占空比的时钟输出信号使用，其频率与通信波特率无关。

表 15-2　不同协议的输出信号

选择的协议	移位数据输出 DOUT[3:0]	移位时钟输出 SCLKOUT	移位控制输出 SELO[7:0]	主时钟输出 MCLKOUT
ASC, LIN	TXD	未使用	未使用	可选： 主时基
标准 SSC, SPI (主机)	DOUT0 (MTSR, MOSI)	主移位时钟	从选择， 片选	可选： 主时基
标准 SSC, SPI (从机)	DOUT0 (MRST, MISO)	可选： 独立时钟输出	未使用	可选： 独立时钟输出
2 位 SSC, SPI (主机)	DOUT[1:0] (MTSR[1:0], MOSI[1:0])	主移位时钟	从选择， 片选	可选： 主时基
2 位 SSC, SPI (从机)	DOUT[1:0] (MRST[1:0], MISO[1:0])	可选： 独立时钟输出	未使用	可选： 独立时钟输出
4 位 SSC, SPI (主机)	DOUT[3:0] (MTSR[3:0], MOSI[3:0])	主移位时钟	从选择， 片选	可选： 主时基
4 位 SSC, SPI (从机)	DOUT[3:0] (MRST[3:0], MISO[3:0])	可选： 独立时钟输出	未使用	可选： 独立时钟输出
IIC	SDA	SCL	未使用	可选： 主时基
IIS (主机)	DOUT0	主移位时钟	WA	可选： 主时基
IIS (从机)	DOUT0	可选： 独立时钟输出	未使用	可选： 独立时钟输出

注：　为了灵活地将所需要的 USIC 输入功能分配到器件的端口引脚，大多数输出信号都在多个端口引脚上可用。端口控制本身逐个引脚地定义用哪个输出信号作为一个端口引脚的输出信号（见端口一章）。
在互连一节列出了可用的 USIC 信号及其端口位置，见 *页 15-193**。*

15.2.1.4　波特率发生器

每个 USIC 通道包含一个波特率生成器，其结构如图 15-2 所示。它基于耦合的分频级，为不同的协议提供所需要的频率。波特率发生器包括：

- 一个产生输入频率 $f_{PIN} = f_{FD}$ 的分数分频器，这是为了基于内部系统频率 f_{PB} 产生波特率。
- 产生输入频率 $f_{PIN} = f_{DX1}$ 的 DX1 输入，这是为了基于一个外部信号产生波特率。
- 两个协议相关的计数器：分频器模式的计数器提供主时钟信号 MCLK、移位时钟信号 SCLK 和其他协议相关的信号；捕获模式定时器用于时间间隔测量，例如波特率检测。
- 一个与协议预处理相关联的时间量子计数器，它定义协议的具体时序，例如基于输入频率 f_{CTQIN} 的移位控制信号或位定时。
- 协议相关分频器的输出信号 MCLKOUT 和 SCLKOUT，在引脚上可用。为了适应不同的应用，这些信号的某些输出特性可以被配置。
  关于 USIC 信号在具体器件引脚上的可用性的详细信息，请参见互连一节。

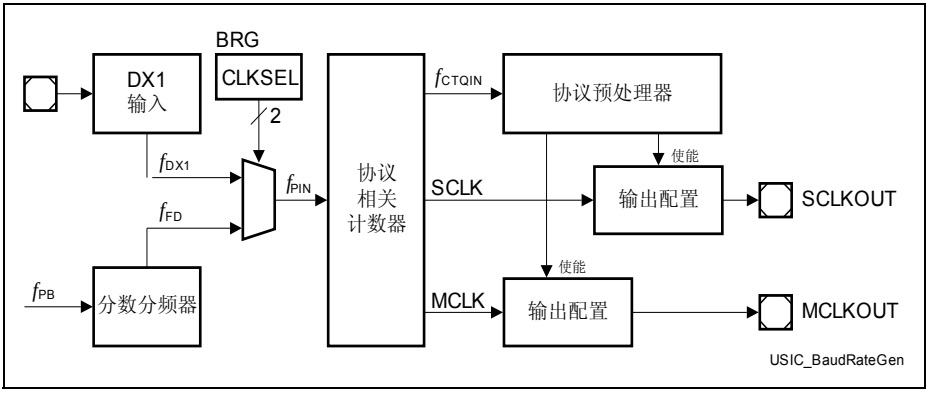


图 15-2　　波特率发生器

15.2.1.5　通道事件和中断

通知用户在数据传输和数据处理期间发生的事件是基于：

- 与发送或接收一个数据字相关的数据传送事件，与所选择的协议无关。
- 协议特定的事件，取决于所选择的协议。
- 与数据处理相关的数据缓冲事件，由可选的 FIFO 数据缓冲区进行数据处理。

15.2.1.6　数据移位和处理

USIC模块的数据处理基于一个独立的数据移位单元(DSU)和一个近似于所支持协议的缓冲器结构。数据移位和缓冲寄存器为 16 位宽（最大数据字长度），但可将多个数据字级联起来以实现更长的数据帧。DSU 的输入是移位数据（由输入级 DX0、DX3、DX4 和 DX5 处理）、移位时钟（由输入级 DX1 处理）和移位控制（由输入级 DX2 处理）。信号 DOUT[3:0] 代表移位数据输出。

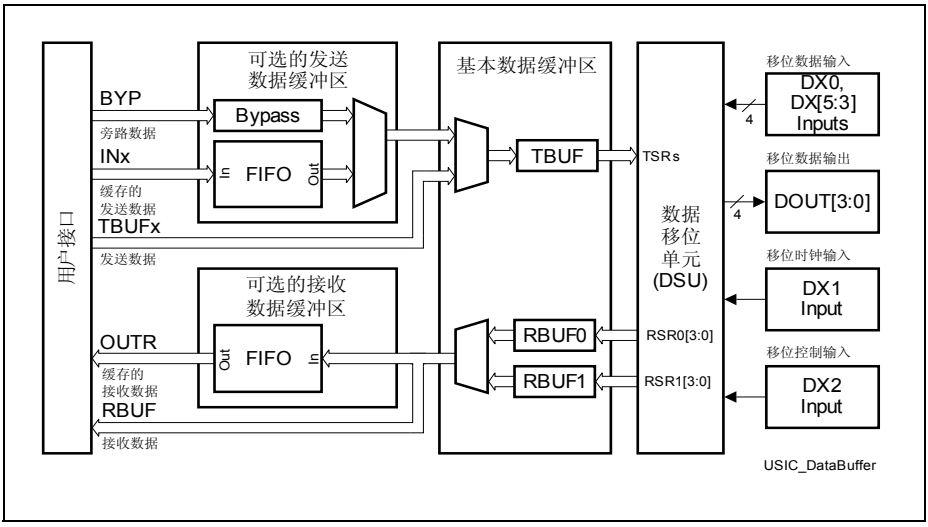


图 15-3　　数据缓冲原理

数据处理的原理包括：

- 一个发送器，包括 DSU 中的发送移位寄存器 (TSR 和 TSR[3:0]) 和一个发送数据缓冲器 (TBUF)。数据验证机制允许在一定条件下由外部事件触发和控制数据传送。
- 一个接收器，包括DSU中的两组交替工作的接收移位寄存器(RSR0[3:0]和RSR1[3:0])和一个双接收缓冲器结构 (RBUF0， RBUF1)。这两个交替接收移位寄存器支持接收长度大于一个数据字的数据流和数据帧。
- 一个用户接口，处理数据、中断、状态和控制信息。

基本数据缓冲器结构

接收数据的读访问和待发送数据的写访问可由一个基本的数据缓冲器结构处理。

可直接从接收缓冲器 RBUF0/RBUF1 中读取存储在这些寄存器中的接收数据。在这种情况下，用户必须谨慎处理接收序列，以便能以正确的顺序读取这些寄存器。为了简化接收缓冲器结构的使用，引入了寄存器 RBUF。对该寄存器的读操作将返回最先接收的数据字 （最老的数据），从而可遵守接收顺序。随着对 RBUF 的读访问 （至少是低字节）的完成，所读数据被自动声明为不再是新数据，下一个接收的数据字在 RBUF 中变为可见，并可在下一次读操作中被读取 。

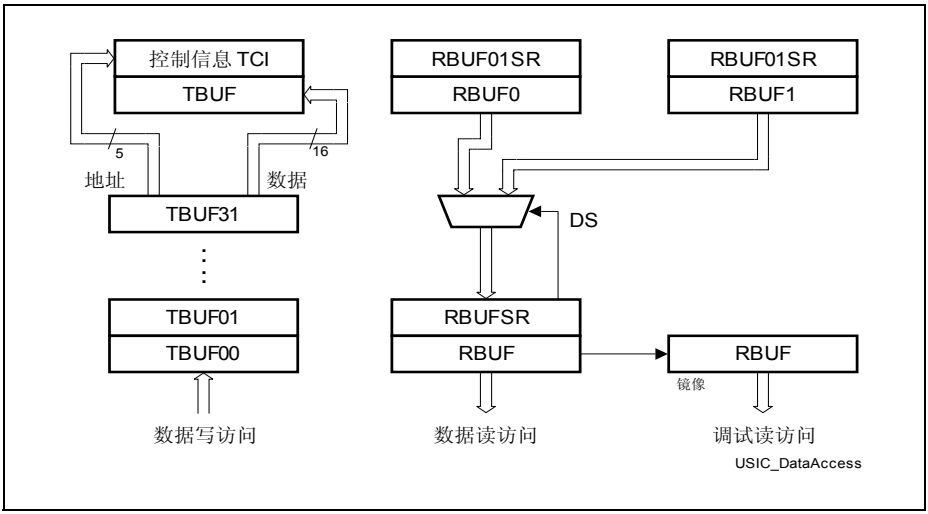


图 15-4　　不使用额外数据缓冲区的数据访问结构

建议通过访问 RBUF 来读取已接收的数据字，以避免直接处理 RBUF0 和 RBUF1。USIC 模块还支持对已接收数据字的调试访问 。调试器的读访问不应干扰接收数据序列，因而读访问的目标不应是 RBUF。为此，引入了寄存器 RBUFD。RBUFD 包含与 RBUF 相同的值，但对 RBUFD 的读访问不会改变数据的状态 （同一个数据可以被多次读取）。除了已接收的数据之外，每个接收数据字还有一些额外状态信息在接收缓冲器状态寄存器 RBUF01SR(与 RBUF0 和 RBUF1 中的数据相关) 和寄存器 RBUFSR (与 RBUF 中的数据相关) 可用。

可以通过用软件写发送缓冲器输入单元 TBUFx (x = 00-31) 的方式将发送数据加载到 TBUF 中，TBUFx 由 32 个连续地址组成。写入到这些输入单元之一的数据被存储在发送缓冲器 TBUF 中。此外，被写单元的地址在被评估后可用于其他控制用途。这个 5 位宽的信息 （称为发送控制信息 TCI) 可在不同协议中用于不同用途。

FIFO 缓冲区结构

为了使数据的建立和处理更加简便，USIC 模块还可选择支持一种额外的数据缓冲机制。数据缓冲区基于先入先出原则 (FIFO)，可确保传送的数据字序列的顺序无误。

如果使用 FIFO 缓冲区结构，则数据处理方案（带相关控制信息的数据）与没有 FIFO 的方案类似。额外的 FIFO 缓冲区的发送和接收功能可被独立地使能 / 禁止 (例如，如果数据 FIFO 缓冲区可以用于一个特定的 USIC 通道，则可以将其配置为发送数据路径不使用 FIFO 缓冲区，而接收数据路径使用 FIFO 缓冲区)。

不管 FIFO 深度如何，发送 FIFO 缓冲区都用 32 个连续的地址单元来寻址，这些单元标注为 INx 而非 TBUFx (x=00-31)。这 32 个地址用来存储与每个 FIFO 条目相关的 5 位 TCI (与被写入的数据一起)。

接收 FIFO 可以在两个独立的地址 OUTR 和 OUTDR 中读出，而不是 RBUF 和 RBUFD。对 OUTR 单元的读操作使下一次读操作要读取的下一个数据包变为可用 （一般的 FIFO 机制）。为了允许非侵入式调试（无数据丢失的风险），引入了第二个地址单元 (OUTDR)。对该地址的读操作可以返回与 OUTR 相同的值，但并未修改 FIFO 的内容。

发送 FIFO 还有旁路数据流和加载旁路数据到 TBUF 的能力。如果发送 FIFO 为空，，这可用于产生高优先级消息，或发送一个紧急消息。FIFO 缓冲区的发送控制还可以使用发送逻辑的传送触发和传送门控机制来进行数据验证 (例如由事件触发数据发送)。

注： 一个 USIC 通道的 FIFO 数据缓冲区的可用大小取决于具体器件。有关 FIFO 可用缓冲能力的详细信息，请参见具体实现章节。

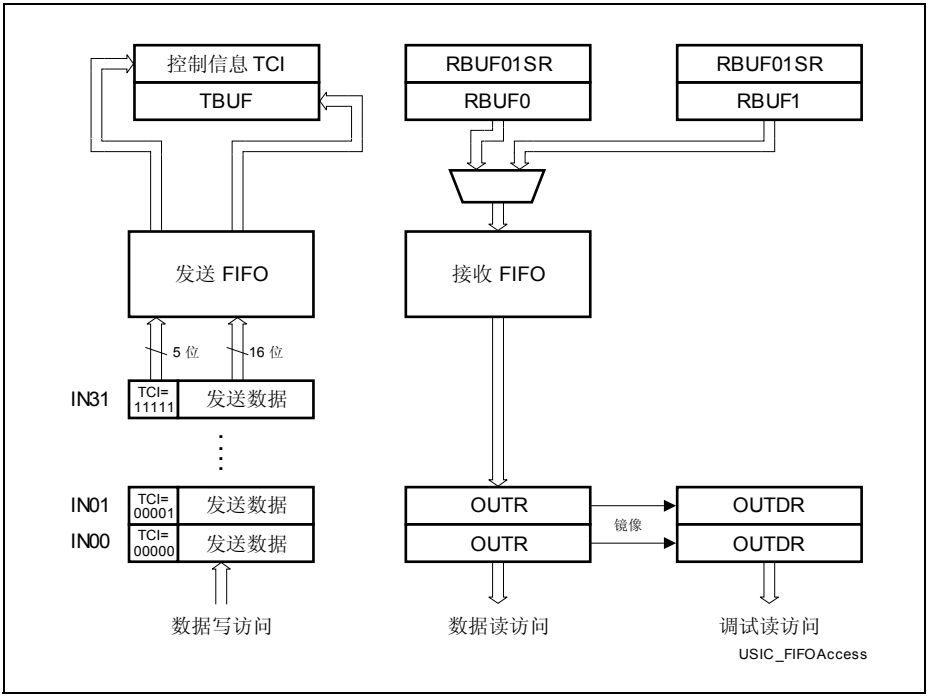


图 15-5 带有 FIFO 的数据访问结构

15.2.2 操作 USIC 通信通道

本节描述如何操作 USIC 的通信通道，包括协议控制和状态、模式控制和中断处理。必须考虑到以下几个方面：

- 使能 USIC 模块运行和配置不同器件工作模式的行为 (见 页 15-11)。
- 配置引脚 (参阅相应协议节中的描述)。
- 配置数据结构 (移位方向、字长、帧长、极性等)。

- 配置可选 FIFO 缓冲区区的数据缓冲区结构。仅在寄存器 CCFG 中的相关位被置 1 时 FIFO 缓冲区才被使能。
- 通过 CCR.MODE 选择一个协议。仅在寄存器 CCFG 中的相关位被置 1 时才能选择一个协议。

15.2.2.1 协议控制和状态

与协议相关的控制和状态信息位于协议控制寄存器 PCR 和协议状态寄存器 PSR 中。这些寄存器在可用的协议之间是共享的。因而在不同的协议里这些寄存器中的各位的含义是不同的。

PCR 位的使用

寄存器 PCR 中各位的含义由不同协议的协议相关别名来指示。

- ASC 协议的 PCR (见 页 15-55)
- SSC 协议的 PCR (见 页 15-83)
- IIC 协议的 PCR (见 页 15-111)
- IIS 协议的 PCR(见 页 15-128)

PSR 标志的使用

寄存器 PSR 中各标志的含义由不同协议的协议相关别名来指示。

- ASC 协议的 PSR 标志位 (见页 15-58)
- SSC 协议的 PSR 标志位 (见 页 15-86)
- IIC 协议的 PSR 标志位 (见 页 15-114)
- IIS 协议的 PSR 标志位 (见 页 15-130)

15.2.2.2 模式控制

对于系统控制任务，例如调试挂起请求，模式控制的概念允许对模块在不同的器件工作条件下的行为进行编程。对于每种器件工作模式 (正常操作、挂起模式)，通信通道的行为都可以被编程。因此，每一个通信通道都有一个相关联的内核状态配置寄存器 KSCFG，它定义了通信通道在以下工作模式下的行为：

- **正常操作：**
  当没有请求被挂起时，该模式是默认的工作模式。模块时钟未被关闭，且 USIC 寄存器是可读可写的。通道的行为由 KSCFG.NOMCFG 定义。
- **挂起模式：**
  当器件中有一个挂起请求时，该工作模式被请求。模块时钟未被关闭，且 USIC 寄存器是可读可写的。通道的行为由 KSCFG.SUMCFG 定义。

由寄存器 KSCFG 定义的四个内核模式如**表 15-3** 所示。

表 15-3 USIC 通信通道行为

内核模式	通道行为	KSCFG.NOMCFG
运行模式 0	通道的工作按照规定，对数据传送没有影响	00_B
运行模式 1		01_B
停止模式 0	在协议的章节中有明确的停止条件描述	10_B
停止模式 1		11_B

通常情况下，位域 KSCFG.NOMCFG 应配置为运行模式 0，这是标准操作的默认设置。如果一个通信通道不应对一个挂起请求作出响应（并继续在正常模式下运行），则位域 KSCFG.SUMCFG 必须被配置为与 KSCFG.NOMCFG 相同的值。如果通信通道应当在一个特定的停止条件发生时表现出不同的行为并停止运行，则必须向 KSCFG.SUMCFG 写入对应停止模式 0 或停止模式 1 的代码。

已经为选择的协议定义了停止条件 (见协议一节中的模式控制描述)。

注： *停止模式的选择在很大程度上取决于应用的需要，而且在同一应用中并行地需要不同的停止模式是不太可能的。其结果是，在寄存器 KSCFG 的位域中只应使用一个停止模式类型（0 或 1）。请勿混用停止模式 0 和停止模式 1，并且要避免在同一个通信通道中从停止模式 0 转换到停止模式 1（反之亦然）。*

注： *在 XMC1300 中，任何复位都可将位域 KSCFG.SUMCFG 复位到其默认值。如果在调试期间需要挂起功能，建议在用户的初始化代码中使能该功能。在对这个位域编程之前，应先使能模块时钟，并且在使能模块时钟时应特别谨慎，详见 SCU 一章中 CCU（时钟门控控制）一节的描述。*

15.2.2.3 通用通道事件和中断

通用事件和中断结构如图 15-6 所示。如果一个定义的条件得到满足，事件会被检测到，并且事件指示标志被自动置位。该标志将一直保持置位状态，直到被软件清除。如果中断被使能，那么检测到事件时可产生中断。事件指示标志的当前状态对中断的产生没有影响。因此，要继续产生中断并不需要清除该事件指示标志。

此外，在一个事件条件被一个中断节点指针选择时，USIC 通道的服务请求输出 SRx 被激活。这种结构允许把事件分配给中断，例如根据应用情况，多个事件可以共享相同的中断例程（多个事件激活同一个 SRx 输出），也可以单独处理每个事件（一个事件仅激活一个 SRx 输出）。

SRx 的输出被连接到中断控制寄存器，以处理 CPU 对服务请求的响应。这种分配在 页 15-133 的实现一节中描述。

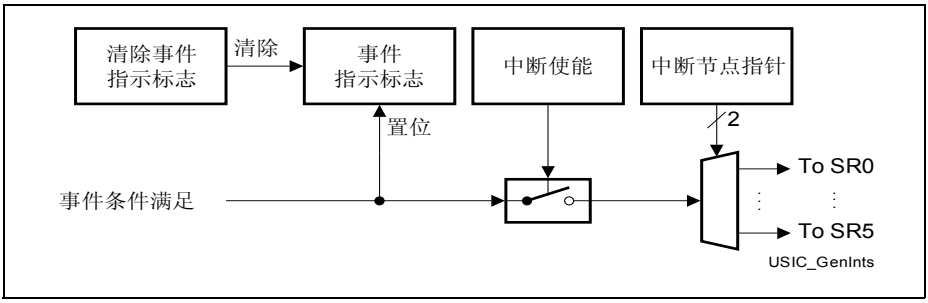


图 15-6　　通用事件和中断结构

15.2.2.4　数据传送事件和中断

数据传送事件基于一个数据字的发送或接收。相关的指示标志位于寄存器 PSR 中。所有事件都可以被单独使能以产生中断。

- 接收事件指示已收到一个数据字：
  如果在接收缓冲器 RBUF0 或 RBUF1 中一个新的接收字变为可用，则会发生一个接收事件或发生一个备用接收事件。
  如果位 RBUFSR.PERR = 0，发生接收事件。它由标志 PSR.RIF 指示，如果被使能，会引起接收中断。
- 接收器启动事件指示已经开始接收一个数据字：
  当检测到对一个新数据字的第一位进行移位的接收时钟边沿并且接收被使能时，发生接收器启动事件。它由标志 PSR.RSIF 指示，如果被使能，会引起发送缓冲器中断。在全双工模式，该事件在发送缓冲器事件之后半个移位时钟周期之际发生，并指示当前数据字的移位控制设置何时被内部“冻结”以及何时可以编程进行新的设置。
  在 SSC 和 IIS 模式，单次模式下的发送数据有效标志 TCSR.TDV 在发生接收启动事件时被清除。
- 备用接收事件指示已经收到一个特定的数据字：
  如果在接收缓冲器 RBUF0 或 RBUF1 中一个新的接收字变为可用，那么则会发生一个接收事件或发生一个备用接收事件。
  如果位 RBUFSR.PERR = 1，发生备用接收事件。它由标志 PSR.AIF 指示，如果被使能，会引起备用接收中断。
  根据所选择的协议不同，位 RBUFSR.PERR 有不同的含义。位 RBUFSR.PERR 置位在 ASC 模式指示发生了奇偶校验错误，在 IIC 模式指示收到一个新数据帧的第一个字符，在 IIS 模式指示关于左 / 右通道的 WA 信息。在 SSC 模式，该位用于指示所接收的字是否为第一个数据字，如果是第一个数据字则它被置位，否则被复位。
- 发送移位事件指示一个数据字已经被发送：
  发送移位事件发生在一个数据字的最后一个移位时钟边沿。该事件由标志 PSR.TSIF 指示，如果被使能，会引起发送移位中断。
- 发送缓冲器事件指示已经启动了一个数据字的发送：
  当发送缓冲器 TBUF 中的一个数据字被加载到移位寄存器并且可以向 TBUF 写入一个

新的数据字时，会发生发送缓冲器事件。在发送被使能的情况下，该事件在一个新数据字的第一位被移出的发送时钟边沿发生。该事件由标志 PSR.TBIF 指示，如果被使能，会引起发送缓冲器中断。
该事件还指示当前数据字发送的移位控制设置（字长、移位方向等）何时被内部“冻结”。
在 ASC 和 IIC 模式，单次模式下的发送数据有效标志位 TCSR.TDV 在发生发送缓冲器事件时被清除。

- 数据丢失事件指示最早接收的数据字丢失：
  如果在寄存器 RBUF （来自 RBUF0 或 RBUF1 的最老的数据）中的可用数据尚未被读出，却被新到来的数据覆盖，则发生该事件。该事件由标志 PSR.DLIF 指示，如果被使能，会引起协议中断。

表 15-4 表明了与数据发送事件和 USIC 通道控制的中断相关的寄存器、位和位域。

表 15-4　　数据传送事件和中断处理

事件	指示标志	标志清除	中断使能	SRx 输出选择
标准接收事件	PSR.RIF	PSCR.CRIF	CCR.RIEN	INPR.RINP
接收启动事件	PSR.RSIF	PSCR.CRSIF	CCR.RSIEN	INPR.TBINP
备用接收事件	PSR.AIF	PSCR.CAIF	CCR.AIEN	INPR.AINP
发送移位事件	PSR.TSIF	PSCR.CTSIF	CCR.TSIEN	INPR.TSINP
发送缓冲器事件	PSR.TBIF	PSCR.CTBIF	CCR.TBIEN	INPR.TBINP
数据丢失事件	PSR.DLIF	PSCR.CDLIF	CCR.DLIEN	INPR.PINP

图 15-7 展示了两个发送事件和中断。

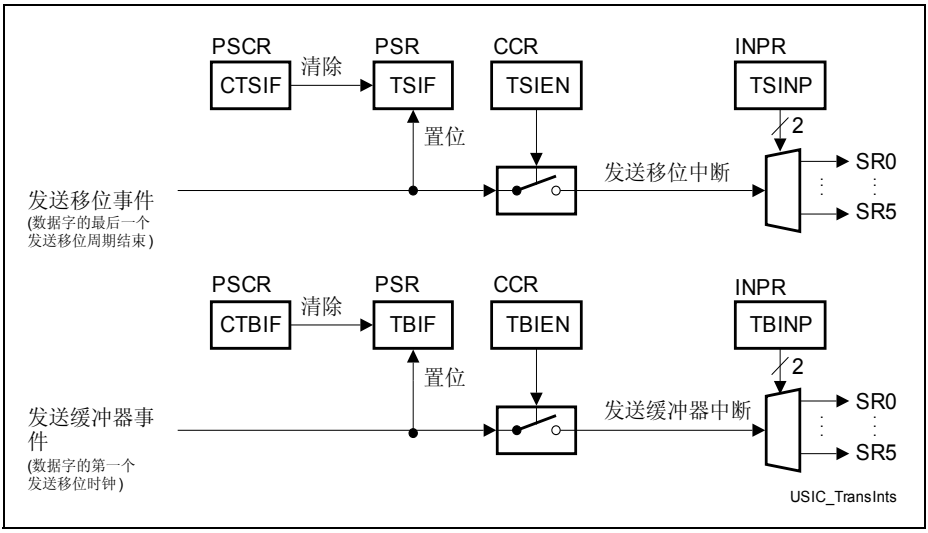


图 15-7　　发送事件和中断

图 15-8 展示了接收事件和中断。

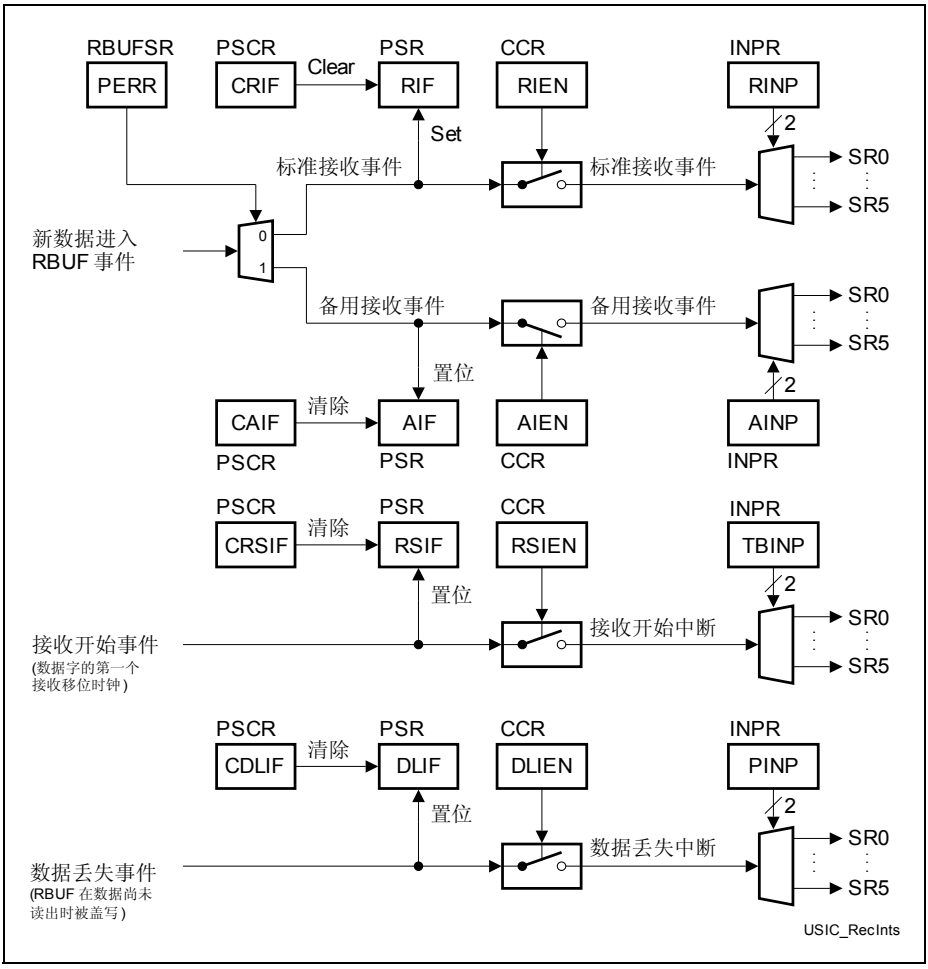


图 15-8　　接收事件和中断

15.2.2.5　波特率发生器事件和中断

波特率发生器事件由捕获模式的定时器在达到其最大值时产生。该事件由标志 PSR.BRGIF 指示，如果被使能，会引起协议中断。

表 15-5 列出了指示 USIC 通道的波特率发生器事件和控制 USIC 通道中断的寄存器、位和位域。

表 15-5　　波特率发生器事件和中断处理

事件	指示标志	标志清除	中断使能	SRx 输出选择
波特率发生器事件	PSR. BRGIF	PSCR. CBRGIF	CCR. BRGIEN	INPR.PINP

图 15-9 展示了波特率发生器事件和中断。

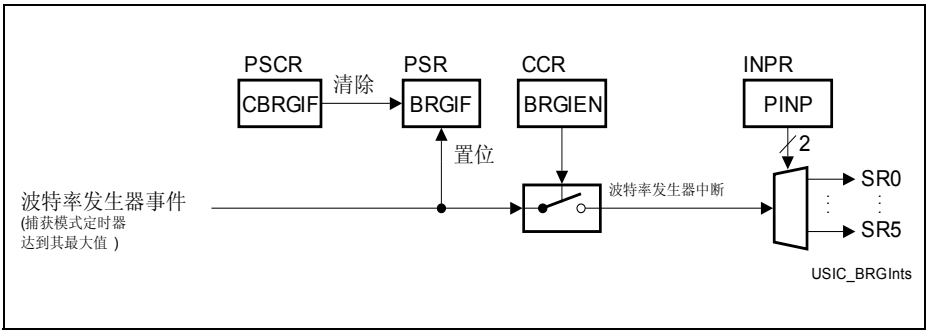


图 15-9　　波特率发生器事件和中断

15.2.2.6　协议特有的事件和中断

这些事件与协议特有的操作相关，这些协议特有的操作在相应协议的章节中描述。相关的指示标志位于寄存器PSR中。所有的事件都可以被单独使能，以产生共同的协议中断。

- ASC 模式下的协议特有事件：
  同步间断、发送线上的数据冲突、接收噪声、停止位的格式错误、接收器帧结束、发送器帧结束
- SSC 模式下的协议特有事件：
  MSLS 事件 (主模式下帧的起始和结束)、DX2T 事件 (在从模式下帧的起始和结束)、基于从选择信号和奇偶校验错误
- IIC 模式下的协议特有事件：
  错误的发送代码 (帧序列中的错误)、收到起始条件、收到重复起始条件、收到停止条件、收到非应答、仲裁失败、从器件读请求、其他一般错误
- IIS 模式下的协议特有事件：
  DX2T 事件 (WA 线的变化)、检测到 WA 下降沿或上升沿、WA 产生完成

表 15-6　协议特有事件和中断处理

事件	指示标志	标志清除	中断使能	SRx 输出选择
ASC 模式下的协议特有事件	PSR.ST[8:2]	PSCR.CST[8:2]	PCR.CTR[7:3]]	INPR.PINP
SSC 模式下的协议特有事件	PSR.ST[3:2]	PSCR.CST[3:2]	PCR.CTR[15:14]	INPR.PINP
IIC 模式下的协议特有事件	PSR.ST[8:1]	PSCR.CST[8:1]	PCR.CTR[24:18]	INPR.PINP
IIS 模式下的协议特有事件	PSR.ST[6:3]	PSCR.CST[6:3]	PCR.CTR[6:4], PCR.CTR[15]	INPR.PINP

15.2.3　输入级的操作

所有输入级都提供相同的功能集合。它们为所有协议使用，因为信号调理可以以非常灵活的方式适应不同的协议，并且数字滤波器可以分别打开和关闭。

15.2.3.1　通用输入结构

通常有两种类型的输入级，一种是数据的输入级 DX0 和 DX[5:3]，另一种是非数据输入级 DX[2:1]，如 图 15-10 和 图 15-11 所示。两种输入级的差别是：对于数据输入级，如果通过 CCR.HPCEN 位使能了硬件端口控制，则可以额外地从端口信号 HWINn 选择输入信号。每个输入级的所有其他使能/禁止功能和选择由寄存器DXnCR中的位独立控制。

所需要的输入信号可以在输入线 DXnA 到 DXnG 之间选择，也可以选择一个固定的 1 电平，这可通过对位域 DSEL 编程来实现（对于数据输入级，为使 DSEL 生效，硬件端口控制必须被禁止）。有关具体器件的输入信号分配，请参见互连一节 (15.12 节)。位 DPOL 允许将所选择输入信号的极性反转，以使输入信号的极性适应数据移位单元和协议状态机的内部极性。对于那些在数据传送时不使用任何额外信号调理的协议，输入信号可以直接被转发到数据移位单元。在这种情况下，数据通路没有任何因同步或滤波而带来的延迟。

在输入信号存在噪声的情况下，可以对输入信号进行同步（信号 DXnS 与 f_{PB} 同步）和在信号通路上额外使能一个数字噪声滤波器。通过设置 DSEN = 1 即可获得同步输入信号（如果 DFEN = 1 则信号被滤波）。请注意，这种同步会导致在信号通路有一个 2-3 倍 f_{PB} 周期的延时。

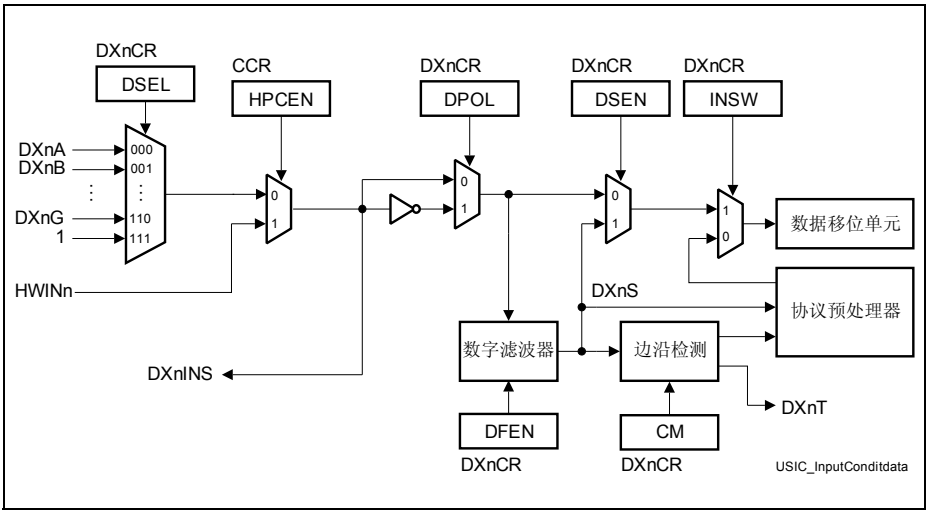


图 15-10　　DX0 和 DX[5:3] 的输入调理

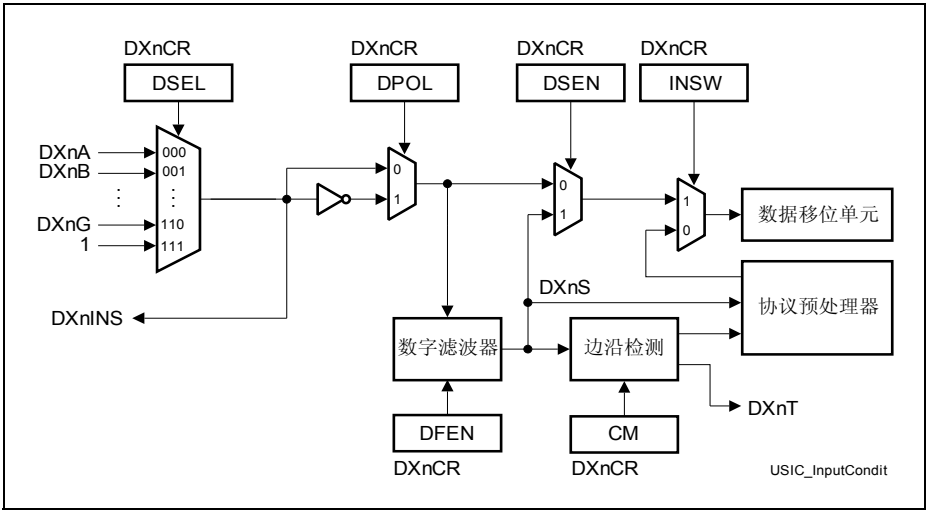


图 15-11　　DX[2:1] 的输入调理

如果输入信号由协议预处理器处理，那么通过设置 INSW = 0 可使数据移位单元被直接连接到协议预处理器。协议预处理器被连接到同步输入信号 DXnS，并且根据所选择的协议对边沿进行评估。

为了支持 SSC 和 IIS 协议中的延迟补偿，DX1 输入级还额外允许通过位 DCEN 使接收移位时钟的控制独立于发送移位时钟。当 DCEN = 0 时，移位时钟源由 INSW 选择，并且对接收和发送是相同的。当 DCEN = 1 时，接收移位时钟源自所选择的输入线，如 图 15-12 所示。

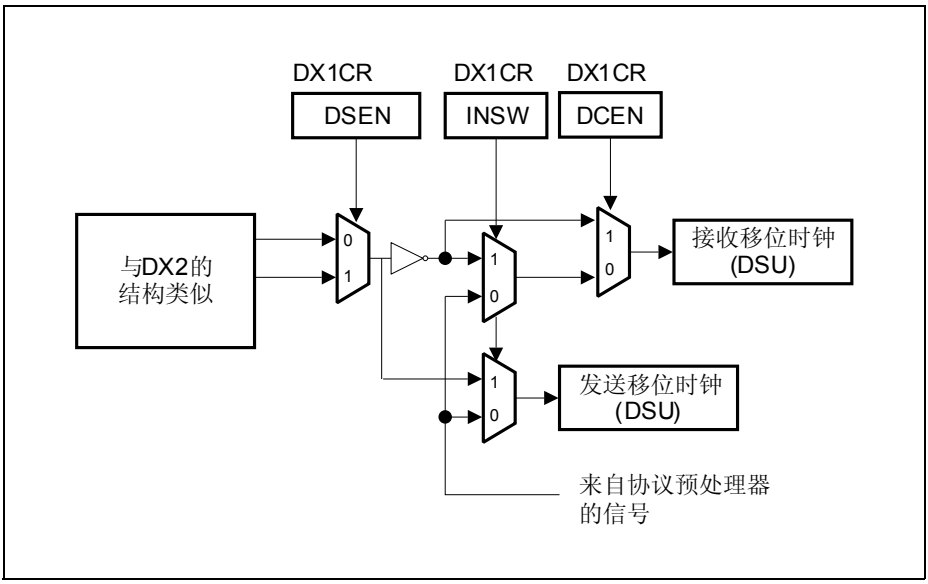


图 15-12　　DX1 的延迟补偿使能

15.2.3.2　数字滤波器

可以通过使能数字滤波器来降低输入信号上的噪声。在被滤波之前，输入信号变为与 f_{PB} 同步。如果禁用了滤波器，则信号 DXnS 对应同步输入信号。如果使能了滤波器，则信号 DXnS 上比一个滤波器采样周期短的脉冲被抑制。在同步输入信号的一个边沿之后，如果检测到新值的两个连续样本，则信号 DXnS 变成新值。

为了使滤波采样周期适应不同的应用，它可以被编程。第一种可能是系统频率 f_{PB}。如果选择小数分频器的输出频率 f_{FD}，则可以抑制更长的脉冲。该频率可在很宽的范围内编程，也可用于确定数据传送的波特率。

除了 2-3 个 f_{PB} 周期的同步延迟外，被使能的滤波器还在所选择的输入和信号 DXnS 之间增加了最多两个滤波器采样周期的延迟。

15.2.3.3　边沿检测

同步 （还可选选择滤波）信号 DXnS 可用作数据移位单元的输入，并且还是所选协议预处理器的一个输入。如果协议预处理器不使用 DXnS 信号进行协议特定处理，则 DXnS

可用于其他任务，例如在主模式下控制数据发送 (一个数据字可被标记为发送有效，请参见关于数据缓冲的章节)。

可编程边沿检测电路通过激活触发信号 DXnT 来指示所期望的事件已经发生 (对这个事件响应之前引入了一个 f_{PB} 周期的延迟)。

15.2.3.4 选择的输入监测

每个输入级的所选输入信号变成可用的 DXnINS 信号。这些信号可以在系统中用于触发其他动作，例如产生中断。

15.2.3.5 环回模式

在环回模式，USIC 发送器的输出信号可以被连接到同一通信通道相应接收器的输入。因此，必须选择所选协议需要的输入级的 “G” 输入。在这种情况下， ASC、 SSC 和 IIS 的驱动器可以在不连接到端口引脚情况下进行片上评估。发送器发送的数据可以被接收器接收，就好像该数据是由其他通信伙伴发送的一样。

15.2.4 操作波特率发生器

下面描述的功能块可以被配置为操作波特率发生器，参见 **图 15-2**。

15.2.4.1 分数分频器

分数分频器通过将输入频率 f_{PB} 除以一个整数系数 n 或乘以 n/1024 来产生其输出频率 f_{FD}。它有两种工作模式：

- 标准分频器模式 (FDR.DM = 01_B)：
  在该模式，输出频率 f_{FD} 是通过将输入时钟 f_{PB} 除以一个 1 到 1024 的整数而得到的。该除法基于一个由 f_{PB} 加 1 的计数器 FDR.RESULT。当计数值达到 $3FF_H$ 后，计数器被加载为 FDR.STEP 的值，然后继续计数。为了使 $f_{FD} = f_{PB}$， STEP 的值必须被编程为 $3FF_H$。
  标准分频器模式下的输出频率由下面的公式定义：

$$f_{FD} = f_{PB} \times \frac{1}{n} \qquad \text{其中 } n = 1024 - STEP \tag{15.1}$$

- 小数分频器模式 (FDR.DM = 10_B)：
  在该模式，输出频率 f_{FD} 来源于输入时钟 f_{PB} 乘以分数 n/1024，n 的值在 0 到 1023 之间。一般来说，与标准器分频器模式相比，小数分频器模式允许编程获得精度更高的平均输出时钟频率。请注意，在小数分频器模式， f_{FD} 的周期抖动最大可为一个 f_{PB} 周期。这种抖动是不对多个周期累加的。
  频率 f_{FD} 是通过在每个 f_{PB} 周期将 FDR.STEP 加到 FDR.RESULT 而产生的。频率 f_{FD}

基于加法结果超过 $3FF_H$ 产生的溢出。
小数分频器模式下的输出频率由下面的公式定义：

$$f_{FD} = f_{PB} \times \frac{n}{1024} \qquad \text{其中 } n = STEP \tag{15.2}$$

通过设置 BRG.CLKSEL = 00_B 可选择将小数分频器的输出频率 f_{FD} 用于波特率发生器。

15.2.4.2 外部频率输入

如果在所选协议中不需要输入级 DX1(DX1CTR.INSW = 0)，那么波特率可由一个外部频率输入 (而不是 f_{PB}) 产生。在这种情况下，出现在 DX1 输入级的一个外部频率输入信号可以被系统频率 f_{PB} 同步和采样。也可以选择在输入级的数字滤波器对其滤波。该功能允许使用非器件本身产生的频率进行数据传送，例如特定音频频率。

如果 BRG.CLKSEL = 10_B, 则触发信号 DX1T 决定 f_{DX1}。在该模式，输入信号的上升沿、下降沿或两个边沿都可以用来产生波特率，这取决于由位域 DX1CTR.CM 实现的 DX1T 触发事件配置。信号 MCLK 在每次发生 DX1T 的触发事件时切换。

如果 BRG.CLKSEL = 11_B，则输入信号的上升沿可以用于产生波特率。信号 MCLK 代表同步后的输入信号 DX1S。

外部输入信号的高电平时间和低电平时间都必须有至少 2 个 f_{PB} 周期的长度才能用于产生波特率。

15.2.4.3 分频器模式计数器

分频器模式计数器用于整数分频，提供输出频率 f_{PDIV}。此外，两个固定 2 分频的分频级提供具有 50% 占空比的输出信号 MCLK 和 SCLK。如果使用小数分频模式，在这些信号中还可能出现 1 个 f_{PB} 周期的最大小数抖动。该分频器的输出频率由寄存器 BRG 控制。

为了定义主时钟 MCLK 和移位时钟 SCLK 之间的频率比，MCLK 的分频级位于分频系数为 PDIV+1 的分频器的前面，而 SCLK 的分频级位于该分频器的输出之后。

$$f_{MCLK} = \frac{f_{PIN}}{2} \tag{15.3}$$

$$f_{SCLK} = \frac{f_{PDIV}}{2} \tag{15.4}$$

在主时钟被用作外部器件 (例如 IIS 组件) 参考时钟，并且需要主时钟与 SCLK 有固定的相位关系以及需要其他定时信号的情况下，建议使用 MCLK 信号作为 PDIV 分频器的输

入。如果不使用 MCLK 信号或者不需要固定的相位关系，则可以选择更快的频率 f_{PIN} 作为输入频率。

$$f_{PDIV} = f_{PIN} \times \frac{1}{PDIV + 1} \quad \text{若 PPPEN = 0}$$
$$f_{PDIV} = f_{MCLK} \times \frac{1}{PDIV + 1} \quad \text{若 PPPEN = 1} \qquad (15.5)$$

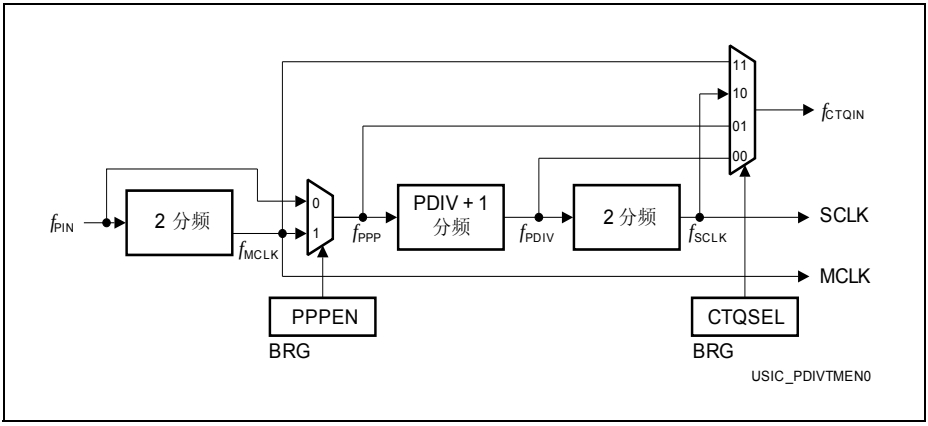


图 15-13　　分频器模式计数器

15.2.4.4　捕获模式定时器

捕获模式定时器用于测量时间间隔，由 BRG.TMEN = 1 使能。该定时器独立于分频器模式计数器工作。因此，当定时器进行定时测量时，任何串行数据接收和发送都可以继续。当该定时器对 f_{PPP} 周期计数，在达到其最大值时停止计数。此外，还产生一个波特率发生器中断事件 (位 PSR.BRGIF 变为置位状态)。

如果 DX0T 或 DX1T 指示发生了一个事件，则当前定时器值被捕获到到位域 CMTR.CTV，定时器从 0 开始重新计数。此外，还产生一个发送移位中断事件 (位 PSR.TSIF 变为置位状态)。

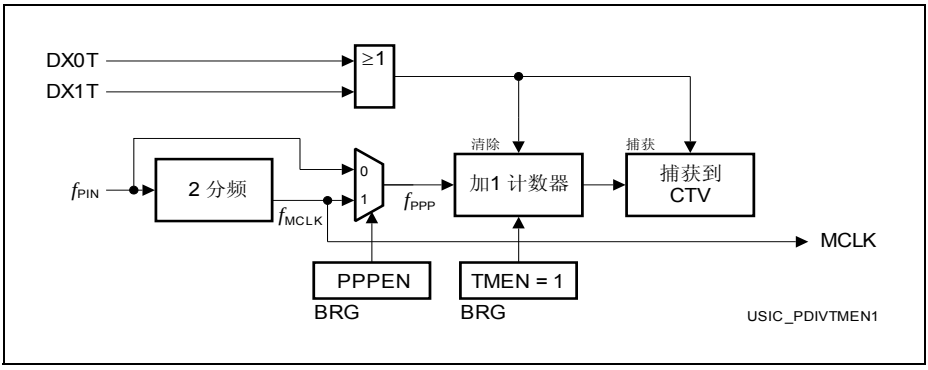


图 15-14　　协议相关的计数器 (捕获模式)

捕获模式定时器可以用来测量数据传送启动之前或数据传送期间的从模式波特率，例如测量在一个数据信号 (由 DX0T) 或一个移位时钟信号 (由 DX1T) 的两个边沿之间的时间。在每个输入级都可以配置激活 DXnT 触发信号的条件。

15.2.4.5　时间量子计数器

与协议预处理器相关的时间量子计数器 CTQ 允许产生用于协议专用的时间间隔。一个时间量子 t_q 的长度由所选输入频率 f_{CTQIN} 和编程的预分频器值给出。
时间量子的含义取决于所选择的协议。更多协议专有信息请参见相应章节。

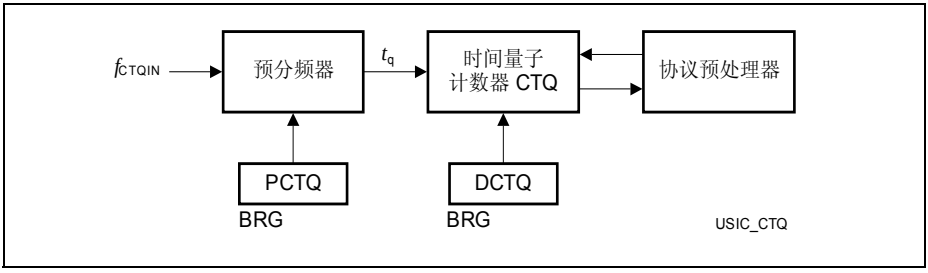


图 15-15　　时间量子计数器

15.2.4.6　主时钟和移位时钟配置

可以配置在相应的输出引脚上可用的主时钟输出信号 MCLKOUT 的极性。可以为每个协议产生 MCLK 信号，以提供一种比移位时钟更高频率的时基。

主时钟输出信号 MCLKOUT 的配置机制确保不会产生缩短的脉冲。每个 MCLK 周期包括两个阶段，一个主动阶段，后面跟随一个被动阶段。在主动阶段期间， MCLKOUT 信号的极性由位 BRG.MCLKCFG 的反相电平定义，在主动阶段的开始进行评估。在被动阶段期间，MCLKOUT 信号的极性由位 BRG.MCLKCFG 定义，在被动阶段的开始进行评估。

如果位 BRG.MCLKOUT 被编程为其他值，这种改变在下一次阶段改变时生效。该机制确保不会在 MCLKOUT 输出产生比一个阶段长度短的脉冲。在 图 15-16 所示的例子中，在 MCLK 周期 2 被动阶段期间， BRG.MCLKCFG 的值从 0 变为 1。

MCLKOUT 信号的产生由协议预处理器通过位 PCR.MCLK 使能 / 禁止。在该位变为置位状态之后，信号 MCLKOUT 在 MCLK 周期的下一个主动阶段产生。如果 PCR.MCLK = 0 (MCLKOUT 产生被禁止)，则被动阶段的电平也应用于主动阶段。

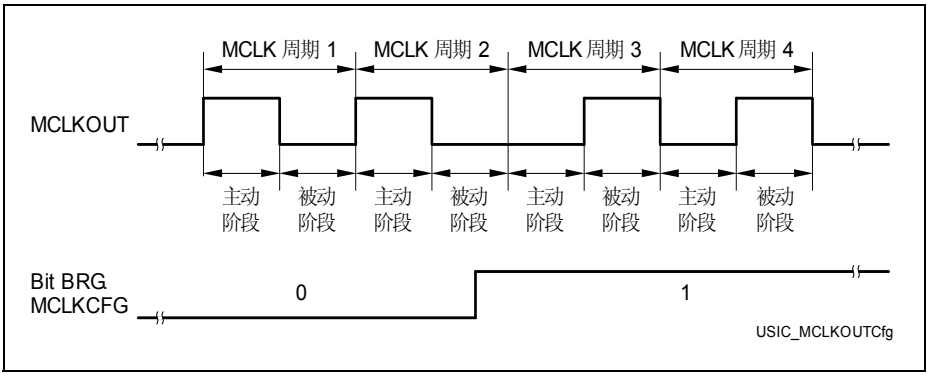


图 15-16　　主时钟输出配置

可以配置在相应的输出引脚上可用的移位时钟输出信号 SCLKOUT 的极性。可以为每个协议产生 MCLK 信号，以提供一种比移位时钟更高频率的时基。另外，可以引入一个 f_{PDIV} 周期 (等于半个 SCLK 周期) 的延迟。该延迟允许使移位时钟边沿的顺序适合应用需求。如果使用该延迟，则必须考虑信号传播时间和回路延迟的计算。

SCLKOUT 信号的极性控制机制与 MCLKOUT 类似，但由位域 BRG. SCLKCFG 控制。SCLKOUT 信号的产生由协议预处理器使能 / 禁止。根据所选择的协议，协议预处理器可以控制独立于分频器链的 SCLKOUT 信号的产生。例如，对于不需要在引脚上提供可用移位时钟的协议， SCLKOUT 产生被禁止。

15.2.5　操作发送数据通路

发送数据通路基于 16 位宽的发送移位寄存器 (TSR 和 TSR[3:0]) 和一个发送缓冲器 TBUF。发送和接收共同的数据传送参数，如数据字长度、数据帧长度或移位方向，由移位控制寄存器 SCTR 控制。发送控制和状态寄存器 TCSR 控制发送数据处理并监视发送状态。

数据移位输出信号 DOUTx 值的改变仅发生在移位时钟输入信号的相应边沿。一个数据字 / 帧的最后一个数据位的电平在 DOUTx 保持不变，直到在移位时钟的下一个相应边沿开始下一个数据字传送。

15.2.5.1 发送缓冲

发送移位寄存器不能被软件直接访问，因为在当前发送的数据字结束并且用于发送的新数据为有效时，这些寄存器会被自动更新为存储在发送缓冲器 TBUF 中的值。可以将数据字直接加载到 TBUF，这可通过写一个发送缓冲器输入单元 TBUFx (见页15-27) 实现，或者选择由一个 FIFO 缓冲区级加载 (见 页 15-33)。

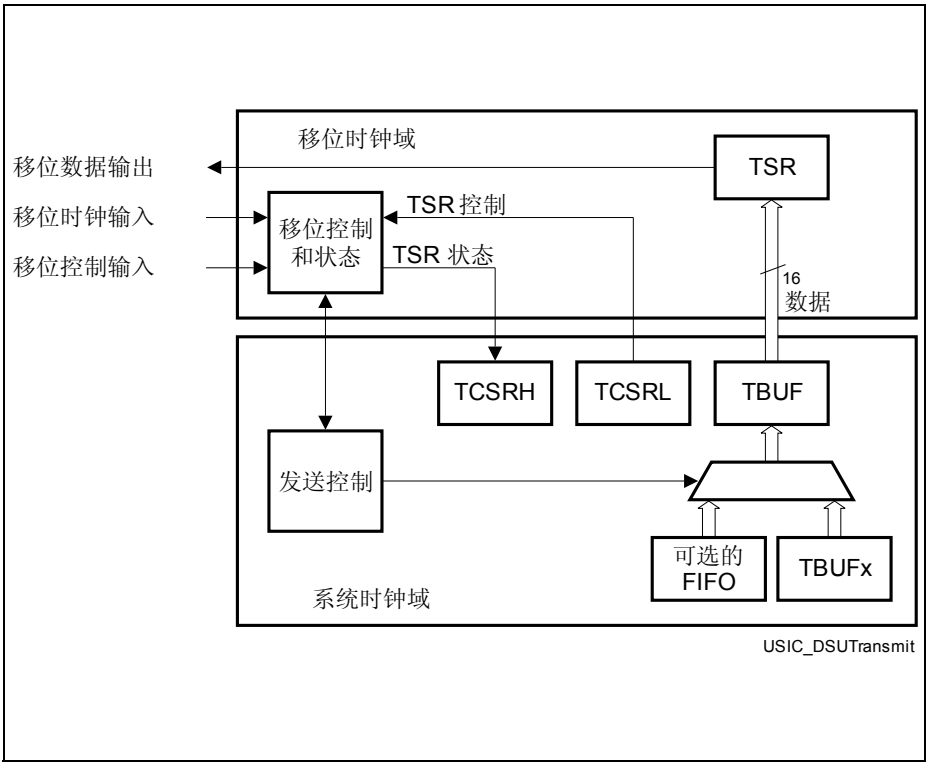


图 15-17　　发送数据通路

15.2.5.2 发送数据移位模式

通过使用相应数量的输出线，可以选择每次移出一位、两位或四位发送移位数据。该选项允许 USIC 支持如双位和四位 SSC 这样的协议。这种选择通过移位控制寄存器 SCTR 中的位域 DSM 实现。

注：　位域 SCTR.DSM 控制发送和接收通路的数据移位模式，以允许通过一到四个数据线来发送和接收数据。

对于具有两位或者四位并行数据输出的移位模式，数据字和帧的长度必须分别是二或四的整数倍。输出一个特定的数据字或数据帧长度所需要的数据移位次数因而被减少到数据字或数据帧长度除以并行数据输出线的数量。例如，要通过 4 个输出线发送一个 16 位的数据字，仅需四次移位。

表 15-7 根据移位模式列出了使用不同位组合的不同发送移位寄存器。请注意，表中的 'n' 表示移位次数减一，即第一次数据移位 n = 0，第二次数据移位 n = 1，以此类推，直到达到总移位次数减一。

对于所有的发送移位寄存器，移出的第一位是MSB 还是LSB取决于SCTR.SDIR的设置。

表 15-7　　发送移位寄存器组成

发送移位寄存器	单数据输出 (SCTR.DSM = 00_B)	双数据输出 (SCTR.DSM = 10_B)	四数据输出 (SCTR.DSM = 11_B)
TSR	所有数据位	未使用	未使用
TSR0	未使用	位 n*2	位 n*4
TSR1	未使用	位 n*2 + 1	位 n*4 + 1
TSR2	未使用	未使用	位 n*4 + 2
TSR3	未使用	未使用	位 n*4 + 3

15.2.5.3　发送控制信息

发送控制信息 TCI 是一个 5 位的值，它源自所写 TBUFx 或 INx 输入单元的地址 x。例如，写 TBUF31 生成一个为 11111_B 的 TCI。

TCI 可以作为一个额外的数据传送控制参数，用于动态改变数据字长度、数据帧长度或其他协议特有功能 (有关该主题的详细信息，请参见相应协议的章节)。TCI 在不同应用中的使用方式可由寄存器 TCSR 中的位 WLEMD、FLEMD、SELMD、WAMD 和 HPCMD 编程。请注意，并非所有可能的设置都能导致有用的系统行为。

- 字长控制：
  如果 TCSR.WLEMD = 1，当向一个发送缓冲器输入单元 TBUFx 写入时，位域 SCTR.WLE 被更新为 TCI[3:0] 的值。该功能可在所有协议中使用，以动态改变每个数据字的长度 （在 1 和 16 个数据位之间）。
  此外，位 TCSR.EOF 被更新为 TCI[4] 的值。该功能 可在 SSC 主模式用于控制从选择产生，以结束数据帧。建议编程为 TCSR.FLEMD = TCSR.SELMD = TCSR.WAMD = TCSR.HPCMD = 0。
- 帧长控制：
  如果 TCSR.FLEMD = 1，当向一个发送缓冲器输入单元 TBUFx 写入时，位域 SCTR.FLE[4:0] 被更新为 TCI[4:0] 的值，并且 SCTR.FLE[5] 变成 0。该功能可在所有协议中使用，以动态改变每个数据帧的长度 （在 1 和 32 个数据位之间）。建议编程为 TCSR.SELMD = TCSR.WLEMD = TCSR.WAMD = TCSR.HPCMD = 0。
- 选择输出控制：
  如果 TCSR.SELMD = 1，当向一个发送缓冲器输入单元 TBUFx 写入时，位域

PCR.CTR[20:16] 被 更新为 TCI[4:0] 的值，并且 PCR.CTR[23:21] 变成 0。该功能可在SSC主模式使用，以定义目标从器件。建议编程为 TCSR.WLEMD = TCSR.FLEMD = TCSR.WAMD = TCSR.HPCMD = 0。

- 字地址控制：
  如果 TCSR.WAMD = 1，当向一个发送缓冲器输入单元 TBUFx 写入时，位 TCSR.WA 被更新为 TCI[4] 的值。该功能可在 IIS 模式使用，以定义数据字是在右通道还是在左通道发送。建议编程为 TCSR.WLEMD = TCSR.FLEMD = TCSR.SELMD = TCSR.HPCMD = 0。
- 硬件端口控制：
  如果 TCSR.HPCMD = 1， 当向一个发送缓冲器输入单元 TBUFx 写入时，位域 SCTR.DSM 被 更新为 TCI[1:0] 的值。该功能可在 SSC 协议中用于动态改变数据输入和输出线的数量，以建立标准、 2 位和 4 位 SSC 格式。
  此外，位 TCSR.HPCDIR 被更新为 TCI[2] 的值。该功能可在 SSC 协议中使用，当通过设置 CCR.HPCEN = 1 使能了硬件端口控制时，该功能控制引脚的方向。建议编程为 TCSR.FLEMD = TCSR.WLEMD = TCSR.SELMD = TCSR.WAMD = 0。

15.2.5.4 发送数据验证

位于发送缓冲器 TBUF 中的数据字可以被位 TCSR.TDV (发送数据有效) 标记为发送有效或无效。用数据流相关和事件相关判断条件的组合来判定数据字是否为有效发送状态。数据验证逻辑检查每个数据字的起始条件。 根据检查结果，发送移位寄存器按照下面的规则被加载不同的值：

- 如果 USIC 通道是通信主机 (它定义每个数据字传送的开始)，则只能使用发送缓冲器 TBUF 中的有效数据开始一次数据字传送。 在这种情况下，发送移位寄存器被装载为 TBUF 的内容， TBUF 的值不会因这次操作而改变。
- 如果 USIC 通道是通信从机 (它不能定义传送开始，但必须做出反应)，由通信主机请求的一次数据字传送必须被启动，与 TBUF 中数据字的状态无关。如果主机请求并启动了一次数据字传送，则发送移位寄存器在第一个相应的移位时钟边沿被加载为 TBUF 中的数据字 (如果发送有效)，或被加载为由位 SCTR.PDL 定义的电平 (如果在发送开始时TBUF的内容还不是有效的)。在这两种情况下，TBUF的内容都不改变。

用于数据验证的控制和状态位位于寄存器 TCSR 中。数据验证基于 图 15-18 所示的逻辑块。

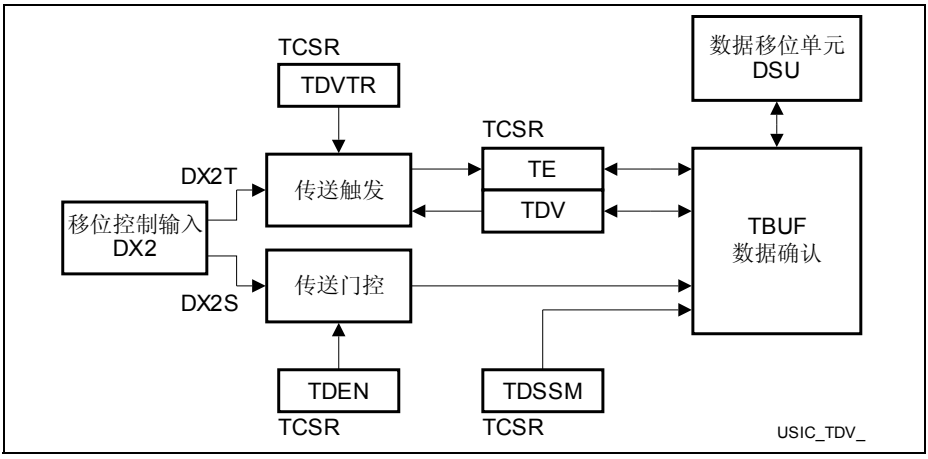


图 15-18　　发送数据验证

- 传送门控逻辑在软件或硬件控制下使能或禁止从 TBUF 发送数据。如果数据移位不需要使用输入级 DX2，则信号 DX2S 可用于门控目的。传送门控逻辑由位域 TCSR.TDEN 控制。
- 传送触发逻辑支持与事件相关的数据字传送，例如基于定时器的事件或与输入引脚相关的事件。如果数据移位不需要使用输入级 DX2，则信号 DX2T 可用于触发目的。传送触发逻辑由位 TCSR.TDVTR 控制，触发事件的发生由位 TCSR.TE 指示。例如，在 RS-232 协议中，在 DX2 上收到清除发送 (CTS) 信号这一事件可以用来触发数据传送。
- 数据验证逻辑将来自门控逻辑、触发逻辑和 DSU 的输入信号组合在一起。只有在门控逻辑允许启动，位 TCSR.TDV = 1 且位 TCSR.TE = 1 的情况下，位于 TBUF 中的数据字的发送才能被启动。
  当发送缓冲器 TBUF 的内容为发送有效时，不应用新数据覆盖 TBUF，并且这时可以启动一次新的发送。如果 TBUF 的内容必须被更改时，建议在更新数据前通过设置 FMR.MTDV = 10_B 来清除 TCSR.TDV 位。当 TBUF 被新数据更新时，位 TCSR.TDV 被自动置位。另一种可能是用中断 TBI (对于 ASC 和 IIC) 或 RSI (对于 SSC 和 IIS) 来指示一次发送已经开始。当发送正在进行时，可以向 TBUF 加载新数据。在这种情况下，用户必须注意要在一次新的发送开始之前更新 TBUF 的内容。

采用这种结构，可以实现下述数据传送功能：

- 如果位 TCSR.TDSSM = 0，发送缓冲器 TBUF 的内容总是被视为发送有效。传送触发机制可用于根据所选择的事件 (例如基于定时器的事件或一个引脚的边沿) 来启动对同一数据字的发送，以实现一种生命体征机制。此外，在从模式下，该机制可以确保总是发送正确的数据字，而不是被动数据电平。
- 为了能使用一种单次机制逐字地进行数据发送，必须将位 TCSR.TDSSM 编程为 1。在每次发送开始之后，必须向发送缓冲器 TBUF 加载一个新数据字。可以通过软件写一个发送缓冲器输入单元 TBUFx 的方式加载新数据，也可以通过一个可选的数据缓

冲区 (例如 FIFO 缓冲区) 加载。为了避免数据字被发送多次，或允许使用额外的数据缓冲区进行数据处理，位 TCSR.TDSSM 必须为 1。

- 当一个新数据字被加载到发送缓冲器 TBUF 时，位 TCSR.TDV 自动变为置位状态，此时可以通过将待发送数据写入到一个发送缓冲器输入单元 TBUFx（至少是低字节）来请求开始一次发送。额外的信息 TCI 可用于控制数据字长度或每个数据字独立的其他参数，仅需一次写访问即可。
- 位域 FMR.MTDV 允许用软件修改 (置位或清除) 位 TCSR.TDV。将该位域与门控控制位域 TCSR.TDEN 一起使用，用户可以设置发送数据字，而不启动发送。一种可能的编程顺序是：设置TCSR.TDEN = 00_B，写数据到 TBUFx，通过写 FMR.MTDV = 10_B 来清除 TCSR.TDV，通过设置 TCSR.TDEN = 01_B 重新使能门控，然后在软件控制下通过写 FMR.MTDV = 01_B 置位 TCSR.TDV。

15.2.6 操作接收数据通路

接收数据通路基于两组 16 位宽的接收移位寄存器 RSR0[3:0] 和 RSR1[3:0]，以及每组一个接收缓冲器 (RBUF0 和 RBUF1)。发送和接收共同的数据传送参数，如数据字的长度、数据帧长度或移位方向，由移位控制寄存器 SCTR 控制。

寄存器 RBUF01SR 监视 RBUF0 和 RBUF1 的状态。

15.2.6.1 接收缓冲

接收移位寄存器不能被软件直接访问，但如果一个完整的数据字已被接收或一个完整的数据帧已经结束，则接收移位寄存器的内容被自动加载到接收缓冲器寄存器 RBUF0 (或 RBUF1)。可以从寄存器 RBUF 以正确的顺序直接读取 RBUF0 或 RBUF1 中的接收数据字，也可以选择从 FIFO 缓冲区级读取 (见 页 15-33)。

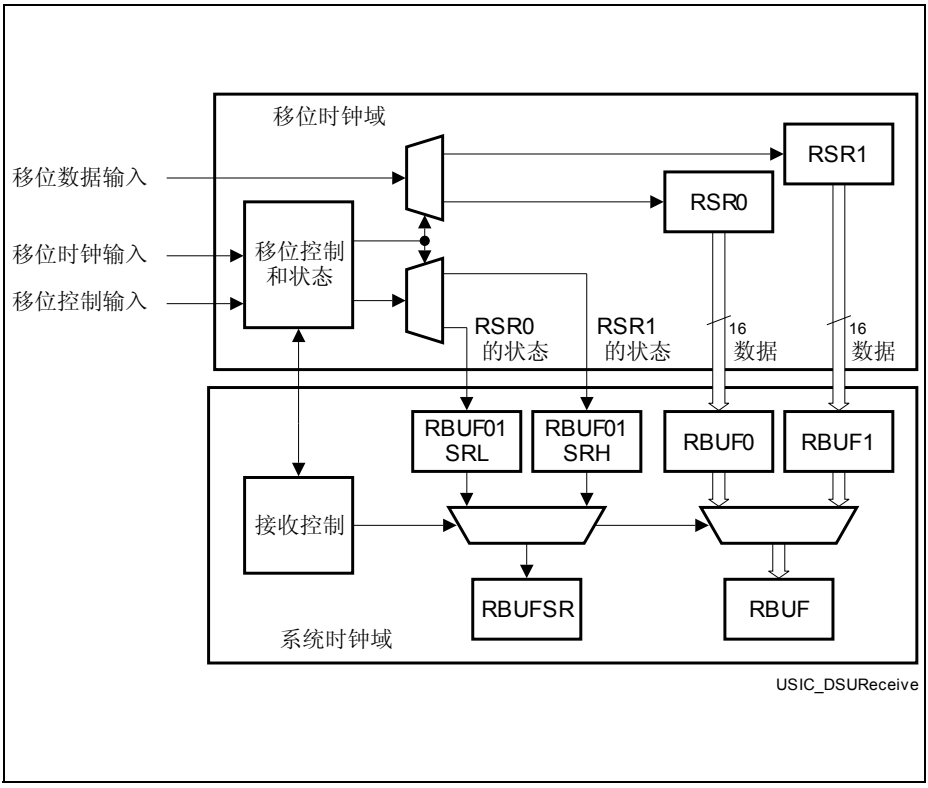


图 15-19　　接收数据通路

15.2.6.2　接收数据移位模式

通过使用相应数量的输入级和数据输入线，可以选择每次移入 1 位、 2 位或 4 位接收数据。该选项允许 USIC 支持如 2 位和 4 位 SSC 这样的协议。这种选择通过移位控制寄存器 SCTR 中的位域 DSM 实现。

注：　位域 SCTR.DSM 控制发送和接收通路的数据移位模式，以允许通过一到四个数据线来发送和接收数据。

对于具有两位或者四位并行数据输入的移位模式，数据字和帧的长度必须分别是二或四的整数倍。因此，输入一个特定的数据字或数据帧长度所需要的数据移位次数被减少到数据字或数据帧长度除以并行数据输入线数。例如，要通过 4 个输入线接收一个 16 位的数据字，仅需 4 次移位。

表 15-8 根据移位模式列出了使用不同位组合的不同接收移位寄存器。请注意，表中的 'n' 表示移位次数减一，即第一次数据移位 n = 0，第二次数据移位 n = 1，以此类推，直到达到总移位次数减一。

对于所有的接收移位寄存器，移出的第一位是 MSB还是LSB取决于 SCTR.SDIR的设置。

表 15-8 接收移位寄存器组成

接收移位寄存器	使用的输入级	单数据输入 (SCTR.DSM = 00_B)	双数据输入 (SCTR.DSM = 10_B)	四数据输入 (SCTR.DSM = 11_B)
RSRx0	DX0	所有数据位	位 n*2	位 n*4
RSRx1	DX3	未使用	位 n*2 + 1	位 n*4 + 1
RSRx2	DX4	未使用	未使用	位 n*4 + 2
RSRx3	DX5	未使用	未使用	位 n*4 + 3

15.2.6.3 波特率约束

必须遵守下面的波特率约束，以确保正确的数据接收和缓冲。当根据模块时钟频率 f_{PB} 来选择波特率和数据字长度时，用户必须注意这些限制。

- 为了确保正确地加载接收缓冲寄存器 RBUFx，相关的接收移位寄存器 RSRx[3:0] 中的接收数据字必须保持不变至少 4 个 f_{PB} 周期。
- 为了正确地检测到一个帧的结束，在两个连续帧之间，移位控制信号必须保持非活动状态不变至少 5 个 f_{PB} 周期。
- 为了正确地检测一个帧 (最短帧)，移位控制信号必须保持活动状态不变至少一个 f_{PB} 周期。
- 必须保证移位控制信号相对于移位时钟信号的最小建立和保持时间。

15.2.7 硬件端口控制

硬件端口控制是为具有半双工配置的 SSC 协议而准备的，以通过一个专用的硬件接口来控制引脚的方向。在这种情况下，一个端口引脚可用于输入和输出数据两种功能。除了输入拉器件选择和输出驱动器类型 (漏极开路或推挽) 以外， Pn_IOCRy.PCx 中的所有设置都由硬件端口控制来管理。

输入拉器件的选择通过前面提到的 Pn_IOCRy.PCx 完成，而输出驱动器在该模式下被固定为推挽方式。

通过硬件端口控制可以选择一个、两个或四个端口引脚，以支持使用多个双向数据线的 SSC 协议，如双位和四位 SSC。这种选择和硬件端口控制的使能 / 禁止是通过 CCR.HPCEN 实现的。所有选定引脚的方向通过单个位 SCTR.HPCDIR 控制。

在每次数据字传送开始时， SCTR.HPCDIR 被自动屏蔽，这是为了防止在一次数据字传送中间改变引脚的方向。

15.2.8 操作 FIFO 数据缓冲区

USIC 模块的 FIFO 数据缓冲区都以类似的方式构建，每个通道都有发送缓冲区和接收缓冲区能力。不同型号器件的可用 FIFO 缓冲区区域可能不同。在 XMC1300 中，共有 64 个缓冲区项，可以分布在 USIC 模块的两个通道的发送或接收 FIFO 缓冲区。

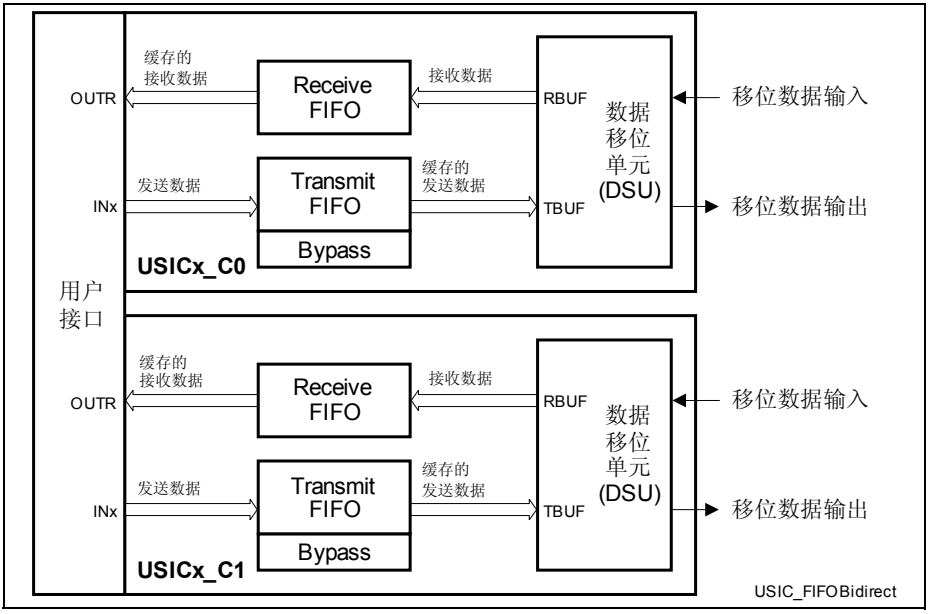


图 15-20 FIFO 缓冲区概览

为了操作 FIFO 数据缓冲区，必须要考虑以下问题：

- FIFO 缓冲区的可用性和选择：
  如果 CCFG.TB = 1，则发送 FIFO 缓冲区和旁路结构是唯一可用的，而如果 CCFG.RB = 1，则接收缓冲区是唯一可用的。
  建议在该 USIC 通道没有数据通信并且通过设置 TBCTR.SIZE = 0 (对发送缓冲区) 或 RBCTR.SIZE = 0 (对接收缓冲区) 禁止了 FIFO 机制时配置所有的缓冲区参数。必须在相应的 FIFO 缓冲区被禁止时通过写 TBCTR 或 RBCTR 来分配一个缓冲区。FIFO 缓冲中断控制位的修改与数据通信无关。
- FIFO 缓冲区设置：
  可用的 FIFO 缓冲区项的总数量限制了每个 USIC 通道发送和接收缓冲区的长度。
- 旁路设置：
  除了发送 FIFO 缓冲区之外，还可以对旁路机制进行配置，见 页 15-42 的描述。

15.2.8.1 FIFO 缓冲区划分

如果可用的话， FIFO 缓冲区由指定数量的 FIFO 缓冲项组成，每项都包含一个数据部分和相关联的控制信息 (用于接收数据的 RCI，用于发送数据的 TCI)。一个 FIFO 缓冲项代表可被分配到一个接收 FIFO 缓冲区或发送 FIFO 缓冲区的最小粒度。一个 USIC 模块的所有可用 FIFO 缓冲项在 FIFO 缓冲区中是连续排列的。总体计数从 FIFO 项 0 开始，然后是 1、 2 等等。

每个 USIC 模块都有一定数量的 FIFO 项可用，这些 FIFO 项可被分配给同一 USIC 模块的多个通道。不可能将 FIFO 缓冲区分配给不在同一 USIC 模块的 USIC 通道。
对于每个 USIC 通道，可以独立地选择发送和接收 FIFO 缓冲区的大小。例如，可以将全部可用的 FIFO 项都作为一个 USIC 通道的发送缓冲区来分配。 FIFO 缓冲区划分的一些可能情况如**图 15-21** 所示。

每个 FIFO 缓冲区由一组连续的 FIFO 项组成。一个 FIFO 数据缓冲区的大小只能被编程为 2 的整数次幂，从 2 个 FIFO 项开始，然后是 4 个 FIFO 项，接着是 8 个 FIFO 项，以此类推。一个 FIFO 数据缓冲区只能从一个与其大小对齐的 FIFO 项开始。例如，一个包含 n 个 FIFO 项的 FIFO 缓冲区只能从 FIFO 项 0，n、2*n、3*n 等开始，由 FIFO 项 [x*n, (x+1)*n-1] 组成，其中 x 是一个整数 (包括 0)。在一个 FIFO 缓冲区中不可能存在由未使用的 FIFO 项构成的 “ 洞 ”，但在两个 FIFO 缓冲区之间可以有未使用的 FIFO 项。

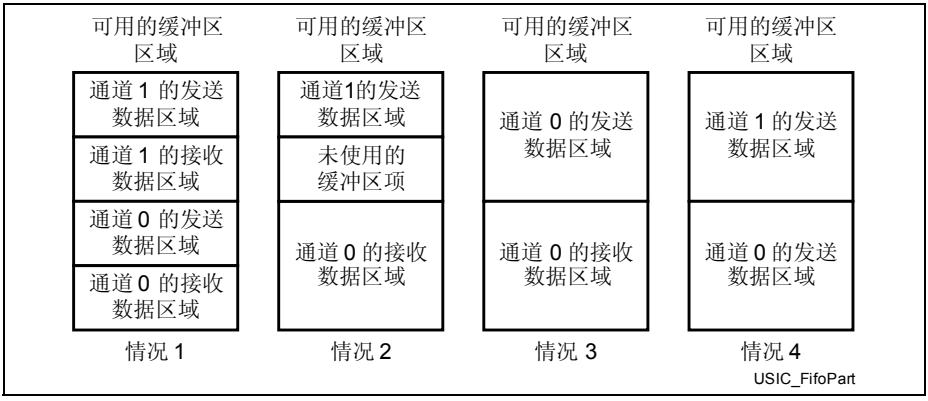


图 15-21 FIFO 缓冲区划分

FIFO 缓冲区内部的数据存储是基于指针的。只要 FIFO 缓冲区的数据内容被修改，指针就会被内部更新。当新数据被放入一个 FIFO 缓冲区或最老的数据被从 FIFO 缓冲区移走时，指针更新会自动发生。其结果是，用户程序在处理数据时不需要修改指针。只能在初始化阶段定义 FIFO 缓冲区的起始项，这可通过向寄存器 RBCTR(对于接收 FIFO 缓冲区) 或 TBCTR (对于发送 FIFO 缓冲区) 中的相应位域 DPTR 写 FIFO 缓冲区的第一个 FIFO 缓冲项的序号来实现，应在该写操作发生之前将相关位域设置为 RBCTR.SIZE=0(或 TBCTR.SIZE = 0，分别对应)。当一个 USIC 通道正在进行数据通信时，不允许为相关的 FIFO 缓冲区分配缓冲项 (关于大小和指针)。

15.2.8.2 发送缓冲区事件和中断

发送 FIFO 缓冲区机制检测到以下事件时会产生中断 (如果被使能):

- 标准发送缓冲区事件
- 发送缓冲区错误事件

标准发送缓冲区事件

当发送缓冲区的填充水平 (由 TRBSR.TBFLVL 给出) 超过 (TBCTR.LOF = 1) 或低于 (TBCTR.LOF = 0)[1] 一个编程极限值 (TBCTR.LIMIT) 时，标准发送缓冲区事件被触发。

如果具有TRBSR.STBT功能的事件触发被禁止 (TBCTR.STBTEN = 0)，则标准发送缓冲区事件的触发基于从等于到低于或高于的转变，而非实际的低于或高于。

如果 TBCTR.STBTEN = 1，则填充水平低于或者高于编程极限值的转变还会置位 TRBSR.STBT。每当发生一个数据传送到 TBUF 的事件或写数据到 INx 的事件时，该位还会触发标准发送缓冲区事件，这取决于 TBCTR.LOF 的设置。

清除 TRBSR.STBT 的方式取决于触发模式 (由 TBCTR.STBTM 选择)。如果 TBCTR.STBTM = 0，那么当缓冲区填充水平再一次等于编程的极限值时 (TRBSR.TBFLVL = TBCTR.LIMIT)， TRBSR.STBT 被硬件清除。若 TBCTR.STBTM = 1，则当缓冲区填充水平等于缓冲区大小的时候 (TRBSR.TBFLVL = TBCTR.SIZE)， TRBSR.STBT 被硬件清除。

注： 只有当发送缓冲区填充水平超过或低于编程的极限值时 (取决于 TBCTR.LOF 的设置)，标志 TRBSR.STBI 才被置位。由 TRBSR.STBT 触发的标准发送缓冲器事件不置位该标志。

图 15-22 示出了具有不同 TBCTR.STBTEN 和 TBCTR.STBTM 设置的标准发送缓冲器事件的例子。这些例子的目的在于说明硬件的行为，可能并不真正代表实际的应用用例。

1) 如果标准发送缓冲区事件用于指示必须向 一个 INx 单元写入新数据，则应该编程为 TBCTR.LOF = 0。

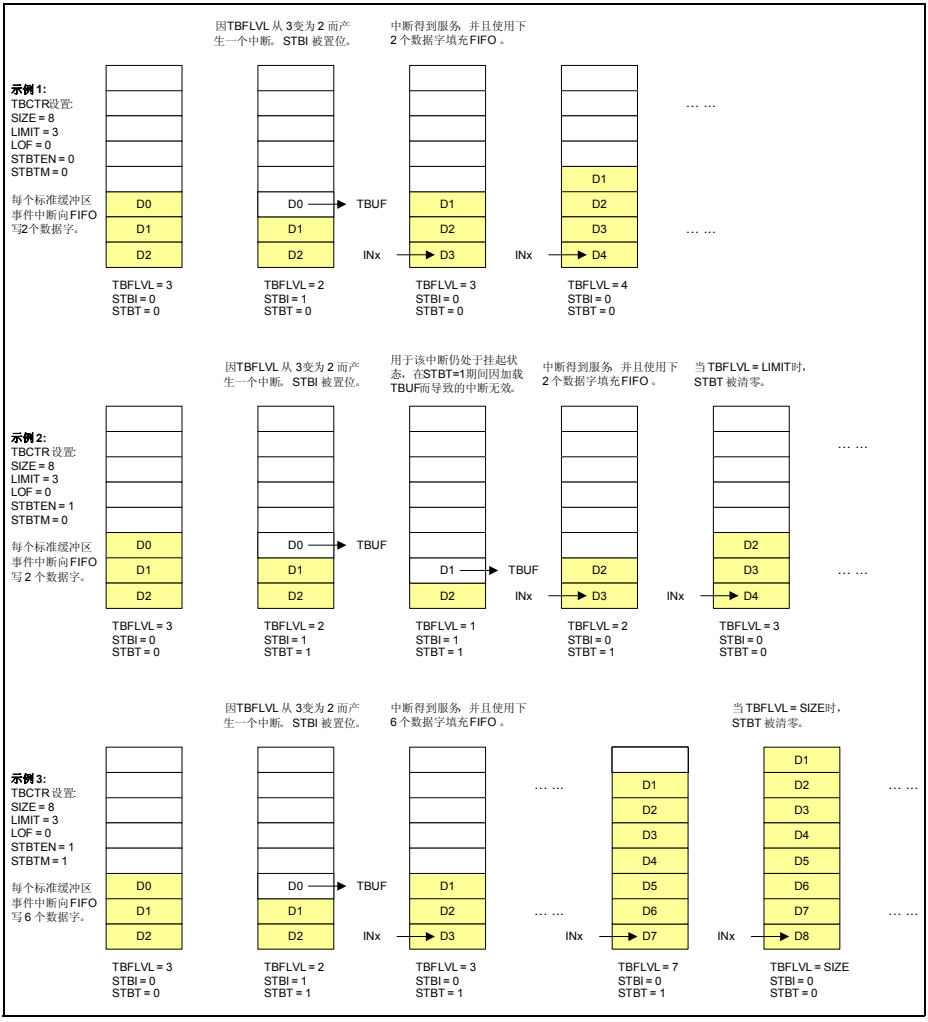


图 15-22　　标准发送缓冲器事件示例

发送缓冲区错误事件

当软件向一个已满缓冲区写入时会触发发送缓冲器错误事件。所写值被忽略。

发送缓冲区事件和中断处理

图 15-23 示出了发送缓冲区事件和中断。

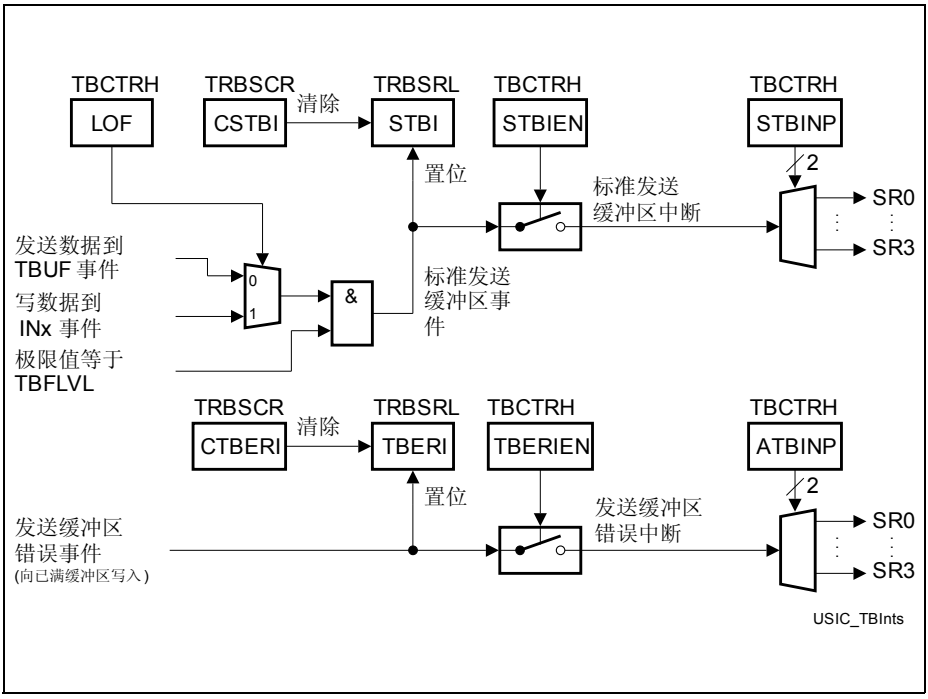


图 15-23　　发送缓冲区事件

表 15-9 列出了一个 USIC 通道中用于指示发送缓冲区事件和控制与发送 FIFO 缓冲区相关中断的寄存器、位和位域。

表 15-9　　发送缓冲区事件和中断处理

事件	指示标志	指示清除位	中断使能位	SRx 输出选择位
标准发送缓冲区事件	TRBSR.STBI	TRBSCR.CSTBI	TBCTR.STBIEN	TBCTR.STBINP
	TRBSR.STBT	由硬件清除		
发送缓冲区错误事件	TRBSR.TBERI	TRBSCR.CTBERI	TBCTR.TBERIEN	TBCTR.ATBINP

15.2.8.3　接收缓冲区事件和中断

接收 FIFO 缓冲区机制检测到以下事件时会导致产生中断 (如果被使能)：

- 标准接收缓冲区事件
- 备用接收缓冲区事件
- 接收缓冲区错误事件

标准接收缓冲事件和备用接收缓冲事件可以编程为两种不同的模式，一种方式涉及接收缓冲区的填充水平，另一种方式与 OUTR 中变为可用数据字的接收控制信息 RCI 中一个位的位置有关。

如果中断的产生与接收 FIFO 缓冲区的填充水平相关，则只能使用标准接收缓冲区事件，而不能使用备用接收缓冲区事件。可以选择使用这种模式来指示已经收到一定数量的数据，与相关联的 RCI 的内容无关。

如果中断的产生与 RCI 相关，则不考虑填充水平。每当在 OUTR 中有新数据变为可用时，就会检测到一个事件。如果位 RCI[4] = 0，会报告一个标准接收缓冲区事件，否则会报告一个备用接收缓冲区事件 (RCI[4] = 1)。根据所选择的的协议和 RBCTR.RCIM 的设置，RCI[4] 的值可以保持不同的信息，该信息可用于协议特有的中断处理 (更详细的信息请见描述协议的各节)。

填充水平模式下的标准接收缓冲区事件

在填充水平模式 (RBCTR.RNM = 0)，当接收缓冲区的填充水平[1](由 TRBSR.RBFLVL 给出) 超过 (RBCTR.LOF = 1) 或低于 (RBCTR.LOF = 0) 一个编程极限值 (RBCTR.LIMIT) 时，标准接收缓冲区事件被触发。

如果具有位 TRBSR.SRBT 功能的事件触发被禁止 (RBCTR.SRBTEN = 0)，那么标准接收缓冲区事件的触发基于填充水平从等于到低于或高于极限值的转变，而非实际的低于或高于。

如果 RBCTR.SRBTEN = 1，则填充水平低于或高于编程极限值的转变还会将 TRBSR.SRBT 置位。每当发生一个数据读出事件或新数据接收事件时，该位还会触发标准接收缓冲区事件，具体取决于 RBCTR.LOF 的设置。

清除 TRBSR.SRBT 的方式取决于触发模式 (由 RBCTR.SRBTM 选择) 。如果 RBCTR.SRBTM = 0，当缓冲区填充水平再次等于编程极限值 (TRBSR.RBFLVL = RBCTR.LIMIT) 时，TRBSR.SRBT 被硬件清除。如果 RBCTR.SRBTM = 1，则当缓冲区填充水平等于 0 (TRBSR.RBFLVL = 0) 时， TRBSR.SRBT 被硬件清除。

注： 仅当接收缓冲区填充水平超过或低于编程设定的极限值 (取决于 RBCTR.LOF 的设置) 时，标志 TRBSR.SRBI 才被置位。由 TRBSR.SRBT 触发的标准接收缓冲区事件不置位该标志。

图 15-24 示出了具有不同 RBCTR.SRBTEN 和 RBCTR.SRBTM 设置的标准接收缓冲区事件的例子。这些示例的目的在于说明硬件的行为，可能不真正代表实际应用的用例。

1) 如果标准接收缓冲区事件用于指示必须从 OUTR 中读出新数据，则应该编程为 RBCTR.LOF = 1。

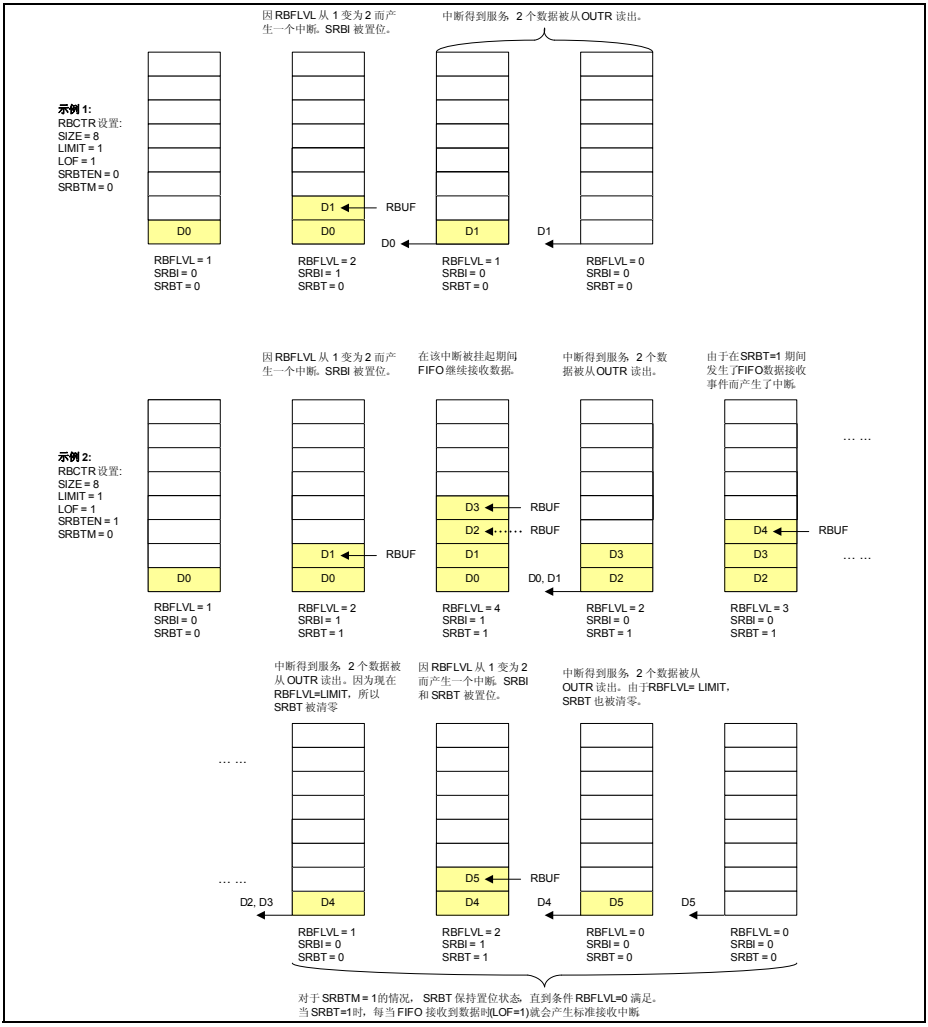


图 15-24　　标准接收缓冲区事件示例

RCI 模式下的标准和备用接收缓冲区事件

在 RCI 模式 (RBCTR.RNM = 1)，当 OUTR 级被一个 RCI[4] = 0 的新数据更新时，标准接收缓冲区事件被触发。

如果 OUTR 级被一个 RCI[4] = 1 的新数据值更新，则备用接收缓冲区事件被触发。

接收缓冲区错误事件

如果软件从一个空缓冲区读取数据，则无论 RBCTR.RNM 为何值，都会触发一个接收缓冲区错误事件。所读数据无效。

接收缓冲区事件和中断处理

图 15-25 示出了填充水平模式下的接收缓冲区事件和中断。

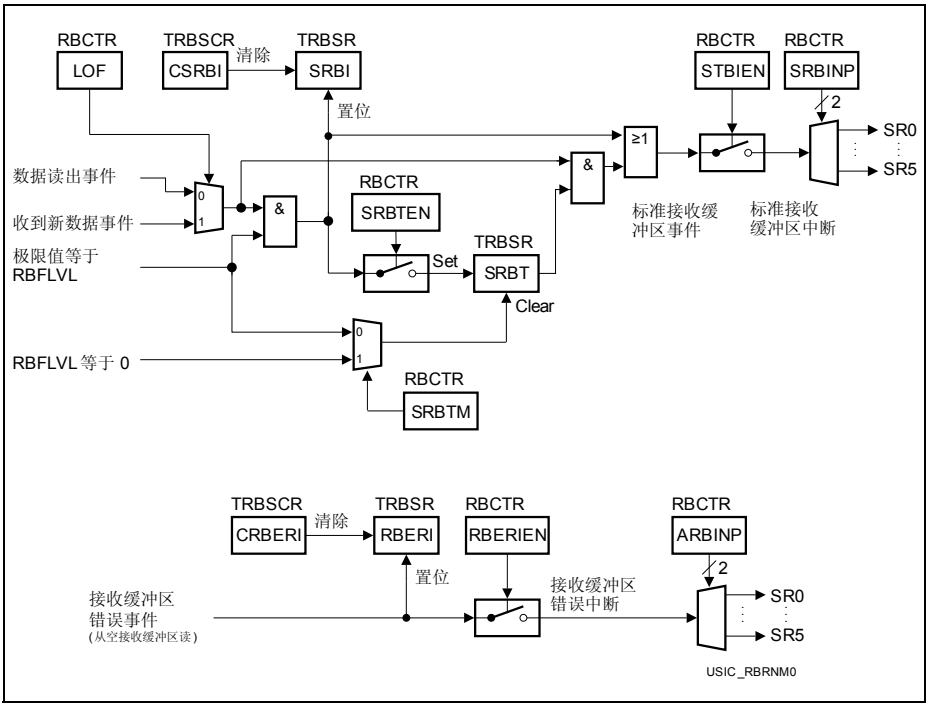


图 15-25　　填充水平模式下的接收缓冲区事件

图 15-26 示出了 RCI 模式下的接收缓冲区事件和中断。

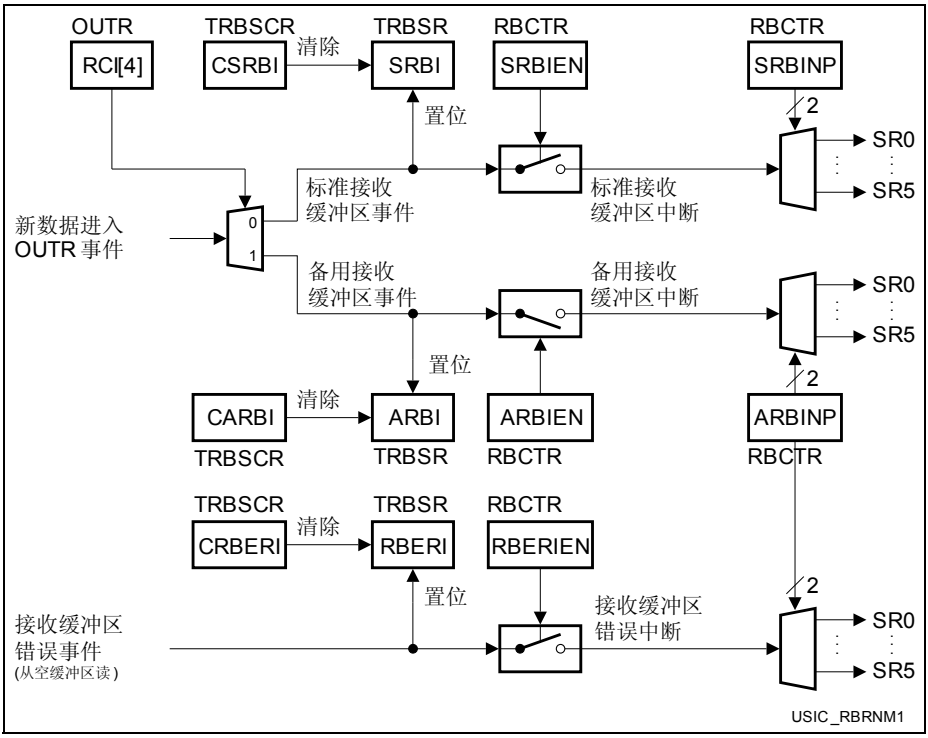


图 15-26　　RCI 模式下的接收缓冲区事件

表 15-10 列出一个 USIC 通道中用于指示接收缓冲区事件和控制与接收 FIFO 缓冲区相关中断的寄存器、位和位域。

表 15-10　　接收缓冲区事件和中断处理

事件	指示标志	指示清除位	中断使能位	SRx 输出选择位
标准接收缓冲区事件	TRBSR.SRBI	TRBSCR.CSRBI	RBCTR.SRBIEN	RBCTR.SRBINP
	TRBSR.SRBT	由硬件清除		
备用接收缓冲区事件	TRBSR.ARBI	TRBSCR.CARBI	RBCTR.ARBIEN	RBCTR.ARBINP
接收缓冲区错误事件	TRBSR.RBERI	TRBSCR.CRBERI	RBCTR.RBERIEN	RBCTR.ARBINTXDP

15.2.8.4 FIFO 缓冲区旁路

数据旁路机制是发送 FIFO 控制块的一部分。它允许在数据流中引入一个数据字，而不需要修改发送 FIFO 缓冲区的内容，例如发送一个高优先级的消息。旁路结构由寄存器 BYP 中的一个旁路数据字 (最大为 16 位) 和寄存器 BYPCR 中的一些相关控制信息构成。例如，这些位定义旁路数据字的字长，配置与发送缓冲器 TBUF 类似的传送触发和门控机制。

旁路数据字可以由位 BYRCR.BDV (旁路数据有效) 标记为发送有效或无效。数据流相关和事件相关判据的组合定义旁路数据字是否被视为发送有效。数据验证逻辑检查这个数据字的起始条件。根据检查结果不同，发送缓冲区寄存器 TBUF 按下面的规则被加载为不同的值：

- 如果 TCSR.TDV = 0(TBUF 为空)，来自发送 FIFO 缓冲区的数据或旁路数据只能被传送到 TBUF 。
- 如果通过 BYPCR.BDEN 使能了旁路功能，或者满足选定的门控条件，则旁路数据只能被传送到 TBUF 中。
- 如果旁路数据为发送有效，并且具有比 FIFO 数据更高的发送优先级，或者如果发送 FIFO 为空，则旁路数据被传送到 TBUF。
- 如果用于传输旁路数据是有效的，且具有比包含有效数据的 FIFO 缓冲区更低的发送优先级，那么最早传输到 FIFO 的数据被传输到 TBUF 中。
- 如果旁路数据为发送有效，并且 FIFO 缓冲区包含有效数据，则最早的 FIFO 数据被传送到 TBUF。
- 如果旁路数据为发送无效，并且 FIFO 缓冲区也不包含有效数据，则 TBUF 保持不变。

旁路数据验证基于如**图 15-27** 所示的逻辑块。

- 发送门控逻辑在软件或硬件控制下使能或禁止旁路数据字传送到 TBUF。如果数据移位不需要输入级 DX2，则信号 DX2S 可用于门控目的。传送门控逻辑由位域 BYPCR.BDEN 控制。
- 传送触发逻辑支持与事件相关的数据字传送，例如基于定时器的事件或与一个输入引脚相关的事件。如果数据移位不需要输入级 DX2 ，则信号 DX2T 可用于触发目的。传送触发逻辑由位 BYPCR.BDVTR 控制。
- 旁路数据验证逻辑将来自门控逻辑、触发逻辑和 TCSR.TDV 的输入组合起来进行检查。

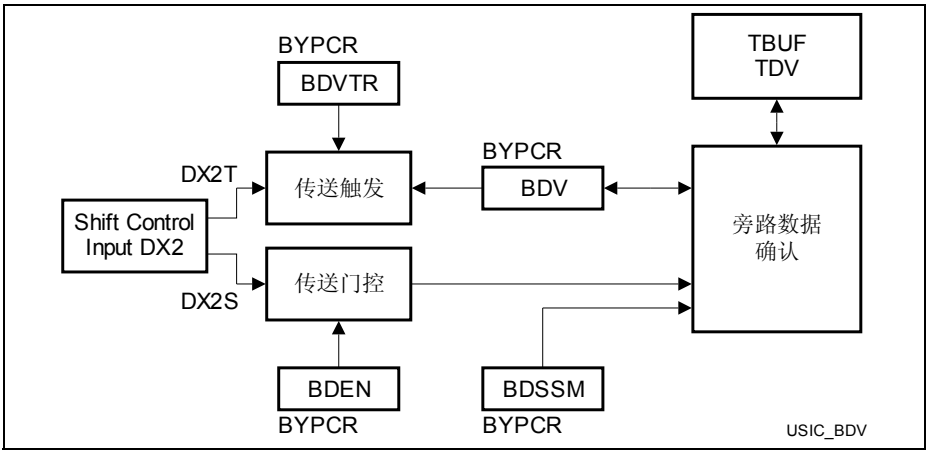


图 15-27　　旁路数据验证

利用这种结构，可实现以下旁路数据传送功能：

- 对于单次触发机制，必须编程为位 BYPCR.BDSSM= 1 。每次将旁路数据字传送到 TBUF 后，必须将旁路数据字再次标记为有效。这可以通过向 BYP 写入一个新的旁路数据字来实现；如果 BDVTR = 1，也可以通过 DX2T 来实现 (例如基于一个定时器或一个引脚上的边沿触发)
- 如果旁路数据为永久性发送有效，则必须编程为位 BYPCR.BDSSM = 0 (例如，如果数据 FIFO 为空，可作为备用数据)。

15.2.8.5　FIFO 访问限制

共享 FIFO 缓冲区中的数据由每个通信通道的数据传送硬件机制访问 (发送和接收)，由软件读出接收的数据或写入要发送的数据。因此，数据传输速率受 FIFO 机制限制。硬件对 FIFO 缓冲区的每次访问都优先于软件访问。在发生访问冲突的情况下，软件访问会被延迟。

为了避免因软件访问被延迟而引起 CPU 的数据丢失和阻塞，必须要考虑波特率、字长和软件访问机制。对 FIFO 数据缓冲区的每一次软件或硬件访问都需要一个 f_{PB} 周期。尤其是很短的连续流、连续数据字会导致访问限制。

15.2.8.6　FIFO 发送控制信息的处理

除了发送数据之外，还可以将发送控制信息 TCI 从发送 FIFO 或旁路结构传送到 USIC 通道。根据所选择的的协议和所使能的更新机制不同，一些 USIC 通道的参数设置可以修改。这种修改是通过将 FIFO 数据字的 TCI 加载到 TBUF 来实现的，如果旁路数据被加载到 TBUF 中，则由旁路控制信息修改。

- TCSR.SELMD = 1：由 FIFO TCI 或 BYPCR.BSELO 更新 PCR.CTR[20:16]，另外还清除 PCR.CTR[23:21]

- TCSR.WLEMD = 1：由 FIFO TCI 或 BYPCR.BWLE 更新 SCTR.WLE 和 TCSR.EOF (如果 WLE 信息被 TCI 或 BWLE 覆盖，则用户必须注意相应地设置 FLE)
- TCSR.FLEMD = 1：由 FIFO TCI 或 BYPCR.BWLE 更新 SCTR.FLE[4:0]，另外还清除 SCTR.FLE[5]
- TCSR.HPCMD = 1：由 FIFO TCI 或 BYPCR.BHPC 更新 SCTR.DSM 和 SCTR.HPCDI
- TCSR.WAMD = 1：由 FIFO TCI[4] 更新 TCSR.WA

有关 TCI 更详细的信息，见 **15.2.5.3 节**。

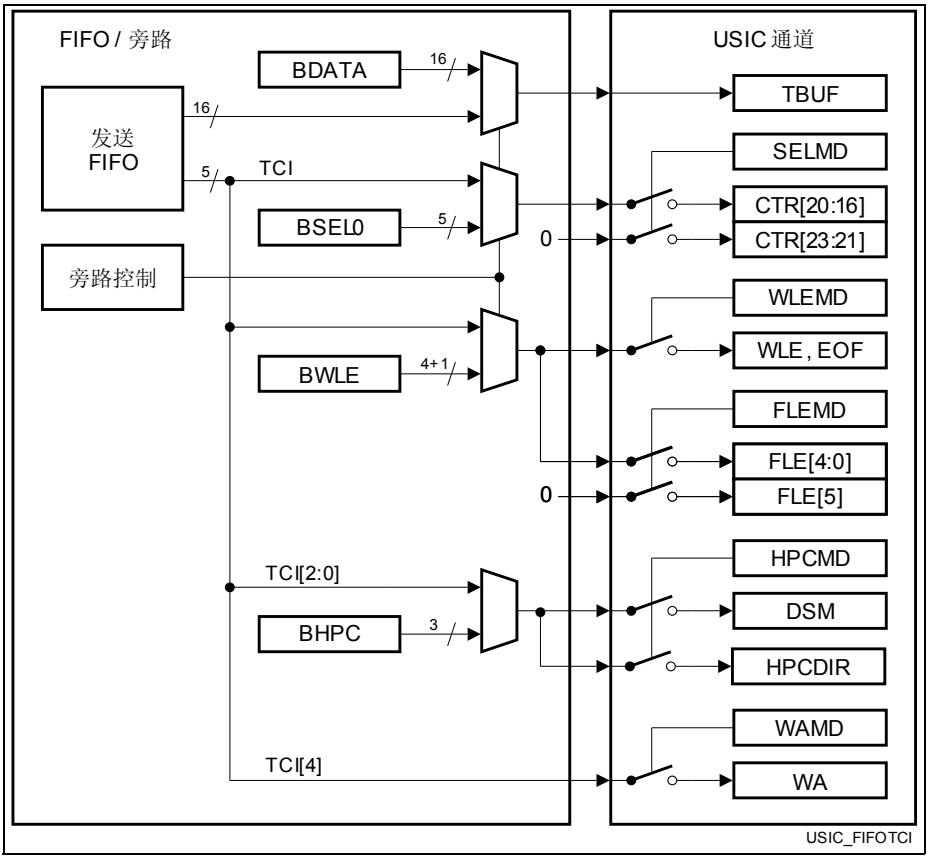


图 15-28　　使用 FIFO / 旁路结构的 TCI 处理

15.3 异步串行通道 (ASC = UART)

异步串行通道 ASC 处理异步数据帧的接收和发送，并提供硬件 LIN 支持。接收器和发送器是独立的，发送和接收数据帧可以在不同的时间点开始。通过设置 CCR.MODE = 0010_B 和 CCFG.ASC = 1 (ASC 模式有效) 来选择 ASC 模式。

15.3.1 信号说明

一个 ASC 连接的特征在于一个发送器和一个接收器之间只使用一个信号连接线。接收器输入信号 RXD 由输入级 DX0 处理。

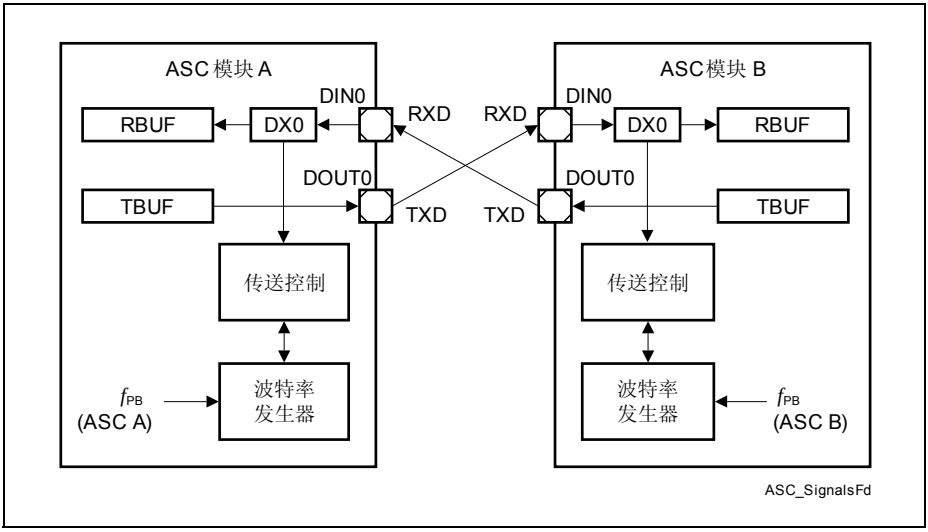


图 15-29 全双工通信的 ASC 信号连接

对于全双工通信，每个传输方向都需要一条独立的通信线。图 15-29 示出了在通信伙伴 ASC A 和 ASC B 之间采用点对点全双工通信的一个例子。

对于半双工或多发送器通信，在通信伙伴之间共享一条通信线。图 15-30 示出了在 ASC A 和 ASC B 之间采用点对点半双工连接的一个例子。在这种情况下，用户必须注意在某一时刻只能有一个发送器是活动的。为了支持发送器冲突检测，可使用输入级 DX1 来监视发送线的电平，并检查该线是处于空闲状态还是发生了冲突。

有两种将接收器的输入 DIN0 连接到发送器的输出 DOUT0 的可能方式。通信伙伴 ASC A 使用一个只具有发送引脚 TXD 的内部连接，TXD 提供其输入值作为 RXD 给 DX0 输入级用于接收，同时提供给 DX1 用于检查发送器冲突。通讯伙伴 ASC B 在两个引脚 TXD 和 RXD 之间使用一个外部连接。

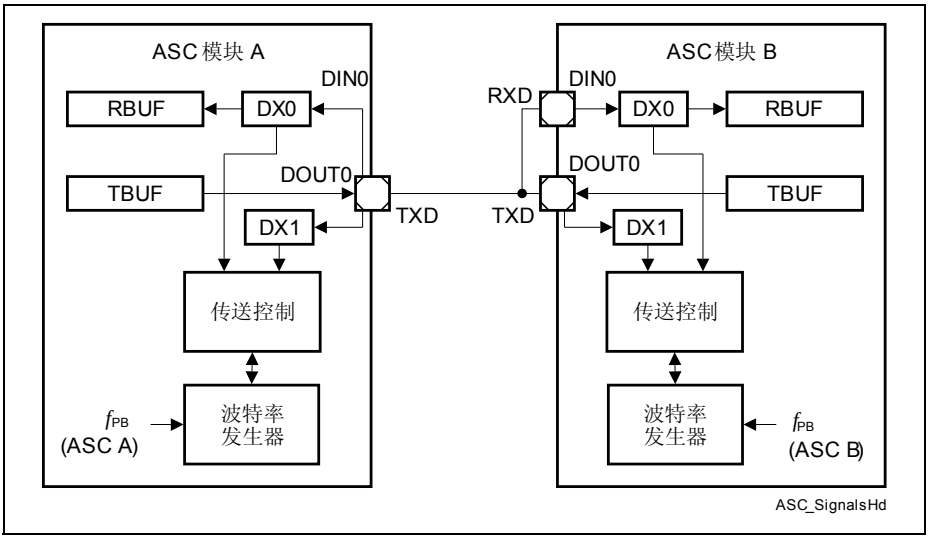


图 15-30　　半双工通信的 ASC 信号连接

15.3.2　帧格式

一个标准的 ASC 帧如图 **图 15-31** 所示。它包括：

- 具有信号电平 1 的空闲时间。
- 具有信号电平 0 的一个帧起始位 (SOF) 。
- 包含可编程数据位数 (1-63) 的数据域。
- 一个奇偶校验位 (P)，可编程为偶校验或奇校验。可以选择不使用校验位进行帧处理。
- 具有信号电平 1 的一个或两个停止位。

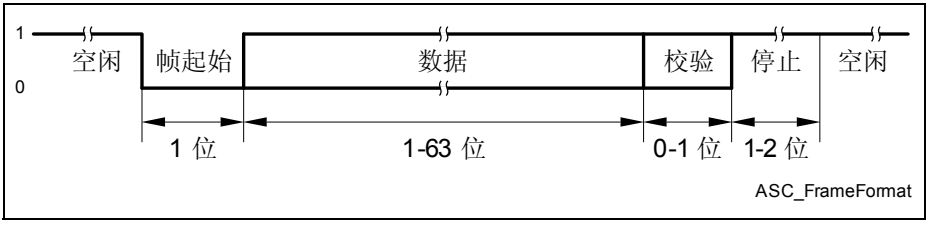


图 15-31　　标准的 ASC 帧格式

协议特有的位 (SOF, P, STOP) 由 ASC 协议状态机自动处理，不会出现在通过接收和发送缓冲区的数据流中。

15.3.2.1 空闲时间

接收器和发送器独立检查各自的数据输入线 (DX0, DX1) 是否闲置 。空闲检测确保一个最近使能的 ASC 模块的 SOF 位不会与另一 ASC 模块已经运行的帧相冲突。

为了启动空闲检测，用户软件必须清除位 PSR.RXIDLE 和 / 或 PSR.TXIDLE，例如在选择 ASC 模式之前或在运行过程中。当一次数据传送正在进行时，如果一个位被软件清除，那么当前正在运行的帧传送会正常结束，然后再启动空闲检测。仅当 PSR.RXIDLE = 1 时才可能进行帧接收；仅当 PSR.TXIDLE = 1 时才可能进行帧发送。空闲检测的持续时间取决于位 PCR.IDM 的设置。在不可能发生冲突的情况下，该持续时间可以缩短，甚至可以通过设置 PCR.IDM = 0，将总线声明为空闲。

在通过设置 PCR.IDM = 1 而使能了完整的空闲检测的情况下，如果检测到一定数量的连续被动位时间，则 DX0 的数据输入被视为空闲 (PSR.RXIDLE 变为置位状态)。同样的方案适用于发送器的 DX1 数据输入。在此，如果检测到该输入信号的空闲状态，则位 PSR.TXIDLE 变为置位状态。

完整的空闲检测的持续时间由所编程的每帧数据位数加 2 (在没有奇偶校验的情况下) 或加 3 (在有奇偶校验的情况下) 给出。在脱离停止模式或 ASC 模式被使能之后，从每次检测到一个边沿开始重新启动对具有 1 电平的连续位时间的计数。

如果空闲检测位 PSR.RXIDLE 和 / 或 TXIDLE 被软件清除，该计数过程不会停止 (不会从边沿开始处重新计数)。因此，如果相应的输入线仍然符合空闲标准，则被清除的位可以立即再次变成置位状态。

请注意，空闲时间检查是基于位时间的，因此，最大时间可以比编程值长 (但不能小于)1 位的时间。

15.3.2.2 起始位检测

接收器输入信号 DIN0 (输入级 DX0 的选定信号) 检查何时出现一个下降沿。当出现一个下降沿时，如果接收器空闲或该下降沿发生在最后一个停止位的采样点之后，则检测到一个 SOF 位。为了提高抗噪声能力，从检测到第一个下降沿开始对 SOF 位计时。如果采样的 SOF 位的值为 1，则认为前一下降沿是由噪声引起的，接收器再次视为空闲。

15.3.2.3 数据域

数据域的长度 (数据位数) 可由位域 SCTR.FLE 编程。它可以在 1 到 63 个数据位之间改变，对应的 SCTR.FLE 值为从 0 到 62 (值 63 被保留，不能在 ASC 模式使用这个编程值)。

数据域可包括多个数据字，例如 12 个数据位的一次传送可以分成两个 8 位字：将 12 个数据位分割成第一个字的 8 位和第二个字的 4 位。用户软件必须注意：一旦一个帧已开始发送，发送数据应及时可用。如果在一次正在运行的数据帧期间发送缓冲区为空，则发送被动数据电平 (SCTR.PDL) 。

移位方向可由 SCTR.SDIR 编程。LSB 在先的这种 ASC 帧标准设置由默认设置 SDIR = 0 实现。

15.3.2.4 奇偶校验位

ASC 允许基于帧为发送生成奇偶校验位和为接收检查奇偶校验位。奇偶校验的类型可由位域 CCR.PM 选择，对发送和接收是相同的 (无校验、偶校验或奇校验)。如果禁用奇偶校验处理，则 ASC 帧不包含任何奇偶校验位。出于一致性的原因，所有通信伙伴都必须被编程为相同的奇偶校验模式。

如果使能了奇偶校验位生成功能，发送器会在数据域的最后一个数据位之后自动发送其计算的校验位。接收器将该位解译为接收到的奇偶校验位，并将其与内部计算的奇偶位进行比较。接收到的奇偶位的值和奇偶校验的结果作为接收缓冲状态信息，在接收缓冲器状态寄存器 RBUFSR 和 RBUF01SR 中被监视。这些寄存器包含用于监视协议相关参数 (PAR) 的位和协议相关错误指示 (PERR) 的位。

15.3.2.5 停止位

每个 ASC 帧由 1 个 或 2 个具有信号电平 1 (与空闲电平相同的电平) 的停止位结束。停止位的数量由位 PSR.STPB 编程。可以在最后一个停止位之后直接开始传送一个新的起始位。

15.3.3 操作 ASC

为了操作 ASC 协议，必须考虑以下问题：

- 选择 ASC 模式：
  建议在 CCR.MODE = 0000_B 期间配置那些不能在运行时改变的所有 ASC 参数。必须将位域 SCTR.TRM 编程为 01_B。输入级的配置必须在 CCR.MODE = 0000_B 时完成，以避免意外的输入信号边沿，然后可以通过设置 CCR.MODE = 0010_B 来使能 ASC 模式。
- 引脚连接：
  通过设置 DX0CR.INSW = 0 建立输入级 DX0 与所选接收数据输入引脚 (信号 DIN0) 的连接，并配置一个发送数据输出引脚 (信号 DOUT0)。对于发送器的冲突或空闲检测，还必须通过设置 DX1CR.INSW = 0 将输入级 DX1 所选择的发送输出引脚连接。另外，还要编程 DX2CR.INSW = 0。
  由于同步协议处理器对输入数据流的处理，必须考虑在输入级产生的同步传播延迟。注意，使能备用输出端口功能的步骤只应在使能了 ASC 模式后完成，以避免在输出端出现意外的尖峰。
- 位时间配置：
  必须选择所期望的波特率设置，包括小数分频器、波特率发生器和位时间。请注意，并非所有功能组合都能在同一时刻应用程序支持，例如由于传播延迟的原因。例如，帧的长度受发送器和接收器器件频率差的限制。此外，为了使用样本的平均值 (SMD = 1)，采样点的选择必须考虑信号建立时间和数据传播时间。
- 数据格式配置：
  必须根据应用需要，通过对寄存器 SCTR 编程来设置字长、帧长和移位方向。如果应用需要，数据输入和输出信号可被反相。
  另外，还必须配置奇偶校验模式 (CCR.PM)。

15.3.3.1 位定时

在 ASC 模式，每一位 (包括协议位) 被划分成时间量子，这是为了提供子位范围内的时间粒度，以调节采样点来满足不同应用的需求。每位的时间量子数目由位域 BRG.DCTQ 和由 BRG.PCTQ 给出的一个时间量子的长度定义。

在图 15-32 给出的例子中，一个位定时由 16 个时间量子组成 (BRG.DCTQ = 15)。建议每个位时间被编程为不少于 4 个时间量子。

位域 PCR.SP 决定位值采样点的位置。 PCR.SP 的值不能被设置为大于 BRG.DCTQ 的值。可以在每个位时间仅采样一次位值，也可以取多个样本的平均值。根据位 PCR.SMD 的设置不同，可以将当前输入值直接采样作为位值，也可以对在最后三个时间量子处的输入采样值进行多数表决来确定位值。标准 ASC 位时间由 16 个时间量子构成，在第 8 个或 第 9 个时间量子之后进行采样并采用多数表决。

发送器和接收器的位定时设置 (时间量子的数目和采样点定义) 是统一的。由于有独立的位定时逻辑块，接收器和发送器在其帧内部可以位于不同的时间量子或位位置。一个帧的发送与时间量子的生成同步。

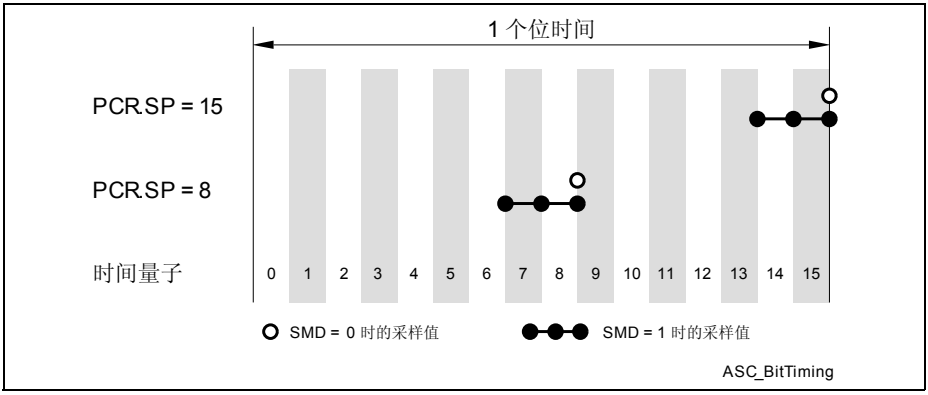


图 15-32 ASC 位定时

如果冲突或空闲检测被使能 (通过 DX1 输入信号)，则采样点的设置必须仔细调整，因为驱动器延迟和一些外部延迟必须加以考虑。发送线的采样点必须被设置为一个使位电平足够稳定的值，以进行评估。
如果采样点位于位时间的后期，那么信号本身有更多的时间变得稳定，但对发送器和接收器的时钟频率差的鲁棒性降低。

15.3.3.2 波特率发生器

ASC 模式下的波特率 f_{ASC} 取决于每个位时间的时间量子数及其时序。波特率的设置只应在发送器和接收器为空闲时改变。寄存器 BRG 中的位定义波特率的设置：

- BRG.CTQSEL
  定义用于时间量子产生的输入频率 f_{CTQIN}

- BRG.PCTQ
  定义时间量子的长度 (f_{CTQIN} 的 1、 2、 3 或 4 分频)
- BRG.DCTQ
  定义每个位时间的时间量子数

标准设置由 CTQSEL = 00_B ($f_{CTQIN} = f_{PDIV}$) 和 PPPEN = 0 ($f_{PPP} = f_{PIN}$) 给出。在这些条件下，波特率由下式给出：

$$f_{ASC} = f_{PIN} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \times \frac{1}{DCTQ + 1} \quad (15.6)$$

为了产生较低的频率，还可以通过设置 CTQSEL = 10_B ($f_{CTQIN} = f_{SCLK}$) 和 PPPEN = 1 ($f_{PPP} = f_{MCLK}$) 来选择两个额外的 2 分频级，这会导致：

$$f_{ASC} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \times \frac{1}{DCTQ + 1} \quad (15.7)$$

15.3.3.3 噪声检测

ASC 接收器一直检查 DX0 级数据输入线的噪声 (这种检查与位 PCR.SMD 的设置无关)。如果参加多数表决的三个输入样本在同一个采样点的位值不完全相同，则位 PSR.RNS (接收器噪声) 变成置位状态。位 PSR.RNS (它必须由软件清零) 中的接收器噪声信息在在多个位进行累加，如果由 PCR.RNIEN 使能，每当检测到噪声时可触发一个协议中断。

15.3.3.4 冲突检测

在有些应用中，例如在一条由多个发送设备共享的数据线上进行的数据传送 (见图 15-30)，多个发送器有可能在同一数据输出线 TXD 上发送数据。为了避免多个发送器在同一时刻处于活动状态而产生冲突，或允许一种仲裁， USIC 中实现了冲突检测功能。

在采样每一个位值之后，在 DX1 级的 TXD 输入读取的数据值与发送的数据位值进行比较。如果通过设置 PCR.CDEN = 1 使能了冲突检测，发送的位不等于读回的位，则检测到冲突并置位 PSR.COL。如果被使能，则位 PSR.COL = 1 会禁止发送器 (数据输出线变为 1) 并产生一个协议中断。发送移位寄存器的内容被视为无效，因此，发送缓冲器必须被重新编程。

15.3.3.5 脉冲整形

对于有些应用来说，位值为 0 的发送位的 0 电平不会在整个位时间都施加到发送输出。代替驱动原始 0 电平的是，只产生一个 0 脉冲，而该位时间的剩余时间量子由 1 电平驱动。一个位时间的长度并不因这种脉冲整形而改变，只有信令发生了改变。

在标准的 ASC 信令方案中，在具有位值 0 的完整位时间期间都发出 0 电平信号 (由编程设置 PCR.PL = 000_B 保证)。在 PCR.PL > 000_B 的情况下，发送输出信号在由 PCR.PL

定义的时间量子数期间变为 0。为了支持正确地接收经过发送器整形的脉冲，在接收器中必须根据所施加的脉冲宽度调整采样点。

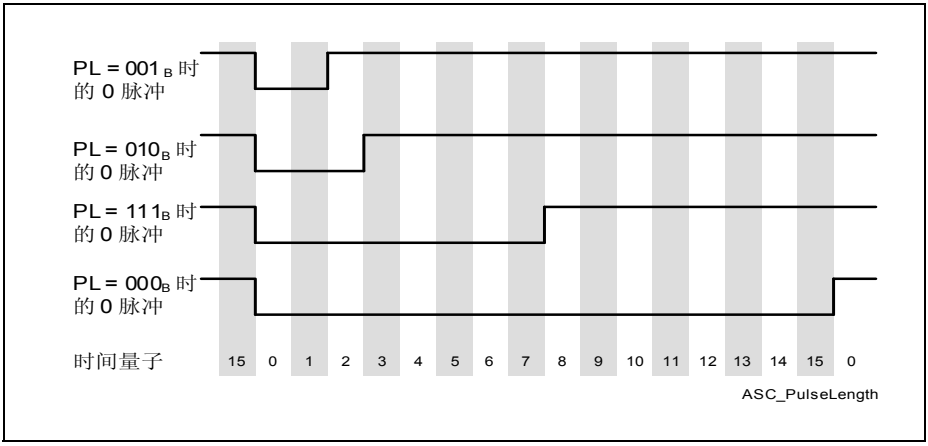


图 15-33　　发送器脉冲宽度控制

图 15-34 示出了一个发送 LSB 在先的 8 位数据字和一个停止位的例子 (例如 IrDA)。通过设置 SCTR.DOCFG = 01_B，发送输出信号的极性被反转。

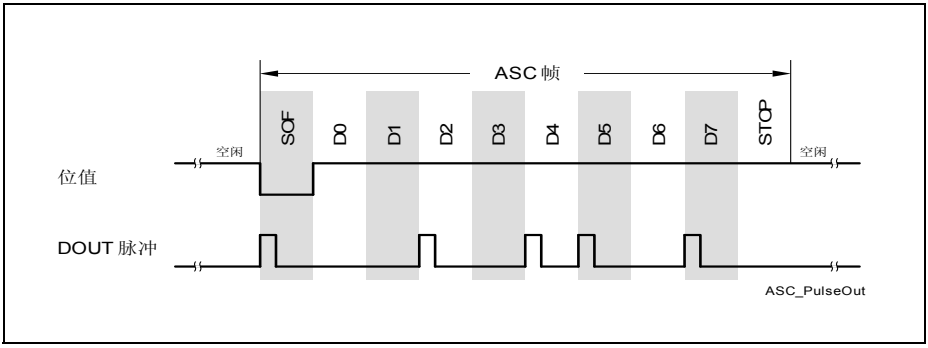


图 15-34　　脉冲输出示例

15.3.3.6　自动映射机制

当自动映射机制传送一个数据帧时 (映射发生在每一帧的开始)，协议控制寄存器 PCR 和位域 SCTR.FLE 中的内容在内部保持不变。可以在任何时间将这些寄存器编程为新的设置，这些新设置在下一个数据帧生效。在一个数据帧期间，虽然在数据帧开始后写入了新设置值，但所使用的 （映射的）设置并未改变。

位域 SCTR.WLE 和 SCTR.SDIR 在每个数据字的开始被自动映射。其结果是，一个数据

帧可由不同长度的数据字组成。建议只在没有数据帧正在运行时改变 SCTR.SDIR，以避免硬件和软件之间互相干扰。

请注意，一个数据字的起始点对于发送器和接收器可以不同。为了确保正确的处理，建议在发送器和接收器都空闲时修改 SCTR.WLE。如果发送器和接收器涉及相同的数据信号 (如在 LIN 总线系统中)，则在一次数据发送正在进行的过程中检测到 RSI 事件后，可以修改 SCTR.WLE。

15.3.3.7 帧结束控制

每个 ASC 帧的位数由位域 SCTR.FLE 定义。为了支持连续发送帧的不同帧长度设置，该位域可以被硬件修改。自动更新机制由 TCSR.FLEMD = 1 使能 (在这种情况下，位 TCSR.WLEMD、 SELMD、 WAMD 和 HPCMD 必须被写入 0 值)。

如果被使能，发送控制信息 TCI 会在 ASC 帧开始时自动覆盖位域 TCSR.FLEMD (导致具有 1 到 32 个数据位的帧)。TCI 的值代表所写的 TBUFxx (不使用额外的数据缓冲区) 或 INxx (使用额外的数据缓冲区) 地址单元。有了该机制，向 TBUF07 (IN07，分别) 写一个数据字即可产生具有 8 个数据位的一个 ASC 帧。

15.3.3.8 模式控制行为

ASC 模式支持以下内核模式：

- 运行模式 0/1：
  行为与编程设置的一样，对数据传送没有影响。
- 停止模式 0：
  位 PSR.TXIDLE 被清零。尚未启动一次新的发送。当前的发送正常完成。位 PSR.RXIDLE 未被修改。仍然可以接收。
  当脱离停止模式 0 时，位 TXIDLE 根据 PCR.IDM 设置。
- 停止模式 1：
  位 PSR.TXIDLE 被清零。尚未启动一次新的发送。当前的发送正常完成。PSR.RXIDLE 位被清零。不能进行一次新的接收。当前接收正常完成。
  当脱离停止模式 1 时，位 TXIDLE 和 RXIDLE 根据 PCR.IDM 设置。

15.3.3.9 禁止 ASC 模式

要在不破坏任何数据的情况下关闭 ASC 模式，接收器和发送器都必须是空闲的。通过在寄存器 KSCFG 中请求停止模式 1 可以保证这一点。在等待帧结束以后，ASC 模式被禁止。

15.3.3.10 协议中断事件

在 ASC 模式可产生下述协议相关的事件，这些事件可导致产生一个协议中断。冲突检测和发送器帧结束事件与发送器相关，而接收器事件由同步停止检测、接收器噪声检测、格式错误检查和接收帧结束给出。

请注意，寄存器 PSR 中的位不能由硬件自动清零，必须用软件清零，这是为了监视新到来的事件。

- 冲突检测：
  该中断指示在一个位的采样点，所发送的值 (DOUT0) 与 DX1 输入级的输入值不匹配。更详细的信息请参见 页 15-50。
- 发送器帧结束：
  该中断指示发送器已经完全发送完一帧。位 PSR.TFF 在最后一个停止位结束时变成置位状态。当没有发送正在进行时，可以改变 DOUT0 信号的端口引脚分配。
- 接收器帧结束：
  该中断指示接收器已经完全接收完一帧。位 PSR.RFF 在最后一个停止位结束时变成置位状态。当没有接收正在进行时，可以改变 DIN0 信号的端口引脚分配。
- 同步间断检测：
  该中断可在 LIN 网络中使用，以指示收到同步间断符号 (在一个 LIN 帧的开始)。
- 接收器噪声检测：
  该中断指示在一个位的采样点的输入值与在前面两个时间量子的采样值不完全相同。
- 格式错误：
  对于 ASC 协议，停止位的位值被定义为 1 电平。如果一个停止位的采样位值是 0，则会报告一个格式错误。

15.3.3.11 数据传送中断处理

数据传送中断指示与 ASC 帧处理相关的事件。

- 发送缓冲区中断 TBI：
  在一个数据字的第一个数据位开始后，位 PSR.TBIF 被置位。这是可以向 TBUF 写入一个新数据字的最早时间点。
  发生该事件时，位 TCSR.TDV 被清零，并且新数据可以被加载到发送缓冲区。
- 发送移位中断 TSI：
  在一个数据字的最后一个数据位开始后，位 PSR.TSIF 被置位。
- 接收器开始中断 RSI：
  在一个数据字的第一个数据位的采样点之后，位 PSR.RSIF 被置位。
- 接收器中断 RI 和备用中断 AI：
  如果一个数据字之后不直接跟随一个奇偶校验位 (校验位产生被禁止或不是一个数据帧的最后一个字)，那么在该数据字的最后一个数据位的采样点之后，位 PSR.RIF 被置位。
  如果数据字后面直接跟随一个奇偶校验位 (一个数据帧的最后一个数据字并且校验位产生被使能)，则在奇偶校验位的采样点之后，如果没有检测到奇偶校验错误，位 PSR.RIF 被置位。如果检测到一个奇偶校验错误，则位 PSR.AIF 被置位而不是位 PSR.RIF。
  对于接收的字，一个数据帧的第一个数据字由 RBUFSR.SOF = 1 指示。
  在 WA = 0 时发生接收器中断 RI，位 PSR.RIF 被置位。在 WA = 1 时发生备用中断 AI，位 PSR.AIF 被置位。

15.3.3.12 波特率发生器中断处理

波特率发生器中断指示捕获模式定时器已经达到其最大值。 发生该事件时，位 PSR.BRGIF 被置位。

15.3.3.13 协议相关的参数和错误

协议相关参数 (RBUFSR.PAR) 和协议相关错误 (RBUFSR.PERR) 是两个标志，它们位于相应的接收缓冲器状态寄存器中，被分配给每一个接收数据字。

在 ASC 模式，接收到的奇偶校验位由协议相关的参数监视，奇偶校验的结果由协议相关错误指示(0 = 接收的奇偶校验位等于计算的校验值)。该信息仅用于阐释每个数据帧中最后接收的数据字，如果数据字不是一个数据帧的最后数据字，或者奇偶校验位生成被禁止，则这两位都为 0。

15.3.3.14 接收缓冲区处理

在 ASC 模式，如果有一个接收 FIFO 缓冲区可用 (CCFG.RB = 1) 并被使能用于数据处理 (RBCTR.SIZE > 0)，则建议设置 RBCTR.RCIM = 11_B。这会导致：位 OUTR.RCI[0] = 1 指示该数据字是一个新数据帧的第一个数据字； OUTR.RCI[4] = 1 指示发生了一个奇偶校验错误，接收的校验位值由 OUTR.RCI[3] 给出。

在RCI模式 (RBCTR.RNM = 1)，标准接收缓冲区事件和备用接收缓冲区事件可以用于执行以下操作：

- 一个标准接收缓冲区事件指示可以从 OUTR 读取一个数据字，这个已接收的数据字无奇偶校验错误。
- 一个备用接收缓冲区事件指示可以从 OUTR 读取一个数据字，这个已接收的数据字有奇偶校验错误。

15.3.3.15 同步间断检测

接收器持续检查 DIN0 信号为 0 电平的连续位时间的数量。该数量由编程设置的每帧的位数加 2 (无校验位的情况) 或加 3 (有校验位的情况) 给出。如果在事件发生后在一个位的采样点检测到 0 电平，则位 PSR.SBD 被置位，另外还可以产生一个协议中断 (如果由 PCR.SBD = 1 使能)。每当在在输入 DIN0 发现一个下降沿时，则重新从 0 开始计数。该功能可以用于在一个 LIN 总线系统中检测从机的同步间断 (主机不检测同步间断)。
例如，在一个使用 8 个数据位且不使用奇偶校验位的配置中，如果在 10 个完整位时间过后的下一个采样点 (即从第一个下降沿算起的第 11 个位时间的采样点) 检测到 0 电平，则位 PCR.SBD 被置位。

15.3.3.16 传送状态指示

如果位 PCR.CTR[16] (接收器状态使能 RSTEN) 被置位，那么接收器的状态可以由标志 PSR[9] = BUSY 监视。在这种情况下，位 BUSY 在整个帧接收期间被置 1，这段时间从帧起始位开始到最后一个停止位结束。
如果位 PCR.CTR[17](发送器状态使能 TSTEN) 被置位，那么发送器的状态可以由标志

PSR[9] = BUSY 监视。在这种情况下，位 BUSY 在整个帧发送期间被置 1，这段时间从帧起始位开始到最后一个停止位结束。
如果两个位 RSTEN 和 TSTEN 都被置 1，则标志 BUSY 指示接收器和发送器状态的逻辑 “ 或 ” 组合。如果两个位都被清零，标志 BUSY 并不根据传送状态而修改 (状态改变被忽略)。

15.3.4 ASC 协议寄存器

在 ASC 模式，寄存器 PCR 和 PSR 处理 ASC 的相关信息。

15.3.4.1 ASC 协议控制寄存器

在 ASC 模式， PCR 寄存器中的位或位域依本节所述定义。

PCR
协议控制寄存器 [ASC 模式]　　($3C_H$)　　初始值：$0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCLK							0							TSTEN	RSTEN
rw							r							rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL			SP					FFIEN	FEIEN	RNIEN	CDEN	SBIEN	IDM	STPB	SMD
rw			rw					rw	rw	rw	rw	rw	rw	rw	rw

域	位	类型	描述
SMD	0	rw	**采样模式** 该位域定义 ASC 接收器的采样模式。所选择的的数据输入信号可在每个位时间被采样一次或三次 (在连续的时间量子处)。当采样三次时，移入到接收移位寄存器中的位值由三次采样值经多数表决给出。 0_B　每个位时间只取一个样本。当前的输入值被采样。 1_B　每个位时间取三个样本，然后进行多数表决。
STPB	1	rw	**停止位** 该位定义一个 ASC 帧中的停止位数目。 0_B　停止位数目为 1。 1_B　停止位数目为 2。

域	位	类型	描述
IDM	2	rw	**空闲检测模式** 该位定义是关闭空闲检测，还是基于帧长度进行检测。 0_B 总线空闲检测被关闭，位 PSR.TXIDLE 和 PSR.RXIDLE 被自动置位，以使能不检查输入即进行数据发送。 1_B 在经过由 SCTR.FLE 加 2 (无奇偶校验位的情况) 或加 3 (有奇偶校验位的情况) 定义的一定数量的连续被动位时间后，总线被视为空闲。
SBIEN	3	rw	**同步间断中断使能** 如果检测到一个同步间断，则该位使能一个协议中断的产生。自动检测总是处于活动状态，因此位 SBD 可以独立于 SBIEN 被置位。 0_B 中断产生被禁止。 1_B 中断产生被使能。
CDEN	4	rw	**冲突检测使能** 该位使能一个发送器对冲突检测的反应。 0_B 冲突检测被禁止。 1_B 如果检测到有冲突，则发送器停止其数据发送，输出一个 1 电平，置位 PSR.COL 并产生一个协议中断。 为了允许再次发送数据， PSR.COL 必须由软件清零。
RNIEN	5	rw	**接收器噪声检测中断使能** 如果检测到接收器噪声，则该位使能一个协议中断的产生。自动检测总是处于活动状态，因此位 PSR.RNS 可以独立于 PCR.RNIEN 被置位。 0_B 中断产生被禁止。 1_B 中断产生被使能。
FEIEN	6	rw	**格式错误中断使能** 如果检测到一个格式错误，则该位使能一个协议中断的产生。自动检测总是处于活动状态，因此位 PSR.FER0/FER1 可以独立于 PCR.FEIEN 被置位。 0_B 中断产生被禁止。 1_B 中断产生被使能。

域	位	类型	描述
FFIEN	7	rw	**帧结束中断使能** 如果接收器或发送器达到一帧的结束，则该位使能一个协议中断的产生。自动检测总是处于活动状态，因此位 PSR.RFF 或 PSR.TFF 可以独立于 PCR.FFIEN 被置位。 0_B 中断产生被禁止。 1_B 中断产生被使能。
SP	[12:8]	rw	**采样点** 该位域定义位值的采样点。采样点不能位于所编程的位时间外部 (PCR.SP ≤ BRG.DCTQ)。
PL	[15:13]	rw	**脉冲宽度** 该位域定义一个 0 数据位的宽度，以时间量子计算，从每个位时间的时间量子 0 开始。每个为 0 的位值可以导致一个比一个位时间短的 0 脉冲，例如用于 IrDA 应用。 PL 不能改变一个位时间的长度，它只能改变输出信号的 0 电平长度。 脉冲宽度不能大于所编程的位时间 (PCR.PL ≤ BRG.DCTQ)。该位域只对发送器有效，被接收器忽略。 000_B 脉冲宽度等于位长度 (不能短到 0)。 001_B 一个 0 位的脉冲长度为 2 个时间量子。 010_B 一个 0 位的脉冲长度为 3 个时间量子。 ... 111_B 一个 0 位的脉冲长度为 8 个时间量子。
RSTEN	16	rw	**接收器状态使能** 该位使能对标志 PSR[9] = BUSY 的修改，根据接收器的状态。 0_B 标志 PSR[9] 并不因接收器的状态而改变。 1_B 标志 PSR[9] 在一个完整的帧接收期间被置位。
TSTEN	17	rw	**发送器状态使能** 该位使能对标志 PSR[9] = BUSY 的修改，根据发送器状态。 0_B 标志 PSR[9] 并不因发送器的状态而改变。 1_B 标志 PSR[9] 在一个完整的帧发送期间被置位。
MCLK	31	rw	**主时钟使能** 该位使能主时钟 MCLK 的产生。 0_B MCLK 产生被禁止，且 MCLK 信号为 0。 1_B MCLK 产生被使能。
0	[30:18]	r	**保留** 读访问返回 0 ；应写入 0 。

15.3.4.2 ASC 协议状态寄存器

在 ASC 模式，PSR 寄存器的位或位域依本节所述定义。寄存器 PSR 中的位和位域不能由硬件清零。

PSR 寄存器中的标志可以通过向寄存器 PSCR 中的对应位写 1 来清除。向 PSR 中的一个位写 1 会置位对应的标志，但不会导致进一步的行为 (不产生中断)。写 0 没有影响。应在使能一个新协议之前用软件清除 PSR 中的标志。

PSR

协议状态寄存器 [ASC 模式]　　(48$_H$)　　复位值：0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															BRGIF
r															rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF	BUSY	TFF	RFF	FER1	FER0	RNS	COL	SBD	RXIDLE	TXIDLE
rwh	rwh	rwh	rwh	rwh	rwh	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
TXIDLE	0	rwh	**发送空闲** 该位指示发送线 (DX1) 是否为空闲。一次帧发送只能在 TXIDLE 被置位的情况下开始。 0_B　发送线尚未空闲。 1_B　发送线是空闲的，可以进行帧发送。
RXIDLE	1	rwh	**接收空闲** 该位指示接收线 (DX0) 是否为空闲。一次帧接收只能在 RXIDLE 被置位的情况下开始。 0_B　接收线尚未空闲。 1_B　接收线是空闲的，可以进行帧接收。
SBD	2	rwh	**同步间断检测**[1)] 如果检测到所编程数量的连续 0 电平位值，则该位被置 1(称为同步间断，例如在一个 LIN 总线系统中)。 0_B　尚未检测到一个同步间断。 1_B　已经检测到一个同步间断。

域	位	类型	描述
COL	3	rwh	**冲突检测 1)** 如果检测到一个冲突 (PCR.CDEN = 1)，则该位被置 1。 0_B 尚未检测到冲突，可以进行帧发送。 1_B 已经检测到冲突，不能进行帧发送。
RNS	4	rwh	**接收器噪声检测 1)** 如果检测到接收器噪声，则该位被置 1。 0_B 尚未检测到接收器噪声。 1_B 已经检测到接收器噪声。
FER0	5	rwh	**停止位 0 格式错误 1)** 如果在停止位 0 采样到 0 电平，则该位被置 1(称为格式错误 0)。 0_B 尚未检测到一个格式错误 0。 1_B 已经检测到一个格式错误 0。
FER1	6	rwh	**在停止位 1 中的格式错误 1)** 如果在停止位 1 采样到 0 电平，则该位被置 1(称为格式错误 1)。 0_B 尚未检测到一个格式错误 1。 1_B 已经检测到一个格式错误 1。
RFF	7	rwh	**接收帧完成 1)** 如果接收器已经完成了最后一个停止位的接收，该位被置 1。 0_B 接收帧尚未完成。 1_B 接收帧已经完成。
TFF	8	rwh	**发送帧完成 1)** 如果发送器已经完成了最后一个停止位的发送，该位被置 1。 0_B 发送帧尚未完成。 1_B 发送帧已经完成。
BUSY	9	r	**传送状态忙** 该位指示了接收器的状态 (如果 PCR.RSTEN = 1) 或发送器的状态 (如果 PCR.TSTEN = 1) 或两者的逻辑或组合 (如果 PCR.RSTEN = PCR.TSTEN = 1)。 0_B 未发生数据发送。 1_B 数据传送正在进行。
RSIF	10	rwh	**接收器启动指示标志** 0_B 未发生接收器启动事件。 1_B 已发生接收器启动事件。

域	位	类型	描述
DLIF	11	rwh	**数据丢失指示标志** 0_B 未发生数据丢失事件。 1_B 已发生数据丢失事件。
TSIF	12	rwh	**发送移位指示标志** 0_B 未发生发送移位事件。 1_B 已发生发送移位事件。
TBIF	13	rwh	**发送缓冲区指示标志** 0_B 未发生发送缓冲事件。 1_B 已发生发送缓冲事件。
RIF	14	rwh	**接收指示标志** 0_B 未发生接收事件。 1_B 已发生接收事件。
AIF	15	rwh	**备用接收指示标志** 0_B 未发生选择接收事件。 1_B 已发生选择接收事件。
BRGIF	16	rwh	**波特率发生器指示标志** 0_B 未发生波特率发生器事件。 1_B 已发生波特率发生器事件。
0	[31:17]	r	**保留** 读访问返回 0；应写入 0。

1) 页 15-115 (见 页 15-17)。通用中断状态标志在通用中断一章中描述。

15.3.5 硬件 LIN 支持

为了支持 LIN 协议，对于主设备来说应设置位 TCSR.FLEMD = 1。对于从设备来说，该位可以被清零，并且应将数据位数设置为固定的 8 个数据位 (SCTR.FLE = 7_H)。主设备和从设备两者的奇偶校验位生成器应该被关闭 (CCR.PM = 00_B)，数据传送采用 LSB 在先 (SCTR.SDIR = 0) 和 1 个停止位 (PCR.STPB = 0)。

本地互连网络 (LIN) 数据交换协议包含多个在 ASC 模式可以被全部处理的符号。每个单一的 LIN 符号代表一个完整的 ASC 帧。LIN 总线是一个具有单一主机和多个从机的主从总线系统 (确切的定义请参见官方 LIN 规范)。

一个完整的 LIN 帧包含以下符号：

- 同步间断
  主机在一个新帧的开始发送一个同步间断信号。它包含至少 13 个连续位时间的 0 电平，然后是至少一个位时间的 1 电平 (对应 1 个停止位)。因此，如果使用发送缓冲区，则必须向 TBUF11(或 IN11，如果使用 FIFO 缓冲区) 写入 0(导致一个以 SOF 开始并跟随 12 个 0 电平数据位的 帧)。
  从设备应通过同步间断检测功能来检测到 11 个连续位时间的 0 电平。如果检测到这

样的一个事件，则位 PSR.SBD 置 1，并且产生一个协议中断。此外，在接收缓冲区会出现值为 0 的接收数据，并且会报告一个格式错误。
如果从机的波特率必须适应主机，则在下一个符号开始之前，必须通过设置 BRG.TMEN = 1、 DX0CR.CM = 10_H 和 DX1CR.CM = 00_H 来使能基于下降沿的波特率测量。

- 同步字节：
  主机在向 TBUF07 (或 IN07) 写入数据值 55_H 后发送该符号。
  从设备可以在接收该符号后不进行任何进一步的操作 (并且可以将其丢弃) ，也可以使用下降沿进行波特率测量。位 PSR.TSIF = 1 (也可以选择产生相应的中断) 指示检测到一个下降沿，并且将自最后一个下降沿以来流逝的时间捕获到 CMTR.CTV 中。有效的捕获值可以在第二、第三、第四和第五次激活 TSIF 后读出。在该符号内第五次激活 TSIF 后，波特率检测可以被禁止 (BRG.TMEN = 0)， BRG.PDIV 可以编程为捕获的 CMTR.CTV 值除以每位所占时间量子数的两倍 (假设 BRG.PCTQ = 00_B)。
- 其他符号：
  一个 LIN 帧的其他符号可以按 ASC 数据帧处理而不进行特有操作。

如果 LIN 帧应由 LIN 主机基于帧来发送，可以将输入 DX2 连接到外部的定时器以触发发送动作 (例如同步间断符号已经准备好，如果发生触发则启动发送)。请注意，在 ASC 接收器进行波特率测量期间，同一 USIC 通道的 ASC 发送器仍然可以进行发送。

15.4 同步串行通道 (SSC)

同步串行通道 SSC 涵盖一个类 SPI 模块的数据传送功能。它可以处理一个工作在主模式的器件与至少一个工作在从模式的器件之间同步数据帧的接收和发送。除了使用一个输入和一个输出数据线的标准 SSC 协议之外，同步串行通道还支持使用两个 (双位 SSC) 或四个 (四位 SSC) 输入 / 输出数据线的 SSC 协议。SSC 模式由 CCR.MODE = 0001_B 和 CCFG.SSC = 1 选择 (SSC 模式有效)。

15.4.1 信号说明

同步 SSC 数据发送器的特征在于，移位时钟信号与发送与 / 或接收数据信号一起同时发送，以决定数据何时有效 (发送和采样点的定义)。

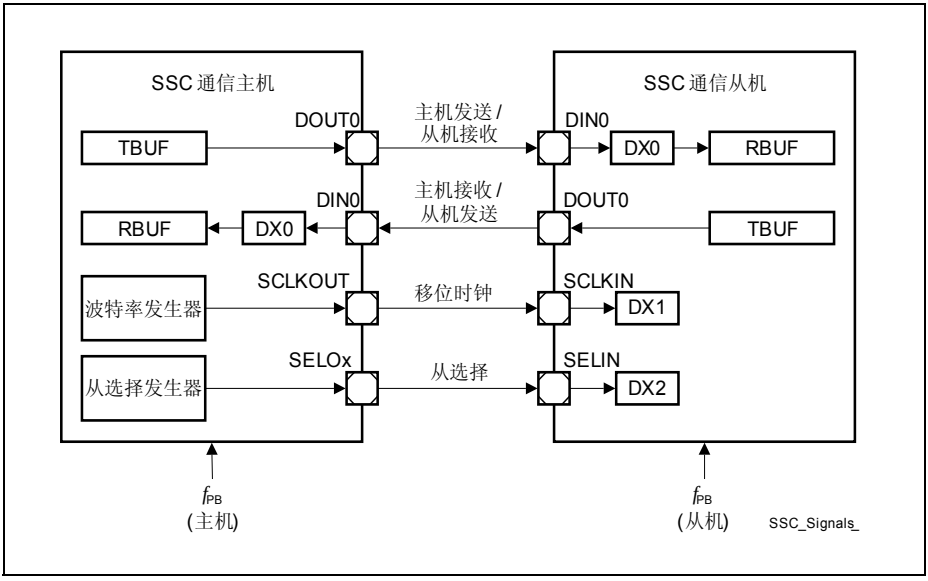


图 15-35 标准全双工通信的 SSC 信号

为了显性指示一次数据传送的开始和结束以及单独访问一个以上的从器件，SSC 模块支持从选择信号的处理。从选择信号对于 SSC 数据传送来说是可选的，不一定需要。SSC 模块工作在主模式时支持最多 8 个不同的从选择输出信号 (命名为 SELOx，其中 x = 0-7)，工作在从模式时支持 1 个从选择输入 SELIN 。在大多数应用中，从选择信号为低电平有效。

工作在主模式下的器件控制一个数据帧的开始和结束，以及移位时钟和从选择信号的产生。这包括移位时钟的波特率设置和移位时钟与从选择输出信号之间的延迟。如果多个 SSC 模块连接在一起，在某一个时刻只能有一个 SSC 主机，但可以有多个从机。从器件接收移位时钟和可选的从选择信号。有关输入级 DXn 的编程，请参见 页 15-18。

表 15-11 SSC 通信信号

SSC 模式	接收数据	发送数据	移位时钟	从选择
标准 SSC 主机	MRST[1)]，输入 DIN0，由 DX0 处理	MTSR[2)]，输出 DOUT0	输出 SCLKOUT	输出 SELOx
标准 SSC 从机	MTSR，输入 DIN0，由 DX0 处理	MRST，输出 DOUT0	输入 SCLKIN，由 DX1 处理	输入 SELIN，由 DX2 处理
双位 SSC 主机	MRST[1:0]，输入 DIN[1:0]，由 DX0 和 DX3 处理	MTSR[1:0]，输出 DOUT[1:0]	输出 SCLKOUT	输出 SELOx
双位 SSC 从机	MTSR[1:0]，输入 DIN[1:0]，由 DX0 和 DX3 处理	MRST[1:0]，输出 DOUT[1:0]	输入 SCLKIN，由 DX1 处理	输入 SELIN，由 DX2 处理
四位 SSC 主机	MRST[3:0]，输入 DIN[3:0]，由 DX0、DX3、DX4 和 DX5 处理	MTSR[3:0]，输出 DOUT[3:0]	输出 SCLKOUT	输出 SELOx
四位 SSC 从机	MTSR[3:0]，输入 DIN[3:0]，由 DX0、DX3、DX4 和 DX5 处理	MRST[3:0]，输出 DOUT[3:0]	输入 SCLKIN，由 DX1 处理	输入 SELIN，由 DX2 处理

1) MRST = 主机接收从机发送，也称为 MISO = 主入从出

2) MTSR = 主机发送从机接收，也称为 MOSI = 主出从入

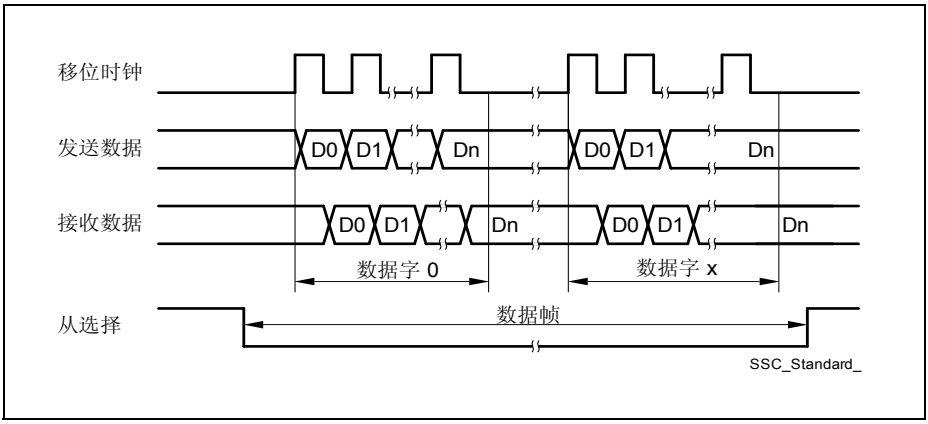


图 15-36　　4 线 SSC 标准通信信号

15.4.1.1　发送和接收数据信号

在标准的 SSC 半双工模式，不论是从主机到从机的数据传送还是自从机到主机的数据传送都只使用同一根数据线。在这种情况下，MRST 和 MTSR 被连接在一起，根据数据传送方向，一个信号作为输入，另一信号作为输出。用户软件必须注意数据的方向，以避免发生数据冲突 (例如，在将漏极开路驱动器通过线与方式连接在一起的情况下，通过准备全 1 的虚数据用于发送即可实现这一目标。驱动器的线与连接可以通过使能 / 禁止推挽输出驱动器来实现，也可以在使能了硬件端口控制的的情况下通过切换引脚方向来实现)。在全双工模式，数据传送在一个主器件和一个从器件之间并行进行，使用两个独立的数据信号 MTSR 和 MRST，如图**图 15-35** 所示。

接收数据输入信号 DIN0 由输入级 DX0 处理。在主模式 (指 MRST) 和从模式 (指 MTSR)，数据输入信号 DIN0 都是从一个输入引脚输入。DOUT0 (数据输出信号) 信号相对于数据位值的极性可以在块 DOCFG (数据输出配置) 中通过位域 SCTR.DOCFG 配置。

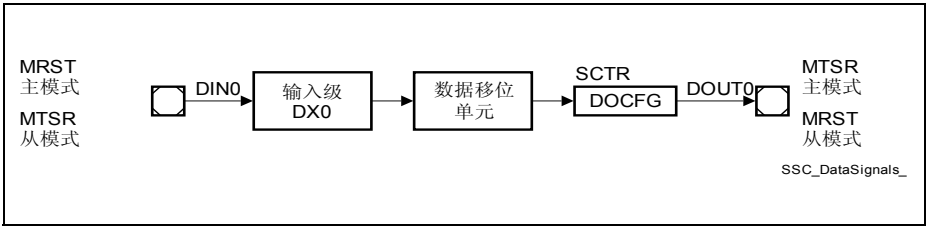


图 15-37　　SSC 数据信号

对于需要使用多个输入和输出数据线以及额外输入级的双位和四位 SSC 模式，需要设置 DINx 和 DOUTx 信号。

15.4.1.2 移位时钟信号

移位时钟信号由输入级 DX1 处理。在从模式，SSC 接收来自外部主机的信号 SCLKIN，因此 DX1 输入级必须被连接到一个输入引脚。该输入级可以将接收到的输入信号反相，以适应用于数据移位单元功能的 SCLKIN 信号的极性 (在上升沿发送数据，在下降沿接收数据)。

在主模式，移位时钟由内部波特率发生器产生。波特率生成器的输出信号 SCLK 被用作数据移位单元的移位时钟输入。内部信号 SCLK 通过信号 SCLKOUT 提供给外部从器件使用。对于从模式下的完全闭环延迟补偿，SCLKOUT 也可以从输入级 DX1 获得发送移位时钟。这种选择通过位 BRG.SCLKOSEL 完成。见 **15.4.6.3 节**。

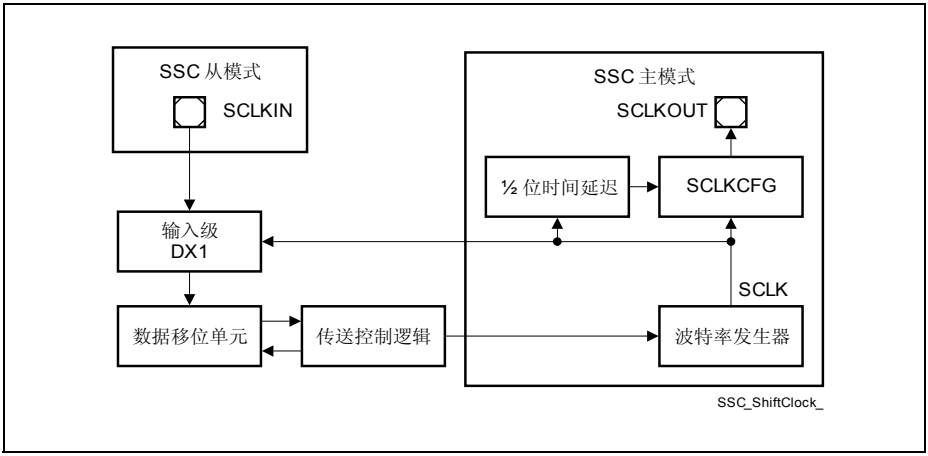


图 15-38 SSC 移位时钟信号

由于存在众多不同的 SSC 应用，所以在主模式下可使用多种不同的方式来配置相对于 SCLK 的移位时钟输出信号 SCLKOUT。这可以在块 SCLKCFG(移位时钟配置) 中通过位域 BRG.SCLKCFG 来完成，可有 4 种可能的设置，如图 **图 15-39** 所示。

- 无延迟，无极性反转 (SCLKCFG = 00_B，SCLKOUT 等于 SCLK)：
  在没有数据帧传送时，SCLKOUT 的无效电平为 0。一个新数据帧的第一个数据位在 SCLKOUT 的第一个上升沿发送，在 SCLKOUT 的第一个下降沿接收第一个数据位。一个数据帧的最后一个数据位在 SCLKOUT 的最后一个上升沿发送，最后一个数据位在 SCLKOUT 的最后一个下降沿接收。该设置可用于主模式和从模式。它符合内部数据移位单元的行为。
- 无延迟，极性反转 (SCLKCFG = 01_B)：
  在没有数据帧传送时，SCLKOUT 的无效电平为 1。一个新数据帧的第一个数据位在 SCLKOUT 的第一个下降沿发送，在 SCLKOUT 的第一个上升沿接收第一个数据位。一个数据帧的最后一个数据位在 SCLKOUT 的最后一个下降沿发送，最后一个数据位在 SCLKOUT 的最后一个上升沿接收。该设置可用于主模式和从模式。

- SCLKOUT 被延迟 1/2 个移位时钟周期，无极性反转 (SCLKCFG = 10_B)：
  在没有数据帧传送时， SCLKOUT 的无效电平为 0。一个新数据帧的第一个数据位在 SCLKOUT 的第一个上升沿之前 1/2 个 移位时钟周期发送。由于该延迟的存在，下一个数据位似乎在 SCLKOUT 的下降沿发送。一个数据帧的最后一个数据位在 SCLKOUT 的最后一个上升沿之前 1/2 个 SCLKOUT 周期发送。第一个数据位在 SCLKOUT 的第一个下降沿之前 1/2 个移位时钟周期接收。由于该延迟的存在，下一个数据位似乎在 SCLKOUT 的上升沿接收。最后一个数据位在 SCLKOUT 的最后一个下降沿之前 1/2 个 SCLKOUT 周期接收。
  该设置只能用于主模式而不能用于从模式 (所连接的从机必须在第一个 SCLKOUT 边沿之前提供第一个数据位。例如，一旦被其从选择输入寻址为从机，即准备第一个数据位)。
- SCLKOUT 被延迟 1/2 个移位时钟周期，极性反转 (SCLKCFG = 11_B)：
  在没有数据帧传送时， SCLKOUT 的无效电平为 1。
  一个新数据帧的第一个数据位在 SCLKOUT 的第一个下降沿之前 1/2 个移位时钟周期发送。由于该延迟的存在，下一个数据位似乎在 SCLKOUT 的上升沿发送。一个 数据帧的最后一个数据位在 SCLKOUT 的最后一个下降沿之前 1/2 个 SCLKOUT 周期发送。第一个数据位在 SCLKOUT 的第一个上升沿之前 1/2 个 移位时钟周期接收。由于该延迟的存在，下一个数据位似乎在 SCLKOUT 的下降沿接收。最后一个数据位在 SCLKOUT 的最后一个上升沿之前 1/2 个 SCLKOUT 周期接收。
  该设置只能用于主模式而不能用于从模式 (所连接的从机必须在第一个 SCLKOUT 边沿之前提供第一个数据位。例如，一旦被其从选择输入寻址为从机，即准备第一个数据位)。

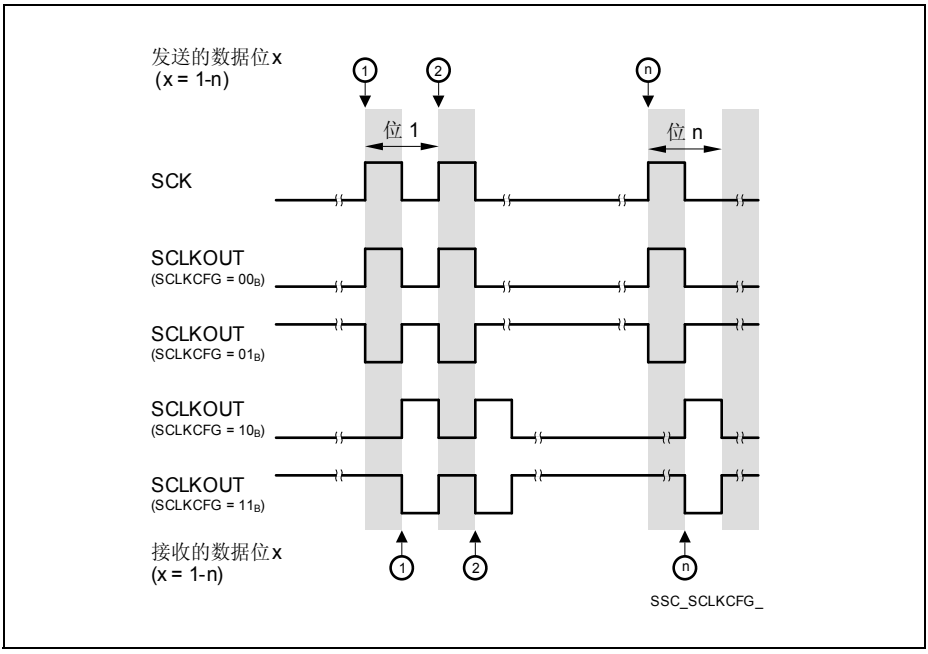


图 15-39　　SSC 主模式下的 SCLKOUT 配置

注：　如果选择了一个带延迟的配置并且使用一个从选择线，则必须相应地设置从选择延迟。

在 SSC 从模式，位 PCR.SLPHSEL 可用于配置数据移位时钟的相位。

- 当 SLPHSEL = 0_B 时，从 SSC 在所选移位时钟输入 (SCLKIN) 的每个前沿发送数据位，在 SCLKIN 的每个后沿接收数据位。
- 当 SLPHSEL = 1_B 时，一旦所选择的从选择输入 (SELIN) 变为有效，从 SSC 即发送第一个数据位。如果不使用 SELIN，则 DX2 级必须提供一个 1 电平给数据移位单元用于移出第一个位。后续的数据位随后在每个 SCLKIN 的后沿发送。SSC 从机在每个 SCLKIN 的前沿接收所有数据位。

对于这两种设置，时钟极性由 SCLKCFG 的位 0 决定。

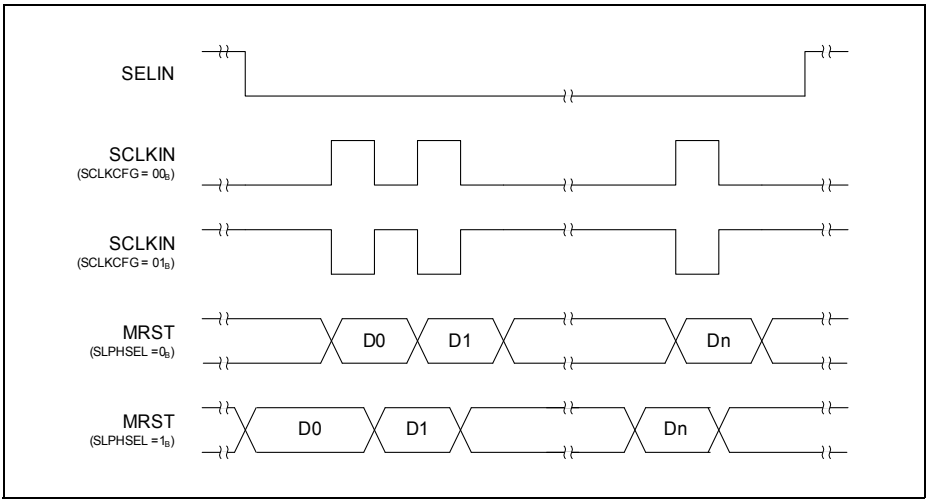


图 15-40　　SSC 从模式下的 SLPHSEL 配置

15.4.1.3　从选择信号

从机选择信号由输入级 DX2 处理。在从模式，SSC 通过一个输入引脚从一个外部主机接收输入信号 SELIN。输入级可以将接收的输入信号反相，以适应用于数据移位单元功能的信号 SELIN 的极性。(模块内部信号被视为高电平有效，因此当数据移位单元的从选择输入为 1 电平时才能进行数据传送，否则，移位时钟脉冲被忽略，不会导致数据传送)。如果输入信号 SELIN 为低电平有效，则应在 DX2 输入级将其反相。

在主模式，主机的从选择信号 MSLS 由内部从选择生成器产生。为了能独立寻址不同的外部从器件，内部 MSLS 信号通过最多 8 个 SELOx 输出信号提供给外部器件， SELOx 信号可由块 SELCFG (选择配置) 配置。

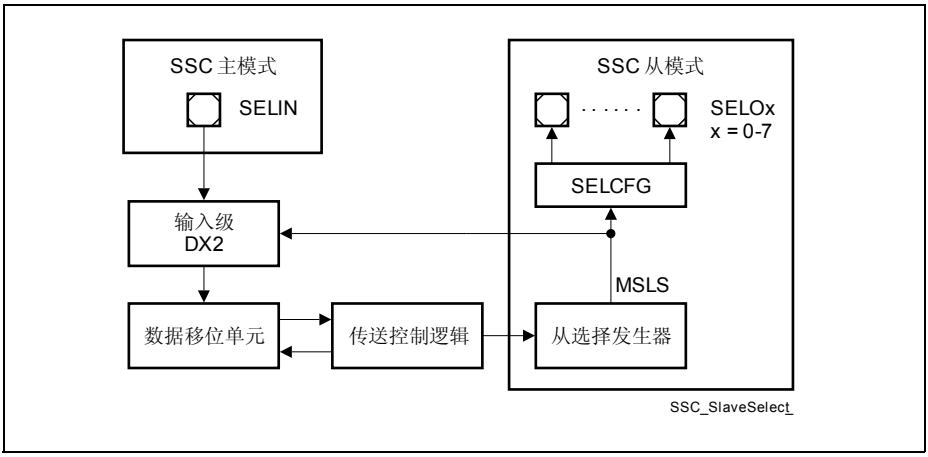


图 15-41　　SSC 从选择信号

SELCFG 块的控制基于协议控制寄存器 PCR 中协议专用的位和位域。MSLS 信号的产生请参见 **15.4.3.2 节**。

- PCR.SELCTR 用于在直接模式和编码选择模式之间进行选择
- PCR.SELINV 用于将 SELOx 输出反相
- PCR.SELO[7:0] 分别作为每个 SELOx 线的值

SELCFG 块支持 SELOx 输出信号的以下配置：

- 直接选择模式 (SELCTR = 1)：
  每个 SELOx 线 (其中 x = 0-7) 都可被直接连接到一个外部的从器件。如果位域 SELO 中的位 x 为 0，则 SELOx 输出永远无效。当内部信号 MSLS 有效 (见 **15.4.3.2 节**) 并且位域 SELO 中的位 x 为 1 时， SELOx 输出变为有效。如果在一个数据帧期间位域 SELO 中有不只一位被置位，则可以并行寻址多个外部从器件。可单独寻址的外部从器件的数量受可用 SELOx 输出数量的限制。
- 编码选择模式 (SELCTR = 0)：
  SELOx 线 (其中 x = 1-7) ，可以被用作一个外部地址译码器的地址，以增加外部从器件的数量。这些线只能在一个新数据帧开始时改变，与 MSLS 没有其他关系。信号 SELO0 可用作外部地址译码器的使能信号。当 MSLS 有效 (在一个数据帧期间) ，并且位域 SELO 中的位 0 为 1 时， SELO0 信号有效。此外，在编码选择模式，该输出线与 MSLS 相比被延迟一个 f_{PB} 周期，以允许其他 SELOx 线在地址译码器被使能之前，变得稳定。

15.4.2　操作 SSC

本节描述 SSC 的一般性问题，与主模块或从模块都没有直接联系。

15.4.2.1 自动映射机制

当自动映射机制传送一个数据帧时 (= 当 MSLS 有效时)，波特率控制寄存器 BRG、位域 SCTR.FLE 和协议控制寄存器 PCR 的内容在内部保持不变。可以在任何时刻将这些寄存器编程为新的设置，这些新设置在下一个数据帧生效。在一个数据帧期间，虽然在数据帧开始后写入了新设置值，但所使用的 （映射的）设置并未改变。

位域 SCTR.WLE、SCTR.DSM、SCTR.HPCDIR 和 SCTR.SDIR 在每个数据字的开始被自动映射。其结果是，一个数据帧可由不同长度的数据字组成，数据字可通过不同数量的数据线发送和接收。建议只在没有数据帧正在运行时改变 SCTR.SDIR，以避免硬件和软件之间互相干扰 。

请注意，一个数据字的起始点对于发送器 （第一个发送位）和接收器 （第一个接收位）可以不同。为了确保正确的处理，建议在接收器开始中断RSI发生后再修改SCTR.WLE。如果 TCSR.WLEMD = 1，建议在接收器开始中断产生后再更新 TCSR 和 TBUFxx。

15.4.2.2 模式控制行为

SSC 模式支持下列内核模式：

- 运行模式 0/1：
  行为与编程设置的一样，对数据传送没有影响。
- 停止模式 0/1：
  发送缓冲区的内容被视为发送无效。虽然被视为 0，但位 TCSR.TDV 不会由停止模式条件修改。
  在主模式，当前正在运行的字传送正常结束，但未开始传送新的数据字（停止条件不被视为帧结束条件）。在从模式，当前正在运行的字传送正常结束。如果外部主器件在从器件处于停止模式时启动了一次数据字传送，则从器件将发送被动数据而不是一个有效数据字。为了避免出现被动从发送数据，如果主器件与从器件的停止模式不一致，建议不将 SSC 从器件编程为停止模式。

15.4.2.3 禁止 SSC 模式

要在不破坏任何数据的情况下禁止 SSC 模式，接收器和发射器都必须是空闲的。通过在寄存器 KSCFG 中请求停止模式 1 可以保证这一点。在停止模式 1 由 KSCFG.2 = 1 应答后，SSC 模式可以被禁止。

15.4.2.4 数据帧控制

一个 SSC 数据帧可以由多个连续的数据字组成，这些数据字可能会被字间延迟分开。这些没有字间延迟的数据字相当于一个数据帧，但似乎组成了一个更长的数据字。一个数据帧内的数据字的长度通常是相同的，但也可以不同。需要计算得出每个新数据字的数据字长度信息 (由 SCTR.WLE 定义)，而帧长度信息 (由 SCTR.FLE 定义) 则在每次启动一个新帧时计算。
一个 SSC 数据帧的长度可以用两种不同的方法定义：

- 由每帧的位数：
  如果定义了每个数据帧的位数 (帧长度 FLE)，则可不需要使用从选择信号来指示一个数据帧的开始和结束。
  如果所编程的每帧位数在一个数据字之内，则该帧被视为结束，最后一个数据字的剩余数据位被忽略，不发送这些位。
  该方法适用于具有最多 63 个数据位的数据帧。
- 由从选择信号：
  如果每个数据帧的位数是未知的，则一个数据帧的开始 / 结束信息可以由一个从选择信号给出。如果在一个数据字内检测到从选择信号失效，则认为该帧已经结束，最后一个数据字的剩余数据位被忽略，不发送这些位。
  对于具有多于 63 个数据位 (FLE 的编程极限值) 的数据帧，必须使用该方法。从选择信号的优点是明确地定义了一个数据流中数据帧的起始和结束条件。此外，从选择信号还允许单独寻址从设备。

15.4.2.5 奇偶校验模式

SSC 允许基于帧为发送生成奇偶校验位和为接收检查奇偶校验位。奇偶校验的类型可由位域 CCR.PM 选择，对发送和接收是统一的 (无校验、偶校验或奇校验)。如果禁止奇偶校验处理，则 SSC 帧不包含任何校验位。出于一致性的原因，所有的通信伙伴都必须被编程为相同的奇偶校验模式。

如果使能了奇偶校验位生成功能，发送器会在数据帧的最后一个数据字之后自动扩展一个周期时钟，并在该周期发送自己计算的奇偶校验位。

图 15-42 展示了如何在一个发送帧的数据位中添加奇偶校验位。带奇偶校验的一个完整帧的发送位数总是比不带奇偶校验的多一位。奇偶校验位被作为一帧的最后一位发送，紧随数据位之后，与移位方向 (SCTR.SDIR) 无关。

注：　对于双位和四位 SSC 协议，在扩展的时钟周期，奇偶校验位仅在 DOUT0 和 DX0 分别发送和接收。

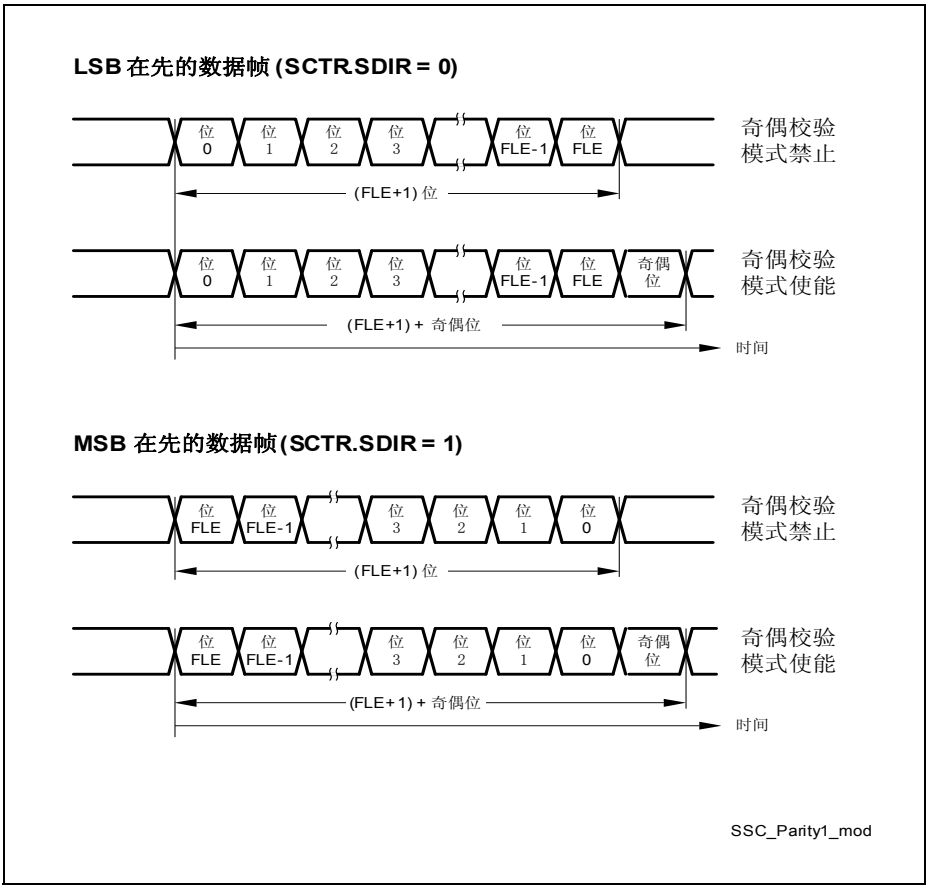


图 15-42　　有 / 无奇偶校验的数据帧

类似地，接收器在接收到由 FLE 定义的一个数据帧的最后一个字之后，它还需要一个额外的时钟周期，该周期包含奇偶校验位。接收器将该位解译为接收的奇偶校验位，并将其与接收的数据分离。接收到的奇偶校验位值在接收缓冲器状态寄存器的协议相关参数 (PAR) 中被监视，作为接收缓冲器状态信息。接收器将该位与其内部计算的校验位进行比较，奇偶校验的结果由标志 PSR.PARERR 指示。如果 PCR.PARIEN = 1，则奇偶校验错误事件产生一个协议中断。

在下列情况下，不支持奇偶校验位的产生和检测：

- 当帧长为 64 个数据位或更多时，如 FLE = 63_H；
- 在从模式，帧结束发生在达到由 FLE 定义的数据位数之前。

15.4.2.6 传送模式

在 SSC 模式，必须编程使位域 SCTR.TRM = 01_B 才能允许数据传送。设置 SCTR.TRM = 00_B 将禁止和立即停止数据传送。

15.4.2.7 数据传送中断处理

数据传送中断指示与 SSC 帧处理相关的事件。

- 发送缓冲器中断 TBI：
  在一个数据字的第一个数据位开始后，位 PSR.TBIF 被置 1。
- 发送移位中断 TSI：
  在一个数据字的最后一个数据位开始后，位 PSR.TSIF 被置 1。
- 接收器启动中断 RSI：
  在接收到一个数据字的第一个数据位后，位 PSR.RSIF 被置 1。
  发生该事件时，位 TCSR.TDV 被清零，可以将新数据加载到发送缓冲器。
- 接收器中断 RI：
  在一个多字帧中，第二、第三和所有后续字的接收总是由位 RBUFSR.SOF = 0 指示。
  如果 RBUFSR.SOF = 0，则接收到一个数据字的最后一个数据位后，位 PSR.RIF 被置 1。
  位 RBUFSR.SOF 指示接收到的数据字是一个多字帧的第一个数据字还是某个后续字。在 SSC 模式，该位决定是产生备用中断还是产生接收中断。
- 备用中断 AI：
  一个帧中第一个字的接收总是由 RBUFSR.SOF = 1 指示。这对多字帧和单字帧接收都是正确的。在 SSC 模式，这将导致置位 PSR.AIF。

15.4.2.8 波特率发生器中断处理

波特率发生器中断指示捕获模式定时器已达到其最大值。发生该事件时，位 PSR.BRGIF 被置 1。

15.4.2.9 协议相关的参数和错误

协议相关参数 (RBUFSR.PAR) 和协议相关错误 (RBUFSR.PERR) 是两个标志，它们位于相应的接收缓冲器状态寄存器中，被分配给每一个接收的数据字。

在 SSC 模式，接收到的奇偶校验位由协议相关的参数监视。接收的帧起始指示由协议相关错误指示位监视 (0 = 接收到的字不是一帧中的第一个字， 1 = 接收到的字是一帧中的第一个字)。

注： 对于SSC，奇偶校验错误事件指示位位于PSR 寄存器中。

15.4.2.10 接收缓冲区处理

在 SSC 模式，如果一个接收 FIFO 缓冲区是可用的 (CCFG.RB = 1)，并且被使能用于数据处理 (RBCTR.SIZE > 0)，则建议设置 RBCTR.RCIM = 01_B。这会导致：位

OUTR.RCI[4] = 1 指示该数据字是一个新数据帧的第一个数据字，所接收数据的字长由 OUTR.RCI[3:0] 给出。

在 RCI 模式 (RBCTR.RNM = 1)，标准接收缓冲区事件和备用接收缓冲区事件可以用于执行以下操作：

- 一个标准接收缓冲区事件指示可以从 OUTR 读取一个数据字，这个已接收的数据字不是一个数据帧的第一个字。
- 一个备用接收缓冲区事件指示可从 OUTR 读取一个新数据帧的第一个数据字。

15.4.2.11 多 I/O SSC 协议

SSC 实现了以下三个功能，以支持多数据输入 / 输出 SSC 协议，如双位和四位 SSC：

1. 数据移位模式 (见 **15.2.5.2 节**)
   通过位域 SCTR.DSM 配置发送和接收数据时并行使用一条、两条或四条数据线。
2. 硬件端口控制 (见 **15.2.7 节**)
   通过位 SCTR. HPCDIR 设置一个专用硬件接口，以控制具有 DINx 和 DOUTx 两种功能的引脚的方向。
3. 发送控制信息 (见 **15.2.5.3 节**)
   在数据传送期间，通过向 SCTR.DSM 和 SCTR.HPCDIR 写 TCI 可动态控制移位模式和引脚方向。

图 15-43 示出了四位 SSC 协议的一个例子，该协议要求主 SSC 首先通过一根数据线发送一个命令字节 (请求自从机一次读取 4 位输出) 和一个空字节。在空字节结束后，主和从 SSC 都切换到四数据线方式，并且发送器和接收器角色互换。然后主 SSC 通过 MRST[3:0] 线在每个移位时钟自从机接收四位数据。

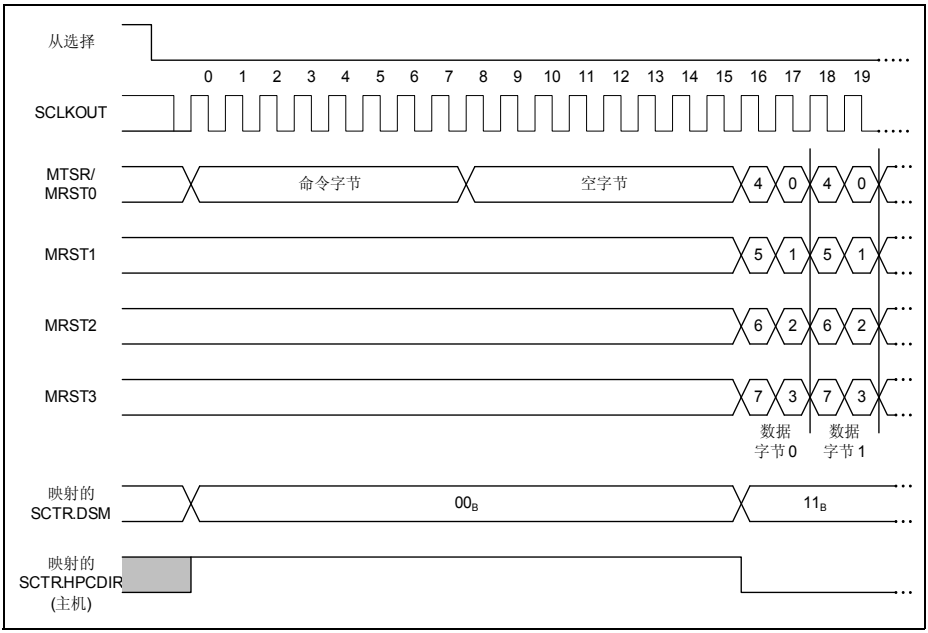


图 15-43　　四位 SSC 示例

为了在上面给出的例子中使用四位 SSC 协议工作，在 **15.4.3 节** 和 **15.4.4 节**所描述的内容之外还必须额外考虑以下问题：

- 在初始化阶段：
  - 设置 CCR.HPCEN 为 11_B，以使能与 DX0/DOUT0、 DX3/DOUT1、DX4/DOUT2 和 DX5/DOUT3 引脚的专用硬件接口。
  - 设置 TCSR.[4:0] 为 10_H，以使能 TCI 中的硬件端口控制。
- 启动数据传送：
  - 对于主 SSC，写命令和空数据到 TBUF04，以选择一条数据线为输出模式并启动数据传送。
  - 对于从 SSC，可以向 TBUF00 预装空数据，以选择一条数据线为输入模式。
- 切换到四数据线方式和并切换引脚方向：
  - 对于主 SSC，写后续的空数据到 TBUF03 ，以选择四条数据线为输入模式，用于读入有效的从数据。
  - 对于从 SSC，写有效数据到 TBUF07，该数据通过工作在输出模式的四条数据线发送。

图 15-44 示出了四位 SSC 连接的例子。

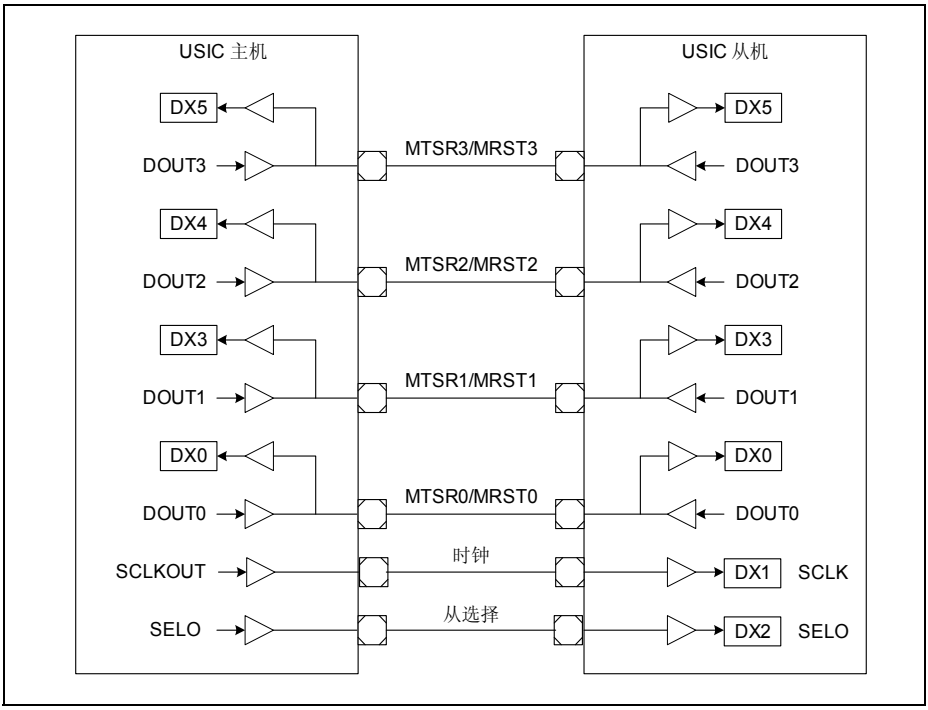


图 15-44　　四位 SSC 连接示例

15.4.3　操作主模式下的 SSC

为了操作主模式下的 SSC，必须考虑以下问题：

- 选择 SSC 模式：
  建议在 CCR.MODE = 0000_B 期间配置那些不能在运行时改变的所有 SSC 参数。必须将位域 SCTR.TRM 编程为 01_B。输入级的配置必须在 CCR.MODE = 0000_B 时完成，以避免意外的输入信号边沿，然后可以通过设置 CCR.MODE = 0001_B 来使能 SSC 模式。
- 引脚连接：
  通过设置 DXnCR.INSW = 1 建立输入级 (DX0, DX3, DX4, DX5) 与所选接收数据输入引脚 (DIN[3:0]) 的连接，并配置发送数据输出引脚 (DOUT[3:0])。根据所选协议，可能需要一个、两个或四个这样的连接。对于半双工配置，还可使用硬件端口控制来建立所需要的连接。
- 波特率产生：
  必须选择所需要的波特率设置，包括小数分频器和波特率生成器。要直接使用波特率

发生器输出 SCLK 作为数据移位单元的输入，必须将位 DX1CR.INSW 编程为 0。还要配置一个移位时钟输出引脚 (信号 SCLKOUT)。

- 从选择产生：
  从选择延迟的产生必须通过设置 PCR.MSLSEN = 1 和对时间量子计数器设置编程来使能。必须设置 DX2CR.INSW= 0，以使用从选择生成器的输出 MSLS 作为数据移位单元的输入。如果需要，还要配置从选择输出引脚 (信号 SELOx)。
- 数据格式的配置：
  必须根据应用需要，通过对寄存器 SCTR 编程来设置字长、帧长、移位方向和移位模式。

注：　如果 USIC 正在发送，则它只能在主模式下接收，因为主帧的处理要借助于发送器一方的位 TDV。

注：　为了避免在输出端出现意外的尖峰脉冲，必须在 SSC 模式被使能后再执行使能备用输出端口功能的步骤。

15.4.3.1　波特率产生

SSC 的波特率 （确定一个数据位的长度）由 SCLK 信号的频率定义 (一个 f_{SCLK} 周期代表一个数据位)。SSC 的波特率产生不需使用任何时间量子计数器。

在一个标准 SSC 应用中，可选的 MCLK 输出信号与 SCLK 之间的相位关系是无关的，且可以被禁止 (BRG.PPPEN 为 0)。在这种情况下，SCLK 信号直接源自协议输入频率f_{PIN}。在 MCLK 信号和 SCLK 之间需要一个固定的相位关系这一例外情况下 (例如，当使用 MCLK 作为外部器件的参考时钟时)，必须考虑使用额外的 2 分频级 (BRG. PPPEN = 1)。

可调节的分频系数由位域 BRG.PDIV 定义。

$$f_{SCLK} = \frac{f_{PIN}}{2} \times \frac{1}{PDIV + 1} \quad \text{若 PPPEN = 0}$$
$$f_{SCLK} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} \quad \text{若 PPPEN = 1} \qquad (15.8)$$

15.4.3.2　MSLS 产生

从选择信号指示一个数据帧的开始和结束，通信主机还用其单独选择所期望的从器件。通信主机的一个从选择输出在帧的数据部分开始之前一段时间 （超前延迟 T_{ld}）变为有效，该时间是可编程的，对使从器件做好准备以进行接下来的通信必不可少。在帧的一个数据部分被传送之后，该从选择信号在最后一位结束后经过一段可编程的时间 （拖尾延迟 T_{td}）再次变为无效。如果数据帧是一个接一个地连续传送，则自从选择失效到从选择下一次有效之间的最小时间是可编程的 （下一帧延迟 T_{nf}）。如果一个数据帧包含不只一个数据字，也可以对数据字之间的可选延迟编程 （字间延迟 T_{iw}）。

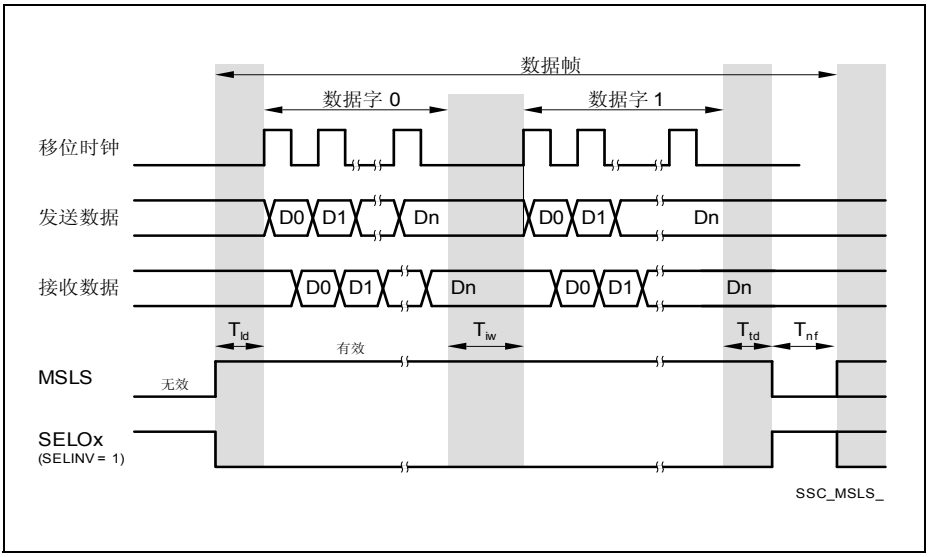


图 15-45　　SSC 主模式下的 MSLS 产生

在 SSC 主模式，从选择延迟定义如下：

- 超前延迟 T_{ld}：
  如果用于发送的数据有效，则超前延迟开始。内部信号 MSLS 随着超前延迟的开始变为有效。在经过了超前延迟这段时间后，波特率发生器产生移位时钟 SCLK 的第一个边沿 （上升沿）。
- 拖尾延迟 T_{td}：
  拖尾延迟在一个数据帧周期的最后一个 SCLK 周期结束时开始。内部信号 MSLS 在拖尾延迟结束后变为无效。
- 字间延迟 T_{iw}：
  该延迟是可选的，可以由 PCR.TIWEN 使能或禁止。如果字间延迟被禁止（TIWEN = 0），则一个数据字的最后一个数据位后面直接跟随同一数据帧的下一个数据字的第一个数据位。如果被使能 (TIWEN = 1)，则字间延迟在一个数据字的最后一个 SCLK 周期结束时开始。同一数据帧的下一个数据字的第一个 SCLK 周期在字间延迟结束后开始。在此期间，不产生移位时钟冲脉，信号 MSLS 保持有效。通信伙伴有时间 “ 消化 ” 以前的数据字或准备下一个数据字。
- 下一帧延迟 T_{nf}：
  下一帧延迟在拖尾延迟结束时开始。在此期间，不产生移位时钟脉冲，信号 MSLS 保持无效。在下一帧延迟消逝后，一个数据帧帧被视为结束。

15.4.3.3 自动从选择更新

如果每个 SSC 帧的位数和字长由位域 SCTR.FLE 和 SCTR.WLE 定义，则可使用发送控制信息 TCI 来更新从选择设置 PCR.CTR[23:16]，以控制 SELOx 选择输出。自动更新机制由 TCSR.SELMD = 1 使能 (位 TCSR.WLEMD、FLEMD 和 WAMD 必须被清零)。在这种情况下，由于自动映射机制的作用，一个帧的第一个数据字的 TCI 定义整个帧的从选择设置 (见 页 15-51)。

15.4.3.4 从选择延时产生

从选择延时的产生是基于时间量子的。一个时间量子的长度 (由 f_{CTQIN} 的周期定义) 和每个延迟的时间量子数目是可编程的。

在标准 SSC 应用中，超前延迟 T_{ld} 和拖尾延迟 T_{td} 主要用于确保输入和输出线上信号的稳定性以及遵守输入级的建立和保持时间。这两个延迟有相同的长度(在大多数情况下比一个位时间短)，可以用相同的位域集编程。

- BRG.CTQSEL
  为 T_{ld} 和 T_{td} 定义用于产生时间量子的输入频率 f_{CTQIN}
- BRG.PCTQ
  为 T_{ld} 和 T_{td} 定义一个时间量子 的长度 (f_{CTQIN} 被 1、 2、 3 或 4 分频)
- BRG.DCTQ
  定义为产生延迟 T_{ld} 和 T_{td} 所需要的时间量子数

字间延迟 T_{iw} 和下一帧延迟 T_{nf} 用于处理接收的数据或为下一个字或帧准备数据。这两个延迟具有相同的长度 (在大多数情况下位于位时间的范围内)，可用另一个独立的位域集编程。

- PCR.CTQSEL1
  为 T_{nf} 和 T_{iw} 定义用于产生时间量子的输入频率 f_{CTQIN}
- PCR.PCTQ1
  为 T_{nf} 和 T_{iw} 定义一个时间量子的长度 (f_{CTQIN} 被 1、 2、 3 或 4 分频)
- PCR.DCTQ1
  定义为产生延迟 T_{nf} 和 T_{iw} 所需要的时间量子数
- PCR.TIWEN
  使能 / 禁止字间延迟 T_{iw}

每个延迟取决于时间量子的长度和由位域CTQSEL/CTQSEL1、PCTQ/DCTQ 和PCTQ1/DCTQ1 (CTQSEL1 的编码类似于 CTQSEL，等等) 所编程的时间量子数。为了给延迟长度的编程提供更高的灵活性，输入频率可以在多种可能之间选择。(例如，基于位时间或更快的协议相关分频器的输入)。延迟时间定义如下：

$$T_{ld} = T_{td} = \frac{(PCTQ + 1) \times (DCTQ + 1)}{f_{CTQIN}}$$
$$T_{iw} = T_{nf} = \frac{(PCTQ1 + 1) \times (DCTQ1 + 1)}{f_{CTQIN}} \quad (15.9)$$

15.4.3.5 协议中断事件

在 SSC 模式可产生下述协议相关的事件，这些事件可导致产生一个协议中断。这些事件与一个数据帧的起始和结束相关。在一个数据帧开始以后，可以对下一数据帧所需要的新设置进行编程，在一个数据帧结束之后，可以改变 SSC 与引脚的连接。

请注意，寄存器 PSR 中的位并不都是由硬件自动清零的，为了监视新到来的事件，必须用软件清零。

- MSLS 中断：
  在主模式 (MSLS 产生被使能)，该中断指示一个数据帧已经开始 (MSLS 激活) 并且已经结束 (MSLS 失活)。内部 MSLS 信号的任何改变都会置位 PSR.MSLSEV，此外，如果 PCR.MSLSIEN = 1，则会产生一个协议中断。当检测到该中断时，内部 MSLS 信号的当前状态可在 PSR.MSLS 读出，以采取适当的动作。
- DX2T 中断：
  该中断监视DX2级输入信号的边沿(虽然该信号不作为数据传送的从选择输入使用)。对 DX2 输入信号的可编程边沿检测会置位 PSR.DX2TEV 位，此外，如果 PCR.DX2TIEN = 1，则会产生一个协议中断。当检测到该中断时，所选输入信号的当前状态可在 PSR.DX2S 读出，以采取适当的动作。
- 奇偶校验错误中断：
  该中断指示接收的奇偶校验位 (在 RBUFSR.PAR 中) 与用一个数据帧最后接收到的字所计算出的校验位不一致。

15.4.3.6 帧结束控制

对于主设备的 MSLS 产生有关帧长度的信息是必需的。除了基于每帧位数的机制 (用 SCTR.FLE 选择，< 63) 之外，SSC 还支持选择下述帧结束处理机制。建议设置 SCTRFLE = 63 (如果多个帧结束机制被并行激活，那么第一个发现的结束条件会结束该帧)。

- 基于软件的帧起始指示 TCSR.SOF：
  如果软件在不使用数据 FIFO 的情况下处理 TBUF 数据，则可使用该机制。如果位 SOF 被置位，则 TBUF 中的有效内容被视为一个新帧的第一个字。在 TBUF 的内容被传送到发送移位寄存器之前，位 SOF 必须被置位，因此建议在向 TBUF 写数据到之前写 SOF 位。在当前数据字传送完全结束并且施加了从选择延迟 T_{td} 和 T_{nf} 之后，一个使用 T_{ld} 下一个有效 TBUF 值的新数据帧方可开始。
  为了对位 SOF 进行软件处理，必须编程使位 TCSR.WLEMD = 0。在这种情况下，所有 TBUF[31:0] 地址单元都表现出完全相同的行为 (数据处理时不考虑 TCI)。
- 基于软件的帧结束指示 TCSR.EOF：
  如果软件在不使用数据 FIFO 的情况下处理 TBUF 数据，则可使用该机制。如果位 EOF 被置位，则 TBUF 中的有效内容被视为一个新帧的最后一个字。在 TBUF 的内容被传送到发送移位寄存器之前，位 EOF 必须被置位，因此建议在向 TBUF 写数据之前写 EOF 位。TBUF 中的数据字完全发送出去后在施加从选择延迟 T_{td} 和 T_{nf}。此时可以开始一个使用 T_{ld} 下一个有效 TBUF 值的新数据帧。
  为了对位 EOF 进行软件处理，必须编程使位 TCSR.WLEMD = 0 。在这种情况下，所有 TBUF[31:0] 地址单元都表现出完全相同的行为 (数据处理时不考虑 TCI)。

- 基于软件的地址相关帧结束处理：
  如果软件在不使用数据 FIFO 的情况下处理 TBUF 数据，则可使用该机制。如果位 TCSR.WLEMD = 1，所写的 TBUF[31:0] 地址被用作发送控制信息 TCI[4:0]，以更新每个数据字的 SCTR.WLE (= TCI[3:0]) 和 TCSR.EOF (= TCI[4])。所写的 TBUF[31:0] 地址单元定义了字长和一帧的结束 (地址单元 TBUF[31:16] 导致一帧结束)。
  例如，向 TBUF[07] 写发送数据会导致发送一个 8 位长的数据字，不会结束该帧；然而向 TBUF[31] 写发送数据会导致发送一个长度为 16 位的数据字，后面跟随 T_{td}、MSLS 失活和 T_{nf}。
  如果 TCSR.WLEMD = 1，在软件写数据到一个 TBUF 单元后，软件不得对位 TCSR.EOF 和 SOF 以及 SCTR.WLE 进行写操作。此外，建议将位 TCSR.SELMD、FLEMD 和 WAMD 清零。
- 基于 FIFO 的地址相关帧结束处理：
  如果使用一个数据 FIFO 来存储发送数据，则可使用该机制。除了发送数据不是被写入到单元 TBUF[31:0] 而是被写入到 FIFO 输入单元之外，该机制的一般行为与基于软件的地址相关帧结束处理类似。在这种情况下，软件不得写任何 TBUF 单元。
- TBUF 相关的帧结束处理：
  当位 PCR.FEM = 0 时，在一个数据字发送结束后，如果发送缓冲区 TBUF 不包含有效发送数据 (TCSR.TDV = 0 或处于停止模式)，则可假定该帧已经结束。在这种情况下，软件必须注意在一个数据帧运行期间 TBUF 不能为空。如果位 PCR.FEM = 1，在发送缓冲区等待新数据期间 (TCSR.TDV 再次变为 1) 或直到脱离停止模式时，信号 MSLS 保持有效。
- 软件控制的显性帧结束：
  软件可通过清除位 PSR.MSLS 来显性停止一个帧，PSR.MSLS 位的清除通过向寄存器 PSCR 中的相关位置写 1 实现。该写操作会立即清除位 PSR.MSLS，而内部的 MSLS 信号在一个正在运行的字传送结束并经过从选择延迟 T_{td} 和 T_{nf} 之后才变为无效。

15.4.4 操作从模式下的 SSC

为了操作从模式下的 SSC，必须要考虑以下问题：

- 选择 SSC 模式：
  建议在 CCR.MODE = 0000_B 期间配置那些不能在运行时改变的所有 SSC 参数。必须将位域 SCTR.TRM 编程为 01_B。输入级的配置必须在 CCR.MODE = 0000_B 时完成以避免意外的输入信号边沿，然后可以通过设置CCR.MODE = 0001_B来使能SSC模式。
- 引脚连接：
  通过设置 DXnCR.INSW = 1 建立输入级 (DX0, DX3, DX4, DX5) 与所选接收数据输入引脚 (DIN[3:0]) 的连接，并配置发送数据输出引脚 (DOUT[3:0])。根据所选协议，可能需要一个、两个或四个这样的连接。对于半双工配置，还可使用硬件端口控制来建立所需要的连接。
  通过设置DX1CR.INSW = 1建立输入级DX1与所选移位时钟输入引脚(信号SCLKIN)的连接。
  通过设置 DX2CR.INSW = 1 建立输入级 DX2 与所选从选择输入引脚 (信号 SELIN) 的连接。如果不使用从选择输入信号，则 DX2 级必须提供一个 1 电平给数据移位单元，

以允许数据接收和发送。如果一个从设备未被选中 (DX2 级提供一个 0 电平给数据移位单元) 并且接收到一个移位时钟脉冲，则到来的数据不被接收，并且 DOUTx 信号输出由 SCTR.PDL 定义的被动数据电平。
注意，只有在使能了 SSC 模式后方可执行使能备用输出端口功能的步骤，以避免在输出的信号上出现意外的尖峰脉冲。

- 波特率发生器：
  从模式不需要波特率发生器，可通过小数分频器将其关闭。
- 数据格式配置：
  如果需要，可通过对寄存器 SCTR 进行一次编程将移位模式设置为接收和 / 或发送两个或四个数据位。
- 从选择产生：
  从选择延迟产生是不需要的，可以关闭。寄存器 PCR 中的位和位域 MSLSEN、SELCTR、SELINV、CTQSEL1、PCTQ1、DCTQ1、MSLSIEN、SELO[7:0] 和 TIWEN 不是必需的，可以编程为 0。

15.4.4.1 协议中断

在 SSC 模式产生的下述协议相关事件可导致产生一个协议中断。这些事件与一个数据帧的开始和结束相关。在一个数据帧开始之后，可以对下一数据帧所需要的新设置进行编程，在一个数据帧结束之后，可以改变 SSC 与引脚的连接。

请注意，寄存器 PSR 中的位并不都是由硬件自动清零的，为了监视新到来的事件，必须用软件清零。

- MSLS 事件：
  MSLS 产生被关闭，该事件不可用。
- DX2T 事件：
  从选择输入信号 SELIN 由 DX2 级处理，所选信号的边沿可产生一个协议中断。该中断允许指示一个数据帧已经启动和 / 或一个数据帧已经完全结束。
  对 DX2 输入信号的可编程边沿检测会激活 DX2T，将位 PSR.DX2TEV 置 1，并且如果 PCR.DX2TIEN = 1，还会产生一个协议中断。当检测到该中断时，所选输入信号的当前状态可以在 PSR.DX2S 读出，以采取适当的动作。
- 奇偶校验错误中断：
  该中断指示接收的奇偶校验位 (在 RBUFSR.PAR 中) 与用一个数据帧最后接收到的字所计算出的校验位不一致。

15.4.4.2 结束帧控制

在从模式，存在以下几种决定帧长度的可能。从器件必须依赖一个外部从选择信号或者依赖接收的数据位数。

- 从器件可事先得知帧的长度，无从选择信号：
  在这种情况下，位域 SCTR.FLE 可以编程为已知值 (如果它不超过 63 位)。如果达到了编程的帧长度，则当前运行的数据字传送被视为结束。
- 从器件不知道帧长度，无从选择信号：
  在这种情况下，从器件的软件必须基于数据字来决定一个帧完是否结束。位域

SCTR.FLE 可以被编程为字长 SCTR.WLE，也可以被编程为其最大值，以禁止从器件通过对接收的数据位计数来计算从机内部的帧长度。

- 通过从选择信号 SELIN 寻址从器件：
  如果从器件由一个通信主机发出的从选择信号寻址，则帧的起始和结束信息由该信号给出。在这种情况下，位域 SCTR.FLE 应被编程为其最大值，以禁止从器件内部的帧长计算。

15.4.5 SSC 协议寄存器

在 SSC 模式，寄存器 PCR 和 PSR 处理与 SSC 相关的信息。

15.4.5.1 SSC 协议控制寄存器

在 SSC 模式， PCR 寄存器的位或位域依本节所述定义。

PCR
协议控制寄存器 [SSC 模式] **($3C_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCLK	0					SLPHSEL	TIWEN	SELO							
rw	rw					rw	rw	rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DX2TIEN	MSLSIEN	PARIEN	DCTQ1					PCTQ1		CTQSEL1		FEM	SELINV	SELCTR	MSLSEN
rw	rw	rw	rw					rw		rw		rw	rw	rw	rw

域	位	类型	描述
MSLSEN	0	rw	**MSLS 使能** 该位使能 / 禁止主器件的从选择信号 MSLS 的产生。如果 SSC 是一次传送中的从器件，则它从一个引脚读取 SLS 信息，不需从内部产生。如果 SSC 是一次传送中的主器件，则它必须提供 MSLS 信号。 0_B 禁止 MSLS 产生 (MSLS = 0)。 这是 SSC 从模式的设置。 1_B 使能 MSLS 产生。 这是 SSC 主模式的设置。

域	位	类型	描述
SELCTR	1	rw	**选择控制** 该位选择 SELO[7:0] 输出的工作模式。 0_B 使能编码选择模式。 1_B 使能直接选择模式。
SELINV	2	rw	**选择信号反相** 该位定义 SELO[7:0] 输出的极性与主器件的从选择信号 MSLS 的关系。 0_B SELO 输出具有与 MSLS 信号相同的极性 (高电平有效)。 1_B SELO 输出具有与 MSLS 信号相反的极性 (低电平有效)。
FEM	3	rw	**帧结束模式** 该位定义是否将一个非发送有效的发送缓冲器内容视为一个帧结束条件，以决定是否产生从选择信号。 0_B 当一个数据字的最后一位被发送且发送缓冲区 TBUF 不包含新数据时 (TDV = 0)，当前数据帧被视为结束。 1_B 在没有新数据可用并且没有其他帧结束条件满足时， MSLS 信号仍保持有效。在这种情况下，软件可接受在多字数据帧中提供数据带来的延迟，不会自动停止 MSLS 信号。
CTQSEL1	[5:4]	rw	**输入频率选择** 该位域定义了用于产生从选择延迟 T_{iw} 和 T_{nf} 的输入频率 f_{CTQIN}。 00_B $f_{CTQIN} = f_{PDIV}$ 01_B $f_{CTQIN} = f_{PPP}$ 10_B $f_{CTQIN} = f_{SCLK}$ 11_B $f_{CTQIN} = f_{MCLK}$
PCTQ1	[7:6]	rw	**T_{iw} 和 T_{nf} 的分频系数 PCTQ1** 该位域代表用于产生字间延迟和下一帧延迟的分频系数 PCTQ1 (范围 = 0 - 3)。 $T_{iw} = T_{nf} = 1/f_{CTQIN}$ x (PCTQ1 + 1) x (DCTQ1 + 1)
DCTQ1	[12:8]	rw	**T_{iw} 和 T_{nf} 的分频系数 DCTQ1** 该位域代表用于产生字间延迟和下一帧延迟的分频系数 DCTQ1 (范围 = 0 - 31)。 $T_{iw} = T_{nf} = 1/f_{CTQIN}$ x (PCTQ1 + 1) x (DCTQ1 + 1)

域	位	类型	描述
PARIEN	13	rw	**奇偶校验错误中断使能** 该位使能 / 禁止在检测到奇偶校验错误时产生一个协议中断。 0_B 在检测到奇偶校验错误时不产生协议中断。 1_B 在检测到奇偶校验错误时产生协议中断。
MSLSIEN	14	rw	**MSLS 中断使能** 该位使能 / 禁止在 MSLS 信号的状态发生改变时 (由 PSR.MSLSEV = 1 指示) 产生一个协议中断。 0_B 在检测到信号 MSLS 发生改变时不产生协议中断。 1_B 在检测到信号 MSLS 发生改变时产生协议中断。
DX2TIEN	15	rw	**DX2T 中断使能** 该位使能 / 禁止在 DX2T 信号变为激活状态时 (由 PSR.DX2TEV = 1 指示)，产生一个协议中断。 0_B 在 DX2T 信号被激活时不产生协议中断。 1_B 在 DX2T 信号被激活时产生协议中断。
SELO	[23:16]	rw	**选择输出** 该位域定义 SELO[7:0] 输出线的设置。 0_B 不能激活相应的 SELOx 线。 1_B 可以激活相应的 SELOx 线 (根据由 SELCTR 选择的模式)。
TIWEN	24	rw	**使能字间延迟 T_{iw}** 该位使能 / 禁止在发送一个数据字之后的字间延迟 T_{iw} 0_B 在同一帧的数据字之间没有延迟。 1_B 使能并引入同一帧中数据字之间的字间延迟 T_{iw}。
SLPHSEL	25	rw	**从模式时钟相位选择** 该位选择从模式下数据移位时钟的相位。 0_B 数据位在移位时钟信号的前沿移出，在后沿锁存。 1_B 数据移位单元在收到来自 DX2 级的一个由低到高的跳变时，移出第一个数据位。后续位在移位时钟信号的后沿移出。数据位总是在前沿锁存。
MCLK	31	rw	**主时钟使能** 该位使能 / 禁止主时钟输出信号 MCLK 的产生，与主或从模式无关。 0_B 禁止 MCLK 产生并输出 MCLK = 0。 1_B 使能 MCLK 产生。
0	[30:26]	rw	**保留** 读访问返回 0；应写入 0。

15.4.5.2 SSC 协议状态寄存器

在 SSC 模式，PSR 寄存器的位和位域依本节所述定义。寄存器 PSR 中的位和位域不能由硬件清零。

通过向寄存器 PSCR 中的一个位写 1 可清除 PSR 寄存器中的相应标志。向 PSR 中的一个位写 1 会置位相应的标志，但不会导致进一步的行为 (不产生中断)。写 0 没有影响。应在使能一个新协议之前用软件清除 PSR 中的标志。

PSR

协议状态寄存器 [SSC 模式] **(48_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															BRGIF
r															rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF	0					PARERR	DX2TEV	MSLSEV	DX2S	MSLS
rwh	rwh	rwh	rwh	rwh	rwh	r					rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
MSLS	0	rwh	**MSLS 状态** 该位指示 MSLS 信号的当前状态。它必须由软件清除，以停止一个正在运行的帧。 0_B 内部信号 MSLS 无效 (0)。 1_B 内部信号 MSLS 有效 (1)。
DX2S	1	rwh	**DX2S 状态** 该位指示 DX2S 信号的当前状态，该信号可用作从选择输入 SELIN。 0_B DX2S 为 0。 1_B DX2S 为 1。
MSLSEV	2	rwh	**MSLS 事件检测[1)]** 该位指示自 MSLSEV 最后一次被清除以来， MSLS 信号已改变其状态。与 MSLS 状态位一起使用，可监视 MSLS 信号的激活 / 停止。 0_B MSLS 信号未改变其状态。 1_B MSLS 信号已改变其状态。

域	位	类型	描述
DX2TEV	3	rwh	**DX2T 事件检测 1)** 该位指示自 DX2TEV 最后一次被清除以来， DX2T 触发器信号已被激活。 0_B DX2T 信号尚未被激活。 1_B DX2T 信号已经被激活
PARERR	4	rwh	**奇偶校验错误事件检测 1)** 该位指示接收的奇偶校验位 (在 RBUFSR.PAR 中) 与用数据帧的最后接收到的字计算出的奇偶校验位不一致。 0_B 奇偶校验错误事件未被激活。 1_B 奇偶校验错误事件已被激活。
RSIF	10	rwh	**接收器开始指示标志** 0_B 未发生接收器开始事件。 1_B 已发生接收器开始事件。
DLIF	11	rwh	**数据丢失指示标志** 0_B 未发生数据丢失事件。 1_B 已发生数据丢失事件。
TSIF	12	rwh	**发送移位指示标志** 0_B 未发生发送移位事件。 1_B 已发生发送移位事件。
TBIF	13	rwh	**发送缓冲器指示标志** 0_B 未发生发送缓冲器事件。 1_B 已发生发送缓冲器事件。
RIF	14	rwh	**接收指示标志** 0_B 未发生接收事件。 1_B 已发生接收事件。
AIF	15	rwh	**备用接收指示标志** 0_B 未发生备用接收事件。 1_B 已发生备用接收事件。
BRGIF	16	rwh	**波特率发生器指示标志** 0_B 未发生波特率发生器事件。 1_B 已发生波特率发生器事件。
0	[9:5], [31:17]	r	**保留** 读访问返回 0 ； 在 SSC 模式不修改。

1) 在 SSC 模式，该状态位可产生一个协议中断 (见 页 15-17)。通用中断状态标志在通用中断一章中描述。

15.4.6 SSC 时序考虑

为了确保正确的数据接收和发送，输入和输出信号必须遵守一定的时序。除了模块内部时序 (因为输入滤波、事件反应时间等) 之外，还要考虑从输入引脚经输入级 (T_{in}) 到模块和从模块经输出驱动级到引脚 (T_{out}) 的时序以及信号在导线上的传播 (T_{prop})。

请注意，在 DXn 输入级中可能会有额外的延迟，因为数字滤波器和同步级会导致系统性延迟。如果使用这些功能，就必须考虑这些延迟。

15.4.6.1 闭环延迟

对于一个 SSC 连接，限制其波特率的系统固有因素是闭环延迟。在一个典型应用设置中，以全双工模式连接的一个通信主器件和一个从器件使用独立的数据线进行发送和接收。在一般情况下，所有的发送器使用一个移位时钟沿进行发送，所有的接收器使用另一个移位时钟沿进行接收。主器件的 SSC 模块发出发送数据、移位时钟和可选的从选择信号。因此，波特率产生 (BRG) 和从选择产生 (SSG) 是主器件的一部分。帧控制对处于主模式和从模式的 SSC 模块是类似的，主要区别在于由哪个模块产生移位时钟和可选的从选择信号。

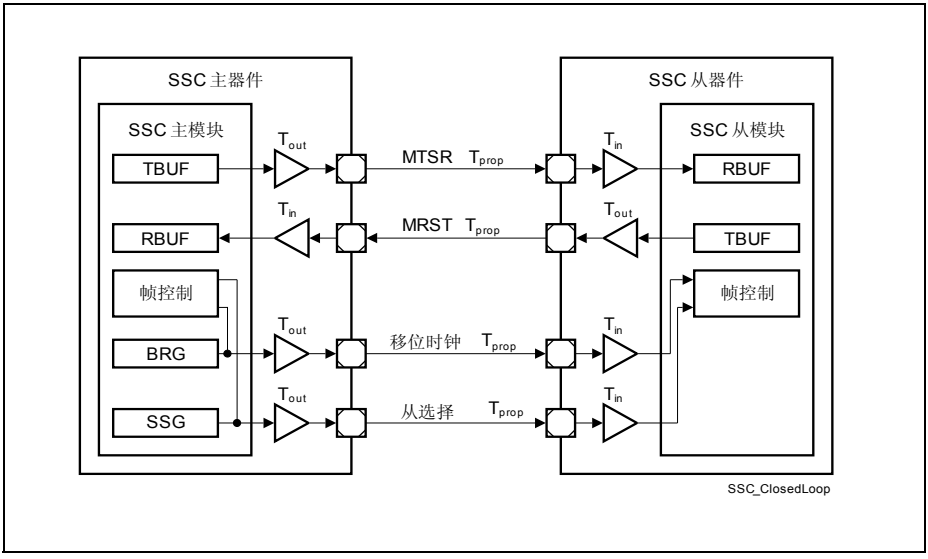


图 15-46 SSC 闭环延迟

主器件和从器件的 SSC 模块之间的信号通路包括主器件的输出驱动器、到从器件的连线和从器件的输入级。从器件在接收的移位时钟的边沿接收主机的发送数据，并通过一个类似的通路在另一方向将自己的数据发送给主器件。主机模块在其内部移位时钟边沿接收从机的发送数据。为了确保主器件接收到正确的数据，当主机用下一个移位时钟边沿 (一般为 1/2 个移位时钟周期) 接收数据时，从机的发送数据必须稳定的 (遵守建立和保持

时间) 。为了避免数据损坏，必须仔细考虑输入和输出级的累计延迟、导线上信号传播时间和发送器 / 接收器的反应时间，尤其是在波特率很高时。
在所给出的例子中，移位时钟信号产生与主器件 SSC 模块评估接收数据之间的时间由下面的和值给出：T_{out_master} + 2 x T_{prop} + T_{in_slave} + T_{out_slave} + T_{in_master} + 模块反应时间 + 输入建立时间。
输入通路的特点在于一个输入延迟主要取决于焊盘的输入级特性。输出通路延迟由输出驱动器延迟及其转换速率、外部负载和驱动器的电流能力决定。具体器件的输入 / 输出驱动器特性值在数据表中给出。

图 15-47 以图形方式描述了闭环延迟和两个延迟补偿选项的效果，这两个延迟补偿选项在 **15.4.6.2 节** 和 **15.4.6.3 节**中讨论。

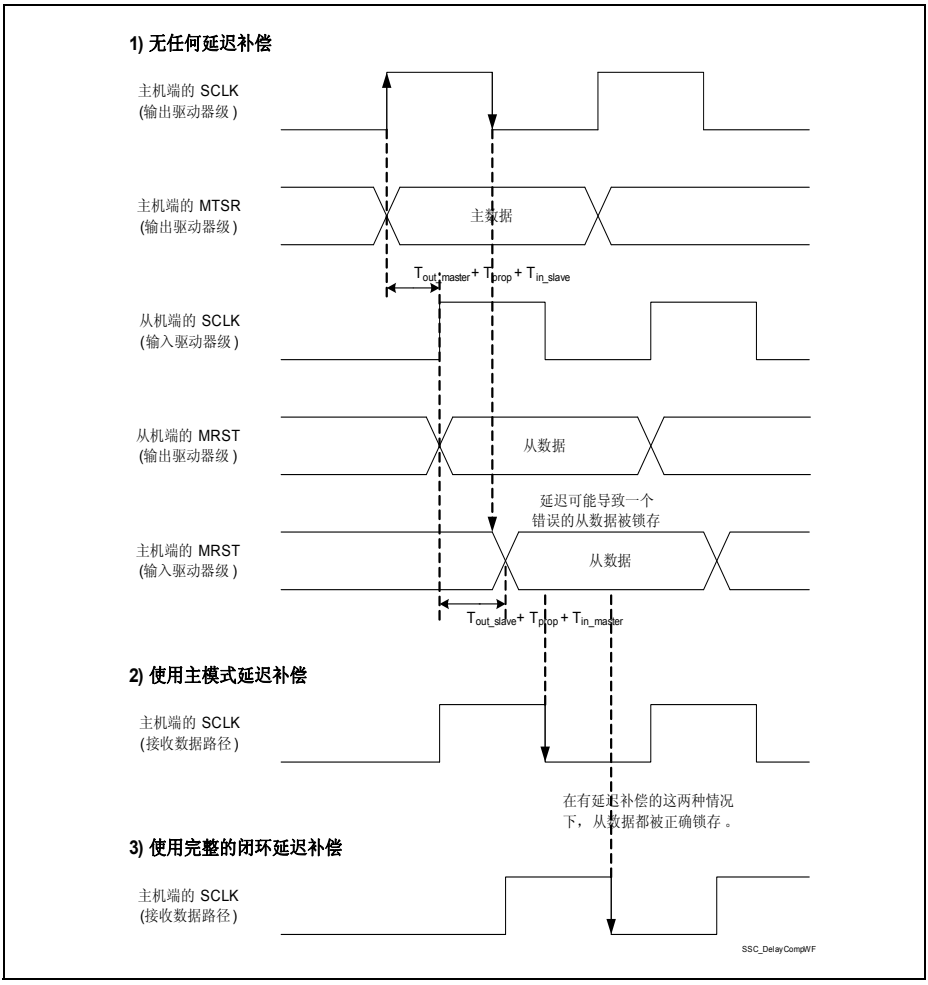


图 15-47　　SSC 闭环延迟时序波形

15.4.6.2　主模式下的延迟补偿

在主模式，通过延迟补偿可以达到更高的波特率。如果 (至少) 移位时钟引脚是双向的，就可以进行这种补偿。

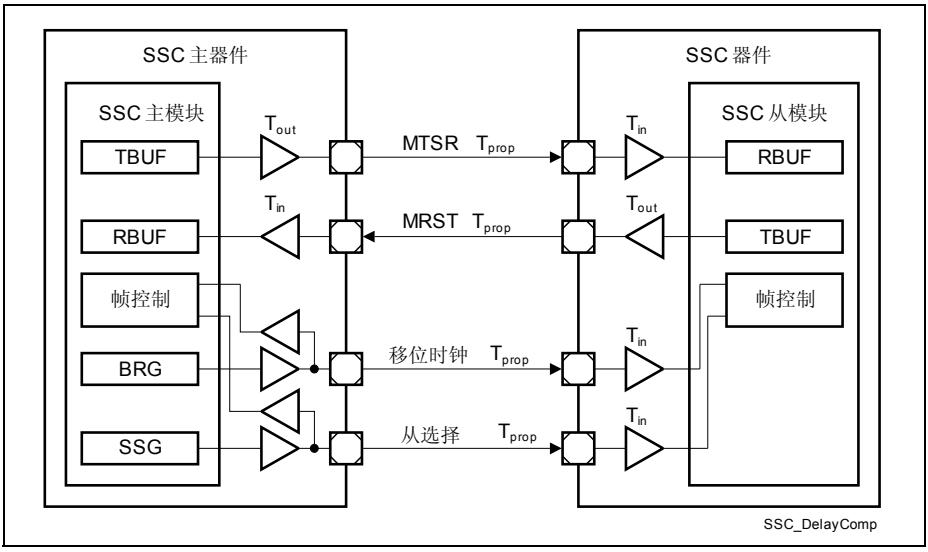


图 15-48　　带延迟补偿的 SSC 主模式

如果主模式下的接收移位时钟信号直接取自与输出信号并行的输入功能，则主器件的移位时钟输出的输出延迟是可补偿的，并且只需考虑主器件的输入延迟和从器件的输入延迟之间的差异，而不是考虑移位时钟路径上完整的主机输出延迟和从机输入延迟。当 DX1CR.INSW = 0 时 (发送移位时钟取自波特率生成器)，延迟补偿由 DX1CR.DCEN = 1 使能。

在所给出的例子中，评估移位时钟信号和主机 SSC 模块接收数据之间的时间减少了 $T_{in\_master} + T_{out\_master}$。

虽然是主模式，但移位时钟输入和可选的从选择信号并未在内部直接连接到数据移位单元，而是被当做来自输入引脚的外部信号。如果移位时钟输出引脚 (或从选择输出引脚，分别) 是双向的，则 SSC 通信的这种延迟补偿并不会导致额外的引脚。在这种情况下，输入信号被与其他内部信号分离，因为它与引脚本身的信号电平相关。

15.4.6.3　完整闭环延迟补偿

对于 SSC 通信，还可以通过使用 SSC 主器件和从器件上一个额外的引脚对完整闭环延迟进行补偿。

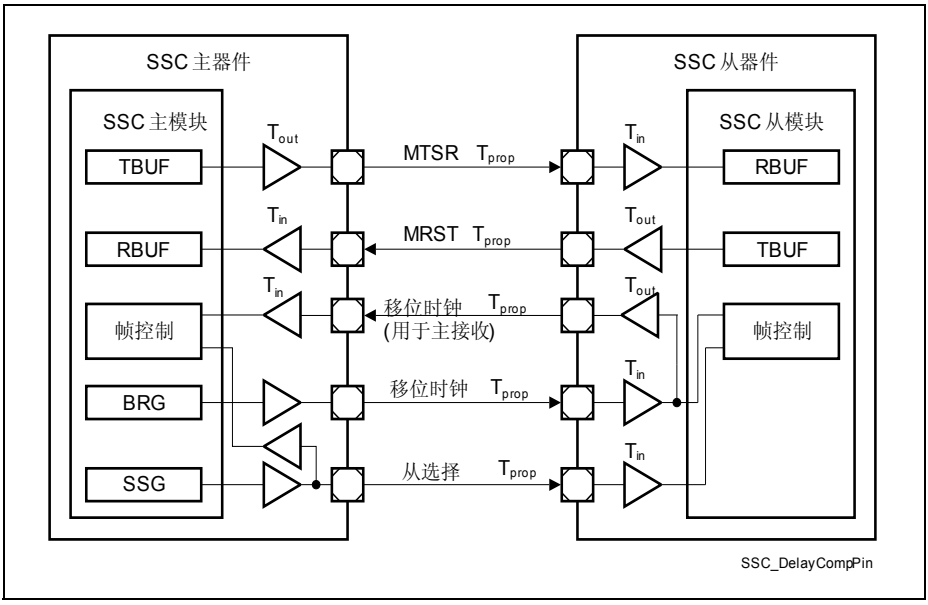


图 15-49　　SSC 完整闭环延迟补偿

这种延迟补偿方法的原理是从机将移位时钟反馈给主机，主机使用它作为接收移位时钟。通过让接收移位时钟经过一个完整的闭环信号通路，即可对接收移位时钟进行全面补偿。

从机必须通过置位 BRG.SCLKOSEL 来将 SCLKOUT 引脚功能设置为输出移位时钟，而主机必须使用 DX1CR.DCEN = 1 和 DX1CR.INSW = 0 来将 DX1 引脚功能设置为接收来自从机的移位时钟，并使能延迟补偿。

15.5 IC 间总线协议 (IIC)

USIC 的 IIC 协议参照 IIC 总线规范 [1]。与该规范不同的是，在所有模式中， USIC 器件假定总线信号的最大上升 / 下降时间为 300 ns。对于驱动能力，请参见 AC/DC 一章中的焊盘特性。不支持 CBUS 模式和 HS 模式。
IIC 模式通过设置 CCR.MODE = 0100_B 和 CCFG.IIC = 1(IIC 模式有效) 来选择。

15.5.1 简介

USIC IIC 特性：

- 两线接口，一根线用于移位时钟传输和同步 (移位时钟 SCL)，另一根线用于数据传输 (移位数据 SDA)。
- 标准模式 (100 kBit/s) 或快速模式 (高达 400 kBit/s) 通信
- 支持 7 位地址和 10 位地址
- 主模式操作
  IIC 控制总线事务并提供时钟信号。
- 从模式操作
  一个外部主机控制总线事务并提供时钟信号。
- 多主模式操作，
  总线上可以连接多个主机，可进行总线仲裁，即 IIC 模块可以是主机或从机。一个 IIC 总线参与者的主 / 从操作可以从帧到帧发生改变。
- 高效的帧处理 (低软件负荷)
- 强大的中断处理，因为有众多的指示标志
- 支持输入延迟补偿

15.5.1.1 信号说明

IIC 连接的特点在于使用两根信号线 (SDA 和 SCL)。这些信号的输出驱动器必须具有漏极开路特性，以允许通过线与方式将所有的 SDA 线连接在一起，将所有的 SCL 线连接在一起，从而形成一个 IIC 总线系统。由于这种结构的特点，由一个输出级驱动的高电平不一定立即导致在相应的输入出现高电平。因此，每个 SDA 或 SCL 连接必须在同一时刻输入和输出，因为输入功能一直监测信号的电平，在发送时也如此。

- 移位数据 SDA：输入由 DX0 级处理，输出信号 DOUT0
- 移位时钟 SCL：输入由 DX1 级处理，输出信号 SCLKOUT

图 15-50 示出了使用 USIC 的两个 IIC 总线参与者 (模块 IIC A 和 IIC B) 的一个连接。在这个例子中，模块 IIC A 的引脚分配显示 SDA 和 SCL 有分离的输入和输出引脚。对于相同的引脚，如果应用不提供具有 DOUT0 和一个 DX0 级输入，则可以使用该分配 (对 SCLKOUT 和 DX1 也类似)。模块 IIC B 的引脚分配显示 DOUT0 和 一个 DX0 输入连接在同一引脚，对 SCLKOUT 和一个 DX1 输入也是如此。

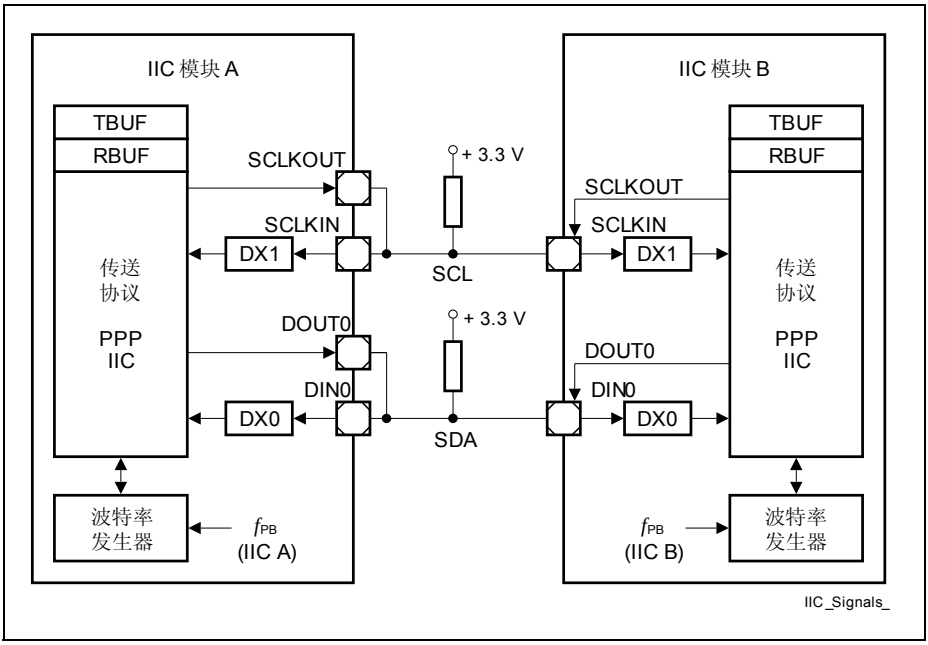


图 15-50　　IIC 信号连接

15.5.1.2　符号

一个符号是在线 SDA 和 SCL 上的一个边沿序列。符号包含 10 或 25 个时间量子 t_q，这取决于所选的波特率。波特率发生器决定时间量子 t_q 的长度，一个符号中的边沿序列由 IIC 协议预处理器处理。根据应用的需要，符号序列可以由用户编程。

以下是所定义的符号：

- 总线空闲：
  SDA 和 SCL 为高电平。当前未发生数据传送。
- 数据位符号：
  SDA 在 SCL 的高电平阶段稳定。SDA 代表传送的位值。对于每一个传送的数据位，都有一个时钟脉冲。在数据传送期间， SDA 只能在 SCL 为低电平时改变。
- 起始符号：
  在 SCL 为高电平期间，信号 SDA 从高电平变为低电平，表示一个起始条件。在总线为空闲时，该起始条件启动一次在 IIC 总线上的数据传送。
- 重复起始符号：
  当总线为非空闲时，该起始条件在一个数据符号之后启动一次在总线上的数据传送。因此， SDA 被置为高电平， SCL 被置为低电平，后面跟随一个起始符号。

- 停止符号：
  在 SCL 为高电平时， SDA 的上升沿指示一个停止条件。该停止条件终止一次数据传送，将总线释放为空闲状态。在一个启动条件和一个停止条件之间，可以传送任意数量的字节。

15.5.1.3 帧格式

数据是通过两线 IIC 总线 (SDA, SCL) 传送的，使用一个确保可靠和高效传送的协议。一个 (数据) 字节的发送者接收并检查紧随该字节之后的应答域的值。 IIC 是一个线与总线系统，一个以上器件上的 0 电平会导致总线上出现 0 电平，该电平可被所有器件接收。

一个数据字包含 8 个数据位符号作为数据值，后面跟随另一数据符号作为应答位。数据字可以被解译为地址信息 (在一个起始符号之后) 或传送的数据 (在地址之后)。

为了能接收一个应答信号，数据位的发送者必须通过一个发送一个 1 作为应答值来释放 SDA 线。根据接收器的内部状态不同，发送的应答位可以是主动电平或被动电平。

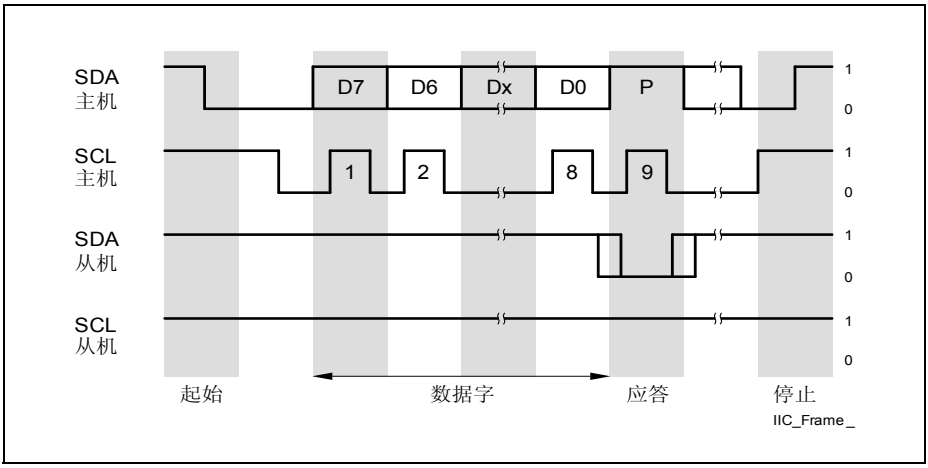


图 15-51　　IIC 帧示例 (简化的)

15.5.2 操作 IIC

为了操作 IIC 协议，必须考虑以下问题：

- 选择 IIC 模式：
  建议在 CCR.MODE = 0000_B 期间配置那些不能在运行时改变的所有 IIC 参数。应编程使位域 SCTR.TRM = 11_B 。 输入级的配置必须在 CCR.MODE = 0000_B 时完成，以避免意外的输入信号边沿，然后可以通过设置 CCR.MODE = 0100_B 来使能 IIC 模式。
- 引脚连接：
  建立输入级 DX0 (通过设置 DX0CR.DPOL = 0) 与所选移位数据引脚 SDA (信号 DIN0) 的连接 (设置 DX0CR.INSW = 0)，并配置发送数据输出信号 DOUT0 (通过设置

SCTR.DOCFG = 00_B) 到同一引脚。如果可用，该引脚对输入和输出可以是同一引脚，也可以将所选择的输入引脚和输出引脚连接在一起构成 SDA 线。
同样的机制也适用于移位时钟线 SCL。在此，SCLKOUT 信号 (通过设置 BRG.SCLKCFG = 00_B)和DX1级的一个输入必须连接(通过设置 DX1CR.DPOL = 0)。IIC 协议不使用输入级 DX2。
如果在 DX0/1 级使能了数字输入滤波器，为了正确地计算信号时序，必须考虑这些延迟。
用于 SDA 和 SCL 的引脚必须被设置为漏极开路模式，以支持 IIC 总线上信号线的线与结构。
需要注意的是，只应在使能了 IIC 模式之后再执行使能备用复用输出端口功能的步骤，以避免在输出信号线上出现意外的尖峰脉冲。

- 位时序配置：
  在标准模式 (100 kBit/s)，模块频率不能低于 2 MHz，而在快速模式 (400 kBit/s)，模块频率不能低于 10 MHz。此外，如果需要使用数字过滤器来消除最宽为 50 ns 的尖峰脉冲，则滤波器频率不能低于 20 MHz。
  如果另一个 IIC 参与者将总线上 SCL 的低电平阶段拉长，有可能使 SCL 高电平阶段时序 (最大值为 $1/f_{PPP}$) 具有不确定性。
  更详细的信息见 **15.5.3 节**。
- 数据格式配置：
  数据格式必须被配置为8个数据位(SCTR.WLE = 7)、无限数据流(SCTR.FLE = $3F_H$)、MSB 先移位 (SCTR.SDIR = 1)。奇偶校验位产生必须被禁止 (CCR.PM = 00_B)。
- 一般提示：
  如果被主机发送的地址选中，IIC 从模块会变为活动状态 (可接收或发送)。在从机向主机发送数据的情况下，它使用发送路径。因此，为了避免冲突，一个主机不得请求从为自身通道定义的从地址读取数据。
  内建的错误检测机制仅在 IIC 模块参加 IIC 总线通信时被激活。如果从器件无法处理过高的频率，它可以延长 SCL 信号的低电平阶段。
  对于符合 IIC 规范的数据传送，移位数据线 SDA 只应在 SCL = 0 期间改变 (由 IIC 总线规范定义)。

15.5.2.1 发送链

IIC总线协议在仲裁阶段和从器件处于发送状态期间要求一种位内响应(in-bit-response)。这种响应机制所导致的发送链环路延迟会限制可达到的最大波特率，这与总线特性 (总线负载、模块频率等) 密切相关。

图 15-50 示出了一般信号路径及一个从器件在发送情况下的延迟。由主器件产生的移位时钟 SCL 输出到 SCL 线上，然后通过输入级和输入滤波器。现在，可以检测到边沿并相应地产生 SDA 数据信号。SDA 信号通过输出级和导线后到达主器件的接收器部分。在此，它在通过输入级和输入滤波器之后被采样。

在 SCL 信号再次变化之前，这个完整的信息处理环 (包括为获得稳定的信号电平所需要的所有建立时间) 必须结束。在计算波特率时必须考虑该路径中的延迟 (波特率是 f_{PB} 和 f_{PPP} 的函数)。

15.5.2.2 字节展宽

如果一个器件被选择作为收发器并应该发送一个数据字节，但发送缓冲器 TBUF 中不包含有效的待发送数据，则该器件在前面的应答位结束后将 SCL 绑定在 0 电平。

15.5.2.3 主仲裁

在地址和数据发送期间，主发送器在 SCL 的上升沿检查每个数据位，看正在发送的值与在 SDA 线上读出的值是否相等。如果相等，下一个数据位值可为 0。如果不相等 (发送的值 = 1，读回的值 = 0)，该主器件在本次发送仲裁中失败。该事件由状态标志 PSR.ARL 指示，如果被 PCR.ARLIEN 使能，可产生一个协议中断。
当在发送仲裁中失败后，软件必须重新初始化该完整帧，即在起始条件后从发送第一个地址字节开始进行一次新的主发送尝试。仲裁也可发生在 ACK 位。

15.5.2.4 非应答和错误条件

在发生一个非应答或错误的情况下，TCSR.TDV 标志仍保持置位状态，但不会再继续发送。用户软件必须首先使发送缓冲器失效并禁止发送 (通过写 FMRL.MTDV = 10_B)，然后再用适当的值重新配置该发送操作 (通过写 TBUF) 作为对之前事件的反应。在使用 FIFO 数据缓冲区的情况下，还需要清空并重新填充 FIFO 缓冲区。

15.5.2.5 模式控制行为

在多主模式，只支持运行模式 0 和停止模式 0，不能将器件编程为其他模式。

- 运行模式 0：
  行为与编程设置的一样。如果 TCSR.TDV = 0 (未找到新的有效 TBUF 条目)，当有一个新的 TBUF 条目需要被处理时，IIC 模块等待 TDV 变为置位状态后再继续操作。
- 运行模式 1：
  行为与编程设置的一样。如果在主模式且 TCSR.TDV = 0 (未找到新的有效 TBUF 条目)，当有一个新的 TBUF 条目需要被处理时，IIC 模块发送一个停止条件以结束该帧。在从模式，与运行模式 0 没有区别。
- 停止模式 0：
  位 TCSR.TDV 在内部被视为 0 (该位本身不会被停止模式修改)。在主模式，当前运行的字正常结束，但新字尚未开始 (等待 TDV 有效)。
  对于主模式和从模式，位 TDV 都被视为 0 时，如果一个外部主机读取 TDV，则从机也会在总线上强加一个等待状态。
  此外，不可能在等待状态强制产生一个停止条件。原因在于，一次主机读传送必须以一个非应答后面跟随一个停止条件来结束，以允许从机释放其 SDA 线。否则，从机可能迫使 SDA 线为 0 (下一字节的第一个数据位)，使得不可能产生停止条件 (SDA 的上升沿)。
  要继续操作，必须将工作模式切换到运行模式 0。
- 停止模式 1：
  与停止模式 0 相同，但主机还要另外发送一个停止条件来结束当前帧。

如果主器件在一个 10 位地址的第一个字节之后请求了停止模式 1 ，则将发送一个停止条件。在这种情况下，从器件会产生一个错误中断。

15.5.2.6 数据传送中断处理

数据传送中断指示与 IIC 帧处理相关的事件。由于在 IIC 协议中数据输入和输出引脚是相同的，IIC 发送器也会在其输入引脚接收到输出数据。但是，在这种情况下不会产生与接收相关的中断。

- 发送缓冲器事件：
  发送缓冲器事件指示标志 PSR.TBIF 在发送缓存器 TBUF 的内容被加载到发送移位寄存器时置位，它指示由 TBUF 条目请求的操作已经开始。
  当该事件发生时，位 TCSR.TDV 被清零。该中断用于在当前 TBUF 条目正在进行时 (由发送器部分处理) 写下一个 TBUF 条目。
- 接收事件：
  该接收事件标志 PSR.RIF 指示一个新的数据字节已被写入到接收缓冲器 RBUF0/1 (一个新帧的第一个数据字节除外，该字节由备用接收中断指示)。当收到数据字节时，该标志被置位 (在 SCL 的下降沿后)。该中断可以用于在一个新数据字节正在进行时 (由接收器部分处理) 读取接收到的数据。
- 备用接收事件：
  备用接收事件标志 AIF 基于位 RBUFSR[9] (同 RBUF[9])，它指示接收的数据字是一个新数据帧的第一个数据字。
- 发送移位事件：
  发送移位事件指示标志 TSIF 在一个数据字节的最后一个数据位开始后被置位。
- 接收开始事件：
  接收开始事件指示标志 RSIF 在一个数据字节的第一个数据位的采样点之后被置位。

注： 如果在 IIC 数据传送期间不需要发送移位事件和接收开始事件，则可以将其忽略。

15.5.2.7 IIC 协议中断事件

在 IIC 模式可产生下述协议相关的事件，这些事件导致产生一个协议中断。

请注意，寄存器 PSR 中的位并不都由硬件自动清零，必须用软件清零，以监视新到来的事件。

- 在一个帧的正确位置接收到起始条件 (PSR.SCR)
- 在一个帧的正确位置接收到重复起始条件 (PSR.RSCR)
- 在一个帧的正确位置传送了停止条件 (PSR.PCR)
- 主仲裁失败 (PSR.ARL)
- 请求了从机读 (PSR.SRR)
- 收到应答 (PSR.ACK)
- 收到非应答 (PSR.NACK)
- 在一个帧的非预期位置检测到起始条件 (PSR.ERR)
- 在一个帧的非预期位置检测到停止条件 (PSR.ERR)

- 作为从机， 10 位地址在第一个地址字节之后被一个停止条件中断 (PSR.ERR)
- 主模式下的 TDF 从代码 (PSR.WTDF)
- 从模式下的 TDF 主代码 (PSR.WTDF)
- 发现保留的 TDF 代码 (PSR.WDTF)
- 起始条件代码 (PSR.WTDF)，在主模式下的一个运行帧期间
- 数据字节发送代码 (PSR.WTDF)，在主模式下传送方向已经变为接收 （主读）之后

如果在 TBUF 中发现了一个错误的 TDF 代码，则错误事件有效，直到 TDF 值被纠正或失效。如果相关的中断被使能，中断处理程序应在处理其他可能的中断事件之前首先检查 PSR.WDTF，并纠正 TBUF 或使其失效。

15.5.2.8 波特率发生器中断处理

波特率发生器中断指示捕获模式定时器已经达到其最大值。发生该事件时，位 PSR.BRGIF 被置 1。

15.5.2.9 接收器地址应答

在一个 (重复) 起始条件后，主机发送一个从地址来确定通信的目标器件。起始地址可以包含一个或两个地址字节 (分别对应 7 位或 10 位寻址方案)。在一个地址字节之后，对所发送的地址敏感的从机必须应答这次接收。

因此，从机的地址可以在器件中编程设置，以便与接收地址进行比较。在发生比较匹配的情况下，从机用一个应答 (SDA = 0) 来响应主机。非寻址目标的从机用一个非应答 (SDA = 1) 响应主机。
除了编程的地址匹配以外，如果从机能处理相应的请求，还必须用一个应答来响应另一个地址字节值。地址字节 00_H 表示通用呼叫地址，该地址也可以被应答。值 01_H 代表一个起始字节产生，这是不可应答的。

为了允许对地址字节的不同值进行选择性的应答， IIC 实现了以下控制机制：

- 如果位 PCR.ACK00 被置位，则地址字节 00_H 被应答。
- 地址字节 01_H 不被应答。
- 接收的第一个地址字节的前 7 位与编程的从地址 (PCR.SLAD[15:9]) 进行比较。如果这些位相符，从机发送一个应答。此外，如果从地址被编程为 1111 $0XX_B$，则从机等待第二个地址字节，将其与 PCR.SLAD[7:0] 进行比较，并根据 10 位寻址模式的要求相应地发送一个应答。用户必须注意保留地址 (更详细的说明请参见 IIC 规范)。仅支持地址 1111 $0XX_B$ 。

在上述每一种条件下，当发生地址匹配时，位 PSR.SLSEL 会被置 1。该位由一个 (重复) 起始条件自动清零。

15.5.2.10 接收器处理

一个被选中的从接收器总是会应答一个接收到的数据字节。如果接收器缓冲器 RBUF0/1 已满，且不能接收更多的数据，则响应的寄存器被覆盖 (在这种情况下， PSR.DLI 被置位并可能产生一个协议中断)。

在检查器件是否被选中之前，地址接收也使用寄存器 RBUF0/1 来存储地址。由于接收的地址并不置位 RDV0/1，因此地址不能像接收的数据那样处理。

15.5.2.11 接收器状态信息

除了接收的数据字节以外，一些 IIC 协议相关信息被存储在接收缓冲器的 16 位数据字中。接收的数据字节在 RBUF[7:0] 位置可用，而附加信息在 RBUF[12:8] 位置被监视。该结构允许确定接收的每个数据字节的含义，不需读取额外的寄存器，这对使用一个 FIFO 数据缓冲区的情况也是一样。

- RBUF[8]：
  接收的应答位的值。该信息在 RBUFSR[8] 中也可用，作为协议参数。
- RBUF[9]：
  该位为 1 指示在一个 (重复) 启动条件和紧随其后的地址接收之后，收到一个新帧的第一个数据字节。该位为 0 指示后续的数据字节。该信息在 RBUFSR[9] 中也可用，允许使用不同的中断例程对地址和数据进行处理。
- RBUF[10]：
  该位为0指示器件在从模式下收到数据字节，该位为1 指示在主模式下收到数据字节。
- RBUF[11]：
  该位为 1 指示在帧中一个错误位置发生的起始或停止条件导致在接收缓冲器中收到一个不完整 / 错误的数据字节。该位与 PSR 中的帧错误状态位不完全相同，因为 PSR 中的位必须由软件清零 (“ 粘滞 ” 位)，而 RBUF[11] 在每个数据字节都被求值。如果 RBUF[11] = 0，接收到的数据字节是正确的，与前面的错误无关。
- RBUF[12]：
  该位为 0 指示已经收到所编程的地址。该位为 1 指示收到通用呼叫地址。

15.5.3 符号时序

IIC 符号的时序由主机激励移位时钟线 SCL 确定。标准和快速 IIC 模式的符号时序是不同的。

- 100 k 波特的标准模式 (PCR.STIM = 0)：
  符号时序以每个符号占据 10 个时间量子 t_q 为基础。要求最小的模块时钟频率 f_{PB} = 2 MHz。
- 400 k 波特的标准模式 (PCR.STIM = 1)：
  符号时序以每个符号占据 25 个时间量子 t_q 为基础。要求最小的模块时钟频率 f_{PB} = 10 MHz。

只应在发送器或接收器为空闲或 CCR.MODE = 0 时改变波特率的设置。寄存器 BRG 中的位定义一个时间量子 t_q 的长度，由一个 f_{PCTQ} 周期给出。

- BRG.CTQSEL
  定义时间量子发生器的输入频率 f_{CTQIN}
- BRG.PCTQ
  定义一个时间量子的长度 (f_{CTQIN} 被 1、2、3 或 4 分频)
- BRG.DCTQ
  定义每个符号的时间量子数 (t_q 的数目 = DCTQ + 1)

标准设置由 CTQSEL = 00_B ($f_{CTQIN} = f_{PDIV}$) 和 PPPEN = 0 ($f_{PPP} = f_{IN}$) 给出。在这些条件下，频率 f_{PCTQ} 由下式给出：

$$f_{PCTQ} = f_{PIN} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \quad (15.10)$$

为了规定的 300 ns 的 SDA 保持时间，在信号 SCL 的下降沿之后引入了一个保持延迟 t_{HDEL} 。该延迟还可防止错误地检测到一个起始或停止条件。该延迟的长度可由位域 PCR.HDEL 编程。考虑到输入采样和输出更新，位域 HDEL 可以根据下面的公式编程：

$$\begin{aligned} HDEL &\geq 300\ ns \times f_{PPP} - \left(3 \times \frac{f_{PPP}}{f_{PB}}\right) + 1 \quad \text{使用数字滤波器且 } HDEL_{min} = 2 \\ HDEL &\geq 300\ ns \times f_{PPP} - \left(3 \times \frac{f_{PPP}}{f_{PB}}\right) + 2 \quad \text{不使用数字量且 } HDEL_{min} = 1 \end{aligned} \quad (15.11)$$

如果使用数字输入滤波器，则 HDEL 可在输入信号有尖峰脉冲的情况下补偿 2 个滤波器周期的滤波器延迟 (应使用 f_{PPP})。这会确保 SDA 线上的一个数据位恰好在 SCL 的上升沿之前或下降沿之后改变，而这样的变化不会被作为一个起始或停止条件处理。

15.5.3.1 起始符号

图 15-52 示出了一般起始符号的时序。

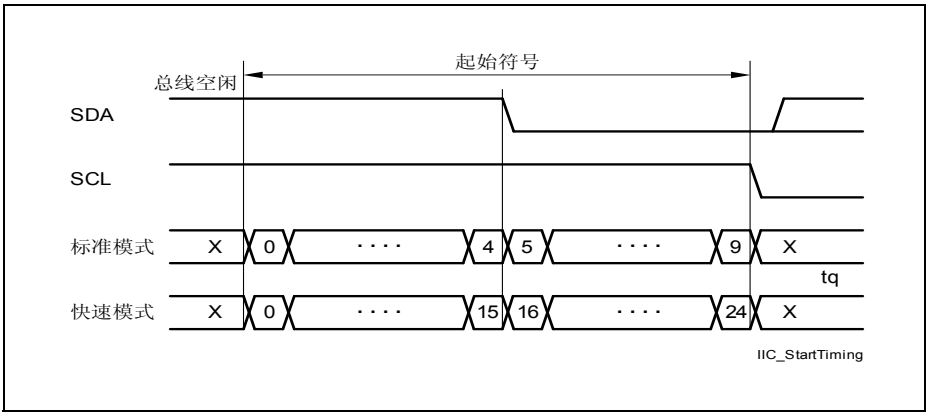


图 15-52　　起始符号的时序

15.5.3.2　重复起始符号

在一个重复起始符号的第一部分期间， SCL 被驱动为低电平，这段时间由指定的时间量子数定义。然后 SCL 输出一个高电平值。在检测到 SCL 输入出现上升沿之后，产生一个正常的起始符号，如 图 15-53 所示。

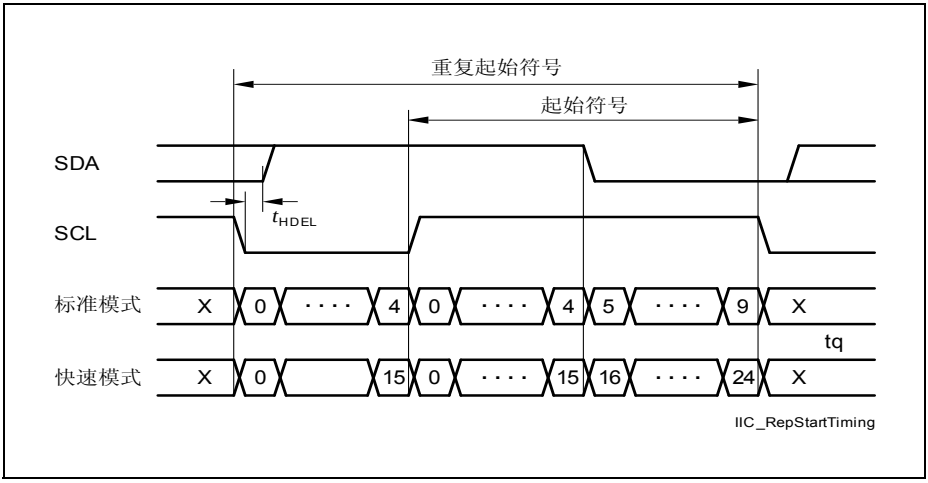


图 15-53　　重复起始符号的时序

15.5.3.3　停止符号

图 15-54 示出了停止符号的时序。

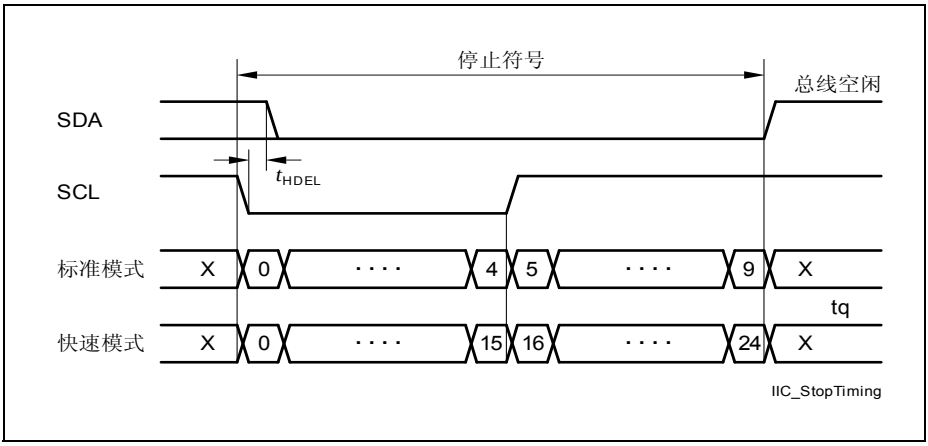


图 15-54　　停止符号的时序

15.5.3.4　数据位符号

图 15-55 示出了一般数据位符号的时序。

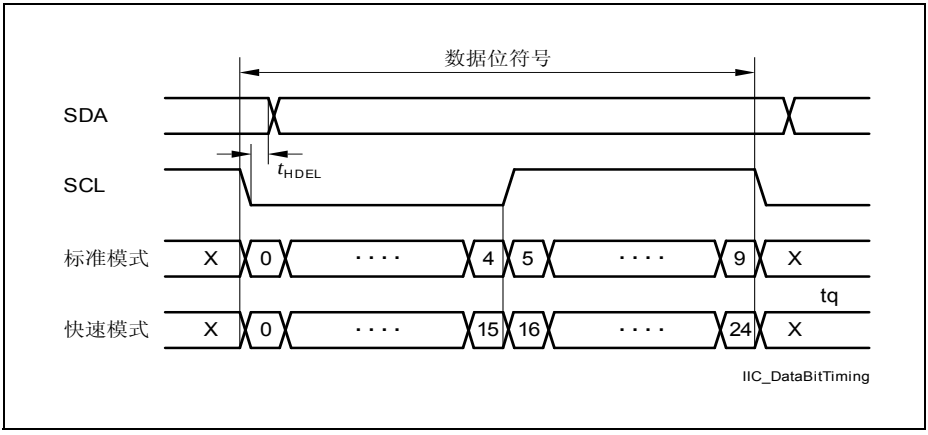


图 15-55　　数据位符号时序

如果在 SCL 输入检测到一个下降沿，则输出 SDA 在经过由 PCR.HDEL 定义的时间 t_{HDEL} 之后方可改变，以确保遵守 SDA 保持时间。 PCR.HDEL 的值允许对 SCL 输入路径 (采样、滤波) 的延迟进行补偿。

在进行一次应答发送的情况下， USIC IIC 等待接收器指示已接收到一个完整的字节。这给路径额外地增加了 3 个 f_{PB} 周期的延迟。必须正确地选择最小模块输入频率，以确保 SDA 相对于 SCL 上升沿的建立时间。

15.5.4 数据流处理

一个 IIC 帧中的数据流和符号序列的处理由 USIC 通信通道的 IIC 发送器部分控制。IIC 总线协议是面向字节的，而一个 USIC 数据缓冲器字可包含最多 16 个数据位。除了要发送的数据字节 (位于 TBUF[7:0]) 之外，用于控制 IIC 序列的位域 TDF (发送数据格式) 位于 TBUF[10:8]。TDF 代码定义应如何发送每个数据字节 (IIC 主机或 IIC 从机)，并控制 (重复) 起始和停止符号的发送。
该结构允许仅通过写 TBUFx 或使用一个 FIFO 缓冲区机制来为一个 IIC 主器件定义一个完整的 IIC 帧，因为不必访问其他控制寄存器。还可以选择对 PSR 寄存器中的 ACK 和 NACK 位进行查询，仅在收到一个 ACK 后发送下一个数据字节。

如果遇到一个错误或意外的 TDF 代码 (例如，因在数据缓冲区建立器件发生一个软件错误)，主机会发送一个停止条件。这会导致当前运行帧的中止。从机模块等待一个有效的 TDF 代码并设置 SCL = 0。然后，软件必须使这个意外的 TDF 代码失效，并写入一个有效的 TDF 代码。

请注意，在多主机总线系统的仲裁阶段期间，可能会由于一个意外的停止条件而引发一个不可预知的总线行为。

15.5.4.1 发送数据格式

在主模式，可使用以下发送数据格式：

表 15-12 主发送数据格式

TDF 代码	描述
000_B	**主机发送数据字节** 该格式用于从主机向一个从机发送一个数据字节。发送器发送其数据字节 (TBUF[7:0])，接收并检查由从机发送的应答位。
010_B	**接收数据字节并发送应答** 该格式用于由主机自从机读取一个数据字节。主机用一个 0 电平应答本次传送，以继续进行数据传送。 TBUF[7:0] 的内容被忽略。
011_B	**接收数据字节并发送非应答信号** 该格式用于由主机自从机中读取一个数据字节。主机用一个 1 电平对本次传送进行非应答以结束数据传送。TBUF[7:0] 的内容被忽略。
100_B	**发送起始条件** 在总线空闲时，如果 TBUF 包含该条目，则会产生一个起始条件。TBUF[7:0] 的内容被作为要发送的第一个地址字 (位 TBUF[7:1] 为地址， LSB 为读 / 写控制)。

表 15-12 主发送数据格式 (续表)

TDF 代码	描述
101_B	**发送重复起始条件** 在 TBUF 包含该项、SCL = 0 并且没有字节传送正在进行的情况下，如果器件是当前的主机，则会发送一个重复起始条件。为当前报文建立起始条件的器件被定义为当前主机。 TBUF[7:0] 的内容被作为要发送的第一个地址字节 (位 TBUF[7:1] 为地址， LSB 为读 / 写控制)。
110_B	**发送停止条件** 在当前主机已经结束了其最后一次字节发送 (包括应答) 的情况下，如果该格式在 TBUF 中，则它会发送一个停止条件。TBUF[7:0] 的内容被忽略。
111_B	**保留** 不得使用该代码。除了释放 TBUF 条目和置位 PSR 中的错误位 (可导致产生一个协议中断) 之外，不执行其他操作。

在从模式，可使用以下发送数据格式 (帧中的符号由主机控制，从机仅在被主机 “ 请求 ” 的情况下才必须发送数据)：

表 15-13 从机发送数据格式

TDF 代码	描述
001_B	**从机发送数据字节** 该格式用于自从机向主机发送一个数据字节 (TBUF[7:0])。发送器发送其数据字节加上一个值为 1 的应答位。

15.5.4.2 有效的主发送数据格式

由于 IIC 帧格式的定义，只有一些特定的 TDF 代码序列式是可能的和有效的。如果 USIC IIC 模块在一个正在运行的帧中检测到一个错误的 TDF 代码，则本次传送被中止并且标志 PCR.WTDF 被置位。此外，如果被用户使能，则会产生一个中断。在出现错误 TDF 代码的情况下，如果 USIC IIC 主机仍然占用 SDA 线，则会产生一个停止条件来立即中止当前帧。但是，如果被访问的从机占用 SDA 线 (读传送)，主机必须执行一次具有非应答的空读操作，以使从机在主机能发送一个停止条件前释放 SDA 线。空读操作所接收到的数据字节会被存储在 RBUF0/1，但 RDV0/1 不会被置位。因此，空读取不会产生一个接收中断，所读数据字节也不会被存储到接收 FIFO 中。

如果在当前帧 (主读访问) 中传送方向发生了改变，则发送数据请求 (TDF = 000_B) 是不可能的，并且不会被接受 (导致一个错误的 TDF 代码指示)。

表 15-14　有效 TDF 代码一览表

帧位置	有效 TDF 代码
第一个 TDF 代码 (主空闲)	起始 (100_B)
读传送：第二个 TDF 代码 (在起始或重复起始之后)	带有应答的接收 (010_B) 或带有非应答的接收 (011_B)
写传送：第二个 TDF 代码 (在起始或重复起始之后)	发送 (000_B)，重复起始 (101_B)，或停止 (110_B)
读传送：在应答之后的第三个以及后续的 TDF 代码	带有应答的接收 (010_B) 或带有非应答的接收 (011_B)
读传送：在非应答之后的第三个以及后续的 TDF 代码	重复起始 (101_B) 或停止 (110_B)
写传送：第三个以及后续的 TDF 代码	发送 (000_B)，重复起始 (101_B)，或停止 (110_B)

- 第一个 TDF 代码：
  一次主传送从 TDF 起始代码 (100_B) 开始。所有其他代码被忽略，但不会指示 WTDF 错误。
- 在一次读访问的情况下，在一个起始 (100_B) 或重复起始代码 (101_B) 之后的 TDF 代码：
  如果一次主读传送已经开始 (由地址字节的 LSB = 1 决定)，则 SDA 的传送方向发生改变，并且从机将主动驱动数据线。在这种情况下，只有代码 010_B 和 011_B 是有效的。要在发生一个错误代码的情况下中止传送，主机在能产生停止条件之前必须执行一次空读。
- 在一次写访问的情况下，在一个起始 (100_B) 或重复起始代码 (101_B) 之后的 TDF 代码：
  如果一次主写传送已经开始 (由地址字节的 LSB = 0 决定)，则主机仍然占用 SDA 线。在这种情况下，发送 (000_B)、重复起始 (101_B) 和停止 (110_B) 代码都是有效的。其他代码都被视为错误代码。要在发生一个错误代码的情况下中止传送，则立即产生停止条件。
- 在一次带有对前一数据字节的应答的读访问情况下，第三个以及后续命令的 TDF 代码：
  如果一次主读传送已经开始 (由地址字节的 LSB 决定)，则 SDA 的传送方向发生改变，并且从机将主动驱动数据线。为了迫使从机释放 SDA 线，主机必须对一次字节传送进行非应答。在这种情况下，只有接收代码 010_B 和 011_B 是有效的。要在发生一个错误代码的情况下中止传送，主机在能产生停止条件之前必须执行一次空读。
- 在一次带有对前一数据字节的非应答的读访问情况下，第三个以及后续命令的 TDF 代码：
  如果一次主读传送已经开始 (由地址字节的 LSB 决定)，则 SDA 的传送方向发生改变，并且从机将主动驱动数据线。为了迫使从机释放 SDA 线，主机必须对一次字节传送进行非应答。在这种情况下，只有起始 (100_B) 和停止代码 (110_B) 是有效的。要在发生一个错误代码的情况下中止传送，则立即产生停止条件。

- 在一次写访问的情况下，第三个以及后续命令的 TDF 代码：
  如果一次主写传送已经开始 (由地址字节的 LSB 决定)，主机仍然占用 SDA 线。在这种情况下，发送 (000_B)、重复起始 (101_B) 和停止代码 (110_B) 是有效的。其他代码被视为错误代码。要在发生一个错误代码的情况下中止传送，则立即产生停止条件。
- 主机在收到来自从机的一个非应答之后，会自动发送一个停止条件，除非随后的 TDF 代码请求一个重复起始条件。在这种情况下，只考虑该 TDF 代码，而其他所有 TDF 代码被忽略。

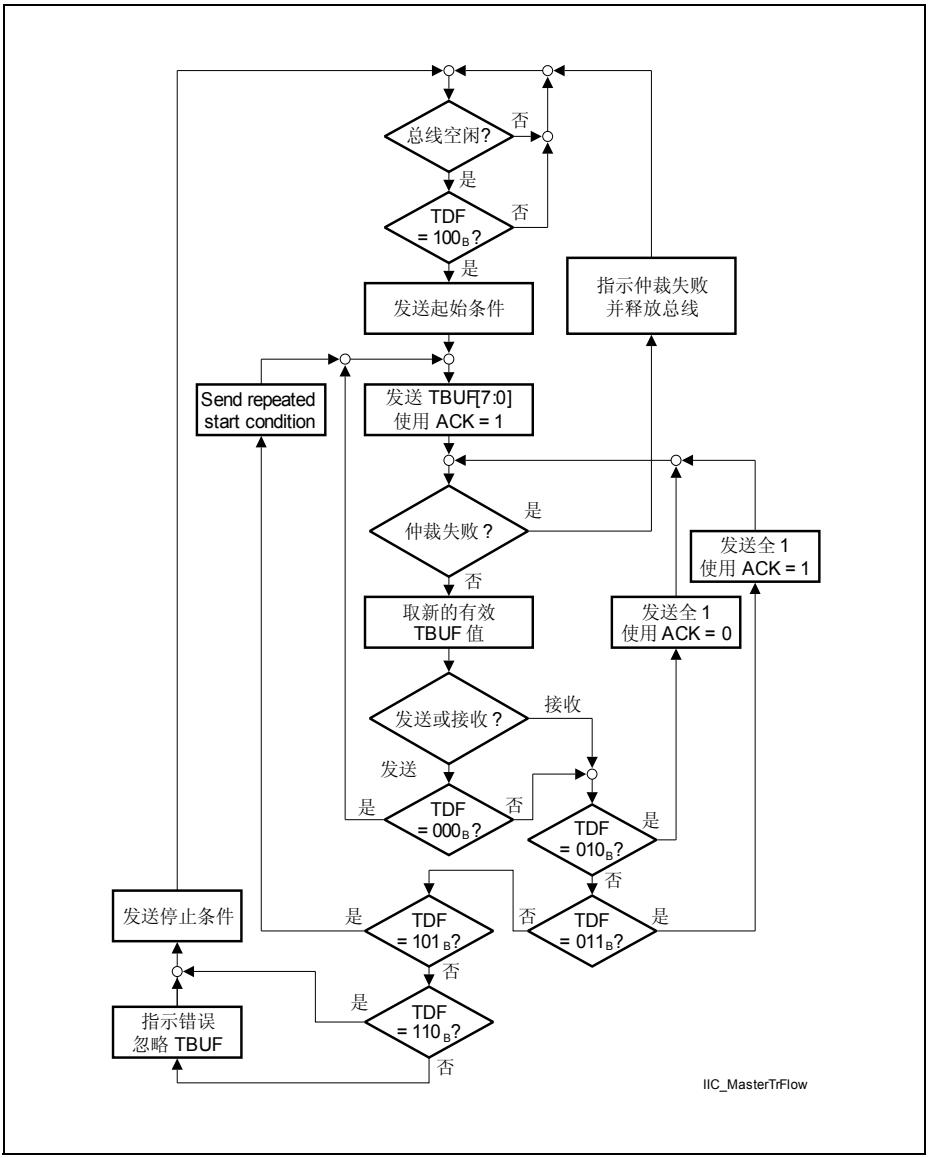


图 15-56　　IIC 主机发送

15.5.4.3 主发送 / 接收模式

在主发送模式，IIC 发送多个数据字节给从接收器。**表 15-15** 列出了主发送模式的 TDF 代码序列。

表 15-15 主发送模式 TDF 代码序列

TDF 代码序列	TBUF[10:8] (TDF 代码)	TBUF[7:0]	IIC 响应	中断事件
第 1 个代码	100_B	从机地址 + 写控制位	发送起始条件、从机地址和写控制位	SCR：指示检测到一个起始条件 TBIF：可向 TBUF 写下一个字
第 2 个代码	000_B	数据或第 2 个从机地址字节	发送数据或第 2 个从机地址字节	TBIF：可向 TBUF 写下一个字
用于数据发送的后续代码	000_B	数据	发送数据	TBIF：可向 TBUF 写下一个字
最后一个代码	110_B	无关	发送停止条件	PCR：指示检测到一个停止条件

在主接收模式，IIC 接收来自一个从发送器的多个数据字节。**表 15-16** 和**表 15-17** 列出了在 7 位和 10 位地址模式下进行主接收的 TDF 代码序列。

表 15-16 主接收模式 TDF 代码序列 (7 位地址模式)

TDF 代码序列	TBUF[10:8] (TDF 代码)	TBUF[7:0]	IIC 响应	中断事件
第 1 个代码	100_B	从机地址 + 读控制位	发送起始条件、从机地址和读控制位	SCR：指示检测到一个起始条件 TBIF：可向 TBUF 写下一个字
第 2 个代码	010_B	无关	接收数据和发送 ACK 位	TBIF：可向 TBUF 写下一个字 AIF：可以读取第一个接收到的数据
用于数据接收的后续代码	010_B	无关	接收数据和发送 ACK 位	TBIF：可向 TBUF 写下一个字 RIF：可以读取接收的后续数据

表 15-16　主接收模式 TDF 代码序列 (7 位地址模式)

TDF 代码序列	TBUF[10:8] (TDF 代码)	TBUF[7:0]	IIC 响应	中断事件
用于接收最后一个数据的代码	011_B	无关	接收数据和发送 NACK 位	TBIF：可向 TBUF 写下一个字 RIF：可以读取接收的最后一个数据
最后一个代码	110_B	无关	发送停止条件	PCR：指示检测到一个停止条件

表 15-17　主接收模式 TDF 代码序列 (10 位地址模式)

TDF 代码序列	TBUF[10:8] (TDF 代码)	TBUF[7:0]	IIC 响应	中断事件
第 1 个代码	100_B	从机地址 (第 1 个字节)+ 写控制位	发送起始条件、从机地址 (第 1 个字节) 和写控制位	SCR：指示检测到一个起始条件 TBIF：可向 TBUF 写下一个字
第 2 个代码	000_B	从机地址 (第 2 个字节)	发送地址 (第 2 个字节)	TBIF：可向 TBUF 写下一个字
第 3 个代码	101_B	第 1 个从机地址 + 读控制位	发送重复起始条件、从机地址 (第 1 个字节) 和读控制位	RSCR：指示检测到一个重复起始条件 TBIF：可向 TBUF 写下一个字
第 4 个代码	010_B	无关	接收数据和发送 ACK 位	TBIF：可向 TBUF 写下一个字 AIF：可以读取第一个接收到的数据
用于数据接收的后续代码	010_B	无关	接收数据和发送 ACK 位	TBIF：可向 TBUF 写下一个字 RIF：可以读取接收到的后续数据
用于接收最后一个数据接收的代码	011_B	无关	接收数据和发送 NACK 位	TBIF：可向 TBUF 写下一个字 RIF：可以读取接收到的最后一个数据
最后一个代码	110_B	无关	发送停止条件	PCR：指示检测到一个停止条件

图 15-57 示出了在主机发送 / 从机接收和主机接收 / 从机发送序列期间的中断事件。

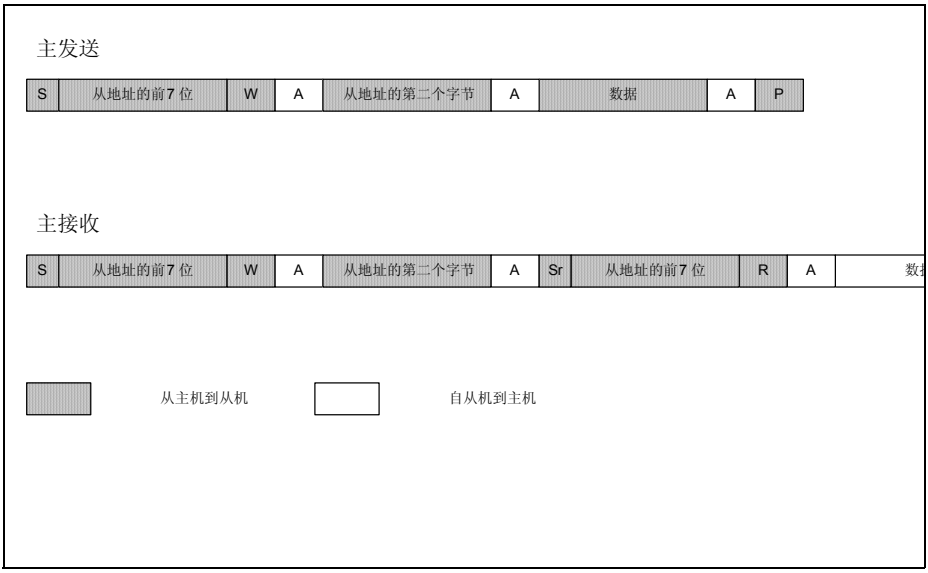


图 15-57　　数据传送中断事件

15.5.4.4　从发送 / 接收模式

在从接收模式，不需要写 TDF 代码，数据接收由备用接收事件 (AIF) 或接收事件 (RIF) 来指示。

在从发送模式，在收到自身的从机地址或者通用呼叫地址时，会触发从机读请求事件 (SRR)(如果该选项被使能)。然后 IIC 从机通过将 TDF 代码 001_B 和主机请求的数据写入 TBUF 来发送数据给主机。从机不检查主机对发送数据回复 ACK 还是 NACK。

在这两种情况下，主机都会发送一个停止条件来结束数据传送，该停止条件由一个 PCR 事件指示。参见 图 15-57。

15.5.5　IIC 协议寄存器

在 IIC 模式，寄存器 PCR 和 PSR 处理与 IIC 相关的信息。

15.5.5.1　IIC 协议控制寄存器

在 IIC 模式，寄存器 PCR 中的位或位域依本节所述定义。

PCR

协议控制寄存器 [IIC 模式] **($3C_H$)** **复位值：0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCLK	ACKIEN	HDEL				SACKDIS	ERRIEN	SRRIEN	ARLIEN	NACKIEN	PCRIEN	RSCRIEN	SCRIEN	STIM	ACK00
rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SLAD															
rw															

域	位	类型	描述
SLAD	[15:0]	rw	**从地址** 此位域包含所编程的从地址。第一个接收的地址字节中的相应位与 SLAD[15:9] 比较，以检查地址是否匹配。如果 SLAD[15:11] = 11110_B，则还要将第二个地址字节与 SLAD[7:0] 比较。
ACK00	16	rw	**应答 00_H** 该位定义一个从器件是否应该对从地址 00_H 敏感。 0_B 从器件对该地址不敏感 1_B 从器件对该地址敏感。
STIM	17	rw	**符号时序** 该位定义一个符号使用多少个时间量子。 0_B 一个符号包含 10 个时间量子。该时序适用于标准模式 (100 kBaud)。 1_B 一个符号包含 25 个时间量子。该时序适用于快速模式 (400 kBaud)。
SCRIEN	18	rw	**起始条件接收中断使能** 该位允许在检测到一个起始条件时产生协议中断 0_B 禁止起始条件中断。 1_B 使能起始条件中断。
RSCRIEN	19	rw	**重复起始条件接收中断使能** 该位允许在检测到一个重复起始条件时产生协议中断。 0_B 禁止重复起始条件中断。 1_B 使能重复起始条件中断。

域	位	类型	描述
PCRIEN	20	rw	**停止条件接收中断使能** 该位允许在检测到一个停止条件时产生协议中断。 0_B 禁止停止条件中断。 1_B 使能停止条件中断。
NACKIEN	21	rw	**非应答中断使能** 该位允许在主机检测到一个非应答时产生协议中断。 0_B 禁止非应答中断。 1_B 使能非应答中断。
ARLIEN	22	rw	**仲裁失败中断使能** 该位允许在检测到一次仲裁失败事件时产生协议中断。 0_B 禁止仲裁失败中断。 1_B 使能仲裁失败中断。
SRRIEN	23	rw	**从机读请求中断使能** 该位允许在检测到一个从机读请求时产生协议中断。 0_B 禁止从机读请求中断。 1_B 使能从机读请求中断。
ERRIEN	24	rw	**错误中断使能** 该位允许在检测到一个 IIC 错误条件 (由 PSR.ERR 或 PSR. WTDF 指示) 时产生协议中断。 0_B 禁止错误中断。 1_B 使能错误中断。
SACKDIS	25	rw	**从机应答禁止** 该位禁止为一个从器件产生有效应答信号 (有效应答 = 0 电平)。一旦由软件置位，该位会在每次 (重复) 起始条件产生时被自动清除。如果该位置 1 发生在收到一个字节后 (由一个中断指示) 字节后，但在下一个应答位开始之前，则下一个应答位将使用被动电平发送。这表示接收器不再接收更多的字节。因此，如果第一个接收中断用来置 1 该位，将会收到最少的 2 个字节。 0_B 允许产生一个有效从应答 (从应答为 0 电平 = 可接收更多字节)。 1_B 禁止产生一个有效从应答 (从应答为 1 电平 = 停止接收)。
HDEL	[29:26]	rw	**硬件延迟** 该位域用于补偿 SCL 信号的内部处理所带来的延迟 (参见 页 15-100)，以遵守 IIC 协议所规定的 SDA 保持时间。

域	位	类型	描述
ACKIEN	30	rw	**应答中断使能** 该位允许在主机检测到一个应答时产生协议中断。 0_B 禁止应答中断。 1_B 使能应答中断。
MCLK	31	rw	**主时钟使能** 该位使能主时钟 MCLK 的产生 (可用做通用频率输出，而不能直接用于 IIC 协议)。 0_B 禁止 MCLK 产生，并且 MCLK 为 0。 1_B 使能 MCLK 产生。

15.5.5.2 IIC 协议状态寄存器

下述 PSR 状态位或位域在 IIC 模式可用。请注意，寄存器 PSR 中的位不能由硬件清零。向 PSCR 寄存器中的一个位写 1 可清除 PSR 寄存器中相应的标志。向 PSR 寄存器中的一个位写 1 可置位相应的标志，但不会导致进一步的动作 (不会产生中断)。写 0 没有任何作用。应在使能一个新协议之前将这些标志清零。

PSR

协议状态寄存器 [IIC 模式] **(48_H)** **复位值：0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															**BRGIF**
r															rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	**RIF**	**TBIF**	**TSIF**	**DLIF**	**RSIF**	**ACK**	**ERR**	**SRR**	**ARL**	**NACK**	**PCR**	**RSCR**	**SCR**	**WTDF**	**SLSEL**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
SLSEL	0	rwh	**从选择** 该位指示该器件被选择作为从机。 0_B 该器件未被选择作为从机。 1_B 该器件被选择作为从机。

域	位	类型	描述
WTDF	1	rwh	**发现错误的 TDF Code 代码[1)]** 该位指示发现了一个意外 / 错误的 TDF 代码。如果 PCR.ERRIEN = 1，可产生一个协议中断。 0_B 未发现错误的 TDF 代码。 1_B 发现了错误的 TDF 代码。
SCR	2	rwh	**收到起始条件[1)]** 该位指示在 IIC 总线上检测到一个起始条件。如果 PCR.SCRIEN = 1，可产生一个协议中断。 0_B 未检测到起始条件。 1_B 检测到一个起始条件。
RSCR	3	rwh	**收到重复起始条件[1)]** 该位指示在 IIC 总线上检测到一个重复起始条件。如果 PCR.RSCRIEN = 1，可产生一个协议中断。 0_B 未检测到重复起始条件。 1_B 检测到一个重复起始条件。
PCR	4	rwh	**收到停止条件[1)]** 该位指示在 IIC 主线上检测到一个停止条件。如果 PCR.PCRIEN = 1，可产生一个协议中断。 0_B 未检测到停止条件。 1_B 检测到一个停止条件。
NACK	5	rwh	**收到非应答[1)]** 该位这指示在主模式下收到一个非应答。该位在从模式下不能被置 1。如果 PCR.NACKIEN = 1，可产生一个协议中断。 0_B 未收到非应答。 1_B 收到一个非应答。
ARL	6	rwh	**仲裁失败[1)]** 该位指示一次仲裁失败。如果 PCR.ARLIEN = 1，可产生一个协议中断。 0_B 冲裁未失败。 1_B 冲裁已失败。
SRR	7	rwh	**从机读请求[1)]** 该位指示检测到一个从机读请求。该位变为有效时将请求位于发送缓冲区中的第一个可用数据字节。为了请求更多的连续数据字节，发送缓冲区会发出更多的中断。为了结束数据传送，主发送器会发送一个停止条件。如果 PCR.SRRIEN = 1，可产生一个协议中断。 0_B 未检测到从机读请求。 1_B 检测到从机读请求。

域	位	类型	描述
ERR	8	rwh	**错误** [1)] 该位指示检测到一个 IIC 错误 error (帧格式或 TDF 代码) 。如果 PCR.ERRIEN = 1，可产生一个协议中断。 0_B 未检测到 IIC 错误。 1_B 检测到 IIC 错误。
ACK	9	rwh	**收到应答** [1)] 该位指示在主机模式下收到一个应答。该位在从模式下不能被置 1。如果 PCR.ACKIEN = 1，可产生一个协议中断。 0_B 未收到应答。 1_B 收到一个应答。
RSIF	10	rwh	**接收器起始指示标志** 0_B 未发生接收器起始事件。 1_B 发生了接收器起始事件。
DLIF	11	rwh	**数据丢失指示标志** 0_B 未发生数据丢失事件。 1_B 发生了数据丢失事件。
TSIF	12	rwh	**发送移位指示标志** 0_B 未发生发送移位事件。 1_B 发生了发送移位事件。
TBIF	13	rwh	**发送缓冲区指示标志** 0_B 未发生发送缓冲区事件。 1_B 发生了发送缓冲区事件。
RIF	14	rwh	**接收指示标志** 0_B 未发生接收事件。 1_B 发生了接收事件。
AIF	15	rwh	**备用接收指示标志** 0_B 未发生备用接收事件。 1_B 发生了备用接收事件。
BRGIF	16	rwh	**波特率发生器指示标志** 0_B 未发生波特率发生器事件。 1_B 发生了波特率发生器事件。
0	[31:17]	r	**保留** 读访问返回 0 ；在 IIC 模式不被修改。

1) 该状态位可产生一个协议中断 (见 页 15-17)。通用中断状态标志在通用中断一章中描述。

15.6 IC 间音频总线协议 (IIS)

本节描述 USIC 模块如何处理 IIS 协议。该串行协议可处理一个工作在模式的器件和一个工作在模式的器件之间的同步数据帧的接收和发送。一个基于 USIC 通信通道的 IIS 连接可支持全双工与半双工数据传送。通过设置 CCR.MODE = 0011_B 和 CCFG.IIS = 1(IIS 模式有效) 可选择 USIC 的 IIS 模式。

15.6.1 引言

IIS 协议是一种同步串行通信协议，它主要用于音频和娱乐应用 [2]。

15.6.1.1 信号描述

一个 IIS 主机和一个 IIS 从机之间的连接基于以下信号：

- 一个移位时钟信号 SCK.SCK。该信号在一个 IIS 连接建立之后一直由数据传送的主机产生，在没有有效数据位被传送时也不会停止。
- 一个字地址信号 WA(也称为 WS)。该信号由数据传送的主机产生。它指示一个新数据字的开始和目标音频通道 (例如左 / 右)。如果 WA 产生被使能 (通过设置 PCR.WAGEN = 1，对传送主机)，则字地址输出信号 WA 在所有 SELOx 输出可用。WA 信号在移位时钟的下降沿同步改变。
- 一个主发从收信号。如果发送器为 IIS 主器件，它产生这个主机发送从机接收数据信号。数据在移位时钟的下降沿同步改变。
- 一个主收从发信号。如果发送器为 IIS 从器件，它产生这个主机接收从机发送数据信号。数据在移位时钟的下降沿同步改变。

USIC 通信通道的发送器部分和接收器部分可一起使用，这样可以在一个 IIS 主器件和一个 IIS 从器件之间建立一个全双工数据连接。

表 15-18 IIS I/O 信号

IIS 模式	接收数据	发送数据	时钟	字地址
主模式	输入 DIN0，由 DX0 处理	输出 DOUT0	输出 SCLKOUT	输出 SELOx
从模式	输入 DIN0，由 DX0 处理	输出 DOUT0	输入 SCLKIN，由 DX1 处理	输入 SELIN，由 DX2 处理

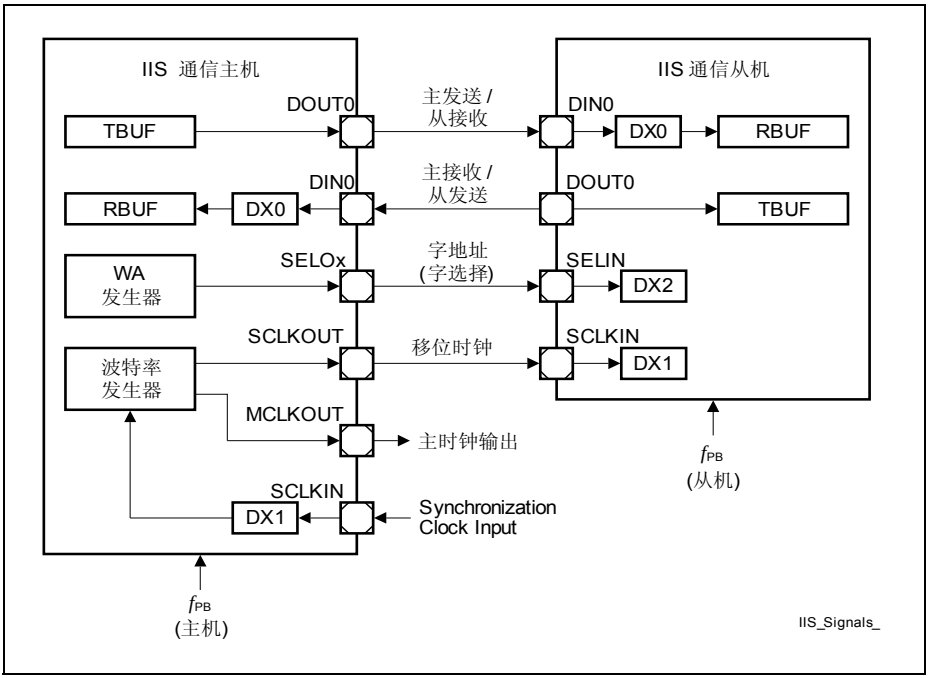


图 15-58　　IIS 信号

对于 USIC IIS 通信主器件，还有两个额外的信号可用：

- 一个主设备时钟输出信号 MCLKOUT。该信号与移位时钟有固定的相位关系，用于支持音频器件的过采样。也可将其用作具有同步IIS连接的一个通信网络的主时钟输出。
- 一个同步时钟输入信号　SCLKIN。该信号用于使移位时钟的产生与一个外部频率同步，以支持不能直接由通信主机的系统时钟 f_{PB} 产生的音频频率。也可将其用作具有同步 IIS 连接的一个通信网络的主时钟输入。

15.6.1.2　协议概述

一个 IIS 连接能通过同一数据线支持两种不同数据帧的传送，例如，一个数据帧用于左声道，另一个数据帧用于右声道。字地址信号 WA 用于区分不同的数据帧。每个数据帧可包含多个数据字。

在一个 USIC 通信通道中，要发送的数据字被标记为用于左声道还是右声道。当数据被接收时接收的数据字同样也包含一个可以识别 WA 状态的标记。

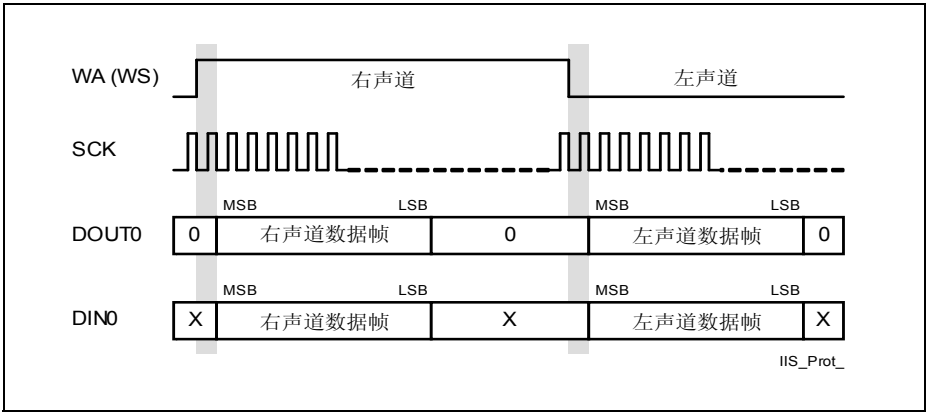


图 15-59　　协议概览

15.6.1.3　传送延迟

传送延迟特性允许数据传送 (发送和接收) 具有一个可编程的延迟 (以移位时钟周期计算)

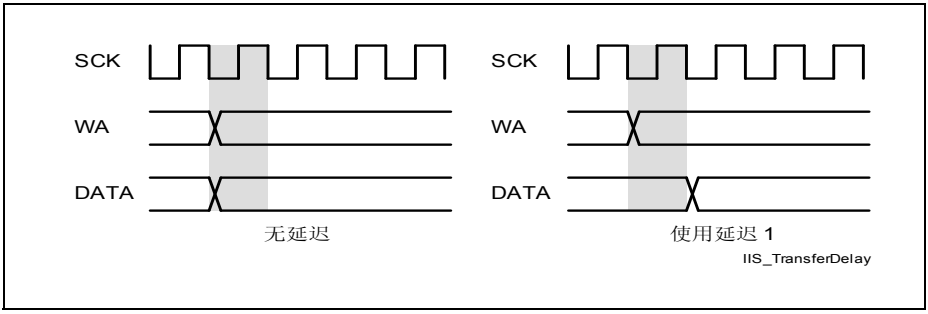


图 15-60　　IIS 传送延迟

15.6.1.4　外部音频器件的连接

IIS 信号可用于与外部音频器件 (例如编解码器) 或者其他音频数据源 / 目的进行通信。

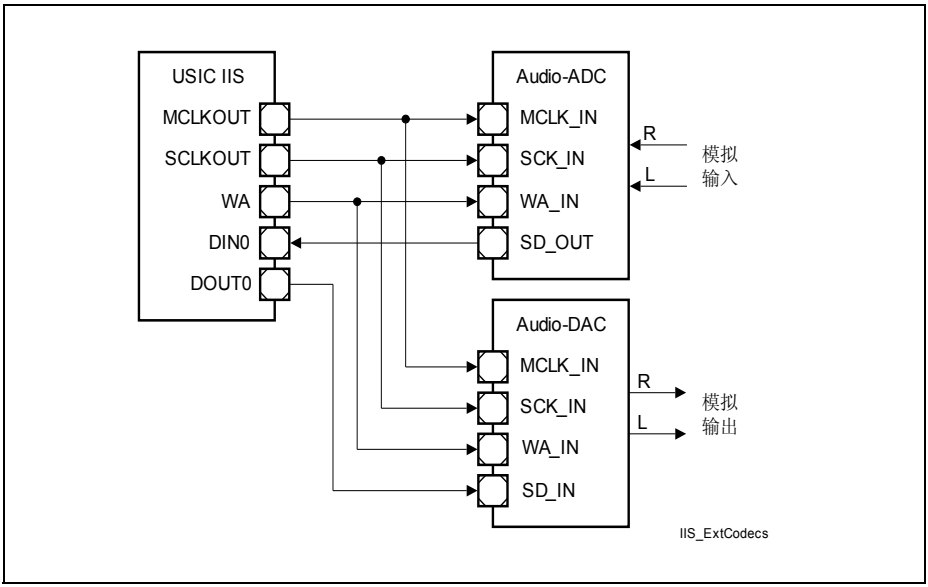


图 15-61　　外部音频器件的连接

在有些应用中，尤其是音频 ADC 或 DAC，要求使用一个与移位时钟信号具有固定相位关系的主时钟信号。MCLKOUT 信号的频率是移位频率 SCLKOUT 的整数倍。该倍数因子决定了外部器件的过采样因子 (一般取值为：256 或 384)。

15.6.2　操作 IIS

本节包含 IIS 的一般性问题，这些问题不与主模式或者从模式直接关联。

15.6.2.1　帧长与字长配置

在 WA 信号每次发生改变之后，就会有一个完整的数据帧被传送 (帧长小于或等于系统字长)。信号 WA 变化之后传送的数据位数由 SCTR.FLE 定义。一个数据帧可包含多个数据字，一个数据字的长度由 SCTR.WLE 定义。WA 信号的改变将系统字长定义为两次 WA 信号改变之间的 SCLK 周期数 (右通道的可用位数和相同的左通道可用位数)。

如果系统字长大于由 SCTR.FLE 定义的帧长度，则使用被动数据电平发送 (SCTR.PDL) 发送额外的位。如果系统字长比器件的帧长度短，则不能将发送数据的所有 LSB 都传送出去。

建议将寄存器 TCSR 中的位 WLEMD、 FLEMD 和 SELMD 都编程为 0。

15.6.2.2 自动映射机制

在通过一种自动映射机制传送一个数据帧时，波特率和移位控制设置在内部保持不变。可以在任何时刻将这些寄存器编程为新的设置，这些新设置在下一个数据帧生效。在一个数据帧期间，虽然在数据帧开始后写入了新设置值，但所使用的 (映射的) 设置并未改变。在每个数据帧开始后，之前写入的设置在内部被 “ 冻结 ”。

尽管器件实现了这种自动映射机制，但还是建议只在 IIS 协议被关闭时改变波特率和移位控制设置。

15.6.2.3 模式控制行为

IIS 模式支持以下内核模式：

- 运行模式 0/1：
  行为与编程设置的一样，不影响数据传送。
- 停止模式 0/1：
  位 PCR.WAGEN 在内部被视为 0(位本身不会改变)。如果 WAGEN = 1，则当前的系统字周期结束， WA 产生也随后被停止，但 PSR.END 位不会被置 1。在进入停止模式之前，完整的数据帧已经结束，包括由 PCR.TDEL 导致的可能的延迟。
  当通过设置 WAGEN = 1 而脱离停止模式时， WA 产生重新开始。

15.6.2.4 传送延迟

传送延迟可用于使一次数据传送与一个事件 (例如 WA 信号的改变) 同步。该事件必须在移位时钟 SCK 的下降沿同步产生 (与发送数据的改变类似)，因为该事件的输入信号在接收器中被直接采样 (其结果是，发送器可使用其下一边沿的检测信息)。

当移位时钟正在运行时，不得使用与移位时钟异步的事件信号。在图 15-60 所示的例子中，事件 (WA 信号改变) 由传送主器件产生，因此与移位时钟 SCK 信号同步。在 SCK 的上升沿，WA 信号被采样并被检查是否发生改变。如果检测到一个变化，传送延迟计数器 TDC 会被自动加载为其可编程的重载值 (PCR.TDEL)。否则，该计数器会在每个 SCK 的上升沿减 1，直到达到 0 后停止计数。数据传送在 TDC 的值变为 0 开始。这种情况会在下面两个条件下发生：

- 当检测到该事件时， TDC 被重新加载为 PCR.TDEL = 0。
- 在向下计数时 TDC 达到 0。

传送延迟计数器位于 IIS 协议预处理器内部，不能通过软件查看。传送延迟以 SCK 周期为单位，由 PCR.TDEL+1 给出。

在图 15-62 所示的例子中，TDC 的重载值 PCR.TDEL 为 0。当接收器端的采样值表现出 WA 信号改变时，计数器 TDC 被重新加载。如果重载值为 0，数据传送在 WA 改变后延迟 1 个移位时钟周期再开始。

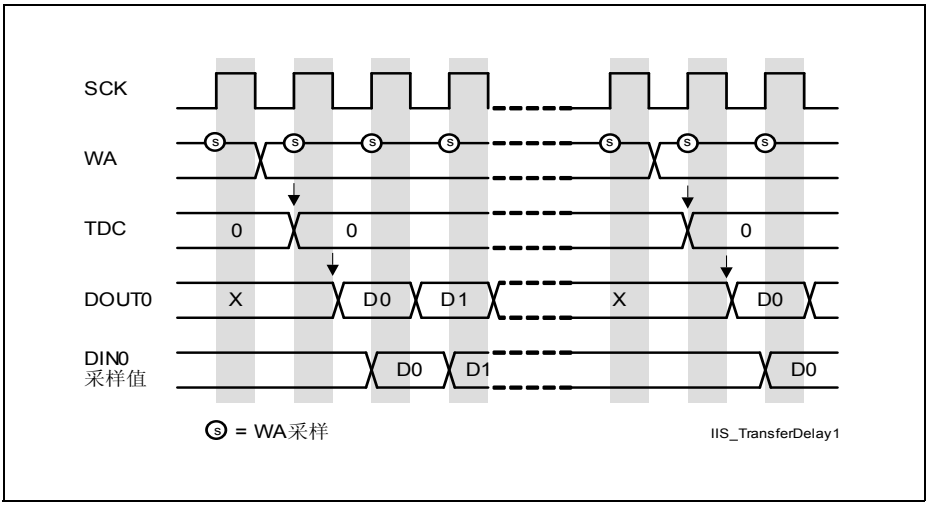


图 15-62 1 个移位时钟周期的传送延迟

没有任何传送延迟的理想情况如**图 15-63** 所示。在 WA 信号变化的同时数据输出值变为有效。这意味着发送器预先“知道”该事件信号将在下一个 TCLK 上升沿发生改变。这可通过在检测到前一 WA 变化后将系统字长减 1，从而延迟数据传送来实现。

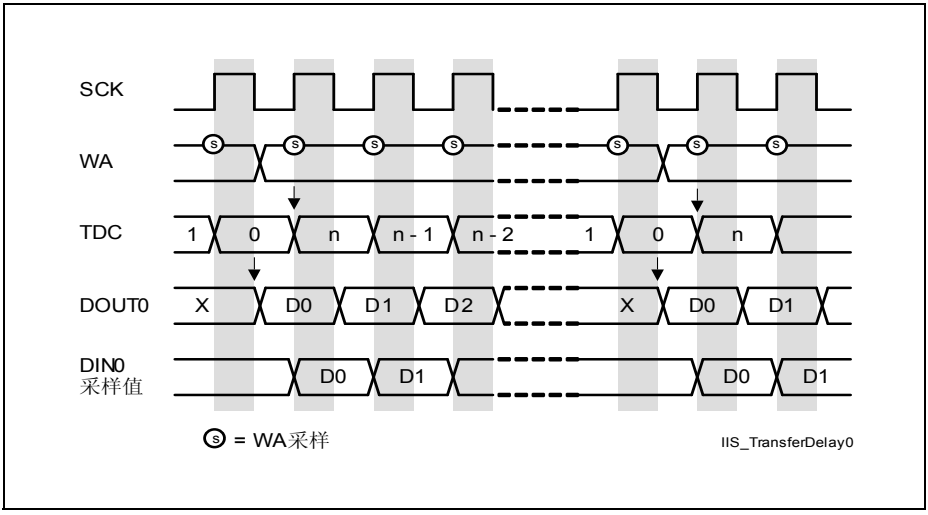


图 15-63 无传送延迟

如果在检测到传送延迟结束时改变 WA，则传送开始，延迟计数器被重新加载为 PCR.TDEL。这样就允许在将 USIC 作为 IIS 器件运行时不带任何延迟。在这种情况下，来自前一事件的内部延迟在一个新事件发生时恰好结束。如果 PCR.TDEL 被设置为一个大于系统字长的数值，则不会发生数据传送。

15.6.2.5 奇偶校验模式

IIS 模式不支持奇偶校验位的产生，必须编程使位域 CCR.PM = 00_B 。

15.6.2.6 传送模式

在IIS模式，必须编程使位域SCTR.TRM = 11_B，以允许数据传送。设置 SCTR.TRM = 00_B 将立即禁止和停止数据传送。

15.6.2.7 数据传送中断处理

数据传送中断指示与 IIS 帧处理相关的事件。

- 发送缓冲器中断 TBI：
  位 PSR.TBIF 在一个数据字的第一个数据位开始后被置 1。
- 发送移位中断 TSI：
  位 PSR.TSIF 在一个数据字的最后一个数据位开始后被置 1。
- 接收器启动中断 RSI：
  位 PSR.RSIF 在接收器收到一个数据字的第一个数据位后被置 1。
  发生事件时，位 TCSR.TDV 不会被清零，新数据会被加载到发送缓冲器。
- 接收中断 RI 与备用中断 AI：
  在 WA = 0 的情况下，位 PSR.RIF 在接收器收到一个数据字的最后一个数据位后被置 1。位 RBUFSR.SOF 指示所接收的数据字是否为一个新数据帧的第一个数据字。
  在 WA = 1 的情况下，位 PSR.AIF 在接收器收到一个数据字的最后一个数据位后被置 1。位 RBUFSR.SOF 指示所接收的数据字是否为一个新数据帧的第一个数据字。

15.6.2.8 波特率发生器中断处理

波特率发生器中断指示捕获模式定时器已经达到其最大值。该事件发生时，位 PSR.BRGIF 被置 1。

15.6.2.9 协议相关的参数和错误

为了区分接收到的数据字用于左声道还是右声道，IIS 协议预处理器采样 WA 输入的电平 (恰好在 WA 变化之后)，并将其作为协议相关错误 (尽管它不是错误，却是一个指示) 传送到接收缓冲器状态寄存器的位 RBUFSR[9]。该位决定在收到一个新数据字时是激活标准接收中断 (如果 RBUFSR[9] = 0) 还是激活备用接收中断 (如果 RBUFSR[9] = 1)。如果相应的事件被定向到不同的中断节点，则左声道和右声道的输入数据由不同的中断处理。标志 PAR 总是为 0。

15.6.2.10 发送数据处理

IIS 协议预处理器允许区分左声道和右声道的数据发送。因此，位 TCSR.WA 指示缓冲区的数据将被发送到哪一个声道。如果 TCSR.WA = 0，数据将在 WA 的一个下降沿之后被发送。如果 TCSR.WA = 1，数据将在 WA 的一个上升沿之后被发送。在 WA 变化后所采样的 WA 值用于区分两个通道 (参见 PSR.WA)。

对于每一个数据字，如果 TCSR.WAMD = 1，位 TCSR.WA 可被发送控制信息 TCI[4] 自动更新。在这种情况下，写入到 TBUF[15:0] 的数据 (或 IN[15:0]，如果使用一个 FIFO 数据缓冲区) 被视为左声道数据，而写入到 TBUF[31:16] 的数据 (或 IN[31:16]，如果使用一个 FIFO 数据缓冲区) 被视为右声道数据。

15.6.2.11 接收缓冲区处理

如果有接收 FIFO 缓冲区可用 (CCFG.RB = 1) 并且被使能用于数据处理 (RBCTR.SIZE > 0)，建议在 IIS 模式设置 RBCTR.RCIM = 11_B。这会导致：位 OUTR.RCI[0] = 1 指示该数据字为一个新数据帧的第一个数据字；通道由采样的 WA 值指示，在 OUTR.RCI[4] 中给出。

在 RCI 模式 (RBCTR.RNM = 1)，标准接收缓冲区事件和备用接收缓冲区事件可用于以下操作：

- 标准接收缓冲区事件指示可以从 OUTR 中读取一个数据字，该数据字所属于的一个数据帧在 WA = 0 时开始。
- 备用接收缓冲区事件指示可以从 OUTR 中读取一个数据字，该数据字所属于的一个数据帧在 WA = 1 时开始。

15.6.2.12 环路延迟补偿

IIS 协议的同步信令机制与 SSC 协议类似，应用设置必须考虑环路延迟。在 IIS 模式，主模式下的环路延迟补偿允许获得更高的波特率。

更详细的信息请参见 SSC 一章中的描述。

15.6.3 操作主模式下的 IIS

为了操作主模式下的 IIS，必须考虑以下问题：

- 选择 IIS 模式：
  建议在 CCR.MODE = 0000_B 期间配置那些不能在运行时改变的所有 IIS 参数。应编程使位域 SCTR.TRM = 11_B。输入级的配置必须在 CCR.MODE = 0000_B 时完成，以避免意外的输入信号边沿，然后可以通过设置 CCR.MODE = 0011_B 来使能 IIS 模式。
- 数据传送的引脚连接：
  通过设置 DX0CR.INSW = 1 来建立输入级 DX0 与所选接收数据输入引脚 (DIN0) 的连接。为发送器配置一个发送数据输出引脚 (DOUT0)。
  数据移位单元允许基于同一 WA 信号进行全双工数据传送，DX0 级提供的值被视为数

据位 (不能独立于发送器来禁止接收功能)。如果只是为了接收 IIS 数据，则发送器配置不是必须的 (不为 DOUT0 信号分配引脚)。

- 波特率产生：
  必须选择所期望的波特率设置，该设置包括分数分频器和波特率发生器。必须编程使位 DX1CR.INSW = 0 ，以利用波特率发生器输出 SCLK 直接作为数据移位单元的输入。将移位时钟输出引脚配置为 SCLKOUT 的反向信号不会带来额外的延迟 (BRG.SCLKCFG = 01_B)。
- 字地址 WA 的产生：
  必须通过设置 PCR.WAGEN = 1 来使能 WA 的产生，并编程设置两次 WA 变化之间的移位时钟周期数。必须编程使位 DX2CR.INSW = 0，以使用 WA 发生器作为数据移位单元的输入。如果需要，还要为信号 SELOx 配置 WA 输出引脚。
- 数据格式配置：
  必须根据应用要求通过编程寄存器 SCTR 来设置字长、帧长和移位方向。通常是 MSB 最先移位 (SCTR.SDIR = 1)。可使用发送控制信息 TCI[4] 来设置位 TCSR.WAMD，以区分在 WA = 0 或 WA = 1 时要发送的数据字。

注： 为了避免在输出端出现意外的尖峰脉冲，必须在 IIS 模式被使能后再执行使能备用输出端口功能的步骤。

15.6.3.1 波特率产生

波特率由 SCLK 信号的频率定义 (一个 f_{SCLK} 周期代表一个数据位)。

如果使用分数分频器模式产生 f_{PIN}，则对于 f_{PIN} 来说会有一个 f_{PB} 周期的不确定性。这种不确定性不会在多个 SCLK 周期累积。因此，可达到平均频率，但 SCLK 和 MCLK 信号的 50% 占空比会有一个 f_{PB} 周期的变化。

在 IIS 应用中，可选的 MCLK 输出信号与 SCLK 之间没有固定的相位关系。 SCLK 可以基于 f_{PIN} 频率 (BRG.PPPEN = 0)。在这种情况下，MCLK 信号与 SCLK 之间需要有固定的相位关系 (例如：当使用 MCLK 作为外部器件的时钟参考时)。必须使用额外的 2 分频器 (BRG.PPPEN = 1)。使用该 2 分频器是因为 MCLK 信号在每个 f_{PIN} 周期切换一次。此后 SCLK 信号即以 MCLK 信号为基础产生，见 ***图 15-64***。

可调节的整数分频因子通过位域 BRG.PDIV 定义。

$$f_{SCLK} = \frac{f_{PIN}}{2} \times \frac{1}{PDIV + 1} \quad \text{如果 PPPEN = 0}$$
$$f_{SCLK} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} \quad \text{如果 PPPEN = 1} \qquad (15.12)$$

注： 在 IIS 协议中，主器件 (产生移位时钟和 WA 信号) 在 SCK 的下降沿改变其数据和 WA 输出线的状态。从发送器也必须在下降沿发送。接收数据的采样在 SCLK 的上升沿完成。输入级 DX1 和 SCLKOUT 必须被编程为将移位时钟信号反相，以适合内部信号。

15.6.3.2 WA 产生

字地址 (或字选择) 线 WA 定期地在 N 个 SCLK 信号周期之后切换。两次 WA 变化之间的时间被称为系统字长，可以通过使用下面的位域编程。

在 IIS 主模式，系统字长定义如下：

- BRG.CTQSEL = 10_B
  基于 SCLK 进行 WA 切换
- BRG.PCTQ
  定义每个系统字长的 SCLK 周期数 N
- BRG.DCTQ
  定义每个系统字长的 SCLK 周期数 N

$$N = (PCTQ + 1) \times (DCTQ + 1) \quad (15.13)$$

15.6.3.3 主时钟输出

主时钟信号 MCLK 可由 IIS 传送的主器件产生 (BRG.PPPEN = 1)。该信号特别用于连接外部编码器件。其极性由位 BRG.MCLKCFG 配置，最后变为输出信号 MCLKOUT。

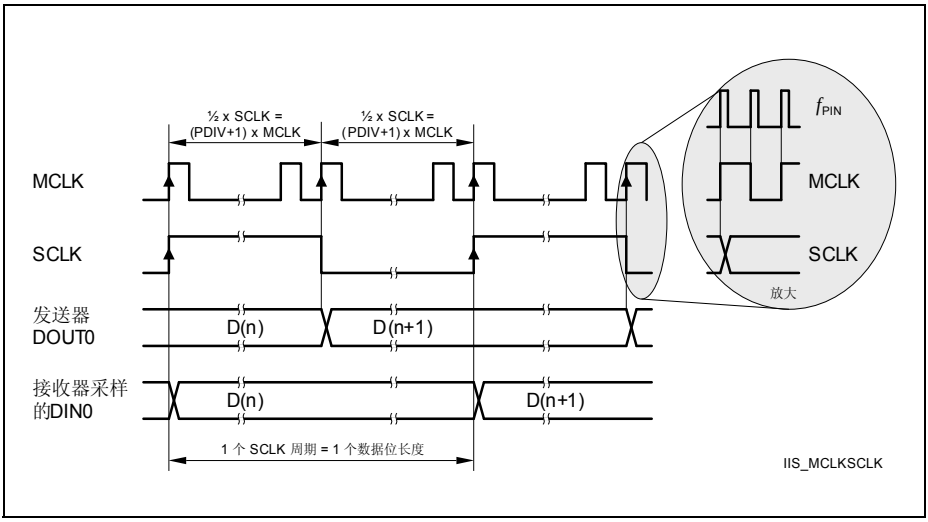


图 15-64　　IIS 的 MCLK 和 SCLK

15.6.3.4　协议中断事件

在 IIS 主模式可产生下述协议相关的事件，这些事件可导致产生一个协议中断。

请注意，寄存器 PSR 中的位并不都是由硬件自动清零的，为了监视新到来的事件，必须用软件清零。

- WA 上升沿 / 下降沿事件：
  WA 产生块指示两个在寄存器 PSR 中被监视的事件。标志 PSR.WAFE 在 WA 信号的下降沿被置位，标志 PSR.WARE 上升沿被置位。如果 PCR.WAFEIEN = 1，会产生一个针对下降沿的协议中断；类似地，如果 PCR.WAREIEN = 1，会产生一个针对上升沿的协议中断。
- WA 结束事件：
  WA 产生块还指示，在通过写 PCR.WAGEN = 0 禁止了 WA 产生功能之后，何时停止 WA 的产生。如果 PCR.ENDIEN = 1，可产生一个协议中断。
- DX2T 事件：
  触发信号 DX2T 的激活由 PSR.DX2TEV = 1 指示，如果 PCR.DX2TIEN = 1，可产生一个协议中断。如果像在 SSC 模式那样使用延迟补偿，则使用该事件来评估补偿值而不使用 WA 上升 / 下降沿信号 (详细信息请参见对应的 SSC 小节)。

15.6.4　从模式下操作 IIS

为了操作从模式下的 IIS，必须考虑以下问题：

- 选择 IIS 模式：
  建议在 CCR.MODE = 0000_B 期间配置那些不能在运行时改变的所有 IIS 参数。应编程使位域 SCTR.TRM = 11_B。输入级的配置必须在 CCR.MODE = 0000_B 时完成，以避免意外的输入信号边沿，然后可以通过设置 CCR.MODE = 0011_B 来使能 IIS 模式。
- 数据传送的引脚连接：
  通过设置 DX0CR.INSW = 1，来建立输入级 DX0 与所选接收数据输入引脚 (DIN0) 的连接。为发送器配置一个发送数据输出引脚 (DOUT0)。
  数据移位单元允许基于同一 WA 信号进行全双工数据传送，DX0 级提供的值被视为数据位 (能独立于发送器来禁止接收功能)。如果只是为了接收 IIS 数据，则发送端配置不是必须的 (不为 DOUT0 信号分配引脚)。
  为了避免在输出端出现意外的尖峰脉冲，必须在 IIS 模式被使能后再执行使能备用输出端口功能的步骤。
- 移位时钟的引脚连接：
  通过设置 DX1CR.INSW = 1 和反向极性 (DX1CR.DPOL = 1) 来建立输入级 DX1 与所选移位时钟输入引脚 (SCLKIN) 的连接。
- WA 输入的引脚连接：
  通过设置 DX2CR.INSW = 1 来建立输入级 DX2 与 WA 输入引脚 (SELIN) 的连接。
- 波特率产生：
  从模式不需要波特率发生器，可通过分数分频器将其关闭。
- WA 产生：
  从模式不需要 WA 产生功能，可将其关闭 (PCR.WAGEN = 0)。

15.6.4.1 协议事件和中断

在 IIS 从模式可产生下述协议相关的事件，这些事件可导致产生一个协议中断。

请注意，寄存器 PSR 中的位并不都是由硬件自动清零的，为了监视新到来的事件，必须采用软件清零。

- WA 上升沿 / 下降沿 / 结束事件：
  因为 WA 发生器被关闭，所以这些事件不可用。
- DX2T 事件 :
  触发信号 DX2T 的激活由 PSR.DX2TEV = 1 指示，如果 PCR.DX2TIEN = 1，可产生一个协议中断。

15.6.5 IIS 协议寄存器

在 IIS 模式，寄存器 PCR 和 PSR 处理与 IIS 相关的信息。

15.6.5.1 IIS 协议控制寄存器

在 IIS 模式， PCR 寄存器中的位或位域依本节所述定义。

PCR
协议控制寄存器 [IIS 模式] **($3C_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MCLK					0					TDEL					
rw					rw					rw					

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DX2TIEN					0				ENDIEN	WAREIEN	WAFEIEN	0	SELINV	DTEN	WAGEN
rw					rw				rw	rw	rw	rw	rw	rw	rw

域	位	类型	描述
WAGEN	0	rw	**WA 产生使能** 该位使能 / 禁止字地址控制输出信号 WA 的产生。 0_B IIS 可作为从器件使用。字地址信号的产生被禁止。输出信号 WA 为 0。 MCLKO 信号的产生取决于 PCR.MCLK。 1_B IIS 可作为主器件使用。字地址信号的产生被使能。该信号在被使能后以 0 开始。 MCLK 的产生被使能，与 PCR.MCLK 无关。 在清除 WAGEN 后， USIC 模块在接下来的 4 个 WA 周期内停止产生 WA 信号。
DTEN	1	rw	**数据传送使能** 该位使能 / 禁止 IIS 帧的传送，即控制是否响应输入字地址控制线信号 WA 变化的变化。 0_B WA 输入信号的变化被忽略，不发生数据传送。 1_B 数据传送被使能。
SELINV	2	rw	**选择反向** 该位定义 SELOx 输出的极性与内部产生的字地址信号 WA 的关系。 0_B SELOx 输出与 WA 信号具有相同的极性。 1_B SELOx 输出与 WA 信号具有相反的极性。
WAFEIEN	4	rw	**WA 下降沿中断使能** 该位使能 / 禁止在产生 WA 下降沿产生时激活一个协议中断。 0_B 在产生 WA 下降沿时不激活协议中断。 1_B 在产生 WA 下降沿时激活一个协议中断。

域	位	类型	描述
WAREIEN	5	rw	**WA 上升沿中断使能** 该位使能 / 禁止在产生 WA 上升沿时激活一个协议中断。 0_B 在产生 WA 上升沿时不激活协议中断。 1_B 在产生 WA 上升沿时激活一个协议中断。
ENDIEN	6	rw	**结束 (END) 中断使能** 该位使能 / 禁止在清除 PCR.WAGEN 后 WA 信号产生停止时激活一个协议中断 (完整的系统字长在 WA 信号停止之前已处理完) 0_B 不激活协议中断 1_B 激活协议中断
DX2TIEN	15	rw	**DX2T 中断使能** 该位使能 / 禁止在 DX2T 信号被激活时产生一个协议中断 (由 PSR.DX2TEV = 1 指示)。 0_B 在 DX2T 被激活时不产生协议中断。 1_B 在 DX2T 被激活时产生协议中断。
TDEL	[21:16]	rw	**传送延迟** 该位域定义在检测到一个事件时要使用的传送延迟。如果位域 TDEL = 0，额外延迟功能被关闭，引入一个移位时钟周期的延迟。
MCLK	31	rw	**主时钟使能** 该位使能主时钟 MCLK 的产生 (不直接用于 IIS 协议，可作为通用频率输出使用)。 0_B 禁止 MCLK 产生， MCLK 为 0。 1_B 使能 MCLK 产生。
0	3, [14:7], [30:22]	rw	**保留位** 读访问返回 0，应写入 0。

15.6.5.2 IIS 协议状态寄存器

下述 PSR 状态位或位域在 IIS 模式可用。请注意，寄存器 PSR 中的位不能由硬件清零。向 PSCR 寄存器中的一个位写 1 可清除 PSR 寄存器中相应的标志。向 PSR 寄存器中的一个位写 1 可置位相应的标志，但不会导致进一步的动作 (不会产生中断)。写 0 无效。应在使能一个新协议之前将这些标志清零。

PSR
协议状态寄存器 [IIS 模式] **(48_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															BRGIF
r															rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF	0			END	WARE	WAFE	DX2TEV	0	DX2S	WA
rwh	rwh	rwh	rwh	rwh	rwh	r			rwh	rwh	rwh	rwh	r	rwh	rwh

域	位	类型	描述
WA	0	rwh	**字地址** 该位指示在检测到 WA 变化之后所采样的 WA 输入信号的状态。该信息被转发到相应的位 RBUFSR[9]，以区分接收到的是右声道还是左声道数据。 0_B WA 的采样值为 0。 1_B WA 的采样值为 1。
DX2S	1	rwh	**DX2S 状态** 该位指示 DX2S 信号的当前状态，该信号被用作字地址信号 WA。 0_B DX2S 为 0。 1_B DX2S 为 1。
DX2TEV	3	rwh	**检测到 DX2T 事件**[1)] 该位指示 DX2T 信号是否被激活。在 IIS 从模式，如果 PCR.DX2TIEN = 1， DX2T 的激活会产生一个协议中断。 0_B DX2T 信号未激活。 1_B DX2T 信号已激活。
WAFE	4	rwh	**WA 下降沿事件**[1)] 该位指示已产生一个 WA 输出信号下降沿，如果 PCR.WAFEIEN = 1，该事件可产生一个协议中断。 0_B 未产生 WA 下降沿。 1_B 已产生 WA 下降沿。

域	位	类型	描述
WARE	5	rwh	**WA 上升沿事件** [1)] 该位指示已产生一个 WA 输出信号上升沿，如果 PCR.WAREIEN = 1，该事件会产生一个协议中断。 0_B 未产生 WA 上升沿。 1_B 已产生 WA 上升沿。
END	6	rwh	**WA 产生结束** [1)] 该位指示在清零 PCR.WAGEN 后 WA 产生已结束。应在软件清零 WAGEN 之前将该位清零。 0_B WA 产生未结束。 (如果它正在运行且 WAGEN 已被清零)。 1_B WA 产生已结束 (如果它一直都在运行)。
RSIF	10	rwh	**接收开始指示标志** 0_B 未发生接收开始事件。 1_B 发生了接收开始事件
DLIF	11	rwh	**数据丢失指示标志** 0_B 未发生数据丢失事件。 1_B 发生了数据丢失事件。
TSIF	12	rwh	**发送移位指示标志** 0_B 未发生发送移位事件。 1_B 发生了发送移位事件。
TBIF	13	rwh	**发送缓冲区指示标志** 0_B 未发生发送缓冲区事件。 1_B 发生了发送缓冲区事件。
RIF	14	rwh	**接收指示标志** 0_B 未发生接收事件。 1_B 发生了接收事件。
AIF	15	rwh	**备用接收指示标志** 0_B 未发生备用接收事件。 1_B 发生了备用接收事件。
BRGIF	16	rwh	**波特率发生器指示标志** 0_B 未发生波特率发生器事件。 1_B 发生了波特率发生器事件。
0	2, [9:7], [31:17]	r	**保留** 读访问返回 0，在 IIS 模式不被修改。

1) 该状态位可产生一个协议中断 (见页 15-17)。通用中断状态标志在通用中断一章中描述。

15.7 服务请求产生

USIC 模块提供六个服务请求输出 SR[5:0]，这些服务请求为两个通道共享。服务请求输出 SR[5:0] 在嵌套向量中断控制器（NVIC）中被连接到中断节点。

每个 USIC 通信通道都可被连接到最多 6 个服务请求处理机 (连接到 USICx.SR[5:0]，尽管一般只使用 3 或 4 个。例如：一个用于发送，一个用于接收，一个或两个用于协议或错误处理，或者用于备用接收事件)。

15.8 调试行为

每个 USIC 通信通道都能被预先配置为在 CPU 的程序执行被调试器停止时进入四个内核模式之一。

详见 **15.2.2.2 节** 。

15.9 电源、复位和时钟

USIC 模块位于内核电源域。可通过一次系统复位将模块复位到其默认状态。

USIC 模块的时钟来自 SCU 的主时钟 MCLK。在默认状态，MCLK 被禁止，可通过 SCU_CGATCLR0 寄存器将其使能。使能和禁止模块时钟可能会引起负载变化并发生时钟消隐，详见 SCU 一章中 CCU(时钟门控) 一节的描述。强烈建议在用户初始代码中设置模块时钟，以避免在运行期间发生时钟消隐。

注：　为了区分 USIC 波特率发生器输出和主时钟 (MCLK)，在本章 (USIC) 中通篇使用 f_{PB} 替代 SCU MCLK。

15.10 初始化和系统相关性

在操作 USIC 模块之前，应用程序必须使用以下初始化序列：

- 通过向 KSCFG 寄存器中的 MODEN 和 BPMODEN 位写 1 来使能 USIC 模块。

15.11 寄存器

表 15-19 列出了对一个 USIC 通道及 FIFO 缓冲区编程所需要的全部寄存器。表中概述了 USIC 通信通道寄存器并定义了相对地址和复位值。

请注意，可用任何访问宽度 (8 位、16 位、32 位) 访问所有寄存器，与所描述宽度无关。

所有的 USIC 寄存器 (位域 KSCFG.SUMCFG 除外) 总是由系统复位来复位。位域 KSCFG.SUMCFG 的复位由调试复位执行。

注：　读那些被标记为 "w" 的寄存器位总是返回 0. 这些位用于修改其他寄存器中的触发器，或用于触发内部操作。

图 15-65 示出了 USIC 模块寄存器和通道寄存器的寄存器类型。在一个具体的微控制器中，USIC 模块 "x" 的模块寄存器用模块前缀 "USICx_" 来标记。USIC 模块 "x" 的通道寄存器用通道前缀 "USICx_CH0_" 和 "USICx_CH1_" 来标记。

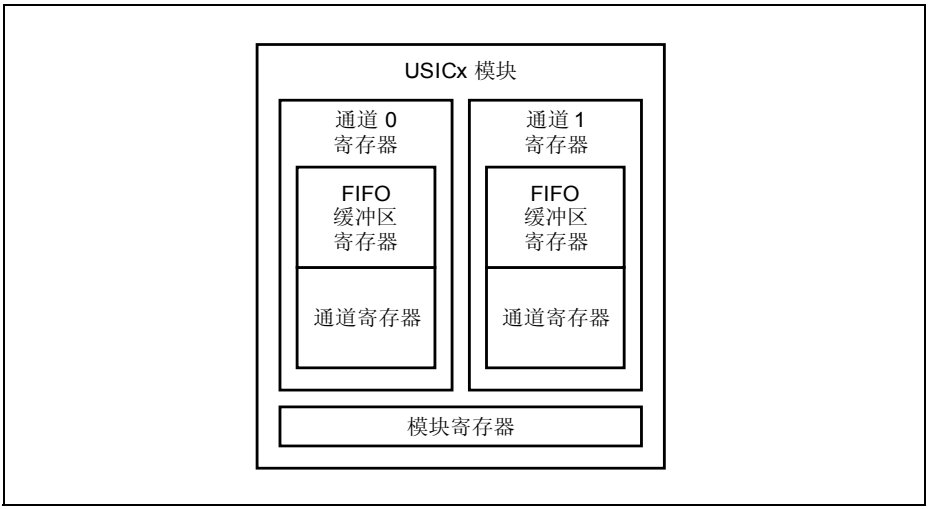


图 15-65　USIC 模块和通道寄存器

表 15-19　USIC 内核相关的寄存器和内核寄存器

寄存器简称	寄存器全称	偏移地址	访问方式		描述见
			读	写	
模块寄存器[1)]					
ID	模块标识寄存器	008_H	U, PV	U, PV	页 15-137
通道寄存器					
–	保留	000_H	BE	BE	–
CCFG	通道配置寄存器	004_H	U, PV	U, PV	页 15-141
KSCFG	内核状态配置寄存器	$00C_H$	U, PV	U, PV	页 15-142
FDR	分数分频器寄存器	010_H	U, PV	PV	页 15-152
BRG	波特率发生器寄存器	014_H	U, PV	PV	页 15-154
INPR	中断节点指针寄存器	018_H	U, PV	U, PV	页 15-143
DX0CR	输入控制寄存器 0	$01C_H$	U, PV	U, PV	页 15-148
DX1CR	输入控制寄存器 1	020_H	U, PV	U, PV	页 15-150
DX2CR	输入控制寄存器 2	024_H	U, PV	U, PV	页 15-148
DX3CR	输入控制寄存器 3	028_H	U, PV	U, PV	
DX4CR	输入控制寄存器 4	$02C_H$	U, PV	U, PV	
DX5CR	输入控制寄存器 5	030_H	U, PV	U, PV	

表 15-19 USIC 内核相关的寄存器和内核寄存器 (续表)

寄存器简称	寄存器全称	偏移地址	访问方式		描述见
			读	写	
SCTR	移位控制寄存器	034_H	U, PV	U, PV	页 15-157
TCSR	发送控制 / 状态寄存器	038_H	U, PV	U, PV	页 15-159
PCR	协议控制寄存器	$03C_H$	U, PV	U, PV	页 15-145[2]
			U, PV	U, PV	页 15-55[3]
			U, PV	U, PV	页 15-83[4]
			U, PV	U, PV	页 15-112[5]
			U, PV	U, PV	页 15-129[6]
CCR	通道控制寄存器	040_H	U, PV	PV	页 15-138
CMTR	捕获模式定时器寄存器	044_H	U, PV	U, PV	页 15-156
PSR	协议状态寄存器	048_H	U, PV	U, PV	页 15-145[2]
			U, PV	U, PV	页 15-58[3]
			U, PV	U, PV	页 15-86[4]
			U, PV	U, PV	页 15-114[5]
			U, PV	U, PV	页 15-131[6]
PSCR	协议状态清除寄存器	$04C_H$	U, PV	U, PV	页 15-147
RBUFSR	接收缓冲区状态寄存器	050_H	U, PV	U, PV	页 15-174
RBUF	接收缓冲区寄存器	054_H	U, PV	U, PV	页 15-172
RBUFD	调试模式接收缓冲区寄存器	058_H	U, PV	U, PV	页 15-173
RBUF0	接收缓冲区寄存器 0	$05C_H$	U, PV	U, PV	页 15-167
RBUF1	接收缓冲区寄存器 1	060_H	U, PV	U, PV	页 15-168
RBUF01SR	接收缓冲区 01 状态寄存器	064_H	U, PV	U, PV	页 15-168
FMR	标志修改寄存器	068_H	U, PV	U, PV	页 15-165
–	保留；不能访问该单元	$06C_H$	U, PV	BE	–
–	保留	070_H - $07C_H$	BE	BE	–
TBUFx	发送缓冲区输入单元 x (x = 00-31)	080_H + x*4	U, PV	U, PV	页 15-166
FIFO 缓冲区寄存器					
BYP	旁路数据寄存器	100_H	U, PV	U, PV	页 15-175
BYPCR	旁路控制寄存器	104_H	U, PV	U, PV	页 15-175

表 15-19 USIC 内核相关的寄存器和内核寄存器 (续表)

寄存器简称	寄存器全称	偏移地址	访问方式		描述见
			读	写	
TBCTR	发送缓冲区控制寄存器	108_H	U, PV	U, PV	页 15-182
RBCTR	接收缓冲区控制寄存器	$10C_H$	U, PV	U, PV	页 15-185
TRBPTR	发送 / 接收缓冲区指针寄存器	110_H	U, PV	U, PV	页 15-191
TRBSR	发送 / 接收缓冲区状态寄存器	114_H	U, PV	U, PV	页 15-178
TRBSCR	发送 / 接收缓冲区状态清除寄存器	118_H	U, PV	U, PV	页 15-181
OUTR	接收缓冲区输出寄存器	$11C_H$	U, PV	U, PV	页 15-190
OUTDR	调试模式接收缓冲区输出寄存器	120_H	U, PV	U, PV	页 15-191
–	保留	124_H - $17C_H$	BE	BE	–
INx	发送 FIFO 缓冲区 输入单元 x (x = 00-31)	180_H + x*4	U, PV	U, PV	页 15-189

1) 模块标识寄存器的详细信息在实现一节描述 (见页 15-137)。
2) 该页说明通用寄存器的情况
3) 该页说明 ASC 模式下通用寄存器的情况。
4) 该页说明 SSC 模式下通用寄存器的情况。
5) 该页说明 IIC 模式下通用寄存器的情况。
6) 该页说明 IIS 模式下通用寄存器的情况。

15.11.1 地址映射

USIC 通信通道的寄存器位于下述基地址。确切的寄存器地址由寄存器的相对地址 (在表 15-19 中给出) 加上通道基地址 (在表 15-20 中给出) 给出。

表 15-20 寄存器地址空间

模块	基地址	结束地址	备注
USIC0_CH0	48000000_H	$480001FF_H$	–
USIC0_CH1	48000200_H	$480003FF_H$	–

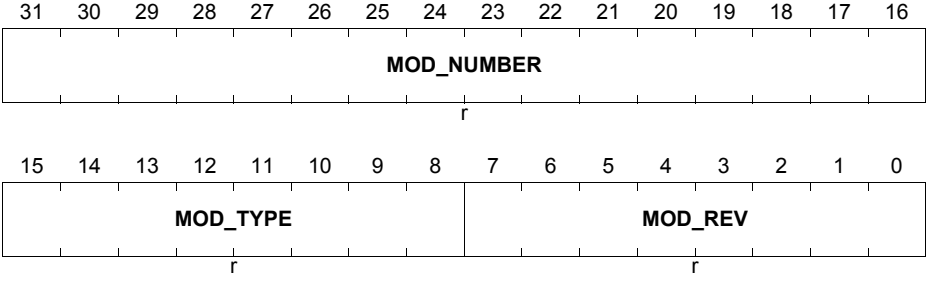
表 15-21 FIFO 和保留地址空间

模块	基地址	结束地址	访问方式		备注
			读	写	
USIC0	48000400_H	$480007FF_H$	nBE	在直接 RAM 测试模式为 nBE；否则为 BE	USIC0 的 RAM 区由 USIC0_CH0 和 USIC0_CH1 共享
保留	48000800_H	$4800FFFF_H$	BE	BE	–

15.11.2 模块标识寄存器

模块标识寄存器指示 USIC 模块的功能和设计步骤。

USIC0_ID
模块标识寄存器 **(4800 0008$_H$)** **复位值 : 00AA C0XX$_H$**



域	位	类型	描述
MOD_REV	[7:0]	r	**模块版本号** MOD_REV 定义版本号。模块的版本号从数值 01_H 开始（第一版）。
MOD_TYPE	[15:8]	r	**模块类型** 该位域为 $C0_H$。它定义该模块为 32 位模块。
MOD_NUMBER	[31:16]	r	**模块号值** 该位域定义了 USIC 模块标识号 ($00AA_H$ = USIC)。

15.11.3 通道控制和配置寄存器

15.11.3.1 通道控制寄存器

通道控制寄存器包含用于硬件端口控制、基于通道事件的中断产生控制、奇偶校验位产生控制和 USIC 通道协议选择控制的使能 / 禁止位。

只有特权模式访问可对 FDR 写入。

CCR
通道控制寄存器 **(40_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															BRGIEN
r															rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIEN	RIEN	TBIEN	TSIEN	DLIEN	RSIEN	PM		HPCEN		0		MODE			
rw	rw	rw	rw	rw	rw	rw		rw		r		rw			

域	位	类型	描述
MODE	[3:0]	rw	**工作模式** 该位域为 USIC 通道选择协议。选择一个不可用的协议 (见寄存器 CCFG) 或保留的组合会禁止 USIC 通道。当在两个协议之间切换时，必须在选择一个新协议之前禁止 USIC 通道。在这种情况下，必须用软件清除或更新寄存器 PCR 和 PSR。 0_H 禁止 USIC 通道。所有协议相关的状态机都被设置为空闲状态。 1_H 选择 SSC(SPI) 协议 2_H 选择 ASC (SCI, UART) 协议。 3_H 选择 IIS 协议。 4_H 选择 IIC 协议。 其他位组合保留。

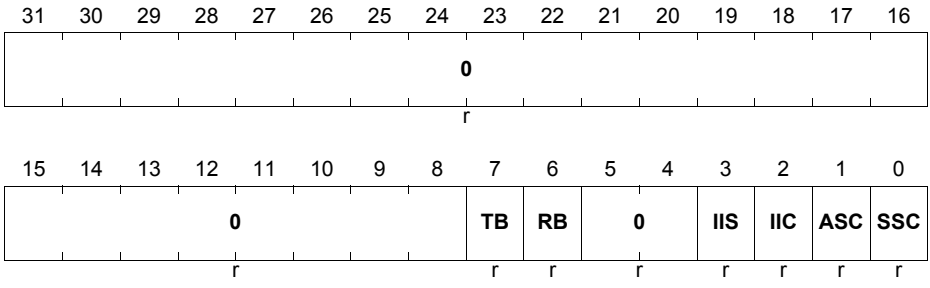
域	位	类型	描述
HPCEN	[7:6]	rw	**硬件端口控制使能** 该位域使能对 DX[3:0] 和 DOUT[3:0] 引脚的硬件端口控制。 00_B 禁止硬件端口控制。 01_B 使能对 DX0 和 DOUT0 的硬件端口控制。 10_B 使能对 DX0 和 DOUT[1:0] 的硬件端口控制。 11_B 使能对 DX0、DX[5:3] 和 DOUT[3:0] 的硬件端口控制。 *注：硬件端口控制功能仅用于如双位和四位 SSC 这种半双工配置的 SSC 协议位 SSC。对于其他所有协议，HPCEN 必须总是被写为 00_B。*
PM	[9:8]	rw	**奇偶校验模式** 该位域定义被采样输入值的奇偶校验位的产生。 00_B 禁止奇偶校验位产生 01_B 保留 10_B 选择偶校验 (数据中 1 的个数为奇数时，奇偶位 = 1 ；数据中 1 的个数为偶数时，奇偶位 = 0)。 11_B 选择奇校验 (数据中 1 的个数为奇数时，奇偶位 = 0 ；数据中 1 的个数为偶数时，奇偶位 = 1)
RSIEN	10	rw	**接收器开始中断使能** 该位使能接收开始事件中断产生。 0_B 禁止接收开始中断。 1_B 使能接收开始中断。 在发生接收开始事件的情况下，由 INPR.TBINP 指示的服务请求输出 SRx 被激活。
DLIEN	11	rw	**数据丢失中断使能** 该位使能数据丢失事件中断产生 (当 RDVx = 1 时，数据被接收到 RBUFx 中)。 0_B 禁止数据丢失中断。 1_B 使能数据丢失中断，在发生数据丢失事件的情况下，由 INPR.PINP 指示的服务请求输出 SRx 被激活。

域	位	类型	描述
TSIEN	12	rw	**发送移位中断使能** 该位使能发送移位事件中断产生。 0_B 禁止发送移位中断。 1_B 使能发送移位中断。在发生发送移位事件的情况下，由 INPR.TSINP 指示的服务请求输出 SRx 被激活。
TBIEN	13	rw	**发送缓冲区中断使能** 该位使能发送缓冲区事件中断产生。 0_B 禁止发送缓冲区中断。 1_B 使能发送缓冲区中断。在发生发送缓冲区事件的情况下，由 INPR.TBINP 指示的服务请求输出 SRx 被激活。
RIEN	14	rw	**接收中断使能** 该位使能接收事件中断产生。 0_B 禁止中断。 1_B 使能接收中断。在发生接收事件的情况下，由 INPR.RINP 指示的服务请求输出 SRx 被激活。
AIEN	15	rw	**备用接收中断使能** 该位使能备用接收事件中断产生。 0_B 禁止备用接收中断。 1_B 使能备用接收中断。在发生备用接收事件的情况下，由 INPR.AINP 指示的服务请求输出 SRx 被激活。
BRGIEN	16	rw	**波特率发生器中断使能** 该位使能波特率发生器事件中断产生。 0_B 禁止波特率发生器中断。 1_B 使能波特率发生器中断。在发生波特率发生器事件的情况下，由 INPR.PINP 指示的服务请求输出 SRx 会被激活。
0	[5:4], [31:17]	r	**保留** 读访问返回 0，应写入 0。

15.11.3.2 通道配置寄存器

通道配置寄存器指示在 USIC 通道中可用的功能。

CCFG
通道配置寄存器 **(04_H)** **复位值 : 0000 00CF$_H$**



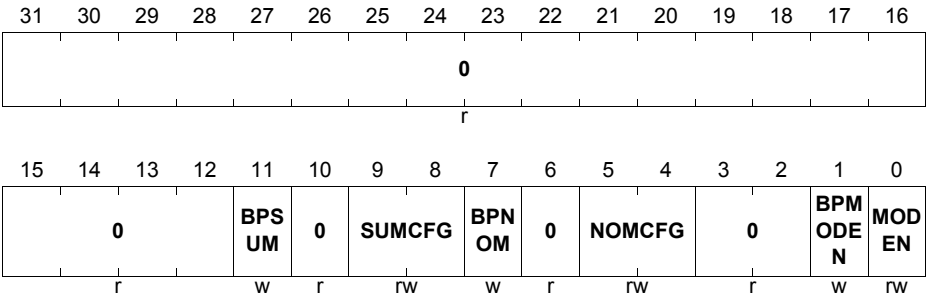
域	位	类型	描述
SSC	0	r	**SSC 协议可用** 该位指示 SSC 协议是否可用。 0_B SSC 协议不可用。 1_B SSC 协议可用。
ASC	1	r	**ASC 协议可用** 该位指示 ASC 协议是否可用。 0_B ASC 协议不可用 1_B ASC 协议可用
IIC	2	r	**IIC 协议可用** 该位指示 IIC 协议是否可用。 0_B IIC 协议不可用。 1_B IIC 协议可用。
IIS	3	r	**IIS 协议可用** 该位指示 IIS 协议是否可用。 0_B IIS 协议不可用。 1_B IIS 协议可用。
RB	6	r	**接收 FIFO 缓冲区可用** 该位指示额外的接收 FIFO 缓冲区是否可用。 0_B 接收 FIFO 缓冲区不可用。 1_B 接收 FIFO 缓冲区可用。
TB	7	r	**发送 FIFO 缓冲区可用** 该位指示额外的发送 FIFO 缓冲区是否可用。 0_B 发送 FIFO 缓冲区不可用。 1_B 发送 FIFO 缓冲区可用。

域	位	类型	描述
0	[5:4], [15:8], [31:16]	r	**保留** 读访问返回 0；应写入 0。

15.11.3.3 内核状态配置寄存器

内核状态配置寄存器 KSCFG 允许为器件的不同操作模式选择所期望的内核模式。

KSCFG
内核状态配置寄存器 **(0C$_H$)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
MODEN	0	rw	**模块使能** 该位使能模块内核时钟和模块功能。 0_B 模块被立即关闭 (不必遵守停止条件)。模块不响应模式控制操作，并且模块时钟被关闭。模块不响应读访问，并且忽略写访问 (KSCFG 除外)。 1_B 模块被打开并且可以运行。在向 MODEN 写 1 后，建议读寄存器 KSCFG，以避免在访问其他服务请求处理 寄存器之前控制块中流水线的影响。
BPMODEN	1	w	**MODEN 位保护** 该位使能对位 MODEN 的写访问。对该位的读访问总是返回 0。 0_B MODEN 保持不变。 1_B MODEN 由写入值更新。

域	位	类型	描述
NOMCFG	[5:4]	rw	**正常工作模式配置** 该位域定义在正常工作模式下使用的内核模式。 00_B 选择运行模式 0 01_B 选择运行模式 1 10_B 选择停止模式 0 11_B 选择停止模式 1
BPNOM	7	w	**NOMCFG 位保护** 该位使能对位域 NOMCFG 的写访问。对该位的读访问总是返回 0。 0_B NOMCFG 保持不变。 1_B NOMCFG 由写入值更新。
SUMCFG	[9:8]	rw	**挂起模式配置** 该位域定义在挂起模式下使用的内核模式。编码同 NOMCFG。
BPSUM	11	w	**SUMCFG 位保护** 该位使能对位域 SUMCFG 的写访问。对该位的读访问总是返回 0。 0_B SUMCFG 保持不变。 1_B SUMCFG 由写入值更新。
0	[3:2], 6, 10, [31:12]	r	**保留** 读访问返回 0；应写入 0。在从 BootROM 退出后，位 2 的读出值可为 1(但可被忽略)。

15.11.3.4 中断节点指针寄存器

中断节点指针寄存器定义服务请求输出SRx，如果相应的事件发生并且中断产生被使能，SRx 会被激活。

INPR
中断节点指针寄存器 **(18_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													PINP		
r													rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	AINP			0	RINP			0	TBINP			0	TSINP		
r	rw			r	rw			r	rw			r	rw		

位域	位	类型	描述
TSINP	[2:0]	rw	**发送移位中断节点指针** 该位域定义在发生一个发送移位中断时哪一个服务请求输出 SRx 被激活。 000_B 输出 SR0 被激活。 001_B 输出 SR1 被激活。 010_B 输出 SR2 被激活。 011_B 输出 SR3 被激活。 100_B 输出 SR4 被激活。 101_B 输出 SR5 被激活。 *注： 该位域的所有其他设置被保留。*
TBINP	[6:4]	rw	**发送缓冲区中断节点指针** 该位域定义在发生一个发送缓冲区中断或接收开始中断时哪一个服务请求输出 SRx 被激活。 编码同 TSINP。
RINP	[10:8]	rw	**接收中断节点指针** 该位域定义在发生一个接收中断时哪一个服务请求输出 SRx 被激活。 编码同 TSINP。
AINP	[14:12]	rw	**可选接收中断节点指针** 该位域定义在发生一个备用接收中断时哪一个服务请求输出 SRx 被激活。 编码同 TSINP。
PINP	[18:16]	rw	**协议中断节点指针** 该位域定义在发生一个协议中断时哪一个服务请求输出 SRx 被激活。 编码同 TSINP。
0	3, 7, 11, 15, [31:19]	r	**保留** 读访问返回 0；应写入 0。

15.11.4 协议相关寄存器

15.11.4.1 协议控制寄存器

协议控制寄存器中的位定义协议特有的功能。必须在使能一个新协议之前用软件配置这些位。只考虑所选协议使用的那些的，其他位的读出值总是为 0。协议特有的含义在相关协议的小节中描述。

PCR
协议控制寄存器 **(3C$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CTR 31	**CTR 30**	**CTR 29**	**CTR 28**	**CTR 27**	**CTR 26**	**CTR 25**	**CTR 24**	**CTR 23**	**CTR 22**	**CTR 21**	**CTR 20**	**CTR 19**	**CTR 18**	**CTR 17**	**CTR 16**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CTR 15	**CTR 14**	**CTR 13**	**CTR 12**	**CTR 11**	**CTR 10**	**CTR 9**	**CTR 8**	**CTR 7**	**CTR 6**	**CTR 5**	**CTR 4**	**CTR 3**	**CTR 2**	**CTR 1**	**CTR 0**
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

域	位	类型	描述
CTRx (x = 0-31)	x	rw	**协议控制位 x** 该位是协议控制位。

15.11.4.2 协议状态寄存器

通过向寄存器 PSCR 中的一个位写 1 可清除协议状态寄存器中对应的标志。向 PSR 中的一个位写 1 可置位相应的标志，但不能导致更进一步的动作 (不会产生中断)。写 0 无效。应在使能一个新协议之前用软件将这些标志清零。协议特有的含义在相关协议的小节中描述。

PSR
协议状态寄存器 **(48$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
							0								**BRG IF**
							r								rwh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	**RIF**	**TBIF**	**TSIF**	**DLIF**	**RSIF**	**ST9**	**ST8**	**ST7**	**ST6**	**ST5**	**ST4**	**ST3**	**ST2**	**ST1**	**ST0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

位域	位	类型	描述
STx (x = 0-9)	x	rwh	**协议状态标志 x** 见协议具体描述。
RSIF	10	rwh	**接收器开始指示标志** 0_B 未发生接收开始事件。 1_B 发生了接收开始事件。
DLIF	11	rwh	**数据丢失指示标志** 0_B 未发生数据丢失事件。 1_B 发生了数据丢失事件。
TSIF	12	rwh	**发送移位指示标志** 0_B 未发生发送移位事件。 1_B 发生了发送移位事件。
TBIF	13	rwh	**发送缓冲区指示标志** 0_B 未发生发送缓冲区事件。 1_B 发生了发送缓冲区事件。
RIF	14	rwh	**接收指示标志** 0_B 未发生接收标志。 1_B 发生了接收事件。
AIF	15	rwh	**备用接收指示标志** 0_B 未发生备用接收事件。 1_B 发生了备用接收事件。
BRGIF	16	rwh	**波特率发生器指示标志** 0_B 未发生波特率发生器事件。 1_B 发生了波特率发生器事件。
0	[31:17]	r	**保留；** 读访问返回 0；应写入 0。

15.11.4.3 协议状态清除寄存器

对该寄存器所有位的读访问总是返回 0。

PSCR
协议状态清除寄存器 **($4C_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															CBRGIF
r															w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAIF	CRIF	CTBIF	CTSIF	CDLIF	CRSIF	CST9	CST8	CST7	CST6	CST5	CST4	CST3	CST2	CST1	CST0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

域	位	类型	描述
CSTx (x = 0-9)	x	w	**清除 PSR 中的状态标志 x** 0_B 无操作 1_B 标志 PSR.STx 被清除。
CRSIF	10	w	**清除接收器开始指示标志** 0_B 无操作 1_B 标志 PSR.RSIF 被清除。
CDLIF	11	w	**清除数据丢失指示标志** 0_B 无操作 1_B 标志 PSR.DLIF 被清除
CTSIF	12	w	**清除发送改变指示标志** 0_B 无操作 1_B 标志 PSR.TSIF 被清除
CTBIF	13	w	**清除发送缓冲指示标志** 0_B 无操作 1_B 标志 PSR.TBIF 被清除
CRIF	14	w	**清除接收指示标志** 0_B 无操作 1_B 标志 PSR.RIF 被清除
CAIF	15	w	**清除备用接收指示标志** 0_B 无操作 1_B 标志 PSR.AIF 被清除
CBRGIF	16	w	**清除波特率发生器指示标志** 0_B 无操作 1_B 标志 PSR.BRGIF 被清除

域	位	类型	描述
0	[31:17]	r	**保留** 读访问返回 0 ； 应写入 0。

15.11.5 输入级寄存器

15.11.5.1 输入控制寄存器

输入控制寄存器包含定义输入级特性的位 (如输入级 DX0 由寄存器 DX0CR 控制等)。

DX0CR
输入控制寄存器 0 **($1C_H$)** **复位值 : 0000 0000$_H$**
DX2CR
输入控制寄存器 2 **(24_H)** **复位值 : 0000 0000$_H$**
DX3CR
输入控制寄存器 3 **(28_H)** **复位值 : 0000 0000$_H$**
DX4CR
输入控制寄存器 4 **($2C_H$)** **复位值 : 0000 0000$_H$**
DX5CR
输入控制寄存器 5 **(30_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXS	0			CM		SFSEL	DPOL	0	DSEN	DFEN	INSW	0	DSEL		
rh	r			rw		rw	rw	r	rw	rw	rw	r	rw		

域	位	类型	描述
DSEL	[2:0]	rw	**输入信号的数据选择** 该位域为协议预处理器定义输入线所对应的输入数据信号。可从输入向量 DXn[G:A] 中选择。 000_B 选择数据输入 DXnA 001_B 选择数据输入 DXnB 010_B 选择数据输入 DXnC 011_B 选择数据输入 DXnD 100_B 选择数据输入 DXnE 101_B 选择数据输入 DXnF 110_B 选择数据输入 DXnG 111_B 数据输入总是为 1
INSW	4	rw	**输入切换** 该位定义数据移位单元输入是来自输入数据路径 DXn 还是来自所选择的协议预处理器。 0_B 数据移位单元的输入由协议预处理器控制。 1_B 数据移位单元的输入被连接到所选数据输入线。如果信号直接来源于一个输入引脚而不经过协议预处理器的处理，则使用该设置。
DFEN	5	rw	**数字滤波器使能** 该位使能 / 禁止信号 DXnS 的数字滤波器。 0_B 不对输入信号执行数字滤波。 1_B 对输入信号执行数字滤波。
DSEN	6	rw	**数据同步使能** 该位选择是使用异步输入信号还是经过同步的 (还可选择滤波) 信号 DXnS 作为数据移位单元的输入。 0_B 可使用未经同步的信号作为数据移位单元的输入。 1_B 可使用经过同步的信号作为数据移位单元的输入。
DPOL	8	rw	**DXn 的数据极性** 该位定义输入信号的信号极性。 0_B 输入信号未反相。 1_B 输入信号被反相。
SFSEL	9	rw	**采样频率选择** 该位为同步后的信号 DXnS 定义数字滤波器的采样频率。 0_B 采样频率为 f_{PB} 1_B 采样频率为 f_{FD}

域	位	类型	描述
CM	[11:10]	rw	**组合模式** 该位域选择使用同步后的 (还可选择滤波) 信号 DXnS 的哪个边沿来激活输入级的触发输出 DXnT。 00_B 触发激活被禁止。 01_B 上升沿激活 DXnT。 10_B 下降沿激活 DXnT。 11_B 两个边沿都激活 DXnT。
DXS	15	rh	**同步后的数据值** 该位指示同步后的 (还可选择滤波) 输入信号的值。 0_B DXnS 的当前值为 0。 1_B DXnS 的当前值为 1。
0	3, 7, [14:12], [31:16]	r	**保留** 读访问返回 0；应写入 0。

DX1CR
输入控制寄存器 1 **(20_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DXS	**0**			**CM**		**SFSEL**	**DPOL**	**0**	**DSEN**	**DFEN**	**INSW**	**DCEN**	**DSEL**		
rh	r			rw		rw	rw	r	rw	rw	rw	rw	rw		

域	位	类型	描述
DSEL	[2:0]	rw	**输入信号的数据选择** 该位域为协议预处理器定义输入线所对应的输入数据信号。可从输入向量 DX1[G:A] 中选择。 000_B 选择数据输入 DX1A。 001_B 选择数据输入 DX1B。 010_B 选择数据输入 DX1C。 011_B 选择数据输入 DX1D。 100_B 选择数据输入 DX1E。 101_B 选择数据输入 DX1F。 110_B 选择数据输入 DX1G。 111_B 数据输入总是为 1。
DCEN	3	rw	**延迟补偿使能** 该位选择接收移位时钟是由 INSW 控制还是源自于输入数据路径 DX1。 0_B 接收移位时钟取决于 INSW 的选择。 1_B 接收移位时钟被连接到所选数据输入线。如果在 SSC 和 IIS 协议中需要延迟补偿，则应使用该设置。否则， DCEN 应总是为 0。
INSW	4	rw	**输入切换** 该位定义数据移位单元输入是来自输入数据路径 DX1 还是来自所选择的协议预处理器。 0_B 数据移位单元的输入由协议预处理器控制。 1_B 数据移位单元的输入被连接到所选择的数据输入线。如果信号直接来自一个输入引脚而不经过协议预处理器的处理，则应使用该设置。
DFEN	5	rw	**数字滤波器使能** 该位使能 / 禁止信号 DX1S 的数字滤波器。 0_B 不对输入信号执行数字滤波。 1_B 对输入信号执行数字滤波。
DSEN	6	rw	**数据同步使能** 该位选择是使用异步输入信号还是同步后的 (还可选择滤波) 信号 DX1S 作为数据移位单元的输入。 0_B 使用未经同步的信号作为数据移位单元的输入。 1_B 使用同步后的信号作为数据移位单元的输入。
DPOL	8	rw	**DXn 的数据极性** 该位定义输入信号的信号极性。 0_B 输入信号未反相。 1_B 输入信号被反相。

域	位	类型	描述
SFSEL	9	rw	**采样频率选择** 该位为同步后的信号 DX1S 定义数字滤波器的采样频率。 0_B 采样频率为 f_{PB}。 1_B 采样频率为 f_{FD}。
CM	[11:10]	rw	**组合模式** 该位域选择使用同步后的 (还可选择滤波) 信号 DX1S 的哪一个边沿来激活输入级的触发输出 DX1T。 00_B 触发激活被禁止。 01_B 上升沿激活 DX1T。 10_B 下降沿激活 DX1T。 11_B 两个边沿都激活 DX1T。
DXS	15	rh	**同步后的数据值** 该位指示同步后的 (还可选择滤波) 输入信号的值。 0_B DX1S 的当前值为 0。 1_B DX1S 的当前值为 1。
0	7, [14:12], [31:16]	r	**保留** 读访问返回 0；应写入 0。

15.11.6 波特率发生器寄存器

15.11.6.1 分数分频器寄存器

分数分频器寄存器 FDR 允许产生内部频率 f_{FD}，该频率来源于系统时钟 f_{PB}。
只有特权模式访问可对 FDR 写入。

FDR
分数分频器寄存器 **(10_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		**0**				**RESULT**									
rw		r				rh									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DM		**0**				**STEP**									
rw		r				rw									

域	位	类型	描述
STEP	[9:0]	rw	**STEP 值** 在标准分频器模式， STEP 包含 RESULT 的重载值。在 RESULT 达到 $3FF_H$ 后， STEP 值被重新加载到 RESULT。在分数分频器模式， STEP 定义在每个输入时钟周期要被加到 RESULT 的值。
DM	[15:14]	rw	**分频器模式** 该位域定义分数分频器块的功能。 00_B 分频器被关闭， f_{FD} = 0。 01_B 选择标准分频器模式。 10_B 选择分数分频模式。 11_B 分频器被关闭， f_{FD} = 0
RESULT	[25:16]	rh	**RESULT 值** 在标准分频器模式，该位域在每个 f_{PB} 周期被更新为 RESULT = RESULT + 1 在分数分频器模式，该位域在每个 f_{PB} 周期被更新为 RESULT = RESULT + STEP 如果位域 DM 被写为 01_B 或 10_B， RESULT 被加载的起始值为 $3FF_H$。
0	[31:30]	rw	**保留为将来使用** 必须写 0，以允许正确的分数分频器操作。
0	[13:10], [29:26]	r	**保留** 读访问返回 0 ；应写入 0。

15.11.6.2 波特率发生器寄存器

用于波特率产生和时间测量的协议相关计数器由寄存器 BRG 控制。

只有特权模式访问可对 FDR 写入。

BRG
波特率发生器寄存器 **(14_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SCLKCFG		MCLKCFG	SCLKOSEL	0		PDIV									
rw		rw	rw	r		rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DCTQ					PCTQ		CTQSEL		0	PPPEN	TMEN	0	CLKSEL	
r	rw					rw		rw		r	rw	rw	r	rw	

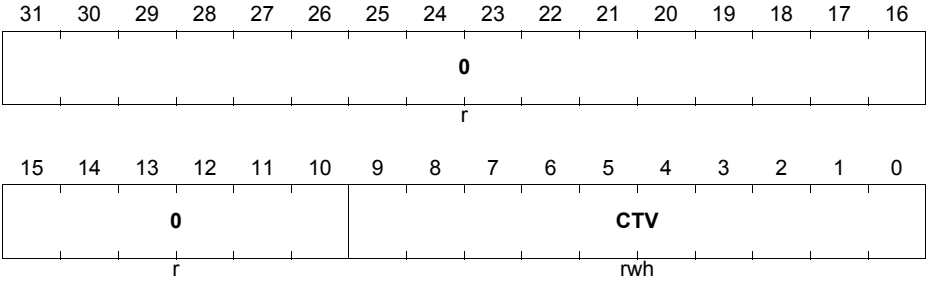
域	位	类型	描述
CLKSEL	[1:0]	rw	**时钟选择** 该位域定义输入频率 f_{PIN} 00_B 选择分数分频器频率 f_{FD}。 01_B 保留，无操作。 10_B 触发信号DX1T定义 f_{PIN}。信号MCLK 在每个f_{PIN} 周期切换。 11_B 信号 MCLK 与 DX1S 一致，f_{PIN} 来源于 DX1S 的上升沿。
TMEN	3	rw	**时间测量使能** 该位使能捕获模式定时器的时间测量功能。 0_B 禁止时间测量：触发信号 DX0T 和 DX1T 被忽略。 1_B 使能时间测量：该 10 位计数器在每个 f_{PPP} 周期进行加 1 计数，在达到其最大值时停止计数。如果触发信号 DX0T 或 DX1T 中有一个被激活，则计数器值被捕获到位域 CTV，计数器被清零并产生一个发送移位事件。
PPPEN	4	rw	**使能 f_{PPP} 的 2:1 分频器** 该位定义输入频率 f_{PPP}。 0_B 禁止 f_{PPP} 的 2:1 分频器 $f_{PPP} = f_{PIN}$ 1_B 使能 f_{PPP} 的 2:1 分频器 $f_{PPP} = f_{MCLK} = f_{PIN} / 2$.

域	位	类型	描述
CTQSEL	[7:6]	rw	**CTQ 输入选择** 该位为协议预处理器定义一个时间量子的长度。 00_B $f_{CTQIN} = f_{PDIV}$ 01_B $f_{CTQIN} = f_{PPP}$ 10_B $f_{CTQIN} = f_{SCLK}$ 11_B $f_{CTQIN} = f_{MCLK}$
PCTQ	[9:8]	rw	**时间量子计数器的预分频器** 该位域为协议预处理器中的时间量子计数器定义一个时间量子 tq 的长度。 $t_Q = (PCTQ + 1) / f_{CTQIN}$
DCTQ	[14:10]	rw	**时间量子计数器的分母** 该位域定义协议预处理器中的时间量子计数器要用到的时间量子 t_q 的数量。
PDIV	[25:16]	rw	**分频器模式：用于产生 f_{PDIV} 的分频因子** 该位域定义输入频率 f_{PPP} 与分频器频率 f_{PDIV} 之间的比率。
SCLKOSEL	28	rw	**移位时钟输出选择** 该位域选择 SCLKOUT 信号的输入源。 0_B 选择来自波特率发生器的 SCLK 作为 SCLKOUT 输入源。 1_B 选择来自 DX1 输入级的发送移位时钟作为 SCLKOUT 输入源。 *注：只能在一个SSC/IIS要求完整的闭环延迟补偿时使用 SCLKOSEL = 1 这一设置。对所有其他情况，应使用默认设置 SCLKOSEL = 0。*
MCLKCFG	29	rw	**主设备时钟配置** 该位域定义 MCLKOUT 信号被动阶段的电平 0_B 被动电平为 0。 1_B 被动电平为 1。
SCLKCFG	[31:30]	rw	**移位时钟输出配置** 该位域定义 MCLKOUT 信号被动阶段的电平，并且使能 / 禁止半个 SCLK 周期的延时。 00_B 被动电平为 0，延时被禁止。 01_B 被动电平为 1，延时被禁止。 10_B 被动电平为 0，延时被使能。 11_B 被动电平为 1，延时被使能。
0	2, 5, 15, [27:26]	r	**保留** 读访问返回 0；应写入 0。

15.11.6.3 捕获模式定时器寄存器

捕获的定时器值由寄存器 CMTR 提供。

CMTR
捕获模式定时器寄存器 **(44_H)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
CTV	[9:0]	rwh	**捕获的定时器值** 如果触发信号 DX0T 或 DX1T 中的一个被相应的输入级激活，则计数器的值被捕获到该位域。
0	[31:10]	r	**保留** 读访问返回 0；应写入 0。

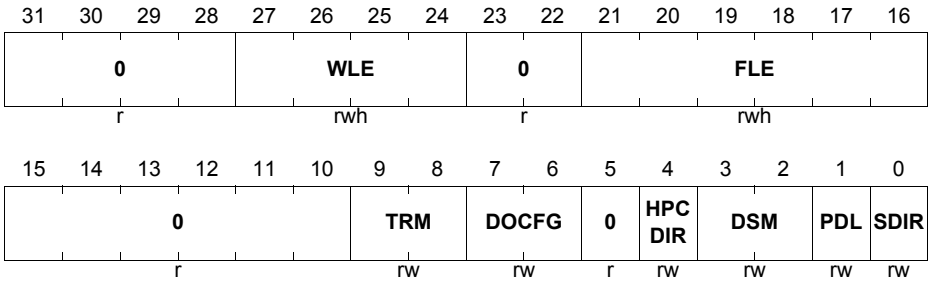
15.11.7 传送控制和状态寄存器

15.11.7.1 移位控制寄存器

数据移位单元由寄存器 SCTR 控制。该寄存器中的数值用于数据发送和接收。

请注意，移位控制设置 SDIR、WLE、FLE、DSM 和 HPCDIR 为发送器和接收器所共享。对于每一次数据字传送，从发送器的第一个发送移位时钟边沿或接收器的第一个接收移位时钟边沿开始，这些设置在内部被 “ 冻结 ” 。软件必须注意，软件对这些位域的更新要保持一致性 （例如，参考接收器开始指示 PSR.RSIF）。

SCTR
移位控制寄存器 **(34_H)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
SDIR	0	rw	**移位方向** 该位定义发送和接收数据字的移位方向。 0_B LSB 在先，一个数据字的第一位位于位 0。 1_B MSB 在先，一个数据的第一位位于由位域 SCTR.WLE 给定的位置。
PDL	1	rw	**被动数据电平** 该位定义在没有数据可发送时移位数据输出信号的输出电平。 PDL 电平在一个数据字的第一个相关联的发送移位时钟边沿输出。 0_B PDL 为 0。 1_B PDL 为 1。
DSM	[3:2]	rw	**数据移位模式** 该位域控制如何移入和移出接收和发送数据。 00_B 接收和发送移位数据每次通过 DX0 和 DOUT0 移入和移出 1 位。 01_B 保留。 10_B 接收和发送数据每次通过两个输入级 (DX0 和 DX3) 和 DOUT[1:0] 分别移入和移出 2 位。 11_B 接收和发送数据每次通过四个输入级 (DX0, DX[5:3]) 和 DOUT[3:0] 分别移入和移出 4 位。 *注：双位和四位输出模式仅用于 SSC 协议。对于所有其他协议，必须总是向 DSM 写入 00_B。*

域	位	类型	描述
HPCDIR	4	rw	**端口控制方向** 该位定义端口引脚的方向，这些引脚允许硬件引脚控制 (CCR.PCEN = 1)。 0_B 硬件引脚控制功能被使能的引脚被选择为输入模式。 1_B 硬件引脚控制功能被使能的引脚被选择为输出模式。
DOCFG	[7:6]	rw	**数据输出配置** 该位定义内部移位数据值与数据输出信号 DOUTx 之间的关系。 $X0_B$ DOUTx = 移位数据值 $X1_B$ DOUTx = 反相的移位数据值
TRM	[9:8]	rw	**发送模式** 该位域描述移位控制信号如何被 DSU 解释。仅在移位控制信号有效时才可能进行数据传送。 00_B 移位控制信号被视为无效，不可能进行数据帧传送。 01_B 移位控制信号在 1 电平时被视为有效。这是允许数据传送的设置。 10_B 移位控制信号在 0 电平时被视为有效。在使用一个低电平有效信号的情况下，建议避免使用该设置，可使用 DX2 级的反相功能。 11_B 移位控制信号被视为有效，而不参看实际信号电平。在该信号的每个边沿后，都可以进行数据帧传送。
FLE	[21:16]	rwh	**帧长度** 该位域定义在一个数据帧中要传送多少位。一个数据帧可包含多个串接起来的数据字。 如果 TCSR.FLEMD = 1，该值可被数据处理机自动更新。

域	位	类型	描述
WLE	[27:24]	rwh	**字长** 对于接发数据，该位域定义接收和发送的数据字长度（每个数据字中被传送的位数）。数据字在数据缓冲器中的位置总是右对齐 [从 WLE 到 0]。 如果 TCSR.WLEMD = 1，该值可被数据处理机自动更新。 0_H 数据字包含 1 个数据位，位于位 0。 1_H 数据字包含 2 个数据位，位于位 [1:0]。 ... E_H 数据字包含 15 个数据位，位于位 [14:0]。 F_H 数据字包含 16 个数据位，位于位 [15:0]。
0	5, [15:10], [23:22], [31:28]	r	**保留** 读访问返回 0；应写入 0。

15.11.7.2 发送控制和状态寄存器

数据传送由寄存器 TCSR 控制和监视。

TCSR

发送控制 / 状态寄存器 **(38_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**TE**	**TVC**	**TV**	**0**	**TSOF**	**0**							
r			rh	rh	rh	r	rh	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		**WA**	**TDVTR**	**TDEN**		**0**	**TDSSM**	**TDV**	**EOF**	**SOF**	**HPCMD**	**WAMD**	**FLEMD**	**SELMD**	**WLEMD**
r		rwh	rw	rw		r	rw	rh	rwh	rw	rw	rw	rw	rw	rw

域	位	类型	描述
WLEMD	0	rw	**WLE 模式** 该位控制数据器是否用发送控制信息 TCI[3:0] 和 TCI[4]（参见页 15-27）自动更新位域 SCTR.WLE 和位 TCSR.EOF。如果该机制被使能，当通过写一个发送缓冲器输入单元 TBUFx 或通过一个可选的数据缓冲区向寄存器 TBUF 加载新数据时，会发生一次自动更新。 0_B 使能对 SCTR.WLE 和 TCSR.EOF 的自动更新。 1_B 禁止对 SCTR.WLE 和 TCSR.EOF 的自动更新。
SELMD	1	rw	**选择模式** 该位主要用于 SSC 协议。它控制数据处理机是否通过发送控制信息 TCI[4:0] 和清除位域 PCR.CTR[23:21]（见页 15-27）自动更新位域 PCR.CTR[20:16]。如果该机制被使能，当通过写一个发送缓冲器输入单元 TBUFx 或通过一个可选的数据缓冲区向寄存器 TBUF 加载新数据时，会发生一次自动更新。 0_B 禁止对 PCR.CTR[23:16] 的自动更新。 1_B 使能对 PCR.CTR[23:16] 的自动更新。
FLEMD	2	rw	**FLE 模式** 该位控制数据处理机是否通过发送控制信息 TCI[4:0] 和清除位 SCTR.FLE[5]（见页 15-27）来自动更新位域 SCTR.FLE[4:0]。如果该机制被使能，当通过写一个发送缓冲器输入单元 TBUFx 或通过一个可选的数据缓冲区向寄存器 TBUF 加载新数据时，会发生一次自动更新。 0_B 禁止对 FLE 的自动更新。 1_B 使能对 FLE 的自动更新。
WAMD	3	rw	**WA 模式** 该位主要用于 IIS 协议。它控制数据处理机是否通过发送控制信息 TCI[4]（见页 15-27）来自动更新位 TCSR.WA。如果该机制被使能，当通过写一个发送缓冲器输入单元 TBUFx 或通过一个可选的数据缓冲区向寄存器 TBUF 加载新数据时，会发生一次自动更新。 0_B 禁止对位 WA 的自动更新。 1_B 使能对位 WA 的自动更新。

域	位	类型	描述
HPCMD	4	rw	**硬件端口控制模式** 该位主要用于双位和四位 SSC 协议。它控制数据处理机是否通过发送控制信息 TCI[1:0] 和 TCI[2]（见页 15-27）来自动更新位域 SCTR.DSM 和位 SCTR.HPCDIR。如果该机制被使能，当通过写一个发送缓冲器输入单元 TBUFx 或通过一个可选的数据缓冲区向寄存器 TBUF 加载新数据时，会发生一次自动更新。 0_B 禁止对位域 SCTR.DSM 和位 SCTR.HPCDIR 的自动更新。 1_B 使能对位域 SCTR.DSM 和位 SCTR.HPCDIR 的自动更新。
SOF	5	rw	**帧起始** 该位仅用于 SSC 协议，在其他情况下被忽略。它指示位于 TBUF 中的数据字被视为一个新 SSC 帧的第一个字，如果该字为发送有效 (TCSR.TDV = 1)。当 TBUF 数据字被传送到发送移位寄存器时，该位被清零。 0_B 位于TBUF中的数据字不被视为一帧中的第一个字。 1_B 位于 TBUF 中的数据字被视为一帧中的第一个字。当前运行的帧结束且 MSLS 变为无效 (遵守所编程的延迟)。
EOF	6	rwh	**帧结束** 该位仅用于 SSC 协议，在其他情况下被忽略。如果位 WLEMD = 1，它可被数据处理机自动修改。该位指示 TBUF 中的数据字被视为一个 SSC 帧的最后一个字。如果是最后一个字， MSLS 信号在本次传送结束并经过所编程的延迟后变为无效。当 TBUF 数据字被传送到发送移位寄存器时，该位被清零。 0_B TBUF 中的数据字不被视为一个 SSC 帧的最后一个字。 1_B TBUF 中的数据字被视为一个 SSC 帧的最后一个字。

域	位	类型	描述
TDV	7	rh	**发送数据有效** 该位指示位于发送数据缓冲器 TBUF 中的数据字被视为发送有效。 TBUF 数据字只能在 TDV = 1 时被发出。当数据被传送到 TBUF 时 (通过写一个发送缓冲器输入单元 TBUFx 或者通过可选的旁路或 FIFO 机制)，该位被置 1。 0_B 在 TBUF 中的数据字被认为是发送无效的。 1_B 在 TBUF 中的数据字被认为是发送有效的，并且发送开始是可能的。当 TDV = 1 时，新数据应当被写入 TBUFx 中的一位。
TDSSM	8	rw	**TBUF 数据单次模式** 该位定义将数据字 TBUF 视为永久有效还是应该只发送一次。 0_B TBUF中的数据字在被加载到发送移位寄存器后仍被视为有效。加载 TBUF 数据到移位寄存器这一操作并不清除 TDV。 1_B TBUF中的数据字在被加载到发送移位寄存器后被视为有效。在 ASC 和 IIC 模式， TDV 通过 TBI 事件清除；而在 SSC 和 IIS 模式，TDV 通过 RSI 事件清除。如果使用一个可选的数据缓冲区，必须编程使 TDSSM = 1。
TDEN	[11:10]	rw	**TBUF 数据使能** 该位域控制发送缓冲器 TBUF 中的数据字的发送开始。 00_B TBUF 中的数据字的发送开始被禁止。如果一次发送开始，将发送被动数据电平。 01_B 如果 TDV = 1，TBUF 中的数据字的发送可以开始。 10_B 如果 TDV = 1 并且 DX2S = 0，TBUF 中的数据字的发送可以开始。 11_B 如果 TDV = 1 并且 DX2S = 1，TBUF 中的数据字的发送可以开始。

域	位	类型	描述
TDVTR	12	rw	**TBUF 数据有效触发** 对于事件驱动的传送开始，例如基于定时器或出现在输入引脚上的一个事件，如果触发信号 DX2T 被激活，该位使能传送触发单元以置位 TCSR.TE。。对于使用输入级 DX2 进行数据移位的协议，位 TDVTR 必须为 0。 0_B 位 TCSR.TE 被永久性置 1。 1_B 当 TDV = 1 时，如果 DX2T 被激活，位 TCSR.TE 被置 1。
WA	13	rwh	**字地址** 该位仅用于 IIS 协议，在其他情况下被忽略。如果位 WAMD = 1，该位可被数据处理机自动修改。位 WA 定义将 TBUF 中存储的数据发给哪一个通道。 0_B TBUF 中的数据字将在检测到 WA 的下降沿后被发送（参见 PSR.WA）。 1_B TBUF 中的数据字将在检测到 WA 的上升沿后被发送（参见 PSR.WA）。
TSOF	24	rh	**发送的帧起始** 该位指示最近开始发送的数据字是否为一个新数据帧的第一个数据字。该位在每个数据字发送开始时被更新。 0_B 最近开始发送的数据字不是一个数据帧的第一个字。 1_B 最近开始发送的数据字是一个数据帧的第一个字。
TV	26	rh	**发送有效** 该位代表发送缓冲区发生下溢，并指示最近开始的一次数据字发送是否使用一个来自发送缓冲器 TBUF 的有效数据字。该位在每个数据字发送开始时被更新。 0_B 在没有有效数据可用的情况下开始了最近一次数据字发送。其结果是，数据字的发送使用被动电平 (SCTR.PDL) 开始。 1_B 最近开始的一次数据字发送使用来自TBUF的有效数据。

域	位	类型	描述
TVC	27	rh	**发送有效累积** 该位累积发送缓冲区下溢指示标志 TV。它和位 TV 一起被自动清零，必须通过写 FMR.ATVC = 1 置位。 0_B 自 TVC 被置位以来，至少发生了一次数据缓冲区下溢条件。 1_B 自 TVC 被置位以来，尚未发生数据缓冲区下溢条件。
TE	28	rh	**触发事件** 如果传送触发机制被使能并且 TCSR.TDV = 1，该位指示检测到一个触发事件。如果事件触发机制被禁止，则位 TE 被永久性置 1。该位通过写 FMR.MTDV = 10_B 清零。当位于 TBUF 中的数据字被加载到移位寄存器时，该位也会被清零。 0_B 未检测到触发事件。 TBUF 中数据字的发送不能开始。 1_B 检测到触发事件 (或触发机制被关闭)。 TBUF 中数据字的发送可以开始。
0	9, [23:14], 25, [31:29]	r	**保留** 读访问返回 0 ；应写入 0。

15.11.7.3 标志修改寄存器

标志修改寄存器 FMR 允许仅通过写访问来修改与数据处理相关的控制和状态标志。对 FMR 中所有位的读访问总是返回 0。

另外，USIC 通道的服务请求输出可用软件激活 (这种激活通过写访问触发，并且自动变为无效)。

FMR
标志修改寄存器 **(68_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0										SIO5	SIO4	SIO3	SIO2	SIO1	SIO0
r										w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CRDV1	CRDV0	0									ATVC	0		MTDV	
w	w	r									w	r		w	

域	位	类型	描述
MTDV	[1:0]	w	**修改发送数据有效** 对该位域进行写操作可修改位 TCSR.TDV 和 TCSR.TE，以通过软件控制一次数据字发送的开始。 00_B 无操作。 01_B 位 TDV 被置 1， TE 保持不变。 10_B 位 TDV 和 TE 被清零。 11_B 保留
ATVC	4	w	**激活位 TVC** 对该位进行写操作可置位 TCSR.TVC，以开始一次新的发送缓冲区下溢条件累积。 0_B 无操作。 1_B 位 TCSR.TVC 被置 1。
CRDV0	14	w	**RBUF0 的清除位 RDV** 向该位写 1 可清除位 RBUF01SR.RDV00 和 RBUF01SR.RDV10，以声明 RBUF0 中的接收数据不再有效 (模拟一次读操作)。 0_B 无操作 1_B 位 RBUF01SR.RDV00 和 RBUF01SR.RDV10 被清零。

域	位	类型	描述
CRDV1	15	w	**RBUF1 的清除位 RDV** 向该位写 1 可清除位 RBUF01SR.RDV01 和 RBUF01SR.RDV11 以声明 RBUF1 中的接收数据不再有效 (模拟一次读操作)。 0_B 无操作 1_B 位 RBUF01SR.RDV01 和 RBUF01SR.RDV11 被清零。
SIO0, **SIO1,** **SIO2,** **SIO3,** **SIO4,** **SIO5**	16, 17, 18, 19, 20, 21	w	**置位中断输出 SRx** 向该位域写 1 会激活 USIC 通道的服务请求输出 SRx。该操作不影响其他 USIC 通道的服务请求输出。 0_B 无操作。 1_B 服务请求输出 SRx 被激活。
0	[3:2], [13:5], [31:22]	r	**保留** 读访问返回 0 ；应写入 0。

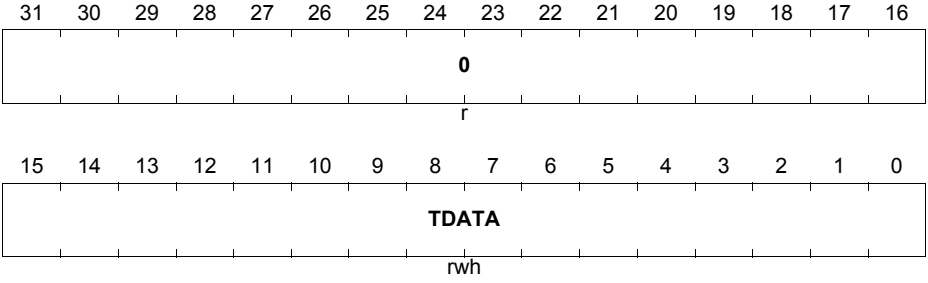
15.11.8 数据缓冲区寄存器

15.11.8.1 发送缓冲区单元

32 个独立的数据输入单元，从 TBUF00 到 TBUF31，可用作发送缓冲区数据项的地址单元。被写入到其中一个单元的数据会出现在一个公用寄存器 TBUF 中。另外，被寻址的数据输入单元的标号 [31:0] 的 5 位编码代表发送控制信息 TCI (更详细的信息请参见描述协议的各节)。

内部的发送缓冲区寄存器 TBUF 包含下一次要发送的数据字，该数据字将被加载到发送移位寄存器。从 TBUF00 到 TBUF31 的所有位地址都可被读出。

TBUFx (x = 00-31)
发送缓冲区输入单元 x **(80_H + x*4)** **复位值 : 0000 0000_H**

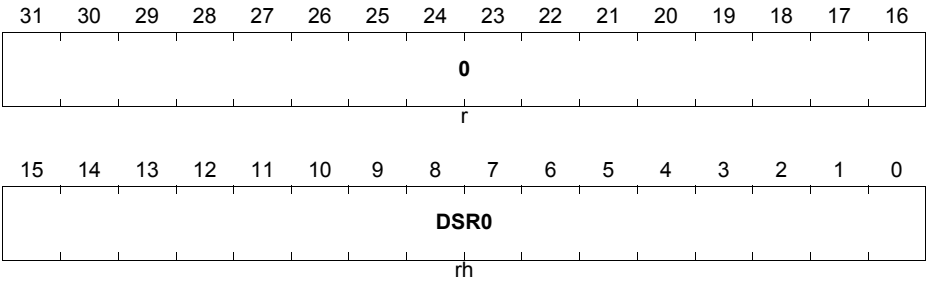


域	位	类型	描述
TDATA	[15:0]	rwh	**发送数据** 该位域包含将要发送的数据 (读视图)。 至少要对 TDATA 的低字节进行一次数据写操作才能置位 TCSR.TDV。
0	[31:16]	r	**保留** 读访问返回 0 ；应写入 0。

15.11.8.2 接收缓冲器寄存器 RBUF0 和 RBUF1

接收缓冲器寄存器 RBUF0 包含从 RSR0[3:0] 接收的数据。一次读操作不会将接收数据的状态从 “ 未读 = 有效 ” 改变为 “ 已读 = 无效 ” 。

RBUF0
接收缓冲器寄存器 0 **(5C$_H$)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
DSR0	[15:0]	rh	**移位寄存器 0[3:0] 的数据**
0	[31:16]	r	**保留** 读访问返回 0 ；应写入 0。

接收缓冲器寄存器 RBUF1 包含从 RSR1[3:0] 接收的数据。一次读操作不会将接收数据的状态从 “ 未读 = 有效 ” 改变为 “ 已读 = 无效 ” 。

RBUF1
接收缓冲器寄存器 1 **(60_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSR1															
rh															

域	位	类型	描述
DSR1	[15:0]	rh	**移位寄存器 1[3:0] 的数据**
0	[31:16]	r	**保留** 读访问返回 0；应写入 0。

接收缓冲器状态寄存器 RBUF01SR 提供位于接收缓冲器寄存器 RBUF0 和 RBUF1 中的数据的状态。

RBUF01SR
接收缓冲器 01 状态寄存器 **(64_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DS1	**RDV 11**	**RDV 10**	**0**			**PER R1**	**PAR 1**	**0**	**SOF 1**	**0**		**WLEN1**			
rh	rh	rh	r			rh	rh	r	rh	r		rh			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DS0	**RDV 01**	**RDV 00**	**0**			**PER R0**	**PAR 0**	**0**	**SOF 0**	**0**		**WLEN0**			
rh	rh	rh	r			rh	rh	r	rh	r		rh			

域	位	类型	描述
WLEN0	[3:0]	rh	**RBUF0 中接收的数据字长度** 该位域指示存储在 RBUF0 中的最新数据字内接收了多少位。该数值指示必须将多少个数据位视为接收数据，而 RBUF0 中的其他位被自动清零。接收的数据位总是右对齐的。 对于除了双位和四位 SSC 以外的所有协议模式， 接收的数据字长度 = WLEN0 + 1。 对于双位 SSC 模式， 接收的数据字长度 = WLEN0 + 2。 对于四位 SSC 模式， 接收的数据字长度 = WLEN0 + 4。
SOF0	6	rh	**RBUF0 中的帧起始** 该位指示 RBUF0 中的数据字是否为一个数据帧的第一个数据字。 0_B RBUF0 中的数据不是一个数据帧的第一个数据字。 1_B RBUF0 中的数据是一个数据帧的第一个数据字。
PAR0	8	rh	**RBUF0 中的协议相关参数** 该位指示协议相关参数的数值。该值的含义取决于所选择的协议，并且会在 RBUF0 的数据字中加入补充信息。 该位的含义在相关协议的小节中描述。
PERR0	9	rh	**RBUF0 中的协议相关错误** 该位指示协议相关参数是否满足预期值。该值的含义取决于所选择的协议，并且会在 RBUF0 的数据字中加入补充信息。 该位的含义在相关协议的小节中描述。 0_B 接收的协议相关参数 PAR 与预期值匹配。该数据字的接收会置位 PSR.RIF 并产生一个接收中断。 1_B 接收的协议相关参数 PAR 与预期值不匹配。该数据字的接收会置位 PSR.AIF 并产生一个备用接收中断。

域	位	类型	描述
RDV00	13	rh	**RBUF0 中的接收数据有效** 该位指示寄存器 RBUF0 的数据内容的状态。该位与位 RBUF01SR.RDV10 完全相同，并且允许读取接收缓冲器寄存器的信息。当一个新数据被保存到 RBUF0 时，该位被置 1；当通过 RBUF 将该数据读出时，该位被自动清零。 0_B 寄存器 RBUF0 不包含尚未读出的数据。 1_B 寄存器 RBUF0 包含尚未读出的数据。
RDV01	14	rh	**RBUF1 中的接收数据有效** 该位指示寄存器 RBUF1 的数据内容的状态。该位与位 RBUF01SR.RDV11 完全相同，并且允许读取接收缓冲器寄存器的信息。当一个新数据被保存到 RBUF1 时，该位被置 1；当通过 RBUF 将该数据读出时，该位被自动清零。 0_B 寄存器 RBUF1 不包含尚未读出的数据。 1_B 寄存器 RBUF1 包含尚未读出的数据。
DS0	15	rh	**数据源** 该位指示哪一个接收缓冲器寄存器 (RBUF0 或 RBUF1) 在寄存器 RBUF(D) 和 RBUFSR 中是当前可见的 (对于相关状态信息)。该位指示哪一个缓冲器包含最老的数据 (最先接收的数据)。该位与位 RBUF01SR.DS1 完全相同，并允许读取接收缓冲器寄存器的信息。 0_B 寄存器 RBUF 包含 RBUF0 的数据 (相关联的状态信息也是如此)。 1_B 寄存器 RBUF 包含 RBUF1 的数据 (相关联的状态信息也是如此)。
WLEN1	[19:16]	rh	**RBUF1 中的接收数据字长度** 该位域指示存储在 RBUF1 中的最新数据字内接收了多少位。该数值指示必须将多少个数据位视为接收数据，而 RBUF1 中的其他位被自动清零。接收的数据位总是右对齐的。 对于除了双位和四位 SSC 以外的所有协议模式， 接收的数据字长度 = WLEN1 + 1。 对于双位 SSC 模式， 接收的数据字长度 = WLEN1 + 2。 对于四位 SSC 模式， 接收的数据字长度 = WLEN1 + 4。

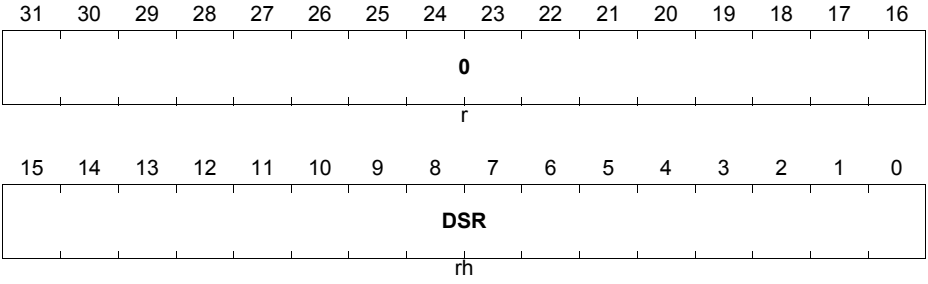
域	位	类型	描述
SOF1	22	rh	**RBUF1 中帧的开始** 该位指示 RBUF1 中的数据字是否为一个数据帧的第一个数据字。 0_B RBUF1 中的数据不是一个数据帧的第一个数据字。 1_B RBUF1 中的数据是一个数据帧的第一个数据字。
PAR1	24	rh	**RBUF1 中的协议相关参数** 该位指示协议相关参数的数值。该值的含义取决于所选择的协议，并且会在 RBUF1 的数据字中加入补充信息。 该位的含义在相关协议的小节中描述。
PERR1	25	rh	**RBUF1 中协议相关错误** 该位指示协议相关参数是否满足预期值。该值的含义取决于所选择的协议，并且会在 RBUF1 的数据字中加入补充信息。 该位的含义在相关协议的小节中描述。 0_B 接收的协议相关参数 PAR 与预期值匹配。该数据字的接收会置位 PSR.RIF 并产生一个接收中断。 1_B 接收的协议相关参数 PAR 与预期值不匹配。该数据字的接收会置位 PSR.AIF 并产生一个备用接收中断。
RDV10	29	rh	**RBUF0 中的接收数据有效** 该位指示寄存器 RBUF0 的数据内容的状态。该位与位 RBUF01SR.RDV00 完全相同，并且允许读取接收缓冲器寄存器的信息。 0_B 寄存器 RBUF0 不包含尚未读出的数据。 1_B 寄存器 RBUF0 包含尚未读出的数据。
RDV11	30	rh	**RBUF1 中的接收数据有效** 该位指示寄存器 RBUF1 的数据内容的状态。该位与位 RBUF01SR.RDV01 完全相同，并且允许读取接收缓冲器寄存器的信息。 0_B 寄存器 RBUF1 不包含尚未读出的数据。 1_B 寄存器 RBUF1 包含尚未读出的数据。

域	位	类型	描述
DS1	31	rh	**数据源** 该位指示哪一个接收缓冲器寄存器 (RBUF0 或 RBUF1) 在寄存器 RBUF(D) 和 RBUFSR 中是当前可见的 (对于相关状态信息)。该位指示哪一个缓冲器包含最老的数据 (最先接收的数据)。该位与位 RBUF01SR.DS0 完全相同，并且允许读取接收缓冲器寄存器的信息。 0_B 寄存器 RBUF 包含 RBUF0 的数据 (相关联的状态信息也是如此)。 1_B 寄存器 RBUF 包含 RBUF1 的数据 (相关联的状态信息也是如此)。
0	[5:4], 7, [12:10], [21:20], 23, [28:26]	r	**保留** 读访问返回 0 ；应写入 0。

15.11.8.3 接收缓冲器寄存器 RBUF、 RBUFD 和 RBUFSR

接收缓冲器寄存器 RBUF 中为 RBUF0 或 RBUF1 的内容，取决于接收顺序。可从 RBUF 读出的数据总是为两个接收缓冲区中最老的数据 (最先接收的数据)。建议从 RBUF 中读取接收的数据而不是从 RBUF0/1 中读取。在读取至少是 RBUF 的低字节时，接收数据的状态自动从 “ 未读 = 有效 ” 改变为 “ 已读 = 无效 ”， RBUF 中的内容被更新，下一个接收的数据字在 RBUF 中将变为可见。

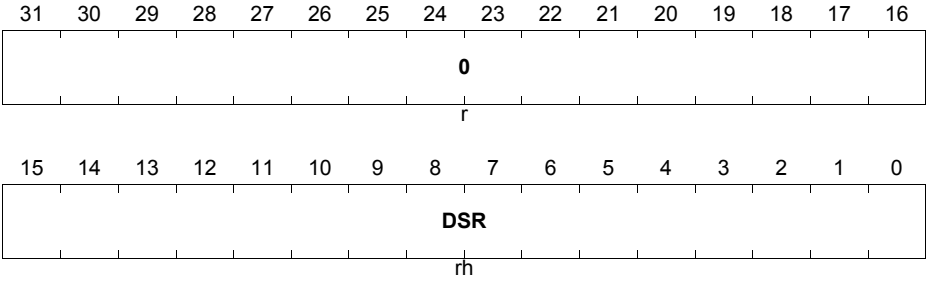
RBUF
接收器缓冲器寄存器 **(54_H)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
DSR	[15:0]	rh	**接收数据** 该位域包含 RBUF0 或 RBUF1 的内容，取决于接收顺序。
0	[31:16]	r	**保留** 读访问返回 0；应写入 0。

如果使用一个调试器来监视接收的数据，则必须使自动更新机制失活，以保证数据的一致性。因此，对于调试功能，有一个接收器缓冲器寄存器 RBUFD 可用。该寄存器类似于 RBUF，但没有通过读操作触发的自动更新机制。因此，一个调试器（或其他监视功能）可读取 RBUFD 而不干扰接收序列。

RBUFD
用于调试器的接收器缓冲器寄存器 (58_H) 复位值：$0000\ 0000_H$



域	位	类型	描述
DSR	[15:0]	rh	**来自移位寄存器的数据** 同 RBUF.DSR，但在一次读操作后不释放缓冲器。
0	[31:16]	r	**保留** 读访问返回 0；应写入 0。

接收缓冲器状态寄存器RBUFSR提供位于接收缓冲器RBUF和RBUFD中的数据的状态。如果位 RBUF01SR.DS0（或 RBUF01SR.DS1）为 0，则 RBUF01SR 的低 16 位的内容在 RBUFSR 中被监视；否则， RBUF01SR 的高 16 位内容被监视。

RBUFSR
接收缓冲器状态寄存器 **(50_H)** **复位值：0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DS	RDV1	RDV0	0			PERR	PAR	0	SOF	0		WLEN			
rh	rh	rh	r			rh	rh	r	rh	r		rh			

域	位	类型	描述
WLEN	[3:0]	rh	**RBUF 或 RBUFD 中接收的数据字长度** 描述见 RBUF01SR.WLEN0 或 RBUF01SR.WLEN1。
SOF	6	rh	**RBUF 或 RBUFD 中的帧起始** 描述见 RBUF01SR.SOF0 或 RBUF01SR.SOF1。
PAR	8	rh	**RBUF 或 RBUFD 中的协议相关参数** 描述见 RBUF01SR.PAR0 或 RBUF01SR.PAR1。
PERR	9	rh	**RBUF 或 RBUFD 中的协议相关错误** 描述见 RBUF01SR.PERR0 或 RBUF01SR.PERR1。
RDV0	13	rh	**RBUF 或 RBUFD 中的接收数据有效** 描述见 RBUF01SR.RDV00 或 RBUF01SR.RDV10。
RDV1	14	rh	**RBUF or RBUFD 中的接收数据有效** 描述见 RBUF01SR.RDV01 或 RBUF01SR.RDV11。
DS	15	rh	**RBUF 或 RBUFD 的数据源** 描述请参见 RBUF01SR.DS0 或 RBUF01SR.DS1。
0	[5:4], 7, [12:10], [31:16]	r	**保留** 读访问返回 0；应写入 0。

15.11.9 FIFO 缓冲区和旁路寄存器

15.11.9.1 旁路寄存器

对旁路数据寄存器 (至少是低字节) 进行一次写操作会设置 BYPCR.BDV = 1(旁路数据被标记为有效)。

BYP
旁路数据寄存器 **(100_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BDATA															
rw															

位 (域)	长度	类型	描述
BDATA	[15:0]	rw	**旁路数据** 该位域包括旁路数据。
0	[31:16]	r	**保留** 读访问返回 0 ； 应写入 0。

BYPCR
旁路控制寄存器 **(104_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								BHPC			BSELO				
r								rw			rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BDV	0	BPRIO	BDVTR	BDEN		0	BDSSM	0				BWLE			
rh	r	rw	rw	rw		r	rw	r				rw			

域	位	类型	描述
BWLE	[3:0]	rw	**旁路字长** 该位域定义旁路数据的字长。字长由 BWLE + 1 给出，数据字在数据缓冲器中的位置为右对齐 [BWLE 到 0]。 旁路数据字总是被视为长度为 BWLE 的特别帧。编码同 SCTR.WLE。
BDSSM	8	rw	**旁路数据单次模式** 该位定义旁路数据是被视为永久有效还是只传送一次（单次模式）。 0_B 旁路数据在被加载到 TBUF 后仍被视为有效。加载数据到 TBUF 的操作不清除 BDV。 1_B 旁路数据在被加载到 TBUF 后被视为无效。加载数据加载到 TBUF 的操作清除 BDV。
BDEN	[11:10]	rw	**旁路数据使能** 该位域定义是否以及如何使能旁路数据到 TBUF 的传送。 00_B 禁止旁路数据传送。 01_B 使能旁路数据到 TBUF 的传送。如果 BDV = 1，旁路数据根据其优先级被传送到 TBUF。 10_B 使能门控的旁路数据传送。如果 BDV = 1 并且 DX2S = 0，旁路数据根据其优先级被传送到 TBUF。 11_B 使能门控的旁路数据传送，如果 BDV = 1 并且 DX2S = 1，旁路数据根据其优先级被传送到 TBUF。
BDVTR	12	rw	**旁路数据有效触发** 当 DX2T 有效时，该位使能对被标为有效的旁路数据的传送 (用于时间限制或超时用途)。 0_B 位 BDV 不受 DX2T 的影响。 1_B 如果 DX2T 有效，则位 BDV 被置 1。
BPRIO	13	rw	**旁路优先级** 该位定义旁路数据和发送 FIFO 数据之间的优先级。 0_B 发送 FIFO 数据具有比旁路数据更高的优先级。 1_B 旁路数据具有比发送 FIFO 数据更高的优先级。

域	位	类型	描述
BDV	15	rh	**旁路数据有效** 该位定义旁路数据对一次到 TBUF 的传送是否有效。当对寄存器 BYP(至少是低字节) 进行一次写访问时，该位被自动被置 1。软件可通过写 TRBSCR.CBDV 将该位清零。 0_B 旁路数据无效。 1_B 旁路数据有效。
BSELO	[20:16]	rw	**旁路选择输出** 如果旁路数据被传送到 TBUF 并且 TCSR.SELMD = 1，则该位域包含被写到 PCR.CTR[20:16] 的值。 在 SSC 协议中，该位域可用于定义在发送旁路数据时哪一个 SELOx 输出线将被激活。
BHPC	[23:21]	rw	**旁路硬件端口控制** 如果旁路数据被传送到 TBUF 并且 TTCSR.HPCMD = 1，则该位域包含被写到 SCTR[4:2] 的值。 在 SSC 协议中，该位域可用于定义数据移位模式，以及在发送旁路数据时是否通过 CCR.HPCEN = 1 使能硬件端口控制。
0	[7:4], 9, 14, [31:24]	r	**保留** 读访问返回 0 ；应写入 0。

15.11.9.2 通用 FIFO 缓冲区控制寄存器

USICx_CHy 的发送和接收 FIFO 状态信息在寄存器 USICx_CHy.TRBSR 中给出。

在该寄存器中，与发送器缓冲区相关的位只能在使能了发送缓冲区功能 (通过设置 CCFG.TB = 1) 时被写入，否则写访问无效。相似的行为也适用于与接收缓冲区 (通过设置 CCFG.RB = 1 使能) 相关的位。

可通过向寄存器 TRBSCR 中的一个位写 1 来清除发送和接收 FIFO 状态寄存器 TRBSR 中的中断标志 (事件标志，而写 0 会对这些位有影响。用软件向 SRBI、RBERI、ARBI、STBI 或 TBERI 写 1 会置位相应的位，以模拟检测到一个发送 / 接收缓冲区事件，但不会激活任何服务请求输出 (因此，见 FMR.SIOx)。

位 TBUS 和 RBUS 用于测试目的。可通过数据处理软件将其屏蔽。请注意，对于这些位，读操作可返回一个 0 或 1。建议将它们当作 “ 无关 ” 位处理。

TRBSR
发送接收缓冲区状态寄存器 **(114_H)** **复位值 : 0000 0808$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	TBFLVL							0	RBFLVL						
r	rh							r	rh						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	STBT	TBUS	TFULL	TEMPTY	0	TBERI	STBI	0	SRBT	RBUS	RFULL	REMPTY	ARBI	RBERI	SRBI
r	rh	rh	rh	rh	r	rwh	rwh	r	rh	rh	rh	rh	rwh	rwh	rwh

域	位	类型	描述
SRBI	0	rwh	**标准接收缓冲区事件** 该位指示检测到一个标准接收缓冲区事件。该位通过写 TRBSCR.CSRBI = 1 清零。 如果该事件被 RBCTR.SRBIEN 使能，由 RBCTR.SRBINP 选择的服务请求输出 SRx 在检测到一个标准接收缓冲区事件时被激活。 0_B 未检测到标准接收缓冲区事件。 1_B 检测到一个标准接收缓冲区事件。
RBERI	1	rwh	**接收缓冲区错误事件** 该位指示检测到一个接收缓冲区错误事件。该位通过写 TRBSCR.CRBERI = 1 清零。 如果该事件被 RBCTR.RBERIEN 使能，由 PRBCTR.ARBINP 选择的服务请求输出 SRx 在检测到一个接收缓冲区错误事件时被激活。 0_B 未检测到接收缓冲区错误事件。 1_B 检测到一个接收缓冲区错误事件。
ARBI	2	rwh	**可选接收缓冲区事件** 该位指示检测到一个备用接收缓冲区事件。该位通过写 TRBSCR.CARBI = 1 清零。 如果该事件被 RBCTR.ARBIEN 使能，由 PRBCTR.ARBINP 选择的服务请求输出 SRx 在检测到一个备用接收缓冲区事件时被激活。 0_B 未检测到备用接收缓冲区事件。 1_B 检测到备用接收缓冲区事件。

域	位	类型	描述
REMPTY	3	rh	**接收缓冲区空** 该位指示接收缓冲区是否为空。 0_B 接收缓冲区未空。 1_B 接收缓冲区已空。
RFULL	4	rh	**接收缓冲区满** 该位指示接收缓冲区是否已满。 0_B 接收缓冲区未满。 1_B 接收缓冲区已满。
RBUS	5	rh	**接收缓冲区忙** 该位指示接收缓冲区是否正被 FIFO 处理器更新。 0_B 接收缓冲区信息已被完全更新。 1_B 来自 FIFO 存储器的 OUTR 更新正在进行。对 OUTR 的读操作将被延迟。源自前一次读操作的 FIFO 指针还没有更新。
SRBT	6	rh	**标准接收缓冲区事件触发** 该位置 1 时触发一次标准接收缓冲区事件。 如果该事件被 RBCTR.SRBIEN 使能，由 RBCTR.SRBINP 选择的服务请求输出 SRx 会被激活，直到该位被清零。 0_B 一次标准接收缓冲区事件不是用该位触发的。 1_B 一次标准接收缓冲区事件是用该位触发的。
STBI	8	rwh	**标准发送缓冲区事件** 该位指示检测到一个标准发送缓冲区事件。该位通过写 TRBSCR.CSTBI = 1 清零。 如果该事件被 TBCTR.STBIEN 使能，由 TBCTR.STBINP 选择的服务请求输出 SRx 在检测到一个标准发送缓冲区事件时被激活。 0_B 未检测到标准发送缓冲区事件。 1_B 检测到一个标准发送缓冲区事件。
TBERI	9	rwh	**发送缓冲区错误事件** 该位指示检测到一个发送缓冲区错误事件。该位通过写 TRBSCR.CTBERI = 1 清零。 如果该事件被 TBCTR.TBERIEN 使能，由 TBCTR.ATBINP 选择服务请求输出 SRx 在检测到一个发送缓冲区错误事件时被激活。 0_B 未检测到发送缓冲区错误事件。 1_B 检测到发送缓冲区错误事件。

域	位	类型	描述
TEMPTY	11	rh	**发送缓冲区空** 该位指示发送缓冲区是否为空。 0_B 发送缓冲区未空。 1_B 发送缓冲区已空。
TFULL	12	rh	**发送缓冲区满** 该位指示发送缓冲区是否已满。 0_B 发送缓冲区未满。 1_B 发送缓冲区已满。
TBUS	13	rh	**发送缓冲区忙** 该位指示发送缓冲区是否正被 FIFO 处理器更新。 0_B 发送缓冲区信息已被完全更新。 1_B 在写INx后FIFO存储器更新正在进行。对INx的写操作将被延迟。源自前一 INx 写操作的 FIFO 指针还没有更新。
STBT	14	rh	**标准发送缓冲区事件触发** 该位置 1 时触发一次标准发送缓冲区事件。 如果该事件被 TBCTR.STBIEN 使能，由 TBCTR.STBINP 选择的服务请求输出 SRx 会激活，直到该位被清零。 0_B 一次标准发送缓冲区事件不是用该位触发的。 1_B 一次标准发送缓冲区事件是用该位触发的。
RBFLVL	[22:16]	rh	**接收缓冲区填充水平** 该位域指示接收缓冲区的填充水平，对于一个空缓冲区，该值从 0 开始。
TBFLVL	[30:24]	rh	**发送缓冲区填充水平** 该位域指示发送缓冲区的填充水平，对于一个空缓冲区，该值从 0 开始。
0	7, 10, 15, 23, 31	r	**保留** 读访问返回 0；应写入 0。

寄存器 TRBSCR 中的位用于清除寄存器 TRBSR 中的指示位或清除发送或接收缓冲区的 FIFO 机制。读操作总是返回 0。

TRBSCR
发送 / 接收缓冲区状态清除寄存器 **(118_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLUSHTB	FLUSHRB	0			CBDV	CTBERI	CSTBI	0					CARBI	CRBERI	CSRBI
w	w	r			w	w	w	r					w	w	w

域	位	类型	描述
CSRBI	0	w	**清除标准接收缓冲区事件** 0_B 无效。 1_B 清除 TRBSR.SRBI。
CRBERI	1	w	**清除接收缓冲区错误事件** 0_B 无效。 1_B 清除 TRBSR.RBERI。
CARBI	2	w	**清除备用接收缓冲区事件** 0_B 无效。 1_B 清除 TRBSR.ARBI。
CSTBI	8	w	**清除标准发送缓冲区事件** 0_B 无效。 1_B 清除 TRBSR.STBI。
CTBERI	9	w	**清除发送缓冲区错误事件** 0_B 无效。 1_B 清除 TRBSR.TBERI。
CBDV	10	w	**清除旁路数据有效** 0_B 无效。 1_B 清除 BYPCR.BDV。
FLUSHRB	14	w	**清空接收缓冲区** 0_B 无效。 1_B 接收 FIFO 缓冲区被清除 (填充水平被清零，并且输出指针被设置为输入指针值)。该操作只应在 FIFO 缓冲区不参与数据通信时使用。

域	位	类型	描述
FLUSHTB	15	w	**清空发送缓冲区** 0_B 无效。 1_B 发送 FIFO 缓冲区被清除 (填充水平被清零，并且输出指针被设置为输入指针值)。该操作只应在 FIFO 缓冲区不参与数据通信时使用。
0	[7:3], [13:11], [31:16]	r	**保留** 读访问返回 0 ； 应写入 0。

15.11.9.3 发送 FIFO 缓冲区控制寄存器

发送 FIFO 缓冲区由寄存器 TBCTR 控制。 TBCTR 只能在发送缓冲区功能被使能 (通过 CCFG.TB = 1) 后被写入，否则写访问被忽略。

TBCTR
发送缓冲区控制寄存器 **(108_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TBERIEN	**STBIEN**	**0**	**LOF**	**0**	**SIZE**			**0**		**ATBINP**			**STBINP**		
rw	rw	r	rw	r	rw			r		rw			rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STBTEN	**STBTM**	**LIMIT**						**0**		**DPTR**					
rw	rw	rw						r		w					

域	位	类型	描述
DPTR	[5:0]	w	**数据指针** 在为发送 FIFO 缓冲区分配 FIFO 项时，该位域定义发送缓冲区指针的起始值。其读出值总是为 0。如果 SIZE = 0，写 DPTR 时寄存器 TRBPTR 中的两个发送器指针 TDIPTR 和 RTDOPTR 都被更新为所写值，并且缓冲区被认为是空的。当 SIZE > 0 时，对 DPTR 的写访问被忽略，并且该写操作不修改这两个指针。
LIMIT	[13:8]	rw	**中断产生极限** 该位域定义发送 FIFO 缓冲区的目标填充水平，该值用于标准发送缓冲区事件检测。

域	位	类型	描述
STBTM	14	rw	**标准发送缓冲区触发模式** 该位选择标准发送缓冲区事件触发模式。 0_B 触发模式 0： 当 TRBSR.STBT=1 时，无论何时有数据被传送到 TBUF 或向 INx 写入数据 (取决于 TBCTR.LOF 的设置)，都会产生一个标准发送缓冲区事件。当 TRBSR.TBFLVL=TBCTR.LIMIT 时， STBT 被清零。 1_B 触发模式 1： 当 TRBSR.STBT=1 时，无论何时有数据被传送到 TBUF 或向 INx 写入数据 (根据 TBCTR.LOF 设置)，都会产生一个标准发送缓冲区事件。当 TRBSR.TBFLVL=TBCTR.SIZE 时， STBT 被清零。
STBTEN	15	rw	**标准发送缓冲区触发使能** 该位使能 / 禁止通过位 TRBSR.STBT 来触发标准发送缓冲区事件。 0_B 禁止通过位 TRBSR.STBT 来触发标准发送缓冲区事件。 1_B 允许通过位 TRBSR.STBT 来触发标准发送缓冲区事件。
STBINP	[18:16]	rw	**标准发送缓冲区中断节点指针** 该位域定义在发生标准发送缓冲区事件时哪一个服务请求输出 SRx 被激活。 000_B 输出 SR0 被激活。 001_B 输出 SR1 被激活。 010_B 输出 SR2 被激活。 011_B 输出 SR3 被激活。 100_B 输出 SR4 被激活。 101_B 输出 SR5 被激活。 *注： 该位域的所有其他设置被保留。*

域	位	类型	描述
ATBINP	[21:19]	rw	**备用发送缓冲区中断节点指针** 该位域定义在发生发送缓冲区错误事件时哪一个服务请求输出 SRx 被激活。 000_B 输出 SR0 被激活。 001_B 输出 SR1 被激活。 010_B 输出 SR2 被激活。 011_B 输出 SR3 被激活。 100_B 输出 SR4 被激活。 101_B 输出 SR5 被激活。 *注： 该位域的所有其他设置被保留。*
SIZE	[26:24]	rw	**缓冲区大小** 该位域定义分配给发送 FIFO 缓冲区的 FIFO 项数量。 000_B FIFO 机制被禁止。缓冲区不接受任何数据请求。 001_B FIFO 缓冲区包含 2 个 FIFO 项。 010_B FIFO 缓冲区包含 4 个 FIFO 项。 011_B FIFO 缓冲区包含 8 个 FIFO 项。 100_B FIFO 缓冲区包含 16 个 FIFO 项。 101_B FIFO 缓冲区包含 32 个 FIFO 项。 110_B FIFO 缓冲区包含 64 个 FIFO 项。 111_B 保留
LOF	28	rw	**极限值溢出缓冲区事件** 该位定义填充水平与所编程的极限值制之间满足何种关系会导致一次标准发送缓冲区事件。 0_B 当填充水平等于所设置的极限值并且由于一次数据字发送而变得更小时，发生一次标准发送缓冲区事件。 1_B 当填充水平等于所设置的极限值并且由于对一个数据输入单元 INx 的一次写访问而变得更大时，发生一次标准发送缓冲区事件。
STBIEN	30	rw	**标准发送缓冲区中断使能** 该位使能 / 禁止在发生一次标准发送缓冲区事件时产生标准发送缓冲区中断。 0_B 禁止标准发送缓冲区中断产生。 1_B 使能标准发送缓冲区中断产生。

域	位	类型	描述
TBERIEN	31	rw	**发送缓冲区错误中断使能** 该位使能 / 禁止在发生一次发送缓冲区错误事件时 (软件向一个已满发送缓冲区写入) 产生发送缓冲区错误中断。 0_B 禁止发送缓冲区错误中断产生。 1_B 使能发送缓冲区错误中断产生。
0	[7:6], [23:22], 27, 29	r	**保留** 读访问返回 0 ； 应写入 0。

15.11.9.4 接收 FIFO 缓冲区控制寄存器

接收 FIFO 缓冲区由寄存器 RBCTR 控制。该寄存器只能在接收缓冲区功能被使能 (通过 CCFG.RB = 1) 后被写入，否则写访问被忽略。

RBCTR
接收缓冲区控制寄存器 **($10C_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
RBERIEN	SRBIEN	ARBIEN	LOF	RNM	SIZE			RCIM		ARBINP			SRBINP		
rw	rw	rw	rw	rw	rw			rw		rw			rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SRBTEN	SRBTM	LIMIT						0		DPTR					
rw	rw	rw						r		w					

域	位	类型	描述
DPTR	[5:0]	w	**数据指针** 在为接收 FIFO 缓冲区分配 FIFO 项时，该位域定义接收缓冲区指针的起始值。其读出值总是为 0。如果 SIZE = 0，写 DPTR 时寄存器 TRBPTR 中的两个接收器指针 RDIPTR 和 RDOPTR 都被更新为所写值，并且缓冲区被认为是空的。当 SIZE > 0 时，对 DPTR 的写访问被忽略，并且该写操作不修改这两个指针。

域	位	类型	描述
LIMIT	[13:8]	rw	**中断产生极限** 该位域定义接收 FIFO 缓冲区的目标填充水平，该值用于标准接收缓冲区事件检测。
SRBTM	14	rw	**标准接收缓冲区触发模式** 该位选择标准接收缓冲区事件触发模式。 0_B 触发模式 0：当 TRBSR.STBT=1 时，无论何时接收到一个新数据或有数据被读出 (取决于 RBCTR.LOF 的 设置)，都会产生一个标准接收缓冲区事件。当 TRBSR.RBFLVL=RBCTR.LIMIT 时，SRBT 被清零。 1_B 触发模式 1：当 TRBSR.SRBT=1 时，无论何时接收到一个新数据或有数据被读出 (取决于 RBCTR.LOF 的 设置)，都会产生一个标准接收缓冲区事件。当 TRBSR.RBFLVL=0 时， SRBT 被清零。
SRBTEN	15	rw	**标准接收缓冲区触发使能** 该位使能 / 禁止通过位 TRBSR.SRBT 来触发标准接收缓冲区事件。 0_B 禁止通过位 TRBSR.SRBT 来触发标准接收缓冲区事件。 1_B 允许通过位 TRBSR.SRBT 来触发标准接收缓冲区事件。
SRBINP	[18:16]	rw	**标准接收缓冲区中断节点指针** 该位域定义在发生标准接收缓冲区事件时哪一个服务请求输出 SRx 被激活。 000_B 输出 SR0 被激活 001_B 输出 SR1 被激活 010_B 输出 SR2 被激活 011_B 输出 SR3 被激活 100_B 输出 SR4 被激活 101_B 输出 SR5 被激活 *注： 该位域的所有其他设置被保留。*

域	位	类型	描述
ARBINP	[21:19]	rw	**备用接收缓冲区中断节点指针** 该位域定义在发生备用接收缓冲区事件或接收缓冲区错误事件时哪一个服务请求输出 SRx 被激活。 000_B 输出 SR0 被激活 001_B 输出 SR1 被激活 010_B 输出 SR2 被激活 011_B 输出 SR3 被激活 100_B 输出 SR4 被激活 101_B 输出 SR5 被激活 *注： 该位域的所有其他设置被保留。*
RCIM	[23:22]	rw	**接收器控制信息模式** 该位域定义来自接收状态寄存器 RBUFSR 的哪些信息作为 5 位接收器控制信息 RCI[4:0] 被传送到接收 FIFO 缓冲区，并可在寄存器 OUT(D)R 中读出。 00_B RCI[4] = PERR, RCI[3:0] = WLEN 01_B RCI[4] = SOF, RCI[3:0] = WLEN 10_B RCI[4] = 0, RCI[3:0] = WLEN 11_B RCI[4] = PERR, RCI[3] = PAR, RCI[2:1] = 00_B, RCI[0] = SOF
SIZE	[26:24]	rw	**缓冲区大小** 该位域定义分配给接收 FIFO 缓冲区的 FIFO 项数量。 000_B FIFO 机制被禁止。缓冲区不接受任何数据请求。 001_B FIFO 缓冲区包括 2 个 FIFO 项。 010_B FIFO 缓冲区包括 4 个 FIFO 项。 011_B FIFO 缓冲区包括 8 个 FIFO 项。 100_B FIFO 缓冲区包括 16 个 FIFO 项。 101_B FIFO 缓冲区包括 32 个 FIFO 项。 110_B FIFO 缓冲区包括 64 个 FIFO 项。 111_B 保留

域	位	类型	描述
RNM	27	rw	**接收器通知模式** 该位定义接收缓冲区事件模式。接收缓冲区错误事件不受到 RNM 的影响。 0_B 填充水平模式：当填充水平等于所编程的极限值并且因一次对 OUTR 的读访问 (LOF = 0) 或一个新接收的数据字而发生改变时，发生一次标准接收缓冲区事件。 1_B RCI 模式：当寄存器 OUTR 被一个新值更新并且 OUTR.RCI[4] 中的相应值 = 0 时，发生一次标准接收缓冲区事件。如果 OUTR.RCI[4] = 1，则发生一次备用接收缓冲区事件而不是标准接收缓冲区事件。
LOF	28	rw	**极限值溢出缓冲区事件** 在填充水平模式 (RNM = 0)，该位定义填充水平与所编程的极限值之间满足何种关系会导致一次标准接收缓冲区事件。在 RCI 模式 (RNM = 1)，位域 LIMIT 和 LOF 被忽略。 0_B 当填充水平等于所设置的极限值并且由于一次对 OUTR 的读访问而变小时，发生一次标准发送缓冲区事件。 1_B 填充水平等于所设置的极限值并且由于收到一个新数据字而变大时，发生一次标准发送缓冲区事件。
ARBIEN	29	rw	**备用接收缓冲区中断使能** 该位使能 / 禁止在发生备用接收缓冲区事件时产生备用接收缓冲区中断。 0_B 禁止备用接收缓冲区中断产生。 1_B 使能备用接收缓冲区中断产生。
SRBIEN	30	rw	**标准接收缓冲区中断使能** 该位使能 / 禁止在发生标准接收缓冲区事件时产生标准接收缓冲中断。 0_B 禁止标准接收缓冲区中断产生。 1_B 使能标准接收缓冲区中断产生。
RBERIEN	31	rw	**接收缓冲区错误中断使能** 该位使能 / 禁止在发生接收缓冲区错误事件时 (软件读一个空接收缓冲区) 产生接收缓冲错误中断。 0_B 禁止接收缓冲区错误中断产生。 1_B 使能接收缓冲区错误中断产生。

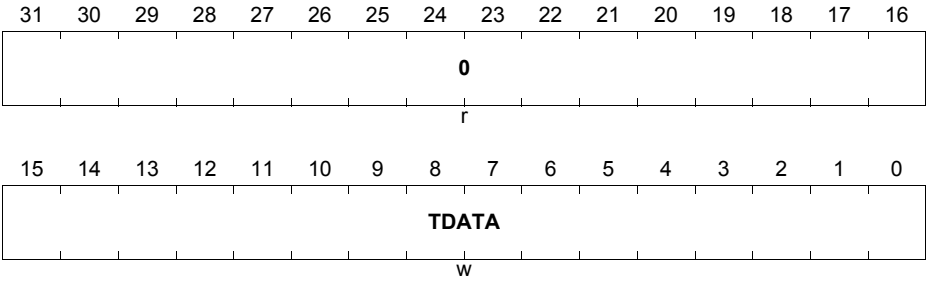
域	位	类型	描述
0	[7:6]	r	**保留** 读访问返回 0；应写入 0。

15.11.9.5 FIFO 缓冲区数据寄存器

32 个独立数据输入单元，从 IN00 到 IN31，可用作发送 FIFO 缓冲区数据项单元的地址。写入到其中一个单元的数据将被保存到发送缓冲区 FIFO 中。另外，被寻址的数据输入单元的标号 [31:0] 的 5 位编码代表发送控制信息 TCI。

如果 FIFO 已满并且有新数据被写入，则这次写访问被忽略，并会报告一个发送缓冲区错误事件。

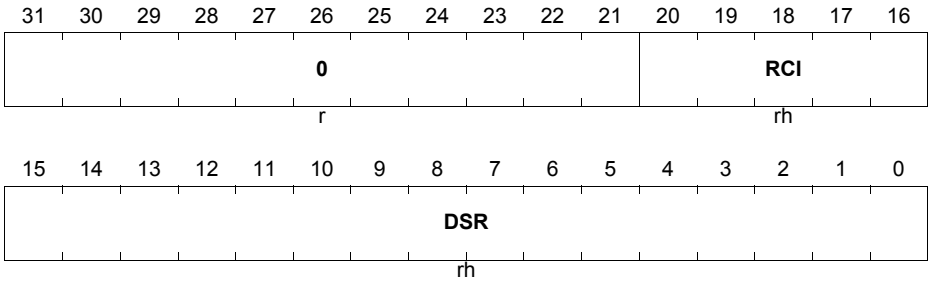
INx (x = 00-31)
发送 FIFO 缓冲区输入单元 x **(180_H + x *4)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
TDATA	[15:0]	w	**发送数据** 该位域包含要被发送的数据 (写视图)，读操作返回 0。 对 TDATA(至少是低字节) 的写操作会导致数据被保存到 FIFO 中。
0	[31:16]	r	**保留** 读访问返回 0；应写入 0。

接收器 FIFO 缓冲区输出寄存器 OUTR 包含位于 FIFO 缓冲区中的最老的接收数据，并且包含接收器控制信息 RCI，而 RCI 包含由 RBCTR.RCIM 选择的信息。对该地址单元的读操作会返回接收的数据。通过对至少是低字节的读访问，数据被声明为已读，并且下一个数据项变为可见。对 OUTR 的写访问被忽略。

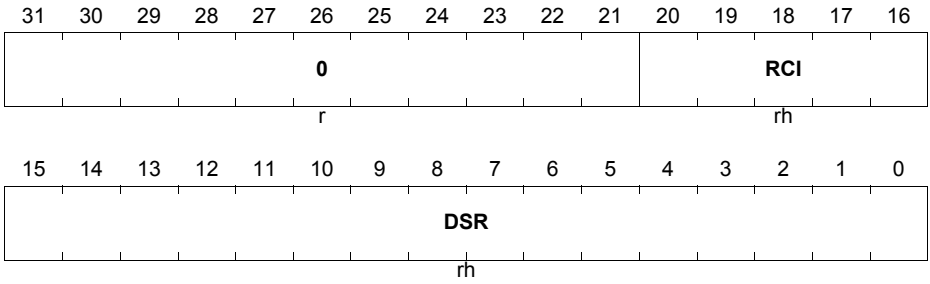
OUTR
接收器缓冲区输出寄存器 **($11C_H$)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
DSR	[15:0]	rh	**接收的数据** 该位域监视位于接收 FIFO 中最老的数据字的内容。读至少是低字节会释放在 DSR 中所指示的缓冲区项。
RCI	[20:16]	rh	**接收器控制信息** 该位域监视与 DSR 相关联的接收器控制信息。 RCI 的位结构取决于位域 RBCTR.RCIM。
0	[31:21]	r	**保留** 读访问返回 0 ；应写入 0。

如果应使用一个调试器来监视位于 FIFO 缓冲区中的接收数据，则不得激活 FIFO 机制，以保证数据的一致性。因此，设置了第二个可用的，命名为 OUTDR(D 即调试器)。该寄存器与原有缓冲输出寄存器 OUTR 具有相同的位域，但无 FIFO 机制。调试器可读该寄存器 (为了监测接收数据流) 而没有损坏数据的风险。对 OUTDR 的写访问被忽略。

OUTDR
用于调试器的接收器缓冲区输出寄存器 (120_H) **复位值 : 0000 0000_H**

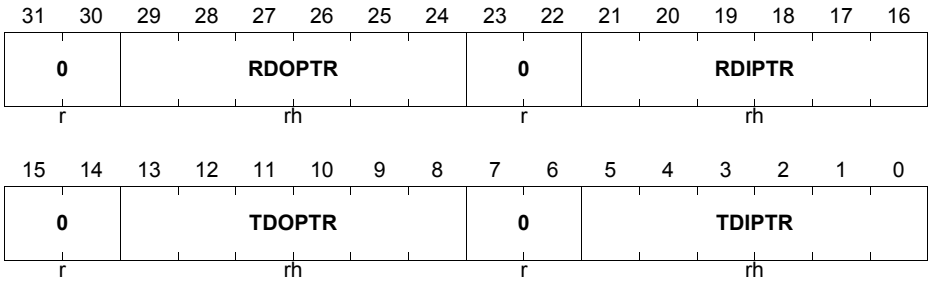


域	位	类型	描述
DSR	[15:0]	rh	**来自移位寄存器的数据** 同 OUTR.DSR，但在一次读操作后不释放缓冲区。
RCI	[20:16]	rh	**来自移位寄存器的接收控制信息** 同 OUTR.RCI。
0	[31:21]	r	**保留** 读访问返回 0；应写入 0。

15.11.9.6 FIFO 缓冲区指针寄存器

发送和接收 FIFO 缓冲区的 FIFO 处理指针位于寄存器 TRBPTR 中。这两个指针由 FIFO 缓冲机制自动处理，并且不需要通过软件修改。因此，软件只能读这些寄存器 (例如用于验证目的)，而写访问被忽略。

TRBPTR
发送 / 接收缓冲区指针寄存器 **(110_H)** **复位值 : 0000 0000_H**



域	位	类型	描述
TDIPTR	[5:0]	rh	**发送器数据输入指针** 该位域指示将被用于来自 INx 地址的下一个发送数据的缓冲区项。
TDOPTR	[13:8]	rh	**发送器数据输出指针** 该位域指示将被用于输出到 TBUF 的下一个发送数据的缓冲区项。
RDIPTR	[21:16]	rh	**接收器数据输入指针** 该位域指示将被用于来自 RBUF 的下一个接收数据的缓冲区项。
RDOPTR	[29:24]	rh	**接收器数据输出指针** 该位域指示将被用于输出到 OUT(D)R 地址的下一个接收数据的缓冲区项。
0	[7:6], [15:14], [23:22], [31:30]	r	**保留** 读访问返回 0；应写入 0。

15.12 互连

XMC1300 器件包含一个具有两个通信通道的 USIC 模块 (USIC0) 。

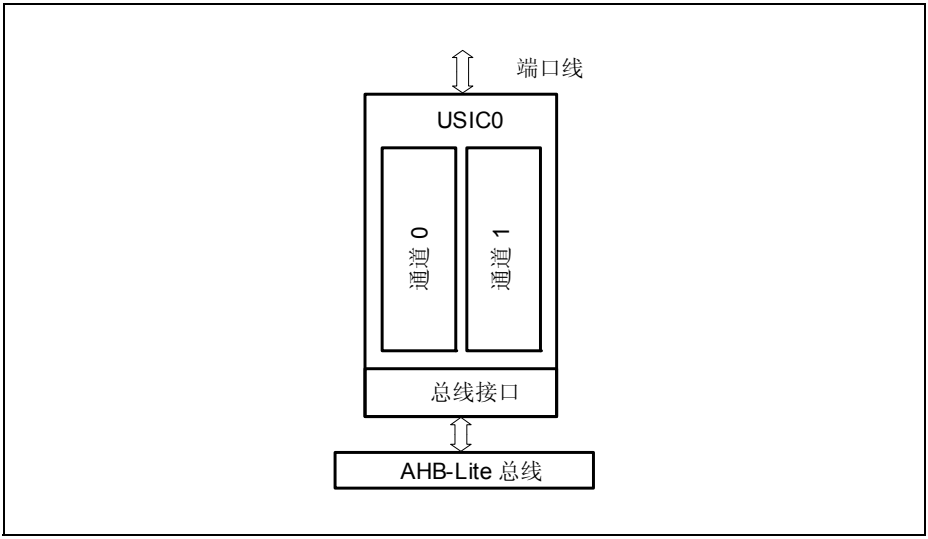


图 15-66 XMC1300 中的 USIC 模块结构

图 15-67 示出了一个 USIC 通道的 I/O 线。XMC1300 器件中的 USIC 通道 I/O 线的引脚分配和内部连接在本节后面给出的表格中定义。命名约定：USICx_CHy 代表 USIC 模块 x 的通道 y。

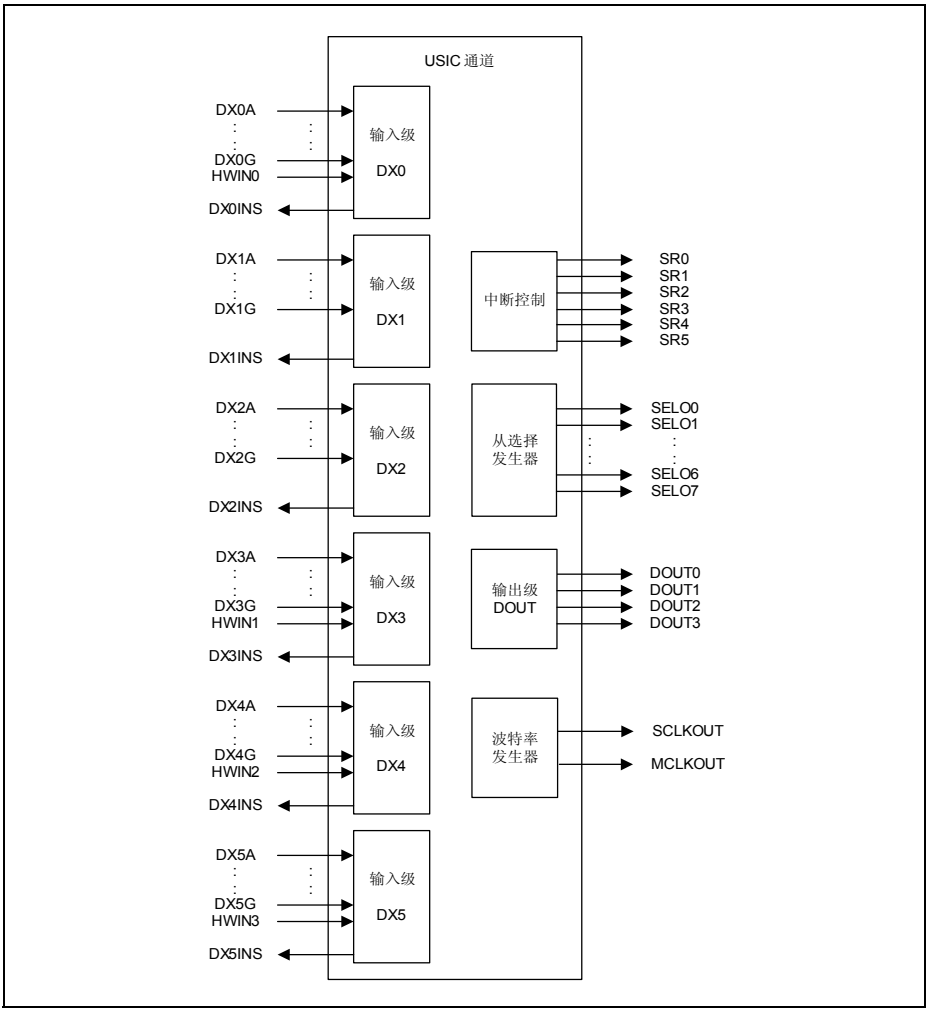


图 15-67　　USIC 通道 I/O 线

一个 USIC 通道的服务请求输出 SR[5:0] 与模块中的另一通道是组合在一起的。因此，每个模块只有 6 个服务请求输出可用。

15.12.1 USIC 模块 0 互连

USIC 模块 0 的互连可分成以下几种：

- **USIC 模块 0 通道 0 互连** (**表 15-22**)
- **USIC 模块 0 通道 1 互连** (**表 15-23**)
- **USIC 模块 0 模块互连** (**表 15-24**)

表 15-22 USIC 模块 0 通道 0 互连

输入 / 输出	I/O	连接到	描述
数据输入 (DX0)			
USIC0_CH0.DX0A	I	P0.14	移位数据输入
USIC0_CH0.DX0B	I	P0.15	移位数据输入
USIC0_CH0.DX0C	I	P1.0	移位数据输入
USIC0_CH0.DX0D	I	P1.1	移位数据输入
USIC0_CH0.DX0E	I	P2.0	移位数据输入
USIC0_CH0.DX0F	I	P2.1	移位数据输入
USIC0_CH0.DX0G	I	USIC0_CH0.DX3INS	移位数据输入
USIC0_CH0.HWIN0	I	P1.0	硬件控制的移位数据输入
时钟输入			
USIC0_CH0.DX1A	I	P0.14	移位时钟输入
USIC0_CH0.DX1B	I	P0.8	移位时钟输入
USIC0_CH0.DX1C	I	P0.7	移位时钟输入
USIC0_CH0.DX1D	I	P1.1	移位时钟输入
USIC0_CH0.DX1E	I	P2.0	移位时钟输入
USIC0_CH0.DX1F	I	USIC0_CH0.DX0INS	移位时钟输入
USIC0_CH0.DX1G	I	USIC0_CH0.DX4INS	环回移位时钟输入
控制输入			
USIC0_CH0.DX2A	I	P0.0	移位控制输入
USIC0_CH0.DX2B	I	P0.9	移位控制输入
USIC0_CH0.DX2C	I	P0.10	移位控制输入
USIC0_CH0.DX2D	I	P0.11	移位控制输入
USIC0_CH0.DX2E	I	P0.12	移位控制输入
USIC0_CH0.DX2F	I	P0.13	移位控制输入
USIC0_CH0.DX2G	I	USIC0_CH0.DX5INS	环回移位控制输入

表 15-22 USIC 模块 0 通道 0 互连 (续表)

输入 / 输出	I/O	连接到	描述
数据输入 (DX3)			
USIC0_CH0.DX3A	I	P2.2	移位数据输入
USIC0_CH0.DX3B	I	P2.4	移位数据输入
USIC0_CH0.DX3C	I	P2.10	移位数据输入
USIC0_CH0.DX3D	I	P2.8	移位数据输入
USIC0_CH0.DX3E	I	P2.6	移位数据输入
USIC0_CH0.DX3F	I	USIC0_CH0.DX5INS	移位数据输入
USIC0_CH0.DX3G	I	USIC0_CH0.DOUT0	移位数据输入
USIC0_CH0.HWIN1	I	P1.1	硬件控制的移位数据输入
数据输入 (DX4)			
USIC0_CH0.DX4A	I	P2.2	移位数据输入
USIC0_CH0.DX4B	I	P2.4	移位数据输入
USIC0_CH0.DX4C	I	P2.10	移位数据输入
USIC0_CH0.DX4D	I	P2.8	移位数据输入
USIC0_CH0.DX4E	I	P2.6	移位数据输入
USIC0_CH0.DX4F	I	USIC0_CH0.DX5INS	移位数据输入
USIC0_CH0.DX4G	I	USIC0_CH0.SCLKOUT	移位数据输入
USIC0_CH0.HWIN2	I	P1.2	硬件控制的移位数据输入
数据输入 (DX5)			
USIC0_CH0.DX5A	I	P2.9	移位数据输入
USIC0_CH0.DX5B	I	P2.3	移位数据输入
USIC0_CH0.DX5C	I	P2.7	移位数据输入
USIC0_CH0.DX5D	I	P2.5	移位数据输入
USIC0_CH0.DX5E	I	P1.4	移位数据输入
USIC0_CH0.DX5F	I	0	移位数据输入
USIC0_CH0.DX5G	I	USIC0_CH0.SELO0	移位数据输入
USIC0_CH0.HWIN3	I	P1.3	硬件控制的移位数据输入
数据输出			

表 15-22 USIC 模块 0 通道 0 互连 (续表)

输入 / 输出	I/O	连接到	描述
USIC0_CH0.DOUT0	O	P0.14 P0.15 P1.0 P1.0 (HW1_OUT) P1.1 P1.5 P2.0 P2.1 USIC0_CH0.DX3G	移位数据输出
USIC0_CH0.DOUT1	O	P1.1 (HW1_OUT)	移位数据输出
USIC0_CH0.DOUT2	O	P1.2 (HW1_OUT)	移位数据输出
USIC0_CH0.DOUT3	O	P1.3 (HW1_OUT)	移位数据输出
时钟输出			
USIC0_CH0.MCLKOUT	O	P0.11	主时钟输出
USIC0_CH0.SCLKOUT	O	P0.7 P0.8 P0.14 P2.0 USIC0_CH0.DX4G	移位时钟输出
控制输出			
USIC0_CH0.SELO0	O	P0.0 P0.9 P1.4 USIC0_CH0.DX5G	移位控制输出
USIC0_CH0.SELO1	O	P0.10 P1.5	移位控制输出
USIC0_CH0.SELO2	O	P0.11	移位控制输出
USIC0_CH0.SELO3	O	P0.12	移位控制输出
USIC0_CH0.SELO4	O	P0.13	移位控制输出
USIC0_CH0.SELO5	O	not connected	移位控制输出
USIC0_CH0.SELO6	O	not connected	移位控制输出
USIC0_CH0.SELO7	O	not connected	移位控制输出
与系统相关的输出			
USIC0_CH0.DX0INS	O	USIC0_CH0.DX1F	选择的 DX0 输入信号

表 15-22 USIC 模块 0 通道 0 互连 (续表)

输入 / 输出	I/O	连接到	描述
USIC0_CH0.DX1INS	O	not connected	选择的 DX1 输入信号
USIC0_CH0.DX2INS	O	CCU40.IN0L	选择的 DX2 输入信号
USIC0_CH0.DX3INS	O	USIC0_CH0.DX0G	选择的 DX3 输入信号
USIC0_CH0.DX4INS	O	USIC0_CH0.DX1G	选择的 DX4 输入信号
USIC0_CH0.DX5INS	O	USIC0_CH0.DX2G USIC0_CH0.DX3F USIC0_CH0.DX4F	选择的 DX5 输入信号

表 15-23 USIC 模块 0 通道 1 互连

输入 / 输出	I/O	连接到	描述
数据输入 (DX0)			
USIC0_CH1.DX0A	I	P1.3	移位数据输入
USIC0_CH1.DX0B	I	P1.2	移位数据输入
USIC0_CH1.DX0C	I	P0.6	移位数据输入
USIC0_CH1.DX0D	I	P0.7	移位数据输入
USIC0_CH1.DX0E	I	P2.11	移位数据输入
USIC0_CH1.DX0F	I	P2.10	移位数据输入
USIC0_CH1.DX0G	I	USIC0_CH1.DX3INS	移位数据输入
USIC0_CH1.HWIN0	I	ERU0.PDOUT0	硬件控制的移位数据输入
时钟输入			
USIC0_CH1.DX1A	I	P1.3	移位时钟输入
USIC0_CH1.DX1B	I	P0.8	移位时钟输入
USIC0_CH1.DX1C	I	P0.7	移位时钟输入
USIC0_CH1.DX1D	I	0	移位时钟输入
USIC0_CH1.DX1E	I	P2.11	移位时钟输入
USIC0_CH1.DX1F	I	USIC0_CH1.DX0INS	移位时钟输入
USIC0_CH1.DX1G	I	USIC0_CH1.DX4INS	环回移位时钟输入
控制输入			
USIC0_CH1.DX2A	I	P0.0	移位控制输入
USIC0_CH1.DX2B	I	P0.9	移位控制输入
USIC0_CH1.DX2C	I	P0.10	移位控制输入

表 15-23 USIC 模块 0 通道 1 互连 (续表)

输入 / 输出	I/O	连接到	描述
USIC0_CH1.DX2D	I	P0.11	移位控制输入
USIC0_CH1.DX2E	I	P1.1	移位控制输入
USIC0_CH1.DX2F	I	P2.0	移位控制输入
USIC0_CH1.DX2G	I	USIC0_CH1.DX5INS	环回移位控制输入
数据输入 (DX3)			
USIC0_CH1.DX3A	I	P2.1	移位数据输入
USIC0_CH1.DX3B	I	P2.9	移位数据输入
USIC0_CH1.DX3C	I	P2.3	移位数据输入
USIC0_CH1.DX3D	I	P2.7	移位数据输入
USIC0_CH1.DX3E	I	P2.5	移位数据输入
USIC0_CH1.DX3F	I	USIC0_CH1.DX5INS	移位数据输入
USIC0_CH1.DX3G	I	USIC0_CH1.DOUT0	移位数据输入
USIC0_CH1.HWIN1	I	0	硬件控制的移位数据输入
数据输入 (DX4)			
USIC0_CH1.DX4A	I	P2.1	移位数据输入
USIC0_CH1.DX4B	I	P2.9	移位数据输入
USIC0_CH1.DX4C	I	P2.3	移位数据输入
USIC0_CH1.DX4D	I	P2.7	移位数据输入
USIC0_CH1.DX4E	I	P2.5	移位数据输入
USIC0_CH1.DX4F	I	USIC0_CH1.DX5INS	移位数据输入
USIC0_CH1.DX4G	I	USIC0_CH1.SCLKOUT	移位数据输入
USIC0_CH1.HWIN2	I	ERU0.PDOUT1	硬件控制的移位数据输入
数据输入 (DX5)			
USIC0_CH1.DX5A	I	P2.2	移位数据输入
USIC0_CH1.DX5B	I	P2.4	移位数据输入
USIC0_CH1.DX5C	I	P2.8	移位数据输入
USIC0_CH1.DX5D	I	P2.6	移位数据输入
USIC0_CH1.DX5E	I	P1.4	移位数据输入
USIC0_CH1.DX5F	I	P1.5	移位数据输入
USIC0_CH1.DX5G	I	USIC0.SR0	移位数据输入

表 15-23 USIC 模块 0 通道 1 互连 (续表)

输入 / 输出	I/O	连接到	描述
USIC0_CH1.HWIN3	I	USIC0_CH1.DOUT0	硬件控制的移位数据输入
数据输出			
USIC0_CH1.DOUT0	O	P0.6 P0.7 P1.2 P1.3 P2.10 P2.11 USIC0_CH1.DX3G USIC0_CH1.HWIN3	移位数据输出
USIC0_CH1.DOUT1	O	无连接	移位数据输出
USIC0_CH1.DOUT2	O	无连接	移位数据输出
USIC0_CH1.DOUT3	O	无连接	移位数据输出
时钟输出			
USIC0_CH1.MCLKOUT	O	P0.6 P0.15	主时钟输出
USIC0_CH1.SCLKOUT	O	P0.8 P1.3 P1.4 P2.1 P2.11 USIC0_CH1.DX4G	移位时钟输出
控制输出			
USIC0_CH1.SELO0	O	P0.0 P0.9 P1.1	移位控制输出
USIC0_CH1.SELO1	O	P0.10 P1.4	移位控制输出
USIC0_CH1.SELO2	O	P0.11 P1.5	移位控制输出
USIC0_CH1.SELO3	O	无连接	移位控制输出
USIC0_CH1.SELO4	O	无连接	移位控制输出
USIC0_CH1.SELO5	O	无连接	移位控制输出
USIC0_CH1.SELO6	O	无连接	移位控制输出

表 15-23 USIC 模块 0 通道 1 互连 (续表)

输入 / 输出	I/O	连接到	描述
USIC0_CH1.SELO7	O	无连接	移位控制输出
与系统相关的输出			
USIC0_CH1.DX0INS	O	USIC0_CH1.DX1F	选择的 DX0 输入信号
USIC0_CH1.DX1INS	O	无连接	选择的 DX1 输入信号
USIC0_CH1.DX2INS	O	CCU40.IN1L	选择的 DX2 输入信号
USIC0_CH1.DX3INS	O	USIC0_CH1.DX0G	选择的 DX3 输入信号
USIC0_CH1.DX4INS	O	USIC0_CH1.DX1G	选择的 DX4 输入信号
USIC0_CH1.DX5INS	O	USIC0_CH1.DX2G USIC0_CH1.DX3F USIC0_CH1.DX4F	选择的 DX5 输入信号

表 15-24 USIC 模块 0 模块互连

输入 / 输出	I/O	连接到	描述
USIC0_SR0	O	NVIC USIC0_CH1.DX5G	中断输出线 (服务请求 SRx)
USIC0_SR[5:1]	O	NVIC	中断输出线 (服务请求 SRx)

模拟外设

16 多功能模 / 数转换器 (VADC)

XMC1300 提供了一系列连接到一个模 / 数转换器集群的模拟输入通道，这些模 / 数转换器采用逐次逼近寄存器 （SAR）原理将模拟输入值 （电压）转换为离散数字值。
XMC1300 中的模 / 数转换器为采样保持转换器，一个集群包含两个采样保持单元，这两个采样保持单元共享一个转换器。
模拟输入通道数量和 ADC 的数量取决于所选产品的类型 （参见 **16-134 页 " 产品的具体配置 "**）。

表 16-1 ADC 一章中的缩略语

缩略语	含义
ADC	模 / 数转换器
ADC 内核	也称为 ADC 组
DNL	微分非线性 （误差）
INL	积分非线性 （误差）
LSB_n	最低有效位：以数字格式表示的最小粒度的模拟值，由 n 位分辨率转换结果的一个最低有效位来表示 （测量范围被等分成 2^n 个区间）
S&H	采样保持
SCU	系统控制单元
SHS	采样保持定序器
TUE	总不可调整误差

16.1 概述

ADC 集群里的每个转换器都可独立于其他的转换器单独工作，每个转换器都由一组专用的寄存器控制并由一组专用的请求源触发。每个通道的结果都可保存到一个通道专用结果寄存器或一个组专用结果寄存器中。

后台请求源可以访问未被分配给任何组请求源的所有模拟输入通道。由后台请求源触发的转换优先级较低，因此后台请求源可被视为额外的后台转换器。

XMC1300 的多功能模 / 数转换器模块（VADC）包含一组转换器单元，每个转换器单元可独立工作，也可以通过一个模拟后台转换器的共用请求源来操作。每个转换器块都配备了一个专用输入多路复用器和专用的请求源，它们一起构成独立的转换器组。

这种基本结构支持面向应用的编程和操作，同时仍然提供对所有资源的访问。 VADC 的所有转换器组几乎完全相同，支持灵活的通道功能分配。

特性列表

ADC 集群的主要特性如下：

- 输入电压范围从 0 V 到模拟电源电压
- 每个通道都可以选择标准参考电压源 （V_{AREF}）和备选参考地 （CH0），以支持比率测量
- 两个独立的采样保持单元，每个采样保持单元支持 8 个模拟输入通道
- 两个 Σ−Δ 保持单元支持过采样
- 外部模拟多路复用器控制，支持可调节的采样时间和通道扫描
- 转换速度和采样时间可调节，以适应不同的传感器输出电平和参考电压
- 转换时间低于 1 µs （取决于转换结果长度和采样时间）
- 灵活的源选择和仲裁
  - 队列请求模式，转换顺序可自由编程 （单次或重复）
  - 每组都能进行可配置的自动扫描转换 （单次或重复）
  - 后台运行的自动扫描模式，支持所有通道 （单次或重复）
  - 转换可以由软件、定时器事件或外部事件触发
  - 撤消 - 插入 - 重新启动模式可降低高优先级通道上的转换延迟
- 强大的结果处理功能
  - 可选择 8/10/12 位的结果宽度
  - 快速比较模式
  - 独立的结果寄存器
  - 边界检测功能，边界值可编程
  - 通过累加多个转换结果来降低数据速率 （累加次数可编程）
  - FIR/IIR 滤波器 （可选择系数）
- 可选择触发事件的灵活服务请求产生 （2 个组专用中断， 2 个共享中断）
- 内建安全功能：自动断线检测能力，检测电平可编程
- 支持调试挂起和省电模式

注： 来自超量程比较器的其他功能也有效 （参见 SCU 一章中的描述）。

表 16-2 VADC 的应用

VADC 使用案例	应用
复杂转换序列的自动调度，包括时间关键型转换的优先化	电机控制，电源转换
对突发型高速转换的有效结果处理	高动态输入信号
多达 4 个输入信号的同步采样	多相电流测量

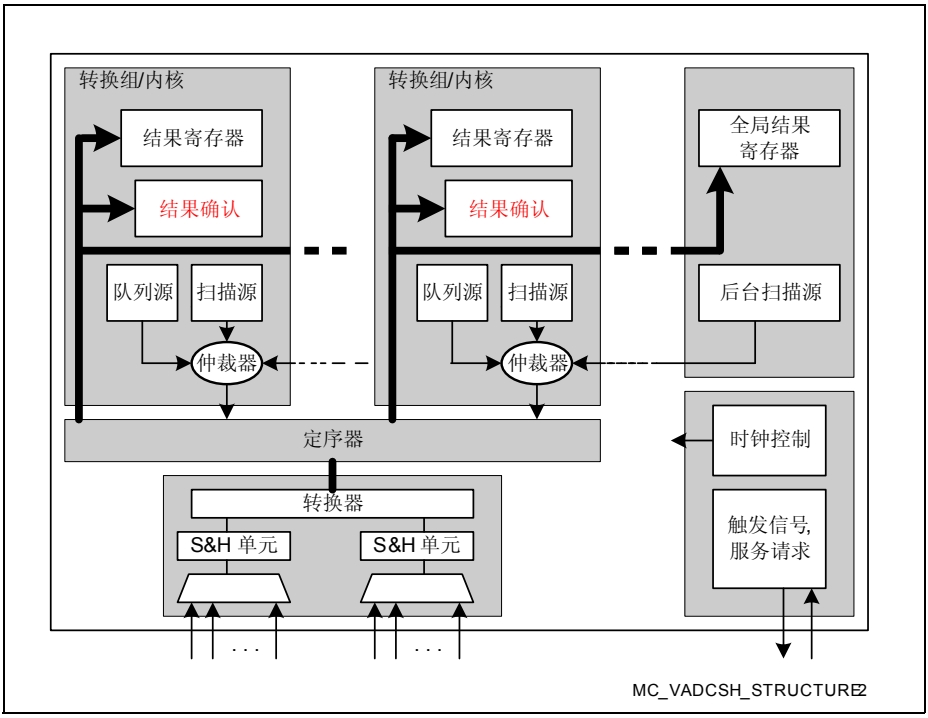


图 16-1　　ADC 结构概览

16.2 简介和基本结构

XMC1300 的多功能模 / 数转换器模块（VADC）包含一组转换器单元，每个转换器单元可以独立操作，也可以通过一个模拟后台转换器的共用请求源进行操作。每个转换器单元都配备了一个专用的输入多路复用器和专用的请求源，它们一起构成独立的转换器组。

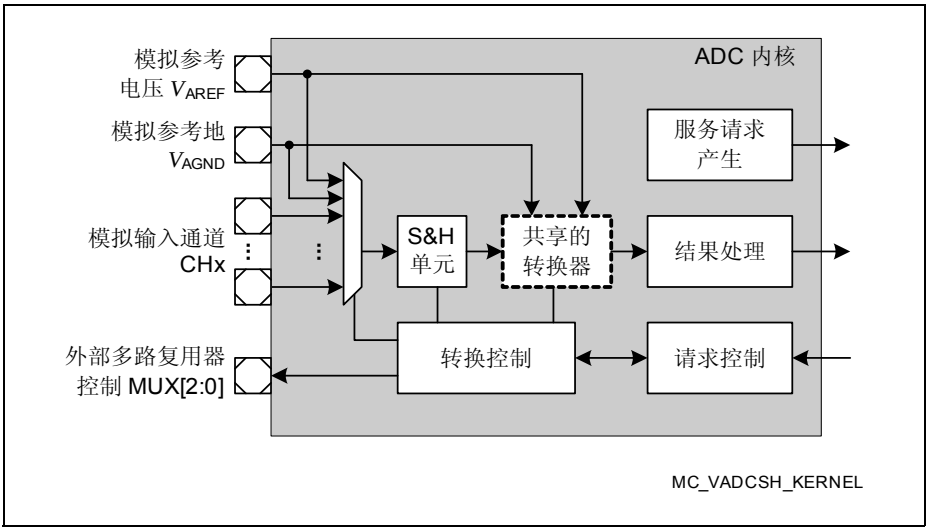


图 16-2 ADC 内核框图

该基本结构支持面向应用的编程和操作，同时仍然提供对所有资源的访问。所有的转换器组几乎完全相同，支持灵活的通道功能分配。

可根据给定应用的要求对一组功能单元进行配置。这些单元构成从输入信号到数字结果的通路。

每个内核集成一个专门的采样保持单元，该单元连接到输入多路复用器。多个内核构成一个集群并共享一个公共的高速转换器，因此各内核具有相同的性能。

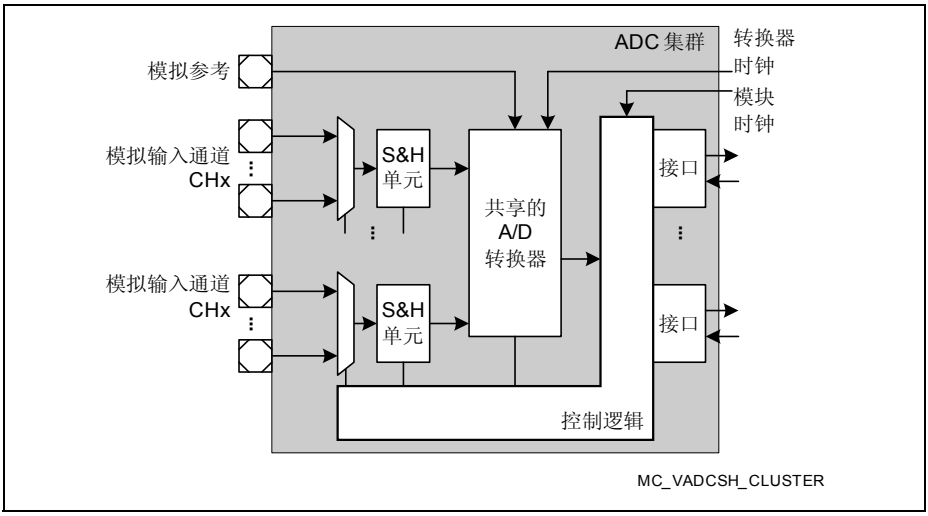


图 16-3 ADC 集群结构

基本模块时钟 f_{ADC} 连接到系统时钟信号 f_{MCLK}。

转换器时钟 f_{CONV} 连接到一个 32-MHz 的时钟信号。

转换模块和请求源

模 / 数转换可被多个请求源 （2 个组请求源和 1 个后台请求源）请求，并且能够在多种转换模式下执行。可同时使能多个请求源并为每个请求源配置相应的优先级。

- **固定通道转换 (单次或连续)**
  一个特定通道源请求对一个可选通道进行转换 （单次或重复）
- **自动扫描转换 （单次或连续）**
  一个通道扫描源 （请求源 1 或 2)）请求对所有需要转换的通道进行线性自动扫描转换 （单次或重复）
- **通道序列转换 （单次或连续）**
  一个队列源 （请求源 0）请求一个转换序列，该转换序列可包含最多 8 个任意可选的通道 （单次或重复）

多个可用请求源可同时使用多种转换模式，即可以同时使能不同工作模式的转换。每个请求源都可被单独使能，并且可由外部事件触发，例如 PWM 的边沿、定时器信号或引脚的跳变。

请求源控制

因为所有的请求源可同时被使能，因此需要有一个仲裁器解析来自不同请求源的并发转换请求。每个请求源都可由外部信号、片上信号或软件触发。

优先级高的请求能撤消当前正运行的低优先级转换 （撤消 – 插入 – 重启动模式），也可以等待正在运行的转换完成后立即进行转换 （等待启动模式）。如果目标结果寄存器尚未被读取，转换可以被推迟 （待读模式）。

有些通道还可与其他 ADC 内核同步，因此可以并行转换 2 个信号。

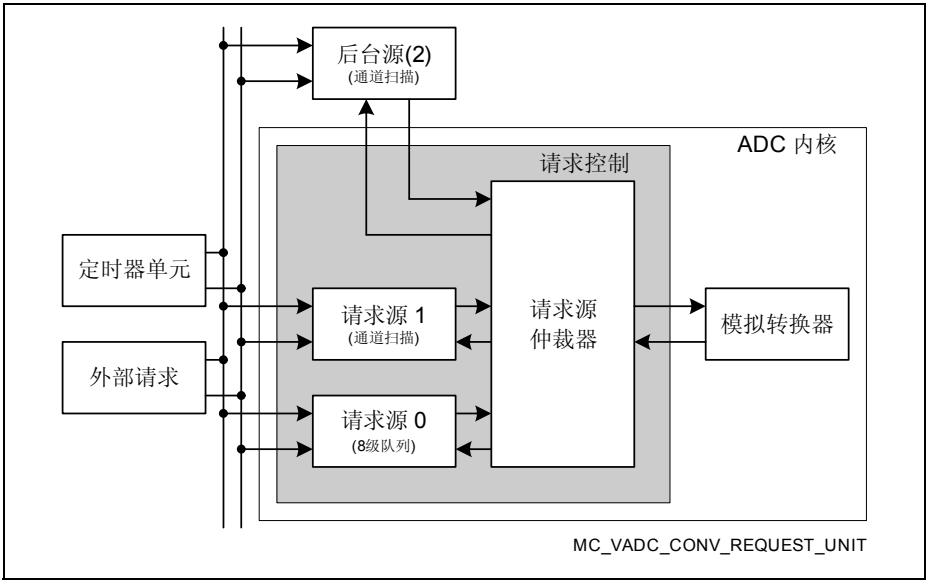


图 16-4　　转换请求单元

输入通道选择

模拟输入多路复用器选择一个将要转换的可用模拟输入 (CH0 - CHx[1])。三个转换源可选择一个线性序列、一个任意序列或一个特定通道。这些源的优先级是可配置的。

如果所需要的独立输入通道多于内建的通道总数，还可使用 VADC 集成的外部模拟多路复用器自动控制功能来增加转换通道数。

注：　由于引脚的限制，并非所有的模拟输入通道都是可用的。详见 **16.17 节** *的具体实现描述。*

1) 专用输入通道的可用性取决于所用产品型号的封装。 **16.17.2 节**给出了模拟连接一览表。

转换控制

转换参数，如采样阶段的持续时间、参考电压或转换结果分辨率，可通过 4 个输入类（2 个组专用类、 2 个全局类）配置。每个通道都可被单独分配给其中的一个输入类。

因此，可根据连接到 ADC 的传感器类型 （或其他模拟源）对输入通道进行调整。

该单元还可控制内建的多路复用器和外部模拟多路复用器。

模 / 数转换器

将所选输入通道转换为数字值的过程是：首先采样所选输入通道的电压，然后产生所选位数的结果。

一个集群内的两个采样保持单元共享一个高速转换器。该转换器顺序地生成所有采样保持单元的结果值。转换器自动执行校验周期。

对于断线检测（参见 **16.13.1 节**），可以在采样所选输入通道之前将转换器网络预充到一个设定的电压。

结果处理

每个模拟输入通道的转换结果都可被直接保存到 16 个组专用结果寄存器之一和一个全局结果寄存器。一个结果寄存器可被一组通道使用，也可以被单个通道使用。

待读模式可以避免数据丢失，因为通过阻塞转换保证了只有先前的结果被读取之后，转换结果才可以被覆盖。

数据缩减功能 （例如对于数字抗混迭滤波）可以自动累加最多 4 个转换结果，然后发出服务请求。

可以选择使能 FIR 或 IIR 滤波器，滤波器可以在转换结果被送到结果寄存器之前对其进行预处理。

另外，结果寄存器可级联起来构建 FIFO 结构，该结构可存储多个转换结果而不会覆盖先前的数据。这就为 CPU 从 ADC 读取转换数据增加了所允许的延迟时间。

中断服务请求的产生

有多个 ADC 事件可以向 CPU 发出中断服务请求：

- **源事件**指示相应请求源的一个转换序列的完成。该事件可用于触发新序列的准备。
- **通道事件**指示某一通道转换的完成。可与极限检测相结合，仅当结果在给定的范围内时才会产生中断。
- **结果事件**指示相应结果寄存器中一个新结果数据的可用性。如果数据缩减模式被激活，则结果事件仅在累加序列完成后才会产生。

每个事件都可以被分配到 8 个中断服务请求节点中的一个。这就允许根据应用需求进行中断服务请求的分组。

安全特性

VADC 具有确保信号通路完整性的机制，因而支持安全性要求高的应用。

断线检测（BWD）功能在采样输入通道之前预加载一个可选电平到转换器网络。如果输入信号不再连接到 VADC 输入，结果就会反映预加载值。如果使用了缓冲电容，需要一定数量的转换才能达到故障指示电平。

多路复用器诊断（MD）功能将一个弱上拉或下拉器件连接到一个输入通道。随后的转换就可确认预期的修改后的信号电平。这就允许检查多路复用器工作是否正常。
上、下拉器件由标准端口控制寄存器控制。

16.3 电气模型

每次将一个模拟输入电压转换到一个数字值都包含两个连续的阶段：

- 在采样阶段，输入电压被采样并保存。
  输入信号通路是该阶段的一个简化模型。
- 在转换阶段，所保存的电压被转换为一个数字结果。

输入信号路径

XMC1300 的 ADC 使用由 C_{AINSW}（呈现在每个输入引脚的小寄生电容）表示的开关电容器。在采样阶段，电容器 C_{AINSW} 通过输入多路复用器（建模为理想开关和串联电阻 R_{AIN}）与所选的模拟输入 CHx 相连。

连接 CHx 的开关在采样阶段闭合，将电容器与输入电压 V_{AINx} 相连。

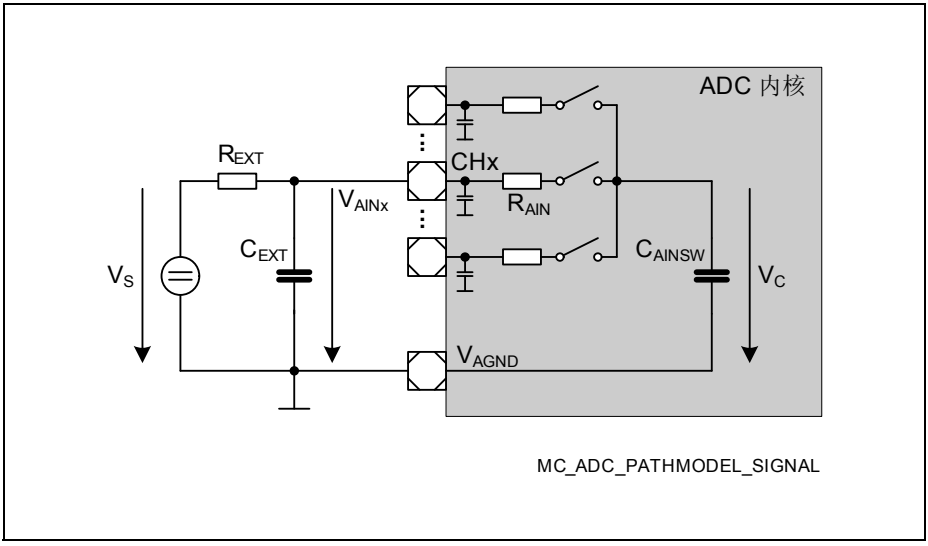


图 16-5 信号通路模型

图 16-5 给出了模拟输入信号通路的简化模型。内部阻抗为 R_{EXT} 的模拟电压源（值为 V_S）为待转换的模拟输入提供电压。
在采样阶段，相应的开关闭合，电容器 C_{AINSW} 被充电。由于 RC 组合电路的低通特性，实际被转换的电压值 V_C 不会立即跟随电压 V_S。模拟电压源的阻抗值 R_{EXT} 和所期望的转换精度决定了需要的采样阶段时间。
为了降低 R_{EXT} 的影响并滤除输入噪声，建议在 ADC 的模拟输入引脚上引入快速外部隔直电容 C_{EXT}。这样一来，在采样阶段主要由 C_{EXT} 给采样电容充电。这种结构比不带隔直电容的机构大大缩短了采样阶段的时间，这是因为决定采样时间的低通时间常数主要由 R_{AIN} 和 C_{AINSW} 的值给出。

由 R_{EXT} （通常是信号源的一个参数）和 C_{EXT} 构成的低通滤波器的参数选择应根据应用需要确定：

- 对于快速变化的动态信号，小的电容会允许 V_{AINx} 在同一模拟输入通道的两个采样阶段之间跟随 V_S 变化。
- 对于高精度转换，外部隔直电容 C_{EXT} 至少应为 $2^n \times C_{AINSW}$，使采样阶段期间 V_{AINx} 的电压变化小于 1 LSB_n。电压的变化是由 C_{EXT} 和 C_{AINSW} 之间的电荷重新分配导致的。

通过 ADC 的模拟输入结构的漏电流会在 R_{EXT} 上产生电压降，因而引入误差。ADC 输入的漏电流在温度高时会增大，而且如果输入电压电平接近模拟电源地 V_{SS} 或模拟电源电压 V_{DDPA}，漏电流也会增大。可通过避免输入电压接近电源电平来降低 ADC 通道的输入漏电流。
相邻模拟输入通道发生过载情况时 （输入电压超出电源电压范围）会注入额外的漏电流（由耦合因子定义）。
在采样阶段开始之初，电容 C_{AINSW} 被自动放电。

传输特性和误差定义

ADC 的传输特性描述了模拟输入电压到 2^n 个离散数字结果值（n 位分辨率）的联系。每个数字结果值 （范围在 0 到 2^n-1）代表由参考电压范围除以 2^n 所定义的输入电压范围。该范围 （称为量化步长或编码宽度）代表 ADC 的粒度 （称为 LSB_n）。数字结果的离散特性使得每个转换结果都会产生一个 ±0.5 LSB_n 的系统固有量化误差。
当模拟输入达到 0.5 LSB_n 时，理想的传输曲线产生第一次数字跳变 （0 和 1 之间）。量化步长在整个输入电压范围内是均匀分布的。
低于或高于参考电压极限值的模拟输入电压会导致数字结果饱和为 0 或 2^n-1。

实际的传输曲线与理想的传输曲线之间会有一定的偏差：

- **偏移误差**是在最低编码情况下实际传输曲线与理想传输曲线的偏差。这是指对所有可能编码的最佳拟合曲线。
- **增益误差**是理想传输曲线的斜率与实际传输曲线斜率的偏差。这是指对所有可能编码的最佳拟合曲线。
- **微分非线性误差** （DNL）是实际的编码宽度 （两个相邻数字转换结果之间的模拟输入电压的变化）与理想编码宽度的偏差。
- **积分非线性误差** （INL）是实际传输曲线与调整后的理想传输曲线之间的偏差（偏移和增益误差同实际曲线，但有相同的编码宽度）。
- **总不可调整误差**（TUE）描述在给定测量范围内实际转换结果与理想传输特性之间的最大偏差。因为上面提到的一些误差可以相互补偿，所以 TUE 的值一般远小于各个误差之和。
  TUE 还包含了生产工艺偏差和内部噪声效应。（如果开关噪声是由系统产生的，这一般会导致 TUE 增大）。

16.4 一般功能的配置

虽然可为每个通道、源或组单独选择许多参数，但有些调整对整个 ADC 集群都有效：

- 时钟控制
- 内核同步
- 外部多路复用器控制

16.4.1 一般时钟方案和控制

XMC1300 的 A/D 转换器配有一个来自系统的全局时钟信号 f_{ADC}。该时钟信号控制所有逻辑块的全部功能并决定总体时序。全局配置寄存器为集群内的所有转换器定义公共时钟基准。这就保证了应并行工作的转换器的行为是确定的。

转换器的时钟 f_{CONV} 决定了转换器本身的性能。

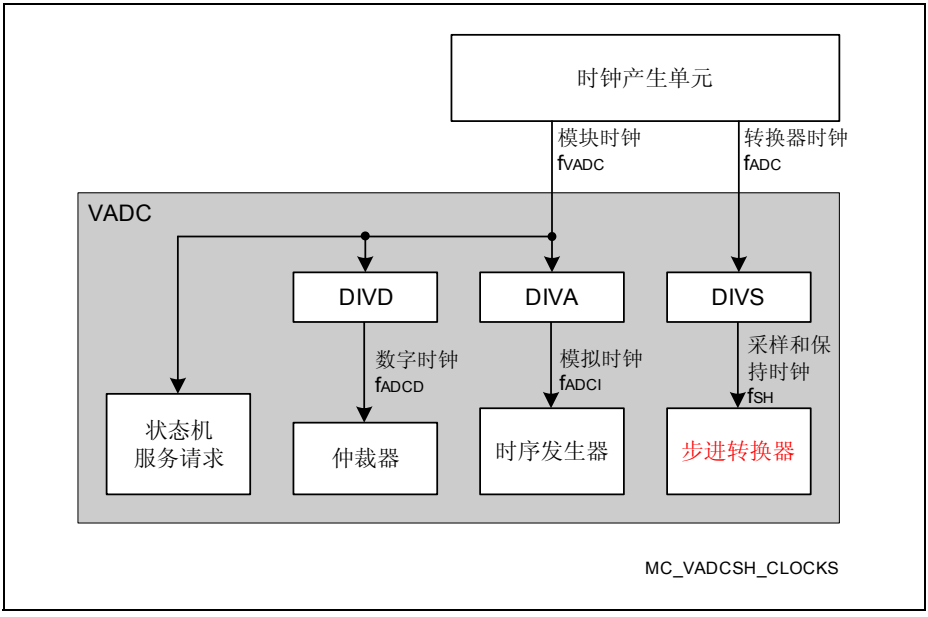


图 16-6 时钟信号概览

16.4.2 寄存器访问控制

VADC 提供了多种保护机制来防止对 VADC 控制位域的意外写访问。

- 一种专用的寄存器访问控制机制提供通用的保护方案，以防止对寄存器内容的意外修改。寄存器 **ACCPROT0** 和 **ACCPROT1** 允许限制对几个寄存器组的写访问。要保护

的寄存器可由用户选择。表 16-10 列出了属于每个寄存器组的寄存器。
寄存器 ACCPROT0/1 本身受 SCU 一章所描述的位保护机制的保护。

- 一个寄存器内的位域组同样可以受相关联的写控制位的保护。在对所关心的位域进行写访问时，也必须向该写控制位 （xxWC）写 1 。

16.4.3 优先通道和优先结果寄存器分配

一个组的每个通道和结果寄存器都可被分配给该组的请求源，然后作为优先通道或优先结果寄存器。

优先通道分配

已分配的优先通道只能被本组内的请求源转换。未分配的通道也可由后台请求源转换。

优先结果寄存器分配

已分配的结果寄存器只能通过本组内的请求源进行写操作。未分配的结果寄存器还可以由后台请求源进行写操作。

16.5 模拟模块的激活和控制

ADC 的模拟转换器是将所选输入电压转换为数字结果值的功能块。它在工作期间一直汲取电流，可以在两次转换之间将其停用以减少总体能耗。

注： 器件复位后模拟转换器是关断的，必须在触发任何与转换器相关的操作之前将其使能。

可通过校准模拟转换器来补偿工艺、温度和电压变化的影响，提高转换精度。

16.5.1 模拟转换器控制

工作模式由位域 **GxARBCFG (x = 0 - 1)**. ANONS 决定：

- ANONS = 11_B：**正常工作模式**
  转换器处于活动状态，可以立即启动转换。
  不需要唤醒时间。
- ANONS = 10_B **保留**
- ANONS = 01_B：**低速待机模式**
  当没有转换请求时，转换器进入省电模式。当有转换请求时，它会自动返回到正常工作模式。低速待机模式使 ADC 电源的总体功耗最低。
  需要唤醒时间（见下面的描述）。
- ANONS = 00_B：**转换器关断** （复位后的默认状态）
  如果相应集群的所有组都关断，则转换器被关断。而且，数字逻辑模块被设置为其初始状态。如果仲裁器正在运行，则它在完成当前的仲裁巡回后停止。
  在启动转换之前，要为 ANONS 选择活动模式。
  需要扩展的唤醒时间 （见下面的描述）：

寄存器 SHSCFG 中的位 ANOFF 强制模拟部分进入掉电模式，而不改变相关组的配置。当 ANOFF = 0 时，模拟部分被数字部分中的位域 ANONS 自动控制。因为几个组（具有独立的 ANONS 位域）由同一个集群提供服务，所以模拟部分由下述方案控制：

表 16-3 模拟部分掉电控制操作

ANON 设置[1)]	掉电模式
所有 ANON 信号都是 00_B	模拟部分关断
ANON 信号是 00_B 或 01_B	在非采样阶段且没有转换正在进行时，模拟部分自动关断
一个或多个 ANON 信号是 11_B	模拟部分总是活动的

1) 考虑在位 ANOFF = 0 期间。

注： 因为关断模拟部分需要一个唤醒阶段，所以最佳配置取决于实际应用。

从模拟掉电唤醒的时间

当转换器被激活时，它需要经过一定的唤醒时间后才能进行正确的转换。唤醒时间可通过在启动一次转换之前等待所需要的周期来建立，也可通过在采样时间上增加所需要的周期来建立。

唤醒时间大约为 15 µs。
准确的数值可在数据手册中找到。

注： 在首次使能转换器之后，也需要唤醒时间。

16.5.2 校准

校准操作可自动补偿由工艺、温度和电压变化所引起的偏差，确保在工作期间获得精准的转换结果。

偏移和增益校准需要执行几个不同的校准周期。为最小化校准所消耗的额外时间，偏移和增益校准周期交替进行。

一旦所有转换器都完成复位，就需要进行初始启动校准。所有的转换器都必须被使能 (ANONS = 11_B)。初始启动校准通过将寄存器 GLOBCFG 中的位 SUCAL 置 1 来全面启动。通过置位寄存器 CALCTR 中的位 SUCAL，可以启动相应集群转换器的启动校准。在初始校准序列结束后，可以启动转换。这可由位 CALS = 1_B 和位 CAL = 0_B 来指示。

启动校准阶段需要 1 920 个周期 (1920 × 31.25 ns = 60µs)。

此后，校准周期将补偿漂移参数的影响。校准周期可在一次转换序列之前或之后 （根据用户配置）进行，也可以禁止校准周期。

在 ADC 的空闲阶段，校准定时器自动触发校准周期。这样可以保证，即使经过一段空闲时间后， ADC 仍能保证其额定性能。

注： ADC 的误差取决于温度。因此校准必须周期性地重复进行。

最大校准间隔

在没有转换请求期间，步进式转换器保持在空闲状态。为了不损失精度，有一个计数器记录自最后一次执行校准以来的时间。在每次校准之后，该定时器都会被加载，加载的初始值来自位域 CALCTR.CALMAX，然后计数 512 × (CALMAX+1) 个时钟。如果计数器期满，它会产生一次校准请求来启动步进转换器执行一个校准周期。

16.5.3 Σ-Δ 回路功能

由于步数，即离散结果值是有限的，每次标准的模 / 数转换都会产生一个量化误差。通过激活 **Σ-Δ** 回路功能，残余的量化误差可以被传递到下一次转换。残余电荷被保存并添加到下一次采样的电荷。因此，对一系列转换结果求平均值 （在使能回路功能的情况下）可以减小引入的量化误差。

注： VADCDIG 的数据累加功能已经支持这种平均。

回路控制位域在寄存器 **SHS0_LOOP** 中是成对可用的。

16.6 转换请求的产生

组内的转换请求单元自动处理转换请求的产生。三个请求源 （2 个组专用请求源和 1 个后台请求源）可以产生对一个模拟通道的转换请求。仲裁器解析并发请求并选择下一个要转换的通道。

在发生触发事件时，请求源请求对某个特定模拟输入通道或一个通道序列进行转换。

- **软件触发**
  直接激活相应的请求源。
- **外部触发**
  用外部事件同步激活请求源，例如用定时器产生的 PWM 信号或来自端口引脚的信号作为触发脉冲。

应用软件可以选择触发信号类型和触发源、要转换的通道以及请求源的优先级。请求源也可以直接由软件激活，而无需外部触发信号。

仲裁器周期性地扫描请求源，看是否有未裁决的转换请求，并选择具有最高优先级的请求源。然后该转换请求被转发到转换器，启动所请求通道的采样和转换。

每个请求源都可以在单次或连续模式下工作：

- **在单次模式**
  一次触发仅请求一次所定义的转换 （序列）。后续的转换 （序列）必须再次被触发。
- **在连续模式**
  一旦被触发，就自动重复请求所定义的转换 （序列）。

对于每个请求源，外部触发信号产生于 16 个可选触发输入 (REQTRx[P:A]) 之一以及 16 个可选门控输入 (REQGTx[P:A]) 之一。 XMC1300 的可用触发信号在 **16.17.3 节**列出。

*注： **页 16-6 的图 16-4 “ 转换请求单元 ”** 概述了请求源。*

有两种可用的请求源类型：

- **队列请求源**能为一个任意序列的输入通道发出转换请求。该序列的通道编号可自由编程[1)]。这就允许支持不能被通道扫描源覆盖的特定应用转换序列，而且还支持在一个序列内对同一通道进行多次转换。
  一个队列请求源连续或周期性地转换一系列的输入通道。例如，如果被编程为中等优先级，一些输入通道就可在一个指定的事件发生时进行转换（例如与一个 PWM 信号同步）。同时，较低优先级的转换被挂起。
  请求源 0 是一个组专用的 8 级队列请求源。
- **通道扫描源**能为一个连贯序列的输入通道发送转换请求。该序列从被使能的最高通道号开始向较低的通道号进行。所有的可用通道 [1)] 都可以被使能为扫描序列中的一员。每个序列中的每个通道都被转换一次。
  一个扫描源连续或周期性地转换一系列的输入通道。例如，如果被编程为低优先级，一些输入通道就可在一后台任务里被扫描，来更新非时间关键性信息。

1) 特定输入通道的可用性取决于所选产品类型的部件。总结请看 **16.17.2 节** .
后台源仅能请求无优先级的通道，即那些未被寄存器 GxCHASS 选中的通道。优先级通道为特定组请求源 0 和 1 保留。

– 请求源 1 是一个组专用的通道扫描源。
– 请求源 2 是一个全局通道扫描源 （后台请求源）。
后台请求源可请求对所有组的所有通道进行转换。

16.6.1 队列请求源处理

队列请求源支持同一组中任意多个（最多 8 个）通道的短转换序列 （与具有固定转换顺序的扫描请求源相反）。编程的序列被保存在队列缓冲区中（基于 FIFO 机制）。请求的通道号通过队列输入寄存器 **GxQINR0 (x = 0 - 1)** 写入，队列级 0 定义下一个要转换的通道。

只有在队列级 0 中保存了一个有效条目的情况下，转换请求才会被发送到请求源仲裁器。

如果由于存在高优先级的请求，仲裁器中止了由队列请求源触发的一次转换，则相应的转换参数被自动保存在备份级。这样可确保被中止的转换请求不丢失，而是加入到下一轮的仲裁 （在级 0 之前）。

触发和门控单元从所选择的外部 （ADC 的外部）触发和门控信号产生触发事件。例如，一个定时器单元能够发出一个请求信号，使转换与 PWM 事件同步。

触发事件启动一个队列序列，触发事件可通过软件或所选择的硬件触发信号产生。触发事件的发生由位 QSRx.EV 指示。当相应的转换被启动时，该标志被清零，也可以通过写位 QMRx.CEV 将其清零。

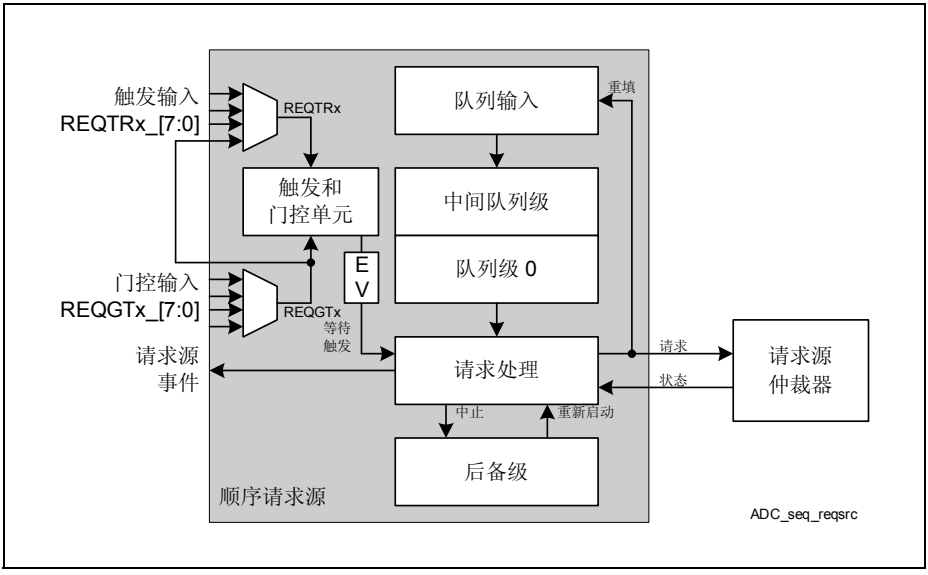


图 16-7 队列请求源

一个序列是通过将转换请求添加到队列输入寄存器 (**GxQINR0 (x = 0 - 1)**) 来定义的。每个条目选择要转换的通道，并且可以使能外部触发信号、中断产生和自动重填 （即转换结束后将该条目复制到队列的最顶端）。条目被保存到队列的缓冲级。

级 0(**GxQ0R0 (x = 0 - 1)**) 的内容选择下一个要转换的通道。当所请求的转换被启动时，该队列级的内容失效，并且被复制到备份级。然后下一个队列条目可以被处理 （如果可用的话）。

注： 队列级的内容不能直接被修改，只能通过写队列输入或清空队列来修改。
*队列的当前状态在寄存器 **GxQSR0 (x = 0 - 1)** 中显示。*
如果所有的队列条目都选择了自动重填，则所定义的转换序列可以重复执行而不需重新编程。

队列请求源的属性

队列请求源 0 提供了 8 个缓冲级，可处理具有多达 8 个输入通道条目的序列。它支持短的特定应用转换序列，尤其那些时间关键型序列和对同一通道进行多次转换的序列。

队列源操作

配置队列请求源，通过执行下面操作完成：

- 通过将条目写入队列输入 **GxQINR0 (x = 0 - 1)** 来定义序列。在使能请求源之前，要先初始化整个序列，因为重填功能被使能，所以不允许软件对 QINRx 进行写操作。
- 如果希望使用硬件触发或门控功能，可通过对 **GxQCTRL0 (x = 0 - 1)** 编程来选择合适的触发和门控输入以及正确的跳变。
  通过对寄存器 **GxQMR0 (x = 0 - 1)** 中的位域 ENGT 编程来使能触发信号并选择门控方式。[1)]
- 使能相应的仲裁时隙（0）以接受队列源的转换请求(见寄存器**GxARBPR (x = 0 - 1)**)。

启动队列序列，通过产生一个触发事件实现：

- 如果已经选择并使能了某一硬件触发信号，在所选输入信号（例如来自一个定时器或一个输入引脚）所配置的跳变沿产生触发事件。
- 通过设置 GxQMR0.TREV = 1 产生一个软件触发事件。
- 写一个新条目到一个空队列的队列输入。这将导致产生一个 （新）有效队列条目，该条目被转发到队列级 0，并启动一次转换请求 （如果已通过 GxQMR0.ENGT 使能并且不需等待外部触发信号）。

注： 如果重填机制被激活，一个被处理过的条目会被自动重新加载到队列中。这样就会一直重复相应的序列 （自动扫描）。
在这种情况下，当队列源正在运行时，不要写队列输入。
对一个完全填充的队列的写操作被忽略。

1) 如果使用来自 ERU 的 PDOUT 信号，要在使能门控输入之前相应地初始化 ERU，以避免不希望的信号跳变。

停止或中止一个正在执行的队列序列，通过执行下列操作之一来完成：

- 如果外部门控被使能，将门控信号切换到无效电平。该操作不会修改队列条目，只是防止发送转换请求给仲裁器。
- 禁止仲裁器中相应的仲裁时隙 （0）。该操作不会修改队列条目，只是防止仲裁器接受来自请求处理块的请求。
- 通过设置位域 ENGT = 00_B 来禁止队列源：
  - 通过设置位 GxQMR0.CLRV = 1，使下一个待处理的队列条目失效。
    如果备份级包含一个有效条目，则该条目失效，否则级 0 失效。
  - 通过设置位 GxQMR0.FLUSH = 1，清空队列的所有条目。

队列请求源事件和服务请求

当一次转换完成时，会发生一个队列源的请求源事件。根据**图 16-8** 所示的结构，一个源事件服务请求可基于一个请求源事件产生。如果检测到一个请求源事件，则寄存器 **GxSEFLAG (x = 0 - 1)** 中的相应指示标志会置 1。也可以通过向相应的位写 1 来置位这些标志，写 0 时不起作用。指示标志可通过软件向寄存器 **GxSEFCLR (x = 0 - 1)** 中的相应位写 1 来清除。
对于一次正常的序列转换，中断使能位取自级 0 ；对于一次中止后的重复转换，中断使能位取自备份级。

每当检测到相关的请求源事件（并且通过 GxQ0R0.ENSI 或 GxQBUR0.ENS 分别使能）或寄存器 **GxSEFLAG (x = 0 - 1)** 中的相关位被写 1（该写操作模拟一次请求源事件）时，由寄存器 **GxSEVNP (x = 0 - 1)** 中的请求源事件中断节点指针位域所选择的服务请求输出线 SRx 被激活。

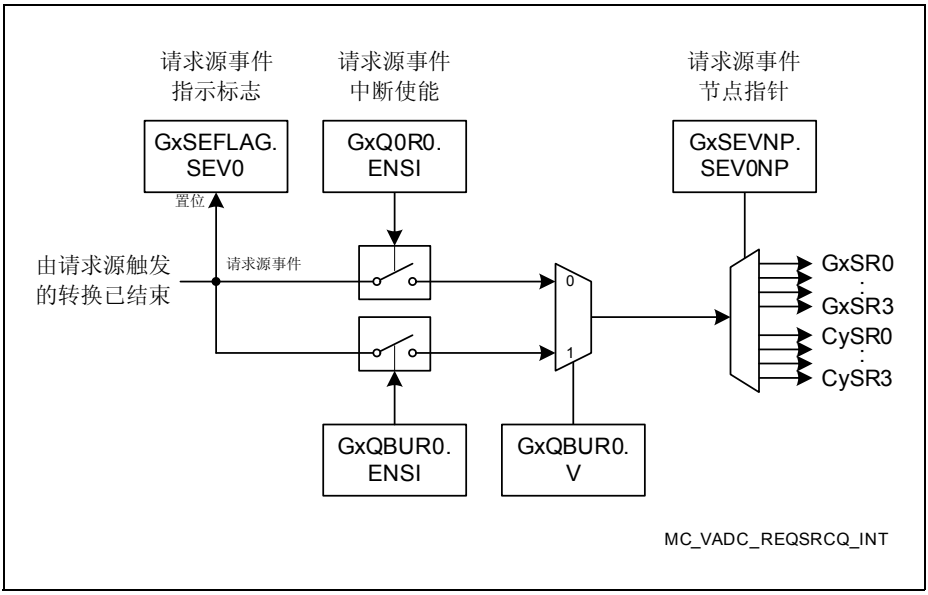


图 16-8　　队列请求源的中断产生

16.6.2　通道扫描请求源处理

VADC 提供两种类型的通道扫描源：

- 源 1：组扫描源
  该扫描源可请求相应组内的所有通道。
- 源 2：后台扫描源
  该扫描源可请求所有组的所有通道。
  寄存器 **GxCHASS (x = 0 - 1)** 中选择的优先通道不能加入后台转换序列。

两类源以相同的方式工作，并且提供相同的寄存器接口。后台源提供更多的请求 / 挂起位，因为它能请求所有组的所有通道。

通过置位或清零寄存器 **GxASSEL (x = 0 - 1)** 或 **BRSSELx (x = 0 - 1)** 中相应的通道选择位，可以将每个模拟输入通道都包含进扫描序列或从扫描序列排除。对于一个正在进行中的扫描序列，编程的寄存器值保持不变。扫描操作从被使能的最高通道号开始向低通道号进行。

在发生一次加载事件时，请求信息被传送到寄存器 **GxASPND (x = 0 - 1)** 或 **BRSPNDx (x = 0 - 1)** 中的挂起位。挂起的转换请求指示在进行中的扫描序列内哪个输入通道将被转换。每次由扫描请求源触发的转换启动都会自动清除相应的挂起位。如果由扫描源触发的最后一次转换结束并且所有的挂起位都被清零，则当前的扫描序列被认为已经结束并产生一个请求源事件。

只有在至少有一个挂起位被置 1 的情况下，才会向请求源仲裁器发出转换请求。

如果仲裁器因为存在高优先级的请求而中止了由扫描请求源触发的一次转换，则相应的挂起位被自动置 1。这就确保了一次中止的转换不会丢失而是加入下一轮仲裁。

触发和门控单元从选定的外部触发（ADC 的外部）和门控信号产生加载事件。例如，一个定时器单元可发出一个请求信号，使转换与 PWM 事件同步。

加载事件启动一个扫描序列，该事件可通过软件或通过选择的硬件触发信号产生。请求源事件也能产生一个自动加载事件，因此编程的序列能自动重复。

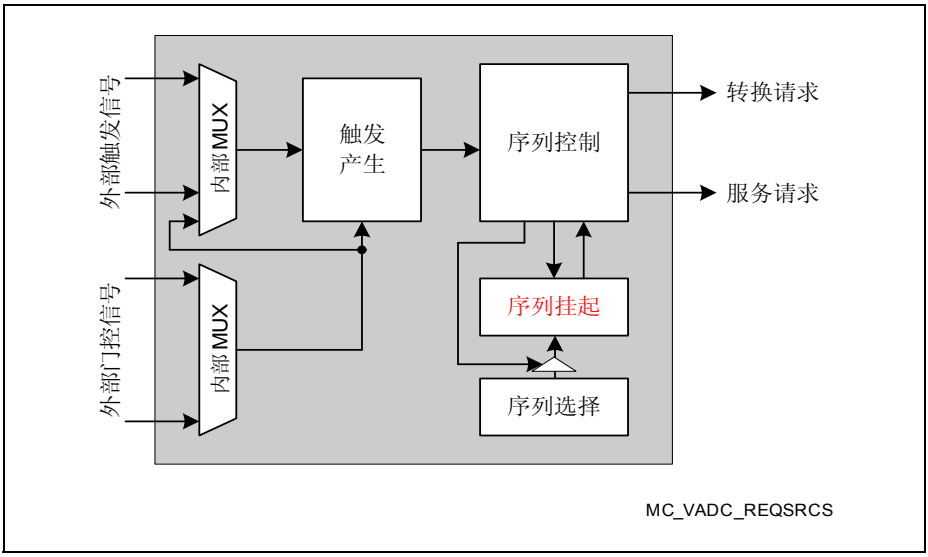


图 16-9　　扫描请求源

扫描源操作

配置扫描请求源，通过执行下面的操作完成：

- 通过对 **GxASSEL (x = 0 - 1)** 或 **BRSSELx (x = 0 - 1)** 编程来选择序列的输入通道
- 若希望使用硬件触发或门控功能，可通过对 **GxASCTRL (x = 0 - 1)** 或 **BRSCTRL** 编程选择合适的触发和门控输入以及正确的信号跳变。通过对 **GxASMR (x = 0 - 1)** 或 **BRSMR**[1] 编程来使能触发信号并选择门控方式。
- 通过对 **GxASMR (x = 0 - 1)** 或 **BRSMR** 编程来定义加载事件操作 （处理挂起位、自动扫描模式）。
  在位 LDM = 0 的情况下，加载事件复制 **GxASSEL (x = 0 - 1)** 或 **BRSSELx (x = 0 - 1)** 的内容到 **GxASPND (x = 0 - 1)** 或 **BRSPNDx (x = 0 - 1)** （覆盖模式）。这将启动一次新的扫描序列并且中止来自前一扫描序列的任何挂起转换。

1) 如果使用来自 ERU 的 PDOUT 信号，在使能门控输入之前要相应地初始化 ERU，以避免不希望的信号跳变。.

在位 LDM = 1 的情况下，加载事件将 **GxASSEL (x = 0 - 1)** 或 **BRSSELx (x = 0 - 1)** 分别与 **GxASPND (x = 0 - 1)** 或 **BRSPNDx (x = 0 - 1)** 的内容进行或组合运算（组合模式）。这将启动一个扫描序列，该序列包含来自前一扫描序列的挂起转换。

- 使能相应的仲裁时隙（1），以接受来自通道扫描源的转换请求（见寄存器 **GxARBPR (x = 0 - 1)**）。

启动一个通道扫描序列，通过产生加载事件实现：

- 如果已经选择并使能了一个硬件触发信号，在所选输入信号（例如，来自一个定时器或一个输入引脚）所配置的跳变沿产生触发事件。
- 通过设置 LDEV = 1(**GxASMR (x = 0 - 1)** 或 **BRSMR**）产生一个软件加载事件。
- 通过直接写扫描信息到 **GxASPND (x = 0 - 1)** 或 **BRSPNDx (x = 0 - 1)** 中的挂起位，来产生一个加载事件。该信息被复制到 **GxASSEL (x = 0 - 1)** 或 **BRSSELx (x = 0 - 1)** 并且自动产生一个加载事件。
  在这种情况下，通过一次数据写操作即可定义并启动一个扫描序列，例如在 PEC 控制下 （假定模式能装进一个寄存器）。

注： 如果自动扫描被使能，每当扫描序列结束并发生请求源事件时，会自动产生一个加载事件。这样会一直重复转换所定义的扫描序列 （自动扫描）。

停止或中止一个正在进行的扫描序列，通过执行下面的操作之一实现：

- 如果外部门控被使能，则将门控信号切换到所定义的无效电平。该操作不会修改转换挂起位，只是防止发送转换请求给仲裁器。
- 禁止仲裁器中的相应仲裁时隙 （1 或 2）。该操作不会修改转换挂起位的内容，只是防止仲裁器接受来自请求处理块的请求。
- 通过设置位域 ENGT = 00_B 来禁止通道扫描源。通过设置位 CLRPND = 1（**GxASMR (x = 0 - 1)** 或 **BRSMR**）来清除挂起请求位。

扫描请求源事件和中断服务请求

如果一个扫描序列的最后一次转换结束（所有挂起位均为 0），会发生一个扫描源的请求源事件。基于请求源事件会产生请求源事件中断。如果检测到一个请求源事件，则寄存器 **GxSEFLAG (x = 0 - 1)** 中相应的指示标志被置 1。也可通过向相应的位写 1 来置位这些标志，写 0 时不起作用。

每当检测到相关的请求源事件（已通过 ENSI 使能）或寄存器 **GxSEFLAG (x = 0 - 1)** 中的相关位被写 1（该写操作模拟一个请求源事件）时，通过寄存器 **GxSEFLAG (x = 0 - 1)** 中的请求源事件中断节点指针位域选择的服务请求输出 SRx 被激活。

指示标志位可通过软件写 1 到寄存器 **GxSEFCLR (x = 0 - 1)** 中的相应的位清零。[1)]

1) 请参见 **16-54 页 “ 服务请求产生 ”**。

16.7 请求源仲裁

请求源仲裁器周期性地为挂起的转换请求逐个查询相应组的请求源。每个请求源都在一轮仲裁周期内被分配一个时隙，称为仲裁时隙。一个仲裁时隙的时间可以由用户通过寄存器 **GLOBCFG** 来配置。

每个请求源的优先级可由用户通过寄存器 **GxARBPR (x = 0 - 1)** 配置，因此在有来自多个源的并发请求的情况下，仲裁器可根据应用需求选择下一个要转换的通道。

未使用的仲裁时隙被认为是空的，并且不参与仲裁。复位后，所有的时隙都被禁止，必须被使能（寄存器 **GxARBPR (x = 0 - 1)**）后才能参与仲裁过程。

图 16-10 总结了仲裁序列。一个仲裁周期由每个可用请求源的一个仲裁时隙组成。同步源总是在最后一个时隙被评估，并且拥有比所有其他源更高的优先级。在每个仲裁周期结束时，仲裁器已经确定了优先级最高的转换请求。

如果在仲裁周期内启动了一次转换，则该仲裁周期不会产生仲裁胜出者。在 XMC1300 中，可用的请求源如下：

- 仲裁时隙 0：**组队列源**，任意顺序的 8 级序列
- 仲裁时隙 1：**组扫描源**，组内设定顺序的序列
- 仲裁时隙 2：**后台扫描源**，设定顺序的序列，所有组
- 最后一个仲裁时隙：**同步源**，来自另一组的同步转换请求（总是在一个同步从组内按最高优先级处理）。

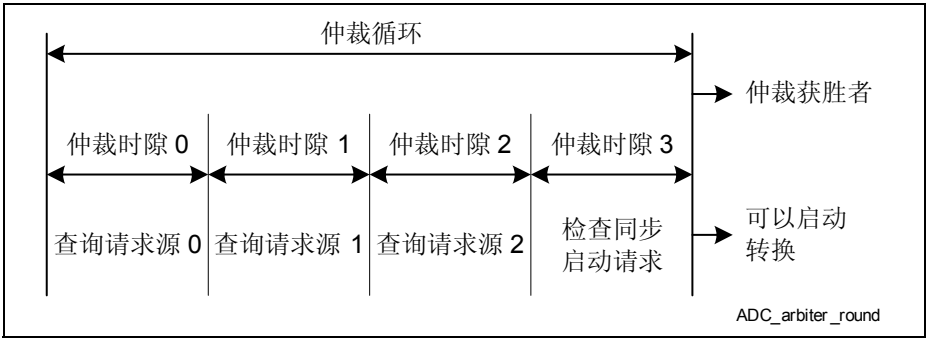


图 16-10 有 4 个仲裁时隙的仲裁周期

16.7.1 仲裁器操作和配置

仲裁器的时序 （即一个仲裁周期的时序）由一个仲裁周期内的仲裁时隙数和一个仲裁时隙的持续时间决定。

一个仲裁周期包含 4 到 20 个仲裁时隙 （由位域 **GxARBCFG (x = 0 - 1)** 定义）。对于 XMC1300，4 个时隙已经足够；对于不同的产品，可编程更多的时隙以获得相同的仲裁时序。

仲裁时隙的持续时间是可配置的，即 t_{Slot} = (DIVD+1) / f_{ADC}。
因此，一个仲裁周期的持续时间是 t_{ARB} = 4 × t_{Slot}。

仲裁周期引入一个时间粒度来检测输入的转换请求信号和启动相关转换的最早时间点。该时间粒度可引入最大为一个仲裁周期的抖动。可通过最小化仲裁周期来减小该抖动。

为了在一个转换请求的触发事件 （例如由一个定时器单元或一个外部事件触发）与启动相关转换之间获得可再生的反应时间 （无抖动的固定延迟），主要有以下两种选择。对于这两种选项，转换器都必须是空闲的，并且在产生触发事件之前至少一个仲裁周期内不允许有其它转换请求处于挂起状态：

- 如果位 **GxARBCFG (x = 0 - 1)**.ARBM = 0，**仲裁器会一直运行**。在该模式下，允许多个 ADC 内核进行同步转换。[1)]
  一次转换请求的触发信号的产生必须与仲裁时序同步。输入的触发信号应精确地为仲裁时间粒度的 n 倍 （n = 1，2, 3,...）。为了实现一定的灵活性，仲裁时隙的持续时间可编程为 f_{ADC} 的周期个数。
- 如果位 **GxARBCFG (x = 0 - 1)**.ARBM = 1，当不再有任何挂起的转换请求时，**仲裁器在一轮仲裁之后停止工作**。如果至少有一个已使能的请求源指示有一个挂起的转换请求，则仲裁器重新启动。转发请求的触发信号不需要与仲裁时序同步。
  在该模式下，不能够为同步从组进行并行转换。

每个请求源都有一个可配置的优先级，因此仲裁器能解析来自不同请求源的并发转换请求。优先级最高的请求被选中进行转换。可以调整这些优先级来适应一个给定应用的需求 （见寄存器 **GxARBPR (x = 0 - 1)**）。

转换启动模式决定对已经赢得了仲裁的转换请求的处理。

16.7.2 转换启动模式

当仲裁器选定了下一个要转换的请求时，对该通道的处理取决于转换器的当前活动状态：

- 转换器当前为空闲：立即启动仲裁胜出方的转换。
- 当前的转换具有相同或更高的优先级：当前的转换完成后，启动仲裁胜出方请求的转换。
- 当前的转换具有较低的优先级：转换器的操作可由用户配置：
  – **等待启动模式**：待当前的转换完成后，启动仲裁胜出方请求的转换。该模式提供最大的吞吐率，但可能对高优先级转换产生抖动。
  图 16-11 中的例子：

1) 更详细的信息请参见 **16-47 页 “ 转换同步 ”** 。

转换 A 被请求 (t1) 并启动 (t2)。转换 B 随后被请求 (t3)。但仅在转换 A 完成后才启动 (t4)。

– **撤消 - 插入 - 重复模式**：当前的转换被中止，中止后 (3 个 f_{ADC} 周期)，启动仲裁胜出方请求的转换。
被中止的转换请求重新被保存到相应的请求源中，并参加下一轮仲裁。该模式为高优先级的转换提供最小的抖动，但降低总体吞吐率。
图 16-11 中的例子：
转换 A 被请求 （t6）并启动 （t7）。转换 B 随后被请求 （t8）并启动 （t9），此时当转换 A 被中止但又再次请求。当转换 B 完成后 （t10），转换 A 被重新启动。
例外：如果两个请求的目标是同一个结果寄存器且待读模式有效（见 **16.11.3 节**），则当前的转换不能被中止。

注： 在上述每种情况下，一个被撤消的转换都可以被自动重复，或者如果被撤消的话它可以被丢弃。可以通过相应源的模式寄存器中的位 RPTDIS 来为每个源做出选择。

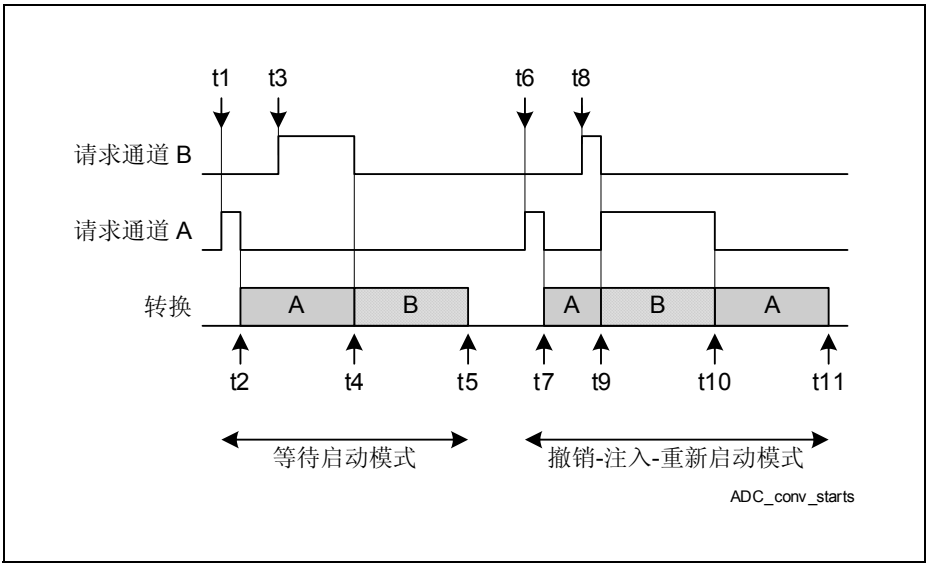


图 16-11　　转换启动模式

每个请求源的转换启动模式都可以通过寄存器 **GxARBPR (x = 0 - 1)** 中的位单独编程，并且适用于由该源请求的所有通道。在该例中，通道 A 的请求是由一个优先级比请求通道 B 转换的请求源要低的请求源发出的。

16.8 模拟输入通道配置

可以为每个模拟输入通道配置一些控制该通道转换的参数。通道控制寄存器定义了下面的参数：

- **通道参数**：通道的采样时间和结果的数据宽度通过输入类来定义。每个通道可在其所在组的两个类中选择一个，也可以在两个全局类中选择一个。
- **参考选择**：可选择一个备用参考地。
- **结果目标**：转换结果值可以保存到组专用寄存器或全局结果寄存器。组专用结果寄存器被选择为通道专用，由寄存器 **G0CHCTRy (y = 0 - 7)** 中的位域 RESREG 选择。
- **结果位置**：结果值能够以左对齐或右对齐的方式保存。确切的位置还取决于所配置的结果宽度和数据累加模式。
  参见**页 16-38 的图 16-18 “ 结果存储选项 ”**。
- **与标准转换比较 (极限检查)**：当一个新的结果值可用时，就会产生通道事件。通道事件产生的条件可被限制为结果值位于用户所定义范围的内部或外部。
  在快速比较模式，通道事件的产生取决于 （1 位）结果的跳变。
- **断线检测**：该安全特性可检测连接到一个模拟信号源 （传感器）的连接是否断开。
- **转换同步**：多个转换器在同一时刻进行同步转换。

别名功能将在通道 CH0 和 / 或 CH1 定义的转换请求重定向到其他模拟输入通道。

16.8.1 通道参数

每个模拟输入通道都由其相关联的通道控制寄存器配置。

注： 对于安全特性 “ 断线检测 ”，参见 **16.13.1 节**。

可为每个通道定义下面的特性：

- 转换类定义结果宽度和采样时间
- 通道事件的产生和结果范围的定义 （如果使用该功能）
- 结果寄存器选择及结果寄存器中转换结果存储位置的定义

组专用输入类寄存器为相关组的每个通道定义采样时间和数据转换模式，该组通过其通道控制寄存器 GxCHCTRy 中的位域 ICLSEL 来选择输入类。

全局输入类寄存器为任意组的每个通道定义采样时间和数据转换模式，该组通过其通道控制寄存器 GxCHCTRy 中的位域 ICLSEL 来选择输入类。

16.8.2 别名功能

别名功能将在通道 CH0 和 / 或 CH1 定义的转换请求重定向到其他模拟输入通道（引脚）。该功能可用于允许多个独立事件触发同一输入引脚的转换，并将转换结果保存到不同的结果寄存器中。

- 可对同一信号进行两次测量而无需读取转换结果，以避免数据丢失。这就允许快速地先后触发两次转换，而且与 CPU 的服务请求延迟无关。
- 传感器信号仅连接到一个模拟输入 （而不是两个模拟输入）。这就允许在低成本应用中节省输入引脚，在计算误差时只需考虑一个输入的漏电流。

- 即使模拟输入 CH0 被用作备选参考电压输入 （见图 16-12），仍可使用通道 CH0 的内部触发和数据处理功能。
- 两次转换的通道设置可以不同 （边界值、服务请求等）。

在典型的低成本交流驱动应用中，只使用一个公共的电流传感器来测量相电流。根据所施加的 PWM 序列，测量值具有不同的含义，采样点必须精确地位于 PWM 周期内。图 16-12 给出了这样一个例子：传感器信号与一个输入通道 (CHx) 连接，但会触发两路通道（CHx 和 CH0）的两次转换。利用别名功能， CH0 的一次转换请求导致一次模拟输入 CHx 的转换而不是 CH0，但使用的是 CH0 的设置。尽管测量的是同一路模拟输入（CHx），但转换结果可保存在不同的寄存器 RESx（对 CHx 触发的转换）和 RESy（对 CH0 触发的转换）中，并可从这些寄存器中读取。此外，可选择并可使能或禁止不同的中断或极限边界值。

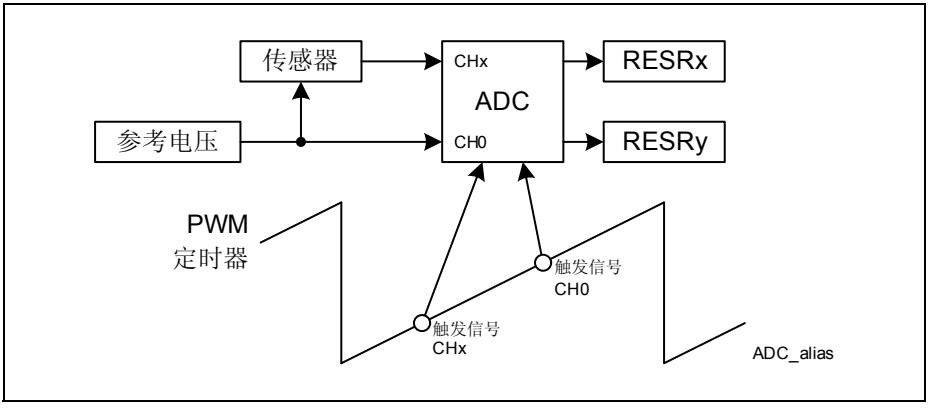


图 16-12　　别名功能

16.8.3　转换模式

ADC 转换能够以多种模式执行。转换模式的选择依据是所要求的数字结果分辨率和可接受的转换时间 (**16.10 节**)。
使用寄存器 **GxICLASS0 (x = 0 - 1)** 中的位域 CMS/CME 来选择转换模式。

标准转换

标准转换返回具有预定义分辨率的结果值。可选择 8 位、 10 位和 12 位的分辨率。
可以对这些结果值进行累加、滤波，也可以用这些结果值进行数字极限检查和确定极值。
注：　转换器可以带校准步骤或不带校准步骤工作。

快速比较模式

在快速比较模式，所选择的输入电压与保存在相应结果寄存器中的一个数字值直接比较。该比较操作返回一个二元结果，用于指示比较的输入电压高于还是低于给定的参考值。该结果的产生速度快，因此支持边界值监视。

快速比较模式使用一个 10 位的比较值，该值以左对齐方式保存在位 11 的位置。可使用独立的正向和负向增量值来定义一个任意的滞回区间。

选择比较值

数字或模拟比较操作的值可以从多个源选择。单独的 GxBOUND 寄存器提供软件定义的比较值。

在快速比较模式，结果寄存器提供比较值，而本地寄存器 GxBOUND 中的位域定义正向和负向增量值。

16.8.4 与标准转换比较 (极限检查)

极限检查机制能自动地将每次数字转换结果与一个上边界值和一个下边界值进行比较。当一次转换 / 比较的结果位于用户定义范围的内部或外部时 （见位域 CHEVMODE 和图 16-13），会产生一次通道事件。

该特性支持自动范围监视，并且通过只在某些预定义条件下发出服务请求来最小化 CPU 的负荷。

注： 对于每一个结果值（忽略结果范围），还可以产生通道事件，也可以完全禁止产生通道事件。

可从多个源选择用来与结果进行比较的边界值 （见寄存器 GxCHCTRy）。

位域 BNDSELU 和 BNDSELL 选择有效的上 / 下边界值，这些值来自组专用的边界寄存器 **GxBOUND (x = 0 - 1)** 或全局边界寄存器 **GLOBBOUND**。可以为相应组的每个通道选择组边界寄存器；每个可用通道都可选择全局边界寄存器。

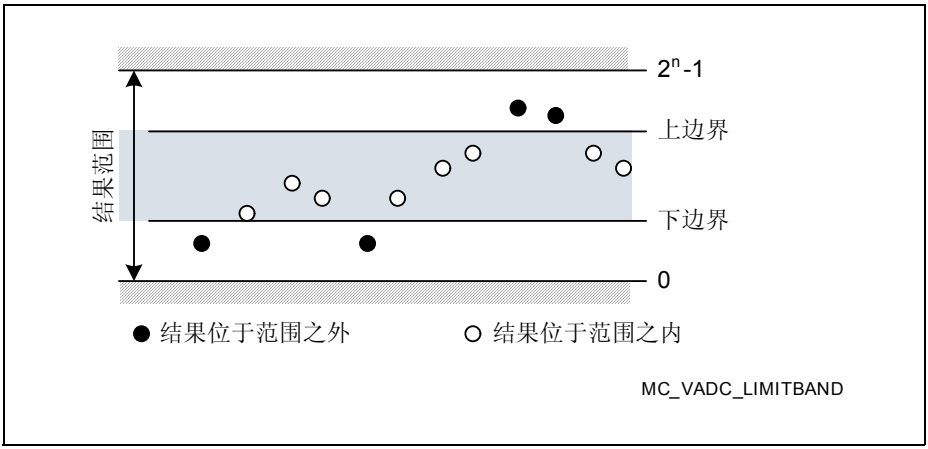


图 16-13　　通过极限检测进行结果监测

当下面两个条件都满足时，认为结果值位于所定义的范围内部：

- 结果值小于或等于所选择的上边界值
- 结果值大于或等于所选择的下边界值

结果范围也可以分成两个区域：
要选择下部为有效范围，设置下边界为最小值 （000_H），上边界为最大的预期值。
要选择上部为有效范围，设置上边界为最大值 （FFF_H）下边界为最小的预期值。

寻找极值 （峰值检测）

极限检查机制使用标准转换，因此总是能够提供用于比较的实际转换结果。将该机制与一种特殊的 FIFO 模式相结合，不仅可提供通常的转换结果，还能同时保存一个转换序列的最大 （或最小）结果。这种特殊的 FIFO 模式只在结果大于 （或小于） FIFO 级中的当前值时才会更新相应的 FIFO 级。对于该操作，必须选择低于标准结果的 FIFO 级作为比较值。模式选择通过寄存器 GxRCRy 中的位域 FEN 来完成。
在启动一次峰值检测序列之前，要向峰值结果寄存器中的结果位域写一个合理的起始值（例如， 0000_H 用于查找最大值， $FFFF_H$ 用于查找最小值）。

16.8.5　使用快速比较模式

在快速比较模式下，输入信号直接与相关联的结果寄存器中的值进行比较。该比较提供一个二元结果 （高 / 低）。如果不需要确切的结果值，这样做可节省转换时间。当输入信号变为高于（或低于）比较值时，会产生通道事件（见位域 CHEVMODE 和**图 16-14**）。
在快速比较模式，比较值取自结果寄存器。寄存器 **GxBOUND (x = 0 - 1)** 中的位域 BOUNDARY1 和 BOUNDARY0 定义这种情况下的增量极限。这些增量被加到原始的比较值 （或从其中减去），并且允许定义一个任意的滞回区间。

实际使用的比较值取决于快速比较结果 FCR （见寄存器 **G0RESy (y = 0 - 15)**）：
GxRESy.FCR = 0：参考值 + 向上增量
(GxRESy.RESULT + GxBOUND.BOUNDARY0)
GxRESy.FCR = 1：参考值 - 向下增量
(GxRESy.RESULT - GxBOUND.BOUNDARY1)

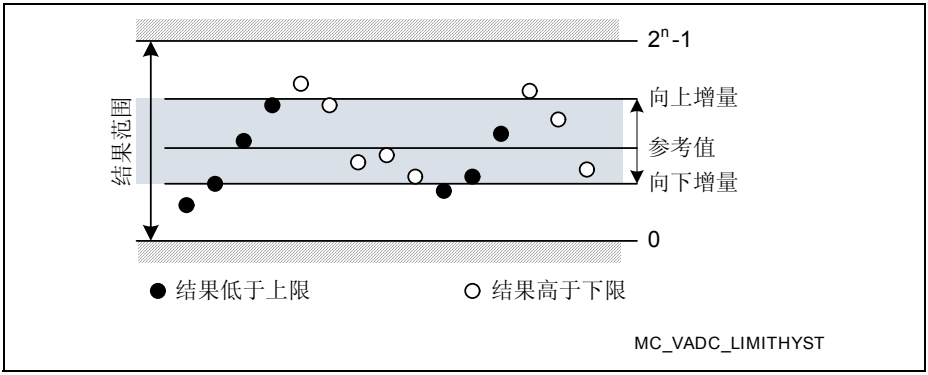


图 16-14　　带滞回电压的比较监视结果

16.8.6　边界标志控制

可以通过配置两种极限检查机制来自动控制边界标志。这些边界标志也可作为其他模块的控制信号。当转换值超过所定义的值时，标志可被置位或清零，输出信号的极性是可以选择的。当门控有效时，可选择一个门控信号来使能边界标志操作。

每个边界标志在组专用输出线上都可用。节点指针还将它们连接到 4 个边界信号之一或相关联的公共服务请求线之一，见寄存器 **GxBFLNP (x = 0 - 1)**。

对于标准转换，当转换结果大于所定义的边界时，边界标志被置位。当转换结果小于所定义的边带时，边界标志被清零。

两个边界值之间的范围定义了一个滞回区间 （对应置位 / 清零边界标志）。
在直流无刷电机应用中，需要检测三个线性霍尔元件的输出信息，这时可以利用这种特性获取有效信息。

在快速比较模式，边界标志反映比较的结果，即当要被比较的输入信号电平高于比较值时，边界标志将被置位 / 清零，并且当信号电平低于比较值时，会被清零 / 置位。增量值定义了围绕比较值的滞回区间。

注：　如果不希望有滞回区间，可将寄存器 GxBOUND (即增量) 清零。

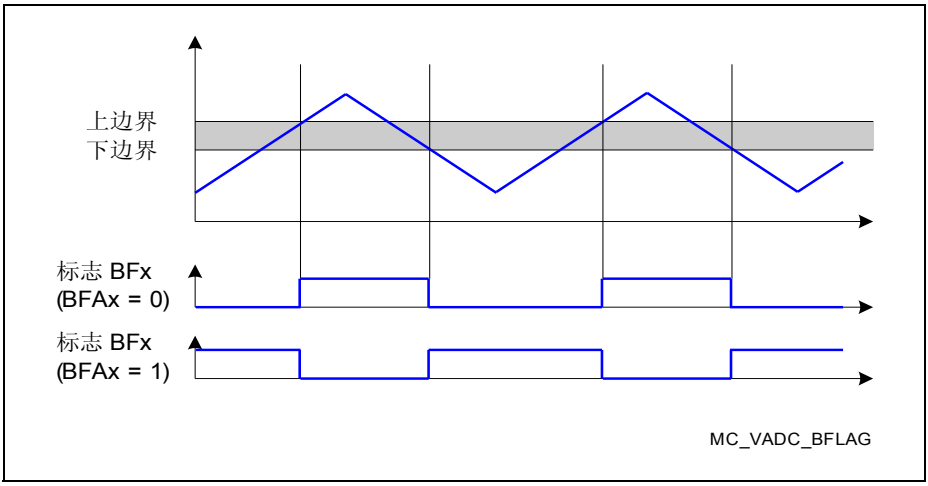


图 16-15　　边界标志切换

每次比较操作都可切换边界标志，或者说比较操作的影响仅限于相应请求源门控信号的活动阶段 (见 **GxBFLC (x = 0 - 1)**)。

注：　如果将边界标志与快速比较模式结合使用，建议不要将来自其他通道的结果定向到相应的结果寄存器。

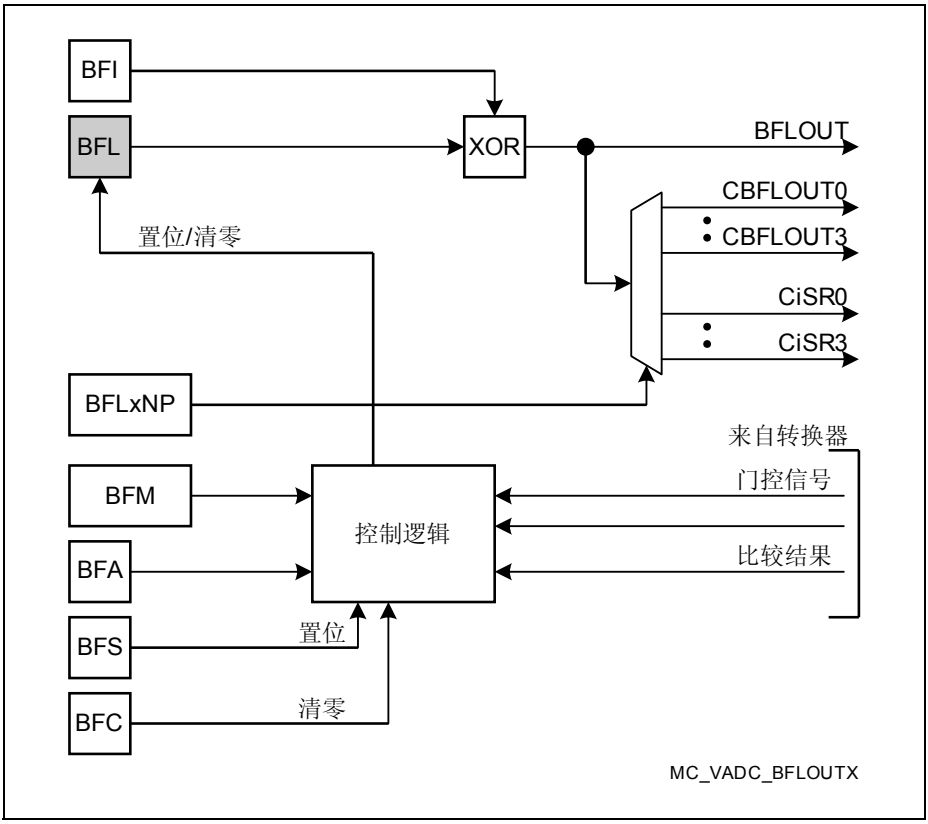


图 16-16　　边界标志控制

边界标志 BFLy 被分配给结果寄存器 GxRESy，因此可以被分配给任何一个通道。

16.9　　转换时序安排

VADC 的转换器集群将一个高速转换器与多个可并行工作的采样保持单元组合在一起。XMC1300 包含 1 个带有 2 个采样保持单元的集群。

由于所有的采样保持单元都使用 1 个转换器，实际的转换必须与特定的采样保持单元相关联。一个 8 级步进转换器连续扫描一个集群中所有采样保持单元的转换请求，如果发现其中有一个处于活动状态，就立即启动转换。

这 8 步中的每一步都可以被分配给一个可用组（通过寄存器 **SHS0_STEPCFG** 中的位域 KSELx），并可被使能参加循环方案。复位后的默认设置建立一个线性循环方案，该方案

使能并一个接一个地扫描所有的接口。在这种情况下，所有的接口被以相同的方式处理。其他步被禁止。

通过在步进序列内多次选择特定组，相关联的采样保持单元可以被更频繁地扫描。尤其在加速时序模式，这使这些内核有更高的转换速率。该功能使 SHADC 的性能这些特性大为提升。

例如：

默认：0 - 1 (2) (3) (4) (5) (6) (7) [2 个采样保持单元：步 2-7 被禁止]

加速 1：0 - 1 - 1 (3) (4) (5) (6) (7) [步 3-7 被禁止]

注：　步进器的默认配置提供向上兼容的操作。

空步，即相关联的采样保持单元没有转换请求的步，被忽略并且步进器继续处理下一个被使能的步。

如果所有的步都为空，则步进器进入其空闲状态，直到有一个转换请求或校准请求 (见节) 被激活。

一般的时序行为

在经过一段转换时间 （取决于 f_{ADCI}、采样时间和转换宽度等参数）后，转换结果被送到数字逻辑处理部分。这种兼容的时序模式提供与以前的模块相同的行为。这有利于将现有的应用程序移植到 XMC1300。

为了充分利用高速转换器的性能，可通过将寄存器 TIMCFG 中的位 AT 置 1 来使能加速时序模式。在这种情况下，只要转换结束，就会立即返回每个结果值。结合对特定组的加速机制 （见上述），可获得很高的转换速率。

当然，还可以通过减少采样时间 （位 SST）来获得更高的转换速率。

16.10 转换时序

一次转换所需的总时间包括从采样阶段[1] 启动到结果可用的时间。有两种模式可供选择：

- **兼容时序模式：**
  转换时序被定义为与以前的 ADC 模块相同。这使现有应用可以实现平滑的产品升级。
- **加速时序模式：**
  转换时序充分利用高速转换器和可编程步进器的性能，可获得更高的转换速率。

转换触发的频率还依赖下面几个可配置的因素：

- 选择的转换时间，根据输入类定义。对使用外部多路复用器的转换，还包括延长的采样时间计数。
- 必须重复执行被撤消的转换所带来的延迟。
- 由于其他通道的等距采样所带来的延迟。
- 已配置的仲裁周期时间。
- 外部触发信号的频率，如果外部触发被使能。

16.10.1 兼容时序模式

转换时序取决于以下用户可定义的因素：

- ADC 转换时钟频率， $f_{ADCI} = f_{ADC} / (DIVA+1)$[2]
- 所选择的采样时间，$t_S = (2 + STC) \times t_{ADCI}$
  (STC = 附加的采样时间，参见**表 16-11**)
- 结果宽度 N (8/10/12 位)
- 后校准时间 PC，如果已选择 (PC = 2，否则为 0)
- 同步时间 （运行时钟为模块时钟）

转换时间为采样时间、转换步数和同步时间的总和，可通过下面的公式计算：

$$t_{CN} = (2 + STC + N + PC) \times t_{ADCI} + 2 \times t_{ADC}$$

1) 从请求相应转换的触发事件起到采样阶段开始为止的这段时间取决于仲裁过程，因此只能在系统设置已知的情况下才能确定。

2) f_{ADCI} 是在兼容时序模式下用来控制转换时序的实际频率。

兼容模式的时序示例

假设系统参数如下：
f_{ADC} = 32 MHz 即 t_{ADC} = 31.25 ns， DIVA = 1， f_{ADCI} = 16 MHz 即 t_{ADCI} = 62.5 ns
根据给定的公式，可以获得下面的最小转换时间：
12 位转换 （带校准）：
$t_{CN12C} = (2 + 12 + 2) \times t_{ADCI} + 2 \times t_{ADC} = 16 \times 62.5\ ns + 2 \times 31.25\ ns = 1\ 062.5\ ns$
10 位转换 （不带校准）：
$t_{CN10} = (2 + 10) \times t_{ADCI} + 2 \times t_{ADC} = 12 \times 62.5\ ns + 2 \times 31.25\ ns = 812.5\ ns$

16.10.2 加速时序模式

在加速时序模式，不使用标准时序配置 （如上描述），每个结果只要一产生，就立即可用。因此实际的响应时间取决于该转换在集群转换周期内的位置。
转换时序取决于以下用户可定义的因素：

- 转换时钟频率，$f_{SH} = f_{CONV} / (DIVS+1)$
- 选择的采样时间，$t_S = SST \times t_{ADC}$ (短采样时间)
- 结果宽度 N (8/10/12 位)
- 后校准时间 PC，如果已选择 (PC = 2, 否则为 0)
- 该转换在步进器序列内的位置
- 同步时间 （运行时钟为模块时钟）

转换时间为采样时间、转换步数和同步时间的总和，可通过下面的公式计算：

$t_{CN} = (SST + 5^{1)}) \times t_{ADC} + (N + 10) \times t_{SH} + (4 \times t_{SH})^{2)}$

加速模式的时间示例

假设系统参数如下：
f_{CONV} = 32 MHz， DIVS = 0，即 $t_{SH} = t_{CONV}$ = 31.25 ns，
f_{ADC} = 32 MHz，即 t_{ADC} = 31.25 ns，
SST = 5，即 t_S = 312.5 ns
根据给定的公式，可以获得下面的最小转换时间：
(括号中的参数适用于一个周期内的第二次转换)
12 位转换：
$t_{CN12} = 10 \times t_{ADC} + 26 \times t_{SH} = 312.5\ ns + 26 \times 31.25\ ns = 1\ 125\ ns\ (1\ 812.5\ ns)$
10 位转换：
$t_{CN10} = 10 \times t_{ADC} + 24 \times t_{SH} = 312.5\ ns + 24 \times 31.25\ ns = 1\ 062.5\ ns\ (1\ 687.5\ ns)$
最大的校准步[3)] 还需要另外 13× t_{SH}，因此含有 2 个采样保持单元的完整转换周期需要：

1) 传送转换结果到数字结果寄存器需要这些时钟周期。只有在计算集群的总体转换性能时才需要计入这些周期。
2) 在采样阶段结束与转换开始之间需要这些时钟周期。如果该阶段发生在另一转换期间，它们可以被排除。
3) 校准步骤交替执行，其中增益校准需要比偏移校准更长的时间。

t_{CR} = 1 812.5 ns + 13 × 31.25 ns = 2 219 ns (12 位转换)

16.11 转换结果处理

A/D转换器可以在一定程度上对转换结果数据进行预处理，然后再保存结果供CPU读取。该特性支持应用软件对结果数据的后续处理 。

转换结果处理有以下功能：

- **转换结果的存储**，保存转换结果到用户可配置的寄存器。
- **数据对齐**，根据结果宽度和端格式
- **待读模式**，避免丢失数据
- **结果事件产生**
- 数据缩减或抗混叠滤波 (见 **16.11.6 节**)

16.11.1 转换结果的存储

一个特定组的转换结果值可保存到 16 个相关联的组结果寄存器之一，或者保存到公共的全局结果寄存器 （例如，可为后台源的通道使用）（见**选择一个结果寄存器**）。

该结构为不同通道组的转换结果提供了不同的存放位置。根据应用的需求不同 （数据缩减、自动扫描、别名功能等），用户可以分配转换结果，从而最小化 CPU 的负荷。

每个结果寄存器有一个与之相关联的独立的数据有效标志 （VF）。该标志指示何时有 “新 ” 有效数据被保存到相应的结果寄存器以及何时被读取。

对于标准转换，结果值在位域 RESULT 中可用。在快速比较模式， ADC 转换使用位域 RESULT 作为参考值，因此操作的结果被保存到位 FCR 中。

结果寄存器可通过两种不同的模式读取。这些模式使用不同的地址但访问的是相同的寄存器数据：

- 当通过**应用模式**读取结果寄存器时，读取结果时相应的有效标志被自动清零。这就在结果产生和读取之间提供了简单的握手。应用模式还支持待读模式。
- 当通过**调试模式**读取结果寄存器时，读取结果时相应的有效标志保持不变。调试模式通过提供结果值但不扰乱与应用的握手来支持系统调试。

应用程序可通过多个结果寄存器读取转换结果：

- 组结果寄存器：
  返回结果值和通道号
- 全局结果寄存器：
  返回结果值和通道号以及组号

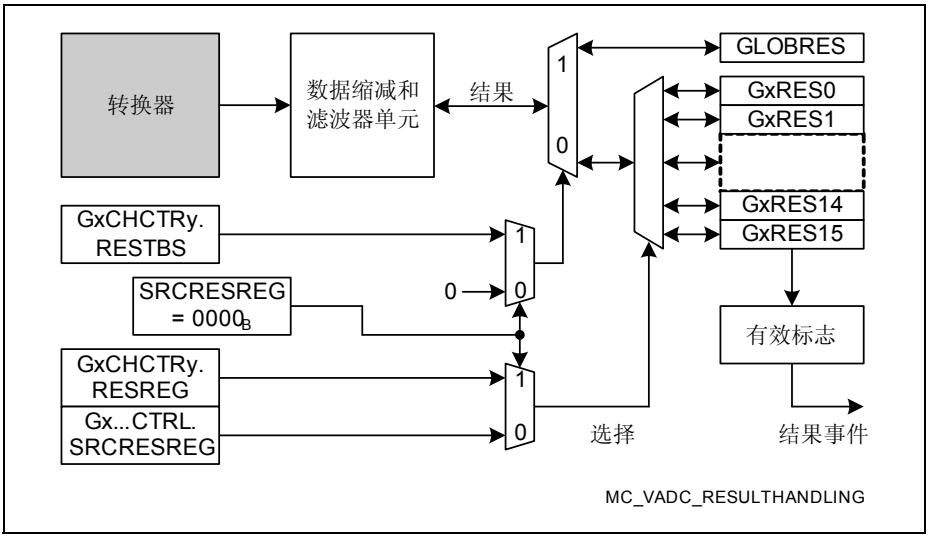


图 16-17　　转换结果存储

选择一个结果寄存器

转换结果被保存到可由用户根据应用需求分配的结果寄存器。下面的位域将结果定向到一个寄存器。

- 寄存器 **GxQCTRL0 (x = 0 - 1)**、**GxASCTRL (x = 0 - 1)** 或 **BRSCTRL** 中的 SRCRESREG
  当使用源专用的结果寄存器时，选择组专用结果寄存器 GxRES1 … GxRES15
- 寄存器 **G0CHCTRy (y = 0 - 7)** 中的 RESTBS
  为由后台源请求的结果选择全局结果寄存器。
- 寄存器 **G0CHCTRy (y = 0 - 7)** 中的 RESREG
  当使用通道专用的结果寄存器时，选择组专用结果寄存器 GxRES0 … GxRES15。

使用源专用结果寄存器允许分离由不同请求源请求的同一通道的结果。一般来说，这些请求源被不同的任务使用，在不同的时间被触发。

16.11.2　数据对齐

一个转换结果值在所选结果寄存器中的位置取决于 3 个配置选项 (见**图 16-18**)：

- 选择的结果宽度 (12/10/8 位，由转换模式选择)
- 选择的结果位置 （左 / 右对齐）
- 选择的数据累加模式 (数据缩减)

这些选项可使应用软件在进行转换结果处理时最大限度地减小处理器负荷。

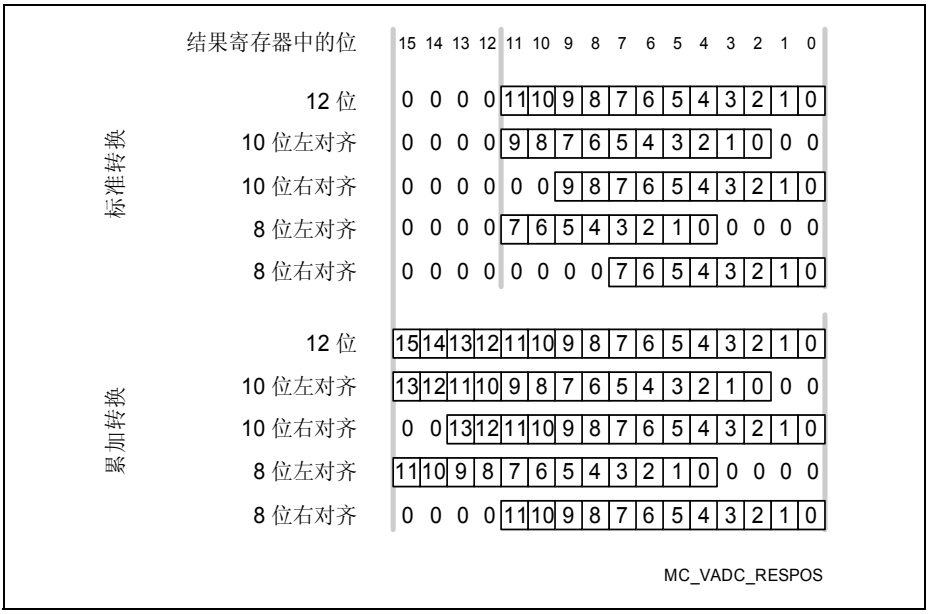


图 16-18　　结果存储选项

可通过软件写位域 RESULT 来为快速比较模式提供参考值。在该模式，使用位 11-2 的值，其他位被忽略。

16.11.3　待读模式

待读模式下可以防止因在 CPU 读取先前的数据之前有新的转换结果覆盖了结果寄存器而引起的数据丢失。例如，自动扫描转换序列或时序要求 “ 不太严格的 ” 其他序列可能使用一个公共的结果寄存器。然而，转换结果来自不同的输入通道，因此一次覆盖会破坏先前的转换结果[1)]。

待读模式会自动将来自该源的这一通道的转换启动挂起，直到当前结果被读取。因此，硬件或软件触发信号可以请求一次转换或一个转换序列，但每次转换都只能在先前的结果被读取后才启动。这就将转换序列根据 CPU 读取先前转换结果的能力（响应延迟）进行了自动调整。

如果为一个结果寄存器使能了待读模式 （位 GxRCRy.WFR = 1），在目标结果寄存器包含有效数据（由有效标志 VF = 1 指示）期间或者当前正运行的一次转换以同一结果寄存器为目标，请求源不会产生转换请求。

1) 一个通道的重复转换在使用不同的结果寄存器的情况下是不会破坏其他结果的，但会更新其自身的先前结果值。因此，结果寄存器中总是保存当前信号数据。

如果两个请求源以同一个被选择为待读模式的结果寄存器为目标，则高优先级的请求源不能中断一个在其产生转换请求之前就已经启动的低优先级转换请求。在这种情况下，撤消 - 插入 - 重复模式不工作。特别是，如果所涉及的一个请求源是后台源（通常具有最低优先级），就必须引起重视。如果高优先级请求的目标是一个不同的结果寄存器，低优先级的转换可被撤消，并在随后重复执行。

注：　对于同步从器件的同步转换，待读模式被忽略。(见 **16.12 节** *)。*

16.11.4　结果 FIFO 缓冲区

可将结果寄存器用作转换结果的直接目标，也可以将它们与同一个 ADC 组的其他结果寄存器级联起来组成一个结果 FIFO 缓冲区（先进先出缓冲机制）。一个结果 FIFO 可存储多个测量结果，CPU 可随后以 “ 不严格的 ” 响应时序读取这些测量结果。可以使用可用的结果寄存器设置多个 FIFO 缓冲区结构。

两个或多个结果寄存器组成的FIFO结构是通过将结果寄存器与其下一个相邻的结果寄存器（下一个具有较高索引值的寄存器，见图　16-19）级联构成。通过设置位域 GxRCRy.FEN = 01_B 来使能这种级联。

转换结果保存在一个 FIFO 结构中具有最高索引值的寄存器中。软件从具有最低索引值的 FIFO 寄存器读取结果值。

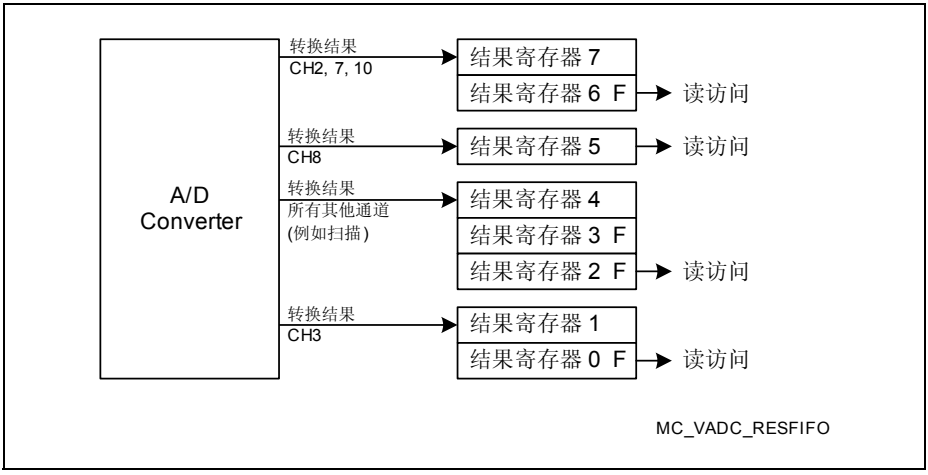


图 16-19　结果 FIFO 缓冲区

在图 16-19 所示的例子中，以下面的方式对结果寄存器进行了配置：

- 2 级缓冲区包含结果寄存器 7-6
- 专用结果寄存器 5
- 3 级缓冲区包含结果寄存器 4-3-2
- 2 级缓冲区包含结果寄存器 1-0

表 16-4 列出了将结果寄存器组合起来构建结果 FIFO 缓冲区所需要的结果寄存器配置。

表 16-4　　结果 FIFO 寄存器的特性

功能	输入级	中间级	输出级
结果目标	是	否	否
应用程序读	否	否	是
数据缩减模式	是	否	否
待读模式	是	否	否
结果事件中断	否	否	是
FIFO 使能 (FEN)	00_B	01_B	01_B
例子中的寄存器	7，4，1	3	6，2，0

注：　如果被使能，进入 FIFO 的每个数据字都会产生一次中断。

16.11.5　结果事件产生

当一个新值被保存到一个结果寄存器时，会产生结果事件。结果事件会因为数据累加而受到限制，并且结果事件只能在累加完成之后产生。
结果事件也可被完全禁止。

16.11.6　数据修改

转换的数据结果在被应用程序使用之前，可被自动修改。有多个选项可供选择（通过寄存器 **G0RCRy (y = 0 - 15)** 中的位 DMM），以减少转存和 / 或处理转换数据所需要的 CPU 负荷。

- **标准数据缩减模式 (对 GxRES0 … GxRES15)：**
  在产生结果中断之前，累加每个结果寄存器的 2、3 或 4 个结果值。该操作可去除来自输入信号的部分噪声。
- **结果滤波模式 (FIR，对 GxRES7、GxRES15)：**
  将一个 3 阶有限冲击响应滤波器 (FIR) 应用于所选结果寄存器中的转换结果，滤波器的参数可由用户选择。
- **结果滤波模式 (IIR，对 GxRES7、GxRES15)：**
  将一个 1 阶无限冲击响应滤波器 (IIR) 应用于所选结果寄存器中的转换结果，滤波器的参数可由用户选择。
- **差分模式 (对 GxRES1 … GxRES15)：**
  从所选结果寄存器的转换结果中减去结果寄存器 GxRES0 的内容。位域 DRCTR 在该模式下未使用。

表 16-5　位域 DRCTR 的功能

DRCTR	标准数据缩减模式 (DMM = 00_B)	DRCTR	结果滤波模式 (DMM = 01_B)[1]
0000_B	数据缩减禁止	0000_B	FIR 滤波器：a=2, b=1, c=0
0001_B	累加 2 个结果值	0001_B	FIR 滤波器：a=1, b=2, c=0
0010_B	累加 3 个结果值	0010_B	FIR 滤波器：a=2, b=0, c=1
0011_B	累加 4 个结果值	0011_B	FIR 滤波器：a=1, b=1, c=1
0100_B	保留	0100_B	FIR 滤波器：a=1， b=0， c=2
0101_B	保留	0101_B	FIR 滤波器：a=3， b=1， c=0
0110_B	保留	0110_B	FIR 滤波器：a=2， b=2， c=0
0111_B	保留	0111_B	FIR 滤波器：a=1， b=3， c=0
1000_B	保留	1000_B	FIR 滤波器：a=3， b=0， c=1
1001_B	保留	1001_B	FIR 滤波器：a=2， b=1， c=1
1010_B	保留	1010_B	FIR 滤波器：a=1， b=2， c=1
1011_B	保留	1011_B	FIR 滤波器：a=2， b=0， c=2
1100_B	保留	1100_B	FIR 滤波器：a=1， b=1， c=2
1101_B	保留	1101_B	FIR 滤波器：a=1， b=0， c=3
1110_B	保留	1110_B	IIR 滤波器：a=2， b=2
1111_B	保留	1111_B	IIR 滤波器：a=3， b=4

1) 当位域 DMM ≠ 01_B 时，滤波器寄存器被清零。

标准数据缩减模式

数据缩减模式可用作抗混叠数字滤波器或用于数据抽取目的。它可最多累加 4 个转换值来产生最终结果。

每个结果寄存器都可被单独使能为数据缩减模式，由寄存器 **G0CHCTRy (y = 0 - 7)** 中的位域 DRCTR 控制。数据缩减计数器 DRC 指示累加的实际状态。

注：　可在要累加的转换值之间插入其他结果寄存器的转换值。

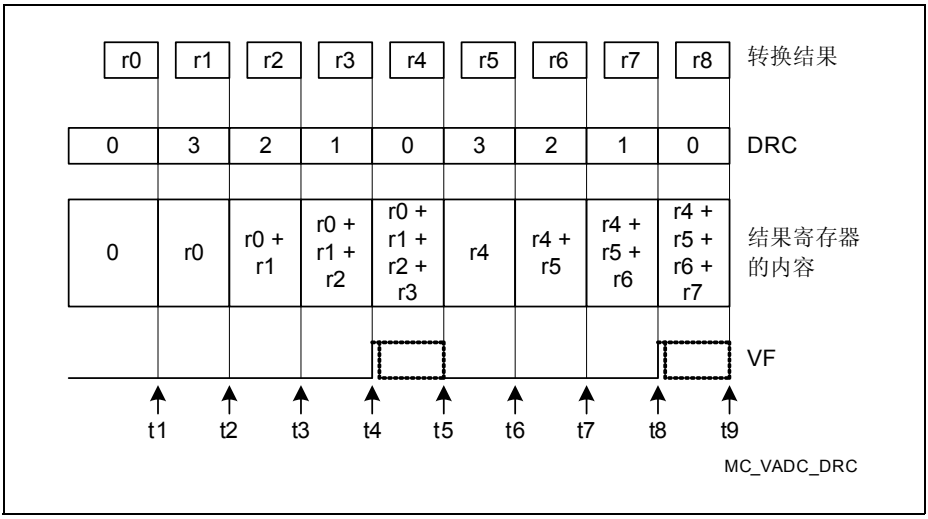


图 16-20　　标准数据缩减滤波器

该示例给出了累加 4 个转换结果的一个数据缩减序列。 8 个转换结果 (r0 … r7) 被累加，产生 2 个最终结果。

当一次转换完成并且数据被保存到数据缩减模式已被使能的结果寄存器时，数据缩减计数器 DRC 控制对数据的处理：

- 如果 DRC = 0 (本例中的 t1、 t5、 t9)，转换结果被保存到寄存器中。DRC 被加载为位域 DRCTR 中的内容。 (即累加开始)。
- 如果 DRC > 0 (本例中的 t2、t3、t4 和 t6、t7、t8)，转换结果与结果寄存器内的值相加。DRC 减 1。
- 如果 DRC 变成 0，不论是从 1 减到 0 （本例中的 t4 和 t8）还是从 DRCTR 加载，相应结果寄存器的有效位被置 1 并发生结果寄存器事件。
  最终结果必须在下一个数据缩减序列启动之前 （本例中的 t5 或 t9 之前）被读取。结果读取操作会自动清除有效标志。

*注：　软件可通过清除相应的有效标志 (通过 **GxVFR (x = 0 - 1)**) 将数据缩减计数器 DRC 清零。*

可通过将相邻结果寄存器联合起来构建一个结果FIFO的方式来增加读取最终数据缩减结果的响应时间 (见**图 16-21**)。在这种情况下， 数据缩减序列的最终结果被加载到相邻的寄存器。在下一个数据缩减序列结束之前 （第二个例子中的 t8），都可以从该相邻寄存器中读取最终结果值。

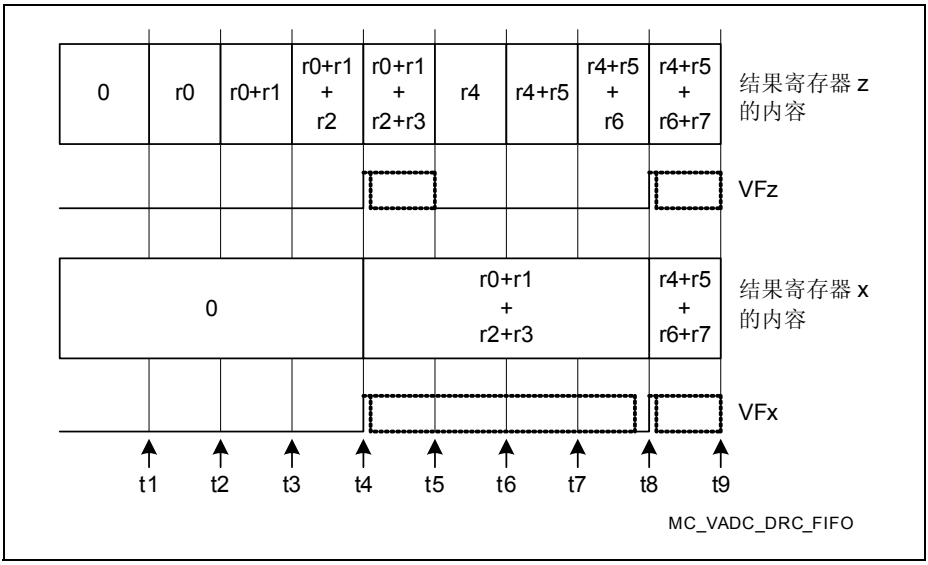


图 16-21　　FIFO 被使能的标准数据缩减滤波器

有限冲激响应滤波器模式 (FIR)

FIR 滤波器 (见**图 16-22**) 为中间结果 （RB1、 RB2）提供 2 个结果缓冲器和 3 个可配置的抽头系数 (a、 b、 c)。

转换结果与中间结果缓冲器中的值以各自对应的抽头系数为权值进行加权相加，形成结果寄存器的最终值。有几组预定义的抽头系数，可通过寄存器 **G0RESy (y = 0 - 15)** 和 **GLOBRES** 中的位域 DRCTR （编码列于**表 16-5** 中）选择。这些系数使 ADC 结果有 3 或 4 倍的增益，产生一个 14 位的结果值。每次采样都会激活有效标志 （VF），即有效标志为每次采样产生一个有效结果。

注：　可在要滤波的转换值之间插入其他结果寄存器的转换值。

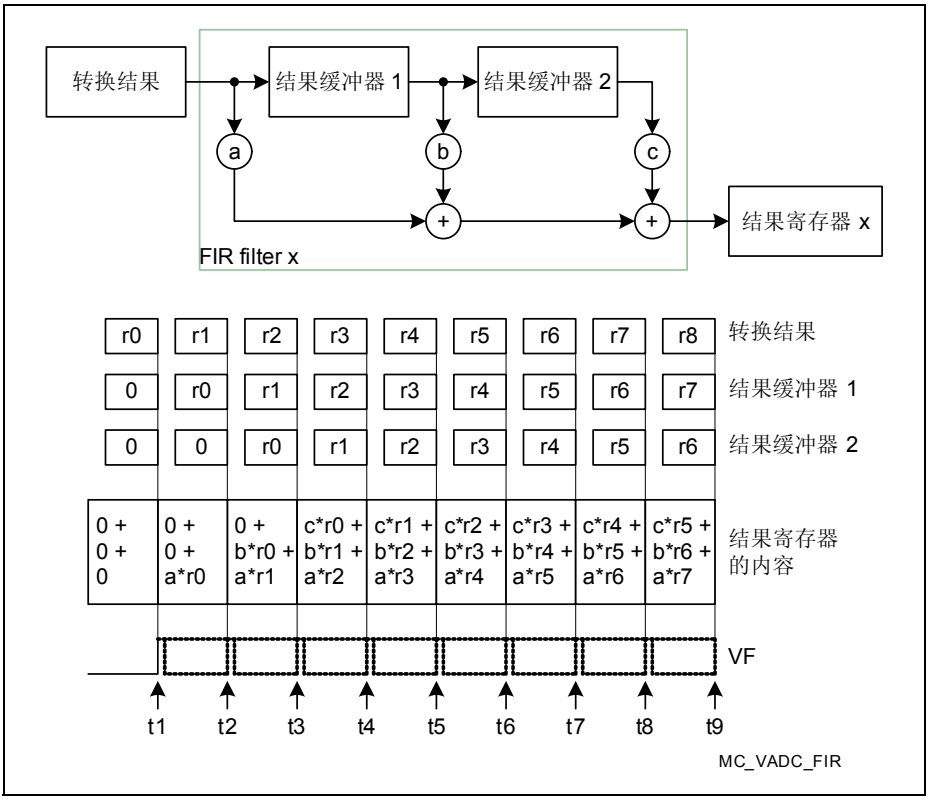


图 16-22　　FIR 滤波器结构

注：　当位域 DMM ≠ 01_B 时，滤波器寄存器被清零。

无限冲激响应滤波器模式 (IIR)

IIR 滤波器（见**图 16-23**）提供一个结果缓冲器和 2 个可配置的抽头系数（a、b）。这是一个一阶低通滤波器。

转换结果用对应的系数加权后与前一结果值的一部分相加，形成结果寄存器的最终值。有几组预定义的抽头系数，可通过寄存器 **G0RESy (y = 0 - 15)** 和 **GLOBRES** 中的位域 DRCTR （编码列于**表 16-5**）选择。这些系数使 ADC 结果有 4 倍的增益，产生一个 14 位值。每次采样都会激活有效标志 （VF），即有效标志为每次采样产生一个有效结果。

注： 可在要滤波的转换值之间插入其他结果寄存器的转换值。

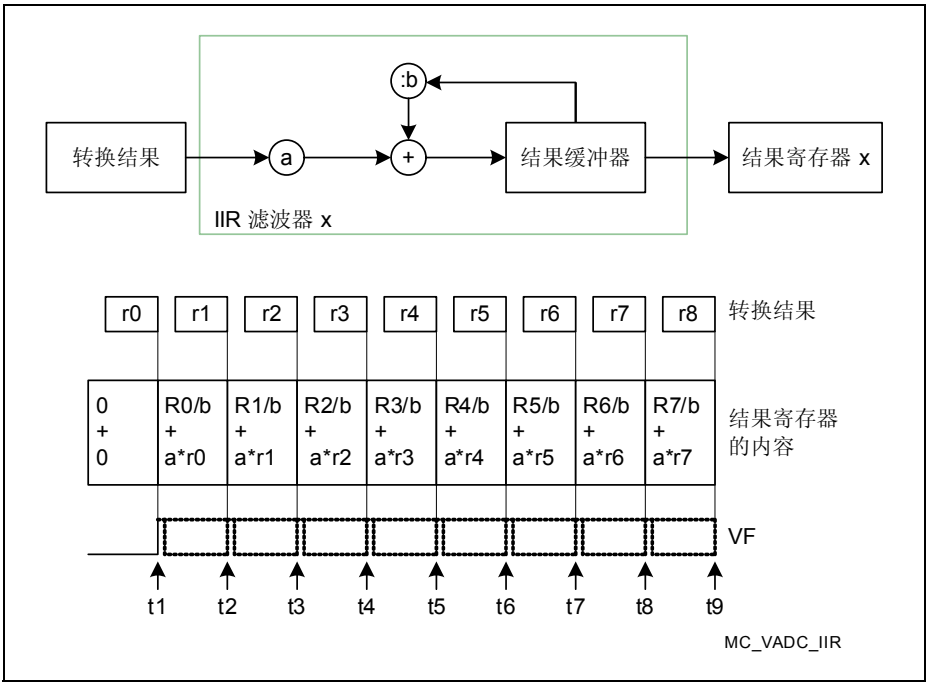


图 16-23 IIR 滤波器结构

注： 当位域 DMM ≠ 01_B 时，滤波器寄存器被清零。

差分模式

从当前结果中减去结果寄存器 0 的内容，得到当前通道相对于参考通道的结果。该模式无需软件操作。

参考通道必须将其结果保存到结果寄存器 0。参考值可一次性确定，然后用于一系列的转换，也可以在每次相关转换之前进行参考通道的转换。

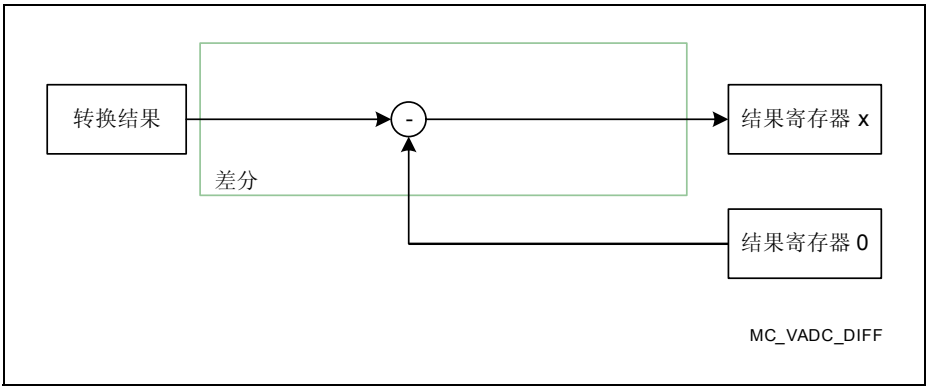


图 16-24　　结果差分

16.12 转换同步

ADC 内核的转换可根据内核配置按自定时间进行，也可由外部（ADC 的外部）信号触发。

同步转换 支持在一个同步组[1] 内进行通道的并行转换。该功能会优化像电气驱动控制这样的应用。

等距抽样 支持具有最小抖动的固定间隔转换。该功能会优化像滤波器算法或音频方面的应用。

16.12.1 并行采样的同步转换

在XMC1300实现的多个独立的ADC内核可同步工作，用于同时测量多个模拟输入通道。在没有并行转换请求时，各内核可独立工作。

并行转换的同步机制确保相关通道的采样阶段同时开始。同步的内核对由主内核请求的同一通道进行转换。参加同步转换的每个内核都可以有不同的分辨率和采样阶段时间。

可为每个输入通道单独请求并行转换 （一个或多个）。在**图 16-25** 所示的例子中， ADC 内核 0 和 1 的输入通道 CH3 同步转换，其他输入通道不引发并行转换。

一个内核作为同步主内核，其他内核作为同步从内核。每个内核都可以作为主内核或从内核。主内核和从内核构成一个 “ 转换组 ” 来控制并行采样：

- **同步主** ADC 内核可请求对某一通道（相应的通道控制寄存器 **G0CHCTRy (y = 0 - 7)** 中的 SYNC = 1）的同步转换，相连的从 ADC 内核也被请求对该通道进行转换。
  主内核支持待读模式。
- **同步从** ADC 内核响应来自主内核的同步转换请求。当没有同步转换请求时，从内核可执行 “ 本地 ” 转换。
  – 从内核的时序必须根据主内核的时序 （寄存器 **GxARBPR (x = 0 - 1)** 中的 ARBRND）来配置，这样才能使能并行转换。
  – 并行转换请求总是被作为最高优先级的请求处理，支持撤消 – 插入 – 重复模式。
  – 待读模式在从内核中被忽略。先前的结果有可能被覆盖，尤其是在其它转换也使用同一结果寄存器时。
- 对于同步从内核，仲裁器必须一直运行 （位 **GxARBPR (x = 0 - 1)**ARBM = 0）。
  为使仲裁器能同步运行，要在初始化主内核之前初始化从内核。
- 一旦启动，并行转换不能被中止。

1) 请参见 **16-135 页 “XMC1300 中的同步组 ”** 。

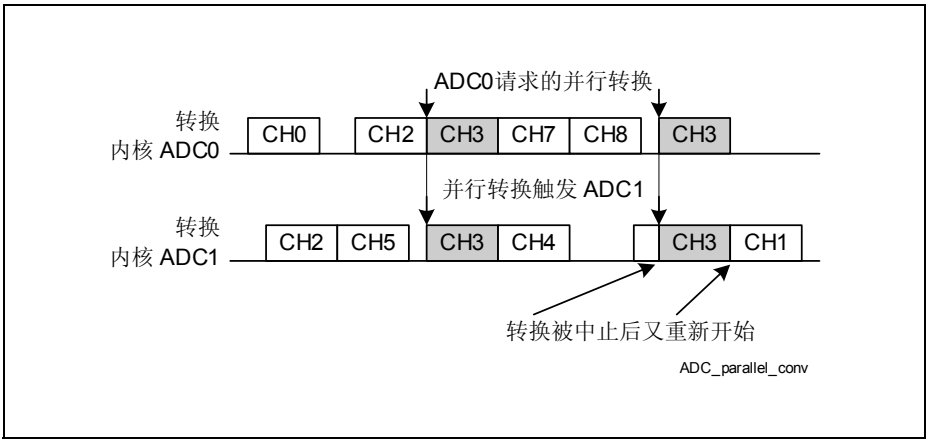


图 16-25　　并行转换

图 16-25 所示的例子使用对通道 CH3 的同步转换。其它通道转换由自己的内核控制。ADC0 是主内核， ADC1 是从内核。

同步主内核通过提供控制信息 **GxARBCFG (x = 0 - 1)**ANONS （见**图 16-26**）和所请求的通道号来控制从内核。位域 **GxSYNCTR (x = 0 - 1)**.STSEL 为主内核 (00_B) 和从内核 (01_B/10_B/11_B) 选择 ANON 的信息源。

STSEL = 00_B 总是选择自己的 ANON 信息，因此意味着这是同步主内核或可以独立工作。其他控制输入 (STSEL = 01_B/10_B/11_B) 按从小到大的顺序被连接到同步组的其他内核 (参见 **16-135 页的表 16-13 “XMC1300 中的同步组 ”**)。

就绪信号指示从内核何时准备好以启动一次并行转换的采样阶段。位 **GxSYNCTR (x = 0 - 1)**.EVALR1 = 1 使能就绪信号的控制功能。

*注：　同步转换会请求由主内核定义的同一通道号。使用别名功能 （见 **16.8.2 节**），可对来自不同输入通道的模拟信号进行转换。如果将 （例如） CH0 用作备用参考，这是很有利的。*

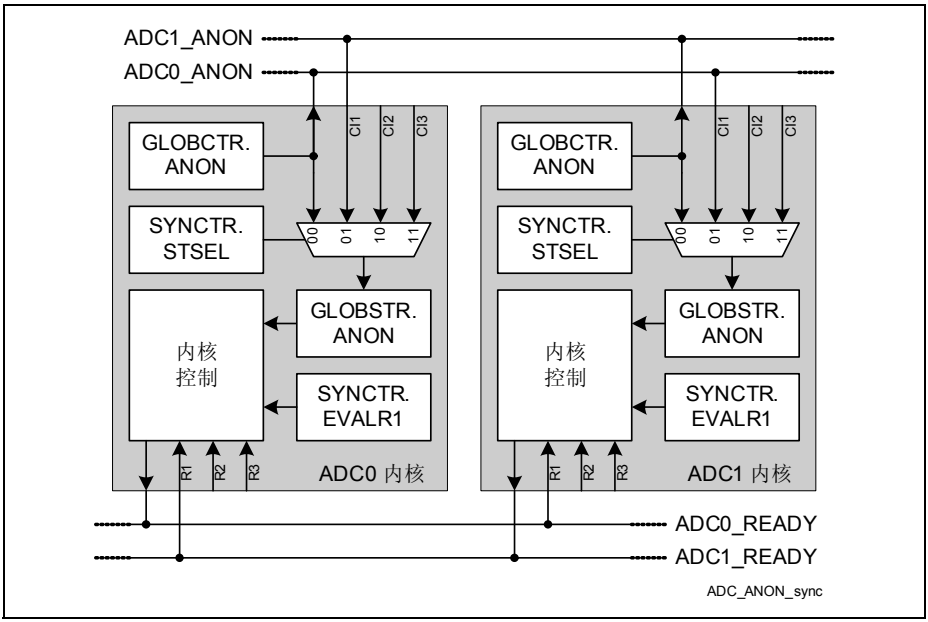


图 16-26　　通过 ANON 和就绪信号实现同步

16.12.2　等距采样

为了优化输入数据，例如在滤波器或音频应用中，转换可按固定的时间间隔执行。等距采样的转换由一个外部信号 （例如，一个定时器）触发。为了产生与仲裁器同步的触发信号，ADC 提供了一个输出信号（ARBCNT），该信号在每个仲裁周期都会被激活一次，并且用作触发定时器的时基。在这种情况下，仲裁器必须一直运行 (**GxARBPR (x = 0 - 1)**.ARBM = 0)。如果定时器有独立的时基，在没有请求处于挂起状态时，仲裁器被停止。

前序时间 （见**图 16-27**）必须长于一个仲裁周期和可能的最长转换时间。

需要为意向的等距转换源选择定时器模式（寄存器 **GxQCTRL0 (x = 0 - 1)** 或 **GxASCTRL (x = 0 - 1)** 中的 TMEN = 1）。在定时器模式，该源的请求被触发并被仲裁，但只能在触发信号被移除 （见**图 16-27**）并且转换器空闲时才被启动。

为了确保转换器是空闲的并且转换的启动可由该触发信号控制，等距转换请求必须具有最高优先级。请求触发和转换启动之间的前序时间必须足够长，以使当前的活动转换完成。

信号 REQTRx 的频率定义采样速率，其高电平时间定义前序时间，相应的请求源在前序时间内参加仲裁。

也可以对一个通道序列进行等距采样，这取决于所用的请求源。如果前序时间与等距转换不重叠，也可以对多个并行请求源进行等距采样。

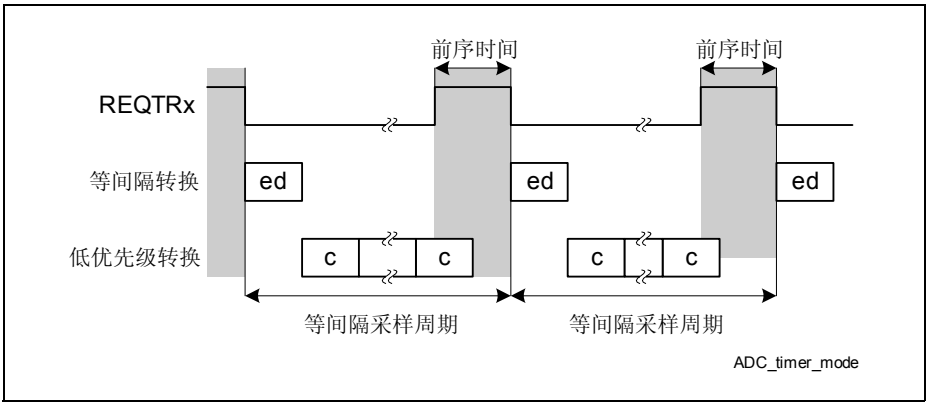


图 16-27　　等距采样的定时器模式

16.13 安全功能

VADC 还提供了几个测试功能，可以在实际应用中使用这些功能来验证模拟输入信号的有效性。

- **断线检测** 验证从传感器到输入引脚的连接是否正确。
- **多路复用器诊断** 验证内部模拟输入多路复用器的操作是否正确。

16.13.1 断线检测

为了测试一个外部模拟传感器与其输入引脚的连接是否正确，可在正常的采样阶段之前将转换器的电容预充到一个可选的值。如果到传感器的连接被中断，则随后的转换值代表预充值而不是预期的传感器结果。通过使用一个超出预期结果范围的预充电压 （断电检测最好使用 V_{AGND} 和 / 或 V_{AREF}），可以将一次有效的测量（与传感器相连）与一次故障 （传感器断开）区分开。

当断线检测被禁止时，转换器的电容被放电。

注： 如果断线检测被使能，完整转换的持续时间会因准备阶段 （与采样阶段相同）而增加。这会影响转换序列的时序。

可通过相应通道控制寄存器 **G0CHCTRy (y = 0 - 7)** 中的位域 BWDEN 来单独使能对每个通道的断线检测。该位域还选择预准备阶段的电平。

图 16-28 断线检测

16.13.2 多路复用器诊断

可通过激活额外的测试结构 （上拉、下拉）来测试从传感器到输入引脚的信号通路以及从输入引脚通过多路复用器到转换器的内部信号通路。这些拉器件为信号通路提供额外的负载。

当与一个已知的外部输入信号结合使用时，该测试功能可指示多路复用器是否将任何引脚连接到转换器的输入以及所连接的引脚是否是正确的引脚。

拉器件由标准端口控制寄存器激活。

16.14 外部多路复用器控制

可通过将外部模拟多路复用器连接到一个输入通道来增加模拟输入通道的数量。ADC 可被配置为能自动控制这些外部多路复用器 （EMUX）。
在该工作模式，可为每个可用的 EMUX 接口 (见寄存器 **EMUXSEL**) 选择一个通道。ADC 支持多个 8 选 1 多路复用器，有下面几个控制选项：

- **序列模式，**在遇到所选通道时，自动转换所有已配置的外部通道。在**图 16-29** 所示的例子中，完成以下转换：--4-32-31-30-2-1-0--4-32-31-30-2-1-0--…
- **单步模式，**在遇到所选通道时，转换所配置序列的一个外部通道。在**图 16-29** 所示的例子中，完成以下转换：--4-32-2-1-0--4-31-2-1-0--4-30-2-1-0--4-32-…
  (在对一个通道转换时，单步模式最佳)
- **稳定模式，**在遇到所选通道时，转换所配置的外部通道。在**图 16-29** 所示的例子中，完成以下转换：--4-32-2-1-0--4-32-2-1-0--4-32-2-1-0--…

*注： 在**图 16-29** 所示的例子中，有一个外部多路复用器连接到 CH3。假设起始选择值 EMUXSET 为 2。*

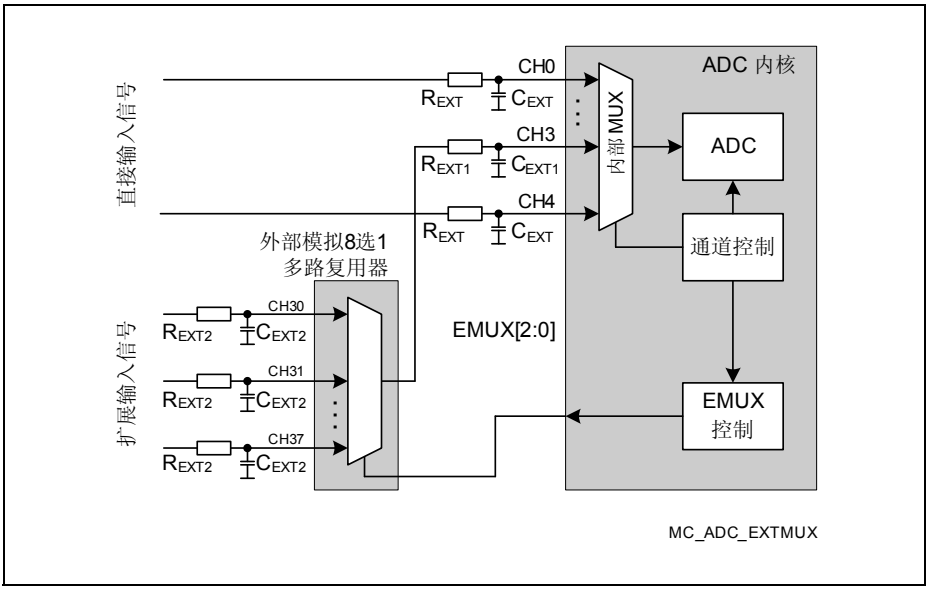


图 16-29 外部模拟多路复用器示例

位域 EMUXACT 决定发送给外部多路复用器的控制信息。
在单步模式， EMUXACT 在对一个已使能通道的每次转换后被更新。如果 EMUXACT = 000_B ，它被重新加载为位域 EMUXSET 的值，否则减 1。

附加的外部通道可能因信号路径被修改而具有不同的特性。可以在附加输入使用本地滤波器 (**图 16-29** 中 CH3x 上的 R_{EXT2}-C_{EXT2})。对于外部多路复用器与 ADC 模拟输入端之

间距离较远的应用，建议直接在 ADC 的模拟输入加一个 RC 滤波器 (图 16-29 中 CH3 上的 R_{EXT1}-C_{EXT1})。

注： 每个 RC 滤波器都会限制模拟输入信号的带宽。

因此，外部通道的转换使用另外的转换模式设置 CME。如果需要，可自动选择一种不同的转换模式。

对输入信号而言，切换外部多路复用器通常需要额外的建立时间。因此，在每次改变外部通道时，要使用另外的采样时间设置 STCE。这就会自动满足该情况下的不同采样时间需求。

在每个组中，任何一个通道都可被分配给外部多路复用器控制 (寄存器 **GxEMUXCTR (x = 0 - 1)**)。每个可用的端口接口都选择控制线为输出的组 (寄存器 **EMUXSEL**)。

控制信号

控制外部多路复用器的外部通道号能够以标准的二进制格式或格雷码格式输出。当选择一个通道序列时，格雷码可避免中间的多路复用器切换，因为每次只有一位发生改变。**表 16-6** 列出了编码结果。

表 16-6 EMUX 控制信号编码

通道	0	1	2	3	4	5	6	7
二进制	000_B	001_B	010_B	011_B	100_B	101_B	110_B	111_B
格雷码	000	001	011	010	110	111	101	100

无外部多路复用器的操作

如果在一个应用中不使用外部多路复用器，则控制寄存器的复位值就是合适的设置。

当 EMUXMODE = 00_B 时，禁止自动进行 EMUX 控制 。

由于控制输出信号是备选的端口输出信号，它们只在被明确选择时才可见于对应的引脚。

16.15 服务请求产生

每个 A/D 转换器组都可激活最多 4 个组专用服务请求输出信号以及最多 4 个共享服务请求输出信号。 2 个组专用和 2 个共享请求信号可发出中断请求 (见 **16-136 页的表 16-15 “XMC1300 中的数字连接 ”**)。

可为每个服务请求输出分配多个事件。服务请求可由 3 种类型的事件产生 :

- **请求源事件：**指示一个请求源已完成所请求转换序列，应用软件可启动进一步的操作。
  对于一个扫描请求源（组或后台请求源）而言，当所定义的整个通道组（挂起位）都转换完成时，会产生请求源事件。
  对于一个组队列请求源，根据编程情况，即当一个源中断被使能的通道转换完或遇到一个无效条目时，会产生请求源事件。

- **通道事件：**指示一次转换完成。另外，通道事件可被限制为在结果值位于一个设定范围内时产生。这就减轻了后台任务的 CPU 负荷，即仅在达到或超过指定的转换结果范围时服务请求才被激活。
- **结果事件：**指示在一个结果寄存器中有新的有效结果。一般来说，这会触发 CPU 的一次读操作。另外，如果数据缩减有效，结果事件只会以缩减后的速率产生。

每个 ADC 事件由一个专用的标志指示，该标志可由软件清除。如果为某一个事件使能了一个服务请求，则每当事件发生时就会产生服务请求，与对应的事件指示标志的状态无关。这就保证了 ADC 事件能在不清除标志位的情况下产生服务请求。

事件标志寄存器指示在 ADC 运行期间发生的所有类型的事件。软件可通过向寄存器 GxCEFLAG/GxRFLAG 中的相应位写 1 来置位每个标志，从而触发一个事件。软件可通过向寄存器 GxCEFCLR/GxREFCLR 中的相应位写 1 来清除每个标志。如果被使能，每当发生一次事件时都会产生服务请求，即使相关联的标志位保持置位状态。

节点指针寄存器

来自每个事件源的请求都可通过相关联的节点指针被定向到一组服务请求节点。来自多个源的请求可以被定向到同一节点；在这种情况下，这些请求相或后生成服务请求输出信号。

软件服务请求激活

可通过软件将寄存器 **GxSRACT (x = 0 - 1)** 中的相应位置 1 来激活每个服务请求。这可用于评估和测试目的。

*注： 对于共享服务请求线，见**表 16-12** 中的公共组。*

16.16 寄存器

多功能 ADC 由一系列的转换器块构建而成，这些块的控制方式完全相同。这就使得程序的通用性好，可伸缩性强。因此，相应的寄存器都被分配了单独的偏移量 (见**表 16-8**)。寄存器的准确地址通过将相应组的基地址（见**表 16-7**）与寄存器各自的偏移地址相加得到。

由于常规的组结构，每个组中有多个寄存器。还为每个通道提供了一些其他寄存器。这由寄存器一览表中的占位符指示：

- X###$_H$ 的含义：x × 0400$_H$ + 0###$_H$，对于 x = 0 - 1
- ###Y$_H$ 的含义：###0$_H$ + y × 0004$_H$，对于 y = 0 - N (取决于寄存器类型)
- Z###$_H$ 的含义：4803 4000$_H$ + 0###$_H$， (SHS0)

表 16-7 寄存器地址空间

模块	基地址	结束地址	备注
VADC0	4803 0000$_H$	4803 0BFF$_H$	

表 16-8 寄存器一览表

寄存器简称	寄存器全称	偏移地址	访问方式		页号
			读	写	
ID	模块标识寄存器	0008_H	U, PV	BE	16-60
CLC	时钟控制寄存器	0000_H	U, PV	PV	16-60
OCS	OCDS 控制和状态寄存器	0028_H	U, PV	PV	16-61
GLOBCFG	全局配置寄存器	0080_H	U, PV	U, PV	16-63
ACCPROT0	访问保护寄存器 0	0088_H	U, PV	U, PV 1)	16-67
ACCPROT1	访问保护寄存器 1	$008C_H$	U, PV	U, PV 1)	16-68
GxARBCFG	仲裁配置寄存器	$X480_H$	U, PV	U, PV	16-74
GxARBPR	仲裁优先级寄存器	$X484_H$	U, PV	U, PV	16-77
GxCHASS	通道分配寄存器，组 x	$X488_H$	U, PV	U, PV	16-70
GxRRASS	结果分配寄存器，组 x	$X48C_H$	U, SV	U, PV	16-71
GxQCTRL0	队列 0 源控制寄存器，组 x	$X500_H$	U, PV	U, PV	16-78
GxQMR0	队列 0 模式寄存器，组 x	$X504_H$	U, PV	U, PV	16-80
GxQSR0	队列 0 状态寄存器，组 x	$X508_H$	U, PV	U, PV	16-81
GxQINR0	队列 0 输入寄存器，组 x	$X510_H$	U, PV	U, PV	16-83
GxQ0R0	队列 0 寄存器 0，组 x	$X50C_H$	U, PV	U, PV	16-84
GxQBUR0	队列 0 备份寄存器，组 x	$X510_H$	U, PV	U, PV	16-85
GxASCTRL	自动扫描源控制寄存器，组 x	$X520_H$	U, PV	U, PV	16-86
GxASMR	自动扫描源模式寄存器，组 x	$X524_H$	U, PV	U, PV	16-88
GxASSEL	自动扫描源通道选择寄存器，组 x	$X528_H$	U, PV	U, PV	16-90
GxASPND	自动扫描源挂起寄存器，组 x	$X52C_H$	U, PV	U, PV	16-91
BRSCTRL	后台请求源控制寄存器	0200_H	U, PV	U, PV	16-92
BRSMR	后台请求源模式寄存器	0204_H	U, PV	U, PV	16-94
BRSSELx	后台请求源通道选择寄存器，组 x	$018Y_H$	U, PV	U, PV	16-96
BRSPNDx	后台请求源通道挂起寄存器，组 x	$01CY_H$	U, PV	U, PV	16-97
GxCHCTRy	通道 x 控制寄存器	$X60Y_H$	U, PV	U, PV	16-98
GxICLASS0	输入类寄存器 0，组 x	$X4A0_H$	U, PV	U, PV	16-100

表 16-8 寄存器一览表 (续表)

寄存器简称	寄存器全称	偏移地址	访问方式		页号
			读	写	
GxICLASS1	输入类寄存器 1，组 x	$X4A4_H$	U, PV	U, PV	16-100
GLOBICLASS0	输入类寄存器 0，全局	$00A0_H$	U, PV	U, PV	16-100
GLOBICLASS1	输入类寄存器 1，全局	$00A4_H$	U, PV	U, PV	16-100
GxALIAS	别名寄存器	$X4B0_H$	U, PV	U, PV	16-113
GxBOUND	边界选择寄存器，组 x	$X4B8_H$	U, PV	U, PV	16-114
GLOBBOUND	全局边界选择寄存器	$00B8_H$	U, PV	U, PV	16-114
GxBFL	边界标志寄存器，组 x	$X4C8_H$	U, PV	U, PV	16-115
GxBFLS	边界标志软件寄存器，组 x	$X4CC_H$	U, PV	U, PV	16-116
GxBFLC	边界标志控制寄存器，组 x	$X4D0_H$	U, PV	U, PV	16-117
GxBFLNP	边界标志结点指针寄存器，组 x	$X4D4_H$	U, PV	U, PV	16-118
GxRCRy	组 x 结果控制寄存器 y	$X68Y_H$	U, PV	U, PV	16-102
GxRESy	组 x 结果寄存器 y	$X70Y_H$	U, PV	U, PV	16-104
GxRESDy	组 x 结果寄存器 y (调试模式)	$X78Y_H$	U, PV	U, PV	16-105
GLOBRCR	全局结果控制寄存器	0280_H	U, PV	U, PV	16-107
GLOBRES	全局结果寄存器	0300_H	U, PV	U, PV	16-108
GLOBRESD	全局结果寄存器 （调试模式）	0380_H	U, PV	U, PV	16-108

表 16-8 寄存器一览表 (续表)

寄存器简称	寄存器全称	偏移地址	访问方式		页号
			读	写	
GxVFR	有效标志寄存器，组 x	$X5F8_H$	U, PV	U, PV	16-109
GxSYNCTR	同步控制寄存器	$X4C0_H$	U, PV	U, PV	16-119
GxEMUXCTR	外部多路复用器控制寄存器，组 x	$X5F0_H$	U, PV	U, PV	16-120
EMUXSEL	外部多路复用器选择寄存器	$03F0_H$	U, PV	U, PV	16-122
GxSEFLAG	源事件标志寄存器，组 x	$X588_H$	U, PV	U, PV	16-124
GxCEFLAG	通道事件标志寄存器，组 x	$X580_H$	U, PV	U, PV	16-124
GxREFLAG	结果事件标志寄存器，组 x	$X584_H$	U, PV	U, PV	16-125
GxSEFCLR	源事件标志清零寄存器，组 x	$X598_H$	U, PV	U, PV	16-126
GxCEFCLR	通道事件标志清零寄存器，组 x	$X590_H$	U, PV	U, PV	16-126
GxREFCLR	结果事件标志清零寄存器，组 x	$X594_H$	U, PV	U, PV	16-127
GLOBEFLAG	全局事件标志寄存器	$00E0_H$	U, PV	U, PV	16-127
GxSEVNP	源事件节点指针寄存器，组 x	$X5C0_H$	U, PV	U, PV	16-128
GxCEVNP0	通道事件节点指针寄存器 0，组 x	$X5A0_H$	U, PV	U, PV	16-129
GxREVNP0	结果事件节点指针寄存器 0，组 x	$X5B0_H$	U, PV	U, PV	16-130
GxREVNP1	结果事件节点指针寄存器 1，组 x	$X5B4_H$	U, PV	U, PV	16-131
GLOBEVNP	全局事件节点指针寄存器	0140_H	U, PV	U, PV	16-131

表 16-8 寄存器一览表 (续表)

寄存器简称	寄存器全称	偏移地址	访问方式		页号
			读	写	
GxSRACT	服务请求软件激活触发寄存器，组 x	$X5C8_H$	U, PV	U, PV	16-132
ID	SHS 模块标识寄存器	$Z008_H$	U, SV	U, SV	16-60
SHSCFG	SHS 配置寄存器	$Z040_H$	U, SV	U, SV	16-64
CALCTR	SHS 校准控制寄存器	$Z0BC_H$	U, SV	U, SV	16-110
CALGC0	SHS 增益校准控制寄存器	$Z0C0_H$	U, SV	U, SV	16-111
CALGC1	SHS 增益校准控制寄存器	$Z0C4_H$	U, SV	U, SV	16-111
GNCTR00	增益控制寄存器 0，采样保持单元 0	$Z180_H$	U, SV	U, SV	16-112
GNCTR10	增益控制寄存器 0，采样保持单元 1	$Z190_H$	U, SV	U, SV	16-112
STEPCFG	SHS 步进器配置寄存器	$Z044_H$	U, SV	U, SV	16-71
TIMCFG0	SHS 时序配置寄存器	$Z080_H$	U, SV	U, SV	16-72
TIMCFG1	SHS 时序配置寄存器	$Z084_H$	U, SV	U, SV	16-72
LOOP	SHS SD 循环控制寄存器	$Z050_H$	U, SV	U, SV	16-123

1) 访问这些寄存器受 SCU 一章所描述的位保护机制的保护。

16.16.1 模块标识

模块标识寄存器指示 XMC1300 中使用的 ADC 模块的版本。

ID
模块标识寄存器 **(0008_H)** **复位值 : 00C5 C0XX$_H$**
SHS0_ID
模块标识寄存器 **($4803\ 4008_H$)** **复位值 : 0099 C0XX$_H$**

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
MOD_NUMBER																MOD_TYPE								MOD_REV							
r																r								r							

域	位	类型	描述
MOD_REV	[7:0]	r	**模块版本** **指示实现的版本号。该信息取决于设计阶段。**
MOD_TYPE	[15:8]	r	**模块类型** 该内部标记被固定为 $C0_H$。
MOD_NUMBER	[31:16]	r	**模块号** 指示模块的标识号 ($00C5_H$ = SARADC, 0099_H = SHS)。

16.16.2 系统寄存器

一组标准化的寄存器，提供对模块的一般访问并控制基本的系统功能。
时钟控制寄存器 **CLC** 允许编程者调整模块的功能和功耗来适应应用需求。寄存器 **CLC** 控制模块时钟信号和对休眠模式信号的反应。

CLC
时钟控制寄存器 **(0000_H)** **复位值 : 0000 0003$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**EDIS**	**0**	**DISS**	**DISR**
r	r	r	r	r	r	r	r	r	r	r	r	rw	r	r	rw

域	位	类型	描述
DISR	0	rw	**模块禁止请求位** 用于使能 / 禁止模块的控制。还可通过清零 ANONS 来禁止模拟部分。 0_B 开请求：使能模块时钟 1_B 关请求：禁止模块时钟
DISS	1	r	**模块禁止状态位** 0_B 开：模块时钟被使能 1_B 关：模块时钟被禁止
0	2	r	**保留 , 读出值为 0，应写入 0**
EDIS	3	rw	**休眠模式使能控制** 用来控制模块对休眠模式的反应。 0_B 休眠模式被使能并工作 1_B 模块忽略休眠模式控制信号
0	[31:4]	r	**保留，读出值为 0，应写入 0**

OCDS 控制和状态寄存器 OCS 控制模块在挂起模式的行为（用于调试）并且包含 OCDS 触发总线的模块相关控制位。

OCDS 控制和状态 (OCS) 寄存器由调试复位清零。

只有在 OCDS 被使能时，才能写 OCS 寄存器。

如果 OCDS 被禁止， OCS 寄存器的值不会改变。

当 OCDS 被禁止时， OCS 挂起控制无效。

写访问只能是 32 位宽，并且需要管理员模式。

OCS

OCDS 控制和状态寄存器 (0028_H) 复位值 : $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**SUS STA**	**SUS _P**	**SUS**				**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	rh	w	rw				r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**TG _P**	**TGB**	**TGS**	
r	r	r	r	r	r	r	r	r	r	r	r	w	rw	rw	

域	位	类型	描述
TGS	[1:0]	rw	**OTGB0/1 的触发设置** 00_B 无触发设置输出 01_B 触发设置 1：TS16_SSIG，输入采样信号 10_B 保留 11_B 保留
TGB	2	rw	**OTGB0/1 总线选择** 0_B 触发设置在 OTGB0 输出 1_B 触发设置在 OTGB1 输出
TG_P	3	w	**TGS、 TGB 写保护** TGS 和 TGB 只能在 TG_P 等于 1 时方可写入，否则不变。读出值为 0。
0	[23:4]	r	**保留，读出值为 0，应写入 0**
SUS	[27:24]	rw	**OCDS 挂起控制** 控制来自 OCDS 触发器开关 （OTGS）的挂起信号的敏感性 0000_B 不会挂起 0001_B 硬挂起：时钟被立即关闭。 0010_B 软挂起模式 0： 当前转换完成并且结果保存之后停止转换。 不会为仲裁器改变 0011_B 软挂起模式 1： 当前转换完成并且结果保存之后停止转换。 在当前仲裁周期之后停止仲裁器。 其他：保留
SUS_P	28	w	**SUS 写保护** 只有在 SUS_P 等于 1 时， SUS 方可被写入，否则不变。读出值为 0。
SUSSTA	29	rh	**挂起状态** 0_B 模块 （尚）未被挂起 1_B 模块被挂起
0	[31:30]	r	**保留，读出值为 0，应写入 0**

表 16-9 VADC 的 TS16_SSIG 触发设置

位	名称	描述
[1:0]	GxSAMPLE	转换器组 x(x = 1-0) 的输入信号采样阶段
[15:2]	0	保留

16.16.3 通用寄存器

全局配置寄存器提供全局控制和配置选项，这些选项对一个集群中的所有转换器都有效。

GLOBCFG
全局配置寄存器 **(0080_H)** **复位值 : 0000 000F$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUCAL	0	0	0	0	0	0	0	0	0	0	0	0	0	DPCAL1	DPCAL0
w	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIVWC	0	0	0	0	0	DIVD		DCMSB	0	0	DIVA				
w	r	r	r	r	r	rw		rw	r	r	rw				

域	位	类型	描述
DIVA	[4:0]	rw	**模拟内部时钟的分频系数** 定义基本转换器时钟的频率 f_{ADCI} (转换和采样阶段的基本时钟)。 00_H $f_{ADCI} = f_{ADC}$ 01_H $f_{ADCI} = f_{ADC} / 2$ 02_H $f_{ADCI} = f_{ADC} / 3$ ... $1F_H$ $f_{ADCI} = f_{ADC} / 32$
0	[6:5]	r	**保留，读出值为 0，应写入 0**
DCMSB	7	rw	**用于 MSB 转换的双时钟** 为 MSB[1)] 转换步骤选择一个额外的时钟周期 0_B MSB 用 1 个时钟周期 （标准） 1_B MSB 用 2 个时钟周期 (f_{ADCI} > 20 MHz)$_B$

域	位	类型	描述
DIVD	[9:8]	rw	**仲裁器时钟的分频系数** 定义仲裁器的时钟频率 f_{ADCD}。 00_B $f_{ADCD} = f_{ADC}$ 01_B $f_{ADCD} = f_{ADC} / 2$ 10_B $f_{ADCD} = f_{ADC} / 3$ 11_B $f_{ADCD} = f_{ADC} / 4$
0	[14:10]	r	**保留，读出值为 0，应写入 0**
DIVWC	15	w	**分频器参数的写控制** 0_B 不能对分频器参数进行写访问 1_B 位域 DIVA、 DCMSB、 DIVD 可被写入
DPCALx **(x = 0 - 1)**	x+16	rw	**禁止后校准** 0_B 组 x 的每次转换后自动进行后校准 1_B 无后校准 *注： 对于给定的产品型号，该位只对校准过的转换器有效。*
0	[30:18]	r	**保留，读出值为 0，应写入 0**
SUCAL	31	w	**初始校准** 位 SUCAL 的 0-1 跳变启动对所有校准过的模拟转换器的初始校准阶段。 0_B 无操作 1_B 启动初始校准阶段 (由位 GxARBCFG.CAL 指示)

1) 请参见 **16-33 页 " 转换时序 "** 一节。

SHS0_SHSCFG
SHS 配置寄存器 **(4803 4040$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
STATE				TC				0	0	0	0	0	0	SP1	SP0
rh				rw				r	r	r	r	r	r	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SCWC	ANRDY	0	ANOFF	AREF		0	0	0	0	0	0	DIVS			
w	rh	r	rw	rw		r	r	r	r	r	r	rw			

域	位	类型	描述
DIVS	[3:0]	rw	**SHS 时钟的分频系数** 为 SHS 逻辑和 SAR 转换器定义时钟频率。 0000_B $f_{SH} = f_{CONV} / 1$ 0001_B $f_{SH} = f_{CONV} / 2$ ... 1111_B $f_{SH} = f_{CONV} / 16$
0	[9:4]	r	**保留，读出值为 0，应写入 0**
AREF	[11:10]	rw	**模拟参考电压选择** 00_B 外部参考，电源电压范围 01_B 保留 10_B 保留 11_B 保留
ANOFF	12	rw	**模拟转换器掉电强制**[1)] 0_B 转换器由位域 ANONS 控制 (数字控制块) 1_B 转换器被永久性关闭
0	13	r	**保留，读出值为 0，应写入 0**
ANRDY	14	rh	**模拟转换器就绪** 0_B 模拟转换器处于掉电模式 1_B 转换器可以工作
SCWC	15	w	**SHS 配置的写控制** 0_B 不能对 SHS 配置进行写访问 1_B 位域 ANOFF、 AREF、 DIVS 可被写入
SP0, **SP1**	16, 17	rh	**采样保持单元 x 的采样挂起** 0_B 无采样挂起 1_B 采样保持单元 x 已结束采样阶段
0	[23:18]	r	**保留，读出值为 0，应写入 0**
TC	[27:24]	rw	**测试控制** 1011_B 使能内部测试功能 其他：正常工作 *注： 对于正常，确保向位域写 0000_B。*

域	位	类型	描述
STATE	[31:28]	rh	**定序器的当前状态** 0000_B 空闲 0001_B 偏移校准有效 0010_B 增益校准有效 0011_B 初始校准有效 1000_B 采样保持单元 0 的步处理有效 1001_B 采样保持单元 1 的步处理有效 其他：正常操作

1) 见 16-13 页 “ 模拟转换器控制 ”。

寄存器访问控制

VADC 提供了多种保护机制，以防止对 VADC 控制位域的意外写访问。

ACCPROT0
访问保护寄存器 **(0088_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
APGC	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**API1**	**API0**
rw	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APEM	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**APC1**	**APC0**
rw	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw

域	位	类型	描述
APC0, APC1	0, 1	rw	**通道控制的访问保护，组 0 - 1** 0_B 允许对寄存器的所有访问 1_B 阻塞对通道控制寄存器的写访问
0	[14:2]	r	**保留，读出值为 0，应写入 0**
APEM	15	rw	**外部多路复用器的访问保护** 0_B 允许对寄存器的所有访问 1_B 阻塞对外部多路复用器寄存器的写访问
API0, API1	16, 17	rw	**初始化的访问保护，组 0 - 1** 0_B 允许对寄存器的所有访问 1_B 阻塞对初始化寄存器的写访问
0	[30:18]	r	**保留，读出值为 0，应写入 0**
APGC	31	rw	**全局配置的访问保护** 0_B 允许对寄存器的所有访问 1_B 阻塞对全局配置寄存器的写访问

ACCPROT1
访问保护寄存器 **(008C$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**APR1**	**APR0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
APTF	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**APS1**	**APS0**
rw	r	r	r	r	r	r	r	r	r	r	r	r	r	rw	rw

域	位	类型	描述
APS0, APS1	0, 1	rw	**服务请求的访问保护，组 0 - 1** 0_B 允许对寄存器的所有访问 1_B 阻塞对服务请求寄存器的写访问
0	[14:2]	r	**保留，读出值为 0，应写入 0**
APTF	15	rw	**测试功能的访问保护** 0_B 允许对寄存器的所有访问 1_B 阻塞对测试功能寄存器的写访问
APR0, APR1	16, 17	rw	**结果寄存器的访问保护，组 0 - 1** 0_B 允许对寄存器的所有访问 1_B 阻塞对结果寄存器的写访问
0	[31:18]	r	**保留，读出值为 0，应写入 0**

表 16-10 寄存器保护组

控制位	寄存器	备注
APCx	GxCHCTR0 … GxCHCTRy	通道控制
APEM	EMUXSEL, GxEMUXCTR	外部多路复用器控制
APIx	GxARBCFG, GxARBPR, GxCHASS, GxRRASS, GxICLASS0/1, GxSYNCTR	初始化
APGC	GLOBCFG	全局配置

表 16-10 寄存器保护组 (续表)

控制位	寄存器	备注
APSx	GxSEFLAG, GxSEVNP, GxCEFLAG, GxCEVNP0/1, GxREFLAG, GxREVNP0/1, GxSRACT	服务请求控制
APTF	-	测试功能
APRx	GxRCR0 … GxRCR15, GxBOUND, GxRES0 … GxRES15	结果控制

优先通道分配

一个组内的每个通道都可被分配给该组的请求源，然后被视为一个优先通道。一个已分配的优先通道只能由本组内的请求源进行转换。未被分配的通道也可由后台请求源转换。

GxCHASS (x = 0 - 1)
通道分配寄存器，组 x **($x * 0400_H + 0488_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**ASSCH7**	**ASSCH6**	**ASSCH5**	**ASSCH4**	**ASSCH3**	**ASSCH2**	**ASSCH1**	**ASSCH0**
r	r	r	r	r	r	r	r	rw	rw	rw	rw	rw	rw	rw	rw

域	位	类型	描述
ASSCHy (y = 0 - 7)	y	rw	**通道 y 分配** 0_B 通道 y 可以是具有最低优先级的后台通道 1_B 通道 y 是组 x 内的一个优先通道
0	[31:8]	r	**保留，读出值为 0，应写入 0**

优先级结果寄存器分配

一个组内的每个结果寄存器都可被分配给该组的请求源，然后被视为一个保留的资源。一个已分配的结果寄存器只能由本组内的请求源写入。未分配的结果寄存器也可由后台请求源写入。

GxRRASS (x = 0 - 1)
结果分配寄存器，组 x **(x * 0400_H + $048C_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ASSRR 15	ASSRR 14	ASSRR 13	ASSRR 12	ASSRR 11	ASSRR 10	ASSRR 9	ASSRR 8	ASSRR 7	ASSRR 6	ASSRR 5	ASSRR 4	ASSRR 3	ASSRR 2	ASSRR 1	ASSRR 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

域	位	类型	描述
ASSRRy (y = 0 - 15)	y	rw	**结果寄存器 y 分配** 0_B 结果寄存器 y 也可由后台请求源写入 1_B 结果寄存器 y 只能由组 x 内的源写入
0	[31:16]	r	**保留，读出值为 0，应写入 0**

步进器配置寄存器选择在步进状态机的每个步期间得到服务的采样保持单元。

SHS0_STEPCFG
步进器配置寄存器 **($4803\ 4044_H$)** **复位值 : $0000\ 0098_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SEN 7	KSEL7			SEN 6	KSEL6			SEN 5	KSEL5			SEN 4	KSEL4		
rw	rw			rw	rw			rw	rw			rw	rw		

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEN 3	KSEL3			SEN 2	KSEL2			SEN 1	KSEL1			SEN 0	KSEL0		
rw	rw			rw	rw			rw	rw			rw	rw		

域	位	类型	描述
KSEL0, **KSEL1,** **KSEL2,** **KSEL3,** **KSEL4,** **KSEL5,** **KSEL6,** **KSEL7**	[2:0], [6:4], [10:8], [14:12], [18:16], [22:20], [26:24], [30:28]	rw	**内核选择** 定义在该步期间被服务的组 （即采样保持单元）。 有效数值取决于所支持的内核 / 接口数量：0 … 1
SEN0, **SEN1,** **SEN2,** **SEN3,** **SEN4,** **SEN5,** **SEN6,** **SEN7**	3, 7, 11, 15, 19, 23, 27, 31	rw	**步 x 使能** 确定是否执行该步。 0_B 关闭： 该步不是步进序列的一部分 1_B 打开： 在序列期间执行该步

时序配置寄存器 TIMCFGx 配置 SHS 的接口时序。

SHS0_TIMCFGx (x = 0 - 1)
时序配置寄存器 x **(4803 4080$_H$ + x * 4$_H$)** **复位值 : 0000 0001$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**TGEN**													
r	r	rh													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**SST**						**FCRT**				**0**	**0**	**0**	**AT**
r	r	rw						rw				r	r	r	rw

域	位	类型	描述
AT	0	rw	**加速时序** 使能增强的转换时序机制。 0_B 兼容的时序： 标准转换时间后结果才可用 1_B 加速时序： 一旦转换结束结果立即可用
0	[3:1]	r	**保留，读出值为 0，应写入 0**
FCRT	[7:4]	rw	**快速比较模式响应时间** 定义一次快速比较操作的持续时间。结果在经过时间 (FCRT+1) × 2 × t_{ADCI} 之后产生。 0_H 在 t_{ADCI} × 2 之后产生结果 ... F_H 在 t_{ADCI} × 32 之后产生结果
SST	[13:8]	rw	**短采样时间** 定义采样阶段的持续时间。 00_H 兼容时序： 采样时间由 DIVA 和 STC 定义。 01_H 采样时间为 t_{ADC} × 1 $3F_H$ 采样时间为 t_{ADC} × 63
0	[15:14]	r	**保留，读出值为 0，应写入 0**
TGEN	[29:16]	rh	**时序发生器** 对采样阶段的持续时间和兼容时序模式下转换的持续时间计数。
0	[31:30]	r	**保留，读出值为 0，应写入 0**

16.16.4 仲裁和源寄存器

仲裁配置寄存器选择仲裁器的时序和行为。

GxARBCFG (x = 0 - 1)
仲裁配置寄存器，组 x **(x * 0400_H + 0480_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SAMPLE	BUSY	CALS	CAL	0	0	SYNRUN	CHNR					CSRC		ANONS	
rh	rh	rh	rh	r	r	rh	rh					rh		rh	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	ARBM	0	ARBRND		0	0	ANONC	
r	r	r	r	r	r	r	r	rw	r	rw		r	r	rw	

域	位	类型	描述
ANONC	[1:0]	rw	**模拟转换器控制** 定义一个独立转换器或主模式转换器中的位域 ANONS 的值。 编码见 ANONS 或 16.5 节。
0	[3:2]	r	**保留，读出值为 0，应写入 0**
ARBRND	[5:4]	rw	**仲裁周期长度** 定义每个仲裁周期内仲裁时隙的数量 (仲裁周期长度 = t_{ARB})。[1)] 00_B 每个仲裁周期内有 4 个仲裁时隙 (t_{ARB} = 4 / f_{ADCD}) 01_B 每个仲裁周期内有 8 个仲裁时隙 (t_{ARB} = 8 / f_{ADCD}) 10_B 每个仲裁周期内有 16 个仲裁时隙 (t_{ARB} = 16 / f_{ADCD}) 11_B 每个仲裁周期内有 20 个仲裁时隙 (t_{ARB} = 20 / f_{ADCD})
0	6	r	**保留，读出值为 0，应写入 0**

域	位	类型	描述
ARBM	7	rw	**仲裁模式** 0_B 仲裁器一直运行。同步从内核 （见 **16.12.1 节**）和使用信号 ARBCNT 的等距抽样 （见 **16.12.2 节**）需要该设置。 1_B 仲裁器只在至少有一个已使能的使能请求源的转换请求被挂起时才运行。如果转换器空闲，该设置能够确保从进入请求到转换启动之间有可重复的延迟。不支持同步转换。
0	[15:8]	r	**保留，读出值为 0，应写入 0**
ANONS	[17:16]	rh	**模拟转换器控制状态** 由一个独立内核或主模式内核中的位域 ANONC 定义。 在从模式，该位域由相应主内核的位域 ANONC 定义。 见 **16.5 节** 00_B 模拟转换器关闭 01_B 保留 10_B 慢速待机模式 11_B 正常工作 (永久性使能)
CSRC	[19:18]	rh	**当前转换请求源** 指示当前 (BUSY = 1) 或最后一次 (BUSY = 0) 转换的仲裁时隙号。 当一次转换启动时，该位域被更新。 00_B 请求源 0 的当前 / 最后一次转换 01_B 请求源 1 的当前 / 最后一次转换 10_B 后台源源的当前 / 最后一次转换 11_B 同步源请求的当前 / 最后一次转换 (从转换器) 其他组合保留。
CHNR	[24:20]	rh	**通道号** 指示当前或最后一次转换的模拟输入通道 当一次转换启动时，该位域被更新。
SYNRUN	25	rh	**同步转换运行** 指示一次同步 （= 并行）转换当前正在运行。 0_B 正常转换或没有转换在运行 1_B 一次同步转换正在运行 (不能被具有更高优先级的请求撤消！)

域	位	类型	描述
0	[27:26]	r	**保留，读出值为 0，应写入 0**
CAL	28	rh	**初始校准活动指示** 指示相应模拟转换器的初始校准阶段： 0_B 已完成或尚未启动 1_B 初始校准阶段正在进行 (在 SUCAL 被置 1 一个时钟周期置位) *注： 在初始校准阶段完成后才能启动转换。*[2)]
CALS	29	rh	**初始校准开始** 指示启动校准已经开始 0_B 已请求但尚未开始 1_B 初始校准已经开始 *注： 当置位SUCAL 时，位CALS 被清零，与位CAL 一起被置位。*
BUSY	30	rh	**转换器忙标志** 0_B 不忙 1_B 转换器正忙于转换
SAMPLE	31	rh	**采样阶段标志** 0_B 正在转换或空闲 1_B 输入信号当前正被采样

1) 默认设置 4 个仲裁时隙的默认设置对于正确的仲裁是足够的。如果需要对请求进行同步，可以增加一个仲裁周期的持续时间。
2) 初始校准的完成由 CAL = 0 和 CALS = 1 指示。
CAL = CALS = 0 指示初始校准已被请求但尚未开始。
CAL = CALS = 1 指示正在进行校准阶段。

仲裁优先级寄存器定义请求源的优先级和每个请求源的转换启动模式。

注： 只能在一个请求源被禁止并且由该源请求的当前运行转换结束时改变该请求源的优先级和转换启动模式。

GxARBPR (x = 0 - 1)
仲裁优先级寄存器，组 x **(x * 0400_H + 0484_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	ASEN2	ASEN1	ASEN0	0	0	0	0	0	0	0	0
r	r	r	r	r	rw	rw	rw	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	CSM2	0	PRIO2		CSM1	0	PRIO1		CSM0	0	PRIO0	
r	r	r	r	rw	r	rw		rw	r	rw		rw	r	rw	

域	位	类型	描述
PRIO0, PRIO1, PRIO2	[1:0], [5:4], [9:8]	rw	**请求源 x 的优先级** 请求源 x 的仲裁优先级 (在时隙 x 内) 00_B 选择最低优先级。 ... 11_B 选择最高优先级。
CSM0, CSM1, CSM2	3, 7, 11	rw	**请求源 x 的转换启动模式** 0_B 待读模式 1_B 撤消 - 插入 - 重复模式，即该源可以撤消其他源的转换。
0	2, 6, 10, [23:12]	r	**保留，读出值为 0，应写入 0**
ASENy (y = 0 - 2)	24 + y	rw	**仲裁时隙 y 使能** 使能一个仲裁周期内相关联的仲裁时隙。请求源位不会被对 ASENR 的写操作修改。 0_B 相应的仲裁时隙被禁止并被认为是空的。忽略来自相关的转换源的挂起转换请求。 1_B 相应的仲裁时隙被使能。仲裁器对来自相关联请求源的挂起转换请求进行仲裁。
0	[31:27]	r	**保留，读出值为 0，应写入 0**

队列源的控制寄存器选择外部门控和 / 或触发信号。
写控制位允许用一次简单的写访问对每个功能单独控制。

GxQCTRL0 (x = 0 - 1)
队列 0 源控制寄存器，组 x　　($x * 0400_H + 0500_H$)　　复位值 : $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TM WC	0	0	TM EN	0	0	0	0	GT WC	0	0	GT LVL	GT SEL			
w	r	r	rw	r	r	r	r	w	r	r	rh	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XT WC	XT MODE		XT LVL	XT SEL				0	0	0	0	SRCRESREG			
w	rw		rh	rw				r	r	r	r	rwr			

域	位	类型	描述
SRCRESREG	[3:0]	rw	**源专用结果寄存器** 0000_B使用 GxCHCTRy.RESREG 来选择一个组结果寄存器 0001_B保存结果到组结果寄存器 GxRES1 … 1111_B保存结果到组结果寄存器 GxRES15
0	[7:4]	r	**保留，读出值为 0，应写入 0**
XTSEL	[11:8]	rw	**外部触发输入选择** 连接的触发输入信号见 **16-136 页的表 16-15 “XMC1300 中的数字连接 ”** *注：　$XTSEL = 1111_B$ 使用所选择的门控输入作为触发源 (ENGT 必须为 $0x_B$)。*
XTLVL	12	rh	**外部触发信号电平** 所选触发输入的当前电平
XTMODE	[14:13]	rw	**触发操作模式** 00_B　无外部触发 01_B　在下降沿产生触发事件 10_B　在上升沿产生触发事件 11_B　在任何一个边沿产生触发事件
XTWC	15	w	**触发配置的写控制** 0_B　不能写触发配置 1_B　可以写位域 XTMODE 和 XTSEL

域	位	类型	描述
GTSEL	[19:16]	rw	**门控输入选择** 连接的门控输入信号见 **16-136 页的表 16-15 “XMC1300 中的数字连接 ”**
GTLVL	20	rh	**门控输入电平** 所选门控输入的当前电平
0	[22:21]	r	**保留，读出值为 0，应写入 0**
GTWC	23	w	**门控配置的写控制** 0_B 不能写门控配置 1_B 可以写位域 GTSEL
0	[27:24]	r	**保留，读出值为 0，应写入 0**
TMEN	28	rw	**定时器模式使能** 0_B 无定时器模式： 可使用标准门控机制 1_B 使能等距采样定时器模式 必须禁止标准门控机制
0	[30:29]	r	**保留，读出值为 0，应写入 0**
TMWC	31	w	**定时器模式的写控制** 0_B 不能写定时器模式 1_B 可以写位域 TMEN

队列模式寄存器配置一个队列请求源的工作模式。

GxQMR0 (x = 0 - 1)
队列 0 模式寄存器，组 x **(x * 0400_H + 0504_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RPT DIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	CEV	FLU SH	TR EV	CLR V	0	0	0	0	0	EN TR	ENGT	
r	r	r	r	w	w	w	w	r	r	r	r	r	rw	rw	

域	位	类型	描述
ENGT	[1:0]	rw	**使能门控** 为源 0/2 选择门控功能。 00_B 不发出转换请求 01_B 如果在队列 0 寄存器或备份寄存器中有被挂起的有效转换请求，发出转换请求 10_B 如果在队列 0 寄存器或备份寄存器中有被挂起的有效转换请求并且 REQGTx = 1，发出转换请求 11_B 如果在队列 0 寄存器或备份寄存器中有被挂起的有效转换请求并且 REQGTx = 0，发出转换请求。 *注： REQGTx 是所选择的门控信号。*
ENTR	2	rw	**使能外部触发** 0_B 禁止外部触发 1_B 在所选触发输入信号 REQTR 的所选边沿产生触发事件。
0	[7:3]	r	**保留，读出值为 0，应写入 0**
CLRV	8	w	**清除有效位** 0_B 无操作 1_B 序列中的下一个挂起有效队列条目和事件标志 EV 被清零。 如果在队列的备份寄存器中有一个有效条目 (QBUR.V = 1)，则该条目被清零，否则在队列寄存器 0 中的条目被清零。

域	位	类型	描述
TREV	9	w	**触发事件** 0_B 无操作 1_B 用软件产生一个触发事件
FLUSH	10	w	**清空队列** 0_B 无操作 1_B 清除所有队列条目 (包括备份级) 和事件标志 EV。队列不再包含有效条目。
CEV	11	w	**清除事件标志** 0_B 无操作 1_B 将位 EV 清零
0	[15:12]	r	**保留，读出值为 0，应写入 0**
RPTDIS	16	rw	**重复禁止** 0_B 重复被撤消的转换 1_B 忽略被撤消的转换
0	[31:17]	r	**保留，读出值为 0，应写入 0**

队列状态寄存器指示队列源的当前状态。队列填充程度和空信息可通过检查队列中间级（如果可用）和队列寄存器 0 得知。保存在备份级的一次被中止的转换不由这些位指示（因此，请见 QBURx.V）。

GxQSR0 (x = 0 - 1)
队列 0 状态寄存器，组 x **(x * 0400_H + 0508_H)** **复位值 : $0000\ 0020_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**EV**	**REQ GT**	**0**	**EMP TY**	**0**	**FILL**			
r	r	r	r	r	r	r	rh	rh	r	rh	r	rh			

域	位	类型	描述
FILL	[3:0]	rh	**队列 2 的填充水平** 指示有效队列条目的数量。 每当一个新的条目被写到 QINRx 或者被一个使能的重填机制写入时，该值加 1。 每当一个请求的转换已经启动时，该值减 1。如果填充水平已达到其最大值，则新条目被忽略。 0000_B队列中有 1 个 （如果 EMPTY = 0）或没有 （如果 EMPTY = 1）有效条目 0001_B队列中有 2 个有效条目 0010_B队列中有 3 个有效条目 … 0111_B队列中有 8 个有效条目 其他：保留
0	4	r	**保留，读出值为 0，应写入 0**
EMPTY	5	rh	**队列空** 0_B 队列中有有效条目 (见 FILL) 1_B 无有效条目 (队列为空)
0	6	r	**保留，读出值为 0，应写入 0**
REQGT	7	rh	**请求门控电平** 监视所选 REQGT 输入的电平。 0_B 门控输入为低电平 1_B 门控输入为高电平
EV	8	rh	**事件检测** 当队列中 (队列寄存器 0 或备份级) 至少有一个有效条目时，该位指示已检测到一个事件。一旦置位，该位在所请求的转换启动时被自动清零。 0_B 无触发事件 1_B 检测到一个触发事件
0	[31:9]	r	**保留，读出值为 0，应写入 0**

队列输入寄存器是一个队列请求源的转换请求的入口点。

GxQINR0 (x = 0 - 1)
队列 0 输入寄存器，组 x **($x * 0400_H + 0510_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**EXTR**	**ENSI**	**RF**	**REQCHNR**				
r	r	r	r	r	r	r	r	w	w	w	w				

域	位	类型	描述
REQCHNR	[4:0]	w	**请求通道号** 定义要被转换的通道号
RF	5	w	**重填** 0_B 不重填：该队列条目被转换一次后即失效 1_B 自动重填：当相关的转换启动时，该队列条目被自动重新加载到 QINRx。
ENSI	6	w	**使能源中断** 0_B 无请求源中断 1_B 在发生请求源事件时 （相关转换完成）产生请求源事件中断。
EXTR	7	w	**外部触发** 使能外部触发功能。 0_B 一个有效队列条目会立即导致产生转换请求。 1_B 在发出转换请求之前，有效队列条目等待一个触发事件发生。
0	[31:8]	r	**保留，读出值为 0，应写入 0**

注： 寄存器 QINRx 与寄存器 QBURx 共享地址。
写操作的目标是寄存器 QINRx 中的控制位；读操作返回寄存器 QBURx 中的状态位。

队列寄存器 0 监视挂起请求的状态 （队列级 0）。

GxQ0R0 (x = 0 - 1)
队列 0 寄存器 0，组 x **($x * 0400_H + 050C_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**V**	**EXTR**	**ENSI**	**RF**	**REQCHNR**				
r	r	r	r	r	r	r	rh	rh	rh	rh	rh				

域	位	类型	描述
REQCHNR	[4:0]	rh	**请求通道号** 保存将被转换的通道号。
RF	5	rh	**重填** 选择对已经处理过的请求的处理。 0_B 转换启动后，请求被丢弃。 1_B 转换启动后，请求被自动重填到队列中。
ENSI	6	rh	**使能源中断** 0_B 无请求源中断 1_B 在发生一个请求源事件时 (相关转换完成) 产生请求源事件中断
EXTR	7	rh	**外部触发** 使能外部触发事件。 0_B 一个有效队列条目会立即导致产生转换请求 1_B 请求处理器等待一个触发事件
V	8	rh	**请求通道号有效** 指示寄存器 0 中有一个有效队列条目。 0_B 无有效队列条目 1_B 队列条目有效并导致产生一次转换请求
0	[31:9]	r	**保留，读出值为 0，应写入 0**

队列备份寄存器监视被中止的队列请求的状态。

GxQBUR0 (x = 0 - 1)
队列 0 备份寄存器，组 x **($x * 0400_H + 0510_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**V**	**EXTR**	**ENSI**	**RF**	**REQCHNR**				
r	r	r	r	r	r	r	rh	rh	rh	rh	rh				

域	位	类型	描述
REQCHNR	[4:0]	rh	**请求通道号** 由该请求源请求的但已被中止的转换的通道号
RF	5	rh	**重填** 被中止的转换的重填控制位
ENSI	6	rh	**使能源中断** 被中止的转换的使能源中断控制位
EXTR	7	rh	**外部触发** 被中止的转换的外部触发控制位
V	8	rh	**请求通道号有效** 指示在队列备份寄存器中的条目 (REQCHNR、 RF、 TR、 ENSI) 是否有效。 当一个运行的转换 （已被该请求源请求）被中止时，位 V 被置位；当被中止的转换重新启动时，该位被清零。 0_B 备份寄存器无效 1_B 备份寄存器包含一个有效条目。在队列寄存器 0 （级 0）中的一个有效条目被请求之前，备份寄存器中的条目被请求。
0	[31:9]	r	**保留，读出值为 0，应写入 0**

注： 寄存器 QBURx 与寄存器 QINRx 共享地址。
读操作从寄存器 QBURx 返回状态位。写操作的目标是寄存器 QINRx 中的控制位。

组扫描源寄存器

每组扫描源都有一个独立的寄存器组。这些源可以独立工作。

自动扫描源的控制寄存器选择外部门控和 / 或触发信号。
写控制位允许通过一次简单的写访问实现对每个功能的独立控制。

GxASCTRL (x = 0 - 1)
自动扫描源控制寄存器，组 x　　(x * 0400_H + 0520_H)　　复位值 : 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
TM WC	0	0	TM EN	0	0	0	0	GT WC	0	0	GT LVL	GT SEL			
w	r	r	rw	r	r	r	r	w	r	r	rh	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XT WC	XT MODE		XT LVL	XT SEL				0	0	0	0	SRCRESREG			
w	rw		rh	rw				r	r	r	r	rwr			

域	位	类型	描述
SRCRESREG	[3:0]	rw	**源专用结果寄存器** 0000_B使用 GxCHCTRy.RESREG 来选择一个组结果寄存器 0001_B保存结果到组结果寄存器 GxRES1 … 1111_B保存结果到组结果寄存器 GxRES15
0	[7:4]	r	**保留，读出值为 0，应写入 0**
XTSEL	[11:8]	rw	**外部触发输入选择** 连接的触发输入信号列于 **16-136 页的表 16-15 “XMC1300 中的数字连接 ”** *注：　XTSEL = 1111_B 使用所选择的门控输入作为触发源 (ENGT 必须为 $0x_B$)。*
XTLVL	12	rh	**外部触发电平** 所选触发输入的当前电平
XTMODE	[14:13]	rw	**触发操作模式** 00_B　无外部触发 01_B　在下降沿产生触发事件 10_B　在上升沿产生触发事件 11_B　在任何一个边沿产生触发事件

域	位	类型	描述
XTWC	15	w	**触发配置的写控制** 0_B 不能写触发配置 1_B 可以写位域 XTMODE 和 XTSEL
GTSEL	[19:16]	rw	**门控输入选择** 连接的门控输入信号见 **16-136 页的表 16-15 “XMC1300 中的数字连接 ”**
GTLVL	20	rh	**门控输入电平** 所选门控输入的当前电平
0	[22:21]	r	**保留，读出值为 0，应写入 0**
GTWC	23	w	**门控配置的写控制** 0_B 不能写门控配置 1_B 可以写位域 GTSEL
0	[27:24]	r	**保留，读出值为 0，应写入 0**
TMEN	28	rw	**定时器模式使能** 0_B 无定时器模式： 可使用标准门控机制 1_B 使能等距采样定时器模式： 必须禁止标准门控机制
0	[30:29]	r	**保留，读出值为 0，应写入 0**
TMWC	31	w	**定时器模式的写控制** 0_B 不能写定时器模式 1_B 可以写位域 TMEN

转换请求模式寄存器配置通道扫描请求源的工作模式。

GxASMR (x = 0 - 1)
自动扫描源模式寄存器，组 x　　(x * 0400_H + 0524_H)　　复位值 : $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RPT DIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	LD EV	CLR PND	REQ GT	0	LDM	SCA N	EN SI	EN TR		
r	r	r	r	r	r	w	w	rh	r	rw	rw	rw	rw	rw	

域	位	类型	描述
ENGT	[1:0]	rw	**使能门控** 为源 1 选择门控功能。 00_B　不发出转换请求 01_B　如果至少有一个挂起位被置位，发送转换请求 10_B　如果至少有一个挂起位被置位并且 REQGTx = 1，发送转换请求。 11_B　如果至少由一个挂起位被置位并且 REQGTx = 0，发送转换请求。 *注：　REQGTx 是所选择的门控信号。*
ENTR	2	rw	**使能外部触发** 0_B　禁止外部触发 1_B　在所选触发输入信号 REQTR 的所选边沿产生加载事件。
ENSI	3	rw	**使能源中断** 0_B　无请求源中断 1_B　在发生请求源事件时 （最后一个挂起的转换完成）产生请求源中断
SCAN	4	rw	**自动扫描使能** 0_B　无自动扫描 1_B　使能自动扫描功能： 一个请求源事件会自动产生一个加载事件。

域	位	类型	描述
LDM	5	rw	**自动扫描加载事件模式** 0_B 覆盖模式： 在发生一次加载事件时，将所选寄存器的所有位复制到挂起寄存器。 1_B 组合模式： 在发生一次加载事件时，将所选寄存器中的所有挂起位都置 1（逻辑或）。
0	6	r	**保留，读出值为 0，应写入 0**
REQGT	7	rh	**请求门控电平** **监视所选 REQGT 输入的电平。** 0_B 门控输入为低电平 1_B 门控输入为高电平
CLRPND	8	w	**清除挂起位** 0_B 无操作 1_B 寄存器 GxASPNDx 中的位被清零。
LDEV	9	w	**产生加载事件** 0_B 无操作 1_B 产生一次加载事件
0	[15:10]	r	**保留，读出值为 0，应写入 0**
RPTDIS	16	rw	**重复禁止** 0_B 重复被撤消的转换 1_B 忽略被撤消的转换
0	[31:17]	r	**保留，读出值为 0，应写入 0**

通道选择寄存器选择组扫描请求源要转换的通道。该寄存器中的位用于在发生加载事件时更新挂起寄存器。

有效通道位的数量取决于相应产品型号中的可用通道（参见 **16-134 页 “ 产品的具体配置 ”**）。

GxASSEL (x = 0 - 1)
自动扫描源通道选择寄存器，组 x　($x * 0400_H + 0528_H$)　　**复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**CHSEL7**	**CHSEL6**	**CHSEL5**	**CHSEL4**	**CHSEL3**	**CHSEL2**	**CHSEL1**	**CHSEL0**
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
CHSELy (y = 0 - 7)	y	rwh	**通道选择** 每个位 （当置位时）使能所在组的相应输入通道加入扫描序列。 0_B　忽略该通道 1_B　该通道是扫描序列的一部分。
0	[31:8]	r	**保留，读出值为 0，应写入 0**

注：　当写一个模式到寄存器 GxASPND 时，寄存器 GxASSEL 也被更新。

通道挂起寄存器指示当前转换序列中将要转换的通道。它们在发生加载事件时被选择寄存器更新。

GxASPND (x = 0 - 1)
自动扫描源挂起寄存器，组 x　　(x * 0400_H + $052C_H$)　　复位值 : $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**CHPND7**	**CHPND6**	**CHPND5**	**CHPND4**	**CHPND3**	**CHPND2**	**CHPND1**	**CHPND0**
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
CHPNDy (y = 0 - 7)	y	rwh	**通道挂起** **每位（当置位时）请求所在组的相应输入通道的转换。** 0_B　忽略该通道 1_B　请求对该通道的转换
0	[31:8]	r	**保留，读出值为 0，应写入 0**

注：　写寄存器 GxASPND 会自动更新寄存器 GxASSEL。

后台扫描源寄存器

后台扫描源只有一组寄存器。该源对整个 VADC 是公用的。

后台请求源的控制寄存器选择外部门和 / 或触发信号。
写控制位允许用一次简单的写访问单独控制每个功能。

BRSCTRL
后台请求源控制寄存器 **(0200_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	GTWC	0	0	GTLVL	GTSEL			
r	r	r	r	r	r	r	r	w	r	r	rh	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
XTWC	XTMODE		XTLVL	XTSEL				0	0	0	0	SRCRESREG			
w	rw		rh	rw				r	r	r	r	rw			

域	位	类型	描述
SRCRESREG	[3:0]	rw	**源专用结果寄存器** 0000_B使用 GxCHCTRy.RESREG 来选择一个组结果寄存器。 0001_B保存结果到组结果寄存器 GxRES1 … 1111_B保存结果到组结果寄存器 GxRES15
0	[7:4]	r	**保留，读出值为 0，应写入 0**
XTSEL	[11:8]	rw	**外部触发输入选择** 连接的触发输入信号见 **16-136 页的表 16-15 “XMC1300 中的数字连接 ”** *注： XTSEL = 1111_B 使用所选门控输入作为触发源 (ENGT 必须是 $0X_B$)。*
XTLVL	12	rh	**外部触发电平** 所选触发输入的当前电平
XTMODE	[14:13]	rw	**触发操作模式** 00_B 无外部触发 01_B 在下降沿产生触发事件 10_B 在上升沿产生触发事件 11_B 在任一边沿产生触发事件

域	位	类型	描述
XTWC	15	w	**触发配置的写控制** 0_B 不能写触发配置 1_B 可以写位域 XTMODE 和 XTSEL
GTSEL	[19:16]	rw	**门控输入选择** 连接的门控输入信号见 **16-136 页的表 16-15 “XMC1300 中的数字连接 ”**
GTLVL	20	rh	**门控输入电平** 所选门控输入的当前电平
0	[22:21]	r	**保留，读出值为 0，应写入 0**
GTWC	23	w	**门控配置的写控制** 0_B 不能写门控配置 1_B 可以写位域 GTSEL
0	[31:24]	r	**保留，读出值为 0，应写入 0**

转换请求模式寄存器配置后台请求源的工作模式。

BRSMR
后台请求源模式寄存器 **(0204_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	RPT DIS
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	LD EV	CLR PND	REQ GT	0	LDM	SCA N	EN SI	EN TR	ENGT	
r	r	r	r	r	r	w	w	rh	r	rw	rw	rw	rw	rw	

域	位	类型	描述
ENGT	[1:0]	rw	**使能门控** **为源 1 选择门控功能。** 00_B 不发出转换请求 01_B 如果至少有一个挂起位被置 1，发出转换请求 10_B 如果至少有一个挂起位被置 1 并且 REQGTx = 1，发出转换请求。 11_B 如果至少有一个挂起位被置 1 并且 REQGTx = 0，发出转换请求。 *注： REQGTx 是所选择的门控信号。*
ENTR	2	rw	**使能外部触发** 0_B 禁止外部触发 1_B 在所选触发输入信号 REQTR 的所选边沿产生加载事件。
ENSI	3	rw	**使能源中断** 0_B 无请求源中断 1_B 在发生一次请求源事件时 （最后一个挂起的转换完成）产生请求源中断。
SCAN	4	rw	**使能自动扫描** 0_B 无自动扫描 1_B 使能自动扫描功能 一个请求源事件会自动产生一次加载事件。

域	位	类型	描述
LDM	5	rw	**自动扫描加载事件模式** 0_B 覆盖模式： 在发生一次加载事件时，将所选寄存器中的所有位复制到挂起寄存器。 1_B 组合模式： 在发生一次加载事件时，将所选寄存器中的所有挂起位都置 1（逻辑或）。
0	6	r	**保留，读出值为 0，应写入 0**
REQGT	7	rh	**请求门控电平** 监视所选 REQGT 输入的电平。 0_B 门控输入为低电平 1_B 门控输入为高电平
CLRPND	8	w	**清除挂起位** 0_B 无操作 1_B 寄存器 GxASPNDx 中的位被清零
LDEV	9	w	**产生加载事件** 0_B 无操作 1_B 产生一次加载事件
0	[15:10]	r	**保留，读出值为 0，应写入 0**
RPTDIS	16	rw	**禁止重复** 0_B 重复被撤消的转换 1_B 忽略被撤消的转换
0	[31:17]	r	**保留，读出值为 0，应写入 0**

通道选择寄存器选择后台请求源 （通道扫描源）将要转换的通道。该寄存器中的位用于在发生加载事件时更新挂起寄存器。

有效通道位的数量取决于相应产品型号中的可用通道（参见 **16-134 页 “ 产品的具体配置 ”**）。

注： 在寄存器 **GxCHASS (x = 0 - 1)** *中选择的优先通道不会被转换。*

BRSSELx (x = 0 - 1)
后台请求源通道选择寄存器，组 x(0180_H + x * 0004_H) **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**CHSELG7**	**CHSELG6**	**CHSELG5**	**CHSELG4**	**CHSELG3**	**CHSELG2**	**CHSELG1**	**CHSELG0**
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
CHSELGy (y = 0 - 7)	y	rwh	**通道选择组 x** 每个位 （当置位时）使能所在组的相应输入通道加入扫描序列。 0_B 忽略该通道 1_B 该通道是扫描序列的一部分。
0	[31:8]	r	**保留，读出值为 0，应写入 0**

通道挂起寄存器指示当前转换序列中要被转换的通道。当发生一次加载事件时，它们被选择寄存器更新。

BRSPNDx (x = 0 - 1)
后台请求源挂起寄存器 , 组 x　　($01C0_H$ + x * 0004_H)　　复位值 : 0000 0000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**CHPNDG7**	**CHPNDG6**	**CHPNDG5**	**CHPNDG4**	**CHPNDG3**	**CHPNDG2**	**CHPNDG1**	**CHPNDG0**
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
CHPNDGy (y = 0 - 7)	y	rwh	**通道挂起组 x** 每个位 （当置位时）请求所在组的相应输入通道的转换。 0_B　忽略该通道 1_B　请求对该通道进行转换
0	[31:8]	r	**保留，读出值为 0，应写入 0**

注：　对寄存器 BRSPNDx 中的任何一个进行写操作会更新相应的寄存器 BRSSELx，并且产生一个加载事件，该事件将所有寄存器 BRSSELx 中的所有位复制到 BRSPNDx。
只能在写请求模式的最后一个字时使用这一快速方式。

16.16.5 通道控制寄存器

G0CHCTRy (y = 0 - 7)
组 0, 通道 y 控制寄存器 **(0600_H + y * 0004_H)** **复位值 : 0000 0000_H**
G1CHCTRy (y = 0 - 7)
组 1, 通道 y 控制寄存器 **($0A00_H$ + y * 0004_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	BWDEN	BWDCH		0	0	0	0	0	0	RESPOS	RESTBS	RESREG			
r	rw	rw		r	r	r	r	r	r	rw	rw	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BNDSELX				REFSEL	SYNC	CHEVMODE		BNDSELU		BNDSELL		0	0	ICLSEL	
rw				rw	rw	rw		rw		rw		r	r	rw	

域	位	类型	描述
ICLSEL	[1:0]	rw	**输入类选择** 00_B 使用组专用类 0 01_B 使用组专用类 1 10_B 使用全局类 0 11_B 使用全局类 1
0	[3:2]	r	**保留，读出值为 0，应写入 0**
BNDSELL	[5:4]	rw	**下边界选择**[1)] 00_B 使用组专用边界 0 01_B 使用组专用边界 1 10_B 使用全局边界 0 11_B 使用全局边界 1
BNDSELU	[7:6]	rw	**上边界选择**[1)] 00_B 使用组专用边界 0 01_B 使用组专用边界 1 10_B 使用全局边界 0 11_B 使用全局边界 1

域	位	类型	描述
CHEVMODE	[9:8]	rw	**通道事件模式** 产生一个通道事件，或者在带有极限检查[2] 的比较模式 （NCM），或者在快速比较模式 （FCM）[3]。 00_B 不产生事件 01_B NCM：如果结果位于边界范围内 FCM：如果结果变高 (高于比较值) 10_B NCM：如果结果超出边界范围 FCM：如果结果变低 (低于比较值) 11_B NCM：总是产生事件 (忽视范围) FCM：如果结果切换到另一电平
SYNC	10	rw	**同步请求** 0_B 无同步请求，独立操作 1_B 请求对该通道进行一次同步转换 (仅考虑主内核)
REFSEL	11	rw	**参考输入选择** 定义对该通道进行转换时使用的参考电压输入。 0_B 标准参考地 V_{SSC} 1_B 来自 CH0 的备选参考地[4]
BNDSELX	[15:12]	rw	**边界扩展 [1]** 0000_B标准模式： 通过 BNDSELU/BNDSELL 选择边界 0001_B使用结果寄存器 GxRES1 作为上边界 … 1111_B使用结果寄存器 GxRES15 作为上边界
RESREG	[19:16]	rw	**结果寄存器** 0000_B保存结果到组结果寄存器 GxRES0 … 1111_B保存结果到组结果寄存器 GxRES15
RESTBS	20	rw	**后台源的结果目标** 0_B 保存结果到所选择的组结果寄存器 1_B 保存结果到全局结果寄存器
RESPOS	21	rw	**结果位置** 0_B 以左对齐方式保存结果 1_B 以右对齐方式保存结果
0	[27:22]	r	**保留，读出值为 0，应写入 0**

域	位	类型	描述
BWDCH	[29:28]	rw	**断线检测通道** 00_B 选择 V_{AREF} 01_B 选择 V_{AGND} 10_B 保留 11_B 保留
BWDEN	30	rw	**断线检测使能** 0_B 正常操作 1_B 使能额外的准备阶段
0	31	r	**保留，读出值为 0，应写入 0**

1) 当 BNDSELX ≠ 0000_B 时，位域 BNDSELU 和 BNDSELL 被级联，并选择相应的结果寄存器作为下边界。
2) 边界范围被定义为这样一个区域：结果小于或等于所选择的上边界并且大于或等于所选择的下边界，见 16.8.4 节。
3) 结果是所选结果寄存器中的位 FCR。
4) 有些通道不能选择备用参考。

GxICLASS0 (x = 0 - 1)
输入类寄存器 0, 组 x **($x * 0400_H + 04A0_H$)** **复位值 : $0000\ 0000_H$**
GxICLASS1 (x = 0 - 1)
输入类寄存器 1, 组 x **($x * 0400_H + 04A4_H$)** **复位值 : $0000\ 0000_H$**
GLOBICLASSy (y = 0 - 1)
输入类寄存器 y, 全局 **($00A0_H + y * 0004_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	CME			0	0	0	STCE				
r	r	r	r	r	rw			r	r	r	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	CMS			0	0	0	STCS				
r	r	r	r	r	rw			r	r	r	rw				

域	位	类型	描述
STCS	[4:0]	rw	**标准转换的采样时间控制** 在最小采样时间 （2 个模拟时钟周期）基础上要额外增加的时钟周期数： 编码和结果采样时间见表 16-11。 对于外部通道转换，可使用来自位域 STCE 的值。

域	位	类型	描述
0	[7:5]	r	**保留，读出值为 0，应写入 0**
CMS	[10:8]	rw	**标准转换的转换模式** 000_B 12 位转换 001_B 10 位转换 010_B 8 位转换 011_B 保留 100_B 保留 101_B 10 位快速比较模式 110_B 保留 111_B 保留
0	[15:11]	r	**保留，读出值为 0，应写入 0**
STCE	[20:16]	rw	**EMUX 转换的采样时间控制** 在最小采样时间 （2 个模拟时钟周期）基础上要额外增加的时钟周期数： 编码和结果采样时间见**表 16-11**。 对于标准通道转换，可使用来自位域 STCS 的值。
0	[23:21]	r	**保留，读出值为 0，应写入 0**
CME	[26:24]	rw	**EMUX 转换的转换模式** 000_B 12 位转换 001_B 10 位转换 010_B 8 位转换 011_B 保留 100_B 保留 101_B 10 位快速比较模式 110_B 保留 111_B 保留
0	[31:27]	r	**保留，读出值为 0，应写入 0**

表 16-11　采样时间编码

STCS / STCE	额外时钟周期	采样时间
$0\ 0000_B$	0	2 / f_{ADCI}
$0\ 0001_B$	1	3 / f_{ADCI}
…	…	…
$0\ 1111_B$	15	17 / f_{ADCI}
$1\ 0000_B$	16	18 / f_{ADCI}
$1\ 0001_B$	32	34 / f_{ADCI}

表 16-11　采样时间编码 (续表)

STCS / STCE	额外时钟周期	采样时间
…	…	…
$1\ 1110_B$	240	242 / f_{ADCI}
$1\ 1111_B$	256	258 / f_{ADCI}

16.16.6　结果寄存器

组结果控制寄存器选择一个给定组的结果寄存器的行为。

G0RCRy (y = 0 - 15)
组 0 结果控制寄存器 y　　**(0680_H + y * 0004_H)**　　**复位值 : $0000\ 0000_H$**
G1RCRy (y = 0 - 15)
组 1 结果控制寄存器 y　　**($0A80_H$ + y * 0004_H)**　　**复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRGEN	0				FEN		WFR	0		DMM		DRCTR			
rw	r				rw		rw	r		rw		rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															
r															

域	位	类型	描述
0	[15:0]	r	**保留，读出值为 0，应写入 0**
DRCTR	[19:16]	rw	**数据缩减控制** 定义如何将结果值保存 / 累加到该寄存器作为最终结果。数据缩减计数器 DRC 可从该位域加载。 位域 DRCTR 的功能由位域 DMM 决定。
DMM	[21:20]	rw	**数据修改模式** 00_B　标准数据缩减 (累加) 01_B　结果滤波模式[1)] 10_B　差分模式 11_B　保留 见 **16-40 页 " 数据修改 "**
0	[23:22]	r	**保留，读出值为 0，应写入 0**

域	位	类型	描述
WFR	24	rw	**待读模式使能** 0_B 覆盖模式 1_B 使能该寄存器的待读模式
FEN	[26:25]	rw	**FIFO 模式使能** 00_B 分离结果寄存器 01_B FIFO 结构的一部分： 复制每个新的有效结果 10_B 最大值模式： 如果新值更大，复制新的结果。 11_B 最小值模式： 如果新值更小，复制新的结果
0	[30:27]	r	**保留，读出值为 0，应写入 0**
SRGEN	31	rw	**服务请求产生使能** 0_B 无服务请求 1_B 发生结果事件后发出服务请求

1) 当位域 DMM ≠ 01_B 时，滤波器寄存器被清零。

组结果寄存器为一个给定组的所有通道提供可选择的存储位置。

注： 快速比较模式中使用的预设值被写入到相应的结果寄存器。调试结果寄存器是不可写的。

G0RESy (y = 0 - 15)
组 0 结果寄存器 y **(0700_H + y * 0004_H)** **复位值 : $0000\ 0000_H$**
G1RESy (y = 0 - 15)
组 1 结果寄存器 y **($0B00_H$ + y * 0004_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
VF	**FCR**	**CRS**		**EMUX**			**CHNR**					**DRC**			
rh	rh	rh		rh			rh					rh			

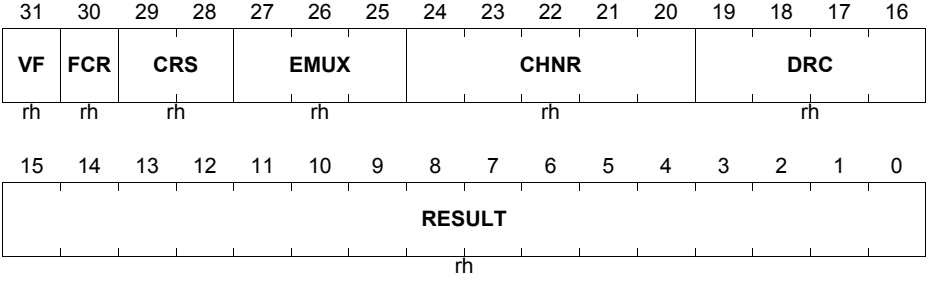
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RESULT															
rwh															

域	位	类型	描述
RESULT	[15:0]	rwh	**最近一次转换的结果** 该位域中结果位的位置取决于所配置的工作模式，请参见 **16.11.2 节**。
DRC	[19:16]	rh	**数据缩减计数器** 指示还要被累加到最终结果的转换值的数目。当位域 DRC 变为 0 时 （通过减 1 计数或重新加载），最终结果可用，有效标志 VF 被置位。 见 **16-40 页 “ 数据修改 ”**
CHNR	[24:20]	rh	**通道号** 指示与位域 RESULT 中的值对应的通道号。
EMUX	[27:25]	rh	**外部多路复用器设置** 指示外部多路复用器的设置，与位域 RESULT 中的值对应。 *注： 仅在 GxRES0 中可用。 如果需要 EMUX 信息，使用 GxRES0。*

域	位	类型	描述
CRS	[29:28]	rh	**转换请求源** 指示请求源，位域 RESULT 中的结果值属于该请求源所请求的转换。 00_B 请求源 0 01_B 请求源 1 10_B 请求源 2 11_B 保留
FCR	30	rh	**快速比较结果** 指示在快速比较模式进行的一次操作的结果。 0_B 信号电平低于比较值 1_B 信号电平高于比较值
VF	31	rh	**有效标志** 指示在位域 RESULT 或位 FCR 中有一个新结果。 0_B 无新结果可用 1_B 位域 RESULT 被更新为新结果值但尚未被读取，或位 FCR 被更新。

组结果寄存器的调试模式提供对一个给定组的所有结果寄存器的访问，而不清除有效标志。

G0RESDy (y = 0 - 15)
组 0 结果寄存器 y, 调试 **(0780_H + y * 0004_H)** **复位值 : 0000 0000_H**
G1RESDy (y = 0 - 15)
组 1 结果寄存器 y, 调试 **($0B80_H$ + y * 0004_H)** **复位值 : 0000 0000_H**



域	位	类型	描述
RESULT	[15:0]	rh	**最新转换结果** 该位域中的结果位的位置取决于所配置的工作模式。 参见 **16.11.2 节**。

域	位	类型	描述
DRC	[19:16]	rh	**数据缩减计数器** 指示还要被累加作为最终结果的转换值的数目。当位域 DRC 变为 0 时 （通过减 1 计数或重新加载），最终结果可用，有效标志 VF 被置位。 见 16-40 页 “ 数据修改 ”
CHNR	[24:20]	rh	**通道号** 指示与位域 RESULT 中的值对应的通道号。
EMUX	[27:25]	rh	**外部多路复用器设置** 指示外部多路复用器的设置，与位域 RESULT 中的值对应。 *注： 仅在 GxRES0 中可用。 如果需要 EMUX 信息，使用 GxRES0。*
CRS	[29:28]	rh	**转换请求源** 指示请求源，位域 RESULT 中的结果值属于该请求源所请求的转换。 00_B 请求源 0 01_B 请求源 1 10_B 请求源 2 11_B 保留
FCR	30	rh	**快速比较结果** 指示在快速比较模式进行的一次操作的结果。 0_B 信号电平低于比较值 1_B 信号电平高于比较值
VF	31	rh	**有效标志** 指示在位域 RESULT 或位 FCR 中有一个新结果。 0_B 无新结果可用 1_B 位域 RESULT 被新结果值更新但尚未被读取，或位 FCR 已被更新。

全局结果控制寄存器选择全局结果寄存器的行为。

GLOBRCR
全局结果控制寄存器 **(0280_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SRGEN	0	0	0	0	0	0	WFR	0	0	0	0	DRCTR			
rw	r	r	r	r	r	r	rw	r	r	r	r	rw			

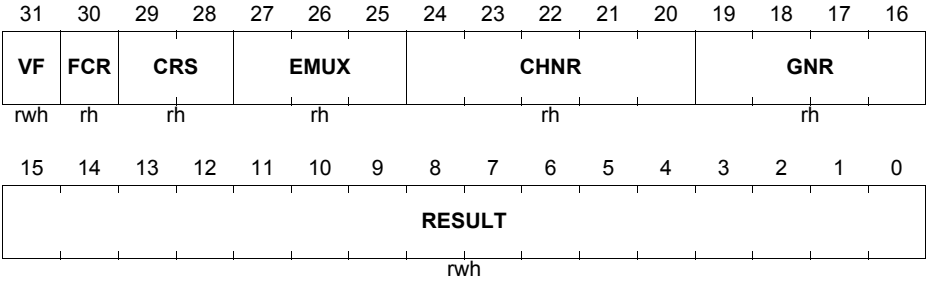
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

域	位	类型	描述
0	[15:0]	r	**保留，读出值为 0，应写入 0**
DRCTR	[19:16]	rw	**数据缩减控制** 定义如何将结果值保存 / 累加到寄存器作为最终结果。数据缩减计数器 DRC 可从该位域加载。 0000_B 禁止数据缩减 其他：见 **16-41 页 " 位域 DRCTR 的功能 "** 1)
0	[23:20]	r	**保留，读出值为 0，应写入 0**
WFR	24	rw	**待读模式使能** 0_B 覆盖模式 1_B 使能该寄存器的待读模式
0	[30:25]	r	**保留，读出值为 0，应写入 0**
SRGEN	31	rw	**服务请求产生使能** 0_B 无服务请求 1_B 在发生一次结果事件后产生服务请求

1) 对于全局结果寄存器，只有标准数据缩减是可用的，即假定 DMM 为 00_B。

全局结果寄存器为所有组的所有通道提供一个公共的存储位置。

GLOBRES
全局结果寄存器 **(0300_H)** **复位值 : 0000 0000$_H$**
GLOBRESD
全局结果寄存器，调试 **(0380_H)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
RESULT	[15:0]	rwh	**最新转换结果** 该位域中的结果位的位置取决于所配置的工作模式 [1]。 参见 **16.11.2 节**。
GNR	[19:16]	rh	**组号** 指示位域 CHNR 中的通道号所在的组。
CHNR	[24:20]	rh	**通道号** 指示与位域 RESULT 中的值对应的通道号。
EMUX	[27:25]	rh	**外部多路复用器设置** 指示与位域 RESULT 中的值对应的外部多路复用器的设置。
CRS	[29:28]	rh	**转换请求源** 指示请求源，位域 RESULT 中的结果值属于该请求源所请求的转换。
FCR	30	rh	**快速比较结果** 指示在快速比较模式下进行的一次操作的结果。 0_B 信号电平低于比较值 1_B 信号电平高于比较值

域	位	类型	描述
VF	31	rwh	**有效标志** 指示在位域 RESULT 或位 FCR 中有一个新结果。 0_B 读访问：无新有效数据可用 写访问：无效 1_B 读访问：位域 RESULT 包含有效数据并且尚未被读取，或者位 FCR 已被更新。 写访问：清除该有效标志和数据缩减计数器。(盖写一次硬件置位操作)[1)]

1) 只能写入寄存器 GLOBRES 中，不能写到寄存器 GLOBRESD 中。

有效标志寄存器包含所有结果寄存器的有效标志。

GxVFR (x = 0 - 1)
有效标志寄存器 , 组 x **($x * 0400_H + 05F8_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
VF15	**VF14**	**VF13**	**VF12**	**VF11**	**VF10**	**VF9**	**VF8**	**VF7**	**VF6**	**VF5**	**VF4**	**VF3**	**VF2**	**VF1**	**VF0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
VFy **(y = 0 - 15)**	y	rwh	**结果寄存器 x 的有效标志** 指示在位域 RESULT 或位 FCR 中有一个新结果。 0_B 读访问：无新有效数据可用 写访问：无效 1_B 读访问：结果寄存器 x 包含有效数据并且尚未被读取，或位 FCR 已被更新。 写访问：清除该有效标志和寄存器 GxRESy 中的位域 DRC。(盖写一次硬件置位操作)
0	[31:16]	r	**保留，读出值为 0，应写入 0**

16.16.7 校准寄存器

校准控制寄存器 CALCTR 选择基本校准功能。

SHS0_CALCTR
校准控制寄存器 **(4803 40BC$_H$)** **复位值 : 0010 0400$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
SUCAL	0	CALMAX						0	SUCALVAL						
w	r	rw						r	rw						

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	CALGNSTC						0	0	0	0	0	0	0	CALORD
r	r	rw						r	r	r	r	r	r	r	rw

域	位	类型	描述
CALORD	0	rw	**校准顺序** 0_B 先转换后校准 1_B 先校准后转换
0	[7:1]	r	**保留，读出值为 0，应写入 0**
CALGNSTC	[13:8]	rw	**增益校准采样时间控制** 定义增益校准循环采样阶段的持续时间。 00_H 保留 其他：t_{SGN} = CALGNSTC / f_{SH} *注： 无需用户控制。*
0	[15:14]	r	**保留，读出值为 0，应写入 0**
SUCALVAL	[22:16]	rw	**初始校准循环** 为初始校准定义校准序列的数量。 *注： 无需用户控制。*
CALMAX	[29:24]	rw	**校准最大时间** 定义直到下一次校准的最大时间： t_{MAX} = 512 / f_{SH} × (CALMAX + 1)
0	30	r	**保留，读出值为 0，应写入 0**

域	位	类型	描述
SUCAL	31	w	**初始校准** 位 SUCAL 的 0-1 跳变触发转换器的初始校准阶段 1)。 当校准阶段开始时， SUCAL 被自动清零。 0_B 无操作 1_B 启动初始校准阶段 (位域 SHSCFG.STATE 中的指示标志)

1) 在模块 VADCDIG 中，启动校准也可以由寄存器 GLOBCFG 中的位 SUCAL 触发。

增益校准控制寄存器 CALGCx 配置相关采样保持单元的增益校准功能。校准值由校准机制自动决定。可读取校准值来补偿因使用另一组的备用参考所引入的误差。

SHS0_CALGCx (x = 0 - 1)
增益校准控制寄存器 x **(4803 40C0$_H$ + x * 4$_H$)** **复位值 : 2000 2000$_H$**

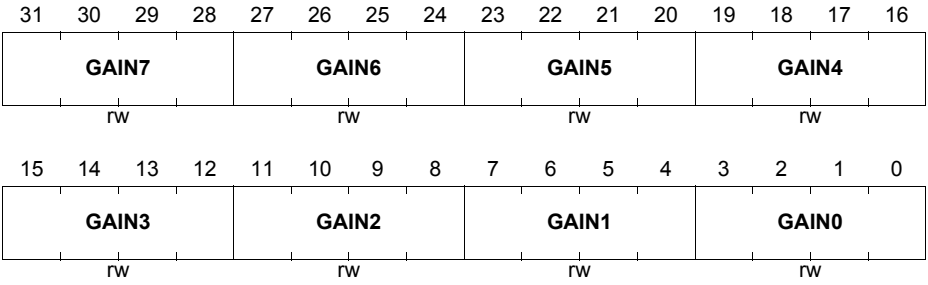
31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
GNAWC	**0**	**CALGNVALA**													
w	r	rwh													

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GNSWC	**0**	**CALGNVALS**													
w	r	rwh													

域	位	类型	描述
CALGNVALS	[13:0]	rwh	**增益校准值，标准参考** 由校准步骤来修改，控制 DAC 的增益。
0	14	r	**保留，读出值为 0，应写入 0**
GNSWC	15	w	**增益校准写控制，标准** 0_B 不能写增益校准参数 1_B 可以写 CALGNVALS
CALGNVALA	[29:16]	rwh	**增益校准值，备用参考** 由校准步骤来修改，控制 DAC 的增益。
0	30	r	**保留，读出值为 0，应写入 0**
GNAWC	31	w	**增益校准写控制，备选** 0_B 不能写增益校准参数 1_B 可以 CALGNVALA

增益控制寄存器 GNCTRx0 为每个输入选择增益。

SHS0_GNCTR00
增益控制寄存器 00 **(4803 4180$_H$)** **复位值 : 0000 0000$_H$**
SHS0_GNCTR10
增益控制寄存器 10 **(4803 4190$_H$)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
GAINz **(z = 0 - 7)**	[z*4+3: z*4]	rw	**增益控制 z** 0000_B 增益因子 = 1 0001_B 增益因子 = 3 0010_B 增益因子 = 6 0011_B 增益因子 = 12 其他：保留

注： 第一个索引 (x0) 指示相关联的采样保持单元。
通道号为 z。

16.16.8 其他寄存器

别名寄存器可用另一通道号替代通道 CH0 和 CH1 的通道号。器件复位后，寄存器中的默认值禁止这种重定向。

GxALIAS (x = 0 - 1)
别名寄存器，组 x **($x * 0400_H + 04B0_H$)** **复位值 : $0000\ 0100_H$**

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
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	ALIAS1					0	0	0	ALIAS0				
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	rw					r	r	r	rw				

域	位	类型	描述
ALIAS0	[4:0]	rw	**CH0 转换请求的别名值** 指示被转换的通道，而不是通道 CH0。转换操作仍使用 CH0 的通道设置。
0	[7:5]	r	**保留，读出值为 0，应写入 0**
ALIAS1	[12:8]	rw	**CH1 转换请求的别名值** 指示被转换的通道，而不是通道 CH1。转换操作仍使用 CH1 的通道设置。
0	[31:13]	r	**保留，读出值为 0，应写入 0**

本地边界寄存器 GxBOUND 定义快速比较模式的组专用边界值或增量极限。
全局边界寄存器 GLOBBOUND 为所有的通道定义通用比较值。

使用位域的左 12/10/8 位，这取决于转换宽度。对于 10/8 位结果，低 2/4 位必须是 0。

GxBOUND (x = 0 - 1)
边界选择寄存器 , 组 x **($x * 0400_H + 04B8_H$)** **复位值 : $0000\ 0000_H$**

GLOBBOUND
全局边界选择寄存器 **($00B8_H$)** **复位值 : $0000\ 0000_H$**

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
0	0	0	0	BOUNDARY1												0	0	0	0	BOUNDARY0											
r	r	r	r	rw												r	r	r	r	rw											

域	位	类型	描述
BOUNDARY0	[11:0]	rw	**用于极限检测的边界值 0** 标准模式：该值与左对齐的转换结果比较。 快速比较模式：该值被加到参考值 （向上增量）。
0	[15:12]	r	**保留，读出值为 0，应写入 0**
BOUNDARY1	[27:16]	rw	**用于极限检测的边界值 1** 标准模式：该值与左对齐的转换结果比较。 快速比较模式：从参考值中减去该值 （向下增量）。
0	[31:28]	r	**保留，读出值为 0，应写入 0**

边界标志寄存器保持自己的边界标志和为每个标志选择激活条件和输出信号极性的位。

GxBFL (x = 0 - 1)
边界标志寄存器 , 组 x **($x * 0400_H + 04C8_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**BFI 3**	**BFI 2**	**BFI 1**	**BFI 0**
r	r	r	r	r	r	r	r	r	r	r	r	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**BFA 3**	**BFA 2**	**BFA 1**	**BFA 0**	**0**	**0**	**0**	**0**	**BFL 3**	**BFL 2**	**BFL 1**	**BFL 0**
r	r	r	r	rw	rw	rw	rw	r	r	r	r	rh	rh	rh	rh

域	位	类型	描述
BFLy **(y = 0 - 3)**	y	rh	**边界标志 y** 0_B 无效状态： 结果未超越边界 （见位域 BFAy）， 或所选门控信号不活动。 或该边界标志被禁止 1_B 有效状态： 结果超越边界。
0	[7:4]	r	**保留，读出值为 0，应写入 0**
BFAy **(y = 0 - 3)**	8 + y	rw	**边界标志 y 激活选择** 0_B 如果结果大于所定义的边界或比较值，则置位边界标志 BFLy ；如果低于该值，则清除该边界标志。 1_B 如果结果小于所定义的边界或比较值，则置位边界标志 BFLy ；如果高于该值，则清除该边界标志。
0	[15:12]	r	**保留，读出值为 0，应写入 0**
BFIy **(y = 0 - 3)**	16 + y	rw	**边界标志 y 取反控制** 0_B 直接使用 BFLy 1_B 取反相值，使用 $\overline{\text{BFLy}}$
0	[31:20]	r	**保留，读出值为 0，应写入 0**

边界标志软件寄存器通过软件提供每个标志的设置或清零。

GxBFLS (x = 0 - 1)
边界标志软件寄存器 , 组 x **($x * 0400_H + 04CC_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**BFS 3**	**BFS 2**	**BFS 1**	**BFS 0**
r	r	r	r	r	r	r	r	r	r	r	r	w	w	w	w

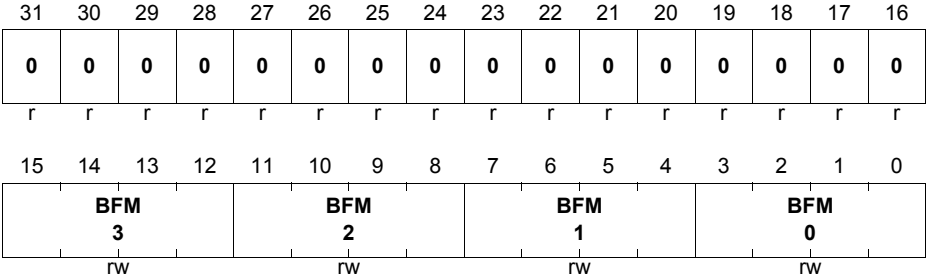
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**BFC 3**	**BFC 2**	**BFC 1**	**BFC 0**
r	r	r	r	r	r	r	r	r	r	r	r	w	w	w	w

域	位	类型	描述
BFCy (y = 0 - 3)	y	w	**边界标志 y 清零** 0_B 无操作 1_B 将位 BFLy 清零
0	[15:4]	r	**保留，读出值为 0，应写入 0**
BFSy (y = 0 - 3)	16 + y	w	**边界标志 y 置位** 0_B 无操作 1_B 将位 BFLy 置位
0	[31:20]	r	**保留，读出值为 0，应写入 0**

注： 如果边界标志与快速比较模式一起使用，建议不要将来自其他通道的结果定向到相应的结果寄存器。

边界标志控制寄存器选择边界标志的基本操作。

GxBFLC (x = 0 - 1)
边界标志控制寄存器，组 x **($x * 0400_H + 04D0_H$)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
BFM0, **BFM1,** **BFM2,** **BFM3**	[3:0], [7:4], [11:8], [15:12]	rw	**边界标志 y 模式控制** 0000_B 禁止边界标志，BFLy 不变 0001_B 总是使能边界标志 （按照比较结果）。 0010_B 当源 0 的门控活动时，使能边界标志；当门控不活动时，清除 BFLy 。 0011_B 当源 1 的门控活动时，使能边界标志；当门控不活动时，清除 BFLy 。 其他：保留
0	[31:16]	r	**保留，读出值为 0，应写入 0**

边界标志节点指针寄存器将信号 GxBFLOUTy 定向到与其他模块相连的备选片上连接（除了组专用输出之外）。可能的目标是相应的公共服务请求线或公共边界标志输出 (CBFL0 … CBFL3)。

GxBFLNP (x = 0 - 1)
边界标志节点指针寄存器 , 组 x　($(x * 0400_H + 04D4_H)$)　　**复位值 : 0000 FFFF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BFL3NP				**BFL2NP**				**BFL1NP**				**BFL0NP**			
rw				rw				rw				rw			

域	位	类型	描述
BFL0NP, BFL1NP, BFL2NP, BFL3NP	[3:0], [7:4], [11:8], [15:12]	rw	**边界标志 y 节点指针** 0000_B 选择公共边界标志输出 0 … 0011_B 选择公共边间标志输出 3 0100_B 选择公共服务请求线 C0SR0 … 0111_B 选择公共服务请求线 C0SR3 1111_B 禁止，无公共输出信号。 其他：保留
0	[31:16]	r	**保留，读出值为 0，应写入 0**

同步控制寄存器为并行转换控制内核的同步。

注： *仅当转换组的所有 ADC 内核中的位域 GxARBCFG.ANONS = 00_B 时，才能对寄存器 GxSYNCTR 编程。然后将主内核的位域 ANONC 设置为 11_B 。*

GxSYNCTR (x = 0 - 1)
同步控制寄存器 , 组 x **(x * 0400_H + $04C0_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**EVALR1**	**0**	**0**	**STSEL**	
r	r	r	r	r	r	r	r	r	r	r	rw	r	r	rw	

域	位	类型	描述
STSEL	[1:0]	rw	**启动选择** 控制 ADC 内核的同步机制。 00_B 该内核是同步主内核：使用自己的位域 GxARBCFG.ANONC 01_B 该内核是同步从内核：控制信息来自输入 CI1 10_B 保留 11_B 保留 *注：* *控制输入 CIx 见图 16-26，连接的内核见表 16-13。*
0	[3:2]	r	**保留，读出值为 0，应写入 0**
EVALR1	4	rw	**评估就绪输入 R1** 为一个转换组的一个内核使能就绪输入信号。 0_B 无就绪输入控制 1_B 就绪输入 R1 被看做该转换组的一次并行转换的开始。
0	[31:5]	r	**保留，读出值为 0，应写入 0**

GxEMUXCTR (x = 0 - 1)
外部多路复用器控制寄存器 , 组 x (x * 0400_H + $05F0_H$) **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EMX WC	EMX CSS	EMX ST	EMX COD	EMUX MODE		EMUX CH									
w	rw	rw	rw	rw		rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0		EMUX ACT		0	0	0	0	0	EMUX SET		
r	r	r	r	r		rh		r	r	r	r	r	rw		

域	位	类型	描述
EMUXSET	[2:0]	rw	**外部多路复用控制起始选择**[1)] 为外部多路复用控制器定义初始设置
0	[7:3]	r	**保留，读出值为 0，应写入 0**
EMUXACT	[10:8]	rh	**外部多路复用器当前选择** 定义外部多路复用器选择的当前值。该位域从 EMUXSET 位域加载，并根据由位域 EMUXMODE 选择的工作模式修改。
0	[15:11]	r	**保留，读出值为 0，应写入 0**
EMUXCH	[25:16]	rw	**外部多路复用器通道选择** 定义外部多路复用器控制所应用的通道 **EMXCSS = 0：通道号** 低 5 位选择一个任意的通道 (有效数值受可用通道数的限制，未使用的位应为 0) **EMXCSS = 1：通道使能** 每个位使能相关联的通道 (可选择 / 使能多个通道)
0	[25:21]	r	**保留，读出值为 0，应写入 0**
EMUXMODE	[27:26]	rw	**外部多路复用器模式** 00_B 软件控制 (无硬件操作) 01_B 稳定模式 (使用 EMUXSET 的值) 10_B 单步模式 [1)2)] 11_B 序列模式 [1)]

域	位	类型	描述
EMXCOD	28	rw	**外部多路复用器编码方案** 0_B 用二进制输出通道号 1_B 用格雷码输出通道号
EMXST	29	rw	**外部多路复用器采样时间控制** 0_B 当设置改变时，使用 STCE 1_B 对一个外部通道的每次转换，使用 STCEI
EMXCSS	30	rw	**外部多路复用通道选择方式** 0_B 通道号： 位域 EMUXCH 选择一个任意通道 1_B 通道使能： 位域 EMUXCH 的每一位选择 EMUX 控制的相关通道
EMXWC	31	w	**EMUX 配置的写控制** 0_B 不能写 EMUX 配置。 1_B 可以写位域 EMXMODE、 EMXCOD、EMXST、EMXCSS

1) 对单步模式和序列模式：在选择相应的模式之前选择起始值。
2) 每当一个 EMUX 使能的通道被转换时，单步模式都会修改 EMUX 通道号。因此，单步模式在只用一个通道工作时为最佳，因为否则的话可能要跳过一些外部通道。

寄存器 EMUXSEL 是一个全局寄存器，它分配一个任意组到每个 EMUX 接口。

EMUXSEL
外部多路复用器选择寄存器 (03F0$_H$) 复位值 : 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**EMUX GRP1**				**EMUX GRP0**			
r	r	r	r	r	r	r	r	rw				rw			

域	位	类型	描述
EMUXGRP0, EMUXGRP1	[3:0], [7:4]	rw	**接口 x 的外部多路复用器组** 定义组，其外部多路复用器控制信号连接到 EMUX 接口 x。[1)]
0	[31:8]	r	**保留，读出值为 0，应写入 0**

1) 与每个 EMUX 接口相关联的引脚列于 **16-136 页的表 16-15 “XMC1300 中的数字连接 ”**。

Σ-Δ 循环控制寄存器 LOOP 配置 Σ-Δ 循环的功能。

SHS0_LOOP
循环控制寄存器 **(4803 4050$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
LP EN1	**0**	**0**	**0**	**0**	**0**	**0**	**LP SH1**	**0**	**0**	**0**	**LPCH1**				
rw	r	r	r	r	r	r	rw	r	r	r	rw				

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LP EN0	**0**	**0**	**0**	**0**	**0**	**0**	**LP SH0**	**0**	**0**	**0**	**LPCH0**				
rw	r	r	r	r	r	r	rw	r	r	r	rw				

域	位	类型	描述
LPCH0, LPCH1	[4:0], [20:16]	rw	**循环 y 通道** 选择输入通道，应使能该通道的 Σ-Δ 循环功能。
0	[7:5]	r	**保留，读出值为 0，应写入 0**
LPSH0, LPSH1	8, 24	rw	**循环 y 采样保持单元** 选择采样保持单元，将所指定的通道分配给该单元。
0	[14:9]	r	**保留，读出值为 0，应写入 0**
LPEN0, LPEN1	15, 31	rw	**循环 y 使能** 0_H 关：标准操作 1_H 开：Σ-Δ 循环活动
0	[23:21]	r	**保留，读出值为 0，应写入 0**
0	[30:25]	r	**保留，读出值为 0，应写入 0**

注： 位域LPSHy和LPCHy一起选择一个单独的与相应的Σ-Δ循环相关联的输入通道。

16.16.9 服务请求寄存器

GxSEFLAG (x = 0 - 1)
源事件标志寄存器 , 组 x **($x * 0400_H + 0588_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**SEV 1**	**SEV 0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	rwh	rwh

域	位	类型	描述
SEV0, SEV1	0, 1	rwh	**源事件 0/1** 0_B 无源事件 1_B 发生了一个源事件
0	[31:2]	r	**保留，读出值为 0，应写入 0**

注： 软件可置位寄存器 GxSEFLAG 中的所有标志，并可通过向相关位写 1 来触发相应的事件。写 0 无效。
软件可通过向寄存器 GxSEFCLR 中相应的位写 1 来清除寄存器 GxREFLAG 中的所有标志。

GxCEFLAG (x = 0 - 1)
通道事件标志寄存器 , 组 x **($x * 0400_H + 0580_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**CEV 7**	**CEV 6**	**CEV 5**	**CEV 4**	**CEV 3**	**CEV 2**	**CEV 1**	**CEV 0**
r	r	r	r	r	r	r	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
CEVy **(y = 0 - 7)**	y	rwh	**通道 y 的通道事件** 0_B 无通道事件 1_B 发生了一个通道事件
0	[31:8]	r	**保留，读出值为 0，应写入 0**

注： 软件可置位寄存器 GxCEFLAG 中的所有标志，并可通过向相关位写 1 到来触发相应的事件。写 0 无效。
软件可通过向寄存器 GxCEFCLR 中的相应位写 1 来清除寄存器 GxREFLAG 中的所有标志。

GxREFLAG (x = 0 - 1)
结果事件标志寄存器 , 组 x **($x * 0400_H + 0584_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV 15	**REV 14**	**REV 13**	**REV 12**	**REV 11**	**REV 10**	**REV 9**	**REV 8**	**REV 7**	**REV 6**	**REV 5**	**REV 4**	**REV 3**	**REV 2**	**REV 1**	**REV 0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
REVy **(y = 0 - 15)**	y	rwh	**结果寄存器 y 的结果事件** 0_B 无结果事件 1_B 新结果被保存到寄存器 GxRESy
0	[31:16]	r	**保留，读出值为 0，应写入 0**

注： 软件可置位寄存器 GxREFLAG 中的所有标志，并可通过向相关位写 1 来触发相应的事件。写 0 无效。
软件可通过向寄存器 GxREFCLR 中的相应位写 1 来清除寄存器 GxREFLAG 中的所有标志。

GxSEFCLR (x = 0 - 1)
源事件标志清除寄存器 , 组 x **(x * 0400_H + 0598_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**SEV 1**	**SEV 0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	w	w

域	位	类型	描述
SEV0, SEV1	0, 1	w	**清除源事件 0/1** 0_B 无操作 1_B 清除 GxSEFLAG 中的源事件标志
0	[31:2]	r	**保留，读出值为 0，应写入 0**

GxCEFCLR (x = 0 - 1)
通道事件标志清除寄存器 , 组 x **(x * 0400_H + 0590_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**CEV 7**	**CEV 6**	**CEV 5**	**CEV 4**	**CEV 3**	**CEV 2**	**CEV 1**	**CEV 0**
r	r	r	r	r	r	r	r	w	w	w	w	w	w	w	w

域	位	类型	描述
CEVy (y = 0 - 7)	y	w	**清除通道 y 的通道事件** 0_B 无操作 1_B 清除 GxCEFLAG 中的通道事件标志
0	[31:8]	r	**保留，读出值为 0，应写入 0**

GxREFCLR (x = 0 - 1)
结果事件标志清除寄存器 , 组 x **(x * 0400_H + 0594_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV 15	**REV 14**	**REV 13**	**REV 12**	**REV 11**	**REV 10**	**REV 9**	**REV 8**	**REV 7**	**REV 6**	**REV 5**	**REV 4**	**REV 3**	**REV 2**	**REV 1**	**REV 0**
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

域	位	类型	描述
REVy (y = 0 - 15)	y	w	**清除结果寄存器 y 结果事件** 0_B 无操作 1_B 清除 GxREFLAG 中的结果事件标志
0	[31:16]	r	**保留，读出值为 0，应写入 0**

GLOBEFLAG
全局事件标志寄存器 **($00E0_H$)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**REV GLB CLR**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**SEV GLB CLR**
r	r	r	r	r	r	r	w	r	r	r	r	r	r	r	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**REV GLB**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**SEV GLB**
r	r	r	r	r	r	r	rwh	r	r	r	r	r	r	r	rwh

域	位	类型	描述
SEVGLB	0	rwh	**源事件 (后台)** 0_B 无源事件 1_B 发生了一个源事件
0	[7:1]	r	**保留，读出值为 0，应写入 0**

域	位	类型	描述
REVGLB	8	rwh	**全局结果事件** 0_B 无结果事件 1_B 有新结果被保存到寄存器 GLOBRES
0	[15:9]	r	**保留，读出值为 0，应写入 0**
SEVGLBCLR	16	w	**清除源事件 (后台)** 0_B 无操作 1_B 清除源事件标志 SEVGLB
0	[23:17]	r	**保留，读出值为 0，应写入 0**
REVGLBCLR	24	w	**清除全局结果事件** 0_B 无操作 1_B 清除结果事件标志 REVGLB
0	[31:25]	r	**保留，读出值为 0，应写入 0**

注： 软件可以置位标志 REVGLB 和 SEVGLB，并可通过向相关位写 1 来触发相应的事件。写 0 无效。
软件可以通过分别向位 REVGLBCLR 和 SECGLBCLR 写 1 来清除这些标志。
将两个位（如SEVGLB 和SEVGLBCLR）同时置1会清除相应标志（如SEVGLB）。

GxSEVNP (x = 0 - 1)
源事件节点指针寄存器，组 x　　(x * 0400_H + $05C0_H$)　　复位值 : $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**SEV1NP**				**SEV0NP**			
r	r	r	r	r	r	r	r	rw				rw			

域	位	类型	描述
SEV0NP, SEV1NP	[3:0], [7:4]	rw	**服务请求节点指针源事件 i[1)]** 连接相应的事件触发信号到一个服务请求线 （节点）。 0000_B选择组 x 的服务请求线 0 … 0011_B选择组 x 的服务请求线 3 0100_B选择共享服务请求线 0 … 0111_B选择共享服务请求线 3 $1xxx_B$ 保留 *注： 对于共享服务请求线，见表16-12 中的公共组。*
0	[31:8]	r	**保留，读出值为 0，应写入 0**

1) 源 0 是一个 8 级队列源，源 1 是一个通道扫描源。

GxCEVNP0 (x = 0 - 1)
通道事件节点指针寄存器 0, 组 x (x * 0400_H + $05A0_H$) **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
CEV7NP				CEV6NP				CEV5NP				CEV4NP			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CEV3NP				CEV2NP				CEV1NP				CEV0NP			
rw				rw				rw				rw			

域	位	类型	描述
CEV0NP, **CEV1NP,** **CEV2NP,** **CEV3NP,** **CEV4NP,** **CEV5NP,** **CEV6NP,** **CEV7NP**	[3:0], [7:4], [11:8], [15:12], [19:16], [23:20], [27:24], [31:28]	rw	**服务请求节点指针通道事件 i** 连接相应的事件触发信号到一个服务请求线 （节点）。 0000_B选择组 x 的服务请求线 0 … 0011_B选择组 x 的服务请求线 3 0100_B选择共享服务请求线 0 … 0111_B选择共享服务请求线 3 $1xxx_B$ 保留 *注： 对于共享服务请求线，见表16-12 中的公共组。*

GxREVNP0 (x = 0 - 1)
结果事件节点指针寄存器 0, 组 x (x * 0400_H + $05B0_H$) **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV7NP				**REV6NP**				**REV5NP**				**REV4NP**			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV3NP				**REV2NP**				**REV1NP**				**REV0NP**			
rw				rw				rw				rw			

域	位	类型	描述
REV0NP, **REV1NP,** **REV2NP,** **REV3NP,** **REV4NP,** **REV5NP,** **REV6NP,** **REV7NP**	[3:0], [7:4], [11:8], [15:12], [19:16], [23:20], [27:24], [31:28]	rw	**服务请求节点指针结果事件 i** 连接相应的事件触发信号与一个服务请求线 （节点）。 0000_B选择组 x 的服务请求线 0 … 0011_B选择组 x 的服务请求线 3 0100_B选择共享服务请求线 0 … 0111_B选择共享服务请求线 3 $1xxx_B$ 保留 *注： 对于共享服务请求线，见表16-12 中的公共组。*

GxREVNP1 (x = 0 - 1)
结果事件结点指针寄存器 1, 组 x (x * 0400_H + $05B4_H$) **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
REV15NP				REV14NP				REV13NP				REV12NP			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REV11NP				REV10NP				REV9NP				REV8NP			
rw				rw				rw				rw			

域	位	类型	描述
REV8NP, **REV9NP,** **REV10NP,** **REV11NP,** **REV12NP,** **REV13NP,** **REV14NP,** **REV15NP**	[3:0], [7:4], [11:8], [15:12], [19:16], [23:20], [27:24], [31:28]	rw	**服务请求节点指针结果事件 i** 连接相应的事件触发信号到一个服务请求线 （节点）。 0000_B选择组 x 的服务请求线 0 … 0011_B选择组 x 的服务请求线 3 0100_B选择共享服务请求线 0 … 0111_B选择共享服务请求线 3 $1xxx_B$ 保留 *注： 对于共享服务请求线，见**表 16-12** 中的公共组。*

GLOBEVNP
全局事件节点指针寄存器 **(0140_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	0	0	0	0	0	0	0	0	0	0	0	REV0NP			
r	r	r	r	r	r	r	r	r	r	r	r	rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	0	0	0	0	0	SEV0NP			
r	r	r	r	r	r	r	r	r	r	r	r	rw			

域	位	类型	描述
SEV0NP	[3:0]	rw	**服务请求节点指针后台源** 连接相应的事件触发信号到一个服务请求线 （节点）。 0000_B选择公共服务请求组 0 的共享服务请求线 0 … 0011_B选择公共服务请求组 0 的共享服务请求线 3 其他：保留 *注： 对于共享服务请求线，见**表16-12** 中的公共组。*
0	[15:4]	r	**保留，读出值为 0，应写入 0**
REV0NP	[19:16]	rw	**服务请求节点指针全局结果** 连接相应的事件触发信号到一个服务请求线 （节点）。 0000_B 选择公共服务请求组 0 的共享服务请求线 0 … 0011_B 选择公共服务请求组 0 的共享服务请求线 3 其他：保留 *注： 对于共享服务请求线，见**表16-12** 中的公共组。*
0	[31:20]	r	**保留，读出值为 0，应写入 0**

GxSRACT (x = 0 - 1)
服务请求软件激活触发器，组 x (x * 0400_H + $05C8_H$) **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**0**	**0**	**0**	**ASSR3**	**ASSR2**	**ASSR1**	**ASSR0**	**0**	**0**	**0**	**0**	**0**	**0**	**AGSR1**	**AGSR0**
r	r	r	r	w	w	w	w	r	r	r	r	r	r	w	w

域	位	类型	描述
AGSRy (y = 0 - 1)	y	w	**激活组服务请求节点 y** 0_B 无操作 1_B 激活相关联的服务请求线

域	位	类型	描述
0	[7:2]	r	**保留，读出值为 0，应写入 0**
ASSRy (y = 0 - 3)	8 + y	w	**激活共享服务请求节点 y** 0_B 无操作 1_B 激活相关联的服务请求线
0	[31:12]	r	**保留，读出值为 0，应写入 0**

16.17 互连

本节描述 XMC1300 中 VADC 模块的具体实现，即与微控制器系统的结合。

16.17.1 产品的具体配置

功能描述部分以通用的形式描述了 A/D 转换器的功能和工作模式。本节概要性地介绍该产品 （XMC1300）中的可用配置。
每个转换器组都配有一个独立的采样保持电路和一个专用的模拟输入多路复用器。

表 16-12 XMC1300 中的通用转换器配置

转换器组	输入通道	可选为备用 GND 的通道	12 位性能	公共服务请求组
G0	0 … 7	8	校准的 采样保持 ADC	C0
G1	0 … 7	8		C0

XMC1300 中的同步组

XMC1300 中的转换器内核可以连接到同步组，以实现对多个输入通道的并行转换。

表 16-13 列出了可以被同步以实现并行转换的内核。

表 16-13　XMC1300 中的同步组

ADC 内核	同步组	由控制输入 CIx 选择的主内核[1]	
		CI0[2]	CI1
ADC00	A	ADC00	ADC01
ADC01	A	ADC01	ADC00

1) 控制输入由寄存器 GxSYNCTR (x = 0 - 1) 中的位域 STSEL 选择。
根据位 EVALRx 选择相应的就绪输入。
2) 控制输入 CI0 总是选择相应 ADC 内核中自己的控制信号。这种选择是针对同步主内核或独立工作的情况。

16.17.2　XMC1300 中的模拟模块连接

VADC 模块接受多个模拟输入信号。模拟输入多路复用器从本产品中的可选信号中选择将要转换的输入通道。

模拟输入通道的确切数量和到端口引脚的可用连接取决于产品型号（见表 16-12）。表 16-14 中列出了所有版本的 XMC1300 的 ADC 模拟输入通道。

表 16-14　XMC1300 中的模拟连接

信号	方向	源 / 目的	描述
电源电压和参考电压			
V_{DDM}	I	VDD	正电源电压
V_{SSM}	I	VSS	负电源电压
V_{AREF}	I	VDD	正模拟参考
V_{AGND}	I	VSS	负模拟参考
组 0 的模拟输入			
G0CH0	I	P2.6	组 0 的模拟输入通道 0
G0CH1	I	P2.8	组 0 的模拟输入通道 1
G0CH2	I	P2.9	组 0 的模拟输入通道 2
G0CH3	I	P2.10	组 0 的模拟输入通道 3
G0CH4	I	P2.11	组 0 的模拟输入通道 4
G0CH5	I	P2.0	组 0 的模拟输入通道 5
G0CH6	I	P2.1	组 0 的模拟输入通道 6
G0CH7	I	P2.2	组 0 的模拟输入通道 7

表 16-14 XMC1300 中的模拟连接 (续表)

信号	方向	源 / 目的	描述
组 1 的模拟输入			
G1CH0	I	P2.8	组 1 的模拟输入通道 0
G1CH1	I	P2.7	组 1 的模拟输入通道 1
G1CH2	I	P2.10	组 1 的模拟输入通道 2
G1CH3	I	P2.11	组 1 的模拟输入通道 3
G1CH4	I	P2.9	组 1 的模拟输入通道 4
G1CH5	I	P2.3	组 1 的模拟输入通道 5
G1CH6	I	P2.4	组 1 的模拟输入通道 6
G1CH7	I	P2.5	组 1 的模拟输入通道 7

16.17.3 XMC1300 中数字模块的连接

VADC 模块接受多个数字输入信号并产生多个输出信号。本节概要介绍这些信号与其它片上模块的连接，以及通过端口引脚与外部资源的连接。

注： 触发和门控的控制位域选择相应的多路复用器输入。值 0000_B ... 1111_B 选择后缀为 -A ... -P 的输入。

表 16-15 XMC1300 中的数字连接

信号	方向	源 / 目的	描述
每组的门控输入			
GxREQGTA	I	CCU40.ST3	门控输入 A
GxREQGTB	I	CCU40.ST2	门控输入 B
GxREQGTC	I	CCU40.ST1	门控输入 C
GxREQGTD	I	CCU40.ST0	门控输入 D
GxREQGTE	I	CCU80.ST3A	门控输入 E
GxREQGTF	I	CCU80.ST3	门控输入 F
GxREQGTG	I	0	门控输入 G
GxREQGTH	I	0	门控输入 H
GxREQGTI	I (s)	保留	门控输入 I
GxREQGTJ	I (s)	保留	门控输入 J
GxREQGTK	I (s)	ERU0.PDOUT2	门控输入 K
GxREQGTL	I (s)	ERU0.PDOUT3	门控输入 L

表 16-15 XMC1300 中的数字连接 (续表)

信号	方向	源 / 目的	描述
GxREQGTM	I	CCU80.ST0	门控输入 M
GxREQGTN	I	CCU80.ST1	门控输入 N
GxREQGTO	I	ERU0.PDOUT0	门控输入 O
GxREQGTP	I	ERU0.PDOUT1	门控输入 E
GxREQGTSEL	O	GxREQTRP[1)]	相应源的所选门控信号
全局后台源的门控输入			
BGREQGTA	I	CCU40.ST3	门控输入 A，后台源
BGREQGTB	I	CCU40.ST2	门控输入 B，后台源
BGREQGTC	I	CCU40.ST1	门控输入 C，后台源
BGREQGTD	I	CCU40.ST0	门控输入 D，后台源
BGREQGTE	I	CCU80.ST3A	门控输入 E，后台源
BGREQGTF	I	CCU80.ST3	门控输入 F，后台源
BGREQGTG	I	0	门控输入 G，后台源
BGREQGTH	I	0	门控输入 H，后台源
BGREQGTI	I (s)	保留	门控输入 I，后台源
BGREQGTJ	I (s)	保留	门控输入 J，后台源
BGREQGTK	I (s)	ERU0.PDOUT2	门控输入 K，后台源
BGREQGTL	I (s)	ERU0.PDOUT3	门控输入 L，后台源
BGREQGTM	I	CCU80.ST0	门控输入 M，后台源
BGREQGTN	I	CCU80.ST1	门控输入 N，后台源
BGREQGTO	I	ERU0.PDOUT0	门控输入 O，后台源
BGREQGTP	I	ERU0.PDOUT1	门控输入 E，后台源
BGREQGTSEL	O	BGREQTRP[1)]	选择的门控信号
每组的触发输入			
GxREQTRA	I	CCU40.SR2	触发输入 A
GxREQTRB	I	CCU40.SR3	触发输入 B
GxREQTRC	I	0	触发输入 C
GxREQTRD	I	0	触发输入 D
GxREQTRE	I	0	触发输入 E
G0REQTRF	I	BCCU0.TRIGOUT0	触发输入 F

表 16-15 XMC1300 中的数字连接 (续表)

信号	方向	源 / 目的	描述
G1REQTRF	I	BCCU0.TRIGOUT1	触发输入 F
GxREQTRG	I	ERU0.IOUT2	触发输入 G
GxREQTRH	I	ERU0.IOUT3	触发输入 H
GxREQTRI	I (s)	CCU80.SR2	触发输入 I
GxREQTRJ	I (s)	CCU80.SR3	触发输入 J
GxREQTRK	I (s)	0	触发输入 K
GxREQTRL	I (s)	0	触发输入 L
GxREQTRM	I	ERU0.IOUT0	触发输入 M
GxREQTRN	I	ERU0.IOUT1	触发输入 N
GxREQTRO	I	POSIF0.SR1	触发输入 O
GxREQTRP	I	GxREQGTSEL[1)]	相应源的所选门控输入的扩展触发信号
GxREQTRSEL	O	-	相应源的所选触发信号
全局后台源的触发输入			
BGREQTRA	I	CCU40.SR2	触发输入 A，后台源
BGREQTRB	I	CCU40.SR3	触发输入 B，后台源
BGREQTRC	I	0	触发输入 C，后台源
BGREQTRD	I	0	触发输入 D，后台源
BGREQTRE	I	0	触发输入 E，后台源
BGREQTRF	I	BCCU0.TRIGOUT0	触发输入 F，后台源
BGREQTRG	I	ERU0.IOUT2	触发输入 G，后台源
BGREQTRH	I	ERU0.IOUT3	触发输入 H，后台源
BGREQTRI	I (s)	CCU80.SR2	触发输入 I，后台源
BGREQTRJ	I (s)	CCU80.SR3	触发输入 J，后台源
BGREQTRK	I (s)	0	触发输入 K，后台源
BGREQTRL	I (s)	0	触发输入 L，后台源
BGREQTRM	I	ERU0.IOUT0	触发输入 M，后台源
BGREQTRN	I	ERU0.IOUT1	触发输入 N，后台源
BGREQTRO	I	POSIF0.SR1	触发输入 O，后台源
BGREQTRP	I	BGREQGTSEL[1)]	后台源的所选门控输入的扩展触发信号

表 16-15 XMC1300 中的数字连接 (续表)

信号	方向	源 / 目的	描述
BGREQTRSEL	O	-	后台源的所选触发信号
系统内部连接			
G0SAMPLE	O	-	指示输入信号采样阶段
G1SAMPLE	O	-	指示输入信号采样阶段
G0ARBCNT	O	CCU40.IN3G	每个仲裁周期输出一个 （计数）脉冲
G1ARBCNT	O	-	每个仲裁周期输出一个 （计数）脉冲
G0SR0	O	NVIC (17)	组 0 的服务请求 0
G0SR1	O	NVIC (18)	组 0 的服务请求 1
G0SR2	O	-	组 0 的服务请求 2
G0SR3	O	-	组 0 的服务请求 3
G1SR0	O	NVIC (19)	组 1 的服务请求 0
G1SR1	O	NVIC (20)	组 1 的服务请求 1
G1SR2	O	-	组 1 的服务请求 2
G1SR3	O	-	组 1 的服务请求 3
C0SR0	O	NVIC (15)	公共块 0 的服务请求 0
C0SR1	O	NVIC (16)	公共块 0 的服务请求 1
C0SR2	O	ERU0.OGU02 ERU0.OGU12	公共块 0 的服务请求 2
C0SR3	O	ERU0.OGU22 ERU0.OGU32	公共块 0 的服务请求 3
EMUX00	O	P0.4 P1.1	外部模拟多路复用器接口 0 的控制
EMUX01	O	P0.3 P1.2	
EMUX02	O	P0.2 P1.3	
EMUX10	O	P1.4	外部模拟多路复用器接口 1 的控制
EMUX11	O	P1.5	
EMUX12	O	保留	
CBFLOUT0	O	POSIF0.IN0C	公共边界标志输出 0
CBFLOUT1	O	POSIF0.IN1C	公共边界标志输出 1
CBFLOUT2	O	POSIF0.IN2C	公共边界标志输出 2

表 16-15 XMC1300 中的数字连接 (续表)

信号	方向	源 / 目的	描述
CBFLOUT3	O	POSIF0.EWHEA	公共边界标志输出 3
G0BFLOUT0	O	ERU0.0A3	组 0 的边界标志 0 输出
G1BFLOUT0	O	ERU0.0B3	组 1 的边界标志 0 输出
G0BFLOUT1	O	ERU0.1A3	组 0 的边界标志 1 输出
G1BFLOUT1	O	ERU0.1B3	组 1 的边界标志 1 输出
G0BFLOUT2	O	ERU0.2A3	组 0 的边界标志 2 输出
G1BFLOUT2	O	ERU0.2B3	组 1 的边界标志 2 输出
G0BFLOUT3	O	ERU0.3A3	组 0 的边界标志 3 输出
G1BFLOUT3	O	ERU0.3B3	组 1 的边界标志 3 输出

1) 内部信号连接。

17 模拟比较器 (ACMP) 和超量程比较器 (ORC)

本章描述模拟比较器 （ACMP）和超量程比较器 （ORC）的控制。

17.1 概述

本节简要介绍模拟比较器和超量程比较器的特性。XMC1300 有 3 个模拟比较器（ACMPx [x=0-2]）和 8 个超量程比较器 （ORCx [x=0-7]）。

17.1.1 特性

模拟比较器和超量程比较器提供下列特性：

- 超量程比较器 (ORC)
  - 对 ADC 模块的模拟输入引脚的过压监测
- 模拟比较器 (ACMP)
  - 监测外部电压电平
  - 可选择低功耗模式
  - 反相输出选项

17.2 模拟比较器 (ACMP)

图 17-1 示出了模拟比较器单元的框图。

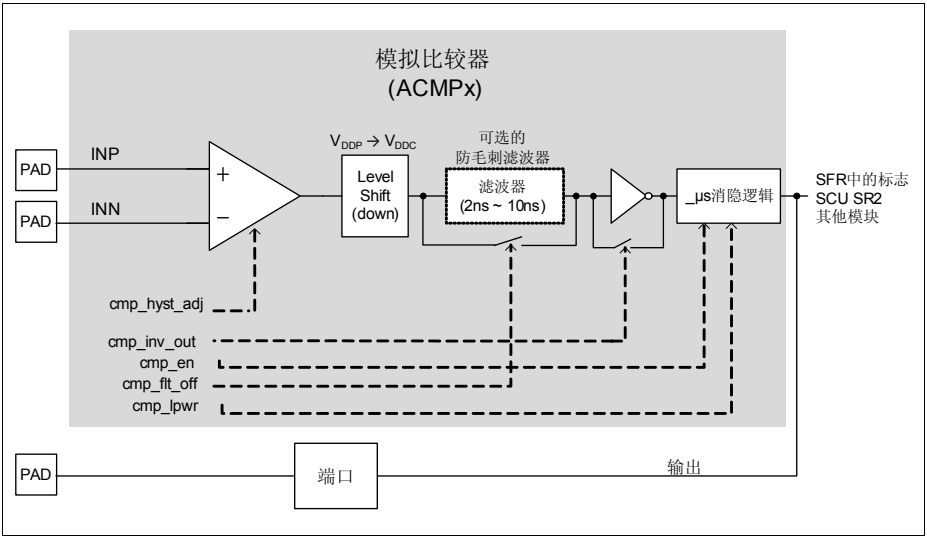


图 17-1 模拟比较器框图

输入 INP 与输入 INN 在焊盘电压域进行比较。比较器输出的数字信号从电源电压电平 V_{DDP} 下移到内核电压电平 V_{DDC}。一个滤波器（如果被使能）用于吸收产生的尖脉冲，该滤波器通过位 ANACMPx.CMP_FLT_OFF 来控制。为了防止比较器在使能后的瞬间产生未定义的状态，模块中增加了一个消隐模块来确保在过渡状态有稳定的输出。消隐时间在几微秒的范围内。借助于位 ANACMPx.CMP_INV_OUT 的帮助，可以将比较器输出反相。比较器的输出可以用来唤醒节电模式下的系统，也可以被发送到一个引脚。

可以通过 ANACMPx 寄存器中的 CMP_EN 位来关闭模拟比较器，以达到节电。

参考电压分压

如果位 REF_DIV_EN 被置 1，一个电阻链 (大约 500kOhm) 将来自 ACMP.REF 输入的参考电压进行分压。根据 **图 17-2**，分压后的电压被传送到引脚 ACMP1.INP。利用位 ANACMP0.ACMP0_SEL 和 ANACMP2.ACMP2_SEL，所有其他比较器都可以由这个分压后的参考电压供电 。

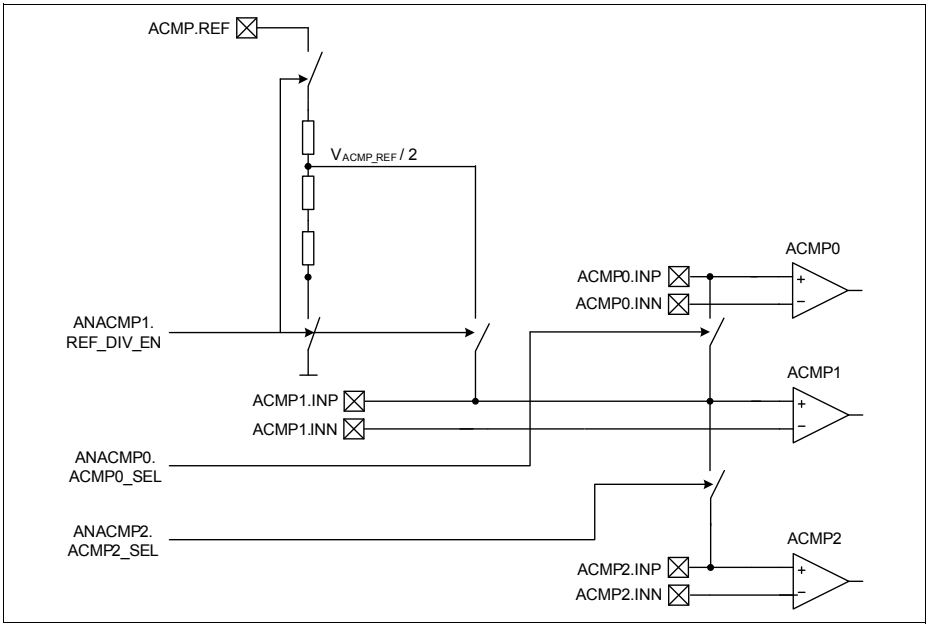


图 17-2　　模拟比较器参考电压分压器功能

低功耗模式

低功耗状态有利于降低总功耗，例如在休眠模式期间即可使用低功耗模式。低功耗模式可通过置位 ANACMPx.LPWR 来使能。在该模式，模拟比较器的性能可能会有所降低。当切换回正常模式时，也要使用消隐时间来确保稳定的输出

17.3 超量程比较器 (ORC)

每个 ADC 通道内都有超量程比较器（ORC），当电压超量程条件发生时，超量程比较器就会触发其他模块或触发中断。当输入通道的电压上升到高于 V_{DDP} 电平的上限或输入通道下降到低于 V_{DDP} 电平的某个电压时就会发生这种情况。

超量程比较器连接到 ERU0 模块。所有的超量程事件都被分配到一个中断节点。

使用超量程比较器前，必须通过置位 **ORCCTRL**.ENORCx 将其使能，并且通过置位 **ORCCTRL**.CNFx 将其配置为检测电压高于还是低于 V_{DDP} (见 **图 17-3**)。

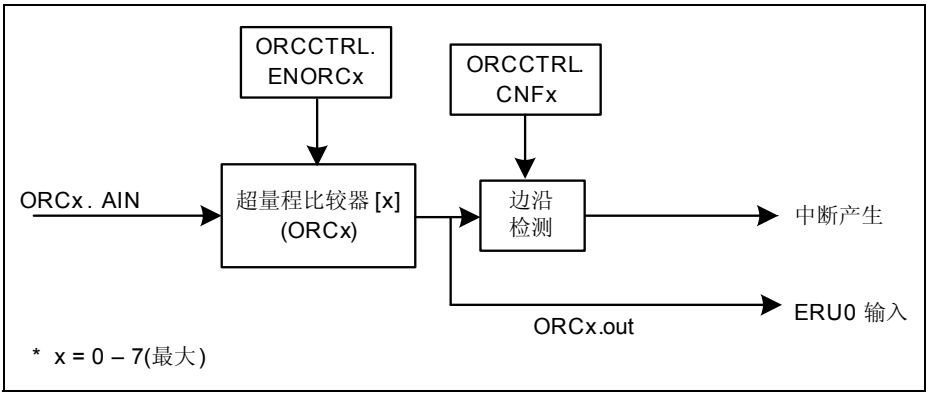


图 17-3 超量程比较器

当发生一个电压超量程事件时，该事件通过 SCU.SR2 置位一个中断请求，并通过输出 ORC (ORCx.out) 触发 ERU0。

在 XMC1300 中，ORCx.AIN[x=0-7] 分别连接到 P2.2 - P2.9 。

17.4 服务请求的产生

ACMP 和 ORC 分别有一个服务请求输出。它们组合成一个输出信号并通过 SCU.SR2 连接到嵌套向量中断控制器 （NVIC）的一个中断节点。

服务请求处理是 SCU 模块中的 GCU 块的一部分。更详细的信息参见 SCU 一章中 GCU 一节的 “ 服务请求 ” 描述。

17.5 调试行为

ACMPx 和 ORCx 不受使用外部调试探针所执行的调试活动的影响。

17.6 寄存器

ACMP 寄存器可通过 APB 总线访问，ORC 可通过 AHB 总线访问。绝对寄存器地址通过下面的加法计算：

模块基地址 + 偏移地址

对 ACMP/ORC 的 SFR 执行下面的访问会导致 AHB/APB 错误响应：

- 读或写未定义的地址
- 写只读寄存器

表 17-1 寄存器地址空间

模块	基地址	结束地址	备注
COMPARATOR	$4001\ 0000_H$	$4001\ FFFF_H$	系统控制单元寄存器

表 17-2 寄存器一览表

简称	描述	偏移地址 .[1]	访问方式		描述见
			读	写	
ORCx 寄存器					
ORCCTRL	超量程比较器控制寄存器	0500_H	U, PV 32	U, PV 32	页 17-5
ACMPx 寄存器					
ANACMP0	模拟比较器 0 控制寄存器	$105C_H$	U, PV	U, PV	页 17-6
ANACMP1	模拟比较器 1 控制比较器	1060_H	U, PV	U, PV	页 17-6
ANACMP2	模拟比较器 2 控制比较器	1064_H	U, PV	U, PV	页 17-6

1) 绝对寄存器地址计算如下：
模块基地址 + 偏移地址 (显示在此列)

17.6.1 ORC 寄存器

ORCCTRL

超量程比较器控制寄存器

寄存器使能超量程比较器并选择标志寄存器是上升沿触发还是下降沿触发。

ORCCTRL
超量程比较器控制寄存器 **(0500_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								CNF7	CNF6	CNF5	CNF4	CNF3	CNF2	CNF1	CNF0
r								rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								ENORC7	ENORC6	ENORC5	ENORC4	ENORC3	ENORC2	ENORC1	ENORC0
r								rw	rw	rw	rw	rw	rw	rw	rw

域	位	类型	描述
ENORCx **(x = 0 - 7)**	x	rw	**使能超量程比较器 x** 该位定义是否使能相应模拟输入通道内的超量程比较器。 0_B 禁止超量程比较器。 1_B 使能超量程比较器。
CNFx **(x = 0 - 7)**	x+16	rw	**超量程比较器标志 x** 该位为超量程比较器标志寄存器选择 CHx 是上升沿触发还是下降沿触发。 0_B 下降沿触发超量程事件寄存器。 1_B 上升沿触发超量程事件寄存器。
0	[15:8], [31:24]	r	**保留** 读出值为 0，应写入 0。

17.6.2 ACMP 寄存器

ANACMP0

模拟比较器 0 控制寄存器

ANACMP0
模拟比较器 0 控制寄存器 **($105C_H$)** **复位值：0020_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP_OUT	0						CMP_LPWR	0	ACMP0_SEL	CMP_HYST_ADJ		CMP_INV_OUT	0	CMP_FLT_OFF	CMP_EN
rh	r						rw	r	rw	rw		rw	r	rw	rw

域	位	类型	描述
CMP_EN	0	rw	**比较器使能** 0B **CMP_EN**，禁止比较器 1B **CMP_DIS**，使能比较器
CMP_FLT_OFF	1	rw	**禁止比较器滤波器** 如果置位，比较器毛刺滤波器关闭。 0B **FIL_ON**，滤波器激活 1B **FIL_OFF**，滤波器关闭 (防止滤波器延迟)
CMP_INV_OUT	3	rw	**反相比较器输出** 比较器输出被简单地取反 0B **INV_OFF**，比较器信号不取反 1B **INV_ON**，比较器信号被取反
CMP_HYST_ADJ	5:4	rw	**比较器滞后调整** 为降低对噪声的敏感度，可选择一个滞后电压。该功能可通过写操作关闭 00B **HYS_OFF**，关闭比较器滞后功能 01B **HYS1**，滞后电压 = 10mV 10B **HYS2**，滞后电压 = 15mV 11B **HYS3**，滞后电压 = 20mV
ACMP0_SEL	6	rw	**连接 ACMP0.INP 到 ACMP1.INP** 一个内部开关将比较器焊盘 ACMP0.INP 和焊盘 ACMP1.INP 连接在一起。两个输入焊盘中只有一个应该由电压源驱动。两个焊盘之间的时间延迟是由开关的阻抗引起的。当位 ANACMP1.REF_DIV_EN 置 1 时，分压后的参考电压不仅加到 ACMP1.INP，还加到 ACMP0.INP。 0B **OFF**， ACMP0.INP 未连接 1B **ON**， ACMP0.INP 连接到 ACMP1.INP

域	位	类型	描述
CMP_LPWR	8	rw	**低功耗模式** 如果被使能，所有 3 个模拟比较器单元都被设为低功耗模式。如果该位的逻辑电平改变，内核必须消隐比较器输出信息一段时间。 0B **HPM**，高功耗模式 1B **LPM**，低功耗模式
CMP_OUT	15	rh	**比较器输出监视位** 该位对应比较器输出的状态 0B **OUT0**，状态 “Vminus > Vplus” 1B **OUT1**，状态 “Vminus < Vplus”
0	2, 7, 14:9	r	保留 读出值为 0，应写入 0。

ANACMP1

模拟比较器 1 控制寄存器

ANACMP1

模拟比较器 1 控制寄存器 **(1060_H)** **复位值 : 0020_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP_OUT	0								REF_DIV_EN	CMP_HYST_ADJ		CMP_INV_OUT	0	CMP_FLT_OFF	CMP_EN
rh	r								rw	rw		rw	r	rw	rw

域	位	类型	描述
CMP_EN	0	rw	**比较器使能** 0B **CMP_EN**，禁止比较器 1B **CMP_DIS**，使能比较器
CMP_FLT_OFF	1	rw	**禁止比较器过滤器** 如果置位，比较器毛刺滤波器关闭。 0B **FIL_ON**，滤波器激活 1B **FIL_OFF**，滤波器关闭 (防止滤波器延迟)
CMP_INV_OUT	3	rw	**反相比较器输出** 比较器输出被简单地取反 0B **INV_OFF**，比较器信号不取反 1B **INV_ON**，比较器信号被取反

域	位	类型	描述
CMP_HYST_ADJ	5:4	rw	**比较器滞后调整** 为降低对噪声的敏感度，可选择一个滞后电压。该功能可通过写操作关闭。 00B **HYS_OFF**，关闭比较器滞后功能 01B **HYS1**，滞后电压 = 10mV 10B **HYS2**，滞后电压 = 15mV 11B **HYS3**，滞后电压 = 20mV
REF_DIV_EN	6	rw	**电阻分压器使能，参考电压加到 ACMP1** 分压器参考电压被加到 ACMP1 的正输入。 0B **OFF**，不连接电阻 1B **ON**，使能分压器电阻，参考电压加到 ACMP1.INP
CMP_OUT	15	rh	**比较器输出监视位** 该位对应比较器的输出状态 0B **OUT0**，状态 “Vminus > Vplus” 1B **OUT1**，状态 “Vminus < Vplus”
0	2, 14:7	r	保留 读出值为 0，应写入 0。

ANACMP2

模拟比较器 2 控制寄存器

ANACMP2

模拟比较器 2 控制寄存器 **(1064_H)** **复位值 : 0020_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CMP_OUT	0								ACMP2_SEL	CMP_HYST_ADJ		CMP_INV_OUT	0	CMP_FLT_OFF	CMP_EN
rh	r								rw	rw		rw	r	rw	rw

域	位	类型	描述
CMP_EN	0	rw	**比较器使能** 0B **CMP_EN**，禁止比较器 1B **CMP_DIS**，使能比较器

域	位	类型	描述
CMP_FLT_OFF	1	rw	**禁止比较过滤器** 如果置位，比较器毛刺滤波器关闭 0B **FIL_ON**，滤波器激活 1B **FIL_OFF**，滤波器关闭 (防止滤波器延迟)
CMP_INV_OUT	3	rw	**反相比较器输出** 比较器输出被简单地取反 0B **INV_OFF**，比较器信号不取反 1B **INV_ON**，比较器信号被取反
CMP_HYST_ADJ	5:4	rw	**比较器滞后调整** 为降低对噪声的敏感度，可以选择一个滞后电压。该功能可通过写操作关闭。 00B **HYS_OFF**，关闭比较器滞后功能 01B **HYS1**，滞后电压 = 10mV 10B **HYS2**，滞后电压 = 15mV 11B **HYS3**，滞后电压 = 20mV
ACMP2_SEL	6	rw	**连接 ACMP2.INP 到 ACMP1.INP** 一个内部开关将比较器焊盘 ACMP2.INP 和焊盘 ACMP1.INP 连接在一起。两个输入焊盘中只有一个应该由电压源驱动。两个焊盘之间的时间延迟是由开关的阻抗引起的。当位 ANACMP1.REF_DIV_EN 置 1 时，分压后的参考电压不仅加到 ACMP1.INP，还加到 ACMP2.INP 0B **OFF**， ACMP2.INP 未连接 1B **ON**， ACMP2.INP 连接到 ACMP1.INP
CMP_OUT	15	rh	**比较器输出监视位** 该位对应比较器输出的状态 0B **OUT0**，状态 “Vminus > Vplus” 1B **OUT1**，状态 “Vminus < Vplus”
0	2, 14:7	r	保留 读出值为 0，应写入 0。

17.7 互连

ACMP 和 ORC 可连接到端口、 ERU0 和 BCCU0

表 17-3 模拟比较器引脚连接

输入 / 输出	I/O	连接到	描述
ACMP0.INN	I	P2.8	ACMP0 的 “-” 输入
ACMP0.INP	I	P2.9	ACMP0 的 “+” 输入
ACMP1.INN	I	P2.6	ACMP1 的 “-” 输入
ACMP1.INP	I	P2.7	ACMP1 的 “+” 输入
ACMP2.INN	I	P2.2	ACMP2 的 “-” 输入
ACMP2.INP	I	P2.1	ACMP2 的 “+” 输入
ACMP.REF	I	P2.11	ACMP 的参考输入
ACMP0.OUT	O	P0.10 P2.10 ERU0.0A0 BCCU0.IN5	ACMP0 的输出
ACMP1.OUT	O	P1.0 ERU0.1A0 BCCU0.IN0 P2.5.HW0 拉控制	ACMP1 的输出
ACMP2.OUT	O	P0.5 P1.2 ERU0.2A0 BCCU0.IN3 P2.3.HW0 拉控制	ACMP2 的输出

表 17-4 超量程比较器引脚连接

输入 / 输出	I/O	连接到	描述
ORC0.AIN	I	P2.2	ORC0 的模拟输入
ORC1.AIN	I	P2.3	ORC1 的模拟输入
ORC2.AIN	I	P2.4	ORC2 的模拟输入
ORC3.AIN	I	P2.5	ORC3 的模拟输入
ORC4.AIN	I	P2.6	ORC4 的模拟输入
ORC5.AIN	I	P2.7	ORC5 的模拟输入
ORC6.AIN	I	P2.8	ORC6 的模拟输入

表 17-4 超量程比较器引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
ORC7.AIN	I	P2.9	模拟输入 ORC7
ORC0.OUT	O	ERU0.0B2	ORC0 的输出
ORC1.OUT	O	ERU0.1B2	ORC1 的输出
ORC2.OUT	O	ERU0.0A2 ERU0.2B2	ORC2 的输出
ORC3.OUT	O	ERU0.1A2	ORC3 的输出
ORC4.OUT	O	ERU0.2A2	ORC4 的输出
ORC5.OUT	O	ERU0.3A2	ORC5 的输出
ORC6.OUT	O	ERU0.3B2	ORC6 的输出
ORC7.OUT	O	ERU0.3A0	ORC7 的输出

18 温度传感器 (TSE)

本章描述温度传感器 （TSE）的控制。

18.1 概述

温度传感器（TSE）产生直接指示当前温度的测量结果。测量结果通过位域 ANATSEMON.TSE_MON 显示。用户获取该值后可利用一个公式 （在目标电气参数一章定义）来计算实际的硅片温度。在使用 TSE 之前，必须通过位 ANATSECTRL.TSE_EN 将其使能。

当测量结果准备好时， SRRAW.TSE_DONE 标志被置 1，。如果通过 SRMSK.TSE_DONE 位使能了 TSE 测量完成中断，则此时会触发中断。测量完成后，结果保存到 TSE_MON， TSE 继续进行下一次测量。当 TSE_EN 被清零时，测量被禁止。因此，读 TSE_MON 的值可以得到最新的温度。

当 TSE 温度测量结果高于和 / 或低于在位域 ANATSEIH.TSE_IH 和 ANATSEIL.TSE_IL 中分别配置的阈值时， TSE 能产生中断请求。数字比较器将实际测量结果与所配置的极限值进行比较，如果测量值超出有效范围则触发中断。比较结果用 SRRAW.TSE_HIGH 和 TSE_LOW 标志指示。用来计算上限和下限温度极限值的公式在数据手册中定义。

在公式中有 3 个常量：k1（16 位）、k2（32 位）、k3（16 位），这些值被保存在 Flash 配置页。

18.2 服务请求产生

TSE 有一个服务请求输出，可用于 TSE Done、TSE HIGH 和 TSE Low 事件。它们与来自其他模块的事件组合成单一输出信号，并通过 SCU.SR1 连接到嵌套向量中断控制器（NVIC）中的一个中断节点。

服务请求处理是 SCU 模块中的 GCU 块的一部分。更详细的信息参见 SCU 一章中 GCU 一节的 “ 服务请求 ” 描述。

18.3 寄存器

TSE 寄存器可通过 APB 总线访问。绝对寄存器的地址通过下面的加法计算：

模块基地址 + 偏移地址

对 TSE 的 SFR 执行下面的访问会导致 AHB/APB 错误响应：

- 读或写未定义地址
- 写只读寄存器

表 18-1 寄存器地址空间

模块	起始地址	结束地址	备注
TSE	$4001\ 0000_H$	$4001\ FFFF_H$	系统控制单元寄存器

表 18-2　寄存器一览表

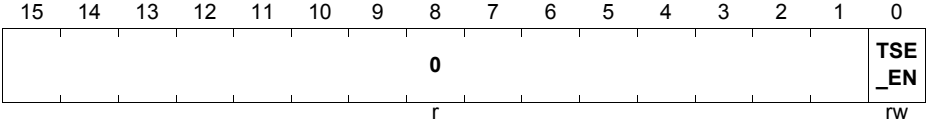
简称	描述	偏移地址	访问模式		描述见
			读	写	
ANATSECTRL	温度传感器控制寄存器	0024_H	U, PV	U, PV	页 18-2
ANATSEIH	温度传感器高温中断寄存器	0030_H	U, PV	U, PV	页 18-3
ANATSEIL	温度传感器低温中断寄存器	0034_H	U, PV	U, PV	页 18-3
ANATSEMON	温度传感器计数器 2 监测寄存器	0040_H	U, PV	U, PV	页 18-4

18.3.1　寄存器

ANATSECTRL

温度传感器控制寄存器

ANATSECTRL
温度传感器控制寄存器　　(1024$_H$)　　复位值 : 0000$_H$

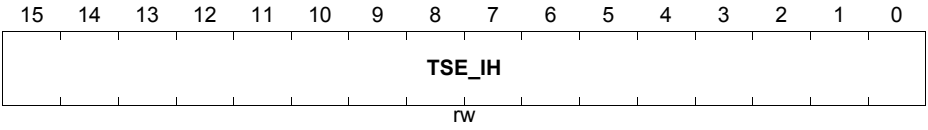


域	位	类型	描述
TSE_EN	0	rw	**温度传感器使能** 0B　禁止温度传感器 1B　使能温度传感器
0	15:1	r	**保留** 读出值为 0，应写入 0。

ANATSEIH

温度传感器高温中断寄存器

ANATSEIH
温度传感器高温中断寄存器 **(1030$_H$)** **复位值 : 0000$_H$**

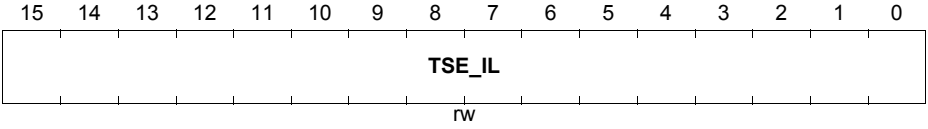


域	位	类型	描述
TSE_IH	15:0	rw	**高温中断的计数值** ANATSEIH 的值与 ANATSEMON （计数器值）比较。如果 ANATSE_MON < ANATSEIH，则触发高温中断。 比较结果可以通过 SCU_SRRAW.TSE_HIG 查看。

ANATSEIL

温度传感器低温中断寄存器

ANATSEIL
温度传感器低温中断寄存器 **(1034$_H$)** **复位值 : FFFF$_H$**



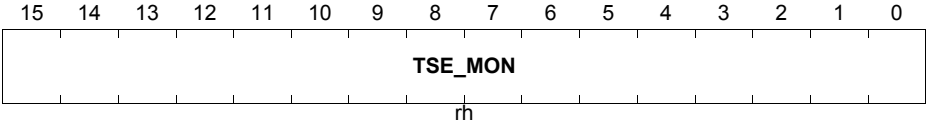
域	位	类型	描述
TSE_IL	15:0	rw	**低温中断的计数器值** ANATSEIL 的值与 ANATSEMON （计数器值）比较。如果 ANATSEMON > ANATSEIL，则触发低温中断。 比较结果可以通过 SCU_SRRAW.TSE_LOW 查看。

ANATSEMON

温度传感器计数器 2 监视寄存器

ANATSEMON

温度传感器计数器 2 监视寄存器　　(1040$_H$)　　复位值 : 0000$_H$



域	位	类型	描述
TSE_MON	15:0	rh	**监视器计数器 2 的值；由 TSE_DONE 加载** 每当发生测量完成事件后（通过位 SCU_SRRAW.TSE_DONE 置 1 指示），测量结果就保存到该寄存器中。

工业控制外设

19 捕获比较单元 4 (CCU4)

对于需要使用通用定时器来完成信号监视 / 调理以及脉冲宽度调制信号产生的系统，CCU4 外设是一个主要的组件。使用 CCU4 所提供的功能，可以很容易实现一些电源电子控制系统，如开关电源或不间断电源。

CCU4 内部的模块化结构使其成为一个软件友好的系统，方便进行快速代码开发和应用程序之间的移植。

表 19-1 缩略语表

PWM	脉冲宽度调制
CCU4x	捕获比较单元 4 模块实例 x
CC4y	捕获比较单元 4 定时器片实例 y
ADC	模 / 数转换器
POSIF	位置接口外设
SCU	系统控制单元
f_{ccu4}	CCU4 模块时钟频率
f_{tclk}	CC4y 定时器时钟频率

注： 寄存器中的小写字母 "y" 或 "x" 代表一个索引。

19.1 概述

每个 CCU4 模块由 4 个完全相同的 16 位捕获 / 比较定时器片 (CC4y) 构成。每个定时器片都可以工作在比较模式或捕获模式。在比较模式，只有一个比较通道可用；而在捕获模式，最多可以并行使用四个捕获寄存器。

每个 CCU4 模块有 4 条服务请求线，每个定时器片都有一个专用的输出信号，这就能够产生最多 4 个独立的 PWM 信号。

也可以将定时器片直接级联，实现最多 64 位的定时操作。这就提供了一种灵活的测频、倍频和脉宽调制方案。

每个定时器片都有一个可编程的功能输入选择器，该选择器能够提供最多 9 种功能，它不需要因为输入端口的可用性而对所有资源进行映射。

CCU4 与其他许多模块之间的内建连接，使实现灵活的数字电机控制回路成为可能。例如，使用霍尔传感器监测，或与编码器直接连接。

19.1.1 特性

CCU4 模块特性

每个 CCU4 由 4 个定时器片组合而成，这些定时器片可以独立工作在比较模式或捕获模式。每个定时器片都有一个用于产生 PWM 信号的专用输出。

CCU4 的所有 4 个定时器片 CC4y 都具有完全相同的可用功能和工作模式。这就使得在实现不同的软件程序时可以避免对所用 CCU4 具体资源的依赖。

这 4 个定时器片之间还有内建连接，可以实现简单的定时器级联和顺序操作。

一般特性

- 16 位定时器单元
- 每个定时器片都有捕获模式和比较模式
  - 捕获模式下有 4 个捕获寄存器
  - 比较模式下有一个比较通道
- 可编程的输入低通滤波器
- 内建定时器级联
  - 32、 48 或 64 位宽
- 周期值和比较值的映射传送
- 可编程的时钟预分频器
- 标准定时器模式
- 门控定时器模式
- 三种计数方式
  - 中心对齐
  - 边沿对齐
  - 单次方式
- PWM 信号产生
- TRAP 功能
- 启停可由外部事件控制
- 对外部事件计数
- 每个 CCU4 有 4 个专用的服务请求线

其他特性

- 外部调制功能
- 由外部事件控制加载
- 抖动 PWM
- 浮动预分频器
- 输出状态可被外部事件覆盖
- 合适而又灵活的模块连接：
  - 电机和电源转换应用
  - 大量的信号调节可能性

CCU4 特性与应用的对应关系

表 19-2 概括了 CCU4 单元的主要特性与最常见应用的对应关系。

表 19-2 应用概览

特性	应用
4 个独立的定时器单元	独立的 PWM 信号产生： • 多个降压 / 升压转换器控制 (使用独立的频率) • 每个定时器有不同的工作模式，提高资源优化程度 • 最多可实现 2 个半桥控制 • 多个零电压开关 (ZVS) 转换器控制，易与 ADC 通道连接。
级联的定时器单元	易于实现可多达 64 位的定时器扩展： • 高动态触发信号捕获 • 高动态信号测量
抖动 PWM	产生极小的 PWM 频率或占空比： • 在低速控制回路应用中可避免频率或占空比的大步距调节 • 提高 PWM 信号随时间的分辨率
浮动预分频器	自动进行控制信号测量： • 减少监测高动态或未知信号的软件工作 • 可仿真一个大于 16 位的定时器用于系统控制
每个定时器可通过外部信号实现最多 9 种功能	灵活的资源优化： • 所有外部功能总是可用 • CCU4 内部可实现多种配置，例如，一个定时器工作在捕获模式，而另一个定时器工作在比较模式
4 条专用的服务请求线	专门用于： • 向微处理器发出中断 • 实现外设间的灵活连接，例如 ADC 触发。
与其它模块的连接	灵活的配置适合： • 霍尔传感器反馈 / 监测 • 电机编码器反馈 / 监测 • PWM 信号并行调制 • 灵活的信号调理

19.1.2 框图

每个 CCU4 定时器片可以独立于其他片工作在所有可用模式。每个定时器片都包含一个专用输入选择器，用于那些需连接外部事件的功能，还有一个用于 PWM 信号产生的专用比较输出信号。

内建的定时器级联功能仅限于相邻的定时器片之间，例如， CC40/CC41。诸如 CC40/CC42 或 CC40/CC43 之类的定时器片级联组合是不可能的。

每个定时器片自身的服务请求 (每个定时器片有 4 个服务请求线) 被多路复用到 4 条模块服务请求线，如图 19-1 所示。

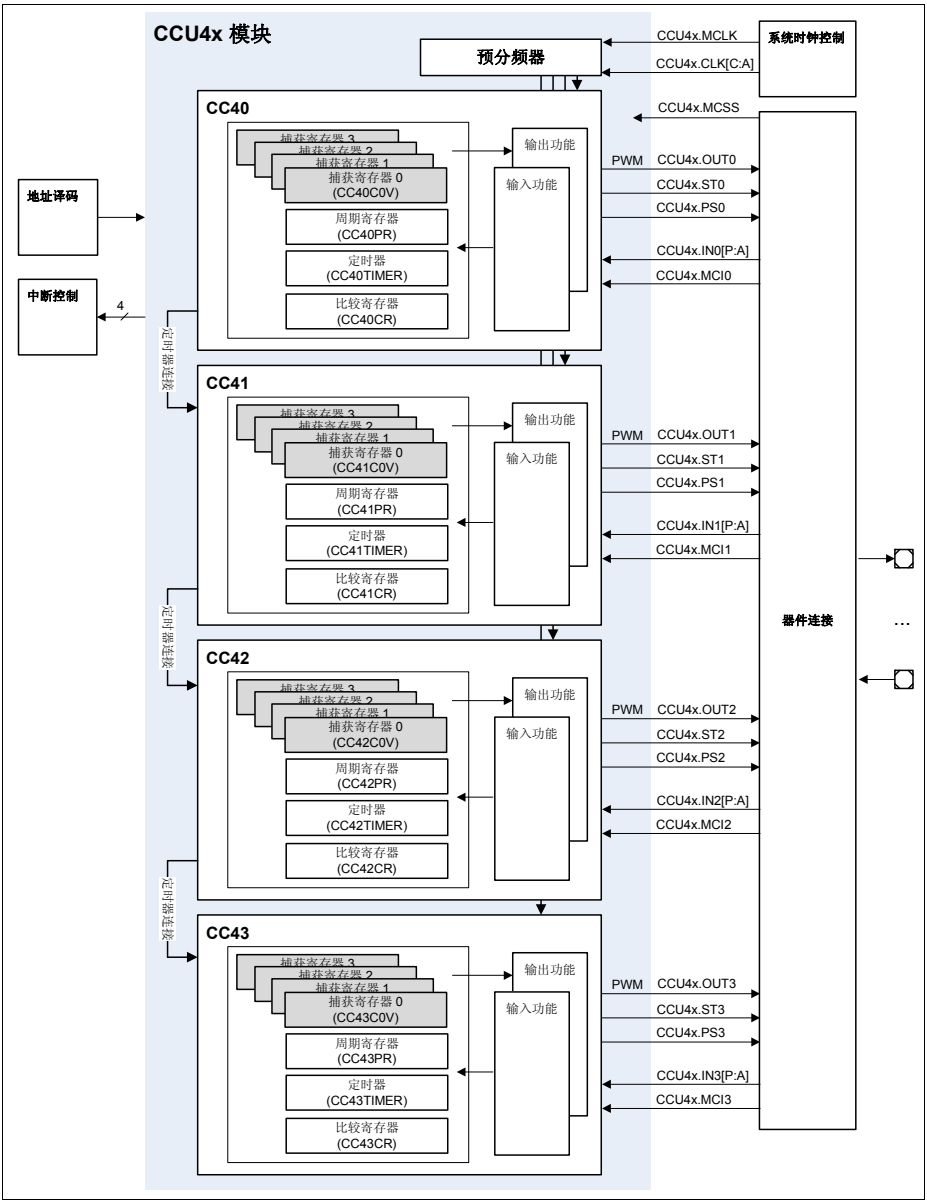

CCU4x 模块
预分频器
CCU4x.MCLK
CCU4x.CLK[C:A]
系统时钟控制
CCU4x.MCSS
CC40
CC41
CC42
CC43
捕获寄存器 3
捕获寄存器 2
捕获寄存器 1
捕获寄存器 0
(CC40C0V)
(CC41C0V)
(CC42C0V)
(CC43C0V)
周期寄存器
(CC40PR)
(CC41PR)
(CC42PR)
(CC43PR)
定时器
(CC40TIMER)
(CC41TIMER)
(CC42TIMER)
(CC43TIMER)
比较寄存器
(CC40CR)
(CC41CR)
(CC42CR)
(CC43CR)
输出功能
输入功能
PWM
CCU4x.OUT0
CCU4x.ST0
CCU4x.PS0
CCU4x.IN0[P:A]
CCU4x.MCI0
CCU4x.OUT1
CCU4x.ST1
CCU4x.PS1
CCU4x.IN1[P:A]
CCU4x.MCI1
CCU4x.OUT2
CCU4x.ST2
CCU4x.PS2
CCU4x.IN2[P:A]
CCU4x.MCI2
CCU4x.OUT3
CCU4x.ST3
CCU4x.PS3
CCU4x.IN3[P:A]
CCU4x.MCI3
地址译码
中断控制
4
定时器级联
器件连接
...

图 19-1　　CCU4 框图

19.2 功能描述

19.2.1 CC4y 概述

一个 CCU4 片的输入通路由一个选择器 (见 **19.2.2 节**) 和一个连接矩阵单元 (见 **19.2.3 节**) 组成。输出通路包含一个服务请求控制单元、一个定时器级联单元和两个直接控制每个特定定时器片输出信号状态的单元 (用于陷阱功能和调制处理)，如**图 19-2** 所示。

在比较模式，定时器内核由一个 16 位的计数器、一个周期寄存器和一个比较寄存器构成；在捕获模式，定时器内核最多由 4 个比较寄存器构成。

在比较模式，周期寄存器设置最大计数值，而比较通道控制着定时器片的专用比较输出的主动 / 被动状态。

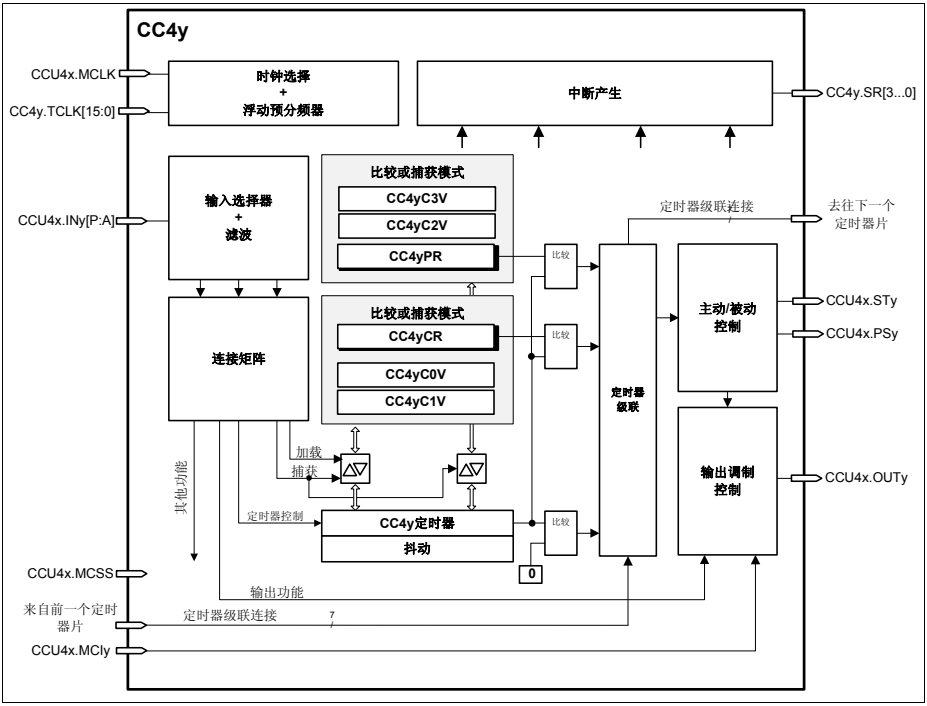


图 19-2 CCU4 定时器片框图

除了第一个 CCU4 片以外，每个 CCU4 片都包含 6 个专用输入和输出，用于内建定时器级联功能。

在 CCU4 的边界看不到的输入和输出被命名为 CC4y.<name>，而 CCU4 模块的输入和输出被描述为 CCU4x.<signal_name>y (变量 y 表示目标定时器片)。

表 19-3　　CCU4 定时器片引脚描述

引脚	I/O	描述
CCU4x.MCLK	I	模块时钟
CC4y.TCLK[15:0]	I	来自预分频器的时钟
CCU4x.INy[P:A]	I	定时器片功能输入 (用于控制通过定时器片外部事件连接的功能)
CCU4x.MCIy	I	多通道模式输入
CCU4x.MCSS	I	多通道映射传送触发信号
CC4y.SR[3...0]	O	定时器片服务请求线
CC4x.STy	O	定时器片比较状态值
CCU4x.PSy	O	多通道序列更新触发信号
CCU4x.OUTy	O	定时器片专用输出引脚

注：

1. *内核的状态位输出 CCU4x.STy 被延长一个内核时钟周期。*
2. *内核输出的服务请求信号被延长一个内核时钟周期。*
3. *输出信号 CCU4x.STy 的最大频率是模块时钟的 4 分频。*

根据所选择的工作模式，片定时器可以向上或向下计数。实际的计数方向在一个方向标志中保存。

定时器与两个独立的比较器相连，其中一个用于周期匹配，另一个用于比较匹配。用于周期匹配和比较匹配的寄存器可以通过编程作为捕获寄存器使用，实现对外部事件的连续捕获。

在标准边沿对齐计数方式下，每当计数值与周期寄存器中设定的周期值相匹配时，计数器被清为 0000_H 。在中心对齐模式，在定时器达到周期值之后，计数方向从向上计数变为向下计数。周期寄存器和比较寄存器各有一个映射寄存器，使用这些映射寄存器可以在运行时修改 PWM 的周期和占空比。

计数器还有单次模式，在该模式下，计数器达到周期寄存器中设定的值后会停止计数。

计数器的启动和停止可以通过软件访问或一个可编程的输入引脚来控制。

像加载、计数方向 (向上 / 向下)、陷阱和输出调制这样的功能也可以用外部事件控制，参见 **19.2.3 节**。

19.2.2 输入选择器

这是定时器片输入通路的第一个单元，用于选择使用哪个输入来控制可用的外部功能。

在该块内部，用户还可以对信号进行低通滤波，选择外部信号的有效边沿或有效电平，如图 19-3 所示。

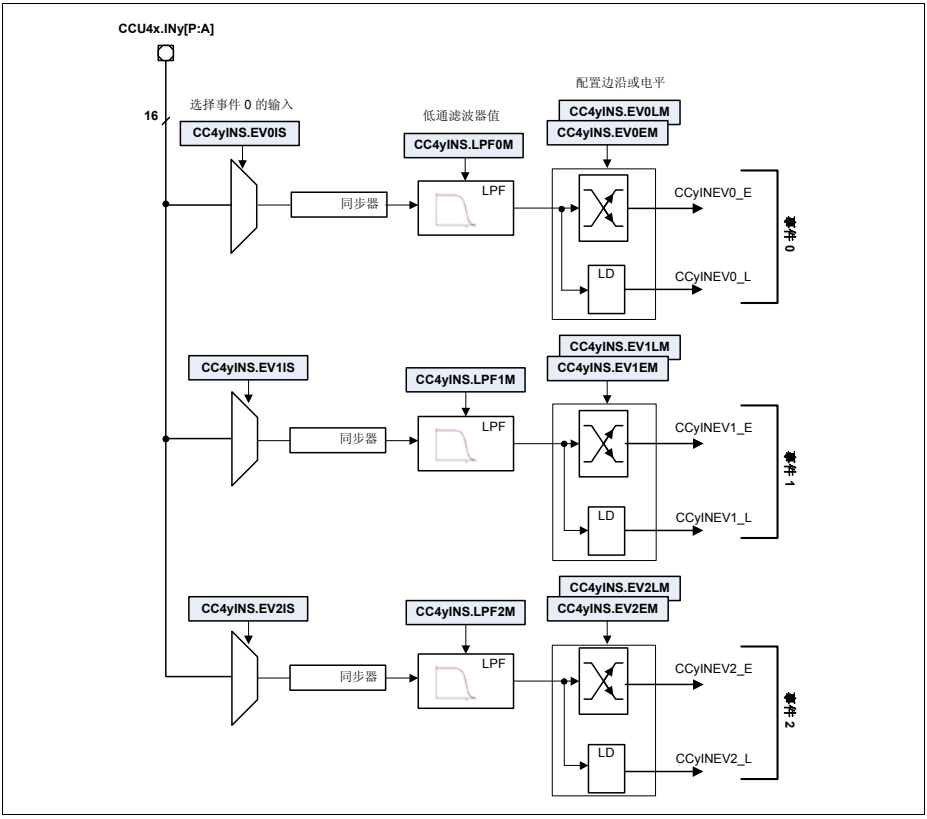


图 19-3 定时器片输入选择器框图

用户可以选择 CCU4x.INy[P:A] 输入中的任何一个作为事件源。

在该单元的输出，用户可以选择三个事件，它们可被配置为上升沿有效、下降沿有效、两个边沿都有效或电平有效。然后，这些选定的事件可以被映射到多种功能。

注意，每个被译码的事件包括两个输出，一个是边沿有效，一个是电平有效。这是因为有些功能如计数、捕获或加载是边沿敏感的，而定时器门控或向上 / 向下计数选择是电平有效的。

19.2.3 连接矩阵

连接矩阵将来自输入选择器的事件映射到多个用户配置的功能，如图 19-4 所示。在连接矩阵中可以使能下述功能：

表 19-4 连接矩阵的可用功能

功能	简要描述	映射到 图 19-4
启动	边沿信号，用于启动定时器	CCystrt
停止	边沿信号，用于停止定时器	CCystp
计数	边沿信号，用于对事件计数	CCycnt
向上 / 向下	电平信号，用于选择向上或向下计数方向	CCyupd
捕获 0	边沿信号，用于触发一次捕获到捕获寄存器 0 和捕获寄存器 1	CCycapt0
捕获 1	边沿信号，用于触发一次捕获到捕获寄存器 2 和捕获寄存器 3	CCycapt1
门控	电平信号，用于门控定时器时钟	CCygate
加载	边沿信号，用于将比较寄存器中的值加载到定时器	CCyload
陷阱	电平信号，用于故障保护操作	CCytrap
调制	电平信号，用于调制或清除输出	CCymod
状态位覆盖	状态位，将被一个输入值覆盖	CCyoval 用作覆盖值 CCyoset 用作触发信号

连接矩阵内部还有一个实现内建定时器级联的单元。这种级联能使被级联的定时器片之间实现完全同步的定时操作，也可以实现捕获操作和加载操作的完全同步。定时器片的级联通过 **CC4yCMC**.TCE 位域来实现。关于级联功能的详细描述，请参见 **19.2.9 节**。

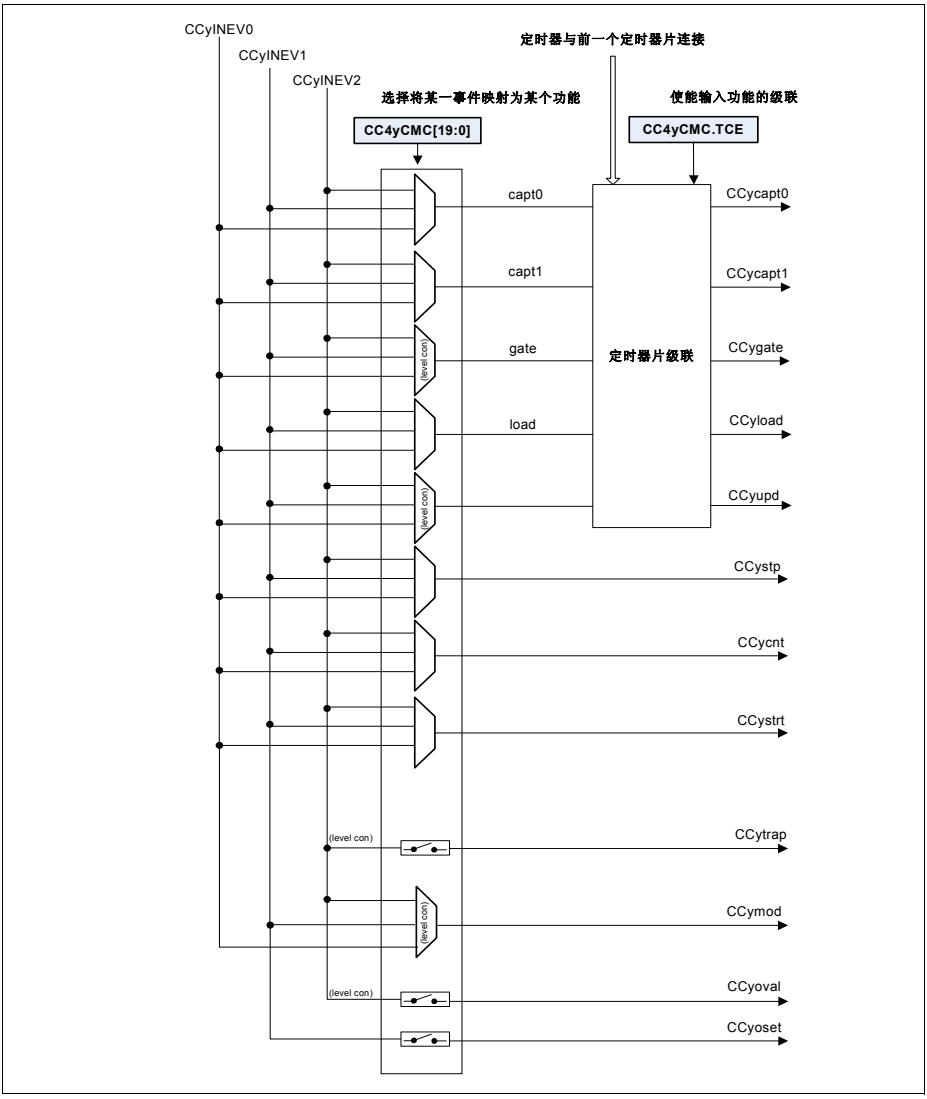


图 19-4　　定时器片连接矩阵示意图

19.2.4 启动 / 停止定时器

每个定时器片都有一个运行位寄存器 **CC4yTCST**.TRB，它指示定时器的当前状态。定时器的启动和停止可以通过软件访问来完成，也可以由外部事件直接控制，如**图 19-5**所示。

选择用一个外部信号作为启动触发信号并不要求用户必须同时使用一个外部信号作为停止触发信号，反之亦然。

选择单次模式意味着在计数器达到周期值后，运行位 **CC4yTCST**.TRB 将被清零，定时器因此而停止工作。

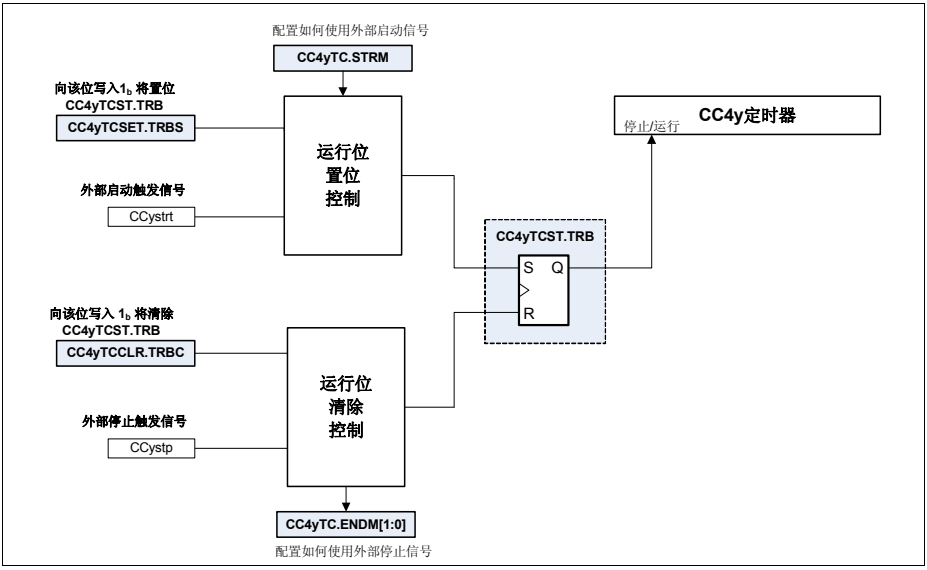


图 19-5 定时器启动 / 停止控制示意图

可以使用外部停止信号来执行下述功能 (通过 **CC4yTC**.ENDM 来配置)：

- 清除运行位 (停止定时器) - 默认
- 清除定时器 (到 0000_H) 但不清除运行位 (定时器仍然运行)
- 清除定时器和运行位

可以使用外部启动信号来执行下述功能 (通过 **CC4yTC**.STRM 来配置):

- 启动定时器 (恢复运行)
- 清除并启动定时器

定时器运行位的置位 (启动定时器) 操作总是优先于清除 (停止定时器) 操作。

为了同时 / 同步启动多个 CCU4 定时器，应当使用一个专用输入作为外部启动信号 (关于如何将一个输入信号配置为启动功能的描述，请参见 **19.2.7.1 节**)。该输入信号应被连接到需要同步启动的所有定时器 (关于模块连接的完整列表，请参见 **19.8 节**)，如**图 19-6** 所示。

对于通过软件来同步启动定时器，要有一个专用的输入信号与所有的 CCU4 定时器相连接，该输入信号由 SCU(系统控制单元) 来控制。该信号应被配置为外部启动信号 (请参见 19.2.7.1 节)，然后软件必须写 1_B 到 CCUCON 寄存器的特定位域 (该寄存器在 SCU 一章描述)。

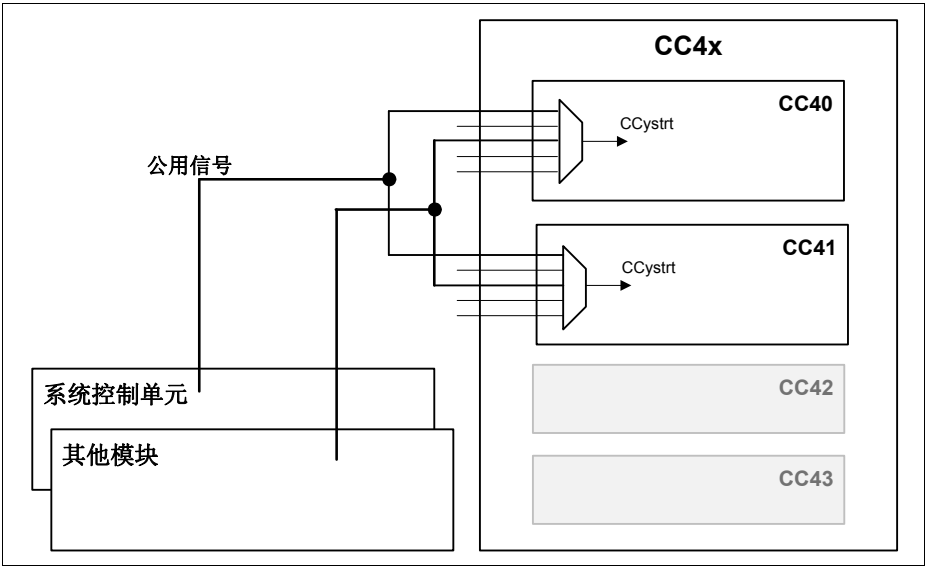


图 19-6　　同步启动多个定时器

19.2.5　计数模式

每个 CC4y 定时器片都可以被编程为三种不同的计数方式：

- 边沿对齐 (默认)
- 中心对齐
- 单次方式 (可以是边沿对齐或中心对齐)

这三种计数方式可以独立使用而不需选定任何输入作为外部事件源。不过，也可以通过外部事件来控制计数操作，如定时器门控信号、计数触发信号、外部停止信号、外部启动信号等。

对于所有计数模式，在运行时修改定时器周期值和比较通道值都是可能的。这就能够在每个时钟周期更新 PWM 信号的频率和占空比。

CC4y 定时器片的每个比较通道都有一个相关联的状态位 (**GCST**.CC4yST)，它指示通道是处于主动态还是被动态，如图 19-7 所示。状态位的置位和清除以及各自 PWM 信号的产生是由定时器的周期值、比较值和当前计数模式来控制的。关于如何置位和清除状态位，请参见 19.2.5.3 节 到 19.2.5.5 节对不同计数模式的描述 。

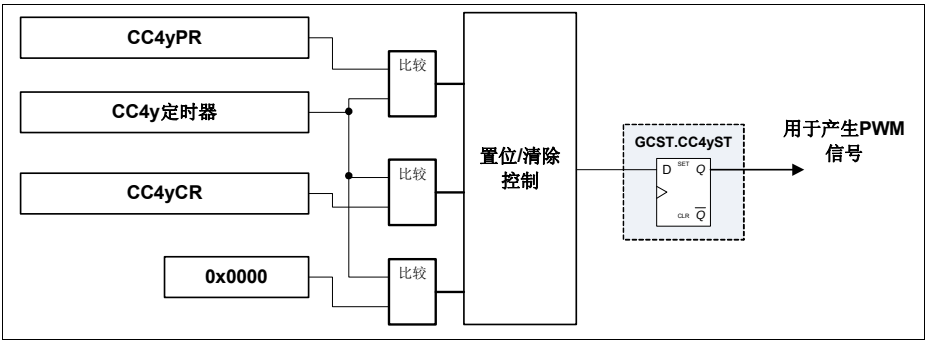


图 19-7　　CC4y 状态位

19.2.5.1　计算 PWM 信号的周期和占空比

定时器的周期由周期寄存器 **CC4yPR** 中的值和定时器的模式决定。

PWM 信号的周期和占空比的基准总是与定时器自身的时钟频率相关，而与模块时钟的频率无关 (因为定时器的时钟可以是模块时钟的一个分频版本)。

在边沿对齐模式，定时器的周期为：

T_{per}= < 周期值 > + 1; 以 f_{tclk} 计数　　(19.1)

在中心对齐模式，定时器的周期为：

T_{per}= (< 周期值 > + 1) x 2; 以 f_{tclk} 计数　　(19.2)

对于上述每种计数方式，所产生的 PWM 信号的占空比由 **CC4yCR** 寄存器中的编程值控制。

在边沿对齐和中心对齐模式， PWM 信号的占空比为：

DC= 1 - < 比较值 >/(< 周期值 > + 1)　　(19.3)

CC4yPR 和 **CC4yCR** 都可以在运行时通过软件修改，使所产生的 PWM 信号在不同周期值及占空比值之间过渡时可以避免出现毛刺，详见 **19.2.5.2 节**。

19.2.5.2　更新周期和占空比

每个 CCU4 定时器片为周期值和比较值分别提供一个相关联的映射寄存器。这就为通过软件来同时更新这两个参数提供了方便，其目的是可以在运行期间修改 PWM 信号的周期和占空比。

除了周期值和比较值的映射寄存器外，还有用于浮动预分频器和抖动功能的映射寄存器，分别是 **CC4yFPCS** 和 **CC4yDITS** (关于这些功能的详细描述，请参见 **19.2.11 节** 和 **19.2.10 节**)。

映射寄存器的结构如图 19-8 所示。

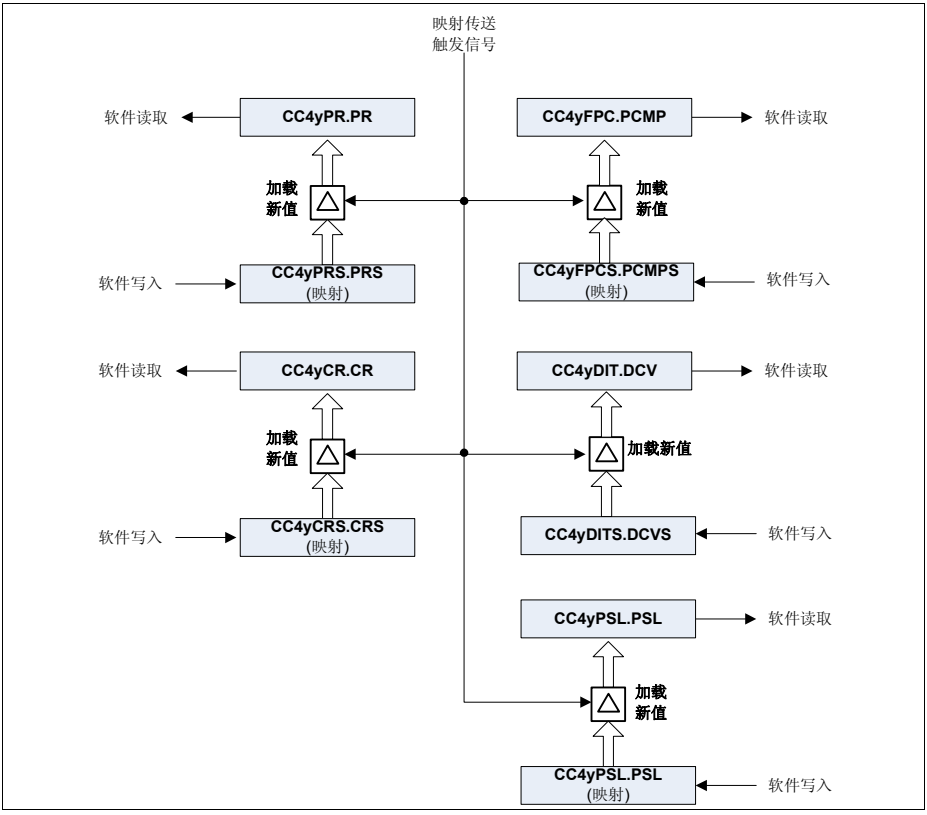


图 19-8　　映射寄存器概览

这些寄存器的更新只能通过向相关联的映射寄存器写一个新值并等待映射传送发生后完成。

每组映射寄存器有一个单独的映射传送使能位，如图 19-9 所示。当需要更新周期值和比较值时，软件必须将该使能位设置为 1_B。当完成值的更新时，这些使能位会被硬件自动清除。因此，每当需要更新寄存器时，软件必须再一次置位特定的使能位。

当然，也可以通过软件来清除使能位。这种方法可以用在值的更新操作需要被取消的情况下 (使能位已被置位之后)。

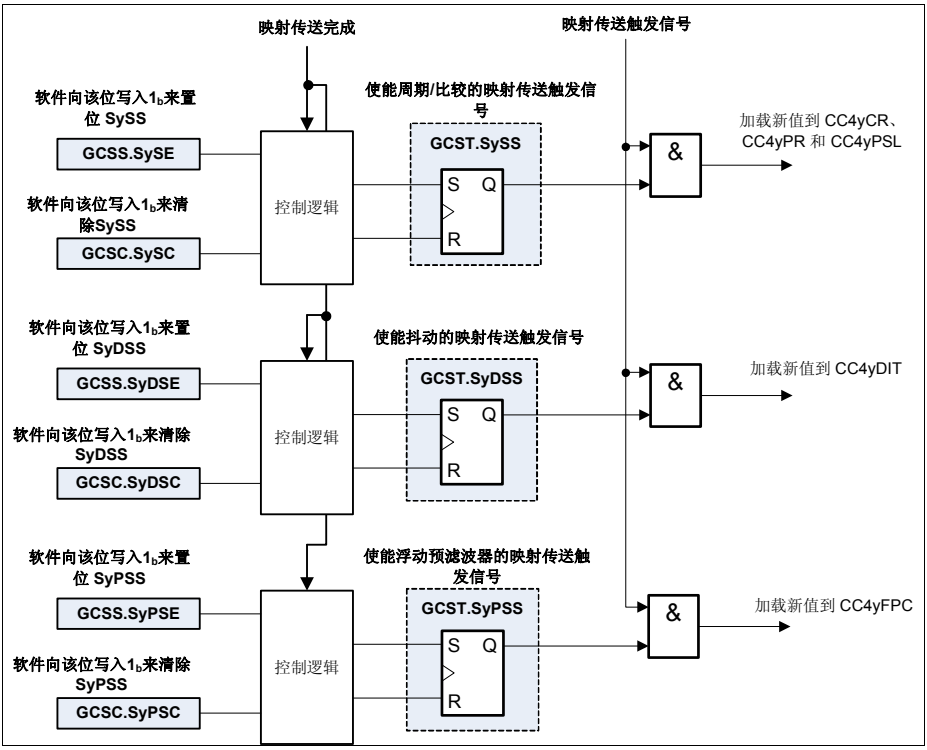


图 19-9　　映射传送使能逻辑

在映射传送使能位被置位 (**GCST**.SySS、**GCST**.SyDSS、**GCST**.SyPSS 设置为 1_B) 后，在紧接着出现映射传送触发信号时，会完成映射传送操作。

映射传送触发信号的发生取决于定时器的计数方式 (边沿对齐或中心对齐)。因此，寄存器值的更新时隙可以是：

- 向上计数期间发生周期匹配后的下一个时钟周期
- 向下计数期间发生 1 匹配后的下一个时钟周期
- 如果定时器停止且映射传送使能位被置位，则立即更新。

图 19-10 给出了映射传送控制的一个例子，此处定时器被配置为中心对齐模式。关于所有定时器片计数模式的详细描述，请参见 **19.2.5.3 节**、**19.2.5.4 节**和 **19.2.5.5 节**。

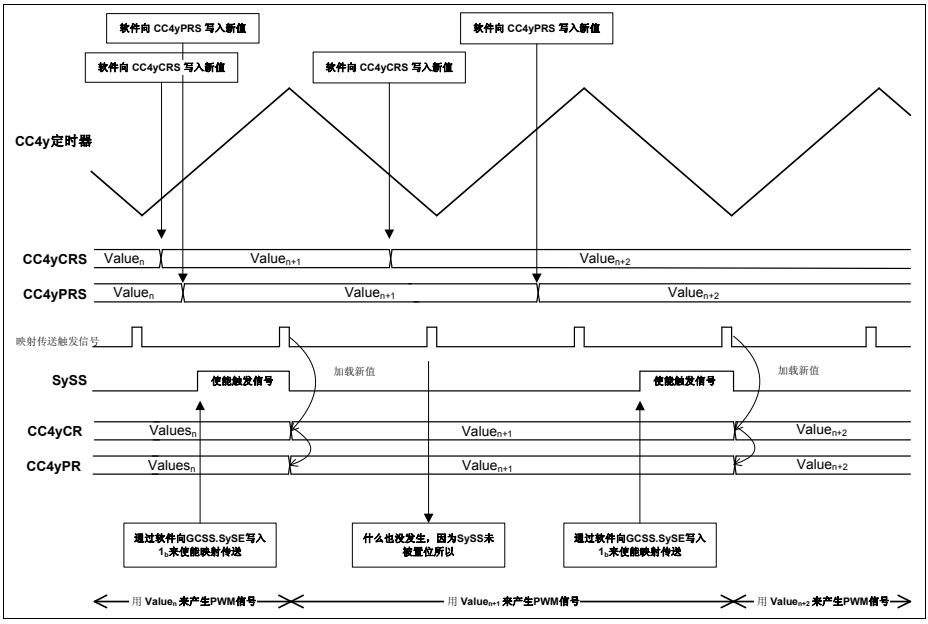


图 19-10　　映射传送时序示例 - 中心对齐模式

在某些应用情况下，可能需要通过硬件而不是软件来请求映射传送。为实现该操作，每个 CCU4 都包含一个专用输入 CCU4x.MCSS，用于通过硬件来请求一次映射传送。

该输入被使能后用于设置特定定时器片的映射传送使能位域 (**GCST**.SySS、**GCST**.SyDSS 和 **GCST**.SyPSS)。通过设置 **GCTRL**.MSEy 位域可以选择由哪一个定时器片使用该输入来执行同步操作。也可以将该输入信号用作三种不同的映射传送信号：比较和周期值、抖动比较值和预分频器比较值。这可以用 **GCTRL**.MSDE 位域来配置。

使用 CCU4x.MCSS 输入信号的结构如**图 19-9** 所示。该信号的使用只是增加了对映射传送的控制，因此所有前述功能仍然可用。

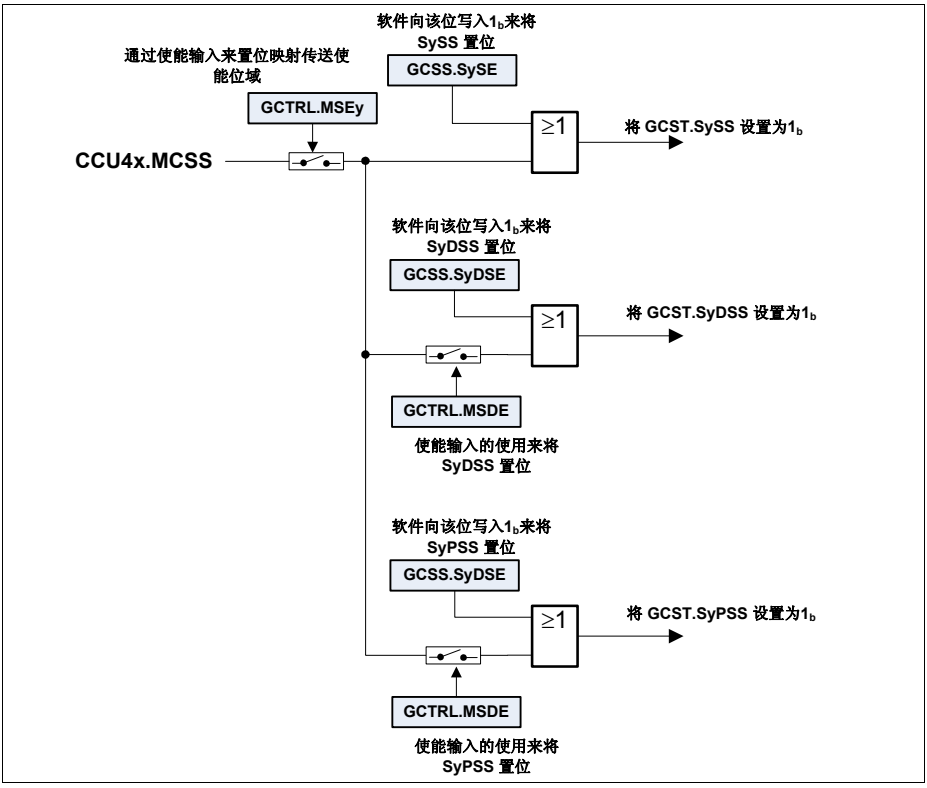


图 19-11　　CCU4x.MCSS 输入信号的使用

19.2.5.3　边沿对齐模式

边沿对齐模式是默认的计数方式。在该模式，定时器一直递增计数，直到其计数值与周期寄存器 **CC4yPR** 中的编程值相匹配。当检测到周期匹配时，定时器被清为 0000_H 并继续递增计数。

在该模式，每当发生溢出 (周期匹配) 时，周期寄存器和比较寄存器的值都会被已由软件写入到对应映射寄存器中的值更新，如**图 19-12** 所示。

在边沿对齐模式，定时器达到比较寄存器中的编程值后，再过一个时钟周期，比较状态位 (CC4yST) 会被置位。定时器达到 0000_H 后，再过一个时钟周期，状态位会被清除。

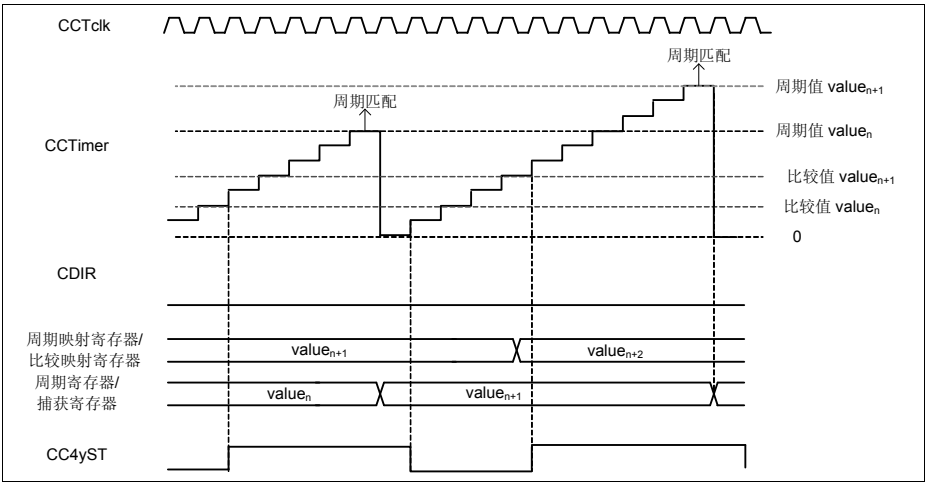


图 19-12　　边沿对齐模式 , CC4yTC.TCM = 0_B

19.2.5.4　中心对齐模式

在中心对齐模式，定时器遵循下述规则进行向上或向下计数：

- 当 **CC4yTCST**.CDIR = 0_B 时，计数器向上计数；当 **CC4yTCST**.CDIR = 1_B 时，计数器向下计数。
- 在向下计数期间，当计数值达到 0001_H 时，计数方向会在下一个时钟周期内被设置为向上计数 (**CC4yTCST**.CDIR = 0_B)。
- 在向上计数期间，当检测到周期匹配时，计数方向会在下一个时钟周期内被设置为向下计数 (**CC4yTCST**.CDIR = 1_B)。

当计数器的值等于或大于比较值时，状态位 (CC4yST) 一直为 1_B，否则，状态位为 0_B 。

在边沿对齐模式，每个周期执行一次比较寄存器和周期寄存器的映射传送。在中心对齐模式，每个周期执行如下两次映射传送。

- 在向上计数 (**CC4yTCST**.CDIR = 0_B) 期间，当计数器达到周期值后，在下一个时钟周期内执行一次。
- 在向下计数 (**CC4yTCST**.CDIR = 1_B) 期间，当计数器值达到 0001_H 后，在下一个时钟周期内执行一次。

*注：　位 **CC4yTCST**.CDIR 在 1 匹配或周期匹配后的下一个定时器时钟周期内改变，这意味着在改变计数方向之前，定时器会沿着以前的计数方向继续计数一个时钟周期。*

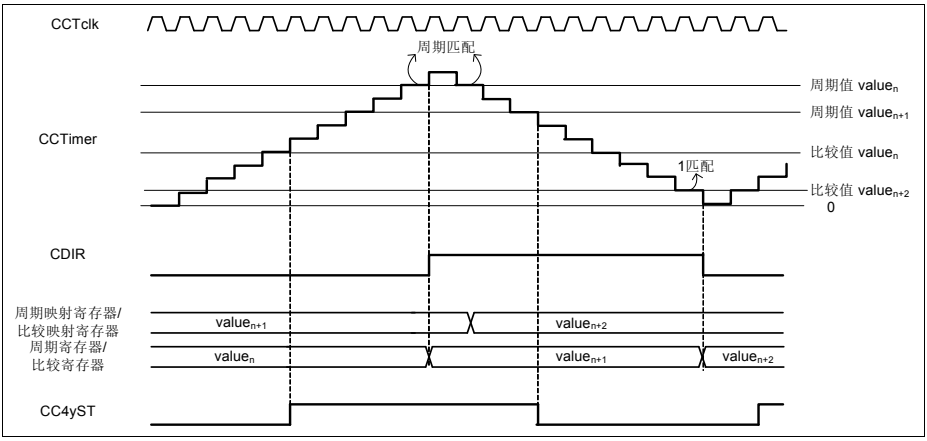


图 19-13　　中心对齐模式， CC4yTC.TCM = 1_B

19.2.5.5　单次模式

在单次模式，定时器会在当前的定时器周期结束后停止计数。该模式可以与中心对齐方式或边沿对齐方式一起使用。

在边沿对齐模式，如**图 19-14** 所示，定时器在达到周期值后被清为 0000_H，并停止计数。在中心对齐模式，如**图 19-15** 所示，当定时器向下计数到 0000_H 时，计数周期结束。

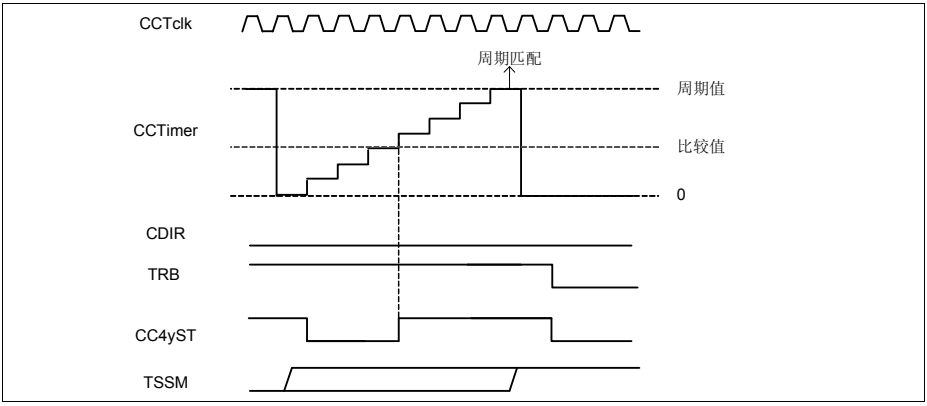


图 19-14　　单次边沿对齐 - CC4yTC.TSSM = 1_B, CC4yTC.TCM = 0_B

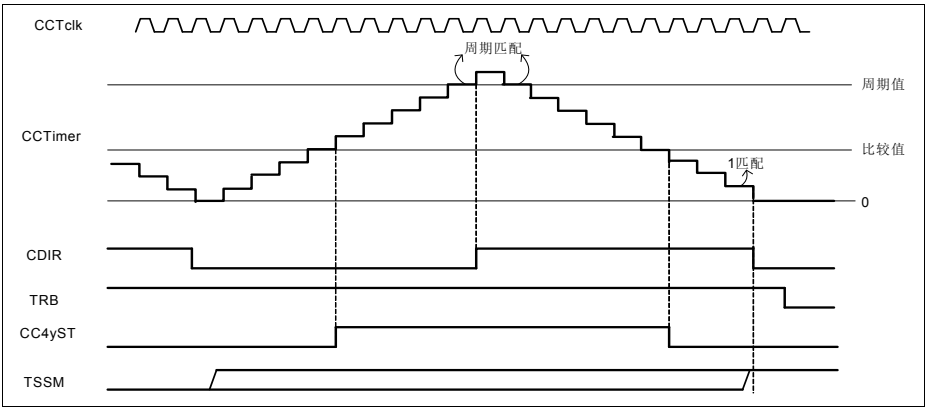


图 19-15　　单次中心对齐 - CC4yTC.TSSM = 1_B, CC4yTC.TCM = 1_B

19.2.6　主动 / 被动规则

可以总结出与定时器计数模式无关的置位或清除相关联的定时器片状态位 (CC4yST) 的一般规则。

下列事件会将状态位 (CC4yST) 设置为主动态：

- 向上计数时在比较匹配后的下一个 f_{tclk} 周期
- 向下计数时在 0 匹配后的下一个 f_{tclk} 周期

下列事件会将状态位 (CC4yST) 设置为被动态：

- 向上计数时在 0 匹配 (而不是比较匹配) 后的下一个 f_{tclk} 周期
- 向下计数时在比较匹配后的下一个 f_{tclk} 周期

如果使用外部事件来控制定时器操作，这些规则仍然适用。

状态位的状态只能通过软件或外部状态位覆盖功能来 “ 覆盖 ”，见 **19.2.7.10 节**。

软件可在任意时刻向 **GCSS**.SySTS 位域写入一个 1_B，这会将特定定时器片的状态位 **GCST**.CC4yST 置 1。向 **GCSC**.SySTC 位域写入一个 1_B 会将特定的状态位清 0。

19.2.7　外部事件控制

每个 CCU4 定时器片都可以最多使用三个不同的输入事件，见 **19.2.2 节**。这三个事件可以被映射为定时器片功能 (关于所有可用功能的描述，请参见 **19.2.3 节**)。

这些事件可以被映射到任何一个 CCU4x.INy[P...A] 输入，既没有限制一个事件不能用于执行多个功能，也没有限制一个输入不能被映射为多个事件 (例如，输入信号 X 用上升沿触发事件 0，而用下降沿触发事件 1)。

19.2.7.1 外部启动 / 停止

为了选择一个外部启动功能，应将一个事件 (输入选择器的输出) 映射到一个特定输入信号，这可以通过在 **CC4yINS**.EVxIS 域设置所需要的值并在 **CC4yINS**.EVxEM 域指定信号的有效边沿来完成。

然后，应通过给 **CC4yCMC**.STRTS (用于启动) 或 **CC4yTC**.ENDM (用于停止) 设定合适的值将该事件映射为启动或停止功能。

注意，启动和停止功能都是边沿有效而非电平有效。因此，主动态或被动态配置只能通过 **CC4yINS**.EVxEM 来设定。

在默认情况下，外部停止信号只会清除运行位 (**CC4yTCST**.TRB)，而启动功能则恰恰相反。不过，可以为外部启动和停止信号选择一个扩展的功能子集。该功能子集由寄存器 **CC4yTC**.ENDM (用于停止) 和 **CC4yTC**.STRM (用于启动) 控制。

对于启动子集 (**CC4yTC**.STRM)：

- 置位运行位 / 启动定时器 (恢复运行)
- 清除定时器，置位运行位 / 启动定时器 (清空并启动)

对于停止子集 (**CC4yTC**.ENDM)：

- 清除运行位 / 停止定时器 (停止)
- 清除定时器 (清空)
- 清除定时器，清除运行位 / 停止定时器 (清空并停止)

如果结合外部启动 / 停止 (也配置为启动 / 仅配置为清空) 功能，并使用一个外部向上 / 向下信号，则在清空操作过程中，若当前计数方向为向上计数，定时器将被设置为 0000_H，若计数方向为向下计数，则用周期寄存器中的值来设置定时器。

图 19-16 到**图 19-19** 展示了实现所有上述子集中的启动 / 停止功能的两个信号的使用。外部信号 (1) 用作高电平有效的启动信号，而外部信号 (2) 用作高电平有效的停止信号。

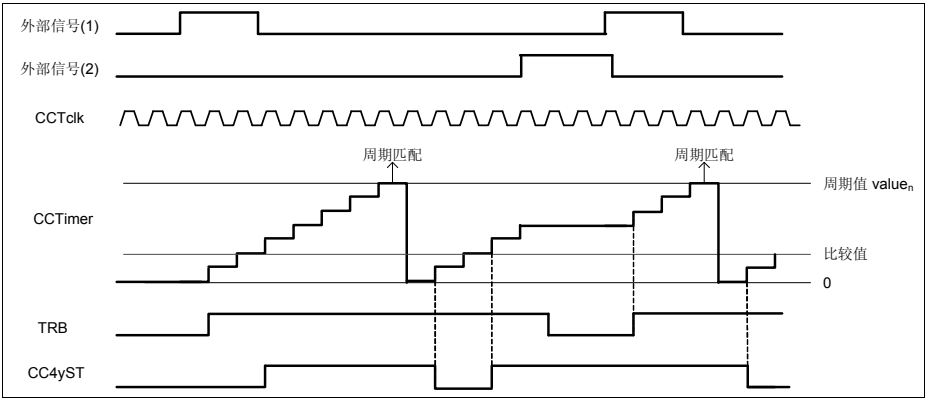


图 19-16　启动 (用作启动)/ 停止 (用作停止) - CC4yTC.STRM = 0_B, CC4yTC.ENDM = 00_B

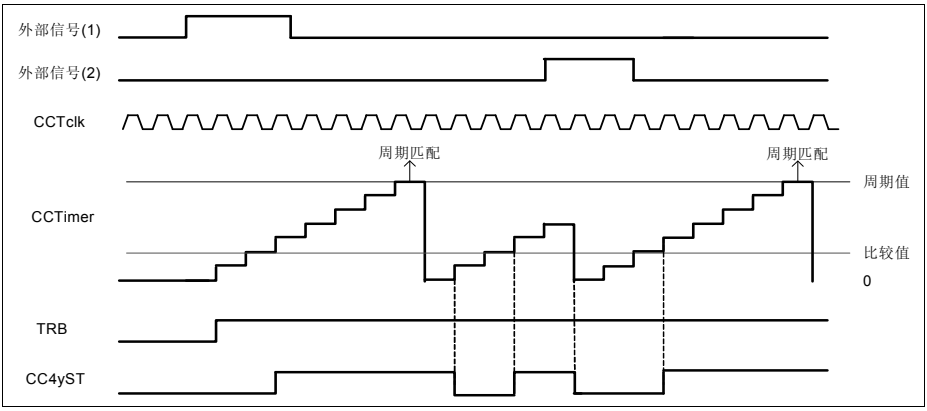


图 19-17　启动 (用作启动)/ 停止 (用作清空) - CC4yTC.STRM = 0_B, CC4yTC.ENDM = 01_B

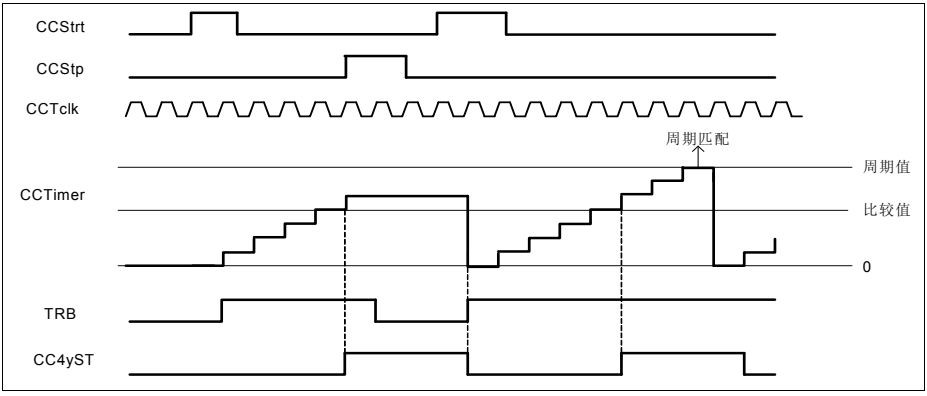


图 19-18　启动 (用作清空和启动)/ 停止 (用作停止) - CC4yTC.STRM = 1_B, CC4yTC.ENDM = 00_B

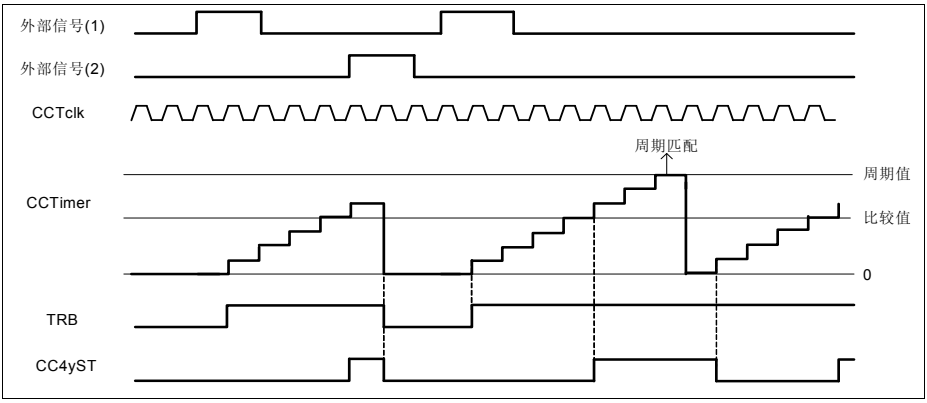


图 19-19　启动 (用作启动)/ 停止 (用作清空和停止) - CC4yTC.STRM = 0_B, CC4yTC.ENDM = 10_B

19.2.7.2　外部计数方向信号

可以选择一个外部输入作为向上 / 向下计数控制信号。 .

要选择一个外部向上 / 向下计数控制信号，应将一个事件 (输入选择器的输出) 映射为一个特定的输入信号，这可以通过在 **CC4yINS**.EVxIS 域设置所需要的值并在 **CC4yINS**.EVxLM 域指定信号的有效电平来完成。然后，应通过将 **CC4yCMC**.UDS 设置为合适的值将该事件映射为向上 / 向下功能。

注意，向上 / 向下功能为电平有效，因此，主动态 / 被动态配置只能通过 **CC4yINS**.EVxLM 来设定。

在定时器值等于或大于保存在比较寄存器中的值时，定时器片的状态位 (CC4yST) 会一直被置位，详见 **19.2.6 节**。

周期寄存器和比较寄存器的值在下列时刻更新：

- 在向上计数 (**CC4yTCST**.CDIR = 0_B) 期间，发生周期匹配后的下一个时钟周期
- 在向下计数 (**CC4yTCST**.CDIR = 1_B) 期间，发生 1 匹配后的下一个时钟周期

随着译码事件的变化，**CC4yTCST**.CDIR 寄存器的值会作相应的更新。向上 / 向下方向总是被理解为：当向下计数时 **CC4yTCST**.CDIR = 1_B，当向上计数时 **CC4yTCST**.CDIR = 0_B。使用一个外部信号来执行向上 / 向下计数功能并将事件配置为高电平有效，意味着该信号为高电平时定时器向上计数，该信号为低电平时定时器向下计数。

图 19-20 展示了用于控制定时器计数方向的一个外部信号。该信号被设定为高电平有效，这意味着该信号为高电平时定时器向下计数，而该信号为低电平时定时器向上计数。

*注：　对于一个低电平时向上计数而高电平时向下计数的方向控制信号，用户需要设置 **CC4yINS**.EVxLM = 0_B。当操作相反时，用户应设置 **CC4yINS**.EVxLM = 1_B。*

注： 使用一个外部计数方向控制信号，会将定时器片设置为边沿对齐模式。

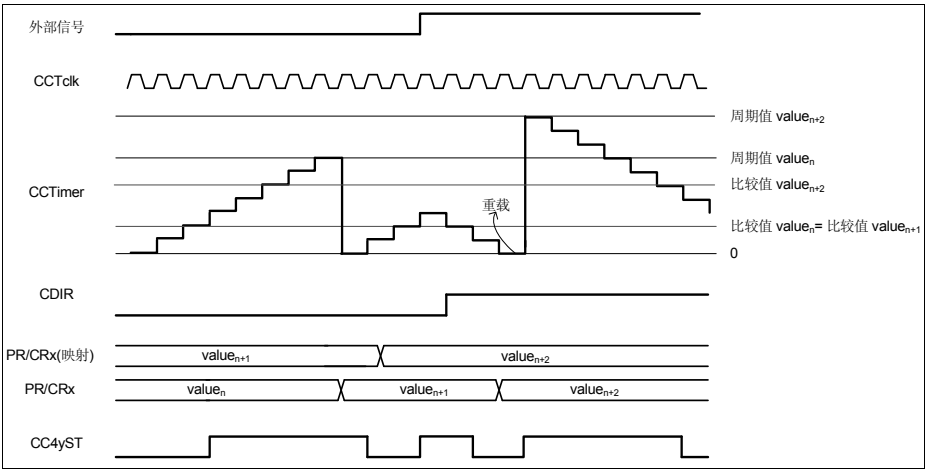


图 19-20 外部计数方向

19.2.7.3 外部门控信号

为了进行脉冲测量，用户可以选择一个输入信号用作计数门控。

要选择一个外部门控信号，应将一个事件 (输入选择器的输出) 映射为一个特定的输入信号，这可以通过在 **CC4yINS**.EVxIS 寄存器中设置所需要的值并在 **CC4yINS**.EVxLM 寄存器中指定信号的有效电平来实现。然后，应通过将 **CC4yCMC**.GATES 设置为合适的值，将该事件映射为门控功能。

注意，门控功能为电平有效，因此主动态/被动态配置只能通过**CC4yINS**.EVxLM来设定。

在一个外部门控信号有效期间，当计数器达到比较值后，状态位继续有效，而当计数器达到 0000_H 时，状态位变为无效。应当注意的是，计数器继续使用周期寄存器来识别回绕条件。**图 19-21** 展示了用于门控片计数器的外部信号的使用。该信号被设定为低电平有效，这意味着当外部信号值为零时，计数器门控功能有效。

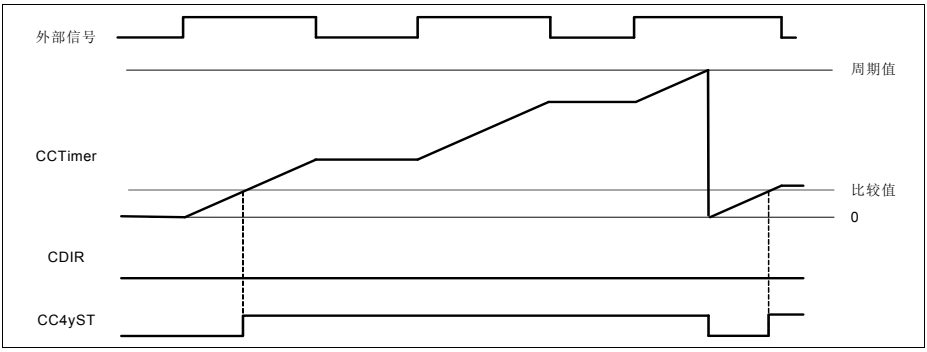


图 19-21　　外部门控信号

为了使用各种类型的外部门控功能，需要置位定时器片的特定运行位 **CC4yTCST**.TRB。这可以通过另外一个外部信号或直接通过软件来完成。

19.2.7.4　外部计数信号

也可以选择一个外部信号作为计数事件。

要选择一个外部计数信号，应将一个事件 (输入选择器的输出) 映射为一个特定的输入信号，这可以通过在 **CC4yINS**.EVxIS 寄存器中设置所需要的值并在 **CC4yINS**.EVxEM 寄存器中指定信号的有效边沿来实现。然后，应通过将 **CC4yCMC**.CNTS 设置为合适的值将该事件映射为计数功能。

注意，计数功能为边沿有效，因此主动态/被动态配置只能通过**CC4yINS**.EVxEM来设定。

可以选择只在上升沿、下降沿或两个边沿执行计数。在**图 19-22** 中，外部信号的上升沿和下降沿都被选择为计数事件。回绕条件仍然采用与周期寄存器进行比较。

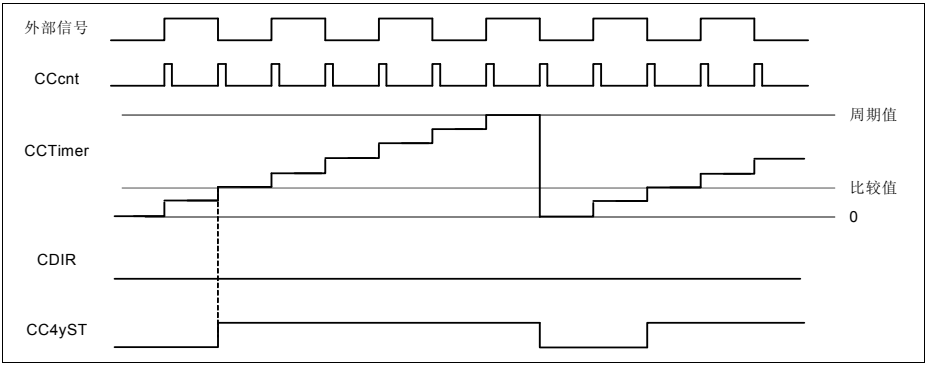


图 19-22　　外部计数信号

为了使用各种类型的外部门控功能，需要置位定时器片的特定运行位 **CC4yTCST**.TRB。这可以通过另外一个外部信号或直接通过软件来完成。

19.2.7.5 外部加载信号

CCU4 的每个定时器片还有一个功能，即允许用户选择一个外部信号作为定时器值重新加载 （重载）的触发信号，重载值可以是比较寄存器的当前值 (如果 **CC4yTCST**.CDIR = 0_B)，也可以是周期寄存器的当前值 (如果 **CC4yTCST**.CDIR = 1_B)。

要选择一个外部加载信号，应将一个事件 (输入选择器的输出) 映射为一个特定的输入信号，这可以通过在 **CC4yINS**.EVxIS 寄存器中设置所需要的值并在 **CC4yINS**.EVxEM 寄存器中指定信号的有效边沿来完成。然后，应通过将 **CC4yCMC**.LDS 设置为合适的值将该事件映射为加载功能。

注意，加载功能为边沿有效，因此主动态/被动态配置只能通过**CC4yINS**.EVxEM来设定。

在**图 19-23** 中，外部信号 (1) 被用作加载触发信号，上升沿有效。每当检测到外部信号 (1) 的一个上升沿时，比较寄存器的当前值就会被加载到定时器。如果使用一个外部信号来控制计数方向，向上或向下，也可以用周期寄存器中的设定值加载定时器。外部信号 (2) 代表计数方向控制 (高电平有效)。如果在检测到加载触发信号的那一刻控制计数方向的信号是向下计数，则定时器中的设置值为周期值。

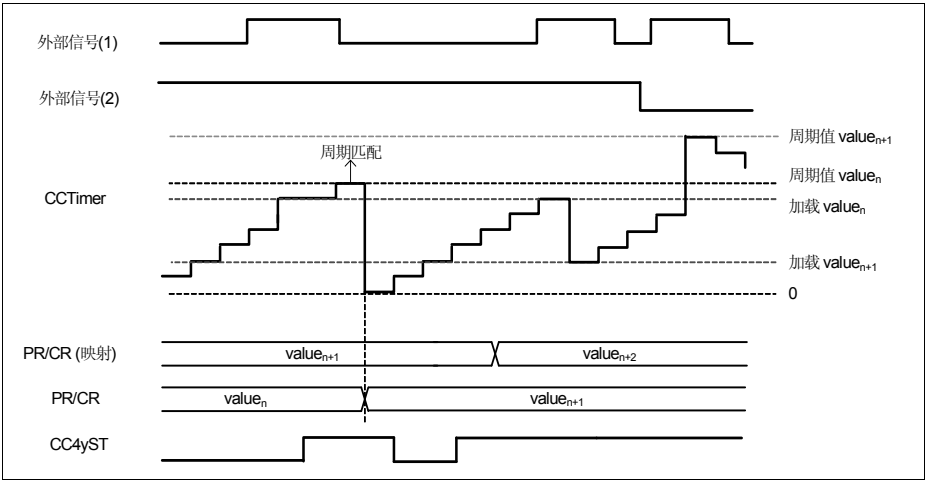


图 19-23 外部加载信号

19.2.7.6 外部捕获信号

当选择一个外部信号用作捕获触发信号时 (若 **CC4yCMC**.CAP0S 或 **CC4yCMC**.CAP1S 不等于 0_H)，特定的定时器片被自动设定为捕获模式。

在捕获模式，用户最多可以有 4 个捕获寄存器，如图 19-26 所示：捕获寄存器 0 (CC4yC0V)、捕获寄存器 1 (CC4yC1V)、捕获寄存器 2 (CC4yC2V) 和捕获寄存器 3 (CC4yC3V)。

这些寄存器在比较模式和捕获模式之间是共享的，这就要求：

- 如果 CC4yC0V 和 CC4yC1V 用于捕获，则比较寄存器 CC4yCR 和 CC4yCRS 不可用 (没有比较通道)
- 如果 CC4yC2V 和 CC4yC3V 用于捕获，则周期寄存器 CC4yPR 和 CC4yPRS 不可用 (没有周期控制)

要选择一个外部捕获信号，应将一个事件 (输入选择器的输出) 映射为一个特定的输入信号，这可以通过在 CC4yINS.EVxIS 寄存器中设置所需要的值并在 CC4yINS.EVxEM 寄存器中指定信号的有效边沿来完成。然后，应通过给 CC4yCMC.CAP0S/CC4yCMC.CAP1S 设置合适的值将该事件映射为捕获功能。

注意，捕获功能为边沿有效，因此主动态/被动态配置只能通过CC4yINS.EVxEM来设定。

用户可以选择下述捕获方案：

- CC4yC0V/CC4yC1V 和 CC4yC2V/CC4yC3V 使用不同的捕获事件
- CC4yC0V/CC4yC1V 和 CC4yC2V/CC4yC3V 使用同一个捕获事件和同一个捕获边沿。对于这种捕获方案，只有 CCcapt1 的功能需要编程设定。要启用该方案，需将 CC4yTC.SCE 域设置为 1_B 。

不同的捕获事件 (SCE = 0_B)

每当捕获触发信号 1(CCcapt1) 发生时，定时器的当前值被捕获到捕获寄存器 3，而先前保存在该寄存器的值被传送到捕获寄存器 2。

每当捕获触发信号 0(CCcapt0) 发生时，定时器的当前值被捕获到捕获寄存器 1，而先前保存在该寄存器的值被传送到捕获寄存器 0。

每次发生捕获到一个寄存器的操作时，该寄存器的满标志被置位。当软件读取该捕获寄存器的值 (通过读取特定的捕获寄存器或读取扩展的捕获读出值，详见 19.2.7.7 节) 时，该标志被硬件自动清除。

将一个新值捕获到特定的捕获寄存器，可以通过满标志描述如下：

$$CC4yC1V_{capt} = \text{NOT}(CC4yC1V_{full\_flag} \text{ AND } CC4yC0V_{full\_flag}) \quad (19.4)$$

$$CC4yC0V_{capt} = CC4yC1V_{full\_flag} \text{ AND NOT}(CC4yC0V_{full\_flag}) \quad (19.5)$$

也可以通过设定 CC4yTC.CCS = 1_B 来禁止满标志的影响。这样一来，不论捕获的值是否被读取，都可以进行连续捕获。

注：　当使用周期寄存器进行捕获时，CC4yCMC.CAP1S 不为 00_B，计数器总是使用其全部 16 位宽作为周期值。

在图 19-24 中，一个外部信号被选择作为一个捕获事件，用于将定时器的值捕获到 CC4yC0V/CC4yC1V 寄存器。在捕获模式期间，当检测到捕获触发信号时，状态位 CC4yST 变为有效，当计数器值为 0000_H 时，状态位变为无效。

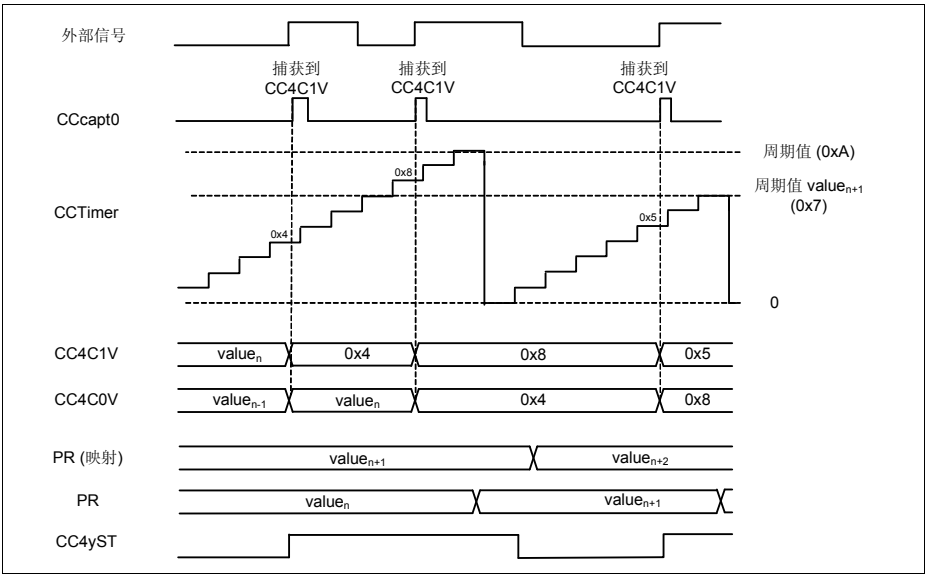


图 19-24　　外部捕获 - CC4yCMC.CAP0S != 00_B, CC4yCMC.CAP1S = 00_B

在图 19-25 中，两个不同的信号被用作将定时器值捕获到 CC4yC0V/CC4yC1V 和 CC4yC2V/CC4yC3V 寄存器的触发源。

外部信号 (1) 被选为 CC4yC0V/CC4yC1V 寄存器的捕获源，上升沿有效。外部信号 (2) 被选为 CC4yC2V/CC4yC3V 寄存器的捕获源，但是与外部信号 (1) 相反，有效沿被选择为下降沿。

关于捕获模式使用的详细描述，请参见 19.2.12.4 节。

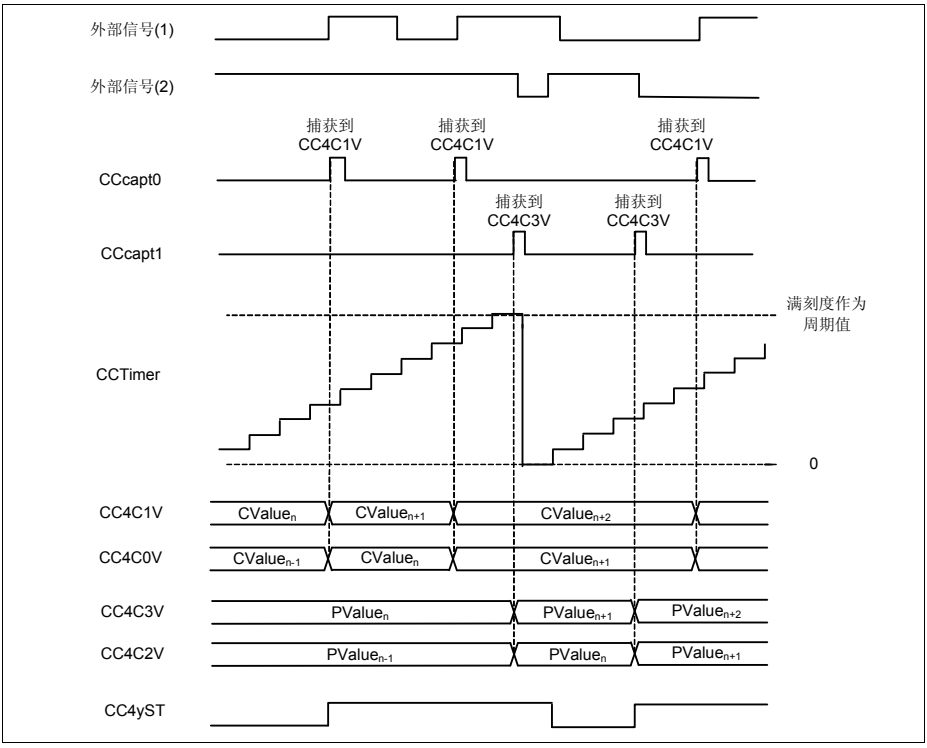


图 19-25　　外部捕获 - CC4yCMC.CAP0S != 00_B, CC4yCMC.CAP1S != 00_B

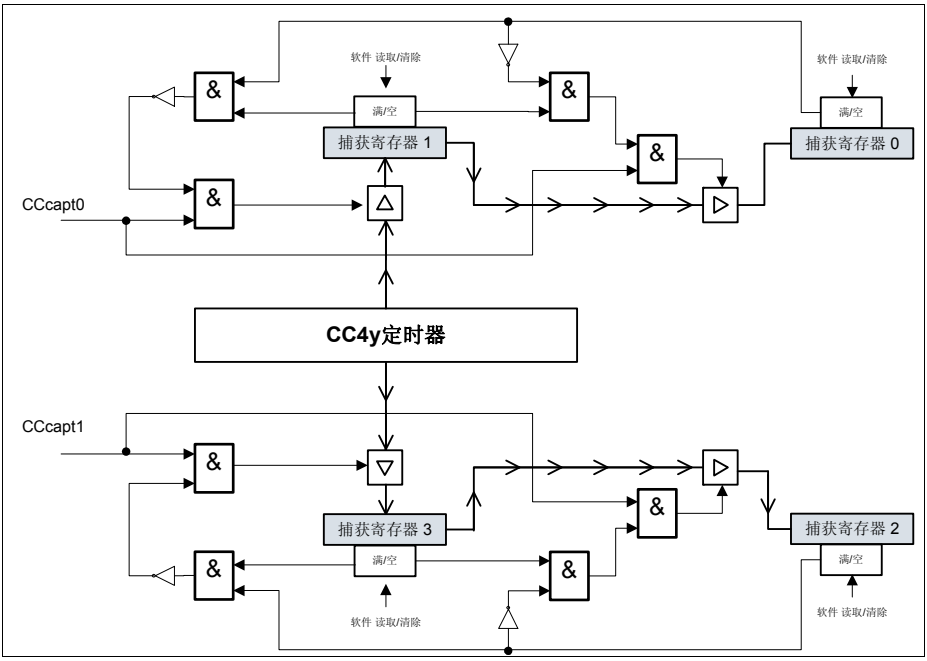


图 19-26　　定时器片捕获逻辑

相同的捕获事件 (SCE = 1_B)

设置位域 **CC4yTC**.SCE = 1_B，可以使 4 个捕获寄存器连接到同一捕获事件，如**图 19-28** 所示。 CCcapt1 控制捕获功能。

捕获逻辑采用如**图 19-26** 所示的相同结构，但是被扩展成一个四寄存器链，如**图 19-27** 所示。四寄存器链采用相同的满标志锁定规则 (也可以通过设置 **CC4yTC**.CCS = 1_B 来禁止)：

$$CC4yC3V_{capt} = \text{NOT}(CC4yC3V_{full\_flag} \text{ AND } CC4yC2V_{full\_flag} \text{ AND } CC4yC2V_{full\_flag} \text{ AND } CC4yC1V_{full\_flag}) \quad (19.6)$$

$$CC4yC2V_{capt} = CC4yC3V_{full\_flag} \text{ AND NOT}(CC4yC2V_{full\_flag} \text{ AND } CC4yC1V_{full\_flag} \text{ AND } CC4yC0V_{full\_flag}) \quad (19.7)$$

$$CC4yC1V_{capt} = CC4yC2V_{full\_flag} \text{ AND NOT}(CC4yC1V_{full\_flag} \text{ AND } CC4yC0V_{full\_flag}) \quad (19.8)$$

$$CC4yC0V_{capt} = CC4yC1V_{full\_flag} \text{ AND NOT}(CC4yC0V_{full\_flag}) \quad (19.9)$$

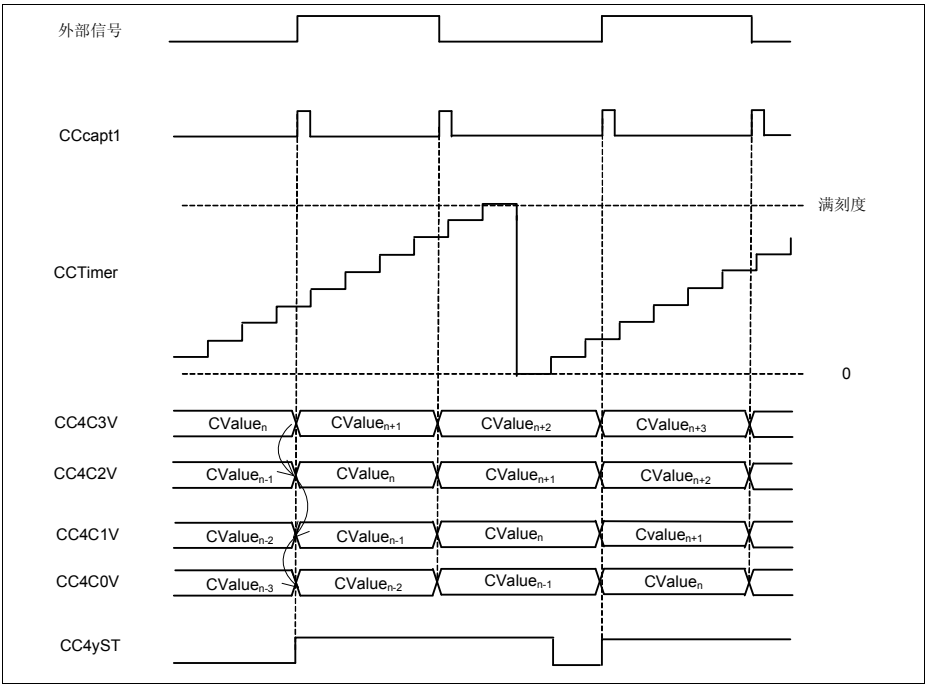


图 19-27　外部捕获 - CC4yTC.SCE = 1_B

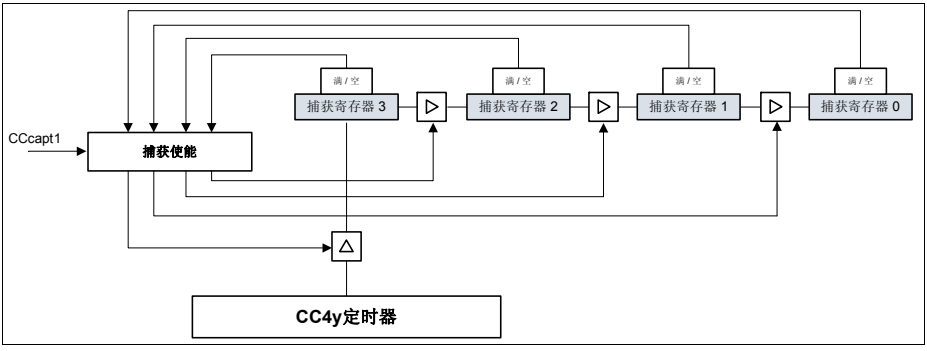


图 19-28　定时器片捕获逻辑 - CC4yTC.SCE = 1_B

19.2.7.7 捕获扩展回读模式

每个定时器片的捕获逻辑都可以工作在 FIFO 回读模式。该模式可通过设置 **CC4yTC**.ECM = 1_B 来使能。该扩展回读模式允许软件总是从同一个地址读回捕获数据 (**CC4yECRD0** 用于与捕获触发信号 0 连接的结构，**CC4yECRD1** 用于与捕获触发信号 1 连接的结构)。该回读操作将总是返回最老的捕获值，从而简化了重建捕获数据的软件程序实现。

该功能允许每个捕获触发信号使用一个单独的 FIFO 结构。当发生多个捕获触发而软件来不及对每个捕获事件执行读操作时，该功能将大大减轻软件回读程序的负担。

该 FIFO 回读功能有两种结构：深度为 4 的 FIFO 结构和深度为 2 的 FIFO 结构。

读回的数据中还包含一个丢失值位域，它指示是否因为 FIFO 结构已满而丢掉了一个捕获触发信号。当检测到一个捕获事件而 FIFO 已满 (无论是否使能了连续捕获模式) 时，该位域被置位。当发生对 **CC4yECRD0**/**CC4yECRD1** 寄存器的下一次读操作时，该位域由硬件自动清除。该位域不能指示出丢失了多少捕获事件，它只能指示在两个 ECRD 读操作之间至少丢失了一个捕获事件(这可帮助软件来分析所读数据的哪一部分可用于计算)。

*注： 当 ECM 位域被置位时，仍可以读取各个捕获寄存器。不过，满标记只能在通过 **CC4yECRD0**/**CC4yECRD1** 地址完成一个回读操作时由硬件来清除。*

深度为 4 的 FIFO 结构

当捕获触发信号 1 被使能且 **CC4yTC**.SCE = 1_B (相同捕获事件) 时，硬件中会存在深度为 4 的 FIFO 结构，如**图 19-29** 所示。

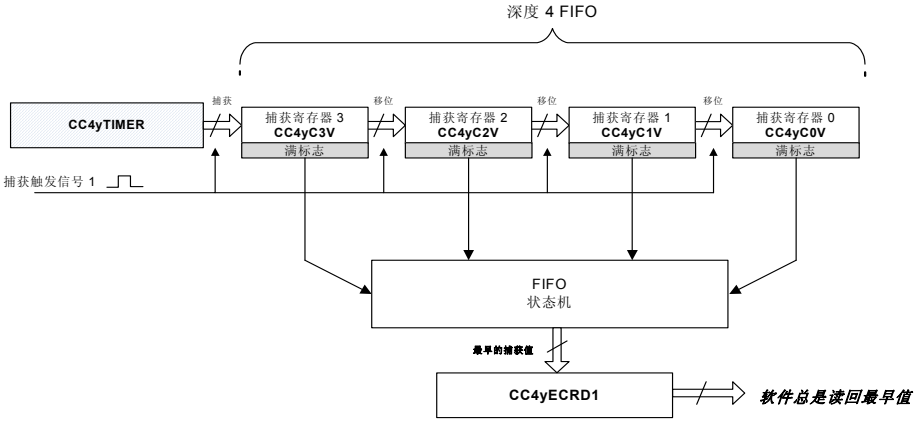


图 19-29 捕获扩展回读 - 深度 4

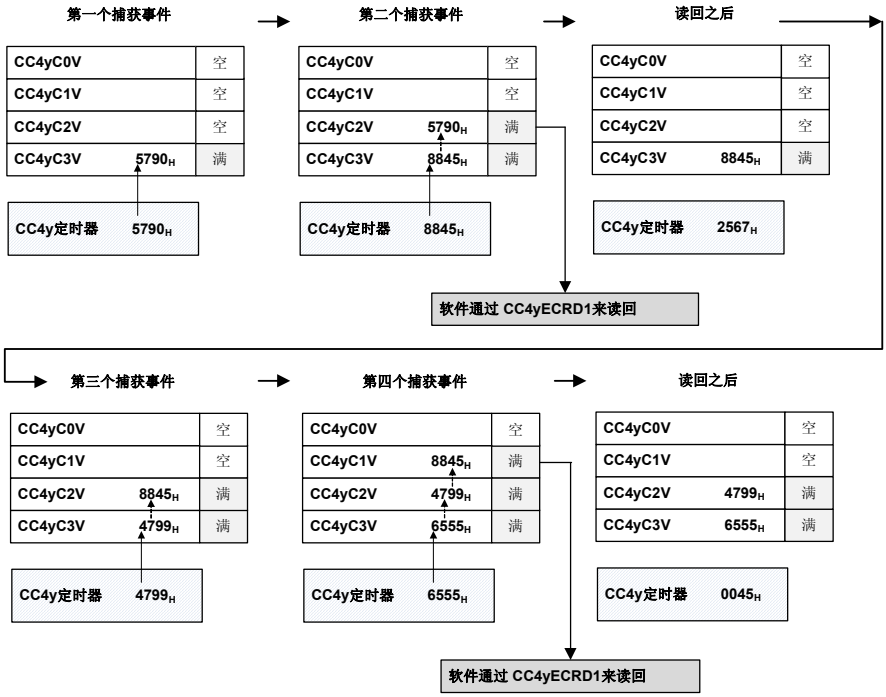


图 19-30　　深度 4 软件访问示例

深度为 2 的 FIFO 结构

每个定时器片有两个深度为 2 的捕获结构：一个与捕获触发信号 0 一起使用，而另一个与捕获触发信号 1 一起使用。

与捕获触发信号 0 连接的捕获结构通过 **CC4yECRD0** 来访问，而与捕获触发信号 1 连接的捕获结构通过 **CC4yECRD1** 来访问，如**图 19-31** 所示。

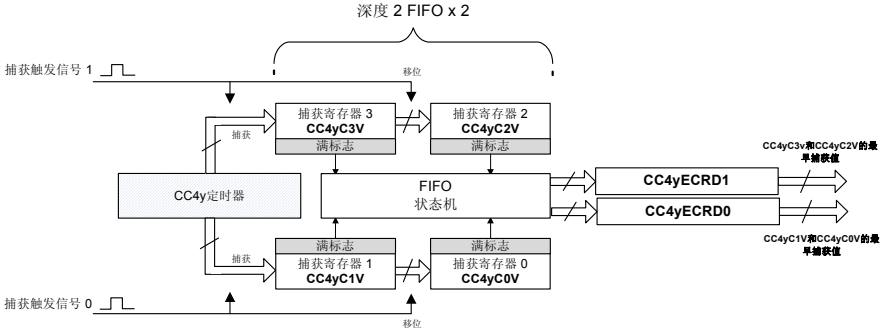


图 19-31　　捕获扩展回读 - 深度 2

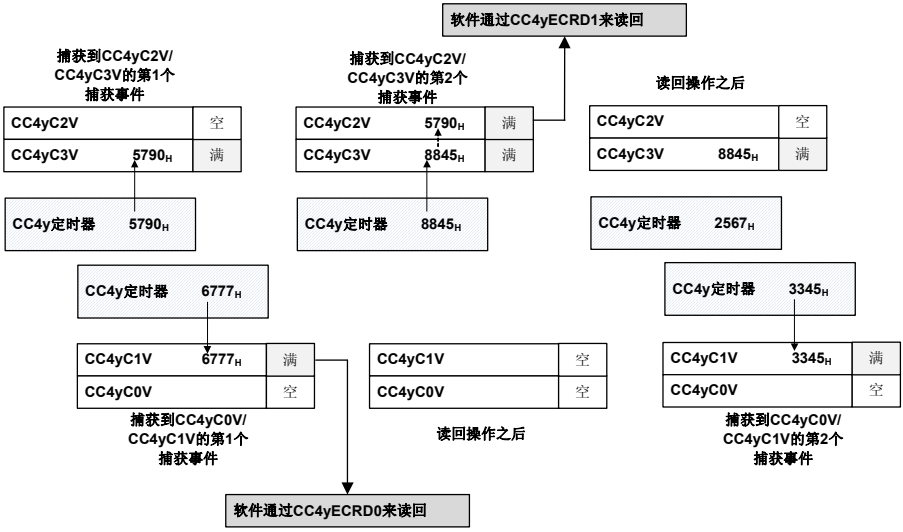


图 19-32　　深度 2 软件访问示例

19.2.7.8　外部调制

可以使用一个外部信号在每个定时器片的输出进行信号调制。

要选择一个外部调制信号，应将一个输入信号映射为一个事件，这可以通过在 **CC4yINS**.EVxIS 寄存器中设置所需要的值并在 **CC4yINS**.EVxLM 寄存器中指定信号的有效电平来完成。然后，应通过设置 **CC4yCMC**.MOS = 01_B (如果使用事件 0) 或

CC4yCMC.MOS = 10_B (如果使用事件 1) 或 **CC4yCMC**.MOS = 11_B (如果使用事件 2) 将该事件映射为调制功能。

注意，调制功能为电平有效，因此主动态/被动态配置只能通过**CC4yINS**.EVxLM来设定。

调制有两种工作模式：

- 调制事件用于清除 CC4yST 位 - **CC4yTC**.EMT = 0_B
- 调制事件用于门控输出 - **CC4yTC**.EMT = 1_B

在 **图 19-33** 中，有一个外部信号被配置为调制源用于清除 CC4yST 位，**CC4yTC**.EMT = 0_B。该信号被编程为低电平有效事件，因此当该信号为低电平时，输出值遵循标准的主动 / 被动规则。

当信号为高电平 (无效状态) 时，CC4yST 位被清零，输出会被强制为被动态。注意，状态位 CC4yST 的值和特定输出 CCU4x.OUTy 未连接在一起。可以通过 PSL 位选择输出为低电平有效或高电平有效。

外部调制无效状态的退出与 PWM 信号同步，这是因为在调制信号处于无效状态期间，CC4yST 位被清除且不能被置位。

通过设定 **CC4yTC**.EMS = 1_B，还可以实现进入无效态与 PWM 信号的同步。这样一来，可以避免输出信号所有可能的假信号，如**图 19-34** 所示。

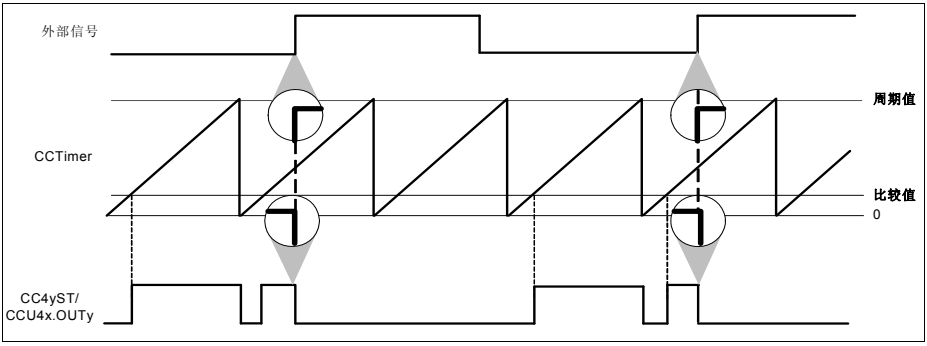


图 19-33　　外部调制信号复位 ST 位 - CC4yTC.EMT = 0_B

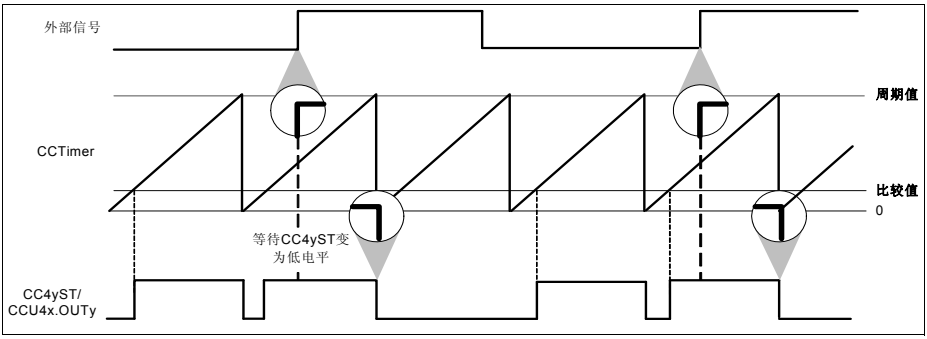


图 19-34　外部调制信号复位 ST 位 - CC4yTC.EMT = 0_B, CC4yTC.EMS = 1_B

在图 19-35 中，外部调制事件被用作输出的门控信号，CC4yTC.EMT = 1_B。外部信号被配置为高电平有效，CC4yINS.EVxLM = 0_B，这意味着当外部信号为高电平时，输出被设定为被动状态。在该模式下，通过设定 CC4yTC.EMS = 1_B，可以使门控事件与 PWM 信号同步。

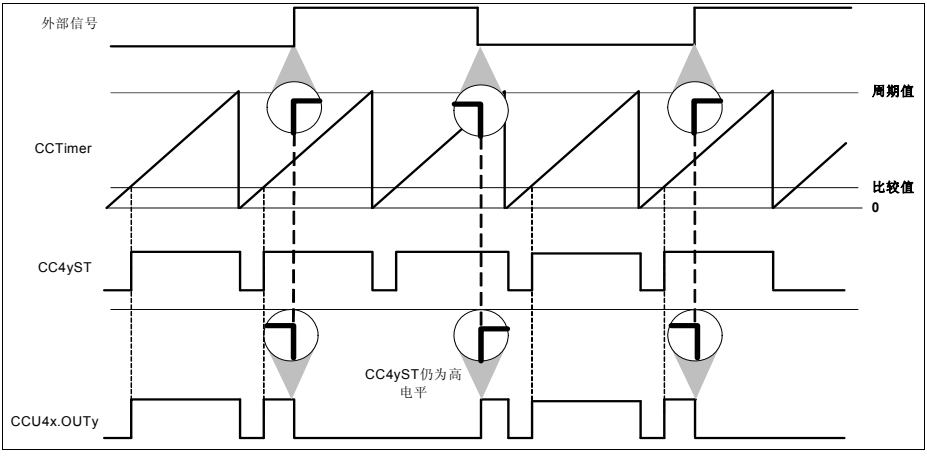


图 19-35　外部调制信号门控输出 - CC4yTC.EMT = 1_B

19.2.7.9　陷阱功能

陷阱功能允许 PWM 输出对输入引脚的状态做出反应。该功能可在陷阱输入变为有效时用于关闭功率器件。

要选择陷阱功能，应将一个输入信号映射为事件 2，这可以通过在 **CC4yINS**.EV2IS 寄存器中设置所需要的值并在**CC4yINS**.EV2LM 寄存器中指定信号的有效电平来完成。然后，应通过设定 **CC4yCMC**.TS = 1_B 将该事件映射为陷阱功能。

注意，陷阱功能为电平有效，因此主动态 / 被动态配置只能通过 **CC4yINTS**.EV2LM 来设定。

可以通过软件来监视两个位域，以交叉检验陷阱功能，如**图 19-36** 所示：

- 陷阱状态位 **CC4yINTS**.E2AS。该位域指示陷阱功能当前是否有效。因此，该位域可将特定定时器片的输出设置为主动态或被动态。
- 陷阱标志 **CC4yINTS**.TRPF。在陷阱条件被硬件自动清除的情况下，该位域被用作遗留标志。该位域需要由软件来清除。

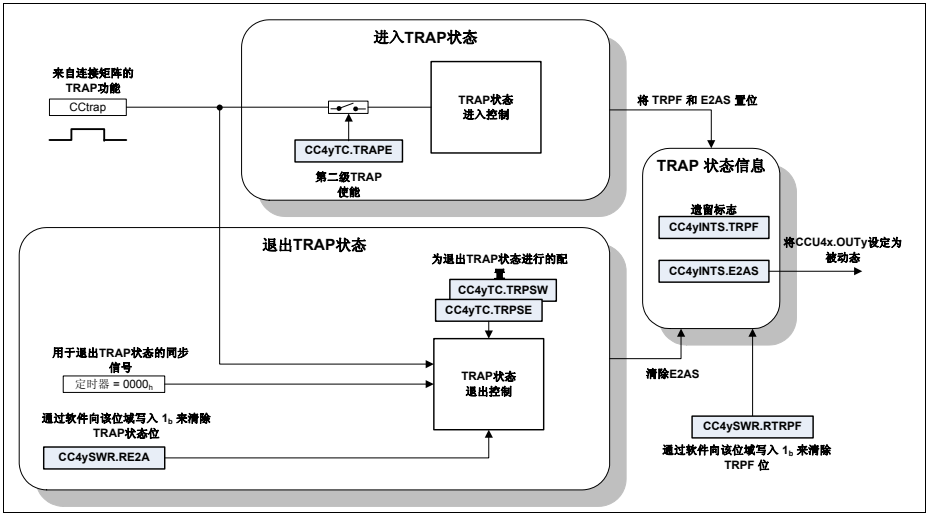


图 19-36　　陷阱控制示意图

当在所选择的输入引脚上检测到陷阱条件时，陷阱标志和陷阱状态位都会被置为 1_B。通过将 CCU4x.OUTy 设置为编程的被动态，会立即进入陷阱状态，如**图 19-37** 所示。

退出陷阱状态有两种方式 (**CC4yTC**.TRPSW 寄存器)：

- 当陷阱信号变为无效时 - **CC4yTC**.TRPSW = 0_B，由硬件自动退出。
- 仅由软件通过清除 **CC4yINTS**.E2AS 来执行退出。仅在输入的陷阱信号处于无效状态时，清除操作才有效 - **CC4yTC**.TRPSW = 1_B。

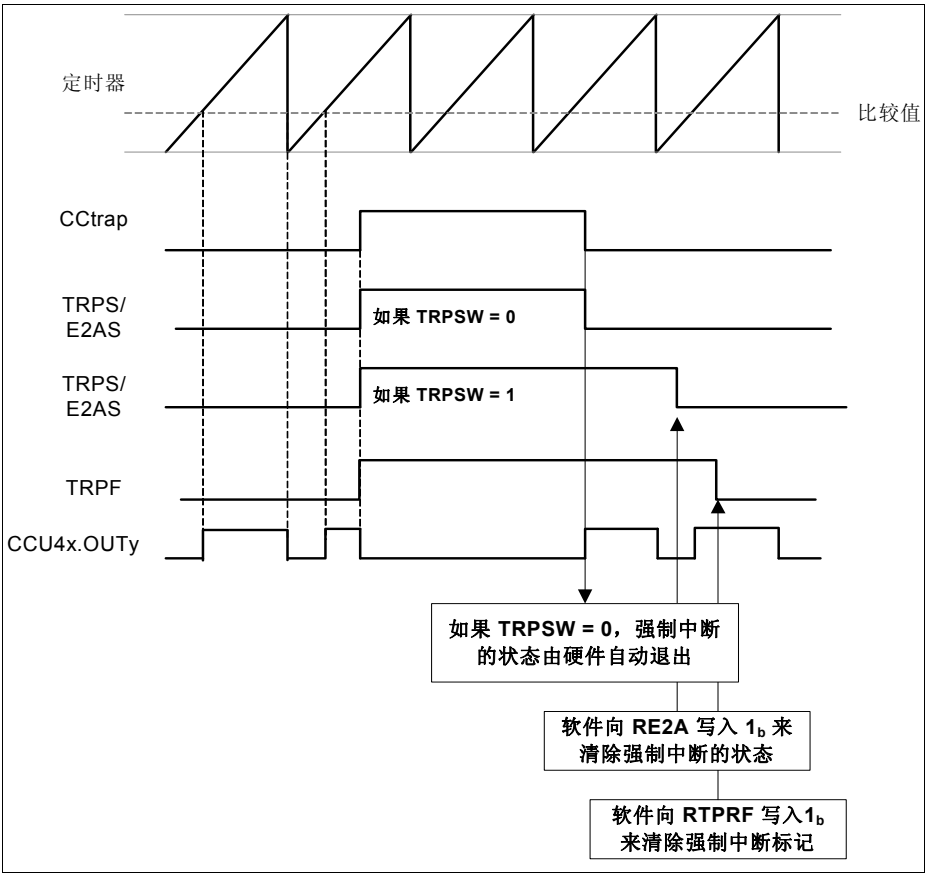


图 19-37　　陷阱时序图 , CC4yPSL.PSL = 0_B (输出被动态电平为 0_B)

还可以将退出陷阱状态与PWM信号进行同步，如图19-38所示。当位域CC4yTC.TRPSE = 1_B 时，该功能被使能。

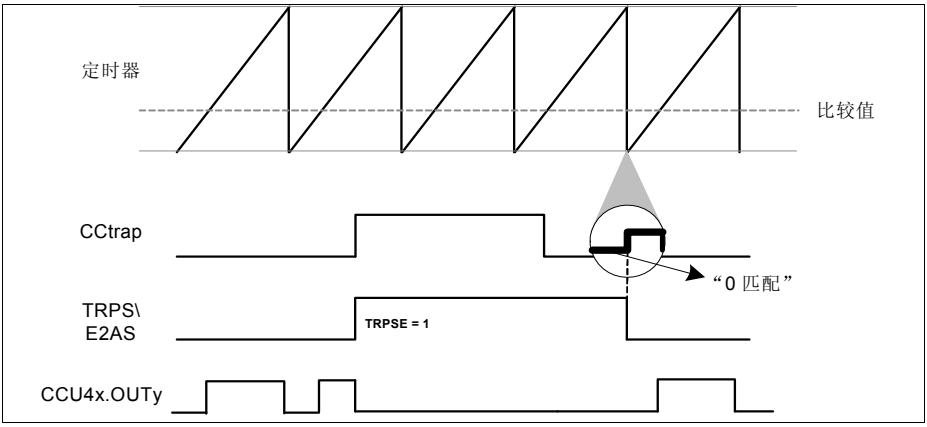


图 19-38　　陷阱与 PWM 信号同步， CC4yTC.TRPSE = 1_B

19.2.7.10 状态位覆盖

对于复杂的定时输出控制，每个定时器片都有用一个来自外部信号的值来覆盖状态位 (CC4yST) 的功能。

状态位覆盖能引起输出引脚 CCU4x.OUTy 的状态变化 (从无效变为有效，或反之)。

要使能该功能，需要两个信号：

- 一个信号用作覆盖状态位的触发信号 (边沿有效)
- 一个信号包含状态位要被设定的值 (电平有效)

要使用状态位覆盖功能，应当将用作触发的信号映射为事件　　1，这可以通过在 **CC4yINS**.EV1IS 寄存器中设置所需要的值并在 **CC4yINS**.EV1EM 寄存器中指定信号的有效边沿来完成。

承载着状态位设定值的信号需要被映射到事件 2，这可通过在 **CC4yINS**.EV2IS 寄存器中设置所需要的值来实现。如果不需要对信号进行反相，则应将 **CC4yINS**.EV2LM 寄存器设置为 0_B ，否则，设置为 1_B。

然后，应通过设置 **CC4yCMC**.OFS = 1_B 将事件映射为状态位覆盖功能。

图 19-39 展示了状态位覆盖功能，在此，外部信号 (1) 被选为触发源 (上升沿有效)，外部信号 (2) 被选为覆盖值。

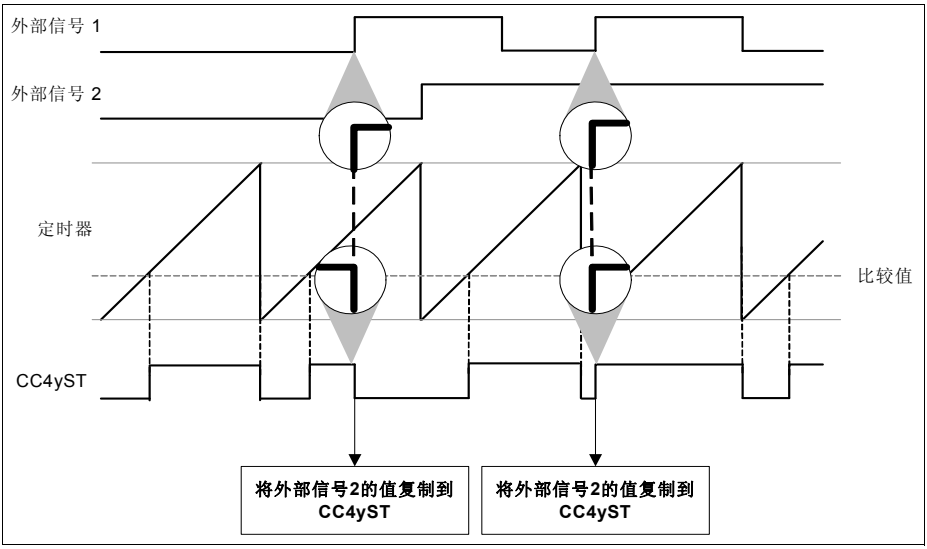


图 19-39　　状态位覆盖

19.2.8　多通道控制

可通过设置 **CC4yTC**.MCME = 1_B 来单独选择每个定时器片的多通道控制模式。

在该模式，定时器片 (设置为多通道模式) 的输出状态可通过一个序列并行控制。

该序列由 CCU4 的输入 CCU4x.MCI0、CCU4x.MCI1、CCU4x.MCI2 和 CCU4x.MCI3 来控制。每个输入直接连接到所关联的定时器片输入，例如， CCU4x.MCI0　连接到 CC40MCI， CCU4x.MCI1 连接到 CC41MCI。

该序列可以由其他模块直接控制。每个器件的连接可以有不同的控制方案，因此，应参阅 **19.8 节** 来确认哪些模块连接到这些输入。

当使用 CCU4 的多通道模式时，可以实现输出状态更新 (CCU4x.OUTy) 和新序列更新的完全同步，如**图 19-40** 所示。这一同步特性可通过使用某些特定模块来使能，因此，应参阅 **19.8 节**来确认哪个模块控制 CCU4x.MCIy 输入。

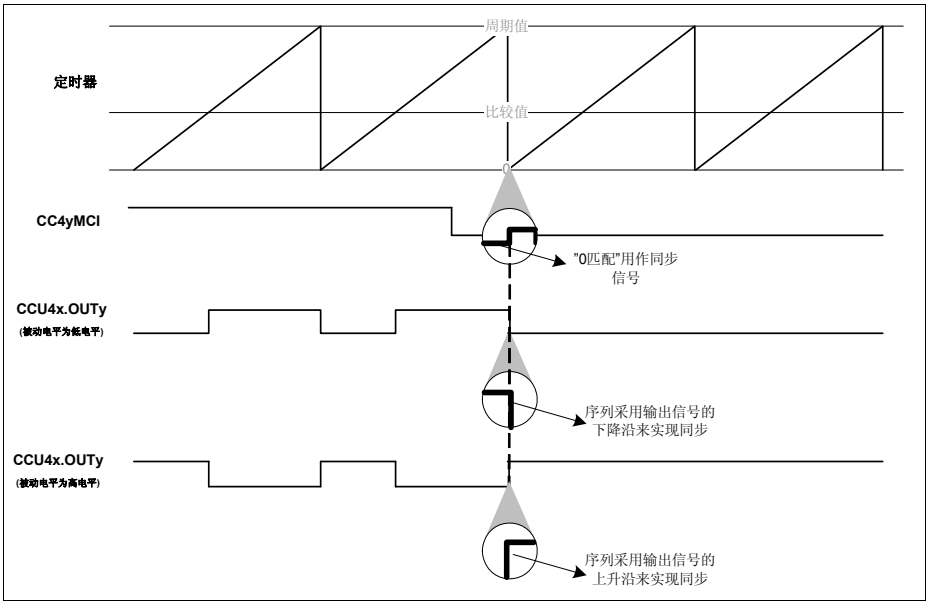


图 19-40　　多通道序列同步

图 19-41 展示了多通道模式与 CCU4 内部四个定时器片的结合使用。

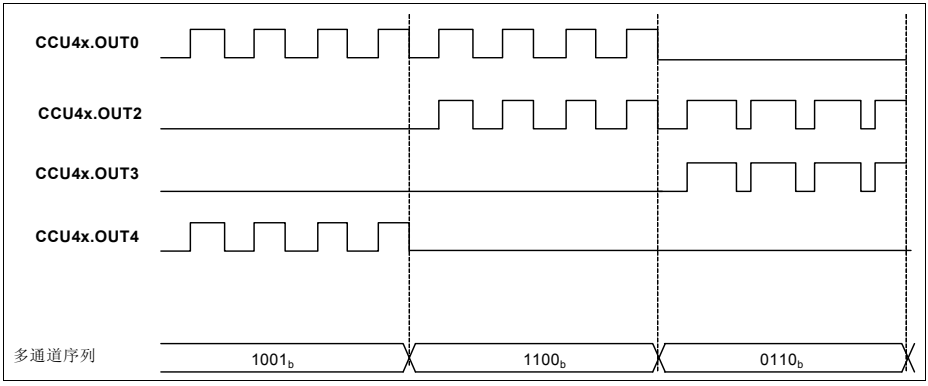


图 19-41　　多个定时器片的多通道模式

CCU4 与控制多通道序列的模块之间的同步可通过在每个定时器片的输出通路上(在状态位 CC4yST 和输出引脚的直接控制之间) 增加三个周期的延时来实现。该通路仅在 **CC4yINS**.MCME = 1_B 时被选择，如图 19-42 所示。

多通道序列输入同步如图 19-43 所示。为了实现状态位更新的同步，新多通道模式序列输入的采样由周期匹配或 1 匹配信号来控制。

在直接使用该同步特性时，控制多通道序列信号的模块接收一个来自 CCU4 的同步信号 CCU4x.PSy。然后，这个信号被该模块用于更新多通道序列。由于 CCU4 内部的同步方案结构，控制多通道序列的模块需要在 CCU4x.PSy 信号有效之后 4 个时钟周期内更新该序列，如图 19-43 所示。

在一个不使用外部信号控制计数方向的标准操作中，在边沿对齐模式下，用于使能序列采样的信号总是周期匹配信号，而在中心对齐模式下，总是用 1 匹配信号使能序列采样。当使用一个外部信号控制计数方向时，根据计数器是向上还是向下计数，分别使用周期匹配信号或 1 匹配信号来使能序列采样。

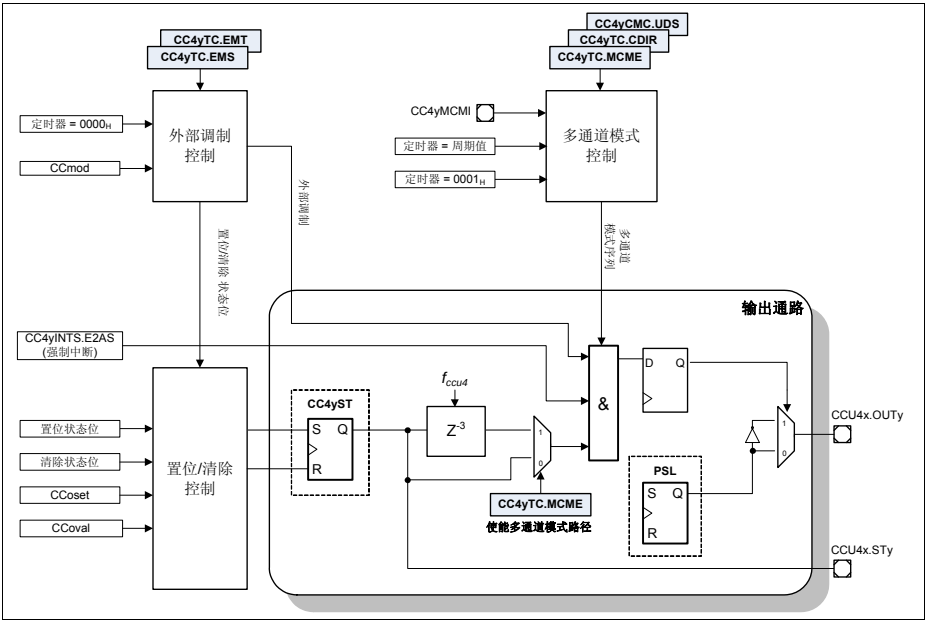


图 19-42　　CC4y 状态位和输出通路

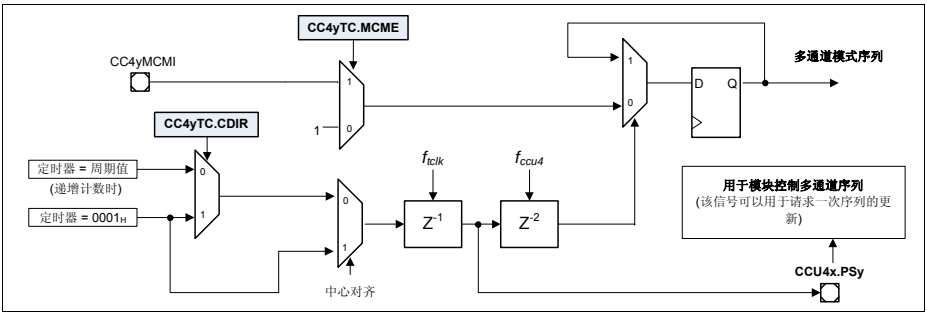


图 19-43　　多通道模式控制逻辑

19.2.9　　定时器级联

CCU4 提供了一个非常简单的机制来完成同步定时器级联。该功能可以通过设置 **CC4yTC**.TCE = 1_B 来启用。使能该功能后，用户可实现当前 CCU4 片和前一个片的级联，如图 **19-44** 所示。

注意，不可能将不相邻的片进行级联，而且定时器级联会自动将定时器片设置为边沿对齐模式。在中心对齐模式，不可能进行定时器级联。

要实现一个 64 位定时器，应在所有定时器片中设置 **CC4yTC**.TCE = 1_B (CC40 除外，因为它不包含这样的控制域)。

要实现一个 48 位定时器，应在两个相邻的定时器片中设置 **CC4yTC**.TCE = 1_B。而要实现一个 32 位定时器，应将包含最高有效位 (MSBs) 的定时器片的 **CC4yTC**.TCE 设置为 1_B。注意，应将包含最低有效位 (LSBs) 的定时器片的 TCE 位域设置为 0_B 。

在一个 CCU4 模块中，可以实现多种定时器级联组合：

- 一个 64 位定时器
- 一个 48 位定时器和一个 16 位定时器
- 两个 32 位定时器
- 一个 32 位定时器和两个 16 位定时器

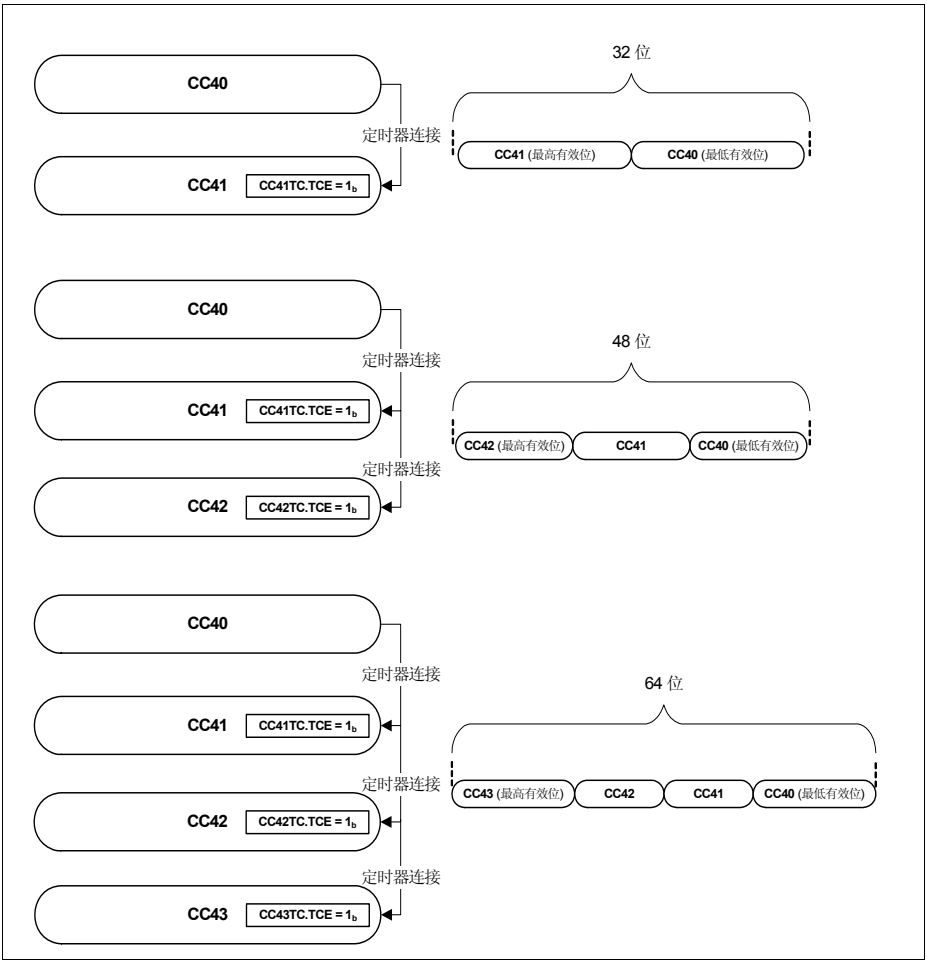


图 19-44　　定时器级联示例

每个定时器片通过一个专用的级联接口与相邻的定时器片连接。该接口不仅允许定时器计数操作的级联，而且还允许捕获和加载操作的输入触发信号的同步处理，如图 19-45 所示。

注：　在所有情况下，CC40 和 CC43 都不被认为是相邻定时器片。

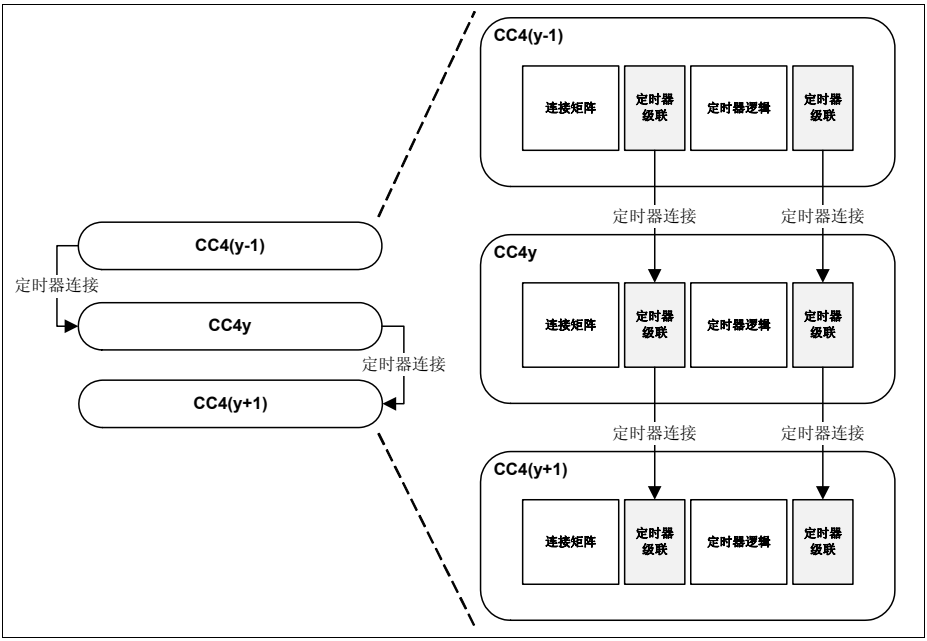


图 19-45　　定时器级联连接

定时器级联接口有 7 个信号：

- 定时器周期匹配 (CC4yPM)
- 定时器 0 匹配 (CC4yZM)
- 定时器比较匹配 (CC4yCM)
- 定时器计数方向功能 (CCupd)
- 定时器加载功能 (CCload)
- CC4yC0V 寄存器和 CC4yC1V 寄存器的定时器捕获功能 (CCcap0)
- CC4yC2V 寄存器和 CC4yC3V 寄存器的定时器捕获功能 (CCcap1)

前 4 个信号用于在定时器逻辑的输出执行同步定时级联，如**图 19-45** 所示。借助于该连接，定时器的长度可很容易地被调整为 32 位、 48 位和 64 位 (向上或向下计数)。

后 3 个信号用于对级联的定时器系统执行捕获和加载功能之间的同步连接。这意味着用户可以在第一个定时器片内通过编程设置捕获或加载功能，并将该捕获或加载触发信号同步地从最低有效位传递到最高有效位，如**图 19-46** 所示。

捕获或加载功能仅需在第一个定时器片 (保持着最低有效位的定时器片) 进行配置。从 **CC4yTC**.TCE 被置为 1_B 那一时刻起，后面定时器片的这些功能之间的连接由硬件自动完成。

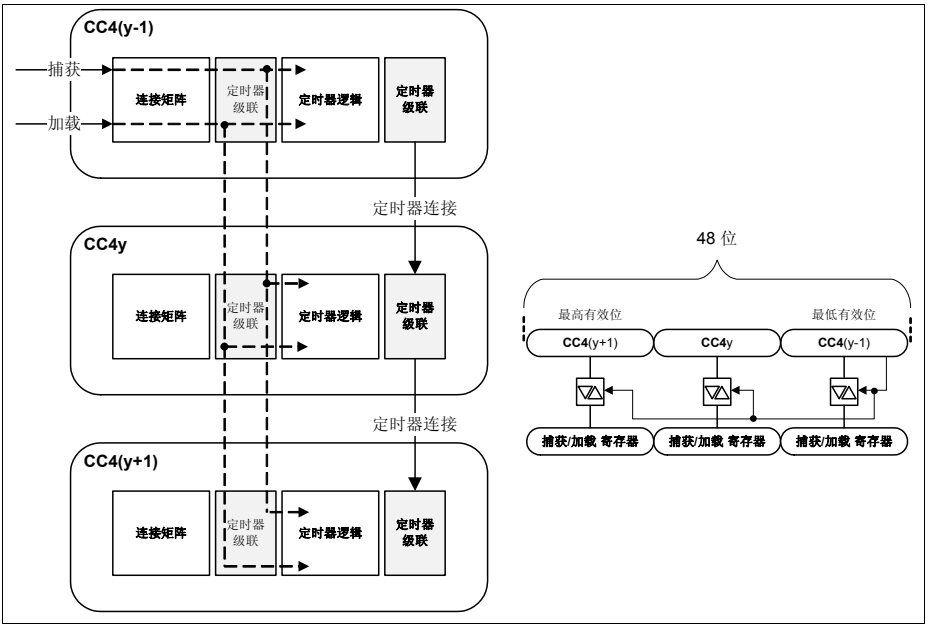


图 19-46　　捕获 / 加载定时器级联

在级联模式，要使用来自前一定时器片 (具有小一级的索引) 的周期匹配 (CC4yPM) 或 0 匹配 (CC4yZM) 作为计数器的门控信号。这意味着最高有效位的计数操作仅发生在检测到一个回绕条件时，避免了提取计数值的额外 DSP 操作。

采用相同的方法，比较匹配 (CC4yCM)、0 匹配和周期匹配由来自前一定时器片的特定信号进行门控。也就是说，定时信息被传播到所有定时器片中，这可以实现最低有效位计数和最高有效位计数之间的完全同步匹配，见图 19-47。

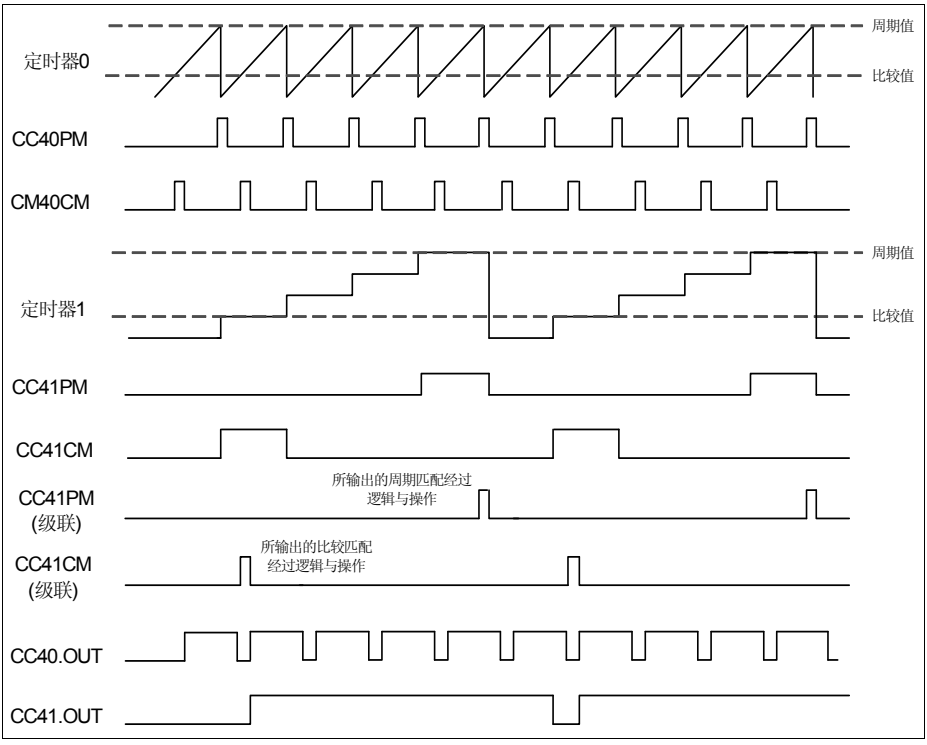


图 19-47　　32 位级联的时序图

注：　级联的定时器的计数方向需要被固定。定时器可以向上或向下计数，但是计数方向却不能够在运行中修改。

图 19-48 给出了定时器级联逻辑的概览。注意，所有机制都由 **CC4yTC**.TCE 位域单独控制。

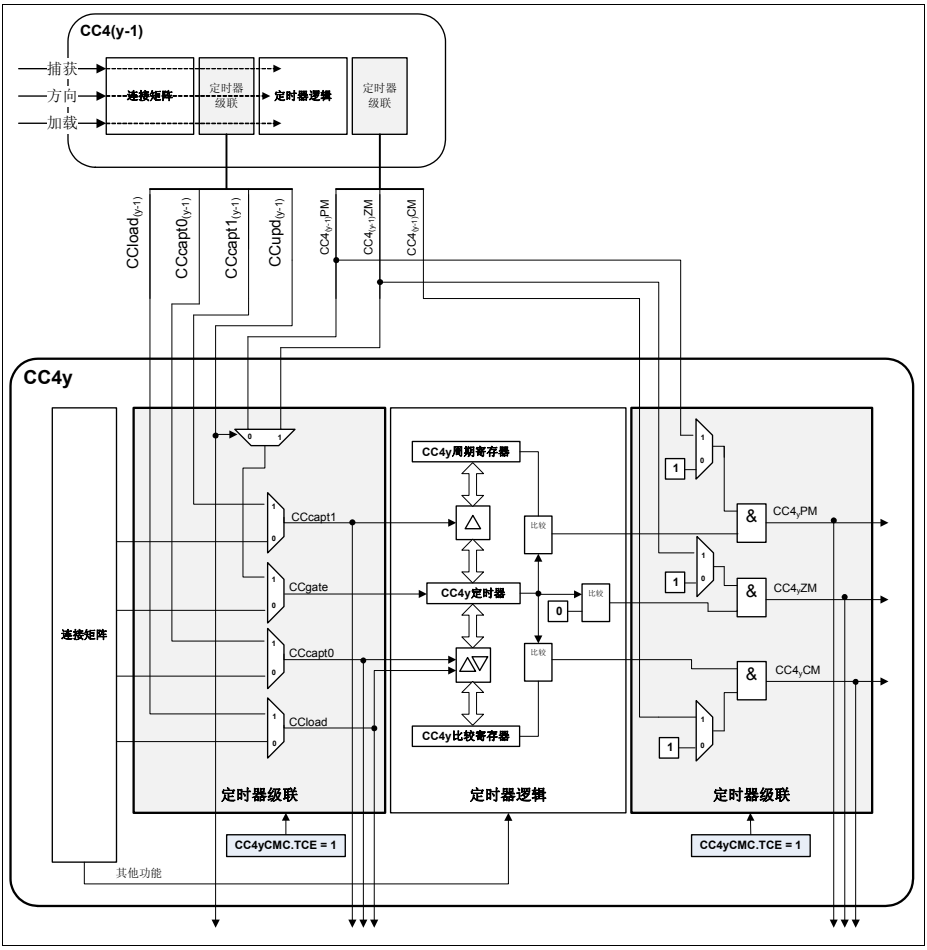


图 19-48　　定时器级联控制逻辑

19.2.10　PWM 抖动

CCU4 有一个自动 PWM 抖动插入功能。该功能可以用在不能快速更新周期 / 比较值的非常慢的控制回路，因为这样的回路在长时间运行时会损失精度。通过引入 PWM 信号的抖动，可对平均频率 / 占空比进行补偿，减小误差。

每个定时器片都包含一个抖动控制单元，如图 19-49 所示。

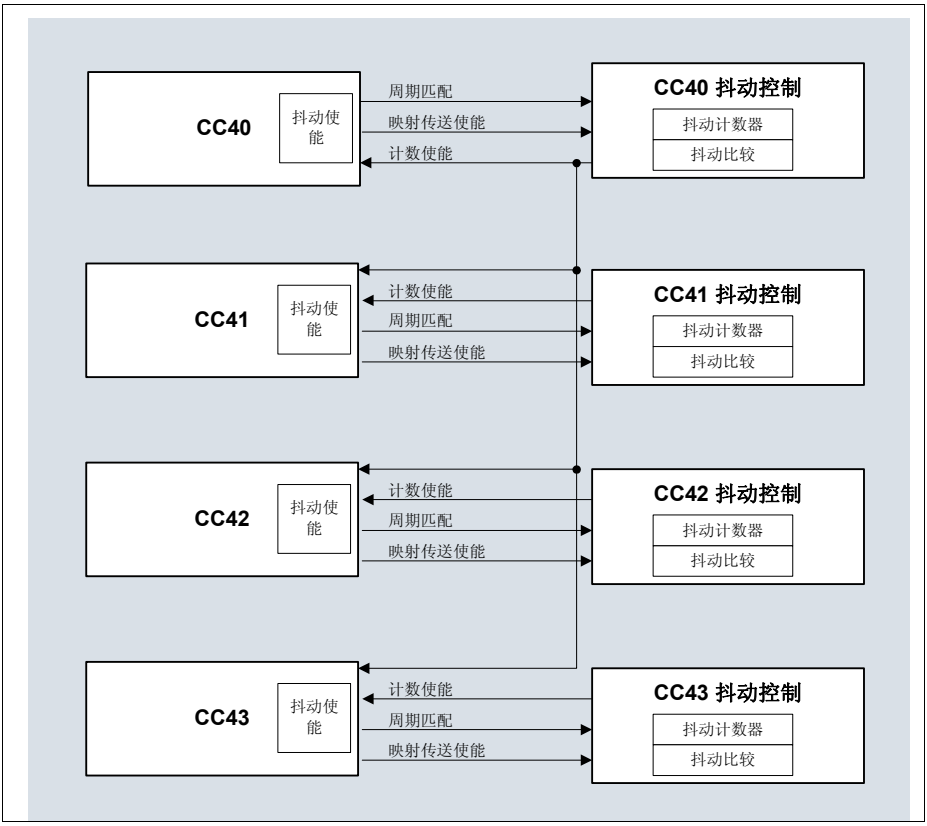


图 19-49　　抖动结构概览

抖动控制单元包含一个 4 位计数器和一个比较值寄存器。每当发生一次周期匹配时，该 4 位计数器增 1。计数器工作在位反转模式，因此在 16 个计数器周期内保持均匀增 1，如 表 19-5 所示。

表 19-5　　抖动位反转计数器

counter[3]	counter[2]	counter[1]	counter[0]
0	0	0	0
1	0	0	0

表 19-5 抖动位反转计数器 (续表)

counter[3]	counter[2]	counter[1]	counter[0]
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0
0	0	0	1
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	1

然后，计数器与一个编程值 **CC4yDIT**.DCV 相比较。如果计数器值小于该编程值，会产生一个门控信号，该信号用于将周期扩展一个时钟周期，或将比较匹配延迟一个时钟周期，或两者兼具 (由 **CC4yTC**.DITHE 域控制，详见**表 19-6**) 。

表 19-6 抖动模式

DITHE[1]	DITH[0]	模式
0	0	禁止抖动
0	1	将周期值增加一个时钟周期
1	0	将比较匹配延迟一个时钟周期
1	1	将周期值增加一个时钟周期，并将比较匹配延迟一个时钟周期

抖动比较值也有一个相关联的映射寄存器，该寄存器可以与 CC4y 的周期 / 比较寄存器同步更新。抖动单元的控制逻辑如**图 19-50** 所示。

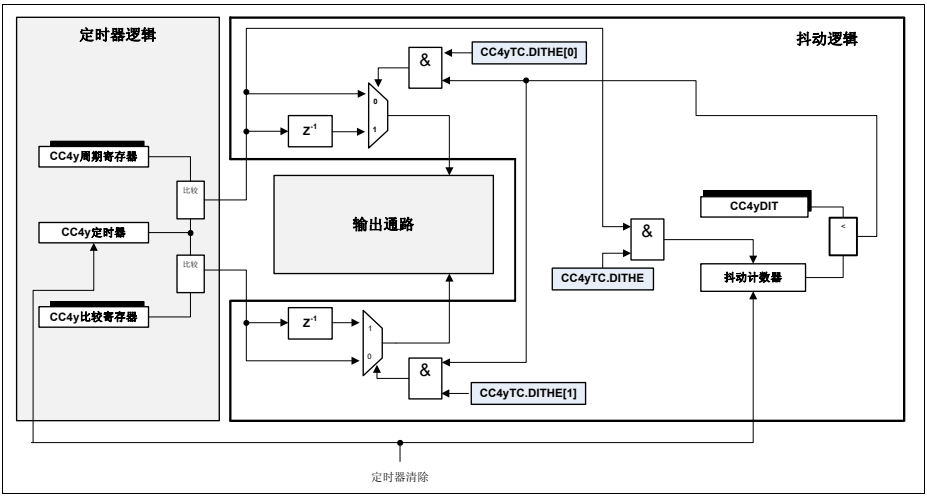


图 19-50　　抖动控制逻辑

图 19-51 到 图 19-56 展示了在边沿对齐和中心对齐这两种计数方式下采用不同抖动配置 (CC4yTC.DITHE) 的效果。在每个图中，抖动计数器用位反转方案来表示，比较值被编程为 8_H。在每个图中，变量 *T* 代表计数器的周期，变量 *d* 表示占空比 (状态位被设置为高电平)。

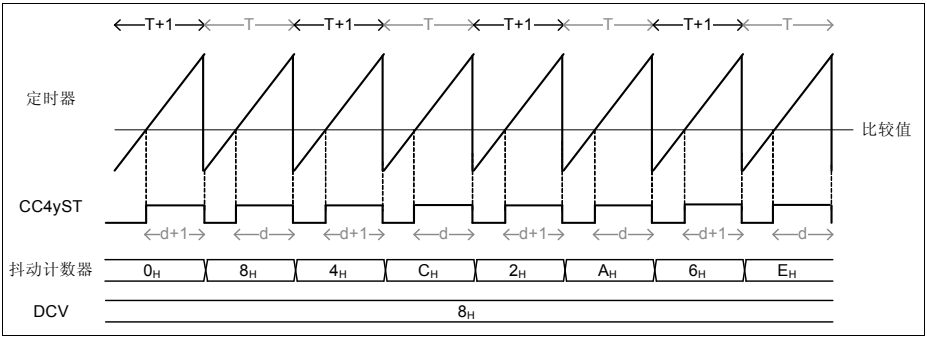


图 19-51　　边沿对齐方式下的抖动时序图 - CC4yTC.DITHE = 01_B

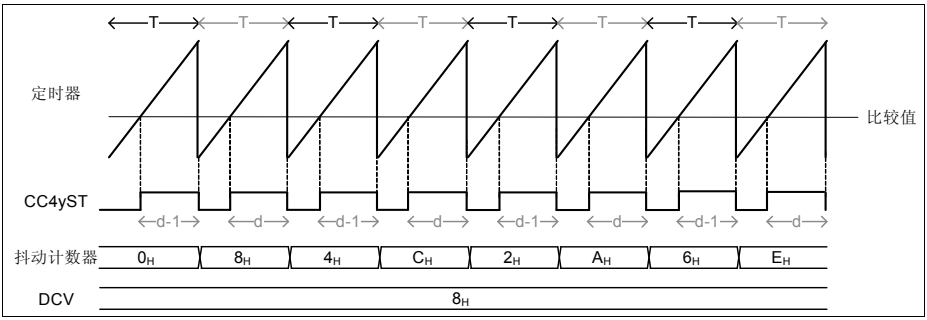


图 19-52 边沿对齐方式下的抖动时序图 - **CC4yTC.DITHE = 10_B**

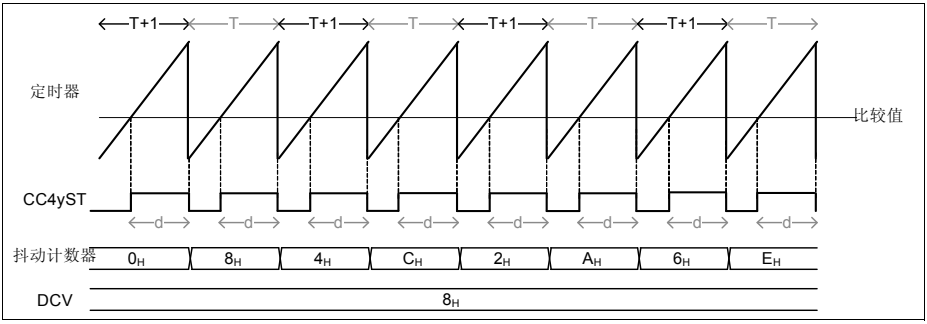


图 19-53 边沿对齐方式下的抖动时序图 - **CC4yTC.DITHE = 11_B**

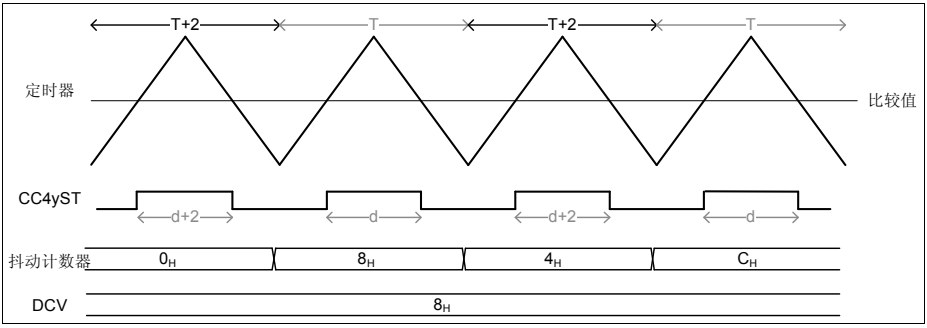


图 19-54 中心对齐方式下的抖动时序图 - **CC4yTC.DITHE = 01_B**

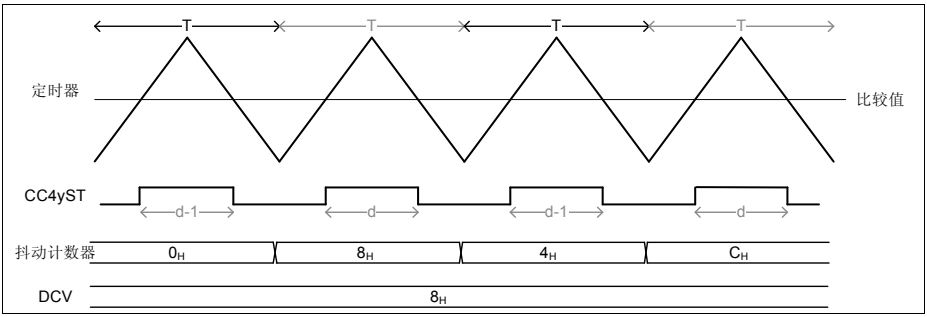


图 19-55 中心对齐方式下的抖动时序图 - **CC4yTC**.DITHE = 10_B

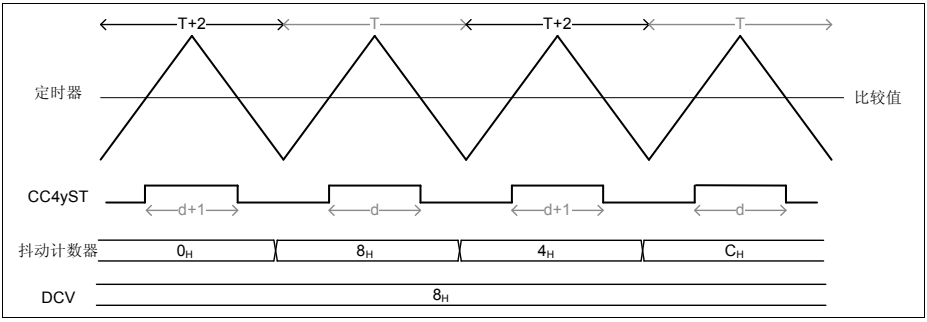


图 19-56 中心对齐方式下的抖动时序图 - **CC4yTC**.DITHE = 11_B

注： 当使用抖动时，在边沿对齐模式下不可能选择一个为 FS 的周期值。在中心对齐模式，周期值必须至少为 FS - 2。

19.2.11 预分频器

CCU4 包含一个 4 位预分频器。对于每个定时器片，该预分频器都有两种工作模式：

- 标准预分频器模式
- 浮动预分频器模式

通过用软件分别向寄存器 **GIDLC**.SPRB 和 **GIDLS**.CPRB 写入，可以置位或清除预分频器的运行位，也可以通过一个特定的定时器片的运行位来清除预分频器的运行位。在采用后一种方案的情况下，在所选定时器片的运行位被清除一个时钟周期后，预分频器的运行位被清除。要选择用哪个定时器片来执行运行位清除操作，应对 **GCTRL**.PRBC 寄存器进行编程。

19.2.11.1 标准预分频器模式

在标准预分频器模式，接入 CC4y 计数器的时钟是对模块时钟的一个固定的 N 分频， N 由 **CC4yPSC**.PSIV 寄存器中的设置值来确定。**表 19-7** 列出了可能的分频值。寄存器 **CC4yPSC**.PSIV 的值只能通过软件访问来修改。注意，每个定时器片有一个专用的预分频器值选择器 (**CC4yPSC**.PSIV)，这意味着用户可以为每个定时器片 (CC4y) 选择不同的计数器时钟。

表 19-7 定时器时钟分频选项

CC4yPSC.PSIV	结果时钟
0000_B	f_{ccu4}
0001_B	$f_{ccu4}/2$
0010_B	$f_{ccu4}/4$
0011_B	$f_{ccu4}/8$
0100_B	$f_{ccu4}/16$
0101_B	$f_{ccu4}/32$
0110_B	$f_{ccu4}/64$
0111_B	$f_{ccu4}/128$
1000_B	$f_{ccu4}/256$
1001_B	$f_{ccu4}/512$
1010_B	$f_{ccu4}/1024$
1011_B	$f_{ccu4}/2048$
1100_B	$f_{ccu4}/4096$
1101_B	$f_{ccu4}/8192$
1110_B	$f_{ccu4}/16384$
1111_B	$f_{ccu4}/32768$

19.2.11.2 浮动预分频器模式

在每个定时器片中都可以独立使用浮动预分频器模式，这可以通过设置寄存器 **CC4yTC**.FPE = 1_B 来配置。使用该模式，用户不仅能为比较操作获得更高的计数器时钟精度，而且还可以减少捕获模式的软件读访问。

除了初始值寄存器 **CC4yPSC**.PSIV 以外，浮动预分频器还包含一个比较寄存器 **CC4yFPC**.PCMP，该寄存器具有相关联的映射寄存器机制。

图 19-57 示出了当特定定时器片工作在比较模式 (无外部信号用于捕获) 时预分频器在浮动模式下的结构。在该模式，每当定时器发生一次上溢 / 下溢 (在边沿对齐模式为上溢，在中心对齐模式为下溢) 时，时钟分频值增 1_D。

在该模式，定时器的比较匹配信号和预分频器的比较匹配信号相与。每当发生定时器的比较匹配事件后，在紧接着的定时器上溢 / 下溢时刻，时钟分频值被更新为 **CC4yPSC**.PSIV 的值。

浮动预分频器比较值 **CC4yFPC**.PCMP 的映射传送遵循 **19.2.5.2 节**所述的相同规则来完成。

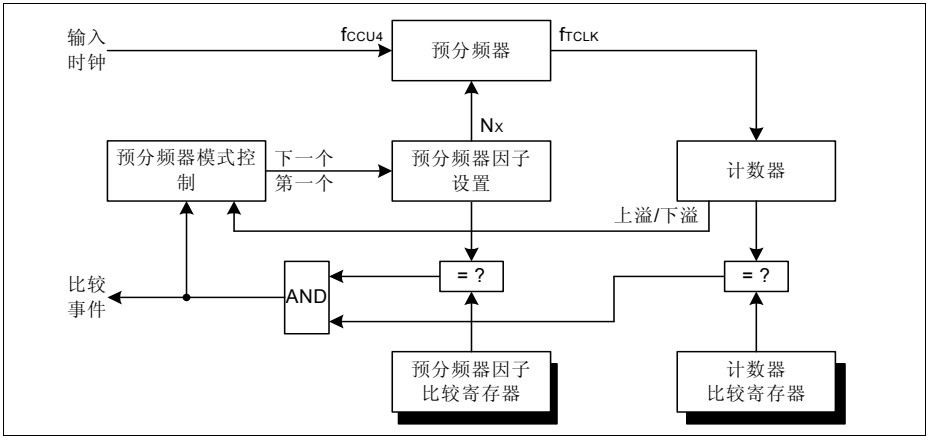


图 19-57 比较模式下浮动预分频器概览

在特定的 CC4y 工作在捕获模式 (至少有一个外部信号被用作捕获功能) 期间，每当发生一次捕获事件时，还需要存储时钟分频的当前值。浮动预分频器最多可有 4 个捕获寄存器 (捕获寄存器的最大数目由特定定时器片中所使用的捕获寄存器数来限定)。

每当定时器发生一次溢出 (在捕获模式，定时器片总是工作在边沿对齐模式) 时，时钟分频值继续增加 1_D，而每检测到一个捕获触发信号时，时钟分频值被加载为 PSIV 值。

关于浮动预分频器模式与比较模式和捕获模式结合使用的完整描述，请参见 **19.2.12.2 节**。

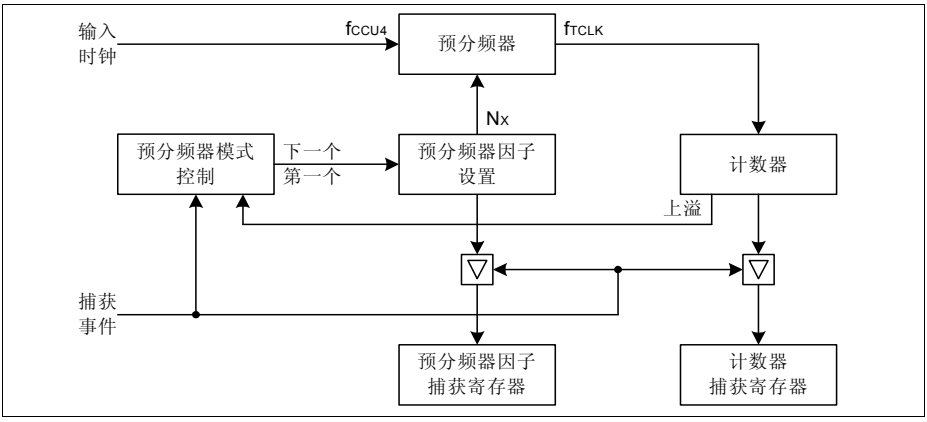


图 19-58　　捕获模式下浮动预分频器概览

19.2.12　CCU4 的使用

19.2.12.1 PWM 信号的产生

CCU4 为占空比的配置提供了一个非常灵活的范围。这个范围是 0 到 100%。

要在边沿对齐模式下产生一个占空比为 100% 的 PWM 信号，应当将比较值 **CC4yCR**.CR 编程为 0000_H，如**图 19-59** 所示。

采用相同的方法，可以在中心对齐模式产生一个占空比为100%的PWM信号，如**图19-60** 所示。

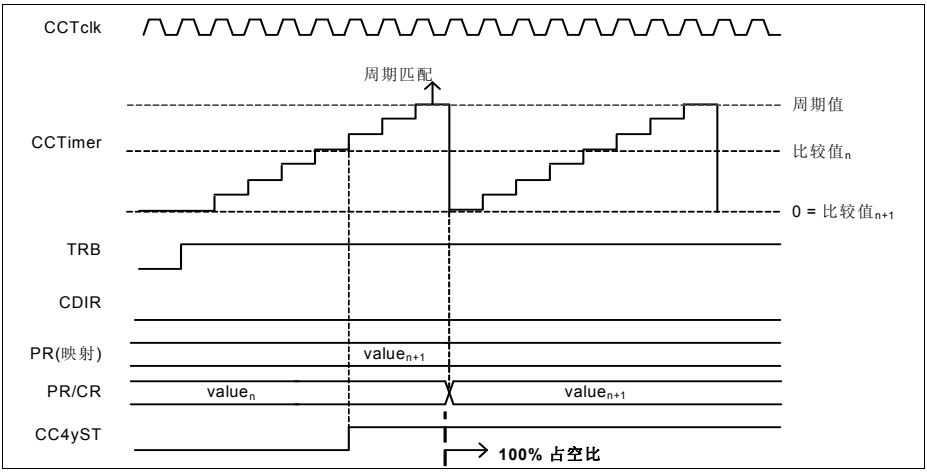


图 19-59　　占空比为 100% 的 PWM 信号 - 边沿对齐模式

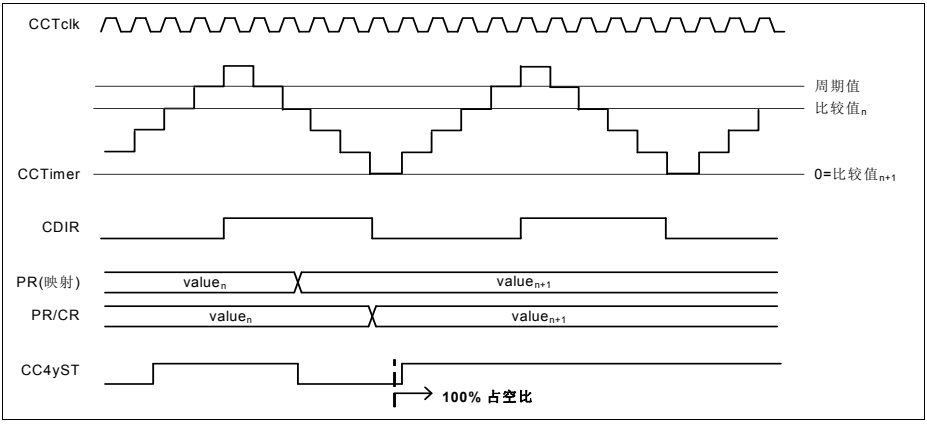


图 19-60　　占空比为 100% 的 PWM 信号 - 中心对齐模式

要在边沿对齐模式下产生一个占空比为 0% 的 PWM 信号，比较寄存器应被设置为周期寄存器的编程值加 1。在使用定时器全部 16 位能力 (从 0 计数到 65535) 的情况下，不可能在比较寄存器中设置一个大于周期值的数值，因此可获得的最小占空比为 1/FS，如图 19-61 所示。

在中心对齐模式，计数器绝不会从 0 计数到 65535_D，因为它必须超过周期寄存器中的设置值值一个时钟周期。因此，用户不会有 FS 计数器，这意味着通过在比较寄存器中设置一个大于周期寄存器编程值的数值，将总是能够产生一个占空比位 0% 的信号，如图 19-62 所示。

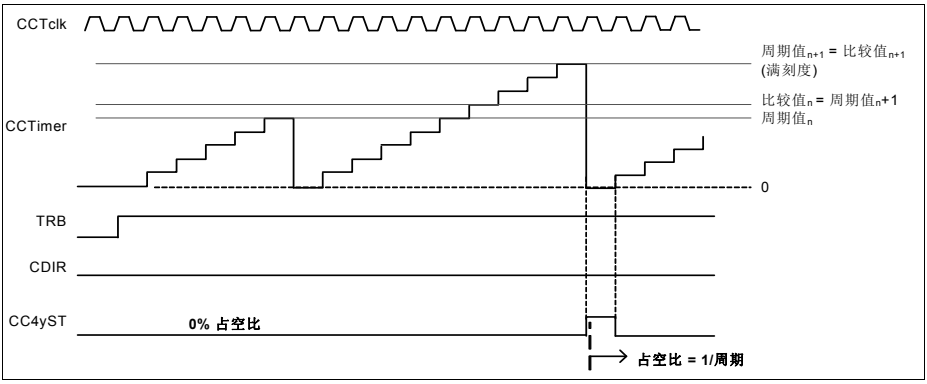


图 19-61　　占空比为 0% 的 PWM 信号 - 边沿对齐模式

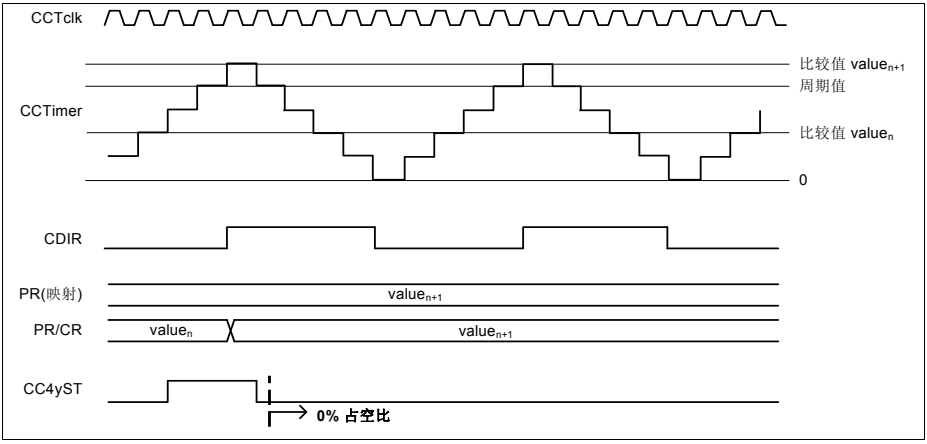


图 19-62　　占空比为 0% 的 PWM 信号 - 中心对齐模式

19.2.12.2 预分频器的使用

在标准预分频器模式，馈送给特定 CC4y 的 f_{tclk} 的频率从**表 19-7** 中进行选择，可通过在 **CC4yPSC**.PSIV 中设置所需要的值来完成。

在浮动预分频器模式， f_{tclk} 的频率可以在一个选定的时段使用**表 19-7** 中的规定值进行修改。在捕获模式，当捕获触发信号的动态性非常缓慢或未知时，该机制特别实用。

在捕获模式，每当发生一次定时器溢出时，浮动预分频器的值增 1，而当发生一个捕获事件时，浮动预分频器的值被设置为初始编程值，如**图 19-63** 所示。

当在捕获模式下使用浮动预分频器模式时，每发生一次捕获事件时，应当清空定时器，即 **CC4yTC**.CAPC = 11_B。通过将捕获模式与浮动预分频器结合使用，可以只使用一个

CCU4 定时器片，而不需要在每个周期监视定时器溢出中断事件，即使对于周期性高于 16 位的捕获信号也是如此。为此，在捕获事件产生时，用户只需要知道定时器的捕获值和预分频器的当前配置。这些值包含在每个 CC4yCxV 寄存器中。

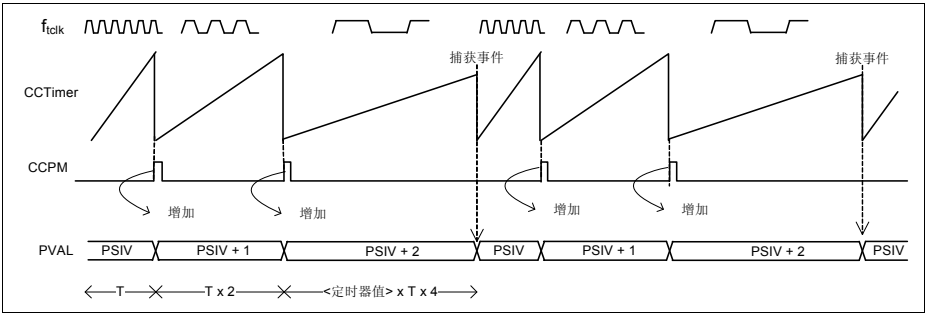


图 19-63　　浮动预分频器捕获模式的使用

当工作在比较模式时，浮动预分频器功能可用来获得一个更精细的 PWM 频率，或用于执行某种频率调制。

每当发生一次定时器上溢 / 下溢并且预分频器的当前值与寄存器 **CC4yFPC**.PCMP 中的编程值不匹配时，会执行相同的增 1 操作。

当定时器发生一次比较匹配并且预分频器的当前值等于寄存器 **CC4yFPC**.PCMP 中的编程值时，预分频器的值会在紧接着的下一次定时器上溢 / 下溢发生时刻立即被设置为初始值 **CC4yPSC**.PSIV。

在**图 19-64** 中，浮动预分频器的比较值被设置为 PSIV + 2。每当发生定时器溢出时，该预分频器的值增 1。这意味着，如果给定 f_{tclk} 为 **CC4yPSC**.PSIV 值对应的参考频率，那么，**CC4yPSC**.PSIV + 1的对应的频率 为f_{tclk}/2，**CC4yPSC**.PSIV + 2对应的频率为f_{tclk}/4。

计数器的周期变为：

$$\textit{周期} = (1/f_{tclk} + 2/f_{tclk} + 4/f_{tclk}) / 3 \qquad (19.10)$$

中心对齐模式使用相同的机制，但是为了保持上升部分和下降部分总是对称，使用定时器下溢而不是上溢，如**图 19-65** 所示。

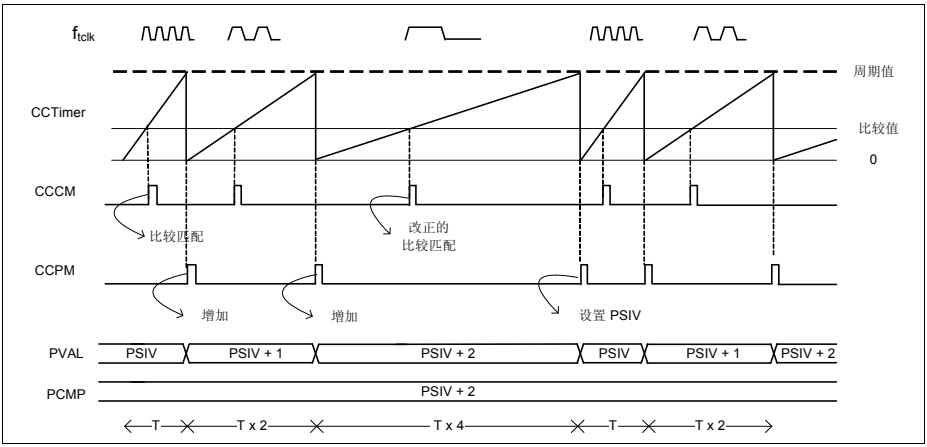


图 19-64　　浮动预分频器比较模式的使用 - 边沿对齐

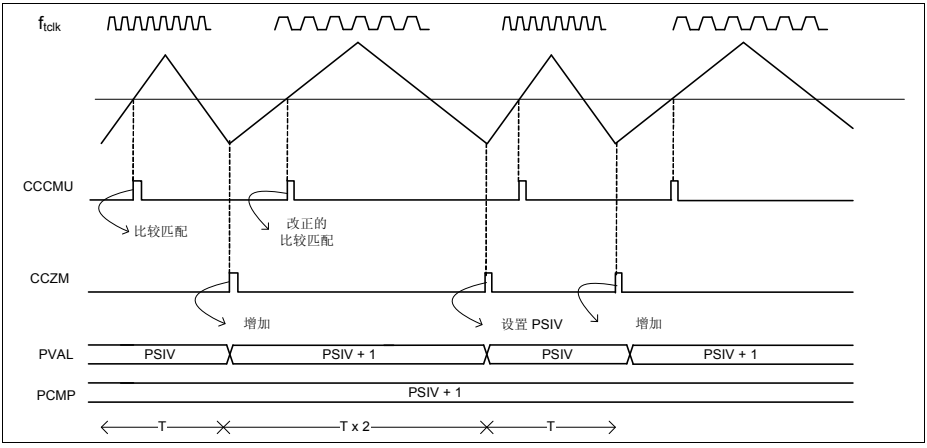


图 19-65　　浮动预分频器比较模式的使用 - 中心对齐

19.2.12.3 PWM 抖动

在比较模式，使用抖动可以获得非常精准的输出状态周期性。抖动比较寄存器 **CC4yDIT**.DCV 中的设定值与抖动计数器的当前值进行对比检查。每当抖动计数器的值小于比较值时，就会执行下列操作之一：

- 周期值被扩展一个时钟周期 - **CC4yTC**.DITHE = 01_B；边沿对齐模式
- 周期值被扩展两个时钟周期 - **CC4yTC**.DITHE = 01_B；中心对齐模式

- 在向上计数 (**CC4yTCST**.CDIR = 0_B) 期间比较匹配被延迟一个时钟周期 (这意味着状态位停留在置位状态的时间少了一个时钟周期) - **CC4yTC**.DITHE = 10_B
- 周期值被扩展一个时钟周期，在向上计数期间比较匹配被延迟一个时钟周期 - **CC4yTC**.DITHE = 11_B ；边沿对齐模式
- 周期值被扩展两个时钟周期，在向上计数期间比较匹配被延迟一个时钟周期；中心对齐模式

位反转计数器将 **CC4yDIT**.DCV 中的编程值分布在由 16 个定时器周期构成的窗口。

表 19-8 描述了位反转分布与 **CC4yDIT**.DCV 域的编程值之间的关系。标记为 “0” 的域表示在那个计数器周期将要执行的上述操作之一。

表 19-8　位反转分布

	DCV 位域															
抖动计数器	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
4	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
C	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
6	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
5	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
3	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
B	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
7	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
F	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

根据位反转分布与 **CC4yDIT**.DCV 编程值的对应关系，可得到如下的周期值和占空比：

DITHE = 01_B

$$\textit{周期} = [(16 - DCV) \times T + DCV \times (T + 1)]/16; \text{ 边沿对齐模式} \quad (19.11)$$

$$\textit{占空比} = [(16 - DCV) \times d/T + DCV \times (d+1)/(T + 1)]/16; \text{ 边沿对齐模式} \quad (19.12)$$

周期= [(16 - DCV) x *T* + DCV x (*T* + 2)]/16; 中心对齐模式 (19.13)

占空比= [(16 - DCV) x *d*/*T* + DCV x (*d*+2)/(*T* + 2)]/16; 中心对齐模式 (19.14)

DITHE = 10_B

周期= *T* ; 边沿对齐模式 (19.15)

占空比= [(16 - DCV) x *d*/*T* + DCV x (*d*-1)/*T*]/16; 边沿对齐模式 (19.16)

周期= *T* ; 中心对齐模式 (19.17)

占空比= [(16 - DCV) x *d*/*T* + DCV x (*d*-1)/*T*]/16; 中心对齐模式 (19.18)

DITHE = 11_B

周期= [(16 - DCV) x *T* + DCV x (*T* + 1)]/16; 边沿对齐模式 (19.19)

占空比= [(16 - DCV) x *d*/*T* + DCV x *d*/(*T* + 1)]/16; 边沿对齐模式 (19.20)

周期= [(16 - DCV) x *T* + DCV x (*T* + 2)]/16; 中心对齐模式 (19.21)

占空比= [(16 - DCV) x *d*/*T* + DCV x (*d*+1)/(*T* + 2)]/16; 中心对齐模式 (19.22)

其中：

T - 信号的初始周期，详见 **19.2.5.1 节**。

d - 信号的初始占空比，详见 **19.2.5.1 节**。

19.2.12.4 捕获模式的使用

每个定时器片可以使用 2 个或 4 个捕获寄存器。只使用两个捕获寄存器意味着仅有一个事件与一个捕获触发信号连接。要使用 4 个捕获寄存器，需要将两个捕获触发信号映射到一个事件（可以是同一个信号的不同边沿，也可以是两个不同信号），或需要将 **CC4yTC**.SCE 域置 1 以连接这 4 个捕获寄存器。

对于捕获触发信号 1 或捕获触发信号 0，用于捕获的内部定时器片机制是相同的。

不同的捕获事件 - SCE = 0_B

捕获触发信号 1 (CCcapt1) 被指定用于捕获寄存器 2 (**CC4yC2V**) 和捕获寄存器 3 (**CC4yC3V**)，而捕获触发信号 0 (CCcapt0) 被指定用于捕获寄存器 1 (**CC4yC1V**) 和捕获寄存器 0 (**CC4yC0V**)。

每当发生 CCcapt0 事件时，定时器值被保存到捕获寄存器 **CC4yC1V**，而捕获寄存器 **CC4yC1V** 的值被传送到捕获寄存器 **CC4yC0V**。

每当发生 CCcapt1 事件时，定时器值被保存到捕获寄存器 **CC4yC3V**，而捕获寄存器 **CC4yC3V** 的值被传送到捕获寄存器 **CC4yC2V**。

捕获 / 传送机制只能发生在特定寄存器未满的情况下。当接收到一个新值时，寄存器变为满状态，而在软件读回该值后，寄存器变为空状态。

每当软件回读 **CC4yC0V**、**CC4yC1V**、**CC4yC2V** 或 **CC4yC3V** 寄存器时，满标志即被清除。通过使能一个连接到特定事件的中断源，可以告知软件有一个新的捕获触发信号。这意味着每当发生一次捕获时就会产生一个中断脉冲。

在使用浮动预分频器模式的情况下，时钟分频的当前值也保存在捕获寄存器 (CC4yCxV) 中。

图 19-66 给出了如何在定时器片中使用捕获 / 传送的一个示例，该定时器片使用一个外部信号作为计数功能 (用来测量一个旋转设备的速度)，使用一个等距的捕获触发信号来记录速度计算需要的时间戳 (绘制了两个定时器波形，一个用来说明每次捕获事件中的定时器清除操作，另一个是清除功能无效的情况) 。

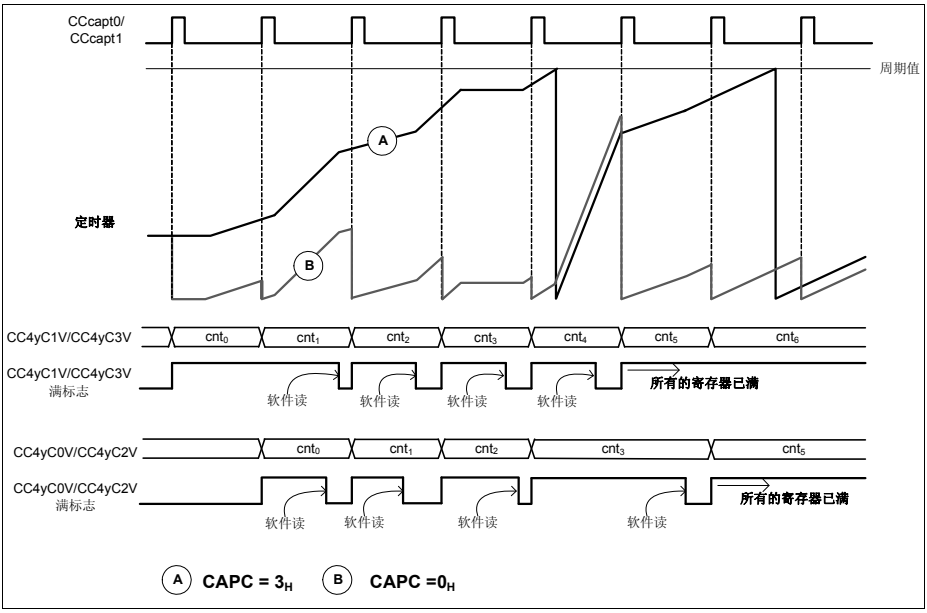


图 19-66　　捕获模式的使用 - 单通道

相同的捕获事件 - SCE = 1_B

如果 **CC4yTC**.SCE 被设置为 1_B，则所有 4 个捕获寄存器都被链接到一起，模拟一个深度 4 的 FIFO 结构。在这种情况下，仅捕获触发信号 1 (CCcapt1) 用于执行一次捕获事件。

作为该模式的一个例子，可以考虑这样一种情况：一个定时器片工作在捕获模式， SCE = 1_B，使用另一个外部信号来控制计数。该定时器片可以根据计数信号的频率以不同的速度递增。

使用另一个定时器片来控制捕获触发信号，记录捕获的时间戳。

从**图 19-67** 中可以看到这种简单机制工作过程。定时器片 0 的 CC40ST 输出被用作定时器片 CC41 的捕获触发信号 (在上升沿和下降沿都有效)。CC40ST 输出被用作已知的时基标记，而用于捕获操作的定时器片由外部事件 (例如，外部计数) 来控制。

由于有4个可用的捕获寄存器，每当软件回读完整的数值集时，可以测量出3个速度廓线。

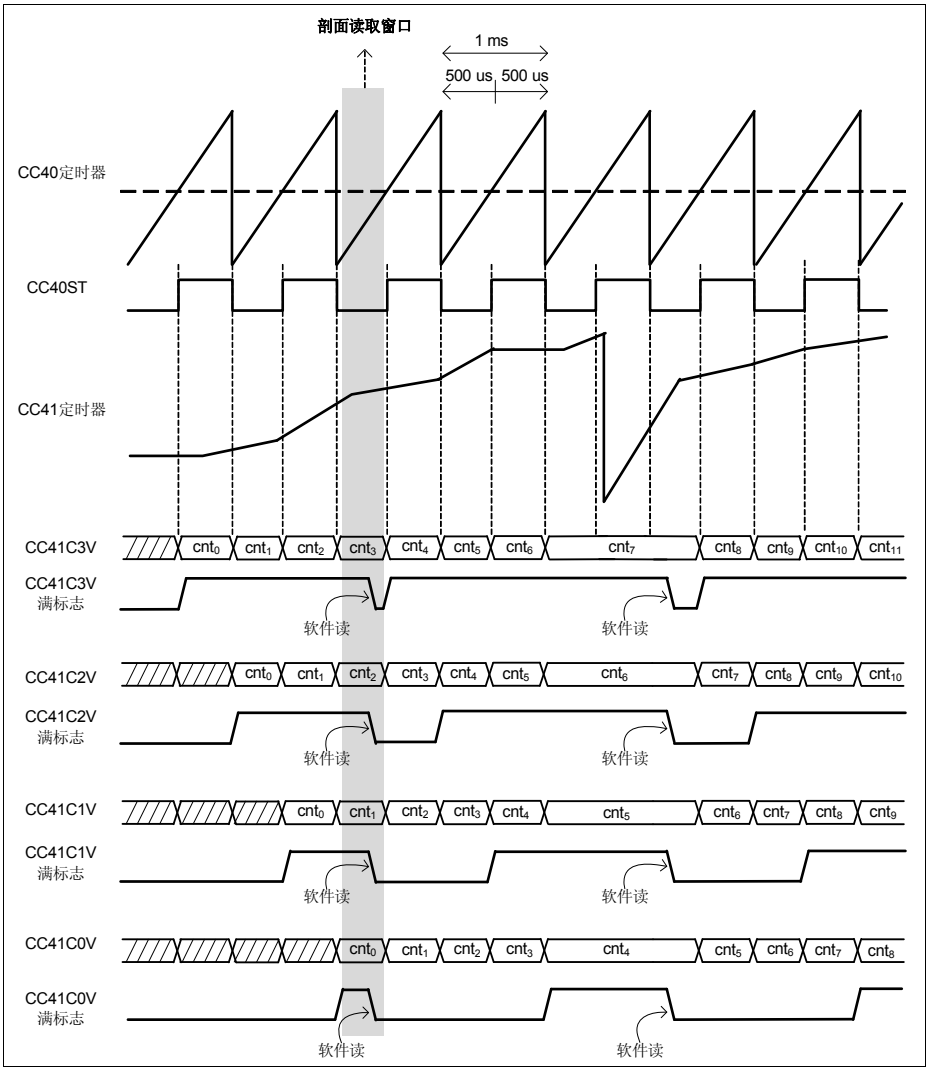


图 19-67　　三个捕获廓线 - CC4yTC.SCE = 1_B

要计算图 19-67 中的三个不同廓线，需要在指定的读窗口期间读取 4 个捕获寄存器。此后，即可完成廓线计算：

廓线 1 = $CC41C1V_{info}$ - $CC41C0V_{info}$

廓线 2 = $CC41C2V_{info}$ - $CC41C1V_{info}$

廓线 3= $CC41C3V_{info}$ - $CC41C2V_{info}$

注： 这仅是一个示例，还可以实现更多的定时器配置和软件回路。

高动态捕获

在有些情况下，捕获触发信号的动态性可能随着时间变化很大。这就要求软件需要为最坏的情况做好准备，例如捕获触发信号的频率可能会非常高。在需要逐个周期进行计算 (在每次发生捕获触发时计算) 的应用场合，这个限制需要由软件来满足。然而，对于不需要每个周期都进行计算的应用，软件可以周期性地回读 FIFO 数据寄存器并读取目前为止所有已经捕获到的数据，如**图 19-68** 所示。

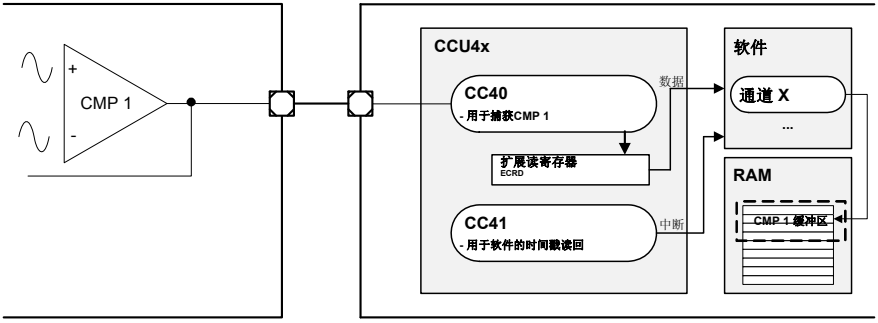


图 19-68 高动态捕获 - 使用软件控制的时间戳

在这种情况下，每当时间戳定时器 (该定时器的周期性也可以在运行时调整) 触发一个中断时，软件 /CPU 会回读完整的捕获寄存器集 (2 个或是 4 个，取决于所选择的配置)。

由于每个捕获寄存器都提供一个满标志状态位，软件 /CPU 总是可以回读完整的捕获寄存器集。在进行数据处理时，软件要检查这个满标志，以确认该值是否需要被处理。

这个 FIFO 回读功能也可以用于一些对系统产生高负载的应用场合，但不能保证回读捕获数据的访问时间是固定的。

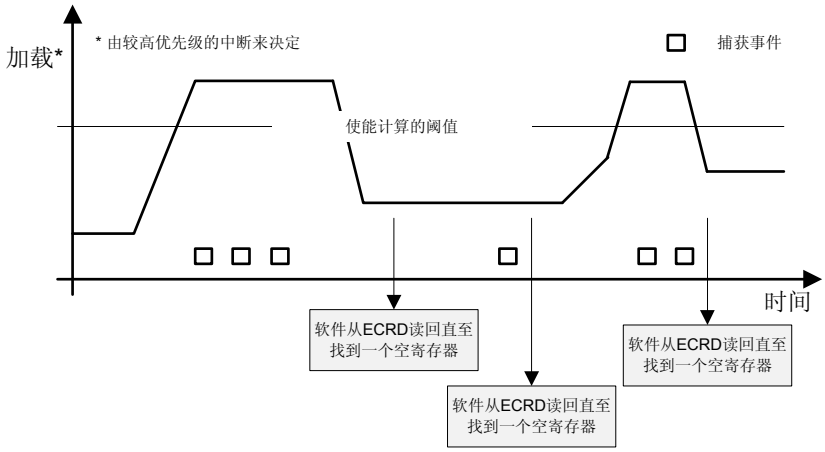


图 19-69　　高负载期间的扩展回读

捕获分组

在需要多个捕获定时器并且捕获程序的优先级并不意味着对每个事件都需要进行逐个周期计算的应用场合，将同一个 CCU4x 单元中的所有定时器进行分组会可能更为合适，如图 19-70 所示。

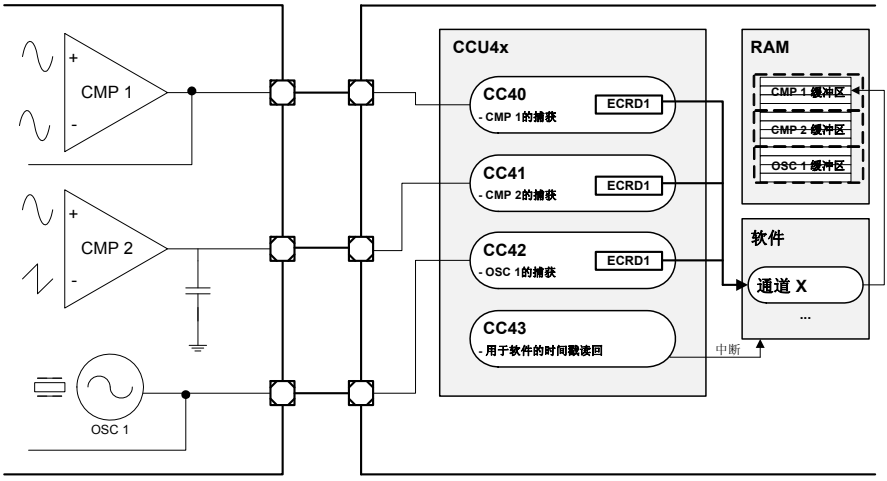


图 19-70　　采用扩展回读的捕获分组

通过将用于捕获的定时器片的 ECM 位域置 1，可使扩展回读模式总能以正确的捕获顺序 (从最老的数据到最新的数据) 回读数据。然后，使用一个时间戳定时器来触发软件 /CPU 去回读定时器片中所有捕获到的数据。

每当检测到中断时，软件 /CPU(在本例中) 会回读所有定时器片的完整捕获寄存器集 (通过 ECRD 地址)。由于针对每个数据读取都有一个满标志指示器，所以软件 /CPU 能够从所有定时器片回读完整的捕获寄存器集。这就允许固定的存储器分配，使分配的存储器与捕获寄存器的数量相同，如图 19-71 所示 (在本例中，每个定时器片使用 4 个捕获寄存器)。

每个 ECRD 数据的首部有一个附加的丢失值位域 (LCV)，它指示在两次读操作之间是否有任何捕获触发信号丢失(如果捕获触发信号比回读数据的程序快，就会发生这种情况)。

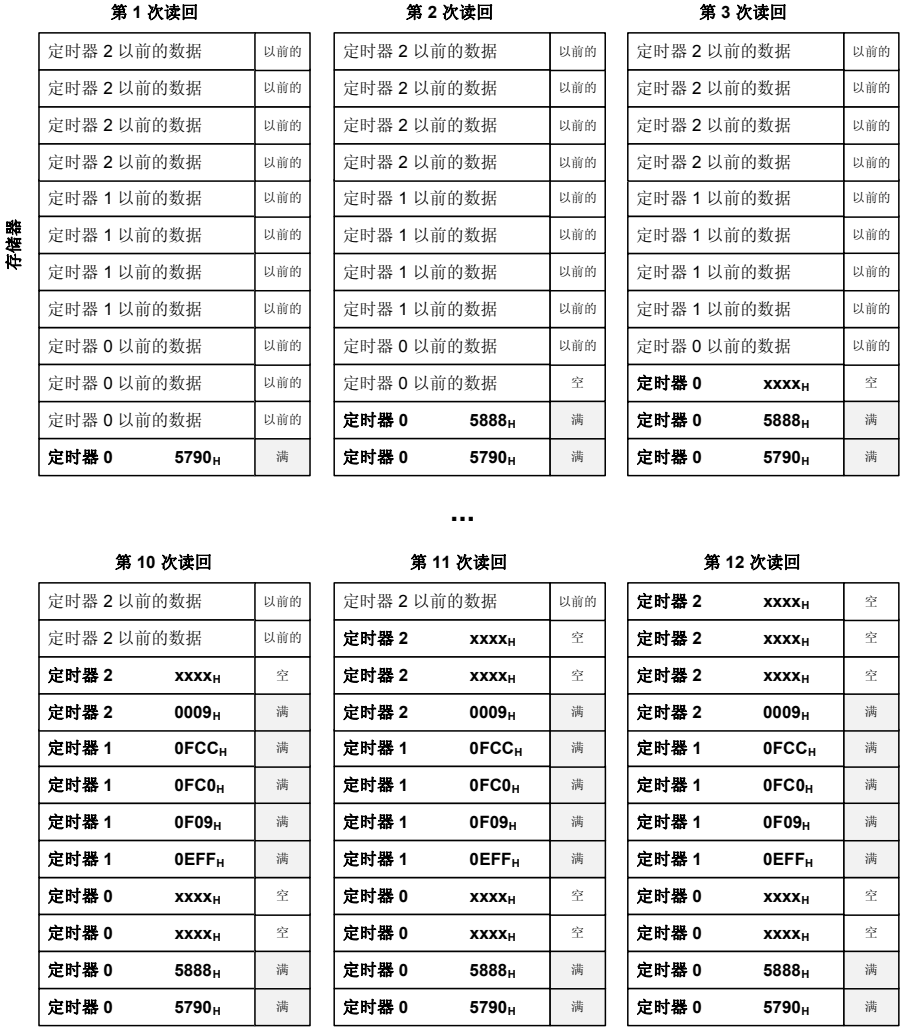


图 19-71　　扩展回读的存储器结构

19.3　　服务请求的产生

每个 CCU4 定时器片都有一个如图 19-72 所示的中断结构。寄存器 **CC4yINTS** 是中断源的状态寄存器。通过分别写 **CC4ySWS** 和 **CC4ySWR** 寄存器中的特定位，可以用软件置位或清除每个专用的中断源。

通过寄存器 **CC4yINTE** 可以使能或禁止每个中断源。一个被使能的中断源总是会在服务请求线上产生一个脉冲，即使特定状态位未被清零。**表 19-9** 描述了每个定时器片的中断源。

向上计数期间的周期匹配和向下计数时的 1 匹配作为两个中断源被或在一起。该机制同样适用于向上计数期间的比较匹配和向下计数期间的比较匹配。

外部事件的中断源根据 **CC4yINS**.EVxEM 中设定的配置直接连接。如果一个事件被编程为在两个边沿都有效，则当检测到外部信号的任何跳变时都将产生中断服务请求脉冲。如果该事件被连接到一个电平功能， **CC4yINS**.EVxEM 仍可被编程为使能服务请求脉冲。 当定时器片进入陷阱状态时，陷阱事件不需要任何额外的配置即可产生服务请求脉冲。

表 19-9 中断源

信号	描述
CCINEV0_E	来自事件选择器的事件 0 边沿信息。在外部信号应该触发中断时使用。
CCINEV1_E	来自事件选择器的事件 1 边沿信息。在外部信号应该触发中断时使用。
CCINEV2_E	来自事件选择器的事件 2 边沿信息。在外部信号应该触发中断时使用。
CCPM_U	向上计数期间的周期匹配
CCCM_U	向上计数期间的比较匹配
CCCM_D	向下计数期间的比较匹配
CCOM_D	向下计数期间的 1 匹配
陷阱状态设置	进入陷阱状态。会置位 E2AS。

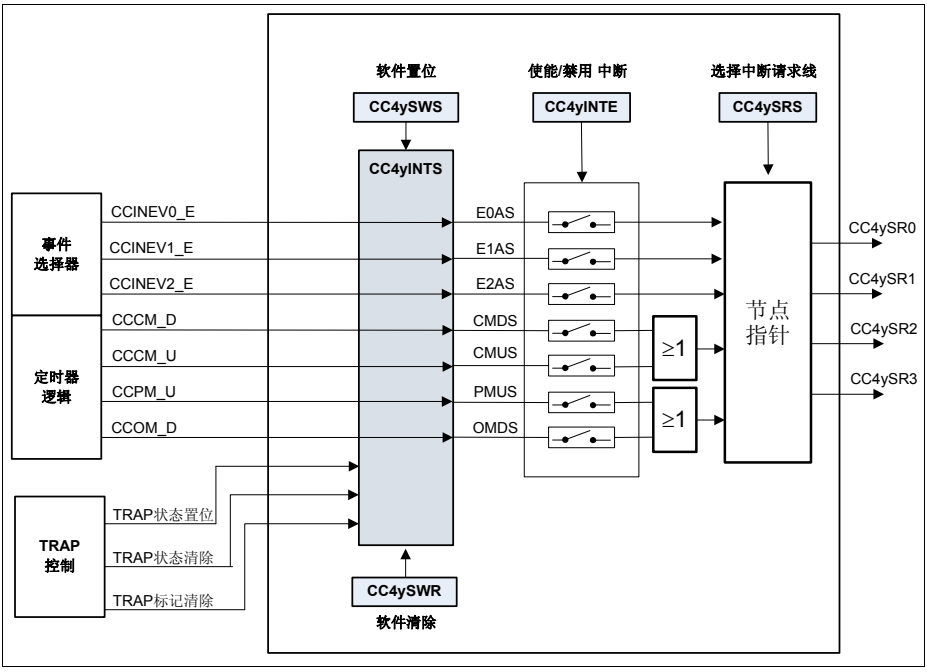


图 19-72　　定时器片中断结构概览

然后，每个中断事件都可以被转发到定时器片的 4 个服务请求线之一，如图 19-73 所示。**CC4ySRS** 的设置值控制将哪一个中断事件映射到哪一条服务请求线。

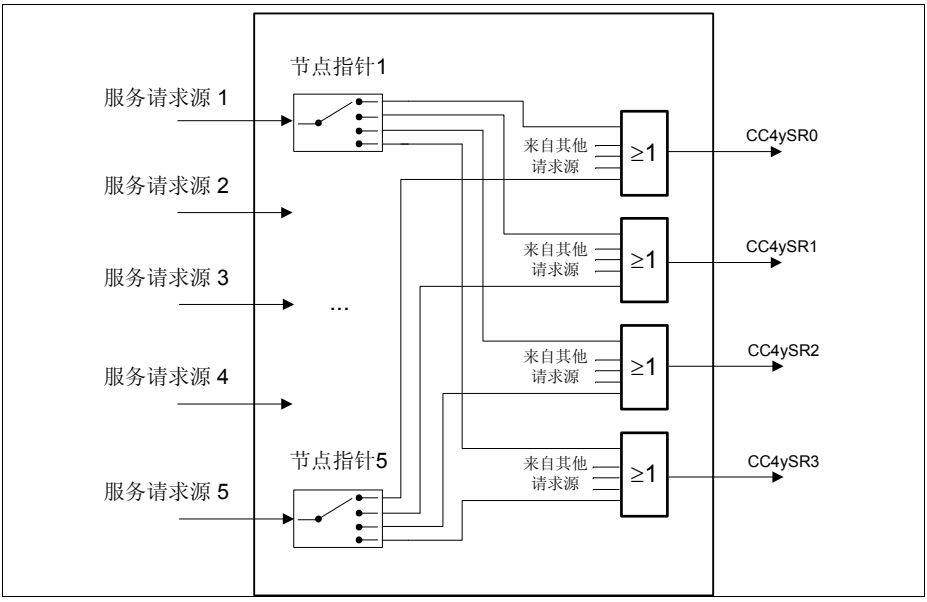


图 19-73　　定时器片中断节点指针概览

在 CCU4 的内核中，每个定时器片的 4 条服务请求线被或在一起，如图 19-74 所示。这意味着每个 CCU4 仅有 4 条服务请求线，可以使每条线都有来自不同定时器片的中断请求。

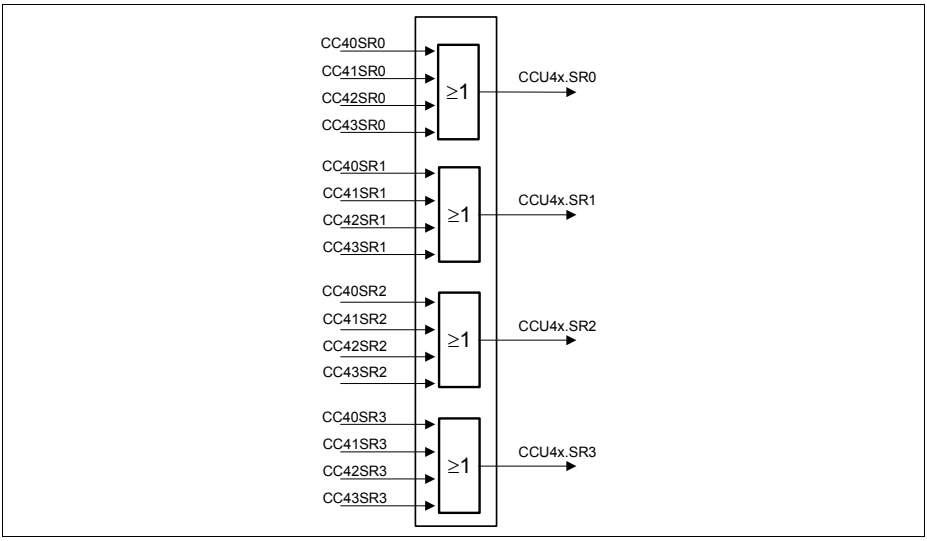


图 19-74　　CCU4 服务请求概览

19.4　　调试行为

在挂起模式，所有定时器片及预分频器的功能时钟都被停止。寄存器仍可被 CPU 访问 (只读)。该模式对于调试目的非常有用，例如，为了获得内部值的快照应冻结当前器件的状态。在挂起模式，所有定时器片的计数器都被停止。挂起模式对于寄存器位来说是非侵入式的。这意味着当进入或离开挂起模式时，寄存器位不会被硬件修改。

可以在内核级通过域 **GCTRL**.SUSCFG 来配置进入挂起模式。

CCU4 模块仅在挂起模式信号变为无效后才能工作。

19.5　　电源、复位和时钟

下面的小节描述 CCU4 的工作条件、特性和时序要求。所有时序信息都基于模块时钟 f_{CCU4}。

19.5.1　　时钟

模块时钟

CCU4 的模块时钟在 SCU 一章中被描述为 f_{PCLK}。

CCU4 模块的总线接口时钟在 SCU 一章中被描述为 f_{MCLK}。

通过 **GSTAT** 寄存器可以禁止 CCU4 的模块时钟，不过，在不同 CCU4 实例中也许存在对 f_{CCU4} 的依赖。关于产品时钟方案的详细描述，请参见 SCU 一章。

如果不同的 IP 实例中存在对模块时钟的依赖，可以通过禁止预分频器 (**GSTAT**.PRB = 0_B) 来禁止特定 CCU4 内部的模块时钟。

外部时钟

可以使用一个外部时钟作为预分频器的时钟源，因而也是所有定时器片 CC4y 的时钟源。该外部时钟源可以被连接到一个 CCU4x.CLK[C...A] 输入。

该外部信号源仍然与 f_{ccu4} 同步。

表 19-10　外部时钟工作条件

参数	符号	值			单位	备注 / 测试条件
		最小值	典型值	最大值		
频率	f_{eclk}	–	–	$f_{ccu4}/4$	MHz	
接通时间	ton_{eclk}	$2T_{ccu4}$ 1)2)	–	–	ns	
断开时间	$toff_{eclk}$	$2T_{ccu4}$ 1)2)	–	–	ns	仅使用上升沿

1) 仅在该信号之前与 fccu4 时钟 (或一个同步时钟) 不同步或不是由 ccu4 时钟产生时有效

2) 不强求占空比为 50%

19.5.2　电源

CCU4 位于内核电源域内，因此对于上电或下电顺序不需要做任何特殊考虑。关于对不同电源域的解释，请参见 SCU (系统控制单元) 一章。

通过禁止 CCU4 自身的内部时钟，可以实现 CCU4 的内部断电模式。为此，应将 **GSTAT** 寄存器设置为默认的复位值 (通过空闲模式设置寄存器 **GIDLS** 来实现)。

19.6　初始化和系统依赖

19.6.1　初始化序列

使用 CCU4 的应用程序应当采用如下的初始化序列：

第 1 步：通过特定的 SCU 寄存器 CGATCLR0 来使能 CCU4 时钟。

第 2 步：通过向 **GIDLC**.SPRB 域写 1_B 来使能预分频器时钟。

第 3 步：配置全局 CCU4 寄存器 **GCTRL**。

第 4 步：配置与所需定时器片功能相关的所有寄存器，包括中断 / 服务请求配置。

第 5 步：如果需要，为一个定时器片的一个特定比较通道状态配置起始值，通过向特定的 **GCSS**.SyTS 写 1_B 来完成。

第 6 步：通过向特定的 **GIDLC**.CSyI 写 1_B 来使能特定的定时器片 CC4y。

第 7 步：对于应该由软件来同步启动的所有定时器片，应寻址位于 SCU 中的特定系统寄存器 CCUCON 来使能定时器的同步启动。在此之前，需要将 SCU.GLCSTxx 输入信号配置为启动功能，详见 **19.2.7.1 节**。

19.6.2 系统相关性

每个 CCU4 都可能与模块时钟和总线时钟频率有不同的相关性。有关这种相关性的详细描述请参见 SCU 和系统架构这两章。

就不同的时钟工作频率而言，多个外设之间也可能存在相关性。在配置 CCU4 与其他外设之间的连接之前，应先处理好这种相关性。

为了更好的 CCU4 和系统操作，必须要考虑以下几点：

- CCU4 模块时钟最多是模块总线接口时钟的两倍。
- CCU4 的模块输入触发信号不能超过模块时钟频率 （如果触发信号在器件内部产生）。
- CCU4 的模块输入触发信号不能超过 **19.5.1 节**所述的频率。
- 用作其他模块之触发信号/功能的CCU4输出的频率必须要在端点处进行交叉检查，。
- 复位时使用和移除 CCU4 会在其他模块中引起不希望的操作。当模块将 CCU4 输出用作触发信号 / 功能时就会发生这种情况。

19.7 寄存器

寄存器概述

寄存器的绝对地址通过下面的加法计算：

模块基地址 + 偏移地址

表 19-11 寄存器地址空间

模块	基地址	结束地址	备注
CCU40	48040000_H	$4804FFFF_H$	

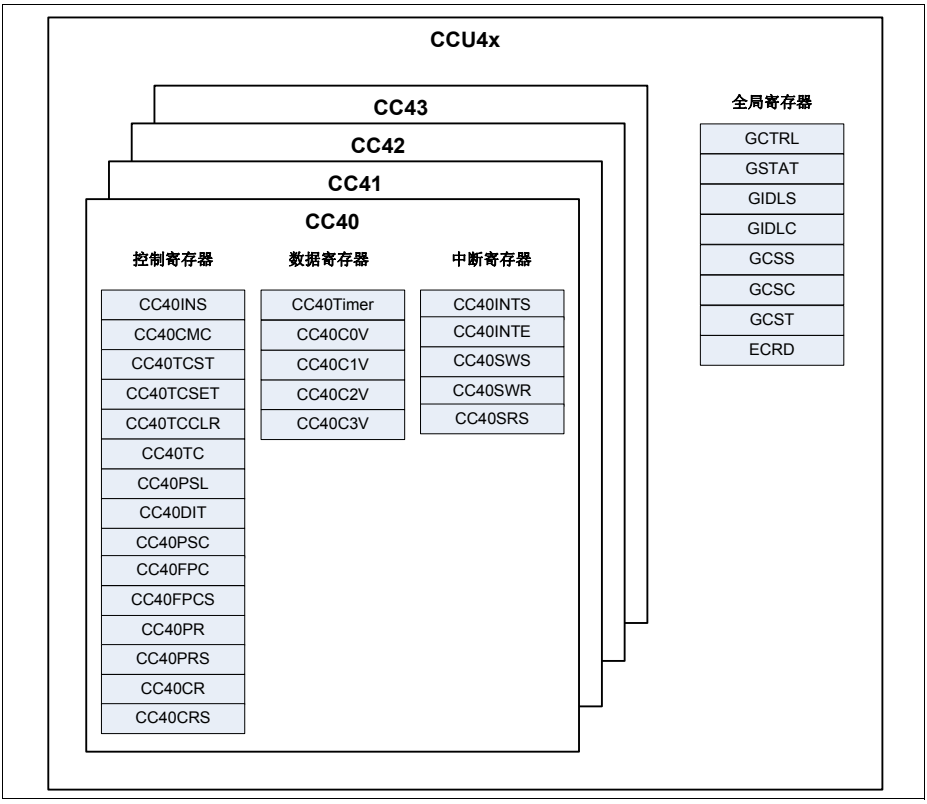


图 19-75　　CCU4 寄存器概览

表 19-12　　CCU4 寄存器一览表

简称	描述	偏移地址[1)]	访问方式		描述见
			读	写	
CCU4 全局寄存器					
GCTRL	模块综合控制寄存器	0000_H	U, PV	U, PV	页 19-81
GSTAT	通用定时器片状态寄存器	0004_H	U, PV	BE	页 19-83
GIDLS	通用空闲使能寄存器	0008_H	U, PV	U, PV	页 19-85
GIDLC	通用空闲禁止寄存器	$000C_H$	U, PV	U, PV	页 19-86
GCSS	通用通道置位寄存器	0010_H	U, PV	U, PV	页 19-87

表 19-12 CCU4 寄存器一览表 (续表)

简称	描述	偏移地址[1]	访问方式		描述见
			读	写	
GCSC	通用通道清除寄存器	0014_H	U, PV	U, PV	页 19-90
GCST	通用通道状态寄存器	0018_H	U, PV	BE	页 19-92
MIDR	模块标识寄存器	0080_H	U, PV	BE	页 19-94
CC40 寄存器					
CC40INS	输入选择器单元配置	0100_H	U, PV	U, PV	页 19-95
CC40CMC	连接矩阵配置	0104_H	U, PV	U, PV	页 19-97
CC40TST	定时器运行状态	0108_H	U, PV	BE	页 19-99
CC40TCSET	定时器运行置位	$010C_H$	U, PV	U,PV	页 19-100
CC40TCCLR	定时器运行清除	0110_H	U, PV	U, PV	页 19-101
CC40TC	通用定时器配置	0114_H	U, PV	U, PV	页 19-102
CC40PSL	输出被动电平配置	0118_H	U, PV	U, PV	页 19-106
CC40DIT	抖动配置	$011C_H$	U, PV	BE	页 19-107
CC40DITS	抖动映射寄存器	0120_H	U, PV	U, PV	页 19-108
CC40PSC	预分频器配置	0124_H	U, PV	U, PV	页 19-108
CC40FPC	预分频器比较值	0128_H	U, PV	U, PV	页 19-109
CC40FPCS	预分频器映射比较值	$012C_H$	U, PV	U, PV	页 19-110
CC40PR	定时器周期值	0130_H	U, PV	BE	页 19-110
CC40PRS	定时器周期映射值	0134_H	U, PV	U, PV	页 19-111
CC40CR	定时器比较值	0138_H	U, PV	BE	页 19-112
CC40CRS	定时器比较映射值	$013C_H$	U, PV	U, PV	页 19-112
CC40TIMER	定时器当前值	0170_H	U, PV	U, PV	页 19-113
CC40C0V	捕获寄存器 0 值	0174_H	U, PV	BE	页 19-114
CC40C1V	捕获寄存器 1 值	0178_H	U, PV	BE	页 19-115
CC40C2V	捕获寄存器 2 值	$017C_H$	U, PV	BE	页 19-116
CC40C3V	捕获寄存器 3 值	0180_H	U, PV	BE	页 19-117
CC40INTS	中断状态	$01A0_H$	U, PV	BE	页 19-118
CC40INTE	中断使能	$01A4_H$	U, PV	U, PV	页 19-119
CC40SRS	中断配置	$01A8_H$	U, PV	U, PV	页 19-121
CC40SWS	中断状态置位	$01AC_H$	U, PV	U, PV	页 19-122

表 19-12 CCU4 寄存器一览表 (续表)

简称	描述	偏移地址[1]	访问方式		描述见
			读	写	
CC40SWR	中断状态清除	$01B0_H$	U, PV	U, PV	页 19-124
CC40ECRD0	扩展回读 0	$01B8_H$	U, PV	BE	页 19-125
CC40ECRD1	扩展回读 1	$01BC_H$	U, PV	BE	页 19-126
CC41 寄存器					
CC41INS	输入选择器单元配置	0200_H	U, PV	U, PV	页 19-95
CC41CMC	连接矩阵配置	0204_H	U, PV	U, PV	页 19-97
CC41TST	定时器运行状态	0208_H	U, PV	BE	页 19-99
CC41TCSET	定时器运行置位	$020C_H$	U, PV	U,PV	页 19-100
CC41TCCLR	定时器运行清除	0210_H	U, PV	U, PV	页 19-101
CC41TC	通用定时器配置	0214_H	U, PV	U, PV	页 19-102
CC41PSL	输出被动电平配置	0218_H	U, PV	U, PV	页 19-106
CC41DIT	抖动配置	$021C_H$	U, PV	BE	页 19-107
CC41DITS	抖动映射寄存器	0220_H	U, PV	U, PV	页 19-108
CC41PSC	预分频器配置	0224_H	U, PV	U, PV	页 19-108
CC41FPC	预分频器比较值	0228_H	U, PV	U, PV	页 19-109
CC41FPCS	预分频器映射比较值	$022C_H$	U, PV	U, PV	页 19-110
CC41PR	定时器周期值	0230_H	U, PV	BE	页 19-110
CC41PRS	定时器周期映射值	0234_H	U, PV	U, PV	页 19-111
CC41CR	定时器比较值	0238_H	U, PV	BE	页 19-112
CC41CRS	定时器比较映射值	$023C_H$	U, PV	U, PV	页 19-112
CC41TIMER	定时器当前值	0270_H	U, PV	U, PV	页 19-113
CC41C0V	捕获寄存器 0 值	0274_H	U, PV	BE	页 19-114
CC41C1V	捕获寄存器 1 值	0278_H	U, PV	BE	页 19-115
CC41C2V	捕获寄存器 2 值	$027C_H$	U, PV	BE	页 19-116
CC41C3V	捕获寄存器 3 值	0280_H	U, PV	BE	页 19-117
CC41INTS	中断状态	$02A0_H$	U, PV	BE	页 19-118
CC41INTE	中断使能	$02A4_H$	U, PV	U, PV	页 19-119
CC41SRS	中断配置	$02A8_H$	U, PV	U, PV	页 19-121
CC41SWS	中断状态置位	$02AC_H$	U, PV	U, PV	页 19-122

表 19-12 CCU4 寄存器一览表 (续表)

简称	描述	偏移地址[1)]	访问方式		描述见
			读	写	
CC41SWR	中断状态清除	$02B0_H$	U, PV	U, PV	页 19-124
CC41ECRD0	扩展回读 0	$02B8_H$	U, PV	BE	页 19-125
CC41ECRD1	扩展回读 1	$02BC_H$	U, PV	BE	页 19-126
CC42 寄存器					
CC42INS	输入选择器单元配置	0300_H	U, PV	U, PV	页 19-95
CC42CMC	连接矩阵配置	0304_H	U, PV	U, PV	页 19-97
CC42TST	定时器运行状态	0308_H	U, PV	BE	页 19-99
CC42TCSET	定时器运行置位	$030C_H$	U, PV	U,PV	页 19-100
CC42TCCLR	定时器运行清除	0310_H	U, PV	U, PV	页 19-101
CC42TC	通用定时器配置	0314_H	U, PV	U, PV	页 19-102
CC42PSL	输出被动电平配置	0318_H	U, PV	U, PV	页 19-106
CC42DIT	抖动配置	$031C_H$	U, PV	BE	页 19-107
CC42DITS	抖动映射寄存器	0320_H	U, PV	U, PV	页 19-108
CC42PSC	预分频器配置	0324_H	U, PV	U, PV	页 19-108
CC42FPC	预分频器比较值	0328_H	U, PV	U, PV	页 19-109
CC42FPCS	预分频器映射比较值	$032C_H$	U, PV	U, PV	页 19-110
CC42PR	定时器周期值	0330_H	U, PV	BE	页 19-110
CC42PRS	定时器周期映射值	0334_H	U, PV	U, PV	页 19-111
CC42CR	定时器比较值	0338_H	U, PV	BE	页 19-112
CC42CRS	定时器比较映射值	$033C_H$	U, PV	U, PV	页 19-112
CC42TIMER	定时器当前值	0370_H	U, PV	U, PV	页 19-113
CC42C0V	捕获寄存器 0 值	0374_H	U, PV	BE	页 19-114
CC42C1V	捕获寄存器 1 值	0378_H	U, PV	BE	页 19-115
CC42C2V	捕获寄存器 2 值	$037C_H$	U, PV	BE	页 19-116
CC42C3V	捕获寄存器 3 值	0380_H	U, PV	BE	页 19-117
CC42INTS	中断状态	$03A0_H$	U, PV	BE	页 19-118
CC42INTE	中断使能	$03A4_H$	U, PV	U, PV	页 19-119
CC42SRS	中断配置	$03A8_H$	U, PV	U, PV	页 19-121
CC42SWS	中断状态置位	$03AC_H$	U, PV	U, PV	页 19-122

表 19-12 CCU4 寄存器一览表 (续表)

简称	描述	偏移地址[1]	访问方式		描述见
			读	写	
CC42SWR	中断状态清除	$03B0_H$	U, PV	U, PV	页 19-124
CC42ECRD0	扩展回读 0	$03B8_H$	U, PV	BE	页 19-125
CC42ECRD1	扩展回读 1	$03BC_H$	U, PV	BE	页 19-126
CC43 寄存器					
CC43INS	输入选择器单元配置	0400_H	U, PV	U, PV	页 19-95
CC43CMC	连接矩阵配置	0404_H	U, PV	U, PV	页 19-97
CC43TST	定时器运行状态	0408_H	U, PV	BE	页 19-99
CC43TCSET	定时器运行置位	$040C_H$	U, PV	U,PV	页 19-100
CC43TCCLR	定时器运行清除	0410_H	U, PV	U, PV	页 19-101
CC43TC	通用定时器配置	0414_H	U, PV	U, PV	页 19-102
CC43PSL	输出被动电平配置	0418_H	U, PV	U, PV	页 19-106
CC43DIT	抖动配置	$041C_H$	U, PV	BE	页 19-107
CC43DITS	抖动映射寄存器	0420_H	U, PV	U, PV	页 19-108
CC43PSC	预分频器配置	0424_H	U, PV	U, PV	页 19-108
CC43FPC	预分频器比较值	0428_H	U, PV	U, PV	页 19-109
CC43FPCS	预分频器映射比较值	$042C_H$	U, PV	U, PV	页 19-110
CC43PR	定时器周期值	0430_H	U, PV	BE	页 19-110
CC43PRS	定时器周期映射值	0434_H	U, PV	U, PV	页 19-111
CC43CR	定时器比较值	0438_H	U, PV	BE	页 19-112
CC43CRS	定时器比较映射值	$043C_H$	U, PV	U, PV	页 19-112
CC43TIMER	定时器当前值	0470_H	U, PV	U, PV	页 19-113
CC43C0V	捕获寄存器 0 值	0474_H	U, PV	BE	页 19-114
CC43C1V	捕获寄存器 1 值	0478_H	U, PV	BE	页 19-115
CC43C2V	捕获寄存器 2 值	$047C_H$	U, PV	BE	页 19-116
CC43C3V	捕获寄存器 3 值	0480_H	U, PV	BE	页 19-117
CC43INTS	中断状态	$04A0_H$	U, PV	BE	页 19-118
CC43INTE	中断使能	$04A4_H$	U, PV	U, PV	页 19-119
CC43SRS	中断配置	$04A8_H$	U, PV	U, PV	页 19-121
CC43SWS	中断状态置位	$04AC_H$	U, PV	U, PV	页 19-122

表 19-12 CCU4 寄存器一览表 (续表)

简称	描述	偏移地址[1]	访问方式		描述见
			读	写	
CC43SWR	中断状态清除	$04B0_H$	U, PV	U, PV	页 19-124
CC43ECRD0	扩展回读 0	$04B8_H$	U, PV	BE	页 19-125
CC43ECRD1	扩展回读 1	$04BC_H$	U, PV	BE	页 19-126

1) 绝对寄存器地址计算如下：
模块基地址 + 偏移地址 (如本列所示)

19.7.1 全局寄存器

GCTRL

该寄存器包含影响 CCU4 内所有定时器片的全局配置位域。

GCTRL
全局控制寄存器 (0000_H) 复位值 : 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSDE		MSE3	MSE2	MSE1	MSE0	SUSCFG		0		PCIS		0	PRBC		
rw		rw	rw	rw	rw	rw		r		rw		r	rw		

域	位	类型	描述
PRBC	[2:0]	rw	**预分频器清除配置** 该域控制如何清除预分频器的运行位和内部寄存器。 000_B 只能由软件清除。 001_B 在 CC40 的运行位被清除时， **GSTAT**.PRB 和预分频器寄存器被清除。 010_B 在 CC41 的运行位被清除时， **GSTAT**.PRB 和预分频器寄存器被清除。 011_B 在 CC42 的运行位被清除时， **GSTAT**.PRB 和预分频器寄存器被清除。 100_B 在 CC43 的运行位被清除时， **GSTAT**.PRB 和预分频器寄存器被清除。

域	位	类型	描述
PCIS	[5:4]	rw	**预分频器输入时钟选择** 00_B 模块时钟 01_B CCU4x.ECLKA 10_B CCU4x.ECLKB 11_B CCU4x.ECLKC
SUSCFG	[9:8]	rw	**挂起模式配置** 该域控制所有 CAPCOM4 定时器片的挂起模式进入。 00_B 忽略挂起请求。模块不能进入挂起模式。 01_B 立即停止所有运行的定时器片。不采用安全停止。 10_B 立即停止模块，并将所有输出钳位到被动态。采用安全停止。 11_B 等待每个定时器片的翻转来停止并钳位定时器片的输出。采用安全停止。
MSE0	10	rw	**定时器片 0 多通道映射传送使能** 当该域被置位时，不仅可以通过软件还可以通过 CCU4x.MCSS 输入来请求定时器片 0 的映射传送。 0_B 只能通过软件来请求映射传送。 1_B 可以通过软件或 CCU4x.MCSS 输入来请求映射传送。
MSE1	11	rw	**定时器片 1 多通道映射传送使能** 当该域被置位时，不仅可以通过软件还可以通过 CCU4x.MCSS 输入来请求定时器片 1 的映射传送。 0_B 只能通过软件来请求映射传送。 1_B 可以通过软件或 CCU4x.MCSS 输入来请求映射传送。
MSE2	12	rw	**定时器片 2 多通道映射传送使能** 当该域被置位时，不仅可以通过软件还可以通过 CCU4x.MCSS 输入来请求定时器片 2 的映射传送。 0_B 只能通过软件来请求映射传送。 1_B 可以通过软件或 CCU4x.MCSS 输入来请求映射传送。

域	位	类型	描述
MSE3	13	rw	**定时器片 3 多通道映射传送使能** 当该域被置位时，不仅可以通过软件还可以通过 CCU4x.MCSS 输入来请求定时器片 3 的映射传送。 0_B 只能通过软件来请求映射传送。 1_B 可以通过软件或 CCU4x.MCSS 输入来请求映射传送。
MSDE	[15:14]	rw	**多通道映射传送请求配置** 该域配置通过 CCU4x.MCSS 输入请求的映射传送类型。为使该配置有效，需要将域 **CC4yTC**.MSEy 置 1。 00_B 只请求周期值和比较值的映射传送 01_B 请求比较值、周期值和预分频器比较值的映射传送 10_B 保留 11_B 请求比较值、周期值、预分频器值和抖动比较值的映射传送
0	3, [7:6], [31:16]	r	**保留** 读时总是返回 0。

GSTAT

该寄存器包含预分频器和每个定时器片的状态 (空闲模式或正在运行)。

GSTAT

全局状态寄存器 **(0004_H)** **复位值 : $0000000F_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							**PRB**	**0**				**S3I**	**S2I**	**S1I**	**S0I**
r							rh	r				r	r	r	r

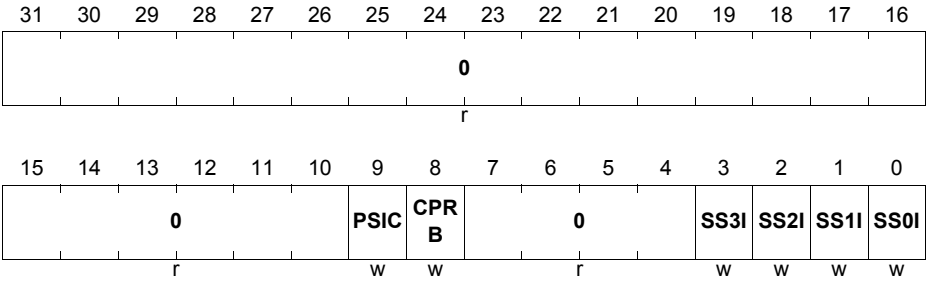
域	位	类型	描述
S0I	0	r	**CC40 空闲状态** 该位指示定时器片 CC40 是否处于空闲模式。在空闲模式，定时器片 CC40 的时钟被停止。 0_B 正在运行 1_B 空闲
S1I	1	r	**CC41 空闲状态** 该位指示定时器片 CC41 是否处于空闲模式。在空闲模式， CC41 的时钟被停止。 0_B 正在运行 1_B 空闲
S2I	2	r	**CC42 空闲状态** 该位指示定时器片 CC42 是否处于空闲模式。在空闲模式， CC42 的时钟被停止。 0_B 正在运行 1_B 空闲
S3I	3	r	**CC43 空闲状态** 该位指示定时器片 CC43 是否处于空闲模式。在空闲模式， CC43 的时钟被停止。 0_B 正在运行 1_B 空闲
PRB	8	rh	**预分频器运行位** 0_B 预分频器被停止 1_B 预分频器正在运行
0	[7:4], [31:9]	r	**保留** 读访问总是返回 0。

GIDLS

通过该寄存器可以将预分频器和特定定时器片设置为空闲模式。

GIDLS

全局空闲置位 (0008_H) 复位值 : 00000000_H



域	位	类型	描述
SS0I	0	w	**CC40 空闲模式置位** 向该位写入 1_B 会将 CC40 定时器片设置为空闲模式。当处于空闲模式时，该定时器片的时钟被停止。在进入空闲模式时，内部的定时器片寄存器不会被清除。读访问总是返回 0。
SS1I	1	w	**CC41 空闲模式置位** 向该位写入 1_B 会将 CC41 定时器片设置为空闲模式。当处于空闲模式时，该定时器片的时钟被停止。在进入空闲模式时，内部的定时器片寄存器不会被清除。读访问总是返回 0。
SS2I	2	w	**CC42 空闲模式置位** 向该位写入 1_B 会将 CC42 定时器片设置为空闲模式。当处于空闲模式时，该定时器片的时钟被停止。在进入空闲模式时，内部的定时器片寄存器不会被清除。读访问总是返回 0。
SS3I	3	w	**CC43 空闲模式置位** 向该位写入 1_B 会将 CC43 定时器片设置为空闲模式。当处于空闲模式时，该定时器片的时钟被停止。在进入空闲模式时，内部的定时器片寄存器不会被清除。读访问总是返回 0。

域	位	类型	描述
CPRB	8	w	**预分频器运行位清除** 向该位写入 1_B 会清除预分频器的运行位。预分频器内部的寄存器不会被清除。读访问总是返回 0。
PSIC	9	w	**预分频器清除** 向该位写入 1_B 会清除预分频器的计数器。它也会将 PSIV 加载到所有定时器片的 PVAL 域。 这会将所有定时器片的定时器时钟进行重新调整。 预分频器的运行位不会被清除。 读访问总是返回 0。
0	[7:4], [31:10]	r	**保留** 读访问总是返回 0。

GIDLC

通过该寄存器可以使预分频器和特定的定时器片退出空闲模式。

GIDLC

全局空闲模式清除 **($000C_H$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							SPRB	0				CS3I	CS2I	CS1I	CS0I
r							w	r				w	w	w	w

域	位	类型	描述
CS0I	0	w	**CC40 空闲模式清除** 向该位写入 1_B 使 CC40 退出空闲模式。读访问总是返回 0。
CS1I	1	w	**CC41 空闲模式清除** 向该位写入 1_B 使 CC41 退出空闲模式。读访问总是返回 0。
CS2I	2	w	**CC42 空闲模式清除** 向该位写入 1_B 使 CC42 退出空闲模式。读访问总是返回 0。

域	位	类型	描述
CS3I	3	w	**CC43 空闲模式清除** 向该位写入 1_B 使 CC43 退出空闲模式。读访问总是返回 0。
SPRB	8	w	**预分频器运行位置位** 向该位写入 1_B 可以将预分频器的运行位置位。读访问总是返回 0。
0	[7:4], [31:9]	r	**保留** 读访问总是返回 0。

GCSS

通过该寄存器可以为特定的定时器片请求一次映射传送，并将每个比较通道的状态位置 1。

GCSS

全局通道置位 **(0010_H)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												S3STS	S2STS	S1STS	S0STS
r												w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3PSE	S3DSE	S3SE	0	S2PSE	S2DSE	S2SE	0	S1PSE	S1DSE	S1SE	0	S0PSE	S0DSE	S0SE
r	w	w	w	r	w	w	w	r	w	w	w	r	w	w	w

域	位	类型	描述
S0SE	0	w	**定时器片 0 映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S0SS 域置 1，然后会启动对周期值、比较值和被动电平值的一次映射传送。 读访问总是返回 0。
S0DSE	1	w	**定时器片 0 抖动映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S0DSS 域置 1，然后会启动对抖动比较值的一次映射传送。 读访问总是返回 0。

域	位	类型	描述
S0PSE	2	w	**定时器片 0 预分频器映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S0PSS 域置 1，然后会启动对预分频器比较值的一次映射传送。 读访问总是返回 0。
S1SE	4	w	**定时器片 1 映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S1SS 域置 1，然后会启动对周期值、比较值和被动电平值的一次映射传送。 读访问总是返回 0。
S1DSE	5	w	**定时器片 1 抖动映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S1DSS 域置 1，然后会启动对抖动比较值的一次映射传送。 读访问总是返回 0。
S1PSE	6	w	**定时器片 1 预分频器映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S1PSS 域置 1，然后会启动对预分频器比较值的一次映射传送。 读访问总是返回 0。
S2SE	8	w	**定时器片 2 映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S2SS 域置 1，然后会启动对周期值、比较值和被动电平值的一次映射传送。 读访问总是返回 0。
S2DSE	9	w	**定时器片 2 抖动映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S2DSS 域置 1，然后会启动对抖动比较值的一次映射传送。 读访问总是返回 0。
S2PSE	10	w	**定时器片 2 预分频器映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S2PSS 域置 1，然后会启动对预分频器比较值的一次映射传送。 读出值总是为 0。
S3SE	12	w	**定时器片 3 映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S3SS 域置 1，然后会启动对周期值、比较值和被动电平值的一次映射传送。 读访问总是返回 0。
S3DSE	13	w	**定时器片 3 抖动映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S3DSS 域置 1，然后会启动对抖动比较值的一次映射传送。 读访问总是返回 0。

域	位	类型	描述
S3PSE	14	w	**定时器片 3 预分频器映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S3PSS 位域置 1，然后会启动对预分频器比较值的一次映射传送。 读访问总是返回 0。
S0STS	16	w	**定时器片 0 状态位置位** 向该位写入 1_B 会将定时器片 0 的状态位 (**GCST**.CC40ST) 设置为 1_B。 读访问总是返回 0。
S1STS	17	w	**定时器片 1 状态位置位** 向该位写入 1_B 会将定时器片 1 的状态位 (**GCST**.CC41ST) 设置为 1_B 读访问总是返回 0。
S2STS	18	w	**定时器片 2 状态位置位** 向该位写入 1_B 会将定时器片 2 的状态位 (**GCST**.CC42ST) 设置为 1_B. 读访问总是返回 0。
S3STS	19	w	**定时器片 3 状态位置位** 向该位写入 1_B 会将定时器片 3 的状态位 (**GCST**.CC43ST) 设置为 1_B. 读访问总是返回 0。
0	3, 7, 11, 15, [31:20]	r	**保留** 读访问总是返回 0。

GCSC

通过该寄存器，可以将特定定时器片的一次映射传送请求复位，并清除每个比较通道的状态位。

GCSC

全局通道清除 **(0014$_H$)** **复位值 : 00000000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												S3STC	S2STC	S1STC	S0STC
r												w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3PSC	S3DSC	S3SC	0	S2PSC	S2DSC	S2SC	0	S1PSC	S1DSC	S1SC	0	S0PSC	S0DSC	S0SC
r	w	w	w	r	w	w	w	r	w	w	w	r	w	w	w

域	位	类型	描述
S0SC	0	w	**定时器片 0 映射传送清除** 向该位写入 1_B 会清除 **GCST**.S0SS 域，从而取消对周期值、比较值和被动电平值的任何即将执行的映射传送。 读访问总是返回 0。
S0DSC	1	w	**定时器片 0 抖动映射传送清除** 向该位写入 1_B 会清除 **GCST**.S0DSS 域，从而取消对抖动比较值的任何即将执行的映射传送。 读访问总是返回 0。
S0PSC	2	w	**定时器片 0 预分频器映射传送清除** 向该位写入 1_B 会清除 **GCST**.S0PSS 域，从而取消对预分频器比较值的任何即将执行的映射传送。 读访问总是返回 0。
S1SC	4	w	**定时器片 1 映射传送清除** 向该位写入 1_B 会清除 **GCST**.S1SS 域，从而取消对周期值、比较值和被动电平值的任何即将执行的映射传送。 读访问总是返回 0。

域	位	类型	描述
S1DSC	5	w	**定时器片 1 抖动映射传送清除** 向该位写入 1_B 会清除 **GCST**.S1DSS 域，从而取消对抖动比较值的任何即将执行的映射传送。 读访问总是返回 0。
S1PSC	6	w	**定时器片 1 预分频器映射传送清除** 向该位写入 1_B 会清除 **GCST**.S1PSS 域，从而取消对预分频器比较值的任何即将执行的映射传送。 读访问总是返回 0。
S2SC	8	w	**定时器片 2 映射传送清除** 向该位写入 1_B 会清除 **GCST**.S2SS 域，从而取消对周期值、比较值和被动电平值的任何即将执行的映射传送。 读访问总是返回 0。
S2DSC	9	w	**定时器片 2 抖动映射传送清除** 向该位写入 1_B 会清除 **GCST**.S2DSS 域，从而取消对抖动比较值的任何即将执行的映射传送。 读访问总是返回 0。
S2PSC	10	w	**定时器片 2 预分频器映射传送清除** 向该位写入 1_B 会清除 **GCST**.S2PSS 域，从而取消对预分频器比较值的任何即将执行的映射传送。 读访问总是返回 0。
S3SC	12	w	**定时器片 3 映射传送清除** 向该位写入 1_B 会清除 **GCST**.S3SS 域，从而取消对周期值、比较值和被动电平值的任何即将执行的映射传送。 读访问总是返回 0。
S3DSC	13	w	**定时器片 3 抖动映射传送清除** 向该位写入 1_B 会清除 **GCST**.S3DSS 域，从而取消对抖动比较值的任何即将执行的映射传送。 读访问总是返回 0。
S3PSC	14	w	**定时器片 3 预分频器映射传送清除** 向该位写入 1_B 会清除 **GCST**.S3PSS 域，从而取消对预分频器比较值的任何即将执行的映射传送。 读访问总是返回 0。
S0STC	16	w	**定时器片 0 状态位清除** 向该位写入 1_B 将定时器片 0 的状态位 (**GCST**.CC40ST) 清除为 0_B。 读访问总是返回 0。

域	位	类型	描述
S1STC	17	w	**定时器片 1 状态位清除** 向该位写入 1_B 将定时器片 1 的状态位 (**GCST**.CC41ST) 清除为 0_B。 读访问总是返回 0。
S2STC	18	w	**定时器片 2 状态位清除** 向该位写入 1_B 将定时器片 2 的状态位 (**GCST**.CC42ST) 清除为 0_B。 读访问总是返回 0。
S3STC	19	w	**定时器片 3 状态位清除** 向该位写入 1_B 将定时器片 3 的状态位 (**GCST**.CC43ST) 清除为 0_B。 读访问总是返回 0。
0	3, 7, 11, 15, [31:20]	r	**保留** 读访问总是返回 0。

GCST

该寄存器保持映射传送请求信息和每个定时器片的状态位信息。

GCST

全局通道状态 **(0018_H)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0												CC43ST	CC42ST	CC41ST	CC40ST
r												rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3PSS	S3DSS	S3SS	0	S2PSS	S2DSS	S2SS	0	S1PSS	S1DSS	S1SS	0	S0PSS	S0DSS	S0SS
r	rh	rh	rh	r	rh	rh	rh	r	rh	rh	rh	r	rh	rh	rh

域	位	类型	描述
S0SS	0	rh	**定时器片 0 映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。

域	位	类型	描述
S0DSS	1	rh	**定时器片 0 抖动映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S0PSS	2	rh	**定时器片 0 预分频器映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S1SS	4	rh	**定时器片 1 映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S1DSS	5	rh	**定时器片 1 抖动映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S1PSS	6	rh	**定时器片 1 预分频器映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S2SS	8	rh	**定时器片 2 映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S2DSS	9	rh	**定时器片 2 抖动映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S2PSS	10	rh	**定时器片 2 预分频器映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S3SS	12	rh	**定时器片 3 映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。

域	位	类型	描述
S3DSS	13	rh	**定时器片 3 抖动映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S3PSS	14	rh	**定时器片 3 预分频器映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
CC40ST	16	rh	**定时器片 0 状态位**
CC41ST	17	rh	**定时器片 1 状态位**
CC42ST	18	rh	**定时器片 2 状态位**
CC43ST	19	rh	**定时器片 3 状态位**
0	3, 7, 11, 15, [31:20]	r	**保留** 读访问总是返回 0。

MIDR

该寄存器包含模块标识号。

MIDR

模块标识号 **(0080_H)** **复位值 : $00A6C0XX_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODN															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODT								MODR							
r								r							

域	位	类型	描述
MODR	[7:0]	r	**模块版本** 该位域指示模块实现 (取决于设计步骤) 的版本号。 给定值 00_H 是实际版本号的占位符。
MODT	[15:8]	r	**模块类型**

域	位	类型	描述
MODN	[31:16]	r	**模块号**

19.7.2 定时器片 (CC4y) 寄存器

CC4yINS

该寄存器包含输入选择器的配置信息。

CC4yINS (y = 0 - 3)

输入选择器配置 **($0100_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**LPF2M**		**LPF1M**		**LPF0M**		**EV2LM**	**EV1LM**	**EV0LM**	**EV2EM**		**EV1EM**		**EV0EM**	
r	rw		rw		rw		rw	rw	rw	rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**EV2IS**				**EV1IS**				**EV0IS**			
r				rw				rw				rw			

域	位	类型	描述
EV0IS	[3:0]	rw	**事件 0 信号选择** 该位域选择哪个引脚用于事件 0。 0000_B CCU4x.INyA 0001_B CCU4x.INyB 0010_B CCU4x.INyC 0011_B CCU4x.INyD 0100_B CCU4x.INyE 0101_B CCU4x.INyF 0110_B CCU4x.INyG 0111_B CCU4x.INyH 1000_B CCU4x.INyI 1001_B CCU4x.INyJ 1010_B CCU4x.INyK 1011_B CCU4x.INyL 1100_B CCU4x.INyM 1101_B CCU4x.INyN 1110_B CCU4x.INyO 1111_B CCU4x.INyP

域	位	类型	描述
EV1IS	[7:4]	rw	**事件 1 信号选择** 描述同 EV0IS
EV2IS	[11:8]	rw	**事件 2 信号选择** 描述同 EV0IS
EV0EM	[17:16]	rw	**事件 0 边沿选择** 00_B 无作用 01_B 信号在上升沿有效 10_B 信号在下降沿有效 11_B 信号在两个边沿都有效
EV1EM	[19:18]	rw	**事件 1 边沿选择** 描述同 EV0EM
EV2EM	[21:20]	rw	**事件 2 边沿选择** 描述同 EV0EM
EV0LM	22	rw	**事件 0 电平选择** 0_B 高电平有效 1_B 低电平有效
EV1LM	23	rw	**事件 1 电平选择** 描述同 EV0LM
EV2LM	24	rw	**事件 2 电平选择** 描述同 EV0LM
LPF0M	[26:25]	rw	**事件 0 低通滤波器配置** 该域设置事件 0 低通滤波器连续计数的数目。输入信号值需要在这些计数 (f_{CCU4}) 周期内保持稳定，以便一个电平 / 跳变能被接受。 00_B 低通滤波器被禁止 01_B 3 个 f_{CCU4} 时钟周期 10_B 5 个 f_{CCU4} 时钟周期 11_B 7 个 f_{CCU4} 时钟周期
LPF1M	[28:27]	rw	**事件 1 低通滤波器配置** 描述同 LPF0M
LPF2M	[30:29]	rw	**事件 2 低通滤波器配置** 描述同 LPF0M
0	[15:12], 31	r	**保留** 读访问总是返回 0。

CC4yCMC

该寄存器包含连接矩阵的配置。

CC4yCMC (y = 0 - 3)

连接矩阵控制 **(0104_H + 0100_H * y)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											TCE	MOS		TS	OFS
r											rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTS		LDS		UDS		GATES		CAP1S		CAP0S		ENDS		STRTS	
rw		rw		rw		rw		rw		rw		rw		rw	

域	位	类型	描述
STRTS	[1:0]	rw	**外部启动功能选择器** 选择要与外部启动功能连接的事件。 00_B 禁用外部启动功能 01_B 外部启动功能由事件 0 触发 10_B 外部启动功能由事件 1 触发 11_B 外部启动功能由事件 2 触发
ENDS	[3:2]	rw	**外部停止功能选择器** 选择要与外部停止功能连接的事件。 00_B 禁用外部停止功能 01_B 外部停止功能由事件 0 触发 10_B 外部停止功能由事件 1 触发 11_B 外部停止功能由事件 2 触发
CAP0S	[5:4]	rw	**外部捕获 0 功能选择器** 选择要与捕获寄存器 1 和捕获寄存器 0 的外部捕获功能连接的事件。 00_B 禁用外部捕获 0 功能 01_B 外部捕获 0 功能由事件 0 触发 10_B 外部捕获 0 功能由事件 1 触发 11_B 外部捕获 0 功能由事件 2 触发

域	位	类型	描述
CAP1S	[7:6]	rw	**外部捕获 1 功能选择器** 选择要与捕获寄存器 3 和捕获寄存器 2 的外部捕获功能连接的事件。 00_B 禁用外部捕获 1 功能 01_B 外部捕获 1 功能由事件 0 触发 10_B 外部捕获 1 功能由事件 1 触发 11_B 外部捕获 1 功能由事件 2 触发
GATES	[9:8]	rw	**外部门控功能选择器** 选择要与计数器门控功能连接的事件。该功能用于控制定时器的增 1/ 减 1 计数过程。 00_B 禁用外部门控功能 01_B 外部门控功能由事件 0 触发 10_B 外部门控功能由事件 1 触发 11_B 外部门控功能由事件 2 触发
UDS	[11:10]	rw	**外部向上 / 向下计数功能选择器** 选择要与向上 / 向下计数方向控制连接的事件。 00_B 禁用外部向上 / 向下计数功能 01_B 外部向上 / 向下计数功能由事件 0 触发 10_B 外部向上 / 向下计数功能由事件 1 触发 11_B 外部向上 / 向下计数功能由事件 2 触发
LDS	[13:12]	rw	**外部定时器加载功能选择器** 选择要与定时器加载功能连接的事件。 00_B - 禁用外部加载功能 01_B - 外部加载功能由事件 0 触发 10_B - 外部加载功能由事件 1 触发 11_B - 外部加载功能由事件 2 触发
CNTS	[15:14]	rw	**外部计数选择器** 选择要与计数功能连接的事件。每当检测到事件的一个特定跳变时，计数器将增 1/ 减 1。 00_B 禁用外部计数功能 01_B 外部计数功能由事件 0 触发 10_B 外部计数功能由事件 1 触发 11_B 外部计数功能由事件 2 触发
OFS	16	rw	**覆盖功能选择器** 该位使能状态位覆盖功能。 0_B 禁用覆盖功能 1_B 状态位触发信号覆盖与事件 1 相连，状态位值覆盖与事件 2 相连。

域	位	类型	描述
TS	17	rw	**陷阱功能选择器** 该位域使能陷阱功能。 0_B 禁止陷阱功能 1_B 陷阱功能与事件 2 相连
MOS	[19:18]	rw	**外部调制功能选择器** 选择要与外部调制功能连接的事件。 00_B - 禁用调制功能 01_B - 调制功能由事件 0 触发 10_B - 调制功能由事件 1 触发 11_B - 调制功能由事件 2 触发
TCE	20	rw	**定时器级联使能** 该位使能与前一个定时器片的级联。 0_B 禁止定时器级联 1_B 使能定时器级联 *注： 在 CC40 中不存在该域。这是一个只读的保留位。读访问总是返回 0。*
0	[31:21]	r	**保留** 读访问总是返回 0

CC4yTCST

该寄存器保持定时器的状态 (运行 / 停止) 和计数方向 (向上 / 向下) 信息。

CC4yTCST (y = 0 - 3)

片定时器的状态 **($0108_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														CDIR	TRB
r														rh	rh

域	位	类型	描述
TRB	0	rh	**定时器运行位** 该位指示定时器是否正在运行 0_B 定时器已停止 1_B 定时器正在运行
CDIR	1	rh	**定时器计数方向** 该位指示定时器是向上计数还是向下计数 0_B 定时器向上计数 1_B 定时器向下计数
0	[31:2]	r	**保留** 读访问总是返回 0

CC4yTCSET

可以通过该寄存器启动定时器。

CC4yTCSET (y = 0 - 3)

片定时器运行置位 **($010C_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**TRBS**
r															w

域	位	类型	描述
TRBS	0	w	**定时器运行位置位** 向该域写入 1_B 将定时器的运行位置 1。 读访问总是返回 0。
0	[31:1]	r	**保留** 读访问总是返回 0

CC4yTCCLR

通过该寄存器可以停止并清除定时器，并清除抖动计数器。

CC4yTCCLR (y = 0 - 3)

片定时器清除 **($0110_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													DITC	TCC	TRBC
r													w	w	w

域	位	类型	描述
TRBC	0	w	**定时器运行位清除** 向该域写入 1_B 将清除定时器的运行位。定时器不被清零。 读访问总是返回 0。
TCC	1	w	**定时器清零** 向该位写入 1_B 会将定时器清除为 0000_H。 读访问总是返回 0。
DITC	2	w	**抖动计数器清零** 向该位写入 1_B 会将抖动计数器清除为 0_H。 读访问总是返回 0。
0	[31:3]	r	**保留** 读访问总是返回 0。

CC4yTC

该寄存器保持定时器操作的几种可能配置。

CC4yTC (y = 0 - 3)
片定时器控制 **($0114_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						MCME	EMT	EMS	TRPSW	TRPSE	0			TRAPE	FPE
r						rw	rw	rw	rw	rw	r			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIM	DITHE		CCS	SCE	STRM	ENDM		0	CAPC		ECM	CMOD	CLST	TSSM	TCM
rw	rw		rw	rw	rw	rw		r	rw		rw	rh	rw	rw	rw

域	位	类型	描述
TCM	0	rw	**定时器计数模式** 该域控制定时器的实际计数方式。 0_B 边沿对齐模式 1_B 中心对齐模式 *注： 当使用一个外部信号来控制计数方向时，计数方式总是为边沿对齐。*
TSSM	1	rw	**定时器单次模式** 该域控制单次模式。该模式可用于边沿对齐模式和中心对齐模式。 0_B 禁止单次模式 1_B 使能单次模式
CLST	2	rw	**清除时执行映射传送** 将该位设置为 1_B 使得在执行定时器清除操作时进行一次映射传送。 注意，仍需通过软件将 **GCST** 寄存器中的映射传送使能位设置为 1_B。
CMOD	3	rh	**捕获比较模式** 该域指示定时器片工作在哪个模式。默认值是比较模式。当一个外部信号被映射为捕获触发信号时，硬件自动将定时器片设置为捕获模式。 0_B 比较模式 1_B 捕获模式

域	位	类型	描述
ECM	4	rw	**扩展捕获模式** 该域控制特定定时器片的捕获模式。该域仅在 CMOD 位为 1_B 时有效。 0_B 标准捕获模式。只能通过单独访问寄存器来清除每个捕获寄存器的满标志 1_B 扩展捕获模式。不仅可以通过单独访问寄存器来清除每个寄存器的满标志，还可以通过访问 ECRD 寄存器来实现。当读取 ECRD 寄存器时，仅清除 ECRD.VPTR 指向的捕获寄存器满标志。
CAPC	[6:5]	rw	**捕获时的清除控制** 00_B 定时器不会在发生捕获事件时被清除 01_B 定时器由一个捕获到捕获寄存器 2 和 3 的捕获事件来清除 (当 SCE = 1_B 时，定时器总是在一次捕获事件中被清除) 10_B 定时器由一个捕获到捕获寄存器 0 和 1 的捕获事件来清除 (当 SCE = 1_B 时，定时器总是在一次捕获事件中被清除) 11_B 定时器总是在一次捕获事件中被清除。
ENDM	[9:8]	rw	**扩展停止功能控制** 该域控制外部停止信号的扩展功能。 00_B 仅清除定时器运行位 (默认为停止) 01_B 仅清除定时器 (清空) 10_B 清除定时器及运行位 (清空 / 停止) 11_B 保留 *注：当使用一个外部向上/向下计数信号时，若计数器向上计数，则清空操作将定时器值设置为 0，若计数器向下计数，则清空操作将定时器值设置为周期值。*
STRM	10	rw	**扩展启动功能控制** 该域控制外部启动信号的扩展功能。 0_B 仅将运行位置 1 (默认为启动) 1_B 清除定时器并将运行位置 1(清空 / 启动) *注：当使用一个外部向上/向下计数信号时，若计数器向上计数，则清空操作将定时器值设置为 0，若计数器向下计数，则清空操作将定时器值设置为周期值。*

域	位	类型	描述
SCE	11	rw	**相同捕获事件使能** 0_B CCycapt0 控制捕获到 **CC4yC0V/CC4yC1V** 寄存器， CCycapt1 控制捕获到 **CC4yC3V/CC4yC2V** 寄存器 1_B 捕获到 **CC4yC0V/CC4yC1V** 寄存器和 **CC4yC3V/CC4yC2V** 寄存器都由 CCycapt1 控制
CCS	12	rw	**连续捕获使能** 0_B 使用如 **19.2.7.6 节**所描述的与满标志相关连的规则来捕获到特定的捕获寄存器。 1_B 无论满标志的状态如何，总是执行到捕获寄存器的捕获 (即使寄存器还没有被回读)。
DITHE	[14:13]	rw	**抖动使能** 该域控制定时器片的抖动模式。详见 **19.2.10 节**。 00_B 禁止抖动 01_B 将抖动应用到周期值 10_B 将抖动应用到比较值 11_B 将抖动应用到周期值和比较值
DIM	15	rw	**抖动输入选择器** 该域选择将抖动控制信号连接到特定定时器片的抖动逻辑还是连接到定时器片 0 的抖动逻辑。注意，即使该域被设置为 1_B，仍需对 DITHE 域编程。 0_B 定时器片使用自身的抖动逻辑单元 1_B 定时器片被连接到定时器片 0 的抖动逻辑单元。
FPE	16	rw	**浮动预分频器使能** 将该位设置为 1_B 使能浮动预分频器模式。 0_B 禁止浮动预分频器模式 1_B 使能浮动预分频器模式
TRAPE	17	rw	**陷阱使能** 将该位设置 1_B 使能输出引脚的陷阱动作。在将一个外部信号映射到陷阱功能后，用户必须将该域设置为 1_B，以激活输出引脚的陷阱作用。 向该位写入 0_B 会禁止陷阱功能的作用，不论输入信号的状态如何。 0_B 陷阱功能不影响输出 1_B 陷阱功能影响输出

域	位	类型	描述
TRPSE	21	rw	**陷阱同步使能** 向该位写入 1_B 会使陷阱状态退出与 PWM 信号同步。 0_B 退出陷阱状态不与 PWM 信号同步 1_B 退出陷阱状态与 PWM 信号同步
TRPSW	22	rw	**陷阱状态清除控制** 0_B 当陷阱条件不存在时，定时器片自动退出陷阱状态。 1_B 退出陷阱状态只能通过软件请求完成。
EMS	23	rw	**外部调制同步** 将该位设置为 1_B 使能外部调制功能与 PWM 信号的同步。 0_B 外部调制功能不与 PWM 信号同步 1_B 外部调制功能与 PWM 信号同步
EMT	24	rw	**外部调制类型** 该域选择外部调制事件是用于清除 CC4yST 位还是用于门控输出。 0_B 外部调制事件用于清除 CC4yST 位。 1_B 外部调制事件用于门控输出。
MCME	25	rw	**多通道模式使能** 0_B 禁止多通道模式 1_B 使能多通道模式
0	7, [20:18], [31:26]	r	**保留** 读访问总是返回 0。

CC4yPSL

该寄存器保持对输出被动电平控制的配置。

CC4yPSL (y = 0 - 3)

被动电平配置 **($0118_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**PSL**
r															rw

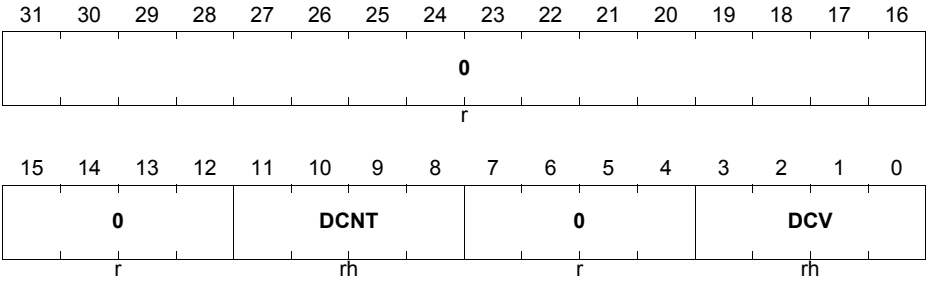
域	位	类型	描述
PSL	0	rw	**输出被动电平** 该域控制输出引脚的被动电平。 0_B 被动电平为低电平 1_B 被动电平为高电平 写操作总是寻址映射寄存器，而读操作总是返回当前使用值。
0	[31:1]	r	**保留** 读访问总是返回 0。

CC4yDIT

该寄存器保存着当前抖动比较值和抖动计数器值。

CC4yDIT (y = 0 - 3)

抖动配置 **($011C_H$ + 0100_H * y)** **复位值 : 00000000_H**

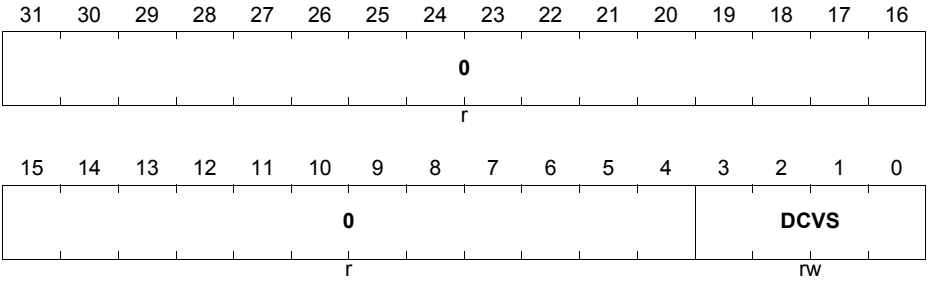


域	位	类型	描述
DCV	[3:0]	rh	**抖动比较值** 该域包含用于抖动比较的值。 当发生一次映射传送时，该值被更新为 **CC4yDITS**.DCVS。
DCNT	[11:8]	rh	**抖动计数器的当前值**
0	[7:4], [31:12]	r	**保留** 读访问总是返回 0。

CC4yDITS

该寄存器包含下一次发生映射传送时将要加载到 **CC4yDIT**.DCV 的值。

CC4yDITS (y = 0 - 3)
抖动映射寄存器 **(0120_H + 0100_H * y)** **复位值 : 00000000_H**

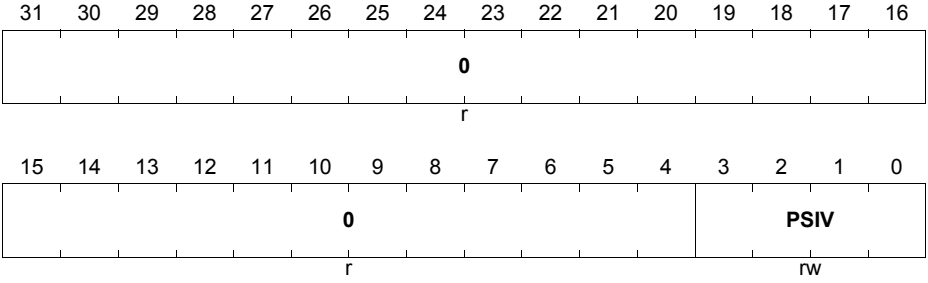


域	位	类型	描述
DCVS	[3:0]	rw	**抖动映射比较值** 该域包含在下一次映射传送过程中将要设置到抖动比较值 **CC4yDIT**.DCV 中的值。
0	[31:4]	r	**保留** 读访问总是返回 0。

CC4yPSC

该寄存器包含在重新启动时要加载到预分频器的值。

CC4yPSC (y = 0 - 3)
预分频器控制 **(0124_H + 0100_H * y)** **复位值 : 00000000_H**

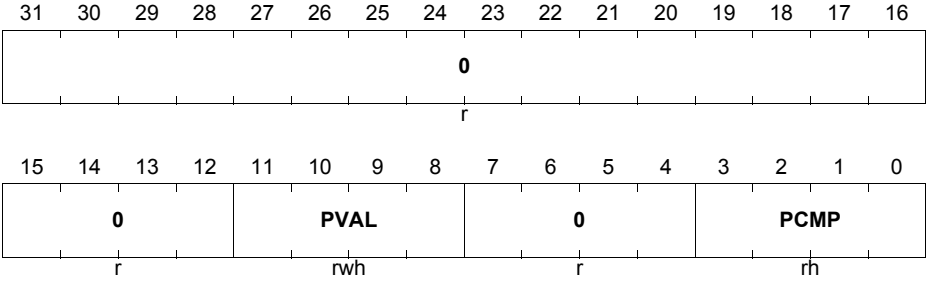


域	位	类型	描述
PSIV	[3:0]	rw	**预分频器初始值** 该域包含启动时加载到预分频器的值。 在采用浮动预分频器的情况下，当发生一次定时器比较匹配和预分频器比较匹配时，或者当一次捕获事件被触发时，加载该值。
0	[31:4]	r	**保留** 读访问总是返回 0。

CC4yFPC

该寄存器包含使用的浮动预分频器比较值和当前预分频器分频值。

CC4yFPC (y = 0 - 3)
浮动预分频器控制 **(0128_H + 0100_H * y)** **复位值 : 00000000_H**



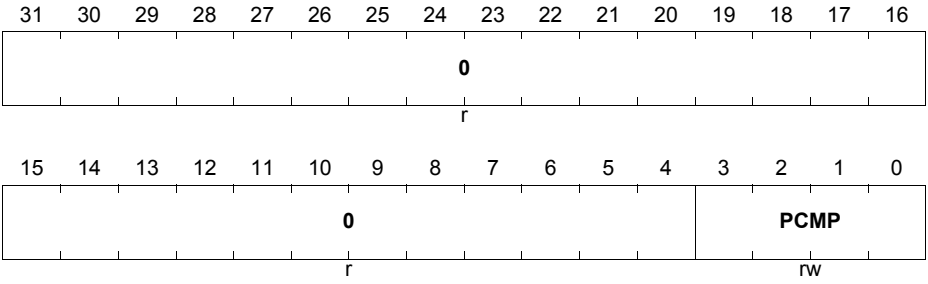
域	位	类型	描述
PCMP	[3:0]	rh	**浮动预分频器比较值** 该域包含在浮动预分频器模式所使用的比较值。比较操作由定时器比较匹配事件触发。详见 **19.2.11.2 节**。
PVAL	[11:8]	rwh	**当前预分频器值** 详见 **表 19-7**。只有在预分频器被停止时才能向该寄存器写入。在不使用浮动预分频器模式时，该值与 **CC4yPSC**.PSIV 值相等。
0	[7:4], [15:12], [31:16]	r	**保留** 读访问总是返回 0。

CC4yFPCS

该寄存器包含在下一次映射传送更新过程中将要被传送到 **CC4yFPC**.PCMP 域的值。

CC4yFPCS (y = 0 - 3)

浮动预分频器映射 **($012C_H + 0100_H * y$)** **复位值 : 00000000_H**



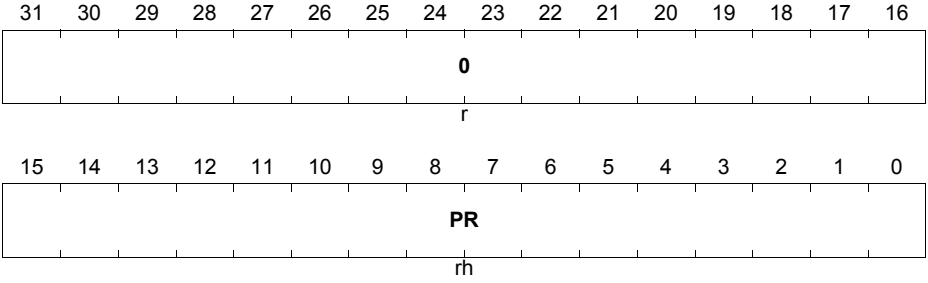
域	位	类型	描述
PCMP	[3:0]	rw	**浮动预分频器映射比较值** 该域包含在下一次映射传送过程中将要设置到 **CC4yFPC**.PCMP 域的值。详见**表 19-7**。
0	[31:4]	r	**保留** 读访问总是返回 0。

CC4yPR

该寄存器包含定时器周期的当前值。

CC4yPR (y = 0 - 3)

定时器周期值 **($0130_H + 0100_H * y$)** **复位值 : 00000000_H**

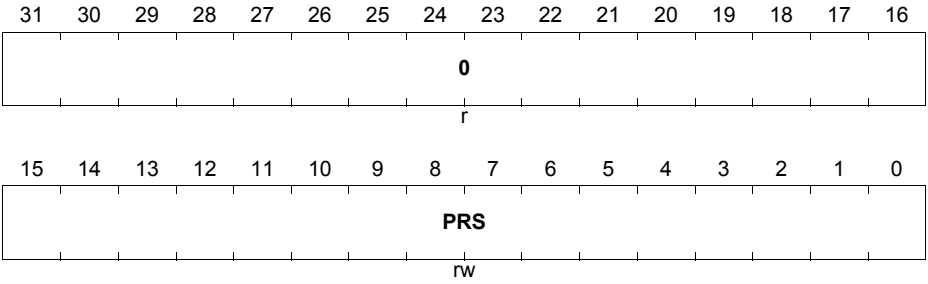


域	位	类型	描述
PR	[15:0]	rh	**周期寄存器** 包含定时器的周期值。 *注：在捕获模式，当选择一个外部信号来捕获定时器值到捕获寄存器2 和3 时，PR 不可写。读访问总是返回0。*
0	[31:16]	r	**保留** 读访问总是返回 0。

CC4yPRS

该寄存器包含在下一次映射传送发生时将要被传送到**CC4yPR**.PR位域的定时器周期值。

CC4yPRS (y = 0 - 3)
定时器映射周期值 **(0134_H + 0100_H * y)** **复位值 : 00000000_H**



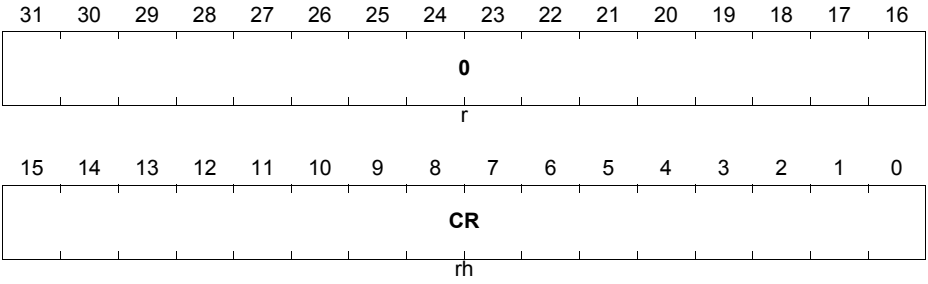
域	位	类型	描述
PRS	[15:0]	rw	**周期寄存器** 包含在下一次映射传送发生时要被传送到 **CC4yPR**.PR 域的定时器周期值。 *注：在捕获模式，当选择一个外部信号来捕获定时器值到捕获寄存器2 和3 时，PRS 不可写。读访问总是返回0。*
0	[31:16]	r	**保留** 读访问总是返回 0。

CC4yCR

该寄存器包含定时器比较值。

CC4yCR (y = 0 - 3)

定时器比较值 **(0138_H + 0100_H * y)** **复位值 : 00000000_H**



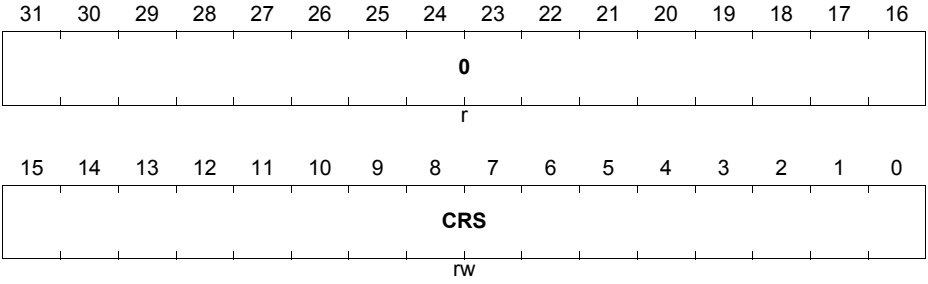
域	位	类型	描述
CR	[15:0]	rh	**比较寄存器** 包含定时器比较值。 *注： 在捕获模式，当选择一个外部信号来捕获定时器值到捕获寄存器0和1时，读访问总是返回0。*
0	[31:16]	r	**保留** 读访问总是返回 0。

CC4yCRS

该寄存器包含在下一次映射传送发生时将要被加载到 **CC4yCR**.CR 域的值。

CC4yCRS (y = 0 - 3)

定时器映射比较值 **($013C_H$ + 0100_H * y)** **复位值 : 00000000_H**



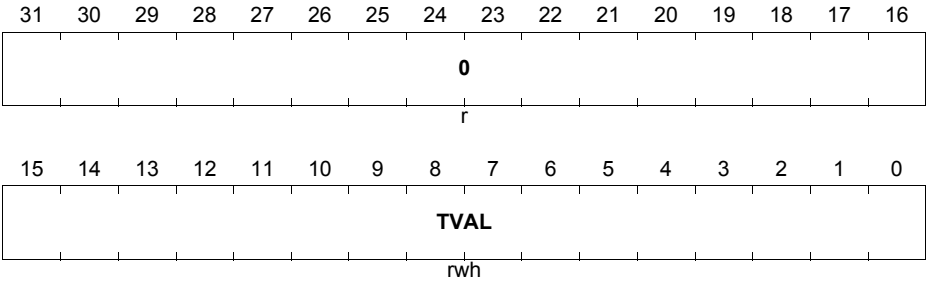
域	位	类型	描述
CRS	[15:0]	rw	**比较寄存器** 包含定时器比较值，该值在下一次映射传送发生时被传送到 **CC4yCR**.CR 域。 *注： 在捕获模式，当选择一个外部信号来捕获定时器值到捕获寄存器0和1时，读访问总是返回0。*
0	[31:16]	r	**保留** 读访问总是返回 0。

CC4yTIMER

该寄存器包含定时器的当前值。

CC4yTIMER (y = 0 - 3)

定时器值 **($0170_H + 0100_H * y$)** **复位值 : 00000000_H**



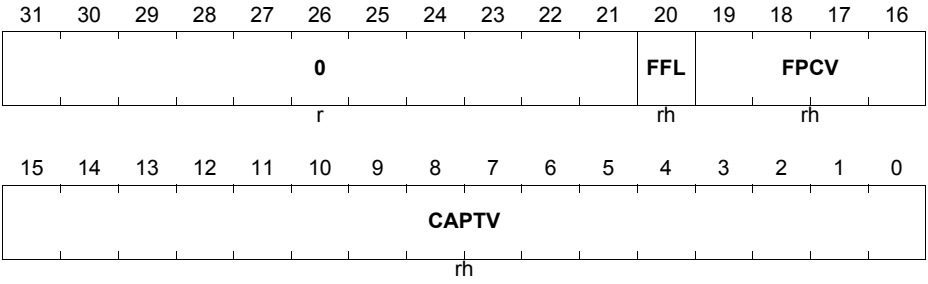
域	位	类型	描述
TVAL	[15:0]	rwh	**定时器值** 该域包含着定时器的当前值。 仅在定时器被停止时才可以进行写访问。
0	[31:16]	r	**保留** 读访问总是返回 0。

CC4yC0V

该寄存器包含与捕获 0 位域相关联的值。

CC4yC0V (y = 0 - 3)

捕获寄存器 0 **($0174_H + 0100_H * y$)** **复位值 : 00000000_H**



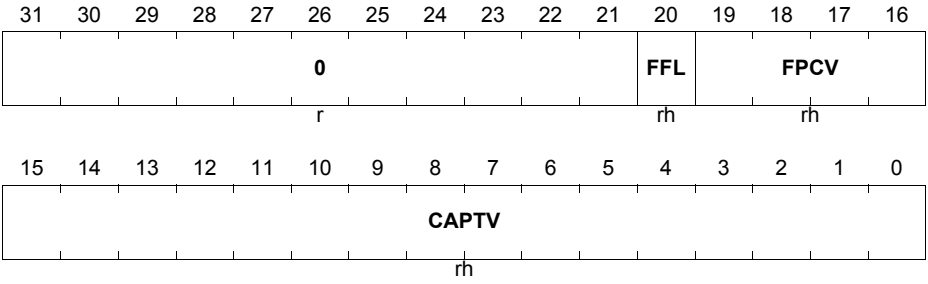
域	位	类型	描述
CAPTV	[15:0]	rh	**捕获值** 该域包含捕获寄存器 0 的值。详见 图 19-26。 在比较模式，读访问总是返回 0。
FPCV	[19:16]	rh	**预分频器值** 该域包含在捕获到捕获寄存器 0 的事件发生时的预分频器值。 在比较模式，读访问总是返回 0。
FFL	20	rh	**满标志** 该位指示在上一次读访问后是否有新值被捕获到捕获寄存器 0。详见图 19-26。在比较模式，读访问总是返回 0。 0_B 没有新值被捕获到特定的捕获寄存器 1_B 一个新值被捕获到特定的捕获寄存器
0	[31:21]	r	**保留** 读访问总是返回 0。

CC4yC1V

该寄存器包含与捕获 1 位域相关联的值。

CC4yC1V (y = 0 - 3)

捕获寄存器 1 **($0178_H + 0100_H * y$)** **复位值 : 00000000_H**



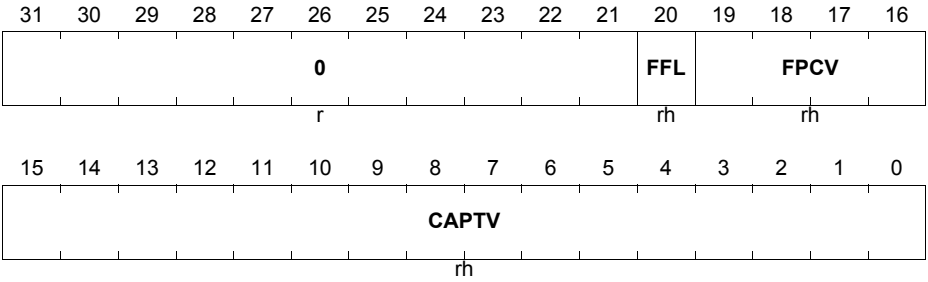
域	位	类型	描述
CAPTV	[15:0]	rh	**捕获值** 该域包含捕获寄存器 1 的值。详见图 19-26。 在比较模式，读访问总是返回 0。
FPCV	[19:16]	rh	**预分频器值** 该域包含在捕获到捕获寄存器 1 的事件发生时的预分频器值。 在比较模式，读访问总是返回 0。
FFL	20	rh	**满标记** 该位指示在上一次读访问之后是否有新值被捕获到捕获寄存器 1。详见图 19-26。在比较模式，读访问总是返回 0。 0_B 没有新值被捕获到特定的捕获寄存器 1_B 一个新值被捕获到特定的捕获寄存器
0	[31:21]	r	**保留** 读访问总是返回 0。

CC4yC2V

该寄存器包含与捕获 1 位域相关联的值。

CC4yC2V (y = 0 - 3)

捕获寄存器 2 **($017C_H + 0100_H * y$)** **复位值 : 00000000_H**



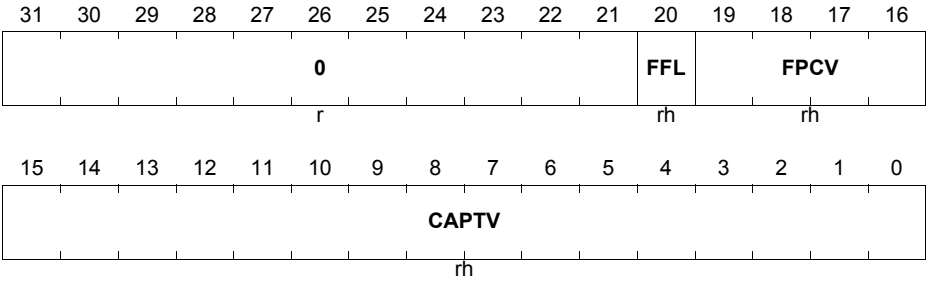
域	位	类型	描述
CAPTV	[15:0]	rh	**捕获值** 该位域包含捕获寄存器 2 的值。详见图 19-26。 在比较模式，读访问总是返回 0。
FPCV	[19:16]	rh	**预分频器值** 该位域包含在捕获到捕获寄存器 2 的事件发生时的预分频器值。在比较模式，读访问总是返回 0。
FFL	20	rh	**满标记** 该位指示在上一次读访问之后是否有新值被捕获到捕获寄存器 2。详见图 19-26。在比较模式，读访问总是返回 0。 0_B 没有新值被捕获到特定的捕获寄存器 1_B 一个新值被捕获到特定的捕获寄存器
0	[31:21]	r	**保留** 读访问总是返回 0

CC4yC3V

该寄存器包含与捕获 3 位域相关联的值。

CC4yC3V (y = 0 - 3)

捕获寄存器 3 **(0180_H + 0100_H * y)** **复位值 : 00000000_H**



域	位	类型	描述
CAPTV	[15:0]	rh	**捕获值** 该域包含捕获寄存器 3 的值。详见图 19-26。在比较模式，读访问总是返回 0。
FPCV	[19:16]	rh	**预分频器值** 该域包含在捕获到捕获寄存器 3 的事件发生时的预分频器值。 在比较模式，读访问总是返回 0。
FFL	20	rh	**满标记** 该位指示在上一次读访问之后是否有新值被捕获到捕获寄存器 3。详见图 19-26。在比较模式，读访问总是返回 0。 0_B 没有新值被捕获到特定的捕获寄存器 1_B 一个新值被捕获到特定的捕获寄存器
0	[31:21]	r	**保留** 读访问总是返回 0

CC4yINTS

该寄存器包含所有中断源的状态。

CC4yINTS (y = 0 - 3)

中断状态 **($01A0_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				TRPF	E2AS	E1AS	E0AS	0				CMDS	CMUS	OMDS	PMUS
r				rh	rh	rh	rh	r				rh	rh	rh	rh

域	位	类型	描述
PMUS	0	rh	**向上计数时的周期匹配** 0_B 未检测到向上计数时的周期匹配 1_B 已检测到向上计数时的周期匹配
OMDS	1	rh	**向下计数时的 1 匹配** 0_B 未检测到向下计数时的 1 匹配 1_B 已检测到向下计数时的 1 匹配
CMUS	2	rh	**向上计数时的比较匹配** 0_B 未检测到向上计数时的比较匹配 1_B 已检测到向上计数时的比较匹配
CMDS	3	rh	**向下计数时的比较匹配** 0_B 未检测到向下计数时的比较匹配 1_B 已检测到向下计数时的比较匹配
E0AS	8	rh	**事件 0 检测状态** 根据用户对 **CC4yINS**.EV0EM 的选择，可以在检测到上升沿、下降沿或任何一个边沿时将该位置 1。 0_B 未检测到事件 0 1_B 已检测到事件 0
E1AS	9	rh	**事件 1 检测状态** 根据用户对 **CC4yINS**.EV1EM 的选择，可以在检测到上升沿、下降沿或任何一个边沿时将该位置 1。 0_B 未检测到事件 1 1_B 已检测到事件 1

域	位	类型	描述
E2AS	10	rh	**事件 2 检测状态** 根据用户对 **CC4yINS**.EV1EM 的选择，可以在检测到上升沿、下降沿或任何一个边沿时将该位置 1。 0_B 未检测到事件 2 1_B 已检测到事件 2 *注： 如果该事件与陷阱功能相连，当定时器片退出陷阱状态时，该域被自动清除。*
TRPF	11	rh	**陷阱标志状态** 该位域包含陷阱标志的状态。
0	[7:4], [31:12]	r	**保留** 读访问总是返回 0。

CC4yINTE

通过该寄存器可以使能或禁止特定的中断源。

CC4yINTE (y = 0 - 3)

中断使能控制 **($01A4_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					E2AE	E1AE	E0AE	0				CMDE	CMUE	OME	PME
r					rw	rw	rw	r				rw	rw	rw	rw

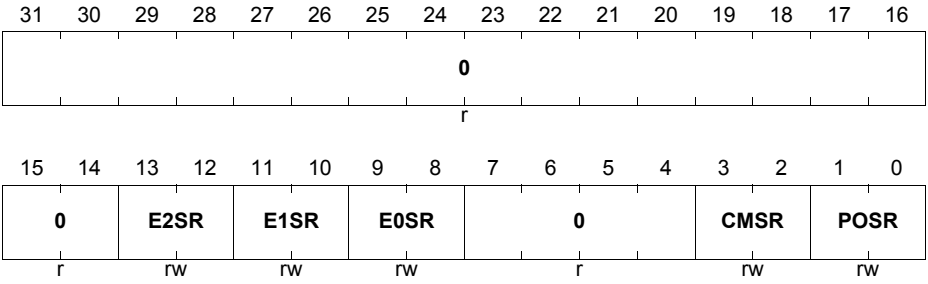
域	位	类型	描述
PME	0	rw	**向上计数周期匹配中断使能** 将该位设置为 1_B 时，在向上计数的情况下，每发生一次周期匹配就会产生一个中断脉冲。 0_B 禁止周期匹配中断 1_B 使能周期匹配中断

域	位	类型	描述
OME	1	rw	**向下计数时 1 匹配中断使能** 将该位设置为 1_B 时，在向下计数的情况下，每发生一次 1 匹配就会产生一个中断脉冲。 0_B 禁止 1 匹配中断 1_B 使能 1 匹配中断
CMUE	2	rw	**向上计数时比较匹配中断使能** 将该位设置为 1_B 时，在向上计数的情况下，每发生一次比较匹配就会产生一个中断脉冲。 0_B 禁止向上计数时的比较匹配中断 1_B 使能向上计数时的比较匹配中断
CMDE	3	rw	**向下计数时比较匹配中断使能** 将该位设置为 1_B 时，在向下计数的情况下，每发生一次比较匹配就会产生一个中断脉冲。 0_B 禁止向下计数时的比较匹配中断 1_B 使能向下计数时的比较匹配中断
E0AE	8	rw	**事件 0 中断使能** 将该位设置为 1_B 时，每当检测到事件 0 时就会产生一个中断脉冲。 0_B 禁止事件 0 检测中断 1_B 使能事件 0 检测中断
E1AE	9	rw	**事件 1 中断使能** 将该位设置为 1_B 时，每当检测到事件 1 时就会产生一个中断脉冲。 0_B 禁止事件 1 检测中断 1_B 使能事件 1 检测中断
E2AE	10	rw	**事件 2 中断使能** 将该位设置为 1_B 时，每当检测到事件 2 时就会产生一个中断脉冲。 0_B 禁止事件 2 检测中断 1_B 使能事件 2 检测中断
0	[7:4], [31:11]	r	**保留** 读访问总是返回 0

CC4ySRS

通过该寄存器可以选择将每个中断源转发到哪条服务请求线。

CC4ySRS (y = 0 - 3)
服务请求选择器 **($01A8_H + 0100_H * y$)** **复位值 : 00000000_H**



域	位	类型	描述
POSR	[1:0]	rw	**周期匹配 /1 匹配服务请求选择器** 该域选择将向上计数时周期匹配所产生的中断和向下计数时 1 匹配所产生的中断转发到哪条服务请求线。 00_B 转发到 CC4ySR0 01_B 转发到 CC4ySR1 10_B 转发到 CC4ySR2 11_B 转发到 CC4ySR3
CMSR	[3:2]	rw	**比较匹配服务请求选择器** 该域选择将向上计数时比较匹配所产生的中断和向下计数时比较匹配所产生的中断转发到哪条服务请求线。 00_B 转发到 CC4ySR0 01_B 转发到 CC4ySR1 10_B 转发到 CC4ySR2 11_B 转发到 CC4ySR3
E0SR	[9:8]	rw	**事件 0 服务请求选择器** 该域选择将事件 0 检测所产生的中断转发到哪条服务请求线。 00_B 转发到 CC4ySR0 01_B 转发到 CC4ySR1 10_B 转发到 CC4ySR2 11_B 转发到 CC4ySR3

域	位	类型	描述
E1SR	[11:10]	rw	**事件 1 服务请求选择器** 该域选择将事件 1 检测所产生的中断转发到哪条服务请求线。 00_B 转发到 CC4ySR0 01_B 转发到 CC4ySR1 10_B 转发到 CC4ySR2 11_B 转发到 CC4ySR3
E2SR	[13:12]	rw	**事件 2 服务请求选择器** 该域选择将事件 2 检测所产生的中断转发到哪条服务请求线。 00_B 转发到 CC4ySR0 01_B 转发到 CC4ySR1 10_B 转发到 CC4ySR2 11_B 转发到 CC4ySR3
0	[7:4], [31:14]	r	**保留** 读访问总是返回 0。

CC4ySWS

通过该寄存器可以用软件将一个特定的中断状态标志置 1。

CC4ySWS (y = 0 - 3)

中断状态置位 **($01AC_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				STRPF	SE2A	SE1A	SE0A	0				SCMD	SCMU	SOM	SPM
r				w	w	w	w	r				w	w	w	w

域	位	类型	描述
SPM	0	w	**向上计数时的周期匹配中断置位** 向该位域写入 1_B 会将 **CC4yINTS**.PMUS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。

域	位	类型	描述
SOM	1	w	**向下计数时的 1 匹配中断置位** 向该域写入 1_B 会将 **CC4yINTS**.OMDS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SCMU	2	w	**向上计数时的比较匹配中断置位** 向该域写入 1_B 会将 **CC4yINTS**.CMUS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SCMD	3	w	**向下计数时的比较匹配中断置位** 向该域写入 1_B 会将 **CC4yINTS**.CMDS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SE0A	8	w	**事件 0 检测中断置位** 向该域写入 1_B 会将 **CC4yINTS**.E0AS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SE1A	9	w	**事件 1 检测中断置位** 向该域写入 1_B 会将 **CC4yINTS**.E1AS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SE2A	10	w	**事件 2 检测中断置位** 向该域写入 1_B 会将 **CC4yINTS**.E2AS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
STRPF	11	w	**陷阱标志状态置位** 向该域写入 1_B 会将 **CC4yINTS**.TRPF 位置 1。 读访问总是返回 0。
0	[7:4], [31:12]	r	**保留** 读访问总是返回 0。

CC4ySWR

通过该寄存器可以用软件清除一个特定的中断状态标志。

CC4ySWR (y = 0 - 3)

中断状态清除 **($01B0_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				RTRPF	RE2A	RE1A	RE0A	0				RCMD	RCMU	ROM	RPM
r				w	w	w	w	r				w	w	w	w

域	位	类型	描述
RPM	0	w	**向上计数时的周期匹配中断清除** 向该位写入 1_B 会清除 **CC4yINTS**.PMUS 位。读访问总是返回 0。
ROM	1	w	**向下计数时的 1 匹配中断清除** 向该位写入 1_B 会清除 **CC4yINTS**.OMDS 位。读访问总是返回 0。
RCMU	2	w	**向上计数时的比较匹配中断清除** 向该位写入 1_B 会清除 **CC4yINTS**.CMUS 位。读访问总是返回 0。
RCMD	3	w	**向下计数时的比较匹配中断清除** 向该位写入 1_B 会清除 **CC4yINTS**.CMDS 位。读访问总是返回 0。
RE0A	8	w	**事件 0 检测中断清除** 向该位写入 1_B 会清除 **CC4yINTS**.E0AS 位。读访问总是返回 0。
RE1A	9	w	**事件 1 检测中断清除** 向该位写入 1_B 会清除 **CC4yINTS**.E1AS 位。读访问总是返回 0。
RE2A	10	w	**事件 2 检测中断清除** 向该位写入 1_B 会清除 **CC4yINTS**.E2AS 位。读访问总是返回 0。

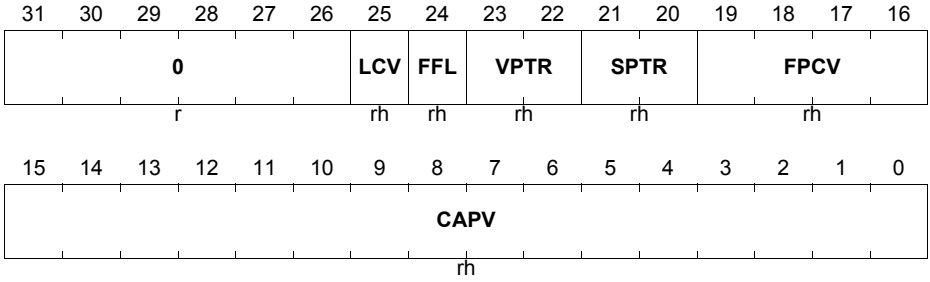
域	位	类型	描述
RTRPF	11	w	**陷阱标志状态清除** 向该位写入 1_B 会清除 **CC4yINTS**.TRPF 位。若 **CC4yTC**.TRPEN = 1_B，则清除操作无效，陷阱状态仍然有效。 读访问总是返回 0。
0	[7:4], [31:12]	r	**保留** 读访问总是返回 0。

CC4yECRD0

通过该寄存器可以回读与捕获触发信号 0 相连的捕获功能的 FIFO 结构。回读操作仅在 **CC4yTC**.ECM = 1_B 时有效。

CC4yECRD0 (y = 0 - 3)

扩展回读 0 **($01B8_H$ + 0100_H * y)** **复位值 : 00000000_H**



域	位	类型	描述
CAPV	[15:0]	rh	**定时器捕获值** 该域包含定时器捕获的值
FPCV	[19:16]	rh	**预分频器捕获值** 该域包含与特定的 CAPV 域相关联的预分频器时钟分频值
SPTR	[21:20]	rh	**定时器片指针** 该域指示捕获值所在的定时器片的索引。 00_B CC40 01_B CC41 10_B CC42 11_B CC43

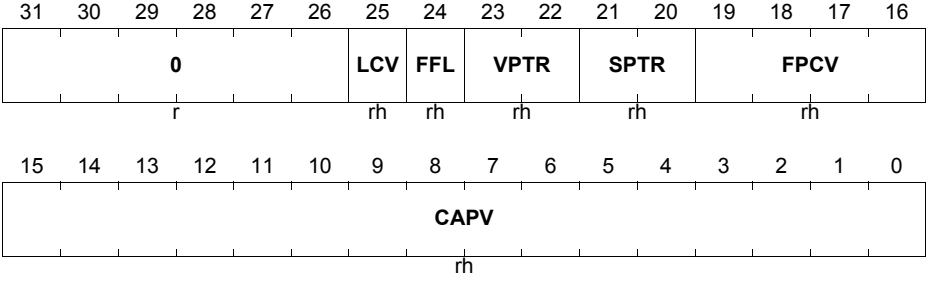
域	位	类型	描述
VPTR	[23:22]	rh	**捕获寄存器指针** 该域指示捕获值所在的捕获寄存器的索引。 00_B 捕获寄存器 0 01_B 捕获寄存器 1 10_B 捕获寄存器 2 11_B 捕获寄存器 3
FFL	24	rh	**满标志** 该位指示相关联的捕获寄存器是否包含一个新值。 0_B 没有新值被捕获到该寄存器 1_B 一个新值被捕获到该寄存器
LCV	25	rh	**丢失捕获值** 该域指示当 FIFO 结构已满时在两次读 ECRD0 之间是否产生了一个捕获触发信号。如果在两次读操作之间产生了一个捕获触发信号，则丢失了一个捕获值。只要发生对 ECRD 的读操作，该域由硬件自动清除。 0_B 没有丢失捕获值 1_B 丢失一个捕获值
0	[31:26]	r	**保留** 读访问总是返回 0。

CC4yECRD1

通过该寄存器可以回读与捕获触发信号 1 相连的捕获功能的 FIFO 结构。回读操作仅在 **CC4yTC**.ECM = 1_B 时有效。

CC4yECRD1 (y = 0 - 3)

扩展回读 1 **($01BC_H + 0100_H * y$)** **复位值 : 00000000_H**



域	位	类型	描述
CAPV	[15:0]	rh	**定时器捕获值** 该域包含定时器捕获的值
FPCV	[19:16]	rh	**预分频器捕获值** 该域包含与特定的 CAPV 域相关联的预分频器时钟分频值
SPTR	[21:20]	rh	**定时器片指针** 该域指示捕获值所在的定时器片的索引。 00_B CC40 01_B CC41 10_B CC42 11_B CC43
VPTR	[23:22]	rh	**捕获寄存器指针** 该域指示捕获值所在的捕获寄存器的索引。 00_B 捕获寄存器 0 01_B 捕获寄存器 1 10_B 捕获寄存器 2 11_B 捕获寄存器 3
FFL	24	rh	**满标记** 该位指示相关联的捕获寄存器包含一个新值。 0_B 没有新值被捕获到该寄存器 1_B 一个新值被捕获到该寄存器
LCV	25	rh	**丢失捕获值** 该域指示当 FIFO 结构已满时在两次读 ECRD0 之间是否产生了一个捕获触发信号。如果在两次读操作之间产生了一个捕获触发信号，则丢失了一个捕获值。只要发生对 ECRD 的读操作，该域由硬件自动清除。 0_B 没有丢失捕获值 1_B 丢失一个捕获值
0	[31:26]	r	**保留** 读访问总是返回 0

19.8 互连

涉及 “ 全局引脚 ” 的表格中都包含每个模块的所有定时器片共同的输入 / 输出。
关于 GPIO 连接，请参见端口一章。

19.8.1 CCU40 引脚

表 19-13 CCU40 引脚连接

全局输入 / 输出	I/O	连接到	描述
CCU40.MCLK	I	PCLK	内核时钟
CCU40.CLKA	I	未连接	预分频器的另一计数源
CCU40.CLKB	I	ERU0.IOUT0	预分频器的另一计数源
CCU40.CLKC	I	ERU0.IOUT1	预分频器的另一计数源
CCU40.MCSS	I	POSIF0.OUT6	多通道序列与映射传送触发信号之间的同步信号
CCU40.SR0	O	NVIC; CCU80.IGBTD; POSIF0.MSETA; ERU0.OGU01;	服务请求线
CCU40.SR1	O	NVIC; ERU0.OGU11;	服务请求线
CCU40.SR2	O	NVIC; VADC0.BGREQTRA; VADC0.G0REQTRA; VADC0.G1REQTRA; CCU80.IN0K; CCU80.IN1K; ERU0.OGU21;	服务请求线
CCU40.SR3	O	NVIC; VADC0.BGREQTRB; VADC0.G0REQTRB; VADC0.G1REQTRB; CCU80.IGBTB; CCU80.IN2K; CCU80.IN3K; ERU0.OGU31;	服务请求线

表 19-14 CCU40 - CC40 引脚连接

输入 / 输出	I/O	连接到	描述
CCU40.IN0A	I	P0.12	通用功能
CCU40.IN0B	I	P0.6	通用功能
CCU40.IN0C	I	P0.0	通用功能
CCU40.IN0D	I	ERU0.PDOUT1	通用功能
CCU40.IN0E	I	POSIF0.OUT0	通用功能
CCU40.IN0F	I	POSIF0.OUT1	通用功能
CCU40.IN0G	I	POSIF0.OUT3	通用功能
CCU40.IN0H	I	CCU80.ST3	通用功能
CCU40.IN0I	I	SCU.GSC40	通用功能
CCU40.IN0J	I	ERU0.PDOUT0	通用功能
CCU40.IN0K	I	ERU0.IOUT0	通用功能
CCU40.IN0L	I	USIC0_CH0.DX2INS	通用功能
CCU40.IN0M	I	CCU40.GP10	通用功能
CCU40.IN0N	I	CCU40.ST1	通用功能
CCU40.IN0O	I	CCU40.ST2	通用功能
CCU40.IN0P	I	CCU40.ST3	通用功能
CCU40.MCI0	I	BCCU0.OUT2	多通道序列输入
CCU40.OUT0	O	P0.0; P0.5; P0.6; P1.0; P2.0; P2.0. 硬件 1 方向控制	定时器片比较输出
CCU40.GP00	O	CCU40.IN3M	事件 0 的选择信号
CCU40.GP01	O	未连接	事件 1 的选择信号
CCU40.GP02	O	CCU80.IN0C	事件 2 的选择信号

表 19-14 CCU40 - CC40 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
CCU40.ST0	O	CCU40.IN1N; CCU40.IN2N; CCU40.IN3N; CCU80.IGBTC; POSIF0.HSDA; VADC0.BGREQGTD; VADC0.G0REQGTD; VADC0.G1REQGTD;	定时器片状态位
CCU40.PS0	O	未连接	多通道序列同步触发信号：当向上计数 (边沿对齐) 时为 PM，当向下计数 (中心对齐) 时为 OM。

表 19-15 CCU40 - CC41 引脚连接

输入 / 输出	I/O	连接到	描述
CCU40.IN1A	I	P0.12	通用功能
CCU40.IN1B	I	P0.7	通用功能
CCU40.IN1C	I	P0.1	通用功能
CCU40.IN1D	I	ERU0.PDOUT0	通用功能
CCU40.IN1E	I	POSIF0.OUT0	通用功能
CCU40.IN1F	I	POSIF0.OUT1	通用功能
CCU40.IN1G	I	POSIF0.OUT3	通用功能
CCU40.IN1H	I	POSIF0.OUT4	通用功能
CCU40.IN1I	I	SCU.GSC40	通用功能
CCU40.IN1J	I	ERU0.PDOUT1	通用功能
CCU40.IN1K	I	ERU0.IOUT1	通用功能
CCU40.IN1L	I	USIC0_CH1.DX2INS	通用功能
CCU40.IN1M	I	CCU40.GP20	通用功能
CCU40.IN1N	I	CCU40.ST0	通用功能
CCU40.IN1O	I	CCU40.ST2	通用功能
CCU40.IN1P	I	CCU40.ST3	通用功能
CCU40.MCI1	I	BCCU0.OUT3	多通道序列输入

表 19-15 CCU40 - CC41 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
CCU40.OUT1	O	P0.1: P0.4; P0.7; P1.1; P2.1; P2.1. 硬件 1 方向控制	定时器片比较输出
CCU40.GP10	O	CCU40.IN0M	事件 0 的选择信号
CCU40.GP11	O	未连接	事件 1 的选择信号
CCU40.GP12	O	CCU80.IN1C	事件 2 的选择信号
CCU40.ST1	O	CCU40.IN0N; CCU40.IN2O; CCU40.IN3O; POSIF0.MSETB; VADC0.BGREQGTC; VADC0.G0REQGTC; VADC0.G1REQGTC;	定时器片状态位
CCU40.PS1	O	POSIF0.MSYNCC	多通道序列同步触发信号：当向上计数 (边沿对齐) 时为 PM，当向下计数 (中心对齐) 时为 OM。

表 19-16 CCU40 - CC42 引脚连接

输入 / 输出	I/O	连接到	描述
CCU40.IN2A	I	P0.12	通用功能
CCU40.IN2B	I	P0.8	通用功能
CCU40.IN2C	I	P0.2	通用功能
CCU40.IN2D	I	ERU0.PDOUT3	通用功能
CCU40.IN2E	I	POSIF0.OUT1	通用功能
CCU40.IN2F	I	POSIF0.OUT2	通用功能
CCU40.IN2G	I	POSIF0.OUT3	通用功能
CCU40.IN2H	I	POSIF0.OUT4	通用功能
CCU40.IN2I	I	SCU.GSC40	通用功能
CCU40.IN2J	I	ERU0.PDOUT2	通用功能
CCU40.IN2K	I	ERU0.IOUT2	通用功能

表 19-16 CCU40 - CC42 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
CCU40.IN2L	I	保留	通用功能
CCU40.IN2M	I	CCU40.GP30	通用功能
CCU40.IN2N	I	CCU40.ST0	通用功能
CCU40.IN2O	I	CCU40.ST1	通用功能
CCU40.IN2P	I	CCU40.ST3	通用功能
CCU40.MCI2	I	BCCU0.OUT4	多通道序列输入
CCU40.OUT2	O	P0.2; P0.8; P1.2; P2.10; P2.8. 硬件 1 拉控制 P2.9. 硬件 1 拉控制 P2.10. 硬件 1 方向控制	定时器片比较输出
CCU40.GP20	O	CCU40.IN1M	事件 0 的选择信号
CCU40.GP21	O	未连接	事件 1 的选择信号
CCU40.GP22	O	CCU80.IN2C	事件 2 的选择信号
CCU40.ST2	O	CCU40.IN0O; CCU40.IN1O; CCU40.IN3P; VADC0.BGREQGTB; VADC0.G0REQGTB; VADC0.G1REQGTB; POSIF0.MSETC;	定时器片状态位
CCU40.PS0	O	未连接	多通道序列同步触发信号：当向上计数 (边沿对齐) 时为 PM，当向下计数 (中心对齐) 时为 OM。

表 19-17 CCU40 - CC43 引脚连接

输入 / 输出	I/O	连接到	描述
CCU40.IN3A	I	P0.12	通用功能
CCU40.IN3B	I	P0.9	通用功能
CCU40.IN3C	I	P0.3	通用功能
CCU40.IN3D	I	ERU0.PDOUT2	通用功能

表 19-17 CCU40 - CC43 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
CCU40.IN3E	I	POSIF0.OUT3	通用功能
CCU40.IN3F	I	POSIF0.OUT5	通用功能
CCU40.IN3G	I	VADC0.G0ARBCNT	通用功能
CCU40.IN3H	I	CCU80.IGBTO	通用功能
CCU40.IN3I	I	SCU.GSC40	通用功能
CCU40.IN3J	I	ERU0.PDOUT3	通用功能
CCU40.IN3K	I	ERU0.IOUT3	通用功能
CCU40.IN3L	I	保留	通用功能
CCU40.IN3M	I	CCU40.GP00	通用功能
CCU40.IN3N	I	CCU40.ST0	通用功能
CCU40.IN3O	I	CCU40.ST1	通用功能
CCU40.IN3P	I	CCU40.ST2	通用功能
CCU40.MCI3	I	BCCU0.OUT5	多通道序列输入
CCU40.OUT3	O	P0.3; P0.9; P1.3; P2.11; P2.2. 硬件 1 拉控制 P2.6. 硬件 1 拉控制 P2.7. 硬件 1 拉控制 P2.11. 硬件 1 方向控制	定时器片比较输出
CCU40.GP30	O	CCU40.IN2M	事件 0 的选择信号
CCU40.GP31	O	未连接	事件 1 的选择信号
CCU40.GP32	O	CCU80.IN3C	事件 2 的选择信号

表 19-17 CCU40 - CC43 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
CCU40.ST3	O	CCU40.IN0P; CCU40.IN1P; CCU40.IN2P; VADC0.BGREQGTA; VADC0.G0REQGTA; VADC0.G1REQGTA; CCU80.IGBTA; POSIF0.MSETD;	定时器片状态位
CCU40.PS3	O	未连接	多通道序列同步触发信号：当向上计数 (边沿对齐) 时为 PM，当向下计数 (中心对齐) 时为 OM。

20 捕获比较单元 8 (CCU8)

对于需要产生复杂脉冲宽度调制 (PWM) 信号并具有互补高边和低边开关、多相控制或输出奇偶校验的应用而言， CCU8 外设功能起重要作用。这些功能与非常灵活和可编程的信号调理机制相结合，使 CCU8 成为现代电机控制、多相和多电平电源电子系统中不可或缺的一个外设。

CCU8 内部的模块化结构使其成为一个软件友好的系统，方便进行快速代码开发和应用程序之间的移植。

表 20-1 缩略语表

PWM	脉冲宽度调制
CCU8x	捕获比较单元 8 模块实例 x
CC8y	捕获比较单元 8 定时器片实例 y
ADC	模 / 数转换器
POSIF	位置接口外设
SCU	系统控制单元
f_{ccu8}	CCU8 模块时钟频率
f_{tclk}	CC8y 定时器时钟频率

注： 寄存器中的小写字母 “y” 或 “x” 代表一个索引。

20.1 概述

CCU8 单元由 4 个完全相同的 16 位捕获 / 比较定时器片 (CC8y) 构成。每个定时器片都可以工作在比较模式或捕获模式。在比较模式，每个定时器片有两个专用比较通道，可产生最多 4 路 PWM 信号 (每个 CCU8 单元可产生最多 16 路 PWM 输出)，并可插入死区时间以防止开关短路。在捕获模式，最多可以使用 4 个捕获寄存器。

每个 CCU8 模块有 4 个服务请求线，可以将这些服务请求线编程为 PWM 信号产生与 ADC 转换之间的同步触发信号。

也可以将定时器片直接级联，实现最多 64 位的定时操作。这就提供了一种灵活的测频、倍频和脉宽调制方案。

每个定时片都有一个可编程的功能输入选择器，最多支持 9 种不同的功能，但可能因可用输入端口的限制而不需对所有资源进行映射。

与 POSIF 模块之间的内建连接使 CCU8 可与无刷直流电机控制所使用的霍尔传感器直接耦合，可实现灵活的数字电机控制回路。

20.1.1 特性

CCU8 模块特性

每个 CCU8 由 4 个定时器片组合而成，这些定时器片可以独立工作在比较模式或捕获模式。每个定时器片都有 4 个用于产生 PWM 信号的专用输出。

CCU8 的所有 4 个定时器片 CC8y 都具有完全相同的可用功能和工作模式。这就使得在实现不同的软件程序时可以避免对所用 CCU8 具体资源的依赖。

这 4 个定时器片之间还有内建连接，可以实现简单的定时器级联和顺序操作。

一般特性

- 16 位定时器单元
- 可编程的输入低通滤波器
- 内建定时器级联
  - 32、 48 或 64 位宽
- 周期值和比较值的映射传送
- 捕获模式有 4 个捕获寄存器
- 可编程的时钟预分频器
- 标准定时器模式
- 门控定时器模式
- 三种计数方式
  - 中心对齐
  - 边沿对齐
  - 单次方式
- PWM 信号产生
- 非对称 PWM 产生
- 陷阱功能
- 死区时间产生
- 启停可由外部事件控制
- 对外部事件计数
- 每个 CCU8 有 4 个专用的服务请求线

其他特性

- 外部调制功能
- 由外部事件控制加载
- 抖动 PWM
- 浮动预分频器
- 输出状态可被外部事件盖写
- 可编程的输出奇偶校验器
- 与 POSIF 连接方便
  - 霍尔传感器模式
  - 旋转编码器模式

– 多通道 / 多相控制

CCU8 特性与应用的对应关系

表 20-2 概括了 CCU8 单元的主要特性与最常见应用的对应关系。

表 20-2 应用概览

特性	应用
4 个独立的定时器单元	独立的 PWM 信号产生： • 多个降压 / 升压转换器控制 (使用独立的频率) • 每个定时器有不同的工作模式，提高资源优化程度 • 最多可实现 2 个半桥控制 • 多个零电压开关 (ZVS) 转换器控制，易与 ADC 通道连接。 • 多电平转换器
每个定时器片有两个比较通道	连接两个比较通道或连接两个定时器片： • 非对称 PWM 信号产生能力减少电流传感器的数量 • 连接两个定时器片的能力允许移相全桥拓扑控制 • 连接两个定时器片的能力允许 N 相 DC/DC 变换器控制
两个死区事件发生器	独立的上升和下降跳变死区时间值以及独立的通道死区时间计数器： • 每个通道都可以使用不同的死区时间值独立工作。这就允许用不同的死区时间值和相同的频率控制最多两个半桥。 • 不同的上升和下降跳变死区时间值可用于优化 MOSFET 的开关行为
级联的定时器单元	易于实现可多达 64 位的定时器扩展： • 高动态触发信号捕获 • 高动态信号测量
抖动 PWM	产生极小的 PWM 频率或占空比： • 在低速控制回路应用中可避免频率或占空比的大步距调节 • 提高 PWM 信号随时间的分辨率
浮动预分频器	自动进行控制信号测量： • 减少监测高动态或未知信号的软件工作 • 可仿真一个大于 16 位的定时器用于系统控制

表 20-2 应用概览 (续表)

特性	应用
每个定时器可通过外部信号实现最多 9 种功能	灵活的资源优化： • 所有外部功能总是可用 • CCU8 内部可实现多种配置，例如，一个定时器工作在捕获模式，而另一个定时器工作在比较模式。
输出奇偶校验器	自动进行 MOSFET 信号监视： • 奇偶校验器可用于监视 IGBT 的输出，并将其与 CCU8 的整组 PWM 输出进行比较。 • 避免在一个多 MOSFET 系统中发生短路。
4 条专用的服务请求线	专门用于： • 向微处理器发出中断 • 实现外设间的灵活连接，例如 ADC 触发。
与 POSIF 连接	灵活的配置适合： • 旋转编码器连接 • 霍尔传感器 • 用软件调制 4 个定时器输出

20.1.2 框图

每个 CCU8 定时器片可以独立于其他片工作在所有可用模式。每个定时器片都包含一个专用输入选择器，用于那些需连接外部事件的功能，还有 4 个用于产生 PWM 信号的专用比较输出信号。

内建的定时器级联功能仅限于相邻的定时器片之间，例如，CC80/CC81。诸如 CC80/CC82 或 CC80/CC83 之类的定时器片级联组合是不可能的。

每个定时器片自身的服务请求 (每个定时器片有 4 个服务请求线) 被多路复用到 4 条模块服务请求线，如图 20-1 所示。

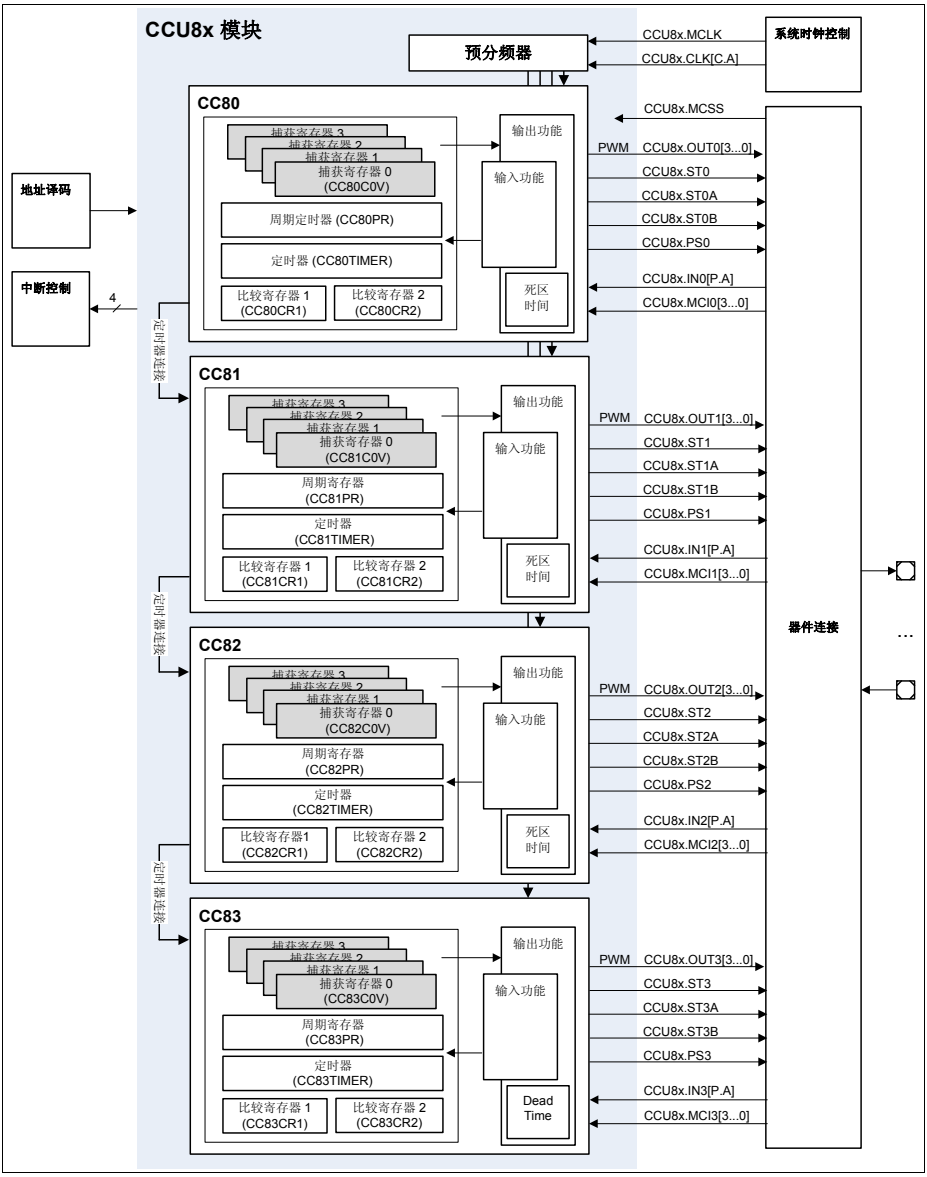

CCU8x 模块
预分频器
CCU8x.MCLK
CCU8x.CLK[C.A]
系统时钟控制
CC80
CCU8x.MCSS
捕获寄存器 3
捕获寄存器 2
捕获寄存器 1
捕获寄存器 0
(CC80C0V)
输出功能
输入功能
PWM
CCU8x.OUT0[3...0]
CCU8x.ST0
CCU8x.ST0A
CCU8x.ST0B
CCU8x.PS0
地址译码
周期定时器 (CC80PR)
定时器 (CC80TIMER)
中断控制
4
比较寄存器 1
(CC80CR1)
比较寄存器 2
(CC80CR2)
死区
时间
CCU8x.IN0[P.A]
CCU8x.MCI0[3...0]
定时器连接
CC81
捕获寄存器 3
捕获寄存器 2
捕获寄存器 1
捕获寄存器 0
(CC81C0V)
周期寄存器
(CC81PR)
定时器
(CC81TIMER)
比较寄存器 1
(CC81CR1)
比较寄存器 2
(CC81CR2)
CCU8x.OUT1[3...0]
CCU8x.ST1
CCU8x.ST1A
CCU8x.ST1B
CCU8x.PS1
CCU8x.IN1[P.A]
CCU8x.MCI1[3...0]
CC82
捕获寄存器 3
捕获寄存器 2
捕获寄存器 1
捕获寄存器 0
(CC82C0V)
周期寄存器
(CC82PR)
定时器
(CC82TIMER)
比较寄存器1
(CC82CR1)
比较寄存器 2
(CC82CR2)
CCU8x.OUT2[3...0]
CCU8x.ST2
CCU8x.ST2A
CCU8x.ST2B
CCU8x.PS2
CCU8x.IN2[P.A]
CCU8x.MCI2[3...0]
CC83
捕获寄存器 3
捕获寄存器 2
捕获寄存器 1
捕获寄存器 0
(CC83C0V)
周期寄存器
(CC83PR)
定时器
(CC83TIMER)
比较寄存器 1
(CC83CR1)
比较寄存器 2
(CC83CR2)
Dead
Time
CCU8x.OUT3[3...0]
CCU8x.ST3
CCU8x.ST3A
CCU8x.ST3B
CCU8x.PS3
CCU8x.IN3[P.A]
CCU8x.MCI3[3...0]
器件连接
...

图 20-1　　CCU8 框图

20.2 功能描述

20.2.1 概述

一个 CCU8 片的输入通路由一个选择器（见 **20.2.2 节**）和一个连接矩阵单元（见 **20.2.3 节**）组成。输出通路包含一个服务请求控制单元、一个定时器级联单元和两个直接控制每个特定定时器片输出信号状态的单元 (用于陷阱功能和调制处理)，如 **图 20-2** 所示。

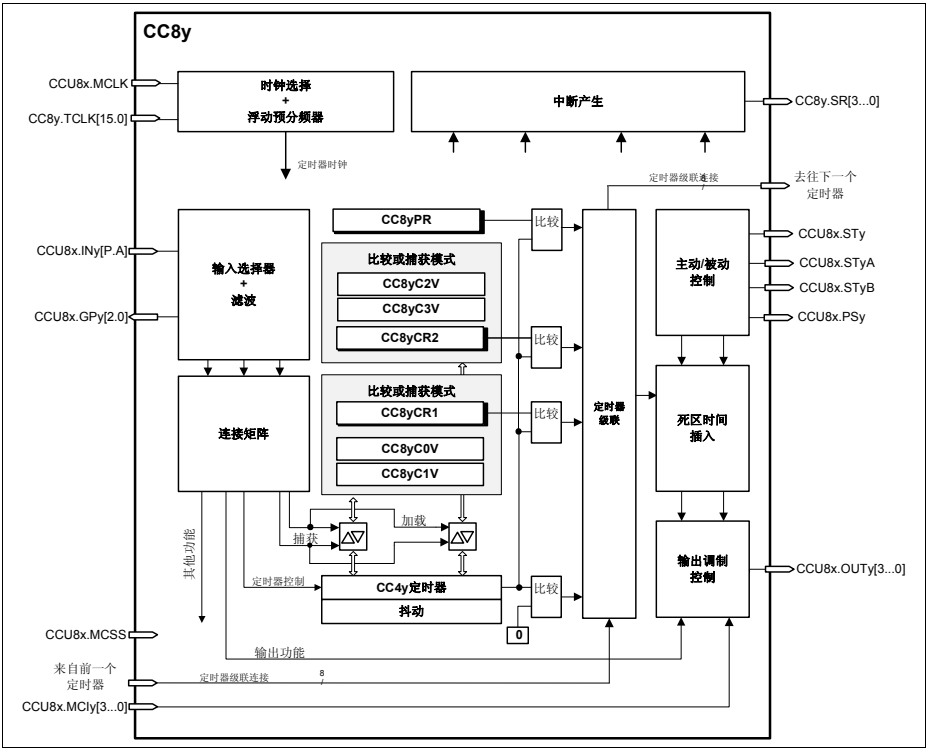


图 20-2 CCU8 定时器片框图

在比较模式，定时器内核由一个 16 位的计数器、一个周期寄存器和 2 个比较通道构成；在捕获模式，定时器内核由 4 个捕获通道加上周期寄存器构成。

可以为每个模块以及每个定时器片单独选择定时器时钟，这就使得可以在每个 CCU8 模块内部进行非常灵活的资源组织。

在比较模式，周期寄存器设置最大计数值，两个比较寄存器则用于控制 4 个专用比较片输出的主动 / 被动状态。

每个 CCU8 定时器片包含一个专用的定时器链接接口，用于执行定时器级联功能，可最多级联到 64 位。这种定时器级联通过配置一个位域来控制。

表 20-3 描述了每个 CCU8 定时器片的输入和输出。

在 CCU8 的边界看不到的输入和输出被命名为 CC8y.<name>，而 CCU8 模块的输入和输出被描述为 CCU8x.<signal_name>y (变量 y 表示目标定时器片)。

表 20-3 CCU8 定时器片引脚描述

引脚	I/O	描述
CCU8x.MCLK	I	模块时钟
CC8y.TCLK		来自预分频器的时钟
CCU8x.INy[P:A]	I	定时器片功能输入 (用于控制通过定时器片外部事件连接的功能)
CCU8x.MCIy[3...0]	I	多通道模式输入
CCU8x.MCSS	I	多通道映射传送触发信号
CC8y.SR[3...0]	O	定时器片服务请求线
CCU8x.GPy[2...0]	O	从输入选择器译码的信号 (用于奇偶校验器功能)
CCU8x.STy	O	该信号可以是通道 1、通道 2 的定时器片比较状态值，也可以这两个比较状态值相与的结果。
CCU8x.STyA	O	通道 1 的定时器片比较状态值
CCU8x.STyB	O	通道 2 的定时器片比较状态值
CCU8x.PSy	O	周期匹配
CCU8x.OUTy[3...0]	O	定时器片专用输出引脚

注：

1. *内核的状态位输出CCU8x.STy、CCU8x.STyA和CCU8x.STyB被延长一个内核时钟周期。*
2. *内核输出的服务请求信号被延长一个内核时钟周期。*
3. *输出信号 CCU8x.STy、CCU8x.STyA 和 CCU8x.STyA 的最大频率是模块时钟的 4 分频。*

根据所选择的工作模式，片定时器可以向上或向下计数。实际的计数方向在一个方向标志中保存。

定时器与 3 个独立的比较器相连，其中一个用于周期匹配，另两个用于每个比较通道的比较匹配。用于两个比较通道比较匹配的寄存器可以被编程作为捕获寄存器使用，实现对外部事件的连续捕获。

在标准边沿对齐计数方式下，每当计数值与周期寄存器中设定的周期值相匹配时，计数器被清为 0000_H 。在中心对齐模式，在定时器达到周期值之后，计数方向从“向上计数”变为“向下计数”。周期寄存器和比较寄存器都有一个共用的映射寄存器，该映射寄存器可以在运行时修改 PWM 的周期和占空比。

计数器还有单次模式，在该模式下，计数器达到周期寄存器中设定的值后会停止计数。

计数器的启动和停止可以通过软件访问或一个可编程的输入引脚来置位 / 清零。

可以将死区时间发生器编程为对输出的上升沿和下降沿使用不同的死区时间值。

像加载、计数方向 (向上 / 向下)、陷阱和输出调制这样的功能也可以用外部事件控制，参见 **20.2.3 节**。

20.2.2 输入选择器

这是定时器片输入通路的第一个单元，用于选择使用哪个输入来控制可用的外部功能。

在该块内部，用户还可以对信号进行低通滤波，选择外部信号的有效边沿或有效电平，如**图 20-3** 所示。

用户可以选择 CCU8x.INy[P:A] 输入中的任何一个作为事件源。

在该单元的输出，用户可以选择三个事件，它们可被配置为上升沿有效、下降沿有效、两个边沿都有效或电平有效。然后，这些选定的事件可以被映射到多种功能。

注意，每个被译码的事件包括两个输出，一个是边沿有效，一个是电平有效。这是因为有些功能如计数、捕获或加载是边沿敏感的，而定时器门控或向上 / 向下计数选择是电平有效的。

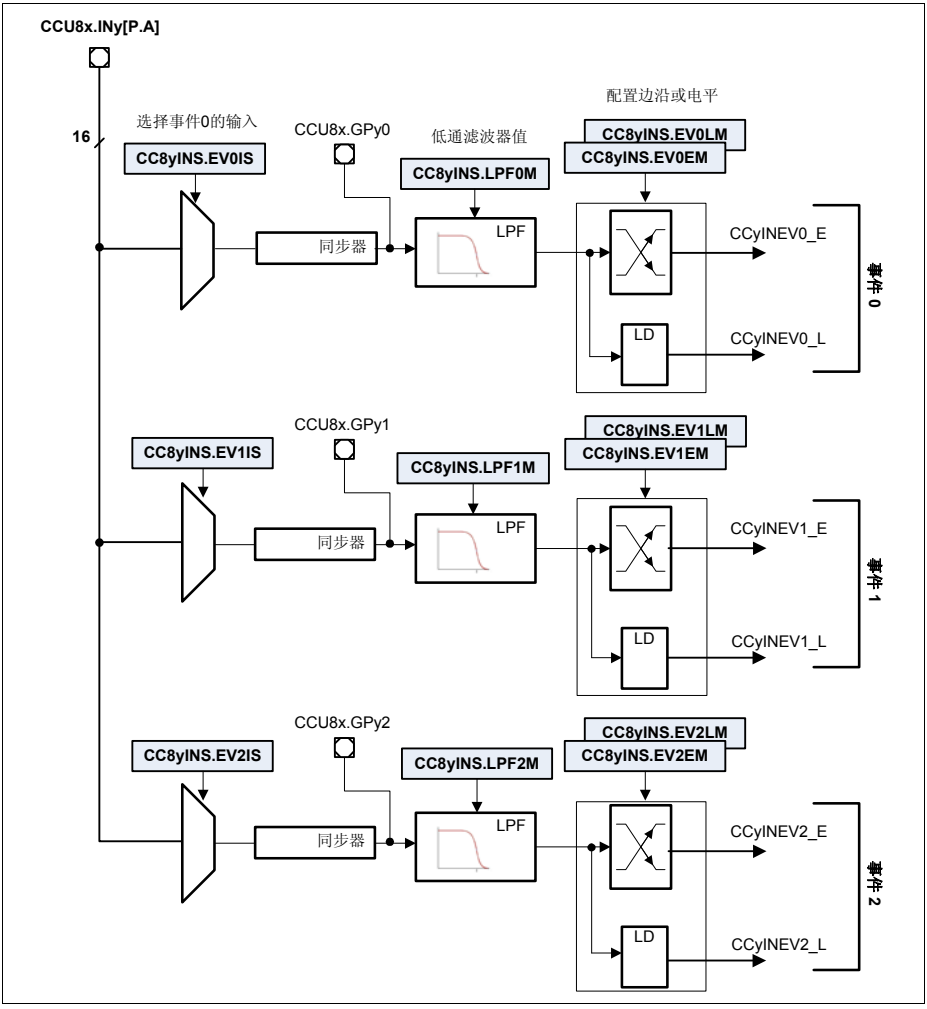


图 20-3　　定时器片输入选择器框图

20.2.3　连接矩阵

连接矩阵将来自输入选择器的事件映射到多个用户配置的功能，如**图 20-4** 所示。在连接矩阵中可以使能下述功能：

表 20-4 连接矩阵的可用功能

功能	简要描述	映射到 图 20-4
启动	边沿信号，用于启动定时器	CCystrt
停止	边沿信号，用于停止定时器	CCystp
计数	边沿信号，用于对事件计数	CCycnt
向上 / 向下	电平信号，用于选择向上或向下计数方向	CCyupd
捕获 0	边沿信号，用于触发一次捕获到捕获寄存器 0 和捕获寄存器 1	CCycapt0
捕获 1	边沿信号，用于触发一次捕获到捕获寄存器 2 和捕获寄存器 3	CCycapt1
门控	电平信号，用于门控定时器时钟	CCygate
加载	边沿信号，用于将比较寄存器中的值加载到定时器	CCyload
陷阱	电平信号，用于故障保护操作	CCytrap
调制	电平信号，用于调制或清除输出	CCymod
状态位覆盖	状态位，将被一个输入值覆盖	CCyoval 用作覆盖值 CCyoset 用作触发信号

连接矩阵内部还有一个实现内建定时器级联的单元。这种级联能使被级联的定时器片之间实现完全同步的定时操作，也可以实现捕获操作和加载操作的完全同步。定时器片的级联通过 **CC8yCMC**.TCE 位域来实现。关于级联功能的详细描述，请参见 **20.2.10 节**。

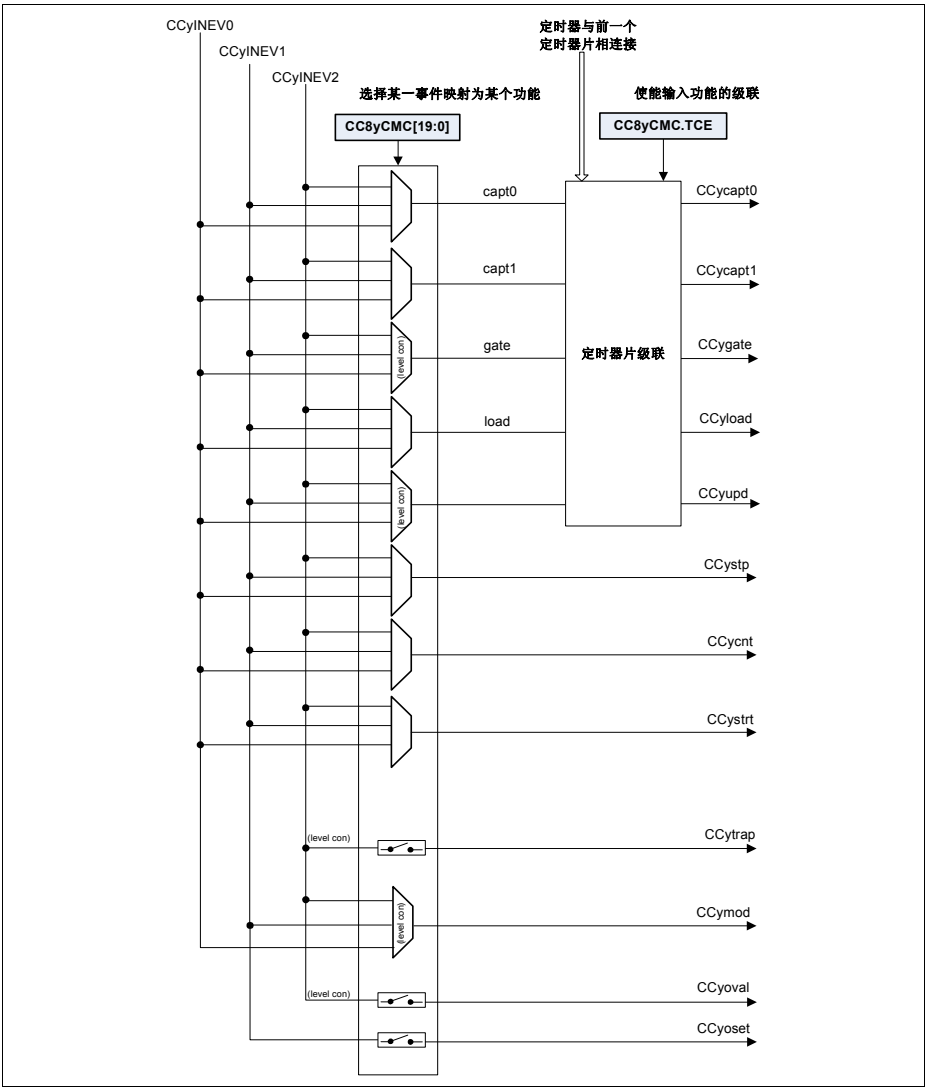


图 20-4　　定时器片连接矩阵示意图

20.2.4 启动 / 停止控制

每个定时器片都有一个运行位寄存器 **CC8yTCST**.TRB，它指示定时器的当前状态。定时器的启动和停止可以通过软件访问来完成，也可以由外部事件直接控制，如**图 20-5** 所示。

选择用一个外部信号作为启动触发信号并不要求用户必须同时使用一个外部信号作为停止触发信号，反之亦然。

选择单次模式意味着在计数器达到周期值后，运行位 **CC8yTCST**.TRB 将被清零，定时器因此而停止工作。

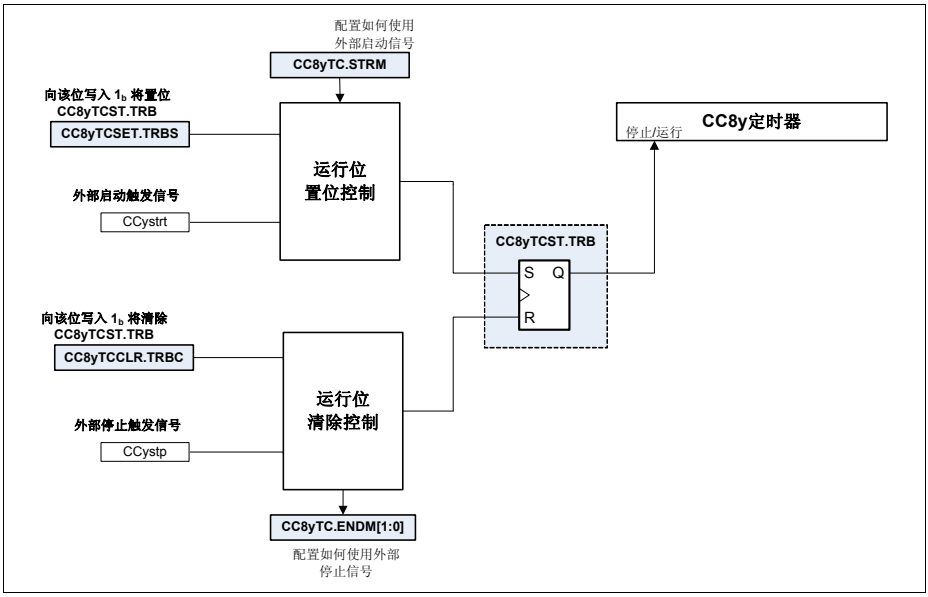


图 20-5 定时器启动 / 停止控制示意图

可以使用外部停止信号来执行下述功能 (通过 **CC8yTC**.ENDM 来配置)：

- 清除运行位 (停止定时器) - 默认
- 清除定时器 (到 0000_H) 但不清除运行位 (定时器仍然运行)
- 清除定时器和运行位

可以使用外部启动信号来执行下述功能 (通过 **CC8yTC**.STRM 来配置)：

- 启动定时器 (恢复运行)
- 清除并启动定时器

定时器运行位的置位 (启动定时器) 操作总是优先于清除 (停止定时器) 操作。

为了同时 / 同步启动多个 CCU8 定时器，应当使用一个专用输入作为外部启动信号 (关于如何将一个输入信号配置为启动功能的描述，请参见 **20.2.8.1 节**)。该输入信号应被连接

到需要同步启动的所有定时器 (关于模块连接的完整列表，请参见 20.8 节)，如图 20-6 所示。

对于通过软件来同步启动定时器，要有一个专用的输入信号与所有的 CCU8 定时器相连接，该输入信号由 SCU(系统控制单元) 来控制。该信号应被配置为外部启动信号 (请参见 20.2.8.1 节)，然后软件必须写 1_B 到 CCUCON 寄存器的特定位域 (该寄存器在 SCU 一章描述)。

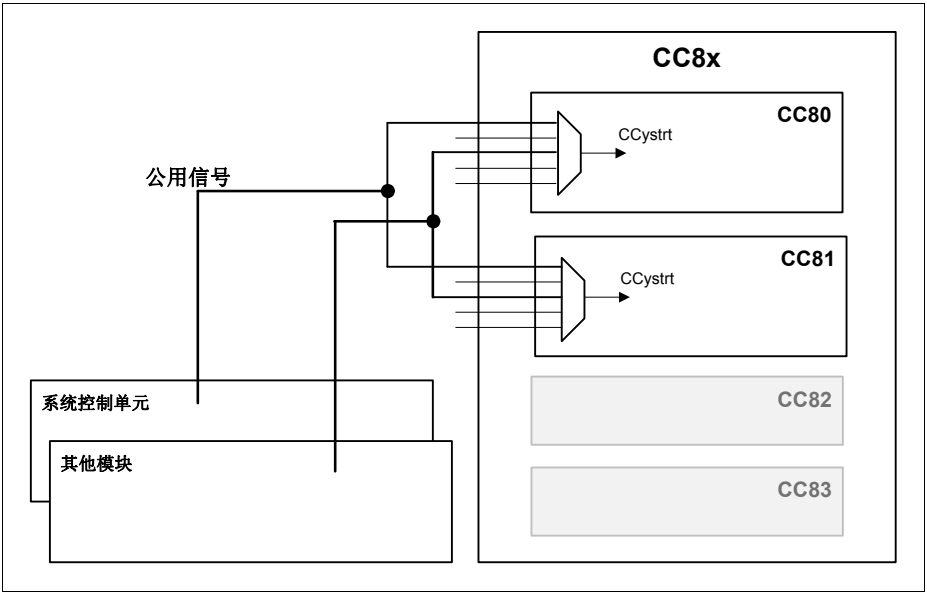


图 20-6　　同步启动多个定时器

20.2.5　计数模式

每个 CC8y 定时器片都可以被编程为三种不同的计数方式：

- 边沿对齐 (默认)
- 中心对齐
- 单次方式 (边沿对齐或中心对齐)

这三种计数方式可以独立使用而不需选定任何输入作为外部事件源。不过，也可以通过外部事件来控制计数操作，如定时器门控信号、计数触发信号、外部停止信号、外部启动信号等。

对于所有计数模式，在运行时修改定时器周期值和比较通道值都是可能的。这就能够在每个时钟周期更新 PWM 信号的频率和占空比。

CC8y 定时器片的每个比较通道都有一个相关联的状态位 (GCST.CC8yST1 用于比较通道 1， GCST.CC8yST2 用于比较通道 2)，它指示通道是处于主动态还是被动态，如图

20-7 所示。状态位的置位和清除以及各自 PWM 信号的产生是由定时器的周期值、比较值和当前计数模式来控制的。关于如何置位和清除状态位，请参见 20.2.5.3 节 到 20.2.5.5 节对不同计数模式的描述 。

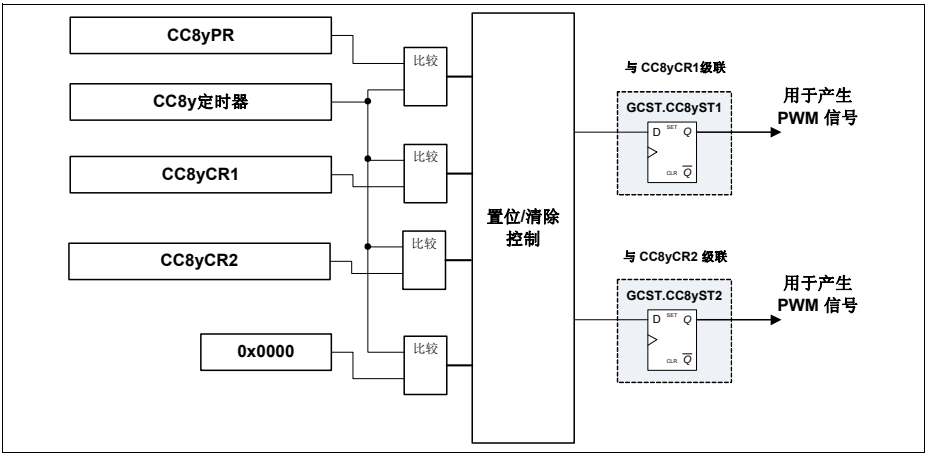


图 20-7　　CC8y 状态位

20.2.5.1　计算 PWM 信号的周期和占空比

定时器的周期由周期寄存器 **CC8yPR** 中的值和定时器的模式决定。

PWM 信号的周期和占空比的基准总是与定时器自身的时钟频率相关，而与模块时钟的频率无关 (因为定时器的时钟可以是模块时钟的一个分频版本)。

在边沿对齐模式，定时器的周期为：

T_{per}= < 周期值 > + 1; 以 f_{tclk} 计数　　(20.1)

在中心对齐模式，定时器的周期为：

T_{per}= (< 周期值 > + 1) x 2; 以 f_{tclk} 计数　　(20.2)

对于上述每种计数方式，所产生的 PWM 信号的占空比由比较通道寄存器 **CC8yCR1** 和 **CC8yCR2** 中的编程值控制。注意：这两个比较通道 可以有不同的占空比值。

在边沿对齐和中心对齐模式， PWM 信号的占空比为：

DC = 1 - < 比较值 >/(< 周期值 > + 1)　　(20.3)

周期寄存器 **CC8yPR** 和比较寄存器 **CC8yCR1**、 **CC8yCR2** 都可以在运行时通过软件修改，这样可使所产生的 PWM 信号在不同周期值及占空比值之间过渡时可以避免出现毛刺，详见 20.2.5.2 节。

20.2.5.2 更新周期和占空比

标准映射传送控制

每个 CCU8 定时器片为周期值和两个比较值分别提供一个相关联的映射寄存器。这就为通过软件来同时更新这 3 个参数提供了方便，其目的是可以在运行期间修改 PWM 信号的周期和占空比。

除了周期值和比较值的映射寄存器外，还有用于浮动预分频器、抖动功能和被动电平的映射寄存器，分别是 **CC8yFPCS**、**CC8yDITS** 和 **CC8yPSL** (关于这些功能的详细描述，请参见 **20.2.13 节** 和 **20.2.12 节**)。

映射寄存器的结构如**图 20-8** 所示。

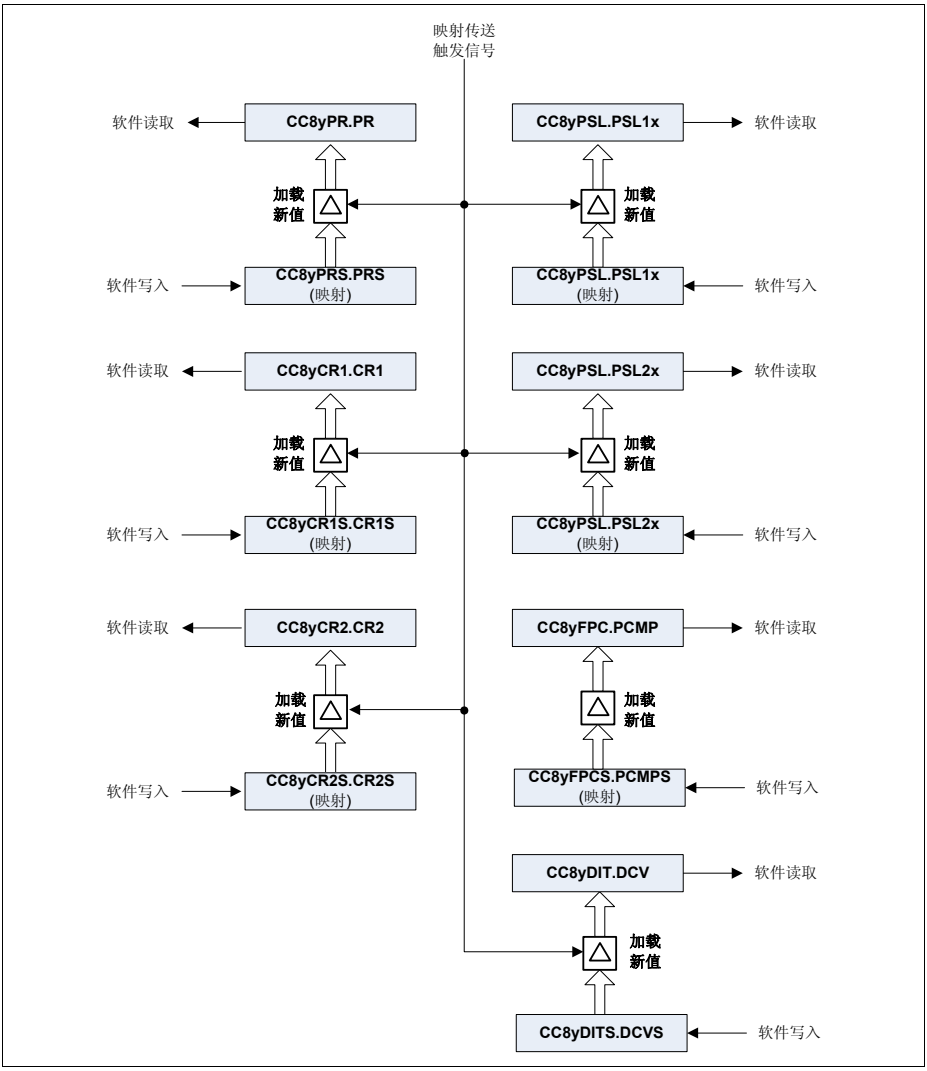


图 20-8　　映射寄存器概览

这些寄存器的更新只能通过向相关联的映射寄存器写一个新值并等待映射传送发生后完成。

每组映射寄存器有一个单独的映射传送使能位，如图 20-9 所示。当需要更新周期值和比较值时，软件必须将该使能位设置为 1_B。当完成值的更新时，这些使能位会被硬件自动清除。因此，每当需要更新寄存器时，软件必须再一次置位特定的使能位。

当然，也可以通过软件来清除使能位。这种方法可以用在值的更新操作需要被取消的情况下 (使能位已被置位之后)。

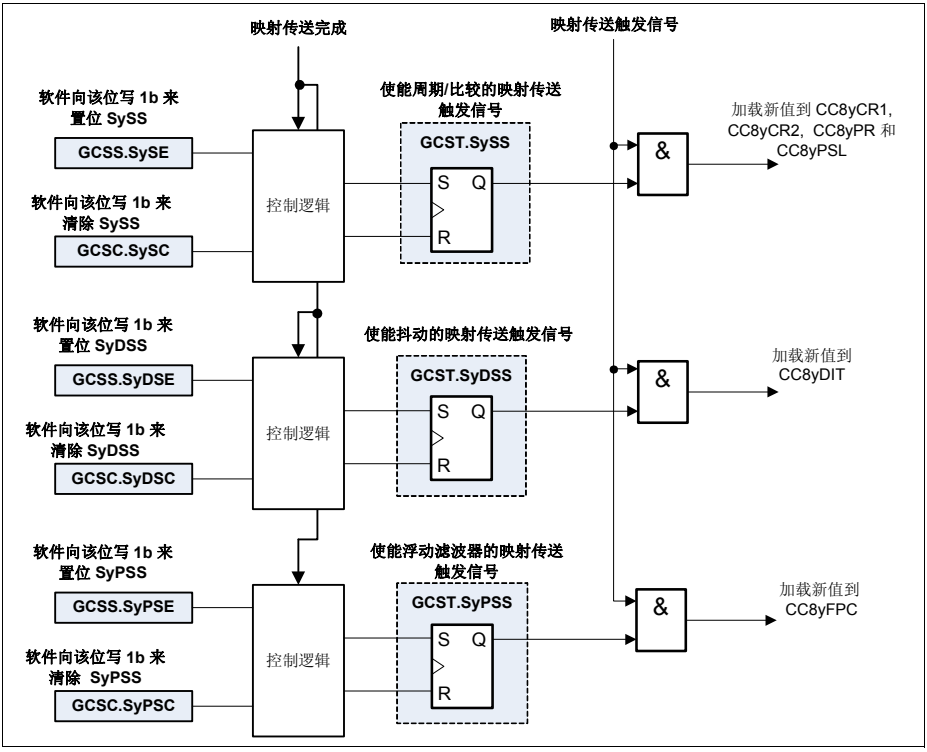


图 20-9　　映射传送使能逻辑

在映射传送使能位被置位 (**GCST**.SySS、**GCST**.SyDSS、**GCST**.SyPSS 设置为 1_B) 后，在紧接着出现映射传送触发信号时，会完成映射传送操作。

映射传送触发信号的发生取决于定时器的计数方式 (边沿对齐或中心对齐)。因此，寄存器值的更新时隙可以是：

- 向上计数期间发生周期匹配后的下一个时钟周期
- 向下计数期间发生 1 匹配后的下一个时钟周期
- 如果定时器停止且映射传送使能位被置位，则立即更新

图 20-10 给出了映射传送控制的一个例子，此处定时器被配置为中心对齐模式。关于所有定时器片计数模式的详细描述，请参见 **20.2.5.3 节**、 **20.2.5.4 节**和 **20.2.5.5 节**。

可以通过 STM 位域控制在哪一个时隙完成映射传送。这仅限于中心对齐模式：

- **CC8ySTC**.STM = 00_B (默认) - 映射传送在周期匹配和 1 匹配时隙完成
- **CC8ySTC**.STM = 01_B - 映射传送只在周期匹配时隙完成
- **CC8ySTC**.STM = 10_B - 映射传送只在 1 匹配时隙完成

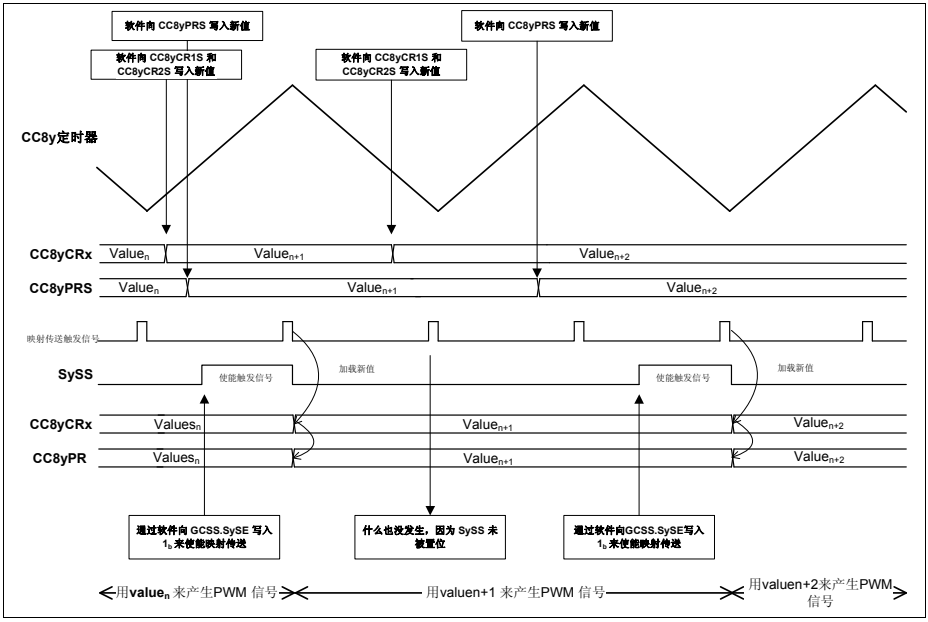


图 20-10　　映射传送时序示例 - 中心对齐模式

当将 CCU8 与 POSIF 结合使用来控制多通道模式时，在某些情况下映射传送的执行必须与多通道序列的更新同步。为了执行这一操作，每个 CCU8 都包含一个可用于同步这两个事件的专用输入 CCU8x.MCSS。

当被使能时，该输入用于置位特定定时器片的映射传送使能位域 (**GCST**.SySS, **GCST**.SyDSS 和 **GCST**.SyPSS) 。可以选择由哪一个定时器片来使用该输入执行该同步操作。CCU8x.MCSS 输入由 **GCTRL**.MSEy 位域使能。还可以将该信号用作 3 个不同的映射传送信号：比较和周期值、抖动比较值、预分频比较值。这可以通过配置 **GCTRL**.MSDE 域来实现。

使用 CCU8x.MCSS 输入信号的结构如**图 20-11** 所示。该信号的使用只是对映射传送控制的一个补充，因此所有前述功能仍然可用。

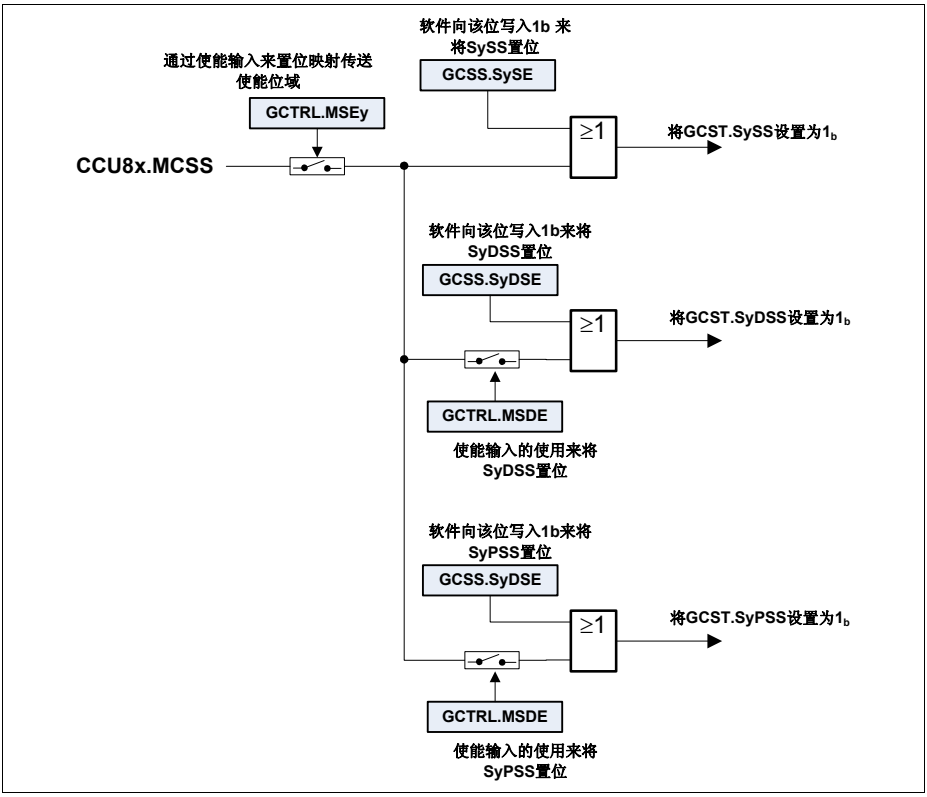


图 20-11　　使用 CCU8x.MCSS 输入

级联映射传送

可以将映射传送操作在 CCU8 定时器片中进行级联。**图 20-12** 给出了一个定时器片的一次特定映射传送与相邻定时器片级联的例子。

要使能级联映射传送功能，需要将特定定时器片的位域 **CC8ySTC**.CSE 设置为 1_B。

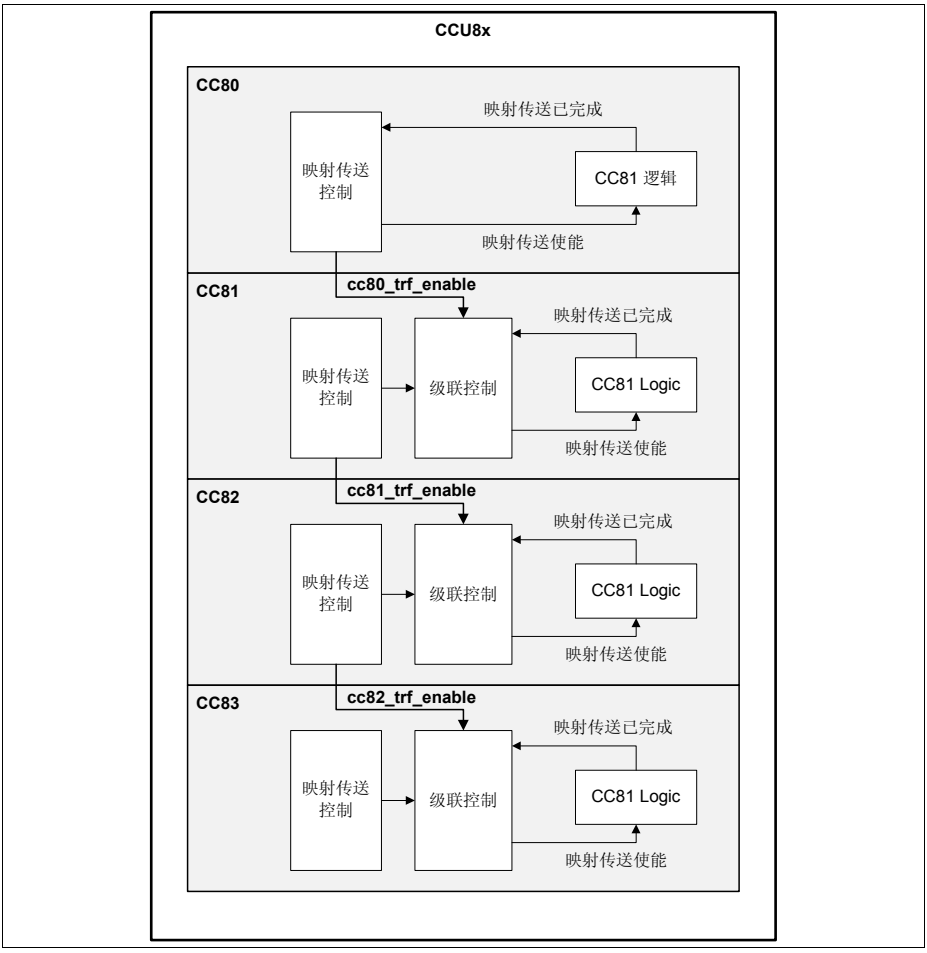


图 20-12　　级联映射传送连接

在这种情况下，仍然需要用软件将每个定时器片的映射传送使能位置1，如图20-13所示。

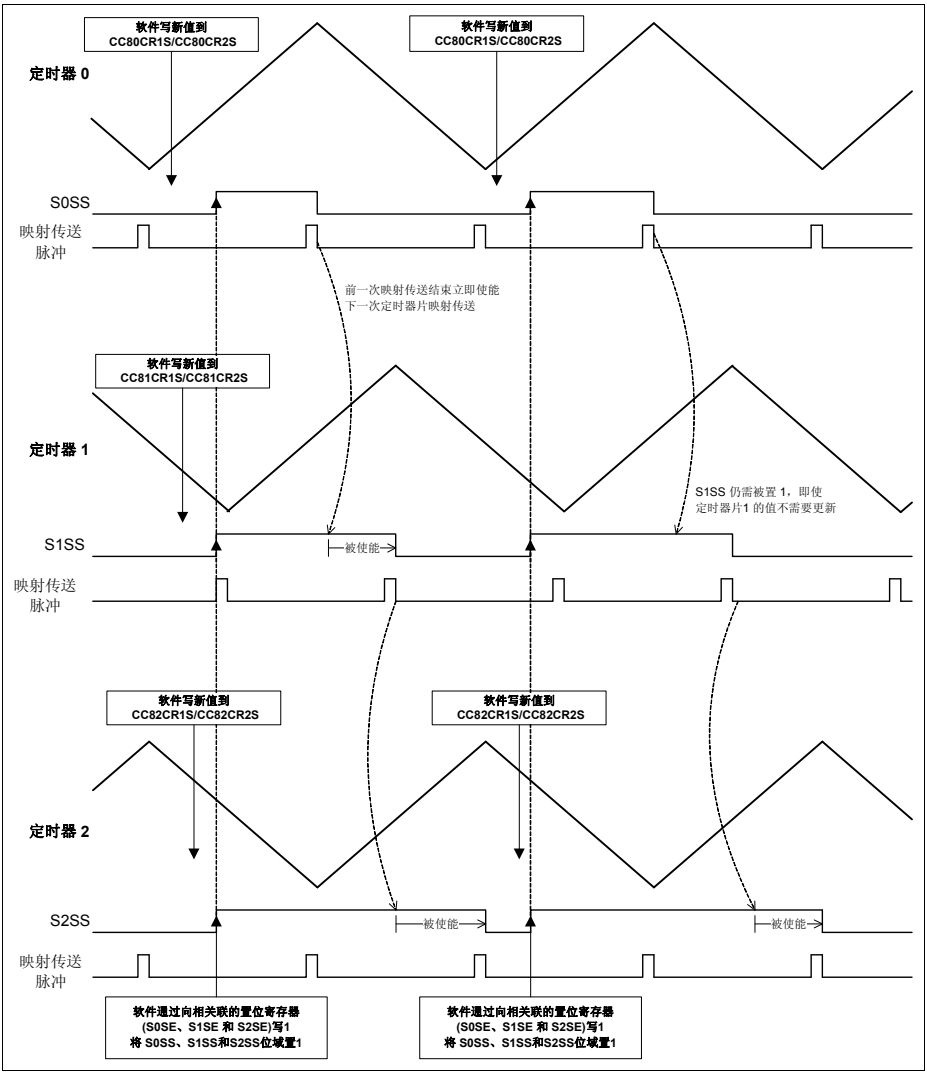


图 20-13　　级联映射传送时序

注：　对于所有要在级联架构中使用的定时器片，其映射传送使能位需要同时被置1。对于所有定时器片，其映射传送使能位也需要被置1，即使有些定时器片的映射值未被更新。

20.2.5.3 边沿对齐模式

边沿对齐模式是默认的计数方式。在该模式，定时器一直递增计数，直到其计数值与周期寄存器 **CC8yPR** 中的编程值相匹配。当检测到周期匹配时，定时器被清为 0000_H 并继续递增计数。

在该模式，每当发生溢出 (周期匹配) 时，周期寄存器和比较寄存器的值都会被已由软件写入到对应映射寄存器中的值更新，如**图 20-14** 所示。

在边沿对齐模式，定时器达到比较寄存器中的编程值后，再过一个时钟周期，比较状态位 (CC8ySTx) 会被置位。定时器达到 0000_H 后，再过一个时钟周期，状态位会被清除。

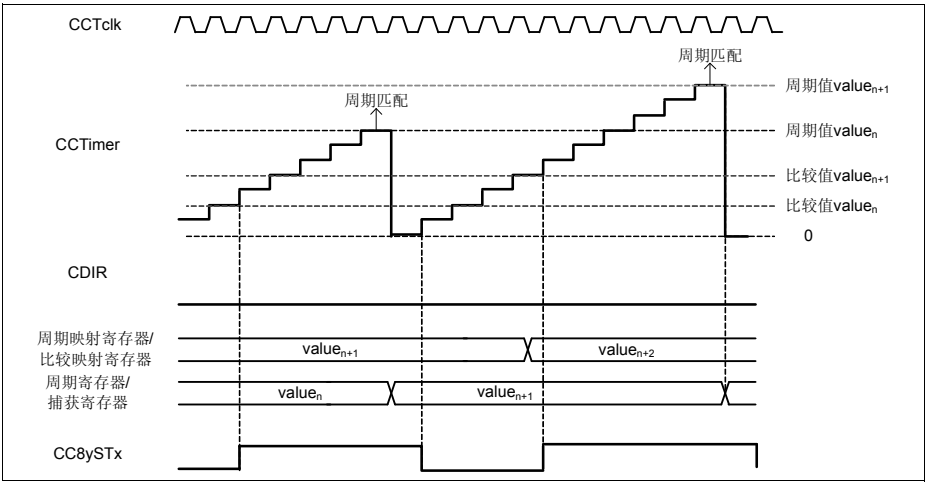


图 20-14　　边沿对齐模式，CC8yTC.TCM = 0_B

20.2.5.4 中心对齐模式

在中心对齐模式，定时器遵循下述规则进行向上或向下计数：

- 当 **CC8yTCST**.CDIR = 0_B 时，计数器向上计数；当 **CC8yTCST**.CDIR = 1_B 时，计数器向下计数。
- 在向下计数期间，当计数值达到 0001_H 时，计数方向会在下一个时钟周期内被设置为向上计数 (**CC8yTCST**.CDIR = 0_B)。
- 在向上计数期间，当检测到周期匹配时，计数方向会在下一个时钟周期内被设置为向下计数 (**CC8yTCST**.CDIR = 1_B)。

当计数器的值等于或大于比较值时，状态位 (CC8ySTx) 一直为 1_B，否则，状态位为 0_B 。

在边沿对齐模式，每个周期执行一次比较寄存器和周期寄存器的映射传送。在中心对齐模式，每个周期执行如下两次映射传送。

- 在向上计数 (**CC8yTCST**.CDIR = 0_B) 期间，当计数器达到周期值后，在下一个时钟周期内执行一次映射传送。

- 在向下计数 (**CC8yTCST**.CDIR = 1_B) 期间，当计数器值达到 0001_H 后，在下一个时钟周期内执行一次映射传送。

注： *位 **CC8yTCST**.CDIR 在 1 匹配或周期匹配后的下一个定时器时钟周期内改变，这意味着在改变计数方向之前，定时器会沿着以前的计数方向继续计数一个时钟周期。*

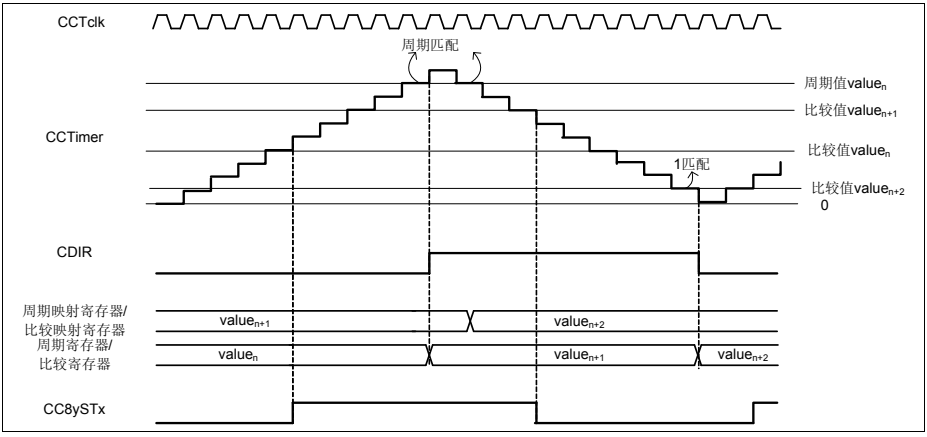


图 20-15　　中心对齐模式，CC8yTC.TCM = 1_B

20.2.5.5　单次模式

在单次模式，定时器会在当前的定时器周期结束后停止计数。该模式可以与中心对齐方式或边沿对齐方式一起使用。

在边沿对齐模式，如图 20-16 所示，定时器在达到周期值后被清为 0000_H，并停止计数。在中心对齐模式，如图 20-17 所示，当定时器向下计数到 0000_H 时，计数周期结束。

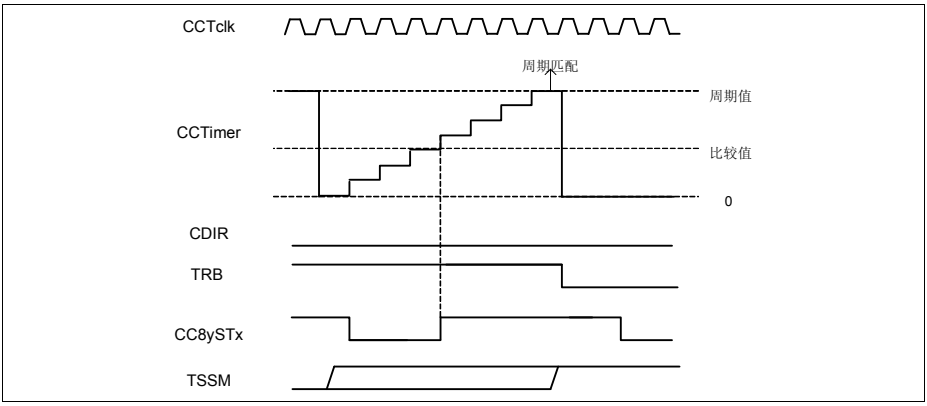


图 20-16　　单次边沿对齐 - CC8yTC.TSSM = 1_B, CC8yTC.TCM = 0_B

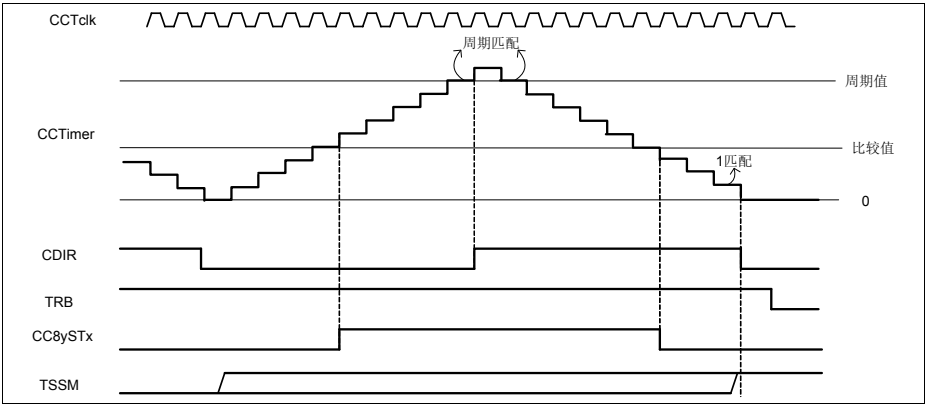


图 20-17　　单次中心对齐 - CC8yTC.TSSM = 1_B, CC8yTC.TCM = 1_B

20.2.6　主动 / 被动规则

可以总结出置位或清除相关联的定时器片状态位的一般规则，而这与定时器计数模式无关。

下列事件会将状态位设置为主动态：

- 向上计数时在比较匹配后的下一个 f_{tclk} 周期
- 向下计数时在 0 匹配后的下一个 f_{tclk} 周期

下列事件会将状态位设置为被动态：

- 向上计数时在 0 匹配 (而不是比较匹配) 后的下一个 f_{tclk} 周期
- 向下计数时在比较匹配后的下一个 f_{tclk} 周期

如果使用外部事件来控制定时器操作，这些规则仍然适用。

状态位的状态只能通过软件或外部状态位覆盖功能来“覆盖”，见 **20.2.8.10 节**。

软件可在任意时刻向 **GCSS**.SySTS 位域写入一个 1_B，这会将特定定时器片的状态位 **GCST**.CC8ySTST 置 1。软件可通过向 **GCSC**.SySTC 位域写入 1_B 将特定的状态位清 0。

20.2.7 比较模式

比较通道方式

每个 CCU8 定时器片有两个比较通道和两个死区时间发生器，即每个通道有一个死区时间发生器，如**图 20-18** 所示。每个比较通道使用状态位 CC8ySTx 的信息来产生两个互补输出。所有的输出，即 CCU8x.OUTy0、CCU8x.OUTy1、CCU8x.OUTy2 和 CCU8x.OUTy3 都有一个专用的被动电平控制位。

每个比较通道都可以独立地工作在边沿对齐或中心对齐模式。这意味着通过使用两个可用的比较通道，可产生两路不同的互补 PWM 信号。这两个通道的 PWM 信号的频率是相同的，但每个通道的占空比可以独立编程。

通过将位域设置为 **CC8yCHC**.ASE = 1_B，可以选择非对称输出方案。在非对称模式，这两个比较通道被组合在一起，能在 CCU8x.OUTy0 和 CCU8x.OUTy1 引脚产生一个互补 PWM 信号。

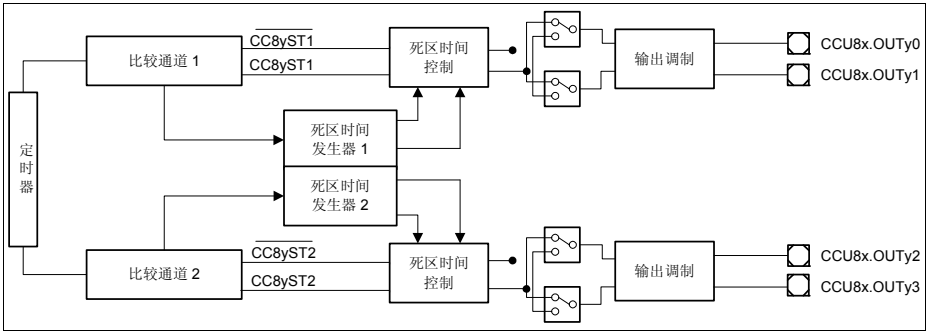


图 20-18 比较通道示意图

死区时间发生器

在大多数情况下，开关的接通和断开时间是不对称的。这种开关行为会在接通时间小于断开时间时导致短路。为了解决这一问题，每个定时器片通道都包含一个死区时间发生器，能延迟输出信号的开关边沿，如**图 20-19** 所示。

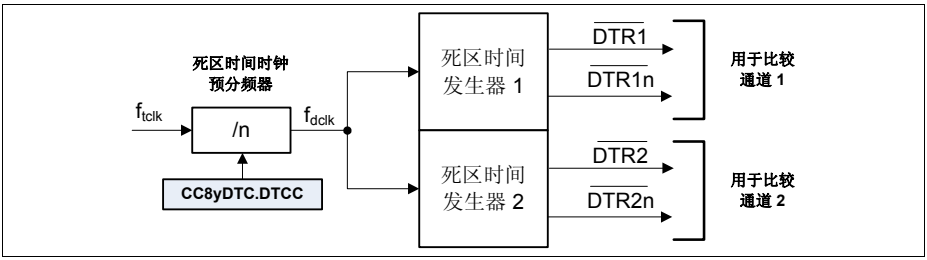


图 20-19　　死区时间原理图

每个死区时间发生器包含一个 8 位计数器，可以为上升和下降时间设置不同的加载值。死区时间发生器包含一个可编程的预分频器，为死区时间计数器提供时钟，以获得大的死区时间插入值，见**表 20-5**。

表 20-5　　死区时间预分频值

CC8yDTC.DTCC[1:0]	**频率**
00_B	f_{tclk}
01_B	$f_{tclk}/2$
10_B	$f_{tclk}/4$
11_B	$f_{tclk}/8$

相关联状态位 CC8ySTx 的任何跳变都会触发对特定死区时间发生器的启动，如**图 20-20** 所示。

当检测到有对 CC8ySTx 位的置位 (CC8ySTx 从 0_B 变为 1_B) 操作时，死区时间计数器被加载为 **CC8yDC1R**.DT1R 或 **CC8yDC2R**.DT2R 中保存的值 (取决于所寻址的通道)。

当检测到有对 CC8ySTx 位的清零 (CC8ySTx 从 1_B 变为 0_B) 操作时，死区时间计数器被加载为 **CC8yDC1R**.DT1F 或 **CC8yDC2R**.DT2F 中保存的值 (取决于所寻址的通道)。

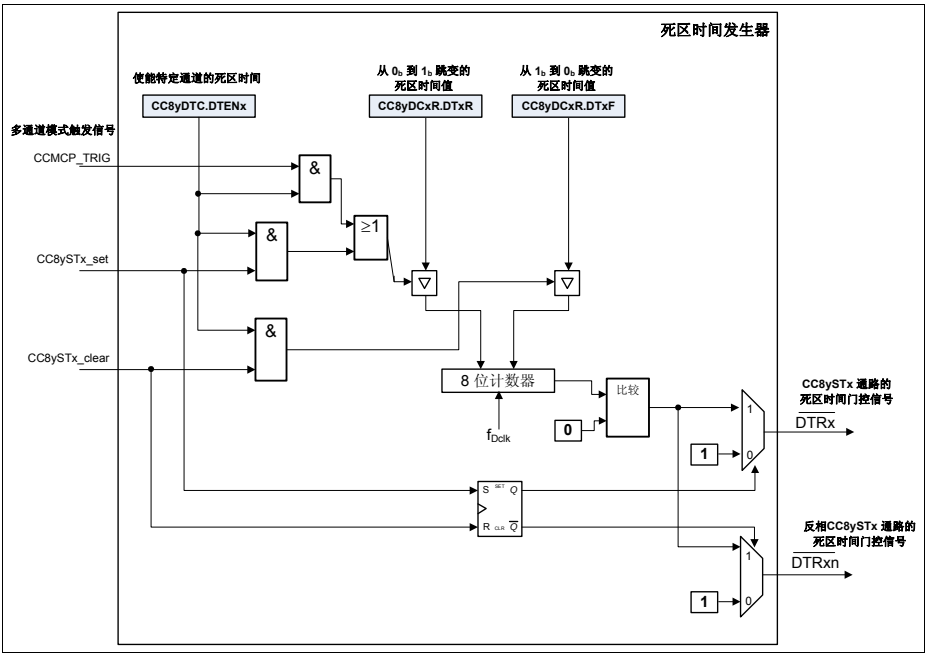


图 20-20　　死区时间发生器原理图

每个死区时间发生器输出两路信号，用于控制两个互补的输出 (CC8ySTx 和反相的 CC8ySTx)。分离这两个控制信号的目的是为了可以灵活地使能 / 禁止每个比较通道的内部电路，如图 20-21 所示。这意味着可以使能一个比较通道的死区时间发生器，但可以放弃对一个输出的死区时间插入。

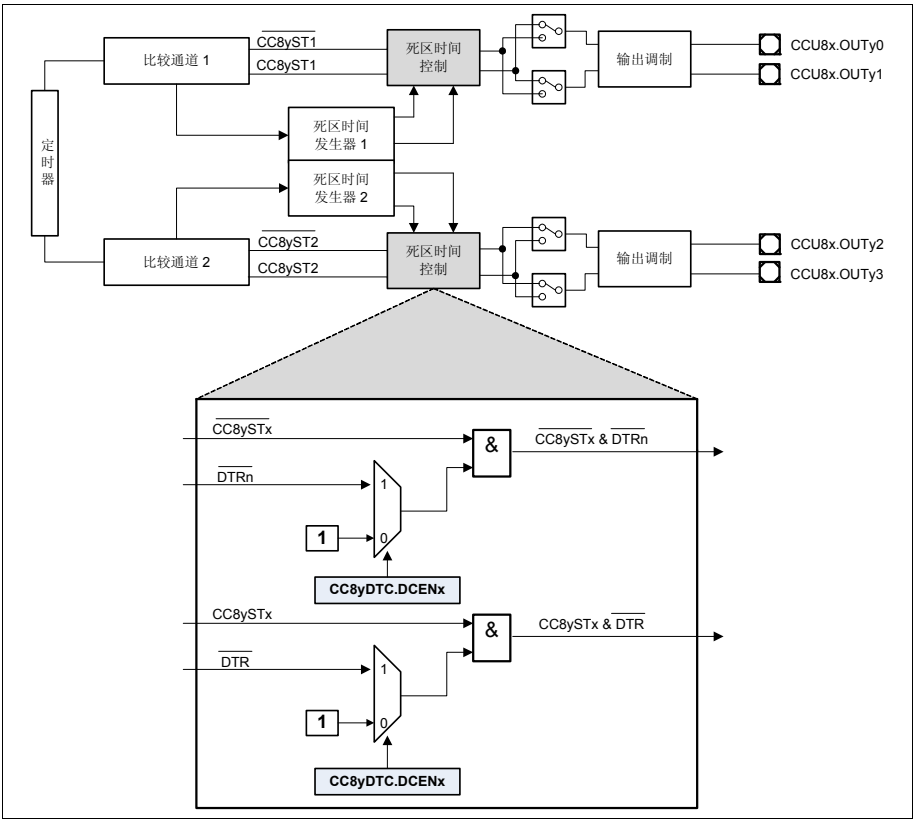


图 20-21　　死区时间控制单元

当使用多通道模式时，即 **CC8yTC**.MCMEy = 1_B，可能会出现所产生的 PWM 信号具有 100% 占空比的情况。这意味着相应的状态位总是处于置位状态，由多通道序列控制输出调制。在这种情况下，即使特定状态位没有发生跳变，输出端也可以发生从非主动到主动状态的跳变，从而造成因没有死区时间插入而导致的开关短路。

为了克服这种可能的开关短路，一个来自多通道控制逻辑的触发信号 CCMP_TRIG（见 **图 20-20**）被馈送给死区时间发生器。**图 20-22** 给出了产生 CCMP_TRIG 的电路，其中信号 CCMCMx0 和 CCMCMx1 代表一个特定通道的多通道序列采样信号。

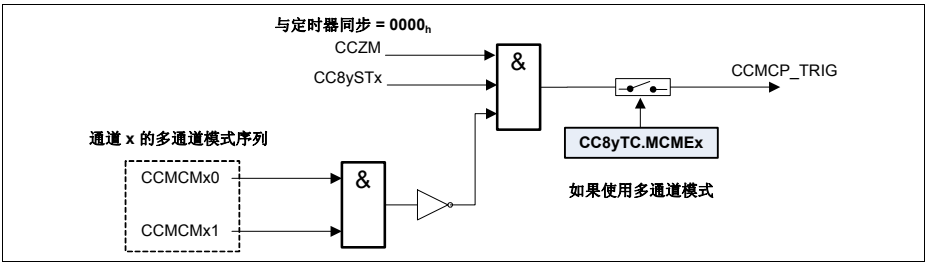


图 20-22　　由多通道序列产生的死区时间触发信号

20.2.7.1　边沿对齐的比较模式

标准边沿对齐模式

当定时器片被编程为边沿对齐模式时，两个通道可以互相独立工作，这意味着两个比较值可以被编程为不同的值（产生不同的占空比）。在这种情况下，每个通道可输出一对 PWM 信号，用于控制一个高边和一个低边开关，见图 20-23。

在该模式，每个通道的上升和下降跳变的死区时间分别由 CC8yDC1R.DT1R、CC8yDC2R.DT2R 和 CC8yDC1R.DT1F、CC8yDC2R.DT2F 位域控制。

图 20-24 示出了在两个通道的比较值不相同的情况下一个特定定时器片的时序图。

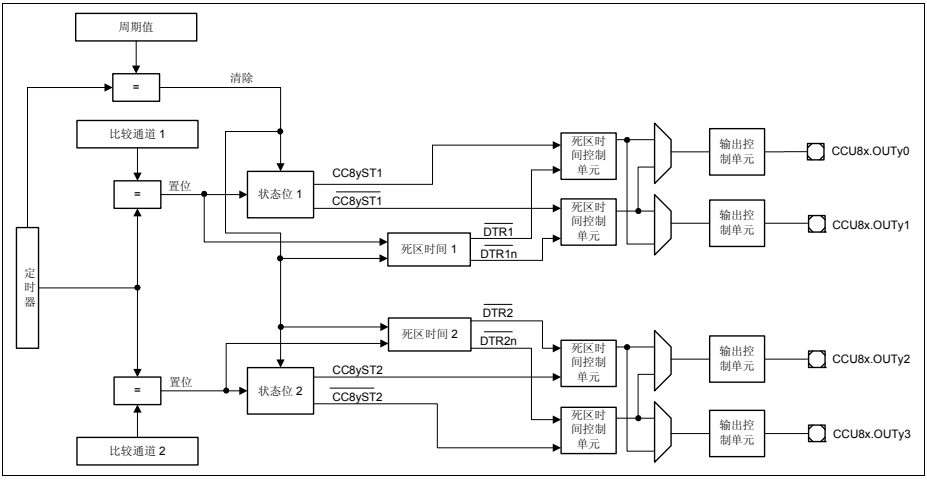


图 20-23　　两个独立通道情况下的边沿对齐模式

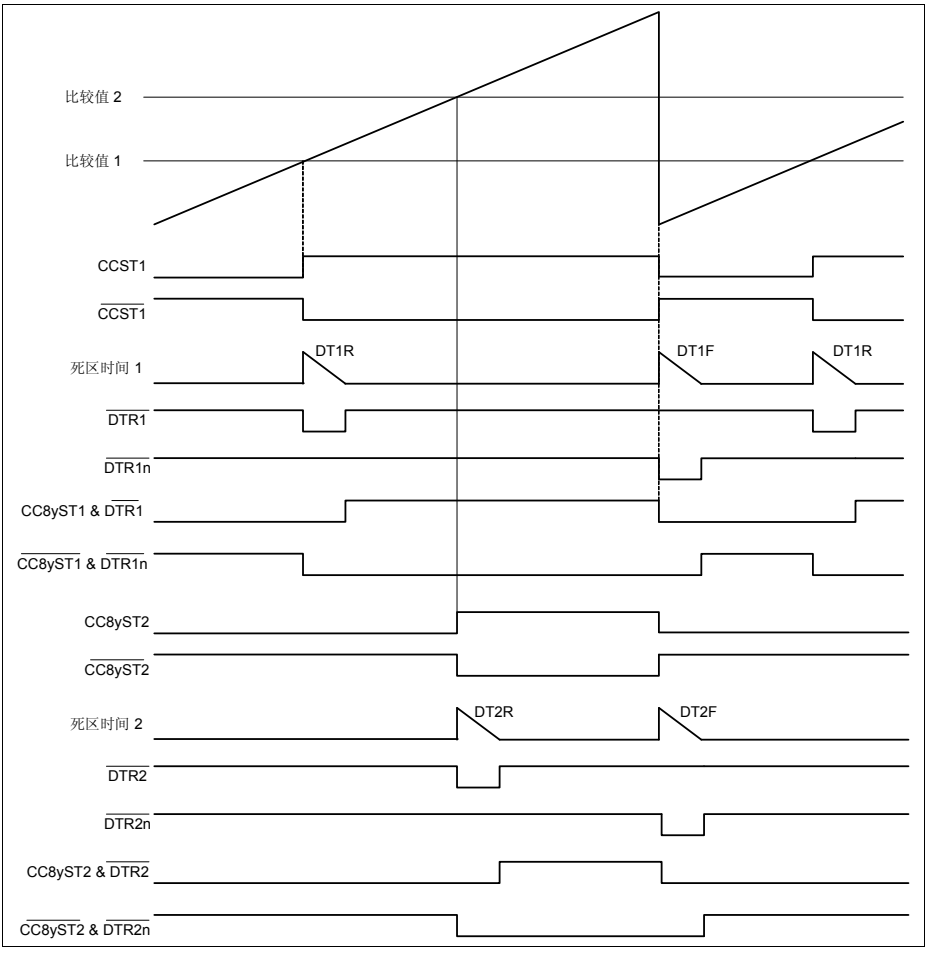


图 20-24　　边沿对齐模式 - 带死区时间的 4 个输出

非对称边沿对齐模式

还有一种使用两个通道的可能方式，即将两个通道组合起来产生一个非对称的 PWM 输出。通过设置位域 **CC8yCHC**.ASE = 1_B 来选择该模式。

在该模式，比较通道 2 被禁止，因此与这一通路连接的输出总是处于被动状态。

当发生与比较值 1(域 **CC8yCR1**.CR1) 的比较匹配时，比较通道 1 的状态位被置 1，而当发生与比较值 2(域 **CC8yCR2**.CR2) 的比较匹配时，该状态位被清零，见**图 20-25**。

当 **CC8yCR2**.CR2 被编程为小于 **CC8yCR1**.CR1 中的值时， CCST1 位总是为 0_B。

上升和下降跳变的死区时间值分别由位域**CC8yDC1R**.DT1R和**CC8yDC1R**.DT1F控制。**图 20-26** 和**图 20-27** 示出了非对称机制有效时的边沿对齐模式时序图。

注： 当使用一个外部信号来控制计数方向时，不能使能非对称模式。

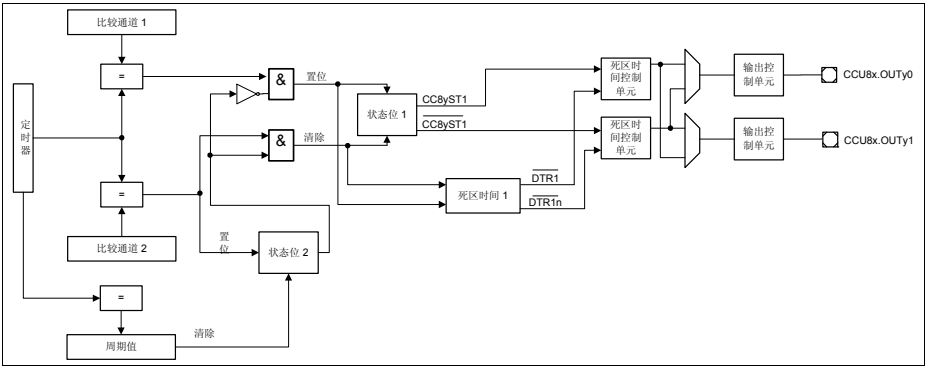


图 20-25 两个通道组合使用时的边沿对齐模式

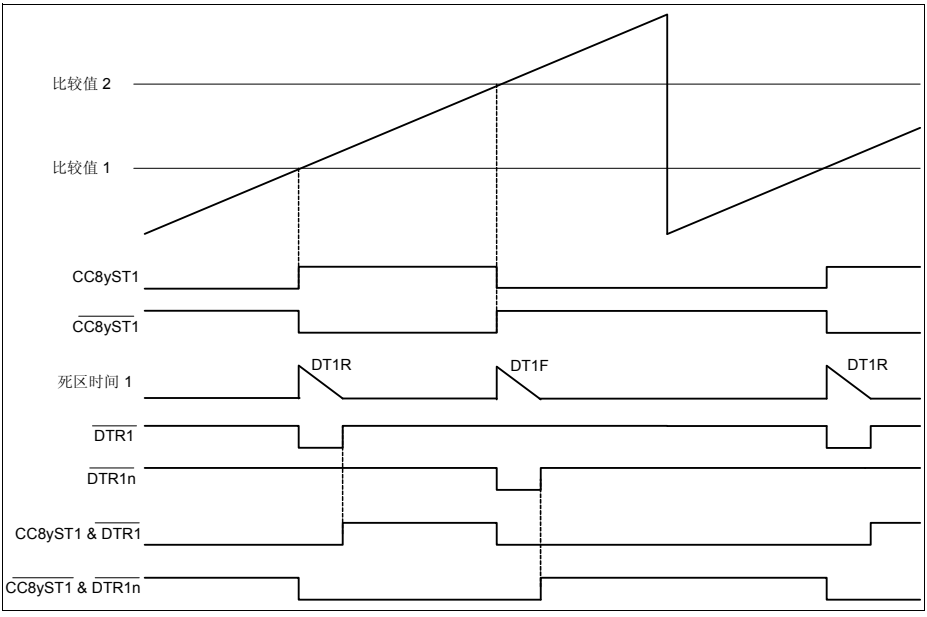


图 20-26 边沿对齐模式 - 非对称 PWM 时序， CC8yCR1.CR1 < CC8yCR2.CR2

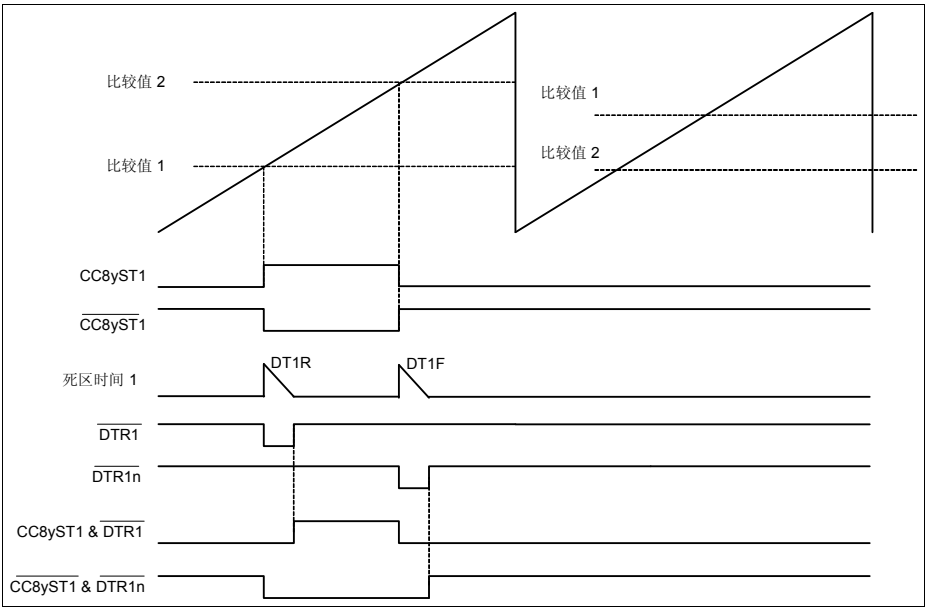


图 20-27　　边沿对齐模式 - 非对称 PWM 时序， CC8yCR1.CR1 > CC8yCR2.CR2

20.2.7.2　中心对齐比较模式

标准中心对齐模式

在中心对齐模式，类似于边沿对齐模式，可以使用两个互相独立的通道。在该模式，每个通道都能产生一对具有不同占空比值的 PWM 互补信号，通道 1 的占空比由 **CC8yCR1** 控制，通道 2 的占空比由 **CC8yCR2** 控制。

对于死区时间插入，每个通道的上升和下降跳变都有一对可编程的值：**CC8yDC1R**.DT1R 和 **CC8yDC1R**.DT1F 用于通道 1，**CC8yDC2R**.DT2R 和 **CC8yDC2R**.DT2F 用于通道 2。

中心对齐与边沿对齐模式的主要差别与状态位的置位 / 清零逻辑直接相关，参见 **20.2.5 节**。

图 20-28 示出了该工作模式下两个通道的原理图，**图 20-29** 示出了一个特定通道的时序图。

注：　当使用一个外部信号来控制计数方向时，计数方式总是边沿对齐。

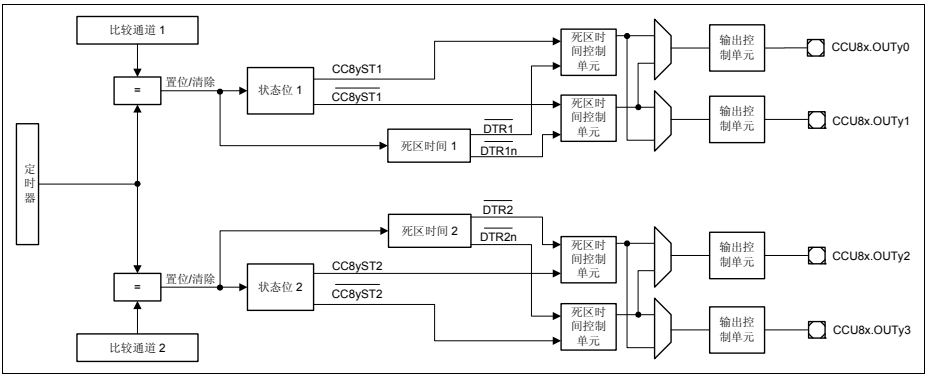


图 20-28　　两个独立通道情况下的中心对齐模式

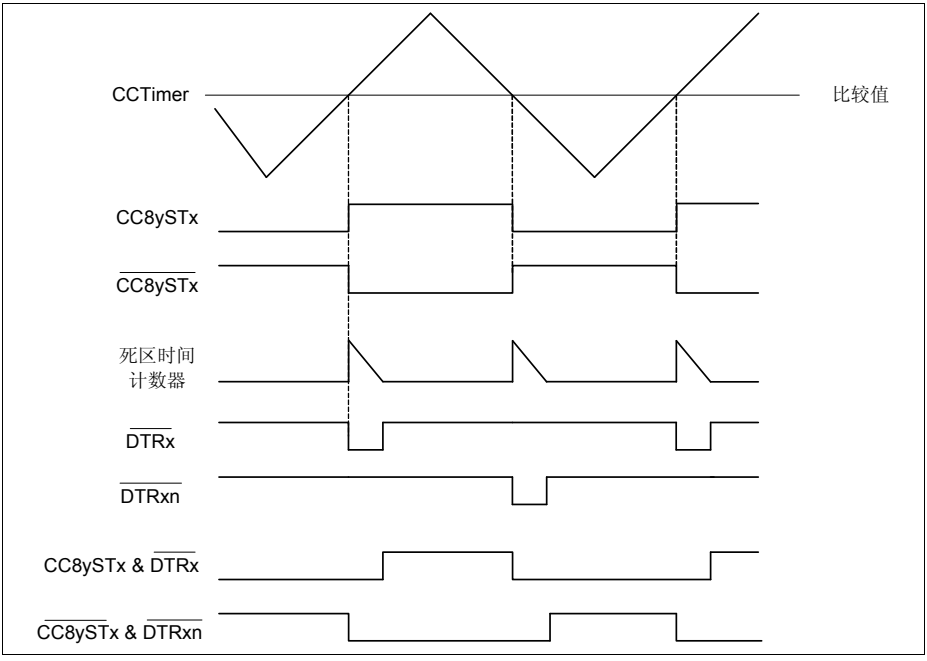


图 20-29　　中心对齐 - 带死区时间的独立通道

非对称中心对齐模式

中心对齐情况下的非对称模式通过将位域 **CC8yCHC**.ASE 设置为 1_B 来使能。
在该模式，类似于边沿对齐模式，与比较通道 2 连接的输出被置为其被动电平。

当向上计数且通道 1 发生一次比较匹配时，状态位 CC8yST1 被置 1，当向下计数且通道 2 发生一次比较匹配时，该状态位被清零，见**图 20-30**。

上升和下降跳变的死区时间分别由位域 **CC8yDC1R**.DT1R 和 **CC8yDC1R**.DT1F 控制。

图 20-31 示出了非对称模式的时序图。注意：即使是在非对称模式，每个输出的死区时间也可以被独立禁止。

注： 当使用一个外部信号来控制计数方向时，不能使能非对称模式。

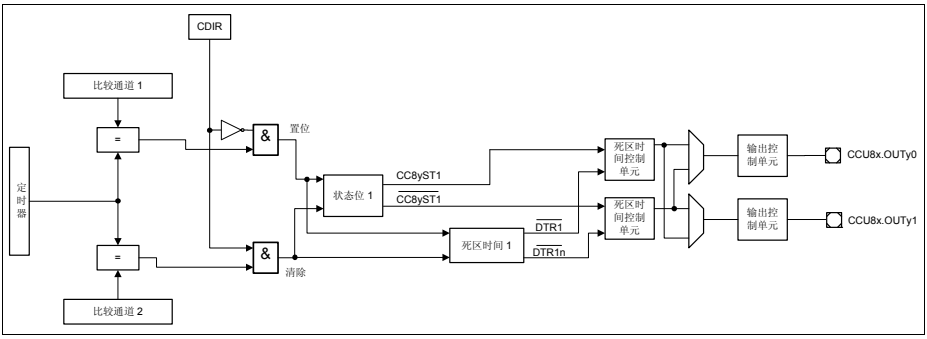


图 20-30 中心对齐的非对称模式原理图

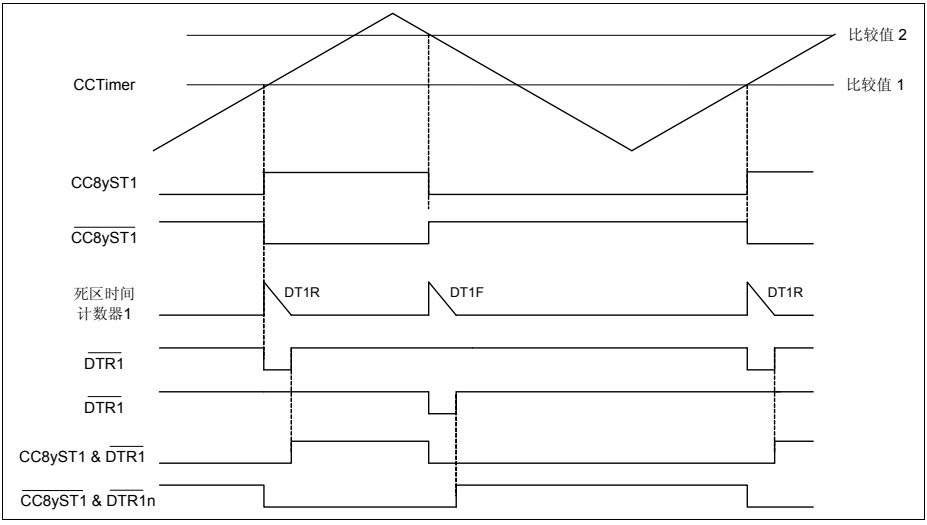


图 20-31 带死区时间的非对称中心对齐模式

20.2.8 外部事件控制

每个 CCU8 定时器片都可以最多使用三个不同的输入事件，见 **20.2.2 节**。这三个事件可以被映射为定时器片功能 (关于所有可用功能的描述，请参见 **20.2.3 节**)。

这些事件可以被映射到任何一个 CCU8x.INy[P...A] 输入，既没有限制一个事件不能用于执行多个功能，也没有限制一个输入不能被映射为多个事件 (例如，输入信号 X 用上升沿触发事件 0，而用下降沿触发事件 1)。

20.2.8.1 外部启动 / 停止

为了选择一个外部启动功能，应将一个事件 (输入选择器输出的) 映射到一个特定输入信号，这可以通过在 **CC8yINS**.EVxIS 域设置所需要的值并在 **CC8yINS**.EVxEM 域指定信号的有效边沿来完成。

然后，应通过给 **CC8yCMC**.STRTS (用于启动) 或 **CC8yTC**.ENDM (用于停止) 设定合适的值将该事件映射为启动或停止功能。

同样的过程适用于停止功能。

注意，启动和停止功能都是边沿有效而非电平有效。因此，主动态或被动态配置只能通过 **CC8yINS**.EVxEM 来设定。

在默认情况下，外部停止信号只会清除运行位 (**CC8yTCST**.TRB)，而启动功能则恰恰相反。不过，可以为外部启动和停止信号选择一个扩展的功能子集。该功能子集由位域 **CC8yTC**.ENDM (用于停止) 和 **CC8yTC**.STRM (用于启动) 控制。

对于启动子集 (**CC8yTC**.STRM)：

- 置位运行位 / 启动定时器 (恢复运行)
- 清除定时器，置位运行位 / 启动定时器 (清空并启动)

对于停止子集 (**CC8yTC**.ENDM)：

- 清除运行位 / 停止定时器 (停止)
- 清除定时器 (清空)
- 清除定时器，清除运行位 / 停止定时器 (清空并停止)

如果结合外部启动 / 停止 (也配置为启动 / 仅配置为清空) 功能，并使用一个外部向上 / 向下信号，则在清空操作过程中，若当前计数方向为向上计数，定时器将被设置为 0000_H，若计数方向为向下计数，则用周期寄存器中的值来设置定时器。

图 20-32 到**图 20-35** 展示了实现所有上述子集中的启动 / 停止功能的两个信号的使用。外部信号 (1) 用作高电平有效的启动信号，而外部信号 (2) 用作高电平有效的停止信号。

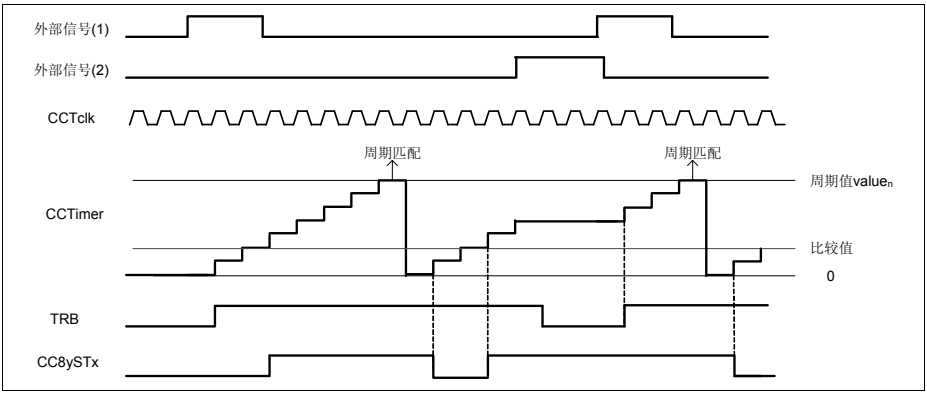


图 20-32　启动 (用作启动)/ 停止 (用作停止) - CC8yTC.STRM = 0_B, CC8yTC.ENDM = 00_B

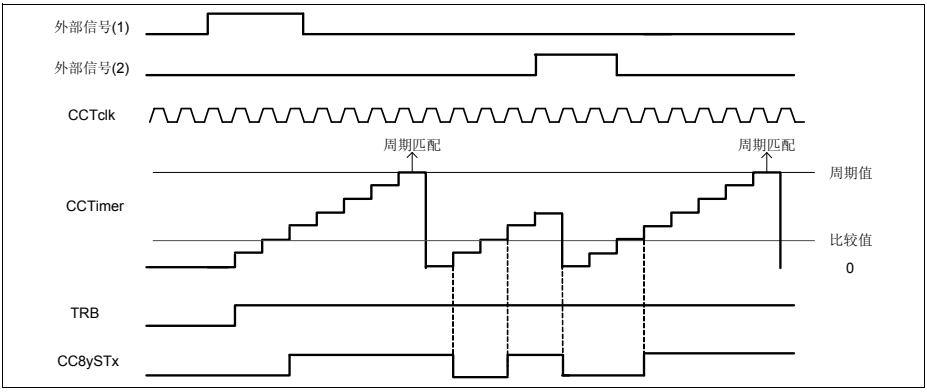


图 20-33　启动 (用作启动)/ 停止 (用作清空) - CC8yTC.STRM = 0_B, CC8yTC.ENDM = 01_B

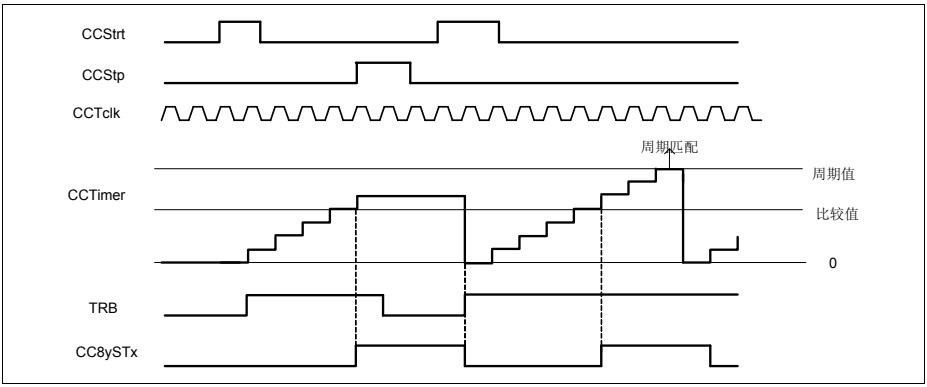


图 20-34　启动 (用作清空和启动)/ 停止 (用作停止) - CC8yTC.STRM = 1_B, CC8yTC.ENDM = 00_B

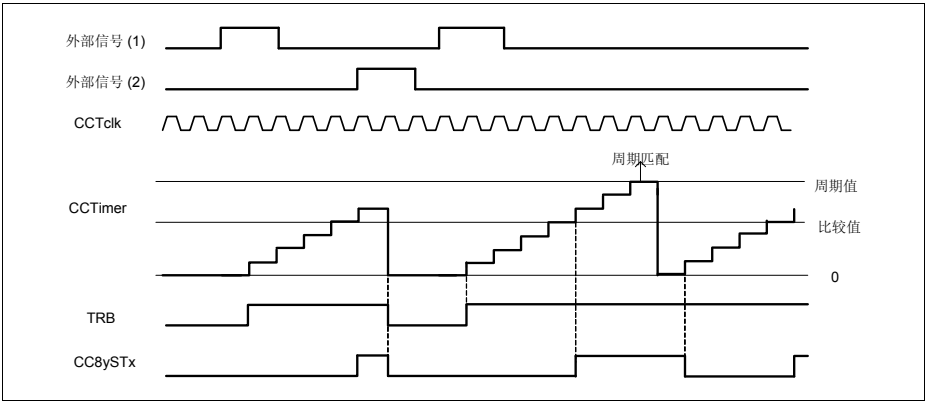


图 20-35　启动 (用作启动)/ 停止 (用作清空和停止) - CC8yTC.STRM = 0_B, CC8yTC.ENDM = 10_B

20.2.8.2　外部计数方向信号

可以选择一个外部输入作为向上 / 向下计数控制信号。

要选择一个外部向上 / 向下计数控制信号，应将一个事件 (输入选择器的输出) 映射为一个特定的输入信号，这可以通过在 CC8yINS.EVxIS 域设置所需要的值并在 CC8yINS.EVxLM 域指定信号的有效电平来完成。然后，应通过将 CC8yCMC.UDS 设置为合适的值将该事件映射为向上 / 向下功能。

注意：向上 / 向下功能为电平有效，因此，主动态 / 被动态配置只能通过 CC8yINS.EVxLM 来设定。

在定时器值等于或大于保存在比较寄存器中的值时，定时器片的状态位 (CCSTx) 会一直被置位，详见 **20.2.6 节**。

周期寄存器和比较寄存器的值在下列时刻更新：

- 在向上计数 (**CC8yTCST**.CDIR = 0_B) 期间，发生周期匹配后的下一个时钟周期
- 在向下计数 (**CC8yTCST**.CDIR = 1_B) 期间，发生 1 匹配后的下一个时钟周期

随着译码事件的变化，**CC8yTCST**.CDIR 寄存器的值会作相应的更新。向上 / 向下方向总是被理解为：当向下计数时 **CC8yTCST**.CDIR = 1_B，当向上计数时 **CC8yTCST**.CDIR = 0_B。使用一个外部信号来执行向上 / 向下计数功能并将事件配置为高电平有效，意味着该信号为高电平时定时器向上计数，该信号为低电平时定时器向下计数。

图 20-36 展示了用于控制定时器计数方向的一个外部信号。该信号被设定为高电平有效，这意味着该信号为高电平时定时器向下计数，而该信号为低电平时定时器向上计数。

*注：　对于一个低电平时向上计数而高电平时向下计数的方向控制信号，用户需要设置 **CC8yINS**.EVxLM = 0_B。当操作相反时，用户应设置 **CC8yINS**.EVxLM = 1_B。*

注：　使用一个外部计数方向控制信号，会将定时器片设置为边沿对齐模式。

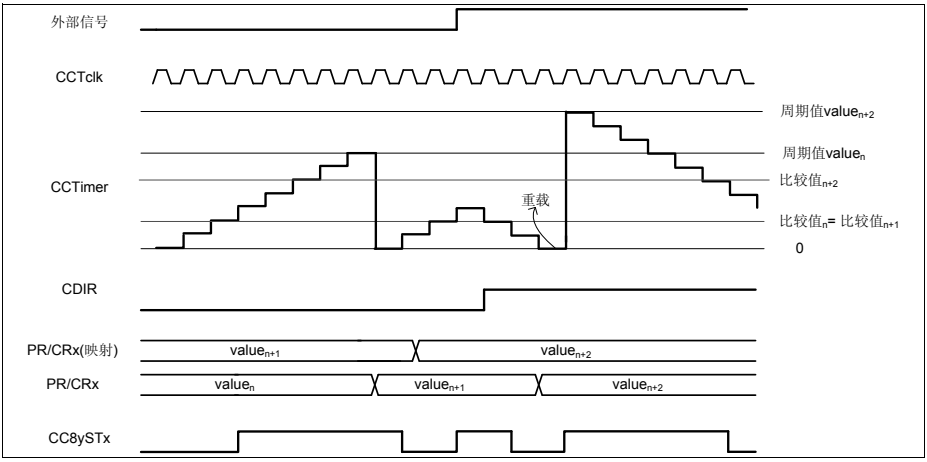


图 20-36　外部计数方向信号

20.2.8.3　外部门控信号

为了进行脉冲测量，用户可以选择一个输入信号用作计数门控。

要选择一个外部门控信号，应将一个事件 (输入选择器的输出) 映射为一个特定的输入信号，这可以通过在 **CC8yINS**.EVxIS 寄存器中设置所需要的值并在 **CC8yINS**.EVxLM 寄存器中指定信号的有效电平来实现。然后，应通过将 **CC8yCMC**.GATES 设置为合适的值，将该事件映射为门控功能。

注意，门控功能为电平有效，因此主动态/被动态配置只能通过**CC8yINS**.EVxLM来设定。

在一个外部门控信号有效期间，当计数器达到比较值后，状态位继续有效，而当计数器达到 0000_H 时，状态位变为无效。应当注意的是，计数器继续使用周期寄存器来识别回绕条件。**图 20-37** 展示了用于门控片计数器的外部信号的使用。该信号被设定为低电平有效，这意味着当外部信号值为零时，计数器门控功能有效。

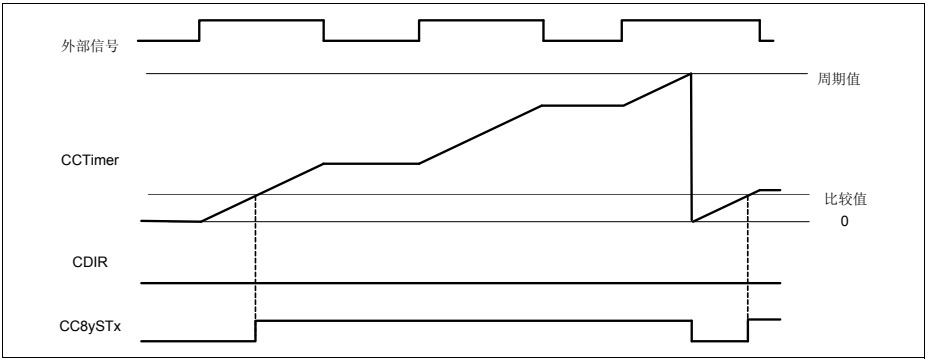


图 20-37　　外部门控信号

20.2.8.4　外部计数信号

也可以选择一个外部信号作为计数事件。

要选择一个外部计数信号，应将一个事件 (输入选择器的输出) 映射为一个特定的输入信号，这可以通过在 **CC8yINS**.EVxIS 寄存器中设置所需要的值并在 **CC8yINS**.EVxEM 寄存器中指定信号的有效边沿来实现。然后，应通过将 **CC8yCMC**.CNTS 设置为合适的值将该事件映射为计数功能。

注意，计数功能为边沿有效，因此主动态/被动态配置只能通过**CC8yINS**.EVxEM来设定。

可以选择只在上升沿、下降沿或两个边沿执行计数。在**图 20-38** 中，外部信号的上升沿和下降沿都被选择为计数事件。回绕条件仍然采用与周期寄存器进行比较。

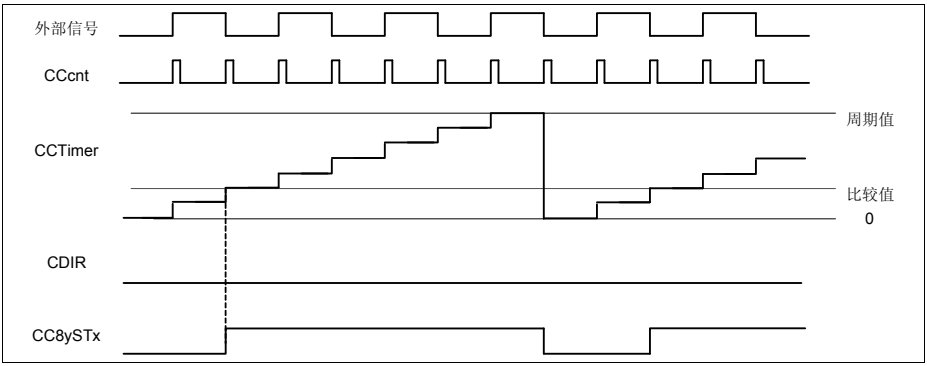


图 20-38　　外部计数信号

20.2.8.5　外部加载信号

CCU8 的每个定时器片还有一个功能，即允许用户选择一个外部信号作为定时器值重新加载（重载）的触发信号，重载值可以是比较寄存器的当前值 (如果 **CC8yTCST**.CDIR = 0_B)，也可以是周期寄存器的当前值 (如果 **CC8yTCST**.CDIR = 1_B)。

定时器可以被加载为比较通道 1 或比较通道 2 的值，这取决于 **CC8yTC**.TLS 域中的设置值，见**图 20-39**。

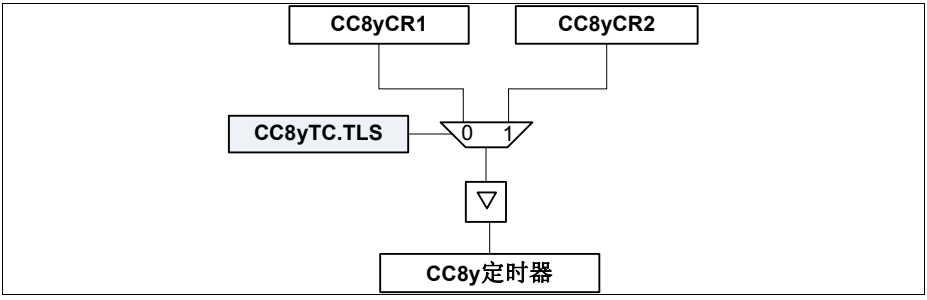


图 20-39　　定时器加载选择

要选择一个外部加载信号，应将一个事件 (输入选择器的输出) 映射为一个特定的输入信号，这可以通过在 **CC8yINS**.EVxIS 寄存器中设置所需要的值并在 **CC8yINS**.EVxEM 寄存器中指定信号的有效边沿来完成。然后，应通过将 **CC8yCMC**.LDS 设置为合适的值将该事件映射为加载功能。

注意，加载功能为边沿有效，因此主动态/被动态配置只能通过**CC8yINS**.EVxEM来设定。

在**图 20-40** 中，外部信号 (1) 被用作加载触发信号，上升沿有效。每当检测到外部信号 (1) 的一个上升沿时，比较寄存器的当前值就会被加载到定时器。如果使用一个外部信号来控制计数方向，向上或向下，也可以用周期寄存器中的设定值加载定时器。外部信号

(2) 代表计数方向控制 (高电平有效)。如果在检测到加载触发信号的那一刻控制计数方向的信号是向下计数，则定时器中的设置值为周期值。

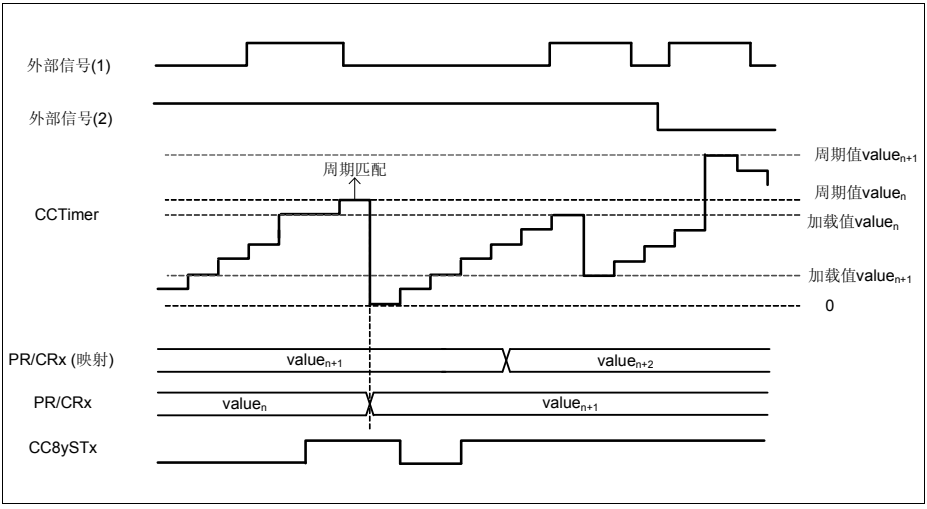


图 20-40　　外部加载信号

20.2.8.6　外部捕获信号

当选择一个外部信号用作捕获触发信号时 (若 **CC8yCMC**.CAP0S 或 **CC8yCMC**.CAP1S 不等于 0_H)，特定的定时器片被自动设定为捕获模式。

在捕获模式，用户最多可以有 4 个捕获寄存器，如**图 20-43** 所示：捕获寄存器 0 (**CC8yC0V**)、捕获寄存器 1 (**CC8yC1V**)、捕获寄存器 2 (**CC8yC2V**) 和捕获寄存器 3 (**CC8yC3V**)。

这些寄存器在比较模式和捕获模式之间是共享的，这就要求：

- 如果 **CC8yC0V** 和 **CC8yC1V** 用于捕获，则比较寄存器 **CC8yCR1** 和 **CC8yCR1S** 不可用 (比较通道 1 不可用)
- 如果 **CC8yC2V** 和 **CC8yC3V** 用于捕获，则比较寄存器 **CC8yCR2** 和 **CC8yCR2S** 不可用 (比较通道 2 不可用)

要选择一个外部捕获信号，应将一个事件 (输入选择器的输出) 映射为一个特定的输入信号，这可以通过在 **CC8yINS**.EVxIS 寄存器中设置所需要的值并在 **CC8yINS**.EVxEM 寄存器中指定信号的有效边沿来完成。

然后，应通过给 **CC8yCMC**.CAP0S/**CC8yCMC**.CAP1S 设置合适的值将该事件映射为捕获功能。

注意，捕获功能为边沿有效，因此主动态/被动态配置只能通过**CC8yINS**.EVxEM来设定。

用户可以选择下述捕获方案：

- **CC8yC0V**/**CC8yC1V** 和 **CC8yC2V**/**CC8yC3V** 使用不同的捕获事件
- **CC8yC0V**/**CC8yC1V** 和 **CC8yC2V**/**CC8yC3V** 使用同一个捕获事件和同一个捕获边沿。对于这种捕获方案，只有 CCcapt1 的功能需要编程设定。要启用该方案，需将 **CC8yTC**.SCE 域设置为 1_B。

不同的捕获事件 (SCE = 0_B)

每当捕获触发信号 1(CCcapt1) 发生时，定时器的当前值被捕获到捕获寄存器 3，而先前保存在该寄存器的值被传送到捕获寄存器 2。

每当捕获触发信号 0(CCcapt0) 发生时，定时器的当前值被捕获到捕获寄存器 1，而先前保存在该寄存器的值被传送到捕获寄存器 0。

每次发生捕获到一个寄存器的操作时，该寄存器的满标志被置位。当软件读取该捕获寄存器的值 (通过读取特定的捕获寄存器或读取扩展的捕获读出值，见 **20.2.8.7 节**) 时，该标志被硬件自动清除。

将一个新值捕获到特定的捕获寄存器，可以通过满标志描述如下：

$$CC8yC1V_{capt} = \text{NOT}(CC8yC1V_{full\_flag} \text{ AND } CC8yC0V_{full\_flag}) \quad (20.4)$$

$$CC8yC0V_{capt} = CC8yC1V_{full\_flag} \text{ AND NOT}(CC8yC0V_{full\_flag}) \quad (20.5)$$

也可以通过设定 **CC8yTC**.CCS = 1_B 来禁止满标志的影响。这样一来，不论捕获的值是否被读取，都可以进行连续捕获。

在**图 20-41** 中，一个外部信号被选择作为一个捕获事件，用于将定时器的值捕获到 **CC8yC0V**/**CC8yC1V** 寄存器。在捕获模式期间，当检测到捕获触发信号时，状态位 CC8ySTx 变为有效，当计数器值为 0000_H 时，状态位变为无效。

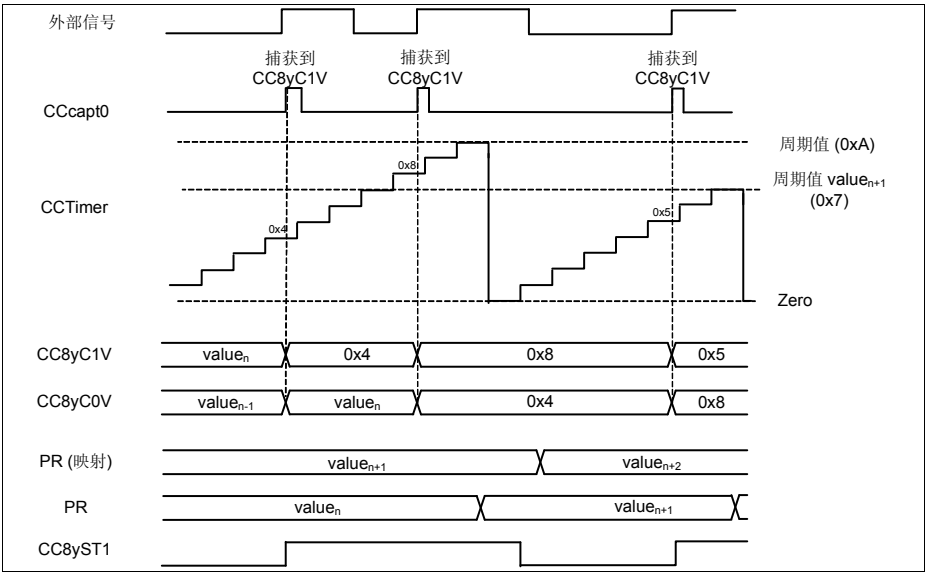


图 20-41　外部捕获 - CC8yCMC.CAP0S != 00_B, CC8yCMC.CAP1S = 00_B

在图 20-42 中，有两个不同的信号被用作将定时器值捕获到 CC8yC0V/CC8yC1V/CC8yC2V/CC8yC3V 寄存器的触发源。外部信号 (1) 被选作通道 1 的捕获触发源，上升沿有效。外部信号 (2) 被选作通道 2 的捕获触发源，但是与外部信号 (1) 相反，有效沿被设置为下降沿。

关于捕获模式使用的详细描述，请参见 20.2.14.4 节。

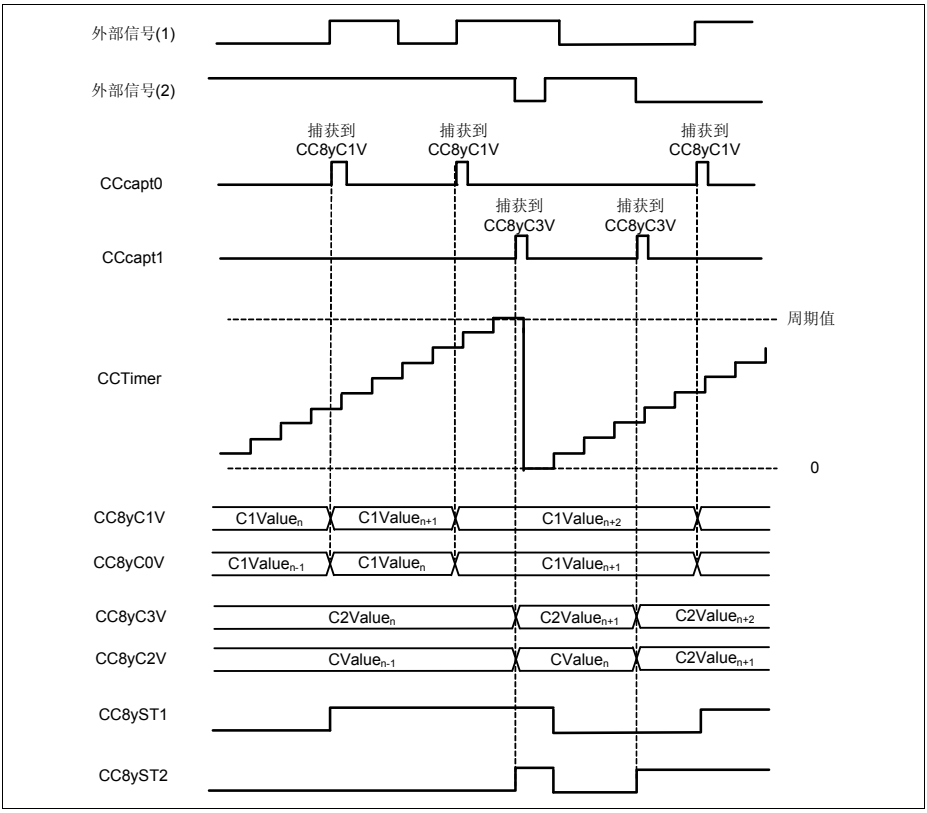


图 20-42　　外部捕获 - CC8yCMC.CAP0S != 00_B, CC8yCMC.CAP1S != 00_B

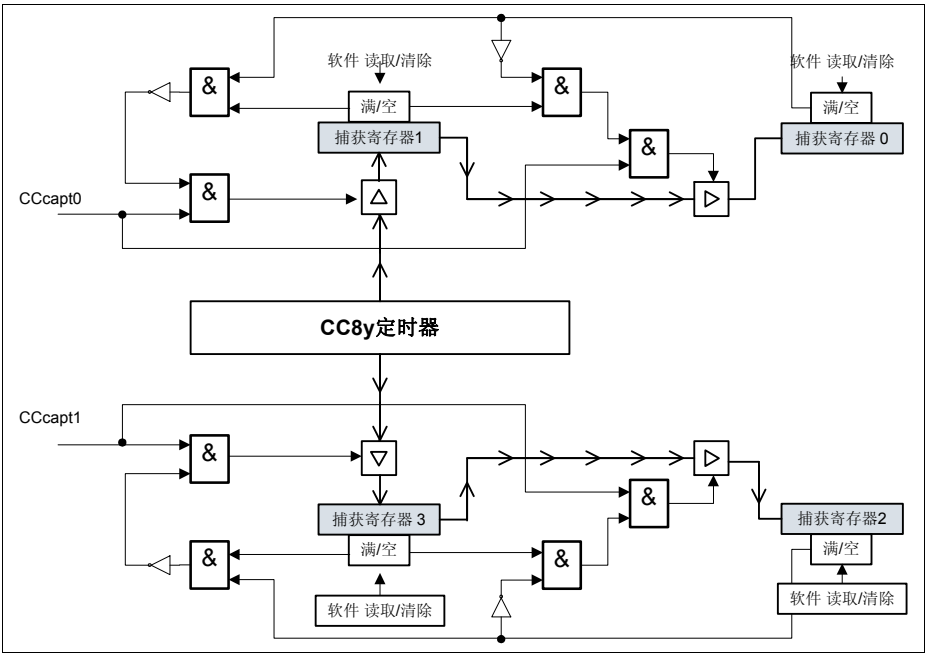


图 20-43　　定时器片捕获逻辑

相同的捕获事件 (SCE = 1_B)

设置位域 **CC8yTC**.SCE = 1_B，可以使 4 个捕获寄存器连接到同一捕获事件，如**图 20-45** 所示。 CCcapt1 控制捕获功能。

捕获逻辑采用如**图 20-43** 所示的相同结构，但是被扩展成一个四寄存器链，如**图 20-44** 所示。四寄存器链采用相同的满标志锁定规则 (也可以通过设置 **CC8yTC**.CCS = 1_B 来禁止)：

$$CC8yC3V_{capt} = \text{NOT}(CC8yC3V_{full\_flag} \text{ AND } CC8yC2V_{full\_flag} \text{ AND } CC8yC2V_{full\_flag} \text{ AND } CC8yC1V_{full\_flag}) \quad (20.6)$$

$$CC8yC2V_{capt} = CC8yC3V_{full\_flag} \text{ AND NOT}(CC8yC2V_{full\_flag} \text{ AND } CC8yC1V_{full\_flag} \text{ AND } CC8yC0V_{full\_flag}) \quad (20.7)$$

$$CC8yC1V_{capt} = CC8yC2V_{full\_flag} \text{ AND NOT}(CC8yC1V_{full\_flag} \text{ AND } CC8yC0V_{full\_flag}) \quad (20.8)$$

$$CC8yC0V_{capt} = CC8yC1V_{full\_flag} \text{ AND NOT}(CC8yC0V_{full\_flag}) \quad (20.9)$$

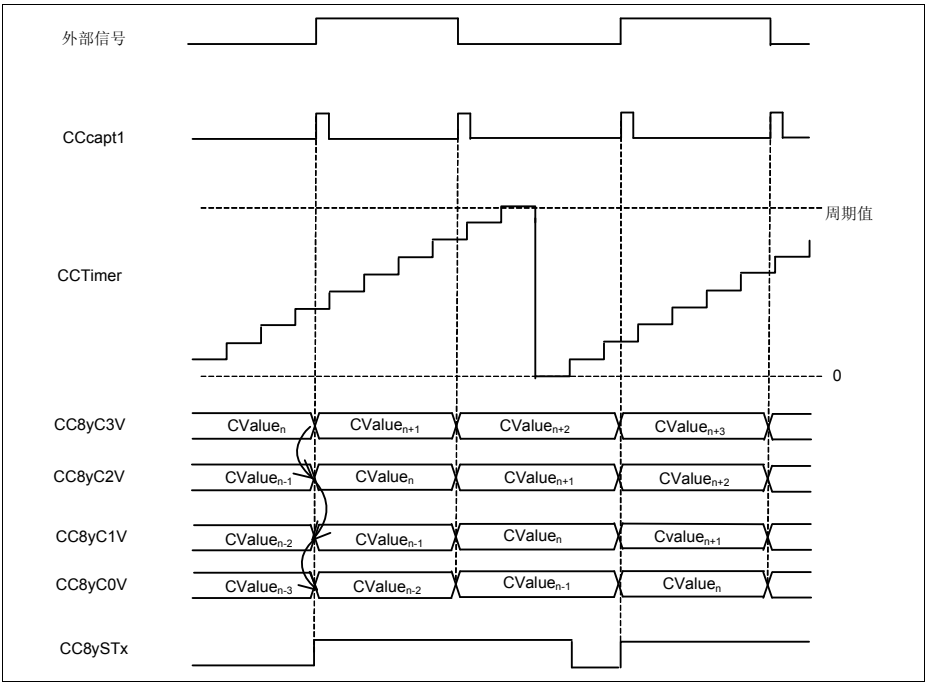


图 20-44　　外部捕获 - CC8yTC.SCE = 1_B

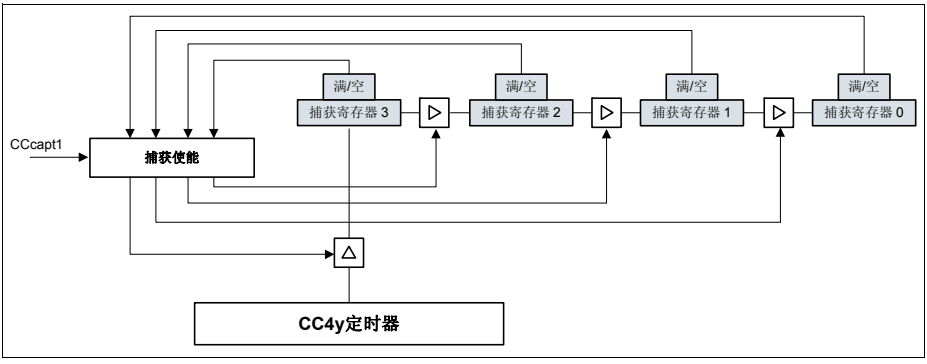


图 20-45　　定时器片捕获逻辑 - CC8yTC.SCE = 1_B

20.2.8.7 捕获扩展回读模式

每个定时器片的捕获逻辑都可以工作在 FIFO 回读模式。该模式可通过设置 **CC8yTC**.ECM = 1_B 来使能。该扩展回读模式允许软件总是从同一个地址读回捕获数据 (**CC8yECRD0** 用于与捕获触发信号 0 连接的结构，**CC8yECRD1** 用于与捕获触发信号 1 连接的结构)。该回读操作将总是返回最老的捕获值，从而简化了重建捕获数据的软件程序实现。

该功能允许每个捕获触发信号使用一个单独的 FIFO 结构。当发生多个捕获触发而软件来不及对每个捕获事件执行读操作时，该功能将大大减轻软件回读程序的负荷。

该 FIFO 回读功能有两种结构：深度为 4 的 FIFO 结构和深度为 2 的 FIFO 结构。

读回的数据中还包含一个丢失值位域，它指示是否因为 FIFO 结构已满而丢掉了一个捕获触发信号。当检测到一个捕获事件而 FIFO 已满 (无论是否使能了连续捕获模式) 时，该位域被置位。当发生对 **CC8yECRD0**/**CC8yECRD1** 寄存器的下一次读操作时，该位域由硬件自动清除。该位域不能指示出丢失了多少捕获事件，它只能指示在两个 ECRD 读操作之间至少丢失了一个捕获事件(这可帮助软件来分析所读数据的哪一部分可用于计算)。

*注： 当 ECM 位域被置位时，仍可以读取各个捕获寄存器。不过，满标记只能在通过 **CC8yECRD0**/**CC8yECRD1** 地址完成一个回读操作时由硬件来清除。*

深度为 4 的 FIFO 结构

当捕获触发信号 1 被使能且 **CC8yTC**.SCE = 1_B (相同的捕获事件) 时，硬件中会存在深度为 4 的 FIFO 结构，如**图 20-46** 所示。

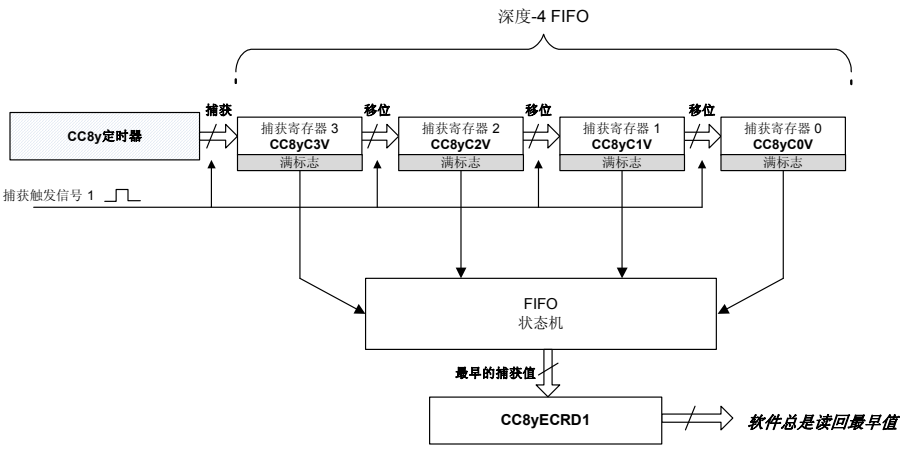


图 20-46 捕获扩展回读 - 深度 4

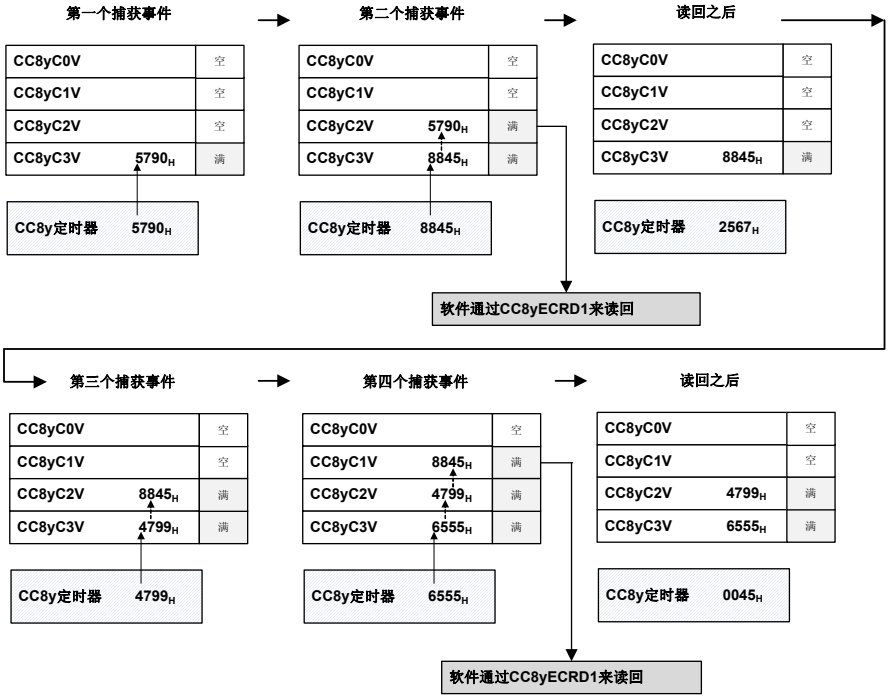


图 20-47　　深度 4 软件访问示例

深度为 2 的 FIFO 结构

每个定时器片有两个深度为 2 的捕获结构：一个与捕获触发信号 0 一起使用，而另一个与捕获触发信号 1 一起使用。

与捕获触发信号 0 连接的捕获结构通过 **CC8yECRD0** 来访问，而与捕获触发信号 1 连接的捕获结构通过 **CC8yECRD1** 来访问，如**图 20-48** 所示。

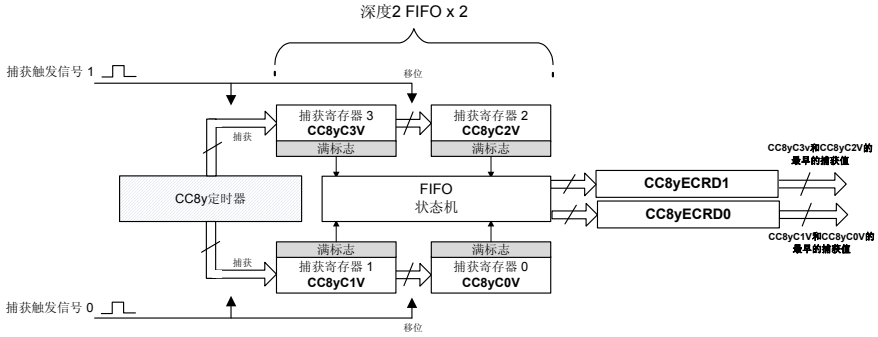


图 20-48　　捕获扩展回读 - 深度 2

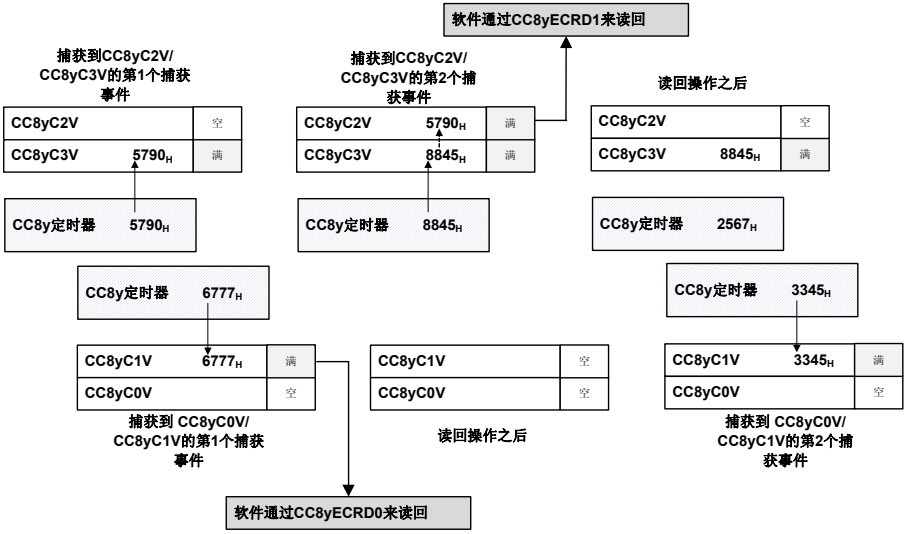


图 20-49　　深度 2 软件访问示例

20.2.8.8　外部调制信号

可以使用一个外部信号在每个定时器片的输出进行信号调制。

要选择一个外部调制信号，应将一个输入信号映射为一个事件，这可以通过在 **CC8yINS**.EVxIS 寄存器中设置所需要的值并在 **CC8yINS**.EVxLM 寄存器中指定信号的有效电平来完成。然后，应通过设置 **CC8yCMC**.MOS = 01_B （如果使用事件 0）或

CC8yCMC.MOS = 10_B (如果使用事件 1) 或 **CC8yCMC**.MOS = 11_B (如果使用事件 2) 将该事件映射为调制功能。

注意，调制功能为电平有效，因此主动态/被动态配置只能通过**CC8yINS**.EVxLM来设定。

外部调制信号可以独立地应用于每个比较通道，也可以通过设置 **CC8yTC**.EME = 11_B 将其应用到两个通道。

调制有两种工作模式：

- 调制事件用于复位 CC8yST 位 - **CC8yTC**.EMT = 0_B
- 调制事件用作输出门控 - **CC8yTC**.EMT = 1_B

在 **图 20-50** 中，有一个外部信号被配置为调制源用于清除 CC8ySTx 位， **CC8yTC**.EMT = 0_B。该信号被编程为低电平有效事件，因此当该信号为低电平时，输出值遵循标准的主动 / 被动规则。

当信号为高电平 (无效状态) 时，CC8yST 位被清零，输出会被强制为被动态。注意，状态位 CC8yST 的值和特定输出 CCU8x.OUTy 未连接在一起。可以通过 **CC8yPSL**.PSLx 位选择输出为低电平有效或高电平有效。

外部调制无效状态的退出与 PWM 周期同步，这是因为在调制信号处于无效状态期间，CC4yST 位被清除且不能被置位。

通过设定 **CC8yTC**.EMS = 1_B，还可以实现进入无效状态与 PWM 周期的同步。这样一来，可以避免输出信号所有可能的假信号，如**图 20-51** 所示。

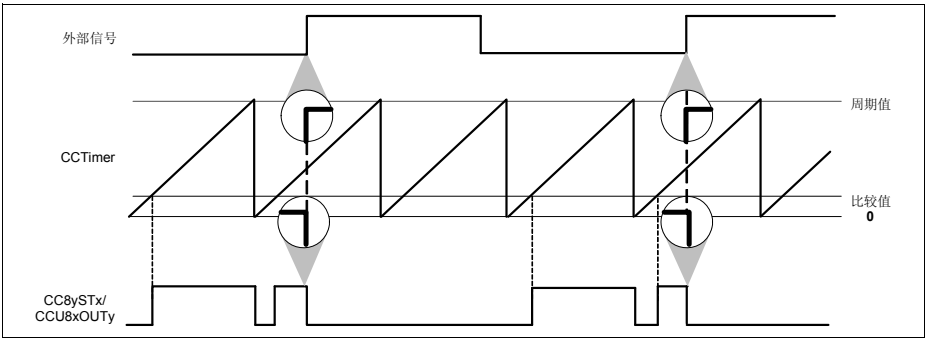


图 20-50　　外部调制信号复位 ST 位 - CC8yTC.EMS = 0_B

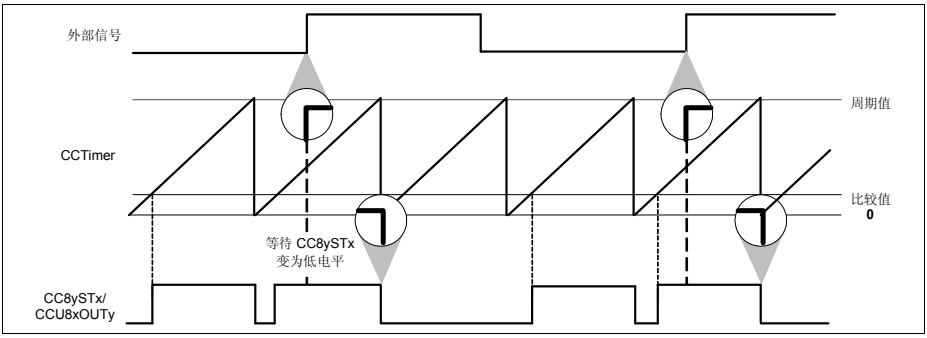


图 20-51　　外部调制信号复位 ST 位 - CC8yTC.EMS = 1_B

在图 20-52 中，外部调制事件被用作输出的门控信号，CC8yTC.EMT = 1_B。外部信号被配置为高电平有效，CC8yINS.EVxLM = 0_B，这意味着当外部信号为高电平时，输出被设定为被动状态。在该模式下，通过设定 CC8yTC.EMS = 1_B，可以使门控事件与 PWM 信号同步。

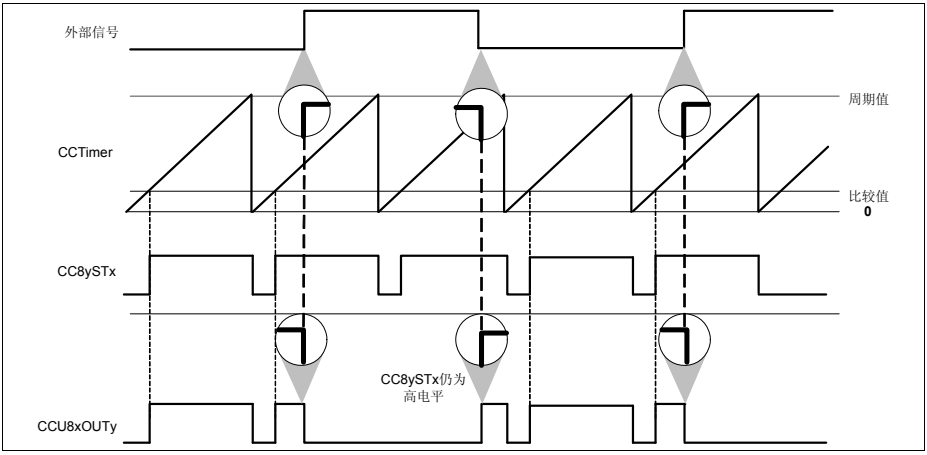


图 20-52　　外部调制信号门控输出 - CC8yTC.EMT = 1_B

20.2.8.9　陷阱 功能

陷阱功能允许 PWM 输出对输入引脚的状态做出反应。该功能可在陷阱输入变为有效时用于关闭功率器件。

要选择陷阱功能，应将一个输入信号映射为事件 2，这可以通过在 CC8yINS.EV2IS 寄存器中设置所需要的值并在CC8yINS.EV2LM寄存器中指定信号的有效电平来完成。然后，应通过设定 CC8yCMC.TS = 1_B 将该事件映射为陷阱功能。

注意，陷阱功能为电平有效，因此主动态/被动态配置只能通过**CC8yINS**.EV2LM来设定。

可以通过软件来监视两个位域，以交叉检验陷阱功能，如**图 20-53** 所示：

- 陷阱状态位 **CC8yINTS**.E2AS。该位域指示陷阱功能当前是否有效。因此，该位域可将特定定时器片的输出设置为主动态或被动态。
- 陷阱标志 **CC8yINTS**.TRPF。在陷阱条件被硬件自动清除的情况下，该位域被用作遗留标志。该位域需要由软件来清除。

可通过 **CC8yTC**.TRAPEy 位域将 E2AS 配置为影响所有 CCU8 定时器片输出，或影响一个特定的输出子集。

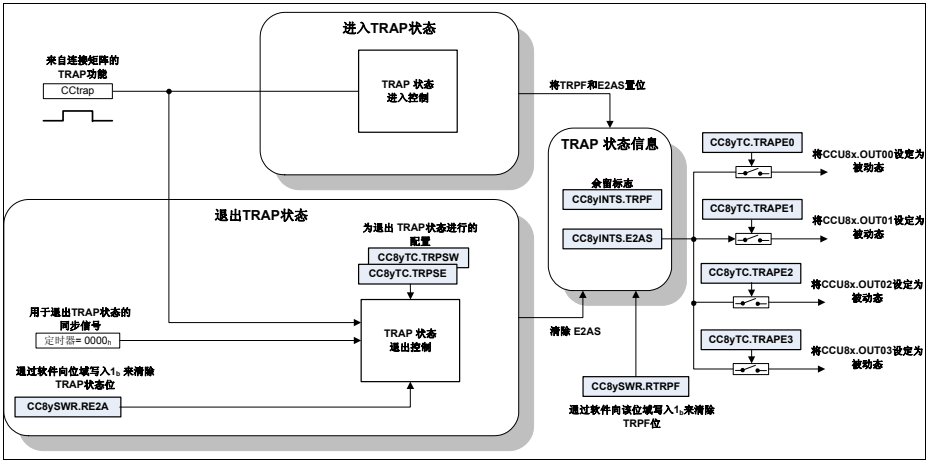


图 20-53　　陷阱 控制示意图

当在所选择的输入引脚上检测到陷阱条件时，陷阱标志和陷阱状态位都会被置为 1_B。通过将 CCU4xOUTy 设置为编程的被动态，会立即进入陷阱状态，见**图 20-54**。

退出陷阱状态有两种方式 (**CC8yTC**.TRPSW 寄存器)：

- 当陷阱信号变为无效时 - **CC8yTC**.TRPSW = 0_B，由硬件自动退出。
- 仅由软件通过清除 **CC8yINTS**.E2AS 来执行退出。仅在输入的陷阱信号处于无效状态时，清除操作才有效 - **CC8yTC**.TRPSW = 1_B。

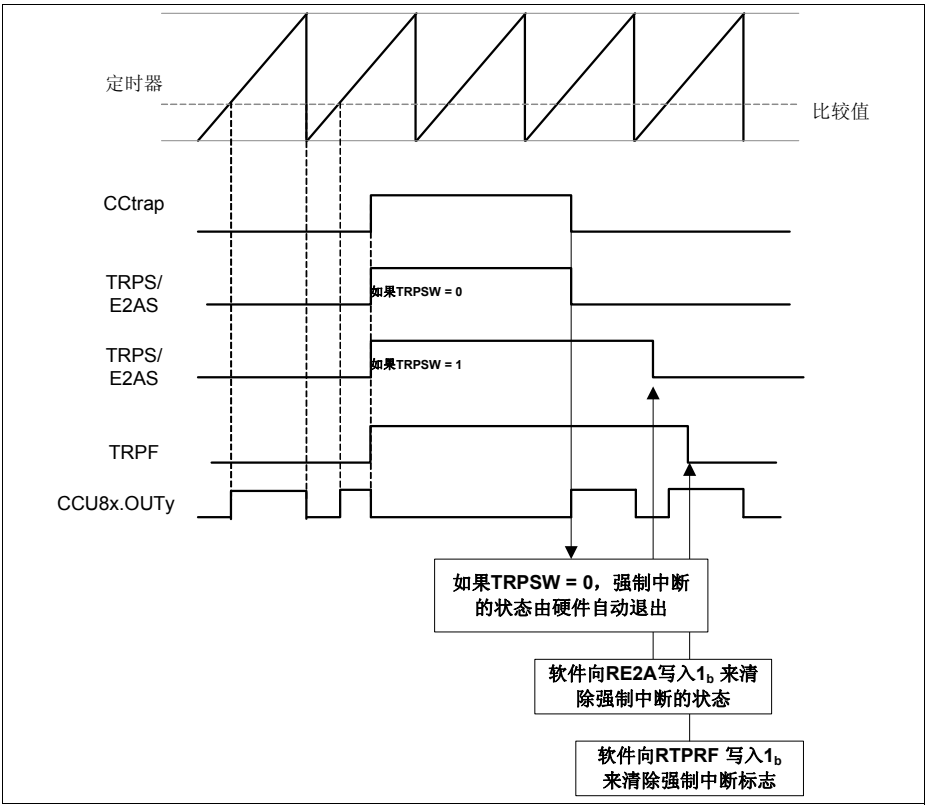


图 20-54　　陷阱时序图 , CC8yTCST.CDIR = 0 CC8yPSL.PSL = 0

还可以将退出陷阱状态与PWM信号进行同步，如图20-55所示。当位域CC8yTC.TRPSE = 1_B 时，该功能被使能。

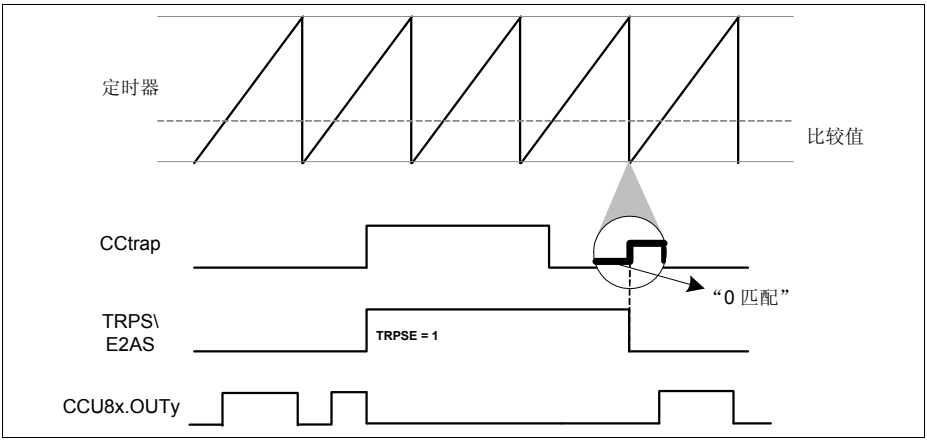


图 20-55　　陷阱与 PWM 信号同步

20.2.8.10 状态位覆盖

对于复杂的定时输出控制，每个定时器片都有用一个外部信号的值来覆盖比较通道 1 状态位 (CC8yST1) 的功能。

状态位覆盖能引起输出引脚 CCU8x.OUTy0 和 CCU8x.OUTy1 的状态变化 (从无效变为有效，或反之)。

要使能该功能，需要两个信号：

- 一个信号用作覆盖状态位的触发信号 (边沿有效)
- 一个信号包含状态位要被设定的值 (电平有效)

要选择状态位覆盖功能，应当将用作触发的信号映射为事件　　1，这可以通过在 **CC8yINS**.EV1IS 寄存器中设置所需要的值并在 **CC8yINS**.EV1EM 寄存器中指定信号的有效边沿来完成。

承载着状态位设定值的信号需要被映射到事件 2，这可通过在 **CC8yINS**.EV2IS 寄存器中设置所需要的值来实现。如果不需要对信号进行反相，则应将 **CC8yINS**.EV2LM 寄存器设置为 0_B，否则，设置为 1_B。

然后，应通过设置 **CC8yCMC**.OFS = 1_B 将事件映射为状态位覆盖功能。

图 20-56 展示了状态位覆盖功能，在此，外部信号 (1) 被选为触发源 (上升沿有效)，外部信号 (2) 被选为覆盖值。

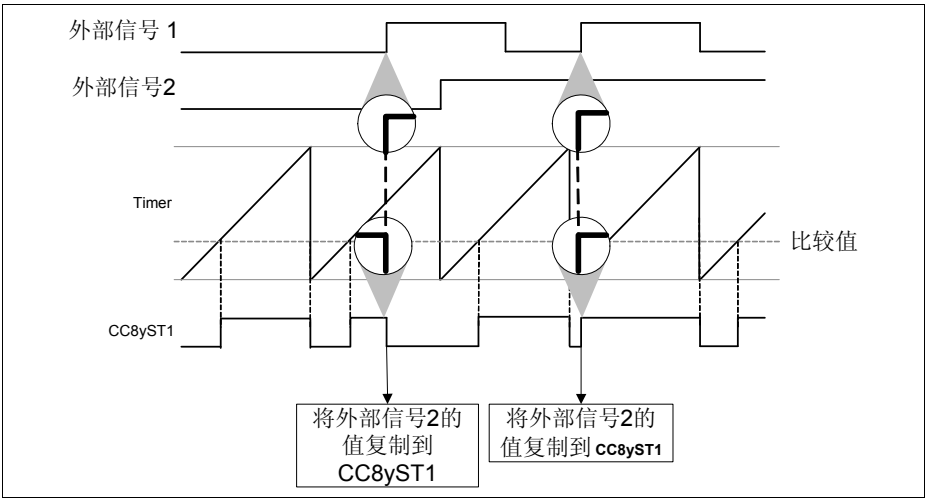


图 20-56　　状态位覆盖

20.2.9　多通道支持

可通过设置 **CC8yTC**.MCME = 1_B 来单独选择每个定时器片的多通道控制模式。

在该模式，定时器片 (设置为多通道模式) PWM 信号的输出状态可通过一个序列并行控制。

该序列由 CCU8 的输入 CCU8x.MCIy[3:0] 来控制。每组输入都被连接到相应的定时器片：定时器片 0 连接 CCU8xMCI0[3:0]，定时器片 1 连接 CCU8xMCI1[3:0]，定时器片 2 连接 CCU8xMCI2[3:0]，定时器片 3 连接 CCU8xMCI1[3:0]。

该序列可由一个 POSIF 模块直接控制，所有定时器片的序列可以并行更新。

将 POSIF 模块与 CCU8 的多通道支持结合起来使用，可以实现输出状态 (CCU8x.OUTy) 更新与一个新序列更新之间的完全同步，见**图 20-57**。

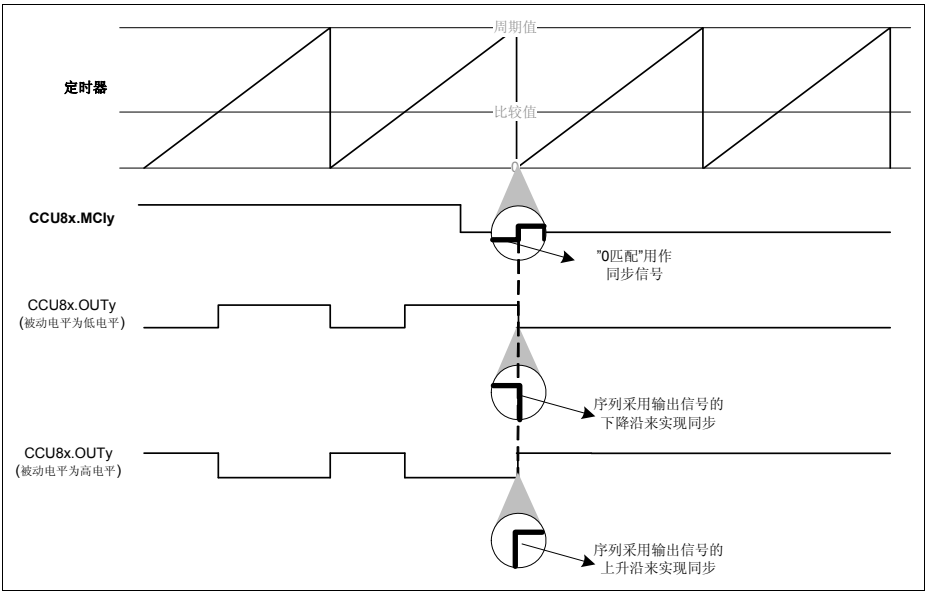


图 20-57　　多通道序列同步

这些序列输入要在输出调制控制单元中使用，以将特定的 PWM 输出置于主动或被动状态：CCU8x.MCIy[0] 影响 CC8yST1 的路径，因此它控制 CC8xOUT00 引脚；CCU8x.MCy[1] 的使用方式相同，针对反相的 CC8yST1 路径； CCU8x.MCIy[2] 和 CCU8x.MCIy[3] 被分别连接到 CC8yST2 和反相的 CC8yST2 路径。**图 20-58** 示出了多通道控制的简化原理图。

图 20-59 展示了多通道模式与 CCU8 的两个定时器片相结合的使用方法。多通道序列由 POSIF 模块驱动，这可以在更新 CCU8 的所有输出时避免出现毛刺。

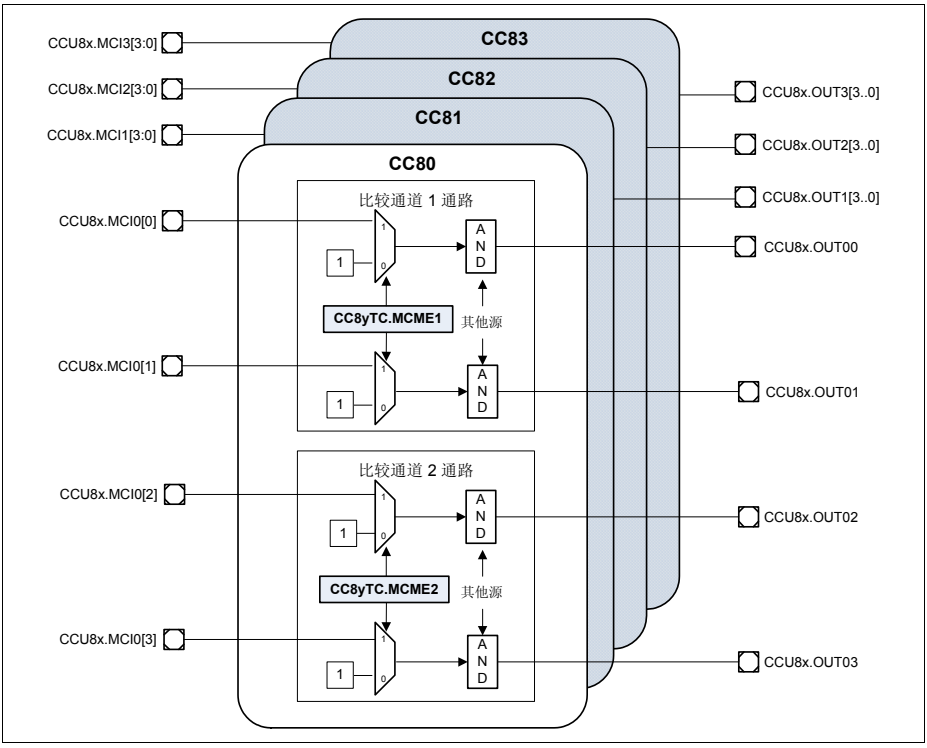


图 20-58　　CCU8 多通道概览

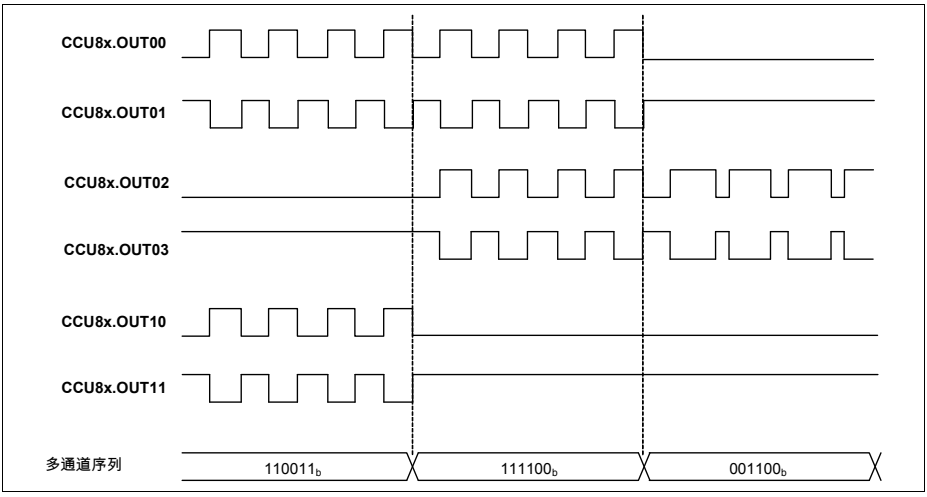


图 20-59　　多个定时器片的多通道模式

CCU8 与 POSIF 之间的同步可通过在每个定时器片的输出通路上 (在状态位 CC8yST 和输出引脚的直接控制之间) 增加三个周期的延时来实现。该通路仅在 **CC8yTC**.MCME = 1_B 时被选择。

图 20-60 展示了 CC8yST1 通路的控制。其余通路的控制遵循相同的机制 (如果 **CC8yTC**.MCME2 = 1_B，则只能对 CC8yST2 通路使能多通道模式)。

多通道序列输入的同步如**图 20-61** 所示。为了实现状态位更新的同步，新多通道模式序列输入的采样由周期匹配或 1 匹配信号来控制。

在一个不使用外部信号控制计数方向的标准操作中，在边沿对齐模式下，用于使能序列采样的信号总是周期匹配信号，而在中心对齐模式下，总是用 1 匹配信号使能序列采样。当使用一个外部信号控制计数方向时，根据计数器是向上还是向下计数，分别使用周期匹配信号或 1 匹配信号来使能序列采样。

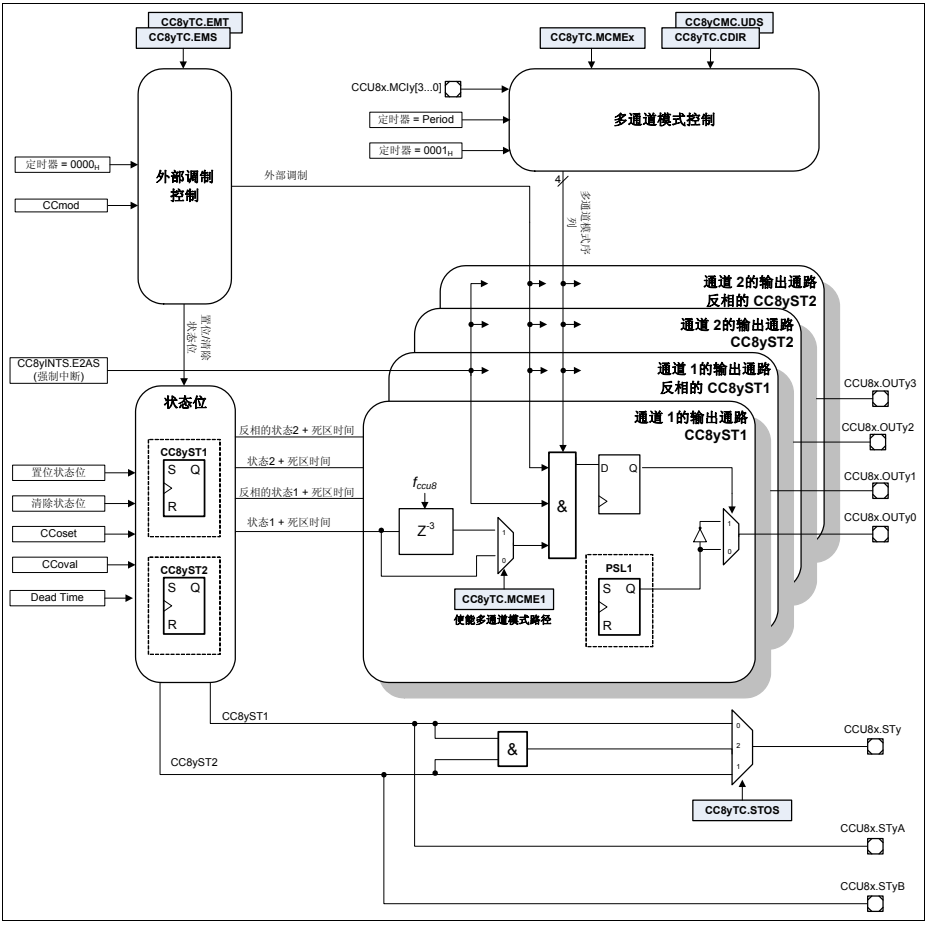


图 20-60　　输出控制示意图

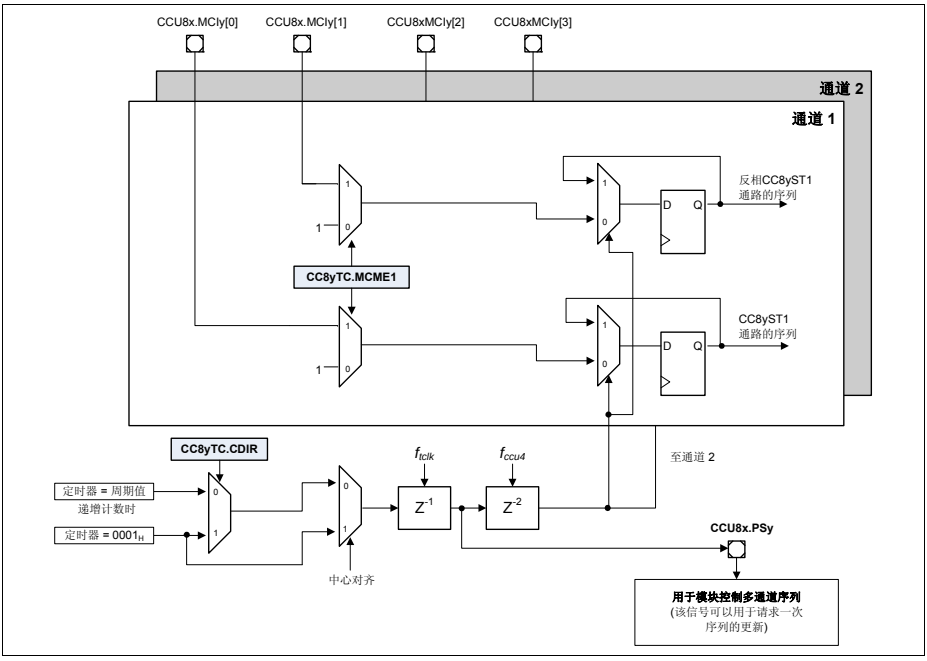


图 20-61　　多通道序列同步控制

20.2.10　定时器级联

CCU8 提供了一个非常简单的机制来完成同步定时器级联。该功能可以通过设置 **CC8yTC**.TCE = 1_B 来启用。使能该功能后，用户可实现当前 CCU8 片和前一个片的级联，如**图 20-62** 所示。

注意，不可能将不相邻的片进行级联，而且定时器级联会自动将定时器片设置为边沿对齐模式。在中心对齐模式，不可能进行定时器级联。

要实现一个 64 位定时器，应在所有定时器片中设置 **CC8yTC**.TCE = 1_B (CC80 除外，因为它不包含这样的控制域)。

要实现一个 48 位定时器，应在两个相邻的定时器片中设置 **CC8yTC**.TCE = 1_B。而要实现一个 32 位定时器，应将包含最高有效位 (MSBs) 的定时器片的 **CC8yTC**.TCE 设置为 1_B。注意，应将包含最低有效位 (LSBs) 的定时器片的 TCE 位域设置为 0_B 。

在一个 CCU8 模块中，可以实现多种定时器级联组合：

- 一个 64 位定时器
- 一个 48 位定时器和一个 16 位定时器
- 两个 32 位定时器
- 一个 32 位定时器和两个 16 位定时器

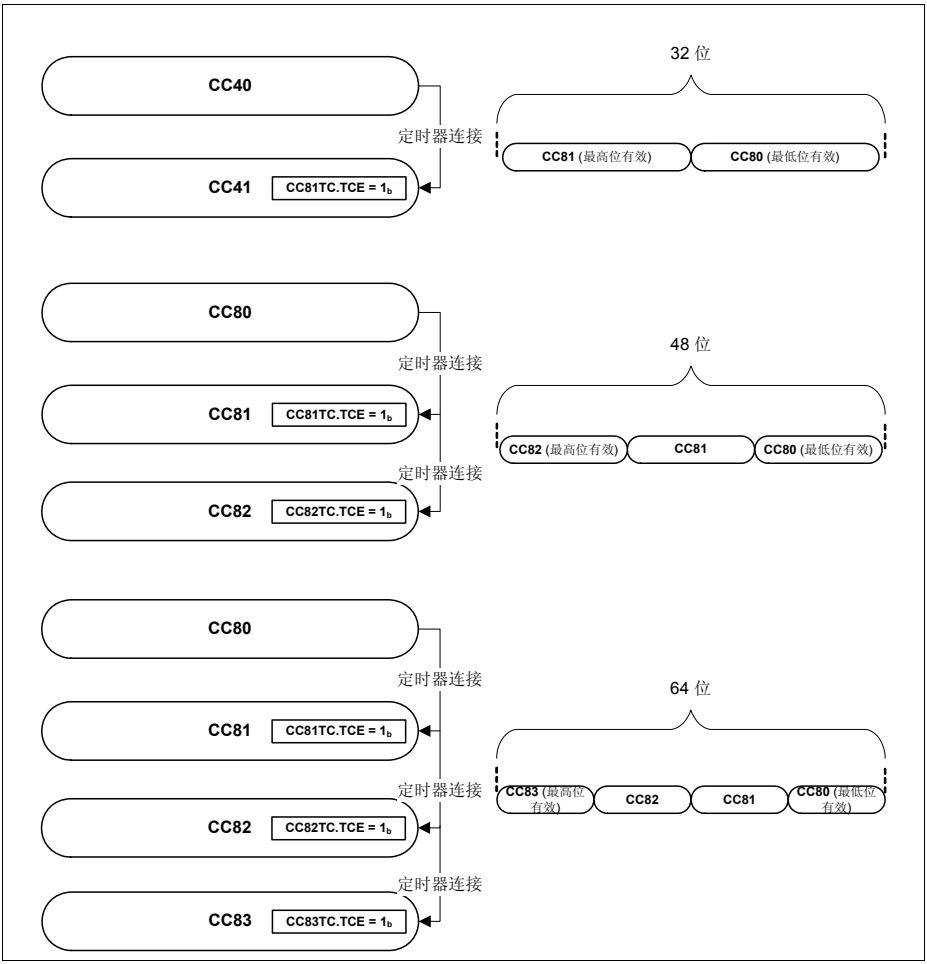


图 20-62　　定时器级联示例

每个定时器片都通过一个专用的级联接口与相邻的定时器片连接。该接口不仅允许定时器计数操作的级联，而且还允许捕获和加载操作的输入触发信号的同步处理，如图 20-63 所示。

注：　在所有情况下，CC80 和 CC83 都不被认为是相邻定时器片。

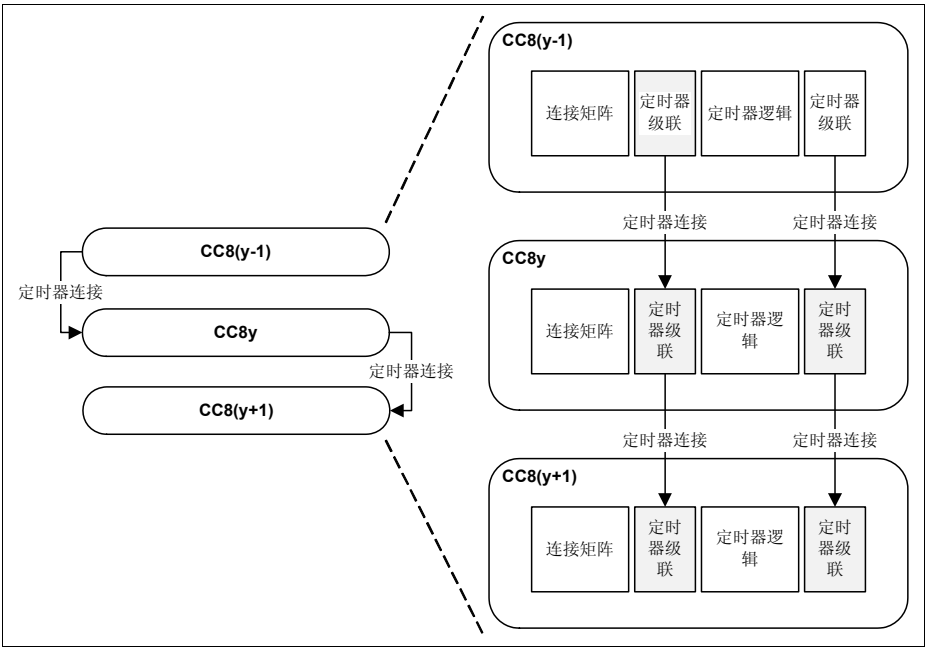


图 20-63　　定时器级联连接

定时器级联接口有 8 个信号：

- 定时器周期匹配 (CC8yPM)
- 定时器 0 匹配 (CC8yZM)
- 通道 1 的定时器比较匹配 (CC8yCM1)
- 通道 2 的定时器比较匹配 (CC8yCM2)
- 定时器计数方向功能 (CCupd)
- 定时器加载功能 (CCload)
- CC8yC0V 寄存器和 CC8yC1V 寄存器的定时器捕获功能 (CCcap0)
- CC8yC2V 寄存器和 CC8yC3V 寄存器的定时器捕获功能 (CCcap1)

前 5 个信号用于在定时器逻辑的输出执行同步定时级联，如**图 20-63** 所示。借助于该连接，定时器的长度可很容易地被调整为 32 位、 48 位和 64 位 (向上或向下计数)。

后 3 个信号用于对级联的定时器系统执行捕获和加载功能之间的同步连接。这意味着用户可以在第一个定时器片内通过编程设置捕获或加载功能，并将该捕获或加载触发信号同步地从最低有效位传递到最高有效位，如**图 20-64** 所示。

捕获或加载功能仅需在第一个定时器片 (保持着最低有效位的定时器片) 进行配置。从 **CC8yTC**.TCE 被置为 1_B 那一时刻起，后面定时器片的这些功能之间的连接由硬件自动完成。

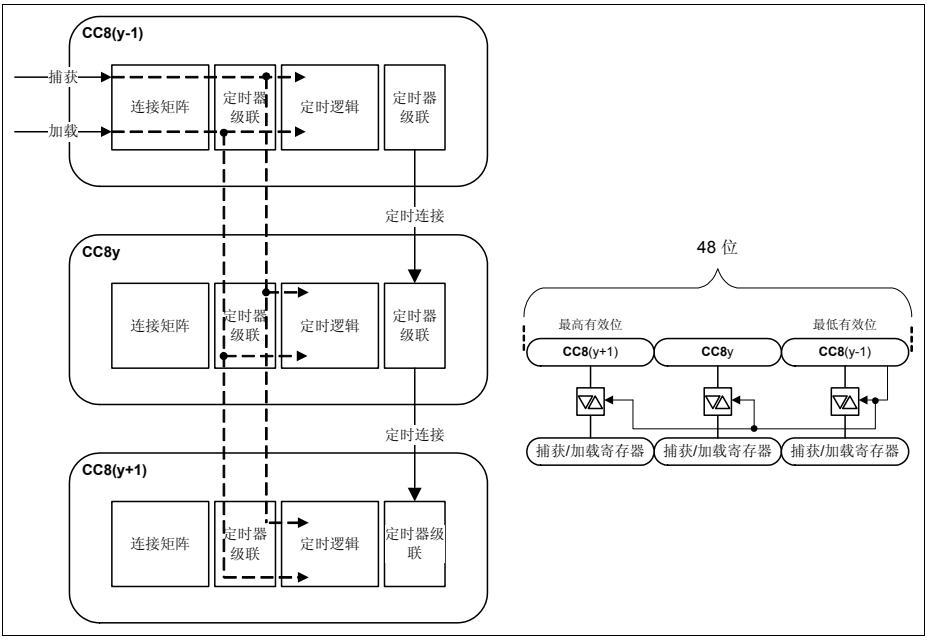


图 20-64　　捕获 / 加载定时器级联

在级联模式，要使用来自前一定时器片 (具有小一级的索引) 的周期匹配 (CC8yPM) 或 0 匹配 (CC8yZM) 作为计数器的门控信号。这意味着最高有效位的计数操作仅发生在检测到一个回绕条件时，避免了提取计数值的额外 DSP 操作。

采用相同的方法，比较匹配 (CC8yCM1 和 CC8yCM2)、0 匹配和周期匹配由来自前一定时器片的特定信号进行门控。也就是说，定时信息被传播到所有定时器片中，这可以实现最低有效位计数和最高有效位计数之间的完全同步匹配，见**图 20-65**。

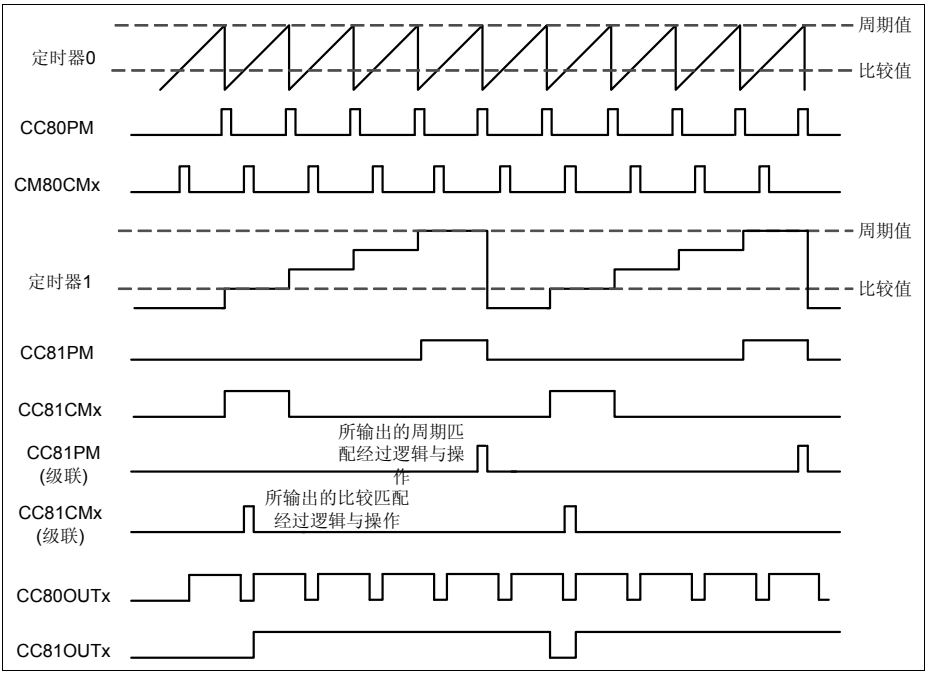


图 20-65　　32 位级联的时序图

注：　级联的定时器的计数方向需要被固定。定时器可以向上或向下计数，但是计数方向却不能够在运行中修改。

图 20-66 给出了定时器级联逻辑的概览。注意，所有机制都由 **CC8yTC**.TCE 位域单独控制。

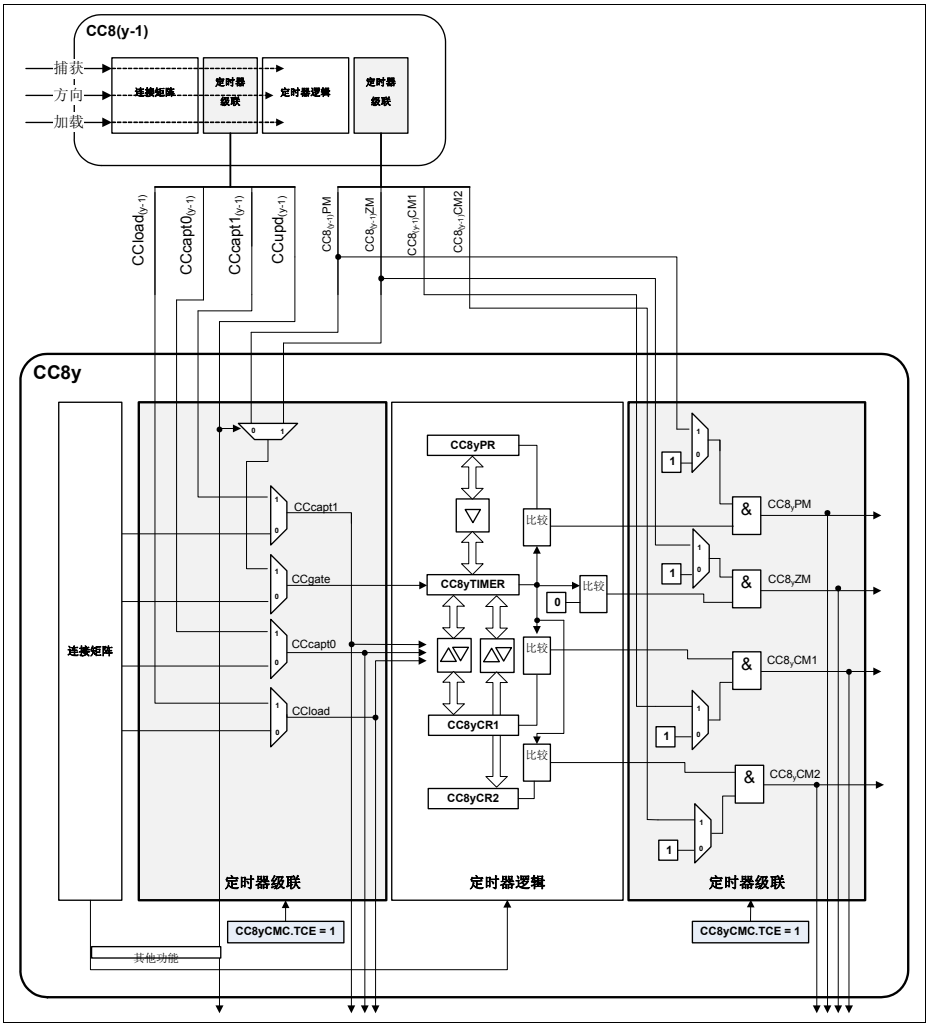


图 20-66　　定时器级联控制逻辑

20.2.11　输出奇偶校验器

奇偶校验器功能可通过将 **GIDLC**.PCH 位域置为 1_B 来使能 (奇偶校验器在 **GSTAT**.PCRB 为 0_B 时被禁止)。

奇偶校验器功能将CCU8模块的输出值与应被连接到一个驱动器XOR结构的输入信号进行对比检查。

可以在输出切换与测定来自该驱动器的输入信号之间添加一个延时，并选择奇偶校验类型，即偶校验或奇校验 (通过 **GPCHK**.PCTS 位域设置)。**图 20-67** 示出了奇偶校验器单元的结构。

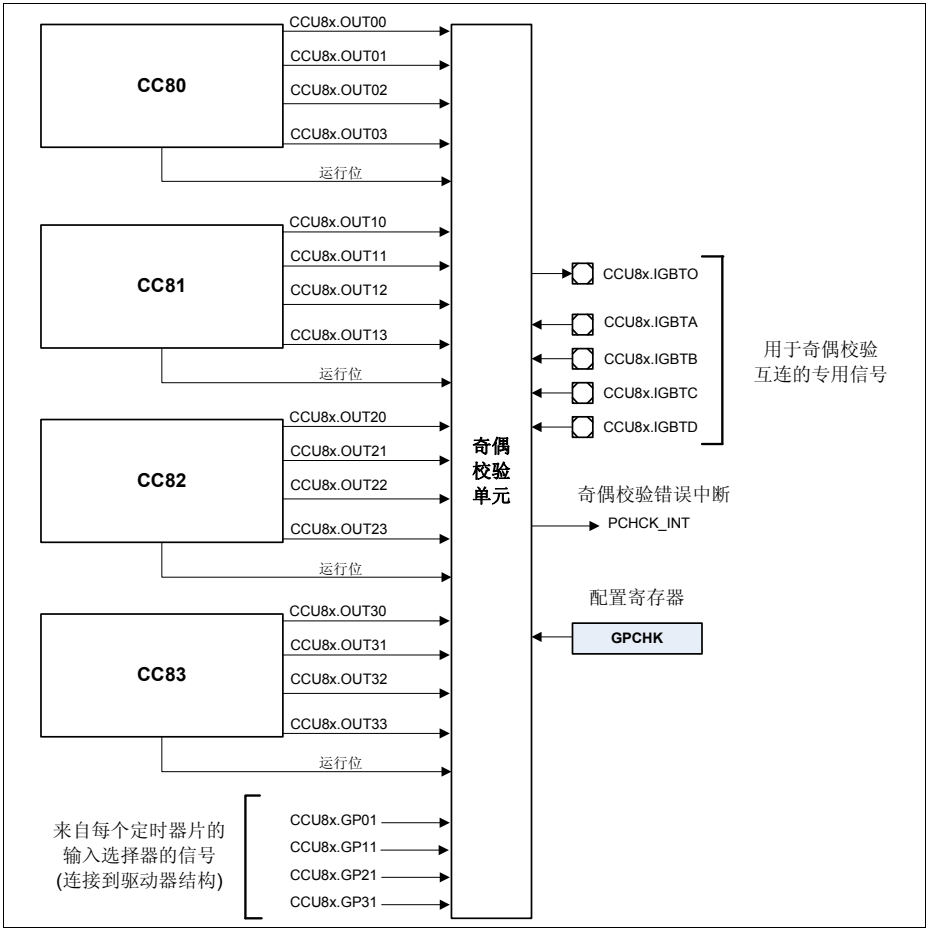


图 20-67　　奇偶校验器结构

要使用奇偶校验功能，用户必须选择将哪一个信号连接到驱动器奇偶结构：

- 该信号可被连接到任何一个定时器片输入

- 该信号必须通过每个定时器片的输入选择器选择。该信号必须被映射到每个定时器片的事件 1。

每个 CCU8 输出都可以被单独选择为奇偶串的一部分，每当来自驱动器结构的输入信号与内部产生的 XOR 结果不匹配时，就会产生中断。

中断被连接到定时器片的 E1AS 状态位，驱动器的奇偶输出也连接到该状态位：

- 如果 **GPCHK**.PISEL = 00_B，则错误状态位于 CC80INTS.E1AS 中
- 如果 **GPCHK**.PISEL = 01_B，则错误状态位于 CC81INTS.E1AS 中
- 如果 **GPCHK**.PISEL = 10_B，则错误状态位于 CC82INTS.E1AS 中
- 如果 **GPCHK**-PISEL = 11_B，则错误状态位于 CC83INTS.E1AS 中

奇偶校验的逻辑结构如**图 20-68** 所示。关于奇偶校验器资源使用的详细描述，请见**20.2.14.5 节**。

配置示例：

驱动器奇偶输出连接到输入 CCU8x.IN1B (其中 x = CCU8 单元)。用于控制开关延迟的输入为 CCU8x.IGBTCCCU8。驱动器使用 12 个来自前三个定时器片的输出，采用偶校验 (PCTS 域为默认状态)。

应对下述寄存器编程：

CC8yINS.EV1IS = 0001_B；选择输入 CCU8x.IN1B

GPCHK.PISEL = 01_B；选择来自定时器片 1 的事件 1

GPCHK.PCDS = 10_B；选择 CCU8x.IGTBC 为延迟控制输入

GPCHK.PCSEL = $0FFF_H$；仅选择前三个定时器片的输出信号用于奇偶校验

GIDLC.SPCH = 1_B；启动奇偶校验功能

当检测到驱动器输出与奇偶校验器不匹配时，在定时器片 1 会产生中断。保存该信息的中断状态位是 **CC8yINTS**.E1AS。

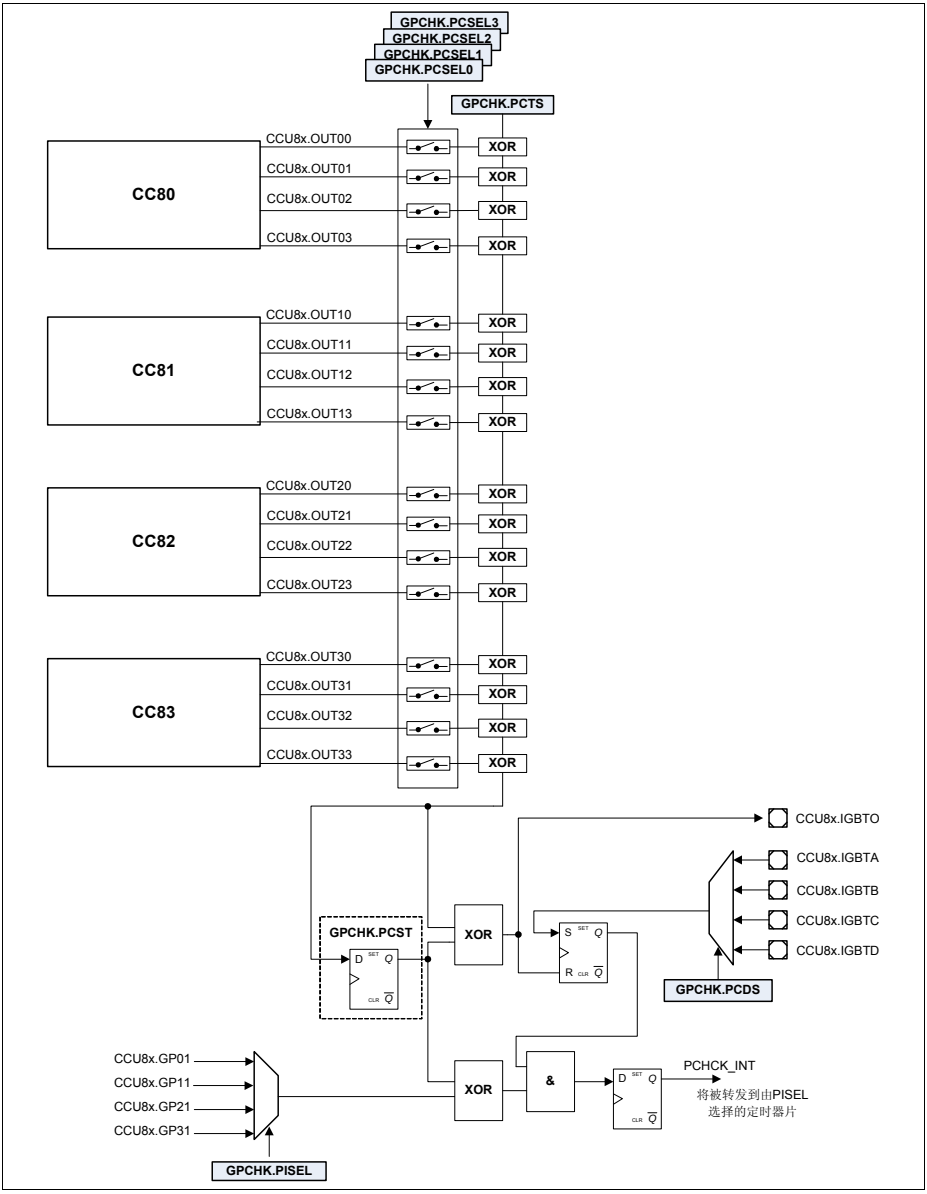


图 20-68　　奇偶校验器逻辑

20.2.12 PWM 抖动

CCU8 有一个自动 PWM 抖动插入功能。该功能可以用在不能快速更新周期 / 比较值的非常慢的控制回路，因为这样的回路在长时间运行时会损失精度。通过引入 PWM 信号的抖动，可对平均频率 / 占空比进行补偿，减小误差。

每个定时器片都包含一个抖动控制单元，如图 20-69 所示。

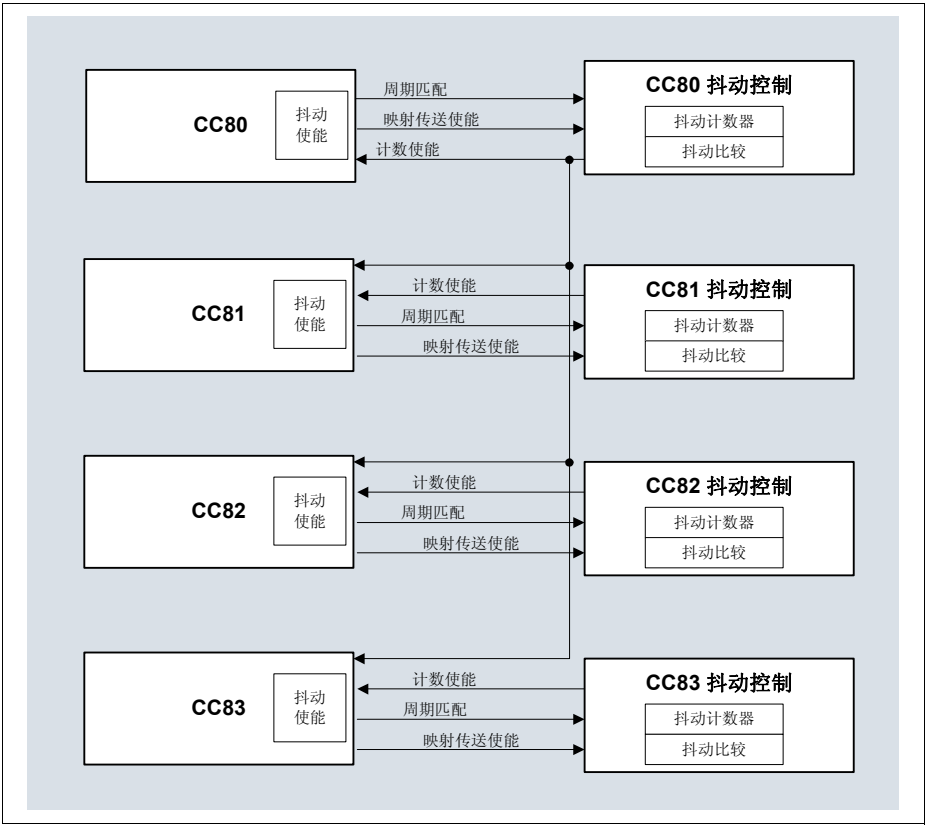


图 20-69　　抖动结构概览

抖动控制单元包含一个 4 位计数器和一个比较值寄存器。每当发生一次周期匹配时，该 4 位计数器增 1。计数器工作在位反转模式，因此在 16 个计数器周期内保持均匀增 1，如表 20-6 所示。

表 20-6 抖动位反转计数器

counter[3]	counter[2]	counter[1]	counter[0]
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	0
0	1	1	0
1	1	1	0
0	0	0	1
1	0	0	1
0	1	0	1
1	1	0	1
0	0	1	1
1	0	1	1
0	1	1	1
1	1	1	1

然后，计数器与一个编程值 **CC8yDIT**.DCV 相比较。如果计数器值小于该编程值，会产生一个门控信号，该信号用于将周期扩展一个时钟周期、或将比较匹配延迟一个时钟周期、或两者兼具 (由 **CC8yTC**.DITHE 域控制，详见**表 20-7**) 。

表 20-7 抖动模式

DITHE[1]	DITH[0]	模式
0	0	禁止抖动
0	1	将周期值增加一个时钟周期
1	0	将比较匹配延迟一个时钟周期
1	1	将周期值增加一个时钟周期，并将比较匹配延迟一个时钟周期

抖动比较值也有一个相关联的映射寄存器，该寄存器可以与 CC8y 的周期 / 比较寄存器同步更新。抖动单元的控制逻辑如**图 20-70** 所示。

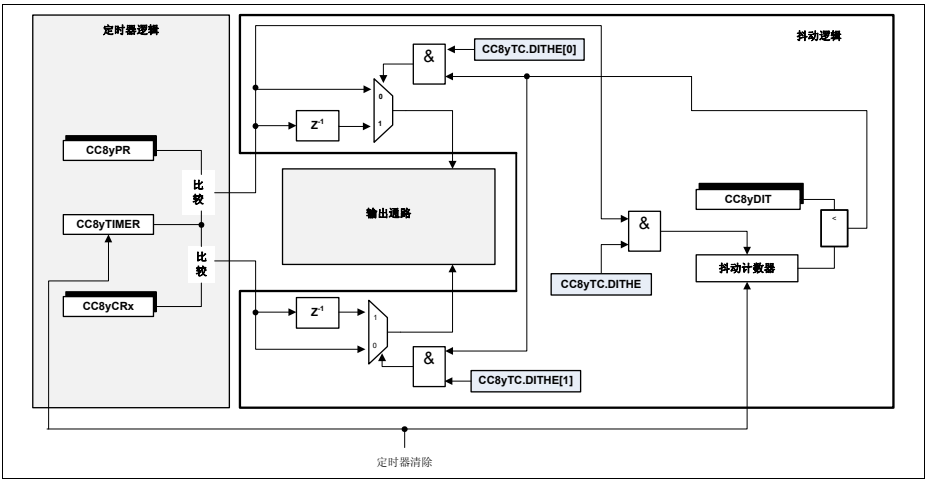


图 20-70　　抖动控制逻辑

图 20-71 到 图 20-76 展示了在边沿对齐和中心对齐这两种计数方式下采用不同抖动配置 (CC8yTC.DITHE) 的效果。在每个图中，抖动计数器用位反转方案来表示，比较值被编程为 8_H。在每个图中，变量 *T* 代表计数器的周期，变量 *d* 表示占空比 (状态位被设置为高电平)。

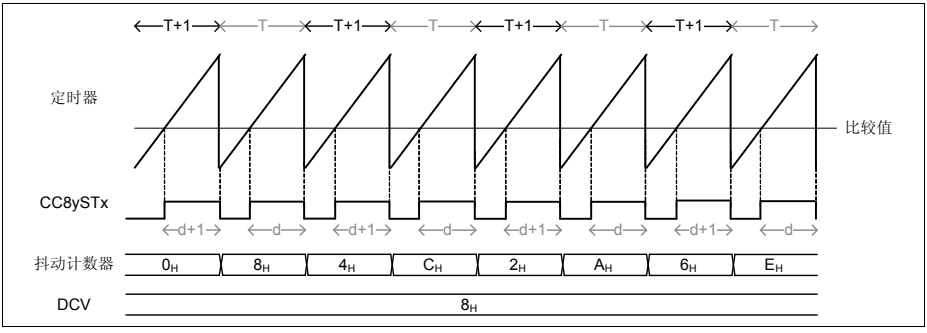


图 20-71　　边沿对齐方式下的抖动时序图 - CC8yTC.DITHE = 01_B

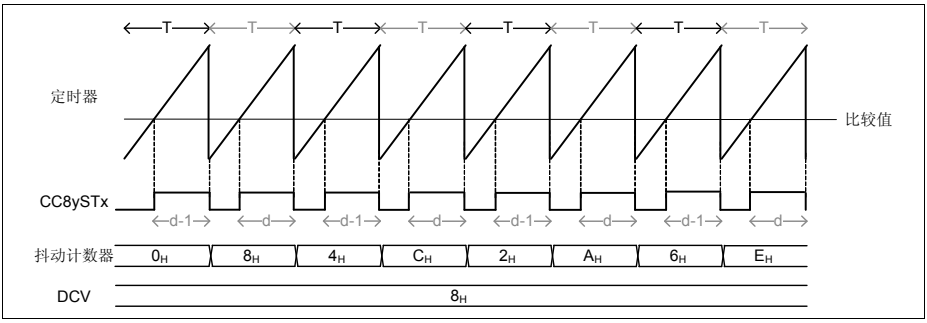


图 20-72 边沿对齐方式下的抖动时序图 - **CC8yTC.DITHE = 10_B**

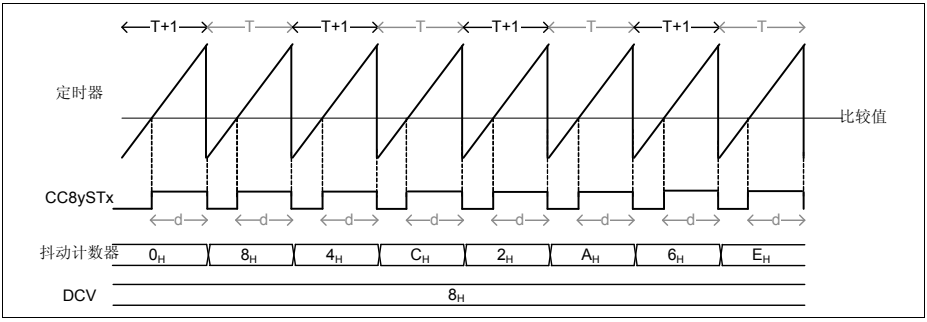


图 20-73 边沿对齐方式下的抖动时序图 - **CC8yTC.DITHE = 11_B**

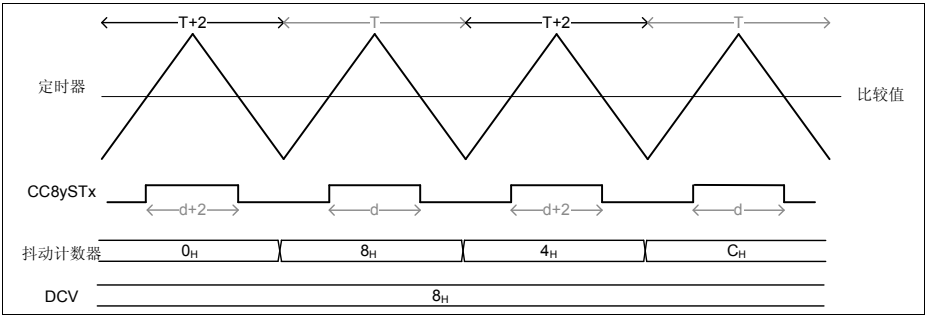


图 20-74 中心对齐方式下的抖动时序图 - **CC8yTC.DITHE = 01_B**

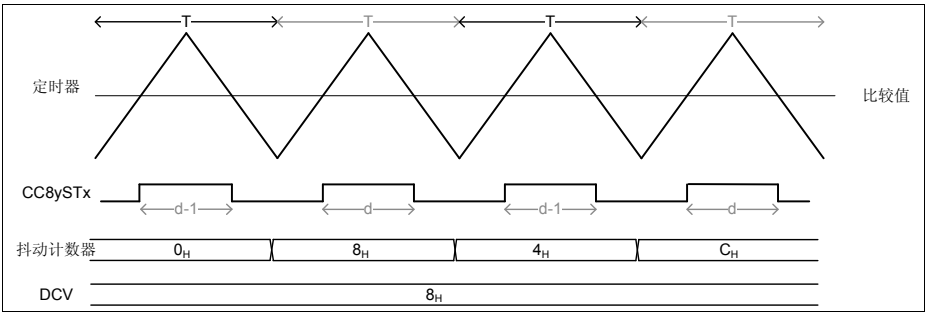


图 20-75 中心对齐方式下的抖动时序图 - **CC8yTC**.DITHE = 10_B

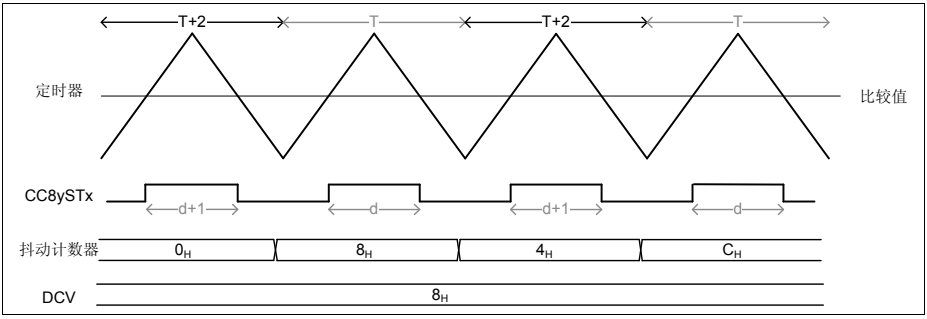


图 20-76 中心对齐方式下的抖动时序图 - **CC8yTC**.DITHE = 11_B

注： 当使用抖动时，在边沿对齐模式下不可能选择一个为 FS 的周期值。在中心对齐模式，周期值必须至少为 FS - 2。

20.2.13 预分频器

CCU8 包含一个 4 位预分频器。对于每个定时器片，该预分频器都有两种工作模式：

- 标准预分频器模式
- 浮动预分频器模式

通过用软件分别向寄存器 **GIDLC**.SPRB 和 **GIDLS**.CPRB 写入，可以置位或清除预分频器的运行位，或者可以通过一个特定的定时器片的运行位来清除预分频器的运行位。在采用后一种方案的情况下，在所选定时器片的运行位被清除一个时钟周期后，预分频器的运行位被清除。要选择用哪个定时器片来执行运行位清除操作，应对 **GCTRL**.PRBC 寄存器进行编程。

20.2.13.1 标准预分频器模式

在标准预分频器模式，接入 CC8y 计数器的时钟是对模块时钟的一个固定的 N 分频， N 由 **CC8yPSC**.PSIV 寄存器中的设置值来确定。**表 20-8** 列出了可能的分频值。寄存器 **CC8yPSC**.PSIV 的值只能通过软件访问来修改。注意，每个定时器片有一个专用的预分频器值选择器 (**CC8yPSC**.PSIV)，这意味着用户可以为每个定时器片 (CC8y) 选择不同的计数器时钟。

表 20-8 定时器时钟分频选项

CC8yPSC.PSIV	结果时钟
0000_B	f_{ccu8}
0001_B	$f_{ccu8}/2$
0010_B	$f_{ccu8}/4$
0011_B	$f_{ccu8}/8$
0100_B	$f_{ccu8}/16$
0101_B	$f_{ccu8}/32$
0110_B	$f_{ccu8}/64$
0111_B	$f_{ccu8}/128$
1000_B	$f_{ccu8}/256$
1001_B	$f_{ccu8}/512$
1010_B	$f_{ccu8}/1024$
1011_B	$f_{ccu8}/2048$
1100_B	$f_{ccu8}/4096$
1101_B	$f_{ccu8}/8192$
1110_B	$f_{ccu8}/16384$
1111_B	$f_{ccu8}/32768$

20.2.13.2 浮动预分频器模式

在每个定时器片中都可以独立使用浮动预分频器模式，这可以通过设置寄存器 **CC8yTC**.FPE = 1_B 来配置。使用该模式，用户不仅能为比较操作获得更高的计数器时钟精度，而且还可以减少捕获模式的软件读访问。

除了初始值寄存器 **CC8yPSC**.PSIV 以外，浮动预分频器还包含一个比较寄存器 **CC8yFPC**.PCMP，该寄存器具有相关联的映射寄存器。

图 20-77 示出了当特定定时器片工作在比较模式 (无外部信号用于捕获) 时预分频器在浮动模式下的结构。在该模式，每当定时器发生一次上溢 / 下溢 (在边沿对齐模式为上溢，在中心对齐模式为下溢) 时，时钟分频值增 1_D。

在该模式，定时器的比较匹配 （两个通道）信号和预分频器的比较匹配信号相与。每当发生定时器的比较匹配事件后，在紧接着的定时器上溢 / 下溢时刻，时钟分频值被更新为 **CC8yPSC**.PSIV 的值。

如果要使用一个比较通道来控制预分频器，则必须禁止另一个比较通道。因此，需要将比较值 **CC8yCR1** 或 **CC8yCR2** (取决于使用哪个通道) 设置为一个比周期 **CC8yPR** 大的值。这意味着在边沿对齐模式定时器周期的最大值为 65534_D，因为一个通道的比较值需要被设置为 65535_D。

浮动预分频器比较值 **CC8yFPC**.PCMP 的映射传送遵循 **20.2.5.2 节**所述的相同规则来完成。

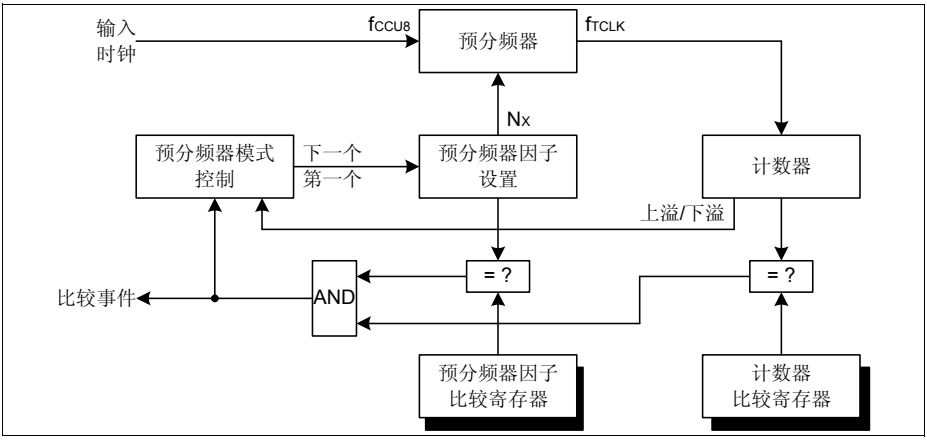


图 20-77　　比较模式下浮动预分频器概览

在特定的 CCU8 工作在捕获模式 (至少有一个外部信号被用作捕获功能) 期间，每当发生一次捕获事件时，还需要存储时钟分频的当前值。浮动预分频器最多可有 4 个捕获寄存器 (捕获寄存器的最大数目由特定定时器片中所使用的捕获寄存器数来限定)。

每当定时器发生一次溢出 (在捕获模式，定时器片总是工作在边沿对齐模式) 时，时钟分频值继续增 1，而每检测到一个捕获触发信号时，时钟分频值被加载为 PSIV 值。

关于浮动预分频器模式与比较模式和捕获模式结合使用的完整描述，请参见 **20.2.14 节**。

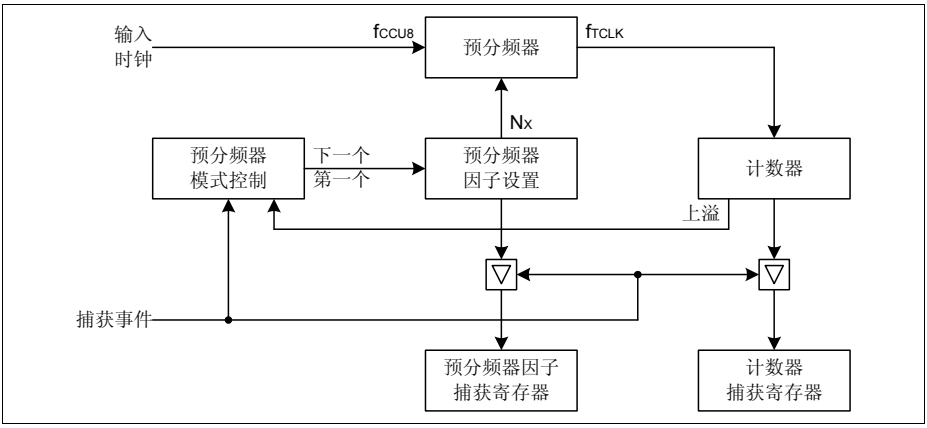


图 20-78　　捕获模式下浮动预分频器概览

20.2.14　CCU8 的使用

20.2.14.1 PWM 信号的产生

CCU8 为占空比的配置提供了一个非常灵活的范围。这个范围是 0 到 100%。

要在边沿对齐模式下产生一个占空比为 100% 的 PWM 信号，应当将比较值 **CC8yCR1**.CR1/**CC8yCR2**.CR2 编程为 0000_H，如**图 20-79** 所示。

采用相同的方法，可以在中心对齐模式产生一个占空比为100%的PWM信号，如**图20-80** 所示。

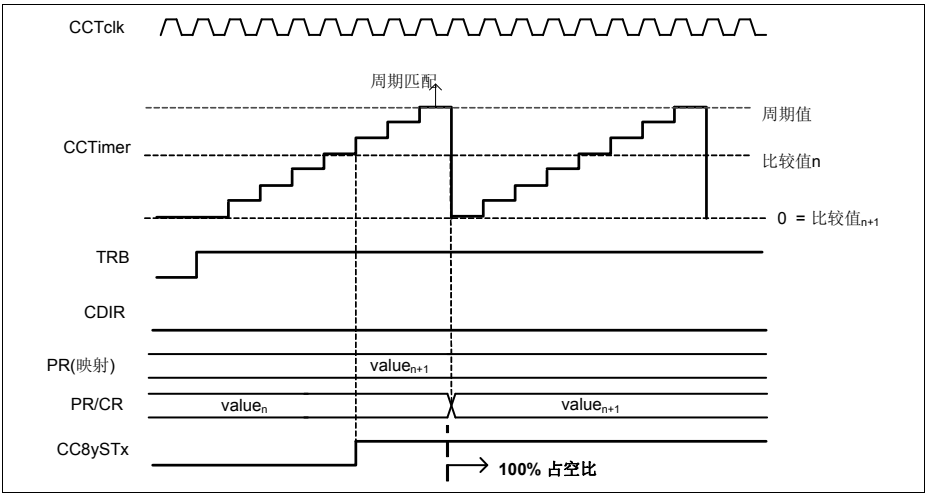


图 20-79　　占空比为 100% 的 PWM 信号 - 边沿对齐模式

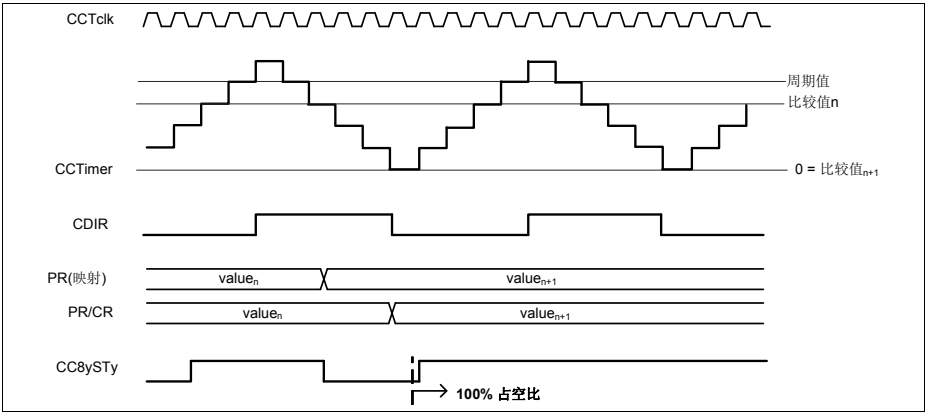


图 20-80　　占空比为 100% 的 PWM 信号 - 中心对齐模式

要在边沿对齐模式下产生一个占空比为 0% 的 PWM 信号，比较寄存器应被设置为周期寄存器的编程值加 1。在使用定时器全部 16 位能力 (从 0 计数到 65535) 的情况下，不可能在比较寄存器中设置一个大于周期值的数值，因此可获得的最小占空比为 1/FS，如图 20-81 所示。

在中心对齐模式，计数器绝不会从 0 计数到 65535_D，因为它必须超过周期寄存器中的设置值一个时钟周期。因此，用户不会有 FS 计数器，这意味着通过在比较寄存器中设置一个大于周期寄存器编程值的数值，将总是能够产生一个占空比为0%的信号，见图20-82。

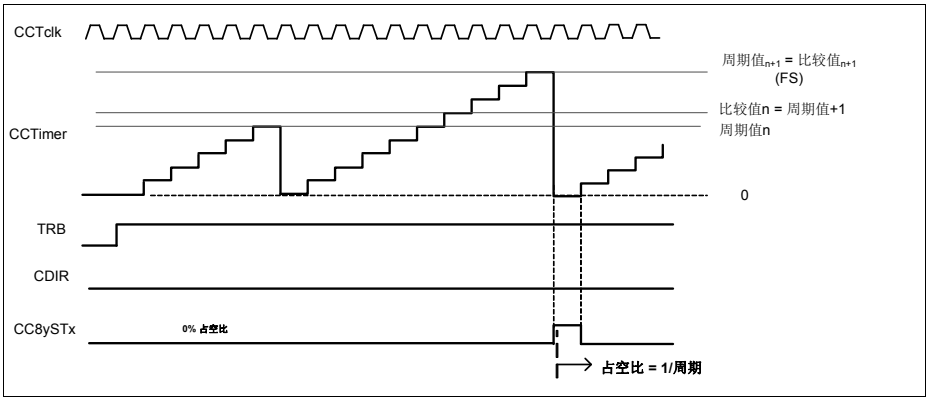


图 20-81　　占空比为 0% 的 PWM 信号 - 边沿对齐模式

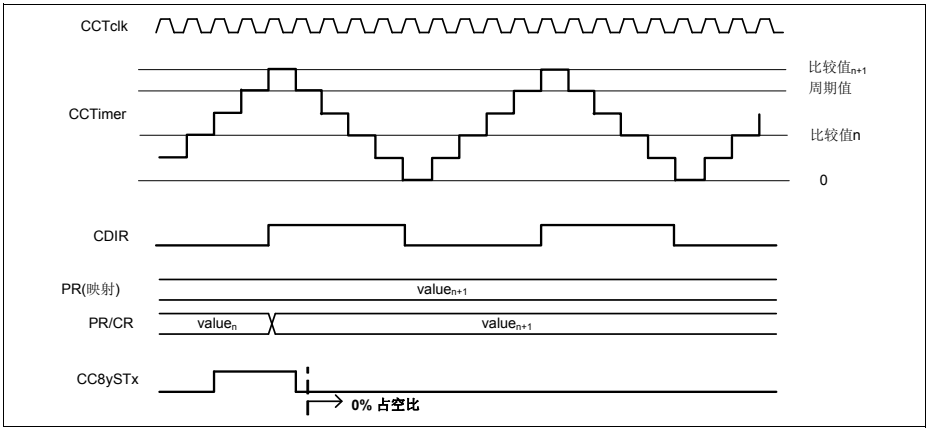


图 20-82　　占空比为 0% 的 PWM 信号 - 中心对齐模式

20.2.14.2 预分频器的使用

在标准预分频器模式，馈送给特定 CC8y 的 f_{tclk} 的频率从**表 20-8** 中进行选择，可通过在 **CC8yPSC**.PSIV 中设置所需要的值来完成。

在浮动预分频器模式，f_{tclk} 的频率可以在一个选定的时段使用**表 20-8** 中的规定值进行修改。在捕获模式，当捕获触发信号的动态性非常缓慢或未知时，该机制特别实用。

在捕获模式，每当发生一次定时器溢出时，浮动预分频器的值增 1，而当发生一个捕获事件时，浮动预分频器的值被设置为初始编程值，如**图 20-83** 所示。

当在捕获模式下使用浮动预分频器模式时，每当发生一次捕获事件时，应当清空定时器，即 **CC8yTC**.CAPC = 11_B。通过将捕获模式与浮动预分频器结合使用，可以只使用一个

CCU8 定时器片，而不需要监视定时器溢出中断事件，即使对于周期性高于 16 位的捕获信号也是如此。为此，在捕获事件产生时，用户只需要知道定时器的捕获值和预分频器的当前配置。这些值包含在每个 CC4yCxV 寄存器中。

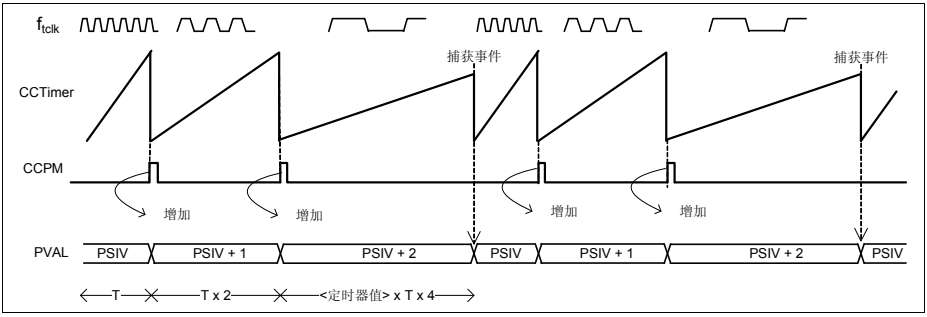


图 20-83　　浮动预分频器捕获模式的使用

当工作在比较模式时，浮动预分频器功能可用来获得一个更精细的 PWM 频率，或用于执行某种频率调制。

每当发生一次定时器上溢 / 下溢 （来自任何一个比较通道）并且预分频器的当前值与寄存器 **CC8yFPC**.PCMP 中的编程值不匹配时，会执行相同的增 1 操作。

当定时器发生一次比较匹配 （来自任何一个比较通道）并且预分频器的当前值等于寄存器 **CC8yFPC**.PCMP 中的编程值时，预分频器的值会在紧接着的下一次定时器上溢 / 下溢发生时刻立即被设置为初始值 **CC8yPSC**.PSIV。

如果要使用一个比较通道来控制浮动预分频器，则必须禁止另一个比较通道。因此，需要将比较值 **CC8yCR1** 或 **CC8yCR2** (取决于使用哪个通道) 设置为一个比周期 **CC8yPR** 大的值。这意味着在边沿对齐模式定时器周期的最大值为 65534_D，因为一个通道的比较值需要被设置为 65535_D(这样一来，相关联通道的比较匹配就被禁止)。

在**图 20-84** 中，浮动预分频器的比较值被设置为 PSIV + 2。每当发生定时器溢出时，该预分频器的值增 1。这意味着，如果给定 f_{tclk} 为 **CC8yPSC**.PSIV 值对应的参考频率，那么，**CC8yPSC**.PSIV + 1 的对应的频率为 $f_{tclk}/2$，**CC8yPSC**.PSIV + 2 对应的频率为 $f_{tclk}/4$。计数器的周期变为：

周期 = $(1/f_{tclk}+2/f_{tclk}+4/f_{tclk})/3$

中心对齐模式使用相同的机制，但是为了保持上升部分和下降部分总是对称，使用定时器下溢而不是上溢，如**图 20-85** 所示。

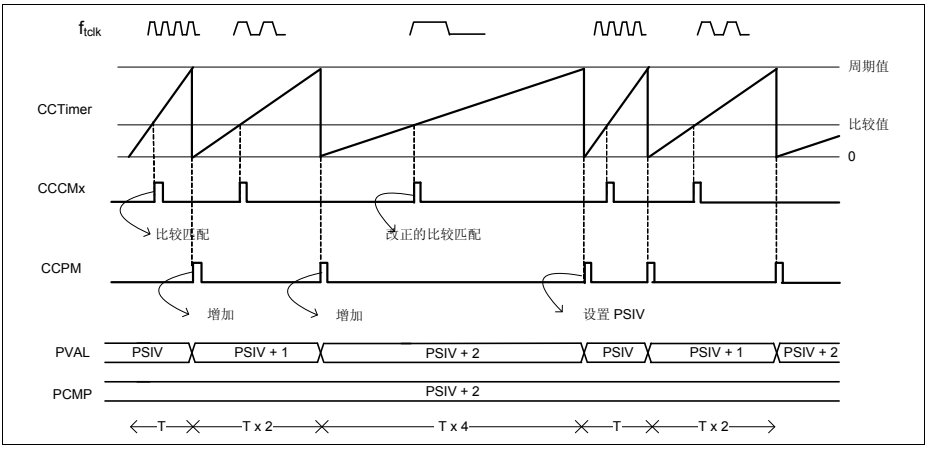


图 20-84　　浮动预分频器比较模式的使用 - 边沿对齐

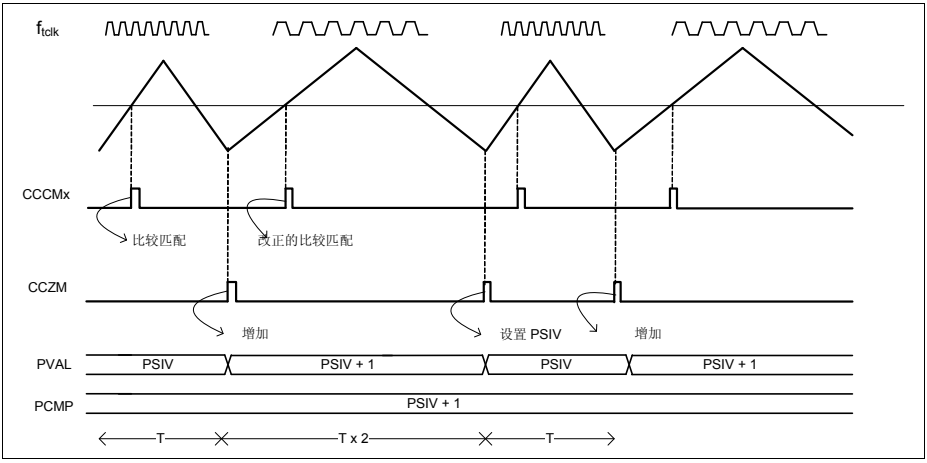


图 20-85　　浮动预分频器比较模式的使用 - 中心对齐

20.2.14.3 PWM 抖动

在比较模式，使用抖动功能可以获得非常精确的输出状态周期。抖动比较寄存器 **CC8yDIT**.DCV 中的设定值与抖动计数器的当前值进行对比检查。每当抖动计数器的值小于比较值时，就会执行下列操作之一：

- 周期值被扩展一个时钟周期 - **CC8yTC**.DITHE = 01_B；边沿对齐模式
- 周期值被扩展两个时钟周期 - **CC8yTC**.DITHE = 01_B；中心对齐模式

- 在向上计数 (**CC8yTCST**.CDIR = 0_B) 期间比较匹配被延迟一个时钟周期 (这意味着状态位停留在置位状态的时间少了一个时钟周期) - **CC8yTC**.DITHE = 10_B
- 周期值被扩展一个时钟周期，在向上计数期间比较匹配被延迟一个时钟周期 - **CC8yTC**.DITHE = 11_B ；边沿对齐模式
- 周期值被扩展两个时钟周期，在向上计数期间比较匹配被延迟一个时钟周期；中心对齐模式

位反转计数器将 **CC8yDIT**.DCV 中的编程值分布在由 16 个定时器周期构成的窗口。

表 20-9 描述了位反转分布与 **CC8yDIT**.DCV 域的编程值之间的关系。标记为 “0” 的域表示在那个计数器周期将要执行的上述操作之一。

表 20-9 位反转分布

	DCV															
抖动计数器	**0**	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**
0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
8	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0
4	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0
C	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
2	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
A	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0
6	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0
E	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
9	1	1	1	1	1	1	1	1	1	1	0	0	0	0	0	0
5	1	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0
D	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
3	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0
B	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	0
7	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
F	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

根据位反转分布与 **CC8yDIT**.DCV 编程值的对应关系，可得到如下的周期值和占空比：

DITHE = 01_B

$$\textit{周期} = [(16 - DCV) \times T + DCV \times (T + 1)]/16; \text{ 边沿对齐模式} \quad (20.10)$$

$$\textit{占空比} = [(16 - DCV) \times d/T + DCV \times (d+1)/(T + 1)]/16; \text{ 边沿对齐模式} \quad (20.11)$$

周期 = [(16 - DCV) x *T* + DCV x (*T* + 2)]/16; 中心对齐模式 (20.12)

占空比 = [(16 - DCV) x *d*/*T* + DCV x (*d*+2)/(*T* + 2)]/16; 中心对齐模式 (20.13)

DITHE = 10_B

周期 = *T* ; 边沿对齐模式 (20.14)

占空比 = [(16 - DCV) x *d*/*T* + DCV x (*d*-1)/*T*]/16; 边沿对齐模式 (20.15)

周期 = *T* ; 中心对齐模式 (20.16)

占空比 = [(16 - DCV) x *d*/*T* + DCV x (*d*-1)/*T*]/16; 中心对齐模式 (20.17)

DITHE = 11_B

周期 = [(16 - DCV) x *T* + DCV x (*T* + 1)]/16; 边沿对齐模式 (20.18)

占空比 = [(16 - DCV) x *d*/*T* + DCV x *d*/(*T* + 1)]/16; 边沿对齐模式 (20.19)

周期 = [(16 - DCV) x *T* + DCV x (*T* + 2)]/16; 中心对齐模式 (20.20)

占空比 = [(16 - DCV) x *d*/*T* + DCV x (*d*+1)/(*T* + 2)]/16; 中心对齐模式 (20.21)

其中：

T - 信号的初始周期，详见 **20.2.5.1 节**

d - 信号的初始占空比，详见 **20.2.5.1 节**

20.2.14.4 捕获模式的使用

每个定时器片可以使用 2 个或 4 个捕获寄存器。只使用两个捕获寄存器意味着仅有一个事件与一个捕获触发信号连接。要使用 4 个捕获寄存器，需要将两个捕获触发信号映射到一个事件（可以是同一个信号的不同边沿，也可以是两个不同信号），或需要将 **CC8yTC**.SCE 域置 1 以连接这 4 个捕获寄存器。

对于捕获触发信号 1 或捕获触发信号 0，用于捕获的内部定时器片机制是相同的。

不同的捕获事件 - SCE = 0_B

捕获触发信号 1 (CCcapt1) 被指定用于捕获寄存器 2 (**CC8yC2V**) 和捕获寄存器 3 (**CC8yC3V**)，而捕获触发信号 0 (CCcapt0) 被指定用于捕获寄存器 1 (**CC8yC1V**) 和捕获寄存器 0 (**CC8yC0V**)。

每当发生 CCcapt0 事件时，定时器值被保存到捕获寄存器 **CC8yC1V**，而捕获寄存器 **CC8yC1V** 的值被传送到捕获寄存器 **CC8yC0V**。

每当发生 CCcapt1 事件时，定时器值被保存到捕获寄存器 **CC8yC3V**，而捕获寄存器 **CC8yC3V** 的值被传送到捕获寄存器 **CC8yC2V**。

捕获 / 传送机制只能发生在特定寄存器未满的情况下。当接收到一个新值时，寄存器变为满状态，而在软件读回该值后，寄存器变为空状态。

每当软件回读 **CC8yC0V**、**CC8yC1V**、**CC8yC2V** 或 **CC8yC3V** 寄存器时，满标志即被清除。通过使能一个连接到特定事件的中断源，可以告知软件有一个新的捕获触发信号。这意味着每当发生一次捕获时就会产生一个中断脉冲。

在使用浮动预分频器模式的情况下，时钟分频的当前值也保存在捕获寄存器 (CC8yCxV) 中。

图 20-86 给出了如何在定时器片中使用捕获 / 传送的一个示例，该定时器片使用一个外部信号作为计数功能 (用来测量一个旋转设备的速度)，使用一个等距的捕获触发信号来记录速度计算需要的时间戳 (绘制了两个定时器波形，一个用来说明每次捕获事件中的定时器清除操作，另一个是清除功能无效的情况) 。

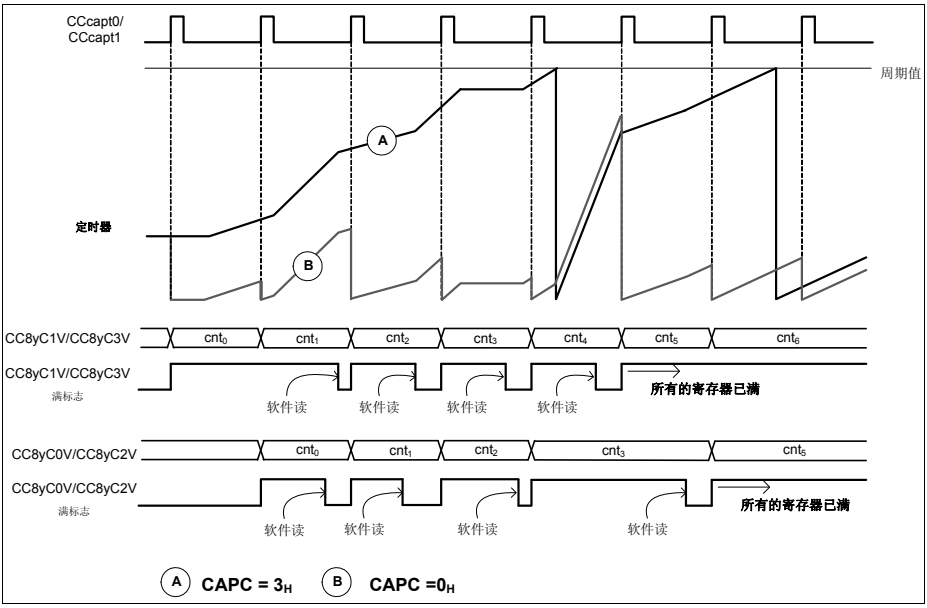


图 20-86　　捕获模式的使用 - 单通道

相同的捕获事件 - SCE = 1_B

如果 **CC8yTC**.SCE 被设置为 1_B，则所有 4 个捕获寄存器都被链接到一起，模拟一个深度 4 的 FIFO 结构。在这种情况下，仅捕获触发信号 1 (CCcapt1) 用于执行一次捕获事件。

作为该模式的一个例子，可以考虑这样一种情况：一个定时器片工作在捕获模式， SCE = 1_B，使用另一个外部信号来控制计数。该定时器片可以根据计数信号的频率以不同的速度递增。

使用另一个定时器片来控制捕获触发信号，记录捕获的时间戳。

从**图 20-87** 中可以看到这种简单机制工作过程。定时器片 0 的 CC80ST 输出被用作定时器片 CC81 的捕获触发信号 (在上升沿和下降沿都有效)。CC80ST 输出被用作已知的时基标记，而用于捕获操作的定时器片由外部事件 (例如，外部计数) 来控制。

由于有4个可用的捕获寄存器，每当软件回读完整的数值集时，可以测量出3个速度廓线。

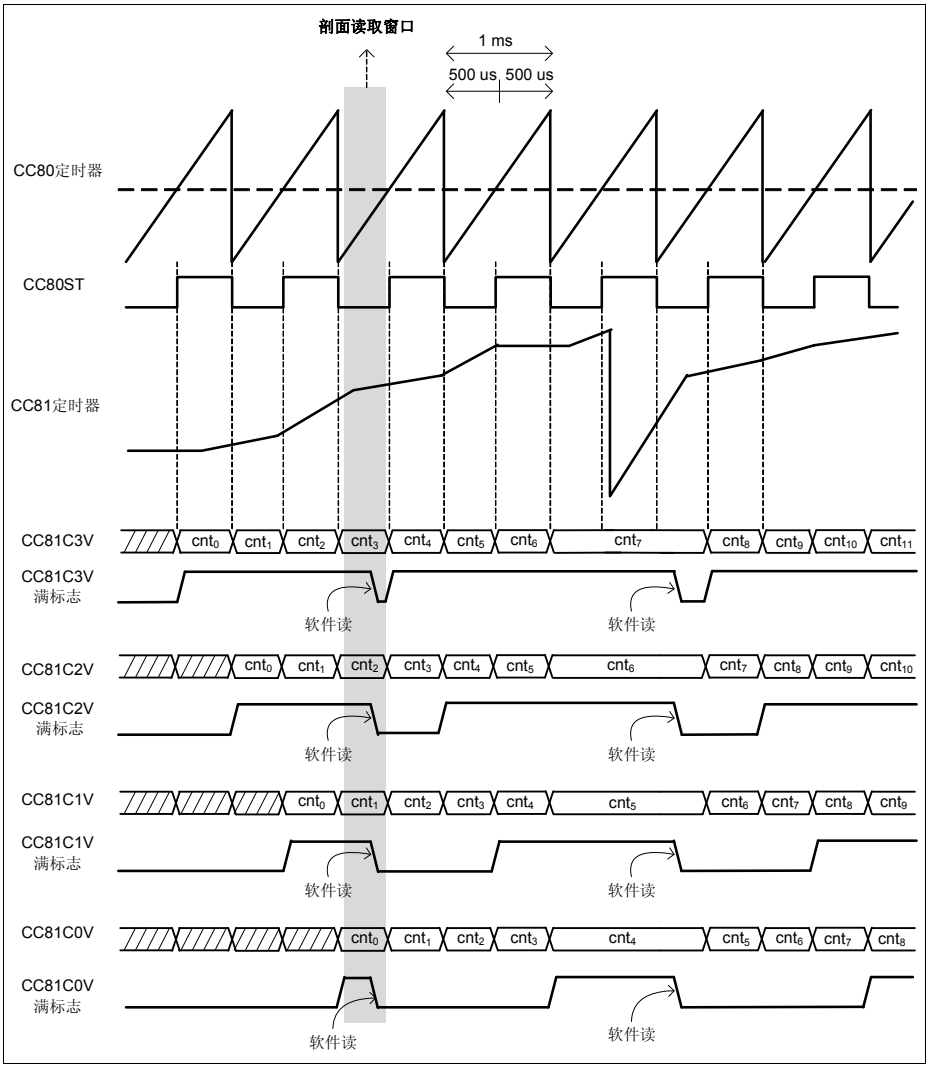


图 20-87　　三个捕获廓线 - CC8yTC.SCE = 1_B

要计算图 20-87 中的三个不同廓线，需要在指定的读窗口期间读取 4 个捕获寄存器。此后，即可完成廓线计算：

廓线 1 = $CC81C1V_{info}$ - $CC81C0V_{info}$

廓线 2 = $CC81C2V_{info}$ - $CC81C1V_{info}$

廓线 3= $CC81C3V_{info}$ - $CC81C2V_{info}$

注： 这仅是一个示例，还可以实现更多的定时器配置和软件回路。

高动态捕获

在有些情况下，捕获触发信号的动态性可能随着时间变化很大。这就要求软件需要为最坏的情况做好准备，例如捕获触发信号的频率可能会非常高。在需要逐个周期进行计算 (在每次发生捕获触发时计算) 的应用场合，这个限制需要由软件来满足。然而，对于不需要每个周期都进行计算的应用，软件可以周期性地回读 FIFO 数据寄存器并读取目前为止所有已经捕获到的数据，如**图 20-88** 所示。

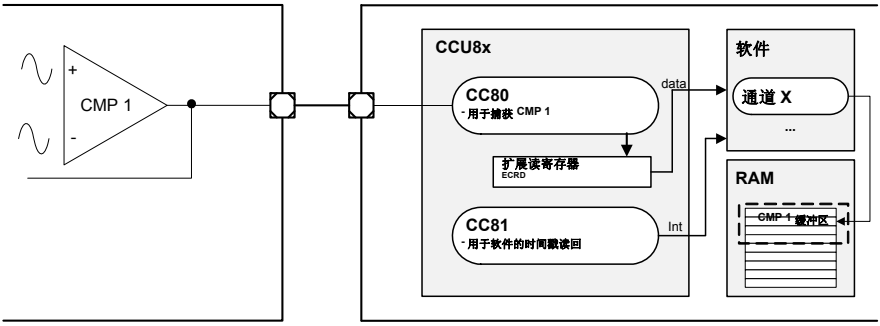


图 20-88 高动态捕获 - 使用软件控制的时间戳

在这种情况下，每当时间戳定时器 (该定时器的周期性也可以在运行时调整) 触发一个中断时，软件会回读完整的捕获寄存器组 (2 个或是 4 个，取决于所选择的配置)。

由于每个捕获寄存器都提供一个满标志状态位，软件总是可以回读完整的捕获寄存器集。在进行数据处理时，软件要检查这个满标志，以确认该值是否需要被处理。

这个 FIFO 回读功能也可以用于一些对系统产生高负荷的应用场合，但不能保证回读捕获数据的访问时间是固定的。

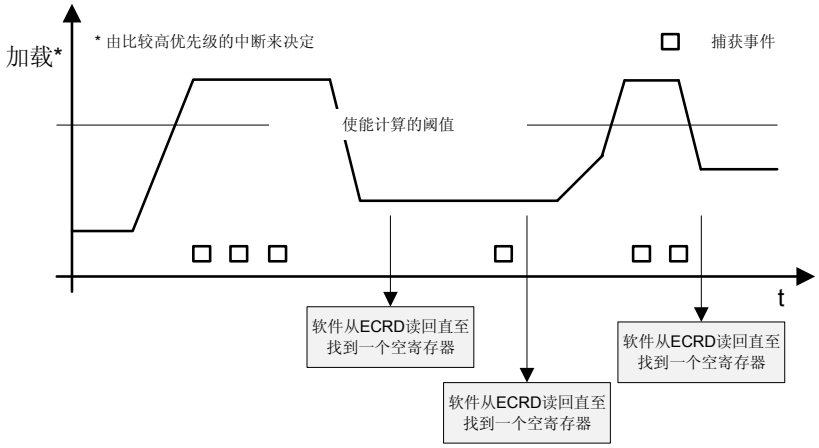


图 20-89　　高负荷期间的扩展回读

捕获分组

在需要多个捕获定时器并且捕获程序的优先级并不意味着对每个事件都需要进行逐个周期计算的应用场合，将同一个 CCU8x 单元中的所有定时器进行分组可能会更为合适，如图 20-90 所示。

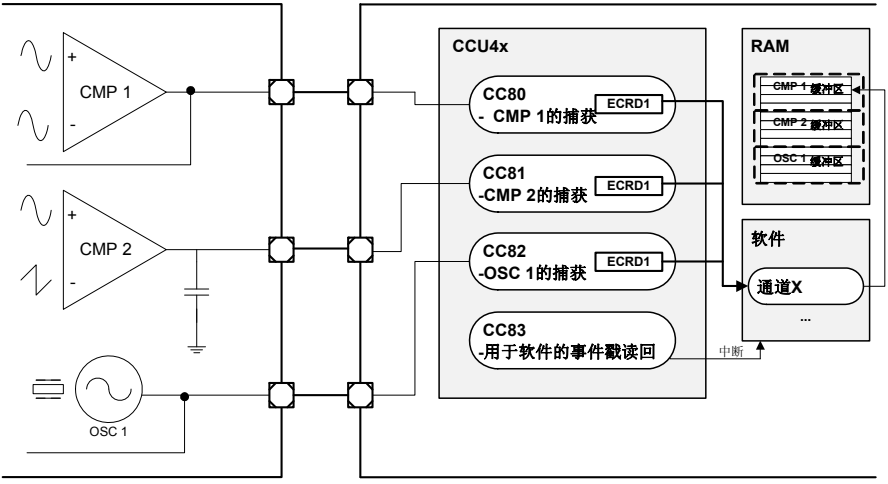


图 20-90　　采用扩展回读的捕获分组

通过将用于捕获的定时器片的 ECM 位域置 1，可使扩展回读模式总能以正确的捕获顺序 (从最老的数据到最新的数据) 回读数据。然后，使用一个时间戳定时器来触发 CPU/ 软件去回读定时器片中所有捕获到的数据。

每当检测到中断时， CPU/ 软件 (在本例中) 会回读所有定时器片的完整捕获寄存器集 (通过 ECRD 地址)。由于针对每个数据读取都有一个满标志指示器，所以 CPU/ 软件能够从所有定时器片回读完整的捕获寄存器集。这就允许固定的存储器分配，使分配的存储器与捕获寄存器的数量相同，如图 20-91 所示 (在本例中，每个定时器片使用 4 个捕获寄存器)。

每个 ECRD 数据的首部有一个附加的丢失值位域 (LCV)，它指示在两次读操作之间是否有任何捕获触发信号丢失(如果捕获触发信号比回读数据的程序快，就会发生这种情况)。

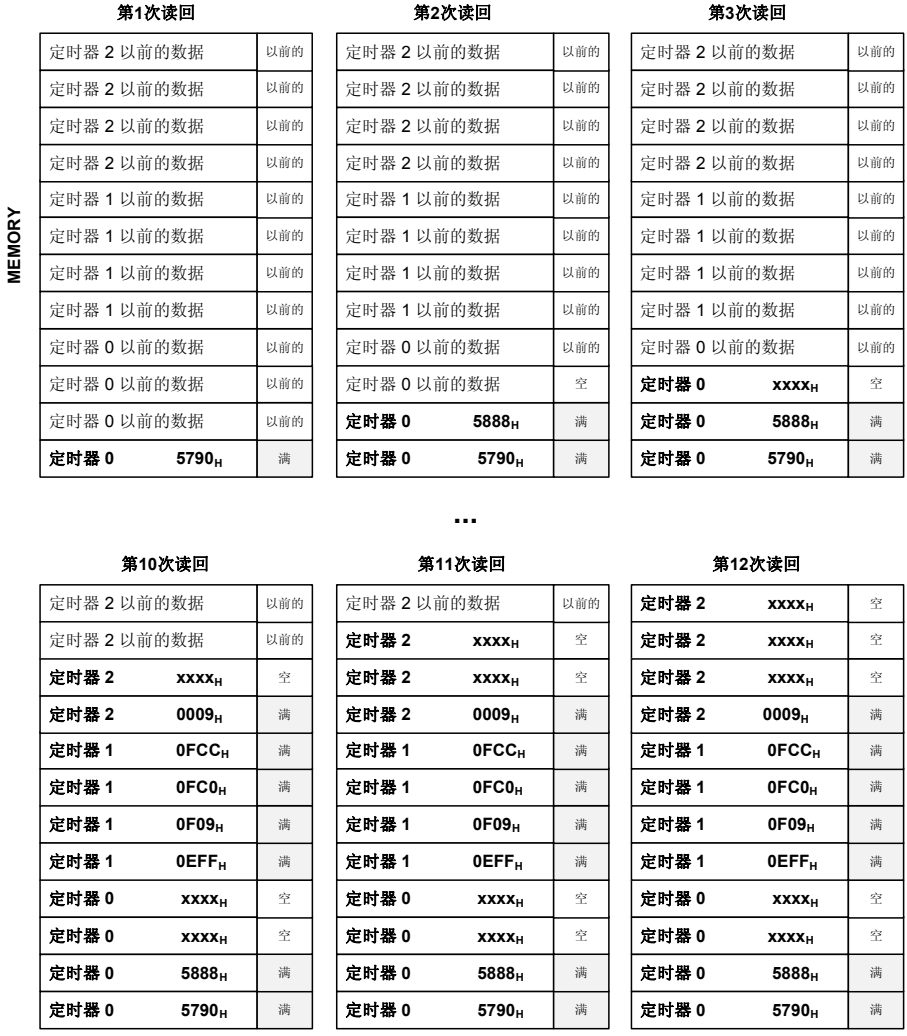


图 20-91　　扩展回读的存储器结构

20.2.14.5 奇偶校验器的使用

CCU8 的奇偶校验器功能使用一个 CCU4 定时器片来控制输出更新与随之产生的外部开关 / 驱动器更新之间的延迟。

该 CCU8 定时器片被配置为工作在边沿对齐的单次模式，并被配置为使用外部清空和启动功能。该功能被连接到奇偶校验器的更新输出信号 CCU8x.IGBTO。定时器片的比较和周期值需要根据 CCU8 的输出与随之发生的驱动器 / 开关奇偶输出之间的预知延迟来编程。 CCU8、 CCU4 和外部硬件之间的连接如**图 20-92** 所示。

图 20-93 示出了使用奇偶校验器的一个例子的时序波形 (使用偶校验，**GPCHK**.PCTS 域取默认值)。在这个例子中，为了避免图形过于复杂，仅使用了 CCU8 的两个输出。

所选输出的每次更新都会引起对 **GPCHK**.PCST (奇偶校验器状态) 值的修改，或者换句话说，每当 XOR 链的结果发生改变时，会在 CCU8x.IGBTO 产生一个触发信号。如前所述，该信号被用作一个 CCU4 定时器片的清空和启动信号。

当 CCU4 定时器片达到比较值时，相应的状态输出 CCU4x.STy 变为有效。经过这段延时之后， CCU8x.GPy1 的值 (来自外部硬件的奇偶值) 与内部 XOR 链 (**GPCHK**.PCST) 的结果进行对比检查。如果两个值不同，会产生一个服务请求脉冲，前提是已经使能了该功能。

注意，当一次 CCU8 输出更新导致一个与 XOR 链相等的结果时，CCU4 定时器片不被触发 (来自外部硬件的奇偶输出保持相同的值)。

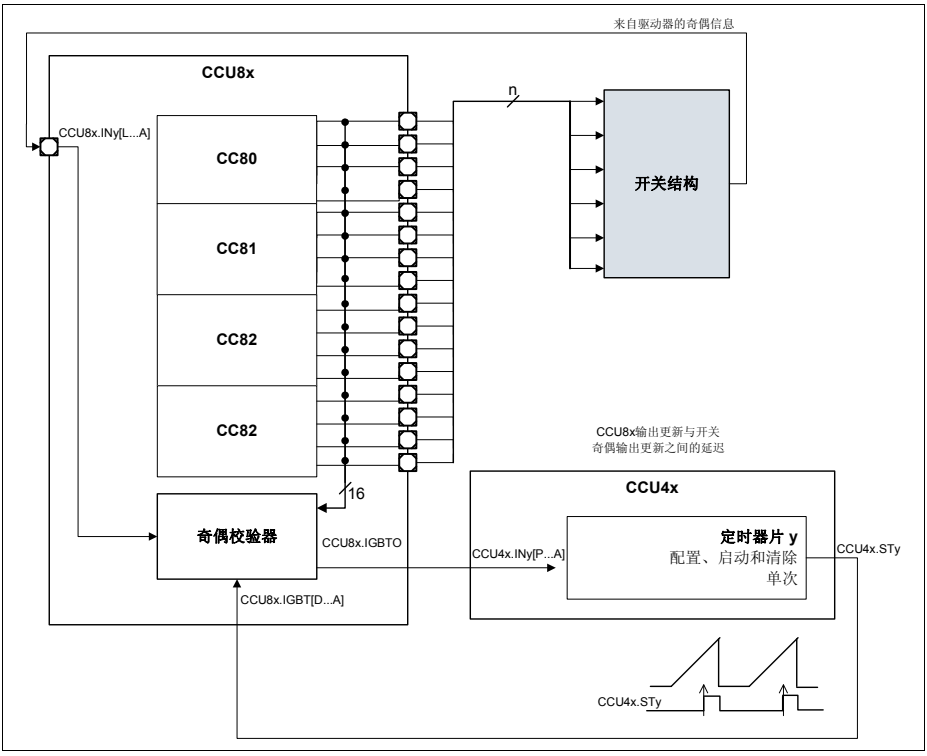


图 20-92　　奇偶校验器连接

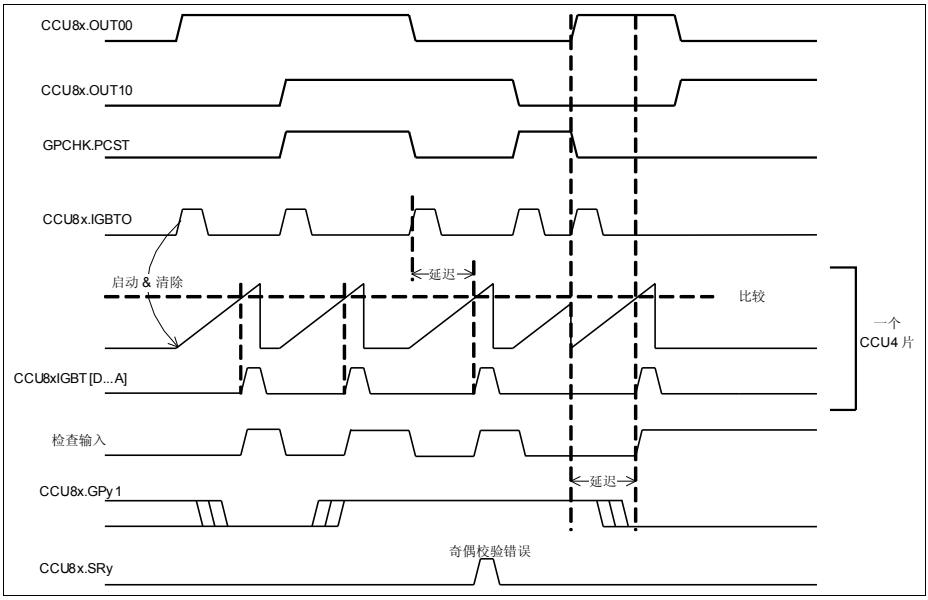


图 20-93　　奇偶校验器时序示例

20.3　　服务请求的产生

每个 CCU8 定时器片都有一个如**图 20-94** 所示的中断结构。寄存器 **CC8yINTS** 是中断源的状态寄存器。通过分别写 **CC8ySWS** 和 **CC8ySWR** 寄存器中的特定位，可以用软件置位或清除每个专用的中断源。

通过寄存器 **CC8yINTE** 可以使能或禁止每个中断源。一个被使能的中断源总是会在服务请求线上产生一个脉冲，即使特定状态位未被清零。**表 20-10** 描述了每个定时器片的中断源。

向上计数期间的周期匹配和向下计数时的 1 匹配这两个中断源被或在一起。该机制同样适用于两个通道的向上计数期间的比较匹配和向下计数期间的比较匹配。

外部事件的中断源根据 **CC8yINS**.EVxEM 中设定的配置直接连接。如果一个事件被编程为在两个边沿都有效，则当检测到外部信号的任何跳变时都将产生中断服务请求脉冲。如果该事件被连接到一个电平功能， **CC8yINS**.EVxEM 仍可被编程为使能服务请求脉冲。 当定时器片进入陷阱状态时，陷阱事件不需要任何额外的配置即可产生服务请求脉冲。

表 20-10 中断源

信号	描述
CCINEV0_E	来自事件选择器的事件 0 边沿信息。在外部信号应该触发中断时使用。
CCINEV1_E	来自事件选择器的事件 1 边沿信息。在外部信号应该触发中断时使用。(也可以是奇偶校验器序列错误信息，如果奇偶校验器被使能)。
CCINEV2_E	来自事件选择器的事件 2 边沿信息。在外部信号应该触发中断时使用。
CCPM_U	向上计数期间的周期匹配。
CCCM1_U	向上计数期间来自通道 1 的比较匹配。
CCCM1_D	向下计数期间来自通道 1 的比较匹配。
CCCM2_U	向上计数期间来自通道 2 的比较匹配。
CCCM2_D	向下计数期间来自通道 2 的比较匹配。
CCOM_D	向下计数期间的 1 匹配
陷阱状态置位	进入陷阱状态。会置位 E2AS。

然后，每个中断事件都可以被转发到定时器片的 4 个服务请求线之一，如图 20-95 所示。**CC8ySRS** 的设置值控制将哪一个中断事件映射到哪一条服务请求线。

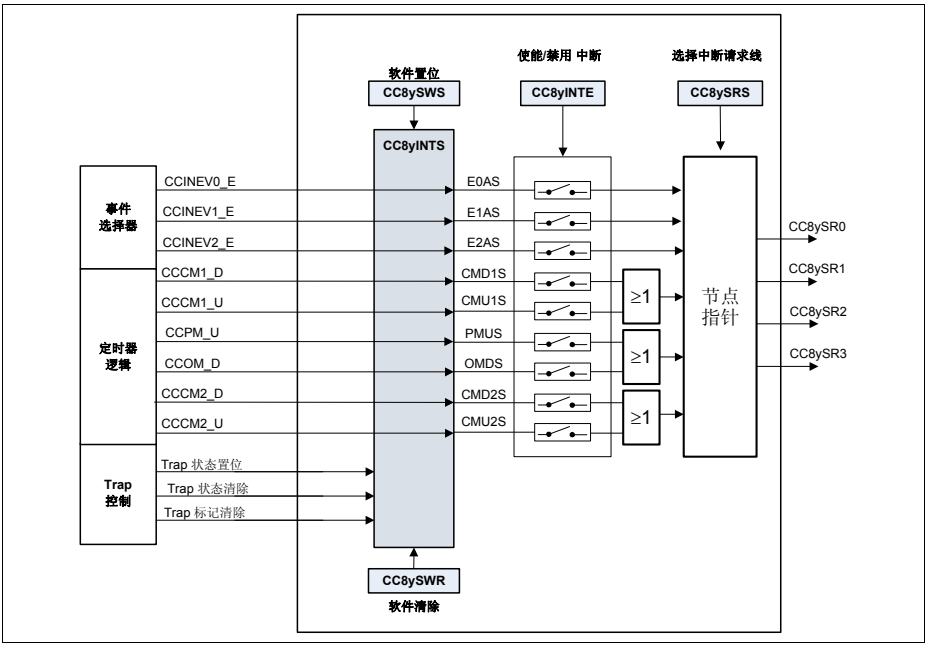


图 20-94　　定时器片中断节点指针概览

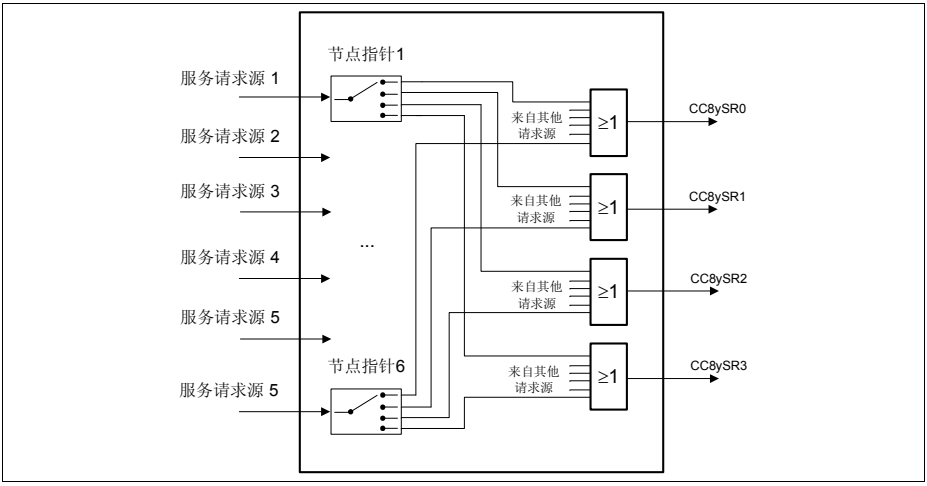


图 20-95　　定时器片中断选择器概览

在 CCU8 的内核中，每个定时器片的 4 条服务请求线被或在一起，如图 20-96 所示。这意味着每个 CCU8 仅有 4 条服务请求线，可以使每条线都有来自不同定时器片的中断请求。

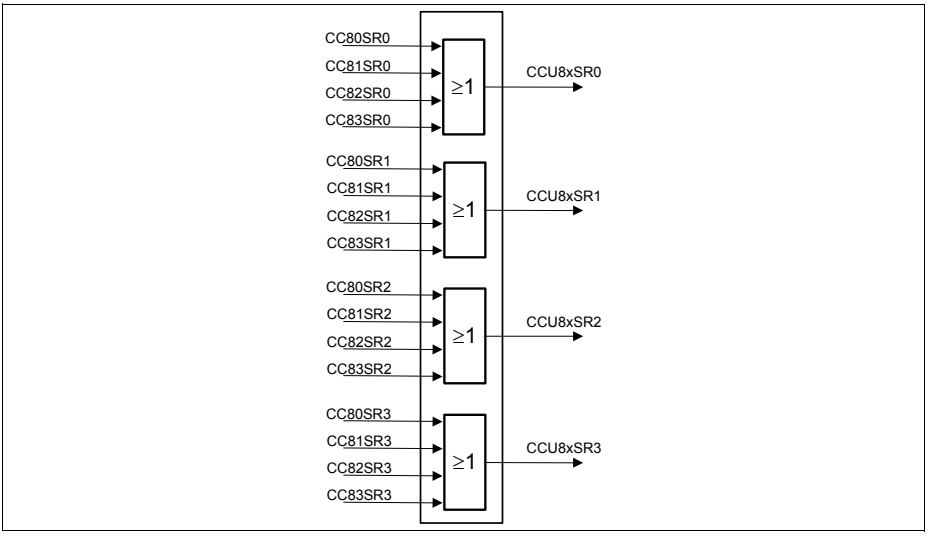


图 20-96　　CCU8 服务请求概览

20.4　　调试行为

在挂起模式，所有定时器片及预分频器的功能时钟都被停止。寄存器仍可被 CPU 访问 (只读)。该模式对于调试目的非常有用，例如，为了获得内部值的快照应冻结当前器件的状态。在挂起模式，所有定时器片的计数器都被停止。挂起模式对于寄存器位来说是非侵入式的。这意味着当进入或离开挂起模式时，寄存器位不会被硬件修改。

可以在内核级通过域 **GCTRL**.SUSCFG 来配置进入挂起模式。

CCU8 模块仅在挂起模式信号变为无效后才能工作。

20.5　　电源、复位和时钟

下面的小节描述 CCU8 的工作条件、特性和时序要求。所有时序信息都基于模块时钟 f_{ccu8}。

20.5.1　　时钟

模块时钟

CCU8 的模块时钟在 SCU 一章中被描述为 f_{PCLK}。

CCU8 模块的总线接口时钟在 SCU 一章中被描述为 f_{MCLK}。

通过 **GSTAT** 寄存器可以禁止 CCU8 的模块时钟，不过，在不同 CCU8 实例中也许存在对 f_{ccu8} 的依赖。关于产品时钟方案的详细描述，请参见 SCU 一章。

如果不同的 IP 实例中存在对模块时钟的依赖，可以通过禁止预分频器 (**GSTAT**.PRB = 0_B) 来禁止特定 CCU8 内部的模块时钟。

外部时钟

可以使用一个外部时钟作为预分频器的时钟源，因而也是所有定时器片 CC8y 的时钟源。该外部时钟源可以被连接到一个 CCU8x.CLK[C...A] 输入。

该外部信号源仍然与 f_{ccu8} 同步。

表 20-11　外部时钟工作条件

参数	符号	值			单位	备注 / 测试条件
		最小值	典型值	最大值		
Frequency	f_{eclk}	–	–	$f_{ccu8}/4$	MHz	
ON time	ton_{eclk}	$2T_{ccu8}$ 1)2)	–	–	ns	
OFF time	$toff_{eclk}$	$2T_{ccu8}$ 1)2)	–	–	ns	仅使用上升沿

1) 仅在该信号之前与 fccu4 时钟 (或一个同步时钟) 不同步或不是由 ccu4 时钟产生时有效

2) 不强求占空比为 50%

20.5.2　电源

CCU8 位于内核电源域内，因此对于上电或下电顺序不需要做任何特殊考虑。关于对不同电源域的解释，请参见 SCU (系统控制单元) 一章。

通过禁止 CCU8 自身的内部时钟，可以实现 CCU8 的内部断电模式。为此，应将 **GSTAT** 寄存器设置为默认的复位值 (通过空闲模式设置寄存器 **GIDLS** 来实现)。

20.6　初始化和系统依赖

20.6.1　初始化序列

使用 CCU8 的应用程序应当采用如下的初始化序列：

第 1 步：通过特定的 SCU 寄存器 CGATCLR0 来使能 CCU8 时钟。

第 2 步：通过向 **GIDLC**.SPRB 域写 1_B 来使能预分频器时钟。

第 3 步：配置全局 CCU8 寄存器 **GCTRL**。

第 4 步：配置与所需定时器片功能相关的所有寄存器，包括中断 / 服务请求配置。

第 5 步：如果需要，为一个定时器片的一个特定比较通道状态配置起始值，通过向特定的 **GCSS**.SyTS 写 1_B 来完成。

第 6 步：配置奇偶校验器功能 （如果使用），通过对 **GPCHK** 寄存器编程实现。

第 7 步：通过向特定的 **GIDLC**.CSyI 写 1_B 来使能特定的定时器片 CC8y。

第 8 步：对于应该由软件来同步启动的所有定时器片，应寻址位于 SCU 中的特定系统寄存器 CCUCON 来使能定时器的同步启动。在此之前，需要将 SCU.GLCSTxx 输入信号配置为启动功能，详见 **20.2.8.1 节**。

20.6.2 系统相关性

每个 CCU8 都可能与模块时钟和总线时钟频率有不同的相关性。有关这种相关性的详细描述请参见 SCU 和系统架构这两章。

就不同的时钟工作频率而言，多个外设之间也可能存在相关性。在配置 CCU8 与其他外设之间的连接之前，应先处理好这种相关性。

为了更好的 CCU8 和系统操作，必须要考虑以下几点：

- CCU8 的模块时钟最多是模块总线接口时钟的两倍。
- CCU8 的模块输入触发信号不能超过模块时钟频率 (如果触发信号在器件内部产生)。
- CCU8 的模块输入触发信号不能超过 **20.5.1 节**所述的频率。
- 用作其他模块之触发信号 / 功能的 CCU8 输出的频率必须要在端点处进行交叉检查。
- 复位时使用和移除 CCU8 可能会在其他模块中引起不希望的操作。当模块将 CCU8 输出用作触发信号 / 功能时就可能发生这种情况。

20.7 寄存器

寄存器概述

寄存器的绝对地址通过下面的加法计算：

模块基地址 + 偏移地址

表 20-12 寄存器地址空间

模块	基地址	结束地址	备注
CCU80	50000000_H	$5000FFFF_H$	

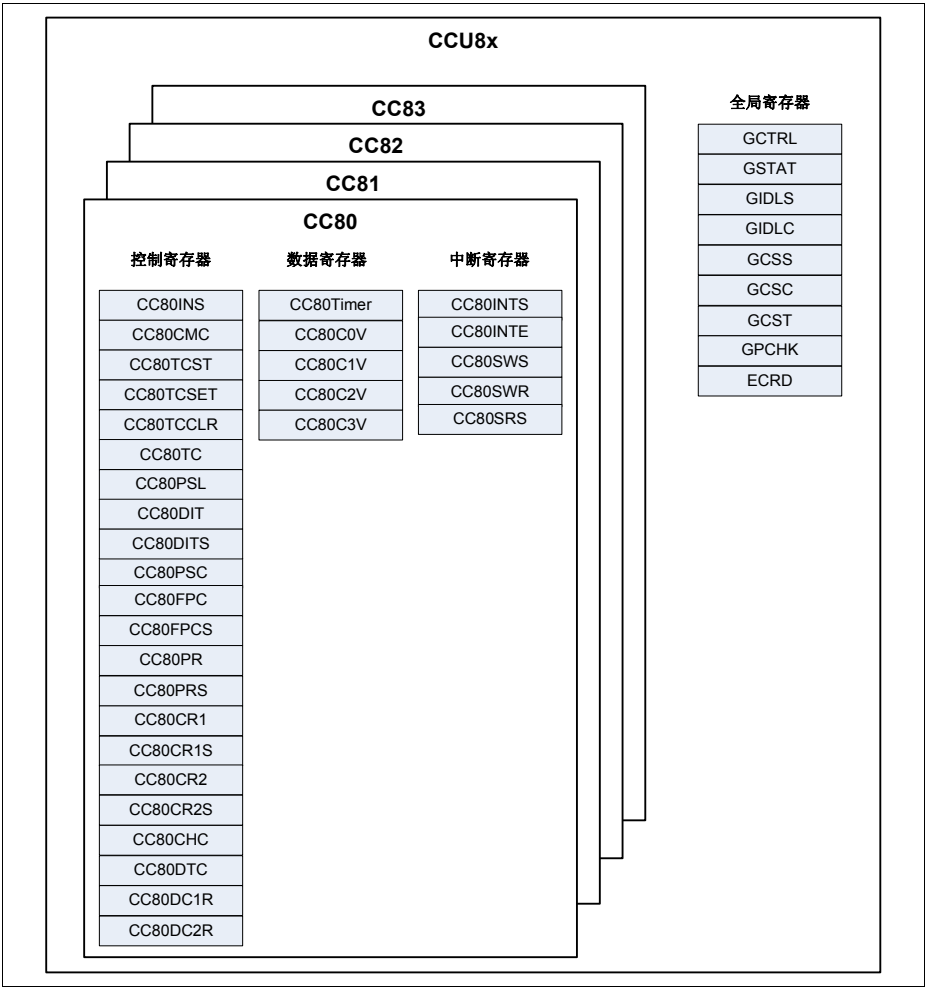


图 20-97　　CCU8 寄存器概览

表 20-13 CCU8 寄存器一览表

简称	描述	偏移地址[1)]	访问方式		描述见
			读	写	
CCU8 全局寄存器					
GCTRL	模块综合控制寄存器	0000_H	U, PV	U, PV	页 20-104
GSTAT	通用定时器片状态寄存器	0004_H	U, PV	BE	页 20-107
GIDLS	通用空闲使能寄存器	0008_H	U, PV	U, PV	页 20-108
GIDLC	通用空闲禁止寄存器	$000C_H$	U, PV	U, PV	页 20-110
GCSS	通用通道置位寄存器	0010_H	U, PV	U, PV	页 20-111
GCSC	通用通道清除寄存器	0014_H	U, PV	U, PV	页 20-114
GCST	通用通道状态寄存器	0018_H	U, PV	BE	页 20-117
GPCHK	奇偶校验器配置寄存器	$001C_H$	U, PV	U, PV	页 20-119
MIDR	模块标识寄存器	0080_H	U, PV	BE	页 20-121
CC80 寄存器					
CC80INS	输入选择器单元配置	0100_H	U, PV	U, PV	页 20-121
CC80CMC	连接矩阵配置	0104_H	U, PV	U, PV	页 20-124
CC80TCST	定时器运行状态	0108_H	U, PV	BE	页 20-126
CC80TCSET	定时器运行置位	$010C_H$	U, PV	U,PV	页 20-127
CC80TCCLR	定时器运行清除	0110_H	U, PV	U, PV	页 20-128
CC80TC	通用定时器配置	0114_H	U, PV	U, PV	页 20-129
CC80PSL	输出被动电平配置	0118_H	U, PV	U, PV	页 20-134
CC80DIT	抖动配置	$011C_H$	U, PV	BE	页 20-135
CC80DITS	抖动映射寄存器	0120_H	U, PV	U, PV	页 20-135
CC80PSC	预分频器配置	0124_H	U, PV	U, PV	页 20-136
CC80FPC	预分频器比较值	0128_H	U, PV	U, PV	页 20-137
CC80FPCS	预分频器映射比较值	$012C_H$	U, PV	U, PV	页 20-138
CC80PR	定时器周期值	0130_H	U, PV	BE	页 20-138
CC80PRS	定时器周期映射值	0134_H	U, PV	U, PV	页 20-139
CC80CR1	通道 1 的定时器比较值	0138_H	U, PV	BE	页 20-140
CC80CR1S	通道 1 的定时器比较映射值	$013C_H$	U, PV	U, PV	页 20-140

表 20-13 CCU8 寄存器一览表 (续表)

简称	描述	偏移地址[1]	访问方式		描述见
			读	写	
CC80CR2	通道 2 的定时器比较值	0140_H	U, PV	BE	页 20-141
CC80CR2S	通道 2 的定时器比较映射值	0144_H	U, PV	U, PV	页 20-142
CC80CHC	通道控制	0148_H	U, PV	U, PV	页 20-143
CC80DTC	死区时间控制	$014C_H$	U, PV	U, PV	页 20-144
CC80DC1R	通道 1 死区时间计数器值	0150_H	U, PV	U, PV	页 20-145
CC80DC2R	通道 2 死区时间计数器值	0154_H	U, PV	U, PV	页 20-146
CC80TIMER	定时器当前值	0170_H	U, PV	U, PV	页 20-147
CC80C0V	捕获寄存器 0 值	0174_H	U, PV	BE	页 20-147
CC80C1V	捕获寄存器 1 值	0178_H	U, PV	BE	页 20-148
CC80C2V	捕获寄存器 2 值	$017C_H$	U, PV	BE	页 20-149
CC80C3V	捕获寄存器 3 值	0180_H	U, PV	BE	页 20-150
CC80INTS	中断状态	$01A0_H$	U, PV	BE	页 20-151
CC80INTE	中断使能	$01A4_H$	U, PV	U, PV	页 20-153
CC80SRS	中断配置	$01A8_H$	U, PV	U, PV	页 20-155
CC80SWS	中断状态置位	$01AC_H$	U, PV	U, PV	页 20-156
CC80SWR	中断状态清除	$01B0_H$	U, PV	U, PV	页 20-158
CC80STC	映射传送控制	$01B4_H$	U, PV	U, PV	页 20-159
CC80ECRD0	扩展回读 0	$01B8_H$	U, PV	BE	页 20-160
CC80ECRD1	扩展回读 1	$01BC_H$	U, PV	BE	页 20-162
CC81 寄存器					
CC81INS	输入选择器单元配置	0200_H	U, PV	U, PV	页 20-121
CC81CMC	连接矩阵配置	0204_H	U, PV	U, PV	页 20-124
CC81TCST	定时器运行状态	0208_H	U, PV	BE	页 20-126
CC81TCSET	定时器运行置位	$020C_H$	U, PV	U,PV	页 20-127
CC81TCCLR	定时器运行清除	0210_H	U, PV	U, PV	页 20-128
CC81TC	通用定时器配置	0214_H	U, PV	U, PV	页 20-129
CC81PSL	输出被动电平配置	0218_H	U, PV	U, PV	页 20-134
CC81DIT	抖动配置	$021C_H$	U, PV	BE	页 20-135
CC81DITS	抖动映射寄存器	0220_H	U, PV	U, PV	页 20-135

表 20-13 CCU8 寄存器一览表 (续表)

简称	描述	偏移地址[1]	访问方式		描述见
			读	写	
CC81PSC	预分频器配置	0224_H	U, PV	U, PV	页 20-136
CC81FPC	预分频器比较值	0228_H	U, PV	U, PV	页 20-137
CC81FPCS	预分频器映射比较值	$022C_H$	U, PV	U, PV	页 20-138
CC81PR	定时器周期值	0230_H	U, PV	BE	页 20-138
CC81PRS	定时器周期映射值	0234_H	U, PV	U, PV	页 20-139
CC81CR1	通道 1 的定时器比较值	0238_H	U, PV	BE	页 20-140
CC81CR1S	通道 1 的定时器比较映射值	$023C_H$	U, PV	U, PV	页 20-140
CC81CR2	通道 2 的定时器比较值	0240_H	U, PV	BE	页 20-141
CC81CR2S	通道 2 的定时器比较映射值	0244_H	U, PV	U, PV	页 20-142
CC81CHC	通道控制	0248_H	U, PV	U, PV	页 20-143
CC81DTC	死区时间控制	$024C_H$	U, PV	U, PV	页 20-144
CC81DC1R	通道 1 死区时间计数器值	0250_H	U, PV	U, PV	页 20-145
CC81DC2R	通道 2 死区时间计数器值	0254_H	U, PV	U, PV	页 20-146
CC81TIMER	定时器当前值	0270_H	U, PV	U, PV	页 20-147
CC81C0V	捕获寄存器 0 值	0274_H	U, PV	BE	页 20-147
CC81C1V	捕获寄存器 1 值	0278_H	U, PV	BE	页 20-148
CC81C2V	捕获寄存器 2 值	$027C_H$	U, PV	BE	页 20-149
CC81C3V	捕获寄存器 3 值	0280_H	U, PV	BE	页 20-150
CC81INTS	中断状态	$02A0_H$	U, PV	BE	页 20-151
CC81INTE	中断使能	$02A4_H$	U, PV	U, PV	页 20-153
CC81SRS	中断配置	$02A8_H$	U, PV	U, PV	页 20-155
CC81SWS	中断状态置位	$02AC_H$	U, PV	U, PV	页 20-156
CC81SWR	中断状态清除	$02B0_H$	U, PV	U, PV	页 20-158
CC81STC	映射传送控制	$02B4_H$	U, PV	U, PV	页 20-159
CC81ECRD0	扩展回读 0	$02B8_H$	U, PV	BE	页 20-160
CC81ECRD1	扩展回读 1	$02BC_H$	U, PV	BE	页 20-162
CC82 寄存器					
CC82INS	输入选择器单元配置	0300_H	U, PV	U, PV	页 20-121
CC82CMC	连接矩阵配置	0304_H	U, PV	U, PV	页 20-124

表 20-13 CCU8 寄存器一览表 (续表)

简称	描述	偏移地址[1]	访问方式		描述见
			读	写	
CC82TCST	定时器运行状态	0308_H	U, PV	BE	页 20-126
CC82TCSET	定时器运行置位	$030C_H$	U, PV	U,PV	页 20-127
CC82TCCLR	定时器运行清除	0310_H	U, PV	U, PV	页 20-128
CC82TC	通用定时器配置	0314_H	U, PV	U, PV	页 20-129
CC82PSL	输出被动电平配置	0318_H	U, PV	U, PV	页 20-134
CC82DIT	抖动配置	$031C_H$	U, PV	BE	页 20-135
CC82DITS	抖动映射寄存器	0320_H	U, PV	U, PV	页 20-135
CC82PSC	预分频器配置	0324_H	U, PV	U, PV	页 20-136
CC82FPC	预分频器比较值	0328_H	U, PV	U, PV	页 20-137
CC82FPCS	预分频器映射比较值	$032C_H$	U, PV	U, PV	页 20-138
CC82PR	定时器周期值	0330_H	U, PV	BE	页 20-138
CC82PRS	定时器周期映射值	0334_H	U, PV	U, PV	页 20-139
CC82CR1	通道 1 的定时器比较值	0338_H	U, PV	BE	页 20-140
CC82CR1S	通道 1 的定时器比较映射值	$033C_H$	U, PV	U, PV	页 20-140
CC82CR2	通道 2 的定时器比较值	0340_H	U, PV	BE	页 20-141
CC82CR2S	通道 2 的定时器比较映射值	0344_H	U, PV	U, PV	页 20-142
CC82CHC	通道控制	0348_H	U, PV	U, PV	页 20-143
CC82DTC	死区时间控制	$034C_H$	U, PV	U, PV	页 20-144
CC82DC1R	通道 1 死区时间计数器值	0350_H	U, PV	U, PV	页 20-145
CC82DC2R	通道 2 死区时间计数器值	0354_H	U, PV	U, PV	页 20-146
CC82TIMER	定时器当前值	0370_H	U, PV	U, PV	页 20-147
CC82C0V	捕获寄存器 0 值	0374_H	U, PV	BE	页 20-147
CC82C1V	捕获寄存器 1 值	0378_H	U, PV	BE	页 20-148
CC82C2V	捕获寄存器 2 值	$037C_H$	U, PV	BE	页 20-149
CC82C3V	捕获寄存器 3 值	0380_H	U, PV	BE	页 20-150
CC82INTS	中断状态	$03A0_H$	U, PV	BE	页 20-151
CC82INTE	中断使能	$03A4_H$	U, PV	U, PV	页 20-153
CC82SRS	中断配置	$03A8_H$	U, PV	U, PV	页 20-155
CC82SWS	中断状态置位	$03AC_H$	U, PV	U, PV	页 20-156

表 20-13 CCU8 寄存器一览表 (续表)

简称	描述	偏移地址[1]	访问方式		描述见
			读	写	
CC82SWR	中断状态清除	$03B0_H$	U, PV	U, PV	页 20-158
CC82STC	映射传送控制	$03B4_H$	U, PV	U, PV	页 20-159
CC82ECRD0	扩展回读 0	$03B8_H$	U, PV	BE	页 20-160
CC82ECRD1	扩展回读 1	$03BC_H$	U, PV	BE	页 20-162
CC83 寄存器					
CC83INS	输入选择器单元配置	0400_H	U, PV	U, PV	页 20-121
CC83CMC	连接矩阵配置	0404_H	U, PV	U, PV	页 20-124
CC83TCST	定时器运行状态	0408_H	U, PV	BE	页 20-126
CC83TCSET	定时器运行置位	$040C_H$	U, PV	U,PV	页 20-127
CC83TCCLR	定时器运行清除	0410_H	U, PV	U, PV	页 20-128
CC83TC	通用定时器配置	0414_H	U, PV	U, PV	页 20-129
CC83PSL	输出被动电平配置	0418_H	U, PV	U, PV	页 20-134
CC83DIT	抖动配置	$041C_H$	U, PV	BE	页 20-135
CC83DITS	抖动映射寄存器	0420_H	U, PV	U, PV	页 20-135
CC83PSC	预分频器配置	0424_H	U, PV	U, PV	页 20-136
CC83FPC	预分频器比较值	0428_H	U, PV	U, PV	页 20-137
CC83FPCS	预分频器映射比较值	$042C_H$	U, PV	U, PV	页 20-138
CC83PR	定时器周期值	0430_H	U, PV	BE	页 20-138
CC83PRS	定时器周期映射值	0434_H	U, PV	U, PV	页 20-139
CC83CR1	通道 1 的定时器比较值	0438_H	U, PV	BE	页 20-140
CC83CR1S	通道 1 的定时器比较映射值	$043C_H$	U, PV	U, PV	页 20-140
CC83CR2	通道 2 的定时器比较值	0440_H	U, PV	BE	页 20-141
CC83CR2S	通道 2 的定时器比较映射值	0444_H	U, PV	U, PV	页 20-142
CC83CHC	通道控制	0448_H	U, PV	U, PV	页 20-143
CC83DTC	死区时间控制	$044C_H$	U, PV	U, PV	页 20-144
CC83DC1R	通道 1 死区时间计数器值	0450_H	U, PV	U, PV	页 20-145
CC83DC2R	通道 2 死区时间计数器值	0454_H	U, PV	U, PV	页 20-146
CC83TIMER	定时器当前值	0470_H	U, PV	U, PV	页 20-147
CC83C0V	捕获寄存器 0 值	0474_H	U, PV	BE	页 20-147

表 20-13 CCU8 寄存器一览表 (续表)

简称	描述	偏移地址[1]	访问方式		描述见
			读	写	
CC83C1V	捕获寄存器 1 值	0478_H	U, PV	BE	页 20-148
CC83C2V	捕获寄存器 2 值	$047C_H$	U, PV	BE	页 20-149
CC83C3V	捕获寄存器 3 值	0480_H	U, PV	BE	页 20-150
CC83INTS	中断状态	$04A0_H$	U, PV	BE	页 20-151
CC83INTE	中断使能	$04A4_H$	U, PV	U, PV	页 20-153
CC83SRS	中断配置	$04A8_H$	U, PV	U, PV	页 20-155
CC83SWS	中断状态置位	$04AC_H$	U, PV	U, PV	页 20-156
CC83SWR	中断状态清除	$04B0_H$	U, PV	U, PV	页 20-158
CC83STC	映射传送控制	$04B4_H$	U, PV	U, PV	页 20-159
CC83ECRD0	扩展回读 0	$04B8_H$	U, PV	BE	页 20-160
CC83ECRD1	扩展回读 1	$04BC_H$	U, PV	BE	页 20-162

1) 绝对寄存器地址计算如下：
模块基地址 + 偏移地址 (如本列所示)

20.7.1 全局寄存器

GCTRL

该寄存器包含影响 CCU8 内所有定时器片的全局配置位域。

GCTRL
全局控制寄存器 (0000_H) 复位值 : 00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MSDE		MSE3	MSE2	MSE1	MSE0	SUSCFG		0		PCIS		0	PRBC		
rw		rw	rw	rw	rw	rw		r		rw		r	rw		

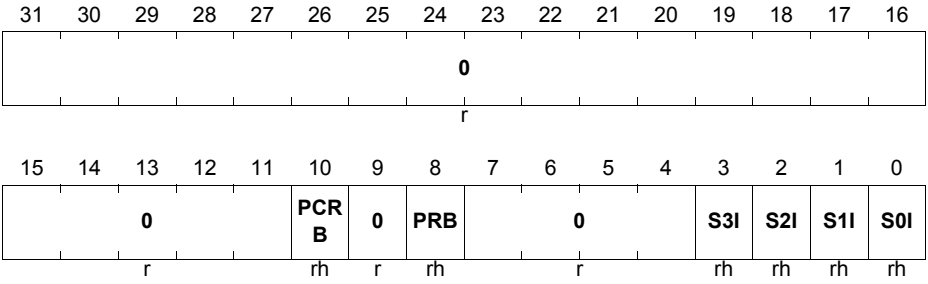
域	位	类型	描述
PRBC	[2:0]	rw	**预分频器清除配置** 该域控制如何清除预分频器的运行位和内部寄存器。 000_B 只能由软件清除。 001_B 在 CC80 的运行位被清除时， **GSTAT**.PRB 和预分频器寄存器被清除。 010_B 在 CC81 的运行位被清除时， **GSTAT**.PRB 和预分频器寄存器被清除。 011_B 在 CC82 的运行位被清除时， **GSTAT**.PRB 和预分频器寄存器被清除。 100_B 在 CC83 的运行位被清除时， **GSTAT**.PRB 和预分频器寄存器被清除。
PCIS	[5:4]	rw	**预分频器输入时钟选择** 00_B 模块时钟 01_B CCU8x.ECLKA 10_B CCU8x.ECLKB 11_B CCU8x.ECLKC
SUSCFG	[9:8]	rw	**挂起模式配置** 该域控制所有 CCU8 定时器片的挂起模式进入。 00_B 忽略挂起请求。模块不能进入挂起模式。 01_B 立即停止所有运行的定时器片。不采用安全停止。 10_B 立即停止模块，并将所有输出钳位到被动态。采用安全停止。 11_B 等待每个定时器片的翻转来停止并钳位定时器片的输出。采用安全停止。
MSE0	10	rw	**定时器片 0 多通道映射传送使能** 当该域被置位时，不仅可以通过软件还可以通过 CCU8x.MCSS 输入来请求定时器片 0 的映射传送。 0_B 只能通过软件来请求映射传送。 1_B 可以通过软件或 CCU8x.MCSS 输入来请求映射传送。

域	位	类型	描述
MSE1	11	rw	**定时器片 1 多通道映射传送使能** 当该域被置位时，不仅可以通过软件还可以通过 CCU8x.MCSS 输入来请求定时器片 1 的映射传送。 0_B 只能通过软件来请求映射传送。 1_B 可以通过软件或 CCU8x.MCSS 输入来请求映射传送。
MSE2	12	rw	**定时器片 2 多通道映射传送使能** 当该域被置位时，不仅可以通过软件还可以通过 CCU8x.MCSS 输入来请求定时器片 2 的映射传送。 0_B 只能通过软件来请求映射传送。 1_B 可以通过软件或 CCU8x.MCSS 输入来请求映射传送。
MSE3	13	rw	**定时器片 3 多通道映射传送使能** 当该域被置位时，不仅可以通过软件还可以通过 CCU8xx.MCSS 输入来请求定时器片 3 的映射传送。 0_B 只能通过软件来请求映射传送。 1_B 可以通过软件或 CCU8x.MCSS 输入来请求映射传送。
MSDE	[15:14]	rw	**多通道映射传送请求配置** 该域配置通过 CCU8x.MCSS 输入请求的映射传送类型。为使该配置有效，需要将域 **CC8yTC**.MSEy 置 1。 00_B 只请求周期值和比较值的映射传送 01_B 请求比较值、周期值和预分频器比较值的映射传送 10_B 保留 11_B 请求比较值、周期值、预分频器值和抖动比较值的映射传送
0	3, [7:6], [31:16]	r	**保留** 读访问总是返回 0。

GSTAT

该寄存器包含预分频器和每个定时器片的状态 (空闲模式或正在运行)。

GSTAT
全局状态寄存器 **(0004_H)** **复位值 : $0000000F_H$**



域	位	类型	描述
S0I	0	rh	**CC80 空闲状态** 该位指示定时器片 CC80 是否处于空闲模式。在空闲模式，定时器片 CC80 的时钟被停止。 0_B 正在运行 1_B 空闲
S1I	1	rh	**CC81 空闲状态** 该位指示定时器片 CC81 是否处于空闲模式。在空闲模式， CC81 的时钟被停止。 0_B 正在运行 1_B 空闲
S2I	2	rh	**CC82 空闲状态** 该位指示定时器片 CC82 是否处于空闲模式。在空闲模式， CC82 的时钟被停止。 0_B 正在运行 1_B 空闲
S3I	3	rh	**CC83 空闲状态** 该位指示定时器片 CC83 是否处于空闲模式。在空闲模式，CC83 的时钟被停止。 0_B 正在运行 1_B 空闲

域	位	类型	描述
PRB	8	rh	**预分频器运行位** 0_B 预分频器被停止 1_B 预分频器正在运行
PCRB	10	rh	**奇偶校验器运行位** 0_B 奇偶校验器被停止 1_B 奇偶校验器正在运行
0	[7:4], 9, [31:11]	r	**保留** 读访问总是返回 0。

GIDLS

通过该寄存器可以将预分频器和特定定时器片设置为空闲模式。

GIDLS

全局空闲置位 **(0008_H)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					**CPCH**	**PSIC**	**CPRB**	**0**				**SS3I**	**SS2I**	**SS1I**	**SS0I**
r					w	w	w	r				w	w	w	w

域	位	类型	描述
SS0I	0	w	**CC80 空闲模式置位** 向该位写入 1_B 会将 CC80 定时器片设置为空闲模式。当处于空闲模式时，该定时器片的时钟被停止。在进入空闲模式时，内部的定时器片寄存器不会被清除。 读访问总是返回 0。
SS1I	1	w	**CC81 空闲模式置位** 向该位写入 1_B 会将 CC81 定时器片设置为空闲模式。当处于空闲模式时，该定时器片的时钟被停止。在进入空闲模式时，内部的定时器片寄存器不会被清除。 读访问总是返回 0。

域	位	类型	描述
SS2I	2	w	**CC82 空闲模式置位** 向该位写入 1_B 会将 CC82 定时器片设置为空闲模式。当处于空闲模式时，该定时器片的时钟被停止。在进入空闲模式时，内部的定时器片寄存器不会被清除。 读访问总是返回 0。
SS3I	3	w	**CC83 空闲模式置位** 向该位写入 1_B 会将 CC83 定时器片设置为空闲模式。当处于空闲模式时，该定时器片的时钟被停止。在进入空闲模式时，内部的定时器片寄存器不会被清除。 读访问总是返回 0。
CPRB	8	w	**预分频器运行位清除** 向该位写入 1_B 会清除预分频器的运行位。预分频器内部的寄存器不会被清除。 读访问总是返回 0。
PSIC	9	w	**预分频器清除** 向该位写入 1_B 会清除预分频器的计数器。它也会将 PSIV 加载到所有定时器片的 PVAL 域。 这会将所有定时器片的定时器时钟进行重新调整。预分频器的运行位不会被清除。 读访问总是返回 0。
CPCH	10	w	**奇偶校验器运行位清除** 向该位写入 1_B 会清除奇偶校验器的运行位。所有内部寄存器被清除。状态位 **GPCHK**.PCST 的值保持不变。 读访问总是返回 0。
0	[7:4], [31:11]	r	**保留** 读访问总是返回 0。

GIDLC

通过该寄存器可以使预分频器和特定的定时器片退出空闲模式。

GIDLC

全局空闲模式清除 **($000C_H$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					SPCH	0	SPRB	0				CS3I	CS2I	CS1I	CS0I
r					w	r	w	r				w	w	w	w

域	位	类型	描述
CS0I	0	w	**CC80 空闲模式清除** 向该位写入 1_B 使 CC80 退出空闲模式。不清除内部定时器片寄存器。读访问总是返回 0。
CS1I	1	w	**CC81 空闲模式清除** 向该位写入 1_B 使 CC81 退出空闲模式。不清除内部定时器片寄存器。读访问总是返回 0。
CS2I	2	w	**CC82 空闲模式清除** 向该位写入 1_B 使 CC82 退出空闲模式。不清除内部定时器片寄存器。读访问总是返回 0。
CS3I	3	w	**CC83 空闲模式清除** 向该位写入 1_B 使 CC83 退出空闲模式。不清除内部定时器片寄存器。读访问总是返回 0。
SPRB	8	w	**预分频器运行位置位** 向该位写入 1_B 可以将预分频器的运行位置位。不清除预分频器内部寄存器。读访问总是返回 0。
SPCH	10	w	**奇偶校验器运行位置位** 向该位写入 1_B 可以将奇偶校验器的运行位置位。读访问总是返回 0。
0	[7:4], 9, [31:11]	r	**保留** 读访问总是返回 0。

GCSS

通过该寄存器可以为特定的定时器片请求一次映射传送，并将每个比较通道的状态位置 1。

GCSS

全局通道置位 (0010_H) 复位值：00000000_H

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								S3S T2S	S2S T2S	S1S T2S	S0S T2S	S3S T1S	S2S T1S	S1S T1S	S0S T1S
r								w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3P SE	S3D SE	S3S E	0	S2P SE	S2D SE	S2S E	0	S1P SE	S1D SE	S1S E	0	S0P SE	S0D SE	S0S E
r	w	w	w	r	w	w	w	r	w	w	w	r	w	w	w

域	位	类型	描述
S0SE	0	w	**定时器片 0 映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S0SS 域置 1，然后会启动对周期值、比较值和被动电平值的一次映射传送。 读访问总是返回 0。
S0DSE	1	w	**定时器片 0 抖动映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S0DSS 域置 1，然后会启动对抖动比较值的一次映射传送。 读访问总是返回 0。
S0PSE	2	w	**定时器片 0 预分频器映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S0PSS 域置 1，然后会启动预分频器比较值的一次映射传送。 读访问总是返回 0。
S1SE	4	w	**定时器片 1 映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S1SS 域置 1，然后会启动对周期值、比较值和被动电平值的一次映射传送。 读访问总是返回 0。
S1DSE	5	w	**定时器片 1 抖动映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S1DSS 域置 1，然后会启动对抖动比较值的一次映射传送。 读访问总是返回 0。

域	位	类型	描述
S1PSE	6	w	**定时器片 1 预分频器映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S1PSS 域置 1，然后会启动对预分频器比较值的一次映射传送。 读访问总是返回 0。
S2SE	8	w	**定时器片 2 映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S2SS 域置 1，然后会启动对周期值、比较值和被动电平值的一次映射传送。 读访问总是返回 0。
S2DSE	9	w	**定时器片 2 抖动映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S2DSS 域置 1，然后会启动对抖动比较值的一次映射传送。 读访问总是返回 0。
S2PSE	10	w	**定时器片 2 预分频器映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S2PSS 域置 1，然后会启动对预分频器比较值的一次映射传送。 读出值总是为 0。
S3SE	12	w	**定时器片 3 映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S3SS 域置 1，然后会启动对周期值、比较值和被动电平值的一次映射传送。 读访问总是返回 0。
S3DSE	13	w	**定时器片 3 抖动映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S3DSS 域置 1，然后会启动对抖动比较值的一次映射传送。 读访问总是返回 0。
S3PSE	14	w	**定时器片 3 预分频器映射传送置位使能** 向该位写入 1_B 会将 **GCST**.S3PSS 位域置 1，然后会启动对预分频器比较值的一次映射传送。 读访问总是返回 0。
S0ST1S	16	w	**定时器片 0 状态位 1 置位** 向该位写入 1_B 会将定时器片 0 的比较通道 1 状态位 (**GCST**.CC80ST1) 置 1。 读访问总是返回 0。
S1ST1S	17	w	**定时器片 1 状态位 1 置位** 向该位写入 1_B 会将定时器片 1 的比较通道 1 状态位 (**GCST**.CC81ST1) 置 1。 读访问总是返回 0。

域	位	类型	描述
S2ST1S	18	w	**定时器片 2 状态位 1 置位** 向该位写入 1_B 会将定时器片 2 的比较通道 1 状态位 (**GCST**.CC82ST1) 置 1。 读访问总是返回 0。
S3ST1S	19	w	**定时器片 3 状态位 1 置位** 向该位写入 1_B 会将定时器片 3 的比较通道 1 状态位 (**GCST**.CC83ST1) 置 1。 读访问总是返回 0。
S0ST2S	20	w	**定时器片 0 状态位 2 置位** 向该位写入 1_B 会将定时器片 0 的比较通道 2 状态位 (**GCST**.CC80ST2) 置 1。 读访问总是返回 0。
S1ST2S	21	w	**定时器片 1 状态位 2 置位** 向该位写入 1_B 会将定时器片 1 的比较通道 2 状态位 (**GCST**.CC81ST2) 置 1。 读访问总是返回 0。
S2ST2S	22	w	**定时器片 2 状态位 2 置位** 向该位写入 1_B 会将定时器片 2 的比较通道 2 状态位 (**GCST**.CC82ST2) 置 1。 读访问总是返回 0。
S3ST2S	23	w	**定时器片 3 状态位 2 置位** 向该位写入 1_B 会将定时器片 3 的比较通道 2 状态位 (**GCST**.CC83ST2) 置 1。 读访问总是返回 0。
0	3, 7, 11, 15, [31:24]	r	**保留** 读访问总是返回 0。

GCSC

通过该寄存器，可以将特定定时器片的一次映射传送请求复位，并清除每个比较通道的状态位。

GCSC

全局通道清除 **(0014$_H$)** **复位值 : 00000000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								S3S T2C	S2S T2C	S1S T2C	S0S T2C	S3S T1C	S2S T1C	S1S T1C	S0S T1C
r								w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3P SC	S3D SC	S3S C	0	S2P SC	S2D SC	S2S C	0	S1P SC	S1D SC	S1S C	0	S0P SC	S0D SC	S0S C
r	w	w	w	r	w	w	w	r	w	w	w	r	w	w	w

域	位	类型	描述
S0SC	0	w	**定时器片 0 映射传送清除** 向该位写入 1_B 会清除 **GCST**.S0SS 域，从而取消对周期值、比较值和被动电平值的任何即将执行的映射传送。 读访问总是返回 0。
S0DSC	1	w	**定时器片 0 抖动映射传送清除** 向该位写入 1_B 会清除 **GCST**.S0DSS 域，从而取消对抖动比较值的任何即将执行的映射传送。 读访问总是返回 0。
S0PSC	2	w	**定时器片 0 预分频器映射传送清除** 向该位写入 1_B 会清除 **GCST**.S0DSS 域，从而取消对抖动比较值的任何即将执行的映射传送。 读访问总是返回 0。
S1SC	4	w	**定时器片 1 映射传送清除** 向该位写入 1_B 会清除 **GCST**.S1SS 域，从而取消对周期值、比较值和被动电平值的任何即将执行的映射传送。 读访问总是返回 0。

域	位	类型	描述
S1DSC	5	w	**定时器片 1 抖动映射传送清除** 向该位写入 1_B 会清除 **GCST**.S1DSS 域，从而取消对抖动比较值的任何即将执行的映射传送。 读访问总是返回 0。
S1PSC	6	w	**定时器片 1 预分频器映射传送清除** 向该位写入 1_B 会清除 **GCST**.S1PSS 域，从而取消对预分频器比较值的任何即将执行的映射传送。 读访问总是返回 0。
S2SC	8	w	**定时器片 2 映射传送清除** 向该位写入 1_B 会清除 **GCST**.S2SS 域，从而取消对周期值、比较值和被动电平值的任何即将执行的映射传送。 读访问总是返回 0。
S2DSC	9	w	**定时器片 2 抖动映射传送清除** 向该位写入 1_B 会清除 **GCST**.S2DSS 域，从而取消对抖动比较值的任何即将执行的映射传送。 读访问总是返回 0。
S2PSC	10	w	**定时器片 2 预分频器映射传送清除** 向该位写入 1_B 会清除 **GCST**.S2PSS 域，从而取消对预分频器比较值的任何即将执行的映射传送。 读访问总是返回 0。
S3SC	12	w	**定时器片 3 映射传送清除** 向该位写入 1_B 会清除 **GCST**.S3SS 域，从而取消对周期值、比较值和被动电平值的任何即将执行的映射传送。 读访问总是返回 0。
S3DSC	13	w	**定时器片 3 抖动映射传送清除** 向该位写入 1_B 会清除 **GCST**.S3DSS 域，从而取消对抖动比较值的任何即将执行的映射传送。 读访问总是返回 0。
S3PSC	14	w	**定时器片 3 预分频器映射传送清除** 向该位写入 1_B 会清除 **GCST**.S3PSS 域，从而取消对预分频器比较值的任何即将执行的映射传送。 读访问总是返回 0。
S0ST1C	16	w	**定时器片 0 状态位 1 清除** 向该位写入 1_B 会清除定时器片 0 的比较通道 1 状态位 (**GCST**.CC80ST1)。 读访问总是返回 0。

域	位	类型	描述
S1ST1C	17	w	**定时器片 1 状态位 1 清除** 向该位写入 1_B 会清除定时器片 1 的比较通道 1 状态位 (**GCST**.CC81ST1)。 读访问总是返回 0。
S2ST1C	18	w	**定时器片 2 状态位 1 清除** 向该位写入 1_B 会清除定时器片 2 的比较通道 1 状态位 (**GCST**.CC82ST1)。 读访问总是返回 0。
S3ST1C	19	w	**定时器片 3 状态位 1 清除** 向该位写入 1_B 会清除定时器片 3 的比较通道 1 状态位 (**GCST**.CC83ST1)。 读访问总是返回 0。
S0ST2C	20	w	**定时器片 0 状态位 2 清除** 向该位写入 1_B 会清除定时器片 0 的比较通道 2 状态位 (**GCST**.CC80ST2)。 读访问总是返回 0。
S1ST2C	21	w	**定时器片 1 状态位 2 清除** 向该位写入 1_B 会清除定时器片 1 的比较通道 2 状态位 (**GCST**.CC81ST2)。 读访问总是返回 0。
S2ST2C	22	w	**定时器片 2 状态位 2 清除** 向该位写入 1_B 会清除定时器片 2 的比较通道 2 状态位 (**GCST**.CC82ST2)。 读访问总是返回 0。
S3ST2C	23	w	**定时器片 3 状态位 2 清除** 向该位写入 1_B 会清除定时器片 3 的比较通道 2 状态位 (**GCST**.CC83ST2)。 读访问总是返回 0。
0	3, 7, 11, 15, [31:24]	r	**保留** 读访问总是返回 0。

GCST

该寄存器保持映射传送请求信息和每个定时器片的状态位信息。

GCST

全局通道状态 (0018$_H$) 复位值 : 00000000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								CC8 3ST2	CC8 2ST2	CC8 1ST2	CC8 0ST2	CC8 3ST1	CC8 2ST1	CC8 1ST1	CC8 0ST1
r								rh	rh	rh	rh	rh	rh	rh	rh

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	S3PSS	S3DSS	S3SS	0	S2PSS	S2DSS	S2SS	0	S1PSS	S1DSS	S1SS	0	S0PSS	S0DSS	S0SS
r	rh	rh	rh	r	rh	rh	rh	r	rh	rh	rh	r	rh	rh	rh

域	位	类型	描述
S0SS	0	rh	**定时器片 0 映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S0DSS	1	rh	**定时器片 0 抖动映射传送状态** 0_B 尚未请求抖动映射传送 1_B 已经请求抖动映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S0PSS	2	rh	**定时器片 0 预分频器映射传送状态** 0_B 尚未请求预分频器映射传送 1_B 已经请求预分频器映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S1SS	4	rh	**定时器片 1 映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S1DSS	5	rh	**定时器片 1 抖动映射传送状态** 0_B 尚未请求抖动映射传送 1_B 已经请求抖动映射传送 在所请求的映射传送被执行后，该域由硬件清除。

域	位	类型	描述
S1PSS	6	rh	**定时器片 1 预分频器映射传送状态** 0_B 尚未请求预分频器映射传送 1_B 已经请求预分频器映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S2SS	8	rh	**定时器片 2 映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S2DSS	9	rh	**定时器片 2 抖动映射传送状态** 0_B 尚未请求抖动映射传送 1_B 已经请求抖动映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S2PSS	10	rh	**定时器片 2 预分频器映射传送状态** 0_B 尚未请求预分频器映射传送 1_B 已经请求预分频器映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S3SS	12	rh	**定时器片 3 映射传送状态** 0_B 尚未请求映射传送 1_B 已经请求映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S3DSS	13	rh	**定时器片 3 抖动映射传送状态** 0_B 尚未请求抖动映射传送 1_B 已经请求抖动映射传送 在所请求的映射传送被执行后，该域由硬件清除。
S3PSS	14	rh	**定时器片 3 预分频器映射传送状态** 0_B 尚未请求预分频器映射传送 1_B 已经请求预分频器映射传送 在所请求的映射传送被执行后，该域由硬件清除。
CC80ST1	16	rh	**定时器片 0 比较通道 1 状态位**
CC81ST1	17	rh	**定时器片 1 比较通道 1 状态位**
CC82ST1	18	rh	**定时器片 2 比较通道 1 状态位**
CC83ST1	19	rh	**定时器片 3 比较通道 1 状态位**
CC80ST2	20	rh	**定时器片 0 比较通道 2 状态位**
CC81ST2	21	rh	**定时器片 1 比较通道 2 状态位**
CC82ST2	22	rh	**定时器片 2 比较通道 2 状态位**
CC83ST2	23	rh	**定时器片 3 比较通道 2 状态位**

域	位	类型	描述
0	3, 7, 11, 15, [31:24]	r	**保留** 读访问总是返回 0。

GPCHK

该寄存器包含 CCU8 奇偶校验功能的配置。

GPCHK
奇偶校验器配置 **($001C_H$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PCSEL3				PCSEL2				PCSEL1				PCSEL0			
rw				rw				rw				rw			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PCST	0							PCTS	PCDS		PISEL		PACS		PASE
rh	r							rw	rw		rw		rw		rw

域	位	类型	描述
PASE	0	rw	**奇偶校验器自动启动 / 停止** 如果该域置 1，当所选定时器片的运行位置 1 时奇偶校验器的运行位被自动置 1；当所选定时器片的运行位清 0 时奇偶校验器的运行位被自动清 0。需要根据所选定时器片对域 PACS 编程。
PACS	[2:1]	rw	**奇偶校验器自动启动 / 停止选择器** 该域选择与奇偶校验器自动启动 / 停止相关联的定时器片： 00_B CC80 01_B CC81 10_B CC82 11_B CC83

域	位	类型	描述
PISEL	[4:3]	rw	**驱动器输入信号选择器** 该域选择包含驱动器奇偶信息的信号： 00_B CC8x.GP01 - 驱动器输出连接到定时器片 0 的事件 1 01_B CC8x.GP11 - 驱动器输出连接到定时器片 1 的事件 1 10_B CC8x.GP21 - 驱动器输出连接到定时器片 2 的事件 1 11_B CC8x.GP31 - 驱动器输出连接到定时器片 3 的事件 1
PCDS	[6:5]	rw	**奇偶校验器延迟输入选择器** 该域选择控制 CCU8 输出改变与驱动器奇偶输出实际改变之间的延迟的信号： 00_B CCU8x.IGBTA 01_B CCU8x.IGBTB 10_B CCU8x.IGBTC 11_B CCU8x.IGBTD
PCTS	7	rw	**奇偶校验器类型选择器** 该域选择使用奇校验还是偶校验： 0_B 使能偶校验 1_B 使能奇校验
PCST	15	rh	**奇偶校验器 XOR 状态** 该域包含 XOR 链的当前值。
PCSEL0	[19:16]	rw	**奇偶校验器定时器片 0 输出选择** 该域选择要用于执行奇偶校验的定时器片 0 输出。相应的位域需要被置为 1_B，以使能奇偶校验功能的输出。 PCSEL0[0] - CCU8x.OUT00 PCSEL0[1] - CCU8x.OUT01 PCSEL0[2] - CCU8x.OUT02 PCSEL0[3] - CCU8x.OUT03
PCSEL1	[23:20]	rw	**奇偶校验器定时器片 1 输出选择** 描述同 PCSEL0。
PCSEL2	[27:24]	rw	**奇偶校验器定时器片 2 输出选择** 描述同 PCSEL0。
PCSEL3	[31:28]	rw	**奇偶校验器定时器片 3 输出选择** 描述同 PCSEL0。

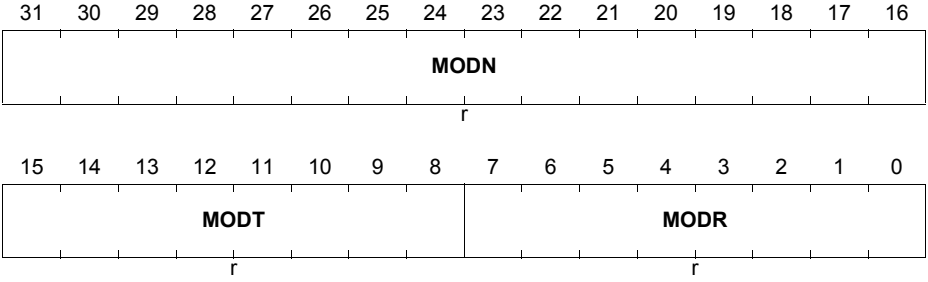
域	位	类型	描述
0	[14:8]	r	**保留** 读访问总是返回 0。

MIDR

该寄存器包含模块标识号。

MIDR

模块标识号 **(0080$_H$)** **复位值 : 00A7C0XX$_H$**



域	位	类型	描述
MODR	[7:0]	r	**模块版本** 该位域指示模块实现 (取决于设计阶段) 的版本号。 给定值 00_H 是实际版本号的占位符。
MODT	[15:8]	r	**模块类型**
MODN	[31:16]	r	**模块号**

20.7.2 定时器片 (CC8y) 寄存器

CC8yINS

该寄存器包含输入选择器的配置信息。

CC8yINS (y = 0 - 3)

输入选择器配置 **(0100_H + 0100_H * y)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	LPF2M		LPF1M		LPF0M		EV2LM	EV1LM	EV0LM	EV2EM		EV1EM		EV0EM	
r	rw		rw		rw		rw	rw	rw	rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				EV2IS				EV1IS				EV0IS			
r				rw				rw				rw			

域	位	类型	描述
EV0IS	[3:0]	rw	**事件 0 信号选择** 该位域选择哪个引脚用于事件 0。 0000_B CCU8x.INyA 0001_B CCU8x.INyB 0010_B CCU8x.INyC 0011_B CCU8x.INyD 0100_B CCU8x.INyE 0101_B CCU8x.INyF 0110_B CCU8x.INyG 0111_B CCU8x.INyH 1000_B CCU8x.INyI 1001_B CCU8x.INyJ 1010_B CCU8x.INyK 1011_B CCU8x.INyL 1100_B CCU8x.INyM 1101_B CCU8x.INyN 1110_B CCU8x.INyO 1111_B CCU8x.INyP
EV1IS	[7:4]	rw	**事件 1 信号选择** 描述同 EV0IS
EV2IS	[11:8]	rw	**事件 2 信号选择** 描述同 EV0IS

域	位	类型	描述
EV0EM	[17:16]	rw	**事件 0 边沿选择** 00_B 无作用 01_B 信号在上升沿有效 10_B 信号在下降沿有效 11_B 信号在两个边沿都有效
EV1EM	[19:18]	rw	**事件 1 边沿选择** 描述同 EV0EM
EV2EM	[21:20]	rw	**事件 2 边沿选择** 描述同 EV0EM
EV0LM	22	rw	**事件 0 电平选择** 0_B 高电平有效 1_B 低电平有效
EV1LM	23	rw	**事件 1 电平选择** 描述同 EV0LM
EV2LM	24	rw	**事件 2 电平选择** 描述同 EV0LM
LPF0M	[26:25]	rw	**事件 0 低通滤波器配置** 该域设置事件 0 低通滤波器连续计数的数目。输入信号值需要在这些计数 (fCCU8) 周期内保持稳定，以便一个电平 / 跳变能被接受。 00_B 低通滤波器被禁止 01_B 3 个 fCCU8 时钟周期 10_B 5 个 fCCU8 时钟周期 11_B 7 个 f_{CCU8} 时钟周期
LPF1M	[28:27]	rw	**事件 1 低通滤波器配置** 描述同 LPF0M
LPF2M	[30:29]	rw	**事件 2 低通滤波器配置** 描述同 LPF0M
0	[15:12], 31	r	**保留** 读访问总是返回 0。

CC8yCMC

该寄存器包含连接矩阵的配置。

CC8yCMC (y = 0 - 3)

连接矩阵控制 **($0104_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											TCE	MOS		TS	OFS
r											rw	rw		rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CNTS		LDS		UDS		GATES		CAP1S		CAP0S		ENDS		STRTS	
rw		rw		rw		rw		rw		rw		rw		rw	

域	位	类型	描述
STRTS	[1:0]	rw	**外部启动功能选择器** 选择要与外部启动功能连接的事件。 00_B 禁用外部启动功能 01_B 外部启动功能由事件 0 触发 10_B 外部启动功能由事件 1 触发 11_B 外部启动功能由事件 2 触发
ENDS	[3:2]	rw	**外部停止功能选择器** 选择要与外部停止功能连接的事件。 00_B 禁用外部停止功能 01_B 外部停止功能由事件 0 触发 10_B 外部停止功能由事件 1 触发 11_B 外部停止功能由事件 2 触发
CAP0S	[5:4]	rw	**外部捕获 0 功能选择器** 选择要与捕获寄存器 1 和捕获寄存器 0 的外部捕获功能连接的事件。该功能用于将定时器的值捕获到捕获寄存器 1 和 0。 00_B 禁用外部捕获 0 功能 01_B 外部捕获 0 功能由事件 0 触发 10_B 外部捕获 0 功能由事件 1 触发 11_B 外部捕获 0 功能由事件 2 触发 *注： 如果域 SCE 被置 1，该功能被禁用。*

域	位	类型	描述
CAP1S	[7:6]	rw	**外部捕获 1 功能选择器** 选择要与捕获寄存器 3 和捕获寄存器 2 的外部捕获功能连接的事件。该功能用于将定时器的值捕获到捕获寄存器 3 和 2。 00_B 禁用外部捕获 1 功能 01_B 外部捕获 1 功能由事件 0 触发 10_B 外部捕获 1 功能由事件 1 触发 11_B 外部捕获 1 功能由事件 2 触发
GATES	[9:8]	rw	**外部门控功能选择器** 选择要与计数器门控功能连接的事件。该功能用于控制定时器的增 1/ 减 1 计数过程。 00_B 禁用外部门控功能 01_B 外部门控功能由事件 0 触发 10_B 外部门控功能由事件 1 触发 11_B 外部门控功能由事件 2 触发
UDS	[11:10]	rw	**外部向上 / 向下计数功能选择器** 选择要与向上 / 向下计数方向控制连接的事件。该功能用于从外部控制定时器的增 1/ 减 1 操作。 00_B 禁用外部向上 / 向下计数功能 01_B 外部向上 / 向下计数功能由事件 0 触发 10_B 外部向上 / 向下计数功能由事件 1 触发 11_B 外部向上 / 向下计数功能由事件 2 触发
LDS	[13:12]	rw	**外部定时器加载功能选择器** 选择要与定时器加载功能连接的事件。 **CC8yCR1/CC8yCR2** (取决于 **CC8yTC**.TLS 的值) 中的值被加载到特定的定时器片。 00_B - 禁用外部加载功能 01_B - 外部加载功能由事件 0 触发 10_B - 外部加载功能由事件 1 触发 11_B - 外部加载功能由事件 2 触发
CNTS	[15:14]	rw	**外部计数选择器** 选择将要与计数功能相连接的事件。每当检测到事件的一个特定跳变时，计数器将增 1/ 减 1。 00_B 禁用外部计数功能 01_B 外部计数功能由事件 0 触发 10_B 外部计数功能由事件 1 触发 11_B 外部计数功能由事件 2 触发 *注： 在 CC80 中，该域不存在。这是一个只读的保留域。读访问总是返会 0。*

域	位	类型	描述
OFS	16	rw	**覆盖功能选择器** 该位使能状态位覆盖功能。 0_B 禁用覆盖功能 1_B 状态位触发信号覆盖与事件 1 相连，状态位值覆盖与事件 2 相连。
TS	17	rw	**陷阱功能选择器** 该位域使能陷阱功能。 0_B 禁止陷阱功能 1_B 陷阱功能与事件 2 相连
MOS	[19:18]	rw	**外部调制功能选择器** 选择要与外部调制功能连接的事件。 00_B - 禁用调制功能 01_B - 调制功能由事件 0 触发 10_B - 调制功能由事件 1 触发 11_B - 调制功能由事件 2 触发
TCE	20	rw	**定时器级联使能** 该位使能与前一个定时器片的级联。 0_B 禁止定时器级联 1_B 使能定时器级联 *注： 在 CC80 中，该域不存在。这是一个只读的保留位。读访问总是返回 0。*
0	[31:21]	r	**保留** 读访问总是返回 0。

CC8yTCST

该寄存器保持定时器的状态 (运行 / 停止) 和计数方向 (向上 / 向下) 信息。

CC8yTCST (y = 0 - 3)

片定时器的状态 **($0108_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											DTR2	DTR1	0	CDIR	TRB
r											rh	rh	r	rh	rh

域	位	类型	描述
TRB	0	rh	**定时器运行位** 该位指示定时器是否正在运行。 0_B 定时器已停止 1_B 定时器正在运行
CDIR	1	rh	**定时器计数方向** 该位指示定时器是向上计数还是向下计数 0_B 定时器向上计数 1_B 定时器向下计数
DTR1	3	rh	**死区时间计数器 1 运行位** 该域指示与通道 1 连接的死区时间计数器是否运行。 0_B 死区时间计数器空闲 1_B 死区时间计数器正在运行
DTR2	4	rh	**死区时间计数器 2 运行位** 该域指示与通道 2 连接的死区时间计数器是否运行。 0_B 死区时间计数器空闲 1_B 死区时间计数器正在运行
0	2, [31:5]	r	**保留** 读访问总是返回 0

CC8yTCSET

可以通过该寄存器启动定时器。

CC8yTCSET (y = 0 - 3)

片定时器运行置位 **($010C_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															TRBS
r															w

域	位	类型	描述
TRBS	0	w	**定时器运行位置位** 向该域写入 1_B 将定时器的运行位置 1。 读访问总是返回 0
0	[31:1]	r	**保留** 读访问总是返回 0

CC8yTCCLR

通过该寄存器可以停止并清除定时器，并清除抖动计数器。

CC8yTCCLR (y = 0 - 3)

片定时器清除 **($0110_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											DTC 2C	DTC 1C	DITC	TCC	TRB C
r											w	w	w	w	w

域	位	类型	描述
TRBC	0	w	**定时器运行位清除** 向该域写入 1_B 将清除定时器的运行位。定时器不被清零。 读访问总是返回 0。
TCC	1	w	**定时器清零** 向该域写入 1_B 将定时器清零。 读访问总是返回 0。
DITC	2	w	**抖动计数器清零** 向该域写入 1_B 将抖动计数器清零。 读访问总是返回 0。

域	位	类型	描述
DTC1C	3	w	**死区计数器 1 清零** 向该域写入 1_B 将通道 1 的死区计数器清零。该计数器被停止，直到检测到一次新的启动触发。 读访问总是返回 0。
DTC2C	4	w	**死区计数器 2 清零** 向该域写入 1_B 将通道 2 的死区计数器清零。该计数器被停止，直到检测到一次新的启动触发。 读访问总是返回 0。
0	[31:5]	r	**保留** 读访问总是返回 0。

CC8yTC

该寄存器保持定时器操作的几种可能配置。

CC8yTC (y = 0 - 3)

片定时器控制 **($0114_H + 0100_H * y$)** **复位值 : 18000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**STOS**		**EME**		**MCME2**	**MCME1**	**EMT**	**EMS**	**TRPSW**	**TRPSE**	**TRAPE3**	**TRAPE2**	**TRAPE1**	**TRAPE0**	**FPE**
r	rw		rw		rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIM	**DITHE**		**CCS**	**SCE**	**STRM**	**ENDM**		**TLS**	**CAPC**		**ECM**	**CMOD**	**CLST**	**TSSM**	**TCM**
rw	rw		rw	rw	rw	rw		rw	rw		rw	rh	rw	rw	rw

域	位	类型	描述
TCM	0	rw	**定时器计数模式** 该域控制定时器的实际计数方式。 0_B 边沿对齐模式 1_B 中心对齐模式 *注：当使用一个外部信号来控制计数方向时，计数方式总是为边沿对齐。*

域	位	类型	描述
TSSM	1	rw	**定时器单次模式** 该域控制单次模式。该模式可用于边沿对齐模式和中心对齐模式。 0_B 禁止单次模式 1_B 使能单次模式
CLST	2	rw	**清除时执行映射传送** 将该位设置为 1_B 使得在通过软件或外部事件完成定时器清除操作时进行一次映射传送。注意，仍需通过软件将 **GCST** 寄存器中的映射传送使能位设置为 1_B。
CMOD	3	rh	**捕获比较模式** 该域指示定时器片工作在哪个模式。默认值是比较模式。当一个外部信号被映射为捕获触发信号时，硬件自动将定时器片设置为捕获模式。 0_B 比较模式 1_B 捕获模式
ECM	4	rw	**扩展捕获模式** 该域控制特定定时器片的捕获模式。该域仅在 CMOD 位为 1_B 时有效。 0_B 标准捕获模式。只能通过单独访问寄存器来清除每个捕获寄存器的满标志 1_B 扩展捕获模式。不仅可以通过单独访问寄存器来清除每个寄存器的满标志，还可以通过访问 ECRD 寄存器来实现。当读取 ECRD 寄存器时，仅清除 ECRD.VPTR 指向的捕获寄存器满标志。
CAPC	[6:5]	rw	**捕获时的清除控制** 00_B 定时器不会在发生捕获事件时被清除 01_B 定时器由一个捕获到捕获寄存器 2 和 3 的捕获事件来清除 (当 SCE = 1_B 时，定时器总是在一次捕获事件中被清除) 10_B 定时器由一个捕获到捕获寄存器 0 和 1 的捕获事件来清除 (当 SCE = 1_B 时，定时器总是在一次捕获事件中被清除) 11_B 定时器总是在一次捕获事件中被清除。
TLS	7	rw	**定时器加载选择器** 0_B 用 CR1 的值加载定时器 1_B 用 CR2 的值加载定时器

域	位	类型	描述
ENDM	[9:8]	rw	**扩展停止功能控制** 该域控制外部停止信号的扩展功能。 00_B 仅清除定时器运行位 (默认为停止) 01_B 仅清除定时器 (清空) 10_B 清除定时器及运行位 (清空 / 停止) 11_B 保留 *注：当使用一个外部向上/向下计数信号时，若计数器向上计数，则清空操作将定时器值设置为 0，若计数器向下计数，则清空操作将定时器值设置为周期值。*
STRM	10	rw	**扩展启动功能控制** 该域控制外部启动信号的扩展功能。 0_B 仅将运行位置 1 (默认为启动) 1_B 清除定时器并将运行位置 1(清空 / 启动) *注：当使用一个外部向上/向下计数信号时，若计数器向上计数，则清空操作将定时器值设置为 0，若计数器向下计数，则清空操作将定时器值设置为周期值。*
SCE	11	rw	**相同捕获事件使能** 0_B CCycapt0 控制捕获到 **CC8yC0V**/**CC8yC1V** 寄存器， CCycapt1 控制捕获到 **CC8yC3V**/**CC8yC2V** 寄存器 1_B 捕获到 **CC8yC0V**/**CC8yC1V** 寄存器和 **CC8yC3V**/**CC8yC2V** 寄存器都由 CCycapt1 控制
CCS	12	rw	**连续捕获使能** 0_B 使用如 **20.2.8.6 节**所描述的与满标志相关连的规则来捕获到特定的捕获寄存器。 1_B 无论满标志的状态如何，总是执行到捕获寄存器的捕获 (即使寄存器还没有被回读)。
DITHE	[14:13]	rw	**抖动使能** 该域控制定时器片的抖动模式。详见 **20.2.12 节**。 00_B 禁止抖动 01_B 将抖动应用到周期值 10_B 将抖动应用到比较值 11_B 将抖动应用到周期值和比较值

域	位	类型	描述
DIM	15	rw	**抖动输入选择器** 该域选择将抖动控制信号连接到特定定时器片的抖动逻辑还是连接到定时器片 0 的抖动逻辑。注意，即使该域被设置为 1_B，仍需对 DITHE 域编程。 0_B 定时器片使用自身的抖动逻辑单元 1_B 定时器片被连接到定时器片 0 的抖动逻辑单元。
FPE	16	rw	**浮动预分频器使能** 将该位设置为 1_B 使能浮动预分频器模式。 0_B 禁止浮动预分频器模式 1_B 使能浮动预分频器模式
TRAPE0	17	rw	**CCU8x.OUTy0 的陷阱使能** 将该位置 1 使能 CCU8x.OUTy0 输出引脚的陷阱功能。在将一个外部信号映射到陷阱功能后，用户必须将该位域置 1，以激活特定输出引脚的陷阱功能。向该域写 0 将禁止陷阱功能，与输入信号的状态无关。 0_B CCU8x.OUTy0 输出的陷阱功能无效 1_B 陷阱功能影响 CCU8x.OUTy0 的输出
TRAPE1	18	rw	**CCU8x.OUTy1 的陷阱使能** CCU8x.OUTy1 的陷阱使能。描述同 TRAPE0 域。
TRAPE2	19	rw	**CCU8x.OUTy2 的陷阱使能** CCU8x.OUTy2 的陷阱使能。描述同 TRAPE0 域。
TRAPE3	20	rw	**CCU8x.OUTy3 的陷阱使能** CCU8x.OUTy3 的陷阱使能。描述同 TRAPE0 域。
TRPSE	21	rw	**陷阱同步使能** 向该位写入 1_B 会使陷阱状态退出与 PWM 信号同步。 0_B 退出陷阱状态不与 PWM 信号同步 1_B 退出陷阱状态与 PWM 信号同步
TRPSW	22	rw	**陷阱状态清除控制** 0_B 当陷阱条件不存在时，定时器片自动退出陷阱状态（陷阱通过硬件和软件清零） 1_B 退出陷阱状态只能通过软件请求完成。

域	位	类型	描述
EMS	23	rw	**外部调制同步** 将该位设置为 1_B 使能外部调制功能与 PWM 信号的同步。 0_B 外部调制功能不与 PWM 信号同步 1_B 外部调制功能与 PWM 信号同步
EMT	24	rw	**外部调制类型** 该域选择外部调制事件是用于清除 CC8ySTx 位还是用于门控输出。 0_B 外部调制事件用于清除 CC8ySTx 位。 1_B 外部调制事件用于门控输出。
MCME1	25	rw	**通道 1 的多通道模式使能** 0_B 禁止通道 1 的多通道模式 1_B 使能通道 1 的多通道模式
MCME2	26	rw	**通道 2 的多通道模式使能** 0_B 禁止通道 2 的多通道模式 1_B 使能通道 2 的多通道模式
EME	[28:27]	rw	**外部调制通道使能** 该域控制调制功能在哪一个通道有效。需要在此之前通过设置 **CC8yCMC**.OFS = 11_B 来使能调制功能。 00_B 外部调制功能不影响任何通道 01_B 外部调制功能只应用于通道 1 10_B 外部调制功能只应用于通道 2 11_B 外部调制功能应用到全部两个通道
STOS	[30:29]	rw	**状态位输出选择器** 该域选择将输出 CC8ySTy 映射到哪个通道。 00_B CC8ySTI 被转发到 CCU8x.STy 01_B CC8ySTI2 被转发到 CCU8x.STy 10_B CC8ySTI AND CC8ySTI2 被转发到 CCU8x.STy 11_B CC8ySTI OR CC8ySTI2 被转发到 CCU8x.STy
0	31	r	**保留** 读访问总是返回 0。

CC8yPSL

该寄存器保持对输出被动电平控制的配置。

CC8yPSL (y = 0 - 3)
被动电平配置 **($0118_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												PSL22	PSL21	PSL12	PSL11
r												rw	rw	rw	rw

域	位	类型	描述
PSL11	0	rw	**CCU8x.OUTy0 的输出被动电平** 该域控制 CCU8x.OUTy0 的被动电平。 0_B 被动电平为低电平 1_B 被动电平为高电平 写操作总是寻址映射寄存器，而读操作总是返回当前使用值。
PSL12	1	rw	**CCU8x.OUTy1 的输出被动电平** 该域控制 CCU8x.OUTy1 的被动电平。 0_B 被动电平为低电平 1_B 被动电平为高电平 写操作总是寻址映射寄存器，而读操作总是返回当前使用值。
PSL21	2	rw	**CCU8x.OUTy2 的输出被动电平** 该域控制 CCU8x.OUTy2 的被动电平。 0_B 被动电平为低电平 1_B 被动电平为高电平 写操作总是寻址映射寄存器，而读操作总是返回当前使用值。

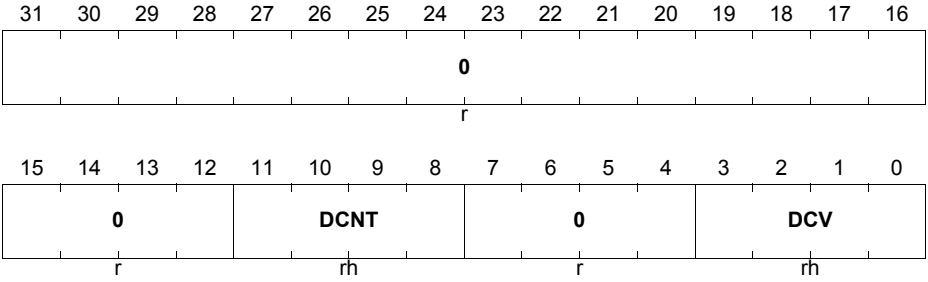
域	位	类型	描述
PSL22	3	rw	**CCU8x.OUTy3 的输出被动电平** 该域控制 CCU8x.OUTy3 的被动电平。 0_B 被动电平为低电平 1_B 被动电平为高电平 写操作总是寻址映射寄存器，而读操作总是返回当前使用值。
0	[31:4]	r	**保留** 读访问总是返回 0。

CC8yDIT

该寄存器保存着当前抖动比较值和抖动计数器值。

CC8yDIT (y = 0 - 3)

抖动配置 **($011C_H$ + 0100_H * y)** **复位值 : 00000000_H**



域	位	类型	描述
DCV	[3:0]	rh	**抖动比较值** 该域包含用于抖动比较的值。 当发生一次映射传送时，该值被更新为 **CC8yDITS**.DCVS。
DCNT	[11:8]	rh	**抖动计数器的当前值**
0	[7:4], [31:12]	r	**保留** 读访问总是返回 0。

CC8yDITS

该寄存器包含下一次发生映射传送时将要加载到 **CC8yDIT**.DCV 的值。

CC8yDITS (y = 0 - 3)

抖动映射寄存器 **(0120_H + 0100_H * y)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												**DCVS**			
r												rw			

域	位	类型	描述
DCVS	[3:0]	rw	**抖动映射比较值** 该域包含在下一次映射传送过程中将要设置到抖动比较值 **CC8yDIT**.DCV 中的值。
0	[31:4]	r	**保留** 读访问总是返回 0。

CC8yPSC

该寄存器包含在重新启动期间要加载到预分频器的值。

CC8yPSC (y = 0 - 3)

预分频器控制 **(0124_H + 0100_H * y)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												**PSIV**			
r												rw			

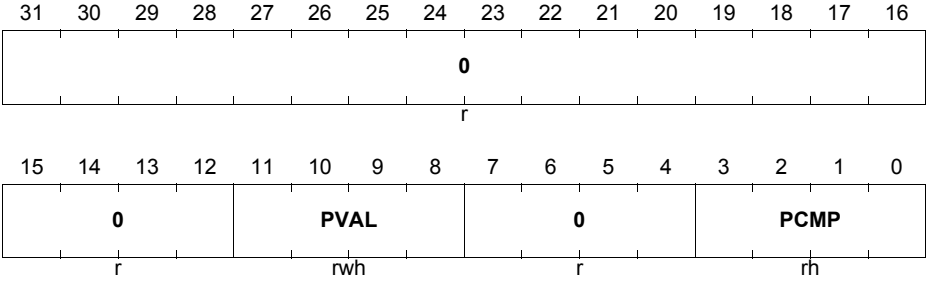
域	位	类型	描述
PSIV	[3:0]	rw	**预分频器初始值** 该域包含启动时加载到预分频器的值。 在采用浮动预分频器的情况下，当发生一次定时器比较匹配和预分频器比较匹配时，或者当一次捕获事件被触发时，加载该值。
0	[31:4]	r	**保留** 读访问总是返回 0。

CC8yFPC

该寄存器包含使用的浮动预分频器比较值和当前预分频器分频值。

CC8yFPC (y = 0 - 3)

浮动预分频器控制 **(0128_H + 0100_H * y)** **复位值 : 00000000_H**



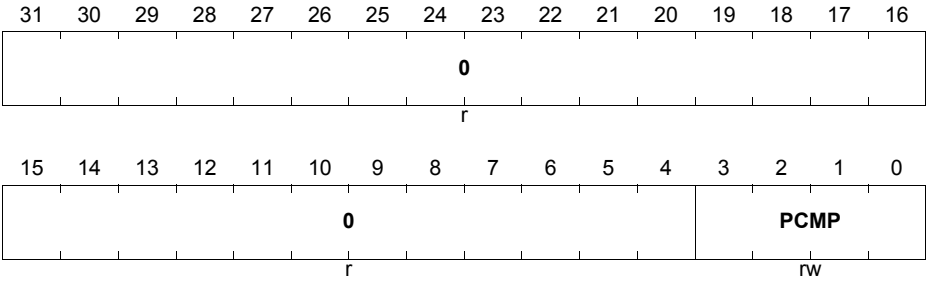
域	位	类型	描述
PCMP	[3:0]	rh	**浮动预分频器比较值** 该域包含用来与当前预分频器的值进行比较的值。比较操作由定时器比较匹配事件触发。详见 **20.2.13.2 节**。
PVAL	[11:8]	rwh	**当前预分频器值** 详见 **表 20-8**。只有在预分频器被停止时才能向该寄存器写入。在不使用浮动预分频器模式时，该值与 **CC8yPSC**.PSIV 值相等。
0	[7:4], [15:12], [31:16]	r	**保留** 读访问总是返回 0。

CC8yFPCS

该寄存器包含在下一次映射传送更新过程中将要被传送到 **CC8yFPC**.PCMP 域的值。

CC8yFPCS (y = 0 - 3)

浮动预分频器映射 **($012C_H + 0100_H * y$)** **复位值 : 00000000_H**



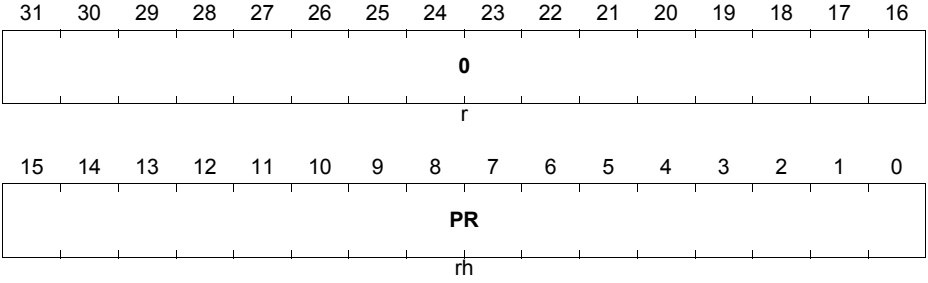
域	位	类型	描述
PCMP	[3:0]	rw	**浮动预分频器映射比较值** 该域包含在下一次映射传送过程中将要设置到 **CC8yFPC**.PCMP 域的值。详见**表 20-8**。
0	[31:4]	r	**保留** 读访问总是返回 0。

CC8yPR

该寄存器包含定时器周期的当前值。

CC8yPR (y = 0 - 3)

定时器周期值 **($0130_H + 0100_H * y$)** **复位值 : 00000000_H**

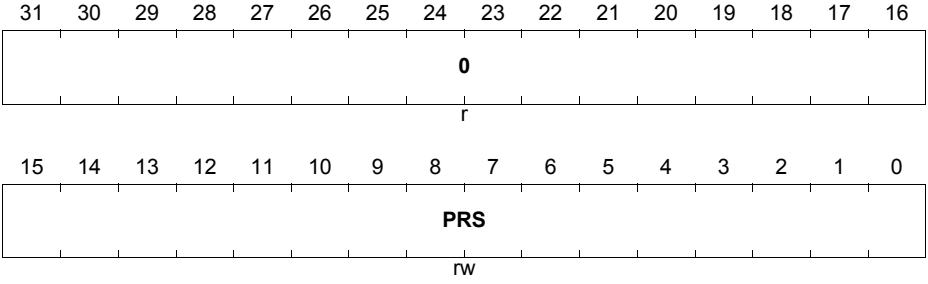


域	位	类型	描述
PR	[15:0]	rh	**周期寄存器** 包含定时器的周期值。
0	[31:16]	r	**保留** 读访问总是返回 0。

CC8yPRS

该寄存器包含在下一次映射传送发生时将要被传送到**CC8yPR**.PR位域的定时器周期值。

CC8yPRS (y = 0 - 3)
定时器映射周期值 **(0134_H + 0100_H * y)** **复位值 : 00000000_H**

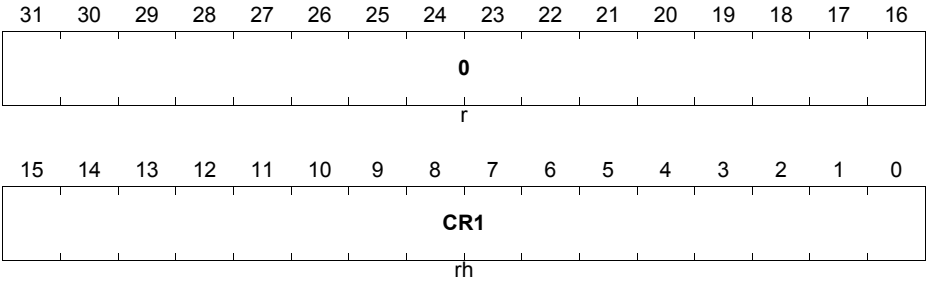


域	位	类型	描述
PRS	[15:0]	rw	**周期寄存器** 包含在下一次映射传送发生时要被传送到 **CC8yPR**.PR 域的定时器周期值。
0	[31:16]	r	**保留** 读访问总是返回 0。

CC8yCR1

该寄存器包含通道 1 的定时器比较值。

CC8yCR1 (y = 0 - 3)
通道 1 比较值 **(0138_H + 0100_H * y)** **复位值 : 00000000_H**



域	位	类型	描述
CR1	[15:0]	rh	**通道 1 比较寄存器** 包含定时器比较值。 写操作总是寻址映射寄存器，而读操作总是返回当前值。 *注： 在捕获模式，当选择一个外部信号来捕获定时器值到捕获寄存器 0 和 1 时，不能对 CR 进行写访问。读访问总是返回 0。*
0	[31:16]	r	**保留** 读访问总是返回 0。

CC8yCR1S

该寄存器包含在下一次映射传送发生时将要被加载到 **CC8yCR1**.CR 域的值。

CC8yCR1S (y = 0 - 3)

通道 1 比较映射值 **($013C_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR1S															
rw															

域	位	类型	描述
CR1S	[15:0]	rw	**通道 1 的映射比较寄存器** 包含定时器比较值，该值在下一次映射传送发生时被传送到 **CC8yCR1**.CR1 域。 *注： 在捕获模式，当选择一个外部信号来捕获定时器值到捕获寄存器 0 和 1 时，不能对 CR 进行写访问。读访问总是返回 0。*
0	[31:16]	r	**保留** 读访问总是返回 0。

CC8yCR2

该寄存器包含通道 2 的定时器比较值。

CC8yCR2 (y = 0 - 3)

通道 2 比较值 **($0140_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

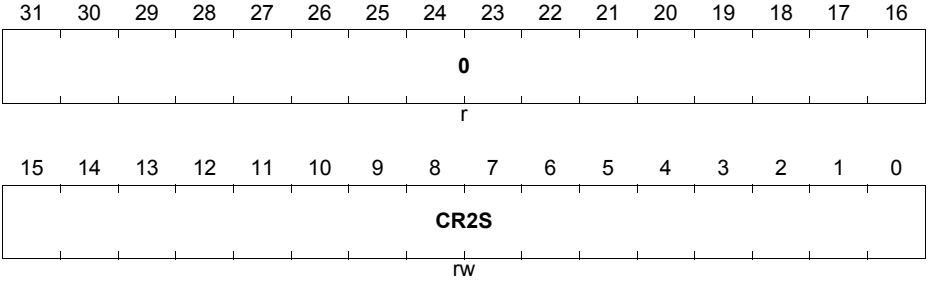
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CR2															
rh															

域	位	类型	描述
CR2	[15:0]	rh	**通道 2 比较寄存器** 包含定时器比较值。 写操作总是寻址映射寄存器，而读操作总是返回当前值。 *注：　在捕获模式，当选择一个外部信号来捕获定时器值到捕获寄存器 2 和 3 时，不能对 CR 进行写访问。读访问总是返回 0。*
0	[31:16]	r	**保留** 读访问总是返回 0。

CC8yCR2S

该寄存器包含在下一次映射传送发生时将要被加载到 **CC8yCR2**.CR 域的值。

CC8yCR2S (y = 0 - 3)
通道 2 比较映射值 **(0144_H + 0100_H * y)** **复位值 : 00000000_H**



域	位	类型	描述
CR2S	[15:0]	rw	**通道 2 的映射比较寄存器** 包含定时器比较值，该值在下一次映射传送发生时被传送到 **CC8yCR2**.CR2 域。 *注：　在捕获模式，当选择一个外部信号来捕获定时器值到捕获寄存器 2 和 3 时，不能对 CR 进行写访问。读访问总是返回 0。*
0	[31:16]	r	**保留** 读访问总是返回 0。

CC8yCHC

该寄存器包含两个比较通道的输出连接配置和非对称模式使能配置。

CC8yCHC (y = 0 - 3)
通道控制 **($0148_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

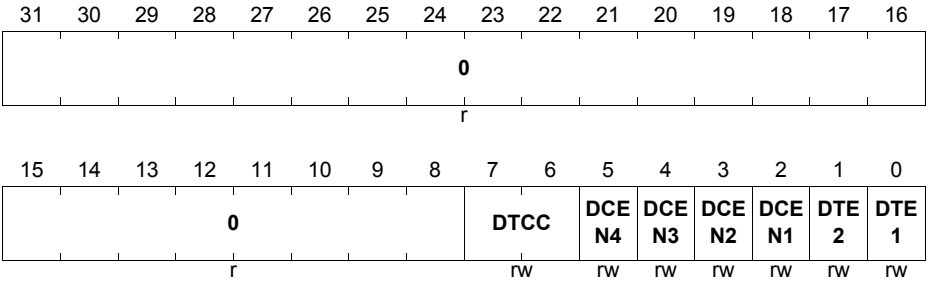
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											OCS4	OCS3	OCS2	OCS1	ASE
r											rw	rw	rw	rw	rw

域	位	类型	描述
ASE	0	rw	**非对称 PWM 模式使能** 0_B 禁止非对称 PWM 1_B 使能非对称 PWM
OCS1	1	rw	**CCU8x.OUTy0 输出选择** 0_B CC8yST1 信号通路连接到 CCU8x.OUTy0 1_B 反相的 CC8yST1 信号通路连接到 CCU8x.OUTy0
OCS2	2	rw	**CCU8x.OUTy1 输出选择** 0_B 反相的 CC8yST1 信号通路连接到 CCU8x.OUTy1 1_B CC8yST1 信号通路连接到 CCU8x.OUTy1
OCS3	3	rw	**CCU8x.OUTy2 输出选择** 0_B CC8yST2 信号通路连接到 CCU8x.OUTy2 1_B 反相的 CCST2 信号通路连接到 CCU8x.OUTy2
OCS4	4	rw	**CCU8x.OUTy3 输出选择** 0_B 反相的 CC8yST2 信号通路连接到 CCU8x.OUTy3 1_B CC8yST2 信号通路连接到 CCU8x.OUTy3
0	[31:5]	r	**保留** 读访问总是返回 0。

CC8yDTC

该寄存器包含死区时间发生器的配置。

CC8yDTC (y = 0 - 3)
死区时间控制 **($014C_H + 0100_H * y$)** **复位时间 : 00000000_H**



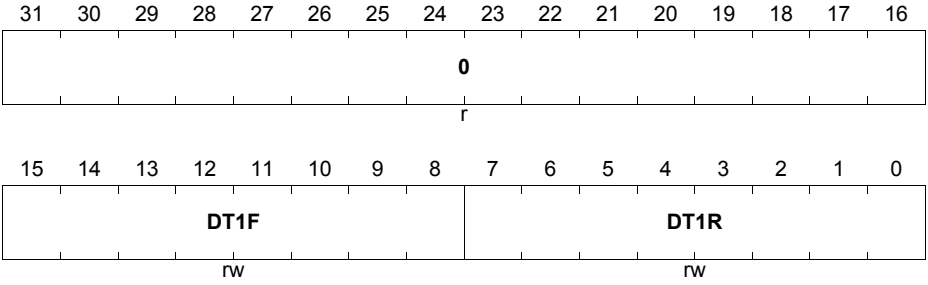
域	位	类型	描述
DTE1	0	rw	**通道 1 死区时间使能** 该域使能比较通道 1 的死区时间计数器。 0_B 禁止通道 1 的死区时间 1_B 使能通道 1 的死区时间
DTE2	1	rw	**通道 2 死区时间使能** 该域使能比较通道 2 的死区时间计数器。 0_B 禁止通道 2 的死区时间 1_B 使能通道 2 的死区时间
DCEN1	2	rw	**CC8yST1 死区时间使能** 0_B 禁止 CC8yST1 通路的死区时间 1_B 使能 CC8yST1 通路的死区时间
DCEN2	3	rw	**反相 CC8yST1 死区时间使能** 0_B 禁止反相 CC8yST1 通路的死区时间 1_B 使能反相 CC8yST1 通路的死区时间
DCEN3	4	rw	**CC8yST2 死区时间使能** 0_B 禁止 CC8yST2 通路的死区时间 1_B 使能 CC8yST2 通路的死区时间
DCEN4	5	rw	**反相 CC8yST2 死区时间使能** 0_B 禁止反相 CC8yST2 通路的死区时间 1_B 使能反相 CC8yST2 通路的死区时间

域	位	类型	描述
DTCC	[7:6]	rw	**死区时间时钟控制** 该域控制死区时间计数器的预分频器时钟配置。 00_B f_{tclk} 01_B $f_{tclk}/2$ 10_B $f_{tclk}/4$ 11_B $f_{tclk}/8$
0	[15:8], [31:16]	r	**保留** 读访问总是返回 0。

CC8yDC1R

该寄存器包含通道 1 的死区时间值。

CC8yDC1R (y = 0 - 3)
通道 1 死区时间值 **(0150_H + 0100_H * y)** **复位值 : 00000000_H**

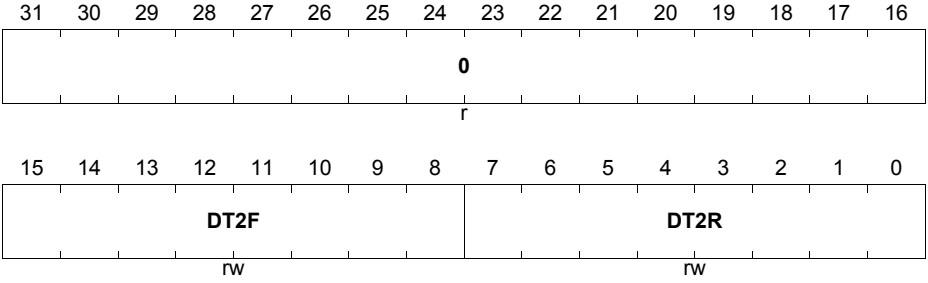


域	位	类型	描述
DT1R	[7:0]	rw	**通道 1 上升跳变死区时间值** 该域包含每当 CC8yST1 发生从 0 到 1 的跳变时要施加的延时值。
DT1F	[15:8]	rw	**通道 1 下降跳变死区时间值** 该域包含每当 CC8yST1 发生从 1 到 0 的跳变时要施加的延时值。
0	[31:16]	r	**保留** 读访问总是返回 0。

CC8yDC2R

该寄存器包含通道 2 的死区时间值。

CC8yDC2R (y = 0 - 3)
通道 2 死区时间值 **(0154_H + 0100_H * y)** **复位值 : 00000000_H**

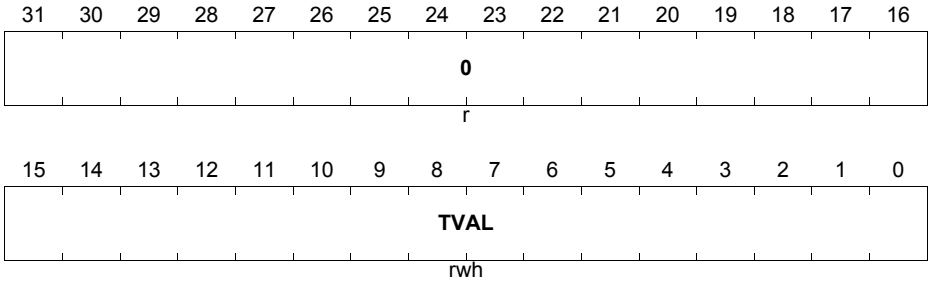


域	位	类型	描述
DT2R	[7:0]	rw	**通道 2 上升跳变死区时间值** 该域包含每当 CC8yST2 发生从 0 到 1 的跳变时要施加的延时值。
DT2F	[15:8]	rw	**通道 2 下降跳变死区时间值** 该域包含每当 CC8yST2 发生从 1 到 0 的跳变时要施加的延时值。
0	[31:16]	r	**保留** 读访问总是返回 0。

CC8yTIMER

该寄存器包含定时器的当前值。

CC8yTIMER (y = 0 - 3)
定时器值 **($0170_H + 0100_H * y$)** **复位值 : 00000000_H**

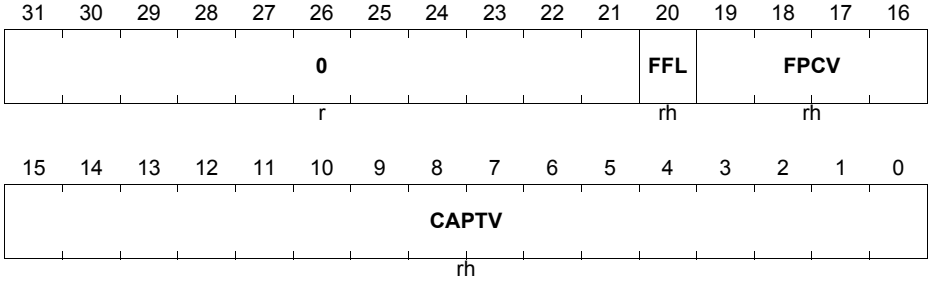


域	位	类型	描述
TVAL	[15:0]	rwh	**定时器值** 该域包含着定时器的当前值。 仅在定时器被停止时才可以进行写访问。
0	[31:16]	r	**保留** 读访问总是返回 0。

CC8yC0V

该寄存器包含与捕获 0 位域相关联的值。

CC8yC0V (y = 0 - 3)
捕获寄存器 0 **($0174_H + 0100_H * y$)** **复位值 : 00000000_H**



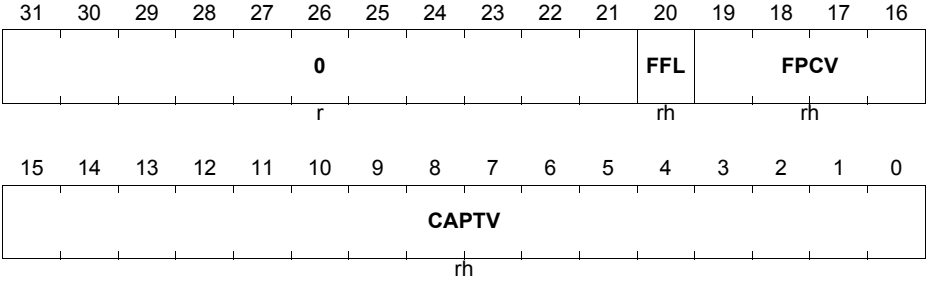
域	位	类型	描述
CAPTV	[15:0]	rh	**捕获值** 该域包含捕获寄存器 0 的值。详见 图 20-43。 在比较模式，读访问总是返回 0。
FPCV	[19:16]	rh	**预分频器值** 该域包含在捕获到捕获寄存器 0 的事件发生时的预分频器值。 在比较模式，读访问总是返回 0。
FFL	20	rh	**满标志** 该位指示在上一次读访问后是否有新值被捕获到捕获寄存器 0。详见图 20-43。在比较模式，读访问总是返回 0。 0_B 没有新值被捕获到特定的捕获寄存器 1_B 一个新值被捕获到特定的捕获寄存器
0	[31:21]	r	**保留** 读访问总是返回 0。

CC8yC1V

该寄存器包含与捕获 1 位域相关联的值。

CC8yC1V (y = 0 - 3)

捕获寄存器 1 **($0178_H + 0100_H * y$)** **复位值 : 00000000_H**



域	位	类型	描述
CAPTV	[15:0]	rh	**捕获值** 该域包含捕获寄存器 1 的值。详见图 20-43。 在比较模式，读访问总是返回 0。

域	位	类型	描述
FPCV	[19:16]	rh	**预分频器值** 该域包含在捕获到捕获寄存器 1 的事件发生时的预分频器值。 在比较模式，读访问总是返回 0。
FFL	20	rh	**满标记** 该位指示在上一次读访问之后是否有新值被捕获到捕获寄存器 1。详见图 20-43。在比较模式，读访问总是返回 0。 0_B 没有新值被捕获到特定的捕获寄存器 1_B 一个新值被捕获到特定的捕获寄存器
0	[31:21]	r	**保留** 读访问总是返回 0。

CC8yC2V

该寄存器包含与捕获 2 位域相关联的值。

CC8yC2V (y = 0 - 3)
捕获寄存器 2 **($017C_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0											FFL	FPCV			
r											rh	rh			

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CAPTV															
rh															

域	位	类型	描述
CAPTV	[15:0]	rh	**捕获值** 该位域包含捕获寄存器 2 的值。详见图 20-43。 在比较模式，读访问总是返回 0。
FPCV	[19:16]	rh	**预分频器值** 该位域包含在捕获到捕获寄存器 2 的事件发生时的预分频器值。在比较模式，读访问总是返回 0。

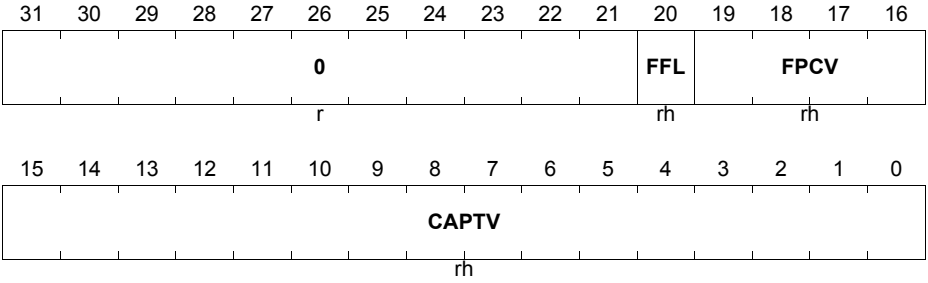
域	位	类型	描述
FFL	20	rh	**满标记** 该位指示在上一次读访问之后是否有新值被捕获到捕获寄存器 2。详见**图 20-43**。在比较模式，读访问总是返回 0。 0_B 没有新值被捕获到特定的捕获寄存器 1_B 一个新值被捕获到特定的捕获寄存器
0	[31:21]	r	**保留** 读访问总是返回 0

CC8yC3V

该寄存器包含与捕获 3 位域相关联的值。

CC8yC3V (y = 0 - 3)

捕获寄存器 3 **(0180_H + 0100_H * y)** **复位值 : 00000000_H**



域	位	类型	描述
CAPTV	[15:0]	rh	**捕获值** 该域包含捕获寄存器 3 的值。详见**图 20-43**。在比较模式，读访问总是返回 0。
FPCV	[19:16]	rh	**预分频器值** 该域包含在捕获到捕获寄存器 3 的事件发生时的预分频器值。 在比较模式，读访问总是返回 0。

域	位	类型	描述
FFL	20	rh	**满标记** 该位指示在上一次读访问之后是否有新值被捕获到捕获寄存器 3。详见**图 20-43**。在比较模式，读访问总是返回 0。 0_B 没有新值被捕获到特定的捕获寄存器 1_B 一个新值被捕获到特定的捕获寄存器
0	[31:21]	r	**保留** 读访问总是返回 0

CC8yINTS

该寄存器包含所有中断源的状态。

CC8yINTS (y = 0 - 3)

中断状态 **($01A0_H$ + 0100_H * y)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				TRPF	E2AS	E1AS	E0AS	0		CMD2S	CMU2S	CMD1S	CMU1S	OMDS	PMUS
r				rh	rh	rh	rh	r		rh	rh	rh	rh	rh	rh

域	位	类型	描述
PMUS	0	rh	**向上计数时的周期匹配** 0_B 未检测到向上计数时的周期匹配 1_B 已检测到向上计数时的周期匹配
OMDS	1	rh	**向下计数时的 1 匹配** 0_B 未检测到向下计数时的 1 匹配 1_B 已检测到向下计数时的 1 匹配
CMU1S	2	rh	**向上计数时的通道 1 比较匹配** 0_B 未检测到向上计数时的比较匹配 1_B 已检测到向上计数时的比较匹配
CMD1S	3	rh	**向下计数时的通道 1 比较匹配** 0_B 未检测到向下计数时的比较匹配 1_B 已检测到向下计数时的比较匹配

域	位	类型	描述
CMU2S	4	rh	**向上计数时的通道 2 比较匹配** 0_B 未检测到向上计数时的比较匹配 1_B 已检测到向上计数时的比较匹配
CMD2S	5	rh	**向下计数时的通道 2 比较匹配** 0_B 未检测到向下计数时的比较匹配 1_B 已检测到向下计数时的比较匹配
E0AS	8	rh	**事件 0 检测状态** 根据用户对 **CC8yINS**.EV0EM 的选择，可以在检测到上升沿、下降沿或任何一个边沿时将该位置 1。 0_B 未检测到事件 0 1_B 已检测到事件 0
E1AS	9	rh	**事件 1 检测状态** 根据用户对 **CC8yINS**.EV1EM 的选择，可以在检测到上升沿、下降沿或任何一个边沿时将该位置 1。 0_B 未检测到事件 1 1_B 已检测到事件 1
E2AS	10	rh	**事件 2 检测状态** 根据用户对 **CC8yINS**.EV1EM 的选择，可以在检测到上升沿、下降沿或任何一个边沿时将该位置 1。 0_B 未检测到事件 2 1_B 已检测到事件 2 *注： 如果该事件与陷阱功能相连，当定时器片退出陷阱状态时，该域被自动清除。*
TRPF	11	rh	**陷阱标志状态** 该位域包含陷阱标志的状态。
0	[7:6], [31:12]	r	**保留** 读访问总是返回 0。

CC8yINTE

通过该寄存器可以使能或禁止特定的中断源。

CC8yINTE (y = 0 - 3)

中断使能控制 **($01A4_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					E2AE	E1AE	E0AE	0		CMD2E	CMU2E	CMD1E	CMU1E	OME	PME
r					rw	rw	rw	r		rw	rw	rw	rw	rw	rw

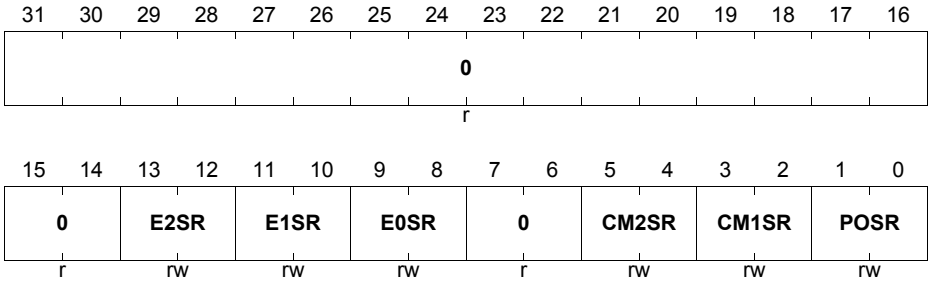
域	位	类型	描述
PME	0	rw	**向上计数周期匹配中断使能** 将该位设置为 1_B 时，在向上计数的情况下，每发生一次周期匹配就会产生一个中断脉冲。 0_B 禁止周期匹配中断 1_B 使能周期匹配中断
OME	1	rw	**向下计数时 1 匹配中断使能** 将该位设置为 1_B 时，在向下计数的情况下，每发生一次 1 匹配就会产生一个中断脉冲。 0_B 禁止 1 匹配中断 1_B 使能 1 匹配中断
CMU1E	2	rw	**向上计数时通道 1 比较匹配中断使能** 将该位设置为 1_B 时，在向上计数的情况下，每发生一次比较匹配就会产生一个中断脉冲。 0_B 禁止向上计数时的比较匹配中断 1_B 使能向上计数时的比较匹配中断
CMD1E	3	rw	**向下计数时通道 1 比较匹配中断使能** 将该位设置为 1_B 时，在向下计数的情况下，每发生一次比较匹配就会产生一个中断脉冲。 0_B 禁止向下计数时的比较匹配中断 1_B 使能向下计数时的比较匹配中断

域	位	类型	描述
CMU2E	4	rw	**向上计数时通道 2 比较匹配中断使能** 将该位设置为 1_B 时，在向上计数的情况下，每发生一次比较匹配就会产生一个中断脉冲。 0_B 禁止向上计数时的比较匹配中断 1_B 使能向上计数时的比较匹配中断
CMD2E	5	rw	**向下计数时通道 2 比较匹配中断使能** 将该位设置为 1_B 时，在向下计数的情况下，每发生一次比较匹配就会产生一个中断脉冲。 0_B 禁止向下计数时的比较匹配中断 1_B 使能向下计数时的比较匹配中断
E0AE	8	rw	**事件 0 中断使能** 将该位设置为 1_B 时，每当检测到事件 0 时就会产生一个中断脉冲。 0_B 禁止事件 0 检测中断 1_B 使能事件 0 检测中断
E1AE	9	rw	**事件 1 中断使能** 将该位设置为 1_B 时，每当检测到事件 1 时就会产生一个中断脉冲。 0_B 禁止事件 1 检测中断 1_B 使能事件 1 检测中断
E2AE	10	rw	**事件 2 中断使能** 将该位设置为 1_B 时，每当检测到事件 2 时就会产生一个中断脉冲。 0_B 禁止事件 2 检测中断 1_B 使能事件 2 检测中断
0	[7:6], [31:11]	r	**保留** 读访问总是返回 0

CC8ySRS

通过该寄存器可以选择将每个中断源转发到哪条服务请求线。

CC8ySRS (y = 0 - 3)
服务请求选择器 **($01A8_H + 0100_H * y$)** **复位值 : 00000000_H**



域	位	类型	描述
POSR	[1:0]	rw	**周期匹配 /1 匹配服务请求选择器** 该域选择将向上计数时周期匹配所产生的中断和向下计数时 1 匹配所产生的中断转发到哪条服务请求线。 00_B 转发到 CC8ySR0 01_B 转发到 CC8ySR1 10_B 转发到 CC8ySR2 11_B 转发到 CC8ySR3
CM1SR	[3:2]	rw	**通道 1 比较匹配服务请求选择器** 该域选择将通道 1 的向上计数比较匹配所产生的中断和向下计数比较匹配所产生的中断转发到哪条服务请求线。 00_B 转发到 CC8ySR0 01_B 转发到 CC8ySR1 10_B 转发到 CC8ySR2 11_B 转发到 CC8ySR3
CM2SR	[5:4]	rw	**通道 2 比较匹配服务请求选择器** 该域选择将通道 2 的向上计数比较匹配所产生的中断和向下计数比较匹配所产生的中断转发到哪条服务请求线。 00_B 转发到 CC8ySR0 01_B 转发到 CC8ySR1 10_B 转发到 CC8ySR2 11_B 转发到 CC8ySR3

域	位	类型	描述
E0SR	[9:8]	rw	**事件 0 服务请求选择器** 该域选择将事件 0 检测所产生的中断转发到哪条服务请求线。 00_B 转发到 CCvySR0 01_B 转发到 CC8ySR1 10_B 转发到 CC8ySR2 11_B 转发到 CC8ySR3
E1SR	[11:10]	rw	**事件 1 服务请求选择器** 该域选择将事件 1 检测所产生的中断转发到哪条服务请求线。 00_B 转发到 CC8ySR0 01_B 转发到 CC8ySR1 10_B 转发到 CC8ySR2 11_B 转发到 CC8ySR3
E2SR	[13:12]	rw	**事件 2 服务请求选择器** 该域选择将事件 2 检测所产生的中断转发到哪条服务请求线。 00_B 转发到 CC8ySR0 01_B 转发到 CCvySR1 10_B 转发到 CC8ySR2 11_B 转发到 CC8ySR3
0	[7:6], [31:14]	r	**保留** 读访问总是返回 0。

CC8ySWS

通过该寄存器可以用软件将一个特定的中断状态标志置 1。

CC8ySWS (y = 0 - 3)

中断状态置位 **($01AC_H + 0100_H * y$)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				STRPF	SE2A	SE1A	SE0A	0		SCM2D	SCM2U	SCM1D	SCM1U	SOM	SPM
r				w	w	w	w	r		w	w	w	w	w	w

域	位	类型	描述
SPM	0	w	**向上计数时的周期匹配中断置位** 向该位域写入 1_B 会将 **CC8yINTS**.PMUS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SOM	1	w	**向下计数时的 1 匹配中断置位** 向该域写入 1_B 会将 **CC8yINTS**.OMDS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SCM1U	2	w	**向上计数时的通道 1 比较匹配中断置位** 向该域写入 1_B 会将 **CC8yINTS**.CMU1S 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SCM1D	3	w	**向下计数时的通道 1 比较匹配中断置位** 向该域写入 1_B 会将 **CC8yINTS**.CMD1S 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SCM2U	4	w	**向上计数时的比较匹配中断置位** 向该域写入 1_B 会将 **CC8yINTS**.CMU2S 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SCM2D	5	w	**向下计数时的比较匹配中断置位** 向该域写入 1_B 会将 **CC8yINTS**.CMD2S 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SE0A	8	w	**事件 0 检测中断置位** 向该域写入 1_B 会将 **CC8yINTS**.E0AS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SE1A	9	w	**事件 1 检测中断置位** 向该域写入 1_B 会将 **CC8yINTS**.E1AS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。
SE2A	10	w	**事件 2 检测中断置位** 向该域写入 1_B 会将 **CC8yINTS**.E2AS 位置 1。如果该中断源被使能，则会产生一个中断脉冲。读访问总是返回 0。

域	位	类型	描述
STRPF	11	w	**陷阱标志状态置位** 向该域写入 1_B 会将 **CC8yINTS**.TRPF 位置 1。 读访问总是返回 0。
0	[7:6], [31:12]	r	**保留** 读访问总是返回 0。

CC8ySWR

通过该寄存器可以用软件清除一个特定的中断状态标志。

CC8ySWR (y = 0 - 3)

中断状态清除 **($01B0_H$ + 0100_H * y)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				RTRPF	RE2A	RE1A	RE0A	0		RCM2D	RCM2U	RCM1D	RCM1U	ROM	RPM
r				w	w	w	w	r		w	w	w	w	w	w

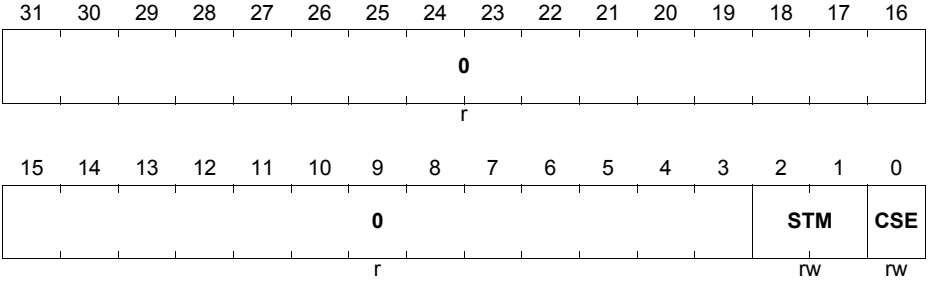
域	位		描述
RPM	0	w	**向上计数时的周期匹配中断清除** 向该位写入 1_B 会清除 **CC8yINTS**.PMUS 位。读访问总是返回 0。
ROM	1	w	**向下计数时的 1 匹配中断清除** 向该位写入 1_B 会清除 **CC8yINTS**.OMDS 位。读访问总是返回 0。
RCM1U	2	w	**向上计数时的通道 1 比较匹配中断清除** 向该位写入 1_B 会清除 **CC8yINTS**.CMU1S 位。读访问总是返回 0。
RCM1D	3	w	**向下计数时的通道 1 比较匹配中断清除** 向该位写入 1_B 会清除 **CC8yINTS**.CMD1S 位。读访问总是返回 0。

域	位		描述
RCM2U	4	w	**向上计数时的通道 2 比较匹配中断清除** 向该位写入 1_B 会清除 **CC8yINTS**.CMU2S 位。读访问总是返回 0。
RCM2D	5	w	**向下计数时的通道 2 比较匹配中断清除** 向该位写入 1_B 会清除 **CC8yINTS**.CMD2S 位。读访问总是返回 0。
RE0A	8	w	**事件 0 检测中断清除** 向该位写入 1_B 会清除 **CC8yINTS**.E0AS 位。读访问总是返回 0。
RE1A	9	w	**事件 1 检测中断清除** 向该位写入 1_B 会清除 **CC8yINTS**.E1AS 位。读访问总是返回 0。
RE2A	10	w	**事件 2 检测中断清除** 向该位写入 1_B 会清除 **CC8yINTS**.E2AS 位。读访问总是返回 0。
RTRPF	11	w	**陷阱标志状态清除** 向该位写入 1_B 会清除 **CC8yINTS**.TRPF 位。若 **CC8yTC**.TRPEN = 1_B，则清除操作无效，陷阱状态仍然有效。 读访问总是返回 0。
0	[7:6], [31:12]	r	**保留** 读访问总是返回 0。

CC8ySTC

通过该寄存器可以为映射传送机制配置扩展选项。

CC8ySTC (y = 0 - 3)
映射传送控制 **($01B4_H + 0100_H * y$)** **复位值 : 00000000_H**



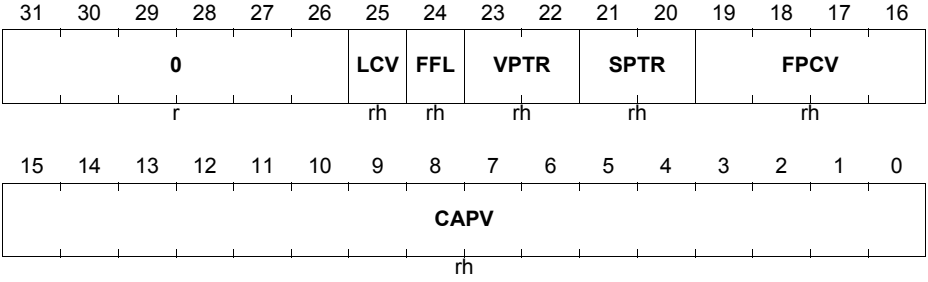
域	位		描述
CSE	0	rw	**级联映射传送使能** 0_B 禁止级联映射传送 1_B 使能级联映射传送
STM	[2:1]	rw	**映射传送模式** 00_B 映射传送在周期匹配和 1 匹配时完成。 01_B 映射传送仅在周期匹配时完成。 10_B 映射传送仅在 1 匹配时完成 11_B 保留 *注： 该域仅在定时器工作在中心对齐模式时有效。*
0	[31:3]	r	**保留** 读访问总是返回 0。

CC8yECRD0

通过该寄存器可以回读与捕获触发信号 0 相连的捕获功能的 FIFO 结构。回读操作仅在 **CC8yTC**.ECM = 1_B 时有效。

CC8yECRD0 (y = 0 - 3)

扩展回读 0 **($01B8_H + 0100_H * y$)** **复位值 : 00000000_H**



域	位	类型	描述
CAPV	[15:0]	rh	**定时器捕获值** 该域包含定时器捕获的值
FPCV	[19:16]	rh	**预分频器捕获值** 该域包含与特定的 CAPV 域相关联的预分频器时钟分频值

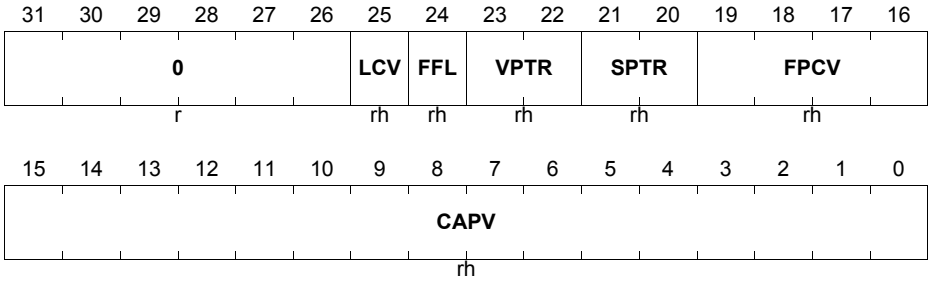
域	位	类型	描述
SPTR	[21:20]	rh	**定时器片指针** **该域指示捕获值所在的定时器片的索引。** 00_B CC80 01_B CC81 10_B CC82 11_B CC83
VPTR	[23:22]	rh	**捕获寄存器指针** **该域指示捕获值所在的捕获寄存器的索引。** 00_B 捕获寄存器 0 01_B 捕获寄存器 1 10_B 捕获寄存器 2 11_B 捕获寄存器 3
FFL	24	rh	**满标志** 该位指示相关联的捕获寄存器是否包含一个新值。 0_B 没有新值被捕获到该寄存器 1_B 一个新值被捕获到该寄存器
LCV	25	rh	**丢失捕获值** 该域指示当 FIFO 结构已满时在两次读 ECRD0 之间是否产生了一个捕获触发信号。如果在两次读操作之间产生了一个捕获触发信号，则丢失了一个捕获值。只要发生对 ECRD 的读操作，该域由硬件自动清除。 0_B 没有丢失捕获值 1_B 丢失一个捕获值
0	[31:26]	r	**保留** 读访问总是返回 0

CC8yECRD1

通过该寄存器可以回读与捕获触发信号 1 相连的捕获功能的 FIFO 结构。回读操作仅在 **CC8yTC**.ECM = 1_B 时有效。

CC8yECRD1 (y = 0 - 3)

扩展回读 1 **($01BC_H + 0100_H * y$)** **复位值 : 00000000_H**



域	位	类型	描述
CAPV	[15:0]	rh	**定时器捕获值** 该域包含定时器捕获的值
FPCV	[19:16]	rh	**预分频器捕获值** 该域包含与特定的 CAPV 域相关联的预分频器时钟分频值
SPTR	[21:20]	rh	**定时器片指针** 该域指示捕获值所在的定时器片的索引。 00_B CC80 01_B CC81 10_B CC82 11_B CC83
VPTR	[23:22]	rh	**捕获寄存器指针** 该域指示捕获值所在的捕获寄存器的索引。 00_B 捕获寄存器 0 01_B 捕获寄存器 1 10_B 捕获寄存器 2 11_B 捕获寄存器 3
FFL	24	rh	**满标记** 该位指示相关联的捕获寄存器包含一个新值。 0_B 没有新值被捕获到该寄存器 1_B 一个新值被捕获到该寄存器

域	位	类型	描述
LCV	25	rh	**丢失捕获值** 该域指示当 FIFO 结构已满时在两次读 ECRD0 之间是否产生了一个捕获触发信号。如果在两次读操作之间产生了一个捕获触发信号，则丢失了一个捕获值。只要发生对 ECRD 的读操作，该域由硬件自动清除。 0_B 没有丢失捕获值 1_B 丢失一个捕获值
0	[31:26]	r	**保留** 读访问总是返回 0

20.8 互连

涉及 “ 全局引脚 ” 的表格中都包含每个模块的所有定时器片共同的输入 / 输出。

关于 GPIO 连接，请参见端口一章。

20.8.1 CCU80 引脚

表 20-14 CCU80 引脚连接

全局输入 / 输出	I/O	连接到	描述
CCU80.MCLK	I	PCLK	内核时钟
CCU80.CLKA	I	未连接	预分频器的另一计数源
CCU80.CLKB	I	ERU0.IOUT0	预分频器的另一计数源
CCU80.CLKC	I	ERU0.IOUT1	预分频器的另一计数源
CCU80.MCSS	I	POSIF0.OUT6	多通道序列与映射传送触发信号之间的同步信号
CCU80.IGBTA	I	CCU40.ST3	奇偶校验器延迟结束触发信号
CCU80.IGBTB	I	CCU40.SR3	奇偶校验器延迟结束触发信号
CCU80.IGBTC	I	CCU40.ST0	奇偶校验器延迟结束触发信号
CCU80.IGBTD	I	CCU40.SR0	奇偶校验器延迟结束触发信号
CCU80.IGBTO	O	CCU40.IN3H;	奇偶校验器延迟开始触发信号
CCU80.SR0	O	NVIC;	服务请求线
CCU80.SR1	O	NVIC; POSIF0.MSETE;	服务请求线

表 20-14 CCU80 引脚连接 (续表)

全局输入 / 输出	I/O	连接到	描述
CCU80.SR2	O	VADC0.BGREQTRI; VADC0.G0REQTRI; VADC0.G1REQTRI; ERU0.OGU03; ERU0.OGU13;	服务请求线
CCU80.SR3	O	VADC0.BGREQTRJ; VADC0.G0REQTRJ; VADC0.G1REQTRJ; ERU0.OGU23; ERU0.OGU33;	服务请求线

表 20-15 CCU80 - CC80 引脚连接

输入 / 输出	I/O	连接到	描述
CCU80.IN0A	I	P0.12	通用功能
CCU80.IN0B	I	P0.4	通用功能
CCU80.IN0C	I	CCU40.GP02	通用功能
CCU80.IN0D	I	POSIF0.OUT2	通用功能
CCU80.IN0E	I	POSIF0.OUT5	通用功能
CCU80.IN0F	I	ERU0.PDOUT0	通用功能
CCU80.IN0G	I	ERU0.IOUT0	通用功能
CCU80.IN0H	I	SCU.GSC80	通用功能
CCU80.IN0I	I	BCCU0.OUT0	通用功能
CCU80.IN0J	I	BCCU0.OUT1	通用功能
CCU80.IN0K	I	CCU40.SR2	通用功能
CCU80.IN0L	I	ERU0.PDOUT1	通用功能
CCU80.IN0M	I	CCU80.GP10	通用功能
CCU80.IN0N	I	CCU80.ST1	通用功能
CCU80.IN0O	I	CCU80.ST2	通用功能
CCU80.IN0P	I	CCU80.ST3	通用功能
CCU80.MCI00	I	POSIF0.MOUT[0]	CCST1 的多通道序列输入
CCU80.MCI01	I	POSIF0.MOUT[1]	反相 CCST1 的多通道序列输入
CCU80.MCI02	I	POSIF0.MOUT[2]	CCST2 的多通道序列输入

表 20-15 CCU80 - CC80 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
CCU80.MCI03	I	POSIF0.MOUT[3]	反相 CCST2 的多通道序列输入
CCU80.OUT00	O	P0.0; P1.0;	通道 1 的定时器片比较输出。可以是 CCST1 或反相的 CCST1。
CCU80.OUT01	O	P0.1; P0.5; P1.1;	通道 1 的定时器片比较输出。可以是 CCST1 或反相的 CCST1。
CCU80.OUT02	O	P0.2;	通道 2 的定时器片比较输出。可以是 CCST2 或反相的 CCST2。
CCU80.OUT03	O	P0.3;	通道 2 的定时器片比较输出。可以是 CCST2 或反相的 CCST2。
CCU80.GP00	O	CCU80.IN3M	事件 0 的选择信号
CCU80.GP01	O	未连接	事件 1 的选择信号
CCU80.GP02	O	未连接	事件 2 的选择信号
CCU80.ST0	O	CCU80.IN1N; CCU80.IN2N; CCU80.IN3N; VADC0.BGREQGTM; VADC0.G0REQGTM; VADC0.G1REQGTM;	状态位多路复用器的输出，可以是 CCST1 或 CCST2。
CCU80.ST0A	O	未连接	通道 1 状态位：CCST1
CCU80.ST0B	O	未连接	通道 1 状态位：CCST2
CCU80.PS0	O	未连接	多通道序列同步触发信号：当向上计数 (边沿对齐) 时为 PM，当向下计数 (中心对齐) 时为 OM。

表 20-16 CCU80 - CC81 引脚连接

输入 / 输出	I/O	连接到	描述
CCU80.IN1A	I	P0.12	通用功能
CCU80.IN1B	I	P0.5;	通用功能
CCU80.IN1C	I	CCU40.GP12	通用功能
CCU80.IN1D	I	POSIF0.OUT2	通用功能
CCU80.IN1E	I	POSIF0.OUT5	通用功能
CCU80.IN1F	I	ERU0.PDOUT1	通用功能

表 20-16 CCU80 - CC81 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
CCU80.IN1G	I	ERU0.IOUT1	通用功能
CCU80.IN1H	I	SCU.GSC80	通用功能
CCU80.IN1I	I	BCCU0.OUT2	通用功能
CCU80.IN1J	I	BCCU0.OUT3	通用功能
CCU80.IN1K	I	CCU40.SR2	通用功能
CCU80.IN1L	I	ERU0.PDOUT0	通用功能
CCU80.IN1M	I	CCU80.GP20	通用功能
CCU80.IN1N	I	CCU80.ST0	通用功能
CCU80.IN1O	I	CCU80.ST2	通用功能
CCU80.IN1P	I	CCU80.ST3	通用功能
CCU80.MCI10	I	POSIF0.MOUT[4]	CCST1 的多通道序列输入
CCU80.MCI11	I	POSIF0.MOUT[5]	反相 CCST1 的多通道序列输入
CCU80.MCI12	I	POSIF0.MOUT[6]	CCST2 的多通道序列输入
CCU80.MCI13	I	POSIF0.MOUT[7]	反相 CCST2 的多通道序列输入
CCU80.OUT10	O	P0.2; P0.7; P1.2;	通道 1 的定时器片比较输出。可以是 CCST1 或反相的 CCST1。
CCU80.OUT11	O	P0.3; P0.6; P1.3;	通道 1 的定时器片比较输出。可以是 CCST1 或反相的 CCST1。
CCU80.OUT12	O	P0.5;	通道 2 的定时器片比较输出。可以是 CCST2 或反相的 CCST2。
CCU80.OUT13	O	P0.4;	通道 2 的定时器片比较输出。可以是 CCST2 或反相的 CCST2。
CCU80.GP10	O	CCU80.IN0M	事件 0 的选择信号
CCU80.GP11	O	未连接	事件 1 的选择信号
CCU80.GP12	O	未连接	事件 2 的选择信号
CCU80.ST1	O	CCU80.IN0N; CCU80.IN2O; CCU80.IN3O; VADC0.BGREQGTN; VADC0.G0REQGTN; VADC0.G1REQGTN;	状态位多路复用器的输出，可以是 CCST1 或 CCST2。

表 20-16 CCU80 - CC81 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
CCU80.ST1A	O	未连接	通道 1 状态位：CCST1
CCU80.ST1B	O	未连接	通道 1 状态位：CCST2
CCU80.PS1	O	POSIF0.MSYNCA	多通道序列同步触发信号：当向上计数 (边沿对齐) 时为 PM，当向下计数 (中心对齐) 时为 OM。

表 20-17 CCU80 - CC82 引脚连接

输入 / 输出	I/O	连接到	描述
CCU80.IN2A	I	P0.12	通用功能
CCU80.IN2B	I	P0.10;	通用功能
CCU80.IN2C	I	CCU40.GP22	通用功能
CCU80.IN2D	I	POSIF0.OUT2	通用功能
CCU80.IN2E	I	POSIF0.OUT5	通用功能
CCU80.IN2F	I	ERU0.PDOUT2	通用功能
CCU80.IN2G	I	ERU0.IOUT2	通用功能
CCU80.IN2H	I	SCU.GSC80	通用功能
CCU80.IN2I	I	BCCU0.OUT4	通用功能
CCU80.IN2J	I	BCCU0.OUT5	通用功能
CCU80.IN2K	I	CCU40.SR3	通用功能
CCU80.IN2L	I	ERU0.PDOUT0	通用功能
CCU80.IN2M	I	CCU80.GP30	通用功能
CCU80.IN2N	I	CCU80.ST0	通用功能
CCU80.IN2O	I	CCU80.ST1	通用功能
CCU80.IN2P	I	CCU80.ST3	通用功能
CCU80.MCI20	I	POSIF0.MOUT[8]	CCST1 的多通道序列输入
CCU80.MCI21	I	POSIF0.MOUT[9]	反相 CCST1 的多通道序列输入
CCU80.MCI22	I	POSIF0.MOUT[10]	CCST2 的多通道序列输入
CCU80.MCI23	I	POSIF0.MOUT[11]	反相 CCST2 的多通道序列输入

表 20-17 CCU80 - CC82 引脚连接

输入 / 输出	I/O	连接到	描述
CCU80.OUT20	O	P0.8; P0.12; P1.4; P2.0;	通道 1 的定时器片比较输出。可以是 CCST1 或反相的 CCST1。
CCU80.OUT21	O	P0.9; P0.13; P1.5; P2.1;	通道 1 的定时器片比较输出。可以是 CCST1 或反相的 CCST1。
CCU80.OUT22	O	P0.10;	通道 2 的定时器片比较输出。可以是 CCST2 或反相的 CCST2。
CCU80.OUT23	O	P0.11;	通道 2 的定时器片比较输出。可以是 CCST2 或反相的 CCST2。
CCU80.GP20	O	CCU80.IN1M	事件 0 的选定信号
CCU80.GP21	O	未连接	事件 1 的选定信号
CCU80.GP22	O	未连接	事件 2 的选定信号
CCU80.ST2	O	CCU80.IN0O; CCU80.IN1O; CCU80.IN3P;	状态位多路复用器的输出，可以是 CCST1 或 CCST2。
CCU80.ST2A	O	未连接	通道 1 状态位：CCST1
CCU80.ST2B	O	未连接	通道 2 状态位：CCST2
CCU80.PS2	O	未连接	多通道序列同步触发信号：当向上计数 (边沿对齐) 时为 PM，当向下计数 (中心对齐) 时为 OM。

表 20-18 CCU80 - CC83 引脚连接

输入 / 输出	I/O	连接到	描述
CCU80.IN3A	I	P0.12	通用功能
CCU80.IN3B	I	P0.13;	通用功能
CCU80.IN3C	I	CCU40.GP32	通用功能
CCU80.IN3D	I	POSIF0.OUT2	通用功能
CCU80.IN3E	I	POSIF0.OUT5	通用功能
CCU80.IN3F	I	ERU0.PDOUT3	通用功能
CCU80.IN3G	I	ERU0.IOUT3	通用功能

表 20-18 CCU80 - CC83 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
CCU80.IN3H	I	SCU.GSC80	通用功能
CCU80.IN3I	I	BCCU0.OUT6	通用功能
CCU80.IN3J	I	BCCU0.OUT7	通用功能
CCU80.IN3K	I	CCU40.SR3	通用功能
CCU80.IN3L	I	ERU0.PDOUT0	通用功能
CCU80.IN3M	I	CCU80.GP00	通用功能
CCU80.IN3N	I	CCU80.ST0	通用功能
CCU80.IN3O	I	CCU80.ST1	通用功能
CCU80.IN3P	I	CCU80.ST2	通用功能
CCU80.MCI30	I	POSIF0.MOUT[12]	CCST1 的多通道序列输入
CCU80.MCI31	I	POSIF0.MOUT[13]	反相 CCST1 的多通道序列输入
CCU80.MCI32	I	POSIF0.MOUT[14]	CCST2 的多通道序列输入
CCU80.MCI33	I	POSIF0.MOUT[15]	反相 CCST2 的多通道序列输入
CCU80.OUT30	O	P0.15; P2.10;	通道 1 的定时器片比较输出。可以是 CCST1 或反相的 CCST1。
CCU80.OUT31	O	P0.14; P2.11;	通道 1 的定时器片比较输出。可以是 CCST1 或反相的 CCST1。
CCU80.OUT32	O	P0.13;	通道 2 的定时器片比较输出。可以是 CCST2 或反相的 CCST2。
CCU80.OUT33	O	P0.12;	通道 2 的定时器片比较输出。可以是 CCST2 或反相的 CCST2。
CCU80.GP30	O	CCU80.IN2M	事件 0 的选择信号
CCU80.GP31	O	未连接	事件 1 的选择信号
CCU80.GP32	O	未连接	事件 2 的选择信号
CCU80.ST3	O	CCU40.IN0H CCU80.IN0P; CCU80.IN1P; CCU80.IN2P; VADC0.BGREQGTF; VADC0.G0REQGTF; VADC0.G1REQGTF;	状态位多路复用器的输出，可以是 CCST1 或 CCST2。

表 20-18 CCU80 - CC83 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
CCU80.ST3A	O	VADC0.BGREQGTE; VADC0.G0REQGTE; VADC0.G1REQGTE;	通道 1 状态位：CCST1
CCU80.ST3B	O	未连接	通道 1 状态位：CCST2
CCU80.PS3	O	POSIF0.MSYNCB	多通道序列同步触发信号：当向上计数 (边沿对齐) 时为 PM，当向下计数 (中心对齐) 时为 OM。

21 位置接口单元 (POSIF)

POSIF 单元是一个灵活而强大的电机控制系统组件，专门针对使用正交解码器和霍尔传感器为反馈回路的电机控制系统而设计。该模块有多种配置方案，能满足非常广泛的电机控制应用需求。该模块能用于构建工业和汽车电机应用的简单和复杂控制反馈回路，面向高性能的运动和位置监测。

表 21-1 缩略语表

PWM	脉冲宽度调制
POSIFx	位置接口模块实例 x
CCU8x	捕获 / 比较单元 8 模块实例 x
CCU4x	捕获 / 比较单元 4 模块实例 x
CC8y	捕获 / 比较单元 8 定时器片实例 y
CC4y	捕获 / 比较单元 4 定时器片实例 y
SCU	系统控制单元
f_{posif}	POSIF 模块的时钟频率

注： 寄存器中的小写字母 “x” 或 “y” 指示一个索引。

21.1 概述

POSIF 模块有三个主要的子单元：正交解码器单元、霍尔传感器控制单元和多通道模式单元 。

正交解码单元用于使用旋转增量式编码器的位置控制 。

霍尔传感器控制单元用于无刷直流电动机的直接控制。

多通道模式单元不仅用在与霍尔传感器模式结合输出需要的电机控制序列，而且可以用于以独立方式对几个控制单元执行简单的多通道控制。

POSIF 模块和 CCU4 或者 CCU8 结合使用，能够对任何类型的应用进行非常灵活的资源配置和优化。

21.1.1 特性

POSIF 模块特性

POSIF 由三个可独立工作的专用控制单元组成。

正交解码器单元包含一个输出接口，与CCU4模块结合使用时能够进行位置和速度测量。霍尔传感器单元能够控制多通道模式，可以最多与 8 个输出控制源连接。多通道单元提供与霍尔传感器模式完整的内置交互并可方便地构建独立控制回路。

一般特性

- 正交解码器
  - 位置测量接口
  - 电机转数测量接口
  - 速度测量接口
  - 相位错误、电机旋转、方向改变和相位解码错误中断源
- 霍尔传感器模式
  - 针对无刷直流电机控制的简单内置模式
  - 多通道序列的映射寄存器
  - 与 PWM 信号和多通道序列更新完全同步
  - 正确的霍尔事件检测、错误的霍尔事件检测中断源
- 多通道模式
  - 易于与霍尔传感器模式结合使用
  - 独立多通道模式
  - 多通道模式的映射寄存器

其他特性

- 正交解码器模式可与独立多通道模式并行使用。
- 可在正交解码器模式用多种配置 （通过 CCU4）进行位置和速度测量。
- 对输入序列评估和霍尔传感器模式的新序列值进行可编程延时 （通过 CCU4)。

POSIF 的特性与应用

表 21-2 列出了 POSIF 单元主要特性与相应的最常见应用的关系。

表 21-2　应用概览

特性	应用
正交解码器模式	易于与旋转编码器配合使用： • 有或没有索引 / 顶端标记信号 • 脱齿或轴弯曲补偿 • 有独立的位置、速度和旋转控制输出，符合不同系统的需求 • 针对位置跟踪的扩展配置 --- 具有旋转测量及多位置触发功能 • 支持脉冲到脉冲 (tick-to-tick) 和脉冲到同步 (tick-to-sync) 捕获方法所导致的高动态速度改变
霍尔传感器模式	易于与使用霍尔传感器的电机控制系统配合： • 2 或 3 个霍尔传感器的拓扑结构 • 扩展的输入滤波功能，以避免因噪声信号引起不必要的模式切换 • 与捕获 / 比较单元的 PWM 信号同步 • 支持死区时间的主动蓄流 / 同步整流 (与 捕获 / 比较单元 8 连接) • 使用一个捕获/比较单元定时器片的简单速度测量功能
多通道模式	调制多个 PWM 信号： • 通过软件控制的并行调制产生 N 路 PWM 信号 —— 针对多电源转换器系统 • 生成专有的 PWM 调制 • 根据系统反馈并行同步关闭 N 路 PWM 信号

21.1.2　原理框图

每个 POSIF 模块都可以有 3 种不同的工作模式，即正交解码器、霍尔传感器和独立的多通道模式。为了完成所有这 3 个模式的控制 / 测量回路，POSIF 需要与一个 CUU4/CUU8 连接。

在使用正交解码器模式的情况下，需要一个 CCU4 单元；对于霍尔传感器模式，一个传感器需要一个 CCU4 定时器片和一个 CCU8 单元。独立多通道模式和 CCU4/CCU8 模块之间的连接没有任何使用限制。

每个 POSIF 模块包含 30 个输入和 25 个输出 (包括服务请求)，见**图 21-1** 。这些输入 / 输出将根据用户所选择的配置被映射到模块的可用功能中。用户可选择的配置为：正交解码、霍尔传感器或独立多通道。

该模块还有两个专用的服务请求线，见 **21.3 节**。

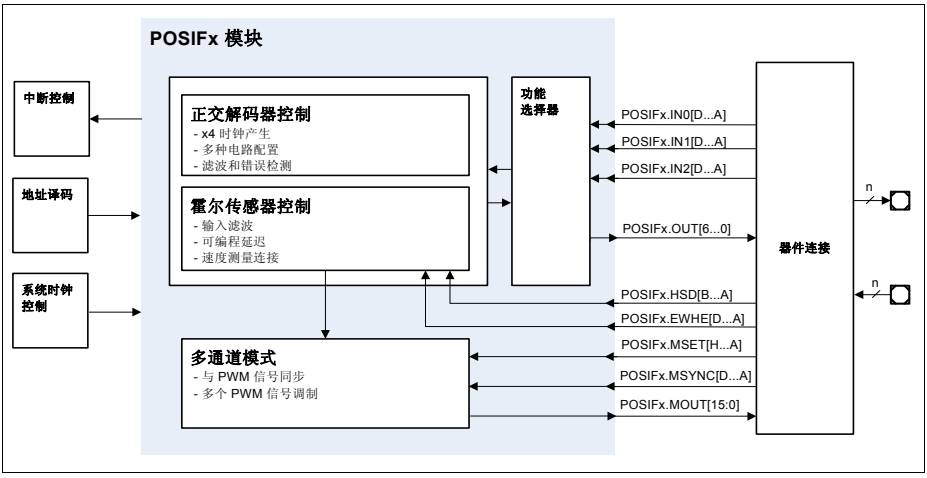


图 21-1　　POSIF 框图

21.2　功能描述

21.2.1　概述

POSIF 包含一个功能选择器单元，该单元由正交解码器和霍尔传感器单元并行使用。该功能块为每个控制单元选择应该使用哪个输入信号解码。功能选择器还对来自这两个模式（正交和霍尔传感器模式）的输出信号解码 。

POSIF 模块还包含一个仅用于霍尔传感器 / 多通道模式的输入子集。这些输入被连接到定时器结构 （CCU4/CCU8）并用于控制序列采样、多通道序列更新和同步等之间的延时。

多通道单元包括 16 个专用输出，其中包括多通道序列输出和可被连接到 CCU8/CCU4 的输出。

POSIF 控制单元包括一个可以由软件置位 / 清除的专用运行位。它还能产生一个可与定时器单元 （CCU4/CCU8）连接的同步起始信号。

对于正交解码器模式，可以定义哪一个信号是超前相位，还能定义每个信号的有效电平。

正交解码器模式也能够直接从一个外部源接收时钟和方向信号。

多通道模式为多通道序列提供一个映射寄存器，可对正在使用的参数进行更新。写访问总是寻址映射寄存器，而读访问总是返回多通道序列的当前值。

表 21-3　　POSIF 片引脚说明

引脚	I/O	霍尔传感器模式	正交解码器模式	多通道模式 (独立式)
POSIFx.IN0[D...A]	I	霍尔输入 1	编码器相位 A 或时钟	未使用
POSIFx.IN1[D...A]	I	霍尔输入 2	编码器相位 B 或方向	未使用
POSIFx.IN2[D...A]	I	霍尔输入 3	索引 / 零标记	未使用
POSIFx.HSD[B...A]	I	霍尔序列采样延时	未使用	未使用
POSIFx.EWHE[D...A]	I	错误霍尔事件仿真	未使用	错误霍尔事件仿真
POSIFx.MSET[H...A]	I	多通道下一模式更新置位	未使用	多通道下一模式更新置位
POSIFx.MSYNC[D...A]	I	多通道模式更新同步	未使用	多通道模式更新同步
POSIFx.OUT0	O	霍尔输入边沿检测触发	正交时钟	未使用
POSIFx.OUT1	O	霍尔正确事件	方向	未使用
POSIFx.OUT2	O	空闲 / 错误霍尔事件	周期时钟	未使用
POSIFx.OUT3	O	停止	清除 / 捕获	未使用
POSIFx.OUT4	O	多通道模式更新	索引	未使用
POSIFx.OUT5	O	同步启动	同步启动	未使用
POSIFx.OUT6	O	多序列同步触发	未使用	多序列同步触发
POSIFx.MOUT[15:0]	O	多通道序列	未使用	多通道序列
POSIFx.SR0	O	服务请求线 0	服务请求线 0	服务请求线 0
POSIFx.SR1	O	服务请求线 1	服务请求线 1	服务请求线 1

注：　内核输出的服务请求信号被额外扩展一个内核时钟周期。

21.2.2　功能选择器

功能选择器将输入功能信号映射到所选操作模式，即正交或霍尔传感器模式，见图 21-2。输出也通过该单元解码。

对每个功能输入 POSI0、POSI1 和 POSI2，用户可以通过域 **PCONF**.INSEL0、**PCONF**.INSEL1 和 **PCONF**.INSEL2 从四个输入引脚中选择其一。也可以对三个输入执行低通滤波，域 **PCONF**.LPC 控制低通滤波器的截止频率。

霍尔传感器模式是默认功能，即 **PCONF**.FSEL = 00_B。在该模式，功能选择器将 POSI0 映射到霍尔输入 1，将 POSI1 映射到霍尔输入 2，将 POSI2 映射到霍尔输入 3。

当选择正交解码器模式时，**PCONF**.FSEL = 01_B，POSI0 被映射到相位 A 或时钟，POSI1 被映射到相位 B 或方向，POSI2 被映射到来自旋转电机编码器的索引信号。注意，也可以通过设置 **PCONF**.FSEL = 11_B 来选择正交解码器和独立多通道模式。

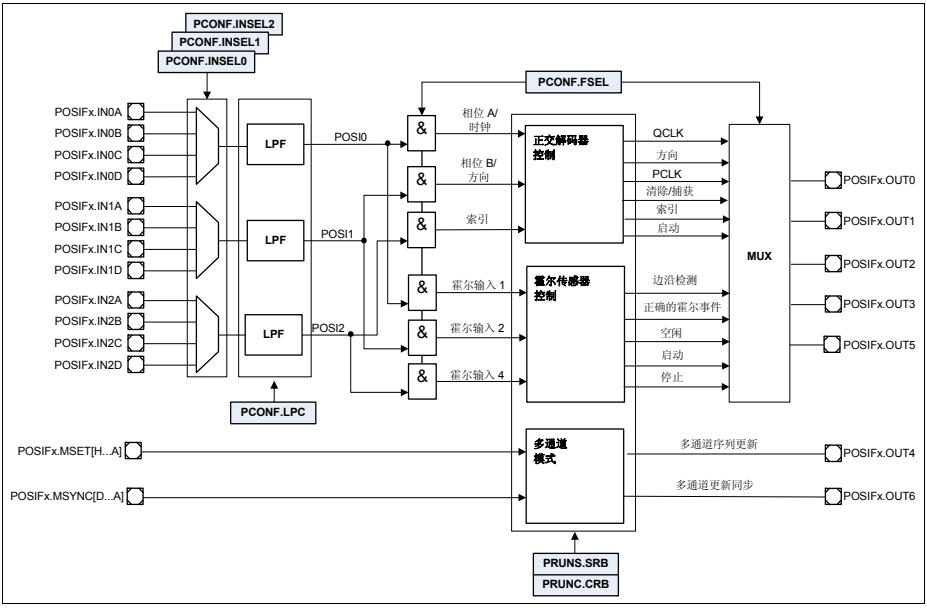


图 21-2　　功能选择器框图

21.2.3　霍尔传感器控制

霍尔传感器模式分成 3 个主要回路：霍尔输入中的任何更新检测、检测和采样霍尔输入之间的延时 （与预期霍尔序列比较）、多通道序列更新。

霍尔输入直接连接到边沿检测电路，这 3 个输入中任一输入的任何改变都会导致在 POSIFx.OUT0 引脚产生一个信号，见**图 21-3**。该引脚可被连接到一个 CCU4 定时器片，用于控制霍尔传感器模式下边沿检测和下一步以及霍尔输入采样之间的延时。

用于触发霍尔输入采样的信号由**PCONF**.DSEL域选择，该触发信号可以是上升或下降边沿有效 （**PCONF**.SPES）。

检测到采样触发信号时，即对霍尔输入采样，并将采样结果与当前序列 **HALP**.HCP 和预期霍尔序列 **HALP**.HEP 进行比较（评估输入序列是否与 HEP 或 HCP 值匹配）。

边沿检测电路为采样霍尔输入产生一个使能信号，每当捕获一个新值时，PIFHSP 和采样逻辑产生一个脉冲 PIFHRDY，该脉冲供序列比较逻辑内部使用。

当采样值与预期霍尔序列匹配时，在 POSIFx.OUT1 引脚产生一个脉冲，用于指示发生了正确霍尔事件。与此同时，下一个已经编程到映像寄存器的值被加载。**HALP**.HCP[LSB] 被连接到霍尔输入 1，**HALP**.HCP[MSB] 被连接到与霍尔输入 3（这同样适用于 **HALP**.HEP 寄存器）。

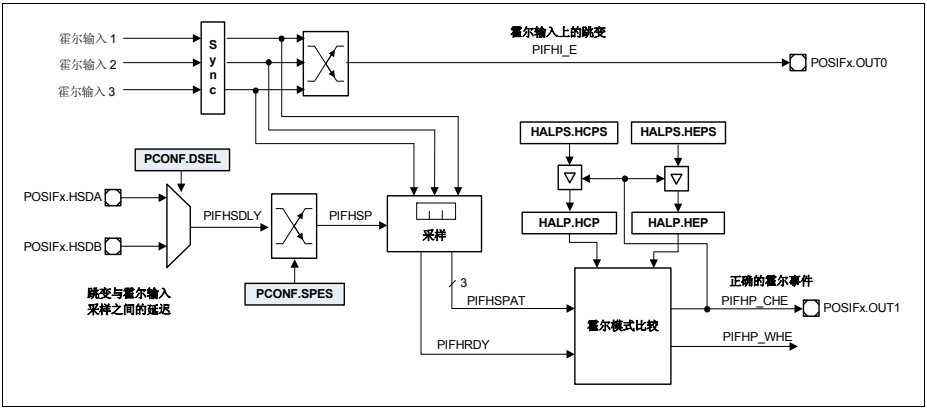


图 21-3　　霍尔传感器控制框图

当采样值与当前霍尔序列匹配时，可认为发生了线路干扰，但不会执行任何动作。当采样值与任何值（当前和预期序列）都不匹配时，可认为发生了严重错误，这时会产生错误霍尔事件信号。每当采样的序列导致一个错误霍尔事件或与当前序列相匹配时，会在 POSIFx.OUT3 引脚产生一个停止信号，见**图 21-4**。

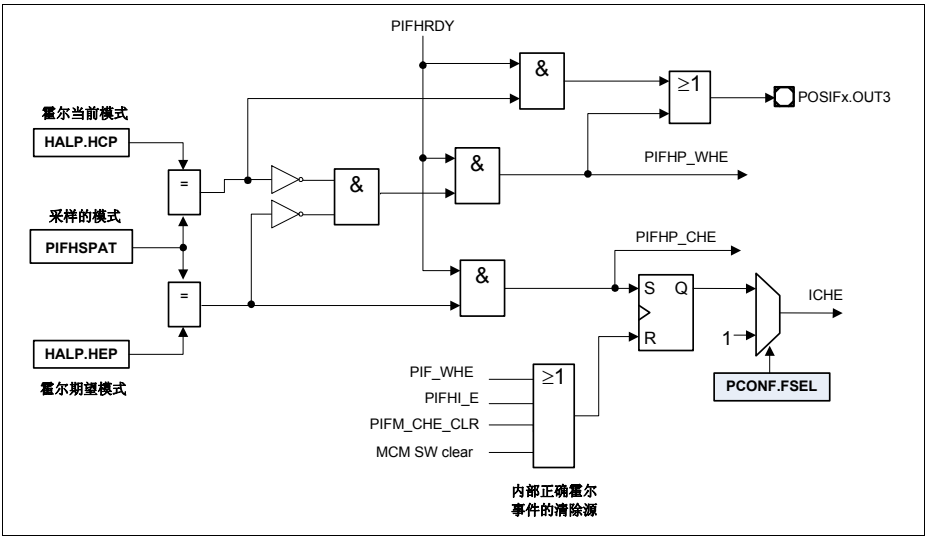


图 21-4　　霍尔传感器比较逻辑

一个错误的霍尔事件能够产生一个空闲信号（IDLE），该信号连接到 POSIFx.OUT2，可用于清除霍尔传感器控制单元的运行位。

空闲信号还可连接到 PWM 单元，用于执行强制停止操作，见图 21-5。错误的霍尔事件 / 空闲功能也可以通过一个引脚控制。

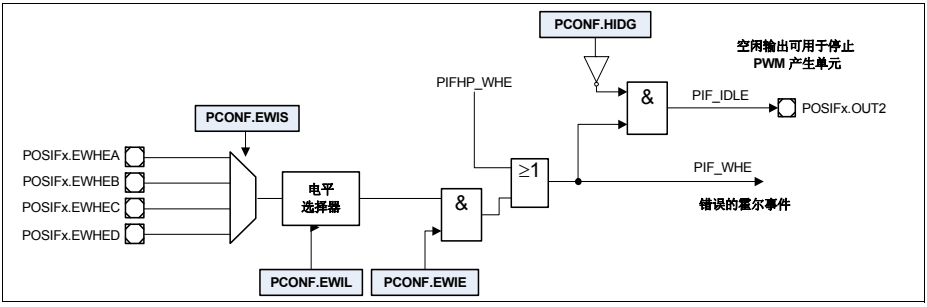


图 21-5　　错误霍尔事件 / 空闲逻辑

检测到一个正确霍尔事件之后，在这个检测和多通道序列更新之间可以产生一个延时。图 21-6 对多通道模式的控制逻辑进行了说明。

多通道序列的更新延时可由一个 CCU4 定时器片直接控制。指示延时结束的触发信号可被映射到一个输入信号 POSIFx.MSET[H...A]（**PCONF**.MSETS 选择用于此目的的输入信号）。还可以通过 **PCONF**.MSES 选择触发的有效沿。

PCONF.MCUE 域选择使能多通道序列更新的信号源。当该域被置为 1_B 时，多通道序列只能在软件向 **MCMS**.MNPS 域写入 1_B 之后才被更新。

经过更新延时之后，多通道序列仍然需要与 PWM 信号同步。为此，用户要通过 **PCONF**.MSYNS 域从 POSIFx.MSYNC[D...A] 输入选择一个信号。当检测到该信号的下降沿时，新的多通道序列被施加到 POSIFx.MOUT[15:0] 输出，**MCM**.MCMP[15] 连接到 POSIFx.MOUT[15]，**MCM**.MCMP[0] 连接到 POSIFx.MOUT[0]。

POSIFxOUT6 引脚被连接到 **MCMF**.MSS 寄存器域。该寄存器域使能多通道序列更新（在收到同步信号 PIFMSYNC 之后完成更新），并可与捕获比较模块联合执行多通道序列与捕获比较单元内部所用比较值的同步更新。

当错误霍尔事件导致霍尔传感器控制进入空闲状态时，多通道序列也被清零。

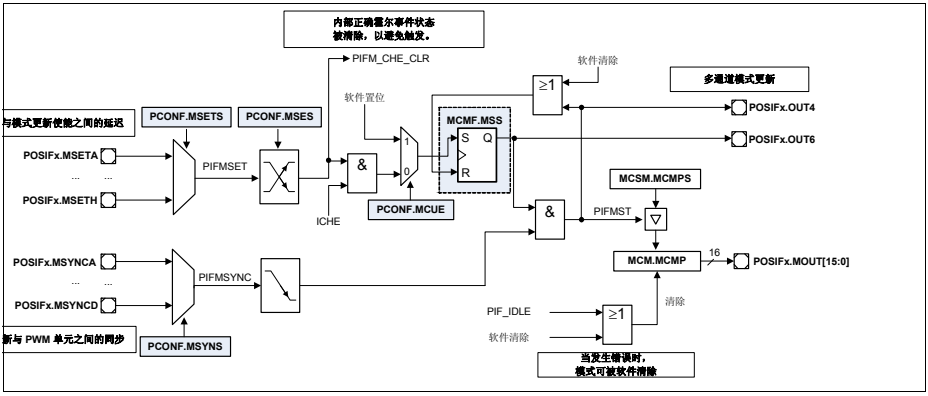


图 21-6　　多通道模式框图

图 21-7 展示了在霍尔传感器模式下所有先前描述的步骤。每当检测到霍尔输入跳变时，引脚POSIFx.OUT0变为有效。该信号用于启动在跳变检测和霍尔输入采样之间的延时。

在这种情况下，一个定时器单元的状态输出（**图 21-7** 中的 ST）用于控制时序 1（跳变和霍尔输入采样之间的延时）。霍尔输入在 ST 信号的上升沿被采样，如果这些输入与预期序列匹配，则信号 POSIFx.OUT1 变为有效。

被映射到同一定时器单元的一个服务请求线（SR 信号）用来控制一次正确霍尔事件与多通道序列更新之间的延时。当检测到周期匹配时，该服务请求变为有效。

另一个定时器单元用于测量每次正确霍尔事件之间的时间。该定时器单元由**图 21-7** 上的时序 2 表示（CR 表示定时器单元的捕获寄存器）。

当时序 1（SR 信号）控制的延时结束以后，多通道序列更新需要与 PWM 信号同步。这可以通过使用产生 PWM 信号的捕获 / 比较单元（例如使用 CCU8）的一个序列同步输出来完成。

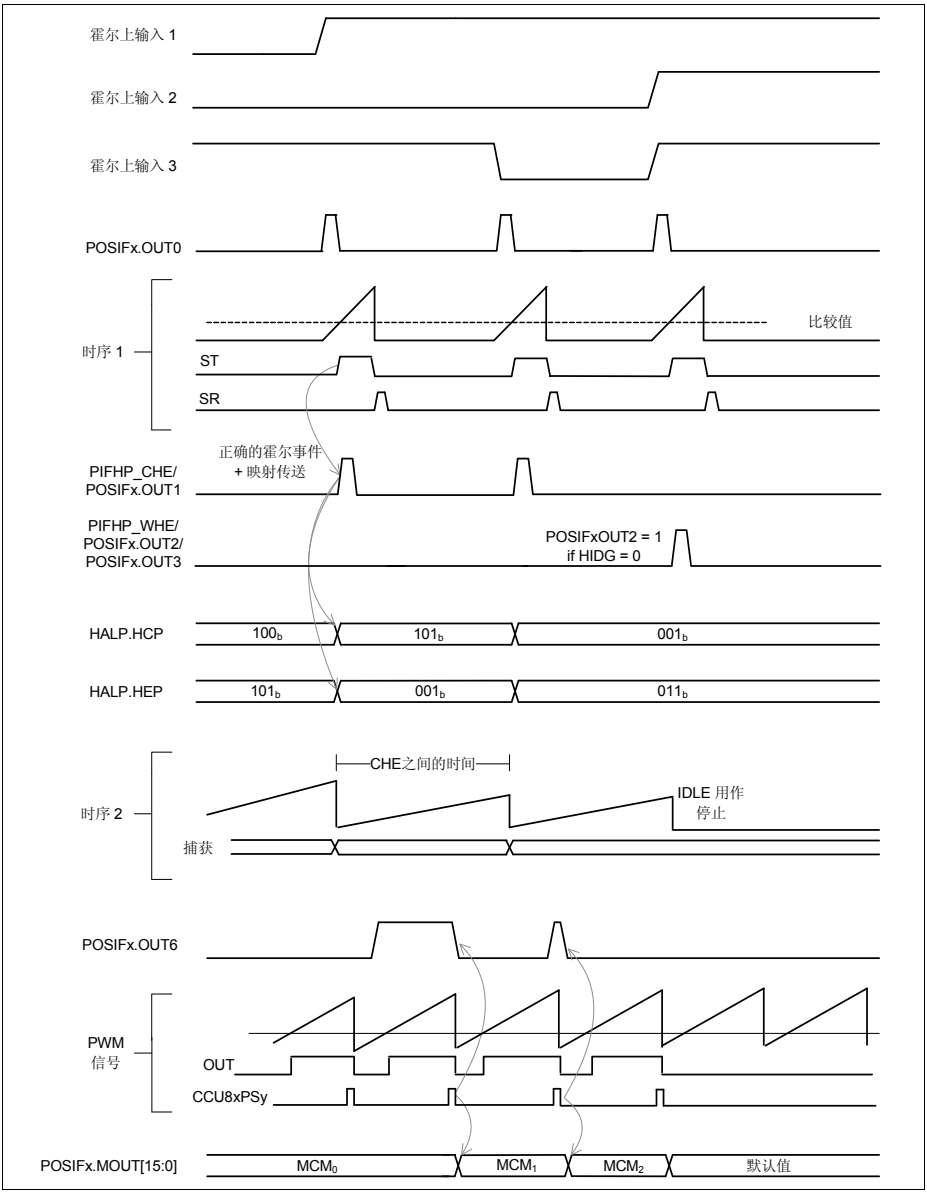


图 21-7　　霍尔传感器时序图

21.2.4 正交解码器控制

正交解码器模式通过设置 **PCONF**.FSEL = 01_B 或 **PCONF**.FSEL = 11_B 来选择 (在这种情况下多通道模式也被使能）。

在正交解码器模式下，有两种不同的子集可用。

- 标准正交模式
- 方向计数模式

当外部旋转编码器提供两个相位信号，并且轴每旋转一周都会产生索引 / 标记信号时，使用标准模式。当外部编码器只提供一个时钟和一个方向信号时方向时，使用方向计数模式，见**图 21-8**。

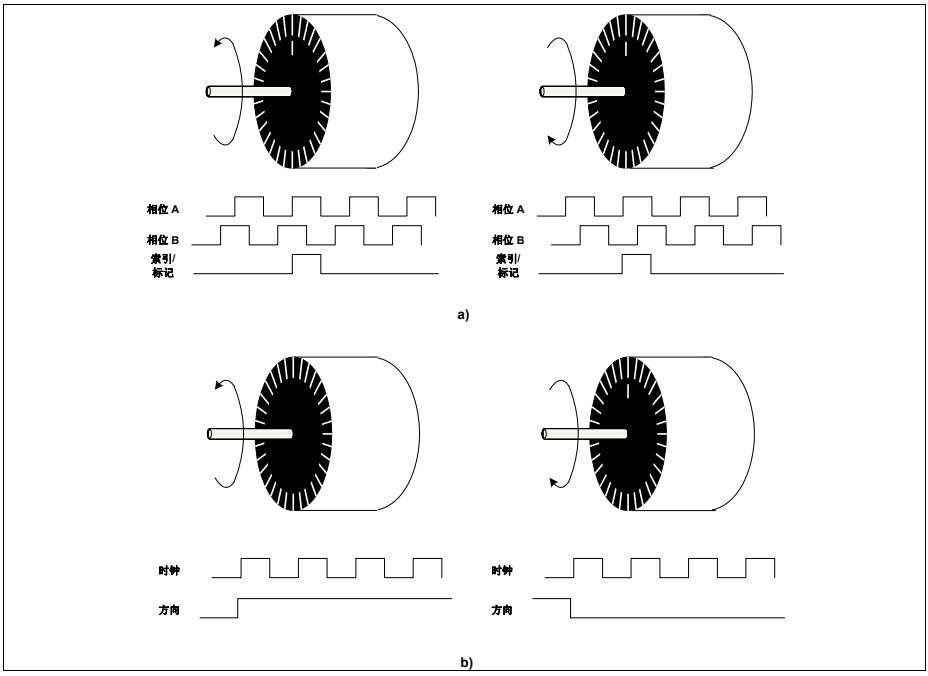


图 21-8 旋转编码器类型 - a) 标准双相位加索引信号 ; b) 时钟加方向

标准正交模式

正交解码器单元提供了一个非常灵活的相位 A/ 相位 B 配置级。一般来说，对于顺时针电机轴旋转，相位 A 应该领先于相位 B，但用户可以配置领先相位，也可以为每个信号配置特定的有效状态，见 **图 21-9**。

正交解码器控制单元由两个主要模块构成：解码正交时钟和电机轴方向的模块，处理索引（电机旋转）控制的模块。

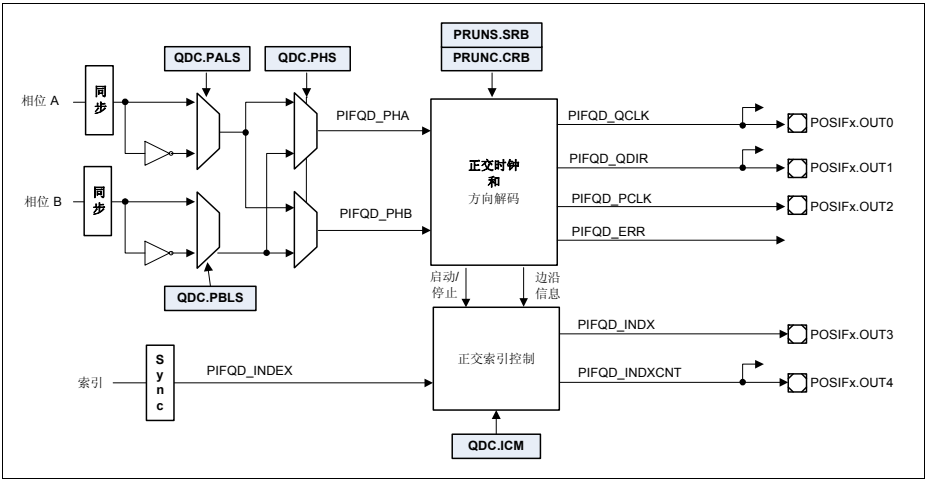


图 21-9　　正交解码器控制概览

正交时钟被连接到 POSIFx.OUT0 引脚，用于位置测量。该时钟是从相位信号的每个边沿解码得到，因此每个相位周期有 4 个时钟脉冲。

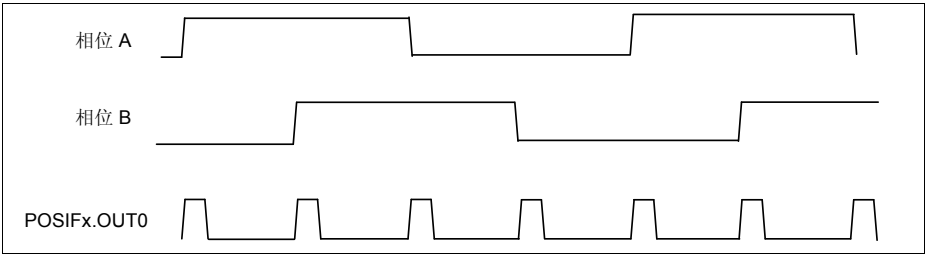


图 21-10　　正交时钟产生

该单元还为速度测量操作产生周期时钟。

电机旋转方向连接到 POSIFx.OUT1 引脚，当电机顺时针旋转时为高电平有效，当逆时针旋转时为低电平有效。

索引控制逻辑记忆索引信号有效后哪个是第一个边沿，以便使用相同的正交跃变进行索引事件操作。该逻辑也能记录第一个索引的方向，所以它能够控制旋转增量信号何时有效。

当检测到错误的相位对时，可以产生一个与标志相连接的错误信号（如果使能，可以产生中断）。

正交解码器控制使用当前相位对和前一相位对的信息对方向和时钟进行解码。两个相位信号都分别通过一个边沿检测逻辑，检测逻辑的输出用于指示是否产生了有效的跳变沿，见图 21-11 。可以通过向 **PRUNC**.CSM 域写入 1_B 来复位解码器状态机（但不对标志和静态配置复位）。

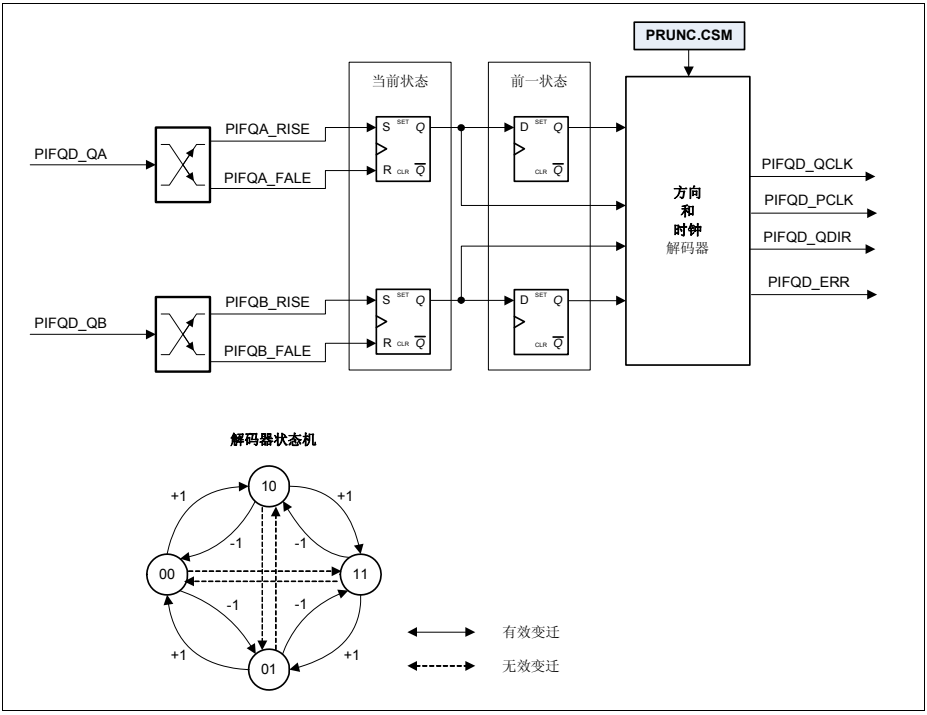


图 21-11　　正交解码器状态

方向计数模式

有些位置编码器没有相位信号作为输出，而是提供了包含时钟和方向信息的两个信号。这就是方向计数模式。

当使用这种类型的位置编码器时，用户应该将 **PCONF**.QDCM 位域置为 1_B (使能方向计数模式)。在这种情况下，从 POSIFx.IN0[D...A] 选择的信号作为时钟，从 POSIFx.IN1[D..A] 选择的信号包含方向信息。

输入信号与 POSIF 模块的时钟同步，并且发送到各自的输出。在这种情况下，与索引 / 零标记连接的输入和输出未使用。POSIFx.OUT2 输出也无效。

21.2.4.1 正交时钟和方向解码

正交解码器单元输出两路时钟。一路时钟用于位置控制，因此在相位信号的每个边沿跳变都会产生。第二路时钟用于速度测量，该时钟不受很低转速情况下在相位信号上可能出现的毛刺的影响。这些毛刺不是一般的线路噪声，而是由于引擎转速低引起的。

方向信号的解码遵循图 21-11 中的规则完成。图 21-12 示出了相位 A 和相位 B 的全部有效跳变。

图 21-13 展现了在相位信号上出现一些毛刺的情况下这两路时钟信号之间的差别。

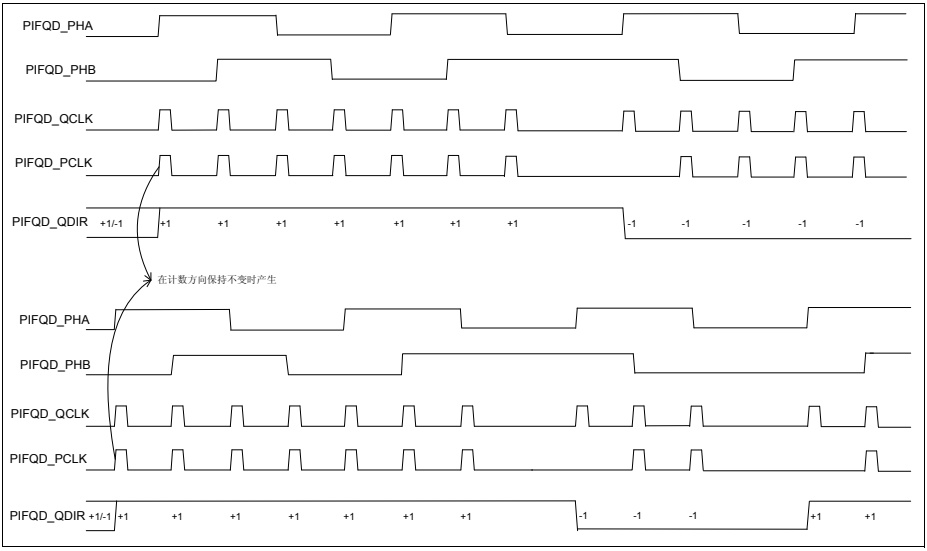


图 21-12 正交时钟和方向信号时序

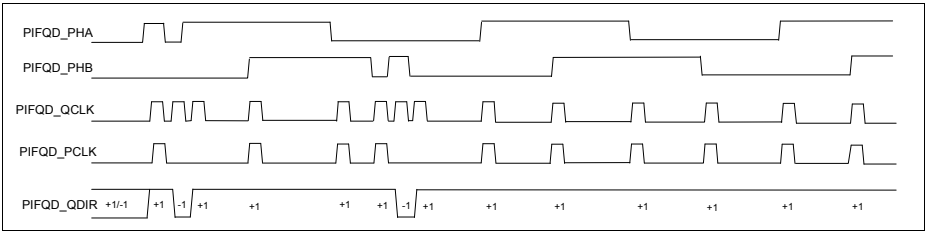


图 21-13 带抖动的正交时钟

21.2.4.2 索引控制

索引控制逻辑有两路不同的输出。一路是 POSIFx.OUT4，在每次检测到索引信号（并且电机轴旋转方向相同）产生一个脉冲，因此它能用于旋转计数。有了该信号，软件不仅可以检测轴位置，还能监测已经发生的总转数。

另一路输出为 POSIFx.OUT3，其行为可以通过对 **QDC**.ICM 域编程来控制。根据 **QDC**.ICM 的设置值不同，该信号的产生有以下几种情况：

- 在索引信号每次出现时产生
- 仅在索引信号第一次出现时产生
- 被禁止 - 该输出永远不会产生脉冲信号。

该路输出的这种特性对需要在每次发生索引事件或仅在第一次发生索引事件发生时复位计数器的应用来说非常有用。

索引控制逻辑会记忆在一个索引之后出现的第一个相位边沿，因此产生的信号总是具有相同的参考点。如果该索引之后的第一个相位边沿是相位 B 信号的上升沿，在发生下一索引事件时，若方向保持不变，索引信号在相位 B 信号的上升沿产生；如果方向发生改变，则索引信号在相位 B 信号的下降沿产生。

图 21-14 示出了索引信号产生的时序图。在这种情况下，**QDC**.ICM = 10_B，这意味着 POSIFx.OUT3 索引信号将会在每次发生输入索引发生时产生。

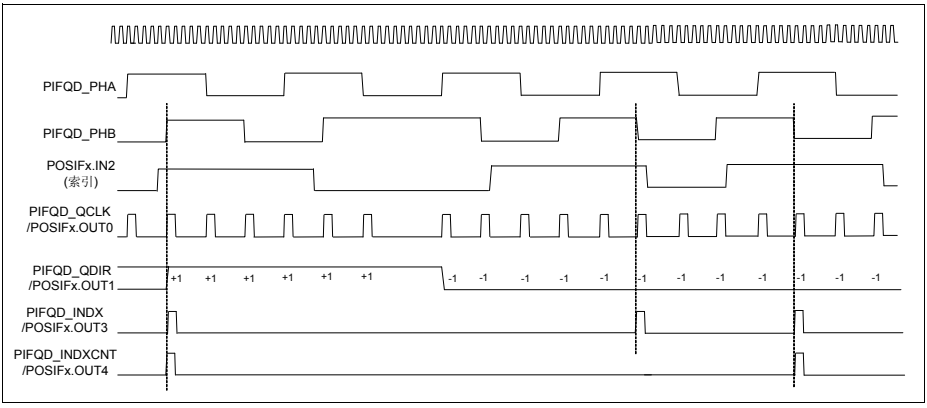


图 21-14 索引信号时序

21.2.5 独立多通道模式

多通道模式（多通道模式逻辑见 **图 21-6**）能够在不使用霍尔传感器控制的情况下使用，通过设置 **PCONF**.FSEL = 10_B 即可实现；如果还需要正交解码器模式，则可设置 **PCONF**.FSEL = 11_B。

在独立多通道模式，更新多通道序列的机制与在 **21.2.3 节**中描述的相同。

模式更新的触发可以来自软件，通过向 **MCMS**.MNPS 写入 1_B 实现；也可以来自像霍尔传感器模式中的外部信号。应通过在 **PCONF**.MSETS 域中选择合适的值，将这个外部信号映射到 PIMSET 功能。

仍然需要完成新多通道序列的更新与控制信号之间的同步。用户需要将 POSIFx.MSYNC[D...A] 中的一个输入信号映射到该功能。

通过向 **MCMC**.MPC 域写入 1，软件可以清除当前的多通道序列。

21.2.6 同步启动

POSIF 模块有一个同步启动输出 POSIFx.OUT5, 该输出可以与捕获比较单元一起使用，使两个模块完全同步启动。该同步启动信号连接到 POSIF 模块的运行位，这就意味着每当运行位被置 1 时， POSIFx.OUT5 引脚将产生一个脉冲。

通过采用同步启动输出，软件在启动 POSIF 和 CCU4/CCU8 时不需执行两次独立的访问，因此保证了两个模块在同一时间启动其操作。

21.2.7 使用 POSIF

POSIF 模块需要连接到一个 CCU4/CCU8 模块，这样才能在每种可能的模式执行全部功能 （因为 POSIF 没有内置的计数器 / 定时器）。

要使 POSIF 在正交解码器模式下工作 , 需要一个 CCU4 模块。霍尔传感器模式需要一个 CCU8 （控制无刷直流电机至少需要 3 个定时器片），并且还需要 （至少）两个 CCU4 或 CCU8 定时器片。独立多通道模式的连接配置与具体用例紧密相关，因此所用的 CCU4/CCU8 定时器片数由用户自由选择。

21.2.7.1 霍尔传感器模式的使用

当使用 POSIF 的霍尔传感器模式时，多通道模式也工作。因此，在用于执行多通道调制时 CCU8 模块需要被配置为多通道模式。

标准霍尔传感器模式使用

在**图 21-15** 中 , 霍尔传感器模式与两个 CCU4 定时器片和一个 CCU8 模块配合使用。第一个 CCU4 定时器片，即定时器片 0，用于控制霍尔输入的边沿检测与实际采样之间的延时，还用于控制一个正确霍尔事件与多通道序列更新使能之间的延时。

CCU4x.ST0 的上升边沿用作第一个延时的结束触发信号，服务请求线用于在发生正确霍尔事件之后触发新序列更新。服务请求被配置为在特定定时器片每次发生周期匹配时有效。

定时器片 0 被配置为单次模式，这样可在 POSIF 每产生一次请求时触发延时。

第二个 CCU4 单元定时器片，即定时器片 1，在捕获模式使用，以捕获正确霍尔事件之间的时间 （用这样的方式存储两个正确霍尔事件之间的电机速度） 。 POSIF 的 POSIFx.OUT1 用作该定时器片的捕获触发信号，而 POSIFx.OUT3 用作停止信号。捕获和停止触发被配置在特定的定时器片，上升沿有效。

CCU8 模块产生控制电机的 PWM 信号，因此多通道序列的输出 POSIFx.MOUT[15:0] 被连接到该单元。

要关闭多通道回路，需要将 CCU8 的一个输出连接到 POSIF 模块（一个 CCU8x.PSy 信号），使多通道序列更新与 PWM 周期同步。

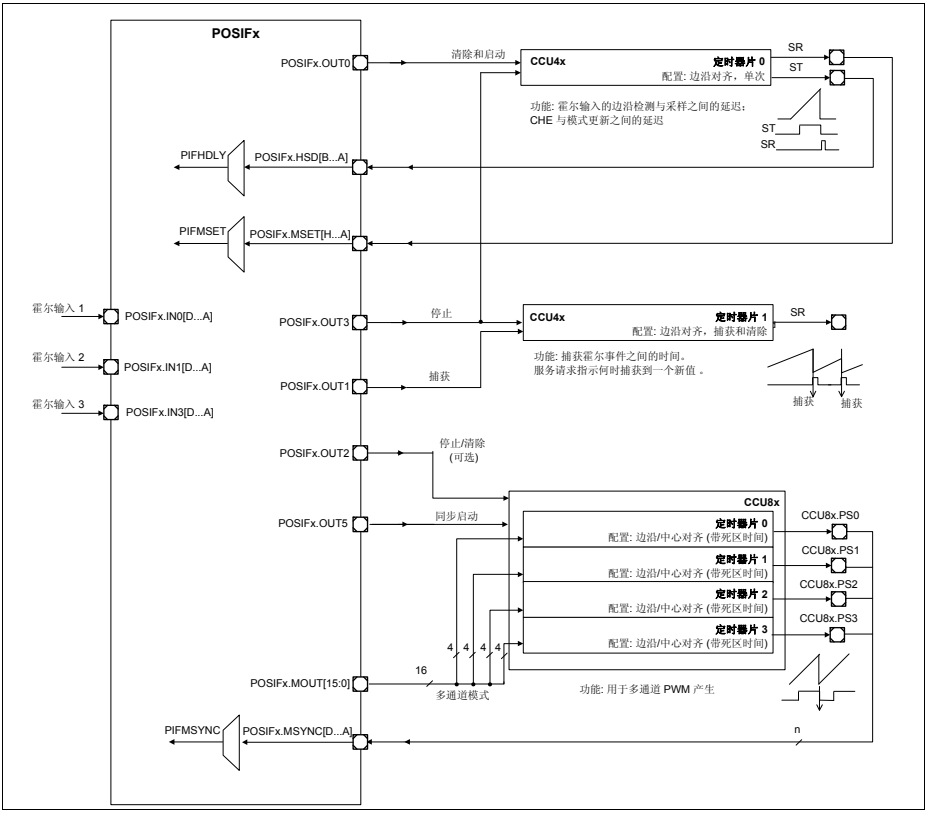


图 21-15　　霍尔传感器模式应用框图 1

霍尔传感器模式使用 - 灵活的时间控制

图 21-16 给出了另一种配置。在这种情况下，霍尔传感器模式使用 3 个 CCU4 定时器片，而不是 2 个 CCU4 定时器片。这种配置在延时配置方面提供了更大的灵活性。该配置使用两个定时器片分别控制霍尔输入跳变与采样之间的延时和正确霍尔事件与多通道序列更新之间的延时。同时该配置还消除了使用服务请求来控制序列更新延时的需要。

定时器片 0 用于控制在霍尔输入跳变和实际采样之间的延时。定时器片 2 用于控制一个正确霍尔事件和多通道序列更新之间的延时。

定时器片 1 用作正确霍尔事件之间的时间戳保持器。（这与配置 1 中定时器片 1 的功能相同）。

PWM 信号与多通道序列更新之间的同步也是通过一个 CCU8x.PSy 输出来实现。

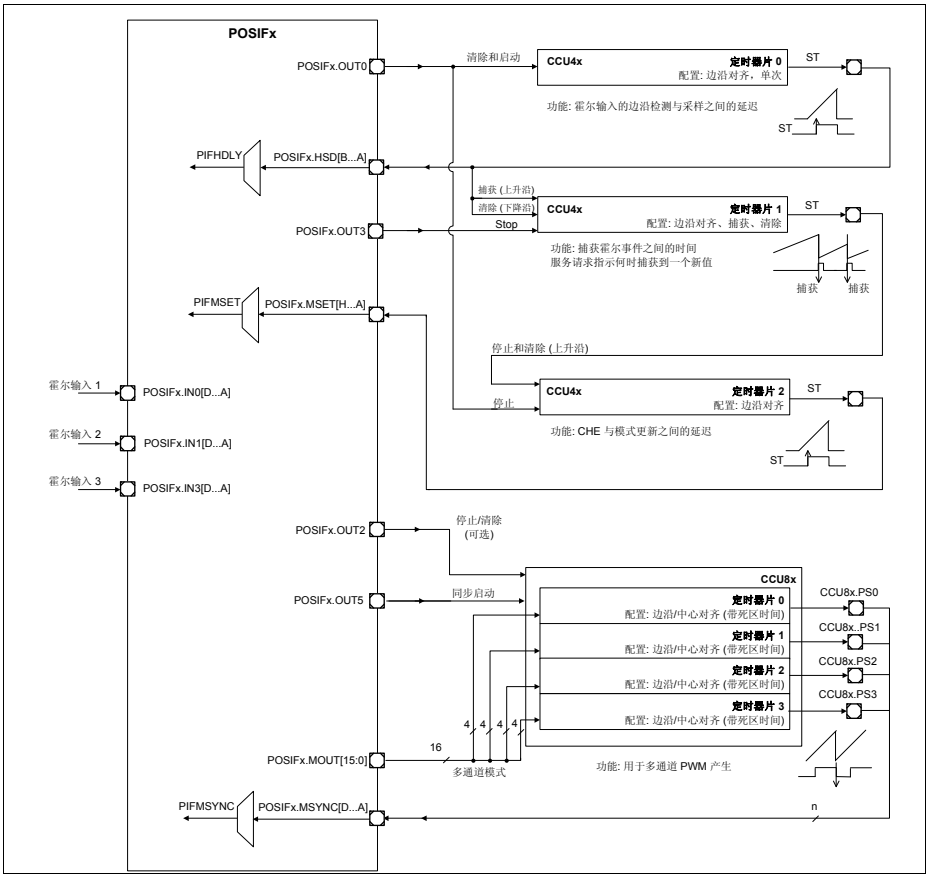


图 21-16　　霍尔传感器模式使用 – 配置 2

21.2.7.2　正交解码器模式使用

正交解码器模式与 CCU4 单元相连时，其使用方式非常灵活。连接配置取决于用户想要并行执行的功能数量：位置控制、旋转控制和速度测量。

正交解码器使用 - 脉冲和转数比较加上 N 个脉冲的 T 法测速

在图 21-17 中， POSIF 与一个 CCU4 连接，使用了所有的模块定时器片。在这种正交解码模式下，用户将一个定时器片用于位置控制 （比较），一个定时器片用于旋转计数，其余两个定时器片合起来用于速度测量。

定时器片 0 连接到 POSIFx.OUT0 和 POSIFx.OUT1 输出，这意味着必须将 POSIFx.OUT0 配置为计数功能，把 POSIFx.OUT1 配置为向上 / 向下计数控制功能。该定时器片用于跟踪系统的当前位置，比较通道被配置为在位置到达某一特定值时触发所需要的动作。

定时器片 1 连接到 POSIFx.OUT4 引脚，意味着它用于电机旋转计数。比较通道可被配置为每当电机旋转 N 次时触发一次中断。

定时器片 2 和定时器片 3 用于执行速度测量。定时器片 2 通过 POSIFx.OUT2 接收正交解码器周期时钟，也就是说被映射为计数功能。该定时器片的比较通道用于在定时器片 3 中触发捕获事件。最后一个定时器片使用模块时钟，因此每当发生捕获事件时，实际系统脉冲都被捕获到。 通过这种方式，用户能够知道 N 个相位周期之间使用了多少时间，进而可以计算出对应的速度分布。

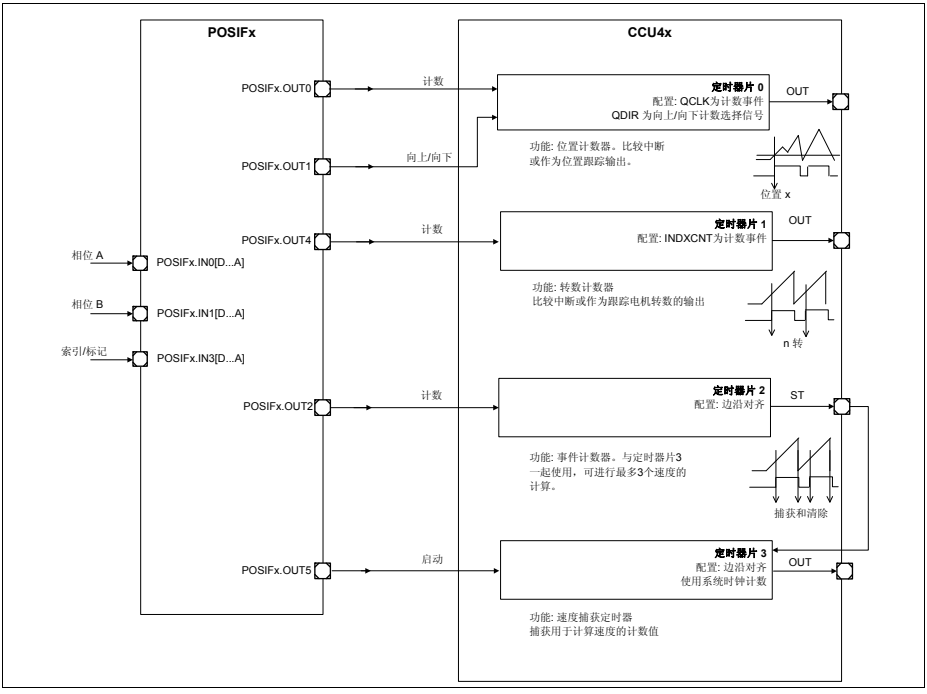


图 21-17　　正交解码器模式使用 – 配置 1

正交解码器使用 - 扩展脉冲比较加 N 个脉冲的 T 法测速

图 21-18 所示的配置使用两个比较通道实现位置控制。该配置对于在电机旋转一周期间要进行多个操作的应用特别有用。

定时器片 0 和定时器片 1 使用 POSIFx.OUT0 作为计数功能， POSIFx.OUT1 作为向上 / 向下计数控制。每个定时器片中的比较通道被编程为不同的比较值。

片 2 和片 3 的使用方法与配置 1 相同，见 图 21-18。

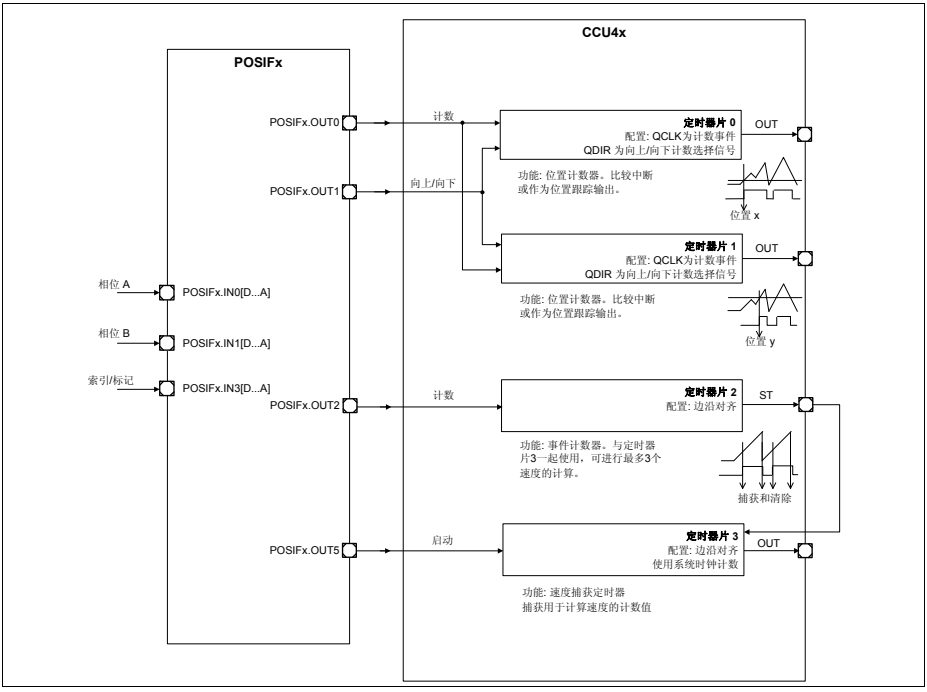


图 21-18　　正交解码器模式使用 – 配置 2

正交解码器使用 - 带索引清零的脉冲和转数比较加上 N 个 脉冲的 T 法测速

在一些应用中，常使用索引标记作为位置和速度控制的零位信号。该清除操作与 POSIFx.OUT3 引脚连接，可被编程为只在第一个索引标记生效或在所有的标记都生效。该引脚被连接到特定的 CCU4 定时器片作为清除功能使用。图 21-19 展示了将配置 1 修改为用索引标记信号作为清除信号的情况。

上述过程同样用在配置2，这样我们就有两个比较通道加上速度测量以及索引作为清除信号。

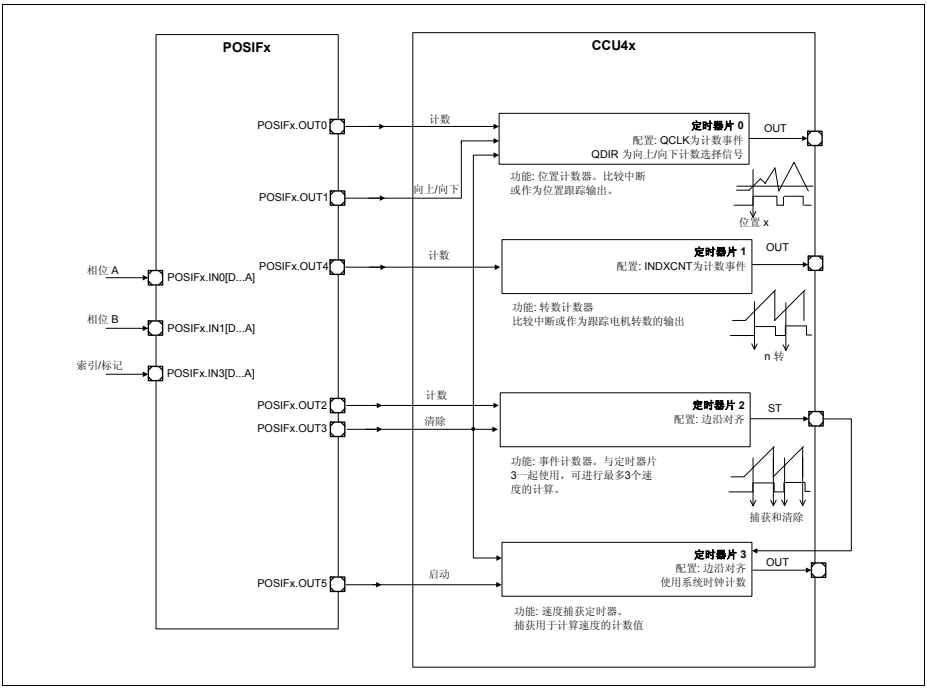


图 21-19　　正交解码器模式使用 – 配置 3

正交解码器使用 - 脉冲比较以及低转速下 M/T 法测速

当电机旋转速度很慢时，忽略掉不足一个脉冲周期的计算方法就不太适用了。

在一个快速旋转系统中，脉冲之间的时间通常被忽略，在进行速度计算时只考虑从上一次 ISR 以来发生的脉冲数。但是在一个低速旋转系统中，记录的脉冲数目可能并不足够（因为相关的错误），而且软件也需要考虑上一个脉冲与实际 ISR 触发之间的时间。

图 21-20 展示了一个低转速系统，速度计算的 ISR 被周期性触发。在这种情况下，因为每次 ISR 触发之间的脉冲数量少，所以软件不仅需要知道已经发生的脉冲数，而且还要知道上一个脉冲与本次 ISR 发生之间的时间。

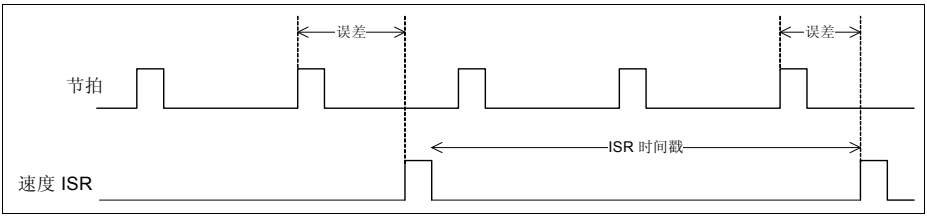


图 21-20　　低转速系统示例

可以用 POSIF 和一个 CCU4 模块构建一个不受这种低转速计算缺陷影响的控制回路。图 21-21 给出了资源使用的例子。

CCU4 模块的一个定时器片，即定时器片 0，用于监测电机轴的当前位置。

还需要三个额外的定时器片来构建低速计算回路：即定时器片 1、定时器片 2 和定时器片 3。

定时器片 1 对每次速度 ISR 之间发生的脉冲 （PCLK）计数。

定时器片 2 对每次脉冲（PCLK）发生之间的间隔计时。该定时器片在每个脉冲内都要清零和启动，因此它总是保持上一脉冲和当前脉冲之间的时间信息。

定时器片 3 控制计算速度的周期。每当速度 ISR 被触发时，定时器片 3 还为定时器片 2 触发一次捕获并为定时器片 1 触发一个捕获和清除。

通过这一机制，每当软件从定时器片 1 和定时器片 2 读取捕获值时，可以根据以下数据计算速度：

- 自上次 ISR 以来产生的脉冲数
- 加上上一脉冲与当前 ISR 之间的时间

这样一来，该控制回路能在低速系统中提供一种非常精确的电机速度计算方法。

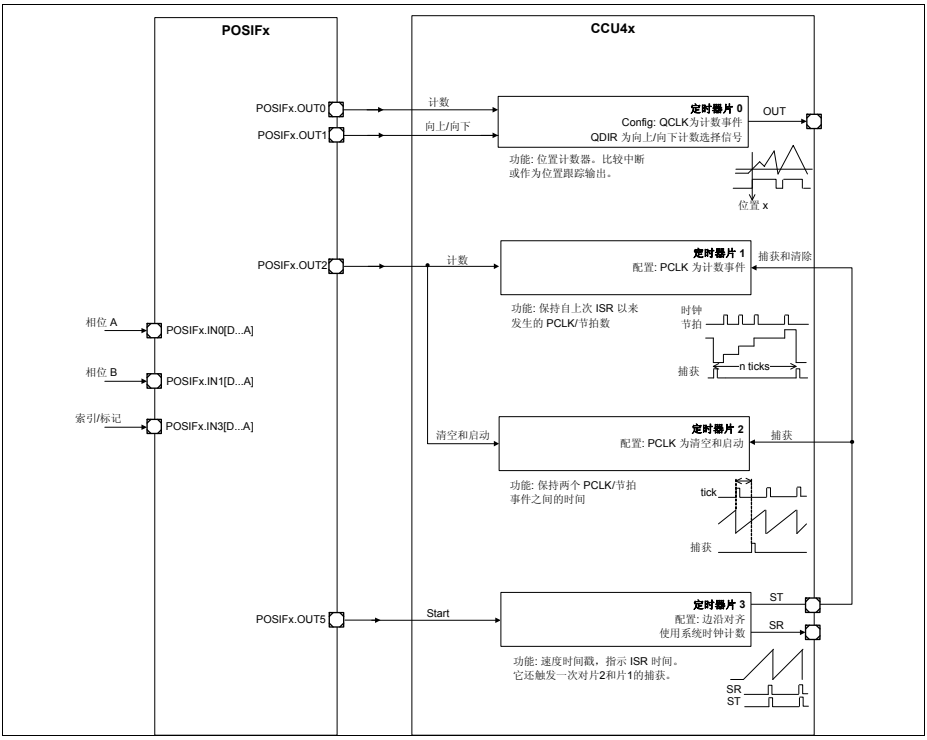


图 21-21　　正交解码器模式使用 – 配置 4

21.2.7.3　独立多通道模式

POSIF 模块的多通道模式能够在不与霍尔传感器模式连接的情况下使用。这对实现需要精确同步的一般控制回路非常有用。

独立多通道模式非常通用，它在很大程度上取决于用户确定的控制回路。多通道模式的一般配置是使用一个 CCU4/CCU8 模块完成信号的生成，一个连接到外部引脚的 CCU4 定时器片用作多通道序列更新的触发信号。

在 **图 21-22** 中，一个 CCU4 定时器片用于控制一个外部信号（例如外部传感器）更新与多通道序列更新之间的延时。注意，如果不需要延时，该外部信号可以连接到 POSIF 模块，也可以由软件控制，通过向 **MCMS**.MNPS 域写入 1 实现。

可以从产生 PWM 信号的 CCU8/CCU4 单元选择一个输出，作为所产生的信号与多通道序列更新间的同步触发信号（在 CCU4 的情况下为 CCU4x.PSy，在 CCU8 情况下为 CCU8x.PSy）。

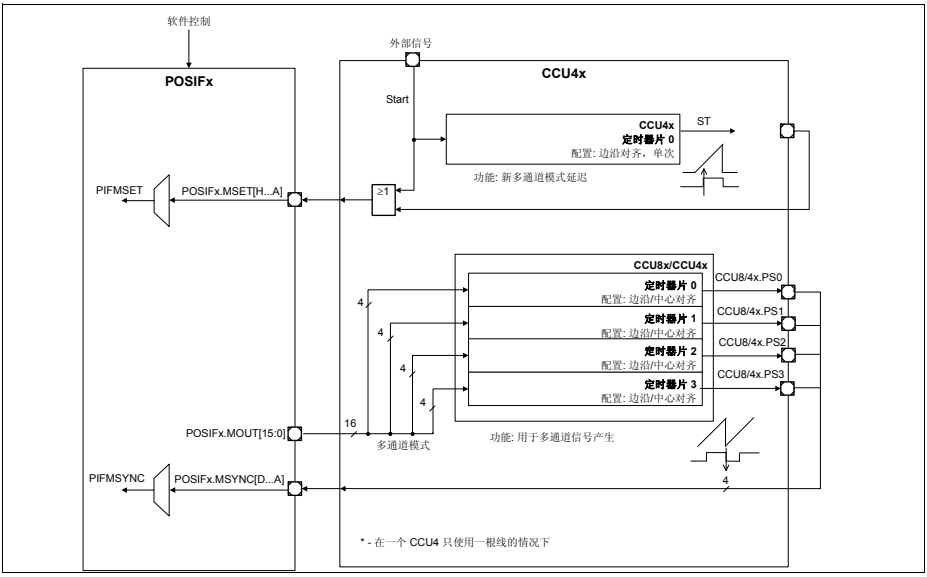


图 21-22　　独立多通道模式使用

21.3　服务请求产生

POSIF 有多个中断源，这些中断源与不同的操作模式连接：霍尔传感器模式、正交解码器模式和独立多通道模式。

霍尔传感器模式的中断标志在 **21.3.1 节**中描述，正交解码器模式的中断标志在 **21.3.2 节**中描述。

与多通道功能相关联的标志在所有模式中都可用。这是因为多通道模式能与正交解码模式并行工作，而且每当霍尔传感器模式被激活时也需要多通道模式。

每个中断源都能被路由到 POSIFx.SR0 或 POSIFx.SR1 输出，取决于 **PFLGE** 寄存器的编程值。

21.3.1　霍尔传感器模式标志

霍尔传感器控制包括四个标志，这些标志可被配置为产生中断请求脉冲，见**图 21-23**。

这四个中断源是：

- 霍尔输入的跳变 （**PFLG**.HIES）
- 正确霍尔事件的发生 （**PFLG**.CHES）
- 错误霍尔事件的发生 （**PFLG**.WHES）
- 多通道序列的映射传送 （**PFLG**.MSTS）

最后一个中断源在每次多通道模式更新时被触发（PIFMST），这意味着 POSIFx.MOUT[15:0] 输出被更新为新值 （见**图 21-6**）。

这些中断源中每一个都可以被单独使能 / 禁止。软件也可以通过向 **SPFLG** 和 **RPFLG** 寄存器的特定域写入 1_B 来对特定标志进行置位和清零。通过使能一个中断源，每当发生一次标志置位操作时，都会产生一个中断脉冲，与该标志是否已经被置位无关。

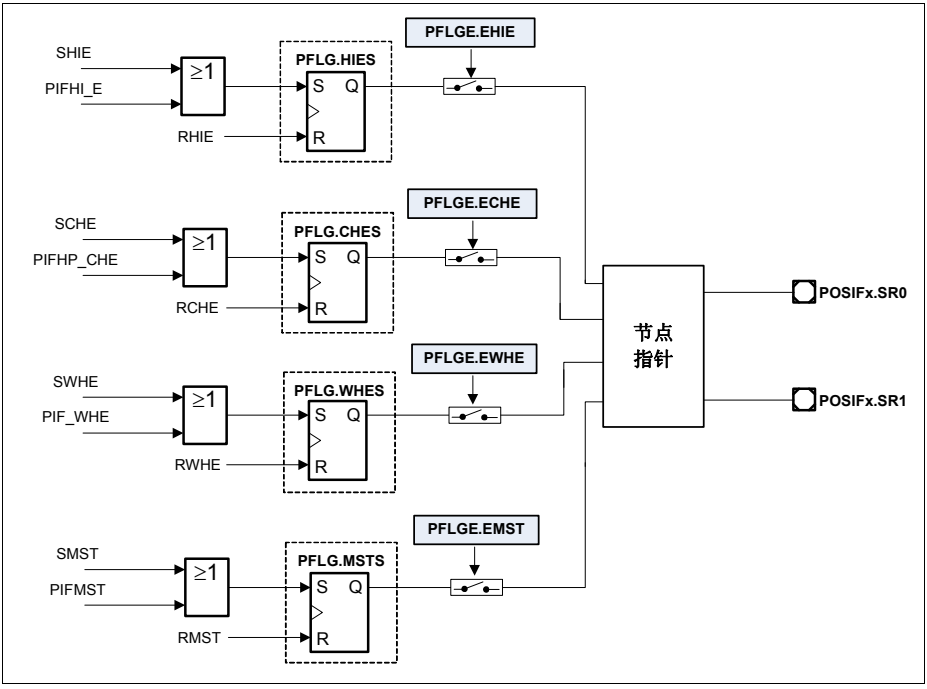


图 21-23　　霍尔传感器模式标志

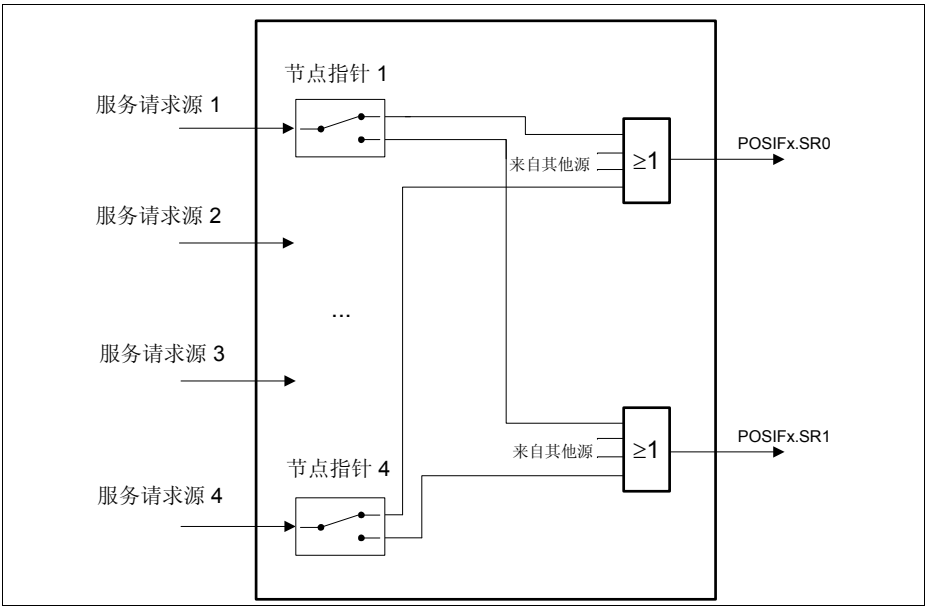


图 21-24　　中断节点指针概览 – 霍尔传感器模式

21.3.2　正交解码器标志

正交解码器模式有 5 个标志，这些标志可以被单独使能为中断源。除了这 5 个标志之外，与多通道模式连接的标志也可用：多通道序列更新 （**PFLG**.MSTS）和错误霍尔事件（**PFLG**.WHES）。如果选择正交模式和多通道独立功能，这可能是很有用的。

这 5 个标志是：

- 索引事件检测 (**PFLG**.INDXS)
- 相位检测错误 (**PFLG**.ERRS)
- 正交时钟产生 (**PFLG**.CNTS)
- 周期时钟产生 （**PFLG**.PCLKS)
- 方向改变 （**PFLG**.DIRS）

通过使能一个中断源，每当发生一次标志置位操作时，都会产生一个中断脉冲，与该标志是否已经被置位无关。

每当检测到索引事件时，索引事件检测标志就会被置 1。

当检测到相位信号发生无效跳变时，相位错误标志就会置 1，见**图 21-11**。

正交时钟和周期时钟标志按**图 21-12** 的时序产生。

每当电机旋转方向改变时，方向改变标志被置 1。

每个标志都能由软件通过写寄存器 **SPFLG** 和 **RPFLG** 的特定域来独立置位 / 清零，见**图 21-25**。

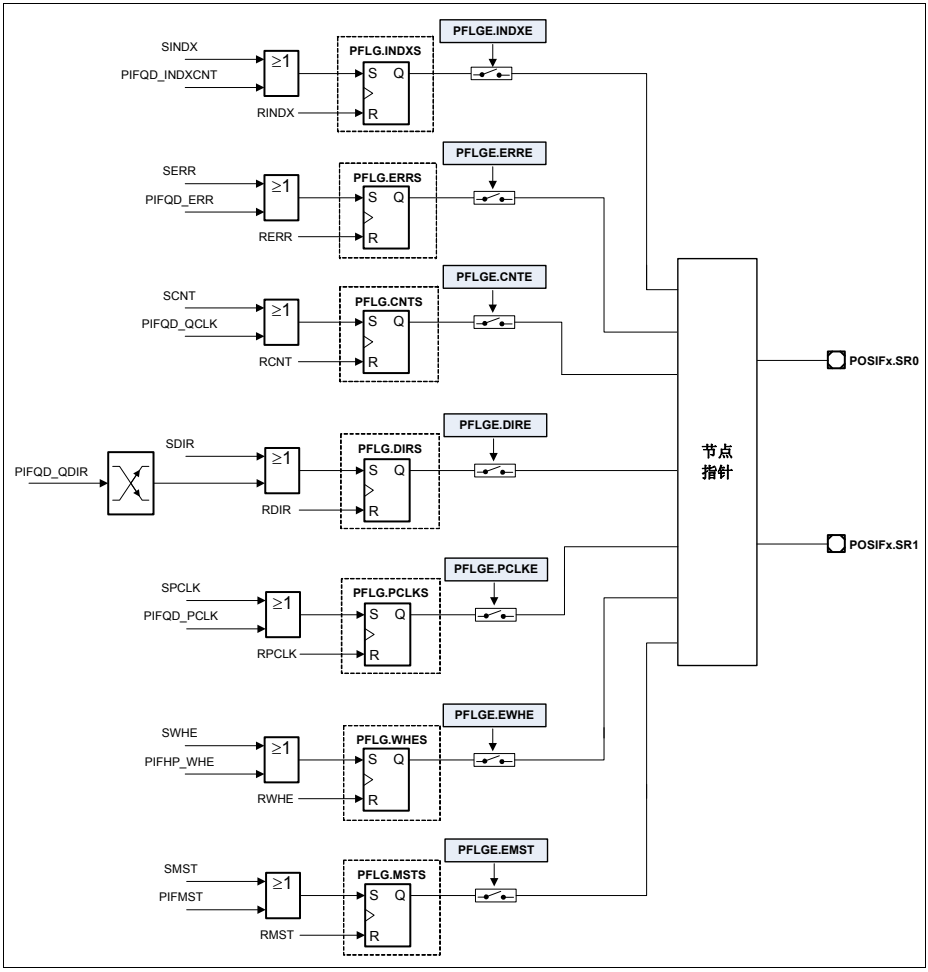


图 21-25　　正交解码器标志

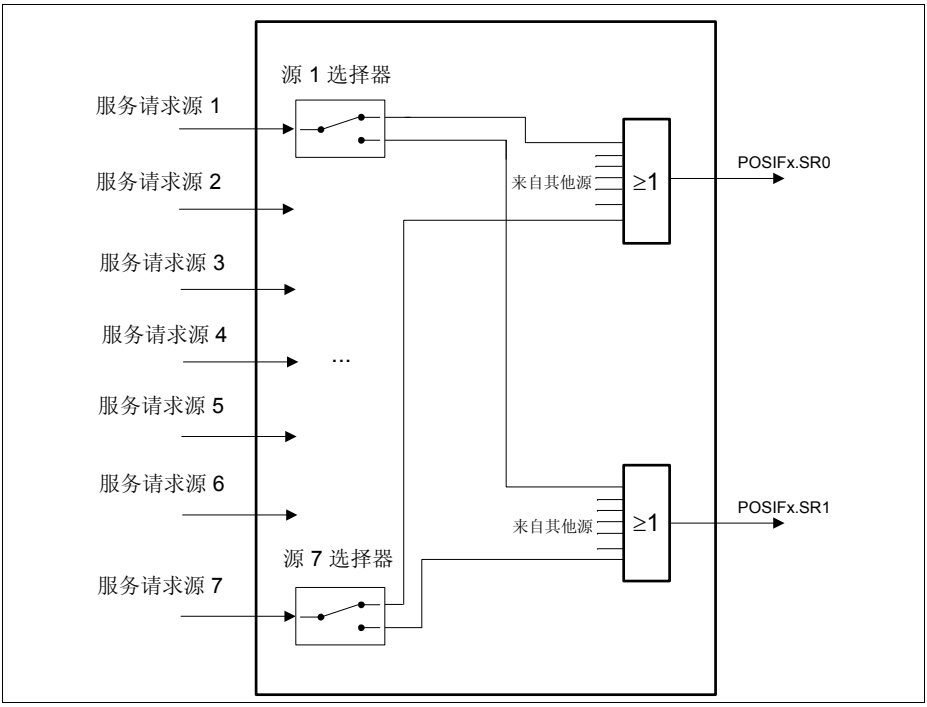


图 21-26　　中断节点指针概览 – 正交解码器模式

21.4　　调试行为

在挂起模式，整个模块都被停止。寄存器仍然能被 CPU 访问 （只读）。该模式对调试而言非常有用。例如，为了获得内部数值的快照，当前器件状态应被冻结。在挂起模式，所有的计数器都停止运行。 挂起模式对于寄存器位而言是非侵入式的。 这就意味着在进入或者离开挂起模式时寄存器位不能被硬件修改。

可以通过使用寄存器 **PSUS** 在内核级将模块配置为进入挂起模式。

该模块仅在挂起信号变为无效后才能正常工作。

21.5 电源、复位和时钟

以下各节描述 POSIF 的工作条件、特性和时序要求。所有的时序信息都与模块的时钟 f_{posif} 相关。

21.5.1 时钟

模块时钟

POSIF 模块的时钟为 f_{PCLK}，在 SCU 一章中描述。

POSIF 模块的总线接口时钟为 f_{MCLK}，在 SCU 一章中描述。

可以通过一个特定的系统控制位来禁止 POSIF 的模块时钟；然而，不同的 POSIF 实例或捕获 / 比较单元可能依赖于 f_{posif}。有关本产品时钟方案的完整描述请见 SCU 一章。

外部信号

与霍尔传感器和旋转编码器输入连接的外部信号的最大频率见**表 21-4**。

表 21-4 外部霍尔 / 旋转信号工作条件

参数	符号	数值			单位	备注 / 测试条件
		Min.	Typ.	Max.		
频率	f_{esig}	–	–	$f_{posif}/4$	MHz	
接通时间	ton_{esig}	$2T_{posif}$	–	–	ns	
关断时间	$toff_{esig}$	$2T_{posif}$	–	–	ns	

21.5.2 电源

POSIF 位于电源核心区域，因此不需要特别注意上电或下电顺序。对不同电源区域的解释，请参见 SCU （系统控制单元）一章。

21.6 初始化和系统相关性

21.6.1 初始化

对于使用 POISF 的应用，初始化过程如下：

第 1 步：通过专用的 SCU 寄存器 CGATCLR0 使能 POSIF 时钟。

第 2 步：通过 **PCONF** 寄存器配置 POSIF 的工作模式。

第 3 步：配置 POSIF 寄存器与所希望的工作模式连接：

- 霍尔传感器模式：**HALPS**/**HALP**, **MCSM**/**MCM**
- 正交解码器模式：**QDC**
- 独立多通道模式：**MCSM**/**MCM**

第 4 步：通过向 **MCMS**.STHR 和 **MCMS**.STMR 域写入 1_B，将预编程序列应用到 **HALP** 和 **MCM** 寄存器。

第 5 步：通过 **PFLGE** 寄存器配置 POSIF 的中断 / 服务请求。

第 6 步：配置与 POSIF 工作模式相关联的捕获 / 比较单元。

第 7 步：如果在相关联的捕获 / 比较单元内部使用了 POSIF 的同步启动功能，则仅需将模块的运行位置 1，即向 **PRUNS**.SRB 写入 1_B。

第 8 步：如果在相关联的捕获 / 比较单元内部不使用 POSIF 的同步启动功能，则通过向 **PRUNS**.SRB 写入 1_B 首先启动 POSIF。然后，通过访问每个定时器片内的特殊位域，或通过使用 SCU 的 CCUCON 寄存器的同步起动功能，来启动相关联的捕获 / 比较单元定时器片。

注：　由于电机本身有不同的启动条件（并且软件控制回路的实现方案也不相同），在软件和硬件控制回路被完全锁定之前会收到一些错误的中断触发。为了克服这一问题，软件可以在启动期间忽略中断，或者仅在检测到软件和硬件之间正确锁定之后才使能中断源。

21.6.2　系统相关性

每个 POSIF 都对模块和总线时钟频率有着不同的相关性。这种相关性在 SCU 和系统架构两章中描述。

就不同的时钟工作频率来说，几个外设之间也可能存在相关性。在配置 POSIF 与某些其他外设之间的连接之前就应该解决这种相关性。

为了更好地实现 POSIF 和系统操作，必须考虑以下几点：

:

- POSIF 模块时钟必须最多比模块总线接口时钟快两倍。
- POSIF　的模块输入触发必须不能超过模块时钟频率 （如果触发信号在器件内部产生）。
- POSIF 的模块输入触发必须不能超过 **21.5.1 节**中规定的频率。
- 将 POSIF 输出用作其他模块的触发信号 / 功能时，必须在终端对其频率进行交叉检测。
- 复位后启用和停止POSIF，可能在其他模块中引起不希望的操作。如果模块将POSIF输出用作触发 / 功能，该问题就可能出现。

21.7 寄存器

寄存器概述

绝对寄存器地址通过下面的加法计算：
模块基地址 + 偏移地址

表 21-5 寄存器地址空间

模块	基地址	结束地址	备注
POSIF0	50010000_H	$5001FFFF_H$	

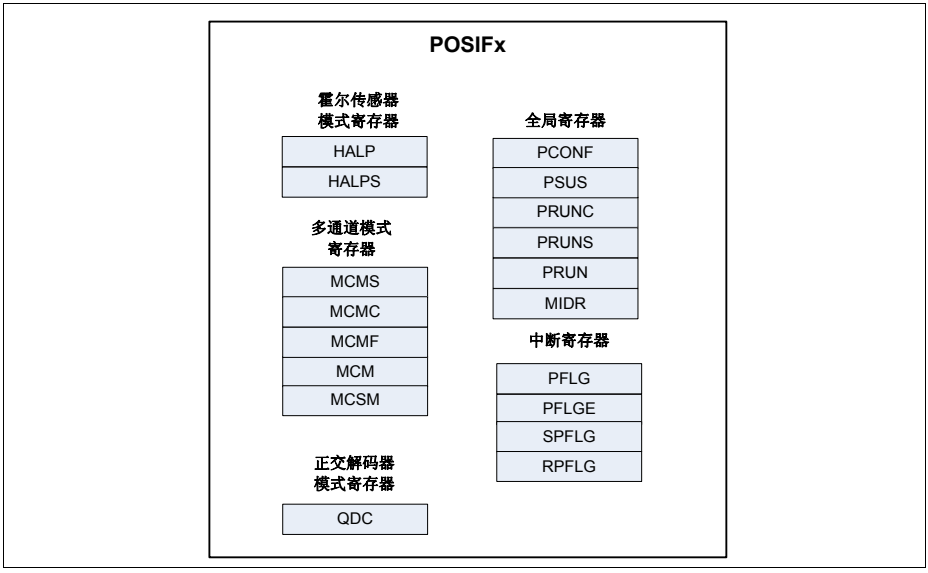


图 21-27 POSIF 寄存器概览

表 21-6 POSIF 寄存器一览表

简称	描述	偏移地址[1)]	访问模式		描述见
			读	写	
POSIF 内核寄存器					
PCONF	全局控制寄存器	0000_H	U, PV	U, PV	页 21-33

表 21-6 POSIF 寄存器一览表 (续表)

简称	描述	偏移地址[1]	访问模式		描述见
			读	写	
PSUS	挂起配置	0004_H	U, PV	U, PV	页 21-36
PRUNS	POSIF 运行位置位	0008_H	U, PV	U, PV	页 21-37
PRUNC	POSIF 运行位清除	$000C_H$	U, PV	U, PV	页 21-38
PRUN	POSIF 运行位状态	0010_H	U, PV	BE	页 21-39
PDBG	调试设计寄存器	0100_H	U, PV	BE	页 21-39
MIDR	模块标识寄存器	0020_H	U, PV	BE	页 21-40
霍尔传感器模式寄存器					
HALP	霍尔传感器当前和预期序列	0030_H	U, PV	BE	页 21-41
HALPS	霍尔传感器当前和预期映射序列	0034_H	U, PV	U, PV	页 21-42
多通道模式寄存器					
MCM	多通道模式序列	0040_H	U, PV	BE	页 21-43
MCSM	多通道模式映射序列	0044_H	U, PV	U, PV	页 21-44
MCMS	多通道模式控制置位	0048_H	U, PV	U, PV	页 21-44
MCMC	多通道模式控制清除	$004C_H$	U, PV	U, PV	页 21-45
MCMF	多通道模式标志状态	0050_H	U, PV	BE	页 21-46
正交解码器模式寄存器					
QDC	正交解码器配置	0060_H	U, PV	U, PV	页 21-47
中断寄存器					
PFLG	POSIF 中断状态	0070_H	U, PV	BE	页 21-48
PFLGE	POSIF 中断使能	0074_H	U, PV	U, PV	页 21-49
SPFLG	中断置位寄存器	0078_H	U, PV	U, PV	页 21-51
RPFLG	中断清除寄存器	$007C_H$	U, PV	U, PV	页 21-53

1) 绝对寄存器地址按下式计算：
模块基址 + 偏移地址 （本列所示）。

21.7.1 全局寄存器

PCONF

该寄存器包含 POSIF 操作的全局配置：操作模式、输入选择、滤波器配置。

PCONF
POSIF 配置 **(0000_H)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0		LPC		EWIL	EWIE	EWIS		MSYNS		MSES	MSETS			SPES	DSEL
r		rw		rw	rw	rw		rw		rw	rw			rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		INSEL2		INSEL1		INSEL0		0		MCUE	HIDG	0	QDCM	FSEL	
r		rw		rw		rw		r		rw	rw	r	rw	rw	

域	位	类型	描述
FSEL	[1:0]	rw	**功能选择器** 00_B 霍尔传感器模式使能 01_B 正交解码器模式使能 10_B 独立多通道模式使能 11_B 正交解码器和独立多通道模式使能
QDCM	2	rw	**位置解码器模式选择** 该域选择位置解码器功能块工作在正交模式还是方向计数模式。在正交模式，位置编码器提供相位信号；在方向计数模式，位置编码器提供一个时钟和一个方向信号。 0_B 位置编码器工作在正交模式 1_B 位置编码器工作在方向计数模式
HIDG	4	rw	**空闲产生使能** 将该域置为 1_B 会禁止产生空闲 （IDLE）信号。该操作强制清除多通道模式和运行位。
MCUE	5	rw	**多通道序列软件更新使能** 0_B 通过硬件控制多通道序列更新 1_B 通过软件控制多通道序列更新

域	位	类型	描述
INSEL0	[9:8]	rw	**相位 A/ 霍尔输入 1 选择器** 该域选择哪个输入用于相位 A 或霍尔输入 1 功能（取决于该模块被设置为正交解码器模式还是霍尔传感器模式）： 00_B POSIFx.IN0A 01_B POSIFx.IN0B 10_B POSIFx.IN0C 11_B POSIFx.IN0D
INSEL1	[11:10]	rw	**相位 B/ 霍尔输入 2 选择器** 该域选择哪个输入用于相位 B 或霍尔输入 2 功能（取决于该模块被设置为正交解码器模式还是霍尔传感器模式）： 00_B POSIFx.IN1A 01_B POSIFx.IN1B 10_B POSIFx.IN1C 11_B POSIFx.IN1D
INSEL2	[13:12]	rw	**索引 / 霍尔输入 3 选择器** 该域选择哪个输入用于索引或霍尔输入 3 功能 （取决于该模块被设置为正交解码器模式还是霍尔传感器模式）： 00_B POSIFx.IN2A 01_B POSIFx.IN2B 10_B POSIFx.IN2C 11_B POSIFx.IN2D
DSEL	16	rw	**延时引脚选择器** 该域选择哪个输入用于触发霍尔输入的边沿检测和霍尔输入的实际采样之间的延时结束。 0_B POSIFx.HSDA 1_B POSIFx.HSDB
SPES	17	rw	**采样触发信号的边沿选择器** 该域选择使用信号的哪个边沿触发对霍尔输入的采样，信号是从 POSIFx.HSD[B...A] 中选择的。 0_B 上升沿 1_B 下降沿

域	位	类型	描述
MSETS	[20:18]	rw	**序列更新信号选择** 选择用于启动一次多通道序列更新的输入信号。 000_B POSIFx.MSETA 001_B POSIFx.MSETB 010_B POSIFx.MSETC 011_B POSIFx.MSETD 100_B POSIFx.MSETE 101_B POSIFx.MSETF 110_B POSIFx.MSETG 111_B POSIFx.MSETH
MSES	21	rw	**多通道模式更新触发边沿** 0_B 用于启动序列更新的信号为上升沿有效 1_B 用于启动序列更新的信号为下降沿有效
MSYNS	[23:22]	rw	**PWM 同步信号选择器** 该域选择哪个输入用作多通道序列更新 （与 PWM 信号同步）的触发信号。 00_B POSIFx.MSYNCA 01_B POSIFx.MSYNCB 10_B POSIFx.MSYNCC 11_B POSIFx.MSYNCD
EWIS	[25:24]	rw	**错误霍尔事件选择** 00_B POSIFx.EWHEA 01_B POSIFx.EWHEB 10_B POSIFx.EWHEC 11_B POSIFx.EWHED
EWIE	26	rw	**外部错误霍尔事件使能** 0_B 外部错误霍尔事件仿真信号 POSIFx.EWHE[D...A] 被禁止 。 1_B 外部错误霍尔事件仿真信号 POSIFx.EWHE[D...A] 被使能。
EWIL	27	rw	**外部错误霍尔事件有效电平** 0_B POSIFx.EWHE[D...A] 信号为高电平有效 1_B POSIFx.EWHE[D...A] 信号为低电平有效

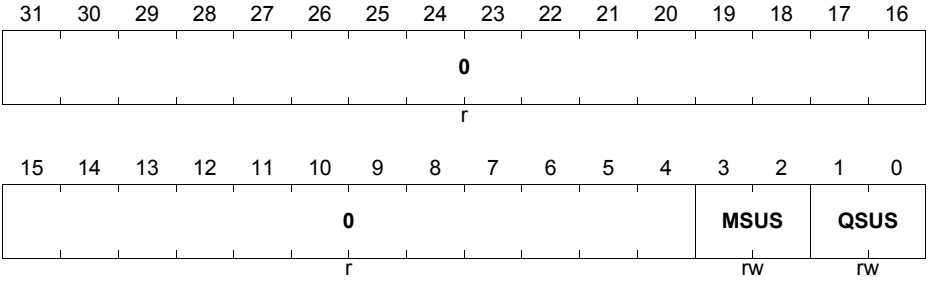
域	位	类型	描述
LPC	[30:28]	rw	**低通滤波器配置** 000_B 低通滤波被禁止 001_B 低通 1 个时钟周期 010_B 低通 2 个时钟周期 011_B 低通 4 个时钟周期 100_B 低通 8 个时钟周期 101_B 低通 16 个时钟周期 110_B 低通 32 个时钟周期 111_B 低通 64 个时钟周期
0	3, [7:6], [15:14], 31	r	**保留** 读访问总是返回 0。

PSUS

该寄存器包含 POSIF 的挂起配置。

PSUS

POSIF 挂起配置 **(0004_H)** **复位值 : 00000000_H**



域	位	类型	描述
QSUS	[1:0]	rw	**正交模式挂起配置** 该域控制正交解码模式进入挂起。 00_B 忽略挂起请求 01_B 立即停止 10_B 下一索引发生时挂起 11_B 下一相位 （相位 A 或相位 B）发生时挂起

域	位	类型	描述
MSUS	[3:2]	rw	**多通道模式挂起配置** 该域控制多通道模式如何进入挂起。霍尔传感器模式也由该配置控制。 00_B 忽略挂起请求 01_B 立即停止。不将多通道序列设置成复位值。 10_B 立即停止。将多通道序列设置成复位值。 11_B 与 PWM 信号同步挂起。在同步的同时多通道模式被设置成复位值。
0	[31:4]	r	**保留** 读访问总是返回 0。

PRUNS

通过该寄存器可以置位该模块的运行位。

PRUNS

POSIF 运行位置 1 **(0008_H)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

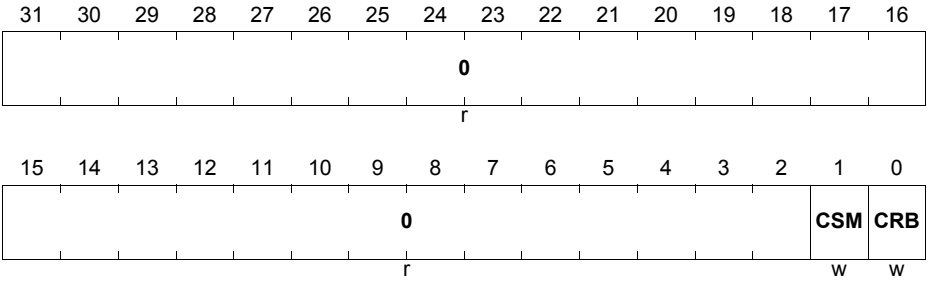
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															**SRB**
r															w

域	位	类型	描述
SRB	0	w	**运行位置 1** 向该位写入 1_B 将该模块的运行位置 1。 读访问总是返回 0。
0	[31:1]	r	**保留** 读访问总是返回 0。

PRUNC

通过该寄存器可以清除模块的运行位和内部状态机。

PRUNC
POSIF 运行位清除 **(000C$_H$)** **复位值：00000000$_H$**



域	位	类型	
CRB	0	w	**清除运行位** 向该位写入 1_B 清除模块的运行位。模块停止工作。 读访问总是返回 0。
CSM	1	w	**清除当前内部状态** 向该位写入 1_B 复位正交解码器的状态机和霍尔传感器的当前状态，以及多通道模式的寄存器。标志和静态配置位域不被清除。 读访问总是返回 0。
0	[31:2]	r	**保留** 读访问总是返回 0。

PRUN

该寄存器包含 POSIF 的运行位状态。

PRUN
POSIF 运行位状态 **(0010_H)** **复位值：00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0															RB
r															rh

域	位	类型	描述
RB	0	rh	**运行位** 该域指示模块处于运行态还是空闲态。 0_B 空闲 1_B 运行
0	[31:1]	r	**保留** 读总是返回 0。

PDBG

POSIF 状态机和霍尔采样值的当前状态调试寄存器。

PDBG
POSIF 调试寄存器 **(0100_H)** **复位值：00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				LPP2						LPP1					
r				rh						rh					

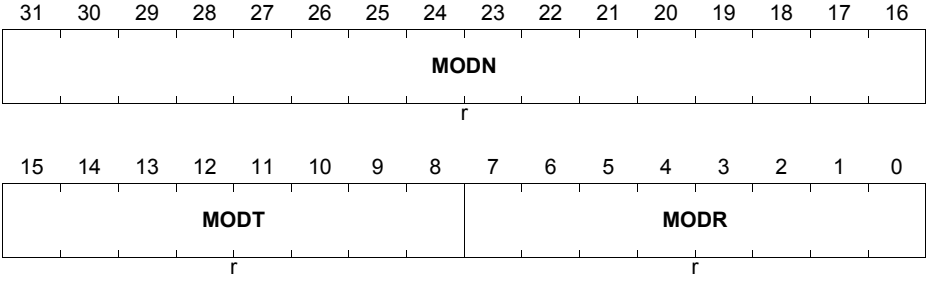
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				LPP0				HSP			IVAL	QPSV		QCSV	
r				rh				rh			rh	rh		rh	

域	位	类型	描述
QCSV	[1:0]	rh	**正交解码器当前状态** QCSV[0] - 相位 A QCSV[1] - 相位 B
QPSV	[3:2]	rh	**正交解码器前一状态** QPSV[0] - 相位 A QPSV[1] - 相位 B
IVAL	4	rh	**当前索引值**
HSP	[7:5]	rh	**霍尔当前采样序列** HSP[0] - 霍尔输入 1 HSP[1] - 霍尔输入 2 HSP[2] - 霍尔输入 3
LPP0	[13:8]	rh	**POSI0 低通滤波器的时钟周期数**
LPP1	[21:16]	rh	**POSI1 低通滤波器的时钟周期数**
LPP2	[27:22]	rh	**POSI2 低通滤波器的时钟周期数**
0	[31:28], [15:14]	r	**保留** 读访问总是返回 0。

MIDR

该寄存器包含模块的标识号。

MIDR
模块标识寄存器 **(0020_H)** **复位值：$00A8C0XX_H$**



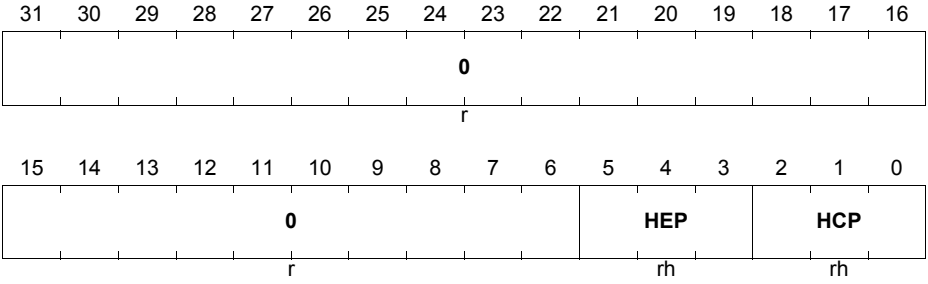
域	位	类型	描述
MODR	[7:0]	r	**模块修订** 该位域标识模块实现的修订号 （取决于设计步骤）。
MODT	[15:8]	r	**模块类型**
MODN	[31:16]	r	**模块编号**

21.7.2 霍尔传感器模式寄存器

HALP

该寄存器包含霍尔预期序列和霍尔当前序列。

HALP
霍尔传感器序列 **(0030_H)** **复位值：00000000_H**



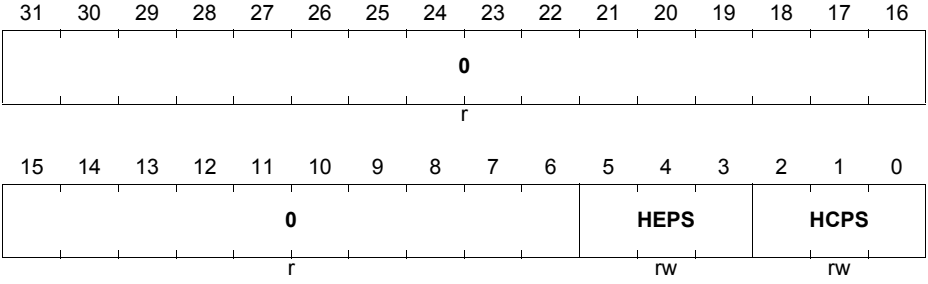
域	位	类型	描述
HCP	[2:0]	rh	**霍尔当前序列** 该域包含霍尔当前序列。每当发生正确的霍尔事件时，该域由 **HALPS**.HCPS 值更新。 HCP[0] - 霍尔输入 1 HCP[1] - 霍尔输入 2 HCP[2] - 霍尔输入 3
HEP	[5:3]	rh	**霍尔预期序列** 该域包含霍尔预期序列。每当发生正确的霍尔事件时，该域由 **HALPS**.HEPS 值更新。 HEP[0] - 霍尔输入 1 HEP[1] - 霍尔输入 2 HEP[2] - 霍尔输入 3

域	位	类型	描述
0	[31:6]	r	**保留** 读访问总是返回 0。

HALPS

该寄存器包含当下一正确霍尔事件发生时将被加载到 **HALP** 寄存器的值。

HALPS
霍尔传感器映射序列 **(0034$_H$)** **复位值：00000000$_H$**



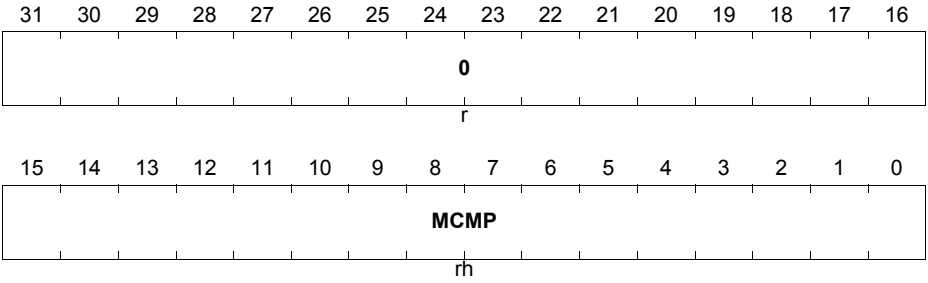
字段	位	类型	描述
HCPS	[2:0]	rw	**映射霍尔当前序列** 该域包含下一霍尔当前序列值。每当发生正确的霍尔事件时，该域被设置到 **HALP**.HCP 域。 HCPS[0] - 霍尔输入 1 HCPS[1] - 霍尔输入 2 HCPS[2] - 霍尔输入 3
HEPS	[5:3]	rw	**映射霍尔预期序列** 该域包含下一霍尔预期序列值。每当发生正确的霍尔事件时，该域被设置到 **HALP**.HEP 域。 HEPS[0] - 霍尔输入 1 HEPS[1] - 霍尔输入 2 HEPS[2] - 霍尔输入 3
0	[31:6]	r	**保留** 读访问总是返回 0。

21.7.3 多通道模式寄存器

MCM

该寄存器包含多通道序列的值，该值施加于 POSIFx.OUT[15:0] 输出。

MCM
多通道序列 **(0040_H)** **复位值：00000000_H**



域	位	类型	描述
MCMP	[15:0]	rh	**多通道序列** 该域包含多通道序列，将被施加于多通道输出 POSIFx.MOUT[15:0]。 在每次多通道序列更新被触发时，该域被更新为 **MCSM**.MCMPS 的值。
0	[31:16]	r	**保留** 读访问总是返回 0

MCSM

该寄存器包含在下一次多通道更新触发发生时将被加载到 **MCM** 寄存器的值。

MCSM
多通道映射序列 **(0044$_H$)** **复位值：00000000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCMPS															
rw															

域	位	类型	描述
MCMPS	[15:0]	rw	**映射多通道序列** 该字段包含下一多通道序列。每次多通道序列传送被触发时，该值被传入到 **MCM**.MCMP 域。
0	[31:16]	r	**保留** 读访问总是返回 0。

MCMS

通过该寄存器可以请求一次多通道序列更新。也可以通过该寄存器请求一次多通道和霍尔传感器序列的立即更新，而不需要等待硬件触发。

MCMS
多通道序列控制置位 **(0048$_H$)** **复位值：00000000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

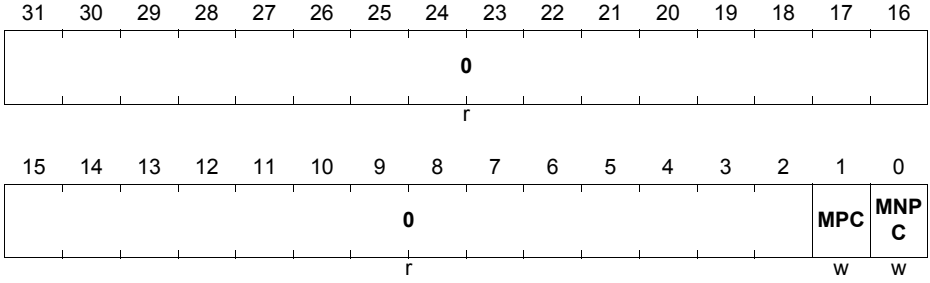
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													**STMR**	**STHR**	**MNPS**
r													w	w	w

域	位	类型	描述
MNPS	0	w	**多通道模式更新使能置位** 向该域写入 1_B 使能多通道模式更新 （将 **MCMF**.MSS 位置 1）。由于仍然需要触发信号来使更新与 PWM 同步，所以该更新不会立刻执行。 读访问总是返回 0
STHR	1	w	**霍尔序列映射传送请求** 向该域写入 1_B 使 **HALP**.HCP 和 **HALP**.HEP 域立即更新。 读访问总是返回 0。
STMR	2	w	**多通道映射传送请求** 向该域写入 1_B 使 **MCM**.MCMP 域立即更新。 读访问总是返回 0。
0	[31:3]	r	**保留** 读访问总是返回 0。

MCMC

通过该寄存器可以取消多通道序列更新，并将多通道序列并恢复为默认值。

MCMC
多通道序列控制清除 **($004C_H$)** **复位值：00000000_H**



域	位	类型	描述
MNPC	0	w	**多通道模式更新使能清除** 向该域写入 1_B 清除 **MCMF**.MSS 位。 读访问总是返回 0。

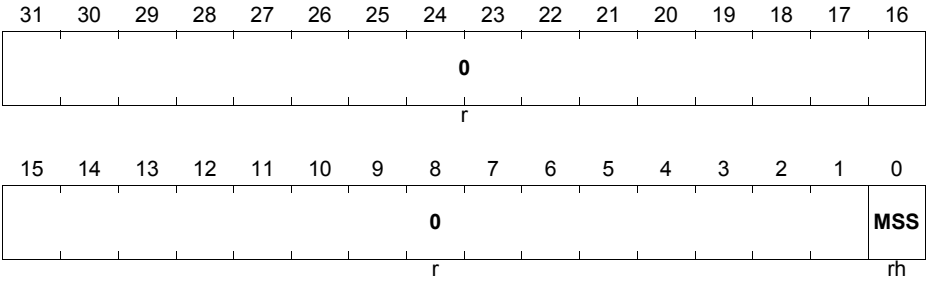
域	位	类型	描述
MPC	1	w	**多通道模式清除** 向该域写入 1_B 将多通道序列值清为 0000_H。 读访问总是返回 0。
0	[31:2]	r	**保留** 读访问总是返回 0。

MCMF

该寄存器包含多通道更新请求的状态。

MCMF

多通道序列控制标志 **(0050_H)** **复位值 : 00000000_H**



域	位	状态	描述
MSS	0	rh	**多通道模式更新状态** 该域指示多通道序列是否准备好被更新。在该域被置 1 的情况下，当从 POSIFx.MSYNC[D..A] 选择的触发信号变为有效时，多通道序列被更新。 0_B 置位多通道序列更新 1_B 不置位多通道模式更新
0	[31:1]	r	**保留** 读访问总是返回 0。

21.7.4 正交解码器寄存器

QDC

该寄存器包含正交解码器模式下的操作配置。

QDC
正交解码器控制 **(0060_H)** **复位值 : 00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							DVAL	0		ICM		0	PHS	PBLS	PALS
r							rh	r		rw		r	rw	rw	rw

域	位	状态	描述
PALS	0	rw	**相位 A 电平选择器** 0_B 相位 A 高电平有效 1_B 相位 A 低电平有效
PBLS	1	rw	**相位 B 电平选择器** 0_B 相位 B 高电平有效 1_B 相位 B 低电平有效
PHS	2	rw	**相位信号调换** 0_B 相位 A 是顺时针旋转的前导信号。 1_B 相位 B 是顺时针旋转的前导信号。
ICM	[5:4]	rw	**索引标记产生控制** 该域控制连接到输出引脚 POSIFx.OUT3 的索引标记的产生。 00_B 在 POSIFx.OUT3 不产生索引标记 01_B 在 POSIFx.OUT3 仅产生第一个索引 10_B 在 POSIFx.OUT3 产生所有索引 11_B 保留
DVAL	8	rh	**当前旋转方向** 0_B 逆时针旋转 1_B 顺时针旋转
0	3, [7:6], [31:9]	r	**保留** 读访问总是返回 0。

21.7.5 中断寄存器

PFLG

该寄存器包含模块的所有中断标志的状态。

PFLG

POSIF 中断标志 **(0070$_H$)** **复位值：00000000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			PCLKS	DIRS	CNTS	ERRS	INDXS	0			MSTS	0	HIES	WHES	CHES
r			rh	rh	rh	rh	rh	r			rh	r	rh	rh	rh

域	位	状态	描述
CHES	0	rh	**正确霍尔事件状态** 0_B 未检测到正确霍尔事件 1_B 检测到正确霍尔事件
WHES	1	rh	**错误霍尔事件状态** 0_B 未检测到错误霍尔事件 1_B 检测到错误霍尔事件
HIES	2	rh	**霍尔输入更新状态** 0_B 未检测到霍尔输入跳变 1_B 检测到霍尔输入跳变
MSTS	4	rh	**多通道序列映射传送状态** 0_B 映射传送未完成 1_B 映射传送已完成
INDXS	8	rh	**正交索引状态** 0_B 未检测到索引事件 1_B 检测到索引事件
ERRS	9	rh	**正交相位错误状态** 0_B 未检测到相位错误事件 1_B 检测到相位错误事件

域	位	状态	描述
CNTS	10	rh	**正交时钟状态** 0_B 未产生正交时钟 1_B 产生了正交时钟
DIRS	11	rh	**正交方向改变** 0_B 未检测到方向改变 1_B 检测到方向改变
PCLKS	12	rh	**正交周期时钟状态** 0_B 未产生周期时钟 1_B 产生了周期时钟
0	3, [7:5], [31:13]	r	**保留** 读访问总是返回 0。

PFLGE

通过该寄存器可以使能或禁止每个可用的中断源。 也可以选择将中断发往哪一个服务请求线。

PFLGE

POSIF 中断使能 **(0074_H)** **复位值：00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0			**PCLSEL**	**DIRSEL**	**CNTSEL**	**ERRSEL**	**INDSEL**	**0**			**MSTSEL**	**0**	**HIESEL**	**WHESEL**	**CHESEL**
r			rw	rw	rw	rw	rw	r			rw	r	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**EPCLK**	**EDIR**	**ECNT**	**EERR**	**EINDX**	**0**			**EMST**	**0**	**EHIE**	**EWHE**	**ECHE**
r			rw	rw	rw	rw	rw	r			rw	r	rw	rw	rw

域	位	状态	描述
ECHE	0	rw	**正确霍尔事件使能** 0_B 禁止正确霍尔事件中断 1_B 使能正确霍尔事件中断
EWHE	1	rw	**错误霍尔事件使能** 0_B 禁止错误霍尔事件中断 1_B 使能错误霍尔事件中断

域	位	状态	描述
EHIE	2	rw	**霍尔输入更新使能** 0_B 禁止霍尔输入中断更新 1_B 使能霍尔输入中断更新
EMST	4	rw	**多通道序列映射传送使能** 0_B 禁止映射传送事件中断 1_B 使能映射传送事件中断
EINDX	8	rw	**正交索引事件使能** 0_B 禁止索引事件中断 1_B 使能索引事件中断
EERR	9	rw	**正交相位错误使能** 0_B 禁止相位错误事件中断 1_B 使能相位错误事件中断
ECNT	10	rw	**正交时钟中断使能** 0_B 禁止正交时钟事件中断 1_B 使能正交时钟事件中断
EDIR	11	rw	**正交方向改变中断使能** 0_B 禁止方向改变事件中断 1_B 使能方向改变事件中断
EPCLK	12	rw	**正交周期时钟中断使能** 0_B 禁止正交周期时钟事件中断 1_B 使能正交周期时钟事件中断
CHESEL	16	rw	**正确霍尔事件服务请求选择器** 0_B 正确的霍尔事件中断发往 POSIFx.SR0 1_B 正确的霍尔事件中断发往 POSIFx.SR1
WHESEL	17	rw	**错误霍尔事件服务请求选择器** 0_B 错误的霍尔事件中断发往 POSIFx.SR0。 1_B 错误的霍尔事件中断发往 POSIFx.SR1
HIESEL	18	rw	**霍尔输入更新事件服务请求选择器** 0_B 霍尔输入更新事件中断发往 POSIFx.SR0 1_B 霍尔输入更新事件中断发往 POSIFx.SR1
MSTSEL	20	rw	**多通道序列更新事件服务请求选择器** 0_B 多通道序列更新事件中断发往 POSIFx.SR0 1_B 多通道序列更新事件中断发往 POSIFx.SR1
INDSEL	24	rw	**正交索引事件服务请求选择器** 0_B 正交索引事件中断发往 POSIFx.SR0 1_B 正交索引事件中断发往 POSIFx.SR1

域	位	状态	描述
ERRSEL	25	rw	**正交相位错误事件服务请求选择器** 0_B 正交相位错误事件中断发往 POSIFx.SR0 1_B 正交相位错误事件中断发往 POSIFx.SR1
CNTSEL	26	rw	**正交时钟事件服务请求选择器** 0_B 正交时钟事件中断发往 POSIFx.SR0 1_B 正交时钟事件中断发往 POSIFx.SR1
DIRSEL	27	rw	**正交方向更新事件服务请求选择器** 0_B 正交方向更新事件中断发往 POSIFx.SR0 1_B 正交方向更新事件中断发往 POSIFx.SR1
PCLSEL	28	rw	**正交周期时钟事件服务请求选择器** 0_B 正交周期时钟事件中断发往 POSIFx.SR0 1_B 正交周期时钟事件中断发往 POSIFx.SR1
0	3, [7:5], [15:13], 19, [23:21], [31:29]	r	**保留** 读访问总是返回 0。

SPFLG

通过该寄存器软件可以设置一个特殊的中断状态标志。

SPFLG

POSIF 中断置位 **(0078_H)** **复位值 :00000000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			**SPCLK**	**SDIR**	**SCNT**	**SERR**	**SINDX**	**0**			**SMST**	**0**	**SHIE**	**SWHE**	**SCHE**
r			w	w	w	w	w	r			w	r	w	w	w

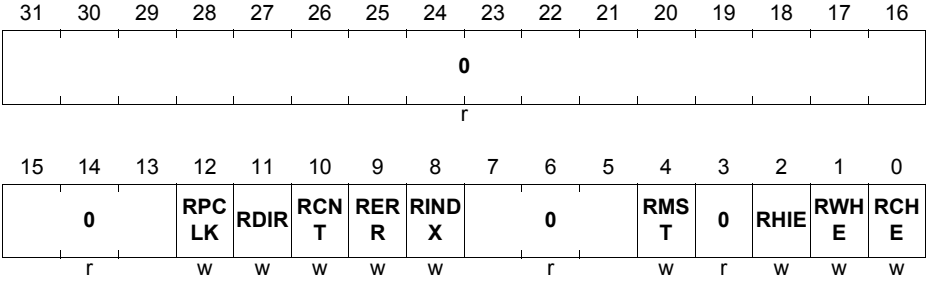
域	位	状态	描述
SCHE	0	w	**正确霍尔事件标志置位** 向该域写入 1_B 将 **PFLG**.CHES 位域置 1。产生一个中断脉冲。 读访问总是返回 0。
SWHE	1	w	**错误霍尔事件标志置位** 向该域写入 1_B 将 **PFLG**.PFLG 位域置 1。产生一个中断脉冲。 读访问总是返回 0。
SHIE	2	w	**霍尔输入更新事件标志置位** 向该域写入 1_B 将 **PFLG**.HIES 位域置 1。产生一个中断脉冲。 读访问总是返回 0。
SMST	4	w	**多通道序列映射传送标志置位** 向该域写入 1_B 将 **PFLG**.MSTS 位域置 1。产生一个中断脉冲。 读访问总是返回 0。
SINDX	8	w	**正交索引标志置位** 向该域写入 1_B 将 **PFLG**.INDXS 位域置 1。产生一个中断脉冲。 读访问总是返回 0.
SERR	9	w	**正交相位错误标志置位** 向该域写入 1_B 将 **PFLG**.ERRS 位域置 1。产生一个中断脉冲。 读访问总是返回 0.
SCNT	10	w	**正交时钟标志置位** 向该域写入 1_B 将 **PFLG**.CNTS 位域置 1。产生一个中断脉冲。 读访问总是返回 0。
SDIR	11	w	**正交方向标志置位** 向该域写入 1_B 将 **PFLG**.DIRS 位域置 1。产生一个中断脉冲。 读访问总是返回 0。
SPCLK	12	w	**正交周期时钟标志置位** 向该域写入 1_B 将 **PFLG**.PCLKS 位域置 1。产生一个中断脉冲。 读访问总是返回 0。

域	位	状态	描述
0	3, [7:5], [31:13]	r	**保留** 读访问总是返回 0。

RPFLG

通过该寄存器软件可以复位 / 清除一个特殊中断状态标志。

RPFLG
POSIF 中断清除 **($007C_H$)** 复位值：**00000000_H**



域	位	状态	描述
RCHE	0	w	**正确霍尔事件标志清除** 向该域写入 1_B 清除 **PFLG**.CHES 位域。 读访问总是返回 0。
RWHE	1	w	**错误霍尔事件标志清除** 向该域写 1_B 清除 **PFLG**.WHES 位域。 读访问总是返回 0。
RHIE	2	w	**霍尔输入更新事件标志清除** 向该域写入 1_B 清除 **PFLG**.HIES 位域。 读访问总是返回 0。
RMST	4	w	**多通道序列映射传送标志清除** 向该域写入 1_B 清除 **PFLG**.MSTS 位域。 读访问总是返回 0。
RINDX	8	w	**正交索引标志清除** 向该域写入 1_B 清除 **PFLG**.INDXS 位域。 读访问总是返回 0。

域	位	状态	描述
RERR	9	w	**正交相位错误标志清除** 向该域写入 1_B 清除 **PFLG**.ERRS 位域。 读访问总是返回 0。
RCNT	10	w	**正交时钟标志清除** 向该域写入 1_B 清除 **PFLG**.CNTS 位域。 读访问总是返回 0。
RDIR	11	w	**正交方向标志清除** 向该域写入 1_B 清除 **PFLG**.DIRS 位域。 读访问总是返回 0.
RPCLK	12	w	**正交周期时钟标志清除** 向该域写入 1_B 清除 **PFLG**.PCLKS 位域。 读访问总是返回 0。
0	3, [7:5], [31:13]	r	**保留** 读访问总是返回 0。

21.8 互连

以下表格描述了器件中每个 POSIF 模块的连接情况。

GPIO 的连接参见端口一章。

21.8.1 POSIF0 引脚

表 21-7 POSIF0 引脚连接

全局输入 / 输出	I/O	连接到	描述
POSIF0.CLK	I	PCLK	模块时钟与捕获比较使用的是同一个时钟
POSIF0.IN0A	I	P1.2	旋转编码器和霍尔传感器的共享连接
POSIF0.IN0B	I	P0.13	旋转编码器和霍尔传感器的共享连接
POSIF0.IN0C	I	VADC0.CBFLOUT0	旋转编码器和霍尔传感器的共享连接
POSIF0.IN0D	I	ERU0.PDOUT0	旋转编码器和霍尔传感器的共享连接
POSIF0.IN1A	I	P1.1	旋转编码器和霍尔传感器的共享连接

表 21-7 POSIF0 引脚连接 (续表)

全局输入 / 输出	I/O	连接到	描述
POSIF0.IN1B	I	P0.14	旋转编码器和霍尔传感器的共享连接
POSIF0.IN1C	I	VADC0.CBFLOUT1	旋转编码器和霍尔传感器的共享连接
POSIF0.IN1D	I	ERU0.PDOUT1	旋转编码器和霍尔传感器的共享连接
POSIF0.IN2A	I	P1.0	旋转编码器和霍尔传感器的共享连接
POSIF0.IN2B	I	P0.15	旋转编码器和霍尔传感器的共享连接
POSIF0.IN2C	I	VADC0.CBFLOUT2	旋转编码器和霍尔传感器的共享连接
POSIF0.IN2D	I	ERU0.PDOUT2	旋转编码器和霍尔传感器的共享连接
POSIF0.HSDA	I	CCU40.ST0	用于霍尔序列采样延时。
POSIF0.HSDB	I	无连接	用于霍尔序列采样延时。
POSIF0.EWHEA	I	VADC0.CBFLOUT3	错误霍尔事件仿真、陷阱等
POSIF0.EWHEB	I	ERU0.IOUT0	错误霍尔事件仿真、陷阱等
POSIF0.EWHEC	I	ERU0.IOUT3	错误霍尔事件仿真、陷阱等
POSIF0.EWHED	I	无连接	错误霍尔事件仿真、陷阱等
POSIF0.MSETA	I	CCU40.SR0	多序列更新置位 。 请求一个新的映射传递
POSIF0.MSETB	I	CCU40.ST1	多序列更新置位 。 请求一次新的映射传送
POSIF0.MSETC	I	CCU40.ST2	多序列更新置位 。 请求一次新的映射传送
POSIF0.MSETD	I	CCU40.ST3	多序列更新置位 。 请求一次新的映射传送
POSIF0.MSETE	I	CCU80.SR1	多序列更新置位 。 请求一次新的映射传送
POSIF0.MSETF	I	ERU0.IOUT2	多序列更新置位 。 请求一次新的映射传送
POSIF0.MSETG	I	ERU0.IOUT3	多序列更新置位 。 请求一次新的映射传送

表 21-7 POSIF0 引脚连接 (续表)

全局输入 / 输出	I/O	连接到	描述
POSIF0.MSETH	I	无连接	多序列更新置位 。 请求一次新的映射传送
POSIF0.MSYNCA	I	CCU80.PS1	更新多通道序列与映射传送的同步
POSIF0.MSYNCB	I	CCU80.PS3	更新多通道序列与映射传送的同步
POSIF0.MSYNCC	I	CCU40.PS1	更新多通道序列与映射传送的同步
POSIF0.MSYNCD	I	无连接	更新多通道序列与映射传送的同步
POSIF0.OUT0	O	CCU40.IN0E; CCU40.IN1E;	正交模式：正交时钟；霍尔传感器模式：霍尔输入边沿检测
POSIF0.OUT1	O	CCU40.IN0F; CCU40.IN1F; CCU40.IN2E;	正交模式：轴方向；霍尔传感器模式：正确霍尔事件
POSIF0.OUT2	O	CCU40.IN2F; CCU80.IN0D; CCU80.IN1D; CCU80.IN2D; CCU80.IN3D;	正交模式：速度的周期时钟；霍尔传感器模式：空闲 / 错误霍尔事件
POSIF0.OUT3	O	CCU40.IN0G; CCU40.IN1G; CCU40.IN2G; CCU40.IN3E;	正交模式：用于清除 / 捕获的索引事件；霍尔传感器模式：在霍尔传感器模式停止
POSIF0.OUT4	O	CCU40.IN1H; CCU40.IN2H;	正交模式：索引事件；霍尔传感器模式：多通道序列更新完成
POSIF0.OUT5	O	CCU40.IN3F; CCU80.IN0E; CCU80.IN1E; CCU80.IN2E; CCU80.IN3E;	同步启动
POSIF0.OUT6	O	CCU40.MCSS; CCU80.MCSS;	多通道序列更新请求
POSIF0.MOUT[0]	O	CCU80.MCI00;	多通道序列
POSIF0.MOUT[1]	O	CCU80.MCI01;	多通道序列

表 21-7 POSIF0 引脚连接 (续表)

全局输入 / 输出	I/O	连接到	描述
POSIF0.MOUT[2]	O	CCU80.MCI02;	多通道序列
POSIF0.MOUT[3]	O	CCU80.MCI03;	多通道序列
POSIF0.MOUT[4]	O	CCU80.MCI10;	多通道序列
POSIF0.MOUT[5]	O	CCU80.MCI11;	多通道序列
POSIF0.MOUT[6]	O	CCU80.MCI12;	多通道序列
POSIF0.MOUT[7]	O	CCU80.MCI13;	多通道序列
POSIF0.MOUT[8]	O	CCU80.MCI20;	多通道序列
POSIF0.MOUT[9]	O	CCU80.MCI21;	多通道序列
POSIF0.MOUT[10]	O	CCU80.MCI22;	多通道序列
POSIF0.MOUT[11]	O	CCU80.MCI23;	多通道序列
POSIF0.MOUT[12]	O	CCU80.MCI30;	多通道序列
POSIF0.MOUT[13]	O	CCU80.MCI31;	多通道序列
POSIF0.MOUT[14]	O	CCU80.MCI32;	多通道序列
POSIF0.MOUT[15]	O	CCU80.MCI33;	多通道序列
POSIF0.SR0	O	NVIC	服务请求线 0
POSIF0.SR1	O	NVIC; VADC0.BGREQTRO; VADC0.G0REQTRO; VADC0.G1REQTRO;	服务请求线 1

22 亮度和色彩控制单元 (BCCU)

BCCU 是面向 LED 照明应用的一个调光控制外设，可以控制多个 LED 通道。BCCU 为每个通道提供一个单比特 sigma-delta 位流来控制亮度。通过使用专门设计的调光引擎，可以按照指数曲线来渐变调节亮度，以使在人眼看来更为自然。该模块通过调节所选通道的相对强度进行色彩控制，使用线性行走方案实现色彩的平滑改变。该模块还支持大功率多通道 LED 灯具，它通过选择性地对控制位流 “ 打包 （packing）”，在输出端提供所设定的接通时间。

22.1 概述

XMC1300 中的 BCCU 包含 3 个完全相同的调光引擎和 9 个完全相同的通道。每个通道都可以产生一个灵活的单比特 sigma-delta 位流，该位流具有用户可调整的 12 位平均值。可以手动提供或由任一指数调光引擎提供这个平均值。控制位流由一个 12 位的 sigma-delta 调制器产生，还有一个可选的打包器（packer）可以用来降低输出切换的平均速率（通过强加一个设定的接通时间）。虽然 BCCU 模块的主要目标应用是多通道 LED 调光并支持色彩控制，但也可将其作为一个多通道数 / 模转换器使用，只需在输出端接低通滤波器即可。

22.1.1 功能特性

BCCU 提供下列功能：

- 3 个独立的调光引擎
- 12 位调光值
- 按照指数曲线调光并具有可调整的调光时间
- 9 个独立通道，每个通道都能产生一个单比特通断信号
- 12 位的通道强度，用于定义色彩
- 通道和调光引擎可自由互连
- 每个通道有一个输入乘法器和一个 12 位 sigma-delta 调制器
- 每个通道有一个位打包器 (bit-packer)，用于降低输出切换的平均速率
- 陷阱 （TRAP）功能
- 两种 ADC 触发模式

22.2 功能描述

每个通道都可以独立于其他通道工作，并可与任何一个调光引擎自由互连。触发和陷阱控制是所有通道统一处理的。

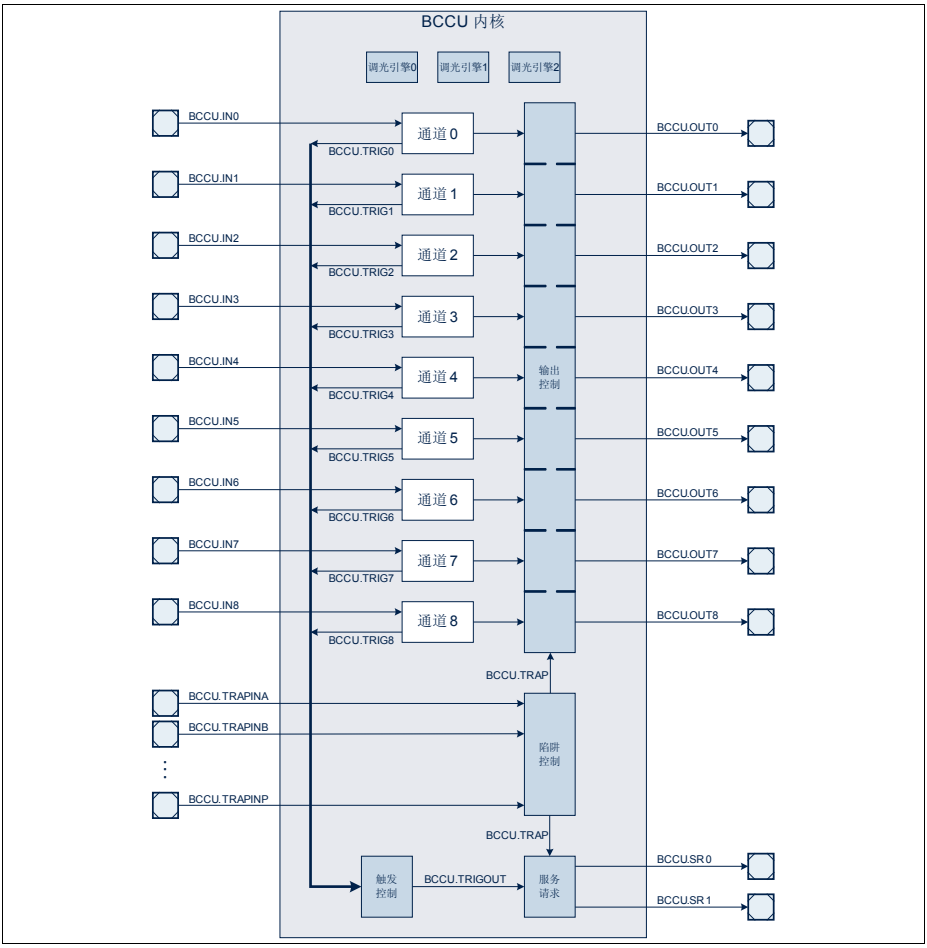


图 22-1　　BCCU 内核框图

每个调光引擎产生一个 12 位的调光值，该输出可被连接到任何一个通道的输入。还可以选择将一个可手动控制的 12 位 “ 全局调光值 ”(BCCU_**GLOBDIM**.GLOBDIM) 连接到通道输入。

22.2.1 通道结构

每个活动通道有一个 12 位的强度值 (BCCU_**INTy (y=0-8)**.CHINT)。亮度值 （与输出位流的平均值相同）是调光值与强度值的乘积。

一般来说，在多通道方案中，强度值用于确定颜色，而调光值用于改变总体亮度。例如，在使用 RGB LED 的情况下，可以使用 3 个通道。每个通道的强度值决定颜色，共同的调光值 (例如来自一个调光引擎) 决定总体亮度。

12 位亮度值是 sigma-delta 调制器的输入，在调制器中被转换为具有相同均值的快速变化的单比特信号。可以用一个位打包器来选择性地降低该信号的平均切换速率，打包器产生一个具有固定接通时间或断开时间的信号。该信号仍然具有相同的均值。

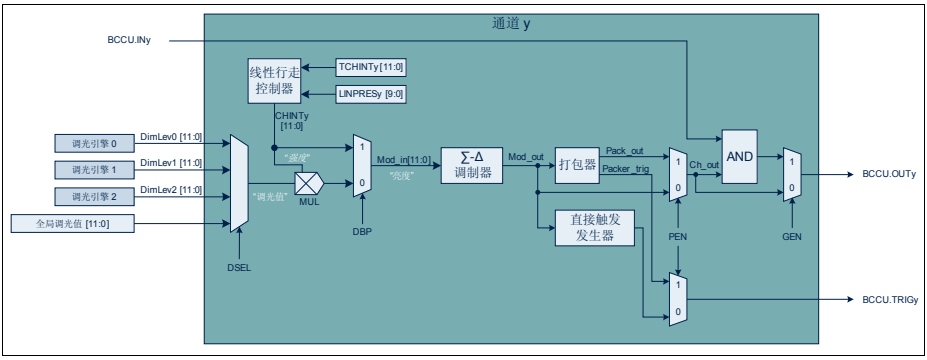


图 22-2 通道框图

BCCU.INy 是一个异步门控信号，在 BCCU_**CHCONFIGy (y=0-8)**.GEN 被置位后可以使用该信号。门控连接由组合逻辑电路实现，仅用于需要快速控制的场合，例如尖峰电流控制。

如果位打包器被禁止， ADC 的触发信号由触发信号发生器块中的 sigma-delta 调制器的输出直接产生。如果 BCCU_**CHCONFIGy (y=0-8)**.TRED 为 0，触发脉冲在上升沿 （从被动态转向主动态）产生；如果 BCCU_**CHCONFIGy (y=0-8)**.TRED 为 1，则触发脉冲在下降沿产生。如果 BCCU_**CHCONFIGy (y=0-8)**.ENFT 为 1，则使能强制触发信号产生。如果输出信号 (Mod_out) 长时间 (256 个位时间) 未改变状态，则产生强制触发信号。

22.2.2 指数调光

BCCU 包含 3 个调光引擎，这 3 个调光引擎可与全部 9 个通道连接。调光引擎能渐近改变亮度值，以使人眼能舒适地适应光强的变化。由于人眼对光的感受灵敏性呈指数规律，因此亮度的渐近改变是按照伪指数曲线实现的。用户必须设置目标调光值 (BCCU_**DLSz (z=0-2)**.TDLEV) 并启动映射传送 (通过置位 BCCU_**DESTRCON**.DEzS 实现)。调光值 (BCCU_**DLz (z=0-2)**.DIMLEV) 是时变信号。映射传送结束后，调光值会按照伪指数曲线逐渐达到目标调光值。达到目标值后， BCCU_**DESTRCON**.DEzS 被硬件清 0。

调光过程可以在任何时刻被中止，这可通过置位 BCCU_**DESTRCON**.DEzA 来实现。如果当前的调光过程尚未结束，建议在启动新的调光过程之前完成该操作。该操作还会将 BCCU_**DESTRCON**.DEzA 清零。

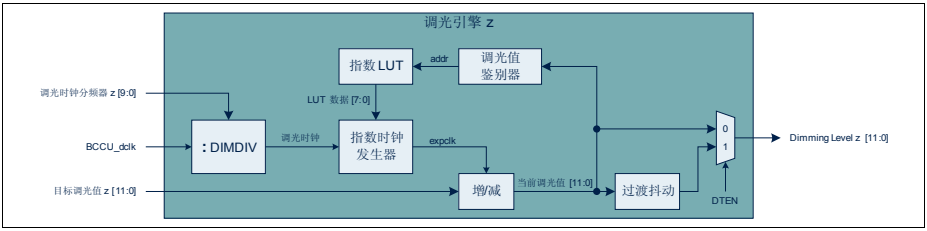


图 22-3　　调光引擎框图

当调光值大于 16 时，理想的指数调光曲线如下：

$$\text{调光值} = 2^{\frac{\text{调光时钟数} + 4096}{2048}} \tag{22.1}$$

实际的调光曲线是将分段直线组合在一起形成的伪指数曲线，以此来逼近理想指数曲线。

调光模块提供两种曲线供选择：一种是低精度的，另一种是高精度的。可以通过设置 BCCU_**DTTz (z=0-2)**.CSEL 来选择要使用的曲线。这两种曲线的区别在于用来逼近理想指数曲线的直线段的数量不同。

当调光值按低精度的曲线升高时，直线段的斜率在每经过 16, 32, 64, 128, 256, 512, 1024 或 2048 中的一个阈值点后加倍。

表 22-1　　低精度分段的伪指数曲线

调光阈值上限	直线段斜率	直线段的调光时钟数
16	1/256	4096
32	1/128	2048
64	1/64	2048
128	1/32	2048
256	1/16	2048
512	1/8	2048
1024	1/4	2048
2048	1/2	2048
4095	1	2047

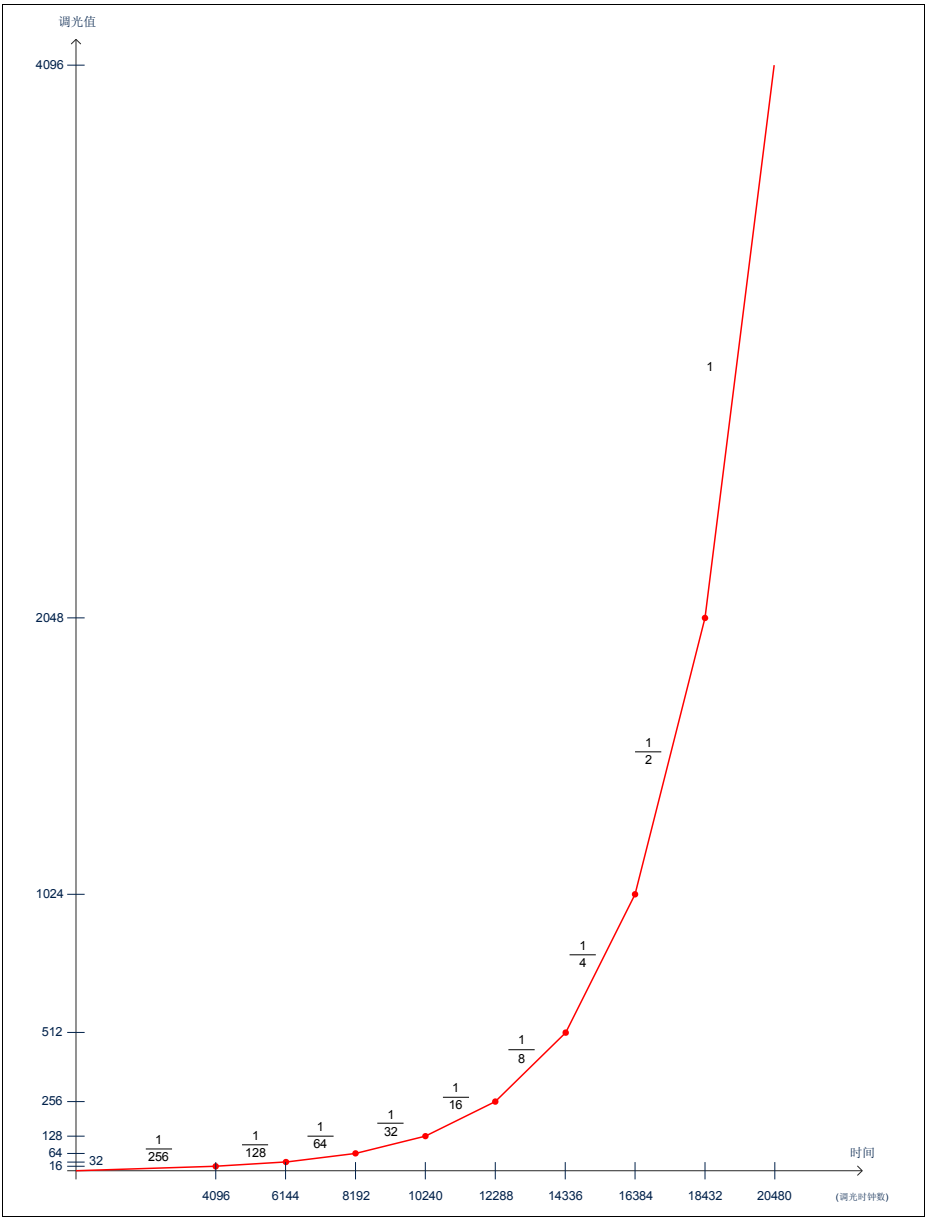


图 22-4　　低精度分段的伪指数曲线

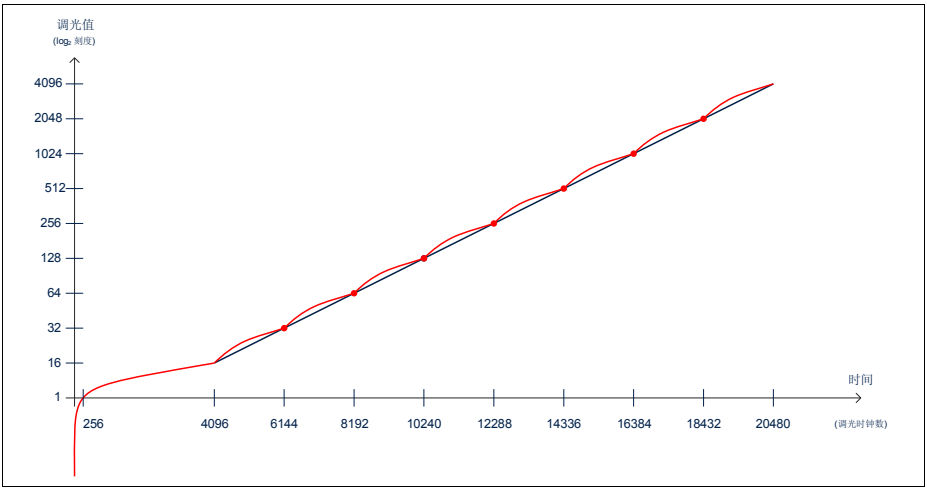


图 22-5　　低精度分段的伪指数曲线 —— 调光值按与人眼感觉 （实际曲线和 理想曲线）相同的对数刻度表示

该分段伪指数曲线被量化为 4095 级。当 BCCU_**DESTRCON**.DEzS 被置 1 时，调光引擎立即从第一级开始向目标值调节，并且在每个调光值都会等待一定数量的调光时钟周期 （“ 等待时间 ”），然后才进入下一级。该等待时间决定了直线段的斜率。一旦达到了目标值并经过相应数量的调光时钟后，BCCU_**DESTRCON**.DEzS 被硬件清 0，告知调光过程结束。从 0 调到 4095 需要 20479 个调光时钟 （未计算同步和总线通信延迟）。

表 22-2　　量化的低精度伪指数曲线

调光值	下一级之前的时钟数 (“ 等待时间 ”)	直线段斜率
0..16	256	1/256
17..32	128	1/128
33..64	64	1/64
65..128	32	1/32
129..256	16	1/16
257..512	8	1/8
513..1024	4	1/4
1025..2048	2	1/2
2049..	1	1

例如，当以 BCCU_**DESTRCON**.DEzS 置 1 和清 0 之间的时间（忽略延迟）计算调光时间时，按低精度曲线从 0 调到 18 需要 4352 个时钟 （16*256 + 2*128）。

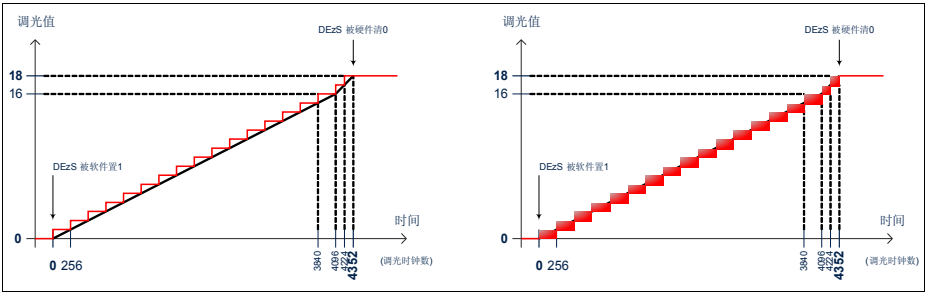


图 22-6　　抖动禁止和使能情况下按低精度曲线从 0 调到 18

不论调光的方向如何，指数时钟的产生是相同的。因此，在给定调光值，不管下一级是升高还是降低，下一级之前的等待时间是相同的。也正因为如此，在设置相同时下调比上调耗费的时间要长。这个额外的时间等于目标值的等待时间减去起始值的等待时间。从 4095 下调到 0 需要耗费 20734 个时钟（20479+256-1）。按低精度曲线从 18 下调到 0 需要耗费 4480 个时钟 （4352+256-128）。

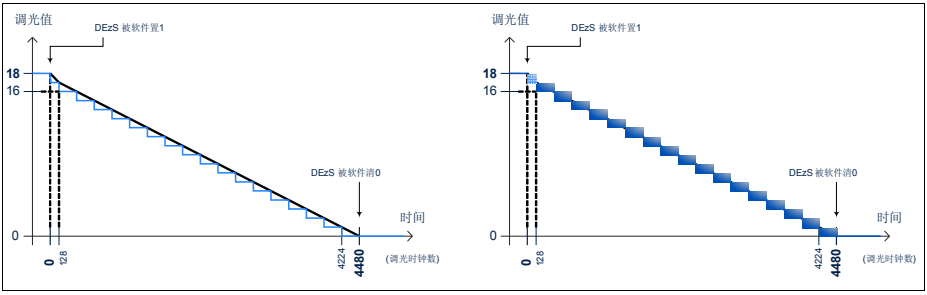


图 22-7　　抖动禁止和使能情况下按低精度曲线从 18 调到 0

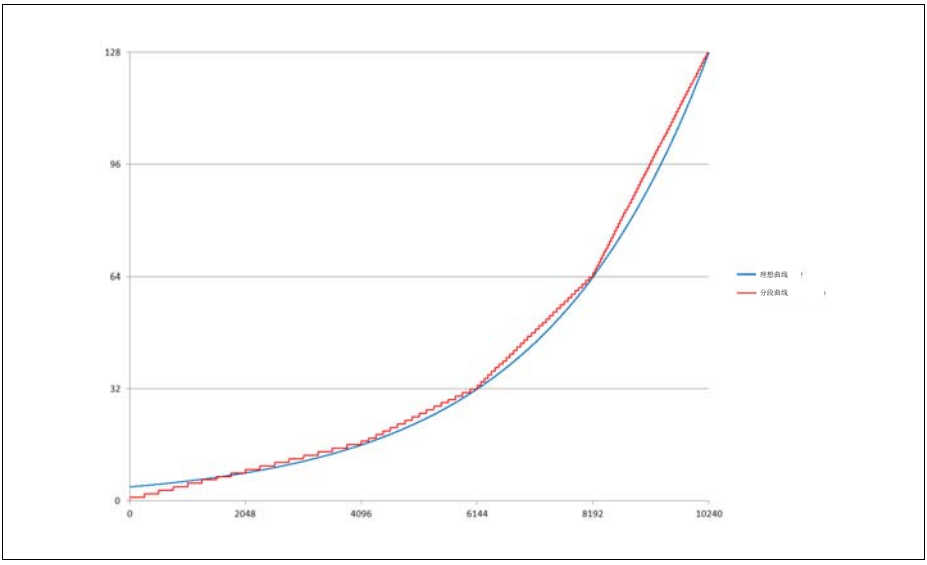


图 22-8　　低精度曲线的下游区域（实际曲线和理想曲线）

高精度曲线与低精度曲线类似，但高精度曲线有更多的线段和不同的直线斜率。

表 22-3　　高精度分段伪指数曲线

调光阈值上限	直线段斜率	直线段的调光时钟数
16	1/256	4096
18	1/196	392
25	1/132	924
34	1/102	918
41	1/81	567
48	1/64	448
69	1/51	1071
79	1/40	400
107	1/32	896
130	1/25	575
167	1/20	740
203	1/16	576
253	1/13	650

表 22-3 高精度分段伪指数曲线

调光阈值上限	直线段斜率	直线段的调光时钟数
343	1/10	900
512	1/7	1183
1024	1/4	2048
2048	1/2	2048
4095	1	2047

表 22-4 高精度分段伪指数曲线

调光值	下一级之前的时钟数 (“ 等待时间 ”)	直线段斜率
0..16	256	1/256
17..18	196	1/196
19..25	132	1/132
26..34	102	1/102
35..41	81	1/81
42..48	64	1/64
49..69	51	1/51
70..79	40	1/40
80..107	32	1/32
108..130	25	1/25
131..167	20	1/20
168..203	16	1/16
204..253	13	1/13
254..343	10	1/10
334..512	7	1/7
513..1024	4	1/4
1025..2048	2	1/2
2049..4095	1	1

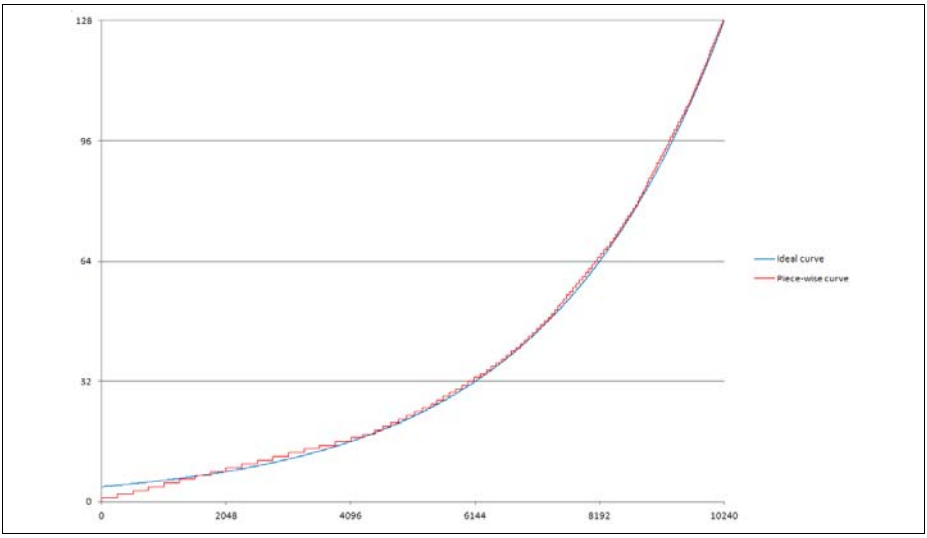


图 22-9　　高精度曲线的下游区域（实际曲线和理想曲线）

按完整的调光曲线上调时需要耗费 20479 个调光时钟，下调时则需要 20734 个时钟。从 0 到 4095 （0% 到 100%）的调光时间 （Fade Time）可通过 “ 调光时钟分频器 ” 位域（BCCU_**DTTz (z=0-2)**.DIMDIV）调整：

$$????_{0-100} = \frac{20479}{\left(\frac{f_{BCCUclk}}{DCLKPS \times DIMDIV}\right)} = \frac{20479}{\left(\frac{f_{BCCUdclk}}{DIMDIV}\right)} \quad (22.2)$$

$$????_{100-0} = \frac{20734}{\left(\frac{f_{BCCUclk}}{DCLKPS \times DIMDIV}\right)} = \frac{20734}{\left(\frac{f_{BCCUdclk}}{DIMDIV}\right)} \quad (22.3)$$

如果 BCCU_**DTTz (z=0-2)**.DIMDIV 为 0，调光值会在映射传送时立即变为与目标调光值相同，这意味着调光引擎实际上被旁路。

调光值低于 128 时，人眼对亮度变化非常敏感，可以通过对调光步长进行抖动处理来避免产生可见的亮度不连续。该功能可通过置位 BCCU_**DTTz (z=0-2)**.DTEN 来使能。仅在使用低精度曲线时方可使用抖动功能。每当调光值（BCCU_**DLz (z=0-2)**.DLEV）应该增 1 或减 1 （“ 调光步长 ”）时，会在其上叠加一个抖动序列，以使过渡过程在人眼看来较为平滑。调光值位于 0 和 15 之间时，抖动序列要持续 256 个调光时钟；调光值位于 16 和 31 之间时，抖动序列持续 128 个调光时钟；调光值位于 32 和 63 之间时，抖动模式持续 64 个调光时钟；调光值位于 64 和 127 之间时，抖动模式持续 32 个调光时钟。

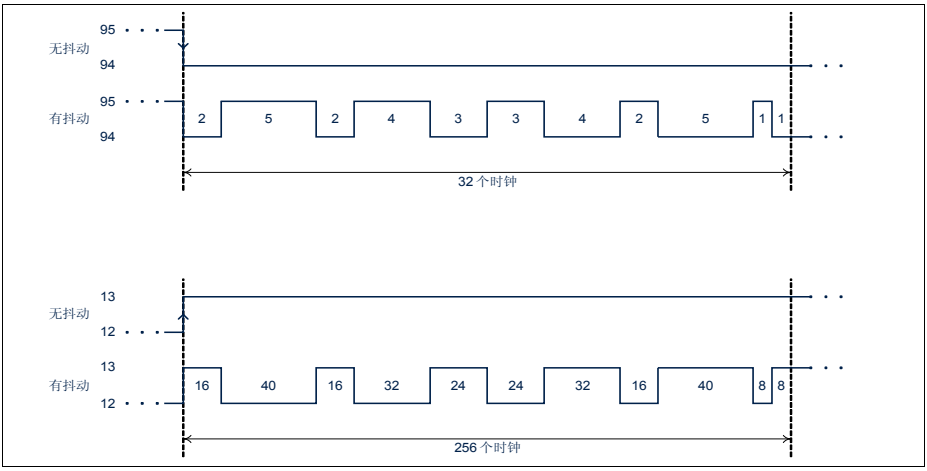


图 22-10　　抖动示例

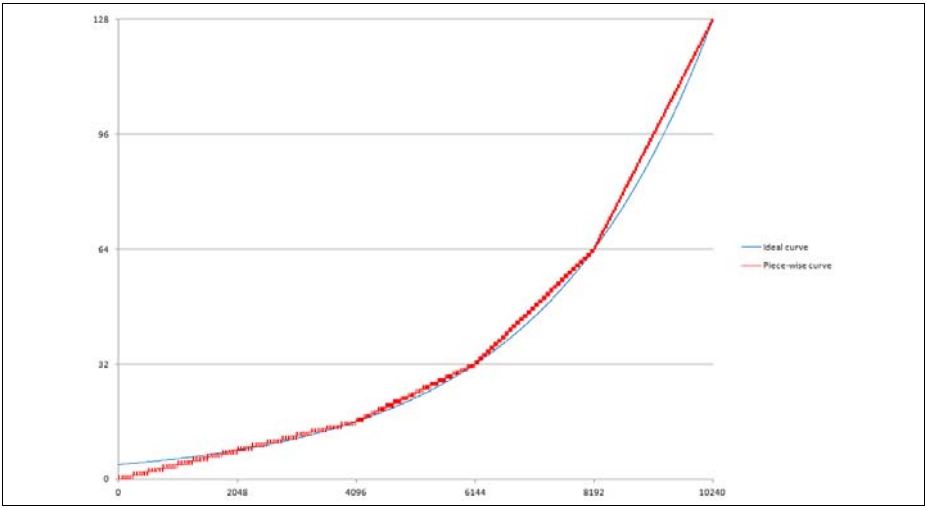


图 22-11　　抖动使能情况下低精度曲线的下游区域（实际曲线和理想曲线）

从调光值 0 到 1 过渡的抖动可能引起明显的闪烁。可通过使能受影响通道的闪烁看门狗来避免这种情况。将 **CHCONFIGy (y=0-8)**.WEN 位置 1 即可使能闪烁看门狗。

22.2.3 线性色彩行走方案

可使用线性行走方案改变强度值 （BCCU_**INTy (y=0-8)**.CHINT）。过渡时间，亦称线性行走时间，是用户可通过对 （BCCU_**CHCONFIGy (y=0-8)**.LINPRES）位域编程来调节的。在映射传送 (置位 BCCU_**CHSTRCON**.CHyS) 发生后，达到目标强度值 (BCCU_**INTSy (y=0-8)**.TCHINT) 所需要的过渡时间可用下面的公式计算：

$$?????? = \frac{LINPRES \times 2^{13}}{f_{BCCUfclk}} \quad (22.4)$$

一旦线性行走过程结束， BCCU_**CHSTRCON**.CHyS 即被硬件清 0。置位 BCCU_**CHSTRCON**.CHyA将中止线性行走过程，并将BCCU_**CHSTRCON**.CHyS清 0。

当线性行走过程正在进行时（BCCU_**CHSTRCON**.CHyS 被置 1），BCCU_**INTSy (y=0-8)**.TCHINT 不能被写入。在此期间，写操作被忽略。用户必须等待行走过程结束或通过置位 BCCU_**CHSTRCON**.CHyA 来手动中止行走过程。

如果 BCCU_**CHCONFIGy (y=0-8)**.LINPRES 为 0，则强度值与映射传送的目标强度相同，这意味着线性行走控制器 （Linear Walker）实际上被旁路。

通过将各通道的过渡时间设置为相同值并且同时启动映射传送，可获得平滑的色彩过渡。强度增量值是按通道映射传送时的通道强度和目标通道强度的差值计算的。

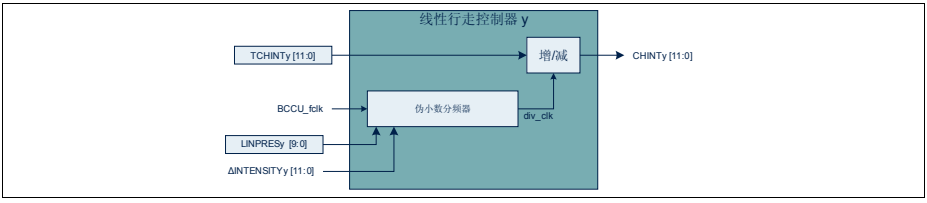


图 22-12 线性行走控制器框图

22.2.4 Sigma-Delta 调制器

Sigma-delta 调制器对缓慢变化的 12 位亮度信号进行过采样，并将其转换为高比特率的单比特信号。该单比特信号的平均模拟值与 12 位的亮度值呈正比。低通滤波功能是靠人眼实现的。

该调制器包含一个闪烁看门狗。如果该看门狗被使能，它会限制输出端出现的连续断开位（0）的个数以避免产生明显的闪烁。BCCU_**GLOBCON**.WDMBN 决定所允许的最大断开 （0）的个数 。如果连续断开位 （0）的个数达到了这个阈值，会在位流中插入一个接通位 （1）。

接通位的最低频率由下面的公式定义：

$$f_{min} = \frac{1}{WDMBN} \times f_{BCCUbclk} \quad (22.5)$$

22.2.5 打包器 （Packer）

打包器的主要用途是降低输出信号的平均切换频率，以减小外部开关电路的负载和改善 EMC 性能。打包器包含两个 8 位计数器、一个 4 级 FIFO 和一个输出发生器。

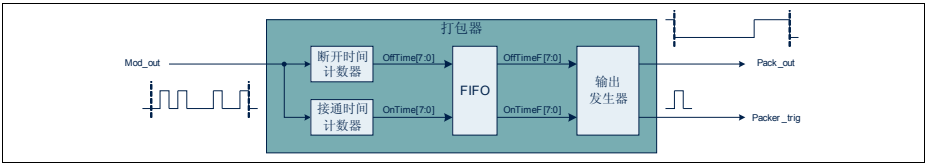


图 22-13 打包器框图

接通时间计数器 (BCU_**PKCNTRy (y=0-8)**.ONCNTVAL) 对输入位流 （Mod_out）的接通位计数，断开时间计数器 (BCU_**PKCNTRy (y=0-8)**.OFFCNTVAL) 对断开位计数。如果断开时间计数器或接通时间计数器的计数值达到了用户可调节的比较值（BCCU_**PKCMPy (y=0-8)**.ONCMP 或 BCCU_**PKCMPy (y=0-8)**.OFFCMP），这两个定时器都会复位并重新开始计数。在定时器复位之前，其计数值（BCCU_**PKCNTRy (y=0-8)**.ONCNTVAL 和 BCCU_**PKCNTRy (y=0-8)**.OFFCNTVAL）被加载到一个队列中（FIFO）。输出发生器从队列中交替地逐一取出这两个计数值，并生成一个打包后的 断 - 通信号包 （Pack_out）。如果队列为空，则输出发生器在重新检查 FIFO 之前会产生持续一个位时钟的断开位 (0) 。在发生微控制器复位或所用通道被禁止后又重新被使能时，所有队列级都为空。当打包器被使能时 （BCCU_**CHCONFIGy (y=0-8)**.PEN），输出发生器仅在有足够的级被填充后 （由 BCCU_**CHCONFIGy (y=0-8)**.PKTH 决定）才开始从队列中取计数值。在此之前，仅在输出端产生断开位。

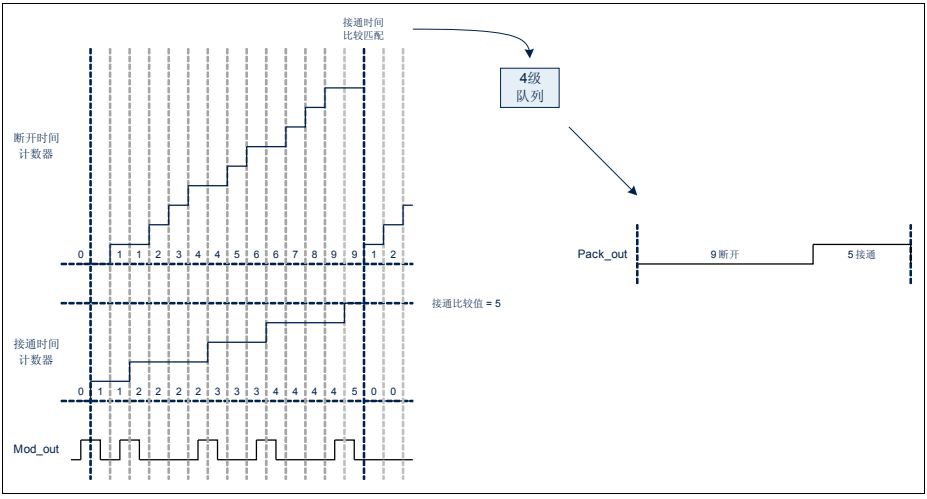


图 22-14 信号打包 (示例)

在通道未被使能的情况下（BCCU_**CHEN**.ECHy 为 0），可以手动更新 BCCU_**PKCNTRy (y=0-8)**.ONCNTVAL 和 BCCU_**PKCNTRy (y=0-8)**.OFFCNTVAL。通过向计数器中装入一个非零的起始值，用户可以在不同通道之间插入“相移”，以避免完全相同的行为。

如果亮度值足够高，以致于达到接通时间比较值，则经过打包的信号有固定的接通时间。在某一亮度值以下，断开时间比较值会在接通时间比较值之前达到。在这种情况下，经过打包的信号有固定的断开时间。打包后信号包的长度由比较值决定，可用下面的公式计算：

$$???_{min} = ONCMP \times \frac{1}{f_{BCCUbclk}}$$

$$???_{max} = (ONCMP + OFFCMP) \times \frac{1}{f_{BCCUbclk}} \quad (22.6)$$

22.2.6 全局触发控制

BCCU 产生两个 ADC 触发信号（BCCU_TRIGOUT0 和 BCCU_TRIGOU1），以同步启动转换。在某些情况下，ADC 采样值仅在相应的控制信号有效时才有意义（例如，相应的 LED 串为接通）。BCCU 通道在最后一个触发信号发生时的输出状态可通过检查“最后触发通道的输出电平寄存器”(BCCU_**LTCHOL**) 来观测。各通道可以在“通道使能寄存器”（BCCU_**CHEN**.ECHy）中被单独使能。每个通道都可以在“通道触发寄存器”（BCCU_**CHTRIG**.ETy）中被单独触发。只有被使能且触发功能也被使能的通道才能加入这种触发机制。在通道输出从无效变为有效或从有效变为无效时发生触发，可以为每个通道单独选择触发边沿 (BCCU_**CHCONFIGy (y=0-8)**.TRED)。

BCCU 提供两种触发方式（BCCU_**GLOBCON**.TM）。在方式 0，任一通道的触发信号都会引起一次 ADC 触发，并且可以产生两个触发信号输出。发生这种情况时，ADC 对每个通道采样，用户软件必须根据 BCCU_**LTCHOL** 的值来决定哪些结果是有效的。

在方式 1，某一时刻只能有一个通道触发信号产生 ADC 触发信号。各通道以循环方式（从 0 开始向上计数）依次产生触发信号。通过 BCCU_**CHTRIG**.TOSy 可以为每个通道单独选择在 BCCU_TRIGOUT0 或在 BCCU_TRIGOUT1 产生触发信号。一个触发信号产生后，通道指针会指向下一个参与的通道，以产生下一个触发信号。每个触发信号只能引起一次 ADC 测量。可能需要额外的用户软件来控制在一给定时刻去触发哪一个 ADC 通道。BCCU_**GLOBCON**.LTRS 指示最后一次触发信号是由哪一个 BCCU 通道产生的。如果所有通道触发使能位（BCCU_**CHTRIG**.ETy）都为0，通道指针被复位并指向通道0。

可以选择将触发信号延迟 1/4 或 1/2 位时间，以避免在有开关噪声期间采样。可通过 BCCU_**GLOBCON**.TRDEL 位域实现这种选择。仅在以正常方式（BCCU_**GLOBCLK**.BCS 为 0）产生 BCCU_bclk 时，该延迟才会起作用。

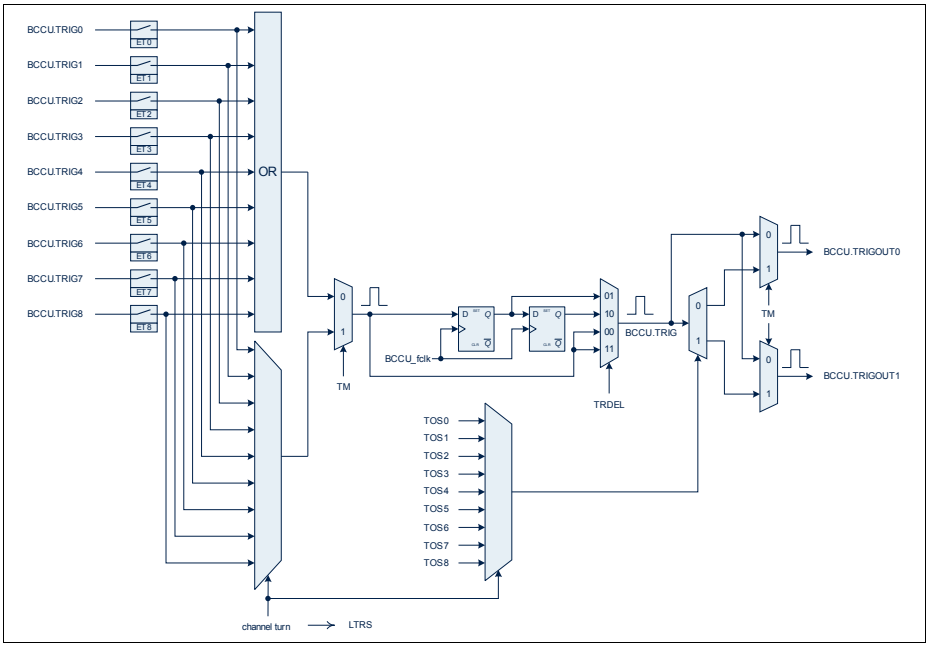


图 22-15　　BCCU 触发信号产生

22.2.7　陷阱 （Trap ）控制

陷阱 （trap）功能允许 BCCU 的输出响应输入引脚的状态。该功能可用于在陷阱输入变为有效 （即发生一次陷阱）时断开所连接的电源器件。陷阱是全局事件，但每个通道的陷阱功能可以通过设置 BCCU_**CHOCON**.CHxTPE 位单独使能。如果对应的位没有被置 1，则指定的通道将忽略陷阱事件。

如果检测到 BCCU.TRAPL 信号有一个上升沿或一个下降沿（用户可通过 BCCU_**GLOBCON**.TRAPED 选择），则发生一次陷阱事件。当发生一次陷阱事件时，受影响通道的输出立即[1)]进入其对应的被动电平（该被动电平由 BCCU_**CHOCON**.CHyOP 决定），事件标志 （BCCU_**EVFR**.TPF）和陷阱状态标志 （BCCU_**EVFR**.TPSF）位被置 1。

可以通过向 BCCU_**EVFCR**.TPFC 写 1 来复位 BCCU_**EVFR**.TPF，该操作对通道输出没有影响。可以通过向 BCCU_**EVFCR**.TPC 写 1 来复位 BCCU_**EVFR**.TPSF。这将清除陷阱，通道输出会根据通道的状态返回到其相应的输出电平。用户可以通过位 BCCU_**EVFR**.TPINL 来监视 BCCU.TRAPL 的电平，可能只是希望在陷阱信号变为无效后退出陷阱状态。

1) 经过 4 个 BCCU_clk 周期的同步延迟之后

也可以通过向 BCCU_**EVFSR**.TPS 写 1 来置位 BCCU_**EVFR**.TPSF，以产生一次软件陷阱。

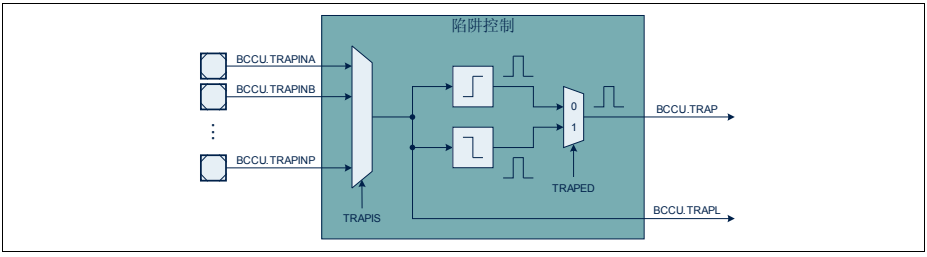


图 22-16　　陷阱控制

22.3　　电源、复位和时钟

BCCU　　位于内核电源域。系统复位可将该模块，包括除了位域 BCCU_**GLOBCON**.SUSCFG 以外的所有寄存器，都复位到默认状态。只有调试复位能将位域 BCCU_**GLOBCON**.SUSCFG 复位到其默认值。

*注：　在 XMC1300 中，由于没有独立的调试复位，位域 BCCU_**GLOBCON**.SUSCFG 需要通过系统复位来复位到其默认状态。*

BCCU 内核时钟 (BCCU_clk) 来自外设总线频率 (PCLK)，对 BCCU 内核的访问也使用该时钟。由 BCCU 内核时钟产生出三个主时钟供该模块的不同部分使用：

- BCCU_bclk (位时钟) 是决定 sigma-delta 转换器的采样频率的时钟，也是该转换器产生信号位时间的时钟。根据 BCCU_**GLOBCLK**.BCS 的值不同， BCCU_bclk 可以从 BCCU_fclk 经过一个 4 分频器产生，也可以与 BCCU_fclk 相同。
- BCCU_fclk (快速时钟)由触发控制电路使用，用于产生触发延时。该时钟由BCCU_clk 经过一个预分频器产生，其频率可通过 BCCU_**GLOBCLK**.FCLK_PS 设置。
- BCCU_dclk (调光引擎时钟) 是调光引擎的输入时钟。该时钟由 BCCU_clk 经过一个预分频器产生，其频率可通过 BCCU_**GLOBCLK**.DCLK_PS 设置。

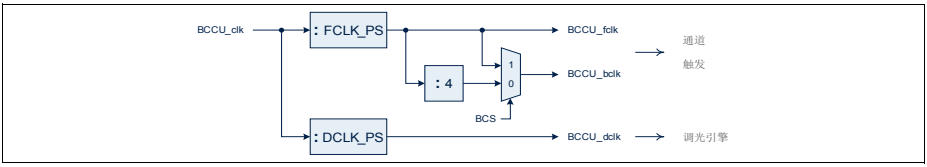


图 22-17　　BCCU 时钟

正常方式下的时钟频率可用下面的公式计算：

$$f_{BCCUfclk} = \frac{1}{FCLKPS} \times f_{BCCUclk} \quad (22.7)$$

$$f_{BCCUbclk} = \frac{1}{FCLKPS \times 4} \times f_{BCCUclk} \quad (22.8)$$

$$f_{BCCUdclk} = \frac{1}{DCLKPS} \times f_{BCCUclk} \quad (22.9)$$

BCCU 内核只能在活动模式下工作。

22.4 服务请求的产生

BCCU 有一个服务请求节点和 5 个中断。这 5 个中断是：1）触发 0 事件，2）触发 1 事件， 3）打包器 FIFO 满事件， 4）打包器 FIFO 空事件和 5）陷阱事件。

如果在 BCCU.TRIGOUT0/1 上产生了一个给 ADC 的触发脉冲，则发生触发 0/1 事件。如果任何一个打包器 FIFO 为满 / 空，则发生一个打包器 FIFO 满 / 空事件。如果在 BCCU.TRAPL 信号上检测到上升沿或下降沿，则发生一个陷阱事件。

这 5 个中断可以由硬件事件产生，也可以由软件通过将 BCCU_**EVFSR** 中的相应位置 1 产生。所有这 5 个中断都被分配到节点 0，在 BCCU_**EVIER** 中有相应的中断使能位，在 BCCU_**EVFR** 中有相应的标志位。标志位可由硬件事件置 1，也可通过置位 BCCU_**EVFSR**中相应的置1位使其置1。标志位由BCCU_**EVFCR**中相应的清除位清0。

另外，触发事件（BCCU.TRIGOUT0 和 BCCU_TRIGOUT1）被连接到 ADC 的触发输入。

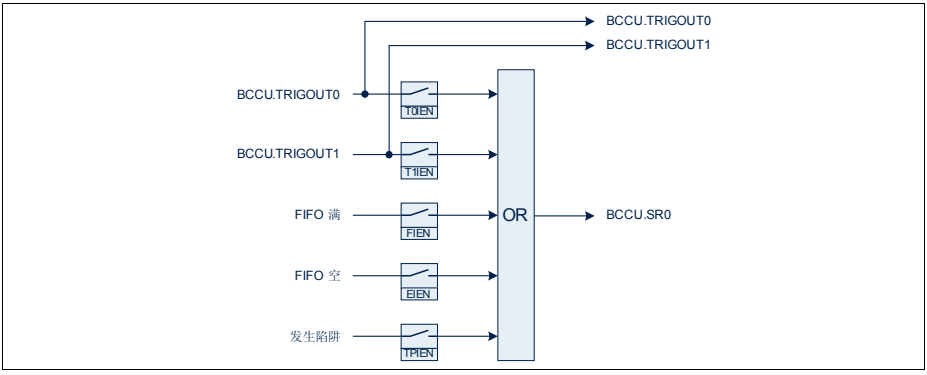


图 22-18 服务请求节点

BCCU 模块的时钟在默认状态下是被禁止的，可以通过 SCU_CGATCLR0 寄存器将其使能。 使能和禁止该模块的时钟可能导致负载发生改变，并可能发生如 SCU 一章的 CCU

（时钟门控控制）部分所描述的时钟消隐。强烈建议在用户初始化代码中设置该模块的时钟，以避免在运行时发生时钟消隐。

22.5 调试行为

在挂起 （suspend）方式，所有通道、调光引擎以及其他功能块的时钟都被停止，寄存器仍可被 CPU 以只读方式访问。该方式对调试非常有用，因为当前的器件状态被冻结，可以获得内部值的快照。由于时钟被停止，所有计数器和定时器都停止计数并被冻结。

通过改变 BCCU_**GLOBCON**.SUSCFG 位域的值可以配置进入挂起方式的模式。

当调试挂起被撤销时，内核根据最近的 SFR 设置恢复运行。

*注： 在 XMC1300 中，任何复位都会将位域 BCCU_**GLOBCON**.SUSCFG 复位到其默认值 。如果在调试期间需要挂起功能，建议在用户初始化代码中使能该功能。在对位域 BCCU_**GLOBCON**.SUSCFG 编程之前，必须首先使能模块的时钟。使能模块时钟时应特别谨慎，详见 SCU 一章中对 CCU （时钟门控控制）的描述。*

22.6 初始化

对 BCCU 寄存器编程时，建议遵循特定的序列。

1. 对全局寄存器编程

1. BCCU_**CHOCON**，设置输出的被动和主动电平
2. BCCU_**GLOBCLK**，设置时钟
3. BCCU_**GLOBCON**，综合配置
4. BCCU_**EVIER** ，设置中断
5. BCCU_**GLOBDIM**，在不使用调光引擎时设置用户定义的调光值
6. BCCU_**CHTRIG** ，设置 ADC 触发

2. 对通道寄存器编程

1. BCCU_**CHCONFIGy (y=0-8)**，一般通道配置
2. BCCU_**PKCMPy (y=0-8)** ，是否使用相应的打包器
3. BCCU_**PKCNTRy (y=0-8)**，在通道之间引入相移

3. 对调光引擎寄存器编程

1. BCCU_**DTTz (z=0-2)**，调整调光曲线

4. 使能通道和调光引擎

1. BCCU_**CHEN** ，使能通道
2. BCCU_**DEEN** ，使能调光引擎

5. 设置目标强度和启动线性行走

1. BCCU_**CHCONFIGy (y=0-8)**.LINPRES，设置线性行走时间
2. BCCU_**INTSy (y=0-8)**，设置目标强度
3. BCCU_**CHSTRCON** ，启动线性行走
4. 当线性行走过程结束时， BCCU_**CHSTRCON**.CHyS 被硬件清 0。

6. 设置目标调光值和启动调光过程 （如果使用了至少一个调光引擎）

1. BCCU_**DTTz (z=0-2)**.DIMDIV，设置调光速率
2. BCCU_**DLSz (z=0-2)**，设置目标调光值
3. BCCU_**DESTRCON**，启动调光过程
4. 当调光过程结束时， BCCU_**DESTRCON**.DEzS 被硬件清 0

22.7 数 / 模转换器

BCCU 还可以作为简单的多通道 12 位分辨率数 / 模转换器 (DAC) 使用。BCCU.OUTy 信号可以控制一些端口引脚的拉器件。外加一个电容即可实现一个 RC 滤波器。

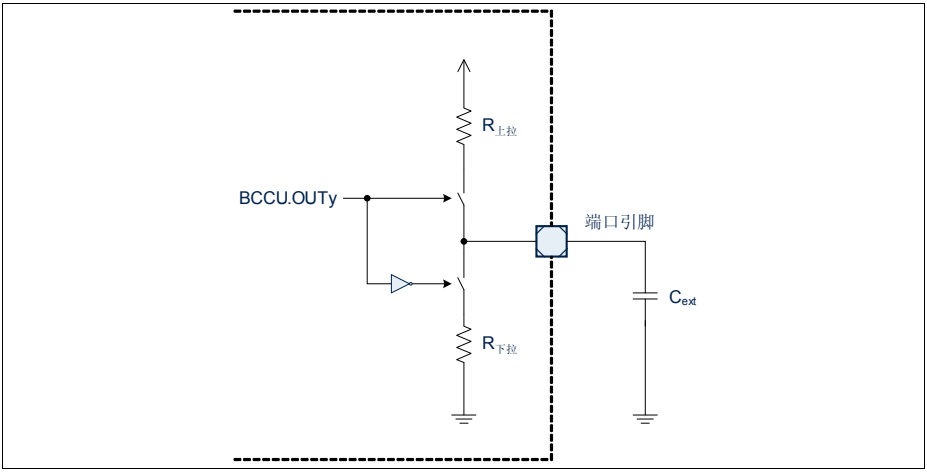


图 22-19 一个引脚上的简单 DAC

要使用 BCCU 的 DAC 功能，用户必须保证为所用端口引脚选择了 HW0 控制通路。这可以通过将所选的 Pn_HWSEL.HWx 位域设置为 01_B 来完成。这些端口引脚会被自动配置为输入引脚，而且仍可被其他模块使用。为内建模拟比较器产生一个模拟电压基准就是这样的一个例子。

建议选择一个高的位时钟频率，这可通过将快速时钟预分频器 （BCCU_**GLOBCLK**.FCLK_PS）设置为一个小的数值并且选择快速模式 （BCCU_**GLOBCLK**.BCS 为 1）来实现。所选用的 BCCU 通道应将调光引擎旁路 （BCCU_**CHCONFIGy (y=0-8)**.DBP 为 1），闪烁看门狗应被禁止 （BCCU_**CHCONFIGy (y=0-8)**.WEN 为 0）。

产生的电压电平由 12 位的通道强度值 （BCCU_**INTy (y=0-8)**.CHINT）控制，可通过更新目标通道强度 （BCCU_**INTSy (y=0-8)**.TCHINT）并启动一次映射传送 （将 BCCU_**CHSTRCON**.CHyS 置 1）来实现。

22.8 寄存器

所有 BCCU 寄存器（位域 BCCU_**GLOBCON**.SUSCFG 除外）都由系统复位来复位。调试复位将位域 BCCU_**GLOBCON**.SUSCFG 复位。

寄存器概述

绝对寄存器地址通过下面的加法计算：

模块基地址 + 偏移地址

表 22-5 寄存器基地址

模块	基地址	结束地址	备注
BCCU	5003 0000$_H$	5003 FFFF$_H$	

表 22-6 寄存器一览表

简称	描述	偏移地址	访问方式		描述见
			读	写	
BCCU 全局寄存器					
GLOBCON	全局控制寄存器	0000$_H$	U, PV, 32	U, PV, 32	页 22-22
GLOBCLK	全局时钟寄存器	0004$_H$	U, PV, 32	U, PV, 32	页 22-25
ID	模块标识寄存器	0008$_H$	U, PV, 32	BE	页 22-26
CHEN	通道使能寄存器	000C$_H$	U, PV, 32	U, PV, 32	页 22-27
CHOCON	通道输出控制寄存器	0010$_H$	U, PV, 32	U, PV, 32	页 22-28
CHTRIG	通道触发寄存器	0014$_H$	U, PV, 32	U, PV, 32	页 22-29
CHSTRCON	通道映射传送控制寄存器	0018$_H$	U, PV, 32	U, PV, 32	页 22-30
LTCHOL	最后触发通道输出电平寄存器	001C$_H$	U, PV, 32	BE	页 22-31

表 22-6 寄存器一览表

简称	描述	偏移地址	访问方式		描述见
			读	写	
DEEN	调光引擎使能寄存器	0020_H	U, PV, 32	U, PV, 32	页 22-32
DESTRCON	调光引擎映射传送控制寄存器	0024_H	U, PV, 32	U, PV, 32	页 22-33
GLOBDIM	全局调光值寄存器	0028_H	U, PV, 32	U, PV, 32	页 22-34
EVIER	事件中断使能寄存器	$002C_H$	U, PV, 32	U, PV, 32	页 22-35
EVFR	事件标志寄存器	0030_H	U, PV, 32	BE	页 22-36
EVFSR	事件标志置位寄存器	0034_H	U, PV, 32	U, PV, 32	页 22-37
EVFCR	事件标志清除寄存器	0038_H	U, PV, 32	U, PV, 32	页 22-39
BCCU 通道寄存器					
INTSy	通道强度映射寄存器 y	$003C_H$+ y*0014_H	U, PV, 32	U, PV, 32	页 22-40
INTy	通道强度寄存器 y	0040_H+ y*0014_H	U, PV, 32	BE	页 22-40
CHCONFIGy	通道配置寄存器 y	0044_H+ y*0014_H	U, PV, 32	U, PV, 32	页 22-41
PKCMPy	打包器比较寄存器 y	0048_H+ y*0014_H	U, PV, 32	U, PV, 32	页 22-43
PKCNTRy	打包器计数器寄存器 y	$004C_H$+ y*0014_H	U, PV, 32	U, PV, 32	页 22-44
Reserved	保留	$00F0_H$ - 0178_H	BE	BE	
BCCU 调光引擎寄存器					
DLSz	调光值映射寄存器 z	$017C_H$+ z*$000C_H$	U, PV, 32	U, PV, 32	页 22-45
DLz	调光值寄存器 z	0180_H+ z*$000C_H$	U, PV, 32	BE	页 22-45

表 22-6　寄存器一览表

简称	描述	偏移地址	访问方式		描述见
			读	写	
DTTz	调光过渡时间寄存器 z	0184_H+ z*$000C_H$	U, PV, 32	U, PV, 32	页 22-46
Reserved	保留	$01A0_H$ - $FFFC_H$	BE	BE	

22.8.1　全局寄存器

GLOBCON

全局控制　　**(0000_H)**　　**复位值 : $0320\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				WDMBN											
r				rw											

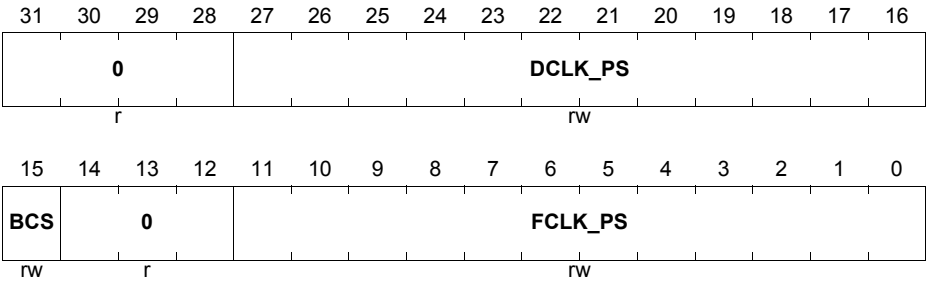
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LTRS				0	TRAPED	TRAPIS				SUSCFG		TRDEL		0	TM
rh				r	rw	rw				rw		rw		r	rw

域	位	类型	描述
TM	0	rw	**触发方式** 0_B　方式 0：如果有任何通道产生触发事件，BCCU 触发就会发生 (或逻辑) 1_B　方式 1：如果在活动通道出现一个通道触发事件，则发生 BCCU 触发。当 BCCU 触发发生时，下一个触发功能被使能的通道会变为活动状态，下一通道的选择遵循循环规则。
0	1	r	**保留** 读出值为 0；应写入 0。

域	位	类型	描述
TRDEL	[3:2]	rw	**触发延迟** 00_B 无延迟 01_B 在引起 BCCU 触发的通道触发事件之后四分之一位时间， BCCU 触发发生；只能在 BCCU_GLOBCLK.BCS 为 0 时使用 10_B 在引起 BCCU 触发的通道触发事件之后二分之一位时间， BCCU 触发发生；只能在 BCCU_GLOBCLK.BCS 为 0 时使用 BCCU 11_B 无延迟
SUSCFG	[5:4]	rw	**挂起方式配置** 该位域决定 BCCU 通道如何进入挂起方式。 00_B 挂起请求被忽略，模块不能被挂起 01_B 所有通道立即停止运行并冻结在最后状态，没有任何安全停止动作 10_B 所有通道立即停止运行并冻结在最后状态；所有输出进入被动态，以实现安全停止 11_B 保留
TRAPIS	[9:6]	rw	**陷阱输入引脚选择** 陷阱输入源 (BCCU.TRAPL) 0000_B BCCU.TRAPINA 0001_B BCCU.TRAPINB 0010_B BCCU.TRAPINC 0011_B BCCU.TRAPIND 0100_B BCCU.TRAPINE 0101_B BCCU.TRAPINF 0110_B BCCU.TRAPING 0111_B BCCU.TRAPINH 1000_B BCCU.TRAPINI 1001_B BCCU.TRAPING 1010_B BCCU.TRAPINK 1011_B BCCU.TRAPINL 1100_B BCCU.TRAPINM 1101_B BCCU.TRAPINN 1110_B BCCU.TRAPINO 1111_B BCCU.TRAPINP
TRAPED	10	rw	**陷阱边沿** 0_B 陷阱在 BCCU.TRAPL 信号的上升沿发生 （陷阱标志置位） 1_B 陷阱在 BCCU.TRAPL 信号的下降沿发生 （陷阱标志置位）

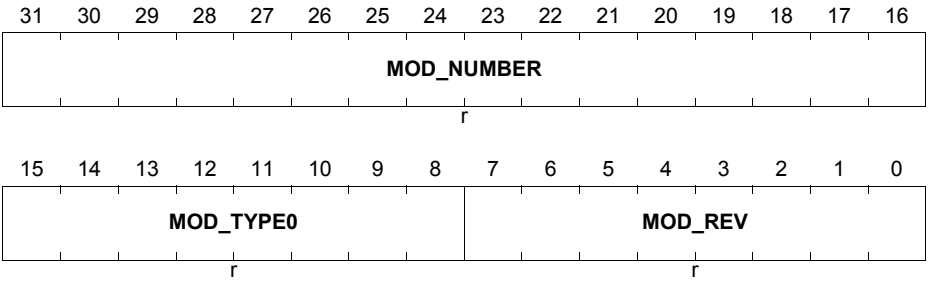
域	位	类型	描述
0	11	r	**保留** 读出值为 0；应写入 0。
LTRS	[15:12]	rh	**最后的触发源** 指示最后一次触发是由哪个通道产生的（在触发方式 0，该值总是为 0） 0000_B 最后一次触发发生在通道顺序 0 0001_B 最后一次触发发生在通道顺序 1 0010_B 最后一次触发发生在通道顺序 2 0011_B 最后一次触发发生在通道顺序 3 0100_B 最后一次触发发生在通道顺序 4 0101_B 最后一次触发发生在通道顺序 5 0110_B 最后一次触发发生在通道顺序 6 0111_B 最后一次触发发生在通道顺序 7 1000_B 最后一次触发发生在通道顺序 8
WDMBN	[27:16]	rw	**看门狗最大位数** 在 sigma-delta 调制器输出端所允许的连续 0 的最大位数
0	[31:28]	r	**保留** 读出值为 0；应写入 0。

GLOBCLK
全局时钟 **(0004$_H$)** **复位值 : 00DB 0190$_H$**



域	位	类型	描述
FCLK_PS	[11:0]	rw	**快速时钟预分频器因子** BCCU 的恒定时钟输入被按该值进行预分频，以产生 BCCU_fclk。BCCU_bclk 由 BCCU_fclk 经过 4 分频产生。 BCCU_bclk 决定 sigma-delta 调制器输出的位时间。 0_D 无时钟 1_D 1 分频 ... 4095_D 4095 分频
BCS	15	rw	**位时钟选择位** 0_B 正常方式：BCCU_bclk 由 BCCU_fclk 经过 4 分频产生。 1_B 快速方式：BCCU_bclk 与 BCCU_fclk 相同。
0	[14:12]	r	**保留** 读出值为 0；应写入 0。
DCLK_PS	[27:16]	rw	**调光时钟预分频器因子** BCCU 的恒定时钟输入被按该值进行预分频，以产生调光引擎的时钟输入 BCCU_dclk。 0_D 无时钟 1_D 1 分频 ... 4095_D 4095 分频
0	[31:28]	r	**保留** 读出值为 0；应写入 0。

ID
模块标识 **(0008_H)** **复位值 : 00F3 C0XX$_H$**



域	位	类型	描述
MOD_REV	[7:0]	r	**模块版本号** MOD_REV 定义版本号。模块的版本值从 01_H (第一个版本) 开始。
MOD_TYPE0	[15:8]	r	**模块类型** 该位域为 $C0_H$。它定义该模块为 32 位模块。
MOD_NUMBER	[31:16]	r	**模块编号值** 该位域定义模块的标识号。

CHEN
通道使能 **($000C_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							ECH 8	ECH 7	ECH 6	ECH 5	ECH 4	ECH 3	ECH 2	ECH 1	ECH 0
r							rw	rw	rw	rw	rw	rw	rw	rw	rw

域	位	类型	描述
ECHy (y=0-8)	y	rw	**通道 y 使能** 0_B 通道被禁止，输出电平为被动电平；线性行走控制器和 Sigma-Delta 调制器被复位，打包器 FIFO 被清空；当通道被禁止时，所有内部逻辑和 INTy 都被复位 1_B 通道被使能。
0	[31:9]	r	**保留** 读出值为 0；应写入 0。

CHOCON
通道输出控制 **(0010_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			0				**CH8 TPE**	**CH7 TPE**	**CH6 TPE**	**CH5 TPE**	**CH4 TPE**	**CH3 TPE**	**CH2 TPE**	**CH1 TPE**	**CH0 TPE**
			r				rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			0				**CH8 OP**	**CH7 OP**	**CH6 OP**	**CH5 OP**	**CH4 OP**	**CH3 OP**	**CH2 OP**	**CH1 OP**	**CH0 OP**
			r				rw	rw	rw	rw	rw	rw	rw	rw	rw

域	位	类型	描述
CHyOP (y=0-8)	y	rw	**通道 y 输出被动电平** 0_B 高电平有效 1_B 低电平有效
0	[15:9]	r	**保留** 读出值为 0；应写入 0。
CHyTPE (y=0-8)	y+16	rw	**通道 y 陷阱使能** 0_B 禁止通道的陷阱功能 1_B 使能通道的陷阱功能。当陷阱发生时，输出变为被动电平
0	[31:25]	r	**保留** 读出值为 0；应写入 0。

CHTRIG
通道触发 **(0014_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			0				**TOS 8**	**TOS 7**	**TOS 6**	**TOS 5**	**TOS 4**	**TOS 3**	**TOS 2**	**TOS 1**	**TOS 0**
			r				rw	rw	rw	rw	rw	rw	rw	rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			0				**ET8**	**ET7**	**ET6**	**ET5**	**ET4**	**ET3**	**ET2**	**ET1**	**ET0**
			r				rw	rw	rw	rw	rw	rw	rw	rw	rw

域	位	类型	描述
ETy (y=0-8)	y	rw	**通道 y 触发使能** 0_B 禁止通道触发 1_B 使能通道触发 如果所有的通道 y 触发使能位都为 0，则通道指针被复位并指向通道 0。
0	[15:9]	r	**保留** 读出值为 0；应写入 0。
TOSy (y=0-8)	y+16	rw	**通道 y 触发输出选择** 只能用于触发模式 1 (GLOBON.TM=1)，否则被忽略 0_B 通道触发脉冲会出现在 BCCU_TRIGOUT0 1_B 通道触发脉冲会出现在 BCCU_TRIGOUT1
0	[31:25]	r	**保留** 读出值为 0；应写入 0。

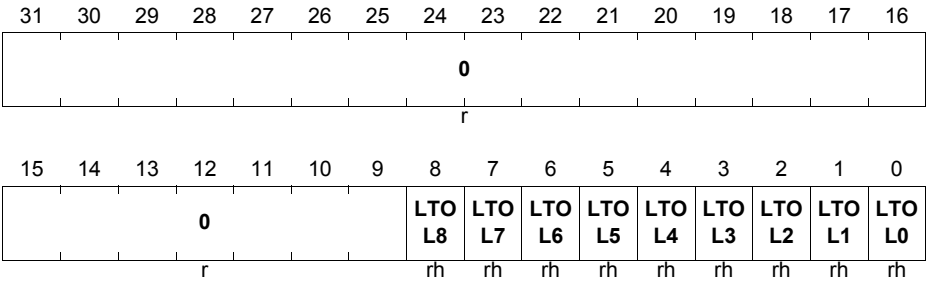
CHSTRCON
通道映射传送 **(0018_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
			0				**CH8A**	**CH7A**	**CH6A**	**CH5A**	**CH4A**	**CH3A**	**CH2A**	**CH1A**	**CH0A**
			r				w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
			0				**CH8S**	**CH7S**	**CH6S**	**CH5S**	**CH4S**	**CH3S**	**CH2S**	**CH1S**	**CH0S**
			r				rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
CHyS (y=0-8)	y	rwh	**通道 y 映射传送** 0_B 无作用 1_B 启动通道 y 的目标强度映射传送。线性行走将启动，通道 y 的强度开始向目标值变化。当线性行走过程结束并且达到目标值后，线性行走被硬件清除
0	[15:9]	r	**保留** 读出值为 0；应写入 0。
CHyA (y=0-8)	y+16	w	**通道 y 线性行走中止** 0_B 无作用 1_B 中止线性行走； CHyS 被清 0，通道 y 强度停止改变 读出值总是为 0
0	[31:25]	r	**保留** 读出值为 0；应写入 0。

LTCHOL
最后触发通道输出电平 **($001C_H$)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
LTOLy (y=0-8)	y	rh	**最后触发通道输出** 最后一次触发发生时通道的输出电平。在触发方式 0，每当 BCCU 触发发生时，输出位被更新。在触发方式 1，当相关通道顺序中发生触发时，输出位被更新。 0_B 被动 1_B 主动
0	[31:9]	r	**保留** 读出值为 0；应写入 0。

DEEN
调光引擎使能 **(0020$_H$)** **Reset Value: 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													EDE2	EDE1	EDE0
r													rw	rw	rw

域	位	类型	描述
EDEz (z=0-2)	z	rw	**调光引擎 z 使能** 0_B 调光引擎被禁止；当调光引擎被禁止时，输出调光值 （DLz.DLEV）被复位到 0 1_B 调光引擎被使能
0	[31:3]	r	**保留** 读出值为 0；应写入 0。

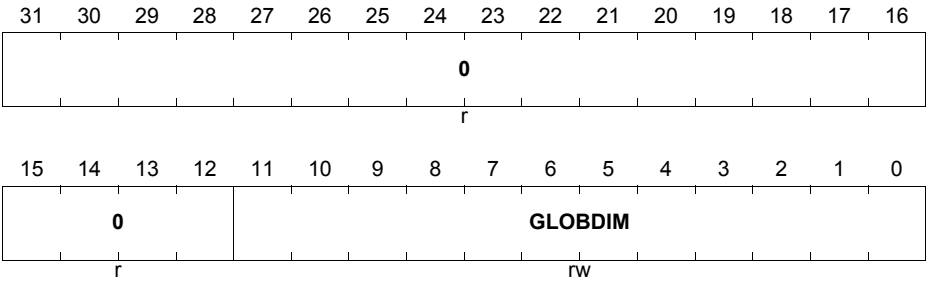
DESTRCON
调光映射传送 **(0024_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0													DE2A	DE1A	DE0A
r													w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													DE2S	DE1S	DE0S
r													rwh	rwh	rwh

域	位	类型	描述
DEzS (z=0-2)	z	rwh	**调光引擎 z 映射传送** 0_B 无作用 1_B 启动目标调光值映射传送。调光过程开始，调光值向目标值变化。当调光过程结束且达到目标值时，被硬件清 0。
0	[15:3]	r	**保留** 读出值为 0；应写入 0。
DEzA (z=0-2)	z+16	w	**调光引擎 z 调光中止** 0_B 无作用 1_B 中止调光过程； DEzS 被清 0，BCCU_DLz.DLEV 停止变化 读出值总是为 0
0	[31:19]	r	**保留** 读出值为 0；应写入 0。

GLOBDIM
全局调光值 **(0028$_H$)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
GLOBDIM	[11:0]	rw	**全局调光值** 可用于软件控制调光
0	[31:12]	r	**保留** 读出值为 0；应写入 0。

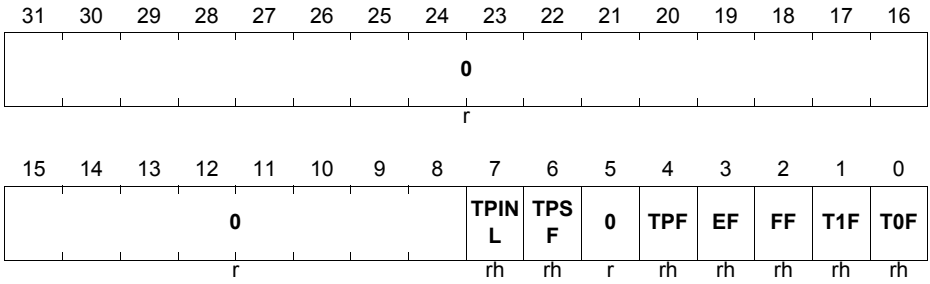
EVIER
事件中断使能 **($002C_H$)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											TPIEN	EIEN	FIEN	T1IEN	T0IEN
r											rw	rw	rw	rw	rw

域	位	类型	描述
T0IEN	0	rw	**触发信号 0 中断使能** 0_B 触发信号 0 中断产生被禁止 1_B BCCU 触发信号 0 （BCCU_TRIGOUT0）在 SR0 产生 中断
T1IEN	1	rw	**触发信号 1 中断使能** 0_B 触发信号 1 中断产生被禁止 1_B BCCU 触发信号 1 （BCCU_TRIGOUT1）在 SR0 产生中断
FIEN	2	rw	**FIFO 满中断使能** 0_B FIFO 满中断产生被禁止 1_B 当接通时间计数器或断开时间计数器试图进行一次写操作时，如果有任何一个打包器 FIFO 已满，则会在 SR0 产生中断
EIEN	3	rw	**FIFO 空中断使能** 0_B FIFO 空中断产生被禁止 1_B 当输出发生器试图进行一次读操作时，如果有任何一个打包器 FIFO 已空，则会在 SR0 产生中断
TPIEN	4	rw	**陷阱中断使能** 0_B 陷阱中断产生被禁止 1_B 如果陷阱发生，会在 SR0 产生中断
0	[31:5]	r	**保留** 读出值为 0 ；应写入 0。

EVFR
事件标志 **(0030_H)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
T0F	0	rh	**触发信号 0 标志** 0_B 在 BCCU 触发信号线 0（BCCU_TRIGOUT0） 未检测到触发事件 1_B 在 BCCU 触发信号线 0（BCCU_TRIGOUT0） 检测到触发事件
T1F	1	rh	**触发信号 1 标志** 0_B 在 BCCU 触发信号线 1（BCCU_TRIGOUT1） 未检测到触发事件 1_B 在 BCCU 触发信号线 1（BCCU_TRIGOUT1） 检测到触发事件
FF	2	rh	**FIFO 满标志** 0_B 未检测到 FIFO 满事件 1_B 检测到 FIFO 满事件，因为有一个打包器 FIFO 已满，并且接通时间计数器或断开时间计数器进行了一次写操作尝试
EF	3	rh	**FIFO 空标志** 0_B 未检测到 FIFO 空事件 1_B 检测到 FIFO 空事件，因为有一个打包器 FIFO 已空，并且输出发生器进行了一次读操作尝试
TPF	4	rh	**陷阱标志** 0_B 未检测到陷阱事件 1_B 检测到陷阱事件
0	5	r	**保留** 读出值为 0；应写入 0。

域	位	类型	描述
TPSF	6	rh	**陷阱状态标志** 0_B BCCU 不处于陷阱状态 1_B BCCU 处于陷阱状态，受影响的通道输出其被动电平
TPINL	7	rh	**陷阱输入电平** 0_B BCCU.TRAPL 的当前电平为低 1_B BCCU.TRAPL 的当前电平为高
0	[31:8]	r	**保留** 读出值为 0；应写入 0。

EVFSR
事件标志置位 **(0034$_H$)** **复位值：0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									**TPS**	**0**	**TPFS**	**EFS**	**FFS**	**T1FS**	**T0FS**
r									w	r	w	w	w	w	w

域	位	类型	描述
T0FS	0	w	**触发信号 0 标志置位** 0_B 无作用 1_B 置位 EVFR 中的触发信号 0 标志，如果在 EVIER 中被使能，则产生中断
T1FS	1	w	**触发信号 1 标志置位** 0_B 无作用 1_B 置位 EVFR 中的触发信号 1 标志，如果在 EVIER 中被使能，则产生中断
FFS	2	w	**FIFO 满标志置位** 0_B 无作用 1_B 置位 EVFR 中的 FIFO 满标志，如果在 EVIER 中被使能，则产生中断

域	位	类型	描述
EFS	3	w	**FIFO 空标志置位** 0_B 无作用 1_B 置位 EVFR 中的 FIFO 空标志，如果在 EVIER 中被使能，则产生中断
TPFS	4	w	**陷阱标志置位** 0_B 无作用 1_B 置位 EVFR 中的陷阱标志，如果 在 EVIER 中被使能，则产生中断，不会发生陷阱动作
0	5	r	**保留** 读出值为 0；应写入 0。
TPS	6	w	**陷阱置位** 0_B 无作用 1_B 置位 EVFR 中的陷阱状态标志和陷阱标志，会产生一次陷阱；如果 在 EVIER 中被使能，则产生中断
0	[31:7]	r	**保留** 读出值为 0；应写入 0。

EVFCR
事件标志清除 **(0038_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

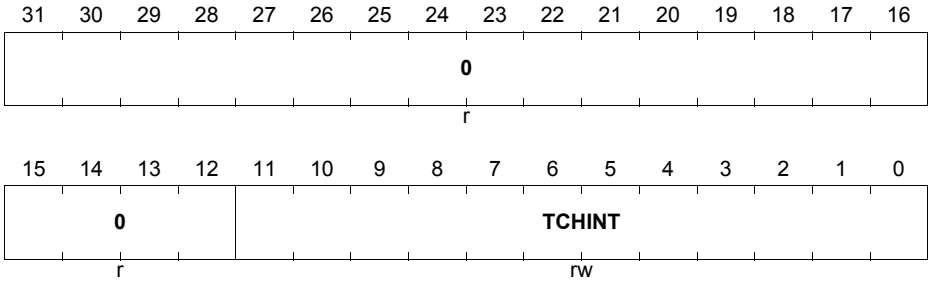
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									TPC	0	TPFC	EFC	FFC	T1FC	T0FC
r									w	r	w	w	w	w	w

域	位	类型	描述
T0FC	0	w	**触发信号 0 标志清除** 0_B 无作用 1_B 清除 EVFRC 中的触发信号 0 标志
T1FC	1	w	**触发信号 1 标志清除** 0_B 无作用 1_B 清除 EVFRC 中的触发信号 1 标志
FFC	2	w	**FIFO 满标志清除** 0_B 无作用 1_B 清除 EVFR 中的 FIFO 满标志
EFC	3	w	**FIFO 空标志清除** 0_B 无作用 1_B 清除 EVFR 中的 FIFO 空标志
TPFC	4	w	**陷阱标志清除** 0_B 无作用 1_B 清除 EVFR 中的陷阱标志
0	5	r	**保留** 读出值为 0；应写入 0。
TPC	6	w	**陷阱清除** 0_B 无作用 1_B 清除 EVFR 中的陷阱状态标志；退出陷阱状态，受影响的通道将返回其正常输出电平
0	[31:7]	r	**保留** 读出值为 0；应写入 0。

22.8.2 通道寄存器

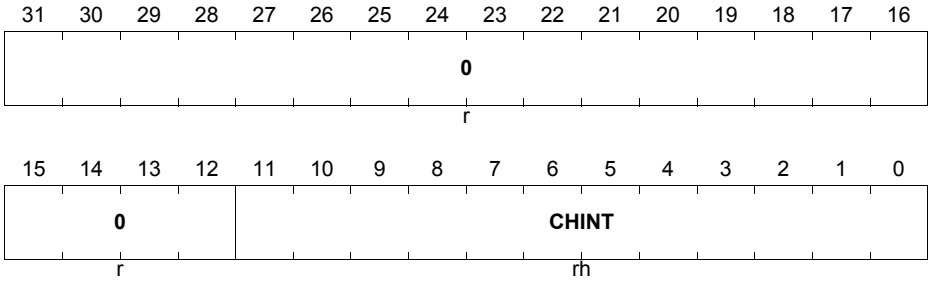
XMC1300 有 9 个通道。 y 表示通道 0 ~ 8。

INTSy (y=0-8)
通道强度映射 **($003C_H$ + y*0014_H)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
TCHINT	[11:0]	rw	**目标通道强度** 只能在 CHSTRCON.CHyS 不为 1 时对该位域写入，否则新写入值被忽略。
0	[31:12]	r	**保留** 读出值为 0；应写入 0。

INTy (y=0-8)
通道强度 **(0040_H + y*0014_H)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
CHINT	[11:0]	rh	**通道强度** 当前的通道强度
0	[31:12]	r	**保留** 读出值为 0；应写入 0。

CHCONFIGy (y=0-8)

通道配置 **(0044_H + y*0014_H)** **复位值 : 0000 0002_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						LINPRES									
r						rw									

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				ENFT	TRED	WEN	GEN	DBP	DSEL			PEN	PKTH		
r				rw	rw	rw	rw	rw	rw			rw	rw		

域	位	类型	描述
PKTH	[2:0]	rw	**打包器阈值** 被使能后，一旦已满队列级的数量达到该阈值，打包器就开始根据保存在 FIFO (队列) 中的值产生输出信号 (直到只产生 0 为止)。
PEN	3	rw	**打包器使能** 0_B 不使用打包器 1_B 接通时间计数器和断开时间计数器运行，打包后的输出位流随打包器触发信号而产生。

域	位	类型	描述
DSEL	[6:4]	rw	**调光选择** 调光输入源 000_B 调光引擎 0 001_B 调光引擎 1 010_B 调光引擎 2 011_B 保留 100_B 保留 101_B 保留 110_B 保留 111_B 全局调光值
DBP	7	rw	**调光输入旁路** 0_B 通道亮度是所选调光输入与通道强度的乘积 1_B 不使用调光输入，通道亮度仅由通道强度值决定
GEN	8	rw	**门控使能** 0_B 禁止门控功能，输入信号 （BCCU.INy）不起作用 1_B 使能门控功能，输出门控信号为 BCCU.INy
WEN	9	rw	**闪烁看门狗使能** 0_B 不使用闪烁看门狗 1_B 闪烁看门狗工作，根据 GLOBCON.WDMBN 限制 sigma-delta 调制器输出的连续 0 的个数
TRED	10	rw	**触发边沿** 0_B 在通道输出从被动电平向主动电平跳变时发生通道触发 1_B 在通道输出从主动电平向被动电平跳变时发生通道触发
ENFT	11	rw	**强制触发使能** 0_B 不产生强制触发 1_B 如果 sigma-delta 调制器的输出在连续 256 个位时间内没有改变状态，则触发信号发生器产生一个触发信号；仅在打包器被禁止时（PEN=0）起作用
0	[15:12]	r	**保留** 读出值为 0；应写入 0。

域	位	类型	描述
LINPRES	[25:16]	rw	**线性行走控制器时钟预分频器** 决定在映射传送发生后（CHSTRCON.CHyS 被置 1 之后）通道强度到达目标通道强度需要耗费的时间。对实现平滑的线性色彩过渡是必须的。如果该值为 0，则强度值会变为与映射传送时的目标强度值相同，线性行走控制器被旁路。
0	[31:26]	r	**保留** 读出值为 0；应写入 0。

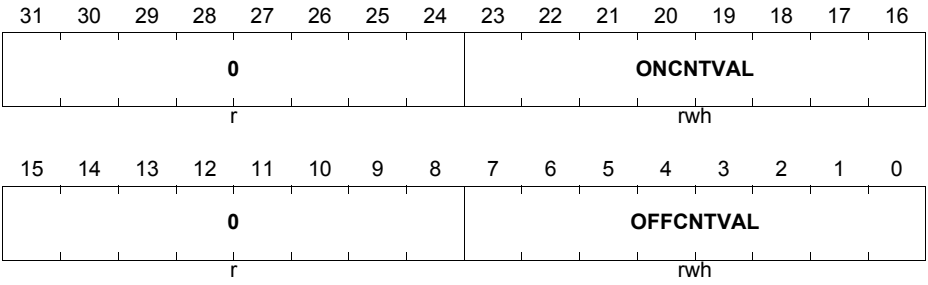
PKCMPy (y=0-8)
打包器比较 **(0048_H + y*0014_H)** **复位值 : 0027 00FF$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								**ONCMP**							
r								rw							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								**OFFCMP**							
r								rw							

域	位	类型	描述
OFFCMP	[7:0]	rw	**打包器断开时间比较值** 当打包器中的断开时间计数器达到该值时，所测量的 sigma-delta 调制器输出的接通时间和断开时间被保存，计数器被复位 0。
0	[15:8]	r	**保留** 读出值为 0；应写入 0。
ONCMP	[23:16]	rw	**打包器接通时间比较值** 当打包器中的接通时间计数器达到该值时，所测量的 sigma-delta 调制器输出的接通时间和断开时间被保存，计数器被复位 0。
0	[31:24]	r	**保留** 读出值为 0；应写入 0。

PKCNTRy (y=0-8)
打包器计数器 **($004C_H + y*0014_H$)** **复位值 : 0000 0000_H**

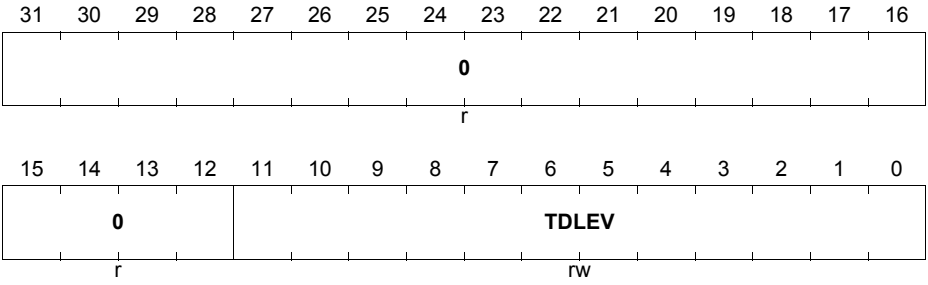


域	位	类型	描述
OFFCNTVAL	[7:0]	rwh	**断开时间计数器值** 在通道被禁止时 （CHEN.ECHy 为 0）可写入，否则写访问被忽略。
0	[15:8]	r	**保留** 读出值为 0；应写入 0。
ONCNTVAL	[23:16]	rwh	**接通时间计数器值** 在通道被禁止时 （CHEN.ECHy 为 0）可写入，否则写访问被忽略。
0	[31:24]	r	**保留** 读出值为 0；应写入 0。

22.8.3 调光引擎寄存器

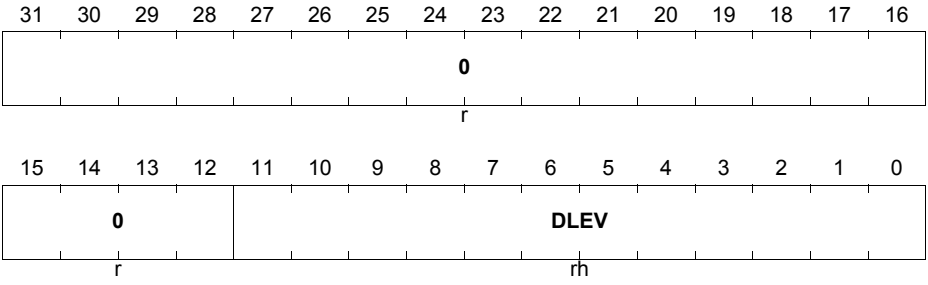
XMC1300 有 3 个可用的调光引擎。 z 表示引擎 0 ~ 2。

DLSz (z=0-2)
调光值映射 **($017C_H$ + z*$000C_H$)** **复位值 : 0000 0000_H**



域	位	类型	描述
TDLEV	[11:0]	rw	**目标调光值**
0	[31:12]	r	**保留** 读出值为 0 ； 应写入 0。

DLz (z=0-2)
调光值 **(0180_H + z*$000C_H$)** **复位值 : 0000 0000_H**



域	位	类型	描述
DLEV	[11:0]	rh	**调光值**
0	[31:12]	r	**保留** 读出值为 0 ； 应写入 0。

DTTz (z=0-2)
调光过渡时间 **(0184_H + z*$000C_H$)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0														CSEL	DTEN
r														rw	rw

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						DIMDIV									
r						rw									

域	位	类型	描述
DIMDIV	[9:0]	rw	**调光时钟分频器** 根据该值对 BCCU_dclk 进行预分频以产生调光时钟；该值可用于调节调光引擎的调光速率。如果该值为 0，则调光值变为与映射传送时的目标调光值相同，调光引擎被旁路。
0	[15:10]	r	**保留** 读出值为 0；应写入 0。
DTEN	16	rw	**抖动使能** 0_B 无抖动 1_B 如果调光值低于 128，每个调光步距都加入抖动；整个调光范围都使用低精度曲线。
CSEL	17	rw	**曲线选择** 0_B 低精度曲线 1_B 高精度曲线 如果抖动被使能 （DTEN=1），该位被忽略并使用低精度曲线。
0	[31:18]	r	**保留** 读出值为 0；应写入 0。

22.9 互连

BCCU0 被连接到端口、ERU、模拟比较器、ADC、NVIC、CCU4、CCU8、挂起信号和内核时钟。

表 22-7 引脚连接

输入 / 输出	I/O	连接到	描述
BCCU0_clk	I	PCLK	BCCU 内核时钟
SUSPEND	I	CPU 停止	挂起信号
BCCU0.IN0	I	ACMP1.OUT	通道 0 的门控输入
BCCU0.IN1	I	未连接 (1)	通道 1 的门控输入
BCCU0.IN2	I	未连接 (1)	通道 2 的门控输入
BCCU0.IN3	I	ACMP2.OUT	通道 3 的门控输入
BCCU0.IN4	I	未连接 (1)	通道 4 的门控输入
BCCU0.IN5	I	ACMP0.OUT	通道 5 的门控输入
BCCU0.IN6	I	未连接 (1)	通道 6 的门控输入
BCCU0.IN7	I	未连接 (1)	通道 7 的门控输入
BCCU0.IN8	I	未连接 (1)	通道 8 的门控输入
BCCU0.TRAPINA	I	P0.12	陷阱输入 A
BCCU0.TRAPINB	I	P0.0	陷阱输入 B
BCCU0.TRAPINC	I	未连接 (0)	陷阱输入 C
BCCU0.TRAPIND	I	未连接 (0)	陷阱输入 D
BCCU0.TRAPINE	I	ERU0.IOUT0	陷阱输入 E
BCCU0.TRAPINF	I	ERU0.IOUT1	陷阱输入 F
BCCU0.TRAPING	I	ERU0.IOUT2	陷阱输入 G
BCCU0.TRAPINH	I	ERU0.IOUT3	陷阱输入 H
BCCU0.TRAPINI	I	SCU.SR2	陷阱输入 I
BCCU0.TRAPINJ	I	未连接 (0)	陷阱输入 J
BCCU0.TRAPINK	I	未连接 (0)	陷阱输入 K
BCCU0.TRAPINL	I	未连接 (0)	陷阱输入 L
BCCU0.TRAPINM	I	未连接 (0)	陷阱输入 M
BCCU0.TRAPINN	I	未连接 (0)	陷阱输入 N
BCCU0.TRAPINO	I	未连接 (0)	陷阱输入 O
BCCU0.TRAPINP	I	未连接 (0)	陷阱输入 P
BCCU0.OUT0	O	P0.4 P1.0 CCU80.IN0I P2.2.HW0 拉控制	通道 0 输出

表 22-7 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
BCCU0.OUT1	O	P0.5 P1.5 CCU80.IN0J P2.0.HW0 拉控制 P2.8.HW0 拉控制	通道 1 输出
BCCU0.OUT2	O	P0.6 CCU40.MCI0 CCU80.IN1I P1.0.HW0 拉控制 P2.6.HW0 拉控制	通道 2 输出
BCCU0.OUT3	O	P0.7 CCU40.MCI1 CCU80.IN1J P1.1.HW0 拉控制	通道 3 输出
BCCU0.OUT4	O	P0.8 CCU40.MCI2 CCU80.IN2I P1.2.HW0 拉控制 P2.10.HW0 拉控制	通道 4 输出
BCCU0.OUT5	O	P0.9 CCU40.MCI3 CCU80.IN2J P1.3.HW0 拉控制 P2.11.HW0 拉控制	通道 5 输出
BCCU0.OUT6	O	P0.10 P0.12 CCU80.IN3I P1.4.HW0 拉控制 P2.1.HW0 拉控制	通道 6 输出
BCCU0.OUT7	O	P0.11 P0.14 CCU80.IN3J P1.5.HW0 拉控制 P2.9.HW0 拉控制	通道 7 输出
BCCU0.OUT8	O	P0.1 P0.15 P2.4.HW0 拉控制 P2.7.HW0 拉控制	通道 8 输出

表 22-7 引脚连接 (续表)

输入 / 输出	I/O	连接到	描述
服务请求连接			
BCCU0.SR0	O	NVIC	服务请求节点 0：中断
BCCU0.TRIGOUT0	O	VADC0.BGREQTRF VADC0.G0REQTRF	ADC 触发信号 0
BCCU0.TRIGOUT1	O	VADC0.G1REQTRF	ADC 触发信号 1

通用 I/O 端口

23 通用 I/O 端口 (端口)

XMC1300 有 34 个连接到片上外设单元的数字通用输入 / 输出 (GPIO) 端口线。

23.1 概述

端口为所有标准数字 I/O 提供一个通用的和非常灵活的软件和硬件接口。每个端口片具有操作通用 I/O 的单独接口，并提供与片上外设的连接和对焊盘特性的控制。**表 23-1** 给出了 XMC1300 不同封装的可用端口和其他引脚的一览表。

表 23-1 端口 / 引脚一览表

端口	TSSOP-38	TSSOP-16	备注
P0	16	8	
P1	6	-	大电流双向焊盘
P2	12	6	模拟 / 数字输入和双向焊盘
电源 VDD, ADC / ORC 参考电压	1	1	VDD
电源 GND, ADC 参考地	1	1	VSS
I/O 端口电源	1	-	VDDP
I/O 端口地	1	-	VSSP

23.1.1 特性

端口的主要特性如下：

- 对每个端口引脚都相同的通用寄存器接口，**23.8 节**
- 针对通用 I/O 功能的简单且健壮的软件访问接口， **23.2 节**
- 与片上外设的直接输入连接，**23.2.1 节**
- 针对 BCCU、 CCU 和 USIC 的专用硬件接口，具有选择选项， **23.3 节**
- 确定的上电 / 掉电行为，**23.6 节**
- 多达 7 个从外设可选的复用输出通路， **23.8.1 节**
- 可编程的漏极开路或推挽输出驱动级， **23.8.1 节**
- 可编程的弱上拉和下拉器件，**23.8.1 节**
- 可编程的输入反相器，**23.8.1 节**
- 可编程的焊盘滞后电压，**23.8.2 节**
- 在共享模拟输入禁止数字输入级， **23.8.3 节**
- 独立的置位和清零输出控制，以避免读 - 修改 - 写操作，**23.8.5 节**
- 可编程的省电行为，在深度休眠模式， **23.8.7 节**
- 在特权模式下，禁止访问配置寄存器，避免意外修改

23.1.2 框图

下面是一个数字端口引脚的通用结构图，分为带控制逻辑的端口片和具有拉器件以及输入和输出级的焊盘，如图 23-1 所示。

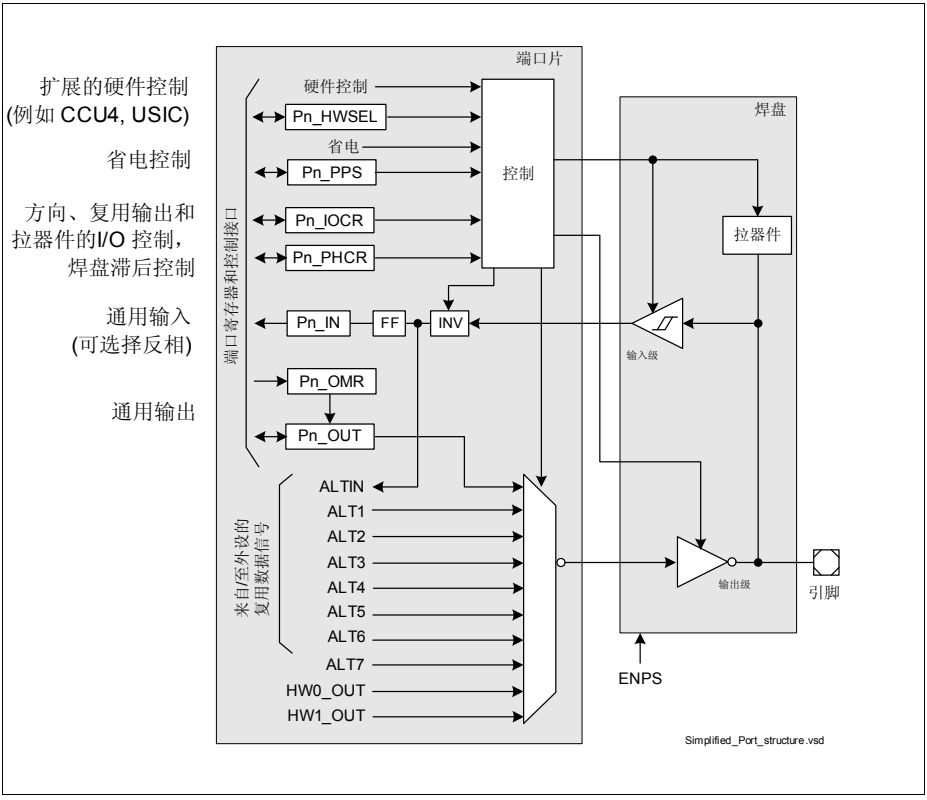


图 23-1　　一个数字端口引脚的通用结构

23.1.3 术语定义

本章使用的一些专用术语如下：

- **引脚 / 球：**器件与 PCB 的外部连接。
- **专用引脚：**具有专用功能的引脚，不受端口逻辑的控制 (即电源引脚)。
- **端口引脚：**受端口逻辑控制的引脚 (例如 P0.1)。
- **端口：**共享相同通用寄存器组的一组端口引脚 (最多 16 个)(例如 P0)。
- **端口片：**用于控制一个端口引脚的寄存器位和控制逻辑的 “ 总和 ”。

- **焊盘：**包含输出驱动器、拉器件和输入施密特触发器的模拟组件。焊盘还将使用 V_{DDC} 运行的内部逻辑与焊盘电源域 V_{DDP} 连接起来。
- **GPIO：**通用输入 / 输出，即具有输入和 / 或输出功能的一个端口引脚，其功能由应用软件控制。
- **复用功能：**一个端口引脚与一个片内外设的直接连接。

23.2 通用和复用功能

端口可作为通用输入 / 输出 (GPIO) 使用，也可以作为片内外设的复用功能，由端口输入 / 输出控制寄存器 (Pn_IOCR，**23.8.1 节**) 配置。它可以选择：

- 直接或反相输入
  - 带或不带拉器件
- 推挽或漏极开路输出，由下面的驱动源驱动：
  - Pn_OUT (GPIO)
  - 所选择的外设输出连接。

当作为 GPIO 使用时，端口引脚由应用软件控制，通过端口输入寄存器 Pn_IN (**23.8.6 节**) 读输入值，通过输出修改寄存器 Pn_OMR (**23.8.5 节**) 设定输出值。与用输出寄存器 Pn_OUT (**23.8.4 节**) 来直接改变输出值相比，应首选用 Pn_OMR 修改输出值，因为 Pn_OMR 允许通过一次访问来单独操作各端口引脚，而不会 “ 干扰 ” 由同一 Pn_OUT 寄存器控制的其他引脚。如果应用程序使用一个 GPIO 作为一条双向 I/O 线，则必须通过写寄存器 Pn_IOCR 来实现输入和输出功能之间的切换。

以复用功能工作时，端口引脚被直接连接到片内外设的输入或输出功能。这允许外设直接评估输入值或驱动端口引脚的输出值，而不需要在初始配置完成后与应用软件进行进一步的交互。复用功能的连接被用作控制和通信接口，像 CAPCOM 单元中的 PWM 或一个 USIC 通道的 SPI 通信。外设与端口引脚的详细连接情况在 **端口 I/O 功能描述** 一节中给出。对于特定功能，有些外设可直接控制 “ 它们的 ” 端口引脚，参见**硬件控制的 I/O**。

23.2.1 输入操作

作为输入，端口引脚的当前电压电平通过焊盘内的一个施密特触发器被转换为逻辑 0_B 或 1_B。可以选择将该结果输入值反相。作为通用输入，信号通过输入寄存器 (Pn_IN，**23.8.6 节**) 被同步和读取。输入还可通过 AITiN 信号连接到多个片内外设。如有必要，这些外设还有内部控制，可以使用一个输入多路复用器来选择一个正确的端口引脚，并负责输入信号的同步和进一步处理 (有关输入选择和处理的更详细信息，请参见相应的外设章节)。还可以用 Pn_IOCR 寄存器 (**23.8.1 节**) 激活焊盘中的弱上拉或下拉器件。

输入寄存器 Pn_IN 和 ALTIN 信号总是代表输入的状态，它与端口引脚是被配置为输入还是输出无关。这意味着，即使端口处于输出模式，软件也可以通过 Pn_IN 读取该引脚的电平，并且 / 或一个外设可以使用该引脚电平作为输入。

焊盘滞后控制

可以根据应用需求通过焊盘滞后控制寄存器 (Pn_PHCR，**23.8.2 节**) 来配置焊盘滞后电压。对于触摸感应应用，选择合适的焊盘滞后为优化焊盘的振荡行为提供了可能。

23.2.2 输出操作

在输出模式，输出驱动器被激活，并通过多路复用器将提供的值驱动到端口引脚。输入和输出模式之间的切换通过 Pn_IOCR 寄存器 (23.8.1 节) 实现。该寄存器

- 使能或禁止输出驱动器，
- 选择漏极开路模式或推挽模式，
- 选择通用或复用功能输出。

输出多路复用器选择输出的信号源 ：

- Pn_IOCR
  - 通用输出 (Pn_OUT，23.8.4 节)
  - 复用的外设功能， ALT1...ALT7
- 硬件控制， Pn_HWSEL
  - HW0_OUT
  - HW1_OUT

注： 建议在端口引脚被切换到输出模式之前完成对端口和外设工作模式及初始值的配置。

输出功能具有排他性，这意味着在任何时刻只有一个外设控制输出通路。

作为通用输出，软件可以直接修改 Pn_OUT 的内容来设定引脚的输出值。对 Pn_OUT 的写操作会更新对应端口中被配置为通用输出的所有端口引脚 (例如 P0)。通过 Pn_OUT 来更新一个或少量选中的通用输出引脚要求一次屏蔽的读 - 修改 - 写操作，以避免干扰那些不应改变的引脚。直接写 Pn_OUT 还会影响被配置用于引脚省电功能的 Pn_OUT 位，23.4 节。

因此，最好通过输出修改寄存器 Pn_OMR (23.8.5 节) 来修改 Pn_OUT 位。Pn_OMR 中的位允许单独置位、清除或切换Pn_OUT寄存器中的位，并且仅更新“被寻址的”Pn_OUT 位。

通过软件写入到输出寄存器 Pn_OUT 中的数据还可作为一个片内外设的输入数据。例如，这使通过软件实现外设测试和仿真成为可能，而无需借助于外部电路。如果通过ALT1到ALT7以及HW0_OUT和HW1_OUT选择了片内外设的输出线，则这些输出线可以直接控制输出驱动器的输出值。初始化完成后，所连接的外设可以直接驱动复杂的控制和通信模式，而不需要软件与端口进行进一步的交互。

可通过读 Pn_IN 的值并将其与所施加的输出电平 (无论是由输出寄存器 Pn_OUT 施加，还是通过一个外设单元的复用输出功能施加)值进行比较来检查引脚的当前逻辑电平是否正确。这可以用来检测由外部电路在该引脚引起的某些电气故障。此外 ， 可用该方法实现软件支持的不同 “ 主机 ” 之间的仲裁机制，此时要使用漏极开路配置和一个外部线与电路。当输出一个高电平 (1_B) 时，通过输入寄存器 Pn_IN 或直接由一个外设 (通过 ALTIN，例如工作在 IIC 模式的一个 USIC 通道) 读取的引脚值却是一个低电平 (0_B)，则认为检测到外部通信线路冲突。

XMC1300 有两种焊盘类型，可提供不同的驱动强度：

- 标准焊盘
- 大电流焊盘

每个端口引脚被分配的焊盘类型在封装引脚汇总表中列出。有关焊盘属性的更详细信息请参见数据表中的焊盘属性描述。

23.3 硬件控制的 I/O

有些端口引脚被外设功能覆盖。对于这些引脚，所连接外设需要硬件直接控制，例如一个双向数据总线的方向。针对这些功能，有一个专用的硬件控制接口。由于有多个外设需要访问该接口，所以 Pn_HWSEL 寄存器 (**23.8.8 节**) 允许选择不同的硬件 “ 主机 ”。

根据工作模式不同，外设可对不同的功能进行控制：

- 引脚方向，即输入或输出，例如双向信号
- 驱动器类型，即漏极开路或推挽
- 受外设控制或通过 Pn_IOCR 控制 (标准控制) 的拉器件

有些配置仍受标准配置接口的控制。焊盘滞后由 Pn_PHCR 控制 ，直接或反相输入通路由 Pn_IOCR 控制。

Pn_HWSEL.HWx 只是将引脚的硬件控制预先分配给一个特定的外设，但由外设自己决定何时控制该引脚。只要外设不控制由 HWx_EN 给定的引脚，则该引脚的配置仍然由配置寄存器定义，并且它可被用作 GPIO 或其他复用功能。这可能是因为所选择的外设只能控制激活其引脚的一个子集，或者因为外设根本就没被激活。

在应用不需要相应的功能并且外设无法选择性地禁止硬件控制的情况下，该机制也可以用于禁止某些引脚对一个外设的硬件控制。

默认的硬件输入配置和拉器件由 Pn_IOCR 控制。

注： 对于那些被配置为用于硬件控制的引脚 (Pn_HWSEL.HWx != 00B)，不要使能引脚省电功能。这样做可能会在设备进入深度休眠状态时导致不确定的引脚行为。

23.4 省电模式操作

在深度休眠模式，引脚的行为取决于引脚省电寄存器 (Pn_PPS， **23.8.7 节**) 的设置。基本上，每个引脚都可以被配置为响应或忽略省电模式请求。如果一个引脚被配置为响应省电模式请求，则其输出驱动器被切换到三态，输入施密特触发器和拉器件被关闭 (参见 **图 23-2**)。到片内外设的输入信号可选择被驱动为静态高电平或低电平，由软件定义是使用存储在 Pn_OUT 中的值还是使用正常运行期间被采样到 Pn_OUT 寄存器的最后一个输入值。在省电条件下，实际的反应由 Pn_IOCR 寄存器配置，见 **表 23-8**。

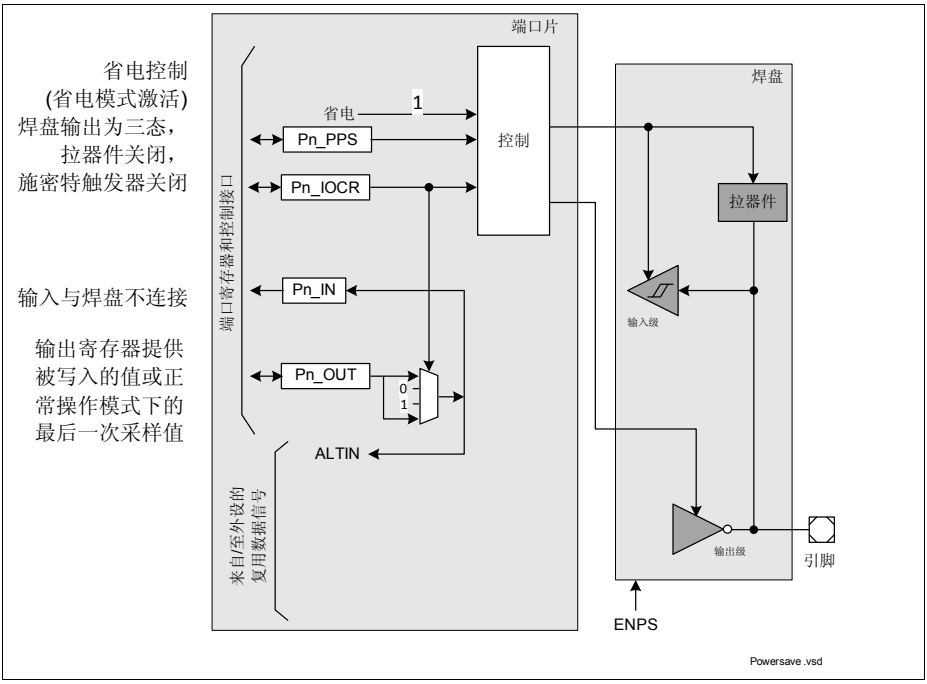


图 23-2　　省电状态下的端口引脚

注：　对于那些被配置为用于硬件控制的引脚 (Pn_HWSEL.HWx != 00_B)，不要使能引脚省电功能。这样做可能会在设备进入深度休眠状态时导致不确定的引脚行为。

23.5　　模拟端口

P2.2 - P2.9 是具有简化的端口和焊盘结构的模拟和数字输入端口，见 **图 23-3**。模拟焊盘没有输出驱动器，其数字输入施密特触发器可由 Pn_PDISC (**23.8.3 节**) 寄存器控制。端口控制接口的功能也相应地减少。Pn_IOCR 寄存器控制拉器件、可选的输入反向和省电模式下的输入源。 Pn_OUT 只有省电功能，如 **23.4 节**所述。

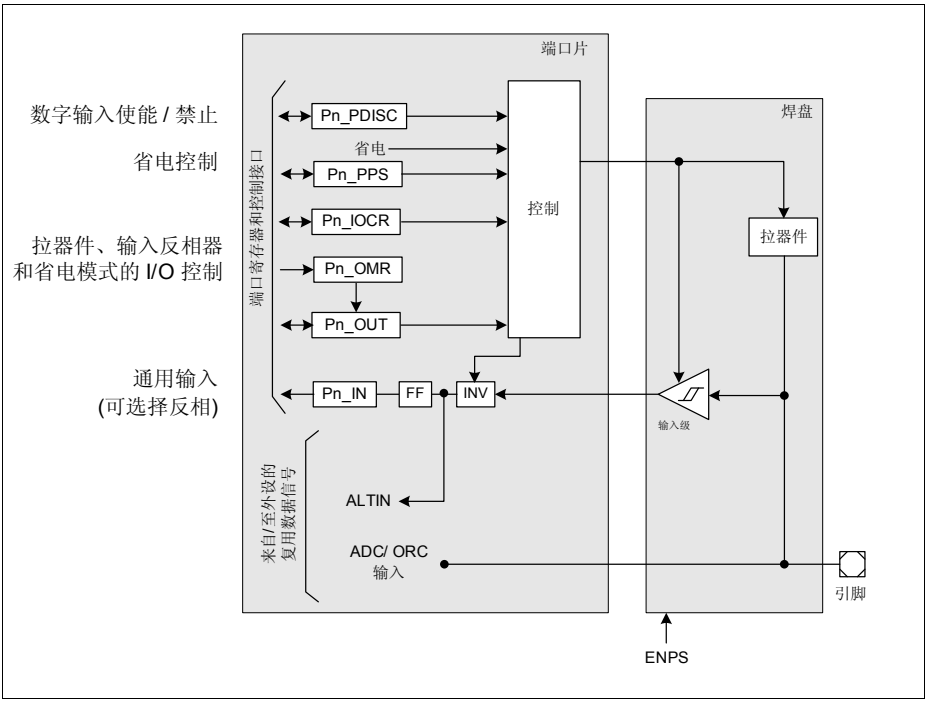


图 23-3　　模拟端口结构

23.6　　电源、复位和时钟

当发生下列情况之一时，所有的数字 I / O 焊盘都保持在一个确定的状态，即三态，输出驱动器被禁止并且拉器件不激活：

- 在上电期间，直到 V_{DDC} 和 V_{DDP} 电压电平稳定且位于极限值之内
- 在电源故障期间，一个或多个电压电平超出极限值之外

有关上电、电源监视和电压极限值的详细信息，请参见SCU一章中的EVR部分和数据表。

所有端口寄存器都会随着系统复位而复位(参见系统控制单元一章中的复位控制单元一节)。标准复位值定义为：端口引脚被配置为三态输入、输出驱动器禁止和拉器件不激活。来自这些标准值的异常与特殊接口或模拟输入通道有关。

端口的所有寄存器都以 f_{MCLK} 为时钟。

23.7　　初始化和系统相关性

建议按照预先定义的例程来初始化端口引脚。

输入

当一个外设应使用一个端口引脚作为输入时，当前的引脚电平可能会立即触发一个意想不到的外设事件 (例如在 SPI 引脚的时钟边沿)。这可以通过上拉 / 下拉编程而强制产生 “ 被动电平 ” 来避免。

需要按照以下步骤将一个端口引脚配置为输入：

- Pn_IOCR
  带拉器件的输入配置和 / 或省电模式配置
- Pn_PHCR
  焊盘滞后电压配置 (如果适用)
- 硬件控制 (如果适用)
  – Pn_HWSEL
    将硬件控制切换到外设
- 引脚省电 (如果适用)
  – Pn_OMR/Pn_OUT
    省电模式的默认值 (如果适用)
  – Pn_PPS
    使能省电控制

输出

当一个端口引脚被配置为一个片内外设的输出时，为了避免在输出端出现尖峰脉冲，要在端口将控制切换到外设之前完成对该外设的配置，这是很重要的。
需要按照以下步骤将一个端口引脚配置为输出：

- Pn_OMR/Pn_OUT
  初始输出值 (作为通用输出)
- GPIO 或复用输出
  – Pn_IOCR
    输出多路复用器选择
    推挽或漏极开路输出驱动器模式
    激活输出驱动器 !
- 硬件控制
  – Pn_IOCR
    Pn_IOCR 可根据硬件功能使能内部拉器件
  – Pn_HWSEL
    将硬件控制切换到外设

转换

如果一个端口引脚用于不同的功能，因而需要重新配置端口寄存器，建议通过一个中间的 “ 中性 ” 三态输入配置来转换。

- Pn_HWSEL
  禁止硬件选择；如果端口引脚不使用硬件控制，则可以忽略

- Pn_PPS
  禁止引脚的省电模式控制；如果端口引脚不使用省电配置，则可以忽略
- Pn_IOCR
  三态输入并且不激活拉器件

23.8 寄存器

寄存器概览

绝对寄存器地址通过下面的加法计算：

模块基地址 + 偏移地址

表 23-2 寄存器地址空间

模块	基地址	结束地址	备注
P0	$4004\ 0000_H$	$4004\ 00FF_H$	
P1	$4004\ 0100_H$	$4004\ 01FF_H$	大电流双向焊盘
P2	$4004\ 0200_H$	$4004\ 02FF_H$	模拟 / 数字输入和双向焊盘

表 23-3 寄存器一览表

简称	描述	偏移地址	访问方式		描述见
			读	写	
Pn_OUT	端口 n 输出寄存器	0000_H	U, PV	U, PV	页 23-26
Pn_OMR	端口 n 输出修改寄存器	0004_H	U, PV	U, PV	页 23-29
–	保留	0008_H-$000C_H$	BE	BE	–
Pn_IOCR0	端口 n 输入 / 输出控制寄存器 0	0010_H	U, PV	PV	页 23-12
Pn_IOCR4	端口 n 输入 / 输出控制寄存器 4	0014_H	U, PV	PV	页 23-13
Pn_IOCR8	端口 n 输入 / 输出控制寄存器 8	0018_H	U, PV	PV	页 23-14
Pn_IOCR12	端口 n 输入 / 输出控制寄存器 12	$001C_H$	U, PV	PV	页 23-14
–	保留	0020_H	BE	BE	–
Pn_IN	端口 n 输入寄存器	0024_H	U, PV	R	页 23-32
–	保留	0028_H-$003C_H$	BE	BE	–
Pn_PHCR0	端口 n 焊盘滞后控制寄存器 0	0040_H	U, PV	PV	页 23-18
Pn_PHCR1	端口 n 焊盘滞后控制寄存器 1	0044_H	U, PV	PV	页 23-19

表 23-3 寄存器一览表

简称	描述	偏移地址	访问方式		描述见
			读	写	
–	保留	0048_H-$005C_H$	BE	BE	–
P0_PDISC P1_PDISC	端口 n 引脚功能控制寄存器 (非 ADC 端口)	0060_H	U, PV	BE	页 23-21
P2_PDISC	端口 n 引脚功能控制寄存器 (ADC 端口)	0060_H	U, PV	PV	页 23-22
–	保留	0064_H-$006C_H$	BE	BE	–
Pn_PPS	端口 n 引脚省电寄存器	0070_H	U, PV	PV	页 23-34
Pn_HWSEL	端口 n 硬件选择寄存器	0074_H	U, PV	PV	页 23-37
–	保留	0078_H-$00FC_H$	BE	BE	–

表 23-4 寄存器访问权限和复位类别

寄存器简称	访问权限		复位类别
	读	写	
Pn_IN	U, PV	R	系统复位
Pn_OUT		U, PV	
Pn_OMR			
Pn_IOCR0		PV	
Pn_IOCR4			
Pn_IOCR8			
Pn_IOCR12			
Pn_PDISC (ADC 端口)			
Pn_PH0			
Pn_PH1			
Pn_PPS			
Pn_PDISC (非 ADC 端口)		BE	

23.8.1 端口输入 / 输出控制寄存器

端口输入 / 输出控制寄存器可以选择一个 GPIO 端口引脚的数字输出和输入驱动器功能和特性。可通过相应位域 PCx (x = 0-15) 来选择端口方向 (输入或输出)、用于输入的上拉或下拉器件和用于输出的推挽或漏极开路功能。每个 32 位宽的端口输入 / 输出控制寄存器控制四个 GPIO 端口线：

寄存器 Pn_IOCR0 控制 Pn.[3:0] 端口线
寄存器 Pn_IOCR4 控制 Pn.[7:4] 端口线
寄存器 Pn_IOCR8 控制 Pn.[11:8] 端口线
寄存器 Pn_IOCR12 控制 Pn.[15:12] 端口线

下面以 PCx 位域给出端口输入 / 输出控制寄存器的寄存器分布。一个 PCx 位域正好控制一个 Pn.x 端口线。

Pn_IOCR0 (n=0-1)
端口 n 输入 / 输出控制寄存器 0 (4004 0010$_H$ + n*100$_H$) **复位值 : 0000 0000$_H$**
P2_IOCR0
端口 2 输入 / 输出控制寄存器 0 (0010$_H$) **复位值 : 0000 0000$_H$**

31 27	26 24	23 19	18 16	15 11	10 8	7 3	2 0
PC3	0	PC2	0	PC1	0	PC0	0
rw	r	rw	r	rw	r	rw	r

域	位	类型	描述
PC0, PC1, PC2, PC3	[7:3], [15:11], [23:19], [31:27]	rw	**端口 n 引脚 0-3 的端口控制** 该位域根据编码表 (见 **表 23-5**) 决定端口 n 线 x 的功能 (x = 0-3)。
0	[2:0], [10:8], [18:16], [26:24]	r	**保留** 读出值为 0； 应写入 0。

P0_IOCR4
端口 0 输入 / 输出控制寄存器 4 **(0014_H)** **复位值 : 0000 0000$_H$**
P2_IOCR4
端口 2 输入 / 输出控制寄存器 4 **(0014_H)** **复位值 : 0000 0000$_H$**

31 27	26 24	23 19	18 16	15 11	10 8	7 3	2 0
PC7	0	PC6	0	PC5	0	PC4	0
rw	r	rw	r	rw	r	rw	r

域	位	类型	描述
PC4, **PC5,** **PC6,** **PC7**	[7:3], [15:11], [23:19], [31:27]	rw	**端口 n 引脚 4-7 的端口控制** 该位域根据编码表 (见 **表 23-5**) 决定端口 n 线 x 的功能 (x = 4-7) 。
0	[2:0], [10:8], [18:16], [26:24]	r	**保留** 读出值为 0； 应写入 0。

P1_IOCR4
端口 1 输入 / 输出控制寄存器 4 **(0014_H)** **复位值 : 0000 0000$_H$**

31 19	18 16	15 11	10 8	7 3	2 0
0	0	PC5	0	PC4	0
r	r	rw	r	rw	r

域	位	类型	描述
PC4, **PC5**	[7:3], [15:11]	rw	**端口 n 引脚 4-5 的端口控制** 该位域根据编码表 (见 **表 23-5**) 决定端口 n 线 x 的功能 (x = 4-5) 。
0	[2:0], [10:8], [18:16], [31:19]	r	**保留** 读出值为 0； 应写入 0。

P0_IOCR8
端口 0 输入 / 输出控制寄存器 8 (0018_H) 复位值 : $0000\ 0000_H$[1)]
P2_IOCR8
端口 2 输入 / 输出控制寄存器 8 (0018_H) 复位值 : $0000\ 0000_H$

31 27	26 24	23 19	18 16	15 11	10 8	7 3	2 0
PC11	0	PC10	0	PC9	0	PC8	0
rw	r	rw	r	rw	r	rw	r

1) **复位时，**PC8 (P0.8) **的值为** 00000_B**。启动软件** (SSW) **将** PC8 **的值改变为输入上拉器件激活，即** 00010_B**。更详细的信息请参见启动一章。**

域	位	类型	描述
PC8, PC9, PC10, PC11	[7:3], [15:11], [23:19], [31:27]	rw	**端口 n 引脚 8-11 的端口控制** 该位域根据编码表 (见 **表 23-5**) 决定端口 n 线 x 的功能 (x = 8-11) 。
0	[2:0], [10:8], [18:16], [26:24]	r	**保留** 读出值为 0； 应写入 0。

P0_IOCR12
端口 0 输入 / 输出控制寄存器 12 ($001C_H$) 复位值 : $0000\ 0000_H$

31 27	26 24	23 19	18 16	15 11	10 8	7 3	2 0
PC15	0	PC14	0	PC13	0	PC12	0
rw	r	rw	r	rw	r	rw	r

域	位	类型	描述
PC12, PC13, PC14, PC15	[7:3], [15:11], [23:19], [31:27]	rw	**端口 n 引脚 12-15 的端口控制** 该位域根据编码表 (见 **表 23-5**) 决定端口 n 线 x 的功能 (x = 12-15) 。

域	位	类型	描述
0	[2:0], [10:8], [18:16], [26:24]	r	**保留** 读出值为 0； 应写入 0。

根据 GPIO 端口功能 (GPIO 端口线的数量) 的不同，并非所有的端口输入 / 输出控制寄存器都被实现。

每个端口引脚都有一个控制位域，该位域位于不同的寄存器字节。这种结构为使用提供面向字节的访问来配置单个引脚的端口引脚功能提供了可能，而无需访问其它PCx位域。

端口控制编码

表 23-5 描述了 PCx 位域的编码，该编码决定端口线的功能。

如果引脚省电选项被使能，则 Pn_IOCRy PCx 位域还可用于控制深度休眠模式下的引脚行为，参见 23.8.7 节。

表 23-5 标准 PCx 编码[1)]

PCx[4:0]	I/O	输出特性	选择上拉 / 下拉 / 输出功能
$0X000_B$	直接输入	–	不激活内部拉器件
$0X001_B$			激活内部下拉器件
$0X010_B$			激活内部上拉器件
$0X011_B$			不激活内部拉器件； Pn_OUTx 连续采样输入值
$0X100_B$	反相输入	–	不激活内部拉器件
$0X101_B$			激活内部下拉器件
$0X110_B$			激活内部上拉器件
$0X111_B$			不激活内部拉器件； Pn_OUTx 连续采样输入值

表 23-5 标准 PCx 编码[1] (续表)

PCx[4:0]	I/O	输出特性	选择上拉 / 下拉 / 输出功能
10000_B	输出 (直接输入)	推挽	通用输出
10001_B			复用输出功能 1
10010_B			复用输出功能 2
10011_B			复用输出功能 3
10100_B			复用输出功能 4
10101_B			复用输出功能 5
10110_B			复用输出功能 6
10111_B			复用输出功能 7
11000_B		漏极开路	通用输出
11001_B			复用输出功能 1
11010_B			复用输出功能 2
11011_B			复用输出功能 3
11100_B			复用输出功能 4
11101_B			复用输出功能 5
11110_B			复用输出功能 6
11111_B			复用输出功能 7

1) 对于模拟和数字输入端口为 P2.2 - P2.9， PCx[4]=1_B 的组合被保留。

23.8.2 焊盘滞后控制寄存器

XMC1300 的 GPIO 线的焊盘结构为选择焊盘滞后提供了可能。这两个参数由焊盘滞后控制寄存器 Pn_PHCR0/1 中的位域控制，与在 Pn_IOCR 寄存器中编程的输入 / 输出和上拉 / 下拉控制功能无关。 Pn_PHCR0 和 Pn_PHCR1 寄存器被分配给每个端口。

焊盘滞后控制寄存器 Pn_PHCR 中的 1 位焊盘滞后选择位域 PHx 使选择端口线功能成为可能，如 表 23-6 所示。注意焊盘滞后控制寄存器对每个端口都是专用的 。

表 23-6 焊盘滞后电压选择

PHx	功能
0	标准滞后电压
1	大滞后电压

注： 关于滞后参数值，请参见 XMC1300 数据表中的输入 / 输出特性表。

焊盘滞后控制寄存器

这是对 PHCR 寄存器的一般性描述。每个端口都包含其自己的专用 PHCR 寄存器，在每个端口的描述中介绍该寄存器。PHCR0 和 PHCR1 寄存器可以分别包含一个到八个 PHx 域。每个域控制一个引脚。PHx 的编码见 页 23-16。

P0_PHCR0
端口 0 焊盘滞后控制寄存器 0 **(0040_H)** **复位值 : $0000\ 0000_H$**
P2_PHCR0
端口 2 焊盘滞后控制寄存器 0 **(0040_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**PH7**	**0**			**PH6**	**0**			**PH5**	**0**			**PH4**	**0**	
r	rw	r			rw	r			rw	r			rw	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**PH3**	**0**			**PH2**	**0**			**PH1**	**0**			**PH0**	**0**	
r	rw	r			rw	r			rw	r			rw	r	

域	位	类型	描述
PH0	2	rw	**Pn.0 焊盘滞后**
PH1	6	rw	**Pn.1 焊盘滞后**
PH2	10	rw	**Pn.2 焊盘滞后**
PH3	14	rw	**Pn.3 焊盘滞后**
PH4	18	rw	**Pn.4 焊盘滞后**
PH5	22	rw	**Pn.5 焊盘滞后**
PH6	26	rw	**Pn.6 焊盘滞后**
PH7	30	rw	**Pn.7 焊盘滞后**
0	[1:0], [5:3], [9:7], [13:11], [17:15], [21:19], [25:23], [29:27], 31	r	**保留** 读出值为 0； 应写入 0。

P1_PHCR0
端口 1 焊盘滞后控制寄存器 0 **(0040$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						**0**			**PH5**	**0**			**PH4**	**0**	
r						r			rw	r			rw	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**PH3**	**0**			**PH2**	**0**			**PH1**	**0**			**PH0**	**0**	
r	rw	r			rw	r			rw	r			rw	r	

域	位	类型	描述
PH0	2	rw	**P1.0 焊盘滞后**
PH1	6	rw	**P1.1 焊盘滞后**
PH2	10	rw	**P1.2 焊盘滞后**
PH3	14	rw	**P1.3 焊盘滞后**
PH4	18	rw	**P1.4 焊盘滞后**
PH5	22	rw	**P1.5 焊盘滞后**
0	[1:0], [5:3], [9:7], [13:11], [17:15], [21:19], [25:23], [31:26]	r	**保留** 读出值为 0； 应写入 0。

P0_PHCR1
端口 0 焊盘滞后控制寄存器 1 **(0044_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0	**PH15**	**0**			**PH14**	**0**			**PH13**	**0**			**PH12**	**0**	
r	rw	r			rw	r			rw	r			rw	r	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	**PH11**	**0**			**PH10**	**0**			**PH9**	**0**			**PH8**	**0**	
r	rw	r			rw	r			rw	r			rw	r	

域	位	类型	描述
PH8	2	rw	**P0.8 焊盘滞后**
PH9	6	rw	**P0.9 焊盘滞后**
PH10	10	rw	**P0.10 焊盘滞后**
PH11	14	rw	**P0.11 焊盘滞后**
PH12	18	rw	**P0.12 焊盘滞后**
PH13	22	rw	**P0.13 焊盘滞后**
PH14	26	rw	**P0.14 焊盘滞后**
PH15	30	rw	**P0.15 焊盘滞后**
0	[1:0], [5:3], [9:7], [13:11], [17:15], [21:19], [25:23], [29:27], 31	r	**保留** 读出值为 0； 应写入 0。

P2_PHCR1
端口 2 焊盘滞后控制寄存器 1 **(0044$_H$)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PH11	0			PH10	0			PH9	0			PH8	0	
r	rw	r			rw	r			rw	r			rw	r	

域	位	类型	描述
PH8	2	rw	**P2.8 焊盘滞后**
PH9	6	rw	**P2.9 焊盘滞后**
PH10	10	rw	**P2.10 焊盘滞后**
PH11	14	rw	**P2.11 焊盘滞后**
0	[1:0], [5:3], [9:7], [13:11], [31:15]	r	**保留** 读出值为 0； 应写入 0。

23.8.3 引脚功能控制寄存器

引脚功能控制寄存器

该寄存器主要用与禁止 / 使能共享的模拟和数字端口中的数字焊盘结构，参见对 **P2_PDISC** 的专门描述。

对于 " 正常 " 的数字通用 I/O 端口 (P0-P1)，该寄存器是只读的，并且读出值对应于给定封装中的可用引脚。

P0_PDISC
端口 0 引脚功能控制寄存器 **(0060_H)** **复位值 : 0000 XXXX$_H$[1)]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDIS 15	PDIS 14	PDIS 13	PDIS 12	PDIS 11	PDIS 10	PDIS 9	PDIS 8	PDIS 7	PDIS 6	PDIS 5	PDIS 4	PDIS 3	PDIS 2	PDIS 1	PDIS 0
r	r	r	r	r	r	r	r	r	r	r	r	r	r	r	r

1) 复位值取决于封装。

域	位	类型	描述
PDISx (x = 0-15)	x	r	**端口 0 引脚 x 焊盘禁止** 0_B 使能焊盘 P0.x 。 1_B 禁止焊盘 P0.x 。
0	[31:16]	r	**保留** 读出值为 0； 应写入 0。

P1_PDISC
端口 1 引脚功能控制寄存器 **(0060_H)** **复位值 : 0000 00XX$_H$[1)]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										PDIS 5	PDIS 4	PDIS 3	PDIS 2	PDIS 1	PDIS 0
r										r	r	r	r	r	r

1) 复位值取决于封装。

域	位	类型	描述
PDISx (x = 0-5)	x	r	**端口 1 引脚 x 焊盘禁止** 0_B 使能焊盘 P1.x 。 1_B 禁止焊盘 P1.x 。
0	[31:6]	r	**保留** 读出值为 0； 应写入 0。

P2_PDISC
端口 2 引脚功能控制寄存器 **(0060_H)** **复位值 : 0000 0XXX$_H$[1)]**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				PDIS 11	PDIS 10	PDIS 9	PDIS 8	PDIS 7	PDIS 6	PDIS 5	PDIS 4	PDIS 3	PDIS 2	PDIS 1	PDIS 0
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

1) 复位值取决于封装。

域	位	类型	描述
PDIS0	0	rw	**端口 2 引脚 0 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入 / 输出。 1_B 禁止焊盘，选择 ADC S&H 0 的模拟输入 5。(默认)
PDIS1	1	rw	**端口 2 引脚 1 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入 / 输出。 1_B 禁止焊盘，选择 ADC S&H 0 模拟输入 6 。(默认)
PDIS2	2	rw	**端口 2 引脚 2 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入。 1_B 禁止焊盘，选择 ADC S&H 0 模拟输入 7 。(默认)
PDIS3	3	rw	**端口 2 引脚 3 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入。 1_B 禁止焊盘，选择 ADC S&H 1 模拟输入 5 。(默认)
PDIS4	4	rw	**端口 2 引脚 4 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入。 1_B 禁止焊盘，选择 ADC S&H 1 模拟输入 6 。(默认)

域	位	类型	描述
PDIS5	5	rw	**端口 2 引脚 5 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入。 1_B 禁止焊盘，选择 ADC S&H 1 模拟输入 7 。(默认)
PDIS6	6	rw	**端口 2 引脚 6 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入。 1_B 禁止焊盘，选择 ADC S&H 0 模拟输入 0 。(默认)
PDIS7	7	rw	**端口 2 引脚 7 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入。 1_B 禁止焊盘，选择 ADC S&H 1 模拟输入 1 。(默认)
PDIS8	8	rw	**端口 2 引脚 8 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入。 1_B 禁止焊盘，选择 ADC S&H 0 模拟输入 1 和 ADC S&H 1 模拟输入 0 。(默认)
PDIS9	9	rw	**端口 2 引脚 9 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入。 1_B 禁止焊盘，选择 ADC S&H 0 模拟输入 2 和 ADC S&H 1 模拟输入 4 。(默认)

域	位	类型	描述
PDIS10	10	rw	**端口 2 引脚 10 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入 / 输出。 1_B 禁止焊盘，选择 ADC S&H 0 模拟输入 3 和 ADC S&H 1 模拟输入 2 。(默认)
PDIS11	11	rw	**端口 2 引脚 11 焊盘禁止** 该位禁止或使能数字焊盘功能。焊盘的禁止 (默认) 状态选择 ADC 模拟输入。一旦被使能，功能切换到数字输入 / 输出。 0_B 使能焊盘，选择数字输入 / 输出。 1_B 禁止焊盘，选择 ADC S&H 0 模拟输入 4 和 ADC S&H 1 模拟输入 3 。(默认)
0	[31:12]	r	**保留** 读出值为 0 ； 应写入 0。

23.8.4 端口输出寄存器

当一个 GPIO 引脚被 Pn_IOCRx 选择为输出时，相应的端口输出寄存器决定该引脚的值。向 Pn_OUT.Px (x = 0-15) 位写 0 会导致在相应的输出引脚输出一个低电平。当相应的位被写入 1 时，输出一个高电平。注意，通过向端口输出修改寄存器 Pn_OMR 写入适当的值，可单独置位 / 复位 Pn_OUT.Px 中的位，这样可以避免因对 Pn_OUT 执行读 - 修改 - 写操作而影响端口的其他引脚。

Pn_OUT 还用于为深度休眠模式下的输入存储 / 驱动一个确定的值。有关这方面的更详细的信息，参见**端口引脚省电寄存器**。这也是在模拟和数字输入端口 P2.2 - P2.9 中的 Pn_OUT 寄存器的唯一用途。

P0_OUT
端口 0 输出寄存器 **(0000_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	**P14**	**P13**	**P12**	**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
Px **(x = 0-15)**	x	rwh	**端口 0 输出位 x** 如果输出被选择作为 GPIO 输出，则该位决定输出引脚 P0.x 的电平。 0_B P0.x 的输出电平为 0。 1_B P0.x 的输出电平为 1。 P0.x 也可以通过 P0_OMR 寄存器的控制位置位 / 复位。
0	[31:16]	r	**保留** 读出值为 0； 应写入 0。

P1_OUT
端口 1 输出寄存器 **(0000_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										P5	P4	P3	P2	P1	P0
r										rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
Px **(x = 0-5)**	x	rwh	**端口 1 输出位 x** 如果输出被选择作为 GPIO 输出，则该位决定输出引脚 P1.x 的电平。 0_B P1.x 的输出电平为 0。 1_B P1.x 的输出电平为 1。 P1.x 也可以通过 P1_OMR 寄存器的控制位置位 / 复位。
0	[31:6]	r	**保留** 读出值为 0； 应写入 0。

P2_OUT
端口 2 输出寄存器 **(0000_H)** **复位值 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
r				rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

域	位	类型	描述
Px **(x = 0-11)**	x	rwh	**端口 2 输出位 x** 如果输出被选择作为 GPIO 输出，则该位决定输出引脚 P2.x 的电平。 0_B P2.x 的输出电平为 0。 1_B P2.x 的输出电平为 1。 P2.x 也可以通过 P2_OMR 寄存器的控制位置位 / 复位。
0	[31:12]	r	**保留** 读出值为 0； 应写入 0。

23.8.5 端口输出修改寄存器

端口输出修改寄存器包含的控制位允许通过操控输出寄存器来单独置位、复位或切换单个端口线的逻辑状态。

P0_OMR
端口 0 输出修改寄存器 **(0004_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PR15	PR14	PR13	PR12	PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PS15	PS14	PS13	PS12	PS11	PS10	PS9	PS8	PS7	PS6	PS5	PS4	PS3	PS2	PS1	PS0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

域	位	类型	描述
PSx (x = 0-15)	x	w	**端口 0 置位 x** 将该位置 1 会置位或切换端口输出寄存器 P0_OUT 中的相应位。该位的功能如 **表 23-7** 所示。
PRx (x = 0-15)	x + 16	w	**端口 0 复位 x** 将该位置 1 会复位或切换端口输出寄存器 P0_OUT 中的相应位。该位的功能如 **表 23-7** 所示。

P1_OMR
端口 1 输出修改寄存器 **(0004_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0										PR5	PR4	PR3	PR2	PR1	PR0
r										w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										PS5	PS4	PS3	PS2	PS1	PS0
r										w	w	w	w	w	w

域	位	类型	描述
PSx (x = 0-5)	x	w	**端口 1 置位 x** 将该位置 1 会置位或切换端口输出寄存器 P1_OUT 中的相应位。该位的功能如 表 23-7 所示。
PRx (x = 0-5)	x + 16	w	**端口 1 复位 x** 将该位置 1 会复位或切换端口输出寄存器 P1_OUT 中的相应位。该位的功能如 表 23-7 所示。
0	[15:6], [31:22]	r	**保留** 读出值为 0； 应写入 0。

P2_OMR
端口 2 输出修改寄存器 **(0004_H)** **复位值 : $0000\ 0000_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				PR11	PR10	PR9	PR8	PR7	PR6	PR5	PR4	PR3	PR2	PR1	PR0
r				w	w	w	w	w	w	w	w	w	w	w	w

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				PS11	PS10	PS9	PS8	PS7	PS6	PS5	PS4	PS3	PS2	PS1	PS0
r				w	w	w	w	w	w	w	w	w	w	w	w

域	位	类型	描述
PSx (x = 0-11)	x	w	**端口 2 置位 x** 将该位置 1 会置位或切换端口输出寄存器 P2_OUT 中的相应位。该位的功能如表 23-7 所示。
PRx (x = 0-11)	x + 16	w	**端口 2 复位 x** 将该位置 1 会复位或切换端口输出寄存器 P2_OUT 中的相应位。该位的功能如表 23-7 所示。
0	[15:12], [31:28]	r	**保留** 读出值为 0； 应写入 0。

注： 寄存器 Pn_OMR 是虚拟的，不包含任何触发器。读操作返回 0 值。8 或 16 位的写操作像 32 位写操作一样执行，以 0 填充剩余位。

表 23-7　　位 PRx 和 PSx 的功能

PRx	PSx	功能
0	0	不改变位 Pn_OUT.Px 。
0	1	置位 Pn_OUT.Px 。
1	0	复位 Pn_OUT.Px 。
1	1	切换位 Pn_OUT.Px 。

23.8.6 端口输入寄存器

GPIO 引脚的逻辑电平可以通过只读端口输入寄存器 Pn_IN 读取。读 Pn_IN 寄存器时总是返回出现在 GPIO 引脚并经过同步后的当前逻辑值，以避免亚稳定性，该读操作与引脚被选择为输入还是输出无关。

P0_IN
端口 0 输入寄存器 **(0024_H)** **复位值 : 0000 XXXX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	P0
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

域	位	类型	描述
Px **(x = 0-15)**	x	rh	**端口 0 输入位 x** 该位指示输入引脚 P0.x 的电平。 0_B P0.x 的输入电平为 0。 1_B P0.x 的输入电平为 1。
0	[31:16]	r	**保留** 读出值为 0。

P1_IN
端口 1 输入寄存器 **(0024_H)** **复位值 : 0000 00XX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										P5	P4	P3	P2	P1	P0
r										rh	rh	rh	rh	rh	rh

域	位	类型	描述
Px **(x = 0-5)**	x	rh	**端口 1 输入位 x** 该位指示输入引脚 P1.x 的电平。 0_B P1.x 的输入电平为 0。 1_B P1.x 的输入电平为 1。
0	[31:6]	r	**保留** 读出值为 0。

P2_IN
端口 2 输入寄存器 **(0024_H)** **复位值 : 0000 0XXX$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**P11**	**P10**	**P9**	**P8**	**P7**	**P6**	**P5**	**P4**	**P3**	**P2**	**P1**	**P0**
r				rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

域	位	类型	描述
Px **(x = 0-11)**	x	rh	**端口 2 输入位 x** 该位指示输入引脚 P2.x 的电平。 0_B P2.x 的输入电平为 0。 1_B P2.x 的输入电平为 1。
0	[31:12]	r	**保留** 读出值为 0。

23.8.7 端口引脚省电寄存器

当 XMC1300 进入深度休眠模式时，引脚省电选项被使能的引脚被设置到一个确定的状态，并且输入施密特触发器以及输出驱动器级被关闭。

注： 对于那些被配置为用于硬件控制的引脚 (Pn_HWSEL.HWx != 00_B)，不要使能引脚省电功能。这样做可能会在器件进入深度休眠模式时导致不确定的引脚行为。

P0_PPS
端口 0 引脚省电寄存器 **(0070_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPS 15	PPS 14	PPS 13	PPS 12	PPS 11	PPS 10	PPS 9	PPS 8	PPS 7	PPS 6	PPS 5	PPS 4	PPS 3	PPS 2	PPS 1	PPS 0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

域	位	类型	描述
PPSx (x = 0-15)	x	rw	**端口 0 引脚省电位 x** 0_B 禁止 P0.x 的引脚省电模式。 1_B 使能 P0.x 的引脚省电模式。
0	[31:16]	r	**保留** 读出值为 0。

P1_PPS
端口 1 引脚省电寄存器 **(0070_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										PPS 5	PPS 4	PPS 3	PPS 2	PPS 1	PPS 0
r										rw	rw	rw	rw	rw	rw

域	位	类型	描述
PPSx (x = 0-5)	x	rw	**端口 1 引脚省电位 x** 0_B 禁止 P1.x 的引脚省电模式。 1_B 使能 P1.x 的引脚省电模式。
0	[31:6]	r	**保留** 读出值为 0。

P2_PPS
端口 2 引脚省电寄存器 **(0070_H)** **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				**PPS 11**	**PPS 10**	**PPS 9**	**PPS 8**	**PPS 7**	**PPS 6**	**PPS 5**	**PPS 4**	**PPS 3**	**PPS 2**	**PPS 1**	**PPS 0**
r				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

域	位	类型	描述
PPSx (x = 0-11)	x	rw	**端口 2 引脚省电位 x** 0_B 禁止 P2.x 的 引脚省电模式。 1_B 使能 P2.x 的 引脚省电模式。
0	[31:12]	r	**保留** 读出值为 0。

深度休眠引脚省电行为

在引脚省电模式被使能的情况下，深度休眠模式下的引脚实际行为由相应引脚的 Pn_IOCRy. PCx 位域控制 (页 23-12) 。 PCx 位域的编码如表 23-8 所示。

表 23-8 深度休眠模式下的 PCx 编码

PCx[4:0]	I/O	正常操作或 PPSx=0_B	深度休眠模式和 PPSx=1_B
$0X000_B$	直接输入	见表 23-5	输入值 = Pn_OUTx
$0X001_B$			输入值 = 0_B; 下拉失效
$0X010_B$			输入值 = 1_B; 上拉失效
$0X011_B$			输入值 = Pn_OUTx, 存储最后采样的输入值
$0X100_B$	反相输入	见 表 23-5	输入值 =$\overline{\text{Pn\_OUTx}}$
$0X101_B$			输入值 = 1_B; 下拉失效
$0X110_B$			输入值 = 0_B; 上拉失效
$0X111_B$			输入值 =$\overline{\text{Pn\_OUTx}}$, 存储最后采样的输入值
$1XXXX_B$	输出	见表 23-5	输出驱动器关闭, 输入施密特触发器关闭 , 不激活拉器件 , 输入值 = Pn_OUTx

23.8.8 端口引脚硬件选择寄存器

有些外设需要硬件直接控制它们的 I/O。由于多个这样的外设 I/O 被映射到相同的引脚，所以使用寄存器 Pn_HWSEL 来选择由哪一个外设拥有该引脚的控制权。

注： Pn_HWSEL.HWx 只是将引脚的硬件控制预先分配给一个特定的外设，但由外设自己决定何时控制该引脚。只要外设不控制由 HWx_EN 给定的引脚，则该引脚的配置仍然由配置寄存器定义，并且它可被用作 GPIO 或其它复用功能。这可能是因为所选择的外设只能控制激活其引脚的一个子集，或者因为外设根本就没被激活。

在应用程序不需要相应的功能并且外设也没有任何措施来选择性地禁止硬件控制的情况下，该机制还可用于禁止对外设的某些引脚的硬件控制。

P0_HWSEL
端口 0 引脚硬件选择寄存器 (0074_H) 复位值 : $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
HW15		HW14		HW13		HW12		HW11		HW10		HW9		HW8	
rw		rw		rw		rw		rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HW7		HW6		HW5		HW4		HW3		HW2		HW1		HW0	
rw		rw		rw		rw		rw		rw		rw		rw	

域	位	类型	描述
HWx (x = 0-15)	[2*x+1: 2*x]	rw	**端口 0 引脚硬件选择位 x** 00_B 仅限软件控制。 01_B HW0 控制通路可以覆盖软件配置。 10_B HW1 控制通路可以覆盖软件配置。 11_B 保留。

P1_HWSEL
端口 1 引脚硬件选择寄存器 **(0074_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				HW5		HW4		HW3		HW2		HW1		HW0	
r				rw		rw		rw		rw		rw		rw	

域	位	类型	描述
HWx (x = 0-5)	[2*x+1: 2*x]	rw	**端口 1 引脚硬件选择位 x** 00_B 仅限软件控制。 01_B HW0 控制通路可以覆盖软件配置。 10_B HW1 控制通路可以覆盖软件配置。 11_B 保留。
0	[31:12]	r	**保留** 读出值为 0 ； 应写入 0。

P2_HWSEL
端口 2 引脚硬件选择寄存器 **(0074_H)** **复位值 : 0000 0000_H**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0								HW11		HW10		HW9		HW8	
r								rw		rw		rw		rw	

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
HW7		HW6		HW5		HW4		HW3		HW2		HW1		HW0	
rw		rw		rw		rw		rw		rw		rw		rw	

域	位	类型	描述
HWx (x = 0-11)	[2*x+1: 2*x]	rw	**端口 2 引脚硬件选择位 x** 00_B 仅限软件控制。 01_B HW0 控制通路可以覆盖软件配置。 10_B HW1 控制通路可以覆盖软件配置。 11_B 保留。
0	[31:24]	r	**保留** 读出值为 0 ； 应写入 0。

23.9 封装引脚概览

下面的通用构件用于描述每个引脚。

表 23-9 封装引脚映射描述

功能	封装 A	封装 B	...	焊盘类型
Px.y	N	N		焊盘类型

下表按 “ 功能 ” 一列排序，从常规的端口引脚 (Px.y) 开始，然后是电源引脚。

“ 功能 ” 列后面的两列以器件所支持的不同封装类型为标题，分别列出被映射了相应功能的封装引脚号。

“ 焊盘类型 ” 指示所使用的焊盘类型：

- STD_INOUT/AN (具有模拟输入的标准双向焊盘)
- 大电流 (大电流双向焊盘)
- STD_IN/AN (具有模拟输入的标准输入焊盘)
- 电源 (供电电源)

焊盘属性的详细信息在数据表中定义。

表 23-10 封装引脚映射

功能	TSSOP 38	TSSOP 16	焊盘类型	备注
P0.0	17	7	STD_INOUT/AN	
P0.1	18	-	STD_INOUT/AN	
P0.2	19	-	STD_INOUT/AN	
P0.3	20	-	STD_INOUT/AN	
P0.4	21	-	STD_INOUT/AN	
P0.5	22	8	STD_INOUT/AN	
P0.6	23	9	STD_INOUT/AN	
P0.7	24	10	STD_INOUT/AN	
P0.8	27	11	STD_INOUT/AN	
P0.9	28	12	STD_INOUT/AN	
P0.10	29	-	STD_INOUT/AN	
P0.11	30	-	STD_INOUT/AN	
P0.12	31	-	STD_INOUT/AN	
P0.13	32	-	STD_INOUT/AN	
P0.14	33	13	STD_INOUT/AN	

表 23-10 封装引脚映射 (续表)

功能	TSSOP 38	TSSOP 16	焊盘类型	备注
P0.15	34	14	STD_INOUT/AN	
P1.0	16	-	大电流	
P1.1	15	-	大电流	
P1.2	14	-	大电流	
P1.3	13	-	大电流	
P1.4	12	-	大电流	
P1.5	11	-	大电流	
P2.0	35	15	STD_INOUT/AN	
P2.1	36	-	STD_INOUT/AN	
P2.2	37	-	STD_IN/AN	
P2.3	38	-	STD_IN/AN	
P2.4	1	-	STD_IN/AN	
P2.5	2	-	STD_IN/AN	
P2.6	3	16	STD_IN/AN	
P2.7	4	1	STD_IN/AN	
P2.8	5	1	STD_IN/AN	
P2.9	6	2	STD_IN/AN	
P2.10	7	3	STD_INOUT/AN	
P2.11	8	4	STD_INOUT/AN	
VSS	9	5	电源	电源 GND，ADC 参考 GND
VDD	10	6	电源	电源 VDD，ADC 参考电压 / ORC 参考电压
VSSP	25	-	电源	I/O 端口地
VDDP	26	-	电源	I/O 端口电源

23.10 端口 I/O 功能

23.10.1 引导模式使用的端口引脚

端口功能可被所选择的引导模式控制。通过 BMI 选择引导模式的类型。表 23-11 列出了用于不同引导模式的端口引脚。

表 23-11 引导模式使用的端口引脚

引脚	引导	引导描述
P0.13	CS(O)	SSC BSL 模式
P0.14	SWDIO_0	调试模式 (SWD)
	SPD_0	调试模式 (SPD)
	RX/TX	ASC BSL 半双工模式
	RX	ASC BSL 全双工模式
	SCLK(O)	SSC BSL 模式
P0.15	SWDCLK_0	调试模式 (SWD)
	TX	ASC BSL 全双工模式
	DATA(I/O)	SSC BSL 模式
P1.2	SWDCLK_1	调试模式 (SWD)
	TX	ASC BSL 全双工模式
P1.3	SWDIO_1	调试模式 (SWD)
	SPD_1	调试模式 (SPD)
	RX/TX	ASC BSL 半双工模式
	RX	ASC BSL 全双工模式

23.10.2 端口 I/O 功能描述

下面的通用构件用于描述每个引脚。

表 23-12 端口 I/O 功能描述

功能	输出			输入		
	ALT1	ALTn	HWO0	HWI0	输入	输入
P0.0		MODA.OUT	MODB.OUT	MODB.INA	MODC.INA	
Pn.y	MODA.OUT				MODA.INA	MODC.INB

Pn.y 是端口引脚的名称，它定义了与其相关的控制和数据位 / 寄存器。当作为 GPIO 时，端口由软件控制。其输入值通过 Pn_IN.y 读取， Pn_OUT 设定其输出值。

可以映射最多七个复用输出功能 (ALT1/2/3/4/5/6/7) 到一个端口引脚，通过 Pn_IOCR.PC 来选择使用该引脚的功能。输出值由相应的模块直接驱动，引脚特性由端口寄存器控制 (在所连接的焊盘的极限值内)。

端口引脚输入可被连接到多个外设。大多数外设都有一个输入多路复用器，以选择不同的可能输入源。

当引脚被配置为输出时，输入通路仍然有效。这就允许将一个输出反馈到片内资源，而不浪费额外的外部引脚。

通过 Pn_HWSEL (23.8.8 节) 可以在不同的硬件之间选择 “ 主机 ” (HWO0/HWI0，HWO1/HWI1)。被选用的外设可以控制引脚。硬件控制功能使相应端口引脚寄存器中的设置无效。

表 23-1 端口 I/O 功能

功能	输入									输出											
	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	HWO0	HWO1	HWI0	HWI1	Input	Input	Input	Input	Input	Input	Input	Input	Input	Input
P0.0	ERU0.PDOUT0		ERU0.GOUT0	CCU40.OUT0	CCU80.OUT00	USIC0_CH0.SELO0	USIC0_CH1.SELO0					BCCU0.TRAPINB	CCU40.IN0C			USIC0_CH0.DX2A	USIC0_CH1.DX2A				
P0.1	ERU0.PDOUT1		ERU0.GOUT1	CCU40.OUT1	CCU80.OUT01	BCCU0.OUT8	SCU.VDROP						CCU40.IN1C								
P0.2	ERU0.PDOUT2		ERU0.GOUT2	CCU40.OUT2	CCU80.OUT02	VADC0.EMUX02	CCU80.OUT10						CCU40.IN2C								
P0.3	ERU0.PDOUT3		ERU0.GOUT3	CCU40.OUT3	CCU80.OUT03	VADC0.EMUX01	CCU80.OUT11						CCU40.IN3C								
P0.4	BCCU0.OUT0			CCU40.OUT1	CCU80.OUT13	VADC0.EMUX00	WWDT.SERVICE_OUT						CCU80.IN0B								
P0.5	BCCU0.OUT1			CCU40.OUT0	CCU80.OUT12	ACMP2.OUT	CCU80.OUT01						CCU80.IN1B								
P0.6	BCCU0.OUT2			CCU40.OUT0	CCU80.OUT11	USIC0_CH1.MCLKOUT	USIC0_CH1.DOUT0						CCU40.IN0B			USIC0_CH1.DX0C					
P0.7	BCCU0.OUT3			CCU40.OUT1	CCU80.OUT10	USIC0_CH0.SCLKOUT	USIC0_CH1.DOUT0						CCU40.IN1B			USIC0_CH0.DX1C	USIC0_CH1.DX0D	USIC0_CH1.DX1C			
P0.8	BCCU0.OUT4			CCU40.OUT2	CCU80.OUT20	USIC0_CH0.SCLKOUT	USIC0_CH1.SCLKOUT						CCU40.IN2B			USIC0_CH0.DX1B	USIC0_CH1.DX1B				
P0.9	BCCU0.OUT5			CCU40.OUT3	CCU80.OUT21	USIC0_CH0.SELO0	USIC0_CH1.SELO0						CCU40.IN3B			USIC0_CH0.DX2B	USIC0_CH1.DX2B				
P0.10	BCCU0.OUT6			ACMP0.OUT	CCU80.OUT22	USIC0_CH0.SELO1	USIC0_CH1.SELO1						CCU80.IN2B			USIC0_CH0.DX2C	USIC0_CH1.DX2C				
P0.11	BCCU0.OUT7			USIC0_CH0.MCLKOUT	CCU80.OUT23	USIC0_CH0.SELO2	USIC0_CH1.SELO2									USIC0_CH0.DX2D	USIC0_CH1.DX2D				
P0.12	BCCU0.OUT6				CCU80.OUT33	USIC0_CH0.SELO3	CCU80.OUT20					BCCU0.TRAPINA	CCU40.IN0A	CCU40.IN1A	CCU40.IN2A	CCU40.IN3A	CCU80.IN0A	CCU80.IN1A	CCU80.IN2A	CCU80.IN3A	USIC0_CH0.DX2E
P0.13	WWDT.SERVICE_OUT				CCU80.OUT32	USIC0_CH0.SELO4	CCU80.OUT21						CCU80.IN3B	POSIF0.IN0B		USIC0_CH0.DX2F					
P0.14	BCCU0.OUT7				CCU80.OUT31	USIC0_CH0.DOUT0	USIC0_CH0.SCLKOUT							POSIF0.IN1B		USIC0_CH0.DX0A	USIC0_CH0.DX1A				
P0.15	BCCU0.OUT8				CCU80.OUT30	USIC0_CH0.DOUT0	USIC0_CH1.MCLKOUT							POSIF0.IN2B		USIC0_CH0.DX0B					
P1.0	BCCU0.OUT0	CCU40.OUT0			CCU80.OUT00	ACMP1.OUT	USIC0_CH0.DOUT0		USIC0_CH0.DOUT0		USIC0_CH0.HWIN0			POSIF0.IN2A		USIC0_CH0.DX0C					
P1.1	VADC0.EMUX00	CCU40.OUT1			CCU80.OUT01	USIC0_CH0.DOUT0	USIC0_CH1.SELO0		USIC0_CH0.DOUT1		USIC0_CH0.HWIN1			POSIF0.IN1A		USIC0_CH0.DX0D	USIC0_CH0.DX1D	USIC0_CH1.DX2E			
P1.2	VADC0.EMUX01	CCU40.OUT2			CCU80.OUT10	ACMP2.OUT	USIC0_CH1.DOUT0		USIC0_CH0.DOUT2		USIC0_CH0.HWIN2			POSIF0.IN0A		USIC0_CH1.DX0B					
P1.3	VADC0.EMUX02	CCU40.OUT3			CCU80.OUT11	USIC0_CH1.SCLKOUT	USIC0_CH1.DOUT0		USIC0_CH0.DOUT3		USIC0_CH0.HWIN3					USIC0_CH1.DX0A	USIC0_CH1.DX1A				
P1.4	VADC0.EMUX10	USIC0_CH1.SCLKOUT			CCU80.OUT20	USIC0_CH0.SELO0	USIC0_CH1.SELO1									USIC0_CH0.DX5E	USIC0_CH1.DX5E				
P1.5	VADC0.EMUX11	USIC0_CH0.DOUT0		BCCU0.OUT1	CCU80.OUT21	USIC0_CH0.SELO1	USIC0_CH1.SELO2									USIC0_CH1.DX5F					

表 23-1 端口 I/O 功能 (续表)

功能	输入									输出											
	ALT1	ALT2	ALT3	ALT4	ALT5	ALT6	ALT7	HWO0	HWO1	HWI0	HWI1	Input	Input	Input	Input	Input	Input	Input	Input	Input	Input
P2.0	ERU0.PDOUT3	CCU40.OUT0	ERU0.GOUT3		CCU80.OUT20	USIC0_CH0.DOUT0	USIC0_CH0.SCLKOUT						VADC0.G0CH5		ERU0.0B0	USIC0_CH0.DX0E	USIC0_CH0.DX1E	USIC0_CH1.DX2F			
P2.1	ERU0.PDOUT2	CCU40.OUT1	ERU0.GOUT2		CCU80.OUT21	USIC0_CH0.DOUT0	USIC0_CH1.SCLKOUT					ACMP2.INP	VADC0.G0CH6		ERU0.1B0	USIC0_CH0.DX0F	USIC0_CH1.DX3A	USIC0_CH1.DX4A			
P2.2												ACMP2.INN	VADC0.G0CH7		ERU0.0B1	USIC0_CH0.DX3A	USIC0_CH0.DX4A	USIC0_CH1.DX5A	ORC0.AIN		
P2.3													VADC0.G1CH5		ERU0.1B1	USIC0_CH0.DX5B	USIC0_CH1.DX3C	USIC0_CH1.DX4C	ORC1.AIN		
P2.4													VADC0.G1CH6		ERU0.0A1	USIC0_CH0.DX3B	USIC0_CH0.DX4B	USIC0_CH1.DX5B	ORC2.AIN		
P2.5													VADC0.G1CH7		ERU0.1A1	USIC0_CH0.DX5D	USIC0_CH1.DX3E	USIC0_CH1.DX4E	ORC3.AIN		
P2.6												ACMP1.INN	VADC0.G0CH0		ERU0.2A1	USIC0_CH0.DX3E	USIC0_CH0.DX4E	USIC0_CH1.DX5D	ORC4.AIN		
P2.7												ACMP1.INP	VADC0.G1CH1		ERU0.3A1	USIC0_CH0.DX5C	USIC0_CH1.DX3D	USIC0_CH1.DX4D	ORC5.AIN		
P2.8												ACMP0.INN	VADC0.G0CH1	VADC0.G1CH0	ERU0.3B1	USIC0_CH0.DX3D	USIC0_CH0.DX4D	USIC0_CH1.DX5C	ORC6.AIN		
P2.9												ACMP0.INP	VADC0.G0CH2	VADC0.G1CH4	ERU0.3B0	USIC0_CH0.DX5A	USIC0_CH1.DX3B	USIC0_CH1.DX4B	ORC7.AIN		
P2.10	ERU0.PDOUT1	CCU40.OUT2	ERU0.GOUT1		CCU80.OUT30	ACMP0.OUT	USIC0_CH1.DOUT0						VADC0.G0CH3	VADC0.G1CH2	ERU0.2B0	USIC0_CH0.DX3C	USIC0_CH0.DX4C	USIC0_CH1.DX0F			
P2.11	ERU0.PDOUT0	CCU40.OUT3	ERU0.GOUT0		CCU80.OUT31	USIC0_CH1.SCLKOUT	USIC0_CH1.DOUT0					ACMP.REF	VADC0.G0CH4	VADC0.G1CH3	ERU0.2B1	USIC0_CH1.DX0E	USIC0_CH1.DX1E				

引导和启动、引导加载程序和用户例程

24 引导和启动

XMC1300 的启动序列是在用户应用软件获得系统控制权之前发生的一个过程，它包括两个主要阶段 (见 图 24-1)，每个阶段又分为几个有明显区别的步骤。

硬件控制的启动阶段

在微控制器上电后，硬件控制的启动阶段自动开始执行。这部分是通用的，它确保完成大多数应用的基本常用配置。为了在控制权被移交给用户软件之前使微控制器能可靠启动，硬件设置需要保证满足数据表中规定的要求。引导代码的执行序列被视为启动序列中硬件控制阶段的一部分。

有关设置要求的详细信息，请参见数据表。

软件控制的启动阶段

软件控制的启动阶段是应用于用户软件的专用配置的一部分。它包括对微控制器在应用程序环境正常运行来说至关重要的几个步骤，它也可能包括一些可选的配置操作，以提高应用程序环境下系统的性能和稳定性。

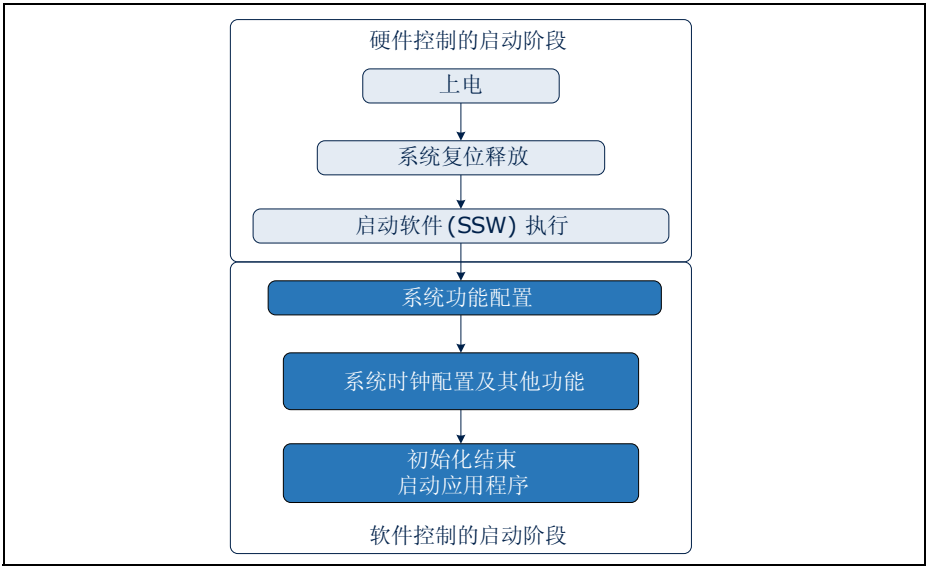


图 24-1 启动序列

24.1 启动序列和系统相关性

下面各小节对启动序列进行详细描述。

24.1.1 上电

微控制器的上电过程通过施加 V_{DDP} 电源 (有关电源要求的详细信息，请参见数据表) 来执行。一旦 V_{DDP} 达到某个阈值， EVR 会自动产生内部的内核电压 V_{DDC}。

24.1.2 系统复位释放

内部上电复位的产生基于电源和内核电压有效。当 V_{DDP} 和 V_{DDC} 达到一个稳定的阈值电平后 ，上电复位开始。接下来，在片内振荡器产生一个稳定的时钟输出后，系统复位被自动释放 , SSW 代码开始执行。

系统复位可以由多种复位源触发，如软件控制的 CPU 复位寄存器或看门狗超时触发的复位。有关复位控制的详细信息，请参见 SCU 一章中的复位控制单元 (RCU) 一节。

最后一次复位的原因被自动保存到 SCU_RSTSTAT 寄存器中，用户软件可通过检查该寄存器来确定系统的状态并用于调试目的。在每次启动后，应使用 SCU_RSTCLR 寄存器对 SCU_RSTSTAT 寄存器中的复位状态复位，以保证下次复位后可得到正确的复位源指示。

复位释放后，MCLK 和 PCLK 运行在 8MHz ，除 CPU、存储器和端口以外的大多数外设的时钟被禁止。建议禁止未使用模块的时钟，以降低功耗。

24.1.3 启动软件 (SSW) 执行

复位解除后，CPU 开始在 ROM 存储器中执行 SSW 代码。为了指示 SSW 执行的开始，P0.8 中的上拉器件被使能。上拉器件在复位期间被禁止。

由 SSW 完成的另一个重要任务是读取存储在 Flash 中的引导模式索引 (BMI)，并决定由用户选择的启动模式 (例如用户生产模式和引导加载程序) 。关于各种启动模式和 BMI 处理的详细信息，请参见 **24.2 节**。

在 SSW 执行期间，为了处理初始硬故障错误，SSW 在 SRAM 单元 $2000'000C_H$ 安放了一条 “ 跳转到自己 ” 的指令作为临时的硬故障处理程序，直到用户代码加载其自己的硬故障处理程序。这种加载只能在主复位之后进行。

在 SSW 执行期间，$2000'00C0_H$ 到 $2000'0200_H$ 之间的 SRAM 区域的被保留供 XMC1300 的 SSW 使用，因此用户软件不应在该区域存储数据，在 (非上电) 复位期间该区域必须被保留。

用于执行 SSW 和启动用户代码的 MCLK 和 PCLK 频率可由用户配置。另外，有些外设的时钟也可以通过 SSW 来使能 (默认为禁止)。详细描述见 **24.1.3.1 节**。

24.1.3.1 通过 SSW 处理时钟系统

XMC1300 的 SSW 能改变器件的时钟设置，这就允许在启动期间灵活地调整器件性能，例如速度与功耗的优化。

SSW 对时钟的处理是用户可配置的，通过在 Flash 中存放相应的值来实现。为此，在 Flash 中分配了两个字地址 CLK_VAL1 和 CLK_VAL2 (参见表 24-1)，这两个地址中的值由 SSW 按如下方式处理：

- 如果 CLK_VAL1[31] 等于 0 - CLK_VAL1[19:0] 位被加载到 SCU_CLKCR[19:0] 寄存器 - 这样即可配置时钟分频器并选择
- 如果 CLK_VAL2[31] 等于 0 - CLK_VAL2[10:0] 位被加载到 SCU_CGATCLR0[10:0] - 这会使能所选外设的时钟

在有些情况下，CLK_VALx 单元不能由用户编程 - 像在器件交付状态 - 它的位 [31] 等于 1 并且不能加载相应的寄存器 。

器件频率变化限制

为了避免发生大的功耗突升 / 下降，在启动期间要被从 CLK_VAL1 加载到 CLKCR.IDIV 的用户配置值被限制在 1~16 的范围内，重新配置后 MCLK 的可能范围从初始的 8MHz 变为 2..32 MHz ，即 MCLK 的频率可以被提高或降低不超过 4 倍 (四舍五入，忽略潜在的 FDIV 影响)。

在 CLK_VAL1 (参见表 24-1) 包含的 IDIV 值位于 1~16 范围之外的情况下：

- 如果 IDIV = 0 - SSW 设置 IDIV = 1
- 如果 IDIV > 16 - SSW 设置 IDIV = 16

24.1.4 特殊系统功能配置为用户代码初始化的一部分

可能需要使用一些特殊系统功能来执行提高系统的稳定性和健壮性的一些操作。在实际的用户应用程序开始前，需要初始化下面的特殊系统功能或模块，这是用户代码的初始化部分要完成的：

- 起始地址和初始堆栈指针值
- 设置中断处理程序
- 电源电压掉电检测
- 看门狗定时器 (WDT)

起始地址和初始堆栈指针值

在 表 24-1 中的起始地址和初始堆栈指针值需要由用户定义。这些值可以和用户代码一起被下载到 Flash 存储器中。

设置中断处理程序

向量表被重新映射到 SRAM 中，该映射基于 CPU 一章中描述的重映射向量表。中断服务程序可以被存放在这些 SRAM 地址，用户还可以使用一条转移指令转向存放在其他存储器位置的中断服务程序。更详细的信息请参见 CPU 一章。

V_{DDP} 掉电检测

V_{DDP} 掉电检测机制允许主动监视电源电压，并在电压电平低于某个编程阈值的情况下做出纠正操作。如果检测到一个编程设置的条件，掉电检测机制会标记一个中断请求。更详细的信息请参见 SCU 一章中的电源控制单元 (PCU) 一节。

看门狗定时器

看门狗定时器需要选择并激活一个时钟源。强烈建议使用可靠的时钟源，为了确保在发生系统故障的情况下纠正动作可将微控制器带入一个安全的运行状态，该时钟源最好独立于系统时钟源。在 XMC1300 中，默认的 WDT 时钟为待机时钟。更详细的信息请参见 WDT 一章。关于 WDT 模块配置的详细信息，请参见 WDT 一章节中的“初始化和控制序列”一节。

24.1.5 时钟系统和其他功能配置

以下功能是可用的，可能需要在用户代码中配置：

- 时钟系统配置
- 系统复位配置
- 调试挂起配置

时钟系统配置

MCLK 和 PCLK 频率可以被配置为不同于 **24.1.3.1 节**所描述的 SSW 执行期间的用户设置。SCU_CRCLK.IDIV/FDIV 位用于编程目标时钟频率。MCLK 和 PCLK 之间的比率可以通过位 SCU_CRCLK.PCLKSEL 选择。另外，一些外设的时钟可以分别通过 SCU_CGATCLR0/SCU_CGATSET0 寄存器来使能 / 禁止。建议使能振荡器看门狗来实现时钟丢失检测。

系统复位配置

有多种事件可用来触发系统复位，如 Flash ECC 错误或 SRAM 的奇偶校验错误等，可以通过 SCU_RSTCON 寄存器来配置触发复位的事件。更详细的信息请参见 SCU 一章中的复位控制单元 (RCU) 一节。

调试挂起配置

XMC1300 器件支持外设的挂起功能，如果 CPU 的程序执行被调试器停止，例如通过断点或 C_HALT 命令。这就允许调试整个微控制器的关键状态。必须根据应用要求和调试方法来在外设本地使能或禁止挂起功能。更详细的信息请参见调试系统一章。

24.2 启动模式

启动软件 (简称 SSW) 是任何器件离开复位状态后由 CPU 执行的第一个软件。

SSW 被存储在 ROM 存储器中，但其执行受其它“灵活”因素的影响如下：

- 触发 SSW 执行的事件类型
- 由用户选择的启动配置，即所谓的引导模式索引 (简称 BMI)

24.2.1 XMC1300 的启动模式

XMC1300 的启动模式选择由 SSW 在复位后完成。这种选择基于存储在 Flash 配置扇区 0 (CS0) 的**引导模式索引 (BMI)**，这意味着没有器件引脚用于配置目的。编程一个新的 BMI，用户需要调用如 “BSL 和用户例程 ” 一章所述的用户函数 “ 请求 BMI 安装 ”。SSW 选择和处理BMI的描述见 **24.2.3节**。在器件的交付状态，默认启动模式为ASC BSL模式。

器件所支持的启动模式概述如下。

24.2.1.1 用户生产模式

在该模式，用户代码从 Flash 存储器启动，不可能进行调试。

第一条用户指令的地址 - SSW 结束时要跳往的地址 - 为 Flash 单元 $1000'1004_H$。

SSW 不检查目标地址是否位于用户 Flash 内部。因此，如果目标地址位于 Flash 外部，将会产生一个硬故障，而程序执行会跳转到 SRAM 中的异常处理程序 (主复位后由 SSW 载入一条死循环指令)。

24.2.1.2 使能了调试功能的用户模式

用户代码从Flash存储器开始执行，第一个地址的确定同**用户生产模式**(见 **24.2.1.1 节**)。

调试接口要根据 **引导模式索引 (BMI)** 的值来配置并使能。

24.2.1.3 使能了调试功能和复位后暂停 (HAR) 的用户模式

与上述两种模式一样，用户代码从 Flash 存储器开始执行。

调试接口要根据 BMI 的值来配置并使能，在从 SSW 退出后到第一条用户指令之前这段事件， CPU 被停机 (有关处理的详细信息请参见 **24.2.3.2 节**)。

24.2.1.4 标准引导加载程序模式

在该模式 (简称标准 BSL)：

- 用户代码被下载到 SRAM
- 在 SSW 结束后程序执行跳转到 SRAM
- 不允许调试

XMC1300 支持的标准 BSL 模式使用 USIC0 模块与主机进行通信：

- ASC - 使用 USIC0 模块的通道 0 或 1 (由固件自动检测) 和 ASC (UART) 协议完成下载。
  这是默认的启动模式 - 由器件交付状态的 BMI 值选择。
- SSC - 使用 USIC0 的通道 0 和 SSC (SPI) 标准协议完成下载 。
  在该模式，需要使用 SPI 兼容的串行 EEPROM 作为从器件连接到 XMC1300。

标准 BSL 例程的功能在 BSL 一章中描述。

24.2.1.5 具有超时功能的引导加载程序模式

这些模式使用标准的 BSL 例程用于下载，但功能不同：

- SSW 首先检查是否连接 / 激活了一个外部器件，这意味着 SSW 等待接收来自主机的起始和首部字节 - 无论通过 USIC 通道 0 还是 1 - 超时时间取自 BMI.BLSTO (参见 **24.2.2 节**)：
  - 如果是
    * 用户代码被下载到 SRAM
    * 在 SSW 结束后程序执行跳转到 SRAM
  - 如果不是 - 在 SSW 结束后程序执行跳转到 Flash ， 与**用户生产模式**一样。
- 在该模式下不允许调试，这与代码从 Flash 还是 SRAM 开始执行无关。

XMC1300 支持两种具有超时功能的 BSL 模式，这两种模式都使用 USIC0 模块与主机进行通信。除了接口选择不同以外，所执行的检查也与上述过程不同：

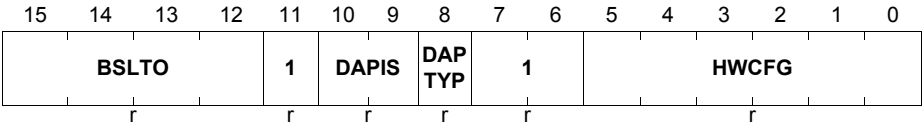
- ASC 模式
  - 在 ASC (UART) 模式， SSW 配置 USIC0 通道 0 和 1
  - SSW 等待接收来自主机的起始和首部字节 - 无论是通过通道 0 还是通道 1 - 超时时间取自 BMI.BLSTO (参见 **24.2.2 节**)
  - 如果起始 + 首部字节被接收 - 会进一步执行标准 ASC BSL 来下载程序，并从 SRAM 开始执行用户代码；否则将从 Flash 执行开始执行。
- SSC 模式
  - 在 SSC (SPI) 模式， SSW 配置 USIC0 的通道 0 作为主器件。
  - SSW 初始化与 SPI 兼容串行 EEPROM 的通信，该 EEPROM 被作为从器件与主器件连接 。
  - 如果在 SSW 发送读命令后接收的数据符合预期的 (串行 EEPROM) 协议 - 会进一步执行标准SSC BSL来下载程序并从SRAM下开始执行用户代码；否则将从 Flash 执行开始执行。

注： 如果 BMI 中的配置为 BSLTO=0 - SSW 不执行任何 BSL 检查，但直接使用用户生产模式。

24.2.2 引导模式索引 (BMI)

引导模式索引是存储在 Flash 中的 2 字节值 (参见 **表 24-1**)，它包含关于启动模式和器件的调试配置信息。除了位于 $1000'0E00_H$ 地址的 BMI 值以外，还在 $1000'0E10_H$ 地址存放了 BMI 的反相值，该值用于检查 BMI 的正确性。

BMI
引导模式索引 **CS0 偏移 $0E00_H$** **交付状态: $FFC0_H$**



域	位	类型	描述
HWCFG	[5:0]	r	**启动模式选择：** 000000_BASC 引导加载程序模式 (ASC_BSL) 000001_B用户生产模式 (UPM) 000011_B使能了调试功能的用户模式 (UMD) 000111_B使能了调试功能和 HAR 的用户模式 (UMHAR) 001000_BSSC 引导加载程序模式 (SSC_BSL) 010000_B超时 ASC BSL 模式 (ASC_BSLTO) 011000_B超时 SSC BSL 模式 (SSC_BSLTO) 101010_B安全引导加载程序模式 (SBSL) 1) 其它 未定义 2)
DAPTYP	8	r	**DAP 类型选择** 编码同 SCU_DAPCON.DAPTYP
DAPDIS	[10:9]	r	**SWD/SPD 输入 / 输出选择** 编码同 SCU_DAPCON.DAPDIS
BSLTO	[15:12]	r	**ASC BSL 超时值** 超时时间为 BSLTO*2664000 个 MCLK 周期， 支持的超时范围为 0.3-5s (333– 4995ms)
1	[7:6], 11	r	**保留，必须被编程为 1**

1) 仅限于支持该模式的器件版本。
2) 注意：加载其中一个值会导致在下一次复位时器件的交付状态被恢复 - 请参见 **24.2.3 节**。

24.2.3 启动模式选择

XMC1300 中的启动模式要根据从 Flash 读取的引导模式索引 (BMI) 值来选择和处理 - 参见 24.2.1 节。

对 BMI 的评估可能会导致：

- 采用用户模式 - SSW 会在其结束时跳转到用户 Flash，如果满足下面条件，则从地址 $1000'1004_H$ 开始执行：
  - 在 BMI 中选择了一种用户模式
  - 在 BMI 中选择了超时 ASC BSL 模式，但在所选择的时间帧内未收到有效的起始 + 首部字节
- 采用引导加载程序模式 - 执行 ASC BSL 后跳转到 SRAM 地址 $2000'0200_H$
- 采用安全引导加载程序(SBSL)模式 - 如果在BMI 中选择了SBSL并且器件支持该模式
- 执行 (无限) 循环 - 在选择了无超时 ASC BSL 模式的情况下，等待 ASC 接收有效的起始 + 首部字节
- 擦除整个用户 Flash 并加载 ASC BSL (或 SBSL，如果支持) 模式作为新的 BMI 值 - 在 BMI 值无效或选择的安全 BSL 不为器件所支持时。之后主复位被触发，器件根据新的 BMI 以另一种模式启动。

24.2.3.1 通过 SSW 处理 BMI

SSW 部分要在用户函数 (位于 ROM 中) “ 请求 BMI 安装 ” (在 BSL 和用户例程一章中描述) 已被调用并触发了一次系统复位后才能执行。因此 SSW 执行 BMI 处理的条件是：

- 最后一次复位是由 CPU 请求的系统复位
- SSW0 寄存器的两个半字互为取反

如果上述所有条件都为真，则 SSW 将 SSW0[15:0] 的内容视为要被加载的新 BMI 值，并执行下面的操作：

- 检查当前的 BMI (位于 CS0 中) 是否为用户生产型并且新值是否为非用户生产型：
  - 如果是 - 擦除整个用户 Flash，并加载 ASC_BSL (或 SBSL，如果支持) 模式作为新的 BMI 值
  - 如果不是 - 从 SSW0[15:0] 加载新的 BMI 值
- 向 SSW0 加载全 0，指示在下一次复位后不执行 BMI 编程
- 请求触发主复位

最后一步完成后，一次主复位被触发，下一次 SSW 执行会使用加载到 CS0 的新 BMI 值启动。

24.2.3.2 调试系统处理

如果调试系统必须被使能 - 则在引导模式索引中选择带调试支持的启动模式 (参见 24.2.2 节) - SSW 执行下述操作：

- 根据 BMI 值配置调试接口
- 使能调试接口

- 如果主复位后在 BMI 中选择了复位请求后停止 (UMHAR) 这一启动模式 – 进入一个无限 NOP（空操作）循环。从此刻起，一个外部工具（调试器）可以接管对器件的控制权，特别是：
  – 如果 / 当一个调试器连接到器件时，它可以停止 CPU 并检查 BMI 和 / 或内核的程序计数器 (PC) 寄存器。
  – 如果选择的启动模式为 HAR (这可由 BMI 值识别，参见 **24.2.2 节**) PC 值将指向内部 ROM。然后调试器可以操作 PC，并按要求启动用户代码 - 默认的起始地址 (无 HAR) 根据用户 Flash 中的 $1000'1004_H$ 单元的内容决定 (参见 **表 24-1**)。

24.3 Flash 中用于 SSW 和 用户 SW 的数据

表 24-1 列出了位于 XMC1300 的 Flash 中与 SSW 执行相关并可为用户软件使用的数据。

表 24-1 XMC1300 中用于 SSW 和用户软件的 Flash 数据

地址	长度	功能	目标单元
		启动模式选择：	
$1000'0E00_H$	2 B	引导模式索引 (BMI)	---
$1000'0E10_H$	2 B	反相 BMI	---
		芯片标识：	
$1000'0F00_H$	4 B	芯片 ID	SCU_IDCHIP
$1000'0F04_H$	28 B	芯片型号标识号	---
$1000'0FF0_H$	16 B	唯一的芯片 ID	---
		与温度传感器相关的数据：	
$1000'0F20_H$	2 B	ANA_TSE_k1	---
$1000'0F22_H$	2 B	ANA_TSE_k3	---
$1000'0F24_H$	4 B	ANA_TSE_k2	---
$1000'0F28_H$	2 B	ANATSEMON_min	---
$1000'0F2A_H$	2 B	ANATSEMON_max	---
		DCO 校准数据：	
$1000'0F40_H$	2 B	DCO_ADJL_RT 在室温下测量的频率低调整值 (5 LS 位)	---
$1000'0F42_H$	2 B	DCO_ADJL_HT 在高温下测量的频率低调整值 (5 LS 位)	---
		与应用软件相关的数据：	
$1000'1000_H$	4 B	SSW 完成后的初始堆栈指针值	SP_main

表 24-1 XMC1300 中用于 SSW 和用户软件的 Flash 数据 (续表)

地址	长度	功能	目标单元
$1000'1004_H$	4 B	在用户模式下 SSW 完成后的起始地址	PC
		与时钟系统相关的数据：	
$1000'1010_H$	4 B	时钟配置值 CLK_VAL1	如果位 [31] 等于 0： 位 [19:0] 被载入 SCU_CLKCR[19:0]
$1000'1014_H$	4 B	时钟门控配置值 CLK_VAL2	如果位 [31] 等于 0： 位 [10:0] 被载入 SCU_CGATCLR0[10:0]

25 引导加载程序 (BSL) 和用户例程

本章介绍位于 ROM 存储器中的引导加载程序 (BSL) 和用户可访问的例程。

进入 ASC (UART) BSL (25.1 节) 和 SSC BSL (25.2 节) 模式所使用的 BMI 设置在 “ 引导和启动”一章种描述。BSL模式的主要用途是允许简单、快速地编程/擦除Flash存储器。

25.3 节 描述了 5 个可被应用软件调用的用户可访问例程。这些例程位于 ROM 存储器中。

25.1 ASC (UART) 引导加载程序

XMC1300 中的 ASC (UART) 标准引导加载程序 (ASC BSL) 使用 USIC0 模块来下载代码 / 数据到起始地址为 2000’0200H 的 SRAM 中。

在最后一个代码 / 数据字节被接收和存储后，启动软件从地址 20000’0200H 开始执行用户代码。

25.1.1 引脚使用

XMC1300 中的 ASC BSL 由一个单一的 BMI 值选定，但是它允许通过多个不同的引脚、模式和 USIC0 通道下载用户代码 / 数据：

- 通道 0
  - 全双工模式：RxD 位于 P0.14，TxD 位于 P0.15
  - 半双工模式：RxD/TxD 位于 P0.14
- 通道 1
  - 全双工模式：RxD 位于 P1.3 ，TxD 位于 P1.2
  - 半双工模式：RxD/TxD 位于 P1.3

注： 上述引脚设置也适合 SWD/SPD (调试) 接口的所有引脚分配。

25.1.2 ASC BSL 执行流程

完整的 ASC 引导加载程序由两部分组成，描述如下。

25.1.2.1 ASC BSL 进入检查序列

只有在通过两个通道中的一个通道接收到一个起始 (0) 和首部 (见下面的描述) 字节后才能启动下载。这两个通道都处于活动状态 (“ 监听 ”)，直到该例程本身做出如下选择：

- 通道选择 - 基于接收到第一个接收起始+首部字节的事实RxD 引脚 (见上述引脚分配)
- 全 / 半双工选择 - 基于接收的首部字节

注： 与主机交换的首部 / 应答字节值的定义 - 参见 25.1.3 节。

ASC 引导加载程序的功能包括：

1. 使能和配置 USIC0 的通信通道 0 和 1 ，如下：
   a) ASC 模式、 8 个数据位、 1 个停止位、无奇偶校验
2. 配置用于测量的 ASC 时钟产生电路
3. 对 USIC0 的两个通道依次继续执行以下序列

a) 检查是否收到起始 0 字节
如果收到 - 则
b) 根据从起始字节测得的波特率来配置用于正常操作的 ASC 时钟产生电路，然后
c) 检查是否收到首部字节
如果收到 - 则
d) 检查收到的首部字节
* 如果等于 BSL_ASC_F 或 BSL_ENC_F - 选择全双工模式 - 继续第 4 步。
* 如果等于 BSL_ASC_H 或 BSL_ENC_H - 选择半双工模式 - 继续第 4 步 。
* 其它值 - 跳转到另一通道序列

4. 选择当前 (Cy) 通道用于后续处理；恢复未选通道 C(y-1) 的默认配置
5. 对于所选通道和模式 (全 / 半双工) :
a) 配置 TxD 引脚和 ASC 发送器
b) 调整 (如果需要) RxD 引脚设置
c) 使能发送，单次模式
6. 检查收到的首部字节为哪个字节：
a) BSL_ASC_F 或 BSL_ASC_H - 执行标准 BSL 进入序列，后续通信使用检测到的波特率 - 继续执行 **ASC BSL 下载序列** (见 **25.1.2.2 节**)
b) BSL_ENC_F 或 BSL_ENC_H - 在 **ASC BSL 下载序列** 期间，通信过程使用的波特率可能与最初检测的波特率不同 (例如更高) - 继续执行**增强的 ASC BSL 进入序列** (见下面的描述)

上述检查序列的执行条件：

- 在无超时 ASC BSL 模式 - 直至收到有效的起始 + 首部字节
- 否则 - 只等待一段时间，该时间由 BMI.BSLTO 设置

注： 如果选择了带超时功能的模式，但 BMI.BSLTO = 0 - 用户模式直接从 Flash 启动。

增强的 ASC BSL 进入序列

这是对标准 **ASC BSL 进入检查序列**的扩展，在通信通道需要更高的波特率时使用。
该序列包括以下步骤：

1. 发送 BSL_ENC_ID 来确认初始波特率的建立和进入增强的 ASC BSL 模式
2. 发送来自 USIC0_CHy_BRG 寄存器的当前 PDIV 分频器的值 - 这是一个 10 位值，以 2 字节发送 (最高有效字节在先)
3. 从主机接收一个 10 位值，将其写到 USIC0_CHy_FDR 寄存器的 STEP 位域
4. 相应地配置 USIC 波特率发生器
5. 对于所选通道和模式 (全 / 半双工):
a) 配置 TxD 引脚和 ASC 发送器
b) 调整 (如果需要) RxD 引脚设置
c) 使能发送，单次模式
6. 发送 BSL_BR_OK 来确认新波特率的建立
7. 接收来自主机的 BSL_BR_OK，它指示通信通道已经成功建立
- 如果未收到确认或收到一个错误的确认值，将触发一次器件软件复位

图 25-1 展示了波特率配置过程。

对主机的要求

为使主机支持增强的 ASC BSL 模式，主机还必须：

- 从 XMC1300 器件接收位于 USIC0_CHy_BRG 寄存器中的 10 位当前 PDIV 值
- 基于接收的 PDIV 值通过下面的公式计算器件运行所使用的 MCLK：

(25.1)

$$MCLK = ????? \times (PDIV + 1) \times 8$$

- 使用给定的 MCLK 选择一个为增强型 ASC BSL 所支持的新波特率 (见 **表 25-1**)
- 计算比例因子，即新波特率与初始波特率的比率。**表 25-2** 中给出了一些例子。比例因子被舍入到最接近的整数，以满足 +/- 3% 的频率偏差。

表 25-1　　支持的波特率

MCLK (MHz)	ASC BSL 支持 的标准波特率 (kHz)		增强型 ASC BSL 支持的最大波特率 (MHz)
	最小值	最大值	
8	1.2	28.8	0.999
16	2.4	57.6	1.998
32	4.8	115.2	3.996

表 25-2　　比例因子示例

初始标准波特率 (kHz) ---A	目标更高 波特率 (kHz) ---B	被舍入为最近整数的比例因子 (B/A)
9.6	1500	156
19.2	1500	78
57.6	1500	26
115.2	1500	13

- 通过下面的公式计算要写到 USIC0_CHy_FDR 寄存器的 STEP 位域中的 10 位值：

(25.2)

$$STEP\ (????????) = \frac{1024 \times ????}{PDIV + 1}$$

- 发送这个 10 位 STEP 值给器件
- 等待直到接收到来自器件的 BSL_BR_OK
- 向器件返回 BSL_BR_OK 作为回应

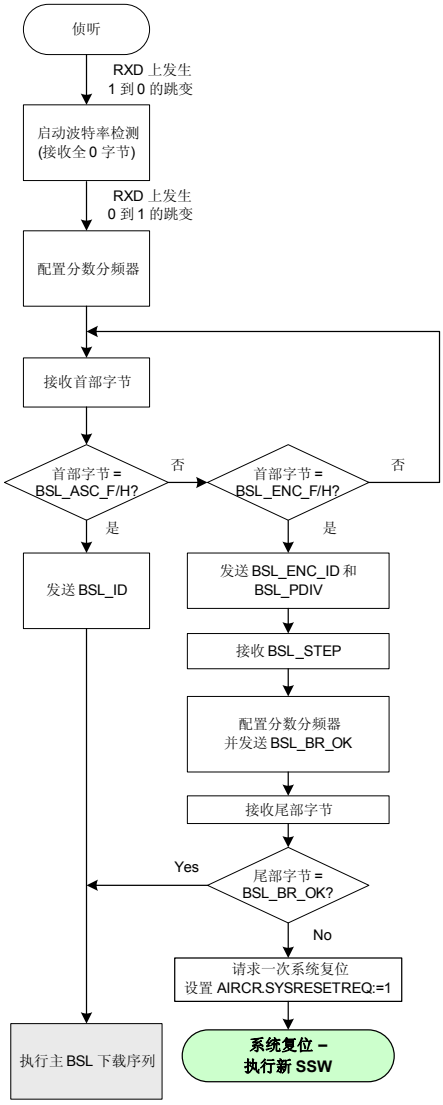


图 25-1　　XMC1300 ASC BSL 进入期间的波特率配置序列

25.1.2.2 ASC BSL 下载序列

在已经检测到 / 配置好波特率并且选定了通道 / 模式 (全 / 半双工) 后，ASC BSL 等待主机发送描述应用程序长度的 4 个字节 (见 图 25-2)。最低有效字节最先接收。

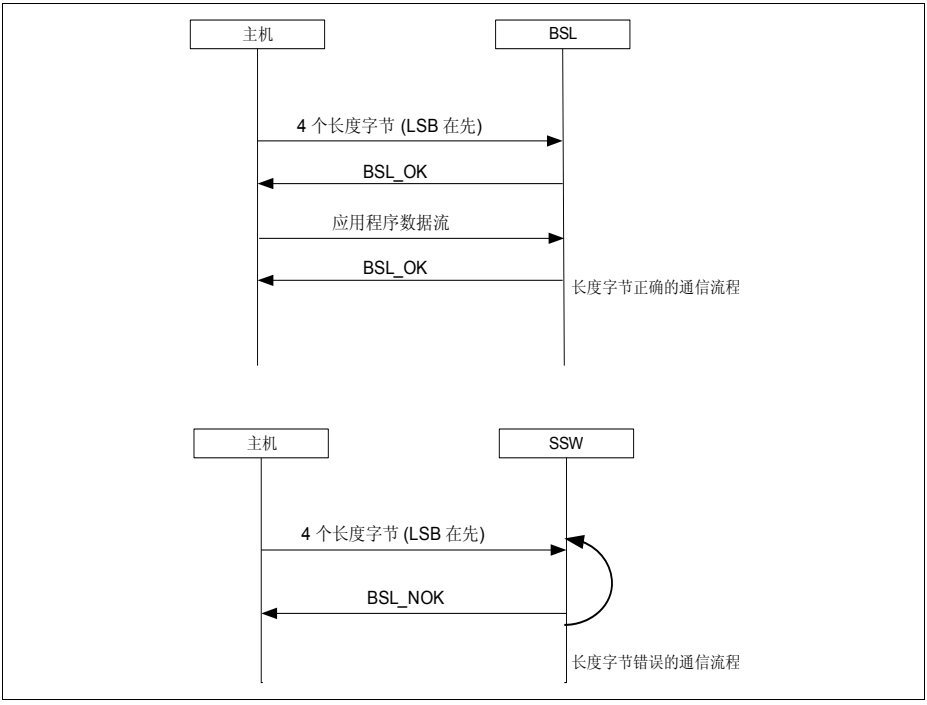


图 25-2 XMC1300 标准 ASC BSL: 应用程序下载协议

如果应用程序长度经 SSW 检查后确定为正确，SSW 会发送一个 BSL_OK 字节给主机，接下来主机发送属于应用程序字节流。在接收完应用程序字节流后，SSW 通过发送一个最终的 OK 字节来终止协议，然后让出控制权给刚下载的应用程序。

如果发现应用程序长度不正确 (应用程序长度大于器件 SRAM 的大小)，SSW 向主机发送一个 BSL_NOK 字节并重新开始等待长度字节。

25.1.3 ASC BSL 协议数据定义

XMC1300 ASC 引导加载程序的例程与主机之间实现了握手协议，该协议基于 图 25-3 和 表 25-3 中定义的请求 / 确认 / 数据字节。

表 25-3 XMC1300 ASC BSL 中的握手协议数据定义

名称	长度，字节	值	描述
主机发送的请求 / 数据 / 确认：			
BSL_ASC_F	1	6CH	首部请求全双工 ASC 模式并使用当前波特率
BSL_ASC_H	1	12H	首部请求半双工 ASC 模式并使用当前波特率
BSL_ENC_F	1	93_H	首部请求全双工 ASC 模式并请求切换波特率
BSL_ENC_H	1	ED_H	首部请求半双工 ASC 模式并请求切换波特率
BSL_STEP	2	$0XXX_H$	10 位值 (LSB 对齐) 被编程到所选 USIC 通道的 FDR.STEP 位域。 6 个最高有效位应包含全 0。
BSL_BR_OK	1	$F0_H$	在增强的 ASC BSL 模式建立起最终波特率
XMC1300 固件发送的确认			
BSL_ID	1	5DH	接收起始和首部字节，建立波特率
BSL_ENC_ID	1	$A2_H$	在增强的 ASC BSL 模式接收起始和首部字节，建立初始波特率
BSL_PDIV	2	$0XXX_H$	1 包含所选 USIC 通道 BRG.PDIV 位域值的 10 位值 (LSB 对齐)。 6 个最高有效位应包含全 0。
BSL_BR_OK	1	$F0_H$	在增强的 ASC BSL 模式建立起最终波特率
BSL_OK	1	01H	数据接收 OK
BSL_NOK	1	02H	数据接收期间遭遇故障

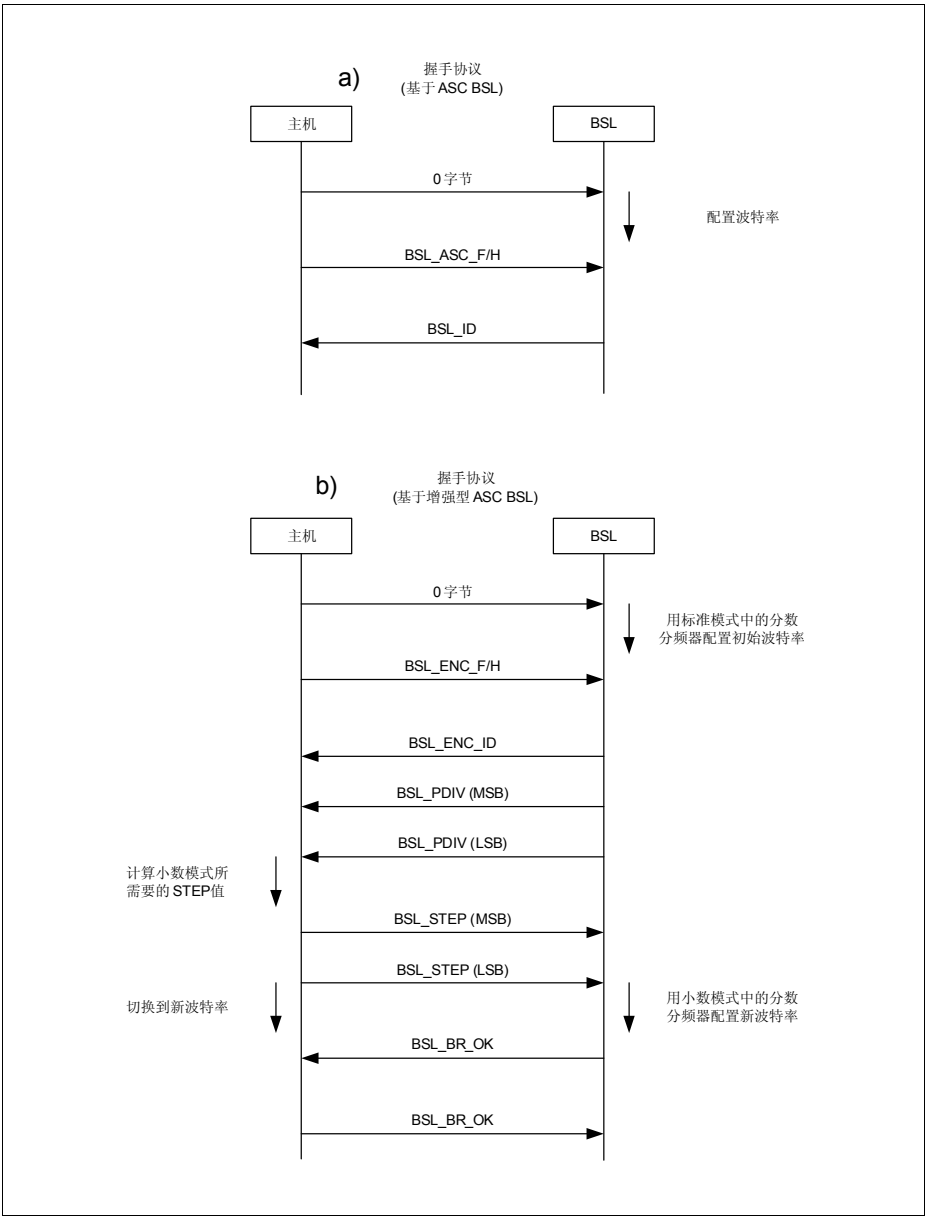


图 25-3　　XMC1300 ASC BSL 进入过程的握手协议

25.2 SSC 引导加载程序

该过程使用 USIC0 模块的通道 0 从一个 SPI 兼容的串行 EEPROM 下载代码到起始地址为 2000’0200H 的 SRAM 区。代码长度可以是任意值，但最大不超过用户可用的 SRAM 大小。

XMC1300 为主 SPI 器件，引导加载程序使用以下引脚：

- MRST (主数据输入) 和 MTSR (主数据输出) - 位于同一 引脚 P0.15，使用半双工模式
- SCLK (时钟信号) - P0.14，漏极开路
- SLS (片选 CS 到 EEPROM) - P0.13，漏极开路

注： 由于引导加载程序的特殊功能 - 只执行一次顺序读操作 - 不需要专用的 SLS 控制

注： 由于 XMC1300 引脚被配置为漏极开路，需要在总线上加外部上拉电阻

必须将一个 25xxx 类型的 SPI 兼容串行 EEPROM 连接到上述定义的引脚。时钟信号在器件启动时被配置为 MCLK/10 频率。

XMC1300 中的 SSC 引导加载程序可以与很宽范围的串行 EEPROM 类型通信：如从使用 8 位寻址 (容量最大为 2K 位，已经相当过时) 到使用 24 位寻址 (1M 位及以上) 的器件。所连接的 EEPROM 类型通过检测接收的首部字节确定，如 **表 25-4** 所示。

注： 除了引导加载程序能支持所有器件类型以外，可以下载的代码 / 数据的长度受可用 SRAM 大小的限制。

表 25-4 SSC BL: 确定 EEPROM 类型和数据流

SSC 帧		与 8 位寻址的 EEPROM 连接		与 16 位寻址的 EEPROM 连接		与 24 位寻址的 EEPROM 连接	
N	数据	P11-发送	P11-接收	P11-发送	P11-接收	P11-发送	P11-接收
1	03_H	读命令	XX_H 默认值	读命令	XX_H 默认值	读命令	XX_H 默认值
2	00_H	地址	XX_H	Address_H	XX_H	Address_H	XX_H
3	00_H	空字节	标识	Address_L	XX_H	Address_M	XX_H
4	00_H	空字节	容量	空字节	标识	Address_L	XX_H
5	00_H	空字节	数据字节 1	空字节	容量、高字节	空字节	标识
6	00_H	空字节	数据字节 2	空字节	容量、低 字节	空字节	容量、高字节

表 25-4 SSC BL: 确定 EEPROM 类型和数据流

SSC 帧		与 8 位寻址的 EEPROM 连接		与 16 位寻址的 EEPROM 连接		与 24 位寻址的 EEPROM 连接	
7	00_H	空字节	数据字节 3	空字节	数据字节 1	空字节	容量、中字节
8	00_H	空字节	数据字节 4	空字节	数据字节 2	空字节	容量、低 字节
9 …	…	空字节	数据字节 5 ...n	空字节	数据字节 3 ...n	空字节	数据字节 1 ...n

所连接的用于下载的 EEPROM 必须包含：

- 标识字节 D5H，在首地址 (0000H)
- 以字节为单位的代码长度 (Code_Length)，如下：
  - 对于 8 位寻址的 EEPROM- 只用 1 个字节表示，存放在地址 0001H 中
  - 对于 16 位寻址的 EEPROM - 用两个字节表示，按高 / 低顺序存放在地址 0001H/0002H 中
  - 对于 24 位寻址的 EEPROM - 用三个字节表示，按高 / 中 / 低顺序存放在地址 0001H/0002H/0003H 中
- 所定义长度的代码紧接着顺序存放

SSC 引导加载程序的功能包括：

1. 通过设置 USIC0_C0_KSCFG.MODEN = 1 来使能通信通道
2. 确保 USIC0 模块的时钟门控无效
3. 按以下步骤配置 USIC0_C0 通道：
   a) SSC 模式、 8 个数据位、 MSB 在先
   b) SCLK 频率 = MCLK/10
   c) 不激活 SLS
   d) 引脚配置如上所述
   e) 使能发送
4. 通过设置 CS=1 来禁止 EEPROM
5. 使能 EEPROM (CS=0) 并发送读命令 (03h)
6. 发送两个 0 字节，这是为了从地址 0000h 请求数据
7. 只要在 EEPROM 仍处于选中状态 (CS=0) 进行下一次字节传送 - 数据就会从下一个地址被读出。
8. 检查收到的最后一个字节：
   a) 如果等于标识字节 (D5H) - EEPROM 采用 8 位寻址：
   -SSW 再读一个字节并将其作为代码长度保存 - 仅一个字节有效
   - 继续 p.10
9. 再读一个字节：

a) 如果等于标识字节 (D5H) -EEPROM 采用 16 位寻址：
-SSW 按高 / 低顺序读取 2 个字节来确定代码长度 (2 字节)
- 继续 p.10

10. 再读一个字节：
a) 如果等于标识字节 (D5H) - EEPROM 采用 24 位寻址：
-SSW 按高 / 中 / 低顺序再读 3 个字节来确定代码长度 (3 字节)
- 继续 p.10
b) 如果不等 - 直接退出该序列

11. 检查代码长度值：
a) 如果为 0 或大于 XMC1300 所允许的值 - 直接退出该序列

12. 读 [Code_Length] 个字节，并将这些字节数据顺序存储在 SRAM 中 (从地址 2000’0200H 开始)
13. 禁止 EEPROM (CS=1)
14. 通过设置 USIC0_C0_KSCFG.MODEN = 0 来禁止通信通道
15. 设置 SSW 标志 BSL_act = 1 ，然后退出该序列

在最后一个字节被接收并存储后，启动软件从地址 20000’0200H 启动前面下载的代码。

25.3 用户可用的固件例程

ROM 内部有几个可用的用户例程 (参见**表 25-5**) ，它们可以被应用软件调用。

表 25-5 XMC1300 ROM 中的用户例程

XMC1300 ROM		描述
地址	内容	
0000’0100H	_NvmErase	指向**擦除 Flash 页**例程的指针
0000’0104H	_NvmProgVerify	指向**擦除、编程和校验 Flash 页**例程的指针
0000’0108H	_BmiInstallationReq	指向**请求 BMI 安装**例程的指针序
0000’010CH	_CalcTemperature	指向**计算芯片温度**例程的指针
0000’0120H	_CalcTSEVAR	指向**计算用于温度比较的目标值**例程的指针

NVM 相关的功能 (见 **25.3.1 节** 和 **25.3.2 节**) 返回如 **表 25-6** 所示的状态指示。

表 25-6 XMC1300 ROM 中的 NVM 例程返回的状态指示标志

状态指示标志		描述
符号名称	值	
NVM_PASS	0001’0000H	函数成功
NVM_E_FAIL	8001’0001H	一般错误

表 25-6 XMC1300 ROM 中的 NVM 例程返回的状态指示标志

状态指示标志		描述
符号名称	值	
NVM_E_SRC_AREA_EXCEEDED	8001’0003H	源数据不在 RAM 中
NVM_E_SRC_ALIGNMENT	8001’0004H	源数据不是 4 字节对齐
NVM_E_DST_AREA_EXCEEDED	8001’0005H	目标数据 不 (完全) 位于 NVM 中
NVM_E_DST_ALIGNMENT	8001’0006H	目标数据没有被正确对齐
NVM_E_NVM_FAIL	8001’0009H	NVM 模块不能被物理访问
NVM_E_VERIFY	8001’0010H	所写页校验不成功

下面概要描述这些例程。

25.3.1 擦除 Flash 页

XMC1300中的Flash可以以一页为最小单位被擦除1页 = 16 个16字节的块 = 256 字节 。

- 输入参数
  – 待擦除Flash页的逻辑地址，必须为页对齐并且位于NVM的地址范围内 (pageAddr)
- 返回状态 (参见 表 25-6)
  – OK (NVM_PASS)
  – 无效地址 (NVM_E_DST_AREA_EXCEEDED, NVM_E_DST_ALIGNMENT)
  – 操作失败 (NVM_E_FAIL, NVM_E_NVM_FAIL, NVM_E_VERIFY)
- 原型
  NVM_STATUS XMC1000_NvmErasePage (unsigned long pageAddr)

25.3.2 擦除、编程和校验 Flash 页

该函数执行擦除 (如无必要可跳过)、编程和校验所选择的 Flash 页。

- 输入参数
  – 目标 Flash 页的逻辑地址 (dstAddr)
  – SRAM 中的数据开始地址 (srcAddr)
- 返回状态 (参见 表 25-6)
  – OK (NVM_PASS)
  – 无效地址 (NVM_E_SRC_AREA_EXCEEDED, NVM_E_SRC_ ALIGNMENT, NVM_E_DST_AREA_EXCEEDED, NVM_E_DST_ALIGNMENT)
  – 操作失败 (NVM_E_FAIL, NVM_E_NVM_FAIL, NVM_E_VERIFY)
- 原型

- NVM_STATUS XMC1000_NvmProgVerify (unsigned long srcAddr, unsigned longdstAddr)

25.3.3 请求 BMI 安装

该程序启动一个新 BMI 值的加载。特别是，它还可以用来恢复一个已经交付并处于用户生产模式的器件的状态。

- 输入参数
  – 要加载的 BMI 值 (requestedBmiValue)
- 返回状态 - 仅在发生错误时返回 , 如果 OK 则该程序触发一次复位，不返回到调用程序
  – 错误的输入 BMI 值 (0001H)
- 原型
  unsigned long XMC1000_BmiInstallationReq (unsigned short requestedBmiValue)

25.3.4 计算芯片温度

该例程根据XMC1300内建温度传感器的测量值来计算当前的芯片温度, 该计算基于Flash中保存的微调值和预先计算的常数 (参见 **表 25-7**) 数据以及实际测量数据 (从温度传感器计数器 2 监视寄存器 ANATSEMON 读取)。

- 输入参数
  – 无
- 返回状态
  – 以开氏度为单位的芯片温度
- 原型
  unsigned XMC1000_CalcTemperature (void)

25.3.5 计算用于温度比较的目标值

该例程为**计算芯片温度** 的逆过程，所计算的值必须存放在 ANATSEVAR 寄存器中，以便在芯片温度高于 / 低于某个目标 / 阈值时从 ANATSESTATUS.TSE_CP_VAR_STAT 位获得指示。

- 输入参数
  – 以开氏度为单位的阈值温度 - 允许的范围为 223...423 (温度)
- 返回状态
  – 与输入参数相等的传感器阈值
- 原型
  unsigned long XMC1000_CalcTSEVAR (unsigned long temperature)

25.4 Flash 中用户例程使用的数据

表 25-7 列出了 XMC1300 Flash 中用户例程使用的数据。

表 25-7 XMC1300 中 SSW 和用户软件的基本 Flash 数据

地址	长度	功能	目标地址
		启动模式选择：	
$1000'0E00_H$	2 B	引导模式索引 (BMI)	---
$1000'0E10_H$	2 B	BMI 的反码值	---
		与温度传感器相关的数据：	
$1000'0F20_H$	2 B	ANA_TSE_k1	---
$1000'0F22_H$	2 B	ANA_TSE_k3	---
$1000'0F24_H$	4 B	ANA_TSE_k2	---
$1000'0F28_H$	2 B	ANATSEMON_min	---
$1000'0F2A_H$	2 B	ANATSEMON_max	---

调试系统

26 调试系统 (DBG)

调试系统是对常规处理器架构的扩展。 XMC1300 系列微控制器提供了一套支持硬件断点和观察点选项的完整硬件调试解决方案。这就为处理器、存储器和可用设备提供了高系统可见性。调试功能采用标准的ARM Cortex M0 配置实现。调试功能被集成到了ARM Cortex-M0 处理器架构中。调试系统支持串行线调试和单引脚调试接口。

参考文献

[1] Cortex-M0 Technical Reference Manual

[2] Cortex-M0 Integration and Implementation Manual

[3] Cortex-M0 User Guide

[4] ARMv6-M Architecture Reference Manual

[5] ARM Debug Interface V5

[6] CoreSight Components Technical Reference Manual

[7] CoreSight DAP-Lite

26.1 概述

Cortex-M0 调试系统的基本调试功能被限制为非侵入式调试，它包括处理器的停止、单步执行、处理器内核寄存器访问、复位和硬故障向量捕获。除了硬件调试组件以外，还可无限制地使用软件断点。调试器通过串行线调试 (SWD) 或单引脚调试 (SPD) 接口的端口连接到调试系统，并使用 CoreSight 基础架构和常规访问流程。这就使调试器可以识别处理器及其调试能力。调试系统允许访问全部系统存储器，并通过连接到系统总线矩阵的内部调试访问端口 (DAP) 访问零等待状态的系统从属器件。这种访问是非入侵式的，调试器可以在处理器停止或运行时访问这些器件，包括存储器。如果处理器停止，则可以访问全部内核寄存器。系统控制空间 (SCS) 通过可用的寄存器来提供调试功能。数据观察点单元 (DWT) 提供两个观察点寄存器组，每组都实现了基于数据地址和 PC 的观察点功能，包括比较器地址屏蔽。处理器断点单元 (BPU) 提供 4 个基于 PC 的断点寄存器。调试工具可以通过 DAP 访问系统。当处理器处于运行、停止状态或保持在复位状态时，DAP 允许访问调试资源。如果处理器被配置为在发生调试事件时停止运行，则当调试事件发生时，处理器进入调试状态。可以通过检查 ROM 表来识别所支持的调试基础架构。

26.1.1 特性

精确的调试和跟踪系统提供了以下功能：

- 串行线调试端口 (SWD) 提供串行线调试，它允许通过 2 个引脚进行调试。
- 单引脚调试 (SPD) 提供使用一个引脚的调试能力。
- 处理器停止、单步执行、处理器内核寄存器访问。
- 复位和硬故障向量捕获。
- 软件断点

- 对全部系统存储器的访问
- 支持 4 个硬故障断点
- 支持 2 个观察点

*注： 有关调试功能的详细信息，请参见 **ARMv6-M Architecture Reference Manual**。*

26.1.2 框图

调试系统框图如 **图 26-1** 所示。

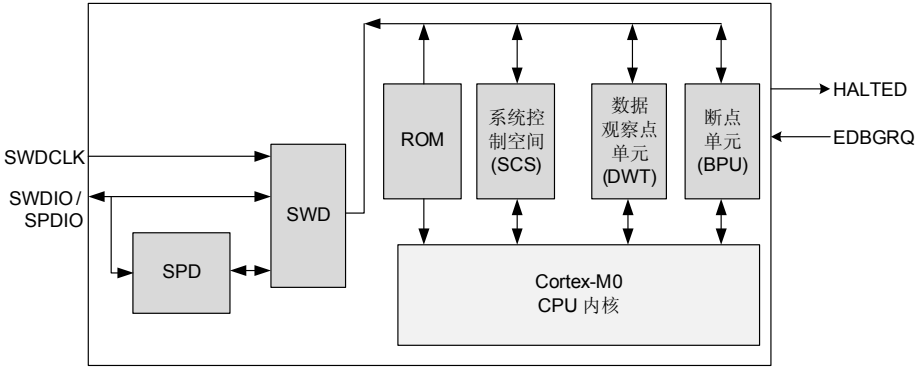


图 26-1 调试和跟踪系统框图

26.2 调试系统操作

调试系统提供了通用的调试选项。调试选项是基于断点和 CPU 停止的。可用的调试资源包括数据观察点和跟踪、断点单元、 ROM 表以及 SCS 系统控制块和调试控制块。调试工具可通过串行线调试接口 (SWD) 或单引脚调试 (SPD) 访问调试功能。访问协议 (SWD 或 SPD) 的选择通过 BMI 模式设置完成。

26.2.1 系统控制空间 (SCS)

SCS 是一个区域，调试器可以在该区域直接访问存储器映射的调试寄存器。SCS 中的调试资源和调试寄存器都可以通过 DAP 接口访问。可访问的资源有系统控制和 ID 寄存器等，包括系统控制块。此外，系统定时器和中断控制器 (NVIC) 也可以通过 SCS 访问。

26.2.2 数据观察点和跟踪 (DWT)

DWT 单元提供了一个外部程序计数器 (PC)，其采样能力基于 PC 采样寄存器和比较器，这就允许支持地址匹配观察点和指令地址匹配 PC 观察点。PC 采样功能和观察点支持相互独立的操作，观察点事件与引起该事件的指令是异步的。XMC1300 支持两个观察点，每个观察点都支持比较、屏蔽和功能寄存器。观察点事件导致处理器停止执行并使处理

器系统进入调试状态。数据地址匹配导致产生一个观察点事件。指令地址匹配导致产生一个 PC 观察点。 DWT 寄存器可通过 DAP 接口访问。

一个 DWT 程序计数器采样寄存器允许调试器定时采样 PC 而不需要停止处理器，并且允许粗粒度分析。PC 采样功能和观察点支持相互独立的操作。寄存器被设定为使调试器访问它时可不改变器件中当前正在执行的任何代码的行为。

在一个单指令地址产生一个断点的推荐机制是使用 BPU。如果要在一个地址范围内产生一个 PC 匹配事件，则必须使用基于 DWT 的机制。一个观察点事件的调试返回地址值必须是在负责产生该观察点的那条指令之后要被执行的指令的地址。

26.2.3 断点单元 (BPU)

断点 (BKPT) 指令提供软件断点。它可导致一个正在运行的系统暂停执行，这取决于调试配置。BKPT 是由执行 BKPT 指令或 BPU 中的匹配事件引起的同步调试事件。一个 BKPT 可以使处理器进入调试状态。调试工具可以利用这种情形在系统执行到位于某一特定地址的指令时查看系统状态。 BPU 支持在取指时执行断点功能，这是基于指令地址比较器实现的。 M0 提供了四个断点比较器寄存器。每个断点比较器寄存器都包含自己的使能位。比较器匹配指令从程序存储器区域获取，并且只能进行指令读访问。比较器不对数据读或数据写访问进行匹配比较。地址匹配比较可以在高半字、低半字或两个半字上进行。产生一个调试事件的可能原因有调试停止、 BKPT、 DWT 陷阱和向量捕获。

26.2.4 ROM 表

为了识别一个 Cortex-M0 处理器及其调试系统组件，调试器要定位和识别 ROM 表。ROM 表的标识值是预定义的，它包含指向 SCS、 BPU 和 DWT 的指针。每个调试模块都需要其自己的 ROM 表，并指示该块是否存在。 ROM 表可以通过 DAP 接口访问。

26.2.5 调试工具接口访问 - SWD

基于 SWD 的工具访问必须使用一个连接序列，以确保热插拔串行连接不会产生无意的数据传送。该连接序列确保 SWD 正确地与首部同步。该序列由 50 个时钟周期的全 1 数据组成。该连接序列还被用作一个线复位序列。该协议要求将数据输入端上出现的任何 50 个连续的 1 检测为线复位，与协议的状态无关。线复位后调试器读取 IDCODE 寄存器，该寄存器给出是否已实现了帧对齐的确认信息。

26.2.5.1 基于 SWD 的传送

SWD 接口要求在任何数据传送的数据阶段结束后时钟信号还要持续多个时钟周期。这可以确保通过SWD提供完成数据传送所需要的完整时钟。传送完成可以通过立刻启动一次新的 SWD 操作来实现，继续为 SWD 接口提供时钟，直到主机启动一次新的 SWD 操作，或者在奇偶校验位之后继续提供至少 8 个时钟上升沿，然后再停止该时钟。

串行线上的每个操作序列由两个或三个阶段组成，这是从调试器的角度定义的。数据包请求和应答响应阶段总是存在。在数据读或数据写请求得到一个有效应答响应或超时检测标志被置位的情况下，才进入数据传送阶段。

SWD 协议对所有的数据包请求和数据传送阶段进行简单的奇偶校验。对于一次数据包请求，只对 4 位的首部进行奇偶校验。对于一次数据传送，对全部 32 个数据位进行奇偶校验。奇偶位是在包请求之后和数据传输之后直接添加的。奇偶校验位计算时不包括奇偶位。

串行线调试协议操作

SWD 协议支持以下操作：

- **写操作。**该操作包括 OK 响应 (3 个阶段：8 位写数据包请求， 3 位 OK ACK， 33 位要写的数据包括最后的 1 个奇偶位)。
- **读操作。**该操作包括 OK 响应 (3 个阶段：8 位读数据包请求 ， 3 位 OK ACK， 33 位读取的数据包括最后的 1 个奇偶位)。
- **等待响应。**等待对读或写操作请求的响应 (2 个阶段：8 位读或写数据包请求， 3 位 WAIT ACK 响应)
- **故障响应。**对读或写操作的相应 (2 个阶段 8 位读或写数据包请求，3 位 FAULT ACK 响应)
- **协议错误**。在目标未能返回一个 ACK 响应时发生协议错误序列 (1 个阶段：8 位数据包请求 - 目标未驱动信号线的时间达到 32 个 位周期)

注： 在各操作阶段之间需要有一个周期的周转期，但这仅限于传送方向改变。如果传送方向不变，则不引入周转期。如果协议在向调试器进行一次数据传送后结束，则引入一个周转期。在单周期的周转期结束后协议必须继续运行。

26.2.5.2 基于 SWD 的错误

在 SWD 接口可能会发生协议错误。可通过对数据进行奇偶校验来检测这些错误。报文首部错误可通过奇偶位错误检测到。调试器收不到响应或一个端口不响应该报文时，调试器必须回退。在回退机制期间，调试器不读 IDCODE 寄存器，并确保调试端口能迅速响应，然后再重试原来的访问。

错误可能是由 DAP 自身返回的，也可能来自一个调试资源。当一个错误发生时，错误位保持粘粘性，直到被清除为止。调试器必须在执行了一系列的事务后必须检查错误的状态来识别错误。

SWD 接口内的错误条件使用粘性的标志记录。可能发生的错误有读和写错误、超限检测、协议错误。当粘性位被置 1 后，它会保持不变，直到被显性清 0。当一个错误标志被置 1 时，当前事务结束，对访问端口的任何后续访问事务都被丢弃，直到该粘性标志位被清 0。调试器必须在执行了一系列的事务后定期检查控制 / 状态寄存器，以检测错误。

读和写错误可能发生在 DAP 内部，也可能来自被访问的资源。另外，如果在调试器执行访问端口事务请求时调试器电源域发生掉电，也可能产生一个读或写错误。

在一个事务序列中可能会检测到超时错误。因此调试器必须在每个序列之后检查这种错误。可以通过清除 STICKYORUN 和 ORUNDETECT 退出超时检测模式。

在串行线接口中也可能发生协议错误，例如由于线路电平引起的错误。这些错误可以通过对数据进行奇偶校验来检测。如果 SWD 接口在一个调试报文的首部检测到一个奇偶校验错误，则该错误会被反映给调试器。如果 SWD 接口未收到对一个报文的响应，调试器必须回退。然后它必须在重试原始访问前读 IDCODE 寄存器，以确保调试端口能迅速响应。如果 SWD 在一次写事务的数据阶段检测到奇偶校验错误，则它会置位控制和状态寄存器中的粘性写数据错误标志 (WDATAERR)。来自调试器的后续访问，除了 IDCODE、CTRL/STAT 或 ABORT 以外，都会导致一个 FAULT 响应。

26.2.6 调试工具接口访问 - 单引脚调试 (SPD)

基于工具接口访问的 SPD 协议允许仅使用一个引脚进行系统调试。位频率为 2 MHz，允许 1.4 Mbits/s 的有效 SWD 报文速率。该协议对工具与器件之间的时钟偏差有很强的鲁棒性。

SPD 协议用 SPD 信号边沿之间的时间间隔对 SWD 位编码。一个 SWD 值 ‘0’ 被编码为一个 0.5 µs 的短间隔，值 ‘1’ 被编码为 1 µs 的间隔。两个传送方向都使用这种编码机制。SPD 信号电平最初总是以一个 1 µs 宽的正 SPD 信号脉冲开始，该脉冲被编码为 SPD 起始位，它不属于 SWD 协议的一部分。报文其余部分的发送与边沿方向无关。由于协议以一个逻辑 1 开始位检测开始，建议用一个逻辑 0 信令信号电平结束协议。如果在报文最后一位之后 SPD 信号电平为高，调试工具会在 0.5 µs 后添加一个下降沿的 ‘0’ 位。这可以由调试工具通过传送奇数个位来实现。

SWD 有一个清晰的请求响应协议。在该协议中，工具总是作为请求者，被调试的器件执行请求并发送响应。因此，SWD 模块会将传送方向改变为由 SWD 协议操作 (26.2.5.1 节) 所定义的方向。

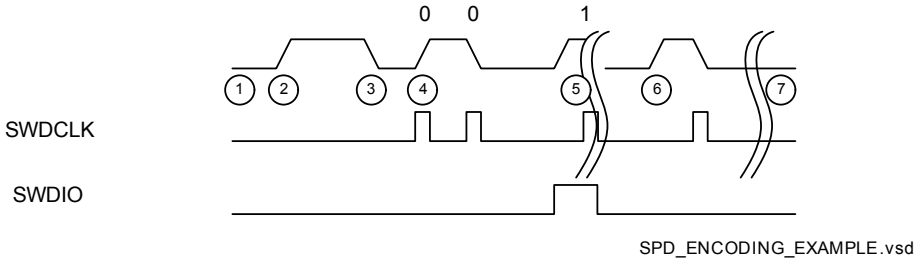


图 26-2 SPD 编码示例

作为一个基本示例，上图描述了一次简单的传送。 SPD 模块最初 (1) 处于空闲状态。在空闲状态期间， SPD 模块不产生给 SWD 模块的时钟。在点 (2)， SPD 检测到一个上升沿并转到接受状态。协议的第一个上升沿只是 SPD 模块的起始位，不会被传送到 SWD

模块。接下来的位是 SWD 的首部信息并被传送到 SWD 模块。在点 (5) 出现一个逻辑 1，在点 (6) 出现一个逻辑 0。在协议方向发生改变之际 (这意味着通过空闲状态实现) ，在状态 (7)，给 SWD 的时钟被关闭。SPD 模块需要一个低电平信号值来结束通信协议，因此它需向接口传送奇数个位。

SWD 和 SPD 模块之间的协议传送要求

去往 SWD 和来自 SWD 的数据传送协议是基于 SWD 时钟的单个位传送。 SWD 时钟由 SPD 模块产生或直接源自被传送的位。在接收方向， SPD 模块将数据逐位传送到 SWD 模块， SWD 模块在每个时钟周期接收一位。

当经过模块空闲状态时， SPD 模块关闭 SWD 时钟。

当向器件写数据到时 (从 SPD 角度看，接收过程)，调试器在 SWD 报文前添加一个 SPD 起始位。 SPD 接收过程的起始位不会被传送到 SWD 模块。接收过程起始位会在每次产生新的接收过程起始位时，被放在原 SWD 协议的前面；而位于 SWD 首部之前的这个新的接收过程起始位又再次位于调试器的写数据之前。另外， SPD 模块在接收方向需要一个以一个逻辑0信号电平为结束的协议，这可以由工具通过传送奇数个位来实现。在SWD基础上，这已经通过在首部阶段添加前面所描述的 SPD 起始位得到实现。应答阶段仅有 3 个位。在协议的数据阶段，调试器必须再添加一个 0 位。(SPD 起始位、32 位数据、校验位、SPD 奇数位为逻辑 0)。

由于 SWD 协议在接收方向的数据传送结束后至少需要 8 个时钟来结束协议操作，调试器必须在该协议上添加 9 个逻辑0 (8 个0是为了满足前述要求，1个0是为了构成奇数个位)。

基于 SPD 协议的工具交互

所有的 SPD 通信均由调试工具启动。工具会发送一个编码的 SWD 首部到一个 SPD 报文 (首部) ，然后切换 SPD 信号方向为输入并等待来自设备的应答。

SPD 接收：

在一个低电平基础上，工具以 SPD 起始位启动协议。起始位由一个上升沿和一个持续时间 1µs 的信号来指示。接下来的位基于 SWD 协议序列。

起始序列：SPD 起始位 + SWD 首部 + 0 位 + 0 位 (在首部结束后额外增加的两个 0 位是为了在信令线获得低电平。)。

接收到首部后，SWD 会输出应答响应。如此一来，SPD 会将状态从接收改变到空闲，再变到发送和应答，之后又变到空闲。

从空闲变为传送写数据的接收状态，也需要从工具发送起始位。

SPD 起始位必须是一个具有确定持续时间的逻辑 1。如果该信号的持续时间过长， SPD 会进入超时状态。如果该信号的持续时间过短，起始位不能被识别。

调试工具在 SPD 接收方向不持续发送数据会导致在 SPD 模块产生一个超时，该超时事件使其再次进入空闲模式。如果一次新通信从空闲状态开始，则工具需要为新通信提供 SPD 起始位。起始位超过 1.4 µs 后被识别为超时。

一次持续的写 (SPD 接收) 允许继续执行协议而不需要 SPD 起始位。在这种情况下，SPD 接口保持在接收状态，并且必须只执行 SWD 接收协议。在这种情况下， SWD 首部必须以一个由 SWD 协议定义的逻辑 1 开始。

SPD 发送：

工具需要在低电平结束一次接收过程。在低电平的基础上，发送过程的起始位由一个上升沿指示。发送过程也必须以一个信号低电平结束。

空闲：

从接收到空闲 - 由 SPD 或超时驱动。

从发送到空闲 - 由引起方向改变的 SWD 模块驱动 。

从接收状态切换到发送状态的时间在 375 和 500ns 之间，并且总是经过空闲阶段。工具必须在 375 ns 内从写 (SPD 接收过程) 方向改变到读 (SPD 发送过程) 方向。发送过程总是以一个上升沿开始。

26.2.7 调试访问和 Flash 保护

XMC1300 Flash 只对扇区 0 实现了读保护。所有其它扇区都可以读取。另外，用户擦除和写保护配置是可用的，该机制允许防止 Flash 内容被意外写入和擦除。调试器不能绕过这种保护。

正因为如此，默认状态和系统复位后的调试接口是被禁止的。引导模式索引 (BMI) 决定在 SSW 结束后是否可以进入调试模式以及是否使能调试接口。

在 BMI 被配置为用户生产模式之前，可以将 BMI 改变为任意值，之后 BMI 只能被恢复到其默认值，默认值不允许调试访问。

除了 BMI 配置以外，调试系统还支持这样一种保护机制：该机制在启动软件 (SSW) 运行期间防止对系统地址区域进行调试访问。固件基于寄存器设置来控制调试访问。

调试系统支持软件断点，但在 SSW 期间软件断点被禁止。

26.2.8 复位后停止

有两种执行复位后停止的可能。第一种可能是英飞凌特有的，它允许在 “ 上电 ”/ ” 主 ” 复位 (HAR) 之后 CPU 停止在应用程序代码中的第一条指令 。HAR 的激活是基于 BMI 设置和 SSW 执行一个死循环，这种情况允许调试器控制器件。在这种情况下，需要一个由连接的调试工具所驱动的确定的配置序列。第二种可能是复位后停止，这是基于断点和系统复位的常规 ARM 流程。这种复位后停止也被称为 “ 热复位 ”。在上述两种情况下，出于安全考虑，不能在 SSW 运行期间从调试端口访问 ARM 系统，因为在这段时间物理引脚是不可用的。基于 BMI 设置，调试功能可在 SSW 结束后由固件使能。默认的 BMI 配置不允许调试访问。

26.2.8.1 HAR

BMI 可以被配置为在 SSW 结束后允许一次 HAR (UMHAR)，这总是需要执行一次上电复位。新 BMI 配置 BMI.UMHAR (使能了调试功能和 HAR 的用户模式) 需要执行一次主复位，以使器件在 UMHAR 模式引导。

在发生一次 HAR (冷复位情况) 时，系统复位来自一个 PORST (或主复位) 。为了实现一次 HAR，工具必须注册 (CDBGPWRUPREQ) 和使能 (DHSCR.C_DEBUGEN) 调试系统。另外，它还必须停止 (DHCSR.C_HALT) CPU 并操作 PC (程序计数器) 使其指向用户代码的起始地址。固件完成后等待工具注册，此时才能进行调试器注册和调试系统使能。在SSW执行期间，调试器没有访问能力，因而不能置位使能位CDBGPWRUPREQ、C_DEBUGEN ，也不能停止 CPU。用户代码的起始地址可以从用户 Flash 中的地址0x10001004读取。在操作PC之前，建议先检查UMHAR引导模式索引BMI.HWCFG位。

下图 (图 26-3 HAR - 复位后停止) 给出了基于模块参与 HAR 的软件流程。

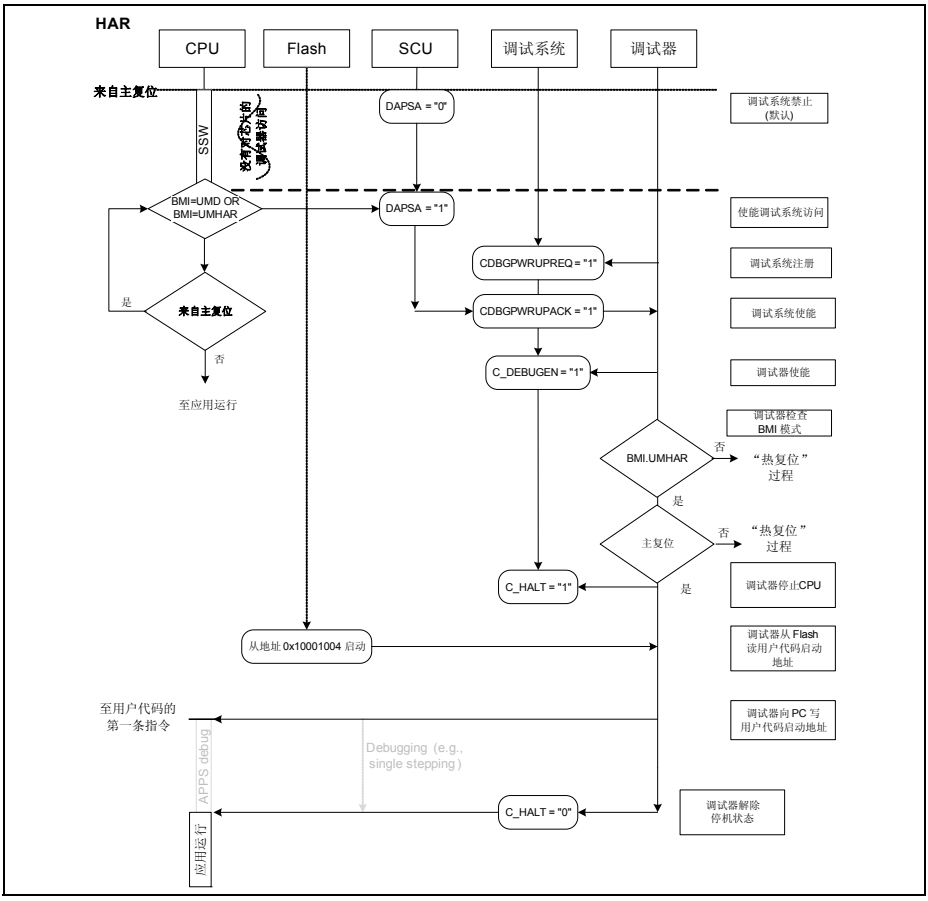


图 26-3　　HAR - 复位后停止流程

26.2.8.2　热复位

系统复位后停止 (热复位) 可以通过在应用程序代码的第一条指令编程一个断点来实现。另外， CDBGPWRUPREQ (工具注册) 和 C_DEBUGEN 必须由调试器置位。在一次系统复位后不考虑 HAR 的情况，因为复位不是来自 PORST (主复位)。

注：　如果CDBGPWRUPREQ 和C_DEBUGEN 在一次系统复位之前已经被置位并且调试器仍然处于注册状态，那么在系统复位后不必再对这两个位进行置位操作。这些位不受系统复位的影响。

工具热插拔允许通过使能 CDBGPWRUPREQ 和 C_DEBUGEN 寄存器来调试系统。建议检查引导模式索引 BMI.HWCFG。热复位流程可以在两种 BMI 模式即 UMD (使能了调试功能的用户模式) 和 UMHAR 模式下完成。在这些情况下，热复位流程是基于断点设置和系统复位的。

一般来说，在一次工具热插拔后即可设置断点，CPU 也可以被直接停止在当前的程序级。为了停在用户代码的第一条指令，工具必须在设置完断点触发一次系统复位。

下图 (图 26-4 热插拔和热复位) 展示了调试工具热插拔的情况或热复位 (系统复位) 后停止以及如何进入停止状态。

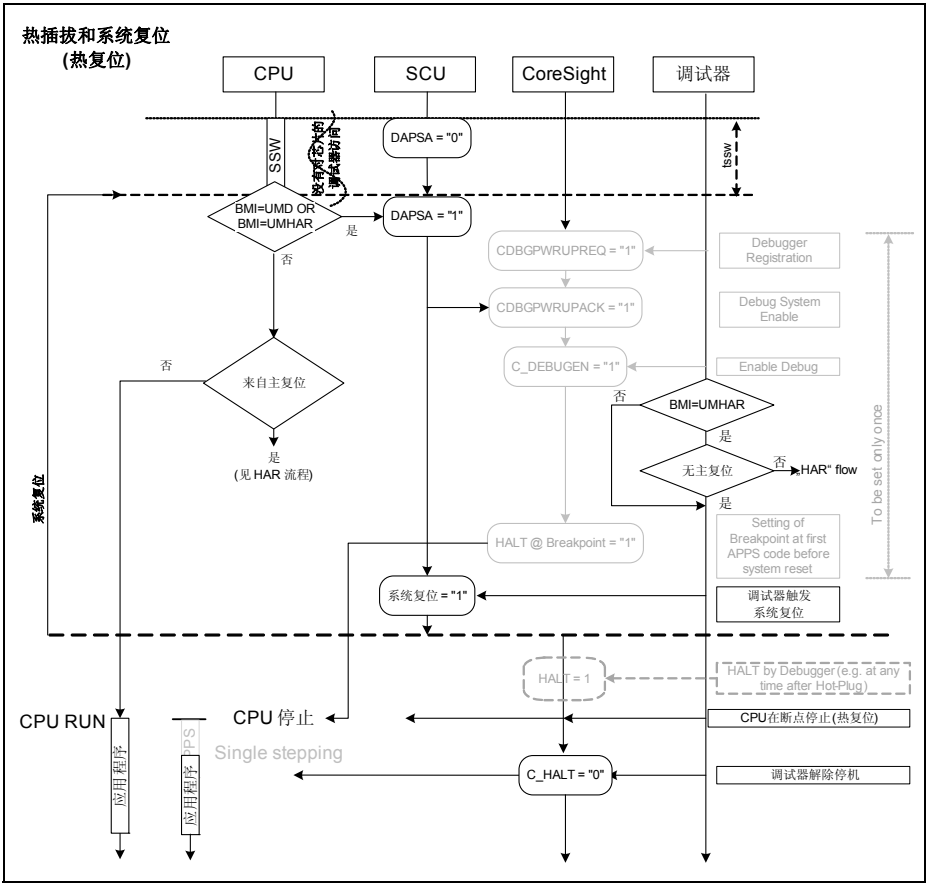


图 26-4　　热插拔或热复位流程

26.2.9 停止调试和外设挂起

XMC1300 器件在 CPU 的执行程序被调试器停止时，例如断点或 C_HALT，支持外设的挂起能力。这就允许调试整个微控制器的关键状态。是否支持挂起功能，必须要在外设本地配置。

在有些情况下，为避免系统错误亦或对应用的致命破坏，保持某些外设运行是重要的，例如一个 PWM 或一个 CAN 节点。因此，允许配置外设在 CPU 进入停止调试模式时的行为。在默认状态，外设对挂起请求是不敏感的。这种敏感性可根据系统的使用情况进行配置。

由外设来决定支持硬挂起还是软挂起。在硬挂起的情况下，外设的时钟被立即关闭，无需等待来自模块的确认。在软挂起时，外设可以决定何时挂起。通常在当前的活动传送结束后挂起。

看门狗定时器仅在挂起总线未激活时运行。这是特别有用的特性，因为已停止的 CPU 不能为它提供服务。也可以使用在挂起期间使能看门狗定时器的选项。如果连接了调试器，该选项允许调试看门狗的行为。

用户必须确保那些外设总是对挂起敏感。为了解决这一问题，每个支持挂起的外设都有一个使能寄存器，该寄存器允许使能挂起功能。下表 (**表 26-1**) 列出了支持和不支持外设挂起的外设。有关软挂起期间外设挂起行为的详细信息可在描述外设的相应章中找到。

表 26-1 外设挂起支持

外设	支持挂起	默认模式	硬挂起	软挂起
RTC	是	未激活	是	否
USIC	是	未激活	否	是
CCU4	是	未激活	是	是
CCU8	是	未激活	是	是
POSIF	是	未激活	是	是
BCCU	是	未激活	是	否
WDT	是	激活	是	否
VADC	是	未激活	是	是
MATH	是	未激活	是	是
PRNG	否	- - -	- - -	- - -

注： 每次复位后，在用户初始化代码中使能调试挂起功能。另外用户必须注意，外设的调试挂起寄存器只能在模块时钟使能后通过 CGATCLR0 寄存器配置。

重要工具提供商注：

外设挂起逻辑被连接到系统复位。一次系统复位激活会导致外设挂起配置信息丢失。因此外设本地的挂起的配置比系统复位要 “ 持久 ”，在系统复位后调试工具需要重新进行

挂起配置。如果挂起功能被激活，这可以通过将用户外设挂起配置映射到工具中并在用户代码的第一条指令处设置一个硬件断点来实现。在 CPU 停在该断点后，工具必须重新进行外设的挂起配置，因为它在系统复位前已经被配置。

注： 挂起激活会导致丢失一个硬件断点，因为系统复位后该断点由工具用来处理挂起配置。

调试工具在系统复位后应重新配置外设挂起选项，可选择在工具重新配置后不需用户交互即可继续执行用户代码，或者仍暂停在用户代码的第一条指令并等待用户操作。

(用户只需一次性地配置所期望的挂起行为，工具可以存储配置信息，在系统复位后自动重新完成配置。)

26.2.10 基于调试系统的处理器唤醒

调试器可以唤醒处于休眠模式 (休眠和深度休眠模式) 的处理器。唤醒是基于一个来自调试系统的新挂起中断事件，这就是停止 (HALT) 事件。如果中断被禁止或其优先级不足以导致异常进入，则中断事件也能唤醒处理器。中断唤醒模式源自处理器系统中可用的 NVIC (可编程的多优先级中断系统) 。调试器可以在休眠和深度休眠模式注册、使能调试系统和停止 CPU 。

注： 通过调试器停止事件从深度休眠模式唤醒仅支持 SWD 接口，不支持 SPD 接口。

26.2.11 调试访问服务器 (DAS)

DAS 的 API 为工具访问提供了一个抽象的物理器件接口。DAS 的关键范例是在目标器件的一个或多个地址空间读取或写入数据。

DAS 特性

- 所有类型工具的标准接口
- 高效和健壮的数据传送方法
- 多个独立的工具可以共享同一物理接口
- 英飞凌的 DAS miniWiggler 支持 SWD 和 SPD

注： DAS 不是 XMC1300 特有的。它可以用于具有 DAP、SWD、SPD 或 JTAG 接口的所有英飞凌 8、16 和 32 位微控制器 。更多详细信息请访问 www.infineon.com/DAS。

26.2.12 调试信号

XMC1300 微控制器家族产品提供的调试功能使用 ARM M0 的调试端口 SWD 或英飞凌专有的 SPD。SWD 端口有 2 个接口信号 (时钟 + 双向数据)。SPD 端口只有一个接口信号 (SPDIO - 双向数据)，它与 SWD 接口的 SWDIO 共用一个引脚。

表 26-2 SWD 顶层 IO 信号

信号	方向	功能
SWDCLK	I	串行线时钟。当运行在串行线调试模式时，该引脚是调试模块的时钟。
SWDIO	I / O	串行线调试数据 IO。外部调试工具使用该信号与 Cortex-M0 调试系统通信和控制 Cortex-M0 调试系统。

HALTED 和 EDBGRQ 信号可以用来基于外部事件停止 CPU。这就允许与 CPU 同步停止外部硬件。停止事件可以通过基于 HALTED 输出的外部硬件来识别。这可以用于与一个逻辑分析仪同步、请求一个断点或用在多 CPU 方案。

26.2.12.1 SWD/SPD 引脚的内部上拉和下拉

内部上拉和下拉是为了确保 SWD/SPD 输入引脚不会处于浮动状态，因为它们直接连接到控制调试功能的触发器。为了避免任何不受控制的 I/O 电压电平，器件在 SWD/SPD 输入引脚提供内部上拉和下拉功能。

- SWDIO/SPD：内部上拉
- SWCLK：内部下拉

26.2.13 复位

调试系统寄存器的位通过上电复位来复位。如果工具已注册，系统中的其它复位对调试寄存器没有影响。

26.3 调试系统省电操作

调试系统位于一直接通 ” 的电源域，这就允许将调试器连接到器件。调试系统不支持任何特殊的电源模式或省电操作。调试系统的电源与包含系统组件的主芯片部分直接连接。

26.4 服务请求生成

调试器可以将 DHCSR.C_MASKINTS 置 1，以防止 PendSV、 SysTick 和外部配置中断发生。

26.5 调试行为

调试系统是基于事件的。如果停止调试被使能，则调试事件是进入调试状态的一个入口。以下调试事件可用：

- 内部停止请求
- 断点

- 观察点
- 向量捕获
- 基于外部的调试请求

以下事件为同步调试事件：

- 断点调试事件，由执行一条 BKPT 指令引起
- 观察点调试事件，由执行一次 BPU 中的匹配引起
- 向量捕获调试事件
- 单步调试事件 (DHCSR.C_STEP)

以下事件为异步调试事件：

- 观察点调试事件，包括 PC 匹配观察点
- 停止请求调试事件 (DHCSR.C_HALT)
- 基于外部信号的调试请求事件 (EDBGRQ 信号)

异步调试事件具有比同步产生的事件低的优先级。一条指令可以产生多个同步调试事件。它也可以产生多个异步异常。

调试故障状态寄存器 (DFSR) 包含每个调试事件的一个状态位。这些位都是写 1 清 0 的。当一个调试事件导致处理器停止或产生一个异常时，这些位被置 1。如果事件被忽略，则这些位也被更新。

为了得到 CPU 端的调试支持，软件必须向 DHCSR.DBGKEQ 寄存器写 0xA05F 。

26.6 电源、复位和时钟

对于电源、复位和时钟的要求都源自 ARM。

26.6.1 电源管理

没有为调试系统实现电源管理。然而，为了注册工具，工具必须通过置位 CDBGPWRUPREQ 来遵循常规的工具流程。

26.6.2 调试系统复位

SWD 寄存器位于上电复位 (PORST) 域。

调试逻辑由系统复位来复位，如果工具未连接到芯片，因而没有被注册，这指示未置位 **CDBGPWRUPREQ**。

处理器复位

一次处理器复位或热复位 (SYSRESETn) 会初始化处理器的大部分功能，但不包括调试逻辑、 BKPT 单元、 DWT 单元和 SCS。

上电复位

PORESETn 复位会初始化 SWD 访问端口、AHB-AP 逻辑和所有其它与调试系统相关的功能。

正常操作

在正常操作期间，复位 PORESETn 和 SYSRESE 无效。

处理器复位影响调试系统寄存器位 CDBGPWRUPACK

如果工具没有被注册，系统复位也会影响调试系统。工具注册通过寄存器位 CDBGPWRUPACK 的激活来指示，该位由调试器通过使能 CDBGPWRUPREQ 来激活。

26.6.3 调试系统时钟

调试系统时钟也称为 DCLK，它直接来自 SCLK(自由运行时钟)。当调试器被连接时，DCLK 必须一直被驱动。

26.7 初始化和系统相关性

26.7.1 ID 代码

接下来的各小节描述可用的 ID。

26.7.1.1 CPUID

CPUID 为 $410CC200_H$。

26.7.1.2 ROM 表

DAP 控制器自带默认的 ARM JTAG ID，该 ID 由 IDCODE 定义。SW-DP 的 ID 值为 $0BB11477_H$ 。

JEP106 ID 是 JEDEC 标准定义，对英飞凌为 $C1_H$。

为了识别英飞凌为制造商和通过器件 ID 识别 XMC1300，器件中添加了一个 ROM 表。在系统存储器中定义了一个 4 k 的内存块，用于存储 ID。

CortexM0 ROM 表位于 0xE00F_F000，由下面的值定义：

- M0 ROM 表，位于 0xE00F_F000 - PID JEP106: 0x3B/0x4; part# = 471
- 数据观察点 / 跟踪，位于 0xE000_1000: PID JEP106: 0x3B/0x4; part# = 0x008
- 断点单元，位于 0xE000_2000: PID JEP106 = 0x3B/0x4; part# = 0x00B

ROM 表的格式如**表 26-3** 所示。

表 26-3 XMC1300 ROM 表的 PID 值

名称	偏移	描述	参考
PID0	$FE0_H$	外设 ID0	器件号 [7:0]
PID1	$FE4_H$	外设 ID1	位 [7:4] JEP106 ID 代码 [3:0] 位 [3:0] 器件编号 [11:8]

表 26-3 XMC1300 ROM 表的 PID 值

名称	偏移	描述	参考
PID2	$FE8_H$	外设 ID2	位 [7:4] 版本号 位 [3] == 1：JEDEC 分配的 ID 域 位 [2:0] JEP106 ID 代码 [6:4]
PID3	FEC_H	外设 ID3	位 [7:4] RevAnd，次版本域 位 [3:0] 如果非零表示客户修改后的块
PID4	$FD0_H$	外设 ID4	位 [7:4] 4KB 计数 位 [3:0] JEP106 连续码

ROM 表的值代表英飞凌的器件号， JEP106 和版本号见 SCU 一章。

26.8 调试系统寄存器

寄存器概览

绝对寄存器地址通过加法计算：

DWT、BPU、ROM 表、DCB 和 SCS 中的调试寄存器都可以通过调试器或从 CPU 以存储器映射方式访问。

表 26-4 寄存器地址空间

模块	基地址	结束地址	备注
DWT	E000 1000_H	E000 $1FFF_H$	数据观察点
BP	E000 2000_H	E000 $2FFF_H$	断点单元
SCS	E000 $E000_H$	E000 $EEFF_H$	SCS (这里为 DBG CTRL)
ROM	E00F $F000_H$	E00F $FFFF_H$	ROM 表区域

表 26-5 寄存器一览表

简称	描述	偏移地址	访问方式		描述见
			读	写	
SCS_DFSR	调试故障状态寄存器	$D30_H$			页 26-18
SCS_DHCSR	调试停止控制和状态寄存器	$DF0_H$			页 26-20
SCS_DCRSR	调试内核寄存器选择器寄存器	$DF4_H$			页 26-24
SCS_DCRDR	调试内核寄存器数据寄存器	$DF8_H$			页 26-25
SCS_DEMCR	调试异常和监视控制寄存器	DFC_H			页 26-26
DWT_CTRL	DWT 控制寄存器	000_H			页 26-27
DWT_PCSR	DWT 程序计数器采样寄存器	$01C_H$			页 26-28
DWT_COMP0	DWT 比较器寄存器	020_H			页 26-29
DWT_COMP1	DWT 比较器寄存器	030_H			页 26-29
DWT_MASK0	DWT 屏蔽寄存器	024_H			页 26-30
DWT_MASK1	DWT 屏蔽寄存器	034_H			页 26-30
DWT_FUNCTION0	DWT 比较器功能寄存器	028_H			页 26-31

表 26-5　寄存器一览表

简称	描述	偏移地址	访问方式		描述见
			读	写	
DWT_FUNCTION1	DWT 比较器功能寄存器	038_H			页 26-31
BP_CTRL	断点控制寄存器	000_H			页 26-32
BP_COMP0	断点比较器寄存器	008_H			页 26-33
BP_COMP1	断点比较器寄存器	008C			页 26-33
BP_COMP2	断点比较器寄存器	010_H			页 26-33
BP_COMP3	断点比较器寄存器	014_H			页 26-33

26.8.1　DFSR - 调试故障状态寄存器

SCS_DFSR

DFSR 寄存器提供一个调试事件发生的顶级原因。向一个寄存器位写 1 会将该位清 0。通过单步执行的指令读 HALTED 位会返回一个 UNKNOWN 值。

SCS_DFSR
调试故障状态寄存器　　($D30_H$)　　复位值 : $0000\ 0000_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0															
r															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											EXTERNAL	VCATCH	DWTTRAP	BKPT	HALTED
r											rw	rw	rw	rw	rw

域	位	类型	描述
HALTED	0	rw	**HALTED** 指示由一个 C_HALT 或一个 C_STEP 请求产生的一个调试事件，由一个写 DHCSR 触发。 0_B　停止请求调试事件未激活。 1_B　停止请求调试事件被激活。

域	位	类型	描述
BKPT	1	rw	**BKPT** 指示由 BKPT 指令执行或 BPU 中的断点匹配产生的一个调试事件。 0_B 无断点调试事件。 1_B 至少有一个断点调试事件。
DWTTRAP	2	rw	**DWTTRAP** 指示由 DWT 产生的一个调试事件。 0_B 没有由 DWT 产生的调试事件。 1_B 至少有一个由 DWT 产生的调试事件。
VCATCH	3	rw	**VCATCH** 指示是否产生一个向量捕获调试事件。 0_B 无向量捕获调试事件产生。 1_B 有向量捕获调试事件产生。 *注：对应*
EXTERNAL	4	rw	**EXTERNAL** 指示由于 EDBGRQ 有效而产生的一个异步调试事件。 0_B 无 EDBGRQ 调试事件。 1_B 有 EDBGRQ 调试事件。
0	[31:5]	r	**保留**

26.8.2 DHCSR - 调试停止控制和状态寄存器

SCS_DHCSR

控制停止调试。当 C_DEBUGEN 被置 1 时，不能在处理器运行期间 (当处理器运行时，S_HALT 为 0) 修改 C_STEP 和 C_MASKINTS。当 C_DEBUGEN 被置 0 时，处理器忽略该寄存器中所有其他位的值。

下面介绍读和写功能分离的一个寄存器，功能的改变基于对某些位的访问。

SCS_DHCSR
调试停止控制和状态寄存器 [读模式] (DF0$_H$) **复位值 : 0000 0000$_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0						S_RESET_ST	S_RETIRE_ST	0				S_LOCKUP	S_SLEEP	S_HALT	S_REGRDY
r						r	r	r				r	r	r	r

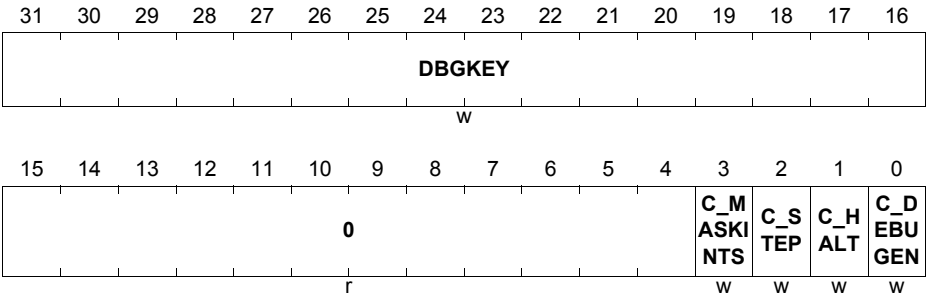
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												C_MASKINTS	C_STEP	C_HALT	C_DEBUGEN
r												r	r	r	r

域	位	类型	描述
C_DEBUGEN	0	r	**停止调试使能位** 0_B 停止调试被禁止。 1_B 停止调试被使能。 *注：如果调试器写 DHCSR 并将该位的值从 0 改变到 1，则它还必须写 0 到 C_MASKINTS 位，否则其行为是不可预见的。该位只能由正在处理器上运行的软件通过从 DAP 到 DHCSR 的访问来写入，该位不能被置位。上电复位后该位为 0 。*
C_HALT	1	r	**处理器停止位** 写该位的影响是： 0_B 请求已停止的处理器运行。 1_B 请求正运行的处理器停止。 *注：上电复位后该位未知。*
C_STEP	2	r	**处理器单步位** 写该位的影响是： 0_B 单步执行被禁止。 1_B 单步执行被使能。 *注：上电复位后该位未知。*

域	位	类型	描述
C_MASKINTS	3	r	**屏蔽 PEDSV、 SysTick 和外部的可配置中断。** 写该位的影响是： 0_B 不屏蔽 1_B 屏蔽 PendSV、SysTick 和外部的可配置中断。 任何改变该位值的尝试的影响是不可预见的，除非： • 在写 DHCSR 前， C_HALT 位的值为 1。 • 在写 DHCSR 时改变 C_MASKINTS 位，也写 1 到 C_HALT 位。 这意味着，单次写 DHCSR 不能将 C_HALT 置 0 和改变 C_MASKINTS 位的值。 当 DHCSR.C_DEBUGEN 被置 0 时，该位的值为未知。 *注： 上电复位后该位未知。*
S_REGRDY	16	r	**S_REGRDY 状态 - 一个通过 DCRDR 传送的握手标志** 如何使用： • 写 DCRSR 会将该位清 0。 • DCRDR 传送结束，然后将该位置 1。 检查 DCRDR 可获得更多信息。 0_B 已经发生了一个写 DCRDR 的操作，但传送未完成。 1_B 去往 / 来自 DCRDR 的传送完成。 *注： 当该位仅在处理器处于调试状态时有效，否则该位为未知。*
S_HALT	17	r	**S_HALT 指示处理器是否处于调试状态。** 0_B 不在调试状态。 1_B 处于调试状态。
S_SLEEP	18	r	**S_SLEEP 指示处理器是否在休眠。** 0_B 不在休眠。 1_B 在休眠。 *注： 调试器必须将 DHCSR.C_HALT 位置 1，以获取控制权，或者等待中断或其它唤醒事件来唤醒系统。*

域	位	类型	描述
S_LOCKKUP	19	r	**S_LOCKUP 指示处理器是否由于一个不可恢复的异常而被锁定。** 0_B 未锁定 1_B 已锁定。 *注：当通过一个远程调试器使用 DAP 访问时，该位只能被读作 1。值为 1 指示处理器正在运行，但由于一个无法恢复的异常而被锁定。*
S_RETIRE_ST	24	r	**S_RETIRE_ST - 当不处于调试状态时，指示自上次读 DHCSR 以来处理器是否已经执行完一条指令：** 0_B 自上次读 DHCSR 以来还没有指令被执行完 。 1_B 自上次读 DHCSR 以来至少执行完一条指令。 该位是一个粘性位，在一次读 DHCSR 的操作后被清 0。 在以下情况，该位为未知： • 一次本地复位后，但只要处理器执行完一条指令，该位就被置 1。 • 当 S_LOCKUP 被置 1 时 • 当 S_HALT 被置 1 时 *注：当处理器不处于调试状态时，调试器可以检测该位来确定处理器是否停滞在一次加载、存储或取指访中。*
S_RESET_ST	25	r	**S_RESET_ST 指示自上次读 DHCSR 以来处理器是否被复位。** 0_B 自上次读 DHCSR 以来未发生复位。 1_B 自上次读 DHCSR 以来至少发生一次复位。 该位是一个粘性位，在一次读 DHCSR 的操作后被清 0。
0	[15:4], [23:20], [31:26]	r	**保留**

SCS_DHCSR
调试停止控制和状态寄存器 [写模式] (DF0$_H$) **复位值 : 0000 0000$_H$**



域	位	类型	描述
C_DEBUGEN	0	w	**停止调试使能位** 0_B 禁止停止调试。 1_B 使能停止调试。 *注：如果调试器写 DHCSR 并将该位的值从 0 改变到 1，则它还必须写 0 到 C_MASKINTS 位，否则其行为是不可预见的。该位只能由正在处理器上运行的软件通过从 DAP 到 DHCSR 的访问来写入，该位不能被置位。上电复位后该位为 0 。。*
C_HALT	1	w	**处理器停止位** 写该位的影响是： 0_B 请求已停止的处理器运行。 1_B 请求正运行的处理器停止。 *注：上电复位后该位为未知。*
C_STEP	2	w	**处理器单步位** 写该位的影响是： 0_B 单步执行被禁止。 1_B 单步执行被使能。 *注：上电复位后该位为未知。*

域	位	类型	描述
C_MASKINTS	3	w	**屏蔽 PEDSV、 SysTick 和外部的可配置中断 .** 写该位的影响时： 0_B 不屏蔽 1_B 屏蔽 PendSV、SysTick 和外部的可配置中断。 任何改变该位值的尝试的影响是不可预见的，除非： • 在写 DHCSR 前， C_HALT 位的值为 1。 • 在写 DHCSR 时改变 C_MASKINTS 位，也写 1 到 C_HALT 位。 这意味着，单次写 DHCSR 不能将 C_HALT 置 0 和改变 C_MASKINTS 位的值。 当 DHCSR.C_DEBUGEN 置 0 时，该位的值为 未知。 上电复位后该位为未知。
DBGKEY	[31:16]	w	**DEBUG 密钥位 [31:16]!!!** 软件必须向该域写 **0xA05F** ，以使能对位 [15:0] 的写访问，否则处理器忽略该写访问。
0	[15:4]	r	**保留**

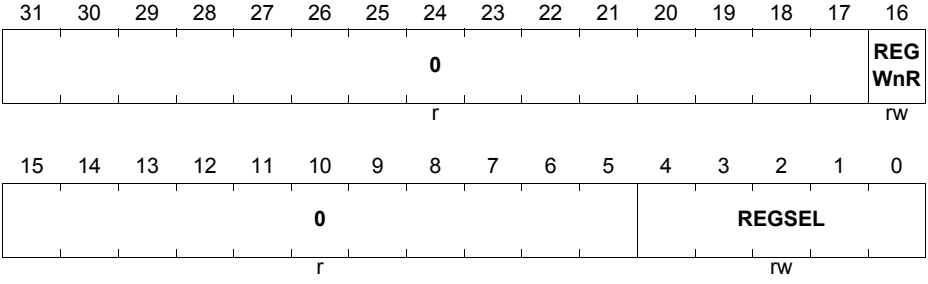
26.8.3 DCRSR - 调试内核寄存器选择器寄存器

SCS_DCRSR

DCRSR 与 DCRDR（ 调试内核寄存器数据寄存器) 一起提供对 ARM 内核寄存器和特殊寄存器的调试访问。一次对 DCRSR 的写操作指定要传送的寄存器，而不管传送是读还是写，并开始传送。该寄存器只能在调试状态访问。

SCS_DCRSR

调试内核寄存器选择器寄存器 (DF4$_H$) 复位值 : 0000 0000$_H$



域	位	类型	描述
REGSEL	[4:0]	rw	**REGSEL - 指定要传送的 ARM 内核寄存器或特殊功能寄存器** 00000_B- 01100_B ARM 内核寄存器 R0-R12。例如，0b00000 指定 R0 ，0b00101 指定 R5。 01101_B 当前 SP。另见值 0b10001 和 0b10010。 01110_B LR。 01111_B 调试返回地址。 10000_B xPSR。 10001_B 主堆栈指针， MSP。 10010_B 进程堆栈指针， PSP。 10100_B 位 [31:24] 控制；位 [23:8] 保留；位 [7:0] 为 PRIMASK。 在每个域，有效位被填满前导零。例如 DCRDR [31L26] 为 0b00000。
REGWnR	16	rw	**REGWnR 指定传送的访问类型** 0_B 读。 1_B 写
0	[31:17], [15:5]	r	**保留**

26.8.4 DCRDR - 调试内核寄存器数据寄存器

SCS_DCRDR

DCRDR 与 DCRSR (调试内核寄存器选择器寄存器) 一起工作。DCRDR 提供了对 ARM 内核寄存器和专用寄存器的调试访问。DCRDR 是这些访问的数据寄存器。

SCS_DCRDR

调试内核寄存器数据寄存器 ($DF8_H$) 复位值 : xxxx $xxxx_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
DBGTMP															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DBGTMP															
rw															

域	位	类型	描述
DBGTMP	[31:0]	rw	**DBGTMP - 数据临时缓存，用于读和写寄存器。** 该寄存器在以下情况为未知： • 复位后 • 当 DHCSR.S_HALT = 0 时 • 当 DHCSR.S_REGRDY = 0 时，在执行一个基于 DCRSR 的事务期间，该事务更新这个寄存器。

26.8.5 DEMCR - 调试异常和监视控制寄存器

SCS_DEMCR

DEMCR 的用途是管理向量捕获行为和使能 DWT。

上电复位将所有寄存器位置 0。本地复位将 DWTENA 置 0，但不影响 VC-HARDERR 或 VC_CORERESET。

SCS_DEMCR
调试异常和监视控制寄存器　　(DFC_H)　　复位值 : 0000 0000$_H$

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0							DWTENA	0							
r							rw	r							

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					VC_HARDERR	0									VC_CORERESET
r					rw	r									rw

域	位	类型	描述
VC_CORERESET	0	rw	**VC_CORERESET - 使能复位向量捕获。这会导致一次本地复位来处理一个正在运行的系统。** 0_B　禁止复位向量捕获。 1_B　使能复位向量捕获。 *注：　如果 DHCSR.C_DEBUGEN 被设置为 0，则处理器忽略该位的值。*

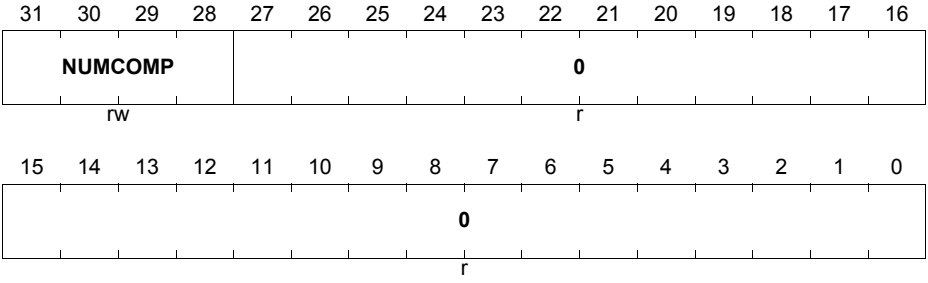
域	位	类型	描述
VC_HARDERR	10	rw	**VC_HARDERR - 使能停止调试陷阱。这会导致一次本地复位来停止一个运行的系统。** 0_B 禁止停止调试陷阱。 1_B 使能停止调试陷阱。 *注：如果 DHCSR.C_DEBUGEN 被设置为 0，则处理器忽略该位的值。*
DWTENA	24	rw	**DWTENA - 对由 DWT 单元配置的所有特性的全局使能。** 0_B 禁止 DWT。 1_B 禁止 DWT。 *注：当 DWTENA 被设置为 0 时，读 DWT 寄存器的返回值为未知。另外，当 DWTENA 为 0 时，处理器忽略对 DWT 的写操作。*
0	[31:25], [23:16], [15:11], [9:1]	r	**保留**

26.8.6 DWT_CTRL - 数据观察点控制寄存器

DWT_CTRL

DWT_CTRL 寄存器定义所实现的比较器的数量。

DWT_CTRL
数据观察点控制寄存器 **(000_H)** **复位值 : $0000\ 0000_H$**



域	位	类型	描述
NUMCOMP	[31:28]	rw	**可用的比较器数量** 0000_B不支持比较器。
0	[27:0]	r	**保留**

26.8.7 DWT_PCSR - 程序计数器采样寄存器

DWT_PCSR

DWT_PCSR 寄存器采样程序计数器的当前值。复位后该寄存器为未知。

DWT_PCSR
程序计数器采样寄存器 **($01C_H$)** **复位值 : xxxx $xxxx_H$**

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
EIASAMPLE															
rw															

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EIASAMPLE															
rw															

域	位	类型	描述
EIASAMPLE	[31:0]	rw	**EIASAMPLE** 执行指令地址的采样寄存器

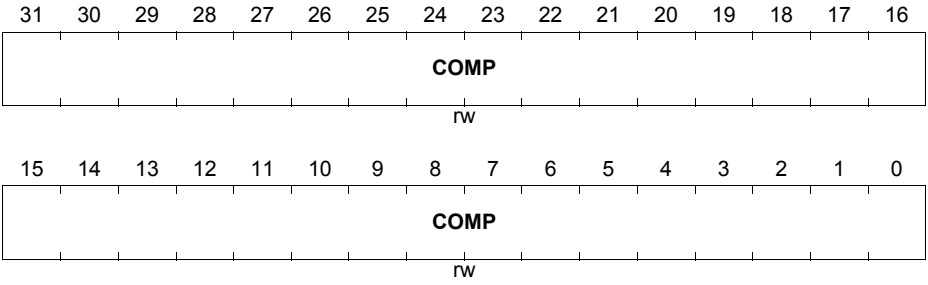
26.8.8 DWT_COMPx - DWT 比较器寄存器

DWT_COMPx

DWT_COMPx 寄存器为比较器 x 提供一个使用的参考值。复位时该值 为未知。DWT_CTRL.NUMCOMP 定义所实现的 DWT_COMPx 寄存器的数量，从 0 到 (NUMCOMP-1)。

DWT_COMP [1 .. 0] 寄存器已实现。

DWT_COMP0
DWT 比较器寄存器 0 **(020_H)** **复位值 : xxxx xxxx$_H$**
DWT_COMP1
DWT 比较器寄存器 1 **(030_H)** **复位值 : xxxx xxxx$_H$**



域	位	类型	描述
COMP	[31:0]	rw	**COMP** 用于比较的参考值

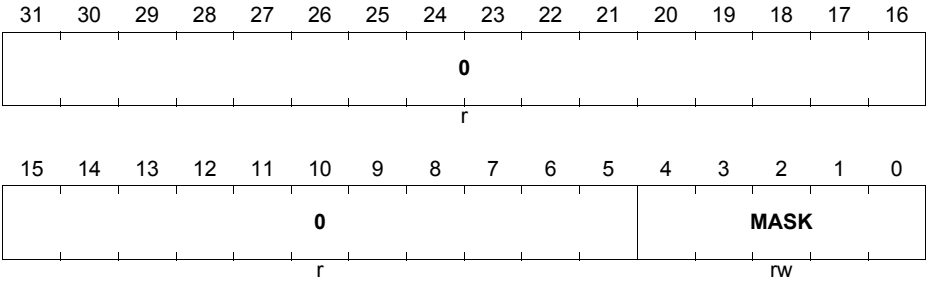
26.8.9 DWT_MASKx - DWT 比较器屏蔽寄存器

DWT_MASKx

DWT_MASKx 寄存器提供由比较器 x 应用于访问地址范围匹配的忽略掩码的大小。复位时该值为未知 。DWT_CTRL.NUMCOMP 定义所实现的 DWT_COMPx 寄存器的数量，从 0 到 (NUMCOMP-1)。

DWT_MASK [1 .. 0] 寄存器已实现。

DWT_MASK0
DWT 比较器屏蔽寄存器 (24_H) 复位值 : 0000 0000$_H$
DWT_MASK1
DWT 比较器屏蔽寄存器 (34_H) 复位值 : 0000 0000$_H$



域	位	类型	描述
MASK	[4:0]	rw	**MASK** 应用于地址范围匹配的忽略掩码的大小。向该域写全 1，然后再读该域的值可以确定所支持的最大屏蔽掩码尺寸 (并非所有屏蔽位都必须实现)。
0	[31:5]	r	**保留**

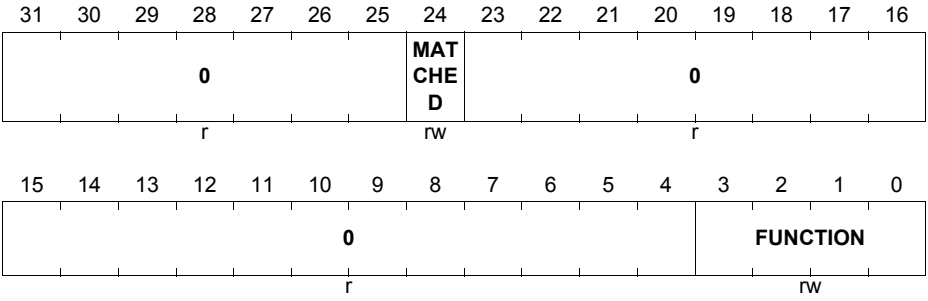
26.8.10 DWT_FUNCTIONx - 比较器功能寄存器

DWT_FUNCTIONx

DWT_FUNCTIONx 寄存器控制比较器 DWT_COMPx 寄存器的操作。DWT_CTRL.NUMCOMP 定义所实现的 DWT_FUNCTIONx 寄存器的数量，从 0 到 (NUMCOMP-1)。

DWT_FUNCTION [1 .. 0] 寄存器已实现。

DWT_FUNCTION0
比较器功能寄存器 **(28_H)** **复位值 : 0000 0000$_H$**
DWT_FUNCTION1
比较器功能寄存器 **(38_H)** **复位值 : 0000 0000$_H$**



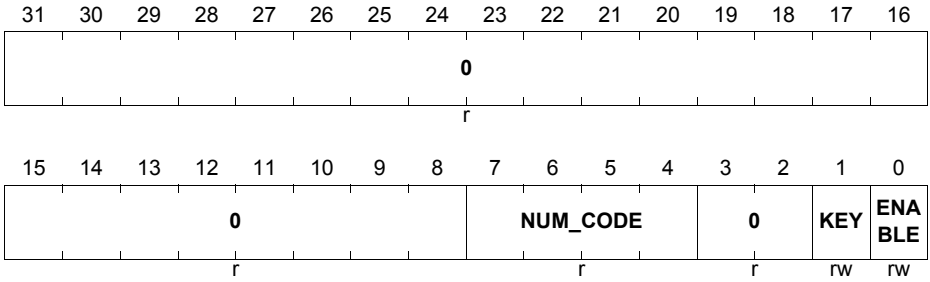
域	位	类型	描述
FUNCTION	[3:0]	rw	**FUNCTION** 选择比较器匹配的动作。 0000_B 禁止。 0001_B 保留。 0010_B 保留。 0011_B 保留。 0100_B PC 观察点事件 - 输入 laddr。 0101_B 观察点事件 - 输入 Daddr (只读) 0110_B 观察点事件 - 输入 Daddr (只写) 0111_B 观察点事件 - 输入 Daddr (读 / 写) $1xxx_B$ 保留。 *注： 上电复位后，该域被置0 。*
MATCHED	24	rw	**MATCHED** 比较器匹配。它指示自该位上次被读取后，由 FUNCTION 定义的操作已经发生。 0_B 相关的比较器发生匹配。 1_B 相关的比较器未发生匹配。 *注： 读寄存器会将该位清0。*
0	[31:25] , [23:16] , [15:4]	r	**保留**

26.8.11 BP_CTRL - 断点控制寄存器

BP_CTRL

断点控制寄存器提供 BPU 的实现信息和对 BPU 的全局使能。

BP_CTRL
断点控制寄存器 **(000_H)** **复位值 : 0000 0040_H**



域	位	类型	描述
ENABLE	0	rw	**使能 BPU** 0_B 禁止 BPU 。 1_B 使能 BPU 。 *注： 上电复位后该位为 0 。*
KEY	1	rw	**密钥 RAZ 用于读， SBO 用于写。** 如果该位被写为 0，则对该寄存器的写操作被忽略。
NUM_CODE	[7:4]	r	**NUM_CODE，断点比较器的数量**
0	[31:8], [3:2]	r	**保留**

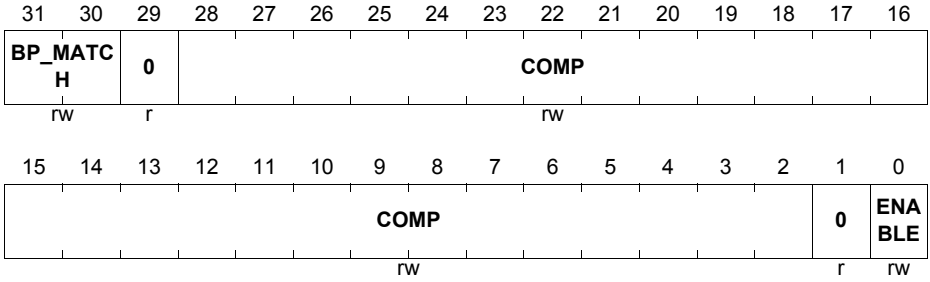
26.8.12 断点比较器寄存器

BP_COMPx

BP_COMPx 寄存器保留一个断点地址，用于与位于程序存储区中的指令地址进行比较。当 BP_CTRL.ENABLE 被置 1 时，只能有一个比较器被使能。BP_CTRL.NUM_CODE 定义所实现的 BP_COMPx 寄存器的数量，从 0 到 (NUM_CODE-1)。

BP_COMP [3 .. 0] 寄存器已实现。

BP_COMP0
断点比较器 X 寄存器 **(008_H)** **复位值 : 0000 0000$_H$**
BP_COMP1
断点比较 器 X 寄存器 **($00C_H$)** **复位值 : 0000 0000$_H$**
BP_COMP2
断点比较 器 X 寄存器 **(010_H)** **复位值 : 0000 0000$_H$**
BP_COMP3
断点比较 器 X 寄存器 **(014_H)** **复位值 : 0000 0000$_H$**



域	位	类型	描述
ENABLE	0	rw	**使能比较器** 0_B 禁止比较器。 1_B 使能比较器。 *注： 该位在上电复位后被置 0。要使能一个比较器，还必须将 BP_CTRL:ENABLE 置 1。*
COMP	[28:2]	rw	**比较地址的存储位 [28:0]。** 比较地址与来自程序存储区的地址进行比较。比较地址的位 [31:29] 和 [1:0] 都为 **0**。 0_B 禁止比较器禁止。 1_B 使能比较器使能。 *注： 该域在上电复位后为未知。*
BP_MATCH	[31:30]	rw	**BP_MATCH 定义发生 COMP 地址匹配时的行为。** 00_B 无断点匹配。 01_B 断点在下半字，上半字不受影响。 10_B 断点在上半字，下半字不受影响。 11_B 断点在上下两个半字。 *注： 该位在复位后为未知。*
0	29, 1	r	**保留**

图表目录

图目录

图目录

图目录

图目录

图目录

图目录

图目录

图目录

表目录

表目录

表目录

表目录

表目录