

# 16 位

架构

## XE167xM/XE164xM/XE 162xM 衍生产品

16 位单片实时信号控制器  
XE166 家族

### 用户手册

V2.0 2009-03

## Microcontrollers

**Edition 2009-03**

**Published by  
Infineon Technologies AG  
81726 Munich, Germany**

**© 2009 Infineon Technologies AG  
All Rights Reserved.**

#### **Legal Disclaimer**

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics. With respect to any examples or hints given herein, any typical values stated herein and/or any information regarding the application of the device, Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind, including without limitation, warranties of non-infringement of intellectual property rights of any third party.

#### **Information**

For further information on technology, delivery terms and conditions and prices, please contact the nearest Infineon Technologies Office ([www.infineon.com](http://www.infineon.com)).

#### **Warnings**

Due to technical requirements, components may contain dangerous substances. For information on the types in question, please contact the nearest Infineon Technologies Office.

Infineon Technologies components may be used in life-support devices or systems only with the express written approval of Infineon Technologies, if a failure of such components can reasonably be expected to cause the failure of that life-support device or system or to affect the safety or effectiveness of that device or system. Life support devices or systems are intended to be implanted in the human body or to support and/or maintain and sustain and/or protect human life. If they fail, it is reasonable to assume that the health of the user or other persons may be endangered.

# 16 位

架构

## XE167xM/XE164xM/XE 162xM 衍生产品

16 位单片实时信号控制器  
XE166 家族

### 用户手册

V2.0 2009-03

Microcontrollers



## XC2200M

### 版本信息: V2.0, 2009-03

先前的版本:

V1.2, 2008-09 (V1.1 略去)

V1.0, 2008-06

页	内容 (对上一版本的主要修正)
所有	两卷 (系统单元, 外设单元) 合并为一个文档
<a href="#">页 3-5</a>	增加 IMB 寄存器的描述
<a href="#">页 3-11</a> <a href="#">页 12-3</a>	增加 SBRAM 的描述
<a href="#">页 3-15</a>	增加保护位的描述 (从架构概述一章移至此处)
<a href="#">页 5-59</a>	改进位保护部分的内容
<a href="#">页 6-3</a>	增加对 256 字节最小地址分区进行 MPU 编程的提示
<a href="#">页 7-37</a>	更正 ERU 中断控制寄存器的名称
<a href="#">页 8-2</a>	更新图: 时钟产生单元
<a href="#">页 8-21</a>	更新 WUOSSCON 的描述
<a href="#">页 8-23</a>	改变位域 HPOSSCON.MODE 的描述
<a href="#">页 8-85</a>	更新一些位域的描述 (EVR 控制寄存器)
<a href="#">页 8-96</a>	改变保留位的描述 (GSCEN、GSCPERSTAT、GSCPERSTATEN)
<a href="#">页 8-119</a>	更正 EXOCON 的复位值
<a href="#">页 8-121</a>	更新图: SCU 中断结构
<a href="#">页 8-122</a>	更正唤醒中断的缩写名
<a href="#">页 8-137</a>	更新位描述
<a href="#">页 8-149</a>	增加位 WDTCS.OE 的描述注解
<a href="#">页 8-151</a>	图更新: SCU 强制中断结构
<a href="#">页 8-152</a>	更正缺省 PLL 强制中断
<a href="#">页 8-171</a>	改进 ECC 错误处理的描述



## XC2200M

版本信息: V2.0, 2009-03

页 8-183	增加寄存器 IDDMPM 和 IDDMP1
页 9-5	增加关于复位操作的描述
页 9-34	规定重叠的模拟输入通道 (ADC0/ADC1)
页 9-35	增加信号 U3C1_SELO1
页 12-29	更正引导程序加载配置表
页 16-35, 页 16-63	重新组织 GPT 定时器寄存器, 删去中断控制寄存器
页 16-69	增加寄存器表
页 16-74	增加端口控制部分的内容
页 17-9	增加信号 T14INT
页 19-26	重新组织 CCU2 寄存器, 将其集中到一节中。
页 20-35	MCC63S 的位编号更正为 6, CCPOSx 更正为 CCPOS6x
页 21-1	更新 USIC 特性
页 23-3	在附录一章 (版本差异归纳) 增加不可读唤醒计数器的内容

### 期待您的指正

本手册中如有不当、错误及遗漏之处, 敬请批评指正, 以便我们不断改进用户手册的质量。请将您的建议 (以及该手册的相关参考资料) 发送至:

[mcdocu-chinesecomments.XIY@infineon.com](mailto:mcdocu-chinesecomments.XIY@infineon.com)



**声明: 本中文用户手册是基于英文版本的翻译, 如出现与英文用户手册不符之处, 请以英文用户手册为主。**

目录:

<b>1</b>	<b>简介 .....</b>	<b>1-1</b>
1.1	16 位微控制器系列产品 .....	1-3
1.2	基本特性总结 .....	1-5
1.3	缩写 .....	1-9
1.4	命名规则 .....	1-11
<b>2</b>	<b>体系架构概况 .....</b>	<b>2-1</b>
2.1	基本 CPU 概念及优化 .....	2-2
2.1.1	存储器保护单元 (MPU) .....	2-4
2.1.2	高指令带宽/快速执行 .....	2-4
2.1.3	功能强大的执行单元 .....	2-5
2.1.4	高性能的分支、调用和循环处理 .....	2-6
2.1.5	统一及优化的指令格式 .....	2-7
2.1.6	可编程多优先级中断系统 .....	2-8
2.1.7	系统资源接口 .....	2-9
2.2	片上系统资源 .....	2-10
2.3	片上外设模块 .....	2-15
2.4	时钟产生 .....	2-30
2.5	功率管理 .....	2-31
2.6	片上调试支持 (OCDS) .....	2-32
<b>3</b>	<b>存储器结构 .....</b>	<b>3-1</b>
3.1	地址映射 .....	3-3
3.2	寄存器区 .....	3-5
3.3	数据存储区 .....	3-10
3.4	程序存储区 .....	3-12
3.4.1	程序/数据 SRAM (PSRAM) .....	3-13
3.4.2	非易失程序存储器 (Flash) .....	3-14
3.5	系统堆栈 .....	3-14
3.6	保护位 .....	3-15
3.7	IO 区 .....	3-16
3.8	外部存储空间 .....	3-16
3.9	存储器边界越界 .....	3-17
3.10	片上 Flash 存储器 .....	3-18

3.10.1	定义 .....	3-18
3.10.2	工作模式 .....	3-20
3.10.3	操作 .....	3-22
3.10.4	命令序列的详述.....	3-24
3.10.5	序列错误 .....	3-33
3.10.6	同时执行擦除和编程操作 .....	3-34
3.10.7	数据完整性.....	3-36
3.10.8	保护处理详述 .....	3-40
3.10.9	保护处理举例 .....	3-47
3.10.10	EEPROM 仿真 .....	3-48
3.10.11	中断产生 .....	3-50
3.10.12	关于优化 Flash 使用的建议 .....	3-50
3.11	片上程序存储器控制 .....	3-52
3.11.1	概述 .....	3-52
3.11.2	寄存器接口.....	3-53
3.11.3	启动, 关闭.....	3-70
3.11.4	错误报告总结 .....	3-71
3.12	数据保持存储器 .....	3-72
3.12.1	备用 RAM 访问.....	3-73
3.12.2	备用 RAM 寄存器 .....	3-74
<b>4</b>	<b>存储器检查器模块 (MCHK) .....</b>	<b>4-1</b>
4.1	操作概述 .....	4-1
4.2	功能描述 .....	4-2
4.2.1	LFSR 原理.....	4-3
4.2.2	MISR 原理.....	4-5
4.2.3	常用的多项式 .....	4-6
4.2.4	存储器检查器模块的结构 .....	4-7
4.2.5	存储器检查器模块的最佳使用 .....	4-8
4.2.6	种子值 (magic word) 的计算.....	4-9
4.2.7	应用举例 .....	4-10
4.3	存储器检查器模块寄存器 .....	4-12
4.3.1	存储器检查器输入寄存器 .....	4-13
4.3.2	存储器检查器结果寄存器 .....	4-14

4.3.3	存储器检查器计数寄存器 .....	4-15
4.3.4	存储器检查器多项式寄存器 .....	4-16
4.4	一般寄存器 .....	4-17
4.4.1	ID 寄存器 .....	4-17
4.5	MCHK 模块的接口 .....	4-17
<b>5</b>	<b>中央处理器 (CPU) .....</b>	<b>5-1</b>
5.1	CPU 的组成 .....	5-3
5.2	指令读取和程序流控制 .....	5-4
5.2.1	分支检测和分支预测规则 .....	5-6
5.2.2	零周期跳转 .....	5-6
5.2.3	Atomic 和 Extend 指令 .....	5-7
5.3	指令处理流水线 .....	5-8
5.3.1	访问 IO 区 .....	5-8
5.3.2	流水线冲突 .....	5-9
5.4	CPU 配置寄存器 .....	5-22
5.5	通用寄存器的使用 .....	5-25
5.5.1	GPR 寻址模式 .....	5-27
5.5.2	上下文切换 .....	5-29
5.6	代码寻址 .....	5-34
5.7	数据寻址 .....	5-36
5.7.1	短寻址模式 .....	5-36
5.7.2	长寻址模式 .....	5-38
5.7.3	间接寻址模式 .....	5-41
5.7.4	DSP 寻址模式 .....	5-44
5.7.5	系统堆栈 .....	5-50
5.8	标准数据处理 .....	5-54
5.8.1	16 位加法器/减法器, 阵列移位器和 16 位逻辑单元 .....	5-58
5.8.2	位操作单元 .....	5-59
5.8.3	乘除单元 .....	5-60
5.9	DSP 数据处理 (MAC 单元) .....	5-62
5.9.1	MAC 单元控制 .....	5-64
5.9.2	数字的表示和舍入 .....	5-64
5.9.3	16 位×16 位有符号/无符号乘法器和换算器 .....	5-65

**目录**

5.9.4	级联单元 .....	5-65
5.9.5	一位换算器.....	5-65
5.9.6	40 位加法/减法器 .....	5-65
5.9.7	数据限制器.....	5-65
5.9.8	累加移位器.....	5-66
5.9.9	40 位有符号累加器寄存器 .....	5-66
5.9.10	MAC 单元状态字 MSW .....	5-68
5.9.11	重复计数器 MRW .....	5-71
5.10	常数寄存器.....	5-73
<b>6</b>	<b>存储器保护单元 (MPU) .....</b>	<b>6-1</b>
6.1	功能概述 .....	6-1
6.2	存储器保护寄存器 .....	6-2
6.2.1	保护范围寄存器.....	6-3
6.2.2	保护模式寄存器.....	6-4
6.2.3	保护范围数据寄存器 .....	6-6
6.2.4	保护范围地址寄存器 .....	6-7
6.3	功能描述 .....	6-9
6.3.1	使能保护 .....	6-9
6.3.2	保护级别 .....	6-9
6.3.3	存储器范围的交集 .....	6-10
6.3.4	MPU 寄存器的保护 .....	6-10
6.3.5	访问 SFR 和 GPR .....	6-10
6.3.6	中断和 PEC 处理.....	6-11
6.3.7	RETI 指令的特殊处理.....	6-11
6.3.8	上下文切换操作.....	6-12
6.3.9	调试器访问许可.....	6-12
6.3.10	无效访问强制中断 .....	6-12
6.4	MPU 的初始化和使用.....	6-14
6.4.1	设置保护 .....	6-14
6.4.2	改变保护级别 .....	6-14
6.4.3	由不具有特权的软件执行特权代码.....	6-15
6.4.4	快速任务切换 .....	6-15
6.4.5	寄存器组选择 .....	6-15

6.4.6	调试器使用示例.....	6-15
<b>7</b>	<b>中断与异常情况控制 .....</b>	<b>7-1</b>
7.1	中断系统结构 .....	7-2
7.2	中断仲裁 .....	7-3
7.3	中断控制 .....	7-5
7.3.1	中断优先级与组优先级 .....	7-7
7.3.2	寄存器 PSW 的全局中断控制功能 .....	7-7
7.3.3	可选的中断禁止.....	7-8
7.3.4	中断类管理.....	7-9
7.4	中断向量表.....	7-10
7.5	中断跳转表缓存 .....	7-13
7.6	CPU 状态保存 .....	7-15
7.7	CPU 上下文切换 .....	7-16
7.8	快速寄存器组切换 .....	7-16
7.9	强制中断功能 .....	7-19
7.9.1	软件强制中断 .....	7-19
7.9.2	硬件强制中断 .....	7-19
7.10	外围事件控制器 .....	7-26
7.10.1	PEC 控制寄存器.....	7-26
7.10.2	PEC 源和目的指针 .....	7-32
7.10.3	PEC 中断处理总结 .....	7-35
7.10.4	PEC 通道分配 .....	7-36
7.11	外部中断 .....	7-37
7.11.1	外部请求单元 .....	7-37
7.11.2	使用外设引脚 .....	7-37
7.12	OCDS 请求 .....	7-39
7.13	服务请求的响应延迟 .....	7-39
7.14	中断节点 .....	7-41
7.14.1	物理中断节点 .....	7-41
7.14.2	中断节点共享 .....	7-45
7.14.3	中断源选择寄存器 .....	7-47
7.15	中断和 PEC 配置寄存器.....	7-49
<b>8</b>	<b>系统控制单元 (SCU) .....</b>	<b>8-1</b>

8.1	时钟产生单元 .....	8-2
8.1.1	概述 .....	8-2
8.1.2	均流控制的唤醒时钟 (OSC_WU) .....	8-3
8.1.3	高精度振荡器电路 (OSC_HP) .....	8-4
8.1.4	锁相环 (PLL) 模块 .....	8-5
8.1.5	时钟控制单元 .....	8-14
8.1.6	外部时钟输出 .....	8-18
8.1.7	CGU 寄存器 .....	8-21
8.2	系统定时器功能 (STM) .....	8-38
8.2.1	STM 寄存器 .....	8-39
8.3	唤醒定时器 (WUT) .....	8-40
8.3.1	唤醒定时器操作 .....	8-41
8.3.2	WUT 寄存器 .....	8-41
8.4	复位操作 .....	8-44
8.4.1	复位架构 .....	8-44
8.4.2	一般复位操作 .....	8-46
8.4.3	调试复位生效 .....	8-48
8.4.4	不同复位类型组合工作 .....	8-48
8.4.5	复位请求触发源 .....	8-48
8.4.6	模块复位行为 .....	8-51
8.4.7	复位控制器寄存器 .....	8-52
8.5	外部服务请求 (ESR) 引脚 .....	8-60
8.5.1	一般操作 .....	8-60
8.5.2	<b>ESR</b> 控制寄存器 .....	8-66
8.5.3	<b>ESR</b> 数据寄存器 .....	8-71
8.6	电源电压及控制 .....	8-72
8.6.1	电压看门狗 (SWD) .....	8-74
8.6.2	监控内核域的电压电平 .....	8-79
8.6.3	控制内核域的电压电平 .....	8-84
8.6.4	控制供电系统 .....	8-91
8.7	全局状态控制器 (GSC) .....	8-92
8.7.1	GSC 控制流 .....	8-92

8.7.2	GSC 寄存器 .....	8-95
8.8	软件 Boot 支持 .....	8-105
8.8.1	启动寄存器.....	8-105
8.9	外部请求单元 (ERU) .....	8-106
8.9.1	概述 .....	8-106
8.9.2	ERU 输入连接.....	8-107
8.9.3	外部请求选择单元 (ERSx) .....	8-108
8.9.4	事件触发逻辑 (ETLx) .....	8-109
8.9.5	连接矩阵 .....	8-110
8.9.6	输出门控单元 (OGUy) .....	8-111
8.9.7	ERU 输出连接.....	8-114
8.9.8	ERU 寄存器.....	8-114
8.10	SCU 中断产生 .....	8-120
8.10.1	中断支持 .....	8-121
8.10.2	SCU 中断请求源 .....	8-122
8.10.3	中断控制寄存器.....	8-123
8.11	温度补偿单元 .....	8-141
8.11.1	温度补偿寄存器.....	8-142
8.12	看门狗定时器 (WDT) .....	8-144
8.12.1	概述 .....	8-144
8.12.2	概述 .....	8-144
8.12.3	功能描述 .....	8-145
8.12.4	WDT 内核寄存器.....	8-148
8.13	SCU 强制中断的产生 .....	8-151
8.13.1	强制中断支持 .....	8-152
8.13.2	SCU 强制中断请求源 .....	8-152
8.13.3	SCU 强制中断控制寄存器 .....	8-153
8.14	存储器内容保护 .....	8-162
8.14.1	保护模式选择 .....	8-163
8.14.2	奇偶校验错误处理 .....	8-165
8.14.3	ECC 错误处理 .....	8-172
8.15	寄存器控制.....	8-176
8.15.1	寄存器访问控制.....	8-176



8.15.2	寄存器保护寄存器 .....	8-179
8.16	其它系统寄存器 .....	8-180
8.16.1	系统寄存器 .....	8-180
8.16.2	ID 模块 .....	8-181
8.16.3	标记存储器 .....	8-185
8.17	实现 .....	8-185
8.17.1	时钟产生单元 .....	8-185
8.17.2	外部服务请求 (ESR) .....	8-185
8.17.3	外部请求单元 (ERU) .....	8-187
8.18	SCU 寄存器地址 .....	8-193
<b>9</b>	<b>并行端口 .....</b>	<b>9-1</b>
9.1	一般描述 .....	9-1
9.1.1	基本端口操作 .....	9-1
9.1.2	输入级控制 .....	9-4
9.1.3	输出驱动控制 .....	9-4
9.2	端口寄存器描述 .....	9-5
9.2.1	引出端驱动控制 .....	9-5
9.2.2	端口输出寄存器 .....	9-8
9.2.3	端口输出修改寄存器 .....	9-8
9.2.4	端口输入寄存器 .....	9-10
9.2.5	端口输入/输出控制寄存器 .....	9-11
9.2.6	端口数字输入禁用寄存器 .....	9-13
9.3	端口描述 .....	9-14
9.3.1	P0 口 .....	9-14
9.3.2	P1 口 .....	9-15
9.3.3	P2 口 .....	9-16
9.3.4	P3 口 .....	9-18
9.3.5	P4 口 .....	9-19
9.3.6	P5 口 .....	9-20
9.3.7	P6 口 .....	9-20
9.3.8	P7 口 .....	9-21
9.3.9	P8 口 .....	9-22
9.3.10	P9 口 .....	9-23

9.3.11	P10 口 .....	9-24
9.3.12	P11 口 .....	9-25
9.3.13	P15 口 .....	9-26
9.4	引脚描述 .....	9-27
<b>10</b>	<b>专用引脚 .....</b>	<b>10-1</b>
<b>11</b>	<b>外部总线控制器 (EBC) .....</b>	<b>11-1</b>
11.1	特性归纳 .....	11-1
11.2	概述 .....	11-1
11.3	命名规则 .....	11-1
11.4	时序描述 .....	11-2
11.4.1	总线节拍 .....	11-2
11.4.2	非复用总线 .....	11-4
11.4.3	复用总线 .....	11-5
11.4.4	最快访问周期 .....	11-7
11.5	地址窗 .....	11-9
11.5.1	地址窗分配 .....	11-9
11.5.2	地址窗重叠 .....	11-10
11.6	READY 控制的总线周期 .....	11-11
11.6.1	使能 READY 控制 .....	11-11
11.6.2	同步和异步 READY .....	11-11
11.6.3	READY 功能与预定义的等待周期相结合 .....	11-12
11.7	外部总线仲裁 .....	11-12
11.7.1	仲裁初始化 .....	11-13
11.7.2	仲裁主设备方案 .....	11-13
11.7.3	仲裁从设备方案 .....	11-15
11.7.4	总线锁定功能 .....	11-15
11.7.5	直接连接主设备和从设备 .....	11-15
11.8	EBC 空闲状态 .....	11-16
11.9	寄存器描述 .....	11-17
11.9.1	EBC 模式寄存器 .....	11-17
11.9.2	时序控制寄存器 .....	11-20
11.9.3	功能控制寄存器 .....	11-22
11.9.4	地址窗选择寄存器 .....	11-23

<b>11.10</b>	<b>XE16xyM 中 EBC 的实现 .....</b>	<b>11-24</b>
11.10.1	未使用的寄存器.....	11-24
11.10.2	对 LXBUS 模块的访问控制.....	11-24
11.10.3	关闭控制 .....	11-24
11.10.4	专用寄存器.....	11-25
<b>12</b>	<b>启动配置和引导程序加载.....</b>	<b>12-1</b>
12.1	启动模式选择 .....	12-1
12.2	启动之后的器件状态 .....	12-2
12.2.1	被启动过程修改的寄存器 .....	12-2
12.2.2	启动之后的系统频率 .....	12-4
12.2.3	看门狗定时器的处理 .....	12-4
12.2.4	启动错误状态 .....	12-5
12.3	支持的启动模式和选择.....	12-5
12.3.1	基本启动模式 .....	12-6
12.3.2	支持调试的启动模式 .....	12-7
12.3.3	特殊启动特性 .....	12-9
12.4	内部启动 .....	12-14
12.5	外部启动 .....	12-14
12.5.1	特定设置 .....	12-15
12.6	引导程序加载 .....	12-17
12.6.1	一般功能 .....	12-17
12.6.2	使用 UART 协议进行引导程序加载 .....	12-18
12.6.3	同步串行通道引导程序加载.....	12-24
12.6.4	CAN 引导程序加载.....	12-26
12.6.5	引导程序加载模式归纳.....	12-29
<b>13</b>	<b>调试系统.....</b>	<b>13-1</b>
13.1	调试接口 .....	13-2
13.1.1	调试信号连接 .....	13-3
13.2	OCDS 模块 .....	13-5
13.2.1	调试事件 .....	13-7
13.2.2	调试动作 .....	13-8
13.3	Cerberus.....	13-9
13.3.1	功能概述 .....	13-9

<b>13.4</b>	<b>仿真器件 .....</b>	<b>13-10</b>
<b>13.4.1</b>	<b>MCDS 使用 .....</b>	<b>13-11</b>
<b>13.4.2</b>	<b>MCDS 特性 .....</b>	<b>13-11</b>
<b>13.5</b>	<b>边界扫描 .....</b>	<b>13-13</b>
<b>14</b>	<b>指令集概述.....</b>	<b>14-1</b>
<b>15</b>	<b>器件规范 .....</b>	<b>15-1</b>

<b>16</b>	<b>通用定时器单元 .....</b>	<b>16-1</b>
16.1	定时器模块 GPT1 .....	16-2
16.1.1	GPT1 核心定时器 T3 的控制 .....	16-4
16.1.2	GPT1 核心定时器 T3 的工作模式 .....	16-9
16.1.3	GPT1 辅助定时器 T2/T4 的控制 .....	16-17
16.1.4	GPT1 辅助定时器 T2/T4 的工作模式 .....	16-22
16.1.5	GPT1 时钟信号控制 .....	16-32
16.1.6	GPT1 定时器寄存器 .....	16-35
16.1.7	GPT1 定时器的中断控制 .....	16-36
16.2	定时器模块 GPT2 .....	16-37
16.2.1	GPT2 核心定时器 T6 的控制 .....	16-39
16.2.2	GPT2 核心定时器 T6 的工作模式 .....	16-43
16.2.3	GPT2 辅助定时器 T5 的控制 .....	16-46
16.2.4	GPT2 辅助定时器 T5 的工作模式 .....	16-49
16.2.5	GPT2 寄存器 CAPREL 工作模式 .....	16-54
16.2.6	GPT2 时钟信号控制 .....	16-59
16.2.7	GPT2 定时器寄存器 .....	16-63
16.2.8	GPT2 定时器和 CAPREL 的中断控制 .....	16-64
16.3	其它寄存器 .....	16-64
16.4	寄存器表 .....	16-69
16.5	GPT 模块接口 .....	16-71
<b>17</b>	<b>实时时钟 .....</b>	<b>17-1</b>
17.1	确定 RTC 的时间基准 .....	17-3
17.2	RTC 运行控制 .....	17-7
17.3	RTC 工作模式 .....	17-9
17.4	48 位定时器操作 .....	17-13
17.5	系统时钟操作 .....	17-13
17.6	周期性中断的产生 .....	17-14
17.7	RTC 中断产生 .....	17-14
17.8	其它寄存器 .....	17-17
<b>18</b>	<b>模数转换器 .....</b>	<b>18-1</b>
18.1	简介 .....	18-1
18.1.1	ADC 框图 .....	18-1

18.1.2	特性集 .....	18-2
18.1.3	缩写 .....	18-3
18.1.4	ADC 内核概述 .....	18-4
18.1.5	转换请求单元 .....	18-5
18.1.6	转换结果单元 .....	18-7
18.1.7	中断结构 .....	18-7
18.1.8	电气模型 .....	18-8
18.1.9	转换特性和误差定义 .....	18-10
18.2	操作 ADC .....	18-11
18.2.1	寄存器概述 .....	18-12
18.2.2	模式控制 .....	18-15
18.2.3	模块激活和省电模式 .....	18-17
18.2.4	时钟方案 .....	18-17
18.2.5	通用 ADC 寄存器 .....	18-18
18.2.6	请求源仲裁器 .....	18-26
18.2.7	仲裁器寄存器 .....	18-30
18.2.8	扫描请求源处理 .....	18-31
18.2.9	扫描请求源寄存器 .....	18-34
18.2.10	顺序请求源处理 .....	18-38
18.2.11	顺序请求源寄存器 .....	18-42
18.2.12	通道相关功能 .....	18-49
18.2.13	通道相关寄存器 .....	18-53
18.2.14	转换结果的处理 .....	18-61
18.2.15	转换结果相关寄存器 .....	18-67
18.2.16	CH7 的复用器测试模式 .....	18-76
18.2.17	外部复用器控制 .....	18-76
18.2.18	支持并行采样的同步转换 .....	18-78
18.2.19	等间隔采样 .....	18-82
18.2.20	断线检测 .....	18-83
18.2.21	附加特性寄存器 .....	18-84
18.3	XE16xyM 中 ADC 的实现 .....	18-89
18.3.1	地址映射 .....	18-90
18.3.2	中断控制寄存器 .....	18-90

18.3.3	模拟信号的连接.....	18-91
18.3.4	数字信号的连接.....	18-93
<b>19</b>	<b>捕获/比较单元.....</b>	<b>19-1</b>
19.1	功能概述.....	19-2
19.1.1	CAPCOM 定时器.....	19-4
19.1.2	定时器中断.....	19-7
19.1.3	捕获/比较通道.....	19-7
19.1.4	捕获模式.....	19-8
19.1.5	比较模式操作.....	19-9
19.1.6	双寄存器比较模式.....	19-16
19.1.7	CAPCOM 中断.....	19-19
19.1.8	比较输出信号的产生.....	19-19
19.1.9	单次事件操作.....	19-20
19.1.10	交错和非交错操作.....	19-20
19.1.11	对外部输入信号的要求.....	19-25
19.2	CAPCOM2 寄存器.....	19-26
19.2.1	ID 寄存器.....	19-29
19.2.2	定时器 T7/T8 寄存器.....	19-29
19.2.3	定时器 T7/T8 控制寄存器.....	19-31
19.2.4	捕获/比较寄存器.....	19-32
19.2.5	捕获/比较模式寄存器.....	19-33
19.2.6	比较输出寄存器.....	19-35
19.2.7	双寄存器比较模式寄存器.....	19-35
19.2.8	IOC 寄存器.....	19-36
19.2.9	单次事件模式寄存器.....	19-37
19.2.10	其它寄存器.....	19-38
19.3	模块实现.....	19-40
19.3.1	CAPCOM2 单元接口.....	19-40
<b>20</b>	<b>捕获/比较单元 6 (CCU6) .....</b>	<b>20-1</b>
20.1	简介.....	20-1
20.1.1	特性概述.....	20-2
20.1.2	框图.....	20-3
20.1.3	寄存器概述.....	20-4

20.2	定时器 T12 模块.....	20-6
20.2.1	T12 概述.....	20-7
20.2.2	T12 计数策略 .....	20-10
20.2.3	T12 的比较模式.....	20-14
20.2.4	比较模式输出路径 .....	20-19
20.2.5	T12 捕获模式 .....	20-23
20.2.6	T12 映射寄存器传送.....	20-28
20.2.7	定时器 T12 工作模式选择.....	20-30
20.2.8	T12 相关寄存器.....	20-31
20.2.9	捕获/比较控制寄存器 .....	20-35
20.3	定时器 T13 模块.....	20-47
20.3.1	T13 模块概述 .....	20-47
20.3.2	T13 计数策略 .....	20-49
20.3.3	T13 的比较模式.....	20-53
20.3.4	比较模式输出路径 .....	20-56
20.3.5	T13 映射寄存器传送.....	20-57
20.3.6	T13 相关寄存器.....	20-59
20.4	强制中断处理 .....	20-62
20.5	多通道模式.....	20-63
20.6	霍尔传感器模式 .....	20-65
20.6.1	霍尔序列评估 .....	20-67
20.6.2	霍尔序列比较逻辑.....	20-69
20.6.3	霍尔模式标志位.....	20-70
20.6.4	实现无刷直流电机控制的霍尔模式 .....	20-71
20.7	调制控制寄存器 .....	20-74
20.7.1	调制控制 .....	20-74
20.7.2	强制中断控制寄存器 .....	20-76
20.7.3	被动态电平寄存器 .....	20-79
20.7.4	多通道模式寄存器 .....	20-80
20.8	中断处理 .....	20-85
20.8.1	中断结构 .....	20-85
20.8.2	中断寄存器.....	20-87
20.9	一般模块操作 .....	20-98



20.9.1	模式控制 .....	20-98
20.9.2	输入选择 .....	20-101
20.9.3	一般寄存器 .....	20-101
20.10	CCU6 的实现 .....	20-109
20.10.1	地址映射 .....	20-109
20.10.2	中断控制寄存器 .....	20-110
20.10.3	同步启动特性 .....	20-111
20.10.4	数字信号的连接 .....	20-111
<b>21</b>	<b>通用串行接口通道 .....</b>	<b>21-1</b>
21.1	简介 .....	21-1
21.1.1	特性概述 .....	21-1
21.1.2	通道结构 .....	21-4
21.1.3	输入级 .....	21-4
21.1.4	输出信号 .....	21-5
21.1.5	波特率发生器 .....	21-6
21.1.6	通道事件和中断 .....	21-7
21.1.7	数据移位和处理 .....	21-7
21.2	操作 USIC .....	21-10
21.2.1	寄存器概述 .....	21-11
21.2.2	操作 USIC 通信通道 .....	21-15
21.2.3	通道控制和配置寄存器 .....	21-22
21.2.4	协议相关寄存器 .....	21-29
21.2.5	操作输入级 .....	21-32
21.2.6	输入级寄存器 .....	21-35
21.2.7	操作波特率发生器 .....	21-37
21.2.8	波特率发生器寄存器 .....	21-41
21.2.9	操作发送数据通路 .....	21-45
21.2.10	操作接收数据通路 .....	21-48
21.2.11	传送控制和状态寄存器 .....	21-50
21.2.12	数据缓存寄存器 .....	21-59
21.2.13	操作 FIFO 数据缓存 .....	21-67
21.2.14	FIFO 缓存和旁路寄存器 .....	21-76
21.3	异步串行通道 (ASC = UART) .....	21-93

**目录**

21.3.1	信号描述 .....	21-93
21.3.2	帧格式 .....	21-95
21.3.3	操作 ASC .....	21-97
21.3.4	ASC 协议寄存器 .....	21-104
21.3.5	硬件 LIN 支持 .....	21-110
21.4	同步串行通道 (SSC) .....	21-111
21.4.1	信号描述 .....	21-112
21.4.2	操作 SSC .....	21-118
21.4.3	主控模式下操作 SSC .....	21-120
21.4.4	从控模式下操作 SSC .....	21-125
21.4.5	SSC 协议寄存器 .....	21-127
21.4.6	SSC 时序 .....	21-132
21.5	IIC 总线协议 (IIC) .....	21-134
21.5.1	简介 .....	21-135
21.5.2	操作 IIC .....	21-137
21.5.3	符号时序 .....	21-143
21.5.4	数据流处理 .....	21-146
21.5.5	IIC 协议寄存器 .....	21-150
21.6	IIS 协议 .....	21-155
21.6.1	简介 .....	21-156
21.6.2	操作 IIS .....	21-159
21.6.3	主控模式下操作 IIS .....	21-163
21.6.4	从控模式下操作 IIS .....	21-166
21.6.5	IIS 协议寄存器 .....	21-167
21.7	XE16xyM 中 USIC 的实现 .....	21-172
21.7.1	模块实现概览 .....	21-172
21.7.2	通道特性 .....	21-173
21.7.3	地址映射 .....	21-174
21.7.4	模块 ID 寄存器 .....	21-175
21.7.5	中断控制寄存器 .....	21-176
21.7.6	输入/输出连接 .....	21-178
<b>22</b>	<b>控制器局域网络 (MultiCAN) 控制器 .....</b>	<b>22-1</b>
22.1	MultiCAN 简述 .....	22-1

**目录**

22.1.2	CAN 特性 .....	22-2
22.2	CAN 功能描述 .....	22-3
22.2.1	命名惯例和定义 .....	22-3
22.2.2	简介 .....	22-3
22.2.3	CAN 节点控制 .....	22-7
22.2.4	报文对象列表结构 .....	22-11
22.2.5	CAN 节点分析模式 .....	22-14
22.2.6	报文验收滤波 .....	22-17
22.2.7	报文后处理接口 .....	22-19
22.2.8	报文对象数据处理 .....	22-22
22.2.9	报文对象功能 .....	22-28
22.3	MultiCAN 内核寄存器 .....	22-36
22.3.1	寄存器地址映射 .....	22-36
22.3.2	全局 MultiCAN 寄存器 .....	22-41
22.3.3	CAN 节点寄存器 .....	22-52
22.3.4	报文对象寄存器 .....	22-67
22.4	通用控制和状态 .....	22-86
22.4.1	时钟控制 .....	22-86
22.4.2	端口输入控制 .....	22-87
22.4.3	挂起模式 .....	22-88
22.4.4	中断结构 .....	22-88
22.5	MultiCAN 模块的实现 .....	22-89
22.5.1	CAN 模块接口 .....	22-89
22.5.2	模块时钟产生 .....	22-90
22.5.3	模式控制行为 .....	22-101
22.5.4	模式控制 .....	22-101
22.5.5	模块控制寄存器描述 .....	22-103
22.5.6	外部信号的连接 .....	22-106
22.5.7	MultiCAN 模块寄存器地址映射 .....	22-111

## 1 简介

嵌入式控制应用领域的快速增长，对当今微控制器的实时性提出更为严格的要求。为了实现复杂的控制算法，就必须能够处理大量的数字和模拟输入信号，并在规定的最大响应时间内产生正确的输出信号。同时，嵌入式控制应用通常要求小的板卡空间、较低功耗以及较低的系统成本。

嵌入式控制应用要求微控制器具备以下功能：

- 提供高度系统集成
- 无需附加外设器件及相关软件开销
- 提供系统安全和故障保护机制
- 提供有效措施控制（并降低）器件功耗

随着嵌入式控制系统的复杂度日益增加，新型高端嵌入式控制系统要求微控制器具有性能更高的 CPU 和功能更强大的外设集。因此，英飞凌决定开发 16/32 位 CMOS 微控制器系列，而不受先前架构后向兼容性的限制。

虽然该系列微控制器采用新架构，其仍沿用流行的英飞凌 8 位微控制器系列中成熟的硬件和软件概念，该系列产品具有 32 位微控制器的性能。

该系列微控制器的卓越性能已经成为各种应用领域系统解决方案的基础，该系列还具有灵活外设模块和高效的功率控制特性。上述特性是功能强大、高效的片上系统的必备条件。

### 工业系统解决方案

XE166 衍生系列具有 DSP 功能、通用控制能力以及复杂的外设模块，可为多种工业应用提供解决方案，因此该系列微控制器适用于下面各种应用领域：

- 电机控制（泵、风机、压缩机、伺服系统、数控机床、机器人、进程控制、传送带、...）
- 可再生能源（风能转换器、光电、燃料电池、蓄电池、水力发电机、微型涡轮机、...）
- 电源（不间断电源 UPS、通用电源、电池充电器、电灯镇流器、...）
- 交通运输（机车、火车、地铁、公交车、拖拉机、叉车、农用车、交通灯、...）

英飞凌公司的高质量标准使得 XE166 实时控制器成为稳健和可靠系统的理想选择。

## 关于本手册

本用户手册描述英飞凌 **XE166** 系列微控制器的功能。

这些微控制器的功能基本一致，但是各型号的产品又具备独特的特性。

本手册涵盖了衍生器件的所有特性。

各衍生产品特性的详细列表请参考相关数据手册。

本手册适用于所有衍生产品，并给出所有型号产品的可用温度范围和封装形式。

为简化起见，各型号器件在本手册中统一用 **XE16xyM** 表示，在相关数据手册中给出符合欧洲电子联盟标准的器件完整型号名。

本手册中的某些章节不适用于现有或计划开发的 **XE16xyM** 的全部衍生产品（如不同器件会集成不同类型的片上存储器或片上外设）。在这些章节中，将尽可能添加注释予以说明。

## 1.1 16 位微控制器系列产品

英飞凌 16 位微控制器系列可满足实时嵌入式控制应用的高性能需求。其体系架构经过优化设计，从而具有更高的指令吞吐能力，而且对外部激励（中断）的响应时间更短。该系列产品集成了智能的外设子系统，从而最大程度的降低了对 CPU 干预的需求，同时最大程度降低了通过外部总线接口进行通信的需要。该架构具有高度的灵活性，可满足诸如汽车、工业控制或数据通信等不同应用领域的各种需求。

16 位微控制器系列内核的开发采用模块化概念。所有产品均采用同一个控制优化的高效指令集（第二代产品提供附加指令）。尽管各种新型系列产品的内部存储器容量不同、工艺不同、片上外设集不同和/或 IO 引脚数目不同，相同的指令集可使它们的应用变得简单方便。

系统除集成标准片上外设之外，还可通过 XBUS/LXBus 概念（外部总线的内部标识）直接集成某些应用特有的外设模块，从而构造出某些应用的专用衍生芯片。

嵌入式控制应用的编程规模日益增大，由于高级语言更易于编写、调试和维护，因此越来越受到编程人员的青睐。英飞凌 C166 系列从第二代产品开始支持高级语言。

80C166 微控制器是英飞凌 16 位微控制器系列的第一代产品，建立了 C166 的体系架构。

C165 和 C167 是第二代 16 位产品，具有支持高级语言编程（HLL）的附加指令集，扩展了地址空间、增加了内部 RAM 容量，并增强了外部总线上各种资源的高效管理，从而该系列产品具有更强大的功能。

第二代增强型产品具有更多特性，如附加内部高速 RAM、片上 CAN 模块和片上锁相环（PLL）等。

高效系统的设计可能需要集成专用外设，提升系统性能的同时使整个系统元件数最少。英飞凌 16 位微控制器（第二代）产品定义了 XBUS 以支持专用外设的集成。XBUS 是外部总线的内部标识，通过将所需接口标准化，开放并简化了外设的集成。该技术的典型应用即 CAN 模块的集成。

C165 系列产品是 C167 的功能简化版本，该系列未集成 ADC、CAPCOM 单元和 PWM 模块，因此其封装更小、功耗更低，设计更为简单。

C164 系列产品（C167CS 的衍生产品）以及部分 C161 系列产品进一步增强了灵活功率管理特性，构成的第三代 16 位产品。该功率管理机制有效控制了微控制器在特定状态下的功耗，从而使得给定应用的总功耗最低。

XC16x 系列是第四代 16 位微控制器的代表产品。通过一些主要改进和新模块的添加，XC166 系列产品显著提高了 16 位微控制器的系统性能。MAC 单元加入 DSP 功能以处理数字滤波器算法，大大缩短了乘/除运算的时间；还增加了 32 位操作和 40 位累加器。五级流水线结构、大多数单周期指令、可遍寻整个地址空间的 PEC 传送等特性均有效提高了系统性能。先前的 XBUS 被优化的 LXBus 取代。可通过片上调试支持单元（OCDS）对目标系统进行调试。

如今 XE166 系列为**第五代 16 位**产品。该系列具有 32 位性能，使得用户和应用通过一些步骤即可实现片上工业目标系统。集成的存储器和各种外设可实现紧凑系统，片上集成的内核电源和控制使得系统仅需要单电源供电。优化的编译器可支持 CPU 与 MAC 单元的强大功能组合，从而使得 XE166 和 32 位系统的性能之间不存在任何差距。

此外，一些产品还集成有专用单元。

英飞凌已开发了各种不同封装、不同温度范围和不同速度级别的产品。

更多的标准产品和专用产品正在规划和开发中。

*注：并非所有的衍生产品都有对应各种温度范围、速度级别、封装或不同程序存储器的产品。*

每种器件都有各自的版本和器件信息。请通过英飞凌代理商获取产品的最新资料，或登陆英飞凌产品网站查询 <http://www.infineon.com/microcontrollers>。

*注：由于目前推出的大多数 XE16xyM 系列产品的体系架构和基本特性（如 CPU 内核和片上外设）均相同，因此除非特别指明，本手册中“XE16xyM”的描述也适用于其它衍生产品。*

## 1.2 基本特性总结

XE16xyM 是英飞凌 XE166 系列全功能单片 CMOS 微控制器的增强型产品。

注：各型号器件在本手册中统称为 XE16xyM。

XE16xyM 将功能和性能扩展的 C166SV2 内核、功能强大的片上外设子系统和各种片上存储器完美结合，并提供几种降低功耗的方法。

高性能 XE16xyM 具有以下主要特性：

### 智能片上外设子系统

- 两个可同步 ADC，转换精度（10 位或 8 位）和转换时间（降低至 1  $\mu$ s）可编程设定，具有多达 24 路模拟输入通道，自动扫描模式和通道插入和数据压缩特性。
- 一个捕获/比较单元（具有两个独立时间基准），具有不同的工作模式下的灵活 PWM 单元/事件记录单元：共包括两个 16 位定时器/计数器，最大精度  $f_{sys}$
- 多达四个用于灵活 PWM 信号产生的捕获/比较单元（CCU6）（3/6 捕获/比较通道和 1 个比较通道）
- 两个多功能通用定时器单元：
  - GPT1：含 3 个 16 位定时器/计数器，最大精度  $f_{sys}/4$
  - GPT2：含 2 个 16 位定时器/计数器，最大精度  $f_{sys}/2$
- 多达 8 个串行通道，具有波特率产生、接收/发送 FIFO、数据长度和移位方向可编程，可用作 UART、类 SPI、IIC、IIS 和 LIN 接口
- 控制器局域网络（MultiCAN）模块，支持 V2.0B active，多达 6 个 CAN 节点，可独立工作或通过网关交换数据，全功能 CAN/基本 CAN
- 具有闹铃中断的实时时钟
- 刷新时间间隔可编程设定的看门狗定时器
- 引导程序加载器，可灵活进行系统初始化
- 系统配置和控制寄存器的保护管理

### 集成的片上存储器

- 8 KB 片上 Stand-By（备用）RAM（SBRAM），用于存储数据
- 高达 2 KB 的双口 RAM（DPRAM），用于存储变量、寄存器组和堆栈
- 高达 16 KB 的片上高速数据 SRAM（DSRAM），用于存储变量和堆栈
- 高达 32 KB 的片上高速程序/数据 SRAM（PSRAM），用于存储代码和数据
- 高达 576 KB 片上 Flash 程序存储器，用于存储代码或常量数据。



注：系统堆栈可位于整个地址空间的任何区域。

### **具有五级流水线和 MAC 单元的高性能 16 位 CPU**

- 单时钟周期指令执行
- 大多数指令为单时钟周期指令
- 单周期的乘法运算（16 位 × 16 位）
- 4+17 周期的除法运算（32 位/16 位），4 周期延时，17 周期后台执行
- 单周期的乘累加指令（MAC）执行
- 32 位加法和 32 位减法（MAC 单元）
- 40 位阵列移位器和 40 位累加器
- 自动饱和或舍入运算
- 多种高带宽的内部数据总线
- 基于寄存器的设计，具有多个可变寄存器组
- 两个附加的快速寄存器组
- 支持快速上下文切换
- 16 MB 线性代码和数据地址空间（冯诺伊曼架构）
- 带有堆栈上溢/下溢自动检测的系统堆栈缓存
- 高性能分支、调用和循环处理
- 零周期跳转指令

### **针对控制应用的高效指令集**

- 位、字节和字数据类型
- 灵活高效的寻址模式，支持高代码密度
- 增强的布尔位操作，具有 6 Kbit 可直接位寻址的地址空间，用作外设控制和用户自定义的标志位
- 硬件强制中断，用于识别运行时的异常情况
- HLL 支持信号操作和高效数据访问

### **功率管理特性**

- 两个 IO 电源域，满足系统从 3 V 到 5 V 的供电需求
- 用于改善功耗和 EMC 的门控时钟概念
- 可通过时钟产生单元编程降低系统运行速度
- 灵活的外设管理，每个模块可单独被禁用

- 频率输出可编程

## **16 级优先级的中断系统**

- 96 个具有独立中断向量的中断节点，16 个优先级（8 个组优先级）
- 执行内部程序时，最短中断延迟为 7 个周期
- 快速外部中断
- 外部中断源可编程选择
- 中断向量表可编程（起始地址和步长）

## **8 通道外围事件控制器（PEC）**

- 由中断驱动的单周期数据传送
- PEC 中断请求优先级可编程（从 15 级到 8 级）
- 传送计数选择  
（执行设定次数的 PEC 传送之后转入标准 CPU 中断）
- 消除了响应中断请求时保存和恢复系统状态的开销
- 24 位源和目的地址指针，支持整个地址空间内的数据传送

## **片上调试支持**

- 通过 DAP 接口（2 线）或 JTAG 接口（5 线）进行通信
- 具有可选断点接口的片上调试控制器
- 硬件、软件和外部引脚断点
- 最多 4 个指令指针断点
- 调试事件控制，如调用监控程序、暂停 CPU、或触发数据传送
- 由 DAP/JTAG 接口控制的专用 DEBUG 指令
- 通过 DAP/JTAG 接口访问任意内部寄存器或存储器地址
- 利用 MOV 插入指令实现单步调试和观察点

## **具有独立位寻址能力的输入/输出线**

- 输入模式三态
- 输出模式推挽或漏极开路
- 端口驱动控制可编程
- 两个 I/O 电源域，电源范围从 3.0 V 到 5.5 V  
（内核逻辑和振荡器输入电压为 1.5 V）

**英飞凌 CMOS 工艺**

- 低功耗 CMOS 工艺，具有灵活的功率管理特性，可实现各种省电模式

**绿色塑料小外形四方扁平（LQFP）封装**

- PG-LQFP-144，本体尺寸 20×20 mm，0.5 mm（19.7 mil）引脚间距，表面贴装工艺
- PG-LQFP-100，本体尺寸 14×14 mm，0.5 mm（19.7 mil）引脚间距，表面贴装工艺
- PG-LQFP-64，本体尺寸 10×10 mm，0.5 mm（19.7 mil）引脚间距，表面贴装工艺

**完整的开发工具支持**

英飞凌借助第三方为用户提供微控制器的开发工具。从本地的小生产商到拥有各种产品的跨国公司，全球的众多开发工具供应商可为英飞凌的所有产品提供功能强大的开发工具，这确保了用户可尽早获取各种不同性价比以及高质量的关键工具，包括编译器、汇编器、仿真器、调试器或在线仿真器。

英飞凌在产品开发的初期就与开发工具战略伙伴进行合作，以保证嵌入式系统开发人员可获得可靠、易于使用的开发工具解决方案，帮助开发人员以最有效的方式在最短的时间内学会使用英飞凌的微控制器产品。

英飞凌 16 位微控制器的开发工具环境包括：

- 编译器（C 和 C++）
- 宏汇编器、连接器、定位器、库管理工具和格式转换工具
- 体系架构仿真器
- HLL 调试器
- 实时操作系统
- 在线仿真器
- 逻辑分析仪反汇编器
- 入门开发套件和评估板
- 芯片配置代码生成工具（DAVE）

## 1.3 缩写

本手册中出现的缩写归纳如下：

ADC	模数转换器
ALE	地址锁存使能
ALU	算术逻辑单元
ASC	异步/同步串行通道
CAN	控制器局域网络（License Bosch）
CAPCOM	捕获和比较单元
CISC	复杂指令集计算
CMOS	互补金属氧化物半导体
CPU	中央处理单元
DAP	器件访问端口
DMU	数据管理单元
EBC	外部总线控制器
ECC	纠错码
ESFR	扩展特殊功能寄存器
EVVR	内置已验证电压调节器
Flash	电可擦除非易失存储器
GPR	通用寄存器
GPT	通用定时器单元
HLL	高级语言
IIC	内部集成电路（总线）
IIS	内部音频集成电路（总线）
IO	输入/输出
JTAG	联合测试行动组
LIN	本地互连网络

LPR	低功率电源参考
LQFP	小外形四方扁平封装
LXBus	外部总线的内部标识
MAC	乘/累加（单元）
MPU	存储器保护单元
OCDS	片上调试支持
OTP	一次性可编程存储器
PEC	外围事件控制器
PLA	可编程逻辑阵列
PLL	锁相环
PMU	程序管理单元
PVC	电源验证电路
PWM	脉冲宽度调制
RAM	随机访问存储器
RISC	精简指令集计算
ROM	只读存储器
RTC	实时时钟
SFR	特殊功能寄存器
SSC	同步串行通道
SWD	电源看门狗
UART	通用异步接收/发送器
USIC	通用串行接口通道

## **1.4 命名规则**

用于控制功能和指示状态的位域、以及存放这些位域的寄存器都对应有唯一的名称。因此在描述位域和寄存器时，通常引用它们的名称，而不引用它们的存放位置，从而使用户更易理解。

描述具有规则结构的寄存器（如端口）时，采用数字编号来描述各位，而不采用大量相似的位名称，例如 P5 口的位 3 通常称为 P5.3。

为了阐明某些命名结构之间的关系，会给相应的名称添加第二级名称，从而使描述更加明确。

ADC0\_GLOBCTR 明确表示表明寄存器 GLOBCTR 是模块 ADC0 的组成部分；  
SYSCON0.CLKSEL 明确表示位域 CLKSEL 是寄存器 SYSCON0 的组成部分。

## 2 体系架构概况

XE16xyM 的内核架构将 RISC 和 CISC 处理器的优点完美结合。因为 MAC 单元的 DSP 功能，其计算和控制功能变得更加完善。XE16xyM 将高性能 CPU 和一组功能强大的外设单元以高效的方式互连，并集成到一颗芯片中。片上存储器模块具有专用总线和控制单元，用于存储代码和数据。这些特性结合起来，使得 XE16xyM 微控制器具有极高的性能，不仅能够满足当前应用的需要，而且可以应对未来的工程应用的挑战。同时 XE16xyM 还采用了一种 LxBus 总线（外部总线接口的内部标识）。LxBus 总线提供了一种标准化的方法，可方便的将附加的专用外设集成到标准 XE16xyM 系列芯片中。

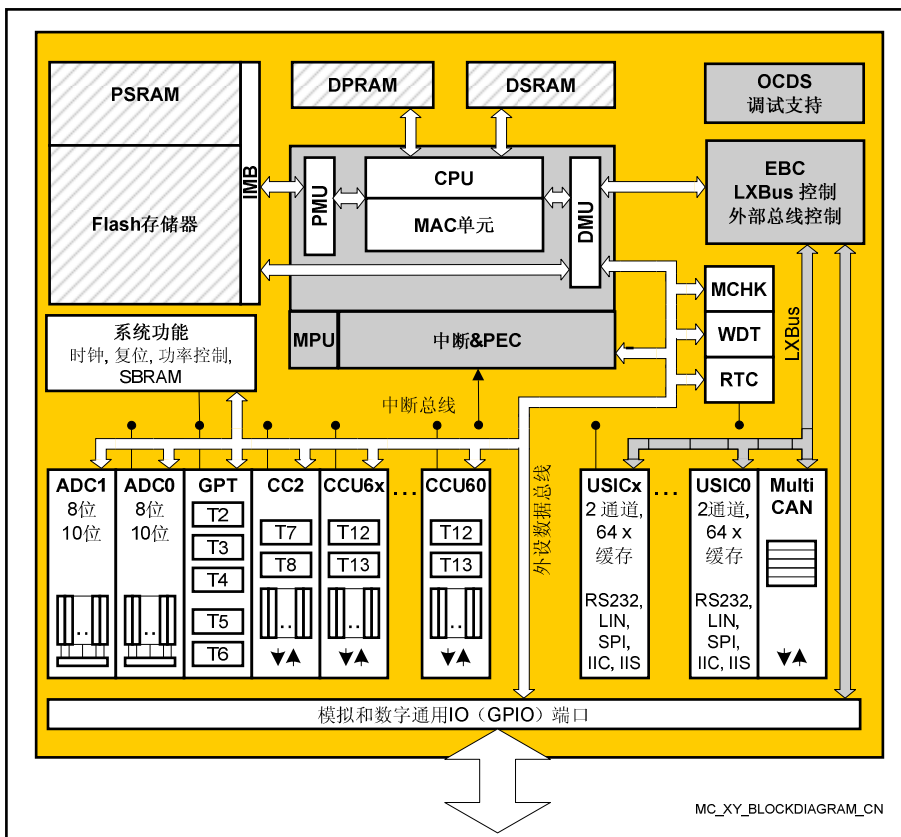


图 2-1 XE16xyM 功能框图

## 2.1 基本 CPU 概念及优化

CPU 内核由一组经过优化的功能单元组成，包括取指/处理流水线、16 位算术逻辑单元 (ALU)、40 位乘累加单元 (MAC)、地址和数据单元 (ADU)、取指单元 (IFU)、寄存器文件 (RF) 以及专用特殊功能寄存器 (SFR)。

XE16xyM 中的大多数指令都能够单周期执行，从而极大的提高了 CPU 性能，同时与 C166 指令保持兼容。此外，XE16xyM 具有出色的 DSP 性能、对各种存储器和外设的并行访问等特性进一步提升了系统的整体性能。

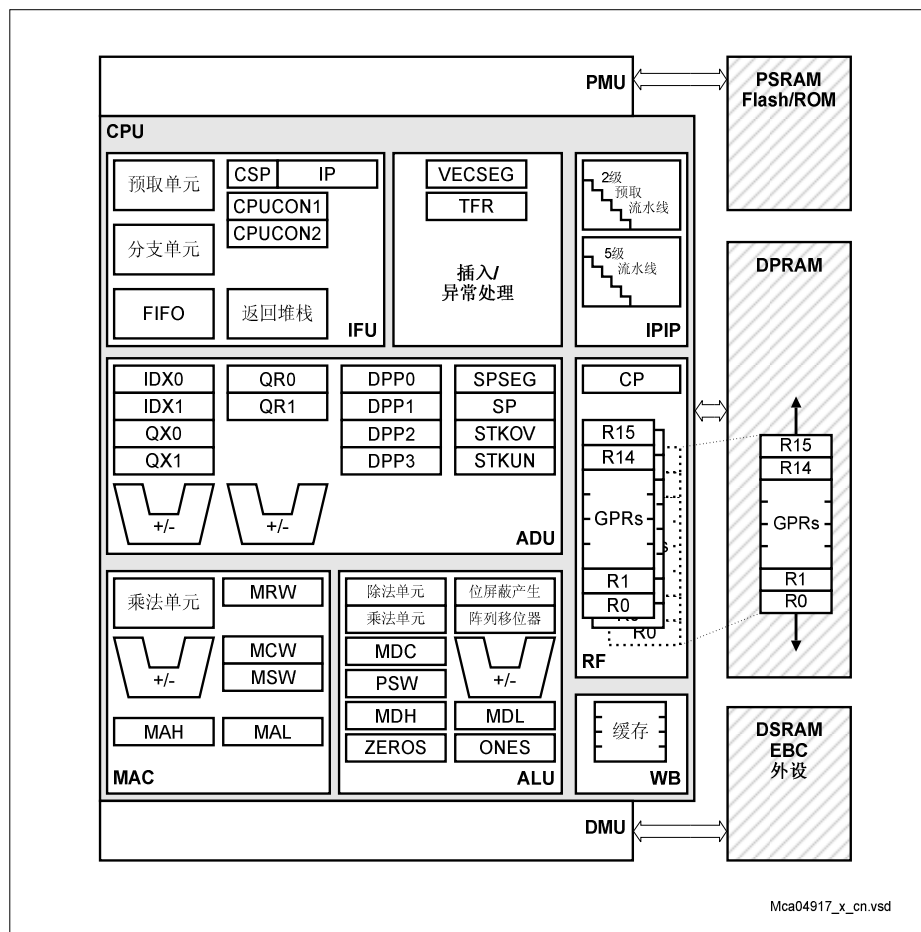


图 2-2 CPU 框图



## CPU 特性概述

- 指令与 C166 系列完全向上兼容
- 2 级取指流水线，具有用于指令预取的 FIFO
- 5 级指令执行流水线
- 专用硬件实现流水线数据前送控制（解决数据相依性问题）
- 多条用于数据和指令传送的高带宽总线
- 代码和数据的线性地址空间（冯诺伊曼架构）
- 几乎所有指令都可在单时钟周期内执行
- 单时钟周期快速乘法运算（16 位×16 位）
- 21 个 CPU 时钟周期的快速后台除法运算（32 位/16 位）
- 内嵌的高级 MAC（乘累加）单元：
  - 零延迟的单周期 MAC 指令；MAC 单元内含 16 位×16 位乘法器
  - 32 位加法和 32 位减法运算
  - 40 位阵列移位器和 40 位累加器用于溢出处理
  - MAC 指令支持 32 位自动饱和或舍入
  - 直接支持小数运算
  - 每阶有限冲激响应滤波器（FIR）需要一个 MAC 周期，无循环缓冲管理
- 增强的布尔位操作能力
- 高性能分支、调用和循环处理
- 零周期跳转执行
- 基于寄存器的设计，具有多个可变的寄存器组（字节或字操作数）
- 两个附加的快速寄存器组
- 带有自动上溢/下溢检测的可变系统堆栈
- “快速中断”和“快速上下文切换”特性
- 内嵌的存储器保护单元（MPU）

CPU 的高性能和灵活性是通过多个经过优化的功能模块来实现的（见图 2-2）。功能模块的优化将在以下章节详细描述。

### 2.1.1 存储器保护单元 (MPU)

存储器保护单元 (MPU) 提供用于实现存储器保护的硬件机制。MPU 可检测用户定义的存储区范围内是否有未经授权的访问 (读、写或取指)。MPU 可为包括外设区域在内的整个地址空间提供保护。

MPU 可将运行在处理器上的不同的应用或软件构件进行“封装”。当今的复杂系统依赖于多方软件，采用这种封装方式可确保软件的完整性、并具有故障隔离能力。

### 2.1.2 高指令带宽/快速执行

基于硬件的支持，大多数 XE16xyM 指令可以在单时钟周期内执行 ( $1/f_{\text{SYS}}$ )，这些指令包括大多数寻址模式下的算术指令、逻辑指令和转移指令。

一些特殊指令如 Jumps，需占用多个机器周期。由于除法指令主要是后台执行，因此可与其它指令并行执行。由于采用预测机制 (见 [章节 5.2](#))，被正确预测的分支指令仅需要一个周期，或者甚至可以和另一条指令并行执行，不单独占用周期 (零周期跳转)。

使用指令流水线大大缩短了指令周期时间。指令流水线使得 CPU 能够并行处理多条顺序指令的不同阶段。最多可 7 级并行工作：

**两级取指流水线**从相应的程序存储器中读取指令并进行预处理：

**指令预取：**按照预测顺序从 PMU 预取指令。在分支检测单元中进行指令预处理以检测分支。预测逻辑决定是否执行分支操作。

**取指：**根据分支预测规则，计算要读取的下一条指令的指针。分支折叠单元对检测到的分支指令进行预处理，并将该指令和上条指令组合 (并行处理)，从而实现零周期的分支指令执行。预取指令保存在指令 FIFO 中；与此同时，先前保存的指令从 FIFO 中取出送至指令处理流水线。

**五级指令处理流水线**执行如下操作：

**解码：**对读取的指令进行解码。如有需要，从寄存器文件中读取用于间接寻址的 GPR。

**寻址：**计算所有操作数的地址。访问系统堆栈时，堆栈指针 (SP) 自动递增或递减。

**存储：**读取所有需要的操作数。

**执行：**对先前读取到的操作数执行指定的操作 (ALU 或 MAC)，更新状态标志，对 CPU-SFR 执行写操作。如有需要，用于间接寻址的 GPR 递增或递减。

**回写：**将结果操作数写回到指定地址。DPRAM 中的操作数通过回写缓存保存。

### 2.1.3 功能强大的执行单元

**16 位算术逻辑单元 (ALU)** 执行所有标准的 (字类型) 算术和逻辑运算。此外, 进行字节操作时, 借助 ALU 运算结果的位 6 和位 7 来正确设置状态标志; 进行多精度运算时, 要将前面计算结果的“进位”信号送给 ALU。

大多数内部执行模块经过优化, 既可执行 8 位运算, 也可执行 16 位运算; XE16xyM 的指令既可处理字节运算, 也为字运算提供了字节的符号扩展; 内部总线结构也支持字节或字传送, 根据外设需要向外设发送、或从外设读取数据。

每次执行算术、逻辑、移位或数据转移操作后, PSW 中的一组标志位会相应自动更新。这些标志可以用作分支转移的特定条件。通过用户自定义的分支测试, 可进行有符号和无符号算术运算。在进入中断或强制中断服务程序时, CPU 自动保存标志位。

16 位阵列移位器在一个周期内可实现多位移位, 它还支持循环移位和算术移位。

**乘累加单元 (MAC)** 支持扩展的算术运算, 如 32 位加、32 位减、单周期 16 位 × 16 位乘法运算。组合的 MAC 运算 (乘累加/减) 体现了 CPU 的主要 DSP 功能。

**地址数据单元 (ADU)** 包括两个独立的算术单元, 用来产生、计算和更新数据访问地址。ADU 执行以下主要任务:

- 标准地址单元支持短寻址、长寻址和间接寻址模式的线性地址运算, 还支持数据分页和堆栈处理。
- DSP 地址产生单元中包括一组附加的地址指针和偏移地址寄存器, 只用于 CoXXX 指令。

CPU 支持多种功能强大的寻址模式, 用于字、字节和位数据访问 (短寻址、长寻址、间接寻址)。不同寻址模式使用不同的格式, 对应不同的应用范围。

专用位处理指令可对外设进行有效的控制及测试, 同时增强了数据处理能力。这些指令可直接访问可位寻址空间中的两个操作数, 而无需先将其移入临时标志位。逻辑指令只用一条指令就能比较和修改外设的某个控制位。多位移位指令 (单周期执行) 可避免多次执行单位移位指令 (长指令流)。位域指令可用一条指令来修改某操作数中的多个位元。

## 2.1.4 高性能的分支、调用和循环处理

流水线结构通过执行连续的指令流使得系统性能最佳。流水线被打断后需要被重填，随后新指令依次通过流水线的各个阶段。由于在控制器应用中，分支操作占很大比例，因此 XE16xyM 对分支指令进行了优化，只在特殊情况下才需要进行流水线重填。通过在指令预取阶段检测和预处理分支指令、并预测相应的分支目标地址，实现了分支指令的优化。

随后从预测的目标地址中继续预取指令。如果预测正确，即使执行分支指令（即，指令执行是非线性的），随后的指令就可以“无间隙”的进入执行流水线。分支目标预测（见[章节 5.2.1](#)）使用如下规则：

- **无条件分支：**在这种情况下分支预测几乎不起作用，始终执行跳转、且目标地址确定。该规则适用于隐含的无条件跳转指令，如 **JMPS**、**CALLR** 或 **RET**，以及含有条件代码“unconditional”的跳转指令（无条件跳转指令），如 **JMPI cc\_UC**。
- **固定预测：**用于实现循环功能的分支指令，如指令执行跳回到前面的地址（循环的起始位置）认为执行了跳转。该规则适用于条件跳转指令，如 **JMPR cc\_XX** 或 **JNB**。
- **可变预测：**在这种情况下，相应预测（执行或不执行跳转）被编码在指令中，每条分支指令的预测可单独选择。因此，软件设计人员可优化指令流，跳转到指定的代码段<sup>1)</sup>。该规则适用于跳转指令 **JMPA** 和 **CALLA**。
- **条件间接分支：**假定始终不执行这些跳转。该规则适用于跳转指令 **JMPI cc\_XX**, **[Rw]** 和 **CALLI cc\_XX**, **[Rw]**。

系统状态信息被自动存入内部系统堆栈，在进入或退出中断或强制中断服务程序时，无需通过指令来保存状态。调用指令将 **IP** 值压入系统堆栈，调用指令的执行时间与跳转指令相同。此外，系统还支持间接分支和调用指令。在宏汇编和高级语言中，该特性支持多分支 **CASE** 语句的实现。

---

1) 编程工具支持每种预测使用专用助记符（由编程人员决定）；或者支持通用助记符，应用自己的预测规则。

### 2.1.5 统一及优化的指令格式

为了使流水线的性能最佳，指令集的设计融入了精简指令集计算（RISC）的概念，能够对指令和操作数快速解码，减少占用流水线的時間。不过，这些概念并不排除微控制器用户需要使用的一些复杂指令。指令集设计满足了以下目标：

- 为频繁执行的操作提供功能强大的指令，这些操作原先需要多条指令才可完成。避免读写暂存寄存器，如累加器和进位位。可并行执行任务，如在进入中断服务程序或子程序时自动保存系统状态。
- 统一每条指令的操作数位置，以避免复杂的编码方案；避免不常用的复杂寻址模式。从而可缩短指令解码时间，并简化编译器和汇编器的开发。
- 最常用的指令采用单字指令格式，其它所有指令均采用双字格式。这样使得所有指令均按字长存放，从而不需要设计复杂的硬件来实现对齐，同时还增加了相关分支指令的跳转范围。

使用功能强大的 XE16xyM 指令集，CPU 硬件的高性能得以充分发挥。指令集包括以下几种指令类型：

- 算术指令
- DSP 指令
- 逻辑指令
- 布尔位操作指令
- 比较和循环控制指令
- 移位和循环移位指令
- 优先化指令
- 数据转移指令
- 系统堆栈指令
- 跳转和调用指令
- 返回指令
- 系统控制指令
- 其它指令

操作数类型包括位、字节、字和双字。特殊指令支持字节到字的转换（扩展）。支持直接寻址、间接寻址和立即寻址等多种操作数寻址方式。

## 2.1.6 可编程多优先级中断系统

XE16xyM 提供了 96 个独立的中断节点，分成 16 级优先级，每级内又分 8 个组优先级。大多数中断源与专用的中断节点相连。有些情况下，多个中断源可共用一个中断节点，以有效使用系统资源。这些节点可由多个中断源请求激活，由中断子节点控制寄存器控制。

XE16xyM 的下列增强特性支持处理多个中断源：

- 外围事件控制器（PEC）：该处理器用来减轻 CPU 的中断请求负荷。中断请求触发 PEC 时，在任意两个地址单元（由 PEC 的源指针和目标指针指定相应地址）之间执行单周期字节或字传送，可选择的地址更新方式是：源指针增加、目标指针增加或者两个地址指针都增加，从而避免了进入和退出中断或强制中断服务程序时的 CPU 开销。只需从当前 CPU 操作中“窃取”一个周期就能实现 PEC 操作。
- 多优先级中断控制器：通过多优先级中断控制器，可为所有中断指定任意的优先级。中断还可分组，从而避免相同优先级的任务互相中断。每个中断节点对应单独的中断控制寄存器，用于存放中断请求标志、中断使能标志和中断优先级位域。某中断被 CPU 响应之后，它只能被更高优先级的服务请求所中断。对于标准中断处理，每个中断节点都对应有专用中断向量地址。
- 多寄存器组：除一个可重新定位的全局寄存器组之外，还附加两个用于快速上下文切换的局部寄存器组。用户可在内部 DPRAM 的任意地址定义多个寄存器组，每个寄存器组最多由 16 个通用寄存器组成。只需一条指令就可实现从一个寄存器组到另一个寄存器组的切换（切换寄存器组会清空流水线，改变全局寄存器组需要一个验证序列）。

XE16xyM 能够快速响应非确定性事件，中断响应时间通常为 7 个时钟周期（在内部程序执行情况下）。快速外部中断输入信号在每个时钟周期都被采样，因此，即使外部信号持续时间极短，仍可被识别。

XE16xyM 还提供了“硬件强制中断”机制，用以识别并处理运行过程中出现的异常或错误情况。硬件强制中断会立即引起非屏蔽系统响应，与标准中断服务相似（跳转到专用的向量表地址）。由强制中断标志寄存器（TFR）中的标志位来指示是否发生硬件强制中断。除非当前正在处理另一个更高优先级的强制中断服务，否则，硬件强制中断将中断正在执行的任何程序。硬件强制中断服务通常不能被标准中断或 PEC 中断所中断。

软件中断通过“TRAP”指令结合一个强制中断编号来实现。

### 2.1.7 系统资源接口

XE16xyM 通过多总线系统将 CPU 与系统资源联接，这些总线同时传送数据，提升了系统整体性能，避免了由于指令或操作数传送而导致 CPU 停滞。

由于全局寄存器组位于双口 RAM (DPRAM) 中，DPRAM 直接与 CPU 相连。对这些单元的读写操作会影响系统性能，因此，对 DPRAM 的访问需要精心优化。

**程序管理单元 (PMU)** 控制对片上程序存储器，如 ROM/Flash 和程序/数据 RAM (PSRAM) 的访问，PMU 还负责从外部存储器读取指令。

PMU 与 CPU 之间的 64 位接口负责 (CPU 请求的) 指令字传送。PMU 决定是从片上存储器还是从外部存储器读取被请求的指令字。

**数据管理单元 (DMU)** 控制对片上数据 RAM (DSRAM)、与外设总线连接的片上外设，以及外部总线上的资源的访问。由外部总线控制器 (EBC) 执行外部访问 (包括访问和片上 LxBus 连接的外设)。

DMU 与 CPU 之间的 16 位接口负责所有的数据传送 (操作数)。根据定义的地址映射，CPU 通过适当的总线进行数据访问。

PMU 和 DMU 紧密配合工作，可进行高速数据互传。互传双向进行：

- **经过 DMU 的 PMU：**从外部地址读取的代码通过 DMU 重新定向到 EBC。从而，XE16xyM 可执行外部存储器中的代码。不能从数据 RAM (DSRAM) 中读取指令。
- **经过 IMB 的 DMU：**通常可对由 PMU 控制的片上资源进行数据访问。包括以下几种传送类型：
  - 从片上程序 ROM/Flash 读取常数
  - 从片上 PSRAM 读取数据
  - 将数据写入片上 PSRAM (在执行程序之前需要先将数据写入)
  - 对片上 Flash 存储器编程/擦除

## 2.2 片上系统资源

XE16xyM 控制器围绕 CPU 提供了多种功能强大的系统资源。CPU 与这些资源的完美结合促成了该微控制器系列产品的高性能。

### 外围事件控制器（PEC）和中断控制

外围事件控制器可通过传送单次数据（字或字节）来响应中断请求，该操作只占用一个指令周期，无需保存和恢复机器状态。每个机器周期，中断控制模块会确定各中断源的优先级次序。若 PEC 服务优先级较高，则启动 PEC 传送。若请求 CPU 中断服务，则查询保存在 PSW 寄存器中当前 CPU 的优先级，以确定是否有更高优先级的中断正在执行。若中断被应答，机器的当前状态被保存在内部系统堆栈，CPU 转入执行相应的中断服务程序。

PEC 内含一组 SFR，用于保存 8 路数据传送通道的计数值和控制位。此外，RAM 中有专门区域存放 PEC 源地址和目的地址。PEC 与其它外设的控制方式相似：通过 SFR 对每个通道进行所需的设置。

每进行一次 PEC 服务，PEC 传送计数器减 1，连续传送模式除外。当计数器计数到零时，执行与相应中断源对应的向量地址处的标准中断。PEC 传送特别适用于一些场合，如将寄存器的内容送入/移出存储器。XE16xyM 有 8 路 PEC 通道，每路通道均可进行快速、由中断驱动的数据传送。

### 存储器区域

XE16xyM 的存储器空间为冯诺伊曼体系架构。即程序存储器、数据存储器、寄存器和 IO 口位于同一个线性地址空间内，容量高达 16 MB。整个存储器空间可按字节或字访问。片上存储器的特殊区域可直接进行位访问。

*注：器件提供的实际存储器容量由其型号决定，上面给出的是该系列产品的最大存储器容量。*

**高达 576 KB 片上 Flash 存储器**存储代码或常数。片上 Flash 存储器由 2 或 3 个 Flash 模块组成，Flash 模块分别由 4 KB 的扇区组成。每个扇区可分别进行写保护<sup>1)</sup>、擦除和编程（以 128 字节为单位）。整个 Flash 区可被读保护。可使用一个由用户设定的密码序列暂时解锁被保护区域。Flash 模块既支持 128 位读访问，又具有安全高效的编程和擦除算法。动态纠错为所有读操作提供了极高的数据安全性。可并行读取不同 Flash 模块。

---

1) 为了节省控制位，多个扇区组合起来统一进行 Flash 保护，而这些扇区在进行编程\擦除操作时各自独立。



*注：从片上程序存储器执行程序是所有可能的方案中速度最快的，可使系统性能最高。片上程序存储器的种类由所选器件型号决定。片上程序存储器还包括 PSRAM。*

**高达32 KB片上程序SRAM（PSRAM）**用于存储用户代码或数据。PSRAM通过 PMU 访问，从而优化了取指操作。部分PSRAM可被写保护，其大小可编程设定。

**高达 16 KB 片上数据 SRAM（DSRAM）**用于存储一般用户数据。DSRAM 通过一个独立接口访问，从而优化了数据访问。

**2 KB 片上双口 RAM（DPRAM）**用于存储用户定义的变量、系统堆栈，以及（特别是）通用寄存器组。一个寄存器组可由多达 16 个字宽（R0 到 R15）和/或字节宽（RL0、RH0、...RL7、RH7）的通用寄存器（GPR）组成。

DPRAM 的高 256 字节可直接位寻址。用作 GPR 时，DPRAM 的任何地址都可位寻址。

**8 KB 片上备用 SRAM（SBRAM）**用于存储与系统相关的、当器件的主要部分处于掉电状态时必须被保留的用户数据。通过特殊接口访问 SBRAM，通过电源域 M 对其供电。

CPU 可随意支配的真正寄存器组由多达 16 个字宽和/或字节宽的全局 GPR 组成，物理上这些 GPR 位于片上 RAM 区。上下文指针（CP）寄存器指定 CPU 每次访问有效全局寄存器组的基地址。寄存器组的数目仅受可用内部 RAM 空间的限制。为了便于进行参数传递，两个寄存器组可相互重叠。

多达 32 K 字的系统堆栈用于存储临时数据。系统堆栈可位于整个地址空间的任意位置，CPU 根据堆栈指针（SP）寄存器和堆栈指针段（SPSEG）寄存器访问堆栈。每次访问堆栈时，两个独立的 SFR：STKOV 和 STKUN 将自动和每次堆栈访问的堆栈指针值进行比较，以检测堆栈是否上溢和下溢。该机制还可控制更大的虚拟堆栈。系统堆栈位于内部数据 RAM 区（DPRAM，DSRAM）时，堆栈操作的性能最佳。

对所选存储空间的硬件检测由内部存储器解码器完成，用户只需直接或间接指定任何地址，即能获得想要的<sup>1</sup>数据，无需使用暂存寄存器或特殊指令。

**特殊功能寄存器**占用三个地址区域：标准特殊功能寄存器区（SFR）占用 512 字节；扩展特殊功能寄存器区（ESFR）占用 512 字节；内部 IO 区（XSFR）占用 4 KB。SFR 是字宽寄存器，用来控制和监控不同片上单元的功能。未使用的 SFR 地址被保留，用于对 XE166 的后续系列产品进行功能扩展。保留地址区不能被访问、或者将其置零，以保证向上兼容。

为了满足要求更大存储空间的设计的需要，可将高达12 MB（见表2-1）的外部 RAM和/或ROM和微控制器相连。外部总线接口还提供对片外外设的访问。

**表 2-1 XE16xyM 存储器映射**

地址区间	起始地址	结束地址	区域大小 <sup>1)</sup>	备注
IMB 寄存器空间	FF'FF00 <sub>H</sub>	FF'FFFF <sub>H</sub>	256 B	-
保留（访问 trap）	F0'0000 <sub>H</sub>	FF'FEFF <sub>H</sub>	< 1 MB	减去 IMB 寄存器
保留用作 EPSRAM	E8'8000 <sub>H</sub>	EF'FFFF <sub>H</sub>	480 KB	镜像 EPSRAM
仿真 PSRAM	E8'0000 <sub>H</sub>	E8'7FFF <sub>H</sub>	32 KB	Flash 时序
保留用作 PSRAM	E0'8000 <sub>H</sub>	E7'FFFF <sub>H</sub>	480 KB	镜像 PSRAM
程序 SRAM	E0'0000 <sub>H</sub>	E0'7FFF <sub>H</sub>	32 KB	最大速度
保留用作程序存储器	CC'0000 <sub>H</sub>	DF'FFFF <sub>H</sub>	< 1.25 MB	-
程序 Flash 2	C8'0000 <sub>H</sub>	CB'FFFF <sub>H</sub>	256 KB	-
程序 Flash 1	C4'0000 <sub>H</sub>	C7'FFFF <sub>H</sub>	256 KB	-
程序 Flash 0	C0'0000 <sub>H</sub>	C3'FFFF <sub>H</sub>	256 KB	<sup>2)</sup>
外部存储器区域	40'0000 <sub>H</sub>	BF'FFFF <sub>H</sub>	8 MB	-
可用外部 IO 区 <sup>3)</sup>	20'C000 <sub>H</sub>	3F'FFFF <sub>H</sub>	< 2 MB	减去 USIC/CAN
MultiCAN/USIC 寄存器	20'8000 <sub>H</sub>	20'BFFF <sub>H</sub>	16 KB	可选的其它地址
可用外部 IO 区 <sup>3)</sup>	20'6000 <sub>H</sub>	20'7FFF <sub>H</sub>	8 KB	-
USIC 寄存器	20'4000 <sub>H</sub>	20'5FFF <sub>H</sub>	8 KB	通过 EBC 访问
MultiCAN 寄存器	20'0000 <sub>H</sub>	20'3FFF <sub>H</sub>	16 KB	通过 EBC 访问
外部存储器区	01'0000 <sub>H</sub>	1F'FFFF <sub>H</sub>	< 2 MB	减去段 0
SFR 区	00'FE00 <sub>H</sub>	00'FFFF <sub>H</sub>	0.5 KB	-
双口 RAM	00'F600 <sub>H</sub>	00'FDFF <sub>H</sub>	2 KB	-
保留用作 DPRAM	00'F200 <sub>H</sub>	00'F5FF <sub>H</sub>	1 KB	-
ESFR 区	00'F000 <sub>H</sub>	00'F1FF <sub>H</sub>	0.5 KB	-
XSFR 区	00'E000 <sub>H</sub>	00'EFFF <sub>H</sub>	4 KB	-
数据 SRAM	00'A000 <sub>H</sub>	00'DFFF <sub>H</sub>	16 KB	-

地址区间	起始地址	结束地址	区域大小 <sup>1)</sup>	备注
保留用作 DSRAM	00'8000 <sub>H</sub>	00'9FFF <sub>H</sub>	8 KB	-
外部存储器区	00'0000 <sub>H</sub>	00'7FFF <sub>H</sub>	32 KB	-

1) 标有“<”的区域大小略小于标注值，见“备注”列。

2) 第一个 Flash 区段的最高 4 KB 扇区保留作内部使用（C0'F000<sub>H</sub>至 C0'FFFF<sub>H</sub>）。

3) 一些流水线优化对外部 IO 区无效，这对合理控制片外外设很有必要。

*注：不同型号产品的可用存储器请参考相关的数据手册。*

## 外部总线接口

为了满足需要更大存储容量的设计需要，可通过外部总线接口将高达 12 MB 的外部 RAM/ROM/Flash 或外设和 XE16xyM 微控制器连接。

所有外部存储器访问通过专用的片上外部总线控制器（EBC）来实现。不需使用外部存储器时，设定为单片模式；需要使用外部存储器时<sup>1)</sup>，需要进行以下选择：

- 地址总线宽度范围为 0...24 位
- 数据总线宽度 8 位或 16 位
- 总线操作复用或非复用

非复用总线模式下，地址通过 P0 口和 P1 口输出，数据分别通过 P10 或 P2 输入/输出。复用总线模式下，地址和数据都通过 P10 口和 P2 口输入/输出。高位（段）地址线使用 P2 口。有效地址线的个数可选，从而限制外部地址空间的范围。当接口线被分配给 P2 口时，需要进行段地址个数选择。

最多可为五个地址区域分别选择总线模式（复用/非复用），数据总线宽度（8 位/16 位），甚至总线周期长度（等待状态，信号延时），从而能够以最高效率对各种存储器和外设元件进行直接访问。

访问速度非常低的存储器或模块时，XE16xyM 通过一个特殊的“READY”功能支持可变的访问时间。控制输入信号的有效电平可选。

总线仲裁使用  $\overline{\text{HOLD}}/\overline{\text{HLDA}}$  协议，允许其它总线主设备共享外部资源。

外部总线时序和参考时钟输出 CLKOUT 的上升沿有关。外部总线协议与标准 C166 系列兼容。

对于所需地址空间小于 64 KB 的应用而言，可选择不分段存储器模型，所有单元均可 16 位寻址。因此 P2 口高位部分不需用作高地址位（Axx...A16）的输出，与分段存储器模型的情况相同。

EBC 还控制与片上 LxBus 相连的资源的访问。LxBus 是外部总线的内部标识，LxBus 以与外部元件相同的方式对片上外设和模块进行访问。

MultiCAN 模块和 USIC 模块通过 LxBus 连接和访问。

*注：外部总线接口在 64 引脚的 XE162xM 器件中不可用。*

---

1) 如果几个地址窗采用不同设置，总线模式会动态切换。

## 2.3 片上外设模块

XE166 系列将外设和内核明确分离，该结构支持最大数量的并行操作；可以方便的添加或删除外设，而无需改动内核。每个功能模块独立处理各自的数据，并通过公共总线交换信息。通过设置相应的特殊功能寄存器（SFR）来控制外设。这些 SFR 位于标准 SFR 区（00'FE00<sub>H</sub> ... 00'FFFF<sub>H</sub>）、扩展 ESFR 区（00'F000<sub>H</sub> ... 00'F1FF<sub>H</sub>），或内部 IO 区（00'E000<sub>H</sub> ... 00'EFF<sub>H</sub>）。

通过这些内嵌外设可使 CPU 与外部世界连接，或者将原本需要在片外添加的功能集成到片内。

XE16xyM 的基本外设包括：

- **存储器检查器单元**
- **通用定时器单元（GPT1、GPT2）**
- **看门狗定时器**
- **捕获/比较单元（CAPCOM2）**
- 多达 **4 个捕获/比较单元 CCU6**（CCU60、CCU61、CCU62、CCU63）
- 两个 10 位**模拟/数字转换器（ADC0、ADC1）**
- **实时时钟（RTC）**
- 13/9 个**并行端口**，总共 119/76 条 I/O 线

因为 LxBus 是外部总线的内部标识，不支持位寻址。通过 EBC 访问与 LxBus 连接的片上资源，如同进行外部访问一样。LxBus 将以下片上外设连接到 CPU：

- 可选的 **MultiCAN 模块**，具有 6 个 CAN 节点和网关功能
- 多达 4 个**通用串行接口通道模块（USIC）**

每个外设包含一组特殊功能寄存器（SFR），用于控制外设的功能、暂时保存中间数据结果；还包含一组相关的状态标志位。每个外设可独立选择时钟源，这些时钟信号由主时钟经 2 的整数倍分频产生。

*注：不同型号产品的可用外设归纳见 **P1-5 “基本特性总结”**。*

## 外设接口

片上外设通常有两种类型的接口：一个与 CPU 的总线接口，一个与其它外设或外部硬件的接口信号。CPU 与片上外设的通信通过特殊功能寄存器（SFR）和中断实现。SFR 用作外设的控制/状态和数据寄存器。中断请求由外设根据操作期间的特定事件产生，如操作完成、操作错误等。

通过并口上的特定引脚（这些引脚已被选做外设的输入或输出功能）实现片上外设与外部硬件的通信。此时，端口引脚由片上外设控制（用作输出），或由外部硬件控制（用作输入）。与引脚的通用 I/O 功能相不同，这称为引脚的“复用（输入或输出）功能”。

## 外设时序

CPU 和外设的内部操作基于系统时钟（ $f_{\text{sys}}$ ）控制。时钟产生单元使用外部（如晶振）或内部时钟源生成系统时钟信号。通过将外设与时钟信号暂时（以降低功耗）或永久（如果特定应用不需要该外设模块）断开的方式禁用某外设。每个时钟周期 CPU 均可访问外设 SFR。如果软件写访问 SFR 和外设修改 SFR 同时发生，软件写操作占优。有关外设时序的信息将在各个外设章节中予以详细说明。

## 编程提示

- **访问 SFR:** SFR 位于存储器空间的各数据页。以下寻址方式支持访问 SFR:
  - 间接或直接 **16 位（mem）寻址**必须确保所用的数据页指针（DPP0 ... DPP3）选择相应的数据页。
  - 通过外围事件控制器（PEC）访问 SFR 时，用 SRCPx 和 DSTPx 指针寻址，不使用数据页指针。
  - 对标准 SFR 区的**短 8 位（reg）寻址**，不使用数据页指针，该 512 字节区域内的寄存器被直接访问。
  - 对扩展 **ESFR 区**的**短 8 位（reg）寻址**，需要切换到 512 字节的扩展 SFR 区，该操作通过扩展指令 EXTR、EXTP（R）、EXTS（R）实现。
- 通过间接或直接 **16 位（mem）寻址**对字宽 SFR 进行**字节写操作**、或通过 PEC 进行字节传送时，未寻址字节应强制置零。通过**短 8 位（reg）寻址**对 SFR 进行字节写操作时，只能访问 SFR 的低位字节，高位字节强制置零。因此推荐使用位域指令（BFLDL 和 BFLDH）对 SFR 的任意字节的任意位进行写操作，该指令不会影响到未寻址字节和未选定位。
- **写访问只写位/寄存器**通常会修改其它寄存器中的位。在一些情况下，修改操作由状态机控制。因此，写访问之后，立即读取被该写访问修改的寄存器时，写操作的效果可能不可见。
- **保留位:** XE16xyM 的 SFR 中有些位标记为“保留”。禁止用户软件对保留位写 1。这些位目前没有使用，可能会用作扩展后续产品的新功能，启用保留位

时，“1”代表新功能的有效状态，“0”代表新功能的无效状态。因此将保留位置 0 保证了当前软件的可移植性。进行读操作时，保留位应被忽略或屏蔽。

### 捕获/比较单元 (CAPCOM2)

CAPCOM 单元支持多达 16 路通道上时序的产生和控制，最大精度为 1 个系统时钟周期（交错模式下最大精度为 8 个系统周期）。CAPCOM 单元通常用于处理高速 I/O 任务，如脉冲和波形的产生、脉宽调制 (PWM)、数模 (D/A) 转换、软件定时、或记录外部事件的时间信息。

CAPCOM 单元中有两个 16 位定时器 (T7/T8)，它们各有相应的重载寄存器，分别为每个捕获/比较寄存器提供两个独立的时间基准。

定时器的输入时钟是经过预分频处理（分频因子可编程设定）的内部系统时钟；或来自模块 GPT2 中定时器 T6 的上溢/下溢信号，这样可提供多种定时器周期和精度，以满足不同应用精度需求。此外，CAPCOM 定时器 T7 的外部计数输入信号触发将外部事件的时间信息记录在捕获/比较寄存器中。

捕获/比较寄存器组由 16 个双功能捕获/比较寄存器组成，每个寄存器可分别指定和定时器 T7 或 T8 配合工作，并分别设定用作捕获或比较功能。

每个 CAPCOM 模块中的所有寄存器都各自对应一个与之相关的端口引脚，作为触发捕获功能的输入引脚，或指示发生比较事件时用作输出。

**表 2-2 比较模式 (CAPCOM2)**

比较模式	功能
模式 0	仅产生中断的比较模式； 每个定时器周期可产生多个比较中断
模式 1	每次比较匹配时引脚翻转； 每个定时器周期可产生多个比较事件
模式 2	仅产生中断的比较模式； 每个定时器周期只产生一个比较中断
模式 3	匹配时引脚置 1；比较定时器溢出时引脚复位到 0； 每个定时器周期只能产生一个比较事件
双寄存器模式	两个寄存器控制同一引脚； 每次比较匹配时输出引脚翻转； 每个定时器周期可产生多个比较事件
单次事件模式	产生单个信号沿或脉冲； 可在任何比较模式下使用



若某个捕获/比较寄存器被选择用作捕获模式，一旦与该寄存器相关的输入引脚上有外部事件发生，定时器的当前值将被锁存（捕获）到该捕获/比较寄存器中。此外，将产生该捕获/比较寄存器的中断请求。可选择外部信号的正跳变、负跳变或任意跳变作为触发事件。

若捕获/比较寄存器被选择用作比较模式（共五种比较模式），保存在该寄存器中的数值将和对应该定时器的计数值进行连续比较。

当定时器的计数值和捕获/比较寄存器的值匹配时，根据选择的比较模式产生特定的动作。

### **捕获/比较单元 CCU6**

CCU6 单元支持最多三路 16 位捕获/比较通道、一路独立的 16 位比较通道上时序的产生和控制。

比较模式下，CCU6 单元的每路通道产生两个输出信号，它们极性相反、脉冲翻转区间互不重叠（死区时间控制）。比较通道产生的单路 PWM 输出信号可进一步用于调制捕获/比较输出信号。

捕获模式下，一旦引脚 CCx 上信号发生跳变，比较定时器 T12 的内容被保存到捕获寄存器中。

CCU6 可产生边沿对齐或中心对齐的 PWM 输出信号。这些 PWM 输出信号可连续产生，也可只产生一次（单次模式下）。

比较定时器 T12 和 T13 为自由运行定时器，其时钟来自经预分频处理的系统时钟。

在电机控制应用中（无刷直流驱动），可由比较定时器 T12、或中断输入上的霍尔传感序列（块交换）控制产生各种不同的多通道 PWM 信号。霍尔传感器工作模式为霍尔输入信号提供了噪声滤波，并支持自动转速测量。

CCU6 具有强制中断功能，可使系统快速急停，而无需 CPU 干预。外部信号（CTR<sub>AP</sub>）会触发输出引脚切换到可选的逻辑电平以保护相连的功率器件。

*注：可用的 CCU6 单元数和通道数由所选择的器件型号决定。*

### **通用定时器单元（GPT1，GPT2）**

GPT12E 单元具有非常灵活的多功能定时器/计数器结构，可用作事件定时和计数、脉宽和占空比测量、脉冲产生、脉冲倍频等多种用途。

GPT12E 单元有五个 16 位定时器，分配给两个独立的模块 GPT1 和 GPT2。每个模块中的各个定时器均可独立工作在不同的工作模式，或者和同模块中的其它定时器级联工作。

**模块 GPT1** 中的三个定时器 T2、T3、T4 可被分别设置为四种基本工作模式之一：定时器模式、门控定时器模式、计数器模式、和增量接口模式。定时器模式下，定时器

的输入时钟来自经过预分频处理的系统时钟（预分频因子可编程设定）；计数器模式下，可用外部事件作为定时器的时钟源。

门控定时器模式支持脉宽或占空比测量，此时定时器操作由外部输入引脚上的“门控”电平控制。此时，每个定时器对应一个相关的端口引脚（TxIN），用作门控或时钟输入。GPT1 模块定时器的最大精度为 4 个系统时钟周期。

每个定时器的计数方向（递增/递减）由软件设定，或者由端口引脚（TxEUD）上的外部信号动态选择，如进行位置跟踪。

增量接口模式下，GPT1 定时器（T2、T3、T4）可以通过各自的输入口 TxIN 和 TxEUD 直接和增量位置传感器信号 A 和 B 相连。方向和计数信号可以从这两个输入信号得到，因此相关定时器 Tx 的内容与传感器位置相对应。第三个位置传感器信号 TOP0 可以和中断输入相连。

定时器 T3 有一个输出翻转锁存器（T3OTL），定时器每次上溢/下溢时 T3OTL 的状态改变。该锁存器的状态从引脚 T3OUT 输出，如可用来监控外部硬件电路的超时现象。T3 的溢出翻转锁存信号可以作为 T2、T4 的计数时钟，从而能够用高精度测量长时间信号的周期。

除基本工作模式以外，定时器 T2 和 T4 还可被设置为 T3 的重载或捕获寄存器。用作捕获或重载寄存器时，定时器 T2 和 T4 停止运行。输入引脚（TxIN）发生跳变时，定时器 T3 的内容被捕获到 T2 或 T4 中。外部信号跳变、或者翻转锁存器 T3OTL 发生选定的状态跳变时，可触发 T2 或 T4 的内容重新装入定时器 T3。如果用 PWM 信号的接通、关闭电平时间分别设置 T2 和 T4，并用 T3OTL 相反的跳变沿触发 T2 和 T4 轮流重载 T3，即可连续产生该 PWM 信号，而无需软件干预。

**GPT2 模块**的最大精度为 2 倍系统时钟周期，可提供准确的事件控制和时间测量。GPT2 包括两个定时器（T5、T6）和一个捕获/重载寄存器（CAPREL）。两个定时器的输入时钟源来自经过预分频处理的 CPU 时钟（预分频因子可编程设定）或外部信号。每个定时器的计数方向（递增/递减）可以由软件设定，或者由引脚（TxEUD）上的外部信号动态改变。定时器的级联通过定时器 T6 的输出翻转锁存器（T6OLT）实现，定时器每次上溢/下溢时 T6OLT 的状态改变。

锁存器的状态可以用作定时器 T5 的输入时钟，可从引脚 T6OUT 输出。定时器 T6 的上溢/下溢信号还可用作 CAPCOM2 定时器的输入时钟，还可触发重载，将 CAPREL 寄存器的值重新装入定时器 T6 中。

一旦端口引脚（CAPIN）上的外部信号发生跳变，CAPREL 寄存器可捕获定时器 T5 的计数值。捕获操作完成后，可选择将定时器 T5 清零。这使得 XE16xyM 能够测量绝对时间差、或者实现脉冲乘法，而无需软件开销。

GPT1 定时器 T3 的输入引脚 T3IN 和/或 T3EUD 发生跳变时，还可触发将定时器 T5 的值捕获到 CAPREL 中。当 T3 工作在增量接口模式，该特性尤其有用。

## 实时时钟（RTC）

XE16xyM 的实时时钟（RTC）模块直接由独立时钟源驱动。由寄存器 RTCCLKCON 从多个内部和外部时钟源中进行选择。因此，RTC 的时钟和所选择的 XE16xyM 的时钟产生模式无关。

RTC 主要由一组分频器模块构成：

- 可选的 32:1 和 8:1 分频器（开启或关闭）
- 可重载 16 位定时器 T14
- 32 位 RTC 定时器模块（可通过寄存器 RTCH 和 RTCL 访问），由以下定时器组成：
  - 可重载 10 位定时器
  - 可重载 6 位定时器
  - 可重载 6 位定时器
  - 可重载 10 位定时器

所有定时器递增计数。每个定时器可单独产生中断请求，所有的中断请求合并成一个公共中断节点请求。

*注：为了保持正确的系统时间，与 RTC 相关的寄存器不受应用复位影响，即使执行中间的复位操作。*

RTC 模块的可被用于：

- 系统时钟，决定当前时间和日期
- 周期性中断，提供与 CPU 频率和其它资源无关的系统时间标记
- 48 位定时器，用于长时间间隔的测量

## 模拟/数字转换器 (ADC0、ADC1)

为了进行模拟信号测量，XE16xyM 片上集成两个精度为 10 位、带有 16（或 8）路复用输入通道和采样保持电路的模数转换器（ADC0、ADC1）。两个 ADC 采用逐次逼近技术，可转换多达 4 路输入。采样时间（进行电容器充电）和转换时间可编程设定，从而与外部电路相匹配。ADC 还可以工作在 8 位转换模式，此时转换时间进一步缩短。

几个独立的转换结果寄存器，可选的中断请求和高度灵活的转换顺序为用户提供最大程度的编程能力，可满足各种应用场合的需要。两个 ADC 模块可同步工作，并行采样两路输入通道。

对于需要更多模拟输入通道的应用，XE16xyM 的 ADC 模块提供可自动控制的外部模拟复用器。对于需要较少模拟输入通道的应用，剩余通道可用作数字输入/出口脚。

XE16xyM 的 ADC 支持两种类型的请求源，可由多个内部和外部事件触发这些请求源。

- 扫描请求同时被激活，之后按照预先设定的顺序执行转换操作
- 按照用户设定的顺序执行队列转换请求

此外，在不干扰转换序列的情况下，将一个特定通道的转换插入正在执行的转换序列中。根据转换请求的优先级对所有请求进行仲裁。

数据压缩特性，如极限检查或结果积累，减少了所需 CPU 访问次数。即使 CPU 以极低的速度运行，仍能以较高的转换速度为模拟输入提供精确的转换结果。

外围事件控制器（PEC）可用来控制 ADC 或将转换结果自动保存在存储器内的一个表格中（稍后进行转换结果评估），无需每次传送数据时进入和退出中断服务程序所带来的软件开销。因此，每个 ADC 包含 8 个可级联起来构成结果 FIFO 的结果寄存器。每个结果寄存器可使能为待读模式，从而防止转换结果丢失。

为了防止数字噪声干扰模拟输入信号，并避免输入触发噪声，可在软件控制下，将模拟输入引脚与数字输入级断开，通过寄存器 P5\_DIDIS 和 P15\_DIDIS（Px 口数字输入禁止）分别设置每个引脚。

不进行模数转换操作时，ADC 的自动掉电特性能够最大程度降低系统功耗。

*注：可用的模拟通道数由所选择的器件型号决定。*

### 通用串行接口通道模块 (USIC)

每路 USIC 通道可被单独配置以满足应用的需要，如可在运行过程中选择或改变协议而无需复位。USIC 模块支持下述协议：

- **UART** (ASC, 异步串行接口)
  - 模块功能：接收单元和发送单元，最大波特率为  $f_{\text{SYS}}/4$
  - 每个数据帧包含的数据位个数：1 到 63
  - MSB 或 LSB 在先
- 硬件 **LIN** 支持 (本地互连网络)
  - 根据 ASC 协议进行数据传送
  - 波特率产生器能够根据捕获事件进行波特率检测
  - 高度灵活的、由软件控制的校验和产生
- **SSC/SPI** (带或不带从机选择线的同步串行通道)
  - 模块功能：最大波特率为  $f_{\text{SYS}}/2$ ，受限于环路延迟
  - 每帧数据包含 1-63 位数据，对于超过 63 位的数据帧，需明确定义数据帧的结束
  - MSB 或 LSB 在先
- **IIC** (内部 IC 总线)
  - 应用波特率 100 kbit/s 到 400 kbit/s
  - 支持 7 位和 10 位寻址
  - 完整的主控和从控器件功能
- **IIS** (信息娱乐音频总线)
  - 模块功能：最大波特率  $f_{\text{SYS}}/2$

除了可灵活选择通信协议，USIC 结构的设计还能有效降低系统负荷 (CPU 负荷)，从而保证高效的数据处理。

USIC 的设计基于以下方面的考虑：

- **数据缓存功能**

标准数据缓存包含一个接收数据的双字缓存和一个发送数据的单字缓存，从而能够延长 CPU 的响应时间 (如中断响应延迟)。

- **附加的 FIFO 缓存功能**

除了标准缓存功能，接收数据和发送数据还可保存在一个 FIFO 缓存结构中。接收和发送 FIFO 缓存的大小可单独编程设置。一个 USIC 模块中共有 64 个数据字的缓存 (USIC 的两路通道共用这 64 个数据字缓存)，可根据实际应用的需要将其分配给接收和发送 FIFO。

除 FIFO 缓存功能之外，旁路机制可在不清空 FIFO 缓存的情况下引入高优先级的数据。

- **发送控制信息**

在将要发送的每个数据字上附加了一个 5 位的发送控制信息，用于自动控制诸如字长、帧长、从控选择（针对 SPI 协议）等发送参数。通过分析将要发送的数据字的存放地址，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）。

可利用该特性对每个数据字进行单独处理。例如，和发送 FIFO 中的数据字相关的发送控制信息可自动修改从控选择输出，从而灵活选择不同的通信目标（从控器件）、无需 CPU 干预。此外，还可利用它来控制帧长。

- **灵活的帧长控制**

数据帧的长度和数据字长无关，可通过两种不同的方式控制帧长。第一种选择是自动生成长度已知（最多 63 位）的数据帧；第二种选择是产生更长（甚至不限长度）、或长度可动态控制的数据帧。

- **中断功能**

根据实际应用的需要，每路 USIC 通道的事件可单独分配给 4 路服务请求输出之一。除特定协议事件之外，还可通过中断指示一帧的开始和结束。

- **灵活的接口连接**

每路 USIC 通道为通信信号提供了多种输入和输出引脚的连接选择，从而无需复位器件即可将改变 USIC 信号的引脚分配。

- **输入信号调整**

每个输入信号都要经过一个可编程的输入调整电路处理，该电路具有可编程滤波和同步功能。

- **波特率产生**

每个 USIC 通道都单独配备一个波特率发生器。可根据内部模块时钟或外部频率输入产生波特率。该结构可产生内部无法产生的频率用于传送数据（如用于同步多个通信方）。

- **传送触发功能**

主控模式下，可由 USIC 模块外部产生的事件（例如输入引脚或定时器单元（发送数据有效性检验））触发数据传送。该特性支持与时间基准相关的数据发送。

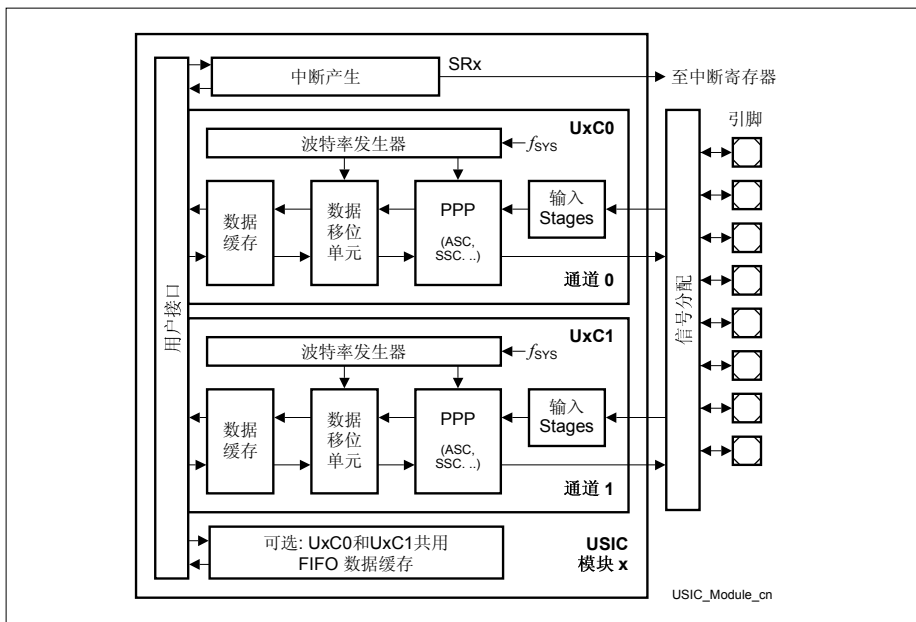
- **支持调试**

USIC 提供特定的地址以读取接收数据，不使用 FIFO 缓存机制。该特性可确保调试器进行读访问时不破坏接收数据序列。

为了达到期望波特率，必须考虑两个标准：模块性能和应用环境。模块性能由模块的输入频率决定，这是模块工作的基础。尽管模块本身可达到的波特率较高（取决于模

块时钟和表示一个数据位所需要的模块时钟周期数），但实际可达到的波特率往往受到应用环境的限制。大多数情况下，由于驱动延迟、信号传输时间或 EMI 等因素的影响，应用环境会限制可达到的最大波特率。

*注：所选择的附加功能（如数字滤波、输入同步、采样点调整、数据结构等）会限制可达到的最大波特率。此外，还需注意附加延迟，比如（内部或外部的）传输延迟和驱动延迟（例如用于 ASC 模式下的冲突检测等）。*



**图 2-3 USIC 通道结构**

USIC 模块包含两个独立的通信通道，结构如图 2-3 所示。

每路通道的数据移位单元和数据缓存支持全双工数据传送。和特定协议相关的操作由协议预处理器（PPP）处理。为了简化数据处理，每个 USIC 模块可通过一个 FIFO 数据缓存来保存每路通道的发送和接收数据。并非所有器件都提供 FIFO（详见 USIC 实现一节的描述）。

由于每路通信通道具有独立的通道控制和波特率产生，因此各通道的通信协议、波特率以及数据格式可单独编程。

## MultiCAN 模块

MultiCAN 模块包含六个全功能 CAN 节点，这些节点可独立工作或者通过网关功能交换数据和远程帧。根据 CAN V2.0B (active) 规范发送和接收 CAN 帧。每个 CAN 节点都可以处理 11 位标识符的标准帧和 29 位标识符的扩展帧。

注：XE16xyM 器件中可用的 CAN 节点个数由所选择的器件型号决定。

所有 CAN 节点共用数目多达 256 个的一套报文对象，每个报文对象可被独立分配给任一 CAN 节点。除了存储接收帧和发送帧外，报文对象可组合起来，在 CAN 节点之间构建网关或者建立 FIFO 缓存器。

可将报文对象组织为双链列表结构，每个 CAN 节点都有各自的报文对象列表。CAN 节点仅将帧存储到分配给该 CAN 节点列表的报文对象中，且仅发送属于该报文对象列表中的报文。功能强大、由命令驱动列表控制器执行所有报文对象列表操作。

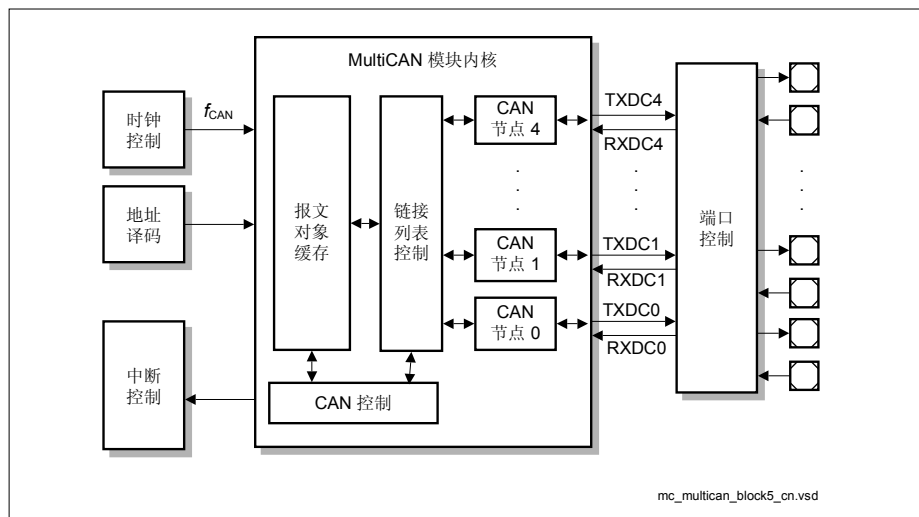


图 2-4 MultiCAN 模块框图

### MultiCAN 特性:

- 根据 CAN V2.0 B active 技术规范确定 CAN 功能（与 ISO 11898 标准兼容）
- 多达 6 个独立的 CAN 节点
- 多达 256 个独立报文对象（所有 CAN 节点共享这些报文对象）
- 每个 CAN 节点都有专用控制寄存器
- 数据传送速率高达 1 Mbit/s，每个节点的数据传送速率可单独设定



- 具有灵活、功能强大的报文传送控制和错误处理能力
- 全 CAN 功能：报文对象可被单独
  - 分配给任一 CAN 节点
  - 配置为发送或者接收对象，或作为报文缓存 FIFO
  - 处理 11 位标识符的标准帧或者 29 位标识符的扩展帧，带有可编程验收屏蔽功能，对接收帧进行验收滤波
  - 远程监控模式，且可通过帧计数器进行监控
- 支持自动网关模式
- 16 个可单独编程的中断节点
- 用于 CAN 总线监控的分析器模式

### 看门狗定时器

看门狗定时器提供了一种故障保险机制，避免系统长时间处于故障状态。

芯片复位后，看门狗定时器始终被使能。无论何时都可通过执行 DISWDT 和 ENWDT 指令禁止和使能看门狗定时器。因此，芯片启动进程始终受看门狗定时器监控。在看门狗定时器溢出之前必须由软件刷新。如果发生硬件或软件错误，软件不能及时刷新，看门狗定时器将溢出，并产生复位请求。

看门狗定时器是一个 16 位定时器，系统时钟经 16,384 或 256 分频得到看门狗定时器的输入时钟。看门狗定时器寄存器的高位字节可设置为预定义的重载值（保存在 WDTREL 中），用于调整监控的时间间隔。应用程序每次服务看门狗定时器之后，看门狗定时器的高位字节被重载，低位字节被清零。

因此，看门狗定时器的监控时间间隔为 3.2  $\mu$ s 到 13.4 s（@ 80 MHz）。

看门狗定时器的复位缺省时间间隔为 6.5 ms（@ 10 MHz）。

### 存储器检查器单元

XE16xyM 存储器检查器模块（MCHK）用于检查存储器、寄存器（如配置寄存器）或通信通道的数据一致性。该模块计算数据块的校验和，通常称为循环冗余码（CRC）。该模块在一个多输入线性反馈移位寄存器（MISR）的基础上生成并行签名。由于基于线性反馈移位寄存器（LFSR），该模块还能产生伪随机数和循环码。

对于编程人员来说，MCHK 是一组与该外设模块相关的寄存器。为了指示相应的错误或操作事件，可使用端口引脚作为信号“MATCH”产生外部事件，可使用中断线作为信号“MISMATCH”产生内部事件。

### 并行端口

XE16xyM 衍生系列有三种封装形式：

- 封装为 LQFP-144 时，提供 119 个 I/O 端口线，组成 11 个输入/输出端口和 2 个输入端口
- 封装为 LQFP-100 时，提供 76 个 I/O 端口线，组成 7 个输入/输出端口和 2 个输入端口
- 封装为 LQFP-64 时，提供 40 个 I/O 端口线，组成 4 个输入/输出端口和 2 个输入端口

所有端口线均可位寻址，通过端口控制寄存器被分别（按位）配置为输入或输出。端口配置可为每个引脚选择方向（输入/输出）、推挽输出或漏极开路输出，激活拉动器件。每四个引脚一组选择边沿特性（形状）以及端口驱动器的驱动特性（输出电流）。这些 I/O 口为真正的双向口，用作输入时切换到高阻态；内部复位期间，无需激活拉动器件，所有引脚配置为输入。

所有端口线均可选择用作复用输入或输出功能。这些复用功能可被分配给各种端口引脚，对于给定应用来说，可最大程度的优化端口线的使用。因此，表 2-3 中某些功能出现了多次。

未用作复用功能的端口可作为通用 IO 口使用。

**表 2-3 XE16xyM 并行端口总结**

端口	144 <sup>1)</sup> 引脚	100 <sup>1)</sup> 引脚	64 <sup>1)</sup> 引脚	复用功能
P0 口	8	8	---	地址线， USIC 和 CAN 的串行接口线， CCU6 的输入/输出线
P1 口	8	8	---	地址线， USIC 的串行接口线， CCU6 的输入/输出线， OCDS 控制，中断
P2 口	14	14	11	地址和/或数据线，总线控制， USIC 和 CAN 的串行接口线 CCU6 和 CAPCOM2 的输入/输出 线，定时器控制信号， DAP/JTAG，中断，系统时钟控制
P3 口	8	---	---	总线仲裁信号， USIC 和 CAN 的串行接口线

端口	144 <sup>1)</sup> 引脚	100 <sup>1)</sup> 引脚	64 <sup>1)</sup> 引脚	复用功能
P4 口	8	4	---	片选信号， CAN 的串行接口线， CAPCOM2 的输入/输出线， 定时器控制信号
P5 口	16	11	7	ADC0 的模拟输入通道， CCU6 的输入/输出线， 定时器控制信号， DAP/JTAG, OCDS 控制，中断
P6 口 <sup>2)</sup>	4	3	2	ADC 控制线， CAN 的串行接口线， 定时器控制信号， OCDS 控制
P7 口	5	5	1	ADC 控制线， USIC 和 CAN 的串行接口线， CCU6 的输入/输出线， 定时器控制信号， DAP/JTAG, OCDS 控制，系统时钟输出
P8 口	7	---	---	CCU6 输入/输出线， USIC 串行接口线， DAP/JTAG, OCDS 控制
P9 口	8	---	---	CAN 串行接口线 CCU6 输入/输出线 DAP/JTAG, OCDS 控制
P10 口	16	16	16	地址和/或数据线，总线控制， USIC 和 CAN 的串行接口线， CCU6 的输入/输出线， DAP/JTAG, OCDS 控制
P11 口	6	---	---	USIC 和 CAN 的串行接口线

端口	144 <sup>1)</sup> 引脚	100 <sup>1)</sup> 引脚	64 <sup>1)</sup> 引脚	复用功能
				CCU6 的输入/输出线
P15 口	8	5	2	ADC1 的模拟输入通道 定时器控制信号

1) 这些列描述不同封装的可用端口引脚数。

2) 这些引脚的驱动器由 V<sub>DDPA</sub> 供电。

## 2.4 时钟产生

时钟产生单元通过可编程片上 PLL 和多个预分频器，为 XE16xyM 提供非常灵活的时钟信号。系统时钟 f<sub>sys</sub> 是参考时钟信号，可输出给外部系统。系统时钟 f<sub>sys</sub> 可来自多个内部和外部时钟源。

片上高精度振荡器（OSC\_HP）可驱动外部晶振或者接收外部时钟信号。振荡器时钟频率可经片上 PLL 倍频（倍频因子可编程设定），或者经预分频器分频（分频因子可编程设定）。

无需外部晶振，内部时钟源可提供时钟信号。

振荡器看门狗（OWD）可监控输入时钟，若输入时钟不可靠，则使能紧急时钟。

## 2.5 功率管理

XE16xyM 可工作在 3 V 到 5 V 之间的电源电压范围内。内部内核电源由片上内嵌的电压调节器产生，并由片上电源验证电路监控。

两个 IO 电源域通过用较低电压（3 V）给器件的主要部分供电的方式，有助于降低热量耗散，同时 ADC（5 V）仍然与模拟的 5 V 传感器信号相连。

XE16xyM 提供多种控制功耗的方式，可控制给定时间内的功耗或某段时间内的平均功耗。可使用三种机制控制功耗（部分机制可同时使用）：

- **电源电压管理** 允许关闭电源。这样可大大的降低由漏电流引起的功耗，尤其在高温情况下。上电复位可重新启动系统。
- **时钟产生管理** 控制内部和外部时钟信号的分配和频率。当前不工作的逻辑部分的时钟信号可被自动禁止，用户可降低 XE16xyM 的 CPU 时钟频率，从而大大的降低功耗。

通过可编程频率输出 EXTCLK 控制外部电路的功耗。

- **外设管理** 允许暂时禁止外设模块。可单独被禁止/使能每个外设模块。

*注：为了避免无意的产生中间状态，选择电源电压、时钟源和时钟产生方法时，需要的参数必须小心写入到相应的位域。建议用户使用本文档推荐的顺序进行电源和时钟参数设置，以确保电源系统和时钟系统的操作正确。*

## 2.6 片上调试支持（OCDS）

XE16xyM 的片上调试支持系统为用户提供各种内嵌的调试和仿真功能。XE16xyM 上运行的用户软件可以方便的在目标系统环境中进行调试。

OCDS通过调试接口和可选的断点接口由外部调试设备进行控制。调试器通过一组专用寄存器控制OCDS，这些寄存器可由JTAG接口访问。此外，OCDS系统还可以由CPU控制（如监控程序）。插入接口允许CPU执行由OCDS产生的指令。

片上硬件、软件或外部输入可触发多个断点。OCDS支持单步执行、插入任意指令，以及对整个内部地址空间的读/写访问。响应断点的方式包括：CPU暂停、调用监控程序、数据传送、或/和外部信号激活。

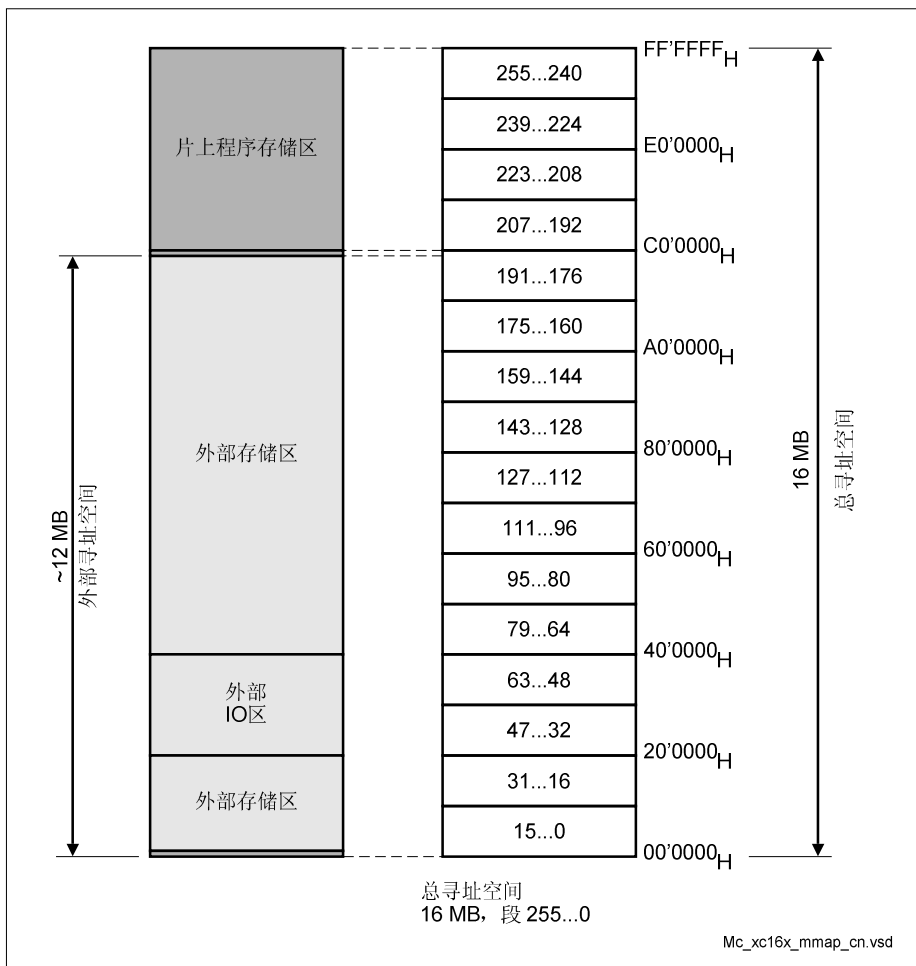
观察点的数据传送（见上文）可以通过调试接口或外部总线接口实现，以提高性能。

可使用两种方式实现调试接口：

- 通过 DAP 端口的调试接口，该接口使用 2 根 DAP 线。
- 通过符合 IEEE-1149 规范的 JTAG 端口进行调试，该接口使用 4 根 JTAG 线。

### 3 存储器结构

XE16xyM 的存储器空间按照“冯诺伊曼”体系结构组织，这意味着代码和数据的访问使用同一个线性地址空间。所有物理上独立的存储区，包括内部 ROM 和 Flash、内部 RAM、内部特殊功能寄存器区（SFR 和 ESFR）、内部 IO 区和外部存储区，这些存储器统一映射到一个共同的地址空间。



**图 3-1 地址空间总览**

XE16xyM 总的可寻址存储器空间为 16 MB，分为 256 段，每段 64 KB。每段又细分为 4 个数据页，每页 16 KB（见图 3-1）。

字节存储在奇字节地址或偶字节地址中。字按升序存储在连续的两个存储单元中，低位字节位于偶字节地址，高位字节位于紧随的奇字节地址（“Little endian”）。双字（只适用于代码）以两个连续字的形式按升序存储在地址单元中。位元始终存储在一个字地址的指定位中。位 0 是偶字节地址的最低有效位，位 15 是下一个奇字节地址的最高有效位。部分特殊功能寄存器、部分内部 RAM 以及通用寄存器都支持位寻址。

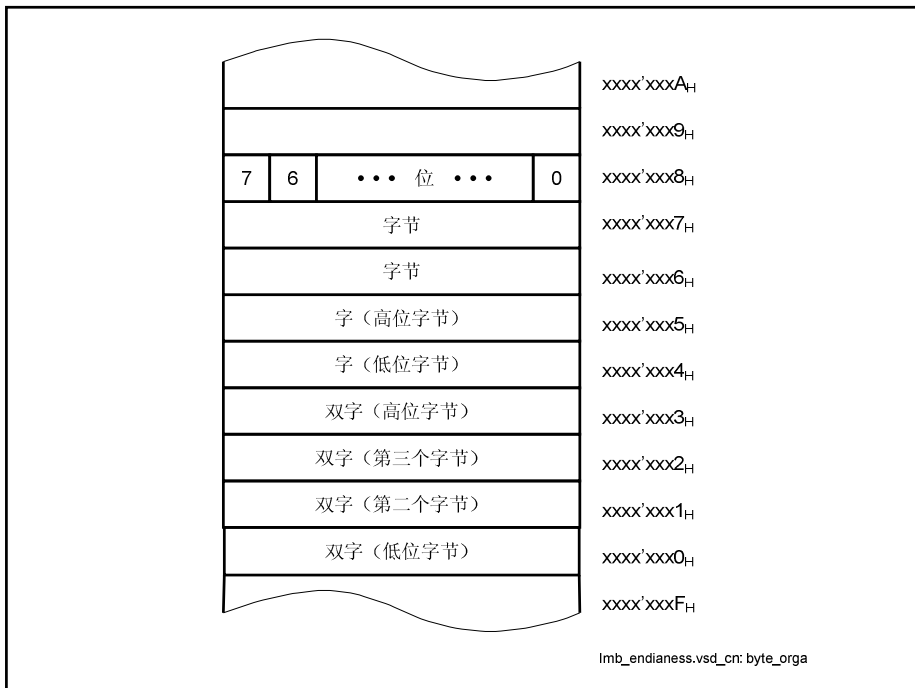


图 3-2 字、字节和位的存储（以字节为单位组织存储器）

注：构成字或双字的字节单元必须位于同一个物理存储空间内(内部、外部、ROM、RAM)，并且位于存储器的同一段、同一页中。



### 3.1 地址映射

不同的存储区以及外设寄存器（见**表 3-1**）全都映射到一个连续的地址空间上。每一个区段都可采用相同的方式进行访问。XE16xyM 的存储映射空间中有一些保留区域，从而能够和后续的增强型衍生产品向上兼容。

*注：表 3-1 给出最大可用存储器空间。实际可用的存储器容量由选择的器件型号决定。*

**表 3-1 XE16xyM 存储器映射<sup>1)</sup>**

地址区域	起始地址	结束地址	区域大小 <sup>2)</sup>	备注
IMB 寄存器空间	FF'FF00 <sub>H</sub>	FF'FFFF <sub>H</sub>	256 B	
保留	F0'0000 <sub>H</sub>	FF'FEFF <sub>H</sub>	< 1 MB	减去 IMB 寄存器
保留用作 EPSRAM	E8'8000 <sub>H</sub>	EF'FFFF <sub>H</sub>	480 KB	镜像 EPSRAM
仿真 PSRAM	E8'0000 <sub>H</sub>	E8'7FFF <sub>H</sub>	高达 32 KB	Flash 时序
保留用作 PSRAM	E0'8000 <sub>H</sub>	E7'FFFF <sub>H</sub>	480 KB	镜像 PSRAM
PSRAM	E0'0000 <sub>H</sub>	E0'7FFF <sub>H</sub>	高达 32 KB	程序 SRAM
保留用作 Flash	CD'0000 <sub>H</sub>	DF'FFFF <sub>H</sub>	1,216 KB	
Flash 3	CC'0000 <sub>H</sub>	CC'FFFF <sub>H</sub>	64 KB	
Flash 2	C8'0000 <sub>H</sub>	CB'FFFF <sub>H</sub>	256 KB	
Flash 1	C4'0000 <sub>H</sub>	C7'FFFF <sub>H</sub>	256 KB	
Flash 0	C0'0000 <sub>H</sub>	C3'FFFF <sub>H</sub>	256 KB <sup>3)</sup>	减去保留段
外部存储器区域	40'0000 <sub>H</sub>	BF'FFFF <sub>H</sub>	8 MB	
外部 IO 区 <sup>4)</sup>	21'0000 <sub>H</sub>	3F'FFFF <sub>H</sub>	1,984 KB	
保留	20'C000 <sub>H</sub>	20'FFFF <sub>H</sub>	16 KB	
USIC0-3 复用寄存器	20'B000 <sub>H</sub>	20'BFFF <sub>H</sub>	4 KB	通过 EBC 访问
MultiCAN 复用寄存器	20'8000 <sub>H</sub>	20'AFFF <sub>H</sub>	12 KB	通过 EBC 访问
保留	20'6000 <sub>H</sub>	20'7FFF <sub>H</sub>	8 KB	

**存储器结构**

地址区域	起始地址	结束地址	区域大小 <sup>2)</sup>	备注
USIC0-3 寄存器	20'4000 <sub>H</sub>	20'5FFF <sub>H</sub>	8 KB	通过 EBC 访问
保留	20'6800 <sub>H</sub>	20'7FFF <sub>H</sub>	6 KB	
MultiCAN 寄存器	20'0000 <sub>H</sub>	20'3FFF <sub>H</sub>	16 KB	通过 EBC 访问
外部存储器空间	01'0000 <sub>H</sub>	1F'FFFF <sub>H</sub>	1984 KB	
SFR 区	00'FE00 <sub>H</sub>	00'FFFF <sub>H</sub>	0.5 KB	
双口 RAM (DPRAM)	00'F600 <sub>H</sub>	00'FDFF <sub>H</sub>	2 KB	
保留用作 DPRAM	00'F200 <sub>H</sub>	00'F5FF <sub>H</sub>	1 KB	
ESFR 区	00'F000 <sub>H</sub>	00'F1FF <sub>H</sub>	0.5 KB	
XSFR 区	00'E000 <sub>H</sub>	00'EFFF <sub>H</sub>	4 KB	
数据 SRAM (DSRAM)	00'A000 <sub>H</sub>	00'DFFF <sub>H</sub>	16 KB	
保留用作 DSRAM	00'8000 <sub>H</sub>	00'9FFF <sub>H</sub>	8 KB	
外部存储器区	00'0000 <sub>H</sub>	00'7FFF <sub>H</sub>	32 KB	

1) 对阴影区域的访问被保留。在具有外部总线接口的器件中，这些访问会产生外部总线访问。

2) 标有“<”的区域大小略小于标注值，见“备注”列。

3) 第一个 Flash 段的最高 4 KB 扇区保留，用于内部使用（C0' F000<sub>H</sub> 至 C0' FFFF<sub>H</sub>）。

4) 一些流水线优化对外部 IO 区无效，这对合理控制片外外设很有必要。

## 3.2 寄存器区

可通过五个地址区域访问控制 XE16xyM 系统和外设功能的寄存器。这些地址的访问特性不同，详情请参考**章节 3.7** 和 CPU 一章。

可通过以下三个区域访问控制 XE16xyM 系统和外设功能的特殊功能寄存器 (SFR)。

- 512 字节 SFR 区 (位于 DPRAM 上方: 00'FFFF<sub>H</sub>...00'FE00<sub>H</sub>)
- 512 字节 ESFR 区 (位于 DPRAM 下方: 00'F1FF<sub>H</sub>...00'F000<sub>H</sub>)
- 4 KB XSFR 区 (位于 ESFR 区下方: 00'EFFF<sub>H</sub>...00'E000<sub>H</sub>)

USIC 和 MultiCAN 寄存器位于外部 IO 区内:

- 64 KB 外部 IO 区 (位于: 20'0000<sub>H</sub>...20'FFFF<sub>H</sub>)

IMB 寄存器位于常规存储器区。访问这个区域时, 必须考虑 CPU 流水线效应。

- 256 字节 IMB 寄存器区 (位于: FF'FF00<sub>H</sub>...FF'FFFF<sub>H</sub>)

这种地址组织方式保证了与 C166 和 XC166 系列产品向上兼容。

### 不在 IO 区内的 IMB 寄存器

请注意 IMB 寄存器不在 IO 区内。只有在 IO 区内, CPU 才能保证按照指令流中数据访问出现的顺序精确执行访问操作。IO 区外, CPU 仅能确保寻址单个地址的访问按照顺序进行。因此, 当访问 IMB 寄存器区时, 用户必须要特别留意。下面的两个例子有助于理解上述重要问题:

1. 序列: 写访问地址 A; 读访问地址 B。
2. 序列: 写访问地址 C; 读访问地址 C。

如果地址 A、B 和 C 位于 IO 区, 则存储器访问的顺序将会和它们在指令流中的顺序相似。

如果地址 A、B 和 C 位于 IO 区外, 那么流水线效应将会导致第一个指令序列中读访问地址 B 的操作在写访问地址 A 的操作之前被执行。CPU 本身能够确保第二个指令序列按照顺序执行。为了解决这个问题并强化指令执行次序, 应当在写访问地址 A 之后才执行对地址 A 的读访问或地址 B 的写访问—两种方式都将确保对地址 B 的读访问出现在对地址 A 的写访问之后。

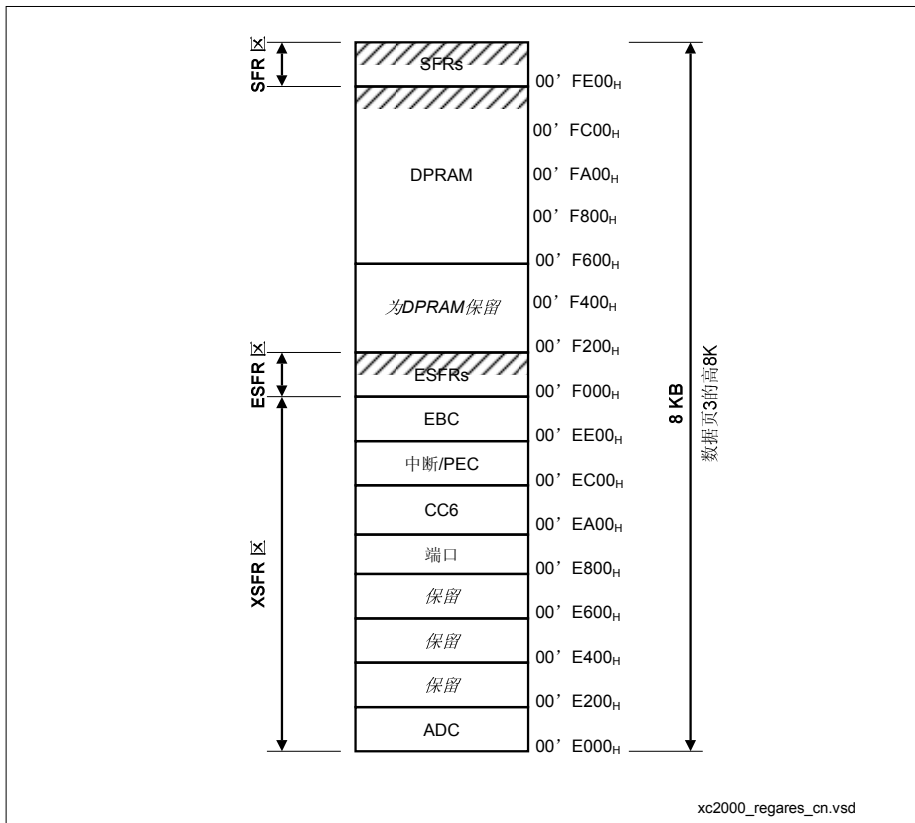


图 3-3 特殊功能寄存器映射

注：SFR 区、ESFR 区以及内部 RAM 的高 256 字节均可位寻址（见图 3-3 中斜杠区域）。

### 特殊功能寄存器

XE16xyM 的 CPU、总线接口、IO 端口及片上外设的功能均通过相关特殊功能寄存器（SFR）进行控制。

所有特殊功能寄存器均可通过间接寻址和 16 位长寻址模式访问。SFR/ESFR 区中的 SFR（字）及相应的低位字节，可通过“8 位偏移量+ 隐含基址”方式寻址。但不能对 SFR 的高位字节寻址。

注：对 SFR 中的任意一个字节进行写操作时，另外一个未被寻址的字节将被清零。

SFR 区的上半部分（地址为 00'FFFF<sub>H</sub>...00'FF00<sub>H</sub>）和 ESFR 区的上半部分（00'F1FF<sub>H</sub>...00'F100<sub>H</sub>）寄存器可位寻址，因此可以通过位寻址方式直接修改或检查各控制/状态位。

采用 8 位寻址或直接位寻址访问 ESFR 区寄存器时，首先需要执行扩展寄存器指令（EXTR），以便将短寻址机制从标准 SFR 区切换到 ESFR 区。对 16 位长寻址和间接寻址则无需如此。GPR 的 R15...R0 自动复制到两个特殊功能寄存器区，无论是在 SFR 区还是在 ESFR 区，都可通过 2 位、4 位和 8 位短寻址方式访问，而无需使用 EXTR 指令进行切换。

ESFR\_SWITCH\_EXAMPLE:

```
EXTR #4                ;Switch to ESFR area for next 4 instr.
MOV STMREL, #data16    ;STMREL uses 8-bit reg addressing
BFLDL STMCON, #mask, #data8 ;Bit addressing for bitfields
BSET WUCR.CLRTRG      ;Bit addressing for single bits
MOV T8REL, R1          ;T8REL uses 16-bit mem address,
                       ;R1 is duplicated into the ESFR space
                       ;(EXTR is not required for this access)
;---- ;-----        ;The scope of the EXTR #4 instruction ...
                       ;... ends here!
MOV T8REL, R1          ;T8REL uses 16-bit mem address,
                       ;R1 is accessed via the SFR space
```

为了尽量减少使用 EXTR 指令，ESFR 区中主要存放控制初始化以及模式选择的寄存器。尽可能将需要频繁访问的寄存器分配给标准 SFR 区。

*注：XE16xyM 有专用工具监控 ESFR 区的访问。访问 ESFR 区时会自动插入 EXTR 指令；若 EXTR 指令缺失或是过多会发出警告。*

访问 XSFR 区寄存器采用 16 位寻址，无需特别的寻址模式或预防措施。

### 通用寄存器

通用寄存器（GPR）位于全局寄存器组、或两个局部寄存器组之一，占用 16 个字长的连续地址。寄存器 PSW 中的位域 BANK 选择当前有效的寄存器组。全局寄存器组被映射到 DPRAM 区，上下文指针（CP）寄存器指定当前有效的全局寄存器组的基地址。寄存器组由最多 16 个字 GPR（R0, R1, ...R15）和/或最多 16 个字节 GPR（RL0, RH0, ...RL7, RH7）组成。16 个字节 GPR 被映射到前 8 个字 GPR 上（见表 3-2）。

与系统堆栈不同，寄存器组中的寄存器按升序（从低地址到高地址）依次存放在相应的地址单元中，最多占用 32 个字节。通用寄存器组中的 GPR 可通过 2 位、4 位或 8 位寻址方式进行访问，使用上下文指针寄存器（CP）作为基地址（与当前 DPP 寄存器的内容无关）。另外，当前有效的寄存器组中的每一位均可被单独访问（可位寻址）。

**表 3-2 通用寄存器映射到 DPRAM**

DPRAM 地址	高字节寄存器	低字节寄存器	字寄存器
<CP>+1E <sub>H</sub>	—	—	R15
<CP>+1C <sub>H</sub>	—	—	R14
<CP>+1A <sub>H</sub>	—	—	R13
<CP>+18 <sub>H</sub>	—	—	R12
<CP>+16 <sub>H</sub>	—	—	R11
<CP>+14 <sub>H</sub>	—	—	R10
<CP>+12 <sub>H</sub>	—	—	R9
<CP>+10 <sub>H</sub>	—	—	R8
<CP>+0E <sub>H</sub>	RH7	RL7	R7
<CP>+0C <sub>H</sub>	RH6	RL6	R6
<CP>+0A <sub>H</sub>	RH5	RL5	R5
<CP>+08 <sub>H</sub>	RH4	RL4	R4
<CP>+06 <sub>H</sub>	RH3	RL3	R3
<CP>+04 <sub>H</sub>	RH2	RL2	R2
<CP>+02 <sub>H</sub>	RH1	RL1	R1
<CP>+00 <sub>H</sub>	RH0	RL0	R0

XE16xyM 支持快速寄存器组（上下文）切换。DPRAM 中可以同时存在多个全局寄存器组。但是，在给定时刻，只有被上下文指针寄存器（CP）选中的全局寄存器组才有效。重新选择全局寄存器组时，只需更新寄存器 CP 即可。可用一个特殊的上下文切换指令（SCXT）执行寄存器组切换，该指令会自动保存先前的上下文，并载入新上下文。所能实现的寄存器组（任意大小）的数目只受限于可用 DPRAM 的容量。

*注：局部 GPR 组并非存储器映射，GPR 不能用长寻址和间接寻址方式进行访问。*

### **PEC 源指针和目的指针**

进行 PEC 数据传送时，PEC 源地址指针和目的地址指针位于 XSFR 区。

每个 PEC 通道使用一对指针，它们占用两个连续的字地址：源指针（SRCPx）位于低地址，目的指针（DSTPx）位于高地址（ $x=7\ldots0$ ）。此外，附加的段寄存器存放相关的源地址段和目的地址段，从而 PEC 能够在整个地址空间内任意地址之间传送数据。

进行 PEC 数据传送时，根据源指针和目的指针（由指定的 PEC 通道号决定）访问所指单元，与当前 DPP 寄存器的内容无关。

若某个 PEC 通道未被使用，相应的指针单元可用作其它用途。

*注：对 PEC 指针中的一个字节进行写操作时，该指针中另外一个未被寻址的字节将被清除。*

### 3.3 数据存储区

XE16xyM 中有两个片上 RAM 区，用于存储数据：

- **双端口 RAM (DPRAM)** 可用于存储全局寄存器组 (GPR)、系统堆栈、变量和其它数据，尤其用于存储 MAC 操作数。
- **数据 SRAM (DSRAM)** 可用于存储系统堆栈（推荐使用）、变量和其它数据。

*注：数据也可以存储在 PSRAM 中（见[章节 3.11](#)）。但是，访问数据存储区速度最快。*

根据器件型号，附加片上存储器区域可能具有特殊用途，当系统电源域关闭时，用来保持数据，XE16xyM 包含：

- **备用 RAM (SBRAM)**
- **标记存储器 (MKMEM)**

#### 双端口 RAM (DPRAM)

XE16xyM 片上集成有 2 KB 的 DPRAM (00'F600<sub>H</sub>...00'FDFF<sub>H</sub>)。如果所选择的 DPP 寄存器指向数据页 3，DPRAM 中的任意字或字节可通过间接寻址或 16 位长寻址模式访问。对任意字数据的访问通过寻址偶字节地址实现。

进行 PEC 数据传送时，根据 PEC 的源指针和目的指针访问 DPRAM，与 DPP 寄存器的内容无关。

DPRAM 的高 256 字节 (00'FD00<sub>H</sub>...00'FDFF<sub>H</sub>) 提供单个位存储功能，因此这 256 个字节可位寻址。

*注：代码不能在 DPRAM 内执行。*

*注：复位之后，初始化期间 DPRAM 地址 00'FBFE<sub>H</sub>...00'FC01<sub>H</sub> 中的值可能改变。因此不应该使用该区域保存那些复位之后仍需保留的数据。*

DPRAM 占用 3 KB 的专用区域 (00'F200<sub>H</sub>...00'FDFF<sub>H</sub>)，目前未使用的 DPRAM 地址保留待用。

#### 数据 SRAM (DSRAM)

XE16xyM 片上集成有 16 KB 的 DSRAM (00'A000<sub>H</sub>...00'DFFF<sub>H</sub>)。如果 DPP 寄存器指向数据页 3（从 00'C000<sub>H</sub> 到 00'DFFF<sub>H</sub>）或数据页 2（从 00'A000<sub>H</sub> 到 00'BFFF<sub>H</sub>），DSRAM 中的任意字或字节可通过间接寻址或 16 位长寻址模式访问。对任意字数据的访问通过寻址偶字节地址实现。

进行 PEC 数据传送时，根据 PEC 的源指针和目的指针访问 DSRAM，与 DPP 寄存器的内容无关。

*注：代码不能在 DSRAM 内执行。*



DSRAM 占用 24 KB 的专用区域 (00'8000<sub>H</sub>...00'DFFF<sub>H</sub>)，目前未使用的 DSRAM 地址保留待用。

### **备用 RAM (SBRAM)**

唤醒电源域 (DMP\_M) 为 SBRAM 提供 8 KB 存储器。当系统电源域 (DMP\_1) 关闭时，使用该存储器保存系统状态信息。

与其它存储器不同，SBRAM 未映射到处理器的地址范围内。需要通过两个地址和两个数据 SFR 读/写该存储器。访问机制的详细描述见 [章节 3.12](#)。

*注：不能从 SBRAM 中执行代码。*

*注：上电复位之后，初始化期间 SBRAM 的高 32 字节中的值可能被改变。因此不应该使用该区域保存那些上电复位之后仍需保留的数据。如果使用快速启动模式，应用软件一定不能改变该区域。*

### **标记存储器 (MKMEM)**

MKMEM 提供 4 字节存储器。在掉电期间，该存储器用于保存系统状态信息。

MKMEM 由两个 16 位 SFR 组成（与所有其它 SFR 一样可被访问），具体描述参见 SCU 一章。

*注：不能从 MKMEM 中执行代码。*

### 3.4 程序存储区

XE16xyM 中有两个片上程序存储区，用于代码/ 数据存储：

- **程序 Flash/ROM** 存储代码和常量数据。Flash 存储器可通过应用软件或 Flash 加载器（反复）编程，而 ROM 只能在工厂掩模编程。
- **程序 SRAM（PSRAM）** 存储临时代码和其它数据。比如，高级引导加载程序可以写在 PSRAM 中，继而执行该程序对片上 Flash 存储器进行编程。

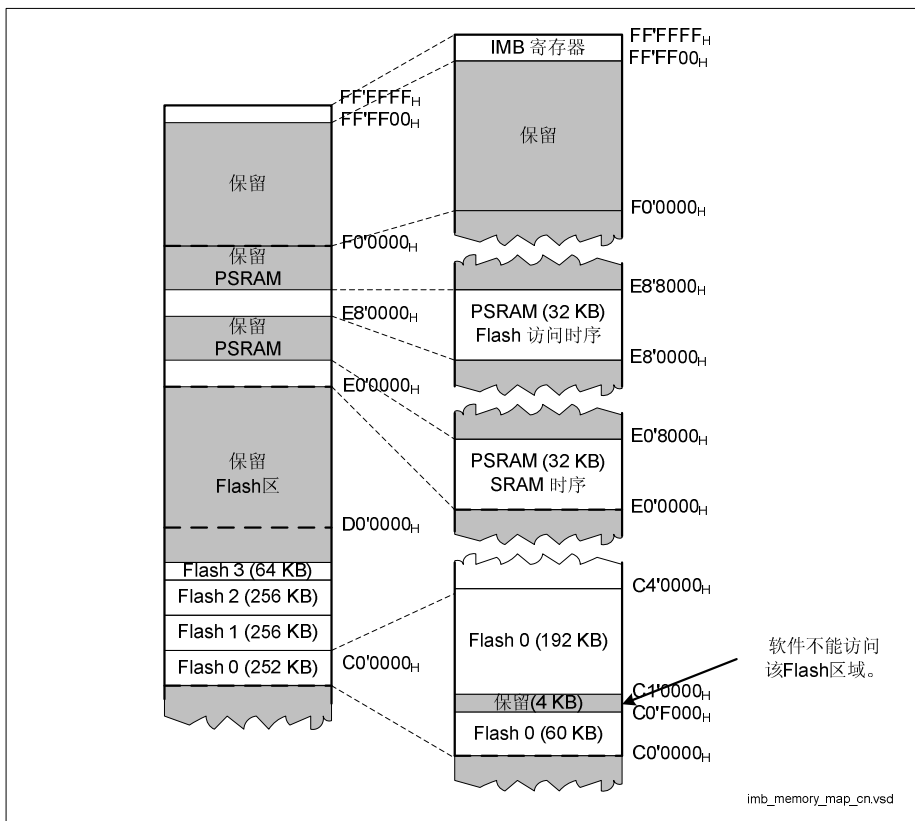


图 3-4 片上程序存储器映射

### 3.4.1 程序/数据 SRAM (PSRAM)

XE16xyM 片上集成有 32 KB 的 PSRAM (E0'0000<sub>H</sub>...E0'7FFF<sub>H</sub>)。PSRAM 能够快速执行代码, 无初始延时。因此, PSRAM 支持非顺序的代码执行 (如通过中断向量表产生)。

当选择的 DPP 寄存器指向数据页之一时, PSRAM 中的任意字或字节可通过间接寻址或 16 位长寻址模式访问。对任意字数据的访问通过寻址偶字节地址实现。

进行 PEC 数据传送时, 根据 PEC 的源指针和目的指针访问 PSRAM, 与 DPP 寄存器的内容无关。

任何数据都可以保存在 PSRAM 中。由于 PSRAM 是为取指操作而优化设计的。所以, 通过数据存储存取数据的性能更好。

*注: PSRAM 不可位寻址。*

*注: 复位之后, 初始化期间, PSRAM 的高 256 字节可能被改变。因此, 不应在该区域内保存那些复位操作之后仍需要保留的数据。*

*而且, 启动程序加载操作期间, 串行接收的数据被保存在从地址 E0'0000<sub>H</sub> 开始的 PSRAM 存储器中。*

PSRAM 占用 512 KB 的专用区域 (E0'0000<sub>H</sub>...F7'FFFF<sub>H</sub>), 目前未使用的 PSRAM 区保留待用。

### Flash 仿真

在代码开发的过程中, 经常使用 PSRAM 存储代码或数据, 这些代码或数据在随后的产品芯片中将被存储在 Flash 存储器中。为了确保执行时间相似, PSRAM 对地址范围 E8'0000<sub>H</sub>...EF'FFFF<sub>H</sub> 的访问支持第二种访问路径, 这种访问的时序参数和 Flash 时序一致。等待周期个数由 Flash 访问时序配置决定 (见 [IMB\\_IMBCTRL.WSFLASH](#)), 执行写访问时始终没有等待周期。

因为和 Flash 存储器的读操作使用相同的读访问逻辑<sup>1)</sup>, 这种对 Flash 访问时序的模拟几乎准确到单个周期。软件使用 PSRAM 进行 Flash 仿真和直接作为 PSRAM 使用, 可能会出现一些差异。仿真期间, 访问冲突可能会导致时序略微不同, 而在产品芯片中, 这种访问冲突不会出现。

时序差异的另一个原因是产品芯片中 Flash 模块中出现的访问冲突。可并行执行从不同 Flash 模块读数据和取指令操作, 而如果这两种操作访问同一个 Flash 模块, 则读数据和取指令的操作顺序进行, 此时先进行数据访问。Flash 仿真时, 这种类型的冲突不会出现。数据和指令访问都引发确定个数的等待周期 (就像是它们访问不同的 Flash

---

1) Flash 读逻辑的双重使用可能会导致无法预料的行为: 当 IMB 内核正在进行更新保护配置 (启动之后或改变安全页面之后), 对 PSRAM 的 Flash 仿真范围的读访问被阻止, 原因是 Flash 数据读访问也会被阻止。

模块) 且如果两种访问在 PSRAM 接口发生冲突, 取指操作将需要另一个附加的等待周期。

### 数据完整性

PSRAM 本身包含错误控制, 可在 ECC 和奇偶校验之间选择。详见 SCU 一章。

### 写保护

由于 PSRAM 经常用于存储对时序要求高的代码或常数, 因此 XE16xyM 为 PSRAM 提供写保护机制。在 PSRAM 中保存关键数据之后, 可使用寄存器位域 **IMB\_IMBCTRH.PSPROT** 将 PSRAM 分成只读区域和可写区域两部分。对只读部分进行写访问的操作会被阻止, 并激活一个强制中断。

### 3.4.2 非易失程序存储器 (Flash)

XE16xyM 片上集成有 832 KB 的程序存储器 Flash, 起始地址为 C0'0000<sub>H</sub>。通过字节选择线选中字和字节数据, 代码和数据的读取始终 64 位对齐。

当选用的 DPP 寄存器指向对应数据页时, 程序存储器中的任意字或字节可通过间接寻址或 16 位长寻址模式访问。对任意字数据的访问通过寻址偶字节地址实现。

进行 PEC 数据传送时, 根据 PEC 的源指针和目的指针访问程序存储器, 与 DPP 寄存器的内容无关。

*注: 该程序存储器不可位寻址。*

程序存储器占用 2 MB 的专用区域 (C0'0000<sub>H</sub>...DF'FFFF<sub>H</sub>), 目前未使用的程序存储器空间保留待用。

关于 Flash 的详细描述见页 3-18 “片上 Flash 存储器”。

## 3.5 系统堆栈

系统堆栈可位于 XE16xyM 存储空间 (包括外部存储器) 的任意位置。

所有的系统堆栈操作, 均由 24 位堆栈指针寻址相关的堆栈存储单元。堆栈指针寄存器 (SP) 中存放着堆栈指针的低 16 位 (堆栈指针的偏移量), 堆栈指针段寄存器 (SPSEG) 中存放着堆栈指针的高 8 位 (堆栈段)。按照从高地址到低地址的顺序生成系统堆栈 (压栈顺序为从高到低)。系统堆栈只支持字访问。

数据压栈前寄存器 SP 递减, 数据出栈后 SP 递增。系统堆栈只支持字访问。

使用寄存器 SP 进行堆栈操作时, 系统堆栈最大 64 KB。堆栈必须位于由寄存器 SPSEG 规定的段内。

堆栈指针总是指向最新的系统堆栈入口, 而非下一个可用的堆栈地址。

堆栈上溢寄存器（STKOV）和堆栈下溢寄存器（STKUN）用来控制所选堆栈区域的上下边界。这两个堆栈边界寄存器均可保护数据不被破坏。

为了发挥系统的最佳性能，建议用户将堆栈安排在 DPRAM 或者 DSRAM 中。将堆栈设置在 DPRAM 中时，可能会和寄存器组或者 MAC 操作数冲突。

### 3.6 保护位

XE16xyM 提供一种特殊机制，以防止保护位（这些位可被片上硬件修改）因软件访问而被无意修改（参见 CPU 一章的“位操作单元”）。下列寄存器中属性为“rwh”和“wh”的位和位域支持位保护：

**表 3-3 XE16xyM 保护位**

寄存器	所属模块	注
TFR	CPU	强制中断标志寄存器
PSW	CPU	处理器状态字
PECISNC	CPU	PEC 通道中断请求标志
MPU_PRA	MPU	保护范围地址
SCU_GSCSWR EQ	SCU	全局状态控制软件请求
RTC_ISNC	RTC	中断节点共享请求标志
CC2_OUT	CC2	比较输出位
GPT12E_T2CON	GPT	GPT1 定时器 T2 标志
GPT12E_T3CON	GPT	GPT1 定时器 T3 标志和输出翻转锁存
GPT12E_T4CON	GPT	GPT1 定时器 T4 标志
GPT12E_T6CON	GPT	GPT1 定时器 T6 输出翻转锁存
xIC	CPU、SCU 和外设 单元	所有中断控制寄存器。 这些寄存器的完整列表在中断和异常情况控制一章中给出。
Px_OUT	端口	所有端口输出寄存器

### 3.7 IO 区

XE16xyM 地址空间中的下列区域属于 IO 区：

- **外部 IO 区** 用于访问片外外设（或存储器），也包含片上 LXBUS 外设，如 MultiCAN 模块或 USIC 模块。外部 IO 区地址范围：20'0000<sub>H</sub>...3F'FFFF<sub>H</sub>（2MB）。
- **内部 IO 区** 用于访问片上外设。内部 IO 区分为三个区：
  - SFR 区，地址范围：00'FE00<sub>H</sub>...00'FFFF<sub>H</sub>（共 512 B）
  - ESFR 区，地址范围：00'F000<sub>H</sub>...00'F1FF<sub>H</sub>（共 512 B）
  - XSFR 区，地址范围：00'E000<sub>H</sub>...00'EFFF<sub>H</sub>（共 4 KB）

*注：外部 IO 区支持真正的字节访问。而内部 IO 区并不支持真正的字节传送，对一个字节进行写操作时，和该字节组成一个字存储单元的另一个字节被清零。*

由于外设模块和存储器的控制方式不同，因此 IO 区具有一些特殊属性：

- IO 区访问不经过缓存（Buffer）和高速缓存（Cached）。保存 IO 区的读写结果时不使用回写缓存和高速缓存。
- 不执行不确定的读操作。延迟直到所有不确定因素都确定后才能执行读操作（如条件分支之后的预取操作）
- 禁止数据前送。由于在执行 IO 写操作后，外设可改变自身的内部状态，因此，直到流水线中所有挂起的 IO 写操作完成之后，才能执行 IO 读操作。

### 3.8 外部存储空间

XE16xyM 的寻址空间高达 16 MB。只有部分寻址空间被内部存储区占用或被保留。外部存储区占用约 12 MB 寻址空间。通过 XE16xyM 的外部总线接口访问外部存储器。

**存储器组的大小可选：**外部存储器组的最大容量取决于被激活的地址位个数，从 64 KB（A15...A0 被激活）到 12 MB（A23...A0 被激活）不等。存储器组的逻辑大小和地址分配通过设定相关的地址窗来决定，大小从 4 KB 到 12 MB 不等。

由可编程的片选信号选择不同的存储器组，通过地址总线直接寻址每个存储器组。

**XE16xyM 还支持四种不同的总线类型：**

- 16 位复用总线（复位后的缺省模式）
- 8 位复用总线
- 16 位非复用总线
- 8 位非复用总线

有关外部总线设置和控制的具体内容请参阅外部总线控制器一章。

外部的字和字节数据只能通过间接或 16 位长寻址模式访问（使用四个 DPP 寄存器之一）。对外部操作数的访问不支持短寻址模式。对任意字的访问通过寻址偶字节地址实现。

进行 PEC 数据传送时，通过 PEC 的源指针和目的指针访问外部存储器，与 DPP 寄存器的内容无关。

*注：外部存储器不可位寻址。*

### 3.9 存储器边界越界

XE16xyM 的地址空间隐含的划分为大小相同的存储区块（区块内实际使用的存储区域大小不同）和不同的逻辑存储区。跨越这些代码或数据区的边界时，需要特别加以注意，以确保控制器能够执行想要的操作。

地址空间被划分为多种不同的**存储区**（如提供）：包括 SFR 区、片上程序或数据 RAM 区、片上 ROM/Flash（若可用）、片上 LXBUS 外设（若有集成）以及外部存储区。

访问连续存放在不同存储区的数据时不会出现问题。然而，执行存放在不同存储区的**代码**时，必须通过分支指令明确的进行存储区切换，系统不支持代码的越界连续访问，这样会导致错误的结果。

*注：从外部存储区切换到片上 RAM 区发生在段 0 内。*

**段**是以 64 KB 为单位的连续区块。取指令时，根据代码段指针 CSP 进行寻址；存取数据时，可直接使用段编号取代标准的 DPP 机制进行寻址。

取指时，代码段不会自动改变、必须使用指令来进行切换。使用 JMPS、CALLS、RETS 指令可实现代码段的切换。

**数据页**是以 16 KB 为单位的连续区块。存取数据时，通过数据页指针 DPP3...DPP0 寻址；还可直接使用数据页编号取代标准的 DPP 机制进行寻址。每个 DPP 寄存器可选择 1024 个数据页中的任意一个。由 16 位数据地址的最高两位选择用于当前数据访问的 DPP 寄存器。因此，跨越 16 KB 数据页边界的 16 位连续数据地址将使用不同的数据页指针，而所寻址的物理空间不必连续。

### 3.10 片上 Flash 存储器

本节描述 XE16xyM 的嵌入式 Flash 存储器。

- **章节 3.10.1** 定义 Flash 的专用名词和 Flash 存储器的结构。
- **章节 3.10.2** 描述 Flash 的工作模式。
- **章节 3.10.3** 概述所有 Flash 操作。
- **章节 3.10.4** 给出 Flash 操作的详细描述。
- **章节 3.10.7、章节 3.10.8 和章节 3.10.9** 描述如何保持数据完整性及存储器保护。
- **章节 3.10.10** 讨论 Flash EEPROM 仿真。
- **章节 3.10.11** 描述 Flash 存储器的中断产生。

**章节 3.11** 说明 Flash 存储器如何嵌入到 XE16xyM 存储器结构中，并列影响 Flash 存储器行为的所有 SFR。

#### 3.10.1 定义

本节首先定义 Flash 的专用名词和一些缩写。Flash 存储器为非易失（“NVM”）存储器，该存储器基于浮栅单晶体管单元技术。称之为“非易失”是因为当存储器的电源关闭时，存储器的内容仍能保持。

##### 逻辑和物理状态

Flash 存储器的内容不能象 SRAM 一样可直接修改。修改数据是一个复杂的过程，所需时间通常比读取 Flash 需要的时间更长。

- **擦除**：存储器单元被擦除后的状态为逻辑 0。强制 Flash 单元进入该状态的操作称为“擦除”。擦除操作的最小单位是一页（见下面的描述）。器件出厂时，Flash 存储器为完全被擦除状态。
- **编程**：存储器单元的编程状态为逻辑 1。被擦除单元进入该状态的操作称为“编程”。每页只能被编程一次，重新编程该页之前，必须先进行擦除。

上面列出的过程有下列限制：

- **数据保持能力**：在该时间持续期间 Flash 单元的数据可被可靠读取。保持时间是一个统计数据，取决于 Flash 阵列的工作条件（温度分布）以及对 Flash 阵列的访问操作。随着编程/擦除次数（见耐受力）的增加，数据保持能力下降。漏极干扰和门极干扰也会降低数据保持能力。
- **耐受力**：如上所述，数据保持特性随着编程/擦除次数的增加而减少。只要一个 Flash 单元所在的页或扇区被擦除或编程，则该 Flash 单元被编程/擦除的次数也加 1。可编程/擦除的次数被称为“耐受力”。与数据保持特性相同，该指标



也是一个统计数据，取决于工作条件、对 Flash 单元的使用以及 Flash 的质量等级。

- **漏极干扰 (Drain Disturb)**：因为使用“单晶体管”Flash 单元技术，每次编程访问会对同一扇区的所有页造成轻微干扰。较长时间之后这些“漏极干扰”使得 0 和 1 值变得无法区分，因而引起读错误。该效应仍然和数据保持性有关。受到漏极干扰次数多的 Flash 单元的数据保持性较低。Flash 阵列的物理扇区彼此隔离，因此对不同扇区的页进行操作不会引起漏极干扰。当使用页擦除特性时，需要考虑干扰效应。

编程和擦除的持续时间以及对于耐受力、数据保持性和漏极干扰等限制都在数据手册中给出。

**注意：器件不具备避免实际应用违反这些限制的预防措施。**

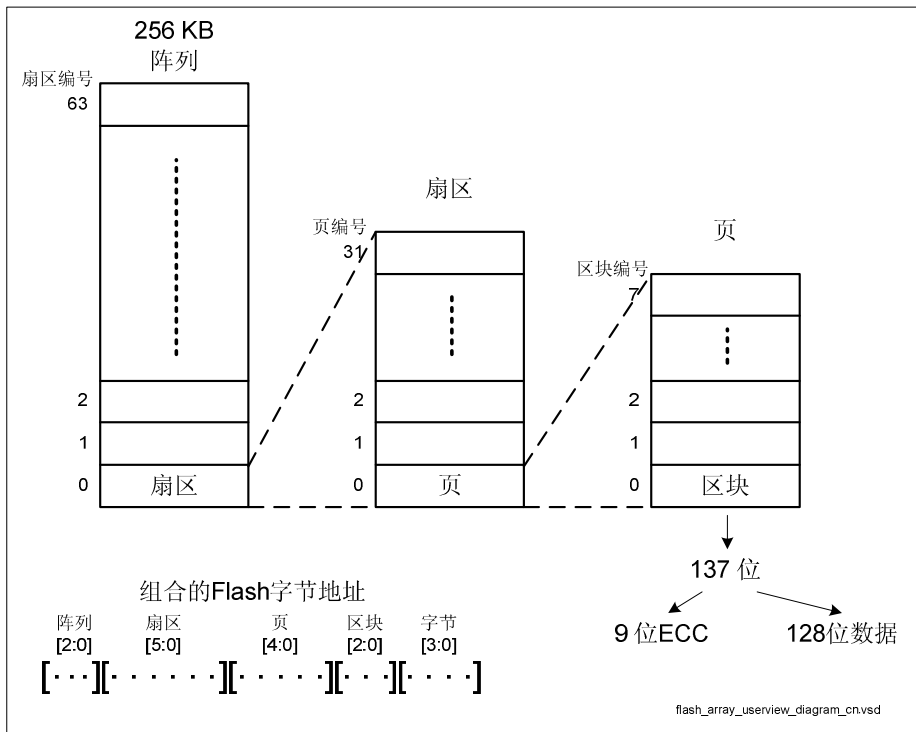
## 阵列结构

Flash 存储器采用一种分层结构：

- **区块**：一个存储器区块由 128 个用户数据位（即 16 个字节）和 9 个 ECC 位组成。每次读访问给出一个区块的内容。
- **页**：一页由 8 个区块组成（即 128 个字节）。编程操作始终改变整页的内容。
- **扇区**：一个扇区由 32 页组成（即 4096 字节）。上文中提到漏极干扰会影响一个扇区内所有页。不同扇区的各页彼此隔离。
- **阵列**：每个 256 KB 阵列具有 64 个扇区<sup>1)</sup>，64 KB 阵列具有 16 个扇区。“阵列”通常还包含所有支持扇区并行工作的伴随逻辑，如写缓存、高电压逻辑和数字逻辑。
- **存储器**：XE16xyM 的整个 Flash 存储器由 4 个 Flash 阵列组成。

图 3-5 给出 256 KB 阵列 Flash 结构。64 KB 阵列 Flash 的结构相似。

1) Flash 0 中，一个扇区被保留用作器件内部使用，软件不能访问该扇区。



**图 3-5 Flash 结构**

### 3.10.2 工作模式

IMB、Flash 存储器以及每个 Flash 模块都有特定的工作模式。其中一些模式定义它的时钟和电源、以及模拟逻辑（如振荡器和电压泵）的工作状态。整个系统的工作模式（如启动模式）也会影响 Flash 存储器的行为。

其它模式定义 Flash 的功能，下面讨论这些工作模式。

#### 3.10.2.1 标准读模式

复位之后和执行一个完整的启动过程之后，Flash 存储器及其所有模块进入到“标准读模式”。该模式下，Flash 存储器和片上 ROM 的行为相同。下列情况下，Flash 进入标准读模式：

- 复位之后，当完整的启动过程执行完毕

- 完成持续时间较长的命令（如“擦除”或“编程”）之后，通过清除“忙”标志来应答命令结束状态。
- 执行完任何一个其它命令之后，立即进入标准读模式
- 检测到执行错误，如试图对写保护的区域进行写访问，发出错误的密码等，以及所有的序列错误之后。

对于持续时间长的命令，读模式一直有效，直到接收到命令序列的最后一个命令且操作被启动为止。

### 3.10.2.2 命令模式

接收到命令序列的最后一条命令之后，被寻址的 Flash 模块（并非整个 Flash 存储器！）进入到命令模式。对于大多数命令来说，用户将不会注意到这种情况，因为这些命令被立即执行，随后 Flash 模块再次进入到读模式。对于持续时间长的命令，Flash 模块处于命令模式的时间为几个 ms，通过设置相应的“忙”标志报告该情况。正处于忙状态的 Flash 模块中的数据不能被读取，但其它不忙的 Flash 模块仍可读。新的命令序列通常不被接受，它会引发序列错误，直到正在运行的操作结束。然而，在某些情况下，为了能够同时编程和擦除独立的 Flash 模块，新的命令序列被接受。

对正忙的 Flash 模块进行读访问会使得 CPU 挂起，直至该 Flash 模块再次进入读模式。挂起的 CPU 只响应复位请求。由于此时 CPU 不能处理任何中断，因此必须避免出现这种状态。

*注：由于发送给正忙 Flash 存储器的命令序列并非总是被硬件以序列错误的方式拒绝，因此，相比先前不支持并行操作的微控制器而言，用户在处理所有命令时必须更加谨慎。只有在检查确定某 Flash 模块不再忙碌时，才能向其发出新的命令序列。当使用“读取正忙的 Flash，导致 CPU 挂起”特性时，上面的措施尤其关键。进一步的建议可在[章节 3.10.5](#)（序列错误）和[章节 3.10.6](#)（并行操作）中找到。*

### 3.10.2.3 页面模式

“**进入页面模式**”命令使得 Flash 模块进入到页面模式。下面描述页面模式的内容。页面模式下，Flash 模块仍可被读取（因此，Flash 模块同时处于“读模式”）。同时仅有一个 Flash 模块能够处于页面模式。

当 Flash 存储器处于页面模式时——即 Flash 模块之一处于页面模式——则不允许执行以下的命令序列：所有的擦除序列和“改变读裕量”序列。这些命令序列被忽略且报告序列错误。

### 3.10.3 操作

Flash 存储器支持以下操作：

- 取指
- 读数据
- 改变数据和控制保护的命令序列

#### 3.10.3.1 从 Flash 存储器中读取指令

PMU 以 64 位对齐的方式读取指令。这些代码读取请求被送至 Flash 存储器。执行读访问需要的周期可变（取决于系统时钟频率）。因为 Flash 不能提供数据可用性的信号，因此 IMB 内核必须知道所需的周期数。因此，等待周期数保存在 **IMB\_IMBCTRL** 寄存器中。

Flash 读访问的整个持续时间为：**IMB\_IMBCTRL.WSFLASH+1** 个时钟周期。

WSFLASH 值和系统时钟频率和具体器件型号有关，具体参数参见相关数据手册。

对 Flash 存储器的一次读访问给出 128 个数据位和一个 9 位 ECC 值。ECC 值用于检测并纠正（如可能）错误。被寻址的 64 位数据所在的 128 位数据块被发送给 PMU。整个 128 个数据位和 9 个 ECC 位以及它们的地址保存在 IMB 内核中。如果下一条取指请求与该地址匹配，则无需再次读访问 Flash 存储器，而是从缓存中得到数据。一个周期之后，可从缓存中得到数据。此情况下，不需要 Flash 读等待周期。

被保存的数据可看作是某种指令高速缓存（lcache）。为了支持指令自修正（如引导加载器），当相应的地址被写入时（即擦除或编程）该高速缓存无效。

除了取指缓存，IMB 内核还具有可提高性能的增强特性——线性代码预取。当由 **IMB\_IMBCTRL.DLCPF = 0** 使能该特性时，CPU 执行自己的缓存或取指缓存内指令的同时，IMB 内核自动读取后续的指令。由于该特性仅按照线性顺序读取后续指令（不分析指令流），因此执行较长线性顺序指令的效率更高。对于跳转或调用指令密度较高的代码序列，该特性甚至可能导致性能的降低，此时，应当关闭该特性。

#### 3.10.3.2 从 Flash 存储器中读取数据

由 DMU 从 Flash 存储器中读取数据。数据读取始终以 16 位字的方式进行。每次读请求，Flash 存储器给出 128 位加 ECC 值，描述见 [页 3-22 “从 Flash 存储器中读取指令”](#)。

IMB 内核必须根据全部 128 位来评估 ECC 数据。所请求的 16 位数据将交付给 DMU。所有数据和 ECC 位保存在数据寄存器中，地址保存在地址寄存器中。对于所有后续的数据读取操作，将其地址与地址寄存器中的地址进行比较，如果地址匹配，则一个周期之后从数据寄存器中得到所请求的数据。未从该高速缓存得到数据的每个数据读取操作使得该高速缓存的内容失效。当得到所请求的数据时，高速缓存再次包含有效数据。

当对该地址进行写访问（即擦除或编程）时，则该小型数据高速缓存失效。

对于数据读取操作，IMB 内核不执行任何自动预取。

### 3.10.3.3 向 Flash 存储器写数据

直接向 Flash 存储器写数据的方式不能改变 Flash 存储器的内容。除了读取操作，执行其它所有 Flash 操作都需要使用命令序列。命令序列由向 Flash 存储器地址范围写某些数据的操作组成。所有目标为该地址范围的数据转移操作被解释为命令序列。如果它们和定义的命令序列不匹配或 IMB 正忙而无法接受新的命令序列，则报告产生序列错误。

### 3.10.3.4 命令序列

如前所述，需要执行命令序列才能改变 Flash 存储器的内容。

表 3-4 命令序列概述

命令序列	描述	详情见
复位到读	复位 Flash，使其进入到读模式并清除错误标志	页 3-25
清除状态	清除错误和状态标志	页 3-25
改变读裕量	改变读裕量	页 3-25
进入页面模式	准备用于编程的页面	页 3-26
进入安全页面模式	准备用于编程的安全页面	页 3-27
加载页面字	用数据加载页面	页 3-27
编程页面	启动页面编程过程	页 3-28
擦除扇区	启动扇区擦除过程	页 3-29
擦除页面	启动页面擦除过程	页 3-30
擦除安全页面	启动安全页面擦除过程	页 3-31
禁止读保护	用密码暂时禁止读保护	页 3-31
禁止写保护	用密码暂时禁止写保护	页 3-32
重新使能读/写保护	重新使能保护	页 3-33

### 3.10.4 命令序列的详述

本节用伪汇编代码定义命令序列。称之为“伪”是因为所有的地址都是直接地址，而在真实的汇编代码中通常这点是不可能满足的。

由 1 到 6 个目标地址为 Flash 存储器的数据转移操作组成的序列被称为命令。数据转移操作必须是“字”类型，即不是字节转移指令。下面各节描述每个命令。描述所涉及的地址和数据需要用到下面的缩写：

- **PA:** “页面地址”，为目标页面的基地址。例如第一个页面的地址为  $C0'0000_H$ 。第二个阵列页面 13 的地址  $PA = C0'0000_H + 1 \cdot 256 \cdot 1024$  (阵列)  $+ 0 \cdot 4 \cdot 1024$  (扇区)  $+ 13 \cdot 128$  (页面)  $= C4'0680_H$ 。
- **SECPA:** “安全页面地址”，为安全页面的虚拟地址。称之为“虚拟”是因为 SECPA 仅被用作命令序列的参数，以识别安全页面，而安全页面的物理存储被隐藏。

定义两个安全页面：

**SecP0:** 地址  $C0'0000_H$

**SecP1:** 地址  $C0'0080_H$

- **WD:** “写数据”。被写入到写缓存中的 16 位数据字。
- **SA:** “扇区地址”，根据 Flash 模块地址，给出如图 3-6 中所定义的物理扇区编号。用两个例子来阐述：
  3. 第一个阵列的物理扇区 16 基地址为  $C0'0000_H$ ， $SA = C0'0000_H + 16 \cdot 4 \cdot 1024 = C1'0000_H$ 。
  4. 第二个 256 KB 阵列的基地址为  $C4'0000_H$  (如表 3-1 所示)。所以该阵列的物理扇区 3 的  $SA = C4'0000_H + 3 \cdot 4 \cdot 1024 = C4'3000_H$ 。
- **PWD:** “密码”，为 64 位密码。以 4 个 16 位数据字的形式传送密码。 $PWD0 = PWD[15:0]$ ， $PWD1 = PWD[31:16]$ ， $PWD2 = PWD[47:32]$ ， $PWD3 = PWD[63:48]$ 。
- 地址 **XX**，随后紧跟两个 16 进制数，例如“ $XXAA_H$ ”。如果命令寻址某个 Flash 模块，**XX** 必须被翻译为其基地址。因此对于寻址 Flash 0 的所有命令“ $XXAA_H$ ”意味着  $C0'00AA_H$ ， $C4'00AA_H$  对应 Flash 1 模块， $C8'00AA_H$  对应 Flash 2 模块。如果一个命令 (如“清除状态”) 寻址整个 Flash 存储器，则必须使用 Flash 0 模块的基地址。
- 数据 **XX**，随后紧跟两个 16 进制数，如  $XXA5_H$ 。这是一个“不必在意”的数据字，仅低字节必须匹配特定序列。所以在本例中，像  $12A5_H$  或  $79A5_H$  的所有数据字都可使用。
- **MR:** “裕量”，用来定义读裕量的 8 位数字。MR 的值为： $00_H$  (正常读)， $01_H$  (hard read 0)， $02_H$  (alternate hard read 0)， $05_H$  (hard read 1)， $06_H$  (alternate hard read 1)，其它所有 MR 值保留。

## 复位到读

参数: -

定义:

MOV XXAA<sub>H</sub>, XXF0<sub>H</sub>

**时序:** 单周期命令，不置位任何“BUSY”标志。但是应当注意紧随其后的、写访问 IMB 内核的操作被停滞数个周期，在此期间 IMB 内核正在中止前面的命令。

**描述:** 内部命令状态机被复位至初始状态，并返回到读模式。已经启动的编程或擦除操作不受影响，继续进行（“**复位到读**”命令——即所有命令——当 IMB 内核正忙时，无论如何都不会被接受）。

“**复位到读**”命令为单周期命令。命令序列期间，该命令可被用于复位命令解读器并使 IMB 内核返回到其初始状态。该命令清除 Flash 状态寄存器 IMB\_FSR 中的所有错误标志并退出当前有效的页面模式。“**复位到读**”命令不能用于中止当前有效的命令模式。当至少一个 Flash 模块忙碌时，该命令不被接受且置位 SQER<sup>1)</sup>。

该命令清除: PROER、PAGE、SQER、OPER、ISBER、IDBER、DSBER、DDBER。

## 清除状态

参数: -

定义:

MOV XXAA<sub>H</sub>, XXF5<sub>H</sub>

**时序:** 单周期命令，不会置位任何 BUSY 标志。

**描述:** Flash 状态寄存器内的标志 OPER、SQER、PROER、ISBER、IDBER、DSBER、DDBER 被清除。另外，过程状态标志: (PROG、ERASE、POWER、MAR) 被清除。

当 Flash 模块处于命令模式时切勿发送该命令，此时该命令被忽略，报告产生序列错误。

## 改变读裕量

参数: MR

定义:

MOV XXAA<sub>H</sub>, XXB0<sub>H</sub>

MOV XX54<sub>H</sub>, XXMR<sub>H</sub>

1) 在 XE16xyM 中，有一种不符合该规则的例外情况：当一个 Flash 模块正在进行编程或擦除操作且 FAPI 已接收到可同时执行的命令序列（“擦除扇区”，“擦除页面”，“进入页面模式”，“加载页面字”，“编程页面”）中的部分而非全部命令周期时，则执行复位到读命令，而不发出序列错误。

**时序:** 双周期命令，被寻址的 Flash 模块的“BUSY”标志置位大约 30 ms。

**描述:** 该命令序列用于改变一个 Flash 模块的读裕量。由第二个转移操作的地址 XX 确定目标 Flash 模块。Flash 模块需要一段时间改变其读电压。在此期间 BUSY 置位且不能访问该 Flash 模块。其它 Flash 模块仍可读。

参数“MR”定义读裕量：

- 00<sub>H</sub>: 正常读裕量
- 01<sub>H</sub>: Hard read 0 裕量
- 02<sub>H</sub>: alternate hard read 0 裕量
- 05<sub>H</sub>: hard read 1 裕量
- 06<sub>H</sub>: alternate hard read 1 裕量
- 其它值: 保留

页 3-42 “读裕量”的描述可帮助用户更好的理解这部分内容。

当 Flash 存储器处于页面模式或任意一个 Flash 模块处于命令模式时，不能发出该命令。此时该命令被忽略并报告一个序列错误。

**注:** 如页 3-69 “裕量控制”中的注解，命令序列“编程页面”，“擦除扇区”，“擦除页面”和“擦除安全页面”命令序列将读裕量复位至 00<sub>H</sub>，即正常读裕量。Flash 被唤醒时，出现的情况相同。

## 进入页面模式

参数: PA

定义:

```
MOV XXAA11, XX5011
MOV PA, XXAA11
```

**时序:** 双周期命令，被寻址 Flash 模块“BUSY”标志置位大约 20 个时钟周期<sup>1)</sup>。

**描述:** 准备对页面地址 PA 进行页面编程操作时，进入到页面模式。（只有“加载页面字”命令进行的写数据操作才能被接受）。

使用该命令，IMB 内核将模块写寄存器的写指针初始化为 0，以使其指向第一个字。由状态寄存器 IMB\_FSR\_BUSY 中的 PAGE 位单独指示每个 Flash 模块是否处于页面模式。允许一个 Flash 模块同时处于页面模式和读模式，因此该 Flash 模块仍可读。当读取被寻址的 PA 时，Flash 存储器给出相应的内容。页面模式可被中止，寄存器 IMB\_FSR\_BUSY 内的相关 PAGE 位可被“复位到读”命令清除。页面模式期间出现的

1) 当使用该命令中止另一个 Flash 模块的页面模式时，“BUSY”标志置位的持续时间增加至大约 30 个时钟周期。



新的“**进入页面模式**”命令中止实际的页面模式，由错误标志 SQER 指示该情况，并启动新的页面操作。如上所述，同时只能有一个 Flash 模块处于页面模式。如果 Flash 处于页面模式，接收到擦除命令或“**改变读裕量**”命令时，这些命令被忽略，并报告序列错误。

当其它 Flash 模块正在执行用户数据擦除或编程命令时（即：不编程或擦除安全页面或不执行其它禁止序列时），一个 Flash 模块可以进入页面模式。

如果要编程的扇区已设置读保护，只有使用带 4 个密码的解锁命令序列“**禁止写保护**”禁止写保护之后，才能接受“**进入页面模式**”命令。如果设置全局写保护（同时进行读保护），且如果没有设置特定保护的扇区，还可以使用命令“**禁止读保护**”。当接收到“**进入页面模式**”命令时，写保护未被禁止，则不执行该命令，IMB\_FSR\_PROT 内的保护错误标志 PROER 被置位。

注：在先前的产品系列中（如 XC16x），“进入页面模式”不置位“BUSY”标志。在这些器件中，可在发出“进入页面模式”命令之后直接发送“加载页面字”命令。在 XE16xyM 中，必须等到“BUSY”标志被清除之后，才能发出“加载页面字”命令序列。

## 进入安全页面模式

参数: SECPA

定义:

```
MOV XXAAn, XX55n
MOV SECPA, XXAAn
```

时序: 双周期命令，Flash 模块 0 的“BUSY”标志置位大约 100 个时钟周期。

描述: 该命令与“**进入页面模式**”命令基本相同（参见上面的描述），不同之处在于：被寻址的页面（SECPA）属于 Flash 存储器的安全页面，而不属于用户 Flash 范围。只有 Flash 读保护和写保护都无效时（RPA = 0 且 WPA = 0），才能执行该命令。否则不接受该命令并且置位 PROER。

当任意一个 Flash 模块处于命令模式时，该命令被拒绝且 SQER 置位。

使用该命令设置密码并再次禁用密码的操作描述见页 3-45 “**保护处理详述**”。

## 加载页面字

参数: WD

定义:

```
MOV XXF2n, WD
```

时序: 单周期命令，不会置位任何“BUSY”标志。请注意接收到该命令时，当 IMB 正在忙着将其模块写寄存器内容复制到 Flash 模块写缓存时，紧随该命令之后的写访问 IMB 内核或读访问 Flash 存储器的操作都被停滞几个时钟周期。停滞期间，CPU

不能执行任何动作！所以用户软件或者能够接受该停滞时间（中断延迟的最坏情况需要将此停滞时间考虑在内）或软件必须避免该阻滞访问。

**描述：**用 16 位字加载 IMB 内核模块写寄存器，并递增写指针。用 64 个“加载页面字”命令序列填充 128 字节的写缓存（即完整一页）。字地址不是由命令决定，“**进入页面模式**”命令将写指针设置至 0，每个“**加载页面字**”命令执行之后该指针递增。

对模块写寄存器的连续数据访问属于且只能被页面模式接受。该单周期命令的地址始终相同（F2<sub>H</sub>）。这些低段地址位还确定“**加载页面字**”命令且连续写数据被加载到模块写寄存器中。高段地址位 XX 应当指向目标页。IMB 内核始终使用上一个“**进入页面模式**”命令所使用的页面地址。

8 个“**加载页面字**”命令之后，IMB 内核的 128 位模块写寄存器被填充完毕，IMB 内核计算 9 位 ECC 并将和数据一起传送到 Flash 模块的写缓存中。随后，IMB 将模块写寄存器的写指针设置为 0。接下来的 8 个“加载页面字”命令再次填充模块。所有 8 个模块填充完毕之后，可使用“**编程页面**”命令触发编程过程（将写缓存的内容传送到 Flash 阵列中）。

当 IMB 内核传送整个模块写寄存器到 Flash 模块时，有几个时钟周期不能接受新数据。在此期间收到的“**加载页面字**”命令被 IMB 内核停滞。

所有模块的写缓存接收到新数据之前，如果调用“**编程页面**”命令，那么剩余的位被清零。

如果使用命令的次数多于 8 次，附加数据会丢失。由序列错误标志指示溢出情况。但是随后的“**编程页面**”命令的执行不会被压制（不会中止页面模式）。

当接收到“**加载页面字**”命令，且 Flash 不处于页面模式时，由 IMB\_FSR\_OP 中的 SQER 报告序列错误。页面模式期间，接收到新的“**进入页面模式**”命令或“**复位到读**”命令，或者应用复位的情况下，写缓存中的写数据丢失。当前的页面模式中止，如果是接收到新的“**进入页面模式**”命令的情况，则再次进入具有新地址的新“**进入页面模式**”命令。

## 编程页面

**参数：**-

**定义：**

```
MOV XXAAn, XXA0n
MOV XX5An, XXAn
```

**时序：**双周期命令，编程操作期间，所选中的 Flash 模块的“BUSY”标志置位。接收到该命令之后，IMB 内核被阻滞几个时钟周期，完成编程操作之前，IMB 再次被阻滞几个时钟周期。在此期间，写访问 Flash 存储器地址范围以执行另一个命令序列的操作使得 CPU 停滞。

**描述：**Flash 模块的写缓存中的内容被编程到 Flash 阵列中。启动编程过程之前，如果最后一个数据块未被完全填充，该命令完成 ECC 计算并将数据复制到写缓存中。

Flash 模块的选择和要编程的页面取决于最后一个“**进入页面模式**”命令使用的页面地址。用户软件应当始终寻址目标页面。

被选中的 Flash 模块自动执行编程过程。CPU 未被该操作占用，因此可继续工作。

只有被寻址的 Flash 模块处于页面模式时，才能接受“**编程页面**”命令（否则不执行该命令，而是报告一个序列错误）。“**编程页面**”命令会中止页面模式，由相关 PAGE 标志被复位指示，并进入到命令模式，状态寄存器 IMB\_FSR\_OP 内的 PROG 标志被激活，IMB\_FSR\_BUSY 中相关的 BUSY 标志被置位。

编程过程结束之后，BUSY 标志被清除，PROG 仍然保持置位，以指示完成了哪种操作，PROG 标志将会被上电复位或“**清除状态**”命令清除。

不可能对正忙的 Flash 模块进行读访问。该读操作被停滞，直到 Flash 模块再次准备好，可以被读访问。

若将被编程的扇区的写保护有效，由于 Flash 不处于页面模式（描述见“**进入页面模式**”），“**编程页面**”命令不被接受。

若将要编程的页面为安全页面（仅在安全页面模式下该命令才能被接受），执行该命令之后，新的保护配置（包括关键字或保护确认码）直接生效。执行该命令期间，所有命令都会被拒绝，并报告序列错误。

当 IMB 内核读取新的保护配置时，对任何一个 Flash 模块进行的所有 DMU 访问被停滞。

## 擦除扇区

参数: SA

定义:

```
MOV XXAAH, XX80H
MOV XX54H, XXAAH
MOV SA, XX33H
```

**时序:** 三周期命令，整个擦除操作期间，被寻址的 Flash 模块的 BUSY 标志置位。接收到该命令之后，IMB 内核被阻滞几个时钟周期，完成擦除操作之前，IMB 再次被阻滞几个时钟周期。在此期间，对 Flash 存储器地址范围的写访问使得 CPU 停滞。

**描述:** Flash 阵列中被寻址的物理扇区被擦除。接下来对该扇区进行的数据读操作将给出带有正确 ECC 值的全零数据。

由选中 Flash 模块自动执行擦除过程。CPU 未被该操作占用，因此可继续工作。

由最后一个命令周期内的 SA（扇区地址）寻址要擦除的扇区。

在“**擦除扇区**”命令的最后一个周期之后进入到命令模式，由 IMB\_FSR\_OP 内的 ERASE 标志指示。开始擦除操作之后，状态寄存器 IMB\_FSR\_BUSY 内的相关 BUSY 标志置位。擦除操作结束之后，BUSY 标志被清除而 ERASE 标志保持置位。ERASE 标志可由上电复位或“**清除状态**”命令清除。

不可能对正忙的 Flash 模块进行读访问。XE16xyM 支持对不忙的 Flash 模块进行读访问。对正忙的 Flash 进行的读访问被停滞，直到 Flash 模块再次准备好，可以进行读取。

如果要擦除的扇区设置了写保护，使用解锁命令序列“**禁止写保护**”成功禁止写保护之后，才能接受擦除扇区命令。如果设置了全局写保护（以及读保护），且如果没有设置特定保护的扇区，还可使用“**禁止读保护**”命令。当接收到“**擦除扇区**”命令时，写保护未被禁止，则不执行该命令，且 IMB\_FSR\_PROT 内的保护标志 PROER 被置位。

当 Flash 存储器处于页面模式时，不能发出此命令。在这种情况下，命令被忽略并报告一个序列错误。

## 擦除页面

参数: PA

定义:

```
MOV XXAAn, XX80n
MOV XX54n, XXAAn
MOV PA, XX03n
```

**时序:** 三周期命令，整个擦除操作期间，被寻址的 Flash 模块的 BUSY 标志置位。接收到该命令之后，IMB 内核被阻滞几个时钟周期，完成擦除操作之前，IMB 再次被阻滞几个时钟周期。在此期间，对 Flash 存储器地址范围的写访问使得 CPU 停滞。

**描述:** 擦除被寻址的页面。接下来的对该页面进行的数据读操作将给出带有正确 ECC 值的全零数据。

在“**擦除页面**”命令的最后一个周期，进入到命令模式，由 IMB\_FSR\_OP 内的 ERASE 标志指示。开始擦除操作之后，状态寄存器 IMB\_FSR\_BUSY 内的相关 BUSY 标志置位。完成擦除操作之后，BUSY 标志被清除而 ERASE 标志保持置位。ERASE 标志可由上电复位或“**清除状态**”命令清除。

不可能对正忙的 Flash 模块进行读访问。XE16xyM 支持对不忙的 Flash 模块进行的读访问。对正忙的 Flash 进行的读访问被停滞，直到 Flash 模块再次准备好，可以进行读取。

如果要擦除的页面所属的扇区设置了写保护，只有禁止写保护之后，才能执行该命令（参见“**擦除扇区**”）。

使用页面擦除功能时，必须要特别小心，不能超过同一扇区内其它页的漏极干扰界限。

Flash 存储器处于页面模式时，一定不能发出此命令。在这种情况下，命令被忽略并报告一个序列错误。

## 擦除安全页面

**参数:** SECPA

**定义:**

```
MOV XXAAH, XX80H
MOV XX54H, XXA5H
MOV SECPA, XX53H
```

**时序:** 三周期命令，整个擦除操作期间，Flash 0 模块的 BUSY 标志被置位。

**描述:** 擦除被寻址的安全页面。

该命令与“**擦除页面**”命令基本相同，不同之处在于：被寻址的页面（SecP0 或 SecP1）不属于用户可见的 Flash 存储器范围。只有在禁止所属扇区的读保护和写保护之后，才能执行该命令。

重新编程安全页面操作的详细描述请参考[页 3-47 “保护处理举例”](#)。

两个安全页面（SecP0 和 SecP1）的结构见[页 3-46 “安全页面结构”](#)。

执行该命令擦除安全页面之后，新的保护配置（包括关键字或保护确认码）立即生效。

当 IMB 内核读取保护配置时，对任意 Flash 模块进行的所有 DMU 访问都被停滞。

Flash 存储器处于页面模式或任意一个 Flash 模块处于命令模式时，一定不能发出该命令。此时该命令被忽略并报告一个序列错误。

## 禁止读保护

**参数:** PWD

**定义:**

```
MOV XX3CH, XXXXH
MOV XX54H, PWD0
MOV XXAAH, PWD1
MOV XX54H, PWD2
MOV XXAAH, PWD3
MOV XX5AH, XX55H
```

**时序:** 六周期命令，不置位任何 BUSY 标志。

**描述:** 暂时禁止 Flash 读保护和—如果激活—整个 Flash 存储器的全局写保护。

IMB\_IMBCTRH 内的 RPA 位被复位。

该命令序列为保护命令序列，使用 4 个用户定义的密码字发布该命令或检查编程设置的关键字。每个密码字需要一个时钟周期。如果第二或第四个密码代表“**复位到读**”命令的代码，其被解释为密码且不执行复位操作。16 位密码字与“安全页面 0”内的关键字进行内部比较。如果一个或多个密码与其对应的关键字不一致，被保护的扇区仍处

于锁定状态，且由 Flash 状态寄存器指示出现保护错误（PROER）。在此情况下，下一个应用复位之后，才能接受新的“**禁止读保护**”命令或“**禁止写保护**”命令。

*注：“禁止读”（或写）保护命令执行期间，只有四个密码字与相应的关键字完成比较之后，才能指示密码比较错误。*

*注：在确认安全页面 1 的保护之前，可使用该命令序列检查关键字的正确性。由 IMB\_FSR\_PROT 中的标志 PROER 指示错误关键字。*

正确执行该命令之后，整个 Flash 存储器被解锁，且 Flash 状态寄存器（IMB\_FSR\_PROT）内的读保护禁止位 RPRODIS 被置位。如果 Flash 存储器仍被全局写保护（WPA = 1），且如果所有扇区未被单独写保护，此时可对所有扇区进行擦除和编程操作。读保护（包括全局写保护，如果选择此特性）保持禁止，直到“**重新使能读/写保护**”命令被执行，或直到下一次应用复位（包括硬件和软件复位）。

当任意一个 Flash 模块处于命令模式时，一定不能发出该命令。此时该命令被忽略并报告一个序列错误。

## 禁止写保护

参数: PWD

定义:

```
MOV XX3Ch, XXXXh
MOV XX54h, PWD0
MOV XXAAh, PWD1
MOV XX54h, PWD2
MOV XXAAh, PWD3
MOV XX5Ah, XX05h
```

*时序：*六周期命令，不置位任何 BUSY 标志。

*描述：*暂时禁止全局 Flash 写保护或/和所有被保护扇区的写保护。IMB\_IMBCTRH 内的 WPA 位被复位。

该命令序列为保护命令序列，使用 4 个用户定义的密码字来解除该保护命令（描述见上面的“**禁止读保护**”命令）。

正确执行该命令之后，所有写保护的扇区被解锁，由 Flash 状态寄存器（IMB\_FSR\_PROT）中的位 WPRODIS 指示该情况。现在可对所有扇区进行擦除和编程操作，直到：

- 执行“**重新使能读/写保护**”，或
- 接收到下一次应用复位（包括硬件和软件复位）

当任意一个 Flash 模块处于命令模式时，一定不能发出该命令。在此情况下，该命令被忽略并报告一个序列错误。

## 重新使能读/写保护

参数: -

定义:

MOV XX5E<sub>H</sub>, XXXX<sub>H</sub>

时序: 单周期命令, 不置位任何 BUSY 标志。

描述: 恢复 Flash 读和写保护。

该单周期命令清除 RPRODIS 和 WPRODIS。触发 IMB 内核从安全页面 0 中恢复 RPA 和 WPA 的保护状态 (参见 [页 3-43 表 3-6 “决定 RPA 和 WPA 的 “Flash 状态”](#)”)。所以实际上该命令将恢复所有类型的、被暂时禁止的保护设置。

该命令执行之后被立即撤销。

当任意一个 Flash 模块处于命令模式时, 不能发出该命令。此时该命令被忽略并报告一个序列错误。

### 3.10.5 序列错误

将数据字 (即 16 位) 转移到 Flash 地址范围的操作被命令解读器解释为命令序列。所有字节转移操作被忽略, 并引发一个序列错误 (通过置位 SQER 报告该错误)。

一旦命令解读器检测到不能作为合法序列执行的数据转移操作, 则立即报告一个序列错误。

*注: 寻址未实现的 Flash 区或掉电 Flash 模块的数据转移操作不会进入命令解读器, 因此不会引发序列错误。通常, 下一个正确的命令序列将会引发序列错误, 因为该命令序列被解释为前一个命令序列的延续。因此, 除了检查 SQER 是否置位, 还需继续评估其它标志 (如 PAGE、PROG、ERASE), 例子见 [章节 3.10.6](#)。*

通常当至少一个 Flash 模块正忙时, 接收到的每个数据转移操作都会导致序列错误。但是为了支持命令序列的同时执行, 在特定条件下, 上述情况不会导致序列错误。下面列出可使得 SQER 置位报告序列错误的情况:

- 如果一个 Flash 模块处于命令模式, 而正在运行的命令不允许同时执行新命令, 立即报告 SQER。
- 如果至少一个 Flash 模块处于命令模式, 且正在运行的命令允许同时执行新命令, 只有当新命令寻址正忙的 Flash 模块时, 才报告 SQER。
- 如果至少一个 Flash 模块处于命令模式, 一旦检测到不允许并行执行的命令序列的命令周期 (即当接收到的数据不属于 “进入页面模式”, “加载页面字”, “编程页面” 或 “擦除页面”), 立即报告 SQER。

命令的同时执行问题总结见 [表 3-5](#)。

**表 3-5 命令同时执行**

当任何 Flash 模块处于以下模式时，接收到新序列	页面模式	正在执行正常擦除或编程	正在执行阻滞序列 <sup>1)</sup>
复位到读	复位到页面模式	SQER <sup>2)</sup>	SQER <sup>2)</sup>
进入页面模式	SQER 并重新进入页面模式	OK <sup>3)</sup>	SQER
进入安全页面模式	SQER 并重新进入页面模式	SQER	SQER
加载页面字	OK	页面模式下，OK	SQER/- <sup>4)</sup>
编程页面	OK	页面模式下，OK	SQER/- <sup>4)</sup>
擦除页面/扇区	SQER	OK <sup>3)</sup>	SQER
擦除安全页面	SQER	SQER	SQER
*保护	OK	SQER	SQER
清除状态	OK	SQER	SQER
改变读裕量	SQER	SQER	SQER

1) 安全页面的“阻滞序列”为“擦除安全页面”、“编程页面”。当正在进行上述操作时，“改变读裕量”、“进入页面模式”、“进入安全页面模式”才被阻滞。

2) 如页 3-25 “复位到读”中的描述，该规则仅有一个例外情况。

3) 如果新的命令序列针对处于读模式的不同 Flash 模块，否则 SQER 置位。

4) 不会发生这种情况，因为仅在页面模式下允许“编程页面”命令，且不能进入页面模式。

引发序列错误的其它条件已在上面的命令描述中给出。

### 3.10.6 同时执行擦除和编程操作

所有 Flash 模块<sup>1)</sup>可被同时编程和擦除。对于高速 Flash 编程而言存在例外。正常情况下，最多一个 Flash 模块进行编程或擦除操作的同时其它 Flash 模块可被读取。

命令序列描述中的限制对并行操作有以下影响：

1) 因为电源和其它器件特定的原因，还可能有一些附加限制。在相关数据手册中描述允许同时进行的过程（包括读操作）。本节进描述逻辑硬件功能。



- 应当在一个不能被中断的序列中开始编程任务：“进入页面模式”，然后 64 个“加载页面模式”，最后是“编程模式”。任何其它 Flash 模块的命令序列都不应当中断该序列。
- 当所有 Flash 模块处于读模式时，才能进行所有安全页面处理。
- 当所有 Flash 模块处于读模式时，才可能清除错误和状态标志。例外情况是通过 **IMB\_ECC\_STAT** 进行的 Flash 模块的特定处理。
- 只有当 IMB 内核不在进行命令解读操作时，才能成功完成正在进行的编程或擦除任务（即只有当 IMB 内核准备好接受新命令序列且没有新的已经启动而并未结束的序列时，才能清除正在进行的任务的 **BUSY** 标志）。

因此同时编程 Flash 模块必须按照如下顺序进行：

1. 向每个 Flash 模块发送“擦除扇区”命令序列。
2. 等待直到所有“BUSY”标志被清除。在此期间，可从外部读取编程数据。
3. 向第一个 Flash 模块发出“进入页面模式”，64 个“加载页面字”和“编程页面”命令序列。继续向其它 Flash 模块发出相同的命令序列。
4. 等待直到所有“BUSY”标志被清除。可利用这段时间从外部读取编程下一页所需的数据。
5. 验证所有 Flash 模块的编程数据。
6. 继续步骤 3 到 5，直到所有被擦除扇区的所有页都被编程。
7. 继续步骤 1 到 6，直到所有扇区都被编程。

建议用户使用能够尽早检测出不正确的序列的如下命令过程：

1. “清除状态”并检查 **SQER** 是否为 0。
2. 向每个 Flash 模块发送“擦除扇区”命令序列。发送每个命令序列时，检查 **SQER**。
3. 等待，直到所有“BUSY”标志被清除。在此期间从外部读取用于编程的数据。
4. 检查 **SQER**，该标志会指示出发出的命令序列不正确。
5. “清除状态”并再次检查 **SQER**。“清除状态”之后，如果 **SQER** 被置位，则前一个“擦除扇区”命令还未完成。
6. 发出“进入页面模式”，检查 **PAGE** 标志是否置位，并检查 **SQER** 是否仍保为 0，向第一个 Flash 模块发出 64 个“加载页面字”和“编程页面”。“编程页面”命令被接受之后，检查 **PAGE** 是否被清除，**SQER** 是否仍保持清除状态。其它 Flash 模块继续进行该命令过程。

7. 等待直到所有“BUSY”标志被清除。可利用这段时间从外部读取编程下一页所需的数据。
8. 验证所有 Flash 模块中的编程数据。
9. 继续步骤 6 到 8，直到被擦除扇区的所有页都被编程。
10. 继续步骤 1 到 9，直到所有扇区都被编程。

### 3.10.7 数据完整性

本节描述检测和防止 Flash 存储器的数据被无意修改的方法。

#### 3.10.7.1 纠错码 (ECC)

Flash 存储器单元在产品规范规定的时间内被干扰或丢失其数据值的概率极低。为了达到规定的器件可靠性标准，每 128 位 Flash 数据块带有一个 9 位 ECC 值。这种冗余设置提供了 SEC-DED 能力，即“单位错误纠正和双位错误检测”。可纠正所有的单位错误（且检测出单位错误），以及检测出所有的双位错误，甚至可检测出大部分三位错误但其中一部分被当作有效数据或纠错之后的数据。

由寄存器 **IMB\_FSR\_PROT** 和 **IMB\_ECC\_STAT** 指示检测到错误。软件能够通过寄存器 **IMB\_INTCTR** 选择触发强制中断的错误类型。系统控制单元还提供修改错误处理的更多方法（见页 8-153 “**SCU 强制中断控制寄存器**”）。被使能的 Flash 模块强制中断请求当作“Flash 访问强制中断”进行处理。出现双位错误时，始终用空数据字代替读取的数据。

#### 3.10.7.2 中止编程/擦除检测

ECC 应当保护 Flash 存储器免受那些内在故障（通常仅影响单位数据）的影响，中断正在运行的编程或擦除过程可能会导致大量数据被破坏：

- 擦除之前，擦除过程首先将所有存储器单元编程到 1。因此根据擦除过程被中断的时间，数据可能处于不同状态。可能是旧的数据、全 1、随机值、弱全 0 者最后是全 0。
- 编程过程将所有位同时从 0 编程至 1。如果该过程被中断，读取所有置位位时，并非都返回 1 或包含弱 1。

寄存器 **IMB\_FSR\_OP** 包含 ERASE 和 PROG 位。这些位保持置位直到下一个“清除状态”命令或上电复位。因此，如果应用复位中断擦除或编程过程，这些位中间的一个会保持置位，从而可检测出擦除或编程过程的中断。在完成编程/擦除过程之后，要求软件发出“清除状态”命令以使能对编程/擦除操作的评估。

另一种可能的避免编程/擦除过程中止的方式是通过正确配置 SCU，防止复位。

如果编程或擦除过程被上电复位中止（如因为电源失效），即使通过读取受影响的 Flash 地址范围，也不存在可靠的方式来检测这种编程/擦除中止操作。即使采用裕量读

取的措施，也有可能无法发现早期或者后期的中止过程，这种中止操作可能将长期影响 Flash 的可靠性。

因此，应用必须确保 Flash 过程的执行能够不被中断，并处于特定工作条件下：如通过早期压降警告，防止软件启动 Flash 过程。

所有受中止的 Flash 过程影响的地址范围都必须被擦除并重新编程。

### **3.10.7.3 裕量读**

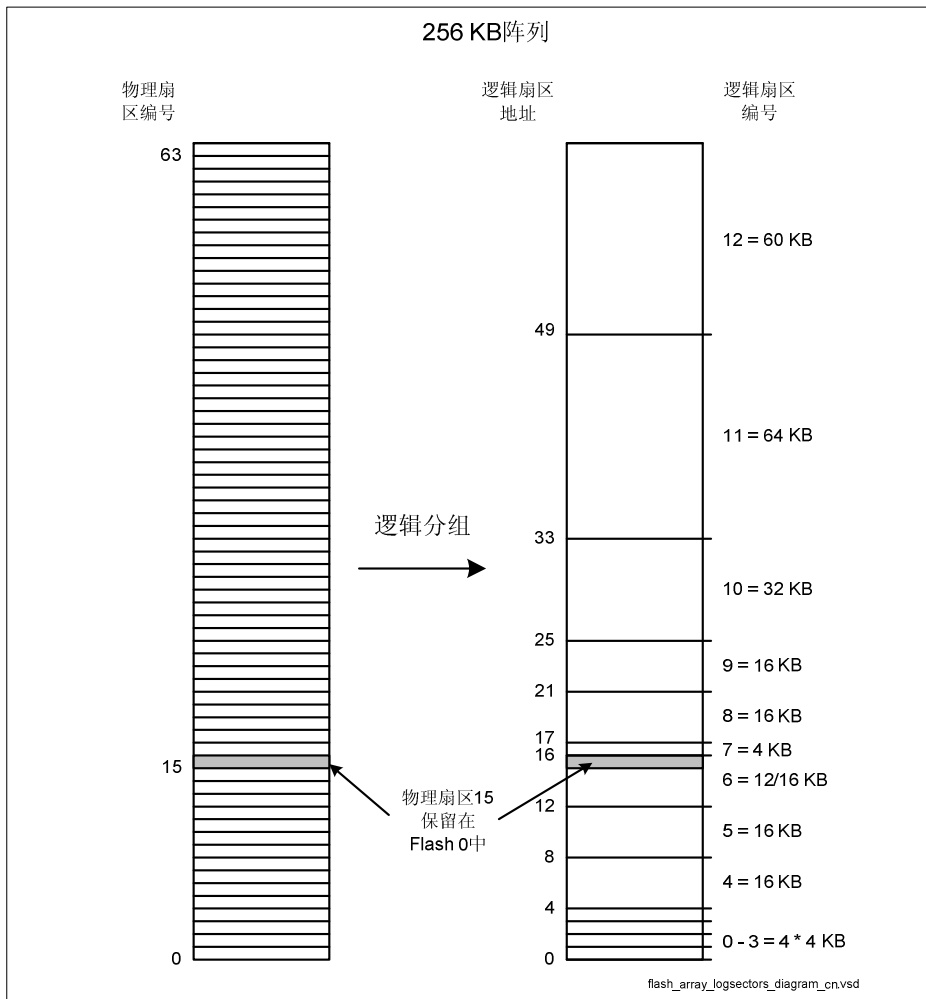
可采用某个裕量进行裕量读以验证 Flash 数据的可读性。通常产品下线编程之后，直接使用裕量读作为附加检查。如前所述，该方法不能作为检测被中断的编程或擦除过程的可靠方法，但是可增加检测到此类情况的概率。

使用“Hard read 0 裕量”读取弱 0 返回 1，使用“Hard read 1 裕量”读取弱 1 返回 0。通过命令序列“**改变读裕量**”来改变读裕量，并由状态寄存器“**IMB\_MAR0**”报告操作状态。

### **3.10.7.4 保护概述**

Flash 存储器支持对整个存储器的读和写保护以及对每个逻辑扇区的单独写保护。

**图 3-6** 给出 256 KB 阵列的逻辑扇区结构。64 KB 阵列的逻辑扇区结构与此相似，区别仅在于逻辑扇区 7 到 12 不存在。



**图 3-6 逻辑扇区**

如果设置读保护并且读保护有效，复位之后从外部存储器或从内部 RAM 启动时，任何 Flash 读访问都会被禁止。调试访问和插入的 OCDS 指令的执行都被禁止。复位之后从内部 Flash 启动时，由 Flash 内部用户代码控制所有 Flash 访问操作，因此只要用户不特别禁止，如使能调试接口之前，一直允许进行 Flash 访问操作。

缺省情况下，读保护包含对整个（全局）Flash 的写保护，包括所有 Flash 模块。为了避免将清除程序编程到 Flash（此操作读取整个 Flash 并通过外部总线或串行接口写数据），上述措施非常必要。如果写保护被单独禁止，只有读保护有效期间，才可能对 Flash 进行编程或擦除访问。用户需要使用正确密码来暂时禁止对 Flash 的写和读保护。

XE16xyM 器件还具有对一个扇区进行特定读保护的特性。由软件锁定 Flash 存储器扇区，以保护代码和数据。该特性禁止所有对被保护扇区的编程和擦除操作。设置写保护时，支持保护整个或部分 Flash 存储器不受未经授权的编程或擦除访问，并保护所有扇区免受病毒软件干扰。

用户可以通过编程位于两个“安全页面”（SecP0/1）中的特定安全配置字来使能 Flash 读和写保护。任何复位之后，命令状态机（IMB 内核）检查安全配置并在相应寄存器中保存（并指示）安全配置。如果任一保护被使能，则相关安全页面也受到保护。

为了使用户可以暂时禁止读保护或/和写保护，Flash 提供一种密码检查机制。只有使用正确的 64 位密码，才能使 Flash 暂时进入解保护状态，且保护命令序列被使能。如果未完成“**重新使能读/写保护**”命令，在下次复位时解保护状态终止。根据由用户直接编程到“安全页面 0”（SecP0）内的四个 16 位关键字（总共 64 位）进行密码检查。

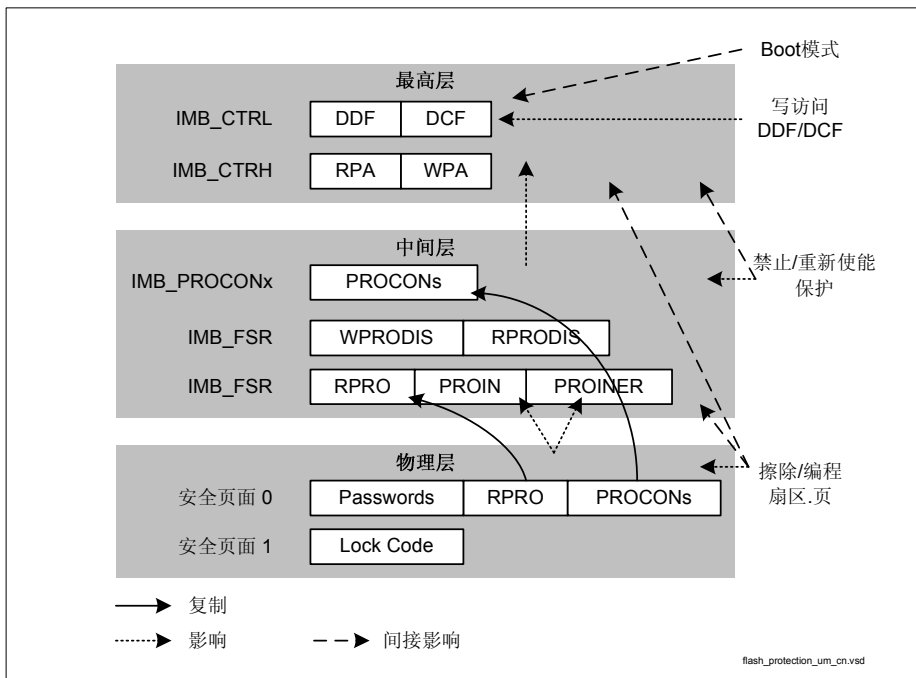
为了使芯片免受任何应力或辐射侵害的影响，当相应的配置被选中后，如果 Flash 阵列内、读路径上或寄存器内发生任何违例情况，缺省进入保护状态。在寄存器和安全页面中，保护控制位始终编码为两位，将“00<sub>B</sub>”和“11<sub>B</sub>”作为非法状态的指示，其它为受保护状态。

### 3.10.8 保护处理详述

页 3-37 “保护概述”简要描述了 Flash 存储器的不同保护状态。保护处理可被分成几个不同层次且这些层次之间相互影响（见图 3-7）：

- 最底层由安全页面 SecP0 和 SecP1 的物理内容组成。启动期间，使用该信息初始化保护系统。
- 中间层由报告物理层状态（IMB\_PROCONx）和保护状态（IMB\_FSR\_PROT）的寄存器组成。命令序列可暂时改变保护状态，该信息反应在 IMB\_FSR\_PROT 中。
- 最高层由 IMB\_IMBCTR 寄存器的 4 个位域代表。这些位域定义客户软件的保护权限（当前允许或不允许读或写访问）。

由 IMB 内核控制所有相连的 Flash 模块的保护状态。此时，IMB 能够监控 CPU 发出的所有访问操作。



**图 3-7 保护层**

### 3.10.8.1 最底层“物理状态”

复位之后，从下述信息中恢复器件的保护状态：

- 安全页面 1 包含“锁定代码”。由两个字数据（32 位）组成。如果其值为 AA55AA55<sub>H</sub>，那么安全页面 0 决定保护状态。否则（即未发现锁定代码）器件处于“未保护状态”。安全页面 0 的内容仍然被复制到页 3-41 “中间层 Flash 状态”所描述的寄存器中，但是在未保护状态下，这些值被忽略。
- 安全页面 0 包含双位 RPRO，写保护位 SnU 和 4 个密码。如果位域 RPRO 包含有效的入口值 01<sub>B</sub> 或 10<sub>B</sub>，页面有效且器件处于“已设置保护状态”。由页面内容决定启动之后的安全设置。如果 SecP0 包含无效的 RPRO 入口值，则器件处于“错误保护”状态。

总结：安全页面的内容决定器件状态：“未保护状态”、“已设置保护状态”或“错误保护状态”。这些状态反映在下一层的寄存器设置中。

通常器件出厂时处于“未保护状态”。

安全页面的详细结构描述见页 3-46 “安全页面的结构”。

### 3.10.8.2 中间层“Flash 状态”

中间层由寄存器 IMB\_PROCONx、IMB\_FSR\_PROT 和控制这些寄存器的命令以及安全页面的内容组成。

启动期间，IMB 内核检查物理状态并通过 IMB\_FSR\_PROT 中的下列各位指示这些状态：

- “未保护状态”：PROIN = 0，PROINER = 0。
- “已设置保护状态”：PROIN = 1，PROINER = 0。
- “错误保护状态”：PROIN = 0，PROINER = 1

第四种可能的设置 PROIN = 1 和 PROINER = 1 无效，不会出现此组合。

启动过程中，用安全页面 0 的内容初始化 IMB\_PROCONx 寄存器。位 DSBER 和 DDBER 指示是否出现 ECC 错误。因此客户软件有可能检测受干扰的安全页面并更新它们的内容。

#### 命令

可由命令序列控制 IMB\_FSR\_PROT 中的其它位：RPRODIS、WPRODIS 和 PROER，这些位与其它位一起定义下一层保护的有效性。系统启动之后，这 3 位都为 0。

如果用户提供了保存在 SecP0 内的正确密码，命令“禁止读保护”将 RPRODIS 设置为 1。如果提供的密码不正确，则位 PROER 被置位，RPRODIS 保持不变。为了防止搜索正确密码的“暴力破解法”，密码检测被锁定。因此提供的第一个密码不正确

之后，所有后续的甚至是正确的密码也会被拒绝并置位 PROER。只有应用复位或通过擦除 SecP0 才能退出该状态。

再次清除 RPRODIS 位的应用复位或“**重新使能读/写保护**”命令，能够再次使能被禁止的保护。

位 PROER 可被应用复位或“**复位到读**”以及“**清除状态**”命令复位。

如果提供了正确密码，命令“**禁止写保护**”置位 WPRODIS 至 1。该位和前面描述的 RPRODIS 操作方式相似。

命令“**重新使能读/写保护**”清除 RPRODIS 和 WPRODIS。

如果写访问寻址不允许的地址范围，则命令“**进入页面模式**”、“**进入安全页面模式**”、“**擦除页面**”、“**擦除安全页面**”和“**擦除扇区**”置位 PROER。由下一层决定是否允许写访问。

**表 3-6** 总结了“Flash 状态”如何决定 IMB\_IMBCTRH 寄存器内的位域 RPA 和 WPA。在该表和以下各节的描述中，使用短符号来表示双位的值：1 代表激活，0 代表非激活，‘#’代表无效且‘-’代表无关（包括无效状态）。符号‘|’代表逻辑或。



**表 3-6 决定 RPA 和 WPA 的“Flash 状态”**

IMB_FSR. PROIN	IMB_FSR. PROIN ER	IMB_FSR. RPR O	IMB_FSR. RPR ODIS	IMB_FSR. WPR ODIS	RPA 和 WPA 中相应的安全等级
0	0	-	-	-	未保护状态： RPA = 0, WPA = 0。
1	0				保护已设置状态（可能被禁用，具体如下）：
		0	-	0	RPA = 0, WPA = 1
		0	0	1	RPA = 0, WPA = 0
		1   #	0	0	RPA = 1, WPA = 1
		-	1	1	RPA = 0, WPA = 0（均被禁用）
		1   #	0	1	RPA = 1, WPA = 0
		1   #	1	0	RPA = 0, WPA = 1
0	1				错误的保护状态（具体如下）：
		-	0	0	RPA = 1, WPA = 1
		-	0	1	RPA = 1, WPA = 0
		-	1	0	RPA = 0, WPA = 1
		-	1	1	RPA = 0, WPA = 0

### 3.10.8.3 最高层“保护状态”

该层主要由 IMB\_IMBCTR<sub>H</sub> 寄存器内的四个位域 DCF、DDF、WPA 和 RPA 组成。这些位域和低层的寄存器共同决定有效的保护状态。上面提到的一些命令序列也直接影响这些位域。为了提高对辐射或电源控制影响的抵抗力，所有四个位域都编码为 2 位。通常“01”代表激活，“10”代表非激活，另外两个状态“00”和“11”代表无效，被当作“受攻击”状态。

#### 有效的安全等级

根据表 3-7 和表 3-8 内总结的 4 个双位位域决定有效的安全等级。对于双位采用以下短符号：1 表示激活，0 表示非激活，‘#’代表无效，‘-’代表不需考虑此值（包括无效状态）。

表 3-7 有效读保护

RPA	DCF	DDF	安全等级
0	-	-	无读保护
1   #	0	0	无读保护
	-	1   #	禁止读数据
	1   #	-	禁止读取指令

表 3-8 有效写保护

WPA	RPA	安全等级
0	-	无写保护
1   #	1   #	全局写保护
1   #	0	由 IMB_PROCON <sub>x</sub> 决定特定扇区的写保护

总结：

- 读保护始终影响整个 Flash 存储器地址范围。可分别控制取指和读数据。
- 当读保护有效，或者为每个逻辑扇区设置特定读保护时，可全局设定写保护。

最底层和中间安全层决定如何预置、改变和软件如何访问 4 个有效 IMB\_IMBCTR 位域。将在下面各段讨论这个问题。

### 初始化有效的安全等级

应用复位之后，保护被激活，因此 RPA、WPA、DDF 和 DCF 被置位。启动期间，由 IMB 内核决定页 3-41 “最底层“物理状态””所描述的安全等级并置位页 3-41 “中间层“Flash 状态””所描述的 IMB\_FSR\_PROT.PROIN 和 IMB\_FSR\_PROT.PROINER 以及 IMB\_PROCONx。根据表 3-6，IMB 内核进一步初始化 IMB\_IMBCTRL 的位域 RPA 和 WPA。

IMB\_IMBCTRL 中的位域 DDF 和 DCF 并不由 IMB 内核初始化，而是在系统启动期间，根据启动条件自动初始化。如果从 Flash 存储器开始取代码，那么这两个位域被设置为非激活状态。在所有其它情况下，它们被激活以防止未经密码验证就对 Flash 存储器进行读访问。

### 改变有效的安全等级

运行期间，可改变有效的安全等级。可通过设置寄存器 IMB\_IMBCTRL 直接改变安全等级；或通过命令“禁止写保护”改变中间层中的位间接改变安全等级；或通过改变安全页的内容间接改变安全等级（安全页改变中间层的位，进而影响安全等级）。

对 IMB\_IMBCTRL 直接进行写操作：

- 只有当 RPA 非激活时才能使 DCF 和 DDF 处于非激活状态。它们可始终被激活。

通过命令序列间接改变安全等级：

- 成功的“禁止读保护”置位 RPRODIS 并清除 RPA。
- 成功的“禁止写保护”置位 WPRODIS 并清除 WPA。
- 根据 PROIN、PROINER、RPRO 的状态，按照表 3-6 给出的规则，“重新使能读/写保护”清除 RPRODIS 和 WPRODIS 并置位 RPA 和 WPA。

通过改变安全页内容间接改变安全等级的过程如下：执行改变安全页内容的命令序列之后，IMB 内核立即读回安全页的内容并决定所有安全数据，如页 3-45 “初始化有效的安全等级”中系统启动的描述。页 3-47 “保护处理举例”中的示例给出如何设置或去除保护或改变密码。

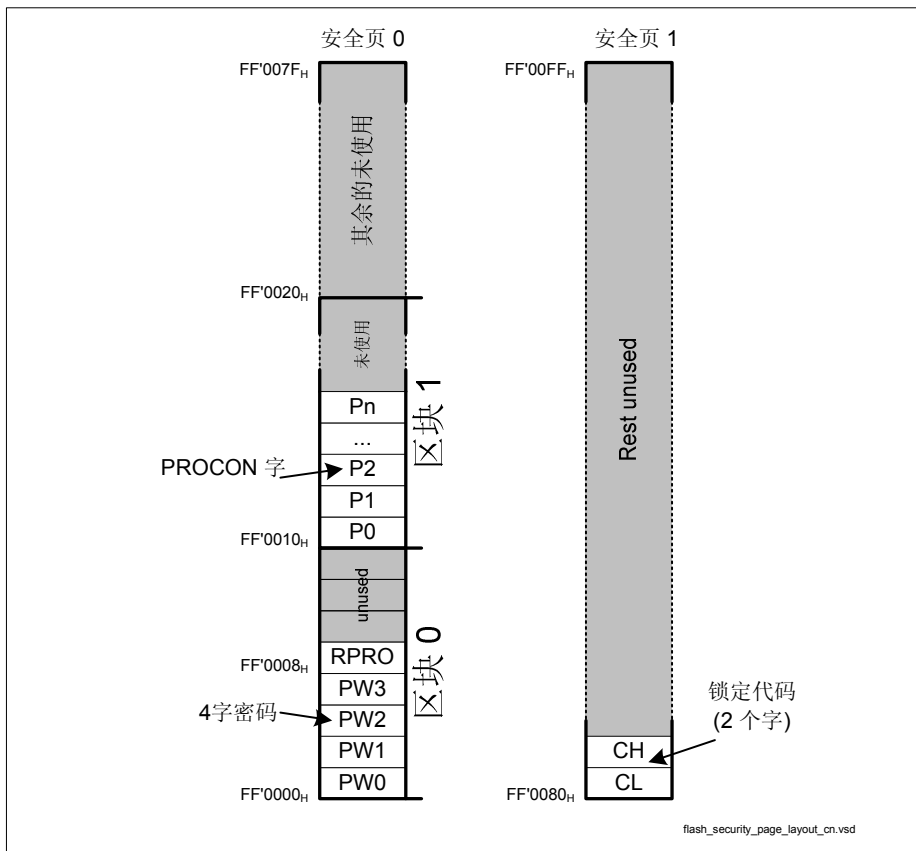
#### 3.10.8.4 违反保护情况下的反应

如果软件试图违反保护规则，则会发生下述情况：

- 读保护有效时，读取数据：位 IMB\_FSR\_PROT.PROER 被置位，如果 IMB\_INTCTR.DPROTRP 为 0，则由 SCU 触发 Flash 访问强制中断。读取操作给出缺省数据。
- 当读保护有效时，读取代码：给出强制中断码“TRAP 15<sub>D</sub>”
- 编程或擦除被写保护的存储器范围：PROER 被置位。

### 3.10.8.5 安全页面的结构

上面几节仅提到了安全页的内容，本节描述它们的详细结构。图 3-8 给出安全页 0 和 1 的结构图。



**图 3-8 安全页的结构**

在 XE16xyM 中，16 位字始终以 little endian 格式（低字节存放在地址低位）保存。

- PW<sub>x</sub> 字包含密码
- 双位 RPRO 保存在相关的 ISFR **IMB\_FSR\_PROT** 的位 15 和 14。该字中的其它位未使用、应保持全 0。

- 如 **IMB\_PROCON<sub>x</sub> (x = 0-3)** ISFR 中定义的那样保存 PROCON 数据。
- 锁定代码由两个字 CL 和 CH 组成，都包含“AA55<sub>H</sub>”以形成正确锁定代码。

安全页中使用的数据块 (SecP0 的数据块 0 和 1 以及 SecP1 的数据块 0) 的所有字节被认为“保留”，且必须保持擦除状态，即全 0 的内容。安全页中未使用的数据块 (SecP0 的数据块 2 到 7 以及 SecP1 的数据块 1 到 7) 应当被编程设置为全 1 数据。

### 3.10.9 保护处理举例

下面给出保护系统的使用示例。

#### 交付状态

器件交付时的状态为“未保护状态”。

安全页 1 被擦除 (因此它不包含“锁定代码”AA55AA55<sub>H</sub>)。

安全页 0 被擦除，因此“无效”。但是因为 SecP1 被擦除，因此不需要评估此数据。仅将其内容复制到相应的寄存器中。

根据启动模式，启动期间位 DDF 和 DCF 被置位。但是由于 RPA 和 WPA 为非激活，允许对 Flash 存储器进行任何访问。

Flash 存储器的数据扇区交付时也是擦除状态。所有扇区都可被编程。编程用户软件之后，用户可设置写和读保护。

#### 第一次设置密码

为了设置密码，通常 SecP1 中的锁定代码必须被擦除。第一次设置密码时，锁定代码不存在。

随后，为了能够改变 RPRO，必须使用“[擦除安全页面](#)”擦除 SecP0。擦除 SecP0 的操作清除 RPRO 至“00<sub>B</sub>”的无效状态。完成擦除命令之后，IMB 内核从 Flash 数据中恢复 IMB\_FSR\_PROT 和 IMB\_IMBCTRH 位域。

因为 SecP1 中不存在锁定代码，RPRO 的无效状态对用户可视的保护无影响。Flash 存储器的所有部分仍可写。

第二步是将需要的安全信息编程到 SecP0。IMB 内核再次立刻读回保存的数据并初始化安全系统。由于 SecP1 仍不包含锁定代码，器件仍然处于“未保护”模式。

用户软件不能直接读取安全页。因此只能间接验证编程到 SecP0 中的数据。通过读取 IMB\_PROCON 和 IMB\_FSR\_PROT 寄存器检查 RPRO 和 SnU 位域的数据。通过命令“[禁止读保护](#)”命令可验证密码。如果密码不匹配，则位 PROER 被置位。因为 SecP1 被擦除，Flash 存储器仍保持可写。因此，擦除出 SecP0 之后，可再次编程设置正确的密码。

SecP0 被成功验证之后，SecP1 被编程为锁定代码 AA55AA55<sub>H</sub>，用于使能 SecP0 的安全设置。

因为密码验证使得 RPRODIS 置位，必须使用命令“**重新使能读/写保护**”以最终激活新的保护。

### 改变密码或安全设置

改变密码是一个精细的操作。必须将两个安全页之间的相互关系牢记在心。

通常被保护状态下，SecP1 包含锁定代码。首先必须使用正确的密码禁止写保护。然后擦除 SecP1 中的锁定代码。如果该操作成功进行，IMB 内核将清除 PROIN。此时可安全擦除 SecP0。

从此时开始，安全页面处于工厂交付时的状态，可按照前面描述的内容设置新的密码和安全设置。

**注意：**安全页面可改变的次数在数据手册中给出。

### 3.10.10 EEPROM 仿真

XE16xyM 中的 Flash 有以下三种用途：

1. 保存程序代码。通常极少发生更新。需要满足的主要标准是产品生命周期内的数据保持特性。
2. 保存常量数据：该数据和程序代码一起保存。因此这些数据也极少被更新。此处 Flash 的耐受力特性不是问题，但是对数据保持特性的要求与代码存储器一致。
3. 运行期间更新数据：此时数据更新的频率非常高，如里程计数器或无匙进入系统的访问密钥。只有在故障情况下，其它数据可能被改变，且在系统掉电之前，其它数据可能仅被从 RAM 传送至非易失存储器。

尤其对于第三种类型的数据来说，非易失存储器需要类似 EEPROM 的特性：

- 精细的编程/擦除存储器区，EEPROM 中通常为一个字节。
- 比 Flash 单元固有的耐受力更高的耐受力特性
- 每个字节短的编程和擦除周期。尤其对于紧急情况下（如电源失效），需要在短的延迟内完成数据保存操作。

在运行过程中改变数据的基本要求是仍能恢复代码执行，尤其是中断请求必须仍被服务。由于四个 Flash 模块可独立工作，因此 XE16xyM 能够满足此要求。如果一个 Flash 模块正在编程或擦除，则可从另一个 Flash 模块中执行代码。

因为 XE16xyM 没有 EEPROM 模块，而 Flash 存储器具有频繁被提到的局限性（大的编程/擦除分区、中等的长编程/擦除周期、擦除/编程周期次数较大时，数据保持特性降低，Flash 单元耐受力特性有限、数据页未隔离，受到漏极干扰的影响），满足其它要求非常困难。

为了缓解运行期间保存数据时这些因素的影响，使用软件进行 EEPROM 仿真。有一些算法可有效将 Flash 存储器作为 EEPROM 使用。下面一节描述这些算法中（最简单的）的一个。

必须注意 XE16xyM 不为用户提供任何进行 EEPROM 仿真的硬件方式，所有下面的内容必须由软件来实现。

### **3.10.10.1 传统的 EEPROM 仿真**

解决耐受力有限问题的关键点是将数据存储在  $N$  个不同的物理区域。在 XE16xyM 中，算法使用  $N$  个连续页面或页面组。如果数据当前存储在页面组“ $x$ ”中，那么下一次数据将编程在页面组“ $(x+1) \bmod N$ ”中。

启动之后，必须找到上一次正确的页面组。通过评估计数器（从 0 到  $2 \cdot N - 1$ ）或者在编程新的页面之后，通过擦除页面使得旧的入口失效。另外，按页面组执行 CRC 检查。

所有涉及的页面被循环使用，从用户的角度来看，耐受力提高到了  $N$  倍。 $N$  的选择必须足够高，以满足耐受力和数据保持性的要求。因为在写入新的数据之前，它们最多受到  $N-1$  次干扰，因此对  $N$  个页的页面组来说，干扰不成问题。然而如果要求一个扇区适应具有不同更新行为的不同页面组，必须特别小心处理。在此情况下，一组页面的更新可能超过另一组页面的干扰极限。因此通常一个扇区应当仅能被一个此类的 EEPROM 循环缓存使用。

算法保持旧的数据，直到新数据通过验证，因此编程期间的电源故障仅能破坏最近的更新而旧的数据仍然可用。电源故障时，仍有一些问题需要进行特别处理：

- 编程期间电源被切断：接下来的引导加载可能会找到一个显然被正确编程的页面。然而，Flash 单元可能未被完全编程，因此具有较低的数据保持特性或读数据不稳定的（如工作条件的改变可能导致读错误）。

如果电源在早期被切断，尽管一些 Flash 单元被部分编程，页面可能表现为擦除状态。当向这些显然被擦除的页面编程不同数据时，可能出现读错误。

- 擦除期间电源被切断：和上面出现的情况相同。数据可能表现为被擦除，但数据保持特性降低。页面擦除期间的电源故障可禁止其所在物理扇区全部数据的可读性。因此仅在不包含任何当前数据的扇区内执行擦除操作的算法具有优势。

可通过一些措施改进算法：如始终编程两个页面，通过编程一个页面标记擦除过程的结束，使 EEPROM 仿真更加稳健。但是通常来说，中止 Flash 过程被当作禁止的“操作情况”。

上面描述的算法的主要不足之处在于，软件设计师需要完全安排 Flash 存储器的使用。用户必须选择正确的  $N$  值。然后所有的数据必须被分配给页面。共享一页的数据必须具有相似或最好一致的更新序列（否则，未改变的数据被无必要的写入）。如果一组数据未填充整个扇区，可用页必须尽可能保留为未使用，因为它们可能会招致过多的漏极干扰。

也有其它一些算法，通过监控 Flash 的使用，以及自适应的将数据自动分配给 Flash 单元来试图消除这些影响。

### 3.10.11 中断产生

当持续时间长的 Flash 操作（主要是：编程页面、擦除页面、擦除扇区和改变裕量，还包括进入页面模式）的请求被接受时，Flash 模块的 IMB\_FSR\_BUSY.BUSY 标志被置位，完成操作过程之后该标志被复位。需要软件查询忙标志以确定操作的结束。为了消除软件查询标志的负担，可产生一个中断。如果由 IMB\_INTCTRL.IEN 使能中断，当 IMB\_FSR\_BUSY.BUSY 标志之一从 1 跳变至 0 时，向 SCU 发出一个中断请求。在 SCU 中（见页 8-120 “SCU 中断产生”），该中断请求（称为“PFI”的程序 Flash 中断）和其它中断源复用后被送至四个中断节点之一。由寄存器位域 INTNP1.PF 选择中断节点。SCU 包含一组中断状态标志（INTSTAT），中断禁止控制（INTDIS）和用于置位（INTSET）和清除（INTCLR）该中断的控制位。

“**进入页面模式**”命令置位 BUSY 仅数个时钟周期，通常建议用户不使能该命令产生中断。

寄存器 **IMB\_INTCTR** 包含标志用于表示中断状态“ISR”的位域，中断请求的使能位“**IEN**”的和清除状态标志“**ICLR**”或置位状态标志“**ISET**”。应当注意，只有当 IEN 已经为 1 且 ISR 变为 1 的情况下，才能发送中断请求。当 ISR 已经为 1 而 IEN 变为 1 的情况以及 IEN 和 ISR 同时置 1 的情况下，都不发送中断。

### 3.10.12 关于优化 Flash 使用的建议

本节描述在特定应用场景下（例如，如何有效使用 ECC 和裕量读），使用 Flash 存储器的最佳实例。硬件特性的描述请参考页 3-36 “**数据完整性**”。

#### 3.10.12.1 编程代码和常量数据

在器件生命周期内，代码和常量数据仅被编程数次，例如 ECU 生产的下线编程或当执行更新服务时。由于该数据的可读性对产品质量起决定性作用，客户可能需要实现精细的“最佳实践”的建议。

#### 基本建议

Flash 操作期间始终确保正确的工作条件并防止电源故障。

防止处理错误的基本措施是，编程之后应当验证所有数据。一位 ECC 错误应被忽略。少量一位错误的出现是已知的物理效应的后果。



### 最佳实践

该方法提供了可能的、最好的编程质量但是存在编程步骤需要重复的风险，即使没有必要（“误报”）：

- 使用“擦除扇区”擦除整个扇区。
- 对扇区进行数据编程。防止软件崩溃的基本做法是用强制中断码填充扇区的未使用部分。
- 改变读电平至 **hard margin 0**。
- 验证编程的数据，注意比较错误和双位 **ECC** 错误并对单位 **ECC** 错误计数。小心评估每 128 位数据块的 **ECC** 错误标志，并在随后清除 **ECC** 标志。
- 使用 **hard margin 1** 重复该检查。
- 编程所有扇区之后：
  - 擦除并重新编程所有出现比较错误和双位 **ECC** 错误的扇区
  - 如果 **Flash** 模块包含多于特定个数（例如 10）的单位 **ECC** 错误，推荐擦除并重新编程受影响的扇区（如，那些包含至少一个单位错误的扇区）
  - 注意：数目较多的单位错误通常指示编程操作违反了操作条件。

为了减少误报的风险，可提高运行中更新所允许的单位 **ECC** 错误的门限。

### 3.10.12.2 EEPROM 仿真

EEPROM 仿真的主要目的通常不是使器件在整个使用寿命内可读，而是使 **Flash** 数据获得可能的、最高的稳健性（防止因违反操作条件、电源故障、甚至过度编程导致的 **Flash** 页面出错等原因造成 **Flash** 错误）。应当将 **Flash** 出错的风险降至最低。

采用下述方法可达到很好的稳健性：

- 编程之后用正常读电平验证数据。单位 **ECC** 错误应当被忽略。
- 当出现比较错误或双位 **ECC** 错误时，数据应被重新编程到下一个 **Flash** 区段（例如，下一个页面或扇区）。
- 应当限定重新编程实验的次数（如 3 次）以防违反操作条件。

显然，只有当算法不从固定地址上获得数据时，这种跳过出错页面的做法才能被最优的使用。

出错页面能够防止“擦除扇区”命令擦除受影响扇区的任何数据。不过“擦除页面”命令仍能够擦除所有其它页面，这些页面仍然可被读取和编程。

### 3.11 片上程序存储器控制

内部存储器块“IMB”包含所有被称为“片上存储器区域”的所有存储器，地址范围从 C0'0000<sub>H</sub> 到 FF'FFFF<sub>H</sub>，包括程序 SRAM、嵌入式 Flash 存储器和被称为“IMB 内核”的中央控制逻辑。

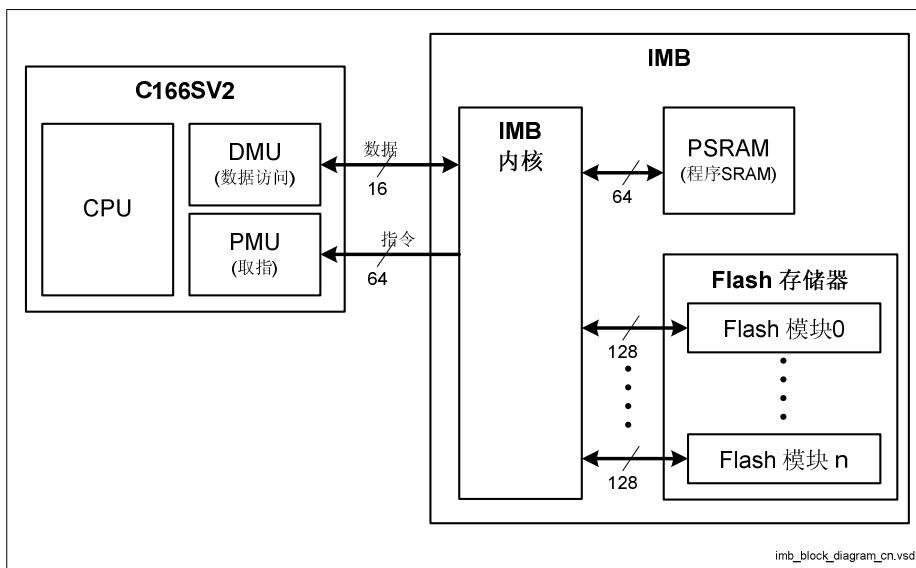
XE16xyM 器件中的 IMB 包含以下存储器：

- 四个独立模块内多达 832 KB Flash 存储器
- 多达 32 KB 程序 SRAM（见章节 3.4.1）

IMB 将这些存储器与 CPU 数据总线和取指总线连接起来，每个存储器都可以包含指令代码、数据或者两者的混合。IMB 管理对存储器的访问并支持 Flash 编程和擦除。

#### 3.11.1 概述

图 3-9 给出 IMB 及其存储器如何集成到器件架构中。图中仅包括主要的数据流。数据总线通常伴随着地址、控制信号和校验和数据（如奇偶校验位或 ECC）。



**图 3-9 IMB 模块框图**

CPU 有两条独立的总线。取指总线由 CPU 的程序管理单元“PMU”控制。以 64 位对齐的方式按组取指令。CPU 的取指单元预测跳转指令的结果，并根据预测的分支指

令提前读取指令。预测错误的情况下，该接口能够中止未完成请求并从继续从正确的分支处取指。由于 CPU 具有单周期内处理一个 32 位指令的性能，因此该接口决定 CPU 的性能。

数据总线由 CPU 的数据管理单元“DMU”控制。该单元以 16 位字读取数据。写访问也使用 16 位字，但是附加字节使得可改变单个字节。

因为 CPU 采用冯·诺依曼架构，数据和指令（以及“特殊功能寄存器”）共用同一个地址范围。当指令被作为数据使用时（如从 IO 接口复制代码到 PSRAM 中），可通过数据总线访问这些指令。CPU 的流水线行为使得取指和数据访问可同时被请求。如果这些操作寻址不同的存储器或 Flash 模块，由 IMB 控制这些同时执行的访问。

IMB 与中央系统控制单元还有一些附加的连接，这些连接关系未在框图中示出。

### 3.11.2 寄存器接口

[页 3-53 “IMB 寄存器”](#) 描述 IMB 的特殊功能寄存器。[页 3-67 “系统控制寄存器”](#) 描述影响 IMB、但不位于 IMB 地址范围内的特殊功能寄存器。

#### 3.11.2.1 IMB 寄存器

本节描述所有 IMB 特殊功能寄存器。

**表 3-9 寄存器概览**

寄存器缩写名	寄存器全名	偏移地址	页码
IMB_IMBCTRL	IMB 控制低位	FF FF00 <sub>H</sub>	<a href="#">页 3-54</a>
IMB_IMBCTRH	IMB 控制高位	FF FF02 <sub>H</sub>	<a href="#">页 3-56</a>
IMB_INTCTR	中断控制	FF FF04 <sub>H</sub>	<a href="#">页 3-57</a>
IMB_FSR_BUSY	Flash 状态忙	FF FF06 <sub>H</sub>	<a href="#">页 3-59</a>
IMB_FSR_OP	Flash 状态操作	FF FF08 <sub>H</sub>	<a href="#">页 3-60</a>
IMB_FSR_PROT	Flash 状态保护	FF FF0A <sub>H</sub>	<a href="#">页 3-61</a>
IMB_MAR0	裕量 0	FF FF0C <sub>H</sub>	<a href="#">页 3-63</a>
IMB_PROCON0	保护配置 0	FF FF10 <sub>H</sub>	<a href="#">页 3-64</a>
IMB_PROCON1	保护配置 1	FF FF12 <sub>H</sub>	<a href="#">页 3-64</a>
IMB_PROCON2	保护配置 2	FF FF14 <sub>H</sub>	<a href="#">页 3-64</a>

### 存储器结构

寄存器缩写名	寄存器全名	偏移地址	页码
IMB_PROCON3	保护配置 3	FF FF16 <sub>H</sub>	<a href="#">页 3-64</a>
IMB_ECC_TRAP	ECC 强制中断控制	FF FF20 <sub>H</sub>	<a href="#">页 3-65</a>
IMB_ECC_STAT	ECC 状态	FF FF22 <sub>H</sub>	<a href="#">页 3-66</a>

### IMB 控制

全局 IMB 控制。

应用复位后，IMB\_IMBCTRL 和 IMB\_IMBCTRH 被复位。

对这两个寄存器的写访问由寄存器安全机制控制，见 SCU 一章[页 8-176 “寄存器控制”](#)。请注意访问 IMB\_IMBCTR 之后，寄存器写保护不会被重新自动激活，该自动激活特性仅限于 SCU 内部寄存器。

### IMB\_IMBCTRL

#### IMB 控制低位

ISFR (FF FF00<sub>H</sub>)

复位值: 55AC<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DDF	DCF	-	-	-	-	-	-	-	-	-	DLC PF	WSFLASH			
rw	rw	-	-	-	-	-	-	-	-	-	rw	rw			

符号	位序号	类型	功能描述
WSFLASH	[2:0]	rw	<b>Flash 存储器访问的等待状态控制</b> 该位域定义等待周期个数，IMB 在该段时间过后从 Flash 存储器中读取数据： $N_{WS} = WSFLASH$ 该位域还决定 Flash 仿真地址范围内 PSRAM 的读时序。见 <a href="#">页 3-13 “Flash 仿真”</a> 。 该位域的设置和系统时钟频率有关，具体见器件数据手册。 <i>注：WSFLASH 一定不能为 0，该值被禁止！</i>
DLCPF	3	rw	<b>禁止线性代码预取</b> 0 <sub>B</sub> “高速模式”：当下次读请求的数据可从缓存中得到，因此 Flash 存储器将会空闲，IMB 内核自动增加直到最后一个地址，并在从 Flash

符号	位序号	类型	功能描述
			<p>存储器中读取下一个 128 位数据块。</p> <p>1<sub>B</sub> “低功耗模式”：该特性被禁止。</p> <p>对于要求功耗最小的代码或线性代码部分较短的代码来说，该特性应当被禁止（DLCPF = 1）。使能该特性仅对具有较长线性序列的代码有好处。具有较低的 WSFLASH 值时，DLCPF = 0 的性能增益减少。WSFLASH 较低的情况下，DLCPF = 1 禁止代码预取可能比使能线性代码预取情况下的性能更好。</p>
DCF	[13:12]	rw	<p><b>禁止从 Flash 存储器读取指令</b></p> <p>01<sub>B</sub> DCF = 1 的短表达。如果 RPA = 1，不能从 Flash 存储器取指令。如果 RPA = 0，该位域无影响。</p> <p>10<sub>B</sub> DCF = 0 的短表达。可从 Flash 存储器读取指令，与 RPA 的设置无关。</p> <p>00<sub>B</sub> 非法状态。</p> <p>11<sub>B</sub> 非法状态。两个非法状态的效果与设置“01”相同。只有通过应用复位才能退出该状态。</p> <p>当 RPA = 0 时，软件可任意修改该位域的值。否则只能禁止取指操作，在下次应用复位之前都不能使能取指操作。</p>
DDF	[15:14]	rw	<p><b>禁止从 Flash 存储器读取数据</b></p> <p>01<sub>B</sub> DDF = 1 的短表达。如果 RPA = 1，不能从 Flash 存储器读取数据。如果 RPA = 0，该位域无影响。</p> <p>10<sub>B</sub> DDF = 0 的短表达。可从 Flash 存储器读取数据，与 RPA 的设置无关。</p> <p>00<sub>B</sub> 非法状态。</p> <p>11<sub>B</sub> 非法状态。两个非法状态的效果与设置“01”相同。只有通过应用复位才能退出该状态。</p> <p>当 RPA = 0 时，软件可任意修改该位域的值。否则只能禁止数据读取操作，在下次应用复位之前都</p>

符号	位序号	类型	功能描述
			不能使能该操作。

IMB 控制高位字。WPA 和 RPA 位域描述见 [页 3-40 “保护处理详述”](#)。

## IMB\_IMBCTRH

**IMB 控制高位**

**ISFR (FF FF02<sub>H</sub>)**

**复位值: 0005<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PSPROT</b>							-	-	-	-	<b>RPA</b>		<b>WPA</b>		
rw							-	-	-	-	rh		rh		

符号	位序号	类型	功能描述
WPA	[1:0]	rh	<b>写保护激活</b> 01 <sub>B</sub> WPA = 1 的短表达。Flash 存储器的写保护被激活。 10 <sub>B</sub> WPA = 0 的短表达。Flash 存储器的写保护未激活。 00 <sub>B</sub> 非法状态。 11 <sub>B</sub> 非法状态。两个非法状态的效果与设置“01”相同。只有通过应用复位才能退出该状态。 只能由 IMB 内核改变该位域。软件写操作被忽略。
RPA	[3:2]	rh	<b>读保护激活</b> 01 <sub>B</sub> RPA = 1 的短表达。Flash 存储器的读保护被激活。 10 <sub>B</sub> RPA = 0 的短表达。Flash 存储器的读保护未激活。 00 <sub>B</sub> 非法状态。 11 <sub>B</sub> 非法状态。两个非法状态的效果与设置“01”相同。只能通过应用复位才能退出该状态。 只能由 IMB 内核改变该位域。软件写操作被忽略。

符号	位序号	类型	功能描述
PSPROT	[15:8]	rw	<b>PSRAM 写保护</b> 该 8 位位域决定 PSRAM 写保护范围地址。 可写范围的起始地址是 $E0'0000_H + 1000_H * PSPROT$ 。结束地址由所实现的存储器决定。具有 Flash 访问时序的 PSRAM 区域内的等效范围也被保护。此处可写范围从 $E8'0000_H + 1000_H * PSPROT$ 开始。 因此, $PSPROT = 00_H$ 时, 整个 PSRAM 均可写。

### 中断控制

中断控制和状态。

由应用复位进行复位。

### IMB\_INTCTR

#### 中断控制

**ISFR (FF FF04<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISR	PSE R	-	-	-	PSE RCLR	ISSET	ICLR	-	-	-	-	DPR OTR P	DDD TRP	DIDT RP	IEN
rh	rh	-	-	-	w	w	w	-	-	-	-	rw	rw	rw	rw

符号	位序号	类型	功能描述
IEN	0	rw	<b>中断使能</b> 如果置位, 当 ISR 置位时, IMB 的中断信号变为激活状态。
DIDTRP	1	rw	<b>禁止取指双位错误强制中断</b> 如果置位, 双位 ECC 错误不会导致由强制中断指令取代所读取的数据。 参见 <a href="#">IMB_ECC_TRAP.DITRP<sub>x</sub></a> 。
DDDTRP	2	rw	<b>禁止数据读取双位错误强制中断</b> 如果置位, 读取数据期间的双位 ECC 错误不会向 SCU 发送“Flash 访问错误”请求, 即不会产生硬

符号	位序号	类型	功能描述
			件强制中断，读取数据不会被缺省数据代替。 IMB_FSR_PROT 和 IMB_ECC_STAT 内的错误标志仍置位。 参见 <a href="#">IMB_ECC_TRAP.DDTRPx</a> 。
DPROTRP	3	rw	<b>禁止保护强制中断</b> 如果置位，对读保护 Flash 存储器的读请求不会产生发送给 SCU 的“Flash 访问错误”，即不产生硬件强制中断。
ICLR	8	w	<b>中断清除</b> 向该位域写 1 时，ISR 被清除。读取该位始终返回 0。向该位写 0 的操作被忽略。
ISSET	9	w	<b>中断置位</b> 向该位写 1 时，ISR 置位。若 IEN 被置位，中断信号被激活。读取该位始终返回 0。向该位写 0 的操作被忽略。
PSERCLR	10	w	<b>清除 PSRAM 错误标志</b> 向该位写 1 时，PSER 被清除。读取该位始终返回 0。向该位写 0 的操作被忽略。
PSER	14	rh	<b>PSRAM 错误标志</b> 当检测到发送至被写保护或未实现的 PSRAM 范围的写请求时，该标志被置位。向 PSERCLR 写 1 则该标志被清除。
ISR	15	rh	<b>中断服务请求</b> 如果置位，指示至少有一个 IMB_FSR_BUSY.BUSY 从 1 变为 0。若 IEN 被置位，向中断控制器发送中断请求。中断被处理后，软件处理器向 ICLR 写 1 清除该标志。

## Flash 状态

Flash 状态。分为三个寄存器 IMB\_FSR\_BUSY、IMB\_FSR\_OP 和 IMB\_FSR\_PROT。IMB\_FSR\_PROT 的相关保护位域的描述见[页 3-40 “保护处理详述”](#)。



### 存储器结构

寄存器可由应用复位进行复位，“ERASE”、“PROG”和“OPER”除外，这三个位域只能由上电复位进行复位。

### IMB\_FSR\_BUSY

**Flash 状态忙**

**ISFR (FF FF06<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	PAG E3	PAG E2	PAG E1	PAG E0	-	-	-	-	BUS Y3	BUS Y2	BUS Y1	BUS Y0
-	-	-	-	rh	rh	rh	rh	-	-	-	-	rh	rh	rh	rh

符号	位序号	类型	功能描述
BUSY0	0	rh	<b>Flash 0 忙标志</b> Flash 模块 0 正忙。由 IMB_FSR_OP 中的 MAR、POWER、ERASE 或 PROG 指示任务类型。任务完成之后，BUSY0 被自动清零。为了使中断处理器能够确定所完成任务，相应的任务类型指示不会被清除。
BUSY1	1	rh	<b>Flash 1 忙标志</b> Flash 模块 1 的忙标志，描述与 BUSY0 相同。
BUSY2	2	rh	<b>Flash 2 忙标志</b> Flash 模块 2 的忙标志，描述与 BUSY0 相同。
BUSY3	3	rh	<b>Flash 3 忙标志</b> Flash 模块 3 的忙标志，描述与 BUSY0 相同。
PAGE0	8	rh	<b>Flash 0 页面模式指示</b> 只要 Flash 模块 0 处于页面模式，则该位置位。由“ <a href="#">进入页面模式</a> ”命令进入页面模式，由“ <a href="#">编程页面</a> ”命令退出页面模式。也可通过“ <a href="#">复位到读</a> ”命令退出页面模式。应用复位也可清除该位。
PAGE1	9	rh	<b>Flash 1 页面模式指示</b> Flash 模块 1 的页面模式指示，描述与 PAGE0 相同。
PAGE2	10	rh	<b>Flash 2 页面模式指示</b> Flash 模块 2 的页面模式指示，描述与 PAGE0 相

符号	位序号	类型	功能描述
			同。
PAGE3	11	rh	<b>Flash 3 页面模式指示</b> Flash 模块 3 的页面模式指示，描述与 PAGE0 相同。

## IMB\_FSR\_OP

### Flash 状态操作

### ISFR (FF FF08<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	OPE R	SQE R	MAR	POW ER	ERA SE	PRO G
-	-	-	-	-	-	-	-	-	-	rh	rh	rh	rh	rh	rh

符号	位序号	类型	功能描述
PROG	0	rh	<b>编程任务指示</b> 启动编程任务时，该位被置位。由相应的 BUSY 位指示进行编程操作的 Flash 模块。PROG 位不会被自动复位，必须由“清除状态”命令清除。该位不会被应用复位清除，只能由上电复位清除。
ERASE	1	rh	<b>擦除任务指示</b> 启动擦除任务时，该位被置位。由相应的 BUSY 位指示进行擦除操作的 Flash 模块。ERASE 位不会被自动复位，必须由“清除状态”命令清除。该位不会被应用复位清除，只能由上电复位清除。
POWER	2	rh	<b>电源改变指示</b> 该位指示 Flash 模块处于启动阶段或关闭阶段。BUSY 标志指示哪个 Flash 模块正忙。该位不会被自动复位，必须由“清除状态”命令复位。
MAR	3	rh	<b>裕量改变指示</b> 如果请求修改读裕量，该位和相应的 BUSY 标志一起置位。当裕量改变生效时，BUSY 标志被清除，此时可再次读取 Flash 模块。MAR 位必须由“清除状态”命令清除。

符号	位序号	类型	功能描述
SQER	4	rh	<b>序列错误</b> 出现错误的命令序列或命令不被接受时，该位被置位。由“清除状态”和“复位到读”命令清除该位。
OPER	5	rh	<b>操作错误</b> IMB 内核保持那些在启动编程或擦除过程时被置位的位。当操作过程完成时，这些位被清除。这些位不会被应用程序复位，而只能被上电复位复位。如果应用复位之后其中一位置位，IMB 内核置位 OPER。因此使用该位指示运行中的擦除或编程操作被应用复位中断。 OPER 由“复位到读”，“清除状态”或上电复位清除。

### IMB\_FSR\_PROT

#### Flash 状态保护

#### ISFR (FF FF0A<sub>H</sub>)

复位值: x000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPRO	-	-	DDB ER	DSB ER	IDBE R	ISBE R	-	-	-	PRO ER	WPR ODIS	RPR ODIS	PROI NER	PROI N	
rh	-	-	rh	rh	rh	rh	-	-	-	rh	rh	rh	rh	rh	

符号	位序号	类型	功能描述
PROIN	0	rh	<b>Flash 保护设置</b> 由 IMB 内核修改该位。由应用复位清除该位。
PROINER	1	rh	<b>Flash 保护设置错误</b> 由 IMB 内核修改该位。由应用复位清除该位。
RPRODIS	2	rh	<b>读保护被禁止</b> 使用“禁止读保护”命令暂时禁止读保护。由 IMB 内核修改该位。由应用复位清除该位。
WPRODIS	3	rh	<b>写保护被禁止</b> 使用“禁止写保护”命令暂时禁止写保护。由 IMB 内核修改该位。由应用复位清除该位。

符号	位序号	类型	功能描述
PROER	4	rh	<b>保护错误</b> 发生违反保护设置的情况时，该位被置位。由“ <b>清除状态</b> ”和“ <b>复位到读</b> ”命令或应用复位清除该位。
ISBER	8	rh	<b>取指单位错误</b> 若在取指期间检测到单位 ECC 错误（并被纠正），该位置位。由“ <b>清除状态</b> ”或“ <b>复位到读</b> ”命令或应用复位清除该位。
IDBER	9	rh	<b>取指双位错误</b> 若在取指期间检测到双位 ECC 错误（未被纠正），该位置位。由“ <b>清除状态</b> ”或“ <b>复位到读</b> ”命令或应用复位清除该位。
DSBER	10	rh	<b>数据读取单位错误</b> 读取数据时检测到单位 ECC 错误的情况，描述与 ISBER 相同。
DDBER	11	rh	<b>数据读取双位错误</b> 读取数据时检测到双位 ECC 错误的情况，描述与 IDBER 相同。
RPRO	[15:14]	rh	<b>读保护配置</b> 由 IMB 内核从安全页面 0 的相应位域复制到该位域。应用复位之后，读保护被激活。 该位域和其他保护位域的解释见表 3-6。 00 <sub>B</sub> 无效 01 <sub>B</sub> 激活 10 <sub>B</sub> 非激活 11 <sub>B</sub> 无效

## 裕量控制

读裕量控制。每个位域对应一个 Flash 模块。Hard read 0 检测未被彻底擦除的单元，读这些位返回 1。Hard read 1 检测未被彻底编程的单元，读这些位返回 0。由命令序列“**改变读裕量**”来改变读裕量。所得的读裕量反映在该状态寄存器中。

命令序列“**编程页面**”，“**擦除扇区**”，“**擦除页面**”和“**擦除安全页**”将读裕量复位回“正常”。Flash 唤醒时，也将读裕量恢复到“正常”。

该寄存器可被应用复位复位。

## IMB\_MAR0

### 裕量控制 0

ISFR (FF FF0C<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
HREAD0	[2:0]	rh	<b>Hard read 0</b> Flash 模块 0 的有效读裕量。 “000”：正常读 “001”：Hard read 0 “010”：alternate hard read 0（通常比 001 更严格） “101”：hard read 1 “110”：alternate hard read 1（通常比 101 更严格） 其它组合：保留。
HREAD1	[5:3]	rh	<b>Hard read 1</b> Flash 模块 1 的有效读裕量，描述同 HREAD0。
HREAD2	[8:6]	rh	<b>Hard read 2</b> Flash 模块 2 的有效读裕量，描述同 HREAD0。
HREAD3	[11:9]	rh	<b>Hard read 3</b> Flash 模块 3 的有效读裕量，描述同 HREAD0。

### 保护配置

为每个 Flash 模块提供一个保护配置寄存器。逻辑扇区编号如图 3-6 所示。

PROCONx 中的每位对应一个逻辑扇区。如果某位被清除，则相应逻辑扇区的写访问（这意味着访问物理扇区范围）被锁定，扇区锁定的条件见页 3-40 “保护处理详述”。PROCON 寄存器由 IMB 内核修改，IMB 内核从安全页面 0 复制这些寄存器的内容。

对于小于 256 KB 的 Flash 模块，未实现的 Flash 范围对应的 SsU 位被保留，应当在安全页面中将其编程为 0。

由应用复位复位该寄存器。

### IMB\_PROCONx (x = 0-3)

#### 保护配置

ISFR (FF FF10<sub>H</sub>+2\*x)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	S12 U	S11 U	S10 U	S9U	S8U	S7U	S6U	S5U	S4U	S3U	S2U	S1U	S0U
-	-	-	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

符号	位序号	类型	功能描述
SsU (s = 0-12)	s	rh	<b>扇区 s 未锁定</b> 0 <sub>B</sub> : Flash 模块 x 的逻辑扇区 s 被写保护 1 <sub>B</sub> : Flash 模块 x 的逻辑扇区 s 未被写保护 <i>注: 在先前的系列器件和基于 TriCore™ 的产品中, 这些位是“锁定位”而不是“未锁定”位。</i>

### ECC 强制中断控制

ECC 强制中断控制寄存器。

由应用复位复位。

寄存器 IMB\_ECC\_TRAP 用来禁止选中 Flash 模块的双位 ECC 错误强制中断的产生，而 IMB\_INTCTR 仅用来全局控制该强制中断。可选的控制使能可将部分 Flash 存储器当作准 ROM（带有错误强制中断）进行操作。但是当 Flash 模块被编程或擦除时，其强制中断的产生被关闭，而不影响“ROM”模块。没有此项特性的话，强制中断必须被全局禁止，Flash 驱动器必须从 SRAM 开始工作，且所有中断都被阻滞。

## IMB\_ECC\_TRAP

### ECC 强制中断控制

### ISFR (FF FF20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	DDT RP3	DDT RP2	DDT RP1	DDT RP0	0	0	0	0	DITR P3	DITR P2	DITR P1	DITR P0
r	r	r	r	rw	rw	rw	rw	r	r	r	r	rw	rw	rw	rw

符号	位序号	类型	功能描述
DITRP0	0	rw	<b>禁止指令读取双位 ECC 强制中断 0</b> 0 <sub>B</sub> : 当从 Flash 模块 0 读取指令出现双位 ECC 错误时, 用强制中断码代替指令。由 IMB_INTCTR.DIDTRP 进行全局处理。 1 <sub>B</sub> : 如置位, 从 Flash 模块 0 读取指令时, 双位 ECC 错误不会导致读取的数据被强制中断码代替。与 IMB_INTCTR.DIDTRP 无关。 另外 IMB_FSR_PROT.ISBER/IDBER 不会因为 Flash 模块 0 的 ECC 错误而置位。
DITRP1	1	rw	<b>禁止指令读取双位 ECC 强制中断 1</b> 对应于 Flash 模块 1, 描述见 DITRP0
DITRP2	2	rw	<b>禁止指令读取双位 ECC 强制中断 2</b> 对应于 Flash 模块 2, 描述见 DITRP0
DITRP3	3	rw	<b>禁止指令读取双位 ECC 强制中断 3</b> 对应于 Flash 模块 3, 描述见 DITRP0
DDTRP0	8	rw	<b>禁止数据读取双位 ECC 强制中断 0</b> 0 <sub>B</sub> : 由 IMB_INTCTR.DDDTRP 全局处理从 Flash 模块 0 读取数据时出现的双位 ECC 错误强制中断。 1 <sub>B</sub> : 从 Flash 模块 0 读取数据时出现双位 ECC 错误不会触发“Flash 访问错误强制中断”, 与 IMB_INTCTR.DDDTRP 无关。此外, IMB_FSR_PROT.DSBER/DSBER 不会因为 Flash 模块 0 的 ECC 错误而置位, 且 Flash 存储器的给出数据不是缺省数据。但是 ECC 错误仍会置位 IMB_ECC_STAT.xBERx。

符号	位序号	类型	功能描述
DDTRP1	9	rw	<b>禁止数据读取双位 ECC 强制中断 1</b> 对应于 Flash 模块 1，描述见 DDTRP0
DDTRP2	10	rw	<b>禁止数据读取双位 ECC 强制中断 2</b> 对应于 Flash 模块 2，描述见 DDTRP0
DDTRP3	11	rw	<b>禁止数据读取双位 ECC 强制中断 3</b> 对应于 Flash 模块 3，描述见 DDTRP0

### ECC 状态

ECC 状态寄存器。

可由应用复位复位。

该寄存器可选择报告每个 Flash 模块的 ECC 数据读取单位和双位错误。可独立清除每一位。该特性使得可使用部分 Flash 存储器作为准“ROM”使用。在这部分，所有错误触发的强制中断都由强制中断处理器进行处理，典型的处理方式是触发应用复位。然而当 Flash 模块被编程或擦除时，所有 ECC 错误都由一个底层驱动器处理，而无需影响整个系统。

### IMB\_ECC\_STAT

#### ECC 状态

ISFR (FF FF22<sub>H</sub>)

复位值：0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	DBE R3	DBE R2	DBE R1	DBE R0	0	0	0	0	SBE R3	SBE R2	SBE R1	SBE R0
r	r	r	r	rwh	rwh	rwh	rwh	r	r	r	r	rwh	rwh	rwh	rwh

符号	位序号	类型	功能描述
SBER0	0	rwh	<b>数据读取单位错误 0</b> 当从 Flash 模块 0 读取数据时，出现单位 ECC 错误时，该位置位。 通过应用复位或向该位写 1 的方式可清除该位。
SBER1	1	rwh	<b>数据读取单位错误 1</b> 对应于 Flash 模块 1，描述见 SBER0。
SBER2	2	rwh	<b>数据读取单位错误 2</b> 对应于 Flash 模块 2，描述见 SBER0。



符号	位序号	类型	功能描述
SBER3	3	rwh	<b>数据读取单位错误 3</b> 对应于 Flash 模块 3，描述见 SBER0。
DBER0	8	rwh	<b>数据读取双位错误 0</b> 当从 Flash 模块 0 读取数据时，出现双位 ECC 错误时，该位置位。 通过应用复位或向该位写 1 的方式可清除该位。
DBER1	9	rwh	<b>数据读取双位错误 1</b> 对应于 Flash 模块 1，描述见 DBER0。
DBER2	10	rwh	<b>数据读取双位错误 2</b> 对应于 Flash 模块 2，描述见 DBER0。
DBER3	11	rwh	<b>数据读取双位错误 3</b> 对应于 Flash 模块 3，描述见 DBER0。

### 3.11.2.2 系统控制寄存器

这些寄存器用于唤醒和关闭部分存储器子系统。

表 3-10 寄存器的地址空间

模块	基址	结束地址	注
SCU	0000 <sub>H</sub>	0FFF <sub>H</sub>	SCU 模块

表 3-11 寄存器概览

寄存器缩写名	寄存器全名	偏移地址	页码
MEM_KSCCFG	存储器内核控制	F012 <sub>H</sub>	<a href="#">页 3-68</a>
FL_KSCCFG	Flash 内核控制	FE22 <sub>H</sub>	<a href="#">页 3-69</a>

### 存储器内核配置

该寄存器控制处理器子系统单元 DMU、PMU、IMB 和 EBC 的关闭请求。该寄存器的结构和其它 KSCCFG 寄存器一致，但是只有 COMCFG 可以使用。该位域可使用两个设置值：00<sub>B</sub> 代表“时钟关闭模式”不会触发处理器子系统的关闭。只有在 DMP\_1 的系统时钟未被禁止（“时钟关闭模式”）的情况下才可以使用该设置。

另一个可用的值为 01<sub>B</sub>，当通过禁止 DMP\_1 的系统时钟而使得“时钟关闭模式”出现的所有情况下，必须使用该设置值。

可由应用复位来复位该寄存器。**注意：**COMCFG 的复位值是 00<sub>B</sub>。

由寄存器安全机制控制对该寄存器的访问（“Sec”类型）。

### MEM\_KSCCFG

存储器内核状态配置寄存器

ESFR (F012<sub>H</sub>/06<sub>H</sub>)

复位值: 0001<sub>H</sub>

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	-		COMCFG	-	-		-	-	-	-	-	-	-	-	-	1
	w	-	rw	-	-		-	-	-	-	-	-	-	-	-	rw

符号	位序号	类型	功能描述
<b>1</b>	0	rw	必须写入 1。
<b>COMCFG</b>	[13:12]	rw	<b>时钟关闭模式配置</b> 该位域定义时钟关闭模式下的关闭请求是否被激活。 如果 COMCFG[13]为 1，时钟关闭模式下的关闭请求被激活（即 CR = 10）。 COMCFG[12]不起作用。
<b>BPCOM</b>	15	w	<b>COMCFG 的位保护</b> 该位使能对 COMCFG 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 COMCFG 不改变 1 由写入值更新 COMCFG。

## Flash 内核配置

该寄存器控制 Flash 模块的电源关闭请求。当工作电压不足时，配置该寄存器需要特别注意，不要使 Flash 模块掉电。在此情况下，所有 CFG 位域应当包含 10<sub>B</sub>。

由寄存器安全机制控制对该寄存器的访问（“Sec”类型）。

## FL\_KSCCFG

**Flash 内核状态配置寄存器**

**SFR (FE22<sub>H</sub>/11<sub>H</sub>)**

**复位值: 0001<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	-	COMCFG	BP SUM	-	SUMCFG	BP NOM	-	NOMCFG	-	BP MOD EN	MOD EN				
w	-	rw	w	-	rw	w	-	rw	-	w	rw				

符号	位序号	类型	功能描述
<b>MODEN</b>	0	rw	<b>模块使能</b> 该位可直接置位电源关闭请求。 0 激活电源关闭请求。 1 该位域无效。
<b>BPMODEN</b>	1	w	<b>MODEN 的位保护</b> 该位使能对 MODEN 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 MODEN 不改变 1 由写入值更新 MODEN。
<b>NOMCFG</b>	[5:4]	rw	<b>正常操作模式配置</b> 正常操作模式下，由该位域定义是否激活掉电请求。 如果 NOMCFG[5]为 1，正常工作模式下激活掉电请求（即 CR = 00 或 01） NOMCFG[4]无效。
<b>BPNO</b>	7	w	<b>NOMCFG 的位保护</b> 该位使能对 NOMCFG 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 NOMCFG 不改变

符号	位序号	类型	功能描述
			1 由写入值更新 NOMCFG。
<b>SUMCFG</b>	[9:8]	rw	<b>挂起模式配置</b> 该位域定义挂起模式下是否激活掉电请求（如果正常工作模式下也激活掉电请求，这才有意义）。 如果 SUMCFG[9]为 1，关闭模式下激活掉电请求（即 CR = 01）。 SUMCFG[8]无效。
<b>BPSUM</b>	11	w	<b>SUMCFG 的位保护</b> 该位使能对 SUMCFG 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 SUMCFG 不改变 1 由写入值更新 SUMCFG。
<b>COMCFG</b>	[13:12]	rw	<b>时钟关闭模式配置</b> 该位域定义时钟关闭模式下是否激活掉电请求。 如果 COMCFG[13]为 1，时钟关闭模式下的掉电请求被激活（即 CR = 10）。 COMCFG[12]不起作用。
<b>BPCOM</b>	15	w	<b>COMCFG 的位保护</b> 该位使能对 COMCFG 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 COMCFG 不改变。 1 由写入值更新 COMCFG。

### 3.11.3 启动，关闭

存储器和处理器子系统的启动和关闭的描述请参考编程指南，该手册同时还描述如何使用内核控制寄存器。

### 3.11.4 错误报告总结

**表 3-12** 给出检测到的错误类型以及可能的响应。

**表 3-12 IMB 错误报告**

错误	响应
从 PSRAM 读取数据时出现的奇偶校验错误	如果 PECON.PEENPS: 硬件强制中断（见 <b>章节 8.14.2</b> ）
从 PSRAM 读取指令时出现的奇偶校验错误	如果 PECON.PEENPS: 硬件强制中断（见 <b>章节 8.14.2</b> ）
从 Flash 存储器读取数据时出现单位错误	自行纠正。IMB_FSR_PROT.DSBER 置位。
从 Flash 存储器读取数据时出现双位错误	位 IMB_FSR_PROT.DDBER 置位。如果 IMB_INTCTR.DDDTRP = 0: 产生 Flash 访问强制中断 <sup>1)</sup> 并给出缺省数据。
从 Flash 存储器取指时出现单位错误	自行纠正。IMB_FSR_PROT.ISBER 置位。
从 Flash 存储器取指时出现双位错误	位 IMB_FSR_PROT.IDBER 置位。如果 IMB_INTCTR.DIDTRP = 0: 送出“TRAP 15 <sub>D</sub> ”取代被破坏的数据。
从受保护的 Flash 存储器读数据	IMB_FSR_PROT.PROER 置位。 如果 IMB_INTCTR.DPROTRP = 0: 产生 Flash 访问强制中断 <sup>1)</sup> 并给出缺省数据。
从受保护的 Flash 存储器取指令	给出“TRAP 15 <sub>D</sub> ”。
目标地址在写保护 Flash 范围内的编程/擦除请求	仅 IMB_FSR_PROT 中的 PROER 被置位。
从正忙的 Flash 存储器读数据或取指令	读访问停滞，直到 Flash 模块结束忙状态。
从 ISFR 地址处取指令	给出缺省数据（“TRAP 15 <sub>D</sub> ”）。
从未实现的 ISFR 地址读数据	给出缺省数据。
向未实现的 ISFR 地址写数据	自行忽略。
从未实现的地址范围读取数据	不可预测。可能返回来自其它存储器的镜像

错误	响应
	数据或缺省值。
从未实现的地址范围取指令	不可预测。可能返回来自其它存储器的镜像数据或缺省值。
尚未实现的 PSRAM 或写保护 PSRAM 地址范围写数据（都由 IMB_IMBCTRH.PSPROT 决定）	位 IMB_INTCTR.PSER 置位。 产生 Flash 访问强制中断 <sup>1)</sup> 不改变 PSRAM 中的任何数据。
编程或擦除命令的目标地址不是已经实现的 Flash 存储器	不可预测。访问被忽略 <sup>2)</sup> 或镜像至已经实现的 Flash 存储器 <sup>3)</sup> 。
从掉电的 Flash 模块读数据	被认为是访问未实现的 Flash 范围。将返回缺省数据或已实现 Flash 模块的数据。
从掉电的 Flash 模块取指令	被认为是访问未实现的存储器范围。将返回缺省数据（“TRAP 15 <sub>D</sub> ”）或已实现 Flash 模块的数据。
目标地址为掉电 Flash 模块的编程或擦除命令	自行忽略 <sup>2)</sup>
当命令序列解读器等待命令序列的最后一个字时，接收到关闭或掉电请求	命令解读器被复位。执行“ <b>复位到读</b> ”命令序列。

1) 有关 Flash 访问强制中断的详细描述，请参见“SCU”一章。

2) 注意：当访问（如 MOV）被忽略时，命令序列解读器仍将等待这个未来到的 MOV 指令。因为下一个命令序列可能给出不期望的 MOV 指令，从而导致 SQER。

3) 不能通过访问保留地址范围，而使得 Flash 保护被旁路。

### 3.12 数据保持存储器

本节描述如何使用特殊用途数据存储：备用 RAM（SBRAM）和标记存储器（MKMEM）。取决于器件型号，仅部分器件具有这些特殊功能存储器。XE16xyM 包含：

- SBRAM
- MKMEM

这两个存储器由唤醒电源域（DMP\_M）供电，当系统电源域（DMP\_1）关闭时，这两个存储器中的数据仍可保持。

### 3.12.1 备用 RAM 访问

SBRAM 未映射到处理器的地址范围。通过四个 SFR SBRAM\_WADD, SBRAM\_RADD, SBRAM\_DATA0 和 SBRAM\_DATA1 实现对该存储器的访问。可采用下述方式进行访问：

- 写操作，不自动递增写地址指针：  
必须由软件首先向 WADD 写入目标地址，然后将数据写入到 DATA0。如果（至少）DATA0 的低位字节被写入，则写入到 DATA0 的数据被传送到所指示的 SBRAM 地址。如果 DATA0 再次被写入，则 SBRAM 中的相同地址用于存储数据。写访问 DATA0 之后，位 WADD.MOD 被清除。
- 写操作，自动递增写地址指针：  
必须由软件将首个目标地址写入到 WADD，然后可逐字将数据写入到 DATA1。如果（至少）SRDR1 的低位字节被写入，则写入到 DATA1 的数据被传送到所指示的 SBRAM 地址。数据存储到 SBRAM 的同时，写地址指针 WADD.WPTR 自动加 1（一个字），用来指示下次数据保存的目标地址。达到最大值时，地址指针自动回绕，在此情况下，WADD.WA 置位。写访问 DATA1 的操作置位 WADD.MOD。
- 读操作，不自动递增读地址指针：  
必须由软件将目标地址写入到 RADD 中，然后从 DATA0 处读取数据。如果再次读取 DATA0，则 SBRAM 中的相同地址的内容被读取。读访问 DATA0 的操作置位 RADD.MOD。
- 读操作，自动递增读地址指针：  
必须由软件将首个目标地址写入到 RADD，然后从寄存器 DATA1 逐字读取数据。从 SBRAM 读取数据的同时，读地址指针 RADD.RPTR 自动加 1（一个字），得到下次读取数据的目标地址。地址指针达到最大值之后，自动进行回绕，由位 RADD.WA 指示发生了回绕。读访问 DATA1 的操作置位 RADD.MOD。

自动递增访问的特性支持进行连续的数据读/写操作。

*注：因为读访问 SBRAM\_DATA0 和 SBRAM\_DATA1 返回的是最近一次更新寄存器 SBRAM\_RADD 的操作所预先读取的值。只能在 SBRAM\_RADD 已经被更新（或者通过写入的方式显性更新，或者通过自动递增的功能隐性更新）为同一个地址之后，写入到地址@SBRAM\_RADD 的任意数据才能被读回。通常从写访问切换至读访问 SBRAM\_RADD 时，在读取 SBRAM\_DATAx 之前，应当再次写入 SBRAM\_RADD。*

*注：因为该预读和自动递增特性，为了防止因预读操作而引起奇偶/ECC 错误，在最后一次数据访问之后，始终初始化地址的做法就非常重要。*

### 3.12.2 备用 RAM 寄存器

本节详细描述 SBRAM 寄存器接口。

**表 3-13 寄存器概览**

寄存器缩写名	寄存器全名	偏移地址	页码
SBRAM_RADD	SBRAM 读地址	FEDC <sub>H</sub>	<a href="#">页 3-74</a>
SBRAM_WADD	SBRAM 写地址	FEDE <sub>H</sub>	<a href="#">页 3-75</a>
SBRAM_DATA0	SBRAM 数据寄存器 0	FEE0 <sub>H</sub>	<a href="#">页 3-76</a>
SBRAM_DATA1	SBRAM 数据寄存器 1	FEE2 <sub>H</sub>	<a href="#">页 3-77</a>

#### 3.12.2.1 SBRAM 读地址寄存器

该寄存器定义要进行读操作的字地址。

由上电复位复位该寄存器。

#### SBRAM\_RADD

##### SBRAM 读地址寄存器

**SFR (FEDC<sub>H</sub>/6E<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD</b>	<b>WA</b>	<b>0</b>	<b>RPTR</b>												<b>0</b>
rwh	rwh	r	rwh												r

符号	位序号	类型	功能描述
<b>RPTR</b>	[12:1]	rwh	<b>读指针</b> 选择 SBRAM 中的字地址，从该地址处读取数据。 读取寄存器 DATA1 时，该指针自动加 1（即，指向下一个字地址）。
<b>WA</b>	14	rwh	<b>回绕</b> 该位指示读指针 RPTR 是否因为自动地址增加而出现了回绕。 0 未发生地址回绕 1 已经检测到地址回绕，必须由软件清除该位。



符号	位序号	类型	功能描述
<b>MOD</b>	15	rwh	<b>修改</b> 该位指示是否最近的 <b>SBRAM</b> 读访问使得 <b>RPTR</b> 自动加 1。  0 最近的数据读访问指向 <b>DATA0</b> ，不自动修改 <b>RPTR</b> 指针。  1 最近的数据读访问指向 <b>DATA1</b> ， <b>RPTR</b> 自动加 1。
<b>0</b>	0, 13	r	<b>保留</b> 读操作返回 0；应写入 0。

### 3.12.2.2 SBRAM 写地址寄存器

该寄存器定义要写入的字地址。

由上电复位复位该寄存器。

#### SBRAM\_WADD

##### SBRAM 写地址寄存器

**SFR (FEDE<sub>H</sub>/6F<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD</b>	<b>WA</b>	<b>0</b>	<b>WPTR</b>												<b>0</b>
rwh	rwh	r	rwh												r

符号	位序号	类型	功能描述
<b>WPTR</b>	[12:1]	rwh	<b>写指针</b> 选择 <b>SBRAM</b> 中的字地址，向该地址处写数据。当寄存器 <b>DATA1</b> 被写入时，该指针自动加 1。
<b>WA</b>	14	rwh	<b>回绕</b> 该位指示写指针 <b>WPTR</b> 是否因为自动地址增加而出现了回绕。  0 未发生地址回绕  1 已经检测到地址回绕，必须由软件清除该位。
<b>MOD</b>	15	rwh	<b>修改</b> 该位指示是否最近的 <b>SBRAM</b> 写访问使得 <b>WPTR</b>

符号	位序号	类型	功能描述
			自动加 1。 0 最近的数据写访问指向 DATA0，不自动修改 WPTR 指针。 1 最近的数据写访问指向 DATA1，WPTR 自动加 1。
<b>0</b>	0, 13	r	<b>保留</b> 读操作返回 0；应写入 0。

### 3.12.2.3 SBRAM 数据寄存器 0

读取 SBRAM 时，由该寄存器给出读数据。写操作时，如果对应的地址指针不修改，则使用该寄存器作为写数据的目标地址。

由上电复位复位该寄存器。

#### SBRAM\_DATA0

#### SBRAM 数据寄存器 0

SFR (FEE0<sub>H</sub>/70<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
<b>DATA</b>	[15:0]	rwh	<b>SBRAM 数据</b> 该位域包含数据：最近的 SBRAM 读访问数据或者是要写入到 SBRAM 的数据。 读访问该位域始终给出保存在由读指针 RADD.RPTR 指向的地址的数据。 写访问（至少）低位字节使得保存在该寄存器中的数据被写入到写指针 WADD.WPTR 指向的地址。

### 3.12.2.4 SBRAM 数据寄存器 1

读取 SBRAM 时，由该寄存器给出读数据。写操作时，如果对应的地址指针修改，则使用该寄存器作为写数据的目标地址。

由上电复位复位该寄存器。

#### SBRAM\_DATA1

**SBRAM 数据寄存器 1**                      **SFR (FEE2<sub>H</sub>/71<sub>H</sub>)**                      **复位值: 0000<sub>H</sub>**

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

符号	位序号	类型	功能描述
DATA	[15:0]	rwh	<p><b>SBRAM 数据</b></p> <p>该位域包含数据：最近的 SBRAM 读访问数据或者是要写入到 SBRAM 的数据。</p> <p>读访问该位域始终给出保存在由读指针 RADD.RPTR 指向的地址的数据。</p> <p>写访问（至少）低位字节使得保存在该寄存器中的数据被写入到写指针 WADD.WPTR 指向的地址。</p>

## 4 存储器检查器模块 (MCHK)

XE16xyM 的存储器检查器模块 (MCHK) 用于检查存储器、寄存器（如配置寄存器）或通信通道的数据一致性。该模块计算数据块的校验和，通常称为循环冗余码 (CRC)。该模块基于一个多输入线性反馈移位寄存器 (MISR)、以生成并行签名的方式来实现。由于基于线性反馈移位寄存器 (LFSR)，它还能产生伪随机数和循环码。

本章结构如下：

- 存储器检查器模块操作概述（见[章节 4.1](#)）
- 存储器检查器模块功能描述（见[章节 4.2](#)）
- 存储器检查器模块寄存器描述（见[章节 4.3](#)）
- 通用寄存器描述（见[章节 4.4](#)）
- 存储器检查器模块的接口（见[章节 4.5](#)）

### 4.1 操作概述

对于编程人员来说，MCHK 是一组与该外设模块相关的寄存器。为了指示相应的错误或操作事件，可使用端口引脚作为信号“MATCH”产生外部事件，可使用中断线作为信号“MISMATCH”产生内部事件。MCHK 和 CPU 同时复位，因此它可用作 CPU 协处理器。这可确保 CPU 退出复位状态之后，MCHK 处于一个确定的状态。

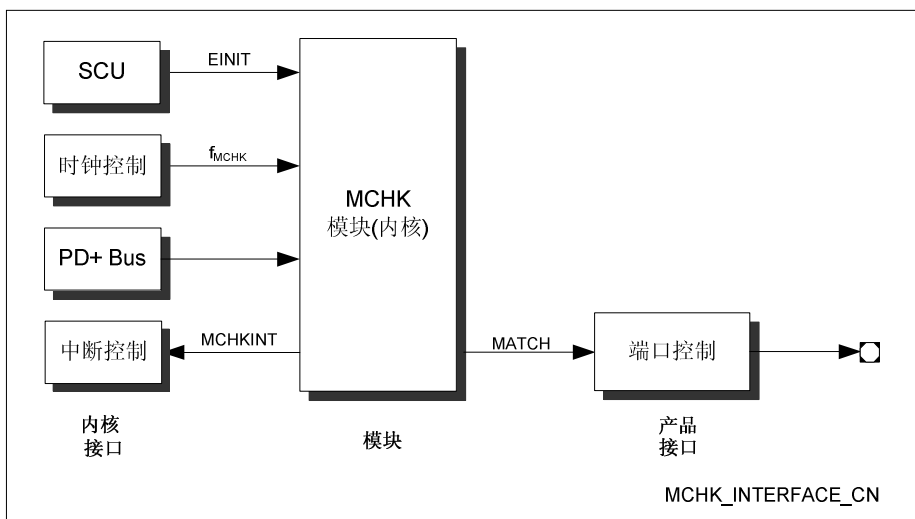


图 4-1 接口框图

注: MATCH 输出仅和不低于 144 引脚封装的器件的外部端口相连。

## 4.2 功能描述

传统的数字处理系统通常配备一些易失或非易失存储器元件。这些存储器向 CPU 提供 (储存) 数据和指令, 以完成嵌入式系统的主要计算功能。这包括系统管理, 通过协调各系统单元以执行系统任务。

通常, 存储器出错对于嵌入式系统的安全和可靠性而言非常致命。因此, 这些存储器具有检查数据一致性的机制, 如奇偶校验或 ECC (纠错码)。这些机制可检测每个字中一定数量 (如双位错误) 的错误 (位线错误)。级联码 (区块码、字代码) 还可检测每个字中的多位错误 (字线错误), 该机制提高了错误覆盖率。

MCHK 是一种并行签名压缩电路, 通过计算级联 CRC 区块码以提高错误检测覆盖率。MCHK 可检测存储器、寄存器中或通过串行通信线传送的数据块是否有错。通过对待检测数据块进行并行检测输入压缩, MCHK 可降低由重复的错误序列所导致的错误屏蔽概率。另外, MCHK 还能生成伪随机数。

MCHK 使用一个多输入线性反馈移位寄存器生成数据块的校验和 (签名)。多输入线性反馈移位寄存器 (MISR) 中的内部反馈输入位是整个移位寄存器中某些位异或 (XOR) 的结果 (LSFR: 线性反馈移位寄存器)。

该校验和生成器包含一个计算区块码的算术电路。该电路由独立的硬件实现, 因此不依赖于待检测的存储器。只有在进行模块配置时才需从存储器获取初始数据。为了避免使用多主机系统 (CPU、DMA 等), 由 CPU (如 PEC、子程序) 处理数据读/写操作。由于不同存储器可能包含各自不同的 CPU 指令代码, 这些读写操作和待检测的存储器类型有关, 因此, MCHK 提供了额外的措施允许由 CPU 检测错误的数据转移操作。

可检测以下错误情况:

- CPU 错误配置 MCHK
- CPU 未提供相应地址范围内的数据
- CPU 未提供正确数量的数据
- CPU 不能正确检查在线生成的 CRC 区块码和期望的离线 (开发期间) 生成的 CRC 区块码是否匹配

该电路的原理是产生一个 CRC 区块码检查的外部编码的活动信号。另外, 在没有外部通知的情况下, 不能改变电路的配置。该活动信号不是一个静态信号, 它以预先设定的方式改变极性从而避免静态错误, 如输出电路部分开路或短路。

该电路由以下部分组成:

- 算术电路, 根据传送到该电路输入寄存器中的数据计算 CRC 区块码。
- 比较单元, 检查计算出的 CRC 区块码是否正确。MCHK 将 CRC 区块码结果寄存器中的内容与固定值 (FADE'EDDA<sub>H</sub>) 进行比较。在计算 CRC 值之前, 用

## 存储器检查器模块 (MCHK)

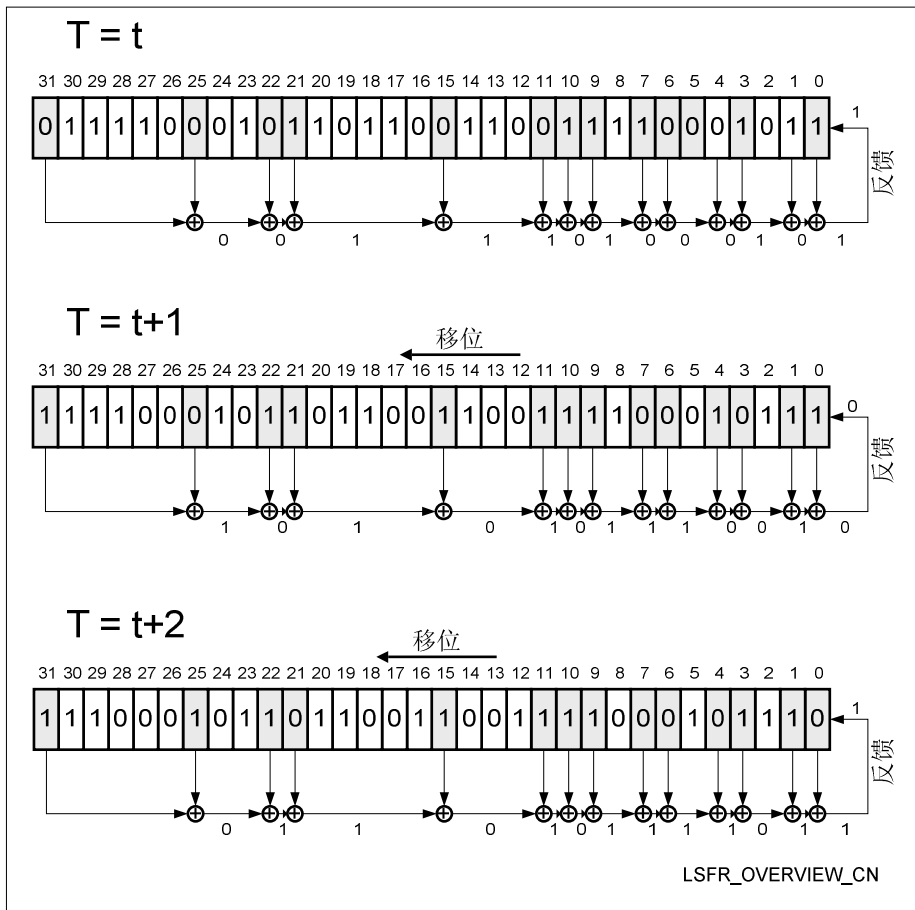
一个特定值（种子值）初始化结果寄存器。选择种子值时必须使得计算出的区块码最终为固定值（FADE'EDDA<sub>H</sub>）。对于线性代码来说（如 CRC 区块码），该方式效果很好。

- 允许对给定数量的数据进行 CRC 区块码计算的方法。因此使用功能上的冗余保证此方法的实现。接收到给定数量的输入数据之后，MCHK 内的计数寄存器触发对计算得到的 CRC 区块码进行比较。当触发产生新 CRC 区块码时（数据转移程序的循环变量或 PEC 内的计数寄存器），由 CPU 重载 CRC 区块码的种子值。
- 内部服务请求产生，用于在出现错误情况使能软件恢复。这可以是存储在其它 Flash 模块（不同于产生错误的 Flash 模块）中的软件程序，如支持缩减操作功能。这样仍然留有风险：在该错误中断子程序中，CPU 可能将空数据写入到存储器检查器并重写 COUNT 寄存器的值。
- 所有配置寄存器由时间冗余机制保护。因此，只有遵循特定的写操作序列才能修改配置寄存器的内容（初始化结束保护）。此外，COUNT 寄存器由与内容相关的访问机制保护。
- 每次成功生成 CRC 区块码时产生一个外部 MATCH 信号，它可用于触发外部时间窗看门狗。若在特定的时间窗内出现过多或过少的 MATCH 信号翻转，该外部看门狗可产生一个复位操作。为了确保区块码产生单元正确工作，应用程序必须通过一个不正确的比较（无 MATCH 信号翻转），在看门狗时间窗外不时执行区块码生成。因为在看门狗时间窗之外，错误的比较电路会产生一个 MATCH 信号翻转，所以该操作可用于检查比较电路的功能是否正常。

### 4.2.1 LFSR 原理

影响内部反馈位的位元位置列表被称为抽头序列。图 4-2 给出 LSFR 的原理。图中假设抽头序列为[32, 26, 23, 22, 16, 12, 11, 10, 8, 7, 5, 4, 2, 1]。LSFR 在激活之后，所有位（向最高有效位的方向）左移一位。抽头位置上的所有位异或的结果作为反馈送至位 0。下一次激活时，重复上述过程。

影响内部反馈输入的 LFSR 输出被称为抽头（在图 4-2 中标示为灰色）。



**图 4-2 LSFR 原理**

LFSR 的抽头序列可表示为多项式模 2。这意味着对于未实现的反馈抽头，相应的多项式系数必须为 0；对于已实现的反馈抽头，多项式系数必须为 1。该多项式被称为反馈多项式或特征多项式。例如，如果抽头位于第 32、26、23、22、16、12、11、8、7、5、4、和 2 位（如下式），所得的 LFSR 多项式为：

$$G^{32} = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad (4.1)$$

**存储器检查器模块 (MCHK)**

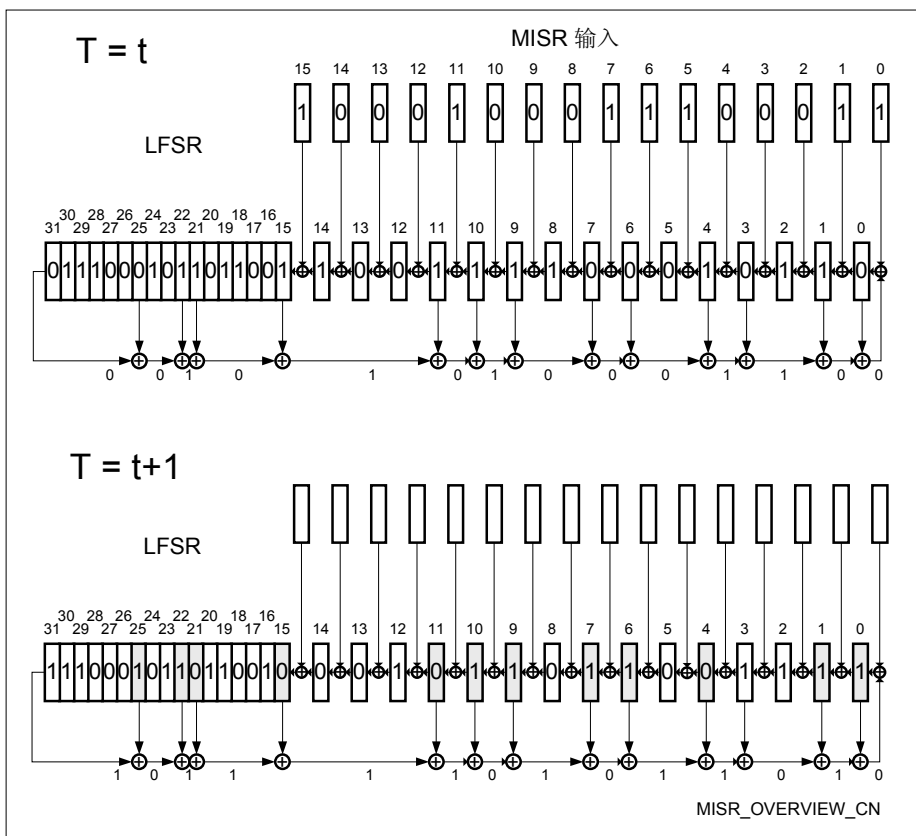
多项式中的“+1”不和抽头对应。各项的幂代表抽头位（从最低有效位开始）。多项式可由二进制数表示。各项的幂以二进制格式 1 表示（从最高有效位开始）。因此，对于上面列出的多项式，对应的数为：

$$G^{32} = 1000\ 0010\ 0110\ 0000\ 1000\ 1110\ 1101\ 1011_B = 8260'8EDB_H$$

MCHK 使用的多项式在抽头多项式寄存器 **TPRH** 和 **TPRL** 中定义。

#### 4.2.2 MISR 原理

除 LFSR 内部反馈输入位之外，还可将 16 个外部位加载到 LFSR 中（多输入）。这 16 个输入位与移出或反馈回 LFSR 的位进行异或运算。



**图 4-3 MISR 原理**



存储器检查器模块 (MCHK)

LFSR/MISR 的初始值被称为种子，可在结果寄存器 **RRL** 或 **RRH** 中设置。由于 MISR 的操作是确定的，因此 MISR 生成的序列值完全由其当前（或先前）状态和输入决定。

### 4.2.3 常用的多项式

在全球标准化系统中使用的循环码多项式，其本身尚未标准化。目前使用的大多数循环码在强度或结构方面具有一些弱点。循环码的标准化有助于设计出更好的常用循环码。下表给出产生顺序 CRC 签名的常用多项式。

表 4-1 一些常用多项式

名称	多项式	最大 数据宽度	常态的倒数 (逆)
CRC-8-ATM	$x^8 + x^2 + x + 1$	8 位	0000' 0083 <sub>H</sub> (0000' 00C1 <sub>H</sub> )
CRC-8- CCITT	$x^8 + x^7 + x^3 + x^2 + 1$	8 位	0000' 00C6 <sub>H</sub>
CRC-8-Dallas	$x^8 + x^5 + x^4 + 1$	8 位	0000' 0098 <sub>H</sub>
CRC-8	$x^8 + x^7 + x^6 + x^4 + x^2 + 1$	8 位	0000' 00EA <sub>H</sub>
CRC-8 SAE J1850	$x^8 + x^4 + x^3 + x^2 + 1$	8 位	0000' 008E <sub>H</sub>
CRC-1 (奇偶校验)	$x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$	8 位	0000' 00FF <sub>H</sub>
CRC-10	$x^{10} + x^9 + x^5 + x^4 + x + 1$	10 位	0000' 0319 <sub>H</sub> (0000' 0263 <sub>H</sub> )
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$	12 位	0000' 0C07 <sub>H</sub> (0000' 0E03 <sub>H</sub> )
CRC-15-CAN	$x^{15} + x^{14} + x^{10} + x^8 + x^7 + x^4 + x^3 + 1$	15 位 (13 位)	0000' 62CC <sub>H</sub> (0000' 19A3 <sub>H</sub> )
CRC-1 (奇偶校验)	$x^{16} + x^{15} + x^{14} + x^{13} + x^{12} + x^{11} + x^{10} + x^9 + x^8 + x^7 + x^6 + x^5 + x^4 + x^3 + x^2 + x + 1$	16 位	0000' FFFF <sub>H</sub>
CRC-16- CCITT	$x^{16} + x^{12} + x^5 + 1$	16 位	0000' 8810 <sub>H</sub>

存储器检查器模块 (MCHK)

名称	多项式	最大数据宽度	常态的倒数 (逆)
CRC-16-IBM	$x^{16} + x^{15} + x^2 + 1$	16 位	0000' C002 <sub>H</sub>
CRC-24-Radix-64	$x^{24} + x^{23} + x^{18} + x^{17} + x^{14} + x^{11} + x^{10} + x^7 + x^6 + x^5 + x^4 + x^3 + x + 1$	16 位 (24 位多项式)	00C3' 267D <sub>H</sub> (BE64'C300 <sub>H</sub> )
CRC-32-IEEE802.3/MPEG2	$x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1$	16 位 (32 位多项式)	8260' 8ED6 <sub>H</sub> (DB71'0641 <sub>H</sub> )
CRC-32C	$x^{32} + x^{28} + x^{27} + x^{26} + x^{25} + x^{23} + x^{22} + x^{20} + x^{19} + x^{18} + x^{14} + x^{13} + x^{11} + x^{10} + x^9 + x^8 + x^6 + 1$	16 位 (32/27 位多项式)	8F6E' 37A0 <sub>H</sub> (05EC'76F1 <sub>H</sub> )

注：上面的多项式通常用于产生顺序签名（通常称之为 CRC），与 MCHK 产生的并行签名的结果不同。

#### 4.2.4 存储器检查器模块的结构

LFSR 由结果寄存器高位 (RRH) 和结果寄存器低位 (RRL) 组成。可通过存放在这两个寄存器中的种子值初始化 MCHK。写访问寄存器 RRL 时，MISR 计数寄存器 COUNT 由上次写入的 COUNT 值重新加载。结果寄存器 RRH 和 RRL 可用于读取加载到 MCHK 中的数据块的最终或中间签名。

可通过将 8 位或 16 位数据写入到 MISR 输入寄存器 IR 的方式将数据加载到 MCHK 中。每次写访问 IR 时，MISR 计数寄存器 COUNT 的值递减。

定义多项式时，向抽头多项式寄存器高位 (TRPH) 和抽头多项式寄存器低位 (TPRL) 中写入二进制常态倒数值。抽头多项式寄存器和结果寄存器进行二进制与操作。如果与操作的结果中 1 的个数为奇数，则反馈回 1<sub>B</sub>，否则返回 0<sub>B</sub>。如果由抽头多项式寄存器定义的多项式的最高 1<sub>B</sub> 位的位置小于反馈回存储器检查器模块的数据的最高 1<sub>B</sub> 位的位置，存储器检查器模块的有效性则显著降低。因此通常抽头多项式寄存器的内容必须大于 80<sub>H</sub>（对于 8 位数据）和 8000<sub>H</sub>（对于 16 位数据）。

MISR 计数寄存器 COUNT 的内容从 0001<sub>H</sub> 递减至 0000<sub>H</sub> 时，若 LFSR 结果寄存器高位 (RRH) 不等于 FADE<sub>H</sub> 或 LFSR 结果寄存器低位 (RRL) 不等于 EDDA<sub>H</sub>，则产生一个服务请求信号。如果 LFSR 结果寄存器的内容等于 FADE'EDDA<sub>H</sub>，则外部 MATCH 信号翻转。图 4-4 示出校验和电路的结构。

## 存储器检查器模块 (MCHK)

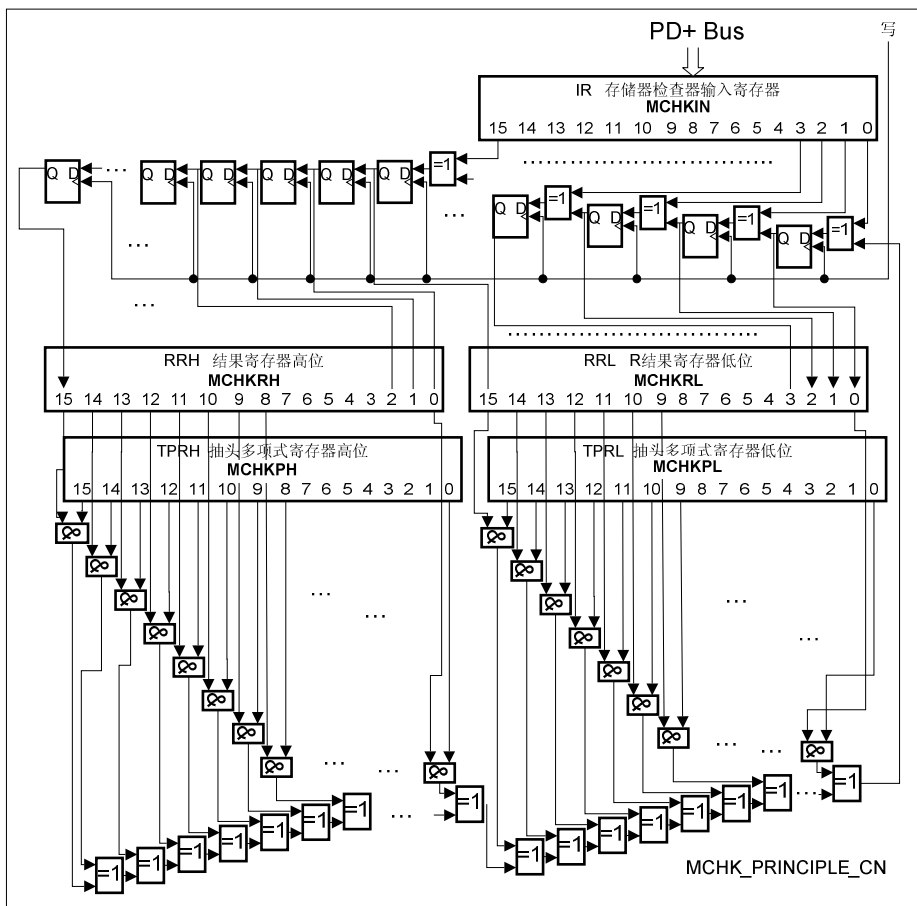


图 4-4 MCHK 校验和电路的实现

### 4.2.5 存储器检查器模块的最佳使用

MCHK 最好和 CPU 配合使用。CPU 从选定的地址范围读取数据块，然后将其写入输入寄存器 **IR**。也可以配置 PEC 使用 8 位或 16 位转移 (PECCx.BWT = 0: 16 位; PECCx.BWT = 1: 8 位) 将数据块移入输入寄存器 **IR**。每次对寄存器 **IR** 的写操作触发一次中间多项式校验和计算，计算结果保存在结果寄存器 **RRL** 和 **RRH** 中。此外，每次对寄存器 **IR** 进行写操作都会对计数寄存器 **COUNT** 的内容减 1。

为了启动存储器检查，必须用种子值初始化结果寄存器 (如写入期望的启动值) 并建立 CPU 或 PEC 操作 (起始地址、长度等)。

当定义的数据块全部写入寄存器 **IR**，如果 LFSR 结果寄存器 **RRH** 和 **RRL** 中的内容不等于 FADE'EDDA<sub>H</sub>，可产生中断。

MCHK 可使用标准以太网 (IEEE802.3/MPEG2) 多项式：

$$G^{32} = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad (4.2)$$

*注：尽管使用上面的多项式，并行签名的结果 (MISR) 和以太网协议使用的顺序签名 (LFSR) 的结果不同。*

#### 4.2.6 种子值 (magic word) 的计算

为了成功进行 CRC 计算并产生 MATCH 或 MISMATCH 信号，必须在计算 CRC 之前将特定的种子值 (也称为 magic word) 载入到 LFSR 结果寄存器 **RRH** 和 **RRL** 中。应当在开发相应数据期间计算 magic word。只有当抽头多项式的阶数等于 32 (抽头多项式 **TPRH** 的最高有效位必须等于 1) 时，才能产生 magic word。否则，高阶位的结果不等于需要的最终结果 FADE EDDA<sub>H</sub>。下面简要描述程序的基本原理。该程序使用 VBA (微软 VB)。**Data\_Array** 包含需要计算 CRC 的所有 16 位数据。**COUNT** 将需要计算 CRC 的数据个数传递给子程序，该数值在存储器检查器计数器寄存器 **COUNT** 中定义。通过 **RRH** 和 **RRL** 将 magic word 传递回调用子程序，因为必须将种子值写入到 LFSR 结果寄存器高位 (**RRH**) 和结果寄存器低位 (**RRL**) 中。因为 VBA 没有无符号整数格式，在该示范代码中使用长整数格式。

```
Sub MagicWord(ByRef Data_Array() As Long, _
              ByVal COUNT, TPRH, TPRL as Long, _
              ByRef RRH, RRL As Long)
    Dim i, j, order, feedback_bit As Integer
    Dim temp As Long
    RRH = &HFADE
    RRL = &HEDDA
    For j = COUNT To 1 Step -1
        If TPRH <> 0 Then                                ' order of polynomial > 16
            order = 31
            Do While TPRH< 2^(order-16) 'calculate order of polynomial,
                order = order - 1        ' determines bit position
            Loop                          ' "rolled" out of LFSR
        Else                                    ' order of polynomial < 17
            order = 15
            Do While TPRL< 2 ^order      ' calculate order of polynomial,
                order = order - 1        ' determines bit position
            Loop                          ' "rolled" out of LFSR
        End If
    Next j
    ' Magic word calculation logic would follow here, but the provided text is truncated.
```

**存储器检查器模块 (MCHK)**

```
RRL = RRL Xor Data_Array(j) ' MISR XOR Input
feedback_bit = RRL And 1 ' Extract CRC feedback bit
RRL = RRL \ 2 ' 32 bit shift right (LFSR)
RRL = RRL + (RRH And 1) * 2 ^ 15
RRH = RRH \ 2
temp = (RRH And TPRH) _
      Xor (RRL And TPRL)      ' generate 32 TAP Bits
For i = 0 To 15
    feedback_bit = feedback_bit _
                  Xor ((temp \ (2 ^ i)) And 1)
Next i
' XOR TAP bits to bit
If feedback_bit <> 0 Then      ' TAP feedback bit is equal 1
    If order > 16 Then
        RRH = RRH Or (2 ^ (order - 16))
    Else
        RRL = RRL Xor (2 ^ order)
    End If
End If
Next j
' calculate CRC of all data
End Sub
```

#### 4.2.7 应用举例

假设使用 MCHK 检测一组 21 个 16 位数据是否出错。存储器检查器模块使用标准以太网 (IEEE802.3/MPEG2) 多项式:

$$G^{32} = x^{32} + x^{26} + x^{23} + x^{22} + x^{16} + x^{12} + x^{11} + x^{10} + x^8 + x^7 + x^5 + x^4 + x^2 + x + 1 \quad (4.3)$$

即多项式为 8260 8EDB<sub>H</sub>，将该值写入 **TPRH** 和 **TPRL**。接着，根据[章节 4.2.6 “种子值 \(magic word\) 的计算”](#)中的程序描述，离线计算该组数据的 magic word。对于[表 4-2](#)中给定的数据，计算出 magic word = AA1F ED4E<sub>H</sub>并将其写入 **RRH** 和 **RRL**。

表 4-2 CRC 检查举例

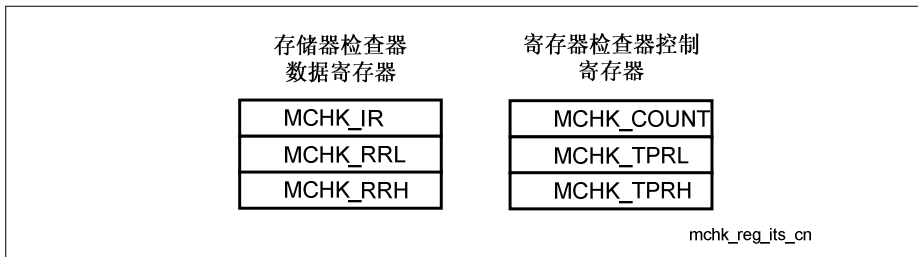
用户操作	数据值	寄存器的内容				
		COUNT	RRH	RRL	TPRH	TPRL
抽头多项式写入到 <b>TPRH</b> 中	8260 <sub>H</sub>	xxxx	xxxx	xxxx	8260 <sub>H</sub>	xxxx
抽头多项式写入到 <b>TPRL</b> 中	8EDB <sub>H</sub>	xxxx	xxxx	xxxx	8260 <sub>H</sub>	8EDB <sub>H</sub>
magic word 写入到 <b>RRH</b> 中	AA1F <sub>H</sub>	0015 <sub>H</sub>	AA1F <sub>H</sub>	xxxx	8260 <sub>H</sub>	8EDB <sub>H</sub>
magic word 写入到 <b>RRL</b> 中	ED4E <sub>H</sub>	0015 <sub>H</sub>	AA1F <sub>H</sub>	ED4E <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据数量写入到 <b>COUNT</b> 中	0015 <sub>H</sub>	0015 <sub>H</sub>	xxxx	xxxx	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 1 写入 <b>IR</b>	8BED <sub>H</sub>	0014 <sub>H</sub>	543F <sub>H</sub>	5171 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 2 写入 <b>IR</b>	AA61 <sub>H</sub>	0013 <sub>H</sub>	A87E <sub>H</sub>	0883 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 3 写入 <b>IR</b>	C64E <sub>H</sub>	0012 <sub>H</sub>	50FC <sub>H</sub>	D749 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 4 写入 <b>IR</b>	17E4 <sub>H</sub>	0011 <sub>H</sub>	A1F9 <sub>H</sub>	B976 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 5 写入 <b>IR</b>	A329 <sub>H</sub>	0010 <sub>H</sub>	43F3 <sub>H</sub>	D1C5 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 6 写入 <b>IR</b>	66B5 <sub>H</sub>	000F <sub>H</sub>	87E7 <sub>H</sub>	C53E <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 7 写入 <b>IR</b>	422A <sub>H</sub>	000E <sub>H</sub>	0FCF <sub>H</sub>	C857 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>

**存储器检查器模块 (MCHK)**

用户操作	数据值	寄存器的内容				
		COUNT	RRH	RRL	TPRH	TPRL
数据 8 写入 <b>IR</b>	4FF6 <sub>H</sub>	000D <sub>H</sub>	1F9F <sub>H</sub>	DF58 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 9 写入 <b>IR</b>	4046 <sub>H</sub>	000C <sub>H</sub>	3F3F <sub>H</sub>	FEF6 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 10 写入 <b>IR</b>	911C <sub>H</sub>	000B <sub>H</sub>	7E7F <sub>H</sub>	6CF0 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 11 写入 <b>IR</b>	1FA0 <sub>H</sub>	000A <sub>H</sub>	FCFE <sub>H</sub>	C640 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 12 写入 <b>IR</b>	BF38 <sub>H</sub>	0009 <sub>H</sub>	F9FD <sub>H</sub>	33B9 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 13 写入 <b>IR</b>	9FE3 <sub>H</sub>	0008 <sub>H</sub>	F3FA <sub>H</sub>	F891 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 14 写入 <b>IR</b>	44DD <sub>H</sub>	0007 <sub>H</sub>	E7F5 <sub>H</sub>	B5FE <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 15 写入 <b>IR</b>	749A <sub>H</sub>	0006 <sub>H</sub>	CFEB <sub>H</sub>	1F67 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 16 写入 <b>IR</b>	8C09 <sub>H</sub>	0005 <sub>H</sub>	9FD6 <sub>H</sub>	B2C7 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 17 写入 <b>IR</b>	D0F5 <sub>H</sub>	0004 <sub>H</sub>	3FAD <sub>H</sub>	B57A <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 18 写入 <b>IR</b>	DC5F <sub>H</sub>	0003 <sub>H</sub>	7F5B <sub>H</sub>	B6AB <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 19 写入 <b>IR</b>	DB06 <sub>H</sub>	0002 <sub>H</sub>	FEB7 <sub>H</sub>	B651 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 20 写入 <b>IR</b>	4604 <sub>H</sub>	0001 <sub>H</sub>	FD6F <sub>H</sub>	2AA7 <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>
数据 21 写入 <b>IR</b>	B894 <sub>H</sub>	0000 <sub>H</sub>	FADE <sub>H</sub>	EDDA <sub>H</sub>	8260 <sub>H</sub>	8EDB <sub>H</sub>

### 4.3 存储器检查器模块寄存器

对编程人员来说，MCHK 由一组 SFR 构成。



**图 4-5 MCHK 内核寄存器**

以下各表给出 MCHK 的寄存器和地址。

**表 4-3 寄存器的地址空间**

模块	基址	结束地址	注
MCHK	0000 <sub>H</sub>		

**表 4-4 寄存器概览**

寄存器缩写名	寄存器全名	偏移地址	页码
<b>ID</b>	模块 ID 寄存器	FFE0 <sub>H</sub>	<a href="#">页 4-17</a>
<b>IR</b>	存储器检查器输入寄存器	FF58 <sub>H</sub>	<a href="#">页 4-14</a>
<b>RRL</b>	存储器检查器结果寄存器低位	F058 <sub>H</sub>	<a href="#">页 4-14</a>
<b>RRH</b>	存储器检查器结果寄存器高位	F05A <sub>H</sub>	<a href="#">页 4-15</a>
<b>COUNT</b>	存储器检查器计数寄存器	FE5A <sub>H</sub>	<a href="#">页 4-15</a>
<b>TPRL</b>	存储器检查器多项式寄存器低位	F05C <sub>H</sub>	<a href="#">页 4-16</a>
<b>TPRH</b>	存储器检查器多项式寄存器高位	F05E <sub>H</sub>	<a href="#">页 4-17</a>

*注：所有寄存器的复位等级和 CPU 的复位等级相同。*



#### 4.3.1 存储器检查器输入寄存器

输入寄存器接收写入到 MCHK 需要进行校验和计算的数据。如果 CPU 转移到寄存器 MCHK\_IR 的数据为 8 位宽，16 位 MCHKIN 值中未使用的寄存器位被当作 0。

##### IR

输入寄存器															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCHKIN															
w															

符号	位序号	类型	功能描述
MCHKIN	[15:0]	w	存储器检查器模块输入 写入到 MCHKIN 的值用于下次校验和计算。 读取该位域返回 0000 <sub>H</sub> 。

注: MCHK\_IR 为只写寄存器, 对该寄存器的任意读操作都将返回 0000<sub>H</sub>。

#### 4.3.2 存储器检查器结果寄存器

结果寄存器中存放存储器检查操作的签名 (结果)。在开始计算校验和之前, 需要向这两个寄存器写入初始校验和计算值 (种子)。

##### RRL

结果寄存器低位															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MCHKRL															
rwh															

符号	位序号	类型	功能描述
MCHKRL	[15:0]	rwh	存储器检查器结果低位 该位域包含 32 位校验和计算结果的低 16 位。

注: 写访问寄存器 RRL 的操作将复位 (重载) MISR 计数寄存器。因此。写访问 RRL 将立即启动新一轮 CRC 计算。

## **RRH**

**结果寄存器高位**

**ESFR (F05A<sub>H</sub>/2D<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MCHKRH</b>															
rwh															

符号	位序号	类型	功能描述
<b>MCHKRH</b>	[15:0]	rwh	<b>存储器检查器结果高位</b> 该位域包含 32 位校验和计算结果的高 16 位。

### **4.3.3 存储器检查器计数寄存器**

每次写访问输入寄存器时，计数寄存器 COUNT 减 1。计数寄存器递减至 0000<sub>H</sub> 时，若结果寄存器的值不等于 FADE'EDDA<sub>H</sub>，则产生一个服务请求（中断）；若结果寄存器的值等于 FADE'EDDA<sub>H</sub>，则输出信号 MATCH 翻转。当 CPU 向 RRL 传送新的种子值（magic word）时，用上次的写入值重载计数寄存器。

当 CPU 或 PEC 写访问寄存器 COUNT 时，若其内容不等于上次的写入值，则产生一个服务请求（中断）且翻转输出信号 MATCH。该特性可检测出软件未正确处理 MCHK 的情况（例如由于错误的程序存储器所导致）。MATCH 信号的及时正确翻转可用作某些单元（如外部窗看门狗）的活动信号。MATCH 信号的复位值 = 0<sub>B</sub>。

因为寄存器 COUNT 控制高安全性的系统功能，因此它受特殊的寄存器安全机制保护，执行 EINIT 指令之后，该关键的系统功能不会被无意修改。

## **COUNT**

**计数寄存器**

**SFR (FE5A<sub>H</sub>/2D<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MCHKCNT</b>															
rwh															

符号	位序号	类型	功能描述
<b>MCHKCNT</b>	[15:0]	rwh	<b>存储器检查器计数</b> MCHKCNT 指示当前数据块中剩余的、即将进入 MCHK 的数据个数。 0001 <sub>H</sub> 还剩 1 个数据要写入寄存器 IR 以触发比较

**存储器检查器模块 (MCHK)**

符号	位序号	类型	功能描述
			操作产生 MATCH 信号或中断信号 NOMATCH。
	0002 <sub>H</sub>		还剩 2 个数据要写入寄存器 <b>IR</b> 。
	.... <sub>H</sub>		....
	FFFE <sub>H</sub>		还剩 65534 个数据要写入寄存器 <b>IR</b> 。
	FFFF <sub>H</sub>		还剩 65535 个数据要写入寄存器 <b>IR</b> 。
	0000 <sub>H</sub>		还剩 65536 个数据要写入寄存器 <b>IR</b> 以触发比较操作产生 MATCH 信号或中断信号 NOMATCH。

注：只有当 **MCHKCNT** 等于上次写入该寄存器的值时，才能对寄存器 **COUNT** 进行写操作。否则将触发一个服务请求（中断）并使 **MATCH** 信号翻转。

写访问寄存器 **RRL** 之后才能修改寄存器 **COUNT**，使用先前的写入值重载 **COUNT**（参见表 4-2）。

注：执行 **EINIT** 之后，**COUNT** 受寄存器安全机制写保护。

#### 4.3.4 存储器检查器多项式寄存器

多项式寄存器中存放用于校验和计算的 LFSR 多项式。

因为多项式寄存器控制高安全性的系统功能，因此它们受特殊的寄存器安全机制保护，执行 **EINIT** 指令之后，该关键的系统功能不会被无意修改。

#### TPRL

抽头多项式寄存器低位						<b>ESFR (F05C<sub>H</sub>2E<sub>H</sub>)</b>						复位值: <b>FFFF<sub>H</sub></b>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MCHKCPL</b>															
rwh															

符号	位序号	类型	功能描述
<b>MCHKPL</b>	[15:0]	rwh	<b>存储器检查器多项式低位</b> 该位域中存放二进制抽头多项式的低 16 位。

## TPRH

抽头多项式寄存器高位

**ESFR (F05E<sub>H</sub>/2F<sub>H</sub>)**

**复位值: FFFF<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MCHKCPH</b>															
rwh															

符号	位序号	类型	功能描述
<b>MCHKPH</b>	[15:0]	rwh	<b>存储器检查器多项式高位</b> 该位域中存放二进制抽头多项式的高 16 位。

*注：执行 EINIT 指令之后，TPRH 和 TPRL 受寄存器安全机制写保护。*

## 4.4 一般寄存器

ID 寄存器是只读寄存器，用于指示 MCHK 模块的 ID。该寄存器提供 8 位模块 ID 和 8 位版本号。

### 4.4.1 ID 寄存器

#### ID

模块 ID 寄存器

**MEM(FFE0<sub>H</sub>)**

**复位值: 3BXX<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD_NUMBER</b>								<b>MOD_REV</b>							
r								r							

符号	位序号	类型	功能描述
<b>MOD_REV</b>	[7:0]	r	<b>模块修订版编号</b> 位 7-0 给出模块修订版编号，从 01 <sub>H</sub> 开始编号（第一版）。
<b>MOD_NUMBER</b>	[15:8]	r	<b>模块 ID 编号</b> 位 15-8 定义模块 ID。MCHK 的模块 ID 为 3B <sub>H</sub> 。

## 4.5 MCHK 模块的接口

MCHK 模块可产生中断请求和外部活动信号。

中断请求信号和 SCU 相连，可被送至 SCU 的中断节点之一。

MATCH 输出和不低于 144 引脚封装的器件的外部引脚相连。

表 4-5 XE16xyM 中 MCHK 的数字信号连接

信号	来自/送至模块	MCHK I/O
MATCH	P8.6	O
INT(MISMATCH)	SCU	O
MCHKIN_Write_Access	来自 MCHKIN 的写触发	I <sup>1)</sup>
MCHKCNT_Write_Access	来自 MCHKCNT 的写触发	I <sup>1)</sup>
EINIT	SCU	I

1) 该信号在模块内部产生，不会出现在模块边界上。

## 5 中央处理器（CPU）

中央处理器（CPU）的基本任务是对指令进行读取和解码，为算术逻辑单元（ALU）和乘累加单元（MAC）提供操作数，并在 ALU 和 MAC 中进行操作数运算，保存先前的计算结果。因为 CPU 是 XE16xyM 微控制器的核心，它还受到外设子系统操作的影响。

XE16xyM 中实现了 5 级处理流水线（外加 2 级取指令流水线），能够并行处理多达 5 条指令。由于采取了该并行机制，XE16xyM 的大多数指令都能在一个时钟周期内完成。

本章将介绍顺序指令和分支指令的流水线工作，以及特别用于加速跳转指令执行的硬件部分；还将介绍一般的指令执行时序，包括标准时序和异常情况下的时序。

对内部存储器的访问通常由 CPU 执行，对片外外设或外部存储器的访问通过片上外部总线控制器（EBC）实现。当程序或数据地址指向外部地址空间时，CPU 会自动调用 EBC 来实现该操作。

访问外部存储器期间，CPU 会尽可能保持运行。如果所需的外部数据尚未准备好，或在完成前一次的访问之前，CPU 请求新的外部存储器访问，EBC 将使 CPU 处于保持状态（暂停），直至完成被请求的访问操作。EBC 模块将在独立一章专门介绍。

XE16xyM 的片上外设单元独立于 CPU 工作。CPU 与这些外设之间通过特殊功能寄存器（SFR）或共享的存储器区域来交换数据和控制信息。

只要外设请求（非确定性的）CPU 操作，片上中断控制器就会比较所有挂起的外设服务请求，并按优先级排序。如果当前 CPU 操作的优先级低于所选外设中断请求的优先级，将会产生中断。

XE16xyM 支持两种基本的中断处理类型：

- **标准中断处理**强制 CPU 在跳转至中断向量表之前，将当前程序状态及返回地址保存在堆栈中。
- **PEC 中断事件**从当前 CPU 操作中仅“窃取”一个机器周期，通过片上外围事件控制器（PEC）执行一次数据传送。

程序执行期间所检测到的系统错误（硬件强制中断）和外部的非屏蔽中断被视为具有很高优先级的标准中断加以处理。

与其它片上外设相比，看门狗定时器与 CPU 之间的联系更紧密。若看门狗被使能，CPU 要在设定的时间之内（可编程设定）服务（刷新）看门狗。否则，看门狗会复位芯片。因而，当执行错误代码时，看门狗能够防止 CPU 程序跑飞。CPU 提供一组用于使能（ENWDT）、禁止（DISWDT）和服务（SRVWDT）看门狗定时器的指令。

除了正常工作状态外，CPU 还可通过执行 IDLE 指令进入空闲模式。空闲模式下，CPU 停止执行程序，但仍可响应中断或 PEC 请求。可通过中断请求或硬件复位操作强制 CPU 从空闲状态转换至有效工作状态。

在 XE16xyM 中未使能 PWRDN 指令。如果执行 PWRDN 指令，则会执行一个 NOP 指令代替。由系统控制单元（SCU）控制系统功率状态的转换。

CPU 内核有一套专用的特殊功能寄存器（CSFR）：

- CPU 状态指示与控制：PSW、CPUCON1、CPUCON2
- 代码访问控制：IP、CSP
- 数据分页控制：DPP0、DPP1、DPP2、DPP3
- 全局 GPR 访问控制：CP
- 系统堆栈访问控制：SP、SPSEG、STKUN、STKOV
- 乘法与除法支持：MDL、MDH、MDC
- 间接寻址偏移量：QR0、QR1、QX0、QX1
- MAC 地址指针：IDX0、IDX1
- MAC 状态指示与控制：MCW、MSW、MAH、MAL、MRW
- ALU 常量支持：ZEROS、ONES
- CPU ID：CPUID

CPU 还使用 CSFR 来访问通用寄存器（GPR）。由于所有 CSFR 都能够由任何可寻址 SFR/CSFR 存储器空间的指令来控制，因此无需特殊的系统控制指令。

不过，为了确保处理器操作正确，还必须设定一些约束，限制用户对某些 CSFR 的访问。例如，绝对禁止直接访问指令指针（CSP，IP），只能通过分支指令间接修改这些寄存器。寄存器 PSW、SP 和 MDC 不仅能由用户修改，还能在正常指令处理过程中由 CPU 修改。

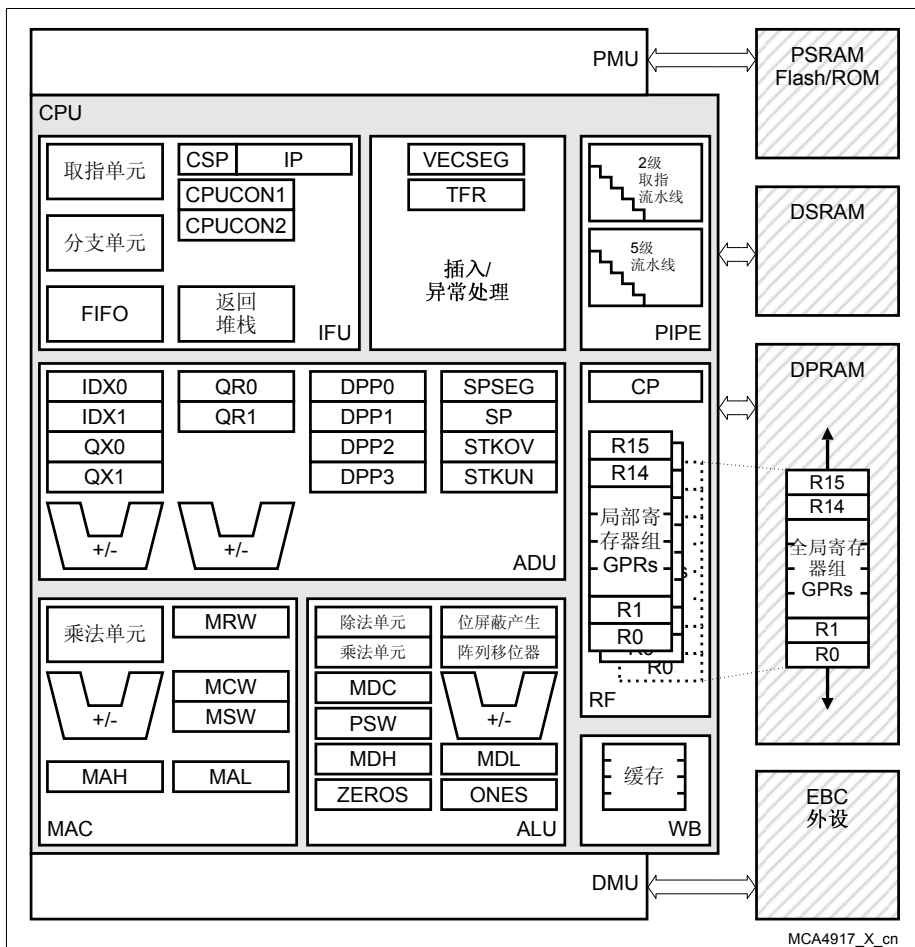
*注：若对同一个 CSFR 的软件写请求和硬件修改同时发生，软件写请求占优。*

所有 CSFR 都可以按字操作、或字节操作（某些寄存器甚至可位操作）。从双字节 CSFR 中读取一个字节是安全的操作；对 CSFR 中的一个字节进行写操作时，另外一个未被寻址的字节将被清零。

**注意：***CSFR 的保留位不能由用户修改，其读取值始终为 0。如果用户需要对保留位进行字或字节操作，必须在 CSFR 保留位中写入 0，以便与将来的版本兼容。*

## 5.1 CPU 的组成

CPU 的良好性能得益于诸多单元的协调工作（见**图 5-1**），这些单元都针对各自功能进行了优化设计。**预取指单元**和**分支单元**向流水线提供数据，使 CPU 因读取指令而引起的停滞时间最短。**寻址单元（ADU）**支持多种复杂的寻址模式，从而避免使用附加指令。**算术逻辑单元（ALU）**和**乘累加单元（MAC）**能处理不同长度的数据，并执行复杂的运算。三个**存储器接口**和**写缓存（WB）**使得由数据传送而引起的 CPU 停滞时间最短。



**图 5-1 CPU 框图**



一般情况下，指令会通过 7 级流水线（见**章节 5.3**），可分为下列两组：

- **2 级取指流水线** — 接收来自程序存储器的指令，并存储到指令 FIFO 中。取指流水线可被旁路。
- **5 级处理流水线** — 执行来自取指阶段的每一条指令。

由于经过流水线的每一级至少需要一个时钟周期且取指流水线阶段可被旁路，因此，任何孤立的指令都至少需要 5 个时钟周期才能完成。不过由于流水线能并行（即同时）处理多达 5 条指令（有分支指令的情况下可以处理 6 条指令），因此复位之后流水线一旦被填满，大部分指令看起来都像是在一个时钟周期内完成。

流水线提高了一段时间内的指令平均吞吐量。

## **5.2 指令读取和程序流控制**

指令读取单元（IFU）对指令进行预取和预处理，从而提供连续的指令流。IFU 能够通过 64 位宽的总线从程序管理单元（PMU）中至少同时读取两条指令。预取出来的指令存储在指令 FIFO 中。

分支指令的预处理能够预测指令流。当 CPU 正在执行从指令 FIFO 中读取的指令时，IFU 开始从预测的目标地址（PMU）中读取一条新指令。在执行指令期间，下一条指令已经被缓存在 FIFO 之中。因此，对 CPU 而言，不存在指令访问的延迟时间。即使执行非顺序指令，通常 IFU 也能够提供连续的指令流。IFU 包括两级流水线：预取指阶段和取指阶段。

在预取指令阶段，分支指令检测和预测逻辑对第一个指令缓存中（最多能保存 6 条指令）的最多 3 条预取指令进行分析。如果检测到一条分支指令，IFU 会按预测规则从 PMU 中取下一条指令。经过分析之后，最多可有 3 条指令保存在第二个指令缓存中（最多能存储 3 条指令）中。第二个指令缓存中的指令将送入取指阶段。

一旦出现预测指令流不正确的情况，取指流水线被旁路，以减少死循环时间。

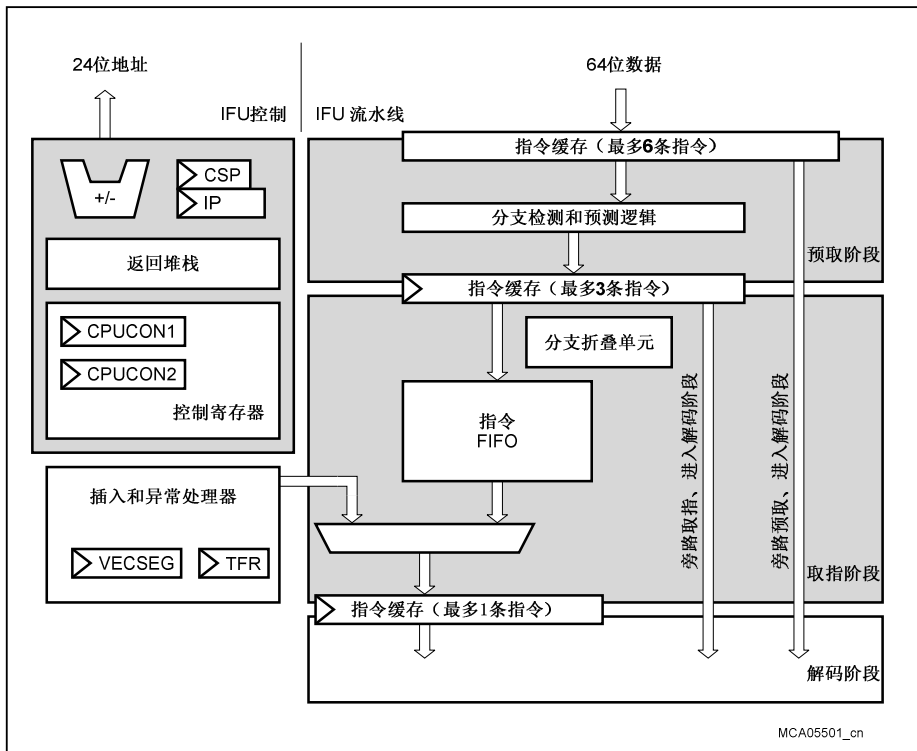


图 5-2 IFU 框图

在取指阶段，预取的指令存储在指令 FIFO 中。分支折叠单元 (BFU) 并行处理分支指令和前一条指令。为了实现这一功能，BFU 对分支指令进行预处理并重新格式化：BFU 首先确定（计算）绝对目标地址，然后将该地址与分支条件、分支属性位组合之后，和前一条指令存放在 FIFO 的同一级中，目标地址也可用于预取下一条指令。

在流水线处理阶段，这两条指令（分支指令和前一条指令）再从 FIFO 中取出，并行执行。如果指令流预测不正确（或 FIFO 为空），IFU 的两级流水线可以被旁路。

*注：被错误预测的指令流，其流水线操作将在后续章节中予以描述。*

### 5.2.1 分支检测和分支预测规则

分支检测单元对指令进行预处理，并对检测到的分支指令进行分类。根据分支指令的类型，分支预测单元采用以下规则预测程序流。

**表 5-1 分支分类和预测规则**

分支指令分类	指令	预测规则（假设）
段间的分支指令	JMPS seg, caddr CALLS seg, caddr	始终执行分支指令
分支预测可编程的分支指令	JMPA- xcc, caddr JMPA+ xcc, caddr CALLA- xcc, caddr CALLA+ xcc, caddr	通过指令长字的位 8（‘a’）位由用户决定 <sup>1)</sup> ： ...+: 跳转（a = 0） ...-: 不跳转（a = 1）
间接分支指令	JMPI cc, [Rw] CALLI cc, [Rw]	无条件：跳转 有条件：不跳转
带条件码的相对分支指令	JMPR cc, rel	无条件或向后跳转：跳转 有条件向前跳转：不跳转
无条件码的相对分支指令	CALLR rel	始终执行分支指令
带位条件的分支指令	JB(C) bitaddr, rel JNB(S) bitaddr, rel	向后跳转：跳转 向前跳转：不跳转
返回指令	RET, RETP RETS, RETI	始终执行分支指令

1) 对一般的 JMPA 和 CALLA 指令，该位也可根据跳转条件由汇编器自动置位/清除。

### 5.2.2 零周期跳转

“**零周期跳转**”是 XE16xyM 特有的特性，得益于复杂的流水线结构，零周期跳转得以实现。

有些情况下，该特性可实现“零时间”的跳转。事实上，跳转指令“依附”到前一条指令上，两条指令如同一条指令一样通过流水线。只有当跳转指令不需用到前一条指令所使用的任何流水线资源时，才可能实现零周期跳转。下面给出基本规则：

- 跳转指令不能依附于另一条跳转指令，因为两条跳转指令不能共享流水线资源“目标 IP”。

- 因为基本上任何前一条指令都可能访问存储器，如果跳转指令需要访问存储器，则不能在零周期内执行。

上面给出了实现零周期跳转的基本条件。但是否能够真正实现零周期跳转还不能可靠预测：这还取决于正确的指令序列、程序存储器的速度等因素。

总结如下：

- 只有当前一条指令**不是**分支指令（任何 JMP、CALL 或 RET）时，**JMPA、JMPR 和 JMPS 指令才能够**转换成零周期。
- 如果执行零周期跳转，事实上该跳转指令的地址不会分配给指令指针。

*注：切勿在满足上述两个条件的零周期跳转指令上设置 IP 断点。否则，调试模块非常可能会丢失该断点。*

### 5.2.3 Atomic 和 Extend 指令

Atomic 和 Extend 指令（ATOMIC、EXTR、EXTP、EXTS、EXTPR、EXTSR）禁止标准和 PEC 中断以及 A 类强制中断，直到紧随其后的指令序列完成为止。指令数可在 1 到 4 之间变化，编码为 2 位常数域 **#irang2**，取值范围从 0 到 3。在该指令序列期间，EXTended 指令还改变寻址机制（见指令描述）。

ATOMIC 和 EXTended 指令立即变为有效，因此无需附加 NOP 指令。需要多个周期或因执行指令而保持状态的所有指令都被认为是一条指令。ATOMIC 和 EXTened 指令可和任意指令类型一起使用。

如果紧跟在 ATOMIC 序列之后的分支指令被零周期执行，那么分支指令也是 ATOMIC 序列的一部分。如果分支指令不是 ATOMIC 序列的一部分，则不应当将其附加到 ATOMIC 序列上，因此可在其间插入 NOP 指令。

*注：如果在 ATOMIC 或 EXTended 序列期间发生 B 类强制中断，那么序列被终止，中断锁定被移除，在执行强制中断程序之前，标准条件被恢复。从强制中断程序返回之后，被终止序列的其余指令在标准条件之下运行。*

*注：当使用嵌套 ATOMIC 和 EXTended 指令时，只有一个计数器控制序列的长度，即在一个序列内发出的 ATOMIC 或 EXTened 指令将会用新指令的值重载计数器。*

### 5.3 指令处理流水线

XE16xyM 采用 5 级流水执行每条指令。所有指令都要依次通过指令处理流水线的每一阶段。下面列出指令处理流水线以及 2 级取指流水线的描述：

**第 1 阶段 → 预取指令：**该阶段按照预测的顺序从 PMU 预取指令。分支检测单元对这些指令进行预处理，检测出分支指令。预测逻辑决定是否执行这些分支/跳转指令。

**第 2 阶段 → 取指令：**根据分支预测规则，计算出要读取的下一条指令的指针。为了实现分支指令的零周期执行，分支折叠单元对检测到的分支指令进行预处理，并将其与前一条指令组合。预取的指令保存在指令 FIFO 中。与此同时，从指令 FIFO 中取出的指令送入指令流水线进行处理。

**第 3 阶段 → 解码：**指令被解码，如有需要，从寄存器文件中读取用于间接寻址的 GPR。

**第 4 阶段 → 寻址：**计算所有操作数的地址。访问系统堆栈时，SP（堆栈指针）自动递增或者递减。

**第 5 阶段 → 存储：**读取所有需要的操作数。

**第 6 阶段 → 执行：**对先前读取的操作数进行 ALU 或 MAC 运算，并更新状态标志。对 CPU-SFR 执行写操作，用作间接寻址指针的 GPR 自动递增/递减。

**第 7 阶段 → 回写：**所有的外部操作数和内部 DPRAM 中剩余的操作数被写回。内部 SRAM 中的操作数写入回写缓存中。

对于超过一个 CPU 时钟周期的指令，内部会产生一些特定的插入指令为其提供时间。它们被自动插入到流水线的解码阶段，然后像标准指令一样经过流水线的其余阶段。还可以利用指令插入来处理程序中断、PEC 传送和调试操作。尽管实际上不需要注意这些内部插入指令，但它们的确有助于说明流水线的操作。

带宽的限制（不同阶段访问相同的资源）以及指令之间数据的相依性都会降低 CPU（流水线）的性能。XE16xyM 的 CPU 具有专用硬件来检测并解决各种类型的数据相依性问题。后续章节将对一些数据相依问题详细说明。

由于最多可以同时处理五条不同的指令，因此 XE16xyM 提供附加的专用硬件来解决可能存在于不同流水线阶段之间的数据相依问题。该附加硬件支持操作数读/写值的“前送”，可以用时间上最优、且不损失性能的方式解决大多数可能的冲突（例如总线的复用）。因此在多数情况下用户不需要注意流水线。然而，在一些情况下，程序员需要注意流水线。

#### 5.3.1 访问 IO 区

对 XE16xyM 存储器空间的 IO 区进行读写访问会导致流水线的特殊行为，因此对寄存器位于该区的外设器件进行操作时，用户需要谨慎处理。

需要考虑外设器件寄存器的以下典型特性：

- 写访问一个外设寄存器时，可能导致任何一个（或者多个）外设寄存器的内容被修改。
- 读访问一个外设寄存器时，可能导致同一个寄存器的内容被修改（如读串行通道的缓存）。

通过以下流水线来处理上面的情况：

### 读操作之前执行写操作

如果流水线中的指令为一个 IO 区写操作紧随其后一个 IO 区读操作，那么读操作被延迟执行（保持在流水线的存储阶段），直至写操作已通过流水线的回写阶段。因此，写操作将始终安排在读操作之前完成。

**注意：因为附加的系统延迟，不能保证在读取目标寄存器之前，写操作会生效。**

附加系统延迟由总线系统累积或由外设本身引起。由于存在附加读延迟和写延迟不同的情况，读操作可能会超过写操作。不过，由于片上延迟是相似的，当使用片外外设通过 EXTBUS（外部总线）分配 IO 区时，编程人员必须特别注意这一点。

### 防止写操作被缓存

对 IO 区的写访问不会被缓存到回写缓存中。

## 5.3.2 流水线冲突

下面的例子描述特殊情况下的流水线行为，并给出通过指令重排的方式优化性能所需要遵循的规则。

*注：XE16xyM 具有完全互锁的流水线，这意味着流水线冲突不会引起任何故障。仅仅是因为要提高性能的原因才需要进行指令重排。*

### 5.3.2.1 使用通用寄存器

GPR（通用寄存器）是 CPU 的工作寄存器，使用 GPR 的指令之间可能存在很多数据相依性。XE16xyM 使用一个高速五端口的寄存器文件防止带宽冲突，并使用专用硬件来检测和解决数据相依性。通过特殊的前送总线将 GPR 值从流水线的阶段送至另一个阶段。大多数情况下，即使存在数据相依性，仍然可以无任何延迟的执行指令。

Conflict\_GPRs\_Resolved:

$I_n$	ADD R0, R1	； 计算 R0 的新值
$I_{n+1}$	ADD R3, R0	； 再次使用 R0
$I_{n+2}$	ADD R6, R0	； 再次使用 R0
$I_{n+3}$	ADD R6, R1	； 再次使用 R6

**表 5-2            解决使用 GPR 引起的流水线数据相依性**

阶段	$T_n$	$T_{n+1}$	$T_{n+2}$	$T_{n+3}$	$T_{n+4}^{1)}$	$T_{n+5}^{2)}$
解码	$I_n = \text{ADD}$ R0, R1	$I_{n+1} = \text{ADD}$ R3, R0	$I_{n+2} = \text{ADD}$ R6, R0	$I_{n+3} = \text{ADD}$ R6, R1	$I_{n+4}$	$I_{n+5}$
寻址	$I_{n-1}$	$I_n = \text{ADD}$ R0, R1	$I_{n+1} = \text{ADD}$ R3, R0	$I_{n+2} = \text{ADD}$ R6, R0	$I_{n+3} = \text{ADD}$ R6, R1	$I_{n+4}$
存储	$I_{n-2}$	$I_{n-1}$	$I_n = \text{ADD}$ R0, R1	$I_{n+1} = \text{ADD}$ R3, <b>R0</b>	$I_{n+2} = \text{ADD}$ R6, <b>R0</b>	$I_{n+3} = \text{ADD}$ <b>R6</b> , R1
执行	$I_{n-3}$	$I_{n-2}$	$I_{n-1}$	$I_n = \text{ADD}$ <b>R0</b> , R1	$I_{n+1} = \text{ADD}$ R3, R0	$I_{n+2} = \text{ADD}$ <b>R6</b> , R0
回写	$I_{n-4}$	$I_{n-3}$	$I_{n-2}$	$I_{n-1}$	$I_n = \text{ADD}$ <b>R0</b> , R1	$I_{n+1} = \text{ADD}$ R3, R0

1) 将 R0 从回写阶段前送至存储阶段。

2) 将 R6 从执行阶段前送至存储阶段。

不过，如果 GPR 被用于间接寻址，在解码阶段就已经需要使用地址指针（即 GPR）。这种情况下，指令一直停滞在寻址阶段，直到执行完 ALU 操作且将运算结果前送至寻址阶段。

Conflict\_GPRs\_Pointer\_Stall:

```

In      ADD R0,R1      ; 计算 R0 的新值
In+1    MOV R3,[R0]    ; 使用 R0 作为地址指针
In+2    ADD R6,R0
In+3    ADD R6,R1
    
```

**表 5-3            GPR 用作指针时引起的流水线数据相依性（停滞）**

阶段	T <sub>n</sub>	T <sub>n+1</sub>	T <sub>n+2</sub> <sup>1)</sup>	T <sub>n+3</sub> <sup>2)</sup>	T <sub>n+4</sub>	T <sub>n+5</sub>
解码	I <sub>n</sub> = ADD R0, R1	I <sub>n+1</sub> = MOV R3, [R0]	I <sub>n+2</sub>	I <sub>n+2</sub>	I <sub>n+2</sub>	I <sub>n+3</sub>
寻址	I <sub>n-1</sub>	I <sub>n</sub> = ADD R0, R1	I <sub>n+1</sub> = MOV R3, [R0]	I <sub>n+1</sub> = MOV R3, [R0]	I <sub>n+1</sub> = MOV R3, [R0]	I <sub>n+2</sub>
存储	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD R0, R1	—	—	I <sub>n+1</sub> = MOV R3, [R0]
执行	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD R0, R1	—	—
回写	I <sub>n-4</sub>	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD R0, R1	—

1)    R0 的新值还不可用。

2)    R0 从执行阶段前送至寻址阶段（下一周期）。



为了避免指令停滞，可以插入一个多周期指令或者两个单周期指令。这些插入的指令绝对不能更新用于间接寻址的 GPR。

Conflict\_GPRs\_Pointer\_NoStall :

$I_n$	ADD R0, R1	; 计算 R0 的新值
$I_{n+1}$	ADD R6, R0	; R0 未更新，只被读取
$I_{n+2}$	ADD R6, R1	
$I_{n+3}$	MOV R3, [R0]	; 使用 R0 作为地址指针

**表 5-4          GPR 用作指针时引起的流水线数据相依性（不停滞）**

阶段	$T_n$	$T_{n+1}$	$T_{n+2}$	$T_{n+3}$ <sup>1)</sup>	$T_{n+4}$	$T_{n+5}$
解码	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R6, R0}$	$I_{n+2} = \text{ADD R6, R1}$	$I_{n+3} = \text{MOV R3, [R0]}$	$I_{n+4}$	$I_{n+5}$
寻址	$I_{n-1}$	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R6, R0}$	$I_{n+2} = \text{ADD R6, R1}$	$I_{n+3} = \text{MOV R3, [R0]}$	$I_{n+4}$
存储	$I_{n-2}$	$I_{n-1}$	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R6, R0}$	$I_{n+2} = \text{ADD R6, R1}$	$I_{n+3} = \text{MOV R3, [R0]}$
执行	$I_{n-3}$	$I_{n-2}$	$I_{n-1}$	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R6, R0}$	$I_{n+2} = \text{ADD R6, R1}$
回写	$I_{n-4}$	$I_{n-3}$	$I_{n-2}$	$I_{n-1}$	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R6, R0}$

1) R0 从执行阶段前送至寻址阶段（下一周期）。

### 5.3.2.2 使用间接寻址模式

读访问使用间接寻址模式时，地址产生单元采用一种推测寻址机制。在对地址解码之前，根据历史记录表，选择从哪个存储器区域（DPRAM、DSRAM 等）读取数据。该历史记录表为每个 GPR 提供了一个入口，其中保存着最近一次通过相应 GPR 寻址的存储区的信息。如果对存储区预测错误，必须重新开始进行读访问。

建议用户在使用 GPR 进行间接寻址时，应使其始终指向同一个存储器区域。如果更新后的 GPR 指向不同的存储器区域，那么下次的读操作将会访问错误的存储器区域，从而必须重新进行读操作，这会导致流水线停滞。

Conflict\_GPRs\_Pointer\_WrongHistory:

```

In      ADD R3, [R0]          ; 例如 R0 指向 DPRAM
In+1    MOV R0, R4
...
Ii      MOV DPPX, ...        ; 改变 DPPx
...
Im      ADD R6, [R0]          ; 例如 R0 现在指向 SRAM
Im+1    MOV R6, R1
    
```

**表 5-5 使用指针引起的流水线数据相依性（有效推测）**

阶段	T <sub>n</sub>	T <sub>n+1</sub>	T <sub>n+2</sub>	T <sub>n+3</sub>	T <sub>n+4</sub>	T <sub>n+5</sub>
解码	I <sub>n</sub> = ADD R3, [R0]	I <sub>n+1</sub> = MOV R0, R4	I <sub>n+2</sub>	I <sub>n+3</sub>	I <sub>n+4</sub>	I <sub>n+5</sub>
寻址	I <sub>n-1</sub>	I <sub>n</sub> = ADD R3, [R0]	I <sub>n+1</sub> = MOV R0, R4	I <sub>n+2</sub>	I <sub>n+3</sub>	I <sub>n+4</sub>
存储	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD R3, [R0]	I <sub>n+1</sub> = MOV R0, R4	I <sub>n+2</sub>	I <sub>n+3</sub>
执行	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD R3, [R0]	I <sub>n+1</sub> = MOV R0, R4	I <sub>n+2</sub>
回写	I <sub>n-4</sub>	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD R3, [R0]	I <sub>n+1</sub> = MOV R0, R4

**表 5-6            使用指针引起的流水线数据相依性（无效推测）**

阶段	$T_m$	$T_{m+1}$	$T_{m+2}^{1)}$	$T_{m+3}$	$T_{m+4}$	$T_{m+5}$
解码	$I_m = \text{ADD R6, [R0]}$	$I_{m+1} = \text{MOV R6, R1}$	$I_{m+1} = \text{MOV R6, R1}$	$I_{m+2}$	$I_{m+3}$	$I_{m+4}$
寻址	$I_{m-1}$	$I_m = \text{ADD R6, [R0]}$	$I_m = \text{ADD R6, [R0]}$	$I_{m+1} = \text{MOV R6, R1}$	$I_{m+2}$	$I_{m+3}$
存储	$I_{m-2}$	$I_{m-1}$	—	$I_m = \text{ADD R6, [R0]}$	$I_{m+1} = \text{MOV R6, R1}$	$I_{m+2}$
执行	$I_{m-3}$	$I_{m-2}$	$I_{m-1}$	—	$I_m = \text{ADD R6, [R0]}$	$I_{m+1} = \text{MOV R6, R1}$
写回	$I_{m-4}$	$I_{m-3}$	$I_{m-2}$	$I_{m-1}$	—	$I_m = \text{ADD R6, [R0]}$

1) 因为历史记录错误（目标区域已经改变），必须重复访问地址 [R0]。

### 5.3.2.3    存储器带宽引起的流水线冲突

如果流水线中的指令同时访问同一存储器区域时，可能会出现存储器带宽冲突。XE16xyM 使用一种特殊的访问机制最大程度的减少冲突。CPU 的 DPRAM 有两个独立的读/写端口，能够无延迟、并行进行读和写操作。对 DSRAM 的写操作可以先保存在回写缓存中，直至完成读操作为止。

除 CoXXX 指令以外的所有指令，每个周期只能读一个存储器操作数。因为 DPRAM 有两个独立的读/写端口，读操作和写操作之间不会出现冲突。只有其它的流水线停滞条件才会产生 DPRAM 带宽冲突。DPRAM 是一个同步流水式存储器。寻址阶段的目标地址有效后开始执行读访问，在存储阶段将读取的数据送出。如果在存储阶段，存储器读访问停滞、处于寻址阶段的随后指令又试图开始新的存储器读操作，那么该读访问必须同样被延迟。但是，该冲突被已有的流水线停滞所掩盖。

CoXXX 指令是唯一能够在一个周期读取两个存储器操作数的指令。如果三个操作数都在 DPRAM 中，那么两个读操作和一个挂起的写操作之间会出现带宽冲突。在执行滤波器程序时，这种情况对性能影响特别重要。其中的一个操作数应该放在 DSRAM 中，以确保 CoXXX 指令的单周期正确执行。

Conflict\_DPRAM\_Bandwidth:

```

In      ADD op1, R1
In+1    ADD R6, R0
In+2    CoMAC [IDX0], [R0]
In+3    MOV R3, [R0]
    
```

**表 5-7 存储器（DPRAM）带宽冲突引起的流水线数据相依性**

阶段	T <sub>n</sub>	T <sub>n+1</sub>	T <sub>n+2</sub>	T <sub>n+3</sub>	T <sub>n+4</sub> <sup>1)</sup>	T <sub>n+5</sub>
解码	I <sub>n</sub> = ADD op1, R1	I <sub>n+1</sub> = ADD R6, R0	I <sub>n+2</sub> = CoMAC ...	I <sub>n+3</sub> = MOV R3, [R0]	I <sub>n+4</sub>	I <sub>n+4</sub>
寻址	I <sub>n-1</sub>	I <sub>n</sub> = ADD op1, R1	I <sub>n+1</sub> = ADD R6, R0	I <sub>n+2</sub> = CoMAC ...	I <sub>n+3</sub> = MOV R3, [R0]	I <sub>n+3</sub> = MOV R3, [R0]
存储	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD op1, R1	I <sub>n+1</sub> = ADD R6, R0	I <sub>n+2</sub> = CoMAC ...	I <sub>n+2</sub> = CoMAC ...
执行	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD op1, R1	I <sub>n+1</sub> = ADD R6, R0	—
回写	I <sub>n-4</sub>	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD op1, R1	I <sub>n+1</sub> = ADD R6, R0

1) COMAC 指令由于存储器带宽冲突而停滞。

DSRAM 是一个单端口读/写存储器。为了减少带宽冲突，XE16xyM 使用回写缓存。它有三个数据入口。仅在回写缓存已满、且同时发生读写访问的情况下，当某个缓存入口被回写时，才必须停滞读访问。

Conflict\_DSRAM\_Bandwidth:

```

In      ADD op1, R1
In+1    ADD R6, R0
In+2    ADD R6, op2
In+3    MOV R3, R2
    
```

**表 5-8 存储器（DSRAM）带宽冲突引起的流水线数据相依性**

阶段	T <sub>n</sub>	T <sub>n+1</sub>	T <sub>n+2</sub>	T <sub>n+3</sub>	T <sub>n+4</sub> <sup>1)</sup>	T <sub>n+5</sub>
解码	I <sub>n</sub> = ADD op1, R1	I <sub>n+1</sub> = ADD R6, R0	I <sub>n+2</sub> = ADD R6, op2	I <sub>n+3</sub> = MOV R3, R2	I <sub>n+4</sub>	I <sub>n+4</sub>
寻址	I <sub>n-1</sub>	I <sub>n</sub> = ADD op1, R1	I <sub>n+1</sub> = ADD R6, R0	I <sub>n+2</sub> = ADD R6, op2	I <sub>n+3</sub> = MOV R3, R2	I <sub>n+3</sub> = MOV R3, R2
存储	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD op1, R1	I <sub>n+1</sub> = ADD R6, R0	I <sub>n+2</sub> = ADD R6, op2	I <sub>n+2</sub> = ADD R6, op2
执行	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD op1, R1	I <sub>n+1</sub> = ADD R6, R0	—
回写	I <sub>n-4</sub>	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = ADD op1, R1	I <sub>n+1</sub> = ADD R6, R0
回写缓存	充满	充满	充满	充满	充满	充满

1) ADD R6, op2 指令由于存储器带宽冲突而停滞。

### 5.3.2.4 CPU-SFR 更新引起的流水线冲突

CPU-SFR 控制 CPU 的功能和行为。对 CSFR 的修改和更新会影响流水线中的指令流。因此，用户必须特别留意保证流水线中的指令使用正确的 CSFR 值。在流水线执行阶段后期更新 CSFR。此时，如果没有冲突检测，处于解码、寻址、存储阶段的指令将仍旧使用未更新的寄存器值。为了保证操作正确，CPU 会检测冲突情况并停滞流水线。为了提高性能，CPU 区分不同类型的 CPU-SFR。根据以下规则对指令进行重新排序，可以改善流水线中指令流的性能。

共有三种类型的 CPU-SFR：

- 不产生流水线冲突的 CSFR（ONES, ZEROS, MCW）

- 在执行阶段后期更新的 CSFR 结果寄存器，会导致一个周期的停滞
- 影响整个 CPU 或流水线的 CSFR，会导致流水线取消

### CSFR 结果寄存器

ALU 和 MAC 单元的 CSFR 结果寄存器 MDH、MDL、MSW、MAH、MAL 和 MRW 在流水线执行阶段的后期被更新。如果处于存储阶段的指令 (CoSTORE 除外) 要访问这些寄存器，寄存器值不能被前送。该指令必须在存储阶段停滞一个周期。

Conflict\_CSFR\_Update\_Stall:

```

In      MUL R0,R1
In+1    MOV R6,MDL
In+2    ADD R6,R1
In+3    MOV R3,[R0]
```

表 5-9 结果 CSFR 引起的流水线数据相依性 (停滞)

阶段	T <sub>n</sub>	T <sub>n+1</sub>	T <sub>n+2</sub>	T <sub>n+3</sub> <sup>1)</sup>	T <sub>n+4</sub>	T <sub>n+5</sub>
解码	I <sub>n</sub> = MUL R0, R1	I <sub>n+1</sub> = MOV R6, MDL	I <sub>n+2</sub> = ADD R6, R1	I <sub>n+3</sub> = MOV R3, [R0]	I <sub>n+3</sub> = MOV R3, [R0]	I <sub>n+4</sub>
寻址	I <sub>n-1</sub>	I <sub>n</sub> = MUL R0, R1	I <sub>n+1</sub> = MOV R6, MDL	I <sub>n+2</sub> = ADD R6, R1	I <sub>n+2</sub> = ADD R6, R1	I <sub>n+3</sub> = MOV R3, [R0]
存储	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = MUL R0, R1	I <sub>n+1</sub> = MOV R6, MDL	I <sub>n+1</sub> = MOV R6, MDL	I <sub>n+2</sub> = ADD R6, R1
执行	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = MUL R0, R1	—	I <sub>n+1</sub> = MOV R6, MDL
回写	I <sub>n-4</sub>	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = MUL R0, R1	—

1) 此处不能读 MDL。

通过重排指令，可以将不占用该资源的指令填充到流水线的空隙周期中。

Conflict\_CSFR\_Update\_Resolved:

```

In          MUL  R0, R1
In+1        MOV  R3, [R0]
In+2        MOV  R6, MDL
In+3        ADD  R6, R1
    
```

**表 5-10      结果 CSFR 引起的流水线数据相依性（无停滞）**

阶段	T <sub>n</sub>	T <sub>n+1</sub>	T <sub>n+2</sub>	T <sub>n+3</sub>	T <sub>n+4</sub> <sup>1)</sup>	T <sub>n+5</sub>
解码	I <sub>n</sub> = MUL R0, R1	I <sub>n+1</sub> = MOV R3, [R0]	I <sub>n+2</sub> = MOV R6, MDL	I <sub>n+3</sub> = ADD R6, R1	I <sub>n+4</sub>	I <sub>n+5</sub>
寻址	I <sub>n-1</sub>	I <sub>n</sub> = MUL R0, R1	I <sub>n+1</sub> = MOV R3, [R0]	I <sub>n+2</sub> = MOV R6, MDL	I <sub>n+3</sub> = ADD R6, R1	I <sub>n+4</sub>
存储	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = MUL R0, R1	I <sub>n+1</sub> = MOV R3, [R0]	I <sub>n+2</sub> = MOV R6, MDL	I <sub>n+3</sub> = ADD R6, R1
执行	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = MUL R0, R1	I <sub>n+1</sub> = MOV R3, [R0]	I <sub>n+2</sub> = MOV R6, MDL
回写	I <sub>n-4</sub>	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = MUL R0, R1	I <sub>n+1</sub> = MOV R3, [R0]

1) 此时MDL可被读取，无需停滞。

## 影响整个 CPU 的 CSFR

某些 CSFR 会影响整个 CPU 或存储阶段之前流水线的操作。CPU-SFR CPUCON1/2、CP、SP、STKUN、STKOV、VECSEG、TFR 和 PSW 被**显性**修改时，会影响整个 CPU 的功能；CPU-SFR IDX0、IDX1、QX1、QX0、DPP0、DPP1、DPP2 和 DPP3 在被**显性**修改时，只影响解码、寻址和存储阶段。这种情况下，流水线行为取决于用于修改 CSFR 的指令和寻址模式。

在使用“POP CSFR”或使用 reg, #data16 寻址模式的指令修改这些 CSFR 寄存器时，在初始化阶段通过一种特殊机制来提高性能。

为便于更进一步解释，修改 CSFR 的指令可以被称作“CSFR 修改指令”。

“CSFR 修改指令”进入处理流水线后，在解码阶段被检测到。下面列出的指令在解码阶段被保持（所有其它指令不被保持）：

- 使用长寻址模式的指令（mem）
- 使用间接寻址模式的指令（[R<sub>w</sub>], [R<sub>w</sub>+]...），JMPI 和 CALLI 除外
- ENWDT、DISWDT、EINIT
- 所有的 CoXXX 指令

如果 CPUCON1/2、CP、SP、STKUN、STKOV、VECSEG、TFR 或 PSW 被修改，并且“CSFR 修改指令”进入执行阶段，那么流水线将被取消。这种修改将影响整个流水线和指令预取，所以需要彻底取消和重启机制来确保正确的指令流。如果 IDX0、IDX1、QX1、QX0、DPP0、DPP1、DPP2 或 DPP3 被修改，只有解码、寻址和存储阶段受到影响，则不需取消流水线。这种修改不影响已经进入寻址、存储阶段的指令，因为这些指令当时不使用该资源。其它类型的指令在解码阶段被保持，直到 CSFR 被修改为止。

下面的例子给出了流水线停滞的一种情况。指令“MOV IDX1, #12”修改 CSFR，其后的指令“MOV R6, mem”将在解码阶段被保持，直到 IDX1 寄存器被更新。第二个例子给出一个优化的初始化程序。

Conflict\_Canceling:

```

In          MOV  IDX1, #12
In+1        MOV  R6, mem
In+2        ADD  R6, R1
In+3        MOV  R3, [R0]
```



**表 5-11 控制 CSFR 引起的流水线数据相依性（取消流水线）**

阶段	$T_n$	$T_{n+1}$	$T_{n+2}$	$T_{n+3}$	$T_{n+4}$	$T_{n+5}$
解码	$I_n = \text{MOV } \text{IDX1}, \#12$	$I_{n+1} = \text{MOV } \text{R6}, \text{mem}$	$I_{n+1} = \text{MOV } \text{R6}, \text{mem}$	$I_{n+1} = \text{MOV } \text{R6}, \text{mem}$	$I_{n+1} = \text{MOV } \text{R6}, \text{mem}$	$I_{n+2} = \text{ADD } \text{R6}, \text{R1}$
寻址	$I_{n-1}$	$I_n = \text{MOV } \text{IDX1}, \#12$	—	—	—	$I_{n+1} = \text{MOV } \text{R6}, \text{mem}$
存储	$I_{n-2}$	$I_{n-1}$	$I_n = \text{MOV } \text{IDX1}, \#12$	—	—	—
执行	$I_{n-3}$	$I_{n-2}$	$I_{n-1}$	$I_n = \text{MOV } \text{IDX1}, \#12$	—	—
回写	$I_{n-4}$	$I_{n-3}$	$I_{n-2}$	$I_{n-1}$	$I_n = \text{MOV } \text{IDX1}, \#12$	—

Conflict\_Canceling\_Optimized:

```

In      MOV  IDX1, #12
In+1    MOV  MAH, #23
In+2    MOV  MAL, #25
In+3    MOV  R3, #08

```

**表 5-12 控制 CSFR 引起的流水线数据相依性（经过优化）**

阶段	$T_n$	$T_{n+1}$	$T_{n+2}$	$T_{n+3}$	$T_{n+4}$	$T_{n+5}$
解码	$I_n = \text{MOV } \text{IDX1}, \#12$	$I_{n+1} = \text{MOV } \text{MAH}, \#23$	$I_{n+2} = \text{MOV } \text{MAL}, \#25$	$I_{n+3} = \text{MOV } \text{R3}, \#08$	$I_{n+4}$	$I_{n+5}$
寻址	$I_{n-1}$	$I_n = \text{MOV } \text{IDX1}, \#12$	$I_{n+1} = \text{MOV } \text{MAH}, \#23$	$I_{n+2} = \text{MOV } \text{MAL}, \#25$	$I_{n+3} = \text{MOV } \text{R3}, \#08$	$I_{n+4}$
存储	$I_{n-2}$	$I_{n-1}$	$I_n = \text{MOV } \text{IDX1}, \#12$	$I_{n+1} = \text{MOV } \text{MAH}, \#23$	$I_{n+2} = \text{MOV } \text{MAL}, \#25$	$I_{n+3} = \text{MOV } \text{R3}, \#08$
执行	$I_{n-3}$	$I_{n-2}$	$I_{n-1}$	$I_n = \text{MOV } \text{IDX1}, \#12$	$I_{n+1} = \text{MOV } \text{MAH}, \#23$	$I_{n+2} = \text{MOV } \text{MAL}, \#25$
回写	$I_{n-4}$	$I_{n-3}$	$I_{n-2}$	$I_{n-1}$	$I_n = \text{MOV } \text{IDX1}, \#12$	$I_{n+1} = \text{MOV } \text{MAH}, \#23$

对于所有其它修改这类 CSFR 的指令，采用简单的停滞和取消机制来保证正确的指令流。

在流水线的存储阶段，用户对这类 CSFR 进行显性写操作可被检测到。随后的指令在寻址和解码阶段被停滞。如果指令进入指令执行阶段，那么整个流水线和 IFU 中的指令 FIFO 被清空，指令流需要彻底重新开始。

Conflict\_Canceling\_Completely:

```

In      MOV PSW, R4
In+1    MOV R6, R1
In+2    ADD R6, R1
In+3    MOV R3, [R0]
```

**表 5-13 控制 CSFR 引起的流水线数据相依性（全部取消）**

阶段	T <sub>n+1</sub>	T <sub>n+2</sub>	T <sub>n+3</sub>	T <sub>n+4</sub>	T <sub>n+5</sub>	T <sub>n+6</sub>
解码	I <sub>n+1</sub> = MOV R6, R1	I <sub>n+2</sub> = ADD R6, R1	I <sub>n+2</sub> = ADD R6, R1	—	—	I <sub>n+1</sub> = MOV R6, R1
寻址	I <sub>n</sub> = MOV PSW, R4	I <sub>n+1</sub> = MOV R6, R1	I <sub>n+1</sub> = MOV R6, R1	—	—	—
存储	I <sub>n-1</sub>	I <sub>n</sub> = MOV PSW, R4	—	—	—	—
执行	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = MOV PSW, R4	—	—	—
回写	I <sub>n-3</sub>	I <sub>n-2</sub>	I <sub>n-1</sub>	I <sub>n</sub> = MOV PSW, R4	—	—

## 5.4 CPU 配置寄存器

CPU 配置寄存器用于选择 XE16xyM CPU 内核的一般特性和功能。通常只能由启动软件来设置这些寄存器，在运行期间应用软件不能改变这些寄存器。

*注：执行 EINIT 指令之后，CPU 配置寄存器受到寄存器安全机制的保护。*

### CPUCON1

#### CPU 控制寄存器 1

**SFR (FE18H/0CH)**

**复位值：0007<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									VECSC	WDTCTL	SGTDIS	INTSCXT	BP	ZCJ	
r									rw	rw	rw	rw	rw	rw	

符号	位序号	类型	功能描述
<b>0</b>	[15:7]	r	<b>保留</b> 读操作返回 0；应写入 0。
<b>VECSC</b>	[6:5]	rw	<b>中断向量间距</b> 00 <sub>B</sub> 中断向量之间的间距为 2 个字 <sup>1)</sup> 01 <sub>B</sub> 中断向量之间的间距为 4 个字 10 <sub>B</sub> 中断向量之间的间距为 8 个字 11 <sub>B</sub> 中断向量之间的间距为 16 个字
<b>WDTCTL</b>	4	rw	<b>看门狗定时器配置</b> 0 <sub>B</sub> 仅在初始化结束 <sup>2)</sup> 后才可以执行 DISWDT 1 <sub>B</sub> 始终可以执行 DISWDT/ENWDT（增强 WDT 模式）
<b>SGTDIS</b>	3	rw	<b>分段禁止/使能控制</b> 0 <sub>B</sub> 分段使能 1 <sub>B</sub> 分段禁止
<b>INTSCXT</b>	2	rw	<b>切换上下文的中断使能</b> 0 <sub>B</sub> 切换上下文不可被中断 1 <sub>B</sub> 切换上下文可被中断
<b>BP</b>	1	rw	<b>分支预测单元使能</b>



符号	位序号	类型	功能描述
			0 <sub>B</sub> 禁止分支预测 1 <sub>B</sub> 使能分支预测
ZCJ	0	rw	零周期跳转使能 0 <sub>B</sub> 禁止零周期跳转功能 1 <sub>B</sub> 使能零周期跳转功能

1) 缺省值（2 个字）与 C166 系列体系结构中的中断向量间距兼容。

2) DISWDT（在执行 EINIT 指令以后）和 ENWDT 指令被内部转换成 NOP 指令。

## CPUCON2

### CPU 控制寄存器 2

SFR（FE1A<sub>H</sub>/0D<sub>H</sub>）

复位值：8FBB<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFODEPTH			FIFOFED	BYP PF	BYP F	1	STE N	LFIC	OV RUN	RET ST	FAS TBL	1	SL		
rw			rw	rw	rw	r	rw	rw	rw	rw	rw	r	rw		

符号	位序号	类型	功能描述
FIFODEPTH	[15:12]	rw	<b>FIFO 缓存设置</b> 0 <sub>H</sub> 没有 FIFO 1 <sub>H</sub> 1 级 FIFO ... 8 <sub>H</sub> 8 级 FIFO 9 <sub>H</sub> 保留 ... F <sub>H</sub> 保留
FIFOFED	[11:10]	rw	<b>FIFO 填充配置</b> 00 <sub>B</sub> 禁止 FIFO 01 <sub>B</sub> 每个周期 FIFO 中至多填充 1 条指令 10 <sub>B</sub> 每个周期 FIFO 中至多填充 2 条指令 11 <sub>B</sub> 每个周期 FIFO 中至多填充 3 条指令
BYP PF	9	rw	<b>预取指令旁路控制</b> 0 <sub>B</sub> 从预取指令到解码之间的旁路通道被禁止

符号	位序号	类型	功能描述
			1 <sub>B</sub> 从预取指令到解码之间的旁路通道被使能
<b>BYPF</b>	8	rw	<b>取指令旁路控制</b> 0 <sub>B</sub> 从取指令到解码之间的旁路通道被禁止 1 <sub>B</sub> 从取指令到解码之间的旁路通道被使能
<b>1</b>	7	r	<b>保留</b> 读操作返回 1；应写入 1。
<b>STEN</b>	6	rw	<b>停滞指令使能</b> （用于调试） 0 <sub>B</sub> 禁用停滞指令 1 <sub>B</sub> 使能停滞指令（详见下面的例子）
<b>LFIC</b>	5	rw	<b>线性跟随器指令缓存</b> 0 <sub>B</sub> 禁用线性跟随器指令高度缓存 1 <sub>B</sub> 使能线性跟随器指令高速缓存
<b>OVRUN</b>	4	rw	<b>流水线控制</b> 0 <sub>B</sub> 不允许过载流水线空隙周期 1 <sub>B</sub> 允许过载流水线空隙周期
<b>RETST</b>	3	rw	<b>返回堆栈使能</b> 0 <sub>B</sub> 禁止返回堆栈 1 <sub>B</sub> 使能返回堆栈
<b>FASTBL</b>	2	rw	<b>禁止块传送快速插入</b> 0 <sub>B</sub> 禁止直接插入 1 <sub>B</sub> 使能直接插入
<b>1</b>	1	r	<b>保留</b> 读操作返回 1；应写入 1。
<b>SL</b>	0	rw	<b>短循环模式使能</b> 0 <sub>B</sub> 禁止短循环模式 1 <sub>B</sub> 使能短循环模式

注：只有在文档（如勘误表）中明确说明的前提下，才能修改该寄存器。

## 5.5 通用寄存器的使用

CPU 有三个寄存器组，每个寄存器组由 16 个寄存器 R0、R1、R2、...R15 组成，通常称之为通用寄存器（GPR）。访问这些寄存器只占用一个 CPU 周期。GPR 是算术逻辑运算单元的工作寄存器，也可用作间接寻址的地址指针。

通过 5 端口寄存器文件可实现对寄存器组的高速访问，以满足 CPU 性能要求。寄存器文件分成三个独立的物理寄存器组，对应**两种类型**：

- **两个局部寄存器组**，是寄存器文件的组成部分。
- **一个全局寄存器组**，映射到存储器，缓存在寄存器文件中。

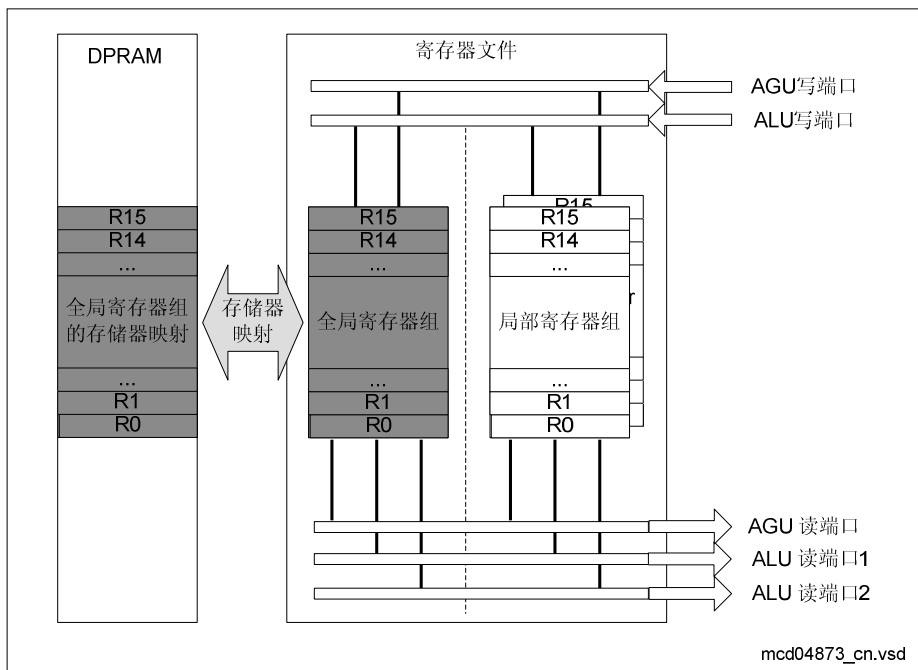


图 5-3 寄存器文件

## 中央处理器（CPU）

由 PSW 寄存器中的位域 BANK 选择激活哪一个物理寄存器组。用户可通过显性修改 PSW 的指令改变所选寄存器组；或通过 RETI 指令、中断或硬件强制中断隐性改变所选寄存器组。发生中断时，通过中断控制器（ITC）中的寄存器 BNKSELx 来选择寄存器组。硬件强制中断始终使用全局寄存器组。

局部寄存器组由专用的物理寄存器组成，全局寄存器组表现为一个缓存。多个全局寄存器组可映射到内部 DPRAM 中。每个寄存器组占用 16 个字长的连续地址区段。上下文指针（CP）寄存器决定当前所选寄存器组的基地址。为保证访问速度，DPRAM 中的 GPR 被缓存到 5 端口寄存器文件中（同一时刻，只能缓存一个存储器映射 GPR 组）。如果激活全局寄存器组，在执行后面的指令前，将对缓存进行验证。验证之后，所有对 GPR 的后续访问会转向全局寄存器组。

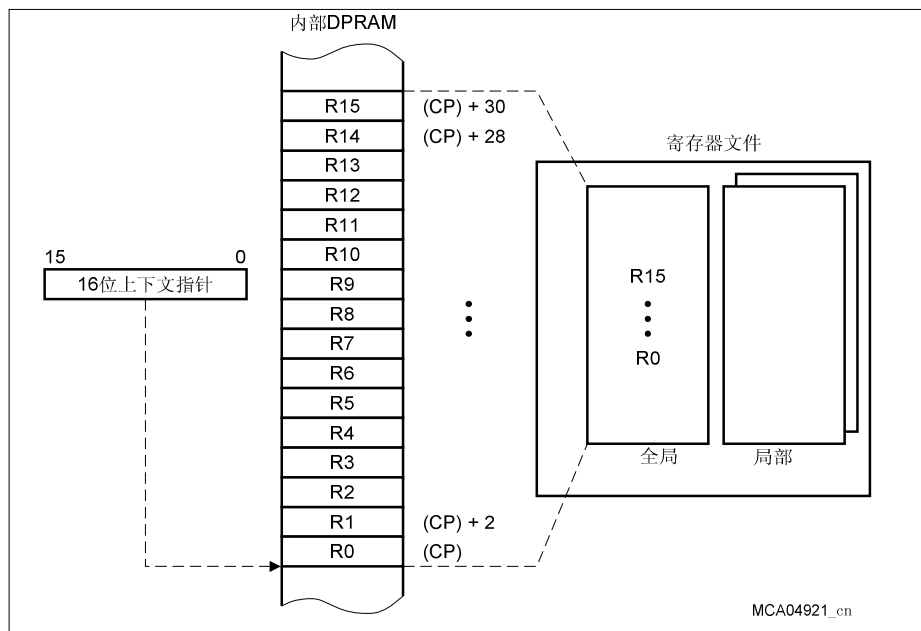


图 5-4 通过寄存器 CP 选择寄存器组

### 5.5.1 GPR 寻址模式

GPR 是工作寄存器，会被频繁访问。因此有三种可能的方式来访问寄存器组：

- **GPR 短寻址**（助记符：Rw 或 Rb）
- **寄存器短寻址**（助记符：reg 或 bitoff）
- **存储器长寻址**（助记符：mem），只适用于全局寄存器组

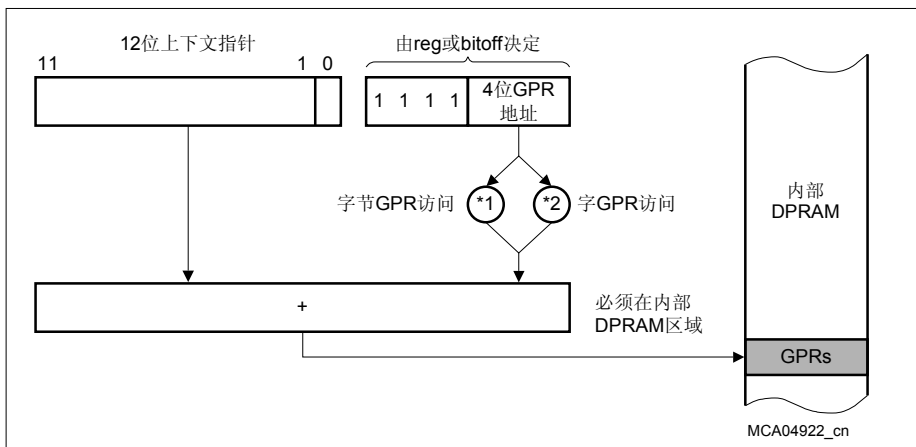
**GPR 短寻址**指定当前寄存器组（通过 BANK 位域选择）内的偏移地址。4 位 GPR 短寻址可以访问所有 16 个寄存器，2 位 GPR 短寻址（部分指令采用）则只能访问最低的 4 个寄存器。

根据寻址方式是寄存器字寻址（Rw）还是字节寻址（Rb），在物理访问寄存器组之前将 GPR 短地址乘 2（Rw），或者不乘 2（Rb）。从而 GPR 字访问和字节访问都可采用这种寻址方式。

*注：用作间接寻址指针的 GPR 总是采用字操作的方式。*

寻址局部寄存器组，结果偏移量可以直接使用；寻址全局寄存器组，结果偏移量要与寄存器 CP 的值逻辑相加。CP 指向当前全局寄存器组的基地址。（见图 5-5）。

**8 位寄存器短寻址**所用的地址范围从 F0<sub>H</sub> 到 FF<sub>H</sub>。与 4 位 GPR 短寻址方式相同，利用低四位来寻址，忽略高四位。GPR 物理地址的计算和 4 位短 GPR 寻址相同。对于 GPR 位操作，采用相同的方法来计算 GPR 的字地址。一个字中被访问位元的位置由单独的附加 4 位值来确定。



**图 5-5 寄存器短寻址模式下 CP 的使用**



**24 位存储器寻址**可直接用于访问位于 DPRAM 中的 GPR（不适用于局部寄存器组）。进行存储器读访问时，命中检测逻辑会检测被寻址的存储器内容是否缓存到全局寄存器组中。如果缓存命中，则读操作转向全局寄存器组，使用从缓存中读取的数据，而忽略从存储器中读取的数据。这会导致一个 CPU 周期的延时（MOV R4, mem [CP ≤ mem ≤ CP+31]）。进行存储器写访问时，命中检测逻辑会提前决定是否缓存命中。不过，地址转换需要额外一个 CPU 周期。该值直接被写入全局寄存器组而无需额外的延迟（MOV mem, R4）。

注：使用 24 位 GPR 寻址模式时，读和写访问都需要额外增加一个周期。

表 5-14 访问 GPR 的寻址模式

字寄存器 <sup>1)</sup>		字节寄存器		短地址 <sup>2)</sup>		
名称	存储器地址 <sup>3)</sup>	名称	存储器地址 <sup>3)</sup>	8 位	4 位	2 位
R0	(CP) + 0	RL0	(CP) + 0	F0 <sub>H</sub>	0 <sub>H</sub>	0 <sub>H</sub>
R1	(CP) + 2	RH0	(CP) + 1	F1 <sub>H</sub>	1 <sub>H</sub>	1 <sub>H</sub>
R2	(CP) + 4	RL1	(CP) + 2	F2 <sub>H</sub>	2 <sub>H</sub>	2 <sub>H</sub>
R3	(CP) + 6	RH1	(CP) + 3	F3 <sub>H</sub>	3 <sub>H</sub>	3 <sub>H</sub>
R4	(CP) + 8	RL2	(CP) + 4	F4 <sub>H</sub>	4 <sub>H</sub>	---
R5	(CP) + 10	RH2	(CP) + 5	F5 <sub>H</sub>	5 <sub>H</sub>	---
R6	(CP) + 12	RL3	(CP) + 6	F6 <sub>H</sub>	6 <sub>H</sub>	---
R7	(CP) + 14	RH3	(CP) + 7	F7 <sub>H</sub>	7 <sub>H</sub>	---
R8	(CP) + 16	RL4	(CP) + 8	F8 <sub>H</sub>	8 <sub>H</sub>	---
R9	(CP) + 18	RH4	(CP) + 9	F9 <sub>H</sub>	9 <sub>H</sub>	---
R10	(CP) + 20	RL5	(CP) + 10	FA <sub>H</sub>	A <sub>H</sub>	---
R11	(CP) + 22	RH5	(CP) + 11	FB <sub>H</sub>	B <sub>H</sub>	---
R12	(CP) + 24	RL6	(CP) + 12	FC <sub>H</sub>	C <sub>H</sub>	---
R13	(CP) + 26	RH6	(CP) + 13	FD <sub>H</sub>	D <sub>H</sub>	---
R14	(CP) + 28	RL7	(CP) + 14	FE <sub>H</sub>	E <sub>H</sub>	---
R15	(CP) + 30	RH7	(CP) + 15	FF <sub>H</sub>	F <sub>H</sub>	---

- 1) 前 8 个 GPR (R7...R0) 也可按字节访问。对 GPR 一个字节的写操作不会影响该 GPR 的另一字节。
- 2) 短寻址模式适用于所有寄存器组。
- 3) 长寻址模式仅适用于映射到存储器的全局寄存器组。

## 5.5.2 上下文切换

当操作系统的任务调度程序激活一个新任务，或者调用、终止某个中断服务子程序时，剩余任务的工作上下文（即寄存器）必须被保存，而新任务的工作上下文必须被恢复。可以用两种方式改变 CPU 上下文：

- 切换选用的寄存器组
- 切换全局寄存器的上下文

### 切换选用的物理寄存器组

通过更新寄存器 PSW 位域 BANK，可以立即切换工作的寄存器组。可在当前存储器映射的全局寄存器组（缓存在全局寄存器组中）（BANK = 00<sub>B</sub>）、局部寄存器组 1（BANK = 10<sub>B</sub>）和局部寄存器组 2（BANK = 11<sub>B</sub>）之间进行切换。

进行中断服务时，通过更新中断控制器中的寄存器 BNKSELx 位域 BANK，可以自动执行寄存器组切换。通过执行 RETI 指令，位域 BANK 将被自动恢复，上下文将被切换回原寄存器组。

还可以通过修改位域 BANK，在寄存器文件的三个物理寄存器组之间进行切换。由于流水线存在数据相依性，因此，若用户对寄存器 PSW 进行显性修改，必须取消流水线。

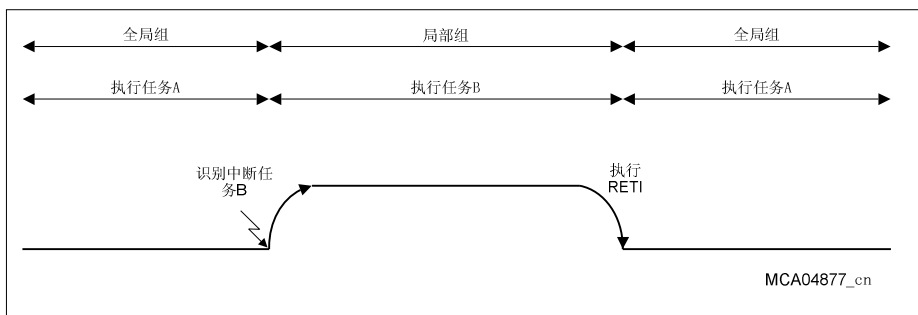


图 5-6 通过改变物理寄存器组进行上下文切换

切换至局部寄存器组之后，可以立即使用新的寄存器组；切换至全局寄存器组之后，在执行任何后续指令前，缓存的存储器映射 GPR 必须有效。如果此时全局寄存器组无效（比如上下文切换过程被中断），将自动启动缓存验证过程。

## 全局寄存器组的上下文切换

通过改变存储器映射 GPR 组的基地址，切换全局寄存器组的内容。基地址由上下文指针 (CP) 给出。

完成 CP 更新之后，状态机开始存储全局寄存器组的旧内容并加载新内容。存储和加载算法需要 19 个 CPU 周期：其中缓存验证过程占用了 16 个周期；为了避免由完成验证过程而引起流水线冲突，指令执行需要再停滞 3 个周期。上下文切换过程包括两个阶段：

- **存储阶段：**通过插入 8 条 STORE 指令，将全局寄存器组<sup>1)</sup>的内容存回到 DPRAM 中。最后一条 STORE 指令之后，全局寄存器组的内容无效。
- **加载阶段：**通过插入 8 条 LOAD 指令，将新的上下文加载到全局寄存器组。最后一条 LOAD 指令之后，全局寄存器组的内容有效。

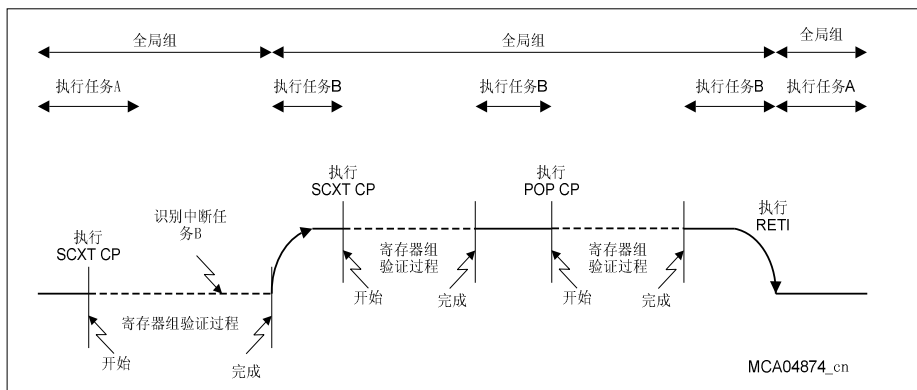
直到全局寄存器组再次有效之后才能开始执行代码。在验证过程中可能出现硬件中断。如何完成验证过程取决于该中断所选择的寄存器组类型：

- 如果中断也使用全局寄存器组，在执行中断服务程序之前，需要完成验证过程（见图 5-7）。
- 如果中断使用局部寄存器组，验证过程被中断，立即执行中断服务程序（见图 5-8）。切换回到全局寄存器组之后，完成验证过程：
  - 如果中断发生在存储阶段，整个验证过程从头开始执行。
  - 如果中断发生在加载阶段，只重复加载阶段的操作。

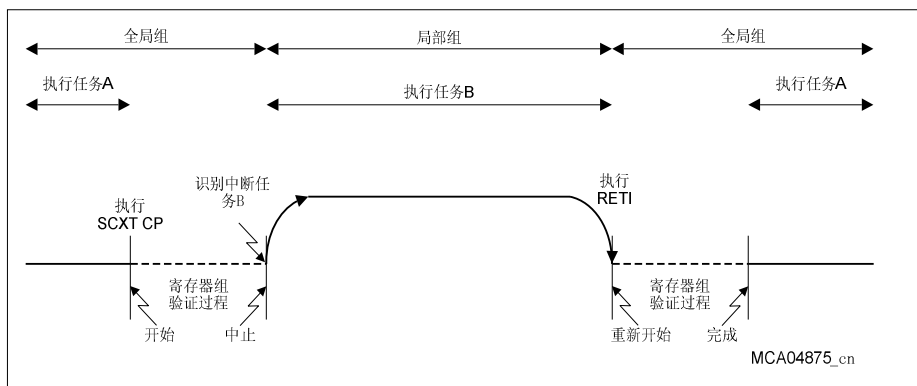
如果使用局部寄存器组的中断服务程序（图 5-9 中的任务 B）再次被一个使用全局寄存器组的中断（任务 C）打断，在执行任务 C 的代码之前，必须先完成已被挂起的验证过程。这意味着任务 A 的验证过程不会影响任务 B 的中断延迟时间，而会影响任务 C 的中断延迟时间。

*注：如果任务 C 立即中断任务 A，将首先完成任务 A 的寄存器组验证过程。最坏情况的中断延迟在两种情况下相同（见图 5-7 和图 5-9）。*

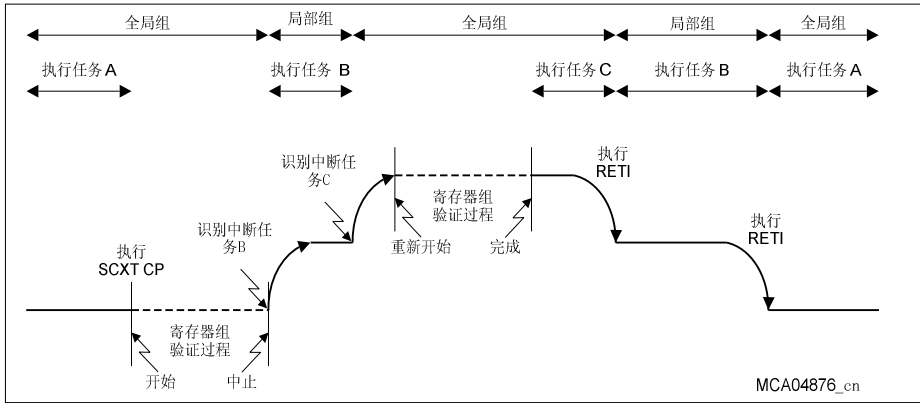
1) 即使应用仅使用该寄存器组的一部分，在上下文切换的存储阶段，要求将整个寄存器组写入到 DPRAM 中。寄存器组一定不能位于地址 FDE0<sub>H</sub> 之上，否则存储阶段将改写 SFR（从 FE00<sub>H</sub> 开始）。



**图 5-7 验证过程被使用全局寄存器组的中断程序中中断**



**图 5-8 验证过程被使用局部寄存器组的中断程序中中断**



**图 5-9 验证过程被使用局部寄存器组和全局寄存器组的中断程序中中断**

### 5.5.2.1 上下文指针 (CP)

该寄存器不可位寻址，用来选择当前的全局寄存器组上下文。任何能够修改SFR的指令都能更新CP寄存器。

#### CP

上下文指针

SFR (FE10<sub>H</sub>/08<sub>H</sub>)

复位值: FC00<sub>H</sub>

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

符号	位序号	类型	功能描述
1	15, 14, 13, 12	r	<b>CP 的固定部分</b> 读操作返回 1。
CP	[11:1]	rw	<b>CP 可修改部分</b> 指定当前全局（映射到存储器）寄存器组 16 位地址的位[11:1]。如果向位 CP[11:9]写入的值 = 000 <sub>B</sub> ，则位 CP[11:10]由硬件设置为 11 <sub>B</sub> 。
0	0	r	<b>CP 的固定部分</b> 读操作返回 0。

中央处理器（CPU）

*注：用户应该确保 CP 值和 GPR 短地址逻辑相加后得到的 GPR 物理地址始终在内部 DPRAM 地址范围内。如果不满足该条件，会出现不可预知的结果。设置 CP（基地址）不能小于内部 DPRAM 的起始地址。不要将 CP 设置为大于 FDE0<sub>H</sub>，否则在存储阶段将改写 SFR（从 FE00<sub>H</sub> 开始）。*

XE16xyM 可使用一条指令切换切换整个存储器映射 GPR 组。切换之后，服务程序在自己独立的上下文内执行。

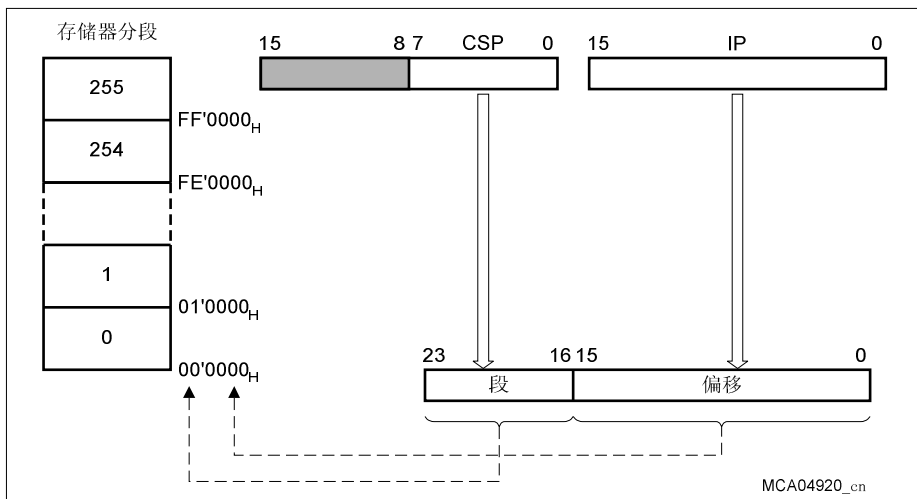
指令“SCXT CP, #New\_Bank”将当前上下文指针（CP）的值压入系统堆栈，并用“New\_Bank”加载 CP，该立即数选定一个新的寄存器组。此时服务程序可以使用它“自己的寄存器”。当服务程序终止时，该存储器寄存器组中内容被保留，即，其内容在下次调用服务程序时依然可用。

在从中断服务程序返回之前（RETI），先前的 CP 从系统堆栈中弹出，切换回先前的寄存器组。

*注：由于内部指令流水执行，对 CP 寄存器的写操作会停滞指令流，直到真正执行完寄存器文件的上下文切换为止。紧随在更新 CP 寄存器指令之后的指令可以使用改变后的 CP 新值。*

## 5.6 代码寻址

XE16xyM 提供了总共 16 MB 的可寻址存储空间，该地址空间分为 256 段，每段 64 KB。使用一个专用 24 位代码地址指针从存储器读取指令。该指针分为两部分：一个 8 位代码段指针 CSP 和一个被称作指令指针（IP）的 16 位偏移指针。CSP 和 IP 级联直接得到 24 位物理存储器地址。



**图 5-10 通过代码段指针和指令指针寻址**

**代码段指针 CSP** 选择当前运行指令所在的代码段。寄存器 CSP 的低 8 位用来从 256 个 64 KB 的代码段中选择一个，高 8 位保留待用。硬件复位值为 0000<sub>H</sub>，但是因为插入的 MOVCSIP 指令，复位之后立即用 VECSEG 寄存器的内容加载 CSP。

*注：寄存器 CSP 为只读类型，不能向其写数据。*

在**存储器分段模式**下（复位后缺省状态），寄存器 CSP 可通过 J MPS 和 C ALLS 指令直接修改，也可通过 R ET S 和 R ET I 指令由堆栈间接修改。

在**存储器不分段模式**下（通过置位寄存器 CPUCON1 的位 S G T D I S 选择），CSP 固定指向用来禁止分段的指令所在的代码段。使用段间调用（C ALLS）或返回（R ET U R N s）指令不可能修改寄存器 CSP。

在处理中断或强制中断时，寄存器 CSP 会自动载入中断向量表的段地址（由寄存器 V EC S E G 确定）。

*注：为了在不分段存储器模式下正确执行中断任务，VECSEG 的内容必须选择和 CSP 当前值相同的段，也就是说，向量表必须位于 CSP 指向的段。*

## CSP

**代码段指针** **SFR（FE08<sub>H</sub>/04<sub>H</sub>）** **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>								<b>SEGNR</b>							
r								rh							

符号	位序号	类型	功能描述
<b>0</b>	[15:8]	r	<b>保留</b> 读操作返回 0；应写入 0。
<b>SEGNR</b>	[7:0]	rh	<b>段编号</b> 指定当前指令所在的段地址。

*注：复位之后，将寄存器 VECSEG 的值自动载入到寄存器 CSP。*

**指令指针IP**决定当前读取指令的16位段内地址，代码段由CSP寄存器选择。寄存器IP未被映射到XE16xyM的地址空间，因此程序员不能直接访问IP。不过，可通过返回指令由堆栈间接修改IP。执行分支指令和取指操作之后，IP被CPU隐性更新。

## IP

**指令指针** **（不可寻址）** **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>IP</b>															<b>0</b>
h															r

符号	位序号	类型	功能描述
<b>IP</b>	[15:1]	h	<b>指令指针</b> 指定当前指令的段内偏移量的位[15:1]。IP 指向当前段<SEGNR>。
<b>0</b>	0	r	<b>IP 的固定部分</b> 读操作返回 0。



## 5.7 数据寻址

地址数据单元（ADU）包含两个独立的算术单元：标准地址产生单元（SAGU）和 DSP 地址产生单元（DAGU），用于产生、计算和更新数据访问地址。ADU 主要执行以下任务：

- 标准地址产生（SAGU）
- DSP 地址产生（DAGU）
- 数据分页（SAGU）
- 堆栈处理（SAGU）

SAGU 支持间接寻址模式的线性算术地址运算，还可产生所有其它短寻址和长寻址模式的地址。

DAGU 包括一套附加的地址指针和偏移寄存器，仅和 CoXXX 指令一起使用。

CPU 提供了多种功能强大的用于字、字节和位数据访问的寻址模式（短、长、间接寻址）。且不同的寻址模式使用不同的格式，具有不同的寻址范围。

### 5.7.1 短寻址模式

短寻址模式允许访问 GPR、SFR 或可位寻址的存储器空间。这些寻址模式通过一个偏移地址（8/4/2 位）和一个隐含的基地址共同确定 24 位物理地址：

**表 5-15 短寻址模式**

助记符	基地址 <sup>1)</sup>	偏移地址	短地址区域	访问范围
Rw	(CP)	$2 \times Rw$	0 ... 15	GPR（字）
Rb	(CP)	$1 \times Rb$	0 ... 15	GPR（字节）
reg	00'FE00 <sub>H</sub>	$2 \times reg$	00 <sub>H</sub> ... EF <sub>H</sub>	SFR（字，低字节）
	00'F000 <sub>H</sub>	$2 \times reg$	00 <sub>H</sub> ... EF <sub>H</sub>	ESFR（字，低字节）
	(CP)	$1 \times (reg \wedge 0F_H)$	F0 <sub>H</sub> ... FF <sub>H</sub>	GPR（字）
	(CP)	$1 \times (reg \wedge 0F_H)$	F0 <sub>H</sub> ... FF <sub>H</sub>	GPR（字节）
bitoff	00'FD00 <sub>H</sub>	$2 \times bitoff$	00 <sub>H</sub> ... 7F <sub>H</sub>	RAM 中可位寻址字
	00'FF00 <sub>H</sub>	$2 \times (bitoff \wedge 7F_H)$	80 <sub>H</sub> ... EF <sub>H</sub>	SFR 中可位寻址字
	00'F100 <sub>H</sub>	$2 \times (bitoff \wedge 7F_H)$	80 <sub>H</sub> ... EF <sub>H</sub>	ESFR 中可位寻址字
	(CP)	$2 \times (bitoff \wedge 0F_H)$	F0 <sub>H</sub> ... FF <sub>H</sub>	GPR 中可位寻址字
bitaddr	可位寻址字 见 bitoff	位元位置	0 ... 15	任意位

1) 对通用寄存器 (GPR) 的访问也可能访问局部寄存器组, 此时不使用 CP。

**物理地址 = 基地址 +  $\Delta \times$  短地址**

*注: 访问字节 GPR,  $\Delta$  等于 1; 访问字 GPR,  $\Delta$  等于 2。*

**Rw, Rb:** 用于直接访问当前工作上下文 (全局寄存器组或局部寄存器组) 中的任意 GPR。指令格式中 'Rw' 和 'Rb' 需要 4 位。全局寄存器组的基地址由寄存器 CP 决定。'Rw' 和 'Rb' 分别给出局部寄存器组内或全局寄存器组内 (以 CP 为基地址) 的 4 位 GPR 字地址和 4 位 GPR 字节地址。

**reg:** 用于直接访问任意 (E)SFR 或当前工作上下文 (全局寄存器组或局部寄存器组) 中的任意 GPR。指令格式中 'reg' 值需要 8 位。短地址范围在 00<sub>H</sub> 和 EF<sub>H</sub> 之间的 'reg' 始终用来指定 (E)SFR 地址。此时, 因子 ' $\Delta$ ' 等于 2, 访问标准 SFR 区时, 基地址为 00' FE00<sub>H</sub>; 访问扩展 ESFR 区时, 基地址为 00' F000<sub>H</sub>。通过 'reg' 访问 ESFR 区之前, 需要用 EXT\*R 指令切换基地址。根据操作码不同, 通过 'reg' 可分别对 SFR 的整个字 (字操作) 或低字节 (字节操作) 进行寻址。请注意: 不能通过 'reg' 寻址模式访问 SFR 的高字节。短地址范围在 F0<sub>H</sub> 和 FF<sub>H</sub> 之间的 'reg' 始终用来指定 GPR 地址。此时, 只使用 'reg' 的低四位来确定 GPR 的物理地址, 因此, 它与 'Rw' 和 'Rb' 寻址模式相同。

**bitoff:** 用于直接访问可位寻址存储空间中的任意字。指令格式中 'bitoff' 需要 8 位。对应不同地址范围的 'bitoff' 选择不同的基地址, 以产生所需物理地址 (见 [表 5-15](#))。通过 'bitoff' 访问 ESFR 区之前, 需要用 EXT\*R 指令切换基地址。

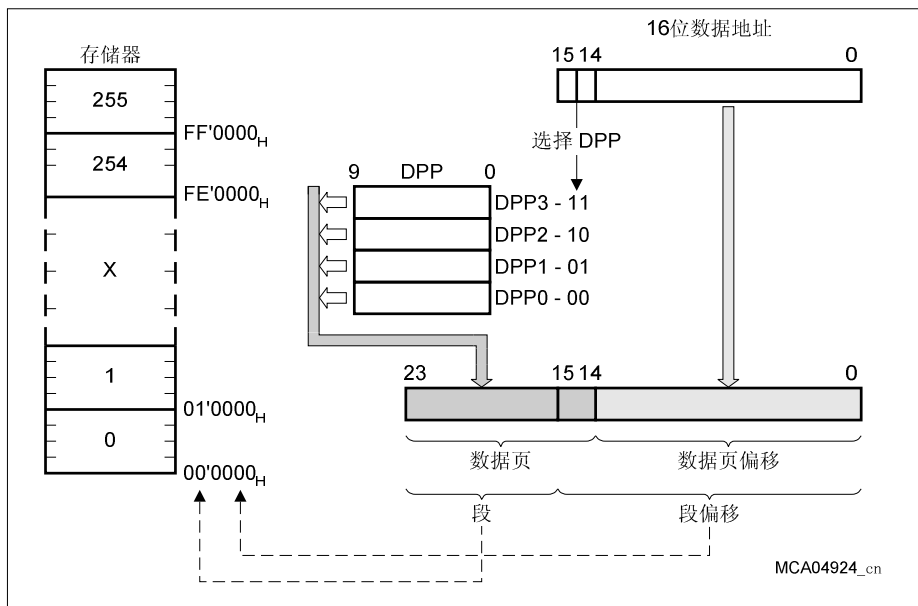
**bitaddr:** 可位寻址存储空间中任意位元的地址均由字地址 (见 'bitoff') 和该字中的位元位置 ('bitpos') 共同决定。因此, 在指令格式中 'bitaddr' 需要 12 位。

### 5.7.2 长寻址模式

长寻址模式指定 24 位地址，因此可访问整个地址空间中的任意字或字节数据。可以采用不同的方式来指定长地址，以产生完整的 24 位地址：

- **使用 4 个数据页指针之一（DPP 寄存器）：**利用 16 位指针中的位 15...14 来选择一个 DPP，利用位 13...0 指定 14 位数据页偏移地址（见图 5-11）。
- **直接选择使用的数据页：**通过前面的 EXTP(R)指令选择数据页，16 位指针中的位 13...0 指定 14 位数据页偏移地址。
- **直接选择使用的段：**通过前面的 EXTS(R)指令选择段地址，16 位指针指定 16 位段偏移地址。

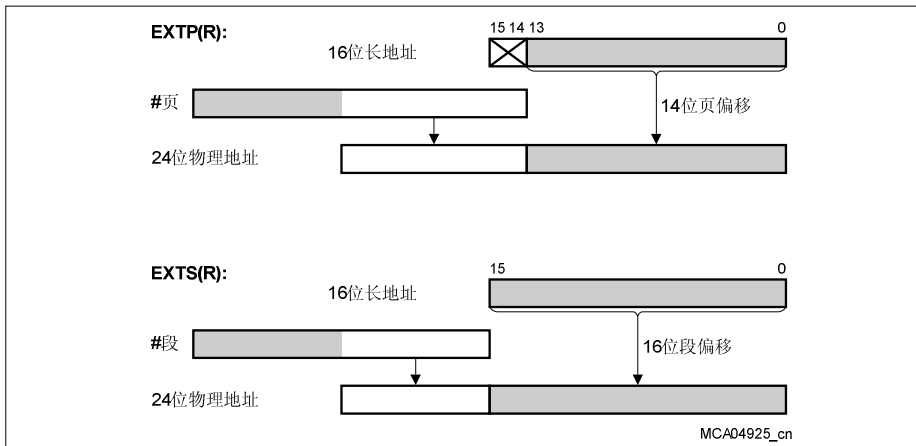
*注：不能执行对奇字节地址的字访问，该操作会引发一个硬件强制中断。*



**图 5-11 数据页指针寻址**



注：由于内部指令流水执行，因此，对 DPPx 寄存器的写操作会停滞指令流，直到 DPP 实际上被更新为止。紧随更新 DPP 寄存器的指令之后的指令可以使用改变后的 DPPx 新值。



**图 5-12      DPP 替代机制**

注：替代页或替代段可由一个常数（#pag, #seg）或通过 GPR 字（Rw）指定。

**表 5-16      长寻址模式**

助记符	基地址 <sup>1)</sup>	偏移量	访问范围
mem	(DPPx)	mem ^ 3FFF <sub>H</sub>	任意字或字节
mem	pag	mem ^ 3FFF <sub>H</sub>	任意字或字节
mem	seg	mem	任意字或字节

1) 代表一个与14位偏移地址串联的10位数据页编号，或一个与16位偏移地址串联的8位段编号。

### 5.7.3 间接寻址模式

可以将间接寻址模式看成是短寻址和长寻址模式的组合。也就是说，16 位“长”指针通过 GPR 字的内容间接提供，GPR 由 4 位短地址直接指定（‘Rw’ = 0 ... 15）。

有些间接寻址模式，在计算 16 位长地址之前，给 GPR 的内容加上一个常量；有些间接寻址模式，能以 2 或 1（对应字或字节）递减或递增间接地址指针（GPR 的内容），或者以偏移地址寄存器 QR0 或 QR1 的值递减或递增间接地址指针。

表 5-17 从间接指针产生物理地址

步骤	执行的动作	计算	备注
1	从短地址计算间接指针的地址（GPR 字）	<b>GPR 地址 =</b> <b>2 × 短地址[+ (CP) ]</b>	见表 5-15
2	根据数据类型（字节操作 $\Delta=1$ 或字操作 $\Delta=2$ ）前减间接指针（‘-Rw’）	<b>(GPR 地址) =</b> <b>(GPR 地址) - <math>\Delta</math></b>	可选步骤，仅在寻址模式需要时执行
3	用一个常数值（‘Rw+const16’）调整指针	<b>指针 =</b> <b>(GPR 地址) + 常数</b>	可选步骤，仅在寻址模式需要时执行
4	利用结果指针计算 24 位物理地址	<b>物理地址 =</b> <b>页/段+指针偏移量</b>	使用 DPP 或页/段替代机制，见表 5-16
5	根据数据类型（字节操作 $\Delta=1$ 或字操作 $\Delta=2$ ），或根据偏移寄存器（ $\Delta=QRx$ ） <sup>1)</sup> 后增/减间接指针（‘Rw ±’）	<b>(GPR 地址) =</b> <b>(GPR 地址) ±<math>\Delta</math></b>	可选步骤，仅在寻址模式需要时执行

1) 后减以及根据 QRx 修改间接指针仅适用于 CoXXX 指令。

注：有些指令只使用 GPR 的最低四个字（R3...R0）作为间接地址指针，这种情况下，GPR 由 2 位短地址指定。



符号	位序号	类型	功能描述
<b>QR</b>	[15:1]	rw	<b>QRx 可修改部分</b> 为间接寻址模式指定 16 位偏移地址 (LSB 始终为 0)。
<b>0</b>	0	r	<b>QRx 的固定部分</b> 读操作返回 0。



### 5.7.4 DSP 寻址模式

除了标准地址产生单元（SAGU），DSP 地址产生单元（DAGU）还提供了一组附加的指针寄存器（IDX0, IDX1）和偏移寄存器（QX0, QX1）。使用指针寄存器 IDX0 和 IDX1，DSP 专用 CoXXX 指令可在一个 CPU 周期内完成。XE16xyM 提供了一个独立的算术单元，可并行完成这些专用指针寄存器的更新和 SAGU 中 GPR 指针的修改。DAGU 只支持使用特殊指针寄存器 IDX0 和 IDX1 的间接寻址模式。

这两个地址指针可用于算术操作，也可用于特殊的 CoMOV 指令。产生 24 位物理地址的方式有所不同：

- 对于 **CoMOV** 指令来说，IDX 指针与 DPP 或选择的数据页/段地址级联，这和长寻址模式相同（总结见图 5-11）。
- 对于**算术 CoXXX** 指令来说，IDX 指针自动扩展成 24 位存储器地址，指向内部 DPRAM 区域，如图 5-13。

#### IDX0

**地址指针** **SFR (FF08<sub>H</sub>/84<sub>H</sub>)** **复位值: 0000<sub>H</sub>**

#### IDX1

**地址指针** **SFR (FF0A<sub>H</sub>/85<sub>H</sub>)** **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>IDX</b>															<b>0</b>
rw															r

符号	位序号	类型	功能描述
<b>IDX</b>	[15:1]	rw	<b>寄存器 IDXx 可修改部分</b> 指定 16 位地址指针。
<b>0</b>	0	r	<b>IDXx 的固定部分</b> 读操作返回 0。

*注：在IDX寄存器初始化期间，指令流可能停滞。正确的操作参见[章节5.3.2.4](#)。*

有些间接寻址模式可在计算 16 位长地址之前进行并行的数据转移操作（示例见图 5-14）。有些间接寻址模式允许以 2、或以偏移地址寄存器 QX0 或 QX1 的值为单位（与 IDX 指针联合使用）递减或递增间接地址指针（IDXx 内容）。

### QX0

偏移寄存器 **ESFR (F000<sub>H</sub>/00<sub>H</sub>)** 复位值: 0000<sub>H</sub>

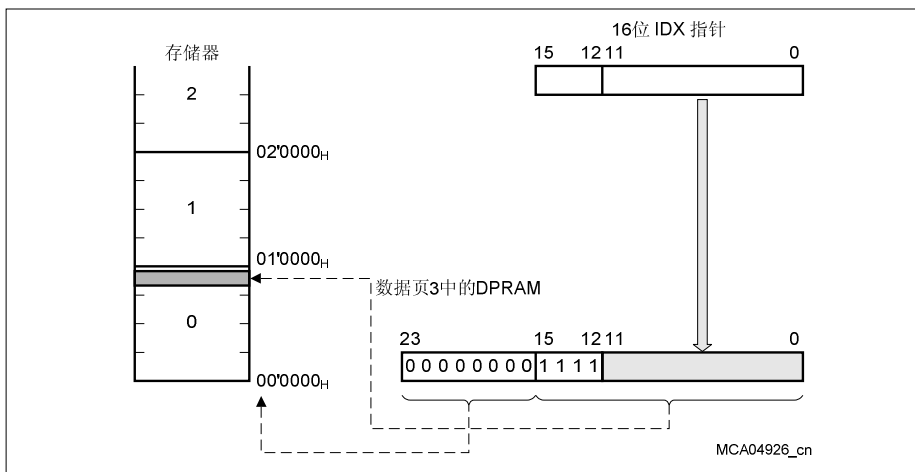
### QX1

偏移寄存器 **ESFR (F002<sub>H</sub>/01<sub>H</sub>)** 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>QX</b>															<b>0</b>
rw															r

符号	位序号	类型	功能描述
<b>QX</b>	[15:1]	rw	寄存器 <b>QXx</b> 可修改部分 为间接寻址模式指定 16 位字偏移地址。
<b>0</b>	0	r	<b>QXx</b> 的固定部分 读操作返回 0。

*注：在QX寄存器初始化期间，指令流可能停滞。正确的操作参见[章节5.3.2.4](#)。*



**图 5-13 算术 MAC 操作和通过 IDX 指针寻址**

**表 5-19      从间接指针 (IDXx) 产生物理地址**

步骤	执行的操作	计算	备注
1	确定所使用的 IDXx 指针	---	-
2	为并行数据转移操作计算中间长地址, 并以 2 ( $\Delta=2$ )或根据偏移寄存器 ( $\Delta=QXx$ ) 递增或递减间接指针 ('IDXx $\pm$ ')	<b>中间地址 =</b> <b>(IDXx 指针) <math>\pm\Delta</math></b>	可选步骤, 仅在指令 CoXXXM 和寻址模式需要时执行
3	计算 16 位长地址	<b>长地址 =</b> <b>(IDXx 指针)</b>	-
4	利用计算结果指针计算 24 位物理地址	<b>物理地址 =</b> <b>页/段 + 指针偏移量</b>	使用 DPP 或页/段替代机制, 见 <b>表 5-16</b> 和 <b>图 5-13</b>
5	以 2 ( $\Delta=2$ )或根据偏移量寄存器 ( $\Delta=QXx$ ), 后递增或递减间接指针 ('IDXx $\pm$ ')	<b>(IDXx 指针) =</b> <b>(IDXx 指针) <math>\pm\Delta</math></b>	可选步骤, 仅在寻址模式需要时执行

XE16xyM提供了下面几种间接寻址模式:

**表 5-20      DSP 寻址模式**

助记符	特殊性
[IDXx]	大多数 CoXXX 指令使用 IDXx (IDX0,IDX1) 作为间接地址指针。
[IDXx+]	访问之后, 间接地址指针自动加 2。
有并行数据转移	对于 CoXXXM 指令, 保存在间接地址指针中的地址自动前减 2 用于并行转移操作, 指针自身不前减。访问之后, 间接地址指针自动加 2。
[IDXx-]	访问之后, 间接地址指针自动减 2。
有并行数据转移	对于 CoXXXM 指令, 保存在间接地址指针中的地址自动前加 2 用于并行转移操作, 指针自身不前加。访问之后, 间接地址指针自

助记符	特殊性
	动减 2。
[IDXx+QXx]	访问之后，间接地址指针自动递增 QXx。
有并行数据转移	对于 CoXXXM 指令，保存在间接地址指针中的地址自动前减 QXx 用于并行转移操作。指针自身不前减。访问之后，间接地址指针自动递增 QXx。
[IDX-QXx]	访问之后，间接地址指针自动递减 QXx。
有并行数据转移	对于 CoXXXM 指令，保存在间接地址指针内的地址自动前加 QXx 用于并行转移操作，指针自身不前加。访问之后，间接地址指针自动递减 QXx。

注：并行数据转移的例子可参见图 5-14。

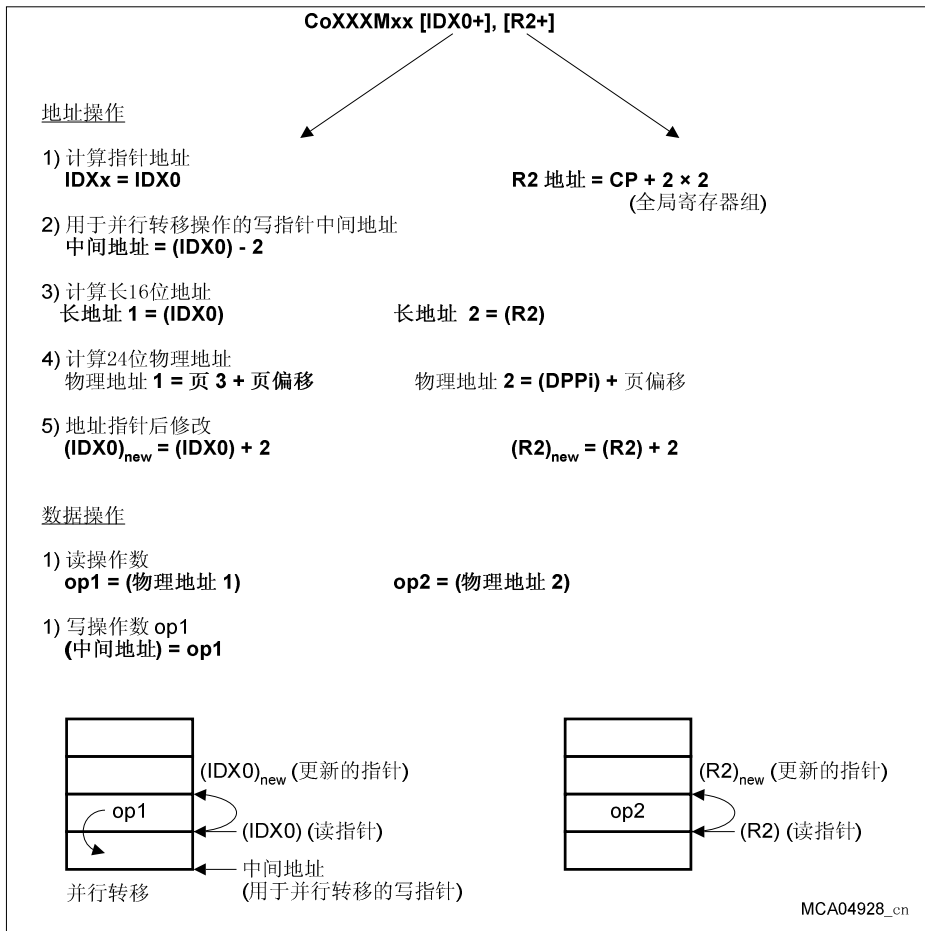
### CoREG 寻址模式

CoSTORE指令使用特殊的CoREG寻址模式，在一次MAC操作之后立即存储MAC单元寄存器。将MAC单元寄存器地址编码在CoSTORE指令格式中，如表 5-21所示。

**表 5-21      CoREG 寻址模式编码**

助记符	寄存器	www:w 位[31:27]编码
MSW	MAC 单元状态字	00000 <sub>B</sub>
MAH	MAC 单元累加器的高位字	00001 <sub>B</sub>
MAS	受限的 MAC 单元累加器高位字	00010 <sub>B</sub>
MAL	MAC 单元累加器的低位字	00100 <sub>B</sub>
MCW	MAC 单元控制字	00101 <sub>B</sub>
MRW	MAC 单元重复字	00110 <sub>B</sub>

图 5-14的例子给出带有并行数据转移操作的CoXXXM指令的复杂操作过程，该操作基于章节 5.7.3（间接寻址模式）和章节 5.7.4（DSP寻址模式）所描述的寻址模式。



**图 5-14 带有并行转移的算术 MAC 操作**

### 5.7.5 系统堆栈

XE16xyM 支持多达 64 KB 的系统堆栈。堆栈可以位于片上存储器或者片外存储器。使用 16 位堆栈指针寄存器（SP）对堆栈段内的 64 KB 进行寻址，堆栈段由堆栈指针段寄存器（SPSEG）选择。还可以用软件实现虚拟堆栈（通常大于 64 KB）。堆栈上溢寄存器 STKOV 和堆栈下溢寄存器 STKUN 支持该机制（说明见下文）。

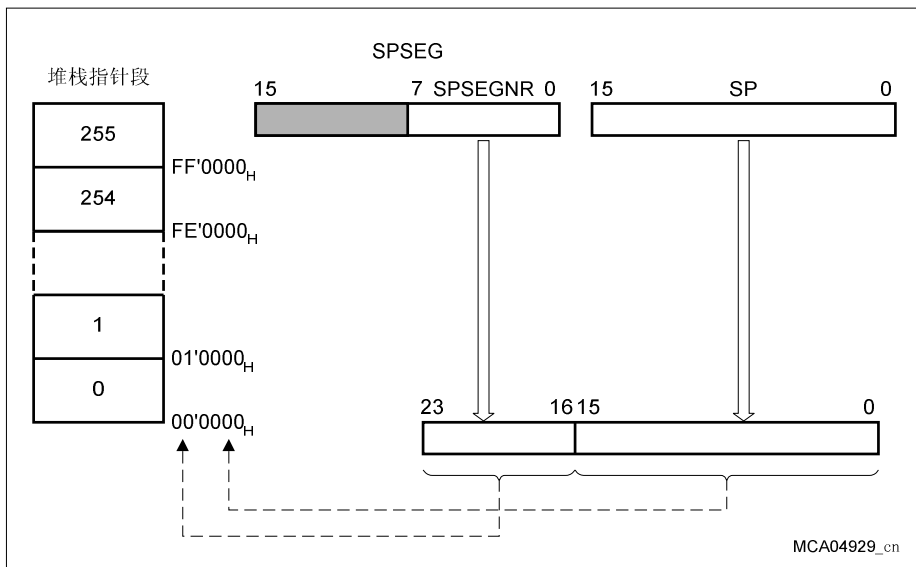
#### 5.7.5.1 堆栈指针寄存器 SP 和 SPSEG

寄存器 SPSEG（不可位寻址）选择当前堆栈所在的段。寄存器 SPSEG 的低 8 位用来从 256 个 64 KB 大小的段中选择一个作为堆栈段，高 8 位保留待用。

堆栈指针 SP（不可位寻址）指向系统堆栈的栈顶（TOS）。数据压栈之前 SP 自动递减；数据出栈后 SP 自动递增。因此，按照从高地址到向低地址的方向生成系统堆栈。

系统堆栈地址由寄存器 SPSEG 的内容与寄存器 SP 的 16 位值共同构成，如图 5-15 所示。

系统堆栈不能跨越 64 KB 的段边界。



**图 5-15 通过堆栈指针寻址**

## SP

堆栈指针

**SFR (FE12<sub>H</sub>/09<sub>H</sub>)**

**复位值: FC00<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>SP</b>															<b>0</b>
rwh															r

符号	位序号	类型	功能描述
<b>SP</b>	[15:1]	rwh	<b>SP 可修改部分</b> 指定 16 位系统堆栈指针的位[15:1]，段内地址。
<b>0</b>	0	r	<b>SP 的固定部分</b> 读操作返回 0。

## SPSEG

堆栈指针段

**SFR (FF0C<sub>H</sub>/86<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>								<b>SPSEGNR</b>							
r								rw							

符号	位序号	类型	功能描述
<b>0</b>	[15:8]	r	<b>保留</b> 读操作返回 0；应写入 0。
<b>SPSEGNR</b>	[7:0]	rw	<b>堆栈指针段编号</b> 指定堆栈所在的段。

*注：可以由任何能够修改 16 位 SFR 的指令更新寄存器 SPSEG 和 SP。由于内部指令流水执行，对 SPSEG 或 SP 的写操作会使指令流停滞，直到寄存器更新完毕。紧随更新 SPSEG 或 SP 的指令之后的指令可以使用改变后的新值。*



### 5.7.5.2 堆栈上溢/下溢指针 STKOV/STKUN

这两个界限寄存器（不可位寻址）用于监视堆栈指针。堆栈指针达到上限或下限时会产生一个强制中断。堆栈指针比较时不需要考虑堆栈指针段寄存器SPSEG。系统堆栈不能跨越64 KB段边界。

每次对SP隐含写操作（该操作会使SP自动递减）之前（指令CALLA、CALLI、CALLR、CALLS、PCALL、TRAP、SCXT或PUSH），STKOV与SP进行比较。如果SP和STKOV相等，则触发堆栈上溢强制中断。

每次对SP隐含读操作（该操作会使SP自动递增）之前（指令RET、RETS、RETP、RETI或POP），STKUN与SP进行比较，如果SP和STKUN相等，则触发堆栈下溢强制中断。

有两种不同的方法来处理堆栈上溢/下溢强制中断：

- **致命错误指示**将堆栈溢出看成系统错误，执行相关的强制中断服务程序。  
在堆栈溢出强制中断情况下，栈底的数据可能已经被执行强制中断服务程序时压栈的状态信息所覆盖。
- **虚拟堆栈控制**可将系统堆栈用作“堆栈缓存”，以支持更大的外部用户堆栈：上溢时清空缓存；下溢时重新填充缓存。

### 堆栈界限控制范围

通过寄存器 STKOV 和 STKUN 可以检测出堆栈指针（SP）被隐含修改后是否超越了规定的堆栈区，从而实现了堆栈界限控制。

如果通过转移指令或算术指令显性的修改堆栈指针，则 SP 不与 STKOV 和 STKUN 进行比较。这种情况下，若修改后的堆栈指针超出规定的堆栈区，即低于 STKOV 或高于 STKUN，将无法检测到堆栈错误。只有当 SP 被隐含修改时，才进行堆栈上溢/下溢检测。

SP 值可能超出允许的 SP 范围，而不触发强制中断。但是，如果由于隐含修改（例如 PUSH 或 POP）SP 值而导致超出允许的界限，则会触发相应的强制中断。

*注：STKOV 和 STKUN 可以由任何能够修改 SFR 的指令进行更新。如果在 ATOMIC/EXT 序列中发生堆栈上溢或下溢，则完成作为序列指令一部分的堆栈操作。在完成所有的 ATOMIC/EXT 序列之后，才发出强制中断。*

## STKOV

堆栈上溢指针

**SFR（FE14<sub>H</sub>/0A<sub>H</sub>）**

**复位值: FA00<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>STKOV</b>															<b>0</b>
rw															r

符号	位序号	类型	功能描述
<b>STKOV</b>	[15:1]	rw	<b>STKOV 的可修改部分</b> 指定系统堆栈下限的段内偏移地址。
<b>0</b>	0	r	<b>STKOV 的固定部分</b> 读操作返回 0。

## STKUN

堆栈下溢指针

**SFR（FE16<sub>H</sub>/0B<sub>H</sub>）**

**复位值: FC00<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>STKUN</b>															<b>0</b>
rw															r

符号	位序号	类型	功能描述
<b>STKUN</b>	[15:1]	rw	<b>STKUN 的可修改部分</b> 指定系统堆栈上限的段内偏移地址。
<b>0</b>	0	r	<b>STKUN 的固定部分</b> 读操作返回 0。

## 5.8 标准数据处理

通过16位ALU执行所有的标准算术、移位和逻辑操作。除实现标准功能以外，XE16xyM的ALU还包括一个位处理单元和一个乘除单元。大多数内部执行模块经过优化设计，用于执行8位或16位数运算。在流水线被填充之后，大多数指令在一个CPU周期内完成。每次ALU操作之后，PSW寄存器的状态标志被自动更新，指示微控制器的当前状态。这些标志可以用作分支转移的特定条件。XE16xyM支持有符号运算和无符号运算，这通过用户可选的分支测试来完成。进入中断或强制中断服务程序时，由CPU自动保存状态标志。另一组标志位表示当前CPU的中断状态。两个独立位（USR0和USR1）用作通用标志。

### PSW

处理器状态字

SFR (FF10<sub>H</sub>/88<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ILVL				IEN	HLD EN PL1	BANK		USR 1	USR 0	PL0	E	Z	V	C	N
rwh				rw	rwh	rwh		rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位序号	类型	功能描述
ILVL	[15:12]	rwh	<b>CPU 优先级</b> 0 <sub>H</sub> 优先级最低 ... F <sub>H</sub> 优先级最高
IEN	11	rw	<b>全局中断/PEC 使能位</b> 0 <sub>B</sub> 禁止中断/PEC 请求 1 <sub>B</sub> 使能中断/PEC 请求
HLDEN_PL1	10	rwh	<b>保持使能/保护级别选择 1</b> 0 <sub>B</sub> 禁止外部总线仲裁或保护级别 0/1 （参见表 5-23） 1 <sub>B</sub> 使能外部总线仲裁或保护级别 2/3 （参见表 5-23）
BANK	[9:8]	rwh	<b>用于寄存器文件组选择</b> 00 <sub>B</sub> 全局寄存器组

符号	位序号	类型	功能描述
			01 <sub>B</sub> 保留 10 <sub>B</sub> 局部寄存器组 1 11 <sub>B</sub> 局部寄存器组 2
<b>USR1</b>	7	rwh	<b>通用标志</b> 应用程序可使用该位。当使用重复 MAC 指令时，也置位该位（ <a href="#">章节 5.9.11</a> ）。
<b>USR0</b>	6	rwh	<b>通用标志</b> 应用程序可使用该位。当使用重复 MAC 指令时，也置位该位（ <a href="#">章节 5.9.11</a> ）。
<b>PL0</b>	5	rwh	<b>保护级别选择 0</b> 0 <sub>B</sub> 保护级别 0/2（参见 <a href="#">表 5-23</a> ） 1 <sub>B</sub> 保护级别 1/3（参见 <a href="#">表 5-23</a> ）
<b>E</b>	4	rwh	<b>表格结束标志</b> 0 <sub>B</sub> 源操作数既不是 8000 <sub>H</sub> 也不是 80 <sub>H</sub> 1 <sub>B</sub> 源操作数是 8000 <sub>H</sub> 或 80 <sub>H</sub>
<b>Z</b>	3	rwh	<b>零标志</b> 0 <sub>B</sub> ALU 结果非零 1 <sub>B</sub> ALU 结果为零
<b>V</b>	2	rwh	<b>溢出标志</b> 0 <sub>B</sub> 没有产生溢出 1 <sub>B</sub> 产生溢出
<b>C</b>	1	rwh	<b>进位标志</b> 0 <sub>B</sub> 没有产生进位/借位 1 <sub>B</sub> 产生进位/借位
<b>N</b>	0	rwh	<b>负值结果</b> 0 <sub>B</sub> ALU 结果非负 1 <sub>B</sub> ALU 结果为负

**ALU/MAC 状态 (N, C, V, Z, E)**

PSW 中的状态标志 (N、C、V、Z、E) 指示最近一次执行 ALU 操作之后, ALU 的状态。大多数指令会根据特定规则 (这取决于执行 ALU 操作或是数据转移操作) 设置这些标志位。

若通过指令显性更新 PSW 寄存器, 则不能按照以下的描述来解读状态标志, 因为对 PSW 的任何显性设置将取代由 CPU 隐含产生的状态标志。用户读取 PSW 寄存器, 将读出前一条指令执行之后的 PSW 状态。

*注: 复位后所有的 ALU 状态位被清零。*

**N 标志:** 对大多数 ALU 操作, 如果结果的最高位为 1, 则 N 标志置 1; 反之清零。对于整数操作, N 标志可以看成结果的符号位 (负数:  $N=1$ , 正数:  $N=0$ )。负数总是用相应正数的 2 补码表示。对于字数据类型, 有符号数的范围是  $-8000_{\text{H}}$  至  $+7FFF_{\text{H}}$ , 对字节数据类型, 从  $-80_{\text{H}}$  至  $+7F_{\text{H}}$ 。对只有一个操作数的布尔位操作, N 标志表示指定位的前一个状态; 对两个操作数的布尔位操作, N 标志表示两个指定位的逻辑异或 (XOR)。

**C 标志:** 一次加法运算之后, C 标志表示指定字或字节数据的最高位是否产生进位。减法或比较操作后, C 标志指示借位, 即加法进位标志的逻辑“非”。

这意味着, 在减法运算中, 若指定字或字节数据的最高位没有产生进位, 则 C 标志置 1 (在 ALU 内部, 通过 2 补码加法来实现减法运算); 当补码加法引起进位时, C 标志清零。

对于逻辑操作、乘除 ALU 操作, C 标志始终清零, 因为这些操作不能产生进位。

对移位和循环移位操作来说, C 标志表示最后移出位的值。如果移位计数到 0, C 标志清零。对优先化 ALU 操作来说, C 标志也被清零, 这是因为在对一个操作数归一化过程中不可能从最高位移出 1。

对只有一个操作数的布尔位操作, C 标志总是被清零。对两个操作数的布尔位操作, C 标志表示两个指定位的逻辑与。

**V 标志:** 进行加法、减法和 2 补码运算时, 如果字运算结果超过 16 位有符号数取值范围 ( $-8000_{\text{H}}$  到  $+7FFF_{\text{H}}$ ), 或者字节运算结果超过 8 位有符号数取值范围 ( $-80_{\text{H}}$  到  $+7F_{\text{H}}$ ), 则 V 标志始终置 1。反之 V 标志清零。请注意, 如果 V 标志指示发生算术溢出, 则整数加、整数减或 2 补码运算的结果无效。

进行乘除运算时, 如果结果不能用一个字数据来表示, 则 V 标志置 1; 反之清零。请注意, 除以零的操作始终导致溢出。与除法结果不同, 不论 V 标志是否置 1, 乘法结果始终有效。

由于逻辑 ALU 操作不可能产生一个无效结果, 因此 V 标志始终清零。

V 标志还能用作循环右移位和右移位操作的“粘着位”。仅使用 C 标志, 可以估算的由右移位操作所导致的舍入误差最多为  $1/2$  LSB。和 V 标志一起使用时, C 标志可以更精确的估算舍入误差 (见 [表 5-22](#))。

对只有一个操作数的布尔位操作，V 标志始终被清零；对有两个操作数的布尔位操作，V 标志表示两个指定位的逻辑或（OR）。

表 5-22 右移位舍入误差的估算

C 标志	V 标志	舍入误差的估算
0	0	无舍入误差
0	1	$0 < \text{舍入误差} < 1/2 \text{ LSB}$
1	0	舍入误差 = $1/2 \text{ LSB}$
1	1	舍入误差 $> 1/2 \text{ LSB}$

**Z 标志：**如果 ALU 操作结果等于 0，通常 Z 标志置 1；反之的清零。

对带进位的加法和减法运算，只有当 Z 标志已经为 1，并且当前 ALU 的操作结果也等于零时，Z 标志才被置 1。该机制用于支持多精度计算。

对只有一个操作数的布尔位操作，Z 标志表示指定位前一状态的逻辑“非”。对有两个操作数的布尔位操作，Z 标志表示两个指定位的逻辑或非（NOR）。对于优先化 ALU 操作，Z 标志指示第二个操作数是否为 0。

**E 标志：**表格结束标志。E 标志可以由执行 ALU 或数据转移操作的指令改变。那些不能合理用来搜索表格的指令将清零 E 标志。在所有其它情况下，E 标志值取决于源操作数的值，以指示是否达到搜索表的末端。如果指令的源操作数的值等于相应指令的数据格式所代表的最小负值（字数据类型为 8000H；字节数据类型为 80H），E 标志置 1；否则清零。

### 通用控制功能（USR0，USR1，BANK，HLDEN）

寄存器 PSW 中包含一些位专用于通用控制功能。在任务切换和中断时，它们会被自动保存和恢复。

**USR0/USR1 标志：**在重复执行 MAC 指令时，USR0/USR1 标志可被自动置位。它们还可用作应用程序的通用标志。

**BANK：**位域 BANK 选择当前工作的寄存器组（局部或全局）。在进入中断服务程序，以及执行 RETI 指令后，硬件自动更新位域 BANK。还可以使用任何能对 PSW 进行操作指令显性修改位域 BANK。

**HLDEN：**首次置位该位将激活所选择的总线仲裁模式。可通过临时清除位 HLDEN 禁止总线仲裁。在这种情况下，总线被锁定，总线仲裁模式仍然保持选定。详细信息请参考外部总线控制器（EBC）一章。请注意仅当存储器保护（MPU）被禁止时，才能访问 HLDEN 位。

### 保护级别（PL0，PL1）

这两个标志指定系统当前的保护级别。系统所实现的存储器保护（如 MPU）需要该信息。根据下表定义了四个不同保护级别。欲详细了解保护系统的工作方式，请参考存储器保护（MPU）一章。

**表 5-23      保护级别解码**

PL1	PL0	保护级别
0	0	保护级别 0
0	1	保护级别 1
1	0	保护级别 2
1	1	保护级别 3

当 MPU 被使能时，写 PSW.10 将被解释为对 PL1 进行写操作，当 MPU 被禁止时，写 PSW.10 被解释为向 HLDEN 进行写操作。考虑到事实上有可能同时使用 EBC 仲裁和 MPU 功能。请注意可使用软件支持外部主机功能，即当 MPU 被使能时，试图对 HLDEN 进行的写操作不可能被许可，除非运行在特权模式下。

### CPU 中断状态（IEN，ILVL）

**IEN:** 中断使能位，允许中断被全局使能（IEN = 1）或禁止（IEN = 0）。

**ILVL:** 四位中断优先级位域，指定当前 CPU 工作的优先级。进入中断服务程序时，CPU 中断优先级由硬件更新。不过它也可由软件修改，以防止 CPU 响应其它中断。如果分配 CPU 的中断优先级为 15，即具有最高优先级，因此除硬件强制中断或外部非屏蔽中断之外，当前 CPU 操作不能被中断。

复位后，所有中断被全局禁止，CPU 最初操作的优先级设定为最低（ILVL = 0）。

### 5.8.1      16 位加法器/减法器，阵列移位器和 16 位逻辑单元

所有的标准算术和逻辑操作由 16 位 ALU 完成。进行字节操作时，ALU 结果位 6 和位 7 用来控制状态标志。进行多精度运算时，将运算前面计算结果的“进位”信号送给 ALU。

16 位阵列移位器支持单周期内多位移位，还支持循环和算术移位。

### 5.8.2 位操作单元

XE16xyM提供了许多用于位处理的指令。通常这些指令可用于：

- 控制 CPU 寄存器、GPR 或 DPRAM 中的软件标志位
  - 通过(E)SFR 中的可位寻址控制位来控制相应的片上+BUS 外设和端口逻辑
- 位处理指令仅支持带bitoff操作数的短寻址模式（见[章节5.7.1](#)）。

*注：所有GPR都可位寻址，与通过上下文指针（CP）分配的寄存器组地址无关。即使将GPR分配到不可位寻址的RAM地址，仍然具备该特性。*

指令BSET、BCLR、BAND、BOR、BXOR、BMOV、BMOVN显性的置位或清除特定位。位域指令BFLDL和BFLDH允许同时屏蔽处理特定字节中的最多8位。在执行跳转指令时，指令JBC和JNBS隐含的清除或置位特定位。指令JB和JNB根据特定位的状态来决定是否进行跳转。

*注：读取未定义的位始终返回 '0'，对其进行写操作无效。*

#### 使用屏蔽保护写进行位保护

对单个位或一组位进行操作的指令采用读取-修改-回写序列或屏蔽保护写操作。

读取-修改-回写序列对包含指定位的整个字进行访问，这对受硬件影响的位（类型为'rw'或'wh'）来说可能很关键。在这些情况下，读取-修改-回写指定位的过程中，硬件可能修改寄存器中其它位的值，此时，回写值会覆盖由硬件产生的新值。

对于位可寻址（E）SFR寄存器而言，为了解决以上的负面影响，可使用屏蔽保护写以支持位保护机制。

举例：

```
BCLR EOPIC.EOPIE ; disable 'end of PEC' interrupts
```

该指令将清除中断使能位 EOPIE，同时'rw'位 EOPIR 将受到屏蔽保护。从而确保刚好同时出现的 EOP 中断将被标志位正确指示。

*注：对于BFLD(LH)指令，必须由编程人员提供保护屏蔽。*

*注：如果对同一位的硬件控制和软件访问同时出现而直接发生冲突，则软件访问占优并决定该位的最终值。*



### 5.8.3 乘除单元

XE16xyM 的乘除单元由两个独立的部分组成：一个是快速 16 × 16 位乘法器，能在一个 CPU 周期内完成乘法运算；一个是除法子单元，能在 18 至 21 个 CPU 周期内（取决于数据和除法类型）完成除法运算。执行除法指令需要 4 个 CPU 周期。为了提高性能，除法算法的剩余 17 个 CPU 周期在后台运行，其间可以并行执行其它指令，也可以无延迟、立即开始执行中断任务。如果在除法运算的过程中，某指令（来自于原指令流或中断任务）试图使用除法单元，则该指令被停滞，直到前面的除法运算完成为止。

为避免指令停滞，在中断任务的前 14 个 CPU 周期内不应使用乘除单元。举例来说：进入中断后，需要执行多达 14 条单周期指令以后，才能再次使用乘除指令（最坏情况）。

乘除运算隐含的使用 32 位乘/除寄存器 MD（由两个不可位寻址的数据寄存器 MDH 和 MDL 级联而成）和相应的控制寄存器 MDC。当 ALU 内部执行乘除操作时，CPU 隐含的使用 MDC（可位寻址的 16 位寄存器）。

乘法运算的 32 位结果保存在 MD 中。若进行长除运算，必须在除法运算开始之前将 32 位被除数加载到 MD 中。除法运算之后，寄存器 MDH 给出 16 位余数，寄存器 MDL 给出 16 位商。

#### MDH

乘/除高位字						SFR (FE0C <sub>H</sub> /06 <sub>H</sub> )										复位值: 0000 <sub>H</sub>					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
MDH																					
rwh																					

符号	位序号	类型	功能描述
MDH	[15:0]	rwh	MD 的高位部分 32 位乘除寄存器 MD 的高 16 位。

#### MDL

乘/除低位字						SFR (FE0EH/07H)										复位值: 0000H					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0						
MDL																					
rwh																					

符号	位序号	类型	功能描述
<b>MDL</b>	[15:0]	rwh	<b>MD 的低位部分</b> 32 位乘除寄存器 MD 的低 16 位。

只要软件更新 MDH 或 MDL，乘/除控制寄存器（MDC）中的乘/除寄存器正在使用标志位（MDRIU）即置为‘1’。只要软件读取寄存器 MDL，则 MDRIU 标志被清除。

## MDC

乘/除控制

**SFR（FF0E<sub>H</sub> /87<sub>H</sub>）**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>											<b>MDRIU</b>	<b>0</b>			
r											rwh	r			

符号	位序号	类型	功能描述
<b>MDRIU</b>	4	rwh	<b>乘/除寄存器正在使用标志</b> 0 <sub>B</sub> 软件读取 MDL 时该位被清除 1 <sub>B</sub> 软件写入 MDL 或 MDH；或当执行乘法或除法指令时，该位置 1
<b>0</b>	[15:5], [3:0]	r	<b>保留</b> 读操作返回 0；应写入 0

*注：MDRIU 标志指示寄存器 MD（MDL 和 MDH）的使用情况，因此必须新的乘法或除法操作前保存 MD。*

## 5.9 DSP 数据处理 (MAC 单元)

在 MAC 单元中执行 CoXXX 算术指令。MAC 单元支持单周期、非流水线的 32 位加、32 位减、右移和左移、16×16 位乘法、乘累加/减。MAC 单元包括以下主要部分，如图 5-16 所示：

- 16×16 位有符号/无符号乘法器，结果带符号<sup>1)</sup>
- 级联单元
- 用于小数计算的换算器（左移一位移位器）
- 40 位加法/减法器
- 40 位有符号累加器
- 数据限制器
- 累加移位器
- 重复计数器

---

1) ALU 中使用相同的硬件乘法器。

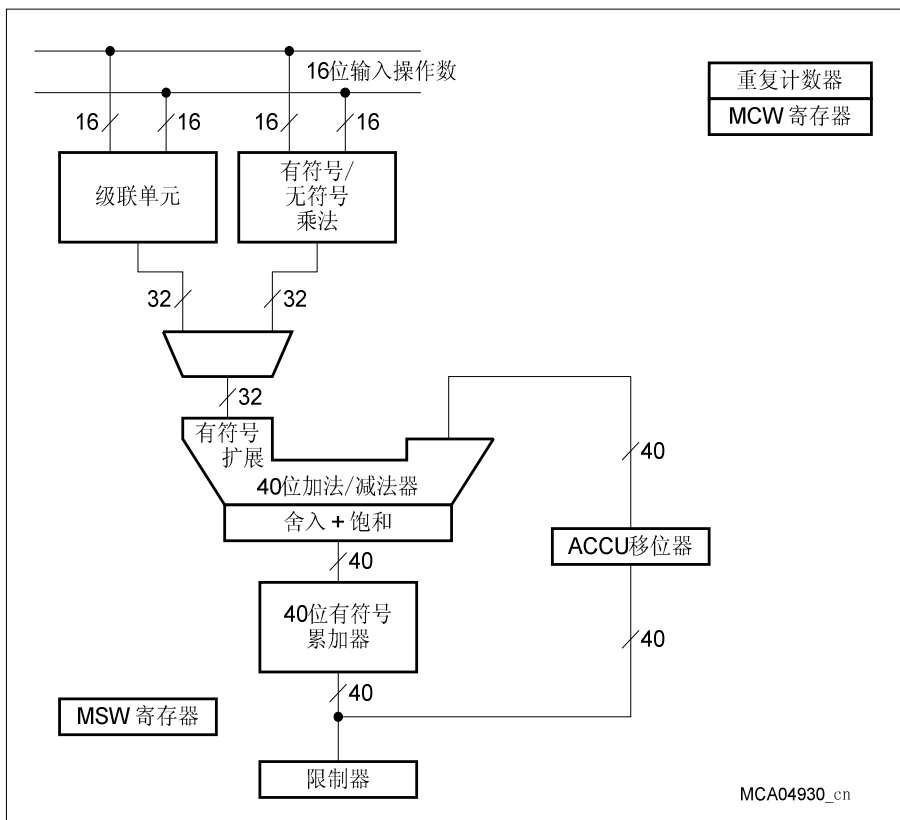


图 5-16 MAC 单元功能框图



### 5.9.3 16 位×16 位有符号/无符号乘法器和换算器

乘法器在一个 CPU 周期内并行完成 16×16 有符号/无符号小数和整数乘法。乘法器支持无符号和有符号操作数的乘法，乘积始终用有符号小数或整数格式表示。

乘积送给一位换算器，可用于补偿由两个 16 位 2 补码数相乘产生的附加符号位。

### 5.9.4 级联单元

级联单元使得 MAC 能够在一个 CPU 周期内完成 32 位的算术运算。在 40 位加法/减法器中执行 32 位算术运算之前，级联单元将两个 16 位操作数级联为一个 32 位操作数。40 位加法/减法器需要的第二个操作数始终是累加器的当前值。级联单元还用来给累加器预载一个 32 位的数值。

### 5.9.5 一位换算器

一位换算器可以将级联单元的结果或乘法器的输出左移一位。换算器由执行级联的指令或寄存器 MCW 中的位 MP 控制。

如果 MP 位置位，则乘积左移一位，用于补偿由两个 16 位 2 补码数相乘产生的附加符号位。只有当两个输入操作数都为有符号数时，才执行自动移位。

### 5.9.6 40 位加法/减法器

在进行一系列乘/累加操作期间，40 位加法/减法器允许中间值溢出。加法/减法器有两个输入端口（一个 40 位输入口和一个 32 位输入口）。通过 ACCU 移位器将累加器的输出值反馈到加法/减法器的 40 位输入端口上。加法/减法器 32 位输入口的操作数来自一位换算器。32 位操作数是有符号数，在执行加/减算术前被扩展成 40 位。

加法/减法器的输出送入累加器。每次累加过后，也可以自动对结果舍入和饱和至 32 位值。通过给结果加上 00'0000'8000<sub>H</sub> 来实现舍入操作；通过置位寄存器 MCW 中的位 MS 使能自动饱和操作。

当累加器处于溢出饱和模式，且发生溢出时，根据溢出方向和所采用的算术运算，将 32 位数据的最大正值或最大负值载入累加器。饱和操作后累加器的值为 00'7FFF'FFFF<sub>H</sub>（正）或 FF'8000'0000<sub>H</sub>（负）。

### 5.9.7 数据限制器

当利用 CoSTORE <destination>., MAS 指令读取累加器时，也可用饱和算法来限制溢出。对 MAC 单元的累加器执行限制操作。如果累加器内容可以用目的操作数的大小表示而不产生溢出，则数据限制器被禁用，操作数不被修改。如果累加器内容用目标操作数的大小来表示会产生溢出，限制器将用一个“限制”数据（见表 5-25）代替累加器内容。

表 5-24 限制器输出

ME 标志	MN 标志	限制器输出
0	x	不变
1	0	7FFF <sub>H</sub>
1	1	8000 <sub>H</sub>

注：在这种特殊情况下，累加器和状态寄存器都不受影响。只能通过 CoSTORE 指令读取 MAS。

### 5.9.8 累加移位器

累加移位器是一个40位输入、40位输出的并行移位器。源累加器移位操作如下：

- 没有移位（不被修改）
- 最多 16 位算术左移
- 最多 16 位算术右移

注意寄存器 MSW 中的 ME、MSV、MSL 受左移影响；因此，如果饱和机制被使能（MS 位），其行为与加法/减法器相似。

注：在饱和使能的情况下执行左移操作，需要特别的注意。通常，如果 MAE 包含有效位，则累加器中的 32 位值将饱和。但是，左移操作有可能将某些有效位移出累加器。这种情况下，40 位结果值将被错误的解释，或者不被饱和，或者被错误的饱和。左移操作的结果还有可能将一个原来的正数饱和成最小的负数，反之亦然。

### 5.9.9 40 位有符号累加器寄存器

40 位累加器由三个寄存器 MAE、MAH、MAL 级联组成。其中，MAE 8 位宽，MAH 和 MAL 16 位宽。MAE 是 40 位累加器的最高有效字节。该字节执行监控功能。MAE 位于 MSW 的低位字节。

当 MAH 被写入时，累加器的值自动调整至符号扩展的 40 位格式。这意味着 MAL 被清零，对正数（MAH 的最高位为 0），向 MAE 中自动载入 00<sub>H</sub>；对负数（MAH 最高位是 1），向 MAE 中自动载入 FF<sub>H</sub>，以 2 补码格式表示被扩展的 40 位负数。可以看出，扩展的 40 位值等于未扩展的 32 位值。换句话说，扩展之后，MAE 不包含有效位。通常，当 40 位有符号结果的最高 9 位相同时，会出现这种情况。

累加器操作期间，可能发生溢出，导致无法用 32 位表示计算结果，这时 MAE 会改变。当累加器中的有符号结果已超出 32 位，MSW 中的扩展标志“E”置位。当 40 位有符号结果的最高 9 位不相同，出现这种情况，即 MAE 包含有效位。

大多数 CoXXX 操作使用 40 位累加器寄存器作为源和/或目的操作数。

## **MAL**

**累加器低位字**

**SFR (FE5C<sub>H</sub>/2E<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MAL</b>															
rwh															

符号	位序号	类型	功能描述
<b>MAL</b>	[15:0]	rwh	<b>累加器的低位部分</b> 与位域 MAE 及累加器高位字(MAH)级联, 构成 40 位累加器。

## **MAH**

**累加器高位字**

**SFR (FE5E<sub>H</sub>/2F<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MAH</b>															
rwh															

符号	位序号	类型	功能描述
<b>MAH</b>	[15:0]	rwh	<b>累加器的高位部分</b> 与位域 MAE 及累加器低位字(MAL)级联, 构成 40 位累加器。



### 5.9.10 MAC 单元状态字 MSW

寄存器 MSW 的高字节（可位寻址）给出 MAC 单元的当前状态。寄存器 MSW 的低字节表示 MAC 累加器扩展的高 8 位，与寄存器 MAH、MAL 级联构成 40 位累加器。

#### MSW

#### MAC 状态字

#### SFR（FFDE<sub>H</sub>/EF<sub>H</sub>）

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	MV	MSL	ME	MSV	MC	MZ	MN	MAE							
r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh							

符号	位序号	类型	功能描述
<b>0</b>	15	r	<b>保留</b> 读操作返回 0；应写入 0
<b>MV</b>	14	rwh	<b>溢出标志</b> 0 <sub>B</sub> 没有产生溢出 1 <sub>B</sub> 产生溢出
<b>MSL</b>	13	rwh	<b>粘着限制标志</b> 0 <sub>B</sub> 结果不饱和 1 <sub>B</sub> 结果饱和
<b>ME</b>	12	rwh	<b>MAC 扩展标志</b> 0 <sub>B</sub> MAE 不包含有效位 1 <sub>B</sub> MAE 包含有效位
<b>MSV</b>	11	rwh	<b>粘着溢出标志</b> 0 <sub>B</sub> 没有溢出发生 1 <sub>B</sub> 溢出发生
<b>MC</b>	10	rwh	<b>进位标志</b> 0 <sub>B</sub> 没有产生进位/借位 1 <sub>B</sub> 产生进位/借位
<b>MZ</b>	9	rwh	<b>零标志</b> 0 <sub>B</sub> MAC 结果非零 1 <sub>B</sub> MAC 结果为零

符号	位序号	类型	功能描述
<b>MN</b>	8	rwh	<b>负标志</b> 0 <sub>B</sub> MAC 结果为正 1 <sub>B</sub> MAC 结果为负
<b>MAE</b>	[7:0]	rwh	<b>MAC 累加器扩展</b> 40 位累加器的最高 8 位，与寄存器 MAH 和 MAL 级联。

**MAC 单元状态 (MV、MN、MZ、MC、MSV、ME、MSL)**

这些状态标志指示最近一次执行 MAC 操作之后 MAC 的状态。大多数指令会根据特定规则（这取决于执行 MAC 操作或是数据转移操作）设置这些标志位。

若通过指令显性更新 MSW 寄存器，则状态标志不再能代表真实的 MAC 状态。对 MSW 的显性设置将取代由 MAC 单元隐含产生的状态标志。用户读取 MSW 寄存器，将读出前一条指令执行之后的 MSW 状态。可以通过任何能够访问 SFR 的指令来访问寄存器 MSW。

*注：复位之后，所有 MAC 状态位被清零。*

**MN 标志：**对大多数 MAC 操作，如果结果的最高位为 1，则 MN 标志置 1；否则清零。整数操作的情况下，MN 标志可以理解成结果的符号位（负值：MN = 1，正值：MN = 0）。负数始终由相应正数的 2 补码来表示。有符号数的范围从 80'0000'0000<sub>H</sub> 到 7F'FFFF'FFFF<sub>H</sub>。

**MZ 标志：**通常如果 MAC 操作结果等于 0，MZ 标志置 1，否则清零。

**MC 标志：**在一次 MAC 加法运算之后，MC 标志指示累加器扩展字节 MAE 的最高位是否产生了“进位”。在一次 MAC 减法或 MAC 比较运算之后，MC 标志指示是否出现“借位”（表示加法“进位”的逻辑“非”）。这意味着，若执行减法运算时，累加器最高位没有产生“进位”，则 MC 标志置 1。MAC 单元通过 2 补码加法实现减法运算，当补码加法导致“进位”时，MC 标志清零。

左移 MAC 操作，MC 标志表示最后移出位的值；右移位 MAC 操作，MC 标志始终清零。对于算术右移 MAC 操作，如果舍入操作使累加器扩展字节 MAE 的最高位产生“进位”，则置位 MC。

**MSV 标志：**进行加法、减法、2 补码和舍入运算时，如果 MAC 结果超出 40 位有符号数的取值范围，则 V 标志始终置 1。如果 V 标志指示发生溢出，则 MAC 结果无效。

MSV 标志是一个“粘着位”。一旦置位，其状态不受其它 MAC 操作的影响。只有直接的写操作能够对其清零。

**ME 标志：**如果累加器扩展字节 MAE 包含有效位，ME 标志置位。也就是说，此时累加器最高 9 位不完全相等。

**MSL 标志：**如果累加器已执行了自动饱和操作，MSL 标志置位。寄存器 MCW 的位 MS 置位时，使能自动饱和。还可以由限制累加器内容的指令置位 MSL 标志。如果累加器内容被限制，则 MSL 标志置位。

MSL 是一个“粘着位”。一旦置位，其状态不受其它 MAC 操作的影响。只有直接的写操作能够对其清零。

**MV 标志：**如果加法、减法和累加操作的结果超出有符号数的最大范围（80'0000'0000<sub>H</sub> ~ 7F'FFFF'FFFF<sub>H</sub>），则 MV 标志置位；否则清零。请注意：如果 MV 标志指示算术溢出，此时整数加、整数减、或累加的结果无效。

**5.9.11 重复计数器 MRW**

重复计数器 MRW 控制循环的重复次数。该寄存器预载之后，才能被指令-USRx CoXXX 使用。MAC 操作能够递减该计数器。执行-USRx CoXXX 指令时，递减 MRW 之前，检查其是否计数到 0。如果 MRW 等于零，则 USRx 位置位，MRW 不再递减。可以通过能够访问 SFR 的任何指令对寄存器 **MRW** 进行访问。

**MRW**

MAC重复字					SFR（FFDA <sub>H</sub> /ED <sub>H</sub> ）					复位值: 0000 <sub>H</sub>				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1 0
REPEATCOUNT														
rwh														

符号	位序号	类型	功能描述
<b>REPEATCOUNT</b>	[15:0]	rwh	<b>MAC 重复计数器</b>

所有CoXXX指令在其操作数域中均有一个3位宽的重复控制域“rrr”（位地址[31:29]），用于控制MRW重复计数器。表 5-25列出可能的编码。

**表 5-25 MAC 重复字控制的编码**

“rrr” 编码	对重复计数器的影响
000 <sub>B</sub>	常规 CoXXX 指令
001 <sub>B</sub>	保留
010 <sub>B</sub>	‘-USR0 CoXXX’ 指令， 递减重复计数器，如果 MRW 为 0，将 USR0 置位
011 <sub>B</sub>	‘-USR1 CoXXX’ 指令， 递减重复计数器，如果 MRW 为 0，将 USR1 置位
1XX <sub>B</sub>	保留

*注：位 USR0 在先前的体系结构中也用作通用标志。为防止程序员或编译器使用该标志而产生冲突，使用 ‘-USR0 CoXXX’ 指令时必须非常小心。*

下面的例子给出一个执行 20 次的循环操作。每执行一次 CoMACM 指令，MRW 计数器减 1。

```
MOV    MRW, #19           ;预装载循环计数器
loop01:
-USR1   CoMACM [IDX0+], [R0+] ;计算和递减 MSW
        ADD R2, #0002H
        JMPA cc_nusr1, loop01 ;重复循环，直到 USR1 置位
```

*注：由于正确预测的 JMPA 指令执行时间为零周期，从而提供了重复指令的功能。*

## 5.10 常数寄存器

这些可位寻址寄存器的所有位由硬件固定为 0 或 1，它们是只读寄存器。寄存器 ZEROS/ONES 中包含全 0 或全 1，可用于位操作或屏蔽码产生。可通过能够寻址 SFR 的任何指令访问常数寄存器。

### ZEROS

常数0寄存器						SFR (FF1C <sub>H</sub> /8E <sub>H</sub> )						复位值: 0000 <sub>H</sub>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ZERO															
r															

符号	位序号	类型	功能描述
ZERO	[15:0]	r	常数 0 位

### ONES

常数1寄存器						SFR (FF1E <sub>H</sub> /8F <sub>H</sub> )						复位值: FFFF <sub>H</sub>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ONE															
r															

符号	位序号	类型	功能描述
ONE	[15:0]	r	常数 1 位

### CPUID

CPU ID寄存器						ESFR (F00CH/06H)						复位值: 0313H			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODULENUMBER								VERSIONNUMBER							
r								r							

符号	位序号	类型	功能描述
<b>MODULENUMBER</b>	[15:8]	r	<b>C166 家族 CPU 模块编号（C166S-V2）</b>
<b>VERSIONNUMBER</b>	[7:0]	r	<b>C166S-V2 CPU 版本号</b>

## 6 存储器保护单元 (MPU)

存储器保护单元 (MPU) 为 XE16xyM 系列微控制器中实现的存储器保护机制提供硬件机制。MPU 能够检测对用户定义的存储器范围进行的未经授权的访问（读、写或取指）。MPU 还为整个地址空间提供保护，外设范围也包括在内。

MPU 还支持对处理器上运行的不同应用或软件构件进行封装，这为当今依赖于多方软件的复杂系统提供了确保一致性和故障隔离能力的有效方式。

### 6.1 功能概述

通常需要不同保护级别以支持编程系统，例如操作系统或软件内核运行并控制不同应用和底层驱动部分。某个保护级别可与操作系统关联，其它需要相互或与操作系统隔离的任务则分配另外的保护级别。对于每个保护级别，可为指令和/或数据定义不同的地址范围以及不同的访问许可。当执行一条代码、存储器保护被使能时，选择与其保护级别对应的权限，每次执行存储器访问时，检查访问是否超出规定的保护范围或是否违反访问权限。如果发生违法操作，访问将不能执行且被标示为无效，并执行一个保护强制中断程序。

MPU 的基本功能见图 6-1。

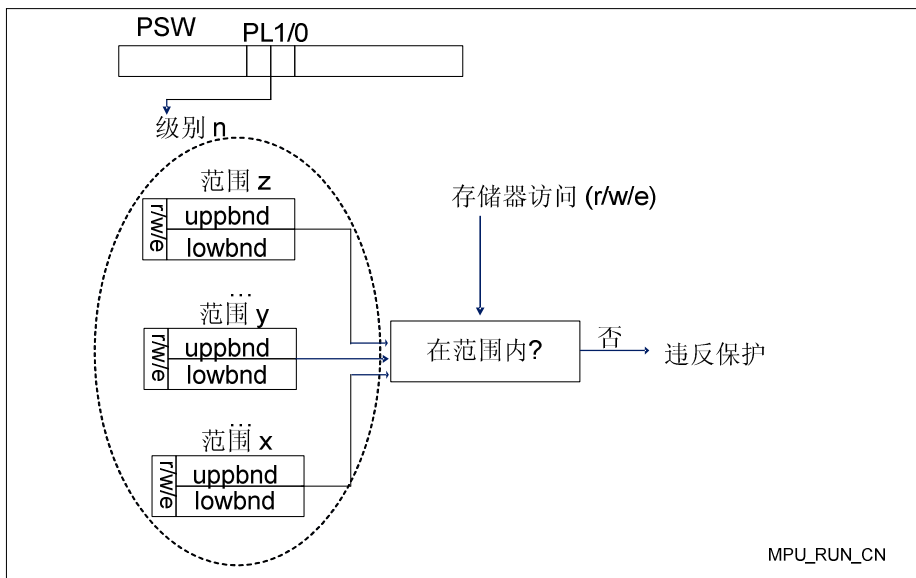


图 6-1 MPU 操作



## 存储器保护单元 (MPU)

该架构中，系统运行期间四个保护级别可共存。可使用处理器状态字 (PSW) 中的两位选择在给定时间内那个保护级别被激活。如果应用需要的保护级别多于 4 个，需要执行将所有保护级别重新映射到 4 个可能的保护级别上的操作，当需要切换保护级别时，需要在运行期间重行编程设置保护寄存器组。

每个保护级别对应一组保护寄存器，每组寄存器包含所有的地址范围和与相关保护级别对应的访问许可。每个保护寄存器组能包含数目可编程设定的范围寄存器。总共支持最多 12 个地址范围。保护模式寄存器定义和每个代码或数据地址范围相对应的访问许可。保护系统需要的 MPU 寄存器及其用法的详细解释请参考下一节。

### 6.2 存储器保护寄存器

保护寄存器组由数目可变的保护范围寄存器对 (PRUx/PRLx) 和对应数目的保护模式寄存器 (PMx) 组成。PMx 寄存器位于 SFR 区域，可通过外设数据总线-PD-Bus-访问。PRUx/PRLx 寄存器未被存储器映射，因此通过存储器映射寄存器保护范围地址寄存器 (PRA) 和保护范围数据寄存器 (PRD) 支持对 PRUx/PRLx 寄存器的访问。

表 6-1 寄存器的地址空间

模块	基址	结束地址	注
MPU	0 <sub>H</sub>		

表 6-2 寄存器概览

寄存器缩写名	寄存器全名	偏移地址	页码
PRUx (x= 0-11)	保护范围寄存器 x 上边界	none <sub>H</sub>	<a href="#">页 6-3</a>
PRLx (x= 0-11)	保护范围寄存器 x 下边界	none <sub>H</sub>	<a href="#">页 6-3</a>
PM5	保护模式寄存器 5	FFD2 <sub>H</sub>	<a href="#">页 6-5</a>
PM4	保护模式寄存器 4	FFD0 <sub>H</sub>	<a href="#">页 6-5</a>
PM3	保护模式寄存器 3	FFCE <sub>H</sub>	<a href="#">页 6-5</a>
PM2	保护模式寄存器 2	FFCC <sub>H</sub>	<a href="#">页 6-5</a>
PM1	保护模式寄存器 1	FFCA <sub>H</sub>	<a href="#">页 6-5</a>
PM0	保护模式寄存器 0	FFC8 <sub>H</sub>	<a href="#">页 6-4</a>
PRD	保护范围数据	FFC6 <sub>H</sub>	<a href="#">页 6-7</a>
PRA	保护范围地址	FFC4 <sub>H</sub>	<a href="#">页 6-7</a>

### 6.2.1 保护范围寄存器

PRUx/PRLx 寄存器是 16 位寄存器，它们分别定义了所有允许数据和/或代码访问的地址范围（最多 12 个）的高 16 位物理地址（上边界和下边界）。只对物理地址的高 16 位进行地址比较，因此最小的地址范围分区是 256 个字节，所有地址范围都要和这个大小对齐。

PRUx 和 PRLx 寄存器分别给出地址范围的高地址和低地址。如果由于编程错误使得 PRLx 值大于 PRUx 值，则相应的地址范围不正确，导致该地址范围无用（即被忽略）。请注意：由于最小的地址分区大小为 256 字节，为了对最小的地址范围进行编程，地址范围的上边界和下边界要设置为同一个值，即该地址范围的基址。

对 PRUx/PRLx 寄存器内的保护范围进行编程时，首先需要将地址编程到 PRA 以选择将要写入的地址范围，然后通过将数据写入到 PRD 的方式执行写数据操作。与此相似，首先需要选择读操作的地址范围（将地址编程到 PRA），然后可通过读取 PRD 执行读操作。编程 PRUx/PRLx 寄存器需要两次写操作。与此相似，读取 PRUx/PRLx 寄存器也需要两次操作（一次写操作和一次读操作）。对于连续地址且使用自动递增特性时，仅需要对 PRA 进行一次初始化，随后每次仅需要写/读访问 PRD 寄存器。寄存器 PRD 和 PRA 的描述分别见[章节 6.2.3](#)和[章节 6.2.4](#)。

#### PRUx (x = 0-11)

保护范围寄存器 x 上边界

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
UPPBNB	[15:0]	rw	上边界地址（高 16 位）

#### PRLx (x = 0-11)

保护范围寄存器 x 下边界

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
LOWBNB	[15:0]	rw	下边界地址（低 16 位）

### 6.2.2 保护模式寄存器

与每个地址范围相关的所有控制信息保存在保护模式寄存器中。在模式寄存器中规定访问许可（执行、读和/或写）和地址范围到保护级别的映射。可单独为每个地址范围设定任何保护级别，甚至可设定多于一个保护级别（但是必须使用同一个访问许可）。在一个保护模式寄存器中提供用于使能保护系统的位域。

请注意，修改这些寄存器时，并未执行清空流水线的硬件机制。因为 MPU 配置寄存器的编程（重新编程）无论如何应当在禁止保护的情况下进行，所以上述情况通常不成问题。而且影响某个保护级别的配置通常被（重新）编程自另一个保护级别，意味着即使当保护被使能时，当前运行的软件也将不受配置改变的影响（一旦根据 [章节 6.4.2](#) 描述的过程改变保护级别，配置才生效）。对于需要立即生效的保护配置改变，软件必须特别留意，在执行下一个受影响的指令（例如读取最近写入的寄存器）之前，写操作必须有效。

下面的 PMx 寄存器的位域描述使用通用范围名称（A，B），它们所映射到的物理范围由其所属的 PMx 寄存器名称给出。对于给定的模式寄存器 x，A 的物理地址范围为 2\*x，B 的物理地址范围为 2\*x+1。

PMx 受 EINIT 保护。

#### PM0

##### 保护模式寄存器 0

##### SFR (FFC8<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
L3E B	L2E B	L1E B	L0E B	WEB	REB	XEB	0	L3E A	L2E A	L1E A	L0E A	WEA	REA	XEA	PRO TEN
rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
<b>PROTEN</b>	0	rw	<b>保护使能位</b> 该位使能保护机制。 0 <sub>B</sub> 未使能保护 1 <sub>B</sub> 使能保护
<b>XEA, XEB</b>	1, 9	rw	<b>执行使能</b> 0 <sub>B</sub> 不允许从相关地址范围（A，B）读取指令 1 <sub>B</sub> 允许从相关地址范围（A，B）读取指令
<b>REA, REB</b>	2, 10	rw	<b>读使能</b> 0 <sub>B</sub> 不允许从相关地址范围（A，B）读数据

**存储器保护单元 (MPU)**

符号	位序号	类型	功能描述
			1 <sub>B</sub> 允许从相关地址范围 (A, B) 读数据
<b>WEA, WEB</b>	3, 11	rw	<b>写使能</b> 0 <sub>B</sub> 不允许向相关地址范围 (A, B) 写数据 1 <sub>B</sub> 允许向相关地址范围 (A, B) 写数据
<b>L0EA, L0EB</b>	4, 12	rw	<b>保护级别 0 使能</b> 0 <sub>B</sub> 地址范围 (A, B) 未使能保护级别 0 1 <sub>B</sub> 地址范围 (A, B) 使能保护级别 0
<b>L1EA, L1EB</b>	5, 13	rw	<b>保护级别 1 使能</b> 0 <sub>B</sub> 地址范围 (A, B) 未使能保护级别 1 1 <sub>B</sub> 地址范围 (A, B) 使能保护级别 1
<b>L2EA, L2EB</b>	6, 14	rw	<b>保护级别 2 使能</b> 0 <sub>B</sub> 地址范围 (A, B) 未使能保护级别 2 1 <sub>B</sub> 地址范围 (A, B) 使能保护级别 2
<b>L3EA, L3EB</b>	7, 15	rw	<b>保护级别 3 使能</b> 0 <sub>B</sub> 地址范围 (A, B) 未使能保护级别 3 1 <sub>B</sub> 地址范围 (A, B) 使能保护级别 3
<b>0</b>	8	r	<b>保留位</b>

仅在保护模式寄存器 0 中有 PROTEN 位。

**PMx (x = 1-5)**

**保护模式寄存器 x**

**SFR (FFC8<sub>H</sub>+2\*x)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>L3E B</b>	<b>L2E B</b>	<b>L1E B</b>	<b>L0E B</b>	<b>WEB</b>	<b>REB</b>	<b>XEB</b>	<b>0</b>	<b>L3E A</b>	<b>L2E A</b>	<b>L1E A</b>	<b>L0E A</b>	<b>WEA</b>	<b>REA</b>	<b>XEA</b>	<b>0</b>
rw	rw	rw	rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	r

符号	位序号	类型	功能描述
<b>0</b>	0	rw	<b>保留位</b>

**存储器保护单元 (MPU)**

符号	位序号	类型	功能描述
<b>XEA, XEB</b>	1, 9	rw	<b>执行使能</b> 0 <sub>B</sub> 不允许从相关地址范围 (A, B) 读取指令 1 <sub>B</sub> 允许从相关地址范围 (A, B) 读取指令
<b>REA, REB</b>	2, 10	rw	<b>读使能</b> 0 <sub>B</sub> 不允许从相关地址范围 (A, B) 读数据 1 <sub>B</sub> 允许从相关地址范围 (A, B) 读数据
<b>WEA, WEB</b>	3, 11	rw	<b>写使能</b> 0 <sub>B</sub> 不允许向相关地址范围 (A, B) 写数据 1 <sub>B</sub> 允许向相关地址范围 (A, B) 写数据
<b>L0EA, L0EB</b>	4, 12	rw	<b>保护级别 0 使能</b> 0 <sub>B</sub> 地址范围 (A, B) 未使能保护级别 0 1 <sub>B</sub> 地址范围 (A, B) 使能保护级别 0
<b>L1EA, L1EB</b>	5, 13	rw	<b>保护级别 1 使能</b> 0 <sub>B</sub> 地址范围 (A, B) 未使能保护级别 1 1 <sub>B</sub> 地址范围 (A, B) 使能保护级别 1
<b>L2EA, L2EB</b>	6, 14	rw	<b>保护级别 2 使能</b> 0 <sub>B</sub> 地址范围 (A, B) 未使能保护级别 2 1 <sub>B</sub> 地址范围 (A, B) 使能保护级别 2
<b>L3EA, L3EB</b>	7, 15	rw	<b>保护级别 3 使能</b> 0 <sub>B</sub> 地址范围 (A, B) 未使能保护级别 3 1 <sub>B</sub> 地址范围 (A, B) 使能保护级别 3
<b>0</b>	8	r	<b>保留位</b>

### 6.2.3 保护范围数据寄存器

保护范围数据寄存器包含编程保护范围寄存器的内容所需的 16 位值，还包含最近一次读访问保护范围寄存器期间所读取的数据。写访问 PRD 的操作立即触发对相应 PRUx/PRLx 寄存器（由 PRA 寄存器内的写指针选择）的写入操作。对 PRD 的读访问将立即给出相应 PRUx/PRLx 数据（由 PRA 寄存器中的读指针选择）。

PRD 寄存器受 EINIT（初始化结束）保护。

## PRD

保护范围数据

SFR (FFC6<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
DATA	[15:0]	rwh	写入/来自 PRUx/PRLx 的数据值

### 6.2.4 保护范围地址寄存器

保护范围地址寄存器包含两个访问指针，写指针和读指针。每个 5 位指针可从一组 24 个寄存器（实现 12 个保护范围共需要 24 个 PRUx/PRLx 寄存器）中选择一个 PRUx/PRLx 寄存器。

访问指针的自动递增功能（通过 WMOD 和 RMOD 位域控制）可被使能，每次 PRD 的写或读操作之后，写或读指针分别递增。该特性使得对保护范围寄存器编程速度更快。当自动递增模式有效时，指针达到其最大值之后自动作一个回绕（即初始化至 0）。

由状态位 WWA 或 RWA 指示回绕操作。软件可通过状态位检查指针是否发生回绕，从而采取相应操作并复位相应的标志。

为了防止指针被无意修改，编程 PRA 寄存器时需要特别注意。推荐用户使用位指令来完成操作（例如位域指令）。当使用自动递增特性，且调试期间需要考虑调试器访问也能够修改指针值，那么调试器软件应当注意恢复该寄存器的初始状态。

PRA 寄存器受 EINIT 保护（初始化结束）。

## PRA

保护范围地址

SFR (FFC4<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RMO D	RWA	0	RPTR				WM OD	WW A	0	WPTR					
rw	rwh	r	rwh				rw	rwh	r	rwh					

符号	位序号	类型	功能描述
WPTR	[4:0]	rwh	写指针

符号	位序号	类型	功能描述
			选择要写入的保护范围寄存器 00000 <sub>B</sub> 选择 PRL0 00001 <sub>B</sub> 选择 PRU0 00010 <sub>B</sub> 选择 PRL1 00011 <sub>B</sub> 选择 PRU1 00100 <sub>B</sub> 选择 PRL2 00101 <sub>B</sub> 选择 PRU2 ... 11110 <sub>B</sub> 选择 PRL15 11111 <sub>B</sub> 选择 PRU15
<b>0</b>	5	r	<b>保留位</b>
<b>WWA</b>	6	rwh	<b>写回绕状态</b> 0 <sub>B</sub> 最近一次写操作未发生回绕 1 <sub>B</sub> 最近一次写操作 WPTR 发生回绕，需要由软件清除该位。
<b>WMOD</b>	7	rw	<b>自动递增写模式</b> 0 <sub>B</sub> 每次写操作 WPTR 不增加 1 <sub>B</sub> 每次写操作 WPTR 自动递增
<b>RPTR</b>	[12:8]	rwh	<b>读指针</b> 选择要读取的保护范围寄存器。 00000 <sub>B</sub> 选择 PRL0 00001 <sub>B</sub> 选择 PRU0 00010 <sub>B</sub> 选择 PRL1 00011 <sub>B</sub> 选择 PRU1 00100 <sub>B</sub> 选择 PRL2 00101 <sub>B</sub> 选择 PRU2 ... 11110 <sub>B</sub> 选择 PRL15 11111 <sub>B</sub> 选择 PRU15
<b>0</b>	13	r	<b>保留位</b>

符号	位序号	类型	功能描述
<b>RWA</b>	14	rwh	<b>读回绕状态</b> 0 <sub>B</sub> 最近一次读操作 RPTR 未发生回绕 1 <sub>B</sub> 最近一次读操作 RPTR 发生回绕, 需要由软件清除该位。
<b>RMOD</b>	15	rw	<b>自动递增读模式</b> 0 <sub>B</sub> 每次写操作 RPTR 不增加 1 <sub>B</sub> 每次写操作 RPTR 自动递增

## 6.3 功能描述

### 6.3.1 使能保护

必须通过软件使能全局保护, 由位 PM0.PROTEN 实现该功能, 具体内容请参考[章节 6.2.2](#)。

### 6.3.2 保护级别

位 PSW.PL1/0 选择当前的保护级别, 即当前有效的保护寄存器设置。PL1/0 的解码见下表:

**表 6-3 保护级别解码**

PL1	PL0	保护级别
0	0	保护级别 0
0	1	保护级别 1
1	0	保护级别 2
1	1	保护级别 3

PL1 和 PL0 分别被映射到 PSW.10 和 PSW.5。请注意由于 PSW.10 位的共享功能, 因此只有当 MPU 自身被使能 (PM0.PROTEN 为 1) 时, 写 PSW.10 的操作才被认为是向 PSW.PL1 进行写操作。MPU 未被使能时, 写该位被认为是向 HLDEN 标志进行写操作。为了保持一致性, 标志 PSW.PL0 的处理与此相似, 只有当 MPU 被使能时, 对 PSW.PL0 的写操作才有效。



### 6.3.2.1 保护级别 0

为了使保护机制正常工作，MPU 的操作必须处于一种特权模式。只有在该模式期间，才允许编程和改变保护信息。即使 C166 家族架构不直接支持这种工作模式（仅在执行 EINIT 指令结束初始化阶段相关联时），在此上下文的基础上定义：当处理器运行在保护级别 0 时，进入特权模式。复位之后进入保护级别 0，转入中断/强制中断处理之后，自动进入保护级别 0。需要访问全部系统资源（尤其是系统控制寄存器和外设区域）的操作系统、软件内核或软件构件应当使用保护级别 0。

但请注意保护级别 0 仍然需要对地址范围和访问许可进行定义和编程（即使它适用于整个地址空间）。缺省情况下（即复位之后），即使保护级别为 0，也不允许对任何地址范围进行任何访问。

*注：在特殊情况下，允许编程保护级别 0，但是对该级别进行编程受到严格的访问限制。有时需要考察在该级别下运行的软件的可靠性。*

### 6.3.3 存储器范围的交集

对存储器地址的访问许可是指存储器访问许可范围的或运算。当两个或多个地址范围相互交叠时，交集区域具有最大许可权限。

### 6.3.4 MPU 寄存器的保护

如[章节 6.3.2.1](#)中提到的，需要对 MPU 寄存器进行保护。使用 MPU 时，保护机制自动起效，事实上整个地址空间，包括 SFR，都在 MPU 控制之下，参见[章节 6.3.5](#)。一旦使能保护，只有在允许访问相应 SFR 区域（即保护寄存器）的保护级别上，才能执行对保护信息的修改。

除了此类固有的保护机制，保护控制寄存器还受到 EINIT 保护。动态重新编程期间，EINIT 保护会产生一些开销，然而对于不同软件构件需要在同一个保护级别下执行的情况（可访问这些控制寄存器的保护级别通常为保护级别 0），EINIT 保护增加了上述情况所需的附加保护级别。

### 6.3.5 访问 SFR 和 GPR

一旦保护系统被激活，任务再也不能通过软件随意访问任何特殊功能寄存器（SFR），除非该存储器所在地址范围被覆盖且许可该任务访问这个区域。该规则适用于内部 IO 区域（SFR、ESFR、XSFR）以及外部 IO 区域（片上 LxBus 外设或外部外设）。因为地址范围的最小分区是 256 字节，IO 区域被分成不同地址块。一个任务将通过地址块中间的一个或不通过任何一个访问全部寄存器组。例如，SFR/ESFR 区域（1 KB），被分成 4 个块（F000<sub>H</sub>...F0FF<sub>H</sub>，F100<sub>H</sub>...F1FF<sub>H</sub>，FE00<sub>H</sub>...FEFF<sub>H</sub>，FF00<sub>H</sub>...FFFF<sub>H</sub>），而对于 4 KB 的 XSFR 空间，被分成了 16 个地址块。

由保护机制处理 CSFR，但是作为用户寄存器的 CSFR 例外。受到保护机制影响的 CFSR 是：

- PSW (部分)、CPUCON1/2, CP, CSP, SP, SPSEG, STKUN, STKOV, TFR, VECSEG

PSW 的 USR0/1 为用户位, 被排除在保护机制之外。PSW 的状态标志也被排除在外。JBC/JNBC 等指令的 PSW 条件标志仍可在用户模式下使用。允许对所有 PSW 位域进行读访问。

不受保护机制影响的 CFSR 是:

- DPP0/1/2/3, MDL, MDH, MSW, MDC, MAH, MAL, MRW, MCW, QR0/1, QX0/1, IDX0/1, ZEROS, ONES, CPUID

DPP 寄存器被当作用户寄存器进行处理, 以支持运行期间对其重新编程 (对于代码优化用途需要经过练习)。当以上述方式使用时, 需要由软件确保对它们的正确处理, 例如任务切换时保存和恢复 DPP 寄存器。严格上来说 CPUID 不是用户寄存器, 然而因为无论如何该寄存器都不可写入, 不需要规定对其进行保护。

GPR 也不受存储器保护机制的保护。由于 CP 自身受到保护, 也确保了对 DPRAM 的保护。类似的, 映射到局部寄存器组的 GPR 也不受存储器保护机制的保护。

### 6.3.6 中断和 PEC 处理

由 CPU 处理的任何中断将保护级别自动切换到 0, 这点对于外设中断、调试器中断、硬件和软件强制中断有效。随后的中断服务子程序 (ISR) 始终从保护级别 0 开始启动, 因此 ISR 通常可访问所有系统资源。之后 ISR 自身可降低保护级别并在一定保护限制下执行用户代码。

也可通过 PEC 传送 (即在两个存储器地址之间进行快速数据传送) 服务中断请求。由 CPU 执行 PEC 传送而无保护。通常在特权模式下 (即保护级别 0), 通过编程 PEC 控制寄存器仍可确保保护, 只有在正确保护级别下, 才能执行 PEC 传送。配置过程中, 软件应当检查 PEC 源指针和目的指针 (根据允许的访问许可) 和 PEC 控制寄存器的正确性。当使用连续模式时, 需要特别留意, 如果 PEC 将来不占据某个区域则在配置时软件无法顾及这点。为了支持该模式 (由特权软件执行的) 需要额外的运行时间进行地址检查或者应该避免使用该模式。

### 6.3.7 RETI 指令的特殊处理

PSW 和保护级别选择标志 (PSW.PL0/1) 在保护机制之下进行处理, 由硬件检测对 PSW 的显性写操作并检查是否在正确的保护级别下触发该操作。不允许进行访问的情况下, 产生一个强制中断, 由保护逻辑避免对 PSW 的修改。

但是 PSW 可被硬件隐性的修改, 保护逻辑几乎无法管理该硬件更新。通过执行 RETI 指令触发 PSW.PL0/1 的硬件更新。只要 PSW (和 PSW.PL0/1) 值来自堆栈 (未受任何用户代码控制), 这些 PSW 硬件更新基本上无关紧要。但是因为不可能禁止用户代码对堆栈进行控制 (用户代码可能使用局部堆栈并对其进行写访问), 唯一的方法是禁止不受信任的用户代码使用 RETI 指令。RETI 指令将会作为一种保护指令受到特殊

处理，只有当保护级别为 0 时，才能执行 RETI。该处理方式是与在保护级别 0 才能处理中断的事实一致，因而应当在相同的保护级别之下从中断返回。

### 6.3.8 上下文切换操作

处理器内核在内部指令（自动插入到流水线中）的帮助下执行上下文切换机制。通常，自动插入的指令应当与引起自动插入的指令运行在同样的保护级别之上。然而，因为上下文切换实际上为可中断操作、某些情况下完成上下文切换的操作可能被延迟，这些上下文切换自动插入指令必须执行而无需考虑保护机制。这意味着，当执行这些指令时，不执行保护检查和/或保护检查被忽略。结果是，只有 CP 更新操作（不是由自动插入指令，而是由上下文切换指令自身执行的）才在保护机制之下执行。在保护机制之下，也不执行寄存器保存至 DPRAM 或从 DPRAM 读取 GPR 的操作。在此情况下，如有需要，则由软件确保 DPRAM 受到保护，软件可检查是否允许访问编程到 CP 的值所寻址的地址范围。

### 6.3.9 调试器访问许可

即使存储器保护有效，调试器也必须能够访问所有存储器空间，包括 IO 区域（即 SFR）。OCDS/Cerbus 基本实现了两种访问系统资源的机制：

- 触发 CPU 执行包含访问资源代码的监控程序（调用监控程序）
- 插入任意自身可访问任何资源的指令

当使用第一种机制时，从插入 ITRAP 指令开始，调试器将自动运行在特权模式，即保护级别 0。如 [章节 6.3.2.1](#) 中的定义，中断之后，自动进入到该保护级别（在此情况下，是在插入调试 TRAP 指令之后）。

当使用第二种机制时（如果 CPU 暂停时情况相同），插入的指令运行而无需存储器保护。CPU 对调试器插入的指令保持跟踪，并禁止对该指令进行保护检查。

对于由调试器访问 OCDS/Cerbus/MCDS SFR，仅需要确保调试监控软件（用于编程调试逻辑）能够以最小的软件开销访问这些寄存器，而对用户代码无任何影响。因为调试监控程序将始终在保护级别 0 之下执行，期望允许对整个存储空间进行访问。同样的，通过插入指令的方式执行对这些寄存器的访问则无上述限制。

### 6.3.10 无效访问强制中断

如果对保护区域进行访问，则会产生一个无效访问强制中断。

共定义三种强制中断：

MPR 存储器保护，读

MPW 存储器保护，写

MPX 存储器保护，执行

它们被定义为 B 类强制中断，分别被映射到 TFR.10, 9, 8 (MPR 为 TFR.10, MPW 为 TFR.9, MPX 为 TFR.8)。TFR 寄存器的完整描述，请参考相关章节中的硬件强制中断的描述。

请注意执行不确定的访问时，不是必须执行强制中断，这就是为什么当知道指令不再被取消时（也就是说，当指令进入到执行阶段），才产生这些中断。

已经存在的强制中断 **PRTFLT** 保护错误强制中断也用于指示在与保护级别 0 不同的保护级别之上执行 **RETI** 指令。即使 **RETI** 引发一个保护错误强制中断，该指令仍被正常执行。

### 6.3.10.1 取消操作

MPU 将检测会引发违反保护的指令，但是不能完全压制这些指令执行，只有引发保护错误的或源于导致保护错误的指令的写操作将会被取消。因为读操作在流水线中会在非常短时间内被执行（有时不确定），因此读操作不能被取消，然而读数据将不会被相应指令写入到任何存储器映射地址。

有一些与上述取消读操作的基本规则不符的例外情况，尤其是，对于执行 2 个写操作的指令，有时第一个写操作不能被取消。下面给出具体情况：

- **SCXT** 指令，当检测到 **mem** 操作数违反读保护或 **reg** 操作数违反写保护时（对于指令 **SCXT reg, mem**），堆栈写操作无法避免。
- **CALLS**, **PCALL** 指令，当检测到第二个堆栈地址（IP 应当被压入该地址）违反写保护时，第一个堆栈写操作（**CSP** 或 **PCALL** 指令时的 **reg**）无法取消。当执行该指令时（第一次压栈在允许地址内，第二次压栈在不允许的地址内），该情况假设堆栈已经超过了允许的地址区域。

结果是，读操作不能被取消，即使 MPU 检测到违反保护的情况，仍然执行对 IO 空间的破坏性读操作。

事实上，执行操作不能被取消，即使它们触发执行保护错误，系统指令以及它们的相应动作可能仍被执行。例如，在相应的硬件强制中断程序能够被执行之前（一旦退出空闲模式就立即执行），**IDLE** 指令可能仍将 CPU 带入到空闲模式。

## 6.4 MPU 的初始化和使用

### 6.4.1 设置保护

本节简要描述初始化和使用保护系统所需的软件序列。还将分析需要的开销（实时性能），共分析在 12 个地址范围内的实现。

用于设置保护的初始化序列是：

- 如果保护未被禁止，则禁止保护（复位之后保护被禁止），将 1 写入到 PM0。
- 编程范围寄存器，1 写入 PRA，24 写入 PRD（绝对最大值，假设使用所有地址范围）
- 编程保护模式寄存器，对 6 个 PMx 进行写操作。
- 使能保护，将 1 写入到 PM0。该最后的写操作可和上面的 PM0 写操作一起执行，但是在此情况下，最后写入该寄存器时必须特别小心。

当使用不同保护级别的应用或软件构件可准确映射到保护组时，该代码序列能够建立保护系统且在运行期间使用 MPU 时不会产生额外开销。初始化阶段之后，一旦需要改变保护级别，则必须选择相应的保护组（改变 PSW.PL）。这是运行期间唯一要进行的附加操作。

该初始化过程以及保护寄存器内的任何改变，应当始终先禁止保护，然后通常在保护级别 0 之下执行此类操作。请注意当保护配置以及激活需要立刻生效的情况下，软件必须考虑到在执行下一个受影响的指令之前（例如从最近被写入的寄存器读取数据），最近的写操作将激活保护。正如 [章节 6.2.2](#) 中的解释，当保护被激活时，没有可用于清空流水线的硬件机制。

在保护需求不能映射到保护组的情况下，需要在运行过程中进行重新编程。最差的情况是该范围必须重新编程，此时需要和初始化过程相似的软件序列。然而，情况可能仅仅是一些已经定义好的范围需要激活/解除保护，在此情况下，仅需要对保护模式寄存器进行重新编程。假设此时已经知道要使用的范围，因此在决定改变哪些设置之前，必须读取 PSW.PL1/0。重新编程期间额外开销来自 EINIT 保护。执行 EINIT 指令之后，对 PMU 寄存器的重新编程仅需要通过进入未保护模式（4 个给出密码的写指令命令序列）而暂时解除保护。重新编程之后，当然需要恢复 EINIT 保护（再次使用 4 个写指令的命令序列）。请注意对于当前未激活的（即通过 PSW.PL 进行选择）以及不会因重新编程而变为有效的保护寄存器，仍然可无需禁止保护而重新编程。

### 6.4.2 改变保护级别

通过显性写访问 PSW 改变保护级别时，需要特别注意。因为任何对 PSW 的写操作都是立即生效的。如果在执行代码/任务切换之前，要求将特权代码处理保护写入到 PSW，则特权代码自身的级别也会改变并且最终不能执行代码/任务切换（例如函数调用）。为了避免当写入 PSW 时，设置立即生效，可使用下列小窍门：堆栈操作和调用函数使用 RETI/RETP 指令，那么带有新保护级别的新 PSW 值必须从堆栈中恢复，而不

是由指令显性写入 PSW。然后，RET 指令将在正确的时间用正确的保护级别更新与新任务相关的 PSW。

#### 6.4.3 由不具有特权的软件执行特权代码

还可能由不具备特权（不受信任）的软件触发特权软件构件（可信的）。不具备特权的部分必须将控制权交给特权部分，可通过执行软件 TRAP 指令实现。该指令自动将保护级别改变至 0。这种机制允许，例如应用软件触发底层驱动并从 ISR 的用户部分返回到 ISR 本身，即由操作系统或软件内核进行部分处理。

#### 6.4.4 快速任务切换

对于触发任务切换的软件来说，不受操作系统或软件内核控制的任何任务切换都被当作可信的任务进行处理。这是因为如果运行在具有足够保护级别（通常为级别 0）的软件控制之下，都不能执行切换，则该任务不可能显性的改变保护级别。

#### 6.4.5 寄存器组选择

因为 PSW.BANK 位域受到保护机制的保护，寄存器组选择（全局或局部 1/2）将由运行在足够访问许可（即足够的保护级别，通常为级别 0）下的软件进行处理。应当在运行于级别 0（所有 ISR 的起始级别）的 ISR 部分完成组选择操作。

#### 6.4.6 调试器使用示例

本节描述使用存储器保护时如何调试系统。

下面给出 4 种使用示例：

- 用户想找到 MPU 强制中断的原因
- 用户想调试 MPU 强制中断程序
- 用户不想调试，只想通过 Cerberus 查询一个变量
- 用户想要在进行调试时不受保护系统的影响

对于第一种使用示例，可使用标准调试资源（OCDS）在处理强制中断的 ISR 上设置 IP 断点。一旦进行到此处，已经知道哪种访问类型导致了中断，因为根据异常情况的类型，需要调用不同的强制中断子程序，因此可通过所调用的强制中断子程序隐性的或者通过读取 TFR 标志显性的了解异常情况的类型。通过读取压栈的 PSW 得到相应的保护级别。导致强制中断的相关 IP 无法直接得到，但线性跟随其后指令的 IP 可以直接得到（保存在堆栈中）。

对于第二种使用示例，即调试 MPU 强制中断子程序，可通过与调试任意强制中断子程序相似的方式完成，并可能通过在相关 ISR 上设置断点的方式启动。

对于第三种使用示例，不论 MPU 是否使能，可在任意时刻执行可变的查询动作。即使 MPU 被使能，调试器必须始终访问所有系统资源。

对于第四种使用示例，即不受 MPU 影响的情况下进行调试，调试器可通过显性的禁止 MPU (PM0.PROTEN) 来实现。因为调试器不知道系统复位之后，应用何时将会使能 MPU，调试器必须监控 MPU 的状态 (PM0.PROTEN) 并且一旦 MPU 被使能就立刻重新禁止 MPU。然而，该使用情况非常奇怪，如果应用导致 MPU 异常情况，就应当在调试期间了解到该情况，因为调试的目的就是要在真实的系统行为下进行。



## 7 中断与异常情况控制

XE16xyM 提供了几种中断机制，可对服务请求作出快速、灵活的响应。这些服务请求由各种内部或外部中断源产生，并送至微控制器。对各种异常情况的处理方式相似：

- 由中断控制器（ITC）产生中断
- 由外围事件控制器（PEC）发出 DMA 传送
- 由 TRAP 指令、错误或特定系统状态引发强制中断

### 正常中断处理

CPU 暂时挂起当前执行的程序，转入中断处理子程序，以服务中断请求设备。因此，当前程序状态（IP、PSW 以及分段模式下的 CSP）被保存到系统堆栈中。

XE16xyM 使用优先级排序机制（共分为 16 个优先级），以确定多个中断请求的执行顺序。

### PEC 中断处理

与正常的中断处理相比，采用 XE16xyM 集成的**外围事件控制器**（PEC）可以更快的服务中断请求设备。被中断请求触发之后，PEC 可在任意两个存储器地址单元之间执行一个字或一个字节的数据传送。PEC 传送期间，CPU 正常程序的执行被中断。不需要保存内部程序状态信息。PEC 服务和正常中断处理使用相同的优先级排序机制。

### 强制中断功能

在指令执行期间出现异常情况时，**强制中断功能**被激活。还可通过外部服务请求引脚  $\overline{\text{ESRx}}$ （例如，用于实现与  $\overline{\text{NMI}}$  相似的行为）从外部触发强制中断。XE16xyM 提供了几种硬件强制中断功能，用来处理程序执行期间出现的错误和异常情况。硬件强制中断的优先级最高，会被系统立即响应。软件强制中断功能由 TRAP 指令引发，它产生一个具有特定中断向量的软件中断。对所有类型的强制中断，当前程序状态都保存到系统堆栈。

### 外部中断处理

尽管 XE16xyM 未提供专用的外部中断输入引脚，但允许将部分输入引脚配置为中断输入。可从标准输入或外部服务请求引脚  $\overline{\text{ESRx}}$  中选择中断（强制中断）输入引脚。可供选择的中断输入引脚的详细信息请参考**外部中断**章节。



## 中断源和中断请求通路

为了激活中断源并正确送出中断请求，必须要考虑对下述片上元件进行编程：

- 每个外设的中断控制
- IMB 存储器控制器 **中断产生**
- SCU **外部请求单元 (ERU)**
- **SCU 中断产生**

如果要使用外部中断源，还需要编程设置端口。

## 7.1 中断系统结构

XE16xyM 具有 96 个单独的中断节点，构成 16 级中断优先级，每级内又分 8 个子级（组优先级）。为支持软件设计的模块化和一致性，大多数中断或 PEC 请求源都对应有独立的中断控制寄存器和中断向量。中断控制寄存器包含相应中断源的中断请求标志、中断使能位和中断优先级。每个中断源请求由特定事件激活，这由相应器件所选择的工作模式决定。为了有效的利用资源，多个中断源的中断节点可以合并成一个。这些中断节点可以由多个中断源请求激活，例如串行接口中的各种错误。不过，可以通过相应外设控制寄存器中的特定状态标志来确定错误类型。通过 **中断节点共享** 来实现中断节点的共享。

XE16xyM 具有一个向量化的中断系统。在该系统中，存储空间中的特定向量地址保留用作复位、强制中断和中断服务功能。一旦发生中断请求，CPU 会跳转到相应中断源所对应的向量地址上。B 类硬件强制中断共用同一个中断向量。由强制中断标志寄存器 (TFR) 中的状态标志来确定引起强制中断的异常事件类型。特殊的软件 TRAP 指令的向量地址由指令中的操作数域指定，该操作数为一个 7 位的强制中断编号。

这些保留的向量地址构成一个跳转表，位于 XE16xyM 地址空间某地址段的低端（由寄存器 VECSEG 选定）。跳转表由适当的跳转指令组成，跳转指令将 CPU 的控制权移交给中断或强制中断服务程序。这些服务程序可以位于地址空间的任何位置。跳转表的入口指向所选代码段的最低端。每个入口占用 2、4、8 或 16 个字（由 CPUCON1 寄存器中的位域 VECSC 决定），至少能够存放一条双字指令。相应的中断向量地址可以通过中断向量编号乘以所选的中断向量间距 ( $2^{(VECSC+2)}$ ) 计算得到。

所有挂起的中断请求都要经过仲裁。仲裁胜出的中断请求连同其中断优先级和操作请求一起发送给 CPU。CPU 根据仲裁胜出者所请求的功能（正常中断、PEC、跳转表缓存等）触发相应的操作。

中断全局使能的情况下，如果中断请求源的优先级高于当前 CPU 的优先级，CPU 会接受该操作请求；如果中断请求源的优先级比当前 CPU 的优先级低（或相等），该中断请求保持挂起。

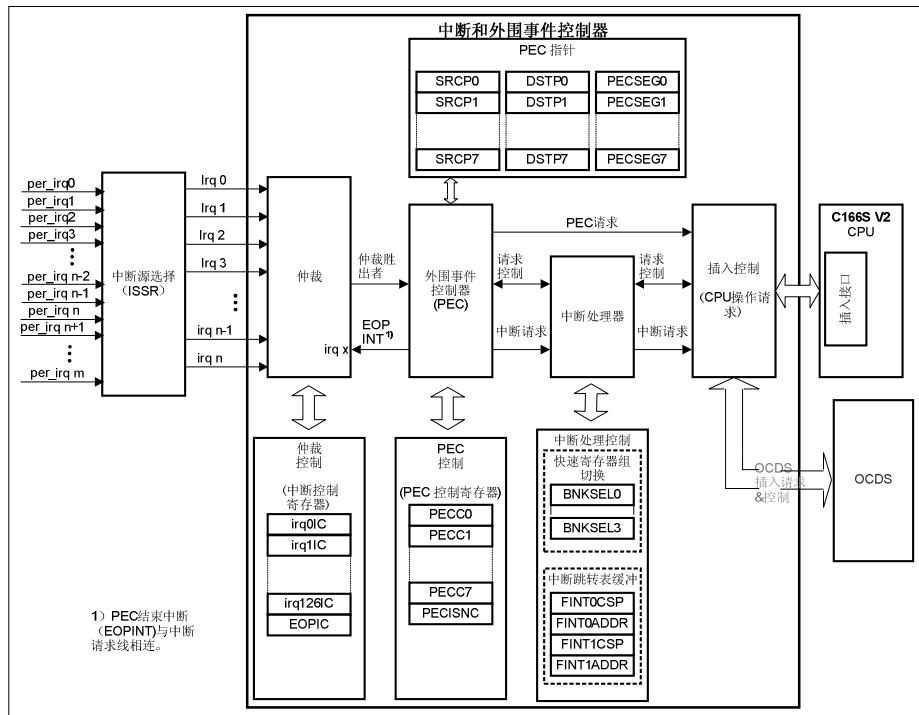


图 7-1 中断和 PEC 控制器框图

## 7.2 中断仲裁

XE16xyM 中断仲裁系统能够处理来自多达 96 个中断源的中断请求。内部外设或外部输入都可以触发中断请求。支持增强 PEC 功能的“PEC 结束”中断可被内部连接至中断请求线之一。

接收到使能的中断请求之后，立即启动仲裁过程。只要有使能的中断请求挂起，则仲裁一直有效。

由中断控制寄存器 xxIC（此处和下文中的 xx 代表对应中断源的助记符）控制每条中断请求线。由中断请求事件置位相应中断控制寄存器内的中断请求标志（位 xxIC.IR）。如果程序置位相关的中断请求位，还可由软件触发中断请求。

如果请求位已经被置位，且由同一个控制寄存器中的中断使能位（位 xxIC.IE）使能，则在下一个时钟周期开始中断仲裁。然而，如果当时正在执行仲裁周期，则新的中断请求被延迟到下一个仲裁周期。如果中断请求（或 PEC 请求）被内核接受，相应的中断请求标志被自动清除。

新的仲裁周期开始时，所有挂起的中断请求被认为是同时发生的。在一个仲裁周期内，仲裁过程与中断实际的请求时间无关。

XE16xyM 使用三级优先级排序机制进行中断仲裁，见图 7-2。

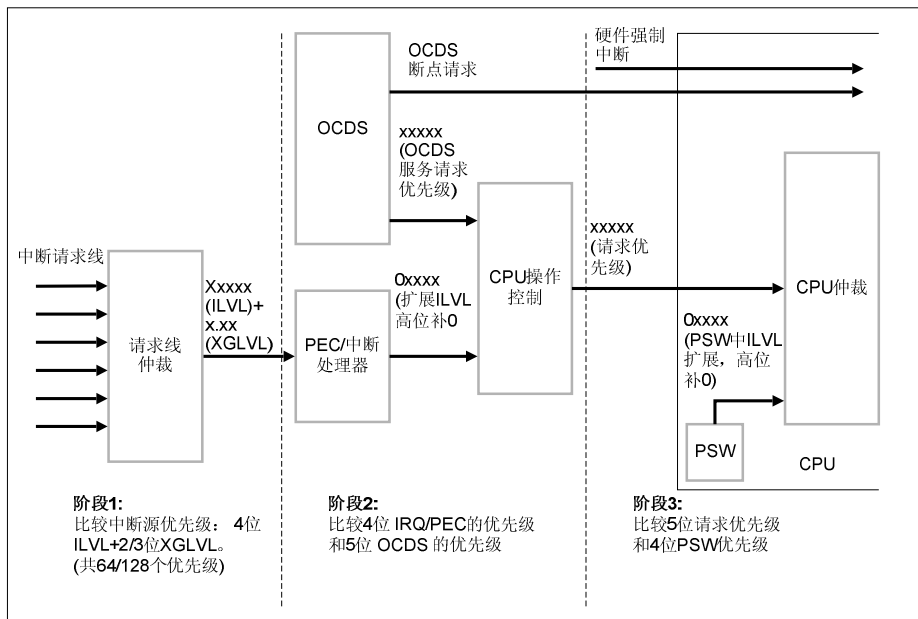


图 7-2 中断仲裁

在仲裁第一阶段，比较中断请求线的优先级。每个中断请求的优先级由中断优先级和组优先级组成。每个中断请求线中断的优先级被编程到相应寄存器 `xxIC` 内的 4 位位域 `ILVL` 中。每个中断请求线的组优先级由寄存器 `xxIC` 的 2 位位域 `GLVL` 和扩展位 `GPX` 决定。`GPX` 和 `GLVL` 组合之后形成 3 位 (扩展) 组优先级 `XGLVL`，控制同一优先级 (共分为 16 级) 内的多达 8 个子优先级。

**注：**所有被使能且设定有相同中断优先级 (`ILVL`) 的中断请求必须具有不同的组优先级。否则可能产生错误的中断向量。

在仲裁第二阶段，将第一阶段胜出的中断请求和 `OCDS` 服务请求的优先级相比较。`OCDS` 服务请求无需经过第一阶段，直接进入 CPU 操作控制单元。CPU 操作控制单元忽略中断/PEC 请求的组优先级，仅处理其中断优先级 (`ILVL`)。为了和 `OCDS` 服务请求的 5 位优先级进行比较，先将中断/PEC 请求的 4 位 `ILVL` 扩展为 5 位，最高位补 0 (`MSB=0`)。这意味着，`MSB=1` 的任意 `OCDS` 请求总是在仲裁第二阶段胜出。但

是，如果 MSB=0 的 OCDS 请求和具有相同优先级的中断/PEC 请求发生冲突时，中断/PEC 请求在该仲裁阶段胜出，被送至 CPU。

在仲裁第三阶段，将第二阶段胜出的中断请求与当前 CPU 任务的优先级相比较。如果中断请求的优先级高于当前 CPU 优先级（PSW 寄存器中位域 ILVL），并且中断被 PSW 寄存器中的全局中断使能位 IEN 全局使能的情况下，中断请求才会被 CPU 接受。IEN 位被清除的情况下，CPU 拒绝所有的请求。为了和第二阶段胜出的中断请求的 5 位优先级进行比较，CPU 当前 4 位优先级 PSW.ILVL 需要先扩展为 5 位，最高位补 0（MSB=0）。这意味着，MSB=1 的任何中断请求总是中断 CPU 的当前任务。若中断请求的优先级低于或等于当前 CPU 任务的优先级，请求保持挂起状态。

*注：CPU 缺省的优先级为 0000<sub>B</sub>，因此当请求优先级为 0000<sub>B</sub> 时，虽然进行仲裁，但该优先级的中断请求永远不会被 CPU 接受。然而，每个被使能的中断请求（包括被 CPU 拒绝的中断请求和优先级为 0000<sub>B</sub> 的请求）都会将 CPU 从空闲状态唤醒，而与全局中断使能位 PSW.IEN 的设置无关。*

### 7.3 中断控制

所有中断控制寄存器的结构都相同。中断控制寄存器的低 9 位包含一轮优先级仲裁（仲裁周期）期间相关中断源的全部中断控制和状态信息，高 7 位保留。所有的中断控制寄存器都可位寻址，所有控制位都可由软件读/写。因此，只需要一条指令即可对每个中断源进行编程或修改。通过指令读取中断控制寄存器时，操作单位是字类型，高 7 位（15...9）将返回 0。建议用户始终向这些位元写入 0。

任何 IC 寄存器的 IR 位的类型是“rwh”，中断发生时由硬件置位。当中断源被使能时，如果软件需要写访问 IC 寄存器，那么软件写操作可能和硬件对 IR 的访问发生冲突。为了解决该冲突情况，所有的 xxIC 寄存器都位于可位寻址的存储器区。因此可以使用 C166 位修改指令并推荐用户使用该方式。这些指令提供一种特殊的“保护屏蔽”特性，防止 IR 被软件写操作无意修改。详情请参考 CPU 位操作单元一节。

下面给出的中断控制寄存器结构适用于所有 xxIC 寄存器。

xxIC

中断控制寄存器

(E)SFR (xxxx<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	GPX	IR	IE	ILVL				GLVL	
r	r	r	r	r	r	r	rw	rwh	rw	rw				rw	

符号	位序号	类型	功能描述
GPX	8	rw	组优先级扩展

符号	位序号	类型	功能描述
			定义组优先级高位的值
<b>IR</b>	7	rw	<b>中断请求标志</b> 0 <sub>B</sub> 无挂起的中断请求 1 <sub>B</sub> 该中断源已产生中断请求
<b>IE</b>	6	rw	<b>中断使能控制位</b> (单独使能/禁止特定中断源) 0 <sub>B</sub> 中断请求被禁止 1 <sub>B</sub> 中断请求被使能
<b>ILVL</b>	[5:2]	rw	<b>中断优先级</b> F <sub>H</sub> 最高优先级 ..... 0 <sub>H</sub> 最低优先级
<b>GLVL</b>	[1:0]	rw	<b>组优先级</b> 3 <sub>H</sub> 最高优先级 ... .. 0 <sub>H</sub> 最低优先级
<b>0</b>	[15:9]	r	<b>保留</b> 读操作返回 0；应写入 0。

通过指令访问中断控制寄存器时，操作单位是字类型，寄存器的高 7 位（15...9）读取时返回 0，写入值被忽略。建议用户始终向这些位元写 0。

**中断请求标志:** 只要相关的中断源产生服务请求，中断请求标志由硬件置位。一旦进入中断服务程序或 PEC 服务，请求标志被自动清除。在进行 PEC 服务时，如果所选 PEC 通道的寄存器 PECCx 中位域 COUNT 递减至 0 且位 EOPINT 清零，中断请求标志保持置位状态。这样可产生一个具有相同优先级的正常 CPU 中断，以响应 PEC 传送的完成。

*注：通过软件修改中断请求标志和通过硬件置位或清除的效果相同。*

**中断使能控制位**决定中断节点是（使能）否（禁止）参与仲裁。无论中断是否被使能，只要产生中断请求，相关的请求标志就会置位。因此，即使在节点被禁止的情况下，也可以通过查询 xxIR 来判断是否产生中断请求。

*注：在这种情况下，xxIR 不会被自动清除，必须由软件清除。*

### 7.3.1 中断优先级与组优先级

4 位位域 ILVL 指定用于仲裁同时有效的中断请求的优先级。ILVL 值越大，优先级越高。因此，0000<sub>B</sub> 代表的优先级最低，1111<sub>B</sub> 最高。

当多个具有相同优先级的中断请求同时被激活时，根据位域 GPX 和 GLVL 级联构成的组优先级来进行第二级仲裁，选出其中组优先级最高的中断请求。同样，组优先级（位域 GPX 和 GLVL 级联构成）值越大，优先级越高。因此，000<sub>B</sub> 代表的组优先级最低，111<sub>B</sub> 最高。

*注：所有使能且设定有相同中断优先级（ILVL）的中断请求必须具有不同的组优先级。否则会产生错误的中断向量。*

进入中断服务程序时，将仲裁胜出的中断源的优先级（高于 CPU 当前优先级）复制到 PSW 寄存器的位域 ILVL 中，在该复制操作之前需要先将旧的 PSW 内容压入堆栈。

XE16xyM 中断系统允许嵌套多达 15 层不同优先级的中断服务程序（0 优先级不会被仲裁）。

若相关 PEC 通道被使能且配置适当（请参阅章节 7.10.4），优先级为 15...8 的中断请求（即 ILVL=1XXX<sub>B</sub>）可以由 PEC 服务。优先级为 7...1 的中断请求只能按照正常中断进行处理。

### 7.3.2 寄存器 PSW 的全局中断控制功能

是否接受中断请求取决于 CPU 的当前优先级（PSW 寄存器的 ILVL 位域）以及 PSW 寄存器中的全局中断使能控制位 IEN（见章节 5.8）。

**CPU 优先级位域 ILVL** 定义了当前 CPU 操作的优先级。该位域指示当前执行的程序的优先级。进入中断服务程序时，用被响应的中断请求的优先级更新该位域。响应中断请求之前，旧的 PSW 被保存到系统堆栈中。CPU 优先级决定了 CPU 能够响应的最低优先级。任何低于或等于 CPU 优先级的请求不予应答。CPU 当前优先级可以通过软件进行修改，用来控制哪些中断请求源能够被应答。PEC 传送不会真正中断 CPU，只是“窃取”一个时钟周期而已。因此，PEC 服务并不会影响 PSW 中的 ILVL 位域。

硬件强制中断将 CPU 优先级切换到最高级（即 15），从而在执行异常强制中断服务程序时，不会响应任何中断和 PEC 请求。

*注：TRAP 指令不改变 CPU 的优先级，软件调用强制中断服务程序时，可能被更高优先级的请求中断。*

**中断使能位 IEN** 全局使能或禁止 PEC 操作以及正常 CPU 中断操作。当 IEN 清零时，CPU 不接受任何新的中断请求。当 IEN 置位时，所有已被分别使能的中断请求源将被全局使能（通过各自中断控制寄存器中的中断使能位来分别使能各中断请求源）。强制中断属于非屏蔽中断，因此不受 IEN 位的影响。

*注：为了产生中断请求，中断源还必须由相关控制寄存器中的中断使能位使能。*

**寄存器组选择位域 BANK** 决定 CPU 操作当前使用的寄存器组。当 CPU 进入中断服务程序时，该位域被更新，选择与中断服务请求相关的寄存器组：

- 优先级为 15...12 的请求使用由相应 BNKSEL 寄存器中位域 GPRSELx 预先选择的寄存器组
- 优先级为 11...1 的请求始终使用全局寄存器组，即 BANK = 00<sub>B</sub>
- 硬件强制中断始终使用全局寄存器组，即 BANK = 00<sub>B</sub>
- TRAP 指令不改变当前寄存器组

### 7.3.3 可选的中断禁止

软件执行期间，可暂时禁止或使能中断请求。根据应用的当前状态，可能需要禁止特定中断源。需要实现时间紧急代码序列的确定性执行时，该特性尤其有用。

可在三个不同级别上，禁止和使能 XE16xyM 的中断请求：

- 对于特定代码序列，禁止所有中断请求
- 全局禁止所有中断请求
- 禁止单个中断请求

**ATOMIC 和 EXTend 指令**在随后的 1...4 条指令期间自动禁止所有中断请求。这对旗语处理等应用场合非常有用，而且不可分的指令序列之后，不需要重新使能中断系统。

使用单个指令实现**全局中断控制**：

```
BCLR IEN      ;Clear IEN flag (causes pipeline restart)
```

**特定中断控制**通过控制相应中断控制寄存器的使能位实现。

```
BCLR T2IE     ;Clear enable flag to disable intr.node
```

如果中断请求与使能标志的清除发生冲突，然而因为流水线的影响，相应节点被禁止之后，中断请求仍可能被执行。

如果某应用必须禁止此情况的发生，必须使用下述代码序列，以确保中断节点被禁止之后，来自该中断源的中断请求不被服务。

```
BCLR IEN      ;Globally disable interrupts
BCLR T2IE     ;Disable Timer 2 interrupt node
JNB T2IE, Next ;Any instruction reading T2IC can be used
Next:         ;(assures that T2IC is written by BCLR
              ;before being read by JNB or other instr.)
BSET IEN      ;Globally enable interrupts again
```

请注意上述代码序列会无分别的控制全局使能标志。如果一定不能改变全局设置，则可使用下述改进的代码序列：

```
JNB IEN, GlobalIntOff
BCLR IEN      ;Globally disable interrupts
```

```

BCLR T2IE          ;Disable Timer 2 interrupt node
JNB T2IE, Next     ;Any instruction reading T2IC can be used
Next:              ;(assures that T2IC is written by BCLR
                  ;before being read by JNB or other instr.)
BSET IEN           ;Globally enable interrupts again
JMPR cc_uc, Continue
GlobalIntOff:      ;Interrupts are globally disabled anyway
BCLR T2IE          ;Disable Timer 2 interrupt node
JNB T2IE, Continue ;Reading T2IC can be omitted if the next
Continue:          ;few instructions do not set IEN
...

```

可用 C 宏代码轻松实现同样的功能:

```

#define Disable_One_Interrupt(IE_bit) \
{if(IEN) {IEN=0; IE_bit=0; while (IE_bit); IEN=1;} else \
{IE_bit=0; while IE_bit);}}
Usage Example:
Disable_One_Interrupt(T2IE) ; // T2 interrupt enable flag

```

当 ATOMIC 或 EXTend 序列开始时，保留中断仲裁状态。ATOMIC 或 EXTend 序列之后，处理被接受的请求，因此，下述代码序列可能无法产生需要的结果:

```

AvoidThis:
ATOMIC #3
NOP
BCLR T2IE          ;Disable Timer 2 interrupt node
NOP                ;Timer 2 request may be processed
                  ;after this instruction!!!

```

### 7.3.4 中断类管理

一类中断由一组重要程度相同的中断源组成。即从系统角度讲，它们具有相同的优先级。同类的中断不能相互中断。XE16xyM 具备两个相关特性以支持该功能:

- 使用相同的中断优先级 (ILVL)，并为每个成员分配一个专用的组优先级，可以建立**最多 8 个成员的中断类**。由中断控制器自动生成和处理该功能。
- 使用相邻的中断优先级 (ILVL) 和相应的组优先级（每个 ILVL 包含 8 个组优先级），可以建立**多于 8 个成员的中断类**。该中断类中的每个中断服务程序都会将 CPU 的优先级设为该类中的最高优先级。所有同级或更低优先级的请求将被拒绝，即该类中的其它请求不会被接受。

下面的例子建立了 3 类中断，每类中断包含 2 或 3 级中断优先级，取决于该类中断的成员个数。在中断类 2 中，优先级为 6 的中断通过将当前 CPU 优先级修改成 8（该类



中的最高优先级），可禁止中断类 2 中的所有其它中断源。此时，中断类 1 的中断请求或 PEC 请求仍能得到服务。

通过这种方式，所有的中断源（PEC 请求除外）被分配给 3 类中断优先级，而不是像硬件支持的那样分配给 7 级不同的优先级。

**表 7-1 软件控制的中断类别（示例）**

ILVL (优先级)	组优先级								说明
	7	6	5	4	3	2	1	0	
15									最多 8 路 PEC 服务通道
14									
13									
12	X	X	X	X	X	X	X	X	中断类 1 9 个中断源分布在 2 级中断优先级上
11	X								
10									
9									
8	X	X	X	X	X	X	X	X	中断类 2 17 个中断源分布在 3 级中断优先级上
7	X	X	X	X	X	X	X	X	
6	X								
5	X	X	X	X	X	X	X	X	中断类 3 9 个中断源分布在 2 级中断优先级上
4	X								
3									
2									
1									
0									无服务！

## 7.4 中断向量表

XE16xyM 具有一个向量化的中断系统。该系统保留了一组特定的存储单元，发生相应的触发事件时，系统会自动访问这些单元。触发事件包括：

- 复位（硬件、软件、看门狗）
- 强制中断（系统出错触发的硬件强制中断或 TRAP 软件强制中断）
- 中断服务请求

一旦接受请求，CPU 将跳转到触发源的相关地址。通过向量位置可直接确认引起请求的中断源，但以下情况例外：

- B 类硬件强制中断共用一个中断向量。强制中断标志寄存器（TFR）中的状态标志用于确定是哪种异常事件引起了强制中断，详见[章节 7.9](#)。
- 多个中断请求（例如一个模块内的多个中断请求）可能会共用一个中断节点。此时，将通过附加的标志位来识别请求源，从而可通过软件分别对每个请求源进行处理。详见[章节 7.14.2](#)。
- 使用中断跳转表缓存特性，详见[章节 7.5](#)。

这些保留的向量地址在 XE16xyM 地址空间中构成一个向量表。向量表中包含着适当的跳转指令，跳转指令将 CPU 的控制移交给中断或强制中断服务程序。这些服务程序可以位于地址空间的任何位置。向量表的位置和结构可以编程设定。

向量段寄存器 VECSEG 定义了向量表所在的段（可以使用保留区域之外的所有段）。

寄存器 CPUCON1 中的位域 VECSC 定义了两个相邻向量之间的间距（可以为 2、4、8 或 16 个字）。CPUCON1 寄存器的描述见[章节 5.4](#)。

每个向量都有一个相对于向量表段基地址（由 VECSEG 决定）的偏移地址。该偏移量可通过向量编号乘以由 VECSC 设定的向量间距计算得到。

**表 7-9** 列出 XE16xyM 中的所有中断请求源或 PEC 服务请求源、对应的中断向量地址、中断向量编号，以及中断控制寄存器。

*注：未被相关模块使用或在实际衍生产品中未和模块连接的中断节点，都可以通过软件置位相应 IR 标志产生软件控制的中断请求。*

## VECSEG

向量段指针

SFR (FF12<sub>H</sub>)

复位值: 00XX<sub>H</sub>

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

符号	位序号	类型	功能描述
<b>VECSEG</b>	[7:0]	rwh	向量表的段编号
<b>0</b>	[15:8]	r	保留 读操作返回 0；应写入 0。

寄存器 VECSEG 的初始用户值由[启动配置和引导程序加载](#)中的设置决定。

## 7.5 中断跳转表缓存

该机制使用向量表地址作为中断服务程序的入口（可被中断控制器 ITC 改写）。对于要求快速响应的中断，XE16xyM 提供一个中断跳转表缓存（也被称为“快速中断”）。ITC 能够向 CPU 传送一个直接作为服务程序起始地址的 24 位向量。该特性省去了通过向量表的步骤（通常省去至少一个分支指令）。因此，避免使用向量表可大大提高中断响应速度。然而，ITC 内的 24 位向量的个数受到限制。

中断优先级大于或等于 12 的两个中断源可使用快速中断。中断跳转表缓存跳过了从中断向量表取指令的步骤，执行一个直接跳转到中断服务程序入口点的跳转指令。由两个中断跳转表缓存寄存器（FINTxCSP、FINTxADDR）控制每个跳转表入口（共两个）的快速中断特性。

每个中断跳转表缓存入口包含一个使能位、相关的仲裁优先级（ILVL 和 GLVL）以及中断服务程序的 24 位地址。请注意在相应的控制寄存器中，只有中断优先级的低两位可选。中断优先级的高两位固定为 11<sub>B</sub>，将允许的中断优先级限定为大于或等于 12。

### FINT0CSP

快速中断控制 0

XSFR (EC00<sub>H</sub>)

复位值: 0000<sub>H</sub>

### FINT1CSP

快速中断控制 1

XSFR (EC04<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	0	0	GPX	ILVL	GLVL	SEG									
rw	r	r	rw	rw	rw	rw									

符号	位序号	类型	功能描述
EN	15	rw	<b>快速中断使能</b> 0 <sub>B</sub> 禁止中断跳转表缓存，不使用快速中断 1 <sub>B</sub> 使能中断跳转表缓存。使用快速中断（直接跳转到中断服务程序），代替从中断向量表取指令的正常操作。
GPX	12	rw	<b>组优先级扩展</b> 该位与 GLVL 联合使用，用来选择相关中断跳转表缓存入口的组优先级（XGLVL）。
ILVL	[11:10]	rw	<b>中断优先级</b> 该位域选择与中断跳转表入口相关的中断优先级的低两位。

**中断与异常情况控制**

符号	位序号	类型	功能描述
			<i>注：中断优先级的高两位固定为 11<sub>B</sub>，因此所得的中断优先级为大于或等于 12。</i>
<b>GLVL</b>	[9:8]	rw	<b>组优先级</b> 与 GPX 位共同确定相关中断跳转表缓存入口的组优先级（XGLVL）。
<b>SEG</b>	[7:0]	rw	<b>中断服务程序段编号</b> 中断服务程序入口点的地址位 23...16 位。
<b>0</b>	[14:13]	r	<b>保留</b> 读操作返回 0；应写入 0。

**FINT0ADDR**

快速中断地址 0

**XSFR (EC02<sub>H</sub>)**

复位值: 0000<sub>H</sub>

**FINT1ADDR**

快速中断地址 1

**XSFR (EC06<sub>H</sub>)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>ADDR</b>															<b>0</b>
rw															r

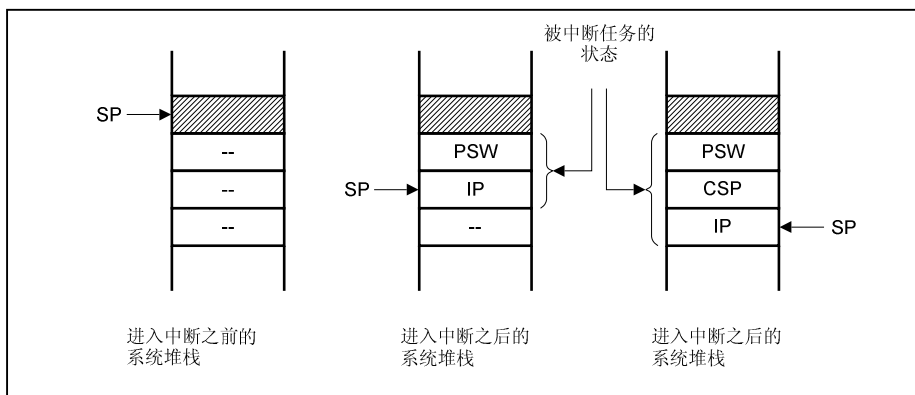
符号	位序号	类型	功能描述
<b>ADDR</b>	[15:1]	rw	<b>中断服务程序的地址</b> 定义中断服务程序入口点的地址位 15...1。
<b>0</b>	0	r	<b>中断服务程序地址位 0</b> 中断服务程序入口点地址的 LSB。因为程序代码是字对齐的，所以该地址位始终为 0。

## 7.6 CPU 状态保存

仲裁之后的中断请求在被真正处理之前，当前工作任务的状态被自动保存到系统堆栈中，这包括要保存 **CPU 状态 (PSW)** 和中断返回地址（执行完中断服务程序，返回被中断的任务所在的地址继续运行）。返回地址由指令指针 (**IP**) 指定，存储器分段模式下，还与代码段指针 (**CSP**) 有关。寄存器 **CPUCON1** 中的位 **SGTDIS** 控制如何保存返回地址。

- 系统堆栈首先接收 **PSW**，然后是 **IP**（不分段模式下）；或者是系统堆栈先保存 **PSW**、然后是 **CSP**、接下来是 **IP**（分段模式下）。如果禁止存储器分段，可使系统堆栈的使用得到优化。
- **CPU 优先级控制位域 (PSW 中的 ILVL)** 更新为被响应的中断请求的优先级，于是，**CPU** 工作在新的优先级上。
- 寄存器组选择位域 (**PSW 中的 BANK**) 相应修改为与中断请求相关的寄存器组。中断请求与寄存器组之间的关联，部分已经被预先定义，部分可以由用户编程。
- 中断请求被响应后，相应的中断请求标志被清除；将请求源对应的中断向量载入 **IP** 和 **CSP**，从中断向量地址中取出中断服务程序的第一条指令（分支指令），进而跳转到真正的服务程序中（使用中断跳转表缓存的情况例外）。所有其它 **CPU** 资源，如数据页指针、上下文指针等不受影响。

当退出中断服务程序时（执行 **RETI** 指令），状态信息从系统堆栈中以“后进先出”的次序弹出。弹出时，需要考虑 **SGTDIS** 的值。



**图 7-3 保存在系统堆栈中的任务状态**

## 7.7 CPU 上下文切换

中断服务程序通常要在堆栈中保存中断服务程序要用到的所有寄存器，并在中断返回前将这些寄存器恢复。XE16xyM 可以自动或者采用一条指令来切换整个 CPU 通用寄存器组（GPR）。这样，中断服务程序可以用其独立的上下文工作（见[章节 5.5.2](#)）。

有两种方式切换上下文：

1. **处理中断时的上下文切换：**自动更新位域 PSW.BANK，从两个局部寄存器组或当前全局寄存器组中选择一个，此时服务程序可以直接使用“自己的上下文”。当服务程序结束时，这个寄存器组被保留。因此，下次调用时，其内容仍然可用。对于优先级为 15...12 的中断，其目标寄存器组可以预先选定。寄存器组选择寄存器 BNKSELx 为每一个优先级提供一个 2 位位域。一旦某中断请求被接受，相应位域值就被复制到 PSW 寄存器的位域 BANK 中，用来选择寄存器组。
2. **由软件显性切换上下文：**由向 CP 或 PSW 寄存器写数据的操作引发该操作。
  - a) 向 PSW.BANK 位域的写操作允许在全局和局部寄存器之间进行切换。
  - b) 向 CP 的写操作允许重新将存储器映射的全局寄存器组映射到另一个存储器地址。

例如，指令“SCXT CP, #New\_Bank”将上下文指针（CP）的内容压入系统堆栈，将立即数“New\_Bank”载入 CP，新 CP 值设置了新的寄存器组。此时中断服务程序可以使用“自己的寄存器”。当服务程序结束时，这个寄存器组被保留，即下次调用时，其内容仍然可用。中断返回（RETI）之前，只需使用“POP CP”指令将原先的 CP 从系统堆栈中弹出，恢复使用原先的全局寄存器组。

*注：中断程序使用的其它资源（如 DPP 寄存器）必须被分别保存和恢复。*

*注：考虑到流水线的操作，上下文切换时，会有一定的时序限制。详见[章节 5.5.2](#)。*

## 7.8 快速寄存器组切换

中断处理器支持一个用于处理正常中断的、被称为快速寄存器组切换的附加增强特性（与 C166 家族相比）。为了加速中断处理，对于优先级大于或等于 12 的中断，内核使用快速通用寄存器（GPR）组切换技术。对于[ILVL = '15<sub>D</sub>'-'12<sub>D</sub>'且 XGLVL = '7<sub>D</sub>'-'0<sub>D</sub>']的每个仲裁优先级，可用特定两位选择寄存器组。选择位位于四个寄存器组选择寄存器 BNKSELx (x = 0...3)内。

下面的表格给出四个寄存器组选择寄存器中的相关位域与仲裁优先级的分配关系：

**表 7-2          寄存器组分配**

ILVL	XGLVL	分配的 GPRSELx 寄存器	ILVL	XGLVL	分配的 GPRSELx 寄存器
15	7	BNKSEL3.GPRSEL7	13	7	BNKSEL2.GPRSEL7
15	6	BNKSEL3.GPRSEL6	13	6	BNKSEL2.GPRSEL6
15	5	BNKSEL3.GPRSEL5	13	5	BNKSEL2.GPRSEL5
15	4	BNKSEL3.GPRSEL4	13	4	BNKSEL2.GPRSEL4
15	3	BNKSEL1.GPRSEL7	13	3	BNKSEL0.GPRSEL7
15	2	BNKSEL1.GPRSEL6	13	2	BNKSEL0.GPRSEL6
15	1	BNKSEL1.GPRSEL5	13	1	BNKSEL0.GPRSEL5
15	0	BNKSEL1.GPRSEL4	13	0	BNKSEL0.GPRSEL4
14	7	BNKSEL3.GPRSEL3	12	7	BNKSEL2.GPRSEL3
14	6	BNKSEL3.GPRSEL2	12	6	BNKSEL2.GPRSEL2
14	5	BNKSEL3.GPRSEL1	12	5	BNKSEL2.GPRSEL1
14	4	BNKSEL3.GPRSEL0	12	4	BNKSEL2.GPRSEL0
14	3	BNKSEL1.GPRSEL3	12	3	BNKSEL0.GPRSEL3
14	2	BNKSEL1.GPRSEL2	12	2	BNKSEL0.GPRSEL2
14	1	BNKSEL1.GPRSEL1	12	1	BNKSEL0.GPRSEL1
14	0	BNKSEL1.GPRSEL0	12	0	BNKSEL0.GPRSEL0



**BNKSEL0**

**寄存器组选择 0                      XSFR (EC20<sub>H</sub>)                      复位值: 0000<sub>H</sub>**

## BNKSEL1

寄存器组选择 1                      XSFR (EC22<sub>H</sub>)                      复位值: 0000<sub>H</sub>

## BNKSEL2

寄存器组选择 2                      XSFR (EC24<sub>H</sub>)                      复位值: 0000<sub>H</sub>

### BNKSEL3

**寄存器组选择 3**                      **XSFR (EC26<sub>H</sub>)**                      **复位值: 0000<sub>H</sub>**

[illegible]

符号	位序号	类型	功能描述
<b>GPRSEL0,</b>	[1:0],	rw	<b>寄存器组选择</b>
<b>GPRSEL1,</b>	[3:2],		00 <sub>B</sub> 全局寄存器组
<b>GPRSEL2,</b>	[5:4],		01 <sub>B</sub> 保留
<b>GPRSEL3,</b>	[7:6],		10 <sub>B</sub> 局部寄存器组 1
<b>GPRSEL4,</b>	[9:8],		11 <sub>B</sub> 局部寄存器组 2
<b>GPRSEL5,</b>	[11:10],		
<b>GPRSEL6,</b>	[13:12],		
<b>GPRSEL7</b>	[15:14],		

注：当前被触发的中断的 GPRSELx 值被自动传送至程序状态字（PSW）中。

## 7.9 强制中断功能

C166SV2 CPU 支持软件和硬件强制中断功能。

### 7.9.1 软件强制中断

TRAP 指令可用于引发软件调用中断服务程序。强制中断编号由强制中断指令的操作数域给定，确定将要用到的、向量表中的向量地址。

执行 TRAP 指令，与在同一向量地址处发生中断的效果类似。PSW、CSP（分段模式）和 IP 压入内部系统堆栈，并跳转到特定的向量地址。执行软件强制中断时，将寄存器 VECSEG 的值载入强制中断服务程序的 CSP。TRAP 指令不影响任何中断请求标志。TRAP 指令调用的中断服务程序必须由 RETI 指令（从中断返回）结束，以确保操作正确。

*注：TRAP 指令不会修改 PSW 寄存器中的 CPU 优先级和所选的寄存器组，所以服务程序和中断任务的优先级相同。因此 TRAP 指令触发的强制中断服务程序可以被其它强制中断或更高优先级的中断请求中断，除非强制中断服务程序是由真实的硬件事件触发的。TRAP 指令触发的强制中断服务程序使用的寄存器组不改变。如果硬件触发同样的服务程序，可由 ITC 选择寄存器组，因此所使用的寄存器组可能不同。*

*注：请求从不允许读取的内部程序存储器空间得到数据时（如用户读访问受保护的 Flash），也可产生并发出软件强制中断。*

### 7.9.2 硬件强制中断

硬件强制中断由系统运行期间的错误或特定系统状态触发（汇编阶段不能识别出来）。XE16xyM 能区分 12 种不同的硬件强制中断功能。检测到硬件强制中断情况之后，CPU 跳转到相应的强制中断向量地址。在进入强制中断处理程序之前，要先执行完引发强制中断事件的那条指令。

硬件强制中断是不可屏蔽的，它始终具有比任何其它 CPU 任务更高的优先级。如果在同一个指令周期内检测到多个强制中断情况，具有最高优先级的强制中断得到处理（见表 7-3）。硬件强制中断的情况下，插入单元将 ITRAP 指令插入到流水线中。

ITRAP 指令执行下述动作：

- 将 PSW、CSP（分段模式下）和 IP 压入系统堆栈
- PSW 寄存器中的 CPU 优先级设置为最高级，禁止所有中断和 PEC 传送
- 为强制中断服务程序选择全局寄存器组
- 跳转到强制中断编号所指定的强制中断向量地址处

内核的硬件强制中断功能被分为两类：

**A 类强制中断:**

- 系统请求 0 (SR0)
- 堆栈上溢
- 堆栈下溢
- 软件断点

这些强制中断的优先级相同，但具有单独的向量地址。

**B 类强制中断:**

- 系统请求 1 (SR1)
- 存储器保护
- 未定义操作码
- 存储器访问出错
- 保护错误
- 非法字操作数访问

B 类强制中断共享同一个中断节点和中断向量。可位寻址的强制中断标志寄存器 (TFR) 允许服务程序确定引起异常情况的强制中断类型。

*注：强制中断服务程序必须清除相应的强制中断标志。否则，退出服务程序之后，将会请求新的强制中断。软件置位强制中断请求标志与硬件置位作用相同。*

复位功能（硬件、软件、看门狗）可视为一类强制中断。复位功能的优先级最高（强制中断优先级 III）。A 类强制中断的系统优先级仅低于最高优先级（强制中断优先级 II），B 类强制中断的优先级低于 A 类优先级（强制中断优先级 I）。因此，A 类强制中断能够中断 B 类强制中断（优先级见表 7-3）。

**A 类强制中断**

由高优先级的系统请求 SR0 或特殊 CPU 事件，如软件断点、堆栈上溢、堆栈下溢事件产生 A 类强制中断。A 类强制中断并不用来指示硬件故障。A 类强制中断事件发生之后，将调用专用服务程序来响应该事件。每个 A 类强制中断在向量表中都有各自的向量地址。执行完服务程序之后，必须保证后续指令流的正确执行，因此 A 类强制中断不能中断正在执行的 atomic/extend 序列和 I/O 访问。例如，被中断的 extend 序列不能重新启动。

除异步外部事件 SR0 外，所有 A 类强制中断都是在指令执行期间在流水线中产生。在同一 CPU 周期内，A 类强制中断不可能由流水线上两条不同的指令产生。只能在指令执行的存储阶段产生 A 类强制中断事件。总是要将引起 A 类强制中断事件的指令执行完毕。A 类强制中断的情况下，流水线直接被取消。最后执行指令的下一条指令的 IP 压入堆栈。如果正在执行 atomic/extend 序列或 I/O 读访问，则先将该序列执行完毕，之后将最后执行指令的下一条指令的 IP 压入堆栈。因此，发生 A 类强制中断时，堆栈始终保存指令流中未执行指令中的第一条指令的 IP。

*注：分支折叠单元允许将分支指令及其前一条指令并行执行。经过预处理的分支指令与前一条指令相结合，跳转指令与引起 A 类强制中断的指令同时执行。指令流中，未执行指令中的第一条指令的 IP 被压入堆栈。*

多个 A 类强制中断同时出现时，会在内部进行优先级排序，SR0 强制中断优先级最高，软件断点优先级最低。

*注：两个不同的 A 类强制中断同时发生时，两个强制中断标志都置位。最后执行指令的下一个指令的 IP 被压入堆栈。先执行具有较高优先级的强制中断服务程序。从该服务程序返回后，IP 从堆栈中弹出。由于另一个挂起的 A 类强制中断尚未处理，IP 被立即再次压入堆栈（除非第一个强制中断服务程序已经将 TFR 中与第二个强制中断对应的强制中断标志清零）。*

## **B 类强制中断**

B 类强制中断由不可恢复的硬件故障产生。硬件出错时，CPU 必须立刻启动故障服务程序。B 类强制中断能够中断 **atomic/extend** 序列和 IO 读访问。执行完 B 类强制中断服务程序之后，不可能恢复被中断的指令流。

所有 B 类强制中断具有相同的优先级（强制中断优先级 1）。多个 B 类强制中断同时有效时，寄存器 TFR 中的相应标志都被置位，并进入强制中断服务程序。由于所有 B 类强制中断具有相同的向量地址，同时发生的 B 类强制中断的优先级由强制中断服务程序中的软件决定。

所有 B 类强制中断与指令执行同步，大多数 B 类强制中断是在指令执行流水线中产生的。在同一个 CPU 周期内，A 类和 B 类强制中断不可能由流水线中的两条不同指令产生。只能在指令执行的存储阶段产生 B 类强制中断事件。SR1 和 ACER 的情况例外，因为它们是由 SCU 产生的。

总是要将引起 B 类强制中断事件的指令执行完毕。发生 B 类强制中断时，流水线直接被取消，引起强制中断的指令的下一条指令的 IP 压入堆栈。因此，堆栈始终保存指令流中未执行指令中的第一条指令的 IP。

*注：分支折叠单元允许将分支指令及其前一条指令并行执行。经过预处理的分支指令与前一条指令相结合，跳转指令与引起 B 类强制中断的指令同时执行。指令流中，未执行指令中的第一条指令的 IP 被压入堆栈。*

在 A 类强制中断服务程序执行期间，任何 B 类强制中断都不会被响应，直到 A 类强制中断服务程序使用 **RETI** 指令退出中断之后，才能处理 B 类强制中断。在这种情况下，TFR 寄存器中保存 B 类强制中断的状态信息，但引起该强制中断的指令 IP 值丢失。

*注：A 类强制中断和 B 类强制中断同时发生时，二者的强制中断标志都置位。引起强制中断的指令的下一条指令 IP 被压入堆栈，执行 A 类强制中断。若正在执行 **atomic/extend** 指令序列或正在进行 IO 读访问时，发生 A 类和 B 类强制中断，B 类强制中断会中断 **atomic/extend** 操作保护，无需等待指令序列执行完*

毕就立即开始执行 **A 类强制中断** 的服务程序。从服务程序返回之后，**IP** 从系统堆栈中弹出。由于尚未处理 **B 类强制中断**，**IP** 会立刻被再次压入堆栈。在这种情况下，不可能恢复被中断的指令流。

- **系统请求 0 强制中断（A 类）**：由 SCU 产生的控制信号，见 SCU 强制中断产生。
- **堆栈上溢强制中断（A 类）**：只要堆栈指针与堆栈上溢寄存器 **STKOV** 相等且堆栈指针将自动减小，则置位 **TFR** 寄存器中的 **STKOF** 标志，CPU 将进入堆栈上溢强制中断服务程序。
- **堆栈下溢强制中断（A 类）**：只要堆栈指针与堆栈下溢寄存器 **STKUN** 相等且堆栈指针将自动增加，则置位 **TFR** 寄存器中的 **STKUF** 标志，CPU 将进入堆栈下溢强制中断服务程序。
- **软件断点强制中断（A 类）**：当 CPU 正在执行 **SBRK** 指令时，则置位寄存器 **TFR** 中的 **SOFTBRK** 标志，CPU 随即进入软件断点调试程序。片上仿真模块可以禁止软件断点指令强制中断标志的产生。这种情况下，该指令只中断指令流，并将该事件通知给调试器，标志位不置位，不执行强制中断服务程序。
- **系统请求 1 强制中断（B 类）**：由 SCU 产生的控制信号，见 SCU 强制中断产生。
- **存储器保护强制中断（B 类）**：当检测到对许可的地址范围之外进行访问时，产生此中断。根据访问类型，分成读（**MPR**）、写（**MPW**）和执行（**MPX**）存储器保护错误。
- **未定义操作码强制中断（B 类）**：若当前经 CPU 解码的指令不包含有效的 **C166SV2** 操作码，即置位 **TFR** 寄存器中的 **UNDOPC** 标志，CPU 进入未定义操作码强制中断服务程序。引起未定义操作码强制中断的指令像 **NOP** 指令一样执行。
- **存储器访问出错（B 类）**：由 SCU 产生的控制信号，见 SCU 强制中断产生。
- **保护错误强制中断（B 类）**：在执行特殊的被保护指令时，只要指令的操作码没有在指令的第二个字中重复两次，而且跟在操作码后的字节不是操作码的补码，则置位 **TFR** 寄存器中的 **PRTFLT** 标志，CPU 进入保护错误强制中断程序。受保护的指令包括 **DISWDT**、**EINIT**、**IDLE**、**PWRDN**、**SRST**、**ENWDT** 和 **SRVWDT**。引起保护错误强制中断的指令像 **NOP** 指令一样执行。对于支持 MPU 的产品，**RETI** 也被定义为受保护指令，因为只有特权代码下才允许执行该指令，即在保护级别为 0 时执行代码。使用该标志指示系统试图在其它保护级别（与级别 0 不同的）之上执行 **RETI** 指令。请注意即使 **RETI** 指令导致一个保护错误强制中断，**RETI** 仍将被执行（该指令的执行不同于 **NOP** 指令）。
- **非法字操作数访问强制中断（B 类）**：只要试图从奇字节地址读取或写入字操作数（包括 **Flash** 命令！），则置位 **TFR** 寄存器中的 **ILLOPA** 标志，CPU 进入非法字操作数访问强制中断服务程序。

## 强制中断向量地址

**表 7-3** 列出硬件强制中断的向量地址以及 TFR 寄存器中的相关状态标志。在同一条指令内可能检测到多个强制中断，该表还列出了这些情况下强制中断服务的优先级。任何复位操作（硬件复位、软件复位指令 **SRST** 或看门狗定时器溢出复位）之后，程序从位于 **xx'0000<sub>H</sub>** 的复位向量开始执行。复位操作比其它任何系统操作的优先级都高，因此具有最高优先级（强制中断优先级 III）。

软件强制中断的向量地址可以是任意确定的向量地址。通过软件 **TRAP** 指令执行中断服务程序时，其优先级始终为当前 **CPU** 优先级，由寄存器 **PSW** 的位域 **ILVL** 指示。这意味着所有的硬件强制中断或更高优先级的中断请求都会中断软件 **TRAP** 指令。

**表 7-3 硬件强制中断总结**

异常情况	强制中断标志	强制中断向量	向量地址 <sup>1)</sup>	向量编号	强制中断优先级
应用复位	-	RESET	xx'0000 <sub>H</sub>	00 <sub>H</sub>	III
<b>A 类硬件强制中断：</b>					
• 系统请求 0	SR0	SR0TRAP	xx'0008 <sub>H</sub>	02 <sub>H</sub>	II
• 堆栈上溢	STKOF	STOTRAP	xx'0010 <sub>H</sub>	04 <sub>H</sub>	II
• 堆栈下溢	STKUF	STUTRAP	xx'0018 <sub>H</sub>	06 <sub>H</sub>	II
• 软件断点	SOFTBRK	SBRKTRAP	xx'0020 <sub>H</sub>	08 <sub>H</sub>	II
<b>B 类硬件强制中断：</b>					
• 系统请求 1	SR1	BTRAP	xx'0028 <sub>H</sub>	0A <sub>H</sub>	I
• 存储器保护	MPR/W/X	BTRAP	xx'0028 <sub>H</sub>	0A <sub>H</sub>	I
• 未定义的操作码	UNDOPC	BTRAP	xx'0028 <sub>H</sub>	0A <sub>H</sub>	I
• 存储器访问错误	ACER	BTRAP	xx'0028 <sub>H</sub>	0A <sub>H</sub>	I
• 受保护指令错误	PRTFLT	BTRAP	xx'0028 <sub>H</sub>	0A <sub>H</sub>	I
• 非法字操作数访问	ILLOPA	BTRAP	xx'0028 <sub>H</sub>	0A <sub>H</sub>	I

1) 寄存器 **VECSEG** 决定向量表所在的段。

**CPUCON1** 寄存器中的位域 **VECSC** 定义了两个相邻向量之间的间距。该表使用 **VECSC** 的缺省值，即两个相邻向量之间的间距为 4 个字节（两个字）。

### 7.9.2.1 强制中断标志寄存器 TFR

XE16xyM 提供多个强制中断向量（A 类和 B 类），在强制中断标志 TFR 中指示这些强制中断。

#### TFR

**强制中断标志寄存器**

**SFR (FFAC<sub>H</sub>/D6<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>SR0</b>	<b>STK OF</b>	<b>STK UF</b>	<b>SOFT BRK</b>	<b>SR1</b>	<b>MPR</b>	<b>MPW</b>	<b>MPX</b>	<b>UND OPC</b>	<b>0</b>	<b>0</b>	<b>AC ER</b>	<b>PRT FLT</b>	<b>ILL OPA</b>	<b>0</b>	<b>0</b>
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	r	r	rwh	rwh	rwh	r	r

符号	位序号	类型	功能描述
<b>SR0</b>	15	rwh	<b>系统请求 0 标志</b> 0 <sub>B</sub> 未检测到触发情况 1 <sub>B</sub> 已经检测到所选择的触发情况
<b>STKOF</b>	14	rwh	<b>堆栈上溢标志</b> 0 <sub>B</sub> 未检测到堆栈上溢事件 1 <sub>B</sub> 当前堆栈指针小于寄存器 STKOV 的值
<b>STKOUF</b>	13	rwh	<b>堆栈下溢标志</b> 0 <sub>B</sub> 未检测到堆栈下溢事件 1 <sub>B</sub> 当前堆栈指针大于寄存器 STKUN 的值
<b>SOFTBRK</b>	12	rwh	<b>软件断点</b> 0 <sub>B</sub> 未检测到软件断点事件 1 <sub>B</sub> 检测到软件断点事件
<b>SR1</b>	11	rwh	<b>系统请求 1 标志</b> 0 <sub>B</sub> 未检测到触发情况 1 <sub>B</sub> 已经检测到触发情况
<b>MPR</b>	10	rwh	<b>存储器保护读</b> 0 <sub>B</sub> 未检测到违反读保护的情况 1 <sub>B</sub> 已经检测到违反读保护的情况

符号	位序号	类型	功能描述
<b>MPW</b>	9	rwh	<b>存储器保护写</b> 0 <sub>B</sub> 未检测到违反写保护的情况 1 <sub>B</sub> 已经检测到违反写保护的情况
<b>MPX</b>	8	rwh	<b>存储器保护执行</b> 0 <sub>B</sub> 未检测到违反执行保护的情况 1 <sub>B</sub> 已经检测到违反执行保护的情况
<b>UNDOPC</b>	7	rwh	<b>未定义操作码</b> 0 <sub>B</sub> 未检测到未定义操作码事件 1 <sub>B</sub> 当前被解码的指令包含无效操作码
<b>ACER</b>	4	rwh	<b>存储器访问错误</b> 0 <sub>B</sub> 未检测到访问错误事件 1 <sub>B</sub> 检测到非法或错误的存储器访问
<b>PRTFLT</b>	3	rwh	<b>保护错误</b> 0 <sub>B</sub> 未检测到保护错误事件 1 <sub>B</sub> 检测到非法格式的受保护指令
<b>ILLOPA</b>	2	rwh	<b>非法字操作数访问</b> 0 <sub>B</sub> 未检测到非法字操作数访问事件 1 <sub>B</sub> 已经检测到试图对奇地址进行字操作数访问 (读或写)
<b>0</b>	[6:5], [1:0]	r	<b>保留</b> 读操作返回 0；应写入 0

注：标志 *TFR.15*、*TFR.11* 和 *TFR.4* 由 *SCU* 产生。*TFR.8*、*TFR.9* 和 *TFR.10* 由 *MPU* 产生。其它标志由 *CPU* 产生。



## 7.10 外围事件控制器

XE16xyM 外围事件控制器（PEC）提供 8 路 PEC 服务通道，在任意两个地址单元之间实现一个字节或字的传送。PEC 传送可由中断服务请求触发，具有最快的中断响应速度。在很多情况下，PEC 传送足以满足相应外设（如串行通道等）的需要。

PEC 传送并不改变当前上下文，只是“窃取”CPU 的几个时钟周期，所以不需要像标准中断那样对当前程序状态和上下文进行保存和恢复操作。

PEC 通道能执行下述操作：

- 字节或字传送
- 连续数据传送
- 完成数据传送时产生的 PEC 特定通道中断请求或用于增强处理的所有通道共用的“PEC 结束”中断
- 源和/或目的指针自动递增，支持从存储器到存储器的传送方式

注：如果 PEC 传送的优先级高于当前 CPU 优先级，才执行 PEC 传送操作。

### 7.10.1 PEC 控制寄存器

每个 PEC 通道由相应的 PEC 通道控制寄存器（PECCx）和一组源和目的指针（SRCPx、DSTPx、PECSEGx），其中 x 代表 PEC 通道编号。PECCx 寄存器控制分配给 PEC 通道的仲裁优先级和将要执行的动作。

#### PECCx (x=0-7)

PEC 通道控制 x						SFR (FEC0 <sub>H</sub> +2*x)						复位值: 0000 <sub>H</sub>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	EOP INT	PLEV		CL	INC		BWT	COUNT							
r	rw	rw		rw	rw		rw	rwh							

符号	位序号	类型	功能描述
EOPINT	14	rw	<b>PEC 结束中断选择</b> 0 <sub>B</sub> PEC 结束中断和触发 PEC 传送的中断的优先级相同 1 <sub>B</sub> 由独立的中断节点控制 PEC 结束中断，具有可编程的中断优先级（EOPIC）和中断共享控制寄存器（PECISNC）。
PLEV	[13:12]	rw	可编程 PEC 中断优先级

符号	位序号	类型	功能描述
			00 <sub>B</sub> 标准（兼容模式）：优先级 15 和 14 01 <sub>B</sub> 优先级 13 和 12 10 <sub>B</sub> 优先级 11 和 10 11 <sub>B</sub> 优先级 9 和 8
<b>CL</b>	11	rw	<b>通道链接控制</b> 0 <sub>B</sub> 各 PEC 通道独立工作 1 <sub>B</sub> PEC 通道成对链接工作
<b>INC</b>	[10:9]	rw	<b>增量控制</b> （PEC 传送之后，修改源和目的指针） 00 <sub>B</sub> 不修改指针 01 <sub>B</sub> 目的指针 DSTPx 加 1（BWT = 1）或 2（BWT = 0） 10 <sub>B</sub> 源指针 SRCPx 加 1（BWT = 1）或 2（BWT = 0） 11 <sub>B</sub> 目的指针 DSTPx 和源指针 SRCPx 都加 1（BWT = 1）或 2（BWT = 0）
<b>BWT</b>	8	rw	<b>字节/字传送选择</b> 0 <sub>B</sub> 传送一个字 1 <sub>B</sub> 传送一个字节
<b>COUNT</b>	[7:0]	rwh	<b>PEC 传送计数</b> 对 PEC 传送次数计数，并影响控制通道操作（见 <a href="#">表 7-4</a> ）
<b>0</b>	15	r	<b>保留</b> 读操作返回 0；应写入 0。

PECCx 寄存器中的**字节/字传送位**（BWT）选择在每个 PEC 服务周期内传送一个字节还是一个字，这将决定指针修改的递增步长。

PECCx 寄存器中的**PEC 传送计数位域**（COUNT）直接控制相应 PEC 通道的操作。位域 COUNT 中的内容可决定确定次数的 PEC 传送，无限次传送或根本不进行 PEC 服务。

### 中断与异常情况控制

- 如果 PEC 传送计数器 COUNT 值设置为 **00<sub>H</sub>**，处理正常中断请求而不是 PEC 数据传送，相应的 PEC 通道保持空闲状态。
- 通过设置位域 COUNT 至 **FF<sub>H</sub>**，选择**连续数据传送**。在此情况下，传送操作不会使 COUNT 递减，相应的 PEC 通道能够服务无限次的 PEC 请求，直到该位域被程序修改为止。
- 如果相应 PEC 通道请求将位域 COUNT 设置为特定值，每次 PEC 传送之后，位域 COUNT 递减，请求标志被清除，指示已经服务该 PEC 传送请求。当 COUNT 递减至 **00<sub>H</sub>**时，激活中断服务程序，其优先级可以是相同的优先级（EOPINT = 0）或触发具有不同优先级的（EOPINT = 1）的“PEC 结束”中断。数据传送之后，当 COUNT 从 **01<sub>H</sub>**递减至 **00<sub>H</sub>**时，如果 EOPINT 设置为 1，则请求标志被清除。如果 EOPINT = 0，则请求标志将不会被清除，在相同优先级上产生另一个中断请求。相应的 PEC 通道保持空闲，因为 COUNT 包含 **00<sub>H</sub>**，因此相应的服务请求程序而不是 PEC 传送被激活。（见**章节 7.10.3**）。

EOPIC 寄存器为 PEC 结束中断的控制寄存器。

## EOPIC

### PEC 结束中断控制

**ESFR (F19E<sub>H</sub>/CF<sub>H</sub>)**

复位值: **0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	0	0	0	0	0	0	GPX	EOP IR	EOP IE	ILVL				GLVL	
r	r	r	r	r	r	r	rw	rwh	rw	rw				rw	

符号	位序号	类型	功能描述
<b>GPX</b>	8	rw	<b>组优先级扩展</b> 定义组优先级高位的值。
<b>EOPIR</b>	7	rwh	<b>中断请求标志</b> 0 <sub>B</sub> 没有挂起的中断请求 1 <sub>B</sub> 中断源已经发出一个中断请求
<b>EOPIE</b>	6	rw	<b>中断请求使能控制位</b> 0 <sub>B</sub> 禁止中断请求 1 <sub>B</sub> 使能中断请求
<b>ILVL</b>	[5:2]	rw	<b>中断优先级</b> F <sub>H</sub> 最高优先级 ... <sub>H</sub> ...

符号	位序号	类型	功能描述
			0 <sub>H</sub> 最低优先级
<b>GLVL</b>	[1:0]	rw	组优先级 3 <sub>H</sub> 最高优先级 ... <sub>H</sub> ... 0 <sub>H</sub> 最低优先级
<b>0</b>	[15:9]	r	保留 读操作返回 0；应写入 0。

注：组优先级扩展位 **GPX** 和组优先级位域 **GLVL** 级联之后构成 3 位扩展组优先级 **XGLVL**，其中 7<sub>H</sub> 为最高优先级，0<sub>H</sub> 为最低优先级。

寄存器 **PECISNC** 包含“PEC 结束”中断节点的标志。增强的“PEC 结束”中断特性使用该节点，相应 **PECCx** 中的控制位 **EOPINT** 设置为 1。

## PECISNC

### PEC 中断子节点控制

**SFR (FFD8<sub>H</sub>/EC<sub>H</sub>)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>C7IR</b>	<b>C7IE</b>	<b>C6IR</b>	<b>C6IE</b>	<b>C5IR</b>	<b>C5IE</b>	<b>C4IR</b>	<b>C4IE</b>	<b>C3IR</b>	<b>C3IE</b>	<b>C2IR</b>	<b>C2IE</b>	<b>C1IR</b>	<b>C1IE</b>	<b>C0IR</b>	<b>C0IE</b>
rwh	rw	rwh	rw	rwh	rw	rwh	rw	rwh	rw	rwh	rw	rwh	rw	rwh	rw

符号	位序号	类型	功能描述
<b>CxIR</b> (x=0-7)	2*x+1	rwh	<b>PEC 通道 x<sup>1)</sup>中断子节点请求标志</b> 0 <sub>B</sub> PEC 通道 x 没有挂起的 PEC 结束中断请求 1 <sub>B</sub> PEC 通道 x 已经发出一个 PEC 结束中断请求
<b>CxIE</b> (x=0-7)	2*x	rw	<b>PEC 通道 x<sup>2)</sup>中断子节点使能控制位</b> (单独使能/禁止特定的中断请求源) 0 <sub>B</sub> 禁止通道 x PEC 结束中断请求 1 <sub>B</sub> 使能通道 x PEC 结束中断请求

1) x = 7...0，由 PEC 通道编号决定。

2) 建议用户在置位相应的使能标志(**CxIE**)之前清除中断请求标志 (**CxIR**)。否则，置位使能位之后，仍然挂起的先前的请求会立即触发一个中断请求。

## 中断与异常情况控制

注：当进入中断服务程序时（CPU 已经接受该中断）时，“PEC 结束”子节点中断请求标志不会被硬件清除，这点与中断节点的中断请求标志不同（请求标志 xxIC.xxIR）。中断服务程序必须检查请求标志，并在执行 RETI 指令之前清除这些标志。

下图给出“PEC 结束”中断子节点的法用：

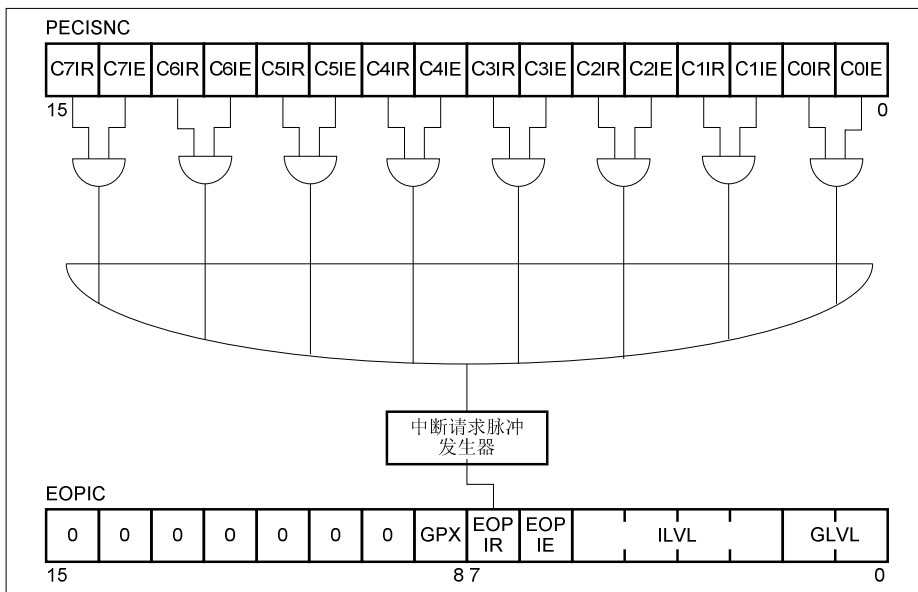


图 7-4 PEC 结束中断子节点

位域 COUNT 的值和对应的 PEC 通道操作归纳见下表。

表 7-4 PEC 通道操作

COUNT 先前值	COUNT 修改值	PEC 通道操作及说明
FFH	FFH	转移一个字节/字 连续传送模式，即不修改 COUNT 的值

COUNT 先前值	COUNT 修改值	PEC 通道操作及说明
FEH ... 02H	FDH ... 01H	转移一个字节/字，COUNT 减 1
01H	00H	<p><b>转移一个字节/字</b></p> <p>根据位 EOPINT 的值，进行两种不同操作：</p> <p><b>EOPINT = 0</b>（兼容模式）</p> <p>相应中断的服务请求标志（xxIR）保持置位状态（对于所有其它 COUNT 值，该标志被清除）。因此，COUNT 值等于 00H 时，在下一个仲裁周期内触发另一个中断请求（见下面的描述）</p> <p><b>EOPINT = 1</b></p> <p>相应中断的服务请求标志（xxIR）被清除。而且，EOP 子节点的中断请求标志（PECISNC.CxIR）被置位。另外，如果子节点请求被使能（PECISNC.CxIE = '1'），PEC 结束中断节点的中断请求标志（EOPIC.EOPIR）被自动置位。</p> <p>（参见 <a href="#">章节 7.10.3</a>）</p>
00H	00H	<p><b>无 PEC 操作！</b></p> <p>请求正常中断，而不是 PEC 数据传送（参见 <a href="#">章节 7.10.3</a>）。</p>

PECCx 寄存器中的**增量控制位域（INC）**定义 PEC 传送之后，是否 PEC 指针中的一个或两个都必须递增。如果不修改指针（INC = '00'），相应的通道将始终从同一个源地址向同一个目的地址传送数据。

### 数据链接的通道连接模式

如果被使能，通道链接将两路通道连接在一起，共同服务某外设的数据传送请求。整个数据传送（例如报文）被分成控制和链接块进行传送。连接在一起的两路 PEC 通道，交替处理链接块传送。由一路 PEC 通道控制的一次数据块传送结束时，自动启动另一路（连接的）PEC 通道，并继续下一个数据块的传送。由一对 PEC 通道（通道 0&1, 2&3 等等）支持通道连接和数据（块）链接。由通道对中的一个 PEC 通道控制每个数据块。当两个通道中的一路通道有效时，CPU 能够更新另一个通道的指针和计数器值，为下一个通道连接之后的连续数据传送做好准备。

如果通道对内有效 PEC 通道的 PECCx 寄存器的通道连接控制位“CL”被置为 1，则通道连接被使能。连接通道的数据传送始终从通道对中的偶数通道启动。如果通

道连接模式下（通道对中至少一个 CL 位置位），通道的数据块传送完成之后，PEC 服务请求处理被自动切换至通道对中的另一路 PEC 通道上。

当第一个（有效）通道的 CL 位（其 PEC 控制寄存器内的）置位且最后一次传送之后，其传送计数器从 1 变为 0 时，通道被连接，因而执行从一路通道到另一路通道的切换。如果发现第一个（被终止的）PEC 控制寄存器的通道连接标志 CL 为 0 或者连接通道的计数位域为 0，则整个数据传送操作结束。

*注：CL 标志完全由软件控制，要完成并终止整个数据传送操作时，应该软件清除该标志。因为连接通道的传送计数位域为 0 值时，通道也可被终止，如果该通道的最后一次通道连接中断之后，其通道计数位域未被更新，则自动终止整个数据传送操作。*

当连接通道的数据块被完全传送之后，PEC 服务切换至通道对中的另一个通道，产生一个通道（旧的通道）连接中断，通知 CPU 该通道现在无效，可为下次数据块传送操作配置该通道。请求通道连接中断时，由相应的 PEC 中断子节点控制寄存器

（PECISNC）进行指示和使能，该寄存器还可用于通道的 PEC 结束中断（见前面的章节）。因此，可用一个 EOP 中断控制寄存器来控制所有的通道连接中断，所以其优先级等于 EOP 中断的优先级。如果 PEC 中断子节点控制寄存器 PECISNC 和中断控制寄存器 EOPIC 中的使能控制位置位，出现一个或多个挂起通道连接请求或 PEC 结束中断请求的情况下，该服务请求节点请求 CPU 中断服务。

*注：如果有效（被终止）通道的 CL 位置位，则通道连接的产生/EOP 中断被自动使能。如果其未被置位，或者产生一个标准中断，或者根据通道的 PEC 控制寄存器中的 EOPINT 位，产生一个 EOP 中断。在此情况下，不切换通道，因为定义最后一次数据块传送的 CL 标志丢失。*

*注：如果通道连接模式有效（通道对的 CL 位至少一个置位），连接至旧的通道（通过优先级）的中断请求仅触发一个标准中断而不是 PEC 传送中断。*

*注：连接通道的数据传送始终从通道对中的偶数通道开始。*

## **7.10.2 PEC 源和目的指针**

PEC 通道的源指针和目的指针指定了数据转移的源和目的地址。两个指针都是 24 位指针，24 位源地址保存在寄存器 SRCPx（地址低 16 位）和寄存器 PECSEGx 的高字节（最高 8 位地址位）。

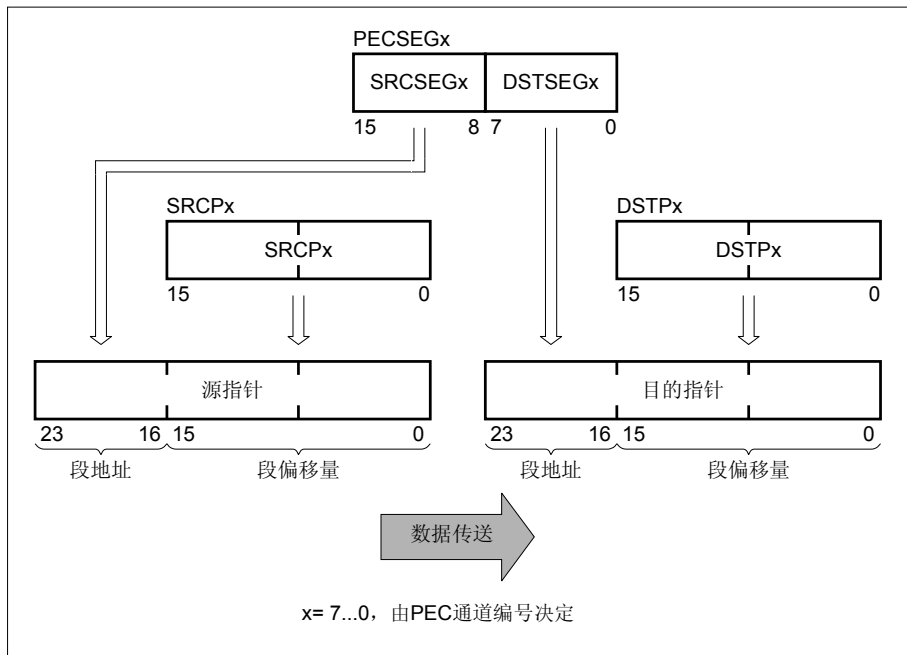


图 7-5 PEC 指针地址处理

24 位目的地址保存在寄存器 DSTPx（地址的低 16 位）和寄存器 PECSEGx 的低字节（地址的高 8 位）。PEC 地址指针的低 16 位（段内偏移）能够被 PEC 传送机制修改（递增）。地址的高 8 位代表段编号，不能由硬件修改。因此，PEC 指针可在一个地址段的地址空间内递增，不能跨越地址段边界。如果字节传送的情况下（BWT = 1）偏移地址指针达到 FFFF<sub>H</sub>，字传送情况下（BWT = 0）偏移地址指针达到 FFFE<sub>H</sub>，下一次不再递增地址。地址寄存器保持最大值（FFFF<sub>H</sub> 或 FFFE<sub>H</sub>）且不会发生溢出。该行为可防止相邻的存储器被无意修改。地址指针饱和的情况下，系统不会产生一个明显的错误事件；因此，需要用户来避免此情况的发生。

注：PEC 数据传送不使用数据页指针 DPP3...DPP0。

注：如果特定 PEC 通道选择字数据传送（即 BWT = 0），相应的源和目的指针必须包含有效的字地址（指向偶字节边界）。否则，使用该通道时，会触发一个非法字操作数访问强制中断。



**SRCPx (x= 0-7)**

### PEC 源指针 x

**XSFR (EC40<sub>H</sub>+4\*x)**

复位值: 0000<sub>H</sub>

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

SRCPx
rwh

符号	位序号	类型	功能描述
<b>SRCPx</b>	[15:0]	rwh	通道 x 的源地址指针偏移量 源地址位 15...0

x = 7...0, 对应 PEC 通道编号。

### DSTPx (x = 0-7)

### PEC 目的指针 x

**XSFR (EC42<sub>H</sub>+4\*x)**

复位值: 0000<sub>H</sub>

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

DSTPx
rwh

符号	位序号	类型	功能描述
<b>DSTPx</b>	[15:0]	rwh	通道 x 的目的地址指针偏移量 目的地址位 15...0

x = 7...0, 对应 PEC 通道编号。

对于 XSFR 地址，参见表 7-11。

### PECSEGx (x = 0-7)

PEC 段指针 x

**XSFR (EC80<sub>H</sub>+2\*x)**

复位值: 0000<sub>H</sub>

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

<b>SRCSEGx</b>	<b>DSTSEGx</b>
rW	rW

符号	位序号	类型	功能描述
<b>SRCSEGx</b>	[15:8]	rw	通道 x 源地址的段指针

符号	位序号	类型	功能描述
			源地址位 23...16
<b>DSTSEGx</b>	[7:0]	rw	<b>通道 x 目的地址的段指针</b> 目的地址位 23...16

x = 7...0, 对应 PEC 通道编号。

### 7.10.3 PEC 中断处理总结

如上所述, 根据位域 COUNT 的状态, PEC 处理器可触发两种不同的中断。

- PEC 通道被使能<sup>1)</sup>且位域 COUNT 的值大于 01<sub>H</sub>。
  - 控制位 EOPINT = 0 或 1
  - 动作:**
    - 处理 PEC 请求
    - 无其它中断动作
- PEC 通道被使能, 由服务请求触发, 使得位域 COUNT 从 01<sub>H</sub>递减至 00<sub>H</sub>,
  - 控制位 EOPINT = 0 (与 C166 兼容)
  - 动作:**
    - 处理 PEC 请求
    - 所请求中断节点的中断请求标志 (xxIR) 不被清除 (仲裁获胜者), 参加下次仲裁周期, 并触发一个正常中断, 其优先级等于所服务的 PEC 请求的优先级。
    - 控制位 EOPINT = 1 (PEC 结束中断的增强特性)
    - 动作:**
      - 处理 PEC 请求
      - 请求中断节点 (仲裁获胜者) 的中断请求标志 (xxIR) 被清除, 不会触发其它动作。
      - PEC 结束中断子节点的中断请求标志将会被置位 (PECISNC.CxIR = 1)
      - 如果之前, 软件置位 PEC 结束中断子节点的相应中断使能标志 (PECISNC.CxIE = 1), 则请求 PEC 结束中断 (EOPINT.EOPIR=1)。

如果之前, 软件使能该中断源 (EOPIC.EOPIR = 1), PEC 结束中断参与下次仲裁周期, 其优先级由 EOPIC.ILVL 和 EOPIC.GLVL 进行选择。具有

1) 当 PEC 通道的 COUNT 值大于 00<sub>H</sub>时, 该 PEC 通道被自动使能。

该行为的 PEC 结束中断可被触发，其优先级可低于已被处理的相应 PEC 请求。

- 如果位域 COUNT 被清除（硬件或软件），PEC 通道被禁止。

- 控制位 EOPINT = 0 或 1

**动作：**

- 请求一个正常中断服务程序（优先级等于所请求 PEC 通道优先级）

#### 7.10.4 PEC 通道分配

执行数据传送的 PEC 通道取决于中断优先级、组优先级和 PEC 优先级。下表给出通道分配情况。

**表 7-5 PEC 通道分配**

ICx ILVL	ICx GLVL	PECCy. PLEV	分配的 PEC 通道	ICx ILVL	ICx GLVL	PECCy. PLEV	分配的 PEC 通道
1111 <sub>B</sub>	11 <sub>B</sub>	00 <sub>B</sub>	7	1011 <sub>B</sub>	11 <sub>B</sub>	10 <sub>B</sub>	7
1111 <sub>B</sub>	10 <sub>B</sub>		6	1011 <sub>B</sub>	10 <sub>B</sub>		6
1111 <sub>B</sub>	01 <sub>B</sub>		5	1011 <sub>B</sub>	01 <sub>B</sub>		5
1111 <sub>B</sub>	00 <sub>B</sub>		4	1011 <sub>B</sub>	00 <sub>B</sub>		4
1110 <sub>B</sub>	11 <sub>B</sub>		3	1010 <sub>B</sub>	11 <sub>B</sub>		3
1110 <sub>B</sub>	10 <sub>B</sub>		2	1010 <sub>B</sub>	10 <sub>B</sub>		2
1110 <sub>B</sub>	01 <sub>B</sub>		1	1010 <sub>B</sub>	01 <sub>B</sub>		1
1110 <sub>B</sub>	00 <sub>B</sub>		0	1010 <sub>B</sub>	00 <sub>B</sub>		0

**表 7-5 PEC 通道分配**

ICx ILVL	ICx GLVL	PECCy. PLEV	分配的 PEC 通道	ICx ILVL	ICx GLVL	PECCy. PLEV	分配的 PEC 通道
1101 <sub>B</sub>	11 <sub>B</sub>	01 <sub>B</sub>	7	1001 <sub>B</sub>	11 <sub>B</sub>	11 <sub>B</sub>	7
1101 <sub>B</sub>	10 <sub>B</sub>		6	1001 <sub>B</sub>	10 <sub>B</sub>		6
1101 <sub>B</sub>	01 <sub>B</sub>		5	1001 <sub>B</sub>	01 <sub>B</sub>		5
1101 <sub>B</sub>	00 <sub>B</sub>		4	1001 <sub>B</sub>	00 <sub>B</sub>		4

**中断与异常情况控制**

ICx ILVL	ICx GLVL	PECCy. PLEV	分配的 PEC 通道	ICx ILVL	ICx GLVL	PECCy. PLEV	分配的 PEC 通道
1100 <sub>B</sub>	11 <sub>B</sub>		<b>3</b>	1000 <sub>B</sub>	11 <sub>B</sub>		<b>3</b>
1100 <sub>B</sub>	10 <sub>B</sub>		<b>2</b>	1000 <sub>B</sub>	10 <sub>B</sub>		<b>2</b>
1100 <sub>B</sub>	01 <sub>B</sub>		<b>1</b>	1000 <sub>B</sub>	01 <sub>B</sub>		<b>1</b>
1100 <sub>B</sub>	00 <sub>B</sub>		<b>0</b>	1000 <sub>B</sub>	00 <sub>B</sub>		<b>0</b>

所有未分配给 PEC 通道的中断请求直接送至中断处理器。

## 7.11 外部中断

尽管 XE16xyM 没有专用的中断输入引脚，但它提供数条 IO 线（可被选作中断输入），可通过多种可能的方式响应外部异步事件。

### 7.11.1 外部请求单元

请参考**外部请求单元（ERU）**一章。ERU 提供连线功能，并允许为中断输入信号定义高级的触发条件。所得的 ERU 中断请求可被送至中断控制器寄存器 ERU\_0IC... ERU\_3IC。

### 7.11.2 使用外设引脚

一些外设引脚中断功能可和引脚的主要功能合并，或如果不需用引脚的主要功能时，用作引脚的其它功能。

**表 7-6 可用作外部中断输入的引脚**

端口引脚	原功能	控制寄存器
P4.7-0/CC31-24IO	CAPCOM 寄存器 31-24 捕获输入	CC31-CC24 <sup>1)</sup>
P2.10-3/CC23-16IO	CAPCOM 寄存器 23-16 捕获输入 <sup>2)</sup>	CC23-CC16 <sup>1)</sup>
P4.2/T2IN	辅助定时器 T2 输入引脚	T2CON
P4.6/T4IN	辅助定时器 T4 输入引脚	T4CON
P2.10/CAPIN	GPT2 捕获输入引脚 <sup>2)</sup>	T5CON

1) 必须由**中断节点共享**使能。

2) 引脚 P2.10 具有两种可能的输入功能。

对于上面的每一个引脚，可以选择由外部输入的正跳变、负跳变或任意跳变来触发中断或 PEC 服务请求。边沿选择在端口引脚相应的外设器件控制寄存器中定义。外设必须设定工作在特定的模式，以允许由外部信号产生中断。中断请求的优先级由各外设中断源的中断控制寄存器确定，通过外设中断源的中断向量来服务相应的外部中断请求。

*注：为了把上述引脚用作外部中断输入，必须通过相应的端口方向控制寄存器将上述引脚切换到输入模式。*

当端口引脚 CCxIO 用作外部中断输入引脚时，对应的捕获/比较寄存器 CCx 中的位域 CCMODx 必须选择捕获模式。若 CCMODx 设置为 001<sub>B</sub>，引脚 CCxIO 上出现外部正跳变时，寄存器 CCxIC 中的中断请求标志 CCxIR 被置位；若 CCMODx 设置为 010<sub>B</sub>，引脚 CCxIO 上的外部负跳变会置位中断请求标志；若 CCMODx 设置为 011<sub>B</sub>，引脚 CCxIO 上的外部任意跳变会置位中断请求标志。在以上三种情况下，无论所分配的 CAPCOM 的定时器是否运行，定时器的内容都将被锁存到捕获寄存器 CCx 中。若中断使能位 CCxIE 置位，将产生 PEC 请求或向量 CCxINT 中断请求。

当 GPT1 模块中相关的辅助定时器 T2 或 T4 配置成捕获模式时，引脚 T2IN 或 T4IN 也可以用作外部中断输入引脚。将控制寄存器 T2CON 或 T4CON 中的模式控制位域 T2M 或 T4M 设置为 101<sub>B</sub>，即可选择捕获模式。外部输入信号的触发有效沿由位域 T2I 或 T4I 决定。若位域 T2I 或者 T4I 设置为 X01<sub>B</sub>，T2IN 或 T4IN 引脚上出现正跳变时，寄存器 T2IC 或 T4IC 中的中断请求标志 T2IR 或 T4IR 置位；若位域 T2I 或者 T4I 设置为 X10<sub>B</sub>，外部引脚负跳变时，相应的中断请求标志置位；若位域 T2I 或者 T4I 设置为 X11<sub>B</sub>，外部引脚任意跳变时，将置位请求标志。在以上三种情况下，引脚 T2IN 或 T4IN 跳变时，内核定时器 T3 的内容被捕获到辅助定时器寄存器 T2 或 T4 中。若中断使能位 T2IE 或 T4IE 置位，将产生 PEC 请求或者向量 T2INT 或向量 T4INT 中断请求。

CAPIN 引脚与定时器输入引脚略有不同，因为 CAPIN 引脚可以用作外部中断输入引脚，而不影响外设功能。当寄存器 T5CON 中的捕获模式使能位 T5SC 被清零时，CAPIN 引脚上的信号跳变只对 CRIC 寄存器中的中断请求标志 CRIR 置位，并不激活寄存器 CAPREL 的捕获功能。

因此，当引脚 CAPIN 用作外部中断输入时，寄存器 CAPREL 仍可用作 GPT2 定时器 T5 的重载寄存器。T5CON 寄存器中的位域 CI 用于选择外部中断输入信号的有效跳变。CI = 01<sub>B</sub> 时，中断请求标志在外部正跳变时置位；CI = 10<sub>B</sub> 时，中断请求标志在外部负跳变时置位；CI = 11<sub>B</sub>，中断请求标志在外部任意跳变时均置位。若中断使能位 CRIE 置位，将产生向量 CRINT 中断请求或 PEC 请求。

## 7.12 OCDS 请求

OCDS 模块发出高优先级的断点请求或标准服务请求。断点请求直接发送至 CPU（与硬件强制中断请求相似），并进行优先级排序。因此，断点请求不必进行标准中断仲裁过程，具有最高优先级。

标准 OCDS 服务请求与经过仲裁的中断/PEC 请求一并被送至 CPU 操作控制单元。优先级较高的请求发送给 CPU，获得 CPU 服务。如果标准 OCDS 请求和中断/PEC 请求的优先级相同，中断/PEC 请求将胜出。

这种方法保证了准确的断点控制，同时对系统操作的影响尽可能小。

CPU 操作控制单元也会将请求应答和拒绝信息从内核送回给相应的请求源。

## 7.13 服务请求的响应延迟

各种 XE16xyM 服务请求（中断或 PEC 服务请求）的产生和指令流的执行不同步。因此，这些请求经过仲裁后，被插入到当前的指令流中。这样可以使中断服务请求的处理与当前执行的指令流无关，但也导致了一定的延迟时间。

请求响应延迟时间是指从中断控制器（ITC）的请求信号被激活直到相关指令到达流水线执行阶段的时间。表 7-7 列出了这一过程的连续步骤。

**表 7-7 每个步骤导致的服务请求延迟**

步骤描述	中断响应	PEC 响应
通过 3 级请求仲裁后，请求被 CPU 接受（见章节 7.2）	3 个周期	3 个周期
在流水线指令流中插入一条内部指令	4 个周期	4 个周期
从中断向量表中取出的第一条指令进入流水线执行阶段	4 个周期 / 0 <sup>1)</sup>	——
最小的请求响应延迟	11/7 个周期	7 个周期

1) 可以使用中断跳转表缓存来节省这段时间（见章节 7.5）。

### 额外的响应延时

由于服务请求插入到当前指令流中，指令流的属性会影响中断请求的响应延迟。

**表 7-8          系统逻辑引起的额外延时**

延迟原因	中断响应	PEC 响应
中断控制器忙 其正在执行一个仲裁周期	最大 7 个周期	最大 7 个周期
流水线停滞 因为流水线中的、在插入的指令之前的指令 需要从/至外设或存储器读/写指令。	$2 \times T_{ACCmax}^{1)}$	$2 \times T_{ACCmax}$
流水线被取消 因为流水线中该插入指令之前的指令更新了 内核 SFR	4 个周期	4 个周期
写堆栈时的存储器访问（如果堆栈不是采用 DPRAM 和 DSRAM）	$2/3 \times T_{ACC}^{2)}$	---
读中断向量表时的存储器访问 （中断跳转表缓存除外）	$2 \times T_{ACC}$	---

1) XE16xyM 系统中可能的最大访问时间。

2) 取决于分段模式是否开启

实际的中断请求响应时间可能还会进一步被延迟，这取决于应用中所采用的编程技术。可以采取以下方式减小响应时间：

- 只通过 **JUMP** 指令从中断向量表跳转到真正的中断服务程序。  
可以将对时间要求严格的指令直接放在中断向量表中，之后紧跟一个分支指令，跳转到中断服务程序的剩余部分。由 **CPUCON1** 寄存器的 **VECSC** 位域选择两个相邻向量的间距。
- 在执行想要运行的指令前，进行上下文切换（见**章节 7.6**）。  
可以将对时间要求严格的指令编程为“非破坏性”指令，在进行上下文切换之前（用于中断服务程序的剩余部分），执行这些指令。

## 7.14 中断节点

### 7.14.1 物理中断节点

XE16xyM 中集成的模块需要的中断节点超过 C166SV2 中断控制器所提供的 96 个。因此，一些可用的物理节点在一些所选择的模块之间共享。

下表总结 96 个物理中断节点和以及相应的：

- 强制中断编号
- 向量地址
- 控制寄存器名以及地址
- 节点共享信息

**表 7-9 XE16xyM 中断节点**

中断源或 PEC 服务请求	强制中断编号	向量地址 <sup>1)</sup>	控制寄存器	寄存器地址
由 ISSR.ISS0 选择	10 <sub>H</sub>	xx'0040 <sub>H</sub>	CC2_CC16IC	F1C0 <sub>H</sub>
由 ISSR.ISS1 选择	11 <sub>H</sub>	xx'0044 <sub>H</sub>	CC2_CC17IC	F1C2 <sub>H</sub>
由 ISSR.ISS2 选择	12 <sub>H</sub>	xx'0048 <sub>H</sub>	CC2_CC18IC	F1C4 <sub>H</sub>
由 ISSR.ISS3 选择	13 <sub>H</sub>	xx'004C <sub>H</sub>	CC2_CC19IC	F1C6 <sub>H</sub>
由 ISSR.ISS4 选择	14 <sub>H</sub>	xx'0050 <sub>H</sub>	CC2_CC20IC	F1C8 <sub>H</sub>
由 ISSR.ISS5 选择	15 <sub>H</sub>	xx'0054 <sub>H</sub>	CC2_CC21IC	F1CA <sub>H</sub>
由 ISSR.ISS6 选择	16 <sub>H</sub>	xx'0058 <sub>H</sub>	CC2_CC22IC	F1CC <sub>H</sub>
由 ISSR.ISS7 选择	17 <sub>H</sub>	xx'005C <sub>H</sub>	CC2_CC23IC	F1CE <sub>H</sub>
由 ISSR.ISS8 选择	18 <sub>H</sub>	xx'0060 <sub>H</sub>	CC2_CC24IC	F1D0 <sub>H</sub>
由 ISSR.ISS9 选择	19 <sub>H</sub>	xx'0064 <sub>H</sub>	CC2_CC25IC	F1D2 <sub>H</sub>
由 ISSR.ISS10 选择	1A <sub>H</sub>	xx'0068 <sub>H</sub>	CC2_CC26IC	F1D4 <sub>H</sub>
由 ISSR.ISS11 选择	1B <sub>H</sub>	xx'006C <sub>H</sub>	CC2_CC27IC	F1D6 <sub>H</sub>
由 ISSR.ISS12 选择	1C <sub>H</sub>	xx'0070 <sub>H</sub>	CC2_CC28IC	F1D8 <sub>H</sub>
由 ISSR.ISS13 选择	1D <sub>H</sub>	xx'0074 <sub>H</sub>	CC2_CC29IC	F1DA <sub>H</sub>



**中断与异常情况控制**

中断源或 PEC 服务请求	强制中断编号	向量地址 <sup>1)</sup>	控制寄存器	寄存器地址
由 ISSR.ISS14 选择	1E <sub>H</sub>	xx'0078 <sub>H</sub>	CC2_CC30IC	F1DC <sub>H</sub>
由 ISSR.ISS15 选择	1F <sub>H</sub>	xx'007C <sub>H</sub>	CC2_CC31IC	F1DE <sub>H</sub>
GPT1 定时器 T2	20 <sub>H</sub>	xx'0080 <sub>H</sub>	GPT12E_T2IC	FF60 <sub>H</sub>
GPT1 定时器 T3	21 <sub>H</sub>	xx'0084 <sub>H</sub>	GPT12E_T3IC	FF62 <sub>H</sub>
GPT1 定时器 T4	22 <sub>H</sub>	xx'0088 <sub>H</sub>	GPT12E_T4IC	FF64 <sub>H</sub>
GPT2 定时器 T5	23 <sub>H</sub>	xx'008C <sub>H</sub>	GPT12E_T5IC	FF66 <sub>H</sub>
GPT2 定时器 T6	24 <sub>H</sub>	xx'0090 <sub>H</sub>	GPT12E_T6IC	FF68 <sub>H</sub>
GPT2 CAPSEL	25 <sub>H</sub>	xx'0094 <sub>H</sub>	GPT12E_CRIC	FF6A <sub>H</sub>
CAPCOM2 定时器 T7	26 <sub>H</sub>	xx'0098 <sub>H</sub>	CC2_T7IC	FF6C <sub>H</sub>
CAPCOM2 定时器 T8	27 <sub>H</sub>	xx'009C <sub>H</sub>	CC2_T8IC	FF6E <sub>H</sub>
A/D 转换器请求 0	28 <sub>H</sub>	xx'00A0 <sub>H</sub>	ADC_0IC	FF70 <sub>H</sub>
A/D 转换器请求 1	29 <sub>H</sub>	xx'00A4 <sub>H</sub>	ADC_1IC	FF72 <sub>H</sub>
A/D 转换器请求 2	2A <sub>H</sub>	xx'00A8 <sub>H</sub>	ADC_2IC	FF74 <sub>H</sub>
A/D 转换器请求 3	2B <sub>H</sub>	xx'00AC <sub>H</sub>	ADC_3IC	FF76 <sub>H</sub>
A/D 转换器请求 4	2C <sub>H</sub>	xx'00B0 <sub>H</sub>	ADC_4IC	FF78 <sub>H</sub>
A/D 转换器请求 5	2D <sub>H</sub>	xx'00B4 <sub>H</sub>	ADC_5IC	FF7A <sub>H</sub>
A/D 转换器请求 6	2E <sub>H</sub>	xx'00B8 <sub>H</sub>	ADC_6IC	FF7C <sub>H</sub>
A/D 转换器请求 7	2F <sub>H</sub>	xx'00BC <sub>H</sub>	ADC_7IC	FF7E <sub>H</sub>
CCU60 请求 0	30 <sub>H</sub>	xx'00C0 <sub>H</sub>	CCU60_0IC	F160 <sub>H</sub>
CCU60 请求 1	31 <sub>H</sub>	xx'00C4 <sub>H</sub>	CCU60_1IC	F162 <sub>H</sub>
CCU60 请求 2	32 <sub>H</sub>	xx'00C8 <sub>H</sub>	CCU60_2IC	F164 <sub>H</sub>
CCU60 请求 3	33 <sub>H</sub>	xx'00CC <sub>H</sub>	CCU60_3IC	F166 <sub>H</sub>
CCU61 请求 0	34 <sub>H</sub>	xx'00D0 <sub>H</sub>	CCU61_0IC	F168 <sub>H</sub>
CCU61 请求 1	35 <sub>H</sub>	xx'00D4 <sub>H</sub>	CCU61_1IC	F16A <sub>H</sub>

**中断与异常情况控制**

中断源或 PEC 服务请求	强制中断编号	向量地址 <sup>1)</sup>	控制寄存器	寄存器地址
CCU61 请求 2	36 <sub>H</sub>	xx'00D8 <sub>H</sub>	CCU61_2IC	F16C <sub>H</sub>
CCU61 请求 3	37 <sub>H</sub>	xx'00DC <sub>H</sub>	CCU61_3IC	F16E <sub>H</sub>
CCU62 请求 0	38 <sub>H</sub>	xx'00E0 <sub>H</sub>	CCU62_0IC	F170 <sub>H</sub>
CCU62 请求 1	39 <sub>H</sub>	xx'00E4 <sub>H</sub>	CCU62_1IC	F172 <sub>H</sub>
CCU62 请求 2	3A <sub>H</sub>	xx'00E8 <sub>H</sub>	CCU62_2IC	F174 <sub>H</sub>
CCU62 请求 3	3B <sub>H</sub>	xx'00EC <sub>H</sub>	CCU62_3IC	F176 <sub>H</sub>
CCU63 请求 0	3C <sub>H</sub>	xx'00F0 <sub>H</sub>	CCU63_0IC	F178 <sub>H</sub>
CCU63 请求 1	3D <sub>H</sub>	xx'00F4 <sub>H</sub>	CCU63_1IC	F17A <sub>H</sub>
CCU63 请求 2	3E <sub>H</sub>	xx'00F8 <sub>H</sub>	CCU63_2IC	F17C <sub>H</sub>
CCU63 请求 3	3F <sub>H</sub>	xx'00FC <sub>H</sub>	CCU63_3IC	F17E <sub>H</sub>
CAN0	40 <sub>H</sub>	xx'0100 <sub>H</sub>	CAN_0IC	F140 <sub>H</sub>
CAN1	41 <sub>H</sub>	xx'0104 <sub>H</sub>	CAN_1IC	F142 <sub>H</sub>
CAN2	42 <sub>H</sub>	xx'0108 <sub>H</sub>	CAN_2IC	F144 <sub>H</sub>
CAN3	43 <sub>H</sub>	xx'010C <sub>H</sub>	CAN_3IC	F146 <sub>H</sub>
CAN4	44 <sub>H</sub>	xx'0110 <sub>H</sub>	CAN_4IC	F148 <sub>H</sub>
CAN5	45 <sub>H</sub>	xx'0114 <sub>H</sub>	CAN_5IC	F14A <sub>H</sub>
CAN6	46 <sub>H</sub>	xx'0118 <sub>H</sub>	CAN_6IC	F14C <sub>H</sub>
CAN7	47 <sub>H</sub>	xx'011C <sub>H</sub>	CAN_7IC	F14E <sub>H</sub>
CAN8	48 <sub>H</sub>	xx'0120 <sub>H</sub>	CAN_8IC	F150 <sub>H</sub>
CAN9	49 <sub>H</sub>	xx'0124 <sub>H</sub>	CAN_9IC	F152 <sub>H</sub>
CAN10	4A <sub>H</sub>	xx'0128 <sub>H</sub>	CAN_10IC	F154 <sub>H</sub>
CAN11	4B <sub>H</sub>	xx'012C <sub>H</sub>	CAN_11IC	F156 <sub>H</sub>
CAN12	4C <sub>H</sub>	xx'0130 <sub>H</sub>	CAN_12IC	F158 <sub>H</sub>
CAN13	4D <sub>H</sub>	xx'0134 <sub>H</sub>	CAN_13IC	F15A <sub>H</sub>
CAN14	4E <sub>H</sub>	xx'0138 <sub>H</sub>	CAN_14IC	F15C <sub>H</sub>

**中断与异常情况控制**

中断源或 PEC 服务请求	强制中断编号	向量地址 <sup>1)</sup>	控制寄存器	寄存器地址
CAN15	4F <sub>H</sub>	xx'013C <sub>H</sub>	CAN_15IC	F15E <sub>H</sub>
USIC0 CH0 SR0	50 <sub>H</sub>	xx'0140 <sub>H</sub>	U0C0_0IC	F120 <sub>H</sub>
USIC0 CH0 SR1	51 <sub>H</sub>	xx'0144 <sub>H</sub>	U0C0_1IC	F122 <sub>H</sub>
USIC0 CH0 SR2	52 <sub>H</sub>	xx'0148 <sub>H</sub>	U0C0_2IC	F124 <sub>H</sub>
USIC0 CH1 SR0	53 <sub>H</sub>	xx'014C <sub>H</sub>	U0C1_0IC	F126 <sub>H</sub>
USIC0 CH1 SR1	54 <sub>H</sub>	xx'0150 <sub>H</sub>	U0C1_1IC	F128 <sub>H</sub>
USIC0 CH1 SR2	55 <sub>H</sub>	xx'0154 <sub>H</sub>	U0C1_2IC	F12A <sub>H</sub>
USIC1 CH0 SR0	56 <sub>H</sub>	xx'0158 <sub>H</sub>	U1C0_0IC	F12C <sub>H</sub>
USIC1 CH0 SR1	57 <sub>H</sub>	xx'015C <sub>H</sub>	U1C0_1IC	F12E <sub>H</sub>
USIC1 CH0 SR2	58 <sub>H</sub>	xx'0160 <sub>H</sub>	U1C0_2IC	F130 <sub>H</sub>
USIC1 CH1 SR0	59 <sub>H</sub>	xx'0164 <sub>H</sub>	U1C1_0IC	F132 <sub>H</sub>
USIC1 CH1 SR1	5A <sub>H</sub>	xx'0168 <sub>H</sub>	U1C1_1IC	F134 <sub>H</sub>
USIC1 CH1 SR2	5B <sub>H</sub>	xx'016C <sub>H</sub>	U1C1_2IC	F136 <sub>H</sub>
USIC2 CH0 SR0	5C <sub>H</sub>	xx'0170 <sub>H</sub>	U2C0_0IC	F138 <sub>H</sub>
USIC2 CH0 SR1	5D <sub>H</sub>	xx'0174 <sub>H</sub>	U2C0_1IC	F13A <sub>H</sub>
USIC2 CH0 SR2	5E <sub>H</sub>	xx'0178 <sub>H</sub>	U2C0_2IC	F13C <sub>H</sub>
USIC2 CH1 SR0	5F <sub>H</sub>	xx'017C <sub>H</sub>	U2C1_0IC	F13E <sub>H</sub>
USIC2 CH1 SR1	60 <sub>H</sub>	xx'0180 <sub>H</sub>	U2C1_1IC	F180 <sub>H</sub>
USIC2 CH1 SR2	61 <sub>H</sub>	xx'0184 <sub>H</sub>	U2C1_2IC	F182 <sub>H</sub>
USIC3 CH0 SR0	62 <sub>H</sub>	xx'0188 <sub>H</sub>	U3C0_0IC	F184 <sub>H</sub>
USIC3 CH0 SR1	63 <sub>H</sub>	xx'018C <sub>H</sub>	U3C0_1IC	F186 <sub>H</sub>
USIC3 CH0 SR2	64 <sub>H</sub>	xx'0190 <sub>H</sub>	U3C0_2IC	F188 <sub>H</sub>
USIC3 CH1 SR0	65 <sub>H</sub>	xx'0194 <sub>H</sub>	U3C1_0IC	F18A <sub>H</sub>
USIC3 CH1 SR1	66 <sub>H</sub>	xx'0198 <sub>H</sub>	U3C1_1IC	F18C <sub>H</sub>
USIC3 CH1 SR2	67 <sub>H</sub>	xx'019C <sub>H</sub>	U3C1_2IC	F18E <sub>H</sub>
SCU 外部请求 0	68 <sub>H</sub>	xx'01A0 <sub>H</sub>	ERU_0IC	F190 <sub>H</sub>

**中断与异常情况控制**

中断源或 PEC 服务请求	强制中断编号	向量地址 <sup>1)</sup>	控制寄存器	寄存器地址
SCU 外部请求 1	69 <sub>H</sub>	xx'01A4 <sub>H</sub>	ERU_1IC	F192 <sub>H</sub>
SCU 外部请求 2	6A <sub>H</sub>	xx'01A8 <sub>H</sub>	ERU_2IC	F194 <sub>H</sub>
SCU 中断 1	6B <sub>H</sub>	xx'01AC <sub>H</sub>	SCU_1IC	F196 <sub>H</sub>
SCU 中断 0	6C <sub>H</sub>	xx'01B0 <sub>H</sub>	SCU_0IC	F198 <sub>H</sub>
SCU 外部请求 3	6D <sub>H</sub>	xx'01B4 <sub>H</sub>	ERU_3IC	F19A <sub>H</sub>
RTC	6E <sub>H</sub>	xx'01B8 <sub>H</sub>	RTC_IC	F19C <sub>H</sub>
PEC 结束子通道	6F <sub>H</sub>	xx'01BC <sub>H</sub>	EOPIC	F19E <sub>H</sub>

1) 寄存器 VECSEG 决定向量表所在的段。

CPUCON1 寄存器中的 VECSC 位域定义了两个相邻向量之间的间隔。在该表中使用缺省设置，即两个相邻向量之间的距离为 4 个字节（两个字）。

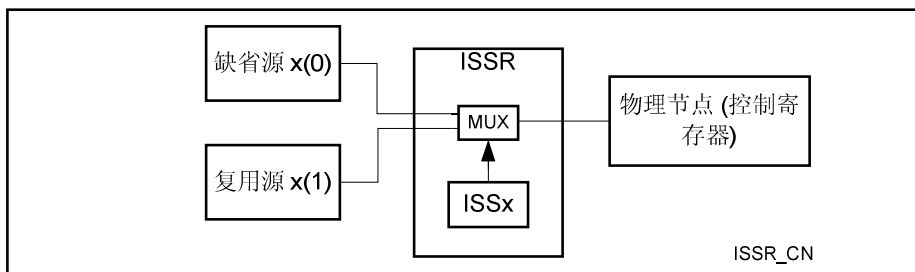
## 7.14.2 中断节点共享

中断源的选择可根据应用的侧重不同加以调整。本章中描述的概念允许将重点放在控制（CAPCOM2）或通信（USIC）方面。

由 SCU 寄存器 ISSR（定义见[章节 7.14.3](#)）控制中断节点共享。

### 由 ISSR 控制的共享节点

下图给出由 ISSR 寄存器控制共享节点的原理。缺省的中断源是 CAPCOM2。其它可选的中断源为 SCU 中断 2 和 3 以及所有 USIC 通道的 SR3 请求。



**图 7-6 由 ISSR 控制的节点共享原理**

下表列出发送至物理中断节点（控制寄存器）的可能的中断请求源。由 ISSR 寄存器的 ISSx 位进行选择。

**表 7-10      由 ISSR 控制的节点共享**

控制寄存器	选择位	缺省源 (ISSx=0)	其它可选源 (ISSx=1)
CC2_CC16IC	ISS0	CAPCOM2 请求 16	(未分配)
CC2_CC17IC	ISS1	CAPCOM2 请求 17	(未分配)
CC2_CC18IC	ISS2	CAPCOM2 请求 18	USIC3 CH0 SR3
CC2_CC19IC	ISS3	CAPCOM2 请求 19	USIC3 CH1 SR3
CC2_CC20IC	ISS4	CAPCOM2 请求 20	USIC0 CH0 SR3
CC2_CC21IC	ISS5	CAPCOM2 请求 21	USIC0 CH1 SR3
CC2_CC22IC	ISS6	CAPCOM2 请求 22	USIC1 CH0 SR3
CC2_CC23IC	ISS7	CAPCOM2 请求 23	USIC1 CH1 SR3
CC2_CC24IC	ISS8	CAPCOM2 请求 24	(未分配)
CC2_CC25IC	ISS9	CAPCOM2 请求 25	(未分配)
CC2_CC26IC	ISS10	CAPCOM2 请求 26	(未分配)
CC2_CC27IC	ISS11	CAPCOM2 请求 27	(未分配)
CC2_CC28IC	ISS12	CAPCOM2 请求 28	USIC2 CH0 SR3
CC2_CC29IC	ISS13	CAPCOM2 请求 29	USIC2 CH1 SR3
CC2_CC30IC	ISS30	CAPCOM2 请求 30	SCU 中断 2
CC2_CC31IC	ISS31	CAPCOM2 请求 31	SCU 中断 3

### 7.14.3 中断源选择寄存器

为了将整个系统内的中断请求源分配给可用的中断节点，部分模块共用一些中断节点。

#### SCU\_ISSR

中断请求源选择寄存器

SFR (FF2E<sub>H</sub>/97<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ISS 15	ISS 14	ISS 13	ISS 12	ISS 11	ISS 10	ISS9	ISS8	ISS7	ISS6	ISS5	ISS4	ISS3	ISS2	ISS1	ISS0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
ISS0	0	rw	<b>CC2_CC16IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 16 中断。 1 <sub>B</sub> 未分配给中断请求源。
ISS1	1	rw	<b>CC2_CC17IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 17 中断。 1 <sub>B</sub> 未分配给中断请求源。
ISS2	2	rw	<b>CC2_CC18IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 18 中断。 1 <sub>B</sub> 选择 USIC3 通道 0 SR3。
ISS3	3	rw	<b>CC2_CC19IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 19 中断。 1 <sub>B</sub> 选择 USIC3 通道 1 SR3。
ISS4	4	rw	<b>CC2_CC20IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 20 中断。 1 <sub>B</sub> 选择 USIC0 通道 0 SR3。
ISS5	5	rw	<b>CC2_CC21IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 21 中断。 1 <sub>B</sub> 选择 USIC0 通道 1 SR3。

符号	位序号	类型	功能描述
<b>ISS6</b>	6	rw	<b>CC2_CC22IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 22 中断。 1 <sub>B</sub> 选择 USIC1 通道 0 SR3。
<b>ISS7</b>	7	rw	<b>CC2_CC23IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 23 中断。 1 <sub>B</sub> 选择 USIC1 通道 1 SR3。
<b>ISS8</b>	8	rw	<b>CC2_CC24IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 24 中断。 1 <sub>B</sub> 未分配中断请求源。
<b>ISS9</b>	9	rw	<b>CC2_CC25IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 25 中断。 1 <sub>B</sub> 未分配中断请求源。
<b>ISS10</b>	10	rw	<b>CC2_CC26IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 26 中断。 1 <sub>B</sub> 未分配中断请求源。
<b>ISS11</b>	11	rw	<b>CC2_CC27IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 27 中断。 1 <sub>B</sub> 未分配中断请求源。
<b>ISS12</b>	12	rw	<b>CC2_CC28IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 28 中断。 1 <sub>B</sub> 选择 USIC2 通道 0 SR3。
<b>ISS13</b>	13	rw	<b>CC2_CC29IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 29 中断。 1 <sub>B</sub> 选择 USIC2 通道 1 SR3。
<b>ISS14</b>	14	rw	<b>CC2_CC30IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 30 中断。 1 <sub>B</sub> 选择 SCU 中断 2。

符号	位序号	类型	功能描述
ISS15	15	rw	<b>CC2_CC31IC 的中断请求源选择</b> 0 <sub>B</sub> 选择 CC2 通道 31 中断。 1 <sub>B</sub> 选择 SCU 中断 3。

## 7.15 中断和 PEC 配置寄存器

下表列出所有用于配置 XE16xyM 中断和 PEC 操作的所有寄存器。寄存器按照地址进行排序。分配到每个中断请求的中断控制寄存器 xxIC 单独列出（见[章节 7.14](#)）。

可位寻址的 SFR 在“寄存器名称”列中用字母“b”标出。

**表 7-11 中断和 PEC 操作寄存器综述 — 按照地址排序**

名称	物理地址	8 位地址	描述	复位值
<b>FINT0CSP</b>	EC00 <sub>H</sub>	--	快速中断 0 CSP 寄存器	0000 <sub>H</sub>
<b>FINT0ADDR</b>	EC02 <sub>H</sub>	--	快速中断 0 地址寄存器	0000 <sub>H</sub>
<b>FINT1CSP</b>	EC04 <sub>H</sub>	--	快速中断 1 CSP 寄存器	0000 <sub>H</sub>
<b>FINT1ADDR</b>	EC06 <sub>H</sub>	--	快速中断 1 地址寄存器	0000 <sub>H</sub>
<b>BNKSEL0</b>	EC20 <sub>H</sub>	--	寄存器组选择寄存器 0	0000 <sub>H</sub>
<b>BNKSEL1</b>	EC22 <sub>H</sub>	--	寄存器组选择寄存器 1	0000 <sub>H</sub>
<b>BNKSEL2</b>	EC24 <sub>H</sub>	--	寄存器组选择寄存器 2	0000 <sub>H</sub>
<b>BNKSEL3</b>	EC26 <sub>H</sub>	--	寄存器组选择寄存器 3	0000 <sub>H</sub>
<b>SRCP0</b>	EC40 <sub>H</sub>	--	PEC 通道 0 源指针	0000 <sub>H</sub>
<b>DSTP0</b>	EC42 <sub>H</sub>	--	PEC 通道 0 目的指针	0000 <sub>H</sub>
<b>SRCP1</b>	EC44 <sub>H</sub>	--	PEC 通道 1 源指针	0000 <sub>H</sub>
<b>DSTP1</b>	EC46 <sub>H</sub>	--	PEC 通道 1 目的指针	0000 <sub>H</sub>
<b>SRCP2</b>	EC48 <sub>H</sub>	--	PEC 通道 2 源指针	0000 <sub>H</sub>
<b>DSTP2</b>	EC4A <sub>H</sub>	--	PEC 通道 2 目的指针	0000 <sub>H</sub>
<b>SRCP3</b>	EC4C <sub>H</sub>	--	PEC 通道 3 源指针	0000 <sub>H</sub>



**中断与异常情况控制**

名称	物理地址	8 位地址	描述	复位值
<b>DSTP3</b>	EC4E <sub>H</sub>	--	PEC 通道 3 目的指针	0000 <sub>H</sub>
<b>SRCP4</b>	EC50 <sub>H</sub>	--	PEC 通道 4 源指针	0000 <sub>H</sub>
<b>DSTP4</b>	EC52 <sub>H</sub>	--	PEC 通道 4 目的指针	0000 <sub>H</sub>
<b>SRCP5</b>	EC54 <sub>H</sub>	--	PEC 通道 5 源指针	0000 <sub>H</sub>
<b>DSTP5</b>	EC56 <sub>H</sub>	--	PEC 通道 5 目的指针	0000 <sub>H</sub>
<b>SRCP6</b>	EC58 <sub>H</sub>	--	PEC 通道 6 源指针	0000 <sub>H</sub>
<b>DSTP6</b>	EC5A <sub>H</sub>	--	PEC 通道 6 目的指针	0000 <sub>H</sub>
<b>SRCP7</b>	EC5C <sub>H</sub>	--	PEC 通道 7 源指针	0000 <sub>H</sub>
<b>DSTP7</b>	EC5E <sub>H</sub>	--	PEC 通道 7 目的指针	0000 <sub>H</sub>
<b>PECSEG0</b>	EC80 <sub>H</sub>	--	PEC 指针 0 段地址	0000 <sub>H</sub>
<b>PECSEG1</b>	EC82 <sub>H</sub>	--	PEC 指针 1 段地址	0000 <sub>H</sub>
<b>PECSEG2</b>	EC84 <sub>H</sub>	--	PEC 指针 2 段地址	0000 <sub>H</sub>
<b>PECSEG3</b>	EC86 <sub>H</sub>	--	PEC 指针 3 段地址	0000 <sub>H</sub>
<b>PECSEG4</b>	EC88 <sub>H</sub>	--	PEC 指针 4 段地址	0000 <sub>H</sub>
<b>PECSEG5</b>	EC8A <sub>H</sub>	--	PEC 指针 5 段地址	0000 <sub>H</sub>
<b>PECSEG6</b>	EC8C <sub>H</sub>	--	PEC 指针 6 段地址	0000 <sub>H</sub>
<b>PECSEG7</b>	EC8E <sub>H</sub>	--	PEC 指针 7 段地址	0000 <sub>H</sub>
<b>PECISNC b</b>	FFD8 <sub>H</sub>	EC <sub>H</sub>	PEC 中断子节点控制	0000 <sub>H</sub>
<b>PECC0</b>	FEC0 <sub>H</sub>	60 <sub>H</sub>	PEC 通道 0 控制寄存器	0000 <sub>H</sub>
<b>PECC1</b>	FEC2 <sub>H</sub>	61 <sub>H</sub>	PEC 通道 1 控制寄存器	0000 <sub>H</sub>
<b>PECC2</b>	FEC4 <sub>H</sub>	62 <sub>H</sub>	PEC 通道 2 控制寄存器	0000 <sub>H</sub>
<b>PECC3</b>	FEC6 <sub>H</sub>	63 <sub>H</sub>	PEC 通道 3 控制寄存器	0000 <sub>H</sub>
<b>PECC4</b>	FEC8 <sub>H</sub>	64 <sub>H</sub>	PEC 通道 4 控制寄存器	0000 <sub>H</sub>
<b>PECC5</b>	FECA <sub>H</sub>	65 <sub>H</sub>	PEC 通道 5 控制寄存器	0000 <sub>H</sub>

**中断与异常情况控制**

名称	物理地址	8 位地址	描述	复位值
<b>PECC6</b>	FECC <sub>H</sub>	66 <sub>H</sub>	PEC 通道 6 控制寄存器	0000 <sub>H</sub>
<b>PECC7</b>	FECE <sub>H</sub>	67 <sub>H</sub>	PEC 通道 7 控制寄存器	0000 <sub>H</sub>

## 8 系统控制单元 (SCU)

XE16xyM 的系统控制单元 (SCU) 负责处理除调试之外的所有系统控制任务, 和调试相关的任务由 OCDS/Cerberus 控制。本章描述的所有系统功能均紧密相关, 因此它们可集中通过 SCU 方便的管理。

SCU 包含以下功能子模块:

- 时钟产生 (见 [章节 8.1](#))
- 系统定时器 (见 [章节 8.2](#))
- 唤醒定时器 (见 [章节 8.3](#))
- 复位操作 (见 [章节 8.4](#))
- 外部服务请求 (见 [章节 8.5](#))
- 电源电压及控制 (见 [章节 8.6](#))
- 全局状态控制 (见 [章节 8.7](#))
- 软件启动支持 (见 [章节 8.8](#))
- 外部请求单元 (见 [章节 8.9](#))
- 中断产生 (见 [章节 8.10](#))
- 温度补偿 (见 [章节 8.11](#))
- 看门狗定时器 (见 [章节 8.12](#))
- 强制中断产生 (见 [章节 8.13](#))
- 存储器内容保护 (见 [章节 8.14](#))
- 寄存器访问控制 (见 [章节 8.15](#))
- 其它系统寄存器 (见 [章节 8.16](#))
- 实现 (见 [章节 8.17](#))
- SCU 寄存器和地址映射 (见 [章节 8.18](#))

### 重要信息: 寄存器编程

XE16xyM 中部分寄存器在系统启动过程中的初始化值和它的复位值 (在相应的寄存器描述中定义) 不同。这些寄存器归纳见章节 “由启动过程修改的寄存器”。

系统控制单元中包含很多特殊功能寄存器, 不能以任意的顺序对其编程 (尤其在使用内部电压调节器的情况下)。为了避免由于不恰当的启动而造成系统出错、为了能够对敏感特性 (如供电电源和时钟产生) 进行便捷的配置和控制, 将在《程序员指南》中给出对寄存器序列进行编程的建议和示例。

尤其是对以下寄存器进行更新时必须更加谨慎:

- 时钟产生单元: WUOSCCON、HPOSCCON、PLLOSCCON、PLLCONx

- 电源: EVR1CON0、EVR1SET15VHP、EVRMCON0、EVRMSET15VHP、PVC1CON0、PVC1MCON0、SWDCON0
- 系统: SYSCON0

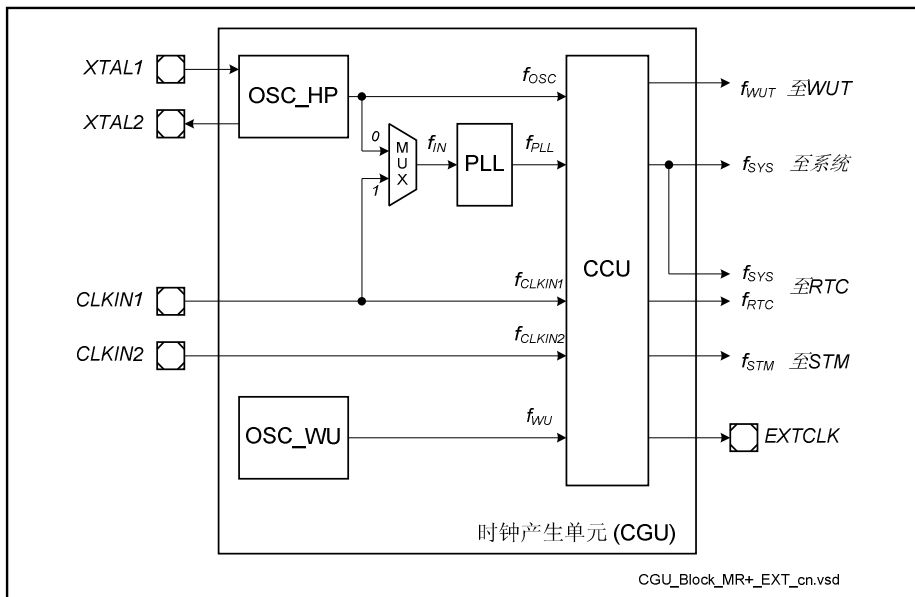
## 8.1 时钟产生单元

时钟产生单元 (CGU) 可非常灵活的产生 XE16xyM 所需的时钟。可在用户程序运行期间设定时钟频率，从而在实际应用状态下实现最佳性能功耗比。

### 8.1.1 概述

CGU 可将低频外部时钟转换为高频系统时钟，或无需外部输入产生高频系统时钟。

CGU 由时钟发生器和时钟控制单元 (CCU) 构成。



**图 8-1 时钟产生单元框图**

CGU 的输入连接描述见章节 8.17.1。

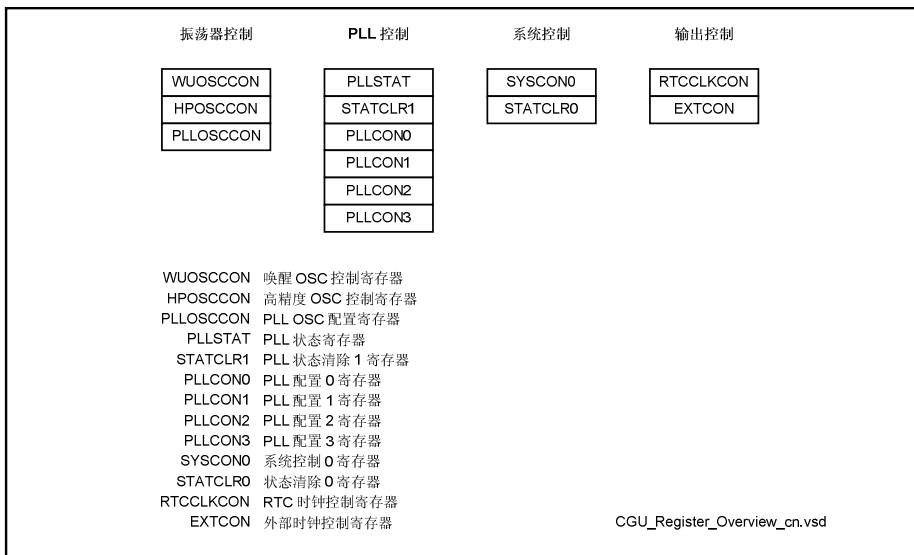
CGU 产生以下信号：

- 系统时钟  $f_{SYS}$
- RTC 计数时钟  $f_{RTC}$
- 唤醒定时器 (WUT) 时钟  $f_{WUT}$
- STM 时钟  $f_{STM}$
- 外部时钟  $f_{EXT}$

章节 8.1.5 和章节 8.1.6 描述了有哪些可选的时钟源、可从中产生哪些时钟信号。

## 寄存器概览

CGU 由下图所示的一组寄存器控制。



**图 8-2 时钟产生单元寄存器概览**

以下各章节分别描述 CGU 的不同单元。

### 8.1.2 均流控制的唤醒时钟 (OSC\_WU)

均流控制的唤醒时钟源提供一个用于控制内部操作的时钟信号，它独立于标准时钟源，无需外部器件。通过位域 **WUOSCCON.FREQSEL** 配置其输出频率  $f_{WU}$ ， $f_{WU}$  的频率范围介于 130 kHz 至 500 kHz 之间。

### 8.1.3 高精度振荡器电路 (OSC\_HP)

高精度振荡器电路可驱动一个外部晶振或接收一个外部时钟源。它包含一个反相放大器，以 XTAL1 为输入、XTAL2 为输出。

图 8-3 和图 8-4 给出外部输入时钟模式和外部晶振模式所对应的外部电路。

#### 8.1.3.1 外部输入时钟模式

在外部输入时钟模式下，可直接输入外部时钟信号而不使用外部晶振、旁路振荡器放大器。最大允许输入频率取决于引脚 XTAL1 的特性。

使用外部时钟信号时，它必须连接到 XTAL1 上。XTAL2 断开（不连接）。

注：XTAL1 上的电压必须遵循数据手册中规定的电压值。

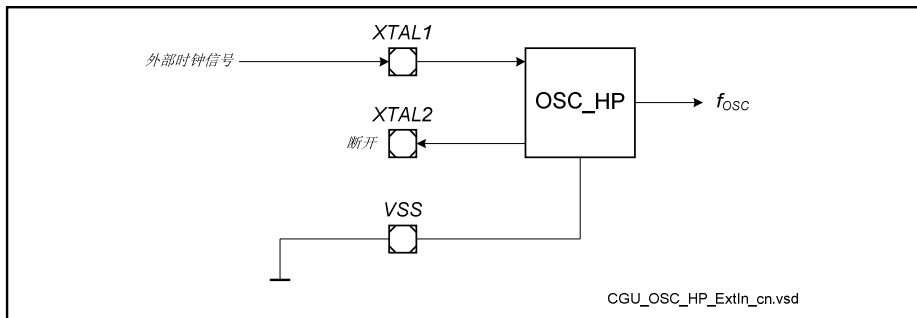


图 8-3 高精度振荡器的外部时钟输入模式

#### 8.1.3.2 外部晶振模式

在外部晶振模式下，必须使用外部振荡器负载电路和引脚 XTAL1、XTAL2。它通常由两个负载电容 C1 和 C2 组成，某些晶振可能还需要一个串联阻尼电阻。这些元件的准确值及工作范围和晶振频率有关，必须结合晶振厂商提供的参考值，通过负阻抗的方法确定并优化。



- 10 位输出分频器 **K2**: ( $K2DIV+1$  分频)
- 10 位 VCO 旁路分频器 **K1**: ( $K1DIV+1$  分频)
- 振荡器运行检测和看门狗
- 不同的工作模式
  - 预分频模式
  - 未锁相模式
  - 正常模式
- 不同的省电模式
  - 掉电模式
  - 休眠模式 (VCO 掉电)
- 无突变干扰编程输出分频器 **K2** 和 VCO 旁路分频器 **K1**
- 正常模式和预分频模式之间无突变干扰的切换
- 均流控制的内部时钟源

#### **8.1.4.2 PLL 功能描述**

PLL 由压控振荡器 (VCO) 和反馈通路组成。反馈通路上的分频器 ( $N$  分频) 对 VCO 频率分频, 得到的结果和外部经  $P$  分频处理的参考频率进行比较。相位检测电路根据这两个时钟信号的差值相应控制 VCO 频率 ( $f_{VCO}$ )。PLL 锁相检测单元监控并指示该状态。相位检测逻辑继续监控这两个时钟信号, 如有需要则调整 VCO 时钟。PLL 输出时钟  $f_{PLL}$  由 VCO 时钟经  $K2$  分频产生、或由振荡器时钟经  $K1$  分频产生。

PLL 框图如下图所示。



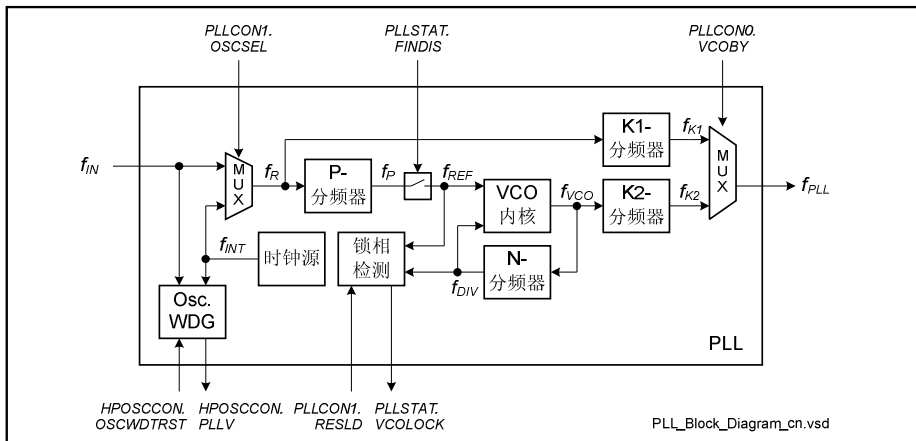


图 8-5 PLL 框图

### 时钟源控制

参考频率  $f_R$  可取自均流控制的内部时钟源  $f_{INT}$ ，也可取自外部时钟源  $f_{IN}$ 。

### PLL 模式

PLL 时钟  $f_{PLL}$  的产生有以下几种模式（软件选择模式）：

- 正常模式
- 预分频模式
- 未锁相模式

**正常模式下**，参考频率  $f_R$  经 P 分频、N 倍频和 K2 分频产生 PLL 时钟：

$$f_{PLL} = \frac{N}{P \times K2} \times f_R \quad (8.1)$$

**预分频模式下**，参考频率  $f_R$  经 K1 分频产生 PLL 时钟：

$$f_{PLL} = \frac{f_R}{K1} \quad (8.2)$$

**未锁相模式下**，PLL 以压控振荡器（VCO）基频  $f_{VCObase}$  工作， $f_{VCObase}$  经 K2 分频产生 PLL 时钟：

$$f_{PLL} = \frac{f_{VCObase}}{K2} \quad (8.3)$$

## PLL 省电模式

**PLL 掉电模式:** 若完全不需要 PLL，PLL 支持掉电模式以降低功耗。PLL 进入掉电模式时，不产生 PLL 输出频率。

**PLL 休眠模式:** PLL 支持休眠模式（也称作 VCO 掉电模式）以降低 PLL 内部的功耗。PLL 进入休眠模式时，只能选用预分频模式。

### 8.1.4.3 PLL 模式的配置和操作

以下各节分别描述不同 PLL 模式的配置和操作。

#### 未锁相模式的配置和操作

未锁相模式下，PLL 以 VCO 基频工作， $f_{VCO}$  经 K2 分频产生  $f_{PLL}$ 。

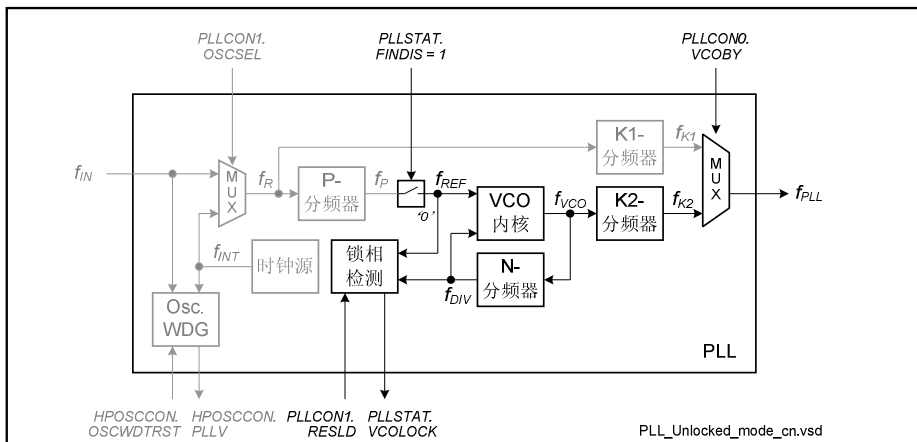


图 8-6 PLL 未锁相模式框图

通过以下设置选择未锁相模式：

- $STATCLR1.SETFINDIS = 1$
- $PLLCON0.VCOBY = 0$

当满足以下所有条件时，进入未锁相模式：

- $PLLSTAT.FINDIS = 1$
- $PLLSTAT.VCOBYST = 1$



正常模式下的操作需要输入时钟  $f_R$ 。若选择  $f_{IN}$  作为  $f_R$ ，建议通过查询 HPOSICON.PLLV 检查和监控输入频率  $f_R$  是否可用。

正常模式下的系统工作频率由 P、N 和 K2 这三个分频因子控制。修改 P 和 N 将直接影响 VCO 频率并导致 VCO 失锁；修改 K2 不会影响 VCO 的锁定状态，但会改变 PLL 输出频率。

*注：通过改变分频因子 K2 以修改系统工作频率时，将直接影响器件的功耗。因此，必须谨慎执行该操作。*

要修改或进入正常模式频率时，需按以下步骤进行：

配置和进入预分频模式。详见预分频模式的描述。

禁止产生 VCO 失锁强制中断。

在使用预分频模式时，可配置正常模式并检查 VCO 的正锁定状态。正常模式的首要目标频率应该选择等于或略高于预分配模式使用的频率。这样可避免系统工作频率的较大改变，从而避免了从预分频模式切换回正常模式时功耗改变较大。应按以下方式选择 P 和 N：

- 若选择 P 和 N 使得  $f_{VCO}$  在允许取值范围的低位区域，将导致功耗略微下降、抖动略微增加
- 若选择 P 和 N 使得  $f_{VCO}$  在允许取值范围的高位区域，将导致功耗略微增加、抖动略微下降

P 和 N 更新后，应等待 VCO 锁定状态 ( $PLLSTAT.VCOLOCK = 1$ )。

*注：在配置新的分频因子之后，建议复位 VCO 锁定检测 ( $PLLCON1.RESLD = 1$ )，从而得到一个已确定的 VCO 锁定检查时间。*

VCO 锁定后，可从预分频模式切换到正常模式。通过清除  $PLLCON0.VCOBY$  请求进入正常模式。状态位  $PLLSTAT.VCOBYST$  置位时，进入正常模式。

此时已进入正常模式。VCO 锁定强制中断的状态标志应被清零并被重新使能。

可通过只修改分频因子 K2 来配置期望的 PLL 输出目标频率。根据 K2 的选取值相应选择时钟的占空比。这会影响到外部通信接口的操作。为了避免频率改变过大，有必要分步修改 K2。若 K2 已被修改，不应在  $PLLSTAT.K2RDY$  置位之前对其进行下一次更新。

*注：编程指南中描述了使用平滑频率阶跃来适当调整内部电压调节器的负载。*

## **PLL VCO 锁定检测**

PLL 具有锁相检测功能，可通过监控 PLL 的 VCO 部分以检测不稳定的 VCO 电路行为。若输入  $f_{REF}$  和  $f_{DIV}$  的差值过大，锁相检测单元将会指示 VCO 电路以及 VCO 输出  $f_{VCO}$  不稳定。这一个（或两个）输入频率的变化若小于某个量级，则不被标记为失锁，这是因为 VCO 可以处理如此小的改变，不会对系统造成任何影响。

## PLL VCO 失锁事件

晶振或外部时钟线损坏可能导致 PLL 失锁，此时产生强制中断（若允许/使能产生该中断）。此外，要将时钟  $f_R$  和 PLL VCO 断开，从而避免振荡器电路中的噪声或偶发时钟脉冲造成 PLL 操作不稳定。没有输入时钟  $f_R$  时，PLL 会逐渐降至 VCO 基频工作。通过置位 PLLCON1.EMFINDISEN，产生 VCO 失锁事件时 VCO 可自动和输入时钟  $f_R$  断开。

## 预分频模式的配置和操作

预分频模式下，PLL 以频率  $f_{PLL}$  工作。参考频率  $f_R$  经 K1 分频产生 PLL 时钟。

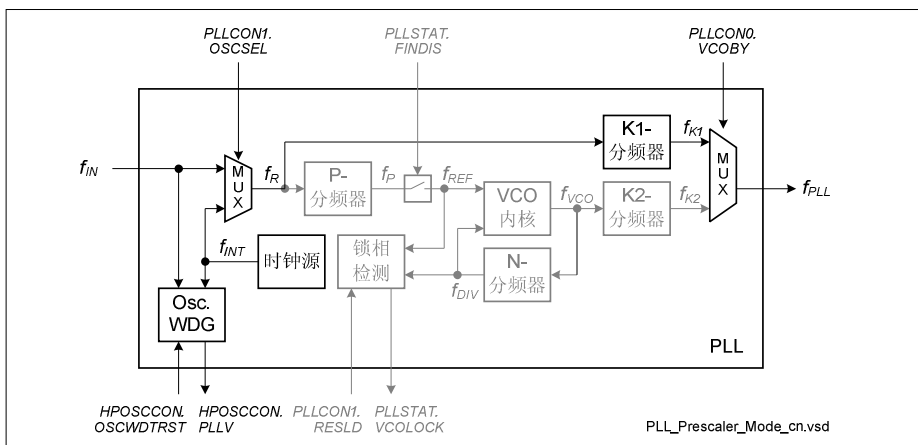


图 8-8 PLL 预分频模式框图

通过以下设置选择预分频模式：

- PLLCON0.VCOBY = 1

当满足以下所有条件时，进入预分频模式：

- PLLSTAT.VCOBYST = 0
- HPOSICON.PLLV = 1

预分频模式下的操作需要输入时钟  $f_R$ 。若选择  $f_{IN}$  作为  $f_R$ ，建议通过查询 HPOSICON.PLLV 检查和监控输入频率  $f_{OSC}$  是否可用。对  $f_R$  的频率无要求。

预分频模式下的系统工作频率由分频器 K1 的值控制。若 PLLCON1.K1DIV 已被修改，不应在 PLLSTAT.K1RDY 置位之前对其进行下一次更新。

*注：通过改变分频因子 K1 以修改系统工作频率时，将直接影响器件的功耗。因此，必须谨慎执行该操作。*

时钟信号的占空比和 K1 的选取值有关。这会影响和外部通信接口的操作。

可通过置位 PLLCON0.VCOBY 请求从未锁相模式或正常模式进入预分频模式。状态位 PLLSTAT.VCOBYST 被清零后，进入预分频模式。

在请求预分频模式之前，K1 应被配置为尽可能使产生的 PLL 输出频率  $f_{PLL}$  和未锁相或正常模式下产生的频率相匹配，这样的话，模式切换所导致的频率变化被降至最小。

通过清除 PLLCON0.VCOBY 请求退出预分频模式。状态位 PLLSTAT.VCOBYST 被置位时退出预分频模式。

### **PLL 掉电模式的配置和操作**

可通过置位 PLLCON0.PLLPWD 进入掉电模式。PLL 进入掉电模式时，不产生 PLL 输出频率。

### **PLL 休眠模式的配置和操作**

可通过置位 PLLCON0.VCOPWD 进入休眠模式（也称作 VCO 掉电模式）。PLL 进入休眠模式时，只能选用预分频模式。选择休眠模式不会自动切换到预分频模式。因此，在进入休眠模式之前必须激活预分频模式。

#### **8.1.4.4 电压调节器**

PLL 的模拟部分（VCO、均流控制的时钟源）工作在特定的电压下，该电压由 PLL 内部的一个专用电压调节器产生。

必须在激活 PLL 的模拟部分之前使能该电压调节器。也就是说，在电压稳定之前，均流控制的时钟源和 VCO 必须保持关闭状态。模拟部分被激活后，PLL 电压调节器需要一定的上电时间以对 PLL 的模拟部分正确供电。

PLL 数字部分掉电时应关闭电压调节器，此时必须考虑到 PLL 的数字部分在进入掉电模式过程中需要一个有效时钟。在这种情况下，该时钟由 PLL 的一个振荡器产生，必须在电压调节器被禁用之前进入 PLL 掉电模式。VCO 和均流控制的时钟源可被同时激活或关闭。

#### **8.1.4.5 分频器握手接口**

PLL 为各分频器提供握手接口。本节说明改变配置时如何执行握手操作。

一般的握手操作对所有接口均适用。这里将结合重新编程分频器举例说明如何执行握手操作。

*注：只有按照握手协议设置新值（如分频值）时，所描述的握手才有效。*

应按以下步骤进行握手：

1. 清除应答位并设置新分频值
2. 查询准备就绪位是否为 0
3. 置位应答位
4. 查询准备就绪位是否为 1

即使对于握手从未被正确执行、准备就绪位从一开始就为 0 的情况，该方法也将有效。在任何情况下，分频值改变都将使准备就绪位为 0。

#### 8.1.4.6 均流控制时钟

均流控制时钟源为 PLL 提供时钟  $f_{INT}$ 。

注：振荡器看门狗工作时也需使用时钟  $f_{INT}$ 。

#### 8.1.4.7 输入时钟选择

参考时钟  $f_R$  可由 PLL 输入时钟源  $f_{IN}$  或均流控制时钟源  $f_{INT}$  提供，由位 **PLLCON1.OSCSEL** 进行该时钟源的选择。

PLL 输入时钟  $f_{IN}$  可选择高精度振荡器时钟  $f_{OSC}$  或直接时钟输入  $f_{CLKIN1}$ ，由 **PLLCON1.INSEL** 进行选择。

#### 8.1.4.8 振荡器看门狗

振荡器看门狗持续监控输入时钟  $f_{IN}$ 。若输入频率过低或输入时钟出错，由 **HPOSCCON.PLLV = 0** 指示振荡器出错，产生中断请求。

置位 **HPOSCCON.OSCWDTRST**，无需复位整个 PLL 即可重新启动检测（比如在 VCO 失锁情况下）。

注：振荡器看门狗需用均流控制的时钟  $f_{INT}$  作为参考时钟。因此，只有当内部时钟源有效时，才能使用振荡器看门狗（**HPOSCCON.PLLV** 有效）。

#### 8.1.4.9 修改 PLL 参数

修改寄存器 **PLLCON0** 和 **PLLCON3** 中的 PLL 参数时，必须遵守以下约束：

- 可随时切换至 VCO 旁路，但必须确保不会超过器件的最大工作频率（见数据手册）。
- 应选择预分频模式。
- 切换到预分频模式后，可修改 **NDIV** 和 **PDIV**。
- 在取消预分频模式之前，必须置位 **RESLD**，随后监控 **VCLOCK** 标志位。只有当 **VCLOCK** 重新置位后才能退出预分频模式。

- 必须在选择预分频模式之后才可修改 VCOSSEL。

*注：在正常模式下也可修改 PDIV 和 NDIV。修改 NDIV 时，在 PLL 锁相之前 VCO 时钟  $f_{VCO}$  可能会超过目标频率。修改 PDIV 或 NDIV 之后，必须等待 PLL 锁相。通常使用上述过程逐步提高 VCO 时钟频率。*

### 8.1.5 时钟控制单元

时钟控制单元用于选择当前的时钟源，通过它们产生器件所需的时钟信号。CCU 可产生以下时钟：

- 系统时钟  $f_{SYS}$
- RTC 计数时钟  $f_{RTC}$
- WUT 时钟  $f_{WUT}$
- 系统定时器时钟  $f_{STM}$
- 输出时钟  $f_{EXT}$

可选择以下时钟信号：

- PLL 时钟  $f_{PLL}$
- 振荡器时钟 (OSC\_HP)  $f_{OSC}$
- 唤醒时钟  $f_{WU}$
- 输入 CLKIN1 作为直接时钟输入  $f_{CLKIN1}$
- 输入 CLKIN2 作为直接时钟输入  $f_{CLKIN2}$

#### 8.1.5.1 时钟产生

可选择不同的时钟源以产生时钟信号。

*注：系统启动过程会影响时钟源的选择。具体参见“启动过后器件的状态”一节中不同启动过程对应的寄存器设置。*

#### 系统时钟产生

可选择以下时钟源产生系统时钟  $f_{SYS}$ ：

- 唤醒时钟  $f_{WU}$
- 振荡器时钟 (OSC\_HP)  $f_{OSC}$
- PLL 时钟  $f_{PLL}$
- 输入 CLKIN1 作为直接时钟输入  $f_{CLKIN1}$



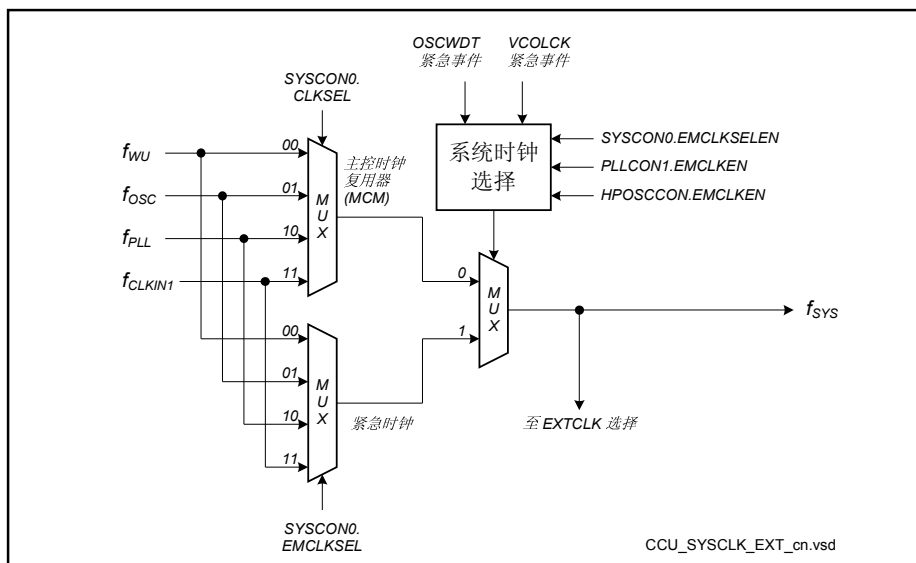


图 8-9 时钟控制单元，系统时钟产生

### RTC 时钟产生

对于 RTC 模块，可选择工作在同步或异步模式。可选择以下时钟源产生 RTC 的异步时钟：

- PLL 时钟  $f_{PLL}$
- 振荡器时钟 (OSC\_HP)  $f_{OSC}$
- 输入 CLKIN2 作为直接时钟输入  $f_{CLKIN2}$
- 唤醒时钟  $f_{WU}$

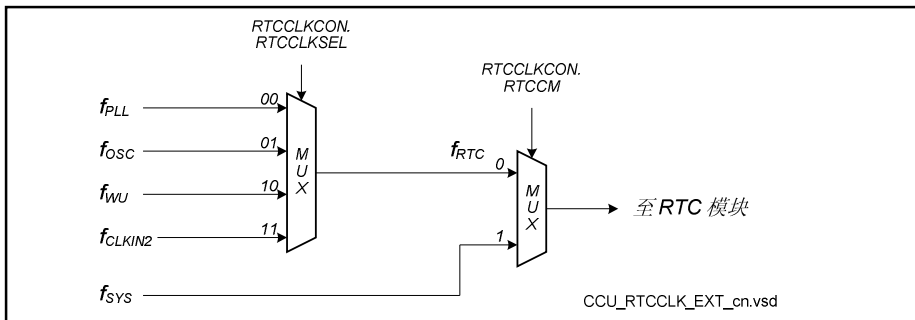


图 8-10 时钟控制单元，RTC 时钟产生

### 系统定时器 (STM) 时钟产生

可选择以下时钟源产生系统定时器时钟：

- 来自振荡器 OSC\_HP 的时钟  $f_{OSC}$
- PLL 时钟  $f_{PLL}$
- 输入 CLKIN1 作为直接时钟输入  $f_{CLKIN1}$
- 输入 CLKIN2 作为直接时钟输入  $f_{CLKIN2}$
- 唤醒时钟  $f_{WU}$

所选择的时钟可被分频，分频因子由 **STMCON.CLKDIV** 定义（见[章节 8.2.1.2](#)）。

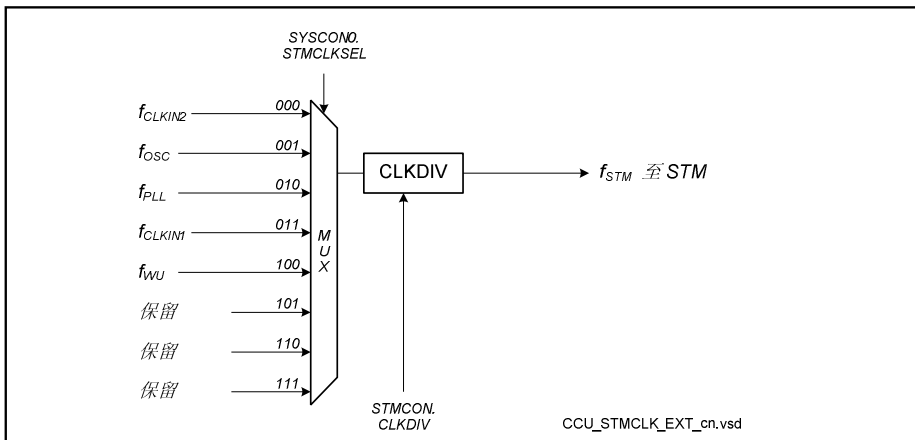


图 8-11 时钟控制单元，STM 时钟产生

## 唤醒定时器 (WUT) 时钟产生

可选择以下时钟源产生唤醒定时器时钟：

- 唤醒时钟  $f_{WU}$
- 来自振荡器 OSC\_HP 的时钟  $f_{OSC}$
- PLL 时钟  $f_{PLL}$
- 输入 CLKIN1 作为直接时钟输入  $f_{CLKIN1}$

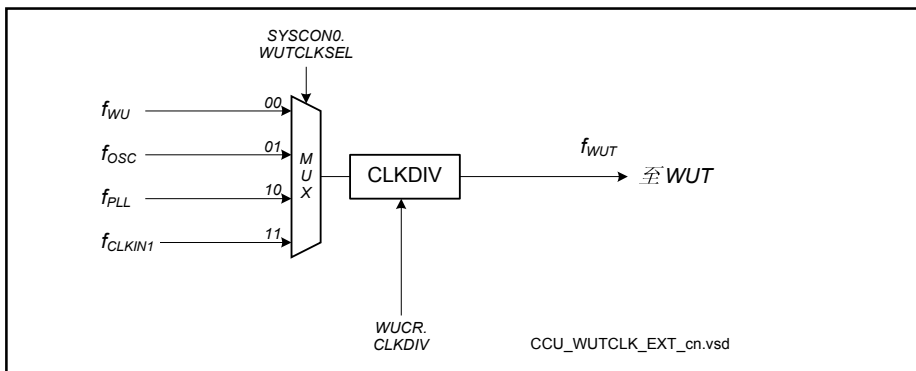


图 8-12 时钟控制单元，WUT 时钟产生

### 8.1.5.2 选择和修改工作频率

选择时钟源和时钟产生方式时，必须谨慎处理所需的参数（将其写入相应位域中）以避免不希望的中间状态。

对于许多实际应用场合，需要在器件工作期间修改系统时钟频率  $f_{SYS}$  以优化系统性能、降低功耗。修改工作频率会改变工作电流大小，从而影响功耗。因此，通过片上嵌入式电压调节器 (EVR) 产生内核电压时，只有在遵守规则（见数据手册）的前提下才能修改工作频率。

*注：为了避免出现问题，建议执行规定的序列，从而确保时钟系统操作与电源系统相配合。具体参见《程序员指南》。*

### 8.1.5.3 系统时钟紧急处理

若 PLL 和输入信号  $f_{IN}$  失锁，或输入时钟  $f_{IN}$  失效，这都会影响系统时钟  $f_{SYS}$  的产生。可检测到这两个事件并通知应用软件。如有必要，时钟系统会执行恰当的操作，因此，没有备选时钟信号时器件也不会停止工作。

### 振荡器看门狗事件

若外部时钟源的频率跌至振荡器看门狗 (OSCWDT) 的限定值以下 ([见章节 8.1.4.8](#))，产生系统时钟  $f_{\text{SYS}}$  的时钟源将切换至备选时钟源 (若  $\text{HPOSCCON.EMCLKEN} = 1$ )。在这种情况下，以下信息可用：

- 振荡器看门狗强制中断标志位 (TRAPSTAT.OSCWDTT) 置位，若允许产生强制中断 (TRAPDIS.OSCWDTT = 0)，发送至 CPU 的强制中断请求被激活。
- 时钟  $f_{\text{IN}}$  丢失时，位  $\text{HPOSCCON.PLLV} = 0$ 。
- 若  $\text{SYSCON0.EMCLKSELEN}$  置位，位  $\text{SYSCON0.EMSOSC}$  置位。
- 若  $\text{SYSCON0.EMCLKSELEN} = 1$ ，系统时钟  $f_{\text{SYS}}$  的时钟源将切换至由  $\text{SYSCON0.EMCLKSEL}$  选择的备选时钟源。这由  $\text{SYSCON0.SELSTAT} = 1$  指示。

### PLL VCO 失锁事件

若 PLL 输出频率不再锁定其输入频率  $f_{\text{IN}}$ ，在  $\text{PLLCON1.EMFINDISEN} = 1$  的情况下，PLL 从正常模式切换到未锁相模式。在这种情况下，以下信息可用：

- PLL VCO 失锁强制中断标志位 (TRAPSTAT.VCOLCKT) 置位，若允许产生强制中断 (TRAPDIS.VCOLCKT = 0)，发送至 CPU 的强制中断请求被激活。
- PLL 未锁相时，位  $\text{PLLSTAT.VCOLOCK} = 0$ 。
- 若  $\text{SYSCON0.EMCLKSELEN}$  置位，位  $\text{SYSCON0.EMSVCO}$  置位。
- PLL VCO 时钟输入被断开 ( $\text{PLLSTAT.FINDIS} = 1$ )，PLL 逐渐降至 VCO 基频工作。

### 系统行为

可用备选时钟 (紧急时钟或 VCO 基频) 执行紧急事件中断程序，系统随后进入安全状态并停止工作、或切换到紧急工作模式 (此时性能和/或特性下降)。

«程序员指南»将说明如何使能这些特性以及如何响应这两种事件。

#### 8.1.6 外部时钟输出

可由引脚 EXTCLK 提供外部时钟输出，用作外部系统的输入时钟或用于观测所选的器件时钟。通过置位  $\text{EXTCON.EN}$ 、选择时钟信号作为引脚 EXTCLK 的复用输出功能从而使能该外部时钟。可由  $\text{EXTCON.SEL}$  选择以下时钟源产生外部时钟  $f_{\text{EXT}}$ ：

- 系统时钟  $f_{\text{SYS}}$
- 可编程时钟输出  $f_{\text{OUT}}$
- 来自振荡器 OSC\_HP 的时钟  $f_{\text{OSC}}$
- 直接时钟输入  $f_{\text{CLKIN1}}$

- PLL 时钟  $f_{PLL}$
- 唤醒时钟  $f_{WU}$
- RTC 时钟  $f_{RTC}$

注：修改位域 `EXTCON.SEL` 会导致引脚 `EXTCLK` 上产生毛刺。

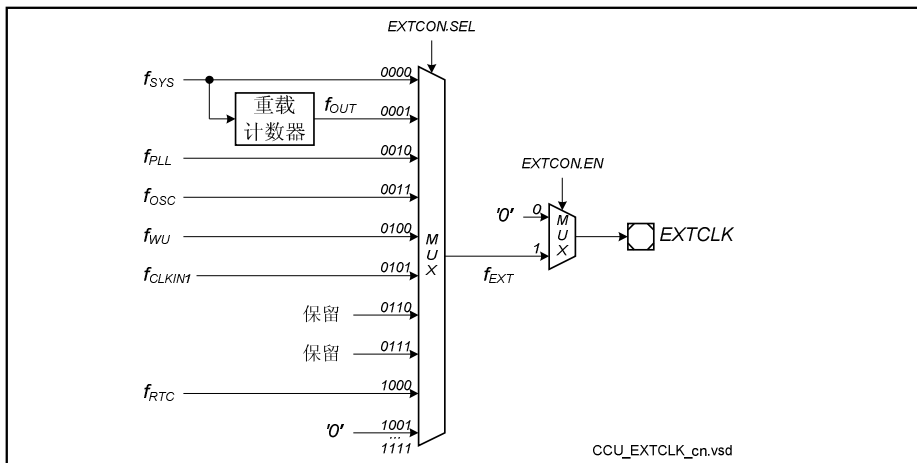


图 8-13 EXTCLK 时钟产生

### 8.1.6.1 可编程频率输出

可编程频率输出  $f_{OUT}$  可选择作为时钟输出 (`EXTCLK`)。该时钟可由软件控制，从而满足不同外部电路的需求。可编程特性还将功率管理提升到了系统级，同样，XE16xyM 的外围电路（外设等）也会受到影响，即外围电路可运行在不同的频率、或暂时完全关闭（无时钟）。

时钟  $f_{OUT}$  由重载计数器产生，因此可通过较小的步长选择输出频率。

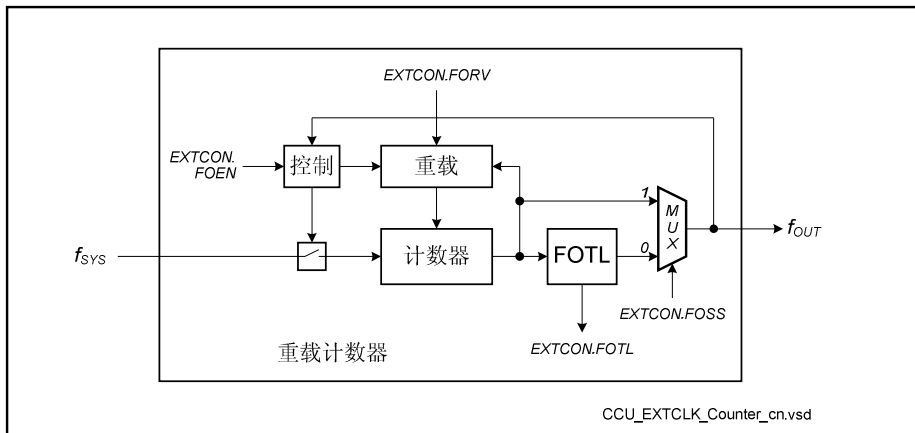
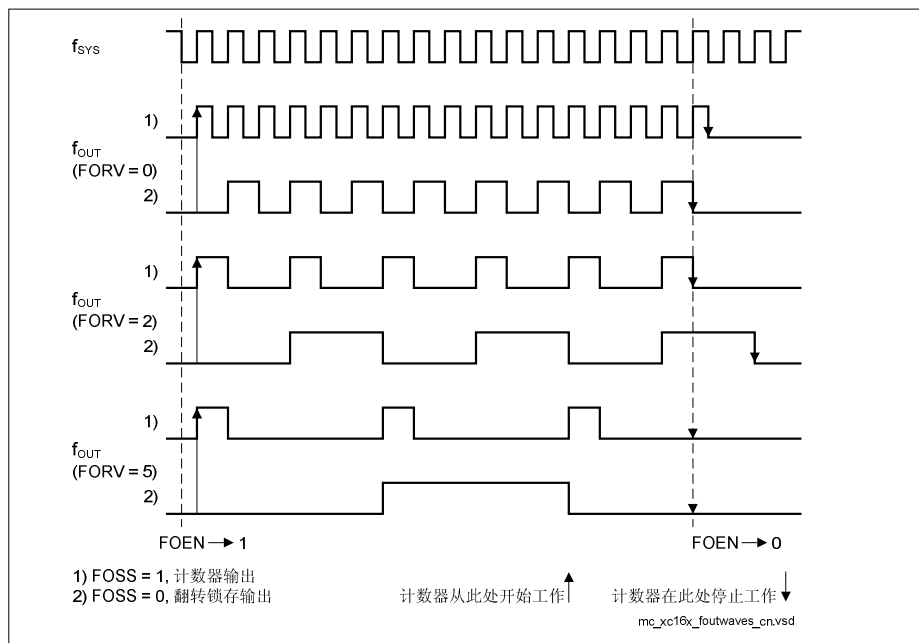


图 8-14 可编程频率输出的产生

$f_{OUT}$  始终提供完整的输出周期 (假定  $f_{SYS}$  可用) :

- 当启动  $f_{OUT}$  时 (EXTCON.FOEN 被置位), 由 EXTCN.FORV 加载计数器 FOCNT
- 当停止  $f_{OUT}$  时 (EXTCON.FOEN 被清零),  $f_{OUT}$  变为 (或等于) 0 时计数器 FOCNT 停止工作

寄存器 EXTCN 用于控制输出的产生 (频率、波形、激活) 并提供所有状态信息 (EXTCON.FOTL) 。



**图 8-15 输出波形示例**

注：若  $EXTCON.FORV > 0$ ，输出 ( $EXTCON.FOSS = 1$ ) 保持一个  $f_{SYS}$  周期的高电平。若  $EXTCON.FORV = 0$ ，输出等于  $f_{SYS}$ 。

若需要参考时钟（如用于总线接口），必须直接选择  $f_{SYS}$ 。

## 8.1.7 CGU 寄存器

### 8.1.7.1 唤醒时钟寄存器

该寄存器用于控制  $OSC\_WU$  的设置。

#### WUOSCCON

唤醒 OSC 控制寄存器

ESFR (F1AE<sub>H</sub>/D7<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
<b>FREQSEL</b>	[1:0]	rw	<b>频率选择</b> 不同设置对应的频率值请参见数据手册。 <i>注: <math>f_{WU}</math> 用作任何电路的时钟源时, 该频率值切勿被修改。</i>
<b>0</b>	[3:2]	rw	<b>保留</b> 设置该寄存器时不要修改该值。
<b>DIS</b>	4	w	<b>时钟禁用</b> 0 <sub>B</sub> 振荡器开启, 允许产生时钟。 1 <sub>B</sub> 振荡器关闭, 禁止产生时钟。
<b>0</b>	[15:5]	r	<b>保留</b> 读操作返回 0; 应写入 0。

### 8.1.7.2 高精度振荡器寄存器

该寄存器用于控制 OSC\_HP 的设置。

#### HPOSCCON

高精度 OSC 控制寄存器

ESFR (F1B4<sub>H</sub>/DA<sub>H</sub>)

复位值: 053C<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>OSC 2L0</b>	<b>OSC 2L1</b>	<b>EM FIN DIS EN</b>	<b>EM CLK EN</b>	<b>SH BY</b>	<b>X1D EN</b>	<b>X1D</b>	<b>GAINSEL</b>	<b>MODE</b>	<b>OSC WDT RST</b>	<b>PLL V</b>				
r	rh	rh	rw	rw	rw	rw	rh	rw	rw	rw	w	rh			

符号	位序号	类型	功能描述
<b>PLL V</b>	0	rh	<b>振荡器有效状态标志</b> 该位置时 OSC_HP 的频率输出是否可用。 这由 PLL 的振荡器看门狗进行检测。 0 <sub>B</sub> OSC_HP 频率不可用, 其频率低于极限值。 1 <sub>B</sub> OSC_HP 频率可用, 其频率不低于极限值。 具体内容参见 <a href="#">章节 8.1.4.8</a> 。



符号	位序号	类型	功能描述
<b>OSCWDTRST</b>	1	w	<b>振荡器看门狗复位</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> PLL 的振荡器看门狗被复位并重新启动。 <i>注：读取该位始终返回 0。</i>
<b>MODE</b>	[3:2]	rw	<b>振荡器模式</b> 00 <sub>B</sub> 振荡器正常工作（外部晶振模式） 01 <sub>B</sub> 保留，不使用 10 <sub>B</sub> 外部输入时钟模式：振荡器处于省电模式 11 <sub>B</sub> OSC_HP 被禁用，进入省电模式
<b>GAINSEL</b>	[5:4]	rw	<b>振荡器增益选择</b> 00 <sub>B</sub> 供电电流通常为 300 $\mu$ A（未测试） 01 <sub>B</sub> 供电电流通常为 530 $\mu$ A（未测试） 10 <sub>B</sub> 供电电流通常为 450 $\mu$ A（未测试） 11 <sub>B</sub> 供电电流通常为 610 $\mu$ A（未测试）
<b>X1D</b>	6	rh	<b>XTAL1 数据值</b> 该位给出引脚 XTAL1 的反相电平。X1DEN 被置位时，通过 f <sub>SYS</sub> 对该位采样。 <i>注：XTAL1 上的电压必须遵守数据手册规定的电压值。</i>
<b>X1DEN</b>	7	rw	<b>XTAL1 数据使能</b> 0 <sub>B</sub> 不更新位 X1D。 1 <sub>B</sub> 可更新位 X1D。
<b>SHBY</b>	8	rw	<b>整形器旁路</b> 整形器将输入信号格式化适当的信号。为了确保操作正确，该位必须为 0。 0 <sub>B</sub> 整形电路不被旁路。 1 <sub>B</sub> 整形电路被旁路。
<b>EMCLKEN</b>	9	rw	<b>OSCWDT 紧急系统时钟源选择使能</b> 出现 OSCWDT 紧急事件时，该位请求主控时钟复用器 (MCM) 切换至备选时钟（由

符号	位序号	类型	功能描述
			SYSCON0.EMCLKSEL 选择)。 0 <sub>B</sub> MCM 由 SYSCON0.CLKSEL 控制。 1 <sub>B</sub> MCM 由 SYSCON0.EMCLKSEL 控制。
<b>EMFINDISEN</b>	10	rw	<b>紧急输入时钟断开使能</b> 出现 OSCWDT 紧急事件时, 该位控制是否置位 PLLSTAT.FINDIS。 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 在紧急情况下置位 PLLSTAT.FINDIS。 <i>注: 关于如何正确处理紧急事件, 请参见«程序员指南»。</i>
<b>OSC2L1</b>	11	rh	<b>OSC_HP 不可用频率事件</b> 该粘着位指示自从 OSC2L1 被上次清零后 (置位 STATCLR1.OSC2L1CLR) 位 PLLV 是否已被清零。 0 <sub>B</sub> PLLV 未被修改。 1 <sub>B</sub> PLLV 至少已被清零一次。
<b>OSC2L0</b>	12	rh	<b>OSC_HP 可用频率事件</b> 该粘着位指示自从 OSC2L0 被上次清零后 (置位 STATCLR1.OSC2L0CLR) 位 PLLV 是否已被置位。 0 <sub>B</sub> PLLV 未被修改。 1 <sub>B</sub> PLLV 至少已被置位一次。
<b>0</b>	[15:13]	r	<b>保留</b> 读操作返回 0; 应写入 0。

### 8.1.7.3 均流控制时钟控制寄存器

该寄存器用于控制均流控制时钟源。

#### PLLOSCCON

##### PLL OSC 配置寄存器

##### ESFR (F1B6<sub>H</sub>/DB<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
OSCPD	0	rw	时钟源省电模式 0 <sub>B</sub> 均流控制时钟源工作。 1 <sub>B</sub> 均流控制时钟源关闭。
OSCTRIM	[9:1]	w	时钟源调整配置 该值用于调整均流控制时钟源的频率范围。 设置该寄存器时不修改该值。
0	[15:10]	r	保留 读操作返回 0；应写入 0。

### 8.1.7.4 PLL 寄存器

#### PLLSTAT

##### PLL 状态寄存器

##### ESFR (F0BC<sub>H</sub>/5E<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSCLOCK	OSCSTAB	0	REGSTAT	VCO L1	VCO L0	FIN DIS	K2 RDY	K1 RDY	N RDY	P RDY	0	VCO LOCK	OSC SEL ST	PWD STAT	VCO BYST
rh	rh	r	rh	rh	rh	rh	rh	rh	rh	rh	r	rh	rh	rh	rh

符号	位序号	类型	功能描述
<b>VCOPYST</b>	0	rh	<b>VCO 旁路状态</b> 0 <sub>B</sub> PLL 时钟来自分频器 K1 (预分频模式) 1 <sub>B</sub> PLL 时钟来自分频器 K2 (正常/未锁相模式) <i>注: PLLCON0.VCOBY 和 VCOBYST 的编码不同。</i>
<b>PWDSTAT</b>	1	rh	<b>PLL 省电模式状态</b> 0 <sub>B</sub> PLL 可正常工作。 1 <sub>B</sub> PLL 的数字部分被关闭。
<b>OSCSELST</b>	2	rh	<b>振荡器输入选择状态</b> 0 <sub>B</sub> 选择外部输入时钟源 ( $f_{IN}$ )。 1 <sub>B</sub> 选择内部输入时钟源。
<b>VCOLOCK</b>	3	rh	<b>PLL VCO 锁定状态</b> 0 <sub>B</sub> $f_{REF}$ 和 $f_{DIV}$ 的频差大于允许值。PLL 无法锁相。 1 <sub>B</sub> PLL 时钟 $f_{PLL}$ 锁定 $f_{REF}$ , 稳定工作。 <i>注: 在失锁的情况下, 若参考频率高于/低于可能的锁相频率, VCO 频率 <math>f_{VCO}</math> 接近所选 VCO 频带的上/下边界值。</i>
<b>PRDY</b>	5	rh	<b>P 分频器准备就绪状态</b> 0 <sub>B</sub> 位域 PLLCON1.PDIV 已修改, 还未使用新的 P 值。 1 <sub>B</sub> P 分频器以位域 PLLCON1.PDIV 定义的分频值工作。
<b>NRDY</b>	6	rh	<b>N 分频器准备就绪状态</b> 0 <sub>B</sub> 位域 PLLCON0.NDIV 已修改, 还未使用新的 N 值。 1 <sub>B</sub> N 分频器以位域 PLLCON0.NDIV 定义的分频值工作。
<b>K1RDY</b>	7	rh	<b>K1 分频器准备就绪状态</b> 0 <sub>B</sub> 位域 PLLCON2.K1DIV 已修改, 还未使用新的 K1 值。 1 <sub>B</sub> K1 分频器以位域 PLLCON2.K1DIV 定义的分频值工作。

符号	位序号	类型	功能描述
<b>K2RDY</b>	8	rh	<b>K2 分频器准备就绪状态</b> 0 <sub>B</sub> 位域 PLLCON3.K2DIV 已修改，还未使用新的 K2 值。 1 <sub>B</sub> K1 分频器以位域 PLLCON2.K1DIV 定义的分频值工作。
<b>FINDIS</b>	9	rh	<b>输入时钟断开选择状态</b> 0 <sub>B</sub> VCO 和参考时钟相连。 1 <sub>B</sub> VCO 和参考时钟断开。 <i>注：可向寄存器 STATCLR1 中的位 SETFINDIS 或 CLRFINDIS 写 1 软件控制该位。</i>
<b>VCOL0</b>	10	rh	<b>VCO 锁定检测失锁状态</b> 该粘着位指示自从 VCOL0 被上次清零后（置位 STATCLR1.VCOL0CLR）位 VCOLOCK 是否已被清零。 0 <sub>B</sub> 未检测到下降沿。 1 <sub>B</sub> VCOLOCK 至少已被清零一次（VCO 失锁）。
<b>VCOL1</b>	11	rh	<b>VCO 锁定检测锁相状态</b> 该粘着位指示自从 VCOL1 被上次清零后（置位 STATCLR1.VCOL1CLR）位 VCOLOCK 是否已被置位。 0 <sub>B</sub> 未检测到上升沿。 1 <sub>B</sub> 已达到 VCO 锁相。
<b>REGSTAT</b>	12	rh	<b>PLL 电源调节器状态</b> PLL 由一个独立的内部调节器供电。 0 <sub>B</sub> PLL 未供电（关闭）。 1 <sub>B</sub> PLL 被供电（可工作）。 <i>注：可向寄存器 PLLCON0 中的位 REGENSET 或 REGENCLR 写 1 软件控制该位。</i>
<b>OSCSTAB</b>	14	rh	<b>OSC_HP 稳定</b> 0 <sub>B</sub> 振荡器正在启动。未开始计数或未计数到 2 <sup>15</sup> 个时钟周期。

系统控制单元 (SCU)

符号	位序号	类型	功能描述
			<p>1<sub>B</sub> 已至少计数 <math>2^{15}</math> 个时钟周期。</p> <p>注: HPOSCCON.MODE 为 1X<sub>B</sub> 时, 该位被清零。</p>
OSCLOCK	15	rh	<p><b>OSC_HP 锁定</b></p> <p>0<sub>B</sub> 振荡器未被锁定。未开始计数或未计数到 <math>2^{11}</math> 个时钟周期。</p> <p>1<sub>B</sub> 振荡器被锁定。已至少计数 <math>2^{11}</math> 个时钟周期。</p> <p>注: HPOSCCON.MODE 为 1X<sub>B</sub> 时, 该位被清零。</p>
0	4,13	r	<p><b>保留</b></p> <p>读操作返回 0; 应写入 0。</p>

STATCLR1

PLL 状态清除 1 寄存器

ESFR (F0E2<sub>H</sub>/71<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										CLR FIN DIS	SET FIN DIS	OSC 2L0 CLR	OSC 2L1 CLR	OSC L1 CLR	OSC L0 CLR
r										w	w	w	w	w	w

符号	位序号	类型	功能描述
VCOL0CLR	0	w	<p><b>VCOL0 清零触发</b></p> <p>0<sub>B</sub> 无操作。</p> <p>1<sub>B</sub> 清除位 PLLSTAT.VCOL0。</p>
VCOL1CLR	1	w	<p><b>VCOL1 清零触发</b></p> <p>0<sub>B</sub> 无操作。</p> <p>1<sub>B</sub> 清除位 PLLSTAT.VCOL1。</p>
OSC2L1CLR	2	w	<p><b>OSC2L1 清零触发</b></p> <p>0<sub>B</sub> 无操作。</p> <p>1<sub>B</sub> 清除位 HPOSCCON.OSC2L1。</p>
OSC2L0CLR	3	w	<p><b>OSC2L0 清零触发</b></p> <p>0<sub>B</sub> 无操作。</p>

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
			1 <sub>B</sub> 清除位 HPOOSCCON.OSC2L0。
<b>SETFINDIS</b>	4	w	<b>置位状态位 PLLSTAT.FINDIS</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 PLLSTAT.FINDIS。断开 VCO 输入时钟。
<b>CLRFINDIS</b>	5	w	<b>清除状态位 PLLSTAT.FINDIS</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除 PLLSTAT.FINDIS。连接 VCO 输入时钟。
<b>0</b>	4,13	r	<b>保留</b> 读操作返回 0；应写入 0。

注：读写类型为 *w* 的这些位读取始终返回 0。

以下这些寄存器控制 PLL 的配置。

**PLLCON0**

**PLL 配置 0 寄存器**

**ESFR (F1B8<sub>H</sub>/DC<sub>H</sub>)**

**复位值: 1302<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>N ACK</b>	<b>0</b>	<b>NDIV</b>				<b>0</b>	<b>IN SEL</b>	<b>REG EN SET</b>	<b>REG EN CLR</b>	<b>VCOSEL</b>	<b>VCO PWD</b>	<b>VCO BY</b>			
rw	r	rw				r	rw	w	w	rw	rw	rw			

符号	位序号	类型	功能描述
<b>VCOLBY</b>	0	rw	<b>VCO 旁路</b> 0 <sub>B</sub> PLL 时钟来自分频器 K2（正常/未锁相模式） 1 <sub>B</sub> PLL 时钟来自分频器 K1（预分频模式，即 VCO 旁路） 位 PLLSTAT.VCOBYST 指示实际选用的分频器。 注：VCOBY 和 PLLSTAT.VCOBYST 的编码不同。
<b>VCOPWD</b>	1	rw	<b>VCO 省电模式</b> 0 <sub>B</sub> 正常工作。 1 <sub>B</sub> VCO 进入省电模式，不能再使用。若之前选

符号	位序号	类型	功能描述
			择预分频模式，只有该模式有效。
<b>VCOSSEL</b>	[3:2]	rw	<b>VCO 范围选择</b> 不同设置对应的 VCO 工作范围列于数据手册中。
<b>REGENCLR</b>	4	w	<b>电源调节器使能清零</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 关闭 PLL 功率调节器。PLL 不被供电（不能工作）。 <i>注：读取该位始终返回 0。</i>
<b>REGENSET</b>	5	w	<b>电源调节器使能置位</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 开启 PLL 功率调节器。PLL 被供电（可工作）。 <i>注：读取该位始终返回 0。</i>
<b>INSEL</b>	6	rw	<b>输入选择</b> 0 <sub>B</sub> f <sub>OSC</sub> 被选作 PLL 的输入。 1 <sub>B</sub> f <sub>CLKIN1</sub> 被选作 PLL 的输入。
<b>NDIV</b>	[13:8]	rw	<b>N 分频因子</b> N 分频器的分频因子为 NDIV+1。 若 VCOSSEL = 00 <sub>B</sub> ，只允许选用 N = 8 到 N = 28 之间的值。 若 VCOSSEL = 01 <sub>B</sub> ，只允许选用 N = 16 到 N = 40 之间的值。 若超出该范围，无法确保 VCO 稳定工作。
<b>NACK</b>	15	rw	<b>N 分频器准备就绪应答</b> 置位该位将对 NRDY 作出应答。
<b>0</b>	7,14	r	<b>保留</b> 读操作返回 0；应写入 0。



## PLLCON1

### PLL 配置 1 寄存器

### ESFR (F1BA<sub>H</sub>/DD<sub>H</sub>)

复位值: 000A<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P ACK	0	PDIV				0	EM FIN DIS EN	EM CLK EN	0	A OSC SEL	RES LD	OSC SEL	PLL PWD		
rw	r	rw				r	rw	rw	r	rw	w	rw	rw		

符号	位序号	类型	功能描述
PLLPWD	0	rw	<b>PLL 省电模式</b> 0 <sub>B</sub> 正常工作。 1 <sub>B</sub> 整个 PLL 模块进入省电模式，停止工作。
OSCSEL	1	rw	<b>振荡器输入选择</b> 0 <sub>B</sub> 选择外部时钟作为输入。 1 <sub>B</sub> 选择均流控制时钟作为输入。
RESLD	2	w	<b>重新启动 VCO 锁定检测</b> 置位该位将复位 PLLSTAT.VCOLOCK 并重新启动 VCO 锁定检测。 <i>注：读取该位始终返回 0。</i>
AOSCSEL	3	rw	<b>异步振荡器输入选择</b> 该位设置优先级高于 OSCSEL 设置。 0 <sub>B</sub> 由 OSCSEL 控制输入配置。 1 <sub>B</sub> 异步选择均流控制的时钟作为 PLL 输入。
EMCLKEN	5	rw	<b>VCOLCK 紧急系统时钟源选择使能</b> 出现 VCOLCK 紧急事件时，该位请求主控时钟复用器 (MCM) 切换至备选时钟 (由 SYSCON0.EMCLKSEL 选择)。 0 <sub>B</sub> MCM 由 SYSCON0.CLKSEL 控制。 1 <sub>B</sub> MCM 由 SYSCON0.EMCLKSEL 控制。
EMFINDISEN	6	rw	<b>紧急输入时钟断开使能</b> 出现 VCOLCK 紧急事件时，该位控制是否置位 PLLSTAT.FINDIS。

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
			<b>0<sub>B</sub></b> 无操作。 <b>1<sub>B</sub></b> 发生 VCOLCK 紧急事件时置位 PLLSTAT.FINDIS。 <i>注：关于如何正确处理紧急事件，请参见“程序员指南”。</i>
<b>PDIV</b>	[11:8]	rw	<b>P 分频因子</b> P 分频器的分频因子为 PDIV+1。
<b>PACK</b>	15	rw	<b>P 分频器准备就绪应答</b> 置位该位将对 PRDY 作出应答。
<b>0</b>	4,7, [14:12]	r	<b>保留</b> 读操作返回 0；应写入 0。

**PLLCON2**

**PLL 配置 2 寄存器**

**ESFR (F1BC<sub>H</sub>/DE<sub>H</sub>)**

复位值: 0001<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>K1 ACK</b>	<b>0</b>					<b>K1DIV</b>									
rw	r					rw									

符号	位序号	类型	功能描述
<b>K1DIV</b>	[9:0]	rw	<b>K1 分频因子</b> K1 分频器的分频因子为 K1DIV+1。
<b>K1ACK</b>	15	rw	<b>K1 分频器准备就绪应答<sup>1)</sup></b> 置位该位将对 K1RDY 作出应答。
<b>0</b>	[14:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

1) 关于如何正确处理应答，请参见“程序员指南”。

## PLLCON3

### PLL 配置 3 寄存器

ESFR (F1BE<sub>H</sub>/DF<sub>H</sub>)

复位值: 00CB<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
K2 ACK	0					K2DIV									
rw	r					rw									

符号	位序号	类型	功能描述
K2DIV	[9:0]	rw	K2 分频因子 K2 分频器的分频因子为 K2DIV+1。
K2ACK	15	rw	K2 分频器准备就绪应答 <sup>1)</sup> 置位该位将对 K2RDY 作出应答。
0	[14:10]	r	保留 读操作返回 0；应写入 0。

1) 关于如何正确处理应答，请参见«程序员指南»。

### 8.1.7.5 系统时钟控制寄存器

以下这些寄存器用于控制系统级的时钟行为。

## SYSCON0

### 系统控制 0 寄存器

SFR (FF4A<sub>H</sub>/A5<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SEL STAT	0	EMS VCO	EMS OSC	STMCLKSEL			WUT CLKSEL		EM CLK SEL EN	0	EMCLKSEL		0	CLKSEL	
rh	r	rh	rh	rw			rw		rw	r	rw		r	rw	

符号	位序号	类型	功能描述
CLKSEL	[1:0]	rw	时钟选择 该位域定义正常工作时用作系统时钟的时钟源。 00 <sub>B</sub> 使用唤醒时钟 f <sub>WU</sub> 。 01 <sub>B</sub> 使用振荡器时钟 (OSC_HP) f <sub>OSC</sub> 。

符号	位序号	类型	功能描述
			<p>10<sub>B</sub> 使用 PLL 时钟 <math>f_{PLL}</math>。</p> <p>11<sub>B</sub> 使用 CLKIN1 作为直接输入时钟 <math>f_{CLKIN1}</math>。</p>
<b>EMCLKSEL</b>	[4:3]	rw	<p><b>紧急时钟选择</b></p> <p>该位域定义产生 OSCWDT 或 VCOLCK 紧急事件时用作系统时钟的时钟源。</p> <p>00<sub>B</sub> 使用唤醒时钟 <math>f_{WU}</math>。</p> <p>01<sub>B</sub> 使用振荡器时钟 (OSC_HP) <math>f_{OSC}</math>。</p> <p>10<sub>B</sub> 使用 PLL 时钟 <math>f_{PLL}</math>。</p> <p>11<sub>B</sub> 使用 CLKIN1 作为直接输入时钟 <math>f_{CLKIN1}</math>。</p>
<b>EMCLKSELEN</b>	6	rw	<p><b>紧急时钟选择使能</b></p> <p>出现 OSCWDT 或 VCOLCK 紧急事件时，该位控制将系统时钟切换至备选时钟。</p> <p>0<sub>B</sub> 禁止切换。</p> <p>1<sub>B</sub> 允许切换。</p>
<b>WUTCLKSEL</b>	[8:7]	rw	<p><b>WUT 时钟选择</b></p> <p>该位域定义用作唤醒定时器时钟的时钟源。</p> <p>00<sub>B</sub> 使用唤醒时钟 <math>f_{WU}</math>。</p> <p>01<sub>B</sub> 使用振荡器时钟 (OSC_HP) <math>f_{OSC}</math>。</p> <p>10<sub>B</sub> 使用 PLL 时钟 <math>f_{PLL}</math>。</p> <p>11<sub>B</sub> 使用 CLKIN1 作为直接输入时钟 <math>f_{CLKIN1}</math>。</p>
<b>STMCLKSEL</b>	[11:9]	rw	<p><b>STM 时钟选择</b></p> <p>该位域定义用作 STM 时钟的时钟源。</p> <p>000<sub>B</sub> 使用 CLKIN2 作为直接输入时钟 <math>f_{CLKIN2}</math>。</p> <p>001<sub>B</sub> 使用振荡器时钟 (OSC_HP) <math>f_{OSC}</math>。</p> <p>010<sub>B</sub> 使用 PLL 时钟 <math>f_{PLL}</math>。</p> <p>011<sub>B</sub> 使用 CLKIN1 作为直接输入时钟 <math>f_{CLKIN1}</math>。</p> <p>100<sub>B</sub> 使用唤醒时钟 <math>f_{WU}</math>。</p> <p>101<sub>B</sub> 保留，不使用该组合。</p> <p>110<sub>B</sub> 保留，不使用该组合。</p> <p>111<sub>B</sub> 保留，不使用该组合。</p>

符号	位序号	类型	功能描述
<b>EMSOSC</b>	12	rh	<b>OSCWDT 紧急事件源的状态</b> 0 <sub>B</sub> 自从 EMSOSC 被上次清零后, 未发生 OSCWDT 紧急事件。 1 <sub>B</sub> 已发生 OSCWDT 紧急事件。 <i>注: 只有当 EMCLKSELEN 置位时该位才置位。</i>
<b>EMSVCO</b>	13	rh	<b>VCOLCK 紧急事件源的状态</b> 0 <sub>B</sub> 自从 EMSVCO 被上次清零后, 未发生 VCOLCK 紧急事件。 1 <sub>B</sub> 已发生 VCOLCK 紧急事件。 <i>注: 只有当 EMCLKSELEN 置位时该位才置位。</i>
<b>SELSTAT</b>	15	rh	<b>时钟选择状态</b> 0 <sub>B</sub> 当前使用位域 CLKSEL 的标准配置。 1 <sub>B</sub> 当前使用位域 EMCLKSEL 的配置。
<b>0</b>	2, 5, 14	r	<b>保留</b> 读操作返回 0; 应写入 0。

## STATCLR0

状态清除 0 寄存器

**ESFR (F0E0<sub>H</sub>/70<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	EMC VCO	EMC OSC	0												
r	w	w	r												

符号	位序号	类型	功能描述
<b>EMCOSC</b>	12	w	<b>EMSOSC 清零触发</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 SYSCON0.EMSOSC。 <i>注: 读取该位始终返回 0。</i>
<b>EMCVCO</b>	13	w	<b>EMSVCO 清零触发</b> 0 <sub>B</sub> 无操作。

符号	位序号	类型	功能描述
			1 <sub>B</sub> 清除位 SYSCON0.EMSVCO。 注：读取该位始终返回 0。
<b>0</b>	[11:0], [15:14]	r	保留 读操作返回 0；应写入 0。

### 8.1.7.6 RTC 时钟控制寄存器

注：只有当 RTC 关闭时才能修改寄存器 RTCCLKCON。

#### RTCCLKCON

RTC 时钟控制寄存器

SFR (FF4E<sub>H</sub>/A7<sub>H</sub>)

复位值: 0006<sub>H</sub>

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

符号	位序号	类型	功能描述
<b>RTCCLKSEL</b>	[1:0]	rw	<b>RTC 时钟选择</b> 该位域定义 RTC 的计数时钟源。 00 <sub>B</sub> 使用 PLL 时钟 f <sub>PLL</sub> 。 01 <sub>B</sub> 使用振荡器时钟 (OSC_HP) f <sub>OSC</sub> 。 10 <sub>B</sub> 使用唤醒时钟 f <sub>WU</sub> 。 11 <sub>B</sub> 使用 CLKIN2 作为直接输入时钟 f <sub>CLKIN2</sub> 。
<b>RTCCM</b>	2	rw	<b>RTC 时钟模式</b> 0 <sub>B</sub> 异步模式： RTC 以时钟 f <sub>RTC</sub> 工作。不能访问寄存器。 1 <sub>B</sub> 同步模式： RTC 以时钟 f <sub>SYS</sub> 工作。可读写寄存器。
<b>0</b>	[15:3]	r	保留 读操作返回 0；应写入 0。

### 8.1.7.7 外部时钟控制寄存器

该寄存器用于控制外部时钟（引脚 P2.8 和 7.1）的设置。

#### EXTCON

外部时钟控制寄存器

SFR (FF5E<sub>H</sub>/AF<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FOEN	FOSS	FORV					0	FOTL	0	SEL				EN	
rw	rw	rw					r	rh	r	rw				rw	

符号	位序号	类型	功能描述
EN	0	rw	<b>外部时钟使能</b> 0 <sub>B</sub> 不提供外部时钟信号。信号恒为 0。 1 <sub>B</sub> 外部时钟信号用作复用输出功能。
SEL	[4:1]	rw	<b>外部时钟选择</b> 选择将被送至 EXTCLK 引脚的时钟信号。 0000 <sub>B</sub> 系统时钟 f <sub>SYS</sub> 。 0001 <sub>B</sub> 可编程时钟信号 f <sub>OUT</sub> 。 0010 <sub>B</sub> PLL 输出时钟 f <sub>PLL</sub> 。 0011 <sub>B</sub> 振荡器时钟 f <sub>OSC</sub> 。 0100 <sub>B</sub> 唤醒时钟 f <sub>WU</sub> 。 0101 <sub>B</sub> 直接输入时钟 f <sub>CLKIN1</sub> 。 1000 <sub>B</sub> RTC 计数时钟 f <sub>RTC</sub> 。 所有其它组合保留不用。
FOTL	6	rh	<b>频率输出翻转锁存</b> FOCNT 每次下溢时发生翻转。
FORV	[13:8]	rw	<b>频率输出重载值</b> FOCNT 每次下溢时复制到 FOCNT 中。
FOSS	14	rw	<b>频率输出信号选择</b> 0 <sub>B</sub> 翻转锁存输出。 1 <sub>B</sub> 重载计数器输出：占空比和 FORV 有关。

符号	位序号	类型	功能描述
FOEN	15	rw	<b>频率输出使能</b> 0 <sub>B</sub> $f_{OUT}$ 为/变为低电平时, 停止产生频率输出。 1 <sub>B</sub> FOCNT 正在工作, $f_{OUT}$ 送至引脚。发生 0-1 跳变时进行首次重载。
0	5,7	r	<b>保留</b> 读操作返回 0; 应写入 0。

## 8.2 系统定时器功能 (STM)

系统定时器为器件提供了实时计数功能。

有关 STM 工作时钟的描述请参见[系统定时器 \(STM\) 时钟产生](#)。

STM 包含一个 16 位计数器, 该计数器可最多产生两个中断。由时钟源驱动的计数器可用于计数定时事件, 或者计数由系统时钟以外的其它时钟源触发的定时中断。很容易使用软件基于这些中断实现时钟功能。

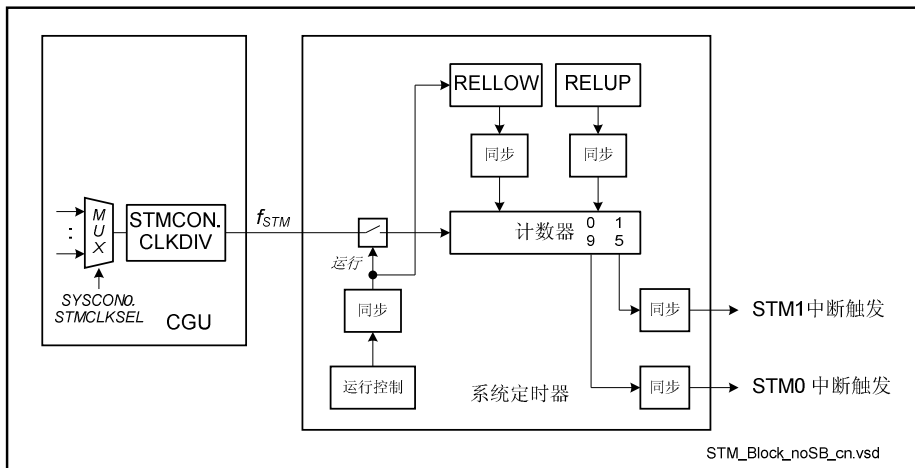


图 8-16 STM 框图



## 8.2.1 STM 寄存器

### 8.2.1.1 寄存器 STMREL

该寄存器用于定义 STM 的重载值（从而得到 STM 的周期值）。

#### STMREL

##### STM 重载寄存器

ESFR (F1A8<sub>H</sub>/D4<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
RELOW	[9:0]	Rw	<b>重载低位值</b> 计数器递增计数，位 9 从 1 <sub>B</sub> 变 0 <sub>B</sub> 时发送中断请求，此时向计数器加载该位域定义的重载值。
RELUP	[15:10]	rw	<b>重载高位值</b> 计数器递增计数，位 15 从 1 <sub>B</sub> 变 0 <sub>B</sub> 时发送中断请求，此时向计数器加载该位域和 RELOW 位域定义的重载值。

### 8.2.1.2 寄存器 STMCON

该寄存器存放 STM 的状态位和控制位。

#### STMCON

##### STM 控制寄存器

ESFR (F1AA<sub>H</sub>/D5<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
RUN	0	rw	<b>运行控制</b> 0 <sub>B</sub> STM 关闭。

符号	位序号	类型	功能描述
			1 <sub>B</sub> STM 工作。 该位置位后，STM 开始工作，向计数器载入重载值 STMREL.REL。
CLKDIV	[4:1]	rw	<b>STM 时钟分频</b> 该位域定义 STM 时钟输入的分频因子。所选时钟被 $2^{<CLKDIV>}$ 分频。
0	[15:5]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 8.3 唤醒定时器（WUT）

唤醒定时器用于在规定的周期之后触发系统功能。

有关唤醒定时器工作时钟的描述请参见[唤醒定时器（WUT）时钟产生](#)。

唤醒定时器时钟  $f_{WUT}$  驱动一个简单的计数器。所有功能由寄存器 WUCR 控制。

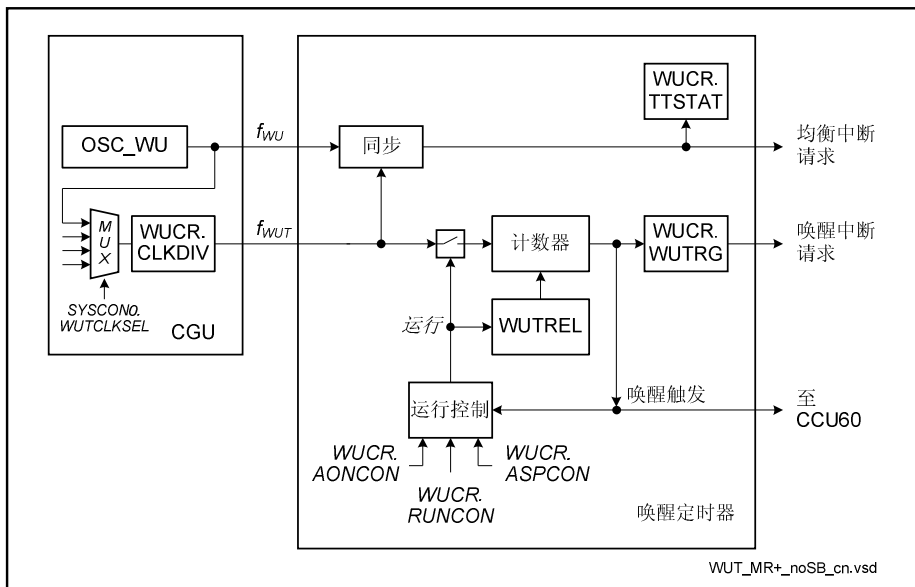


图 8-17 唤醒定时器逻辑框图

### 8.3.1 唤醒定时器操作

由运行控制逻辑控制启动和终止唤醒定时器工作。可通过以下方式启动定时器：

- 置位 WUCR.RUN

定时器开始工作时预分频器复位，计数器开始递减计数。

唤醒间隔计数器由时钟  $f_{WUT}$  控制，递减计数到 0，随后产生唤醒触发事件并置位 WUCR.WUTRG。

可通过以下方式终止定时器工作：

- 清除 WUCR.RUN
- 置位 WUCR.ASP 且产生唤醒触发事件

计数器若计数到 0 后不停止工作，它将继续从 WUTREL 递减计数。

#### 决定唤醒周期

为了调整被计数的时钟周期数（写入计数器的值）从而确定被唤醒的间隔时间，在进入省电模式之前可测量均流控制的唤醒时钟（OSC\_WU）的实际频率。可通过评估（被同步的）触发事件（该事件可产生中断请求）来测量 OSC\_WU 的周期，或由位 WUCR.TTSTAT 进行监控。

由于使用中断同时包含软件会有一些不确定性，可通过另一种方式确定唤醒周期：将 WUT 产生的唤醒触发事件送至 CCU60 单元，从而和正确的系统时钟进行比较。

### 8.3.2 WUT 寄存器

#### 8.3.2.1 寄存器 WUTREL

该寄存器用于配置计数器的重载值。

#### WUTREL

唤醒定时器重载寄存器										ESFR (F0B0 <sub>H</sub> /58 <sub>H</sub> )					复位值: FFFF <sub>H</sub>				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
RELVAL																			

rw

符号	位序号	类型	功能描述
RELVAL	[15:0]	rw	<b>唤醒定时器重载值</b> 定时器开始工作时，WUT 计数器加载该值并开始递减计数。

### 8.3.2.2 寄存器 WUCR

该寄存器存放 WUT 的状态位和控制位。

#### WUCR

唤醒控制寄存器

ESFR (F1B0<sub>H</sub>/D8<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
WU TRG	TTS TAT	CLKDIV	ASP	AON	RUN	CLR TRG	0	ASPCON	AONCON	RUNCON					
rh	rh	rw	rh	rh	rh	w	r	w	w	w					

符号	位序号	类型	功能描述
<b>RUNCON</b>	[1:0]	w	<b>RUN 控制</b> 00 <sub>B</sub> 无操作。 01 <sub>B</sub> 置位 RUN。 10 <sub>B</sub> 清除 RUN。 11 <sub>B</sub> 保留, 不使用该组合。 <i>注: 读取该位始终返回 0。</i>
<b>AONCON</b>	[3:2]	w	<b>AON 控制</b> 00 <sub>B</sub> 无操作。 01 <sub>B</sub> 保留, 不使用该组合。 10 <sub>B</sub> 清除 AON。 11 <sub>B</sub> 保留, 不使用该组合。 <i>注: 读取该位始终返回 0。</i>
<b>ASPCON</b>	[5:4]	w	<b>ASP 控制</b> 00 <sub>B</sub> 无操作。 01 <sub>B</sub> 置位 ASP。 10 <sub>B</sub> 清除 ASP。 11 <sub>B</sub> 保留, 不使用该组合。 <i>注: 读取该位始终返回 0。</i>
<b>CLRTRG</b>	7	w	<b>清除 WUTRG</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除 WUTRG。

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
			<i>注：读取该位始终返回 0。</i>
<b>RUN</b>	8	rh	<b>运行指示位</b> 0 <sub>B</sub> 唤醒计数器停止工作。 1 <sub>B</sub> 唤醒计数器正递减计数。 <i>注：若通过设置 RUNCON 清除该位，在 4 个 <math>f_{WUT}</math> 周期之后 WUT 停止工作。</i>
<b>AON</b>	9	rh	<b>自动启动指示位</b> 0 <sub>B</sub> 唤醒计数器只能由软件启动工作。 1 <sub>B</sub> 保留，不使用该组合。 <i>注：通过设置 AONCON = 10<sub>B</sub> 清除该位。</i>
<b>ASP</b>	10	rh	<b>自动终止指示位</b> 0 <sub>B</sub> 唤醒计数器连续工作。 1 <sub>B</sub> 唤醒计数器计数到 0、产生触发事件后停止工作。
<b>CLKDIV</b>	[13:11]	rw	<b>WUT 时钟分频</b> 该位域定义 WUT 时钟输入的分频因子。所选时钟被 $2^{<CLKDIV>}$ 分频。
<b>TTSTAT</b>	14	rh	<b>均衡触发事件状态</b> 0 <sub>B</sub> 均衡触发事件无效。不产生均衡触发中断。 1 <sub>B</sub> 均衡触发事件有效。产生均衡触发中断。 <i>注：若配置寄存器 SYSCON0.WUTCLKSEL 使 <math>f_{WUT} = f_{WU}</math>，该位无效。</i>
<b>WUTRG</b>	15	rh	<b>WUT 触发指示位</b> 0 <sub>B</sub> 自从 WUTRG 被上次清零后，未发生唤醒触发事件，不产生唤醒中断。 1 <sub>B</sub> 已发生唤醒触发事件，产生唤醒中断。
<b>0</b>	6	r	<b>保留</b> 读操作返回 0；应写入 0。

*注：寄存器 WUCR 的高位字节指示唤醒计数器逻辑的当前状态。写操作对它们没有影响，但它们受控于相关的控制位域（低位字节）或硬件。*

寄存器 **WUCR** 低位字节中的控制位 (域) 决定了唤醒计数器逻辑状态位 (高位字节) 的状态。软件置位触发相应的操作, 置 0 对它们没有影响。

## **8.4 复位操作**

由复位控制模块产生所有复位类型。它处理复位触发控制、复位长度和复位时序。复位将系统或部分系统 (这取决于复位类型) 初始化到设定的状态。

### **8.4.1 复位架构**

XE16xyM 中包含一个成熟的复位架构以最大程度支持各种不同的应用场合。该复位架构支持不同的电源域。

系统支持多种复位类型。

#### **8.4.1.1 器件复位等级**

根据电源域的不同 (见 [章节 8.6](#)) 将器件复位划分为以下等级:

级别 1: I/O 域 (电源域 DMP\_B)

级别 2: 系统电源域 DMP\_M

级别 3: 系统电源域 DMP\_1

若某个电源域 (复位级别) 失效, 则该电源域以及更低等级电源域的所有复位均被激活。

#### **8.4.1.2 复位类型**

XE16xyM 支持以下复位类型。

##### **电源复位**

- 上电复位

该类复位使整个系统进入设定的状态。只应通过上电事件请求该类复位。

- 电源域 DMP\_M 和 DMP\_1 的电源复位

DMP\_M 或 DMP\_1 出现电源故障时, 利用该类复位恢复数据的一致性。

##### **功能/用户复位**

- 调试复位

该类复位使整个调试系统进入设定的状态。

- 内部应用复位

## 系统控制单元 (SCU)

该类复位使包括以下部分的整个应用系统进入设定的状态：所有外设（RTC 除外）、CPU、部分 SCU 和 flash 存储器。

- 应用复位

该类复位使包括以下部分的整个应用系统进入设定的状态：所有外设（端口和 RTC 除外）、CPU、部分 SCU 和 flash 存储器。

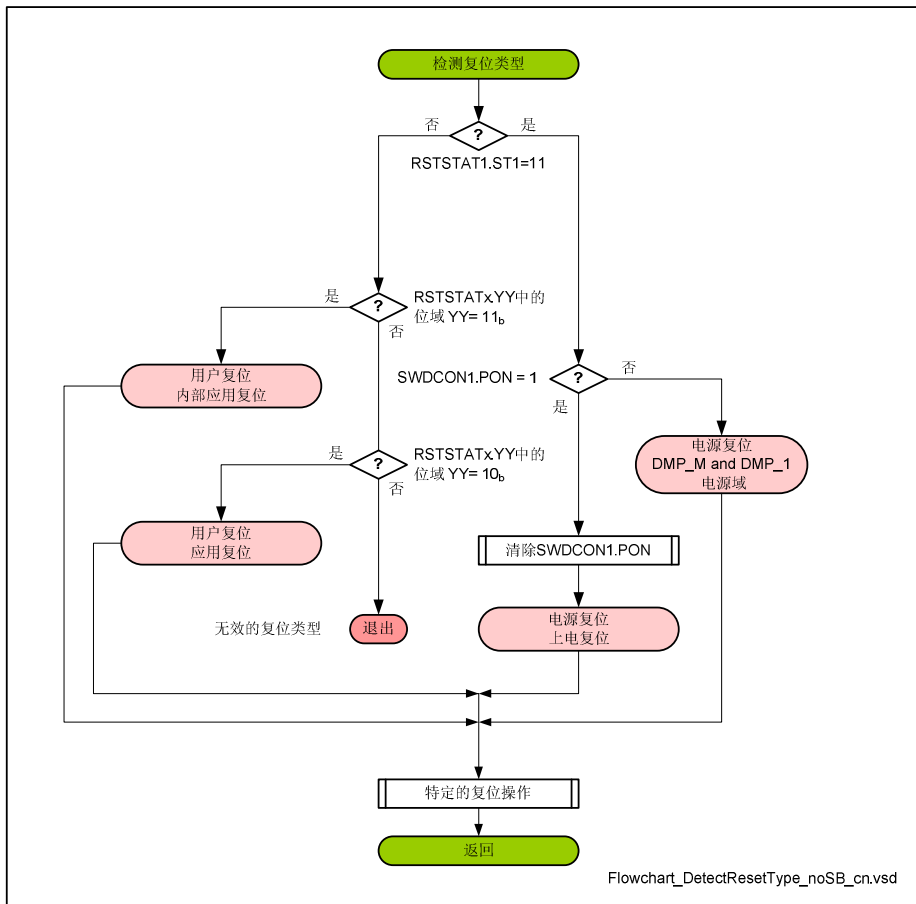
复位之后，复位状态寄存器 RSTSTATx 指示最近一次复位的类型。

可根据表 8-1 评估寄存器 SWDCON1 的值，从而识别最近一次复位的复位类型和复位请求源。最近一次发生的复位始终具有最高复位等级。若寄存器的值指示两个复位请求源请求同一种复位，即意味着这两个复位请求源同时有效。若寄存器的值指示有多个复位请求源请求不同类型的复位，则最高等级的复位生效。

表 8-1 复位识别

复位类型（按等级顺序，从高到低）	复位类型识别
上电复位	SWDCON1.PON = 1 <sub>B</sub> RSTSTAT1.STM = 11 <sub>B</sub> RSTSTAT1.ST1 = 11 <sub>B</sub> 进一步的操作：清除 PON 以能够识别电源域 DMP_M 或 DMP_1 的电源复位
电源域 DMP_M 或 DMP_1 的电源复位	SWDCON1.PON = 0 <sub>B</sub> （清除后不改变） RSTSTAT1.STM = 11 <sub>B</sub> RSTSTAT1.ST1 = 11 <sub>B</sub>
内部应用复位	RSTSTAT1.ST1 = 00 <sub>B</sub> RSTSTATx.y 中的任意位域 y = 10 <sub>B</sub>
应用复位	RSTSTAT1.ST1 = 00 <sub>B</sub> RSTSTATx.y 中的任意位域 y = 11 <sub>B</sub> （RSTSTA1.ST1 和 RSTSTA1.STM 除外）

图 8-18 所示的算法表明如何根据表 8-1 所列出的条件检测各种复位类型。



**图 8-18 复位类型检测算法**

### 8.4.2 一般复位操作

若复位请求触发源有效，则产生复位。大多数复位请求源可配置选择触发产生何种复位，也可配置为无操作（不产生复位），通过向复位配置寄存器中的相应位写入 **00<sub>B</sub>** 来实现。调试复位只能由专用的复位请求触发源请求、不能通过复位配置寄存器选择。详见寄存器 **RSTCON0** 和 **RSTCON1** 的描述。

复位长度由两个独立的计数器定义：一个用于系统和应用复位；另一个用于调试复位。用于调试复位的计数器可根据调试器的需求调整复位长度、无需修改应用设置。



#### 8.4.2.1 复位计数器 (RSTCNTA 和 RSTCNTD)

复位计数器 RSTCNTA 用于控制所有和应用相关的复位类型（内部应用复位和应用复位）的复位长度。复位计数器 RSTCNTD 用于控制调试复位的复位长度。

复位计数器控制内部复位的长度，这可用于通过引脚 ESRx 配置复位输出的长度，从而和连接到这些信号的外围模块的复位输入需求相匹配。

复位计数器 RSTCNT 为 8 位计数器，从 RSTCNTCON.RELx (x = A 或 D) 定义的重载值递减计数。一旦复位请求触发条件生效（详见表 8-2 和表 8-3），复位控制模块即启动计数器工作。是否必须启动计数器工作取决于复位请求触发源的类型以及计数器是否已工作。若计数器未开始工作（未递减计数），则始终启动其开始计数；若计数器已开始工作，则根据新复位请求源触发产生的复位类型决定是否重新开始计数。具体操作归纳见表 8-2 和表 8-3。

表 8-2 RSTCNTA 重新开始计数

复位有效	新复位触发			
	上电复位	调试复位	内部应用复位	应用复位
内部应用复位	在缺省延迟后重新开始计数	不变	不变	不变
应用复位	在缺省延迟后重新开始计数	不变	在缺省延迟后重新开始计数	不变

表 8-3 RSTCNTD 重新开始计数

复位有效	新复位触发			
	上电复位	调试复位	内部应用复位	应用复位
调试复位	在缺省延迟后重新开始计数	不变	不变	不变

复位计数器 RSTCNTx 确保了可配置的最小复位长度。若计数器已开始递减计数后复位请求触发源仍保持有效，则不启动计数器工作，复位控制模块保持复位状态有效、直至复位请求源失效。

#### 8.4.2.2 复位失效

当满足以下所有条件时，复位失效：

- 复位计数器已过期（计数到 0）。
- 当前无特定复位类型（或更高类型）的复位请求触发被配置。

##### 举例 1

复位请求触发源 A 有效，产生应用复位。若 RSTCNTA 递减计数到 0 之前复位请求源失效，RSTCNTA 递减计数到 0 时应用复位失效。若 RSTCNTA 递减计数到 0 之后复位请求源失效，复位请求触发源失效时应用复位失效。

##### 举例 2

复位请求触发源 A 有效，产生应用复位。RSTCNTA 递减计数到 0 之前复位请求源 A 失效。复位请求触发源 B 在复位请求触发源 A 之后、RSTCNTA 递减计数到 0 之前有效。复位请求触发源 B 也配置为产生应用复位。若 RSTCNTA 递减计数到 0 之前复位请求源 B 失效，RSTCNTA 递减计数到 0 时应用复位失效。若 RSTCNTA 递减计数到 0 之后复位请求源 B 失效，复位请求触发源 B 失效时应用复位失效。

#### 8.4.3 调试复位生效

和其它复位类型不同，只有在满足以下两个条件时调试复位才生效：

- 请求调试复位的复位请求触发源有效
- 应用复位已有效

#### 8.4.4 不同复位类型组合工作

不同复位类型可组合工作：

- 上电复位生效时会自动使以下复位类型生效：
  - 调试复位
  - 内部应用复位
  - 应用复位
- 内部应用复位生效时会自动使以下复位类型生效：
  - 应用复位

#### 8.4.5 复位请求触发源

系统内各种复位请求触发源归纳如下。

## 上电复位引脚 $\overline{\text{PORST}}$

通过拉低引脚  $\overline{\text{PORST}}$  异步请求上电复位。

## 电源看门狗 (SWD)

若 I/O 域的供电电压低于规定值，则产生未同步的复位请求触发源（若允许产生 SWD）。这确保了出现电源故障时的行为具有可重复性。这也可用来重启系统而无需使用引脚  $\overline{\text{PORST}}$ 。只要 I/O 电源域未达到要求的电压值，系统保持复位状态。

## 内核供电电压检验 (PVC\_M 和 PVC\_1)

若内核供电电压低于主电源域 (PVC\_M) 正常工作所需的电压值，可产生复位请求触发源。设置  $\text{PVC\_MCON0.L1RSTEN} = 1_{\text{B}}$  配置产生上电复位。若  $\text{PVC\_MCON0.L1RSTEN} = 1_{\text{B}}$ ，电平检查匹配时 PVC\_M1 的复位请求生效；若  $\text{PVC\_MCON0.L2RSTEN} = 1_{\text{B}}$ ，电平检查匹配时 PVC\_M2 的复位请求生效。

若内核供电电压低于应用电源域 (PVC\_1) 正常工作所需的电压值，可产生复位请求触发源。设置  $\text{PVC\_1CON0.L1RSTEN} = 1_{\text{B}}$  配置产生上电复位（仅应用电源域）。若  $\text{PVC\_1CON0.L1RSTEN} = 1_{\text{B}}$ ，电平检查匹配时 PVC\_11 的复位请求生效；若  $\text{PVC\_1CON0.L2RSTEN} = 1_{\text{B}}$ ，电平检查匹配时 PVC\_12 的复位请求生效。

有关电源检验电路的具体描述请参见 [章节 8.6.2](#)。

## $\overline{\text{ESRx}}$

$\overline{\text{ESRx}}$  复位请求触发源会产生可配置的复位。可由  $\text{RSTCON1.ESRx}$  配置复位类型。

引脚  $\overline{\text{ESRx}}$  可用作内部应用复位和应用复位的外部复位输入和复位输出（漏极开路）。此外，通过寄存器  $\text{ESREXCONx}$  ( $x = 1, 2$ ) 使能的多个 GPIO 引出端会干扰  $\overline{\text{ESR}}$  引脚的功能。GPIO 和  $\overline{\text{ESRx}}$  引脚可被单独使能/禁止和组合触发产生复位。

若  $\overline{\text{ESRx}}$  引脚被使能用作复位输出且输入电平为低（指示它仍由外部驱动至低电平）的同时输出级被禁用，复位电路使芯片保持复位状态直至检测到  $\overline{\text{ESRx}}$  上出现高电平。RSTCNTCON.RELA 的最小值必须为复位值。

*注：只在复位计数器 RSTCNTA 工作期间将复位输出驱动至低电平。若复位时间延长，不再驱动复位输出。*

## 软件

软件复位请求触发源会产生可配置的复位。可由  $\text{RSTCON0.SW}$  配置复位类型。

### 看门狗定时器

WDT 复位请求触发源会产生可配置的复位。可由 **RSTCON1**.WDT 配置复位类型。WDT 溢出事件请求 WDT 复位，具体描述见 [章节 8.12](#)。

### CPU

CPU 复位请求触发源会产生可配置的复位。可由 **RSTCON0**.CPU 配置复位类型。执行 SRST 指令时请求 CPU 复位。

### 存储器奇偶校验

MP 复位请求触发源会产生可配置的复位。可由 **RSTCON1**.MP 配置复位类型。具体描述见 [章节 8.14.2](#)。

### OCDS 模块

OCDS 模块有多种请求不同复位类型的选择：

1. 通过 OCDS 复位功能或位 CBS\_OJCONF.RSTCL1 AND 位 CBS\_OJCONF.RSTCL3 请求调试复位
2. 通过位 CBS\_OJCONF.RSTCL2 请求内部应用复位
3. 通过位 CBS\_OJCONF.RSTCL3 请求应用复位

#### 8.4.5.1 复位请求源概览

不同复位请求源和它们所请求的复位类型归纳见 [表 8-4](#)。

**表 8-4 不同复位请求源所激活的复位类型**

复位请求触发源	应用复位	内部应用复位	调试复位
<b>PORST</b>	激活	激活	激活
<b>SWD</b>	激活	激活	激活
<b>PVC_M1</b>	激活	激活	激活
<b>PVC_M2</b>	激活	激活	激活
<b>PVC_11</b>	激活	激活	激活
<b>PVC_12</b>	激活	激活	激活
<b>ESR0</b>	可配置	可配置	不激活

**系统控制单元 (SCU)**

复位请求触发源	应用复位	内部应用复位	调试复位
<b>ESR1</b>	可配置	可配置	不激活
<b>ESR2</b>	可配置	可配置	不激活
<b>WDT</b>	可配置	可配置	不激活
<b>SW</b>	可配置	可配置	不激活
<b>CPU</b>	可配置	可配置	不激活
<b>MP</b>	可配置	可配置	不激活
<b>OCDS 复位</b>	不激活	不激活	激活 <sup>1)</sup>
<b>CBS_OJCONF.RSTCL1</b>	不激活	不激活	激活 <sup>1)</sup>
<b>CBS_OJCONF.RSTCL2</b>	激活	激活	不激活
<b>CBS_OJCONF.RSTCL3</b>	激活	不激活	不激活

1) 只有当应用复位有效或被并行请求时调试复位被激活。

### 8.4.6 模块复位行为

**表 8-5** 罗列出不同复位类型对 XE16xyM 各功能的影响。“X”表示该模块中至少有部分寄存器/位受该种复位的影响。

**表 8-5 复位对器件功能的影响**

模块/功能		应用复位	内部应用复位	调试复位
<b>CPU 内核</b>		X	X	X
<b>外设 (SCU 和 RTC 除外)</b>		X	X	X
<b>SCU</b>		X	不受影响	不受影响
<b>RTC</b>		不受影响	不受影响	X
<b>片上静态 RAM<sup>1)</sup></b>	<b>DPRAM</b>	不受影响, 可靠	不受影响, 可靠	不受影响, 可靠
	<b>PSRAM</b>	不受影响, 可靠	不受影响, 可靠	不受影响, 可靠

**系统控制单元 (SCU)**

模块/功能		应用复位	内部应用复位	调试复位
	<b>DSRAM</b>	不受影响, 可靠	不受影响, 可靠	不受影响, 可靠
<b>Flash 存储器</b>		X <sup>2)</sup>	X <sup>2)</sup>	不受影响, 可靠
<b>JTAG 接口</b>		不受影响	不受影响	不受影响
<b>OCDS</b>		不受影响	不受影响	X
<b>振荡器, PLL</b>		不受影响	不受影响	不受影响
<b>端口引脚</b>		不受影响 <sup>3)</sup>	X	不受影响
<b>引脚 ESRx</b>		不受影响	X	不受影响

1) 此处“可靠”表示冗余也不受复位影响。

2) 部分 flash 存储器只能由上电复位来复位, 详见 flash 一章的描述。

3) 片内外设复位可改变输出上的数据驱动, 参见“并行端口”一章“复位操作”一节中有关端口行为的描述。

## 8.4.7 复位控制器寄存器

### 8.4.7.1 状态寄存器

执行复位操作后, 复位状态寄存器指示上次复位的类型。每次的复位操作将更新复位状态寄存器。

#### RSTSTAT0

复位状态 0 寄存器

ESFR (F0B2H/59H)

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>SW</b>		<b>CPU</b>		<b>0</b>											
rh		rh		r											

符号	位序号	类型	功能描述
<b>CPU</b>	[13:12]	rh	<b>CPU 复位类型状态</b> 00 <sub>B</sub> 上次复位和 CPU 复位请求无关。

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
			01 <sub>B</sub> 保留。 10 <sub>B</sub> 上次复位和 CPU 复位请求有关。产生内部应用复位和应用复位。 11 <sub>B</sub> 上次复位和 CPU 复位请求有关。产生应用复位。
<b>SW</b>	[15:14]	rh	<b>软件复位类型状态</b> 00 <sub>B</sub> 上次复位和软件复位请求无关。 01 <sub>B</sub> 保留。 10 <sub>B</sub> 上次复位和软件复位请求有关。产生内部应用复位和应用复位。 11 <sub>B</sub> 上次复位和软件复位请求有关。产生应用复位。
<b>0</b>	[11:0]	r	<b>保留</b> 读操作返回 0；应写入 0。

**RSTSTAT1**

**复位状态 1 寄存器**

**ESFR (F0B4<sub>H</sub>/5A<sub>H</sub>)**

**复位值: F000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>ST1</b>	<b>STM</b>	<b>0</b>				<b>MP</b>		<b>WDT</b>		<b>ESR2</b>		<b>ESR1</b>			<b>ESR0</b>
rh	rh	r				rh		rh		rh		rh			rh

符号	位序号	类型	功能描述
<b>ESR0</b>	[1:0]	rh	<b>ESR0 复位状态</b> 00 <sub>B</sub> 上次复位和 $\overline{\text{ESR0}}$ 复位请求无关。 01 <sub>B</sub> 保留。 10 <sub>B</sub> 上次复位和 $\overline{\text{ESR0}}$ 复位请求有关。产生内部应用复位和应用复位。 11 <sub>B</sub> 上次复位和 $\overline{\text{ESR0}}$ 复位请求有关。产生应用复位。

符号	位序号	类型	功能描述
<b>ESR1</b>	[3:2]	rh	<p><b>ESR1 复位状态</b></p> <p>00<sub>B</sub> 上次复位和 <math>\overline{\text{ESR1}}</math> 复位请求无关。</p> <p>01<sub>B</sub> 保留。</p> <p>10<sub>B</sub> 上次复位和 <math>\overline{\text{ESR1}}</math> 复位请求有关。产生内部应用复位和应用复位。</p> <p>11<sub>B</sub> 上次复位和 <math>\overline{\text{ESR1}}</math> 复位请求有关。产生应用复位。</p>
<b>ESR2</b>	[5:4]	rh	<p><b>ESR2 复位状态</b></p> <p>00<sub>B</sub> 上次复位和 <math>\overline{\text{ESR2}}</math> 复位请求无关。</p> <p>01<sub>B</sub> 保留。</p> <p>10<sub>B</sub> 上次复位和 <math>\overline{\text{ESR2}}</math> 复位请求有关。产生内部应用复位和应用复位。</p> <p>11<sub>B</sub> 上次复位和 <math>\overline{\text{ESR2}}</math> 复位请求有关。产生应用复位。</p>
<b>WDT</b>	[7:6]	rh	<p><b>WDT 复位状态</b></p> <p>00<sub>B</sub> 上次复位和 WDT 复位请求无关。</p> <p>01<sub>B</sub> 保留。</p> <p>10<sub>B</sub> 上次复位和 WDT 复位请求有关。产生内部应用复位和应用复位。</p> <p>11<sub>B</sub> 上次复位和 WDT 复位请求有关。产生应用复位。</p>
<b>MP</b>	[9:8]	rh	<p><b>MP 复位状态</b></p> <p>00<sub>B</sub> 上次复位和 MP 复位请求无关。</p> <p>01<sub>B</sub> 保留。</p> <p>10<sub>B</sub> 上次复位和 MP 复位请求有关。产生内部应用复位和应用复位。</p> <p>11<sub>B</sub> 上次复位和 MP 复位请求有关。产生应用复位。</p>



符号	位序号	类型	功能描述
<b>STM</b>	[13:12]	rh	<b>DMP_M 上电复位状态</b> 00 <sub>B</sub> 上次复位和 DMP_M 上电复位请求无关。 01 <sub>B</sub> 上次复位和 DMP_M 上电复位请求无关。 10 <sub>B</sub> 上次复位和 DMP_M 上电复位请求无关。 11 <sub>B</sub> 上次复位和 DMP_M 上电复位请求有关。
<b>ST1</b>	[15:14]	rh	<b>DMP_1 上电复位状态</b> 00 <sub>B</sub> 上次复位和 DMP_1 上电复位请求无关。 01 <sub>B</sub> 上次复位和 DMP_1 上电复位请求无关。 10 <sub>B</sub> 上次复位和 DMP_1 上电复位请求无关。 11 <sub>B</sub> 上次复位和 DMP_1 上电复位请求有关。
<b>0</b>	[11:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

## RSTSTAT2

复位状态 2 寄存器

**ESFR (F0B6H/5BH)**

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>					<b>OJCONF3</b>	<b>OJCONF2</b>	<b>OJCONF1</b>	<b>OJCONF0</b>	<b>DB</b>						
r					rh	rh	rh	rh	rh						

符号	位序号	类型	功能描述
<b>DB</b>	[1:0]	rh	<b>调试复位状态</b> 00 <sub>B</sub> 上次复位和 DB 复位请求无关。 01 <sub>B</sub> 上次复位和 DB 复位请求无关。 10 <sub>B</sub> 上次复位和 DB 复位请求无关。 11 <sub>B</sub> 上次复位和 DB 复位请求有关。
<b>OJCONF0</b>	[3:2]	rh	<b>OJCONF0 复位状态</b> 读取返回 0；应写入 0。
<b>OJCONF1</b>	[5:4]	rh	<b>OJCONF1 复位状态</b> 00 <sub>B</sub> 上次复位和 OJCONF1 复位请求无关。

符号	位序号	类型	功能描述
			01 <sub>B</sub> 上次复位和 OJCONF1 复位请求无关。 10 <sub>B</sub> 上次复位和 OJCONF1 复位请求无关。 11 <sub>B</sub> 上次复位和 OJCONF1 复位请求有关。产生调试复位。
<b>OJCONF2</b>	[7:6]	rh	<b>OJCONF2 复位状态</b> 00 <sub>B</sub> 上次复位和 OJCONF2 复位请求无关。 01 <sub>B</sub> 上次复位和 OJCONF2 复位请求无关。 10 <sub>B</sub> 上次复位和 OJCONF2 复位请求有关。产生内部应用复位和应用复位。 11 <sub>B</sub> 上次复位和 OJCONF2 复位请求无关。
<b>OJCONF3</b>	[9:8]	rh	<b>OJCONF3 复位状态</b> 00 <sub>B</sub> 上次复位和 OJCONF3 复位请求无关。 01 <sub>B</sub> 上次复位和 OJCONF3 复位请求无关。 10 <sub>B</sub> 上次复位和 OJCONF3 复位请求无关。 11 <sub>B</sub> 上次复位和 OJCONF3 复位请求有关。产生应用复位。
<b>0</b>	[15:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 8.4.7.2 配置寄存器

以下这些寄存器对各复位请求触发源进行配置。

#### RSTCON0

复位配置 0 寄存器

ESFR (F0B8H/5CH)

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>SW</b>	<b>CPU</b>	<b>0</b>													
rw	rw	rw													

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
<b>CPU</b>	[13:12]	rw	<b>CPU 复位类型选择</b> 该位域定义 CPU 复位请求源触发产生哪种复位。 00 <sub>B</sub> 不产生复位。 01 <sub>B</sub> 保留，不使用该组合。 10 <sub>B</sub> 产生内部应用复位和应用复位。 11 <sub>B</sub> 产生应用复位。
<b>SW</b>	[15:14]	rw	<b>软件复位类型选择</b> 该位域定义软件复位请求源触发产生哪种复位。 00 <sub>B</sub> 不产生复位。 01 <sub>B</sub> 保留，不使用该组合。 10 <sub>B</sub> 产生内部应用复位和应用复位。 11 <sub>B</sub> 产生应用复位。
<b>0</b>	[11:10]	rw	<b>保留</b> 必须写入复位值 0。

**RSTCON1**

**复位配置 1 寄存器**

**ESFR (F0BA<sub>H</sub>/5D<sub>H</sub>)**

**复位值: 0002<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>						<b>MP</b>		<b>WDT</b>		<b>ESR2</b>		<b>ESR1</b>		<b>ESR0</b>	
rw						rw		rw		rw		rw		rw	

符号	位序号	类型	功能描述
<b>ESR0</b>	[1:0]	rw	<b>ESR0 复位类型选择</b> 该位域定义 $\overline{\text{ESR0}}$ 复位请求源触发产生哪种复位。 00 <sub>B</sub> 不产生复位。 01 <sub>B</sub> 保留，不使用该组合。 10 <sub>B</sub> 产生内部应用复位和应用复位。 11 <sub>B</sub> 产生应用复位。

符号	位序号	类型	功能描述
<b>ESR1</b>	[3:2]	rw	<b>ESR1 复位类型选择</b> 该位域定义 $\overline{\text{ESR1}}$ 复位请求源触发产生哪种复位。 00 <sub>B</sub> 不产生复位。 01 <sub>B</sub> 保留，不使用该组合。 10 <sub>B</sub> 产生内部应用复位和应用复位。 11 <sub>B</sub> 产生应用复位。
<b>ESR2</b>	[5:4]	rw	<b>ESR2 复位类型选择</b> 该位域定义 $\overline{\text{ESR2}}$ 复位请求源触发产生哪种复位。 00 <sub>B</sub> 不产生复位。 01 <sub>B</sub> 保留，不使用该组合。 10 <sub>B</sub> 产生内部应用复位和应用复位。 11 <sub>B</sub> 产生应用复位。
<b>WDT</b>	[7:6]	rw	<b>WDT 复位类型选择</b> 该位域定义 $\overline{\text{WDT}}$ 复位请求源触发产生哪种复位。 00 <sub>B</sub> 不产生复位。 01 <sub>B</sub> 保留，不使用该组合。 10 <sub>B</sub> 产生内部应用复位和应用复位。 11 <sub>B</sub> 产生应用复位。
<b>MP</b>	[9:8]	rw	<b>MP 复位类型选择</b> 该位域定义 $\overline{\text{MP}}$ 复位请求源触发产生哪种复位。 00 <sub>B</sub> 不产生复位。 01 <sub>B</sub> 保留，不使用该组合。 10 <sub>B</sub> 产生内部应用复位和应用复位。 11 <sub>B</sub> 产生应用复位。
<b>0</b>	[15:10]	rw	<b>保留</b> 应写入 0。

## RSTCNTCON

复位计数器控制寄存器

ESFR (F1B2<sub>H</sub>/D9<sub>H</sub>)

复位值: 0A0A<sub>H</sub>

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

符号	位序号	类型	功能描述
RELA	[7:0]	rw	<b>应用复位计数器重载值</b> 该位域定义 RSTCNTA 的重载值。计数器 RSTCNTA 始终从该重载值开始计数。 该计数器的值用于内部应用复位和应用复位。 对于 ESRx 复位，计数器的值一定不能小于复位值。
RELD	[15:8]	rw	<b>调试复位计数器重载值</b> 该位域定义 RSTCNTD 的重载值。计数器 RSTCNTD 始终从该重载值开始计数。 该计数器的值用于调试复位。 对于 ESRx 复位，计数器的值一定不能小于复位值。

## 软件复位控制寄存器

该寄存器用于控制软件复位操作。

## SWRSTCON

软件复位控制寄存器

ESFR (F0AE<sub>H</sub>/57<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SWCFG								0				SW RST REQ	SW BOOT		
rw								r				w	rw		

符号	位序号	类型	功能描述
SWBOOT	0	rw	<b>软件 Boot 配置选择</b> 0 <sub>B</sub> 应用复位时，不由 SWCFG 的值更新位域

符号	位序号	类型	功能描述
			STSTAT.HWCFG。 1 <sub>B</sub> 应用复位时，由 SWCFG 的值更新位域 STSTAT.HWCFG。
SWRSTREQ	1	w	软件复位请求 0 <sub>B</sub> 不请求软件复位。 1 <sub>B</sub> 请求软件复位。 <i>注：读取该位始终返回 0。</i>
SWCFG	[15:8]	rw	软件 Boot 配置 可通过该位域进行软件 boot 配置（不同于外部硬件配置）。 SWCFG 和寄存器 STSTAT 中的 HWCFG 编码相同。
0	[7:2]	r	保留 读操作返回 0；应写入 0。

## 8.5 外部服务请求 (ESR) 引脚

$\overline{\text{ESR}}$  引脚可提供多种功能：

- 用作复位请求触发源输入
- 用作复位输出
- 用作强制中断输入
- 用作 GSC 的触发输入
- 和其它产品功能复用
- 独立的引出端配置

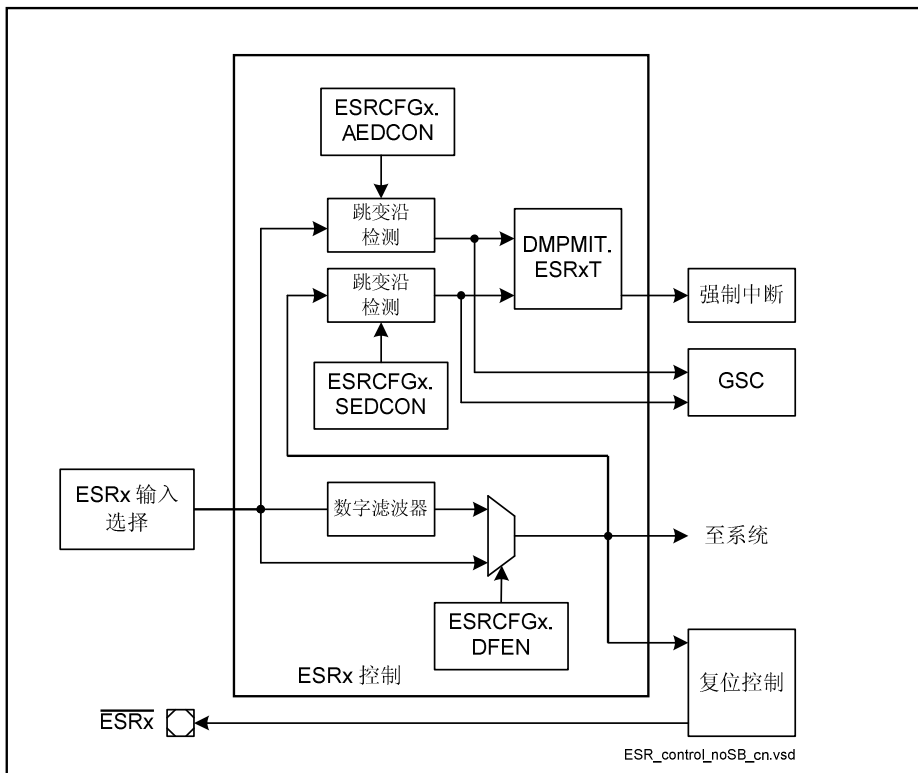
### 8.5.1 一般操作

每个  $\overline{\text{ESR}}$  引脚均配有跳变沿检测，可选择用作触发源的跳变沿。可通过位域  $\text{ESRCFGx.SEDCON}$  选择由单沿、双沿或不使用跳变沿触发。此外，通过一个数字（3-级取中值）滤波器（DF）来抑制毛刺。为了产生触发信号， $\overline{\text{ESRx}}$  引脚上的信号必须至少保持 2 个系统时钟周期（ $f_{\text{SYS}}$ ）的有效电平。可通过清除  $\text{ESRCFGx.DFEN}$  禁用数字滤波器。

每个  $\overline{\text{ESRx}}$  引脚可单独配置。

需要注意：若产生一个  $\overline{\text{ESR}}$  触发信号，将会引发所有相关功能（复位、强制中断、GSC 和非 SCU 模块功能）。若该触发信号所引发的部分操作不应发生，必须单独禁止相关特性。

用作 ESR 触发输入的引脚必须被配置为输入引脚。



**图 8-19 ESRx 框图**

此外，可通过寄存器  $\text{ESREXCONx}$  配置 ESR 和其它产品功能（如串口输入）复用，将它们的输入逻辑与（AND）用以产生触发信号。因而，若有多个引脚用于产生 ESR 触发信号，则这些引脚上的任意信号必须高电平无效。此外，可控制部分输入翻转以支持高电平有效。

为了扩展复用的可能，ESRx 输入选择结构的多路输入逻辑组合（AND），生成一个公共事件，该公共事件若由  $\text{ESREXCONx.ESRIN12EN}$  使能，则和 ESR 输入、以及

另一个输入组合模块的组合输入的输出进行第二次逻辑与。这样的话，即使另一个 ESR 逻辑用作其它用途，也可实现由所有可能的输入来触发 ESR 功能。

引脚  $\overline{\text{ESR0}}$  不和其它产品功能复用。

**章节 8.17.2** 给出哪些外设输入信号位于 ESR 复用引脚上。

$\overline{\text{ESR1}}$  和  $\overline{\text{ESR2}}$  的 ESR 输入选择功能如 **图 8-20** 所示。



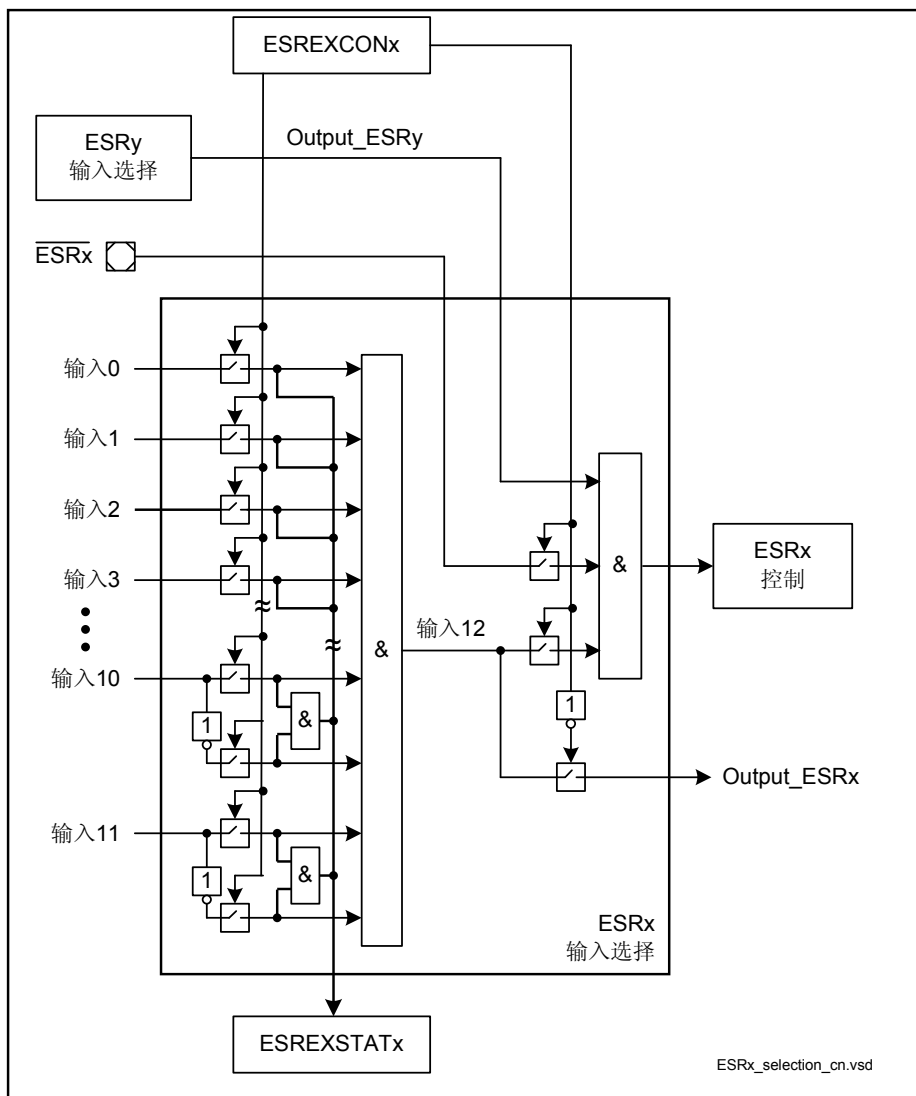


图 8-20  $\overline{\text{ESRx}}$  输入选择

最多有三个  $\overline{\text{ESR}}$  引脚 ( $\overline{\text{ESR0}}$  /  $\overline{\text{ESR1}}$  /  $\overline{\text{ESR2}}$ ) 可用。引脚  $\overline{\text{ESR1}}$  /  $\overline{\text{ESR2}}$  是否可用取决于器件型号及封装，详见数据手册。

即使引脚  $\overline{\text{ESR1}}$  或  $\overline{\text{ESR2}}$  不可用，可通过寄存器 **ESREXCON1** 和 **ESREXCON2** 配置 **ESR** 和其它产品功能（如串口输入）复用以触发 **ESR** 操作。

### 8.5.1.1 $\overline{\text{ESR}}$ 用作复位输入

引脚  $\overline{\text{ESRx}}$  可用作内部应用复位和应用复位的外部复位输入和复位输出（漏极开路）。此外，通过寄存器 **ESREXCONx** ( $x = 1, 2$ ) 使能的多个 **GPIO** 引出端会干扰 **ESR** 引脚的功能。**GPIO** 和  $\overline{\text{ESRx}}$  引脚可被单独使能/禁止和组合触发产生复位。有关复位系统的详细描述，请参见 [章节 8.4](#)。

*注：复位输出只在复位计数器 **RSTCNTA** 工作期间有效。若复位时间延长，复位输出不再生效。*

### 8.5.1.2 $\overline{\text{ESR}}$ 用作复位输出

若  $\overline{\text{ESRx}}$  引脚被使能用作复位输出且输入电平为低（指示它仍由外部驱动至低电平）的同时输出级被禁用，复位电路使芯片保持复位状态直至检测到  $\overline{\text{ESRx}}$  上出现高电平。只有当 **RSTCNTA** 计数时，内部输出级才在复位期间驱动一个低电平。在经过由 **RSTCNTCON.RELA** 定义的时间段之后，输出级失效。有关复位系统的详细描述，请参见 [章节 8.4](#)。

### 8.5.1.3 $\overline{\text{ESR}}$ 用作强制中断触发源

$\overline{\text{ESR}}$  可请求产生强制中断。其控制机制（是否产生强制中断、请求产生哪种强制中断）位于强制中断控制电路中。具体描述见 [章节 8.13](#)。

### 8.5.1.4 $\overline{\text{ESR}}$ 用作 **GSC** 的触发输入

在控制模式时， $\overline{\text{ESR}}$  可用于请求改变工作模式，具体描述见 [章节 8.7](#)。

### 8.5.1.5 和其它产品功能复用

还可使用其它端口输入（如串口输入）产生  $\overline{\text{ESR}}$  操作。[章节 8.17](#) 给出哪些其它外设输入位于 **ESR** 复用引脚上。

该特性可用于不同的应用场合：

- 外部中断或 **CCU6x** 触发源、以及 **CAN** 或 **USIC** 操作可将系统从时钟关闭模式唤醒

**系统控制单元 (SCU)**

- 外部中断或 CCU6x 触发源、以及 CAN 或 USIC 操作可请求进入时钟关闭模式
- 终止由外部事件触发 CCU6x 模块的输入

有关外部中断触发的具体描述，请参见 [章节 8.9](#)。

#### 8.5.1.6 ESR 引脚配置

由位域 ESRCFGx.PC 选择引脚配置。

每个引脚的引出端功能可被单独配置，包括：

- 驱动类型选择（漏极开路或推挽输出）
- 输出驱动使能（输入和/或输出能力）
- 上拉/下拉电阻使能

寄存器 ESRCFG0、ESRCFG1 和 ESRCFG2 中位域 PC 的编码定义见下表。

*注：和端口寄存器位域 Pn\_IOC Rx.PC 的编码相同。*

**表 8-6 PC 编码**

PCx[3:0]	选择上拉/下拉/输出功能	I/O	输出特性
0000 <sub>B</sub>	不激活上拉/下拉器件	输入不反相， 掉电模式下 输入级正常工作	
0001 <sub>B</sub>	激活下拉器件		
0010 <sub>B</sub>	激活上拉器件		
0011 <sub>B</sub>	不激活上拉/下拉器件		
0100 <sub>B</sub>	不激活上拉/下拉器件	输入反相， 掉电模式下 输入级正常工作	
0101 <sub>B</sub>	激活下拉器件		
0110 <sub>B</sub>	激活上拉器件		
0111 <sub>B</sub>	不激活上拉/下拉器件		
1000 <sub>B</sub>	输出 ESRCFGx.OUT	输出， 掉电模式下 输入级不反相、 不工作	推挽
1001 <sub>B</sub>	输出 ESRCFGx.OUT		
1010 <sub>B</sub>	内部应用复位时输出驱动 0， 否则输出 1。		
1011 <sub>B</sub>	应用复位时输出驱动 0， 否则 输出 1。		

**系统控制单元 (SCU)**

PCx[3:0]	选择上拉/下拉/输出功能	I/O	输出特性
1100 <sub>B</sub>	输出 ESRCFGx.OUT		漏极开路， 当输出不驱动 0 时，激活上拉 器件
1101 <sub>B</sub>	输出 ESRCFGx.OUT		
1110 <sub>B</sub>	内部应用复位时输出驱动 0		
1111 <sub>B</sub>	应用复位时输出驱动 0		

### 8.5.2 $\overline{\text{ESR}}$ 控制寄存器

#### 8.5.2.1 配置寄存器

#### $\overline{\text{ESR}}$ 外部控制寄存器

$\overline{\text{ESR}}$  外部控制寄存器中存放不同输入的使能/禁止位，这些输入可用于激活  $\overline{\text{ESR}}$  操作。 $\overline{\text{ESR0}}$  不使用该寄存器。

#### ESREXCON1

**ESR1 外部控制寄存器**                      **SFR (FF32<sub>H</sub>/99<sub>H</sub>)**                      **复位值: 0001<sub>H</sub>**

#### ESREXCON2

**ESR2 外部控制寄存器**                      **SFR (FF34<sub>H</sub>/9A<sub>H</sub>)**                      **复位值: 0001<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ESR N IN11 EN	ESR N IN10 EN	ESR IN12 EN	ESR IN11 EN	ESR IN10 EN	ESR IN9 EN	ESR IN8 EN	ESR IN7 EN	ESR IN6 EN	ESR IN5 EN	ESR IN4 EN	ESR IN3 EN	ESR IN2 EN	ESR IN1 EN	ESR IN0 EN	ESR EN
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

符号	位序号	类型	功能描述
<b>ESREN</b>	0	rW	<p><b><math>\overline{\text{ESRy}}</math> 引脚使能</b></p> <p>该位使能/禁用 <math>\overline{\text{ESRy}}</math> 引脚来激活 <math>\overline{\text{ESRy}}</math> 的所有相关操作。</p> <p>0<sub>B</sub> 来自引脚 <math>\overline{\text{ESRy}}</math> 的输入被禁用。</p>

符号	位序号	类型	功能描述
			1 <sub>B</sub> 来自引脚 $\overline{\text{ESRy}}$ 的输入被使能。
<b>ESRINxEN</b> (x = 0-11)	X+1	rw	<b>ESR 输入 X 使能</b> 该位使能/禁用输入 x 来激活 $\overline{\text{ESRy}}$ 的所有相关操作的。 0 <sub>B</sub> 输入被禁用。 1 <sub>B</sub> 输入被使能。
<b>ESRIN12EN</b>	13	rw	<b>ESR 输入 12 使能</b> 该位使能/禁用输入 12 来激活 $\overline{\text{ESRy}}$ 的所有相关操作。 0 <sub>B</sub> 禁止使用输入 12 来激活 $\overline{\text{ESRy}}$ 的所有相关操作。它可用作另一个 ESRz 输入选择结构中第二个逻辑与组合的输入。 1 <sub>B</sub> 使能使用输入 12 来激活 $\overline{\text{ESRy}}$ 的所有相关操作。
<b>ESRNIN10EN</b>	14	rw	<b>反相 ESR 输入 10 使能</b> 该位使能/禁用反相输入 10 来激活 $\overline{\text{ESRy}}$ 的所有相关操作的。 0 <sub>B</sub> 输入被禁用。 1 <sub>B</sub> 输入被使能。
<b>ESRNIN11EN</b>	15	rw	<b>反相 ESR 输入 11 使能</b> 该位使能/禁用反相输入 11 来激活 $\overline{\text{ESRy}}$ 的所有相关操作的。 0 <sub>B</sub> 输入被禁用。 1 <sub>B</sub> 输入被使能。

### ESREXSTAT1

**ESR1 外部状态寄存器**

**SFR (FF36H/9B<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

### ESREXSTAT2

**ESR2 外部状态寄存器**

**SFR (FF38H/9C<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

**系统控制单元 (SCU)**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>IN11</b>	<b>IN10</b>	<b>IN9</b>	<b>IN8</b>	<b>IN7</b>	<b>IN6</b>	<b>IN5</b>	<b>IN4</b>	<b>IN3</b>	<b>IN2</b>	<b>IN1</b>	<b>IN0</b>	<b>ESR</b>		
r	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

符号	位序号	类型	功能描述
<b>ESR</b>	0	rh	<p><b>输入 <math>\overline{\text{ESR}}_y</math> 的状态</b></p> <p>若 <math>\text{ESREXCON}_y.\text{ESREN}</math> 已置位, 输入 x 上有触发信号时该位置位。该位只能由软件清零。</p> <p>0<sub>B</sub> 输入 <math>\overline{\text{ESR}}_y</math> 上无触发信号。</p> <p>1<sub>B</sub> 自从被上次清零后, 输入 <math>\overline{\text{ESR}}_y</math> 上有触发信号。</p>
<b>INx</b> (x = 0-11)	x+1	rh	<p><b>输入 x 的状态</b></p> <p>若 <math>\text{ESREXCON}_y.\text{ESRINxEN}</math> 已置位, 输入 x 上有触发信号时该位置位。该位只能由软件清零。</p> <p>0<sub>B</sub> 输入 x 上无触发信号。</p> <p>1<sub>B</sub> 自从被上次清零后, 输入 x 上有触发信号。</p>
<b>0</b>	[15:13]	r	<p><b>保留</b></p> <p>读操作返回 0; 应写入 0。</p>

**CLRESREXSTAT1**

清除 **ESR1 外部状态寄存器**      **SFR (FF3A<sub>H</sub>/9D<sub>H</sub>)**      复位值: 0000<sub>H</sub>

**CLRESREXSTAT2**

清除 **ESR2 外部状态寄存器**      **SFR (FF3C<sub>H</sub>/9E<sub>H</sub>)**      复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>IN11</b>	<b>IN10</b>	<b>IN9</b>	<b>IN8</b>	<b>IN7</b>	<b>IN6</b>	<b>IN5</b>	<b>IN4</b>	<b>IN3</b>	<b>IN2</b>	<b>IN1</b>	<b>IN0</b>	<b>ESR</b>		
r	w	w	w	w	w	w	w	ww	w	w	w	w	w	w	w

符号	位序号	类型	功能描述
<b>ESR</b>	0	w	<p><b>清除输入 <math>\overline{\text{ESR}}_y</math> 的状态</b></p> <p>置位该位将清除 <math>\text{ESREXSTAT}_y.\text{ESR}</math>。读取该位始终返回 0。</p>

符号	位序号	类型	功能描述
			0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 ESREXSTATy.ESR。
<b>INx</b> (x = 0-11)	x+1	w	清除输入 x 的状态 置位该位将清除 ESREXSTATy.INx。读取该位始终返回 0。 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 ESREXSTATy.INx。
<b>0</b>	[15:13]	r	保留 读操作返回 0；应写入 0。

## ESR 配置寄存器

ESR 配置寄存器用于控制 ESR 引脚的行为。

### ESRCFG0

**ESR0 配置寄存器**                      **ESFR (F100<sub>H</sub>/80<sub>H</sub>)**                      复位值: 000E<sub>H</sub>

### ESRCFG1

**ESR1 配置寄存器**                      **ESFR (F102<sub>H</sub>/81<sub>H</sub>)**                      复位值: 0002<sub>H</sub>

### ESRCFG2

**ESR2 配置寄存器**                      **ESFR (F104<sub>H</sub>/82<sub>H</sub>)**                      复位值: 0002<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>					<b>AEDCON</b>	<b>SEDCON</b>	<b>IN</b>	<b>OUT</b>	<b>DFEN</b>	<b>PC</b>					
r					rw		rw	rh	rh	rw	rw				

符号	位序号	类型	功能描述
<b>PC</b>	[3:0]	rw	<b>ESRx 引脚控制</b> 该位域控制相关 ESRx 引脚的行为。 编码见 <a href="#">表 8-6</a> 。
<b>DFEN</b>	4	rw	<b>数字滤波器使能</b> 该位定义是否使用 ESRx 的 3 级中值滤波器。

符号	位序号	类型	功能描述
			<p>0<sub>B</sub> 旁路滤波器。</p> <p>1<sub>B</sub> 使用滤波器。</p>
<b>OUT</b>	5	rh	<p><b>数据输出</b></p> <p>该位可用作相关 <math>\overline{\text{ESRx}}</math> 引脚的输出值。</p> <p>0<sub>B</sub> 输出电平为 0。</p> <p>1<sub>B</sub> 输出电平为 1。</p> <p>该位由位域 <math>\text{ESRDAT.MOUTx}</math> 控制。</p>
<b>IN</b>	6	rh	<p><b>数据输入</b></p> <p>该位监控相关 <math>\overline{\text{ESRx}}</math> 引脚的输入值。</p>
<b>SEDCON</b>	[8:7]	rw	<p><b>同步跳变沿检测控制</b></p> <p>该位域定义同步通路上产生 <math>\overline{\text{ESRx}}</math> 触发信号的方式。</p> <p>00<sub>B</sub> 不产生触发信号。</p> <p>01<sub>B</sub> 上升沿产生触发信号。</p> <p>10<sub>B</sub> 下降沿产生触发信号。</p> <p>11<sub>B</sub> 上升沿和下降沿都产生触发信号。</p> <p>只有当位域 <math>\text{AEDCON}</math> 配置为 00<sub>B</sub> 时，该位域才能选用 00<sub>B</sub> 之外的配置。</p>
<b>AEDCON</b>	[10:9]	rw	<p><b>异步跳变沿检测控制</b></p> <p>该位域定义异步通路上产生 <math>\overline{\text{ESRx}}</math> 触发信号的方式。</p> <p>00<sub>B</sub> 不产生触发信号。</p> <p>01<sub>B</sub> 上升沿产生触发信号。</p> <p>10<sub>B</sub> 下降沿产生触发信号。</p> <p>11<sub>B</sub> 上升沿和下降沿都产生触发信号。</p> <p>只有当位域 <math>\text{SEDCON}</math> 配置为 00<sub>B</sub> 时，该位域才能选用 00<sub>B</sub> 之外的配置。</p>
<b>0</b>	[15:11]	r	<p><b>保留</b></p> <p>读操作返回 0；应写入 0。</p>



### 8.5.3 ESR 数据寄存器

#### 8.5.3.1 ESRDAT

若  $\overline{\text{ESRx}}$  用作数据端口， $\overline{\text{ESR}}$  数据寄存器中存放所需的控制位。

#### ESRDAT

**ESR 数据寄存器**

**ESFR (F106H/83H)**

**复位值: 0000H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										MOUT2		MOUT1		MOUT0	
r										w		w		w	

符号	位序号	类型	功能描述
<b>MOUT0</b>	[1:0]	w	<p><b>修改 <math>\overline{\text{ESR0}}</math> 的数据输出</b></p> <p>可通过该位域修改 <math>\overline{\text{ESRCFG0.OUT}}</math> 的值，从而更新 <math>\overline{\text{ESR0}}</math> 的输出。读取该位域始终返回 0。</p> <p>00<sub>B</sub> <math>\overline{\text{ESR0}}</math> 的输出（位 <math>\overline{\text{ESRCFG0.OUT}}</math>）不变。</p> <p>01<sub>B</sub> <math>\overline{\text{ESR0}}</math> 的输出（位 <math>\overline{\text{ESRCFG0.OUT}}</math>）被置位。</p> <p>10<sub>B</sub> <math>\overline{\text{ESR0}}</math> 的输出（位 <math>\overline{\text{ESRCFG0.OUT}}</math>）被清零。</p> <p>11<sub>B</sub> 保留，不使用该组合。</p>
<b>MOUT1</b>	[3:2]	w	<p><b>修改 <math>\overline{\text{ESR1}}</math> 的数据输出</b></p> <p>可通过该位域修改 <math>\overline{\text{ESRCFG1.OUT}}</math> 的值，从而更新 <math>\overline{\text{ESR1}}</math> 的输出。读取该位域始终返回 0。</p> <p>00<sub>B</sub> <math>\overline{\text{ESR1}}</math> 的输出（位 <math>\overline{\text{ESRCFG1.OUT}}</math>）不变。</p> <p>01<sub>B</sub> <math>\overline{\text{ESR1}}</math> 的输出（位 <math>\overline{\text{ESRCFG1.OUT}}</math>）被置位。</p> <p>10<sub>B</sub> <math>\overline{\text{ESR1}}</math> 的输出（位 <math>\overline{\text{ESRCFG1.OUT}}</math>）被清零。</p> <p>11<sub>B</sub> 保留，不使用该组合。</p>
<b>MOUT2</b>	[5:4]	w	<p><b>修改 <math>\overline{\text{ESR2}}</math> 的数据输出</b></p> <p>可通过该位域修改 <math>\overline{\text{ESRCFG2.OUT}}</math> 的值，从而更新 <math>\overline{\text{ESR2}}</math> 的输出。读取该位域始终返回 0。</p>

符号	位序号	类型	功能描述
			00 <sub>B</sub> $\overline{\text{ESR2}}$ 的输出 (位 ESRCFG2.OUT) 不变。 01 <sub>B</sub> $\overline{\text{ESR2}}$ 的输出 (位 ESRCFG2.OUT) 被置位。 10 <sub>B</sub> $\overline{\text{ESR2}}$ 的输出 (位 ESRCFG2.OUT) 被清零。 11 <sub>B</sub> 保留, 不使用该组合。
0	[15:6]	r	<b>保留</b> 读操作返回 0; 应写入 0。

## 8.6 电源电压及控制

XE16xyM 可由外部电源供电。内核供电电压可由片上电压调节器产生。

### 电源域

I/O 被分为 DMP\_A 和 DMP\_B 两部分。DMP\_A 包含所有和 ADC 相关的 I/O; DMP\_B 包含其余系统和通信 I/O。

片上逻辑的主要部分位于独立的内核电源域内 (DMP\_1), 另一个电源域 (DMP\_M, 下图中的灰色区域) 控制器件的重要低层部分以及备用 RAM (SBRAM)。

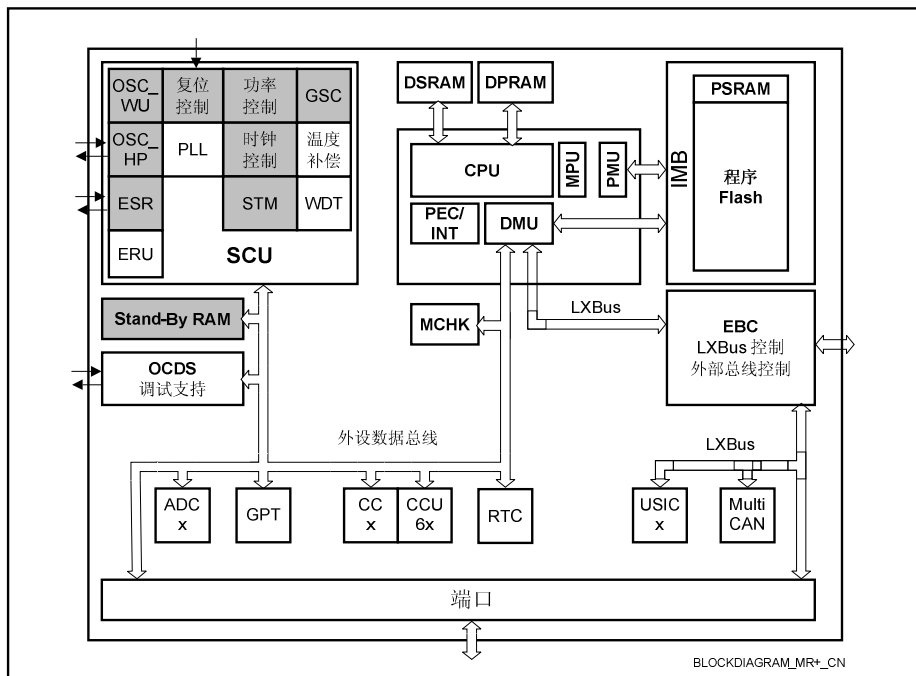


图 8-21 XE16xyM 电源域结构

### 电源电压及控制功能

电源电压及控制分为以下各部分：

- 监控供电电压
- 控制和调整供电电压

系统和通信 I/O 的引出端 I/O 域的供电电压（电源域 DMP\_B）由电压看门狗（SWD，见[章节 8.6.1](#)）监控。

两个内核电源域各自的内核电压由一个独立的电源检验电路（PVC）监控，PVC 提供两个监控电平。每个监控电平可根据实际电压值请求中断（如电源故障警告）或复位。PVC 可用于检测外部短路造成的低电压（见[章节 8.6.2](#)）。

可通过控制电压调节器关闭内核电源以降低漏电流（见[章节 8.6.3](#)）。

**表 8-7      XE16xyM 电源域电压及控制**

电源域	供电电源	供电电压 [V]	电压监控
引出端 I/O 域 (DMP_B)	外部电源	V <sub>DDPB</sub> : 3.0...5.5V 见数据手册	SWD
ADC I/O 域 (DMP_A)	外部电源	V <sub>DDPA</sub> : 3.0...5.5V 见数据手册	-
内核域 (DMP_M 和 DMP_1)	EVR_M EVR_1	V <sub>DDIM</sub> 、V <sub>DDI1</sub> : 1.5V 见数据手册	PVC_1 PVC_M

### 8.6.1      电压看门狗 (SWD)

系统和通信 I/O 的引出端 I/O 域的供电电压 (电源域 DMP\_B) 由电压看门狗监控。监控外部供电电压具有如下用途:

- **POR**

检测外部供电电压的提升, 因此无需外部上电复位 ( $\overline{\text{PORST}}$ ) 即可启动器件工作。

- **欠压**

检测外部供电电压的跌落, 因此无需外部上电复位 ( $\overline{\text{PORST}}$ ) 即可使器件进入省电状态。

- 可采用无附加状态信号的低成本电压调节器 (标准 3 引脚器件)。
- 确保在各种工作条件下 EVR 的供电电压足以产生有效的内核电压。

### 特性概述

SWD 的功能归纳如下:

- 一旦供电电压降至 V<sub>VAL</sub> 以下或当电压低于 V<sub>VAL</sub> 时, 触发上电复位
- 两个完全独立的阈值电平和比较器
- 16 个可选的阈值电平
- 省电模式 (仅 V<sub>VAL</sub> 检测正常工作)

## SWD 工作

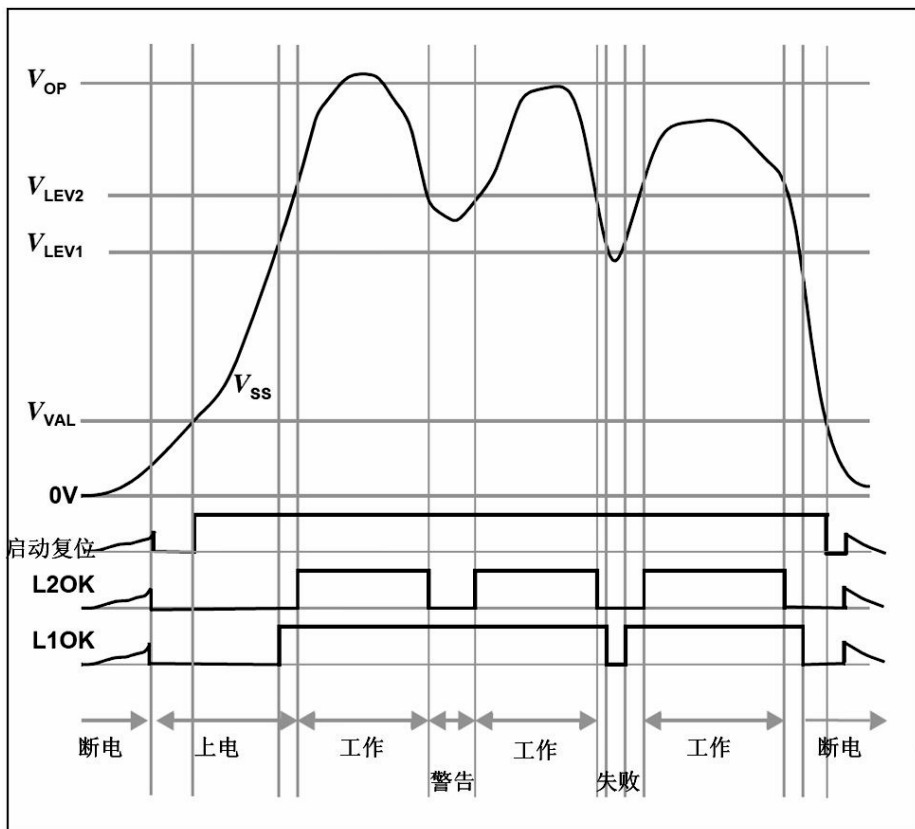


图 8-22 SWD 电源检验示例

固定阈值  $V_{VAL}$  定义 IO 域的绝对最小工作电压。若还未达到  $V_{VAL}$ ，器件保持复位状态。若  $V_{DDPB}$  超过  $V_{VAL}$ ，位 **SWDCON1.PON** 置位。

注：  $V_{VAL}$  的数值可查阅 XE16xyM 数据手册。

SWD 提供两个可调整的阈值电平 (LEV1 和 LEV2)，可由 **SWDCON0.LEV1V** 和 **SWDCON0.LEV2V** 分别编程，两个比较结果可由 **SWDCON0.L1OK** 和 **SWDCON0.L2OK** 监控。当电压小于或等于/大于设定的阈值电平时，可产生复位或中断请求。可由 **SWDCON0.LxRSTEN** 和 **SWDCON0.LxINTEN** 控制是否请求复位或中断；由 **SWDCON0.LxALEV** ( $x = 1, 2$ ) 控制触发哪种操作。

只能由软件控制 SWD (编程阈值电平)。

具备这些特性的 SWD 可替代（集成在外部 VR 中的）外部电压看门狗。它检测规定的最小供电电压，可配置用于监控其它电压电平。

*注：若使用  $\overline{PORST}$  引脚，它和 SWD 最低功率检测的功能相同。*

## SWD 的省电模式

这两个可编程阈值电平可被禁用（如不需要），称为 SWD 省电模式。最小工作电压检测（POR/欠压检测）始终有效、不能被关闭。置位 **SWDCON1.POWENSET** 进入 SWD 省电模式；置位 **SWDCON1.POWENCLR** 退出 SWD 省电模式。由 **SWDCON1.POWEN** 指示 SWD 省电模式是否工作。

*注：在进入省电模式之前，应关闭复位请求和中断请求操作。*

### 8.6.1.1 SWD 控制寄存器

以下这些寄存器用于控制 SWD。

#### SWDCON0

##### SWD 控制 0 寄存器

**ESFR (F080H/40H)**

**复位值: 0941H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>L2A LEV</b>	<b>L2 RST EN</b>	<b>L2 INT EN</b>	<b>L2 OK</b>	<b>LEV2V</b>			<b>L1A LEV</b>	<b>L1 RST EN</b>	<b>L1 INT EN</b>	<b>L1 OK</b>	<b>LEV1V</b>				
rw	rw	rw	rh	rw			rw	rw	rw	rh	rw				

符号	位序号	类型	功能描述
<b>LEV1V</b>	[3:0]	rw	<b>电平阈值 1 的电压</b> 该位域定义用作阈值 1 检测的电压电平。 电平阈值见数据手册。
<b>L1OK</b>	4	rh	<b>电平阈值 1 的检测结果</b> 0 <sub>B</sub> 供电电压小于电平阈值 1 的电压 LEV1V。 1 <sub>B</sub> 供电电压等于或大于电平阈值 1 的电压 LEV1V。
<b>L1INTEN</b>	5	rw	<b>电平阈值 1 中断请求使能</b> 该位域定义当供电电压和生效电平 L1ALEV 比较相匹配时是否请求中断。

符号	位序号	类型	功能描述
			<b>0<sub>B</sub></b> 不请求中断。 <b>1<sub>B</sub></b> 请求中断。
<b>L1RSTEN</b>	6	rw	<b>电平阈值 1 复位请求使能</b> 该位域定义当供电电压和生效电平 <b>L1ALEV</b> 比较相匹配时是否请求复位。 <b>0<sub>B</sub></b> 不请求复位。 <b>1<sub>B</sub></b> 请求复位。
<b>L1ALEV</b>	7	rw	<b>电平阈值 1 生效电平</b> <b>0<sub>B</sub></b> 当供电电压小于电平阈值 1 的电压 <b>LEV1V</b> 时, 请求执行 <b>L1INTEN</b> 和 <b>L1RSTEN</b> 所配置的操作。 <b>1<sub>B</sub></b> 当供电电压等于或大于电平阈值 1 的电压 <b>LEV1V</b> 时, 请求执行 <b>L1INTEN</b> 和 <b>L1RSTEN</b> 所配置的操作。
<b>LEV2V</b>	[11:8]	rw	<b>电平阈值 2 的电压</b> 该位域定义用作阈值 2 检测的电压电平。 电平阈值见数据手册。
<b>L2OK</b>	12	rh	<b>电平阈值 2 的检测结果</b> <b>0<sub>B</sub></b> 供电电压小于电平阈值 2 的电压 <b>LEV2V</b> 。 <b>1<sub>B</sub></b> 供电电压等于或大于电平阈值 2 的电压 <b>LEV2V</b> 。
<b>L2INTEN</b>	13	rw	<b>电平阈值 2 中断请求使能</b> 该位域定义当供电电压和生效电平 <b>L2ALEV</b> 比较相匹配时是否请求中断。 <b>0<sub>B</sub></b> 不请求中断。 <b>1<sub>B</sub></b> 请求中断。
<b>L2RSTEN</b>	14	rw	<b>电平阈值 2 复位请求使能</b> 该位域定义当供电电压和生效电平 <b>L2ALEV</b> 比较相匹配时是否请求复位。 <b>0<sub>B</sub></b> 不请求复位。 <b>1<sub>B</sub></b> 请求复位。

符号	位序号	类型	功能描述
L2ALEV	15	rw	<p><b>电平阈值 2 生效电平</b></p> <p>0<sub>B</sub> 当供电电压小于电平阈值 2 的电压 LEV2V 时, 请求执行 L2INTEN 和 L2RSTEN 所配置的操作。</p> <p>1<sub>B</sub> 当供电电压等于或大于电平阈值 2 的电压 LEV2V 时, 请求执行 L2INTEN 和 L2RSTEN 所配置的操作。</p>

## SWDCON1

### SWD 控制 1 寄存器

ESFR (F082<sub>H</sub>/41<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0											CLR PON	PON	POW EN	POW EN SET	POW EN CLR
r											w	rh	rh	w	w

符号	位序号	类型	功能描述
POWERCLR	0	w	<p><b>SWD 省电模式使能清零</b></p> <p>0<sub>B</sub> 无操作。</p> <p>1<sub>B</sub> 清除位 POWER。</p> <p>注: 读取该位始终返回 0。</p>
POWENSET	1	w	<p><b>SWD 省电模式使能置位</b></p> <p>0<sub>B</sub> 无操作。</p> <p>1<sub>B</sub> 置位位 POWER。</p> <p>注: 读取该位始终返回 0。</p>
POWER	2	rh	<p><b>SWD 省电模式使能</b></p> <p>0<sub>B</sub> 使能所有 SWD 功能。</p> <p>1<sub>B</sub> 使能 SWD 省电模式。比较器禁用。</p>
PON	3	rh	<p><b>上电状态标志</b></p> <p>0<sub>B</sub> 产生上电事件。</p> <p>1<sub>B</sub> 产生上电事件 (V<sub>DDP</sub> 变为大于 V<sub>VAL</sub>)。</p>



符号	位序号	类型	功能描述
CLRPN	4	w	清除上电状态标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 PON。 <i>注：读取该位始终返回 0。</i>
0	[15:5]	r	保留 读操作返回 0；应写入 0。

8.6.2 监控内核域的电压电平

电源检验电路（PVC）监控内核域的供电电压。它可用来监控两个可编程的、独立的电压电平。

内核域的电压由 PVC\_1 和 PVC\_M 监控。

特性概述

PVC 的特性归纳如下：

- 两个独立的比较器
- 阈值电平可选
- 关闭，禁用整个模块
- 可配置的生效电平

PVC 提供两个可调整的阈值电平（LEV1 和 LEV2），可由 PVCxCON0.LEV1V 和 PVCxCON0.LEV2V（x = M 或 1）分别编程。当前的供电电压和这两个阈值进行比较，比较结果可由 PVCxCON0.LEV1OK 和 PVCxCON0.LEV2OK（x = M 或 1）监控。

当内核域电压电平小于或等于/大于设定的阈值电平时，可产生复位或中断请求。若 PVCxCON0.L1INTEN 和/或 PVCxCON0.L2INTEN（x = M 或 1）置位，则请求中断；若 PVCxCON0.L1RSTEN 和/或 PVCxCON0.L2RSTEN（x = M 或 1）置位，则请求复位。

*注：单个阈值不允许同时请求中断和复位。*

8.6.2.1 PVC 状态和控制寄存器

以下这些寄存器是 PVC\_1 和 PVC\_M 的软件接口。

PVC1CON0

PVC\_1 控制 0 寄存器

ESFR（F014<sub>H</sub>/0A<sub>H</sub>）

复位值：0504<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>L2 ASEN</b>	<b>L2 RST EN</b>	<b>L2 INT EN</b>	<b>L2 ALEV</b>	<b>LEV2 OK</b>	<b>LEV2V</b>			<b>L1 ASEN</b>	<b>L1 RST EN</b>	<b>L1 INT EN</b>	<b>L1 ALEV</b>	<b>LEV1 OK</b>	<b>LEV1V</b>		
rw	rw	rw	rw	rh	rw			rw	rw	rw	rw	rh	rw		

符号	位序号	类型	功能描述
<b>LEV1V</b>	[2:0]	rw	<p><b>电平阈值 1 的电压</b></p> <p>该位域定义用来和 DMP_1 内核电压进行比较的电平阈值 1 的电压。</p> <p>不同配置对应的电平阈值见数据手册。</p>
<b>LEV1OK</b>	3	rh	<p><b>电平阈值 1 的检测结果</b></p> <p>0<sub>B</sub> DMP_1 的内核供电电压小于电平阈值 1 的电压 LEV1V。</p> <p>1<sub>B</sub> DMP_1 的内核供电电压等于或大于电平阈值 1 的电压 LEV1V。</p>
<b>L1ALEV</b>	4	rw	<p><b>电平阈值 1 生效电平</b></p> <p>0<sub>B</sub> 当内核供电电压小于电平阈值 1 的电压 LEV1V 时，请求执行 L1INTEN、L1RSTEN 和 L1ASEN 所配置的操作。</p> <p>1<sub>B</sub> 当内核供电电压等于或大于电平阈值 1 的电压 LEV1V 时，请求执行 L1INTEN、L1RSTEN 和 L1ASEN 所配置的操作。</p>
<b>L1INTEN</b>	5	rw	<p><b>电平阈值 1 中断请求使能</b></p> <p>该位域定义当比较检查成功时是否请求触发中断。由位 L1ALEV 定义何时比较检查成功。</p> <p>0<sub>B</sub> 不请求中断。</p> <p>1<sub>B</sub> 请求中断。</p>
<b>L1RSTEN</b>	6	rw	<p><b>电平阈值 1 复位请求使能</b></p> <p>该位域定义当比较检查成功时是否请求触发复位。由位 L1ALEV 定义何时比较检查成功。</p> <p>0<sub>B</sub> 不请求复位。</p> <p>1<sub>B</sub> 请求复位。</p>

符号	位序号	类型	功能描述
<b>L1ASEN</b>	7	rw	<b>电平阈值 1 异步操作使能</b> 该位域定义当比较检查成功时是否执行异步操作。 由位 <b>L1ALEV</b> 定义何时比较检查成功。 0 <sub>B</sub> 不执行异步操作。 1 <sub>B</sub> 执行异步操作。
<b>LEV2V</b>	[10:8]	rw	<b>电平阈值 2 的电压</b> 该位域定义用来和 <b>DMP_1</b> 内核电压进行比较的电平阈值 2 的电压。 不同配置对应的电平阈值见数据手册。
<b>LEV2OK</b>	11	rh	<b>电平阈值 2 的检测结果</b> 0 <sub>B</sub> <b>DMP_1</b> 的内核供电电压小于电平阈值 2 的电压 <b>LEV2V</b> 。 1 <sub>B</sub> <b>DMP_1</b> 的内核供电电压等于或大于电平阈值 2 的电压 <b>LEV2V</b> 。
<b>L2ALEV</b>	12	rw	<b>电平阈值 2 生效电平</b> 0 <sub>B</sub> 当内核供电电压小于电平阈值 2 的电压 <b>LEV2V</b> 时, 请求执行 <b>L2INTEN</b> 、 <b>L2RSTEN</b> 和 <b>L2ASEN</b> 所配置的操作。 1 <sub>B</sub> 当内核供电电压等于或大于电平阈值 2 的电压 <b>LEV2V</b> 时, 请求执行 <b>L2INTEN</b> 、 <b>L2RSTEN</b> 和 <b>L2ASEN</b> 所配置的操作。
<b>L2INTEN</b>	13	rw	<b>电平阈值 2 中断请求使能</b> 该位域定义当比较检查成功时是否请求触发中断。 由位 <b>L2ALEV</b> 定义何时比较检查成功。 0 <sub>B</sub> 不请求中断。 1 <sub>B</sub> 请求中断。
<b>L2RSTEN</b>	14	rw	<b>电平阈值 2 复位请求使能</b> 该位域定义当比较检查成功时是否请求触发复位。 由位 <b>L2ALEV</b> 定义何时比较检查成功。 0 <sub>B</sub> 不请求复位。 1 <sub>B</sub> 请求复位。

符号	位序号	类型	功能描述
<b>L2ASEN</b>	15	rw	<b>电平阈值 2 异步操作使能</b> 该位域定义当比较检查成功时是否执行异步操作。 由位 L2ALEV 定义何时比较检查成功。 0 <sub>B</sub> 不执行异步操作。 1 <sub>B</sub> 执行异步操作。

### PVCMCON0

#### PVC\_M 控制 0 寄存器

**MEM (F1E4<sub>H</sub>- -)**

**复位值: 0544<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>L2 ASEN</b>	<b>L2 RST EN</b>	<b>L2 INT EN</b>	<b>L2 ALEV</b>	<b>LEV2 OK</b>	<b>LEV2V</b>			<b>L1 ASEN</b>	<b>L1 RST EN</b>	<b>L1 INT EN</b>	<b>L1 ALEV</b>	<b>LEV1 OK</b>	<b>LEV1V</b>		
rw	rw	rw	rw	rh	rw			rw	rw	rw	rw	rh	rw		

符号	位序号	类型	功能描述
<b>LEV1V</b>	[2:0]	rw	<b>电平阈值 1 的电压</b> 该位域定义用来和 DMP_M 内核电压进行比较的电平阈值 1 的电压。 不同配置对应的电平阈值见数据手册。
<b>LEV1OK</b>	3	rh	<b>电平阈值 1 的检测结果</b> 0 <sub>B</sub> DMP_M 的内核供电电压小于电平阈值 1 的电压 LEV1V。 1 <sub>B</sub> DMP_M 的内核供电电压等于或大于电平阈值 1 的电压 LEV1V。
<b>L1ALEV</b>	4	rw	<b>电平阈值 1 生效电平</b> 0 <sub>B</sub> 当内核供电电压小于电平阈值 1 的电压 LEV1V 时，请求执行 L1INTEN、L1RSTEN 和 L1ASEN 所配置的操作。 1 <sub>B</sub> 当内核供电电压等于或大于电平阈值 1 的电压 LEV1V 时，请求执行 L1INTEN、L1RSTEN 和 L1ASEN 所配置的操作。

符号	位序号	类型	功能描述
<b>L1INTEN</b>	5	rw	<b>电平阈值 1 中断请求使能</b> 该位域定义当比较检查成功时是否请求触发中断。 由位 <b>L1ALEV</b> 定义何时比较检查成功。 0 <sub>B</sub> 不请求中断。 1 <sub>B</sub> 请求中断。
<b>L1RSTEN</b>	6	rw	<b>电平阈值 1 复位请求使能</b> 该位域定义当比较检查成功时是否请求触发复位。 由位 <b>L1ALEV</b> 定义何时比较检查成功。 0 <sub>B</sub> 不请求复位。 1 <sub>B</sub> 请求复位。
<b>L1ASEN</b>	7	rw	<b>电平阈值 1 异步操作使能</b> 该位域定义当比较检查成功时是否执行异步操作。 由位 <b>L1ALEV</b> 定义何时比较检查成功。 0 <sub>B</sub> 不执行异步操作。 1 <sub>B</sub> 执行异步操作。
<b>LEV2V</b>	[10:8]	rw	<b>电平阈值 2 的电压</b> 该位域定义用来和 <b>DMP_M</b> 内核电压进行比较的电平阈值 2 的电压。 不同配置对应的电平阈值见数据手册。
<b>LEV2OK</b>	11	rh	<b>电平阈值 2 的检测结果</b> 0 <sub>B</sub> <b>DMP_M</b> 的内核供电电压小于电平阈值 2 的电压 <b>LEV2V</b> 。 1 <sub>B</sub> <b>DMP_M</b> 的内核供电电压等于或大于电平阈值 2 的电压 <b>LEV2V</b> 。
<b>L2ALEV</b>	12	rw	<b>电平阈值 2 生效电平</b> 0 <sub>B</sub> 当内核供电电压小于电平阈值 2 的电压 <b>LEV2V</b> 时，请求执行 <b>L2INTEN</b> 、 <b>L2RSTEN</b> 和 <b>L2ASEN</b> 所配置的操作。 1 <sub>B</sub> 当内核供电电压等于或大于电平阈值 2 的电压 <b>LEV2V</b> 时，请求执行 <b>L2INTEN</b> 、 <b>L2RSTEN</b> 和 <b>L2ASEN</b> 所配置的操作。

符号	位序号	类型	功能描述
<b>L2INTEN</b>	13	rw	<b>电平阈值 2 中断请求使能</b> 该位域定义当比较检查成功时是否请求触发中断。 由位 L2ALEV 定义何时比较检查成功。 0 <sub>B</sub> 不请求中断。 1 <sub>B</sub> 请求中断。
<b>L2RSTEN</b>	14	rw	<b>电平阈值 2 复位请求使能</b> 该位域定义当比较检查成功时是否请求触发复位。 由位 L2ALEV 定义何时比较检查成功。 0 <sub>B</sub> 不请求复位。 1 <sub>B</sub> 请求复位。
<b>L2ASEN</b>	15	rw	<b>电平阈值 2 异步操作使能</b> 该位域定义当比较检查成功时是否执行异步操作。 由位 L2ALEV 定义何时比较检查成功。 0 <sub>B</sub> 不执行异步操作。 1 <sub>B</sub> 执行异步操作。

### 8.6.3 控制内核域的电压电平

内核电压由两个**嵌入式电压调节器** (EVR) 控制。

内核域的电压由 EVR\_1 和 EVR\_M 控制。

#### 8.6.3.1 嵌入式电压调节器

器件的主要电路工作在 1.5V 电压下。该供电电压由嵌入式电压调节器 (EVR) 产生 (EVR 通过引脚供电)。外部需要加缓冲电容以稳定电压。

##### 特性概述

- 包括 0 在内的多个内核电压电平
- 基于高精度带隙产生内核电压
- EVR 关闭时可通过电容引脚由外部供电
- 内核限流

可通过 EVR 设置寄存器 EVRxSETyyV (x = M 或 1, yy = 15) 配置选择期望电压和参考电压值: 位域 VRSEL 用于选择电压电平及参考值; 位域 VLEV 用于对电压电平进行精调。EVR 使用高精度带隙 (GP) 作为参考。

可通过位域 **EVRxSETyyV.VLEV** 调整每种设置的 BG 电压，从而补偿应用和环境的影响。缺省设置 **VLEV** 或在每种器件的产品测试阶段调整 **VLEV** 以达到缺省的设置值。

### 高精度带隙 (HP)

系统的 HP 带隙具有以下用途：

- 为两个 **EVR** 提供非常稳定的参考电压
- 为 **Flash** 存储器提供正确的参考电压，详见 **flash** 存储器的描述。

**XE16xyM** 只实现了一个 HP 带隙。可由 **EVRMCON1.HPEN** 使能/禁用 HP 带隙。

### EVR 状态和控制寄存器

#### EV1CON0

**EV1CON0 寄存器**

**ESFR (F088H/44H)**

**复位值: DF20H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EVR DIS</b>	-	0	<b>CC DIS</b>	<b>CCLEV</b>	<b>LPR DIS</b>	1	0	<b>LPRLEV</b>			0				
rh	-	rw	rh	rw	rh	rw	rw	rw			r				

符号	位序号	类型	功能描述
<b>LPRLEV</b>	[5:3]	rw	<b>保留</b> 设置该寄存器时请勿改动该值。
<b>0</b>	[7:6]	rw	<b>保留</b> 设置该寄存器时请勿改动该值。
<b>1</b>	8	rw	<b>保留</b> 设置该寄存器时请勿改动该值。
<b>LPRDIS</b>	9	rh	<b>保留</b> 该值未限定。
<b>CCLEV</b>	[11:10]	rw	<b>当前控制电平</b> 使能/禁止当前控制 ( <b>CCDIS</b> ) 时需要设置该位域。 其有效值参见程序员指南。

符号	位序号	类型	功能描述
<b>CCDIS</b>	12	rh	<b>当前控制禁用</b> 0 <sub>B</sub> 当前控制使能。 1 <sub>B</sub> 当前控制禁用。 该位由 EVR1SETy.CCDIS 更新。
<b>0</b>	13	rw	<b>保留</b> 设置该寄存器时请勿改动该值。
<b>EVRDIS</b>	15	rh	<b>EVR_1 禁用</b> 0 <sub>B</sub> EVR_1 使能。 1 <sub>B</sub> EVR_1 禁用。 该位由 EVR1SETy.EVRDIS 更新。
<b>0</b>	[2:0]	r	<b>保留</b> 读操作返回 0；应写入 0。

### EVR1SET15VHP

**1.5V HP EVR\_1 设置寄存器**

**ESFR (F09E<sub>H</sub>/4F<sub>H</sub>)**

**复位值: 001B<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EVR DIS</b>	<b>0</b>	<b>CC DIS</b>	<b>0</b>	<b>LPR DIS</b>	<b>0</b>	<b>VRSEL</b>	<b>VLEV</b>								
rw	rw	rw	rw	rw	rw	rw	rw								

符号	位序号	类型	功能描述
<b>VLEV</b>	[5:0]	rw	<b>电压电平调整</b> 该位域调整 BG 电压，在每种器件的产品测试阶段调整 VLEV 以达到缺省目标值。 设置该寄存器时请勿改动该值。
<b>VRSEL</b>	[7:6]	rw	<b>电压参考选择</b> 00 <sub>B</sub> 选择 15VHP 高精度带隙的完整电压 01 <sub>B</sub> 保留，不使用该组合 10 <sub>B</sub> 保留，不使用该组合 11 <sub>B</sub> 保留，不使用该组合



系统控制单元 (SCU)

符号	位序号	类型	功能描述
			注：应始终向该位域写入复位值。
LPRDIS	9	rw	保留 设置该寄存器时请勿改动该值。
CCDIS	12	rw	当前控制禁用 0 <sub>B</sub> 当前控制使能。 1 <sub>B</sub> 当前控制禁用。 该位更新 EVR1CON0.CCDIS。 注：在关闭当前控制之前，EVR1CON0 中的 CCLEV 设置必须置为 00 <sub>B</sub> 。
EVRDIS	15	rw	EVR_1 禁用 0 <sub>B</sub> EVR_1 使能。 1 <sub>B</sub> EVR_1 禁用。 该位更新 EVR1CON0.EVRDIS。
0	8, [11:10], [14:13]	rw	保留 应写入 0。

EVRMCON0

EVR\_M 控制 0 寄存器

ESFR (F084<sub>H</sub>/42<sub>H</sub>)

复位值: 0D20<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVR DIS	0	CC DIS	CCLEV	LPR DIS	1	0	LPRLEV	0							
rh	r	rh	rw	rh	rw	rw	rw								r

符号	位序号	类型	功能描述
LPRLEV	[5:3]	rw	保留 设置该寄存器时请勿改动该值。
0	[7:6]	rw	保留 设置该寄存器时请勿改动该值。
1	8	rw	保留

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
			设置该寄存器时请勿改动该值。
<b>LPRDIS</b>	9	rh	<b>低功率参考禁用</b> 0 <sub>B</sub> LPR 使能。 1 <sub>B</sub> LPR 禁用。 该位由 EVRMSETy.LPRDIS 更新。
<b>CCLEV</b>	[11:10]	rw	<b>当前控制电平</b> 使能/禁止当前控制 (CCDIS) 时需要设置该位域。 其有效值参见程序员指南。
<b>CCDIS</b>	12	rh	<b>当前控制禁用</b> 0 <sub>B</sub> 当前控制使能。 1 <sub>B</sub> 当前控制禁用。 该位由 EVRMSETy.CCDIS 更新。
<b>EVRDIS</b>	15	rh	<b>EVR_M 禁用</b> 0 <sub>B</sub> EVR_M 使能。 1 <sub>B</sub> EVR_M 禁用。 该位由 EVRMSETy.EVRDIS 更新。
<b>0</b>	[2:0], [14:13]	rw	<b>保留</b> ;读操作返回 0; 应写入 0。

**EVRMCON1**

**EVR\_M 控制 1 寄存器**

**ESFR (F086<sub>H</sub>/43<sub>H</sub>)**

**复位值: 0101<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>							<b>HP EN</b>	<b>HPADJUST</b>							
r							rw	rw							

符号	位序号	类型	功能描述
<b>HPADJUST</b>	[7:0]	rw	<b>HP 带隙调整</b> 该位域存放 HP 带隙的调整值, 该调整值和器件型号有关。

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
			设置该寄存器时请勿改动该值。
<b>HPEN</b>	8	rw	<b>HP 带隙使能</b> 0 <sub>B</sub> HP 带隙禁用。 1 <sub>B</sub> HP 带隙使能。
<b>0</b>	[15:9]	r	<b>保留</b> 读操作返回 0；应写入 0。

**EVRMSET15VHP**

**1.5V HP EVR\_M 设置寄存器 ESFR (F096<sub>H</sub>/4B<sub>H</sub>)**

**复位值: 001B<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EVR DIS</b>	<b>0</b>	<b>CC DIS</b>	<b>0</b>	<b>LPR DIS</b>	<b>0</b>	<b>VRSEL</b>	<b>VLEV</b>								
rw	rw	rw	rw	rw	rw	rw	rw								

符号	位序号	类型	功能描述
<b>VLEV</b>	[5:0]	rw	<b>电压电平调整</b> 该位域调整 BG 电压，在每种器件的产品测试阶段调整 VLEV 以达到缺省目标值。 设置该寄存器时请勿改动该值。
<b>VRSEL</b>	[7:6]	rw	<b>电压参考选择</b> 00 <sub>B</sub> 选择 15VHP 高精度带隙的完整电压 01 <sub>B</sub> 保留，不使用该组合 10 <sub>B</sub> 保留，不使用该组合 11 <sub>B</sub> 保留，不使用该组合 <i>注：应始终向该位域写入复位值。</i>
<b>LPRDIS</b>	9	rw	<b>保留</b> 设置该寄存器时请勿改动该值。
<b>CCDIS</b>	12	rw	<b>当前控制禁用</b> 0 <sub>B</sub> 当前控制使能。 1 <sub>B</sub> 当前控制禁用。

符号	位序号	类型	功能描述
			该位更新 EVRMCON0.CCDIS。 <i>注：在关闭当前控制之前，EVRMCON0 中的 CCLEV 设置必须置为 00<sub>B</sub>。</i>
<b>0</b>	14	rw	<b>保留</b> 设置该寄存器时请勿改动该值。
<b>EVRDIS</b>	15	rw	<b>EVR_M 禁用</b> 0 <sub>B</sub> EVR_M 使能。 1 <sub>B</sub> EVR_M 禁用。 该位更新 EVRMCON0.EVRDIS。
<b>0</b>	8, [11:10], 13	rw	<b>保留</b> 应写入 0。

### 8.6.3.2 内核电压源

片上 EVR 可从（外部提供的）IO 电压产生 XE16xyM 的内核供电电压。

#### 由片上 EVR 产生的内核供电电压

最好通过片上 EVR 产生内核供电电压，因为这样可避免使用外部电压调节器。片上 EVR 由 V<sub>DDPB</sub> 供电。

正确操作 EVR 需要外部缓冲电容（其参数值见数据手册）。电流由片上旁路器件送出。

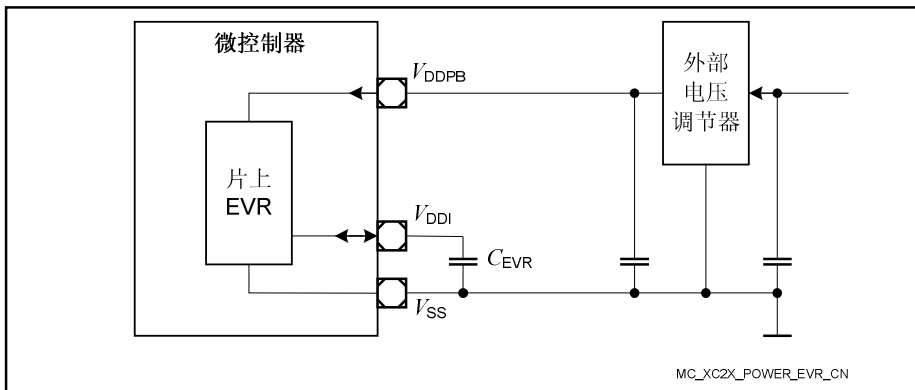


图 8-23 选择 EVR 产生内核供电电压

由片上资源产生内核供电电压可完全控制系统的省电模式，因此，若实际应用采用嵌入式供电机制（无需额外的外部电路），则可控制并使 XE16xyM 的功耗最小。

#### 8.6.4 控制供电系统

正确使用供电系统是省电的关键。为了最大程度降低功耗，可根据实际应用定义不同的操作状态。XE16xyM 支持以下省电机制：

- 降低系统性能
  - 功耗直接由系统频率决定
  - 系统性能由时钟操作机制控制
- 终止单个未使用的外设
  - 应用中使用不到的外设被禁用
  - 模块操作由寄存器 MOD\_KSCCFG 控制
- 终止多个未使用的外设
  - 应用中使用不到的外设被禁用
  - 系统外设操作由全局状态控制器（GSC）控制
- 终止单个未使用的模拟部分
  - 应用中使用不到的模拟部分终止工作
  - 操作由 SCU（PLL、时钟、PVC、SWD、温度补偿）或 ADC 中的寄存器控制

## 8.7 全局状态控制器 (GSC)

除省电模式和时钟管理之外，系统外设的模式控制提供了另一种根据应用需求配置系统的方式。

通过全局状态控制器 (GSC) 实现模式控制。允许用户根据实际应用的需求通过 GSC 便捷的配置操作模式。

### 特性概述

GSC 用于以下控制：

- 外设时钟操作的控制
- 调试的挂起控制
- 不同请求源的仲裁

根据来自 OCDS、SWD 预警检测或其它模块的请求，GSC 进行内部优先级排序并将结果作为命令请求广播给所有外设。GSC 对请求源进行内部优先级排序的机制如表 8-8 所示。

### 8.7.1 GSC 控制流

当 SoC 中至少有一个请求源有效以请求模式改变时，进入控制序列。若多个请求同时挂起，则通过仲裁机制进行处理。请求源不由 GSC 保存，因此请求源必须保持有效直至其触发信号失效。

若请求当前仍挂起，或系统已进入并应答命令请求，请求源保持有效状态。GSC 和外设基于命令进行通信。共定义了三种命令，分别请求三种模式：

- 唤醒命令：请求正常模式
- 时钟关闭命令：请求终止模式
- 调试命令：请求挂起模式

各个外设在这三种模式下的特定操作由模块寄存器 `mod_KSCCFG` 定义。

#### 8.7.1.1 请求源仲裁

仲裁的最高优先级为 0（见表 8-8）。

每个系统时钟周期启动一轮新的仲裁。仲裁胜出的请求源请求下一个命令。需要注意，仲裁胜出不会自动产生新的命令。只有当 SoC 中当前没有命令在广播时，才能产生并广播新的命令。若本轮和前一轮仲裁胜出的请求源相同、或未检测到胜出请求源，则不产生新的命令请求。

**表 8-8 请求源的优先级**

请求源	优先级
OCDS 退出	4
ESR0	5
ESR1	6
ESR2	7
WUT	8
ITC	9
GPT12E	10
SW1	11
SW2	12
OCDS 进入	14

### 8.7.1.2 新命令的产生

若当前没有命令被广播，当检测到新的请求源且仲裁胜出，产生新的命令请求。

**表 8-9 请求源和对应的命令请求**

请求源	命令说明
OCDS 退出	唤醒；正常模式
ESR0	唤醒；正常模式
ESR1	唤醒；正常模式
ESR2	时钟关闭；终止模式
WUT	唤醒；正常模式
ITC	唤醒；正常模式
GPT12E	唤醒；正常模式
SW1	唤醒；正常模式
SW2	时钟关闭；终止模式
OCDS 进入	调试；挂起模式

### 8.7.1.3 命令的使用

对各从控模块不同操作模式的完整控制机制可分为两部分：

- 中央控制和配置部分：全局状态控制器 (GSC)
- 每个从控模块的本地控制：内核状态控制器 (KSC)

不同的硬件（如 WUT 或 OCDS）或软件请求源可通过 GSC 请求系统进入某特定模式。受该模式影响的部分可由各模块的 KSC 预先设定。不同模块对每种命令请求的响应操作可由各自的 KSC 预先配置。

*注：清零 `mod_KSCCFG.MODEN` 请求某外设永久关闭不启动 GSC 工作。不过，若关闭该外设和触发 GSC 运行同时发生，只要该外设未发送应答或 `GSCPERSTATEN` 中的相应位被清零，GSC 的有限状态机将会等待该外设的应答。*

*建议用户通过相应置位 `GSCEN` 禁止 GSC 自动运行（若应用需要外设的关闭应答信息）、或禁用 `GSCPERSTATEN` 中的相应位，从而忽略缺少的应答。*

*注：当已成功进入某 GSC 模式，GSC 仲裁器可接收任何新请求。若新的请求源也请求进入当前模式，该请求被置于流水线中、保持挂起。*

*建议由软件请求命令。特别是，应由 SW2 触发时钟关闭命令。使用由硬件请求的命令时必须谨慎。只应选择请求正常模式的硬件资源。若软件已检测到唤醒请求，则可通过清除 `GSCEN` 中所选资源对应的控制位、随后重新使能 `GSCEN` 去除挂起模式修改请求。*

### 8.7.1.4 终止请求源

若请求源被禁用或其使能位被清零，该请求源失效，不再对其进行仲裁。

### 8.7.1.5 挂起控制流

由 OCDS 模块控制挂起特性。GSC 仅作为系统的控制和通信接口。挂起特性由两种条件来产生：

请求挂起模式时必须进入的模式。

退出挂起模式时必须进入的模式。

OCDS 产生进入挂起模式的请求。一旦请求挂起模式，则期望系统尽快停止工作，进入空闲状态（内部无进程挂起）；要求进入该系统状态的方式不会对内部或外部造成任何破坏、也能够无任何破坏的退出该状态。因此，要求系统中的所有外设进入时钟可关闭的模式。通过发送调试命令来实现。

调试是非破坏性操作，因此，退出挂起模式后不应只进入一种专用的系统模式，而应进入转入挂起模式之前的系统模式。GSC 检测到挂起模式请求时保存系统模式；检测到退出挂起模式请求时，该系统模式用作目标系统模式。



### 8.7.1.6 模式转换的错误反馈

当至少有一个外设报错时，寄存器 GSCSTAT 中的错误标志置位。若请求新的系统模式时 GSC 未检测到出错，错误标志被清零。可产生中断向系统汇报该错误状态。

## 8.7.2 GSC 寄存器

### 8.7.2.1 GSC 控制和状态寄存器

以下寄存器用于控制和配置 GSC。

#### GSCSWREQ

##### GSC 软件请求寄存器

**SFR (FF14<sub>H</sub>/8A<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														SW TRG2	SW TRG1
r														rwh	rwh

符号	位序号	类型	功能描述
<b>SWTRG1</b>	0	rwh	<b>软件请求 1 (SW1)</b> 0 <sub>B</sub> 不产生 SW1 请求。 1 <sub>B</sub> 产生 SW1 请求。 若 SW1 请求在仲裁中胜出并向系统广播，该位被自动清零。
<b>SWTRG2</b>	1	rwh	<b>软件请求 2 (SW2)</b> 0 <sub>B</sub> 不产生 SW2 请求。 1 <sub>B</sub> 产生 SW2 请求。 若 SW2 请求在仲裁中胜出并向系统广播，该位被自动清零。
<b>0</b>	[15:2]	r	<b>保留</b> 读操作返回 0；应写入 0。

**GSCEN**

**GSC 使能寄存器**

**SFR (FF16<sub>H</sub>/8B<sub>H</sub>)**

**复位值: 7FFF<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	OCDS ENEN	1	SW2 EN	SW1 EN	GPT EN	ITC EN	WUT EN	ESR2 EN	ESR1 EN	ESR0 EN	OCDS EXEN	PSCA ENEN	PSCA EXEN	PSCB ENEN	PSCB EXEN
r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
<b>PSCBEXEN</b>	0	rw	保留 必须写入 0 <sub>B</sub> 。
<b>PSCBENEN</b>	1	rw	保留 必须写入 0 <sub>B</sub> 。
<b>PSCAEXEN</b>	2	rw	保留 必须写入 0 <sub>B</sub> 。
<b>PSCAENEN</b>	3	rw	保留 必须写入 0 <sub>B</sub> 。
<b>OCDSEXEN</b>	4	rw	<b>OCDS 退出请求使能</b> 0 <sub>B</sub> 禁用 OCDS 退出请求。 1 <sub>B</sub> 使能 OCDS 退出请求。
<b>ESR0EN</b>	5	rw	<b><math>\overline{\text{ESR0}}</math> 请求使能</b> 0 <sub>B</sub> 禁用 $\overline{\text{ESR0}}$ 请求。 1 <sub>B</sub> 使能 $\overline{\text{ESR0}}$ 请求。
<b>ESR1EN</b>	6	rw	<b><math>\overline{\text{ESR1}}</math> 请求使能</b> 0 <sub>B</sub> 禁用 $\overline{\text{ESR1}}$ 请求。 1 <sub>B</sub> 使能 $\overline{\text{ESR1}}$ 请求。
<b>ESR2EN</b>	7	rw	<b><math>\overline{\text{ESR2}}</math> 请求使能</b> 0 <sub>B</sub> 禁用 $\overline{\text{ESR2}}$ 请求。 1 <sub>B</sub> 使能 $\overline{\text{ESR2}}$ 请求。

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
<b>WUTEN</b>	8	rw	<b>WUT 请求使能</b> 0 <sub>B</sub> 禁用 WUT 请求。 1 <sub>B</sub> 使能 WUT 请求。
<b>ITCEN</b>	9	rw	<b>ITC 请求使能</b> 0 <sub>B</sub> 禁用 ITC 请求。 1 <sub>B</sub> 使能 ITC 请求。
<b>GPTEN</b>	10	rw	<b>GPT12E 请求使能</b> 0 <sub>B</sub> 禁用 GPT12E 请求。 1 <sub>B</sub> 使能 GPT12E 请求。
<b>SW1EN</b>	11	rw	<b>SW1 请求使能</b> 0 <sub>B</sub> 禁用 SW1 请求。 1 <sub>B</sub> 使能 SW1 请求。
<b>SW2EN</b>	12	rw	<b>SW2 请求使能</b> 0 <sub>B</sub> 禁用 SW2 请求。 1 <sub>B</sub> 使能 SW2 请求。
<b>1</b>	13	rw	<b>保留</b> 读取返回 1；应写入 1。
<b>OCDS ENEN</b>	14	rw	<b>OCDS 进入请求使能</b> 0 <sub>B</sub> 禁用 OCDS 进入请求。 1 <sub>B</sub> 使能 OCDS 进入请求。
<b>0</b>	15	r	<b>保留</b> 读操作返回 0；应写入 0。

**GSCSTAT**

**GSC 状态寄存器**

**SFR (FF18<sub>H</sub>/8C<sub>H</sub>)**

**复位值: 3C00<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		SOURCE				PEN	ERR	0		NEXT		0		CURRENT	
r		rh				rh	rh	r		rh		r		rh	

符号	位序号	类型	功能描述
<b>CURRENT</b>	[1:0]	rh	<b>当前使用的命令</b> 该位域指示当前所使用的系统模式。
<b>NEXT</b>	[5:4]	rh	<b>下一个使用的命令</b> 该位域指示将使用的下一种系统模式。
<b>ERR</b>	8	rh	<b>错误状态标志</b> 该标志位指示向系统广播的上一个命令是否被应答至少有一个错误。当广播新命令时，该位被自动清零。
<b>PEN</b>	9	rh	<b>命令挂起标志</b> 该标志位指示当前是否有命令挂起。命令被广播后，只要还有模块未完成命令所请求的操作、未作出应答，该命令将保持挂起。
<b>SOURCE</b>	[13:10]	rh	<b>请求源状态</b> 该位域监控触发上次请求的请求源。 0000 <sub>B</sub> 保留 0001 <sub>B</sub> 保留 0010 <sub>B</sub> 保留 0011 <sub>B</sub> 保留 0100 <sub>B</sub> OCDS 退出 0101 <sub>B</sub> $\overline{\text{ESR0}}$ 0110 <sub>B</sub> $\overline{\text{ESR1}}$ 0111 <sub>B</sub> $\overline{\text{ESR2}}$ 1000 <sub>B</sub> WUT 1001 <sub>B</sub> ITC 1010 <sub>B</sub> GPT12E 1011 <sub>B</sub> SW1 1100 <sub>B</sub> SW2 1101 <sub>B</sub> 保留，不使用该组合 1110 <sub>B</sub> OCDS 进入 1111 <sub>B</sub> 保留，不使用该组合

符号	位序号	类型	功能描述
<b>0</b>	[3:2], [7:6], [15:14]	r	<b>保留</b> 读操作返回 0；应写入 0。

## GSCPERSTATEN

### GSC 外设状态使能寄存器

**SFR (FF04<sub>H</sub>/82<sub>H</sub>)**

**复位值: FFFF<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>USIC 3</b>	<b>USIC 2</b>	<b>USIC 1</b>	<b>USIC 0</b>	<b>FL</b>	<b>MEM</b>	<b>RTC</b>	<b>CCU 63</b>	<b>CCU 62</b>	<b>CCU 61</b>	<b>CCU 60</b>	<b>M CAN</b>	<b>CC2</b>	<b>1</b>	<b>GPT 12E</b>	<b>ADC</b>
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
<b>ADC</b>	0	rw	<b>ADC 应答使能</b> 该位域定义是否使用 ADC 模块的应答状态（考虑在内或忽略）。 0 <sub>B</sub> 不使用 ADC 模块应答，始终被当作有效。 1 <sub>B</sub> 使用 ADC 模块应答。
<b>GPT12E</b>	1	rw	<b>GPT12E 应答使能</b> 该位域定义是否使用 GPT12E 模块的应答状态（考虑在内或忽略）。 0 <sub>B</sub> 不使用 GPT12E 模块应答，始终被当作有效。 1 <sub>B</sub> 使用 GPT12E 模块应答。
<b>1</b>	2	rw	<b>保留</b> 读取返回 1；应写入 1。
<b>CC2</b>	3	rw	<b>CC2 应答使能</b> 该位域定义是否使用 CC2 模块的应答状态（考虑在内或忽略）。 0 <sub>B</sub> 不使用 CC2 模块应答，始终被当作有效。 1 <sub>B</sub> 使用 CC2 模块应答。
<b>MCAN</b>	4	rw	<b>MultiCAN 应答使能</b> 该位域定义是否使用 MultiCAN 模块的应答状态

符号	位序号	类型	功能描述
			<p>(考虑在内或忽略)。</p> <p>0<sub>B</sub> 不使用 MultiCAN 模块应答, 始终被当作有效。</p> <p>1<sub>B</sub> 使用 MultiCAN 模块应答。</p>
<b>CCU60</b>	5	rw	<p><b>CCU60 应答使能</b></p> <p>该位域定义是否使用 CCU60 模块的应答状态 (考虑在内或忽略)。</p> <p>0<sub>B</sub> 不使用 CCU60 模块应答, 始终被当作有效。</p> <p>1<sub>B</sub> 使用 CCU60 模块应答。</p>
<b>CCU61</b>	6	rw	<p><b>CCU61 应答使能</b></p> <p>该位域定义是否使用 CCU61 模块的应答状态 (考虑在内或忽略)。</p> <p>0<sub>B</sub> 不使用 CCU61 模块应答, 始终被当作有效。</p> <p>1<sub>B</sub> 使用 CCU61 模块应答。</p>
<b>CCU62</b>	7	rw	<p><b>CCU62 应答使能</b></p> <p>该位域定义是否使用 CCU62 模块的应答状态 (考虑在内或忽略)。</p> <p>0<sub>B</sub> 不使用 CCU62 模块应答, 始终被当作有效。</p> <p>1<sub>B</sub> 使用 CCU62 模块应答。</p>
<b>CCU63</b>	8	rw	<p><b>CCU63 应答使能</b></p> <p>该位域定义是否使用 CCU63 模块的应答状态 (考虑在内或忽略)。</p> <p>0<sub>B</sub> 不使用 CCU63 模块应答, 始终被当作有效。</p> <p>1<sub>B</sub> 使用 CCU63 模块应答。</p>
<b>RTC</b>	9	rw	<p><b>RTC 应答使能</b></p> <p>该位域定义是否使用 RTC 模块的应答状态 (考虑在内或忽略)。</p> <p>0<sub>B</sub> 不使用 RTC 模块应答, 始终被当作有效。</p> <p>1<sub>B</sub> 使用 RTC 模块应答。</p>
<b>MEM</b>	10	rw	<p><b>C166SV2 子系统应答使能</b></p> <p>该位域定义是否使用 C166SV2 子系统模块的应答状态 (考虑在内或忽略)。</p>

符号	位序号	类型	功能描述
			<p>0<sub>B</sub> 不使用 C166SV2 子系统模块应答，始终被当作有效。</p> <p>1<sub>B</sub> 使用 C166SV2 子系统模块应答。</p>
<b>FL</b>	11	rw	<p><b>Flash 应答使能</b></p> <p>该位域定义是否使用 Flash 模块的应答状态（考虑在内或忽略）。</p> <p>0<sub>B</sub> 不使用 Flash 模块应答，始终被当作有效。</p> <p>1<sub>B</sub> 使用 Flash 模块应答。</p>
<b>USIC0</b>	12	rw	<p><b>USIC0 应答使能</b></p> <p>该位域定义是否使用 USIC0 模块的应答状态（考虑在内或忽略）。</p> <p>0<sub>B</sub> 不使用 USIC0 模块应答，始终被当作有效。</p> <p>1<sub>B</sub> 使用 USIC0 模块应答。</p>
<b>USIC1</b>	13	rw	<p><b>USIC1 应答使能</b></p> <p>该位域定义是否使用 USIC1 模块的应答状态（考虑在内或忽略）。</p> <p>0<sub>B</sub> 不使用 USIC1 模块应答，始终被当作有效。</p> <p>1<sub>B</sub> 使用 USIC1 模块应答。</p>
<b>USIC2</b>	14	rw	<p><b>USIC2 应答使能</b></p> <p>该位域定义是否使用 USIC2 模块的应答状态（考虑在内或忽略）。</p> <p>0<sub>B</sub> 不使用 USIC2 模块应答，始终被当作有效。</p> <p>1<sub>B</sub> 使用 USIC2 模块应答。</p>
<b>USIC3</b>	15	rw	<p><b>USIC3 应答使能</b></p> <p>该位域定义是否使用 USIC3 模块的应答状态（考虑在内或忽略）。</p> <p>0<sub>B</sub> 不使用 USIC3 模块应答，始终被当作有效。</p> <p>1<sub>B</sub> 使用 USIC3 模块应答。</p>

## GSCPERSTAT

### GSC 外设状态寄存器

**SFR (FF1A<sub>H</sub>/8D<sub>H</sub>)**

**复位值: FFFF<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>USIC 3</b>	<b>USIC 2</b>	<b>USIC 1</b>	<b>USIC 0</b>	<b>FL</b>	<b>MEM</b>	<b>RTC</b>	<b>CCU 63</b>	<b>CCU 62</b>	<b>CCU 61</b>	<b>CCU 60</b>	<b>M CAN</b>	<b>CC2</b>	<b>-</b>	<b>GPT 12E</b>	<b>ADC</b>
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	-	rh	rh

符号	位序号	类型	功能描述
<b>ADC</b>	0	rh	<b>ADC 应答状态</b> 该位指示 ADC 模块的应答状态。 0 <sub>B</sub> ADC 模块当前正在修改其内核状态。其应答还未接收到。 1 <sub>B</sub> 使用 ADC 模块应答, 已被接收到或不被使用。
<b>GPT12E</b>	1	rh	<b>GPT12E 应答状态</b> 该位指示 GPT12E 模块的应答状态。 0 <sub>B</sub> GPT12E 模块当前正在修改其内核状态。其应答还未接收到。 1 <sub>B</sub> 使用 GPT12E 模块应答, 已被接收到或不被使用。
<b>CC2</b>	3	rh	<b>CC2 应答状态</b> 该位指示 CC2 模块的应答状态。 0 <sub>B</sub> CC2 模块当前正在修改其内核状态。其应答还未接收到。 1 <sub>B</sub> 使用 CC2 模块应答, 已被接收到或不被使用。
<b>MCAN</b>	4	rh	<b>MultiCAN 应答状态</b> 该位指示 MultiCAN 模块的应答状态。 0 <sub>B</sub> MultiCAN 模块当前正在修改其内核状态。其应答还未接收到。 1 <sub>B</sub> 使用 MultiCAN 模块应答, 已被接收到或不被使用。
<b>CCU60</b>	5	rh	<b>CCU60 应答状态</b> 该位指示 CCU60 模块的应答状态。 0 <sub>B</sub> CCU60 模块当前正在修改其内核状态。其应



符号	位序号	类型	功能描述
			<p>答还未接收到。</p> <p><b>1<sub>B</sub></b> 使用 CCU60 模块应答, 已被接收到或不被使用。</p>
<b>CCU61</b>	6	rh	<p><b>CCU61 应答状态</b></p> <p>该位指示 CCU61 模块的应答状态。</p> <p><b>0<sub>B</sub></b> CCU61 模块当前正在修改其内核状态。其应答还未接收到。</p> <p><b>1<sub>B</sub></b> 使用 CCU61 模块应答, 已被接收到或不被使用。</p>
<b>CCU62</b>	7	rh	<p><b>CCU62 应答状态</b></p> <p>该位指示 CCU62 模块的应答状态。</p> <p><b>0<sub>B</sub></b> CCU62 模块当前正在修改其内核状态。其应答还未接收到。</p> <p><b>1<sub>B</sub></b> 使用 CCU62 模块应答, 已被接收到或不被使用。</p>
<b>CCU63</b>	8	rh	<p><b>CCU63 应答状态</b></p> <p>该位指示 CCU63 模块的应答状态。</p> <p><b>0<sub>B</sub></b> CCU63 模块当前正在修改其内核状态。其应答还未接收到。</p> <p><b>1<sub>B</sub></b> 使用 CCU63 模块应答, 已被接收到或不被使用。</p>
<b>RTC</b>	9	rh	<p><b>RTC 应答状态</b></p> <p>该位指示 RTC 模块的应答状态。</p> <p><b>0<sub>B</sub></b> RTC 模块当前正在修改其内核状态。其应答还未接收到。</p> <p><b>1<sub>B</sub></b> 使用 RTC 模块应答, 已被接收到或不被使用。</p>
<b>MEM</b>	10	rh	<p><b>C166SV2 子系统应答状态</b></p> <p>该位指示 C166SV2 子系统模块的应答状态。</p> <p><b>0<sub>B</sub></b> C166SV2 子系统模块当前正在修改其内核状态。其应答还未接收到。</p> <p><b>1<sub>B</sub></b> 使用 C166SV2 子系统模块应答, 已被接收到或不被使用。</p>

符号	位序号	类型	功能描述
<b>FL</b>	11	rh	<b>Flash 应答状态</b> 该位指示 Flash 模块的应答状态。 <b>0<sub>B</sub></b> Flash 模块当前正在修改其内核状态。其应答还未接收到。 <b>1<sub>B</sub></b> 使用 Flash 模块应答, 已被接收到或不被使用。
<b>USIC0</b>	12	rh	<b>USIC0 应答状态</b> 该位指示 USIC0 模块的应答状态。 <b>0<sub>B</sub></b> USIC0 模块当前正在修改其内核状态。其应答还未接收到。 <b>1<sub>B</sub></b> 使用 USIC0 模块应答, 已被接收到或不被使用。
<b>USIC1</b>	13	rh	<b>USIC1 应答状态</b> 该位指示 USIC1 模块的应答状态。 <b>0<sub>B</sub></b> USIC1 模块当前正在修改其内核状态。其应答还未接收到。 <b>1<sub>B</sub></b> 使用 USIC1 模块应答, 已被接收到或不被使用。
<b>USIC2</b>	14	rh	<b>USIC2 应答状态</b> 该位指示 USIC2 模块的应答状态。 <b>0<sub>B</sub></b> USIC2 模块当前正在修改其内核状态。其应答还未接收到。 <b>1<sub>B</sub></b> 使用 USIC2 模块应答, 已被接收到或不被使用。
<b>USIC3</b>	15	rh	<b>USIC3 应答状态</b> 该位指示 USIC3 模块的应答状态。 <b>0<sub>B</sub></b> USIC3 模块当前正在修改其内核状态。其应答还未接收到。 <b>1<sub>B</sub></b> 使用 USIC3 模块应答, 已被接收到或不被使用。

若使用模块 x 的应答，其应答已被接收或不相关时，应答状态位置位。若不使用模块 x 的应答、或模块 x 当前未修改其内核状态，模块 x 的应答无关，此时假定已接收模块应答。

## 8.8 软件 Boot 支持

为了决定正确的软件操作起始点，需要最基本硬件资源的支持。尽量通过软件决定何时开始执行某操作。但有些决定必须在软件运行之前获知，这些决定必须由硬件完成。

### 8.8.1 启动寄存器

#### 8.8.1.1 启动状态寄存器

寄存器 STSTAT 中存放 boot 软件所需的信息，用于确定（可选用的）不同的启动设置。

#### STSTAT

启动状态寄存器					MEM (F1E0 <sub>H</sub> --)								复位值: 8000 <sub>H</sub>		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
1	0							HWCFG							
r					r					rh					

符号	位序号	类型	功能描述
HWCFG	[7:0]	rh	<b>硬件配置设置</b> 该位域存放 boot 软件使用的值。 应用复位时，该位域由寄存器 SWRSTCON.SWCFG 的内容更新（若 SWRSTCON.SWBOOT 置位）。
0	[14:8]	r	<b>保留</b> 读操作返回 0；应写入 0。
1	15	r	<b>保留</b> 读操作返回 1；应写入 1。

## 8.9 外部请求单元 (ERU)

外部请求单元 (ERU) 是通用的事件和序列检测单元。其主要任务是**基于不同输入上可选的触发事件产生中断**，如某输入引脚上发生信号跳变时产生外部中断请求。

检测到的事件也可被其它模块用来触发或门控特定的模块操作（如 ADC 模块的转换操作）。

### 8.9.1 概述

XE16xyM 中的 ERU 可分为三个主要的功能部分：

- 4 路独立的**输入通道 x**，选择输入和定义触发或门控功能。
- 事件分配：**连接矩阵**定义引发输出通道 y 响应的输入通道 x 的事件。
- 4 路独立的**输出通道 y**，用于组合事件，定义它们的作用及向系统的分配（中断产生、ADC 转换触发）。

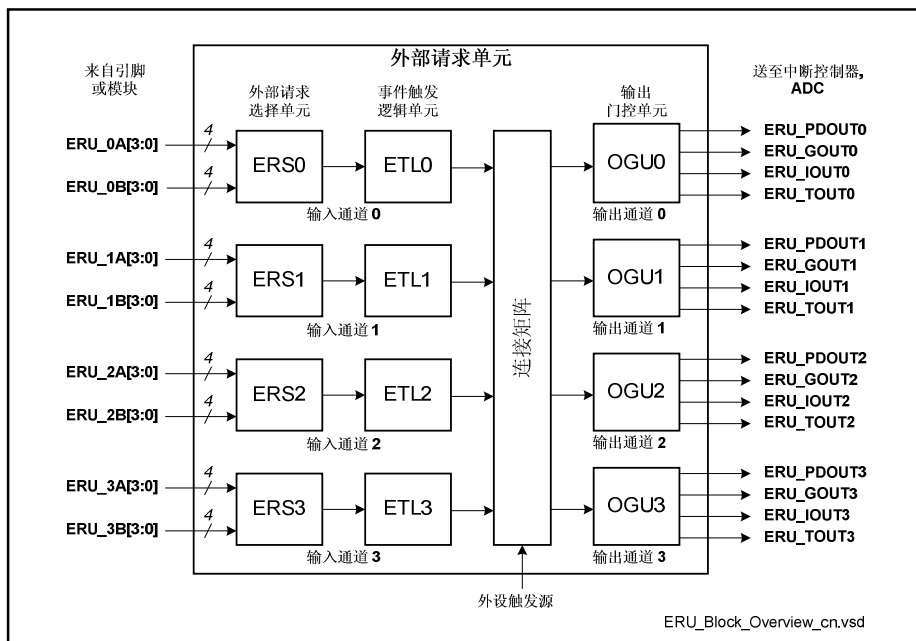


图 8-24 外部请求源单元

这些任务由以下模块处理：

- 每路输入通道的**外部请求选择单元 (ERS)** 可从两个输入信号 (ERU\_xA、ERU\_xB) 中选择一个、或将它们逻辑组合生成一个触发信号。这两个信号每个都有 4 种输入可选 (如：实际输入 ERU\_xA 可从 ERU\_xA[3:0] 中选择；实际输入 ERU\_xB 可从 ERU\_xB[3:0] 中选择)。
- 每路输入通道的**事件触发逻辑 (ETLx)** 可定义导致触发事件的转换 (跳变沿选择、或由软件控制) 并可保存该状态。这里，所选信号的输入电平被翻译为事件 (检测到事件 = 事件标志位变为置位，和原始输入信号的极性无关)。
- **连接矩阵** 将输入通道产生的事件和状态标志分配给输出通道。此外，来自其它模块 (如 CC2) 的一些外设触发信号也可用，它们可和 ERU 输入通道产生的触发信号逻辑组合。
- 每路输出通道的**输出门控单元 (OGUy)** 将来自输入通道的触发事件和状态信息相组合。一路输入通道的单个事件可引发多路输出通道的响应；同样，多路输入通道的事件可组合引发一路输出通道的响应 (序列检测)。

支持不同类型的响应，如产生中断 (基于信号 ERU\_IOUTy)、触发 ADC 转换 (基于信号 ERU\_TOUTy)、门控 ADC 转换 (基于信号 ERU\_GOUTy)。

ERU 由图 8-25 所示的多个寄存器控制，寄存器描述见章节 8.9.8。

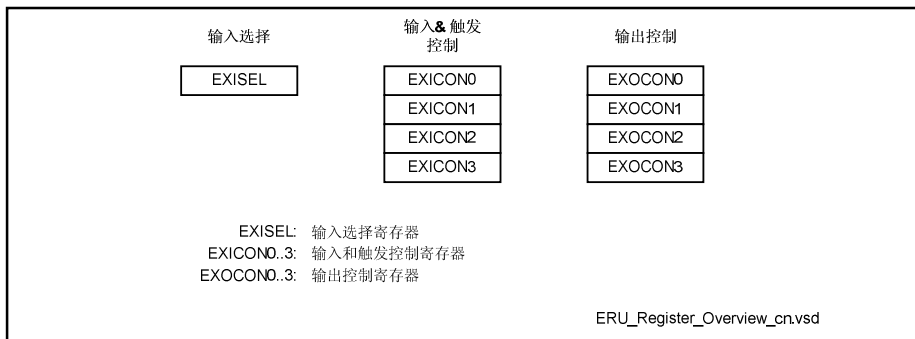


图 8-25 ERU 寄存器概览

## 8.9.2 ERU 输入连接

ERU 输入可从众多输入信号中选择：一些可直接来自于引脚，另一些使用来自不同外设模块 (如 USIC 和 MultiCAN 模块，USIC 模块的信号前缀 UxCy 指示通信通道编号) 的信号，这些信号是已被选择用作 USIC 或 MultiCAN 功能的引脚上的输入信号，输入信号的选择在 USIC 或 MultiCAN 模块内部完成。

ERU 输入连接的具体描述见章节 8.17.3.1。

通常，当 USIC 或 MultiCAN 模块不使用其输入功能或这些外设模块完全关闭时，可将其输入信号用于 ERU 功能。不过，若某输入信号被 USIC 或 MultiCAN 模块使用，该信号也可被 ERU 使用，用来提供某个触发功能并最终和其它信号相组合（如，检测到一帧开始时产生中断请求）。

基于这样的结构，输入引脚数大大增加，因为不仅使用 ERU 的选择功能，还使用通信模块的选择功能。

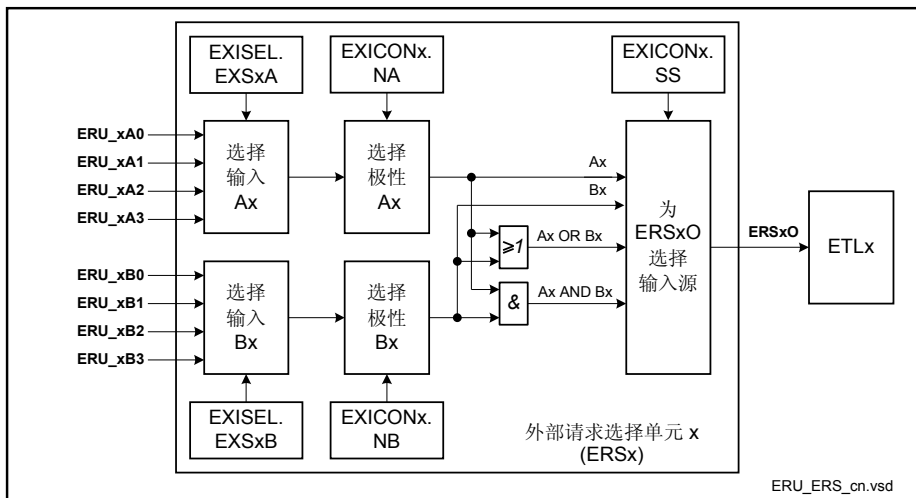
*注：ERU 的所有功能输入在送入内部电路之前需要和  $f_{sys}$  同步。这会导致  $2/f_{sys}$  的延迟和  $1/f_{sys}$  的不确定误差，在进行时序计算时必须考虑在内。*

*只有当输入信号的高电平和低电平平均长于  $1/f_{sys}$ ，才能正确检测到输入信号的跳变。*

### 8.9.3 外部请求选择单元 (ERSx)

对于每路输入通道  $x$  ( $x = 0-3$ )，ERSx 单元为关联的 ETLx 单元选择输入信号。每个 ERSx 将 Ax、Bx 两个信号逻辑组合，向关联的 ETLx 单元提供最终输出 ERSxO。Ax 可从 4 个输入 ERU\_xA[3:0] 中选择（还可选择输入反相）；Bx 可从 4 个输入 ERU\_xB[3:0] 中选择（也可选择输入反相）。

除了可直接选择输入 Ax、Bx、或其反相值，还可选择 Ax 和 Bx 逻辑或、逻辑与。



**图 8-26 外部请求选择单元**

ERS 单元由寄存器 **EXISEL** (一个寄存器控制所有 ERSx 单元) 和寄存器 **EXICONx** (一个寄存器控制一个 ERSx 和 ETLx 单元, 如 **EXICON0** 对应输入通道 0) 控制。

#### 8.9.4 事件触发逻辑 (ETLx)

对于每路输入通道  $x$  ( $x = 0-3$ ), 事件触发逻辑 ETLx 从输入 ERSxO (由关联的 ERSx 单元产生) 得到触发事件和状态。每个 ETLx 包含一个跳变沿检测模块, 可单独使能上升沿或下降沿检测。若上下沿使能位均置位, 双沿均会产生触发事件 (如处理翻转输入)。

每个 ETLx 单元有一个和它关联的、控制 ETLx 所有选择的 EXICONx 寄存器 (该寄存器还控制关联的 ERSx 单元, 如 **EXICON0** 控制 ESR0 和 ETL0)。

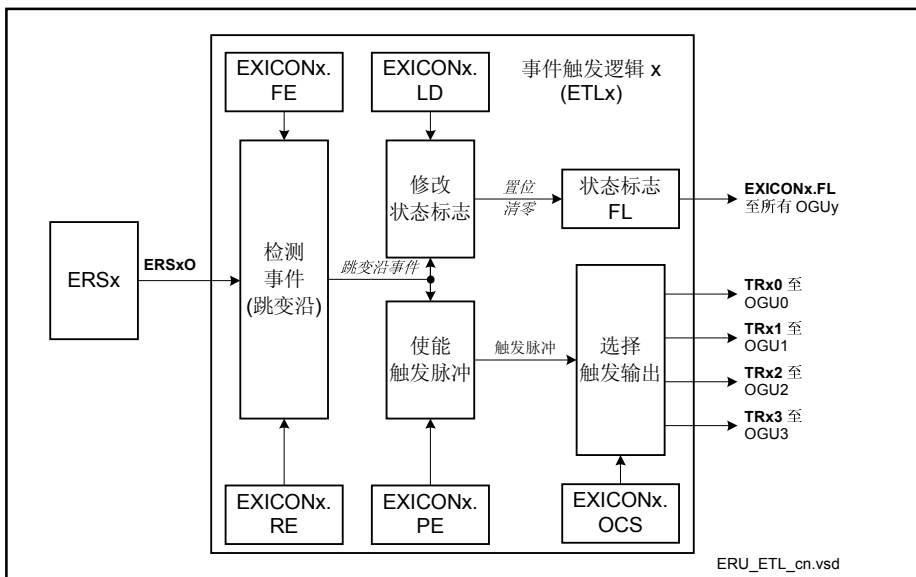


图 8-27 事件触发逻辑

检测到所选择的事件 (跳变沿) 后, 状态标志 **EXICONx.FL** 变为置位。该标志也可由软件修改 (置位或清零), 它支持两种不同的操作模式。

它可用作“粘着”标志, 检测到期望事件后由硬件置位、必须由软件清零。在该操作模式下, 该标志指示某事件已发生, 但不指示输入的实际状态。

在另一种操作模式下，检测到“相反”的事件时标志被自动清零。比如，若只允许下降沿跳变置位状态标志，检测到上升沿时标志被清零。可使用该模式进行序列检测（该模式不使用双沿检测）。

状态标志的输出和所有输出门控单元（OGUy）并行连接（见图 8-28），所有 OGUy 单元根据不同或相同的状态标志进行序列检测。

除修改状态标志之外，可使能（由 EXICONx.PE 控制）并选择 ETLx 的触发脉冲输出 TRxy，用于触发某个 OGUy 单元的操作。目标 OGUy 由位域 EXICON.OCS 选择。

检测到所选择的跳变沿事件时触发脉冲被激活，和状态标志 EXICONx.FL 无关。

### **8.9.5 连接矩阵**

连接矩阵将不同 ETLx 单元输出的触发信号（TRxy）和状态信号（EXICON.FL）分配给各 OGUy 单元。此外，它还接收外设触发信号，这些信号可在 OGUy 单元中和 ETLx 触发信号进行逻辑或。ETLx 和 OGUy 单元之间的完整连接如图 8-28 所示。



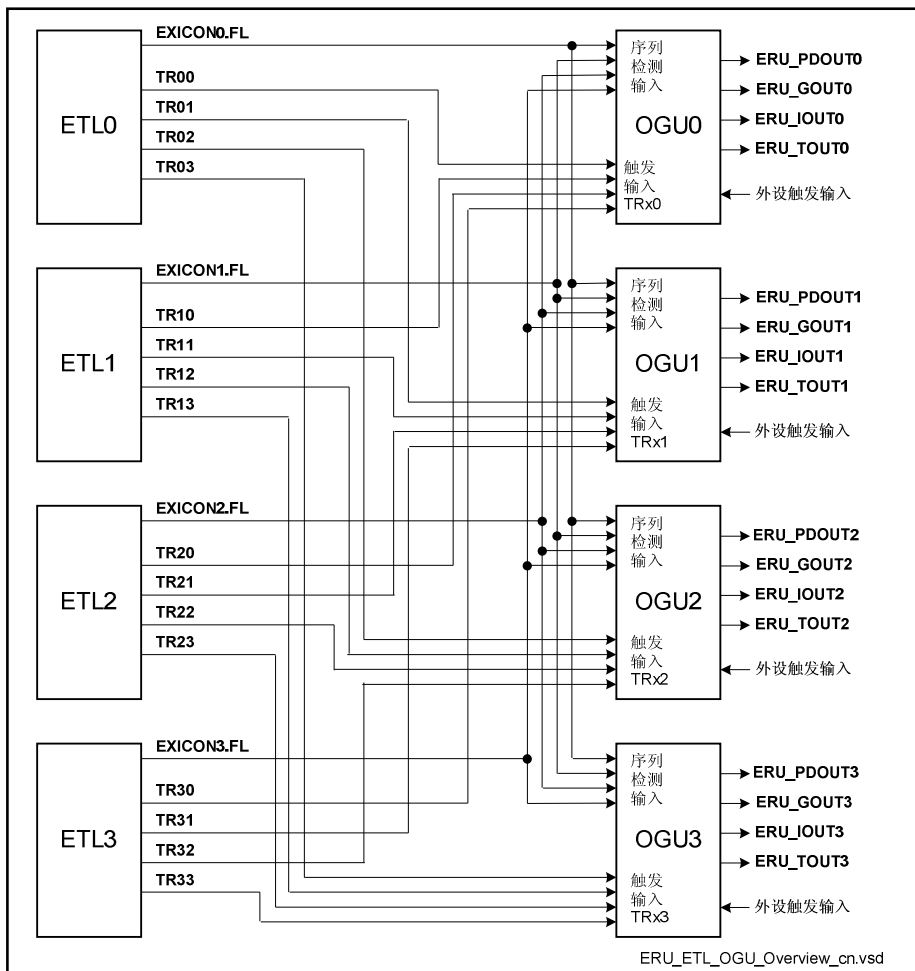


图 8-28 ETLx 和 OGUy 之间的连接矩阵

### 8.9.6 输出门控单元 (OGUy)

每个 OGUy (y = 0-3) 单元将输入通道送出的有效触发事件和状态标志进行组合、将结果分配给系统。OGUy 单元的内部逻辑如图 8-29 所示。OGUy 单元的所有功能由和它关联的 EXOCONy 寄存器控制，如 EXOCON0 控制 OGU0。OGUy 单元的功能可分为两部分：

• 触发组合 (见 [章节 8.9.6.1](#)) :

来自输入通道的所有 (被使能并送入 OGUy 的) 触发信号 TRxy、所选择的外设相关触发事件、序列改变事件 (若被使能) 逻辑或。

• 序列检测 (见 [章节 8.9.6.2](#)) :

可使能输入通道的状态标志 EXICONx.FL 参与序列检测。所有使能的状态标志被置位时, 检测到序列匹配。

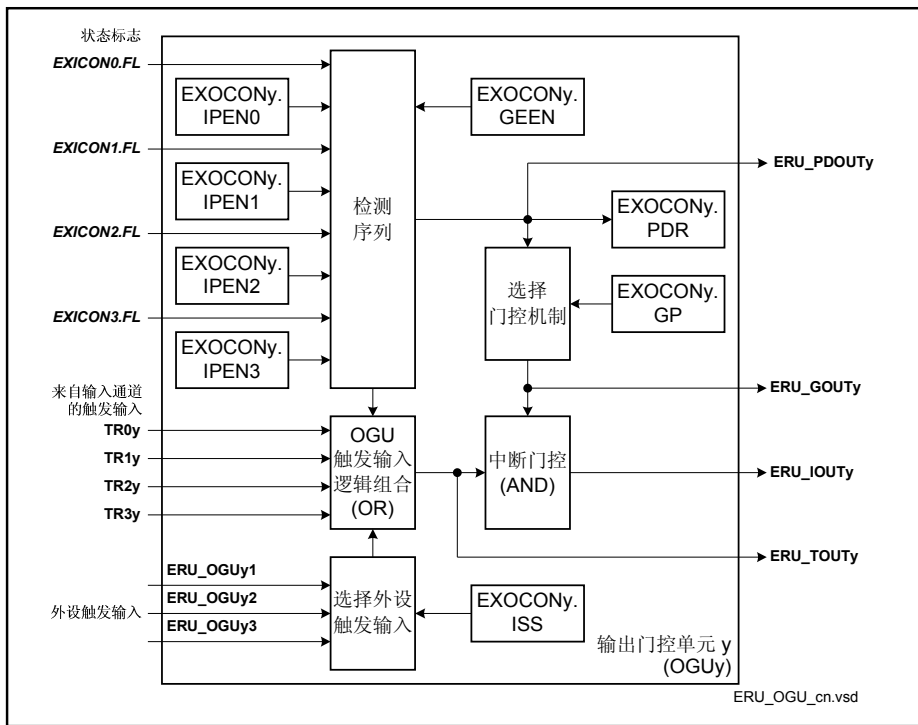


图 8-29 输出通道 y 的输出门控单元

每个 OGUy 单元产生 4 个输出信号 (并非所有的信号均被系统使用, 详见 [章节 8.9.7](#)) :

- **ERU\_PDOUTy**: 直接输出序列匹配信息, 用作其它模块的门控信号 (序列匹配 = 1)。

- **ERU\_GOUTy**: 输出序列匹配或序列信息遗失（序列反转匹配），或在软件控制下输出恒 0 或 1，用作其它模块的门控信号。
- **ERU\_TOUTy**: 将外设触发事件、序列改变事件、ETLx 触发输出 TRxy 逻辑组合，用于触发其它模块的操作。
- **ERU\_IOUTy**: 门控触发输出（ERU\_GOUTy 和 ERU\_TOUTy 逻辑与），用于触发中断。

### 8.9.6.1 触发组合

触发组合将不同的触发输入逻辑或、从而产生一个公共触发信号 ERU\_TOUTy。有以下触发输入信号：

- 每路**输入通道**的 ETLx 单元的触发输出 TRxy，它可被使能并送入某个 OGUy 单元。
- 每个 OGUy 单元可选择三个**外设触发**信号中的一个作为附加触发源。这些外设触发信号由片上外设模块产生（如捕获/比较或定时器单元）。外设触发信号的选择由位域 EXOCONy.ISS 控制。
- 若至少有一个**序列检测**输入被使能（EXOCONy.IPENx），当检测到序列改变时（从匹配到不匹配，或反之），产生一个触发事件以指示序列检测结果事件（若由 EXOCONy.GEEN 使能）。

触发组合可针对多个输入信号（各路输入通道相互独立）或外设信号设定不同的触发条件，并将它们逻辑组合产生一个信号输出，如用于产生中断或触发 ADC 转换。该触发组合功能允许由多个输入信号触发每个 OGU 单元产生中断（多个请求源 → 一个响应）。

**章节 8.17.3.2** 给出 OGUy 单元外设触发信号的连接。

外设触发信号的选择由寄存器 **EXOCON0** (OGU0)、**EXOCON1** (OGU1)、**EXOCON2** (OGU2)、**EXOCON3** (OGU3) 中的位域 ISS 控制。

### 8.9.6.2 序列检测

序列检测逻辑允许将所有 ETLx 单元的状态标志进行组合。每个状态标志可分别参与或不参与各 OGUy 单元的逻辑检测。序列检测模块输出以下检测结果：

- **序列匹配**（EXOCONy.PDR = 1 且 ERU\_PDOUTy = 1）：  
参与序列检测的所有状态标志 FL 均为 1 时，指示序列匹配。
- **序列遗失**（EXOCONy.PDR = 0 且 ERU\_PDOUTy = 0）：  
参与序列检测的所有状态标志 FL 中至少有一个为 0 时，指示序列不匹配。

此外，当序列检测的结果改变时（从匹配到不匹配，或反之），若 EXOCONy.GEEN = 1，序列检测可产生一个触发事件。序列检测结果改变事件和其它

触发事件逻辑或，用以产生中断或触发其它模块（如 ADC）的功能。序列检测的结果改变且 EXOCONy.PDR 被更新时，指示产生触发事件。

OGUy 基于可被序列检测结果 ERU\_PDOUTy 门控（屏蔽）的触发信号 ERU\_TOUTy 产生中断，从而可在某个时窗内自动、再生性的产生中断。此时，请求事件由触发组合逻辑产生，时窗信息（门控）由序列检测给出。例如，当出现期望的输入组合时（基于 ETLx 状态位的序列检测）规律的（选择来自捕获/比较单元的外设触发输入）产生中断。

可编程的门控机制可根据应用需求灵活设计，可在不同条件下产生中断请求 ERU\_IOUTy:

- **序列匹配** (EXOCONy.GP = 10<sub>B</sub>) :  
序列检测指示序列匹配时，发生触发事件时产生中断。
- **序列遗失** (EXOCONy.GP = 11<sub>B</sub>) :  
序列检测指示序列不匹配时，发生触发事件时产生中断。
- **独立于序列检测** (EXOCONy.GP = 01<sub>B</sub>) :  
该模式下，每个发生的触发事件都会引发中断请求。序列检测输出可单独使用，和触发组合无关 (ERU\_TOUTy 和 ERU\_PDOUTy 相互独立使用，基于触发事件产生中断请求)。
- **无中断** (EXOCONy.GP = 00<sub>B</sub>, 缺省设置) :  
该模式下，发生的触发事件不引发中断请求。序列检测输出可单独使用，和触发组合无关 (ERU\_TOUTy 和 ERU\_PDOUTy 相互独立使用，触发事件不产生中断请求)。

### 8.9.7 ERU 输出连接

**章节 8.17.3.3** 描述 ERU 输出用于门控或触发其它模块功能的信号连接、以及和中断控制寄存器的连接。

### 8.9.8 ERU 寄存器

#### 8.9.8.1 外部输入选择寄存器 EXISEL

该寄存器为所有四个 ERS 单元选择 A 和 B 输入。可选的输入信号见 **表 8-19**。

#### EXISEL

外部输入选择寄存器								ESFR (F1A0 <sub>H</sub> /D0 <sub>H</sub> )								复位值: 0000 <sub>H</sub>							
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																

符号	位序号	类型	功能描述
<b>EXS0A</b>	[1:0]	rw	<b>A0 的外部输入选择 (ERS0)</b> 该位域选择 A0 的外部输入。 00 <sub>B</sub> 选择输入 ERU_0A0 01 <sub>B</sub> 选择输入 ERU_0A1 10 <sub>B</sub> 选择输入 ERU_0A2 11 <sub>B</sub> 选择输入 ERU_0A3
<b>EXS0B</b>	[3:2]	rw	<b>B0 的外部输入选择 (ERS0)</b> 该位域选择 B0 的外部输入。 00 <sub>B</sub> 选择输入 ERU_0B0 01 <sub>B</sub> 选择输入 ERU_0B1 10 <sub>B</sub> 选择输入 ERU_0B2 11 <sub>B</sub> 选择输入 ERU_0B3
<b>EXS1A</b>	[5:4]	rw	<b>A1 的外部输入选择 (ERS1)</b> 该位域选择 A1 的外部输入。 00 <sub>B</sub> 选择输入 ERU_1A0 01 <sub>B</sub> 选择输入 ERU_1A1 10 <sub>B</sub> 选择输入 ERU_1A2 11 <sub>B</sub> 选择输入 ERU_1A3
<b>EXS1B</b>	[7:6]	rw	<b>B1 的外部输入选择 (ERS1)</b> 该位域选择 B1 的外部输入。 00 <sub>B</sub> 选择输入 ERU_1B0 01 <sub>B</sub> 选择输入 ERU_1B1 10 <sub>B</sub> 选择输入 ERU_1B2 11 <sub>B</sub> 选择输入 ERU_1B3
<b>EXS2A</b>	[9:8]	rw	<b>A2 的外部输入选择 (ERS2)</b> 该位域选择 A2 的外部输入。 00 <sub>B</sub> 选择输入 ERU_2A0 01 <sub>B</sub> 选择输入 ERU_2A1 10 <sub>B</sub> 选择输入 ERU_2A2 11 <sub>B</sub> 选择输入 ERU_2A3

<b>EXS2B</b>	[11:10]	rw	<b>B2 的外部输入选择 (ERS2)</b> 该位域选择 B2 的外部输入。 00 <sub>B</sub> 选择输入 ERU_2B0 01 <sub>B</sub> 选择输入 ERU_2B1 10 <sub>B</sub> 选择输入 ERU_2B2 11 <sub>B</sub> 选择输入 ERU_2B3
<b>EXS3A</b>	[13:12]	rw	<b>A3 的外部输入选择 (ERS3)</b> 该位域选择 A3 的外部输入。 00 <sub>B</sub> 选择输入 ERU_3A0 01 <sub>B</sub> 选择输入 ERU_3A1 10 <sub>B</sub> 选择输入 ERU_3A2 11 <sub>B</sub> 选择输入 ERU_3A3
<b>EXS3B</b>	[15:14]	rw	<b>B3 的外部输入选择 (ERS3)</b> 该位域选择 B3 的外部输入。 00 <sub>B</sub> 选择输入 ERU_3B0 01 <sub>B</sub> 选择输入 ERU_3B1 10 <sub>B</sub> 选择输入 ERU_3B2 11 <sub>B</sub> 选择输入 ERU_3B3

### 8.9.8.2 外部输入控制寄存器 EXICONx

这些寄存器控制 ERSx 单元的输入和 ETLx 单元的触发功能 (x = 0...3)。

#### EXICON0

外部输入控制 0 寄存器      **ESFR (F030<sub>H</sub>/18<sub>H</sub>)**      复位值: 0000<sub>H</sub>

#### EXICON1

外部输入控制 1 寄存器      **ESFR (F032<sub>H</sub>/19<sub>H</sub>)**      复位值: 0000<sub>H</sub>

#### EXICON2

外部输入控制 2 寄存器      **ESFR (F034<sub>H</sub>/1A<sub>H</sub>)**      复位值: 0000<sub>H</sub>

#### EXICON3

外部输入控制 3 寄存器      **ESFR (F036<sub>H</sub>/1C<sub>H</sub>)**      复位值: 0000<sub>H</sub>

**系统控制单元 (SCU)**

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

符号	位序号	类型	功能描述
<b>PE</b>	0	rw	<b>ETLx 的输出触发脉冲使能</b> 检测到所选择的跳变沿时（状态标志 <b>FL</b> 置位），该位使能在 <b>TRxy</b> 上产生输出触发脉冲。 <b>0<sub>B</sub></b> 禁止产生触发脉冲。 <b>1<sub>B</sub></b> 使能产生触发脉冲。
<b>LD</b>	1	rw	<b>ETLx 状态标志的重建电平检测</b> 该位选择状态标志 <b>FL</b> 被用作“粘着”位、还是用于重建电平检测。 <b>0<sub>B</sub></b> 状态标志 <b>FL</b> 不由硬件清零、用作“粘着”位。 <b>FL</b> 一旦置位，在被软件清零之前不受任何跳变沿的影响。 <b>1<sub>B</sub></b> 状态标志 <b>FL</b> 重建期望事件的电平检测。若 <b>RE = 1</b> ，出现上升沿事件 <b>FL</b> 被自动置位；若 <b>FE = 1</b> ，出现下降沿事件时 <b>FL</b> 被自动置位。若 <b>RE = 0</b> ，出现上升沿事件 <b>FL</b> 被自动清零；若 <b>FE = 0</b> ，出现下降沿事件时 <b>FL</b> 被自动清零。
<b>RE</b>	2	rw	<b>ETLx 的上升沿检测使能</b> 该位使能/禁止上升沿事件作为跳变沿事件。跳变沿事件用于置位状态标志 <b>FL</b> 或产生触发脉冲 <b>TRxy</b> 。 <b>0<sub>B</sub></b> 上升沿不被用作跳变沿事件。 <b>1<sub>B</sub></b> 上升沿被用作跳变沿事件。
<b>FE</b>	3	rw	<b>ETLx 的下降沿检测使能</b> 该位使能/禁止下降沿事件作为跳变沿事件。跳变沿事件用于置位状态标志 <b>FL</b> 或产生触发脉冲 <b>TRxy</b> 。 <b>0<sub>B</sub></b> 下降沿不被用作跳变沿事件。 <b>1<sub>B</sub></b> 下降沿被用作跳变沿事件。

符号	位序号	类型	功能描述
<b>OCS</b>	[6:4]	rw	<b>ETLx 输出触发脉冲的输出通道选择</b> 该位定义触发脉冲 TR <sub>xy</sub> 被送入哪路输出通道 OGU <sub>y</sub> 。 000 <sub>B</sub> 触发脉冲被送入 OGU0 001 <sub>B</sub> 触发脉冲被送入 OGU1 010 <sub>B</sub> 触发脉冲被送入 OGU2 011 <sub>B</sub> 触发脉冲被送入 OGU3 1XX <sub>B</sub> 保留, 不使用该组合
<b>FL</b>	7	rwh	<b>ETLx 的状态标志</b> 该位为状态标志, 由跳变沿检测置位或清零。 0 <sub>B</sub> 还未检测到选择的跳变沿事件。 1 <sub>B</sub> 已检测到选择的跳变沿事件。
<b>SS</b>	[9:8]	rw	<b>ERSx 的输入源选择</b> 该位域定义执行哪种逻辑操作以产生 ERSxO。 00 <sub>B</sub> 输入 A, 无附加的逻辑操作。 01 <sub>B</sub> 输入 B, 无附加的逻辑操作。 10 <sub>B</sub> 输入 A 和输入 B 逻辑或。 11 <sub>B</sub> 输入 A 和输入 B 逻辑与。
<b>NA</b>	10	rw	<b>ERSx 的输入 A 反相选择</b> 该位选择输入 A 的极性。 0 <sub>B</sub> 输入 A 直接使用。 1 <sub>B</sub> 输入 A 反相。
<b>NB</b>	11	rw	<b>ERSx 的输入 B 反相选择</b> 该位选择输入 B 的极性。 0 <sub>B</sub> 输入 B 直接使用。 1 <sub>B</sub> 输入 B 反相。
<b>0</b>	[15:12]	r	<b>保留</b> 读操作返回 0; 应写入 0。



### 8.9.8.3 输出控制寄存器 EXOCONy

这些寄存器控制输出门控单元 y 的输出 (y = 0...3)。

#### EXOCON0

外部输出触发控制 0 寄存器      **ESFR (FE30<sub>H</sub>/18<sub>H</sub>)**      复位值: 0008<sub>H</sub>

#### EXOCON1

外部输出触发控制 1 寄存器      **ESFR (FE32<sub>H</sub>/19<sub>H</sub>)**      复位值: 0008<sub>H</sub>

#### EXOCON2

外部输出触发控制 2 寄存器      **ESFR (FE34<sub>H</sub>/1A<sub>H</sub>)**      复位值: 0008<sub>H</sub>

#### EXOCON3

外部输出触发控制 3 寄存器      **ESFR (FE36<sub>H</sub>/1B<sub>H</sub>)**      复位值: 0008<sub>H</sub>

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	GE EN	ISS	
rw	rw	rw	rw	r						rw		rh	rw	rw	

符号	位序号	类型	功能描述
<b>ISS</b>	[1:0]	rw	<b>内部触发源选择</b> 该位域定义哪路输入用作 OGUy 的外设触发输入。可用的输入信号归纳见表 8-20。 00 <sub>B</sub> 外设触发功能被禁用 01 <sub>B</sub> 选择输入 ERU_OGUy1 10 <sub>B</sub> 选择输入 ERU_OGUy2 11 <sub>B</sub> 选择输入 ERU_OGUy3
<b>GEEN</b>	2	rw	<b>门控事件使能</b> 当序列检测的结果改变时（从匹配到不匹配，或反之），位 GEEN 使能产生触发事件。 0 <sub>B</sub> 事件检测禁用。 1 <sub>B</sub> 事件检测使能。
<b>PDR</b>	3	rh	<b>序列检测结果标志</b> 该位指示序列检测的结果。 0 <sub>B</sub> 检测到序列不匹配。

符号	位序号	类型	功能描述
			1 <sub>B</sub> 检测到序列匹配。
<b>GP</b>	[5:4]	rw	<b>序列检测结果的门控选择</b> 该位域定义中断产生的门控机制（OGU 输出 ERU_PDOUT <sub>y</sub> 和 ERU_GOUT <sub>y</sub> 之间的关系）。 00 <sub>B</sub> ERU_GOUT <sub>y</sub> 始终被禁用，ERU_IOUT <sub>y</sub> 不能被激活。 01 <sub>B</sub> ERU_GOUT <sub>y</sub> 始终被使能，每次激活 ERU_TOUT <sub>y</sub> 时 ERU_IOUT <sub>y</sub> 被激活。 10 <sub>B</sub> ERU_GOUT <sub>y</sub> 等于 ERU_PDOUT <sub>y</sub> ，当检测到期望序列时（序列匹配 PDR = 1），ERU_TOUT <sub>y</sub> 被激活时 ERU_IOUT <sub>y</sub> 被激活。 11 <sub>B</sub> ERU_GOUT <sub>y</sub> 和 ERU_PDOUT <sub>y</sub> 反相，未检测到期望序列时（序列不匹配 PDR = 0），ERU_TOUT <sub>y</sub> 被激活时 ERU_IOUT <sub>y</sub> 被激活。
<b>IPEN<sub>x</sub></b> <b>(x =0-3)</b>	12+x	rw	<b>ETL<sub>x</sub> 的序列检测使能</b> 位 IPEN <sub>x</sub> 定义 ETL <sub>x</sub> 的触发事件状态标志 EXICON <sub>x</sub> .FL 是否参与 OGU <sub>y</sub> 的序列检测。 0 <sub>B</sub> 标志 EXICON <sub>x</sub> .FL 不参与序列检测。 1 <sub>B</sub> 标志 EXICON <sub>x</sub> .FL 参与序列检测。
<b>0</b>	[11:6]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 8.10 SCU 中断产生

SCU 的中断结构如图 8-30 所示。

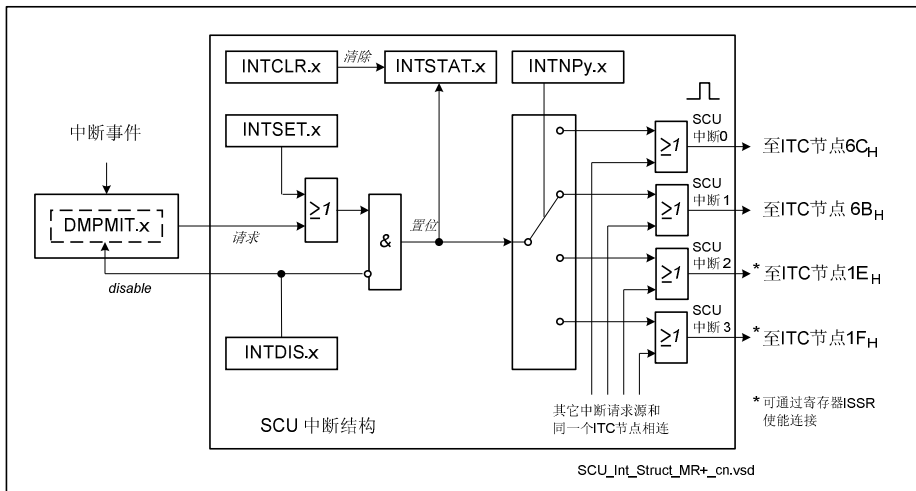


图 8-30 SCU 的中断结构

若使能寄存器 INTDIS 的相应位，中断可通过中断请求线或软件置位寄存器 INTSET 触发。中断请求触发源置位寄存器 INTSTAT 中的相应标志位，将该中断请求送至某个中断节点（由节点指针寄存器 INTNP0 或 INTNP1 选择）。

可通过置位寄存器 INTCLR 软件清除中断标志。

若多个中断请求源和同一个中断节点指针（寄存器 INTNPx 中）相连，这些请求源最终组合成一条公共的请求线。

### 中断节点的分配

SCU 模块的中断请求源可被映射到专用中断节点 6C<sub>H</sub> 或 6B<sub>H</sub> 上，通过编程中断节点指针寄存器 INTNP0 和 INTNP1 来实现。

此外，若不使用 CAPCOM2 通道 30 和 31 的中断，SCU 中断可通过寄存器 ISSR 映射到分配给 CAPCOM2 的中断节点 1E<sub>H</sub> 或 1F<sub>H</sub> 上。因此，对于 SCU 中断而言，除可选择中断节点 6C<sub>H</sub> 或 6B<sub>H</sub> 之外，还可通过寄存器 ISSR 选择节点 1E<sub>H</sub> 或 1F<sub>H</sub>。

中断请求源的缺省节点分配归纳见表 8-10。

#### 8.10.1 中断支持

一些中断请求首先被送入 DMP\_M 电源域内的粘着标志寄存器。这些标志由触发置位而且一旦置位会触发电源域 DMP\_1 产生中断。

哪些中断请求源在寄存器 DMPMIT 中具有粘着标志归纳见表 8-10。

## 系统控制单元 (SCU)

注：响应 SCU 强制中断请求时，确保在处理完相应的请求后清除所有相关请求标志。要清除保存在寄存器 **DMPMIT** 中的中断请求时，首先清除相应的中断请求源（如 **WUTRG**），然后通过 **DMPMITCLR** 清除 **DMP\_M** 中的请求，之后再通过 **INTCLR** 清除 **DMP\_1** 中的请求。

### 8.10.2 SCU 中断请求源

SCU 的中断请求源归纳见表 8-10。

表 8-10 SCU 中断概览

中断请求源	缩略名	DMPMIT 中的粘着标志	INTNPx 中的缺省中断节点分配
SWD OK 1 中断	SWDI1	有	6C <sub>H</sub>
SWD OK 2 中断	SWDI2	有	6B <sub>H</sub>
PVC_M OK 1 中断	PVCM1	有	6C <sub>H</sub>
PVC_M OK 2 中断	PVCM2	有	6B <sub>H</sub>
PVC_1 OK 1 中断	PVC1I1	有	6C <sub>H</sub>
PVC_1 OK 2 中断	PVC1I2	有	6B <sub>H</sub>
唤醒定时器中断	WUI	有	6B <sub>H</sub>
唤醒定时器均衡中断	WUTI	有	6C <sub>H</sub>
看门狗定时器中断	WDTI	--	6B <sub>H</sub>
GSC 中断	GSC1	有	6C <sub>H</sub>
STM0 中断	STM0I	有	6B <sub>H</sub>
STM1 中断	STM1I	有	6C <sub>H</sub>
MCHK 中断	MCHKI	--	6B <sub>H</sub>
程序 Flash 中断	PFI	--	6C <sub>H</sub>

### 8.10.3 中断控制寄存器

#### 8.10.3.1 寄存器 INTSTAT

该寄存器中存放 SCU 的所有中断请求源的状态标志。可分别通过寄存器 INTSET 和 INTCLR 软件置位和清除这些状态位。

#### INTSTAT

中断状态寄存器

SFR (FF00<sub>H</sub>/80<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	PFI	M CHKI	STM1 I	STM0 I	GSC I	WDT I	WUI	WUT I	PVC1 I2	PVC1 I1	PVC MI2	PVC MI1	SWD I2	SWD I1	
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

符号	位序号	类型	功能描述
<b>SWDI1</b>	0	rh	<b>SWD 中断请求标志 1</b> 若位 DMPMIT.SWDI1 置位, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 SWDI1 中断请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 SWDI1 中断请求。
<b>SWDI2</b>	1	rh	<b>SWD 中断请求标志 2</b> 若位 DMPMIT.SWDI2 置位, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 SWDI2 中断请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 SWDI2 中断请求。
<b>PVCMI1</b>	2	rh	<b>PVC_M 中断请求标志 1</b> 若位 DMPMIT.PVCMI1 置位, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 PVCMI1 中断请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 PVCMI1 中断请求。

符号	位序号	类型	功能描述
<b>PVCM12</b>	3	rh	<b>PVC_M 中断请求标志 2</b> 若位 DMPMIT.PVCM12 置位, 该位置位。 <b>0<sub>B</sub></b> 自该位被上次清零后, 未发生 PVCM12 中断请求。 <b>1<sub>B</sub></b> 自该位被上次清零后, 已发生 PVCM12 中断请求。
<b>PVC111</b>	4	rh	<b>PVC_1 中断请求标志 1</b> 若位 DMPMIT.PVC111 置位, 该位置位。 <b>0<sub>B</sub></b> 自该位被上次清零后, 未发生 PVC111 中断请求。 <b>1<sub>B</sub></b> 自该位被上次清零后, 已发生 PVC111 中断请求。
<b>PVC112</b>	5	rh	<b>PVC_1 中断请求标志 2</b> 若位 DMPMIT.PVC112 置位, 该位置位。 <b>0<sub>B</sub></b> 自该位被上次清零后, 未发生 PVC112 中断请求。 <b>1<sub>B</sub></b> 自该位被上次清零后, 已发生 PVC112 中断请求。
<b>WUTI</b>	6	rh	<b>唤醒定时器均衡中断请求标志</b> 若发生 WUT 均衡触发事件且位 INTDIS.WUTI = 0, 该位置位。 <b>0<sub>B</sub></b> 自该位被上次清零后, 未发生 WUT 中断请求。 <b>1<sub>B</sub></b> 自该位被上次清零后, 已发生 WUT 中断请求。
<b>WUI</b>	7	rh	<b>唤醒定时器中断请求标志</b> 若发生 WU 触发事件且位 INTDIS.WUI = 0, 该位置位。 <b>0<sub>B</sub></b> 自该位被上次清零后, 未发生 WU 中断请求。 <b>1<sub>B</sub></b> 自该位被上次清零后, 已发生 WU 中断请求。
<b>WDTI</b>	8	rh	<b>看门狗定时器中断请求标志</b> 若进入 WDT 预警模式且位 INTDIS.WDTI = 0, 该

符号	位序号	类型	功能描述
			位置位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 WDT 中断请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 WDT 中断请求。
<b>GSCI</b>	9	rh	<b>GSC 中断请求标志</b> 若 GSC 错误位置位且位 INTDIS.GSCI = 0, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 GSC 中断请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 GSC 中断请求。
<b>STM0I</b>	10	rh	<b>STM 中断 0 请求标志</b> 若 STM 中断触发 0 置位且位 INTDIS.STM0I = 0, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 STM0 中断请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 STM0 中断请求。
<b>STM1I</b>	11	rh	<b>STM 中断 1 请求标志</b> 若 STM 中断触发 1 置位且位 INTDIS.STM1I = 0, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 STM1 中断请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 STM1 中断请求。
<b>MCHKI</b>	12	rh	<b>MCHK 中断请求标志</b> 若 MCHK 中断触发置位且位 INTDIS.MCHKI = 0, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 MCHK 中断请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 MCHK 中断请求。
<b>PFI</b>	13	rh	<b>程序 Flash 中断请求标志</b>

符号	位序号	类型	功能描述
			若程序 Flash 中断触发置位且位 INTDIS.PFI = 0，该位置位。 0 <sub>B</sub> 自该位被上次清零后，未发生 PF 中断请求。 1 <sub>B</sub> 自该位被上次清零后，已发生 PF 中断请求。
<b>0</b>	[15:14]	rh	<b>保留</b> 读操作返回 0；应写入 0。

### 8.10.3.2 寄存器 INTCLR

可通过该寄存器软件清除 SCU 的所有中断请求源的状态标志。

#### INTCLR

中断清除寄存器

SFR (FE82<sub>H</sub>/41<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>PFI</b>	<b>M</b> <b>CHKI</b>	<b>STM1</b> <b>I</b>	<b>STM0</b> <b>I</b>	<b>GSC</b> <b>I</b>	<b>WDT</b> <b>I</b>	<b>WUI</b>	<b>WUT</b> <b>I</b>	<b>PVC1</b> <b>I2</b>	<b>PVC1</b> <b>I1</b>	<b>PVC</b> <b>MI2</b>	<b>PVC</b> <b>MI1</b>	<b>SWD</b> <b>I2</b>	<b>SWD</b> <b>I1</b>	
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

符号	位序号	类型	功能描述
<b>SWDI1</b>	0	w	<b>清除 SWD 中断请求标志 1</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT.SWDI1。
<b>SWDI2</b>	1	w	<b>清除 SWD 中断请求标志 2</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT.SWDI2。
<b>PVCMI1</b>	2	w	<b>清除 PVC_M 中断请求标志 1</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT.PVCMI1。
<b>PVCMI2</b>	3	w	<b>清除 PVC_M 中断请求标志 2</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT.PVCMI2。



符号	位序号	类型	功能描述
<b>PVC1I1</b>	4	w	清除 <b>PVC_1</b> 中断请求标志 1 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT.PVC1I1。
<b>PVC1I2</b>	5	w	清除 <b>PVC_1</b> 中断请求标志 2 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT. PVC1I2。
<b>WUTI</b>	6	w	清除唤醒定时器均衡中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT. WUTI。
<b>WUI</b>	7	w	清除唤醒定时器中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT. WUI。
<b>WDTI</b>	8	w	清除看门狗定时器中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT. WDTI。
<b>GSCI</b>	9	w	清除 <b>GSC</b> 中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT. GSCI。
<b>STM0I</b>	10	w	清除 <b>STM0</b> 中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT. STM0I。
<b>STM1I</b>	11	w	清除 <b>STM1</b> 中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT. STM1I。
<b>MCHKI</b>	12	w	清除 <b>MCHK</b> 中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT. MCHKI。

符号	位序号	类型	功能描述
<b>PFI</b>	13	w	<b>清除程序 Flash 中断请求标志</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 INTSTAT.PFI。
<b>0</b>	[15:14]	w	<b>保留</b> 必须写入 0。

注：读取这些位始终返回 0。

### 8.10.3.3 寄存器 INTSET

可通过该寄存器软件置位 SCU 的所有中断请求源的状态标志。

#### INTSET

中断置位寄存器

SFR (FE80<sub>H</sub>/40<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>PFI</b>	<b>M CHKI</b>	<b>STM1 I</b>	<b>STM0 I</b>	<b>GSC I</b>	<b>WDT I</b>	<b>WUI</b>	<b>WUT I</b>	<b>PVC1 I2</b>	<b>PVC1 I1</b>	<b>PVC MI2</b>	<b>PVC MI1</b>	<b>SWD I2</b>	<b>SWD I1</b>	
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

符号	位序号	类型	功能描述
<b>SWDI1</b>	0	w	<b>置位 SWD 中断请求标志 1</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.SWDI1。
<b>SWDI2</b>	1	w	<b>置位 SWD 中断请求标志 2</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.SWDI2。
<b>PVCMI1</b>	2	w	<b>置位 PVC_M 中断请求标志 1</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.PVCMI1。
<b>PVCMI2</b>	3	w	<b>置位 PVC_M 中断请求标志 2</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.PVCMI2。

符号	位序号	类型	功能描述
<b>PVC1I1</b>	4	w	<b>置位 PVC_1 中断请求标志 1</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.PVC1I1。
<b>PVC1I2</b>	5	w	<b>置位 PVC_1 中断请求标志 2</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.PVC1I2。
<b>WUTI</b>	6	w	<b>置位唤醒定时器均衡中断请求标志</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.WUTI。
<b>WUI</b>	7	w	<b>置位唤醒定时器中断请求标志</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.WUI。
<b>WDTI</b>	8	w	<b>置位看门狗定时器中断请求标志</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.WDTI。
<b>GSCI</b>	9	w	<b>置位 GSC 中断请求标志</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.GSCI。
<b>STM0I</b>	10	w	<b>置位 STM0 中断请求标志</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.STM0I。
<b>STM1I</b>	11	w	<b>置位 STM1 中断请求标志</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.STM1I。
<b>MCHKI</b>	12	w	<b>置位 MCHK 中断请求标志</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.MCHKI。

符号	位序号	类型	功能描述
<b>PFI</b>	13	w	<b>置位程序 Flash 中断请求标志</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 INTSTAT.PFI。
<b>0</b>	[15:14]	w	<b>保留</b> 必须写入 0。

注：读取这些位始终返回 0。

### 8.10.3.4 寄存器 INTDIS

该寄存器中存放所有 SCU 中断请求源的软件禁用控制位。

#### INTDIS

中断禁用寄存器

SFR (FE84<sub>H</sub>/42<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>PFI</b>	<b>M</b> <b>CHKI</b>	<b>STM1</b> <b>I</b>	<b>STM0</b> <b>I</b>	<b>GSC</b> <b>I</b>	<b>WDT</b> <b>I</b>	<b>WUI</b>	<b>WUT</b> <b>I</b>	<b>PVC1</b> <b>I2</b>	<b>PVC1</b> <b>I1</b>	<b>PVC</b> <b>MI2</b>	<b>PVC</b> <b>MI1</b>	<b>SWD</b> <b>I2</b>	<b>SWD</b> <b>I1</b>	
rW	rW	rW	rW	rW	rW	rW	w	rW	rW	rW	rW	rW	rW	rW	w

符号	位序号	类型	功能描述
<b>SWDI1</b>	0	rW	<b>禁用 SWD 中断请求 1</b> 0 <sub>B</sub> 使能 SWDI1 中断请求。 1 <sub>B</sub> 禁用 SWDI1 中断请求。
<b>SWDI2</b>	1	rW	<b>禁用 SWD 中断请求 2</b> 0 <sub>B</sub> 使能 SWDI2 中断请求。 1 <sub>B</sub> 禁用 SWDI2 中断请求。
<b>PVCMI1</b>	2	rW	<b>禁用 PVC_M 中断请求 1</b> 0 <sub>B</sub> 使能 PVCMI1 中断请求。 1 <sub>B</sub> 禁用 PVCMI1 中断请求。
<b>PVCMI2</b>	3	rW	<b>禁用 PVC_M 中断请求 2</b> 0 <sub>B</sub> 使能 PVCMI2 中断请求。 1 <sub>B</sub> 禁用 PVCMI2 中断请求。

符号	位序号	类型	功能描述
<b>PVC1I1</b>	4	rw	<b>禁用 PVC_1 中断请求 1</b> 0 <sub>B</sub> 使能 PVC1I1 中断请求。 1 <sub>B</sub> 禁用 PVC1I1 中断请求。
<b>PVC1I2</b>	5	rw	<b>禁用 PVC_1 中断请求 2</b> 0 <sub>B</sub> 使能 PVC1I2 中断请求。 1 <sub>B</sub> 禁用 PVC1I2 中断请求。
<b>WUTI</b>	6	rw	<b>禁用唤醒定时器均衡中断请求</b> 0 <sub>B</sub> 使能 WUT 中断请求。 1 <sub>B</sub> 禁用 WUT 中断请求。
<b>WUI</b>	7	rw	<b>禁用唤醒定时器中断请求</b> 0 <sub>B</sub> 使能 WU 中断请求。 1 <sub>B</sub> 禁用 WU 中断请求。
<b>WDTI</b>	8	rw	<b>禁用看门狗定时器中断请求</b> 0 <sub>B</sub> 使能 WDT 中断请求。 1 <sub>B</sub> 禁用 WDT 中断请求。
<b>GSCI</b>	9	rw	<b>禁用 GSC 中断请求</b> 0 <sub>B</sub> 使能 GSC 中断请求。 1 <sub>B</sub> 禁用 GSC 中断请求。
<b>STM0I</b>	10	rw	<b>禁用 STM0 中断请求</b> 0 <sub>B</sub> 使能 STM0 中断请求。 1 <sub>B</sub> 禁用 STM0 中断请求。
<b>STM1I</b>	11	rw	<b>禁用 STM1 中断请求</b> 0 <sub>B</sub> 使能 STM1 中断请求。 1 <sub>B</sub> 禁用 STM1 中断请求。
<b>MCHKI</b>	12	rw	<b>禁用 MCHK 中断请求</b> 0 <sub>B</sub> 使能 MCHK 中断请求。 1 <sub>B</sub> 禁用 MCHK 中断请求。

符号	位序号	类型	功能描述
<b>PFI</b>	13	rw	<b>禁用程序 Flash 中断请求</b> 0 <sub>B</sub> 使能 PF 中断请求。 1 <sub>B</sub> 禁用 PF 中断请求。
<b>0</b>	[15:14]	rw	<b>保留</b> 应写入 0。

### 8.10.3.5 寄存器 INTNP0 和 INTNP1

这些寄存器用于控制 SCU 的所有中断请求源的中断节点指针。

#### INTNP0

中断节点指针 0 寄存器

SFR (FE86<sub>H</sub>/43<sub>H</sub>)

复位值: 4444<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>WU</b>	<b>WUT</b>	<b>PVC12</b>	<b>PVC11</b>	<b>PVCM2</b>	<b>PVCM1</b>	<b>SWD2</b>	<b>SWD1</b>								
rw	rw	rw	rw	rw	rw	rw	rw								

符号	位序号	类型	功能描述
<b>SWD1</b>	[1:0]	rw	<b>SWD 1 中断的中断节点指针</b> 该位域定义 INTSTAT.SWDI1 置位时所请求的中断节点（若由 INTDIS.SWDI1 使能）。 00 <sub>B</sub> 选择中断节点 6C <sub>H</sub> 01 <sub>B</sub> 选择中断节点 6B <sub>H</sub> 10 <sub>B</sub> 若由位 ISSR.ISS14 使能（该位置位），选择中断节点 1E <sub>H</sub> 11 <sub>B</sub> 若由位 ISSR.ISS15 使能（该位置位），选择中断节点 1F <sub>H</sub>
<b>SWD2</b>	[3:2]	rw	<b>SWD 2 中断的中断节点指针</b> 该位域定义 INTSTAT.SWDI2 置位时所请求的中断节点（若由 INTDIS.SWDI2 使能）。 00 <sub>B</sub> 选择中断节点 6C <sub>H</sub> 01 <sub>B</sub> 选择中断节点 6B <sub>H</sub> 10 <sub>B</sub> 若由位 ISSR.ISS14 使能（该位置位），选

符号	位序号	类型	功能描述
			<p>择中断节点 1E<sub>H</sub></p> <p>11<sub>B</sub> 若由位 ISSR.ISS15 使能 (该位置位), 选择中断节点 1F<sub>H</sub></p>
<b>PVCM1</b>	[5:4]	rw	<p><b>PVC_M 1 中断的中断节点指针</b></p> <p>该位域定义 INTSTAT.PVCM1 置位时所请求的中断节点 (若由 INTDIS.PVCM1 使能)。</p> <p>00<sub>B</sub> 选择中断节点 6C<sub>H</sub></p> <p>01<sub>B</sub> 选择中断节点 6B<sub>H</sub></p> <p>10<sub>B</sub> 若由位 ISSR.ISS14 使能 (该位置位), 选择中断节点 1E<sub>H</sub></p> <p>11<sub>B</sub> 若由位 ISSR.ISS15 使能 (该位置位), 选择中断节点 1F<sub>H</sub></p>
<b>PVCM2</b>	[7:6]	rw	<p><b>PVC_M 2 中断的中断节点指针</b></p> <p>该位域定义 INTSTAT.PVCM2 置位时所请求的中断节点 (若由 INTDIS.PVCM2 使能)。</p> <p>00<sub>B</sub> 选择中断节点 6C<sub>H</sub></p> <p>01<sub>B</sub> 选择中断节点 6B<sub>H</sub></p> <p>10<sub>B</sub> 若由位 ISSR.ISS14 使能 (该位置位), 选择中断节点 1E<sub>H</sub></p> <p>11<sub>B</sub> 若由位 ISSR.ISS15 使能 (该位置位), 选择中断节点 1F<sub>H</sub></p>
<b>PVC11</b>	[9:8]	rw	<p><b>PVC_1 1 中断的中断节点指针</b></p> <p>该位域定义 INTSTAT.PVC11 置位时所请求的中断节点 (若由 INTDIS.PVC11 使能)。</p> <p>00<sub>B</sub> 选择中断节点 6C<sub>H</sub></p> <p>01<sub>B</sub> 选择中断节点 6B<sub>H</sub></p> <p>10<sub>B</sub> 若由位 ISSR.ISS14 使能 (该位置位), 选择中断节点 1E<sub>H</sub></p> <p>11<sub>B</sub> 若由位 ISSR.ISS15 使能 (该位置位), 选择中断节点 1F<sub>H</sub></p>
<b>PVC12</b>	[11:10]	rw	<p><b>PVC_1 2 中断的中断节点指针</b></p> <p>该位域定义 INTSTAT.PVC12 置位时所请求的中</p>

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
			断节点 (若由 INTDIS.PVC1I2 使能)。 00 <sub>B</sub> 选择中断节点 6C <sub>H</sub> 01 <sub>B</sub> 选择中断节点 6B <sub>H</sub> 10 <sub>B</sub> 若由位 ISSR.ISS14 使能 (该位置位), 选择中断节点 1E <sub>H</sub> 11 <sub>B</sub> 若由位 ISSR.ISS15 使能 (该位置位), 选择中断节点 1F <sub>H</sub>
<b>WUT</b>	[13:12]	rw	<b>唤醒定时器均衡中断的中断节点指针</b> 该位域定义 INTSTAT.WUTI 置位时所请求的中断节点 (若由 INTDIS.WUTI 使能)。 00 <sub>B</sub> 选择中断节点 6C <sub>H</sub> 01 <sub>B</sub> 选择中断节点 6B <sub>H</sub> 10 <sub>B</sub> 若由位 ISSR.ISS14 使能 (该位置位), 选择中断节点 1E <sub>H</sub> 11 <sub>B</sub> 若由位 ISSR.ISS15 使能 (该位置位), 选择中断节点 1F <sub>H</sub>
<b>WU</b>	[15:14]	rw	<b>唤醒定时器中断的中断节点指针</b> 该位域定义 INTSTAT.WUI 置位时所请求的中断节点 (若由 INTDIS.WUI 使能)。 00 <sub>B</sub> 选择中断节点 6C <sub>H</sub> 01 <sub>B</sub> 选择中断节点 6B <sub>H</sub> 10 <sub>B</sub> 若由位 ISSR.ISS14 使能 (该位置位), 选择中断节点 1E <sub>H</sub> 11 <sub>B</sub> 若由位 ISSR.ISS15 使能 (该位置位), 选择中断节点 1F <sub>H</sub>

**INTNP1**

**中断节点指针 1 寄存器**

**SFR (FE88<sub>H</sub>/44<sub>H</sub>)**

**复位值: 1111<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-				PF		MCHK		STM1		STM0		GSC		WDT	
-				rw		rw		rw		rw		rw		rw	



符号	位序号	类型	功能描述
<b>WDT</b>	[1:0]	rw	<b>WDT 中断的中断节点指针</b> 该位域定义 INTSTAT.WDTI 置位时所请求的中断节点（若由 INTDIS.WDTI 使能）。 00 <sub>B</sub> 选择中断节点 6C <sub>H</sub> 01 <sub>B</sub> 选择中断节点 6B <sub>H</sub> 10 <sub>B</sub> 若由位 ISSR.ISS14 使能（该位置位），选择中断节点 1E <sub>H</sub> 11 <sub>B</sub> 若由位 ISSR.ISS15 使能（该位置位），选择中断节点 1F <sub>H</sub>
<b>GSC</b>	[3:2]	rw	<b>GSC 中断的中断节点指针</b> 该位域定义 INTSTAT.GSCI 置位时所请求的中断节点（若由 INTDIS.GSCI 使能）。 00 <sub>B</sub> 选择中断节点 6C <sub>H</sub> 01 <sub>B</sub> 选择中断节点 6B <sub>H</sub> 10 <sub>B</sub> 若由位 ISSR.ISS14 使能（该位置位），选择中断节点 1E <sub>H</sub> 11 <sub>B</sub> 若由位 ISSR.ISS15 使能（该位置位），选择中断节点 1F <sub>H</sub>
<b>STM0</b>	[5:4]	rw	<b>STM0 中断的中断节点指针</b> 该位域定义 INTSTAT.STM0I 置位时所请求的中断节点（若由 INTDIS.STM0I 使能）。 00 <sub>B</sub> 选择中断节点 6C <sub>H</sub> 01 <sub>B</sub> 选择中断节点 6B <sub>H</sub> 10 <sub>B</sub> 若由位 ISSR.ISS14 使能（该位置位），选择中断节点 1E <sub>H</sub> 11 <sub>B</sub> 若由位 ISSR.ISS15 使能（该位置位），选择中断节点 1F <sub>H</sub>
<b>STM1</b>	[7:6]	rw	<b>STM1 中断的中断节点指针</b> 该位域定义 INTSTAT.STM1I 置位时所请求的中断节点（若由 INTDIS.STM1I 使能）。 00 <sub>B</sub> 选择中断节点 6C <sub>H</sub> 01 <sub>B</sub> 选择中断节点 6B <sub>H</sub>



符号	位序号	类型	功能描述
			10 <sub>B</sub> 若由位 ISSR.ISS14 使能（该位置位），选择中断节点 1E <sub>H</sub> 11 <sub>B</sub> 若由位 ISSR.ISS15 使能（该位置位），选择中断节点 1F <sub>H</sub>
MCHK	[9:8]	rw	<b>MCHK 中断的中断节点指针</b> 该位域定义 INTSTAT.MCHKI 置位时所请求的中断节点（若由 INTDIS. MCHKI 使能）。 00 <sub>B</sub> 选择中断节点 6C <sub>H</sub> 01 <sub>B</sub> 选择中断节点 6B <sub>H</sub> 10 <sub>B</sub> 若由位 ISSR.ISS14 使能（该位置位），选择中断节点 1E <sub>H</sub> 11 <sub>B</sub> 若由位 ISSR.ISS15 使能（该位置位），选择中断节点 1F <sub>H</sub>
PF	[11:10]	rw	<b>程序 Flash 中断的中断节点指针</b> 该位域定义 INTSTAT.PFI 置位时所请求的中断节点（若由 INTDIS. PFI 使能）。 00 <sub>B</sub> 选择中断节点 6C <sub>H</sub> 01 <sub>B</sub> 选择中断节点 6B <sub>H</sub> 10 <sub>B</sub> 若由位 ISSR.ISS14 使能（该位置位），选择中断节点 1E <sub>H</sub> 11 <sub>B</sub> 若由位 ISSR.ISS15 使能（该位置位），选择中断节点 1F <sub>H</sub>

8.10.3.6 寄存器 DMPMIT

该寄存器中存放 DMP\_M 电源域内的粘着中断和强制中断标志。

DMPMIT

DMP\_M 中断和强制中断触发寄存器

SFR（FE96 <sub>H</sub> /4B <sub>H</sub> ）														复位值: 0000 <sub>H</sub>	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RAT	0	ESR 2T	ESR 1T	ESR 0T	STM 1	STM 0	GSC	WUI	WUT I	PVC1 I2	PVC1 I1	PVC MI2	PVC MI1	SWD I2	SWD I1
rh	r	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

符号	位序号	类型	功能描述
<b>SWDI1</b>	0	rh	<b>SWD 中断请求标志 1</b> 若 SWDCON0.L1OK 和 SWDCON0.L1ALEV 定义的生效电平相匹配且 SWDCON0.L1INTEN = 1 <sub>B</sub> , 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 SWDI1 中断。 1 <sub>B</sub> 自该位被上次清零后, 已请求 SWDI1 中断。
<b>SWDI2</b>	1	rh	<b>SWD 中断请求标志 2</b> 若 SWDCON0.L2OK 和 SWDCON0.L2ALEV 定义的生效电平相匹配且 SWDCON0.L2INTEN = 1 <sub>B</sub> , 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 SWDI2 中断。 1 <sub>B</sub> 自该位被上次清零后, 已请求 SWDI2 中断。
<b>PVCM11</b>	2	rh	<b>PVC_M 中断请求标志 1</b> 若 PVCMDCON0.LEV1OK 被清零且 PVCMDCON0.L1INTEN = 1 <sub>B</sub> , 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 PVCM11 中断。 1 <sub>B</sub> 自该位被上次清零后, 已请求 PVCM11 中断。
<b>PVCM12</b>	3	rh	<b>PVC_M 中断请求标志 2</b> 若 PVCMDCON0.LEV2OK 被清零且 PVCMDCON0.L2INTEN = 1 <sub>B</sub> , 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 PVCM12 中断。 1 <sub>B</sub> 自该位被上次清零后, 已请求 PVCM12 中断。
<b>PVC111</b>	4	rh	<b>PVC_1 中断请求标志 1</b> 若 PVC1CON0.LEV1OK 被清零且 PVC1CON0.L1INTEN = 1 <sub>B</sub> , 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 PVC111 中断。 1 <sub>B</sub> 自该位被上次清零后, 已请求 PVC111 中断。
<b>PVC112</b>	5	rh	<b>PVC_1 中断请求标志 2</b> 若 PVC1CON0.LEV2OK 被清零且 PVC1CON0.L2INTEN = 1 <sub>B</sub> , 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 PVC112 中断。

**系统控制单元 (SCU)**

			1 <sub>B</sub> 自该位被上次清零后, 已请求 PVC1I2 中断。
<b>WUTI</b>	6	rh	<b>唤醒定时器均衡中断请求标志</b> 若发生唤醒定时器均衡触发事件, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 WUT 中断。 1 <sub>B</sub> 自该位被上次清零后, 已请求 WUT 中断。
<b>WUI</b>	7	rh	<b>唤醒定时器中断请求标志</b> 若发生唤醒定时器触发事件, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 WU 中断。 1 <sub>B</sub> 自该位被上次清零后, 已请求 WU 中断。
<b>GSC</b>	8	rh	<b>GSC 中断请求标志</b> 若发生 GSC 触发事件, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 GSC 中断。 1 <sub>B</sub> 自该位被上次清零后, 已请求 GSC 中断。
<b>STM0</b>	9	rh	<b>STM0 中断请求标志</b> 若发生 STM0 触发事件, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 STM0 中断。 1 <sub>B</sub> 自该位被上次清零后, 已请求 STM0 中断。
<b>STM1</b>	10	rh	<b>STM1 中断请求标志</b> 若发生 STM1 触发事件, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 STM1 中断。 1 <sub>B</sub> 自该位被上次清零后, 已请求 STM1 中断。
<b>ESR0T</b>	11	rh	<b>ESR0 强制中断请求标志</b> 若引脚 $\overline{\text{ESR0}}$ 生效, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 ESR0 强制中断。 1 <sub>B</sub> 自该位被上次清零后, 已请求 ESR0 强制中断。
<b>ESR1T</b>	12	rh	<b>ESR1 强制中断请求标志</b> 若引脚 $\overline{\text{ESR1}}$ 生效, 该位置位。 0 <sub>B</sub> 自该位被上次清零后, 未请求 ESR1 强制中

			<p>断。</p> <p><b>1<sub>B</sub></b> 自该位被上次清零后，已请求 ESR1 强制中断。</p>
<b>ESR2T</b>	13	rh	<p><b>ESR2 强制中断请求标志</b></p> <p>若引脚 <math>\overline{\text{ESR2}}</math> 生效，该位置位。</p> <p><b>0<sub>B</sub></b> 自该位被上次清零后，未请求 ESR2 强制中断。</p> <p><b>1<sub>B</sub></b> 自该位被上次清零后，已请求 ESR2 强制中断。</p>
<b>RAT</b>	15	rh	<p><b>寄存器访问强制中断请求标志</b></p> <p>若对受保护寄存器进行未授权的访问，该位置位。</p> <p><b>0<sub>B</sub></b> 自该位被上次清零后，未请求 RA 强制中断。</p> <p><b>1<sub>B</sub></b> 自该位被上次清零后，已请求 RA 强制中断。</p>
<b>0</b>	14	r	<p><b>保留</b></p> <p>读操作返回 0；应写入 0。</p>

### 8.10.3.7 寄存器 DMPMITCLR

可通过该寄存器软件清除 DMP\_M 电源域内所有中断和强制中断请求源的粘着状态标志。

#### DMPMITCLR

#### DMP\_M 中断和强制中断清除寄存器

**SFR (FE98<sub>H</sub>/4C<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RAT</b>	<b>0</b>	<b>ESR 2T</b>	<b>ESR 1T</b>	<b>ESR 0T</b>	<b>STM 1</b>	<b>STM 0</b>	<b>GSC</b>	<b>WUI</b>	<b>WUT 1</b>	<b>PVC1 I2</b>	<b>PVC1 I1</b>	<b>PVC MI2</b>	<b>PVC MI1</b>	<b>SWD I2</b>	<b>SWD I1</b>
w	r	w	w	w	w	w	w	w	w	w	w	w	w	w	w

符号	位序号	类型	功能描述
<b>SWDI1</b>	0	w	<p><b>清除 SWD 中断请求标志 1</b></p> <p><b>0<sub>B</sub></b> 无操作。</p> <p><b>1<sub>B</sub></b> 清除位 DMPMIT.SWDI1。</p>

符号	位序号	类型	功能描述
<b>SWDI2</b>	1	w	清除 <b>SWD</b> 中断请求标志 2 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.SWDI2。
<b>PVCMI1</b>	2	w	清除 <b>PVC_M</b> 中断请求标志 1 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.PVCMI1。
<b>PVCMI2</b>	3	w	清除 <b>PVC_M</b> 中断请求标志 2 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.PVCMI2。
<b>PVC1I1</b>	4	w	清除 <b>PVC_1</b> 中断请求标志 1 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.PVC1I1。
<b>PVC1I2</b>	5	w	清除 <b>PVC_1</b> 中断请求标志 2 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT. PVC1I2。
<b>WUTI</b>	6	w	清除唤醒定时器均衡中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.WUTI。
<b>WUI</b>	7	w	清除唤醒定时器中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.WUI。
<b>GSC</b>	8	w	清除 <b>GSC</b> 中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.GSCI。
<b>STM0</b>	9	w	清除 <b>STM0</b> 中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.STM0I。

符号	位序号	类型	功能描述
STM1	10	w	清除 STM1 中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.STM1I。
ESR0T	11	w	清除 $\overline{\text{ESR0}}$ 强制中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.ESR0T。
ESR1T	12	w	清除 $\overline{\text{ESR1}}$ 强制中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.ESR1T。
ESR2T	13	w	清除 $\overline{\text{ESR2}}$ 强制中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.ESR2T。
RAT	15	w	清除寄存器访问强制中断请求标志 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除位 DMPMIT.RAT。
0	14	r	保留 读操作返回 0；应写入 0。

注：读写类型为 w 的这些位读取始终返回 0。

## 8.11 温度补偿单元

对端口驱动进行温度补偿可确保在给定的温度范围内输出驱动特性稳定。

温度补偿振荡器（传感器）由一个自由运行的且对温度敏感的振荡器产生参考时钟。一个使能触发信号用来定义这样一个计数周期数，在该周期内参考时钟脉冲累加计数构成该传感器值 TCLR.THCOUNT。使能触发信号来自系统时钟经预分频和分频处理（见图 8-31），分频因子可根据系统频率编程设定。

每个计数周期结束后，将得到的计数值（即参考时钟周期数）复制到位域 TCLR.THCOUNT 中。因此，每个计数周期后更新 TCLR.THCOUNT（若使能温度补偿）。

软件可将和温度相关的计数值（TCLR.THCOUNT）和多个阈值（温度等级）进行比较以确定控制值 TCCR.TCC。

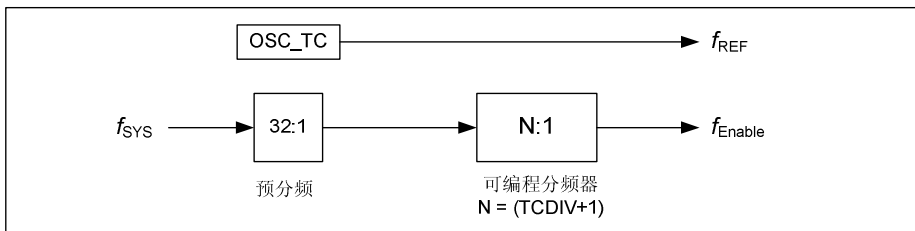


图 8-31 温度补偿时钟产生

由位域 **TCCR.TCDIV** 设定时钟分频因子，可根据数据手册提供的公式计算应使用的分频值。

通常，温度补偿由用户控制。可通过温度补偿控制寄存器 **TCCR** 获得实际的补偿值（由传感器产生）并由软件控制引出端。系统工作期间，可由片上传感器的值或外部提供的补偿值控制器件（即引出端）。寄存器 **TCCR** 还提供可编程的分频因子。

*注：对于不同的器件，其计数值和温度之间的关系可能不同。因此，每种器件必须单独评估。*

*注：温度补偿电路不会连续产生温度补偿值。设计思路是，软件利用温度补偿寄存器中的值定时更新引出端控制（如通过定时器产生的中断来实现）。由于温度是连续函数，因此，温度读取值是否最新或是前次的测量值，均无关紧要。*

## 8.11.1 温度补偿寄存器

### 8.11.1.1 TCCR

通过该寄存器控制温度补偿单元。

#### TCCR

##### 温度补偿控制寄存器

ESFR (F1AC<sub>H</sub>/D6<sub>H</sub>)

复位值: 0003<sub>H</sub>

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

符号	位序号	类型	功能描述
TCC	[1:0]	rw	温度补偿控制 该值用于控制引出端的温度补偿。



**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
			<b>00<sub>B</sub></b> 最大缩减量 = 最小驱动能力， 即温度极低 <b>11<sub>B</sub></b> 无缩减量 = 最大驱动能力， 即温度极高
<b>TCDIV</b>	[6:2]	rw	<b>温度补偿时钟分频</b> 该位域控制温度补偿电路的时钟分频。
<b>TCE</b>	7	rw	<b>温度补偿使能</b> <b>0<sub>B</sub></b> 无操作 <b>1<sub>B</sub></b> 使能计数以产生新的温度值 清除该位也会终止温度补偿振荡器工作。
<b>0</b>	[15:8]	r	<b>保留</b> 读操作返回 0；应写入 0。

**TCLR**

**温度补偿等级寄存器**

**ESFR (F0AC<sub>H</sub>/56<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>								<b>THCOUNT</b>							
r								rh							

符号	位序号	类型	功能描述
<b>THCOUNT</b>	[7:0]	rh	<b>阈值计数器</b> 返回温度传感器最新计数周期的结果，该值用于和阈值比较。
<b>0</b>	[15:8]	r	<b>保留</b> 读操作返回 0；应写入 0。

*注：阈值计数器不会溢出，但计数到 255 时会停止工作。*

## 8.12 看门狗定时器 (WDT)

本节将描述看门狗定时器 (WDT) 及其功能。

### 8.12.1 概述

看门狗定时器 (WDT) 提供了一种避免系统死锁的安全机制。若在规定的时间内未服务 WDT，它将产生系统复位。

#### 特性

WDT 的功能归纳如下：

- 16 位看门狗计数器
- 工作频率可选： $f_{IN}/256$  或  $f_{IN}/16384$
- 定时器溢出错误检测
- 单独禁止定时器功能
- 双复位检测

看门狗定时器的寄存器归纳见图 8-32。

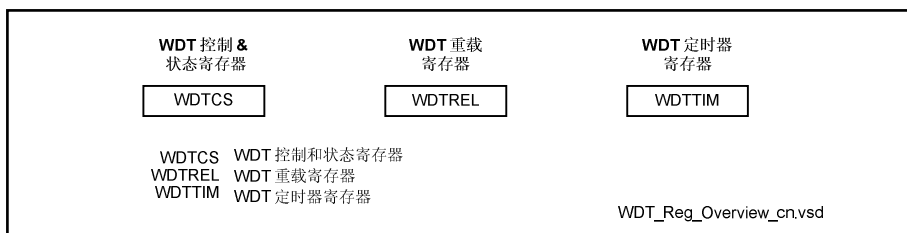


图 8-32 看门狗定时器寄存器概览

### 8.12.2 概述

WDT 提供了一种非常安全可靠的方法来检测软件和硬件故障并使系统从故障中恢复。WDT 可在用户设定的周期内中止 XE16xyM 的偶发故障。若在用户规定的时间周期内未服务 WDT，WDT 将使 XE16xyM 系统复位。为了防止 WDT 复位，CPU 必须在该时间间隔内服务 WDT。因此，规律的服务 WDT 可确保系统功能正常。

XE16xyM WDT 还增加了复位预警操作。当它检测到错误时，不象标准看门狗那样立即复位系统，而是首先通过一个中断请求发送预警输出。这种预警机制使得系统在复位之前进入一个确定、可预测的状态。

### 8.12.3 功能描述

下面将详细描述 WDT 的所有功能。

#### 8.12.3.1 看门狗定时器的操作

正确执行指令 ENWDT（使能看门狗定时器）后，WDT 被使能。

WDT 使用输入时钟  $f_{IN}$ （= 系统时钟  $f_{SYS}$ ）。经过时钟分频，产生两个输出频率  $f_{IN}/256$  和  $f_{IN}/16384$ 。由位 **WDTCS.IR** 选择计数频率。

##### WDT 周期

看门狗定时器周期的计算公式如下：

$$\text{周期} = \frac{(2^{16} - \text{startvalue}) \times 256 \times 2^{(1-IR) \times 6}}{f_{IN}} \quad (8.4)$$

计算正常模式下的 WDT 周期时，**startvalue**（起始值）代表用户可编程的重载值 **WDTREL.RELV**（缺省为 **FFFC<sub>H</sub>**）；计算预警模式下的 WDT 周期时，**startvalue** 为固定值 **FFFF<sub>H</sub>**。

##### WDT 定时器重载

发生以下情况时，会重载计数器并清除预分频器。

- 成功访问寄存器 **WDTREL**
- 通过指令 **SRVWDT** 服务 WDT
- WDT 溢出（进入预警模式）  
预警模式下计数器的重载值为 **FFFF<sub>H</sub>**
- 进入禁用模式（当执行 **DISWDT** 指令时）
- 任何复位操作

#### 8.12.3.2 看门狗定时器的工作模式

看门狗定时器可工作在以下三种模式：

- 正常模式
- 禁用模式
- 预警模式

**图 8-33** 给出 WDT 的不同工作模式及模式转换状态图。模式转换条件的具体描述如下。

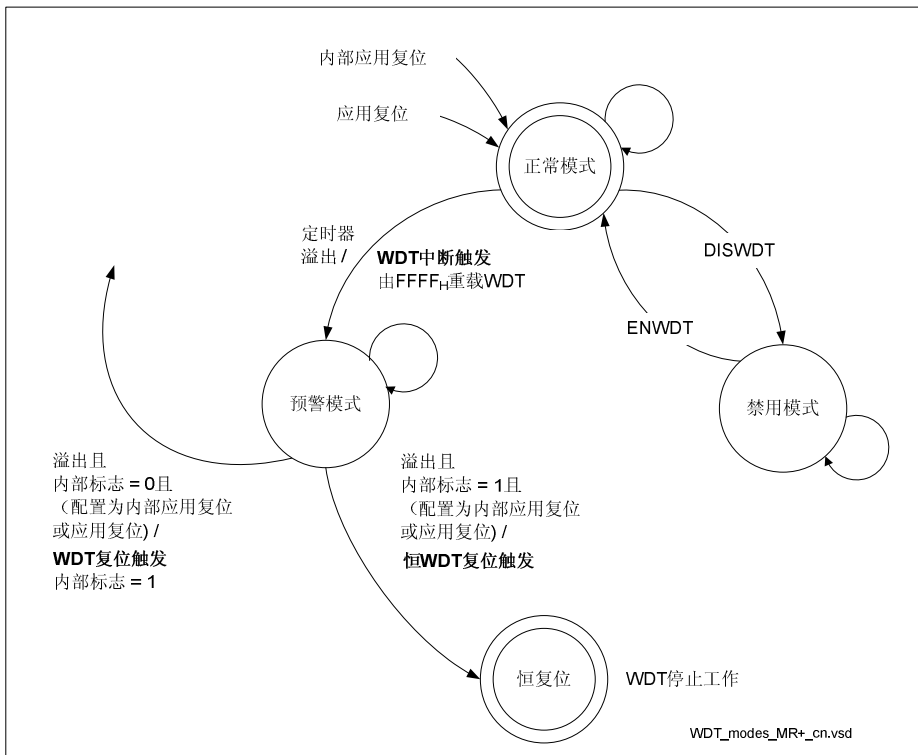


图 8-33 WDT 工作模式状态图

### 正常模式

正常模式是应用复位或内部应用复位后的缺省模式。只有当执行 ENWDT 指令时，才能从禁用模式进入正常模式。

进入正常模式后，定时器由 RELV 加载，随后开始递增计数。复位后，定时器由 FFFCH (RELV 的缺省值) 加载。

计数溢出前必须服务 WDT。CPU 通过指令 SRWDT 和/或 ENWDT 服务 WDT。定时器溢出前若未服务 WDT，则认为出现系统故障，此时执行以下操作：

- 发送 WDT 中断请求
- 进入预警模式
- 由 FFFF<sub>H</sub> 加载定时器

### 禁用模式

对于完全不需要 WDT 功能的应用，可采用禁用模式。执行指令 DISWDT 进入禁用模式：若 CPUCON1.WDTCTL = 0，在初始化结束前执行该指令；若 CPUCON1.WDTCTL = 1，可在任意时刻执行该指令。

进入禁用模式后，定时器由 WDTREL.RELV 的值加载。

若在 CPUCON1.WDTCTL = 1 时执行 ENWDT 指令，将从禁用模式转入正常模式。

### 预警模式

检测到看门狗错误时，始终进入预警模式。若正常模式下定时器溢出，则进入预警模式。此时，XE16xyM WDT 不会立即请求对芯片复位，它会首先发送预警输出，确保系统在复位之前进入安全的状态。接收到预警信号后，CPU 和系统完成当前挂起的操作请求，不产生新的操作请求。通过中断告知 CPU 出现预警，CPU 可通过寄存器 INTSTAT 识别 WDT 预警中断。完成所有挂起的操作后，在复位之前 CPU 应执行 IDLE 指令以终止所有进一步的操作。

预警模式下，WDT 从 FFFF<sub>H</sub> 开始递增计数，从 FFFF<sub>H</sub> 翻转到 0000<sub>H</sub> 溢出后请求 WDT 复位。不能避免产生由 RSTCON1.WDT 配置的复位请求。若 RSTCON1.WDT 被清零，则不产生复位请求。在复位之前，WDT 不再响应对其寄存器的访问、不再响应指令 ENWDT 或 DISWDT，也不能改变其状态。

WDT 还有一个特性：检测到两次错误后，整个系统将进入恒 WDT 复位。该特性可避免 XE16xyM 长时间（超过溢出间隔）执行错误代码，还可避免 XE16xyM 重复由 WDT 复位。

### 双 WDT 复位

如果看门狗引发两次复位（应用或内部应用复位），则认为出现了严重的系统故障，XE16xyM 保持由 RSTCON1.WDT 配置的复位状态，直至发生上电复位。例如，当器件和外部存储器的连接丢失，以致于无法完成系统初始化时，器件会被周期性复位。这种双复位检测方式可防止各种情况引起的器件周期性复位。

*注：WDT 复位触发的 PORST 决不会导致 WDT 双溢出。*

若 RSTCON1.WDT 配置为请求应用复位或内部应用复位，第二个复位请求将恒定生效（复位配置无任何改变）以引发由 RSTCON.WDT 配置的复位。

第一个 WDT 复位请求的相关信息保存在内部标志中：预警模式下发生 WDT 溢出并产生复位请求时，该标志置位。若内部标志已置位，则已发生 WDT 双复位事件，产生恒复位请求。

该内部标志由上电复位清零、或当 WDTCS.CLRIRF 置位时被清零。正确服务 WDT 不清除该标志，也不访问 WDT 的相关寄存器。因此，若第一次 WDT 复位之后已

正确服务 WDT，下一次 WDT 复位一定不会立即引发双错状态，应用软件必须清除内部标志。

*注：在处理内部标志时，无需考虑是由 WDT 复位请求产生了复位还是在两次复位请求之间改变了复位配置。*

*注：产生双 WDT 复位请求之后，计数器在溢出后停止工作。*

### WDT 复位期间的端口配置

不同复位类型所对应的 ESRx 端口的行为由位域 ESRCFGx.PC 定义。PC 编码见表 8-6。

GPIO 端口的配置和复位类型有关。若发生应用复位，引出端的配置不变；如发生内部应用复位，端口需配置为输入口。

#### 8.12.3.3 挂起模式支持

在调试阶段，看门狗功能可导致意外系统复位。为了避免这些复位，OCDS 可通过 CBS\_IOSR.DB 控制使能或禁用（复位后的缺省状态）WDT。

**表 8-11 OCDS WDT 的操作**

WDTCS.DS	CBS_DBGSR.DBGEN	CBS_IOSR.DB	WDT 操作
1	X	X	停止
0	0	X	工作
0	1	0	停止
0	1	1	工作

### 8.12.4 WDT 内核寄存器

#### 8.12.4.1 WDT 重载寄存器

该寄存器定义 WDT 的重载值。

#### WDTREL

**WDT 重载寄存器**

**ESFR (F0C8<sub>H</sub>/64<sub>H</sub>)**

**复位值: FFFC<sub>H</sub>**

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

rw

符号	位序号	类型	功能描述
<b>RELV</b>	[15:0]	rw	看门狗定时器的重载值 该位域定义 WDT 的重载值。

#### 8.12.4.2 WDT 控制和状态寄存器

只能在安全模式下访问控制和状态寄存器。

##### WDTCs

**WDT 控制和状态寄存器**

**ESFR (F0C6<sub>H</sub>/63<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
0							IR	0					CLR IRF	PR	DS	OE	
r							rw		r					w	rh	rh	rh

符号	位序号	类型	功能描述
<b>OE</b>	0	rh	<b>溢出错误状态标志</b> 0 <sub>B</sub> 无 WDT 溢出错误 1 <sub>B</sub> 发生 WDT 溢出错误 看门狗定时器溢出时 (从 FFFF <sub>H</sub> 到 0000 <sub>H</sub> )，该位由硬件置位。 只能通过以下方式对该位清零： <ul style="list-style-type: none"> <li>上电复位</li> <li>正确执行 SRVWDT 或 ENWDT 指令</li> </ul> <i>注：在启动过程中，始终由指令 ENWDT 使能 WDT (见“看门狗定时器处理”一节的描述)。因此，发生应用复位或内部应用复位时，该位被清零。</i> <i>注：在预警模式下，不可能使用 SRVWDT 或 ENWDT 指令清除该位。</i>
<b>DS</b>	1	rh	<b>定时器使能/禁用状态标志</b> 0 <sub>B</sub> 定时器被使能 (复位后的缺省状态) 1 <sub>B</sub> 定时器被禁用

符号	位序号	类型	功能描述
			<p>执行 ENWDT 指令且 CPUCON1.WDTCTL=1 时, 该位被清零。</p> <p>执行 DISWDT 指令或 CPUCON1.WDTCTL=1 时, 该位被置位。</p> <p><i>注: ENWDT 和 DISWDT 指令将反映在该位上, 但在预警模式下, WDT 模式不改变。</i></p>
<b>PR</b>	2	rh	<p><b>预警模式标志</b></p> <p>0<sub>B</sub> 正常模式 (复位后的缺省状态)</p> <p>1<sub>B</sub> 预警模式</p> <p>执行 ENWDT 指令后, 该位被清零。</p> <p>执行 DISWDT 指令后, 该位被置位。</p>
<b>CLRIRF</b>	3	w	<p><b>清除内部复位标志</b></p> <p>通过该位请求清除保存第一个 WDT 复位请求信息的内部标志。</p> <p>0<sub>B</sub> 无操作</p> <p>1<sub>B</sub> 请求清除内部复位标志</p> <p><i>注: 读取该位始终返回 0。</i></p>
<b>IR</b>	8	rw	<p><b>输入频率请求位</b></p> <p>0<sub>B</sub> 请求将输入频率设置为 <math>f_{IN}/16384</math></p> <p>1<sub>B</sub> 请求将输入频率设置为 <math>f_{IN}/256</math></p> <p>在下次成功执行指令 SRVWDT 或 ENWDT 后、当设置寄存器 WDTREL 时、以及 WDT 处于禁用模式时, 使用该位的更新值。</p>
<b>0</b>	[7:4], [15:9]	r	<p><b>保留</b></p> <p>读操作返回 0; 应写入 0。</p>



### 8.12.4.3 WDT 定时器寄存器

#### WDTTIM

WDT 定时器寄存器

ESFR (F0CA<sub>H</sub>/65<sub>H</sub>)

复位值: FFFC<sub>H</sub>

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

符号	位序号	类型	功能描述
TIM	[15:0]	rh	定时器的值 该位域指示 WDT 的当前值。

### 8.13 SCU 强制中断的产生

SCU 强制中断的基本结构如 图 8-34 所示。

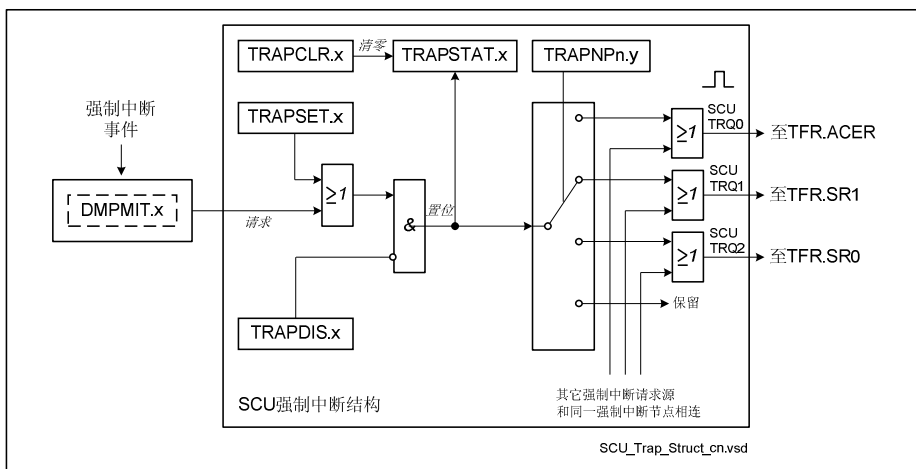


图 8-34 SCU 强制中断的结构

若寄存器 TRAPDIS 使能产生强制中断，可通过强制中断请求线或软件置位寄存器 TRAPSET 触发强制中断。强制中断请求源置位寄存器 TRAPSTAT 中的相应标志位，将该请求送至某个强制中断节点（由节点指针寄存器 TRAPNP 或 TRAPNP1 选择）。

可通过置位寄存器 TRAPCLR 软件清除寄存器 TRAPSTAT 中的强制中断标志。

若多个强制中断请求源连接到同一个强制中断上，这些请求源最终组合成一条公共的请求线。

### 强制中断节点的分配

系统的强制中断请求源可被映射到三个强制中断节点上，通过编程强制中断节点指针寄存器 TRAPNP 或 TRAPNP1 来实现。强制中断请求源的缺省节点分配归纳见表 8-12。

#### 8.13.1 强制中断支持

一些中断请求首先被送入 DMP\_M 电源域内的粘着标志寄存器。这些标志由触发置位而且一旦置位会触发电源域 DMP\_1 产生中断。

哪些强制中断请求源在寄存器 DMPMIT 中具有粘着标志归纳见表 8-12。

*注：响应 SCU 强制中断请求时，确保在处理完相应的请求后清除所有相关请求标志。要清除保存在寄存器 DMPMIT 中的中断请求时，首先清除相应的中断请求源，然后通过 DMPMITCLR 清除 DMP\_M 中的请求，之后再通过 TRAPCLR 清除 DMP\_1 中的请求。*

#### 8.13.2 SCU 强制中断请求源

SCU 的强制中断请求源归纳见表 8-12。

**表 8-12 SCU 强制中断请求源概览**

强制中断请求源	缩略名	DMPMIT 中的粘着标志	寄存器 TFR 中的缺省强制中断标志分配
Flash 访问强制中断	FAT	--	TFR.ACER (SCU_TRQ0)
ESR0 强制中断	ESR0T	有	TFR.SR1 (SCU_TRQ1)
ESR1 强制中断	ESR1T	有	TFR.SR1 (SCU_TRQ1)
ESR2 强制中断	ESR2T	有	TFR.SR1 (SCU_TRQ1)
PLL 强制中断	OSCWDTT	--	TFR.SR0 (SCU_TRQ1)
寄存器访问强制中断	RAT	有	TFR.ACER (SCU_TRQ0)
奇偶校验错误强制中断	PET	--	TFR.ACER (SCU_TRQ0)

**系统控制单元 (SCU)**

强制中断请求源	缩略名	DMPMIT 中的粘着标志	寄存器 TFR 中的缺省强制中断标志分配
VCO 锁相强制中断	VCOLCKT	--	TFR.SR0 (SCU_TRQ2)
ECC 错误强制中断	ECCT	--	TFR.ACER (SCU_TRQ0)

### 8.13.3 SCU 强制中断控制寄存器

#### 8.13.3.1 寄存器 TRAPSTAT

该寄存器中存放 SCU 所有强制中断请求源的状态标志。可分别通过寄存器 TRAPSET 和 TRAPCLR 软件置位和清除这些状态位。

#### TRAPSTAT

强制中断状态寄存器

SFR (FF02<sub>H</sub>/81<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						ECC T	VCO LCKT	PET	RAT	OSC WDTT	ESR 2T	ESR 1T	ESR 0T	FAT	
r						rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

符号	位序号	类型	功能描述
<b>FAT</b>	0	rh	<b>Flash 访问强制中断请求标志</b> 当发生 Flash 访问错误且 TRAPDIS.FAT = 0 时, TRAPSTAT.FAT 被置位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 FA 强制中断请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 FA 强制中断请求。
<b>ESR0T</b>	1	rh	<b>ESR0 强制中断请求标志</b> 当 DMPMIT.ESR0T 置位且 TRAPDIS.ESR0T = 0 时, TRAPSTAT.ESR0T 被置位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 ESR0 强制中断请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 ESR0 强制中断请求。

符号	位序号	类型	功能描述
<b>ESR1T</b>	2	rh	<p><b>ESR1 强制中断请求标志</b></p> <p>当 DMPMIT.ESR1T 置位且 TRAPDIS.ESR1T = 0 时, TRAPSTAT.ESR1T 被置位。</p> <p>0<sub>B</sub> 自该位被上次清零后, 未发生 ESR1 强制中断请求。</p> <p>1<sub>B</sub> 自该位被上次清零后, 已发生 ESR1 强制中断请求。</p>
<b>ESR2T</b>	3	rh	<p><b>ESR2 强制中断请求标志</b></p> <p>当 DMPMIT.ESR2T 置位且 TRAPDIS.ESR2T = 0 时, TRAPSTAT.ESR2T 被置位。</p> <p>0<sub>B</sub> 自该位被上次清零后, 未发生 ESR2 强制中断请求。</p> <p>1<sub>B</sub> 自该位被上次清零后, 已发生 ESR2 强制中断请求。</p>
<b>OSCWDTT</b>	4	rh	<p><b>OSCWDT 强制中断请求标志</b></p> <p>当发生 OSCWDT 紧急事件且 TRAPDIS.OSCWDTT= 0 时, TRAPSTAT.OSCWDTT 被置位。</p> <p>0<sub>B</sub> 自该位被上次清零后, 未发生 OSCWDT 强制中断请求。</p> <p>1<sub>B</sub> 自该位被上次清零后, 已发生 OSCWDT 强制中断请求。</p>
<b>RAT</b>	5	rh	<p><b>寄存器访问强制中断请求标志</b></p> <p>当 DMPMIT.RAT 置位且 TRAPDIS.RAT = 0 时, TRAPSTAT.RAT 被置位。</p> <p>0<sub>B</sub> 自该位被上次清零后, 未发生 RA 强制中断请求。</p> <p>1<sub>B</sub> 自该位被上次清零后, 已发生 RA 强制中断请求。</p>
<b>PET</b>	6	rh	<p><b>奇偶校验错误强制中断请求标志</b></p> <p>当发生存储器奇偶校验错误且 TRAPDIS.PET= 0 时, TRAPSTAT.PET 被置位。</p> <p>0<sub>B</sub> 自该位被上次清零后, 未发生 PE 强制中断请</p>

符号	位序号	类型	功能描述
			求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 PE 强制中断请求。
<b>VCOLCKT</b>	7	rh	<b>VCOLCK 强制中断请求标志</b> 当发生 VCOLCK 紧急事件且 TRAPDIS.VCOLCKT= 0 时, TRAPSTAT. VCOLCKT 被置位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 VCOLCK 强制 中断请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 VCOLCK 强制 中断请求。
<b>ECCT</b>	8	rh	<b>ECC 错误强制中断请求标志</b> 当发生存储器 (SRAM) ECC 错误且 TRAPDIS.ECCT= 0 时, TRAPSTAT.ECCT 被置 位。 0 <sub>B</sub> 自该位被上次清零后, 未发生 ECC 强制中断 请求。 1 <sub>B</sub> 自该位被上次清零后, 已发生 ECC 强制中断 请求。
<b>0</b>	[15:9]	r	<b>保留</b> 读操作返回 0; 应写入 0。

### 8.13.3.2 寄存器 TRAPCLR

可通过该寄存器软件清除寄存器 TRAPSTAT 中的强制中断状态标志。对该寄存器清零不起效, 读取该寄存器始终返回 0。

#### TRAPSTAT

**强制中断清除寄存器**

**SFR (FE8EH/47H)**

**复位值: 0000H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>							<b>ECC T</b>	<b>VCO LCKT</b>	<b>PET</b>	<b>RAT</b>	<b>OSC WDTT</b>	<b>ESR 2T</b>	<b>ESR 1T</b>	<b>ESR 0T</b>	<b>FAT</b>
r							w	w	w	w	w	w	w	w	w

符号	位序号	类型	功能描述
<b>FAT</b>	0	w	清除 <b>Flash</b> 访问强制中断请求标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志 TRAPSTAT.FAT
<b>ESR0T</b>	1	w	清除 <b>ESR0</b> 强制中断请求标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志 TRAPSTAT.ESR0T
<b>ESR1T</b>	2	w	清除 <b>ESR1</b> 强制中断请求标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志 TRAPSTAT.ESR1T
<b>ESR2T</b>	3	w	清除 <b>ESR2</b> 强制中断请求标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志 TRAPSTAT.ESR2T
<b>OSCWDTT</b>	4	w	清除 <b>OSCWDT</b> 强制中断请求标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志 TRAPSTAT.OSCWDTT
<b>RAT</b>	5	w	清除寄存器访问强制中断请求标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志 TRAPSTAT.RAT
<b>PET</b>	6	w	清除奇偶校验错误强制中断请求标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志 TRAPSTAT.PET
<b>VCOLCKT</b>	7	w	清除 <b>VCOLCK</b> 强制中断请求标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志 TRAPSTAT.VCOLCKT
<b>ECCT</b>	8	w	清除 <b>ECC</b> 错误强制中断请求标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志 TRAPSTAT.ECCT

符号	位序号	类型	功能描述
<b>0</b>	[15:9]	r	<b>保留</b> 读操作返回 0；应写入 0。

*注：读取这些位始终返回 0。*

### 8.13.3.3 寄存器 TRAPSET

可通过该寄存器软件置位寄存器 TRAPSTAT 中的强制中断状态标志。对该寄存器清零不起效，读取该寄存器始终返回 0。

#### TRAPSET

**强制中断置位寄存器**

**SFR (FE8C<sub>H</sub>/46<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>							<b>ECC T</b>	<b>VCO LCKT</b>	<b>PET</b>	<b>RAT</b>	<b>OSC WDTT</b>	<b>ESR 2T</b>	<b>ESR 1T</b>	<b>ESR 0T</b>	<b>FAT</b>
r							w	w	w	w	w	w	w	w	w

符号	位序号	类型	功能描述
<b>FAT</b>	0	w	<b>置位 Flash 访问强制中断请求标志</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位标志 TRAPSTAT.FAT
<b>ESR0T</b>	1	w	<b>置位 ESR0 强制中断请求标志</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位标志 TRAPSTAT.ESR0T
<b>ESR1T</b>	2	w	<b>置位 ESR1 强制中断请求标志</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位标志 TRAPSTAT.ESR1T
<b>ESR2T</b>	3	w	<b>置位 ESR2 强制中断请求标志</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位标志 TRAPSTAT.ESR2T
<b>OSCWDTT</b>	4	w	<b>置位 OSCWDT 强制中断请求标志</b> 0 <sub>B</sub> 无操作

符号	位序号	类型	功能描述
			1 <sub>B</sub> 置位标志 TRAPSTAT.OSCWDTT
<b>RAT</b>	5	w	<b>置位寄存器访问强制中断请求标志</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位标志 TRAPSTAT.RAT
<b>PET</b>	6	w	<b>置位奇偶校验错误强制中断请求标志</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位标志 TRAPSTAT.PET
<b>VCOLCKT</b>	7	w	<b>置位 VCOLCK 强制中断请求标志</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位标志 TRAPSTAT.VCOLCKT
<b>ECCT</b>	8	w	<b>置位 ECC 错误强制中断请求标志</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位标志 TRAPSTAT.ECCT
<b>0</b>	[15:9]	r	<b>保留</b> 读操作返回 0；应写入 0。

注：读取这些位始终返回 0。

### 8.13.3.4 寄存器 TRAPDIS

该寄存器中存放所有强制中断请求源的软件禁用控制位。需要注意，该寄存器中的位 ESRxT 和 RAT 也控制寄存器 DMPMIT 中的相应位（见[章节 8.10.1](#)）。

#### TRAPSET

**强制中断禁用寄存器**

**SFR (FE90<sub>H</sub>/48<sub>H</sub>)**

**复位值: 009E<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							ECC T	VCO LCKT	PET	RAT	OSC WDTT	ESR 2T	ESR 1T	ESR 0T	FAT
r							rW	rW	rW	rW	rW	rW	rW	rW	rW



符号	位序号	类型	功能描述
<b>FAT</b>	0	rw	<b>禁用 Flash 访问强制中断请求</b> 0 <sub>B</sub> 使能 FA 强制中断请求 1 <sub>B</sub> 禁用 FA 强制中断请求
<b>ESR0T</b>	1	rw	<b>禁用 ESR0 强制中断请求</b> 0 <sub>B</sub> 使能 ESR0 强制中断请求 1 <sub>B</sub> 禁用 ESR0 强制中断请求
<b>ESR1T</b>	2	rw	<b>禁用 ESR1 强制中断请求</b> 0 <sub>B</sub> 使能 ESR1 强制中断请求 1 <sub>B</sub> 禁用 ESR1 强制中断请求
<b>ESR2T</b>	3	rw	<b>禁用 ESR2 强制中断请求</b> 0 <sub>B</sub> 使能 ESR2 强制中断请求 1 <sub>B</sub> 禁用 ESR2 强制中断请求
<b>OSCWDTT</b>	4	rw	<b>禁用 OSCWDT 强制中断请求</b> 0 <sub>B</sub> 使能 OSCWDT 强制中断请求 1 <sub>B</sub> 禁用 OSCWDT 强制中断请求
<b>RAT</b>	5	rw	<b>禁用寄存器访问强制中断请求</b> 0 <sub>B</sub> 使能 RA 强制中断请求 1 <sub>B</sub> 禁用 RA 强制中断请求
<b>PET</b>	6	rw	<b>禁用奇偶校验错误强制中断请求</b> 0 <sub>B</sub> 使能 PE 强制中断请求 1 <sub>B</sub> 禁用 PE 强制中断请求
<b>VCOLCKT</b>	7	rw	<b>禁用 VCOLCK 强制中断请求</b> 0 <sub>B</sub> 使能 VCOLCK 强制中断请求 1 <sub>B</sub> 禁用 VCOLCK 强制中断请求
<b>ECCT</b>	8	rw	<b>禁用 ECC 错误强制中断请求</b> 0 <sub>B</sub> 使能 ECC 强制中断请求 1 <sub>B</sub> 禁用 ECC 强制中断请求

符号	位序号	类型	功能描述
<b>0</b>	[15:9]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 8.13.3.5 寄存器 TRAPNP 和 TRAPNP1

这些寄存器用于控制 SCU 所有强制中断请求源的强制中断节点指针。

#### TRAPNP

**强制中断节点指针寄存器**

**SFR（FE92H/49H）**

**复位值: 8254H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>VCOLCK</b>	<b>PE</b>	<b>RA</b>	<b>OSCWDT</b>	<b>ESR2</b>	<b>ESR1</b>	<b>ESR0</b>	<b>FA</b>								
rw	rw	rw	rw	rw	rw	rw	rw								

符号	位序号	类型	功能描述
<b>FA</b>	[1:0]	rw	<b>Flash 访问强制中断的节点指针</b> TRAPNP.FA 选择 FAT 强制中断请求输出线。 00 <sub>B</sub> 选择请求输出 SCU_TRQ0（TFR.ACER） 01 <sub>B</sub> 选择请求输出 SCU_TRQ1（TFR.SR1） 10 <sub>B</sub> 选择请求输出 SCU_TRQ2（TFR.SR0） 11 <sub>B</sub> 保留，不使用该组合
<b>ESR0</b>	[3:2]	rw	<b>ESR0 强制中断的节点指针</b> TRAPNP.ESR0 选择 ESR0 强制中断请求输出线。 00 <sub>B</sub> 选择请求输出 SCU_TRQ0（TFR.ACER） 01 <sub>B</sub> 选择请求输出 SCU_TRQ1（TFR.SR1） 10 <sub>B</sub> 选择请求输出 SCU_TRQ2（TFR.SR0） 11 <sub>B</sub> 保留，不使用该组合
<b>ESR1</b>	[5:4]	rw	<b>ESR1 强制中断的节点指针</b> TRAPNP.ESR1 选择 ESR1 强制中断请求输出线。 00 <sub>B</sub> 选择请求输出 SCU_TRQ0（TFR.ACER） 01 <sub>B</sub> 选择请求输出 SCU_TRQ1（TFR.SR1） 10 <sub>B</sub> 选择请求输出 SCU_TRQ2（TFR.SR0）

符号	位序号	类型	功能描述
			11 <sub>B</sub> 保留, 不使用该组合
<b>ESR2</b>	[7:6]	rw	<b>ESR2 强制中断的节点指针</b> TRAPNP.ESR2 选择 ESR2 强制中断请求的输出。 00 <sub>B</sub> 选择请求输出 SCU_TRQ0 (TFR.ACER) 01 <sub>B</sub> 选择请求输出 SCU_TRQ1 (TFR.SR1) 10 <sub>B</sub> 选择请求输出 SCU_TRQ2 (TFR.SR0) 11 <sub>B</sub> 保留, 不使用该组合
<b>OSCWDT</b>	[9:8]	rw	<b>OSCWDT 强制中断的节点指针</b> TRAPNP.OSCWDT 选择 OSCWDT 强制中断请求输出线。 00 <sub>B</sub> 选择请求输出 SCU_TRQ0 (TFR.ACER) 01 <sub>B</sub> 选择请求输出 SCU_TRQ1 (TFR.SR1) 10 <sub>B</sub> 选择请求输出 SCU_TRQ2 (TFR.SR0) 11 <sub>B</sub> 保留, 不使用该组合
<b>RA</b>	[11:10]	rw	<b>寄存器访问强制中断的节点指针</b> TRAPNP.RA 选择 RAT 强制中断请求输出线。 00 <sub>B</sub> 选择请求输出 SCU_TRQ0 (TFR.ACER) 01 <sub>B</sub> 选择请求输出 SCU_TRQ1 (TFR.SR1) 10 <sub>B</sub> 选择请求输出 SCU_TRQ2 (TFR.SR0) 11 <sub>B</sub> 保留, 不使用该组合
<b>PE</b>	[13:12]	rw	<b>奇偶校验错误强制中断的节点指针</b> TRAPNP.PE 选择 PET 强制中断请求输出线。 00 <sub>B</sub> 选择请求输出 SCU_TRQ0 (TFR.ACER) 01 <sub>B</sub> 选择请求输出 SCU_TRQ1 (TFR.SR1) 10 <sub>B</sub> 选择请求输出 SCU_TRQ2 (TFR.SR0) 11 <sub>B</sub> 保留, 不使用该组合
<b>VCOLCK</b>	[15:14]	rw	<b>VCOLCK 强制中断的节点指针</b> TRAPNP.VCOLCK 选择 VCOLCK 强制中断请求输出线。 00 <sub>B</sub> 选择请求输出 SCU_TRQ0 (TFR.ACER)

符号	位序号	类型	功能描述
			01 <sub>B</sub> 选择请求输出 SCU_TRQ1 (TFR.SR1)
			10 <sub>B</sub> 选择请求输出 SCU_TRQ2 (TFR.SR0)
			11 <sub>B</sub> 保留, 不使用该组合

## TRAPNP1

**强制中断节点指针 1 寄存器**

**SFR (FE94<sub>H</sub>/4A<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														<b>ECC</b>	
r														rw	

符号	位序号	类型	功能描述
<b>ECC</b>	[1:0]	rw	<b>ECC 错误强制中断的节点指针</b> TRAPNP.ECC 选择 ECCT 强制中断请求输出线。 00 <sub>B</sub> 选择请求输出 SCU_TRQ0 (TFR.ACER) 01 <sub>B</sub> 选择请求输出 SCU_TRQ1 (TFR.SR1) 10 <sub>B</sub> 选择请求输出 SCU_TRQ2 (TFR.SR0) 11 <sub>B</sub> 保留, 不使用该组合
<b>0</b>	[15:2]	r	<b>保留</b> 读操作返回 0; 应写入 0。

## 8.14 存储器内容保护

XE16xyM 提供了两种监控片上 RAM 内容的机制 (此处不考虑 Flash 存储器):

- 纠错控制 (ECC)
- 奇偶校验

每次向 RAM 写入数据时, 将产生相应的保护位 (奇偶校验位或 ECC), 这些保护位连同用户数据一起保存在 RAM 中。

ECC 支持一位检错 (SED) 和一位纠错 (SEC)。FlexRay 模块还支持双位检错 (DED)。

对于标准 RAM 而言，由寄存器 MCHKCON 来选择期望的存储器保护模式；对于内嵌于外设模块的 RAM 而言，存储器保护模式固定不变。可用的存储器保护模式归纳见表 8-13。

表 8-13 可用的存储器保护机制

存储器	奇偶校验	ECC 检错	ECC 纠错	ECC 编码
程序 SRAM (PSRAM)	有	SED	SEC	每字节 4 位
数据 SRAM (DSRAM)	有	SED	SEC	每字节 4 位
双端口 SRAM (DPRAM)	有	SED	SEC	每字节 4 位
备用 RAM (SBRAM)	有	SED	SEC	每字节 4 位
USICx SRAM (UxRAM)	有	无	无	-
MultiCAN SRAM (MCRAM)	无	SED	SEC	每 32 位 7 位

注：缺省情况下禁用带有奇偶校验或 ECC 的存储器保护机制。

必须在使用 RAM 之前选择并使能期望的保护模式（见章节 8.14.1）。

若存储器保护出错，将产生相应的强制中断，具体描述见章节 8.13。

### 8.14.1 保护模式选择

上电复位后，存储器保护被禁用。必须在使用 RAM 之前选择并使能期望的保护模式。对于每个标准 RAM 区，均可选择采用奇偶校验保护或 ECC 保护。

通过寄存器 MCHKCON 为标准 RAM 区选择期望的保护模式。缺省选择 ECC 保护模式，但缺省禁用存储器保护。

#### ECC 保护模式

若选择 ECC 保护模式，每次向选定的 RAM 写入数据时，ECC 逻辑将产生 ECC 保护位，它们和数据一起被保存在 RAM 中。

寄存器 ECCCON 用作 ECC 检错的使能控制。

若使能 ECC 检错，在读取数据的过程中检测一位错误（SED）。数据被送至 CPU 之前，这些一位错误被自动纠正（SEC）。由寄存器 ECCSTAT 指示 ECC 错误。

若禁止 ECC 检错，读出的数据被直接送至 CPU，无错误标志置位。

### 奇偶校验保护模式

若选择奇偶校验保护模式，每次向选定的 RAM 写入数据时，奇偶校验逻辑将产生奇偶校验位，它们和数据一起被保存在 RAM 中。

寄存器 **PEEN** 用作奇偶校验的使能控制。

若使能奇偶校验，由寄存器 **PECON** 指示检测到奇偶校验错误。读出的数据不会被修改。

### 保护模式使能

要激活 RAM 保护，必须首先通过寄存器 **MCHKCON** 选择保护模式。

为某 RAM 存储区选定保护模式后，该存储区必须被初始化，从而产生正确的保护位以确保 RAM 被正确读取。

CPU 在访问存储器时可能会进行一些预读取操作（即推测会用到某些数据而提前读取）。在有些情况下，比如 **SBRAM** 接口的自动递增模式将在指针递增之后访问新的地址单元，这会导致数据访问超出应用实际所使用的 RAM 区域。因此，强烈建议用户在使用内容受保护的 RAM 之前对其进行初始化，从而避免不可预期的错误。

*注：有关如何激活存储器保护模式，请参见“准备激活存储器内容保护”一节的描述。*

## MCHKCON

### 存储器检查控制寄存器

**ESFR (F0DC<sub>H</sub>/6E<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												SEL SB	SEL PS	SEL DS	SEL DP
rw												rw	rw	rw	rw

符号	位序号	类型	功能描述
<b>SELDP</b>	0	rw	<b>为双端口存储器选择保护模式</b> 0 <sub>B</sub> 为 DPRAM 选择 ECC 模式 1 <sub>B</sub> 为 DPRAM 选择奇偶校验模式
<b>SELDS</b>	1	rw	<b>为数据 SRAM 选择保护模式</b> 0 <sub>B</sub> 为 DSRAM 选择 ECC 模式 1 <sub>B</sub> 为 DSRAM 选择奇偶校验模式

符号	位序号	类型	功能描述
<b>SELPS</b>	2	rw	<b>为程序 SRAM 选择保护模式</b> 0 <sub>B</sub> 为 PSRAM 选择 ECC 模式 1 <sub>B</sub> 为 PSRAM 选择奇偶校验模式
<b>SELSB</b>	3	rw	<b>为备用存储器选择保护模式</b> 0 <sub>B</sub> 为 SBRAM 选择 ECC 模式 1 <sub>B</sub> 为 SBRAM 选择奇偶校验模式
<b>0</b>	[15:4]	rw	<b>保留</b> 应写入 0。

#### 8.14.2 奇偶校验错误处理

向 RAM 写入数据时产生奇偶校验信息，该信息连同数据一起保存在 RAM 中。读取数据时检查该奇偶校验信息，若奇偶校验出错且使能强制中断（寄存器 PEEN），则产生强制中断请求（寄存器 PECON）。来自所有 RAM 的强制中断请求可逻辑组合并触发强制中断，其状态由寄存器 **TRAPSTAT** 中的位 PET 指示。

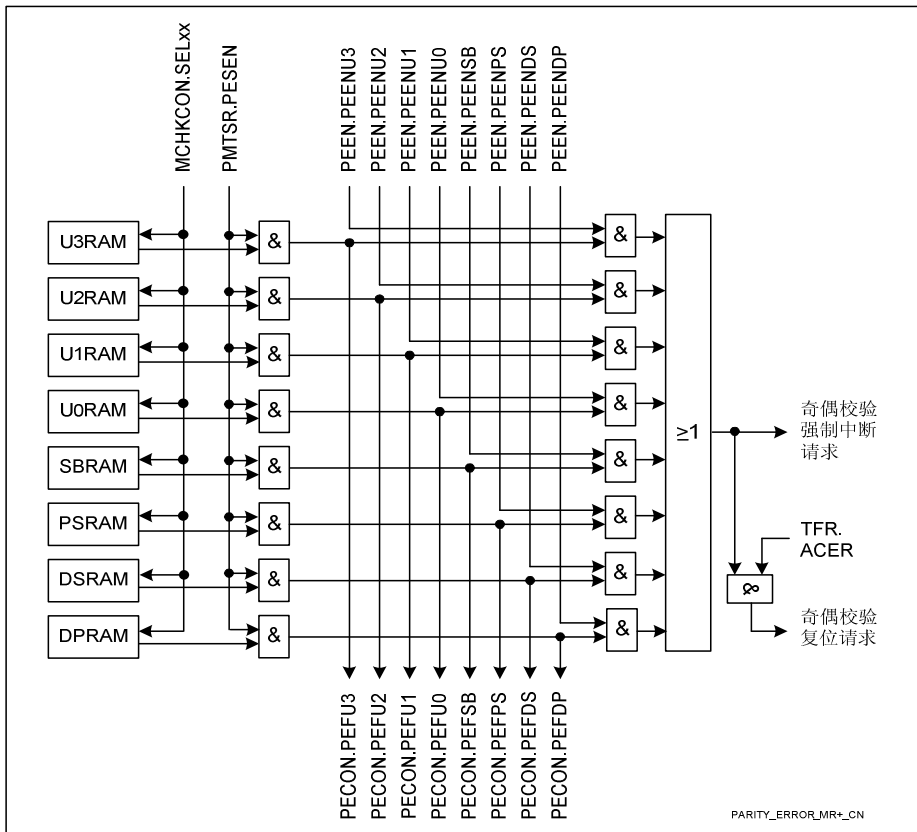


图 8-35 奇偶校验错误控制逻辑

若在强制中断标志 **TFR.ACER** 置位的情况下（即在执行强制中断服务程序的过程中）检测到奇偶校验错误，则产生复位请求。这是因为第二个错误强制中断将激活同一个强制中断处理，因此不能被 CPU 处理。

注：奇偶校验错误强制中断请求应激活访问错误强制中断（**ACER**）以支持该特性。

#### 8.14.2.1 奇偶校验软件测试支持

为了支持奇偶校验错误强制中断程序的测试算法，标准 RAM 存储区（PSRAM、DSRAM、DPRAM 和 SBRAM）实现了存储器奇偶校验测试逻辑。



## 系统控制单元（SCU）

该逻辑由寄存器 PMTPR 和 PMTSR 控制。若通过 PMTSR.MTE<sub>x</sub> 使能测试，可通过位域 PMTPR.PWR 将奇偶校验值写入 RAM 的任意地址单元。每次读访问该 RAM 存储区时，奇偶校验信息保存在寄存器 PMTPR 的位域 PRD 中。

**表 8-14** 给出寄存器 PMTPR 中有效的奇偶校验测试位，它们和奇偶校验位的宽度相关。

**表 8-14 有效的奇偶校验测试位**

存储器	奇偶校验位的位数	PWR/PRD 中的有效位
双端口（DP）存储器	2	PWR[1:0]/PRD[9:8]
数据 SRAM（DS）存储器	2	PWR[1:0]/PRD[9:8]
程序 SRAM（PS）存储器	8	PWR[7:0]/PRD[15:8]
备用 RAM（SB）存储器	2	PWR[1:0]/PRD[9:8]

测试软件应存放在片外存储器中，应以不执行预取操作的方式写该软件。

### 8.14.2.2 奇偶校验错误寄存器

寄存器 PEEN 用于分别使能不同 RAM 存储区的奇偶校验机制。

*注：寄存器 PMTSR 中的位 PESEN 全局使能奇偶校验机制。*

## PEEN

奇偶校验错误使能寄存器

ESFR (F0C4<sub>H</sub>/41<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							PEEN SB	0	PEEN U3	PEEN U2	PEEN U1	PEEN U0	PEEN PS	PEEN DS	PEEN DP
rw							rw	rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
PEENDP	0	rw	双端口存储器奇偶校验错误强制中断使能 0 <sub>B</sub> 禁止产生 DPRAM 奇偶校验错误强制中断 1 <sub>B</sub> 允许产生 DPRAM 奇偶校验错误强制中断
PEENDS	1	rw	数据 SRAM 奇偶校验错误强制中断使能 0 <sub>B</sub> 禁止产生 DSRAM 奇偶校验错误强制中断

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
			1 <sub>B</sub> 允许产生 DSRAM 奇偶校验错误强制中断
<b>PEENPS</b>	2	rw	<b>程序 SRAM 奇偶校验错误强制中断使能</b> 0 <sub>B</sub> 禁止产生 PSRAM 奇偶校验错误强制中断 1 <sub>B</sub> 允许产生 PSRAM 奇偶校验错误强制中断
<b>PEENU0</b>	3	rw	<b>USIC0 存储器奇偶校验错误强制中断使能</b> 0 <sub>B</sub> 禁止产生 USIC0 存储器奇偶校验错误强制中 断 1 <sub>B</sub> 允许产生 USIC0 存储器奇偶校验错误强制中 断
<b>PEENU1</b>	4	rw	<b>USIC1 存储器奇偶校验错误强制中断使能</b> 0 <sub>B</sub> 禁止产生 USIC1 存储器奇偶校验错误强制中 断 1 <sub>B</sub> 允许产生 USIC1 存储器奇偶校验错误强制中 断
<b>PEENU2</b>	5	rw	<b>USIC2 存储器奇偶校验错误强制中断使能</b> 0 <sub>B</sub> 禁止产生 USIC2 存储器奇偶校验错误强制中 断 1 <sub>B</sub> 允许产生 USIC2 存储器奇偶校验错误强制中 断
<b>PEENU3</b>	6	rw	<b>USIC3 存储器奇偶校验错误强制中断使能</b> 0 <sub>B</sub> 禁止产生 USIC3 存储器奇偶校验错误强制中 断 1 <sub>B</sub> 允许产生 USIC3 存储器奇偶校验错误强制中 断
<b>PEENSB</b>	8	rw	<b>备用存储器奇偶校验错误强制中断使能</b> 0 <sub>B</sub> 禁止产生 SBRAM 奇偶校验错误强制中断 1 <sub>B</sub> 允许产生 SBRAM 奇偶校验错误强制中断
<b>0</b>	7, [15:9]	rw	保留 应写入 0。

寄存器 PECON 用于控制奇偶校验机制。

若被使能，当检测到奇偶校验错误时，相应的错误标志置位。否则不指示错误。

可通过向标志位写 1 软件清除错误标志，写 0 不起作用。

## PECON

奇偶校验错误控制寄存器

ESFR (F0DA<sub>H</sub>/6D<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						PEF SB	0	PEF U3	PEF U2	PEF U1	PEF U0	PEF PS	PEF DS	PEF DP	
rwh						rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位序号	类型	功能描述
PEFDP	0	rwh	<b>双端口存储器奇偶校验错误标志</b> 0 <sub>B</sub> 无 DPRAM 奇偶校验错误 1 <sub>B</sub> 检测到 DPRAM 奇偶校验错误。若允许产生强制中断，可触发强制中断请求
PEFDS	1	rwh	<b>数据 SRAM 奇偶校验错误标志</b> 0 <sub>B</sub> 无 DSRAM 奇偶校验错误 1 <sub>B</sub> 检测到 DSRAM 奇偶校验错误。若允许产生强制中断，可触发强制中断请求
PEFPS	2	rwh	<b>程序 SRAM 奇偶校验错误标志</b> 0 <sub>B</sub> 无 PSRAM 奇偶校验错误 1 <sub>B</sub> 检测到 PSRAM 奇偶校验错误。若允许产生强制中断，可触发强制中断请求
PEFU0	3	rwh	<b>USIC0 存储器奇偶校验错误标志</b> 0 <sub>B</sub> 无 USIC0 存储器奇偶校验错误 1 <sub>B</sub> 检测到 USIC0 存储器奇偶校验错误。若允许产生强制中断，可触发强制中断请求
PEFU1	4	rwh	<b>USIC1 存储器奇偶校验错误标志</b> 0 <sub>B</sub> 无 USIC1 存储器奇偶校验错误 1 <sub>B</sub> 检测到 USIC1 存储器奇偶校验错误。若允许产生强制中断，可触发强制中断请求

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
<b>PEFU2</b>	5	rwh	<b>USIC2 存储器奇偶校验错误标志</b> 0 <sub>B</sub> 无 USIC2 存储器奇偶校验错误 1 <sub>B</sub> 检测到 USIC2 存储器奇偶校验错误。若允许产生强制中断，可触发强制中断请求
<b>PEFU3</b>	6	rwh	<b>USIC3 存储器奇偶校验错误标志</b> 0 <sub>B</sub> 无 USIC3 存储器奇偶校验错误 1 <sub>B</sub> 检测到 USIC3 存储器奇偶校验错误。若允许产生强制中断，可触发强制中断请求
<b>PEFSB</b>	8	rwh	<b>Standby 存储器奇偶校验错误标志</b> 0 <sub>B</sub> 无 SBRAM 奇偶校验错误 1 <sub>B</sub> 检测到 SBRAM 奇偶校验错误。若允许产生强制中断，可触发强制中断请求
<b>0</b>	7, [15:9]	rwh	保留 应写入 0。

**PMTPR**

奇偶校验存储器测试序列寄存器

**ESFR (F0E4<sub>H</sub>/72<sub>H</sub>)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PRD</b>								<b>PWR</b>							
rh								rw							

符号	位序号	类型	功能描述
<b>PRD</b>	[15:8]	rh	<b>存储器测试的奇偶校验读取值</b> 对于存储器模块中的每个字节，该位指示最近一次读访问所产生的奇偶校验位。
<b>PWR</b>	[7:0]	rw	<b>存储器测试的奇偶校验写入值</b> 对于存储器模块中的每个字节，该位存放下次写访问对应的奇偶校验位。

**PMTSR**

奇偶校验存储器测试选择寄存器

**ESFR (F0E6H/73H)**

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PES EN</b>	<b>0</b>					<b>MTEN SB</b>	<b>0</b>					<b>MTEN PS</b>	<b>MTEN DS</b>	<b>MTEN DP</b>	
rw	r					rw	rw					rw	rw	rw	

符号	位序号	类型	功能描述
<b>MTENDP</b>	0	rw	<b>双端口存储器的存储器测试使能控制</b> 控制测试复用器选择 DPRAM。 0 <sub>B</sub> 标准操作 1 <sub>B</sub> 使用测试奇偶校验位 (来自 PMTPR)
<b>MTENDS</b>	1	rw	<b>数据 SRAM 的存储器测试使能控制</b> 控制测试复用器选择 DSRAM。 0 <sub>B</sub> 标准操作 1 <sub>B</sub> 使用测试奇偶校验位 (来自 PMTPR)
<b>MTENPS</b>	2	rw	<b>程序 SRAM 的存储器测试使能控制</b> 控制测试复用器选择 PSRAM。 0 <sub>B</sub> 标准操作 1 <sub>B</sub> 使用测试奇偶校验位 (来自 PMTPR)
<b>MTENSB</b>	8	rw	<b>备用存储器的存储器测试使能控制</b> 控制测试复用器选择 SBRAM。 0 <sub>B</sub> 标准操作 1 <sub>B</sub> 使用测试奇偶校验位 (来自 PMTPR)
<b>PESEN</b>	15	rw	<b>奇偶校验错误敏感使能</b> 0 <sub>B</sub> 奇偶校验错误不起作用 1 <sub>B</sub> 产生奇偶校验错误并可触发强制中断请求 (若被使能)
<b>0</b>	[7:3]	rw	<b>保留</b> 应写入 0。

符号	位序号	类型	功能描述
0	[14:9]	r	保留 读操作返回 0；应写入 0。

注：寄存器 **PMTSR** 中同时只应置位某一位 **MTEN<sub>xx</sub>**，否则奇偶校验软件测试的结果不可信。

### 8.14.3 ECC 错误处理

向 RAM 写入数据时产生 ECC 信息，该信息连同数据一起保存在 RAM 中。读取数据时检查该 ECC 信息，若 ECC 出错且使能强制中断，则产生强制中断请求（寄存器 **ECCSTAT**）。来自所有 RAM 的强制中断请求可逻辑组合（寄存器 **ECCCON**）并触发强制中断，其状态由寄存器 **TRAPSTAT** 中的位 **ECCT** 指示。

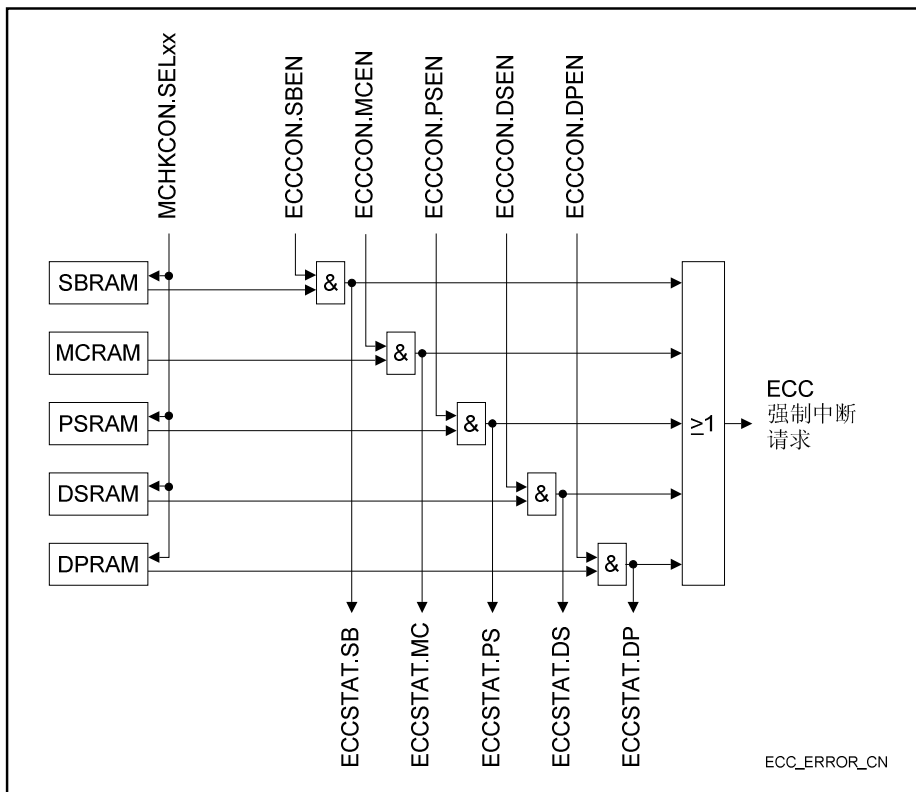


图 8-36 ECC 错误控制逻辑

注：对 Flash ECC 错误的处理不能由 SCU 完全完成，还需对 IMB 执行操作。

注：当使用自动递增模式读取 SBRAM 并使能 ECC 机制时，不仅要初始化已使用的地址，还需初始化最后一次用到的地址之后的一个地址。若整个 SBRAM 已被初始化，则不存在这个问题。

### 8.14.3.1 ECC 软件测试支持

可专门触发 ECC 检错以测试检错本身和相关的强制中断程序。标准 RAM 存储区（即 DPRAM、DSRAM、PSRAM 和 SBRAM）实现了 ECC 测试逻辑。

ECC 软件测试既使用 ECC 也使用奇偶校验。执行以下步骤可产生一位 ECC 错误：

- 为 RAM 选择奇偶校验模式 (MCHKCON.SELx = 1)
- 读回 MCHKCON 以使奇偶校验逻辑激活
- 选择奇偶校验的情况下向该 RAM 内的某个地址写入 2000<sub>H</sub>
- 为该 RAM 选择 ECC 模式 (MCHKCON.SELx = 0)
- 读回 MCHKCON 以使 ECC 逻辑激活
- 选择 ECC 的情况下读取被选中的 RAM 单元

这将会产生一位 ECC 错误（位 9）：

ECC 操作若被使能 (ECCCON.xEN = 1)，则读回已被纠错的数据值 (2000<sub>H</sub>) 并指示 ECC 出错。

ECC 操作若被禁止 (ECCCON.xEN = 0)，则读回未被纠错的数据值 (2200<sub>H</sub>)，不指示 ECC 出错。

注：由于 PSRAM 结构的特殊性，该测试将修改整个一条存储线（即 8 字节：...000<sub>B</sub> -...111<sub>B</sub>）。因此，非破坏性的测试必须保存并恢复相应存储线上的所有数据（4 个字）。

### 8.14.3.2 ECC 寄存器

#### 寄存器 ECCCON

#### ECCCON

#### ECC 控制寄存器

ESFR (F0A8<sub>H</sub>/54<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0										SB EN	MC EN	0	PS EN	DS EN	DP EN
rw										rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
<b>DPEN</b>	0	rw	双端口存储器 <b>ECC 检查使能</b> 0 <sub>B</sub> DPRAM 禁用 ECC 1 <sub>B</sub> DPRAM 使能 ECC
<b>DSEN</b>	1	rw	数据 <b>SRAM ECC 检查使能</b> 0 <sub>B</sub> DSRAM 禁用 ECC 1 <sub>B</sub> DSRAM 使能 ECC
<b>PSEN</b>	2	rw	程序 <b>SRAM ECC 检查使能</b> 0 <sub>B</sub> PSRAM 禁用 ECC 1 <sub>B</sub> PSRAM 使能 ECC
<b>MCEN</b>	4	rw	<b>MultiCAN 存储器 ECC 检查使能</b> 0 <sub>B</sub> MultiCAN 存储器禁用 ECC 1 <sub>B</sub> MultiCAN 存储器使能 ECC
<b>SBEN</b>	5	rw	备用存储器 <b>ECC 检查使能</b> 0 <sub>B</sub> SBRAM 禁用 ECC 1 <sub>B</sub> SBRAM 使能 ECC
<b>0</b>	3, [15:6]	rw	保留 应写入 0。

## ECCSTAT

### ECC 状态寄存器

**ESFR (F0AA<sub>H</sub>/55<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>										<b>SB</b>	<b>MC</b>	<b>0</b>	<b>PS</b>	<b>DS</b>	<b>DP</b>
rh										rh	rh	rh	rh	rh	rh

符号	位序号	类型	功能描述
<b>DP</b>	0	rh	双端口存储器 <b>ECC 错误状态</b> 0 <sub>B</sub> DPRAM 中未检测到 ECC 错误 1 <sub>B</sub> DPRAM 中检测到 ECC 错误



符号	位序号	类型	功能描述
<b>DS</b>	1	rh	<b>数据 SRAM ECC 错误状态</b> 0 <sub>B</sub> DSRAM 中未检测到 ECC 错误 1 <sub>B</sub> DSRAM 中检测到 ECC 错误
<b>PS</b>	2	rh	<b>程序 SRAM ECC 错误状态</b> 0 <sub>B</sub> PSRAM 中未检测到 ECC 错误 1 <sub>B</sub> PSRAM 中检测到 ECC 错误
<b>MC</b>	4	rh	<b>MultiCAN 存储器 ECC 错误状态</b> 0 <sub>B</sub> MultiCAN 存储器中未检测到 ECC 错误 1 <sub>B</sub> MultiCAN 存储器中检测到 ECC 错误
<b>SB</b>	5	rh	<b>备用存储器 ECC 错误状态</b> 0 <sub>B</sub> SBRAM 中未检测到 ECC 错误 1 <sub>B</sub> SBRAM 中检测到 ECC 错误
<b>0</b>	3, [15:6]	rh	保留 读取返回 0。

## ECCCLRSTAT

**ECC 状态清零寄存器**

**ESFR (F0DE<sub>H</sub>/6F<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>										<b>SB</b>	<b>MC</b>	<b>0</b>	<b>PS</b>	<b>DS</b>	<b>DP</b>
w										w	w	w	w	w	w

符号	位序号	类型	功能描述
<b>DP</b>	0	w	<b>清除双端口存储器 ECC 错误状态</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位该位将清除 ECCSTAT.DP
<b>DS</b>	1	w	<b>清除数据 SRAM ECC 错误状态</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位该位将清除 ECCSTAT.DS

符号	位序号	类型	功能描述
<b>PS</b>	2	w	清除程序 <b>SRAM ECC</b> 错误状态 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位该位将清除 ECCSTAT.PS
<b>MC</b>	4	w	清除 <b>MultiCAN</b> 存储器 <b>ECC</b> 错误状态 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位该位将清除 ECCSTAT.MC
<b>SB</b>	5	w	清除备用存储器 <b>ECC</b> 错误状态 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位该位将清除 ECCSTAT.SB
<b>0</b>	3, [15:6]	w	保留

注：读取这些位始终返回 0。

## 8.15 寄存器控制

采用以下一种保护模式对所有系统寄存器进行寄存器访问控制：

- 无保护模式
- 写保护模式
- 安全模式

### 8.15.1 寄存器访问控制

XE16xyM 中有一些控制关键系统功能和模式的专用寄存器。这些寄存器受特殊的寄存器安全机制保护，因而在执行 **EINIT** 指令后，重要的系统功能不会被无意修改。不过，由于这些寄存器控制重要的系统操作，因此需要在系统工作期间对其进行访问。系统控制软件通过特殊的安全状态机执行相关的访问操作。

若检测到非法访问，产生强制中断请求。

寄存器安全机制控制以下安全等级，可通过寄存器 **SLC** 来配置：

#### • 无保护模式

保护不激活。寄存器可随时被写访问。应用复位后进入该模式。

#### • 写保护模式

受保护寄存器被锁定、不能进行写访问。对这些寄存器的写访问无效。执行 **EINIT** 指令后自动进入该模式。

## • 安全模式

可使用特殊的指令对受保护寄存器进行写访问。受该模式保护的寄存器在表 8-23 中标记为“Sec”保护。

安全模式下，在执行期望的写访问之前先向寄存器 SLC 写入“命令 4”，此时保护机制失效，直到对 PD+总线上的寄存器（SFR、ESFR、XSFR 区）进行下一次写访问时保护机制被激活；也就是说，访问该区域之外的寄存器（如 CSFR）不会自动重新激活保护机制。因此，寄存器地址不同，写入“命令 4”之后的保护机制操作不同。通常使能对受保护寄存器进行单次写访问。执行该次写访问之后，受保护寄存器被自动重新锁定。此后，为了使能对受保护寄存器进行下一次访问，必须重新写入“命令 4”。对 CSFR 寄存器或 LXBus 外设寄存器（XLOC 区，如 USIC、CAN、IMB）进行写访问之后，寄存器保护机制不被重新激活。

*注：安全模式下，写入“命令 4”之后重新使能寄存器保护会导致不期望的操作。*

*若随后对受保护寄存器进行写访问，若此时正在处理中断且 ISR 本身使用“命令 4”机制，该写访问将延迟进行。在 ISR 中设置“命令 4”之后，期望重新激活保护机制（而不是使其失效），接下来的写操作将导致 ACER 强制中断。可使用 ATOMIC 指令使保护失效、写访问受保护寄存器。*

*若随后对不重新使能保护机制的寄存器进行写访问，成功执行该次写访问，但接下来向 SLC 寄存器写入“命令 4”之后，重新激活保护机制，不能对受保护寄存器进行写访问。*

受安全机制保护的寄存器除通常的访问参数（如只读、位读写类型 r 或 rh）之外，还有访问限制（由安全等级决定）。所有受保护寄存器均可读，这和安全等级无关。

### 8.15.1.1 控制安全等级

安全等级命令寄存器（SLC）和安全等级状态寄存器（SLS）用于控制安全等级。SLC 寄存器接受命令，从而控制状态机修改安全等级；SLS 寄存器给出实际的密码、实际的安全等级、以及状态机的当前状态。

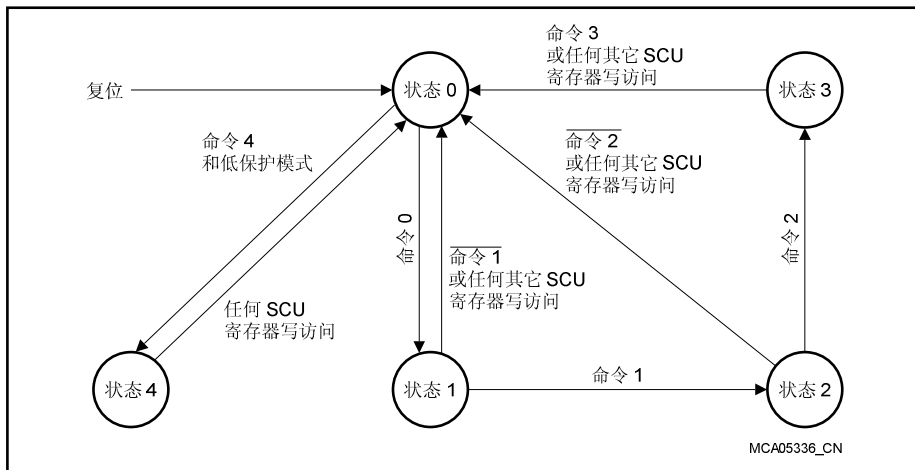


图 8-37 安全等级控制状态机

可使用以下机制控制实际的安全等级：

- 可通过执行以下命令序列**修改安全等级**：  
“命令 0–命令 1–命令 2–命令 3”。
- 该序列可设立新的安全等级和/或新密码。

表 8-15 安全等级控制命令

命令	定义	注
命令 0	AAAA <sub>H</sub>	-
命令 1	5554 <sub>H</sub>	-
命令 2	96 <sub>H</sub> + <sup>1)</sup> <密码取反>	-
命令 3	000 <sub>B</sub> + <新等级> + 000 <sub>B</sub> + <新密码>	-
命令 4	8E <sub>H</sub> + <密码取反>	只用于安全模式

1) "+"表示位域串联。

注：建议使用不可分指令序列（atomic sequence）锁定所有命令序列。

## 8.15.2 寄存器保护寄存器

### 寄存器 SLC

该寄存器是保护命令的接口。

#### SLC

安全等级命令寄存器

ESFR (F0C0<sub>H</sub>/60<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
COMMAND	[15:0]	rw	安全等级控制命令 必须将控制安全等级的命令写入该寄存器（见表 8-15）。

### 寄存器 SLS

该寄存器监控寄存器保护的状态。

#### SLS

安全等级状态寄存器

ESFR (F0C2<sub>H</sub>/61<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STATE		SL		0		PASSWORD									
rh		rh		r		rh									

符号	位序号	类型	功能描述
PASSWORD	[7:0]	rh	当前安全控制密码 复位后缺省值 = 00 <sub>H</sub>
SL	[12:11]	rh	安全等级 <sup>1)</sup> 00 <sub>B</sub> 无保护模式（缺省值） 01 <sub>B</sub> 安全模式 10 <sub>B</sub> 保留，不使用该组合

符号	位序号	类型	功能描述
			11 <sub>B</sub> 写保护模式
<b>STATE</b>	[15:13]	rh	<b>切换状态机的当前状态</b> 000 <sub>B</sub> 等待命令 0 或 命令 4 (缺省) 001 <sub>B</sub> 等待命令 1 010 <sub>B</sub> 等待命令 2 011 <sub>B</sub> 等待新的安全等级和密码 100 <sub>B</sub> 安全模式下许可的下次访问 101 <sub>B</sub> 保留, 不使用该组合 11X <sub>B</sub> 保留, 不使用该组合
<b>0</b>	[10:8]	r	<b>保留</b> 读操作返回 0; 应写入 0。

1) 复位后安全等级为“无保护”，执行指令 EINIT 后转换到“写保护”。

## 8.16 其它系统寄存器

本章对各种（不针对特定应用的）通用寄存器予以描述。

### 8.16.1 系统寄存器

#### 8.16.1.1 系统控制寄存器

SYSCON1 用于控制多种不同的系统任务。

#### SYSCON1

系统控制 1 寄存器

SFR (FF4C<sub>H</sub>/A6<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												GLC CST	OCDS EN	0	
r												rw		r	

符号	位序号	类型	功能描述
<b>OCDSSEN</b>	2	rw	<b>OCDS/Cerberus 使能</b> 0 <sub>B</sub> OCDS 和 Cerberus 仍处于复位状态

**系统控制单元 (SCU)**

符号	位序号	类型	功能描述
			1 <sub>B</sub> OCDS 和 Cerberus 当前可工作
<b>GLCCST</b>	3	rw	<b>全局 CAPCOM 启动</b> 该位同步启动所有 CAPCOM 单元（若被使能）。 0 <sub>B</sub> CAPCOM 定时器由各单元内部独立启动 1 <sub>B</sub> 所有 CAPCOM 定时器被同步启动 在执行新的 CAPCOM 启动之前，必须由软件对该位清零。
<b>0</b>	[1:0], [15:4]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 8.16.2 ID 模块

为了识别最重要的芯片参数，通过一组 ID 寄存器提供有关芯片制造商、芯片类型及属性的信息。

### 8.16.2.1 ID 寄存器

#### 寄存器 IDMANUF

该寄存器提供制造商信息。

#### IDMANUF

##### 制造商 ID 寄存器

**ESFR (F07E<sub>H</sub>/3F<sub>H</sub>)**

**复位值: 1820<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MANUF</b>											<b>DEPT</b>				
r											r				

符号	位序号	类型	功能描述
<b>DEPT</b>	[4:0]	r	<b>部门</b> 指示英飞凌的部门。 00 <sub>H</sub> AIM MC

符号	位序号	类型	功能描述
<b>MANUF</b>	[15:5]	r	<b>制造商</b> JEDEC 正规化的制造商代码。 0C1 <sub>H</sub> 英飞凌科技

## 寄存器 IDCHIP

该寄存器提供器件信息。

### IDCHIP

芯片 ID 寄存器

**ESFR (F07C<sub>H</sub>/3E<sub>H</sub>)**

复位值: XXXX<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CHIPID</b>								<b>Revision</b>							
r								r							

符号	位序号	类型	功能描述
<b>Revision</b>	[7:0]	r	<b>器件修订版编号</b> 指示器件修订版编号。 具体参数值请参见数据手册。
<b>CHIPID</b>	[15:8]	r	<b>器件 ID</b> 指示器件 ID。 具体参数值请参见数据手册。

## 寄存器 IDMEM

该寄存器提供程序存储器的信息。

### IDMEM

程序存储器 ID 寄存器

**ESFR (F07A<sub>H</sub>/3D<sub>H</sub>)**

复位值: 3XXX<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>TYPE</b>				<b>SIZE</b>											
r				rw											



符号	位序号	类型	功能描述
<b>SIZE</b>	[11:0]	rw	<b>片上程序存储器的大小</b> 所实现的程序存储器的容量以 4K 字节为单位，即 存储器大小 = <SIZE> * 4K 字节。 具体参数值请参见数据手册。
<b>TYPE</b>	[15:12]	r	<b>片上程序存储器的类型</b> 指示该芯片上的存储器类型。 具体参数值请参见数据手册。

## 寄存器 IDPROG

该寄存器提供 Flash 编程电压的信息。

## IDPROG

编程电压 ID 寄存器

ESFR (F078<sub>H</sub>/3C<sub>H</sub>)

复位值: 1313<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PROGVPP</b>								<b>PROGVDD</b>							

r

r

符号	位序号	类型	功能描述
<b>PROGVDD</b>	[7:0]	r	<b>编程 VDD 电压</b> 编程或擦除片上程序存储器所需的标准供电电压。 具体参数值请参见数据手册。
<b>PROGVPP</b>	[15:8]	r	<b>编程 VPP 电压</b> 编程或擦除片上程序存储器所需的特殊供电电压 （若可用）。 具体参数值请参见数据手册。

## 寄存器 IDDMPPM

该寄存器提供 DMP\_M SCU 的信息。

### IDDMPPM

**DMP\_M 模块 ID 寄存器**                      **SFR (FFE2<sub>H</sub>)**                      **复位值: 60XX<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD_TYPE</b>								<b>MOD_REV</b>							
r								r							

符号	位序号	类型	功能描述
<b>MOD_REV</b>	[7:0]	r	模块修订版编号 从 01 <sub>H</sub> 开始编号。
<b>MOD_TYPE</b>	[15:8]	r	模块 ID 编号 SCU_M = 60 <sub>H</sub>

## 寄存器 IDDMPP1

该寄存器提供 DMP\_1 SCU 的信息。

### IDDMPP1

**DMP\_1 模块 ID 寄存器**                      **SFR (FFE4<sub>H</sub>)**                      **复位值: 61XX<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD_TYPE</b>								<b>MOD_REV</b>							
r								r							

符号	位序号	类型	功能描述
<b>MOD_REV</b>	[7:0]	r	模块修订版编号 从 01 <sub>H</sub> 开始编号。
<b>MOD_TYPE</b>	[15:8]	r	模块 ID 编号 SCU_1 = 61 <sub>H</sub>

### 8.16.3 标记存储器

#### 8.16.3.1 标记存储器寄存器

标记存储器由位于 DMP\_M 中的以下 SFR 组成，用于自由使用用户软件。

##### **MKMEM0**

**标记存储器 0 寄存器**                      **SFR (FED0<sub>H</sub>/68<sub>H</sub>)**                      **复位值: 0000<sub>H</sub>**

##### **MKMEM1**

**标记存储器 1 寄存器**                      **SFR (FED2<sub>H</sub>/69<sub>H</sub>)**                      **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MARKER</b>															
rw															

符号	位序号	类型	功能描述
<b>MARKER</b>	[15:0]	rw	标记内容

### 8.17 实现

本节给出模块和系统的连接。

#### 8.17.1 时钟产生单元

下表给出时钟产生单元的输入连接（见 [章节 8.1](#)）。

**表 8-16      CGU 输入连接**

输入	连接至
XTAL 1	XTAL 1
XTAL 2	XTAL 2
CLKIN1	P2.9
CLKIN2	P4.4

#### 8.17.2 外部服务请求 (ESR)

引脚 ESR1 和 ESR2 是否可用和器件型号及封装有关，具体参见数据手册。

引脚  $\overline{\text{ESR0}}$  不和其它产品功能复用。

ESR1 和 ESR2 的输入连接分别归纳见 [表 8-17](#) 和 [表 8-18](#)。即使某个 ESRx 引脚不可用，表中列出的 ESRx 输入的复用功能仍可用。ESRx 逻辑部分的功能完备。

来自不可用 ESRx 引脚的输入连接到固定值（“1” – 由 ESRCFGx.PC 定义）。

**表 8-17 ESR1 输入连接**

输入	连接至
输入 0	P2.4
输入 1	P3.0
输入 2	P10.0
输入 3	P1.0
输入 4	P1.2
输入 5	P2.1
输入 6	P6.1
输入 7	P11.0
输入 8	P4.1
输入 9	P10.4
输入 10	P2.5
输入 11	P0.0

**表 8-18 ESR2 输入连接**

输入	连接至
输入 0	P2.3
输入 1	P7.0
输入 2	P10.14
输入 3	P1.1
输入 4	P1.3

系统控制单元 (SCU)

输入	连接至
输入 5	P2.2
输入 6	P2.6
输入 7	P2.7
输入 8	P0.4
输入 9	XTAL1
输入 10	P4.5
输入 11	保留，不连接

### 8.17.3 外部请求单元 (ERU)

ERU 的连接（见[章节 8.9](#)）和器件型号有关。XE16xyM 中 ERU 的具体连接描述如下。

#### 8.17.3.1 ERU 输入连接

ERU 和引脚直接相连、或和通信模块（如 USIC 或 MultiCAN）相连，ERU 的输入连接如[图 8-38](#)所示。这些通信模块在各自的模块内部选择提供给 ERU 的输入信号（如 CAN 接收输入、USIC 数据、时钟或控制输入）。

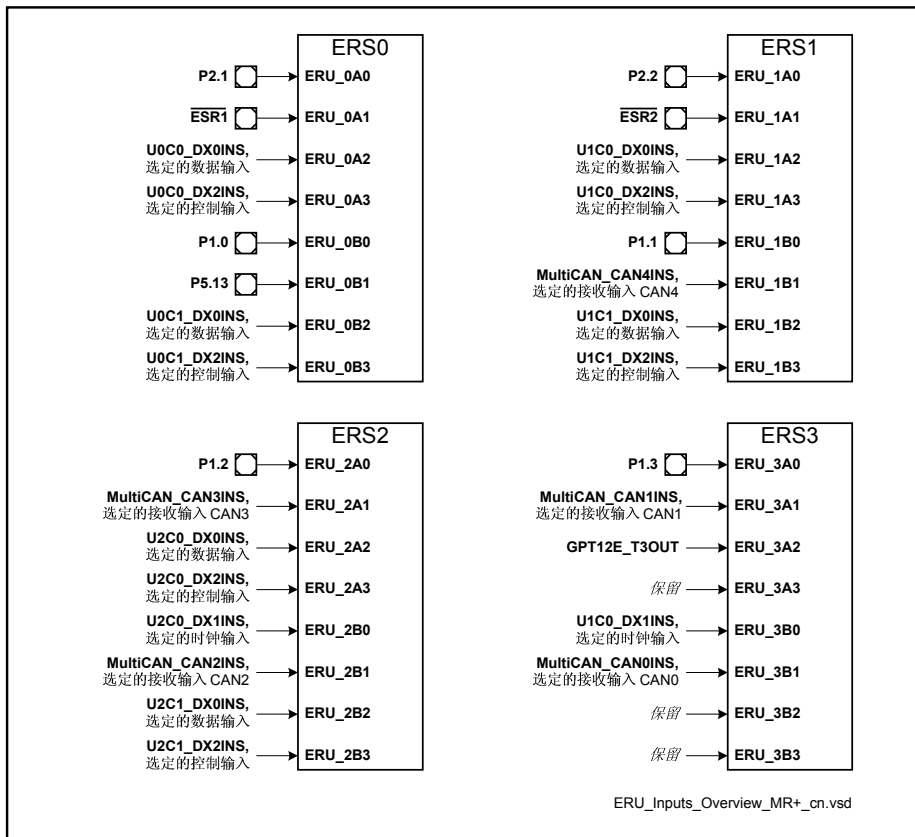


图 8-38 ERU 输入概览

ERSx 的输入连接归纳见下表。输入选择由寄存器 **EXISEL** 定义。

**表 8-19 XE16xyM 中 ERSx 的连接**

输入	来自/送至模块	ERSx 的 I/O 属性	可用于/用作
ERS0 输入			
ERU_0A0	P2.1	I	ERS0 输入 A
ERU_0A1	$\overline{\text{ESR1}}$	I	
ERU_0A2	U0C0_DX0INS	I	
ERU_0A3	U0C0_DX2INS	I	
ERU_0B0	P1.0	I	ERS0 输入 B
ERU_0B1	P5.13	I	
ERU_0B2	U0C1_DX0INS	I	
ERU_0B3	U0C1_DX2INS	I	
ERS1 输入			
ERU_1A0	P2.2	I	ERS1 输入 A
ERU_1A1	$\overline{\text{ESR2}}$	I	
ERU_1A2	U1C0_DX0INS	I	
ERU_1A3	U1C0_DX2INS	I	
ERU_1B0	P1.1	I	ERS1 输入 B
ERU_1B1	MultiCAN_CAN4INS	I	
ERU_1B2	U1C1_DX0INS	I	
ERU_1B3	U1C1_DX2INS	I	
ERS2 输入			
ERU_2A0	P1.2	I	ERS2 输入 A
ERU_2A1	MultiCAN_CAN3INS	I	
ERU_2A2	U2C0_DX0INS	I	
ERU_2A3	U2C0_DX2INS	I	
ERU_2B0	U2C0_DX1INS	I	ERS2 输入 B
ERU_2B1	MultiCAN_CAN2INS	I	

**系统控制单元 (SCU)**

输入	来自/送至模块	ERSx 的 I/O 属性	可用于/用作
ERU_2B2	U2C1_DX0INS	I	
ERU_2B3	U2C1_DX2INS	I	
ERS3 输入			
ERU_3A0	P1.3	I	ERS3 输入 A
ERU_3A1	MultiCAN_CAN1INS	I	
ERU_3A2	GPT12E_T3OUT	I	
ERU_3A3	0	I	
ERU_3B0	U1C0_DX1INS	I	ERS3 输入 B
ERU_3B1	MultiCAN_CAN0INS	I	
ERU_3B2	0	I	
ERU_3B3	0	I	

### 8.17.3.2 输出门控单元 (OGUy)

OGUy 单元外设触发信号的连接归纳见下表。

**表 8-20 XE16xyM 中 OGUy 外设触发信号的连接**

输入	来自/送至模块	OGUy 的 I/O 属性	可用于/用作
OGU0 输入			
ERU_OGU01	CCU60_MCM_ST	I	OGU0 的外设触发信号
ERU_OGU02	CCU60_T13_PM	I	
ERU_OGU03	CC2_28	I	
OGU1 输入			
ERU_OGU11	CCU61_MCM_ST	I	OGU1 的外设触发信号
ERU_OGU12	CCU61_T13_PM	I	
ERU_OGU13	CC2_29	I	
OGU2 输入			



**系统控制单元 (SCU)**

输入	来自/送至模块	OGUy 的 I/O 属性	可用于/用作
ERU_OGU21	CCU62_MCM_ST	I	OGU2 的外设触发信号
ERU_OGU22	CCU62_T13_PM	I	
ERU_OGU23	CC2_30	I	

**OGU3 输入**

ERU_OGU31	CCU63_MCM_ST	I	OGU3 的外设触发信号
ERU_OGU32	CCU63_T13_PM	I	
ERU_OGU33	CC2_31	I	

### 8.17.3.3 ERU 输出连接

下表给出 ERU 输出用于门控或触发其它模块功能的信号连接、以及和中断控制寄存器的连接。

**表 8-21 XE16xyM 中 ERU 的输出连接**

输出	来自/送至模块	OGUy 的 I/O 属性	可用于/用作
<b>OGU0 输出</b>			
ERU_PDOUT0	ADC0 (REQGT0E) ADC0 (REQGT1E) ADC0 (REQGT2E) ADC1 (REQGT0E) ADC1 (REQGT1E) ADC1 (REQGT2E)	O	序列检测输出
ERU_GOUT0	未连接	O	门控序列检测输出
ERU_GOUT0	未连接	O	门控序列检测输出
ERU_TOUT0	未连接	O	触发输出
ERU_IOUT0	ITC (SCU_ERU_0IC)	O	中断输出

**OGU1 输出**

**系统控制单元 (SCU)**

输出	来自/送至模块	OGUy 的 I/O 属性	可用于/用作
ERU_PDOUT1	ADC0 (REQGT0F) ADC0 (REQGT1F) ADC0 (REQGT2F) ADC1 (REQGT0F) ADC1 (REQGT1F) ADC1 (REQGT2F)	O	序列检测输出
ERU_GOUT1	未连接	O	门控序列检测输出
ERU_TOUT1	ADC0 (REQTR0B) ADC0 (REQTR1B) ADC0 (REQTR2B) ADC1 (REQTR0B) ADC1 (REQTR1B) ADC1 (REQTR2B)	O	触发输出
ERU_IOUT1	ITC (SCU_ERU_1IC)	O	中断输出

**OGU2 输出**

ERU_PDOUT2	CCU60 (CTRAPD)	O	序列检测输出
ERU_GOUT2	未连接	O	门控序列检测输出
ERU_TOUT2	未连接	O	触发输出
ERU_IOUT2	ITC (SCU_ERU_2IC)	O	中断输出

**OGU3 输出**

ERU_PDOUT3	CCU63 (CTRAPD)	O	序列检测输出
ERU_GOUT3	未连接	O	门控序列检测输出
ERU_TOUT3	未连接	O	触发输出
ERU_IOUT3	ITC (SCU_ERU_3IC)	O	中断输出

## 8.18 SCU 寄存器地址

SCU 寄存器位于 XE16xyM 的(E)SFR 区。因此，其绝对地址等于偏移地址。

**表 8-22 寄存器地址空间**

模块	起始地址	结束地址	注
SCU	00 0000 <sub>H</sub>	00 FFFE <sub>H</sub>	

### SCU 寄存器概览

**表 8-23 SCU 寄存器归纳**

寄存器缩写名	寄存器完整名	偏移地址	保护 <sup>1)</sup>	复位 <sup>2)</sup>
<b>WUOSCCON</b>	唤醒 OSC 控制寄存器	F1AE <sub>H</sub>	Sec	上电复位
<b>HPOSCCON</b>	高精度振荡器配置寄存器	F1B4 <sub>H</sub>	Sec	上电复位
<b>PLLOSCCON</b>	PLL 控制寄存器	F1B6 <sub>H</sub>	Sec	上电复位
<b>PLLSTAT</b>	PLL 状态寄存器	F0BC <sub>H</sub>	-	上电复位
<b>STATCLR1</b>	PLL 状态清除 1 寄存器	F0E2 <sub>H</sub>	Sec	上电复位
<b>PLLCON0</b>	PLL 配置 0 寄存器	F1B8 <sub>H</sub>	Sec	上电复位
<b>PLLCON1</b>	PLL 配置 1 寄存器	F1BA <sub>H</sub>	Sec	上电复位
<b>PLLCON2</b>	PLL 配置 2 寄存器	F1BC <sub>H</sub>	Sec	上电复位
<b>PLLCON3</b>	PLL 配置 3 寄存器	F1BE <sub>H</sub>	Sec	上电复位
<b>SYSCON0</b>	系统配置 0 寄存器	FF4A <sub>H</sub>	Sec	上电复位
<b>STATCLR0</b>	状态清除 0 寄存器	F0E0 <sub>H</sub>	Sec	上电复位
<b>RTCCLKCON</b>	RTC 时钟控制寄存器	FF4E <sub>H</sub>	Sec	上电复位
<b>EXTCON</b>	外部时钟控制寄存器	FF5E <sub>H</sub>	Sec	上电复位
<b>STMREL</b>	STM 重载寄存器	F1A8 <sub>H</sub>	Sec	上电复位
<b>STMCON</b>	STM 控制寄存器	F1AA <sub>H</sub>	Sec	上电复位

**系统控制单元 (SCU)**

寄存器缩写名	寄存器完整名	偏移地址	保护 <sup>1)</sup>	复位 <sup>2)</sup>
<b>WUTREL</b>	唤醒定时器重载寄存器	F0B0 <sub>H</sub>	Sec	上电复位
<b>WUCR</b>	唤醒控制寄存器	F1B0 <sub>H</sub>	Sec	上电复位
<b>RSTSTAT0</b>	复位状态 0 寄存器	F0B2 <sub>H</sub>	-	上电复位
<b>RSTSTAT1</b>	复位状态 1 寄存器	F0B4 <sub>H</sub>	-	上电复位
<b>RSTSTAT2</b>	复位状态 2 寄存器	F0B6 <sub>H</sub>	-	上电复位
<b>RSTCON0</b>	复位配置 0 寄存器	F0B8 <sub>H</sub>	Sec	上电复位
<b>RSTCON1</b>	复位配置 1 寄存器	F0BA <sub>H</sub>	Sec	上电复位
<b>RSTCNTCON</b>	复位计数器配置寄存器	F1B2 <sub>H</sub>	Sec	上电复位
<b>SWRSTCON</b>	SW 复位控制寄存器	F0AE <sub>H</sub>	Sec	上电复位
<b>ESREXCON1</b>	ESR1 外部控制寄存器	FF32 <sub>H</sub>	Sec	上电复位
<b>ESREXCON2</b>	ESR2 外部控制寄存器	FF34 <sub>H</sub>	Sec	上电复位
<b>ESREXSTAT1</b>	ESR1 外部状态寄存器	FF36 <sub>H</sub>	-	上电复位
<b>ESREXSTAT2</b>	ESR2 外部状态寄存器	FF38 <sub>H</sub>	-	上电复位
<b>CLRESREXSTAT1</b>	清除 ESR1 外部状态寄存器	FF3A <sub>H</sub>	Sec	上电复位
<b>CLRESREXSTAT2</b>	清除 ESR2 外部状态寄存器	FF3C <sub>H</sub>	Sec	上电复位
<b>ESRCFG0</b>	ESR0 配置寄存器	F100 <sub>H</sub>	Sec	上电复位
<b>ESRCFG1</b>	ESR1 配置寄存器	F102 <sub>H</sub>	Sec	上电复位
<b>ESRCFG2</b>	ESR2 配置寄存器	F104 <sub>H</sub>	Sec	上电复位
<b>ESRDAT</b>	ESR 数据寄存器	F106 <sub>H</sub>	Sec	上电复位
<b>SWDCON0</b>	SWD 控制 0 寄存器	F080 <sub>H</sub>	Sec	上电复位
<b>SWDCON1</b>	SWD 控制 1 寄存器	F082 <sub>H</sub>	Sec	上电复位
<b>PVC1CON0</b>	PVC_1 控制 0 寄存器	F014 <sub>H</sub>	Sec	上电复位
<b>PVCMCON0</b>	PVC_M 控制 0 寄存器	F1E4 <sub>H</sub>	Sec	上电复位
<b>EVR1CON0</b>	EVR_1 控制 0 寄存器	F088 <sub>H</sub>	Sec	上电复位

**系统控制单元 (SCU)**

寄存器缩写名	寄存器完整名	偏移地址	保护 <sup>1)</sup>	复位 <sup>2)</sup>
<b>EVR1SET15VHP</b>	1.5V HP EVR_1 设置寄存器	F09E <sub>H</sub>	Sec	上电复位
<b>EVRMCON0</b>	EVR_M 控制 0 寄存器	F084 <sub>H</sub>	Sec	上电复位
<b>EVRMCON1</b>	EVR_M 控制 1 寄存器	F086 <sub>H</sub>	Sec	上电复位
<b>EVRMSET15VHP</b>	1.5V HP EVR_M 设置寄存器	F096 <sub>H</sub>	Sec	上电复位
<b>GSCSWREQ</b>	GSC SW 请求寄存器	FF14 <sub>H</sub>	Sec	应用复位
<b>GSCEN</b>	GSC 使能寄存器	FF16 <sub>H</sub>	Sec	应用复位
<b>GSCSTAT</b>	GSC 状态寄存器	FF18 <sub>H</sub>	-	应用复位
<b>GSCPERSTATEN</b>	GSC 外设状态使能寄存器	FF04 <sub>H</sub>	Sec	应用复位
<b>GSCPERSTAT</b>	GSC 外设状态寄存器	FF1A <sub>H</sub>	-	应用复位
<b>STSTAT</b>	启动状态寄存器	F1E0 <sub>H</sub>	-	应用复位
<b>EXISEL</b>	外部中断输入选择寄存器	F1A0 <sub>H</sub>	Sec	应用复位
<b>EXICON0</b>	外部中断输入触发控制 0 寄存器	F030 <sub>H</sub>	Sec	应用复位
<b>EXICON1</b>	外部中断输入触发控制 1 寄存器	F032 <sub>H</sub>	Sec	应用复位
<b>EXICON2</b>	外部中断输入触发控制 2 寄存器	F034 <sub>H</sub>	Sec	应用复位
<b>EXICON3</b>	外部中断输入触发控制 3 寄存器	F036 <sub>H</sub>	Sec	应用复位
<b>EXOCON0</b>	外部输出触发控制 0 寄存器	FE30 <sub>H</sub>	Sec	应用复位
<b>EXOCON1</b>	外部输出触发控制 1 寄存器	FE32 <sub>H</sub>	Sec	应用复位
<b>EXOCON2</b>	外部输出触发控制 2 寄存器	FE34 <sub>H</sub>	Sec	应用复位
<b>EXOCON3</b>	外部输出触发控制 3 寄存器	FE36 <sub>H</sub>	Sec	应用复位
<b>INTSTAT</b>	中断状态寄存器	FF00 <sub>H</sub>	-	应用复位
<b>INTCLR</b>	中断清除寄存器	FE82 <sub>H</sub>	Sec	应用复位
<b>INTSET</b>	中断置位寄存器	FE80 <sub>H</sub>	Sec	应用复位

**系统控制单元 (SCU)**

寄存器缩写名	寄存器完整名	偏移地址	保护 <sup>1)</sup>	复位 <sup>2)</sup>
<b>INTDIS</b>	中断禁用寄存器	FE84 <sub>H</sub>	Sec	应用复位
<b>INTNP0</b>	中断节点指针 0 寄存器	FE86 <sub>H</sub>	Sec	应用复位
<b>INTNP1</b>	中断节点指针 1 寄存器	FE88 <sub>H</sub>	Sec	应用复位
<b>DMPMIT</b>	DMP_M 中断和强制中断触发寄存器	FE96 <sub>H</sub>	-	上电复位
<b>DMPMITCLR</b>	DMP_M 中断和强制中断清除寄存器	FE98 <sub>H</sub>	Sec	上电复位
<b>TCCR</b>	温度补偿控制寄存器	F1AC <sub>H</sub>	Sec	应用复位
<b>TCLR</b>	温度补偿等级寄存器	F0AC <sub>H</sub>	Sec	应用复位
<b>WDTREL</b>	WDT 重载寄存器	F0C8 <sub>H</sub>	Sec	应用复位
<b>WDTCS</b>	WDT 控制和状态寄存器	F0C6 <sub>H</sub>	Sec	应用复位
<b>WDTTIM</b>	WDT 定时器寄存器	F0CA <sub>H</sub>	Sec	应用复位
<b>TRAPSTAT</b>	强制中断状态寄存器	FF02 <sub>H</sub>	-	上电复位
<b>TRAPCLR</b>	强制中断清除寄存器	FE8E <sub>H</sub>	Sec	上电复位
<b>TRAPSET</b>	强制中断置位寄存器	FE8C <sub>H</sub>	Sec	上电复位
<b>TRAPDIS</b>	强制中断禁用寄存器	FE90 <sub>H</sub>	Sec	上电复位
<b>TRAPPNP</b>	强制中断节点指针寄存器	FE92 <sub>H</sub>	Sec	上电复位
<b>TRAPPNP1</b>	强制中断节点指针 1 寄存器	FE94 <sub>H</sub>	Sec	上电复位
<b>MCHKCON</b>	存储器检查控制寄存器	F0DC <sub>H</sub>	Sec	上电复位
<b>PEEN</b>	奇偶校验错误使能寄存器	F0C4 <sub>H</sub>	Sec	上电复位
<b>PECON</b>	奇偶校验错误控制寄存器	F0DA <sub>H</sub>	Sec	上电复位
<b>PMPTR</b>	奇偶校验存储器测试序列寄存器	F0E4 <sub>H</sub>	Sec	应用复位
<b>PMTSR</b>	奇偶校验存储器测试选择寄存器	F0E6 <sub>H</sub>	Sec	应用复位
<b>ECCCON</b>	ECC 控制寄存器	F0A8 <sub>H</sub>	Sec	应用复位

**系统控制单元 (SCU)**

寄存器缩写名	寄存器完整名	偏移地址	保护 <sup>1)</sup>	复位 <sup>2)</sup>
<b>ECCSTAT</b>	ECC 状态寄存器	F0AA <sub>H</sub>	-	应用复位
<b>ECCCLRSTAT</b>	ECC 清除状态寄存器	F0DE <sub>H</sub>	Sec	应用复位
<b>SLC</b>	安全等级命令寄存器	F0C0 <sub>H</sub>	-	应用复位
<b>SLS</b>	安全等级状态寄存器	F0C2 <sub>H</sub>	-	应用复位
<b>SYSCON1</b>	系统控制 1 寄存器	FF4C <sub>H</sub>	Sec	应用复位
<b>IDMANUF</b>	制造商 ID 寄存器	F07E <sub>H</sub>	-	上电复位
<b>IDCHIP</b>	芯片 ID 寄存器	F07C <sub>H</sub>	-	上电复位
<b>IDMEM</b>	程序存储器 ID 寄存器	F07A <sub>H</sub>	-	上电复位
<b>IDPROG</b>	编程电压 ID 寄存器	F078 <sub>H</sub>	-	上电复位
<b>IDDMPM</b>	DMP_M ID 寄存器	FFE2 <sub>H</sub>	-	上电复位
<b>IDDMP1</b>	DMP_1 ID 寄存器	FFE4 <sub>H</sub>	-	上电复位
<b>MKMEM0</b>	标记存储器 0 寄存器	FED0 <sub>H</sub>	Sec	上电复位
<b>MKMEM1</b>	标记存储器 1 寄存器	FED2 <sub>H</sub>	Sec	上电复位

1) 寄存器写保护机制: "Sec" = 寄存器安全机制; "-" = 始终可访问 (无保护), 否则不可访问。

2) 复位类型的定义见 [章节 8.4.1.2](#)。

## 9 并行端口

XE16xyM 提供一组可由软件或片上外设单元控制的通用输入/输出端口（GPIO）。

**表 9-1 XE16xyM 的端口**

端口	宽度	I/O	连接的模块
P0	8	I/O	EBC (A7...A0)、CCU6、USIC、CAN
P1	8	I/O	EBC (A15...A8)、CCU6、USIC
P2	14	I/O	EBC (READY、 $\overline{\text{BHE}}$ 、A23...A16、AD15...AD13、D15...D13)、CAN、CC2、GPT12E、USIC、DAP/JTAG
P3	8	I/O	EBC 仲裁 ( $\overline{\text{BREQ}}$ 、 $\overline{\text{HLDA}}$ 、 $\overline{\text{HOLD}}$ )、CAN、USIC
P4	8	I/O	EBC ( $\overline{\text{CS4}}$ ... $\overline{\text{CS0}}$ )、CC2、CAN、GPT12E、USIC
P5	16	I	模拟输入、CCU6、DAP/JTAG、GPT12E、CAN
P6	4	I/O	ADC、CAN、GPT12E
P7	5	I/O	CAN、GPT12E、SCU、DAP/JTAG、CCU6、ADC、USIC
P8	7	I/O	CCU6、DAP/JTAG、USIC
P9	8	I/O	CCU6、DAP/JTAG、CAN
P10	16	I/O	EBC (ALE、 $\overline{\text{RD}}$ 、 $\overline{\text{WR}}$ 、AD12...AD0、D12...D0)、CCU6、USIC、DAP/JTAG、CAN
P11	6	I/O	CCU6、USIC、CAN
P15	8	I	模拟输入、GPT12E

注：可用的端口及端口引脚和选用的器件类型有关。本章描述最多可用的端口组。

### 9.1 一般描述

本章描述端口引脚的数字控制电路的架构。

#### 9.1.1 基本端口操作

共有三种数字控制电路：针对数字 GPIO 的带有/不带硬件覆盖功能的数字控制电路和针对模拟输入的数字控制电路。每个端口引脚包含一种数字控制电路。



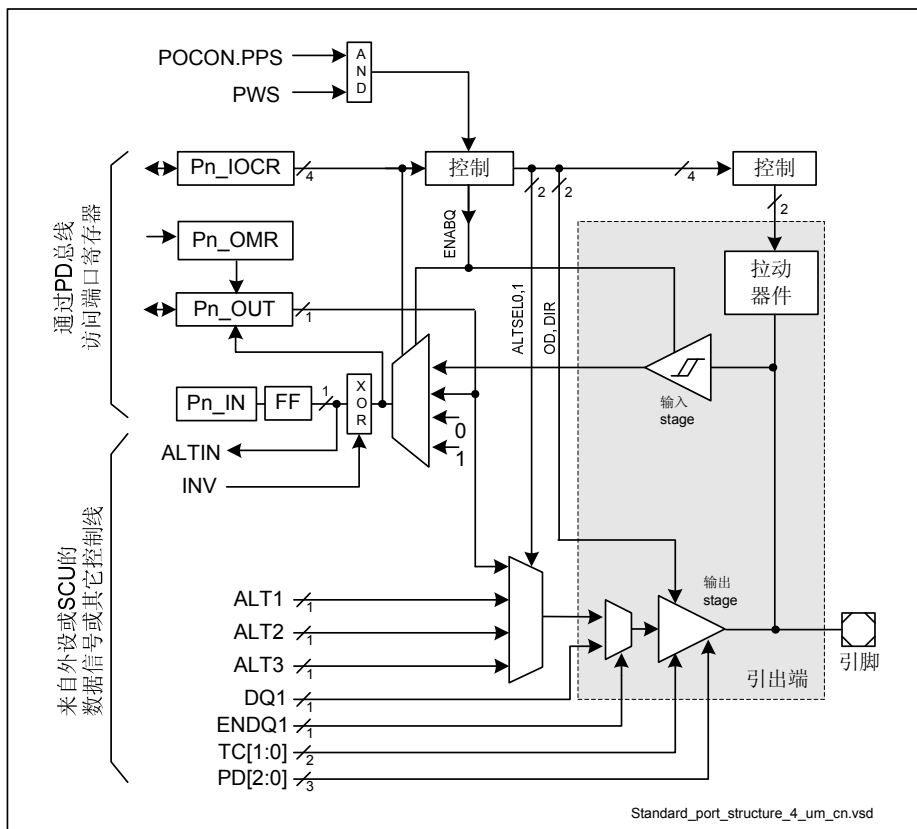


图 9-1 不带硬件覆盖功能的端口结构

注: INV 信号来自 Pn\_IOCR.PC[3:2]。

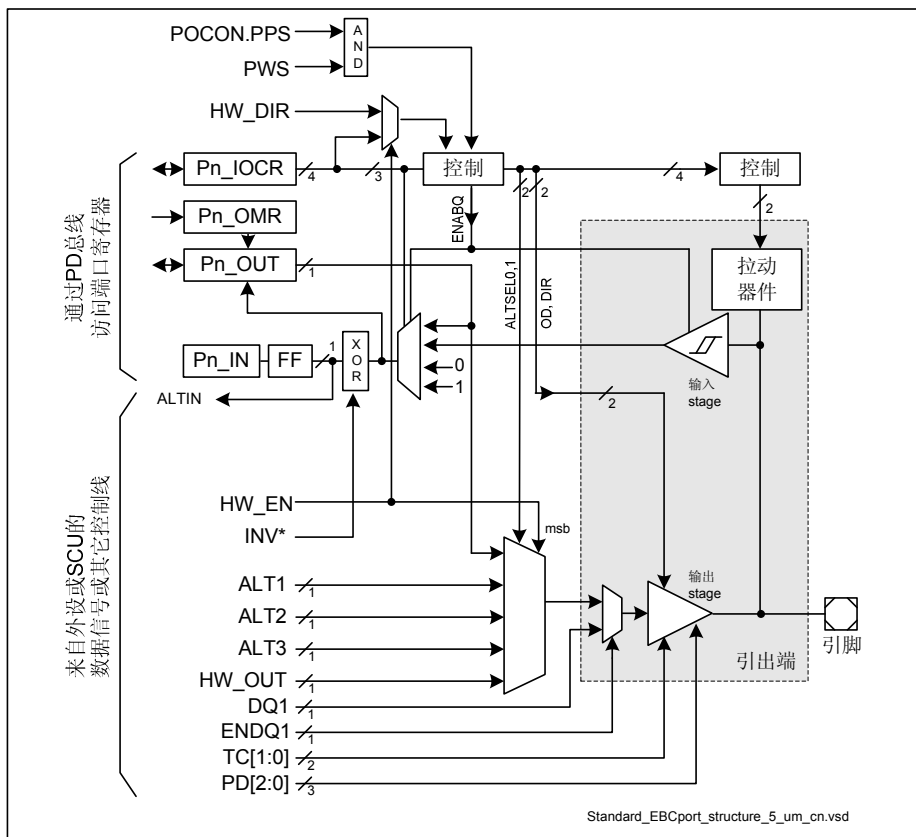


图 9-2 带有硬件覆盖功能的端口结构

注：若  $HW\_EN$  被激活， $INV^*$  信号始终为 0。

注：若  $HW\_EN$  被禁用，相应端口和所有其它端口一样进入省电模式。若  $HW\_EN$  有效，用户应设置  $POCON.PPSx = 0$ 。

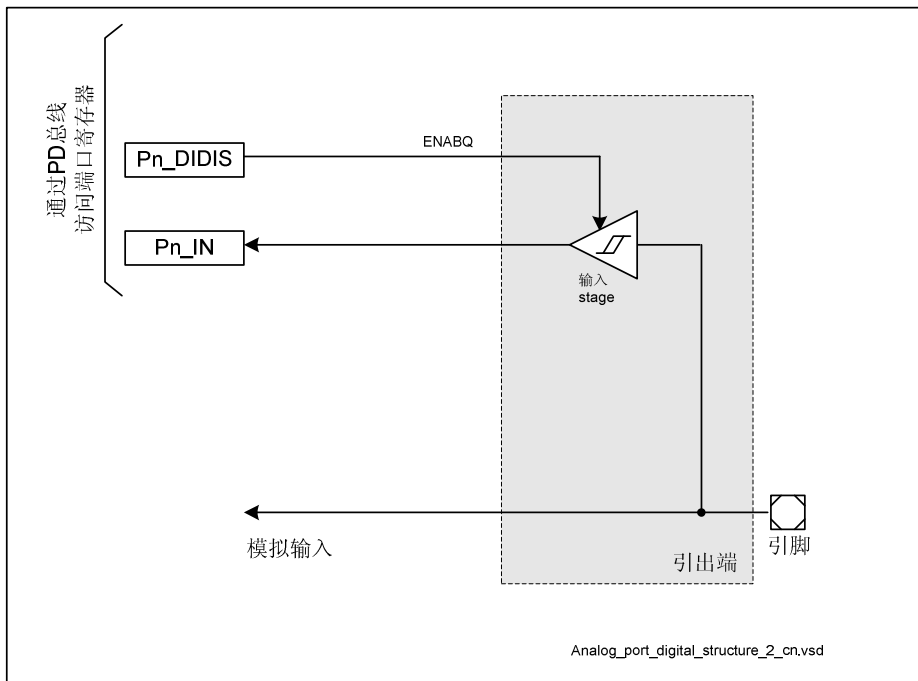


图 9-3 P5 口和 P15 口的结构

注：每个模拟输入口始终并行连接一个标准数字输入。

### 9.1.2 输入级控制

输入级由一个施密特触发器（可由软件使能或禁用）和一个输入复用器（缺省选择输入施密特触发器的输出）组成。

若输入驱动器被禁用，输出逻辑高电平。在复位期间和复位之后，所有输入级缺省被使能。

### 9.1.3 输出驱动控制

输出级由输出驱动器、输出复用器和对它们进行控制的寄存器位域组成。

#### 9.1.3.1 正常工作模式下的操作

每个输出驱动器可配置为推挽或漏极开路模式，也可配置为失效状态（三态）。输出驱动器的前级输出复用器用于选择信号源，信号源可来自 Pn\_OUT 寄存器，也可来自

外设单元（从最多 3 路信号线中选择 1 路），见图 9-1。通过寄存器 Pn\_IOCRR 进行信号选择。Pn\_OUT.Px 可由软件置位或清零，若输出复用器选择 Pn\_OUT.Px，则 Pn\_OUT.Px 驱动端口引脚。

具有硬件覆盖功能的输出驱动器可选择一路（来自外设单元的）附加输出信号。若硬件覆盖功能被激活，该信号的优先级高于所有其它输出信号，不可由端口取消选定。在这种情况下，外设控制引脚的方向。

### 9.1.3.2 省电模式下的操作

在省电模式下（内核和 IO 供电电压可用），引脚的行为取决于 POCOnx.PPSx 位的设置。通常，一个端口内的四个引脚构成一组，可配置为响应或忽略省电模式请求。若某组引脚被配置为响应省电模式请求，组内的每个引脚根据各自的配置（参见表 9-4）对省电请求予以响应。

### 9.1.3.3 复位操作

在内部应用复位过程中，GPIO 的所有输出级进入三态模式，无任何上拉或下拉器件。

应用复位不改变 GPIO 的配置，但内部外设的复位可改变驱动到输出上的数据值。用户必须谨慎，避免那些不期望的信号跳变和输出值对所连器件造成损坏。

### 9.1.3.4 电源故障情况下的操作

若内核电源出现故障的同时引出端电源保持稳定，输出级进入三态模式。

## 9.2 端口寄存器描述

### 9.2.1 引出端驱动控制

XE16xyM 采用的引出端结构可用于选择输出驱动能力和摆率。这两个参数和输出端口的功能（如漏极开路、推/挽或单向输入）无关。

为了尽量降低 EMI，可通过位域 PDMx 修改端口驱动能力以满足不同应用的需求（四个引脚一组进行设置）。

**端口输出控制寄存器 POCOn** 中存放端口控制位。4 位控制域用于配置驱动能力和边沿形状。每个字端口需要 4 个控制半字节；每个字节端口需要 2 个控制半字节；每个控制半字节控制 4 个引脚。

*注：XE16xyM 中的 P2\_POCON 寄存器有一个例外情况，P2.8 引脚上除标准输出驱动器之外，还并行连接一个附加强输出驱动器。详见相关端口一节的描述。*

Px\_POCON (x=0-4)

端口 x 输出控制寄存器

XSFR (E8A0<sub>H</sub>+2\*x)

复位值: 0000<sub>H</sub>

Px\_POCON (x=6-11)

端口 x 输出控制寄存器

XSFR (E8A0<sub>H</sub>+2\*x)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PPS3	PDM3	PPS2	PDM2	PPS1	PDM1	PPS0	PDM0								
rw	rw	rw	rw	rw	rw	rw	rw								

符号	位序号	类型	功能描述
PDM0, PDM1, PDM2, PDM3	[2:0], [6:4], [10:8], [14:12]	rw	<p>端口驱动模式 x</p> <p>编码 驱动能力 <sup>1)</sup> 边沿形状 <sup>2)</sup></p> <p>000<sub>B</sub> 强驱动 陡边沿模式</p> <p>001<sub>B</sub> 强驱动 中等坡度边沿模式</p> <p>010<sub>B</sub> 强驱动 平缓边沿模式</p> <p>011<sub>B</sub> 弱驱动</p> <p>100<sub>B</sub> 中驱动</p> <p>101<sub>B</sub> 中驱动</p> <p>110<sub>B</sub> 中驱动</p> <p>111<sub>B</sub> 弱驱动</p>
PPS0, PPS1, PPS2, PPS3	3, 7, 11, 15	rw	<p>引脚省电</p> <p>0<sub>B</sub> 以正常工作模式操作引脚，不考虑省电管理。</p> <p>1<sub>B</sub> 以省电模式操作引脚（描述见<a href="#">表 9-4</a>）。</p>

1) 定义各驱动器可送至外部电路的电流。

2) 定义转换为新输出电平的切换特性（switching characteristics）。发生电平跳变时，这也会影响通过驱动器的峰值电流。

## POCON 寄存器和引脚及端口的映射关系

下表罗列出所定义的 POCON 寄存器以及控制位域和端口引脚的分配。

**表 9-2 端口输出控制寄存器的分配**

控制寄存器	受控引脚（由 Px_POCON.[y:z]控制） <sup>1)</sup>				端口宽度
	[15:12]	[11:8]	[7:4]	[3:0]	
P0_POCON	---	---	P0.[7:4]	P0.[3:0]	8
P1_POCON	---	---	P1.[7:4]	P1.[3:0]	8
P2_POCON	P2.8 上的 CLOCKOUT 驱动 <sup>2)</sup>	P2.[11:8] + P2.[13:12] <sup>3)</sup>	P2.[7:4]	P2.[3:0]	14
P3_POCON	---	---	P3.[7:4]	P3.[3:0]	8
P4_POCON	---	---	P4.[7:4]	P4.[3:0]	8
P6_POCON	---	---	---	P6.[3:0]	4
P7_POCON	---	---	P7.4	P7.[3:0]	5
P8_POCON	---	---	P8.[6:4]	P8.[3:0]	7
P9_POCON	---	---	P9.[7:4]	P9.[3:0]	8
P10_POCON	P10.[15:12]	P10.[11:8]	P10.[7:4]	P10.[3:0]	16
P11_POCON	---	---	P11.[5:4]	P11.[3:0]	6

1) X 代表端口编号，[y:z]代表位域段。

2) 对附加驱动器的控制描述见 [章节 9.3.3](#)。

3) P2.[13:12]的输出控制和标准定义不同，具体描述见 [章节 9.3.3](#)。

*注：将功能信号分配给端口引脚时，需考虑到驱动能力的选择是针对引脚组的。在同一个 POCON 控制组内对具有相似要求的引脚进行功能配置。*

### 9.2.2 端口输出寄存器

某引脚用作 GPIO 输出时，由端口输出寄存器决定该引脚的电平。

#### **Pn\_OUT (n=0-4)**

端口 n 输出寄存器                      **SFR (FFA2<sub>H</sub>+2\*n)**                      复位值: 0000<sub>H</sub>

#### **Pn\_OUT (n=6-11)**

端口 n 输出寄存器                      **SFR (FFA2<sub>H</sub>+2\*n)**                      复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>P15</b>	<b>P14</b>	<b>P13</b>	<b>P12</b>	<b>P11</b>	<b>P10</b>	<b>P9</b>	<b>P8</b>	<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位序号	类型	功能描述
<b>Px</b> (x=0-15)	x	rwh	<b>端口输出位 x</b> 若端口 Pn 的引脚 x 用作 GPIO 输出，该位定义该输出引脚的电平值。 0 <sub>B</sub> Pn.x 的输出电平为 0。 1 <sub>B</sub> Pn.x 的输出电平为 1。

### 9.2.3 端口输出修改寄存器

端口输出修改寄存器可用于置位、清零或翻转端口 n 输出寄存器中的相应位。

#### **P2\_OMRH**

**P2 口输出修改高位寄存器**                      **XSFR (E9CA<sub>H</sub>)**                      复位值: XXXX<sub>H</sub>

#### **P10\_OMRH**

**P10 口输出修改高位寄存器**                      **XSFR (E9EA<sub>H</sub>)**                      复位值: XXXX<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PC15</b>	<b>PC14</b>	<b>PC13</b>	<b>PC12</b>	<b>PC11</b>	<b>PC10</b>	<b>PC9</b>	<b>PC8</b>	<b>PS15</b>	<b>PS14</b>	<b>PS13</b>	<b>PS12</b>	<b>PS11</b>	<b>PS10</b>	<b>PS9</b>	<b>PS8</b>
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

符号	位序号	类型	功能描述
<b>PSx</b> (x=8-15)	x-8	w	<b>端口置位位 x</b> 置位该位将会置位或翻转端口输出寄存器 Pn_OUT 中的相应位 (见表 9-3)。 读取该位将返回未定义的值。
<b>PCx</b> (x=8-15)	x	w	<b>端口清零位 x</b> 置位该位将会清除或翻转端口输出寄存器 Pn_OUT 中的相应位 (见表 9-3)。 读取该位将返回未定义的值。

**Pn\_OMRL (n=0-4)**

端口 n 输出修改低位寄存器      **XSFR (E9C0<sub>H</sub>+4\*n)**

复位值: XXXX<sub>H</sub>

**Pn\_OMRL (n=6-11)**

端口 n 输出修改低位寄存器      **XSFR (E9C0<sub>H</sub>+4\*n)**

复位值: XXXX<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PC7</b>	<b>PC6</b>	<b>PC5</b>	<b>PC4</b>	<b>PC3</b>	<b>PC2</b>	<b>PC1</b>	<b>PC0</b>	<b>PS7</b>	<b>PS6</b>	<b>PS5</b>	<b>PS4</b>	<b>PS3</b>	<b>PS2</b>	<b>PS1</b>	<b>PS0</b>
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

符号	位序号	类型	功能描述
<b>PSx</b> (x=0-7)	x	w	<b>端口置位位 x</b> 置位该位将会置位或翻转端口输出寄存器 Pn_OUT 中的相应位 (见表 9-3)。 读取该位将返回未定义的值。
<b>PCx</b> (x=0-7)	x+8	w	<b>端口清零位 x</b> 置位该位将会清除或翻转端口输出寄存器 Pn_OUT 中的相应位 (见表 9-3)。 读取该位将返回未定义的值。



## PCx 和 PSx 的功能

**表 9-3 位 PCx 和 PSx 的功能**

PCx	PSx	功能
0 或无写访问	0 或无写访问	位 Pn_OUT.Px 不改变
0 或无写访问	1	位 Pn_OUT.Px 被置位
1	0 或无写访问	位 Pn_OUT.Px 被清零
1	1	位 Pn_OUT.Px 被翻转

### 9.2.4 端口输入寄存器

端口输入寄存器存放当前从输入引脚读取的电平（端口线用作输出时同样如此）。

#### Pn\_IN (n=0-11)

端口 n 输入寄存器

**SFR (FF80<sub>H</sub>+2\*n)**

复位值: 0000<sub>H</sub><sup>1)</sup>

#### P15\_IN

端口 15 输入寄存器

**SFR (FF9E<sub>H</sub>)**

复位值: 0000<sub>H</sub><sup>1)</sup>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>P15</b>	<b>P14</b>	<b>P13</b>	<b>P12</b>	<b>P11</b>	<b>P10</b>	<b>P9</b>	<b>P8</b>	<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

1) 未实现的 Px 位读取始终返回 0。

符号	位序号	类型	功能描述
<b>Px</b> <b>(x=0-15)</b>	x	rh	<b>端口输入位 x</b> 该位指示端口 Pn 的输入引脚 x 的电平值。 0 <sub>B</sub> Pn.x 的输入电平为 0。 1 <sub>B</sub> Pn.x 的输入电平为 1。

### 9.2.5 端口输入/输出控制寄存器

端口输入/输出控制寄存器用于选择引脚的数字输入输出驱动特性，如选择输入引脚的上拉/下拉器件、端口方向（输入/输出）、输出引脚的漏极开路或复用输出功能等。

PC 编码见 [表 9-4](#)。

并非所有的输入/输出控制寄存器功能都需要实现，它取决于端口功能。每个端口引脚的控制位域分别位于不同寄存器中，从而可通过字写访问单独配置某个引脚的功能而不会访问到同一寄存器中的其它 PCx 位域。

#### P0\_IOCRx (x=00-07)

**P0 口输入/输出控制寄存器 x**    **XSFR (E800<sub>H</sub>+2\*x)**    **复位值: 0000<sub>H</sub>**

#### P1\_IOCRx (x=00-07)

**P1 口输入/输出控制寄存器 x**    **XSFR (E820<sub>H</sub>+2\*x)**    **复位值: 0000<sub>H</sub>**

#### P2\_IOCRx (x=00-13)

**P2 口输入/输出控制寄存器 x**    **XSFR (E840<sub>H</sub>+2\*x)**    **复位值: 0000<sub>H</sub>**

#### P3\_IOCRx (x=00-07)

**P3 口输入/输出控制寄存器 x**    **XSFR (E860<sub>H</sub>+2\*x)**    **复位值: 0000<sub>H</sub>**

#### P4\_IOCRx (x=00-07)

**P4 口输入/输出控制寄存器 x**    **XSFR (E880<sub>H</sub>+2\*x)**    **复位值: 0000<sub>H</sub>**

#### P6\_IOCRx (x=00-03)

**P6 口输入/输出控制寄存器 x**    **XSFR (E8C0<sub>H</sub>+2\*x)**    **复位值: 0000<sub>H</sub>**

#### P7\_IOCRx (x=00-04)

**P7 口输入/输出控制寄存器 x**    **XSFR (E8E0<sub>H</sub>+2\*x)**    **复位值: 0000<sub>H</sub>**

#### P8\_IOCRx (x=00-06)

**P8 口输入/输出控制寄存器 x**    **XSFR (E900<sub>H</sub>+2\*x)**    **复位值: 0000<sub>H</sub>**

#### P9\_IOCRx (x=00-07)

**P9 口输入/输出控制寄存器 x**    **XSFR (E920<sub>H</sub>+2\*x)**    **复位值: 0000<sub>H</sub>**

#### P10\_IOCRx (x=00-15)

**P10 口输入/输出控制寄存器 x**    **XSFR (E940<sub>H</sub>+2\*x)**    **复位值: 0000<sub>H</sub>**

#### P11\_IOCRx (x=00-05)

**P11 口输入/输出控制寄存器 x**    **XSFR (E960<sub>H</sub>+2\*x)**    **复位值: 0000<sub>H</sub>**

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

符号	位序号	类型	功能描述
<b>PC</b>	[7:4]	rw	端口输入/输出控制位 见表 9-4
<b>0</b>	[3:0], [15:8]	r	保留

### 位域 PC 的编码

由端口控制寄存器 Pn\_IOCR<sub>x</sub> 定义 GPIO 的端口行为。每个端口引脚对应有一个控制位域 PC，每个端口引脚的 PC 位于不同的控制寄存器中，从而可通过简单的转移操作修改单个引脚的功能（而不会影响其它引脚）。

*注：当引脚方向切换为输出且为测试模式时，必须只能选择推挽输出。*

**表 9-4 PC 编码**

PCx[3:0]	I/O	选择上拉/下拉/输出功能	省电模式下的操作 <sup>1)</sup>
0000 <sub>B</sub>	直接输入	不与上拉/下拉器件连接	输入值 = Pn_OUT; 无上拉/下拉
0001 <sub>B</sub>		与下拉器件连接	输入值 = 0; 下拉
0010 <sub>B</sub>		与上拉器件连接	输入值 = 1; 上拉
0011 <sub>B</sub>		不与上拉/下拉器件连接。该模式下，Pn_OUT 连续采样引出端的输入值。	输入值 = Pn_OUT; Pn_OUT 中的内容为进入省电模式之前的输入值 = 输入值冻结; 无上拉/下拉
0100 <sub>B</sub>	输入反相	不与上拉/下拉器件连接	输入值 = $\overline{\text{Pn\_OUT}}$ ; 无上拉/下拉
0101 <sub>B</sub>		与下拉器件连接	输入值 = 1; 下拉
0110 <sub>B</sub>		与上拉器件连接	输入值 = 0; 上拉
0111 <sub>B</sub>		不与上拉/下拉器件连接。该模式下，Pn_OUT 连续采样引出端的输入值。	输入值 = $\overline{\text{Pn\_OUT}}$ ; Pn_OUT 中的内容为进入省电模式之前的输入值 = 输入值冻结; 无上拉/下拉 <sup>2)</sup>
1000 <sub>B</sub>	输出	通用输出	输出驱动关闭。

PCx[3:0]	I/O	选择上拉/下拉/输出功能	省电模式下的操作 <sup>1)</sup>
1001 <sub>B</sub>	(直接输入) 推挽	输出功能 ALT1	输入施密特触发器关闭。 Pn_OUT 送至内部逻辑； 无上拉/下拉
1010 <sub>B</sub>		输出功能 ALT2	
1011 <sub>B</sub>		输出功能 ALT3	
1100 <sub>B</sub>	输出 (直接输入) 漏极开路	通用输出	
1101 <sub>B</sub>		输出功能 ALT1	
1110 <sub>B</sub>		输出功能 ALT2	
1111 <sub>B</sub>		输出功能 ALT3	

- 1) 在省电模式下，输入施密特触发器始终关闭。将设定的输入值送至内部电路（取代输入引脚上检测到的值）。
- 2) 若 IOCR 设置为“反相输入”，则内部驱动 Pn\_OUT 的反相信号。Pn\_OUT 寄存器本身始终存放引脚真正的、不反相的输入值。见 [图 9-1](#) 和 [图 9-2](#)。

### 9.2.6 端口数字输入禁用寄存器

P5 和 P15 口不仅具有模拟输入功能，还具有数字输入功能。为了避免数字输入的施密特触发器的开关切换，数字输入功能可通过 Px\_DIDIS 寄存器被禁用。

#### P5\_DIDIS

**P5 口数字输入禁用寄存器**

**SFR (FE8A<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

#### P15\_DIDIS

**P15 口数字输入禁用寄存器**

**SFR (FE9E<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

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
rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW

符号	位序号	类型	功能描述
<b>Py</b> (y=0-15)	y	rW	<b>位 y 数字输入控制</b> 0 <sub>B</sub> 使能数字输入级（施密特触发器）。 1 <sub>B</sub> 禁用数字输入级（施密特触发器），引脚用作模拟输入时需要这样设置。

## 9.3 端口描述

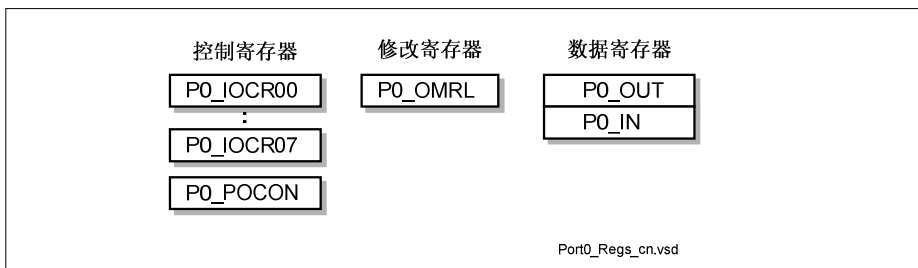
端口寄存器中的各位始终右对齐。例如，包含 8 个引脚的端口只使用寄存器的位 [7:0]，其余位填 0（只读）。

每个端口的引出端驱动模式寄存器可能不同。因此，在各端口的独立章节中分别描述。

### 9.3.1 P0 口

P0 口为 8 位 GPIO 端口。P0 口的寄存器如图 9-4 所示。

该端口的所有引脚可用作 GPIO 从端口输入寄存器读取电平值。



**图 9-4 P0 口寄存器概览**

**表 9-5 P0 口寄存器**

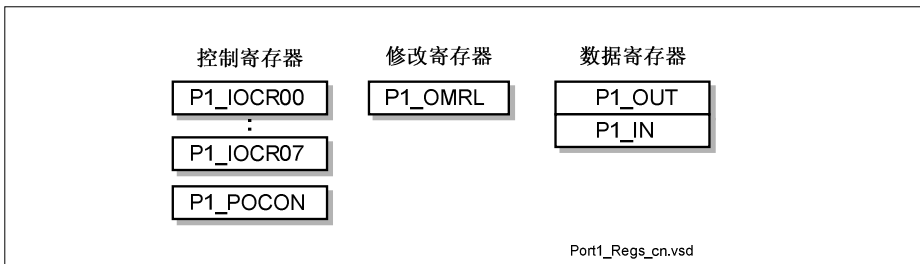
寄存器缩略名	寄存器完整名	偏移地址	复位值
P0_OUT	P0 口输出寄存器	FFA2 <sub>H</sub>	0000 <sub>H</sub>
P0_IN	P0 口输入寄存器	FF80 <sub>H</sub>	0000 <sub>H</sub>
P0_OMRL	P0 口输出修改低位寄存器	E9C0 <sub>H</sub>	XXXX <sub>H</sub>
P0_POCON	P0 口输出控制寄存器	E8A0 <sub>H</sub>	0000 <sub>H</sub>
P0_IOCRO0	P0 口输入/输出控制寄存器 0	E800 <sub>H</sub>	0000 <sub>H</sub>
P0_IOCRO1	P0 口输入/输出控制寄存器 1	E802 <sub>H</sub>	0000 <sub>H</sub>
P0_IOCRO2	P0 口输入/输出控制寄存器 2	E804 <sub>H</sub>	0000 <sub>H</sub>
P0_IOCRO3	P0 口输入/输出控制寄存器 3	E806 <sub>H</sub>	0000 <sub>H</sub>

寄存器缩略名	寄存器完整名	偏移地址	复位值
P0_IOCRR04	P0 口输入/输出控制寄存器 4	E808 <sub>H</sub>	0000 <sub>H</sub>
P0_IOCRR05	P0 口输入/输出控制寄存器 5	E80A <sub>H</sub>	0000 <sub>H</sub>
P0_IOCRR06	P0 口输入/输出控制寄存器 6	E80C <sub>H</sub>	0000 <sub>H</sub>
P0_IOCRR07	P0 口输入/输出控制寄存器 7	E80E <sub>H</sub>	0000 <sub>H</sub>

### 9.3.2 P1 口

P1 口为 8 位 GPIO 端口。P1 口的寄存器如图 9-5 所示。

该端口的所有引脚可用作 GPIO 从端口输入寄存器读取电平值。



**图 9-5 P1 口寄存器概览**

**表 9-6 P1 口寄存器**

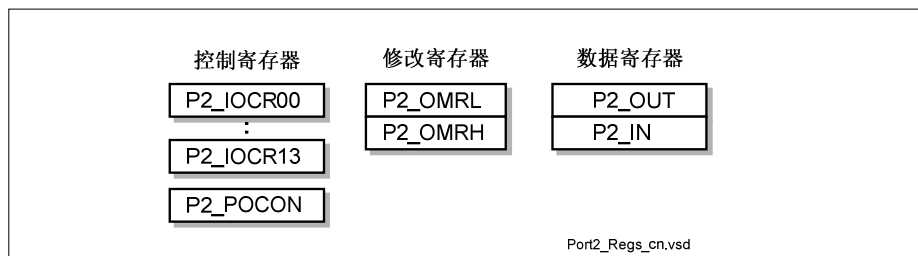
寄存器缩略名	寄存器完整名	偏移地址	复位值
P1_OUT	P1 口输出寄存器	FFA4 <sub>H</sub>	0000 <sub>H</sub>
P1_IN	P1 口输入寄存器	FF82 <sub>H</sub>	0000 <sub>H</sub>
P1_OMRL	P1 口输出修改低位寄存器	E9C4 <sub>H</sub>	XXXX <sub>H</sub>
P1_POCON	P1 口输出控制寄存器	E8A2 <sub>H</sub>	0000 <sub>H</sub>
P1_IOCRR00	P1 口输入/输出控制寄存器 0	E820 <sub>H</sub>	0000 <sub>H</sub>
P1_IOCRR01	P1 口输入/输出控制寄存器 1	E822 <sub>H</sub>	0000 <sub>H</sub>
P1_IOCRR02	P1 口输入/输出控制寄存器 2	E824 <sub>H</sub>	0000 <sub>H</sub>

寄存器缩略名	寄存器完整名	偏移地址	复位值
P1_IOCRO3	P1 口输入/输出控制寄存器 3	E826 <sub>H</sub>	0000 <sub>H</sub>
P1_IOCRO4	P1 口输入/输出控制寄存器 4	E828 <sub>H</sub>	0000 <sub>H</sub>
P1_IOCRO5	P1 口输入/输出控制寄存器 5	E82A <sub>H</sub>	0000 <sub>H</sub>
P1_IOCRO6	P1 口输入/输出控制寄存器 6	E82C <sub>H</sub>	0000 <sub>H</sub>
P1_IOCRO7	P1 口输入/输出控制寄存器 7	E82E <sub>H</sub>	0000 <sub>H</sub>

### 9.3.3 P2 口

P2 口为 14 位 GPIO 端口。P2 口的寄存器如图 9-6 所示。

该端口的所有引脚可用作 GPIO 从端口输入寄存器读取电平值。



**图 9-6 P2 口寄存器概览**

**表 9-7 P2 口寄存器**

寄存器缩略名	寄存器完整名	偏移地址	复位值
P2_OUT	P2 口输出寄存器	FFA6 <sub>H</sub>	0000 <sub>H</sub>
P2_IN	P2 口输入寄存器	FF84 <sub>H</sub>	0000 <sub>H</sub>
P2_OMRL	P2 口输出修改低位寄存器	E9C8 <sub>H</sub>	XXXX <sub>H</sub>
P2_OMRH	P2 口输出修改高位寄存器	E9CA <sub>H</sub>	XXXX <sub>H</sub>
P2_POCON	P2 口输出控制寄存器	E8A4 <sub>H</sub>	0000 <sub>H</sub>
P2_IOCRO0	P2 口输入/输出控制寄存器 0	E840 <sub>H</sub>	0000 <sub>H</sub>

寄存器缩略名	寄存器完整名	偏移地址	复位值
P2_IOCR01	P2 口输入/输出控制寄存器 1	E842 <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR02	P2 口输入/输出控制寄存器 2	E844 <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR03	P2 口输入/输出控制寄存器 3	E846 <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR04	P2 口输入/输出控制寄存器 4	E848 <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR05	P2 口输入/输出控制寄存器 5	E84A <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR06	P2 口输入/输出控制寄存器 6	E84C <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR07	P2 口输入/输出控制寄存器 7	E84E <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR08	P2 口输入/输出控制寄存器 8	E850 <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR09	P2 口输入/输出控制寄存器 9	E852 <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR10	P2 口输入/输出控制寄存器 10	E854 <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR11	P2 口输入/输出控制寄存器 11	E856 <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR12	P2 口输入/输出控制寄存器 12	E858 <sub>H</sub>	0000 <sub>H</sub>
P2_IOCR13	P2 口输入/输出控制寄存器 13	E85A <sub>H</sub>	0000 <sub>H</sub>

### CLKOUT 引出端 P2.8

为了驱动高频时钟信号，P2.8 引脚上除标准输出驱动器之外，还并行连接一个强输出驱动器。当位域 P2\_POCON.PDM3 = xx1<sub>B</sub> 时，使能该强驱动器代替标准驱动器。

强时钟驱动器只在强驱动、陡边沿模式下工作，即它不受标准驱动器的端口驱动模式设置的控制（P2\_POCON.PDM2）。

它无上拉/下拉器件，但可通过寄存器 P2\_IOCR08 切换到输入或输出。

### 引脚 P2.[13:12]的输出控制

由于位域 P2\_POCON.PDM3 控制 P2.8 的强时钟驱动器，因此引脚 P2.[13:12]（连同引脚 P2.[11:8]）的驱动模式由位域 P2\_POCON.PDM2 控制。

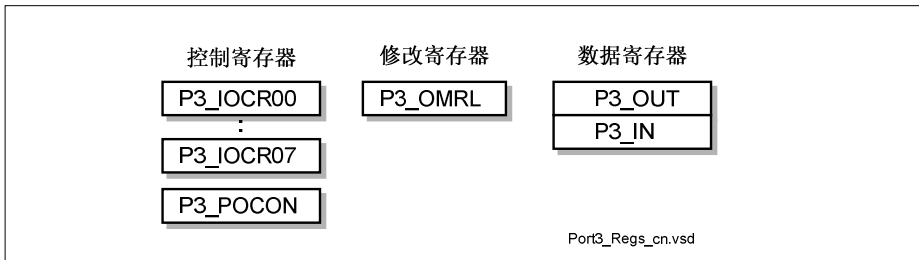
引脚 P2.[13:12]的省电模式操作由位 P2\_POCON.PPS3 控制。



### 9.3.4 P3 口

P3 口为 8 位 GPIO 端口。P3 口的寄存器如图 9-7 所示。

该端口的所有引脚可用作 GPIO 从端口输入寄存器读取电平值。



**图 9-7 P3 口寄存器概览**

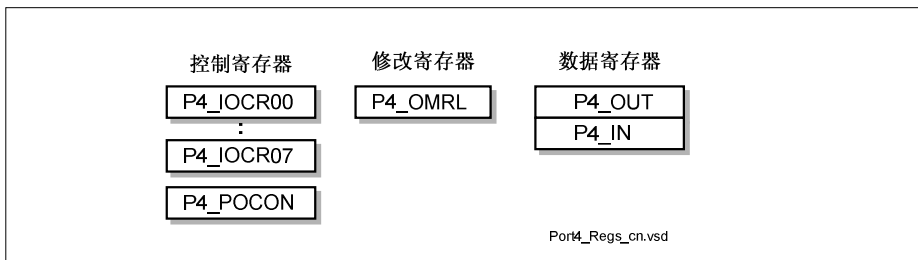
**表 9-8 P3 口寄存器**

寄存器缩略名	寄存器完整名	偏移地址	复位值
P3_OUT	P3 口输出寄存器	FFA8 <sub>H</sub>	0000 <sub>H</sub>
P3_IN	P3 口输入寄存器	FF86 <sub>H</sub>	0000 <sub>H</sub>
P3_OMRL	P3 口输出修改低位寄存器	E9CC <sub>H</sub>	XXXX <sub>H</sub>
P3_POCON	P3 口输出控制寄存器	E8A6 <sub>H</sub>	0000 <sub>H</sub>
P3_IOCRO0	P3 口输入/输出控制寄存器 0	E860 <sub>H</sub>	0000 <sub>H</sub>
P3_IOCRO1	P3 口输入/输出控制寄存器 1	E862 <sub>H</sub>	0000 <sub>H</sub>
P3_IOCRO2	P3 口输入/输出控制寄存器 2	E864 <sub>H</sub>	0000 <sub>H</sub>
P3_IOCRO3	P3 口输入/输出控制寄存器 3	E866 <sub>H</sub>	0000 <sub>H</sub>
P3_IOCRO4	P3 口输入/输出控制寄存器 4	E868 <sub>H</sub>	0000 <sub>H</sub>
P3_IOCRO5	P3 口输入/输出控制寄存器 5	E86A <sub>H</sub>	0000 <sub>H</sub>
P3_IOCRO6	P3 口输入/输出控制寄存器 6	E86C <sub>H</sub>	0000 <sub>H</sub>
P3_IOCRO7	P3 口输入/输出控制寄存器 7	E86E <sub>H</sub>	0000 <sub>H</sub>

### 9.3.5 P4 口

P4 口为 8 位 GPIO 端口。P4 口的寄存器如图 9-8 所示。

该端口的所有引脚可用作 GPIO 从端口输入寄存器读取电平值。



**图 9-8 P4 口寄存器概览**

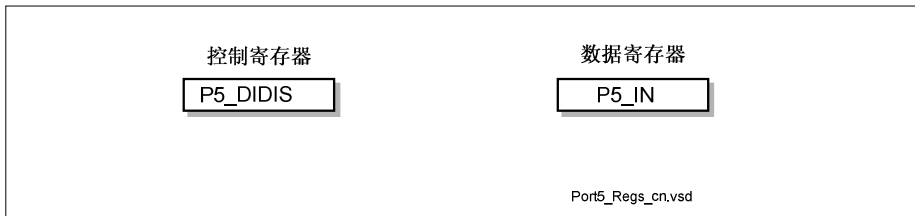
**表 9-9 P4 口寄存器**

寄存器缩写名	寄存器完整名	偏移地址	复位值
P4_OUT	P4 口输出寄存器	FFAA <sub>H</sub>	0000 <sub>H</sub>
P4_IN	P4 口输入寄存器	FF88 <sub>H</sub>	0000 <sub>H</sub>
P4_OMRL	P4 口输出修改低位寄存器	E9D0 <sub>H</sub>	XXXX <sub>H</sub>
P4_POCON	P4 口输出控制寄存器	E8A8 <sub>H</sub>	0000 <sub>H</sub>
P4_IOCR00	P4 口输入/输出控制寄存器 0	E880 <sub>H</sub>	0000 <sub>H</sub>
P4_IOCR01	P4 口输入/输出控制寄存器 1	E882 <sub>H</sub>	0000 <sub>H</sub>
P4_IOCR02	P4 口输入/输出控制寄存器 2	E884 <sub>H</sub>	0000 <sub>H</sub>
P4_IOCR03	P4 口输入/输出控制寄存器 3	E886 <sub>H</sub>	0000 <sub>H</sub>
P4_IOCR04	P4 口输入/输出控制寄存器 4	E888 <sub>H</sub>	0000 <sub>H</sub>
P4_IOCR05	P4 口输入/输出控制寄存器 5	E88A <sub>H</sub>	0000 <sub>H</sub>
P4_IOCR06	P4 口输入/输出控制寄存器 6	E88C <sub>H</sub>	0000 <sub>H</sub>
P4_IOCR07	P4 口输入/输出控制寄存器 7	E88E <sub>H</sub>	0000 <sub>H</sub>

### 9.3.6 P5 口

P5 口为 16 位模拟或数字输入端口。

P5 口用作模拟输入时，必须禁用施密特触发器，通过置位寄存器 P5\_DIDIS 中的相应位来实现。



**图 9-9 P5 口寄存器概览**

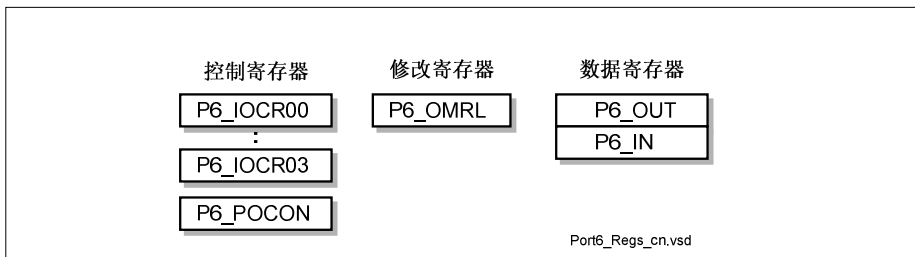
**表 9-10 P5 口寄存器**

寄存器缩略名	寄存器完整名	偏移地址	复位值
P5_IN	P5 口输入寄存器	FF8A <sub>H</sub>	0000 <sub>H</sub>
P5_DIDIS	P5 口数字输入禁用寄存器	FE8A <sub>H</sub>	0000 <sub>H</sub>

### 9.3.7 P6 口

P6 口为 4 位 GPIO 端口。P6 口的寄存器如图 9-10 所示。

该端口的所有引脚可用作 GPIO 从端口输入寄存器读取电平值。



**图 9-10 P6 口寄存器概览**

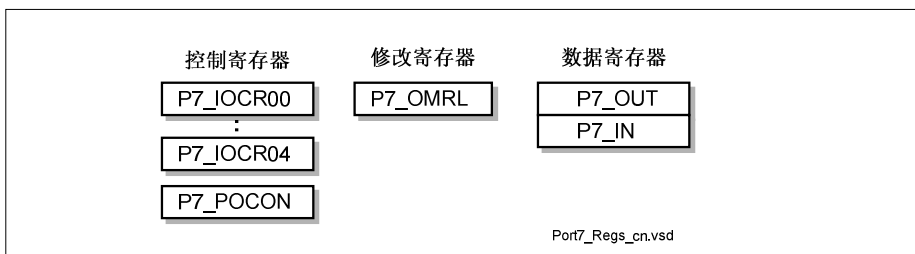
**表 9-11 P6 口寄存器**

寄存器缩略名	寄存器完整名	偏移地址	复位值
P6_OUT	P6 口输出寄存器	FFAE <sub>H</sub>	0000 <sub>H</sub>
P6_IN	P6 口输入寄存器	FF8C <sub>H</sub>	0000 <sub>H</sub>
P6_OMRL	P6 口输出修改低位寄存器	E9D8 <sub>H</sub>	XXXX <sub>H</sub>
P6_POCON	P6 口输出控制寄存器	E8AC <sub>H</sub>	0000 <sub>H</sub>
P6_IOCRO0	P6 口输入/输出控制寄存器 0	E8C0 <sub>H</sub>	0000 <sub>H</sub>
P6_IOCRO1	P6 口输入/输出控制寄存器 1	E8C2 <sub>H</sub>	0000 <sub>H</sub>
P6_IOCRO2	P6 口输入/输出控制寄存器 2	E8C4 <sub>H</sub>	0000 <sub>H</sub>
P6_IOCRO3	P6 口输入/输出控制寄存器 3	E8C6 <sub>H</sub>	0000 <sub>H</sub>

### 9.3.8 P7 口

P7 口为 5 位 GPIO 端口。P7 口的寄存器如图 9-11 所示。

该端口的所有引脚可用作 GPIO 从端口输入寄存器读取电平值。



**图 9-11 P7 口寄存器概览**

**表 9-12 P7 口寄存器**

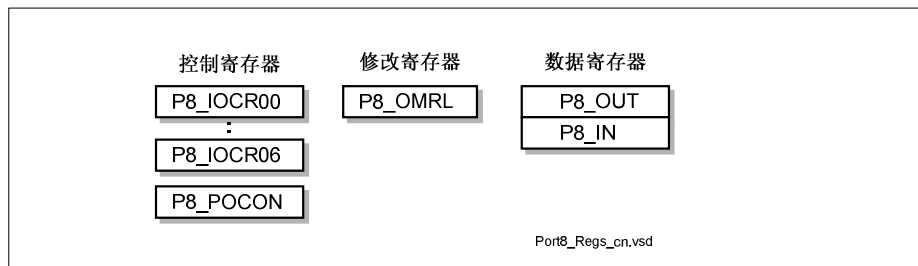
寄存器缩略名	寄存器完整名	偏移地址	复位值
P7_OUT	P7 口输出寄存器	FFB0 <sub>H</sub>	0000 <sub>H</sub>
P7_IN	P7 口输入寄存器	FF8E <sub>H</sub>	0000 <sub>H</sub>

寄存器缩略名	寄存器完整名	偏移地址	复位值
P7_OMRL	P7 口输出修改低位寄存器	E9DC <sub>H</sub>	XXXX <sub>H</sub>
P7_POCON	P7 口输出控制寄存器	E8AE <sub>H</sub>	0000 <sub>H</sub>
P7_IOCR00	P7 口输入/输出控制寄存器 0	E8E0 <sub>H</sub>	0000 <sub>H</sub>
P7_IOCR01	P7 口输入/输出控制寄存器 1	E8E2 <sub>H</sub>	0000 <sub>H</sub>
P7_IOCR02	P7 口输入/输出控制寄存器 2	E8E4 <sub>H</sub>	0000 <sub>H</sub>
P7_IOCR03	P7 口输入/输出控制寄存器 3	E8E6 <sub>H</sub>	0000 <sub>H</sub>
P7_IOCR04	P7 口输入/输出控制寄存器 4	E8E8 <sub>H</sub>	0000 <sub>H</sub>

### 9.3.9 P8 口

P8 口为 7 位 GPIO 端口。P8 口的寄存器如图 9-12 所示。

该端口的所有引脚可用作 GPIO 从端口输入寄存器读取电平值。



**图 9-12 P8 口寄存器概览**

**表 9-13 P8 口寄存器**

寄存器缩略名	寄存器完整名	偏移地址	复位值
P8_OUT	P8 口输出寄存器	FFB2 <sub>H</sub>	0000 <sub>H</sub>
P8_IN	P8 口输入寄存器	FF90 <sub>H</sub>	0000 <sub>H</sub>
P8_OMRL	P8 口输出修改低位寄存器	E9E0 <sub>H</sub>	XXXX <sub>H</sub>
P8_POCON	P8 口输出控制寄存器	E8B0 <sub>H</sub>	0000 <sub>H</sub>

寄存器缩略名	寄存器完整名	偏移地址	复位值
P8_IOCRR00	P8 口输入/输出控制寄存器 0	E900 <sub>H</sub>	0000 <sub>H</sub>
P8_IOCRR01	P8 口输入/输出控制寄存器 1	E902 <sub>H</sub>	0000 <sub>H</sub>
P8_IOCRR02	P8 口输入/输出控制寄存器 2	E904 <sub>H</sub>	0000 <sub>H</sub>
P8_IOCRR03	P8 口输入/输出控制寄存器 3	E906 <sub>H</sub>	0000 <sub>H</sub>
P8_IOCRR04	P8 口输入/输出控制寄存器 4	E908 <sub>H</sub>	0000 <sub>H</sub>
P8_IOCRR05	P8 口输入/输出控制寄存器 5	E90A <sub>H</sub>	0000 <sub>H</sub>
P8_IOCRR06	P8 口输入/输出控制寄存器 6	E90C <sub>H</sub>	0000 <sub>H</sub>

### 9.3.10 P9 口

P9 口为 8 位 GPIO 端口。P9 口的寄存器如图 9-13 所示。

该端口的所有引脚可用作 GPIO 从端口输入寄存器读取电平值。

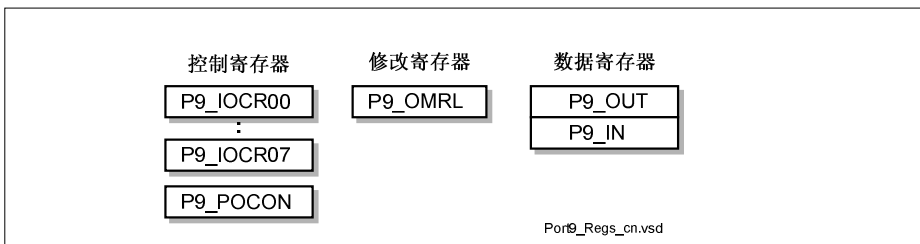


图 9-13 P9 口寄存器概览

表 9-14 P9 口寄存器

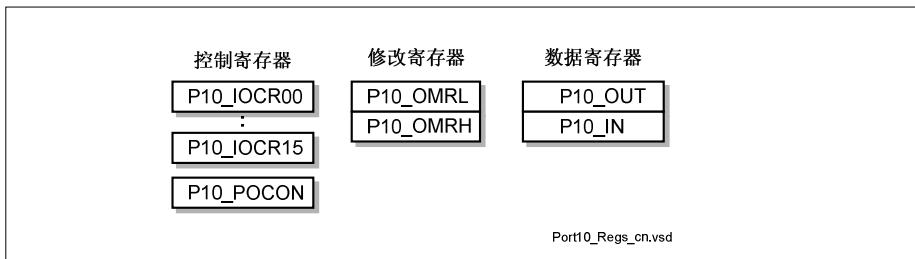
寄存器缩略名	寄存器完整名	偏移地址	复位值
P9_OUT	P9 口输出寄存器	FFB4 <sub>H</sub>	0000 <sub>H</sub>
P9_IN	P9 口输入寄存器	FF92 <sub>H</sub>	0000 <sub>H</sub>
P9_OMRL	P9 口输出修改低位寄存器	E9E4 <sub>H</sub>	XXXX <sub>H</sub>
P9_POCON	P9 口输出控制寄存器	E8B2 <sub>H</sub>	0000 <sub>H</sub>

寄存器缩略名	寄存器完整名	偏移地址	复位值
P9_IOCR00	P9 口输入/输出控制寄存器 0	E920 <sub>H</sub>	0000 <sub>H</sub>
P9_IOCR01	P9 口输入/输出控制寄存器 1	E922 <sub>H</sub>	0000 <sub>H</sub>
P9_IOCR02	P9 口输入/输出控制寄存器 2	E924 <sub>H</sub>	0000 <sub>H</sub>
P9_IOCR03	P9 口输入/输出控制寄存器 3	E926 <sub>H</sub>	0000 <sub>H</sub>
P9_IOCR04	P9 口输入/输出控制寄存器 4	E928 <sub>H</sub>	0000 <sub>H</sub>
P9_IOCR05	P9 口输入/输出控制寄存器 5	E92A <sub>H</sub>	0000 <sub>H</sub>
P9_IOCR06	P9 口输入/输出控制寄存器 6	E92C <sub>H</sub>	0000 <sub>H</sub>
P9_IOCR07	P9 口输入/输出控制寄存器 7	E92E <sub>H</sub>	0000 <sub>H</sub>

### 9.3.11 P10 口

P10 口为 16 位 GPIO 端口。P10 口的寄存器如图 9-14 所示。

该端口的所有引脚可用作 GPIO 从端口输入寄存器读取电平值。



**图 9-14 P10 口寄存器概览**

**表 9-15 P10 口寄存器**

寄存器缩略名	寄存器完整名	偏移地址	复位值
P10_OUT	P9 口输出寄存器	FFB6 <sub>H</sub>	0000 <sub>H</sub>
P10_IN	P9 口输入寄存器	FF94 <sub>H</sub>	0000 <sub>H</sub>
P10_OMRL	P9 口输出修改低位寄存器	E9E8 <sub>H</sub>	XXXX <sub>H</sub>

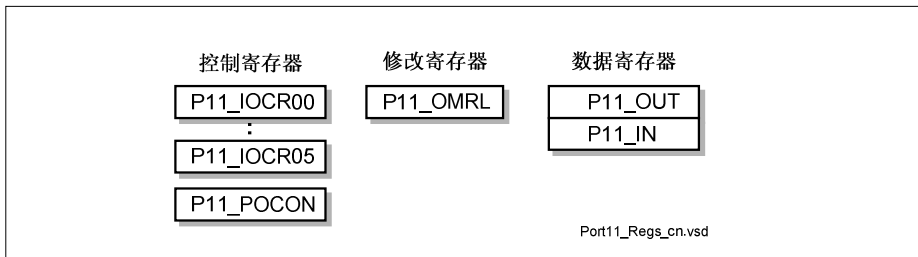
寄存器缩略名	寄存器完整名	偏移地址	复位值
P10_OMRH	P9 口输出修改高位寄存器	E9EA <sub>H</sub>	XXXX <sub>H</sub>
P10_POCON	P9 口输出控制寄存器	E8B4 <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO0	P9 口输入/输出控制寄存器 0	E940 <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO1	P9 口输入/输出控制寄存器 1	E942 <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO2	P9 口输入/输出控制寄存器 2	E944 <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO3	P9 口输入/输出控制寄存器 3	E946 <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO4	P9 口输入/输出控制寄存器 4	E948 <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO5	P9 口输入/输出控制寄存器 5	E94A <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO6	P9 口输入/输出控制寄存器 6	E94C <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO7	P9 口输入/输出控制寄存器 7	E94E <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO8	P9 口输入/输出控制寄存器 8	E950 <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO9	P9 口输入/输出控制寄存器 9	E952 <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO10	P9 口输入/输出控制寄存器 10	E954 <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO11	P9 口输入/输出控制寄存器 11	E956 <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO12	P9 口输入/输出控制寄存器 12	E958 <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO13	P9 口输入/输出控制寄存器 13	E95A <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO14	P9 口输入/输出控制寄存器 14	E95C <sub>H</sub>	0000 <sub>H</sub>
P10_IOCRO15	P9 口输入/输出控制寄存器 15	E95E <sub>H</sub>	0000 <sub>H</sub>

### 9.3.12 P11 口

P11 口为 6 位 GPIO 端口。P11 口的寄存器如图 9-15 所示。

该端口的所有引脚可用作 GPIO 从端口输入寄存器读取电平值。





**图 9-15 P11 口寄存器概览**

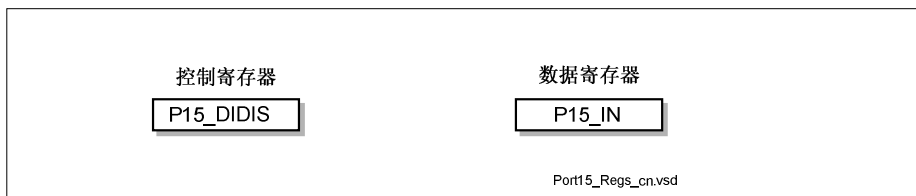
**表 9-16 P11 口寄存器**

寄存器缩略名	寄存器完整名	偏移地址	复位值
P11_OUT	P11 口输出寄存器	FFB8 <sub>H</sub>	0000 <sub>H</sub>
P11_IN	P11 口输入寄存器	FF96 <sub>H</sub>	0000 <sub>H</sub>
P11_OMRL	P11 口输出修改低位寄存器	E9EC <sub>H</sub>	XXXX <sub>H</sub>
P11_POCON	P11 口输出控制寄存器	E8B6 <sub>H</sub>	0000 <sub>H</sub>
P11_IOCRO0	P11 口输入/输出控制寄存器 0	E960 <sub>H</sub>	0000 <sub>H</sub>
P11_IOCRO1	P11 口输入/输出控制寄存器 1	E962 <sub>H</sub>	0000 <sub>H</sub>
P11_IOCRO2	P11 口输入/输出控制寄存器 2	E964 <sub>H</sub>	0000 <sub>H</sub>
P11_IOCRO3	P11 口输入/输出控制寄存器 3	E966 <sub>H</sub>	0000 <sub>H</sub>
P11_IOCRO4	P11 口输入/输出控制寄存器 4	E968 <sub>H</sub>	0000 <sub>H</sub>
P11_IOCRO5	P11 口输入/输出控制寄存器 5	E96A <sub>H</sub>	0000 <sub>H</sub>

### 9.3.13 P15 口

P15 口为 8 位模拟或数字输入端口。

P15 口用作模拟输入时，必须禁用施密特触发器，通过置位寄存器 P15\_DIDIS 中的相应位来实现。



**图 9-16 P15 口寄存器概览**

**表 9-17 P15 口寄存器**

寄存器缩略名	寄存器完整名	偏移地址	复位值
P15_IN	P15 口输入寄存器	FF9E <sub>H</sub>	0000 <sub>H</sub>
P15_DIDIS	P15 口数字输入禁用寄存器	FE9E <sub>H</sub>	0000 <sub>H</sub>

## 9.4 引脚描述

XE16xyM 的每个端口引脚可提供不同模块的多种功能。同样，大多数功能可从多个端口引脚上获得。因而，实际应用可根据自身特定的需求选择最优的引脚连接。

引脚可输出本身的端口输出信号或来自外设的信号（从最多 3 路信号线中选择 1 路输出）。引脚的输入信号可从其本身的输入寄存器中或多个外设上获得。

*注：由相应的端口引脚选择输出信号；由相应的外设选择输入信号。*

某引脚可选择完全由外设控制（假如外设被使能，如 EBC）。

XE16xyM 中各端口和引脚的功能归纳见**表 9-18**。“引脚”一列对应 PG-LQFP-144 封装。

### 引脚定义备注

- 控制：** 端口引脚的输出信号由相关寄存器  $Px\_IOCRy$  中的位域  $PC$  进行选择。设置位域  $PC$  为  $1x00_B$  选择输出  $O0$ ；设置为  $1x01_B$  选择输出  $O1$ ，以此类推。输出信号  $OH$  由硬件控制。
- 类型：** 指示所使用的引出端类型（ $St$ =标准引出端； $Sp$ =特殊引出端； $DP$ =双引出端； $In$ =输入引出端； $PS$ =电源）和电源域（ $A$ 、 $B$ 、 $M$ 、 $1$ ）。

**表 9-18 引脚定义和功能**

引脚	符号	控制	类型	功能
3	$\overline{\text{TESTM}}$	I	In/B	<b>测试模式使能</b> 使能工厂测试模式，正常工作时必须保持高电平（连至 $V_{DDPB}$ ）。 该引脚不被驱动时，内部上拉器件将其拉高。
4	P7.2	O0/I	St/B	<b>P7 口的位 2，通用输入/输出</b>
	EMUX0	O1	St/B	<b>外部模拟复用器的控制输出 0（ADC1）</b>
	TxDC4	O2	St/B	<b>CAN 节点 4 发送数据输出</b>
	TxDC5	O3	St/B	<b>CAN 节点 5 发送数据输出</b>
	CCU62_CCP OS0A	I	St/B	<b>CCU62 位置输入 0</b>
	TDI_C	I	St/B	<b>JTAG 测试数据输入</b>
5	P8.4	O0/I	St/B	<b>P8 口的位 4，通用输入/输出</b>
	CCU60_COU T61	O1	St/B	<b>CCU60 通道 1 输出</b>
	CCU62_CC6 1	O2	St/B	<b>CCU62 通道 1 输出</b>
	TMS_D	I	St/B	<b>JTAG 测试模式选择输入</b>
	CCU62_CC6 1INB	I	St/B	<b>CCU62 通道 1 输入</b>
6	$\overline{\text{TRST}}$	I	In/B	<b>测试系统复位输入</b> 系统正常工作时，该引脚应保持低电平。 $\overline{\text{PORST}}$ 发生上升沿跳变时，该引脚上的高电平将激活 XE16xyM 的调试系统。在这种情况下，一旦复位调试系统，引脚 $\overline{\text{TRST}}$ 必须拉低。 该引脚不被驱动时，内部下拉器件将其拉低。
7	P8.3	O0/I	St/B	<b>P8 口的位 3，通用输入/输出</b>

引脚	符号	控制	类型	功能
	CCU60_COU T60	O1	St/B	<b>CCU60 通道 0 输出</b>
	CCU62_CC6 0	O2	St/B	<b>CCU62 通道 0 输出</b>
	TDI_D	I	St/B	<b>JTAG 测试数据输入</b>
	CCU62_CC6 0INB	I	St/B	<b>CCU62 通道 0 输入</b>
8	P7.0	O0/I	St/B	<b>P7 口的位 0，通用输入/输出</b>
	T3OUT	O1	St/B	<b>GPT12E 定时器 T3 翻转锁存输出</b>
	T6OUT	O2	St/B	<b>GPT12E 定时器 T6 翻转锁存输出</b>
	TDO_A	OH/I	St/B	<b>JTAG 测试数据输出/DAP1 输入/输出</b>
	ESR2_1	I	St/B	<b>ESR2 触发输入 1</b>
	RxDC4B	I	St/B	<b>CAN 节点 4 接收数据输入</b>
9	P7.3	O0/I	St/B	<b>P7 口的位 3，通用输入/输出</b>
	EMUX1	O1	St/B	<b>外部模拟复用器的控制输出 1（ADC1）</b>
	U0C1_DOUT	O2	St/B	<b>USIC0 通道 1 移位数据输出</b>
	U0C0_DOUT	O3	St/B	<b>USIC0 通道 0 移位数据输出</b>
	CCU62_CCP OS1A	I	St/B	<b>CCU62 位置输入 1</b>
	TMS_C	I	St/B	<b>JTAG 测试模式选择输入</b>
	U0C1_DX0F	I	St/B	<b>USIC0 通道 1 移位数据输入</b>
10	P8.2	O0/I	St/B	<b>P8 口的位 2，通用输入/输出</b>
	CCU60_CC6 2	O1	St/B	<b>CCU60 通道 2 输出</b>
	TxDC1	O2	St/B	<b>CAN 节点 1 发送数据输出</b>
	U1C1_DOUT	O3	St/B	<b>USIC1 通道 1 移位数据输出</b>
	CCU60_CC6 2INB	I	St/B	<b>CCU60 通道 2 输入</b>

引脚	符号	控制	类型	功能
11	P7.1	O0/I	St/B	<b>P7</b> 口的位 <b>1</b> ，通用输入/输出
	EXTCLK	O1	St/B	可编程时钟信号输出
	TxDC4	O2	St/B	<b>CAN</b> 节点 <b>4</b> 发送数据输出
	CCU62_CTR APA	I	St/B	<b>CCU62</b> 急停强制中断输入
	BRKIN_C	I	St/B	<b>OCDS</b> 断点信号输入
12	P7.4	O0/I	St/B	<b>P7</b> 口的位 <b>4</b> ，通用输入/输出
	EMUX2	O1	St/B	外部模拟复用器的控制输出 <b>2</b> ( <b>ADC1</b> )
	U0C1_DOUT	O2	St/B	<b>USIC0</b> 通道 <b>1</b> 移位数据输出
	U0C1_SCLK OUT	O3	St/B	<b>USIC0</b> 通道 <b>1</b> 移位时钟输出
	CCU62_CCP OS2A	I	St/B	<b>CCU62</b> 位置输入 <b>2</b>
	TCK_C	I	St/B	<b>DAP0/JTAG</b> 时钟输入
	U0C0_DX0D	I	St/B	<b>USIC0</b> 通道 <b>0</b> 移位数据输入
	U0C1_DX1E	I	St/B	<b>USIC0</b> 通道 <b>1</b> 移位时钟输入
13	P8.1	O0/I	St/B	<b>P8</b> 口的位 <b>1</b> ，通用输入/输出
	CCU60_CC6 1	O1	St/B	<b>CCU60</b> 通道 <b>1</b> 输出
	CCU60_CC6 1INB	I	St/B	<b>CCU60</b> 通道 <b>1</b> 输入
	RxDC1F	I	St/B	<b>CAN</b> 节点 <b>1</b> 接收数据输入
14	P8.0	O0/I	St/B	<b>P8</b> 口的位 <b>0</b> ，通用输入/输出
	CCU60_CC6 0	O1	St/B	<b>CCU60</b> 通道 <b>0</b> 输出
	CCU60_CC6 0INB	I	St/B	<b>CCU60</b> 通道 <b>0</b> 输入
16	P6.0	O0/I	St/A	<b>P6</b> 口的位 <b>0</b> ，通用输入/输出

引脚	符号	控制	类型	功能
	EMUX0	O1	St/A	外部模拟复用器的控制输出 0 (ADC0)
	TxDC2	O2	St/A	CAN 节点 2 发送数据输出
	BRKOUT	O3	St/A	OCDS 断点信号输出
	ADCx_REQG TyG	I	St/A	ADC0/1 的外部请求门控输入
	U1C1_DX0E	I	St/A	USIC1 通道 1 移位数据输入
17	P6.1	O0/I	St/A	P6 口的位 1, 通用输入/输出
	EMUX1	O1	St/A	外部模拟复用器的控制输出 1 (ADC0)
	T3OUT	O2	St/A	GPT12E 定时器 T3 翻转锁存输出
	U1C1_DOUT	O3	St/A	USIC1 通道 1 移位数据输出
	ADCx_REQT RyE	I	St/A	ADC0/1 的外部请求触发输入
	RxDC2E	I	St/A	CAN 节点 2 接收数据输入
	ESR1_6	I	St/A	ESR1 触发输入 6
18	P6.2	O0/I	St/A	P6 口的位 2, 通用输入/输出
	EMUX2	O1	St/A	外部模拟复用器的控制输出 2 (ADC0)
	T6OUT	O2	St/A	GPT12E 定时器 T6 翻转锁存输出
	U1C1_SCLK OUT	O3	St/A	USIC1 通道 1 移位时钟输出
	U1C1_DX1C	I	St/A	USIC1 通道 1 移位时钟输入
19	P6.3	O0/I	St/A	P6 口的位 3, 通用输入/输出
	T3OUT	O2	St/A	GPT12E 定时器 T3 翻转锁存输出
	U1C1_SELO 0	O3	St/A	USIC1 通道 1 选择/控制 0 输出
	U1C1_DX2D	I	St/A	USIC1 通道 1 移位控制输入
	ADCx_REQT RyF	I	St/A	ADC0/1 的外部请求触发输入

引脚	符号	控制	类型	功能
21	P15.0	I	In/A	<b>P15 口的位 0，通用输入</b>
	ADC1_CH0	I	In/A	<b>ADC1 的模拟输入通道 0</b>
22	P15.1	I	In/A	<b>P15 口的位 1，通用输入</b>
	ADC1_CH1	I	In/A	<b>ADC1 的模拟输入通道 1</b>
23	P15.2	I	In/A	<b>P15 口的位 2，通用输入</b>
	ADC1_CH2	I	In/A	<b>ADC1 的模拟输入通道 2</b>
	T5INA	I	In/A	<b>GPT12E 定时器 T5 计数/门控输入</b>
24	P15.3	I	In/A	<b>P15 口的位 3，通用输入</b>
	ADC1_CH3	I	In/A	<b>ADC1 的模拟输入通道 3</b>
	T5EUDA	I	In/A	<b>GPT12E 定时器 T5 外部递增/递减控制输入</b>
25	P15.4	I	In/A	<b>P15 口的位 4，通用输入</b>
	ADC1_CH4	I	In/A	<b>ADC1 的模拟输入通道 4</b>
	T6INA	I	In/A	<b>GPT12E 定时器 T6 计数/门控输入</b>
26	P15.5	I	In/A	<b>P15 口的位 5，通用输入</b>
	ADC1_CH5	I	In/A	<b>ADC1 的模拟输入通道 5</b>
	T6EUDA	I	In/A	<b>GPT12E 定时器 T6 外部递增/递减控制输入</b>
27	P15.6	I	In/A	<b>P15 口的位 6，通用输入</b>
	ADC1_CH6	I	In/A	<b>ADC1 的模拟输入通道 6</b>
28	P15.7	I	In/A	<b>P15 口的位 7，通用输入</b>
	ADC1_CH7	I	In/A	<b>ADC1 的模拟输入通道 7</b>
29	V <sub>AREF1</sub>	-	PS/A	<b>A/D 转换器 ADC1 的参考电压</b>
30	V <sub>AREF0</sub>	-	PS/A	<b>A/D 转换器 ADC0 的参考电压</b>
31	V <sub>AGND</sub>	-	PS/A	<b>A/D 转换器 ADC0 的参考地</b>

引脚	符号	控制	类型	功能
32	P5.0	I	In/A	<b>P5 口的位 0，通用输入</b>
	ADC0_CH0	I	In/A	<b>ADC0 的模拟输入通道 0</b>
33	P5.1	I	In/A	<b>P5 口的位 1，通用输入</b>
	ADC0_CH1	I	In/A	<b>ADC0 的模拟输入通道 1</b>
34	P5.2	I	In/A	<b>P5 口的位 2，通用输入</b>
	ADC0_CH2	I	In/A	<b>ADC0 的模拟输入通道 2</b>
	TDI_A	I	In/A	<b>JTAG 测试数据输入</b>
35	P5.3	I	In/A	<b>P5 口的位 3，通用输入</b>
	ADC0_CH3	I	In/A	<b>ADC0 的模拟输入通道 3</b>
	T3INA	I	In/A	<b>GPT12E 定时器 T3 计数/门控输入</b>
39	P5.4	I	In/A	<b>P5 口的位 4，通用输入</b>
	ADC0_CH4	I	In/A	<b>ADC0 的模拟输入通道 4</b>
	CCU63_T12H RB	I	In/A	<b>CCU63 T12 的外部运行控制输入</b>
	T3EUDA	I	In/A	<b>GPT12E 定时器 T3 外部递增/递减控制输入</b>
	TMS_A	I	In/A	<b>JTAG 测试模式选择输入</b>
40	P5.5	I	In/A	<b>P5 口的位 5，通用输入</b>
	ADC0_CH5	I	In/A	<b>ADC0 的模拟输入通道 5</b>
	CCU60_T12H RB	I	In/A	<b>CCU60 T12 的外部运行控制输入</b>
41	P5.6	I	In/A	<b>P5 口的位 6，通用输入</b>
	ADC0_CH6	I	In/A	<b>ADC0 的模拟输入通道 6</b>
42	P5.7	I	In/A	<b>P5 口的位 7，通用输入</b>
	ADC0_CH7	I	In/A	<b>ADC0 的模拟输入通道 7</b>
43	P5.8	I	In/A	<b>P5 口的位 8，通用输入</b>



引脚	符号	控制	类型	功能
	ADC0_CH8	I	In/A	<b>ADC0</b> 的模拟输入通道 8
	ADC1_CH8	I	In/A	<b>ADC1</b> 的模拟输入通道 8
	CCU6x_T12H RC	I	In/A	<b>CCU60/1/2/3 T12</b> 的外部运行控制输入
	CCU6x_T13H RC	I	In/A	<b>CCU60/1/2/3 T13</b> 的外部运行控制输入
	U2C0_DX0F	I	In/A	<b>USIC2</b> 通道 0 移位数据输入
44	P5.9	I	In/A	<b>P5</b> 口的位 9，通用输入
	ADC0_CH9	I	In/A	<b>ADC0</b> 的模拟输入通道 9
	ADC1_CH9	I	In/A	<b>ADC1</b> 的模拟输入通道 9
	CC2_T7IN	I	In/A	<b>CAPCOM2</b> 定时器 T7 计数输入
45	P5.10	I	In/A	<b>P5</b> 口的位 10，通用输入
	ADC0_CH10	I	In/A	<b>ADC0</b> 的模拟输入通道 10
	ADC1_CH10	I	In/A	<b>ADC1</b> 的模拟输入通道 10
	$\overline{\text{BRKIN\_A}}$	I	In/A	<b>OCDS</b> 断点信号输入
	U2C1_DX0F	I	In/A	<b>USIC2</b> 通道 1 移位数据输入
	CCU61_T13H RA	I	In/A	<b>CCU61 T13</b> 的外部运行控制输入
46	P5.11	I	In/A	<b>P5</b> 口的位 11，通用输入
	ADC0_CH11	I	In/A	<b>ADC0</b> 的模拟输入通道 11
	ADC1_CH11	I	In/A	<b>ADC1</b> 的模拟输入通道 11
47	P5.12	I	In/A	<b>P5</b> 口的位 12，通用输入
	ADC0_CH12	I	In/A	<b>ADC0</b> 的模拟输入通道 12
48	P5.13	I	In/A	<b>P5</b> 口的位 13，通用输入
	ADC0_CH13	I	In/A	<b>ADC0</b> 的模拟输入通道 13
	CCU63_T13H	I	In/A	<b>CCU63 T13</b> 的外部运行控制输入

引脚	符号	控制	类型	功能
	RF			
49	P5.14	I	In/A	<b>P5 口的位 14, 通用输入</b>
	ADC0_CH14	I	In/A	<b>ADC0 的模拟输入通道 14</b>
50	P5.15	I	In/A	<b>P5 口的位 15, 通用输入</b>
	ADC0_CH15	I	In/A	<b>ADC0 的模拟输入通道 15</b>
	RxDC2F	I	In/A	<b>CAN 节点 2 接收数据输入</b>
51	P2.12	O0/I	St/B	<b>P2 口的位 12, 通用输入/输出</b>
	U0C0_SELO 4	O1	St/B	<b>USIC0 通道 0 选择/控制 4 输出</b>
	U0C1_SELO 3	O2	St/B	<b>USIC0 通道 1 选择/控制 3 输出</b>
	TXDC2	O3	St/B	<b>CAN 节点 2 发送数据输出</b>
	READY	I	St/B	<b>外部总线接口 READY 输入</b>
52	P2.11	O0/I	St/B	<b>P2 口的位 11, 通用输入/输出</b>
	U0C0_SELO 2	O1	St/B	<b>USIC0 通道 0 选择/控制 2 输出</b>
	U0C1_SELO 2	O2	St/B	<b>USIC0 通道 1 选择/控制 2 输出</b>
	U3C1_DOUT	O3	St/B	<b>USIC3 通道 1 移位数据输出</b>
	$\overline{\text{BHE}} / \overline{\text{WRH}}$	OH	St/B	<b>外部总线接口高字节控制输出</b> 可用作高字节使能 ( $\overline{\text{BHE}}$ ) 或高字节写选通 ( $\overline{\text{WRH}}$ )
53	P11.5	O0/I	St/B	<b>P11 口的位 5, 通用输入/输出</b>
	CCU61_CC6 0	O1	St/B	<b>CCU61 通道 0 输出</b>
	CCU61_COU T63	O2	St/B	<b>CCU61 通道 3 输出</b>
	U3C1_SELO 1	O3	St/B	<b>USIC3 通道 1 选择/控制 1 输出</b>

引脚	符号	控制	类型	功能
	CCU61_CC60INB	I	St/B	<b>CCU61 通道 0 输入</b>
	U3C1_DX2B	I	St/B	<b>USIC3 通道 1 移位控制输入</b>
55	P2.0	O0/I	St/B	<b>P2 口的位 0，通用输入/输出</b>
	TxDC5	O1	St/B	<b>CAN 节点 5 发送数据输出</b>
	CCU63_CC60	O2	St/B	<b>CCU63 通道 0 输出</b>
	AD13	OH/I	St/B	<b>外部总线接口地址/数据线 13</b>
	RxDC0C	I	St/B	<b>CAN 节点 0 接收数据输入</b>
	CCU63_CC60INB	I	St/B	<b>CCU63 通道 0 输入</b>
	T5INB	I	St/B	<b>GPT12E 定时器 T5 计数/门控输入</b>
56	P2.1	O0/I	St/B	<b>P2 口的位 1，通用输入/输出</b>
	TxDC0	O1	St/B	<b>CAN 节点 0 发送数据输出</b>
	CCU63_CC61	O2	St/B	<b>CCU63 通道 1 输出</b>
	AD14	OH/I	St/B	<b>外部总线接口地址/数据线 14</b>
	RxDC5C	I	St/B	<b>CAN 节点 5 接收数据输入</b>
	CCU63_CC61INB	I	St/B	<b>CCU63 通道 1 输入</b>
	T5EUIDB	I	St/B	<b>GPT12E 定时器 T5 外部递增/递减控制输入</b>
	ESR1_5	I	St/B	<b>ESR1 触发输入 5</b>
57	P11.4	O0/I	St/B	<b>P11 口的位 4，通用输入/输出</b>
	CCU61_CC62	O1	St/B	<b>CCU61 通道 2 输出</b>
	U3C1_DOUT	O2	St/B	<b>USIC3 通道 1 移位数据输出</b>
	RxDC5B	I	St/B	<b>CAN 节点 5 接收数据输入</b>

引脚	符号	控制	类型	功能
	CCU61_CC62INB	I	St/B	<b>CCU61 通道 2 输入</b>
	U3C1_DX0B	I	St/B	<b>USIC3 通道 1 移位数据输入</b>
58	P2.2	O0/I	St/B	<b>P2 口的位 2, 通用输入/输出</b>
	TxDC1	O1	St/B	<b>CAN 节点 1 发送数据输出</b>
	CCU63_CC62	O2	St/B	<b>CCU63 通道 2 输出</b>
	AD15	OH/I	St/B	<b>外部总线接口地址/数据线 15</b>
	CCU63_CC62INB	I	St/B	<b>CCU63 通道 2 输入</b>
	ESR2_5	I	St/B	<b>ESR2 触发输入 5</b>
59	P11.3	O0/I	St/B	<b>P11 口的位 3, 通用输入/输出</b>
	CCU61_COUT63	O1	St/B	<b>CCU61 通道 3 输出</b>
	CCU61_COUT62	O2	St/B	<b>CCU61 通道 2 输出</b>
	TxDC5	O3	St/B	<b>CAN 节点 5 发送数据输出</b>
	CCU61_T13H RF	I	St/B	<b>CCU61 T13 的外部运行控制输入</b>
60	P4.0	O0/I	St/B	<b>P4 口的位 0, 通用输入/输出</b>
	CC2_CC24	O3/I	St/B	<b>CAPCOM2 CC24IO 捕获输入/比较输出</b>
	$\overline{\text{CS0}}$	OH	St/B	<b>外部总线接口片选 0 输出</b>
61	P2.3	O0/I	St/B	<b>P2 口的位 3, 通用输入/输出</b>
	U0C0_DOUT	O1	St/B	<b>USIC0 通道 0 移位数据输出</b>
	CCU63_COUT63	O2	St/B	<b>CCU63 通道 3 输出</b>
	CC2_CC16	O3/I	St/B	<b>CAPCOM2 CC16IO 捕获输入/比较输出</b>
	A16	OH	St/B	<b>外部总线接口地址线 16</b>

引脚	符号	控制	类型	功能
	ESR2_0	I	St/B	<b>ESR2 触发输入 0</b>
	U0C0_DX0E	I	St/B	<b>USIC0 通道 0 移位数据输入</b>
	U0C1_DX0D	I	St/B	<b>USIC0 通道 1 移位数据输入</b>
	RxDC0A	I	St/B	<b>CAN 节点 0 接收数据输入</b>
62	P11.2	O0/I	St/B	<b>P11 口的位 2, 通用输入/输出</b>
	CCU61_CC61	O1	St/B	<b>CCU61 通道 1 输出</b>
	U3C1_DOUT	O2	St/B	<b>USIC3 通道 1 移位数据输出</b>
	CCU63_CCP OS2A	I	St/B	<b>CCU63 位置输入 2</b>
	CCU61_CC61INB	I	St/B	<b>CCU61 通道 1 输入</b>
63	P4.1	O0/I	St/B	<b>P4 口的位 1, 通用输入/输出</b>
	U3C0_SELO3	O1	St/B	<b>USIC3 通道 0 选择/控制 3 输出</b>
	TxDC2	O2	St/B	<b>CAN 节点 2 发送数据输出</b>
	CC2_CC25	O3/I	St/B	<b>CAPCOM2 CC25IO 捕获输入/比较输出</b>
	$\overline{\text{CS1}}$	OH	St/B	<b>外部总线接口片选 1 输出</b>
	CCU62_CCP OS0B	I	St/B	<b>CCU62 位置输入 0</b>
	T4EUDB	I	St/B	<b>GPT12E 定时器 T4 外部递增/递减控制输入</b>
	ESR1_8	I	St/B	<b>ESR1 触发输入 8</b>
64	P2.4	O0/I	St/B	<b>P2 口的位 4, 通用输入/输出</b>
	U0C1_DOUT	O1	St/B	<b>USIC0 通道 1 移位数据输出</b>
	TxDC0	O2	St/B	<b>CAN 节点 0 发送数据输出</b>
	CC2_CC17	O3/I	St/B	<b>CAPCOM2 CC17IO 捕获输入/比较输出</b>

引脚	符号	控制	类型	功能
	A17	OH	St/B	外部总线接口地址线 17
	ESR1_0	I	St/B	<b>ESR1 触发输入 0</b>
	U0C0_DX0F	I	St/B	<b>USIC0 通道 0 移位数据输入</b>
	RxDC1A	I	St/B	<b>CAN 节点 1 接收数据输入</b>
65	P11.1	O0/I	St/B	<b>P11 口的位 1, 通用输入/输出</b>
	CCU61_COU T61	O1	St/B	<b>CCU61 通道 1 输出</b>
	TxDC0	O2	St/B	<b>CAN 节点 0 发送数据输出</b>
	U3C1_SELO 0	O3	St/B	<b>USIC3 通道 1 选择/控制 0 输出</b>
	CCU63_CCP OS1A	I	St/B	<b>CCU63 位置输入 1</b>
	CCU61_CTR APD	I	St/B	<b>CCU61 急停强制中断输入</b>
	U3C1_DX2A	I	St/B	<b>USIC3 通道 1 移位控制输入</b>
66	P11.0	O0/I	St/B	<b>P11 口的位 0, 通用输入/输出</b>
	CCU61_COU T60	O1	St/B	<b>CCU61 通道 0 输出</b>
	U3C1_SCLK OUT	O2	St/B	<b>USIC3 通道 1 移位时钟输出</b>
	CCU63_CCP OS0A	I	St/B	<b>CCU63 位置输入 0</b>
	RxDC0F	I	St/B	<b>CAN 节点 0 接收数据输入</b>
	U3C1_DX1A	I	St/B	<b>USIC3 通道 1 移位时钟输入</b>
	ESR1_7	I	St/B	<b>ESR1 触发输入 7</b>
67	P2.5	O0/I	St/B	<b>P2 口的位 5, 通用输入/输出</b>
	U0C0_SCLK OUT	O1	St/B	<b>USIC0 通道 0 移位时钟输出</b>
	TxDC0	O2	St/B	<b>CAN 节点 0 发送数据输出</b>

引脚	符号	控制	类型	功能
	CC2_CC18	O3/I	St/B	<b>CAPCOM2 CC18IO</b> 捕获输入/比较输出
	A18	OH	St/B	<b>外部总线接口地址线 18</b>
	U0C0_DX1D	I	St/B	<b>USIC0 通道 0</b> 移位时钟输入
	ESR1_10	I	St/B	<b>ESR1 触发输入 10</b>
68	P4.2	O0/I	St/B	<b>P4 口的位 2</b> , 通用输入/输出
	U3C0_SCLK OUT	O1	St/B	<b>USIC3 通道 0</b> 移位时钟输出
	TxDC2	O2	St/B	<b>CAN 节点 2</b> 发送数据输出
	CC2_CC26	O3/I	St/B	<b>CAPCOM2 CC26IO</b> 捕获输入/比较输出
	$\overline{\text{CS2}}$	OH	St/B	<b>外部总线接口片选 2</b> 输出
	T2INA	I	St/B	<b>GPT12E 定时器 T2</b> 计数/门控输入
	CCU62_CCP OS1B	I	St/B	<b>CCU62 位置输入 1</b>
	U3C0_DX1B	I	St/B	<b>USIC3 通道 0</b> 移位时钟输入
69	P2.6	O0/I	St/B	<b>P2 口的位 6</b> , 通用输入/输出
	U0C0_SELO 0	O1	St/B	<b>USIC0 通道 0</b> 选择/控制 0 输出
	U0C1_SELO 1	O2	St/B	<b>USIC0 通道 1</b> 选择/控制 1 输出
	CC2_CC19	O3/I	St/B	<b>CAPCOM2 CC19IO</b> 捕获输入/比较输出
	A19	OH	St/B	<b>外部总线接口地址线 19</b>
	U0C0_DX2D	I	St/B	<b>USIC0 通道 0</b> 移位控制输入
	RxDC0D	I	St/B	<b>CAN 节点 0</b> 接收数据输入
	ESR2_6	I	St/B	<b>ESR2 触发输入 6</b>
70	P4.4	O0/I	St/B	<b>P4 口的位 4</b> , 通用输入/输出
	U3C0_SELO 2	O1	St/B	<b>USIC3 通道 0</b> 选择/控制 2 输出

引脚	符号	控制	类型	功能
	CC2_CC28	O3/I	St/B	<b>CAPCOM2 CC28IO</b> 捕获输入/比较输出
	$\overline{\text{CS4}}$	OH	St/B	外部总线接口片选 4 输出
	CLKIN2	I	St/B	时钟信号输入 2
	U3C0_DX2C	I	St/B	<b>USIC3</b> 通道 0 移位控制输入
71	P4.3	O0/I	St/B	<b>P4</b> 口的位 3, 通用输入/输出
	U0C1_DOUT	O1	St/B	<b>USIC0</b> 通道 1 移位数据输出
	CC2_CC27	O3/I	St/B	<b>CAPCOM2 CC27IO</b> 捕获输入/比较输出
	$\overline{\text{CS3}}$	OH	St/B	外部总线接口片选 3 输出
	RxDC2A	I	St/B	<b>CAN</b> 节点 2 接收数据输入
	T2EUDA	I	St/B	<b>GPT12E</b> 定时器 T2 外部递增/递减控制输入
	CCU62_CCP OS2B	I	St/B	<b>CCU62</b> 位置输入 2
75	P0.0	O0/I	St/B	<b>P0</b> 口的位 0, 通用输入/输出
	U1C0_DOUT	O1	St/B	<b>USIC1</b> 通道 0 移位数据输出
	CCU61_CC6 0	O3	St/B	<b>CCU61</b> 通道 0 输出
	A0	OH	St/B	外部总线接口地址线 0
	U1C0_DX0A	I	St/B	<b>USIC1</b> 通道 0 移位数据输入
	CCU61_CC6 0INA	I	St/B	<b>CCU61</b> 通道 0 输入
	ESR1_11	I	St/B	<b>ESR1</b> 触发输入 11
76	P4.5	O0/I	St/B	<b>P4</b> 口的位 5, 通用输入/输出
	U3C0_DOUT	O1	St/B	<b>USIC3</b> 通道 0 移位数据输出
	CC2_CC29	O3/I	St/B	<b>CAPCOM2 CC29IO</b> 捕获输入/比较输出
	CCU61_CCP OS0A	I	St/B	<b>CCU61</b> 位置输入 0



引脚	符号	控制	类型	功能
	U3C0_DX0B	I	St/B	<b>USIC3 通道 0 移位数据输入</b>
	ESR2_10	I	St/B	<b>ESR2 触发输入 10</b>
77	P4.6	O0/I	St/B	<b>P4 口的位 6, 通用输入/输出</b>
	U3C0_DOUT	O1	St/B	<b>USIC3 通道 0 移位数据输出</b>
	CC2_CC30	O3/I	St/B	<b>CAPCOM2 CC30IO 捕获输入/比较输出</b>
	T4INA	I	St/B	<b>GPT12E 定时器 T4 计数/门控输入</b>
	CCU61_CCP OS1A	I	St/B	<b>CCU61 位置输入 1</b>
78	P2.7	O0/I	St/B	<b>P2 口的位 7, 通用输入/输出</b>
	U0C1_SELO 0	O1	St/B	<b>USIC0 通道 1 选择/控制 0 输出</b>
	U0C0_SELO 1	O2	St/B	<b>USIC0 通道 0 选择/控制 1 输出</b>
	CC2_CC20	O3/I	St/B	<b>CAPCOM2 CC20IO 捕获输入/比较输出</b>
	A20	OH	St/B	<b>外部总线接口地址线 20</b>
	U0C1_DX2C	I	St/B	<b>USIC0 通道 1 移位控制输入</b>
	RxDC1C	I	St/B	<b>CAN 节点 1 接收数据输入</b>
	ESR2_7	I	St/B	<b>ESR2 触发输入 7</b>
79	P0.1	O0/I	St/B	<b>P0 口的位 1, 通用输入/输出</b>
	U1C0_DOUT	O1	St/B	<b>USIC1 通道 0 移位数据输出</b>
	TxDC0	O2	St/B	<b>CAN 节点 0 发送数据输出</b>
	CCU61_CC6 1	O3	St/B	<b>CCU61 通道 1 输出</b>
	A1	OH	St/B	<b>外部总线接口地址线 1</b>
	U1C0_DX0B	I	St/B	<b>USIC1 通道 0 移位数据输入</b>
	CCU61_CC6 1INA	I	St/B	<b>CCU61 通道 1 输入</b>

引脚	符号	控制	类型	功能
80	U1C0_DX1A	I	St/B	<b>USIC1 通道 0</b> 移位时钟输入
	P2.8	O0/I	DP/B	<b>P2</b> 口的位 <b>8</b> ，通用输入/输出
	U0C1_SCLK OUT	O1	DP/B	<b>USIC0 通道 1</b> 移位时钟输出
	EXTCLK	O2	DP/B	可编程时钟信号输出 <sup>1)</sup>
	CC2_CC21	O3/I	DP/B	<b>CAPCOM2 CC21IO</b> 捕获输入/比较输出
	A21	OH	DP/B	外部总线接口地址线 <b>21</b>
81	U0C1_DX1D	I	DP/B	<b>USIC0 通道 1</b> 移位时钟输入
	P4.7	O0/I	St/B	<b>P4</b> 口的位 <b>7</b> ，通用输入/输出
	CC2_CC31	O3/I	St/B	<b>CAPCOM2 CC31IO</b> 捕获输入/比较输出
	T4EUDA	I	St/B	<b>GPT12E</b> 定时器 <b>T4</b> 外部递增/递减控制输入
82	CCU61_CCP OS2A	I	St/B	<b>CCU61</b> 位置输入 <b>2</b>
	P2.9	O0/I	St/B	<b>P2</b> 口的位 <b>9</b> ，通用输入/输出
	U0C1_DOUT	O1	St/B	<b>USIC0 通道 1</b> 移位数据输出
	TxDC1	O2	St/B	<b>CAN</b> 节点 <b>1</b> 发送数据输出
	CC2_CC22	O3/I	St/B	<b>CAPCOM2 CC22IO</b> 捕获输入/比较输出
	A22	OH	St/B	外部总线接口地址线 <b>22</b>
	CLKIN1	I	St/B	时钟信号输入 <b>1</b>
83	TCK_A	I	St/B	<b>DAP0/JTAG</b> 时钟输入
	P0.2	O0/I	St/B	<b>P0</b> 口的位 <b>2</b> ，通用输入/输出
	U1C0_SCLK OUT	O1	St/B	<b>USIC1 通道 0</b> 移位时钟输出
	TxDC0	O2	St/B	<b>CAN</b> 节点 <b>0</b> 发送数据输出
84	CCU61_CC6 2	O3	St/B	<b>CCU61 通道 2</b> 输出

引脚	符号	控制	类型	功能
	A2	OH	St/B	外部总线接口地址线 2
	U1C0_DX1B	I	St/B	<b>USIC1 通道 0</b> 移位时钟输入
	CCU61_CC6 2INA	I	St/B	<b>CCU61 通道 2</b> 输入
84	P10.0	O0/I	St/B	<b>P10</b> 口的位 0, 通用输入/输出
	U0C1_DOUT	O1	St/B	<b>USIC0 通道 1</b> 移位数据输出
	CCU60_CC6 0	O2	St/B	<b>CCU60 通道 0</b> 输出
	AD0	OH/I	St/B	外部总线接口地址/数据线 0
	CCU60_CC6 0INA	I	St/B	<b>CCU60 通道 0</b> 输入
	ESR1_2	I	St/B	<b>ESR1</b> 触发输入 2
	U0C0_DX0A	I	St/B	<b>USIC0 通道 0</b> 移位数据输入
	U0C1_DX0A	I	St/B	<b>USIC0 通道 1</b> 移位数据输入
85	P3.0	O0/I	St/B	<b>P3</b> 口的位 0, 通用输入/输出
	U2C0_DOUT	O1	St/B	<b>USIC2 通道 0</b> 移位数据输出
	$\overline{\text{BREQ}}$	OH	St/B	外部总线请求输出
	ESR1_1	I	St/B	<b>ESR1</b> 触发输入 1
	U2C0_DX0A	I	St/B	<b>USIC2 通道 0</b> 移位数据输入
	RxDC3B	I	St/B	<b>CAN</b> 节点 3 接收数据输入
	U2C0_DX1A	I	St/B	<b>USIC2 通道 0</b> 移位时钟输入
86	P10.1	O0/I	St/B	<b>P10</b> 口的位 1, 通用输入/输出
	U0C0_DOUT	O1	St/B	<b>USIC0 通道 0</b> 移位数据输出
	CCU60_CC6 1	O2	St/B	<b>CCU60 通道 1</b> 输出
	AD1	OH/I	St/B	外部总线接口地址/数据线 1
	CCU60_CC6	I	St/B	<b>CCU60 通道 1</b> 输入

引脚	符号	控制	类型	功能
	11NA			
	U0C0_DX1A	I	St/B	<b>USIC0 通道 0 移位时钟输入</b>
	U0C0_DX0B	I	St/B	<b>USIC0 通道 0 移位数据输入</b>
87	P0.3	O0/I	St/B	<b>P0 口的位 3, 通用输入/输出</b>
	U1C0_SELO 0	O1	St/B	<b>USIC1 通道 0 选择/控制 0 输出</b>
	U1C1_SELO 1	O2	St/B	<b>USIC1 通道 1 选择/控制 1 输出</b>
	CCU61_COU T60	O3	St/B	<b>CCU61 通道 0 输出</b>
	A3	OH	St/B	<b>外部总线接口地址线 3</b>
	U1C0_DX2A	I	St/B	<b>USIC1 通道 0 移位控制输入</b>
	RxDC0B	I	St/B	<b>CAN 节点 0 接收数据输入</b>
88	P3.1	O0/I	St/B	<b>P3 口的位 1, 通用输入/输出</b>
	U2C0_DOUT	O1	St/B	<b>USIC2 通道 0 移位数据输出</b>
	TxDC3	O2	St/B	<b>CAN 节点 3 发送数据输出</b>
	$\overline{\text{HLDA}}$	OH/I	St/B	<b>外部总线保持应答输出/输入</b> 主控模式下输出；从控模式下输入。
	U2C0_DX0B	I	St/B	<b>USIC2 通道 0 移位数据输入</b>
89	P10.2	O0/I	St/B	<b>P10 口的位 2, 通用输入/输出</b>
	U0C0_SCLK OUT	O1	St/B	<b>USIC0 通道 0 移位时钟输出</b>
	CCU60_CC6 2	O2	St/B	<b>CCU60 通道 2 输出</b>
	U3C0_SELO 1	O3	St/B	<b>USIC3 通道 0 选择/控制 1 输出</b>
	AD2	OH/I	St/B	<b>外部总线接口地址/数据线 2</b>
	CCU60_CC6 2INA	I	St/B	<b>CCU60 通道 2 输入</b>

引脚	符号	控制	类型	功能
	U0C0_DX1B	I	St/B	<b>USIC0 通道 0</b> 移位时钟输入
	U3C0_DX2B	I	St/B	<b>USIC3 通道 0</b> 移位控制输入
90	P0.4	O0/I	St/B	<b>P0</b> 口的位 4，通用输入/输出
	U1C1_SELO 0	O1	St/B	<b>USIC1 通道 1</b> 选择/控制 0 输出
	U1C0_SELO 1	O2	St/B	<b>USIC1 通道 0</b> 选择/控制 1 输出
	CCU61_COU T61	O3	St/B	<b>CCU61 通道 1</b> 输出
	A4	OH	St/B	外部总线接口地址线 4
	U1C1_DX2A	I	St/B	<b>USIC1 通道 1</b> 移位控制输入
	RxDC1B	I	St/B	<b>CAN 节点 1</b> 接收数据输入
	ESR2_8	I	St/B	<b>ESR2 触发输入 8</b>
92	P2.13	O0/I	St/B	<b>P2</b> 口的位 13，通用输入/输出
	U2C1_SELO 2	O1	St/B	<b>USIC2 通道 1</b> 选择/控制 2 输出
	RxDC2D	I	St/B	<b>CAN 节点 2</b> 接收数据输入
93	P3.2	O0/I	St/B	<b>P3</b> 口的位 2，通用输入/输出
	U2C0_SCLK OUT	O1	St/B	<b>USIC2 通道 0</b> 移位时钟输出
	TxDC3	O2	St/B	<b>CAN 节点 3</b> 发送数据输出
	U2C0_DX1B	I	St/B	<b>USIC2 通道 0</b> 移位时钟输入
	$\overline{\text{HOLD}}$	I	St/B	外部总线主控方保持请求输入
94	P2.10	O0/I	St/B	<b>P2</b> 口的位 10，通用输入/输出
	U0C1_DOUT	O1	St/B	<b>USIC0 通道 1</b> 移位数据输出
	U0C0_SELO 3	O2	St/B	<b>USIC0 通道 0</b> 选择/控制 3 输出
	CC2_CC23	O3/I	St/B	<b>CAPCOM2 CC23IO</b> 捕获输入/比较输出

引脚	符号	控制	类型	功能
	A23	OH	St/B	外部总线接口地址线 <b>23</b>
	U0C1_DX0E	I	St/B	<b>USIC0</b> 通道 <b>1</b> 移位数据输入
	CAPINA	I	St/B	<b>GPT12E</b> 寄存器 <b>CAPREL</b> 捕获输入
	U3C1_DX0A	I	St/B	<b>USIC3</b> 通道 <b>1</b> 移位数据输入
95	P10.3	O0/I	St/B	<b>P10</b> 口的位 <b>3</b> ，通用输入/输出
	CCU60_COU T60	O2	St/B	<b>CCU60</b> 通道 <b>0</b> 输出
	AD3	OH/I	St/B	外部总线接口地址/数据线 <b>3</b>
	U0C0_DX2A	I	St/B	<b>USIC0</b> 通道 <b>0</b> 移位控制输入
	U0C1_DX2A	I	St/B	<b>USIC0</b> 通道 <b>1</b> 移位控制输入
	U3C0_DX0A	I	St/B	<b>USIC3</b> 通道 <b>0</b> 移位数据输入
96	P0.5	O0/I	St/B	<b>P0</b> 口的位 <b>5</b> ，通用输入/输出
	U1C1_SCLK OUT	O1	St/B	<b>USIC1</b> 通道 <b>1</b> 移位时钟输出
	U1C0_SELO 2	O3	St/B	<b>USIC1</b> 通道 <b>0</b> 选择/控制 <b>2</b> 输出
	CCU61_COU T62	O2	St/B	<b>CCU61</b> 通道 <b>2</b> 输出
	A5	OH/I	St/B	外部总线接口地址线 <b>5</b>
	U1C1_DX1A	I	St/B	<b>USIC1</b> 通道 <b>1</b> 移位时钟输入
	U1C0_DX1C	I	St/B	<b>USIC1</b> 通道 <b>0</b> 移位时钟输入
	RxDC3E	I	St/B	<b>CAN</b> 节点 <b>3</b> 接收数据输入
97	P3.3	O0/I	St/B	<b>P3</b> 口的位 <b>3</b> ，通用输入/输出
	U2C0_SELO 0	O1	St/B	<b>USIC2</b> 通道 <b>0</b> 选择/控制 <b>0</b> 输出
	U2C1_SELO 1	O2	St/B	<b>USIC2</b> 通道 <b>1</b> 选择/控制 <b>1</b> 输出
	U2C0_DX2A	I	St/B	<b>USIC2</b> 通道 <b>0</b> 移位控制输入

引脚	符号	控制	类型	功能
	RxDC3A	I	St/B	<b>CAN 节点 3 接收数据输入</b>
98	P10.4	O0/I	St/B	<b>P10 口的位 4, 通用输入/输出</b>
	U0C0_SELO 3	O1	St/B	<b>USIC0 通道 0 选择/控制 3 输出</b>
	CCU60_COU T61	O2	St/B	<b>CCU60 通道 1 输出</b>
	U3C0_DOUT	O3	St/B	<b>USIC3 通道 0 移位数据输入</b>
	AD4	OH/I	St/B	<b>外部总线接口地址/数据线 4</b>
	U0C0_DX2B	I	St/B	<b>USIC0 通道 0 移位控制输入</b>
	U0C1_DX2B	I	St/B	<b>USIC0 通道 1 移位控制输入</b>
	ESR1_9	I	St/B	<b>ESR1 触发输入 9</b>
99	P3.4	O0/I	St/B	<b>P3 口的位 4, 通用输入/输出</b>
	U2C1_SELO 0	O1	St/B	<b>USIC2 通道 1 选择/控制 0 输出</b>
	U2C0_SELO 1	O2	St/B	<b>USIC2 通道 0 选择/控制 1 输出</b>
	U0C0_SELO 4	O3	St/B	<b>USIC0 通道 0 选择/控制 4 输出</b>
	U2C1_DX2A	I	St/B	<b>USIC2 通道 1 移位控制输入</b>
	RxDC4A	I	St/B	<b>CAN 节点 4 接收数据输入</b>
100	P10.5	O0/I	St/B	<b>P10 口的位 5, 通用输入/输出</b>
	U0C1_SCLK OUT	O1	St/B	<b>USIC0 通道 1 移位时钟输出</b>
	CCU60_COU T62	O2	St/B	<b>CCU60 通道 2 输出</b>
	U2C0_DOUT	O3	St/B	<b>USIC2 通道 0 移位数据输出</b>
	AD5	OH/I	St/B	<b>外部总线接口地址/数据线 5</b>
	U0C1_DX1B	I	St/B	<b>USIC0 通道 1 移位时钟输入</b>

引脚	符号	控制	类型	功能
101	P3.5	O0/I	St/B	<b>P3</b> 口的位 5, 通用输入/输出
	U2C1_SCLK OUT	O1	St/B	<b>USIC2</b> 通道 1 移位时钟输出
	U2C0_SELO 2	O2	St/B	<b>USIC2</b> 通道 0 选择/控制 2 输出
	U0C0_SELO 5	O3	St/B	<b>USIC0</b> 通道 0 选择/控制 5 输出
	U2C1_DX1A	I	St/B	<b>USIC2</b> 通道 1 移位时钟输入
102	P0.6	O0/I	St/B	<b>P0</b> 口的位 6, 通用输入/输出
	U1C1_DOUT	O1	St/B	<b>USIC1</b> 通道 1 移位数据输出
	TxDC1	O2	St/B	<b>CAN</b> 节点 1 发送数据输出
	CCU61_COU T63	O3	St/B	<b>CCU61</b> 通道 3 输出
	A6	OH	St/B	外部总线接口地址线 6
	U1C1_DX0A	I	St/B	<b>USIC1</b> 通道 1 移位数据输入
	CCU61_CTR APA	I	St/B	<b>CCU61</b> 急停强制中断输入
	U1C1_DX1B	I	St/B	<b>USIC1</b> 通道 1 移位时钟输入
103	P10.6	O0/I	St/B	<b>P10</b> 口的位 6, 通用输入/输出
	U0C0_DOUT	O1	St/B	<b>USIC0</b> 通道 0 移位数据输出
	TxDC4	O2	St/B	<b>CAN</b> 节点 4 发送数据输出
	U1C0_SELO 0	O3	St/B	<b>USIC1</b> 通道 0 选择/控制 0 输出
	AD6	OH/I	St/B	外部总线接口地址/数据线 6
	U0C0_DX0C	I	St/B	<b>USIC0</b> 通道 0 移位数据输入
	U1C0_DX2D	I	St/B	<b>USIC1</b> 通道 0 移位控制输入
	CCU60_CTR APA	I	St/B	<b>CCU60</b> 急停强制中断输入



引脚	符号	控制	类型	功能
104	P3.6	O0/I	St/B	<b>P3</b> 口的位 6, 通用输入/输出
	U2C1_DOUT	O1	St/B	<b>USIC2</b> 通道 1 移位数据输出
	TxDC4	O2	St/B	<b>CAN</b> 节点 4 发送数据输出
	U0C0_SELO 6	O3	St/B	<b>USIC0</b> 通道 0 选择/控制 6 输出
	U2C1_DX0A	I	St/B	<b>USIC2</b> 通道 1 移位数据输入
	U2C1_DX1B	I	St/B	<b>USIC2</b> 通道 1 移位时钟输入
105	P10.7	O0/I	St/B	<b>P10</b> 口的位 7, 通用输入/输出
	U0C1_DOUT	O1	St/B	<b>USIC0</b> 通道 1 移位数据输出
	CCU60_COU T63	O2	St/B	<b>CCU60</b> 通道 3 输出
	AD7	OH/I	St/B	外部总线接口地址/数据线 7
	U0C1_DX0B	I	St/B	<b>USIC0</b> 通道 1 移位数据输入
	CCU60_CCP OS0A	I	St/B	<b>CCU60</b> 位置输入 0
	RxDC4C	I	St/B	<b>CAN</b> 节点 4 接收数据输入
	T4INB	I	St/B	<b>GPT12E</b> 定时器 T4 计数/门控输入
106	P0.7	O0/I	St/B	<b>P0</b> 口的位 7, 通用输入/输出
	U1C1_DOUT	O1	St/B	<b>USIC1</b> 通道 1 移位数据输出
	U1C0_SELO 3	O2	St/B	<b>USIC1</b> 通道 0 选择/控制 3 输出
	TxDC3	O3	St/B	<b>CAN</b> 节点 3 发送数据输出
	A7	OH	St/B	外部总线接口地址线 7
	U1C1_DX0B	I	St/B	<b>USIC1</b> 通道 1 移位数据输入
	CCU61_CTR APB	I	St/B	<b>CCU61</b> 急停强制中断输入
107	P3.7	O0/I	St/B	<b>P3</b> 口的位 7, 通用输入/输出

引脚	符号	控制	类型	功能
	U2C1_DOUT	O1	St/B	<b>USIC2 通道 1 移位数据输出</b>
	U2C0_SELO 3	O2	St/B	<b>USIC2 通道 0 选择/控制 3 输出</b>
	U0C0_SELO 7	O3	St/B	<b>USIC0 通道 0 选择/控制 7 输出</b>
	U2C1_DX0B	I	St/B	<b>USIC2 通道 1 移位数据输入</b>
111	P1.0	O0/I	St/B	<b>P1 口的位 0，通用输入/输出</b>
	U1C0_MCLK OUT	O1	St/B	<b>USIC1 通道 0 主控时钟输出</b>
	U1C0_SELO 4	O2	St/B	<b>USIC1 通道 0 选择/控制 4 输出</b>
	A8	OH	St/B	<b>外部总线接口地址线 8</b>
	ESR1_3	I	St/B	<b>ESR1 触发输入 3</b>
	CCU62_CTR APB	I	St/B	<b>CCU62 急停强制中断输入</b>
	T6INB	I	St/B	<b>GPT12E 定时器 T6 计数/门控输入</b>
112	P9.0	O0/I	St/B	<b>P9 口的位 0，通用输入/输出</b>
	CCU63_CC6 0	O1	St/B	<b>CCU63 通道 0 输出</b>
	CCU63_CC6 0INA	I	St/B	<b>CCU63 通道 0 输入</b>
	T6EADB	I	St/B	<b>GPT12E 定时器 T6 外部递增/递减控制 输入</b>
113	P10.8	O0/I	St/B	<b>P10 口的位 8，通用输入/输出</b>
	U0C0_MCLK OUT	O1	St/B	<b>USIC0 通道 0 主控时钟输出</b>
	U0C1_SELO 0	O2	St/B	<b>USIC0 通道 1 选择/控制 0 输出</b>
	U2C1_DOUT	O3	St/B	<b>USIC2 通道 1 移位数据输出</b>
	AD8	OH/I	St/B	<b>外部总线接口地址/数据线 8</b>

引脚	符号	控制	类型	功能
	CCU60_CCP OS1A	I	St/B	<b>CCU60 位置输入 1</b>
	U0C0_DX1C	I	St/B	<b>USIC0 通道 0 移位时钟输入</b>
	$\overline{\text{BRKIN\_B}}$	I	St/B	<b>OCDS 断点信号输入</b>
	T3EUIDB	I	St/B	<b>GPT12E 定时器 T3 外部递增/递减控制输入</b>
114	P9.1	O0/I	St/B	<b>P9 口的位 1, 通用输入/输出</b>
	CCU63_CC6 1	O1	St/B	<b>CCU63 通道 1 输出</b>
	CCU63_CC6 1INA	I	St/B	<b>CCU63 通道 1 输入</b>
115	P10.9	O0/I	St/B	<b>P10 口的位 9, 通用输入/输出</b>
	U0C0_SELO 4	O1	St/B	<b>USIC0 通道 0 选择/控制 4 输出</b>
	U0C1_MCLK OUT	O2	St/B	<b>USIC0 通道 1 主控时钟输出</b>
	AD9	OH/I	St/B	<b>外部总线接口地址/数据线 9</b>
	CCU60_CCP OS2A	I	St/B	<b>CCU60 位置输入 2</b>
	TCK_B	I	St/B	<b>DAP0/JTAG 时钟输入</b>
	T3INB	I	St/B	<b>GPT12E 定时器 T3 计数/门控输入</b>
116	P1.1	O0/I	St/B	<b>P1 口的位 1, 通用输入/输出</b>
	CCU62_COU T62	O1	St/B	<b>CCU62 通道 2 输出</b>
	U1C0_SELO 5	O2	St/B	<b>USIC1 通道 0 选择/控制 5 输出</b>
	U2C1_DOUT	O3	St/B	<b>USIC2 通道 1 移位数据输出</b>
	A9	OH	St/B	<b>外部总线接口地址线 9</b>
	ESR2_3	I	St/B	<b>ESR2 触发输入 3</b>

引脚	符号	控制	类型	功能
	U2C1_DX0C	I	St/B	<b>USIC2 通道 1 移位数据输入</b>
117	P10.10	O0/I	St/B	<b>P10 口的位 10，通用输入/输出</b>
	U0C0_SELO 0	O1	St/B	<b>USIC0 通道 0 选择/控制 0 输出</b>
	CCU60_COU T63	O2	St/B	<b>CCU60 通道 3 输出</b>
	AD10	OH/I	St/B	<b>外部总线接口地址/数据线 10</b>
	U0C0_DX2C	I	St/B	<b>USIC0 通道 0 移位控制输入</b>
	U0C1_DX1A	I	St/B	<b>USIC0 通道 1 移位时钟输入</b>
	TDI_B	I	St/B	<b>JTAG 测试数据输入</b>
118	P10.11	O0/I	St/B	<b>P10 口的位 11，通用输入/输出</b>
	U1C0_SCLK OUT	O1	St/B	<b>USIC1 通道 0 移位时钟输出</b>
	$\overline{\text{BRKOUT}}$	O2	St/B	<b>OCDS 断点信号输出</b>
	U3C0_SELO 0	O3	St/B	<b>USIC3 通道 0 选择/控制 0 输出</b>
	AD11	OH/I	St/B	<b>外部总线接口地址/数据线 11</b>
	U1C0_DX1D	I	St/B	<b>USIC1 通道 0 移位时钟输入</b>
	RxDC2B	I	St/B	<b>CAN 节点 2 接收数据输入</b>
	TMS_B	I	St/B	<b>JTAG 测试模式选择输入</b>
	U3C0_DX2A	I	St/B	<b>USIC3 通道 0 移位控制输入</b>
119	P9.2	O0/I	St/B	<b>P9 口的位 2，通用输入/输出</b>
	CCU63_CC6 2	O1	St/B	<b>CCU63 通道 2 输出</b>
	CCU63_CC6 2INA	I	St/B	<b>CCU63 通道 2 输入</b>
	CAPINB	I	St/B	<b>GPT12E 寄存器 CAPREL 捕获输入</b>
120	P1.2	O0/I	St/B	<b>P1 口的位 2，通用输入/输出</b>

引脚	符号	控制	类型	功能
	CCU62_CC62	O1	St/B	<b>CCU62 通道 2 输出</b>
	U1C0_SELO6	O2	St/B	<b>USIC1 通道 0 选择/控制 6 输出</b>
	U2C1_SCLKOUT	O3	St/B	<b>USIC2 通道 1 移位时钟输出</b>
	A10	OH	St/B	<b>外部总线接口地址线 10</b>
	ESR1_4	I	St/B	<b>ESR1 触发输入 4</b>
	CCU61_T12H RB	I	St/B	<b>CCU61 T12 的外部运行控制输入</b>
	CCU62_CC62INA	I	St/B	<b>CCU62 通道 2 输入</b>
	U2C1_DX0D	I	St/B	<b>USIC2 通道 1 移位数据输入</b>
	U2C1_DX1C	I	St/B	<b>USIC2 通道 1 移位时钟输入</b>
121	P10.12	O0/I	St/B	<b>P10 口的位 12，通用输入/输出</b>
	U1C0_DOUT	O1	St/B	<b>USIC1 通道 0 移位数据输出</b>
	TxDC2	O2	St/B	<b>CAN 节点 2 发送数据输出</b>
	TDO_B	OH/I	St/B	<b>JTAG 测试数据输出/DAP1 输入/输出</b>
	AD12	OH/I	St/B	<b>外部总线接口地址/数据线 12</b>
	U1C0_DX0C	I	St/B	<b>USIC1 通道 0 移位数据输入</b>
	U1C0_DX1E	I	St/B	<b>USIC1 通道 0 移位时钟输入</b>
122	P9.3	O0/I	St/B	<b>P9 口的位 3，通用输入/输出</b>
	CCU63_COUT60	O1	St/B	<b>CCU63 通道 0 输出</b>
	$\overline{\text{BRKOUT}}$	O2	St/B	<b>OCDS 断点信号输出</b>
123	P10.13	O0/I	St/B	<b>P10 口的位 13，通用输入/输出</b>
	U1C0_DOUT	O1	St/B	<b>USIC1 通道 0 移位数据输出</b>

引脚	符号	控制	类型	功能
	TxDC3	O2	St/B	<b>CAN 节点 3 发送数据输出</b>
	U1C0_SELO 3	O3	St/B	<b>USIC1 通道 0 选择/控制 3 输出</b>
	$\overline{WR}$ / $\overline{WRL}$	OH	St/B	外部总线接口写选通输出 当 $\overline{WR}$ ，每次外部写访问有效 当 $\overline{WRL}$ ，每次外部写访问低字节有效
	U1C0_DX0D	I	St/B	<b>USIC1 通道 0 移位数据输入</b>
124	P1.3	O0/I	St/B	<b>P1 口的位 3，通用输入/输出</b>
	CCU62_COU T63	O1	St/B	<b>CCU62 通道 3 输出</b>
	U1C0_SELO 7	O2	St/B	<b>USIC1 通道 0 选择/控制 7 输出</b>
	U2C0_SELO 4	O3	St/B	<b>USIC2 通道 0 选择/控制 4 输出</b>
	A11	OH	St/B	外部总线接口地址线 11
	ESR2_4	I	St/B	<b>ESR2 触发输入 4</b>
	CCU62_T12H RB	I	St/B	<b>CCU62 T12 的外部运行控制输入</b>
125	P9.4	O0/I	St/B	<b>P9 口的位 4，通用输入/输出</b>
	CCU63_COU T61	O1	St/B	<b>CCU63 通道 1 输出</b>
	U2C0_DOUT	O2	St/B	<b>USIC2 通道 0 移位数据输出</b>
	CCU62_COU T63	O3	St/B	<b>CCU62 通道 3 输出</b>
126	P9.5	O0/I	St/B	<b>P9 口的位 5，通用输入/输出</b>
	CCU63_COU T62	O1	St/B	<b>CCU63 通道 2 输出</b>
	U2C0_DOUT	O2	St/B	<b>USIC2 通道 0 移位数据输出</b>
	CCU62_COU	O3	St/B	<b>CCU62 通道 2 输出</b>

引脚	符号	控制	类型	功能
	T62			
	U2C0_DX0E	I	St/B	<b>USIC2 通道 0</b> 移位数据输入
	CCU60_CCP OS2B	I	St/B	<b>CCU60</b> 位置输入 2
128	P10.14	O0/I	St/B	<b>P10</b> 口的位 14, 通用输入/输出
	U1C0_SELO 1	O1	St/B	<b>USIC1 通道 0</b> 选择/控制 1 输出
	U0C1_DOUT	O2	St/B	<b>USIC0 通道 1</b> 移位数据输出
	U3C0_SCLK OUT	O3	St/B	<b>USIC3 通道 0</b> 移位时钟输出
	$\overline{RD}$	OH	St/B	外部总线接口读选通输出
	ESR2_2	I	St/B	<b>ESR2</b> 触发输入 2
	U0C1_DX0C	I	St/B	<b>USIC0 通道 1</b> 移位数据输入
	RxDC3C	I	St/B	<b>CAN</b> 节点 3 接收数据输入
	U3C0_DX1A	I	St/B	<b>USIC3 通道 0</b> 移位时钟输入
129	P1.4	O0/I	St/B	<b>P1</b> 口的位 4, 通用输入/输出
	CCU62_COU T61	O1	St/B	<b>CCU62</b> 通道 1 输出
	U1C1_SELO 4	O2	St/B	<b>USIC1 通道 1</b> 选择/控制 4 输出
	U2C0_SELO 5	O3	St/B	<b>USIC2 通道 0</b> 选择/控制 5 输出
	A12	OH	St/B	外部总线接口地址线 12
	U2C0_DX2B	I	St/B	<b>USIC2 通道 0</b> 移位控制输入
	RxDC5A	I	St/B	<b>CAN</b> 节点 5 接收数据输入
130	P10.15	O0/I	St/B	<b>P10</b> 口的位 15, 通用输入/输出
	U1C0_SELO 2	O1	St/B	<b>USIC1 通道 0</b> 选择/控制 2 输出

引脚	符号	控制	类型	功能
	U0C1_DOUT	O2	St/B	<b>USIC0 通道 1 移位数据输出</b>
	U1C0_DOUT	O3	St/B	<b>USIC1 通道 0 移位数据输出</b>
	ALE	OH	St/B	外部总线接口地址锁存使能输出
	U0C1_DX1C	I	St/B	<b>USIC0 通道 1 移位时钟输入</b>
131	P1.5	O0/I	St/B	<b>P1 口的位 5，通用输入/输出</b>
	CCU62_COU T60	O1	St/B	<b>CCU62 通道 0 输出</b>
	U1C1_SELO 3	O2	St/B	<b>USIC1 通道 1 选择/控制 3 输出</b>
	BRKOUT	O3	St/B	<b>OCDS 断点信号输出</b>
	A13	OH	St/B	外部总线接口地址线 13
	U2C0_DX0C	I	St/B	<b>USIC2 通道 0 移位数据输入</b>
132	P9.6	O0/I	St/B	<b>P9 口的位 6，通用输入/输出</b>
	CCU63_COU T63	O1	St/B	<b>CCU63 通道 3 输出</b>
	CCU63_COU T62	O2	St/B	<b>CCU63 通道 2 输出</b>
	CCU62_COU T61	O3	St/B	<b>CCU62 通道 1 输出</b>
	CCU63_CTR APA	I	St/B	<b>CCU63 急停强制中断输入</b>
	CCU60_CCP OS1B	I	St/B	<b>CCU60 位置输入 1</b>
133	P1.6	O0/I	St/B	<b>P1 口的位 6，通用输入/输出</b>
	CCU62_CC6 1	O0/I	St/B	<b>CCU62 通道 1 输出</b>
	U1C1_SELO 2	O2	St/B	<b>USIC1 通道 1 选择/控制 2 输出</b>
	U2C0_DOUT	O3	St/B	<b>USIC2 通道 0 移位数据输出</b>



引脚	符号	控制	类型	功能
	A14	OH	St/B	外部总线接口地址线 14
	U2C0_DX0D	I	St/B	<b>USIC2 通道 0</b> 移位数据输入
	CCU62_CC61INA	I	St/B	<b>CCU62 通道 1</b> 输入
134	P9.7	O0/I	St/B	<b>P9</b> 口的位 7，通用输入/输出
	CCU62_COUT60	O1	St/B	<b>CCU62 通道 0</b> 输出
	CCU62_COUT63	O2	St/B	<b>CCU62 通道 3</b> 输出
	CCU63_CTRAPB	I	St/B	<b>CCU63</b> 急停强制中断输入
	U2C0_DX1D	I	St/B	<b>USIC2 通道 0</b> 移位时钟输入
	CCU60_CCP0S0B	I	St/B	<b>CCU60</b> 位置输入 0
135	P1.7	O0/I	St/B	<b>P1</b> 口的位 7，通用输入/输出
	CCU62_CC60	O1	St/B	<b>CCU62 通道 0</b> 输出
	U1C1_MCLKOUT	O2	St/B	<b>USIC1 通道 1</b> 主控时钟输出
	U2C0_SCLKOUT	O3	St/B	<b>USIC2 通道 0</b> 移位时钟输出
	A15	OH	St/B	外部总线接口地址线 15
	U2C0_DX1C	I	St/B	<b>USIC2 通道 0</b> 移位时钟输入
	CCU62_CC60INA	I	St/B	<b>CCU62 通道 0</b> 输入
	RxDC4E	I	St/B	<b>CAN</b> 节点 4 接收数据输入
136	XTAL2	O	Sp/M	晶振放大器输出
137	XTAL1	I	Sp/M	晶振放大器输入 由外部时钟源提供器件的工作时钟时，驱动 XTAL1，XTAL2 断开。

引脚	符号	控制	类型	功能
				XTAL1 上的电压必须和内核供电电压 V <sub>DDIM</sub> 相匹配。
	ESR2_9	I	St/B	<b>ESR2 触发输入 9</b>
138	$\overline{\text{PORST}}$	I	In/B	<p>上电复位输入</p> <p>该引脚上的低电平复位 XE16xyM。毛刺滤波器会抑制 &lt;10 ns 的输入脉冲。大于 100ns 的输入脉冲可安全通过滤波器。可确保被识别的最小脉冲宽度应为 120ns。</p> <p>该引脚不被驱动时，内部上拉器件将其拉高。</p>
139	$\overline{\text{ESR1}}$	O0/I	St/B	<b>外部服务请求 1</b>
	RxDC0E	I	St/B	<b>CAN 节点 0 接收数据输入</b>
	U1C0_DX0F	I	St/B	<b>USIC1 通道 0 移位数据输入</b>
	U1C0_DX2C	I	St/B	<b>USIC1 通道 0 移位控制输入</b>
	U1C1_DX0C	I	St/B	<b>USIC1 通道 1 移位数据输入</b>
	U1C1_DX2B	I	St/B	<b>USIC1 通道 1 移位控制输入</b>
	U2C1_DX2C	I	St/B	<b>USIC2 通道 1 移位控制输入</b>
140	$\overline{\text{ESR2}}$	O0/I	St/B	<b>外部服务请求 2</b>
	RxDC1E	I	St/B	<b>CAN 节点 1 接收数据输入</b>
	CCU60_CTR APC	I	St/B	<b>CCU60 急停强制中断输入</b>
	CCU61_CTR APC	I	St/B	<b>CCU61 急停强制中断输入</b>
	CCU62_CTR APC	I	St/B	<b>CCU62 急停强制中断输入</b>
	CCU63_CTR APC	I	St/B	<b>CCU63 急停强制中断输入</b>
	U1C1_DX0D	I	St/B	<b>USIC1 通道 1 移位数据输入</b>

引脚	符号	控制	类型	功能
	U1C1_DX2C	I	St/B	<b>USIC1 通道 1 移位控制输入</b>
	U2C1_DX0E	I	St/B	<b>USIC2 通道 1 移位数据输入</b>
	U2C1_DX2B	I	St/B	<b>USIC2 通道 1 移位控制输入</b>
141	$\overline{\text{ESR0}}$	O0/I	St/B	<b>外部服务请求 0</b> <i>注：上电后，ESR0 用作带有弱上拉的漏极开路双向复位引脚。</i>
	U1C0_DX0E	I	St/B	<b>USIC1 通道 0 移位数据输入</b>
	U1C0_DX2B	I	St/B	<b>USIC1 通道 0 移位控制输入</b>
142	P8.6	O0/I	St/B	<b>P8 口的位 6，通用输入/输出</b>
	CCU60_COU T63	O1	St/B	<b>CCU60 通道 3 输出</b>
	MCHK_MAT CH	O3	St/B	<b>存储器检测器匹配输出</b>
	CCU60_CTR APB	I	St/B	<b>CCU60 急停强制中断输入</b>
	$\overline{\text{BRKIN\_D}}$	I	St/B	<b>OCDS 断点信号输入</b>
	CCU62_CTR APD	I	St/B	<b>CCU62 急停强制中断输入</b>
143	P8.5	O0/I	St/B	<b>P8 口的位 5，通用输入/输出</b>
	CCU60_COU T62	O1	St/B	<b>CCU60 通道 2 输出</b>
	CCU62_CC6 2	O2	St/B	<b>CCU62 通道 2 输出</b>
	TCK_D	I	St/B	<b>DAP0/JTAG 时钟输入</b>
	CCU62_CC6 2INB	I	St/B	<b>CCU62 通道 2 输入</b>
15	V <sub>DDIM</sub>	-	PS/M	<b>电源域 M 的数字内核电源</b> 使用陶瓷电容去耦，详见数据手册。
54,	V <sub>DDH1</sub>	-	PS/1	<b>电源域 1 的数字内核电源</b>

引脚	符号	控制	类型	功能
91, 127				使用陶瓷电容去耦, 详见数据手册。 所有 $V_{DDI1}$ 引脚必须互连。
20	$V_{DDPA}$	-	PS/A	<b>电源域 A 的数字引出端电源</b> 连接相邻引脚对 $V_{DDP}/V_{SS}$ 的去耦电容尽可能靠近这些引脚。 <i>注: A/D 转换器、P5、P6 和 P15 口由电源 <math>V_{DDPA}</math> 供电。</i>
2, 36, 38, 72, 74, 108, 110, 144	$V_{DDPB}$	-	PS/B	<b>电源域 B 的数字引出端电源</b> 连接相邻引脚对 $V_{DDP}/V_{SS}$ 的去耦电容尽可能靠近这些引脚。 <i>注: 片内电压调节器和除 P5、P6 和 P15 之外的所有端口由电源 <math>V_{DDPB}</math> 供电。</i>
1, 37, 73, 109	$V_{SS}$	-	PS/--	<b>数字地</b> 所有 $V_{SS}$ 必须和地线相连。 <i>注: 裸引出端同样内部连接到 <math>V_{SS}</math>。为了改善 EMC, 建议将裸引出端和板级电源地相连。</i> <i>有关散热的描述, 请参阅数据手册。版图设计示例参见应用笔记。</i>

- 1) 若要产生参考时钟输出用于测量总线时序, 必须选择  $f_{SYS}$  作为 EXTCLK 的时钟源, P2.8 必须用作输出引脚。此外, 必须使能高速时钟引出端。该配置用于参考时钟输出信号 CLKOUT。

## 10 专用引脚

XE16xyM 中大多数功能模块的输入/输出或控制信号由并行端口引脚的复用功能来实现。不过也有一些信号需要使用专用引脚，如振荡器、特殊控制信号以及电源。

**表 10-1** 列出 XE16xyM 中的专用引脚。

**表10-1 XE16xyM 专用引脚**

引脚	功能
$\overline{\text{PORST}}$	上电复位输入
$\overline{\text{ESR0}}$	外部服务请求输入 0
$\overline{\text{ESR1}}$	外部服务请求输入 1
$\overline{\text{ESR2}}$	外部服务请求输入 2
XTAL1, XTAL2	振荡器输入/输出（主振荡器）
$\overline{\text{TESTM}}$	测试模式使能
$\overline{\text{TRST}}$	测试系统复位输入
$V_{\text{AREF}x}, V_{\text{AGND}}$	ADC 参考电压
$V_{\text{DDIM}}$	电源域 M 的数字内核电源（1 个引脚）
$V_{\text{DDI1}}$	电源域 1 的数字内核电源（3 个引脚）
$V_{\text{DDPA}}$	电源域 A（包括 ADC 在内）的数字引出端电源（1 个引脚）
$V_{\text{DDPB}}$	电源域 B 的数字引出端电源（8 个引脚）
$V_{\text{SS}}$	数字地（4 个引脚）

**上电复位输入  $\overline{\text{PORST}}$** ：在上电或外部事件（如硬件故障或手动复位）的情况下，该输入引脚可使得 XE16xyM 进入一个明确定义好的复位状态中。

**外部服务请求输入  $\overline{\text{ESR0}}$ ， $\overline{\text{ESR1}}$ ， $\overline{\text{ESR2}}$** ：这三个引脚可用作各种与系统相关的功能：

- 通过外部信号（如电源失效信号）触发中断或强制中断（A 类或 B 类）请求
- 产生唤醒请求信号

- 产生硬件复位请求 ( $\overline{\text{ESR0}}$  缺省为双向信号, 可选择  $\overline{\text{ESR1}}$  和  $\overline{\text{ESR2}}$  输出复位信号)
- CCU6x、MultiCAN 和 USIC 的数据/控制输入 ( $\overline{\text{ESR1}}$  或  $\overline{\text{ESR2}}$ )
- 软件控制的输入/输出信号

**振荡器输入 XTAL1 和输出 XTAL2** 将内部主振荡器连接到外部晶振。主振荡器由一个反相器和一个反馈元件组成。标准外部振荡电路由晶振、两个小终端电容和用于限制晶振电流的串联电阻组成。主振荡器可为 XE16xyM 产生高精度工作时钟信号。

当使用外部时钟信号输入时, 由 XTAL1 端输入, 此时 XTAL2 端断开。通过读取状态标志可以知道 XTAL1 的当前逻辑状态。因此, 当 XTAL1 既没有用作振荡器接口, 也没有用作时钟输入时可以用作数字输入。

*注: 引脚 XTAL1 属于内核电源域 DMP\_M。因此, 所有输入信号都必须在该内核电压范围内。*

**测试模式输入 TESTM** 使 XE16xyM 进入到测试模式, 该模式用于器件的生产测试过程。测试模式下, XE16xyM 的系统行为与正常操作不同。因此, 在应用系统正常操作过程中, 引脚 TESTM 必须保持高电平 (与  $V_{\text{DDPB}}$  相连)。

**测试复位输入 TRST** 使 XE16xyM 的调试系统进入复位状态。正常工作期间, 该输入应保持低电平。 $\overline{\text{PORST}}$  上升沿时, 通过将 TRST 引脚驱动为高电平, 使得片上调试系统能够进行调试。

**模数转换器电源引脚  $V_{\text{AREFX}}$  和  $V_{\text{AGND}}$**  为片上 ADC 模块提供独立参考电压。当  $V_{\text{AREFX}}$  和  $V_{\text{AGND}}$  与  $V_{\text{DD}}$  和  $V_{\text{SS}}$  分开之后, 可降低数字逻辑部分对模拟输入信号的耦合噪声, 从而改善转换结果的稳定性。此外, 由于转换结果的产生和参考电压有关, 因此易于实现比例转换。

*注: 每个模块的通道 0 还能用作备用参考电压输入端口。*

**内核电源引脚  $V_{\text{DDIM}}/V_{\text{DDH}}$**  有两种用途: 当片上 EVVR 为 XE16xyM 的内核逻辑供电时, 这些引脚将 EVVR 连接至它们的外部缓冲电容。对于外部电源而言, 内核电压由这些引脚提供。去耦电容应该紧靠  $V_{\text{DDI}}/V_{\text{SS}}$  引脚。可使用陶瓷电容, 在相应的数据手册中可查到推荐值。

**电源电压引脚  $V_{\text{DDPA}}/V_{\text{DDPB}}$**  为 XE16xyM 的所有模拟和数字逻辑提供电源。每个电源域 (DMP\_A 和 DMP\_B) 都能提供在特定电源电压范围内的任意电压 (请参考相关的数据手册)。除了外部内核电源, 这些引脚为输出驱动器、片上 EVVR ( $V_{\text{DDPB}}$ ) 供电。去耦电容应该紧靠  $V_{\text{DDP}}/V_{\text{SS}}$  引脚。

**参考地引脚  $V_{\text{SS}}$**  为电源以及输入信号的参考电压提供参考地电压。

*注: 所有  $V_{\text{DDx}}$  和  $V_{\text{SS}}$  引脚必须分别连接到相应的电源和地。*

## 11 外部总线控制器 (EBC)

C166SV2 CPU 通过 EBC 访问片外和片内外设和存储器, 可进行取指或数据交换。当 EBC 用于和片外器件连接时, 也被称作 EXTBUS; 若用于和片内 (本地) 元件接口, 也被称作 LXBUS。

### 11.1 特性归纳

EBC 的功能和时序特性可灵活配置, 从而可满足不同应用的需求。

- 非复用和复用模式
- 高达 24 条地址线
- 8 位或 16 位数据总线
- 同步和异步 Ready
- 支持外部总线仲裁
- 高达 8 路总线通道
- 最多 7 路通道的地址窗口可编程
- 每路通道的总线时序和功能可编程

### 11.2 概述

EBC 的功能和时序特性通过一组配置寄存器控制。

EBC 的基本功能和一般行为由模式选择寄存器 EBCMOD0 和 EBCMOD1 设置。

EBC 支持高达 8 路 ( $x = 0-7$ ) 总线通道, 每路对应一个片选信号 ( $\overline{CSx}$ )。每路通道由一组专用寄存器配置。FCONCSx 寄存器设定其功能特性; TCONCSx 寄存器设定其周期时序。其中 7 路通道的地址范围可通过寄存器 ADDRSEL (1...7) 来设置。其余未覆盖的地址区分配给  $\overline{CS0}$ 。

使用外部  $\overline{CSx}$  信号可节省用于片选的外部逻辑电路。通过特殊的 READY 功能, 可对非确定时序的外部器件进行访问。总线仲裁可采用  $\overline{HOLD}/\overline{HLDA}$  协议。

外部总线时序和参考时钟输出 (CLKOUT) 相关。所有总线信号在该时钟的上升沿产生。外部总线协议与 C166 系列兼容, 不过, 外部总线时序在等待状态间隔和信号灵活性方面有所改进。

### 11.3 命名规则

在描述 EBC 时序和功能时, 将会用到以下总线信号名。

控制信号:

- ALE – 地址锁存使能 (高有效)。指示地址有效。

- $\overline{\text{CS}}$  – 片选。
- $\overline{\text{WR}} / \overline{\text{WRL}}$  – 写选通，写低字节选通（低有效）。配置为一般的写请求或低字节写请求。
- $\overline{\text{BHE}} / \overline{\text{WRH}}$  – 高字节使能，写高字节选通（低有效）。配置为高字节使能或高字节写请求。
- $\overline{\text{RD}}$  – 读选通（低有效）。
- $\text{READY}$  – 操作结束指示（极性可编程）。
- $\overline{\text{HOLD}}$  – 外部总线请求的保持输入（低有效）。
- $\overline{\text{HLDA}}$  – 保持应答（低有效），主控方输出以授权总线。
- $\text{BREQ}$  – 总线请求（低有效）。

总线信号：

- $\text{A}$  或  $\text{ADDR}$  – 地址总线。
- $\text{D}$  或  $\text{DATA}$  – 数据总线。
- $\text{AD}$  – 共用的数据/地址[15:0]总线。

## 11.4 时序描述

总线特性可设置为以下访问模式：

- 16 位数据总线，地址非复用
- 16 位数据总线，地址复用
- 8 位数据总线，地址非复用
- 8 位数据总线，地址复用

复用模式意味着数据总线被地址（仅低 16 位）和数据分时复用。在非复用模式下，数据总线仅用于数据传送，必须使用额外的地址总线。

### 11.4.1 总线节拍

外部总线时序细分为 6 个不同时序节拍（A-F）。这些节拍会影响访问总线器件所需的控制信号。

当一个节拍开始时，在给定的输出延迟内输出信号改变。延迟时间的详细信息请参见 XE16xyM 的数据手册。输出延迟过后，控制输出信号在该节拍内保持稳定。每个节拍可占用多个 CLKOUT 周期，周期个数可编程设定。



### A 拍 – $\overline{CS}$ 改变

A 拍占用 0 - 3 个时钟周期，将来自前一周期的数据总线驱动器设置成高阻态（片选信号切换后的高阻等待周期）。

并非在每个访问周期都会插入 A 节拍，只有改变  $\overline{CS}$  时才插入。如果一次访问使用的  $\overline{CS}$  ( $\overline{CS_x}$ ) 和下一次访问使用的  $\overline{CS}$  ( $\overline{CS_y}$ ) 不同，则根据第一次  $\overline{CS}$  ( $\overline{CS_x}$ ) 设置的控制位来执行 A 拍的周期。该特性可用于优化某些器件的等待周期，这些器件的数据总线驱动器有较长的关闭延迟，如 EPROM 和 Flash 存储器。

在下一个周期的地址和 ALE 有效后，插入节拍 A。

如果在两次访问之间存在一些空闲周期，则应将这些周期考虑在内，A 拍相应被缩短。例如，如果设置 3 个高阻态周期，而出现了 2 个空闲周期，则 A 拍只占用一个时钟周期。

### B 拍 – 地址建立/ALE 有效时间长度

B 拍占用 1 - 2 个时钟周期，在命令之前对器件寻址，并定义 ALE 有效的时间长度。在复用总线模式下，地址被锁存。

### C 拍 – 延迟

C 拍与 A、B 拍相似，只是 ALE 已经变低。C 拍占用 0 - 3 个时钟周期。在复用总线模式，地址被保持，以保证被安全锁存。C 拍的周期可用于延迟命令信号 (RW 延迟)。

### D 拍 – 写数据建立/复用模式的高阻态

D 拍占用 0 - 1 个时钟周期。当执行读周期时，将复用总线上的地址设置为高阻态。对于所有写周期，保证在命令执行之前总线上的数据已有效。

### E 拍 – $\overline{RD}/\overline{WR}$ 命令

E 拍为命令或访问节拍，占用 1 - 32 个时钟周期。在该节拍中，读取数据，写数据送至总线，命令信号有效。该节拍结束时，读取的数据被锁存。

READY 功能拉长了该节拍。受 READY 控制的访问周期可能具有无限的周期时间。

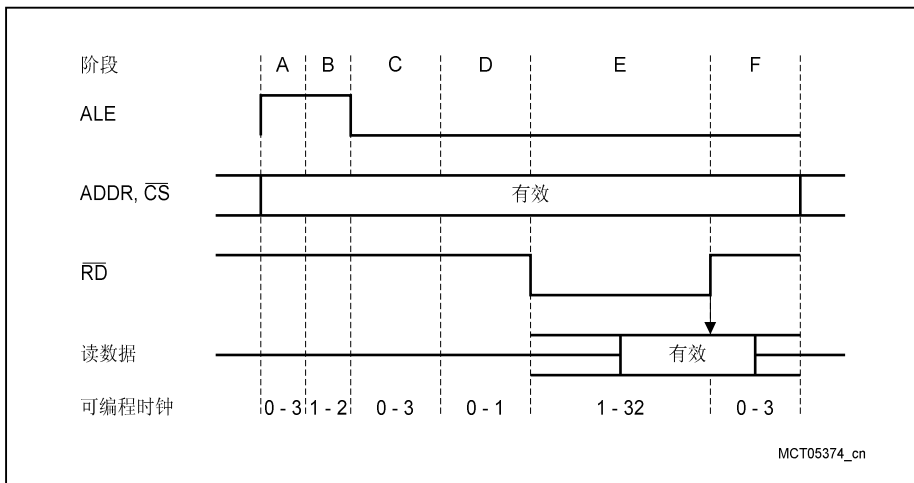
### F 拍 – 地址/写数据保持

F 拍为一个访问周期的最后一拍，占用 0 - 3 个时钟周期。

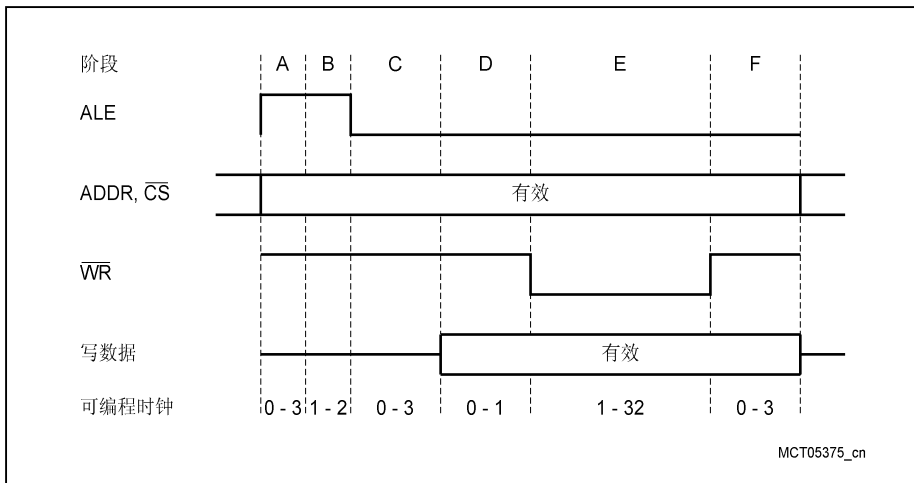
命令无效后，地址和写数据被保持。对于读访问和写访问，在该节拍期间插入的等待周期数可单独编程设定。**F** 拍用来设置双向数据总线上的高阻等待周期以避免总线冲突。

### 11.4.2 非复用总线

以下框图显示非复用总线访问的读和写时序。



**图 11-1 非复用总线读时序**

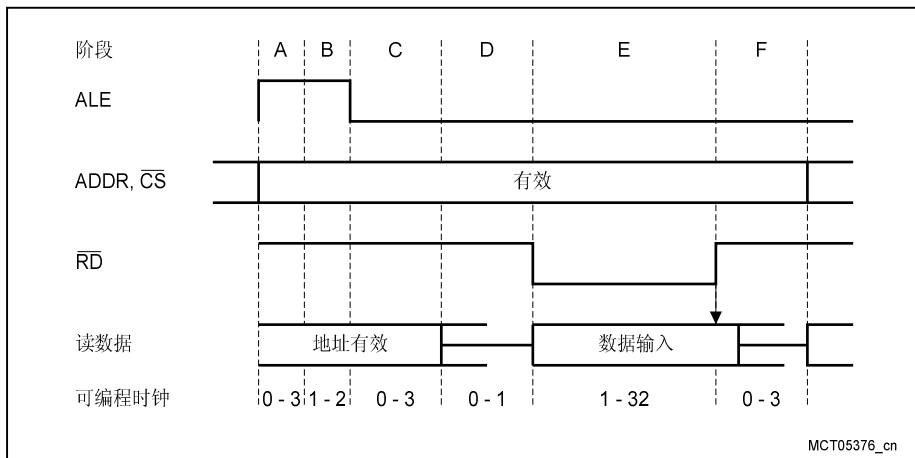


**图 11-2 非复用总线写时序**

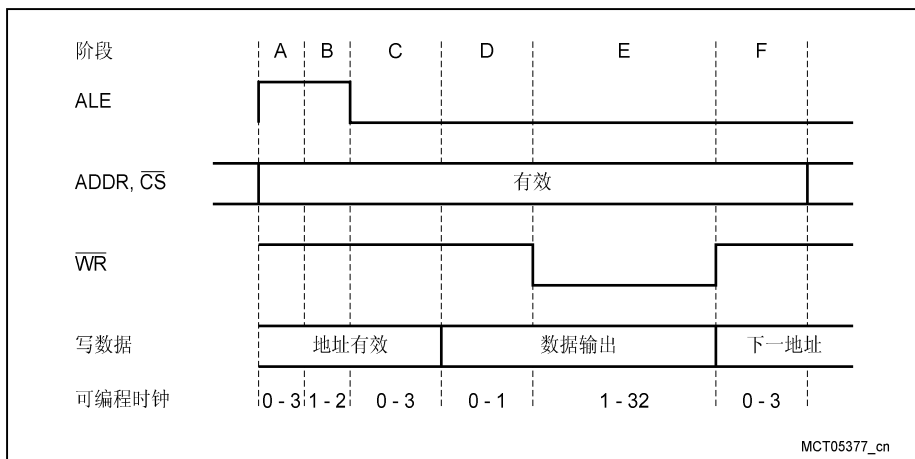
- 节拍 A: 地址有效, ALE 高, 无命令。 $\overline{CS}$  切换后的高阻等待周期
- 节拍 B: 地址有效, ALE 高, 无命令。ALE 有效时间长度
- 节拍 C: 地址有效, ALE 低, 无命令。R/W 延迟
- 节拍 D: 写数据有效, ALE 低, 无命令。写周期数据有效
- 节拍 E: 命令 (读或写) 有效。访问时间
- 节拍 F: 命令无效, 地址保持。读数据呈高阻态时间, 写数据保持时间

### 11.4.3 复用总线

以下框图显示复用总线访问的读和写时序。



**图 11-3 复用总线读时序**



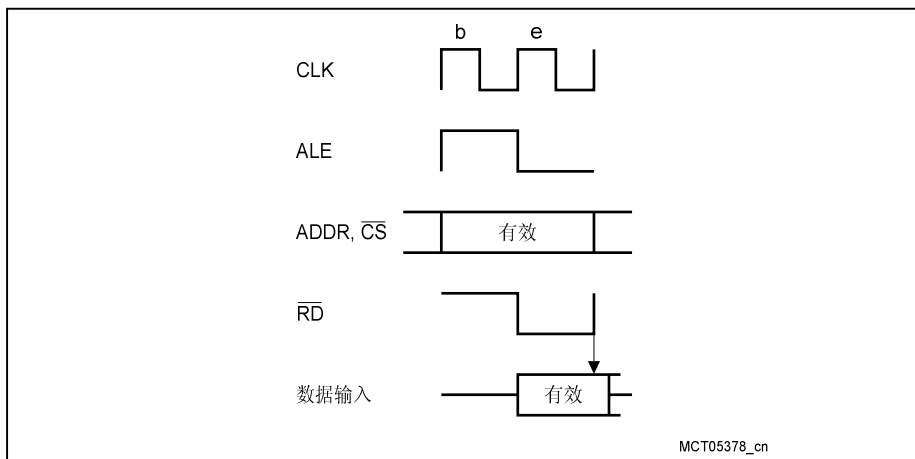
**图 11-4 复用总线写时序**

- 节拍 A: 地址有效, ALE 高, 无命令。  $\overline{CS}$  切换后的高阻等待周期
- 节拍 B: 地址有效, ALE 高, 无命令。 ALE 有效时间长度
- 节拍 C: 地址有效, ALE 低, 无命令。地址保持, R/W 延迟

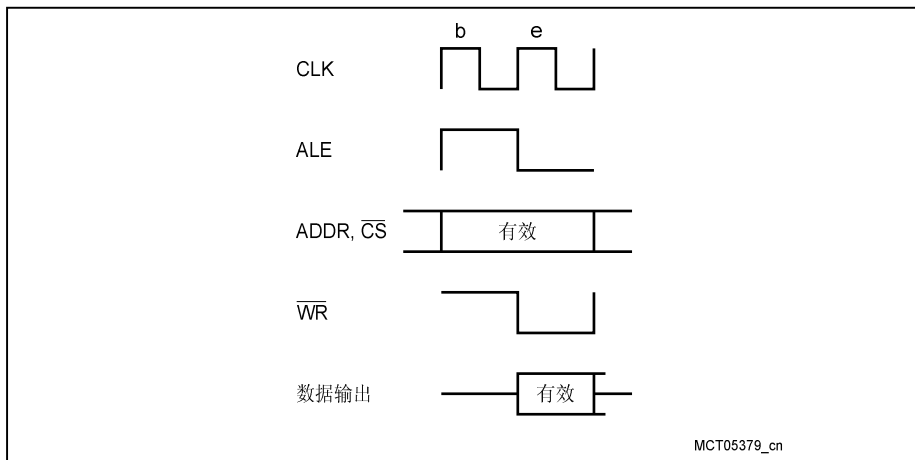
- 节拍 D: 读周期地址高阻态, 写周期数据有效, ALE 低, 无命令
- 节拍 E: 命令 (读或写) 有效。访问时间
- 节拍 F: 命令无效, 地址保持。读数据呈高阻态时间, 写数据保持时间

#### 11.4.4 最快访问周期

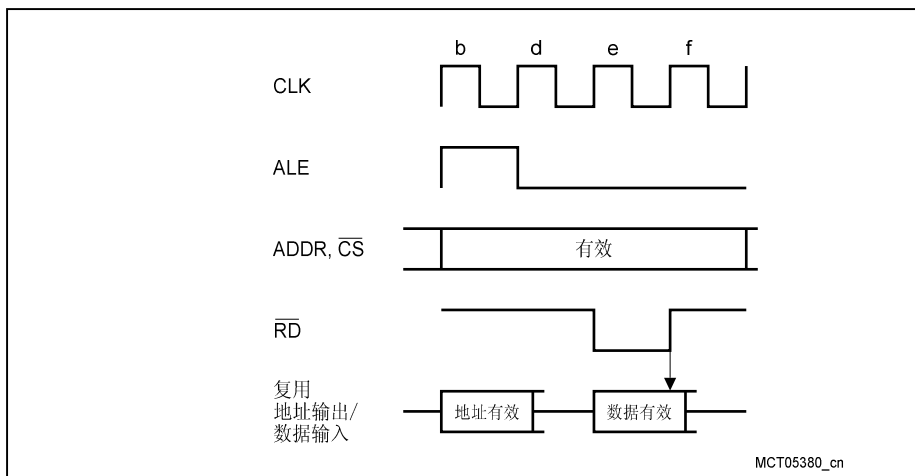
系统中最快的总线周期还和引出端的时序有关。因此, 总线访问所需的周期数和当前系统频率有关。在极高的系统频率下, 无法达到下图所示的最快总线周期。



**图 11-5 最快读周期, 非复用总线**



**图 11-6 最快写周期，非复用总线**



**图 11-7 最快读周期，复用总线**

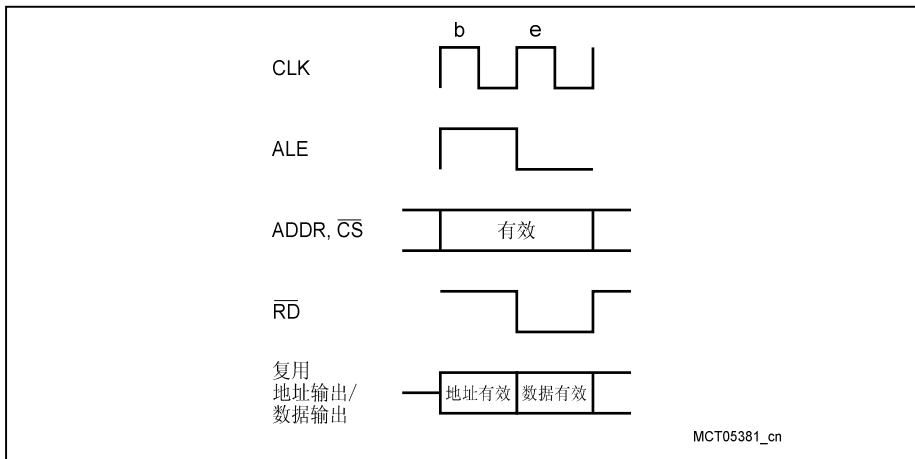


图 11-8 最快写周期，复用总线

## 11.5 地址窗

EBC 提供高达 8 路总线通道。其中 7 路通道（分配给  $\overline{CS1} \dots \overline{CS7}$ ）的地址窗可通过 ADDRSEL（1...7）寄存器编程设定。其余通道（分配给  $\overline{CS0}$ ）没有地址选择寄存器，它会覆盖剩余的 EBC 地址空间。

需要注意：在 XE16xyM 中，并非所有地址空间可被 EBC 访问。有关整个地址空间的分配情况，请参阅“存储器结构”一章。

### 11.5.1 地址窗分配

根据表 11-1 定义地址窗的大小和起始地址。地址窗的大小由 ADDRSELx.RGSZ 选择。根据地址窗的大小，ADDRSELx.RGSAD 中的相关位（由“R”标记）用于选择相应窗口的起始地址。ADDRSELx.RGSAD 的低位（标记为“x”）被忽略。

另外，必须通过置位 FCONCSx.ENCS 使能所设定的地址窗。

表 11-1 由 ADDRSELx 选定的地址区域和大小

ADDRSELx		地址窗	
区域大小 RGSZ	RGSAD 的相关位 (R)	选定的地址区域	地址窗起始地址 A[23:0] 用 RGSAD 中的 R 位选择

ADDRSELx		地址窗	
区域大小 RGSZ	RGSAD 的相关位 (R)	选定的地址区域	地址窗起始地址 A[23:0] 用 RGSAD 中的 R 位选择
3 ... 0	15 ... 4	大小	A23 ... A0
0000	RRRR RRRR RRRR	4 KB	RRRR RRRR RRRR 0000 0000 0000
0001	RRRR RRRR RRRx	8 KB	RRRR RRRR RRR0 0000 0000 0000
0010	RRRR RRRR RRxx	16 KB	RRRR RRRR RR00 0000 0000 0000
0011	RRRR RRRR Rxxx	32 KB	RRRR RRRR R000 0000 0000 0000
0100	RRRR RRRR xxxx	64 KB	RRRR RRRR 0000 0000 0000 0000
0101	RRRR RRRx xxxx	128 KB	RRRR RRR0 0000 0000 0000 0000
0110	RRRR RRxx xxxx	256 KB	RRRR RR00 0000 0000 0000 0000
0111	RRRR Rxxx xxxx	512 KB	RRRR R000 0000 0000 0000 0000
1000	RRRR xxxx xxxx	1 MB	RRRR 0000 0000 0000 0000 0000
1001	RRRx xxxx xxxx	2 MB	RRR0 0000 0000 0000 0000 0000
1010	RRxx xxxx xxxx	4 MB	RR00 0000 0000 0000 0000 0000
1011	Rxxx xxxx xxxx	8 MB	R000 0000 0000 0000 0000 0000
11xx	xxxx xxxx xxxx	保留 <sup>1)</sup>	---- ---- ---- ---- ---- ----

### 11.5.2 地址窗重叠

由于允许地址区重叠，因此定义了仲裁机制以处理这些情况。每次经由 EBC 进行总线访问时，EBC 会将当前的地址和所有使能的地址窗的地址选择寄存器进行比较。比较分三级进行。

#### 优先级 1:

首先评估使能地址窗的寄存器 ADDRSELx [x = 2, 4, 6]。一旦比较匹配，则开始访问相应通道。本组的地址窗重叠会导致不可预知的行为。

#### 优先级 2:

接下来评估使能地址窗的寄存器 ADDRSELy [y = 1, 3, 5, 7]。一旦比较匹配，则开始访问相应通道。本组的地址窗重叠会导致不可预知的行为。与优先级 1 的 ADDRSELx 重叠时，(x, y) 只能是 (2, 1)、(4, 3) 或 (6, 5)。

#### 优先级 3:

如果通道 0 被使能，则访问该通道。否则不执行总线操作。



## 11.6 READY 控制的总线周期

在一些情况下，外设的响应时间不是恒定的，或者可编程的等待周期不能满足外设响应时间的需要，此时可采用 XE16xyM EBC 提供的“READY 控制的总线访问方案”。

在该方案中，通过 READY 输入信号来终止总线访问。在节拍 E 期间，EBC 首先计数若干个时钟周期（1...32，个数可编程设定），在最后一个等待周期开始监控内部 READY 线（见图 11-9 的“READY 内部”），从而确定当前总线周期何时真正结束。外部器件将 READY 驱动为有效以指示数据已经被锁存（写周期）或数据可用（读周期）。

由 READY 控制的总线周期需要一个同步周期才能终止。周期可编程的节拍 F 包括该同步周期在内。因此，设置 TCONCSx 节拍 F 为 0 个时钟周期和将其设置为 1 个时钟周期的效果相同。

### 11.6.1 使能 READY 控制

为了激活相应的端口引脚，将 EBCMOD0 中的位 RDYDIS 设置为“0”以使能 READY 引脚。还可通过 EBCMOD0 寄存器中的位 RYDPOL 定义 READY 的极性。

可通过 FCONCSx 寄存器中的位 RDYEN 使能特定地址窗的 READY 功能。通过 FCONCSx.RDYM0D 设置 READY 信号的处理模式为同步还是异步（参见图 11-9）。

当特定地址窗的 READY 功能被使能时，该地址窗内的每个总线周期都必须由一个有效的 READY 信号完全终止。否则，EBC 将被该挂起的访问完全中止。

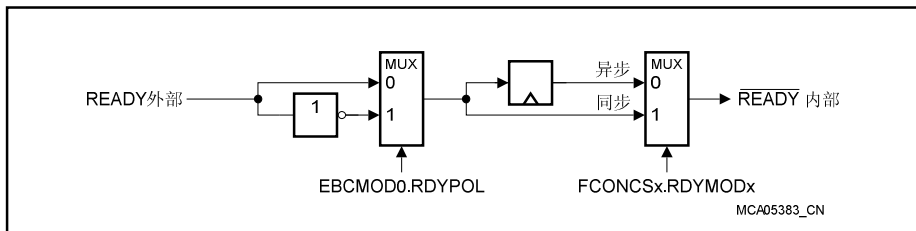


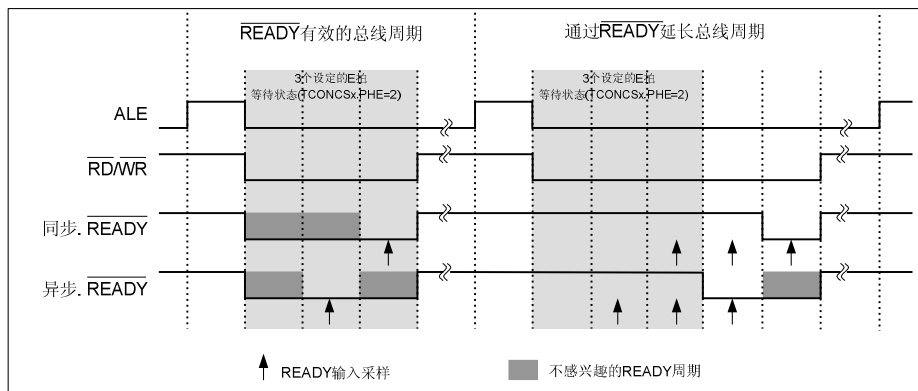
图 11-9 外部至内部的 READY 转换

### 11.6.2 同步和异步 READY

同步 READY 提供更快的总线周期，但是需要满足建立和保持时间的要求。此情况下可使能 CLKOUT 信号以允许外设逻辑使用该信号控制 READY 的时序。

异步 READY 所受的限制较少，但是内部同步需要一个附加等待周期。由于之前就对异步 READY 进行采样，因此需要可编程的等待周期，以提供正确的总线周期。

已被外部器件激活的 **READY** 信号（尤其是异步 **READY**）应该在响应相应命令（**RD** 或 **WR**）的尾沿（上升沿）后失效。



**图 11-10 READY 控制的总线周期**

### 11.6.3 READY 功能与预定义的等待周期相结合

当一个周期开始时，通常外部等待周期或 **READY** 控制逻辑需要一些时间来产生 **READY** 信号。预先定义个数的时钟周期之后，**XE16xyM** 开始检查其 **READY** 线，以确定总线周期的结束。

当被称为“正常 **READY**”外设使用 **READY** 功能时，如果过早的对 **READY** 线采样，可能导致错误的总线周期。空闲时，这些外设将其 **READY** 输出设置为有效。这些外设被访问时，它们将 **READY** 输出驱动为无效，直到总线周期结束，然后再次将 **READY** 驱动为有效。然而，如果外设将 **READY** 驱动为无效的时间稍晚，即在 **XE16xyM** 第一个采样点之后，控制器采样到有效的 **READY** 信号并过早的终止当前总线周期。通过插入预先定义的等待周期，第一个 **READY** 采样点可被延迟一段时间，此时外设已经安全的对 **READY** 线进行了控制。

## 11.7 外部总线仲裁

**XE16xyM** 通过外部总线仲裁机制支持外部总线上的多主设备系统。总线仲裁允许外部主设备请求外部总线，**XE16xyM** 将释放外部总线，将数据和地址总线悬空并通过上拉/下拉器件强制控制信号无效。

**注意：**只有当存储器保护单元（MPU）禁用时该特性才可用。

### 11.7.1 仲裁初始化

复位期间，所有的仲裁引脚都为高阻态，引脚  $\overline{\text{BREQ}}$  除外（它保持无效状态）。复位之后，XE16xyM EBC 始终在“初始化模式”下启动，此时外部总线可用，但是仲裁未被使能。此状态下，忽略所有的仲裁引脚。假设其它与外部总线相连的 XE16xyM EBC 也具有总线控制权，因此潜在的总线冲突并未解决。对于具有多个主设备的系统，在启动任意总线访问之前，首先应当进行仲裁初始化。通过编程设置寄存器 EBCMOD0 中的 SLAVE 位，EBC 可被选作仲裁主设备或仲裁从设备。对寄存器（CPU 的）PSW 寄存器中的 HLDEN 位进行第一次设置之后，所选择的模式和仲裁过程变为有效。之后如果不复位器件，则不可能改变主/从模式的设置。当然，可通过置位 EBCMOD0.ARBEN 来激活对专用引脚的仲裁。

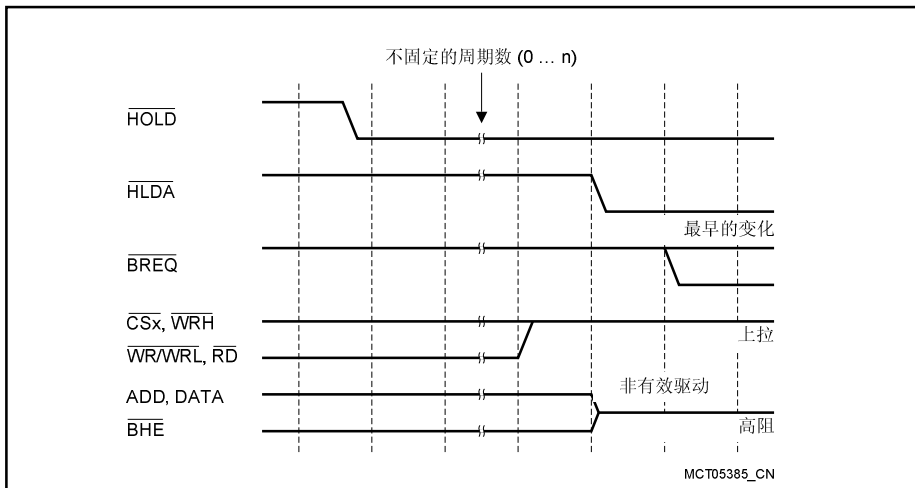
### 11.7.2 仲裁主设备方案

如果 XE16xyM 被配置为仲裁主设备，那么它作为外部总线控制权的缺省所有者来控制仲裁协议，空闲期间，在没有总线请求的情况下驱动总线。为了执行仲裁握手， $\overline{\text{HOLD}}$  输入接收来自仲裁主设备的对外部总线的请求。当仲裁主设备将总线控制权移交给总线请求设备时，将保持应答引脚  $\overline{\text{HLDA}}$  驱动为低电平指示总线控制权的移交，该引脚保持低电平，直到仲裁从设备通过使  $\overline{\text{HOLD}}$  输入上的请求无效的方式来释放总线控制权。如果仲裁主设备不是总线的所有者，那么它会依照以下方式处理外部总线接口：

- 地址和数据总线变为高阻态
- 通过内部上拉器件将命令线拉高（ $\overline{\text{RD}}$ 、 $\overline{\text{WR}}$  /  $\overline{\text{WRL}}$ 、 $\overline{\text{BHE}}$  /  $\overline{\text{WRH}}$ ）
- 通过内部下拉器件将地址锁存控制线 ALE 拉低
- 通过内部上拉器件将  $\overline{\text{CSx}}$  输出拉高

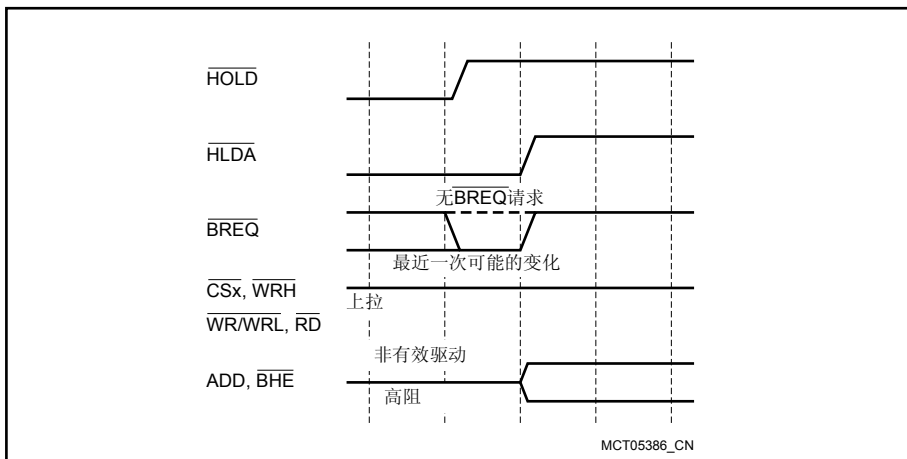
在该状态下，仲裁从设备能够接管总线。

如果仲裁主设备再次需要总线所有权，可通过总线请求信号  $\overline{\text{BREQ}}$  请求总线。一旦仲裁主设备再次获得总线所有权，其立即释放  $\overline{\text{BREQ}}$  信号并将  $\overline{\text{HLDA}}$  驱动为高电平。



**图 11-11 仲裁主设备释放总线**

注：上图给出  $\overline{BREQ}$  变为有效的第一种可能性。在授权外部总线之前，XE16xyM 将会完成当前运行的总线周期，如图中的折线所示。



**图 11-12 仲裁主设备重获总线所有权**

注:  $\overline{BREQ}$  的下降沿显示由  $\overline{BREQ}$  触发的重获总线所有权过程的最近一次机会, 即使在通过将  $\overline{HOLD}$  变为高电平来启动重获总线所有权的过程之前,  $\overline{BREQ}$  被激活。请注意 XE16xyM 不请求总线的情况下,  $\overline{HOLD}$  也可能失效。

### 11.7.3 仲裁从设备方案

如果 EBC 被配置为仲裁从设备, 则缺省设置为外部总线所有权不属于 EBC, 因此必须首先请求总线。只要还未完成所有请求队列, 且总线主设备未请求总线, 则仲裁从设备拥有总线所有权。握手信号处理的描述请参考[章节 11.7.2](#)。对于仲裁从设备来说, 保持应答引脚 HLDA 被配置为输入。

### 11.7.4 总线锁定功能

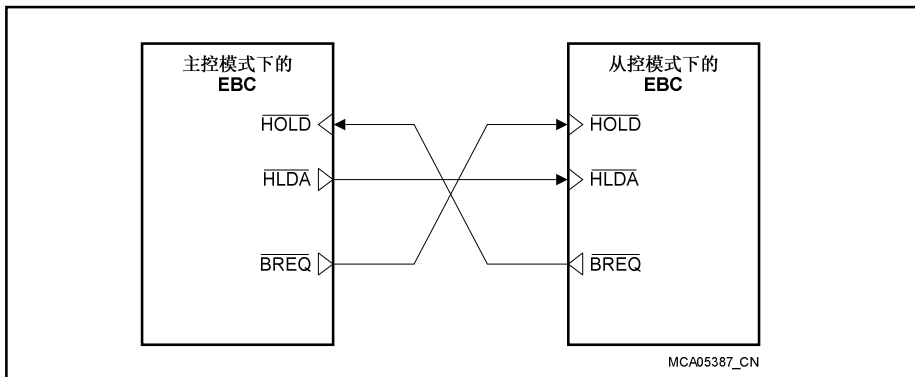
如果多主设备系统的应用需要一个不被干扰的总线访问序列, 可 (与 EBC 为总线主设备还是从设备无关) 通过设置 PSW 位 HLDEN 为 ‘0’ 的方式锁定<sup>1)</sup> 总线。在此情况下, 锁定的 EBC 不回应对来自其它外部总线主设备的  $\overline{HOLD}$  请求, 直到 HLDEN 再次变为 ‘1’ 为止。当然被锁定的、但并不拥有总线控制权的总线主设备可以请求外部总线。如果主设备和从设备同时请求外部总线进行多次访问操作, 如果总线未被锁定, 则每个访问周期之后, 总线所有权在主设备和从设备之间切换。

### 11.7.5 直接连接主设备和从设备

如果一个 XE16xyM 被配置为主机, 而另一个被配置为从机, 这两个微控制器连接在同一个外部总线上, 且都是总线主设备, 则它们可直接相连进行总线仲裁 (如[图 11-13](#)所示)。因为两个 EBC 都认为复位之后, 都假设自己拥有外部总线控制权, 因此在启动 “主机” CPU 之前, “从机” CPU 先不进行复位和初始化。另一种方式是同时启动两个系统, 然后在第一个外部总线请求之前, 必须由内部存储器配置两个 EBC 且 PSW.HLDEN 位必须置位。

---

1) 不允许通过复位 EBCMOD0.ARBEN 位的方式锁定总线, 因为这种情况会导致总线冲突。



**图 11-13 使用主/从仲裁连接两个 XE16xyM**

当多个（多于两个）总线主设备（XE16xyM 或其它主设备）共享同一个外部资源时，则需要额外的总线仲裁逻辑，以确定当前有效的总线主设备并控制必须的信号序列。

## 11.8 EBC 空闲状态

当 EBC 的外部总线接口被使能但当前并不执行内部或外部访问时，EBC 空闲。只要有一个片上资源，如 RAM、外设（不包含通过 EBC 连接的 LXBUS 外设）或寄存器等被使用，外部总线接口不改变（见 [表 11-2](#)）。

在 EBC 空闲期间，外部控制信号（ $\overline{RD}$  和  $\overline{WR}$  或  $\overline{WRL} / \overline{WRH}$ ，若被使能）保持无效（高电平）状态。

**表 11-2 EBC 空闲期间外部总线输出的状态**

引脚	EBC 空闲期间引脚的状态
AD15...AD0	高阻（悬空）
A15...A0	未定义的地址（若用于总线接口）
A23...A16	未定义的段地址（在选定的引脚上）
$\overline{CS7} \dots \overline{CS0}$	无效（高电平）
$\overline{BHE}$	对应上次外部访问的电平

引脚	EBC 空闲期间引脚的状态
ALE	无效（低电平）
$\overline{RD}$ 、 $\overline{WR}$ 、 $\overline{WRL}$ 、 $\overline{WRH}$	无效（高电平）

## 11.9 寄存器描述

EBC 寄存器位于内部 I/O 区。

**表 11-3 寄存器地址空间**

模块	起始地址	结束地址	注
EBC	00EE00 <sub>H</sub>	00EEFF <sub>H</sub>	

### 11.9.1 EBC 模式寄存器

两个模式寄存器控制 XE16xyM EXTBUS 引脚的使用和配置。被禁用的 EXTBUS 引脚可用作通用 IO 引脚，详见“并行端口”一章的描述。

#### EBCMOD0

#### EBC模式寄存器0

**XSFR (00<sub>H</sub>)**

**复位值: 5000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RDY POL</b>	<b>RDY DIS</b>	<b>ALE DIS</b>	<b>BYT DIS</b>	<b>WR CFG</b>	<b>EBC DIS</b>	<b>SLA VE</b>	<b>ARB EN</b>	<b>CSPEN</b>				<b>SAPEN</b>			
rw	rw	rw	rw	rw	rw	rw	rw	rw				rw			

符号	位序号	类型	功能描述
<b>RDYPOL</b>	15	rw	<b>READY 引脚极性</b> 0 <sub>B</sub> READY 低有效 1 <sub>B</sub> READY 高有效
<b>RDYDIS</b>	14	rw	<b>READY 引脚禁止</b> 0 <sub>B</sub> READY 使能 1 <sub>B</sub> READY 禁止
<b>ALEDIS</b>	13	rw	<b>ALE 引脚禁止</b>

符号	位序号	类型	功能描述
			$0_B$ ALE 使能 $1_B$ ALE 禁止
<b>BYTDIS</b>	12	rw	<b><math>\overline{BHE}</math> 引脚禁止</b> $0_B$ $\overline{BHE}$ 使能 $1_B$ $\overline{BHE}$ 禁止
<b>WRCFG</b> <sup>1)</sup>	11	rw	$\overline{WR}$ / $\overline{WRL}$ , $\overline{BHE}$ / $\overline{WRH}$ 引脚配置 $0_B$ $\overline{WR}$ 和 $\overline{BHE}$ $1_B$ $\overline{WRL}$ 和 $\overline{WRH}$
<b>EBCDIS</b>	10	rw	<b>EBC 引脚禁止</b> $0_B$ EBC 使用外部总线引脚 $1_B$ EBC 引脚禁止
<b>SLAVE</b>	9	rw	<b>从模式使能</b> $0_B$ 总线仲裁器工作在主模式 $1_B$ 总线仲裁器工作在从模式
<b>ARBEN</b>	8	rw	<b>总线仲裁引脚使能</b> $0_B$ $\overline{HOLD}$ , $\overline{HLDA}$ 和 $\overline{BREQ}$ 引脚被禁止 $1_B$ 引脚用作 $\overline{HOLD}$ , $\overline{HLDA}$ 和 $\overline{BREQ}$
<b>CSPEN</b>	[7:4]	rw	<b><math>\overline{CSx}</math> 引脚使能（只针对外部 <math>\overline{CSx}</math>）</b> $0_H$ 所有外部片选引脚禁止 $1_H$ $\overline{CS0}$ 引脚使能 $2_H$ $\overline{CS1}$ 和 $\overline{CS0}$ 引脚使能 ... $5_H$ 五个 $\overline{CSx}$ 引脚使能: $\overline{CS4}$ - $\overline{CS0}$ 不支持其它位组合（保留）
<b>SAPEN</b>	[3:0]	rw	<b>段地址引脚使能</b> $0_H$ 所有段地址引脚禁止 $1_H$ 1 个: $A[16]$ 使能



符号	位序号	类型	功能描述
			... .. 8 <sub>H</sub> 8 个: A[23:16] 使能 不支持其它位组合（保留）

1) 在下一个外部总线访问周期之前，该位的内容改变无效。

### 字节写配置

在 16 位数据总线配置中，字节写特性可通过位域 WRCFG 编程设定。下表给出相关信号的功能。

**表 11-4 字节写配置**

写字节		WRCFG = 0			WRCFG = 1		
低	高	$\overline{\text{WR}}$	BHE	ADDR[0]	$\overline{\text{WRL}}$	$\overline{\text{WRH}}$	ADDR[0]
-	-	无效	不关心	0/1	无效	无效	0/1
写	-	有效	无效	0	有效	无效	0/1
-	写	有效	有效	1	无效	有效	0/1
写	写	有效	有效	0	有效	有效	0/1

### EBCMOD1

#### EBC模式寄存器1

#### XSFR (02<sub>H</sub>)

复位值: 003F<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							WRP DIS	DHP DIS	ALP DIS	A0P DIS	APDIS				
r							rw	rw	rw	rw	rw				

符号	位序号	类型	功能描述
<b>0</b>	[15:8]	r	保留 读取返回 0，应写入 0。
<b>WRPDIS</b>	7	rw	$\overline{\text{WR}} / \overline{\text{WRL}}$ 引脚禁止 0 <sub>B</sub> 引脚 $\overline{\text{WR}} / \overline{\text{WRL}}$ 使能

符号	位序号	类型	功能描述
			1 <sub>B</sub> 引脚 $\overline{WR}$ / $\overline{WRL}$ 禁止
<b>DHPDIS</b>	6	rw	高位数据端口引脚禁止 0 <sub>B</sub> 地址/数据总线引脚 15 - 8 使能 1 <sub>B</sub> 地址/数据总线引脚 15 - 8 禁止
<b>ALPDIS</b>	5	rw	低位地址引脚禁止 0 <sub>B</sub> 地址总线引脚 7 - 0 使能 (取决于 APDIS/A0PDIS) 1 <sub>B</sub> 地址总线引脚 7 - 0 禁止
<b>A0PDIS</b>	4	rw	地址位 A0 引脚禁止 0 <sub>B</sub> 地址总线引脚 0 使能 1 <sub>B</sub> 地址总线引脚 0 禁止
<b>APDIS</b>	[3:0]	rw	地址端口引脚禁止 0 <sub>H</sub> 地址总线引脚15 - 1使能 1 <sub>H</sub> 引脚A15 禁止, A14 - A1使能 2 <sub>H</sub> 引脚A15 - A14 禁止, A13-A1 使能 3 <sub>H</sub> 引脚A15 - A13 禁止, A12-A1 使能 ... E <sub>H</sub> 引脚A15 - A2 禁止, A1 使能 F <sub>H</sub> 地址总线引脚15 - 1禁止

### 11.9.2 时序控制寄存器

时序控制寄存器用来设置总线周期中各节拍占用的周期数。从受影响的地址窗读取代码的过程中，该寄存器可以被重新设置，新设置在随后的访问中有效。

#### TCONCS0

**CS0时序控制寄存器**

**XSFR (10<sub>H</sub>)**

**复位值: 7C3D<sub>H</sub>**

**TCONCSx (x=1-4)**

**CSx时序控制寄存器**

**XSFR (10<sub>H</sub>+x\*8)**

**复位值: 0000<sub>H</sub>**

**外部总线控制器（EBC）**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>WRPHF</b>	<b>RDPHF</b>	<b>PHE</b>			<b>PHD</b>	<b>PHC</b>	<b>PHB</b>	<b>PHA</b>						
r	rw	rw	rw			rw	rw	rw	rw						

符号	位序号	类型	功能描述
<b>0</b>	15	r	<b>保留</b> 读取返回 0，应写入 0。
<b>WRPHF</b>	[14:13]	rw	<b>写节拍 F</b> 00 <sub>B</sub> 0 个时钟周期 ... ... 11 <sub>B</sub> 3 个时钟周期
<b>RDPHF</b>	[12:11]	rw	<b>读节拍 F</b> 00 <sub>B</sub> 0 个时钟周期 ... ... 11 <sub>B</sub> 3 个时钟周期
<b>PHE</b>	[10:6]	rw	<b>节拍 E</b> 0 <sub>H</sub> 1 个时钟周期 ... ... 1F <sub>H</sub> 32 个时钟周期
<b>PHD</b>	5	rw	<b>节拍 D</b> 0 <sub>B</sub> 0 个时钟周期 1 <sub>B</sub> 1 个时钟周期
<b>PHC</b>	[4:3]	rw	<b>节拍 C</b> 00 <sub>B</sub> 0 个时钟周期 ... ... 11 <sub>B</sub> 3 个时钟周期
<b>PHB</b>	2	rw	<b>节拍 B</b> 0 <sub>B</sub> 1 个时钟周期 1 <sub>B</sub> 2 个时钟周期
<b>PHA</b>	[1:0]	rw	<b>节拍 A</b>

符号	位序号	类型	功能描述
			00 <sub>B</sub> 0 个时钟周期
			... ...
			11 <sub>B</sub> 3 个时钟周期

### 11.9.3 功能控制寄存器

功能配置寄存器用于控制选定地址窗的总线和 **READY** 功能。总线有 8 位和 16 位以及复用和非复用之分。此外，还可定义地址窗（及其片选信号  $\overline{CS_x}$ ）是否被使能。

#### FCONCS0

**CS0 功能控制寄存器**

**XSFR (12<sub>H</sub>)**

**复位值: 0011<sub>H</sub>**

**FCONCS<sub>x</sub> (x=1-4)**

**CS<sub>x</sub> 功能控制寄存器**

**XSFR (12<sub>H</sub>+x\*8)**

**复位值: 0000<sub>H</sub>**

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	BTYP	0	RDY MOD	RDY EN	EN CS	
r	r	r	r	r	r	r	r	r	r	rw		r	rw	rw	rw

符号	位序号	类型	功能描述
<b>0</b>	[15:6]	r	保留 读取返回 0，应写入 0。
<b>BTYP</b>	[5:4]	rw	总线类型选择 00 <sub>B</sub> 8 位非复用总线 01 <sub>B</sub> 8 位复用总线 10 <sub>B</sub> 16 位非复用总线 11 <sub>B</sub> 16 位复用总线
<b>0</b>	3	r	保留 读取返回 0，应写入 0。
<b>RDYMOD</b>	2	rw	准备就绪模式 0 <sub>B</sub> 异步 READY 1 <sub>B</sub> 同步 READY

符号	位序号	类型	功能描述
<b>RDYEN</b>	1	rw	<b>准备就绪使能</b> 0 <sub>B</sub> 访问时间由位域 PHE <sub>x</sub> 控制 1 <sub>B</sub> 访问时间由位域 PHE <sub>x</sub> 和 READY 信号控制
<b>ENCS</b>	0	rw	<b>片选使能</b> 0 <sub>B</sub> 禁止 1 <sub>B</sub> 使能

注：FCONCS<sub>x</sub> 寄存器中的位 ENCS<sub>x</sub> 用于使能相关地址窗、总线功能以及对应的片选信号  $\overline{CS_x}$ 。此外，它还和寄存器 EBCMOD0 中位域 CSPEN 的定义有关（有多少个  $\overline{CS_x}$  引脚可供外部系统使用）。如果一个地址窗被使能，但是没有外部引脚可供使用，则在不插入外部片选信号的情况下执行总线周期。

#### 11.9.4 地址窗选择寄存器

##### ADDRSEL<sub>x</sub> (x = 1-4)

CS<sub>x</sub>对应的地址区域/大小

XSFR (16<sub>H</sub>+x\*8)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RGSAD</b>												<b>RGSZ</b>			
rw												rw			

符号	位序号	类型	功能描述
<b>RGSAD</b>	[15:4]	rw	<b>地址区起始地址</b> 地址区起始地址选择
<b>RGSZ</b>	[3:0]	rw	<b>地址区大小</b> 地址区大小选择（见表 11-1）

## 11.10 XE16xyM 中 EBC 的实现

XE16xyM 中的 EBC 会自动影响其它器件元件的行为。尤其在存储器映射和端口分配方面，EBC 优先级较高。为了更好的理解 EBC 和存储器及端口的关系，请参阅以下章节：

- 存储器结构
- 并行端口

### 11.10.1 未使用的寄存器

XE16xyM 外部片选信号的个数受限于芯片的封装。该限制导致相应的 EBC 通道不可用。因此，和通道 x (x=5,6) 相关的寄存器 ADDRSELx、TCONCSx、FCONCSx 不可用。出于软件兼容的考虑，切勿读写对应的寄存器地址空间。

### 11.10.2 对 LXBUS 模块的访问控制

在 XE16xyM 中，EBC 通道 7 保留用于访问片内 LXBUS 外设。通常，访问 LXBUS 在外部总线 EXTBUS 上不可见。在 LXBUS 访问周期内，EXTBUS 保持使能、但被驱动至无效状态（控制信号）或切换到读模式（总线）。

### 11.10.3 关闭控制

接收到 SCU 发出的关闭请求之后，在整个芯片被切换到省电模式之前，EBC 必须确保其所有功能处于非工作状态。除了要完成正在运行的总线周期，还要执行已经请求的总线周期。根据 EBC 的主设备/从设备配置，在执行请求周期之前，控制外部总线仲裁器重新获得总线（主机）；或者在完成已经请求的总线周期之后，必须释放外部总线。只有当该关闭序列结束时，EBC（或其它模块，如 SCU 中的描述）才产生关闭应答，此时芯片能够进入到所请求的模式。

**表 11-5** 给出不同 EBC 配置下的 EBC 关机控制。

**表11-5      EBC 关机控制**

仲裁模式	主模式		从模式	
总线控制	带总线控制	不带总线控制	带总线控制	不带总线控制
-	完成所有挂起的周期请求。利用总线控制发送关机应答。	请求总线。完成所有挂起的周期请求。利用总线控制发送关机应答。	完成所有挂起的周期请求。释放总线后发送关机应答。	如果需要，请求总线。完成所有挂起请求。释放总线后发送关机应答。

## 11.10.4 专用寄存器

EBC 专用寄存器位于内部 I/O 区，属于 XSFR 区的寄存器。

### 11.10.4.1 LXBUS 模块专用寄存器

访问 MultiCAN 和 USIC 时，将使用  $\overline{\text{CS7}}$  和其控制寄存器 ADDRSEL7、TCONCS7 和 FCONCS7。LXBUS 通过  $\overline{\text{CS7}}$  在片内控制选择。

#### 寄存器 TCONCS7

LXBUS 的周期时序由寄存器 TCONCS7 控制。它使用最快的总线周期，即一个总线周期包含两个时钟周期。模块若使用 READY 功能，总线周期将被插入的等待状态所延长。TCONCS7 的复位值会影响该周期时序。

#### TCONCS7

##### CS7 时序控制寄存器

##### XSFR (48H)

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	WRPHF	RDPHF	PHE			PHD	PHC	PHB	PHA						
r	rw	rw	rw			rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
0	15	r	保留 读取返回 0，应写入 0。
WRPHF	[14:13]	rw	写节拍 F 00 <sub>B</sub> 0 个时钟周期 ... ... 11 <sub>B</sub> 3 个时钟周期
RDPHF	[12:11]	rw	读节拍 F 00 <sub>B</sub> 0 个时钟周期 ... ... 11 <sub>B</sub> 3 个时钟周期
PHE	[10:6]	rw	节拍 E 0 <sub>H</sub> 1 个时钟周期 ... ...

**外部总线控制器（EBC）**

符号	位序号	类型	功能描述
			1F <sub>H</sub> 32 个时钟周期
<b>PHD</b>	5	rw	节拍 <b>D</b> 0 <sub>B</sub> 0 个时钟周期 1 <sub>B</sub> 1 个时钟周期
<b>PHC</b>	[4:3]	rw	节拍 <b>C</b> 00 <sub>B</sub> 0 个时钟周期 ... .. 11 <sub>B</sub> 3 个时钟周期
<b>PHB</b>	2	rw	节拍 <b>B</b> 0 <sub>B</sub> 1 个时钟周期 1 <sub>B</sub> 2 个时钟周期
<b>PHA</b>	[1:0]	rw	节拍 <b>A</b> 00 <sub>B</sub> 0 个时钟周期 ... .. 11 <sub>B</sub> 3 个时钟周期

**寄存器 FCONCS7**

根据内部连接模块的需求（16 位非复用总线、访问时间由同步 READY 控制）来选择该专用总线功能控制寄存器的值。

**FCONCS7**

**CS7 功能控制寄存器**

**XSFR (4A<sub>H</sub>)**

**复位值: 0027<sub>H</sub>**

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	BTYP	0	RDY MOD	RDY EN	EN CS	
r	r	r	r	r	r	r	r	r	r	rw	r	rw	rw	rw	

符号	位序号	类型	功能描述
<b>0</b>	[15:6]	r	保留 读取返回 0，应写入 0。



**外部总线控制器（EBC）**

符号	位序号	类型	功能描述
<b>BTYP</b>	[5:4]	rw	<b>总线类型选择</b> 00 <sub>B</sub> 8 位非复用总线 01 <sub>B</sub> 8 位复用总线 10 <sub>B</sub> 16 位非复用总线 11 <sub>B</sub> 16 位复用总线
<b>0</b>	3	r	<b>保留</b> 读取返回 0，应写入 0。
<b>RDYMOD</b>	2	rw	<b>准备就绪模式</b> 0 <sub>B</sub> 异步 READY 1 <sub>B</sub> 同步 READY
<b>RDYEN</b>	1	rw	<b>准备就绪使能</b> 0 <sub>B</sub> 访问时间由位域 PHE <sub>x</sub> 控制 1 <sub>B</sub> 访问时间由位域 PHE <sub>x</sub> 和 READY 信号控制
<b>ENCS</b>	0	rw	<b>片选使能</b> 0 <sub>B</sub> 禁止 1 <sub>B</sub> 使能

**寄存器 ADDRSEL7**

该专用地址选择寄存器的值为相连的片内模块分配外部 IO 区地址段 200000<sub>H</sub> – 20FFFF<sub>H</sub>。

**ADDRSEL7**

**CS7 对应的地址区域/大小**

**XSFR (4E<sub>H</sub>)**

**复位值: 2004<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RGSAD</b>												<b>RGSZ</b>			
rw												rw			

符号	位序号	类型	功能描述
<b>RGSAD</b>	[15:4]	rw	<b>地址区起始地址</b>

外部总线控制器 (EBC)

符号	位序号	类型	功能描述
			地址区起始地址选择
RGSZ	[3:0]	rw	地址区大小 地址区大小选择 (见表 11-1)

## 12 启动配置和引导程序加载

启动之后，XE16xyM 从片上或片外程序存储器执行代码。可通过以下几种方式执行初始化代码（具体参见[章节 12.1](#)和[章节 12.3](#)）：

- **内部启动**模式：从片上程序 Flash 开始执行代码。
- **外部启动**模式：从与外部总线接口相连的片外存储器开始执行代码。
- **引导程序加载**模式：从片上程序 SRAM（PSRAM）开始执行代码，事先通过可选的串行接口下载这些代码。

### 12.1 启动模式选择

每次启动之后，由寄存器 SCU\_STSTAT 中的位域 HWCFG 指示当前有效的启动配置。XE16xyM 支持的启动配置和模式的描述见[章节 12.3](#)。

基本上可通过两种方式选择启动配置：

#### 1. 上电复位时的外部硬件配置：

通过专用  $\overline{\text{TRST}}$  引脚和端口 P10 引脚（P10[6:0]）进行硬件配置。

对于每次复位都使用硬件配置特性的系统来讲，在复位期间激活启动配置的硬件可能仅是上拉电阻；对于暂时使用硬件配置的系统，用户可能想要一个可切换的解决方案（通过跳线器或外部信号实现）。

#### 2. 执行以下代码序列（使用 SCU\_SWRSTCON 和 SCU\_RSTCON1 寄存器）：

- a) 向位域 SCU\_SWRSTCON.SWCFG 写入相应的配置值（请参考[表 12-2](#)和[表 12-3](#)）；
- b) 通过设置位域 SCU\_RSTCON1.SW 为软件复位请求分配期望的复位类型（缺省设置 SCU\_RSTCON1.SW = 00<sub>B</sub>意味着软件复位请求不产生复位）；
- c) 置位软件引导配置位：SCU\_SWRSTCON.SWBOOT = 1；
- d) 通过激活软件复位请求触发软件复位：SCU\_SWRSTCON.SWRSTREQ = 1。

另外，必须注意几种启动配置处理的特殊情况：

1. 由硬件请求触发的应用复位（例如：WDT、ESRx）– 不检测  $\overline{\text{TRST}}$  和 P10 引脚的值，仍使用与前一次复位后相同的启动配置。
2. 由软件触发的应用或内部应用复位（SCU\_SWRSTCON.SWRSTREQ = 1）– 可产生不同的结果：
  - a) 若选择软件引导配置（SCU\_SWRSTCON.SWBOOT = 1）– 由 SCU\_SWRSTCON.SWCFG 更新启动配置 SCU\_STSTAT.HWCFG；
  - b) 否则 – 使用与前一次复位之后相同的启动配置。

3. 由硬件请求触发的内部应用复位 – 只有  $\overline{\text{TRST}}$  引脚被检测（参考 [页 12-6](#)）：
  - a) 如果  $\overline{\text{TRST}} = 0$  – 选择从 Flash 内部启动，不允许进行调试；
  - b) 否则 – 使用与前一次复位之后相同的启动配置。

## 12.2 启动之后的器件状态

执行完第一条用户指令之后，XE16xyM 状态的主要参数总结如下：

### 12.2.1 被启动过程修改的寄存器

**表 12-1** 列出 XE16xyM 在启动过程中被初始化的、其初始化和复位值不同的寄存器：

根据启动方式的不同，这些寄存器被分为两组：

1. 每次启动操作之后被初始化的寄存器；
2. 由电源域 DMP\_1 的上电操作触发的启动过程之后被初始化的寄存器；

*注：电源域 DMP\_M 的上电操作意味着电源域 DMP\_1 也发生上电操作。*

**表 12-1** 中的寄存器根据上述规则进行分组。

启动复位之后，关于寄存器的内容还有两点需要考虑：

- **表 12-1** 中寄存器的修改与当前所选择的模式无关，对 **内部启动** 模式来说也是一样。

对于其它模式 – **外部启动** 和 **引导程序加载**（[章节 12.5](#)，[章节 12.6](#)）– 在启动过程中有更多寄存器被修改，具体见各模式的描述。

- 启动之后寄存器的值不仅受复位过程本身的影响，还受到其它事件（前一次启动期间甚至前一次启动之前的）的影响，例如紧急事件可改变时钟系统的状态。

因此，可能出现不同于上述值（以及复位之后的缺省寄存器值）的例外情况，这主要是一些时钟控制/状态标志。如需获得更多特殊情况的详细信息及处理，请参考 XE16xyM 编程指南。

**表 12-1      由启动过程设置的 XE16xyM 寄存器**

寄存器	值	注释
1. 每次启动之后：		
TRAPDIS	019F <sub>H</sub>	所有受 SCU 控制的强制中断被禁止，PET 和 RAT 除外

**启动配置和引导程序加载**

寄存器	值	注释
RSTCON1	UU: 10uu: U <sub>H</sub>	由 WDT 产生的内部应用复位请求
IMBCTRL	558C <sub>H</sub>	外部或带有 Flash 保护的引导程序加载模式
	A58C <sub>H</sub>	内部启动模式或 Flash 不受保护
R8...R15	XXXX	本地 Bank 1 中的 GPR – 启动过程中被用到

**2. 电源域 DMP\_1 上电之后:**

PLL_CON0	0F00 <sub>H</sub>	PLL 工作在正常模式, N 分频器 = 16
PLL_CON3	0007 <sub>H</sub>	K2 分频器 = 8
SYSCON0	0002 <sub>H</sub>	PLL 输出 (f <sub>PLL</sub> ) 用作系统时钟
WUOSCCON	0000 <sub>H</sub>	唤醒振荡器使能, f <sub>WU</sub> 大约 500 kHz
HPOSCCON	U: u0uu: UU <sub>H</sub>	在 OSCWDT 紧急情况下, 位 PLLSTAT.FINDIS 将不被置位
PLLOSCCON	XXXX <sub>H</sub>	与器件相关的特定值 (各系列芯片略有不同)
EVRMCON0	0100 <sub>H</sub>	EVR_M 控制 0 寄存器
EVR1CON0	0D00 <sub>H</sub>	EVR_1 控制 0 寄存器
EVR1SET15V HP	401B <sub>H</sub>	1.5V HP EVR_1 设置寄存器
EVR1SET10V	405B <sub>H</sub>	1.0V EVR_1 设置寄存器
EVR1SET15V LP	40DB <sub>H</sub>	1.5V LP EVR_1 设置寄存器
PVCMCON0	2544 <sub>H</sub>	Step0 PVC_M 控制寄存器
PVC1CON0	2544 <sub>H</sub>	Step0 PVC_1 控制寄存器

### 12.2.2 启动之后的系统频率

当执行第一条用户指令时，有效的系统时钟取决于当前所选择的启动模式以及最近一次的启动触发源：

- 上电之后除 CAN 引导程序加载（[章节 12.6.4](#)）之外的所有模式 – XE16xyM 的内部振荡器，10 MHz（标称值）（双倍频）；
- 上电之后，进行 CAN 引导程序加载（[章节 12.6.4](#)）– 连接到 XTAL 引脚的外部晶振频率，最低 4 MHz；
- 每次功能复位（非上电复位）之后 – 系统启动过程不改变时钟系统配置，相应的系统频率保持不变（和复位之前的系统频率相同）。

### 12.2.3 看门狗定时器的处理

XE16xyM 在启动过程中，看门狗定时器（WDT）始终被使能并配置为产生内部应用复位。

因此，用户软件必须：

- 程序若预知会使用 WDT，启动之后大约 65500 个系统时钟周期之内首次服务 WDT；
- 否则，在上述的时间段之内、但要在初始化结束（EINIT）之前禁用 WDT

当器件出现故障时，若该故障会导致用户软件不能正确执行、WDT 不能被定期服务，WDT 将产生复位请求。该复位引发器件重新启动，随后重新执行用户软件。

内部应用复位（这是 WDT 的缺省配置）不会影响到系统中的所有模块，请参见 SCU 一章中“模块复位行为”的描述。未受影响的模块的状态不变，因此，若该状态“错误”，内部应用复位不会使它恢复到“正确”的状态。因此，缺省的 WDT 配置及使用只能解决纯软件故障，它无法解决诸如时钟或电源系统中的故障。

通过  $\overline{\text{PORST}}$  引脚产生的复位请求可将 XE16xyM 中的所有模块带入一个已知的、功能正确的初始状态。因此，若实际应用要求 WDT 复位可确保器件正确重启，则必须采取特定的实现方式，具体描述见[章节 12.2.3.1](#)。

#### 12.2.3.1 由 WDT 触发上电复位

该特性要求  $\overline{\text{ESR1}}$  或  $\overline{\text{ERS2}}$  引脚专门用于触发上电复位，不可用作其它功能。

用户必须执行以下操作由 WDT 触发上电复位：

- 硬件方面：将选定的  $\overline{\text{ESRx}}$  引脚（ $x=1, 2$ ）连接到器件的  $\overline{\text{PORST}}$  引脚上；
- 软件方面：向相应寄存器  $\text{ESRCFGx}$ （ $x=1, 2$ ）的位[3:0]中写入 1110<sub>B</sub>。

*注：启动过程中所设置的 WDT 复位配置（即内部应用复位）保持不变。*

做好上述准备之后，包括由 WDT 触发在内的任意内部应用复位都将把选定的

ESRx 引脚拉低，从而导致上电复位，器件重启。

**注意：**采取上述解决方案时，内部应用复位不再是一种独立的复位类型。

*内部应用复位后，所有器件资源将被初始化，其初始化的方式和上电复位时完全相同，之前的所有信息将丢失。*

*因此，当通过 WDT 控制该特性且需要保持复位前的信息及状态时，不要将内部应用复位分配给其它复位请求源。*

有关该特性的详细内容，请参见应用指南 AP16146。

### 12.2.4 启动错误状态

启动过程中若出现了不可恢复的错误，为了避免对器件和/或系统造成不利的影响，XE16xyM 会进入一个稳定的、被动的、中性的状态 – 即 DMP\_1 关闭、DMP\_M 由 1V 供电的省电模式。

该状态只能通过上电复位退出。

## 12.3 支持的启动模式和选择

XE16xyM 支持各种启动模式，允许用户在三个方面进行选择：

- 主要功能 – 从哪里开始执行用户代码（片上 Flash、PSRAM 还是外部存储器）；
- 可选性 – 启动之前将初始代码下载到 PSRAM 的方式：
  - 通过通信接口从外部主机下载 – UART、CAN、SSC
- 与调试相关的 - 能否进行调试，如果可以 – 使用哪个调试接口（JTAG、DAP，可选的引脚分配）

根据上述不同的选项，XE16xyM 的启动模式可被分成几组，相关描述见 [章节](#)

[12.3.1](#)、[章节 12.3.2](#) 和 [章节 12.3.3](#)。

### 12.3.1 基本启动模式

这些模式（请参考表 12-2）不支持调试和特殊特性。

**表 12-2 XE16xyM 的基本启动模式**

启动模式	STSTAT.HWC FG 值 <sup>1)</sup>	配置引脚 <sup>2)</sup>							
		TRST	P10 [6 : 0]						
从 Flash 进行内部启动	00 <sub>H</sub>	0	x	x	x	x	x	x	x
UART 引导加载器 2.x <sup>3)</sup>	02 <sub>H</sub>	1	x	x	x	x	0	1	0
UART 引导加载器 7.x <sup>4)</sup>	06 <sub>H</sub>	1	x	x	x	x	1	1	0
SSC 引导加载器	09 <sub>H</sub>	1	x	x	x	1	0	0	1
CAN 引导加载器	0D <sub>H</sub>	1	x	x	x	1	1	0	1
UART 增强引导加载器 2.x <sup>3)</sup>	10 <sub>H</sub>	1	0	0	1	0	0	0	0
外部启动 <sup>5)</sup>	70 <sub>H</sub>	1	1	1	1	0	0	0	0

1) 可从端口 P10 或从寄存器 SWRSTCON 中的位域 SWCFG 位域加载 HWCFG。

2) x 代表与相应引脚上的电平无关。

3) 2.x 代表：P2.3 引脚上的 TxD（发送数据），P2.4 引脚上的 RxD（接收数据）

4) 7.x 代表：P7.3 引脚上的 TxD（发送数据），P7.4 引脚上的 RxD（接收数据），在 64 引脚封装器件中不可用。

5) 在 64 引脚封装中不可用。

XE16xyM 在不同模式-内部启动、外部启动、引导程序加载 – 下的功能，分别在章节 12.4，章节 12.5 和章节 12.6 中作详细描述。

### 0 引脚配置

该特性是 XE16xyM 的新特性，意味着不使用任何通用输入/输出（GPIO）引脚 – 包括分配给一个引脚的任何复用功能 – 来选择最常用的启动配置，即从 Flash 进行内部启动。

检查专用引脚 TRST 的值，如下所述：

- 复位期间 TRST = 0 – 不评估其它引脚（也不评估端口 P10），从内部 Flash 存储器启动用户代码 – 请参考表 12-2 第一行。



- 复位期间  $\overline{\text{TRST}} = 1$  – 评估端口 P10 的一些引脚，进一步决定器件功能。用于评估的 P10 引脚个数从 3 到 7 不等，参见表 12-2 和表 12-3。

### 12.3.2 支持调试的启动模式

这些启动选择（请参考表 12-3）使得用户代码从内部 Flash 或外部存储器启动。因此，从功能上看，这些选择与表 12-2 的基本启动模式中的两项相似，但是本节中的附加选项还支持与外部工具（调试器）相连，在开发工程中可使用这些工具。

表 12-3 支持调试功能的 XE16xyM 的启动模式

启动模式	调试接口	STSTAT.HW CFG 值 <sup>1)</sup>	CFG 引脚 P10 [6:0] <sup>2)</sup>							
			$\overline{\text{TRST}} = 1$							
从 Flash 进行 内部启动	JTAG pos.B	03 <sub>H</sub>	x	x	x	x	0	1	1	
	DAP pos.1	04 <sub>H</sub>	x	x	x	x	1	0	0	
	从 Flash <sup>3)</sup>	07 <sub>H</sub>	x	x	x	x	1	1	1	
	DAP pos.0	01 <sub>H</sub>	x	x	x	0	0	0	1	
	DAP pos.2 <sup>4)</sup>	05 <sub>H</sub>	x	x	x	0	1	0	1	
	JTAG pos.C <sup>4)</sup>	40 <sub>H</sub>	1	0	0	0	0	0	0	
	JTAG pos.D <sup>5)</sup>	50 <sub>H</sub>	1	0	1	0	0	0	0	
外部启动 <sup>4)</sup>	从 Flash <sup>3)</sup>	60 <sub>H</sub>	1	1	0	0	0	0	0	

1) 可从端口 P10 或从寄存器 SWRSTCON 中的位域 SWCFG 位域加载 HWCFG。

2) x 代表与相应引脚上的电平无关。

3) Flash 内确定的地址 (C0'01F0<sub>H</sub>) 必须包含 DBGPRR 寄存器的值 (2 个字节)，下一个字地址 (C0'01F2<sub>H</sub>) 必须包含对应的取反值。

来自 Flash 的 16 位值的高 4 位是不予关心的 – 它们由启动过程处理。

如果不满足取反的要求 – 值被认为是无效的，缺省配置位置 A 处的 JTAG 引脚。

4) 在 64 引脚封装器件中不可用。

5) 在 64 引脚和 100-引脚封装器件中不可用。

在此情况下，各种启动配置（表 12-3）用于选择调试类型（DAP 或 JTAG）和使能的调试接口的引脚位置，– 选择通过 P10 引脚或 SWRSTCON.HWCFG 位域完成– 的选项除外。调试接口处理还有两个基本条件：

**启动配置和引导程序加载**

- 调试接口配置作为整个器件启动配置的一部分。因此只有以某些方式（描述见 [章节 12.1](#)）更新启动配置（如启动模式选择）时，才能进行调试接口使能/禁止 /（重新）配置的操作。

例如，由硬件（例如 WDT）触发应用复位时，不应改变调试接口。

- 如果选择内部启动，且片上 Flash 受保护，则调试接口将始终被禁止。

不同调试接口选择的准确意义 – 接口类型和引脚分配 - 见 [表 12-4](#)。该表的最后一列给出配置值，如果已经选择从 Flash 进行调试接口配置，则必须将该值写入到 Flash 地址 C0 01F0<sub>H</sub>处（参考[表 12-3](#)）。

**表 12-4      XE16xyM 调试选择：接口类型和引脚分配**

调试接口		用于调试的引脚		Flash 地址 C0 01F0 <sub>H</sub> 处的 DBGPRR 值
类型	位置	主要接口 (必选)	BRKIN (可选)	
DAP	pos.0	P2.9, P7.0	P5.10	1000 <sub>H</sub>
	pos.1	P10.9, P10.12	P10.8	1155 <sub>H</sub>
	pos.2 <sup>1)</sup>	P7.0, P7.4	P7.1	12AA <sub>H</sub>
JTAG	pos.A	P2.9, P5.2, P5.4, P7.0	P5.10	1800 <sub>H</sub>
	pos.B	P10.9, P10.10, P10.11, P10.12	P10.8	1955 <sub>H</sub>
	pos.C <sup>1)</sup>	P7.0, P7.2, P7.3, P7.4	P7.1	1AAA <sub>H</sub>
	pos.D <sup>2)</sup>	P8.3, P8.4, P8.5, P10.12	P8.6	1BFF <sub>H</sub>
不可用	---	---	---	0000 <sub>H</sub>

1) 在 64 引脚封装中不可用。

2) 在 64 引脚和 100 引脚封装中不可用。

与调试相关的接口信号/引脚分为两种类型：

### 主调试接口

[表 12-4](#) 的第三列列出 2（对应 DAP）或 4 个引脚（对应 JTAG 接口）的分配。

如果调试被使能，这些引脚始终被分配给调试接口，因此应用软件一定不能使用其中任何一个。

### 可选的断点接口

XE16xyM 调试系统的断点接口包括两个信号： $\overline{\text{BRKIN}}$  和  $\overline{\text{BRKOUT}}$ 。

可选择使用该接口，也可选择使用两个信号中的任何一个或者两个信号都使用。

使用断点接口需要进行附加的准备工作 - 当请求时 - 一旦主接口可用，由外部调试器进行完成。

只要未完成该准备工作且“断点输入 Break-In”/“断点输出 Break-Out”特性未被激活，则主机选择的（潜在的） $\overline{\text{BRKIN}}$  /  $\overline{\text{BRKOUT}}$  信号仍可被应用软件用作其它功能。

两个断点信号/引脚的处理方式不同：

- $\overline{\text{BRKIN}}$  - 作为该用途的引脚只能由启动选择决定 - 请参考表 12-4 的  $\overline{\text{BRKIN}}$  一列
- $\overline{\text{BRKOUT}}$  - 启动期间，无需为该信号选择引脚

可用作  $\overline{\text{BRKOUT}}$  的一些引脚 - 在激活“断点输出 Break-Out”特性之前，由外部工具进行选择：

- P6.0;
- P1.5 – 在 64 引脚封装中不可用；
- P9.3 – 在 64 引脚和 100 引脚封装中不可用；
- P10.11 – 如果选择 JTAG 接口位置 B，则不能使用。

### 12.3.3 特殊启动特性

XE16xyM 支持一些特殊特性，允许用户软件影响器件的启动，提供不同于上述内容（[章节 12.3.1](#) 和 [章节 12.3.2](#)）的附加功能。

**注意：**正确使用这些特性需要对 XE16xyM 结构、行为和编程有良好和详细的理解。特殊的启动特性是为专门为高级用户提供的，要求用户熟悉器件整体，尤其是对系统控制单元了解透彻，不管从硬件（见本用户手册 SCU 章节）还是软件控制方面（描述见 XE16xyM 编程指南）。

#### 12.3.3.1 启动信息补充

特殊启动特性需要从应用软件获取或向应用软件提供附加信息，使用系统控制单元内的专用寄存器 STMEM0 实现此功能。

#### STMEM0 寄存器

SCU\_STMEM0 寄存器位于 DMP\_M 电源域，受到安全机制的保护。

使用该寄存器和用户软件交换下述启动信息：

- 用户软件可通过设置 STMEM0 的位[15:6]影响器件的下次启动；

所支持特性的描述见[章节 12.3.3.2](#)。

- 如果启动之后 **STMEM0[4] = 0** – 则该启动过程由功能（如应用或内部应用）复位触发；

在此情况下，由 **SCU\_SYSCON0** 内的位[15:12]指示紧急状态标志，用户程序可从 **SCU\_STMEM0** 中的位[3:0]读取：

- 位[0] – **OSCWDT** 紧急事件源状态
- 位[1] – **VCOLCK** 紧急事件源状态
- 位[2] – **PVC1** 紧急事件源状态
- 位[3] – 时钟选择状态

- 如果启动之后 **STMEM0[4] = 1** – 则启动操作由上电操作触发；

在此情况下，由 **SCU\_STMEM0[3]**位提供附加信息。

- **STMEM0[3] = 0**– 仅 **DMP\_1** 域进行上电；
- **STMEM0[3] = 1**– **DMP\_1** 和 **DMP\_M** 域都上电

## STMEM0

启动存储器0寄存器

**ESFR (F0A0H/50H)**

复位值: **0000H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>USSET</b>	<b>0</b>	<b>RINDP</b>	<b>RINDS</b>	<b>RINPS</b>	<b>0</b>				<b>0</b>	<b>0</b>	<b>STSIND</b>				
rw	rw	rw	rw	rw	rw				rw	rw	rw				

符号	位序号	类型	功能描述
<b>STSIND</b>	[4:0]	rw	<b>启动状态指示：</b> <i>注：此处未描述的值保留不用。</i> <b>0xxxx<sub>B</sub></b> 功能复位 – 位[3:0]指示状态标志（请参阅文字描述） <b>10000<sub>B</sub></b> 仅电源域 <b>DMP_1</b> 上电 <b>11000<sub>B</sub></b> 电源域 <b>DMP_1</b> 和 <b>DMP_M</b> 均上电
<b>0</b>	5	rw	<b>保留，必须写入复位值 0</b>
<b>0</b>	6, [10:7]	rw	<b>保留，必须写入复位值 0</b>
<b>RINPS</b>	11	rw	<b>PSRAM 初始化</b> <b>0</b> 未请求

符号	位序号	类型	功能描述
			1 启动之后执行
<b>RINDS</b>	12	rw	<b>DSRAM 初始化</b> 0 未请求 1 启动之后执行
<b>RINDP</b>	13	rw	<b>DPRAM 初始化</b> 0 未请求 1 启动之后执行
<b>0</b>	14	rw	<b>保留，必须写入复位值 0</b>
<b>USSET</b>	15	rw	<b>启动之后的 RAM 初始化</b> 0 未请求 1 在 STMEM0[13:11]中请求

### 12.3.3.2 准备激活存储器内容保护

XE16xyM 支持两种保护存储器内容的机制：ECC（纠错码）和奇偶校验，缺省状态下，两者都被禁止。

只能通过用户软件激活各保护机制，执行图 12-1 所示的序列：

- 器件上电之后（由 **STMEM0**[4] = 1，**STMEM0**[15]在此情况下将为 0）– 需要进行 RAM 初始化：
  - 可选：若实际应用的系统时钟频率高于 10 MHz（上电之后的系统频率）– 此时仍能重新配置时钟，使用更高的频率进行 RAM 初始化。
  - 若某些存储器将使用奇偶校验 – 向 **SCU\_MCHKCON** 寄存器的相应位中写入 1（缺省选择 ECC 用作存储器内容保护）
  - 通过 **STMEM0**[15:11] = 10111<sub>B</sub> 设置 RAM 初始化请求；  
也可只选择设置部分位[13:11]，激活相应存储器的内容保护（参见寄存器 **STMEM0** 的描述和图 12-1）。该操作并不会使启动过程加快太多 – 因为所有存储器均并行初始化，因此初始化一个或所有 RAM 存储器所需的时间差别不大。
  - 触发应用复位以重新启动器件  
在此次器件启动过程中，根据 **STMEM0**[13:11]所设置的请求进行 RAM 初始化。
- 若启动之后 **STMEM0**[15] = 1 – 意味着 RAM 已经完成初始化。
  - 清除 RAM 初始化请求 – **STMEM0**[15:11] = 00000<sub>B</sub>；

- 根据应用需要配置/使能 ECC/奇偶校验;
- 继续进行后续的系统初始化（如有需要）并启动应用;
- 若功能复位（非上电复位）之后 **STMEM0[15] = 0** – 不需进行 RAM 初始化，请求无效：
  - 根据应用需要配置/使能 ECC/奇偶校验 – 需要进行此操作，因为每次启动都会复位某些控制寄存器;
  - 继续进行后续的系统初始化（如有需要）并启动应用。

从初始化之后的存储器中读取数据不会产生错误，给出的数据为：

- 如果 ECC 有效 – 0600<sub>H</sub>;
- 如果奇偶校验有效 – 3000<sub>H</sub>。

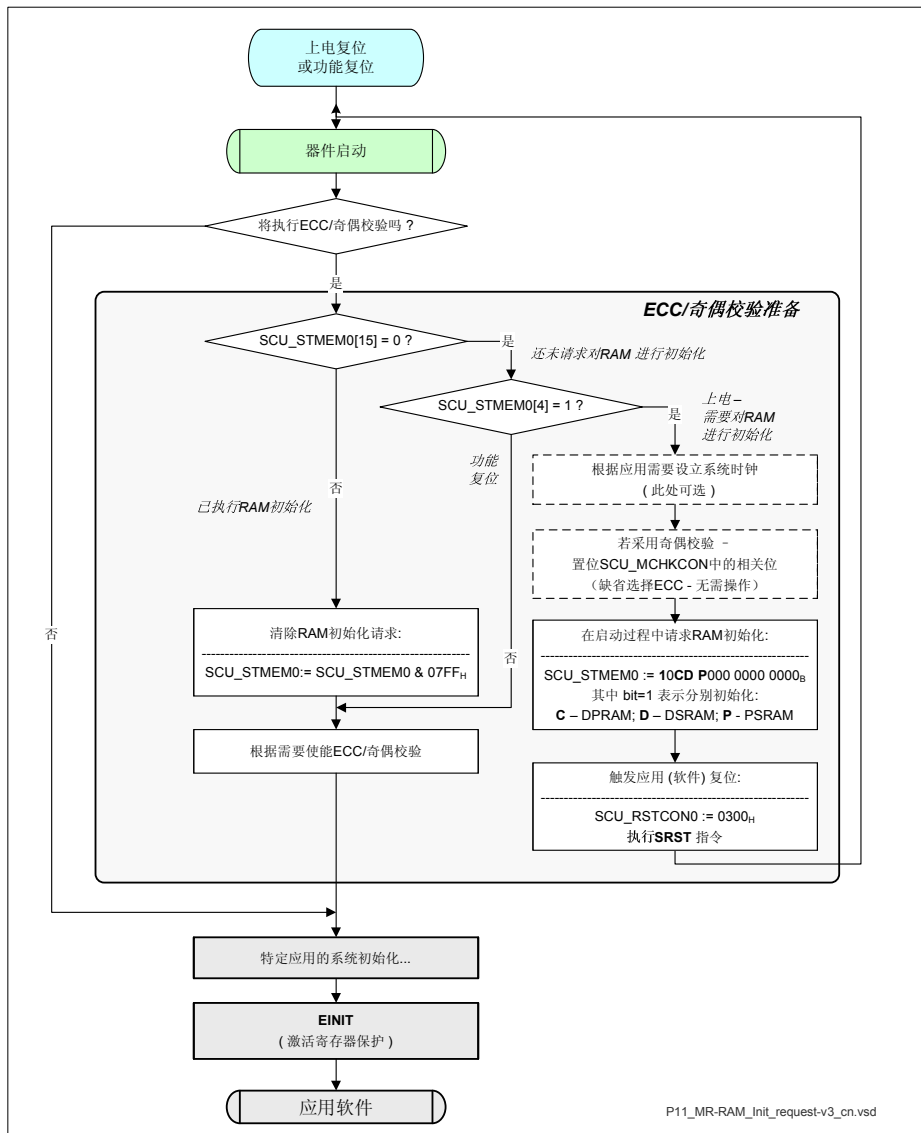


图 12-1 准备采用 ECC/奇偶校验的软件序列

## 12.4 内部启动

当配置为内部启动模式时，XE16xyM 立即开始执行片上 Flash 存储器中的代码（第一条指令从地址 C0' 0000<sub>H</sub> 开始）。

由于大多数情况下期望使用的是不支持调试的内部启动模式，因此可通过仅将专用引脚  $\overline{\text{TRST}}$ （如对于应用来说不可用的）拉低来选择该模式，称之为 **0-引脚配置**。

如果需要调试支持 – 则可使用附加的配置选择，参见 **章节 12.3.2**。

*注：在未禁止保护的情况下，从内部模式启动的应用可以读取受到读保护的 Flash。*

## 12.5 外部启动

当配置为外部启动模式时，XE16xyM 开始从片外存储器执行代码（第一条指令位于 00' 0000<sub>H</sub> 处），片外存储器与 XE16xyM 的外部总线接口相连。

**注意：64 引脚封装器件不支持外部启动模式。**

通过评测附加的配置引脚，使外部总线控制器和所选择的片外存储器匹配。

使用 P10 口的 7 个引脚选择 EBC 模式（P10.[10:8]）、地址宽度（P10.[12:11]）以及片选线的个数（P10.[14:13]）。具体选择归纳见下表。

**表 12-5 EBC 配置：EBC 模式**

EBC 启动模式	配置引脚 P10[10:8]			EBC 所使用的引脚
8 位数据，复用	0	0	0	P2.0 ... P2.2, P10.0 ... P10.15
8 位数据，非复用	0	0	1	P0.0 ... P0.7, P1.0...P1.7, P2.0 ...P2.2, P10.0 ... P10.7, P10.13, P10.14
16 位数据，复用， $\overline{\text{BHE}}$ 模式	0	1	0	P2.0 ... P2.2, P2.11, P10.0 ...P10.15
16 位数据，复用， $\overline{\text{WRH}}$ 模式	0	1	1	P2.0 ... P2.2, P2.11, P10.0 ...P10.15
16 位数据，非复用， $\overline{\text{BHE}}$ 模式，A0	1	0	0	P0.0 ... P0.7, P1.0 ...P1.7, P2.0 ...P2.2, P2.11, P10.0 ...P10.14
16 位数据，非复用， $\overline{\text{WRH}}$ 模式，A0	1	0	1	P0.0 ... P0.7, P1.0 ...P1.7, P2.0 ...P2.2, P2.11, P10.0 ...P10.14



EBC 启动模式	配置引脚 P10[10:8]			EBC 所使用的引脚
16 位数据，非复用， $\overline{\text{BHE}}$ 模式，A1	1	1	0	P2.0 ... P2.2, P10.0 ... P10.15
16 位数据，非复用， $\overline{\text{WRH}}$ 模式，A1	1	1	1	P0.0 ... P0.7, P1.0 ... P1.7, P2.0 ... P2.2, P10.0 ... P10.7, P10.13, P10.14

**表 12-6      EBC 配置：地址宽度**

可用地址线	配置引脚 P10[12:11]		附加地址引脚
A15 ... A0	0	0	无
A17 ... A0	0	1	P2.3, P2.4
A19 ... A0	1	0	P2.3 ... P2.6
A23 ... A0	1	1	P2.3 ... P2.10

**表 12-7      EBC 配置：片选线**

可用片选线	配置引脚 P10[14:13]		使用的引脚
$\overline{\text{CS0}} \dots \overline{\text{CS4}}$	0	0	P4.0 ... P4.4
CS0	0	1	P4.0
CS0 ... CS1	1	0	P4.0, P4.1
无	1	1	无

### 12.5.1 特定设置

当 XE16xyM 进入外部启动模式后，根据**表 12-8** 和**表 12-9** 自动进行以下配置。

需要注意，启动过程不配置 ADDRSELx 寄存器中的任意地址窗。因此，即使配置某些  $\overline{\text{CS}}$  信号（参见**表 12-7**），启动过程也只进行适当的设置以确保所选 EBC 模式所需的引脚功能正确。用户软件必须谨慎处理：

- 为将要使用的  $\overline{\text{CSx}}$  引脚配置地址窗（在 ADDRSELx 寄存器中）
- 通过设置 FCONCSx.ENCS 使能这些引脚

**表 12-8 外部启动模式 – EBC 寄存器中的特定状态**

<b>P10[10:8]上的配置</b>	<b>EBCMOD0 [15:8]</b>	<b>EBCMOD1</b>	<b>FCONCSx<sup>1)</sup></b>	<b>注 (EBC 模式)</b>
000 <sub>B</sub>	30 <sub>H</sub>	001F <sub>H</sub>	0011 <sub>H</sub>	8 位复用
001 <sub>B</sub>	70 <sub>H</sub>	0020 <sub>H</sub>	0001 <sub>H</sub>	8 位非复用
010 <sub>B</sub>	40 <sub>H</sub>	0000 <sub>H</sub>	0031 <sub>H</sub>	16 位复用, $\overline{\text{BHE}}$
011 <sub>B</sub>	48 <sub>H</sub>	0000 <sub>H</sub>	0031 <sub>H</sub>	16 位复用, $\overline{\text{WRH}}$
100 <sub>B</sub>	60 <sub>H</sub>	0000 <sub>H</sub>	0021 <sub>H</sub>	16 位非复用, $\overline{\text{BHE}}$ , A0
101 <sub>B</sub>	61 <sub>H</sub>	0000 <sub>H</sub>	0021 <sub>H</sub>	16 位非复用, $\overline{\text{WRH}}$ , A0
110 <sub>B</sub>	60 <sub>H</sub>	0010 <sub>H</sub>	0021 <sub>H</sub>	16 位非复用, $\overline{\text{BHE}}$ , A1
111 <sub>B</sub>	61 <sub>H</sub>	0010 <sub>H</sub>	0021 <sub>H</sub>	16 位非复用, $\overline{\text{WRH}}$ , A1

1) 影响哪个 FCONCSx 取决于 P10[14:13]的设置:

11<sub>B</sub> 或 01<sub>B</sub> – 影响 FCONCS0

10<sub>B</sub> – 影响 FCONCS1

00<sub>B</sub> – 影响 FCONCS0 ... FCONCS4

其它 (不受影响的) FCONCS 寄存器保持其缺省值, 参见 EBC 章节。

**表 12-9 外部启动模式 – EBCMOD0[7:0]中的特定状态**

<b>P10[14:13]上的配置</b>	<b>在 P10[12:11]上的配置</b>			
	<b>00<sub>B</sub> (0 段)</b>	<b>01<sub>B</sub> (2 段)</b>	<b>10<sub>B</sub> (4 段)</b>	<b>11<sub>B</sub> (8 段)</b>
00 <sub>B</sub> (5 CS)	50 <sub>H</sub>	52 <sub>H</sub>	54 <sub>H</sub>	58 <sub>H</sub>
01 <sub>B</sub> (1 CS)	10 <sub>H</sub>	12 <sub>H</sub>	14 <sub>H</sub>	18 <sub>H</sub>
10 <sub>B</sub> (2 CS)	20 <sub>H</sub>	22 <sub>H</sub>	24 <sub>H</sub>	28 <sub>H</sub>
11 <sub>B</sub> (0 CS)	00 <sub>H</sub>	02 <sub>H</sub>	04 <sub>H</sub>	08 <sub>H</sub>

## 12.6 引导程序加载

引导程序加载是一项通过特定接口（通常是串口）将代码传送至 XE16xyM 的技术。从非易失性程序存储器执行常规代码之前，XE16xyM 执行先前接收的代码。

启动代码可能是完整的（例如，测试或校准用的临时软件），修补非易失性存储器中现有的代码（如特定产品的数据或程序）或者加载附加代码（如使用更高级或更安全的协议）。引导程序加载的一种可能的应用是在生产下线时编程空白的 Flash 存储器，初始化代码不需要外部存储器或内部 Flash。

BSL 机制可用于标准的系统启动，也可仅用于特殊的场合，如系统维护（固件更新）或下线前编程或测试。

XE16x 采用多种协议/模式支持引导程序加载：

- 标准 UART 协议，加载 32 个字节（见[章节 12.6.2.1](#)）
- UART 协议，增强引导程序加载，可传送任意个字节（见[章节 12.6.2.2](#)）
- 同步串行协议（见[章节 12.6.3](#)）
- CAN 协议（见[章节 12.6.4](#)）

这些模式归纳见[表 12-16](#)。

### 12.6.1 一般功能

尽管每种引导程序加载都有其独特的功能，但处理方式一般是相同的。

#### 进入引导程序加载

通过选择特定的启动配置使能引导程序加载（见[章节 12.1](#)）。

在[表 12-16](#)中描述引导程序加载所需的配置，总结见[表 12-2](#)。

#### 加载启动代码

建立通信之后，BSL 进入到一个循环中，接收相应个数的字节。这些字节顺序保存 to 片上 PSRAM 中，从地址 E0' 0000<sub>H</sub> 开始。为了执行加载的代码，之后 BSL 使寄存器 VECSEG 指向地址 E0' 0000<sub>H</sub>，即要加载的第一条指令处，然后跳转到该指令。

加载的代码可能是最后的应用代码或是其它更为复杂的程序，该加载程序添加了传送协议，从而增强了加载代码或数据的一致性，还可能包含改变系统配置并使能总线接口，从而将接收到的数据保存在外部存储器中的代码序列。

该过程可能经过几次反复或可能直接在最后应用中执行。

#### 退出引导加载器模式

只要引导程序加载器工作，则看门狗定时器和调试系统都会被禁止。当从主机接收到最后一个字节之后，BSL 终止时，看门狗定时器和调试系统才会被自动释放。

如果使用第二级加载器，加载程序应当通过指令 DISWDT 使得看门狗定时器无效，从而延长下载周期。

非 BSL 复位之后，按照外部配置，XE16xyM 将从用户存储器开始执行代码。

## 与主机的接口

引导程序加载器通过预先定义的一组接口引脚与外部主机通信。这些接口引脚被自动使能，并由引导程序加载器进行控制。主机必须和这些预先定义好的接口引脚相连。

**表 12-16** 给出每种引导程序加载模式使用的接口引脚。

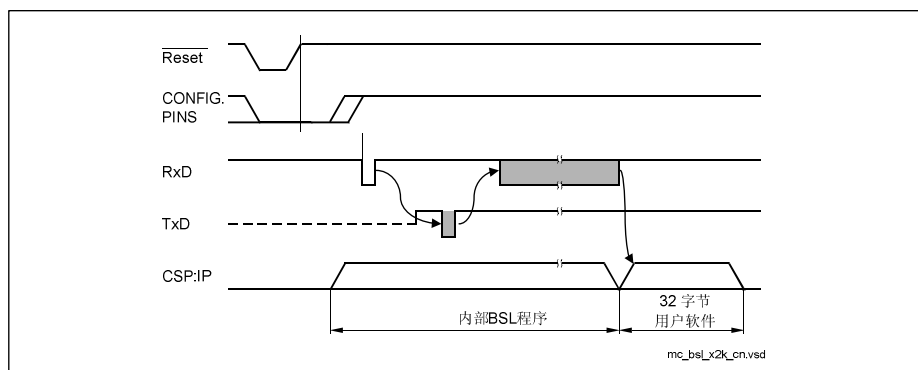
## 12.6.2 使用 UART 协议进行引导程序加载

在 XE16xyM 中，用户可以选择几种基于 UART（通用异步接收和发送）协议的通信方式下载代码/数据。

### 12.6.2.1 标准 UART 引导程序加载

标准 UART 引导程序加载器通过 USIC0 通道 0（U0C0）将程序代码/数据传送到 PSRAM 中。发送 ID 字节之后才能使能 U0C0 接收。与主机进行半双工连接，因此完全满足 BSL 的要求。

使用不带奇偶校验的异步 8 位数据帧（1 个起始位，1 个停止位）将数据从外部主机传送至 XE16xyM。标准 UART 引导模式中，要接收的数据字节个数固定为 32 个，可由多达 16 条双字节指令组成。



**图 12-2 引导程序加载序列**

## 启动配置和引导程序加载

进入 UART BSL 模式并完成相应的初始化之后，XE16xyM 扫描 RxD 线，接收一个零字节。零字节被认为是包含 1 个起始位，8 个 0 数据位以及 1 个停止位。在该零字节传送期间，XE16xyM 以当前 CPU 时钟为单位，计算相应的波特率因子，相应的初始化串行接口 U0C0 并将引脚 TxD 切换到输出。并使用该波特率，向提供加载数据的主机返回一个 ID 字节 (D5<sub>H</sub>)。

一旦发送 ID 字节，BSL 进入一个循环，通过 U0C0 接收 32 个字节。这些字节顺序保存到内部 PSRAM 地址 E0' 0000<sub>H</sub> 到 E0' 001F<sub>H</sub> 处，然后开始执行这些指令。

*注：要加载更多的代码，有两种方法可供选择：*

- 通过第二级加载 – 见下文。
- 使用**增强 UART 引导程序加载** – 参见[章节 12.6.2.2](#)。

## 二级引导程序加载

最初加载的程序很可能需要加载附加的代码或数据，因为大多数应用可能将需要后续的多于 32 个字节的指令。第二级接收循环可直接使用已经初始化的 U0C0 接口，接收数据并将其保存到任意用户指定的地址处。

下面的样例代码给出如何将第二级程序加载器放入到可用的 32 字节中，这种方式之所以可行是因为使用了预先初始化的串行通道和预先设置的寄存器（见[表 12-10](#)）。

```
;Example for Secondary UART Bootstrap Loader Routine
;-----
TargetStart LIT '0E00020H'      ;Definition of target area:
TargetEnd LIT '0E001FFH'        ;480 bytes in this example
StartOfCode LIT '0E00100H'      ;Continue executing here...
                                ;...after download

Level2Loader:
DISWDT                          ;No WDT for further download
MOV DPP0, #(PAG TargetStart)
MOV R10, #(DPP0:TargetStart)    ;Set pointer to target area
Level2MainLoop:
MOV [R1], R3                    ;Clear RIF for new byte
Level2RecLoop:
MOV R4, [R0]                    ;Access PSR
JNB R4.14, Level2RecLoop        ;Wait for RIF
MOVB [R10], [R2]                ;Copy new byte to target
CMPIL R10, #POF (TargetEnd)     ;All bytes received??
JMPL cc_NE, Level2MainLoop      ;Repeat for complete area
Level2Terminate:
JMPL SEG StartOfCode, SOF StartOfCode
```

### 特定设置

当 XE16xyM 进入标准 UART BSL 模式时，自动进行以下配置。

**表 12-10 标准 UART BSL 的特定状态**

配置项	值	注释
U0C0_CCR	0002 <sub>H</sub>	USIC0 通道 0 选择 ASC 模式
U0C0_PCRL	0401 <sub>H</sub>	1 个停止位，在步骤 4 三次采样 RxD 引脚电平
U0C0_SCTRL	0002 <sub>H</sub>	被动数据电平 = 1
U0C0_SCTRH	0707 <sub>H</sub>	8 个数据位
U0C0_FDRL	43FF <sub>H</sub>	选择正常分频模式 1: 1
U0C0_BRGH	0XXX <sub>H</sub>	测量 PDIV 值（零字节），在位[9:0]中
U0C0_BRGL	1C00 <sub>H</sub>	正常模式，FDIV，8 个时钟/位
U0C0_DX0CR	0003 <sub>H</sub>	数据输入选择
DPP1	0081 <sub>H</sub>	指向 USIC0 基地址 <sup>1)</sup>
R0	4044 <sub>H</sub>	指向 U0C0_PSR <sup>1)</sup>
R1	4048 <sub>H</sub>	指向 U0C0_PSCR <sup>1)</sup>
R2	405C <sub>H</sub>	指向 U0C0_RBUF <sup>1)</sup>
R3	4000 <sub>H</sub>	屏蔽，以清除 RIF <sup>1)</sup>
在 144/100 引脚封装的器件中：		
P7_IOCRO3	00B0 <sub>H</sub>	P7.3 为推挽输出（TxD）
P7_IOCRO4	0020 <sub>H</sub>	P7.4 为带上拉的输入（RxD）
在 64 引脚封装的器件中：		
P2_IOCRO3	00B0 <sub>H</sub>	P2.3 为推挽输出（TxD）
P2_IOCRO4	0020 <sub>H</sub>	P2.4 为带上拉的输入（RxD）

1) 该寄存器的设置由第二级引导加载程序提供（见[页 12-19](#)）。

ID 字节确定要启动的器件。定义下述代码：

55<sub>H</sub>: 8xC166

A5<sub>H</sub>: C167 的先前版本（旧的）

B5<sub>H</sub>: C165 的先前版本

C5<sub>H</sub>: C167 衍生系列

D5<sub>H</sub>: 所有配备有 ID 寄存器的器件（包括 XE16xyM）

*注：ID 字节 D5<sub>H</sub> 不直接指定特定衍生产品型号。在此情况下，可从 ID 寄存器获得器件 ID 信息。*

### 12.6.2.2 增强 UART 引导加载

增强 UART 引导加载器通过 USIC0 模块的通道 0（U0C0）将程序代码/数据传送到 PSRAM 中。

使用不带奇偶校验位的异步 8 位数据帧（1 个启动位，1 个停止位）将数据从外部主机传送至 XE16xyM。与**标准 UART 引导程序加载**不同，代码/数据的长度不固定，可将任意字节的数据传送至 PSRAM（最多可达 PSRAM 容量减去 256 字节）。也可从 PSRAM 任意地址处开始执行代码，还可以改变初始的波特率 - 例如，传送较长代码/数据块时使用更高的波特率。

该模式的最初几步与**标准 UART 引导程序加载**相同。XE16xyM 首先扫描 RxD 线，接收一个零字节（即 1 个起始位，8 个 0 数据位和 1 个停止位）。零字节传送过程中，XE16xyM 计算相应的波特率因子（以当前 CPU 时钟为单位），相应的初始化串行接口 U0C0，并将 TxD 切换到输出。使用该波特率，将 ID 字节（DA<sub>H</sub>）返回给主机。

使用该模式的下述几个步骤处理处理称为引导加载器报文头：

1. XE16xyM 发送从 U0C0\_BRHG 寄存器中得到的当前 PDIV 分频器设置 - 10 位值，用 2 个字节表示；

*注：该引导加载模式中，多字节值的发送顺序是从高到低。*

2. XE16xyM 接收并将报文头\_代码（1B）发回给主机；
3. XE16xyM 接收要传送的代码\_长度（3B）并发回给主机；
4. XE16xyM 接收代码执行的初始地址 STADD（3B）并发回给主机；
  - 对 XE16xyM 来讲，段地址（STADD 最高字节）必须等于 E0H
5. XE16xyM 接收 PDIV 分频器值（2B，仅位[9:0]有效）并发回给主机；
  - 如果新值与当前值不同 - 新值被写入到 U0C0\_BRGH 寄存器，使用已经改变的波特率将零确认字节发回给主机。

6. XE16xyM 接收尾\_代码（1B）并发回给主机；

a) 如果头\_代码和尾\_代码都等于 XE16xyM ID 字节（DA<sub>H</sub>），引导加载器向主机发送一个零字节并继续进行后续操作。

## 启动配置和引导程序加载

b) 如果上述条件不为真 – 引导加载器发送 ID 字节 (DA<sub>H</sub>) 给主机并从第 1 点重新启动报文头处理过程。

一旦根据上述步骤成功处理报文头，引导程序加载器接收代码\_长度字节并顺序保存在 PSRAM 中，从地址 E0' 0000<sub>H</sub> 开始。

**注意：用户必须注意，发送的字节数一定不能大于器件 PSRAM 的可用容量减去 256 字节所得的值。**

接收并保存最后一个字节之后，引导程序加载器启动代码的执行。从接收的报文头中的地址 STADD 处开始执行代码。

### 特定设置

当 XE16xyM 进入增强 UART BSL 模式时，自动进行以下配置。

**表 12-11 增强 UART BSL 的特定状态**

配置项	值	注释
U0C0_CCR	0002 <sub>H</sub>	USIC0 通道 0 选择 ASC 模式
U0C0_PCRL	0401 <sub>H</sub>	1 个停止位，在步骤 4 三次采样 Rx D 引脚电平
U0C0_SCTRL	0002 <sub>H</sub>	被动数据电平 = 1
U0C0_SCTRH	0707 <sub>H</sub>	8 个数据位
U0C0_FDRL	43FF <sub>H</sub>	选择正常分频模式 1: 1
U0C0_BRGH	0XXX <sub>H</sub>	PDIV-值，等于主机在报文头中发送的值
U0C0_BRGL	1C00 <sub>H</sub>	正常模式，FDIV，8 个时钟/位
U0C0_DX0CR	0003 <sub>H</sub>	数据输入选择
在 144/100 引脚封装的器件中：		
P7_IOCRO3	00B0 <sub>H</sub>	P7.3 为推挽输出 (Tx D)
P7_IOCRO4	0020 <sub>H</sub>	P7.4 为带上拉的输入 (Rx D)
在 64 引脚封装的器件中：		
P2_IOCRO3	00B0 <sub>H</sub>	P2.3 为推挽输出 (Tx D)
P2_IOCRO4	0020 <sub>H</sub>	P2.4 为带上拉的输入 (Rx D)



ID 字节指示要启动的器件。XE16xyM 为支持增强 UART BSL 模式的第一个微控制器系列，ID 定义为 DA<sub>H</sub>。

*注：ID 字节不直接指定特定衍生产品的型号。在此情况下，可从 ID 寄存器获得器件的 ID 信息。*

### 12.6.2.3 选择 BSL 的波特率

可从接收到的第一个零字节的长度计算 U0C0 的串行波特率，允许 XE16xyM 在很宽的波特率范围内进行引导程序加载操作。然而为了确保正确的数据传送操作，必须满足上限和下限的要求。

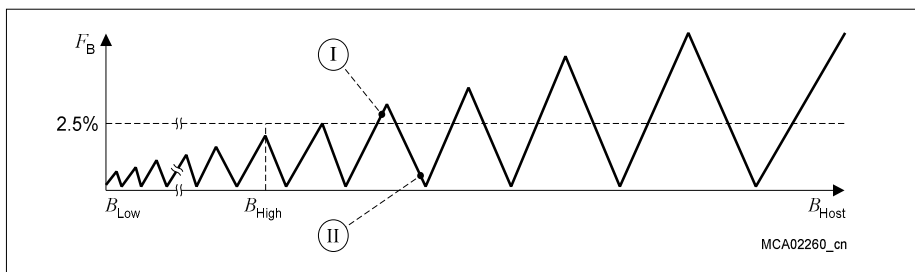
XE16xyM 使用位域 PDIV 测量第一个零字节的长度。量化误差使得测量值可能与实际波特率有一定偏差。

为了正确从主机向 XE16xyM 传送数据，U0C0 内部初始化波特率和主机的实际波特率之间的最大偏差应该低于 2.5%。可根据公式 (12.1) 计算 XE16xyM 的波特率与主机波特率之间的偏差 ( $F_B$ ，百分数)

$$F_B = \left| \frac{B_{\text{Contr}} - B_{\text{Host}}}{B_{\text{Contr}}} \right| \times 100\% \quad F_B \leq 2.5\% \quad (12.1)$$

*注：函数 ( $F_B$ ) 不考虑振荡器和其它支持串口通信的器件的容差。*

该波特率偏差是一个非线性函数，和系统时钟以及主机波特率有关。由于较小的波特率预分频因子容易导致较大的量化误差，因此函数 ( $F_B$ ) 的最大值随主机波特率增加而增大 (见图 12-3)。



**图 12-3 主机和 XE16xyM 之间的波特率偏差**

测量零字节时，最小波特率 (图 12-3 中的  $B_{\text{Low}}$ ) 由位域 PDIV 最大计数能力决定，即取决于系统时钟。在波特率公式中使用 PDIV 的最大计数值  $2^{10}$  可得到最小的波

特率值。低于  $B_{Low}$  的值会导致 PDIV 溢出。在此情况下，U0C0 不能被正确初始化，因此很可能无法与外部主机进行通信。

**最大波特率**（图 12-3 中的  $B_{High}$ ）为未超过偏差极限的最高波特率，即  $B_{Low}$  和  $B_{High}$  之间的所有波特率都低于偏差极限。 $B_{High}$  表示无需额外测试或调查、能与外部主机正常通讯的最大波特率。

只要实际偏差超过偏差极限，也可采用**更高的波特率**。某个波特率（图 12-3 中标记 I）可能超过偏差极限，而比它更高的波特率（图 12-3 中标记 II）可能低于偏差极限。满足以下三个条件的任何波特率都可被使用：

- 波特率在 U0C0 指定的工作范围内
- 外部主机能够使用该波特率
- 计算所得偏差低于极限值

*注：电源复位之后，当进入引导程序加载模式时，引导程序加载器的工作频率将为  $f_{sys} = f_{osc} \times 2$ （大约为 10 MHz），此值限制了 U0C0 的最大波特率。*

*为了达到更高的下载波特率，更高级的引导程序加载序列可切换时钟产生模式。*

### 12.6.3 同步串行通道引导程序加载

同步串行通道（SSC）引导程序加载器通过 USIC0 通道 0（U0C0）将程序代码/数据从外部串行 EEPROM 传送至 PSRAM。这里 XE16xyM 为主机，因此无需额外的（EEPROM 除外）的元件。

SSC 引导程序加载是一种在软件开发期间进行最初和基本（通过/失败）测试非常方便的方法 – 通过重新编程串行 EEPROM，使得各版本的代码在目标系统上的启动变得非常容易。

SSC 引导程序加载期间，使用同步 8 位数据帧（MSB 在先）将数据从外部 EEPROM 传送至 XE16xyM。用户可选择 SSC 启动模式中接收的数据字节个数。串行时钟速率设置为  $f_{sys}/10$ ，电源复位之后大约为 1 MHz。

一旦进入到 SSC BSL 模式并完成相应的初始化，XE16xyM 首先读取来自目标 EEPROM 首地址（00...0）的报文头。

报文头由两部分组成：

- 存储器 ID 字节：D5<sub>H</sub>
- 数据大小位域：1、2 或 3 字节，取决于 EEPROM 的寻址模式（8 位、16 位或 24 位，见章节 12.6.3.1）

如果这两部分都有效，BSL 进入一个循环，通过 U0C0 读取由数据大小域定义的多字节。

这些数据字节顺序保存在从地址 E0' 0000<sub>H</sub> 开始的 PSRAM 中，之后执行这些代码。因此，相应的衍生器件中 PSRAM 的容量决定实际的最大下载数据大小。

**注意：**用户必须注意，数据大小一定不能大于器件 **PSRAM** 的可用容量减去 **256** 字节所得的值、不能超过 **32512**。

通过  $\overline{\text{CS}}$  信号三次翻转到低电平的方式来指示报文头无效（ID 字节  $\neq \text{D5}_\text{H}$ ，数据大小位域 = 0 或大于允许值）。此方式有助于系统建立期间进行调试。

### 12.6.3.1 支持的 EEPROM 类型

XE16xyM 的 SSC 引导加载器假定与 SPI 兼容 EEPROM（25xxx 系列）相连，它支持 8 位、16 位和 24 位寻址的器件。通过检查接收到的报文头字节，确定与 XE16xyM 相连的 EEPROM 类型，见表 12-12。

注：数据大小  $n$  以字节为单位。

**表 12-12 确定 EEPROM 类型**

SSC 帧		8 位寻址 EEPROM		16 位寻址 EEPROM		24 位寻址 EEPROM	
N	数据	P11 发送	P11 接收	P11 发送	P11 接收	P11 发送	P11 接收
1	03 <sub>H</sub>	读命令	XX <sub>H</sub> 缺省电平	读命令	XX <sub>H</sub> 缺省电平	读命令	XX <sub>H</sub> 缺省电平
2	00 <sub>H</sub>	地址	XX <sub>H</sub>	地址高位		地址高位	XX <sub>H</sub>
3	00 <sub>H</sub>	空	D5 <sub>H</sub> : ID 字节	地址低位	XX <sub>H</sub>	地址中位	XX <sub>H</sub>
4	00 <sub>H</sub>	空	大小 $n$	空	D5 <sub>H</sub> : ID 字节	地址低位	XX <sub>H</sub>
5	00 <sub>H</sub>	空	数据字节 1	空	大小 $n$ , 高字节	空	D5 <sub>H</sub> : ID 字节
6	00 <sub>H</sub>	空	数据字节 2	空	大小 $n$ , 低字节	空	大小 $n$ , 高字节
7	00 <sub>H</sub>	空	数据字节 3	空	数据字节 1	空	大小 $n$ , 中字节
8	00 <sub>H</sub>	空	数据字节 4		数据字节 2	空	大小 $n$ , 低字节
9 ...	...	空	数据字节 5 ... $n$	空	数据字节 3 ... $n$	空	数据字节 1 ... $n$

注：返回的缺省字节的值（由 XX<sub>H</sub> 表示）取决于所使用的 EEPROM 类型。

### 12.6.3.2 特定设置

当 XE16xyM 进入 SSC BSL 模式时，自动进行以下配置。

**表 12-13 SSC BSL 的特定状态**

配置项	值	注释
U0C0_CCR	0001 <sub>H</sub>	USIC0 通道 0 选择 SSC 模式
U0C0_PCRL	0011 <sub>H</sub>	SSC 主模式，频率来自 f <sub>PP</sub>
U0C0_PCRH	8000 <sub>H</sub>	使能产生 MCLK
U0C0_SCTRL	0103 <sub>H</sub>	MSB 在先，被动数据电平 = 1
U0C0_SCTRH	073F <sub>H</sub>	8 个数据位，起始帧
U0C0_DX0CR	0015 <sub>H</sub>	数据输入选择
U0C0_FDRL	43FF <sub>H</sub>	选择正常分频模式 1: 1
U0C0_BRGL	0000 <sub>H</sub>	正常模式，FDIV – 复位之后的缺省值
U0C0_BRGH	8004 <sub>H</sub>	被动电平 MCLK/SCLK = 0, PDIV = 4
P2_IOCRO3	00D0 <sub>H</sub>	P2.3 为漏极开路输出 (MTSR)
P2_IOCRO4	0020 <sub>H</sub>	P2.4 为带上拉的输入 (MRST)
P2_IOCRO5	00D0 <sub>H</sub>	P2.5 为漏极开路输出 (SCLK)
P2_IOCRO6	00C0 <sub>H</sub>	P2.6 为漏极开路输出 (SLS)

### 12.6.4 CAN 引导程序加载

CAN 引导程序加载器通过 MultiCAN 模块节点 0 将程序代码/数据传送至 PSRAM 中。使用 8 字节数据帧将数据从外部主机传送至 XE16xyM。要接收的数据帧个数可编程设定，取决于 16 位数据报文计数值 DMSGC。

XE16xyM 和外部主机之间的通信基于下面的 CAN 标准帧：

- 起始帧 – 外部主机发送给 XE16xyM
- 应答帧 – XE16xyM 发送给外部主机
- 数据帧 – 外部主机发送给 XE16xyM

XE16xyM 使用起始帧进行波特率检测。完成波特率检测之后，通过发送应答帧向外部主机报告该操作的成功进行，使用数据帧发送数据。表 12-14 给出使用的三种 CAN 标准帧所需的参数和设置。

*注：CAN 引导程序加载需要与主机进行点对点连接，即 XE16xyM 必须是唯一连接到网络上的 CAN 节点。CAN 引导程序加载操作需要频率至少为 4 MHz 的晶振。*

## 起始阶段

第一个 BSL 任务是确定与外部主机通信的 CAN 波特率。因此，外部主机必须连续向 XE16xyM 发送起始帧。起始帧的头两个数据字节必须包括 2 字节波特率检测序列（5555<sub>H</sub>），应答帧的 11 位（在 2 个字节中传送）标识符 ACKID<sup>1)</sup>，16 位数据报文计数值 DMSGC 以及数据帧使用的 11 位（2 字节）标识符 DMSGID<sup>1)</sup>。

通过分析接收到的波特率检测序列（5555<sub>H</sub>）确定 CAN 波特率，并相应的设置 MultiCAN 模块的波特率寄存器。此时 XE16xyM 已经作好准备以外部主机的波特率接收 CAN 帧。

## 应答阶段

在应答阶段，引导程序加载器等待，直到接收到来自外部主机的下一个被正确识别的起始帧，引导程序加载器通过在其 ACK 时隙产生一个显性位应答该起始帧。随后，引导程序加载器将应答帧发回给外部主机，指示其已作好接收数据帧的准备。应答帧使用接收到的起始帧的报文标识符 ACKID。

总结如下：外部主机必须连续发送起始帧（内容在上文中定义），直到接收到 XE16xyM 发回的应答帧，应答帧的标识符为主机发送的起始帧的数据字节 2/3，之后进入数据发送阶段。

## 数据发送阶段

在数据发送阶段，外部主机发送，XE16xyM 接收数据帧。数据帧使用起始帧中发送的 11 位数据报文标识符 DMSGID。每个数据帧发送 8 个数据字节。第一个数据字节保存在 PSRAM 中，起始地址为 E0' 0000H，按照地址升序的方式保存随后的数据。

通信双方评估数据报文计数值 DMSGC 直到已经接收完所请求个数的 CAN 数据帧。接收最后一个 CAN 数据帧之后，引导程序加载器完成操作并开始执行加载的代码。

---

1) CAN 引导程序加载器将接收到的起始帧中的两个标识符字节直接复制到 MOAR 中。因此，起始帧中的相应位域必须按如下方式扩展：在低端为需要的标识符加上两个空位，在高端添加扩展位域 IDE (= 0<sub>B</sub>) 和 PRI (= 01<sub>B</sub>)

### 时序参数

对外部主机的 CAN 时序并没有通用的限制。起始阶段，外部主机发送起始帧。如果在外部主机定义的某时间段内 XE16xyM 没有发回应答帧（即发送特定个数的起始帧之后），外部主机能够确定 XE16xyM 不能建立 CAN 启动通信连接。

**表 12-14 CAN 引导加载器帧**

帧类型	参数	描述
起始帧	标识符	11 位，无关
	DLC = 8	数据长度代码，CAN 帧内有 8 个字节
	数据字节 0/1	波特率检测序列（5555 <sub>H</sub> ）
	数据字节 2/3	应答报文的标识符 ACKID （完整的寄存器内容）
	数据字节 4/5	数据报文计数 DMSGC，16 位
	数据字节 6/7	数据报文标识符 DMSGID （完整的寄存器内容）
应答帧	标识符	应答帧报文标识符 ACKID，与起始帧发送的数据字节[3:2]相同
	DLC = 4	数据长度代码，CAN 帧内有 4 个字节
	数据字节 0/1	位时序寄存器的内容
	数据字节 2/3	从起始帧中复制的应答标识符
数据帧	标识符	数据帧标识符 DMSGID，与起始帧发送的数据字节[7:6]相同
	DLC = 8	数据长度代码，CAN 帧内有 8 个字节
	数据字节 0 到 7	数据字节，按照目的地址升序的方式保存在 PSRAM 中

#### 12.6.4.1 特定设置

当 XE16xyM 进入 CAN BSL 模式时，自动进行以下配置。

**表 12-15 CAN BSL 的特定状态**

配置项	值	注释
P2_IOCR5	00A0 <sub>H</sub>	P2.5 为推挽输出 (TxD)
P2_IOCR6	0020 <sub>H</sub>	P2.6 为带上拉的输入 (RxD)
SCU_HPOSCCON	0030 <sub>H</sub>	OSC_HP 使能, 外部晶振/时钟模式
SCU_SYSCON0	0001 <sub>H</sub>	选择 OSC_HP 作为系统时钟
CAN_MOCTR0L	0008 <sub>H</sub>	报文对象 0 控制寄存器, 低位
CAN_MOCTR0H	00A0 <sub>H</sub>	报文对象 0 控制寄存器, 高位
CAN_MOCTR1L	0000 <sub>H</sub>	报文对象 1 控制寄存器, 低位
CAN_MOCTR1H	0F28 <sub>H</sub>	报文对象 1 控制寄存器, 高位
CAN_MOFCTR1H	0400 <sub>H</sub>	报文对象功能控制寄存器, 高位
CAN_MOAMR0H	1FFF <sub>H</sub>	报文对象 0 – 验收屏蔽位设置
CAN_NPCR0	0003 <sub>H</sub>	数据输入选择

### 12.6.5 引导程序加载模式归纳

激活引导程序加载所需的外部硬件配置归纳见下表。

**表 12-16 引导程序加载模式的配置数据**

引导程序加载模式	P10.[7:0] <sup>1)</sup> 上的配置	来自主机的接收线	送至主机的发送线	传送的数据	支持的主机速度
标准 UART	xxxx x110 <sub>B</sub> <sup>2)</sup>	RxD = P7.4	TxD = P7.3	32 字节	2.4 - 19.2 kbit/s
	xxxx x010 <sub>B</sub>	RxD = P2.4	TxD = P2.3		
增强 UART	x001 0000 <sub>B</sub>	RxD = P2.4	TxD = P2.3	1 字节 <sup>2)</sup>	开始为 2.4 - 19.2 kbit/s, 随后可由报文头修改
同步串行	xxxx 1001 <sub>B</sub>	MRST = P2.4	MTSR = P2.3	m 字节 <sup>4)</sup>	--- (由 XE16xyM 控制)

**启动配置和引导程序加载**

引导程序 加载模式	P10.[7:0] <sup>1)</sup> 上的配置	来自主机的 接收线	送至主机的 发送线	传送的 数据	支持的 主机速度
			SCLK = P2.5 SLS = P2.6		
MultiCAN	xxxx 1101 <sub>B</sub>	RxDC0 = P2.6	TxDC0 = P2.5	8 × n 字节 <sup>5)</sup>	125 - 500 kbit/s

1) x 代表与相应引脚上的电平无关；

2) 64 引脚封装中不可用。

3) l = 主机发送的代码\_长度，取值范围从 1... (PSRAM 容量-256)。

4) m = 读取自 EEPROM 的数据大小，取值范围从 1... (PSRAM 容量-256) 但不超过 32512。

5) n = DMSGC，主机在起始帧中发送的数据报文计数值，取值范围从 1...(PSRAM 容量-256)/8。



## 13 调试系统

XE16xyM 包含一个片上调试支持（OCDS）系统，通过调试接口引脚由外部设备直接控制，为用户提供了方便的调试功能。

OCDS 组成：

- 调试接口
- Cerberus
- OCDS 模块

### 片上调试支持（OCDS）

OCDS 系统（图 13-1）支持断点和跟踪存储单元等多种调试特性。OCDS 的典型应用是在实时系统环境下调试运行在 XE16xyM 上的用户软件。

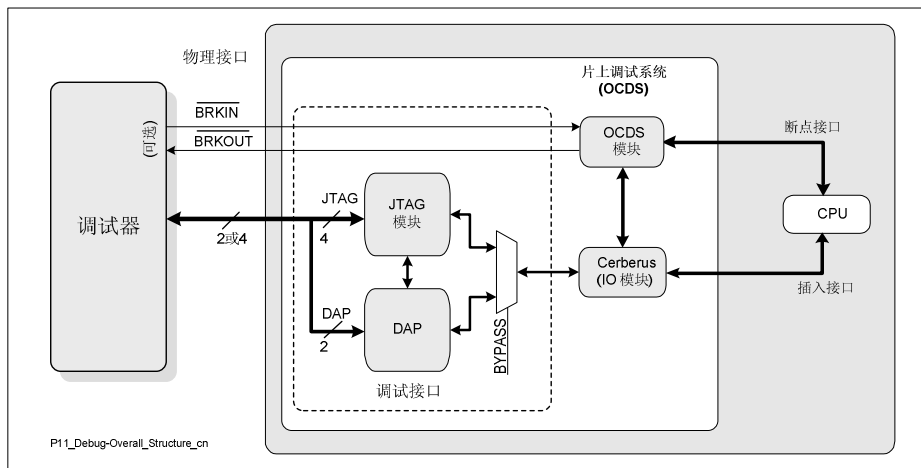


图 13-1 OCDS 框图

OCDS 通过调试接口由外部调试设备控制。其物理接口为 DAP 或 JTAG 外加一个可选的、带有一个或两个引脚的断点接口。断点接口支持 XE16xyM 和工具（或组成的系统环境）之间很慢的延迟触发机制（如有需要）。可使用 Cerberus 通过 DAP/JTAG 接口访问 OCDS 寄存器。OCDS 中还有一组专用的 Cerberus 调试 IO 指令。此外，OCDS 还可由调试监控程序控制（该程序通过用户接口（如 CAN）和调试工具通信）。OCDS 系统与 CPU 内核可以通过插入接口进行交互从而能执行 Cerberus 生成的指令，还可通过断点接口进行交互。

## OCDS 系统特性

- 硬件、软件和外部引脚断点
- 可通过 CPU 暂停、监控程序调用、数据传送和/或  $\overline{\text{BRKOUT}}$  信号触发操作
- 对整个地址空间的读/写访问
- 单步调试
- 非介入式调试（不需调试监控程序）
- 可通过用户接口（如 CAN）进行调试（利用调试监控程序）
- DAP 或 JTAG 接口以及可选的断点接口
- 插入任意 CPU 指令
- 通过外部总线传送进行快速存储器跟踪（若可用）

## 13.1 调试接口

可通过调试接口访问 OCDS 资源。通过该调试接口，可以与所有片上、片外存储器和存储器映射控制寄存器交换数据。

### 功能和特性

- 用于 OCDS 的独立接口
- DAP（器件访问端口）或 JTAG
- 用作外部触发输入和内部触发指示的断点接口
- 一般存储器访问功能
- 独立的数据传送通道，比如用来对 Flash 存储器编程

调试接口包括：

- **DAP 接口**
- 基于 IEEE 1149.1 JTAG 标准的可选 **JTAG 接口**
- 两个附加的 XE16xyM 特定信号 – **OCDS 断点接口**

**注：DAP/JTAG 时钟频率必须低于当前的 CPU 频率。**

### DAP 接口

DAP 接口为标准化的器件访问端口，最新的英飞凌微控制器都会配备该端口。DAP 接口的引脚数减至 2 个，具有良好的抗噪声能力和高度的稳健性。

该接口由以下信号组成：

- **DAP0** – 时钟
- **DAP1** – 串行数据输入/输出

## JTAG 接口

JTAG 接口是一个标准化的专用端口，主要用于边界扫描板测试。

该接口由 JTAG IEEE 1149.1-2001 标准信号组成：

- **TDI** - 串行数据输入
- **TDO** - 串行数据输出
- **TCK** - JTAG 时钟
- **TMS** - 状态机控制信号

## OCDS 断点 - 接口

在调试器和 XE16xyM **OCDS 模块**之间，由两个（可选的）附加信号提供一个直接的触发接口：

- **BRKIN**（断点输入请求）可直接触发一个**调试动作**。
- **BRKOUT**（断点输出信号）可以由 OCDS 激活，用于通知外部环境某些预设的调试事件已经发生

### 13.1.1 调试信号连接

实际应用需要尽可能多的 IO 引脚，通过 JTAG 接口和断点接口与外部调试器连接的信号通常和该应用需求发生冲突。在 XE16xyM 中，这些信号仅作为引脚的复用功能（非专用引脚）。为了尽可能减少调试接口引脚带来的影响，这些信号可被映射到不同的位置上。因此，每种应用可选择受影响最小的位置。这由调试引脚连线寄存器

**DBGPRR** 进行选择。引脚 BRKOUT 可被分配给 P6.0、P10.11、P1.5 或 P9.3，通过相应的 IOC 寄存器设置为标准的复用输出信号。

#### 13.1.1.1 寄存器 DBGPRR

该寄存器控制 JTAG/DAP 的引脚连接。由寄存器 **DBGPRR** 控制连接方式。

**DBGPRR** 中的位域描述适用于所有带有 DAP/JTAG 接口的该系列衍生产品，涵盖了所有可能的连接方式。对于低引脚数封装的产品，不可用的引脚位置应看作保留。

在启动过程中设置 **DBGPRR**，具体描述见**章节 12.3.2**。

## DBGPRR

调试引脚连线寄存器

ESFR (F06E<sub>H</sub>/37<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRS TL	TRS TS	TRS TGT	DBG EN	JTAG DAP	DPR E	DPR BRKIN		DPR TCK		DPR TMS		DPR TDI		DPR TDO	
rh	rh	rw	rw	rw	r	rw		rw		rw		rw		rw	

符号	位序号	类型	功能描述
<b>DPRTDO</b>	[1:0]	rw	<b>DAP1/TDO 的引脚连线</b> 00 <sub>B</sub> P7.0 01 <sub>B</sub> P10.12 10 <sub>B</sub> 保留 11 <sub>B</sub> 保留
<b>DPRTDI</b>	[3:2]	rw	<b>TDI 的引脚连线</b> 00 <sub>B</sub> P5.2 01 <sub>B</sub> P10.10 10 <sub>B</sub> P7.2 11 <sub>B</sub> P8.3
<b>DPRTMS</b>	[5:4]	rw	<b>TMS 的引脚连线</b> 00 <sub>B</sub> P5.4 01 <sub>B</sub> P10.11 10 <sub>B</sub> P7.3 11 <sub>B</sub> P8.4
<b>DPRTCK</b>	[7:6]	rw	<b>DAP0/TCK 的引脚连线</b> 00 <sub>B</sub> P2.9 01 <sub>B</sub> P10.9 10 <sub>B</sub> P7.4 11 <sub>B</sub> P8.5
<b>DPRBRKIN</b>	[9:8]	rw	<b>BRKIN 的引脚连线</b> 00 <sub>B</sub> P5.10 01 <sub>B</sub> P10.8 10 <sub>B</sub> P7.1 11 <sub>B</sub> P8.6

符号	位序号	类型	功能描述
DPRE	10	rw	<b>DAP/JTAG P13 口的连接</b> DAP0/TCK P13.6、DAP1/TDO P13.8、TMS P13.3、TDI P13.5。该连接设置的优先级高于这些引脚的其它连接设置。
JTAG_DAP	11	rw	<b>选择调试接口</b> 0 <sub>B</sub> 使用 DAP 1 <sub>B</sub> 使用 JTAG
DBGEN	12	rw	<b>使能所选择的调试接口</b> 0 <sub>B</sub> 接口被禁止 1 <sub>B</sub> 接口被使能
TRSTGT	13	rw	<b>TRST 引脚的门控</b> 0 <sub>B</sub> DAP/JTAG 复位内部保持有效 1 <sub>B</sub> TRST 引脚被连接至 DAP/JTAG 复位
TRSTS	14	rh	<b>TRST 引脚值</b> TRST 引脚的当前值
TRSTL	15	rh	<b>锁存 TRST 引脚的启动值</b> 释放 PORST 时锁存的 TRST 引脚值

## 13.2 OCDS 模块

OCDS模块的应用是在用户系统环境下调试运行在CPU上的用户软件。该应用由外部调试器实现，该调试器通过独立的[调试接口](#)来控制[OCDS模块](#)。

### 特性

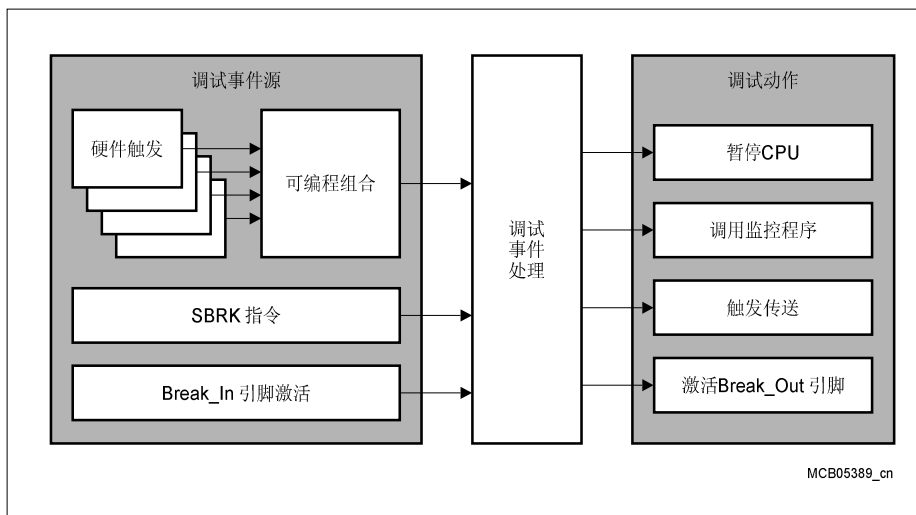
- 硬件、软件和外部引脚断点
- 硬件触发产生断点和外部引脚输出
  - 指令或数据的四个地址或两个地址范围
  - 指令（范围）和数据地址（范围）组合
  - 数据地址（范围）和数据值（范围）组合
  - 任务 ID，可选择和指令或数据的地址（范围）组合

- 地址和数据的屏蔽比较
- 还可通过调试监控程序来配置 OCDS
- 通过监控程序或 CPU 暂停进行单步调试
- 若 CPU 暂停，仍可响应高优先级中断
- 暂停模式下中断指针可见

### 基本概念

片上调试的概念分为两个部分。第一部分包含调试事件的产生，第二部分定义产生调试事件时采取何种动作。

- 调试事件：
  - **硬件断点**
  - **软件断点**
  - 激活**断点输入引脚**
- 调试事件动作：
  - **CPU 暂停模式**
  - **调用监控程序**
  - **触发传送**
  - **激活外部引脚**



**图 13-2 OCDS 概念：框图**

### 13.2.1 调试事件

调试事件可以有几种不同的来源。

#### 硬件断点

**硬件断点**是一种调试事件。当单个触发信号或多个触发信号的组合与设定的条件相匹配时，产生调试事件。

可以使用如下的硬件触发源：

**表 13-1 硬件触发**

触发源	大小
任务标识符	16 位
指令指针	24 位
读数据地址（两个总线被监控）	$2 \times 24$ 位
写数据地址	24 位
数据值（读或写）	16 位

## 软件断点

特殊的 SBRK（软件断点）指令的操作码定义为 0x8C00。调试器可使用该指令临时替代 RAM 中的代码，以此实现**软件断点**。当 SBRK 指令被解码并进入执行阶段时，整个流水线被取消，包括 SBRK 指令本身。这意味着将从 SBRK 被发现的地址处获取下一条指令。

下一步的操作取决于 OCDS 的设置：

- 如果 OCDS 被使能、软件断点也被使能，则 CPU 进入**暂停模式**
- 如果 OCDS 被禁止，或者软件断点被禁止，则执行**软件断点强制中断**（SBRKTRAP），该强制中断是 A 类强制中断，中断编号 08<sub>H</sub>

## 断点引脚输入

外部的调试断点引脚（**BRKIN**）可使调试器异步中断处理器。

### 13.2.2 调试动作

当 OCDS 使能、并产生调试事件时，会发生下列动作：

#### 触发传送

调试事件产生时可能发生的一个动作是触发**Cerberus**：

- 执行数据传送。这可用于在不能被中断的重要子程序中进行存储器数据传送
- 向内核插入一条指令，利用该机制，可将任意一条指令插入 XE16xyM 流水线

#### 暂停模式

该动作发生后，OCDS 模块向 CPU 发送一个断点请求。

如果 OCDS 断点优先级高于当前 CPU 优先级，则 CPU 接受该断点请求；如果断点请求被接收，系统进入挂起状态，指令流暂停。

更高优先级的用户中断仍然可以中断暂停模式。如果发生这种情况，外部调试系统可以通过调试接口进行读取和更新操作来监控目标系统。

## 调用监控程序

调试事件发生时可能采取的一个动作是调用监控程序。快速转入监控程序使得能够定义灵活的调试环境，可以满足调试实时系统的很多需求。通常情况下监控程序具有最高优先级，不能被任何请求源中断。

还有一种可被中断的监控程序。这种情况下，当监控程序（调试器）工作时，仍然可以执行保障安全的重要代码，这给用户带来了极大的灵活性。



## 激活外部引脚

该动作激活 **OCDS断点接口** 的外部引脚 **BRKOUT**。在不能被中断的重要子程序中，通过外部引脚 **BRKOUT** 向外部环境发出一个信号，指示已经发生了一个特殊的事件。需要注意，代码执行时序不受影响。

## 13.3 Cerberus

Cerberus 模块提供和控制外部调试器（通过 **调试接口**）、**OCDS 模块**、XE16xyM 内部系统之间进行交互所必需的所有操作。

### 特性

- DAP/JTAG 接口用作控制 and 数据通道
- 一般读/写功能（RW 模式），可以访问整个地址空间
- 读/写通用寄存器（GPR）
- 插入任意指令
- 外部主机控制所有任务
- 在正常运行和暂停模式下，可以获得所有任务的信息
- 任务的优先级可配置
- 完全支持监控程序和调试器之间的通信
- 可选择错误保护
- 通过将数据值传送到外部总线实现存储器跟踪功能
- 内部总线锁定状态的分析寄存器

Cerberus 模块的目标应用是将 DAP/JTAG 接口作为独立的端口进行片上调试。外部调试器通过插入机制能够访问 OCDS 寄存器和任意存储器地址。

### 13.3.1 功能概述

Cerberus 模块通过 DAP/JTAG 接口由外部调试器控制。调试器使用 Cerberus IO 指令执行双向数据传送。Cerberus 有两种主要的工作模式：

#### 读/写（RW）模式

RW 模式是操作 Cerberus 最常见的模式。该模式用于读写存储器地址或向 CPU 流水线中插入指令。该模式下，CPU 的插入接口被激活。

#### 通信（COM）模式

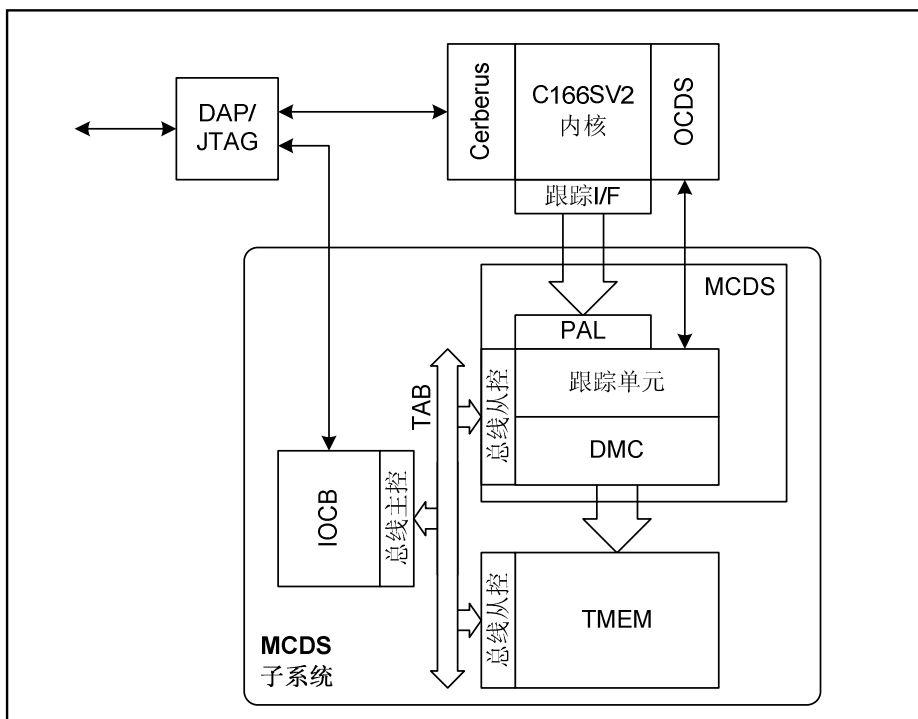
COM 模式下，调试器与运行在 CPU 上的监控程序进行通信。

和**读/写（RW）模式**不同，该模式下 Cerberus 不执行读写请求，它只是置位 CPU 可访问寄存器中的请求位，通知监控程序调试器想要发送或接收数据。监控程序必须查询该状态寄存器（比如由定时器中断触发该操作），相应执行正确的动作。

COM 模式是复位后的缺省模式。可通过该模式和锁定（RW 模式禁用）器件的应用软件交换密钥，一旦密钥匹配，解锁 RW 模式。

### 13.4 仿真器件

可通过基于 MCDS（多核调试方案）、带有片上跟踪缓存的仿真器件实现 XE16xyM 仿真（**图 13-3**）。英飞凌工具供应商可为用户提供这样的仿真器。



**图 13-3 仿真器件框图**

### **13.4.1 MCDS 使用**

MCDS 具备非介入式跟踪能力，可用于触发调试、性能分析和数据测量。

#### **调试**

- 支持灵活的组合触发条件
- 记录 bug 的热点区域
- 跟踪缓存已满时系统暂停。读出数据后继续工作
- 高压缩比、高精度（周期）跟踪数据

#### **性能分析**

- 性能指标的连续测量
- 性能指标可用作触发源

#### **数据测量**

- 数据读写的连续跟踪
- 由地址范围和软件任务限定

### **13.4.2 MCDS 特性**

MCDS 提供了一组丰富的特性，可对软件和系统行为进行所有层次上的详尽分析。

#### **CPU 程序跟踪**

- 完整的程序跟踪（24 位指令指针）
- 四个独立的范围比较器

#### **CPU 所有权跟踪**

- 对流水线“所有者”（如 PEC 通道）的完整跟踪
- 两个独立的用于数据传送的可屏蔽比较器

#### **CPU 状态跟踪**

- CPU 工作模式的完整跟踪
- 非介入式访问当前状态

### **写数据跟踪**

- 写回操作的完整跟踪（24 位地址，8 或 16 位数据）
- 四个独立的范围比较器（用于绝对写地址的比较）
- 四个独立的有符号数据比较器（用于数据比较）

### **读数据跟踪**

- 非 CPU 存储器读取操作的完整跟踪（24 位地址，8 或 16 位数据）
- 四个独立的范围比较器（用于绝对地址的比较）
- 四个独立的有符号数据比较器（用于数据比较）

### **跟踪和触发控制**

- 每个跟踪单元的跟踪使能均可进行编程
- 触发输出至 OCDS
- 8 个通用 16 位计数器
- 触发源可编程组合用作计数和清零信号
- 每个计数器中包含一个可编程的限定比较器
- 每一个计数器均可输出计数值越界的触发信号
- 计数值可被跟踪
- 计数器可级联以实现状态机
- 预分频参考时钟可用作触发源
- 四个直接来自 CPU 的性能指示信号

### **观察点跟踪**

- 8 个不同观察点的信息
- 事件/性能计数器的当前计数值的信息

### **时间戳**

- 基于仿真时钟的精确时间戳（32 位）
- 基于参考时钟的精确时间戳（32 位）
- 基于参考时钟的可编程周期触发源

### 13.5 边界扫描

XE16xyM 提供符合 IEEE 标准 1149.1 的边界扫描功能，易于对应用系统进行板级分析。XE16xyM 还支持焊装在 PCB 板上的多个器件之间的交互测试。

通过 JTAG 模块实现边界扫描，符合标准 JTAG 指令（IEEE 1149.1）。

*注：为了使边界扫描正确工作，JTAG 接口必须使用缺省的引脚。寄存器 DBGPRR 的复位值可确保该设置。*

#### 边界扫描的初始化

为了激活边界扫描模式，需要进行下述操作：

- 设置  $\overline{\text{PORST}} = 1$ ； $\overline{\text{TRST}} = 1$ ； $\overline{\text{TESTM}} = 1$
- $\overline{\text{PORST}}$  上的负脉冲
- 等待电源域启动
- $\overline{\text{TRST}}$  上的负脉冲，用来复位 JTAG 控制器

至此边界扫描的测试访问接口被使能。边界扫描测试能够使用如 PRELOAD 和 EXTEST 等指令进行板级测试。

## 14 指令集概述

本章简要总结 XE16xyM 的指令集，按照指令的类型进行分类。本章可使读者基本上了解功能强大、种类繁多的 XE16xyM 指令集及其一般使用。

XE166 系列的“**指令集手册**”给出每条指令的**详细描述**，包括操作数类型、状态标志设置、寻址模式、指令长度（字节数）和目标代码格式等；本手册仅提供了按照不同准则归类的指令表，便于用户快速查询。

### 指令类型总结

将指令按照类型分组可以帮助用户区分相似的指令（例如 SHR，ROR）以及特定指令的变体（例如 ADD，ADDB）；该方法为用户提供了 XE16xyM 指令功能的便捷访问。

*注：所使用的助记符参见指令详细描述。*

**表 14-1 算术运算指令**

两个字或字节加法：	ADD	ADDB
带进位的两个字或字节加法：	ADDC	ADDCB
两个字或字节减法：	SUB	SUBB
带借位的两个字或字节减法：	SUBC	SUBCB
16×16 位有符号或无符号数乘法：	MUL	MULU
16/16 位有符号或无符号数除法：	DIV	DIVU
32/16 位有符号或无符号数除法：	DIVL	DIVLU
字或字节的 1 补码：	CPL	CPLB
字或字节的 2 补码（负）：	NEG	NEGB

**表 14-2 逻辑运算指令**

两个字或字节按位与：	AND	ANDB
两个字或字节按位或：	OR	ORB
两个字或字节按位异或：	XOR	XORB

**表 14-3 比较和循环控制指令**

比较两个字或字节：	CMP	CMPB
两个字比较后，自行增加 1 或 2：	CMPI1	CMPI2
两个字比较后，自行减少 1 或 2：	CMPD1	CMPD2

**表 14-4 布尔位操作指令**

对字的高字节或低字节中的可屏蔽位域操作：	BFLDH	BFLDL
对某位置 ‘1’：	BSET	-
对某位清 ‘0’：	BCLR	-
转移某一位：	BMOV	-
对某位取反后转移：	BMOVN	-
两位与操作：	BAND	-
两位或操作：	BOR	-
两位异或操作：	BXOR	-
两位比较：	BCMP	-

**表 14-5 移位和循环移位指令**

字右移：	SHR	-
字左移：	SHL	-
字循环右移：	ROR	-
字循环左移：	ROL	-
字算术右移（符号位移位）：	ASHR	-

**表14-6 优先化指令**

确定归一化字操作数所需的移位次数（支持浮点数）：	PRIOR	-
--------------------------	-------	---

**表 14-7 数据转移指令**

字或字节的标准数据转移:	MOV	MOVB
将字节数据转移到字地址, 进行符号或零字节扩展:	MOVBS	MOVVBZ

*注: 数据转移指令可以和很多不同的寻址方式一起使用, 包括间接寻址和自动指针增加/减少。*

**表14-8 系统堆栈指令**

将一个字压入系统堆栈:	PUSH	-
将一个字从系统堆栈中弹出:	POP	-
将一个字存入系统堆栈, 然后用新值更新旧值 (用于寄存器组切换):	SCXT	-

**表 14-9 跳转指令**

在当前程序段内, 有条件跳转至绝对、间接或相对寻址的目标指令:	JMPA	JMPI	JMPR
在任意程序段内, 无条件跳转至绝对寻址的目标指令:	JMPS	-	-
在当前程序段内, 根据可选位的状态, 有条件跳转至相对寻址的目标指令:	JB	JNB	-
在当前程序段内, 根据可选位的状态, 有条件跳转至相对寻址的目标指令, 跳转后对测试位取反 (支持旗语):	JBC	JNBC	-

**表 14-10 调用指令**

在当前程序段内, 有条件调用绝对或间接寻址的子程序:	CALLA	CALLI
在当前程序段内, 无条件调用相对寻址的子程序:	CALLR	-
在任意程序段内, 无条件调用绝对寻址的子程序:	CALLS	-
在当前程序段内, 无条件调用绝对寻址的子程序, 并将选择的寄存器压入系统堆栈:	PCALL	-



**指令集概述**

无条件跳转至代码段<VECSEG>内的中断或强制中断向量跳转表:	TRAP	-
----------------------------------	------	---

**表14-11      返回指令**

在当前程序段内，从子程序返回:	RET	-
在任何程序段内，从子程序返回:	RETS	-
在当前程序段内，从子程序返回，并将选择的寄存器从系统堆栈弹出:	RETP	-
从中断服务程序返回:	RETI	-

**表14-12      系统控制指令**

通过软件复位 XE16xyM:	SRST	-
进入空闲模式:	IDLE	-
无功能，不要使用 <sup>1)</sup> :	PWRDN	-
服务看门狗定时器	SRVWDT	-
禁止看门狗定时器	DISWDT	-
使能看门狗定时器（只能在 WDT 增强模式下执行）:	ENWDT	-
指示初始化子程序结束（将寄存器安全机制切换至“受保护”，而且在 WDT 兼容模式下，初始化之后禁止执行 DISWDT 指令）:	EINIT	-

1) 在先前的 16 位架构中，指令 PWRDN 用于进入掉电模式。然而在 XE16xyM 器件中，PWRND 无任何意义，象 NOP 指令一样被执行。

**表 14-13      其它指令**

空操作，需要 2 个字节的存储空间和最短执行时间:	NOP	-
定义不可分的指令序列:	ATOMIC	-
将 ‘reg’，‘bitoff’，‘bitaddr’ 寻址模式切换至扩展 SFR 区:	EXTR	-

## 指令集概述

直接选择特定的数据页（代替 DPP）替代 DPP 寻址模式，可选择切换至 ESFR 区：	EXTP	EXTPR
直接选择特定的段（代替 DPP）替代 DPP 寻址模式，可选择切换至 ESFR 区：	EXTS	EXTSR

注：ATOMIC 和 EXT\* 指令支持不可中断的代码序列，例如用于信号操作。还支持超过当前 DPP 极限（ATOMIC 除外）的数据寻址，该特性对于利用高级语言对较大存储器模型的编程非常有用。

表 14-14 MAC 单元指令

乘（和累加）：	CoMUL	CoMAC
加/减：	CoADD	CoSUB
右移/左移：	CoSHR	CoSHL
算术右移：	CoASHR	-
加载累加器：	CoLOAD	-
存储 MAC 寄存器：	CoSTORE	-
比较值：	CoCMP	-
最小/最大：	CoMIN	CoMAX
绝对值：	CoABS	-
舍入：	CoRND	-
数据转移：	CoMOV	-
累加器取负：	CoNEG	-
空操作：	CoNOP	-

## 受保护指令

XE16xyM 指令集中的一些指令对于微控制器的功能十分重要，这些指令被称为受保护指令。这些指令使用最长的 32 位指令格式，而常规指令仅使用最长指令格式的一部分（比如低 8 位），其它位用来提供附加信息，如所用到的寄存器。在取指期间数据可能被破坏，通过对 32 位受保护双字指令的所有位进行解码可以增强数据安全性。重要的操作，如软件复位，只有在指令解码没有任何错误的情况下才能执行。这增强了微控制器系统的安全性和可靠性。

## 15 器件规范

器件规范描述了器件的电气参数，给出器件的直流特性（如输入、输出、电源或电流）和交流特性（如时序特性和时序要求）。

这些直流和交流特性不仅受 XE16xyM 内核架构、指令集、内核基本功能以及片上外设的影响、而且会随器件改进、或者标准器件各衍生产品的不同而改变。

本用户手册没有给出这些电气参数，如有需要，请参阅数据手册（不断被更新）。

所有的电气参数都可从最新版的相关数据手册中查阅到。

*注：在开始新的设计之前，必须先核实器件的特性，从而确保使用最新信息。*

XE16xyM 系列衍生产品提供几种封装形式。下面各图给出 XE16xyM 的基本引脚框图。它们给出不同的电源和 IO 引脚的排列。所有引脚及其可选功能的详细描述请查询相关数据手册。

*注：并非所有的衍生产品都能提供图中给出的全部封装形式。*

*具体描述请参阅相关产品的数据手册。*







## 16 通用定时器单元

通用定时器单元 GPT1 和 GPT2 模块具有非常灵活的多功能定时器结构，可用作定时、事件计数、脉宽测量、脉冲生成、倍频及其它用途。五个 16 位定时器组成两个定时器模块 GPT1 和 GPT2。每个模块中的各个定时器均可独立工作在不同的工作模式，如门控定时器模式、计数器模式、或者和同模块中的其它定时器级联工作。每个模块具有复用输入/输出功能及与其相关的专用中断。

*注：可通过寄存器 PISEL 选择来自多个源的输入信号。*

**模块 GPT1** 中包含三个定时器/计数器：核心定时器 T3 和两个辅助定时器 T2 和 T4。最大计数精度为  $f_{GPT}/4$ 。GPT1 的辅助定时器可被配置为核心定时器的重载或捕获寄存器。寄存器描述请参阅 [章节 16.1.6](#)。

- 最大计数精度  $f_{GPT}/4$
- 3 个独立的定时器/计数器
- 定时器/计数器可级联
- 4 种工作模式
  - 定时器模式
  - 门控定时器模式
  - 计数器模式
  - 增量接口模式
- 重载和捕获功能
- 独立的中断线

**模块 GPT2** 中包含两个定时器/计数器：核心定时器 T6 和辅助定时器 T5。最大计数精度为  $f_{GPT}/2$ 。附加的捕获/重载寄存器 (CAPREL) 支持扩展功能的捕获和重载操作。寄存器描述请参阅 [章节 16.2.7](#)。核心定时器 T6 可和 CAPCOM 单元中的定时器 (T7 和 T8) 级联工作。

模块 GPT2 的特性总结如下：

- 最大计数精度  $f_{GPT}/2$
- 2 个独立的定时器/计数器
- 定时器/计数器可级联
- 3 种工作模式
  - 定时器模式
  - 门控定时器模式
  - 计数器模式
- 由 16 位捕获/重载寄存器 CAPREL 实现的扩展捕获/重载功能
- 独立的中断线

## 16.1 定时器模块 GPT1

从编程人员的角度来看，GPT1 由下面列出的一组特殊功能寄存器（SFR）组成。端口和方向寄存器中用作 GPT1 功能的部分用阴影标出。

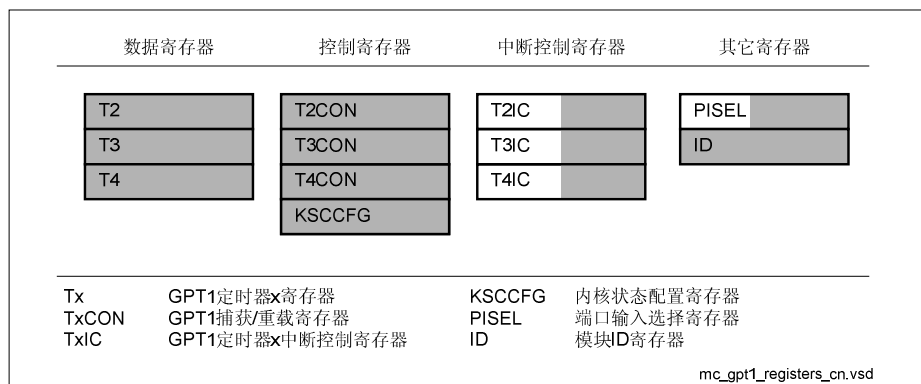


图 16-1 定时器模块 GPT1 的相关特殊功能寄存器（SFR）

GPT1 中的三个定时器（T2、T3、T4）可工作在 4 种基本模式：定时器模式、门控定时器模式、计数器模式或增量接口模式。三个定时器均可递增或递减计数。GPT1 的每个定时器由各自的控制寄存器 TxCON 控制。

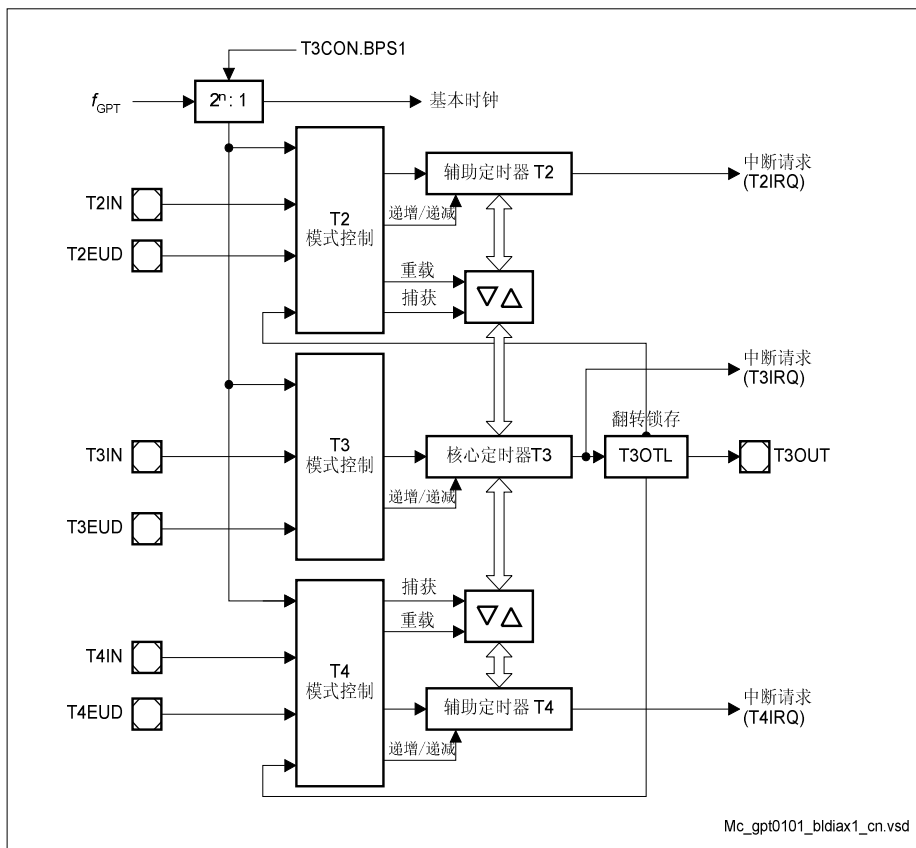
每个定时器有一个对应的输入引脚 TxIN（复用引脚功能），用作门控定时器模式中的门控信号，或计数器模式中的计数输入。计数方向（递增/递减）可由软件设定，或由外部递增/递减控制输入引脚 TxEUD（复用引脚功能）上的信号动态控制。输出翻转锁存 T3OTL 指示核心定时器 T3 发生上溢/下溢，T3OTL 的状态可从引脚 T3OUT（复用引脚功能）输出。辅助定时器 T2 和 T4 还可以（通过 T3OTL）和核心定时器 T3 级联工作，或者用作核心定时器 T3 的捕获或重载寄存器。

定时器计数寄存器 T2、T3 或 T4 位于不可位寻址的 SFR 区，CPU 访问这些寄存器，可读取或修改每个定时器的当前值（请参阅[章节 16.1.6](#)）。若定时器将执行递增、递减、重载或捕获操作时，CPU 恰好正在对（任意）某个定时器寄存器写入，CPU 的写操作则占优以确保结果正确。

由中断控制寄存器 TxIC 控制 GPT1 的中断产生，这些寄存器不在 GPT1 模块中。GPT1 的输入和输出线与 P3 和 P5 口的引脚相连。端口功能的控制寄存器位于相应的端口模块中。

注：对外部输入信号的时序要求请参阅[章节 16.1.5](#)，包括引脚在内的模块接口信号归纳见[章节 16.5](#)。





**图 16-2 GPT1 框图 (n = 2...5)**

### 16.1.1 GPT1 核心定时器 T3 的控制

核心定时器 T3 的当前值可从计数寄存器 T3 中读取，该寄存器也可由 CPU 写入，例如 CPU 设置定时器的初始值。

由可位寻址控制寄存器 T3CON 配置和控制核心定时器 T3。

#### GPT12E\_T3CON

定时器 T3 控制寄存器

SFR (FF42H/A1H)

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T3 R DIR	T3 CH DIR	T3 ED GE	BPS1		T3 OTL	T3 OE	T3 UDE	T3 UD	T3R	T3M			T3I		
rh	rwh	rwh	rw		rwh	rw	rw	rw	rw	rw			rw		

符号	位序号	类型	功能描述
T3I	[2:0]	rw	<b>定时器 T3 输入参数选择</b> 不同工作模式的输入参数选择请分别参阅相关内容： <b>表 16-7</b> 对应定时器模式和门控定时器模式 <b>表 16-2</b> 对应计数器模式 <b>表 16-3</b> 对应增量接口模式
T3M	[5:3]	rw	<b>定时器 T3 模式控制</b> （基本工作模式） 000 <sub>B</sub> 定时器模式 001 <sub>B</sub> 计数器模式 010 <sub>B</sub> 门控定时器模式，门控信号低电平有效 011 <sub>B</sub> 门控定时器模式，门控信号高电平有效 100 <sub>B</sub> 保留，不要使用该组合 101 <sub>B</sub> 保留，不要使用该组合 110 <sub>B</sub> 增量接口模式 （旋转检测模式） 111 <sub>B</sub> 增量接口模式 （边沿检测模式）

符号	位序号	类型	功能描述
<b>T3R</b>	6	rw	<b>定时器 T3 运行控制位</b> 0 <sub>B</sub> 定时器 T3 停止运行 1 <sub>B</sub> 定时器 T3 运行
<b>T3UD</b>	7	rw	<b>定时器 T3 递增/递减计数控制<sup>1)</sup></b> 0 <sub>B</sub> 定时器 T3 递增计数 1 <sub>B</sub> 定时器 T3 递减计数
<b>T3UDE</b>	8	rw	<b>定时器 T3 外部递增/递减计数使能<sup>1)</sup></b> 0 <sub>B</sub> 输入 T3EUD 被断开 1 <sub>B</sub> 计数方向受输入 T3EUD 的控制
<b>T3OE</b>	9	rw	<b>上溢/下溢输出使能</b> 0 <sub>B</sub> 复用输出功能被禁止 1 <sub>B</sub> T3OTL 的状态从引脚 T3OUT 输出
<b>T3OTL</b>	10	rwh	<b>定时器 T3 溢出翻转锁存</b> 每次 T3 上溢/下溢时翻转锁存。可由软件置位或复位。（具体描述请参阅相关章节）
<b>BPS1</b>	[12:11]	rw	<b>模块 GPT1 预分频控制</b> 选择模块 GPT1 的基本时钟 （请参阅 <a href="#">章节 16.1.5</a> ） 00 <sub>B</sub> $f_{GPT}/8$ 01 <sub>B</sub> $f_{GPT}/4$ 10 <sub>B</sub> $f_{GPT}/32$ 11 <sub>B</sub> $f_{GPT}/16$
<b>T3EDGE</b>	13	rwh	<b>定时器 T3 计数沿检测标志</b> 每次检测到计数沿该位被置位。T3EDGE 必须由软件清零。 0 <sub>B</sub> 未检测到计数沿 1 <sub>B</sub> 已检测到计数沿

符号	位序号	类型	功能描述
<b>T3CHDIR</b>	14	rwh	<b>定时器 T3 计数方向改变标志</b> 每次定时器 T3 的计数方向改变时，该位被置位。 T3CHDIR 必须由软件清零。 0 <sub>B</sub> 未检测到计数方向改变 1 <sub>B</sub> 已检测到计数方向改变
<b>T3RDIR</b>	15	rh	<b>定时器 T3 计数方向标志</b> 0 <sub>B</sub> 定时器 T3 递增计数 1 <sub>B</sub> 定时器 T3 递减计数

1) 位 T3UD 和 T3UDE 的编码见 [表 16-1](#)。

### 定时器 T3 运行控制

可通过软件设定 T3R（定时器 T3 运行控制位），启动或终止核心定时器 T3 工作。该控制位对 T3 的所有工作模式均有效。置位 T3R 将启动定时器工作；清除 T3R 将终止定时器工作。

门控定时器模式下，只有当 T3R = 1 且门控信号有效（编程设置的有效电平：高电平或低电平）时，定时器才能工作。

*注：若定时器控制寄存器 T2CON 或 T4CON 中的位 T2RC 或 T4RC 被置位，位 T3R 也将控制（启动或终止）辅助定时器 T2 和/或 T4 的运行。*

### 计数方向控制

GPT1 定时器（核心定时器和辅助定时器）的计数方向可由控制寄存器 TxCON 中的位 TxUD 和 TxUDE 控制，选择由软件、或外部输入引脚 TxEUD（定时器 Tx 外部递增/递减控制输入）控制递增/递减计数。由软件控制计数方向时（TxUDE = 0），置位或清除 TxUD 可改变计数方向；位 TxUDE = 1 时，由引脚 TxEUD 控制计数方向。但是，仍可用 TxUD 来翻转实际的计数方向，如表 16-1 所示。无论定时器是否工作，计数方向均可被改变。

*注：当引脚 TxEUD 用作外部计数方向控制输入时，必须被配置为输入引脚。*

**表 16-1 GPT1 定时器计数方向控制**

引脚 TxEUD	位 TxUDE	位 TxUD	计数方向	位 TxRDIR
X	0	0	递增计数	0
X	0	1	递减计数	1
0	1	0	递增计数	0
1	1	0	递减计数	1
0	1	1	递减计数	1
1	1	1	递增计数	0

### 定时器 T3 输出翻转锁存

定时器 T3 的上溢/下溢信号送入定时器工作模式框图中所示的“翻转锁存”模块。

**图 16-3** 示出该模块的内部逻辑。T3 的上溢或下溢信号作为两个锁存器的输入时钟：第一个锁存器代表控制寄存器 T3CON 中的位 T3OTL；第二个锁存器是由 T3OTL 的输出来控制翻转的内部锁存器。两个锁存器的输出均送入辅助定时器 T2 和 T4 的输入控制模块。映射锁存器的输出电平会与 T3OTL 的输出电平相匹配，只是延迟一个时钟周期。若 T3OTL 的值发生变化，将会导致 T3OTL 和映射锁存器的输出电平暂时不同，从而可触发定时器 T2 和/或 T4 的计数事件。

软件写入 T3OTL 时，两个锁存器被同时置位或清除。此时送入辅助定时器的两个信号的电平相同，辅助定时器不会检测到跳变沿。寄存器 T3CON 中的位 T3OE（上溢/下溢输出使能）可使能由外部引脚 T3OUT 监控 T3OTL 的状态。若将 T3OTL 连接至端口引脚上（必须将端口相应的配置为输出），可用 T3OUT 控制外部硬件。若 T3OE = 1，引脚 T3OUT 输出 T3OTL 的状态。若 T3OE = 0，引脚 T3OUT 输出高电平（只要选择此端口用作 T3OUT 复用输出功能）。

触发信号可用作辅助定时器 T2 和 T4 计数器功能的输入，或者作为 T2 和 T4 重载功能的触发源。

如图 16-3 所示，当软件修改锁存器 T3OTL 的值、以决定输出状态时，内部映射锁存器也相应被置位或清除。此时 T2/T4 不会检测到触发事件。

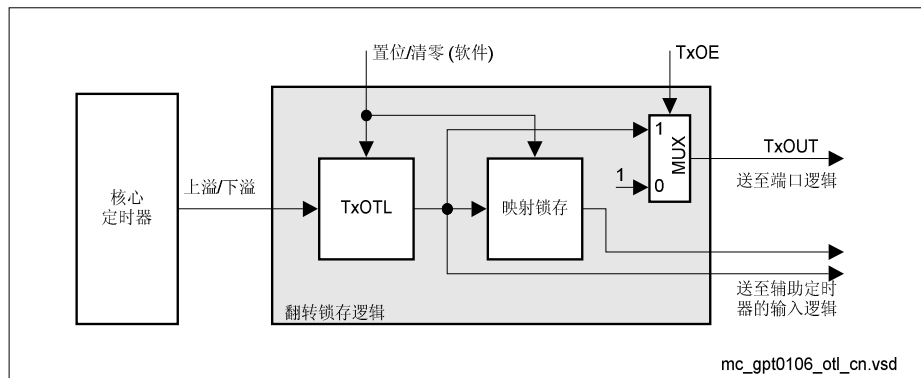


图 16-3 核心定时器 T3 翻转锁存逻辑框图

### 16.1.2 GPT1 核心定时器 T3 的工作模式

定时器 T3 可工作在下面几种模式之一。

#### 定时器 T3 工作在定时器模式

将寄存器 T3CON 中的位域 T3M 设置为 000B，核心定时器 T3 工作在定时器模式。定时器模式下，模块输入时钟  $f_{GPT}$  经两个可编程的预分频因子（由寄存器 T3CON 中的位域 BPS1 和 T3I 控制）分频后，用作 T3 的输入时钟信号。输入时钟选择的具体内容请参阅[章节 16.1.5](#)。

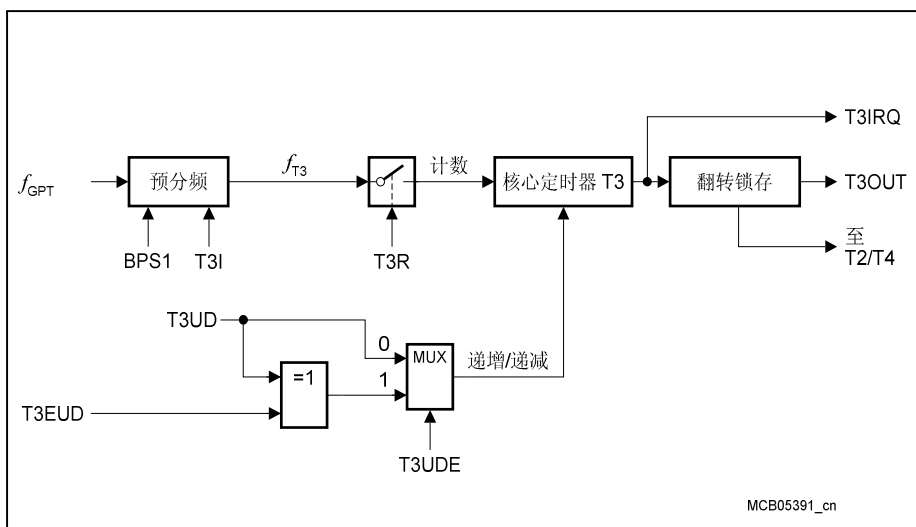
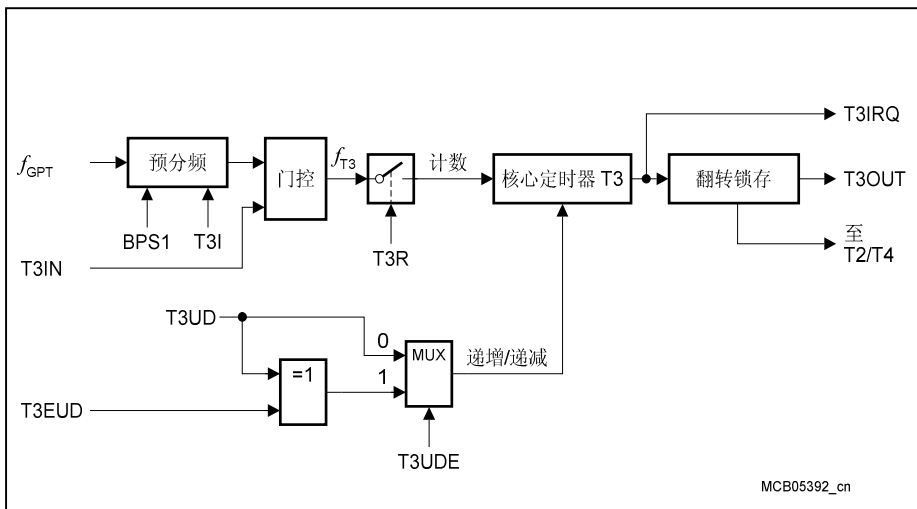


图 16-4 定时器模式下核心定时器 T3 框图

### 门控定时器模式

将寄存器 **T3CON** 中的位域 **T3M** 设置为 **010<sub>B</sub>** 或 **011<sub>B</sub>**，核心定时器 **T3** 工作在门控定时器模式。位 **T3M.0** (**T3CON.3**) 选择门控输入的有效电平。门控定时器模式下输入时钟的频率选择和定时器模式下相同（请参阅[章节 16.1.5](#)），但该模式下定时器的输入时钟受外部输入引脚 **T3IN**（定时器 **T3** 的外部输入）门控控制。

**T3** 工作在该模式时，引脚 **T3IN** 必须被配置为输入。



**图 16-5 门控定时器模式下核心定时器 T3 框图**

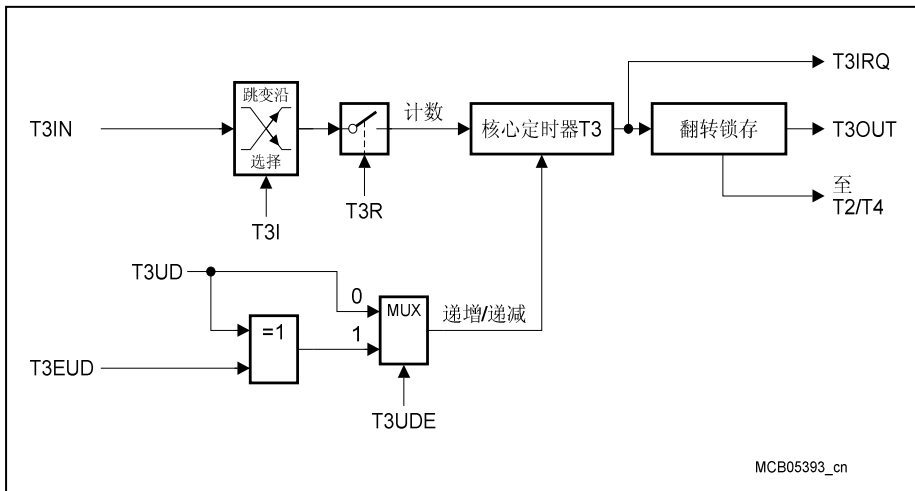
若 **T3M = 010<sub>B</sub>**，引脚 **T3IN** 为低电平时定时器被使能，为高电平时定时器被终止。若 **T3M = 011<sub>B</sub>**，**T3IN** 必须保持高电平才能使能定时器。此外，通过软件修改 **T3R** 可控制开启或关闭定时器。只有当 **T3R** 为 **1** 且门控信号有效时定时器才可以工作。若 **T3R** 为 **0** 或者门控信号无效，定时器将停止工作。

*注：引脚 **T3IN** 上门控信号的跳变不会引发中断请求。*



## 计数器模式

将寄存器 **T3CON** 中的位域 **T3M** 设置为 **001<sub>B</sub>**，核心定时器 **T3** 工作在计数器模式。计数器模式下，外部输入引脚 **T3IN** 上的跳变用作定时器 **T3** 的计数时钟。**T3IN** 上的正跳变、负跳变或任意跳变均可触发定时器递增或递减计数。控制寄存器 **T3CON** 中的位域 **T3I** 控制选择触发跳变（见**表 16-2**）。



**图 16-6 计数器模式下核心定时器 T3 方框图**

**表 16-2 GPT1 核心定时器 T3（计数器模式）输入沿选择**

<b>T3I</b>	<b>计数器递增/递减计数触发沿</b>
000 <sub>B</sub>	无，计数器 T3 被禁止
001 <sub>B</sub>	T3IN 上的正跳变（上升沿）
010 <sub>B</sub>	T3IN 上的负跳变（下降沿）
011 <sub>B</sub>	T3IN 上的任意跳变（上升沿或下降沿）
1XX <sub>B</sub>	保留，不要使用该组合

计数器模式下，必须将引脚 **T3IN** 配置为输入。计数器模式允许的最大输入频率取决于所选择的预分频因子。为了确保能够正确识别 **T3IN** 上计数输入信号的跳变，输入

电平必须至少保持（高或低）规定数目的模块时钟周期之后才能改变。具体内容请参阅章节 16.1.5。

### 增量接口模式

将寄存器 T3CON 中的位域 T3M 设置为 110<sub>B</sub> 或 111<sub>B</sub>，核心定时器 T3 工作在增量接口模式。增量接口模式下，与核心定时器 T3 相关的两个输入信号（T3IN，T3EUD）用作增量编码器的接口。外部输入引脚的其中一个或这两个信号上的每次跳变都触发 T3 计数，从而提供了 2 倍或 4 倍于编码器输入的计数精度。

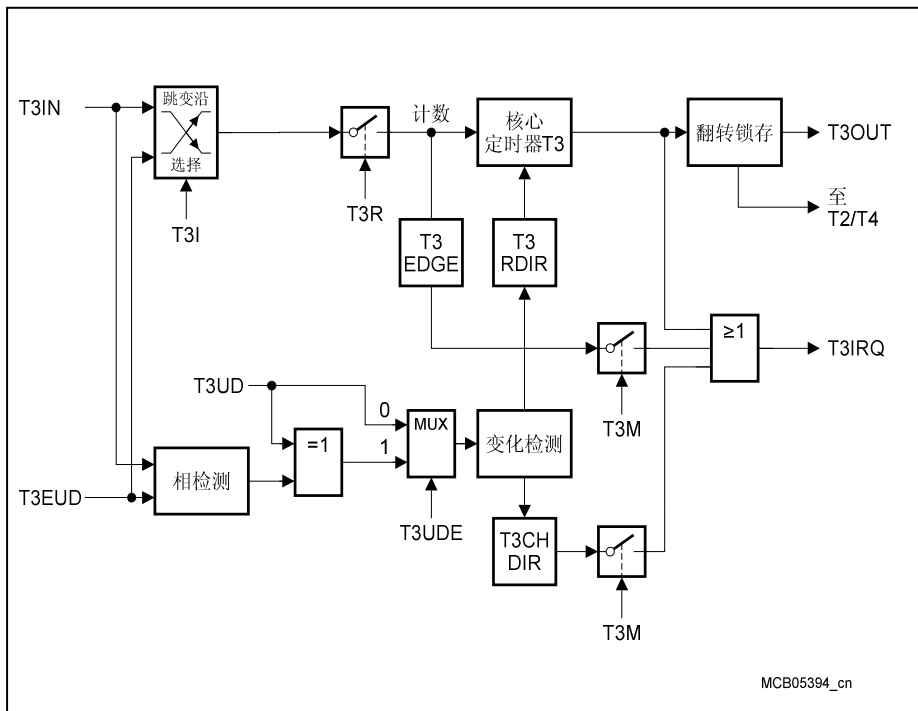


图 16-7 增量接口模式下核心定时器 T3 的框图

控制寄存器 T3CON 中的位域 T3I 控制选择触发沿（见表 16-3）。两个输入信号的跳变序列经评估后产生计数脉冲和方向信号。因此，根据增量编码器的速度和方向可自动修改 T3 的值，该数值始终表示编码器的当前位置。

产生中断请求（T3IRQ）的方式可以有以下选择：旋转检测模式下（T3M = 110<sub>B</sub>），每次 T3 的计数方向改变时产生中断请求；边沿检测模式下（T3M = 111<sub>B</sub>），每次检测到 T3 的计数沿时产生中断请求。寄存器 T3CON 中的状态位 T3RDIR、T3CHDIR 和 T3EDGE 分别监控 T3 的计数方向、计数方向的变化和计数请求。

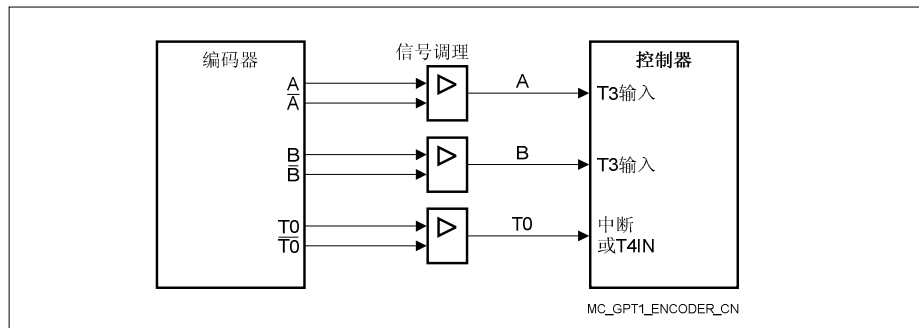
**表 16-3      核心定时器 T3（增量接口模式）输入沿选择**

T3I	计数器递增/递减计数触发沿
000 <sub>B</sub>	无，计数器 T3 被禁止
001 <sub>B</sub>	T3IN 上的任意跳变（上升沿或下降沿）
010 <sub>B</sub>	T3EUD 上的任意跳变（上升沿或下降沿）
011 <sub>B</sub>	T3 任意一个输入引脚（T3IN 或 T3EUD）上的任意跳变（上升沿或下降沿）
1XX <sub>B</sub>	保留，不要使用该组合

增量编码器无需外部接口逻辑、它可直接和 XE16xyM 相连。不过，在标准系统中会使用比较器将编码器的差分输出（比如 A， $\bar{A}$ ）转换为数字信号（比如 A），从而大大增强了抗噪能力。

*注：编码器的第三个输出信号 T0（指示机械零位置）可以和 XE16xyM 外部中断输入相连，触发定时器 T3 复位（例如将 ZEROS 通过 PEC 传送至 T3）。*

*如果输入 T4IN 可供使用，T0 能和 T4IN 相连，无需中断就可以自动清除 T3。*



**图 16-8      编码器和 XE16xyM 的连接**

要执行增量接口操作，必须满足以下条件：

- 位域 T3M 必须为 110<sub>B</sub> 或 111<sub>B</sub>
- 必须将引脚 T3IN 和 T3EUD 配置为输入。
- 如果和 T0 相连，引脚 T4IN 必须配置为输入。
- 位 T3UDE 必须为 1，从而允许由外部信号来控制计数方向

增量接口模式下允许的最大计数频率取决于所选择的预分频因子。为了确保能够正确识别输入信号的任意一个跳变，输入电平必须至少保持（高或低）规定数目的模块时钟周期之后才能改变。具体内容请参阅[章节 16.1.5](#)。

由于增量接口模式下所评估的两个输入信号具有 90°相移，其最大输入频率可为最大计数频率的一半。

增量接口模式下，可从输入信号变化的序列中自动获取计数方向信息；该序列和所连接的传感器的旋转方向相对应。各种可能的信号变化组合总结如[表 16-4](#)。

**表 16-4 GPT1 核心定时器 T3（增量接口模式）计数方向**

另一个输入信号 的电平 (T3EUD/T3IN)	T3IN 输入		T3EUD 输入	
	上升沿	下降沿	上升沿	下降沿
高电平	递减计数	递增计数	递增计数	递减计数
低电平	递增计数	递减计数	递减计数	递增计数

[图 16-9](#) 和 [图 16-10](#) 举例说明计数信号的产生，以及在计数方向的控制下 T3 如何工作，同时说明如何进行输入信号的抖动补偿（传感器位于切换点附近时可能会出现抖动现象）。

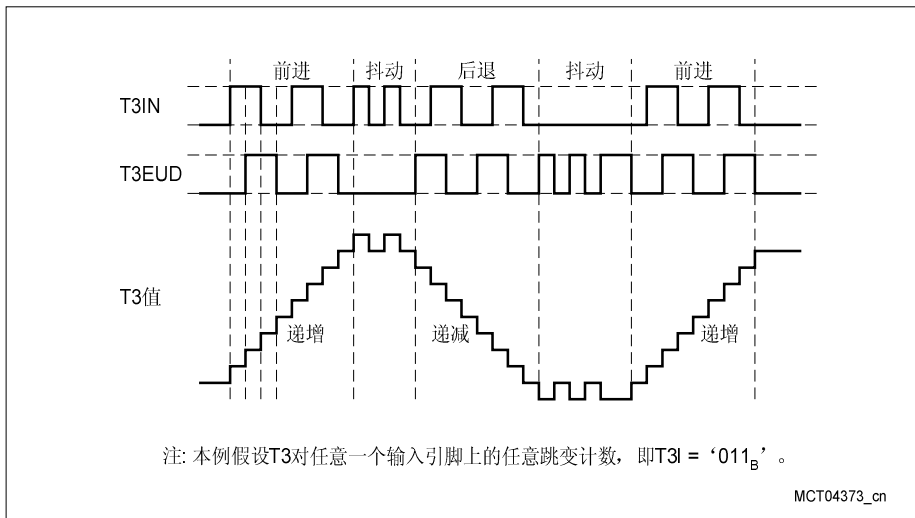


图 16-9 增量编码器信号评估, 2 个计数输入

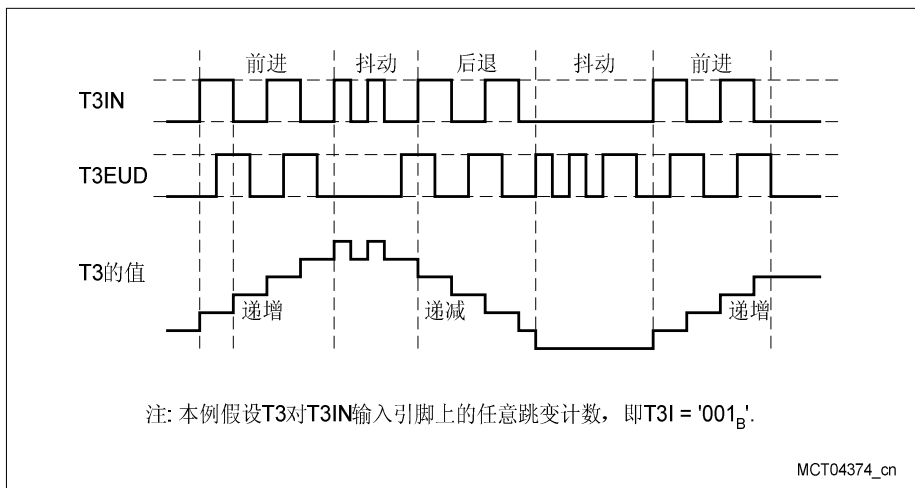


图 16-10 增量编码器信号评估, 1 个计数输入

注：工作在增量接口模式下的定时器 T3 可自动提供传感器的当前位置信息。通过测量输入信号的周期值可获取传感器的动态信息（速度、加速度、减速度）。（请参考页 16-58 “组合捕获模式”）。

### 16.1.3 GPT1 辅助定时器 T2/T4 的控制

辅助定时器 T2 和 T4 功能相同。它们可被配置为定时器模式、门控定时器模式、计数器模式或增量接口模式。定时器输入时钟的频率选择以及计数输入信号的选择和核心定时器 T3 相同。除了这四种基本工作模式，辅助定时器还可以和核心定时器级联工作，或者用作核心定时器的重载或捕获寄存器。辅助定时器的启动/终止功能可由 T3 的运行位远程控制，从而可控制多个定时器同步运行。

辅助定时器 T2 或 T4 的当前计数值可分别从计数寄存器 T2 或 T4 中读取。这些寄存器也可由 CPU 写入，例如 CPU 设置定时器的初始值。

通过可位寻址控制寄存器 T2CON 和 T4CON（结构完全相同）分别配置定时器 T2 和 T4。请注意：模块 GPT1 中三个定时器均具备的功能由各自控制寄存器中相同位置的位元/域、以相同的方式进行控制。

*注：辅助定时器无输出翻转锁存，无复用输出功能。*

#### GPT12E\_T2CON

##### 定时器 T2 控制寄存器

##### SFR (FF40H/A0H)

##### 复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T2 R DIR	T2 CH DIR	T2 ED GE	T2 IR DIS	-	-	T2 RC	T2 UDE	T2 UD	T2R	T2M			T2I		
rw	rwh	rwh	rw	-	-	rw	rw	rw	rw	rw			rw		

符号	位序号	类型	功能描述
T2I	[2:0]	rw	<b>定时器 T2 输入参数选择</b> 不同工作模式的输入参数选择请分别参阅相关内容： <b>表 16-7</b> 对应定时器模式和门控定时器模式 <b>表 16-2</b> 对应计数器模式 <b>表 16-3</b> 对应增量接口模式
T2M	[5:3]	rw	<b>定时器 T2 模式控制</b> （基本工作模式） 000 <sub>B</sub> 定时器模式 001 <sub>B</sub> 计数器模式 010 <sub>B</sub> 门控定时器模式，门控信号低电平有效 011 <sub>B</sub> 门控定时器模式，门控信号高电平有效 100 <sub>B</sub> 重载模式

符号	位序号	类型	功能描述
			101 <sub>B</sub> 捕获模式 110 <sub>B</sub> 增量接口模式（旋转检测） 111 <sub>B</sub> 增量接口模式（边沿检测）
<b>T2R</b>	6	rw	<b>定时器 T2 运行控制</b> 0 <sub>B</sub> 定时器 T2 停止 1 <sub>B</sub> 定时器 T2 运行 <i>注：只有 T2RC = 0 时该位才可控制定时器 T2。</i>
<b>T2UD</b>	7	rw	<b>定时器 T2 递增/递减计数控制<sup>1)</sup></b> 0 <sub>B</sub> 定时器 T2 递增计数 1 <sub>B</sub> 定时器 T2 递减计数
<b>T2UDE</b>	8	rw	<b>定时器 T2 外部递增/递减计数使能<sup>1)</sup></b> 0 <sub>B</sub> 输入 T2EUD 被断开 1 <sub>B</sub> 计数方向受输入 T2EUD 的控制
<b>T2RC</b>	9	rw	<b>定时器 T2 远程控制位</b> 0 <sub>B</sub> 定时器 T2 由自身的运行位 T2R 控制 1 <sub>B</sub> 定时器 T2 由核心定时器 T3 的运行位 T3R 控制，不由 T2R 控制。
<b>T2IRDIS</b>	12	rw	<b>定时器 T2 中断请求禁止控制</b> 0 <sub>B</sub> 增量接口模式下，由 T2CHDIR 和 T2EDGE 产生中断请求被使能 1 <sub>B</sub> 增量接口模式下，由 T2CHDIR 和 T2EDGE 产生中断请求被禁止
<b>T2EDGE</b>	13	rwh	<b>定时器 T2 计数沿检测标志</b> 每次检测到计数沿时该位被置位。T2EDGE 必须由软件清零。 0 <sub>B</sub> 未检测到计数沿 1 <sub>B</sub> 已检测到计数沿
<b>T2CHDIR</b>	14	rwh	<b>定时器 T2 计数方向改变标志</b> 每次定时器 T2 计数方向改变时该位被置位。 T2CHDIR 必须由软件清零。



符号	位序号	类型	功能描述
			0 <sub>B</sub> 未检测到计数方向改变 1 <sub>B</sub> 已检测到计数方向改变
T2RDIR	15	rh	定时器 T2 旋转方向标志 0 <sub>B</sub> 定时器 T2 递增计数 1 <sub>B</sub> 定时器 T2 递减计数

1) 位 T2UD 和 T2UDE 的编码见 [表 16-1](#)。

### GPT12E\_T4CON

定时器 T4 控制寄存器

SFR (FF44<sub>H</sub>/A2<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T4 R DIR	T4 CH DIR	T4 ED GE	T4 IR DIS	CLR T3 EN	CLR T2 EN	T4 RC	T4 UDE	T4 UD	T4R	T4M			T4I		
rh	rwh	rwh	rw	rw	rw	rw	rw	rw	rw	rw			rw		

符号	位序号	类型	功能描述
T4I	[2:0]	rw	定时器 T4 输入参数选择 不同工作模式的输入参数选择请分别参阅相关内容: <a href="#">表 16-7</a> 对应定时器模式和门控定时器模式 <a href="#">表 16-2</a> 对应计数器模式 <a href="#">表 16-3</a> 对应增量接口模式
T4M	[5:3]	rw	定时器 T4 模式控制 (基本工作模式) 000 <sub>B</sub> 定时器模式 001 <sub>B</sub> 计数器模式 010 <sub>B</sub> 门控定时器模式, 门控信号低电平有效 011 <sub>B</sub> 门控定时器模式, 门控信号高电平有效 100 <sub>B</sub> 重载模式 101 <sub>B</sub> 捕获模式 110 <sub>B</sub> 增量接口模式 (旋转检测) 111 <sub>B</sub> 增量接口模式 (边沿检测)

符号	位序号	类型	功能描述
<b>T4R</b>	6	rw	<b>定时器 T4 运行控制</b> 0 <sub>B</sub> 定时器 T4 停止 1 <sub>B</sub> 定时器 T4 运行 <i>注：只有 T4RC = 0 时该位才可控制定时器 T4。</i>
<b>T4UD</b>	7	rw	<b>定时器 T4 递增/递减计数控制<sup>1)</sup></b> 0 <sub>B</sub> 定时器 T4 递增计数 1 <sub>B</sub> 定时器 T4 递减计数
<b>T4UDE</b>	8	rw	<b>定时器 T4 外部递增/递减计数使能<sup>1)</sup></b> 0 <sub>B</sub> 输入 T4EUD 被断开 1 <sub>B</sub> 计数方向受输入 T4EUD 的控制
<b>T4RC</b>	9	rw	<b>定时器 T4 远程控制位</b> 0 <sub>B</sub> 定时器 T4 由自身的运行位 T4R 控制 1 <sub>B</sub> 定时器 T4 由核心定时器 T3 的运行位 T3R 控制，不由 T4R 控制。
<b>CLRT2EN</b>	10	rw	<b>清除定时器 T2 使能</b> 所选择的 T4EUD 输入上出现下降沿时，使能自动清除 T2 的操作 0 <sub>B</sub> T4EUD 对 T2 无影响 1 <sub>B</sub> T4EUD 上的下降沿将会清除定时器 T2
<b>CLRT3EN</b>	11	rw	<b>清除定时器 T3 使能</b> 所选择的 T4IN 输入上出现下降沿时，使能自动清除 T3 的操作。 0 <sub>B</sub> T4IN 对 T3 无影响 1 <sub>B</sub> T4IN 上的下降沿将会清除定时器 T3
<b>T4IRDIS</b>	12	rw	<b>定时器 T4 中断请求禁止控制</b> 0 <sub>B</sub> 增量接口模式下，由 T4CHDIR 和 T4EDGE 产生中断请求被使能 1 <sub>B</sub> 增量接口模式下，由 T4CHDIR 和 T4EDGE 产生中断请求被禁止
<b>T4EDGE</b>	13	rwh	<b>定时器 T4 计数沿检测标志</b>

符号	位序号	类型	功能描述
			每次检测到计数沿时该位被置位。T4EDGE 必须由软件清零。 0 <sub>B</sub> 未检测到计数沿 1 <sub>B</sub> 已检测到计数沿
<b>T4CHDIR</b>	14	rwh	<b>定时器 T4 计数方向改变标志</b> 每次定时器 T4 计数方向改变时该位被置位。T4CHDIR 必须由软件清零。 0 <sub>B</sub> 未检测到计数方向改变 1 <sub>B</sub> 已检测到计数方向改变
<b>T4RDIR</b>	15	rh	<b>定时器 T4 旋转方向标志</b> 0 <sub>B</sub> 定时器 T4 递增计数 1 <sub>B</sub> 定时器 T4 递减计数

1) 位 T4UD 和 T4UDE 的编码见 [表 16-1](#)。

### 定时器 T2/T4 运行控制

可通过两种不同的方式（软件）启动或终止辅助定时器 T2 和 T4 工作：

- 由相应的定时器运行位（T2R 或 T4R）控制，此时要求  $TxRC = 0$ 。
- 由核心定时器的控制位（T3R）控制，此时要求相应的远程控制位必须被置位（ $TxRC = 1$ ）。

所选择的运行控制位对 T2/T4 的所有工作模式均有效。该位被置位将启动定时器工作；该位被清除将终止定时器工作。

门控定时器模式下，只有当所选择的运行位被置位且门控信号有效（设置为高或低电平）时，定时器才能工作。

*注：若选择远程控制 T2/T4 运行，T3R 将同步启动或终止定时器 T3 和辅助定时器。*

### 计数方向控制

GPT1 模块各定时器（核心定时器和辅助定时器）计数方向的控制方式相同：均由软件控制、或由外部输入引脚 TxEUD 控制，具体内容见 [表 16-1](#)。

*注：引脚 TxEUD 用作外部计数方向控制输入时，必须被配置为输入引脚。*

#### 16.1.4 GPT1 辅助定时器 T2/T4 的工作模式

除极少数例外情况，辅助定时器在基本工作模式下的操作和核心定时器几乎完全相同。此外，T2/T4 可工作在一些组合工作模式下。

##### 定时器 T2 和 T4 工作在定时器模式

将寄存器 TxCON 中的位域 TxM 设置为 000B，辅助定时器 Tx 工作在定时器模式。

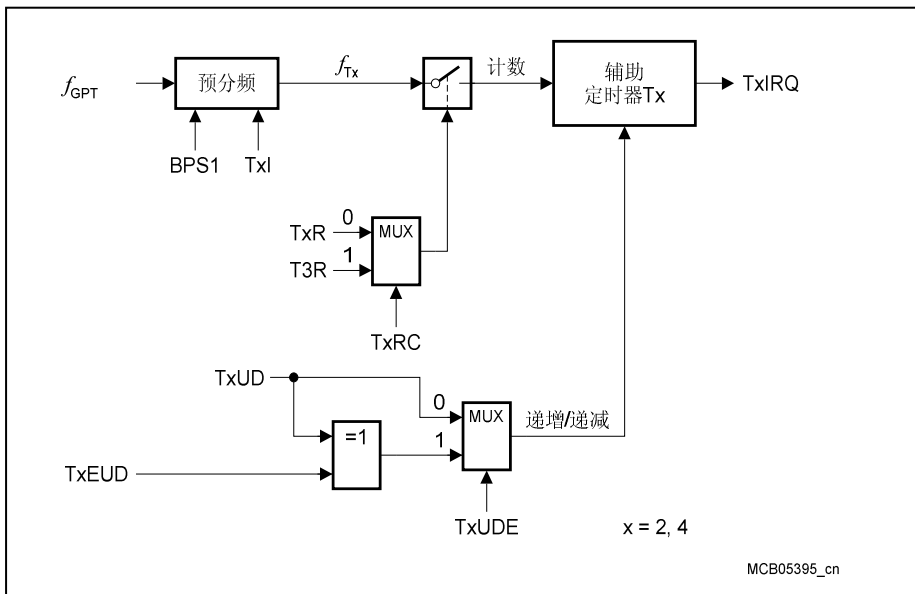
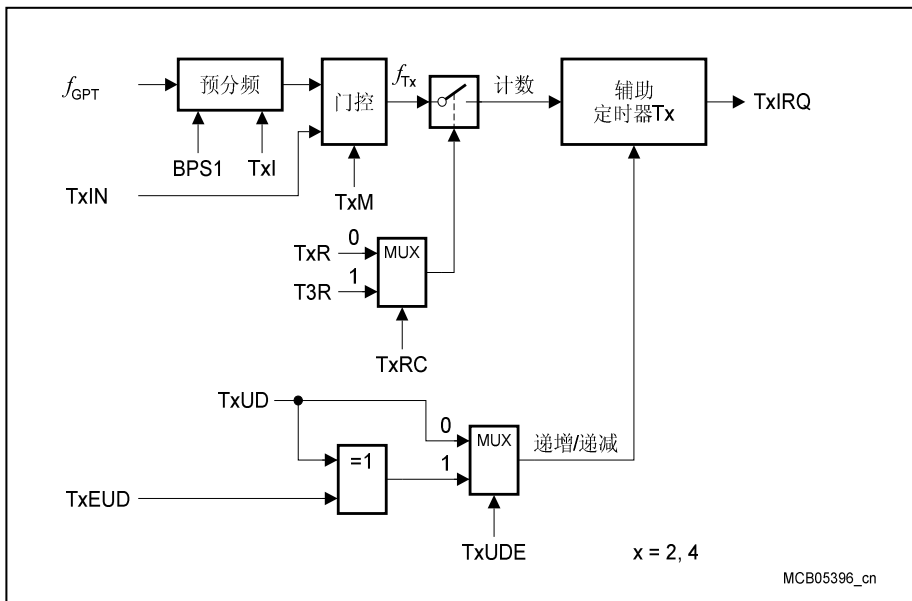


图 16-11 定时器模式下辅助定时器框图

### 定时器 T2 和 T4 工作在门控定时器模式

将寄存器 TxCON 中的位域 TxM 设置为 010<sub>B</sub> 或 011<sub>B</sub>，辅助定时器 Tx 工作在门控定时器模式。位 TxM.0 (TxCON.3) 选择门控输入的有效电平。

注：TxIN 上门控信号的跳变不会引发中断请求。



**图 16-12 门控定时器模式下辅助定时器框图**

注：T2 和 T4 输出不经翻转锁存。

可通过自身或 T3 的运行位本地或远程控制辅助定时器的启动/终止。

### 定时器 T2 和 T4 工作在计数器模式

将寄存器 TxCON 中的位域 TxM 设置为 001<sub>B</sub>，辅助定时器 Tx 工作在计数器模式。计数器模式下，外部输入引脚 TxIN 跳变、或定时器 T3 的溢出翻转锁存 T3OTL 跳变时，可触发辅助定时器计数。TxIN 或 T3OTL 上的正跳变、负跳变或任意跳变可触发定时器递增或递减计数。控制寄存器 TxCON 中的位域 TxI 控制选择触发事件（见表 16-5）。

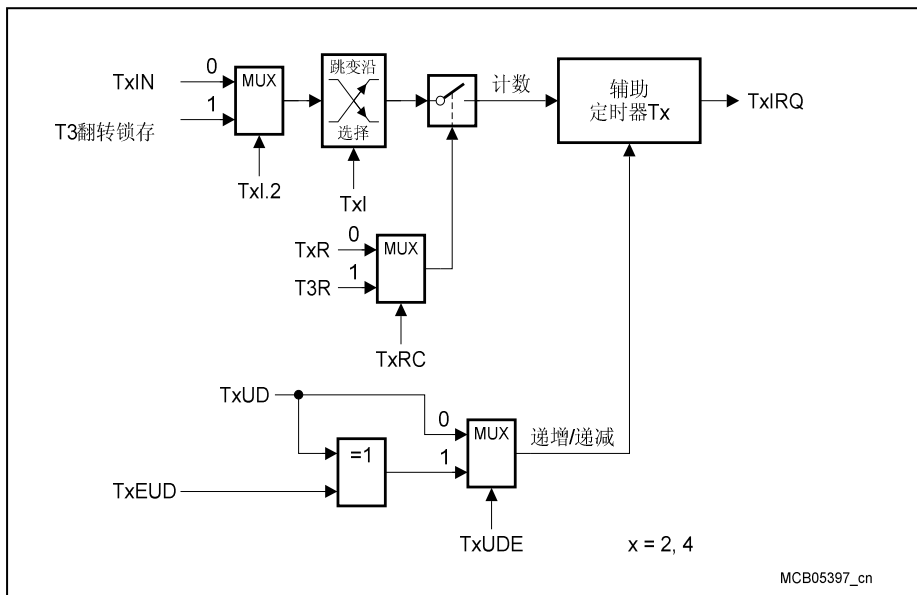


图 16-13 计数器模式下辅助定时器框图

表 16-5 GPT1 辅助定时器（计数器模式）输入沿选择

T2/T4I	计数器递增/递减计数触发沿
X00 <sub>B</sub>	无。计数器 Tx 被禁止
001 <sub>B</sub>	TxIN 上的正跳变（上升沿）
010 <sub>B</sub>	TxIN 上的负跳变（下降沿）
011 <sub>B</sub>	TxIN 上的任意跳变（上升沿或下降沿）
101 <sub>B</sub>	T3 翻转锁存 T3OTL 上的正跳变（上升沿）
110 <sub>B</sub>	T3 翻转锁存 T3OTL 上的负跳变（下降沿）
111 <sub>B</sub>	T3 翻转锁存 T3OTL 上的任意跳变（上升沿或下降沿）

注：只有 T3 上溢/下溢引发的 T3OTL 的状态跳变才能触发 T2/T4 计数；软件修改 T3OTL 的状态不会触发 T2/T4 计数。

计数器模式下，必须将引脚 TxIN 配置为输入。计数器模式下允许的最大输入频率取决于所选择的预分频因子。为了确保能够正确识别 TxIN 上计数输入信号的跳变，输入电平必须至少保持（高或低）规定数目的模块时钟周期之后才能改变。具体内容请参阅 [章节 16.1.5](#)。

### 定时器 T2 和 T4 工作在增量接口模式

将寄存器 TxCON 中的位域 TxM 设置为 110<sub>B</sub> 或 111<sub>B</sub>，辅助定时器 Tx 工作在增量接口模式。增量接口模式下，与辅助定时器 Tx 相关的两个输入信号（TxIN，TxEUD）用作增量编码器的接口。其中一个或这两个信号上的跳变都触发 Tx 计数，从而提供了 2 倍或 4 倍于编码器输入的计数精度。

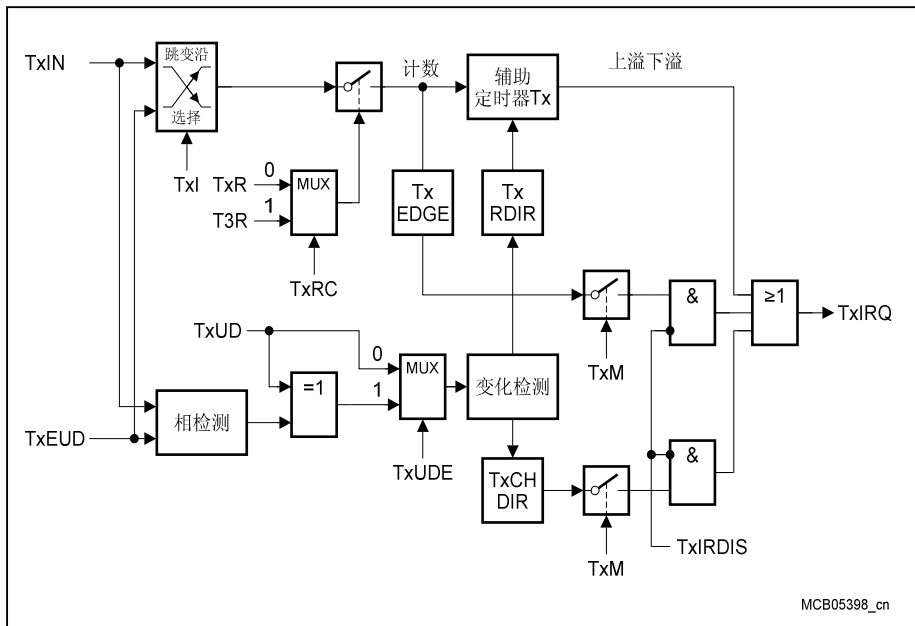


图 16-14 增量接口模式下辅助定时器 Tx 框图

增量接口模式下辅助定时器 T2 和 T4 的操作及中断产生和定时器 T3 相同。其运行描述、图和表格也同样适用。

注：工作在增量接口模式下的定时器 T2 和 T4 可自动提供传感器的当前位置信息。要获得动态信息（速度、加速度、减速度），请参考页 16-58 “组合捕获模式”。



## 定时器级联

溢出翻转锁存位 T3OTL 用作计数器模式下辅助定时器的计数时钟时，核心定时器 T3 和辅助定时器级联工作。定时器级联可构成 32 位或 33 位定时器/计数器，这取决于选择何种 T3OTL 跳变触发辅助定时器计数。

- **32 位定时器/计数器：**若选择 T3OTL 的任意跳变触发辅助定时器计数，核心定时器 T3 在每次上溢/下溢时，辅助定时器计数。因此 T2/T4 和 T3 构成 32 位定时器。
- **33 位定时器/计数器：**若仅选择 T3OTL 的正跳变或负跳变触发辅助定时器计数，核心定时器 T3 在每两次上溢/下溢后，辅助定时器计数一次。因此它们构成 33 位定时器（16 位核心定时器 + T3OTL + 16 位辅助定时器）。

只要 T3OTL 不被软件修改，它即表示内部翻转锁存器的状态，可被视为 33 位定时器的组成部分。

两个级联定时器的计数方向可以不同，从而为用户提供了多种不同的配置组合。

T3 代表级联定时器的低 16 位，在此情况下，T3 可工作在定时器模式、门控定时器模式或计数器模式。

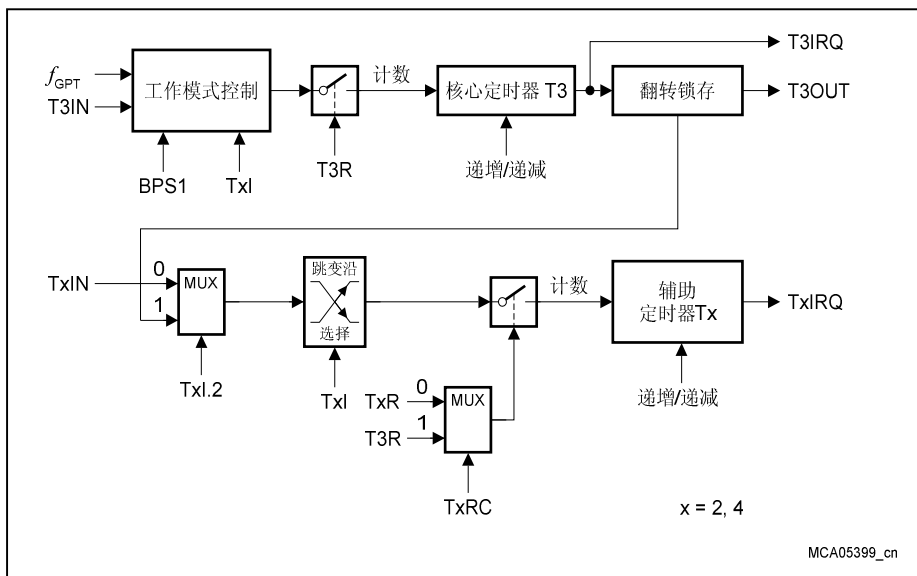


图 16-15 核心定时器 T3 和辅助定时器级联工作

### 辅助定时器工作在重载模式

将寄存器 TxCON 中的位域 TxM 设置为 100B，辅助定时器 Tx 工作在重载模式。重载模式下，两个信号中的任意一个可触发重载，将辅助定时器寄存器的内容重新装入核心定时器 T3 中。触发信号的选择和计数模式下计数时钟的选择方式相同（见表 16-5），即辅助定时器的输入 TxIN 跳变时、或翻转锁存 T3OTL 跳变时可触发重载。

注：辅助定时器（T2 或 T4）工作在重载模式时，定时器的停止不受自身运行标志 T2R 或 T4R 控制。

若定时器的输入引脚TxIN用于触发重载操作，必须将其配置为输入。

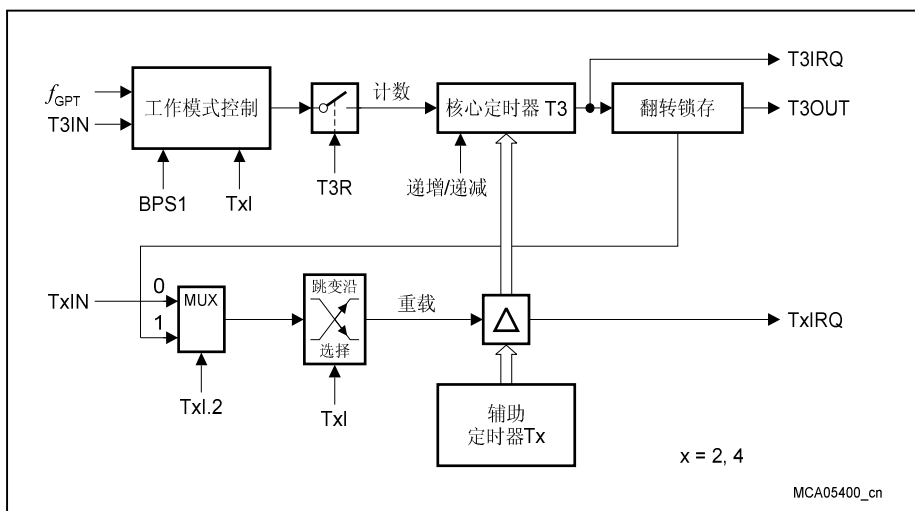


图 16-16 重载模式下 GPT1 的辅助定时器

出现触发信号时，辅助定时器寄存器（T2 或 T4）的内容重新装入 T3，相应的中断请求标志（T2IR 或 T4IR）被置位。

注：若选择 T3OTL 的跳变作为触发信号产生触发时，中断请求标志 T3IR 也将被置位，指示 T3 上溢或下溢。软件修改 T3OTL 不会触发 T2/T4 的计数功能。

为了确保能够正确识别 TxIN 上重载输入信号的跳变，输入电平必须至少保持（高或低）规定数目的模块时钟周期之后才能改变。具体内容请参阅[章节 16.1.5](#)。

由 T3OTL 触发的重载工作模式可有多种不同的配置。选择不同的有效跳变沿，将执行以下功能：

- 若选择 T3OTL 的任意跳变触发重载，T3 在每次上溢或下溢时，辅助定时器的值重新装入核心定时器。此配置为标准重载模式（上溢/下溢时发生重载）
- 若选择 T3OTL 的正跳变或负跳变触发重载，T3 在每两次上溢或下溢后，辅助定时器的值重新装入核心定时器。
- 使用“单沿跳变”重载模式可实现非常灵活的脉宽调制（PWM）。即设定一个辅助定时器在 T3OTL 正跳变时重载；另一个辅助定时器在 T3OTL 负跳变时重载。通过这种组合，核心定时器可轮流从两个辅助定时器中获得重载值。

**图 16-17** 举例说明如何利用“单沿跳变”重载模式产生 PWM 信号。T2 定义了 PWM 信号的接通时间（正跳变触发重载），T4 定义了 PWM 信号的关闭时间（负跳变触发重载）。若 T3OE = 1，PWM 信号可从引脚 T3OUT 输出。利用该方法，PWM 信号的接通/关闭时间可在较大范围内变化。

*注：可通过软件改变输出翻转锁存 T3OTL 的值，进而修改 PWM 信号。*

*但该操作不会触发 T3 重载。*

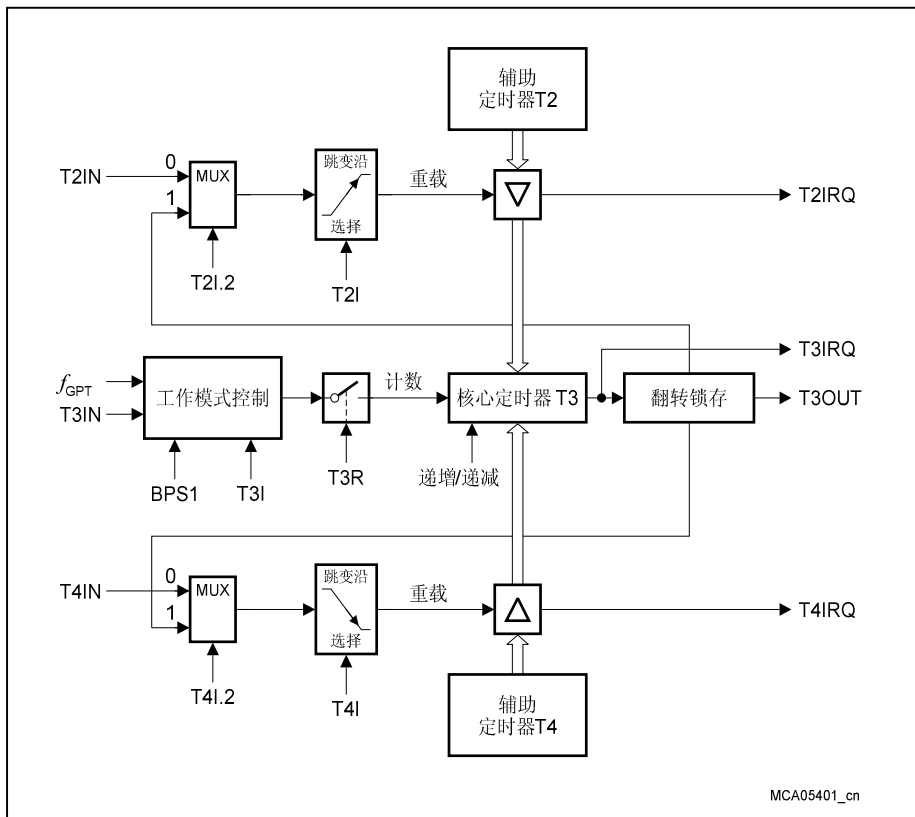


图 16-17 用于产生 PWM 的 GPT1 定时器重载配置

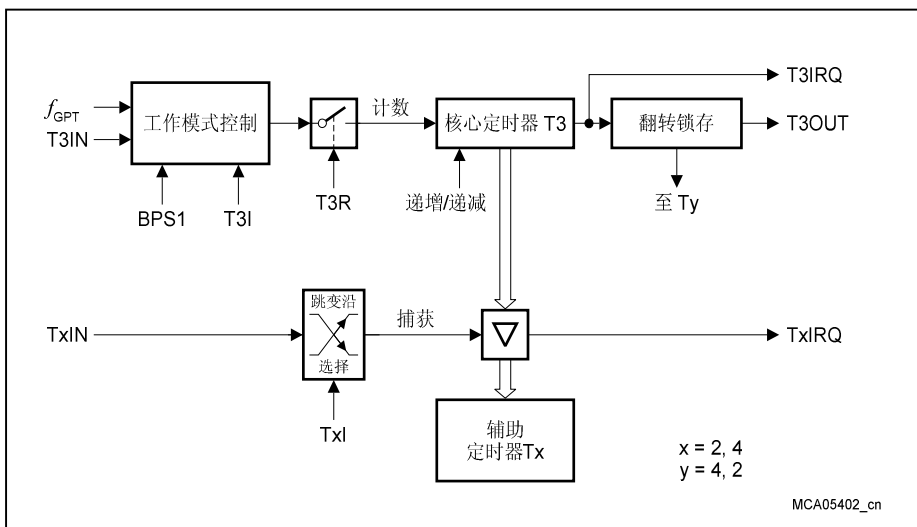
注：应该尽量避免为两个辅助定时器选择相同的重载触发事件（尽管该情况可能发生）。发生该情况时，两个重载寄存器会试图同时加载核心定时器。此时丢弃 T2 的值；将 T4 的值载入 T3。

### 辅助定时器工作在捕获模式

将寄存器 TxCON 中的位域 TxM 设置为 101<sub>B</sub>，辅助定时器 Tx 工作在捕获模式。捕获模式下，辅助定时器的外部输入引脚 TxIN 发生跳变时，核心定时器 T3 的值被锁入辅助定时器寄存器。捕获触发信号可以是 TxIN 的正跳变、负跳变或任意跳变。

TxI 中的低两位用于选择有效跳变沿（见**表 16-5**）。TxI 的位 2 与捕获模式无关，必须被清除（TxI.2 = 0）。

*注：辅助定时器（T2 或 T4）工作在捕获模式时，定时器停止不受自身运行标志 T2R 或 T4R 控制。*



**图 16-18 捕获模式下 GPT1 辅助定时器框图**

输入引脚 TxIN 上产生触发信号（所选择的跳变）时，核心定时器的内容被锁存到辅助定时器寄存器中，相关中断请求标志 TxIR 被置位。

捕获模式下，必须将输入引脚 TxIN 配置为输入。为了确保能够正确识别 TxIN 上捕获输入信号的跳变，输入电平必须至少保持（高或低）规定数目的模块时钟周期之后才能改变。具体内容请参阅**章节 16.1.5**。

### 16.1.5 GPT1 时钟信号控制

定时器模块 GPT1 的所有操作均由其基本时钟触发。系统时钟经过基本模块预分频产生该基本时钟，预分频因子由寄存器 T3CON 中的位域 BPS1 控制（见图 16-2）。有两种不同的方式产生计数时钟：

- **内部计数时钟**，经过预分频处理（预分频因子可编程设定）的 GPT1 基本时钟，该时钟用于（门控）定时器模式。
- **外部计数时钟**，取自定时器的输入引脚，该时钟用于计数器模式。

GPT1 模块的基本时钟决定了最大计数频率和定时器精度。

表 16-6 模块 GPT1 的基本时钟选择

模块预分频 <sup>1)</sup>	BPS1 = 01 <sub>B</sub>	BPS1 = 00 <sub>B</sub> <sup>2)</sup>	BPS1 = 11 <sub>B</sub>	BPS1 = 10 <sub>B</sub>
GPT1 的预分频因子: F(BPS1)	F(BPS1) = 4	F(BPS1) = 8	F(BPS1) = 16	F(BPS1) = 32
最大外部计数频率	f <sub>GPT</sub> /8	f <sub>GPT</sub> /16	f <sub>GPT</sub> /32	f <sub>GPT</sub> /64
输入信号稳定时间	4× t <sub>GPT</sub>	8× t <sub>GPT</sub>	16× t <sub>GPT</sub>	32× t <sub>GPT</sub>

1) 请注意位域 BPS1 为非线性编码。

2) 复位后的缺省值。

#### 内部计数时钟产生

定时器模式和门控定时器模式下，GPT1 的每个定时器的计数时钟来自经过预分频处理的 GPT1 基本时钟，该预分频因子可由定时器控制寄存器 TxCON 中的位域 TxI 编程设定。

定时器 Tx 的计数频率 f<sub>Tx</sub> 和相应计数精度 r<sub>Tx</sub> 随较低的时钟频率线性变化，计算公式如下：

$$f_{Tx} = \frac{f_{GPT}}{F(BPS1) \times 2^{\langle TxI \rangle}} \quad r_{Tx} [\mu s] = \frac{F(BPS1) \times 2^{\langle TxI \rangle}}{f_{GPT} [MHz]} \quad (16.1)$$

计数频率不仅取决于公共的模块时钟预分频因子 F(BPS1)；还取决于定时器各自的预分频因子 2<sup>⟨TxI⟩</sup>。GPT1 定时器的总预分频因子总结见表 16-7。

根据总预分频因子和系统频率计算得到定时器的各项参数（如计数时钟频率、计数时钟精度和计数周期），见表 16-8。请注意有些结果已四舍五入。

**表 16-7      GPT1 内部计数时钟的总预分频因子**

定时器 Tx 的 预分频因子	公共预分频因子 <sup>1)</sup>			
	BPS1 = 01 <sub>B</sub>	BPS1 = 00 <sub>B</sub>	BPS1 = 11 <sub>B</sub>	BPS1 = 10 <sub>B</sub>
<b>Txl = 000<sub>B</sub></b>	4	8	16	32
<b>Txl = 001<sub>B</sub></b>	8	16	32	64
<b>Txl = 010<sub>B</sub></b>	16	32	64	128
<b>Txl = 011<sub>B</sub></b>	32	64	128	256
<b>Txl = 100<sub>B</sub></b>	64	128	256	512
<b>Txl = 101<sub>B</sub></b>	128	256	512	1024
<b>Txl = 110<sub>B</sub></b>	256	512	1024	2048
<b>Txl = 111<sub>B</sub></b>	512	1024	2048	4096

1) 请注意位域 BPS1 为非线性编码。

**表 16-8      GPT1 定时器参数**

系统时钟 = 10 MHz			总预 分频 因子	系统时钟 = 40 MHz		
频率	精度	周期		频率	精度	周期
2.5 MHz	400 ns	26.21 ms	4	10.0 MHz	100 ns	6.55 ms
1.25 MHz	800 ns	52.43 ms	8	5.0 MHz	200 ns	13.11 ms
625.0 kHz	1.6 µs	104.9 ms	16	2.5 MHz	400 ns	26.21 ms
312.5 kHz	3.2 µs	209.7 ms	32	1.25 MHz	800 ns	52.43 ms
156.25 kHz	6.4 µs	419.4 ms	64	625.0 kHz	1.6 µs	104.9 ms
78.125 kHz	12.8 µs	838.9 ms	128	312.5 kHz	3.2 µs	209.7 ms
39.06 kHz	25.6 µs	1.678 s	256	156.25 kHz	6.4 µs	419.4 ms
19.53 kHz	51.2 µs	3.355 s	512	78.125 kHz	12.8 µs	838.9 ms
9.77 kHz	102.4 µs	6.711 s	1024	39.06 kHz	25.6 µs	1.678 s
4.88 kHz	204.8 µs	13.42 s	2048	19.53 kHz	51.2 µs	3.355 s
2.44 kHz	409.6 µs	26.84 s	4096	9.77 kHz	102.4 µs	6.711 s

外部计数时钟输入

用 GPT1 的基本时钟对外部输入信号采样（见 [图 16-2](#)）。为了确保信号能被正确识别，外部信号的当前输入电平值（高或低）必须至少保持一个完整的采样周期之后才能改变。若输入信号连续两次采样的电平值不同，即可识别到信号发生了跳变。因此，至少需要两个基本时钟周期对外部输入信号采样，故输入信号的最大频率一定不能高于基本时钟频率的一半。

GPT1 外部输入信号的参数限制总结见 [表 16-9](#)。

表 16-9 GPT1 外部输入信号的参数限制

系统时钟 = 10 MHz		输入 频率因子	GPT1 分频 因子 BPS1	输入信号 持续时间	系统时钟 = 40 MHz	
最大输入 频率	电平保持 稳定的 最短时间				最大输入 频率	电平保持 稳定的 最短时间
1.25 MHz	400 ns	$f_{GPT}/8$	<b>01<sub>B</sub></b>	$4 \times t_{GPT}$	5.0 MHz	100 ns
625.0 kHz	800 ns	$f_{GPT}/16$	<b>00<sub>B</sub></b>	$8 \times t_{GPT}$	2.5 MHz	200 ns
312.5 kHz	1.6 $\mu$ s	$f_{GPT}/32$	<b>11<sub>B</sub></b>	$16 \times t_{GPT}$	1.25 MHz	400 ns
156.25 kHz	3.2 $\mu$ s	$f_{GPT}/64$	<b>10<sub>B</sub></b>	$32 \times t_{GPT}$	625.0 kHz	800 ns

上表中的各项参数限制对 GPT1 的所有外部输入信号均有效，包括计数器模式下和增量接口模式下的外部计数信号、门控定时器模式下的门控输入信号以及外部方向信号。



## 16.1.6 GPT1 定时器寄存器

### GPT12E\_T2

定时器 T2 计数寄存器      **SFR (FE40<sub>H</sub>/20<sub>H</sub>)**      复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>T2</b>															
rwh															

符号	位序号	类型	功能描述
<b>T2</b>	[15:0]	rwh	定时器 <b>T2</b> 当前值 保存定时器 T2 的当前值。

### GPT12E\_T3

定时器 T3 计数寄存器      **SFR (FE42<sub>H</sub>/21<sub>H</sub>)**      复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>T3</b>															
rwh															

符号	位序号	类型	功能描述
<b>T3</b>	[15:0]	rwh	定时器 <b>T3</b> 当前值 保存定时器 T3 的当前值。

### GPT12E\_T4

定时器 T4 计数寄存器      **SFR (FE44<sub>H</sub>/22<sub>H</sub>)**      复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>T4</b>															
rwh															

符号	位序号	类型	功能描述
<b>T4</b>	[15:0]	rwh	定时器 <b>T4</b> 当前值 保存定时器 T4 的当前值。

### **16.1.7 GPT1 定时器的中断控制**

当定时器从  $FFFF_H$  上溢至  $0000_H$ （递增计数），或定时器从  $0000_H$  下溢至  $FFFF_H$ （递减计数）时，寄存器  $GPT12E\_TxIC$  ( $x = 2, 3, 4$ ) 中的中断请求标志被置位。若相应中断使能位被置位，将会产生指向相应定时器中断向量的中断，或者触发 PEC 操作。

三个定时器（T2、T3、T4）各对应一个中断控制寄存器。所有的中断控制寄存器与中断控制章节中所描述的中断寄存器结构相同。

## 16.2 定时器模块 GPT2

从编程人员的角度来看，GPT2 下面列出的一组特殊功能寄存器（SFR）组成。端口和方向寄存器中用作 GPT2 功能的部分用阴影标出。

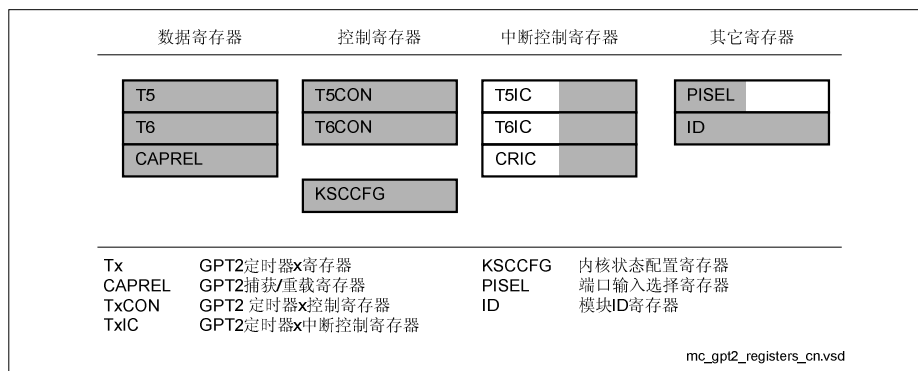


图 16-19 定时器模块 GPT2 的相关特殊功能寄存器（SFR）

GPT2 中的两个定时器（T5，T6）可工作在 3 种基本模式：定时器模式、门控定时器模式或计数器模式。所有定时器均可递增或递减计数。GPT2 的每个定时器由各自的控制寄存器 TxCON 控制。

每个定时器有一个对应的输入引脚 TxIN（复用引脚功能），用作门控定时器模式中的门控信号，或计数器模式中的计数输入。计数方向（递增/递减）可由软件设定，或由外部递增/递减控制输入引脚 TxEUD（复用引脚功能）上的信号动态改变。输出翻转锁存 T6OTL 指示核心定时器 T6 发生上溢/下溢，T6OTL 的状态可从引脚 T6OUT（复用引脚功能）输出。辅助定时器 T5 可以（通过 T6OTL）和核心定时器 T6 级联工作。

捕获/重载寄存器 CAPREL 可用于捕获定时器 T5 的内容，或者重载定时器 T6。系统支持一种特殊的模式，即通过寄存器 CAPREL 同时实现捕获、重载两种功能，采用该模式可实现变频。输入引脚 CAPIN，或 GPT1 的定时器 T3 的 T3IN 和 T3EUD 触发捕获；定时器 T6 上溢或下溢触发重载。定时器 T6 的溢出信号还可作为 CAPCOM 定时器的输入时钟。

定时器计数寄存器 T5 和 T6 位于不可位寻址的 SFR 区，CPU 访问这些寄存器，可读取或修改每个定时器的当前计数值（请参阅[章节 16.2.7](#)）。若定时器将执行递增、递减、重载或捕获操作时，CPU 恰好正在对（任意）某个定时器寄存器写入，CPU 的写操作则占优以确保结果正确。

由中断控制寄存器 TxIC 控制 GPT2 的中断产生，这些寄存器不在 GPT2 模块中。GPT2 的输入和输出线与 P3 和 P5 口的引脚相连。端口功能的控制寄存器位于端口模块中。

注：外部输入信号的时序要求请参阅[章节 16.2.6](#)，包括引脚在内的模块接口信号归纳见[章节 16.5](#)。

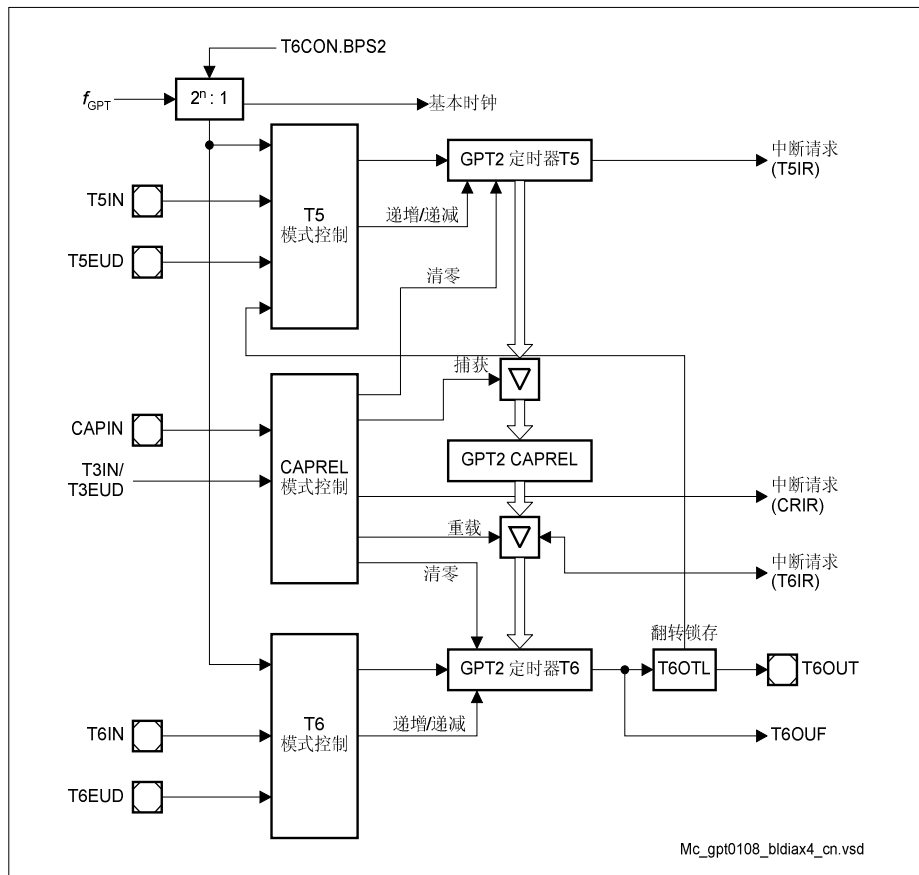


图 16-20 GPT2 框图

### 16.2.1 GPT2 核心定时器 T6 的控制

核心定时器 T6 的当前值可从计数寄存器 T6 中读取，该寄存器也可由 CPU 写入，例如 CPU 设置定时器的初始值。

由可位寻址控制寄存器 T6CON 配置和控制核心定时器 T6。

#### GPT12E\_T6CON

定时器 T6 控制寄存器

SFR (FF48H/A4H)

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T6 SR	T6 CLR	-	BPS2	T6 OTL	T6 OE	T6 UDE	T6 UD	T6R	T6M			T6I			
rw	rw	-	rw	rw	rw	rw	rw	rw	rw			rw			

符号	位序号	类型	功能描述
T6I	[2:0]	rw	<b>定时器 T6 输入参数选择</b> 不同工作模式的输入参数选择请分别参阅相关内容： <b>表 16-15</b> 对应定时器模式和门控定时器模式 <b>表 16-11</b> 对应计数器模式
T6M	[5:3]	rw	<b>定时器 T6 模式控制</b> （基本工作模式） 000 <sub>B</sub> 定时器模式 001 <sub>B</sub> 计数器模式 010 <sub>B</sub> 门控定时器模式，门控信号低电平有效 011 <sub>B</sub> 门控定时器模式，门控信号高电平有效 100 <sub>B</sub> 保留，不使用该组合 101 <sub>B</sub> 保留，不使用该组合 110 <sub>B</sub> 保留，不使用该组合 111 <sub>B</sub> 保留，不使用该组合
T6R	6	rw	<b>定时器 T6 运行控制位</b> 0 <sub>B</sub> 定时器 T6 停止运行 1 <sub>B</sub> 定时器 T6 运行
T6UD	7	rw	<b>定时器 T6 递增/递减计数控制</b> <sup>1)</sup> 0 <sub>B</sub> 定时器 T6 递增计数

符号	位序号	类型	功能描述
			1 <sub>B</sub> 定时器 T6 递减计数
<b>T6UDE</b>	8	rw	<b>定时器 T6 外部递增/递减计数使能<sup>1)</sup></b> 0 <sub>B</sub> 输入 T6EUD 被断开 1 <sub>B</sub> 计数方向受输入 T6EUD 的控制
<b>T6OE</b>	9	rw	<b>上溢/下溢输出使能</b> 0 <sub>B</sub> 复用输出功能被禁止 1 <sub>B</sub> T6 翻转锁存器的状态输出至引脚 T6OUT
<b>T6OTL</b>	10	rwh	<b>定时器 T6 溢出翻转锁存</b> 每次 T6 上溢/下溢时翻转。可由软件置位或复位。 （具体描述请参阅相关章节）
<b>BPS2</b>	[12:11]	rw	<b>模块 GPT2 预分频控制</b> 选择模块 GPT2 的基本时钟 （请参阅 <a href="#">章节 16.2.6</a> ） 00 <sub>B</sub> $f_{GPT}/4$ 01 <sub>B</sub> $f_{GPT}/2$ 10 <sub>B</sub> $f_{GPT}/16$ 11 <sub>B</sub> $f_{GPT}/8$
<b>T6CLR</b>	14	rwh	<b>定时器 T6 清零使能位</b> 0 <sub>B</sub> 捕获事件不触发定时器 T6 清零 1 <sub>B</sub> 捕获事件触发定时器 T6 清零
<b>T6SR</b>	15	rw	<b>定时器 T6 重载模式使能</b> 0 <sub>B</sub> 由寄存器 CAPREL 重载 T6 被禁止 1 <sub>B</sub> 由寄存器 CAPREL 重载 T6 被使能

1) 位 T6UD 和 T6UDE 的编码见[表 16-10](#)。

### 定时器 T6 运行控制

可通过软件设定 T6R（定时器 T6 运行控制位），启动或终止核心定时器 T6 工作。该控制位对 T6 的所有工作模式均有效。置位 T6R 将启动定时器工作；清除 T6R 将终止定时器工作。

在门控定时器模式下，只有当 T6R = 1 且门控信号有效（设置为高电平或低电平有效）时，定时器才能工作。

*注：若定时器控制寄存器 T5CON 中的位 T5RC 被置位，位 T6R 也将控制（启动或终止）辅助定时器 T5。*

### 计数方向控制

GPT2 定时器（核心定时器和辅助定时器）的计数方向可由控制寄存器 TxCON 中的位 TxUD 和 TxUDE 控制，选择由软件、或外部输入引脚 TxEUD（定时器 Tx 外部递增/递减控制输入）控制递增/递减计数。由软件控制计数方向时（TxUDE = 0），置位或清零 TxUD 可改变计数方向；位 TxUDE = 1 时，由引脚 TxEUD 控制计数方向。但是，仍可用 TxUD 来翻转实际的计数方向，如表 16-10 所示。无论定时器是否工作，计数方向均可被改变。

**表 16-10 GPT2 定时器计数方向控制**

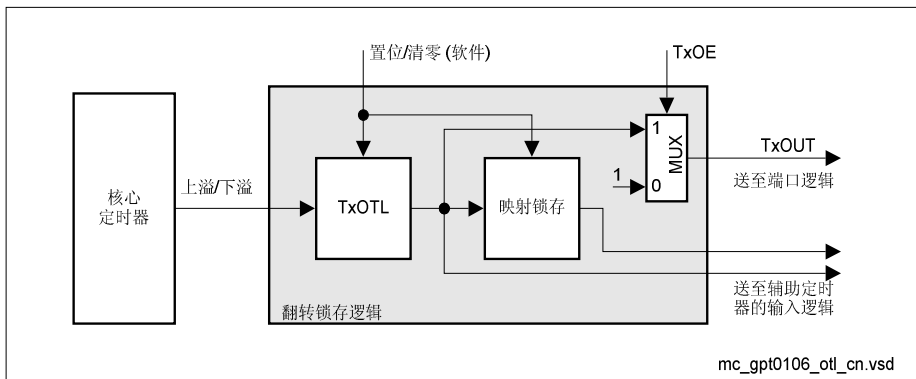
引脚 TxEUD	位 TxUDE	位 TxUD	计数方向
X	0	0	递增计数
X	0	1	递减计数
0	1	0	递增计数
1	1	0	递减计数
0	1	1	递减计数
1	1	1	递增计数

### 定时器 T6 输出翻转锁存

定时器 T6 的上溢/下溢信号送入定时器工作模式方框图中所示的“翻转锁存”模块。图 16-21 示出该模块的内部逻辑。T6 的上溢或下溢信号作为两个锁存器的输入时钟：第一个锁存器代表控制寄存器 T6CON 中的位 T6OTL；第二个锁存器是由 T6OTL 的输出来控制翻转的内部锁存器。两个锁存器的输出均送入辅助定时器 T5 的输入控制模块。映射锁存器的输出电平会与 T6OTL 的输出电平相匹配，只是延迟一个时钟周期。若 T6OTL 的值发生变化，将会导致 T6OTL 和映射锁存器的输出电平暂时不同，从而可触发定时器 T5 的计数事件。

软件写入 T6OTL 时，两个锁存器被同时置位或清零。此时送入辅助定时器的两个信号的电平相同，T5 不会检测到跳变沿。寄存器 T6CON 中的位 T6OE（上溢/下溢输出使能）可使能由外部引脚 T6OUT 监控 T6OTL 的状态。若将 T6OTL 连接至端口引脚上（必须将端口相应配置为输出），可用 T6OUT 控制外部硬件。若 T6OE = 1，引脚 T6OUT 输出 T6OTL 的状态。若 T6OE = 0，（当端口引脚用作定时器输出功能）引脚 T6OUT 输出高电平。

如图 16-21 所示，当软件修改锁存器 T6OTL 的值、以决定输出状态时，内部映射锁存器也相应被置位或清零。此时 T5 不会检测到触发事件。



**图 16-21 核心定时器 T6 翻转锁存逻辑框图**

*注：T6 还可用作 CAPCOM 单元定时器的输入时钟。此时，T6 的上溢/下溢输出线和 CAPCOM 定时器（信号 T6OUF）内部直接连接。*



### 16.2.2 GPT2 核心定时器 T6 的工作模式

定时器 T6 可工作在以下几种模式之一。

### 定时器 T6 工作在定时器模式

将寄存器 T6CON 中的位域 T6M 设置为 000<sub>B</sub>，核心定时器 T6 工作在定时器模式。定时器模式下，模块输入时钟 f<sub>GPT</sub> 经两个可编程的预分频因子（由寄存器 T6CON 中的位域 BPS2 和 T6I 控制）分频后，用作 T6 的输入时钟信号。输入时钟选择的具体内容请参阅[章节 16.2.6](#)。

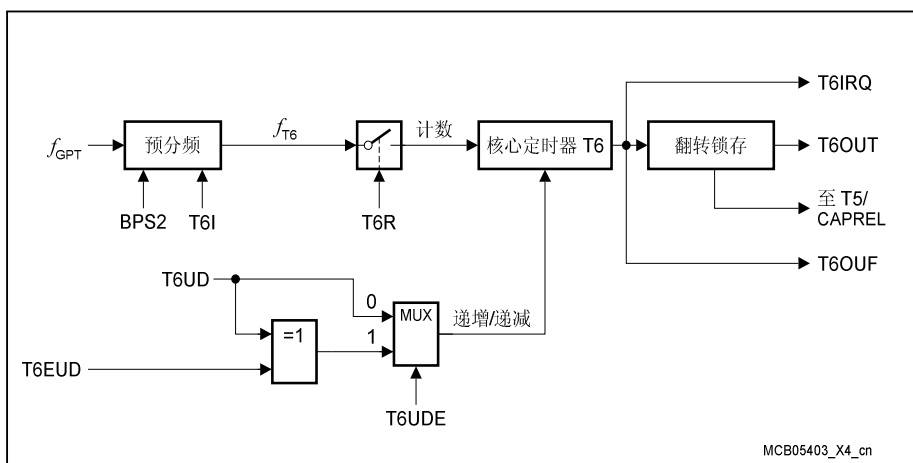


图 16-22 定时器模式下核心定时器 T6 框图

### 门控定时器模式

将寄存器 T6CON 中的位域 T6M 设置为 010<sub>B</sub> 或 011<sub>B</sub>，核心定时器 T6 工作在门控定时器模式。位 T6M.0 (T6CON.3) 选择门控输入的有效电平。门控定时器模式下输入时钟的频率选择和定时器模式下相同（请参阅[章节 16.2.6](#)）。但该模式下定时器的输入时钟受外部输入引脚 T6IN（定时器 T6 的外部输入）门控控制。

T6 工作在该模式时，引脚 T6IN 必须被配置为输入。

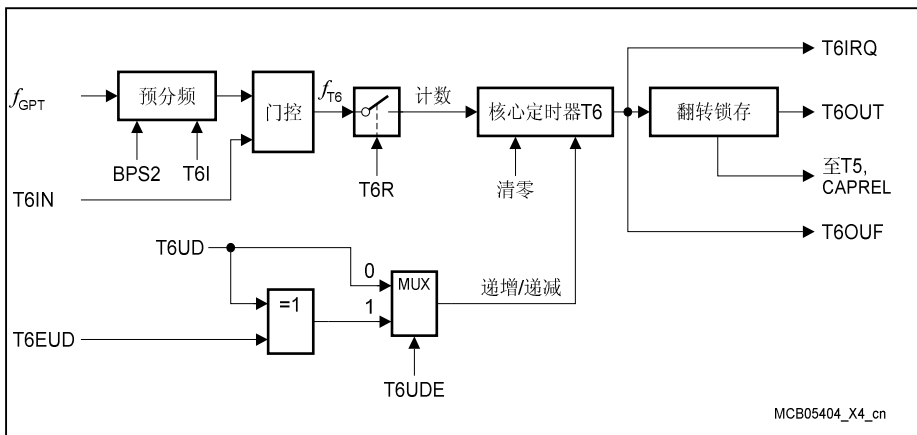


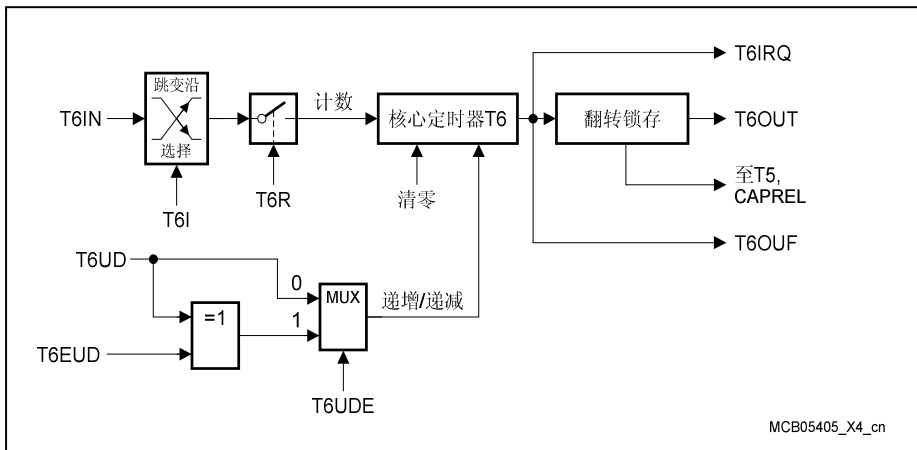
图 16-23 门控定时器模式下核心定时器 T6 框图

若 T6M = 010<sub>B</sub>，引脚 T6IN 为低电平时定时器被使能；为高电平时定时器被终止。若 T6M = 011<sub>B</sub>，T6IN 必须保持高电平才能使能定时器。此外，通过软件修改 T6R 可控制开启或关闭定时器。只有当 T6R 为 1 且门控信号有效时定时器才能工作。若 T6R 为 0 或者门控信号无效，定时器将停止工作。

注：引脚 T6IN 上门控信号的跳变不会引发中断请求。

## 计数器模式

将寄存器 **T6CON** 中的位域 **T6M** 设置为 **001<sub>B</sub>**，核心定时器 **T6** 工作在计数器模式。计数器模式下，外部输入引脚 **T6IN** 上的跳变用作定时器 **T6** 的计数时钟。**T6IN** 上的正跳变、负跳变或任意跳变可触发定时器递增或递减计数。控制寄存器 **T6CON** 中的位域 **T6I** 控制选择触发事件（见**表 16-11**）。



**图 16-24 计数器模式下核心定时器 T6 框图**

**表 16-11 GPT2 核心定时器 T6（计数器模式）输入沿选择**

<b>T6I</b>	<b>计数器递增/递减计数触发沿</b>
000 <sub>B</sub>	无，计数器 T6 被禁止
001 <sub>B</sub>	T6IN 上的正跳变（上升沿）
010 <sub>B</sub>	T6IN 上的负跳变（下降沿）
011 <sub>B</sub>	T6IN 上的任意跳变（上升沿或下降沿）
1XX <sub>B</sub>	保留，不要使用这些组合

计数器模式下，必须将引脚 **T6IN** 配置为输入。计数器模式下允许的最大输入频率取决于所选择的预分频因子。为了确保能够正确识别 **T6IN** 上计数输入信号的跳变，输入电平必须至少保持（高或低）规定数目的模块时钟周期之后才能改变。具体内容请参阅**章节 16.2.6**。

### 16.2.3 GPT2 辅助定时器 T5 的控制

辅助定时器 T5 可被配置为定时器模式、门控定时器模式或计数器模式，定时器输入时钟的频率选择以及计数输入信号的选择和核心定时器 T6 相同。除了这三种计数模式，辅助定时器还可以和核心定时器级联工作，外部或内部事件可触发将 T5 的内容捕获到寄存器 CAPREL 中的操作。辅助定时器的启动/终止功能可由 T6 的运行位控制，从而可同步控制多个定时器的运行。

辅助定时器的当前值可从计数寄存器 T5 中读取。该寄存器也可由 CPU 写入，例如 CPU 设置定时器的初始值。

通过可位寻址控制寄存器 T5CON 配置定时器 T5。该寄存器中的某些位元/域也控制 CAPREL 寄存器的功能。请注意：模块 GPT2 中两个定时器均具备的功能由各自控制寄存器中相同位置的位元/域、以相同的方式进行控制。

*注：辅助定时器无输出翻转锁存，无复用输出功能。*

#### GPT12E\_T5CON

##### 定时器 T5 控制寄存器

SFR (FF46H/A3H)

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T5 SC	T5 CLR	CI	-	CT3	T5 RC	T5 UDE	T5 UD	T5R	T5M			T5I			
rw	rw	rw	-	rw	rw	rw	rw	rw	rw	rw			rw		

符号	位序号	类型	功能描述
T5I	[2:0]	rw	<b>定时器 T5 输入参数选择</b> 不同工作模式的输入参数选择请分别参阅相关内容： <b>表 16-15</b> 对应定时器模式和门控定时器模式 <b>表 16-11</b> 对应计数器模式
T5M	[5:3]	rw	<b>定时器 T5 模式控制（基本工作模式）</b> 000 <sub>B</sub> 定时器模式 001 <sub>B</sub> 计数器模式 010 <sub>B</sub> 门控定时器模式，门控信号低电平有效 011 <sub>B</sub> 门控定时器模式，门控信号高电平有效 100 <sub>B</sub> 保留，不使用该组合 101 <sub>B</sub> 保留，不使用该组合 110 <sub>B</sub> 保留，不使用该组合

符号	位序号	类型	功能描述
			111 <sub>B</sub> 保留，不使用该组合
<b>T5R</b>	6	rw	<b>定时器 T5 运行控制位</b> 0 <sub>B</sub> 定时器 T5 停止运行 1 <sub>B</sub> 定时器 T5 运行 <i>注：只有 T5RC = 0 时该位才可控制定时器 T5。</i>
<b>T5UD</b>	7	rw	<b>定时器 T5 递增/递减计数控制<sup>1)</sup></b> 0 <sub>B</sub> 定时器 T5 递增计数 1 <sub>B</sub> 定时器 T5 递减计数
<b>T5UDE</b>	8	rw	<b>定时器 T5 外部递增/递减计数使能<sup>1)</sup></b> 0 <sub>B</sub> 输入 T5EUD 被断开 1 <sub>B</sub> 计数方向受输入 T5EUD 的控制
<b>T5RC</b>	9	rw	<b>定时器 T5 远程控制位</b> 0 <sub>B</sub> 定时器 T5 由自身的运行位 T5R 控制 1 <sub>B</sub> 定时器 T5 由核心定时器 T6 的运行位 T6R 控制，不由 T5R 控制
<b>CT3</b>	10	rw	<b>定时器 T3 捕获触发使能</b> 0 <sub>B</sub> 输入线 CAPIN 触发捕获 1 <sub>B</sub> T3 的输入线 T3IN 和/或 T3EUD 触发捕获
<b>CI</b>	[13:12]	rw	<b>寄存器 CAPREL 捕获触发事件选择<sup>2)</sup></b> （与位 CT3 有关） 00 <sub>B</sub> 捕获被禁止 01 <sub>B</sub> CAPIN <sup>3)</sup> 上的正跳变（上升沿）或 T3IN 上的任意跳变 10 <sub>B</sub> CAPIN 上的负跳变（下降沿）或 T3EUD 上的任意跳变 11 <sub>B</sub> CAPIN 上的任意跳变（上升沿或下降沿）或 T3IN 或 T3EUD 上的任意跳变
<b>T5CLR</b>	14	rw	<b>定时器 T5 清零使能位</b> 0 <sub>B</sub> 捕获事件不触发定时器 T5 清零 1 <sub>B</sub> 捕获事件触发定时器 T5 清零

符号	位序号	类型	功能描述
<b>T5SC</b>	15	rw	<b>定时器 T5 捕获模式使能</b> 0 <sub>B</sub> 捕获到寄存器 CAPREL 被禁止 1 <sub>B</sub> 捕获到寄存器 CAPREL 被使能

- 1) 位 T5UD 和 T5UDE 的编码见表 16-10。
- 2) 为了定义相应的触发源信号，必须考虑位 CT3（见表 16-13）。
- 3) 如果由内部 GPT1 读信号触发捕获操作（见页 16-58 “组合捕获模式”），则必须选择上升沿。

### 定时器 T5 运行控制

可通过两种不同的方式（软件设定）启动或终止辅助定时器 T5 工作：

- 由相关的定时器运行位（T5R）控制，此时要求 T5RC = 0。
- 由核心定时器的控制位（T6R）控制，此时要求远程控制位必须被置位（T5RC = 1）。

所选择的运行控制位对 T5 的所有工作模式均有效。位置该位将启动定时器工作；清除该位将终止定时器工作。

门控定时器模式下，只有当所选择的运行位被置位且门控信号有效（设置为高或低电平有效）时，定时器才能工作。

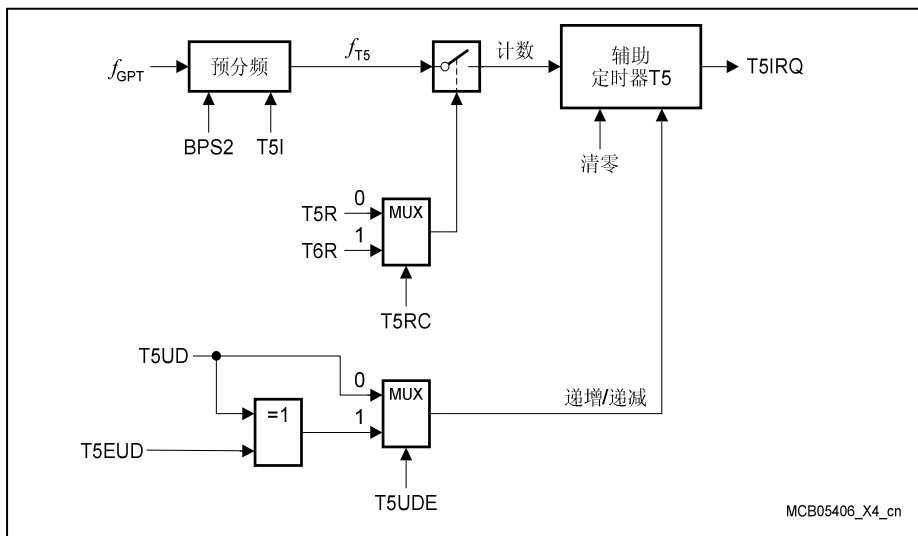
*注：若选择远程控制 T5 运行，T6R 将同步启动或终止定时器 T6 和辅助定时器 T5。*

#### 16.2.4 GPT2 辅助定时器 T5 的工作模式

除极少数例外情况，辅助定时器在基本工作模式下的操作和核心定时器几乎完全相同。此外，T5 可工作在一些组合工作模式下。

##### 定时器 T5 工作在定时器模式

将寄存器 T5CON 中的位域 T5M 设置为 000<sub>B</sub>，辅助定时器 T5 工作在定时器模式。



**图 16-25 定时器模式下辅助定时器 T5 框图**

### 定时器 T5 工作在门控定时器模式

将寄存器 **T5CON** 中的位域 **T5M** 设置为 **010<sub>B</sub>** 或 **011<sub>B</sub>**, 辅助定时器 **T5** 工作在门控定时器模式。位 **T5M.0** (**T5CON.3**) 选择门控输入的有效电平。

注：T5IN 上门控信号的跳变不会引发中断请求。

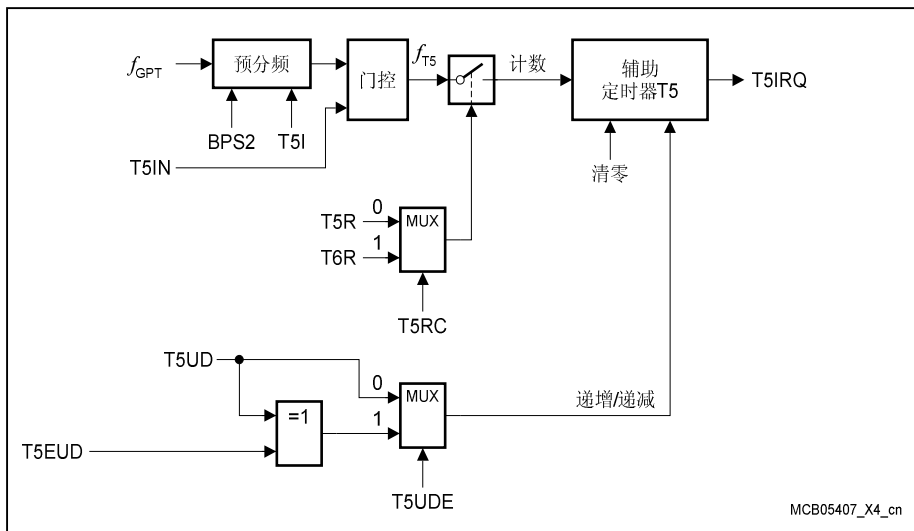


图 16-26 门控定时器模式下辅助定时器 T5 框图

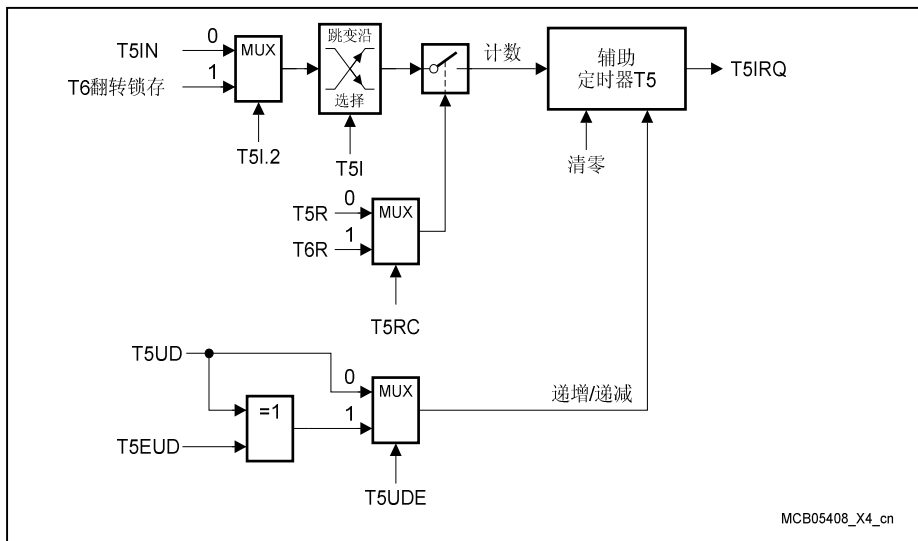
注: T5 无输出翻转锁存。

可通过自身或 T6 的运行位本地或远程控制辅助定时器的启动/终止。



### 定时器 T5 工作在计数器模式

将寄存器 T5CON 中的位域 T5M 设置为 001<sub>B</sub>，辅助定时器 T5 工作在计数器模式。计数器模式下，外部输入引脚 T5IN 跳变、或定时器 T6 的溢出翻转锁存 T6OTL 跳变时，可触发辅助定时器的计数。T5IN 或者 T6OTL 上的正跳变、负跳变或任意跳变可触发定时器递增或递减计数。控制寄存器 T5CON 中的位域 T5I 控制选择触发事件（见表 16-12）。



**图 16-27 计数器模式下辅助定时器 T5 框图**

表 16-12 GPT2 辅助定时器（计数器模式）输入沿选择

T5I	计数器递增/递减计数触发沿
X00 <sub>B</sub>	无。计数器 T5 被禁止
001 <sub>B</sub>	T5IN 上的正跳变（上升沿）
010 <sub>B</sub>	T5IN 上的负跳变（下降沿）
011 <sub>B</sub>	T5IN 上的任意跳变（上升沿或下降沿）
101 <sub>B</sub>	T6 翻转锁存 T6OTL 上的正跳变（上升沿）
110 <sub>B</sub>	T6 翻转锁存 T6OTL 上的负跳变（下降沿）
111 <sub>B</sub>	T6 翻转锁存 T6OTL 上的任意跳变（上升沿或下降沿）

注：只有 T6 上溢/下溢引发的 T6OTL 的状态改变才能触发 T5 计数；软件修改 T6OTL 的状态不会触发 T5 计数。

计数器模式下，必须将引脚 T5IN 配置为输入。计数器模式下允许的最大输入频率取决于所选择的预分频因子。为了确保能够正确识别 T5IN 上计数输入信号的跳变，输入电平必须至少保持（高或低）规定数目的模块时钟周期之后才能改变。具体内容请参阅 [章节 16.2.6](#)。

## 定时器级联

溢出翻转锁存位 T6OTL 用作计数器模式下辅助定时器的计数时钟时，核心定时器 T6 和辅助定时器 T5 级联工作。定时器级联可构成 32 位或 33 位定时器/计数器，这取决于选择何种 T6OTL 跳变触发辅助定时器计数。

- **32 位定时器/计数器：**若选择 T6OTL 的任意跳变触发辅助定时器计数，核心定时器 T6 在每次上溢/下溢时，触发辅助定时器计数。因此 T5 和 T6 构成 32 位定时器。
- **33 位定时器/计数器：**若仅选择 T6OTL 的正跳变或负跳变触发辅助定时器计数，核心定时器 T6 在每两次上溢/下溢后，辅助定时器计数一次。因此它们构成 33 位定时器（16 位核心定时器 + T6OTL + 16 位辅助定时器）。

只要 T6OTL 不被软件修改，它即表示内部翻转锁存器的状态，可被视为 33 位定时器的组成部分。

两个级联定时器的计数方向可以不同，从而为用户提供了多种不同的配置组合。

级联定时器的低 16 位由定时器 T6 表示，此时 T6 可工作在定时器模式、门控定时器模式或计数器模式。

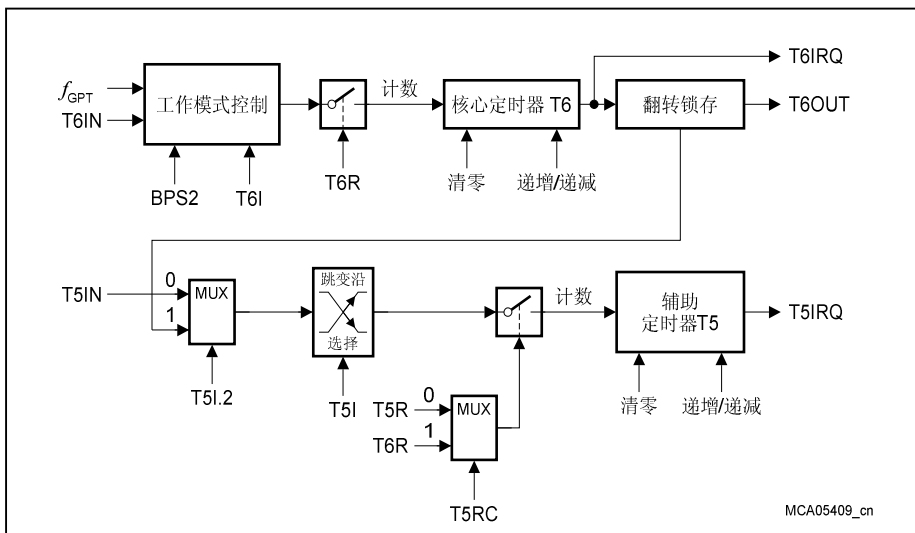


图 16-28 核心定时器 T6 和辅助定时器 T5 级联工作

### 16.2.5 GPT2 寄存器 CAPREL 工作模式

捕获/重载寄存器 CAPREL 可捕获定时器 T5 的内容，或者重载定时器 T6。系统支持一种特殊的模式，即通过寄存器 CAPREL 同时实现捕获、重载两种功能，采用该模式可实现倍频。通过输入引脚 CAPIN，或 GPT1 的定时器 T3 输入线的 T3IN 和 T3EUD 或读访问 GPT1 定时器的方式触发捕获；由定时器 T6 上溢或下溢触发重载功能。

除捕获功能之外，还可用捕获触发信号分别对定时器 T5 和 T6 清零。

寄存器 CAPREL 的功能由定时器控制寄存器 T5CON 和 T6CON 中相关的位/位域控制。

#### GPT2 的捕获/重载寄存器 CAPREL 工作在捕获模式

置位控制寄存器 T5CON 中的位 T5SC，寄存器 CAPREL 工作在捕获模式（将寄存器 T5CON 中的位域 CI 设置为非零值，可选择有效触发沿）。捕获模式下，用户选择的外部输入引脚发生跳变时，辅助定时器 T5 的值被锁存到寄存器 CAPREL 中。位 CT3 选择由外部输入 CAPIN 或 GPT1 的定时器 T3 的输入 T3IN 和/或 T3EUD 作为捕获功能的触发源：CAPIN 上的正跳变、负跳变或任意跳变可触发捕获；T3IN、T3EUD 或二者任意一个输入上的跳变可触发捕获。寄存器 T5CON 中的位域 CI 控制选择有效沿。触发沿选择总结见表 16-13。

表 16-13 CAPREL 寄存器输入沿选择

CT3	CI	计数器递增/递减计数触发沿
X	00 <sub>B</sub>	无，捕获模式被禁止
0	01 <sub>B</sub>	CAPIN 上的正跳变（上升沿） <sup>1)</sup>
0	10 <sub>B</sub>	CAPIN 上的负跳变（下降沿）
0	11 <sub>B</sub>	CAPIN 上的任意跳变（上升沿或下降沿）
1	01 <sub>B</sub>	T3IN 上的任意跳变（上升沿或下降沿）
1	10 <sub>B</sub>	T3EUD 上的任意跳变（上升沿或下降沿）
1	11 <sub>B</sub>	T3IN 或 T3EUD 上的任意跳变（上升沿或下降沿）

1) 如果由 GPT1 读信号触发捕获操作，必须选择上升沿（见寄存器 PISEL 和页 16-58 “组合捕获模式”）。

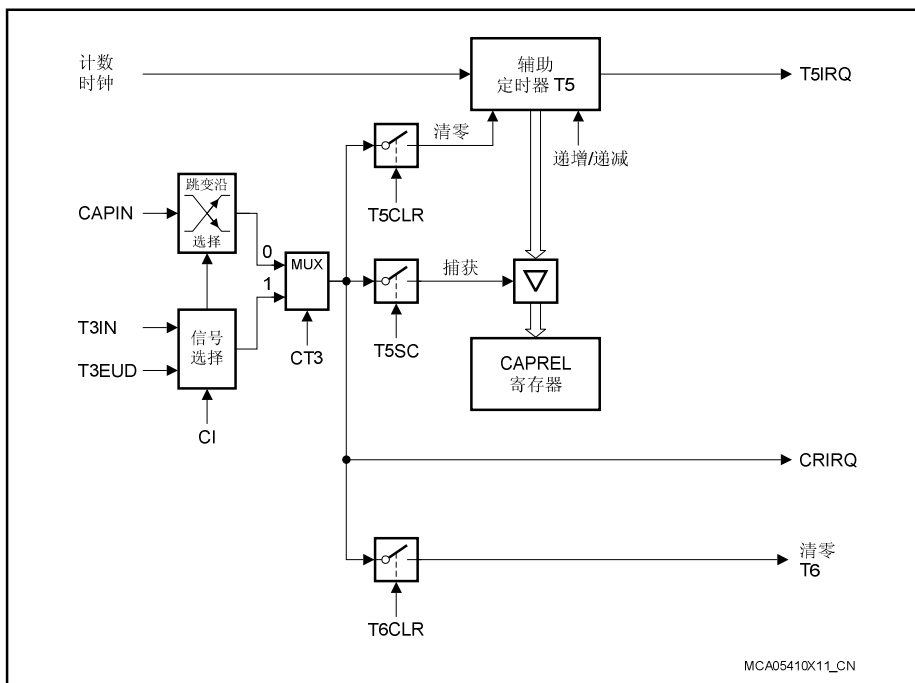


图 16-29 捕获模式下 GPT2 寄存器 CAPREL 框图

检测到触发事件时，辅助定时器 T5 的值被锁存到寄存器 CAPREL 中，中断请求线 CRIRQ 被激活；可通过寄存器 T5CON 中的位 T5CLR 和寄存器 T6CON 中的位 T6CLR 来控制该触发事件是否分别对定时器 T5 和/或 T6 清零。若 TxCLR = 0，捕获事件不影响定时器 Tx 的值；若 TxCLR = 1，定时器 T5 当前值被捕获到寄存器 CAPREL 之后，定时器 Tx 被清零。

注：位 T5SC 仅控制是否执行捕获操作。若 T5SC 被清零，外部输入信号仍可对定时器 T5 和/或 T6 清零，或者用作外部中断输入。该中断由 CAPREL 中断控制寄存器 CRIC 控制。

若由 T3IN 或 T3EUD (CT3 = 1) 触发捕获, 所选输入信号发生跳变时, T5 的内容被捕获到寄存器 CAPREL 中。可利用这些捕获值测量 T3 的输入信号。这些捕获值非常有用, 例如当 T3 工作在增量接口模式时, 要从这些捕获值获取输入信号的动态信息 (速度、加速度)。

捕获模式下，必须将引脚 **CAPIN**、**T3IN** 或 **T3EUD** 配置为输入。为了确保能够正确识别触发输入信号的跳变，输入电平必须至少保持（高或低）规定数目的模块时钟周期之后才能改变。具体内容请参阅[章节 16.2.6](#)。

### GPT2 的捕获/重载寄存器 CAPREL 工作在重载模式

置位控制寄存器 T6CON 中的位 T6SR，寄存器 CAPREL 工作在重载模式。重载模式下，T6 上溢或下溢会触发将寄存器 CAPREL 的值重新装入核心定时器 T6 中的操作。CAPREL 寄存器的相关中断请求线 CRIRQ 不会被激活；但中断请求线 T6IRQ 会被激活，指示定时器 T6 发生上溢/下溢。

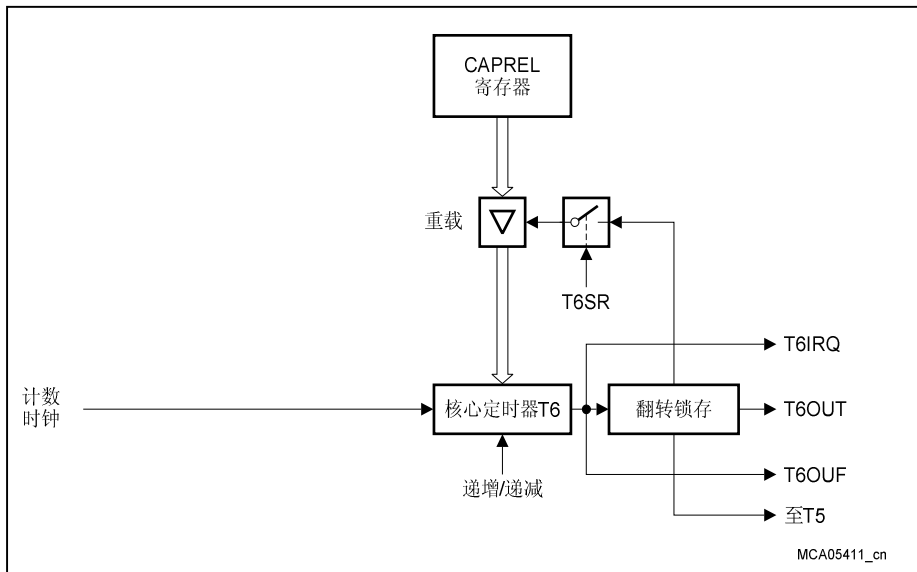


图 16-30 重载模式下 GPT2 寄存器 CAPREL 框图

### GPT2 的捕获/重载寄存器 CAPREL 工作在捕获 – 重载模式

T5SC 和 T6SR 可分别使能寄存器 CAPREL 的重载和捕获功能，因此同时置位 T5SC 和 T6SR 可同时开启这两种功能。可利用该特性产生输入频率的倍频输出。

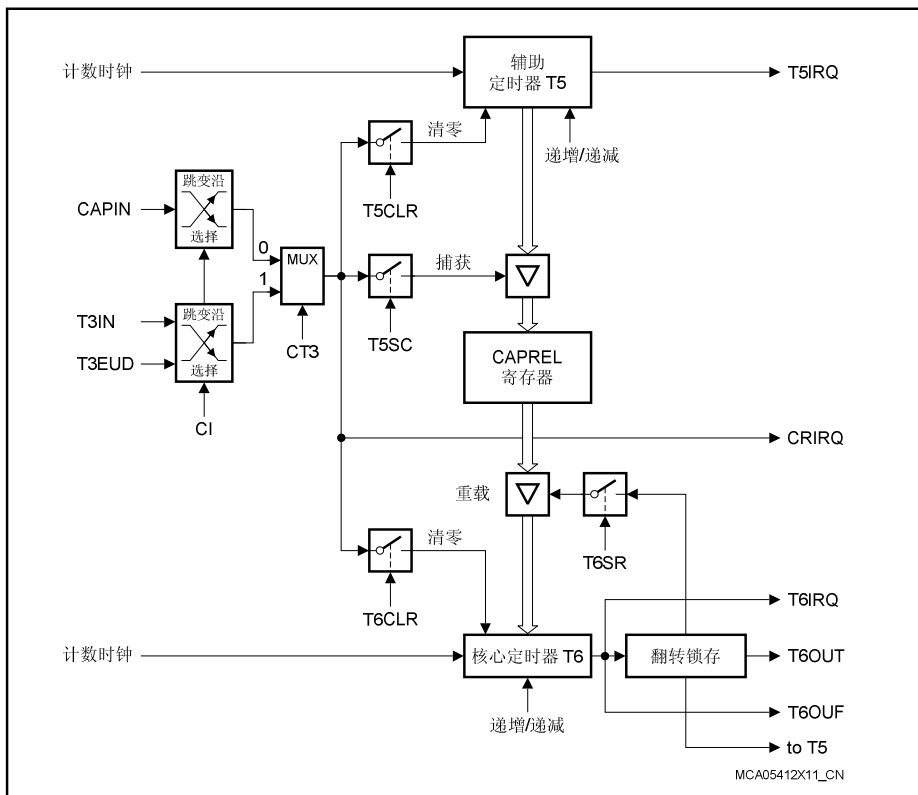


图 16-31 捕获 – 重载模式下 GPT2 寄存器 CAPREL 框图

可用该组合模式检测非周期性的连续外部事件，但需要采用更高的计数精度，这意味着，两次外部事件之间需要更多的“时钟标记”。

为了实现该功能，用定时器 T5 和 CAPREL 寄存器测量外部事件的时间间隔。定时器 T5 工作在定时器模式，以某个频率，如  $f_{GPT}/32$  递增计数。外部事件从引脚 CAPIN 输入。外部事件发生时，定时器 T5 的内容被锁存到寄存器 CAPREL 中，定时器 T5 被清零 ( $T5CLR = 1$ )。因此，寄存器 CAPREL 中始终保存两次事件之间的正确时间，用定时器 T5 的增量表示。定时器 T6 工作在定时器模式，以某个频率，如  $f_{GPT}/4$  递减计

数，发生下溢时将 **CAPREL** 的值重新装入 **T6** 中。因此，寄存器 **CAPREL** 的值代表定时器 **T6** 两次下溢之间的时间，用定时器 **T6** 的增量表示。由于（此例中）定时器 **T6** 的计数频率为 **T5** 的 8 倍，因此 **T6** 在两次外部事件之间会下溢 8 次，从而定时器 **T6** 的下溢信号会产生 8 个“时钟标记”。每次下溢时，中断请求线 **T6IRQ** 被激活，位 **T6OTL** 翻转。**T6OTL** 的状态可从引脚 **T6OUT** 输出。该信号的跳变频率为引脚 **CAPIN** 上信号频率的 8 倍。

*注：定时器 **T6** 的下溢信号可进一步用作 **CAPCOM** 定时器的输入时钟，从而用户能够以比外部事件更高的计数精度设置比较事件。通过 **T6OUF** 信号实现单元之间的连接。*

### 捕获校正

以下原因会造成输出频率的偏差：定时器 **T5** 对实际时间计数（例如，工作在 1MHz 的 **T5** 对 10 kHz 的输入信号会计数到  $64_H/100_D$ ）；但是 **T6OTL** 只在 **T6** 下溢时翻转（如从  $0000_H$  翻转到  $FFFF_H$ ）。依照上面的举例，**T6** 将从  $64_H$  开始递减计数，因此 **T6** 每计数 101 次之后溢出。故实际输出频率为 79.2 kHz，而非期望值 80 kHz。

另一种可能是使用 **T6** 溢出进行校正。在此情况下，**T5** 递减计数，**T6** 递增计数。引脚 **CAPIN** 上的信号跳变触发 **T5** 的计数值被捕获到 **CAPREL**，**T5** 被清零至  $0000_H$ 。在下一个时钟周期，**T5** 下溢至  $FFFF_H$ ，在随后的时钟周期继续递减计数。溢出时用 **CAPREL** 的值重载到 **T6**，在随后的时钟周期内 **T6** 继续递增计数（上例中 **T6** 的计数频率是 **T5** 的 8 倍）。在此情况下，**T5** 和 **T6** 以各自的内部计数频率进行了相同步数的计数。

在上例中，**T5** 的计数频率为 1 MHz，对于 **CAPIN** 上的 10 kHz 的输入信号，**T5** 将递减计数至  $FF9C_H/-100_D$ ，而 **T6** 递增计数，从  $FF9C_H$  经  $FFFF_H$  之后，计数值上溢至  $0000_H$ 。因此 **T6** 在 100 个时序标记之后发生溢出，且 **T6OUT** 上的实际输出频率为期望的 80 kHz。

然而，在此情况下，**CAPREL** 不直接包含两次 **CAPIN** 事件之间的时间，而是该时间的 2 补码。如果操作需要，那么由软件转换该值。

### 组合捕获模式

尤其对于增量接口应用，几个定时器特性可组合起来以获得动态信息，如速度、加速度或减速度。可直接从定时器寄存器（**T2**、**T3**、**T4**）获得当前位置信息。

通过将自由运行定时器 **T5** 的内容捕获到寄存器 **CAPREL** 来产生决定动态参数的时间信息。可为该事件选择两个触发源：

- 传感器信号跳变引起的捕获触发
- 位置读操作引起的捕获触发



可在定时器 T3 输入上捕获传感器信号跳变。通过置位 CT3 和通过寄存器 T5CON 的位域 CI 选择期望信号的方式选择该模式。那么 CAPREL 值将指示两个所选择的跳变之间的时间（以 T5 计数值表示）。

定时器 T2、T3 和 T4 支持对位置读操作的捕获。通过清除 CT3 和寄存器 T5CON 的位域 CI 的选择上升沿的方式选择该模式。那么寄存器 PISEL 中的位域 ISCAPIN 或者读访问 T3，或者读访问 T2 或 T3 或 T4 中任意一个。此时 CAPREL 指示两次读访问之间的时间。

这些操作模式直接支持位置和转动速度的测量。通过评估连续的速度测量值可获得加速度和减速度信息。

### 16.2.6 GPT2 时钟信号控制

定时器模块 GPT2 内的所有操作均由其基本时钟的跳变触发。系统时钟经过基本模块预分频产生该基本时钟，预分频因子由寄存器 T6CON 中的位域 BPS2 控制（见图 16-20）。有两种不同的方式产生计数时钟：

- **内部计数时钟**，经过预分频处理（预分频因子可编程设定）的 GPT2 基本时钟，该时钟用于（门控）定时器模式。
- **外部计数时钟**，来自定时器的输入引脚，该时钟用于计数器模式。

GPT2 模块的基本时钟决定了最大计数频率和定时器精度。

**表 16-14 模块 GPT2 的基本时钟选择**

模块预分频 <sup>1)</sup>	BPS2 = 01 <sub>B</sub>	BPS2 = 00 <sub>B</sub> <sup>2)</sup>	BPS2 = 11 <sub>B</sub>	BPS2 = 10 <sub>B</sub>
GPT2 的预分频因子: F(BPS2)	F(BPS2) = 2	F(BPS2) = 4	F(BPS2) = 8	F(BPS2) = 16
最大外部计数频率	f <sub>GPT</sub> /4	f <sub>GPT</sub> /8	f <sub>GPT</sub> /16	f <sub>GPT</sub> /32
输入信号稳定时间	2× t <sub>GPT</sub>	4× t <sub>GPT</sub>	8× t <sub>GPT</sub>	16× t <sub>GPT</sub>

1) 请注意位域 BPS2 为非线性编码。

2) 复位后的缺省值。

### 内部计数时钟产生

定时器模式和门控定时器模式下，GPT2 的每个定时器的计数时钟来自经过预分频处理的 GPT2 基本时钟，该预分频因子可由定时器控制寄存器 TxCON 中的位域 TxI 编程设定。

定时器 Tx 的计数频率  $f_{Tx}$  和相应计数精度  $r_{Tx}$  随较低的时钟频率线性变化，计算公式如下：

$$f_{Tx} = \frac{f_{GPT}}{F(BPS2) \times 2^{<TxI>}} \quad r_{Tx} [\mu s] = \frac{F(BPS2) \times 2^{<TxI>}}{f_{GPT} [MHz]} \quad (16.2)$$

计数频率不仅取决于公共的模块时钟预分频因子 F(BPS2)；还取决于定时器各自的预分频因子  $2^{<TxI>}$ 。GPT2 定时器的总预分频因子总结见表 16-15。

**表 16-15 GPT2 内部计数时钟的总预分频因子**

定时器 Tx 的 预分频因子	公共预分频因子 <sup>1)</sup>			
	BPS2 = 01 <sub>B</sub>	BPS2 = 00 <sub>B</sub>	BPS2 = 11 <sub>B</sub>	BPS2 = 10 <sub>B</sub>
TxI = 000 <sub>B</sub>	2	4	8	16
TxI = 001 <sub>B</sub>	4	8	16	32
TxI = 010 <sub>B</sub>	8	16	32	64
TxI = 011 <sub>B</sub>	16	32	64	128
TxI = 100 <sub>B</sub>	32	64	128	256
TxI = 101 <sub>B</sub>	64	128	256	512
TxI = 110 <sub>B</sub>	128	256	512	1024
TxI = 111 <sub>B</sub>	256	512	1024	2048

1) 请注意位域 BPS2 为非线性编码。

根据总预分频因子和系统频率计算得到定时器的各项参数（如计数时钟频率、计数时钟精度和计数周期），见表 16-16。请注意有些结果已四舍五入。

**表 16-16     GPT2 定时器参数**

系统时钟 = 10 MHz			总预 分频 因子	系统时钟 = 40 MHz		
频率	精度	周期		频率	精度	周期
5.0 MHz	200 ns	13.11 ms	2	20.0 MHz	50 ns	3.28 ms
2.5 MHz	400 ns	26.21 ms	4	10.0 MHz	100 ns	6.55 ms
1.25 MHz	800 ns	52.43 ms	8	5.0 MHz	200 ns	13.11 ms
625.0 kHz	1.6 µs	104.9 ms	16	2.5 MHz	400 ns	26.21 ms
312.5 kHz	3.2 µs	209.7 ms	32	1.25 MHz	800 ns	52.43 ms
156.25 kHz	6.4 µs	419.4 ms	64	625.0 kHz	1.6 µs	104.9 ms
78.125 kHz	12.8 µs	838.9 ms	128	312.5 kHz	3.2 µs	209.7 ms
39.06 kHz	25.6 µs	1.678 s	256	156.25 kHz	6.4 µs	419.4 ms
19.53 kHz	51.2 µs	3.355 s	512	78.125 kHz	12.8 µs	838.9 ms
9.77 kHz	102.4 µs	6.711 s	1024	39.06 kHz	25.6 µs	1.678 s
4.88 kHz	204.8 µs	13.42 s	2048	19.53 kHz	51.2 µs	3.355 s

### 外部计数时钟输入

用 GPT2 的基本时钟对外部输入信号采样（见**图 16-20**）。为了确保信号能被正确识别，外部信号的当前输入电平值（高或低）必须至少保持一个完整的采样周期之后才能改变。若输入信号连续两次采样的电平值不同，即可识别到信号发生了跳变。因此，至少需要两个基本时钟周期对外部输入信号采样，故输入信号的最大频率一定不能高于基本时钟频率的一半。

GPT2 外部输入信号的参数限制总结见**表 16-17**。

**表 16-17 GPT2 外部输入信号的参数限制**

系统时钟 = 10 MHz		外部输入 频率因子	GPT2 分频 因子 BPS2	输入信号 持续时间	系统时钟 = 40 MHz	
最大输入 频率	电平持续 稳定的 最短时间				最大输入 频率	电平持续 稳定的 最短时间
2.5 MHz	200 ns	$f_{\text{GPT}}/4$	<b>01<sub>B</sub></b>	$2 \times t_{\text{GPT}}$	10.0 MHz	50 ns
1.25 MHz	400 ns	$f_{\text{GPT}}/8$	<b>00<sub>B</sub></b>	$4 \times t_{\text{GPT}}$	5.0 MHz	100 ns
625.0 kHz	800 ns	$f_{\text{GPT}}/16$	<b>11<sub>B</sub></b>	$8 \times t_{\text{GPT}}$	2.5 MHz	200 ns
312.5 kHz	1.6 $\mu\text{s}$	$f_{\text{GPT}}/32$	<b>10<sub>B</sub></b>	$16 \times t_{\text{GPT}}$	1.25 MHz	400 ns

上表中的各项参数限制对 GPT2 的所有外部输入信号均有效，包括计数器模式下的外部计数信号和门控计数器模式下的门控输入信号。

## 16.2.7 GPT2 定时器寄存器

### GPT12E\_T5

定时器 T5 计数寄存器

**SFR (FE46H/23H)**

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>T5</b>															
rwh															

符号	位序号	类型	功能描述
<b>T5</b>	[15:0]	rwh	定时器 <b>T5</b> 当前值 保存定时器 T5 的当前值。

### GPT12E\_T6

定时器 T6 计数寄存器

**SFR (FE48H/24H)**

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>T6</b>															
rwh															

符号	位序号	类型	功能描述
<b>T6</b>	[15:0]	rwh	定时器 <b>T6</b> 当前值 保存定时器 T6 的当前值。

### GPT12E\_CAPREL

捕获/重载寄存器

**SFR (FE4AH/25H)**

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CAPREL</b>															
rwh															

符号	位序号	类型	功能描述
<b>CAPREL</b>	[15:0]	rwh	当前重载值或捕获值 保存 CAPREL 寄存器的当前值。

### 16.2.8 GPT2 定时器和 CAPREL 的中断控制

当定时器从  $FFFF_H$  上溢至  $0000_H$  (递增计数), 或定时器从  $0000_H$  下溢至  $FFFF_H$  (递减计数) 时, 寄存器 **GPT12E\_TxIC** ( $x = 5, 6$ ) 中的中断请求标志被置位。只要检测到引脚 **CAPIN** 上的信号跳变 (由位域 **CI** 选择跳变沿), 寄存器 **GPT12E\_CRIC** 中的中断请求标志被置位。若相应中断使能位被置位, 将会产生指向相应定时器或 **CAPREL** 中断向量的中断, 或者触发 **PEC** 操作。

两个定时器 (**T5**, **T6**) 和 **CAPREL** 寄存器各对应一个中断控制寄存器。所有的中断控制寄存器与中断控制章节中所描述的中断寄存器结构相同。

### 16.3 其它寄存器

下面描述的寄存器并未分配给特定定时器模块。它们控制一般功能和/或给出一般信息:

在软件控制下, 通过寄存器 **GPT12E\_PISEL** 在数个信号源中选择定时器输入信号。

#### **GPT12E\_PISEL**

**端口输入选择寄存器**

**SFR (FE4C<sub>H</sub>/26<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>ISCAPIN</b>	<b>IST6EUD</b>	<b>IST6IN</b>	<b>IST5EUD</b>	<b>IST5IN</b>	<b>IST4EUD</b>	<b>IST4IN</b>	<b>IST3EUD</b>	<b>IST3IN</b>	<b>IST2EUD</b>	<b>IST2IN</b>					
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
<b>IST2IN</b>	0	rw	<b>T2IN 输入选择</b> 0 <sub>B</sub> 选择信号 T2INA 1 <sub>B</sub> 选择信号 T2INB
<b>IST2EUD</b>	1	rw	<b>T2EUD 输入选择</b> 0 <sub>B</sub> 选择信号 T2EUDA 1 <sub>B</sub> 选择信号 T2EUDB

符号	位序号	类型	功能描述
<b>IST3IN</b>	[3:2]	rw	<b>T3IN 输入选择</b> 00 <sub>B</sub> 选择信号 T3INA 01 <sub>B</sub> 选择信号 T3INB 10 <sub>B</sub> 选择信号 T3INC 11 <sub>B</sub> 选择信号 T3IND
<b>IST3EUD</b>	[5:4]	rw	<b>T3EUD 输入选择</b> 00 <sub>B</sub> 选择信号 T3EUDA 01 <sub>B</sub> 选择信号 T3EADB 10 <sub>B</sub> 选择信号 T3EUDC 11 <sub>B</sub> 选择信号 T3EUDD
<b>IST4IN</b>	[7:6]	rw	<b>T4IN 输入选择</b> 00 <sub>B</sub> 选择信号 T4INA 01 <sub>B</sub> 选择信号 T4INB 10 <sub>B</sub> 选择信号 T4INC 11 <sub>B</sub> 选择信号 T4IND
<b>IST4EUD</b>	[9:8]	rw	<b>T4EUD 输入选择</b> 00 <sub>B</sub> 选择信号 T4EUDA 01 <sub>B</sub> 选择信号 T4EADB 10 <sub>B</sub> 选择信号 T4EUDC 11 <sub>B</sub> 选择信号 T4EUDD
<b>IST5IN</b>	10	rw	<b>T5IN 输入选择</b> 0 <sub>B</sub> 选择信号 T5INA 1 <sub>B</sub> 选择信号 T5INB
<b>IST5EUD</b>	11	rw	<b>T5EUD 输入选择</b> 0 <sub>B</sub> 选择信号 T5EUDA 1 <sub>B</sub> 选择信号 T5EADB
<b>IST6IN</b>	12	rw	<b>T6IN 输入选择</b> 0 <sub>B</sub> 选择信号 T6INA 1 <sub>B</sub> 选择信号 T6INB

符号	位序号	类型	功能描述
<b>IST6EUD</b>	13	rw	<b>T6EUD 输入选择</b> 0 <sub>B</sub> 选择信号 T6EUDA 1 <sub>B</sub> 选择信号 T6EUDB
<b>ISCAPIN</b>	[15:14]	rw	<b>CAPIN 输入选择</b> 00 <sub>B</sub> 选择信号 CAPINA 01 <sub>B</sub> 选择信号 CAPINB 10 <sub>B</sub> 选择信号 CAPINC（来自 T3 的读触发） 11 <sub>B</sub> 选择信号 CAPIND（来自 T2 或 T3 或 T4 的读触发）

注：PISEL 的复位值表示先前版本中的可用连接。

GPT12E\_KSCCFG 控制定时器模块的整体操作。

### GPT12E\_KSCCFG

内核状态配置寄存器

SFR (FE1C<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>BP COM</b>	0	<b>COMCFG</b>	<b>BP SUM</b>	0	<b>SUMCFG</b>	<b>BP NOM</b>	0	<b>NOMCFG</b>	0	<b>BP MOD EN</b>	<b>MOD EN</b>				
w	r	rw	w	r	rw	w	r	rw	r	w	rw				

符号	位序号	类型	功能描述
<b>MODEN</b>	0	rw	<b>模块使能</b> 该位使能模块内核时钟和模块功能。 0 <sub>B</sub> 模块关闭。 该模块不响应模块控制操作，模块时钟立即关闭（不需要停止条件）。模块不能响应读访问并且忽略写访问。 1 <sub>B</sub> 模块开启，可以正常工作。 向 MODEN 写 1 之后，访问其它 GPT 寄存器之前，推荐读取寄存器 KSCCFG 以避免流水线对控制模块的影响。 <i>注：应用复位可复位该位。</i>



符号	位序号	类型	功能描述
<b>BPMODEN</b>	1	w	<b>MODEN 的位保护</b> 该位使能对 MODEN 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 <sub>B</sub> MODEN 不改变 1 <sub>B</sub> 由写入值更新 MODEN。 <i>注：应用复位可复位该位。</i>
<b>NOMCFG</b>	[5:4]	rw	<b>正常操作模式配置</b> 该位域定义适用于正常操作模式的内核模式。 0X <sub>B</sub> 模块开启 1X <sub>B</sub> 模块关闭 仅在 CR = 00 或 11 时，该位域才起效。 <i>注：应用复位可复位该位域。</i>
<b>BPNO M</b>	7	w	<b>NOMCFG 的位保护</b> 该位使能对 NOMCFG 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 <sub>B</sub> NOMCFG 不改变 1 <sub>B</sub> 由写入值更新 NOMCFG。 <i>注：应用复位可复位该位。</i>
<b>SUMCFG</b>	[9:8]	rw	<b>挂起模式配置</b> 该位域定义适用于挂起模式的内核模式。 0X <sub>B</sub> 模块开启 1X <sub>B</sub> 模块关闭 仅在 CR = 01 时，该位域才起效。 <i>注：调试复位可复位该位域。</i>
<b>BPSUM</b>	11	w	<b>SUMCFG 的位保护</b> 该位使能对 SUMCFG 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 <sub>B</sub> SUMCFG 不改变。 1 <sub>B</sub> 由写入值更新 SUMCFG。 <i>注：调试复位可复位该位。</i>

符号	位序号	类型	功能描述
<b>COMCFG</b>	[13:12]	rw	<b>时钟关闭模式配置</b> 该位域定义适用于时钟关闭模式的内核模式。 0X <sub>B</sub> 模块开启。 1X <sub>B</sub> 模块关闭。 仅在 CR = 10 时，该位域才起效。 <i>注：应用复位可复位该位域。</i>
<b>BPCOM</b>	15	w	<b>COMCFG 的位保护</b> 该位使能对位域 COMCFG 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 <sub>B</sub> COMCFG 不改变 1 <sub>B</sub> 由写入值更新 COMCFG。 <i>注：应用复位可复位该位。</i>
<b>0</b>	[3:2], 6, 10, 14	r	<b>保留；</b> 读操作返回 0；应写入 0。

GPT12E\_ID 给出模块版本信息。

### GPT12E\_ID

模块 ID 寄存器

MEM (FFE6<sub>H</sub>)

复位值: 58XX<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD_TYPE</b>								<b>MOD_REV</b>							
r								r							

符号	位序号	类型	功能描述
<b>MOD_REV</b>	[7:0]	r	<b>模块版本编号</b> MOD_REV 定义版本编号。模块版本编号从 01 <sub>H</sub> 开始（第一版）。
<b>MOD_TYPE</b>	[15:8]	r	<b>模块 ID 号</b> 该位域定义模块 ID 编号（58 <sub>H</sub> = GPT12E）。

## 16.4 寄存器表

**表 16-18** 给出编程 GPT12E 模块所需的所有寄存器。该表总结出 GPT12E 内核寄存器以及模块外部寄存器，并给出它们的地址和复位值。

**表 16-18 GPT12E 寄存器总结**

寄存器名	描述	地址		复位值
		16 位	8 位	

### 通用定时器单元（GPT12E）

<b>GPT12E_ID</b>	GPT12E 模块 ID 寄存器	FFE6 <sub>H</sub>	F3 <sub>H</sub>	58XX <sub>H</sub>
<b>GPT12E_PISEL</b>	输入信号选择	FE4C <sub>H</sub>	26 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T2CON</b>	GPT12E 定时器 T2 控制寄存器	FF40 <sub>H</sub>	A0 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T3CON</b>	GPT12E 定时器 T3 控制寄存器	FF42 <sub>H</sub>	A1 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T4CON</b>	GPT12E 定时器 T4 控制寄存器	FF44 <sub>H</sub>	A2 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T5CON</b>	GPT12E 定时器 T5 控制寄存器	FF46 <sub>H</sub>	A3 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T6CON</b>	GPT12E 定时器 T6 控制寄存器	FF48 <sub>H</sub>	A4 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_CAPREL</b>	GPT12E 捕获/重载寄存器	FF4A <sub>H</sub>	25 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T2</b>	GPT12E 定时器 T2 寄存器	FE40 <sub>H</sub>	20 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T3</b>	GPT12E 定时器 T3 寄存器	FE42 <sub>H</sub>	21 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T4</b>	GPT12E 定时器 T4 寄存器	FE44 <sub>H</sub>	22 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T5</b>	GPT12E 定时器 T5 寄存器	FE46 <sub>H</sub>	23 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T6</b>	GPT12E 定时器 T6 寄存器	FE48 <sub>H</sub>	24 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T2IC</b>	GPT12E 定时器 T2 中断控制寄存器	FF60 <sub>H</sub>	B0 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T3IC</b>	GPT12E 定时器 T3 中断控制寄存器	FF62 <sub>H</sub>	B1 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T4IC</b>	GPT12E 定时器 T4 中断控制寄存器	FF64 <sub>H</sub>	B2 <sub>H</sub>	0000 <sub>H</sub>

**通用定时器单元**

寄存器名	描述	地址		复位值
		16 位	8 位	
<b>GPT12E_T5IC</b>	GPT12E 定时器 T5 中断控制寄存器	FF66 <sub>H</sub>	B3 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_T6IC</b>	GPT12E 定时器 T6 中断控制寄存器	FF68 <sub>H</sub>	B4 <sub>H</sub>	0000 <sub>H</sub>
<b>GPT12E_CRIC</b>	GPT12E CAPREL 中断控制寄存器	FF6A <sub>H</sub>	B5 <sub>H</sub>	0000 <sub>H</sub>

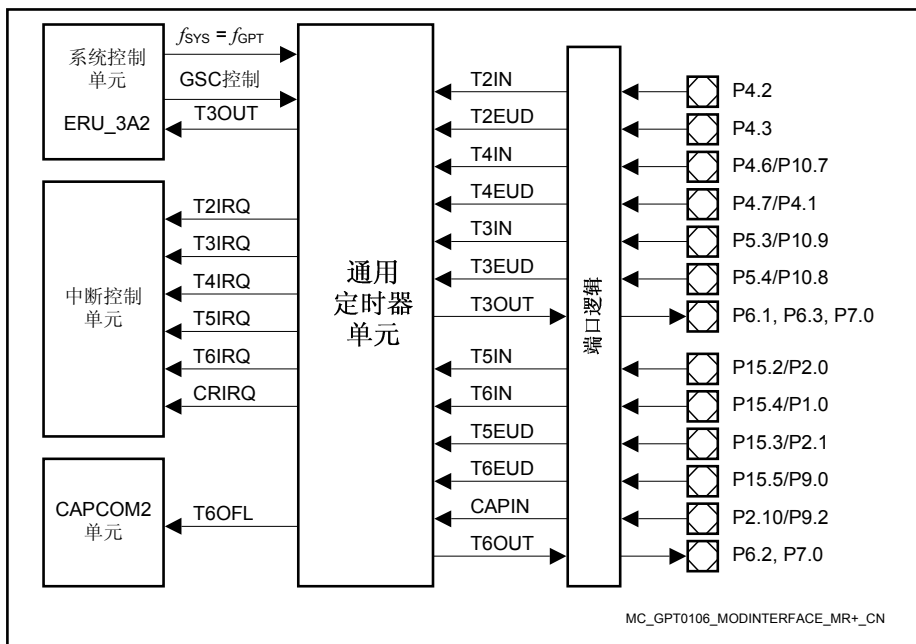
## 16.5 GPT 模块接口

除了所描述的模块内部连接，定时器单元模块 GPT1 和 GPT2 以两种基本方式和周围资源接口：

- **内部连接：** 定时器和片上资源（如时钟产生单元、中断控制器，或其它定时器）接口。

GPT 模块的输入时钟是 XE16xyM 系统时钟，所以  $f_{GPT} = f_{SYS}$ 。

- **外部连接：** 定时器通过端口引脚和外部资源接口。



**图 16-32 GPT 模块接口**

注：GPT12E 输出信号 ‘T6OFL’ 与 CAPCOM2 输入 ‘TOUF’ 和 GSC 相连。

**表 16-19 XE16xyM 内的 GPT 数字信号连接**

信号	来自/送至 哪个模块	I/O 至 GPT	可用作...
T2INA	P4.2	I	定时器 T2 的计数输入信号
T2INB	0	I	
T2EUDA	P4.3	I	定时器 T2 的方向输入信号
T2EUSB	0	I	
T2IRQ	ICU	O	来自定时器 T2 的中断请求
T3INA	P5.3	I	定时器 T3 的计数输入信号
T3INB	P10.9	I	
T3INC	0	I	
T3IND	0	I	
T3EUDA	P5.4	I	定时器 T3 的方向输入信号
T3EUSB	P10.8	I	
T3EUDC	0	I	
T3EUDD	0	I	
T3OUT	P7.0	O	定时器 T3 的计数输出信号
	P6.1	O	
	P6.3	O	
	ERU_3A2 (SCU)	O	
T3IRQ	ICU	O	来自定时器 T3 的中断请求
T4INA	P4.6	I	定时器 T4 的计数输入信号
T4INB	P10.7	I	
T4INC	0	I	
T4IND	0	I	
T4EUDA	P4.7	I	定时器 T4 的方向输入信号

信号	来自/送至 哪个模块	I/O 至 GPT	可用作...
T4EUDB	P4.1	I	
T4EUDC	0	I	
T4EUDD	0	I	
T4IRQ	ICU	O	来自定时器 T4 的中断请求
T5INA	P15.2	I	定时器 T5 的计数输入信号
T5INB	P2.0	I	
T5EUDA	P15.3	I	定时器 T5 的方向输入信号
T5EUDB	P2.1	I	
T5IRQ	ICU	O	来自定时器 T5 的中断请求
T6INA	P15.4	I	定时器 T6 的计数输入信号
T6INB	P1.0	I	
T6EUDA	P15.5	I	定时器 T6 的方向输入信号
T6EUDB	P9.0	I	
T6OUT	P7.0	O	定时器 T6 的计数输出信号
	P6.2	O	
T6IRQ	ICU	O	来自定时器 T6 的中断请求
T6OFL	CC2_TOUF, SCU (GSC)	O	来自定时器 T6 的上溢/下溢信号
CAPINA	P2.10	I	输入捕获信号
CAPINB	P9.2	I	
CAPINC	来自 T3 的读 触发	I	
CAPIND	来自 T2 或 T3 或 T4 的 读触发	I	
CRIRQ	ICU	O	来自捕获控制的中断请求

### 端口控制

必须将用作定时器输入信号的端口引脚配置为输入（相应端口控制寄存器内的位域 **PC** 必须为 **0xxx<sub>B</sub>**），必须通过寄存器 **PISEL** 进行输入选择。

必须将用作定时器输出信号的端口引脚配置为输出，且必须选择复用定时器输出信号（相应端口控制寄存器内的位域 **PC** 必须为 **1xxx<sub>B</sub>**）。

当输入引脚分配到 **P5** 口和 **P15** 口（单向输入端口），必须通过数字输入控制寄存器 **PxDIDIS** 使能数字输入。

*注：复位之后，P5DIDIS 和 P15DIDIS 寄存器的缺省值将直接使能 P5/P15 输入。*

*注：端口控制寄存器的描述，请参考“并行端口”一章。*

### 中断

**GPT12** 具有 6 条中断请求线。

必须使能用于定时器中断请求的中断节点并设定相应的中断优先级。

### 调试详情

**GPT** 模块禁止时，该模块的寄存器仍可被读取；而且被禁止时，下面的两个寄存器可被写入：**PISEL**、**T5CON**。

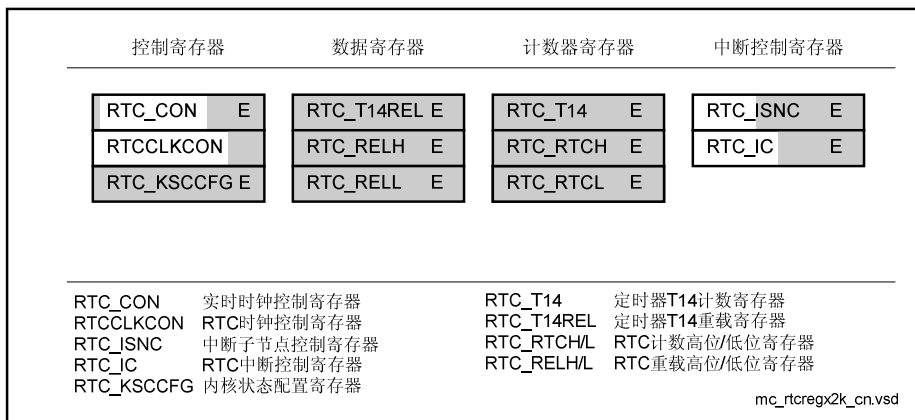


## 17 实时时钟

XE16xyM 的实时时钟（RTC）模块由一组预分频器和定时器级联构成。由辅助振荡器或主振荡器经预分频处理为 RTC 提供计数时钟。RTC 的用途包括：

- 48 位定时器，测量长时间间隔
- 系统时钟，指示当前时间和日期  
 （RTC 的结构支持由系统时钟直接指示时间和日期）
- 周期性中断（可由级联链中的任意一个定时器产生）

不同的编程设定和中断请求信号可使 RTC 满足不同的应用需求。即使 XE16xyM 工作在某个省电模式 RTC 仍可继续工作，从而提供了系统的实时日期和时间信息。



**图 17-1 RTC 模块的相关特殊功能寄存器（SFR）**

RTC 模块由 3 个分频模块组成：

- 8:1 分频器（可编程选择开启或关闭）
- 可重载的 16 位定时器 T14
- 32 位 RTC 定时器模块（通过 RTC\_RTCH 和 RTC\_RTCL 进行访问），它由以下定时器级联构成：
  - 可重载的 10 位定时器 CNT0
  - 可重载的 6 位定时器 CNT1
  - 可重载的 6 位定时器 CNT2
  - 可重载的 10 位定时器 CNT3

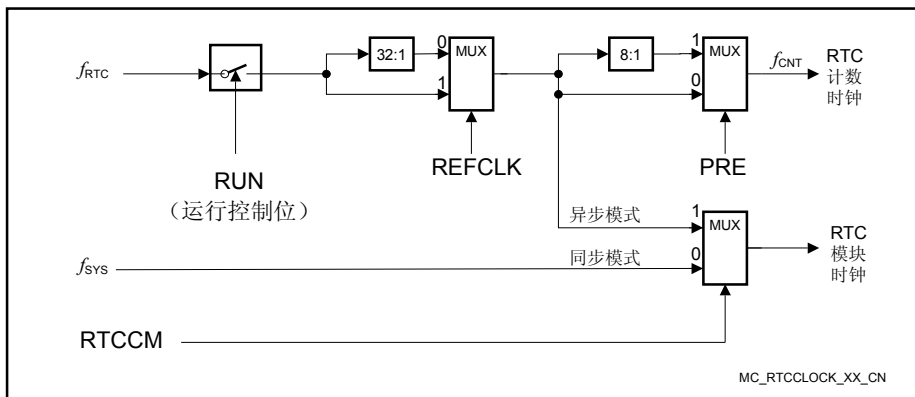
所有定时器递增计数。五个定时器均可产生各自的中断请求，所有的中断请求合并成一个公共中断节点请求。

*注：RTC 寄存器仅受上电复位的影响，目的是为了确保始终能够指示正确的系统时间（即使在执行系统复位或应用复位时）。*

## 17.1 确定 RTC 的时间基准

RTC 的定时器链由时钟信号  $f_{\text{RTC}}$  触发计数，该时钟信号来自内部时钟源（振荡器或 PLL）或外部源（引脚）。由寄存器 **RTCCLKCON** 中的位域 **RTCCLKSEL** 选择当前有效的时钟源。 $f_{\text{RTC}}$  可直接或经过预分频（因子为 32 和/或 8）得到基本 RTC 时钟。根据工作模式，定时器 **T14** 可提供不同应用情况下的计数增量，从而确定了 RTC 定时器的输入频率，即 RTC 的时间基准（见表 17-2）。

XE16xyM 的系统时钟  $f_{\text{SYS}}$  也为 RTC 模块提供时钟信号。该时钟信号控制 RTC 的逻辑模块和总线接口。为了能够和计数时钟正确同步，系统时钟必须至少是计数时钟的四倍，即  $f_{\text{SYS}} \geq 4 \times f_{\text{CNT}}$ 。



**图 17-2 RTC 时钟源**

**表 17-1** 例举 T14 中断周期的取值范围和 T14 的重载值（时间基准为 1 s 和 1 ms）：

**表 17-1      RTC 时间基准举例**

振荡器 频率	T14 中断周期		重载值 A		重载值 B	
	最小值	最大值	T14REL	时间基准	T14REL	时间基准
32.768 kHz	30.52 μs	16.0 s	8000 <sub>H</sub> / F000 <sub>H</sub>	1.000 s	FFDF <sub>H</sub> / FFFC <sub>H</sub>	1.007 ms/ 0.977 ms

注：根据是否使能 8:1 预分频器，从这些重载值中选择相应的值。

### 异步工作

当系统时钟频率低于  $4 \times f_{CNT}$  时，将无法与计数时钟正确同步、计数事件可能会被遗漏，例如当 XE16xyM 为了省电而降低系统频率时，就可能出现上述情况。

发生上述情况时，RTC 可切换到异步模式（通过清除寄存器 RTCCLKCON 中的位 RTCCM 实现）。异步模式下，计数寄存器直接由计数时钟控制，和系统时钟无关（异步模式因而得名）。RTC 异步工作确保了即使在省电模式下，仍能正确计时。

但是，在异步模式下，由于无法维持计数寄存器和总线接口之间的同步，RTC 寄存器不能被写入。读操作可能会干扰计数事件，因此必须对读操作进行验证（比如，连续三次读取同一个值）。

*注：当然，只有在系统时钟未关闭时，考虑异步模式下对寄存器读写操作的限制才有意义。*

### 时钟模式切换

由寄存器 RTCCLKCON 中的位 RTCCM 控制选择 RTC 的时钟模式（同步或异步）。复位后 RTC 工作在同步模式（RTCCM = 1）。

时钟模式的选择还影响对 RTC 寄存器的读写操作。寄存器 RTC\_CON 中的位 ACCPOS 指示是否可对寄存器进行完全读写访问（复位后的缺省值 ACCPOS = 1 表示可对寄存器读写；ACCPOS = 0 反之）；ACCPOS 也指明了当前的时钟模式。

**请注意：软件应查询位 ACCPOS 的值，以确定是否切换到所期望的时钟模式。**

切换至异步模式（RTCCM = 0）之后，位 ACCPOS = 0 指示异步模式下的正确操作。此时系统时钟可被关闭或其频率被降低。

*注：必须在切换到异步模式工作之前为其选定时钟源，而且只有在同步模式下才能改变时钟源。*

切换至同步模式（RTCCM = 1）之后，位 ACCPOS = 1 指示同步模式下的正确操作。在该模式下可重新访问（读和写）RTC 寄存器。

*注：若 8:1 预分频被禁止，RTC 从同步模式切换到异步模式时可能会丢失一个计数事件（ $f_{CNT}$  的跳变沿）。因此推荐使能 8:1 预分频器来配置 RTC。*

### 通过软件校正提高 RTC 的精度

振荡器频率和预分频因子（包括或不包括 T14 和 8:1 预分频器）决定了 XE16xyM RTC 的精度。预分频产生的精度限制由二进制计数器的量化引起（平均值为 0）；振荡器频率产生的精度限制由理想频率和实际频率之间的差值引起（因此差值会随时间累加），该影响可被预估并补偿。针对时间精度要求较高的应用，可通过软件进一步提高 RTC 总精度。

提高精度的关键是要知道实际的振荡频率。可通过实际频率和期望频率之间的误差测量 RTC 的偏差。可很容易的计算得出：经过多少（N）周期该误差将累积到 $\pm 1$ 个周期。因此，只需在每 N 个周期之后即可通过 $\pm 1$ 校正计数脉冲。比如可通过中断服务程序实现周期性的误差校正；或在需要时对所读取的 RTC 寄存器数值进行评估以实现差值校正（使用这种方式需储存 RTC 的前一个值）。

*注：不过对于大多数应用，RTC 所提供的标准精度已远远满足应用需要。*

调整 RTC 的当前值需先读取再写入完整的 48 位数值，只有通过连续三次访问方可完成。为了避免多字读取/写入发生冲突，RTC 可增加或减少一个计数脉冲进行纠错。

这可通过置位寄存器 RTC\_CON 中的 T14INC 或 T14DEC 实现：下次计数事件到来时增加一个额外的计数脉冲（T14INC），或减少一个计数脉冲（T14DEC）。操作结束前这些位始终保持置位状态，操作结束后硬件自动对相关位清零。

*注：同时置位 T14INC 和 T14DEC 将不会影响计数值。*

## 17.2 RTC 运行控制

若启动 RTC，必须置位寄存器 RTC\_CON 中的位 RUN（复位后的缺省值）。可以将位 RUN 清除，比如在计时过程中，某些操作阶段不需要 RTC 工作。

*注：当然，需要有效的计数时钟  $f_{RTC}$ ，RTC 才能正常操作。*

RTC 只能由上电复位操作复位，系统和应用复位不影响 RTC 寄存器及其操作（然而 RTC\_IC 会被复位）。初始化软件必须确保 RTC 工作在恰当的工作模式下。

### 初始化和禁止 RTC

发生上电复位时，寄存器 RTC\_CON 复位至 8003<sub>H</sub>（复位值），该值使能 RTC 以及两个 RTC 预分频器（因此分频因子 =  $8 \times 32 = 256$ ）。

由寄存器 RTCCLKCON 中的位 RTCCM 选择 RTC 的时钟模式（同步/异步）。上电复位时，寄存器 RTCCLKCON 复位至 0006<sub>H</sub>，该复位值选择同步工作模式，且 WUT 作为时钟源。

对于应用复位，RTC 模块的初始化紧随其后，推荐使用下述步骤：

- 选择同步 RTC 时钟模式，即置位 RTCCLKCON.RTCCM
- 选择想要（运行）的时钟源
- 写访问 RTC 寄存器之前，确保位 ACCPOS 置位
- 初始化 RTC

上电复位之后，当不使用 RTC 模块且 RTC 模块应当被禁止的情况，推荐使用下述步骤：

- 通过清除运行控制位 RTC\_CON.RUN 终止 RTC
- 通过清除其使能位 RTC\_KSCCFG.MODEN 来禁止 RTC 模块

当 RTC 模块工作在异步模式，且在省电模式下需要停止 RTC 操作时，软件必须确保 RTC 时钟复用器所选择的信号没有有效的时钟信号。

RTC 控制寄存器 RTC\_CON 选择 RTC 模块的基本操作。

## RTC\_CON

控制寄存器

ESFR (F110H/88H)

复位值: 8003H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC POS	-	-	-	-	-	-	-	-	-	-	REF CLK	T14 INC	T14 DEC	PRE	RUN
rh	-	-	-	-	-	-	-	-	-	-	rw	rwh	rwh	rw	rw

符号	位序号	类型	功能描述
<b>ACCPOS</b>	15	rh	<b>RTC 寄存器访问控制</b> 0 不能执行写操作，只能异步读取 1 可执行读/写操作
<b>REFCLK</b>	4	rw	<b>RTC 输入时钟预分频器 (32: 1) 禁止</b> 0 使能输入预分频器 1 禁止输入预分频器
<b>T14INC</b>	3	rwh	<b>定时器 T14 加 1</b> 该位置位时，下次计数事件到来时增加一个计数脉冲，即 T14 加 1。 加 1 操作完成后硬件自动对该位清零。
<b>T14DEC</b>	2	rwh	<b>定时器 T14 减 1</b> 该位置位时，下次计数事件到来时减少一个计数脉冲，即 T14 减 1。 减 1 操作完成后硬件自动对该位清零。
<b>PRE</b>	1	rw	<b>RTC 输入时钟预分频 (8: 1) 使能</b> 0 预分频被禁止 1 预分频被使能
<b>RUN</b>	0	rw	<b>RTC 运行位</b> 0 RTC 停止运行 1 RTC 运行



### 17.3 RTC 工作模式

根据实际的应用需求，RTC 可被配置为不同的工作模式。这些工作模式的设置可通过选择恰当的重载值和中断信号来实现。

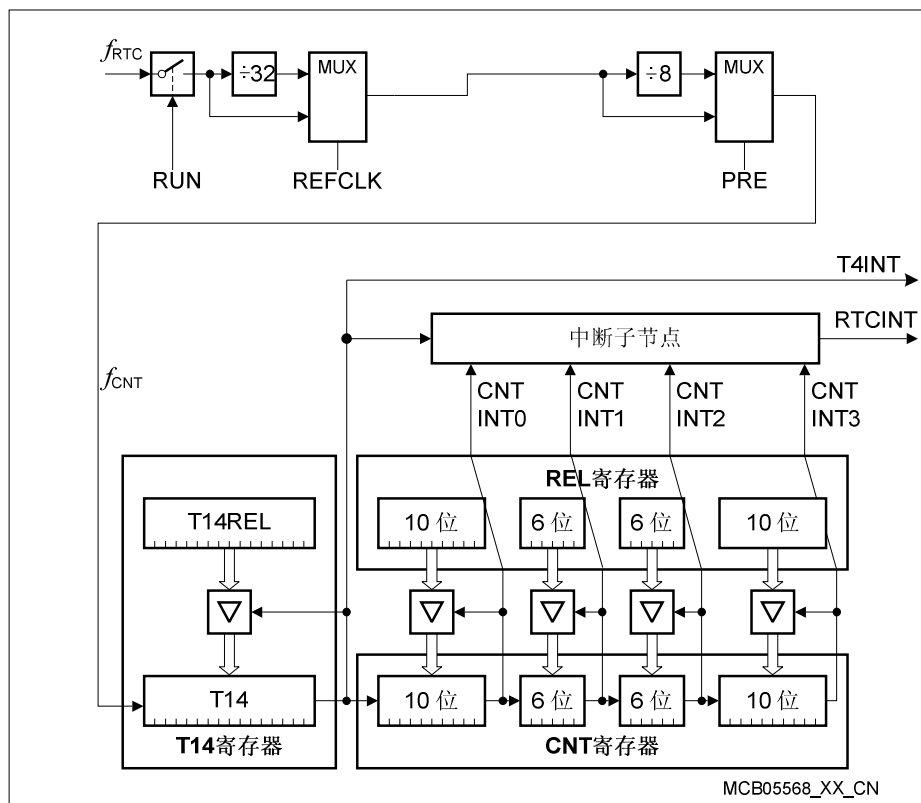


图 17-3 RTC 框图

注：信号 T14INT 可触发串行通信通道的传送操作。

#### RTC 寄存器读写访问

RTC 的值由三个寄存器 T14、RTCL 和 RTCH 表示。由于这些寄存器级联构成 RTC 的计数器链，RTC 工作时内部会产生溢出。读取或写入 RTC 值时，必须考虑可能的内部溢出以避免读取/写入错误值。

读取定时器值时必须谨慎：因为这需要对不同的寄存器执行三次读操作，读操作之间存在固有的时间延迟。必须考虑到在两次读取访问之间可能发生的溢出(T14 溢出送至 RTCL 和/或 RTCL 溢出送至 RTCH)而造成读写错误。

例如，从 RTCH 中读取或向 RTCH 中写入 0000<sub>H</sub>后，在执行 RTCL 读写操作时会产生错误，因为此时 RTCL 可能产生溢出。这种情况下，RTCH 将变为 0001<sub>H</sub>。在对 T14 和 T14REL 执行读写操作时也应采取同样的处理。

通过专用地址方式访问定时器 T14 和 T14 重载寄存器：通过两个 16 位 RTC 定时器寄存器 RTCH 和 RTCL 访问 RTC 的四个计数器 CNT3...CNT0；通过两个 16 位 RTC 重载寄存器 RELH 和 RELL 访问四个相关的重载值 REL3...REL0。

## 预分频器 T14 和 T14 重载寄存器

### RTC\_T14

RTC 计数寄存器 ESFR (F0D2<sub>H</sub>/69<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T14CNT															
rwh															

符号	位序号	类型	功能描述
T14CNT	[15:0]	rwh	T14 计数器

### RTC\_T14REL

RTC 重载寄存器 ESFR (F0D0<sub>H</sub>/68<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T14REL															
rw															

符号	位序号	类型	功能描述
T14REL	[15:0]	rw	T14 重载值

## RTC\_RTCH

**RTC 定时器高位寄存器**

**ESFR (F0D6<sub>H</sub>/6B<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CNT3</b>										<b>CNT2</b>					
rwh										rwh					

符号	位序号	类型	功能描述
<b>CNT<sub>x</sub></b> ( <b>x = 3 ... 2</b> )	[15:6], [5:0]	rwh	<b>RTC 定时器计数值 CNT<sub>x</sub></b> 位域 CNT2 溢出会触发 CNT3 的计数脉冲，紧接着将位域 REL2 的值重新装入 CNT2。 此外，溢出会触发中断请求。

## RTC\_RTCL

**RTC 定时器低位寄存器**

**ESFR (F0D4<sub>H</sub>/6A<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CNT1</b>										<b>CNT0</b>					
rwh										rwh					

符号	位序号	类型	功能描述
<b>CNT<sub>x</sub></b> ( <b>x = 1 ... 0</b> )	[15:10] [9:0]	rwh	<b>RTC 定时器计数值 CNT<sub>x</sub></b> CNT <sub>x</sub> 位域溢出时会触发计数脉冲，控制下一个计数器 CNT <sub>x</sub> +1 计数，紧接着将位域 REL <sub>x</sub> 的值重新装入 CNT <sub>x</sub> 。 此外，溢出会触发中断请求。

### **RTC\_RELH**

**RTC 重载高位寄存器**                      **ESFR (F0CE<sub>H</sub>/67<sub>H</sub>)**                      **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>REL3</b>										<b>REL2</b>					
rw										rw					

符号	位序号	类型	功能描述
<b>RELx</b> ( x = 3 ... 2 )	[15:6], [5:0]	rw	<b>RTC 重载值 RELx</b> CNTx 计数溢出时, 将该位域的值复制到位域 CNTx 中。

### **RTC\_RELL**

**RTC 重载低位寄存器**                      **ESFR (F0CC<sub>H</sub>/66<sub>H</sub>)**                      **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>REL1</b>										<b>RELO</b>					
rw										rw					

符号	位序号	类型	功能描述
<b>RELx</b> ( x = 1 ... 0 )	[15:10] [9:0]	rw	<b>RTC 重载值 RELx</b> CNTx 计数溢出时, 将该位域的值复制到位域 CNTx 中。

*注: 只有在上电复位的情况下, 所有的 RTC 寄存器才会复位。*

## 17.4 48 位定时器操作

定时器 T14 和 COUNT0...COUNT3 级联可构成 48 位定时器，由 RTC 的输入时钟控制计数（时钟的预分频可以选择）。重载寄存器 T14REL, REL1 和 RELH 必须被清零以实现真正的二进制 48 位定时器。不过也允许采用其它重载值。RTC 的各定时器使用非零重载值时，各自的溢出情况不同，因此使用非零重载值必须谨慎。

最大可用的时间跨度为  $2^{48}$  ( $\approx 10^{14}$ ) 个 T14 输入时钟。假设不进行预分频，计数频率为 32 kHz 时，该时间值将超过 200 年。

## 17.5 系统时钟操作

即使在省电模式（可选择的）下，实时系统时钟也能继续运行，维持稳定的时钟信号和供电并指示当前时间和日期。原因是只有上电复位才能复位和清除 RTC 模块。

系统时钟的精度由定时器 T14 的输入时钟决定。选择恰当的重载值，各级联定时器可分别表示当前时间和/或日期的一部分。T14 可将 RTC 调整到期望的工作范围（时间或日期）。T14 的重载值为 0000<sub>H</sub> 时对应的可用时间跨度最大，即 T14 将输入时钟  $2^{16}$  分频。

### 系统时钟举例

RTC 的计数时钟为 32.768 kHz。选择恰当的重载值，RTC 的定时器可直接表示当前时间（见图 17-4 和表 17-2）。

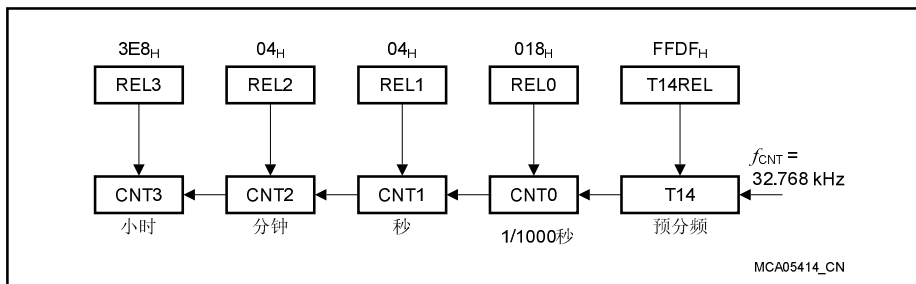


图 17-4 RTC 配置举例

注：该配置可每毫秒、每秒、每分钟、每小时或每天产生一个中断请求。

级联链中的每个定时器递增计数时，将输入时钟 ( $2^{\text{<定时器宽度>}} - \text{<重载值>}$ ) : 1 分频。表 17-2 给出表示一天/一周时间时，各定时器所对应的重载值。

表 17-2 重载值举例

		REL3	REL2	REL1	REL0	T14REL
一天的时间表示 (图 17-4)	公式	$2^{10}-24$	$2^6-60$	$2^6-60$	$2^{10}-1000$	$2^{16}-33$
	重载值	3E8 <sub>H</sub>	04 <sub>H</sub>	04 <sub>H</sub>	018 <sub>H</sub>	FFDF <sub>H</sub>
	功能	小时	分钟	秒	1/1000 秒	预分频
	中断周期	天	小时	分钟	秒	毫秒 <sup>1)</sup>
一周的时间表示	公式	$2^{10}-7$	$2^6-24$	$2^6-60$	$2^{10}-600$	$2^{16}-32768$
	重载值	3F9 <sub>H</sub>	28 <sub>H</sub>	04 <sub>H</sub>	3C4 <sub>H</sub>	8000 <sub>H</sub>
	功能	天	小时	分钟	秒	预分频
	中断周期	周	天	小时	分钟	秒

1) 第一个例子 (ms) 中的 T14 误差可通过为 REL0 选择调整值、或通过软件修正的方式进行补偿。

## 17.6 周期性中断的产生

只要 RTC 的任一定时器溢出并被重载, RTC 产生中断请求。该中断请求可用作比如: 提供和 CPU 频率无关的系统时间标记 (无需重载通用定时器)。选择恰当的重载值并便能对应的中断请求, 可调整中断周期的长短。

该模式可和其它模式组合工作。比如, 重载值  $T14REL = F9C0_H$  ( $2^{16} - 1600$ ), 这样, 每隔 50 ms 产生一个 T14 中断请求。而级联链中 T14 之后的定时器仍可用来表示时间或构成二进制计数器, 只是时间基准不同。

## 17.7 RTC 中断产生

RTC 定时器链中每个定时器溢出时均可产生中断请求。RTC 的中断子节点控制寄存器 ISNC 将这些中断请求信号组合起来激活 RTC 的公共中断请求线 RTC\_IRQ。

每个定时器溢出时, 寄存器 ISNC 中对应的中断请求标志被置位。各自的中断使能位则决定所对应的中断请求是否激活公共中断请求线。被使能的中断请求逻辑或之后连接到 RTC\_ITRQ 线上 (见图 17-5)。因为同步的原因, 公共中断请求信号被延迟两个系统时钟周期。

中断处理器通过查询特定中断请求标志确定中断请求源, 在执行适当的操作后必须清除这些标志位 (不由硬件清除)。进入执行中断处理器指向的中断服务程序时, 公共节点请求位被自动清除。

**注:** 如果只有一个中断请求源被使能, 自然不需额外的软件检查。该中断请求和公共中断节点必须被使能。

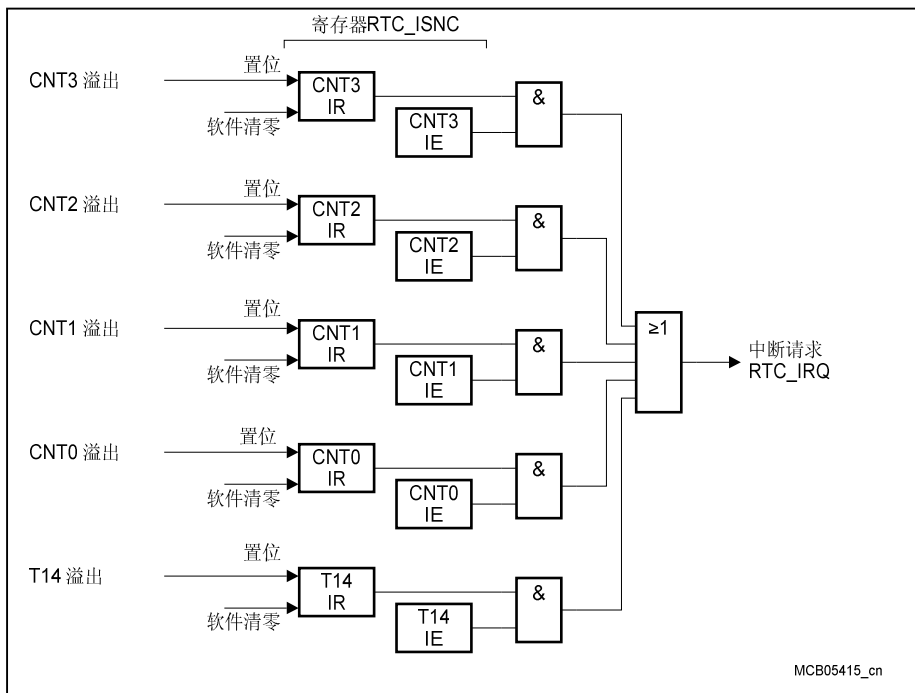


图 17-5 中断框图

## RTC\_ISNC

中断子节点控制寄存器

ESFR (F10C<sub>H</sub>/86<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	CNT 3IR	CNT 3IE	CNT 2IR	CNT 2IE	CNT 1IR	CNT 1IE	CNT 0IR	CNT 0IE	T14 IR	T14 IE
-	-	-	-	-	-	rwh	rw	rwh	rw	rwh	rw	rwh	rw	rwh	rw

符号	位序号	类型	功能描述
<b>CNTxIR</b> (x = 3...0)	9, 7, 5, 3	rwh	<b>CNTx 中断请求标志</b> 0 无中断请求挂起 1 已产生中断请求
<b>CNTxIE</b> (x = 3...0)	8, 6, 4, 2	rw	<b>CNTx 中断使能控制位</b> 0 中断请求被禁止 1 中断请求被使能
<b>T14IR</b>	1	rwh	<b>T14 溢出中断请求标志</b> 0 无中断请求挂起 1 已产生中断请求
<b>T14IE</b>	0	rw	<b>T14 溢出中断使能控制位</b> 0 中断请求被禁止 1 中断请求被使能

注：寄存器 ISNC 中的中断请求标志必须由软件清除，进入中断服务程序时这些标志位不能被硬件自动清除。

## RTC\_IC

RTC 中断控制寄存器

ESFR (F19C<sub>H</sub>/CE<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	GPX	RTC IR	RTC IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

注：控制位域的具体解释请参阅中断控制寄存器的描述。

寄存器 RTC\_IC 不在 RTC 模块中，任何应用复位均可复位该寄存器。



## 17.8 其它寄存器

### RTC\_KSCCFG

内核状态配置寄存器

ESFR (F010<sub>H</sub>)

复位值: 0001<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	0	COMCFG	BP SUM	0	SUMCFG	BP NOM	0	NOMCFG	0	BP MOD EN	MOD EN				
w	r	rw	w	r	rw	w	r	rw	r	w	rw				

符号	位序号	类型	功能描述
<b>MODEN</b>	0	rw	<b>模块使能</b> 该位使能模块内核时钟和模块功能。 0 <sub>B</sub> 模块立即关闭（不需要停止条件）。模块不能响应模块控制动作或读访问，并且忽略写访问（KSCCFG 除外）。 1 <sub>B</sub> 模块开启，可以正常工作。为了避免流水线效应，向 MODEN 写 1 之后，访问其它 RTC 寄存器之前，推荐读取寄存器 KSCCFG。
<b>BPMODEN</b>	1	w	<b>MODEN 的位保护</b> 0 <sub>B</sub> 位 MODEN 不改变 1 <sub>B</sub> 由写入值更新 MODEN。
<b>NOMCFG</b>	[5:4]	rw	<b>正常操作模式内核配置</b> 0X <sub>B</sub> 模块开启 1X <sub>B</sub> 模块关闭
<b>BPNO M</b>	7	w	<b>NOMCFG 的位保护</b> 0 <sub>B</sub> NOMCFG 不改变 1 <sub>B</sub> 由写入值更新 NOMCFG
<b>SUMCFG</b>	[9:8]	rw	<b>挂起模式配置</b> 编码与 <b>NOMCFG</b> 相同
<b>BPSUM</b>	11	w	<b>SUMCFG 的位保护</b> 0 <sub>B</sub> SUMCFG 不改变

符号	位序号	类型	功能描述
			1 <sub>B</sub> 由写入值更新 SUMCFG。
<b>COMCFG</b>	[13:12]	rw	时钟关闭模式配置 编码与 <b>NOMCFG</b> 相同
<b>BPCOM</b>	15	w	<b>COMCFG</b> 的位保护 0 <sub>B</sub> COMCFG 不改变 1 <sub>B</sub> 由写入值更新 COMCFG。
<b>0</b>	[3:2], 6,10, 14	r	保留; 读操作返回 0; 应写入 0。

注：保护位 **BPxxx** 置位时，则使能写访问对应的位域。可通过一个简单的写操作修改对应的位域，而无需读取-修改-回写的过程。只有在写访问期间这些保护位才有效，读取时返回 0。

调试复位可复位 **SUMCFG** 位域，所有其它位域都由应用复位进行复位。

## ID

### ID 寄存器

**MEM (FFF8<sub>H</sub>/FC<sub>H</sub>)**

**复位值: 5AXX<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD_TYPE</b>								<b>MOD_REV</b>							
r								r							

符号	位序号	类型	功能描述
<b>MOD_REV</b>	[7:0]	r	<b>模块版本编号</b> MOD_REV 定义版本编号。模块版本编号的值从 01 <sub>H</sub> 开始（第一版）。
<b>MOD_TYPE</b>	[15:8]	r	<b>模块类型</b> 该位域定义模块 ID 编号（5A <sub>H</sub> = RTC）。

## 18 模数转换器

XE16xyM 的模数转换器模块（ADC）采用逐次逼近技术实现了模拟输入值到离散数字值的转换。采用此方法，从 MSB 开始、逐位给出转换结果。因此，完成模数转换需要多个时钟周期（见[章节 18.1.4](#)和相关数据手册）。

本章所涉及的内容如下：

- 简介（见[章节 18.1](#)）
- 操作 ADC（见[章节 18.2](#)）
- XE16xyM 中模块的实现（见[章节 18.3](#)）

### 18.1 简介

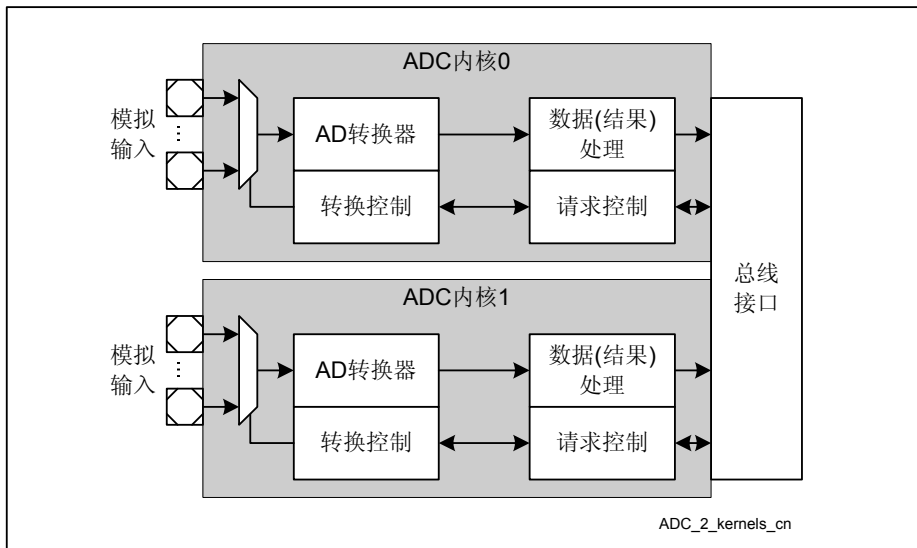
本节将简要介绍 ADC 的特性及结构。内容包括：

- 具有两个内核的 ADC 框图（见[章节 18.1.1](#)）
- 特性集说明（见[章节 18.1.2](#)）
- 缩写（见[章节 18.1.3](#)）
- 内核概述（见[章节 18.1.4](#)）
- 转换请求处理（见[章节 18.1.5](#)）
- 转换结果处理（见[章节 18.1.6](#)）
- 中断结构（见[章节 18.1.7](#)）
- 电气模型（见[章节 18.1.8](#)）
- 传送特性和错误定义（见[章节 18.1.9](#)）

#### 18.1.1 ADC 框图

ADC 模块包含两个独立的内核（ADC0、ADC1），它们可独自工作、也可同步工作。ADC 内核用于进行模数转换，并提供多种触发转换、数据处理和结果保存的方式。

ADC 模块最多可支持两路模拟输入通道的并行转换。



**图 18-1 ADC 模块框图**

### 18.1.2 特性集

每个 ADC 内核具有以下特性：

- 模拟电源电压  $V_{DDPA}$  的范围：3.3V（最小值）- 5V（标称值）
- 输入电压的范围：0V - 模拟电源电压  $V_{DDPA}$
- 最多支持 16 路模拟输入通道（CH0 – CH15）的输入复用器
- 16 路模拟输入通道（CH0 – CH15）的转换建立完全可配置
- 基于安全的考虑：输入复用器还提供对  $V_{AGND}$ （连接至 CH16）和  $V_{AREF}$ （连接至 CH17）的间接访问（替换特性）
- 10 位转换的时间小于 1  $\mu s$
- 提供一个标准参考输入（ $V_{AREF}$ ）和一个备选参考输入（CH0）
- 模拟输入 CH7 的复用器测试模式
- 每路输入通道可使能断线检测
- 3 个转换请求源，可选外部事件或定时器事件触发，自动扫描，可编程序列及软件触发等转换方式。
- 两个 ADC 内核可同步工作，用于同时开始转换、对模拟输入信号并行采样和测量（如用于测量 AC 驱动器的相电流）。

- 外部模拟复用器的控制功能，考虑附加的建立时间和扫描支持
- 采样时间可调整，从而可和不同模拟信号源（传感器等）的输出阻抗匹配
- 可取消正在执行的转换、并根据需要自动重新启动转换
- 灵活的中断产生（支持 PEC）
- 极限检查功能用于降低中断负载（如用于温度测量或过载检测，只有当数值超过可编程边界值时才产生中断）
- 可编程数据压缩滤波器（如用于数字抗混叠滤波），可叠加多个（个数可编程）转换结果
- 独立的结果寄存器（8 个独立的寄存器）
- 支持转换结果 FIFO 缓存机制，从而允许更长的中断延迟
- 支持挂起和省电模式
- 每路通道的参考输入可单独编程选择，从而可通过同一个 ADC 内核对 3.3V 和 5V 范围内的信号进行测量

### 18.1.3 缩写

本章用到的缩写归纳如下：

**表 18-1      ADC 一章中的缩写**

缩写	含义
ADC	模数转换器
DNL	差分非线性错误
FIFO	先入先出数据缓存机制
INL	积分非线性错误
LSB <sub>n</sub>	以数字格式表示的模拟电压的最小刻度，代表 n 位转换精度的最低有效位（将测量范围等分成 $2^n$ 个区间）。
PEC	外设事件控制器
SCU	系统控制单元
TUE	总不可调整误差

#### 18.1.4 ADC 内核概述

每个 ADC 内核包含以下单元：

- **模数转换器**，最多带有 16 路模拟输入（CH0 - CH15）。该模块选择一路输入信号 CHx、将该通道上的模拟电压转换为数据值。  
并非所有模拟输入通道在所有封装中均可用，具体参见“模块的实现”一节（[章节 18.3](#)）的描述。
- **转换控制单元**，定义诸如采样时间、转换精度和参考电压等转换参数。采样时间和转换精度和连接到 ADC 的传感器类型（或其它模拟输入源）有关。由于这些转换参数将应用于多路通道，因此可分组构成“输入组”。每路通道可单独分配一个输入组。  
转换控制还用于处理转换的开始条件，比如立即开始（取消-插入-重复）、覆盖前一个转换结果（等待读取）、或 ADC 内核同步（并行转换）。  
此外，外部模拟复用器可由每个 ADC 内核的输出信号 EMUX[2:0]控制。
- **请求控制单元**，定义接下来将要转换哪路模拟输入通道。它包含 3 个请求源，可根据不同的事件（如 PWM 或定时器信号的跳变沿、或端口引脚上的事件）触发转换。每个请求源可顺序触发 1 路、4 路、至最多 16 路转换。
- **结果处理单元**，提供 8 个结果寄存器用于保存转换结果。每路模拟输入通道的转换结果可直接送至一个结果寄存器中。结果处理模块还支持数据压缩（比如用于数字抗混叠滤波），通知 CPU 新数据可用之前自动叠加 4 次的转换结果。  
此外，结果寄存器可级联构成 FIFO 结构，从而可同时保存多个转换结果（前一个转换结果不被覆盖）。该特性还有助于处理 CPU 中断响应。
- **中断产生单元**，根据 ADC 事件向 CPU 发送中断请求。ADC 内核可基于不同的机制产生中断，如一些中断和转换结果所在范围（极限检查）有关，一些中断可将转换结果传送到内存单元中以备后续处理；在完成整个一系列转换之后也会产生一些中断。

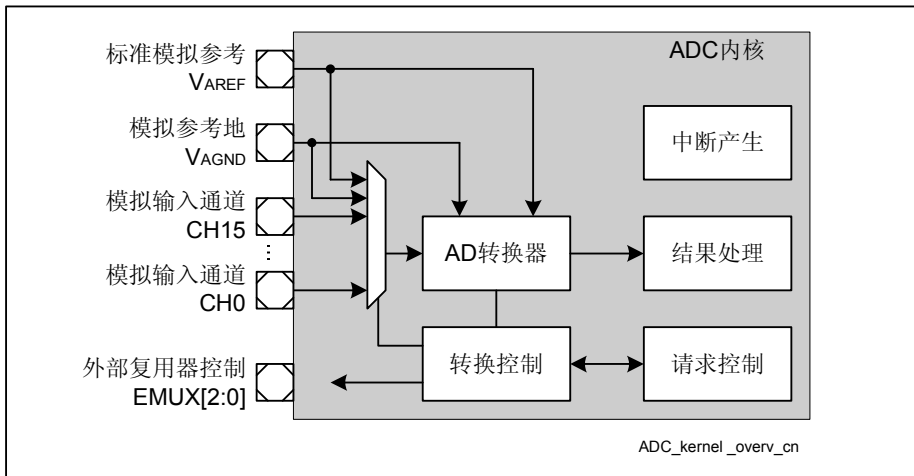


图 18-2 ADC 内核框图

转换所需的时间和结果位宽和采样时间有关。这两个参数和 ADC 时钟频率有关。转换时间的计算公式如下：

$$t_{CN} = ((N+1) \times t_{ADCI}) + ((2+STC) \times t_{ADCI}) + 2 \times t_{ADC} \quad (18.1)$$

其中  $N$  = 转换结果的位宽 (10/8)， $STC$  = 附加采样时间 (参见 [章节 18.2.4](#))。

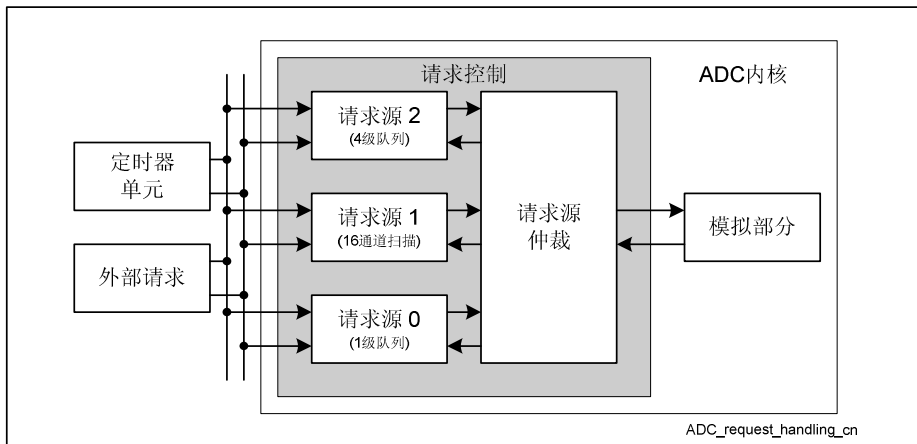
### 18.1.5 转换请求单元

每个 ADC 内核的转换请求单元自主产生转换请求。

它包含三个独立的请求源，这些请求源和不同的模块相连，用于触发模数转换。若发生设定的事件，请求源选择将要进行转换的模拟输入通道。例如，来自定时器单元的触发脉冲可启动单路输入通道的转换，也可启动一组输入通道（序列可编程）的转换。

根据实际应用的需求，请求源可由不同的事件触发。这些事件或由其它模块产生、或在软件控制下产生。因此，可有两个或多个转换请求同时挂起。为了使请求源机制能够满足不同应用的需求，每个请求源的触发功能、将要转换的通道编号、以及优先级均可单独编程。

仲裁模块规律扫描请求源、查看是否有挂起的转换请求，并对最高优先级的转换请求作出应答。该转换请求随后被送至转换器开始进行转换。



**图 18-3 转换请求单元**

请求源的功能特性可满足大多数应用的需求。对于所有请求源，均可选择连续转换模式或单次转换模式。连续转换模式下，一旦开始转换，将连续送出设定序列的转换请求；单次转换模式下，每次必须明确的启动一系列的转换请求。触发单次转换请求或序列转换请求可由软件控制、或与 ADC 外部事件（如定时器信号或端口引脚）同步。对于每一个请求源，用户可从 8 路信号 REQTRx[H:A]中选择一路作为触发输入 REQTRx；从 8 路信号 REQGTx[H:A]中选择一路作为门控输入 REQGTx。

- **请求源 0**（1 级顺序请求源）可产生单路输入通道的转换请求。通道编号可直接编程。

可通过该机制产生（单路输入通道的）软件控制的、或硬件触发的转换请求。若编程该转换请求具有高优先级，它可中断其它请求源序列、插入单个转换。

- **请求源 1**（16 通道扫描请求源）可最多产生 16 路输入通道的转换请求。哪路通道参与转换可编程设定。转换始终从被使能的最高通道编号开始，从高路通道向低路通道依次转换（转换顺序由通道编号决定，每路通道只能转换一次）。

可通过该机制恒定的、或规律性的扫描输入通道。例如，若某些输入通道被编程为低优先级，可在后台扫描这些通道以更新（非严格时序的）信息。

- **请求源 2**（4 级顺序请求源）可最多产生 4 路输入通道的转换请求。通道编号可随意编程，特别是可支持同一通道进行多次转换。

该机制是对请求源 1 扫描机制的补充、可用于支持特定应用的转换序列。特别是对于同一通道进行多次转换、对时序要求很严格的情况，应使用请求源 2。例如，若某些输入通道被编程为中等优先级，当产生指定的事件时可对这些通



道进行转换（比如和 PWM 信号同步），此时后台运行的其它输入通道的扫描操作（由请求源 1 处理）被中断。

### 18.1.6 转换结果单元

转换结果单元包含：

- **8 个结果寄存器**，用于保存转换结果。通过每路模拟输入通道的指针机制将转换结果分配给结果寄存器。尤其在自动扫描的应用中，该特性简化了 PEC 的使用（只需一路 PEC 通道即可将整个自动扫描序列传送到器件内存中）。
- 结果寄存器的**有效标志**，用于指示自上次数据读出之后已存入新数据（新数据指示）。
- **转换结果 FIFO 机制**，以“宽松的”CPU 访问时序处理转换结果。结果寄存器不直接用于保存转换结果，转换结果可级联构成一个结果 FIFO。该结构使得 CPU 在处理之前可保存多个转换结果。
- **数字抗混叠或数据压缩滤波器**，可在累加多个（个数可编程）转换结果之后再产生结果事件中断。

若多个转换结果需叠加用于后续处理（尤其是对于快速转换序列和平均转换结果的情况），可通过该特性避免 CPU 干预每次的转换结果。

- **等待读取机制**（每个结果寄存器可被单独使能）：若结果寄存器中的内容还未读出，可通过该机制使指向该结果寄存器的转换延迟进行。
- **灵活的中断产生**（基于结果寄存器事件）。若结果寄存器中新的有效数据可用并可被读出，产生结果寄存器事件。特别是当使用数据压缩或数字抗混叠滤波时，结果寄存器事件指示最终结果可用。
- 和 XC16x 器件的 ADC 结果寄存器兼容的结果寄存器，其值可从一个地址单元中读取；支持数据压缩功能的结果寄存器，其值可从另一个地址单元中读取。
- ADC 结果寄存器的**调试支持功能**，可读出 ADC 的转换结果而不改变结果的状态（新数据指示）。

### 18.1.7 中断结构

每个 ADC 内核提供 4 路独立的服务请求输出信号（ADCx\_SR[3:0]）用于中断处理（SRx 信号连接到中断控制寄存器上）。ADC 内核基于三种不同的事件产生中断。

- **通道事件**

若完成一次转换、转换结果在设定的数值范围内，产生通道事件。

可通过该事件检查模拟输入值是否在额定的工作范围内，尤其可用于降低 CPU 处理后台任务的负载。只有当转换结果不超出（或超出）指定的转换结果范围时 CPU 才被中断，从而不需通过软件比较每次转换的结果。

- **结果事件**

若结果寄存器中新的转换结果可用并可被读出（比如读出后存放在内存中以备用后续软件处理），产生结果事件。

可通过该事件触发 CPU（或 PEC）执行读操作。特别是当使用数据压缩或数字抗混叠滤波时，并非所有完成的转换都会产生新的结果。进一步讲，当使用结果 FIFO 时，结果事件使得 CPU（PEC）读取和通道事件无关、可允许更长的中断延迟。若所有参与自动扫描序列的通道的转换结果均指向同一个结果寄存器（可采用 FIFO 或等待读取机制避免数据丢失），通过结果事件触发读取操作时，结果寄存器结构实现了利用单路 PEC 通道来处理整个自动扫描序列。

### • 请求源事件

若扫描请求源已完成请求的转换序列，产生请求源事件。对于顺序请求源，用户可定义在转换序列中产生请求源事件的位置。

可通过该事件通知 CPU 转换序列以达到指定的状态，软件可开始对相关结果进行处理。

每个 ADC 事件可由专用的标志位指示、该标志位可由软件清零。若允许产生中断，每个事件可产生一个中断，这和事件指示标志的状态无关。该结构可确保 PEC 高效处理 ADC 事件（ADC 事件可产生中断而无需清除指示标志位）。中断节点指针机制可将中断事件分组，定义哪个事件将激活哪路服务请求输出信号 SRx。为了便于满足应用的需求，每个 ADC 事件可单独送至一路服务请求输出信号上。

*注：一次转换可产生三种不同类型的中断。在这种情况下，ADC 模块首先触发请求源事件中断、然后是通道事件中断、最后是结果事件中断（所有操作在几个  $f_{ADC}$  时钟周期内完成）。*

## 18.1.8 电气模型

将模拟输入电压转换为数字值的操作包含采样和转换两个阶段。在采样阶段，输入电压被采样、并为随后的转换阶段做准备。采样阶段的简化模型描述了输入信号通路；转换阶段的简化模型和参考电压的处理有关。

### 18.1.8.1 输入信号通路

XE16xyM 的 ADC 内核基于一个用于测量的开关电容器（总容量由  $C_{AIN}$  来表示），另外每个输入端分别有一个小容量的静态电容器。在采样阶段，开关电容器  $C_{AIN}$  通过输入复用器和一路模拟输入 CHx 相连。复用器由理想开关和串联电阻  $R_{AIN}$  构成。在采样阶段，只闭合所选模拟输入的开关。在转换阶段或当不执行转换时（ADC 空闲），所有开关均断开。模拟输入通道 CHx 的电压由  $V_{AINx}$  表示。

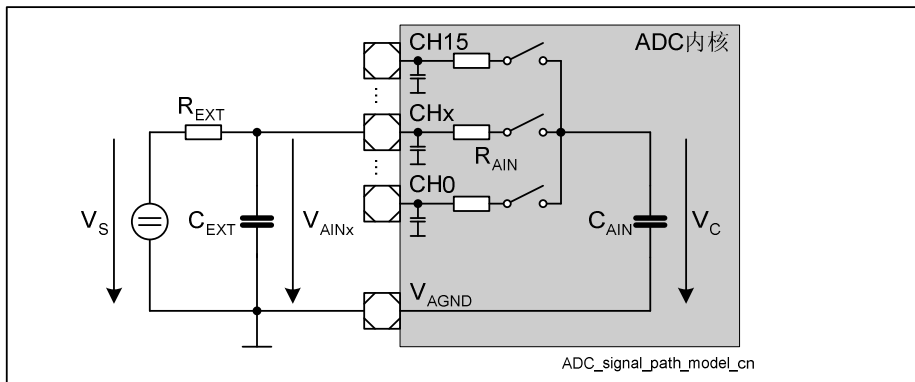


图 18-4 信号通路模型

模拟输入信号通路的简化模型如图 18-4 所示。具有内部阻抗  $R_{EXT}$  的模拟电压源 ( $V_S$ ) 作为被转换的模拟输入。

在采样阶段，相应的开关闭合、对  $C_{AIN}$  充电。由于 RC 电路具有低通特性，因此，实际待转换的电压  $V_C$  不会立即等于电压  $V_S$ 。模拟电压源的  $R_{EXT}$  和期望的转换精度主要决定了采样时间的长短。

为了降低  $R_{EXT}$  的影响并过滤输入噪声，建议在 ADC 的模拟输入引脚上引入快速外部隔直电容  $C_{EXT}$ ，此时，采样阶段主要由  $C_{EXT}$  放电。带隔直电容的结构比不带隔直电容的结构大大缩短了采样时间，这是由于决定采样时间的低通时间常量主要由  $R_{AIN}$  和  $C_{AIN}$  的值给出。

此外，为了使采样开始阶段的  $V_{AINX}$  和  $V_C$  之间的均差最小，电容  $C_{AIN}$  被自动预充电到约为标准参考电压  $V_{AREF}$  的一半电压值。由于可变参数和寄生效应，通常  $C_{AIN}$  的预充电电压小于  $V_{AREF}/2$ 。另一方面，采样期间  $C_{EXT}$  和  $C_{AIN}$  间的电荷重分配会导致  $V_{AINX}$  电压改变。为了使该电压变化小于 1 个  $LSB_n$ ，建议选用外部隔直电容  $C_{EXT}$  至少为  $2^n \times C_{AIN}$ 。

$R_{EXT}$  和  $C_{EXT}$  低通滤波器的选取应该遵循：使每路模拟输入通道的  $V_{AINX}$  能在两个采样阶段的时间内达到  $V_S$ 。

需要注意，尤其是在高温的情况下，ADC 的模拟输入结构可导致漏电流并引入误差（由  $R_{EXT}$  上的压降引起）。若输入电压接近模拟电源地信号  $V_{SS}$  或模拟电源电压  $V_{DDPA}$ ，ADC 的输入漏电流会增加。建议输入电压的工作范围介于  $V_{DDPA}$  的 3%-97% 之间，从而降低 ADC 各通道的输入漏电流。

进一步讲，漏电受相邻模拟输入过载状况的影响。在过载情况下，输入端上施加的电压超过了供电电压，内置的保护电路会限制最终的输入电压。这会导致一个通过保护电路的过载电流，（通过耦合因子）转化为相邻输入上的附加漏电。

### 18.1.8.2 参考通路

在转换阶段， $C_{AIN}$  被切换至参考输入或  $V_{AGND}$ 。ADC 内核支持两种参考输入： $V_{AREF}$  作为标准参考输入； $CH0$  作为备选参考输入。每路模拟输入通道可单独选择参考输入。例如，该结构可实现基于 5V 和 3.3V 的模拟输入信号的转换。

转换结果的高准确度要求转换阶段具有稳定、无噪声的参考电压和模拟电源电压。电源或参考输入上的不稳电压或噪声会使转换准确度降低。需要注意：其它模块也会向 ADC 模块引入噪声（如切换临近引脚的操作）。强烈建议用户将模拟和数字信号域明确分开。

由于需要切换  $C_{AIN}$ ，在选定的参考输入上 ADC 需要动态电流。因此，参考电压源  $V_R$  的阻抗  $R_{AREF}$  必须低到足以提供转换阶段所需的参考电流。应使用外部隔直电容  $C_{AREF}$  提供峰值电流，使参考输入提供的电流最小。

在转换期间， $C_{AREF}$  和  $C_{AIN}$  之间的电荷会重分配，因此电压  $V_{AREF}$  将降低。为了把相应的误差限制到  $1/2$  个  $LSB_n$ ，参考输入的外部隔直电容  $C_{AREF}$  至少应该是  $2^n \times C_{AIN}$ 。

参考电流  $I_{AREF}$  会导致  $R_{AREF}$  上的压降，在计算总准确度时应考虑在内。转换期间的平均参考电流和参考电压以及两次转换之间的时间  $t_{CONV}$  有关。

$$I_{AREF} = C_{AIN} \times V_{AREF} / t_{CONV}$$

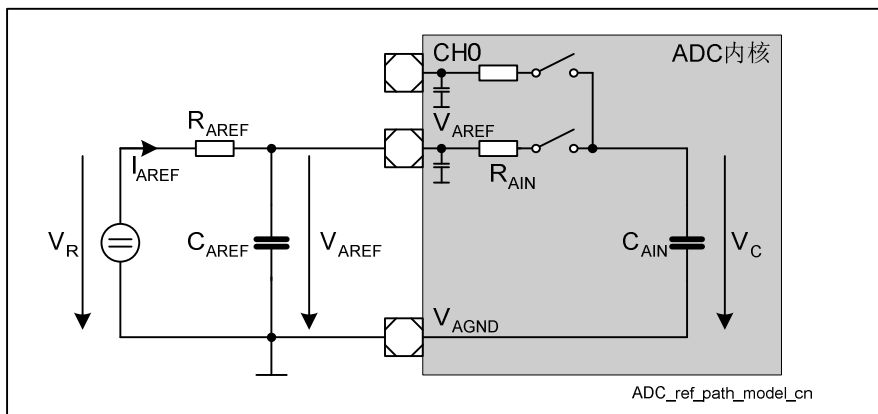


图 18-5 参考通路模型

### 18.1.9 转换特性和误差定义

ADC 的理想转换特性是将 0-参考电压之间的连续的模拟输入值翻译为离散的数字值， $n$  位转换精度的结果在  $2^n$  步长之内。可用结果范围内（0 -  $2^n - 1$ ）的每个数字值代表一个由参考电压/ $2^n$  决定的输入电压范围。该范围（称为量化步长）代表可由 ADC 处理的最小刻度（称作  $LSB_n$ ）。由于数字结果具有离散特性，ADC 的每个转换结果会有一

个 $\pm 0.5 \text{ LSB}_n$ 的系统固有量化误差。根据理想转换特性，当模拟输入达到  $0.5 \text{ LSB}_n$  时发生第一次数字跳变（在数字值 0 和 1 之间）。

超过参考电压的模拟输入电压使转换的数字结果始终为  $2^n - 1$ （饱和）。

实际的转换结果可产生偏差：

- **偏移误差：**输入电压接近 0 时，转换结果偏离理想转换特性的误差。它给出第一次数字跳变（数字值 0 和 1 之间）时  $0.5 \text{ LSB}_n$  和输入电压之间的差值。
- **增益误差：**输入电压接近参考电压时，转换结果偏离理想转换特性的误差。它给出最后一次数字跳变（数字值  $2^n - 2$  和  $2^n - 1$  之间）时，参考电压和输入电压之间的差值。
- **差分非线性误差（DNL）：**它给出在整个测量范围内，相邻两个数字转换结果对应的模拟输入电压之间的差值。若数字转换结果  $x$  和  $x+1$  之间的每个步长恰好为  $1 \text{ LSB}_n$ ，DNL 为 0。若 DNL 小于  $1 \text{ LSB}_n$ ，排除编码丢失的可能。若无法得到所有可能的转换结果值，发生编码丢失。
- **积分非线性误差（INL）：**它给出从第一个到最后一个测量点之间的实际转换特性和理想转换特性之间的最大差值（无量化误差、偏移和增益误差）。
- **总不可调整误差（TUE）：**它给出在给定的测量范围内，实际转换结果和理想转换特性之间的最大偏差。由于有些误差可相互补偿，因此，TUE 通常远小于各种误差的总和。

TUE 还包含了生产工艺偏差和内部噪声的影响（若系统产生开关噪声，这通常会导致 TUE 增加）。

## 18.2 操作 ADC

本节说明 ADC 内核功能以及如何操作内核，给出功能以及相关寄存器的描述。

- 寄存器概览（见[章节 18.2.1](#)）

通用的模块、内核和仲裁器操作：

- 配置 ADC 模块在不同器件操作模式下的行为（见[章节 18.2.2](#)“模式控制”）
- 使能转换器正常工作或选择期望的省电模式（见[章节 18.2.3](#)）
- 为转换器和请求源仲裁器选择恰当的输入频率（见[章节 18.2.4](#)）
- 通用 ADC 寄存器（见[章节 18.2.5](#)）
- 配置请求源仲裁器（见[章节 18.2.6](#)）
- 仲裁器寄存器（见[章节 18.2.7](#)）

请求源操作：

- 扫描请求源处理（见[章节 18.2.8](#)）
- 扫描请求源寄存器（见[章节 18.2.9](#)）
- 顺序请求源处理（见[章节 18.2.10](#)）

- 顺序请求源寄存器（见[章节 18.2.11](#)）

**通道和结果寄存器操作：**

- 配置通道相关功能（见[章节 18.2.12](#)）
- 通道相关寄存器（见[章节 18.2.13](#)）
- 转换结果处理（见[章节 18.2.14](#)）
- 转换请求处理（见[章节 18.2.15](#)）

**附加特性：**

- CH7 的复用器测试模式（见[章节 18.2.16](#)）
- 外部复用器控制（见[章节 18.2.17](#)）
- 并行同步转换（见[章节 18.2.18](#)）
- 等间隔采样（见[章节 18.2.19](#)）
- 断线检测（见[章节 18.2.20](#)）
- 附加特性寄存器（见[章节 18.2.21](#)）

## 18.2.1 寄存器概述

**表 18-2** 给出编程 ADC 模块所需的所有寄存器。它归纳了 ADC 内核寄存器并给出其偏移地址和复位值。偏移地址和基地址相加得到每个寄存器的绝对地址（见[章节 18.3.1](#)）。

为了区分不同内核的寄存器，必须给**表 18-2** 中的每个寄存器添加前缀“**ADCx\_**”，此时 x 代表内核编号。

所有 ADC 寄存器（包括位域 KSCFG.NOMCFG 和 KSCFG.COMCFG）由应用复位来复位。位域 KSCFG.SUMCFG 由调试复位来复位。

*注：当寄存器中的某些位标记为“w”时，读取它们始终返回 0。*

**表 18-2      ADC 模块寄存器归纳**

寄存器缩写名	寄存器完整名	偏移地址 <sup>1)</sup>	描述见
<b>一般寄存器</b>			
<b>ID</b>	模块 ID 寄存器	08 <sub>H</sub>	<a href="#">页 18-20</a>
<b>KSCFG<sup>2)</sup></b>	内核状态配置寄存器	0C <sub>H</sub>	<a href="#">页 18-18</a>
<b>GLOBCTR</b>	全局控制寄存器	10 <sub>H</sub>	<a href="#">页 18-21</a>
<b>GLOBSTR</b>	全局状态寄存器	12 <sub>H</sub>	<a href="#">页 18-22</a>

寄存器缩写名	寄存器完整名	偏移地址 <sup>1)</sup>	描述见
<b>RISR0</b>	请求源 0 输入选择寄存器	00 <sub>H</sub>	<a href="#">页 18-24</a>
<b>RISR1</b>	请求源 1 输入选择寄存器	02 <sub>H</sub>	<a href="#">页 18-24</a>
<b>RISR2</b>	请求源 2 输入选择寄存器	04 <sub>H</sub>	<a href="#">页 18-24</a>

**仲裁器寄存器**

<b>ASENR</b>	仲裁时隙使能寄存器	18 <sub>H</sub>	<a href="#">页 18-30</a>
<b>RSPR0</b>	请求源优先级寄存器 0	14 <sub>H</sub>	<a href="#">页 18-31</a>

**通道相关寄存器**

<b>CHCTR0-15</b>	通道控制寄存器 0-15	20 <sub>H</sub> - 3E <sub>H</sub>	<a href="#">页 18-53</a>
<b>INPCR0</b>	输入组寄存器 0	C0 <sub>H</sub>	<a href="#">页 18-53</a>
<b>INPCR1</b>	输入组寄存器 1	C2 <sub>H</sub>	<a href="#">页 18-53</a>
<b>LCBR0</b>	极限检查边界寄存器 0	84 <sub>H</sub>	<a href="#">页 18-55</a>
<b>LCBR1</b>	极限检查边界寄存器 1	86 <sub>H</sub>	<a href="#">页 18-55</a>
<b>LCBR2</b>	极限检查边界寄存器 2	88 <sub>H</sub>	<a href="#">页 18-55</a>
<b>LCBR3</b>	极限检查边界寄存器 3	8A <sub>H</sub>	<a href="#">页 18-55</a>
<b>CHINFR</b>	通道事件指示标志寄存器	90 <sub>H</sub>	<a href="#">页 18-57</a>
<b>CHINCR</b>	通道事件指示清零寄存器	92 <sub>H</sub>	<a href="#">页 18-57</a>
<b>CHINPR0</b>	通道中断节点指针寄存器 0	98 <sub>H</sub>	<a href="#">页 18-58</a>
<b>CHINPR4</b>	通道中断节点指针寄存器 4	9A <sub>H</sub>	<a href="#">页 18-58</a>
<b>CHINPR8</b>	通道中断节点指针寄存器 8	9C <sub>H</sub>	<a href="#">页 18-59</a>
<b>CHINPR12</b>	通道中断节点指针寄存器 12	9E <sub>H</sub>	<a href="#">页 18-60</a>
<b>ALR0</b>	替换寄存器 0	1C <sub>H</sub>	<a href="#">页 18-60</a>

**结果寄存器**

<b>RESR0-7</b>	结果寄存器 0-7, 正常读取	40 <sub>H</sub> - 4E <sub>H</sub>	<a href="#">页 18-67</a>
<b>RESRA0-7</b>	结果寄存器 0-7, 读取 A	50 <sub>H</sub> - 5E <sub>H</sub>	<a href="#">页 18-68</a>
<b>RESRV0-7</b>	结果寄存器 0-7, 读取 V	60 <sub>H</sub> - 6E <sub>H</sub>	<a href="#">页 18-67</a>

寄存器缩写名	寄存器完整名	偏移地址 <sup>1)</sup>	描述见
<b>RESRAV0-7</b>	结果寄存器 0-7, 读取 AV	70 <sub>H</sub> - 7E <sub>H</sub>	<a href="#">页 18-68</a>
<b>VFR</b>	有效标志寄存器	80 <sub>H</sub>	<a href="#">页 18-69</a>
<b>RSSR</b>	结果状态映射寄存器	82 <sub>H</sub>	<a href="#">页 18-69</a>
<b>RCR0-7</b>	结果控制寄存器 0-7	B0 <sub>H</sub> - BE <sub>H</sub>	<a href="#">页 18-70</a>
<b>EVINFR</b>	事件指示标志寄存器	A0 <sub>H</sub>	<a href="#">页 18-72</a>
<b>EVINCR</b>	事件指示清零寄存器	A2 <sub>H</sub>	<a href="#">页 18-73</a>
<b>EVINPRO</b>	事件中断节点指针寄存器 0	A8 <sub>H</sub>	<a href="#">页 18-74</a>
<b>EVINPR8</b>	事件中断节点指针寄存器 8	AC <sub>H</sub>	<a href="#">页 18-74</a>
<b>EVINPR12</b>	事件中断节点指针寄存器 12	AE <sub>H</sub>	<a href="#">页 18-75</a>

**请求源 0 寄存器**

<b>QMR0</b>	队列 0 模式寄存器	E0 <sub>H</sub>	<a href="#">页 18-42</a>
<b>QSR0</b>	队列 0 状态寄存器	E2 <sub>H</sub>	<a href="#">页 18-44</a>
<b>Q0R0</b>	队列 0 寄存器 0	E4 <sub>H</sub>	<a href="#">页 18-45</a>
<b>QBUR0</b>	队列 0 备份寄存器	E6 <sub>H</sub> 共用	<a href="#">页 18-46</a>
<b>QINR0</b>	队列 0 输入寄存器		<a href="#">页 18-48</a>

**请求源 1 寄存器**

<b>CRCR1</b>	转换请求 1 控制寄存器	E8 <sub>H</sub>	<a href="#">页 18-35</a>
<b>CRPR1</b>	转换请求 1 挂起寄存器	EA <sub>H</sub>	<a href="#">页 18-35</a>
<b>CRMR1</b>	转换请求 1 模式寄存器	EC <sub>H</sub>	<a href="#">页 18-36</a>

**请求源 2 寄存器**

<b>QMR2</b>	队列 2 模式寄存器	F0 <sub>H</sub>	<a href="#">页 18-42</a>
<b>QSR2</b>	队列 2 状态寄存器	F2 <sub>H</sub>	<a href="#">页 18-44</a>
<b>Q0R2</b>	队列 2 寄存器 0	F4 <sub>H</sub>	<a href="#">页 18-45</a>
<b>QBUR2</b>	队列 2 备份寄存器	F6 <sub>H</sub> 共用	<a href="#">页 18-46</a>
<b>QINR2</b>	队列 2 输入寄存器		<a href="#">页 18-48</a>



寄存器缩写名	寄存器完整名	偏移地址 <sup>1)</sup>	描述见
<b>附加特性寄存器</b>			
<b>SYNCTR</b>	同步控制寄存器	1A <sub>H</sub>	页 18-87
<b>EMENR</b>	外部复用器使能寄存器	D6 <sub>H</sub>	页 18-84
<b>EMCTR</b>	外部复用器控制寄存器	D0 <sub>H</sub>	页 18-86
<b>BWDENR</b>	断线检测使能寄存器	C8 <sub>H</sub>	页 18-88
<b>BWDCFG</b>	断线检测配置寄存器	CA <sub>H</sub>	页 18-89

1) 对于该模块的内核寄存器，8 位短地址不可用。

2) 寄存器 KSCFG 仅在 ADC0 的地址段可用，称为 ADC0\_KSCFG。

注：偏移地址 06<sub>H</sub>、16<sub>H</sub>、C4<sub>H</sub>、C6<sub>H</sub>、8C<sub>H</sub>、8E<sub>H</sub>、A4<sub>H</sub>、A6<sub>H</sub> 和 AA<sub>H</sub> 预留以备将来使用，切勿访问这些地址。

## 18.2.2 模式控制

对于不同的系统控制任务（如降低功耗、或挂起请求进行调试等），可通过模式控制设定模块在不同器件操作条件下的操作。ADC 内核在每种器件操作模式下的行为可编程设定，不同的器件操作模式由 SCU 的全局状态控制部分请求。在不同的器件操作模式下，ADC 模块中两个内核的行为相同（从而避免未挂起内核等待被挂起内核以开始同步转换的情况）。因此，ADC 模块中有一个共用相关寄存器 **ADC0\_KSCFG**，它用来定义所有内核在不同器件操作模式下的行为：

- **正常操作：**

该模式是既无挂起请求、也无时钟关闭请求时的缺省操作模式。模块时钟不关闭、ADC 寄存器可被读写。内核行为由 KSCFG.NOMCFG 定义。

- **挂起模式：**

挂起请求（由调试器发送）有效时进入该模式。模块时钟不关闭、ADC 寄存器可被读写。内核行为由 KSCFG.SUMCFG 定义。

- **时钟关闭模式：**

为了降低系统功耗，请求进入该模式。当 ADC 模块中的所有内核达到其在终止模式下指定的状态时，模块时钟自动关闭。此时 ADC 寄存器不能被访问。内核行为由 KSCFG.COMCFG 定义。

以下内部操作可受模式控制的影响：

- **模拟电压值的当前转换：**

若请求控制单元检测到一个挂起的转换请求，可以启动转换。开始转换的操作必须由模式控制使能。若当前的内核模式允许开始转换（运行模式 0 和 1），

则执行转换操作；若当前的内核模式不允许开始转换（终止模式 0 和 1），则不能执行转换操作。此时，开始转换的请求未被取消、但被冻结。当内核模式切换回运行模式后，重新开始执行“被冻结的”转换。

- 仲裁周期：

开始新一轮仲裁必须由内核模式使能。对于终止模式 1，不能开始新一轮仲裁。

ADC 内核在每种器件操作模式（正常操作、挂起模式、时钟关闭模式）下的行为可编程设定。因此，ADC 内核支持四种内核模式，归纳见表 18-3。

**表 18-3 ADC 内核行为**

内核模式	内核行为	编码
运行模式 0	执行所设定的内核操作；不影响数据传送（运行模式 0 和运行模式 1 的行为相同）	00 <sub>B</sub>
运行模式 1		01 <sub>B</sub>
终止模式 0	完成当前正在执行的转换并对结果进行处理。对挂起的转换请求不予响应（但不删除）。在切换回运行模式之后执行先前已请求的转换。仲裁器继续执行所设定的操作。	10 <sub>B</sub>
终止模式 1	和终止模式 0 的操作相似，区别在于仲裁器在完成本轮仲裁后停止工作。  此外，当内核达到设定的终止条件时，位域 GLOBSTR.ANON 被视为 00 <sub>B</sub> （位域本身不改变）。	11 <sub>B</sub>

通常，位域 KSCFG.NOMCFG 应配置为运行模式 0，这是正常操作的缺省模式。若 ADC 内核不应响应挂起请求（并继续其操作），必须使位域 KSCFG.SUMCFG 的值和 KSCFG.NOMCFG 相同。当达到某个特定的终止条件时，若 ADC 内核应执行不同的终止操作，必须将 KSCFG.SUMCFG 设置为终止模式 0 或终止模式 1。

在时钟关闭模式下，通过位域 KSCFG.COMCFG 设置所期望的操作（和挂起模式的机制相似）。

根据所选择的协议来定义终止条件（参见协议章节中有关模式控制的描述）。

*注：终止模式的选择主要取决于实际的应用需求，在同一种应用中几乎不可能同时请求不同的终止模式。因此，在寄存器 KSCFG 中只应选用一种终止模式类型（0 或 1）。在 ADC 模块中，禁止终止模式 0 和终止模式 1 混用，避免从终止模式 0 切换到终止模式 1（反之亦然）。*

若设置 `KSCFG.MODEN = 0` 关闭模块时钟，或满足终止条件（终止模式 0 或 1）进入时钟关闭模式，模块不能被读写（寄存器 `KSCFG` 除外，它始终可被访问），因而它不能被配置。

需要注意：只有当所有配置域被配置为运行模式 0 时，位 `KSCFG.MODEN` 才能由软件置位。

### 18.2.3 模块激活和省电模式

ADC 的转换器支持特定的省电模式，在两次转换之间可自动降低功耗。通过位域 `GLOBSTR.ANON` 选择以下各种省电模式：

- **ANON = 00<sub>B</sub>: 转换器关闭**（复位缺省模式）

整个转换器被关闭、保持其复位状态，不可能进行转换。为了启动转换，`ANON` 必须设置成期望的模式。启动转换之前必须将 10  $\mu$ s 左右的最大唤醒时间考虑在内。此外，数字逻辑模块被置为其初始状态。

- **ANON = 01<sub>B</sub>: 低速待机模式**

在完成每次转换之后，转换器进入功耗降低模式。若请求转换，它会自动切换到正常操作。必须将 10  $\mu$ s 左右的最大唤醒时间叠加到采样时间上。该模式的功耗最低。

- **ANON = 10<sub>B</sub>: 快速待机模式**

在完成每次转换之后，转换器进入功耗降低模式，该模式比低速待机模式的功耗高。若请求转换，它会自动切换到正常操作。必须将 3  $\mu$ s 左右的最大唤醒时间叠加到采样时间上。该模式比正常操作的功耗低。

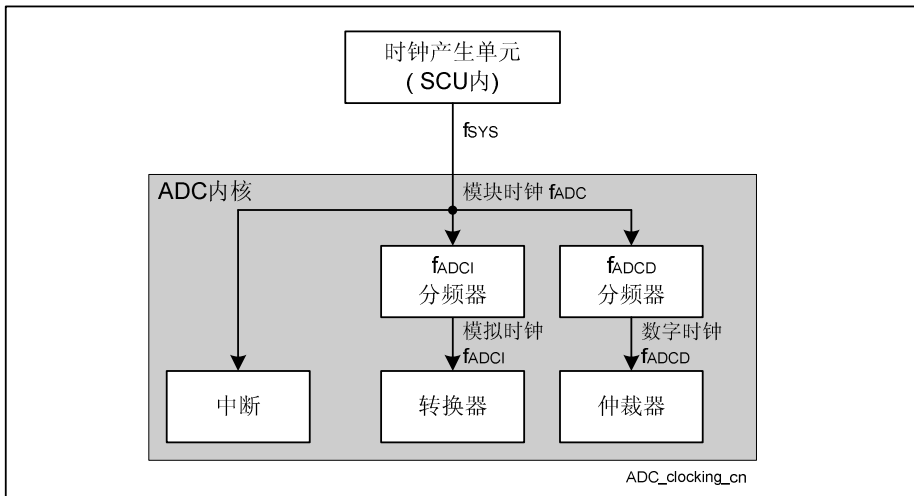
- **ANON = 11<sub>B</sub>: 正常操作**

始终可进行转换操作。转换器恒保持有效。

### 18.2.4 时钟方案

ADC 内核中不同组成单元的驱动时钟不同，这些时钟均基于  $f_{\text{ADC}}$  时钟产生。由于 XE16xyM 中的 ADC 和系统时钟相连，因此  $f_{\text{ADC}} = f_{\text{SYS}}$ 。

- 模拟时钟  $f_{\text{ADCI}}$  用作转换器的内部时钟，定义转换时间和采样时间。 $f_{\text{ADCI}}$  可通过编程位域 `GLOBCTR.DIVA` 进行修改。
- $f_{\text{ADCD}}$  用作仲裁器的输入时钟，定义仲裁周期的长短。 $f_{\text{ADCD}}$  可通过编程位域 `GLOBCTR.DIVD` 进行修改。
- 所有其它数字结构（如中断等）直接由模块时钟  $f_{\text{ADC}}$  驱动。



**图 18-6 时钟方案**

*注：在模数转换过程中，若产生的转换器时钟小于规定的最小值、或该时钟停止，转换结果可被破坏。为了获得正确的转换结果， $f_{ADCI}$  的频率切勿超过规定的范围，频率范围的具体信息参见相关数据手册。*

## 18.2.5 通用 ADC 寄存器

### 18.2.5.1 内核状态配置寄存器

可通过内核状态配置寄存器 KSCFG 选择不同器件操作模式下所期望的内核模式。

位域 KSCFG.NOMCFG 和 KSCFG.COMCFG 通过应用复位来复位。位域 KSCFG.SUMCFG 由调试复位来复位。

KSCFG 是所有 ADC 内核的共用寄存器，可从 ADC0 的地址段对其进行访问。

*注：位域 NOMCFG、SUMCFG 和 COMCFG 的编码说明见[表 18-3](#)。*

#### ADC0\_KSCFG

##### 内核状态配置寄存器

XSFR (0C<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	0	COMCFG	BP SUM	0	SUMCFG	BP NOM	0	NOMCFG	0					BP MOD EN	MOD EN
w	r	rw	w	r	rw	w	r	rw	r					w	w

符号	位序号	类型	功能描述
<b>MODEN</b>	0	rw	<b>模块使能</b> 该位使能模块内核时钟和模块功能。 <b>0<sub>B</sub></b> 立即关闭模块（不考虑终止条件）。模块对控制操作不予响应、模块时钟关闭。模块不响应读操作，忽略写操作（KSCFG 除外）。 <b>1<sub>B</sub></b> 开启模块正常工作。 <b>MODEN</b> 置 1 时，建议在访问其它 ADC 寄存器之前先读取 KSCFG，从而避免流水线造成影响。
<b>BPMODEN</b>	1	w	<b>MODEN 的位保护</b> 该位使能对 <b>MODEN</b> 进行写操作。读取始终为 0。 <b>0<sub>B</sub></b> 不改变 <b>MODEN</b> 。 <b>1<sub>B</sub></b> 写操作更新 <b>MODEN</b> 。
<b>NOMCFG</b>	[5:4]	rw	<b>正常操作模式配置</b> 该位域选择正常操作模式下所用的内核模式。 <b>00<sub>B</sub></b> 选择运行模式 0。 <b>01<sub>B</sub></b> 选择运行模式 1。 <b>10<sub>B</sub></b> 选择终止模式 0。 <b>11<sub>B</sub></b> 选择终止模式 0。
<b>BPNO M</b>	7	w	<b>NOMCFG 的位保护</b> 该位使能对 <b>NOMCFG</b> 进行写操作。读取始终为 0。 <b>0<sub>B</sub></b> 不改变 <b>NOMCFG</b> 。 <b>1<sub>B</sub></b> 写操作更新 <b>NOMCFG</b> 。
<b>SUMCFG</b>	[9:8]	rw	<b>挂起模式配置</b> 该位域选择挂起模式下所用的内核模式。 <b>SUMCFG</b> 和 <b>NOMCFG</b> 的编码相同。
<b>BPSUM</b>	11	w	<b>SUMCFG 的位保护</b> 该位使能对 <b>SUMCFG</b> 进行写操作。读取始终为 0。 <b>0<sub>B</sub></b> 不改变 <b>SUMCFG</b> 。 <b>1<sub>B</sub></b> 写操作更新 <b>SUMCFG</b> 。

COMCFG	[13:12]	rw	<b>时钟关闭模式配置</b> 该位域选择时钟关闭模式下所用的内核模式。 COMCFG 和 NOMCFG 的编码相同。
BPCOM	15	w	<b>COMCFG 的位保护</b> 该位使能对 COMCFG 进行写操作。读取始终为 0。 0 <sub>B</sub> 不改变 COMCFG。 1 <sub>B</sub> 写操作更新 COMCFG。
0	[3:2], 6,10, 14	r	<b>保留</b> 读操作返回 0；应写入 0。

注：若要修改配置位域，进行写操作时必须置位相关的保护位 BPxxx。这样则可以通过单次写操作修改配置位域。

#### 18.2.5.2 ID 寄存器

ID 寄存器是只读寄存器，用于指示 ADC 模块的 ID 信息。其中 8 位指示模块 ID 编号、8 位指示版本编号。

#### ID

模块 ID 寄存器

XSFR (08<sub>H</sub>)

复位值: 33XX<sub>H</sub>

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

符号	位序号	类型	功能描述
MOD_REV	[7:0]	r	<b>模块修订版编号</b> 该位域定义模块修订版编号，从 01 <sub>H</sub> 开始编号（第一版），接下来是 02 <sub>H</sub> 、03 <sub>H</sub> 、直至 FF <sub>H</sub> 。
MOD_NUMBER	[15:8]	r	<b>模块 ID 编号</b> 该位域定义模块的 ID 编号：33 <sub>H</sub> = ADC。

### 18.2.5.3 全局控制寄存器

全局控制寄存器用于控制仲裁器时序和转换器的通用使能功能。

#### GLOBCTR

##### 全局控制寄存器

**XSFR (10<sub>H</sub>)**

**复位值: 00FF<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>ARB M</b>	<b>0</b>		<b>ARBRND</b>	<b>ANON</b>		<b>DIVD</b>		<b>DIVA</b>							
rw	rw		rw	rw		rw		rw							

符号	位序号	类型	功能描述
<b>DIVA</b>	[5:0]	rw	<b>模拟内部时钟的分频因子</b> 该位域定义转换器时钟 $f_{ADCI}$ （用于计算转换时间和采样时间），由 $f_{ADC}$ 时钟周期数表示。 $00_H \quad f_{ADCI} = f_{ADC}$ $01_H \quad f_{ADCI} = f_{ADC} / 2$ $02_H \quad f_{ADCI} = f_{ADC} / 3$ ... $3F_H \quad f_{ADCI} = f_{ADC} / 64$
<b>DIVD</b>	[7:6]	rw	<b>数字仲裁器时钟的分频因子</b> 该位域定义每个仲裁时隙的 $f_{ADCD}$ 时钟周期数（每个仲裁时隙对应一个 $f_{ADCD}$ 周期）。 建议使用缺省设置 $00_B$ 以获得最小仲裁响应时间。 $00_B \quad f_{ADCD} = f_{ADC}$ $01_B \quad f_{ADCD} = f_{ADC} / 2$ $10_B \quad f_{ADCD} = f_{ADC} / 3$ $11_B \quad f_{ADCD} = f_{ADC} / 4$
<b>ANON</b>	[9:8]	rw	<b>模拟部分开启控制</b> 若该内核是同步主控方或不具备同步特性（见寄存器 <b>SYNCTR</b> ），该位域定义 <b>GLOBSTR.ANON</b> 的设置。若内核为同步从控方，该位域不予考虑。
<b>ARBRND</b>	[11:10]	rw	<b>仲裁周期长度</b> 该位域定义每个仲裁周期中的仲裁时隙数（仲裁周

符号	位序号	类型	功能描述
			期长度 = $t_{ARB}$ <sup>1)</sup> 。 <b>00<sub>B</sub></b> 1 个仲裁周期包含 4 个仲裁时隙 ( $t_{ARB} = 4 / f_{ADCD}$ )。 <b>01<sub>B</sub></b> 1 个仲裁周期包含 8 个仲裁时隙 ( $t_{ARB} = 8 / f_{ADCD}$ )。 <b>10<sub>B</sub></b> 1 个仲裁周期包含 16 个仲裁时隙 ( $t_{ARB} = 16 / f_{ADCD}$ )。 <b>11<sub>B</sub></b> 1 个仲裁周期包含 20 个仲裁时隙 ( $t_{ARB} = 20 / f_{ADCD}$ )。
<b>0</b>	[14:12]	rw	<b>保留</b> 该位域保留待用，必须设置为 000 <sub>B</sub> 。
<b>ARB<sub>M</sub></b>	15	rw	<b>仲裁模式</b> 该位域定义仲裁器是恒定工作、还是只有当转换请求挂起时才工作。 <b>0<sub>B</sub></b> 仲裁器恒定工作。对于同步从控内核（ <a href="#">章节 18.2.18</a> ）、使用信号 ARBCNT 等间隔采样时（ <a href="#">章节 18.2.19</a> ），必须选择该设置。 <b>1<sub>B</sub></b> 只有当至少有一个转换请求挂起时，仲裁器才工作。若转换器空闲，该设置可确保从发起转换请求到开始转换之间有可重复的延迟。不支持同步转换。

1) 4 个仲裁时隙的缺省设置足以确保正确仲裁。若要求转换请求同步，仲裁周期可被加长。

### 18.2.5.4 全局状态寄存器

全局状态寄存器用于指示当前的转换状态。

#### GLOBSTR

全局状态寄存器

XSFR (12<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CSRC		SYN RUN	ANON		CHNR				0	SAM PLE	BU SY			
r	rh		rh	rh		rh				r	rh	rh			



符号	位序号	类型	功能描述
<b>BUSY</b>	0	rh	<b>模拟部分忙碌</b> 该位指示当前正在进行转换。 0 <sub>B</sub> 转换器空闲。 1 <sub>B</sub> 转换器正在工作。
<b>SAMPLE</b>	1	rh	<b>采样阶段</b> 该位指示当前正在对模拟输入信号采样。 0 <sub>B</sub> 转换器不处于采样阶段。 1 <sub>B</sub> 转换器正处于采样阶段。
<b>CHNR</b>	[7:3]	rh	<b>通道编号</b> 该位域指示当前正在转换哪路模拟输入通道。开始进行新的转换时更新该信息。
<b>ANON</b>	[9:8]	rh	<b>模拟部分开启控制</b> 该位域定义转换器的工作模式。它监控同一个 ADC 内核的位域 <b>GLOBCTR.ANON</b> （主控模式或不具备同步特性），或监控选择用作该内核的同步主控内核的位域 <b>GLOBCTR.ANON</b> （从控模式）。这样确保了只需对同步主控内核的 <b>CLOBCTR</b> 进行单次写操作即可控制所有的同步内核。 00 <sub>B</sub> 转换器关闭、不可能进行转换。仲裁器结束当前一轮仲裁之后保持空闲状态。 01 <sub>B</sub> 转换器开启、可进行转换。使能转换器的自动掉电功能，从而使 ADC 的功耗最低同时具备唤醒功能，但每次转换之前的唤醒时间较长。 10 <sub>B</sub> 转换器开启、可进行转换。使能转换器的自动掉电功能，从而可降低 ADC 的功耗，但每次转换之前的唤醒时间较短。 11 <sub>B</sub> 转换器开启、可进行转换。禁用转换器的自动掉电功能，使 ADC 工作在额定功耗。
<b>SYNRUN</b>	10	rh	<b>同步转换运行指示</b> 该位域指示正在进行同步（并行）转换。 0 <sub>B</sub> 无同步转换（当前无转换或还未请求同步转

符号	位序号	类型	功能描述
			<p>换)。若新的转换请求的优先级更高，当前的转换被取消。</p> <p>1B 正在进行同步转换。一旦开始同步转换，它不能被取消。只有当同步转换结束后，更高优先级的请求才能触发转换。</p>
<b>CSRC</b>	[13:11]	rh	<p><b>当前转换的请求源</b></p> <p>该位指示当前转换（若 <b>BUSY</b> = 1，转换仍在进行）或最后一次转换（若 <b>BUSY</b> = 0，当前无转换）的仲裁时隙编号。每次开始转换时更新该位。</p> <p>000<sub>B</sub> 对仲裁时隙 0 的请求源所请求的通道进行转换。</p> <p>001<sub>B</sub> 对仲裁时隙 1 的请求源所请求的通道进行转换。</p> <p>010<sub>B</sub> 对仲裁时隙 2 的请求源所请求的通道进行转换。</p> <p>其它位组合保留。</p>
<b>0</b>	2, [15:14]	r	<p><b>保留</b></p> <p>读操作返回 0；应写入 0。</p>

#### 18.2.5.5 输入选择寄存器

寄存器 **RSIRx** 用于选择请求源的触发输入和门控输入信号。信号连接和器件的具体实现有关，详见“器件的实现”一节。

## RISRO

请求源 0 输入寄存器                      XSFR (00<sub>H</sub>)                      复位值: 0000<sub>H</sub>

## RISR1

请求源 1 输入寄存器                      XSFR (02<sub>H</sub>)                      复位值: 0000<sub>H</sub>

## RISR2

请求源 2 输入寄存器                      XSFR (04<sub>H</sub>)                      复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
TRI	0	REN	FEN	0	TRSEL			GTI	0	TM EN		0	GTSEL		
rw	r	rw	rw	r	rw			rw	r	rw		r	rw		

符号	位序号	类型	功能描述
<b>GTSEL</b>	[2:0]	rw	<p><b>请求源 x 的 REQGTx 的输入选择</b></p> <p>该位域定义请求源 x 的门控输入信号。</p> <p>000<sub>B</sub> 选择输入信号 REQGTxA。</p> <p>001<sub>B</sub> 选择输入信号 REQGTxB。</p> <p>010<sub>B</sub> 选择输入信号 REQGTxC。</p> <p>011<sub>B</sub> 选择输入信号 REQGTxD。</p> <p>100<sub>B</sub> 选择输入信号 REQGTxE。</p> <p>101<sub>B</sub> 选择输入信号 REQGTxF。</p> <p>110<sub>B</sub> 选择输入信号 REQGTxG。</p> <p>111<sub>B</sub> 选择输入信号 REQGTxH。</p>
<b>TMEN</b>	4	rw	<p><b>请求源 x 定时器模式使能</b></p> <p>该位域使能请求源 x 的定时器模式（用于等间隔采样）。</p> <p>0<sub>B</sub> 禁用定时器模式。可使用标准门控机制。</p> <p>1<sub>B</sub> 使能定时器模式（用于等间隔采样）。必须恒使能标准门控机制（门控信号无影响）。</p>
<b>GTI</b>	7	rh	<p><b>请求源 x 的门控输入</b></p> <p>该标志监控请求源 x 的门控输入信号 REQGTx 的状态。</p> <p>0<sub>B</sub> 门控信号为 0。</p> <p>1<sub>B</sub> 门控信号为 1。</p>
<b>TRSEL</b>	[10:8]	rw	<p><b>请求源 x 的 REQTRx 的输入选择</b></p> <p>该位域定义请求源 x 的触发输入信号。</p> <p>000<sub>B</sub> 选择输入信号 REQTRxA。</p> <p>001<sub>B</sub> 选择输入信号 REQTRxB。</p> <p>010<sub>B</sub> 选择输入信号 REQTRxC。</p> <p>011<sub>B</sub> 选择输入信号 REQTRxD。</p> <p>100<sub>B</sub> 选择输入信号 REQTRxE。</p> <p>101<sub>B</sub> 选择输入信号 REQTRxF。</p> <p>110<sub>B</sub> 选择输入信号 REQTRxG。</p> <p>111<sub>B</sub> 选择输入信号 REQTRxH。</p>

<b>FEN</b>	12	rw	<b>请求源 x 的下降沿使能</b> 该位使能请求源 x 的 REQTRx 的下降沿用作请求触发信号。 0 <sub>B</sub> 禁止下降沿作为请求触发信号。 1 <sub>B</sub> 使能下降沿作为请求触发信号。
<b>REN</b>	13	rw	<b>请求源 x 的上升沿使能</b> 该位使能请求源 x 的 REQTRx 的上升沿用作请求触发信号。 0 <sub>B</sub> 禁止上升沿作为请求触发信号。 1 <sub>B</sub> 使能上升沿作为请求触发信号。
<b>TRI</b>	15	rh	<b>请求源 x 的触发输入</b> 该标志监控请求源 x 的触发输入信号 REQTRx 的状态。 0 <sub>B</sub> 触发信号为 0。 1 <sub>B</sub> 触发信号为 1。
<b>0</b>	3,[6:5], 11,14	r	<b>保留</b> 读操作返回 0；应写入 0。

## 18.2.6 请求源仲裁器

请求源仲裁器评估必须转换哪路模拟输入通道。因此，它规律的轮询请求源、查看是否有转换请求挂起。轮询操作基于长度可编程的“仲裁时隙”完成。若某个仲裁时隙的请求源被禁用或请求源不可用，该时隙被看作为空、不参与仲裁。复位后，所有请求源禁用，必须通过寄存器 **ASENR** 使能仲裁时隙以参与仲裁。

仲裁周期由各可用请求源的仲裁时隙和一个同步时隙组成（见图 18-7）。在每个仲裁周期的结束阶段，仲裁器必须决出有转换请求挂起的、优先级最高的请求源。该仲裁结果作为仲裁胜出方被保存以备后续处理。若在仲裁过程中开始进行转换，该轮仲裁不给出仲裁胜出方。

在 XE16xyM 中，以下请求源可用：

- 仲裁时隙 0 中的请求源 0: **1 级顺序请求源**  
该请求源可产生单路输入通道的转换请求。
- 仲裁时隙 1 中的请求源 1: **16 通道扫描请求源**  
该请求源可最多产生 16 路输入通道的转换请求，转换顺序已设定。
- 仲裁时隙 2 中的请求源 2: **4 级顺序请求源**

该请求源可最多产生 4 路输入通道的转换请求，转换顺序可随意编程。

- 仲裁周期中最后一个仲裁时隙：同步请求源

在该时隙中，仲裁器检查是否有来自其它 ADC 内核的同步请求，它不评估任何内部请求源。同步转换请求在同步从控内核中始终被当作最高优先级进行处理（其它请求源的挂起请求不予考虑）。

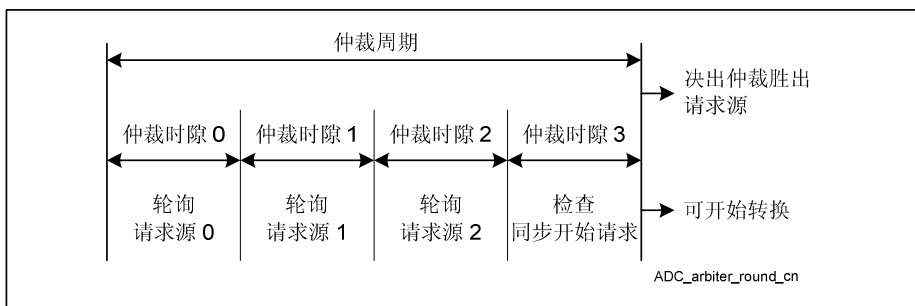


图 18-7 仲裁周期

仲裁周期的计算公式如下：

$$t_{ARB} = N \times (GLOBCTR.DIVD + 1) / f_{ADC}$$

其中  $N = 4、8、16$  或  $20$ ，由 **GLOBCTR.ARBND** 选择

为了使不同器件的仲裁时序相似，组成仲裁周期的仲裁时隙数可编程设定，即使各器件中可用的请求源数目不同。

由于 XE16xyM 中的 ADC 有 3 个请求源，因此每个仲裁周期包含 4 个仲裁时隙已足够。

仲裁周期引入一个时序刻度以检测输入的转换请求信号和启动相关转换的最早时间点。该时序刻度可引入最大一个仲裁周期的抖动。可通过尽量使仲裁周期（仲裁时隙数及其长度）最短的方式降低该抖动。

为了确保从发起转换触发事件（如由定时器单元或外部事件触发）到开始进行相关转换之间有可重复的响应时间（无抖动的固定延迟），主要有以下两种选择。此时，转换器必须空闲，在产生触发事件之前，其它转换请求必须至少一个仲裁周期不挂起：

- 若 **GLOBCTR.ARBND** = 0，仲裁器恒定工作。在该模式下，支持多个 ADC 内核的同步转换。

转换触发信号必须同步于仲裁时序产生。转换触发事件应为仲裁时序刻度的  $n$  倍（ $n = 1、2、3...$ ）。为了实现一定的灵活性，仲裁时隙长度可编程（ $f_{ADC}$  周期的整数倍）。

- 若 **GLOBCTR.ARB** = 1，当不再有转换请求挂起时，一轮仲裁之后仲裁器停止工作。若至少有一个转换请求挂起，仲裁器重新开始工作。转换触发信号不必和仲裁时序同步。在该模式下，不支持同步从控内核的并行转换。

### 18.2.6.1 请求源优先级

为了满足不同应用的需求，每个请求源的优先级可单独编程（见寄存器 **RSRPO**）。当两个或多个请求源同时指示有转换请求挂起时，由优先级决定仲裁器处理这些请求源的顺序。

仲裁器从请求源 0 开始、依次检查各请求源是否有转换请求挂起。仲裁器必须决出有转换请求挂起的、优先级最高的请求源，该请求源即为仲裁胜出方。

### 18.2.6.2 转换启动模式

启动仲裁胜出方所请求的转换时，仲裁器会自动考虑以下情况：

- 若转换器当前空闲（无转换），立即开始执行仲裁胜出方的转换。若当前正在进行转换，则比较仲裁胜出方和当前转换的优先级。

仲裁器通过以下方式处理仲裁胜出方所请求的转换：

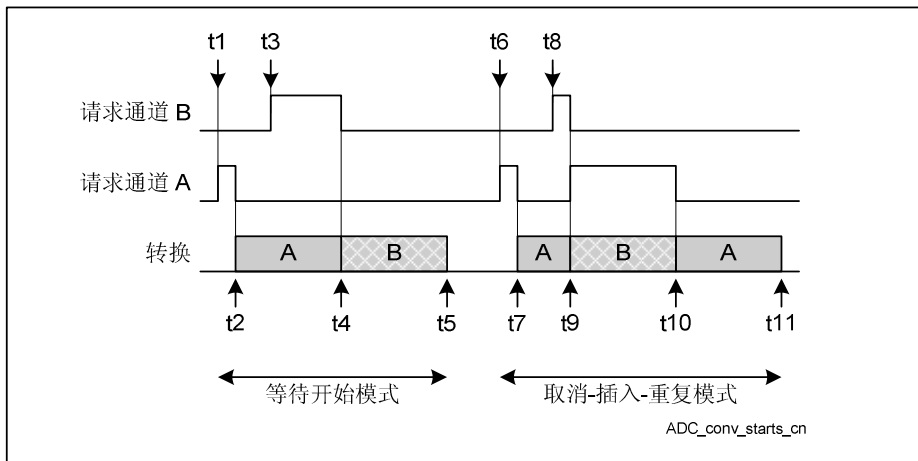
- 若当前转换的优先级相同或较高，先完成当前的转换，然后执行仲裁胜出方请求的转换。
- 若当前转换的优先级较低且仲裁胜出方被设定为**等待开始模式**，先完成当前的转换，然后执行仲裁胜出方请求的转换。

若高优先级转换的时序要求允许一个较大的抖动（**图 18-8** 的  $t_3 - t_4$  之间），可使用该模式。

- 若当前转换的优先级较低且仲裁胜出方被设定为**取消-插入-重复模式**，若检测到优先级较高的新请求，则立即中止当前的转换，除非两个转换请求指向同一个结果寄存器且等待读模式有效（见**章节 18.2.14.2**）。中止当前的转换之后，开始执行仲裁胜出方请求的转换。被中止的转换请求保存在相关请求源中。因此，它会参与下一轮仲裁。

需要注意：中止机制可占用 1-3 个  $f_{ADCI}$  周期，这取决于当前转换的状态。

若高优先级转换的时序要求只允许一个小的抖动（**图 18-8** 的  $t_8 - t_9$  之间），可使用该模式。



**图 18-8 转换启动模式**

每个请求源的转换启动模式可通过寄存器 **RSPRO** 单独设定，所设定的转换启动模式适用于该请求源所请求的所有通道。在两个请求源产生转换请求的情况下，**图 18-8** 给出两种转换启动方式对转换序列的影响。在该示例中，通道 A 比通道 B 的优先级低。

- t1: 发生通道 A 的转换触发事件，激活转换请求。
- t2: 在仲裁周期的结束阶段，决出通道 A 为仲裁胜出方，开始执行通道 A 的转换。一旦开始转换，转换请求 A 被清除。
- t3: 发生通道 B 的转换触发事件，激活转换请求。在等待读取模式下，正常完成通道 A 的转换。
- t4: 通道 A 的转换结束后，开始执行通道 B 的转换。一旦开始转换，转换请求 B 被清除。
- t5: 完成通道 B 的转换。
- t6: 发生通道 A 的转换触发事件，激活转换请求。
- t7: 在仲裁周期的结束阶段，决出通道 A 为仲裁胜出方，开始执行通道 A 的转换。一旦开始转换，转换请求 A 被清除。
- t8: 发生通道 B 的转换触发事件，激活转换请求。
- t9: 在仲裁周期的结束阶段，决出通道 B 为仲裁胜出方。在取消-插入-重复模式下，中止当前通道 A 的转换，开始进行通道 B 的转换。一旦中止转换 A，转换请求 A 被重新置位。一旦开始转换 B，转换请求 B 被清除。
- t10: 完成通道 B 的转换。同时，通道 A 的挂起请求仲裁胜出，开始执行通道 A 的转换。一旦开始转换，转换请求 A 被清除。
- t11: 完成通道 A 的转换。

## 18.2.7 仲裁器寄存器

### 18.2.7.1 仲裁时隙使能寄存器

仲裁时隙使能寄存器用于使能/禁止仲裁时隙中的转换请求处理。

#### ASENR

仲裁时隙使能寄存器

**XSFR (18<sub>H</sub>)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0													AS EN2	AS EN1	AS EN0
r													rw	rw	rw

符号	位序号	类型	功能描述
<b>ASENx</b> ( <b>x = 0-2</b> )	x	rw	<b>仲裁时隙 x 使能</b> 每一位使能仲裁周期中的相关仲裁时隙。对 ASENR 的写操作不会修改请求位。 <b>0<sub>B</sub></b> 相关仲裁时隙被禁用、看作为空。不请求转换、即使请求源的请求位挂起。 若仲裁器不应连续工作 (ARBM = 1)，仲裁时隙 x 的转换请求源的转换请求不必有效。在禁用仲裁时隙之前清除相关请求源的转换请求。 <b>1<sub>B</sub></b> 相关仲裁时隙被使能。若请求源的请求位挂起，则请求转换。
<b>0</b>	[15:3]	r	<b>保留</b> 读操作返回 0；应写入 0。



### 18.2.7.2 请求源优先级寄存器

请求源优先级寄存器用于定义请求源的优先级和转换启动模式。若请求源被使能，切勿软件修改这些设置。若请求源被禁用，它所请求的当前转换结束后，软件可修改这些设置。

#### RSPR0

请求源优先级寄存器 0

XSFR (14<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				CSM 2	0	PRIO2		CSM 1	0	PRIO		CSM 0	0	PRIO0	
r				rw	r	rw		rw	r	rw		rw	r	rw	

符号	位序号	类型	功能描述
<b>PRI00, PRI01, PRI02</b>	[1:0], [5:4], [9:8],	rw	<b>请求源 x 的优先级</b> 该位域定义位于仲裁时隙 x 中的转换请求源 x 的优先级。 00 <sub>B</sub> 选择最低优先级。 ... 11 <sub>B</sub> 选择最高优先级。
<b>CSM0, CSM1, CSM2</b>	3, 7, 11	rw	<b>请求源 x 的转换启动模式</b> 该位域定义位于仲裁时隙 x 中的转换请求源 x 的转换启动模式。 0 <sub>B</sub> 选择等待开始模式。 1 <sub>B</sub> 选择取消-插入-重复模式。
<b>0</b>	2, 6, 10, [15:12]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.8 扫描请求源处理

扫描请求源可最多产生 16 路输入通道的转换请求。每路输入通道可单独编程是否参与扫描序列。转换始终从被使能的最高通道编号开始，从高路通道向低路通道依次转换（转换顺序由通道编号决定，每路通道只能转换一次）。

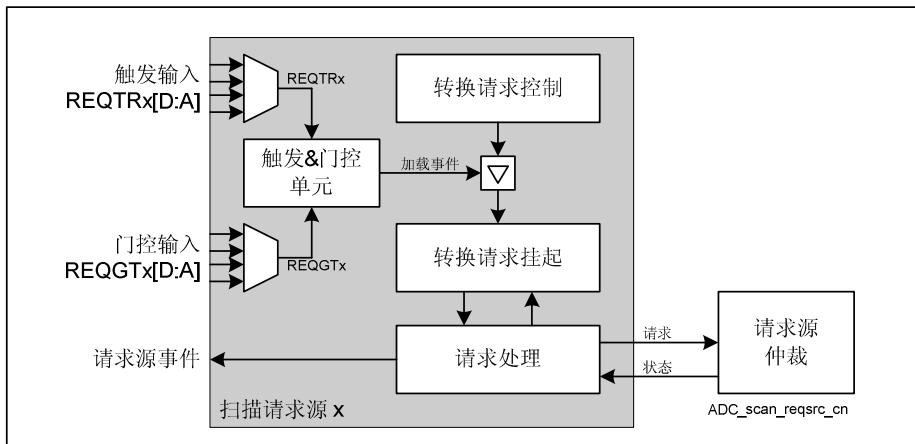


图 18-9 扫描请求源

### 18.2.8.1 概述

扫描请求源执行：

- 转换请求控制：

转换请求控制定义哪些模拟输入通道参与扫描序列（见寄存器 **CRCR1**）。在扫描序列的转换过程中，设定的寄存器值保持不变。

- 转换请求挂起：

转换请求挂起标志指示哪些输入通道请求转换（见寄存器 **CRPR1**）。只有当至少有一位挂起位置位时，才向仲裁器发送转换请求。一旦开始执行转换，相应的挂起位被自动清零。完成扫描序列中的最后一个转换、且所有挂起位被清零后，扫描序列结束、产生请求源事件。

- 转换请求的处理：

转换请求处理模块和请求源仲裁器接口。它根据请求挂起位的状态请求转换并处理转换状态信息。若其它请求源发送的转换请求的优先级较高，扫描请求源触发的转换将被中止。该机制可确保被中止的转换参与下一轮仲裁、不会丢失。通过寄存器 **CRM1** 控制扫描序列。

- 转换触发和门控信号的处理：

转换触发和门控单元和 ADC 模块外围的、可请求转换的信号和模块接口。例如，定时器单元可发送请求信号使转换和 PWM 事件同步。根据请求控制位的内容修改请求挂起位来加载事件启动扫描序列转换。

### 18.2.8.2 扫描序列的操作

**操作扫描请求源**时，应执行以下操作：

- 必须编程寄存器 **CRCR1** 定义哪些通道参与扫描序列。
- 若希望通过外部信号进行触发和门控控制，必须通过寄存器 **RSIR1** 中的位域 **TRSEL** 和 **GTSEL** 定义门控和触发输入。该寄存器还用于选择触发事件的跳变沿。
- 必须由 **CRM1.ENG**T 定义门控机制。
- 对应的仲裁时隙必须被使能以接受来自扫描请求源的转换请求（见寄存器 **ASENR**）。
- 必须通过 **CRM1** 定义加载事件以启动扫描序列。
- 若 **CRM1.LDM** = 0，发生加载事件时，**CRCR1** 的内容被复制到挂起 **CRPR1** 中（覆盖）。若执行扫描转换的过程中发生加载事件，该设置可启动新的扫描序列、“忽略”其余挂起位。
- 若 **CRM1.LDM** = 1，发生加载事件时，**CRCR1** 的内容和 **CRPR1** 按位逻辑或、结果存放在 **CRPR1** 中（不覆盖）。若执行扫描转换的过程中发生加载事件，该设置可启动新的扫描序列、不“忽略”其余挂起位。

**启动扫描序列**时，支持以下机制产生加载事件：

- 可由外部模块或信号（如定时器单元或某个输入引脚）硬件控制启动扫描序列。触发特性由 **CRM1.ENTR** = 1 使能。若检测到触发输入上发生期望的跳变，产生加载事件，由寄存器 **RSIRx** 进行跳变沿选择。
- 可设置 **CRM1.LDEV** = 1 软件控制产生加载事件。该机制可启动扫描序列而不修改寄存器 **CRCR1** 的内容。对 **CRCR1** 的写操作不会导致加载事件（首先准备通道控制，然后启动扫描序列）。
- 若对寄存器 **CRPR1** 写操作，写入的数据将保存在寄存器 **CRCR1** 中、自动产生加载事件。该机制利用写入的数据启动扫描序列（可通过一次写操作定义并启动扫描序列，如在 **PEC** 控制下）。
- 若 **CRM1.SCAN** = 1，每次扫描序列结束、产生请求源事件时，产生加载事件。该设置会使扫描序列无限制的重复执行。

**终止或中止扫描序列**时，支持以下机制：

- 可通过外部门控信号在任意时刻终止或继续扫描序列，该信号可来自定时器单元或某个输入引脚。可通过 **CRM1.ENG**T 使能门控功能并选择门控信号的极性。门控机制不修改转换挂起位的值、只是阻止请求处理模块向仲裁器发送转换请求。
- 可通过清除 **ASENR.ASENx** 软件禁用该仲裁时隙。该机制不修改转换挂起位的值、只是阻止仲裁器接受来自请求处理模块的转换请求。

- 可通过设置 **CRMRI.CLRPND = 1** 清除挂起请求位。建议在清除挂起位之前终止扫描序列。

### 18.2.8.3 请求源事件和中断

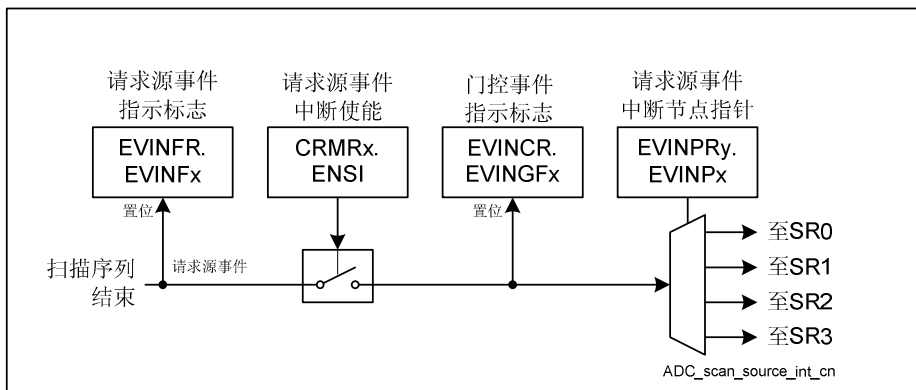
完成扫描序列中的最后一个转换后（所有挂起位 = 0），产生请求源事件。根据图 18-10 所示的中断结构，可基于请求源事件产生相应的请求源事件中断。若检测到请求源事件，置位寄存器 **EVINFR** 中相应的指示标志。对 **EVINFR** 中的相应位写 1 时，也可置位相应的标志位，写 0 不起作用。此外，门控事件标志 **EVINCR.EVINGFx** 指示已激活请求源中断。也可通过向 **EVINCR** 写 1 的方式软件清除相应的标志位。

一旦检测到请求源事件（并由 **CRMRx.ENS** 使能）或寄存器 **EVINFR** 中的相关位被置 1（该写操作触发请求源事件），由寄存器 **EVINPR0** 中的请求源事件中断节点指针来选择服务请求输出 **SRx**。

此外，若服务请求输出被激活，门控事件标志 **EVINCR.EVINGFx** 置位。

请求源事件和结果事件共用同一个寄存器。请求源事件位于寄存器 **EVINFR** 中：

- 事件 1（由位 **EVINF0** 指示）：  
位于仲裁时隙 1 的扫描请求源请求源事件。



**图 18-10 扫描请求源的中断产生**

## 18.2.9 扫描请求源寄存器

### 18.2.9.1 转换请求控制寄存器

转换请求控制寄存器用于存放扫描请求源的控制和状态位。**CRCR1** 中的数字 1 代表参与仲裁的请求源所位于的仲裁时隙的编号。

发生加载事件时，转换请求控制寄存器的内容被复制到挂起寄存器中。可通过写访问寄存器 **CRPR1** 间接更新 **CRCR1**、也可直接访问（读和写）**CRCR1**。

## CRCR1

**转换请求 1 控制寄存器**

**XSFR (E8<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

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

符号	位序号	类型	功能描述
<b>CHx</b> (x = 0-15)	x	rwh	<b>通道位 x</b> 每一位对应一路模拟通道，由寄存器中各位的位置 x 定义通道编号 CHx。 0 <sub>B</sub> 该扫描请求源未请求转换模拟通道 CHx。 1 <sub>B</sub> 该扫描请求源将请求转换模拟通道 CHx。

### 18.2.9.2 转换请求挂起寄存器

转换请求挂起寄存器用于请求进行相应模拟通道的转换。

对 **CRPR1** 的写操作导致数据写入 **CRCR1** 并自动产生加载事件。

对 **CRPR1** 的读操作给出挂起位。

## CRPR1

**转换请求 1 挂起寄存器**

**XSFR (EA<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

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

符号	位序号	类型	功能描述
<b>CHPx</b> (x = 0-15)	x	rwh	<b>通道挂起位 x</b> <u>写操作:</u> 对该地址的写访问将指向寄存器 <b>CRCR1</b> 。

符号	位序号	类型	功能描述
			<p><u>读操作:</u></p> <p>每一位对应一路模拟通道, 由寄存器中各位的位置 x 定义通道编号 CHx。</p> <p>0<sub>B</sub> 该请求源未请求转换模拟通道 CHx。</p> <p>1<sub>B</sub> 该请求源请求转换模拟通道 CHx。</p>

### 18.2.9.3 转换请求模式寄存器

转换请求模式寄存器用于配置扫描请求源的工作模式。

#### CRMR1

##### 转换请求 1 模式寄存器

**XSFR (EC<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						LD EV	CLR PND	REQ GT	0	LDM	SC AN	EN SI	EN TR	ENGT	
r						w	w	rh	r	rw	rw	rw	rw	rw	

符号	位序号	类型	功能描述
<b>ENGT</b>	[1:0]	rw	<p><b>使能门控</b></p> <p>该位使能请求源的门控功能。</p> <p>00<sub>B</sub> 请求源不发送转换请求。</p> <p>01<sub>B</sub> 当至少有一位挂起位置位时, 请求源发送转换请求。</p> <p>10<sub>B</sub> 当至少有一位挂起位置位、且选定的门控信号 REQGT<sub>x</sub> = 1 时, 请求源发送转换请求。</p> <p>11<sub>B</sub> 当至少有一位挂起位置位、且选定的门控信号 REQGT<sub>x</sub> = 0 时, 请求源发送转换请求。</p>
<b>ENTR</b>	2	rw	<p><b>使能外部触发</b></p> <p>该位使能外部触发功能。若该功能被使能, 当检测到触发输入信号 REQTR 发生 (选定的) 跳变时, 产生加载事件。</p> <p>0<sub>B</sub> 禁止外部触发事件。</p> <p>1<sub>B</sub> 使能外部触发事件。</p>

符号	位序号	类型	功能描述
<b>ENSI</b>	3	rw	<b>使能请求源中断</b> 若发生请求源事件（已完成最后一个挂起的转换），该位使能产生请求源中断。 0 <sub>B</sub> 禁止产生请求源中断。 1 <sub>B</sub> 使能产生请求源中断。
<b>SCAN</b>	4	rw	<b>自动扫描使能</b> 该位使能自动扫描功能。若该功能被使能，发生请求源事件时自动产生加载事件。 0 <sub>B</sub> 禁止自动扫描功能。 1 <sub>B</sub> 使能自动扫描功能。
<b>LDM</b>	5	rw	<b>加载事件模式</b> 该位定义由加载事件触发的传送机制。 0 <sub>B</sub> 发生加载事件时，寄存器 CCRx 的内容被复制到挂起寄存器 CRPRx 中（覆盖）。 1 <sub>B</sub> 发生加载事件时，寄存器 CCRx 值和挂起寄存器 CRPRx 按位逻辑或。
<b>REQGT</b>	7	rh	<b>请求门控电平</b> 该位监控输入 REQGT 的电平。 0 <sub>B</sub> 电平为 0。 1 <sub>B</sub> 电平为 1。
<b>CLRPND</b>	8	w	<b>清除挂起位</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除寄存器 CRPRx 中各挂起位。
<b>LDEV</b>	9	w	<b>产生加载事件</b> 0 无操作。 1 产生加载事件。
<b>0</b>	6, [15:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.10 顺序请求源处理

顺序请求源可产生几路输入通道的转换请求，通道编号可随意编程（和转换顺序固定的扫描请求源不同）。每个 ADC 内核中有两种顺序请求源可用：

- 仲裁时隙 2 中的请求源：

该请求源可最多处理 4 路输入通道的转换序列（4 个入口的 4 级队列）。该机制可支持特定应用的转换序列，特别是同一通道包含多次转换、对时序要求很严格的转换序列。

- 仲裁时隙 0 中的请求源：

该请求源可处理单路输入通道的转换（1 个入口的 1 级队列）。可通过该机制产生软件控制的、或硬件触发的转换请求（可将单次转换“插入”正在执行的转换序列中）。

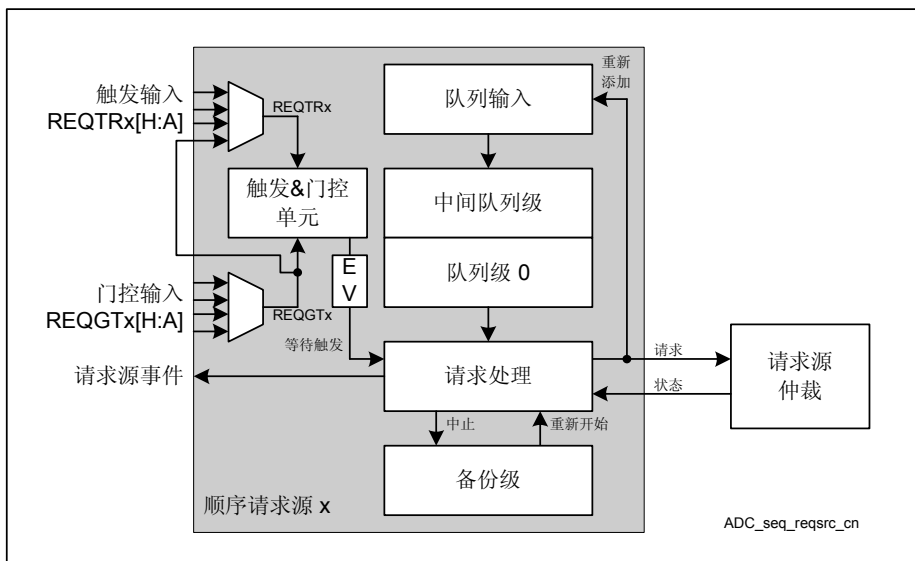


图 18-11 顺序请求源

两种顺序请求源的内部结构以及请求源的处理方式均相似。设定的转换序列保存在队列缓存中（基于 FIFO 机制），该缓存至少包含一级队列（级 0）和一级备份（用于存放被中止的转换）。两种顺序请求源的唯一差别在于用来保存序列的中间队列级的个数不同。仲裁时隙 0 中的请求源不提供中间队列级（只带有队列级 0 的 1 级队列）；仲裁时隙 2 中的请求源提供 3 级中间队列（因此共为 4 级队列）。



### 18.2.10.1 概述

顺序请求源执行：

- 队列输入：

队列输入代表定义序列的编程接口（见寄存器 **QINR0**、**QINR2**）。它不提供缓存功能，而是通过向队列缓存（队列级 0+可选的中间队列级）写数据对其进行填充处理。队列中各级的内容不能由软件直接修改，除非通过命令清除整个队列的内容。

队列输入还支持重新添加机制，可将已经启动的转换（来自队列级 0）当作新的输入自动重新插入到队列中（包括控制参数）。该特性可（通过软件）一次建立起转换时序，之后重复同一转换序列、无需每次重新设定。若序列中的所有队列入口均设置为重新添加模式，该转换序列可自动重复执行。

- 队列级 0：

该级的内容定义下一次请求转换的通道编号（见寄存器 **Q0R0**、**Q0R2**）。它还定义是否由外部事件触发请求、或尽快执行所请求的转换。它还使能在转换之后产生请求源中断。

开始执行转换并可处理下一个队列入口（若可用）时，清除队列级 0 的内容。

- 队列备份级：

若顺序请求源所请求的转换被中止，队列备份级用于保存相关的请求控制参数。有效位指示在处理队列级 0 请求的转换之前，先请求执行被中止的转换，从而保持原始的请求序列（见寄存器 **QBUR0**、**QBUR2**）。

- 转换请求的处理：

转换请求处理模块和请求源仲裁器接口。它根据队列级 0 中的有效信息请求转换并处理转换状态信息。通过寄存器 **QMR0**（对应仲裁时隙 0 的请求源）和 **QMR2**（对应仲裁时隙 2 的请求源）控制队列序列。

- 转换触发和门控信号的处理：

转换触发和门控单元和 ADC 模块外围的、可请求转换的信号和模块接口。例如，定时器单元可发送请求信号使转换和 PWM 事件同步。触发事件可启动队列级 0 的转换请求（见寄存器 **QMR0**、**QMR2**）。事件标志位 **QSRx.EV** 指示已检测到一个触发事件（若 **QMRx.ENTR** 使能外部触发，触发输入信号 **REQTRx** 发生期望的跳变时，硬件产生触发事件；或 **QMRx.TREV = 1**，软件产生触发事件）。一旦开始执行转换，或设置 **CEV = 1**、**FLUSH = 1** 或 **CLRV = 1**，该标志位被清零。

### 18.2.10.2 顺序请求源的操作

**操作顺序请求源**时，应执行以下操作：

- 必须通过设置队列输入 **QINR0**（对应仲裁时隙 0 的请求源）和 **QINR2**（对应仲裁时隙 2 的请求源）初始化转换序列（当使用重新添加机制）。每一次写操作对应一个转换请求。

在使能请求源之前，应先完成整个期望序列的初始化，这是因为一旦使能重新添加特性，不允许软件对 **QINRx** 进行写操作。

- 若希望通过外部信号进行触发和门控控制，必须通过寄存器 **RSIRO**（对应仲裁时隙 0）或 **RSIR2**（对应仲裁时隙 2）中的位域 **TRSEL** 和 **GTSEL** 定义门控和触发输入。这些寄存器还用于选择触发事件的跳变沿。
- 必须由 **QMRx.ENG** 定义门控机制。
- 若期望采用外部触发机制，必须设置 **QMRx.ENTR = 1**。
- 必须使能相应的仲裁时隙以接受顺序请求源的转换请求（见寄存器 **ASENR**）。

**启动顺序请求源的转换序列**时，支持以下机制：

- 可由外部模块或信号（如定时器单元或某个输入引脚）硬件控制启动队列序列的转换。触发特性由 **QMRx.ENTR = 1** 使能。若检测到触发输入上发生期望的跳变，产生触发事件。
- 可通过设置 **QMRx.TREV = 1** 软件控制产生触发事件。若队列 0 寄存器（或队列备份寄存器）包含有效数据，该机制可启动转换请求。
- 对队列输入进行一次写操作，将会产生一个（新的）有效的队列入口。若队列已空（无有效入口），写入的数据进入队列级 0 并启动转换请求（若由 **QMRx.ENG** 使能且不需等待外部触发）。若采用重新添加机制，队列运行过程中切勿对队列输入进行写操作。若队列已满，写操作被忽略。

**终止或中止顺序请求源的转换序列**时，支持以下机制：

- 可通过外部门控信号在任意时刻终止或继续转换序列，该信号可来自定时器单元或某个输入引脚。可通过 **QMRx.ENG** 使能门控功能并选择门控信号的极性。门控机制不修改队列入口、只是阻止请求处理模块向仲裁器发送转换请求。
- 可通过清除 **ASENR.ASENx** 软件禁用相应的仲裁时隙。该机制不修改队列入口、只是阻止仲裁器接受来自请求处理模块的转换请求。
- 可通过设置 **QMRx.CLRV = 1** 清除下一个挂起的队列入口。建议在清除队列入口之前终止转换序列（**ENG** = 00<sub>B</sub>）。若队列备份寄存器中包含有效入口，则清除该入口，否则清除队列寄存器 0 中的有效入口。
- 可通过设置 **QMRx.FLUSH = 1** 清除所有队列入口。建议在清除队列入口之前终止转换序列。

### 18.2.10.3 请求源事件和中断

完成顺序请求源所请求的一次转换之后，产生请求源事件。中断使能位存放在队列 0 寄存器（若这不是被中止后重新启动的转换）或队列备份寄存器（若这是被中止后重新启动的转换）中。

根据图 18-12 所示的中断结构，可基于请求源事件产生相应的请求源事件中断。若检测到请求源事件，置位寄存器 **EVINFR** 中相应的指示标志。对 **EVINFR** 中的相应位写 1 时，也可置位相应的标志位，写 0 不起作用。也可通过向 **EVINCR** 写 1 的方式软件清除相应的标志位。

一旦检测到请求源事件（并由 **Q0Rx. ENSI** 或 **QBURx. ENSI** 使能）或寄存器 **EVINFR** 中的相关位被置 1（该写操作触发请求源事件），由寄存器 **EVINPR0** 中的请求源事件中断节点指针选择激活哪路服务请求输出 **SRx**。

此外，若服务请求输出被激活，门控事件标志 **EVINCR.EVINGFx** 置位。

请求源事件和结果事件共用同一个寄存器。请求源事件位于寄存器 **EVINFR** 中：

- 事件 0：位于仲裁时隙 0 的顺序请求源的请求源事件。
- 事件 2：位于仲裁时隙 2 的顺序请求源的请求源事件。

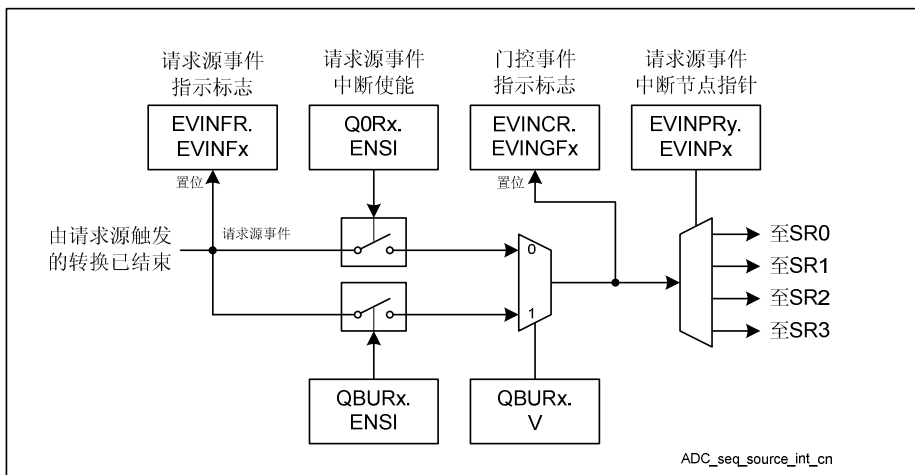


图 18-12 顺序请求源的中断产生

## 18.2.11 顺序请求源寄存器

### 18.2.11.1 队列模式寄存器

这些寄存器用于存放顺序请求源的控制位。

寄存器 QMR0 和 QMR2 中的数字 0 和 2 代表参与仲裁的请求源所位于的仲裁时隙的编号。

*注：通过 QMR.CLRV 或 QMR.FLUSH 软件修改队列内容之前，必须先完成所有和该队列相关的硬件操作。因此，必须禁用相应的仲裁时隙，软件必须至少等待两个仲裁周期（从而确保该请求源不再会是仲裁获胜方）。随后检查 **GLOBSTR.CRSC** 和 **GLOBSTR.BUSY** 以确保不再执行由该请求源触发的转换。随后软件可读取 QBURx 和 Q0Rx 并可开始修改队列内容。*

#### QMR0

队列 0 模式寄存器

XSFR (E0<sub>H</sub>)

复位值: 0000<sub>H</sub>

#### QMR2

队列 2 模式寄存器

XSFR (F0<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				CEV	FLU SH	TR EV	CLR V	0					EN TR	ENGT	
r				w	w	w	w	r					rw	rw	

符号	位序号	类型	功能描述
ENGT	[1:0]	rw	<b>使能门控</b> 该位使能请求源的门控功能。 00 <sub>B</sub> 请求源不发送转换请求。 01 <sub>B</sub> 若队列 0 寄存器或备份寄存器中有一个有效的转换请求挂起，请求源发送转换请求。 10 <sub>B</sub> 若队列 0 寄存器或备份寄存器中有一个有效的转换请求挂起、且选定的门控信号 REQGTx = 1，请求源发送转换请求。 11 <sub>B</sub> 若队列 0 寄存器或备份寄存器中有一个有效的转换请求挂起、且选定的门控信号 REQGTx = 0，请求源发送转换请求。
ENTR	2	rw	<b>使能外部触发</b>

符号	位序号	类型	功能描述
			<p>该位使能外部触发功能。</p> <p>0<sub>B</sub> 禁止外部触发，不产生触发事件。</p> <p>1<sub>B</sub> 使能外部触发，若检测到触发输入 REQTRx 上发生期望的跳变，产生触发事件。</p>
<b>CLRV</b>	8	w	<p><b>V 位清零</b></p> <p>0<sub>B</sub> 无操作。</p> <p>1<sub>B</sub> 序列中下一个挂起的有效队列入口和事件标志 EV 被清零。若队列备份寄存器中有一个有效入口（QBUR.V = 1），该入口被清零；否则，队列寄存器 0 中的入口被清零。</p>
<b>TREV</b>	9	w	<p><b>触发事件控制</b></p> <p>0<sub>B</sub> 无操作。</p> <p>1<sub>B</sub> 软件产生触发事件。若请求源中的有效入口等到触发事件，启动转换请求。</p>
<b>FLUSH</b>	10	w	<p><b>队列全部清零</b></p> <p>0<sub>B</sub> 无操作。</p> <p>1<sub>B</sub> 队列中的所有入口（包括备份部分）和事件标志 EV 被清零。队列中不再包含有效入口。</p>
<b>CEV</b>	11	w	<p><b>事件标志清零</b></p> <p>0<sub>B</sub> 无操作。</p> <p>1<sub>B</sub> 位 EV 被清零。</p>
<b>0</b>	[7:3], [15:12]	r	<p><b>保留</b></p> <p>读操作返回 0；应写入 0。</p>

### 18.2.11.2 队列状态寄存器

队列状态寄存器用于指示顺序请求源的状态。填充级和队列已空信息适用于队列中间级（若可用）和队列寄存器 0。保存在备份寄存器中的被中止的转换不由这些标志位指示（因此，见 QBURx.V）。

#### QSR0

队列 0 状态寄存器

XSFR (E2<sub>H</sub>)

复位值: 0020<sub>H</sub>

#### QSR2

队列 2 状态寄存器

XSFR (F2<sub>H</sub>)

复位值: 0020<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							EV	REQGT	0	EMPTY	0			FILL	
r							rh	rh	r	rh	r			rh	

符号	位序号	类型	功能描述
FILL	[1:0]	rh	<b>填充级<sup>1)</sup></b> 该位域指示在顺序请求源中有多少个有效的队列入口。每次有新入口写入 QINRx、或采用重新添加机制时，该位域加 1；每开始进行一次转换，该位域减 1。若填充级达到其最大值，则忽略新入口。 00 <sub>B</sub> EMPTY = 1: 队列中无有效入口。 EMPTY = 0: 队列中有 1 个有效入口。 01 <sub>B</sub> 队列中有 2 个有效入口。 10 <sub>B</sub> 队列中有 3 个有效入口。 11 <sub>B</sub> 队列中有 4 个有效入口。
EMPTY	5	rh	<b>队列已空</b> 该位指示顺序请求源是否包含有效入口。 0 <sub>B</sub> 队列中有 FILL+1 个有效入口。 1 <sub>B</sub> 队列中无有效入口（队列已空）。
REQGT	7	rh	<b>请求门控电平</b> 该位监控输入 REQGT 的电平。 0 <sub>B</sub> 电平为 0。 1 <sub>B</sub> 电平为 1。

符号	位序号	类型	功能描述
<b>EV</b>	8	rh	<b>事件检测</b> 当队列（队列寄存器 0 或备份寄存器）中至少有一个有效入口时，该位指示已检测到触发事件。一旦置位，转换开始后该位被自动清零。 0 <sub>B</sub> 未检测到触发事件。 1 <sub>B</sub> 已检测到触发事件。
<b>0</b>	[4:2], 6, [15:9]	r	<b>保留</b> 读操作返回 0；应写入 0。

1) 对于仲裁时隙 0 中的 1 级队列，该位域始终为 00<sub>B</sub>。

### 18.2.11.3 队列 0 寄存器

队列 0 寄存器用于监控挂起请求的状态。

#### Q0R0

**队列 0 寄存器 0**

**XSFR (E4<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

#### Q0R2

**队列 2 寄存器 0**

**XSFR (F4<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							V	EXTR	ENSI	RF	REQCHNR				
r							rh	rh	rh	rh	rh				

符号	位序号	类型	功能描述
<b>REQCHNR</b>	[4:0]	rh	<b>请求转换通道编号</b> 该位域指示请求转换的通道编号。
<b>RF</b>	5	rh	<b>重新添加</b> 该位指示当挂起请求被处理后（转换开始）被丢弃还是被自动重新添加到请求队列中。 0 <sub>B</sub> 开始转换之后丢弃请求。 1 <sub>B</sub> 开始转换之后将请求重新添加到队列中。

符号	位序号	类型	功能描述
<b>ENSI</b>	6	rh	<b>使能请求源中断</b> 该位指示转换结束后是否产生请求源事件中断。 0 <sub>B</sub> 禁止产生请求源事件中断。 1 <sub>B</sub> 使能产生请求源事件中断。
<b>EXTR</b>	7	rh	<b>外部触发</b> 该位指示有效的队列入口是否立刻产生转换请求、或请求处理器是否需要等待触发事件。 0 <sub>B</sub> 请求处理器不需等待触发事件。 1 <sub>B</sub> 请求处理器需等待触发事件。
<b>V</b>	8	rh	<b>请求通道编号有效</b> 该位指示队列寄存器 0 是否包含有效的队列入口。 0 <sub>B</sub> 队列入口无效，不产生转换请求。 1 <sub>B</sub> 队列入口有效，产生转换请求。
<b>0</b>	[15:9]	r	<b>保留</b> 读操作返回 0；应写入 0。

#### 18.2.11.4 队列备份寄存器

队列备份寄存器监控被中止的顺序请求源的状态。

寄存器 QBURx 和 QINRx 共用一个寄存器地址。对该寄存器地址的读操作将给出寄存器 QBURx 的“rh”位；对该寄存器地址的写操作将指向寄存器 QINRx 的“w”位。

##### QBUR0

队列 0 备份寄存器

**XSFR (E6<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

##### QBUR2

队列 2 备份寄存器

**XSFR (F6<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>							<b>V</b>	<b>EXTR</b>	<b>ENSI</b>	<b>RF</b>	<b>REQCHNR</b>				
r							rh	rh	rh	rh	rh				



符号	位序号	类型	功能描述
<b>REQCHNR</b>	[4:0]	rh	<b>请求转换通道编号</b> 该位域存放由顺序请求源请求的、被中止的转换的通道编号。
<b>RF</b>	5	rh	<b>重新添加</b> 该位存放由顺序请求源请求的、被中止的转换的重新添加位。
<b>ENSI</b>	6	rh	<b>使能请求源中断</b> 该位存放由顺序请求源请求的、被中止的转换的请求源事件中中断使能位。
<b>EXTR</b>	7	rh	<b>外部触发</b> 该位存放由顺序请求源请求的、被中止的转换的外部触发位。
<b>V</b>	8	rh	<b>请求通道编号有效</b> 该位指示队列备份寄存器中的入口是否有效（REQCHNR、RF、ENSI 和 EXTR 是否有效）。当顺序请求源所请求的、正在进行的转换被中止时，位 V 被置位。重新开始转换后该位被清零。 <b>0<sub>B</sub></b> 备份寄存器中不含有效数据。 <b>1<sub>B</sub></b> 备份寄存器中包含一个有效入口。在请求转换队列寄存器 0 中的有效入口之前，先请求执行被中止的转换。
<b>0</b>	[15:9]	r	<b>保留</b> 读操作返回 0；应写入 0。

#### 18.2.11.5 队列输入寄存器

队列输入寄存器是顺序请求源的入口寄存器。

寄存器 QBURx 和 QINRx 共用一个寄存器地址。对该寄存器地址的读操作将给出寄存器 QBURx 的“rh”位；对该寄存器地址的写操作将指向寄存器 QINRx 的“w”位。

## QINR0

### 队列 0 输入寄存器

**XSFR (E6<sub>H</sub>)**

复位值: 0000<sub>H</sub>

## QINR2

### 队列 2 输入寄存器

**XSFR (F6<sub>H</sub>)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								EXTR	ENSI	RF	REQCHNR				
r								w	w	w	w				

符号	位序号	类型	功能描述
<b>REQCHNR</b>	[4:0]	rh	<b>请求转换通道编号</b> 该位域定义请求转换的通道编号。
<b>RF</b>	5	rh	<b>重新添加</b> 该位定义该队列入口的重新添加功能。 <b>0<sub>B</sub></b> 开始执行转换之后，该队列入口的内容不被重新添加到 <b>QINR<sub>x</sub></b> 中。 <b>1<sub>B</sub></b> 开始执行转换之后，该队列入口的内容被重新添加到 <b>QINR<sub>x</sub></b> 中。
<b>ENSI</b>	6	rh	<b>使能请求源中断</b> 该位定义请求源事件中断功能。 <b>0<sub>B</sub></b> 转换结束后不产生请求源事件中断。 <b>1<sub>B</sub></b> 转换结束后产生请求源事件中断。
<b>EXTR</b>	7	rh	<b>外部触发</b> 该位定义外部触发功能。 <b>0<sub>B</sub></b> 有效的队列入口立刻产生转换请求。 <b>1<sub>B</sub></b> 有效的队列入口需等到发生触发事件之后才产生转换请求。

符号	位序号	类型	功能描述
0	[15:8]	r	保留 读操作返回 0；应写入 0。

### 18.2.12 通道相关功能

通道控制单元定义每路模拟输入通道的转换设置，每路通道 CHx 有一个对应的通道控制寄存器 CHCTR<sub>x</sub>（见[章节 18.2.13.1](#)）。仲裁器决出将要转换的通道之后，使用所设定的转换设置进行转换。转换设置包含以下内容：

- **转换参数：**  
位域 ICLSEL 定义转换所使用的输入组（见[章节 18.2.12.1](#)）。
- **参考选择：**  
位域 REFSEL 定义转换所使用的参考输入（见[章节 18.2.12.2](#)）。
- **通道事件处理：**  
位域 LCC、BNDASEL 和 BNDBSEL 定义使用哪个边界进行极限检查（见[章节 18.2.12.4](#)）、哪个通道事件导致通道事件中断（见[章节 18.2.12.5](#)）。
- **同步转换请求：**  
位 SYNC 定义通道是否触发同步转换（见[章节 18.2.18](#)）。

除此之外，ADC 内核还支持一种机制（称为替换特性，见[章节 18.2.12.3](#)），可将转换请求重指向其它通道。

#### 18.2.12.1 输入组

输入组定义采样时间和转换精度。在大多数应用中，多个模拟输入信号的输入电路特性（信号源的 RC 输入低通滤波器和阻抗）非常相似，因而这些通道在采样阶段的时序相似。具有相似参数的输入通道可组成一个输入组。

具有相同 ICLSEL 设置的所有通道属于同一个输入组、采样时间和转换精度相同。在 XE16xyM 中，共有两个输入组。可单独编程每个输入组的相关寄存器 INPCRx（ $x = 0 - 1$ ），根据应用需求调整采样时间和转换精度。这些寄存器的缺省设置对应 2 个  $f_{\text{ADCI}}$  周期的最短采样时间和 10 位转换精度。若该缺省设置满足应用的要求，则不必修改位域 CHCTR<sub>x</sub>.ICLSEL 和寄存器 INPCRx（ $x = 0 - 1$ ）。

#### 18.2.12.2 参考选择

ADC 的转换结果始终参照一个参考电压。若模拟输入电压和参考电压相等，得到最大数字转换结果。为了支持多个测量范围，用户可选用标准参考输入 V<sub>AREF</sub>，也可选用每个 ADC 内核的模拟输入通道 CH0 上的备选参考输入。每路输入通道的参考选择可单独设定。

可利用该特性将基于 5V 和 3.3V 的传感器同时连接到一个 ADC 内核上。在这种情况下，一组输入通道使用标准参考输入；另一组使用 CH0 上的备选参考输入。

需要注意：n 位转换精度的最小刻度  $1 \text{ LSB}_n$  和所选的参考电压有关。若使用低参考电压，刻度会变得非常小，从而（由于噪声影响）导致 TUE 增加。因此，建议避免使用小参考电压。

### 18.2.12.3 替换特性

ADC 内核提供替换特性，允许将通道 CH0 或 CH1 的转换请求重指向其它通道。可利用该特性测量一路输入通道并将转换结果保存在两个不同的结果寄存器中。

- 可将同一个信号测量两次、无需读取转换结果以避免数据丢失，这样可快速触发两次连续的转换并和 CPU 中断延迟无关。
- 传感器信号仅和一路输入通道相连（不需和两路输入相连）。这样节省了低成本应用的输入引脚、在计算误差时只需考虑一个输入的漏电。
- 即使模拟输入 CH0 用作备选参考输入（见图 18-13），可使用通道 CH0 的内部触发和数据处理特性。
- 两次转换的通道设置可以不同（边界值、中断等）
- 若已设立顺序转换请求源，通道 CH0 或 CH1 的转换请求可轻易重指向其它输入通道、不需清除整个队列。

在典型的低成本 AC 驱动应用中，只使用一个电流传感器来决定相电流。根据所施加的 PWM 序列，测量值的含义不同、采样点必须位于 PWM 信号周期内。在图 18-13 的示例中，采样信号和一路输入通道（CHx）相连，但会触发两路通道（CHx 和 CH0）的两次转换。利用替换特性，通道 CH0 的转换请求不触发转换 CH0、而是触发转换 CHx，但仍使用 CH0 的通道设置。尽管测量一路模拟输入（CHx），转换结果可存入不同的寄存器 RESRx 和 RESR0 并从中读取。此外，可选择不同的中断或极限边界。

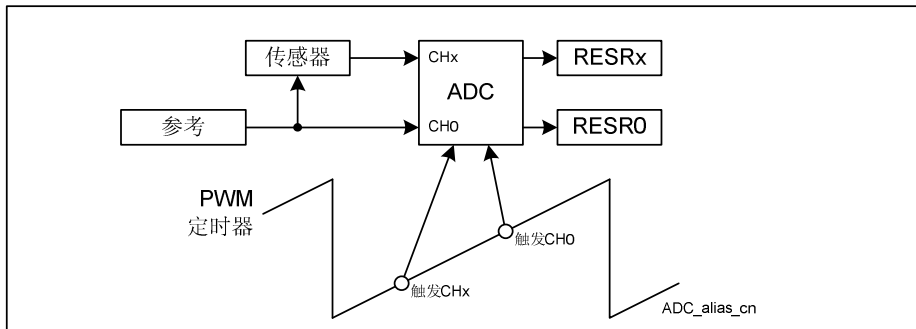


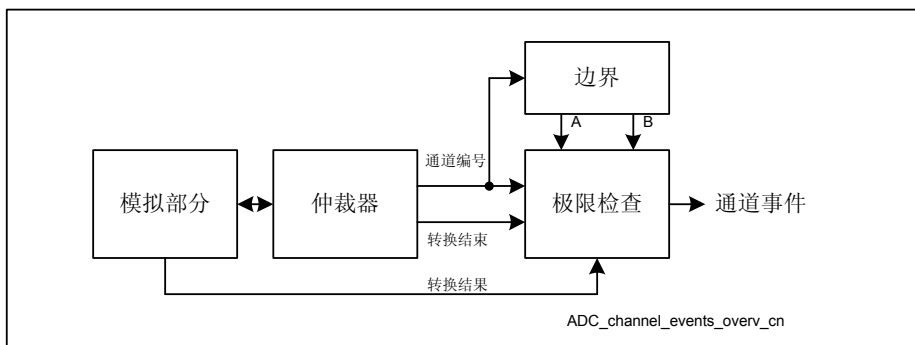
图 18-13 替换特性

#### 18.2.12.4 极限检查

极限检查单元自动将每个转换结果和两个边界值（边界 A 和 B）进行比较。对于每一路通道，有 4 个可编程边界值（**LCBR0 - LCBR3**）供用户选择。

利用该结构，转换结果的范围被划分为 3 个区域：

- 区 I：转换结果小于或等于两个边界值。
- 区 II：转换结果大于一个边界值、小于或等于另一个边界值。
- 区 III：转换结果大于两个边界值。



**图 18-14 通道事件的产生**

由通道控制寄存器中的位域 LCC 定义产生触发事件的条件：

- LCC = 000<sub>B</sub>：不产生通道事件。
- LCC = 001<sub>B</sub>：转换结果不在区 I 时产生通道事件。
- LCC = 010<sub>B</sub>：转换结果不在区 II 时产生通道事件。
- LCC = 011<sub>B</sub>：转换结果不在区 III 时产生通道事件。
- LCC = 100<sub>B</sub>：始终产生通道事件（和边界无关）。
- LCC = 101<sub>B</sub>：转换结果在区 I 时产生通道事件。
- LCC = 110<sub>B</sub>：转换结果在区 II 时产生通道事件。
- LCC = 111<sub>B</sub>：转换结果在区 III 时产生通道事件。

在图 18-15 的极限检查示例中，只有当转换结果不在区 II 定义的正常工作范围内时（LCC = 010<sub>B</sub>），才产生通道事件。

极限检查的典型应用有温度监控和过电流感测。只要测量的温度值低于边界值，则不必通知 CPU。在这种情况下，只有当转换结果在区 III 时（LCC = 111<sub>B</sub>）才产生通道事件，指示已超出正常的温度范围。若自动扫描序列（基于规律的时间间隔触发其转

换)中包含对某模拟温度输入信号的转换,则在检测到超出正常温度之前,无须 CPU 监控温度 (CPU 负载为零)。

在过流保护的情况下,可使用通道事件禁止产生 PWM 以降低电流 (在 XE16xyM 中,ADC 的中断输出和 CCU6x 单元的输入相连,从而可快速响应中断而无需 CPU 干预)。

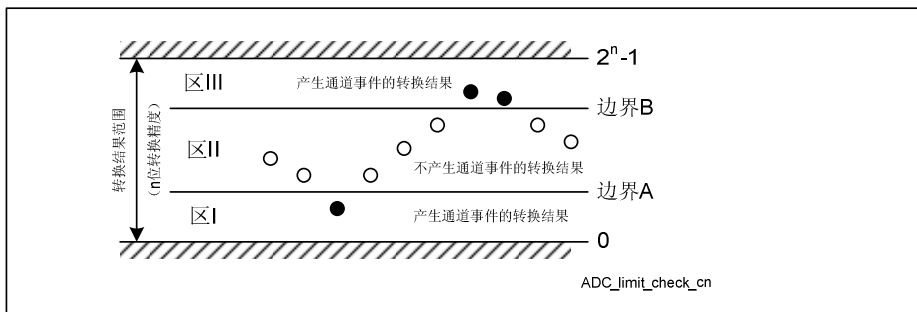
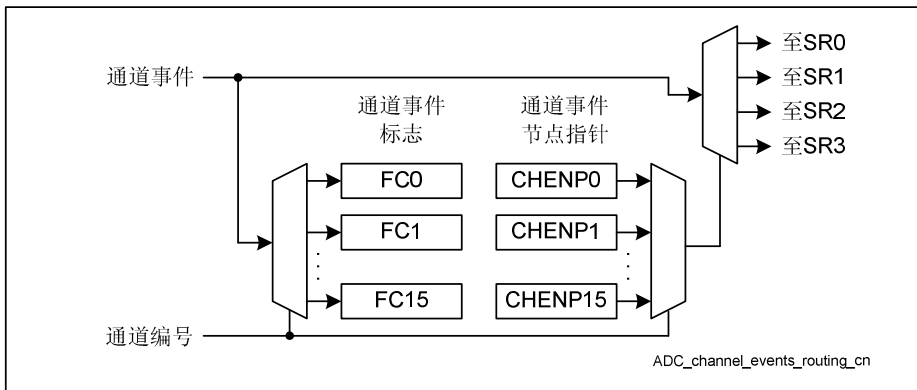


图 18-15 极限检查

注: 边界 A 和 B 也可选择相同的边界寄存器。在这种情况下,转换结果的范围被划分为两个区域 (区 II 为空)。

### 18.2.12.5 通道事件中断

根据图 18-16 所示的中断结构,可基于通道事件产生相应的通道事件中断。若检测到通道事件,置位寄存器 CHINFR 中相应的指示标志。对 CHINFR 中的相应位写 1 时,也可置位相应的标志位,写 0 不起作用。也可通过向 CHINCR 写 1 的方式软件清除相应的标志位。



**图 18-16 通道事件中中断的产生**

一旦检测到通道事件或寄存器 **CHINFR** 中的相关位被置 1，由寄存器 **CHINPRO**、**CHINPR4**、**CHINPR8** 或 **CHINPR12** 中的通道事件中中断节点指针选择激活哪路服务请求输出 SRx。

### 18.2.13 通道相关寄存器

#### 18.2.13.1 通道控制寄存器

通道控制寄存器用于选择目标结果寄存器、控制极限检查机制并选择输入组。

通道控制寄存器 0 定义输入通道 CH0 的设置，依此类推。

#### CHCTR<sub>x</sub> (x = 0-15)

通道 **x** 控制寄存器

**XSFR (20<sub>H</sub>+x\*2)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	RESRSEL	ICLSEL	REFSEL	SYNC	LCC	BNDSEL	BNDASEL								
r	rw	rw	rw	rw	rw	rw	rw								

符号	位序号	类型	功能描述
<b>BNDASEL</b>	[1:0]	rw	<b>边界 A 选择</b> 该位域定义哪个边界将作为极限检查的边界 A。 00 <sub>B</sub> 选用寄存器 LCBR0 给出的值。

符号	位序号	类型	功能描述
			01 <sub>B</sub> 选用寄存器 LCBR1 给出的值。 10 <sub>B</sub> 选用寄存器 LCBR2 给出的值。 11 <sub>B</sub> 选用寄存器 LCBR3 给出的值。
<b>BNDBSEL</b>	[3:2]	rw	<b>边界 B 选择</b> 该位域定义哪个边界将作为极限检查的边界 B。 00 <sub>B</sub> 选用寄存器 LCBR0 给出的值。 01 <sub>B</sub> 选用寄存器 LCBR1 给出的值。 10 <sub>B</sub> 选用寄存器 LCBR2 给出的值。 11 <sub>B</sub> 选用寄存器 LCBR3 给出的值。
<b>LCC</b>	[6:4]	rw	<b>极限检查控制</b> 该位域定义极限检查机制的操作，该位域的编码见 <a href="#">页 18-51 的章节 18.2.12.4</a> 。
<b>SYNC</b>	7	rw	<b>同步请求</b> 该位定义该通道的转换请求是否会引发其它 ADC 内核的同步（并行）转换。只有当 ADC 内核为潜在转换主控方时（ <b>SYNCTR.STSEL</b> = 00），该位才有效。 0 <sub>B</sub> 该通道不请求同步转换。 1 <sub>B</sub> 若 ADC 内核为潜在转换主控方，该通道请求同步转换。
<b>REFSEL</b>	[9:8]	rw	<b>参考输入选择</b> 该位域定义该通道的参考输入源。 00 <sub>B</sub> 选择标准参考输入 V <sub>AREF</sub> 。 01 <sub>B</sub> 选择备选参考输入 CH0。 10 <sub>B</sub> 保留不用。 11 <sub>B</sub> 保留不用。
<b>ICLSEL</b>	[11:10]	rw	<b>输入组选择</b> 该位域选择输入组。 00 <sub>B</sub> 选择输入组 0。 01 <sub>B</sub> 选择输入组 1。 10 <sub>B</sub> 保留不用。



符号	位序号	类型	功能描述
			11 <sub>B</sub> 保留不用。
<b>RESRSEL</b>	[14:12]	rw	<b>结果寄存器选择</b> 该位域定义哪个结果寄存器将作为该通道转换结果的目标寄存器。 000 <sub>B</sub> 选择结果寄存器 0。 001 <sub>B</sub> 选择结果寄存器 1。 ... 111 <sub>B</sub> 选择结果寄存器 7。
<b>0</b>	15	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.13.2 输入组寄存器

输入组寄存器用于控制采样时间和转换精度。

输入组寄存器 0 定义输入组 0 的设置，依此类推。

#### INPCR<sub>x</sub> (x = 0-1)

输入组寄存器 x						XSFR (C0 <sub>H</sub> +x*2)						复位值: 0000 <sub>H</sub>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						DW		STC							
r						rw		rw							

符号	位序号	类型	功能描述
<b>STC</b>	[7:0]	rw	<b>采样时间控制</b> 该位域定义附加的采样时间，以模拟时钟周期 f <sub>ADCl</sub> 为单位。通过设置该位域可得到最小两个模拟时钟周期的采样时间。 采样时间 = (2+ STC) / f <sub>ADCl</sub>
<b>DW</b>	[9:8]	rw	<b>数据宽度</b> 该位定义转换精度 <sup>1)</sup> 。具有不同 DW 设置的转换结果的 MSB 在结果寄存器中左对齐。未转换的位为 0。

符号	位序号	类型	功能描述
			00 <sub>B</sub> 转换结果 10 位宽。 01 <sub>B</sub> 保留不用。 10 <sub>B</sub> 转换结果 8 位宽。 11 <sub>B</sub> 保留不用。
<b>0</b>	[15:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

1) 00<sub>B</sub> 为缺省设置。采用 10 位 AD 转换时，01<sub>B</sub> 和 11<sub>B</sub> 被转换器忽略、当作 00<sub>B</sub> 处理。

### 18.2.13.3 极限检查边界寄存器

这些寄存器用于定义极限检查单元的比较值（边界）。边界的复位值分别为整个转换结果范围（12 位表示）的 10%、90%、33%和 66%。

#### LCBR0

**极限检查边界寄存器 0**                      **XSFR (84<sub>H</sub>)**                      **复位值: 0198<sub>H</sub>**

#### LCBR1

**极限检查边界寄存器 1**                      **XSFR (86<sub>H</sub>)**                      **复位值: 0E64<sub>H</sub>**

#### LCBR2

**极限检查边界寄存器 2**                      **XSFR (88<sub>H</sub>)**                      **复位值: 0554<sub>H</sub>**

#### LCBR3

**极限检查边界寄存器 3**                      **XSFR (8A<sub>H</sub>)**                      **复位值: 0AA8<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>				<b>BOUNDARY</b>										<b>0</b>	
r				rw										r	

符号	位序号	类型	功能描述
<b>BOUNDARY</b>	[11:2]	rw	<b>极限检查的边界</b> 该位域定义极限检查单元中用于和实际转换结果进行比较的值。极限检查的结果用来产生通道中断，见 <a href="#">章节 18.2.12.4</a> 。
<b>0</b>	[1:0], [15:12]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.13.4 通道事件指示标志寄存器

通道事件指示标志寄存器 CHINFR 用于监控通道事件。

#### CHINFR

通道事件指示标志寄存器

XSFR (90<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHIN F15	CHIN F14	CHIN F13	CHIN F12	CHIN F11	CHIN F10	CHIN F9	CHIN F8	CHIN F7	CHIN F6	CHIN F5	CHIN F4	CHIN F3	CHIN F2	CHIN F1	CHIN F0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位序号	类型	功能描述
<b>CHINF<sub>x</sub></b> ( <b>x = 0-15</b> )	x	rwh	<b>通道 x 事件指示标志</b> 标志 CHINF <sub>x</sub> 指示已检测到通道 x 的通道事件。 写 0 不起作用，写 1 将置位相应的标志位并产生中断请求。 0 <sub>B</sub> 还未检测到通道 x 事件。 1 <sub>B</sub> 已检测到通道 x 事件。

### 18.2.13.5 通道事件指示标志清零寄存器

向通道事件指示标志清零寄存器 CHINCR 中的某位写 1，可清除寄存器 **CHINF<sub>R</sub>** 中相应的通道事件指示标志 CHINF<sub>x</sub>。若检测到通道事件时相应位被置 1，标志 CHINF<sub>x</sub> 被清零。

#### CHINCR

通道事件指示标志清零寄存器

XSFR (92<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
CHIN C15	CHIN C14	CHIN C13	CHIN C12	CHIN C11	CHIN C10	CHIN C9	CHIN C8	CHIN C7	CHIN C6	CHIN C5	CHIN C4	CHIN C3	CHIN C2	CHIN C1	CHIN C0
w	w	w	w	w	w	w	w	w	w	w	w	w	w	w	w

符号	位序号	类型	功能描述
<b>CHINC<sub>x</sub></b> ( <b>x = 0-15</b> )	x	w	<b>清除通道事件指示标志</b> 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除标志 CHINFR.x。

### 18.2.13.6 通道中断节点指针寄存器

这些寄存器用于定义产生通道事件中断时激活哪路服务请求输出 ADCx\_SR[3:0]。

#### CHINPR0

**通道中断节点指针寄存器 0**

**XSFR (98<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CHINP3	0	CHINP2	0	CHINP1	0	CHINP0								
r	rw	r	rw	r	rw	r	rw					r			rw

符号	位序号	类型	功能描述
<b>CHINP0, CHINP1, CHINP2, CHINP3</b>	[1:0], [5:4], [9:8], [13:12]	rw	<b>通道 x 的中断节点指针</b> 该位域选择由哪条服务请求输出线指示已产生通道 x 的通道事件中断。 00 <sub>B</sub> 选择服务请求输出 SR0。 01 <sub>B</sub> 选择服务请求输出 SR1。 10 <sub>B</sub> 选择服务请求输出 SR2。 11 <sub>B</sub> 选择服务请求输出 SR3。
<b>0</b>	[3:2], [7:6], [11:10], [15:14]	r	<b>保留</b> 读操作返回 0; 应写入 0。

#### CHINPR4

**通道中断节点指针寄存器 4**

**XSFR (9A<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CHINP7	0	CHINP6	0	CHINP5	0	CHINP4								
r	rw	r	rw	r	rw	r	rw					r			rw

符号	位序号	类型	功能描述
<b>CHINP4, CHINP5, CHINP6, CHINP7</b>	[1:0], [5:4], [9:8], [13:12]	rw	<b>通道 x 的中断节点指针</b> 该位域选择由哪条服务请求输出线指示已产生通道 x 的通道事件中断。

符号	位序号	类型	功能描述
			00 <sub>B</sub> 选择服务请求输出 SR0。 01 <sub>B</sub> 选择服务请求输出 SR1。 10 <sub>B</sub> 选择服务请求输出 SR2。 11 <sub>B</sub> 选择服务请求输出 SR3。
<b>0</b>	[3:2], [7:6], [11:10], [15:14]	r	<b>保留</b> 读操作返回 0；应写入 0。

### CHINPR8

**通道中断节点指针寄存器 8**

**XSFR (9C<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>CHINP11</b>	<b>0</b>	<b>CHINP10</b>	<b>0</b>	<b>CHINP9</b>	<b>0</b>	<b>CHINP8</b>								
r	rW	r	rW	r	rW	r	rW	r	rW	r	rW	r	rW	r	rW

符号	位序号	类型	功能描述
<b>CHINP8, CHINP9, CHINP10, CHINP11</b>	[1:0], [5:4], [9:8], [13:12]	rW	<b>通道 x 的中断节点指针</b> 该位域选择由哪条服务请求输出线指示已产生通道 x 的通道事件中断。 00 <sub>B</sub> 选择服务请求输出 SR0。 01 <sub>B</sub> 选择服务请求输出 SR1。 10 <sub>B</sub> 选择服务请求输出 SR2。 11 <sub>B</sub> 选择服务请求输出 SR3。
<b>0</b>	[3:2], [7:6], [11:10], [15:14]	r	<b>保留</b> 读操作返回 0；应写入 0。

## CHINPR12

通道中断节点指针寄存器 12

XSFR (9E<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	CHINP15	0	CHINP14	0	CHINP13	0	CHINP12								
r	rw	r	rw	r	rw	r	rw	r	rw	r	rw	r	rw	r	rw

符号	位序号	类型	功能描述
CHINP12, CHINP13, CHINP14, CHINP15	[1:0], [5:4], [9:8], [13:12]	rw	<b>通道 x 的中断节点指针</b> 该位域选择由哪条服务请求输出线指示已产生通道 x 的通道事件中断。 00 <sub>B</sub> 选择服务请求输出 SR0。 01 <sub>B</sub> 选择服务请求输出 SR1。 10 <sub>B</sub> 选择服务请求输出 SR2。 11 <sub>B</sub> 选择服务请求输出 SR3。
0	[3:2], [7:6], [11:10], [15:14]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.13.7 替换寄存器

替换寄存器把 CH0 和 CH1 请求转换的通道编号修改为其它通道编号，具体描述见 [章节 18.2.12.3](#)。所设定的替换通道编号将代替内部请求转换的通道编号。仍使用内部请求转换的实际通道编号来处理所有其它内部操作及同步请求。

## ALR0

替换寄存器 0

XSFR (1C<sub>H</sub>)

复位值: 0100<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	ALIAS1	0	ALIAS0												
r	rw	r	rw												

符号	位序号	类型	功能描述
ALIAS0	[4:0]	rw	<b>CH0 转换请求的替换值</b> 对该位域定义的通道进行转换以替代通道 CH0。

符号	位序号	类型	功能描述
			转换操作仍使用 CH0 的通道设置。
<b>ALIAS1</b>	[12:8]	rw	<b>CH1 转换请求的替换值</b> 对该位域定义的通道进行转换以替代通道 CH1。 转换操作仍使用 CH1 的通道设置。
<b>0</b>	[7:5], [15:13]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.14 转换结果的处理

转换结果的处理包括：

- 转换结果的保存（见[章节 18.2.14.1](#)）
- 等待读取模式（见[章节 18.2.14.2](#)）
- 结果事件中断（见[章节 18.2.14.3](#)）
- 结果 FIFO 缓存（见[章节 18.2.14.4](#)）
- 数据压缩或抗混叠滤波（见[章节 18.2.14.5](#)）

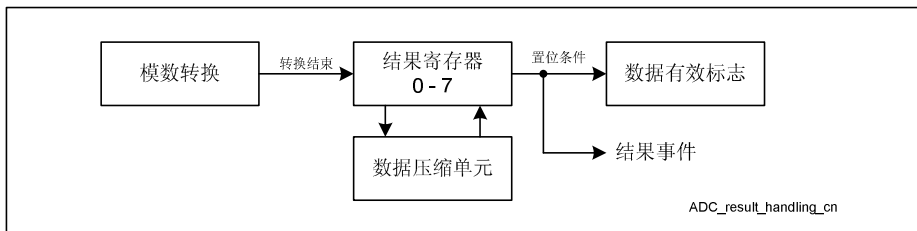


图 18-17 转换结果的处理

#### 18.2.14.1 转换结果的保存

每一路模拟输入通道的通道控制寄存器 **CHCTR<sub>x</sub>** (**x = 0-15**) 中包含一个指针位域 (RESRSEL)，它选择使用哪个结果寄存器存放该通道的转换结果。用户可利用该结构将不同通道的转换结果存放在一个或多个结果寄存器中。可根据应用需求（数据压缩、自动扫描、替换特性、结果 FIFO 等）灵活分配转换结果，从而使 CPU 负载最小、或允许更长的中断延迟。

每个结果寄存器中的数据有效标志 **VFR.VF<sub>x</sub>** 指示“新”有效数据已保存到相应的结果寄存器中、可被读取。

由于转换结果的处理机制不同，因此可采用不同的方式来表示转换结果：

- **数据压缩滤波器禁用：**

转换结果最大 10 位宽，转换结果的 MSB 始终在位 11，其余 LSB 填 0。

一旦有新转换结果保存到结果寄存器中，数据有效标志置位、发生结果事件。

多路模拟输入通道可共用一个结果寄存器。

- **数据压缩滤波器使能：**

转换结果最大 10 位宽，转换结果的 MSB 始终在位 11，其余 LSB 填 0。附加位[13:12]存放数据累加的 MSB。

一旦数据压缩序列结束、最终结果保存到结果寄存器中，数据有效标志置位、发生结果事件。

相应结果寄存器中不含通道编号信息。

为了支持等待读取模式和 FIFO 缓存功能，软件执行读访问、或转换结果传送到另一个 FIFO 单元（若结果 FIFO 缓存被使能）时，有效标志必须被自动清零。

该操作和调试要求相互矛盾。进行调试时，必须在正常程序中插入读写命令（如用于监控转换结果）。若调试器读出结果寄存器的内容，它会把转换结果的状态从有效 = “新”（还未读取）修改到“旧”（已读取），这将对应用产生不期望的影响。

因此，引入带“V”的读取寄存器，它的值和不带“V”的寄存器的值相同，不过不清除有效标志。此时，调试器使用带“V”的读取寄存器，它可监控转换结果而不影响其状态。

根据实际应用的需求（转换结果使能或禁用数据压缩滤波、支持调试器访问），结果寄存器可支持四种不同的读取方式。这些读取方式得到的结果寄存器的内容相同，但不同读取地址对应的操作不同：

- 标准读取方式 **RESR<sub>x</sub> (x = 0-7)**：

数据压缩滤波器必须被禁用，寄存器中包含通道编号信息，指示哪路通道已被转换，读取该寄存器时清除相应的有效标志。转换结果的这种表示方式和 XC16x 器件中的 ADC 结果寄存器相兼容。

- 读取方式 **RESRA<sub>x</sub> (x = 0-7)**：

数据压缩滤波器可被使能，寄存器中不含通道编号信息，读取该寄存器时清除相应的有效标志。

- 支持调试的读取方式 **RESRV<sub>x</sub> (x = 0-7)**：

数据压缩滤波器必须被禁用，寄存器中包含通道编号信息，但读取该寄存器时不清除相应的有效标志。

- 支持调试的读取方式 **RESRAV<sub>x</sub> (x = 0-7)**：



数据压缩滤波器可被使能，寄存器中不含通道编号信息，但读取该寄存器时不清除相应的有效标志。

#### 18.2.14.2 等待读取模式

结果寄存器的等待读取模式可确保 CPU（或 PEC）能够独立处理每次的转换结果而没有丢失数据的危险。若结果寄存器中的转换结果还未被 CPU 读出就被新转换结果覆盖，会产生数据丢失。

特别是对于自动扫描转换序列（或时序要求不太严格的其它序列），等待读取模式可基于某事件（硬件或软件）请求一个转换序列，但只有在 CPU 读取前一次的转换结果之后才启动新的转换。

若置位寄存器 RCRx (x = 0-7) 中的 WFR 使能结果寄存器的等待读取模式，当目标结果寄存器中包含有效数据（由有效标志 VFx = 1 指示），或当前执行的转换指向同一个结果寄存器时，请求源不产生转换请求。

只有当目标结果寄存器被读取之后才产生新的转换请求。

若两个请求源指向同一个（具有等待读取特性的）结果寄存器，当低优先级请求源在高优先级请求源之前请求转换，它不会被高优先级请求所中断。若高优先级请求源指向不同的结果寄存器，低优先级的转换可先被取消、以后再重新执行。

#### 18.2.14.3 结果事件中断

根据图 18-18 所示的中断结构，可基于结果事件产生相应的结果事件中断。若检测到结果事件，置位寄存器 EVINFR 中相应的指示标志。对 EVINFR 中的相应位写 1 时，也可置位相应的标志位，写 0 不起作用。也可通过向 EVINCR 写 1 的方式软件清除相应的标志位。

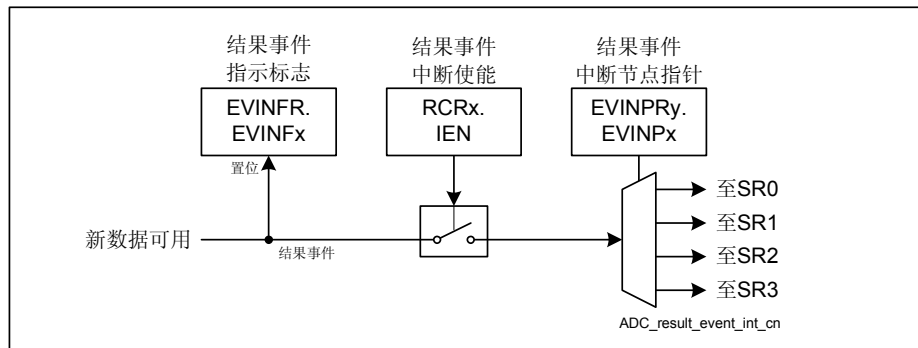


图 18-18 结果事件中断的产生

一旦检测到结果事件或寄存器 **EVINFR** 中的相关位被置 1，由寄存器 **EVINPR8** 或 **EVINPR12** 中的结果事件中断节点指针选择激活哪路服务请求输出 **SRx**。

结果事件和请求源事件共用同一个寄存器。结果事件位于寄存器 **EVINFR** 中以下各位：

- 事件 8：结果寄存器 0 的结果事件。
- 事件 9：结果寄存器 1 的结果事件。
- ...
- 事件 15：结果寄存器 7 的结果事件。

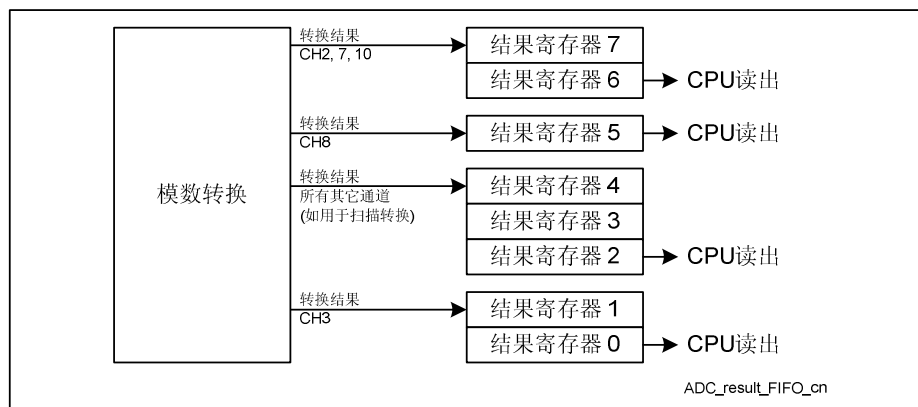
#### 18.2.14.4 结果 FIFO 缓存

若结果寄存器不直接用于保存转换结果，它可和同一个 ADC 内核中的其它结果寄存器级联构成结果 FIFO 缓存（先入先出缓存机制）。这样可实现以“宽松的”CPU 访问时序保存测量结果以备将来读取。可建立多个 FIFO 缓存结构。

FIFO 缓存至少由两个相邻的结果寄存器  $x$  和  $z = x+1$  构成，其中结果寄存器  $z$  代表 FIFO 缓存的输入；结果寄存器  $x$  代表输出。必须将转换结果送至 FIFO 缓存的输入、必须从 FIFO 缓存的输出读取数据。

置位寄存器 **RCRx ( $x = 0-7$ )** 中的位 **FEN** 使能 FIFO 缓存功能，**RCRz** 除外。

在图 18-19 的示例中，结果寄存器被配置为两个二级 FIFO 缓存（分别为结果寄存器 0/1 和 6/7）、一个三级 FIFO 缓存（结果寄存器 2/3/4），结果寄存器 5 用作“正常”结果寄存器（无 FIFO 缓存）。



**图 18-19 结果 FIFO 缓存**

若多个相邻的结果寄存器级联构成 FIFO 缓存（从结果寄存器  $z$  到  $x$ ,  $z > x$ ），最高编号的寄存器（ $z$ ）始终是输入；最低编号的寄存器（ $x$ ）始终是输出。所有中间结果寄存器  $y$ （ $x < y < z$ ）用作 FIFO 的中间级、无数据输入或输出功能。

每个 FIFO 缓存的结果寄存器特性如下：

- 结果寄存器  $z$ （**FIFO 缓存输入**,  $FEN = 0$ ）：

该结果寄存器可使得能用于数据压缩。若 FIFO 已满，支持等待读取模式以避免数据丢失。不支持产生结果事件中断。切勿以修改有效位的方式读取该寄存器。

- 结果寄存器  $y$ （**FIFO 缓存中间级**,  $FEN = 1$ ）：

这些结果寄存器不支持等待读取模式、也不能用于数据压缩。不支持产生结果事件中断。切勿以修改有效位的方式读取这些寄存器，这些寄存器也不能用作转换结果的目标寄存器。

- 结果寄存器  $x$ （**FIFO 缓存输出**,  $FEN = 0$ ）：

该结果寄存器支持产生结果事件中断，通知 CPU 可从该寄存器读取新数据。不支持等待读取模式和数据压缩，必须被禁止。该寄存器切勿用作转换结果的目标寄存器。

若允许产生中断，FIFO 中的每个数据字均可产生一个结果中断。

#### 18.2.14.5 数据压缩滤波

数据压缩滤波可最多累加 4 次的转换结果，用于抗混迭滤波或结果平均。

每个结果寄存器可单独使能数据压缩功能，该特性由 **RCR $x$  ( $x = 0-7$ )** 中的位域 **DRCTR** 控制，其实际状态由同一个寄存器中的位域 **DRC**（数据压缩计数器）给出。

指向其它结果寄存器的转换不影响结果寄存器  $x$  的数据压缩滤波。因此，在指向结果寄存器  $x$  的两次转换之间，可以执行其它通道的转换。

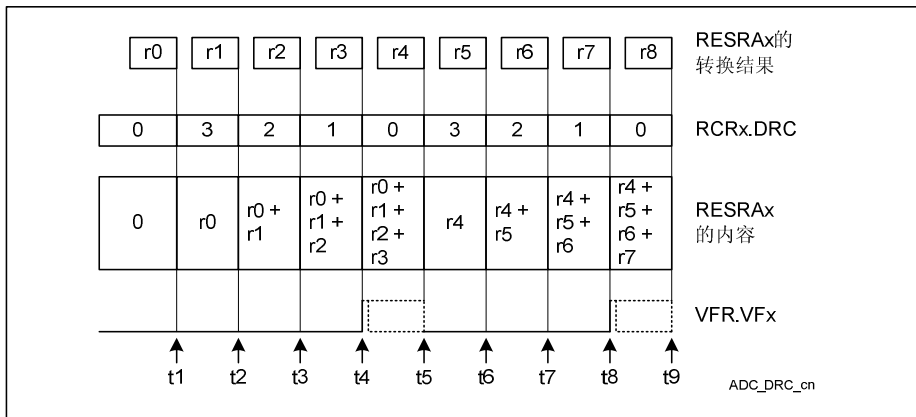


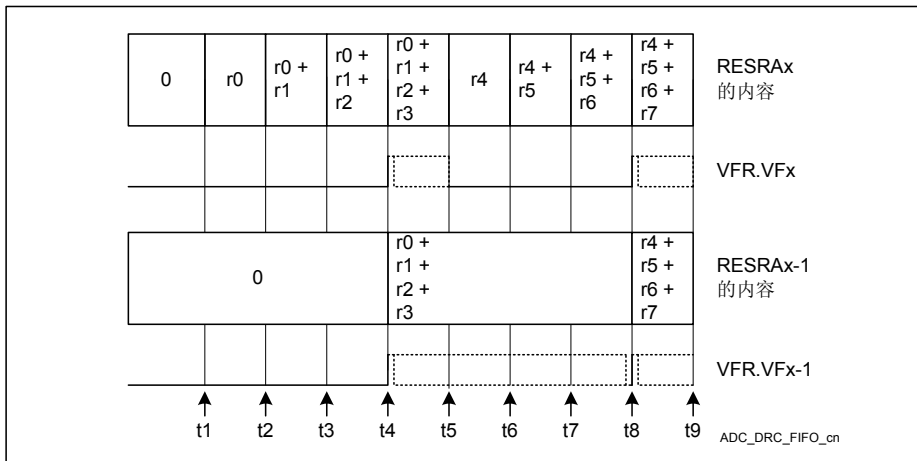
图 18-20 数据压缩滤波

在图 18-20 的示例中，数据压缩累加 4 次的转换结果。数据压缩基于三个原则：

- 每次当位域 DRC 为 0 且指向结果寄存器 x 的转换结束时（t1、t5、t9），位域 RCRx.DRC 的值载入位域 DRC，转换结果保存到结果寄存器 x 中（即开始累加转换结果）。
- 每次当位域 DRC 非 0 且指向结果寄存器 x 的转换结束时（t2、t3、t4 对应第一个最终结果，t6、t7、t8 对应下一个最终结果），位域 DRC 减 1，转换结果累加到结果寄存器 x 中。
- 每次当位域 DRC 递减到 0、或由 RCRx.DRC = 0 加载 DRC（t4 对应第一个最终结果，t8 对应下一个最终结果），结果寄存器 x 的有效标志置位，产生结果寄存器事件。

在开始执行新的数据压缩序列之前，必须从结果寄存器 x 中读出已完成的数据压缩的最终结果（t4 和 t5 之间，或 t8 和 t9 之间）。从该寄存器读出最终结果之后，有效标志被自动清零。

若该时间间隔太短，建议采用结果 FIFO 缓存机制，将结果寄存器 z 和结果寄存器 x 相关联，如图 18-21 所示（ $z = x + 1$ ）。在这种情况下，数据压缩结束后，将结果寄存器 z 处理过的最终结果载入结果寄存器 x。在完成新的数据压缩序列之前，必须从结果寄存器 x 中读出上一次的最终结果（t4 和 t8 之间）。



**图 18-21 带有结果 FIFO 的数据压缩滤波**

可通过软件置位寄存器 **VFR** 中的位 **x** 清除 **RCRx (x = 0-7)** .DRC。

## 18.2.15 转换结果相关寄存器

### 18.2.15.1 标准读取 RESRx 和 RESRVx

这些结果寄存器给出转换结果和相关通道编号。

读取寄存器 **RESRx** 时，相应的有效标志被清零；读取 **RESRVx** 时，有效标志保持不变。

#### RESRx (x = 0-7)

结果寄存器 **x**                      **XSFR (40H+2\*x)**                      复位值: 0000H

#### RESRVx (x = 0-7)

结果寄存器 **x**，读取 **V**                      **XSFR (60H+2\*x)**                      复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CHNR</b>				<b>RESULT</b>											<b>0</b>
rh				rh											r

符号	位序号	类型	功能描述
<b>RESULT</b>	[11:2]	rh	<b>转换结果</b> 该位域存放转换结果。
<b>CHNR</b>	[15:12]	rh	<b>通道编号</b> 该位域存放最新更新寄存器的通道编号。若使能外部复用器控制，由复用器设置 EMUX[2:0]代替位 CHNR[3:1]。
<b>0</b>	[1:0]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.15.2 数据压缩读取 RESRAx 和 RESRAVx

这些结果寄存器给出累加的转换结果，它们不存放相关的通道编号信息。

读取寄存器 RESRAx 时，相应的有效标志被清零；读取 RESRAVx 时，有效标志保持不变。

#### RESRAx (x = 0-7)

结果寄存器 x，读取 A                      XSFR (50<sub>H</sub>+2\*x)                      复位值: 0000<sub>H</sub>

#### RESRAVx (x = 0-7)

结果寄存器 x，读取 AV                      XSFR (70<sub>H</sub>+2\*x)                      复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>		<b>RESULT</b>													<b>0</b>
r		rh													r

符号	位序号	类型	功能描述
<b>RESULT</b>	[13:2]	rh	<b>转换结果</b> 该位域存放数据压缩滤波的结果。
<b>0</b>	[1:0], [15:14]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.15.3 结果状态映射寄存器

结果状态映射寄存器中存放已被读取的最新结果寄存器（不带“V”的寄存器）的相关状态信息。读取结果寄存器的内容之后更新该位域。

*注：标准读取结果寄存器 RESRx 给出 4 位通道编号。若实际应用需要 5 位通道编号，则在读取 RESRx 或 RESRax 之后从位域 RSSR.CHINR 读出相应值。*

#### RSSR

结果状态映射寄存器										XSFR (82 <sub>H</sub> )										复位值: 0000 <sub>H</sub>									
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0														
0		RRNR				0										CHNR													
r		rh				r										rh													

符号	位序号	类型	功能描述
CHNR	[3:0]	rh	<b>通道编号</b> 该位域指示已被读出的最新转换结果对应的通道编号。
RRNR	[14:12]	rh	<b>结果寄存器编号</b> 该位域指示保存在 CHNR 中的信息属于哪个结果寄存器。
0	[11:4], 15	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.15.4 有效标志寄存器

有效标志寄存器用于指示结果寄存器的内容是否有效（有效 = “新” = 未被读取）。

#### VFR

有效标志寄存器								XSFR (80 <sub>H</sub> )								复位值: 0000 <sub>H</sub>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
0								VF7	VF6	VF5	VF4	VF3	VF2	VF1	VF0				
r								rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh				

符号	位序号	类型	功能描述
<b>VFx</b> ( <b>x = 0-7</b> )	x	rw	<b>结果寄存器 x 的有效标志</b> 该位指示结果寄存器 x 中的内容是否有效。 写 0 不起作用，写 1 将清除相应的标志位以及寄存器 <b>RCRx (x = 0-7)</b> 中的位域 <b>DRC</b> 。 若硬件事件触发置位 <b>VFx</b> 的同时软件对该位置 1， <b>VFx</b> 被清零（软件占优）。 0 <sub>B</sub> 结果寄存器 x 中不含有有效数据。该寄存器已被读取、或无新数据存入。 1 <sub>B</sub> 结果寄存器 x 中包含还未读取的有效数据。
<b>0</b>	[15:8]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.15.5 结果控制寄存器

结果控制寄存器用于控制结果寄存器的行为并监控其状态。

#### RCRx (x = 0-7)

结果控制寄存器 x

**XSFR (B0<sub>H</sub>+x\*2)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>VF</b>	<b>0</b>	<b>DRC</b>	<b>0</b>	<b>WFR</b>	<b>FEN</b>	<b>IEN</b>	<b>0</b>	<b>DRCTR</b>						
r	rh	r	rh	r	rw	rw	rw	r	rw						

符号	位序号	类型	功能描述
<b>DRCTR</b>	[1:0]	rw	<b>数据压缩控制</b> 该位域定义几个转换结果累加用于数据压缩（见 <a href="#">章节 18.2.14.5</a> ）。它定义位域 <b>DRC</b> 的重载值。 00 <sub>B</sub> 禁止数据压缩滤波。DRC 的重载值为 0，故无累加。 01 <sub>B</sub> 使能数据压缩滤波。DRC 的重载值为 1，故累加 2 次的转换结果。 10 <sub>B</sub> 使能数据压缩滤波。DRC 的重载值为 2，故累加 3 次的转换结果。 11 <sub>B</sub> 使能数据压缩滤波。DRC 的重载值为 3，故



符号	位序号	类型	功能描述
			累加 4 次的转换结果。
<b>IEN</b>	4	rw	<b>中断使能</b> 若检测到结果寄存器 x 的结果事件，该位使能产生结果事件中断。 0 <sub>B</sub> 禁止结果事件中断。 1 <sub>B</sub> 使能结果事件中断。
<b>FEN</b>	5	rw	<b>FIFO 使能</b> 该位使能结果寄存器 x 的 FIFO 功能，见 <a href="#">章节 18.2.14.4</a> 。 0 <sub>B</sub> 禁止 FIFO 功能。 1 <sub>B</sub> 使能 FIFO 功能。
<b>WFR</b>	6	rw	<b>等待读取模式</b> 该位使能结果寄存器 x 的等待读取模式。 0 <sub>B</sub> 禁止等待读取模式。 1 <sub>B</sub> 使能等待读取模式。
<b>DRC</b>	[9:8]	rh	<b>数据压缩计数器</b> 该位域指示还需累加几次的转换结果可产生最终的数据压缩结果。该位域变为 0 时（通过递减或重载操作），有效标志自动置位、产生结果事件。 可通过软件置位寄存器 <b>VFR</b> 中的位 x 相应清除位域 RCRx.DRC。 00 <sub>B</sub> 最终的转换结果已产生，可从结果寄存器中取出。 01 <sub>B</sub> 必须再累加 1 个转换结果才可得到最终的转换结果。 10 <sub>B</sub> 必须再累加 2 个转换结果才可得到最终的转换结果。 11 <sub>B</sub> 必须再累加 3 个转换结果才可得到最终的转换结果。
<b>VF</b>	12	rh	<b>有效标志</b> 该位指示结果寄存器 x 中的内容是否有效。它和寄存器 <b>VFR</b> 中相应位的内容相同。

符号	位序号	类型	功能描述
			<b>0<sub>B</sub></b> 结果寄存器 x 中不含有效数据。 <b>1<sub>B</sub></b> 结果寄存器 x 中含有有效数据。
<b>0</b>	[3:2],7, [11:10], [15:13]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.15.6 事件指示标志寄存器

事件指示标志寄存器 **EVINFR** 用于监控检测到的请求源事件（标志 **EVINF0 - EVINF2**）和结果事件（标志 **EVINF8 -EVINF15**）。

#### EVINFR

事件指示标志寄存器

**XSFR (A0<sub>H</sub>)**

复位值: **0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EVINF15</b>	<b>EVINF14</b>	<b>EVINF13</b>	<b>EVINF12</b>	<b>EVINF11</b>	<b>EVINF10</b>	<b>EVINF9</b>	<b>EVINF8</b>	<b>0</b>					<b>EVINF2</b>	<b>EVINF1</b>	<b>EVINF0</b>
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	r					rwh	rwh	rwh

符号	位序号	类型	功能描述
<b>EVINF<sub>x</sub></b> <b>(x = 0-2)</b>	x	rwh	<b>请求源 x 的事件指示标志</b> 标志 <b>EVINF<sub>x</sub></b> 指示已检测到请求源 x 的请求源事件。 写 0 不起作用，写 1 将置位相应的标志位并产生中断请求。 <b>0<sub>B</sub></b> 还未检测到请求源 x 的请求源事件。 <b>1<sub>B</sub></b> 已检测到请求源 x 的请求源事件。
<b>EVINF<sub>x</sub></b> <b>(x = 8-15)</b>	x	rwh	<b>结果寄存器 x-8 的事件指示标志</b> 标志 <b>EVINF<sub>x</sub></b> 指示已检测到结果寄存器 x-8 的结果事件。 写 0 不起作用，写 1 将置位相应的标志位并产生中断请求。 <b>0<sub>B</sub></b> 还未检测到结果寄存器 x-8 的结果事件。 <b>1<sub>B</sub></b> 已检测到结果寄存器 x-8 的结果事件。

符号	位序号	类型	功能描述
0	[7:3]	r	保留 读操作返回 0；应写入 0。

### 18.2.15.7 事件指示标志清零寄存器

向事件指示标志清零寄存器 EVINCR 中的某位写 1，可清除寄存器 EVINFR 中相应的事件指示标志 EVINFx。若检测到请求源或结果事件时相应位被置 1，标志 EVINFx 被清零。

#### EVINCR

##### 事件指示标志清零寄存器

XSFR (A2<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EVIN C15	EVIN C14	EVIN C13	EVIN C12	EVIN C11	EVIN C10	EVIN C9	EVIN C8	0					EVIN GF2	EVIN GF1	EVIN GF0
w	w	w	w	w	w	w	w	r					rwh	rwh	rwh

符号	位序号	类型	功能描述
<b>EVINGFx</b> (x = 0-2)	x	rwh	<b>请求源 x 的事件指示 GF</b>  0 <sub>B</sub> 读：请求源 x 的请求源事件还未激活服务请求输出。 写：无操作。  1 <sub>B</sub> 读：请求源 x 的请求源事件已激活服务请求输出。 写：清除位 EVINFR.x 和 EVINGFx。
<b>EVINCx</b> (x = 8-15)	x	w	<b>清除结果寄存器 x-8 的事件指示标志</b>  0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除标志 EVINFR.x。
0	[7:3]	r	保留 读操作返回 0；应写入 0。

### 18.2.15.8 事件中断节点指针寄存器

这些寄存器用于定义产生请求源或结果事件中断时激活哪路服务请求输出 SR[3:0]。

#### EVINPR0

事件中断节点指针寄存器 0

XSFR (A8<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						EVINP2		0		EVINP1		0		EVINP0	
r						rw		r		rw		r		rw	

符号	位序号	类型	功能描述
EVINP0, EVINP1, EVINP2	[1:0], [5:4], [9:8]	rw	<b>请求源 x 的中断节点指针</b> 该位域选择由哪条服务请求输出线指示已产生请求源 x 的请求源事件中断。 00 <sub>B</sub> 选择服务请求输出 SR0。 01 <sub>B</sub> 选择服务请求输出 SR1。 10 <sub>B</sub> 选择服务请求输出 SR2。 11 <sub>B</sub> 选择服务请求输出 SR3。
0	[3:2], [7:6], [15:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

#### EVINPR8

事件中断节点指针寄存器 8

XSFR (AC<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		EVINP11		0		EVINP10		0		EVINP9		0		EVINP8	
r		rw		r		rw		r		rw		r		rw	

符号	位序号	类型	功能描述
EVINP8, EVINP9, EVINP10, EVINP11	[1:0], [5:4], [9:8], [13:12]	rw	<b>结果事件 x-8 的中断节点指针</b> 该位域选择由哪条服务请求输出线指示已产生结果寄存器 x-8 的结果事件中断。

符号	位序号	类型	功能描述
			00 <sub>B</sub> 选择服务请求输出 SR0。 01 <sub>B</sub> 选择服务请求输出 SR1。 10 <sub>B</sub> 选择服务请求输出 SR2。 11 <sub>B</sub> 选择服务请求输出 SR3。
<b>0</b>	[3:2], [7:6], [11:10], [15:14]	r	<b>保留</b> 读操作返回 0；应写入 0。

## EVINPR12

**事件中断节点指针寄存器 12**

**XSFR (AE<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>EVINP15</b>	<b>0</b>	<b>EVINP14</b>	<b>0</b>	<b>EVINP13</b>	<b>0</b>	<b>EVINP12</b>								
r	rw	r	rw	r	rw	r	rw								

符号	位序号	类型	功能描述
<b>EVINP12, EVINP13, EVINP14, EVINP15</b>	[1:0], [5:4], [9:8], [13:12]	rw	<b>结果事件 x-8 的中断节点指针</b> 该位域选择由哪条服务请求输出线指示已产生结果寄存器 x-8 的结果事件中断。 00 <sub>B</sub> 选择服务请求输出 SR0。 01 <sub>B</sub> 选择服务请求输出 SR1。 10 <sub>B</sub> 选择服务请求输出 SR2。 11 <sub>B</sub> 选择服务请求输出 SR3。
<b>0</b>	[3:2], [7:6], [11:10], [15:14]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.16 CH7 的复用器测试模式

模拟输入 CH7 实现了一种特定的测试模式，用户可在 ADC 工作期间使能该模式以检查和传感器的连接。

- **复用器测试模式禁用** ( $EMENR.MTM7 = 0$ ) :  
分压器和静态负载  $R_{MTM7}$  的开关打开。模拟输入 CH7 可用于常规测量。
- **复用器测试模式使能** ( $EMENR.MTM7 = 1$ ) :  
分压器和静态负载  $R_{MTM7}$  的开关关闭。模拟输入 CH7 由合成电阻加载，所测得的电压值被分压器降低。

接地电阻值及其电流能力的具体描述请参阅 AC/DC 一章。

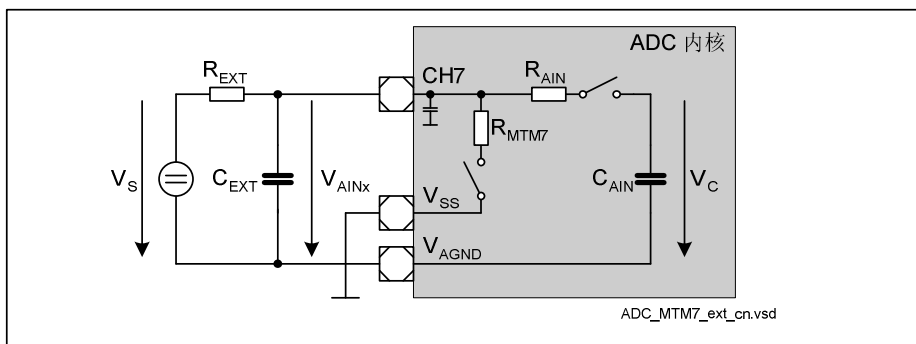


图 18-22 CH7 的复用器测试模式

### 18.2.17 外部复用器控制

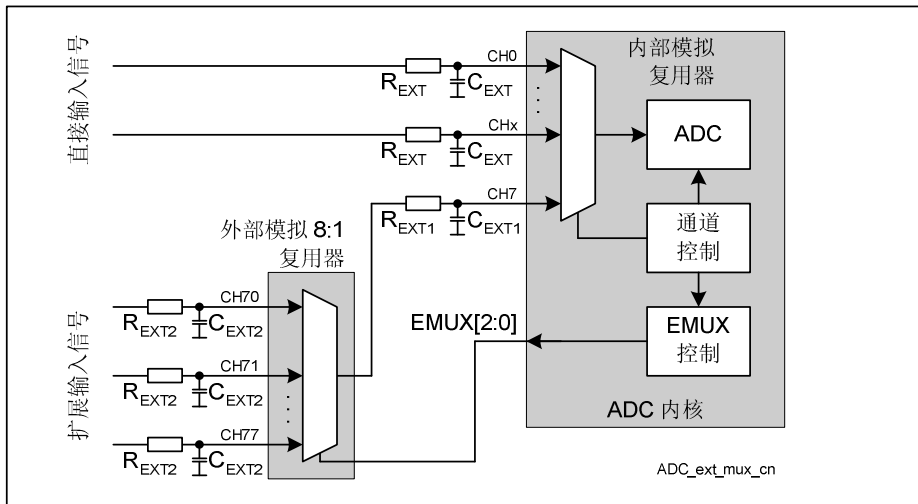
若某应用场合所需的模拟输入通道数多于 XE16xyM 中可用的通道数，ADC 内核通过一个外部模拟复用器来支持通道扩展。每个 ADC 内核产生三路输出信号 EMUX[2:0]，用于控制外部模拟复用器（8 选 1）的设置。

由寄存器 **EMCTR** 和 **EMENR** 控制外部复用器的操作。

复用器的当前设置 EMUX[2:0] 由位域 EMUX 给出。若要转换另一路扩展的输入通道，必须将位域 SETEMUX 设置为期望值或使能扫描功能。对相关模拟输入通道开始进行下一次转换时，自动应用 SETEMUX 值。

输入通道 CH0 - CH15 的每一路均可使能外部复用器支持。

在图 18-23 的示例和以下的描述中，模拟输入 CH7 被扩展为额外的模拟输入 CH70 - CH77。



**图 18-23 外部模拟复用器**

若外部复用器离 ADC 的模拟输入端较远，建议在 ADC 的模拟输入 CH7 上直接引入 RC 滤波器  $R_{EXT1}$ - $C_{EXT1}$ 。若需要进行信号滤波，可选择在外部模拟复用器的输入上添加内部 RC 滤波器  $R_{EXT2}$ - $C_{EXT2}$ 。

若外部复用器离 ADC 的模拟输入端较近，则不需使用  $R_{EXT1}$  和  $C_{EXT1}$ 。在这种情况下，强烈建议在复用器的输入上引入 RC 滤波器 ( $R_{EXT2}$ ,  $C_{EXT2}$ )。

需要注意：每个 RC 滤波器会限制模拟输入信号的带宽。

相比其它（直接）的模拟输入，从 ADC 模拟输入 CH7 端来“看”，外部复用器采用 RC 滤波器可导致另一阻抗。可选用不同的输入组以改变采样阶段的长度（延长采样时间）。假如在开始转换 CH7 之前 EMUX[2:0] 的设置早已修改完毕，可利用新采样时间值执行转换。“在...之前早已...”意味着由 EMUX[2:0] 变化所导致的模拟输入上的信号跳变已结束、输入信号足够稳定。

修改 EMUX[2:0] 的设置之后，在切换到的模拟信号稳定并可被测量之前还需一个额外的稳定时间。为了补偿该稳定时间，在 EMUX[2:0] 改变之后，自动选用备选的采样时间长度（代替输入组定义的采样时间）进行 CH7 的第一次转换。该采样时间可通过位域 **EMCTR.EMSAMPLE** 编程。EMUX[2:0] 改变之后，若 CH7 的第一次转换被更高优先级的请求所中止，重新转换 CH7 时仍使用 EMSAMPLE 的值。在完成 CH7 的整个转换之后稳定时间视为结束。

外部复用器控制模块支持不同的模式，可由寄存器 **EMENR** 编程：

- 无需任何硬件控制的**软件控制** (EMUXEN = 0)：

禁止自动控制外部复用器的设置和采样时间。位域 EMUX 始终由 SETEMUX 的值更新。修改 EMUX 和 SETEMUX 的写操作相关、和转换时序无关。

EMSAMPLE 的设置不予考虑。建议在 EMUXEN = 0 时向 SETEMUX 写入第一个扫描序列的起始值。

- 无扫描的**硬件控制** (EMUXEN = 1、SCANEN = 0) :

每次开始转换 EMUXCHNR 所选择的通道时, EMUX 由 SETEMUX 的值更新。利用新 EMUX 值进行第一次转换时, 使用 EMSAMPLE 的设置。

- 带有单输入扫描的**硬件控制** (EMUXEN = 1、SCANEN = 1、TROEN = 0) :

每次转换 EMUXCHNR 所选择的通道之后, EMUX 被更新, 即自动减 1。若 EMUX = 0, 下次更新时由 SETEMUX 的值重载。每次对所选通道进行转换时, 使用 EMSAMPLE 的设置。

在这种设置下, 若自动扫描序列请求转换 EMUXCHNR 所定义的通道, 则执行一次和外部复用器相连的通道的转换(触发选择禁用)。因此, 每次完成自动扫描序列后, 使用另一种 EMUX 设置。

假设输入 1、2、70、71 和 72 选择用于扫描转换, 将按以下序列执行: 1、2、72、1、2、71、1、2、70、1、2、72、1、2、71、1、2、70、1、2、72...

- 带有多输入扫描的**硬件控制** (EMUXEN = 1、SCANEN = 1、TROEN = 1) :

每次转换 EMUXCHNR 所选择的通道之后, EMUX 被更新, 即自动减 1。若 EMUX = 0, 下次更新时由 SETEMUX 的值重载。每次对所选通道进行转换时, 使用 EMSAMPLE 的设置。

若使能触发选择, 当 EMUX > 0 时, 一旦开始进行转换, 外部复用器控制模块将触发一次新的、由 EMUXCHNR 定义的通道的转换请求。

在扫描请求源中, 相应的挂起位置位; 在顺序请求源中, 备份寄存器的内容变为有效(位 V 置位)。

在这种设置下, 在一次自动扫描序列期间扫描所有外部复用器输入, 从 SETEMUX 指示的通道开始。

假设输入 1、2、70、71 和 72 选择用于扫描转换, 将按以下序列执行: 1、2、72、71、70、1、2、72、71、70、1、2、72、71、70、1、2、72...

### 18.2.18 支持并行采样的同步转换

XE16xyM 中相互独立的 ADC 内核可同步工作, 用于同时测量多个模拟输入通道。若不请求并行转换, 内核可独立工作。

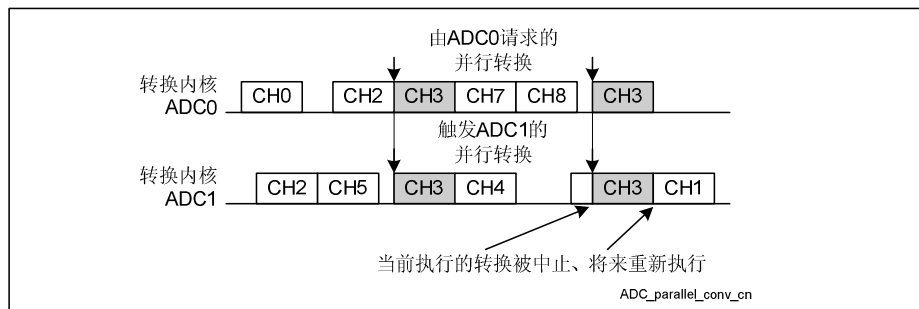
并行转换的同步机制确保了相关通道同时启动采样阶段。支持每个内核具有不同的转换精度和采样时间。



可分别请求每一路输入通道进行并行转换（也可使能多路通道进行并行转换）。在图 18-24 的示例中，同时转换 ADC 内核 ADC0 和 ADC1 的输入通道 CH3，其它输入通道不引发并行转换。

这会导致以下结构：

- **同步主控** ADC 内核可请求转换模拟通道。若该通道被选择用于同步转换，它也会被从控 ADC 内核请求。
- **同步从控** ADC 内核对主控内核发出的同步转换请求作出响应。主控内核不发送请求时，从控内核可转换自己的请求。
- ADC 模块中的所有 ADC 内核相似，每个内核均可作为同步主控或同步从控内核（这取决于实际应用的需求，比如请求源的触发能力）。



**图 18-24 并行转换**

主控内核和从控内核构成一个支持并行采样的“转换组”：

- 同一转换组中的内核可执行并行转换。
- 一个转换组至少包含一个 ADC 内核。
- 转换组中只包含一个同步主控内核，它发送并行转换请求、定义内部频率  $f_{\text{ADCI}}$  和  $f_{\text{ADCD}}$ 、定义转换组中并行转换的通道编号。
- 转换组中的其它内核均为同步从控内核，其 **GLOBCTR.DIVA**、**DIVD** 和 **ARBRND** 必须和同步主控内核的设置相同。
- 若无需并行转换，每个内核可视为自身构成一个转换组，组内只有一个同步主控内核、无同步从控内核。
- 若同步主控内核通过 **CHCTR<sub>x</sub> (x =0-15).SYNC = 1** 请求转换，则由该主控内核向转换组中的其它内核发送通道编号和同步请求。
- 一旦开始并行转换，不可被中止。

- 在同步从控内核中，并行转换请求的优先级始终最高，支持取消-插入-重复模式（见**章节 18.2.6.2**）。
- 同步从控内核的位 **GLOBCTR.ARB** 必须为 0。
- 同步主控内核支持等待读取模式，不过该设置在从控内核中忽略不用（先前的结果可能被覆盖）。

主控内核的同步请求发送机制基于位域 **GLOBSTR.ANON** 的状态。转换组中同步主控内核的 **GLOBCTR.ANON** 信息将分配给所有从控内核（设置所有内核的位域 **SYNCTR.STSEL** 时，必须确保所有内核将使用相同的信息）。除 **ANON** 信息之外，主控内核还将所请求的通道编号发送给从控内核（在**图 18-25**中未明确示出）。

转换组中的所有内核是否开始转换基于 **ready** 信号的状态，该信号指示内核何时准备就绪、可进入采样阶段。位 **SYNCTR.EVALR1** 定义某内核是否必须等待其它内核（进行并行转换）或不需等待可开始转换（不进行并行转换）。为了支持并行转换，必须考虑转换组中内核的就绪信号。

替换特性不受同步转换的影响。转换组中的所有内核请求转换同一路通道（由主控内核定义），但可转换不同输入上的模拟信号。所请求的通道编号可通过其替换特性重新定向至其它通道。例如，若转换组请求转换通道 **CH0**，但某内核的备选参考电压和该输入相连，实际将要转换的模拟输入则可修改为任意值，可通过相应设置位域 **ALIAS0** 来实现。

*注：若某结果寄存器已用于其它通道的数据压缩，从控内核的并行转换结果不应指向该结果寄存器。*

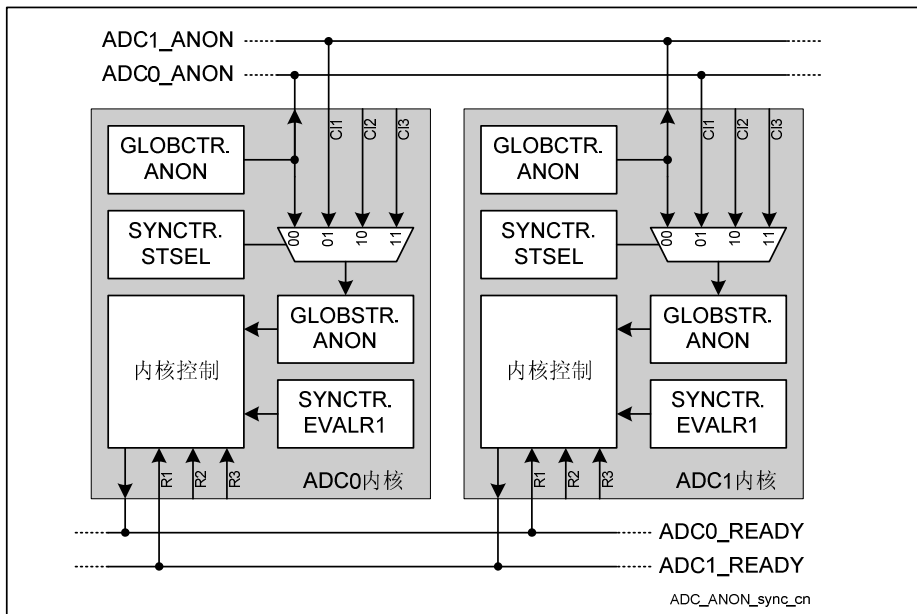


图 18-25 由 ANON 和 Ready 信号控制的同步转换

表 18-4 内核同步的 SYNCTR 设置

操作模式	SYNCTR.EVALR1	SYNCTR.STSEL
<b>ADC0 内核（编程到 ADC0_SYNCTR 的值）</b>		
无同步	0 <sub>B</sub>	00 <sub>B</sub>
ADC1 的主控内核	1 <sub>B</sub>	00 <sub>B</sub>
ADC1 的从控内核	1 <sub>B</sub>	01 <sub>B</sub>
<b>ADC1 内核（编程到 ADC1_SYNCTR 的值）</b>		
无同步	0 <sub>B</sub>	00 <sub>B</sub>
ADC0 的主控内核	1 <sub>B</sub>	00 <sub>B</sub>
ADC0 的从控内核	1 <sub>B</sub>	01 <sub>B</sub>

### 18.2.19 等间隔采样

每个 ADC 内核支持单路（或多路）模拟输入通道的等间隔采样。

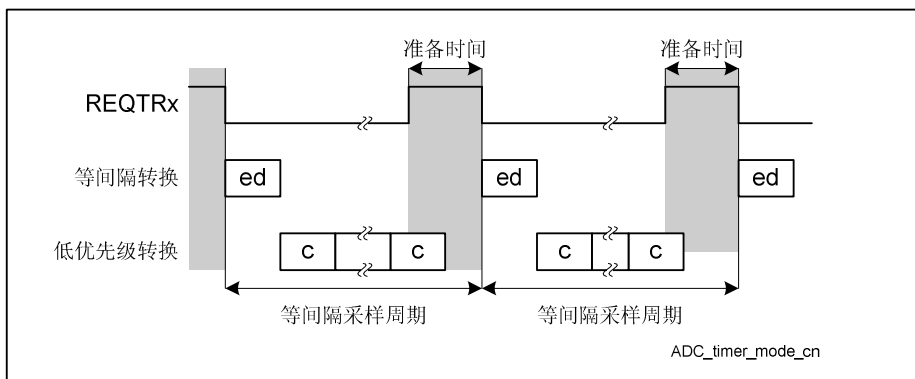
因此，每个请求源均可编程参与仲裁并有可能仲裁胜出（取决于设定的优先级），但不会立即启动转换。真正启动转换的操作由被选作请求源  $x$  的触发输入  $REQTRx$  的信号来控制（该信号在 ADC 模块外围产生，如通过定时器模块产生）。若  $REQTRx$  信号的产生和仲裁器时序同步，则可确保等间隔采样。每个 ADC 内核提供一个输出  $ARBCNT$ ，每一轮仲裁时该信号被激活一次，从而由 ADC 模块外围的定时器单元对仲裁周期计数，基于该时序执行等间隔采样。

若转换器空闲且仲裁器已决出待转换的通道，被请求的等间隔转换可开始采样。为了确保转换器空闲，仲裁器决定转换哪路通道（仲裁胜出通道），但需等到产生定时器控制信号之后才能真正开始进行测量（准备时间）。若被选作等间隔采样的请求源的优先级最高，其他请求源无法中断等间隔采样。

使能请求源输入寄存器的定时器模式（ $RSIRx.TMEN = 1$ ），可利用触发信号  $REQTRx$  进行等间隔采样。信号  $REQTRx$  的频率决定了采样率，其高位时间决定了该请求源参与仲裁的准备时间间隔的长度。在准备时间内可完成当前的转换。编程准备时间时必须确保转换器可变为空闲。

若信号  $ARBCNT$  用作定时器的计数输入信号，必须编程仲裁器恒定工作（ $GLOBCTR.ARBm = 0$ ）。若定时器具有独立的时间基准，无请求挂起时仲裁器可被终止。准备时间必须长于一个仲裁周期。

根据请求等间隔采样的请求源不同，可依次转换一路或多路通道。若请求源所请求的通道顺序固定，也支持对多路通道等间隔采样。只要准备时间和等间隔转换不重叠，也支持多个请求源的并行等间隔采样。



**图 18-26 用于等间隔采样的定时器模式**

### 18.2.20 断线检测

为了支持安全性应用的自检，每个 ADC 内核提供一种断线检测机制，检查传感器的连接或其它电压源和模拟输入的连接。

该机制允许在开始采样和转换之前准备启用电容  $C_{AIN}$ 。若  $BWDENR.ENx = 1$ ，在输入通道的每次转换中附加准备阶段（每路输入通道 CH0-CH15 可单独使能断线检测）。

模数转换包括以下各阶段：

- **可选的准备阶段：**

若某通道允许断线检测，在开始采样阶段之前，将电容  $C_{AIN}$  连接到由  $BWDCFR.CHx$  定义的模拟输入  $CHx$  上。准备阶段和本次转换的采样阶段的长度相同。

若某通道禁止断线检测，准备阶段省略（缺省设置）。

- **采样阶段：**

在采样阶段，通过输入复用器将电容  $C_{AIN}$  连接到由一路模拟输入  $CHx$  上（见**章节 18.1.8.1**）。请求源和仲裁器决定哪路模拟输入的优先级最高。

- **转换阶段：**

在转换阶段，电容  $C_{AIN}$  和模拟输入断开，执行模数转换。该阶段结束时， $C_{AIN}$  被充电至约  $V_{AREF}/2$ 。

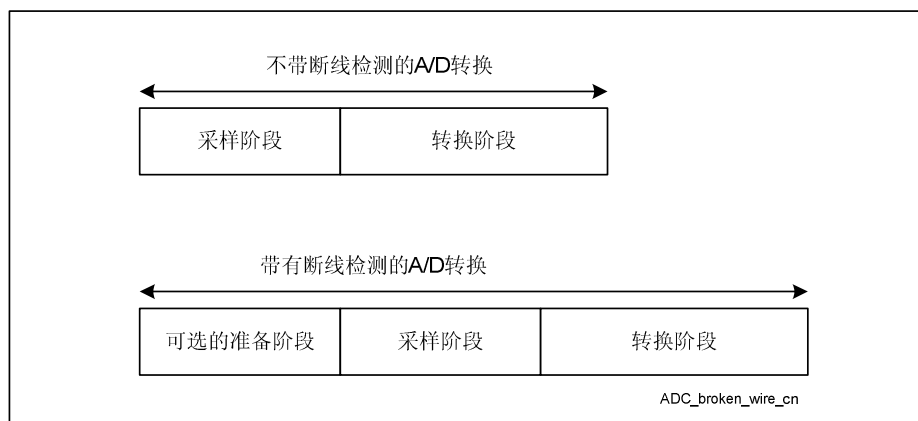


图 18-27 断线检测

断线检测机制允许施加一个超过期望测量范围的电压值。若使能断线检测，当实际的数字转换结果超出期望的测量范围时（如采用极限检查），会检测到一个错误连接。

建议用户确保在准备阶段所施加的电压和传感器输出范围之间有足够的裕量，从而使寄生效应和漏电造成的影响最小。

特别引入输入通道 CH16 ( $V_{AGND}$ ) 和 CH17 ( $V_{AREF}$ )，从而支持选择测量范围的最大和最小值。

*注：若使能断线检测，模数转换的总时间会因插入准备阶段而延长，这会影响转换序列的时序。*

## 18.2.21 附加特性寄存器

### 18.2.21.1 外部复用器使能寄存器

外部复用器使能寄存器用于定义哪路模拟输入通道用于控制外部模拟复用器的设置和其工作模式。该寄存器中还包含位 MTM7，用于控制 CH7 的复用器测试模式。

#### EMENR

##### 外部复用器使能寄存器

##### XSFR (D6<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MTM7	0						EMUXEN	SCANEN	TROEN	0	EMUXCHNR				
rw	r						rw	rw	rw	r	rw				

符号	位序号	类型	功能描述
EMUXCHNR	[3:0]	rw	<b>和外部复用器相连的通道编号</b> 若使能外部复用器控制 ( $EMUXEN = 1$ )，该位域定义和外部模拟复用器相连的模拟输入通道。
TROEN	5	rw	<b>触发选择使能</b> 该位选择外部复用器（若被使能）的扫描模式的操作。具体描述见 <a href="#">章节 18.2.17</a> 。 0 <sub>B</sub> 选择单输入扫描。禁用触发选择（不自动触发 CH <sub>x</sub> 的更多转换）。 1 <sub>B</sub> 选择多输入扫描。使能触发选择，自动扫描所有外部接在复用器上的输入端（自动触发 CH <sub>x</sub> 的其它转换直至 $EMUX = 0$ ）。
SCANEN	6	rw	<b>扫描使能</b> 该位使能/禁止自动扫描外部复用器的输入以转换

符号	位序号	类型	功能描述
			<p>位域 EMUXCHNR 所选择的通道（只有当 EMUXEN = 1 时才有效）。</p> <p>0<sub>B</sub> 禁用扫描模式。开始对所选通道进行转换时，由位域 SETEMUX 更新 EMUX。若位域 EMUX 被修改，使用 EMSAMPLE 的值。</p> <p>1<sub>B</sub> 使能扫描模式。每次转换所选择的通道之后，位域 EMUX 减 1。当 EMUX 递减至 0 后被 SETEMUX 的值更新。始终使用 EMSAMPLE 的值。</p> <p>建议在 EMUXEN = 0 时向 SETEMUX 写入第一个扫描序列的起始值。</p>
<b>EMUXEN</b>	7	rw	<p><b>外部复用器控制使能</b></p> <p>该位使能/禁止自动控制外部复用器。</p> <p>0<sub>B</sub> 禁止硬件控制外部复用器。通过软件写操作 SETEMUX 立即更新位域 EMUX。SCANEN 和 TROEN 的设置忽略不用。</p> <p>1<sub>B</sub> 使能硬件控制外部复用器。通过硬件更新位域 EMUX（考虑转换时序）。</p>
<b>MTM7</b>	15	rw	<p><b>CH7 的复用器测试模式</b></p> <p>该位使能/禁止 CH7 的复用器测试模式（见 <a href="#">章节 18.2.6</a>）。</p> <p>0<sub>B</sub> 禁止复用器测试模式。</p> <p>1<sub>B</sub> 使能复用器测试模式。</p>
<b>0</b>	4, [14:8]	r	<p><b>保留</b></p> <p>读操作返回 0；应写入 0。</p>

### 18.2.21.2 外部复用器控制寄存器

外部复用器控制寄存器定义外部模拟复用器的设置和可选的采样时间。

#### EMCTR

#### 外部复用器控制寄存器

XSFR (D0<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EMSAMPLE								0	EMUX			0	SETEMUX		
rw								r	rh			r	rw		

符号	位序号	类型	功能描述
SETEMUX	[2:0]	rw	<p><b>外部复用器的设置</b></p> <p>若禁用外部复用器控制，EUMX 由 SETEMUX 的值加载。若使能外部复用器控制，有以下两种选择：</p> <p><b>扫描模式禁用：</b></p> <p>该位域定义将被选择用于对 EMUXCHNR 选择的通道进行下一次转换的外部复用器的输入。下一次开始转换该通道时，位域 EMUX 由 SETEMUX 的值更新。</p> <p><b>扫描模式使能：</b></p> <p>该位域定义扫描外部复用器输入的起始值。扫描从设定的输入值开始、从高到低直至输入 0。若 EMUX = 0，完成该通道的转换后，位域 EMUX 由 SETEMUX 的值更新。</p>
EMUX	[6:4]	rh	<p><b>外部复用器的当前设置</b></p> <p>该位域定义将进行转换的外部复用器的输入。其值可从输出线 EMUX[2:0]上得到。</p> <p>若禁用外部复用器控制，EUMX 由 SETEMUX 的值加载。若使能外部复用器控制，有以下两种选择：</p> <p><b>扫描模式禁用：</b></p> <p>开始转换 EMUXCHNR 选择的通道时，该位域由 SETEMUX 的值更新。</p> <p><b>扫描模式使能：</b></p>



符号	位序号	类型	功能描述
			完成 EMUXCHNR 选择的通道的转换后，该位域减 1。EUMX 递减至 0 后被 SETEMUX 值重载。
<b>EMSAMPLE</b>	[15:8]	rw	<b>外部复用器的采样时间</b> 若开始进行转换后外部复用器的设置已改变，该位域定义了一个备选的采样时间长度（此时输入组定义的采样时间无效）。 通过设置该位域可增加至少两个模拟时钟周期的采样时间。 $\text{采样时间} = (2 + \text{EMSAMPLE}) / f_{\text{ADCI}}$
<b>0</b>	3,7	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.21.3 同步控制寄存器

同步控制寄存器用于控制各内核同步工作（并行转换）。必须在各内核的位域 **GLOBSTR.ANON** = 00<sub>B</sub> 的情况下设置它们的寄存器 **SYNCTR**，随后同步主控内核的位域 **ANON** 可置为 11<sub>B</sub>。在并行转换时建议不选用省电模式（**ANON** = 01<sub>B</sub> 或 10<sub>B</sub>）。

只有当主控内核请求同步、并行转换时，位 **EVALRx** 才有效。这确保了同步转换组中的各内核同时开始转换、进入并行采样阶段（尽管可能有一个内核空闲，主控内核以及和它相连的被控内核必须等到所有内核准备就绪）。

### SYNCTR

同步控制寄存器

**XSFR (1A<sub>H</sub>)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0									0	EVALR1	0	STSEL			
r									rw		rw	r	rw		

符号	位序号	类型	功能描述
<b>STSEL</b>	[1:0]	rw	<b>开始选择</b> 该位域控制 ADC 内核的同步机制。 00 <sub>B</sub> 内核为同步主控内核。 使用内核自身的位域 <b>GLOBCTR.ANON</b> 。

符号	位序号	类型	功能描述
			<b>01<sub>B</sub></b> 内核为同步从控内核。 使用输入 <b>CI1</b> 的控制信息（见图 18-25）。 <b>10<sub>B</sub></b> 保留不用（内核关闭）。 <b>11<sub>B</sub></b> 保留不用（内核关闭）。
<b>EVALR1</b>	<b>4</b>	<b>rw</b>	<b>评估 Ready 输入 R1</b> 该位域定义某内核是否属于转换组。只有当转换组中的同步主控和从控内核指示它们已作好并行转换的准备，才能开始进行并行转换。 <b>0<sub>B</sub></b> Ready 输入 R1 不用来控制启动并行转换。 <b>1<sub>B</sub></b> Ready 输入 R1 用来控制启动并行转换。
<b>0</b>	<b>[6:5]</b>	<b>r</b>	<b>保留</b> 读操作返回 0；必须写入 0。
<b>0</b>	<b>[3:2], [15:7]</b>	<b>r</b>	<b>保留</b> 读操作返回 0；应写入 0。

### 18.2.21.4 断线检测使能寄存器

断线检测使能寄存器用于定义某通道是否允许断线检测（即在采样阶段之前插入准备阶段）。通道编号为仲裁胜出方对应的编号（可通过替换特性指向其它输入）。

#### BWDENR

断线检测使能寄存器

**XSFR (C8<sub>H</sub>)**

复位值: **0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EN 15</b>	<b>EN 14</b>	<b>EN 13</b>	<b>EN 12</b>	<b>EN 11</b>	<b>EN 10</b>	<b>EN 9</b>	<b>EN 8</b>	<b>EN 7</b>	<b>EN 6</b>	<b>EN 5</b>	<b>EN 4</b>	<b>EN 3</b>	<b>EN 2</b>	<b>EN 1</b>	<b>EN 0</b>
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
<b>EN<sub>x</sub></b> <b>(x = 0-15)</b>	<b>x</b>	<b>rw</b>	<b>通道 CH<sub>x</sub> 断线检测使能</b> 该位定义是否使能 CH <sub>x</sub> 的断线检测机制。 <b>0<sub>B</sub></b> 禁止断线检测。 <b>1<sub>B</sub></b> 使能断线检测。

### 18.2.21.5 断线检测配置寄存器

断线检测配置寄存器用于定义哪路通道用于插入附加的准备阶段。

#### BWDCFGR

##### 断线检测配置寄存器

**XSFR (CA<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>										<b>CHP</b>					
r										rw					

符号	位序号	类型	功能描述
<b>CHP</b>	[4:0]	rw	<b>准备阶段的通道编号</b> 该位域定义哪路输入通道用于插入准备阶段、进行断线检测。
<b>0</b>	[15:5]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 18.3 XE16xyM 中 ADC 的实现

本节将描述 XE16xyM 中 ADC 内核的实现。它包含以下内容：

- 地址映射（见[章节 18.3.1](#)）
- 中断控制寄存器（见[章节 18.3.2](#)）
- ADC0 模拟信号的连接（见[章节 18.3.3.1](#)）
- ADC1 模拟信号的连接（见[章节 18.3.3.2](#)）
- ADC0 数字信号的连接（见[章节 18.3.4.1](#)）
- ADC1 数字信号的连接（见[章节 18.3.4.2](#)）

### 18.3.1 地址映射

ADC 内核 ADC0 和 ADC1 寄存器的基地址归纳见下表。寄存器的准确地址为相对地址（由表 18-2 给出）加内核基地址（由表 18-5 给出）。

**表 18-5 寄存器地址空间**

模块	起始地址	结束地址	注
ADC0	E000 <sub>H</sub>	E0FF <sub>H</sub>	
ADC1	E100 <sub>H</sub>	E1FF <sub>H</sub>	

**表 18-6 寄存器概览**

寄存器缩写名	寄存器完整名	偏移地址	页码
参见章节 18.2.1 的寄存器列表			

### 18.3.2 中断控制寄存器

中断控制寄存器位于 SFR 区，在中断一章已对它们进行了详细描述。

**表 18-7 ADC 中断控制寄存器**

缩写名	说明
ADC_0IC	ADC0 中 SR0 的中断控制寄存器
ADC_1IC	ADC0 中 SR1 的中断控制寄存器
ADC_2IC	ADC0 中 SR2 的中断控制寄存器
ADC_3IC	ADC0 中 SR3 的中断控制寄存器
ADC_4IC	ADC1 中 SR0 的中断控制寄存器
ADC_5IC	ADC1 中 SR1 的中断控制寄存器
ADC_6IC	ADC1 中 SR2 的中断控制寄存器
ADC_7IC	ADC1 中 SR3 的中断控制寄存器

### 18.3.3 模拟信号的连接

两个 ADC 内核的输入通道分配如下：

- ADC0 的 16 路通道（CH0-CH15）连接至 P5
- ADC1 的 8 路通道（CH0-CH7）连接至 P15
- ADC1 的 4 路通道（CH8-CH11）和 ADC0 的通道（CH8-CH11）重叠
- 每个 ADC 内核的 CH16 连接至它的  $V_{AGND}$  输入
- 每个 ADC 内核的 CH17 连接至它的  $V_{AREF}$  输入

每个 ADC 内核有自己的参考输入线  $V_{AGND}$  和  $V_{AREF}$ 。根据封装的不同，这些输入线在引脚多的封装中可作为独立的引脚使用、在引脚少的封装中可合并使用。

两个转换器各自的电源线是相连的。

#### 18.3.3.1 ADC0 模拟信号的连接

ADC0 模拟信号的连接归纳见下表。

**表 18-8 XE16xyM 中 ADC0 模拟信号的连接**

信号	来自/送至 模块	和 <b>ADC0</b> 连接的 I/O	可用于/用作、连接至
电源和标准参考			
V <sub>DDPA</sub>	参见引脚 一章	输入	模拟电源
V <sub>SS</sub>		输入	模拟电源
V <sub>AREF0</sub>		输入	正模拟参考
V <sub>AGND</sub>		输入	负模拟参考
模拟输入通道			
CH0	P5.0	输入	模拟输入通道 0
CH1	P5.1	输入	模拟输入通道 1
CH2	P5.2	输入	模拟输入通道 2
CH3	P5.3	输入	模拟输入通道 3
CH4	P5.4	输入	模拟输入通道 4
CH5	P5.5	输入	模拟输入通道 5

信号	来自/送至模块	和 ADC0 连接的 I/O	可用于/用作、连接至
CH6	P5.6	输入	模拟输入通道 6
CH7	P5.7	输入	模拟输入通道 7
CH8	P5.8	输入	模拟输入通道 8, 和 ADC1 的通道 8 重叠
CH9	P5.9	输入	模拟输入通道 9, 和 ADC1 的通道 9 重叠
CH10	P5.10	输入	模拟输入通道 10, 和 ADC1 的通道 10 重叠
CH11	P5.11	输入	模拟输入通道 11, 和 ADC1 的通道 11 重叠
CH12	P5.12	输入	模拟输入通道 12
CH13	P5.13	输入	模拟输入通道 13
CH14	P5.14	输入	模拟输入通道 14
CH15	P5.15	输入	模拟输入通道 15
CH16	V <sub>AGND</sub>	输入	模拟输入通道 16, 内部连接至 ADC0 的 V <sub>AGND</sub> 输入
CH17	V <sub>AREF</sub>	输入	模拟输入通道 17, 内部连接至 ADC0 的 V <sub>AREF</sub> 输入
CH18..31	n.c.	输入	不可用, 不请求转换

### 18.3.3.2 ADC1 模拟信号的连接

ADC1 模拟信号的连接归纳见下表。

**表 18-9 XE16xyM 中 ADC1 模拟信号的连接**

信号	来自/送至模块	和 ADC0 连接的 I/O	可用于/用作、连接至
<b>电源和标准参考</b>			
V <sub>DDPA</sub>	参见引脚一章	输入	模拟电源
V <sub>SS</sub>		输入	
V <sub>AREF1</sub>		输入	正模拟参考

信号	来自/送至模块	和 ADC0 连接的 I/O	可用于/用作、连接至
V <sub>AGND</sub>		输入	负模拟参考

#### 模拟输入通道

CH0	P15.0	输入	模拟输入通道 0
CH1	P15.1	输入	模拟输入通道 1
CH2	P15.2	输入	模拟输入通道 2
CH3	P15.3	输入	模拟输入通道 3
CH4	P15.4	输入	模拟输入通道 4
CH5	P15.5	输入	模拟输入通道 5
CH6	P15.6	输入	模拟输入通道 6
CH7	P15.7	输入	模拟输入通道 7
CH8	P5.8	输入	模拟输入通道 8, 和 ADC0 的通道 8 重叠
CH9	P5.9	输入	模拟输入通道 9, 和 ADC0 的通道 9 重叠
CH10	P5.10	输入	模拟输入通道 10, 和 ADC0 的通道 10 重叠
CH11	P5.11	输入	模拟输入通道 11, 和 ADC0 的通道 11 重叠
CH16	V <sub>AGND</sub>	输入	模拟输入通道 16, 内部连接至 ADC1 的 V <sub>AGND</sub> 输入
CH17	V <sub>AREF</sub>	输入	模拟输入通道 17, 内部连接至 ADC1 的 V <sub>AREF</sub> 输入
CH12..15, 18..31	n.c.	输入	不可用, 不请求转换

### 18.3.4 数字信号的连接

以下各表给出 XE16xyM 器件中 ADC 内核和其它模块或引脚的数字信号连接。

以下各节只涉及模块之间的连接, 每个内核的服务请求输出 SR[3:0]和中断控制寄存器的连接在[章节 18.3.2](#)给出。

注：标记为 “I(s)” 的 ADC 功能输入需额外和  $f_{\text{SYS}}$  同步，这会导致  $2/f_{\text{SYS}}$  加一个不确定的  $1/f_{\text{SYS}}$  的延迟，在进行时序计算时必须考虑在内。只有当输入信号的高电平和低电平平均长于  $1/f_{\text{SYS}}$ ，才能正确检测到输入信号的跳变。

标记为 “I” 的 ADC 功能输入视为已和  $f_{\text{SYS}}$  同步。

### 18.3.4.1 ADC0 数字信号的连接

ADC0 数字信号的连接归纳见下表。

**表 18-10 XE16xyM 中 ADC0 数字信号的连接**

信号	来自/送至模块	ADC0 的 I/O	可用于/用作、连接至
<b>仲裁时序</b>			
ARBCNT	CCU60_T12HRE CCU60_T13HRE CCU61_T12HRE CCU61_T13HRE	输出	对 CCU60、CCU61 等间隔采样的时间基准
<b>外部复用器控制</b>			
EMUX[0]	P6.0	输出	外部模拟复用器的控制
EMUX[1]	P6.1	输出	外部模拟复用器的控制
EMUX[2]	P6.2	输出	外部模拟复用器的控制
<b>请求源 0</b>			
REQGT0A	CCU60_COUT63	输入	CCU60
REQGT0B	CCU61_COUT63	输入	CCU61
REQGT0C	0	输入	请求源 0 的请求门控信号
REQGT0D	0	输入	请求源 0 的请求门控信号
REQGT0E	ERU_PDOUT0	输入	ERU
REQGT0F	ERU_PDOUT1	输入	ERU
REQGT0G	P6.0	输入	外部引脚
REQGT0H	CCU60_CC60	输入	CCU60



信号	来自/送至模块	ADC0 的 I/O	可用于/用作、连接至
REQTR0A	CC2_CC16	输入	CC2
REQTR0B	ERU_TOUT1	输入	ERU
REQTR0C	CCU61_SR3	输入	CCU61
REQTR0D	0	输入	请求源 0 的请求触发信号
REQTR0E	P6.1	输入	外部引脚
REQTR0F	P6.3	输入	外部引脚
REQTR0G	ADC0_REQGT0	输入	通过门控输入 (ENG <sub>T</sub> = 0X) 扩展触发转换的输入选择
REQTR0H	ADC0_SR3	输入	ADC0 的服务请求输出 3
REQTR0	-	输出	请求源 0 选定的触发信号
REQGT0	ADC0_REQTR0G	输出	请求源 0 选定的门控信号

**请求源 1**

REQGT1A	CCU60_COUT63	输入	CCU60
REQGT1B	CCU61_COUT63	输入	CCU61
REQGT1C	0	输入	请求源 1 的请求门控信号
REQGT1D	0	输入	请求源 1 的请求门控信号
REQGT1E	ERU_PDOUT0	输入	ERU
REQGT1F	ERU_PDOUT1	输入	ERU
REQGT1G	P6.0	输入	外部引脚
REQGT1H	CCU60_CC61	输入	CCU60
REQTR1A	CC2_CC17	输入	CC2
REQTR1B	ERU_TOUT1	输入	ERU
REQTR1C	CCU61_SR3	输入	CCU61
REQTR1D	0	输入	请求源 1 的请求触发信号

信号	来自/送至模块	ADC0 的 I/O	可用于/用作、连接至
REQTR1E	P6.1	输入	外部引脚
REQTR1F	P6.3	输入	外部引脚
REQTR1G	ADC0_REQGT1	输入	通过门控输入 (ENGT = 0X) 扩展触发转换的输入选择
REQTR1H	ADC0_SR3	输入	ADC0 的服务请求输出 3
REQTR1	-	输出	请求源 1 选定的触发信号
REQGT1	ADC0_REQTR1G	输出	请求源 1 选定的门控信号
<b>请求源 2</b>			
REQGT2A	CCU60_COUT63	输入	CCU60
REQGT2B	CCU61_COUT63	输入	CCU61
REQGT2C	0	输入	请求源 2 的请求门控信号
REQGT2D	0	输入	请求源 2 的请求门控信号
REQGT2E	ERU_PDOUT0	输入	ERU
REQGT2F	ERU_PDOUT1	输入	ERU
REQGT2G	P6.0	输入	外部引脚
REQGT2H	CCU60_CC62	输入	CCU60
REQTR2A	CC2_CC18	输入	CC2
REQTR2B	ERU_TOUT1	输入	ERU
REQTR2C	CCU61_SR3	输入	CCU61
REQTR2D	0	输入	请求源 2 的请求触发信号
REQTR2E	P6.1	输入	外部引脚
REQTR2F	P6.3	输入	外部引脚
REQTR2G	ADC0_REQGT0	输入	通过门控输入 (ENGT = 0X) 扩展触发转换的输入选择
REQTR2H	ADC0_SR3	输入	ADC0 的服务请求输出 3

信号	来自/送至模块	ADC0 的 I/O	可用于/用作、连接至
REQTR2	-	输出	请求源 2 选定的触发信号
REQGT2	ADC0_REQTR2G	输出	请求源 2 选定的门控信号
<b>服务请求输出</b>			
SR3	CCU60_CCPOS2C	输出	CCU60 霍尔输入触发

#### 18.3.4.2 ADC1 数字信号的连接

ADC1 数字信号的连接归纳见下表。

**表 18-11 XE16xyM 中 ADC1 数字信号的连接**

信号	来自/送至模块	ADC1 的 I/O	可用于/用作、连接至
<b>仲裁时序</b>			
ARB CNT	-	输出	仲裁时序
<b>外部复用器控制</b>			
EMUX[0]	P7.2	输出	外部模拟复用器的控制
EMUX[1]	P7.3	输出	外部模拟复用器的控制
EMUX[2]	P7.4	输出	外部模拟复用器的控制
<b>请求源 0</b>			
REQGT0A	CCU60_COUT63	输入	CCU60
REQGT0B	CCU61_COUT63	输入	CCU61
REQGT0C	0	输入	请求源 0 的请求门控信号
REQGT0D	0	输入	请求源 0 的请求门控信号
REQGT0E	ERU_PDOUT0	输入	ERU
REQGT0F	ERU_PDOUT1	输入	ERU
REQGT0G	P6.0	输入	外部引脚

信号	来自/送至模块	ADC1 的 I/O	可用于/用作、连接至
REQGT0H	0	输入	请求源 0 的请求门控信号
REQTR0A	CC2_CC24	输入	CC2
REQTR0B	ERU_TOUT1	输入	ERU
REQTR0C	0	输入	请求源 0 的请求触发信号
REQTR0D	0	输入	请求源 0 的请求触发信号
REQTR0E	P6.1	输入	外部引脚
REQTR0F	P6.3	输入	外部引脚
REQTR0G	ADC1_REQGT0	输入	通过门控输入 (ENG <sub>T</sub> = 0X) 扩展触发转换的输入选择
REQTR0H	ADC1_SR3	输入	ADC1 的服务请求输出 3
REQTR0	-	输出	请求源 0 选定的触发信号
REQGT0	ADC1_REQTR0G	输出	请求源 0 选定的门控信号

**请求源 1**

REQGT1A	CCU60_COUT63	输入	CCU60
REQGT1B	CCU61_COUT63	输入	CCU61
REQGT1C	0	输入	请求源 1 的请求门控信号
REQGT1D	0	输入	请求源 1 的请求门控信号
REQGT1E	ERU_PDOUT0	输入	ERU
REQGT1F	ERU_PDOUT1	输入	ERU
REQGT1G	P6.0	输入	外部引脚
REQGT1H	0	输入	请求源 1 的请求门控信号
REQTR1A	CC2_CC25	输入	CC2
REQTR1B	ERU_TOUT1	输入	ERU
REQTR1C	0	输入	请求源 1 的请求触发信号

信号	来自/送至模块	ADC1 的 I/O	可用于/用作、连接至
REQTR1D	0	输入	请求源 1 的请求触发信号
REQTR1E	P6.1	输入	外部引脚
REQTR1F	P6.3	输入	外部引脚
REQTR1G	ADC1_REQGT1	输入	通过门控输入 (ENGT = 0X) 扩展触发转换的输入选择
REQTR1H	ADC1_SR3	输入	ADC1 的服务请求输出 3
REQTR1	-	输出	请求源 1 选定的触发信号
REQGT1	ADC1_REQTR1G	输出	请求源 1 选定的门控信号

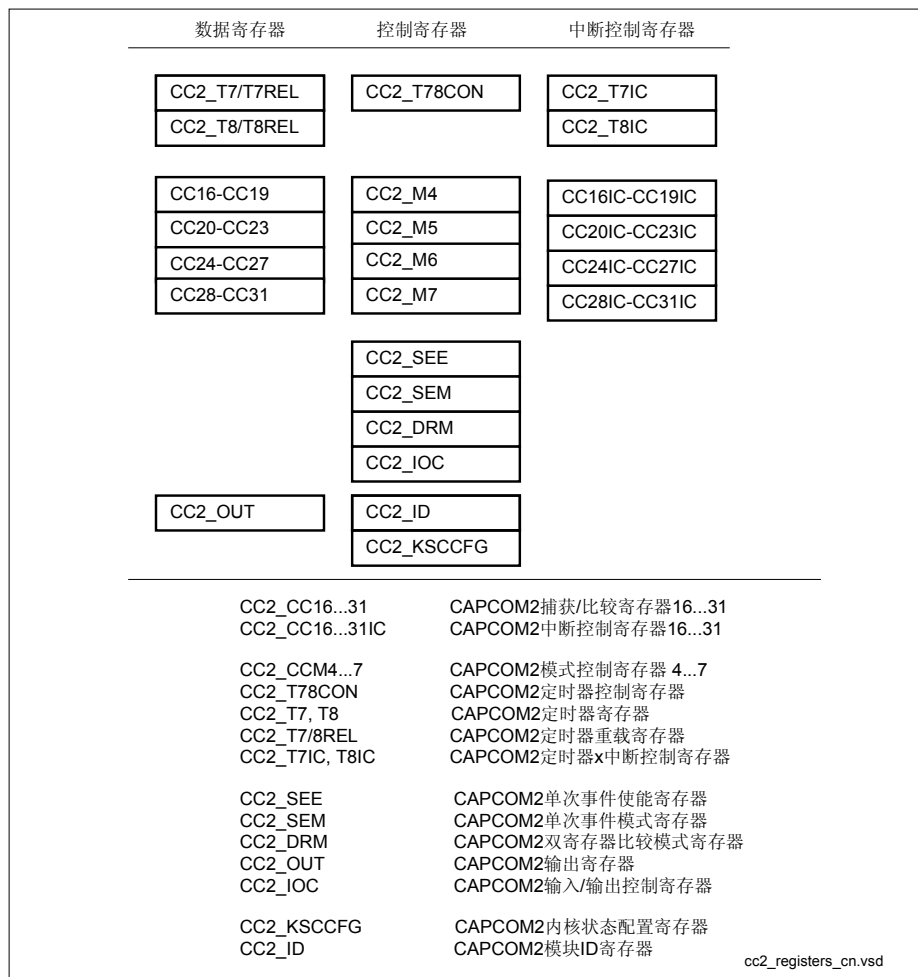
**请求源 2**

REQGT2A	CCU60_COUT63	输入	CCU60
REQGT2B	CCU61_COUT63	输入	CCU61
REQGT2C	0	输入	请求源 2 的请求门控信号
REQGT2D	0	输入	请求源 2 的请求门控信号
REQGT2E	ERU_PDOUT0	输入	ERU
REQGT2F	ERU_PDOUT1	输入	ERU
REQGT2G	P6.0	输入	外部引脚
REQGT2H	0	输入	请求源 2 的请求门控信号
REQTR2A	CC2_CC26	输入	CC2
REQTR2B	ERU_TOUT1	输入	ERU
REQTR2C	0	输入	请求源 2 的请求触发信号
REQTR2D	0	输入	请求源 2 的请求触发信号
REQTR2E	P6.1	输入	外部引脚
REQTR2F	P6.3	输入	外部引脚
REQTR2G	ADC1_REQGT2	输入	通过门控输入 (ENGT = 0X) 扩展触发转换的输入选择

信号	来自/送至模块	ADC1 的 I/O	可用于/用作、连接至
REQTR2H	ADC1_SR3	输入	ADC1 的服务请求输出 3
REQTR2	-	输出	请求源 2 选定的触发信号
REQGT2	ADC1_REQTR2G	输出	请求源 2 选定的门控信号
<b>服务请求输出</b>			
SR3	CCU61_CCPOS2C	输出	CCU61 霍尔输入触发

## 19 捕获/比较单元

XE16xyM 提供一个捕获/比较（CAPCOM2）单元，该模块具有 16 路和 2 个定时器紧密配合工作的捕获/比较通道。CAPCOM 通道可以由特定的内部或外部事件触发捕获定时器的内容；或将定时器的内容和给定值进行比较，匹配时修改输出信号。



**图 19-1 CAPCOM2 单元的相关特殊功能寄存器（SFR）**

CAPCOM2 的两个定时器命名为 T7 和 T8，CAPCOM2 的 16 个通道命名为 CC16\_31。

从编程人员的角度来看，“CAPCOM 单元”由一组与该外设模块相关的寄存器组成（参见图 19-1），其中还包括可用作复用输入/输出功能的引脚及其控制位。

## 19.1 功能概述

通常 CAPCOM2 单元用于处理高速 IO 任务，如脉冲和波形产生、脉宽调制、或记录发生特定事件的时间。CAPCOM2 还支持多达 16 个由软件控制的中断事件。

CAPCOM2 单元由两个 16 位定时器（T7/T8）组成，每个定时器都有对应的重载寄存器（TxREL），以及一组（16 个）双功能 16 位捕获/比较寄存器（CCy）。

CAPCOM 定时器的输入时钟来自经过预分频处理的模块输入时钟（ $f_{cc}$ ），预分频因子可编程设定；或来自定时器 T6 的上溢/下溢信号。T7 也可由外部事件触发、工作在计数器模式（对外部输入计数）。

每个捕获/比较寄存器可分别被设定为捕获或比较操作，并被指定和两个定时器之一配合工作。每个捕获/比较寄存器有一个对应的信号，用作捕获操作的输入信号或比较操作的输出信号。

当执行捕获操作时，一旦输入信号跳变，当前定时器的值将被复制到对应的捕获/比较寄存器中。该事件也将激活相关的中断请求线。

当执行比较操作时，一旦所分配的定时器递增计数至和保存在捕获/比较寄存器中的比较值相等时，一个相关的输出信号将跳变。该比较匹配事件也将激活相关的中断请求线。双寄存器比较模式下，一对寄存器共同控制一个公共的输出信号。

比较输出信号可从专用输出寄存器读取。用户可编程选择输出路径。

可选择两种时序方案（请参阅章节 19.1.10）进行输出信号切换：

**交错模式下**，输出信号依次分 8 步连续切换，各切换步骤占用一定时间。交错模式下，最大精度为  $8 t_{cc}$ 。

**非交错模式下**，各输出信号同时立即切换。非交错模式下，最大精度为  $1 t_{cc}$ 。



图 19-2 所示为 CAPCOM2 单元的基本结构。

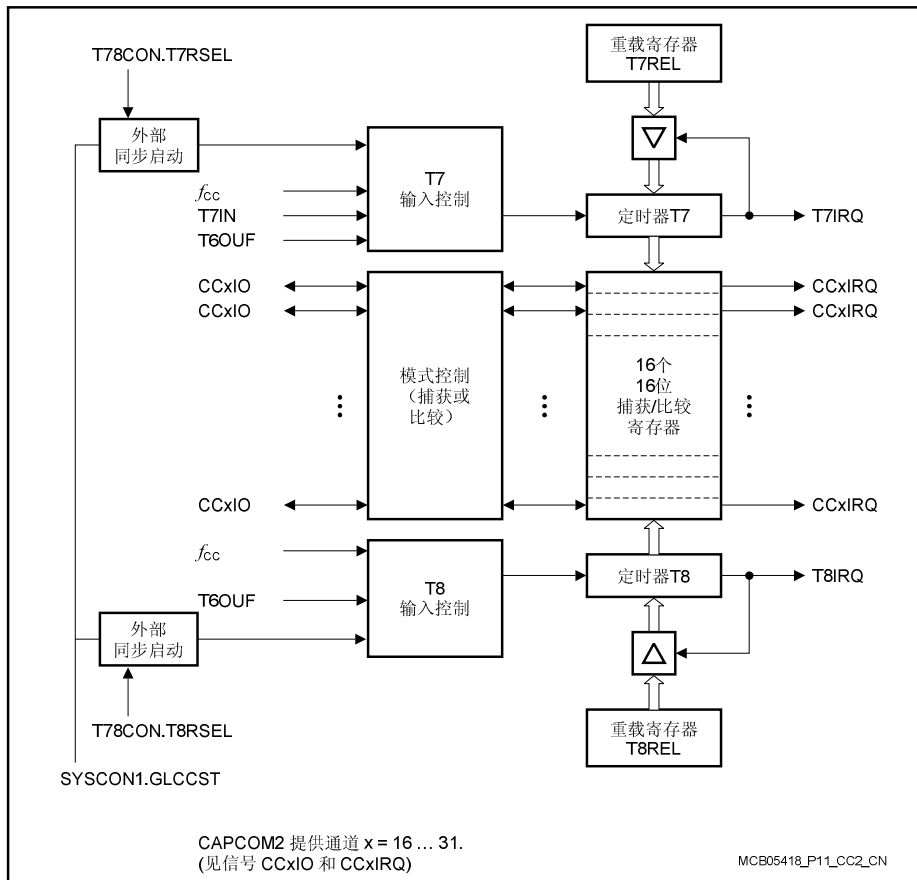


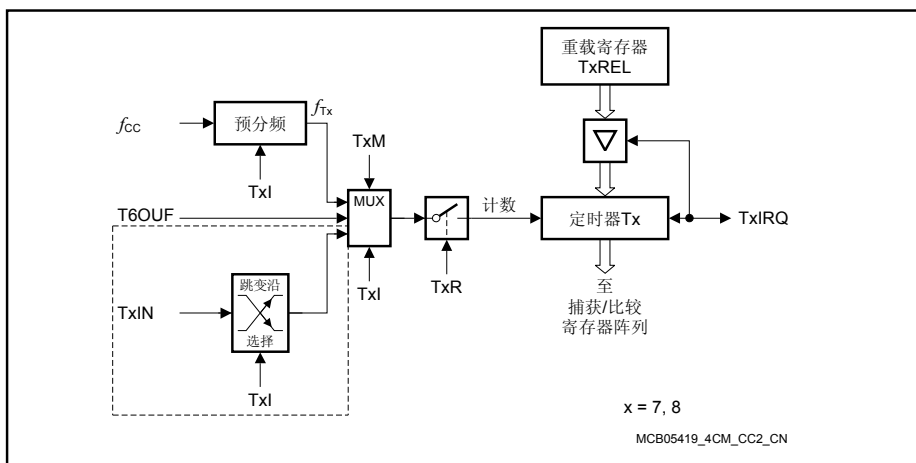
图 19-2 CAPCOM2 单元框图

通过置位 SYSCON1.GLCCST，定时器 T7 和 T8 可以与 CAPCOM6 定时器同步启动。

### 19.1.1 CAPCOM 定时器

定时器 T7 和 T8 主要为 CAPCOM2 单元的各路捕获/比较通道提供两个独立的时间基准。交错模式下最大精度为  $8 t_{cc}$ ，非交错模式下最大精度为  $1 t_{cc}$ 。

除输入引脚（图中虚框标出）之外，定时器的基本结构完全相同，如图 19-3 所示。



**图 19-3 CAPCOM 定时器框图**

CAPCOM 定时器的功能由可位寻址的控制寄存器 **CC2\_T78CON** 控制。**CC2\_T78CON** 的高位字节控制定时器 T8，**CC2\_T78CON** 的低位字节控制定时器 T7。两个定时器的控制选择相同（外部输入除外）。

在所有模式中，定时器始终递增计数。CPU 可通过定时器寄存器 Tx 访问当前定时器的值，Tx 不可位寻址。当 CPU 要对寄存器 Tx 进行写操作时，如果此时定时器正准备加 1 或被重载，CPU 写操作占优，加 1 或重载操作被禁止以保证正确的定时器操作。

定时器运行标志 TxR 控制定时器的启动和停止。以下定时器模式和操作的描述只适用于定时器被使能的状态，即假设相关的运行标志已被置位。

## 定时器模式

定时器模式下 ( $TxM = 0$ )，CAPCOM 定时器的输入时钟来自经过预分频处理的时钟信号  $f_{CC}$ ，预分频因子可编程设定。每个定时器有各自的预分频器，预分频的设置由相应定时器控制寄存器  $CC2\_T78CON$  中的位域  $TxI$  控制。

定时器  $Tx$  的输入频率  $f_{Tx}$  和相应精度  $r_{Tx}$  根据以下公式计算：

交错模式：

$$f_{Tx}[MHz] = \frac{f_{CC}[MHz]}{2^{(<TxI>+3)}} \quad r_{Tx}[\mu s] = \frac{2^{(<TxI>+3)}}{f_{CC}[MHz]} \quad (19.1)$$

非交错模式：

$$f_{Tx}[MHz] = \frac{f_{CC}[MHz]}{2^{<TxI>}} \quad r_{Tx}[\mu s] = \frac{2^{<TxI>}}{f_{CC}[MHz]} \quad (19.2)$$

定时器从  $FFFF_H$  溢出至  $0000_H$  时，对应重载寄存器  $TxREL$  中的内容重新装入定时器中。重载值决定了  $Tx$  连续两次溢出之间的周期  $P_{Tx}$ ：

交错模式：

$$P_{Tx}[\mu s] = \frac{(2^{16} - <TxREL>) \times 2^{(<TxI>+3)}}{f_{CC}[MHz]} \quad (19.3)$$

非交错模式：

$$P_{Tx}[\mu s] = \frac{(2^{16} - <TxREL>) \times 2^{<TxI>}}{f_{CC}[MHz]} \quad (19.4)$$

置位运行标志 ( $TxR$ ) 启动定时器工作之后，首次加 1 操作的执行时间短于设定的定时器精度；接下来的所有递增操作时间均等于设定的定时器精度。

模块输入时钟为 40 MHz 时、不同预分频因子所对应的定时器输入频率、精度和周期值列于表 19-1 中，定时器的周期值通过假设重载值为  $0000_H$  计算得出。请注意某些数值经过舍入。

**表 19-1      定时器模式下定时器 Tx 的输入时钟选择,  $f_{CC} = 40 \text{ MHz}$**

<b>Txl</b>	<b>预分频因子</b>	<b>输入频率</b>	<b>精度</b>	<b>周期</b>
<b>交错模式</b>				
000 <sub>B</sub>	8	5 MHz	200 ns	13.11 ms
001 <sub>B</sub>	16	2.5 MHz	400 ns	26.21 ms
010 <sub>B</sub>	32	1.25 MHz	800 ns	52.43 ms
011 <sub>B</sub>	64	625 kHz	1.6 $\mu\text{s}$	104.86 ms
100 <sub>B</sub>	128	312.5 kHz	3.2 $\mu\text{s}$	209.72 ms
101 <sub>B</sub>	256	156.25 kHz	6.4 $\mu\text{s}$	419.43 ms
110 <sub>B</sub>	512	78.125 kHz	12.8 $\mu\text{s}$	838.86 ms
111 <sub>B</sub>	1024	39.0625 kHz	25.6 $\mu\text{s}$	1677.72 ms
<b>非交错模式</b>				
000 <sub>B</sub>	1	40 MHz	25 ns	1.6384 ms
001 <sub>B</sub>	2	20 MHz	50 ns	3.2768 ms
010 <sub>B</sub>	4	10 MHz	100 ns	6.5536 ms
011 <sub>B</sub>	8	5 MHz	200 ns	13.11 ms
100 <sub>B</sub>	16	2.5 MHz	400 ns	26.21 ms
101 <sub>B</sub>	32	1.25 MHz	800 ns	52.43 ms
110 <sub>B</sub>	64	625 kHz	1.6 $\mu\text{s}$	104.86 ms
111 <sub>B</sub>	128	312.5 kHz	3.2 $\mu\text{s}$	209.72 ms

### 计数器模式

计数器模式下 ( $TxM = 1$ )，CAPCOM 定时器的输入时钟来自外部输入引脚 T7IN，或者 GPT 定时器 T6 的上溢/下溢信号。

用连接到 TxIN 引脚上的外部信号作为计数时钟只适用于定时器 T7。计数器模式下，定时器 T8 只能使用 GPT 定时器 T6 的上溢/下溢信号作为计数时钟 ( $Txl = 000_B$ )。

位域 T7I 选择引脚 T7IN 上外部信号的正跳变、负跳变或任意跳变触发定时器 T7 递增计数。请注意计数器模式下，外部输入信号和端口引脚的设定必须满足某些条件以确保操作正确，具体内容请参阅[章节 19.1.11](#)。

### 定时器溢出和重载

若 CAPCOM 定时器计数到 FFFF<sub>H</sub> 时发生新的计数触发事件，将产生定时器中断请求，并将重载寄存器 TxREL 的值重新装入定时器中。下次计数事件触发定时器从重载值开始继续递增计数。

重载寄存器 TxREL 不可位寻址。复位值为 0000<sub>H</sub>。

### 19.1.2 定时器中断

定时器溢出时，相应的定时器中断请求标志 TxIR 被置位。若中断使能位 TxIE 被置位，该中断请求标志可用来产生中断或触发 PEC 服务请求。

每个定时器各有相应的可位寻址中断控制寄存器 CC2\_TxIC 和中断向量。中断控制寄存器 TxIC 的结构与其它中断控制寄存器相同。

### 19.1.3 捕获/比较通道

16 位捕获/比较寄存器 **CC2\_CCy (y = 16-31)** 分别用作定时器 T7 和 T8 捕获或比较操作的数据寄存器。捕获/比较寄存器不可位寻址。

16 个捕获/比较寄存器由 4 个结构完全相同的可位寻址 16 位模式控制寄存器 **CC2\_M4...CC2\_M7** 控制（详见以下寄存器描述）。每个寄存器分别控制四个捕获/比较寄存器的模式选择和定时器分配。

每个寄存器 CCy 可被设定为捕获模式或四种比较模式之一；并选择与之配合工作的定时器。

特殊的双寄存器比较模式将两个寄存器组合起来控制公共的输出信号。某个 CCy 寄存器的捕获或比较操作被禁止时，该寄存器可保存通用变量。

**表 19-2      捕获模式和比较模式的选择**

模式	MODy	选中的工作模式
禁用	000 <sub>B</sub>	捕获和比较模式禁用 CAPCOM 寄存器可存储通用变量
捕获	001 <sub>B</sub>	引脚 CCyIO 上的正跳变（上升沿）触发捕获操作
	010 <sub>B</sub>	引脚 CCyIO 上的负跳变（下降沿）触发捕获操作

模式	MODy	选中的工作模式
	<b>011<sub>B</sub></b>	引脚 CCyIO 上的任意跳变（上升沿和下降沿）触发捕获操作
<b>比较</b>	<b>100<sub>B</sub></b>	<b>比较模式 0:</b> 只产生中断 每个定时器周期产生多个中断 可使能 Bank2 寄存器的双寄存器比较模式
	<b>101<sub>B</sub></b>	<b>比较模式 1:</b> 每次匹配时翻转输出引脚 每个定时器周期产生多个比较事件 可使能 Bank1 寄存器的双寄存器比较模式
	<b>110<sub>B</sub></b>	<b>比较模式 2:</b> 只产生中断 每个定时器周期只产生一个中断
	<b>111<sub>B</sub></b>	<b>比较模式 3:</b> 每次匹配时置位输出引脚 每次定时器溢出时复位输出引脚；每个定时器周期只产生一个中断

对捕获和比较模式的详细描述适用于所有捕获/比较通道，因此各寄存器、各位和各引脚用符号 x 代表。

#### 19.1.4 捕获模式

捕获模式下，一旦由外部事件触发，CAPCOM 定时器的当前值将被复制（捕获）到相应的捕获/比较寄存器中。该模式可用于记录外部事件的发生时刻、或测量两个外部事件的时间间隔（用定时器的增量表示）。

可设定（和输入引脚连接的）外部信号的正跳变、负跳变或任意跳变作为捕获触发事件，通过对应的模式控制寄存器中的位域 MODy 进行选择。所选择的外部信号跳变发生时，定时器的内容被复制到捕获/比较寄存器中，相应的中断请求线 CCyIRQ 被激活。若中断被使能，该中断请求会引发中断或 PEC 服务请求。

*注：捕获输入可用作附加外部中断输入，此时捕获操作不可用。*

由模式控制寄存器的定时器分配控制位 ACCy 选择捕获定时器 T7 或 T8 的内容。

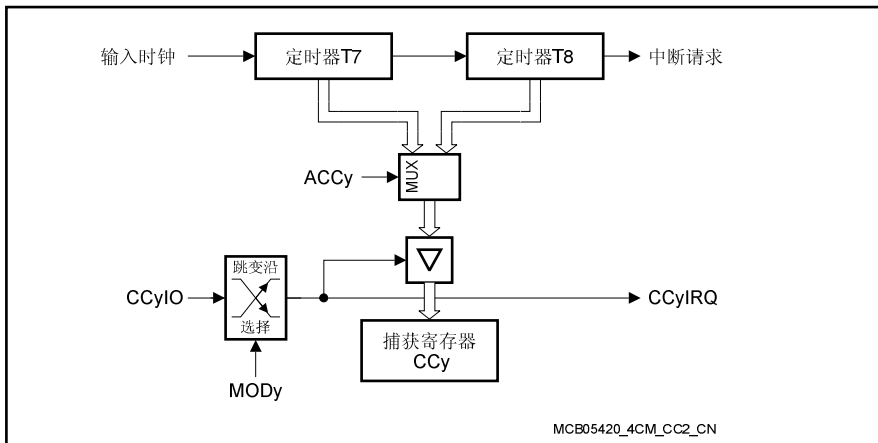


图 19-4 捕获模式框图

捕获操作中的相关引脚必须设定为输入。为了确保能够正确识别输入信号的跳变，输入电平必须至少保持（高或低）规定数目的模块时钟周期之后才能改变。具体内容请参阅[章节 19.1.11](#)。

### 19.1.5 比较模式操作

比较模式可在最低的软件开销下，产生触发事件（中断和/或输出信号跳变）或脉冲序列。所有比较模式中，将保存在捕获/比较寄存器 **CCy** 中的 16 位数值（以下描述中也称为“比较值”）和所分配的定时器（T7 或 T8）的计数值进行连续比较。若定时器的当前值和比较值相匹配，与寄存器 **CCy** 相关的中断请求线被激活，是否在对应的输出引脚 **CCyIO** 上产生输出信号取决于所选择的比较模式。

**CAPCOM2** 支持四种不同的比较模式。通过模式控制寄存器中的位域 **MODy** 可分别为每个捕获/比较寄存器选择对应的比较模式。模式 0 和模式 2 不影响输出信号。下面将对每种模式进行详细说明。

除这些“单寄存器”模式之外，**CAPCOM2** 还支持“双寄存器”比较模式，该模式允许两个寄存器共同控制一个输出引脚。由于可设定两个不同的比较值控制同一个信号的跳变序列，“双寄存器”比较模式进一步降低了软件开销。具体内容请参阅[章节 19.1.6](#)。

所有比较模式中，比较器执行“等于”比较。也就是说，只有当定时器的计数值和比较寄存器的值相等时才检测到匹配。此外，只有在定时器硬件执行加 1 操作后紧接着的一个时钟周期内，比较器才被使能，目的在于防止定时器不以最高输入时钟工作时

（定时器模式或计数器模式）出现重复匹配，此时定时器中的一个值会保持多个或多达上千个时钟周期。该操作的副作用在于，软件修改定时器的值无法获得比较匹配。若软

件设定定时器的值和某个比较寄存器的值相同，不会检测到匹配。比较寄存器的设定值小于定时器的当前计数值时，不执行任何操作。

端口输出功能的具体操作请参阅[章节 19.1.8](#)。

若两个或多个比较寄存器设定的比较值相同<sup>1)</sup>，所分配定时器递增计数到该比较值后，对应的各中断请求标志被置位，并产生预先选择的输出信号。定时器再次递增计数或者被软件写入之前，同一个比较值的其它比较事件被禁止<sup>2)</sup>。复位后，只有当定时器开始递增计数、或已由软件写入，并为捕获/比较寄存器 **CCy** 选择了对应的比较模式，**CCy** 的比较事件才被使能。

#### **19.1.5.1 比较模式 0**

比较模式 0 只产生中断，可用作软件定时。该模式下，每次检测到比较寄存器 **CCy** 中的比较值和对应该定时器的计数值匹配时，激活中断请求线 **CCyIRQ**。“匹配”即指定定时器和比较寄存器的值相等（“=”）。若在一个定时器周期内更新寄存器 **CCy** 中的比较值，其间可能产生多个比较事件。该模式下，比较事件不会影响对应的端口信号 **CCyIO**，因此该端口可用作通用 IO。

*注：若 **bank2** 中的某寄存器设定为比较模式 0，该寄存器的双寄存器比较模式可被使能（请参阅[章节 19.1.6](#)）。*

#### **19.1.5.2 比较模式 1**

比较模式 1 和比较模式 0 的基本操作相同，差别仅在于该模式会影响相关的输出信号。该模式下，每次检测到比较寄存器 **CCy** 中的比较值和对应该定时器的计数值匹配时，激活中断请求线 **CCyIRQ**。同时，相关输出信号翻转。若在一个定时器周期内更新寄存器 **CCy** 中的比较值，其间可能产生多个比较事件。

*注：若 **bank1** 中的某寄存器设定为比较模式 1，该寄存器的双寄存器比较模式可被使能（请参阅[章节 19.1.6](#)）。*

端口输出信号的具体操作请参阅[章节 19.1.8](#)。

---

1) 交错模式下，这些中断和输出信号顺序产生（请参阅[章节 19.1.10](#)）。

2) 定时器执行加 1 操作之前（定时器工作在较低频率），即使经历多个比较周期，给定的一个比较值只产生一个比较事件。



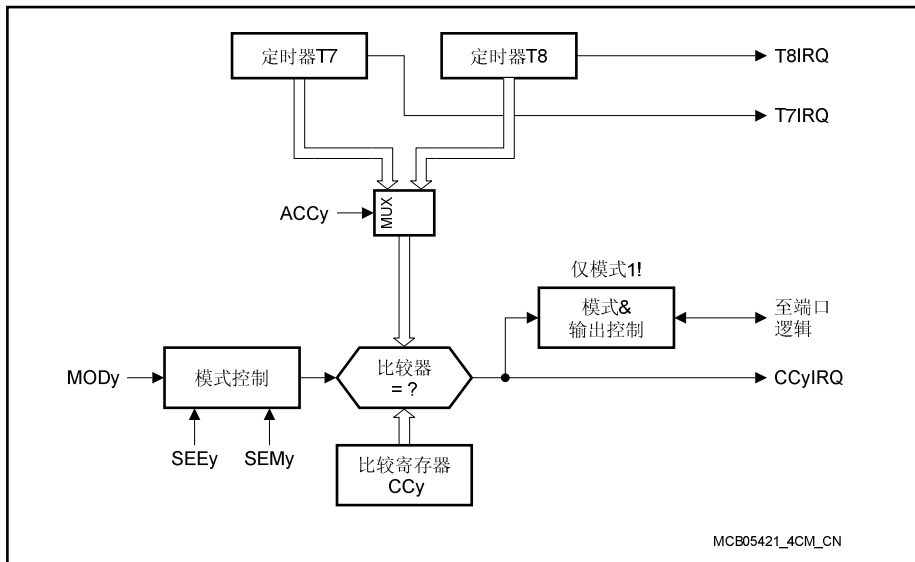


图 19-5 比较模式 0 和 1 框图

注：比较模式 0 输出信号不受影响。

图 19-6 举例说明比较模式 0 和 1 的不同操作情况。

各例中定时器的重载值设定为  $FFF9_H$ 。每当定时器溢出，从该值重新开始递增计数。

**例 1 中**，寄存器  $CCy$  的值为  $FFFC_H$ 。定时器递增计数到该值时检测到匹配，中断请求线  $CCyIRQ$  被激活。比较模式 0 的所有操作完毕；比较模式 1，相关的端口输出翻转，导致输出信号翻转。若寄存器  $CCy$  的值不改变，定时器每次计数到设定的比较值时执行该操作。

**例 2 中**，第一次比较值  $FFFC_H$  匹配之后，软件将  $FFFF_H$  重新装入比较寄存器  $CCy$  中。定时器继续递增计数，计数到新的比较值并检测到新匹配后，激活中断请求线（比较模式 0 和 1）并翻转输出信号（比较模式 1）。若比较值不再改变，定时器再次计数到  $FFFF_H$  时发生匹配。

该例图示说明在当前的定时器周期内可能出现多次比较匹配（和比较模式 2 和 3 相比）。

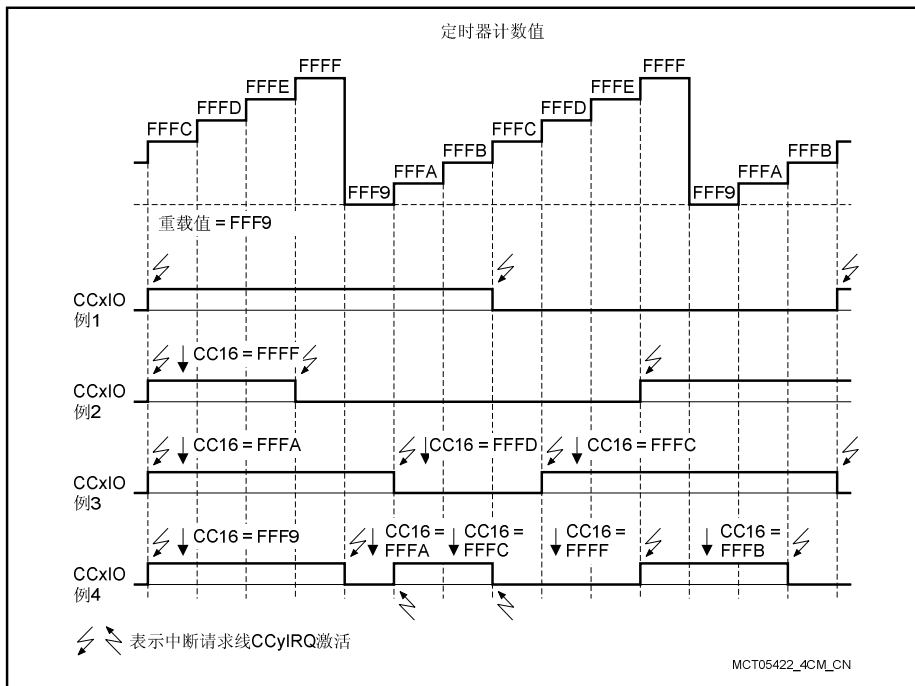


图 19-6 比较模式 0 和 1 时序举例

**例 3 中**，在当前定时器周期内，只有大于定时器当前计数值的新比较值才会引发新匹配。首次匹配（ $FFFC_H$ ）之后将  $FFFA_H$  重新装入比较寄存器中，由于定时器计数已超过新比较值，因此直到下个定时器周期，定时器计数到  $FFFA_H$  时才引发比较匹配。此时将大于定时器当前值的新比较值重新装入寄存器  $CCy$  中，在本周期内将引发新匹配。

**例 4 中**，比较值等于定时器的重载值、或者等于最大计数值  $FFFF_H$ 。

### 19.1.5.3 比较模式 2

比较模式 2 和比较模式 0 类似，只产生中断。不过在给定的定时器周期内，比较模式 2 只发生一次比较匹配，产生一次中断请求。

该模式下，在一个定时器计数周期内首次检测到匹配时，激活中断请求线  $CCyIRQ$ 。此外，即使在该周期内将大于定时器当前值的比较值写入比较寄存器中，所有其它的比较匹配被禁止。只有当定时器溢出时才解禁。首次匹配之后写入比较寄存器中的新值只在下个定时器周期内生效。

#### 19.1.5.4 比较模式 3

比较模式 3 的基本操作和比较模式 2 相同，差别仅在于该模式会影响相关的输出引脚。在一个定时器周期内只可能产生一个比较事件。

该模式下，在一个定时器计数周期内首次检测到匹配时，激活中断请求线 CCyIRQ，并将相关的输出信号置 1。此外，即使在该周期内将大于定时器当前值的比较值写入寄存器中，所有其它的比较匹配被禁止。只有当定时器溢出时才被解禁。首次匹配之后写入比较寄存器中的新值只在下个定时器周期内生效。

溢出信号还将相关的输出信号复位为 0。

必须注意：当所设定的比较值和定时器重载值相等时，比较匹配信号试图置位输出信号，同时定时器溢出试图复位输出信号。为了避免冲突，这种情况下输出信号的状态保持不变。

*注：将比较值从大于定时器的当前值修改为小于定时器的当前值，新值只有在下个定时器周期才能被识别。*

端口输出信号的具体操作请参阅[章节 19.1.8](#)。

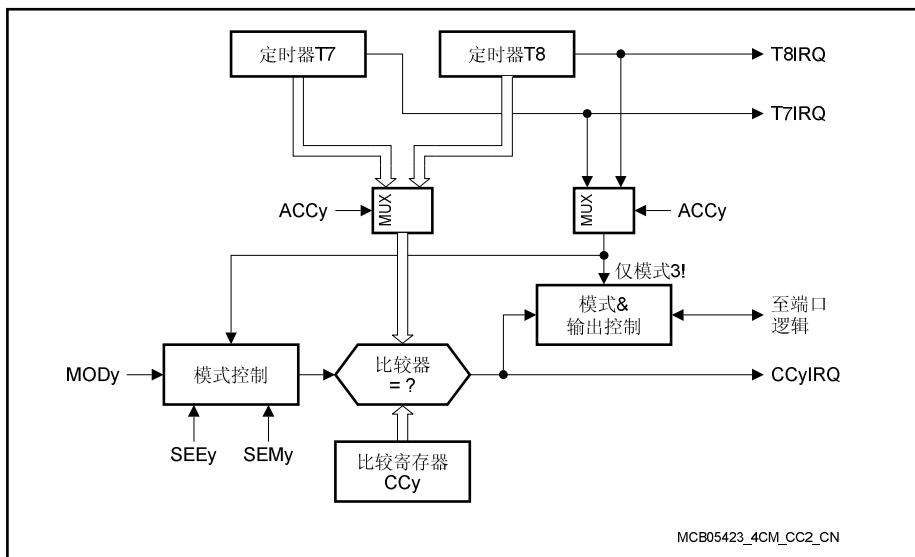


图 19-7 比较模式 2 和 3 框图

*注：比较模式 2 的端口锁存和输出信号不受影响。*

图 19-8 给出比较模式 2 和 3 的时序举例。

各例中定时器的重载值设定为  $\text{FFF9}_\text{H}$ 。每当定时器溢出，从该值重新开始递增计数。

**例 1 中**，寄存器  $\text{CCy}$  的值为  $\text{FFFC}_\text{H}$ 。定时器递增计数到该值时检测到匹配，中断请求线  $\text{CCyIRQ}$  被激活。比较模式 2 的所有操作完毕；而在比较模式 3 下，相关的端口输出被置 1。定时器继续递增计数直至溢出。溢出时端口输出复位为 0。请注意，尽管图中未标出，定时器的溢出信号也激活相关的中断请求线  $\text{TxlIRQ}$ 。若寄存器  $\text{CCy}$  中的比较值不改变，端口输出在下一个定时器周期内再次置位，定时器溢出时再次复位。这是产生脉宽调制 (PWM) 信号的理想操作，此时的软件开销最低。改变比较值相应改变脉冲宽度。

**例 2 中**，在一个定时器周期内首次匹配之后比较操作被禁止。首次  $\text{FFFC}_\text{H}$  匹配之后，产生中断请求并置位端口输出。此外后续匹配被禁止。若此时将新比较值（尽管该值大于定时器的当前计数值）写入寄存器  $\text{CCy}$ ，不会产生中断请求、端口输出不受影响。只有在定时器溢出后，比较逻辑再次被使能，在  $\text{FFF}_\text{H}$  处将检测到下次匹配。可见，由于软件可写入任意的比较值（大于或小于定时器的当前值），这是产生 PWM 的理想操作。该机制确保了新比较值（通常在恰当的中断服务程序中写入比较寄存器）只会在下一个定时器周期内生效。

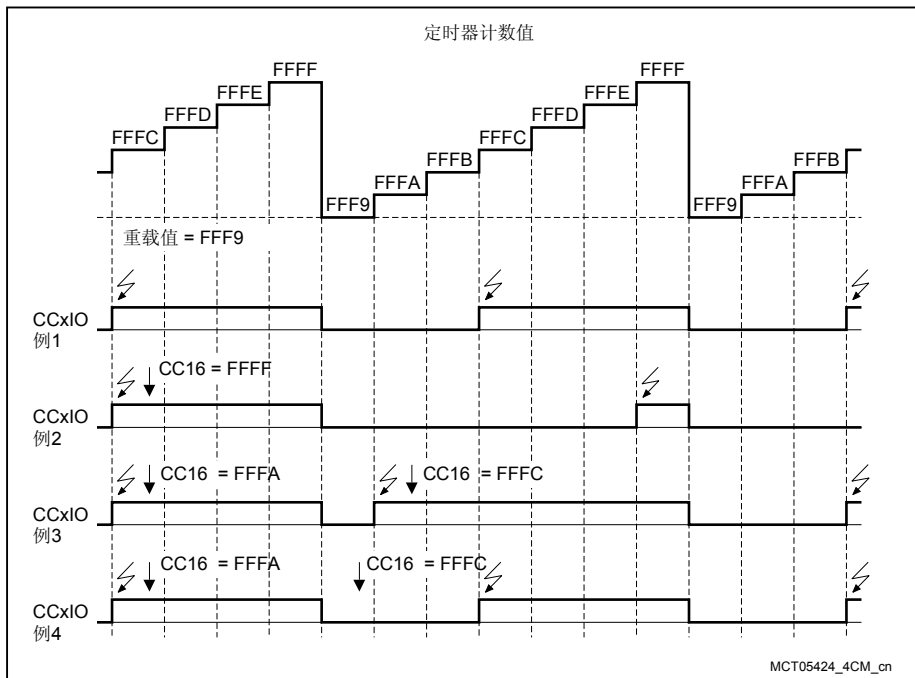


图 19-8 比较模式 2 和 3 时序举例

*注：比较模式 2 只产生中断；模式 3 还会产生输出信号。*

**例 3 中**，进一步图示说明比较匹配操作被禁止。

**例 4 中**，在一个定时器周期内，首次匹配之前将新比较值写入比较寄存器，显然原先设定的比较匹配（ $\text{FFFA}_{\text{H}}$  处）不会发生，将在  $\text{FFFC}_{\text{H}}$  处检测到首次匹配。但必须注意，重新设定比较寄存器的操作是异步的，即寄存器写操作的时刻和定时器的当前值无关。该操作具有一定风险，因为其结果不易被预测。如果软件将新比较值写入寄存器的时刻，定时器已计数到原先设定的比较值  $\text{FFFA}_{\text{H}}$ ，那么在  $\text{FFFA}_{\text{H}}$  处会检测到匹配，新的设定值只能在下个定时器周期生效。

**图 19-9** 给出比较模式 2 和 3 的特例。例 1 所示为比较值和定时器的重载值相等的情况。两种模式均产生中断。模式 3 输出信号不受影响，保持高电平。将比较值设置成等于定时器的重载值，就能产生占空比为 100% 的 PWM 信号。该方法的主要优点在于仍产生比较中断并可用来重载下次的比较值，因此无需特殊处理（见特例 3）。

特例 2、4 和 5 所示为以不同的方式产生占空比为 0% 的 PWM 信号。在特例 2 中，异步重新设定比较值，其值和重载值相等。当前定时器周期结束时产生比较中断，该中断使能由软件设置下次的比较值。该方法的缺点在于新的常规比较值至少在两个定时器周期之后才可生效。和重载值  $\text{FFF9}_{\text{H}}$  的比较匹配将阻止当前定时器周期内的其它比较匹配。特例 4 对此作了进一步说明。

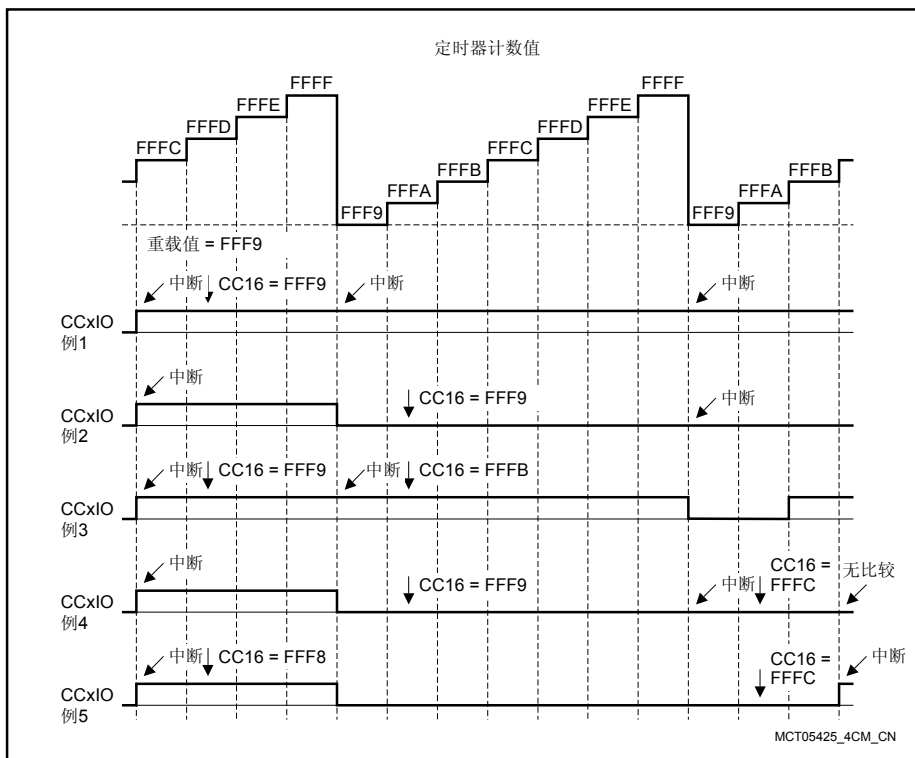


图 19-9 比较模式 2 和 3 的特例

特例 5 就上述问题（新的常规比较值至少在两个定时器周期之后才可生效）给出一种解决方案。比较寄存器中重新载入小于定时器重载值的比较值 FFF8H。定时器永远不会计数到该值，因此不会检测到匹配。首次定时器溢出后输出信号被设置为 0。然而，第二次溢出之后，软件将一个常规比较值重新载入比较寄存器。由于之前未发生比较阻滞（因为未出现比较匹配），新写入的比较值将在当前定时器周期内生效。

### 19.1.6 双寄存器比较模式

双寄存器比较模式可进一步降低软件开销。该模式下，两个比较寄存器共同控制一个输出。可通过 DRM 寄存器、或两个寄存器比较模式的特殊组合选择该模式。

双寄存器比较模式下，CAPCOM 单元的 16 个捕获/比较寄存器被看作两组寄存器，每组由 8 个寄存器组成。低 8 个寄存器构成 bank1，高 8 个寄存器构成 bank2。双

寄存器模式下，bank1 的寄存器和 bank2 的寄存器构成寄存器对。寄存器对中的两个寄存器共同控制和 bank1 寄存器相关的输出引脚。

一个寄存器对与 bank1 和 bank2 寄存器的对应关系和它们所控制的输出引脚列于表 19-3 中。

**表 19-3      双寄存器比较模式下的 CAPCOM2 寄存器对**

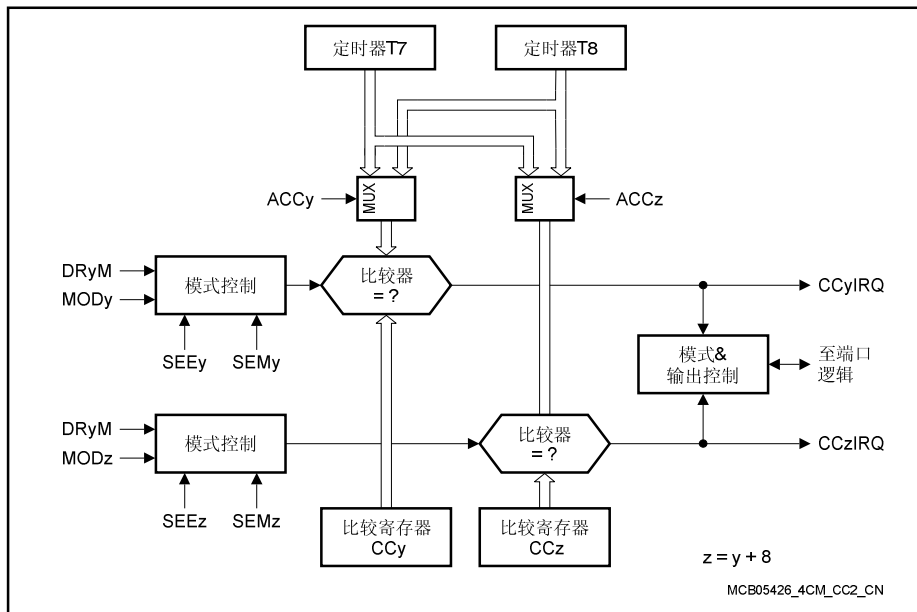
寄存器对		使用的输出引脚	CC2_DRM 中的控制位域
Bank1	Bank2		
CC16	CC24	CC16IO	DR0M
CC17	CC25	CC17IO	DR1M
CC18	CC26	CC18IO	DR2M
CC19	CC27	CC19IO	DR3M
CC20	CC28	CC20IO	DR4M
CC21	CC29	CC21IO	DR5M
CC22	CC30	CC22IO	DR6M
CC23	CC31	CC23IO	DR7M

可分别将每对寄存器设定为双寄存器比较模式。某寄存器对中的两个寄存器的比较模式为某种特定组合时，可设定为双寄存器比较模式：Bank1 的寄存器必须设定为模式 1（影响端口输出）；bank2 的寄存器必须设定为模式 0（只产生中断）。

可通过寄存器 CC2\_DRM 中的位域 DRxM 控制每对寄存器的双寄存器比较模式（即，使能或禁止该模式）。

可分别控制每对寄存器的双寄存器比较模式。

双寄存器比较模式的框图中（图 19-10），bank2 的寄存器用 CCz 表示，bank1 的寄存器用 CCy 表示。



**图 19-10 双寄存器比较模式框图**

当检测到寄存器对（CCy 或 CCz）中的任意一个寄存器发生匹配时，相关中断请求线（CCyIRQ 或 CCzIRQ）被激活，和 bank1 的寄存器 CCy 对应的引脚 CCyIO 被翻转。产生的中断始终和引起匹配的寄存器相对应。

*注：若寄存器对中的两个寄存器 CCy 和 CCz 同时发生匹配，引脚 CCyIO 只翻转一次，不过会产生两个独立的比较中断请求。*

寄存器对中的两个寄存器可分别被指定和 CAPCOM 单元的两个定时器中的任意一个配合工作。由于两个定时器能以不同的精度和频率在不同的模式下工作，这种灵活的分配方式提供了广泛的应用。不过，这可能需要复杂软件算法来处理不同的定时器周期。

*注：信号 CCzIO（双寄存器比较模式中不使用该信号）可用作通用 IO。*



### 19.1.7 CAPCOM 中断

发生捕获或比较事件时，相应捕获/比较寄存器 CCy 的中断请求标志 CCyIR 被自动置位。若中断使能位 CCyIE 被置位，该中断标志可产生中断或触发 PEC 服务请求。

捕获中断可看作具有附加特性（可记录发生触发事件的时刻）的外部中断请求。

每个捕获/比较寄存器（CCy）都有各自的可位寻址中断控制寄存器 CC2\_CCyIC 和中断向量。这些寄存器和所有其它中断控制寄存器的结构相同。这些寄存器与所有其它中断控制寄存器的结构相同。

### 19.1.8 比较输出信号的产生

本节讨论 CAPCOM2 单元和端口逻辑之间的相互作用。图 19-11 详细描述了图 19-5、图 19-7、图 19-10 中所示模块“模式 & 输出控制”的逻辑结构。

每个输出信号被保存到比较输出寄存器 CC2\_OUT 中的对应位。每次产生比较事件时对对应位被更新。这些输出寄存器中的各位和相关端口引脚相连，作为该端口的复用输出。

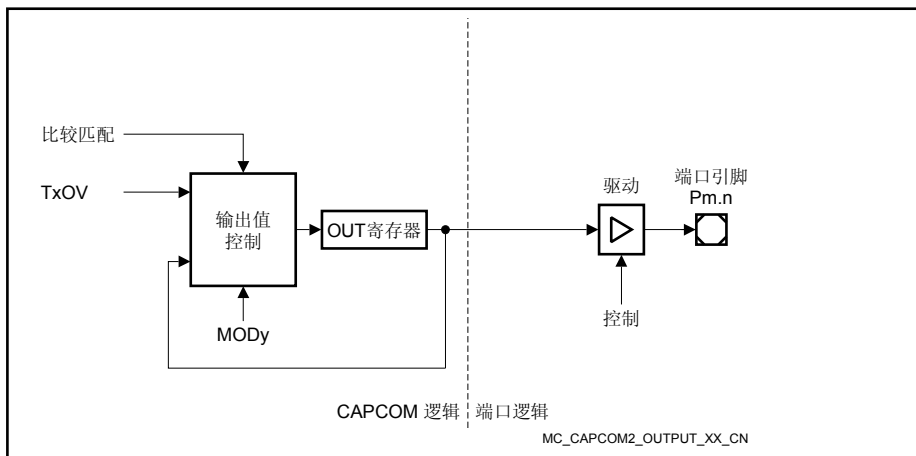


图 19-11 比较模式的端口输出框图

注:只有在比较模式 1 或模式 3，比较输出信号才送到引脚上。

比较事件的输出信号可以为 1、0、当前电平的反向或当前电平。模块“输出值控制”根据比较事件、定时器溢出信号以及输出寄存器位的当前状态决定正确的新电平。对于输出翻转功能（如比较模式 1），输出寄存器位的状态被读出、取反、然后写回。

### 19.1.9 单次事件操作

若某应用只需要一次比较事件（在某段时间间隔内），单次事件操作有助于降低软件开销，无需软件在匹配事件后作出快速响应。

若没有该特性，要实现单次事件操作时，在设定的比较匹配发生后，只能通过软件禁止比较模式；或者将超过定时器计数范围的新比较值写入比较寄存器。因此，通常需要中断服务程序来执行该操作。若定时器周期非常短，则中断响应速度很关键——在定时器再次计数到相同的比较值之前要完成禁止操作。

单次事件操作不需要软件在首次比较匹配之后作出响应。可以在比较事件之前设置好所有的操作，比较事件发生后无需任何操作。硬件负责产生唯一的比较事件，并禁止所有后续的比较匹配。

通过单次事件模式寄存器 **CC2\_SEM** 和单次事件使能寄存器 **CC2\_SEE** 设定该功能。寄存器中的每一位和 **CAPCOM2** 单元中的每个 **CCy** 寄存器一一对应。

将 **CCy** 寄存器设置为单次事件操作：软件首先设定需要的比较操作和比较值，然后置位寄存器 **CC2\_SEM** 中的相应位以使能单次事件模式。最后，置位寄存器 **CC2\_SEE** 中相应的单次事件使能位。

当设定的比较匹配发生时，执行比较模式的所有相应操作。此外，硬件自动禁止所有后续的比较匹配并将寄存器 **CC2\_SEE** 中的使能位复位至 0。只要该使能位被清零，任何比较操作被禁止。设置新事件时，必须首先将该位重新置 1。

### 19.1.10 交错和非交错操作

**CAPCOM2** 单元可工作在两种基本模式：交错模式和非交错模式。通过寄存器 **IOC** 进行模式选择。

交错模式下，**CAPCOM** 的工作周期由 8 个模块时钟周期组成，不同寄存器的比较事件交错输出，也就是说，具有相同比较值的不同寄存器，发生比较匹配时不会同时切换相应的输出，而是按照固定的时延依次切换。该操作有助于降低由同时切换输出引起的噪声和峰值功耗。

非交错模式下，**CAPCOM** 的一个工作周期等于 1 个模块时钟周期，具有相同比较值的比较事件在同一时钟周期切换所有比较输出。该模式下，**CAPCOM** 单元运行速度更快、精度更高（为交错模式的 8 倍）。

#### 交错模式

**图 19-12** 说明 **CAPCOM2** 交错模式的操作。该例中，所有 **CCy** 寄存器均设定工作在比较模式 3。

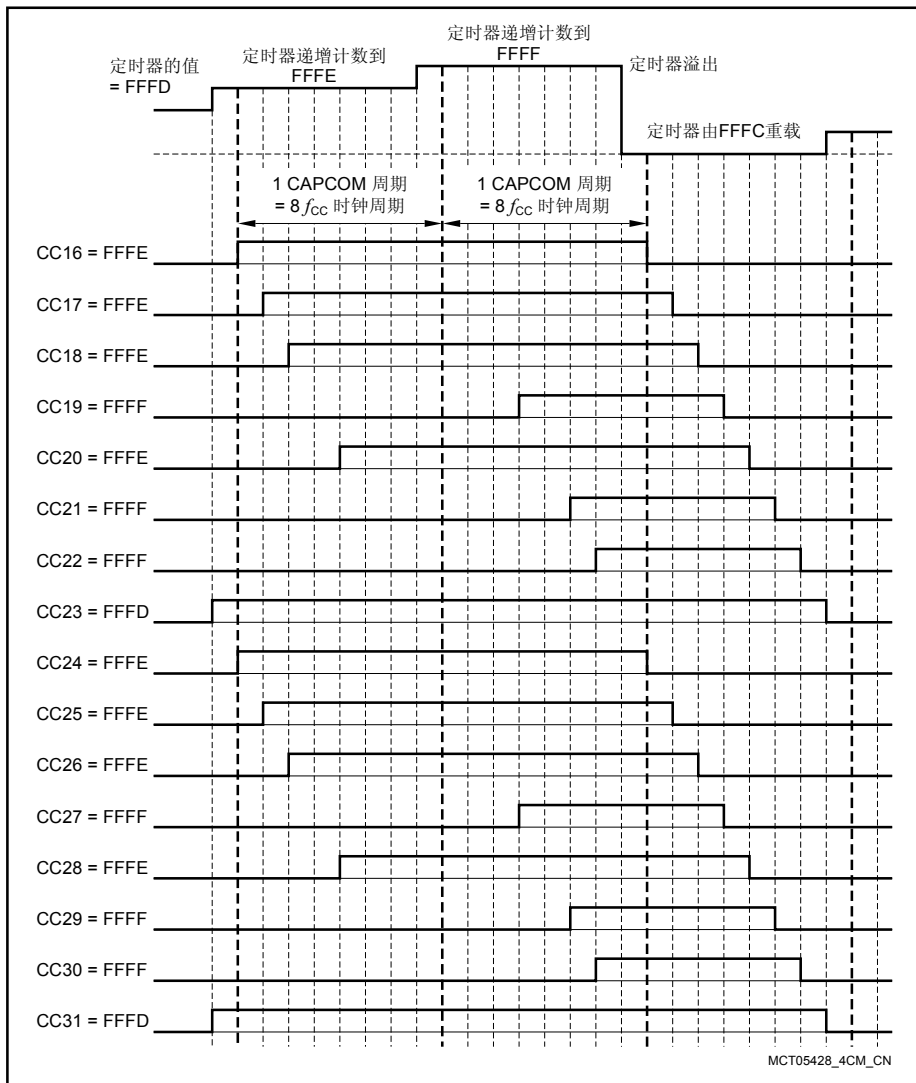
寄存器 **CC16**、**CC17** 和 **CC18** 设定的比较值均为 **FFFE<sub>H</sub>**。定时器递增计数到 **FFFE<sub>H</sub>**，比较器检测到这三个寄存器的比较匹配。比较器匹配后的下一个周期，寄存器 **CC16** 的输出 **CC16IO** 切换到 1。但输出 **CC17IO** 和 **CC18IO** 不会同时切换、而是在随后的两个周期内分别切换到 1。该交错输出模式依次进行下去，直到寄存器 **CC23** 的输

出完成切换。寄存器编号代表以时钟周期为单位的输出信号的时延 – 寄存器 CC23 比寄存器 CC16 的输出切换晚 7 个周期。图示举例中，寄存器 CC23 的比较值为 FFFD<sub>H</sub>。因此，定时器计数到 FFFD<sub>H</sub> 时，在 CAPCOM 工作周期的最后一个时钟周期对 CC23 的输出进行切换。

定时器溢出时，所有比较输出复位到 0（比较模式 3）。从图 19-12 再次看到输出信号复位交错进行。

寄存器 CC24 到 CC31 的输出切换和寄存器 CC16 到 CC23 的输出切换并行进行。事实上，低寄存器组和高寄存器组同时输出。采用这种方式，可确保双寄存器比较模式下寄存器对的两个比较信号同时操作。

*注：上述为交错模式的基本描述，只适用于和引脚相连的通道。*



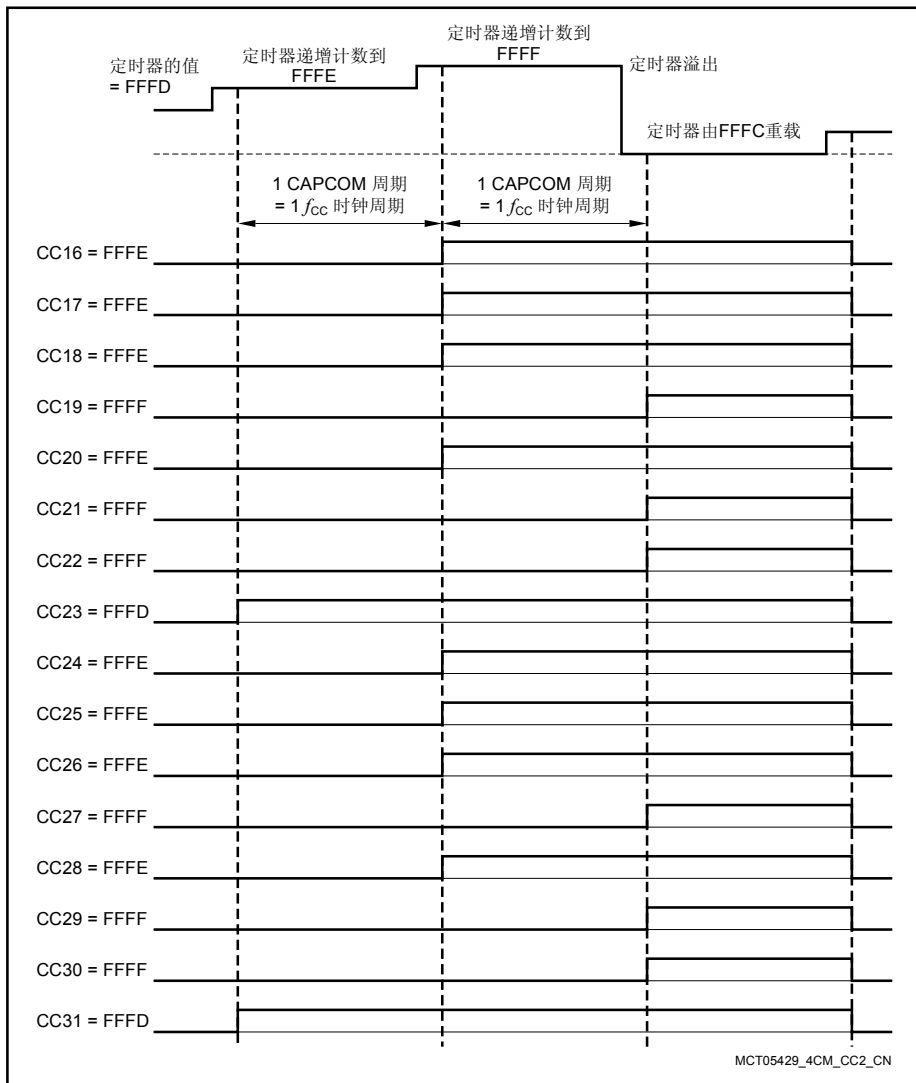
**图 19-12 交错模式操作**

### 非交错模式

为了使 CAPCOM 单元以最快的速度 and 最高的精度运行，可选择非交错模式。该模式下，CAPCOM 的工作周期等于 1 个模块时钟周期。定时器加 1 操作、新计数值和比较值的比较在一个时钟周期之内完成，相应的输出信号在下个时钟周期切换（和下次定时器加 1 和比较同步进行）。

**图 19-13** 说明 CAPCOM2 单元的非交错模式操作。请注意，当定时器溢出时，也需要额外一个时钟周期来切换输出信号。

*注：上述为非交错模式的基本描述，只适用于和引脚相连的通道。*



**图 19-13 非交错模式操作**

### 19.1.11 对外部输入信号的要求

根据模块时钟和基本工作模式（交错或非交错），由 CAPCOM2 逻辑对外部输入信号采样。为了确保外部信号能够被正确识别，它的当前电平值（高或低）必须至少保持一个完整的采样周期。

非交错模式下的采样周期为 1 个模块时钟；交错模式下的采样周期为 8 个模块时钟。为了识别信号跳变，必须对其采样两次。若连续两次采样的电平值不同，即可识别到信号发生了跳变。因此，对外部输入信号采样至少需要两个采样周期，故输入信号的最大频率一定不能高于非交错模式下模块时钟的一半、或者交错模式下模块时钟的 1/16。

外部输入信号参数限制见 [表 19-4](#)。

**表 19-4 CAPCOM2 外部输入信号参数限制**

	非交错模式	交错模式
输入信号频率的最大允许值	$f_{CC}/2$	$f_{CC}/16$
输入信号电平保持稳定的最短时间	$1/f_{CC}$	$8/f_{CC}$

外部信号用作计数或捕获输入时，和其相连的端口引脚必须设置为输入。

*注：进行测试时，用作计数或捕获输入的引脚可设置为输出。可通过软件或其它外设控制对应信号来触发计数或捕获事件。*

为了将比较输出信号送至外部，相关端口引脚必须设置为输出。对于比较输出信号，寄存器 CC2\_OUT 用作端口的复用输出功能。

## 19.2 CAPCOM2 寄存器

下表总结 CAPCOM2 模块的所有寄存器。

表 19-5 CAPCOM2 寄存器总结

寄存器名	描述	地址		复位值
		16 位	8 位	

### 捕获/比较单元 2 (CAPCOM2)

CC2_ID	CAPCOM2 ID 寄存器	FFEE <sub>H</sub>	-	50XX <sub>H</sub>
CC2_M4	CAPCOM2 模式控制寄存器 4	FF22 <sub>H</sub>	91 <sub>H</sub>	0000 <sub>H</sub>
CC2_M5	CAPCOM2 模式控制寄存器 5	FF24 <sub>H</sub>	92 <sub>H</sub>	0000 <sub>H</sub>
CC2_M6	CAPCOM2 模式控制寄存器 6	FF26 <sub>H</sub>	93 <sub>H</sub>	0000 <sub>H</sub>
CC2_M7	CAPCOM2 模式控制寄存器 7	FF28 <sub>H</sub>	94 <sub>H</sub>	0000 <sub>H</sub>
CC2_SEE	CAPCOM2 单次事件使能寄存器	FE2A <sub>H</sub>	15 <sub>H</sub>	0000 <sub>H</sub>
CC2_SEM	CAPCOM2 单次事件模式寄存器	FE28 <sub>H</sub>	14 <sub>H</sub>	0000 <sub>H</sub>
CC2_DRM	CAPCOM2 双寄存器模式寄存器	FF2A <sub>H</sub>	95 <sub>H</sub>	0000 <sub>H</sub>
CC2_OUT	CAPCOM2 输出寄存器	FF2C <sub>H</sub>	96 <sub>H</sub>	0000 <sub>H</sub>
CC2_T7	CAPCOM2 定时器 T7 寄存器	F050 <sub>H</sub>	28 <sub>H</sub>	0000 <sub>H</sub>
CC2_T8	CAPCOM2 定时器 T8 寄存器	F052 <sub>H</sub>	29 <sub>H</sub>	0000 <sub>H</sub>
CC2_T7REL	CAPCOM2 定时器 T7 重载寄存器	F054 <sub>H</sub>	2A <sub>H</sub>	0000 <sub>H</sub>
CC2_T8REL	CAPCOM2 定时器 T8 重载寄存器	F056 <sub>H</sub>	2B <sub>H</sub>	0000 <sub>H</sub>
CC2_T78CON	CAPCOM2 定时器 T7/T8 控制寄存器	F020 <sub>H</sub>	90 <sub>H</sub>	0000 <sub>H</sub>
CC2_IOC	CAPCOM2 I/O 控制寄存器	F066 <sub>H</sub>	33 <sub>H</sub>	0000 <sub>H</sub>
CC2_CC16	CAPCOM2 寄存器 16	FE60 <sub>H</sub>	30 <sub>H</sub>	0000 <sub>H</sub>
CC2_CC17	CAPCOM2 寄存器 17	FE62 <sub>H</sub>	31 <sub>H</sub>	0000 <sub>H</sub>
CC2_CC18	CAPCOM2 寄存器 18	FE64 <sub>H</sub>	32 <sub>H</sub>	0000 <sub>H</sub>
CC2_CC19	CAPCOM2 寄存器 19	FE66 <sub>H</sub>	33 <sub>H</sub>	0000 <sub>H</sub>



寄存器名	描述	地址		复位值
		16 位	8 位	
<b>CC2_CC20</b>	CAPCOM2 寄存器 20	FE68 <sub>H</sub>	34 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC21</b>	CAPCOM2 寄存器 21	FE6A <sub>H</sub>	35 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC22</b>	CAPCOM2 寄存器 22	FE6C <sub>H</sub>	36 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC23</b>	CAPCOM2 寄存器 23	FE6E <sub>H</sub>	37 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC24</b>	CAPCOM2 寄存器 24	FE70 <sub>H</sub>	38 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC25</b>	CAPCOM2 寄存器 25	FE72 <sub>H</sub>	39 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC26</b>	CAPCOM2 寄存器 26	FE74 <sub>H</sub>	3A <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC27</b>	CAPCOM2 寄存器 27	FE76 <sub>H</sub>	3B <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC28</b>	CAPCOM2 寄存器 28	FE78 <sub>H</sub>	3C <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC29</b>	CAPCOM2 寄存器 29	FE7A <sub>H</sub>	3D <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC30</b>	CAPCOM2 寄存器 30	FE7C <sub>H</sub>	3E <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC31</b>	CAPCOM2 寄存器 31	FE7E <sub>H</sub>	3F <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_T7IC</b>	CAPCOM2 定时器 T7 中断控制寄存器	FF6C <sub>H</sub>	BD <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_T8IC</b>	CAPCOM2 定时器 T8 中断控制寄存器	FF6E <sub>H</sub>	BE <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC16IC</b>	CAPCOM2 寄存器 16 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1C0 <sub>H</sub>	B0 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC17IC</b>	CAPCOM2 寄存器 17 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1C2 <sub>H</sub>	B1 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC18IC</b>	CAPCOM2 寄存器 18 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1C4 <sub>H</sub>	B2 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC19IC</b>	CAPCOM2 寄存器 19 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1C6 <sub>H</sub>	B3 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC20IC</b>	CAPCOM2 寄存器 20 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1C8 <sub>H</sub>	B4 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC21IC</b>	CAPCOM2 寄存器 21 中断控制寄存器	F1CA <sub>H</sub>	B5 <sub>H</sub>	0000 <sub>H</sub>

寄存器名	描述	地址		复位值
		16 位	8 位	
	共享的中断节点, 见 ISSR 寄存器			
<b>CC2_CC22IC</b>	CAPCOM2 寄存器 22 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1CC <sub>H</sub>	B6 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC23IC</b>	CAPCOM2 寄存器 23 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1CE <sub>H</sub>	B7 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC24IC</b>	CAPCOM2 寄存器 24 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1D0 <sub>H</sub>	B8 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC25IC</b>	CAPCOM2 寄存器 25 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1D2 <sub>H</sub>	B9 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC26IC</b>	CAPCOM2 寄存器 26 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1D4 <sub>H</sub>	BA <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC27IC</b>	CAPCOM2 寄存器 27 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1D6 <sub>H</sub>	BB <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC28IC</b>	CAPCOM2 寄存器 28 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1D8 <sub>H</sub>	BC <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC29IC</b>	CAPCOM2 寄存器 29 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1DA <sub>H</sub>	C2 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC30IC</b>	CAPCOM2 寄存器 30 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1DC <sub>H</sub>	C6 <sub>H</sub>	0000 <sub>H</sub>
<b>CC2_CC31IC</b>	CAPCOM2 寄存器 31 中断控制寄存器 共享的中断节点, 见 ISSR 寄存器	F1DE <sub>H</sub>	CA <sub>H</sub>	0000 <sub>H</sub>

### 19.2.1 ID 寄存器

CAPCOM2 单元具有特定的只读 ID 寄存器，提供模块类型和版本 ID。

#### CC2\_ID

CAPCOM2 ID 寄存器								MEM (FFEE <sub>H</sub> )				复位值: 50XX <sub>H</sub>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MOD_TYPE								MOD_REV							
r								r							

符号	位序号	类型	功能描述
MOD_REV	[7:0]	r	<b>模块版本编号</b> 该位域定义版本编号。模块版本编号的值从 01 <sub>H</sub> 开始（第一版）。
MOD_TYPE	[15:8]	r	<b>模块 ID 编号</b> 该位域定义模块 ID 编号（50 <sub>H</sub> = CAPCOM2）。

### 19.2.2 定时器 T7/T8 寄存器

#### CC2\_T7

CAPCOM2 定时器 T7 寄存器								ESFR (F050 <sub>H</sub> /28 <sub>H</sub> )				复位值: 0000 <sub>H</sub>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T7															
rw															

符号	位序号	读写类型	功能描述
T7	[15:0]	rw	<b>定时器 T7 当前值</b> 定时器 T7 的当前值

## CC2\_T8

**CAPCOM2 定时器 T8 寄存器**    **ESFR (F052<sub>H</sub>/29<sub>H</sub>)**    **复位值: 0000<sub>H</sub>**

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

<b>T8</b>															
-----------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

rw

符号	位序号	读写类型	功能描述
<b>T8</b>	[15:0]	rw	<b>定时器 T8 当前值</b> 定时器 T8 的当前值

## CC2\_T7REL

**CAPCOM2 定时器 T7 重载寄存器**    **ESFR (F054<sub>H</sub>/2A<sub>H</sub>)**    **复位值: 0000<sub>H</sub>**

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

<b>T7REL</b>															
--------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

rw

符号	位序号	读写类型	功能描述
<b>T7REL</b>	[15:0]	rw	<b>定时器 T7 重载值</b> 定时器 T7 的重载值

## CC2\_T8REL

**CAPCOM2 定时器 T8 重载寄存器**    **ESFR (F056<sub>H</sub>/2B<sub>H</sub>)**    **复位值: 0000<sub>H</sub>**

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

<b>T8REL</b>															
--------------	--	--	--	--	--	--	--	--	--	--	--	--	--	--	--

rw

符号	位序号	读写类型	功能描述
<b>T8REL</b>	[15:0]	rw	<b>定时器 T8 重载值</b> 定时器 T8 的重载值

### 19.2.3 定时器 T7/T8 控制寄存器

#### CC2\_T78CON

定时器 T7/T8 控制寄存器

SFR (FF20H/90H)

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	T8R	T8RSEL	T8M		T8I	-	T7R	T7RSEL	T7M		T7I				
-	rW	rW	rW		rW	-	rW	rW	rW		rW				

符号	位序号	读写类型	功能描述
T7I, T8I	[2:0], [10:8]	rW	<p>定时器/计数器 Tx 输入选择</p> <p><b>定时器模式</b> (TxM = 0) :</p> <p>输入频率 <math>f_{TX} = f_{CC} / 2^{(&lt;TxI&gt;+3)}</math> 或 <math>f_{CC} / 2^{(&lt;TxI&gt;)}</math>, 分别对应交错模式和非交错模式, 参见表 19-1。</p> <p><b>计数器模式</b> (TxM = 1) :</p> <p>000<sub>B</sub> GPT 定时器 T6 上溢/下溢</p> <p>001<sub>B</sub> 引脚 TxIN 的正跳变 (上升沿)</p> <p>010<sub>B</sub> 引脚 TxIN 的负跳变 (下降沿)</p> <p>011<sub>B</sub> 引脚 TxIN 的任意跳变 (上升和下降沿)</p> <p>1XX<sub>B</sub> 保留, 不使用该组合!</p> <p><i>注: 定时器 T8 在计数器模式下的唯一有效设置为 000<sub>B</sub>。其它设置下 T8 停止。</i></p>
T7M, T8M	3, 11	rW	<p><b>定时器/计数器 Tx 模式选择</b></p> <p>0<sub>B</sub> 定时器模式</p> <p>1<sub>B</sub> 计数器模式</p>
T7R, T8R	6, 14	rW	<p><b>定时器/计数器 Tx 运行控制</b></p> <p>0<sub>B</sub> 定时器/计数器 Tx 被禁止</p> <p>1<sub>B</sub> 定时器/计数器 Tx 被使能</p>
T7RSEL	[5:4]	rW	<p><b>定时器 T7 外部运行选择</b></p> <p>位域 T7RSEL 定义可由信号 T7HR 的何种事件来硬件置位 T7R。</p>

符号	位序号	读写类型	功能描述
			00 <sub>B</sub> 禁止外部置位 T7R 01 <sub>B</sub> 如果在 T7HR 上检测到上升沿，则位 T7R 置位。 10 <sub>B</sub> 如果在 T7HR 上检测到下降沿，则位 T7R 置位。 11 <sub>B</sub> 如果在 T7HR 上检测到任意边沿，则位 T7R 置位。
<b>T8RSEL</b>	[13:12]	rw	<b>定时器 T8 外部运行选择</b> 位域 T8RSEL 定义可由信号 T8HR 的何种事件来硬件置位 T8R。 00 <sub>B</sub> 禁止外部置位 T8R 01 <sub>B</sub> 如果在 T8HR 上检测到上升沿，则位 T8R 置位。 10 <sub>B</sub> 如果在 T8HR 上检测到下降沿，则位 T8R 置位。 11 <sub>B</sub> 如果在 T8HR 上检测到任意边沿，则位 T8R 置位。

#### 19.2.4 捕获/比较寄存器

##### CC2\_CCy (y = 16-31)

**CAPCOM2 捕获/比较寄存器 y    SFR (FE60<sub>H</sub>-32+2\*y/30<sub>H</sub>-16+y)    复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CC</b>															
rwh															

符号	位序号	类型	功能描述
<b>CC</b>	[15:0]	rwh	<b>捕获寄存器值</b> 捕获/比较寄存器 y 的当前值。

## 19.2.5 捕获/比较模式寄存器

### CC2\_M4

#### CAPCOM2 模式控制寄存器 4

SFR (FF22<sub>H</sub>/91<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC 19	MOD19			ACC 18	MOD18			ACC 17	MOD17			ACC 16	MOD16		
rw	rw			rw	rw			rw	rw			rw	rw		

符号	位序号	类型	功能描述
<b>ACCy</b> ( y = 16-19)	4*y-61	rw	<b>CAPCOM 寄存器 CCy(y = 16-19)的定时器分配</b> 0 <sub>B</sub> 指定 CCy 和定时器 T7 配合工作 1 <sub>B</sub> 指定 CCy 和定时器 T8 配合工作
<b>MODy</b> ( y = 16-19)	[4*y-62:4*y-64]	rw	<b>CAPCOM 寄存器 CCy(y = 16-19)的模式选择</b> 见表 19-2。

### CC2\_M5

#### CAPCOM2 模式控制寄存器 5

SFR (FF24<sub>H</sub>/92<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC 23	MOD23			ACC 22	MOD22			ACC 21	MOD21			ACC 20	MOD20		
rw	rw			rw	rw			rw	rw			rw	rw		

符号	位序号	类型	功能描述
<b>ACCy</b> ( y = 20-23)	4*y-77	rw	<b>CAPCOM 寄存器 CCy(y = 20-23)的定时器分配</b> 0 <sub>B</sub> 指定 CCy 和定时器 T7 配合工作 1 <sub>B</sub> 指定 CCy 和定时器 T8 配合工作
<b>MODy</b> ( y = 20-23)	[4*y-78:4*y-80]	rw	<b>CAPCOM 寄存器 CCy(y = 20-23)的模式选择</b> 见表 19-2。

### CC2\_M6

**CAPCOM2 模式控制寄存器 6**

**SFR (FF26<sub>H</sub>/93<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC 27		MOD27			ACC 26	MOD26			ACC 25	MOD25			ACC 24	MOD24	
rw		rw			rw	rw			rw	rw			rw	rw	

符号	位序号	类型	功能描述
<b>ACCy</b> ( y = 24-27)	4*y-93	rw	<b>CAPCOM 寄存器 CCy(y = 24-27)的定时器分配</b> 0 <sub>B</sub> 指定 CCy 和定时器 T7 配合工作 1 <sub>B</sub> 指定 CCy 和定时器 T8 配合工作
<b>MODy</b> ( y = 24-27)	[4*y-94:4*y-96]	rw	<b>CAPCOM 寄存器 CCy(y = 24-27)的模式选择</b> 见表 19-2。

### CC2\_M7

**CAPCOM2 模式控制寄存器 7**

**SFR (FF28<sub>H</sub>/94<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ACC 31		MOD31		ACC 30		MOD30		ACC 29		MOD29		ACC 28		MOD28	
rw		rw		rw		rw		rw		rw		rw		rw	

符号	位序号	类型	功能描述
<b>ACCy</b> ( y = 28-31)	4*y-109	rw	<b>CAPCOM 寄存器 CCy(y = 28-31)的定时器分配</b> 0 <sub>B</sub> 指定 CCy 和定时器 T7 配合工作 1 <sub>B</sub> 指定 CCy 和定时器 T8 配合工作
<b>MODy</b> ( y = 28-31)	4*y-110:4*y-112]	rw	<b>CAPCOM 寄存器 CCy(y = 28-31)的模式选择</b> 见表 19-2。



### 19.2.6 比较输出寄存器

CAPCOM2 的比较输出同时服务两个寄存器：端口输出寄存器用于和二进制兼容，另一个独立的寄存器用于增强功能。在端口逻辑中进行 CAPCOM2 比较输出和端口输出锁存的复用选择。

如果 CAPCOM2 设置为比较模式 1 或 3，可在引脚上得到比较输出。

#### CC2\_OUT

##### CAPCOM2 比较输出寄存器

**SFR (FF2C<sub>H</sub>/96<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

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

符号	位序号	类型	功能描述
<b>CCyIO</b> (y = 16-31)	y-16	rwh	<b>通道 y 比较输出</b> 相关端口引脚的复用端口输出

### 19.2.7 双寄存器比较模式寄存器

#### CC2\_DRM

##### CAPCOM2 双寄存器比较模式寄存器

**SFR (FF2A<sub>H</sub>/95<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DR7M	DR6M	DR5M	DR4M	DR3M	DR2M	DR1M	DR0M								
rw	rwh	rwh	rwh	rwh	rwh	rwh	rwh								

符号	位序号	读写类型	功能描述
<b>DRxM</b> (x= 0-7)	[2*x+1:2*x]	rwh	<b>双寄存器 x 比较模式选择</b> 00 <sub>B</sub> 通过比较模式 1 和 0 的组合控制 DRM (兼容模式) 01 <sub>B</sub> 禁止 DRM 操作, 和其它比较模式无关 10 <sub>B</sub> 使能 DRM 操作, 和其它比较模式无关 11 <sub>B</sub> 保留

符号	位序号	读写类型	功能描述
			注：“x”表示 bank 中寄存器对的编号。

## 19.2.8 IOC 寄存器

### CC2\_IOC

#### CAPCOM2 I/O 控制寄存器

**ESFR (F066H/33H)**

**复位值: 0000H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-												0	<b>ST AG</b>	0	-
-												r	rw	rw	-

符号	位序号	类型	功能描述
<b>0</b>	1	rw	保留 读返回 ‘0’，不要置位该位。
<b>STAG</b>	2	rw	交错模式控制 0 <sub>B</sub> CAPCOM2 工作在交错模式 1 <sub>B</sub> CAPCOM2 工作在非交错模式
<b>0</b>	3	r	保留 读返回 ‘0’，不要置位该位。
<b>0</b>	[15:4]	r	保留 读返回 ‘0’。

## 19.2.9 单次事件模式寄存器

### CC2\_SEM

**CAPCOM2 单次事件模式控制寄存器 SFR (FE28H/14H) 复位值: 0000H**

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

符号	位序号	类型	功能描述
<b>SEMy</b> (y = 16-31)	y-16	rw	单次事件模式控制 0 <sub>B</sub> 通道 y 的单次事件模式被禁止 1 <sub>B</sub> 通道 y 的单次事件模式被使能

### CC2\_SEE

**CAPCOM2 单次事件使能寄存器 SFR (FE2AH/15H) 复位值: 0000H**

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

符号	位序号	类型	功能描述
<b>SEEy</b> (y = 16-31)	y-16	rwh	单次事件使能控制 0 <sub>B</sub> 通道 y 的单次事件被禁止 1 <sub>B</sub> 通道 y 的单次事件被使能 <i>注: 事件发生后由硬件清除该位。</i>

## 19.2.10 其它寄存器

### CC2\_KSCCFG

CAPCOM2 内核状态配置寄存器

SFR (FE24H/12H)

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	0	COMCFG	BP SUM	0	SUMCFG	BP NOM	0	NOMCFG	0	BP MOD EN	MOD EN				
w	r	rw	w	r	rw	w	r	rw	r	w	rw				

符号	位序号	类型	功能描述
<b>MODEN</b>	0	rw	<b>模块使能</b> 该位使能模块内核时钟和模块功能。 <b>0<sub>B</sub></b> 模块关闭。 该模块不响应模块控制操作，模块时钟立即关闭（不需要停止条件）。模块不能响应读访问并且忽略写访问。 <b>1<sub>B</sub></b> 模块开启，可以正常工作。 向 MODEN 写 1 之后，访问其它 CAPCOM2 寄存器之前，推荐读取寄存器 KSCCFG 以避免流水线对控制模块的影响。 <i>注：应用复位可复位该位。</i>
<b>BPMODEN</b>	1	w	<b>MODEN 的位保护</b> 该位使能对 MODEN 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 <b>0<sub>B</sub></b> MODEN 不改变 <b>1<sub>B</sub></b> 由写入值更新 MODEN。 <i>注：应用复位可复位该位。</i>
<b>NOMCFG</b>	[5:4]	rw	<b>正常操作模式配置</b> 该位域定义适用于正常操作模式的内核模式。 <b>0X<sub>B</sub></b> 模块开启。 <b>1X<sub>B</sub></b> 模块关闭 仅在 CR = 00 或 11 时，该位域才起效。

符号	位序号	类型	功能描述
			<i>注：应用复位可复位该位域。</i>
<b>BPNO M</b>	7	w	<b>NOMCFG 的位保护</b> 该位使能对 NOMCFG 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 <sub>B</sub> NOMCFG 不改变 1 <sub>B</sub> 由写入值更新 NOMCFG。 <i>注：应用复位可复位该位。</i>
<b>SUMCFG</b>	[9:8]	rw	<b>挂起模式配置</b> 该位域定义适用于挂起模式的内核模式。 0X <sub>B</sub> 模块开启。 1X <sub>B</sub> 模块关闭。 仅在 CR = 01 时，该位域才起效。 <i>注：调试复位可复位该位域。</i>
<b>BPSUM</b>	11	w	<b>SUMCFG 的位保护</b> 该位使能对 SUMCFG 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 <sub>B</sub> SUMCFG 不改变 1 <sub>B</sub> 由写入值更新 SUMCFG。 <i>注：调试复位可复位该位。</i>
<b>COMCFG</b>	[13:12]	rw	<b>时钟关闭模式配置</b> 该位域定义适用于时钟关闭模式的内核模式。 0X <sub>B</sub> 模块开启。 1X <sub>B</sub> 模块关闭。 仅在 CR = 10 时，该位域才起效。 <i>注：应用复位可复位该位域。</i>
<b>BPCOM</b>	15	w	<b>COMCFG 的位保护</b> 该位使能对 COMCFG 的写操作。读该位始终返回 0。仅在写访问周期内该位有效。 0 <sub>B</sub> COMCFG 不改变 1 <sub>B</sub> 由写入值更新 COMCFG。

符号	位序号	类型	功能描述
			<i>注：应用复位可复位该位。</i>
<b>0</b>	[3:2], 6,10, 14	r	<b>保留；</b> 读操作返回 0；应写入 0。

## 19.3 模块实现

本节描述 CAPCOM 单元和与其周围资源的连接。

### 19.3.1 CAPCOM2 单元接口

CAPCOM2 单元以不同的方式和周围资源接口，这些连接见[表 19-6](#)。

#### 内部连接

GPT2 定时器 T6 的上溢/下溢信号 T6OFL 连接至 CAPCOM2 单元输入 T6OUF，为 CAPCOM 定时器提供了一个可选时钟源。

由 SCU\_SYSCON1.GLCCST 位支持同步启动 CAPCOM2 定时器。

比较输出信号可触发 A/D 转换，触发串行传送（USIC），并可为外部请求单元（ERU）产生请求信号。

CAPCOM2 单元的 18 条中断请求线送至中断控制模块。通道中断请求线和其它中断请求源共用中断节点。使用 SCU\_ISSR 完成中断源的选择。

CAPCOM2 模块的输入时钟等于 XE16xyM 系统时钟，因此  $f_{CC} = f_{SYS}$ 。

#### 外部连接

CAPCOM2 单元的 16 个（100 引脚封装器件中为 12 个）捕获/比较信号和 XE16xyM 的输入/输出端口相连。根据设定的方向，这些端口可接受来自外部系统的捕获触发信号（输入）、或将比较输出信号送到外部电路（输出）。

*注：捕获触发信号也可取自输出引脚。例如，在这种情况下，软件可产生触发沿。*

定时器 T7 的时钟源可来自外部信号。

**表 19-6 XE16xyM 内的 CAPCOM2 信号相互连接**

信号	来自/送至 哪个模块	I/O 至 CAPCOM2	可用作...
T7IN	P5.9	I	来自端口的定时器 T7 输入
T6OUF	T6OFL (GPT12)	I	GPT12 定时器 T6 上溢
T7HR	GLCCST (SCU)	I	全局 CAPCOM 启动
T8HR	GLCCST (SCU)	I	全局 CAPCOM 启动
CC16IO	P2.3	I	捕获/比较输入
	P2.3 ADC0_REQTR0A	O	捕获/比较输出 ADC0 请求触发
CC17IO	P2.4	I	捕获/比较输入
	P2.4 ADC0_REQTR1A	O	捕获/比较输出 ADC0 请求触发
CC18IO	P2.5	I	捕获/比较输入
	P2.5 ADC0_REQTR2A	O	捕获/比较/输出 ADC0 请求触发
CC19IO	P2.6	I	捕获/比较输入
	P2.6	O	捕获/比较 输出
CC20IO	P2.7	I	捕获/比较输入
	P2.7	O	捕获/比较输出
CC21IO	P2.8	I	捕获/比较输入
	P2.8	O	捕获/比较输出
CC22IO	P2.9	I	捕获/比较输入
	P2.9	O	捕获/比较输出
CC23IO	P2.10	I	捕获/比较输入
	P2.10	O	捕获/比较输出

信号	来自/送至 哪个模块	I/O 至 CAPCOM2	可用作...
CC24IO	P4.0	I	捕获/比较输入
	P4.0 U0C0_DX2E ADC1_REQTR0A	O	捕获/比较输出 USIC0 通道 0 时隙 ADC1 请求触发
CC25IO	P4.1	I	捕获/比较输入
	P4.1 U1C0_DX2E ADC1_REQTR1A	O	捕获/比较输出 USIC1 通道 0 时隙 ADC1 请求触发
CC26IO	P4.2	I	捕获/比较输入
	P4.2 U2C0_DX2E ADC1_REQTR2A	O	捕获/比较输出 USIC2 通道 0 时隙 ADC1 请求触发
CC27IO	P4.3	I	捕获/比较输入
CC27IO	P4.3 U3C0_DX2E	O	捕获/比较输出 USIC3 通道 0 时隙



## 20 捕获/比较单元 6 (CCU6)

CCU6 为具有特定应用模式的高精度 16 位捕获和比较单元，主要用于 AC 电机控制应用。CCU6 的特殊工作模式支持使用霍尔传感器或反电动势检测控制方案的无刷直流电机控制。此外，CCU6 还支持块切换和多相电机控制算法。

CCU6 还提供可同步启动多个定时器的输入，这是具有多个 CCU6 模块的器件的重要特性。

本章结构如下：

- 简介 (见 [章节 20.1](#))  
包括寄存器概览 (见 [章节 20.1.3](#))
- 定时器 T12 的操作 (见 [章节 20.2](#))  
包括与 T12 相关的寄存器 (见 [章节 20.2.8](#))  
和捕获/比较控制寄存器 (见 [章节 20.2.9](#))
- 定时器 T13 的操作 (见 [章节 20.3](#))  
包括与 T13 相关的寄存器 (见 [章节 20.3.6](#))
- 强制中断处理 (见 [章节 20.4](#))
- 多通道模式 (见 [章节 20.5](#))
- 霍尔传感器模式 (见 [章节 20.6](#))
- 调制控制寄存器 (见 [章节 20.7](#))
- 中断处理 (见 [章节 20.8](#))  
包括中断寄存器 (见 [章节 20.8.2](#))
- 一般模块操作 (见 [章节 20.9](#))  
包括通用寄存器 (见 [章节 20.9.3](#))
- 模块实现 (见 [章节 20.10](#))

### 20.1 简介

CCU6 单元由带有三路捕获/比较通道的定时器 T12 模块和带有一路比较通道的定时器 T13 模块组成。T12 的各通道既能独立产生 PWM 信号或接受捕获信号，各通道也可共同产生驱动交流电机或逆变器的控制信号序列。

CCU6 中丰富的状态位、参数的同步更新（通过映射寄存器实现）和中断请求信号的灵活产生为用户提供了高效的软件控制方式。

*注：捕获/比较模块自身被称为 CCU6（捕获/比较单元 6）。*

*模块内的捕获/比较通道被称为 CC6x。*

### 20.1.1 特性概述

本节给出 CCU6 各组成模块及其主要特性。

#### 定时器 T12 模块特性:

- 3 路捕获/比较通道，各路可单独用作捕获或比较通道
- 支持三相 PWM 产生（6 路输出，每相两路信号分别用于控制上桥臂或下桥臂开关）
- 16 位精度，最大计数频率 = 外设时钟频率
- 各路通道的死区时间控制，以避免功率器件短路
- T12 寄存器同步更新
- 支持中心对齐和边沿对齐 PWM 产生
- 支持单次模式
- 可由外部事件启动定时器
- 能够对外部事件计数
- 支持多个中断请求源
- 支持类磁滞控制模式

#### 定时器 T13 模块特性:

- 一个独立的比较通道输出
- 16 位精度，最大计数频率 = 外设时钟频率
- 可与 T12 同步
- 周期匹配和比较匹配时产生中断
- 支持单次模式
- 可由外部事件启动定时器
- 能够对外部事件计数

#### 附加特性:

- 支持用于驱动无刷直流电机的块切换
- 通过霍尔传感器序列进行位置检测
- 支持位置输入信号的噪声滤波
- 块切换的自动转速测量和切换控制
- 综合错误处理
- 由外部信号（ $\overline{\text{CTRAP}}$ ）控制快速急停，无需 CPU 干预

- 多通道交流驱动的控制模式
- 输出电平可选，以配合功率器件

### 20.1.2 框图

定时器 T12 的三路通道可工作在捕获和/或比较模式。三个通道的工作模式还可组合起来（即一个通道工作在比较模式，而另一个通道工作在捕获模式）。定时器 T13 只能工作在比较模式。由多通道控制单元产生输出序列（可由 T12 和/或 T13 对其进行调制）。信号的调制源可选，并可组合使用。

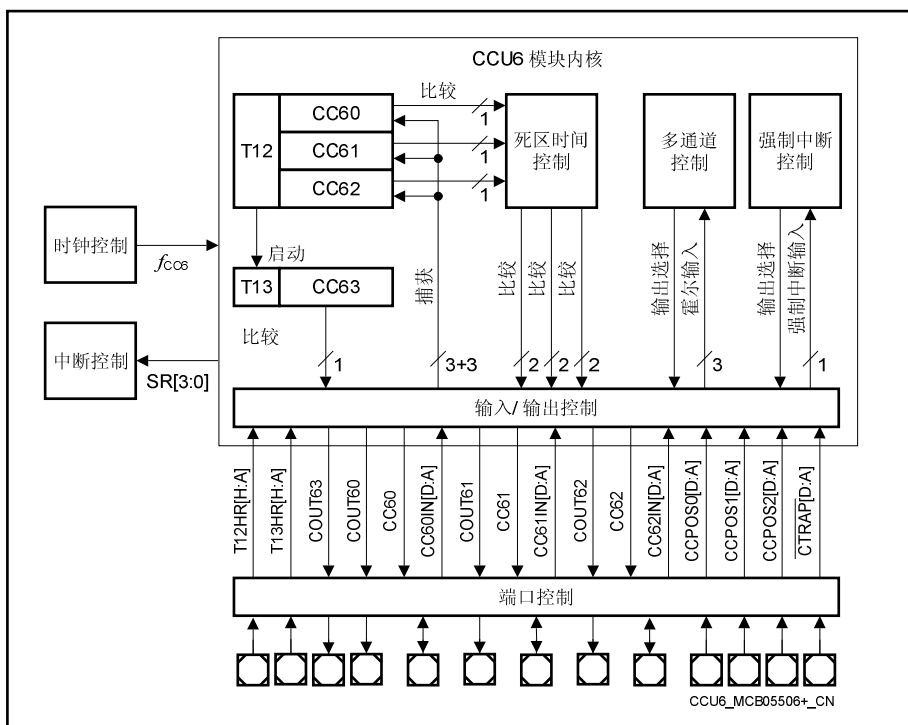


图 20-1 CCU6 框图

### 20.1.3 寄存器概述

为了统一起见，给不同 CCU6 模块的寄存器名称增加前缀“CCU6x\_”以确定寄存器所在模块。在此命名惯例中，x 代表模块编号。

**表 20-1** 给出编程设置 CCU6 模块所需的全部寄存器。该表总结了 CCU6 内核寄存器，给出其偏移地址以及复位值。

该模块中不能使用 8 位短地址。

T12 相关寄存器	捕获/比较 控制寄存器	中断状态/ 控制寄存器	通用寄存器
<b>T12</b>	<b>CMPSTAT</b>	<b>IS</b>	<b>KSCFG</b>
<b>T12PR</b>	<b>CMPMODIF</b>	<b>ISS</b>	<b>KSCSR</b>
<b>T12DTC</b>	<b>T12MSEL</b>	<b>ISR</b>	<b>PISELH</b>
<b>CC60R</b>	<b>TCTR0</b>	<b>INP</b>	<b>PISELL</b>
<b>CC60SR</b>	<b>TCTR2</b>	<b>IEN</b>	<b>ID</b>
<b>CC61R</b>	<b>TCTR4</b>	<b>0IC</b>	
<b>CC61SR</b>		<b>1IC</b>	
<b>CC62R</b>		<b>2IC</b>	
<b>CC62SR</b>		<b>3IC</b>	
	调制控制寄存器		
	<b>MODCTR</b>		
	<b>TRPCTR</b>		
	<b>PSLR</b>		
	<b>MCMCTR</b>		
	<b>MCMOUTS</b>		
	<b>MCMOUT</b>		
T13 相关寄存器			
<b>T13</b>			
<b>T13PR</b>			
<b>CC63R</b>			
<b>CC63SR</b>			

CCU6\_regs2\_cn

**图 20-2 CCU6 寄存器**

**表 20-1 CCU6 模块寄存器总结**

寄存器缩写名	描述	偏移	复位值	见页
<b>通用寄存器</b>				
<b>ID</b>	模块 ID 寄存器	08 <sub>H</sub>	54XX <sub>H</sub>	<a href="#">页 20-101</a>
<b>PISELL</b>	模块端口输入选择寄存器	04 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-102</a>
<b>PISELH</b>	模块端口输入选择寄存器	06 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-104</a>

**捕获/比较单元 6 (CCU6)**

寄存器缩写名	描述	偏移	复位值	见页
<b>KSCCFG</b>	内核状态配置寄存器	00 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-106</a>
<b>KSCSR</b>	内核状态控制敏感寄存器	0E <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-108</a>

**定时器 T12 相关寄存器**

<b>T12</b>	定时器 T12 计数寄存器	10 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-31</a>
<b>T12PR</b>	定时器 T12 周期寄存器	12 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-31</a>
<b>T12DTC</b>	定时器 T12 死区时间控制寄存器	14 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-34</a>
<b>CC60R</b>	通道 CC60 捕获/比较寄存器	18 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-32</a>
<b>CC61R</b>	通道 CC61 捕获/比较寄存器	1A <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-32</a>
<b>CC62R</b>	通道 CC62 捕获/比较寄存器	1C <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-32</a>
<b>CC60SR</b>	通道 CC60 捕获/比较映射寄存器	20 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-33</a>
<b>CC61SR</b>	通道 CC61 捕获/比较映射寄存器	22 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-33</a>
<b>CC62SR</b>	通道 CC62 捕获/比较映射寄存器	24 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-33</a>

**捕获/比较控制寄存器**

<b>CMPSTAT</b>	比较状态寄存器	28 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-35</a>
<b>CMPMODIF</b>	比较状态修改寄存器	2A <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-37</a>
<b>T12MSEL</b>	T12 捕获/比较模式选择寄存器	46 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-38</a>
<b>TCTR0</b>	定时器控制寄存器 0	2C <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-39</a>
<b>TCTR2</b>	定时器控制寄存器 2	2E <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-42</a>
<b>TCTR4</b>	定时器控制寄存器 4	26 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-45</a>

**定时器 T13 相关寄存器**

<b>T13</b>	定时器 T13 计数寄存器	30 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-59</a>
<b>T13PR</b>	定时器 T13 周期寄存器	32 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-60</a>
<b>CC63R</b>	定时器 T13 比较寄存器	34 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-60</a>

**捕获/比较单元 6 (CCU6)**

寄存器缩写名	描述	偏移	复位值	见页
<b>CC63SR</b>	定时器 T13 比较映射寄存器	36 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-61</a>

**调制控制寄存器**

<b>MODCTR</b>	调制控制寄存器	40 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-74</a>
<b>TRPCTR</b>	强制中断控制寄存器	42 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-76</a>
<b>PSLR</b>	被动态电平寄存器	44 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-79</a>
<b>MCMOUTS</b>	多通道模式输出映射寄存器	4A <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-82</a>
<b>MCMOUT</b>	多通道模式输出寄存器	4C <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-83</a>
<b>MCMCTR</b>	多通道模式控制寄存器	4E <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-80</a>

**中断状态和节点寄存器**

<b>IS</b>	中断状态寄存器	50 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-87</a>
<b>ISS</b>	中断状态置位寄存器	52 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-90</a>
<b>ISR</b>	中断状态复位寄存器	54 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-92</a>
<b>INP</b>	中断节点指针寄存器	56 <sub>H</sub>	3940 <sub>H</sub>	<a href="#">页 20-96</a>
<b>IEN</b>	中断使能寄存器	58 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 20-93</a>

*注：写访问地址范围内的地址（由同一个片选信号所涵盖的），但是该地址未在模块中明确给出的情况下，模块对该写访问操作不予响应。该规则同样适用于读访问。读访问另一个地址的情况下，模块不予响应。*

## 20.2 定时器 T12 模块

定时器 T12 是产生三相 PWM 信号的主要单元。一个 16 位计数器通过比较器和 3 路通道寄存器相连，当计数器的值和某个通道寄存器的值匹配时，输出匹配信号。T12 模块提供了多种控制功能，以便适用于不同应用的需求。

除可产生三相 PWM 信号之外，T12 模块还支持单通道比较和捕获功能、以及死区时间控制和类滞滞比较模式。

本节提供下述内容：

- T12 模块概述（见 [章节 20.2.1](#)）
- 计数策略（见 [章节 20.2.2](#)）

- 比较模式 (见 [章节 20.2.3](#))
- 比较模式输出通路 (见 [章节 20.2.4](#))
- 捕获模式 (见 [章节 20.2.5](#))
- 映射传送 (见 [章节 20.2.6](#))
- T12 工作模式选择 (见 [章节 20.2.7](#))
- T12 计数寄存器描述 (见 [章节 20.2.8](#))

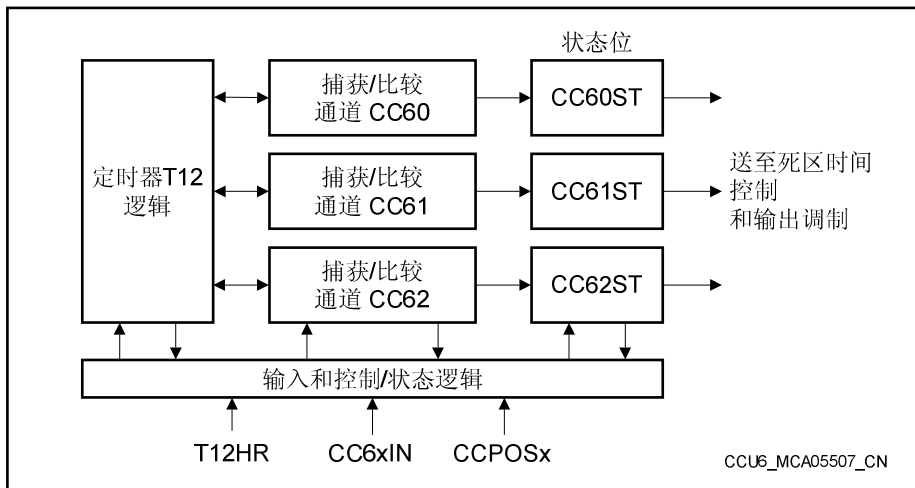


图 20-3 定时器 T12 模块总览

### 20.2.1 T12 概述

图 20-4 为定时器 T12 的内部结构框图。由寄存器 **TCTR0**、**TCTR2**、**TCTR4** 和 **PISELL** 中的位控制 T12 模块的功能。

定时器 T12 的输入时钟 ( $f_{T12}$ ) 来自经过预分频处理的模块时钟  $f_{CC6}$  (预分频因子可编程设定) 和 1/256 分频 (可选), 或者来自输入信号 T12HR。分频因子分别由位域 T12CLK 和 T12PRE 选择 (见 [表 20-2](#))。T12 可递增或递减计数, 计数方向取决于所选择的工作模式。计数方向标志 CDIR 指示当前的计数方向。

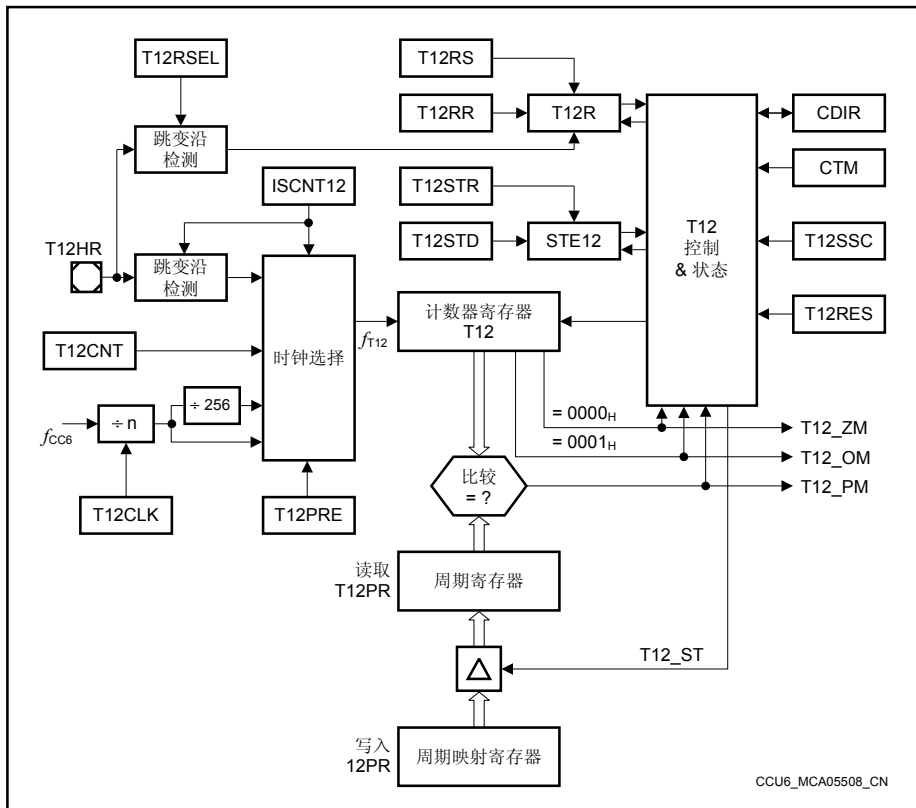


图 20-4 定时器 T12 的逻辑和周期比较器

T12 计数寄存器 **T12** 通过比较器和周期寄存器 **T12PR** 相连，该寄存器的内容决定了 T12 的最大计数值。

边沿对齐模式下，T12 计数到由 T12PR 决定的周期值后复位为 0000<sub>H</sub>。中心对齐模式下，T12 计数到周期值后，计数方向从“递增”变为“递减”（请注意，在该模式下，T12 递增计数到周期值加 1 后转为递减计数）。上述两种情况下，均产生信号 T12<sub>PM</sub>（T12 周期匹配）。周期寄存器的新周期值从其映射周期寄存器中获得。

读取 T12PR 得到比较器上的当前周期值，对映射周期寄存器进行写访问则给出另一个周期值。“T12 映射传送”控制信号 T12<sub>ST</sub> 控制将新周期值从映射寄存器传送到 T12PR 中的操作（见 [章节 20.2.6](#)），该信号的产生取决于当前的工作模式和映射传送使能位 STE12 的状态。CCU6 单元提供了周期值、以及（和产生 PWM 信号相关的）其它数值的映射寄存器，便于软件同步更新所有相关参数。



两个信号 T12\_ZM 和 T12\_OM 分别指示计数器的值是否等于 0000<sub>H</sub> (T12\_ZM = 0-匹配) 或 0001<sub>H</sub> (T12\_OM = 1-匹配)。这些信号控制 T12 的计数和切换操作。

**T12 基本工作模式：**边沿对齐模式（见图 20-5）或中心对齐模式（见图 20-6），由位 CTM 进行选择。单次模式控制位 T12SSC 控制定时器在当前计数周期结束后自动停止运行（见图 20-7 和图 20-8）。

运行位 T12R 控制 T12 定时器的启动和停止，可由寄存器 TCTR4 中的位修改。T12R 可由软件设置，置位 T12RS 则对 T12R 置位，置位 T12RR 则对 T12R 清零，T12R 还可由输入信号 T12HR (TCTR2.T12RESL) 的可选信号沿置位，或者根据预先选定的条件由硬件复位。

当设定的 T12 周期值为 0 时，定时器 T12 的运行位 T12R 一定不能置位。定时器 T12 可由控制位 T12RES 清零。置位“只写”控制位 T12RES 只清除定时器的内容，不会产生其它后续影响，如不会终止定时器工作。

由位 STE12 使能 T12 映射传送控制信号 T12\_ST 的产生，该控制位可通过相关置位控制位 T12STR，或复位控制位 T12STD 间接置位/复位。

定时器 T12 工作时，写入计数寄存器 T12 的操作无效。若 T12 停止工作且死区时间计数器为 0，写入寄存器 T12 的值立刻生效。

**20.2.2 T12 计数策略**

本节描述 T12 的时钟和计数功能。

**20.2.2.1 时钟选择**

**定时器模式** (**PISELH.ISCNT12 = 00<sub>B</sub>**) 下, 定时器 T12 的输入时钟  $f_{T12}$  来自经过预分频处理的模块时钟  $f_{CC6}$  (预分频因子可编程设定) 和 1/256 分频 (可选)。所得预分频因子列于**表 20-2**中。定时器 T12 不工作时 (**TCTR0.T12R = 0**) , 预分频器复位, 从而保证产生可重复的时序和延迟。

**表 20-2      定时器 T12 输入时钟选择**

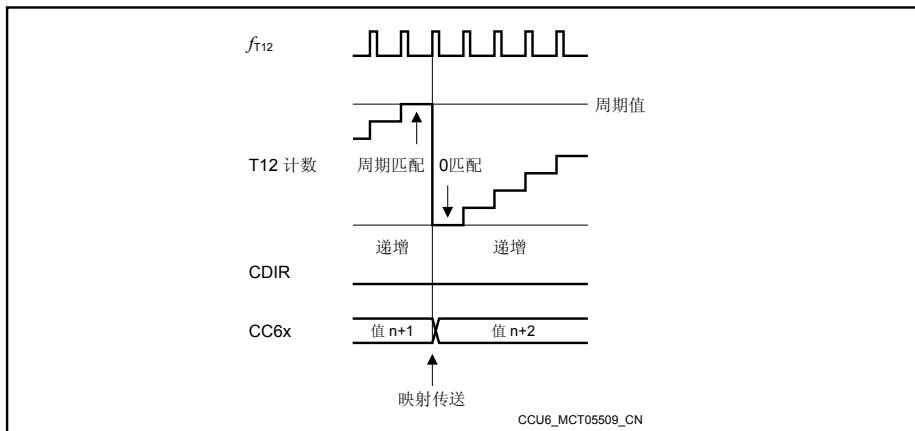
<b>T12CLK</b>	<b>输入时钟 <math>f_{T12}</math> 预分频器关闭 (T12PRE = 0)</b>	<b>输入时钟 <math>f_{T12}</math> 预分频器开启 (T12PRE = 1)</b>
000 <sub>B</sub>	$f_{CC6}$	$f_{CC6}/256$
001 <sub>B</sub>	$f_{CC6}/2$	$f_{CC6}/512$
010 <sub>B</sub>	$f_{CC6}/4$	$f_{CC6}/1024$
011 <sub>B</sub>	$f_{CC6}/8$	$f_{CC6}/2048$
100 <sub>B</sub>	$f_{CC6}/16$	$f_{CC6}/4096$
101 <sub>B</sub>	$f_{CC6}/32$	$f_{CC6}/8192$
110 <sub>B</sub>	$f_{CC6}/64$	$f_{CC6}/16384$
111 <sub>B</sub>	$f_{CC6}/128$	$f_{CC6}/32768$

**计数器模式**下, 满足以下条件, 则定时器 T12 计数一步:

- 如果向 **TCTR4.T12CNT** 写入 1 且 **PISELH.ISCNT12 = 01<sub>B</sub>**
- 如果检测到输入信号 T12HR 的上升沿且 **PISELH.ISCNT12 = 10<sub>B</sub>**
- 如果检测到输入信号 T12HR 的下降沿且 **PISELH.ISCNT12 = 11<sub>B</sub>**

### 20.2.2.2 边沿对齐/中心对齐模式

**边沿对齐模式下** ( $CTM = 0$ )，定时器 T12 始终递增计数 ( $CDIR = 0$ )。计数至周期寄存器给出的周期值时 (周期匹配 T12\_PM)，下一个计数脉冲来到时，T12 的值被清除 (锯齿形)。



**图 20-5 边沿对齐模式下 T12 的操作**

边沿对齐模式下，定时器实际周期为：

$$T12_{PER} = \langle \text{周期寄存器的值} \rangle + 1; \text{以 } T12 \text{ 时钟频率 } (f_{T12}) \text{ 计数} \quad (20.1)$$

在**中心对齐模式** ( $CTM = 1$ ) 下，定时器 T12 递增或递减计数 (三角形)。递增计数 ( $CDIR = 0$ ) 至周期寄存器给出的值时 (周期匹配 T12\_PM)，下一个计数脉冲来到时，计数方向控制位 CDIR 变为递减 ( $CDIR = 1$ )。

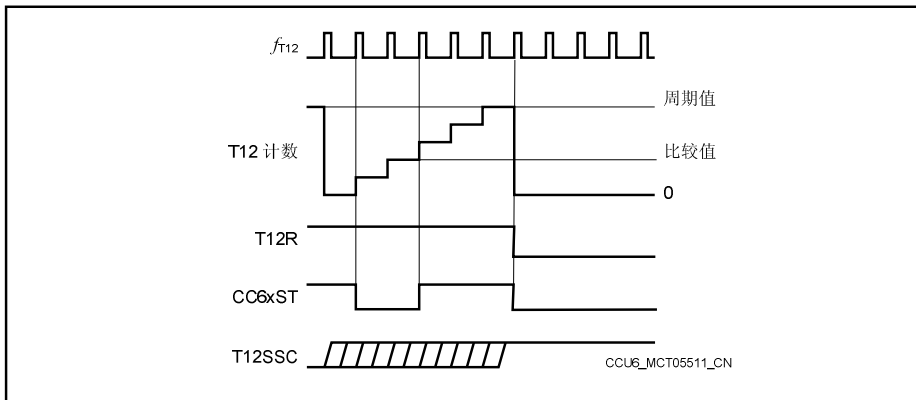
当递减计数至 0001<sub>H</sub> (1-匹配 T12\_OM)，下一个计数脉冲到来时，计数方向控制位变为递增计数。

中心对齐模式下，定时器实际周期为：

$$T12_{PER} = (\langle \text{周期寄存器的值} \rangle + 1) \times 2; \text{以 } T12 \text{ 时钟频率 } (f_{T12}) \text{ 计数} \quad (20.2)$$

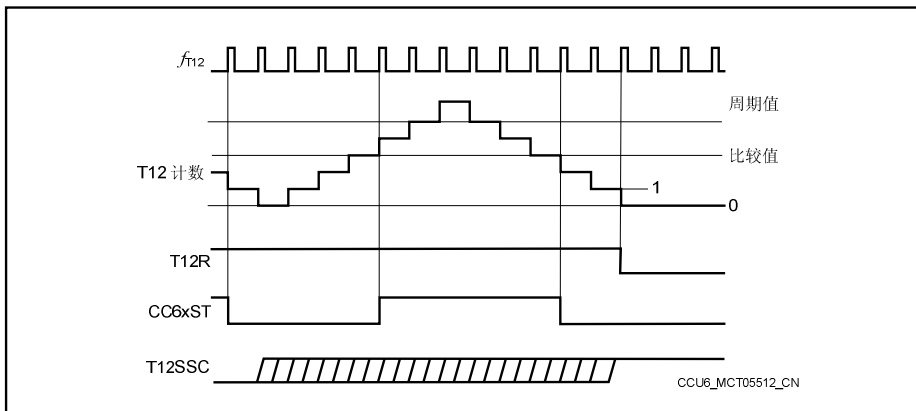
- 计数器递减计数到 0001<sub>H</sub> 时，下一个  $f_{T12}$  时钟 T12 计数方向被设置为递增计数 ( $CDIR = 0$ )。
- 计数器递增计数到周期匹配时，下一个  $f_{T12}$  时钟 T12 计数方向被设置为递减计数 ( $CDIR = 1$ )。
- $CDIR = 0$  时，下一个  $f_{T12}$  时钟计数器递增计数； $CDIR = 1$  时，下一个  $f_{T12}$  时钟计数器递减计数。





**图 20-7 边沿对齐模式下的单次操作**

中心对齐模式下，定时器递减计数至零时，计数周期结束（递减计数至 1-匹配之后一个时钟周期，见图 20-8）。



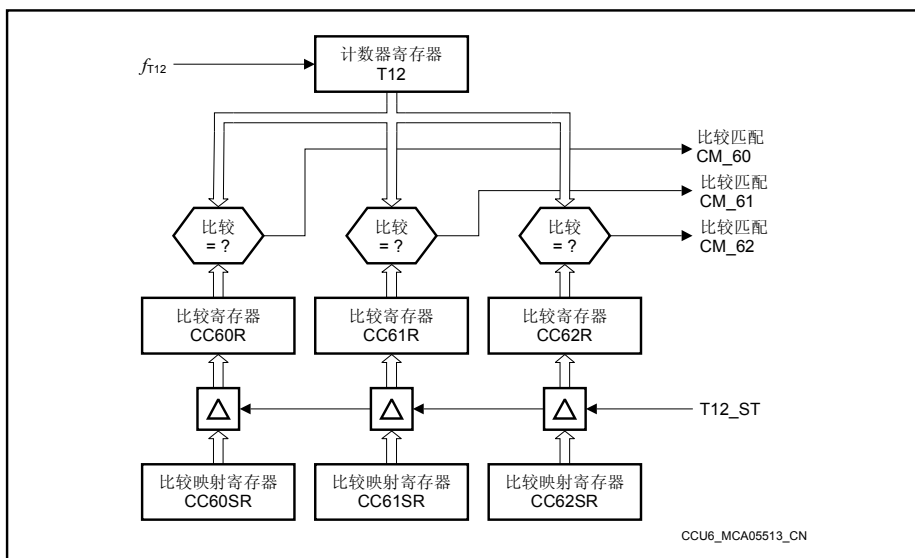
**图 20-8 中心对齐模式下的单次操作**

### 20.2.3 T12 的比较模式

定时器 T12 的三路独立的捕获/比较通道可协同 T12 的计数值，分别执行比较或捕获操作。捕获功能的具体内容请参阅[章节 20.2.5](#)。

#### 20.2.3.1 比较通道

比较模式下（见[图 20-9](#)），三路比较通道 CC60、CC61 和 CC62 可产生三相 PWM 序列。



**图 20-9 T12 通道比较器**

每路通道通过各自的相等比较器和 T12 的计数寄存器相连，当计数器的值和比较寄存器中的值匹配时产生匹配信号。每路通道由比较器和双寄存器结构（实际比较寄存器 CC6xR 和相关映射寄存器 CC6xSR）组成。CC6xR 与比较器直接相连、CC6xSR 由软件预先加载。当 T12 的映射传送信号 T12\_ST 有效时，映射寄存器中的内容被传送到实际比较寄存器中。CAPCOM6 单元提供了比较值以及（和产生 PWM 信号相关的）其它数值的映射寄存器，便于软件同步更新所有相关参数。

### 20.2.3.2 通道状态位

每个（比较）通道都有对应的状态位，**CMPSTAT.CC6xST**，用来保存比较（或捕获）操作的状态（见图 20-10）。比较模式下，根据一套切换规则（取决于定时器 T12 的当前状态）修改状态位。

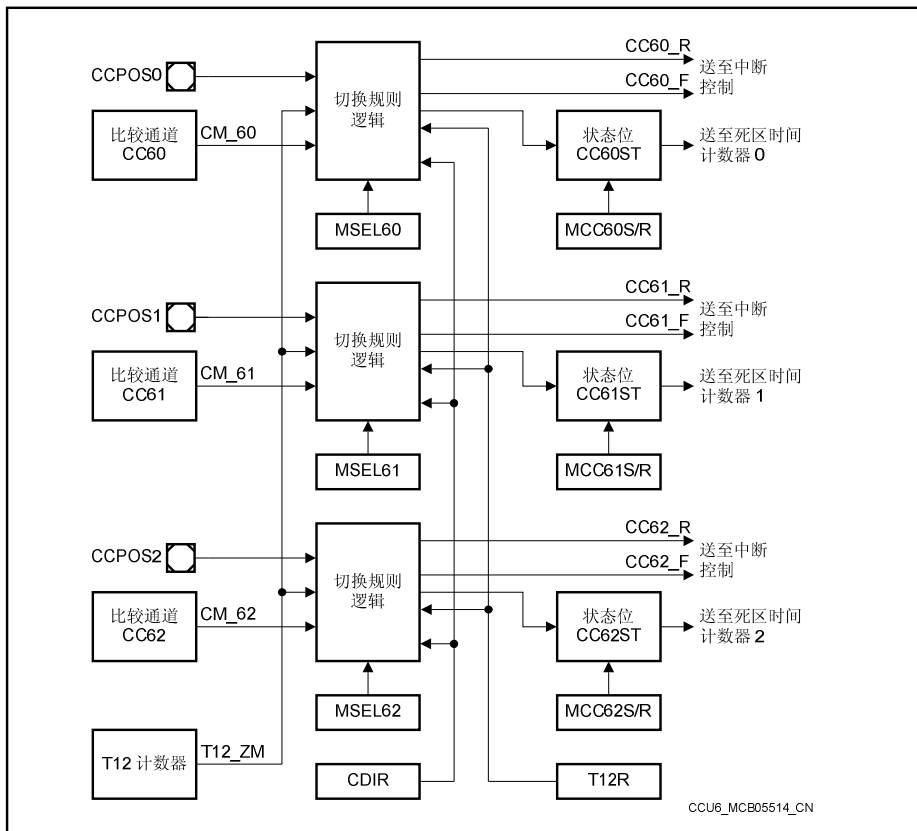


图 20-10 比较模式下的比较状态位框图

CC6xST 切换规则逻辑的输入包括：计数方向标志位 (CDIR)，定时器运行位 (T12R)，定时器 0-匹配信号 (T12\_ZM)，各通道的比较匹配信号 **CM\_6x** 以及模式控制位 **T12MSEL.MSEL6x**。

## 捕获/比较单元 6 (CCU6)

此外，可通过软件设定寄存器 **CMPMODIF** 内的置位控制位 **MCC6xS** 和复位控制位 **MCC6xR** 分别置位或复位各状态位。输入信号 **CCPOSx** 用于类磁滞比较模式，正常比较模式下，这些输入被忽略。

*注：霍尔传感器模式下，还需要考虑附加的/不同的规则（请参阅相关章节）。*

递增计数检测到比较匹配时，则指示发生比较匹配事件 **CC6x\_R**；递减计数检测到比较匹配事件时，则指示发生比较中断事件 **CC6x\_F**。状态位的实际设置对比较模式的中断产生没有影响。

只有当时器 **T12** 工作时 ( $T12R = 1$ ) 才可能由切换规则逻辑根据比较动作修改状态位 **CC6xST**。硬件修改状态位时，使用以下规则置位和复位状态位（如图 20-11 和图 20-12 所示）：

**状态位 CC6xST 被置 1：**

- **T12** 递增计数到比较匹配后，下一个 **T12** 时钟 ( $f_{T12}$ ) 置位 **CC6xST**（即，计数器递增计数到超过比较值）；
- **T12** 递增计数到 0-匹配且同时为比较匹配时，下一个 **T12** 时钟 ( $f_{T12}$ ) 置位 **CC6xST**。

**状态位 CC6xST 被复位为 0：**

- **T12** 递减计数到比较匹配后，下一个 **T12** 时钟 ( $f_{T12}$ ) 复位 **CC6xST**（即，计数器递减计数到低于比较值）；
- **T12** 递增计数到 0-匹配但同时不是比较匹配时，下一个 **T12** 时钟 ( $f_{T12}$ ) 复位 **CC6xST**。

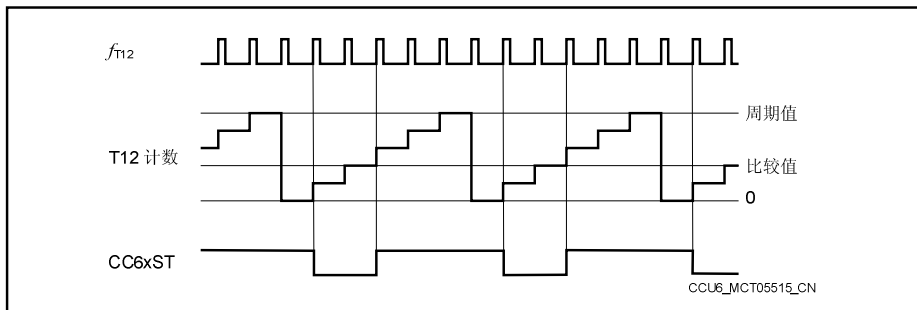


图 20-11 边沿对齐模式下的比较操作

**图 20-13** 例举出更多的比较波形。请务必注意：图中假设定时器计数过程中某些比较值被修改。修改值由软件预先载入映射寄存器 **CC6xSR** 中，当 **T12** 的映射传送信号 **T12\_ST**（假定被使能）有效时，该修改值被传送到实际比较寄存器 **CC6xR** 中。



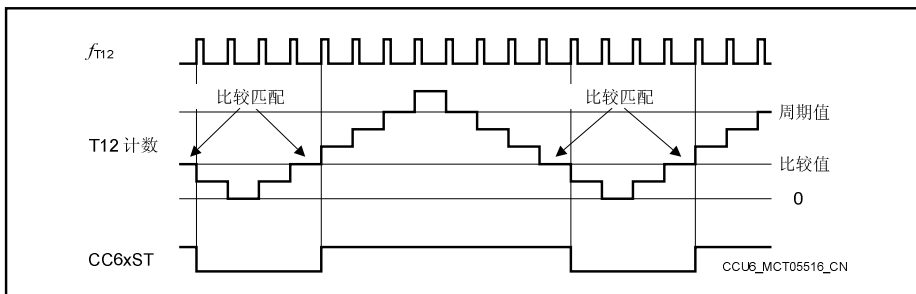


图 20-12 中心对齐模式下的比较操作

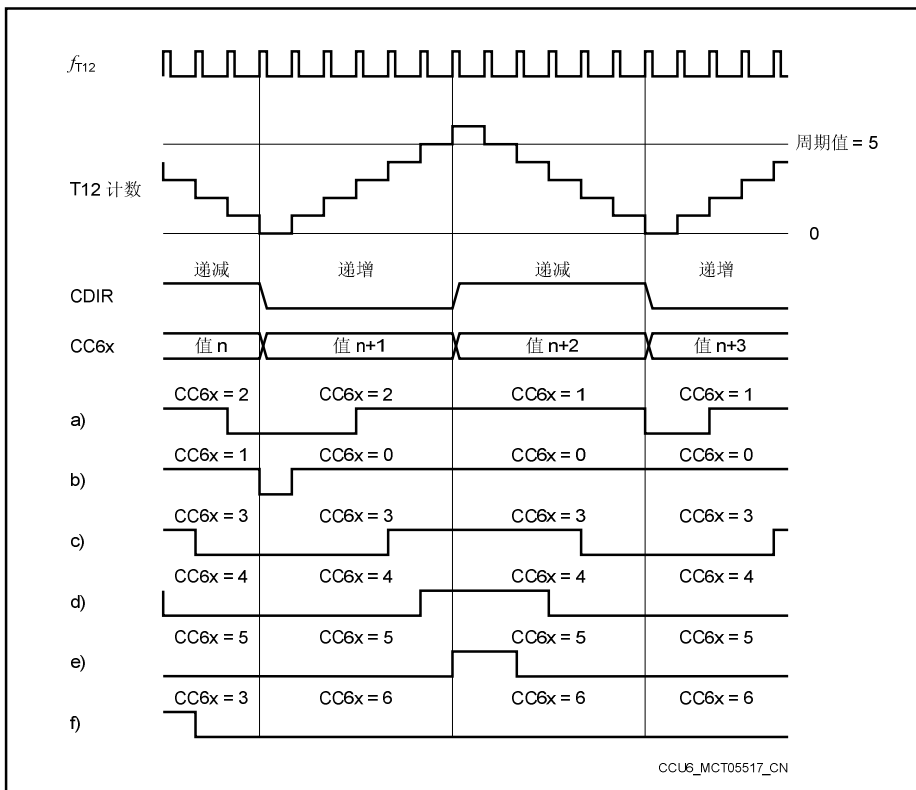


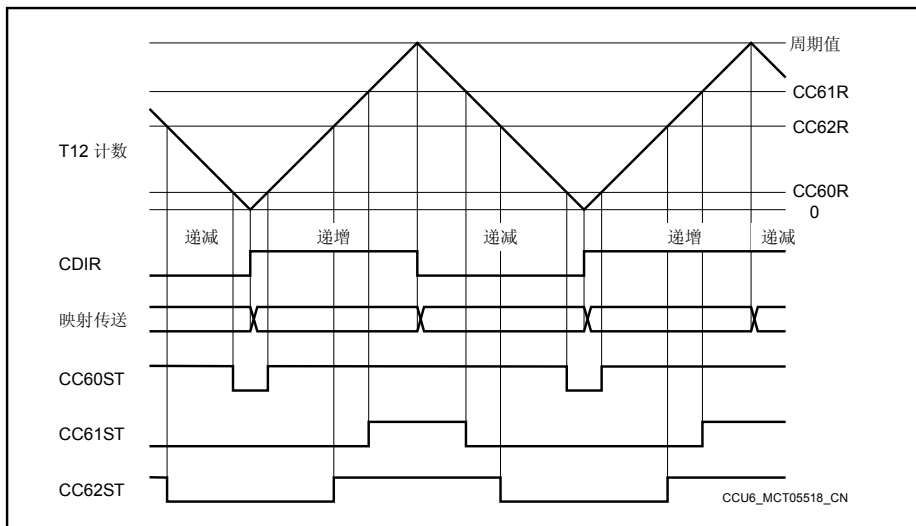
图 20-13 比较波形举例

## 捕获/比较单元 6 (CCU6)

例 b) 图示说明跳变到占空比为 100%的情况。先使用比较值 0001<sub>H</sub>，然后改为 0000<sub>H</sub>。请注意：在新比较值生效的定时器周期内，仍产生一个 T12 时钟周期的低脉冲（该脉冲由前面的比较值 0001<sub>H</sub>产生）。接下来的定时器周期，状态位 CC6xST 始终保持为 1，产生占空比为 100%的信号。此时，比较规则“0-匹配且同时为比较匹配”生效。

例 f) 图示说明跳变到占空比为 0%的情况。新比较值设定为 <周期寄存器的值> + 1，状态位 CC6xST 始终保持为 0。

**图 20-14** 举例说明三通道比较波形的产生。通过恰当的死区时间控制和输出调制，可产生高效的三相 PWM 信号。



**图 20-14** 三通道比较波形

### 20.2.3.3 类磁滞控制模式

类磁滞控制模式（**T12MSEL**.MSEL6x = 1001<sub>B</sub>）提供了这样的可能性：如果输入 CCPOSx 变为 0，通过清除状态位 CC6xST 可切断 PWM 输出。利用该模式可实现简单的电机控制，例如用比较器指示过流。当 CCPOSx = 0 时，相应通道的 PWM 输出驱动其被动电平。只有当 CCPOSx = 1 时才可能置位 CC6xST。

只要输入 CCPOSx 为 0，相应的状态位保持为 0。当 CCPOSx 处于高电平时，输出可进入主动态，并由位 CC6xST 决定（见**图 20-10** 状态位逻辑和**图 20-15** 输出路径）。

## 捕获/比较单元 6 (CCU6)

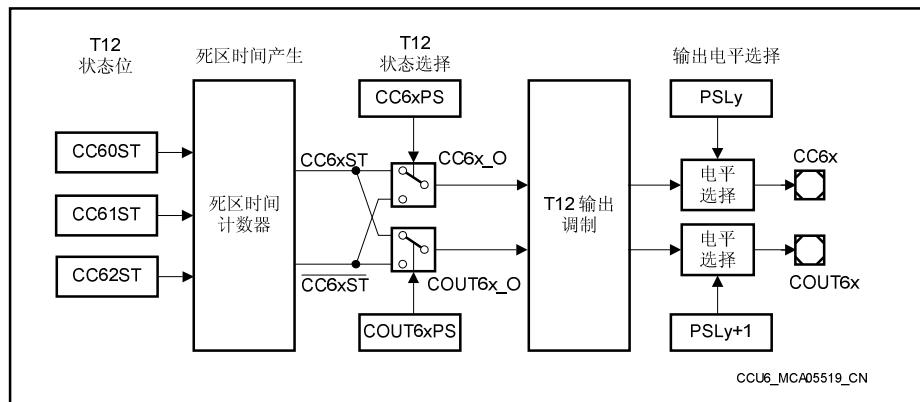
该模式可将和时间相关的控制行为引入类磁滞控制器。标准的类磁滞控制器检测某值是否已超过极限，并根据比较结果切换输出。根据运行条件不同，切换频率和占空比不固定、可不断改变。

若要实现基于类磁滞控制器（控制内环）的时间相关控制环路（外部环路），如果外部环路与内环同步，将会表现出更好的行为特性。因此，可采用本产品提供的类磁滞模式，它结合了按时间切换的控制和类磁滞控制两种功能。例如，在该模式，可在某个固定时刻接通输出；一旦检测到输入引脚 **CCPOSx** 的下降沿，则立刻切断输出。

类磁滞模式可产生带有过流保护的标准 **PWM**。只要引脚 **CCPOSx** 上不出现低电平，则以正常方式产生输出信号（先前章节已作说明）；只有当引脚 **CCPOSx** 上出现低电平，如检测到过流时，输出才被切断以避免损坏系统。

### 20.2.4 比较模式输出路径

**图 20-15** 给出从通道状态位到其输出引脚最简化的信号路径。如图所示，用户可通过多种控制选择，决定当前状态位 **CC6xST** 所对应的（期望）输出信号的切换操作。输出调制的具体内容请参阅**章节 20.2.4.3**。



**图 20-15** 比较模式简化输出路径框图

输出路径基于被定义为主动或被动的信号。术语主动和被动与输出电平无关，而和内部动作有关。主要适用于信号调制，此时 **T12** 和 **T13** 信号与多通道信号以及强制中断功能相结合。输出电平选择功能允许用户定义被动态时输出引脚上的输出电平（主动状态的电平反向）。推荐用户采用下述方式配置该模块，当 **CCU6** 在被动态输出信号时，外部功率开关断开。

#### 20.2.4.1 死区时间产生

功率逆变器中某一相的上下桥臂开关信号（信号互补）由同一个比较通道产生。例如，若 T12 计数值大于比较值时，上桥臂的控制开关导通（状态位 = 1），那么，当 T12 计数值小于比较值时，下桥臂的控制开关导通（状态位 = 0）。

大多数情况下,相互连接的功率开关的开关特性不对称,即接通和断开所需时间不同。若功率器件的接通时间小于断开时间,通常会引发问题,导致逆变器桥臂短路,从而可能会损坏整个系统。为了通过硬件解决该问题,捕获/比较单元中实现了一个可编程的死区时间产生模块,使信号从被动态到主动态的切换延迟产生(主动态到被动态的切换不延迟)。

T12 三路通道的死区时间产生模块的结构相似，如图 20-16 所示。由寄存器 **T12DTC** 中的位进行控制。只要状态位 **CC6xST** 变化，即激活对应的死区时间计数器工作，计数时钟与 **T12** 输入时钟 ( $f_{T12}$ ) 相同。死区时间的长度可由位域 **DTM** 编程设定。对于 **T12** 三路通道来说，死区时间的长度一致。写 **TCTR4.DTRES = 1**，则将所有死区时间设置为被动。

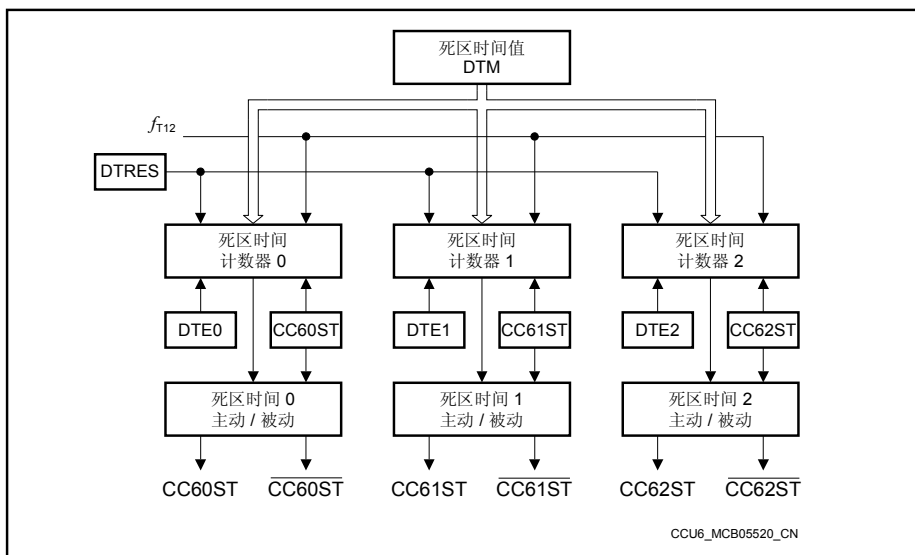
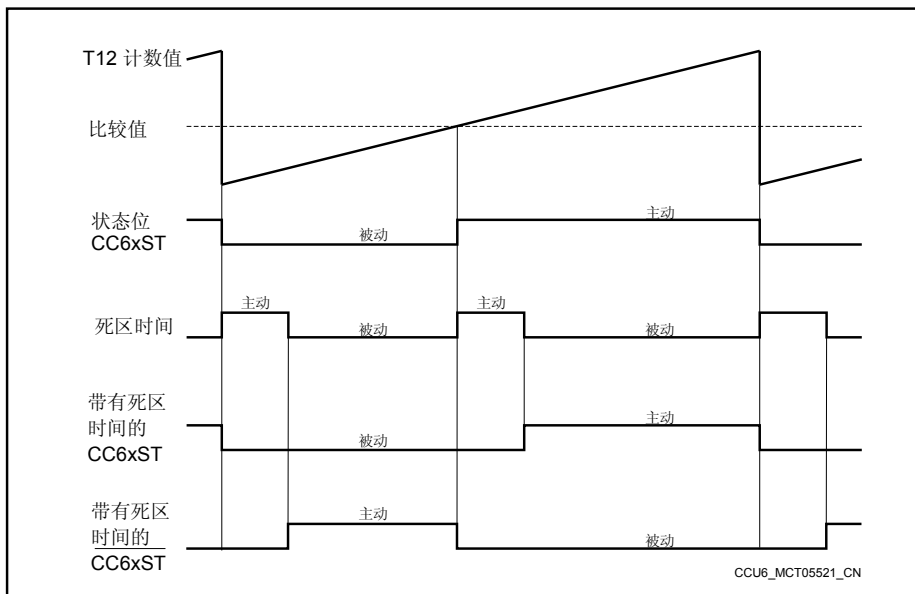


图 20-16 死区时间产生框图

三个死区时间计数器中的每一个对应有各自的死区时间使能位 **DTE<sub>x</sub>**。使能的死区时间计数器产生通道输出信号从被动到主动的死区时间延迟。当某通道正在进行死区时间产生时（主动），状态位 **CC6xST** 的改变无效。这就避免了状态位 **CC6xST** 变化过

早时，无意引起的附加死区时间。被禁止的死区时间计数器始终被认为是被动的，且不会延迟任何 CC6xST 边沿。

根据状态位 CC6xST，死区时间产生模块为每个比较通道输出一个直接信号 CC6xST 和一个反向信号  $\overline{\text{CC6xST}}$ ，由相关的死区时间计数器的结果屏蔽每个信号（波形见图 20-17）。



**图 20-17 死区时间产生波形**

#### 20.2.4.2 状态选择

为了支持更多类型的功率开关和驱动器，可定义何时输出可为主动并被调制，状态选择为用户提供更大的灵活性，该特性对于**互补或多相 PWM** 信号尤其有用。

根据输入到死区时间产生器的信号 CC6xST 和  $\overline{\text{CC6xST}}$ （见图 20-15）进行状态选择。这两个信号决不会同时为主动态，但是可以同时处于被动态。每个比较通道相应的状态位 CC6xST 改变之后的死区时间内会出现此情况。

如果在达到比较值之前或之后（见寄存器 **CMPSTAT**）输出信号应该为主动态，用户可单独为每个输出信号 CC6xO 和 COUT6xO 选择状态。进行状态选择之后，功率逆变器桥臂中的互补功率开关的主动（接通）阶段和比较值之间的对应关系确定（如达到比较值之前，信号 CC6xO 变为主动态，而在达到比较值之后，COUT6xO 变为主动

态。与此类似，尽管同一个比较通道对应有两个输出信号，但每个输出信号的输出调制，强制中断逻辑和输出电平选择都可单独编程设定。

### 20.2.4.3 输出调制和电平选择

数据最后送入输出调制控制模块。该模块将所有的调制源和强制中断功能逻辑组合，以控制输出引脚的实际电平（由寄存器 **MODCTR** 内的调制使能位 **T1xMODENy** 和 **MCMEN** 控制）。对于每个 **T12 输出信号**（比较通道 **CC60** 的情况参见图 20-18），由下列信号源组合得到：

- **T12 相关比较信号** **CC6x\_O**（对于输出 **CC6x**）或 **COUT6x\_O**（对于输出 **COUT6x**）由 **T12** 模块给出（带死区时间的状态选择），每个输出信号具有单独的使能位 **T12MODENy**（对于输出 **CC6x**， $y = 0, 2, 4$ ；对于输出 **COUT6x**， $y = 1, 3, 5$ ）。
- **T13 相关比较信号** **CC63\_O** 由 **T13** 状态选择给出，每个输出信号具有一个单独的使能位 **T13MODENy**（对于输出 **CC6x**， $y = 0, 2, 4$ ；对于输出 **COUT6x**， $y = 1, 3, 5$ ）。
- **多通道输出信号** **MCMPy**（对于输出 **CC6x**， $y = 0, 2, 4$ ；对于输出 **COUT6x**， $y = 1, 3, 5$ ）具有一个公共使能位 **MCMEN**。
- **强制中断状态** **TRPS** 每个输出信号都有一个单独的使能位 **TRPENy**（对于输出 **CC6x**， $y = 0, 2, 4$ ；对于输出 **COUT6x**， $y = 1, 3, 5$ ）。

若输出调制模块的某个调制输入信号 **CC6x\_O/COUT6x\_O**，**CC63\_O**，或 **MCMPy** 被使能且处于被动态，则被调制信号也处于被动态，和其它被使能信号的状态无关；只有当所有被使能信号均处于主动态，调制输出才为主动态。如果调制输入都未被使能，输出处于被动态。

若强制中断状态有效（**TRPS** = 1），所有使能的强制中断（**TRPENy** = 1）的输出被设置为被动态。

每个调制控制模块的输出和一个电平选择模块相连（由寄存器 **PSLR** 对电平选择模块进行配置）。根据输出线的状态，由该模块决定引脚的实际输出电平。输出线状态由被动态选择位 **PSLy** 决定（与主动/被动态和输出极性无关）。如果被调制的输出处于被动态，**PSLy** 指定的电平即为输出的实际电平；如果被调制的输出处于主动态，**PSLy** 的反相电平为输出的实际电平。该特性允许用户调整与电路连接的主动输出信号的极性。

控制位 **PSLy** 对应有映射寄存器，从而在更新 **PSLy** 时避免了输出线上多余的脉冲。由 **T12** 的映射传送信号（**T12\_ST**）触发 **CC6x** 和 **COUT6x**（ $x = 0, 1, 2$ ）相关的位的更新。读操作读取实际使用值；写操作写入映射位。**CCU6** 单元提供了 **PSL**、以及（和产生 **PWM** 信号相关的）其它数值的映射寄存器，便于软件同步更新所有相关参数。

图 20-18 给出比较通道 **CC60** 的输出调制结构（输出信号 **CC60** 和 **COUT60**）。比较通道 **CC61** 和 **CC62** 与 **CC60** 的输出调制结构相似。

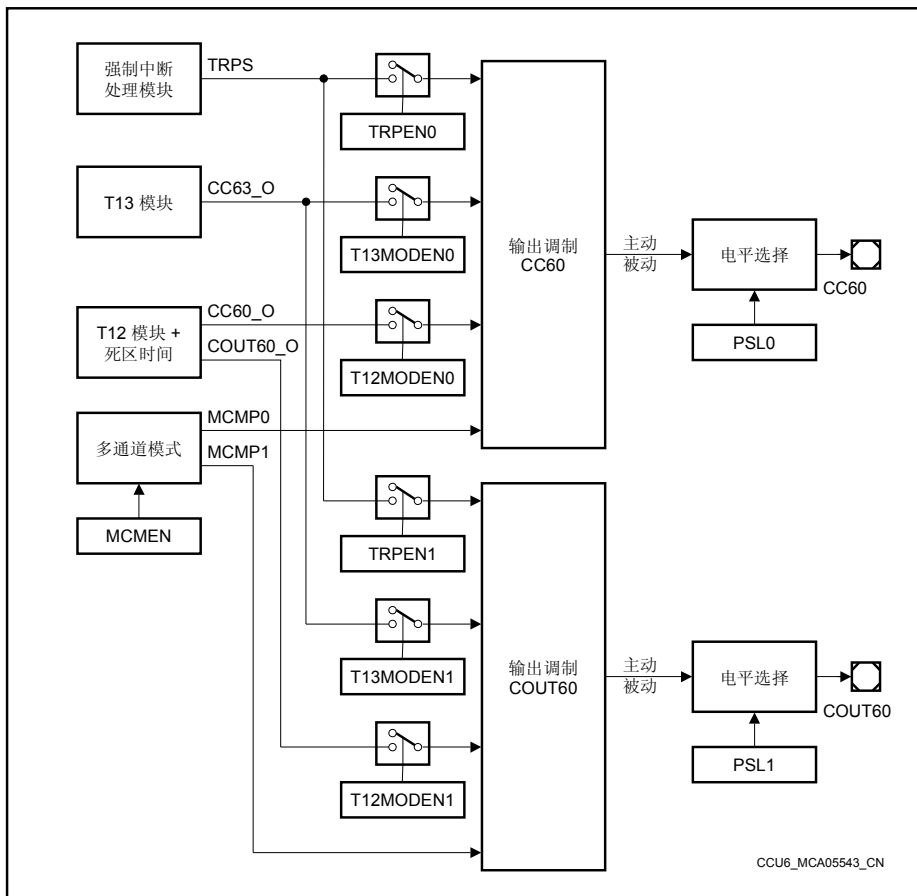


图 20-18 比较通道 CC60 的输出调制

### 20.2.5 T12 捕获模式

T12 模块的三路通道均可响应外部信号 CC6xIN，用于捕获 T12 的时间信息。

捕获模式下，当在输入 CC6xIN 上检测到上升沿时，则发生中断事件 CC6x\_R，在输入 CC6xIN 上检测到下降沿时，则发生中断事件 CC6x\_F。

CCU6 单元提供了多种不同的捕获模式。所有捕获模式下，均使用各路通道的两个寄存器（实际寄存器以及映射寄存器）。通过 **T12MSEL.MSEL6x** 位域分别为各通道选择捕获模式。

表 20-3 捕获模式总览

MSEL6x	模式	引脚	有效沿	CC6nSR 保存到	T12 保存到
0100 <sub>B</sub>	1	CC6xIN	上升沿	-	CC6xR
		CC6xIN	下降沿	-	CC6xSR
0101 <sub>B</sub>	2	CC6xIN	上升沿	CC6xR	CC6xSR
0110 <sub>B</sub>	3	CC6xIN	下降沿	CC6xR	CC6xSR
0111 <sub>B</sub>	4	CC6xIN	任意沿	CC6xR	CC6xSR

**图 20-19** 说明**捕获模式 1**的操作。检测到输入信号 CC6xIN 的上升沿（0 到 1 跳变）时，将定时器 T12 的当前计数值捕获到寄存器 CC6xR 中；检测到输入引脚 CC6xIN 的下降沿（1 到 0 跳变）时，将定时器 T12 的内容捕获到 CC6xSR 中。



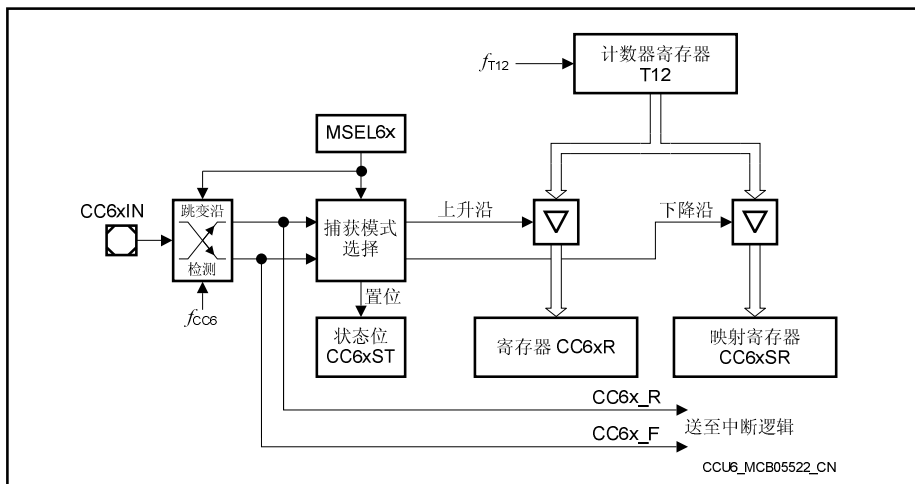
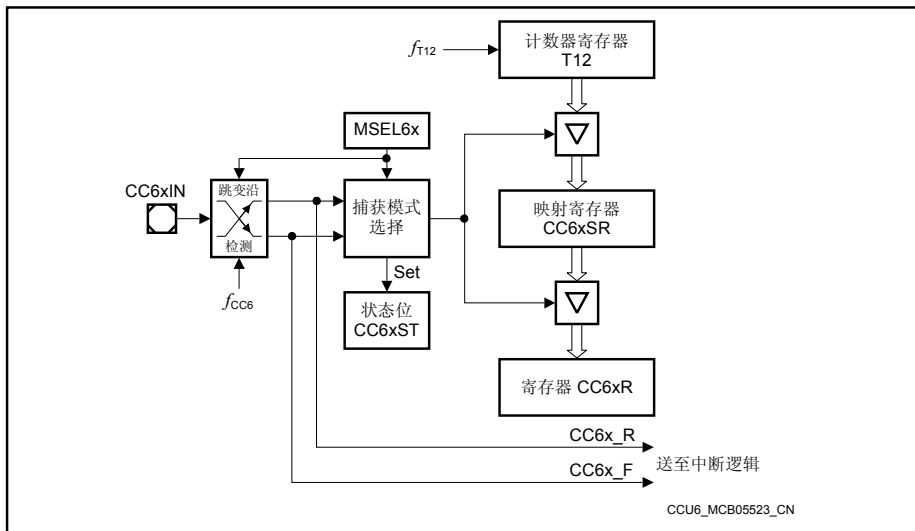


图 20-19 捕获模式 1 框图

**捕获模式 2、3 和 4** 如图 20-20 所示，各模式的差别仅在于引发捕获操作的有效跳变沿不同。当处于三种捕获模式中的任意一种，检测到输入引脚 CC6xIN 发生有效跳变时，对应的映射寄存器 CC6xSR 的当前值被传送到寄存器 CC6xR 中，定时器 T12 的当前值捕获到寄存器 CC6xSR 中（同时传送）。捕获模式 2 的有效沿为引脚 CC6xIN 的上升沿；捕获模式 3 的有效沿为引脚 CC6xIN 的下降沿；捕获模式 4 的有效沿为引脚 CC6xIN 的任意沿，见表 20-3。当输入信号连续两次跳变间隔很短时，这些捕获模式非常有用。



**图 20-20 捕获模式 2、3 和 4 框图**

## 捕获/比较单元 6 (CCU6)

此外，还有五种**多输入捕获模式**，使用两个外部信号输入 CC6xIN 和 CCPOSx 触发捕获。

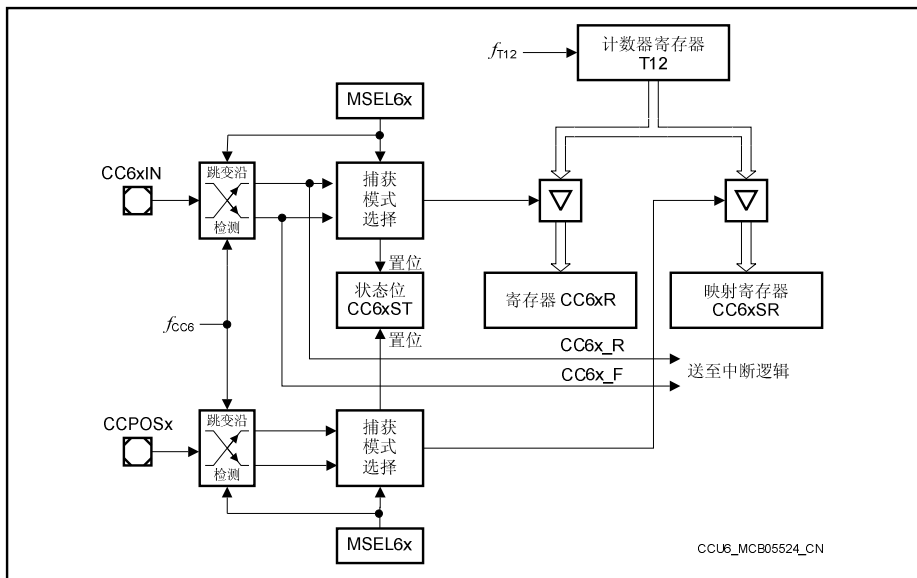


图 20-21 多输入捕获模式框图

多输入捕获模式下，信号 CC6xIN 上选定的事件触发 T12 的当前值锁存到寄存器 CC6xR 中；信号 CCPOSx 上选定的事件触发 T12 的当前值锁存到寄存器 CC6xSR 中。触发事件包括：两个输入引脚上相反的跳变沿、相同的跳变沿、或任意跳变沿。触发事件选择见表 20-4。

各种不同的捕获模式下，当信号 CC6xIN 或 CCPOSx 上发生所选定的捕获触发事件时，通道状态位 CC6xST 被置 1。该通道状态位不能由硬件清除，可由软件清除。

此外，中断逻辑的相应信号线被激活，向 CPU 发送中断请求。与所选择的有效边沿无关，信号 CC6xIN 上检测到的所有边沿都会激活相应的中断请求线（请参阅章节 20.8）。

**表 20-4      多输入捕获模式总览**

MSEL6x	模式	信号	有效沿	T12 保存到
1010 <sub>B</sub>	5	CC6xIN	上升沿	CC6xR
		CCPOSx	下降沿	CC6xSR
1011 <sub>B</sub>	6	CC6xIN	下降沿	CC6xR
		CCPOSx	上升沿	CC6xSR
1100 <sub>B</sub>	7	CC6x	上升沿	CC6xR
		CCPOSx	上升沿	CC6xSR
1101 <sub>B</sub>	8	CC6xIN	下降沿	CC6xR
		CCPOSx	下降沿	CC6xSR
1110 <sub>B</sub>	9	CC6xIN	任意沿	CC6xR
		CCPOSx	任意沿	CC6xSR
1111 <sub>B</sub>	-	保留（无捕获或比较操作）		

### 20.2.6 T12 映射寄存器传送

采用一个特殊的映射传送信号 (T12\_ST) 便于使比较通道 CC60、CC61 和 CC62 周期值、比较值的更新和 T12 的操作同步进行。映射寄存器使得软件能够同时更新定义 PWM 周期的所有相关参数。下一个 PWM 周期可使用一组新的参数。软件通过位 **TCTR0.STE12**（向只写位 **TCTR4.T12STR** 写入 1 将置位 STE12；向只写位 **TCTR4.T12STD** 写入 1 将清除 STE12）。

**图 20-22** 给出映射寄存器结构和映射传送信号，以及各种寄存器的读/写访问特性。

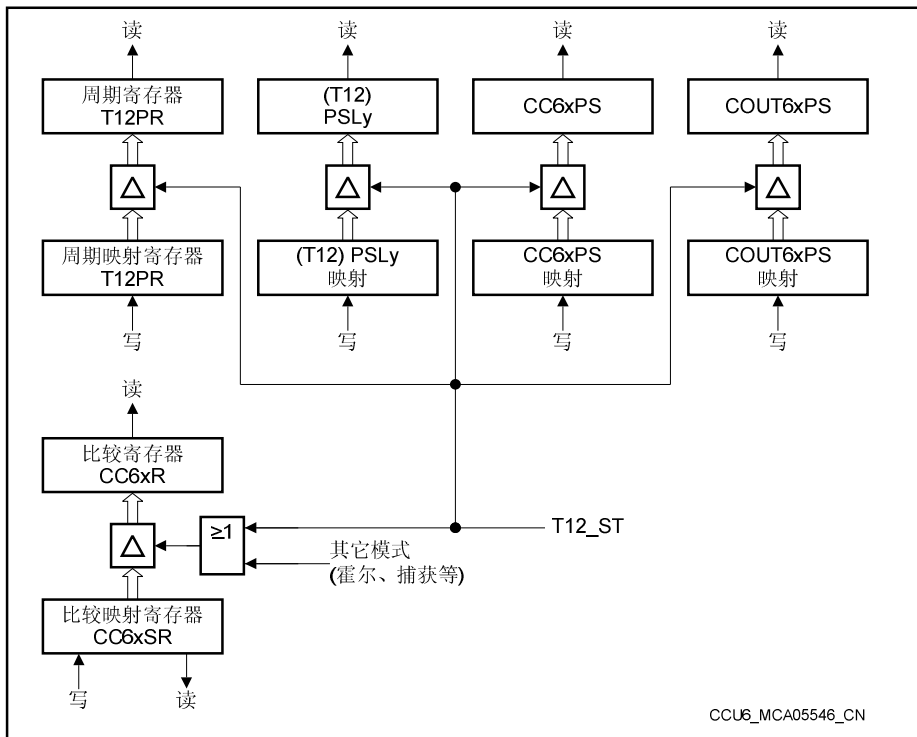


图 20-22 T12 映射寄存器概览

在以下条件下，发生 T12 映射寄存器传送（T12\_ST 有效）：

- 定时器 T12 停止工作（T12R = 0），或
- STE = 1 且递增计数时检测到周期匹配，或
- STE = 1 且递减计数时检测到 1-匹配

当信号 T12\_ST 有效，下一个 T12 时钟周期触发映射寄存器传送。映射寄存器传送之后，位 STE12 自动清零。

## 20.2.7 定时器 T12 工作模式选择

由位域 **T12MSEL**.MSEL6x 定义 T12 各通道的工作模式。

**表 20-5 T12 捕获/比较模式总览**

MSEL6x	选择的工作模式
0000 <sub>B</sub> , 1111 <sub>B</sub>	比较/捕获模式关闭
0001 <sub>B</sub> , 0010 <sub>B</sub> , 0011 <sub>B</sub>	比较模式，见 <a href="#">章节 20.2.3</a> 三种编码的操作相同
01XX <sub>B</sub>	双寄存器捕获模式，见 <a href="#">章节 20.2.5</a> 。
1000 <sub>B</sub>	霍尔传感器模式，见 <a href="#">章节 20.6</a> 。 为确保能够正确使能该模式，三组位域 MSEL6x 必须全部被设置为霍尔传感器模式。
1001 <sub>B</sub>	类磁滞模式，见 <a href="#">章节 20.2.3.3</a> 。
1010 <sub>B</sub> 1011 <sub>B</sub> 1100 <sub>B</sub> 1101 <sub>B</sub> 1110 <sub>B</sub>	多输入捕获模式，见 <a href="#">章节 20.2.5</a>

由定时器控制寄存器 **TCTR0** 和 **TCTR2** 控制定时器的时钟和计数策略。通过写寄存器 **TCTR4** 可触发特定的操作。

## 20.2.8 T12 相关寄存器

### 20.2.8.1 T12 计数寄存器

寄存器 T12 的内容表示定时器 T12 的计数值。只有当定时器 T12 停止工作时，才可对其进行写操作；定时器 T12 运行时写操作无效。寄存器 T12 始终可由软件读取。

在边沿对齐模式下，T12 仅递增计数；中心对齐模式下，T12 可递增和递减计数。

#### T12

**定时器 T12 计数寄存器**                      **XSFR (10<sub>H</sub>)**                      **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>T12CV</b>															
rwh															

符号	位序号	类型	功能描述
<b>T12CV</b>	[15:0]	rwh	<b>定时器 T12 的计数值</b> 该寄存器表示定时器 T12 的 16 位计数值。

*注：定时器 T12 停止运行时，为了保证能够产生可重复的时序和延迟，内部时钟分频器被复位。*

### 20.2.8.2 周期寄存器

寄存器 T12PR 中保存定时器 T12 的周期值。周期值和 T12 的当前计数值进行比较，根据规定的计数规则执行相关操作。该寄存器对应一个映射寄存器，位 STE12 控制映射传送。软件读操作读出当前用于周期比较的周期值；而写操作将周期值写入映射寄存器中。映射寄存器结构便于用户同步更新 T12 的所有相关值。

#### T12PR

**定时器 T12 周期寄存器**                      **XSFR (12<sub>H</sub>)**                      **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>T12PV</b>															
rwh															

### 捕获/比较单元 6 (CCU6)

符号	位序号	类型	功能描述
<b>T12PV</b>	[15:0]	rwh	<b>定时器 T12 的周期值</b> T12 的计数值和 T12PV 相等时产生周期匹配。计数到该值后，定时器 T12 被置 0（边沿对齐模式）或将其计数方向改为递减计数（中心对齐模式）。

### 20.2.8.3 捕获/比较寄存器

比较模式下，寄存器 CC6xR (x=0-2) 是 T12 的实际比较寄存器。保存在 CC6xR 中的值和 T12 的计数值进行比较（三路通道同时进行）。捕获模式下，当检测到相关的捕获事件时，T12 计数寄存器的当前值被捕获到寄存器 CC6xR 中。

### CC6xR (x = 0-2)

通道 CC6x 的捕获/比较寄存器      XSFR (18<sub>H</sub>+2\*x)      复位值: 0000<sub>H</sub>

rh

符号	位序号	类型	功能描述
<b>CCV</b>	[15:0]	rh	<b>捕获/比较值</b> 比较模式下，位域 <b>CCV</b> 中存放着和 <b>T12</b> 的计数值进行比较的值；捕获模式下， <b>T12</b> 的捕获值可从这些寄存器读出。



**20.2.8.4 捕获/比较映射寄存器**

寄存器 CC6xR 可由软件读取，若要修改 CC6xR 的值，则需要将映射寄存器 CC6xSR 中的修改值映射传送到 CC6xR 中。映射寄存器可由软件读写。捕获模式下，当检测到相关的捕获事件时（取决于所选择的捕获模式），T12 计数寄存器的值被捕获到寄存器 CC6xSR 中。

**CC6xSR (x = 0-2)**

**通道 CC6x 捕获/比较映射寄存器 XSFR ( $20_H + 2 * x$ )** **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CCS</b>															
rwh															

符号	位序号	类型	功能描述
<b>CCS</b>	[15:0]	rwh	<b>通道 x 捕获/比较值映射寄存器</b> 比较模式下，映射传送期间，位域 CCS 的内容被传送到位域 CCV 中；捕获模式下，T12 的捕获值可从这些寄存器读出。

*注：捕获模式下，软件也可写映射寄存器。此时，如果硬件捕获操作和软件写操作发生在同一个周期内，则硬件捕获操作占优（软件写操作被丢弃）。*

### 20.2.8.5 死区时间控制寄存器

寄存器 **T12DTC** 控制定时器 **T12** 比较通道的死区时间产生。可分别使能/禁止各路通道的死区时间产生。若被使能，从被动态到主动态的跳变被延迟（由位域 **DTM** 决定延迟时间）。

死区时间计数器采用和 **T12** 相同的时钟频率。

在中心对齐和边沿对齐 **PWM** 模式，该结构可实现对称的死区时间产生。占空比为 50%使得 **CC6x**, **COUT6x** 开启时间为：0.5\*周期 – 死区时间。

*注：死区时间计数器不是由位 **T12RES** 复位，而是由位 **DTRES** 复位。*

### T12DTC

#### T12 死区时间控制寄存器

**XSFR (14<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DTR 2	DTR 1	DTR 0	0	DTE 2	DTE 1	DTE 0	DTM							
r	rh	rh	rh	r	rw	rw	rw	rw							

符号	位序号	类型	功能描述
<b>DTM</b>	[7:0]	rw	<b>死区时间值</b> 位域 <b>DTM</b> 规定了所选择的输出从被动态切换到主动态的可编程延迟。从主动态切换到被动态没有延迟。
<b>DTE2, DTE1, DTE0</b>	10, 9, 8	rw	<b>死区时间产生使能位</b> DTE0...DTE2 使能和禁止定时器 <b>T12</b> 各路比较通道 (0, 1, 2) 的死区时间产生。  <b>0<sub>B</sub></b> 死区时间计数器 <b>x</b> 被禁止。根据实际比较状态，相应的输出从被动态立即切换到主动态（无延迟）。  <b>1<sub>B</sub></b> 死区时间计数器 <b>x</b> 被使能。根据实际比较状态，相应的输出从被动态到主动态的切换操作被延迟（在位域 <b>DTM</b> 内的编程设定延迟）。
<b>DTR2, DTR1, DTR0</b>	14, 13, 12	rh	<b>死区时间运行指示标志</b> 位 <b>DTR0...DTR2</b> 指示定时器 <b>T12</b> 各路比较通道 (0, 1, 2) 死区时间产生的状态。  <b>0<sub>B</sub></b> 死区时间计数器 <b>x</b> 当前处于被动状态

**捕获/比较单元 6 (CCU6)**

符号	位序号	类型	功能描述
			1 <sub>B</sub> 死区时间计数器 x 当前处于主动状态
<b>0</b>	15, 11	r	保留; 读操作返回 0; 应写入 0。

## 20.2.9 捕获/比较控制寄存器

### 20.2.9.1 通道状态位

通道状态寄存器 **CMPSTAT** 中的状态位指示当前的捕获或比较状态，控制位决定比较通道处于主动态还是被动态。

#### **CMPSTAT**

##### 比较状态寄存器

##### **XSFR (28<sub>H</sub>)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>T13 IM</b>	<b>C OUT 63PS</b>	<b>C OUT 62PS</b>	<b>CC 62PS</b>	<b>C OUT 61PS</b>	<b>CC 61PS</b>	<b>C OUT 60PS</b>	<b>CC 60PS</b>	<b>0</b>	<b>CC 63ST</b>	<b>CC POS 62</b>	<b>CC POS 61</b>	<b>CC POS 60</b>	<b>CC 62ST</b>	<b>CC 61ST</b>	<b>CC 60ST</b>
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	r	rh	rh	rh	rh	rh	rh	rh

符号	位序号	类型	功能描述
<b>CC60ST</b> <b>CC61ST</b> <b>CC62ST</b> <b>CC63ST</b> 1)	0, 1, 2, 6	rh	<b>捕获/比较状态位</b> 位 CC6xST 监控捕获/比较通道的状态。位 CC6xST(x = 0, 1, 2)与 T12 相关，位 CC63ST 与 T13 相关。 0 <sub>B</sub> 比较模式下，定时器计数值小于比较值。 捕获模式下，自从上次软件清除该位，还未检测到所选择的跳变沿。 1 <sub>B</sub> 比较模式下，定时器计数值大于或等于比较值 捕获模式下，已检测到选定的跳变沿。
<b>CCPOS60,</b> <b>CCPOS61,</b> <b>CCPOS62</b>	3, 4, 5	rh	<b>霍尔序列采样值</b> 位 CCPOS6x (x = 0, 1, 2) 指示已经与当前和期望霍尔值进行比较的输入霍尔序列的值。发生 HCRDY (霍尔比较准备就绪) 事件时，该值被采

**捕获/比较单元 6 (CCU6)**

符号	位序号	类型	功能描述
			<p>样。</p> <p>0<sub>B</sub> 采样到的输入 CCPOS6x 为 0</p> <p>1<sub>B</sub> 采样到的输入 CCPOS6x 为 1</p>
<b>CC60PS</b> <b>CC61PS</b> <b>CC62PS</b> <b>COU60PS</b> <b>COU61PS</b> <b>COU62PS</b> <b>COU63PS</b> 2)	8, 10, 12, 9, 11, 13, 14	rwh	<p><b>比较输出的被动态选择</b></p> <p>位 CC6xPS, COU6xPS 选择相应比较通道的被动态。处于被动态时, 输出引脚驱动 (由寄存器 PSLR 定义的) 被动电平。位 CC6xPS, COU6xPS (x = 0, 1, 2) 与 T12 相关, 位 CC63PS 与 T13 相关。</p> <p>0<sub>B</sub> CC6xST 为 0 时相应比较信号处于被动电平</p> <p>1<sub>B</sub> CC6xST 为 1 时相应比较信号处于被动电平</p> <p>捕获模式下, 不使用这些位。</p>
<b>T13IM</b> <sup>3)</sup>	15	rwh	<p><b>T13 反相调制控制</b></p> <p>位 T13IM 将 T13 信号反相用于调制 CC6x 和 COU6x (x = 0, 1, 2) 信号。</p> <p>0<sub>B</sub> T13 输出 CC63_O 等于 CC63ST</p> <p>1<sub>B</sub> T13 输出 CC63_O 等于 <math>\overline{\text{CC63ST}}</math></p>
<b>0</b>	7	r	<p><b>保留;</b></p> <p>读操作返回 0; 应写入 0。</p>

- 1) 根据 T12 和 T13 的切换规则置位和复位这些状态位。
- 2) 这些位具有映射位, 分别与 T12 和 T13 的捕获/比较寄存器同时更新。读操作读取实际使用值, 写操作写入映射位。
- 3) 该位具有映射位, 和 T13 的比较和周期寄存器同时更新。读操作读取实际使用值, 写操作写入到映射位。

**捕获/比较单元 6 (CCU6)**

比较状态修改寄存器 **CMPMODIF** 支持软件修改（独立的置位和清除控制条件）通道状态位 **CC6xST**。该特性允许用户软件单独修改输出线的状态，例如比较定时器停止计数时，用户可自行修改相应的比较状态。

**CMPMODIF**

**比较状态修改寄存器**

**XSFR (2A<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	MCC 63R	0	MCC 62R	MCC 61R	MCC 60R	0	MCC 63S	0	MCC 62S	MCC 61S	MCC 60S				
r	w	r	w	w	w	r	w	r	w	r	w	w	w		

符号	位序号	类型	功能描述
<b>MCC60S,</b> <b>MCC61S,</b> <b>MCC62S,</b> <b>MCC63S,</b> <b>MCC60R,</b> <b>MCC61R,</b> <b>MCC62R,</b> <b>MCC63R</b>	0, 1, 2, 6, 8, 9, 10, 14	w	<b>捕获/比较通道状态修改位</b>  这些位用于软件置位（MCC6xS）或清除（M、MCC6xR）对应的 CC6xST 位。  该特性允许用户通过软件改变输出线的状态，例如，当相应的比较定时器停止时。允许通过单个数据写操作对 CC6xST 位进行控制。  写访问同一个捕获/比较状态位的修改位，则按照下述规则进行： [MCC6xR, MCC6xS] = 00 <sub>B</sub> 位 CC6xST 不变 01 <sub>B</sub> 位 CC6xST 置位 10 <sub>B</sub> 位 CC6xST 被清除 11 <sub>B</sub> 保留
<b>0</b>	[5:3], 7 [13:11] 15	r	<b>保留；</b> 读操作返回 0；应写入 0。

### 20.2.9.2 T12 模式控制寄存器

寄存器 T12MSEL 用于设置定时器 T12 模块中的三路通道对应的捕获/比较功能。

#### T12MSEL

#### T12 模式选择寄存器

**XSFR (46<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>D</b>	<b>HSYNC</b>		<b>MSEL62</b>			<b>MSEL61</b>			<b>MSEL60</b>						
<b>BYP</b>															
	rw		rw			rw			rw			rw			

符号	位序号	类型	功能描述
<b>MSEL60, MSEL61, MSEL62</b>	[3:0], [7:4], [11:8]	rw	<b>捕获/比较模式选择</b> 这些位域为 T12 的三路捕获/比较通道选择相应的工作模式。每路通道 (x = 0, 1, 2) 可分别被设定为其中一种模式 (霍尔传感器模式除外), 见 <a href="#">表 20-5</a> 。
<b>HSYNC</b>	[14:12]	rw	<b>霍尔同步</b> 位域 HSYNC 定义霍尔输入序列采样和与当前以及期望霍尔序列位域进行比较的信号源。编码见 <a href="#">表 20-11</a> 。
<b>DBYP</b>	15	rw	<b>延迟旁路</b> DBYP 控制是否对霍尔输入序列 (由 HSYNC 选择) 采样的信号源进行延迟, 由死区时间计数器 0 产生延迟。 <b>0<sub>B</sub></b> 旁路无效 源信号变为有效之后, 死区时间计数器 0 产生一个延迟。 <b>1<sub>B</sub></b> 旁路有效 不使用死区时间计数器 0 产生延迟。

### 20.2.9.3 定时器控制寄存器

寄存器 TCTR0 控制定时器 T12 和 T13 的基本功能。

*注：定时器 T12 不运行时 ( $T12R = 0$ )，对位域 T12CLK 或 T12PRE 的写操作才有效。定时器 T13 不运行时 ( $T13R = 0$ )，对位域 T13CLK 或 T13PRE 的写操作才有效。*

#### TCTR0

定时器控制寄存器 0

XSFR (2C<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0		STE 13	T13R	T13 PRE	T13CLK			CTM	CDIR	STE 12	T12R	T12 PRE	T12CLK		
r		rh	rh	rw	rw			rw	rh	rh	rh	rw	rw		

符号	位序号	类型	功能描述
<b>T12CLK</b>	[2:0]	rw	<b>定时器 T12 输入时钟选择</b> 根据等式 $f_{T12} = f_{CC6} / 2^{<T12CLK>}$ ，选择定时器 T12 的输入时钟。 000 <sub>B</sub> $f_{T12} = f_{CC6}$ 001 <sub>B</sub> $f_{T12} = f_{CC6} / 2$ 010 <sub>B</sub> $f_{T12} = f_{CC6} / 4$ 011 <sub>B</sub> $f_{T12} = f_{CC6} / 8$ 100 <sub>B</sub> $f_{T12} = f_{CC6} / 16$ 101 <sub>B</sub> $f_{T12} = f_{CC6} / 32$ 110 <sub>B</sub> $f_{T12} = f_{CC6} / 64$ 111 <sub>B</sub> $f_{T12} = f_{CC6} / 128$
<b>T12PRE</b>	3	rw	<b>定时器 T12 预分频使能位</b> 使能 T12 的附加预分频因子 1/256。 0 <sub>B</sub> 附加预分频被禁止 1 <sub>B</sub> 附加预分频被使能
<b>T12R</b>	4	rh	<b>定时器 T12 运行控制位<sup>1)</sup></b> T12R 启动和终止定时器 T12。软件置位 T12RS 可置位 T12R，软件置位 T12RR 可复位 T12R；或根据位域 T12SSC 定义的功能硬件复位 T12R。

符号	位序号	类型	功能描述
			<b>0<sub>B</sub></b> 定时器 T12 被终止 <b>1<sub>B</sub></b> 定时器 T12 在运行
<b>STE12</b>	5	rh	<b>定时器 T12 映射传送使能</b> 如果发生 T12 映射传送事件, 位 STE12 使能或禁止 T12 的周期值、比较值以及被动态选择位和电平从其映射寄存器传送至实际寄存器的映射传送。映射传送之后硬件清除位 STE12。 <b>0<sub>B</sub></b> 映射寄存器传送被禁止 <b>1<sub>B</sub></b> 映射寄存器传送被使能
<b>CDIR</b>	6	rh	<b>定时器 T12 的计数方向</b> 根据 T12 计数规则, 该位被置位/清除。 <b>0<sub>B</sub></b> T12 递增计数 <b>1<sub>B</sub></b> T12 递减计数
<b>CTM</b>	7	rw	<b>T12 的工作模式</b> <b>0<sub>B</sub></b> 边沿对齐模式: T12 始终递增计数, 到达周期值之后, 从 0 开始继续计数。 <b>1<sub>B</sub></b> 中心对齐模式: 检测到周期匹配之后, T12 递减计数。检测到 1-匹配之后, T12 递增计数。
<b>T13CLK</b>	[10:8]	rw	<b>定时器 T13 输入时钟选择</b> 根据等式 $f_{T13} = f_{CC6} / 2^{<T13CLK>}$ , 选择定时器 T13 的输入时钟。 <b>000<sub>B</sub></b> $f_{T13} = f_{CC6}$ <b>001<sub>B</sub></b> $f_{T13} = f_{CC6} / 2$ <b>010<sub>B</sub></b> $f_{T13} = f_{CC6} / 4$ <b>011<sub>B</sub></b> $f_{T13} = f_{CC6} / 8$ <b>100<sub>B</sub></b> $f_{T13} = f_{CC6} / 16$ <b>101<sub>B</sub></b> $f_{T13} = f_{CC6} / 32$ <b>110<sub>B</sub></b> $f_{T13} = f_{CC6} / 64$ <b>111<sub>B</sub></b> $f_{T13} = f_{CC6} / 128$
<b>T13PRE</b>	11	rw	<b>定时器 T13 预分频使能位</b>



符号	位序号	类型	功能描述
			<p>为了支持更高时钟频率，可使能 T13 的附加预分频因子 1/256。</p> <p>0<sub>B</sub> T13 的附加预分频被禁止</p> <p>1<sub>B</sub> T13 的附加预分频被使能</p>
<b>T13R</b>	12	rh	<p><b>定时器 T13 运行控制位</b> <sup>2)</sup></p> <p>T13R 启动和终止定时器 T13。软件置位 T13RS 可置位 T13R，软件置位 T13RR 可复位 T13R；或根据位域 T13SSC、T13TEC 和 T13TED 定义的功能硬件置位/清除 T13R。</p> <p>0<sub>B</sub> 定时器 T13 被终止</p> <p>1<sub>B</sub> 定时器 T13 在运行</p>
<b>STE13</b>	13	rh	<p><b>定时器 T13 映射传送使能</b></p> <p>如果发生 T13 映射传送事件，位 STE13 使能或禁止 T13 的周期值、比较值以及被动态选择位和电平从其映射寄存器传送至实际寄存器的映射传送。映射传送之后硬件清除位 STE13。</p> <p>T13 映射传送事件为周期匹配。</p> <p>0<sub>B</sub> 映射寄存器传送被禁止</p> <p>1<sub>B</sub> 映射寄存器传送被使能</p>
<b>0</b>	[15:14]	r	<p><b>保留；</b></p> <p>读操作返回 0；应写入 0。</p>

1) (由 T12SSC, T12RR 或 T12RS) 同时置位/复位 T12R 的操作无效，位 T12R 保持不变。

2) (由 T13SSC, T13TEC, T12RR 或 T13RS) 同时置位/复位 T13R 的操作无效，位 T13R 保持不变。

**捕获/比较单元 6 (CCU6)**

寄存器 **TCTR2** 控制定时器 **T12** 和 **T13** 的单次模式和同步功能。两个定时器都可以工作在单次模式。在该模式，一个计数周期之后计数值为 0，此时定时器自动停止计数。在 **T12** 完成精心定义的 **PWM** 操作之后，**T13** 与 **T12** 的单次模式和同步特性允许产生延迟可编程设定的事件。

**TCTR2**

**定时器控制寄存器 2**

**XSFR (2E<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0			<b>T13 RSEL</b>		<b>T12 RSEL</b>		0		<b>T13 TED</b>		<b>T13 TEC</b>		<b>T13 SSC</b>		<b>T12 SSC</b>
r			rw		rw		r		rw		rw		rw		rw

符号	位序号	类型	功能描述
<b>T12SSC</b>	0	rw	<b>定时器 T12 单次模式控制</b> 该位控制 <b>T12</b> 的单次模式。 0 <sub>B</sub> 单次模式被禁止，硬件不控制 <b>T12R</b> 。 1 <sub>B</sub> 单次模式被使能。如果满足以下条件，位 <b>T12R</b> 被硬件清除： <ul style="list-style-type: none"> <li>– 边沿对齐模式下，<b>T12</b> 达到周期值</li> <li>– 中心对齐模式下，递减计数时，<b>T12</b> 计数至 1。</li> </ul> 清除位 <b>T12R</b> 的同时，位 <b>CC6xST</b> ( $x = 0, 1, 2$ ) 也被清除。
<b>T13SSC</b>	1	rw	<b>定时器 T13 单次模式控制</b> 该位控制 <b>T13</b> 的单次模式。 0 <sub>B</sub> 单次模式被禁止，硬件不控制 <b>T13R</b> 。 1 <sub>B</sub> 单次模式被使能，如果 <b>T13</b> 达到其周期值，位 <b>T13R</b> 被硬件清除。清除 <b>T13R</b> 的同时，位 <b>CC63ST</b> 也被清除。
<b>T13TEC</b>	[4:2]	rw	<b>T13 触发事件控制</b> 根据下面的组合，位域 <b>T13TEC</b> 选择启动 <b>T13</b> 的触发事件（自动置位 <b>T13R</b> 以与 <b>T12</b> 的比较信号同步）。 000 <sub>B</sub> 无操作

**捕获/比较单元 6 (CCU6)**

符号	位序号	类型	功能描述
			001 <sub>B</sub> 由 T12 通道 0 比较事件置位 T13R 010 <sub>B</sub> 由 T12 通道 1 比较事件置位 T13R 011 <sub>B</sub> 由 T12 通道 2 比较事件置位 T13R 100 <sub>B</sub> 由 T12 任意 (通道 0, 1, 2) 比较事件置位 T13R 101 <sub>B</sub> 由 T12 周期匹配事件置位 T13R 110 <sub>B</sub> 由 T12 0-匹配事件 (递增计数时) 置位 T13R 111 <sub>B</sub> 输入 CCPOSx 的任意边沿置位 T13R
<b>T13TED</b>	[6:5]	rw	<b>定时器 T13 触发事件方向<sup>1)</sup></b> 检测到由 T13TEC 定义的触发事件时, 位域 T13TED 给出控制位 T13R 自动置位的附加信息。 00 <sub>B</sub> 保留, 无操作。 01 <sub>B</sub> 当 T12 递增计数时 10 <sub>B</sub> 当 T12 递减计数时 11 <sub>B</sub> 与 T12 的计数方向无关
<b>T12RSEL</b>	[9:8]	rw	<b>定时器 T12 外部运行选择</b> 位域 T12RSEL 定义可硬件置位运行位 T12R 的信号 T12HR 上的事件。 00 <sub>B</sub> 禁止外部置位 T12R。 01 <sub>B</sub> 在 T12HR 上检测到上升沿时, 位 T12R 被置位。 10 <sub>B</sub> 在 T12HR 上检测到下降沿时, 位 T12R 被置位。 11 <sub>B</sub> 在 T12HR 上检测到任意信号沿时, 位 T12R 被置位。
<b>T13RSEL</b>	[11:10]	rw	<b>定时器 T13 外部运行选择</b> 位域 T13RSEL 定义可硬件置位运行位 T13R 的信号 T13HR 上的事件。 00 <sub>B</sub> 禁止外部置位 T13R。 01 <sub>B</sub> 在 T13HR 上检测到上升沿时, 位 T13R 被置位。

**捕获/比较单元 6 (CCU6)**

符号	位序号	类型	功能描述
			<p><b>10<sub>B</sub></b> 在 T13HR 上检测到下降沿时，位 T13R 被置位。</p> <p><b>11<sub>B</sub></b> 在 T13HR 上检测到任意信号沿时，位 T13R 被置位。</p>
<b>0</b>	<b>7,</b> <b>[15:12]</b>	<b>r</b>	<b>保留；</b> 读操作返回 0；应写入 0。

1) 例如:

如果想要由定时器 T12 任意比较事件 (T13TEC=100) 来启动定时器 T13，触发事件方向可被编程为:

- 递增计数>>仅当 T12 递增计数时，T12 通道 0, 1, 2 比较匹配触发 T13R。
- 递减计数>>仅当 T12 递减计数时，T12 通道 0, 1, 2 比较匹配触发 T13R。
- 与 CDIR 无关>>每个 T12 通道 0, 1, 2 比较匹配触发 T13R。

定时器计数方向来自位 CDIR。从而，如果 T12 运行在边沿对齐模式 (仅递增计数)，只有位域 T13TED = 01 或 11，才能自动启动 T13。

**捕获/比较单元 6 (CCU6)**

寄存器 TCTR4 支持软件控制（独立的置位和复位控制条件）运行位 T12R 和 T13R。此外，定时器（运行中）可被复位；位 STE12 和 STE13 可由软件控制，读取这些位始终返回 0。

**TCTR4**

**定时器控制寄存器 4**

**XSFR (26<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>T13 STD</b>	<b>T13 STR</b>	<b>T13 CNT</b>	<b>0</b>	<b>T13 RES</b>	<b>T13 RS</b>	<b>T13 RR</b>	<b>T12 STD</b>	<b>T12 STR</b>	<b>T12 CNT</b>	<b>0</b>	<b>DT RES</b>	<b>T12 RES</b>	<b>T12 RS</b>	<b>T12 RR</b>	
w	w	w	r	w	w	w	w	w	w	r	w	w	w	w	

符号	位序号	类型	功能描述
<b>T12RR</b>	0	w	<b>定时器 T12 运行位复位</b> 置位该位可清除 T12R。 0 <sub>B</sub> T12R 不受影响 1 <sub>B</sub> T12R 被清除，T12 停止计数
<b>T12RS</b>	1	w	<b>定时器 T12 运行位置位</b> 置位该位可置位 T12R。 0 <sub>B</sub> T12R 不受影响 1 <sub>B</sub> T12R 被置位，T12 开始计数
<b>T12RES</b>	2	w	<b>定时器 T12 复位</b> 0 <sub>B</sub> 对 T12 无影响 1 <sub>B</sub> T12 计数寄存器被复位为 0。根据切换规则切换输出信号。置位 T12RES 不影响位 T12R。
<b>DTRES</b>	3	w	<b>死区时间计数器复位</b> 0 <sub>B</sub> 对死区时间计数器无影响 1 <sub>B</sub> 三个死区时间计数器均被清零
<b>T12CNT</b>	5	w	<b>定时器 T12 计数事件</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 如果被使能 (PISELH)，定时器 T12 计数一次。
<b>T12STR</b>	6	w	<b>定时器 T12 映射传送请求</b>

符号	位序号	类型	功能描述
			0 <sub>B</sub> 无操作 1 <sub>B</sub> STE12 被置位，使能映射传送
<b>T12STD</b>	7	w	<b>定时器 T12 映射传送禁止</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> STE12 被清除，不触发映射传送
<b>T13RR</b>	8	w	<b>定时器 T13 运行位复位</b> 置位该位可清除 T13R。 0 <sub>B</sub> T13R 不受影响 1 <sub>B</sub> T13R 被清除，T13 停止计数
<b>T13RS</b>	9	w	<b>定时器 T13 运行位置位</b> 置位该位可置位 T13R。 0 <sub>B</sub> T13R 不受影响 1 <sub>B</sub> T13R 被置位，T13 开始计数
<b>T13RES</b>	10	w	<b>定时器 T13 复位</b> 0 <sub>B</sub> 对 T13 无影响 1 <sub>B</sub> T13 计数寄存器被复位为 0。根据切换规则切换输出信号。置位 T13RES 不影响位 T13R。
<b>T13CNT</b>	13	w	<b>定时器 T13 计数事件</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 如果被使能 (PISELH)，定时器 T13 计数一次。
<b>T13STR</b>	14	w	<b>定时器 T13 映射传送请求</b> 0 无操作 1 STE13 被置位，使能映射传送
<b>T13STD</b>	15	w	<b>定时器 T13 映射传送禁止</b> 0 无操作 1 STE13 被清除，不触发映射传送
<b>0</b>	4, [12:11]	r	<b>保留；</b> 读操作返回 0；应写入 0。

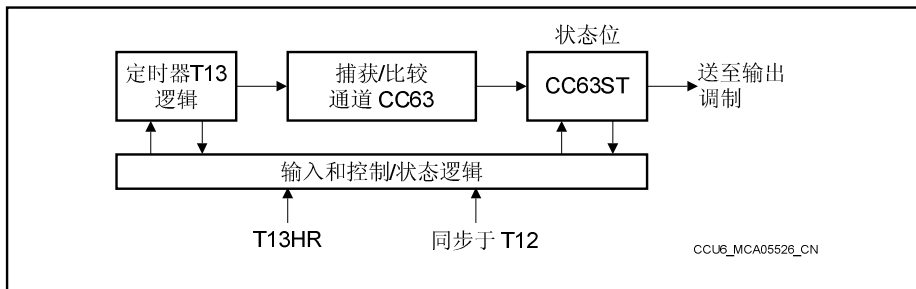
1) 同时向置位控制位和复位控制位写 1 不会触发任何操作，相关位保持不变。

## 20.3 定时器 T13 模块

定时器 T13 和定时器 T12 类似，但只有一路比较通道。一个 16 位的递增计数器通过比较器和通道寄存器相连，计数器的值和通道寄存器的值匹配时，输出匹配信号。CCU6 单元提供了多种控制功能，以便 T13 模块适用于不同的应用需求。此外，T13 可由定时器 T12 的事件同步触发。

本节提供下述内容：

- T13 模块概述（见**章节 20.3.1**）
- 计数策略（见**章节 20.3.2**）
- 比较模式（见**章节 20.3.3**）
- 比较输出通路（见**章节 20.3.4**）
- 映射寄存器传送（见**章节 20.3.5**）
- T13 计数寄存器描述（见**章节 20.3.6**）



**图 20-23 定时器 T13 模块总览框图**

### 20.3.1 T13 模块概述

**图 20-24** 所示为定时器 T13 的内部结构框图。由寄存器 **TCTR0**、**TCTR2**、**TCTR4** 和 **PISELH** 控制定时器 T13 模块。

定时器 T13 的输入时钟  $f_{T13}$  来自经过预分频处理的模块时钟  $f_{CC6}$ （预分频因子可编程设定）和 1/256 分频（可选）。T13 只能递增计数（类似于 T12 的边沿对齐模式）。

计数器寄存器 **T13** 通过比较器和周期寄存器 **T13PR** 相连。该寄存器中存放着 T13 的最大计数值。T13 计数到周期值后，在下一个 T13 的时钟沿，产生信号 **T13\_PM**（T13 周期匹配）并将 T13 复位为 0000<sub>H</sub>。周期寄存器的新周期值从映射周期寄存器 **T13PS** 中获取，T13PS 的值由软件载入。“T13 映射传送”控制信号 **T13\_ST** 控制将新周期值从映射寄存器传送到 T13PR 中的操作，该信号的产生取决于控制位 **STE13**。

## 捕获/比较单元 6 (CCU6)

CCU6 单元提供了周期值、以及（和产生 PWM 信号相关的）其它数值的映射寄存器，便于软件同步更新所有相关参数（请参阅[章节 20.3.5](#)）。

信号 T13\_ZM 指示计数器的值是否等于 0000<sub>H</sub>。

单次模式控制位 T13SSC 控制定时器在当前计数周期结束后自动停止运行（见[图 20-26](#)）。

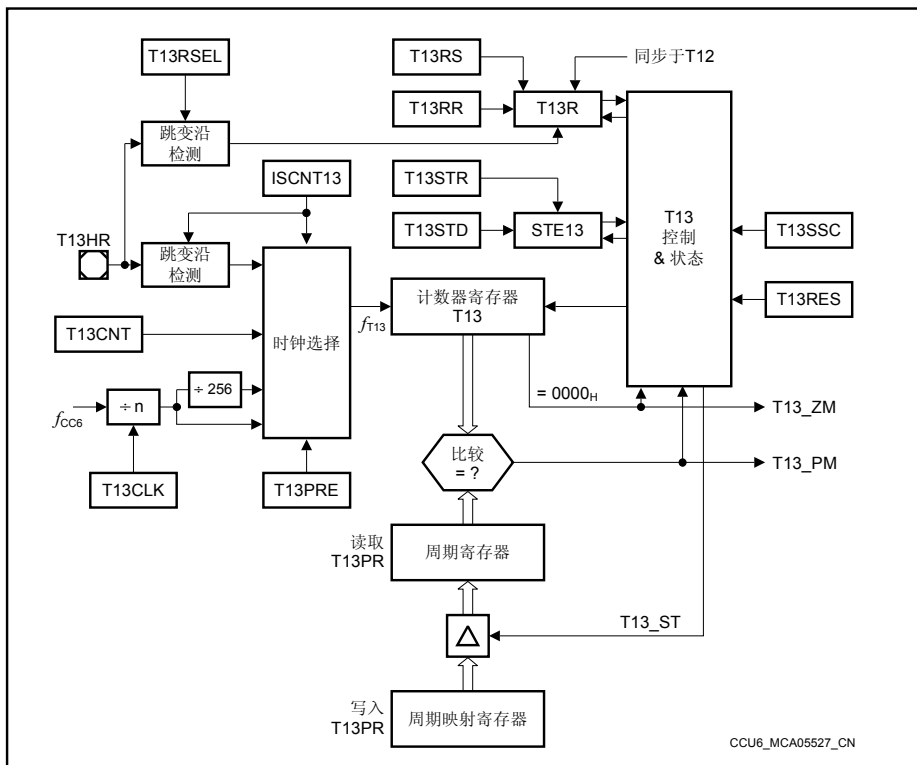


图 20-24 T13 计数器逻辑和周期比较器

运行位 T13R 控制启动或终止定时器 T13 工作。T13R 可由相关置位控制位 T13RS 或复位控制位 T13RR 设定（T13RR 和 T13RS 位于寄存器 [TCTR4](#) 中）；或根据预先选定的条件硬件复位（单次模式）。

当设置的 T13 周期值为 0 时，一定不能置位定时器 T13 运行位 T13R。如果检测到 T12 事件与 T13 时序同步，则位 T13R 可被自动置位，例如 T12 比较通道的信号沿之



后，触发 AD 转换操作之前，由 T13 产生一个可编程设定的延迟（T13 能触发 ADC 转换）。

定时器 T13 可由控制位 T13RES 清零。置位“只写”控制位 T13RES 仅清除定时器的内容、不会产生其它影响，如不会终止定时器工作。

T13 映射传送控制信号 T13\_ST 由位 STE13 控制产生，该控制位可通过相关置位控制位 T13STR，或复位控制位 T13STD 间接设置。

位域 T13TEC 和 T13TED 控制 T13 与定时器 T12 的事件同步。T13TEC 选择触发事件；T13TED 选择使触发事件有效的定时器 T12 计数方向。

定时器 T13 运行时，对计数寄存器 T13 的写操作无效。如果 T13 停止，对寄存器 T13 的写操作立即生效。

*注：T13 的周期寄存器和对应的映射寄存器占用相同的物理地址。写操作对应写入映射寄存器；读操作对应读取实际周期寄存器的值。*

## 20.3.2 T13 计数策略

本节描述 T13 的时钟和计数功能。

### 20.3.2.1 时钟选择

定时器模式下（**PISELH.ISCNT13** = 00<sub>B</sub>），定时器 T13 的输入时钟  $f_{T13}$  来自经过预分频处理的模块时钟  $f_{CC6}$ （预分频因子可编程设定）和 1/256 分频（可选）。预分频因子列于表 20-6 中。定时器 T13 不工作时（**TCTR0.T13R** = 0），预分频器复位，从而保证产生可重复的时序和延迟。

表 20-6 定时器 T13 输入时钟选择

T13CLK	输入时钟 $f_{T13}$ 预分频器关闭（T13PRE = 0）	输入时钟 $f_{T13}$ 预分频器开启（T13PRE = 1）
000 <sub>B</sub>	$f_{CC6}$	$f_{CC6}/256$
001 <sub>B</sub>	$f_{CC6}/2$	$f_{CC6}/512$
010 <sub>B</sub>	$f_{CC6}/4$	$f_{CC6}/1024$
011 <sub>B</sub>	$f_{CC6}/8$	$f_{CC6}/2048$
100 <sub>B</sub>	$f_{CC6}/16$	$f_{CC6}/4096$
101 <sub>B</sub>	$f_{CC6}/32$	$f_{CC6}/8192$
110 <sub>B</sub>	$f_{CC6}/64$	$f_{CC6}/16384$
111 <sub>B</sub>	$f_{CC6}/128$	$f_{CC6}/32768$

在计数器模式下，定时器 T13 计数一步：

- 如果向 **TCTR4.T13CNT** 写入 1 且 **PISELH.ISCNT13 = 01<sub>B</sub>**
- 如果检测到输入信号 T13HR 的上升沿且 **PISELH.ISCNT13 = 10<sub>B</sub>**
- 如果检测到输入信号 T13HR 的下降沿且 **PISELH.ISCNT13 = 11<sub>B</sub>**

### 20.3.2.2 定时器 T13 计数操作

定时器周期由周期寄存器 T13PR 决定，计算公式如下：

$$T13_{PER} = \langle \text{周期寄存器的值} \rangle + 1; \text{以 } T13 \text{ 时钟频率 (} f_{T13} \text{) 计数} \quad (20.3)$$

T13 只能递增计数，类似于 T12 的边沿对齐模式，因此 T13 计数器的“计数规则”非常简单：

- 检测到周期匹配时，下一个 T13 时钟沿计数器复位为 0。T13 始终递增计数。

T13 的操作如图 20-25 所示。

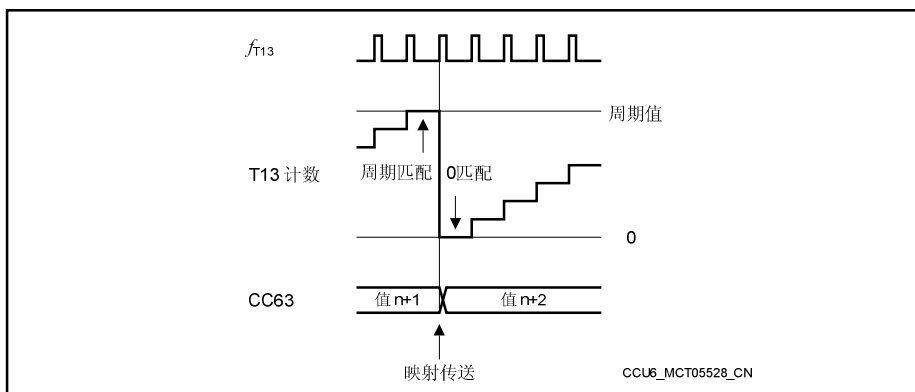
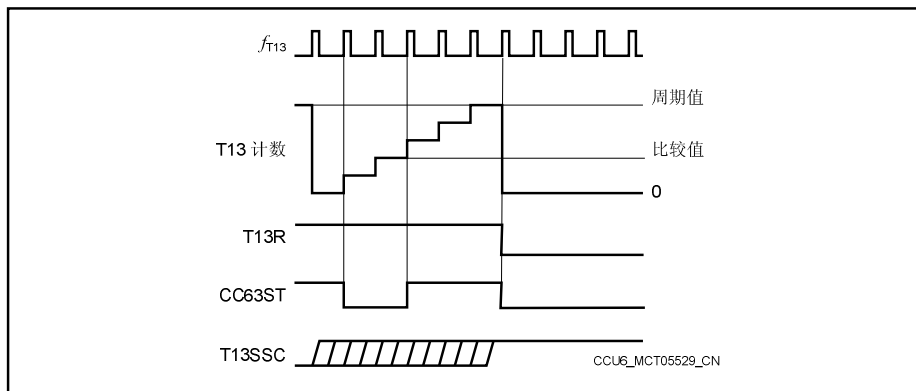


图 20-25 T13 计数序列

### 20.3.2.3 单次模式

单次模式下，运行位 T13R 可由硬件清除。如果位 T13SSC = 1，当前定时器周期结束之后，定时器 T13 停止。

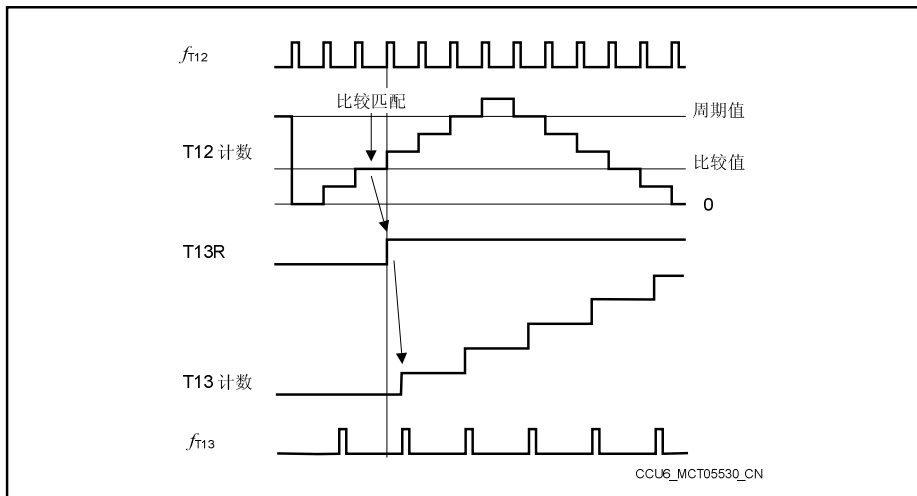


**图 20-26 定时器 T13 的单次操作**

#### 20.3.2.4 与 T12 同步

定时器 T13 可与 T12 事件同步。位域 T13TEC 和 T13TED 选择触发事件，用来启动定时器 T13 工作。所选择的事件硬件置位 T13R，T13 开始计数。该特性和单次模式相结合，可在 T12 事件之后产生可编程的时延。

**图 20-27** 举例说明 T13 如何与 T12 的事件同步，图中所选择的事件为递增计数时的比较匹配（比较值 = 2）。T12 和 T13 的输入时钟可以不同（选择其它预分频因子），图中所示  $f_{T13}$  为  $f_{T12}$  的一半。



**图 20-27 T13 和 T12 比较匹配事件同步**

位域 T13TEC 选择启动 T13 的触发事件（自动置位 T13R 以与 T12 的比较信号同步），事件选择列于表 20-7 中；另外由 T13TED 选择使触发事件生效的定时器 T12 的计数方向（见表 20-8）。

**表 20-7 T12 触发事件选择**

<b>T13TEC</b>	<b>事件选择</b>
000 <sub>B</sub>	无
001 <sub>B</sub>	通道 0 上的 T12 比较事件 (CM_CC60)
010 <sub>B</sub>	通道 1 上的 T12 比较事件 (CM_CC61)
011 <sub>B</sub>	通道 2 上的 T12 比较事件 (CM_CC62)
100 <sub>B</sub>	任意通道 (0, 1, 2) 上的 T12 比较事件
101 <sub>B</sub>	T12 周期匹配 (T12_PM)
110 <sub>B</sub>	T12 递增计数时 0-匹配 (T12_ZM 且 CDIR = 0)
111 <sub>B</sub>	霍尔状态的任何变化

**表 20-8 T12 触发事件附加条件**

<b>T13TED</b>	<b>事件选择附加条件</b>
00 <sub>B</sub>	保留, 无操作
01 <sub>B</sub>	T12 递增计数时选定的触发事件有效 (CDIR = 0)
10 <sub>B</sub>	T12 递减计数时选定事件的触发有效 (CDIR = 1)
11 <sub>B</sub>	无论 T12 递增/递减计数, 选定的触发事件均有效

### 20.3.3 T13 的比较模式

定时器 T13 的单路比较通道可执行 (和 T13 的计数值有关的) 比较操作。

**图 20-23** 给出比较模式下 T13 的通道操作。T13 通道通过等于比较器和 T13 的计数寄存器相连, 当计数器的值和比较寄存器中的值匹配时产生匹配信号。

通道由比较器和双寄存器结构 (实际比较寄存器 **CC63R** 和相关映射寄存器 **CC63SR**) 组成。CC63R 与比较器直接相连、CC63SR 由软件预先加载。当 T13 的映射传输信号 T13\_ST 有效时, 映射寄存器中的内容被传送到实际比较寄存器中。CCU6 单元提供了比较值、以及 (和产生 PWM 信号相关的) 其它数值的映射寄存器, 便于用户编程同步更新所有相关参数。

T13 的通道状态位 **CMPSTAT.CC63ST** 用于保存比较操作的状态, **图 20-28** 给出状态位的逻辑操作。

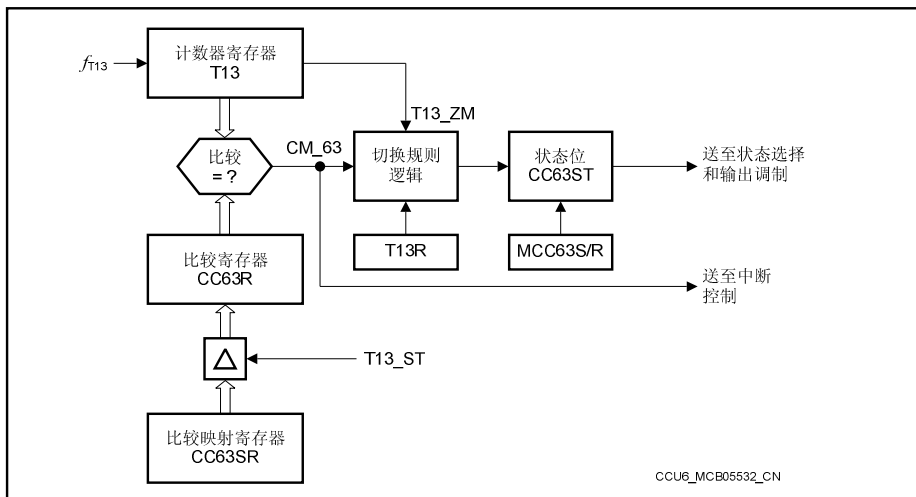


图 20-28 T13 状态位框图

检测到比较匹配时，产生比较中断事件 CM\_63。状态位的实际设置对中断产生无影响。

状态位 CC63ST 置位/复位控制逻辑的输入包括：定时器运行位 (T13R)，定时器 0-匹配信号 (T13\_ZM)，以及实际比较匹配信号 CM\_63。此外，可通过设定置位控制位 MCC63S 和复位控制位 MCC63R（位于寄存器 CMPMODIF 中）相应置位或复位 CC63ST。

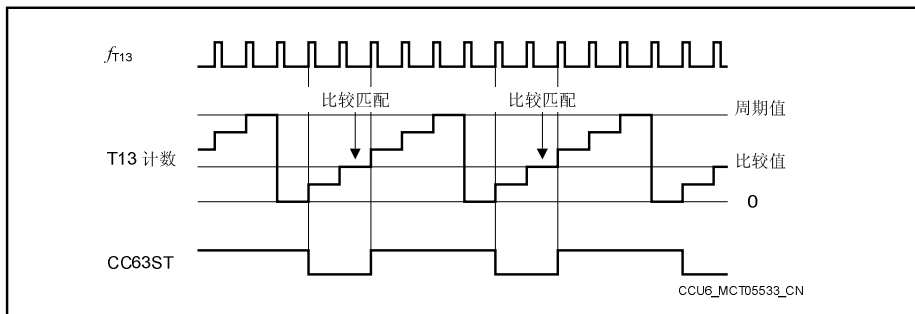
只有当定时器 T13 工作时 (T13R = 1)，才可硬件修改状态位 CC63ST。硬件修改状态位时，应用以下切换规则置位和复位状态位：

状态位 CC63ST 被置位为 1：

- 比较匹配时 (T13 始终递增计数)，下一个 T13 时钟 ( $f_{T13}$ ) 置位 CC63ST (即，计数器递增计数到超过比较值)；
- 0-匹配且同时为比较匹配时，下一个 T13 时钟 ( $f_{T13}$ ) 置位 CC63ST。

状态位 CC63ST 被复位为 0：

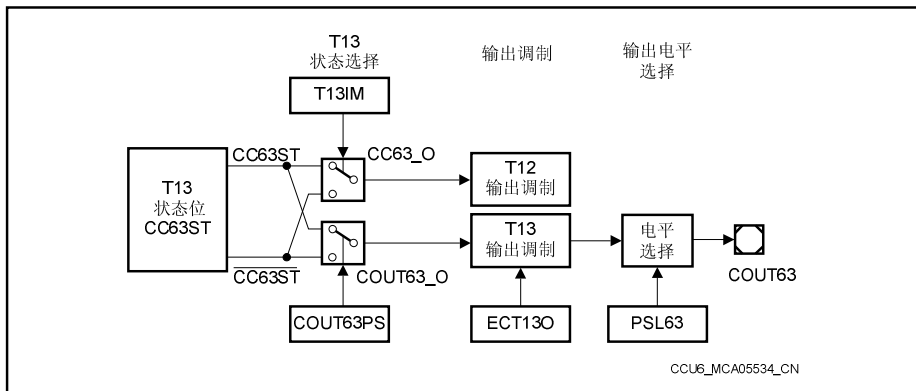
- 0-匹配但同时不是比较匹配时，下一个 T13 时钟 ( $f_{T13}$ ) 复位 CC63ST。



**图 20-29 T13 的比较操作**

### 20.3.4 比较模式输出路径

**图 20-30** 给出从通道状态位 CC63ST 到其输出引脚 COUT63 的信号路径。如图所示，用户可通过多种控制选择，决定当前状态位 CC63ST 所对应的（期望）输出信号的切换操作。T12 信号输出调制控制的具体内容请参阅**章节 20.2.4.3**。



**图 20-30 通道 CC63 输出路径**

输出线 COUT63\_O 可产生 T13 的 PWM 信号，从引脚 COUT63 输出。信号 CC63\_O 产生的 T13 PWM 信号用来调制 T12 的相关输出信号。为了使 COUT63 和内部调制信号无关，可分别由 T13IM 和 COUT63PS 选择引发有效信号的比较状态。

数据最后送入输出调制模块。因此，T13 的调制源和强制中断功能相结合，用来控制输出引脚 COUT63 的实际电平（见**图 20-31**）：

- **T13 相关比较信号** COUT63\_O 由 T13 状态选择输出，由 **MODCTR.ECT13O** 使能。
- **强制中断状态** TRPS 具有单独使能位 **TRPCTR.TRPEN13**

如果调制输入信号 COUT63\_O 被使能（ECT13O = 1）且处于被动态，被调制的信号也处于被动态。如果调制输入未被使能，输出处于被动态。

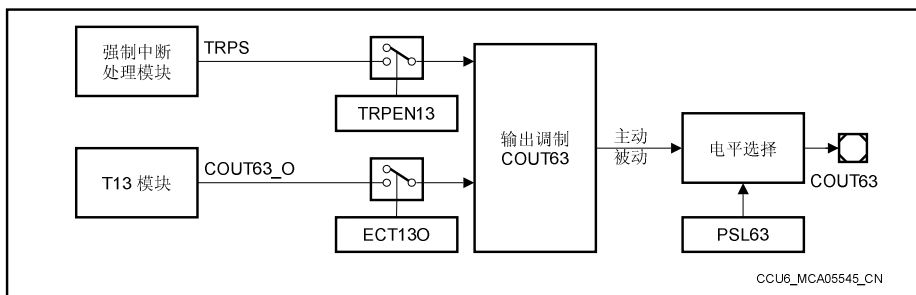
如果强制中断状态有效（TRPS = 1），那么使能（由 TRPEN13 = 1）的强制中断信号输出被设置为被动态。

调制控制模块的输出和一个电平选择模块相连。根据被动态选择位 **PSLR.PSL63** 确定的输出线的状态（使得主动/被动态和输出极性无关），通过该模块选择并决定引脚的实际输出电平。如果被调制的输出信号处于被动态，直接输出 **PSL63** 电平。如果被



调制的输出处于主动态，输出 PSL63 的反向电平。该特性使得用户能够根据所连接的电路调整主动态输出信号的极性。

控制位 PSL63 对应有映射寄存器，从而由 T13 的映射传送信号 (T13\_ST) 触发的 PSL63 更新时避免了输出线上多余的脉冲。读操作读取实际使用值；写操作写入映射位。CCU6 单元提供了 PSL 值、以及 (和产生 PWM 信号相关的) 其它数值的映射寄存器，便于软件同步更新所有相关参数。



**图 20-31 T13 输出调制**

### 20.3.5 T13 映射寄存器传送

采用一个特殊的映射传送信号 (T13\_ST) 便于使比较通道 CC63 周期值和比较值的更新和 T13 的操作同步。映射寄存器使得软件能够同时更新定义 PWM 周期的所有相关参数。下一个 PWM 周期可使用一组新的参数。软件通过位 **TCTR0.STE13** (向只写位 **TCTR4.T13STR** 写入 1 将置位 STE13; 向只写位 **TCTR4.T13STD** 写入 1 将清除 STE13)。

当信号 T13\_ST 有效时，在下一个 T13 时钟周期触发映射寄存器传送。映射传送之后位 STE13 自动清零。

在以下条件下，发生 T13 映射寄存器传送 (T13\_ST 有效)：

- 定时器 T13 不运行 (T13R = 0)，或
- STE13 = 1 且 T13R = 1 时检测到周期匹配

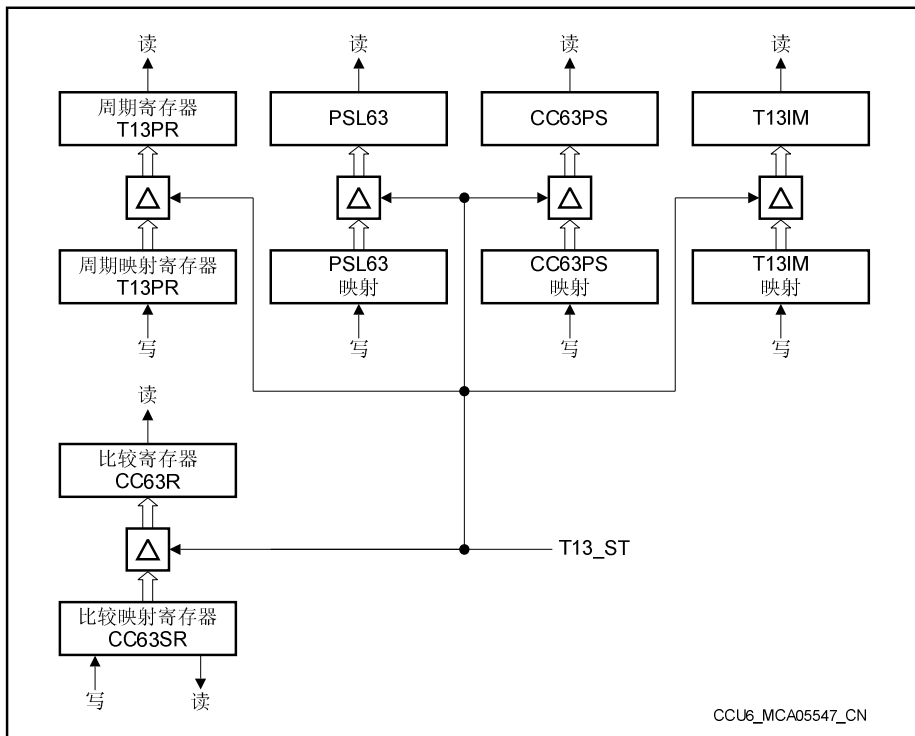


图 20-32 T13 映射寄存器概览

## 20.3.6 T13 相关寄存器

### 20.3.6.1 T13 计数寄存器

基于定时器 T13 产生单个通道脉宽调制 (PWM) 信号的序列。相关的定时器 T13 寄存器可同时更新 (以精心定义的条件) 以确保 PWM 信号的一致性。T13 还可和多个定时器 T12 事件同步。

定时器 T13 仅支持比较通道 CC63 的比较模式。

寄存器 T13 的内容显示了当前定时器 T13 的计数值。只有在定时器 T13 停止时，才可对寄存器 T13 进行写操作；定时器 T13 工作时，写操作无效。寄存器 T13 始终可由软件读取。

定时器 T13 仅支持边沿对齐模式 (递增计数)。

### T13

定时器 T13 计数寄存器

XSFR (30<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13CV															
rwh															

符号	位序号	类型	功能描述
T13CV	[15:0]	rwh	定时器 T13 的计数值 定时器 T13 的 16 位计数值

注：当定时器 T13 停止时，内部时钟分频器复位以确保产生可重复的时序和延迟。

### 20.3.6.2 周期寄存器

寄存器 T13PR 中存放着定时器 T13 的周期值。周期值和 T13 的当前计数值进行比较，由计数规则决定计数器的后续操作。该寄存器对应有一个映射寄存器，由位 STE13 控制映射传送。读操作读取当前用于比较操作的周期值；写操作写入映射寄存器；映射寄存器结构便于用户同时更新 T13 的所有相关值。

#### T13PR

定时器 T13 周期寄存器						XSFR (32 <sub>H</sub> )						复位值: 0000 <sub>H</sub>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
T13PV															
rwh															

符号	位序号	类型	功能描述
<b>T13PV</b>	[15:0]	rwh	<b>定时器 T13 的周期值</b> T13 的计数值和 T13PV 相等时触发周期匹配。计数到该周期值，定时器 T13 被清零。

### 20.3.6.3 比较寄存器

寄存器 CC63R 是 T13 的实际比较寄存器。保存在 CC63R 中的值和 T13 的计数值进行比较。状态位 CC63ST 位于寄存器 **CMPSTAT** 内。

#### CC63R

T13 比较寄存器						XSFR (34 <sub>H</sub> )						复位值: 0000 <sub>H</sub>			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<div>CCV</div>															
rh															

符号	位序号	类型	功能描述
<b>CCV</b>	[15:0]	rh	<b>通道 CC63 的比较值</b> 位域 CCV 中存放着和 T13 计数值进行比较的值。

### 20.3.6.4 比较映射寄存器

寄存器 CC63R 只能由软件读取，若要改变 CC63R 的值，需要将映射寄存器 CC63SR 中的修改值映射传送到 CC63R 中。对应的映射寄存器可由软件读写。

#### CC63SR

#### T13 比较映射寄存器

**XSFR (36<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CCS</b>															
rw															

符号	位序号	类型	功能描述
<b>CCS</b>	[15:0]	rw	通道 <b>CC63</b> 比较值映射寄存器 映射传送期间，位域 <b>CCS</b> 的内容被传送到位域 <b>CCV</b> 中。

## 20.4 强制中断处理

强制中断功能使 PWM 输出能够响应输入信号  $\overline{\text{CTRAP}}$  的变化。如果强制中断输入有效，可使用该功能关闭功率器件（如，执行紧急停止）。强制中断处理和输出调制的结果由强制中断控制寄存器 **TRPCTR** 中的位控制。强制中断标志 **TRPF** 和 **TRPS** 位于寄存器 **IS** 中，通过对寄存器 **ISS** 和 **ISR** 进行写操作可软件置位/清除这些标志。

图 20-33 给出强制中断功能的概览。

强制中断标志 **TRPF** 监控强制中断输入的变化并启动进入强制中断状态。强制中断状态位 **TRPS** 决定输出结果并控制强制中断状态的退出。

检测到强制中断条件（ $\overline{\text{CTRAP}} = 0$ ）且输入被使能（ $\text{TRPPEN} = 1$ ）时，强制中断标志 **TRPF** 和强制中断状态位 **TRPS** 均被置 1（强制中断状态有效）。强制中断状态 **TRPS** 的输出送入输出调制模块（**T12** 和 **T13**），且可使输出无效（将输出设置为被动态）。CCU6 单元分别为 **T12** 的六路输出和 **T13** 的单路输出提供了对应的强制中断使能控制位，以灵活适应不同的应用需求。

有多种退出强制中断状态的方式，可软件为给定应用选择最佳方式。退出强制中断状态的方式有：强制中断条件撤销后（ $\overline{\text{CTRAP}} = 1$  或  $\text{TRPPEN} = 0$ ）立即退出、软件控制退出、或者使退出和（由定时器 **T12** 或 **T13** 产生的）PWM 信号同步。

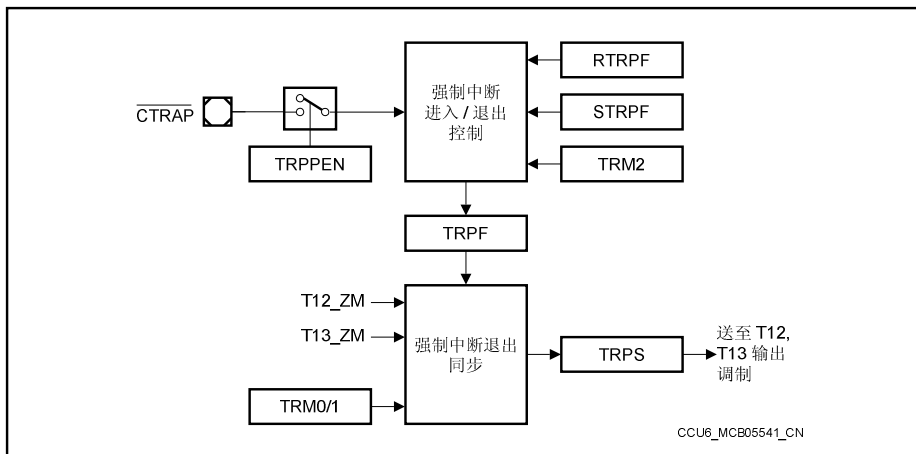
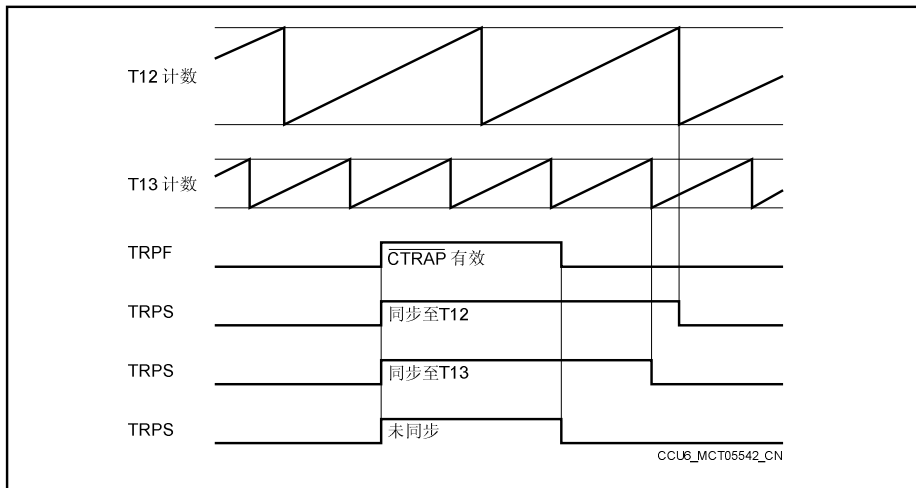


图 20-33 强制中断逻辑框图

模式控制位 **TRPM2** 可控制 **TRPF** 的复位操作。若  $\text{TRPM2} = 0$ ， $\overline{\text{CTRAP}}$  返回无效电平时（ $\overline{\text{CTRAP}} = 1$ ），或如果强制中断输入被禁止（ $\text{TRPPEN} = 0$ ），**TRPF** 由硬件自动清零；若  $\text{TRPM2} = 1$ ， $\overline{\text{CTRAP}}$  变为无效之后，**TRPF** 必须由软件复位。

模式控制位 **TRPM1** 和 **TRPM0**（位于强制中断控制寄存器 **TRPCTR** 中）控制 **TRPS** 复位。复位 **TRPS** 将终止强制中断状态并返回正常操作。有三种方式复位 **TRPS**，分别由 **TRPM1** 和 **TRPM0** 控制选择：一种方式为当强制中断标志 **TRPF** 被清零后，立即退出强制中断状态，不和定时器 **T12** 或 **T13** 同步；另两种方式使退出强制中断状态分别和定时器 **T12** 或 **T13** 的计数周期同步。[图 20-34](#) 给出相关操作。



**图 20-34 强制中断状态同步 (TRPM2 = 0)**

## 20.5 多通道模式

多通道模式可用一条指令调制全部六路 **T12** 的相关输出信号。位域 **MCMOUT.MCMP** 规定哪些输出有效。若多通道模式被使能（位 **MODCTR.MCMEN** = 1），位域 **MCMP** 中被置 1 的位对应的输出才有效。

该位域对应的映射位域为 **MCMOUTS.MCMPS**，可由软件写入。可由 **T12** 或 **T13** 事件触发将 **MCMP** 中的新值传送到位域 **MCMP** 中，该映射传送操作和 **T12** 或 **T13** 事件同步。该结构支持由软件写入新值，在定义好的时刻发生映射传送，使映射传送和 **PWM** 信号同步。这样可避免由不同步的调制源造成的多余脉冲。

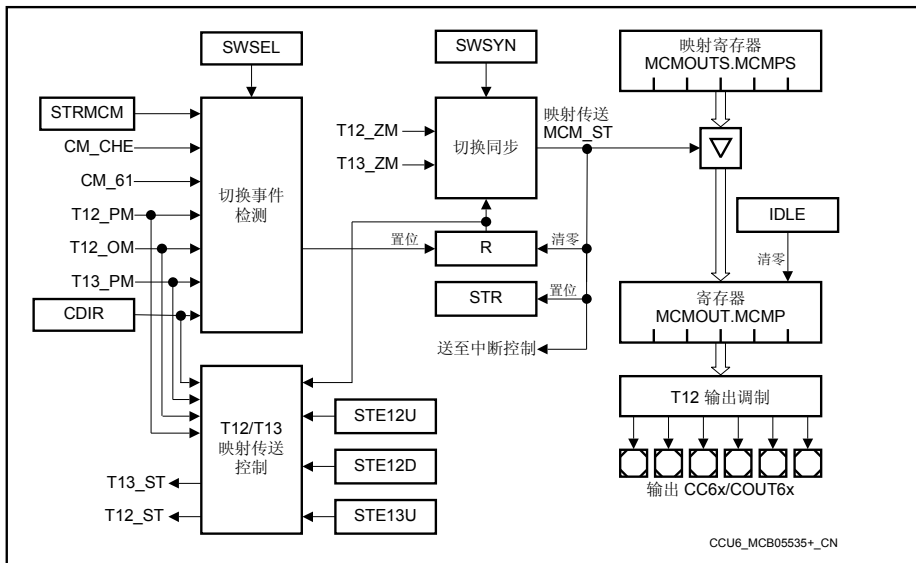


图 20-35 多通道模式框图

图 20-35 所示为多通道操作的功能框图，由寄存器 **MCMCTR** 中的位域控制。由 **SWSEL** 选择触发位域 **MCMP** 更新的事件。为了使 **MCMP** 更新和 **T12** 或 **T13** 产生的 **PWM** 信号同步，由位域 **SWSYN** 选择引发映射传送（从 **MCMPS** 至 **MCMP**）的同步事件。该逻辑结构决定了在新的 **PWM** 周期更新 **MCMP**。选定的切换事件发生时（该事件不必与调制的 **PWM** 同步），提示标志 **R** 被置位；映射传送后 **R** 被复位。软件监控该标志以检查该逻辑的状态。如果发生从 **MCMPS** 到 **MCMP** 的映射传送，位 **IS.STR** 被置位并产生一个中断请求。

除了多通道映射传送事件 **MCM\_ST**，可产生 **T12** 映射传送（**T12\_ST**）和 **T13** 映射传送（**T13\_ST**）事件以允许同时更新 **T12** 和/或 **T13** 调制占空比以及多通道序列。

如有需要，当选择直接同步模式时，选定的触发事件发生时立刻更新 **MCMP**。也可由软件请求更新 **MCMP**，将更新值写入到 **MCMPS** 且设置映射传送请求位 **STRMCM** = 1。对应于所有 **SWSEL** 设置，都可选择软件触发 **MCMP** 更新。

若使用直接模式且位 **STRMCM** = 1，则 **MCMP** 的更新完全由软件控制。

触发事件选择和同步事件选择归纳于表 20-9 和表 20-10 中。



表 20-9 多通道模式切换事件选择

SWSEL	选择的事件（见寄存器 <b>MCMCTR</b> ）
000 <sub>B</sub>	无自动事件检测
001 <sub>B</sub>	输入信号 CCPOSx 上的检测到的正确霍尔事件（CM_CHE），无附加延迟
010 <sub>B</sub>	T13 周期匹配（T13_PM）
011 <sub>B</sub>	T12 递减计数时 1-匹配（T12_OM 且 CDIR = 1）
100 <sub>B</sub>	T12 递增计数时比较通道 1 的比较事件（CM_61 和 CDIR = 1），CC61 支持相位延迟功能以实现块切换模式。
101 <sub>B</sub>	T12 递增计数时周期匹配（T12_PM 且 CDIR = 0）
110 <sub>B</sub> , 111 <sub>B</sub>	保留，无操作

表 20-10 多通道模式切换同步

SWSYN	同步事件（见寄存器 <b>MCMCTR</b> ）
00 <sub>B</sub>	直接模式：触发事件直接引发映射传送
01 <sub>B</sub>	T13 的 0-匹配（T13_ZM）， MCM 映射传送与 T13 PWM 同步
10 <sub>B</sub>	T12 的 0-匹配（T12_ZM）， MCM 映射传送与 T12 PWM 同步
11 <sub>B</sub>	保留，无操作

## 20.6 霍尔传感器模式

对于工作在块切换模式的无刷直流电机，通常使用多通道模式以产生高效的调制序列控制电机正确工作。调制序列必须根据电机的角度位置相应产生。通常使用霍尔传感器或反电动势检测来决定转子的角度位置。CCU6 的三个输入，CCPOS0、CCPOS1 和 CCPOS2 可用作霍尔传感器或反电动势检测信号的输入。

电机转动位置和输出调制序列之间紧密关联。当电机转动到某个位置时，由霍尔传感器的输入采样（霍尔序列）表示当前位置，接下来必须输出预先确定的多通道调制序列。电机类型不同，驱动电机的调制序列有所不同。因此，希望能够比较灵活的定义霍

## 捕获/比较单元 6 (CCU6)

尔序列和相应调制序列之间的关联性。另外，CCU6 的硬件机制大大的减少了块切换操作时的 CPU 负荷。

CCU6 通过特定的寄存器实现了该特性，该寄存器中存放着当前实际的霍尔序列 (CURH)、下次期望的霍尔序列 (EXPH)、以及相应的输出序列 (MCMP)。当采样的霍尔序列和期望序列 (EXPH) 匹配时，输出新的调制序列。CCU6 监控霍尔输入的变化以检测电机下一次的转动相位（块切换的某段，两两导通的某个状态）。检测到期望的霍尔序列时，输出相应的调制序列。

CCU6 引入霍尔输入的采样时延，从而可增加（在一定程度上）抗噪声能力。此外，霍尔序列采样和当前霍尔序列 (CURH) 进行比较，从而可容忍短毛刺的存在。霍尔序列比较逻辑将霍尔输入和下次的期望序列以及当前霍尔序列进行比较以滤除毛刺。

霍尔传感器模式下，通过双寄存器结构控制霍尔和调制序列的产生。寄存器 **MCMOUT** 中存放着实际使用的值；对应的映射寄存器 **MCMOUTS** 中存放着可由软件加载的、取自预定义表的正确的霍尔和调制序列（针对给定电机）。

当检测到正确的霍尔序列跳变时，即霍尔序列采样和期望序列相匹配时，映射寄存器的内容映射传送到寄存器 **MCMOUT** 中。接着由软件将新值装入 **MCMOUTS** 中。从 **MCMOUTS** 到 **MCMOUT** 的映射传送也可由软件强制执行。

注：霍尔输入信号 **CCPOSx** 和 **CURH** 以及 **EXPH** 位域对应关系如下：

**CCPOS0** 与 **CURH.0 (LSB)** 和 **EXPH.0 (LSB)** 对应

**CCPOS1** 与 **CURH.1** 和 **EXPH.1** 对应

**CCPOS2** 与 **CURH.2 (MSB)** 和 **EXPH.2 (MSB)** 对应

### 20.6.1 霍尔序列评估

跳变沿检测模块以模块时钟  $f_{CC6}$  永久监控霍尔传感器的输入 CCPOSx。为了抑制恶劣的逆变器工作环境引起的霍尔输入上的毛刺，霍尔逻辑提供两种可选的噪声滤波方法（两种方法可组合使用）。

- 带延迟的噪声滤波：

选择此方法时，要求所有 T12 比较通道的模式控制位域 MSEL6x 必须编程为 1000<sub>B</sub> 且 DBYP = 0。所选择的事件触发死区时间计数器 0 产生一个可编程设置的延迟（由位域 DTM 定义）。当延迟到期之后，评估信号 HCRDY 被激活。

此模式不可能实现带 T12 PWM 信号的输出调制。

- 与 PWM 同步的噪声滤波：

跳变沿检测模块并非永久监控霍尔输入，而是在 PWM 周期内的确定时间点对其采样。当切换噪声（由 PWM 引起的）不会干扰霍尔输入信号时，使用该方法采样霍尔输入。

如果既不使用死区时间计数器 0 的延迟功能进行霍尔序列评估，也不使能霍尔模式进行无刷直流电机控制，则可使用定时器 T12 模块进行 PWM 产生和输出调制。

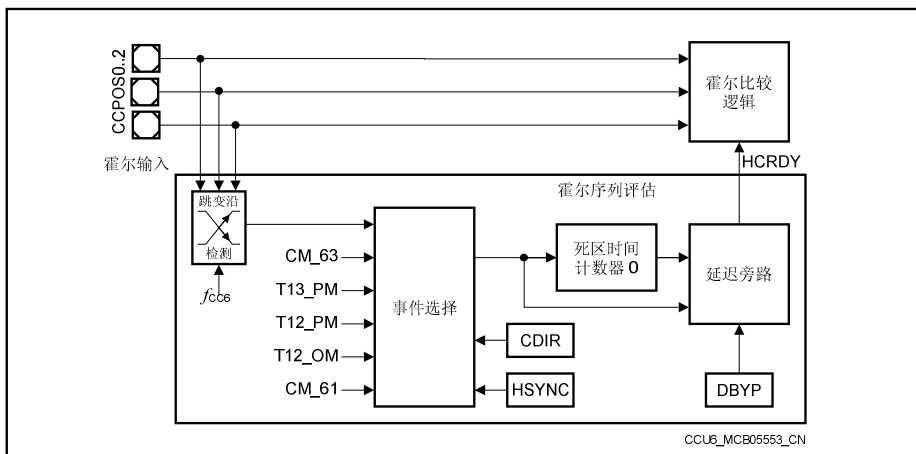


图 20-36 霍尔序列评估

如果评估信号 HCRDY（霍尔比较准备就绪，见图 20-37）被激活，霍尔输入被采样，且霍尔比较逻辑开始评估霍尔输入。

图 20-36 给出霍尔序列评估事件和噪声滤波逻辑。

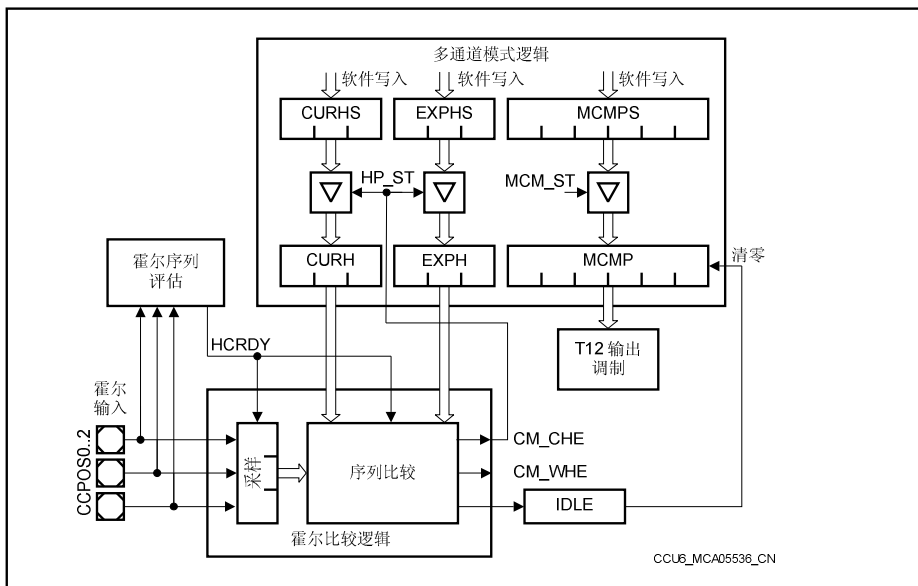
表 20-11 总结出可选的触发输入信号。

**表 20-11      霍尔传感器模式触发事件选择**

HSYNC	事件选择 (见寄存器 <b>T12MSEL</b> )
000 <sub>B</sub>	CCPOSx 输入上的任意跳变沿, 与任何 PWM 信号无关 (一直检查)
001 <sub>B</sub>	T13 比较匹配 (CM_63)
010 <sub>B</sub>	T13 周期匹配 (T13_PM)
011 <sub>B</sub>	由硬件源触发的霍尔采样被关闭
100 <sub>B</sub>	T12 递增计数时的周期匹配 (T12_PM 且 CDIR = 0)
101 <sub>B</sub>	T12 递减计数时的 1-匹配 (T12_OM 且 CDIR = 1)
110 <sub>B</sub>	递增计数时, 比较通道 CC61 的 T12 比较匹配 (CM_61 且 CDIR = 0)
111 <sub>B</sub>	递减计数时, 比较通道 CC61 的 T12 比较匹配 (CM_61 且 CDIR = 1)

## 20.6.2 霍尔序列比较逻辑

**图 20-37** 给出双寄存器结构和序列比较逻辑。新调制序列 (MCMPs)、当前霍尔序列 (CURHS) 和期望霍尔序列 (EXPHS) 软件写入映射寄存器 MCMOUTS 中。寄存器 MCMOUT 中存放着实际使用 CURH 和 EXPH 值。调制序列 MCMP 被送入 T12 输出调制控制模块。当前霍尔序列 (CURH) 和期望霍尔序列 (EXPH) 和霍尔传感器输入的采样 (可从寄存器 **CMPSTAT** 中得到) 进行比较。信号 HCRDY (霍尔比较准备就绪) 控制霍尔输入采样和比较输出评估, 下一节将对此进行详细说明。



**图 20-37 霍尔序列比较逻辑**

- 如果霍尔序列采样和 CURH 内编程设置的值匹配，检测到的跳变为毛刺（无霍尔事件），不需要进行后续操作。
- 如果霍尔序列采样和 EXPH 内的值匹配，则检测到的跳变为期望事件（正确霍尔事件 CM\_CHE）且 MCMP 值必须改变。
- 如果霍尔序列采样和 CURH 以及 EXPH 都不匹配，跳变是由严重错误引起的（错误霍尔事件 CM\_CWE）可导致紧急关闭（IDLE）。

每次产生正确的霍尔事件时（CM\_CHE），将映射寄存器 MCMOUTS 中的新霍尔序列传送到 MCMOUT 中（霍尔序列映射传送 HP\_ST），新的霍尔序列连同相应的输出

序列（如存储器内的预定义表）可由软件写入 MCMOUTS 中。信号 MCM\_ST 用于触发调制序列的映射传送。

将 MCMOUTS.STRHP 置 1（对于 EXPH 和 CURH 来讲）或将 MCMOUTS.STRMCMP 置 1（对于 MCMP 来讲），也可实现映射传送。

### 20.6.3 霍尔模式标志位

根据霍尔序列的比较操作置位相关的标志位，指示模块状态并触发其它操作和中断请求。

霍尔序列采样和期望序列（EXPH）匹配时，由信号 CM\_CHE 置位寄存器 **IS.CHE**（正确的霍尔事件）。也可通过软件置位寄存器 **ISS.SCHE** 对该标志位置位。若寄存器 **IEN.ENCHE** 被使能，CHE 的置位信号还可向 CPU 发送中断请求。位域 **INP.INPCHE** 定义发生中断请求时，哪个服务请求被激活。可通过软件设置寄存器 **ISR.RCHE** = 1，对标志 CHE 清零。

标志位 **IS.WHE** 指示产生了错误的霍尔事件。该标志位的置位、复位以及中断请求产生方式和标志位 CHE 的机制相似。

标志位 **STR** 的实现和 CHE、WHE 相同，该标志位由映射传送信号 MCM\_ST 硬件置位（见图 20-35）。

请注意：标志位 CHE、WHE 和 STR 的中断请求由相应的置位信号触发产生。这意味着，即使标志位已经被置位，仍可产生中断请求，无需复位标志位以使能后续的中断请求。

不过标志位 **IDLE** 的实现有所不同。若使能位 **ENIDLE** 被置位，可由信号 **CM\_WHE** 硬件置位 **IDLE**。也可通过软件置位 **SIDLE** 置位该标志位。只要 **IDLE** 被置位，调制序列位域 **MCMP** 将被清零、强制输出为被动态。必须通过置位 **RIDLE** 清零标志位 **IDLE**，返回正常操作。要从 **IDLE** 模式彻底重启，必须通过设定寄存器 MCMOUTS 中的位 **STRMCM** 和 **STRHP** 触发寄存器 MCMOUTS 到 MCMOUT 的映射传送请求。这种方式下，从 **IDLE** 模式返回正常模式的操作由软件控制，不过可以和 PWM 信号同步。

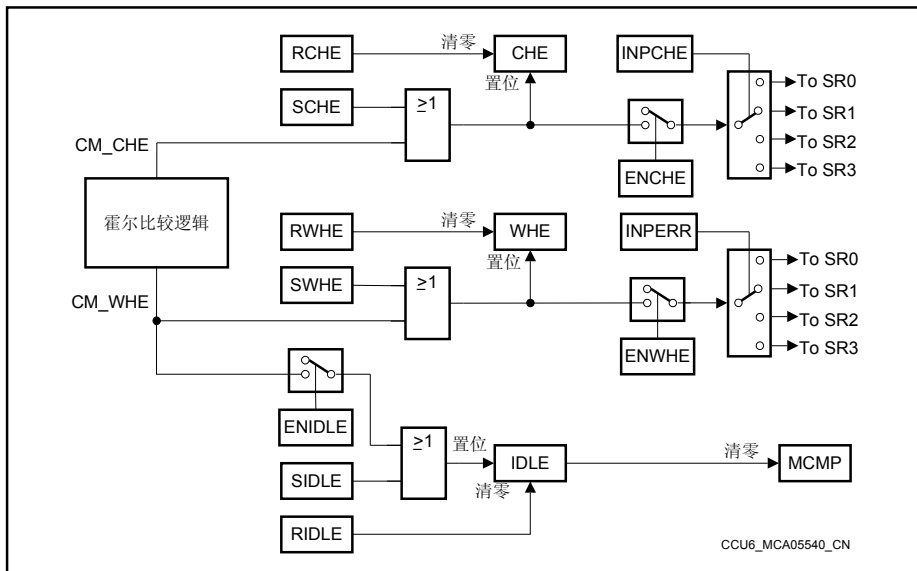


图 20-38 霍尔模式标志逻辑

#### 20.6.4 实现无刷直流电机控制的霍尔模式

CCU6 提供了一种便于无刷直流电机控制的块切换模式，通过定时器 T12 模块实现。将 T12 三路通道的位域 **T12MSEL.MSELx** 全部设定为 1000<sub>B</sub>，选定该模式。

该模式下，如图 20-39 所示，通道 CC60 工作在捕获模式，以测量最近两次正确霍尔事件之间的间隔；通道 CC61 工作在比较模式，为霍尔事件和新的 PWM 输出序列应用之间提供可编程的相位延迟；通道 CC62 也工作在比较模式，作为第一个超时标准。可由 T12 周期匹配事件构成第二个超时标准。

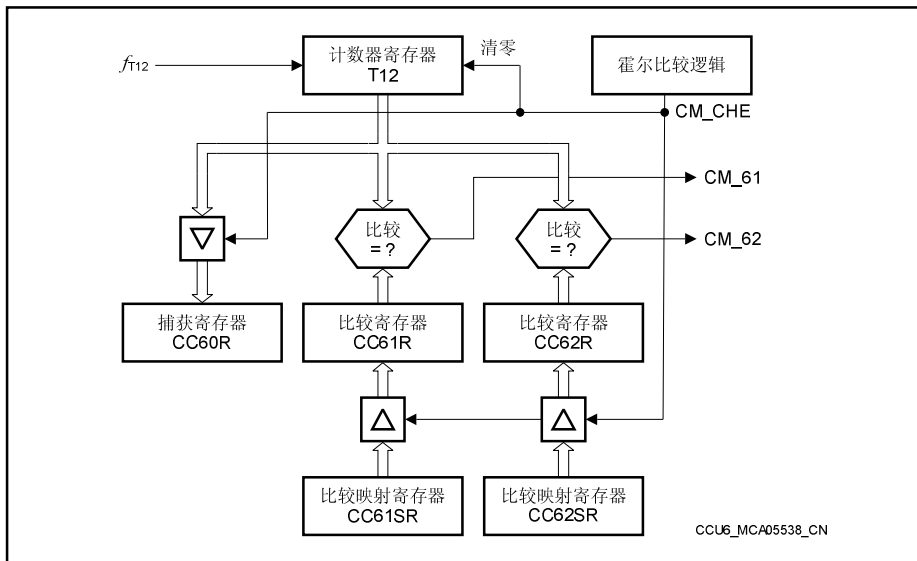
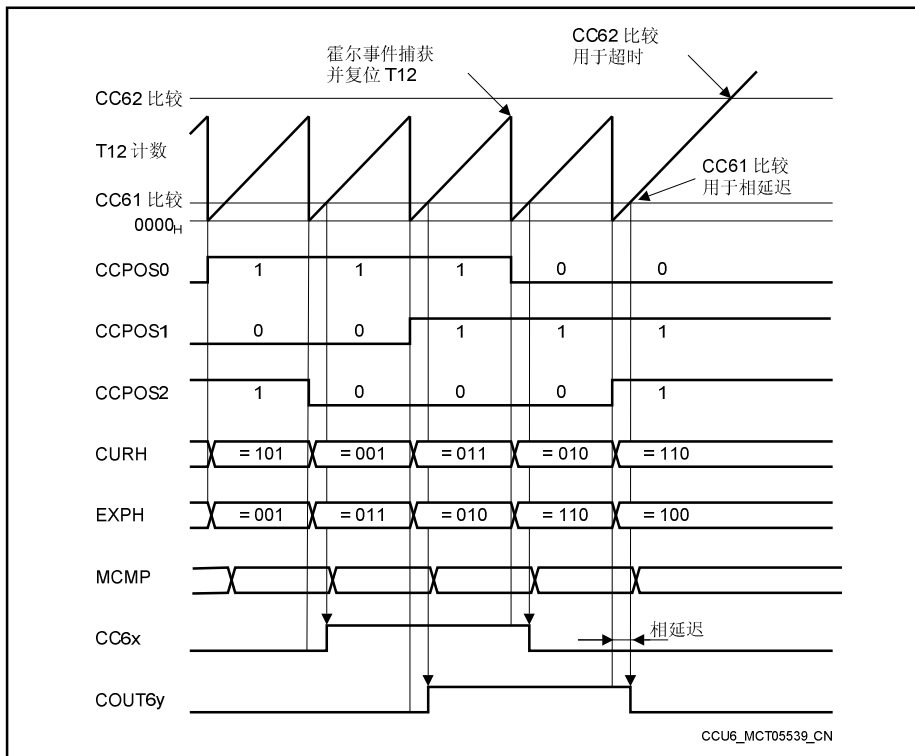


图 20-39 霍尔传感器模式下的 T12 模块

来自霍尔比较逻辑的信号 CM\_CHE 可用于：传送新比较值（从映射寄存器 CC6xSR 到实际比较寄存器 CC6xR），执行 T12 周期寄存器的映射传送以及将 T12 的当前计数值捕获到寄存器 CC60R 中，并清除定时器 12。

注：该模式下，不产生映射传送信号 T12\_ST。部分映射位，如 PSLy，将被传送到主寄存器（实际寄存器）中。只有当定时器 T12 停止工作时才可软件设定主寄存器。此时，映射寄存器和主寄存器均可由软件编程设定。





**图 20-40 无刷直流电机控制举例 (所有 MSEL6x = 1000<sub>B</sub>)**

检测到期望的霍尔序列之后 (CM\_CHE 有效), T12 的计数值被捕获到通道 CC60 中 (通过测量最近两次正确霍尔事件之间的间隔来代表电机实际转速) 并复位 T12。当定时器计数至通道 CC61 的比较值时, 触发位域 MCMP 的映射传送 (若位域 SWEN 使能映射传送), 因而切换到下个多通道状态。下个多通道状态与 PWM 调制源的同步可和该触发事件相结合 (以避免输出线上的毛刺, 见 [章节 20.5](#))。如果使用无传感器反电动势技术或使用霍尔传感器, 有必要使用通道 CC61 的比较功能产生从位置传感器的输入信号到输出切换之间的相位延迟。通道 CC62 的比较值可用作超时触发 (中断), 指示电机的实际转速远远低于期望值。可通过此特性可检测异常的负载变化并可禁止 PWM 的产生。

## 20.7 调制控制寄存器

### 20.7.1 调制控制

寄存器 MODCTR 控制由定时器 T12 和 T13 产生的 PWM 序列调制相应的输出信号。此外，多通道模式可被使能作为输出信号的附加调制源。

#### MODCTR

##### 调制控制寄存器

**XSFR (40<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>ECT 130</b>	<b>0</b>	<b>T13MODEN</b>						<b>MCM EN</b>	<b>0</b>	<b>T12MODEN</b>					
rw	r	rw						rw	r	rw					

符号	位序号	类型	功能描述
<b>T12MODEN</b>	[5:0]	rw	<b>定时器 T12 调制使能</b> 这些位使能由 T12 产生的 PWM 序列调制相应的比较输出信号。 T12MODEN0 = MODCTR.0 对应输出 CC60 T12MODEN1 = MODCTR.1 对应输出 COUT60 T12MODEN2 = MODCTR.2 对应输出 CC61 T12MODEN3 = MODCTR.3 对应输出 COUT61 T12MODEN4 = MODCTR.4 对应输出 CC62 T12MODEN5 = MODCTR.5 对应输出 COUT62 0 <sub>B</sub> 禁止 T12 的 PWM 序列调制相应输出信号 1 <sub>B</sub> 使能 T12 的 PWM 序列调制相应输出
<b>MCMEN</b>	7	rw	<b>多通道模式使能</b> 0 <sub>B</sub> 禁止根据位域 MCMOUT 的设置，由多通道序列调制相应输出信号 1 <sub>B</sub> 使能根据位域 MCMOUT 的设置，由多通道序列调制相应输出
<b>T13MODEN</b>	[13:8]	rw	<b>定时器 T13 调制使能</b> 这些位使能由 T13 产生的 PWM 序列 CC63_O 调制相应的输出信号。

符号	位序号	类型	功能描述
			T13MODEN0 = MODCTR.8 对应输出 CC60 T13MODEN1 = MODCTR.9 对应输出 COUT60 T13MODEN2 = MODCTR.10 对应输出 CC61 T13MODEN3 = MODCTR.11 对应输出 COUT61 T13MODEN4 = MODCTR.12 对应输出 CC62 T13MODEN5 = MODCTR.13 对应输出 COUT62 0 <sub>B</sub> 禁止 T13 的 PWM 序列调制相应输出信号 1 <sub>B</sub> 使能 T13 的 PWM 序列调制相应输出
<b>ECT130</b>	15	rw	<b>定时器 T13 比较输出使能</b> 0 <sub>B</sub> 信号 COUT63 处于被动态 1 <sub>B</sub> 信号 COUT63 被使能, 输出 T13 产生的 PWM 信号。
<b>0</b>	6, 14	r	<b>保留;</b> 读操作返回 0; 应写入 0。

## 20.7.2 强制中断控制寄存器

寄存器 TRPCTR 控制强制中断功能。TRPCTR 中存放着各路输出信号的强制中断使能位、以及发生强制中断时的操作选择控制位。出现强制中断时， $\overline{\text{CTRAP}}$  输入引脚为低电平，由位 IS.TRPF 监控（反向电平）。当  $\text{TRPF} = 1$ （强制中断输入有效），强制中断状态位 IS.TRPS 被设置为 1。

### TRPCTR

#### 强制中断控制寄存器

**XSFR (42<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>TRP PEN</b>	<b>TRP EN 13</b>	<b>TRPEN</b>						<b>0</b>				<b>TRP M2</b>	<b>TRP M1</b>	<b>TRP M0</b>	
rw	rw	rw						r				rw	rw	rw	

符号	位序号	类型	功能描述
<b>TRPM1, TRPM0</b>	1, 0	rw	<b>强制中断模式控制位 1, 0</b> 强制中断再次变为无效之后，由这两位控制相应输出如何退出强制中断状态。 退出强制中断时，与定时器驱动的 PWM 序列同步以避免产生多余脉冲。 <b>[TRPM1, TRPM0]的组合导致：</b> 00 <sub>B</sub> 与 T12 同步：检测到 T12 0-匹配（递增计数）时，TRPF 再次变为 0 之后，退出强制中断状态（返回到正常操作）。 01 <sub>B</sub> 和 T13 同步：检测到 T13 0-匹配，TRPF 再次变为 0 之后，退出强制中断状态（返回到正常操作） 10 <sub>B</sub> 保留 11 <sub>B</sub> 不和 T12 或 T13 同步：强制中断标志 TRPF 再次变为 0 之后，立即退出强制中断状态（返回到正常操作）。
<b>TRPM2</b>	2	rw	<b>强制中断模式控制位 2</b> TRPM2 定义强制中断输入条件（ $\overline{\text{CTRAP}} = 0$ 且 $\text{TRPPEN} = 1$ ）不再有效（或者 $\overline{\text{CTRAP}} = 1$ ，或者

符号	位序号	类型	功能描述
			<p>TRPPEN = 0) 之后, 如何清除强制中断标志。</p> <p><b>0<sub>B</sub> 自动模式:</b> 如果强制中断输入条件不再有效, 由硬件自动清除位 TRPF。</p> <p><b>1<sub>B</sub> 手动模式:</b> 强制中断输入条件不再有效之后, 位 TRPF 保持为 0。通过编程设置 ISR.RTRPF = 1, 软件清除 TRPF。</p>
<b>TRPEN</b>	[13:8]	rw	<p><b>强制中断使能控制</b></p> <p>置位各控制位可使能下述对应输出信号的强制中断功能:</p> <p>TRPEN0 = TRPCTR.8 对应输出 CC60            TRPEN1 = TRPCTR.9 对应输出 COUT60            TRPEN2 = TRPCTR.10 对应输出 CC61            TRPEN3 = TRPCTR.11 对应输出 COUT61            TRPEN4 = TRPCTR.12 对应输出 CC62            TRPEN5 = TRPCTR.13 对应输出 COUT62</p> <p><b>0<sub>B</sub></b> 相应输出信号的强制中断功能被禁止。输出状态与位 IS.TRPS 无关。</p> <p><b>1<sub>B</sub></b> 相应输出信号的强制中断功能被使能。当 IS.TRPS = 1 时, 输出被设置为被动态。</p>
<b>TRPEN13</b>	14	rw	<p><b>T13 输出的强制中断使能控制</b></p> <p><b>0<sub>B</sub></b> 输出 COUT63 的强制中断功能被禁止。输出状态与位 IS.TRPS 无关</p> <p><b>1<sub>B</sub></b> 输出 COUT63 的强制中断功能被使能。位 IS.TRPS = 1 时, 输出状态被设置为被动态</p>
<b>TRPPEN</b>	15	rw	<p><b>强制中断输入引脚使能控制</b></p> <p>该位使能强制中断产生的输入 (引脚) 功能。当 TRPPEN = 1 时, 如果在引脚 <math>\overline{\text{CTRAP}}</math> 上检测到下降沿, 才能产生中断。</p> <p><b>0<sub>B</sub></b> 根据输入引脚 <math>\overline{\text{CTRAP}}</math> 产生 CCU6 强制中断的功能被禁止。CCU6 强制中断只能通过软件置</p>

**捕获/比较单元 6 (CCU6)**

符号	位序号	类型	功能描述
			位 TRPF 产生。  1 <sub>B</sub> 根据输入引脚 $\overline{\text{CTRAP}}$ 产生 CCU6 强制中断的功能被使能。CCU6 强制中断既可通过软件置位 TRPF 产生，也可由 $\overline{\text{CTRAP}} = 0$ 产生。
<b>0</b>	[7:3]	r	<b>保留；</b> 读操作返回 0；应写入 0。

### 20.7.3 被动态电平寄存器

寄存器 PSLR 定义了模块 PWM 输出的被动态电平。输出端口处于被动态时，引脚驱动的电平为被动态电平。输出端口处于主动态时，引脚驱动的电平为主动态电平，该电平和被动态电平反相。被动态电平可使输出电平和所连接的功率器件的驱动极性（反相或不反相）相匹配。该寄存器内的位具有映射位域，允许同时更新所有 PWM 相关参数（用 T12\_ST 更新位域 PSL，而用 T13\_ST 更新 PSL63）。实际使用的值可被读取（属性“rh”），而映射位为只写（属性“w”）。

#### PSLR

##### 被动态电平寄存器

XSFR (44<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								PSL 63	0	PSL					
r								rwh	r	rwh					

符号	位序号	类型	功能描述
PSL	[5:0]	rwh	<b>比较输出的被动态电平</b> 这些位决定模块输出为被动态时所驱动的被动态电平。 PSL0 = PSLR.0 对应输出 CC60 PSL1 = PSLR.1 对应输出 COUT60 PSL2 = PSLR.2 对应输出 CC61 PSL3 = PSLR.3 对应输出 COUT61 PSL4 = PSLR.4 对应输出 CC62 PSL5 = PSLR.5 对应输出 COUT62 0 <sub>B</sub> 被动态电平为 0 1 <sub>B</sub> 被动态电平为 1
PSL63	7	rwh	<b>输出 COUT63 的被动态电平</b> 该位决定输出引脚 COUT63 的被动态电平。 0 <sub>B</sub> 被动态电平为 0 1 <sub>B</sub> 被动态电平为 1
0	6, [15:8]	r	<b>保留；</b> 读操作返回 0；应写入 0。

## 20.7.4 多通道模式寄存器

寄存器 MCMCTR 用于控制多通道功能。

### MCMCTR

多通道模式控制寄存器

**XSFR (4E<sub>H</sub>)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					STE 13U	STE 12D	STE 12U	0		SWSYN	0		SWSEL		
r					rw	rw	rw	r		rw	r		rw		

符号	位序号	类型	功能描述
<b>SWSEL</b>	[2:0]	rw	<p><b>切换选择</b></p> <p>位域 SWSEL 选择从 MCMPS 到 MCMP 映射传送 MCM_ST 的触发请求源（下一次多通道事件）。触发请求保存在提示标志 R 中，直到完成映射传送。映射传送完成后标志 R 被自动清除。映射传送和位域 SWSYN 选定的事件同步发生。</p> <p>000<sub>B</sub> 不产生触发请求</p> <p>001<sub>B</sub> 检测到正确霍尔事件 (CM_CHE)</p> <p>010<sub>B</sub> 检测到 T13 周期匹配 (递增计数时)</p> <p>011<sub>B</sub> T12 1-匹配 (递减计数时)</p> <p>100<sub>B</sub> 检测到 T12 通道 1 比较匹配 (相位延迟功能)</p> <p>101<sub>B</sub> 检测到 T12 周期匹配 (递增计数时)</p> <p>110<sub>B</sub> 保留，不产生触发请求</p> <p>111<sub>B</sub> 保留，不产生触发请求</p>
<b>SWSYN</b>	[5:4]	rw	<p><b>切换同步</b></p> <p>如果已经请求映射传送 (标志 R 被 SWSEL 所选定的事件置位) 且如果 MCMEN = 1，由位域 SWSYN 定义映射传送事件 MCM_ST 的同步机制。该特性允许输出和用于调制的 (T12 或 T13) PWM 源同步。</p> <p>00<sub>B</sub> 直接；触发事件立即引发映射传送</p> <p>01<sub>B</sub> T13 0-匹配触发映射传送</p>



符号	位序号	类型	功能描述
			<b>10<sub>B</sub></b> T12 0-匹配（递增计数时）触发映射传送 <b>11<sub>B</sub></b> 保留；无操作。
<b>STE12U</b>	8	rw	<b>T12 递增计数映射传送使能</b> 如果标志 MCMOUT.R 被置位，或者 T12 递增计数时，检测到周期匹配而使得标志 R 被置位，则由该位使能映射传送 T12_ST。 <b>0<sub>B</sub></b> 无操作 <b>1<sub>B</sub></b> 如果 MCMEN = 1，T12_ST 映射传送机制被使能。
<b>STE12D</b>	9	rw	<b>T12 递减计数映射传送使能</b> 如果标志 MCMOUT.R 被置位，或者 T12 递减计数时，检测到 1-匹配而使得标志 R 被置位，则由该位使能映射传送 T12_ST。 <b>0<sub>B</sub></b> 无操作 <b>1<sub>B</sub></b> 如果 MCMEN = 1，T12_ST 映射传送机制被使能。
<b>STE13U</b>	8	rw	<b>T13 递增计数映射传送使能</b> 如果标志 MCMOUT.R 被置位，或者检测到 T13 周期匹配而使得标志 R 被置位，则由该位使能映射传送 T13_ST。 <b>0<sub>B</sub></b> 无操作 <b>1<sub>B</sub></b> 如果 MCMEN = 1，T13_ST 映射传送机制被使能。
<b>0</b>	3, [7:6], [15:11]	r	<b>保留；</b> 读操作返回 0；应写入 0。

寄存器 MCMOUTS 保存用于多通道模式和霍尔模式的序列输入。该寄存器（可读且可写）为 MCMOUT 的映射寄存器，指示当前有效的信号。

## MCMOUTS

多通道模式输出映射寄存器

XSFR (4A<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STR HP	0	CURHS	EXPHS	STR MCM	0	MCMPS									
w	r	rw	rw	w	r	rw									

符号	位序号	类型	功能描述
<b>MCMP5</b>	[5:0]	rw	<b>多通道 PWM 序列映射位域</b> MCMP5 是位域 MCMP 的映射位域。根据寄存器 MCMCTR 定义的传送条件由 MCM_ST 触发多通道映射传送。
<b>STRMCM</b>	7	w	<b>MCMP5 的映射传送请求</b> 置位 STRMCM = 1 立即激活 MCM_ST，用位域 MCMP5 的值更新位域 MCMP。 读取该位时始终返回 0。 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位域 MCMP 被更新
<b>EXPHS</b>	[10:8]	rw	<b>期望霍尔序列映射位域</b> EXPHS 是位域 EXPH 的映射位域。检测到正确的霍尔事件 (CM_CHE) 时，该位域的内容传送到 EXPH 中。
<b>CURHS</b>	[13:11]	rw	<b>当前霍尔序列映射位域</b> CURHS 是位域 CURH 的映射位域。检测到正确的霍尔事件 (CM_CHE) 时，该位域的内容传送到 CURH 中。
<b>STRHP</b>	15	w	<b>霍尔序列映射传送请求</b> 置位 STRHP = 1 立即激活 HP_ST，用位域 CURHS 和 EXPHS 中的值更新位域 CURH 和 EXPH。 读取该位时始终返回 0。 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位域 EXPH 和 CURH 被更新。

符号	位序号	类型	功能描述
<b>0</b>	6, 14	r	保留; 读操作返回 0; 应写入 0。

## MCMOUT

### 多通道模式输出寄存器

**XSFR (4C<sub>H</sub>)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>CURH</b>				<b>EXPH</b>	<b>0</b>	<b>R</b>	<b>MCMP</b>							
r	rh				rh	r	rh	rh							

符号	位序号	类型	功能描述
<b>MCMP</b>	[5:0]	rh	<p><b>多通道 PWM 序列</b></p> <p>MCMP 中存放着多通道模式的输出调制序列。如果由 MODCTR.MCMEN = 1 使能该模式, 所有 T12 相关的 PWM 输出状态可被修改。</p> <p>当 IS.IDLE = 1 时, 该位域为 0。</p> <p>MCMP0 = MCMOUT.0 对应输出 CC60</p> <p>MCMP1 = MCMOUT.1 对应输出 COUT60</p> <p>MCMP2 = MCMOUT.2 对应输出 CC61</p> <p>MCMP3 = MCMOUT.3 对应输出 COUT61</p> <p>MCMP4 = MCMOUT.4 对应输出 CC62</p> <p>MCMP5 = MCMOUT.5 对应输出 COUT62</p> <p>0<sub>B</sub> 输出设定为被动态。与 T12 或 T13 产生的 PWM 无关。</p> <p>1<sub>B</sub> 输出可处于主动态, 由被使能的 T12、T13 产生的 PWM 调制信号和强制中断状态决定输出。</p>
<b>R</b>	6	rh	<p><b>提示标志</b></p> <p>该标志表明选定的触发源已请求位域 MCMPS 到 MCMP 的映射传送。当 MCMEN = 0 或发生映射传送之后, 该位被清除。</p> <p>0<sub>B</sub> 未请求映射传送 MCM_ST。</p>

**捕获/比较单元 6 (CCU6)**

符号	位序号	类型	功能描述
			1 <sub>B</sub> 已请求映射传送 MCM_ST, 但是因为选定的同步条件还未出现, 所以映射传送还未执行。
<b>EXPH</b>	[10:8]	rh	<p><b>期望的霍尔序列</b></p> <p>由映射传送 HP_ST 用位域 EXPHS 的值更新位域 EXPH。</p> <p>如果 HCRDY = 1, 为了检测到出现了期望霍尔序列或错误序列, 位域 EXPH 和 CCPOSx 输入采样进行比较。</p> <p>如果霍尔输入引脚上的霍尔序列采样等于位域 EXPH, 则已经检测到正确的霍尔事件 (CM_CHE)。</p>
<b>CURH</b>	[13:11]	rh	<p><b>当前的霍尔序列</b></p> <p>由映射传送 HP_ST 用位 CURHS 的值更新位域 CURH。</p> <p>如果 HCRDY = 1, 为了检测毛刺, CURH 和 CCPOSx 输入采样进行比较。</p> <p>如果霍尔输入引脚上的霍尔序列采样等于位域 CURH, 则未检测到任何霍尔事件。</p> <p>如果霍尔序列采样和 CURH 或 EXPH 都不相等, 则发生的事件并不是期望的霍尔事件, 该事件可能由致命的错误 (如转子停转等) 引起。在此情况下, 检测到错误霍尔事件 (CM_WHE)。</p>
<b>0</b>	7, [15:14]	r	<p><b>保留;</b></p> <p>读操作返回 0; 应写入 0。</p>

## 20.8 中断处理

本节描述 CCU6 模块的中断处理。

### 20.8.1 中断结构

硬件中断事件或软件置位（寄存器 ISS 中的）相应中断置位控制位皆可置位事件指示标志（寄存器 IS 中的）并触发中断产生。中断脉冲的产生和寄存器 IS 中的中断标志无关（为了产生另一个中断，无需清除相应的状态位）。置位寄存器 ISR 中的相关位可软件清除中断标志。

若寄存器 IEN 中的某中断使能位被置位，可在某中断输出线上（CCU6 单元共有四条中断线 SR0 至 SR3）产生中断脉冲。若多个中断源和同一个中断节点指针（位于寄存器 INP 中）相连，各中断请求进行逻辑或，合并为一条公共的中断请求输出线（见[图 20-41](#)）。

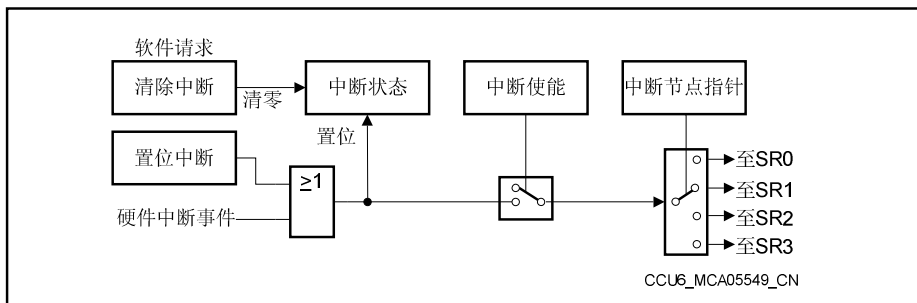


图 20-41 一般中断结构

CCU6 模块的可用中断事件如[图 20-42](#)所示。

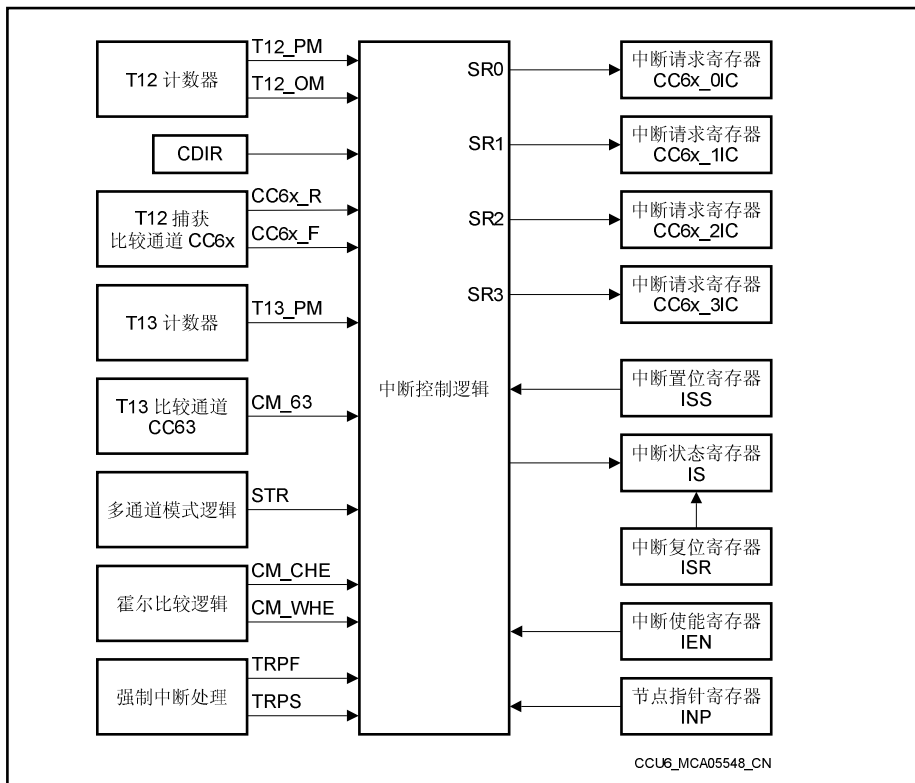


图 20-42 中断源和中断事件

## 20.8.2 中断寄存器

### 20.8.2.1 中断状态寄存器

寄存器 IS 保存各中断请求位。IS 为只读寄存器，写操作对该寄存器的内容无任何影响。通过寄存器 ISS（置位对应位）或寄存器 ISR（清除对应位）可软件置位或清除寄存器 IS 中的各位。

中断的产生和寄存器 IS 中的值无关，例如，即使对应位已经被置位，还将产生中断（如被使能）。中断产生的触发条件为检测到寄存器 IS 中的对应位被置位（由硬件或软件）。

比较模式（及霍尔模式）下，只有当定时器运行（ $T1xR = 1$ ）时，才能产生与定时器相关的中断。捕获模式下，当定时器 T12 停止时，还能产生捕获中断。

*注：并非寄存器 IS 中的所有位均可产生中断。该寄存器还包括其它一些状态位。这些位和中断请求位的置位和清除操作相似。建议用户软件按位检查中断位（而不是将这些中断位相或得到一个公共位）*

#### IS

##### 中断状态寄存器

XSFR (50<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
STR	IDLE	WHE	CHE	TRP S	TRP F	T13 PM	T13 CM	T12 PM	T12 OM	ICC 62F	ICC 62R	ICC 61F	ICC 61R	ICC 60F	ICC 60R
rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

符号	位序号	类型	功能描述
ICC60R, ICC61R, ICC62R	0, 2, 4	rh	<p><b>捕获，比较匹配上升沿标志位</b></p> <p>该位指示已经检测到 CC6x_R 事件。比较模式下，T12 递增计数（CM_6x 和 CDIR = 0）时检测到比较匹配；捕获模式下，在相关输入 CC6xIN 上检测到上升沿跳变，则通过该位指示已经检测到 CC6x_R 事件。</p> <p>0<sub>B</sub> 还未检测到上述事件 1<sub>B</sub> 已检测到上述事件</p>
ICC60F ICC61F ICC62F	1, 3, 5	rh	<p><b>捕获，比较匹配下降沿标志位</b></p> <p>该位指示已经检测到 CC6x_F 事件。</p> <p>比较模式下，T12 递减计数（CM_6x 且 CDIR = 1）时检测到比较匹配；捕获模式下，检测到输入</p>

**捕获/比较单元 6 (CCU6)**

符号	位序号	类型	功能描述
			<p>CC6xIN 发生下降沿跳变。则通过该位指示已经检测到 CC6x_F 事件。</p> <p>0<sub>B</sub> 还未检测到上述事件</p> <p>1<sub>B</sub> 已检测到上述事件</p>
<b>T12OM</b>	6	rh	<p><b>定时器 T12 1- 匹配标志位</b></p> <p>该位指示递减计数时 (T12_OM 且 CDIR = 1) , 检测到定时器 T12 1-匹配事件。</p> <p>0<sub>B</sub> 还未检测到上述事件</p> <p>1<sub>B</sub> 已经检测到上述事件</p>
<b>T12PM</b>	7	rh	<p><b>定时器 T12 周期匹配标志位</b></p> <p>该位指示递增计数时 (T12_PM 且 CDIR = 0) , 检测到 T12 周期匹配事件。</p> <p>0<sub>B</sub> 还未检测到上述事件</p> <p>1<sub>B</sub> 已经检测到上述事件</p>
<b>T13CM</b>	8	rh	<p><b>定时器 T13 比较匹配标志位</b></p> <p>该位指示已经检测到定时器 T13 比较匹配 (CM_63) 。</p> <p>0<sub>B</sub> 还未检测到上述事件</p> <p>1<sub>B</sub> 已经检测到上述事件</p>
<b>T13PM</b>	9	rh	<p><b>定时器 T13 周期匹配标志位</b></p> <p>该位指示已经检测到定时器 T13 周期匹配 (T13_PM) 。</p> <p>0<sub>B</sub> 还未检测到上述事件</p> <p>1<sub>B</sub> 已经检测到上述事件</p>
<b>TRPF</b>	10	rh	<p><b>强制中断标志位</b></p> <p>该位指示检测/已经检测到强制中断情况 (输入 <math>\overline{\text{CTRAP}} = 0</math> 或通过软件设定)。若 TRPM2 = 0, 输入 <math>\overline{\text{CTRAP}} = 1</math> 或者 TRPPEN = 0 时, 硬件自动清除 TRPF; 若 TRPM2 = 1, 必须由软件设置 RTRPF = 1 以清除该标志。</p> <p>0<sub>B</sub> 还未检测到强制中断情况</p>



**捕获/比较单元 6 (CCU6)**

符号	位序号	类型	功能描述
			1 <sub>B</sub> 检测到/已经检测到强制中断情况
<b>TRPS</b>	11	rh	<b>强制中断状态位<sup>1)</sup></b> 该位指示实际强制中断状态。如果 TRPF = 1 则该位置位；根据寄存器 TRPCTR 中选定的模式清除该位。  0 <sub>B</sub> 强制中断状态无效。 1 <sub>B</sub> 强制中断状态有效。
<b>CHE</b>	12	rh	<b>正确霍尔事件标志位</b> 该位指示已经检测到正确霍尔事件 (CM_CHE)  0 <sub>B</sub> 还未检测到上述事件 1 <sub>B</sub> 已检测到上述事件
<b>WHE</b>	13	rh	<b>错误霍尔事件标志位</b> 该位指示已经检测到错误霍尔事件 (CM_WHE)  0 <sub>B</sub> 还未检测到上述事件 1 <sub>B</sub> 已检测到上述事件
<b>IDLE</b>	14	rh	<b>空闲状态标志位</b> 若空闲状态被使能 (ENIDLE = 1)，IDLE 位和位 WHE 同时置位；该位必须由软件清除。  0 <sub>B</sub> 无操作 1 <sub>B</sub> 位域 MCMP 被清零，选定的输出被设置为被动态
<b>STR</b>	15	rh	<b>多通道模式映射传送请求</b> 该位指示已经发生了从 MCMPS 到 MCMP 的映射传送 (MCM_ST)。  0 <sub>B</sub> 还未检测到上述事件 1 <sub>B</sub> 已经检测到上述事件

- 1) 强制中断状态期间，选定的输出被设置为被动态。由寄存器 PSLR 内的相关位确定被动态期间驱动的逻辑电平。如果强制中断条件不再有效，但是所选择的同步事件还未发生，则会出现位 TRPS = 1 和 TRPF = 0 的情况。

### 20.8.2.2 中断状态置位寄存器

通过寄存器 ISS 中的各中断请求置位控制位可软件产生 CCU6 中断请求。向该寄存器中的位写 1，则置位寄存器 IS 中的对应位，可用于产生中断事件（若该中断被使能并且相应功能可用）。

读取所有位则返回 0。

#### ISS

##### 中断状态置位寄存器

(52<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
S STR	S IDLE	S WHE	S CHE	S WHC	S TRP F	S T13 PM	S T13 CM	S T12 PM	S T12 OM	S CC 62F	S CC 62R	S CC 61F	S CC 61R	S CC 60F	S CC 60R
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

符号	位序号	类型	功能描述
SCC60R, SCC61R, SCC62R	0, 2, 4	W	置位捕获，比较匹配上升沿标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 CC6xR 将被置位
SCC60F, SCC61F, SCC62F	1, 3, 5	W	置位捕获，比较匹配下降沿标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 CC6xF 将被置位
ST12OM	6	W	置位定时器 T12 1- 匹配标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T12OM 将被置位
ST12PM	7	W	置位定时器 T12 周期匹配标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T12PM 将被置位
ST13CM	8	W	置位定时器 T13 比较匹配标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T13CM 将被置位
ST13PM	9	W	置位定时器 T13 周期匹配标志位 0 <sub>B</sub> 无操作

符号	位序号	类型	功能描述
			1 <sub>B</sub> 位 T13PM 将被置位
<b>STRPF</b>	10	w	置位强制中断标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 TRPF 和 TRPS 将被置位
<b>SWHC</b>	11	w	软件霍尔比较 0 <sub>B</sub> 无操作 1 <sub>B</sub> 触发霍尔比较操作
<b>SCHE</b>	12	w	置位正确霍尔事件标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 CHE 将被置位
<b>SWHE</b>	13	w	置位错误霍尔事件标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 WHE 将被置位
<b>SIDLE</b>	14	w	置位 IDLE 标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 IDLE 将被置位
<b>SSTR</b>	15	w	置位 STR 标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 STR 将被置位

### 20.8.2.3 状态复位寄存器

通过寄存器 **ISR** 中的各位可软件清除中断事件标志。向该寄存器中的位写 1，则寄存器 **IS** 中的对应位清零。

读取所有位则返回 0。

#### ISR

**中断状态复位寄存器**

**XSFR (54<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R STR	R IDLE	R WHE	R CHE	0	R TRP F	R T13 PM	R T13 CM	R T12 PM	R T12 OM	R CC 62F	R CC 62R	R CC 61F	R CC 61R	R CC 60F	R CC 60R
w	w	w	w	r	w	w	w	w	w	w	w	w	w	w	w

符号	位序号	类型	功能描述
<b>RCC60R, RCC61R, RCC62R</b>	0, 2, 4	w	<b>复位捕获，比较匹配上升沿标志位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 CC6xR 将被清除
<b>RCC60F, RCC61F, RCC62F</b>	1, 3, 5	w	<b>复位捕获，比较匹配下降沿标志位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 CC6xF 将被清除
<b>RT12OM</b>	6	w	<b>复位定时器 T12 1- 匹配标志位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T12OM 将被清除
<b>RT12PM</b>	7	w	<b>复位定时器 T12 周期匹配标志位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T12PM 将被清除
<b>RT13CM</b>	8	w	<b>复位定时器 T13 比较匹配标志位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T13CM 将被清除
<b>RT13PM</b>	9	w	<b>复位定时器 T13 周期匹配标志位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T13PM 将被清除

符号	位序号	类型	功能描述
<b>RTRPF</b>	10	w	<b>复位强制中断标志位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 TRPF 将被清除（当输入 $\overline{\text{CTRAP}} = 0$ 且 TRPPEN = 1 时，则不考虑）
<b>RCHE</b>	12	w	<b>复位正确霍尔事件标志位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 CHE 将被清除
<b>RWHE</b>	13	w	<b>复位错误霍尔事件标志位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 WHE 将被清除
<b>RIDLE</b>	14	w	<b>复位 IDLE 标志位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 IDLE 将被清除
<b>RSTR</b>	15	w	<b>复位 STR 标志位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 STR 将被清除
<b>0</b>	11	r	<b>保留；</b> 读操作返回 0；应写入 0。

#### 20.8.2.4 中断使能寄存器

寄存器 IEN 保存中断使能位和一个控制位，错误霍尔序列的情况下，该控制位用于使能自动空闲功能。

#### IEN

##### 中断使能寄存器

**XSFR (58<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>EN STR</b>	<b>EN IDLE</b>	<b>EN WHE</b>	<b>EN CHE</b>	<b>0</b>	<b>EN TRP F</b>	<b>EN T13 PM</b>	<b>EN T13 CM</b>	<b>EN T12 PM</b>	<b>EN T12 OM</b>	<b>EN CC 62F</b>	<b>EN CC 62R</b>	<b>EN CC 61F</b>	<b>EN CC 61R</b>	<b>EN CC 60F</b>	<b>EN CC 60R</b>
rw	rw	rw	rw	r	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

**捕获/比较单元 6 (CCU6)**

符号	位序号	类型	功能描述
<b>ENCC60R, ENCC61R, ENCC62R</b>	0, 2, 4	rw	<b>通道 CC6x 的捕获, 比较匹配上升沿中断使能</b> 0 <sub>B</sub> 如果出现寄存器 IS 中的位 CC6xR 置位情况, 不产生中断。 1 <sub>B</sub> 如果出现寄存器 IS 中的位 CC6xR 置位情况, 则产生中断请求。位域 INPCC6x 所选择的服务请求输出被激活。
<b>ENCC60F, ENCC61F, ENCC62F</b>	1, 3, 5	rw	<b>通道 CC6x 的捕获, 比较匹配下降沿中断使能</b> 0 <sub>B</sub> 如果出现寄存器 IS 中的位 CC6xF 置位情况, 不产生中断。 1 <sub>B</sub> 如果出现寄存器 IS 中的位 CC6xF 置位情况, 则产生中断请求。位域 INPCC6x 所选择的服务请求输出被激活。
<b>ENT12OM</b>	6	rw	<b>定时器 T12 1- 匹配使能</b> 0 <sub>B</sub> 如果出现寄存器 IS 中的位 T12OM 置位情况, 不产生中断。 1 <sub>B</sub> 如果出现寄存器 IS 中的位 T12OM 置位情况, 则产生中断请求。位域 INPT12 所选择的服务请求输出被激活。
<b>ENT12PM</b>	7	rw	<b>定时器 T12 周期匹配中断使能</b> 0 <sub>B</sub> 如果出现寄存器 IS 中的位 T12PM 置位情况, 不产生中断。 1 <sub>B</sub> 如果出现寄存器 IS 中的位 T12PM 置位情况, 则产生中断请求。位域 INPT12 所选择的服务请求输出被激活。
<b>ENT13CM</b>	8	rw	<b>定时器 T13 比较匹配中断使能</b> 0 <sub>B</sub> 如果出现寄存器 IS 中的位 T13CM 置位情况, 不产生中断。 1 <sub>B</sub> 如果出现寄存器 IS 中的位 T13CM 置位情况, 则产生中断请求。位域 INPT13 所选择的服务请求输出被激活。
<b>ENT13PM</b>	9	rw	<b>定时器 T13 周期匹配中断使能</b> 0 <sub>B</sub> 如果出现寄存器 IS 中的位 T13PM 置位情况, 不产生中断。

符号	位序号	类型	功能描述
			<b>1<sub>B</sub></b> 如果出现寄存器 IS 中的位 T13PM 置位情况，则产生中断请求。位域 INPT13 所选择的服务请求输出被激活。
<b>ENTRPF</b>	10	rw	<b>强制中断标志中断使能</b>  <b>0<sub>B</sub></b> 如果出现寄存器 IS 中的位 TRPF 置位情况，不产生中断。  <b>1<sub>B</sub></b> 如果出现寄存器 IS 中的位 TRPF 置位情况，则产生中断请求。位域 INPERR 所选择的服务请求输出被激活。
<b>ENCHE</b>	12	rw	<b>正确霍尔事件中断使能</b>  <b>0<sub>B</sub></b> 如果出现寄存器 IS 中的位 CHE 置位情况，不产生中断。  <b>1<sub>B</sub></b> 如果出现寄存器 IS 中的位 CHE 置位情况，则产生中断请求。位域 INPCHE 所选择的服务请求输出被激活。
<b>ENWHE</b>	13	rw	<b>错误霍尔事件中断使能</b>  <b>0<sub>B</sub></b> 如果出现寄存器 IS 中的位 WHE 置位情况，不产生中断。  <b>1<sub>B</sub></b> 如果出现寄存器 IS 中的位 WHE 置位情况，则产生中断请求。位域 INPERR 所选择的服务请求输出被激活。
<b>ENIDLE</b>	14	rw	<b>使能 IDLE</b> 由该位使能检测到错误霍尔事件之后（位 WHE 被置位），自动进入到空闲状态（位 IDLE 将被置位）。空闲状态期间，位域 MCMP 被自动清除。 <b>0<sub>B</sub></b> 检测到错误霍尔事件时，位 IDLE 不被自动置位。 <b>1<sub>B</sub></b> 检测到错误霍尔事件时，位 IDLE 被自动置位。
<b>ENSTR</b>	15	rw	<b>多通道模式映射传送中断使能</b>  <b>0<sub>B</sub></b> 如果出现寄存器 IS 中的位 STR 置位情况，不产生中断。  <b>1<sub>B</sub></b> 如果出现寄存器 IS 中的位 STR 置位情况，

符号	位序号	类型	功能描述
			则产生中断请求。位域 INPCHE 所选择的服务请求输出被激活。
<b>0</b>	11	r	<b>保留；</b> 读操作返回 0；应写入 0。

### 20.8.2.5 中断节点指针寄存器

寄存器 INP 保存中断节点指针，允许用户进行灵活中断处理。如果发生相应中断事件且使能该事件产生中断，由这些位域定义哪个服务请求输出将被激活。

#### INP

#### 中断节点指针寄存器

**XSFR (56<sub>H</sub>)**

**复位值: 3940<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>INP T13</b>	<b>INP T12</b>	<b>INP ERR</b>	<b>INP CHE</b>	<b>INP CC62</b>	<b>INP CC61</b>	<b>INP CC60</b>								
r	rw	rw	rw	rw	rw	rw	rw								

符号	位序号	类型	功能描述
<b>INPCC60, INPCC61, INPCC62</b>	[1:0], [3:2], [5:4]	rw	<b>通道 CC6x 中断的中断节点指针</b> 该位域定义了由中断请求源 CC6xR（如果由位 ENCC6xR 使能）或 CC6xF（如果由位 ENCC6xF）激活的服务请求输出线。  00 <sub>B</sub> 选择服务请求输出 SR0 01 <sub>B</sub> 选择服务请求输出 SR1 10 <sub>B</sub> 选择服务请求输出 SR2 11 <sub>B</sub> 选择服务请求输出 SR3
<b>INPCHE</b>	[7:6]	rw	<b>CHE 中断的中断节点指针</b> 该位域定义了位 CHE（如果由 ENCHE 使能）或位 STR（如果由 ENSTR 使能）置位情况所激活的服务请求输出线。 编码见 INPCC6x。
<b>INPERR</b>	[9:8]	rw	<b>错误中断的中断节点指针</b>



符号	位序号	类型	功能描述
			该位域定义了位 TRPF（如果由 ENTRPF 使能）或位 WHE（如果由 ENWHE 使能）的置位情况所激活的服务请求输出线。 编码见 INPCC6x。
<b>INPT12</b>	[11:10]	rw	<b>定时器 T12 中断的中断节点指针</b> 该位域定义了位 T12OM（如果由 ENT12OM 使能）或位 T12PM（如果由 ENT12PM 使能）的置位情况所激活的服务请求输出线。 编码见 INPCC6x。
<b>INPT13</b>	[13:12]	rw	<b>定时器 T13 中断的中断节点指针</b> 该位域定义了位 T13CM（如果由 ENT13CM 使能）或位 T13PM（如果由 ENT13PM 使能）的置位情况所激活的服务请求输出线。 编码见 INPCC6x。
<b>0</b>	[15:14]	r	<b>保留；</b> 读操作返回 0；应写入 0。

## 20.9 一般模块操作

本节提供下述信息：

- 器件在不同工作模式下的配置（见[章节 20.9.1](#) 的模式控制描述）
- 输入选择（见[章节 20.9.2](#)）
- 一般寄存器描述（见[章节 20.9.3](#)）

### 20.9.1 模式控制

对于系统控制任务，如省电或用于调试的挂起请求，模式控制的概念允许编程设置不同器件工作条件下的模块行为。对于每个工作模式，CCU6 内核行为为可编程设定，通过 SCU 的全局状态控制部分进行请求。因此，CCU6 模块提供一个内核状态配置寄存器 **KSCCFG**，用来定义下述器件工作模式下的行为：

- **正常操作：**

该工作模式为缺省工作模式。当挂起请求或时钟关闭请求不被响应时，模块处于该模式。模块时钟不会被关闭且 CCU6 寄存器可读或写。内核行为由寄存器 **KSCCFG.NOMCFG** 定义。

- **挂起模式：**

当挂起请求（由调试器发出的）有效时，请求该工作模式。不关闭模块时钟，CCU6 寄存器可读或写。由 **KSCCFG.SUMCFG** 定义内核行为。

- **时钟关闭模式：**

为了降低功耗，可请求此工作模式。停止模式下，当 CCU6 模块内核达到它们的特定状态时，模块时钟自动关闭。在此情况下，不能访问 CCU6 寄存器。由 **KSCCFG.COMCFG** 定义内核行为。

内核分为四个不同模块（T12、T13、霍尔逻辑和强制中断逻辑）。这些模块停止模式 0 和停止模式 1 的请求可由敏感位 **KSCSR.SBx** 单独使能。如果禁止对请求敏感，则模块继续进行正常操作。如果使能对请求敏感，则该模块工作在选定的停止模式。

当所有四个模块达到定义的终止条件时，CCU6 的完整信息在 **GSC** 中给出。

表 20-12 CCU6 功能模块

模块	功能	敏感位
0	<b>定时器 T12:</b> 一直使能 T12 正常工作，直到达到特定停止条件。此时，T12 停止计数，CC6xIN 输入部分被冻结。	KSCSR.SB0
1	<b>定时器 T13:</b> 一直使能 T13 正常工作，直到达到特定停止条件。此时，T13 停止计数。	KSCSR.SB1
2	<b>霍尔逻辑:</b> 霍尔逻辑被立即停止且 CCPOSx 输入部分被冻结。	KSCSR.SB2
3	<b>强制中断逻辑:</b> 强制中断逻辑被立即停止且 CTRAP 输入部分被冻结。	KSCSR.SB3

对于每种器件工作模式（正常操作、挂起模式、时钟关闭模式），CCU6 内核的行为可编程设定。因此，CCU6 共支持四种内核模式，如表 20-13 所示。

表 20-13 CCU6 内核行为

内核模式	内核行为	编码
运行模式 0	按照规定进行内核操作，对 CCU6 的操作没有影响（运行模式 0 和运行模式 1 下，内核行为相同）	00 <sub>B</sub>
运行模式 1		01 <sub>B</sub>
停止模式 0	<p>需要考虑敏感位：</p> <p><b>T12 模块:</b> 定时器 T12 继续正常工作（如果其正在运行），直到 PWM 周期结束，然后停止运行（与单次模式下的停止条件相同）。当定时器停止时，CC6xIN 被冻结。</p> <p><b>T13 模块:</b> 定时器 T13 继续正常工作（如果其正在运行）直到 PWM 周期结束，然后停止运行（与单次模式下的停止条件相同）。</p> <p><b>霍尔逻辑模块:</b> CCPOSx 输入值被冻结。</p> <p><b>强制中断逻辑模块:</b> CTRAP 输入值被冻结</p>	10 <sub>B</sub>

内核模式	内核行为	编码
停止模式 1	<p>使能用作强制中断条件的输出线被设置为被动态（与强制中断状态相似）。需要考虑敏感位的影响：</p> <p><b>T12 模块：</b>定时器 T12 立即停止且 CC6xIN 输入被冻结。</p> <p><b>T13 模块：</b>定时器 T13 停止。</p> <p><b>霍尔逻辑模块：</b>CCPOSx 输入值被冻结。</p> <p><b>强制中断逻辑模块：</b>CTRAP 输入值被冻结。</p>	11 <sub>B</sub>

总的来说，位域 **KSCCFG.NOMCFG** 应该将运行模式 0 配置为标准操作的缺省设置。如果 **CCU6** 内核不应响应挂起请求（且继续工作在正常模式下），位域 **KSCCFG.SUMCFG** 的配置必须和 **KSCCFG.NOMCFG** 的值一样。如果 **CCU6** 内核操作行为应该有所不同，当达到特定停止条件时，**CCU6** 内核应该停止工作，停止模式 0 或停止模式 1 的编码应该写入到位域 **KSCCFG.SUMCFG**。

同样的机制适用于时钟关闭模式，可通过位域 **KSCCFG.COMCFG** 编程设置需要的内核操作行为。

*注：应用需要在很大程度上决定停止模式的选择，同一个应用不太可能同时需要两种不同的停止模式。因此，寄存器 **KSCCFG** 内应当仅需要一种停止模式类型（停止模式 0 或 1）。停止模式 0 和停止模式 1 不应混合使用，从而避免 **CCU6** 模块从停止模式 0 跳变到停止模式 1（反之亦然）。*

达到停止条件时（停止模式 0 或 1），如果 **KSCCFG.MODEN = 0** 禁止模块时钟或模块处于时钟关闭模式，读或写操作不能访问 **CCU6** 模块（寄存器 **KSCCFG** 例外，该寄存器始终可被访问）。结果是，无法配置 **CCU6** 模块。

请注意当所有配置位域设置为运行模式 0 时，位 **KSCCFG.MODEN** 只应由软件设置。

## 20.9.2 输入选择

通过编程设置端口输入选择寄存器 **PISELL** 和 **PISELH** 从 4 个或 8 个可能的输入中为每个 CCU6 输入选择信号。该特性允许根据器件应用需要调整引脚功能。

模块输出信号的输出引脚在端口中选择。

命名规则：

输入信号 CC60IN 的输入向量 CC60IN[D:A]由信号 CC60INA 到 CC60IND 组成。

*注：在这些输入影响模块内部逻辑之前，CCU6 的所有功能输入和  $f_{CC6}$  同步。从而会带来  $2/f_{CC6}$  的延迟，对于异步信号，还必须额外增加大约  $1/f_{CC6}$  的不确定延迟以进行精确时序计算。如果输入信号为高电平和为低电平的持续时间都大于  $1/f_{CC6}$ ，才能够正确检测到输入信号的边沿。*

## 20.9.3 一般寄存器

### 20.9.3.1 ID 寄存器

ID 寄存器为只读寄存器，给出 CCU6 模块 ID。该寄存器包含 8 位模块 ID 和 8 位版本号信息。

#### ID

模块 ID 寄存器

**XSFR (08<sub>H</sub>)**

复位值: **54XX<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MOD_NUMBER</b>								<b>MOD_REV</b>							
r								r							

符号	位序号	类型	功能描述
<b>MOD_REV</b>	[7:0]	r	<b>模块版本编号</b> 位 7-0 定义模块版本编号。模块版本编号的值从 01 <sub>H</sub> （第一版）开始，02 <sub>H</sub> 、03 <sub>H</sub> ，...，直到 FF <sub>H</sub> 。
<b>MOD_NUMBER</b>	[15:8]	r	<b>模块 ID 编号</b> 位 15-8 定义模块 ID 编号。 CCU6 的模块编号为 54 <sub>H</sub> 。

### 20.9.3.2 端口输入选择寄存器

使用寄存器 PISELL 和 PISELH 中的位域为模块输入选择实际的输入信号。

#### PISELL

端口输入选择寄存器低位

**XSFR (04<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>IST12HR</b>	<b>ISPOS2</b>	<b>ISPOS1</b>	<b>ISPOS0</b>	<b>ISTRP</b>	<b>ISCC62</b>	<b>ISCC61</b>	<b>ISCC60</b>								
rw	rw	rw	rw	rw	rw	rw	rw								

符号	位序号	类型	功能描述
<b>ISCC60</b>	[1:0]	rw	<b>CC60 的输入选择</b> 该位域定义用作 CC60 捕获输入的信号。 00 <sub>B</sub> 选择信号 CC60INA 01 <sub>B</sub> 选择信号 CC60INB 10 <sub>B</sub> 选择信号 CC60INC 11 <sub>B</sub> 选择信号 CC60IND
<b>ISCC61</b>	[3:2]	rw	<b>CC61 的输入选择</b> 该位域定义用作 CC61 捕获输入的信号。 00 <sub>B</sub> 选择信号 CC61INA 01 <sub>B</sub> 选择信号 CC61INB 10 <sub>B</sub> 选择信号 CC61INC 11 <sub>B</sub> 选择信号 CC61IND
<b>ISCC62</b>	[5:4]	rw	<b>CC62 的输入选择</b> 该位域定义用作 CC62 捕获输入的信号。 00 <sub>B</sub> 选择信号 CC62INA 01 <sub>B</sub> 选择信号 CC62INB 10 <sub>B</sub> 选择信号 CC62INC 11 <sub>B</sub> 选择信号 CC62IND
<b>ISTRP</b>	[7:6]	rw	<b>CTRAP 的输入选择</b> 该位域定义用作 CTRAP 输入的信号。 00 <sub>B</sub> 选择信号 CTRAPA 01 <sub>B</sub> 选择信号 CTRAPB

符号	位序号	类型	功能描述
			10 <sub>B</sub> 选择信号 CTRAPC 11 <sub>B</sub> 选择信号 CTRAPD
<b>ISPOS0</b>	[9:8]	rw	<b>CCPOS0 的输入选择</b> 该位域定义用作 CCPOS0 输入的信号。 00 <sub>B</sub> 选择信号 CCPOS0A 01 <sub>B</sub> 选择信号 CCPOS0B 10 <sub>B</sub> 选择信号 CCPOS0C 11 <sub>B</sub> 选择信号 CCPOS0D
<b>ISPOS1</b>	[11:10]	rw	<b>CCPOS1 的输入选择</b> 该位域定义用作 CCPOS1 输入的信号。 00 <sub>B</sub> 选择信号 CCPOS1A 01 <sub>B</sub> 选择信号 CCPOS1B 10 <sub>B</sub> 选择信号 CCPOS1C 11 <sub>B</sub> 选择信号 CCPOS1D
<b>ISPOS2</b>	[13:12]	rw	<b>CCPOS2 的输入选择</b> 该位域定义用作 CCPOS2 输入的信号。 00 <sub>B</sub> 选择信号 CCPOS2A 01 <sub>B</sub> 选择信号 CCPOS2B 10 <sub>B</sub> 选择信号 CCPOS2C 11 <sub>B</sub> 选择信号 CCPOS2D
<b>IST12HR</b>	[15:14]	rw	<b>T12HR 的输入选择</b> 该位域定义用作 T12HR 输入的信号。 00 <sub>B</sub> 选择信号 T12HRA (如果 T12EXT = 0) 或 T12HRE (如果 T12EXT = 1) 01 <sub>B</sub> 选择信号 T12HRB (如果 T12EXT = 0) 或 T12HRF (如果 T12EXT = 1) 10 <sub>B</sub> 选择信号 T12HRC (如果 T12EXT = 0) 或 T12HRG (如果 T12EXT = 1) 11 <sub>B</sub> 选择信号 T12HRD (如果 T12EXT = 0) 或 T12HRH (如果 T12EXT = 1)

**PISELH**

端口输入选择寄存器高位

**XSFR (06<sub>H</sub>)**

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								T13 EXT	T12 EXT	ISCNT13	ISCNT12		IST13HR		
r								rw	rw	rw	rw		rw		

符号	位序号	类型	功能描述
<b>IST13HR</b>	[1:0]	rw	<p><b>T13HR 的输入选择</b></p> <p>该位域定义用作 T13HR 输入的信号。</p> <p>00<sub>B</sub> 选择信号 T13HRA (如果 T13EXT = 0) 或信号 T13HRE (如果 T13EXT = 1)</p> <p>01<sub>B</sub> 选择信号 T13HRB (如果 T13EXT = 0) 或信号 T13HRF (如果 T13EXT = 1)</p> <p>10<sub>B</sub> 选择信号 T13HRC (如果 T13EXT = 0) 或信号 T13HRG (如果 T13EXT = 1)</p> <p>11<sub>B</sub> 选择信号 T13HRD (如果 T13EXT = 0) 或信号 T13HRH (如果 T13EXT = 1)</p>
<b>ISCNT12</b>	[3:2]	rw	<p><b>T12 计数输入的输入选择</b></p> <p>该位域定义引起 T12 计数动作的输入事件。</p> <p>00<sub>B</sub> 由 T12 预分频器产生计数事件。无需考虑位 TCTR4.T12CNT。</p> <p>01<sub>B</sub> 向位 TCTR4.T12CNT 写 1 的操作作为计数事件。无需考虑 T12 预分频器</p> <p>10<sub>B</sub> 对于选定的 T12HR 信号上检测的每个上升沿, 由定时器 T12 进行计数</p> <p>11<sub>B</sub> 对于选定的 T12HR 信号上检测的每个下降沿, 由定时器 T12 进行计数</p>
<b>ISCNT13</b>	[5:4]	rw	<p><b>T13 计数输入的输入选择</b></p> <p>该位域定义引起 T13 计数动作的输入事件。</p> <p>00<sub>B</sub> 由 T13 预分频器产生计数事件。无需考虑位 TCTR4.T13CNT。</p> <p>01<sub>B</sub> 向位 TCTR4.T13CNT 写 1 的操作作为计数事件。无需考虑 T13 预分频器</p>



**捕获/比较单元 6 (CCU6)**

符号	位序号	类型	功能描述
			<b>10<sub>B</sub></b> 对于选定的 T13HR 信号上检测的每个上升沿，由定时器 T13 进行计数  <b>11<sub>B</sub></b> 对于选定的 T13HR 信号上检测的每个下降沿，由定时器 T13 进行计数
<b>T12EXT</b>	6	rw	<b>T12HR 输入扩展</b> 该位域扩展 2 位位域 IST12HR。 <b>0<sub>B</sub></b> 从信号 T12HR[D:A]中选择一个 <b>1<sub>B</sub></b> 从信号 T12HR[H:E]中选择一个
<b>T12EXT</b>	7	rw	<b>T12HR 输入扩展</b> 该位域扩展 2 位位域 IST13HR。 <b>0<sub>B</sub></b> 从信号 T13HR[D:A]中选择一个。 <b>1<sub>B</sub></b> 从信号 T13HR[H:E]中选择一个。
<b>0</b>	[15:8]	r	<b>保留；</b> 读操作返回 0；应写入 0。

### 20.9.3.3 内核状态配置寄存器

对于不同的器件工作模式，通过内核状态配置寄存器 KSCCFG 选择期望的内核模式。

应用复位可复位位域 KSCCFG.NOMCFG 和位域 KSCCFG.COMCFG。位域 KSCCFG.SUMCFG 可由调试复位进行复位。

注：NOMCFG、SUMCFG 和 COMCFG 的编码描述见[表 20-13](#)。

#### KSCCFG

内核状态配置寄存器

XSFR (00H)

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	0	COMCFG	BP SUM	0	SUMCFG	BP NOM	0	NOMCFG	0	BP MOD EN	MOD EN				
w	r	rw	w	r	rw	w	r	rw	r	w	rw				

符号	位序号	类型	功能描述
<b>MODEN</b>	0	rw	<b>模块使能</b> 该位使能模块内核时钟和模块功能。 0 <sub>B</sub> 模块被立即关闭（无需考虑停止条件）。 该模块不响应模块控制操作，模块时钟关闭。 模块不能响应读访问并且忽略写访问（KSCCFG 除外）。 1 <sub>B</sub> 模块开启，可以正常工作。 向 MODEN 写 1 之后，访问其它 CCU6 寄存器之前，推荐读取寄存器 KSCCFG 以避免流水线对控制模块的影响。
<b>BPMODEN</b>	1	w	<b>MODEN 的位保护</b> 该位使能对 MODEN 的写操作。读该位始终返回 0。 0 <sub>B</sub> MODEN 不改变 1 <sub>B</sub> 由写入值更新 MODEN
<b>NOMCFG</b>	[5:4]	rw	<b>正常操作模式配置</b> 该位域定义适用于正常操作模式的内核模式。

符号	位序号	类型	功能描述
			00 <sub>B</sub> 选择运行模式 0 01 <sub>B</sub> 选择运行模式 1 10 <sub>B</sub> 选择停止模式 0 11 <sub>B</sub> 选择停止模式 1
<b>BP<sub>NOM</sub></b>	7	w	<b>NOMCFG 的位保护</b> 该位使能对 NOMCFG 的写操作。读该位始终返回 0。 0 <sub>B</sub> NOMCFG 不改变 1 <sub>B</sub> 由写入值更新 NOMCFG。
<b>SUMCFG</b>	[9:8]	rw	<b>挂起模式配置</b> 该位域定义适用于挂起模式的内核模式。 编码与 NOMCFG 相似。
<b>BPSUM</b>	11	w	<b>SUMCFG 的位保护</b> 该位使能对 SUMCFG 的写操作。读该位始终返回 0。 0 <sub>B</sub> SUMCFG 不改变 1 <sub>B</sub> 由写入值更新 SUMCFG
<b>COMCFG</b>	[13:12]	rw	<b>时钟关闭模式配置</b> 该位域定义适用于时钟关闭模式的内核模式。编码与 NOMCFG 相似。
<b>BPCOM</b>	15	w	<b>COMCFG 的位保护</b> 该位使能对 COMCFG 的写操作。读该位始终返回 0。 0 <sub>B</sub> COMCFG 不改变 1 <sub>B</sub> 由写入值更新 COMCFG
<b>0</b>	[3:2], 6, 10, 14	r	<b>保留；</b> 读操作返回 0；应写入 0。

注：位 BP<sub>xxx</sub> 的位保护允许由单次写操作修改部分配置位（无需 CPU 使用读取-修改-回写机制）。

### 20.9.3.4 内核状态控制敏感寄存器

内核状态控制敏感寄存器位用来定义停止模式 0 和 1 影响哪个内部模块。

#### KSCSR

##### 内核状态控制敏感寄存器

**XSFR (0EH)**

**复位值: 0000H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>												<b>SB3</b>	<b>SB2</b>	<b>SB1</b>	<b>SB0</b>
r												rw	rw	rw	rw

符号	位序号	类型	功能描述
<b>SB0,</b> <b>SB1,</b> <b>SB2,</b> <b>SB3</b>	0, 1, 2, 3	rw	<b>模块 x 的敏感位</b> 该位定义 CCU6 内核的模块 x 是否对停止模式 0 或停止模式 1 敏感。模块功能的定义由表 20-12 给出。 0 <sub>B</sub> 模块 x 对停止模式 0 或停止模式 1 不敏感，该模块的操作与运行模式 0 相似。无需考虑定义的停止条件，继续正常操作。 1 <sub>B</sub> 模块 x 对停止模式 0 或停止模式 1 敏感，需要考虑定义的停止条件
<b>0</b>	[15:4]	r	<b>保留；</b> 读操作返回 0；应写入 0。

## 20.10 CCU6 的实现

本节描述 XE16xyM 器件中的 CCU6 模块的具体实现。

- 地址映射（见[章节 20.10.1](#)）
- 中断控制寄存器（见[章节 20.10.2](#)）
- 同步启动（见[章节 20.10.3](#)）
- CCU60 的连接（见[章节 20.10.4.1](#)）
- CCU61 的连接（见[章节 20.10.4.2](#)）
- CCU62 的连接（见[章节 20.10.4.3](#)）
- CCU63 的连接（见[章节 20.10.4.4](#)）

### 20.10.1 地址映射

XE16xyM 的四个 CCU6 模块（命名为 CCU60 到 CCU63）的访问地址见下面的表格。

准确寄存器地址可由模块的内核基址（见[表 20-14](#)）加上偏移地址（见[表 20-1](#)）得到。

**表 20-14 寄存器的地址空间**

模块	基址	结束地址	注
CC60	EA00 <sub>H</sub>	EA7E <sub>H</sub>	
CC61	EA80 <sub>H</sub>	EAFE <sub>H</sub>	
CC62	EB00 <sub>H</sub>	EB7E <sub>H</sub>	
CC63	EB80 <sub>H</sub>	EBFE <sub>H</sub>	

**表 20-15 寄存器概述**

寄存器缩略名	寄存器全名	偏移地址	页码
请参考 <a href="#">章节 20.1.3</a> 中的寄存器表			

## 20.10.2 中断控制寄存器

中断控制寄存器位于 SFR 区域内。关于这些寄存器的描述见中断一章。

**表 20-16 CCU6 中断控制寄存器**

缩略名	描述
<b>CCU60_0IC</b>	CCU60 模块 SR0 中断控制寄存器
<b>CCU60_1IC</b>	CCU60 模块 SR1 中断控制寄存器
<b>CCU60_2IC</b>	CCU60 模块 SR2 中断控制寄存器
<b>CCU60_3IC</b>	CCU60 模块 SR3 中断控制寄存器
<b>CCU61_0IC</b>	CCU61 模块 SR0 中断控制寄存器
<b>CCU61_1IC</b>	CCU61 模块 SR1 中断控制寄存器
<b>CCU61_2IC</b>	CCU61 模块 SR2 中断控制寄存器
<b>CCU61_3IC</b>	CCU61 模块 SR3 中断控制寄存器
<b>CCU62_0IC</b>	CCU62 模块 SR0 中断控制寄存器
<b>CCU62_1IC</b>	CCU62 模块 SR1 中断控制寄存器
<b>CCU62_2IC</b>	CCU62 模块 SR2 中断控制寄存器
<b>CCU62_3IC</b>	CCU62 模块 SR3 中断控制寄存器
<b>CCU63_0IC</b>	CCU63 模块 SR0 中断控制寄存器
<b>CCU63_1IC</b>	CCU63 模块 SR1 中断控制寄存器
<b>CCU63_2IC</b>	CCU63 模块 SR2 中断控制寄存器
<b>CCU63_3IC</b>	CCU63 模块 SR3 中断控制寄存器

### 20.10.3 同步启动特性

由 SCU 模块中的位 **SYSCON1.GLCCST** (全局捕获/比较启动) 支持同步启动特性。该位和所有 CCU6x 模块的 T12HR 和 T13HR 输入相连。

为了同步启动捕获/比较单元定时器，同样的信号还连接到其它捕获/比较单元。

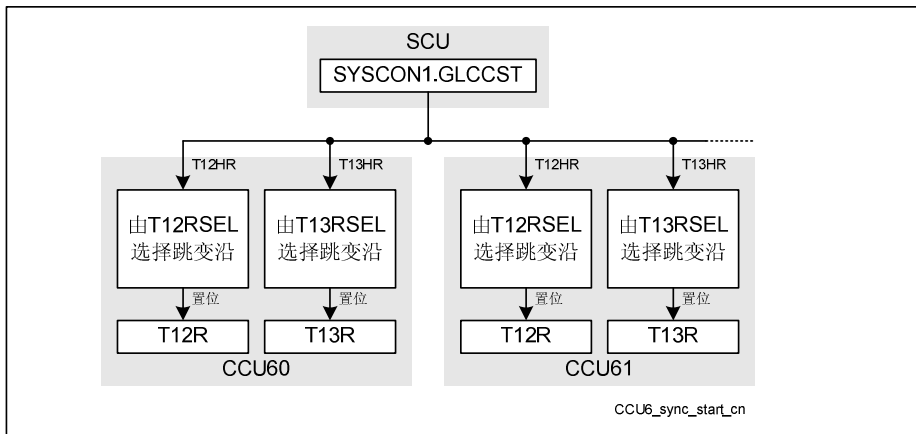


图 20-43 同步概念

### 20.10.4 数字信号的连接

下表给出 XE16xyM 器件中 CCU6x 模块与其它模块或引脚的信号连接方式。

从 4 个或 8 个可能的输入线中选择每个输入信号，如输入信号 CC60IN 的输入向量由 CC60IN[D:A]组成，而 T12HR 和 T13HR 的输入向量由 T12HR[H:A]和 T13HR[H:A]组成。

下面各节给出接口信号，每个 CCU6x 中断控制寄存器与服务请求输出 SR[3:0]的连接见[章节 20.10.2](#)。

CCU6x 模块的时钟为 XE16xyM 系统时钟，所以  $f_{CC6} = f_{SYS}$ 。

**注：**在影响模块内部逻辑之前，CCU6 的所有功能性输入和  $f_{CC6}$  同步。从而会带来  $2/f_{CC6}$  的延迟，进行精确时序计算还必须额外增加大约  $1/f_{CC6}$  的不确定延迟。

如果输入信号为高电平和为低电平的持续时间都大于  $1/f_{CC6}$ ，才能够正确检测到输入信号的边沿。

### 20.10.4.1 CCU60 的连接

下表描述 CCU60 模块的相互连接。

**表 20-17 XE16xyM 内 CCU60 的数字信号相互连接**

信号	来自/送至 哪个模块	I/O 至 CCU60	可用作...
CC60INA	P10.0	I	通道 CC60 捕获事件的输入信号
CC60INB	P8.0	I	
CC60INC	0	I	
CC60IND	RTC 中断	I	
CC61INA	P10.1	I	通道 CC61 捕获事件的输入信号
CC61INB	P8.1	I	
CC61INC	0	I	
CC61IND	WUT 触发 (SCU)	I	
CC62INA	P10.2	I	通道 CC62 捕获事件的输入信号
CC62INB	P8.2	I	
CC62INC	0	I	
CC62IND	0	I	
CTRAPA	P10.6	I	CTRAP 的输入信号, ESRx 输入指同步之后的输入信号, 该信号可被滤波 (如果被使能)
CTRAPB	P8.6	I	
CTRAPC	ESR2	I	
CTRAPD	ERU_PDOUT2	I	
CCPOS0A	P10.7	I	CCPOS0 的输入信号
CCPOS0B	P9.7	I	
CCPOS0C	CCU61_SR3	I	
CCPOS0D	0	I	
CCPOS1A	P10.8	I	CCPOS1 的输入信号



**捕获/比较单元 6 (CCU6)**

信号	来自/送至 哪个模块	I/O 至 CCU60	可用作...
CCPOS1B	P9.6	I	边沿检测关闭
CCPOS1C	CC63_SR3	I	
CCPOS1D	0	I	
CCPOS2A	P10.9	I	CCPOS2 的输入信号
CCPOS2B	P9.5	I	
CCPOS2C	ADC0_SR3	I	
CCPOS2D	0	I	
T12HRA	CCU63_MCM_ST	I	T12HR 的输入信号
T12HRB	P5.5	I	
T12HRC	P5.8	I	
T12HRD	SYSCON1.GLCCST	I	
T12HRE	ADC0_ARBCNT	I	
T12HRF	0	I	
T12HRG	0	I	
T12HRH	0	I	
T13HRA	EXTCLK (SCU)	I	T13HR 的输入信号
T13HRB	CCU60_T12_ZM	I	
T13HRC	P5.8	I	
T13HRD	SYSCON1.GLCCST	I	
T13HRE	ADC0_ARBCNT	I	
T13HRF	0	I	
T13HRG	U0C0_SR3	I	
T13HRD	0	I	
CC60	P10.0 P8.0	O	通道 CC60 比较输出

**捕获/比较单元 6 (CCU6)**

信号	来自/送至 哪个模块	I/O 至 CCU60	可用作...
CC60	ADC0_REQGT0H	O	
COUT60	P10.3 P8.3	O	
CC61	P10.1 P8.1	O	通道 CC61 比较输出
CC61	ADC0_REQGT1H	O	
COUT61	P10. 4 P8.4	O	
CC62	P10.2 P8.2	O	通道 CC62 比较输出
CC62	ADC0_REQGT2H	O	
COUT62	P10. 5 P8.5	O	
COUT63	P10. 7 P10.10 P8.6 U0C0_DX2F U0C1_DX2F	O	通道 CC63 比较输出
COUT63	ADCx_REQGTyA	O	ADC 触发
T12_ZM	CCU60_T13HRB	O	T12 0-匹配
T13_PM	ERU_OGU02	O	T13 周期匹配
MCM_ST	CCU61_T12HRA ERU_OGU01	O	MCM 映射传送
SR3	CCU61_CCPOS0C	O	CCU61 触发
SR3	CCU63_CCPOS1C	O	CCU63 触发

#### 20.10.4.2 CCU61 的连接

下表描述 CCU61 模块的相互连接。

**表 20-18 XE16xyM 内 CCU61 的数字信号相互连接**

信号	来自/送至 哪个模块	I/O 至 CCU61	可用作...
CC60INA	P0.0	I	通道 CC60 捕获事件的输入信号
CC60INB	P11.5	I	
CC60INC	0	I	
CC60IND	0	I	
CC61INA	P0.1	I	通道 CC61 捕获事件的输入信号
CC61INB	P11.2	I	
CC61INC	0	I	
CC61IND	0	I	
CC62INA	P0.2	I	通道 CC62 捕获事件的输入信号
CC62INB	P11.4	I	
CC62INC	0	I	
CC62IND	0	I	
CTRAPA	P0.6	I	CTRAP 的输入信号，ESRx 输入指同步之后的输入信号，该信号可被滤波（如果被使能）
CTRAPB	P0.7	I	
CTRAPC	ESR2	I	
CTRAPD	P11.1	I	
CCPOS0A	P4.5	I	CCPOS0 的输入信号
CCPOS0B	0	I	
CCPOS0C	CCU60_SR3	I	
CCPOS0D	0	I	
CCPOS1A	P4.6	I	CCPOS1 的输入信号

**捕获/比较单元 6 (CCU6)**

信号	来自/送至 哪个模块	I/O 至 CCU61	可用作...
CCPOS1B	0	I	边沿检测关闭
CCPOS1C	CCU62_SR3	I	
CCPOS1D	0	I	
CCPOS2A	P4.7	I	CCPOS2 的输入信号
CCPOS2B	0	I	
CCPOS2C	ADC1_SR3	I	
CCPOS2D	0	I	
T12HRA	CCU60_MCM_ST	I	T12HR 的输入信号
T12HRB	P1.2	I	
T12HRC	P5.8	I	
T12HRD	SYSCON1.GLCCST	I	
T12HRE	ADC0_ARBCNT	I	
T12HRF	0	I	
T12HRG	0	I	
T12HRH	0	I	
T13HRA	P5.10	I	T13HR 的输入信号
T13HRB	CCU61_T12_ZM	I	
T13HRC	P5.8	I	
T13HRD	SYSCON1.GLCCST	I	
T13HRE	ADC0_ARBCNT	I	
T13HRF	P11.3	I	
T13HRG	U1C0_SR3	I	
T13HRH	0	I	
CC60	P0.0 P11.5	O	通道 CC60 比较输出

**捕获/比较单元 6 (CCU6)**

信号	来自/送至 哪个模块	I/O 至 CCU61	可用作...
COUT60	P0.3 P11.0	O	
CC61	P0.1 P11.2	O	通道 CC61 比较输出
COUT61	P0.4 P11.1	O	
CC62	P0.2 P11.4	O	通道 CC62 比较输出
COUT62	P0.5 P11.3	O	
COUT63	P0.6 P11.3 P11.5 U1C0_DX2F U1C1_DX2F	O	通道 CC63 比较输出
COUT63	ADCx_REQGTyB	O	ADC 触发
T12_ZM	CCU61_T13HRB	O	T12 0-匹配
T13_PM	ERU_OGU12	O	T13 周期匹配
MCM_ST	CCU62_T12HRA ERU_OGU11	O	MCM 映射传送
SR3	CCU60_CCPOS0C	O	CCU60 触发
SR3	CCU62_CCPOS1C	O	CCU62 触发
SR3	ADC0_REQTRyC	O	ADC0 触发

### 20.10.4.3 CCU62 的连接

下表描述 CCU62 模块的相互连接。

**表 20-19 XE16xyM 内 CCU62 的数字信号相互连接**

信号	来自/送至 哪个模块	I/O 至 CCU62	可用作...
CC60INA	P1.7	I	通道 CC60 捕获事件的输入信号
CC60INB	P8.3	I	
CC60INC	0	I	
CC60IND	0	I	
CC61INA	P1.6	I	通道 CC61 捕获事件的输入信号
CC61INB	P8.4	I	
CC61INC	0	I	
CC61IND	0	I	
CC62INA	P1.2	I	通道 CC62 捕获事件的输入信号
CC62INB	P8.5	I	
CC62INC	0	I	
CC62IND	0	I	
CTRAPA	P7.1	I	CTRAP 的输入信号，ESRx 输入指同步之后的输入信号，该信号可被滤波（如果被使能）
CTRAPB	P1.0	I	
CTRAPC	ESR2	I	
CTRAPD	P8.6	I	
CCPOS0A	P7.2	I	CCPOS0 的输入信号
CCPOS0B	P4.1	I	
CCPOS0C	CCU63_SR3	I	
CCPOS0D	0	I	
CCPOS1A	P7.3	I	CCPOS1 的输入信号

**捕获/比较单元 6 (CCU6)**

信号	来自/送至 哪个模块	I/O 至 CCU62	可用作...
CCPOS1B	P4.2	I	边沿检测关闭
CCPOS1C	CCU61_SR3	I	
CCPOS1D	0	I	
CCPOS2A	P7.4	I	CCPOS2 的输入信号
CCPOS2B	P4.3	I	
CCPOS2C	ADC0_SR3	I	
CCPOS2D	0	I	
T12HRA	CCU61_MCM_ST	I	T12HR 的输入信号
T12HRB	P1.3	I	
T12HRC	P5.8	I	
T12HRD	SYSCON1.GLCCST	I	
T12HRE	ADC0_ARBCNT	I	
T12HRF	0	I	
T12HRG	0	I	
T12HRH	0	I	
T13HRA	CAN_INT_O15	I	T13HR 的输入信号
T13HRB	CCU62_T12_ZM	I	
T13HRC	P5.8	I	
T13HRD	SYSCON1.GLCCST	I	
T13HRE	ADC1_ARBCNT	I	
T13HRF	0	I	
T13HRG	U2C0_SR3	I	
T13HRH	0	I	
CC60	P1.7 P8.3	O	通道 CC60 比较输出

**捕获/比较单元 6 (CCU6)**

信号	来自/送至 哪个模块	I/O 至 CCU62	可用作...
COUT60	P1.5 P9.7	O	
CC61	P1.6 P8.4	O	通道 CC61 比较输出
COUT61	P1.4 P9.6	O	
CC62	P1.2 P8.5	O	通道 CC62 比较输出
COUT62	P1.1 P9.5	O	
COUT63	P1.3 P9.4 P9.7 U2C0_DX2F U2C1_DX2F	O	通道 CC63 比较输出
COUT63	ADCx_REQGTyC	O	ADC 触发
T12_ZM	CCU62_T13HRB	O	T12 0-匹配
T13_PM	ERU_OGU22	O	T13 周期匹配
MCM_ST	CCU63_T12HRA ERU_OGU21	O	MCM 映射传送
SR3	CCU61_CCPOS1C	O	CCU61 触发
SR3	CCU63_CCPOS0C	O	CCU63 触发
SR3	ADC1_REQTRyC	O	ADC1 触发



#### 20.10.4.4 CCU63 的连接

下表描述 CCU63 模块的相互连接。

**表 20-20 XE16xyM 内 CCU63 的数字信号相互连接**

信号	来自/送至 哪个模块	I/O 至 CCU63	可用作...
CC60INA	P9.0	I	通道 CC60 捕获事件的输入信号
CC60INB	P2.0	I	
CC60INC	0	I	
CC60IND	0	I	
CC61INA	P9.1	I	通道 CC61 捕获事件的输入信号
CC61INB	P2.1	I	
CC61INC	0	I	
CC61IND	0	I	
CC62INA	P9.2	I	通道 CC62 捕获事件的输入信号
CC62INB	P2.2	I	
CC62INC	0	I	
CC62IND	0	I	
CTRAPA	P9.6	I	CTRAP 的输入信号，ESRx 输入指同步之后的输入信号，该信号可被滤波（如果被使能）
CTRAPB	P9.7	I	
CTRAPC	ESR2	I	
CTRAPD	ERU_PDOUT3	I	
CCPOS0A	P11.0	I	CCPOS0 的输入信号
CCPOS0B	0	I	
CCPOS0C	CCU62_SR3	I	
CCPOS0D	0	I	
CCPOS1A	P11.1	I	CCPOS1 的输入信号

**捕获/比较单元 6 (CCU6)**

信号	来自/送至 哪个模块	I/O 至 CCU63	可用作...
CCPOS1B	0	I	边沿检测关闭
CCPOS1C	CCU60_SR3	I	
CCPOS1D	0	I	
CCPOS2A	P11.2	I	CCPOS2 的输入信号
CCPOS2B	0	I	
CCPOS2C	ADC1_SR3	I	
CCPOS2D	0	I	
T12HRA	CCU62_MCM_ST	I	T12HR 的输入信号
T12HRB	P5.4	I	
T12HRC	P5.8	I	
T12HRD	SYSCON1.GLCCST	I	
T12HRE	ADC1_ARBCNT	I	
T12HRF	0	I	
T12HRG	0	I	
T12HRH	0	I	
T13HRA	CAN_INT_O15	I	T13HR 的输入信号
T13HRB	CCU63_T12_ZM	I	
T13HRC	P5.8	I	
T13HRD	SYSCON1.GLCCST	I	
T13HRE	ADC1_ARBCNT	I	
T13HRF	P5.13	I	
T13HRG	U3C0_SR3	I	
T13HRH	0	I	
CC60	P9.0	O	通道 CC60 比较输出

**捕获/比较单元 6 (CCU6)**

信号	来自/送至 哪个模块	I/O 至 CCU63	可用作...
	P2.0		
CC60	ADC1_REQGT0H	O	
COUT60	P9.3	O	
CC61	P9.1 P2.1	O	通道 CC61 比较输出
CC61	ADC1_REQGT1H	O	
COUT61	P9.4	O	
CC62	P9.2 P2.2	O	通道 CC62 比较输出
CC62	ADC1_REQGT2H	O	
COUT62	P9.5 P9.6	O	
COUT63	P9.6 P2.3 U3C0_DX2F U3C1_DX2F	O	通道 CC63 比较输出
COUT63	ADCx_REQGTyD	O	
T12_ZM	CCU63_T13HRB	O	T12 0-匹配
T13_PM	ERU_OGU32	O	T13 周期匹配
MCM_ST	CCU60_T12HRA ERU_OGU31	O	MCM 映射传送
SR3	CCU60_CCPOS1C	O	CCU60 触发
SR3	CCU62_CCPOS0C	O	CCU62 触发

## 21 通用串行接口通道

通用串行接口通道（USIC）是一个灵活的接口模块，支持多种串行通信协议。一个 USIC 模块包含两个独立的通信通道 UxC0 和 UxC1，x 代表 USIC 模块的编号（比如，USIC 模块 x 中的通道 y 用 UxCy 表示）。用户可在器件工作期间进行编程，设定每路通信通道将处理哪种协议、使用哪些引脚。

本章所涉及的内容如下：

- 简介（见[页 21-1](#)）
- USIC 操作（见[页 21-10](#)）
- 支持 UART 和 LIN 的 ASC 协议（见[页 21-93](#)）
- SSC 协议（见[页 21-111](#)）
- IIC 协议（见[页 21-134](#)）
- IIS 协议（见[页 21-155](#)）
- XE16xyM 中模块的实现（见[页 21-172](#)）

### 21.1 简介

本节将简要介绍 USIC 的特性及结构。内容包括：

- USIC 特性概述（见[页 21-1](#)）
- 通道结构（见[页 21-4](#)）
- 输入级（见[页 21-4](#)）
- 输出信号（见[页 21-5](#)）
- 波特率发生器（见[页 21-6](#)）
- 通道事件和中断（见[页 21-7](#)）
- 数据移位和处理（见[页 21-7](#)）

#### 21.1.1 特性概述

每路 USIC 通道可被单独配置以满足应用的需要，比如可在运行期间选择或修改协议类型、无需复位。USIC 支持以下协议：

- **UART**（ASC，异步串行通道）
  - 模块功能：接收器/发送器，最大波特率  $f_{\text{sys}}/4$
  - 波特率范围宽，最低可达单个位波特率
  - 每个数据帧包含的数据位个数：1-63
  - MSB 或 LSB 在先

- 硬件支持的 **LIN** 协议（本地互连网络）
  - 基于 **ASC** 协议的数据传送
  - 可由内置波特率发生器的捕获事件实现波特率检测
  - 软件控制校验和产生（实现更高的灵活性）
- **SSC/SPI**（带有/不带从控选择线的同步串行通道）
  - 模块功能：最大波特率  $f_{\text{SYS}}/2$ ，受限于环路延迟
  - 每帧数据包含 1-63 位数据，对于超过 63 位的数据帧，需明确定义数据帧的结束
  - **MSB** 或 **LSB** 在先
- **IIC**（Inter-IC 总线）
  - 波特率范围：100 kbit/s – 400 kbit/s
  - 支持 7 位或 10 位寻址
  - 完整的主控和从控器件功能
- **IIS**（信息娱乐音频总线）
  - 模块功能：最大波特率  $f_{\text{SYS}}/2$

注：实际应用中可以获得的波特率由器件的工作频率、数据手册中描述的时序参数、PCB 上的信号延迟、以及器件时序共同决定。

**USIC** 结构不仅能灵活选择通信协议，还能降低系统负载（CPU 负载）、从而保证高效的数据处理。**USIC** 的结构设计基于以下方面的考虑：

- **数据缓存功能**

标准数据缓存包含一个接收数据的双字缓存和一个发送数据的单字缓存，从而能够延长 CPU 的反应时间（如中断响应延迟）。

- **附加的 FIFO 缓存功能**

除标准数据缓存之外，接收到的数据和将要发送的数据可缓存在一个 **FIFO** 缓存结构中。可分别设定接收和发送 **FIFO** 缓存的大小。根据实际应用的需要，一个 **USIC** 模块中共有 64 个数据字的缓存可分配给接收和发送 **FIFO**（**USIC** 的两路通道共用 64 个数据字缓存）。

除 **FIFO** 缓存功能之外，旁路机制可在不刷新 **FIFO** 缓存的情况下引入高优先级数据。

- **发送控制信息**

在将要发送的每个数据字上附加了一个 5 位的发送控制信息，用于自动控制诸如字长、帧长、从控选择（针对 **SPI** 协议）等发送参数。通过分析将要发送的数据字的存放地址，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）。

可利用该特性对每个数据字进行单独处理。例如，和发送 FIFO 中的数据字相关的发送控制信息可自动修改从控选择输出，从而灵活选择不同的通信目标（从控器件）、无需 CPU 干预。此外，还可利用它来控制帧长。

- **灵活的帧长控制**

数据帧的长度和数据字长无关，可通过两种不同的方式控制帧长。第一种选择是自动生成已知长度（最多 63 位）的数据帧；第二种选择是产生更长（甚至不限长度）、或动态控制长度的数据帧。

- **中断功能**

根据实际应用的需要，每路 USIC 通道的事件可单独分配给 4 路服务请求输出 SR[3:0]之一。除特定协议事件之外，还可通过中断指示一帧的开始和结束。

- **灵活的接口连接**

每路 USIC 通道为通信信号提供了多种输入和输出引脚的连接选择，从而无需复位器件即可将改变 USIC 信号的引脚分配。

- **输入调整**

每个输入信号由一个可编程输入调整电路进行处理，该电路具有可编程滤波和同步功能。

- **波特率产生**

每路 USIC 通道包含一个专用波特率发生器，可基于内部模块时钟或外部频率输入产生波特率。该结构可产生内部无法产生的频率用于传送数据（如用于同步多个通信方）。

- **传送触发功能**

主控模式下，可由 USIC 模块外部产生的事件（例如输入引脚或定时器单元（发送数据有效性检验））触发数据传送。该特性实现了基于时间进行数据传送。

- **支持调试**

USIC 提供特定的地址以读取接收数据、不影响 FIFO 缓存。该特性可确保调试器进行读访问时不破坏接收数据序列。

为了达到期望波特率，必须考虑两个标准：模块功能和应用环境。模块功能由模块的输入频率决定，这是模块工作的基础。尽管模块本身可达到的波特率较高（取决于模块时钟和表示一个数据位所需要的模块时钟周期数），但实际可达到的波特率往往受到应用环境的限制。大多数情况下，由于驱动延迟、信号传输时间或 EMI 等因素的影响，应用环境会限制可达到的最大波特率。

*注：所选择的额外功能（如数字滤波、输入同步、采样点调整、数据结构等）会限制可达到的最大波特率。此外，还需注意附加延迟，比如（内部或外部的）传输延迟和驱动延迟（例如用于 ASC 模式下的冲突检测等）。*

### 21.1.2 通道结构

一个 USIC 模块包含两个独立的通信通道，通道结构如图 21-1 所示。

每路通道的数据移位单元和数据缓存支持全双工数据传送。和特定协议相关的操作由协议预处理器（PPP）处理。为了简化数据处理，每个 USIC 模块可通过一个 FIFO 数据缓存来保存每路通道的发送和接收数据。FIFO 数据缓存不一定在所有器件中均可用（参见章节 21.7.2）。

由于每路通信通道具有独立的通道控制和波特率产生，因此各通道的通信协议、波特率以及数据格式可单独编程。

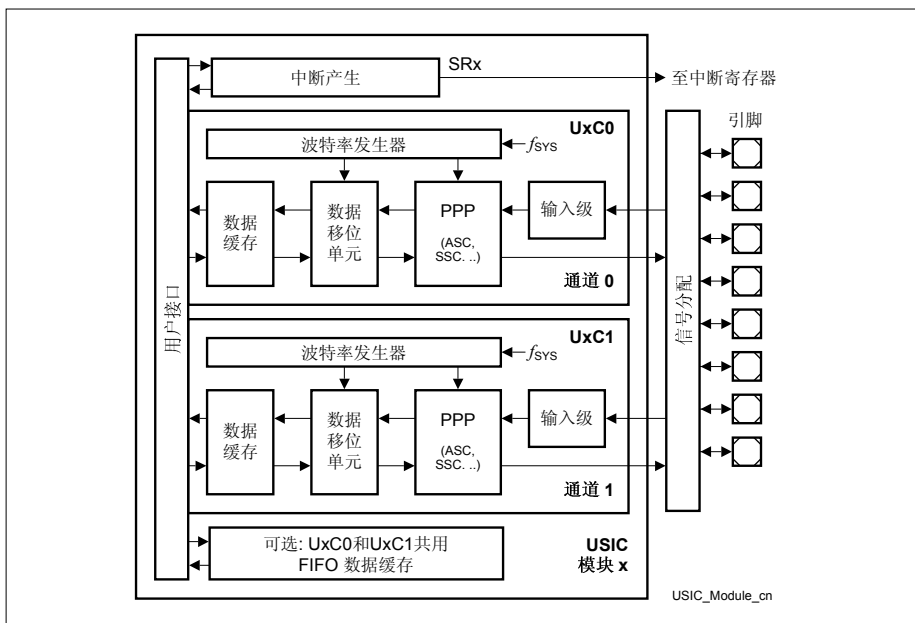


图 21-1 USIC 模块/通道结构

### 21.1.3 输入级

每种协议最多使用 3 路输入信号，实际用到的输入信号个数和选择的协议有关。每路输入信号经输入级（称作 DX0、DX1、DX2）处理，进行输入选择、极性控制或数字输入滤波等信号调整。根据信号在协议中的含义进行信号分类，见表 21-1。

标注为“可选”的输入信号在实现协议的标准功能时不使用、可用于功能扩展。和特定协议相关的信号描述将在相应的协议章节中给出，关于外部频率输入的描述请参见波特率发生器；关于发送数据有效性检验的描述请参见数据处理一节。

**表 21-1      不同协议的输入信号**

所选协议	移位数据输入 (经 DX0 处理)	移位时钟输入 (经 DX1 处理)	移位控制输入 (经 DX2 处理)
<b>ASC、LIN</b>	RXD	可选： 外部频率输入 或 TXD 冲突检测	可选： 发送数据有效性检验
<b>SSC、SPI (主控)</b>	DIN (MRST、MISO)	可选： 外部频率输入 或延迟补偿	可选： 发送数据有效性检验 或延迟补偿
<b>SSC、SPI (从控)</b>	DIN (MTSR、MOSI)	SCLKIN	SELIN
<b>IIC</b>	SDA	SCL	可选： 发送数据有效性检验
<b>IIS (主控)</b>	DIN	可选： 外部频率输入 或延迟补偿	可选： 发送数据有效性检验 或延迟补偿
<b>IIS (从控)</b>	DIN	SCLKIN	WAIN

*注：将所需的 USIC 输入功能分配给器件的端口引脚时，为了实现一定的灵活性，每个输入级可从多个可选的输入位置中选择所期望的输入位置。*

*在“模块的实现”一节中列出所有可用的 USIC 信号和它们的端口位置，参见 [页 21-178](#)。*

### 21.1.4 输出信号

每种协议最多使用 11 路输出信号，实际用到的输出信号个数和选择的协议有关。根据信号在协议中的含义进行信号分类，见[表 21-2](#)。

标注为“可选”的输出信号在实现协议的标准功能时不使用、可用于功能扩展。和特定协议相关的信号描述将在相应的协议章节中给出。为了使从控器件和主控器件同步，MCLKOUT 输出和移位时钟输出之间具有固定的频率关系（MCLKOUT 频率可高于 SCLKOUT 频率）。若某协议不使用波特率发生器（比如 SSC 从控模式），SCLKOUT 和 MCLKOUT 信号可用作占空比为 50%、频率和通信波特率无关的时钟输出。



**表 21-2      不同协议的输出信号**

所选协议	移位数据输出 <b>DOUT</b>	移位时钟输出 <b>SCLKOUT</b>	移位控制输出 <b>SELO[7:0]</b>	主控时钟输出 <b>MCLKOUT</b>
<b>ASC、LIN</b>	TXD	不使用	不使用	可选： 主控时间基准
<b>SSC、SPI (主控)</b>	DOUT (MTSR、 MOSI)	主控移位时钟	从控选择、 片选	可选： 主控时间基准
<b>SSC、SPI (从控)</b>	DOUT (MRST、 MISO)	可选： 独立的时钟输出	不使用	可选： 独立的时钟输出
<b>IIC</b>	SDA	SCL	不使用	可选： 主控时间基准
<b>IIS (主控)</b>	DOUT	主控移位时钟	<b>WA</b>	可选： 主控时间基准
<b>IIS (从控)</b>	DOUT	可选： 独立的时钟输出	不使用	可选： 独立的时钟输出

*注：将所需的 **USIC** 输出功能分配给器件的端口引脚时，为了实现一定的灵活性，多数输出信号可从多个端口引脚上输出。端口控制电路按照引脚顺序依次定义哪些信号可用作一个端口引脚的输出信号（见端口一章的描述）。*

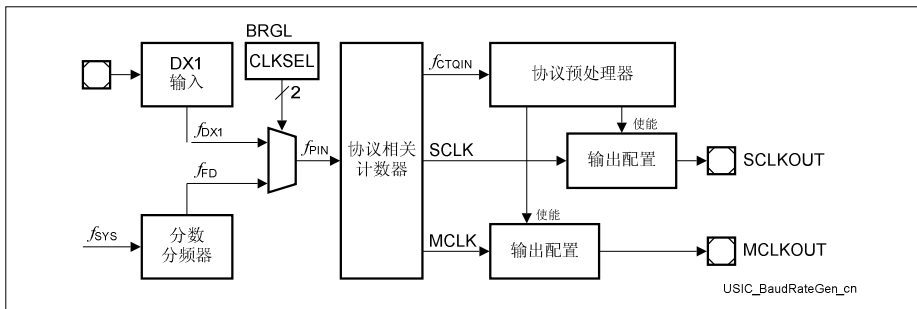
*在“模块的实现”一节中列出所有可用的 **USIC** 信号和它们的端口位置，参见 [页 21-178](#)。*

### 21.1.5    波特率发生器

每路 **USIC** 通道包含一个波特率发生器，结构如[图 21-2](#)所示。它基于多步分频处理，为不同协议提供所需的频率。波特率发生器包含：

- 分数分频器，基于内部系统频率  $f_{SYS}$  产生输入频率  $f_{PIN} = f_{FD}$ 。
- **DX1** 输入，基于外部信号产生输入频率  $f_{PIN} = f_{DX1}$ 。
- 协议相关计数器，提供主控时钟信号 **MCLK**、移位时钟信号 **SCLK** 以及和协议相关的信号。它还可用来测量时间间隔，比如波特率检测。
- 和协议预处理器关联的时间单元计数器，基于输入频率  $f_{CTQIN}$  定义特定协议的时序、移位控制信号或位时序。

- 协议相关分频器在引脚上的输出信号 MCLKOUT 和 SCLKOUT。为了满足不同的应用需要，可对这些信号的输出特性进行配置。



**图 21-2 波特率发生器**

### 21.1.6 通道事件和中断

在数据通信和数据处理过程中发生的、需要通知用户的事件包括：

- 数据传送事件，和发送或接收数据字相关、和所选协议无关。
- 特定协议事件，和所选协议有关。
- 数据缓存事件，和通过 FIFO 数据缓存进行的数据处理相关。

### 21.1.7 数据移位和处理

USIC 模块的数据处理由一个独立的数据移位单元 (DSU) 和一个缓存结构组成。数据移位和缓存寄存器的宽度为 16 位 (最大数据字长)，但多个数据字可串联构成更长的数据帧。DSU 的输入包括移位数据 (由 DX0 处理)、移位时钟 (由 DX1 处理) 和移位控制 (由 DX2 处理)。信号 DOUT 代表移位数据输出。

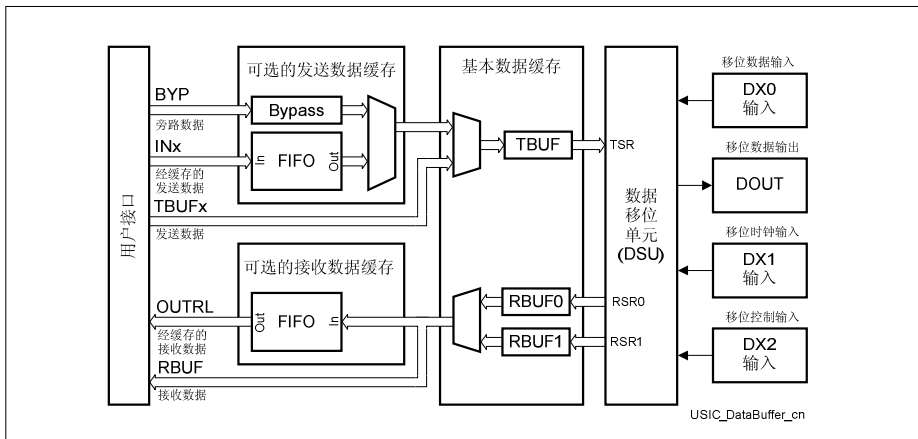


图 21-3 数据缓存机制

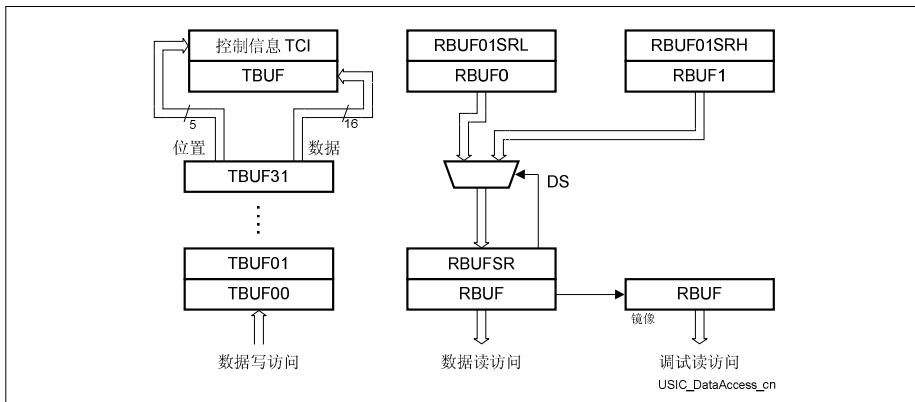
数据处理机制包括：

- 带有一个发送移位寄存器（TSR，位于 DSU 中）和一个发送数据缓存（TBUF）的发送器。数据有效性检验机制使得在某些情况下可通过外部事件触发和控制数据传送。
- 带有两个接收移位寄存器（RSR0 和 RSR1，位于 DSU 中）和两个接收缓存（RBUF0 和 RBUF1）的接收器。交替使用的接收移位寄存器支持接收含有多个数据字的数据流和数据帧。
- 基于先进先出（FIFO）原理的发送和接收数据缓存（可选），该结构不一定在所有器件中均可用（参见[章节 21.7.2](#)）。
- 用于处理数据、中断以及状态和控制信息的用户接口。

#### 21.1.7.1 基本的数据缓存结构

读取接收到的数据和设置将要发送的数据可由一个基本的数据缓存结构处理。

保存在接收缓存 RBUF0/RBUF1 中的数据可被直接读取，此时用户必须谨慎、按照正确顺序读取这些寄存器。为了简化接收缓存结构，引入寄存器 RBUF。读取 RBUF 时，将得到最先接收到的数据字（接收序列中最早的数据）。即使只读取 RBUF 中的数据的低位字节，该数据也会被自动声明不再是新数据、下一个接收到的数据字将出现在 RBUF 中可被读取。



**图 21-4 无附加数据缓存的数据访问结构**

建议用户通过访问 RBUF 读取接收到的数据，避免操作 RBUF0 和 RBUF0。USIC 模块还支持在调试过程中访问接收数据字。调试器进行读操作时不应干扰接收数据序列，因此不应从 RBUF 中读取数据，故引入寄存器 RBUFD。RBUFD 和 RBUF 保存的内容相同，但读取 RBUFD 时不会改变数据的状态（同一数据可被多次读取）。除接收数据之外，每个接收数据字的其它状态信息保存在接收缓存状态寄存器 RBUF01SRL/RBUF01SRH（和 RBUF0 和 RBUF1 中的数据相关）和 RBUFSR（和 RBUF 中的数据相关）中。

可通过将发送数据写入发送缓存输入单元 TBUFx（x = 00-31）的方式软件加载 TBUF，发送缓存输入单元共由 32 个连续地址组成。写入 TBUFx 的数据被保存在发送缓存 TBUF 中。此外，还可通过分析 TBUFx 的地址用于附加控制，该 5 位信息（称为发送控制信息 TCI）在不同协议中的用途不同。

### 21.1.7.2 FIFO 缓存结构

为了实现较简便的数据建立和处理，可选用附加的数据缓存机制。该数据缓存基于先进先出原理（FIFO），从而确保传送数据字的顺序正确。

使用 FIFO 结构和不用 FIFO 结构的数据处理机制（数据及相关控制信息）相似。发送和接收可单独使能/禁用 FIFO 缓存（例如，若某个特定的 USIC 通道使用数据 FIFO 缓存，可配置发送数据通路不带 FIFO 缓存、接收数据通路带有 FIFO 缓存）。

通过 32 个连续的地址单元（使用 Inx 替代 TBUFx（x = 00-31））寻址发送 FIFO 缓存，和 FIFO 深度无关。使用这 32 个地址单元来保存和每个 FIFO 入口相关联的 5 位 TCI（和写入数据）。

可从两个独立的地址单元 OUTR 和 OUTDRL（替代 RBUF 和 RBUFD）读取接收 FIFO 的内容。从 OUTR 读取数据之后，下一个数据包则可用于下一次读取（通用 FIFO

机制)。为了实现无干扰调制（无数据丢失的风险），引入了另一个地址单元（OUTDRL），OUTDRL 和 OUTR 中保存的数据相同，但读取 OUTDRL 不会修改 FIFO 的内容。

发送 FIFO 还具有旁路数据流并将旁路数据加载到 TBUF 的功能。该功能可用于在发送 FIFO 为空的情况下产生高优先级信息或发送紧急信息。FIFO 缓存的发送控制还可利用传送逻辑中的传送触发功能和门控机制进行数据有效性检验（如通过事件触发数据传送）。

注：操作 FIFO 缓存的详细内容参见[章节 21.2.13](#)。

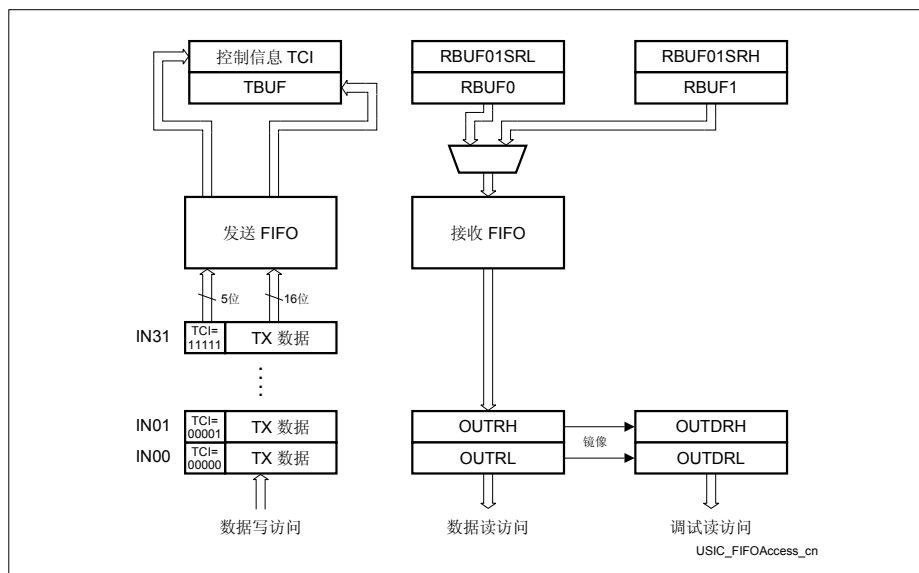


图 21-5 使用 FIFO 的数据访问结构

## 21.2 操作 USIC

本节将说明如何操作 USIC 通信通道，内容包括：

- 寄存器概述（见[页 21-11](#)）
- 一般通道操作（见[页 21-15](#)）
- 通道控制和配置寄存器（见[页 21-22](#)）
- 协议相关寄存器（见[页 21-29](#)）
- 输入级（见[页 21-32](#)）

- 输入级控制寄存器（见页 21-35）
- 波特率产生（见页 21-37）
- 波特率和移位控制寄存器（见页 21-41）
- 操作发送通路（见页 21-45）
- 操作接收通路（见页 21-48）
- 传送控制和状态寄存器（见页 21-50）
- 数据缓存寄存器（见页 21-59）
- 操作 FIFO 数据缓存（见页 21-67）
- FIFO 缓存和旁路寄存器（见页 21-76）

### 21.2.1 寄存器概述

USIC 模块本身为 32 位，有些寄存器已被拆分为两部分、各以 16 位实现。这两部分和原先的 32 位寄存器名称相同，仅添加后缀予以区分：低位部分后缀为 L；高位部分后缀为 H。那些仅用到 16 位的 32 位寄存器仍保持原名称（无后缀），因为只有用到的位会出现在寄存器映射图中。

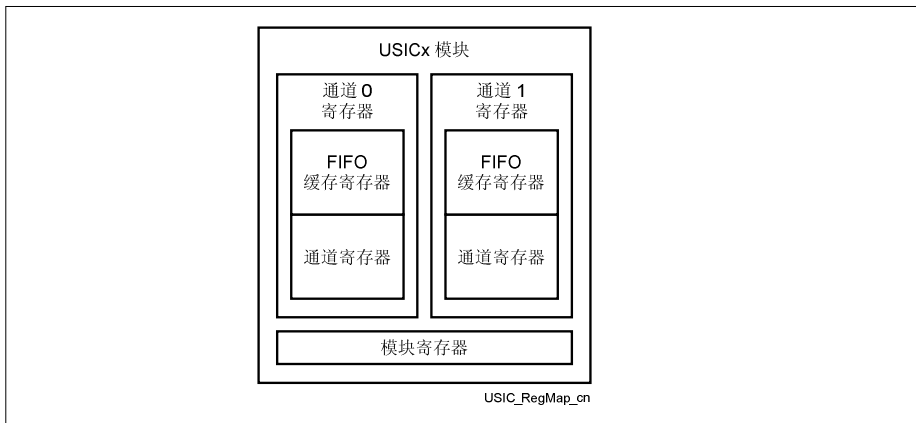
**表 21-3** 给出编程 USIC 通道和 FIFO 缓存所需的所有寄存器。它归纳了 USIC 通信通道寄存器并给出相对地址和复位值。

请注意：可以任意宽度（8 位、16 位）访问所有寄存器，和描述的寄存器宽度无关。不支持短寻址。

所有 USIC 寄存器（位域 KSCFG.SUMCFG 除外）始终通过应用复位来复位。位域 KSCFG.SUMCFG 由调试复位来复位。

*注：当寄存器中的某些位标记为“w”时，读取它们始终返回 0。它们用于修改其寄存器中的触发器或触发内部操作。*

**图 21-6** 示出 USIC 模块寄存器和通道寄存器的寄存器类别。在特定的微控制器中，USIC 模块“x”的模块寄存器使用模块前缀“USICx\_”标记；USIC 模块“x”的通道寄存器使用通道前缀“UxC0\_”和“UxC1\_”标记。



**图 21-6 USIC 模块和通道寄存器**

**表 21-3 USIC 内核相关寄存器和内核寄存器**

寄存器缩写名	寄存器完整名	偏移地址	复位值	描述见
<b>模块寄存器 <sup>1)</sup></b>				
IDL	模块 ID 寄存器 L	008 <sub>H</sub>	C0XX <sub>H</sub>	<a href="#">页 21-175</a>
IDH	模块 ID 寄存器 H	00A <sub>H</sub>	003A <sub>H</sub>	<a href="#">页 21-176</a>
<b>通道寄存器</b>				
FDRL	分数分频寄存器 L	004 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-41</a>
FDRH	分数分频寄存器 H	006 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-42</a>
KSCFG	内核状态配置寄存器	00C <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-26</a>
CCR	通道控制寄存器	010 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-22</a>
INPRL	中断节点指针寄存器 L	014 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-28</a>
INPRH	中断节点指针寄存器 H	016 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-29</a>
CCFG	通道配置寄存器	018 <sub>H</sub>	00CF <sub>H</sub>	<a href="#">页 21-28</a>
BRGL	波特率发生器寄存器 L	01C <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-43</a>
BRGH	波特率发生器寄存器 H	01E <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-44</a>

寄存器缩写名	寄存器完整名	偏移地址	复位值	描述见
DX0CR	输入控制寄存器 0	020 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-35</a>
DX1CR	输入控制寄存器 1	024 <sub>H</sub>	0000 <sub>H</sub>	
DX2CR	输入控制寄存器 2	028 <sub>H</sub>	0000 <sub>H</sub>	
SCTRL	移位控制寄存器 L	030 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-50</a>
SCTRH	移位控制寄存器 H	032 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-51</a>
FMRL	标志修改寄存器 L	038 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-57</a>
FMRH	标志修改寄存器 H	03A <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-58</a>
TCSRL	发送控制/状态寄存器 L	03C <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-52</a>
TCSRH	发送控制/状态寄存器 H	03E <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-55</a>
PCRL	协议控制寄存器 L	040 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-29<sup>2)</sup></a>
				<a href="#">页 21-104<sup>3)</sup></a>
				<a href="#">页 21-127<sup>4)</sup></a>
				<a href="#">页 21-150<sup>5)</sup></a>
				<a href="#">页 21-167<sup>6)</sup></a>
PCRH	协议控制寄存器 H	042 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-30<sup>2)</sup></a>
				<a href="#">页 21-107<sup>3)</sup></a>
				<a href="#">页 21-129<sup>4)</sup></a>
				<a href="#">页 21-150<sup>5)</sup></a>
				<a href="#">页 21-169<sup>6)</sup></a>
PSR	协议状态寄存器	044 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-30<sup>2)</sup></a>
				<a href="#">页 21-108<sup>3)</sup></a>
				<a href="#">页 21-130<sup>4)</sup></a>
				<a href="#">页 21-153<sup>5)</sup></a>
				<a href="#">页 21-170<sup>6)</sup></a>
PSCR	协议状态清零寄存器	048 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-31</a>



寄存器缩写名	寄存器完整名	偏移地址	复位值	描述见
RBUFD	调试专用接收缓存寄存器	04C <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-65</a>
RBUF0	接收缓存寄存器 0	050 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-59</a>
RBUF1	接收缓存寄存器 1	054 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-62</a>
RBUFSR	接收缓存状态寄存器	058 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-66</a>
RBUF	接收缓存寄存器	05C <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-65</a>
RBUF01SRL	接收缓存 01 状态寄存器 L	060 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-60</a>
RBUF01SRH	接收缓存 01 状态寄存器 H	062 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-62</a>
-	保留不用	06C <sub>H</sub>	-	-
-	保留不用	06E <sub>H</sub>	-	-
TBUFx	发送缓存输入单元 x (x = 00-31)	080 <sub>H</sub> + x*4	0000 <sub>H</sub>	<a href="#">页 21-59</a>

### **FIFO 缓存寄存器**

BYP	旁路数据寄存器	100 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-76</a>
BYPCRL	旁路控制寄存器 L	104 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-76</a>
BYPCRH	旁路控制寄存器 H	106 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-77</a>
TRBPTL	发送/接收缓存指针寄存器 L	108 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-92</a>
TRBPTRH	发送/接收缓存指针寄存器 H	10A <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-92</a>
TBCTRL	发送缓存控制寄存器 L	110 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-83</a>
TBCTRH	发送缓存控制寄存器 H	112 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-84</a>
RBCTRL	接收缓存控制寄存器 L	114 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-86</a>
RBCTRH	接收缓存控制寄存器 H	116 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-87</a>
TRBSRL	发送/接收缓存状态寄存器 L	118 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-79</a>
TRBSRH	发送/接收缓存状态寄存器 H	11A <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-81</a>
TRBSCR	发送/接收缓存状态清零 寄存器	11C <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-82</a>
OUTRL	接收缓存输出寄存器 L	120 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-90</a>

寄存器缩写名	寄存器完整名	偏移地址	复位值	描述见
OUTRH	接收缓存输出寄存器 H	122 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-90</a>
OUTDRL	调试专用接收缓存输出寄存器 L	124 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-91</a>
OUTDRH	调试专用接收缓存输出寄存器 H	126 <sub>H</sub>	0000 <sub>H</sub>	<a href="#">页 21-91</a>
INx	发送 FIFO 缓存输入单元 x (x = 00-31)	180 <sub>H</sub> + x*4	0000 <sub>H</sub>	<a href="#">页 21-89</a>

- 1) 模块 ID 寄存器的具体描述参见“模块的实现”一节（见[页 21-175](#)）。
- 2) 本页给出通用的寄存器结构。
- 3) 本页给出 ASC 模式下的寄存器结构。
- 4) 本页给出 SSC 模式下的寄存器结构。
- 5) 本页给出 IIC 模式下的寄存器结构。
- 6) 本页给出 IIS 模式下的寄存器结构。

## 21.2.2 操作 USIC 通信通道

本节说明如何操作 USIC 通信通道，包括协议控制和状态、模式控制和中断处理。具体内容包括：

- 使能 USIC 模块并配置 USIC 在不同器件操作模式下的行为（见[页 21-16](#)）。
- 配置引脚（参见相应协议章节中的描述）。
- 配置数据结构（移位方向、字长、帧长、极性等等）。
- 配置 FIFO 缓存（可选）的数据缓存结构。只有当寄存器 CCFG 中的相应位被置位时才能使能 FIFO 缓存。
- 通过 CCR.MODE 选择协议。只有当寄存器 CCFG 中的相应位被置位时才能进行协议选择。

### 21.2.2.1 协议控制和状态

协议的控制和状态信息分别保存在协议控制寄存器 PCRL/PCRH 和协议状态寄存器 PSR 中。由于这些寄存器被多种协议所共用，因此，寄存器中各位的含义在不同协议中有所不同。

#### PCRL/H 中控制位的使用

在不同协议下寄存器 PCRL/PCRH 中各位的含义不同：

- ASC 协议下的 PCRL/PCRH（见[页 21-104](#)）

- SSC 协议下的 PCRL/PCRH（见[页 21-127](#)）
- IIC 协议下的 PCRL/PCRH（见[页 21-150](#)）
- IIS 协议下的 PCRL/PCRH（见[页 21-167](#)）

### PSR 标志位的使用

在不同协议下寄存器 PSR 中各标志位的含义不同：

- ASC 协议下的 PSR 标志位（见[页 21-108](#)）
- SSC 协议下的 PSR 标志位（见[页 21-130](#)）
- IIC 协议下的 PSR 标志位（见[页 21-153](#)）
- IIS 协议下的 PSR 标志位（见[页 21-170](#)）

#### 21.2.2.2 模式控制

对于不同的系统控制任务（如降低功耗、或挂起请求进行调试等），可通过模式控制设定模块在不同器件操作条件下的行为。通信通道在每种器件操作模式下的行为可编程设定。因此，每路通信通道具有一个与之关联的内核状态配置寄存器 **KSCFG**，用来定义它在各操作模式下的行为：

- 正常操作：  
该模式是既无挂起请求、也无时钟关闭请求时的缺省操作模式。模块时钟不关闭、USIC 寄存器可被读写。通道行为由 **KSCFG.NOMCFG** 定义。
- 挂起模式：  
挂起请求有效时进入该模式。模块时钟不关闭、USIC 寄存器可被读写。通道行为由 **KSCFG.SUMCFG** 定义。
- 时钟关闭模式：  
为了降低系统功耗，请求进入该模式。当 USIC 模块中的所有通道达到其在终止模式下指定的状态时，模块时钟自动关闭。此时 USIC 寄存器不能被访问。通道行为由 **KSCFG.COMCFG** 定义。

USIC 通信通道在每种器件操作模式（正常操作、挂起模式、时钟关闭模式）下的行为可编程设定。因此，USIC 通信通道共有四种内核模式，归纳见[表 21-4](#)。

**表 21-4      USIC 通信通道的行为**

内核模式	通道行为	KSCFG.NOMCFG
运行模式 0	执行所设定的通道操作；不影响数据传送	00 <sub>B</sub>
运行模式 1		01 <sub>B</sub>

内核模式	通道行为	KSCFG.NOMCFG
终止模式 0	终止条件参见协议章节中的描述	10 <sub>B</sub>
终止模式 1		11 <sub>B</sub>

通常，位域 KSCFG.NOMCFG 应配置为运行模式 0，这是正常操作的缺省模式。若通信通道不应响应挂起请求（并继续其操作），必须使位域 KSCFG.SUMCFG 的值和 KSCFG.NOMCFG 相同。当达到某个特定的终止条件时，若通信通道应执行不同的终止操作，必须将 KSCFG.SUMCFG 设置为终止模式 0 或终止模式 1。

在时钟关闭模式下，通过位域 KSCFG.COMCFG 设置所期望的通道操作（和挂起模式的机制相似）。

根据所选择的协议来定义终止条件（参见协议章节中有关模式控制的描述）。

*注：终止模式的选择主要取决于实际的应用需求，在同一种应用中几乎不可能同时请求不同的终止模式。因此，在寄存器 KSCFG 中只应选用一种终止模式类型（0 或 1）。禁止终止模式 0 和终止模式 1 混用，避免在同一路通信通道中出现从终止模式 0 到终止模式 1 的转换（反之亦然）。*

若设置 KSCFG.MODEN = 0 关闭模块时钟，或满足终止条件（终止模式 0 或 1）进入时钟关闭模式，模块不能被读写（寄存器 KSCFG 除外，它始终可被访问）。

### 21.2.2.3 一般的通道事件和中断

一般的事件和中断结构如图 21-7 所示。当满足所设定的条件时，检测到事件发生、事件指示标志位被自动置位。该标志位保持置位状态直至由软件清零。若允许产生中断，检测到某个事件后可产生中断。事件指示标志位的实际状态对是否产生中断无影响。因此，无需对事件指示标志位清零以产生后续中断。

此外，当检测到某事件时，可通过中断节点指针选择激活哪路服务请求输出 SRx。该结构实现了根据应用需求将事件灵活分配给中断：多个事件可共享一个中断服务程序（多个事件激活同一个 SRx 输出），或者每个事件可被单独处理（每个事件激活一路 SRx 输出）。

SRx 输出和中断控制寄存器相连，用于处理 CPU 对服务请求的响应。具体的分配情况见“模块的实现”一节中页 21-176 的描述。

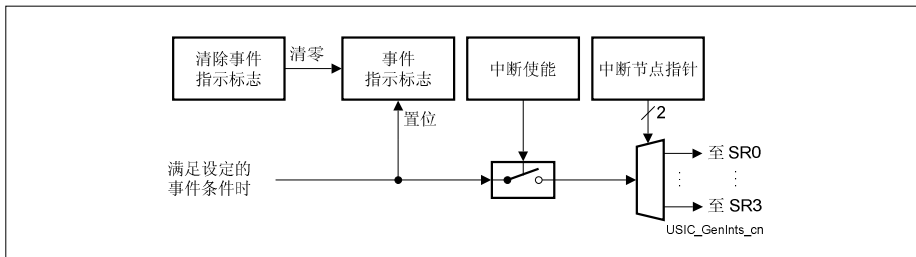


图 21-7 一般的事件和中断结构

#### 21.2.2.4 数据传送事件和中断

发送和接收数据字时产生数据传送事件。相关指示标志位存放在寄存器 PSR 中。每个事件可被单独使能以产生中断。

- 接收事件，指示已接收到一个数据字：

接收缓存 RBUF 中新接收到的数据字可用时，产生接收事件或备选接收事件。

若 RBUF.SR.PERR = 0，产生接收事件，由标志位 PSR.RIF 指示，若使能中断，则产生接收中断。

- 接收开始事件，指示已开始接收一个数据字：

当检测到移入新数据字第一位的接收时钟沿时、若接收被使能，产生接收开始事件，由标志位 PSR.RSIF 指示，若使能中断，则产生接收开始中断。

在全双工模式下，该事件比发送缓存事件滞后半个移位时钟周期产生，它指示何时内部“冻结”移位控制设置用于接收当前的数据字、何时可设定新设置。

在 SSC 和 IIS 模式下，一旦产生接收开始事件，单次模式下的发送数据有效标志 TCSRL.TDV 被清零。

- 备选接收事件，指示已接收到一个特定的数据字：

接收缓存 RBUF 中新接收到的数据字可用时，产生接收事件或备选接收事件。

若 RBUF.SR.PERR = 1，产生备选接收事件，由标志位 PSR.AIF 指示，若使能中断，则产生备选接收中断。

在不同协议中置位 RBUF.SR.PERR 的意义不同：ASC 模式下指示奇偶校验错误；IIC 模式下指示接收到新数据帧中的第一个字节；IIS 模式下指示右/左通道的 WA 信息。SSC 模式下 RBUF.SR.PERR 指示接收到的数据字是否为第一个数据字，是则置位、不是则复位。

- 发送移位事件，指示已发送一个数据字：

在一个数据字的最后一个移位时钟沿上产生发送移位事件，由标志位 PSR.TSIF 指示，若使能中断，则产生发送移位中断。

- 发送缓存事件，指示已开始发送一个数据字：

当发送缓存 TBUF 中的数据字已载入移位寄存器，TBUF 可写入新数据字时，产生发送缓存事件。在移出新数据字第一位的发送时钟沿处（且发送被使能）产生发送缓存事件，由标志位 PSR.TBIF 指示，若使能中断，则产生发送缓存事件。

该事件也用来指示何时内部“冻结”移位控制设置（字长、移位方向等）用于发送当前的数据字。

在 ASC 和 IIC 模式下，一旦产生发送缓存事件，单次模式下的发送数据有效标志 TCSRL.TDV 被清零。

- 数据丢失事件，指示最早接收到的数据字丢失：

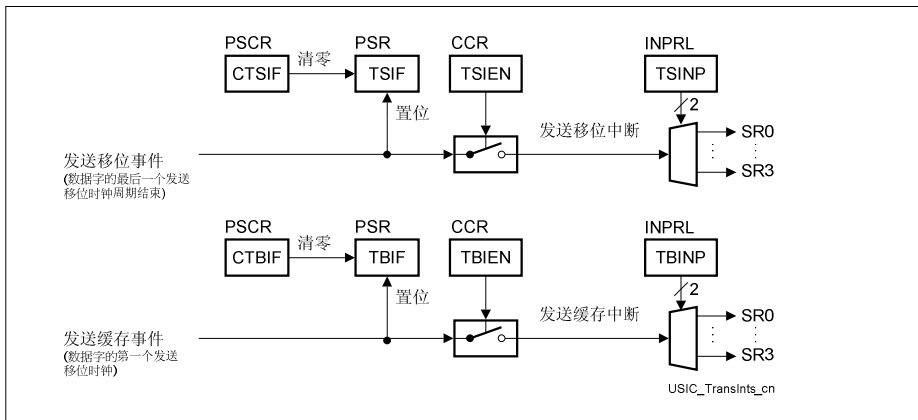
若寄存器 RBUF 中可用的数据字（来自 RBUF0 或 RBUF1 的最早的数据字）还未读出就被新数据覆盖，将产生数据丢失事件，由标志位 PSR.DLIF 指示，若使能中断，则产生数据丢失中断。

用于指示数据传送事件以及控制中断产生的各寄存器、寄存器指示/控制位和位域归纳见表 21-5。

**表 21-5      数据传送事件和中断处理**

事件	指示标志位	标志清零控制位	中断使能控制位	SRx 输出选择控制位
标准接收事件	PSR.RIF	PSCR.CRIF	CCR.RIEN	INPRL.RINP
接收开始事件	PSR.RSIF	PSCR.CRSIF	CCR.RSIEN	INPRL.RSINP
备选接收事件	PSR.AIF	PSCR.CAIF	CCR.AIEN	INPRL.AINP
发送移位事件	PSR.TSIF	PSCR.CTSIF	CCR.TSIEN	INPRL.TSINP
发送缓存事件	PSR.TBIF	PSCR.CTBIF	CCR.TBIEN	INPRL.TBINP
数据丢失事件	PSR.DLIF	PSCR.CDLIF	CCR.DLIEN	INPRH.PINP

两个发送事件和中断如图 21-8 所示。



**图 21-8 发送事件和中断**

接收事件和中断如图 21-9 所示。

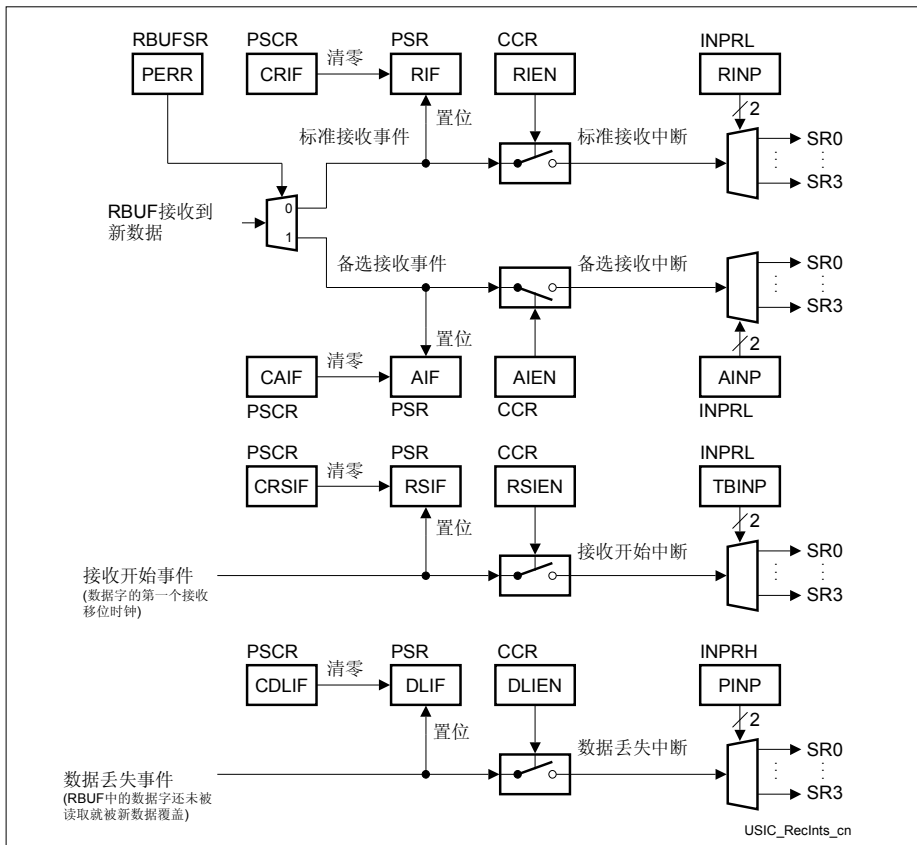


图 21-9 接收事件和中断

### 21.2.2.5 特定协议事件和中断

这些事件和所选协议有关，相关指示标志位存放在寄存器 PSR 中。每个事件可被单独使能以产生协议中断。

- **ASC 模式下的特定协议事件：**

同步分隔符、发送线上的数据冲突、接收噪声、停止位格式错误、接收帧结束、发送帧结束

- **SSC 模式下的特定协议事件：**

MSLS 事件（主控模式下数据帧的开始/结束）、DX2T 事件（从控模式下数据帧的开始/结束），均基于从控选择信号



- IIC 模式下的特定协议事件：  
错误的发送码（帧序列错误）、接收到开始条件、接收到重新开始条件、接收到停止条件、接收到无应答、仲裁丢失、从控读请求、其他一般错误
- IIS 模式下的特定协议事件：  
DX2T 事件（WA 线上有跳变）、检测到 WA 的下降沿或上升沿、生成 WA 结束

**表 21-6      特定协议事件和中断处理**

事件	指示标志位	标志清零控制位	中断使能控制位	SRx 输出选择控制位
ASC 模式下的特定协议事件	PSR.ST[8:2]	PSCR.CST[8:2]	PCRL.CTR[7:3]	INPRH.PINP
SSC 模式下的特定协议事件	PSR.ST[3:2]	PSCR.CST[3:2]	PCRL.CTR[15:14]	INPRH.PINP
IIC 模式下的特定协议事件	PSR.ST[8:1]	PSCR.CST[8:1]	PCRH.CTR[24:18]	INPRH.PINP
IIS 模式下的特定协议事件	PSR.ST[6:3]	PSCR.CST[6:3]	PCRL.CTR[6:4] PCRL.CTR[15]	INPRH.PINP

## 21.2.3 通道控制和配置寄存器

### 21.2.3.1 通道控制寄存器

通道控制寄存器用来使能/禁止基于通道事件产生中断、控制奇偶检验产生以及选择 USIC 通道采用的协议。

#### CCR

##### 通道控制寄存器

(10<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AI EN	RI EN	TBI EN	TSI EN	DLI EN	RSI EN	PM		0				MODE			
rW	rW	rW	rW	rW	rW	rW		r				rW			

符号	位序号	类型	功能描述
<b>MODE</b>	[3:0]	rw	<p><b>操作模式</b></p> <p>该位域用于选择 USIC 通道采用的协议。若选择不可用的协议（见寄存器 CCFG）或保留的位组合，则该通道关闭。在两种协议之间切换时，选用新协议之前必须先关闭该通道。在这种情况下，必须通过软件清除或更新寄存器 PCRH、PCRL 和 PSR。</p> <p>0<sub>H</sub> 关闭 USIC 通道。所有协议相关状态机设置成空闲状态。</p> <p>1<sub>H</sub> 选择 SSC（SPI）协议。</p> <p>2<sub>H</sub> 选择 ASC（SCI、UART）协议。</p> <p>3<sub>H</sub> 选择 IIS 协议。</p> <p>4<sub>H</sub> 选择 IIC 协议。</p> <p>其它位组合保留不用。</p>
<b>PM</b>	[9:8]	rw	<p><b>奇偶校验模式</b></p> <p>该位域定义采样输入值的奇偶校验产生。</p> <p>00<sub>B</sub> 禁止奇偶校验。</p> <p>01<sub>B</sub> 保留不用。</p> <p>10<sub>B</sub> 选择偶校验（数据位为 1 的个数为奇数时，奇偶校验位 = 1；数据位为 1 的个数为偶数时，奇偶校验位 = 0）</p> <p>11<sub>B</sub> 选择奇校验（数据位为 1 的个数为奇数时，奇偶校验位 = 0；数据位为 1 的个数为偶数时，奇偶校验位 = 1）</p>
<b>RSIEN</b>	10	rw	<p><b>接收开始中断使能</b></p> <p>发生接收开始事件时，该位使能产生中断。</p> <p>0<sub>B</sub> 禁止产生接收开始中断。</p> <p>1<sub>B</sub> 允许产生接收开始中断。若产生接收开始事件，由 INPRL.TBINP 选择的服务请求输出 SRx 被激活。</p>
<b>DLIEN</b>	11	rw	<p><b>数据丢失中断使能</b></p> <p>发生数据丢失事件时，该位使能产生中断（当 RDVx = 1 时将数据接收到 RBUFx 中）。</p>

符号	位序号	类型	功能描述
			<p>0<sub>B</sub> 禁止产生数据丢失中断。</p> <p>1<sub>B</sub> 允许产生数据丢失中断。若产生数据丢失事件，由 INPRH.PINP 选择的服务请求输出 SRx 被激活。</p>
<b>TSIEN</b>	12	rw	<p><b>发送移位中断使能</b></p> <p>发生发送移位事件时，该位使能产生中断。</p> <p>0<sub>B</sub> 禁止产生发送移位中断。</p> <p>1<sub>B</sub> 允许产生发送移位中断。若产生发送移位事件，由 INPRL.TSINP 选择的服务请求输出 SRx 被激活。</p>
<b>TBIEN</b>	13	rw	<p><b>发送缓存中断使能</b></p> <p>发生发送缓存事件时，该位使能产生中断。</p> <p>0<sub>B</sub> 禁止产生发送缓存中断。</p> <p>1<sub>B</sub> 允许产生发送缓存中断。若产生发送缓存事件，由 INPRL.TBINP 选择的服务请求输出 SRx 被激活。</p>
<b>RIEN</b>	14	rw	<p><b>接收中断使能</b></p> <p>发生接收事件时，该位使能产生中断。</p> <p>0<sub>B</sub> 禁止产生接收中断。</p> <p>1<sub>B</sub> 允许产生接收中断。若产生接收事件，由 INPRL.RINP 选择的服务请求输出 SRx 被激活。</p>
<b>AIEN</b>	15	rw	<p><b>备选接收中断使能</b></p> <p>发生备选接收事件时，该位使能产生中断。</p> <p>0<sub>B</sub> 禁止产生备选接收中断。</p> <p>1<sub>B</sub> 允许产生备选接收中断。若产生备选接收事件，由 INPRL.AINP 选择的服务请求输出 SRx 被激活。</p>
<b>0</b>	[7:4]	r	<p><b>保留</b></p> <p>读操作返回 0；应写入 0。</p>

### 21.2.3.2 通道配置寄存器

通道配置寄存器指示 USIC 通道中可用的协议。

#### CCFG

#### 通道配置寄存器

(18<sub>H</sub>)

复位值: 00CF<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								TB	RB	0	IIS	IIC	ASC	SSC	
r								r	r	r	r	r	r	r	r

符号	位序号	类型	功能描述
<b>SSC</b>	0	r	<b>SSC 协议可用</b> 该位指示 SSC 协议是否可用。 0 <sub>B</sub> SSC 协议不可用。 1 <sub>B</sub> SSC 协议可用。
<b>ASC</b>	1	r	<b>ASC 协议可用</b> 该位指示 ASC 协议是否可用。 0 <sub>B</sub> ASC 协议不可用。 1 <sub>B</sub> ASC 协议可用。
<b>IIC</b>	2	r	<b>IIC 协议可用</b> 该位指示 IIC 协议是否可用。 0 <sub>B</sub> IIC 协议不可用。 1 <sub>B</sub> IIC 协议可用。
<b>IIS</b>	3	r	<b>IIS 协议可用</b> 该位指示 IIS 协议是否可用。 0 <sub>B</sub> IIS 协议不可用。 1 <sub>B</sub> IIS 协议可用。
<b>RB</b>	6	r	<b>接收 FIFO 缓存可用</b> 该位指示附加的接收 FIFO 缓存是否可用。 0 <sub>B</sub> 接收 FIFO 缓存不可用。 1 <sub>B</sub> 接收 FIFO 缓存可用。

符号	位序号	类型	功能描述
<b>TB</b>	7	r	<b>发送 FIFO 缓存可用</b> 该位指示附加的发送 FIFO 缓存是否可用。 0 <sub>B</sub> 发送 FIFO 缓存不可用。 1 <sub>B</sub> 发送 FIFO 缓存可用。
<b>0</b>	[5:4], [15:8]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 21.2.3.3 内核状态配置寄存器

可通过内核状态配置寄存器 KSCFG 选择不同器件操作模式下所期望的内核模式。

#### KSCFG

内核状态配置寄存器

(0C<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>BP COM</b>	0	<b>COMCFG</b>	<b>BP SUM</b>	0	<b>SUMCFG</b>	<b>BP NOM</b>	0	<b>NOMCFG</b>	0	0	<b>BP MOD EN</b>	<b>MOD EN</b>			
w	r	rw	w	r	rw	w	r	rw	r	r	w	w			

符号	位序号	类型	功能描述
<b>MODEN</b>	0	rw	<b>模块使能</b> 该位使能模块内核时钟和模块功能。 0 <sub>B</sub> 立即关闭模块（不考虑终止条件）。模块对控制操作不予响应、模块时钟关闭。模块不响应读操作，忽略写操作（KSCFG 除外）。 1 <sub>B</sub> 开启模块正常工作。MODEN 置 1 后，建议在访问其它 USIC 寄存器之前先读取 KSCFG，从而避免流水线对控制模块造成的影响。
<b>BPMODEN</b>	1	w	<b>MODEN 的位保护</b> 该位使能对 MODEN 进行写操作。读取始终为 0。 0 <sub>B</sub> 不改变 MODEN。

符号	位序号	类型	功能描述
			1 <sub>B</sub> 写操作更新 MODEN。
<b>NOMCFG</b>	[5:4]	rw	<b>正常操作模式配置</b> 该位域选择正常操作模式下所用的内核模式。 00 <sub>B</sub> 选择运行模式 0。 01 <sub>B</sub> 选择运行模式 1。 10 <sub>B</sub> 选择终止模式 0。 11 <sub>B</sub> 选择终止模式 0。
<b>BPNOM</b>	7	w	<b>NOMCFG 的位保护</b> 该位使能对 NOMCFG 进行写操作。读取始终为 0。 0 <sub>B</sub> 不改变 NOMCFG。 1 <sub>B</sub> 写操作更新 NOMCFG。
<b>SUMCFG</b>	[9:8]	rw	<b>挂起模式配置</b> 该位域选择挂起模式下所用的内核模式。 SUMCFG 和 NOMCFG 的编码相同。
<b>BPSUM</b>	11	w	<b>SUMCFG 的位保护</b> 该位使能对 SUMCFG 进行写操作。读取始终为 0。 0 <sub>B</sub> 不改变 SUMCFG。 1 <sub>B</sub> 写操作更新 SUMCFG。
<b>COMCFG</b>	[13:12]	rw	<b>时钟关闭模式配置</b> 该位域选择时钟关闭模式下所用的内核模式。 COMCFG 和 NOMCFG 的编码相同。
<b>BPCOM</b>	15	w	<b>COMCFG 的位保护</b> 该位使能对 COMCFG 进行写操作。读取始终为 0。 0 <sub>B</sub> 不改变 COMCFG。 1 <sub>B</sub> 写操作更新 COMCFG。
<b>0</b>	[3:2], 6,10, 14	r	<b>保留</b> 读操作返回 0；应写入 0。

### 21.2.3.4 中断节点指针寄存器

中断节点指针寄存器用来定义产生某事件中断时激活哪路服务请求输出 SRx。

#### INPRL

**中断节点指针寄存器 L (14<sub>H</sub>)** **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>AINP</b>	<b>0</b>	<b>RINP</b>	<b>0</b>	<b>TBINP</b>	<b>0</b>	<b>TSINP</b>								
r	rw	r	rw	r	rw	r	rw	r	rw	r	rw	r	rw	r	rw

符号	位序号	类型	功能描述
<b>TSINP</b>	[1:0]	rw	<b>发送移位中断节点指针</b> 该位域定义在产生发送移位中断时激活哪路服务请求输出 SRx。 00 <sub>B</sub> 激活输出 SR0 01 <sub>B</sub> 激活输出 SR1 10 <sub>B</sub> 激活输出 SR2 11 <sub>B</sub> 激活输出 SR3
<b>TBINP</b>	[5:4]	rw	<b>发送缓存中断节点指针</b> 该位域定义在产生发送缓存中断或接收开始中断时激活哪路服务请求输出 SRx。TBINP 和 TSINP 的编码相同。
<b>RINP</b>	[9:8]	rw	<b>接收中断节点指针</b> 该位域定义在产生接收中断时激活哪路服务请求输出 SRx。RINP 和 TSINP 的编码相同。
<b>AINP</b>	[13:12]	rw	<b>备选接收中断节点指针</b> 该位域定义在产生备选接收中断时激活哪路服务请求输出 SRx。AINP 和 TSINP 的编码相同。
<b>0</b>	[3:2], [7:6], [11:10] [15:14]	r	<b>保留</b> 读操作返回 0；应写入 0。

## INPRH

**中断节点指针寄存器 H (16<sub>H</sub>)** **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0														<b>PINP</b>	
r														rw	

符号	位序号	类型	功能描述
<b>PINP</b>	[1:0]	rw	<b>协议中断节点指针</b> 该位域定义在产生协议中断时激活哪路服务请求输出 SR <sub>x</sub> 。 00 <sub>B</sub> 激活输出 SR0 01 <sub>B</sub> 激活输出 SR1 10 <sub>B</sub> 激活输出 SR2 11 <sub>B</sub> 激活输出 SR3
<b>0</b>	[15:2]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 21.2.4 协议相关寄存器

### 21.2.4.1 协议控制寄存器

协议控制寄存器用来定义特定协议的功能。在采用一种新协议之前必须先对它们进行软件配置。只有所选协议用到的位才有效，其它位读取始终返回 0。

## PCRL

**协议控制寄存器 L (40<sub>H</sub>)** **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CTR 15</b>	<b>CTR 14</b>	<b>CTR 13</b>	<b>CTR 12</b>	<b>CTR 11</b>	<b>CTR 10</b>	<b>CTR 9</b>	<b>CTR 8</b>	<b>CTR 7</b>	<b>CTR 6</b>	<b>CTR 5</b>	<b>CTR 4</b>	<b>CTR 3</b>	<b>CTR 2</b>	<b>CTR 1</b>	<b>CTR 0</b>
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
<b>CTR<sub>x</sub></b> ( <b>x = 0-15</b> )	x	rw	<b>协议控制位 x</b> 该位是协议控制位。



## PCR<sub>H</sub>

协议控制寄存器 H

(42<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
<b>CTR<sub>x</sub></b> ( <b>x = 16-30</b> )	x - 16	rwh	<b>协议控制位 x</b> 该位是协议控制位，可由特定协议的信息覆盖。
<b>CTR31</b>	15	rwh	<b>协议控制位 31</b> 在不同的协议中，该位控制 MCLK 信号的开始和停止。 <b>0<sub>B</sub></b> 不产生信号 MCLK (MCLK = 0)。 <b>1<sub>B</sub></b> 允许产生信号 MCLK。

### 21.2.4.2 协议状态寄存器

可通过置位寄存器 PS<sub>CR</sub> 中的清零控制位相应清除协议状态寄存器中的标志位。向 PS<sub>R</sub> 写 1 时，相应的标志位置位，但不会引发其它操作（不产生中断），写 0 不起作用。在采用新协议之前应该先用软件清除这些标志位。

## PS<sub>R</sub>

协议状态寄存器

(44<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
<b>ST<sub>x</sub></b> ( <b>x = 0-9</b> )	x	rwh	<b>协议状态标志位 x</b> 参见特定协议的描述。
<b>RSIF</b>	10	rwh	<b>接收开始指示标志位</b>

符号	位序号	类型	功能描述
			0 <sub>B</sub> 未发生接收开始事件 1 <sub>B</sub> 已发生接收开始事件
<b>DLIF</b>	11	rwh	<b>数据丢失指示标志位</b> 0 <sub>B</sub> 未发生数据丢失事件 1 <sub>B</sub> 已发生数据丢失事件
<b>TSIF</b>	12	rwh	<b>发送移位指示标志位</b> 0 <sub>B</sub> 未发生发送移位事件 1 <sub>B</sub> 已发生发送移位事件
<b>TBIF</b>	13	rwh	<b>发送缓存指示标志位</b> 0 <sub>B</sub> 未发生发送缓存事件 1 <sub>B</sub> 已发生发送缓存事件
<b>RIF</b>	14	rwh	<b>接收指示标志位</b> 0 <sub>B</sub> 未发生接收事件 1 <sub>B</sub> 已发生接收事件
<b>AIF</b>	15	rwh	<b>备选接收指示标志位</b> 0 <sub>B</sub> 未发生备选接收事件 1 <sub>B</sub> 已发生备选接收事件

### 21.2.4.3 协议状态清零寄存器

读取该寄存器中的所有位均返回 0。

### PSCR

协议状态清零寄存器

(48<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>C AIF</b>	<b>C RIF</b>	<b>C TBIF</b>	<b>C TSIF</b>	<b>C DLIF</b>	<b>C RSIF</b>	<b>C ST9</b>	<b>C ST8</b>	<b>C ST7</b>	<b>C ST6</b>	<b>C ST5</b>	<b>C ST4</b>	<b>C ST3</b>	<b>C ST2</b>	<b>C ST1</b>	<b>C ST0</b>
W	W	W	W	W	W	W	W	W	W	W	W	W	W	W	W

符号	位序号	类型	功能描述
----	-----	----	------

符号	位序号	类型	功能描述
<b>CSTx</b> (x = 0-9)	x	w	清除 <b>PSR</b> 中的状态标志位 <b>x</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志位 <b>PSR.STx</b>
<b>CRSIF</b>	10	w	清除接收开始指示标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志位 <b>PSR.RSIF</b>
<b>CDLIF</b>	11	w	清除数据丢失指示标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志位 <b>PSR.DLIF</b>
<b>CTSIF</b>	12	w	清除发送移位指示标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志位 <b>PSR.TSIF</b>
<b>CTBIF</b>	13	w	清除发送缓存指示标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志位 <b>PSR.TBIF</b>
<b>CRIF</b>	14	W	清除接收指示标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志位 <b>PSR.RIF</b>
<b>CAIF</b>	15	w	清除备选接收指示标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 清除标志位 <b>PSR.AIF</b>

## 21.2.5 操作输入级

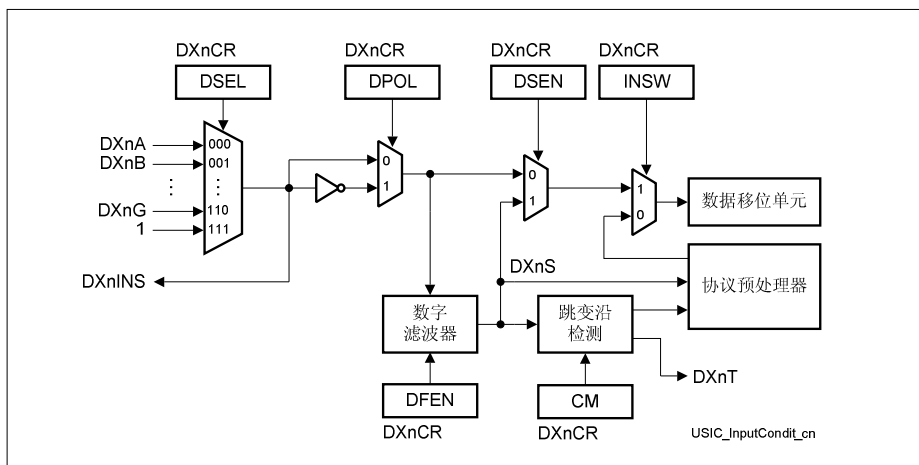
三个输入级具有相同的特性。由于可以非常灵活的进行信号调整、允许独立开启或关闭数字滤波器，因此这些输入级适用于所有协议。

### 21.2.5.1 一般的输入结构

所有输入级的结构相似，如**图 21-10** 所示。通过寄存器 **DX0CR**、**DX1CR** 和 **DX2CR** 分别控制每个输入级的功能。

可以从输入线  $DXnA - DXnG$  以及一个恒 1 电平中选择所需的输入信号，通过位域 **DSEL** 编程设定。关于特定器件的输入信号分配，请参见“模块的实现”一节。可通过位 **DPOL** 翻转所选输入信号的极性，从而使其与数据移位单元和协议状态机的内部极性相匹配。在某些协议中，输入信号可直接送至数据移位单元进行数据传送（**DSEN** = 0, **INSW** = 1），无需进行进一步的信号调整。在这种情况下，由于不进行同步或滤波，数据通路中不含任何延迟。

若输入信号有噪声，可在数据通路上进行信号同步（信号  $DXnS$  与  $f_{sys}$  同步）并启用数字滤波器。若设置 **DSEN** = 1，则对已同步的输入信号（若 **DFEN** = 1，信号还会被滤波）进行处理。需要注意：同步操作会使数据通路上附加一个长度为 2-3 倍于  $f_{sys}$  时钟周期的延迟。



**图 21-10 输入调整**

输入信号若需经过协议预处理器处理，设置 **INSW** = 0 使数据移位单元和协议预处理器直接相连。协议预处理器和已同步的输入信号  $DXnS$  相连，根据协议的需要，协议预处理器还会评估信号的跳变沿。

### 21.2.5.2 数字滤波器

可使用数字滤波器降低输入信号的噪声。输入信号在滤波之前已经与  $f_{sys}$  同步。若禁用滤波器，信号  $DXnS$  代表已同步的输入信号。若使用滤波器，信号  $DXnS$  上小于一个滤波采样周期的脉冲将被过滤掉。同步输入信号发生跳变之后，若已连续两次检测到新采样值，信号  $DXnS$  变为新值。

用户可编程滤波采样周期以满足不同应用的需要。第一种可能是系统频率  $f_{\text{SYS}}$ 。若选用分数分频输出频率  $f_{\text{FD}}$ ，则可过滤掉更长的脉冲。该频率的可编程范围很宽，它还可用来决定数据传送的波特率。

在所选输入和  $\text{DXnS}$  信号之间，除了一个长度为 2-3 倍于时钟周期的延迟之外，滤波器还会附加一个最长两个滤波采样周期的延迟。

### **21.2.5.3 跳变沿检测**

已被同步（和滤波）的信号  $\text{DXnS}$  可用作数据移位单元的输入，也可用作协议预处理器的输入。若协议预处理器不使用  $\text{DXnS}$  信号， $\text{DXnS}$  可用于其它任务，比如在主控模式下控制数据发送（数据字可标记为有效，具体参见“数据缓存”一节的描述）。

可编程跳变沿检测逻辑通过激活触发信号  $\text{DXnT}$  指示期望事件已发生（在响应该事件之前引入一个  $f_{\text{SYS}}$  时钟周期的延迟）。

### **21.2.5.4 所选输入信号的监控**

每个输入级所选用的输入信号由  $\text{DX0INS}$ 、 $\text{DX1INS}$  和  $\text{DX2INS}$  表示。可用这些信号在系统内触发其它操作（比如产生中断）。

### **21.2.5.5 回环模式**

在回环模式下，发送输出信号可以和同一通信通道中的接收输入信号相连。此时必须选择输入级的输入“G”。该模式下，无需将信号连接至端口引脚、在片内即可对 **ASC**、**SSC** 和 **IIS** 的驱动进行评估。接收器可接收同一通道中发送器传送的数据，如同接收另一个通信方发送的数据。

## 21.2.6 输入级寄存器

### 21.2.6.1 输入控制寄存器

输入控制寄存器用来定义输入级的特性（输入级 DX0 由寄存器 DX0CR 控制，以此类推）。

#### **DX0CR**

**输入控制寄存器 0** (20<sub>H</sub>) 复位值: 0000<sub>H</sub>

#### **DX1CR**

**输入控制寄存器 1** (24<sub>H</sub>) 复位值: 0000<sub>H</sub>

#### **DX2CR**

**输入控制寄存器 2** (28<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DXS</b>	<b>0</b>	<b>CM</b>	<b>SF SEL</b>	<b>D POL</b>	<b>0</b>	<b>DS EN</b>	<b>DF EN</b>	<b>IN SW</b>	<b>0</b>	<b>DSEL</b>					
rw	r	rw	rw	rw	r	rw	rw	rw	r	rw					

符号	位序号	类型	功能描述
<b>DSEL</b>	[2:0]	rw	<b>输入信号的数据选择</b> 该位域用于选择输入数据，有多路输入 DXn[G:A] 可供选择。  000 <sub>B</sub> 选择数据输入 DXnA。 001 <sub>B</sub> 选择数据输入 DXnB。 010 <sub>B</sub> 选择数据输入 DXnC。 011 <sub>B</sub> 选择数据输入 DXnD。 100 <sub>B</sub> 选择数据输入 DXnE。 101 <sub>B</sub> 选择数据输入 DXnF。 110 <sub>B</sub> 选择数据输入 DXnG。 111 <sub>B</sub> 数据输入恒为 1。
<b>INSW</b>	4	rw	<b>输入切换</b> 该位域选择是从输入数据通路 DXn 还是从协议预处理器获取数据移位单元的输入。  0 <sub>B</sub> 数据移位单元的输入由协议预处理器控制。

符号	位序号	类型	功能描述
			<p><b>1<sub>B</sub></b> 数据移位单元的输入和数据输入线相连。如果输入信号直接来自输入引脚、无需协议预处理器处理，则选用该设置。</p>
<b>DFEN</b>	5	rw	<p><b>数字滤波使能</b></p> <p>该位使能/禁止对信号 <b>DXnS</b> 进行数字滤波。</p> <p><b>0<sub>B</sub></b> 输入信号不被数字滤波。</p> <p><b>1<sub>B</sub></b> 输入信号被数字滤波。</p>
<b>DSEN</b>	6	rw	<p><b>数字同步使能</b></p> <p>该位选择是由异步输入信号还是由同步（和滤波）信号 <b>DXnS</b> 作为数据移位单元的输入。</p> <p><b>0<sub>B</sub></b> 未同步信号可用作数据移位单元的输入。</p> <p><b>1<sub>B</sub></b> 同步信号可用作数据移位单元的输入。</p>
<b>DPOL</b>	8	rw	<p><b>DXn 的信号极性</b></p> <p>该位定义输入信号的信号极性。</p> <p><b>0<sub>B</sub></b> 输入信号不被翻转。</p> <p><b>1<sub>B</sub></b> 输入信号被翻转。</p>
<b>SFSEL</b>	9	rw	<p><b>采样频率选择</b></p> <p>该位定义对同步信号 <b>DXnS</b> 数字滤波的采样频率。</p> <p><b>0<sub>B</sub></b> 采样频率为 <math>f_{sys}</math>。</p> <p><b>1<sub>B</sub></b> 采样频率为 <math>f_{FD}</math>。</p>
<b>CM</b>	[11:10]	rw	<p><b>组合模式</b></p> <p>该位域选择由同步信号 <b>DXnS</b> 的上升沿还是下降沿激活输入级的触发输出 <b>DXnT</b>。</p> <p><b>00<sub>B</sub></b> 不激活 <b>DXnT</b>。</p> <p><b>01<sub>B</sub></b> 上升沿激活 <b>DXnT</b>。</p> <p><b>10<sub>B</sub></b> 下降沿激活 <b>DXnT</b>。</p> <p><b>11<sub>B</sub></b> 上下双沿激活 <b>DXnT</b>。</p>
<b>DXS</b>	15	rh	<p><b>同步信号的数值</b></p> <p>该位指示同步输入信号 <b>DXnS</b> 的当前值。</p> <p><b>0<sub>B</sub></b> <b>DXnS</b> 的当前值为 0。</p>

符号	位序号	类型	功能描述
			1 <sub>B</sub> DXnS 的当前值为 1。
<b>0</b>	3, 7, [14:12]	r	保留 读操作返回 0；应写入 0。

## 21.2.7 操作波特率发生器

可通过配置以下单元来操作波特率发生器，参见 [图 21-2](#)。

### 21.2.7.1 分数分频器

分数分频器对输入频率  $f_{SYS}$  进行  $n$  分频、或  $n/1024$  倍频后产生输出频率  $f_{FD}$ 。分数分频器有两种操作模式：

- 正常分频模式（ $FDRL.DM = 01_B$ ）：

该模式下，输出频率  $f_{FD}$  来自输入频率  $f_{SYS}$   $n$  分频， $n$  可取 1-1024 之间的任意整数。分频操作通过计数器  $FDRH.RESULT$  对  $f_{SYS}$  递增计数来实现。计数到  $3FF_H$  后，计数器被  $FDRL.STEP$  重载，随后从重载值继续计数。为了使  $f_{FD} = f_{SYS}$ ， $STEP$  必须设定为  $3FF_H$ 。

正常分频模式下输出频率的计算公式如下：

$$f_{FD} = f_{SYS} \times \frac{1}{n}, \text{ 其中 } n = 1024 - STEP \quad (21.1)$$

- 分数分频模式（ $FDRL.DM = 10_B$ ）：

该模式下，输出频率  $f_{FD}$  来自输入频率  $f_{SYS}$   $n/1024$  倍频， $n$  可取 1-1024 之间的任意整数。通常，分数分频模式可产生比正常分频模式精度更高的平均输出频率。需要注意：分数分频模式下， $f_{FD}$  的最大抖动为一个  $f_{SYS}$  时钟周期，该抖动不会跨周期累积。

在分数分频模式下，由  $FDRL.RESULT$  和  $FDRH.STEP$  的相加结果决定输出频率  $f_{FD}$ ；如果相加结果大于  $3FF_H$  导致溢出，产生一个  $f_{FD}$  脉冲。

分数分频模式下输出频率的计算公式如下：

$$f_{FD} = f_{SYS} \times \frac{n}{1024}, \text{ 其中 } n = STEP \quad (21.2)$$

设置  $BRGL.CLKSEL = 00_B$  选择由分数分频器的输出频率  $f_{FD}$  产生波特率。



### 21.2.7.2 外部频率输入

若所选协议不需输入级 DX1 (DX1CR.INSW = 0)，则可由外部频率输入（代替 f<sub>sys</sub>）产生波特率。在这种情况下，DX1 的外部频率输入信号可由系统频率 f<sub>sys</sub> 同步和采样，还可选择由数字滤波器滤波。该特性使得可用器件自身无法产生的频率（比如特定的音频）进行数据传送。

若 BRGL.CLKSEL = 10<sub>B</sub>，由触发信号 DX1T 决定 f<sub>DX1</sub>。该模式下，根据位域 DX1CR.CM 对 DX1T 触发事件的配置，可由输入信号的上升沿、下降沿或上下双沿产生波特率。每次产生 DX1T 触发事件时，信号 MCLK 翻转。

若 BRGL.CLKSEL = 11<sub>B</sub>，可由输入信号的上升沿产生波特率。信号 MCLK 代表已被同步的输入信号 DX1S。

以上两种模式下，外部输入信号的高电平和低电平时间都必须至少为 f<sub>sys</sub> 的 2 倍。

### 21.2.7.3 分频模式下的协议相关计数器

分频模式下，协议相关计数器用于整数倍分频以产生输出频率 f<sub>PDIV</sub>。此外，两个固定 2 分频的分频器用于产生占空比为 50% 的输出信号 MCLK 和 SCLK。若使用分数分频模式，最大抖动为一个 f<sub>sys</sub> 时钟周期。该分频器的输出频率由寄存器 BRGL 和 BRGH 控制。

为了定义主控时钟 MCLK 和移位时钟 SCLK 的频率比例，用于产生 MCLK 的分频器位于 PDIV+1 分频器的前级，用于产生 SCLK 的分频器位于 PDIV+1 分频器的后级。

$$f_{MCLK} = \frac{f_{PIN}}{2} \quad (21.3)$$

$$f_{SCLK} = \frac{f_{PDIV}}{2} \quad (21.4)$$

若使用主控时钟作为外部器件的参考时钟（如对于 IIS 协议），并要求 MCLK 和 SCLK 及其它时序信号的相位关系固定，建议使用 MCLK 作为 PDIV 分频器的输入。若不使用 MCLK 信号或不必和其它信号有固定的相位关系，可选择更快的频率 f<sub>PIN</sub> 作为输入频率。

$$\begin{aligned} f_{PDIV} &= f_{PIN} \times \frac{1}{PDIV+1} & \text{若 } PPPEN = 0 \\ f_{PDIV} &= f_{MCLK} \times \frac{1}{PDIV+1} & \text{若 } PPPEN = 1 \end{aligned} \quad (21.5)$$

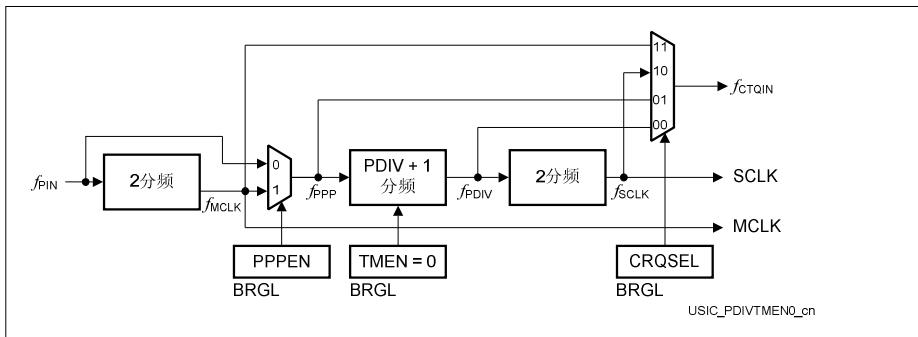


图 21-11 协议相关计数器（分频模式）

#### 21.2.7.4 捕获模式下的协议相关计数器

捕获模式下，协议相关计数器可用来测量时间间隔（BRGL.TMEN = 1）。在该模式下，禁止分频操作（不可能接收和发送），计数器用作对  $f_{PP}$  进行计数的捕获定时器。计数到最大值后停止计数；若 DX0T 或 DX1T 被激活，实际计数值被捕获到位域 BRGH.PDIV 中，计数器清零后重新计数。此外，产生发送移位中断事件（位 PSRL.TSIF 置位）。

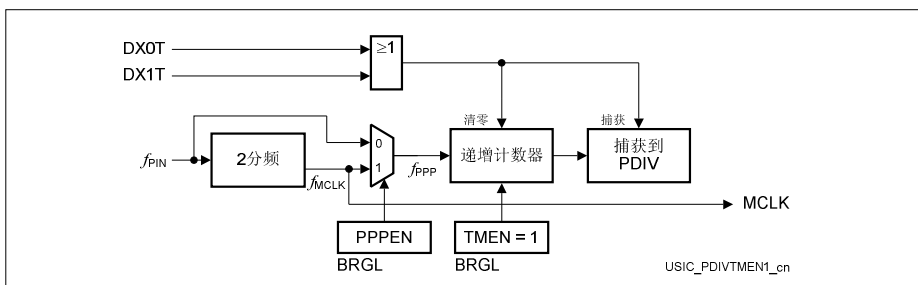
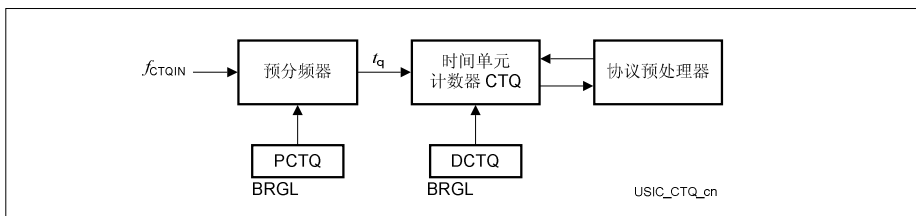


图 21-12 协议相关计数器（捕获模式）

在从模式下，开始传送数据之前可通过捕获模式来测量波特率，比如测量数据信号（DX0T）或移位时钟信号（DX1T）两个跳变沿之间的时间间隔。可在每个输入级中配置激活 DXnT 触发信号的条件。

### 21.2.7.5 时间单元计数器

和协议预处理器关联的时间单元计数器 CTQ 可用来产生特定协议所需的时间间隔。时间单元  $t_q$  的长度由所选择的输入频率  $f_{CTQIN}$  和预分频值决定。时间单元的含义和所选协议有关，具体请参见各协议章节的描述。



**图 21-13 时间单元计数器**

### 21.2.7.6 移位时钟输出配置

相应输出引脚上的主控时钟输出信号 **MCLKOUT** 极性可配置。为了提供比移位时钟频率更高的时间基准，可产生 **MCLK** 信号供每种协议使用。

主控时钟输出信号 **MCLKOUT** 的配置机制可确保不会出现脉冲缩短的情况。每个 **MCLK** 信号周期由主动和被动两个阶段组成（主动在前、被动在后）。

**MCLKOUT** 信号在主动与被动阶段的极性都由 **BRGH.MCLKCFG** 电平决定。被动阶段极性取值于其正相电平，而主动阶段极性取值于其反相电平。若 **BRGH.MCLKOUT** 编程为新值，该值将在顺次的阶段变化时生效。该机制可确保在 **MCLKOUT** 输出上不会出现小于一个阶段长度的脉冲。在图 21-14 的示例中，在 **MCLK** 周期 2 的被动阶段 **BRGH.MCLKOUT** 的值从 0 变为 1。

根据 **PCR.H.MCLK** 的设置，由协议预处理器使能/禁止产生 **MCLKOUT** 信号。若该位置位，在 **MCLK** 信号的下一次主动阶段产生信号 **MCLKOUT**。若该位置 0（禁止产生 **MCLKOUT**），被动阶段的电平也应用于主动阶段。

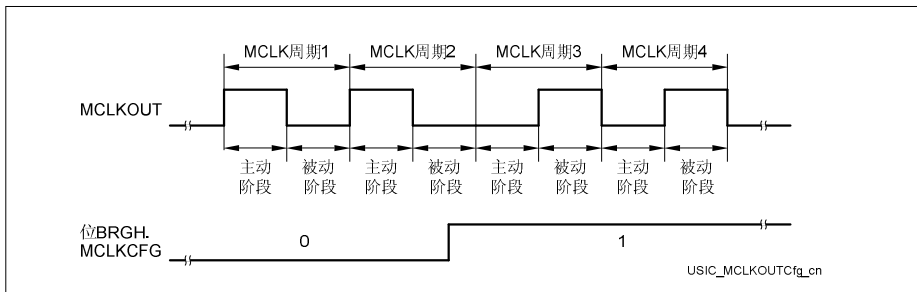


图 21-14 主控时钟输出配置

相应输出引脚上的移位时钟输出信号 SCLKOUT 极性可配置，并会引入一个  $f_{PDIV}$  周期 ( $=0.5$  个 SCLK 周期) 的延迟。该延迟使得可根据应用需求来调整移位时钟沿的顺序。若使用该延迟，在计算信号传输时间和环路延迟时必须将其考虑在内。

SCLKOUT 信号和 MCLKOUT 的极性控制机制相似，基于位域 BRGH.SCLKCFG 的配置。由协议预处理器使能/禁止产生 SCLKOUT 信号。根据所选的协议，协议预处理器可控制产生 SCLKOUT 信号（无论是否进行多级分频处理）；若协议不需从引脚上送出移位时钟，则禁止产生 SCLKOUT。

## 21.2.8 波特率发生器寄存器

### 21.2.8.1 分数分频器寄存器

分数分频器寄存器 FDRL 和 PDRH 用于控制基于系统时钟  $f_{SYS}$  产生内部频率  $f_{FD}$ 。

#### FDRL

分数分频器寄存器 L

(04<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
STEP	[9:0]	rw	<b>STEP 值</b> 在正常分频模式下，STEP 中存放 RESULT 计数到 3FF <sub>H</sub> 后的 RESULT 重载值。

符号	位序号	类型	功能描述
			在分数分频模式下，STEP 定义每个输入时钟周期和 RESULT 相加的数值。
<b>DM</b>	[15:14]	rw	<b>分频模式</b> 该位域选择分数分频器的功能。 00 <sub>B</sub> 分数分频器关闭， $f_{FD} = 0$ 。 01 <sub>B</sub> 选择正常分频模式。 10 <sub>B</sub> 选择分数分频模式。 11 <sub>B</sub> 分数分频器关闭， $f_{FD} = 0$ 。
<b>0</b>	[13:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

## FDRH

分数分频器寄存器 H

(06<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>		<b>0</b>				<b>RESULT</b>									
rw		r				rh									

符号	位序号	类型	功能描述
<b>RESULT</b>	[9:0]	rh	<b>结果值 (RESULT 值)</b> 在正常分频模式下，基于系统频率 $f_{sys}$ 更新该位域： $RESULT = RESULT + 1$ 。 在分数分频模式下，基于系统频率 $f_{sys}$ 更新该位域： $RESULT = RESULT + STEP$ 。 若 DM 设置为 00 <sub>B</sub> 或 11 <sub>B</sub> ，RESULT 的重载值为 3FF <sub>H</sub> 。
<b>0</b>	[15:14]	rw	<b>保留待用</b> 必须设置为 0 从而确保正确的分数分频操作。
<b>0</b>	[13:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 21.2.8.2 波特率发生器寄存器

寄存器 BRGL 和 BRGH 用来控制协议相关分频器以产生期望波特率。

#### BRGL

#### 波特率发生器寄存器 L

(1C<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	DCTQ					PCTQ	CTQSEL	0	PPP EN	TM EN	0	CLKSEL			
r	rw					rw	rw	r	rw	rw	r	rw			

符号	位序号	类型	功能描述
CLKSEL	[1:0]	rw	<b>时钟选择</b> 该位域定义输入频率 $f_{PIN}$ 。 00 <sub>B</sub> 选择分数分频器的输出频率 $f_{FD}$ 。 01 <sub>B</sub> 保留，无操作。 10 <sub>B</sub> 由触发信号 DX1T 产生 $f_{PIN}$ 。信号 MCLK 随 $f_{PIN}$ 翻转。 11 <sub>B</sub> 信号 MCLK 对应信号 DX1S，由 DX1S 的上升沿产生 $f_{PIN}$ 。
TMEN	3	rw	<b>时序测量使能</b> 该位定义协议相关分频器的功能。 0 <sub>B</sub> 分频模式: $f_{PDIV} = f_{PPP} / (PDIV+1)$ 可进行数据传送，忽略触发信号 DX0T 和 DX1T。 1 <sub>B</sub> 捕获模式: 10 位计数器对 $f_{PPP}$ 递增计数，计数到最大值后停止。触发信号 DX0T 或 DX1T 被激活时，计数器的值被捕获到位域 PDIV 中，随后计数器清零、产生发送移位事件。不可进行数据传送。
PPPEN	4	rw	<b>使能 2:1 分频产生 <math>f_{PPP}</math></b> 该位定义输入频率 $f_{PPP}$ 。 0 <sub>B</sub> 禁止 2:1 分频产生 $f_{PPP}$ 。

符号	位序号	类型	功能描述
			$f_{PPP} = f_{PIN}$ $1_B$ 允许 2:1 分频产生 $f_{PPP}$ 。 $f_{PPP} = f_{MCLK} = f_{PIN}/2$
<b>CTQSEL</b>	[7:6]	rw	<b>CTQ 的输入选择</b> 该位定义协议预处理器中一个时间单元的长度。 $00_B$ $f_{CTQIN} = f_{PDIV}$ $01_B$ $f_{CTQIN} = f_{PPP}$ $10_B$ $f_{CTQIN} = f_{SCLK}$ $11_B$ $f_{CTQIN} = f_{MCLK}$
<b>PCTQ</b>	[9:8]	rw	<b>时间单元计数器预分频</b> 该位域定义时间单元计数器的时间单元 $t_Q$ 的长度。 $t_Q = (PCTQ + 1) / f_{CTQIN}$
<b>DCTQ</b>	[14:10]	rw	<b>时间单元计数器的分母</b> 该位域定义时间单元计数器所用的时间单元 $t_Q$ 的个数。
<b>0</b>	2, 5, 15	r	<b>保留</b> 读操作返回 0；应写入 0。

## BRGH

波特率发生器寄存器 H

(1E<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>SCLKCFG</b>	<b>MCLKCFG</b>	<b>0</b>	<b>PDIV</b>												
rw	rw	r	rwh												

符号	位序号	类型	功能描述
<b>PDIV</b>	[9:0]	rwh	<b>分频模式：产生 <math>f_{PDIV}</math> 的分频因子</b> 该位域定义输入频率 $f_{PPP}$ 和分频频率 $f_{PDIV}$ 之间的比例因子。 <b>捕获模式：捕获的时间间隔</b>

符号	位序号	类型	功能描述
			当输入级激活相应的触发信号 DX0T 或 DX1T 时，计数器的值被捕获到该位域中。
<b>MCLKCFG</b>	13	rw	<b>主控时钟配置</b> 该位域定义 MCLKOUT 信号被动阶段的电平。 0 <sub>B</sub> 被动电平为 0。 1 <sub>B</sub> 被动电平为 1。
<b>SCLKCFG</b>	[15:14]	rw	<b>移位时钟输出配置</b> 该位定义 SCLKOUT 信号被动阶段的电平并使能/禁止长度为 0.5 个 SCLK 周期的延迟。 00 <sub>B</sub> 被动电平为 0，禁止延迟。 01 <sub>B</sub> 被动电平为 1，禁止延迟。 10 <sub>B</sub> 被动电平为 0，允许延迟。 11 <sub>B</sub> 被动电平为 1，允许延迟。
<b>0</b>	[12:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 21.2.9 操作发送数据通路

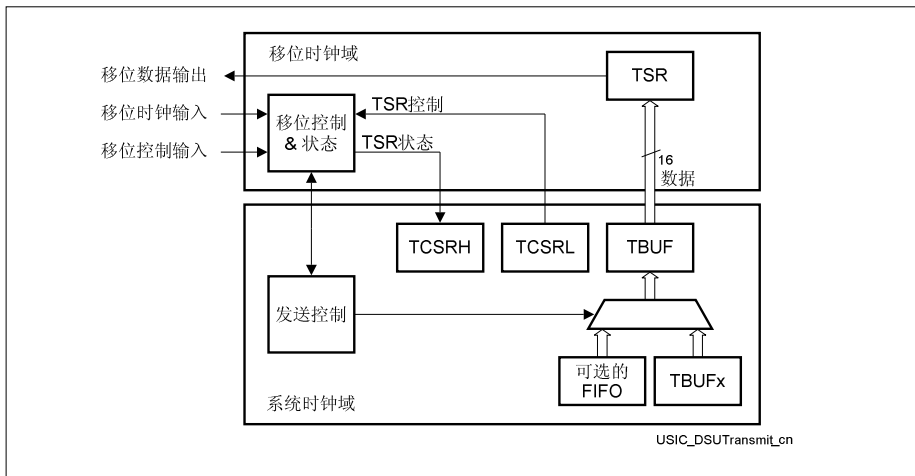
发送数据通路由一个 16 位发送移位寄存器 TSR 和一个发送缓存 TBUF 构成。发送和接收所涉及的数据传送参数（如数据字长、帧长或移位方向）均由移位控制寄存器控制。寄存器 TCSRL 主要控制发送数据的处理；TCSRH 监控发送状态。

只有当移位时钟输入信号发生相应的跳变时，数据移位输出信号 DOUT 才会改变。在移位时钟的下一次跳变沿未到来、未开始发送新数据字之前，DOUT 将保持前一个数据字/帧中最后一位数据的电平。

### 21.2.9.1 发送缓存

发送移位寄存器 TSR 不能由软件直接访问。若已完成当前的数据字发送、可发送新数据时，保存在发送缓存 TBUF 中的值会自动更新 TSR。可通过将发送数据写入发送缓存输入单元 TBUFx（见页 21-46）或通过 FIFO 缓存（可选，见页 21-67）直接加载 TBUF。





**图 21-15 发送数据通路**

### 21.2.9.2 发送控制信息

发送控制信息 TCI 可用作数据传送的附加控制参数。TCI 的值对应被写入的发送缓存输入单元 TBUFx 的地址 x。

可通过 TCI 动态改变数据字长、数据帧长或其它特定的协议功能（具体请参见相关协议章节的描述）。可通过编程寄存器 TCSRL 中的位 WLEMD、FLEMD、SELMD 和 WAMD 控制如何利用 TCI 来满足不同应用的需求。需要注意：并非所有可能的设置都对系统行为有所帮助。

- 字长控制

若 TCSRL.WLEMD = 1，当发送数据写入发送缓存输入单元 TBUFx 时，由 TCI[3:0]更新位域 SCTR.H.WLE。所有协议均可使用该功能动态改变数据字长（每个数据字可包括 1-16 位数据）。此外，由 TCI[4]更新位域 TCSRL.EOF，在 SSC 主控模式下可使用该功能控制从控选择的产生以完成数据帧传送。建议编程 TCSRL.FLEMD = TCSRL.SLEMD = 0。

- 帧长控制

若 TCSRL.FLEMD = 1，当发送数据写入发送缓存输入单元 TBUFx 时，由 TCI[4:0]更新位域 SCTR.H.FLE[4:0]；SCTR.H.FLE[5]变为 0。所有协议均可使用该功能动态改变数据帧长（每个数据帧可包括 1-32 位数据）。建议编程 TCSRL.SLEMD = TCSRL.WLEMD = TCSRL.WAMD = 0。

- 选择输出控制

若  $TCSR.LSLEMD = 1$ ，当发送数据写入发送缓存输入单元  $TBUFx$  时，由  $TCI[4:0]$  更新位域  $PCR.CTR[20:16]$ ； $PCR.CTR[23:21]$  变为 0。在 SSC 主控模式下可使用该功能定义目标从控设备。建议编程  $TCSR.WLEMD = TCSR.FLEMD = TCSR.WAMD = 0$ 。

- 字地址控制

若  $TCSR.WAMD = 1$ ，当发送数据写入发送缓存输入单元  $TBUFx$  时，由  $TCI[4]$  更新位域  $TCSR.WA$ 。在 IIS 模式下可使用该功能定义数据字发送至左通道还是右通道。建议编程  $TCSR.LSLEMD = TCSR.FLEMD = 0$ 。

### 21.2.9.3 发送数据有效性检验

发送缓存  $TBUF$  中的数据字可通过位  $TCSR.TDV$ （发送数据有效）标记为有效或无效。和数据流以及和事件相关的标准共同决定了数据字是否有效可进行发送。数据有效性检验逻辑检查每个数据字开始传送的条件。根据检查结果向发送移位寄存器加载不同的值，具体规则如下：

- 若  $USIC$  通道是通信主控方（它决定开始传送每个数据字），只有当发送缓存  $TBUF$  中的数据有效时，才能开始数据字传送。在这种情况下，将  $TBUF$  的内容载入发送移位寄存器，该操作不会改变  $TBUF$  的值。
- 若  $USIC$  通道是通信从控方（它自身不能决定开始传送，但必须作出应答），当通信主控方请求传送数据字时，无论  $TBUF$  中数据字的状态如何，都必须开始进行传送。若主控方请求并启动传送，在第一个相应的移位时钟沿到来时，将  $TBUF$  中的数据字（若它是有效数据）或  $SCTRL.PDL$  定义的电平（若开始发送时  $TBUF$  的内容无效）载入发送移位寄存器。在以上两种情况下， $TBUF$  的内容不改变。

数据有效性检验的控制和状态位存放在寄存器  $TCSR$  和  $TCSRH$  中。数据有效性检验的逻辑框图如图 21-16 所示。

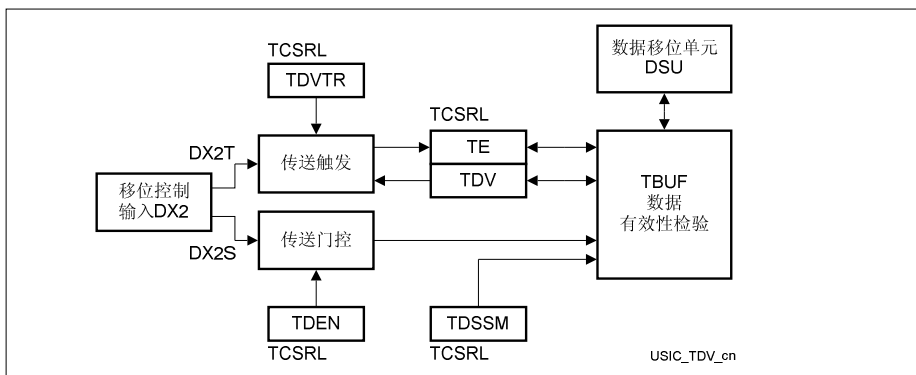


图 21-16 发送数据有效性检验

- 传送门控逻辑使能/禁止传送 TBUF 中的数据字（软件或硬件控制）。若数据移位不需使用输入级 DX2，信号 DX2S 可用于门控控制。由位域 TCSRL.TDEN 控制传送门控逻辑。
- 传送触发逻辑支持由事件（如定时器事件或输入引脚上的事件）触发数据字传送。若数据移位不需使用输入级 DX2，信号 DX2T 可用于触发控制。由位 TCSRL.TDVTR 控制传送触发逻辑，位 TCSRH.TE 指示是否发生触发事件。
- 数据有效性检验逻辑的输入来自门控逻辑、触发逻辑和 DSU 信号。只有当门控逻辑允许发送、位 TCSRL.TDV = 1 且 TCSRH.TE = 1 时，才能开始发送 TBUF 中的数据字。当 TBUF 中的内容标记为有效、可开始新的发送时，它不应被新数据覆盖。若必须修改 TBUF 的值，建议在更新数据之前通过设置 FMRL.MTDV = 10<sub>B</sub> 对 TCSRL.TDV 清零。TBUF 被新数据更新后，TCSRL.TDV 自动置位。另一种可能是通过中断 TBI（ASC 或 IIC）或 RSI（SSC 或 IIS）指示已开始发送。在发送过程中，可向 TBUF 载入新数据。因此，用户必须注意，在开始发送新数据之前必须更新 TBUF 的内容。

基于这样的结构，可实现以下数据传送功能：

- 若位 TCSRL.TDSSM = 0，发送缓存 TBUF 中的内容始终被认为是有效的。可采用传送触发机制：当发生设定的事件时（如定时器事件或输入引脚上的跳变沿事件），将开始传送相同的数据字，从而实现了一种“活跃”状态机制。在从控模式下，这样可确保始终发送正确的数据字（代替被动数据电平）。
- 采用一种“单次”机制实现逐字传送时，必须设置位 TCSRL.TDSSM = 1。每次开始传送后，必须向 TBUF 载入新数据，这可通过将发送数据写入发送缓存输入单元 TBUFx、或通过数据缓存（如 FIFO 缓存）来实现。为了避免多次发送数据字、或者为了使用附加数据缓存（如 FIFO）处理数据，位 TCSRL.TDSSM 必须置 1。
- 新数据载入发送缓存 TBUF 后，位 TCSRL.TDV 自动置位。可通过将发送数据写入发送缓存输入单元 TBUFx（至少写入其低位字节）的方式请求数据传送。可利用附加信息 TCI 控制每个数据字的字长或其它参数。
- 可通过位域 FMRL.MTDV 软件修改（置位或清零）位 TCSRL.TDV。此外，结合对门控位域 TCSRL.TDEN 的设置，用户可建立发送数据字而不开始发送。可按以下步骤编程：TCSRL.TDEN = 00<sub>B</sub>，向 TBUFx 写入数据，设置 FMRL.MTDV = 10<sub>B</sub> 清除位 TCSRL.TDV，设置 TCSRL.TDEN = 01<sub>B</sub> 重新使能门控控制，之后通过设置 FMRL.MTDV = 01<sub>B</sub> 软件置位 TCSRL.TDV。

### 21.2.10 操作接收数据通路

接收数据通路由两个 16 位接收移位寄存器 RSR0、RSR1 和各自的接收缓存（RBUF0 和 RBUF1）构成。发送和接收所涉及的数据传送参数（如数据字长、帧长或移位方向）均由移位控制寄存器控制。

寄存器 RBUF01SRL 监控 RBUF0 的状态；RBUF01SRH 监控 RBUF1 的状态。



## 21.2.11 传送控制和状态寄存器

### 21.2.11.1 移位控制寄存器

数据移位单元由以下寄存器控制。这些寄存器的值用于数据发送和接收。

需要注意：发送器和接收器共用移位控制设置 **SDIR**、**WLE** 和 **FLE**。对于每一次数据字传送，由第一个发送移位时钟沿（发送器）和第一个接收移位时钟沿（接收器）内部“冻结”这些位域的值。用户必须注意，对这些位域的软件更新要一致（如参照接收开始事件指示标志 **PSR.RSIF**）。

#### SCTRL

##### 移位控制寄存器 L

(30<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						TRM		DOCFG		0			PDL	SDIR	
r						rw		rw		r			rw	rw	

符号	位序号	类型	功能描述
<b>SDIR</b>	0	rw	<b>移位方向</b> 该位定义数据字的移位方向。 0 <sub>B</sub> LSB 移位在先。数据字的第一位位于位 0。 1 <sub>B</sub> MSB 移位在先。数据字的第一位位于 SCTRL.WLE 指定的位置。
<b>PDL</b>	1	rw	<b>被动数据电平</b> 该位定义无数据发送时移位数据输出信号的电平。在数据字的第一个相关发送移位时钟沿输出 PDL 电平。 0 <sub>B</sub> 被动数据电平为 0。 1 <sub>B</sub> 被动数据电平为 1。
<b>DOCFG</b>	[7:6]	rw	<b>数据输出配置</b> 该位域定义内部移位数据值和数据输出信号 DOUT 的关系。 X0 <sub>B</sub> DOUT = 移位数据值 X1 <sub>B</sub> DOUT = 移位数据值反相

符号	位序号	类型	功能描述
<b>TRM</b>	[9:8]	rw	<b>发送模式</b> 该位域定义 DSU 如何解读移位控制信号。只有当移位控制信号有效时，才能进行数据传送。 <b>00<sub>B</sub></b> 移位控制信号视为无效，不能进行数据帧传送。 <b>01<sub>B</sub></b> 若移位控制信号的电平为 1，则视为有效。这是用来进行数据传送的标准设置。 <b>10<sub>B</sub></b> 若移位控制信号的电平为 0，则视为有效。建议避免这样设置，出现低有效信号时使用其反相信号。 <b>11<sub>B</sub></b> 不参照实际的信号电平，移位控制信号视为有效。信号每次跳变后可进行数据帧传送。
<b>0</b>	[5:2], [15:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

## SCTRH

**移位控制寄存器 H**

**(32<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>				<b>WLE</b>				<b>0</b>		<b>FLE</b>					
r				rwh				r		rwh					

符号	位序号	类型	功能描述
<b>FLE</b>	[5:0]	rwh	<b>帧长</b> 该位域定义一个数据帧由多少位数据组成。一个数据帧可由多个数据字串联组成。 若 TCSRL.FLEMD = 1，可由数据处理器自动更新 FLE。
<b>WLE</b>	[11:8]	rwh	<b>字长</b> 该位域定义发送和接收的数据字的长度（每个数据字由多少位数据组成）。数据字在数据缓存中始终右对齐，即位于[WLE:0]。 若 TCSRL.WLEMD = 1，可由数据处理器自动更

符号	位序号	类型	功能描述
			新 WLE。 0 <sub>H</sub> 一个数据字包含 1 位数据，位置在位 0。 1 <sub>H</sub> 一个数据字包含 2 位数据，位置在位[1:0]。 ... E <sub>H</sub> 一个数据字包含 15 位数据，位置在位 [14:0]。 F <sub>H</sub> 一个数据字包含 16 位数据，位置在位 [15:0]。
<b>0</b>	[7:6], [15:12]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 21.2.11.2 发送控制和状态寄存器

由寄存器 TCSR<sub>L</sub> 控制数据发送。

#### TCSR<sub>L</sub>

发送控制/状态寄存器 L

(3C<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	WA	TD VTR	TDEN			0	TD SSM	TDV	EOF	SOF	0	WA MD	FLE MD	SEL MD	WLE MD
r	rw	rw	rw			r	rw	rh	rw	rw	r	rw	rw	rw	rw

符号	位序号	类型	功能描述
<b>WLEMD</b>	0	rw	<b>WLE 模式</b> 该位使能自动更新位域 SCTRH.WLE 和 TCSR.EOF。由发送控制信息 TCI[3:0]更新 SCTRH.WLE；TCI[4]更新 TCSR.EOF（见 <a href="#">页 21-46</a> ）。若该位被使能，通过将新数据写入发送缓存输入单元 TBUF <sub>x</sub> 或通过 FIFO 缓存（可选）加载寄存器 TBUF 时，自动进行更新。 0 <sub>B</sub> 禁止自动更新 SCTRH.WLE 和 TCSR.EOF。 1 <sub>B</sub> 允许自动更新 SCTRH.WLE 和 TCSR.EOF。

符号	位序号	类型	功能描述
<b>SLEMD</b>	1	rw	<p><b>选择模式</b></p> <p>该位主要用于 SSC 协议。它使能自动更新位域 PCRH.CTR[20:16]、清除 PCRH.CTR[23:21]。由发送控制信息 TCI[4:0]更新 PCRH.CTR[20:16]（见页 21-46）。若该位被使能，通过将新数据写入发送缓存输入单元 TBUFx 或通过 FIFO 缓存（可选）加载寄存器 TBUF 时，自动进行更新。</p> <p>0<sub>B</sub> 禁止自动更新 PCRH.CTR[23:16]。 1<sub>B</sub> 允许自动更新 PCRH.CTR[23:16]。</p>
<b>FLEMD</b>	2	rw	<p><b>FLE 模式</b></p> <p>该位使能自动更新位域 SCTRH.FLE[4:0]、清除 SCTRH.FLE[5]。由发送控制信息 TCI[4:0]更新 SCTRH.FLE[4:0]（见页 21-46）。若该位被使能，通过将新数据写入发送缓存输入单元 TBUFx 或通过 FIFO 缓存（可选）加载寄存器 TBUF 时，自动进行更新。</p> <p>0<sub>B</sub> 禁止自动更新 FLE。 1<sub>B</sub> 允许自动更新 FLE。</p>
<b>WAMD</b>	3	rw	<p><b>WA 模式</b></p> <p>该位主要用于 IIS 协议。它使能由发送控制信息 TCI[4]自动更新位 TCSRL.WA（见页 21-46）。若该位被使能，通过将新数据写入发送缓存输入单元 TBUFx 或通过 FIFO 缓存（可选）加载寄存器 TBUF 时，自动进行更新。</p> <p>0<sub>B</sub> 禁止自动更新 WA。 1<sub>B</sub> 允许自动更新 WA。</p>
<b>SOF</b>	5	rwh	<p><b>数据帧开始</b></p> <p>该位只用在 SSC 协议中，否则被忽略不用。</p> <p>TBUF 中的数据字有效（TCSRL.TDV = 1）时，SOF 指示它是否是 SSC 新一帧的第一个字。当 TBUF 中的数据字传送到发送移位寄存器之后，该位被清零。</p> <p>0<sub>B</sub> TBUF 中的数据字不是新一帧的第一个字。 1<sub>B</sub> TBUF 中的数据字是新一帧的第一个字。结束</p>



符号	位序号	类型	功能描述
			当前正在发送的数据帧，MSLS 变为无效（考虑设定的延迟）。
<b>EOF</b>	6	rwh	<p><b>数据帧结束</b></p> <p>该位只用在 SSC 协议中，否则被忽略不用。若位 WLEMD = 1，该位可被自动修改。</p> <p>EOF 指示 TBUF 中的数据字是否是一帧中的最后一个字。若是最后一个字，当 TBUF 中的数据字传送到发送移位寄存器之后，MSLS 变为无效（考虑设定的延迟）、该位被清零。</p> <p>0<sub>B</sub> TBUF 中的数据字不是一帧中的最后一个字。</p> <p>1<sub>B</sub> TBUF 中的数据字是一帧中的最后一个字。</p>
<b>TDV</b>	7	rh	<p><b>发送数据有效</b></p> <p>该位指示发送缓存 TBUF 中的数据字是否有效。只有当 TDV = 1 时才能发送 TBUF 中的数据字。通过向发送缓存输入单元 TBUFx 写入数据或通过 FIFO 缓存（可选）加载寄存器 TBUF 时，该位自动置位。</p> <p>0<sub>B</sub> TBUF 中的数据字无效。</p> <p>1<sub>B</sub> TBUF 中的数据字有效，可以开始发送。当 TDV = 1 时不应向 TBUFx 写入新数据。</p>
<b>TDSSM</b>	8	rw	<p><b>TBUF 数据单次模式</b></p> <p>该位定义 TBUF 中的数据字是恒有效还是应该只传送一次。</p> <p>0<sub>B</sub> TBUF 中的数据字载入发送移位寄存器之后仍视为有效，该操作不会清除 TDV。</p> <p>1<sub>B</sub> TBUF 中的数据字载入发送移位寄存器之后被视为无效。在 ASC 和 IIC 模式下，TDV 由 TBI 事件清零；在 SSC 和 IIS 模式下，TDV 由 RSI 事件清零。</p> <p>若使用数据缓存，必须设置 TDSSM = 1。</p>
<b>TDEN</b>	[11:10]	rw	<p><b>TBUF 数据使能</b></p> <p>该位域对开始传送 TBUF 中的数据字进行门控控制。</p>

符号	位序号	类型	功能描述
			<p>00<sub>B</sub> 禁止传送 TBUF 中的数据字。若启动数据传送，则输出 pasive 数据电平。</p> <p>01<sub>B</sub> 若 TDV = 1，可开始传送 TBUF 中的数据字。</p> <p>10<sub>B</sub> 若 TDV = 1 且 DX2S = 0，可开始传送 TBUF 中的数据字。</p> <p>11<sub>B</sub> 若 TDV = 1 且 DX2S = 1，可开始传送 TBUF 中的数据字。</p>
<b>TDVTR</b>	12	rw	<p><b>TBUF 数据有效触发</b></p> <p>若触发信号 DX2T 变为有效，该位使能传送触发单元置位 TCSRH.TE，从而支持由事件（如定时器事件或输入引脚上的事件）触发数据传送。若输入级 DX2 用于数据移位，位 TDVTR 必须置 0。</p> <p>0<sub>B</sub> 位 TCSRH.TE 恒置位。</p> <p>1<sub>B</sub> 当 TDV = 1 时，若触发信号 DX2T 变为有效，位 TCSRH.TE 置位。</p>
<b>WA</b>	13	rwh	<p><b>字地址</b></p> <p>该位只用于 IIS 协议，否则忽略不用。若 WAMD = 1，它可被自动更新。位 WA 定义 TBUF 中的数据字发送至哪路通道。</p> <p>0<sub>B</sub> 检测到 WA 的下降沿后（参照 PSR.WA）发送 TBUF 中的数据字。</p> <p>1<sub>B</sub> 检测到 WA 的上升沿后（参照 PSR.WA）发送 TBUF 中的数据字。</p>
<b>0</b>	4,9, [15:14]	r	<p><b>保留</b></p> <p>读操作返回 0；应写入 0。</p>

由寄存器 TCSRH 监控数据发送的状态。

**TCSRH**

**发送控制/状态寄存器 H**

**(3E<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>	<b>TE</b>	<b>TVC</b>	<b>TV</b>	<b>0</b>	<b>TSOF</b>	<b>0</b>									
r	rh	rh	rh	r	rh	r									

符号	位序号	类型	功能描述
<b>TSOF</b>	8	rh	<b>数据帧开始发送</b> 该位指示是否已经开始发送新数据帧中的第一个数据字。每次开始发送数据字之后该位更新。 <b>0<sub>B</sub></b> 未开始发送数据帧中的第一个数据字。 <b>1<sub>B</sub></b> 已开始发送数据帧中的第一个数据字。
<b>TV</b>	10	rh	<b>发送有效</b> 该位表示发送缓存下溢并指示最近开始发送的数据字是否是发送缓存 <b>TBUF</b> 中的有效数据。每次开始发送数据字之后该位更新。 <b>0<sub>B</sub></b> 已开始发送数据字，但 <b>TBUF</b> 中无有效数据可用。因此，所发送数据字的电平为被动电平（ <b>SCTRL.PDL</b> ）。 <b>1<sub>B</sub></b> 最近开始发送的数据字是发送缓存 <b>TBUF</b> 中的有效数据。
<b>TVC</b>	11	rh	<b>发送有效累积</b> 该位累积发送缓存下溢指示位 <b>TV</b> 。该位和位 <b>TV</b> 一并自动清零，必须由 <b>FMRL.ATVC = 1</b> 置位。 <b>0<sub>B</sub></b> 自从 <b>TVC</b> 被置位，至少已出现一次数据缓存下溢。 <b>1<sub>B</sub></b> 自从 <b>TVC</b> 被置位，还未出现数据缓存下溢。
<b>TE</b>	12	rh	<b>触发事件</b> 若传送触发机制使能， <b>TCSRL.TDV = 1</b> 时该位指示已检测到触发事件（ <b>DX2T = 1</b> ）。若传送触发机制禁用，位 <b>TE</b> 恒置位。设置 <b>FMRL.MTDV = 10<sub>B</sub></b> 或 <b>TBUF</b> 中的数据字载入移位寄存器时，该位被清零。 <b>0<sub>B</sub></b> 未检测到触发事情。不能开始发送 <b>TBUF</b> 中的数据字。 <b>1<sub>B</sub></b> 已检测到触发事情（或触发机制关闭）。可以开始发送 <b>TBUF</b> 中的数据字。
<b>0</b>	[7:0], 9, [15:13]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 21.2.11.3 标志修改寄存器

标志修改寄存器 FMRL 和 FMRH 可用来修改和数据处理相关的控制和状态标志。读取 FMRL 和 FMRH 始终返回 0。

此外，该 USIC 通道的服务请求输出可由软件激活（由写操作激活、自动失效）。

#### FMRL

##### 标志修改寄存器 L

(38<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C RDV1	C RDV0	0										A TVC	0	MTDV	
w	w	r										w	r	w	

符号	位序号	类型	功能描述
<b>MTDV</b>	[1:0]	w	<b>修改发送数据有效</b> 软件操作该位域可修改位 TCSR.L.TDV 和 TCSR.H.TE，从而控制开始发送数据字。 00 <sub>B</sub> 无操作。 01 <sub>B</sub> 置位 TDV，TE 不变。 10 <sub>B</sub> 清除 TDV 和 TE。 11 <sub>B</sub> 保留。
<b>ATVC</b>	4	w	<b>激活位 TVC</b> 软件操作该位可置位 TCSR.H.TVC，从而对发送缓存下溢开始重新累积。 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 置位 TCSR.H.TVC。
<b>CRDV0</b>	14	w	<b>清除 RBUF0 的位 RDV</b> 软件对该位置 1 将清除 RBUF0.SRL.RDV00 和 RBUF0.SRH.RDV10，从而声明 RBUF0 中的接收数据不再有效。 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除 RBUF0.SRL.RDV00 和 RBUF0.SRH.RDV10。

符号	位序号	类型	功能描述
CRDV1	15	w	<b>清除 RBUF1 的位 RDV</b> 软件对该位置 1 将清除 RBUF01SRL.RDV01 和 RBUF01SRH.RDV11，从而声明 RBUF1 中的接收数据不再有效。 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 清除 RBUF01SRL.RDV01 和 RBUF01SRH.RDV11。
0	[3:2], [13:5]	r	<b>保留</b> 读操作返回 0；应写入 0。

## FMRH

### 标志修改寄存器 H

(3A<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0												SI O3	SI O2	SI O1	SI O0
r												w	w	w	w

符号	位序号	类型	功能描述
SIO0, SIO1, SIO2, SIO3	0, 1, 2, 3	w	<b>设置中断输出 SRx</b> 软件对该位域置 1 将激活该 USIC 通道的服务请求输出 SRx。该操作不影响其它 USIC 通道的服务请求输出。 0 <sub>B</sub> 无操作。 1 <sub>B</sub> 激活服务请求输出 SRx。
0	[15:4]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 21.2.12 数据缓存寄存器

### 21.2.12.1 发送缓存输入单元

32 个独立的数据输入单元 TBUF00 - TBUF31 可用作发送缓存的数据入口地址单元。写入其中某一单元的数据将出现在共用寄存器 TBUF 中。此外，数据入口单元地址编号[31:0]对应的 5 位编码代表发送控制信息 TCI（具体请参阅协议章节的描述）。

内部发送缓存寄存器 TBUF 中保存的数据在下次发送时将被载入发送移位寄存器。它可从 TBUF00 - TBUF31 中读出。

#### TBUF<sub>x</sub> (x = 00-31)

发送缓存输入单元																(80 <sub>H</sub> + x*4)																复位值: 0000 <sub>H</sub>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																
TDATA																																															
rwh																																															

符号	位序号	类型	功能描述
TDATA	[15:0]	rwh	发送数据 该位域保存将要发送的数据。 向 TDATA 写数据的操作（至少写入其低位字节） 置位 TCSRL.TDV。

### 21.2.12.2 接收缓存寄存器 RBUF0、RBUF1

接收缓存寄存器 RBUF0 中保存从 RSR0 接收的数据。对 RBUF0 的读操作不会使接收数据的状态从“还未读取 = 有效”变为“已读取 = 无效”。

#### RBUF0

接收缓存寄存器 0																(50 <sub>H</sub> )																复位值: 0000 <sub>H</sub>															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																
DSR0																																															
rh																																															

符号	位序号	类型	功能描述
<b>DSR0</b>	[15:0]	rh	<b>移位寄存器 0 的数据</b>

接收缓存状态寄存器 RBUF01SRL 给出接收缓存 RBUF0 中数据的状态。

### RBUF01SRL

**接收缓存 01 状态寄存器 L**

**(60<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DS</b> <b>0</b>	<b>RDV</b> <b>01</b>	<b>RDV</b> <b>00</b>	<b>0</b>			<b>P</b> <b>ERR0</b>	<b>PAR</b> <b>0</b>	<b>0</b>	<b>SOF</b> <b>0</b>	<b>0</b>			<b>WLEN0</b>		
rh	rh	rh	r			rh	rh	r	rh	r			rh		

符号	位序号	类型	功能描述
<b>WLEN0</b>	[3:0]	rh	<b>RBUF0 中接收数据的字长</b> 该位域指示保存在 RBUF0 中的最后一个数据字的字长。该数字指示有多少位数据必须视为接收数据，RBUF0 中的其它位已被自动清零。接收到的数据位始终右对齐。 0 <sub>H</sub> 已接收 1 位数据。 ..... F <sub>H</sub> 已接收 16 位数据。
<b>SOF0</b>	6	rh	<b>RBUF0 中的起始数据字</b> 该位指示 RBUF0 中的数据字是否是一帧数据中的第一个数据字。 0 <sub>H</sub> RBUF0 中的数据已不是一帧数据中的第一个数据字。 1 <sub>H</sub> RBUF0 中的数据是一帧数据中的第一个数据字。
<b>PAR0</b>	8	rh	<b>RBUF0 中的协议相关参数</b> 该位指示协议相关参数的值。根据所选协议对该位进行详细说明，它为 RBUF0 中的数据字附加了额外信息。 该位的具体含义在相关协议章节中描述。

符号	位序号	类型	功能描述
<b>PERR0</b>	9	rh	<p><b>RBUF0 中的协议相关错误</b></p> <p>该位指示协议相关参数的值是否和期望值一致。根据所选协议对该位进行详细说明，它为 RBUF0 中的数据字附加了额外信息。</p> <p>该位的具体含义在相关协议章节中描述。</p> <p><b>0<sub>B</sub></b> 接收到的协议相关参数 PAR 和期望值一致。接收数据字之后置位 PSR.RIF 并可产生接收中断。</p> <p><b>1<sub>B</sub></b> 接收到的协议相关参数 PAR 和期望值不一致。接收到数据字之后置位 PSR.AIF 并可产生备选接收中断。</p>
<b>RDV00</b>	13	rh	<p><b>RBUF0 中的接收数据有效</b></p> <p>该位指示寄存器 RBUF0 中数据的状态。该位和 RBUF01SRH.RDV10 相同，从而确保读取信息一致。当新数据字保存到 RBUF0 时该位置位；数据字被 RBUF 读出后该位自动清零。</p> <p><b>0<sub>B</sub></b> 寄存器 RBUF0 中不包含还未读出的数据。</p> <p><b>1<sub>B</sub></b> 寄存器 RBUF0 中包含还未读出的数据。</p>
<b>RDV01</b>	14	rh	<p><b>RBUF1 中的接收数据有效</b></p> <p>该位指示寄存器 RBUF1 中数据的状态。该位和 RBUF01SRH.RDV11 相同，从而确保读取信息一致。当新数据字保存到 RBUF1 时该位置位；数据字被 RBUF 读出后该位自动清零。</p> <p><b>0<sub>B</sub></b> 寄存器 RBUF1 中不包含还未读出的数据。</p> <p><b>1<sub>B</sub></b> 寄存器 RBUF1 中包含还未读出的数据。</p>
<b>DS0</b>	15	rh	<p><b>数据源</b></p> <p>该位指示哪个接收缓存寄存器（RBUF0 或 RBUF1）当前在寄存器 RBUF(D) 中可见、可由 RBUFSR 指示相关的状态信息。它指示哪个缓存中包含最早的数据（最先接收到的数据）。该位和 RBUF01SRH.DS1 相同，从而确保读取信息一致。</p> <p><b>0<sub>B</sub></b> 寄存器 RBUF 中包含 RBUF0 的数据（RBUFSR 中包含 RBUF0 的状态信息）。</p>



符号	位序号	类型	功能描述
			1 <sub>B</sub> 寄存器 RBUF 中包含 RBUF1 的数据 (RBUFSR 中包含 RBUF1 的状态信息)。
0	[5:4],7, [12:10]	r	保留 读操作返回 0；应写入 0。

接收缓存寄存器 RBUF1 中保存从 RSR1 接收的数据。对 RBUF1 的读操作不会使接收数据的状态从“还未读取 = 有效”变为“已读取 = 无效”。

## RBUF1

### 接收缓存寄存器 1

(54<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DSR1															
rh															

符号	位序号	类型	功能描述
DSR1	[15:0]	rh	移位寄存器 1 的数据

接收缓存状态寄存器 RBUF01SRH 给出接收缓存 RBUF1 中数据的状态。

## RBUF01SRH

### 接收缓存 01 状态寄存器 H

(62<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DS 1	RDV 11	RDV 10	0			P ERR1	PAR 1	0	SOF 1	0					WLEN1
rh	rh	rh	r			rh	rh	r	rh	r					rh

符号	位序号	类型	功能描述
WLEN1	[3:0]	rh	<b>RBUF1 中接收数据的字长</b> 该位域指示保存在 RBUF1 中的最后一个数据字的字长。该数字指示有多少位数据必须视为接收数据，RBUF1 中的其它位已被自动清零。接收到的

符号	位序号	类型	功能描述
			<p>数据位始终右对齐。</p> <p>0<sub>H</sub> 已接收 1 位数据。</p> <p>.....</p> <p>F<sub>H</sub> 已接收 16 位数据。</p>
<b>SOF1</b>	6	rh	<p><b>RBUF1 中的帧起始数据字</b></p> <p>该位指示 RBUF1 中的数据字是否是一帧数据中的第一个数据字。</p> <p>0<sub>H</sub> RBUF1 中的数据已不是一帧数据中的第一个数据字。</p> <p>1<sub>H</sub> RBUF1 中的数据是一帧数据中的第一个数据字。</p>
<b>PAR1</b>	8	rh	<p><b>RBUF1 中的协议相关参数</b></p> <p>该位指示协议相关参数的值。根据所选协议对该位进行详细说明，它为 RBUF1 中的数据字附加了额外信息。</p> <p>该位的具体含义在相关协议章节中描述。</p>
<b>PERR1</b>	9	rh	<p><b>RBUF1 中的协议相关错误</b></p> <p>该位指示协议相关参数的值是否和期望值一致。根据所选协议对该位进行详细说明，它为 RBUF1 中的数据字附加了额外信息。</p> <p>该位的具体含义在相关协议章节中描述。</p> <p>0<sub>B</sub> 接收到的协议相关参数 PAR 和期望值一致。接收数据字之后置位 PSR.RIF 并可产生接收中断。</p> <p>1<sub>B</sub> 接收到的协议相关参数 PAR 和期望值不一致。接收到数据字之后置位 PSR.AIF 并可产生备选接收中断。</p>
<b>RDV10</b>	13	rh	<p><b>RBUF0 中的接收数据有效</b></p> <p>该位指示寄存器 RBUF0 中数据的状态。该位和 RBUF01SRL.RDV00 相同，从而确保读取信息一致。</p> <p>0<sub>B</sub> 寄存器 RBUF0 中不包含还未读出的数据。</p> <p>1<sub>B</sub> 寄存器 RBUF0 中包含还未读出的数据。</p>

符号	位序号	类型	功能描述
<b>RDV11</b>	14	rh	<b>RBUF1 中的接收数据有效</b> 该位指示寄存器 RBUF1 中数据的状态。该位和 RBUF01SRL.RDV01 相同，从而确保读取信息一致。 0 <sub>B</sub> 寄存器 RBUF1 中不包含还未读出的数据。 1 <sub>B</sub> 寄存器 RBUF1 中包含还未读出的数据。
<b>DS1</b>	15	rh	<b>数据源</b> 该位指示哪个接收缓存寄存器（RBUF0 或 RBUF1）当前在寄存器 RBUF(D) 中可见、可由 RBUFSR 指示相关的状态信息。它指示哪个缓存中包含最早的数据（最先接收到的数据）。该位和 RBUF01SRL.DS0 相同，从而确保读取信息一致。 0 <sub>B</sub> 寄存器 RBUF 中包含 RBUF0 的数据（RBUFSR 中包含 RBUF0 的状态信息）。 1 <sub>B</sub> 寄存器 RBUF 中包含 RBUF1 的数据（RBUFSR 中包含 RBUF1 的状态信息）。
<b>0</b>	[5:4],7, [12:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 21.2.12.3 接收缓存寄存器 RBUF、RBUFD、RBUFSR

接收缓存寄存器 RBUF 根据接收顺序相应显示 RBUF0 或 RBUF1 的内容。RBUF 始终可从 RBUF0 或 RBUF1 中读出最早的数据（最先接收到的数据）。建议用户从 RBUF、而不从 RBUF0/1 读取接收数据。读取 RBUF（至少其低位字节）时，接收数据的状态从“还未读取 = 有效”自动变为“已读取 = 无效”，RBUF 的内容被更新，RBUF 中显示下一次接收的数据字。

#### RBUF

接收缓存寄存器

(5C<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
DSR	[15:0]	rh	接收数据 该位域显示 RBUF0 或 RBUF1 的内容（根据接收顺序）。

若要使用调试器监控接收数据，必须使自动更新机制失效以确保数据的一致性。因此，可使用调试专用接收缓存寄存器 RBUFD。RBUFD 和 RBUF 相似，差别仅在于读取后不被自动更新。因此，调试器（或其它监控功能）可读取 RBUFD 而不干扰接收序列。

#### RBUFD

调试专用接收缓存寄存器

(4C<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
DSR	[15:0]	rh	来自移位寄存器的数据 和 RBUF.DSR 相同，但被读取后不释放缓存。

接收缓存状态寄存器 RBUFSR 给出接收缓存 RBUF 和 RBUFD 中数据的状态。若位 RBUF01SRL.DS0 (RBUF01SRH.DS1) 为 0，RBUFSR 监控 RBUF01SRL 的内容，否则监控 RBUF01SRH 的内容。

## RBUFSR

### 接收缓存状态寄存器

(58<sub>H</sub>)

复位值: 0000<sub>H</sub>

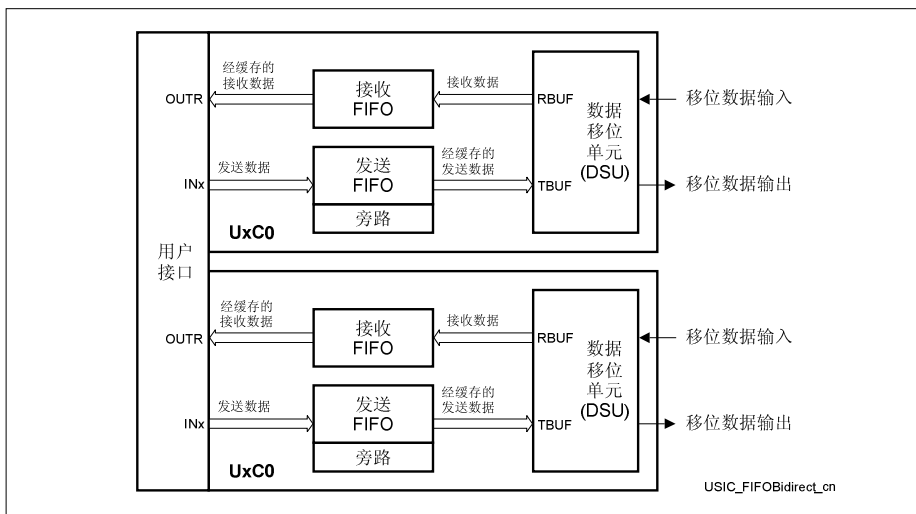
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DS	RDV1	RDV0	0			PERR	PAR	0	SOF	0			WLEN		
rh	rh	rh		r		rh	rh	r	rh		r			rh	

符号	位序号	类型	功能描述
WLEN	[3:0]	rh	<b>RBUF 或 RBUFD 中接收数据的字长</b> 描述见 RBUF01SRL.WLEN0 或 RBUF01SRH.WLEN1。
SOF	6	rh	<b>RBUF 或 RBUFD 中的起始数据字</b> 描述见 RBUF01SRL.SOF0 或 RBUF01SRH.SOF1。
PAR	8	rh	<b>RBUF 或 RBUFD 中的协议相关参数</b> 描述见 RBUF01SRL.PAR0 或 RBUF01SRH.PAR1。
PERR	9	rh	<b>RBUF 或 RBUFD 中的协议相关错误</b> 描述见 RBUF01SRL.PERR0 或 RBUF01SRH.PERR1。
RDV0	13	rh	<b>RBUF 或 RBUFD 中的接收数据有效</b> 描述见 RBUF01SRL.RDV00 或 RBUF01SRH.RDV10。
RDV1	14	rh	<b>RBUF 或 RBUFD 中的接收数据有效</b> 描述见 RBUF01SRL.RDV01 或 RBUF01SRH.RDV11。
DS	15	rh	<b>RBUF 或 RBUFD 的数据源</b> 描述见 RBUF01SRL.DS0 或 RBUF01SRH.DS1。

符号	位序号	类型	功能描述
0	[5:4],7, [12:10]	r	保留 读操作返回 0；应写入 0。

### 21.2.13 操作 FIFO 数据缓存

USIC 模块中的 FIFO 数据缓存结构相似，每路通道带有发送缓存和接收缓存。不同器件中 FIFO 缓存区的大小可以不同。在 XE16xyM 中，共有 64 个缓存入口可分配给 USIC 模块中两路通道的发送或接收 FIFO 缓存。



**图 21-18 FIFO 缓存基本结构**

为了正确操作 FIFO 数据缓存，必须考虑以下因素：

- FIFO 缓存可用及选择：

只有当  $CCFG.TB = 1$  时发送 FIFO 缓存和旁路结构才可用；只有当  $CCFG.RB = 1$  时接收 FIFO 缓存才可用。

建议在满足以下两个条件的情况下配置所有缓存参数：该 USIC 通道当前无数据通信；设置  $TBCTRL.SIZE = 0$ （发送缓存）或  $RBCTRL.SIZE = 0$ （接收缓存）禁用 FIFO 机制。必须在 FIFO 缓存被禁用时才能通过设置  $TBCTRL$  或  $RBCTRL$  分配缓存区。FIFO 缓存中断控制位可被修改，这和当前是否进行数据通信无关。

- FIFO 缓存的建立：  
可用的 FIFO 缓存入口总数限制了每路 USIC 通道发送和接收缓存的长度。
- 旁路的建立：  
除发送 FIFO 缓存之外，还可配置成 FIFO 缓存旁路，具体描述见[页 21-73](#)。

### 21.2.13.1 FIFO 缓存分区

若 FIFO 缓存可用，它由一组指定个数的 FIFO 缓存入口组成，每个入口包含数据部分和相关的控制信息（接收数据对应 RCI，发送数据对应 TCI）。每个 FIFO 缓存入口代表可分配给接收 FIFO 缓存或发送 FIFO 缓存的最小单元。一个 USIC 模块中所有可用的 FIFO 缓存入口顺序排列于 FIFO 缓存区内。FIFO 入口编号从 0 开始，接下来是 1、2，依次类推。

每个 USIC 模块中有多个 FIFO 缓存入口可用，可分配给同一模块中的不同通道。不可能将 FIFO 缓存区分配给不在同一 USIC 模块中的通道。对于每一路 USIC 通道，发送和接收 FIFO 缓存的大小均可单独选择。比如，可将一路 USIC 通道中所有可用的 FIFO 缓存入口全部分配用作发送缓存。[图 21-19](#) 给出一些可能的 FIFO 缓存分区的情况。

FIFO 缓存由一组连续的 FIFO 入口组成。FIFO 数据缓存的大小可编程为 2 的整数次幂，从 2 个入口开始、然后是 4 个、8 个，依此类推。FIFO 数据缓存的起始入口只能和其长度对齐。比如，含有  $n$  个入口的 FIFO 缓存的起始入口只能是  $0$ 、 $2 \times n$ 、 $3 \times n$ ...，由  $[x \times n, (x+1) \times n - 1]$  个 FIFO 入口组成，其中  $x$  取整数（包括 0）。在一个 FIFO 缓存内部不可能有未使用的 FIFO 入口，但在两个 FIFO 缓存之间可以有未使用的 FIFO 入口。

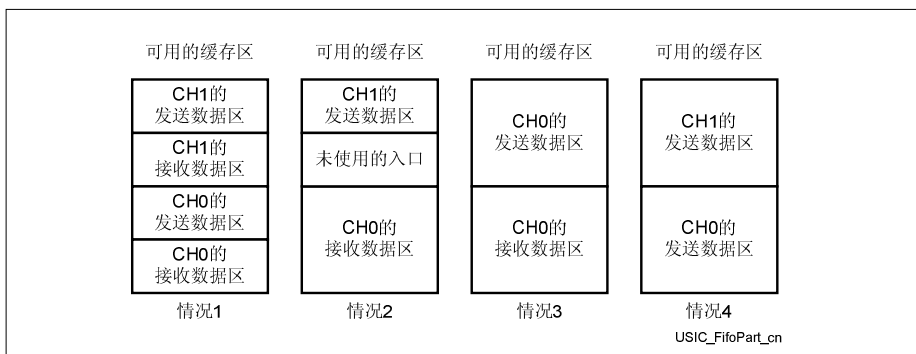


图 21-19 FIFO 缓存分区

FIFO 缓存基于指针保存数据，一旦 FIFO 中的数据已修改，指针值内部更新。当新数据存入 FIFO 缓存、或将最早的数据从 FIFO 缓存中取出时，自动执行该操作。因

此，用户程序无需修改指针。只有在初始化阶段必须由软件设定 FIFO 缓存的起始入口：位域 RBCTRH.SIZE = 0（或 TBCTRH.SIZE = 0）时，向寄存器 RBCTRL（针对接收 FIFO）或 TBCTRL（针对发送 FIFO）的位域 DPTR 写入 FIFO 缓存的起始入口编号。USIC 通道正在进行数据通信时，切勿修改缓存大小和指针。

### 21.2.13.2 数据缓存事件和中断

发送 FIFO 缓存机制检测到以下事件时可产生中断（若被使能）。

- 标准发送缓存事件：

发送缓存的填充级（由 TRBSRH.TBFLVL 给出）大于（TBCTRH.LOF = 1）或小于（TBCTRH.LOF = 0）设定的极限值（TBCTRL.LIMIT）。从等于到小于或大于的转换操作（并非小于或大于本身）将触发该事件。

若使用标准发送缓存事件指示新数据必须写入一个 INx 单元，应设置 TBCTRH.LOF = 0。

- 发送缓存错误事件：

软件向已满的缓存写数据。写入值被忽略。

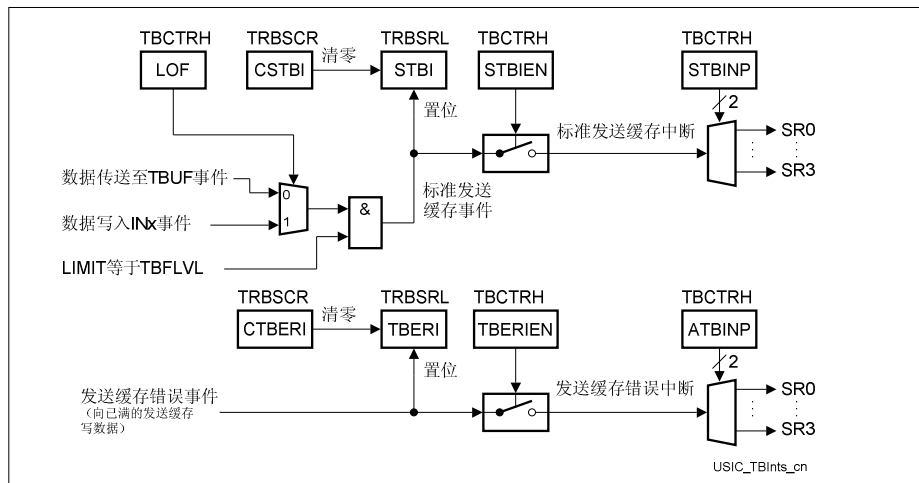


图 21-20 发送缓存事件

接收 FIFO 缓存机制检测到以下事件时可产生中断（若被使能）。标准接收缓存事件和备选接收缓存事件可设定给两种不同的模式：一种模式和接收缓存的填充级有关；另一种模式和 OUTRL 中可用的数据字的接收控制信息 RCI 的某位有关。



若中断产生和接收 FIFO 缓存的填充级有关，则只使用标准接收缓存事件，不使用备选接收缓存事件。可选择该模式指示已接收到一批数据，不考虑相关 RCI 的内容。

若中断产生和 RCI 有关，则不考虑填充级。OUTRL 中每次有新数据可用时，检测到一个事件。若位 RCI[4] = 0，产生标准接收缓存事件，否则产生备选接收缓存事件（位 RCI[4] = 1）。根据所选协议及 RBCTRH.RCIM 的设置，RCI[4] 可包含不同的信息、用于特定协议的中断处理（具体请参阅协议章节的描述）。

- 填充级模式下的标准接收缓存事件（RBCTRH.RNM = 0）：

接收缓存的填充级（由 TRBSRH.RBFLVL 给出）超过（RBCTRH.LOF = 1）或低于（RBCTRH.LOF = 0）设定的极限值（RBCTRL.LIMIT）。从等于到小于或大于的转换操作（并非小于或大于本身）将触发该事件。

若使用标准接收缓存事件指示新数据必须从 OUTRL 中读出，应设置 RBCTRH.LOF = 1。

- RCI 模式下的标准接收缓存事件（RBCTRH.RNM = 1）：

由新数据值更新 OUTR，RCI[4] = 0。

- 填充级模式下的备选接收缓存事件（RBCTRH.RNM = 0）：不使用

- RCI 模式下的备选接收缓存事件（RBCTRH.RNM = 1）：

由新数据值更新 OUTR，RCI[4] = 1。

- 接收缓存错误事件：

软件从已空的缓存读数据。读数据无效。

填充级模式下的接收缓存事件和中断如 [图 21-21](#) 所示。

*注：在填充级模式下，只有当填充级从极限值变为其它值时才能产生缓存事件。若填充级不从极限值开始变化，不产生触发事件。*

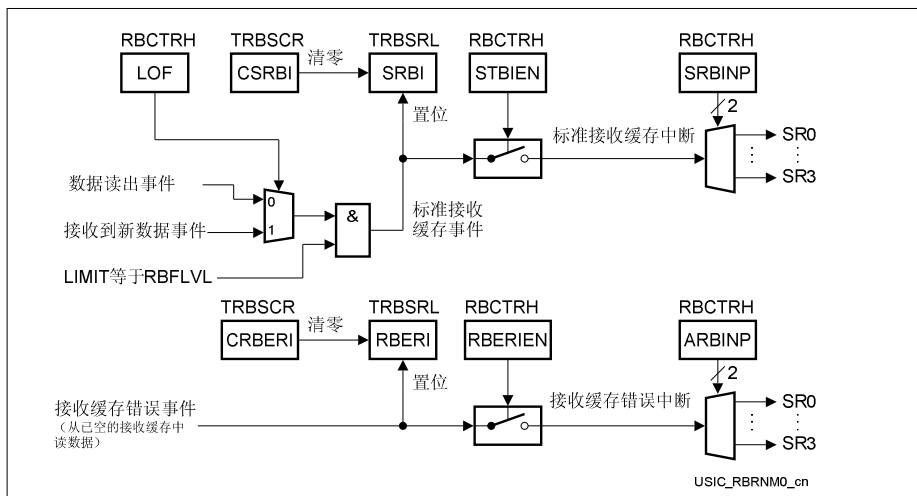


图 21-21 填充模式下的接收缓存事件

RCI 模式下的接收缓存事件和中断如图 21-22 所示。

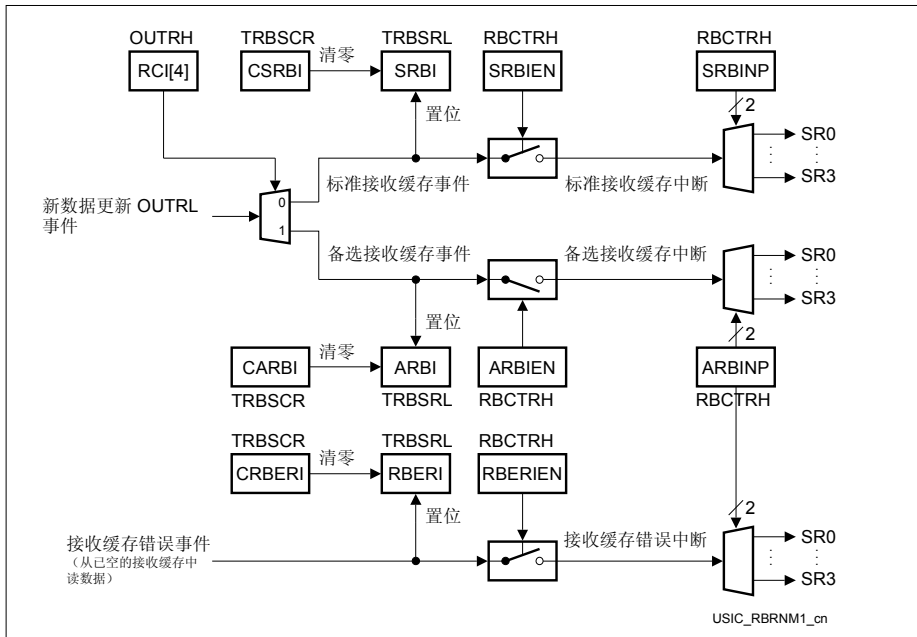


图 21-22 RCI 模式下的接收缓存事件

用于指示缓存事件以及控制相关中断产生（发送和接收）的各寄存器、寄存器指示/控制位和位域归纳见表 21-7。

表 21-7 缓存事件和中断处理

事件	指示标志位	标志清零控制位	中断使能控制位	SRx 输出选择控制位
标准发送缓存事件	TRBSRL. STBI	TRBSCR. CSTBI	TBCTRH. STBIEN	TBCTRH. STBINP
发送缓存错误事件	TRBSRL. TBERI	TRBSCR. CTBERI	TBCTRH. TBERIEN	TBCTRH. ATBINP
标准接收缓存事件	TRBSRL. SRBI	TRBSCR. CSRBI	RBCTRH. SRBIEN	RBCTRH. SRBINP
备选接收缓存事件	TRBSRL. ARBI	TRBSCR. CARBI	RBCTRH. ARBIEN	RBCTRH. ARBINP

事件	指示标志位	标志清零控制位	中断使能控制位	SRx 输出选择控制位
接收缓存错误事件	TRBSRL. RBERI	TRBSCR. CRBERI	RBCTRH. RBERIEN	RBCTRH. ARBINTXDP

### 21.2.13.3 FIFO 缓存旁路

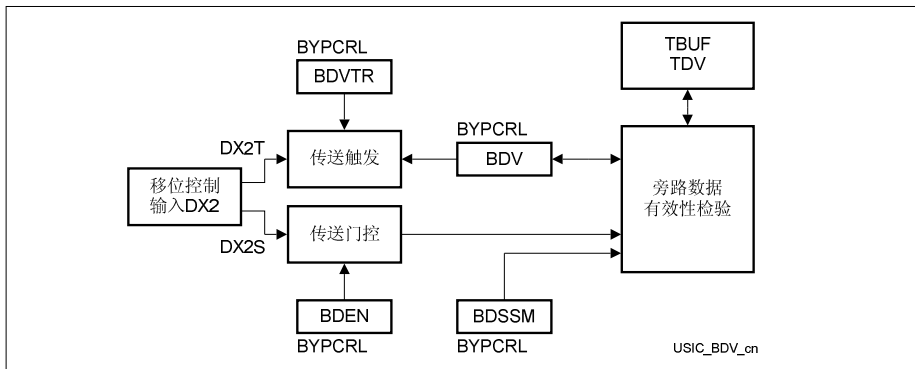
数据旁路机制是发送 FIFO 控制模块的一部分。通过数据旁路可在数据流中插入数据字（以发送高优先级的信息）而不改变发送 FIFO 缓存的内容。旁路结构由寄存器 BYP 中的旁路数据字（最长 16 位）和寄存器 BYPCRL 和 BYPCRH 中的一些相关控制信息构成。这些控制位定义了旁路数据的字长、配置传送触发和门控机制（和对发送缓存 TBUF 的配置相似）。

旁路数据字可通过位 BYPCRL.BDV（旁路数据有效）标记为有效或无效。和数据流以及和事件相关的标准共同决定了旁路数据字是否有效可进行发送。数据有效性检验逻辑检查开始传送该数据字的条件。根据检查结果向发送缓存寄存器 TBUF 加载不同的值，具体规则如下：

- 只有当 TCSRL.TDV = 0（TBUF 已空）时，才能将发送 FIFO 缓存中的数据或旁路数据传送到 TBUF 中。
- 只有当旁路使能（由 BYPCRL.BDEN 控制）或满足选定的门控条件时，才能将旁路数据传送到 TBUF 中。
- 若旁路数据有效，其发送优先级高于 FIFO 数据、或发送 FIFO 已空，将旁路数据传送到 TBUF 中。
- 若旁路数据有效，其发送优先级低于包含有效数据的 FIFO 缓存，将最早的 FIFO 数据传送到 TBUF 中。
- 若旁路数据无效，FIFO 缓存中包含有效数据，将最早的 FIFO 数据传送到 TBUF 中。
- 若旁路数据无效且 FIFO 缓存中不包含有效数据，TBUF 不变。

旁路数据有效性检验的逻辑框图如图 21-23 所示。

- 传送门控逻辑使能/禁止将旁路数据字传送到 TBUF 中（软件或硬件控制）。若数据移位不需使用输入级 DX2，信号 DX2S 可用于门控控制。由位域 BYPCRL.BDEN 控制传送门控逻辑。
- 传送触发逻辑支持由事件（如定时器事件或输入引脚上的事件）触发数据字传送。若数据移位不需使用输入级 DX2，信号 DX2T 可用于触发控制。由位 BYPCRL.BDVTR 控制传送触发逻辑。
- 旁路数据有效性检验逻辑的输入来自门控逻辑、触发逻辑和 TCSRL.TDV。



**图 21-23 旁路数据有效性检验**

基于这样的结构，可实现以下旁路数据传送功能：

- 采用单次传送机制时，必须设置位 **BYPCTRL.BDSSM = 1**。每次将旁路数据字传送到 **TBUF** 中之后，旁路数据字必须重新标记为有效。这可通过将新的旁路数据写入 **BYP**、或 **BDVTR = 1** 时由有效的 **DX2T** 信号（由定时器事件或输入引脚上的事件触发）来实现。
- 若旁路数据恒有效，必须设置位 **BYPCTRL.BDSSM = 0**（作为数据 **FIFO** 为空时的备选数据）。

### 21.2.13.4 FIFO 访问限制

共享 **FIFO** 缓存区的数据可由软硬件访问。硬件访问用于每路通信通道的数据传送（发送和接收）；软件访问用于读出接收数据或写入发送数据。因此，**FIFO** 机制可限制数据传输率。硬件对 **FIFO** 缓存的访问优先级较高，出现访问冲突时软件访问延迟进行。

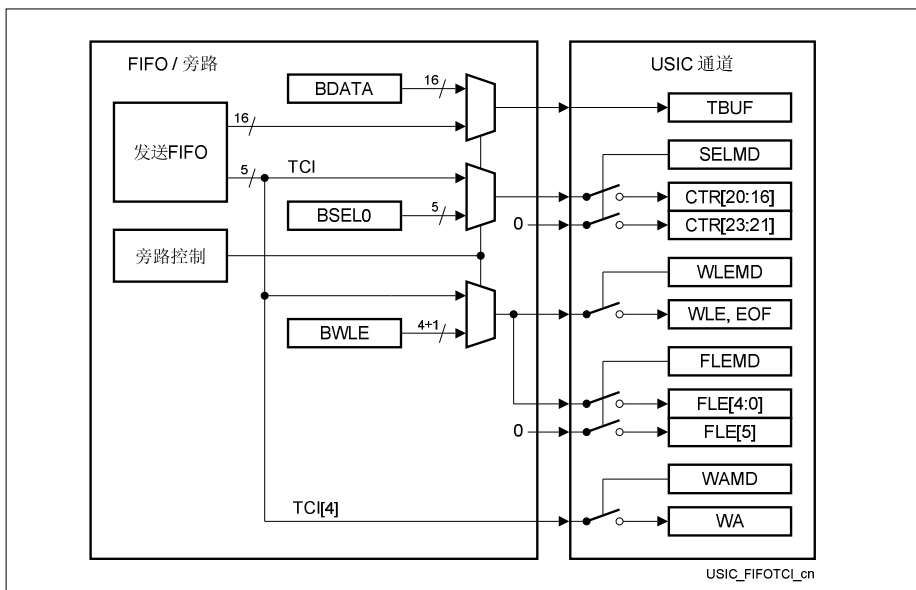
为了避免数据丢失和 **CPU** 停滞（由延迟的软件访问造成），必须将波特率、字长和软件访问这些因素考虑在内。软件或硬件每次访问 **FIFO** 缓存会占用 1 个  $f_{sys}$  时钟周期。特别是，由极短的、连续的数据字构成的数据流可导致 **FIFO** 访问限制。

### 21.2.13.5 FIFO 发送控制信息的处理

除发送数据之外，发送控制信息 **TCI** 也可从发送 **FIFO** 或旁路结构传送至 **USIC** 通道。根据所选协议以及使能的更新机制，可修改部分 **USIC** 通道参数的设定。当 **FIFO** 数据或旁路数据载入 **TBUF** 时，利用 **TCI** 或旁路控制信息修改通道参数。

- **TCSRL.SELMD = 1**：由 **FIFO TCI** 或 **BYPCTRL.BSELO** 更新 **PCR.H.CTR[20:16]**，同时清除 **PCR.H.CTR[23:21]**

- TCSRL.WLEMD = 1: 由 FIFO TCI 或 BYPCRL.BWLE 更新 SCTRH.WLE 和 TCSRL.EOF (若 WLE 由 TCI 或 BWLE 覆盖, 用户须注意: FLE 应被相应设置)
- TCSRL.FLEMD = 1: 由 FIFO TCI 或 BYPCRL.BWLE 更新 SCTRH.FLE[4:0], 同时清除 SCTRH.FLE[5]
- TCSRL.WAMD = 1: 由 FIFO TCI[4]更新 TCSRL.WA



**图 21-24 FIFO/旁路结构 TCI 处理**

## 21.2.14 FIFO 缓存和旁路寄存器

### 21.2.14.1 旁路寄存器

向旁路数据寄存器写数据的操作（至少写入其低位字节）置位 BYPCRL.BDV（旁路数据标记为有效）。

#### BYP

旁路数据寄存器 (100<sub>H</sub>) 复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
BDATA	[15:0]	rw	旁路数据 该位域保存旁路数据。

#### BYPCRL

旁路控制寄存器 L (104<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BDV	0	B PRIO	BD VTR	BDEN	0	BD SSM	0					BWLE			
rh	r	rw	rw	rw	r	rw	r					rw			

符号	位序号	类型	功能描述
BWLE	[3:0]	rw	旁路字长 该位域定义旁路数据的长度，字长为 BWLE +1。 数据字在数据缓存中始终右对齐，即位于 [BWLE:0]。 始终将旁路数据看作是独立的一帧数据，长度为 BWLE。 和 SCTRH.WLE 的编码相同。
BDSSM	8	rw	旁路数据单次模式 该位定义旁路数据字恒有效还是只传送一次（单次

符号	位序号	类型	功能描述
			模式)。 <b>0<sub>B</sub></b> 旁路数据载入 TBUF 之后仍视为有效, 该操作不会清除 BDV。 <b>1<sub>B</sub></b> 旁路数据载入 TBUF 之后被视为无效, 该操作将清除 BDV。
<b>BDEN</b>	[11:10]	rw	<b>旁路数据使能</b> 该位域定义是否使能旁路数据传送、以及在什么情况下允许将旁路数据传送给 TBUF。 <b>00<sub>B</sub></b> 禁止传送旁路数据。 <b>01<sub>B</sub></b> 可将旁路数据传送给 TBUF。若 BDV = 1, 可根据优先级传送旁路数据。 <b>10<sub>B</sub></b> 使能门控旁路数据传送。若 BDV = 1 且 DX2S = 0, 可根据优先级将旁路数据传送给 TBUF。 <b>11<sub>B</sub></b> 使能门控旁路数据传送。若 BDV = 1 且 DX2S = 1, 可根据优先级将旁路数据传送给 TBUF。
<b>BDVTR</b>	12	rw	<b>旁路数据有效触发</b> 若 DX2T 有效, 该位使能旁路数据标记为有效。 <b>0<sub>B</sub></b> 位 BDV 不受 DX2T 的控制。 <b>1<sub>B</sub></b> DX2T 有效时位 BDV 置位。
<b>BPRI0</b>	13	rw	<b>旁路优先级</b> 该位定义旁路数据和发送 FIFO 数据的优先级。 <b>0<sub>B</sub></b> 发送 FIFO 数据的优先级高于旁路数据。 <b>1<sub>B</sub></b> 旁路数据的优先级高于发送 FIFO 数据。
<b>BDV</b>	15	rh	<b>旁路数据有效</b> 该位定义旁路数据是否有效。向寄存器 BYP 写入数据后 (至少写入其低位字节) 该位被自动置位。可通过置位 TRBSCR.CBDV 对该位软件清零。 <b>0<sub>B</sub></b> 旁路数据无效。 <b>1<sub>B</sub></b> 旁路数据有效。



符号	位序号	类型	功能描述
<b>0</b>	[7:4], 9,14	r	<b>保留</b> 读操作返回 0；应写入 0。

## **BYPCRH**

### **旁路控制寄存器 H**

**(106<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>											<b>BSELO</b>				
r											rw				

符号	位序号	类型	功能描述
<b>BSELO</b>	[4:0]	rw	<b>旁路选择输出</b> 若 TCSRL.SELMD = 1，旁路数据传送给 TBUF 时，该位域的值将写入 PCRH.CTR[20:16]。 在 SSC 协议中，该位域可用来定义发送旁路数据时将激活哪路 SELOx 输出。
<b>0</b>	[15:5]	r	<b>保留</b> 读操作返回 0；应写入 0。

## **21.2.14.2 FIFO 缓存控制寄存器**

UxCy 发送和接收 FIFO 的状态信息由寄存器 UxCy\_TRBSRL/H 给出。

只有当 CCFG.TB = 1 使能发送 FIFO 缓存功能时，才能对该寄存器中和发送 FIFO 相关的位进行设置，否则写操作被忽略。只有当 CCFG.TB = 0 使能接收 FIFO 缓存功能时，才能对该寄存器中和接收 FIFO 相关的位进行设置，否则写操作被忽略。

可通过对寄存器 TRBSCR 中的位置 1，相应清除发送和接收 FIFO 状态寄存器 TRBSRL 中的中断标志（事件标志）；置 0 对这些位无影响。软件置位 SRBI、RBERI、ARBI、STBI 或 TBERI 可用来仿真检测到发送/接收缓存事件，但不激活任何服务请求输出（故，参见 FMR.SIOx）。

位 TBUS 和 RBUS 用于测试，数据处理程序可忽略这两位。需要注意：读取这两位可返回 0 或 1。建议不要对其操作（“不关心”）。

**TRBSRL**

**发送/接收缓存状态寄存器 L**

**(118<sub>H</sub>)**

**复位值: 0808<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	T BUS	T FUL L	T EMP TY	0	TB ERI	ST BI	0	R BUS	R FUL L	R EMP TY	AR BI	RB ERI	SR BI		
r	rh	rh	rh	r	rwh	rwh	r	rh	rh	rh	rwh	rwh	rwh		

符号	位序号	类型	功能描述
<b>SRBI</b>	0	rwh	<b>标准接收缓存事件</b> 该位指示已检测到标准接收缓存事件。通过设置 TRBSCR.CSRBI = 1 清除该标志位。 若 RBCTRH.SRBIEN 使能中断，检测到标准接收缓存事件后，激活由 RBCTRH.SRBINP 选定的服务请求输出 SRx。 0 <sub>B</sub> 未检测到标准接收缓存事件 1 <sub>B</sub> 已检测到标准接收缓存事件
<b>RBERI</b>	1	rwh	<b>接收缓存错误事件</b> 该位指示已检测到接收缓存错误事件。通过设置 TRBSCR.CRBERI = 1 清除该标志位。 若 RBCTRH.RBERIEN 使能中断，检测到接收缓存错误事件后，激活由 RBCTRH.ARBINP 选定的服务请求输出 SRx。 0 <sub>B</sub> 未检测到接收缓存错误事件 1 <sub>B</sub> 已检测到接收缓存错误事件
<b>ARBI</b>	2	rwh	<b>备选接收缓存事件</b> 该位指示已检测到备选接收缓存事件。通过设置 TRBSCR.CARBI = 1 清除该标志位。 若 RBCTRH.ARBIEN 使能中断，检测到备选接收缓存事件后，激活由 RBCTRH.ARBINP 选定的服务请求输出 SRx。 0 <sub>B</sub> 未检测到备选接收缓存事件 1 <sub>B</sub> 已检测到备选接收缓存事件

符号	位序号	类型	功能描述
<b>EMPTY</b>	3	rh	<b>接收缓存已空</b> 该位指示接收缓存是否为空。 0 <sub>B</sub> 接收缓存未空 1 <sub>B</sub> 接收缓存已空
<b>RFULL</b>	4	rh	<b>接收缓存已满</b> 该位指示接收缓存是否已满。 0 <sub>B</sub> 接收缓存未满 1 <sub>B</sub> 接收缓存已满
<b>RBUS</b>	5	rh	<b>接收缓存正忙</b> 该位指示接收缓存是否正被 FIFO 处理器更新。 0 <sub>B</sub> 接收缓存的信息已更新完毕 1 <sub>B</sub> OUTRL/H 正被 FIFO 更新。OUTRL/H 的读取操作将延迟进行，FIFO 指针（自前次读取）还未被更新。
<b>STBI</b>	8	rwh	<b>标准发送缓存事件</b> 该位指示已检测到标准发送缓存事件。通过设置 TRBSCR.CSTBI = 1 清除该标志位。 若 TBCTRH.SRBIEN 使能中断，检测到标准发送缓存事件后，激活由 TBCTRH.SRBINP 选定的服务请求输出 SRx。 0 <sub>B</sub> 未检测到标准发送缓存事件 1 <sub>B</sub> 已检测到标准发送缓存事件
<b>TBERI</b>	9	rwh	<b>发送缓存错误事件</b> 该位指示已检测到发送缓存错误事件。通过设置 TRBSCR.CTBERI = 1 清除该标志位。 若 TBCTRH.TBERIEN 使能中断，检测到发送缓存错误事件后，激活由 TBCTRH.ATBINP 选定的服务请求输出 SRx。 0 <sub>B</sub> 未检测到发送缓存错误事件 1 <sub>B</sub> 已检测到发送缓存错误事件
<b>TEMPY</b>	11	rh	<b>发送缓存已空</b>

符号	位序号	类型	功能描述
			该位指示发送缓存是否为空。 0 <sub>B</sub> 发送缓存未空 1 <sub>B</sub> 发送缓存已空
<b>TFULL</b>	12	rh	<b>发送缓存已满</b> 该位指示发送缓存是否已满。 0 <sub>B</sub> 发送缓存未满 1 <sub>B</sub> 发送缓存已满
<b>TBUS</b>	13	rh	<b>发送缓存正忙</b> 该位指示发送缓存是否正被 FIFO 处理器更新。 0 <sub>B</sub> 发送缓存的信息已更新完毕 1 <sub>B</sub> 向 INx 写入数据之后 FIFO 正在更新。向 INx 写数据的操作将延迟进行，FIFO 指针（自前次写入 INx）还未被更新。
<b>0</b>	[7:6], 10, [15:14]	r	<b>保留</b> 读操作返回 0；应写入 0。

### TRBSRH

发送/接收缓存状态寄存器 H

(11A<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	TBFLVL							0	RBFLVL						
r	rh							r	rh						

符号	位序号	类型	功能描述
<b>RBFLVL</b>	[6:0]	rh	<b>接收缓存填充级</b> 该位域指示接收缓存的填充级，填充级从 0 开始（对应缓存为空的情况）。
<b>TBFLVL</b>	[14:8]	rh	<b>发送缓存填充级</b> 该位域指示发送缓存的填充级，填充级从 0 开始（对应缓存为空的情况）。

符号	位序号	类型	功能描述
<b>0</b>	7,15	r	保留 读操作返回 0；应写入 0。

寄存器 TRBSCR 用于清除寄存器 TRBSRL 中的标志位，或用于清除发送或接收 FIFO 缓存的内容。读取该寄存器始终返回 0。

## TRBSCR

发送/接收缓存状态清零寄存器

(11C<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FLU SH TB	FLU SH RB	0			C BDV	C TB ERI	C ST BI	0					C AR BI	C RB ERI	C SR BI
w	w	r			w	w	w	r					w	w	w

符号	位序号	类型	功能描述
<b>CSRBI</b>	0	w	清除标准接收缓存事件 0 <sub>B</sub> 无影响 1 <sub>B</sub> 清除 TRBSRL.SRBI
<b>CRBERI</b>	1	w	清除接收缓存错误事件 0 <sub>B</sub> 无影响 1 <sub>B</sub> 清除 TRBSRL.RBERI
<b>CARBI</b>	2	w	清除备选接收缓存事件 0 <sub>B</sub> 无影响 1 <sub>B</sub> 清除 TRBSRL.ARBI
<b>CSTBI</b>	8	w	清除标准接收缓存事件 0 <sub>B</sub> 无影响 1 <sub>B</sub> 清除 TRBSRL.STBI
<b>CTBERI</b>	9	w	清除发送缓存错误事件 0 <sub>B</sub> 无影响 1 <sub>B</sub> 清除 TRBSRL.TBERI

符号	位序号	类型	功能描述
<b>CBDV</b>	10	w	清除旁路数据有效 0 <sub>B</sub> 无影响 1 <sub>B</sub> 清除 BYPCRL.BDV
<b>FLUSHRB</b>	14	w	清除接收 FIFO 缓存 0 <sub>B</sub> 无影响 1 <sub>B</sub> 清除接收 FIFO 缓存（填充级清零、输出指针设置为输入指针值）。只有当 FIFO 缓存不参与数据通信时才能使用该位。
<b>FLUSHTB</b>	15	w	清除发送 FIFO 缓存 0 <sub>B</sub> 无影响 1 <sub>B</sub> 清除发送 FIFO 缓存（填充级清零、输出指针设置为输入指针值）。只有当 FIFO 缓存不参与数据通信时才能使用该位。
<b>0</b>	[7:3], [13:11]	r	保留 读操作返回 0；应写入 0。

### 21.2.14.3 发送 FIFO 缓存控制寄存器

发送 FIFO 缓存由寄存器 TBCTRL 和 TBCTRH 控制。只有当 CCFG.TB = 1 使能发送 FIFO 缓存功能时，才能对这些寄存器进行写操作，否则写操作被忽略。

#### TBCTRL

发送缓存控制寄存器 L (110<sub>H</sub>) 复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
<b>DPTR</b>	[5:0]	w	数据指针 当 FIFO 入口分配给发送 FIFO 缓存时，该位域定义发送缓存指针的起始值。读取该位域始终返回 0。当 SIZE = 0 时设置 DPTR，该值将更新寄存器

符号	位序号	类型	功能描述
			TRBPTRL 中的发送指针 TDIPTR 和 RTDOPTR, 缓存看作为空。当 SIZE > 0 时, 对 DPTR 的写操作被忽略, 不会修改指针值。
<b>LIMIT</b>	[13:8]	rw	<b>用于产生中断的填充级极限值</b> 该位域定义发送 FIFO 缓存的目标填充级, 利用该值检测标准发送缓存事件。
<b>0</b>	[7:6], [15:14]	r	<b>保留</b> 读操作返回 0; 应写入 0。

### TBCTRH

**发送缓存控制寄存器 H**

**(112<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>TB ERI EN</b>	<b>ST BI EN</b>	<b>0</b>	<b>LOF</b>	<b>0</b>	<b>SIZE</b>	<b>0</b>	<b>ATBINP</b>	<b>0</b>	<b>STBINP</b>						
rw	rw	r	rw	r	rw	r	rw	r	rw	r	rw				

符号	位序号	类型	功能描述
<b>STBINP</b>	[1:0]	rw	<b>标准发送缓存中断节点指针</b> 该位域定义产生标准发送缓存事件时激活哪路服务请求输出 SRx。  00 <sub>B</sub> 激活输出 SR0 01 <sub>B</sub> 激活输出 SR1 10 <sub>B</sub> 激活输出 SR2 11 <sub>B</sub> 激活输出 SR3
<b>ATBINP</b>	[4:3]	rw	<b>备选发送缓存中断节点指针</b> 该位域定义产生发送缓存错误事件时激活哪路服务请求输出 SRx。  00 <sub>B</sub> 激活输出 SR0 01 <sub>B</sub> 激活输出 SR1 10 <sub>B</sub> 激活输出 SR2

符号	位序号	类型	功能描述
			11 <sub>B</sub> 激活输出 SR3
<b>SIZE</b>	[10:8]	rw	<b>缓存大小</b> 该位域定义分配给发送 FIFO 缓存的 FIFO 入口个数。 000 <sub>B</sub> FIFO 机制禁用。FIFO 缓存不接受任何数据请求。 001 <sub>B</sub> FIFO 缓存包含 2 个入口。 010 <sub>B</sub> FIFO 缓存包含 4 个入口。 011 <sub>B</sub> FIFO 缓存包含 8 个入口。 100 <sub>B</sub> FIFO 缓存包含 16 个入口。 101 <sub>B</sub> FIFO 缓存包含 32 个入口。 110 <sub>B</sub> FIFO 缓存包含 64 个入口。 111 <sub>B</sub> 保留。
<b>LOF</b>	12	rw	<b>基于极限溢出的缓存事件</b> 该位定义填充级和设定的极限值之间的关系，基于这种关系产生标准发送缓存事件。 0 <sub>B</sub> 当填充级等于极限值、且由于发送数据字造成填充级变小时，产生标准发送缓存事件。 1 <sub>B</sub> 当填充级等于极限值、且由于向数据输入单元 INx 写入数据造成填充级变大时，产生标准发送缓存事件。
<b>STBIEN</b>	14	rw	<b>标准发送缓存中断使能</b> 该位使能/禁止基于标准发送缓存事件产生标准发送缓存中断。 0 <sub>B</sub> 禁止产生标准发送缓存中断。 1 <sub>B</sub> 允许产生标准发送缓存中断。
<b>TBERIEN</b>	15	rw	<b>发送缓存错误中断使能</b> 该位使能/禁止基于发送缓存错误事件（软件向已满的发送缓存写数据）产生发送缓存错误中断。 0 <sub>B</sub> 禁止产生发送缓存错误中断。 1 <sub>B</sub> 允许产生发送缓存错误中断。



符号	位序号	类型	功能描述
<b>0</b>	2,[7:5], 11,13	r	保留 读操作返回 0；应写入 0。

#### 21.2.14.4 接收 FIFO 缓存控制寄存器

接收 FIFO 缓存由寄存器 RBCTRL 和 RBCTRH 控制。只有当 CCFG.RB = 1 使能接收 FIFO 缓存功能时，才能对这些寄存器进行写操作，否则写操作被忽略。

#### RBCTRL

接收缓存控制寄存器 L

(114<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>		<b>LIMIT</b>						<b>0</b>		<b>DPTR</b>					
r		rw						r		w					

符号	位序号	类型	功能描述
<b>DPTR</b>	[5:0]	w	<b>数据指针</b> 当 FIFO 入口分配给接收 FIFO 缓存时，该位域定义接收缓存指针的起始值。读取该位域始终返回 0。当 SIZE = 0 时设置 DPTR，该值将更新寄存器 TRBPTRH 中的发送指针 RDIPTTR 和 RDOPTTR，缓存看作为空。当 SIZE > 0 时，对 DPTR 的写操作被忽略，不会修改指针值。
<b>LIMIT</b>	[13:8]	rw	<b>用于产生中断的填充级极限值</b> 该位域定义接收 FIFO 缓存的目标填充级，利用该值检测标准接收缓存事件。
<b>0</b>	[7:6], [15:14]	r	保留 读操作返回 0；应写入 0。

**RBCTRH**

**接收缓存控制寄存器 H**

**(116<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>RB ERI EN</b>	<b>SR BI EN</b>	<b>AR BI EN</b>	<b>LOF</b>	<b>RNM</b>	<b>SIZE</b>		<b>RCIM</b>	<b>0</b>	<b>ARBINP</b>	<b>0</b>	<b>SRBINP</b>				
rw	rw	rw	rw	rw	rw		rw	r	rw	r	rw				

符号	位序号	类型	功能描述
<b>STBINP</b>	[1:0]	rw	<b>标准接收缓存中断节点指针</b> 该位域定义产生标准接收缓存事件时激活哪路服务请求输出 SRx。 00 <sub>B</sub> 激活输出 SR0 01 <sub>B</sub> 激活输出 SR1 10 <sub>B</sub> 激活输出 SR2 11 <sub>B</sub> 激活输出 SR3
<b>ATBINP</b>	[4:3]	rw	<b>备选接收缓存中断节点指针</b> 该位域定义产生备选接收缓存事件时激活哪路服务请求输出 SRx。 00 <sub>B</sub> 激活输出 SR0 01 <sub>B</sub> 激活输出 SR1 10 <sub>B</sub> 激活输出 SR2 11 <sub>B</sub> 激活输出 SR3
<b>RCIM</b>	[7:6]	rw	<b>接收控制信息模式</b> 该位域定义接收状态寄存器 RBUFSR 中的哪些信息作为 5 位接收控制信息 RCI[4:0]传送给接收 FIFO 缓存，可从寄存器 OUT(D)RH 中读出。 00 <sub>B</sub> RCI[4] = PERR, RCI[3:0] = WLEN 01 <sub>B</sub> RCI[4] = SOF, RCI[3:0] = WLEN 10 <sub>B</sub> RCI[4] = 0, RCI[3:0] = WLEN 11 <sub>B</sub> RCI[4] = PERR, RCI[3] = PAR, RCI[2:1] = 00 <sub>B</sub> , RCI[0] = SOF

符号	位序号	类型	功能描述
<b>SIZE</b>	[10:8]	rw	<b>缓存大小</b> 该位域定义分配给接收 FIFO 缓存的 FIFO 入口个数。  000 <sub>B</sub> FIFO 机制禁用。FIFO 缓存不接受任何数据请求。  001 <sub>B</sub> FIFO 缓存包含 2 个入口。 010 <sub>B</sub> FIFO 缓存包含 4 个入口。 011 <sub>B</sub> FIFO 缓存包含 8 个入口。 100 <sub>B</sub> FIFO 缓存包含 16 个入口。 101 <sub>B</sub> FIFO 缓存包含 32 个入口。 110 <sub>B</sub> FIFO 缓存包含 64 个入口。 111 <sub>B</sub> 保留。
<b>RNM</b>	11	rw	<b>接收器通知模式</b> 该位定义接收缓存事件的模式。接收缓存错误事件不受 RNM 的影响。  0 <sub>B</sub> 填充级模式： 当填充级等于极限值、且由于读取 OUTRL (LOF = 0) 或接收新数据字 (LOF = 1) 造成填充级改变时，产生标准接收缓存事件。  1 <sub>B</sub> RCI 模式： 若 OUTRH.RCI[4] = 0，寄存器 OUTRL 被更新时产生标准接收缓存事件。 若 OUTRH.RCI[4] = 1，寄存器 OUTRL 被更新时产生备选接收缓存事件。
<b>LOF</b>	12	rw	<b>基于极限溢出的缓存事件</b> 该位定义填充级和设定的极限值之间的关系，基于这种关系，在填充级模式下 (RNM = 0) 产生标准接收缓存事件。在 RCI 模式下 (RNM = 1) 忽略 LIMIT 和 LOF。  0 <sub>B</sub> 当填充级等于极限值、且由于读取 OUTRL 造成填充级变小时，产生标准接收缓存事件。  1 <sub>B</sub> 当填充级等于极限值、且由于接收新数据造成填充级变大时，产生标准接收缓存事件。

符号	位序号	类型	功能描述
<b>ARBIEN</b>	13	rw	<b>备选接收缓存中断使能</b> 该位使能/禁止基于备选接收缓存事件产生备选接收缓存中断。 0 <sub>B</sub> 禁止产生备选接收缓存中断。 1 <sub>B</sub> 允许产生备选接收缓存中断。
<b>SRBIEN</b>	14	rw	<b>标准接收缓存中断使能</b> 该位使能/禁止基于标准接收缓存事件产生标准接收缓存中断。 0 <sub>B</sub> 禁止产生标准接收缓存中断。 1 <sub>B</sub> 允许产生标准接收缓存中断。
<b>RBERIEN</b>	15	rw	<b>接收缓存错误中断使能</b> 该位使能/禁止基于接收缓存错误事件（软件从已空的缓存读数据）产生接收缓存错误中断。 0 <sub>B</sub> 禁止产生接收缓存错误中断。 1 <sub>B</sub> 允许产生接收缓存错误中断。
<b>0</b>	2,[7:5], 11,13	r	<b>保留</b> 读操作返回 0；应写入 0。

#### 21.2.14.5 FIFO 缓存数据寄存器

32 个独立的数据输入单元 IN00 - IN31 可用作发送 FIFO 缓存的数据入口地址单元。写入其中某一单元的数据将保存在发送 FIFO 缓存中。此外，数据入口单元地址编号[31:0]对应的 5 位编码代表发送控制信息 TCI。

若向已满的 FIFO 写入新数据，写操作被忽略，产生发送缓存错误事件。

**Inx (x = 00-31)**

**发送 FIFO 缓存输入单元 x**                      **(180<sub>H</sub> + x\*4)**                      **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>TDATA</b>															

w

符号	位序号	类型	功能描述
<b>TDATA</b>	[15:0]	w	<b>发送数据</b> 该位域保存将要发送的数据。读取返回 0。 写入 TDATA 的数据（至少写入其低位字节）将保存在 FIFO 中。

接收 FIFO 缓存输出寄存器 OUTRL 给出 FIFO 缓存最早接收到的数据。读取 OUTRL 返回接收数据。一旦读取 OUTRL（至少读取其低位字节），数据被声明已读、下一个数据将出现。寄存器 OUTRH 中存放接收控制信息 RCI（具体信息由 RBCTRH.RCIM 选择）。写访问 OUTRL/H 被忽略。

### OUTRL

**接收缓存输出寄存器 L** (120<sub>H</sub>) **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>DSR</b>															
rh															

符号	位序号	类型	功能描述
<b>DSR</b>	[15:0]	rh	<b>接收数据</b> 该位域监控接收 FIFO 中最早的数据内容。 一旦读取 OUTRL（至少读取其低位字节），将释放当前的缓存入口。

### OUTRH

**接收缓存输出寄存器 H** (122<sub>H</sub>) **复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>												<b>RCI</b>			
r												rh			

符号	位序号	类型	功能描述
<b>RCI</b>	[4:0]	rh	<b>接收控制信息</b> 该位域监控和 DSR 相关的接收控制信息。RCI 的位结构由位域 RBCTRH.RCIM 决定。

符号	位序号	类型	功能描述
<b>0</b>	[15:5]	r	保留 读操作返回 0；应写入 0。

若要使用调试器监控 FIFO 缓存中的接收数据，必须使 FIFO 机制失效以确保数据的一致性。因此，可使用另外一组寄存器，调试专用接收缓存寄存器 OUTDRL/H（D 代表调试器），它们和 OUTRL/H 相似，差别仅在于无 FIFO 缓存。调试器可从 OUTDRL/H 中读取数据（以监控接收数据流）而不干扰接收序列。写访问 OUTDRL/H 被忽略。

### OUTDRL

**调试专用接收缓存输出寄存器 L (124<sub>H</sub>)** **复位值: 0000<sub>H</sub>**

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

符号	位序号	类型	功能描述
<b>DSR</b>	[15:0]	rh	来自移位寄存器的数据 和 OUTRL.DSR 相同，但被读取后不释放缓存。

### OUTDRH

**调试专用接收缓存输出寄存器 H (126<sub>H</sub>)** **复位值: 0000<sub>H</sub>**

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

符号	位序号	类型	功能描述
<b>RCI</b>	[4:0]	rh	来自移位寄存器的接收控制信息 和 OUTRH.RCI 相同。
<b>0</b>	[15:5]	r	保留 读操作返回 0；应写入 0。

## 21.2.14.6 FIFO 缓存指针寄存器

发送和接收 FIFO 缓存所使用的指针分别存放在寄存器 TRBPTRL（用于发送缓存）和 TRBPTRH 中（用于接收缓存）中。这些指针由 FIFO 缓存机制自动更新、无需软件修改。因此，软件只能读取这些寄存器的值（如用于验证），对它们的写访问被忽略。

### TRBPTRL

发送/接收缓存指针寄存器 L (108<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	TDOPTR							0	TDIPTR						
r	rh							r	rh						

符号	位序号	类型	功能描述
TDIPTR	[5:0]	rh	发送数据输入指针 该位域指示来自 INx 地址单元的下一个发送数据所使用的缓存入口。
TDOPTR	[13:8]	rh	发送数据输出指针 该位域指示输出至 TBUF 的下一个发送数据所使用的缓存入口。
0	[7:6], [15:14]	r	保留 读操作返回 0；应写入 0。

### TRBPTRH

发送/接收缓存指针寄存器 H (10A<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	RDOPTR							0	RDIPTTR						
r	rh							r	rh						

符号	位序号	类型	功能描述
RDIPTTR	[5:0]	rh	接收数据输入指针 该位域指示来自 RBUF 的下一个接收数据所使用的缓存入口。

符号	位序号	类型	功能描述
<b>RDOPTR</b>	[13:8]	rh	<b>接收数据输出指针</b> 该位域指示输出至 OUT(D)RL 的下一个接收数据所使用的缓存入口。
<b>0</b>	[7:6], [15:14]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 21.3 异步串行通道（ASC = UART）

异步串行通道 ASC 支持异步数据帧的发送和接收、并提供硬件 LIN 支持。接收和发送相互独立、可在不同的时刻发送和接收数据帧。若 CCFG.ASC = 1（ASC 模式可用），由 CCR.MODE = 0010<sub>B</sub> 选择 ASC 模式。

本章包含以下内容：

- 信号描述（见 [页 21-93](#)）
- 帧格式（见 [页 21-95](#)）
- 位时序（见 [页 21-98](#)）
- 操作 ASC（见 [页 21-97](#)）
- 协议寄存器（见 [页 21-104](#)）
- 硬件 LIN 支持（见 [页 21-110](#)）

### 21.3.1 信号描述

发送器和接收器之间通过一条线来实现 ASC 的连接。接收输入 RXD 信号由输入级 DX0 处理。





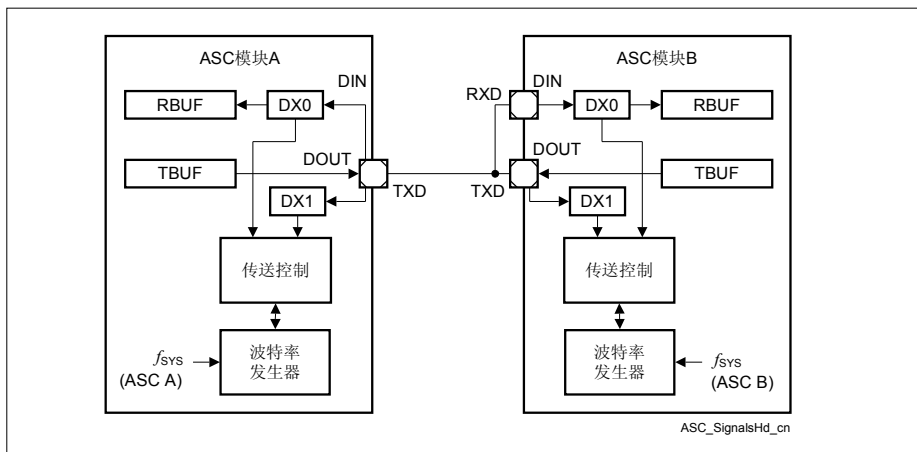
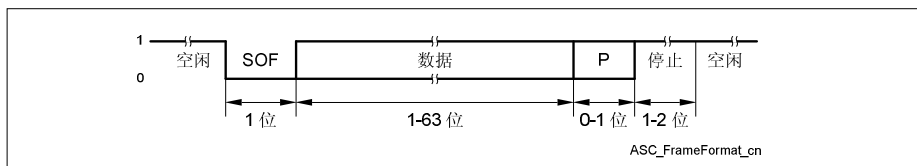


图 21-26 半双工通信的 ASC 信号连接

### 21.3.2 帧格式

标准 ASC 帧如图 21-27 所示。由以下部分组成：

- 空闲时间，信号电平为 1
- 帧起始位（SOF），信号电平为 0
- 数据域，所包含的数据位个数可编程（1-63）
- 奇偶校验位（P），奇校验或偶校验可编程。也可选择不带奇偶校验位
- 1 个或 2 个停止位，信号电平为 1



**图 21-27 标准 ASC 帧格式**

协议特定位（SOF、P、STOP）由 ASC 协议状态机自动处理，不会经接收和发送缓存存在数据流中出现。

### 21.3.2.1 空闲时间

接收器和发送器分别检查各自的数据输入线（DX0、DX1）是否空闲。空闲检测可确保 ASC 模块的 SOF 位不会和另一个 ASC 模块已在工作的数据帧发生冲突。

为了启动空闲检测，在选择 ASC 模式之前、或在 ASC 工作过程中，软件必须清除位 PSR.RXIDLE 和/或 PSR.TXIDLE。若在数据传送过程中软件清除空闲检测位，在重新启动空闲检测之前先完成当前帧的传送。只有当 PSR.RXIDLE = 1 时才能接收数据帧；只有当 PSR.TXIDLE = 1 时才能发送数据帧。空闲检测的时间由 PCRL.IDM 设定。在不发生冲突的情况下，该时间可缩短，可通过设置 PCRL.IDM = 0 声明总线空闲。

若通过 PCRL.IDM = 1 使能空闲检测，在检测到多个连续的被动位时间之后，DX0 的数据输入视为空闲（PSR.RXIDLE 变为 1）。对于发送器而言，若检测到 DX1 的数据输入上出现多个连续的被动位时间，PSR.TXIDLE 变为 1。

空闲检测所占用的时间为每帧所设定的数据位数加 2（无奇偶校验）或加 3（带有奇偶校验）。在退出终止模式之后、或 ASC 模式变为使能时，每次检测到跳变沿之后，重新开始对连续的、电平为 1 的位时间进行计数。

若空闲检测位 PSR.RXIDLE 和/或 PSR.TXIDLE 由软件清零，计数机制不会被终止（不重新开始计数）。因此，若输入线仍满足空闲条件，被清零位可立即重新置位。

需要注意：由于空闲检测是基于位时间进行的，因此最大时间可比设定值多 1 个位时间（但不少于）。

### 21.3.2.2 起始位检测

对接收输入信号 DIN（输入级 DX0 的信号）进行下降沿检测。当接收器空闲、或在停止位的采样点之后出现下降沿时，检测到 SOF 位。为了提高抗噪能力，SOF 位时序从检测到的第一个下降沿开始。若 SOF 的采样值为 1，之前的下降沿被看作是由噪声引起的，接收器被重新视为空闲。

### 21.3.2.3 数据域

数据域的长度（数据位的个数）可由位域 SCTRH.FLE 编程，由 1-63 个数据位组成，对应 SCTRH.FLE = 0 - 62（63 保留，在 ASC 模式下切勿使用）。

数据域可由多个数据字组成，如 12 位数据可由两个 8 位数据字构成，第一个数据字包含 8 位、第二个数据字包含 4 位。用户必须注意：一旦开始传送数据帧，发送数据及时可用。若发送缓存已空，将发送被动数据电平（SCTRL.PDL）。

移位方向可由 SCTRL.SDIR 编程。通过缺省设置 SDIR = 0 实现 ASC 帧的标准设置，即 LSB 移位在先。

### 21.3.2.4 奇偶校验位

ASC 可在发送时产生奇偶校验位，在接收时进行奇偶检查（逐帧进行）。奇偶校验的类型可由位域 CCR.PM 选择，发送和接收使用相同的奇偶校验类型（无校验、偶校验用户手册

或奇校验)。若奇偶校验被禁用, ASC 帧中不含奇偶校验位。出于一致性的考虑, 所有通信方必须设定相同的奇偶校验模式。

在数据域的最后一位数据之后, 发送器自动送出所计算的奇偶校验位(若奇偶校验使能)。接收器解读接收到的奇偶校验位并和内部计算出的奇偶校验位进行比较。接收到的奇偶校验值以及奇偶校验的检查结果(作为接收缓存状态信息)存放在接收缓存状态寄存器中。这些寄存器中还包含协议相关参数(PAR)和协议相关错误指示位(PERR)。

### 21.3.2.5 停止位

每帧以 1 个或 2 个信号电平为 1 的停止位结束(和空闲状态的电平相同)。停止位的个数由 PSR.STPB 编程。在停止位之后可直接传送新的起始位。

## 21.3.3 操作 ASC

为了执行 ASC 协议, 必须考虑以下因素:

- 选择 ASC 模式:

建议在 CCR.MODE = 0000<sub>B</sub> 时配置 ASC 的所有参数(在运行期间这些参数不改变)。必须设置 SCTRL.TRM = 01<sub>B</sub>。必须在 CCR.MODE = 0000<sub>B</sub> 时配置输入级, 从而避免输入信号上出现的未预计跳变, 随后可通过 CCR.MODE = 0010<sub>B</sub> 使能 ASC 模式。

- 引脚连接:

通过 DX0CR.INSW = 0 建立输入级 DX0 和接收数据输入引脚(信号 DIN)之间的连接; 配置发送数据输出引脚(信号 DOUT)。为了检测发送器的发送冲突或空闲状态, 必须将输入级 DX1 连接到选定的发送输出引脚上, 同样 DX1CR.INSW = 0。此外, 设置 DX2CR.INSW = 0。

- 位时序配置:

必须选择期望的波特率设置, 包括分数分频器、波特率发生器和位时序设置。需要注意: 实际应用不一定可同时支持所有的特性组合(如由传输延迟造成)。例如, 帧长受发送器和接收器的频差所限制。此外, 为了使用平均采样值(SMD = 1), 所选择的采样点必须考虑到信号稳定和数据传输时间。

- 数据格式配置:

根据应用需要, 通过寄存器 SCTRL 和 SCTRH 设置字长、帧长和移位方向。还可根据应用需要使数据输入和输出信号反相。此外, 必须配置奇偶校验模式(CCP.PM)。

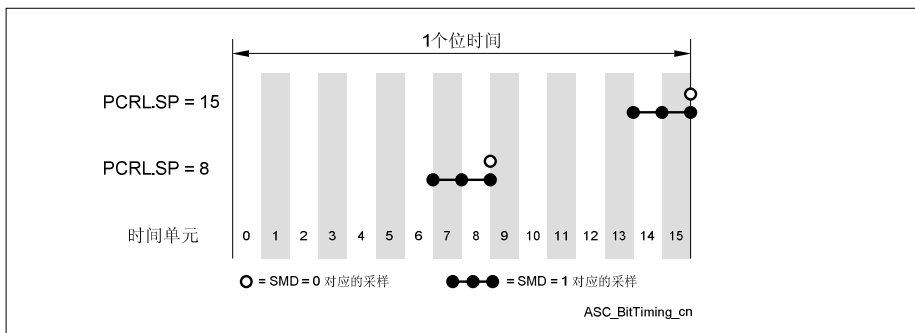
### 21.3.3.1 位时序

在 ASC 模式下，为了能够利用更小的位时间精度来调整采样点以满足应用的需求，每一位（包括协议位）被划分为多个时间单元。每一位的时间单元个数由位域 BRGL.DCTQ 定义，时间单元长度由 BRGL.PCTQ 定义。

在图 21-28 的示例中，一个位时间由 16 个时间单元组成（BRGL.DCTQ = 15）。每一位的时间单元不建议设置为小于 4。

位域 PCRL.SP 决定采样点的位置，PCRL.SP 的值切勿大于 BRGL.DCTQ。可以对每一个位时间采样一次、也可采样多次取平均值。根据 PCRL.SMD 的设置，可以对当前的输入值直接采样，也可对最近三个时间单元的采样值进行“多数判决”。标准 ASC 位时序由 16 个时间单元组成，对 8 个或 9 个时间单元之后的采样值进行“多数判决”。

发送器和接收器具有相同的位时序设立机制（时间单元个数和采样点定义）。由于接收器和发送器具有独立的位时序模块，因此它们的时间单元个数或采样点位置可以不同。数据帧的发送是与时间单元的产生同步的。



**图 21-28 ASC 位时序**

若使能冲突或空闲检测（通过 DX1 输入信号），由于必须考虑驱动延迟和一些外部延迟，因此需要仔细调整采样点的设置。只有当发送线上的位电平足够稳定时，才能设置相应的采样点位置。

若采样点位于位时间的末端，信号本身具有更长的时间变稳定，但将降低其对发送器和接收器时钟频率差异的鲁棒性。

### 21.3.3.2 波特率产生

ASC 模式下的波特率  $f_{ASC}$  和每一位的时间单元个数以及它们的时序有关。只有当发送器和接收器处于空闲状态时才能修改波特率的设置。通过寄存器 BRGL 设置波特率:

- **BRGL.CTQSEL**  
定义用于产生时间单元的输入频率  $f_{CTQIN}$
- **BRGL.PCTQ**  
定义时间单元的长度 (对  $f_{CTQIN}$  进行 1、2、3、4 分频)
- **BRGL.DCTQ**  
定义每一位中时间单元的个数

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, 波特率的计算公式如下:

$$f_{ASC} = f_{PIN} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \times \frac{1}{DCTQ + 1} \quad (21.6)$$

为了产生较低的频率, 可选用两个附加的 2 分频器, 设置  $CTQSEL = 10_B$  ( $f_{CTQIN} = f_{SCLK}$ ) 和  $PPPEN = 1$  ( $f_{PPP} = f_{MCLK}$ ), 波特率的计算公式如下:

$$f_{ASC} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \times \frac{1}{DCTQ + 1} \quad (21.7)$$

### 21.3.3.3 噪声检测

ASC 接收器始终对 DX0 的数据输入线进行噪声检测 (该检测和 PCRL.SMD 的设置无关)。若多数判决的三个输入采样值不同, 位 PSR.RNS (接收噪声) 置位。有关接收噪声的信息可在多个数据位上累积、保存在 PSR.RNS 中 (必须由软件清零), 每次检测到噪声后它可触发协议中断 (若由 PCRL.RNIEN 使能该中断)。

### 21.3.3.4 冲突检测

在某些应用中, 比如多个发送方共用一条数据线进行数据传送 (见 [图 21-26](#)), 多个发送器可能同时向同一条数据输出线 TXD 发送数据。为了避免同一时刻有多个发送器工作冲突、或者为了实现某种仲裁, 引入冲突检测机制。

从 DX1 的 TXD 输入端读出数据值并和发送数据进行比较。若通过 PCRL.CDEN = 1 使能冲突检测, 当送出的数据和读回的数据不同, 则检测到发送冲突, 位 PSR.COL 置位。若允许产生中断,  $PSR.COL = 1$  禁止发送器工作 (数据输出线变为 1), 产生协议中断。发送移位寄存器的内容被视为无效, 因此需要重新设定发送缓存。

### 21.3.3.5 脉冲整形

在某些应用中，当发送数据位为 0 时，在整个位时间内并不恒输出 0 电平，而是产生一个 0 脉冲，其余的时间单元输出 1 电平。脉冲整形不改变位时间的长度，只改变信号长度。

在标准 ASC 信令方案中，数据位为 0 时，在整个位时间内输出 0 电平（由  $PCRL.PL = 000_B$  确保）。当  $PCRL.PL > 000_B$  时，发送输出信号为 0 的脉冲长度由  $PCRL.PL$  定义的时间单元个数决定。为了能够正确接收经发送器整形的脉冲，接收器必须根据脉冲长度相应调整采样点。

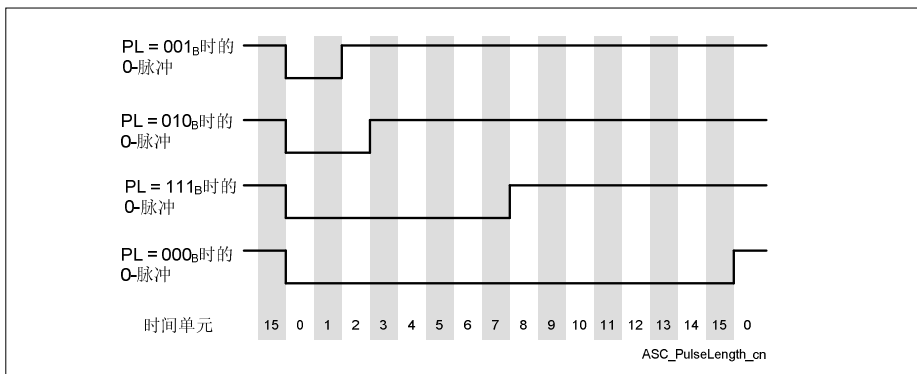


图 21-29 发送器脉冲长度控制

图 21-30 给出发送一个 8 位数据字（移位 LSB 在先、带有一个停止位）的示例（比如用于 IrDA）。发送输出信号的极性已由  $SCTRL.DOCFG = 01_B$  反相。

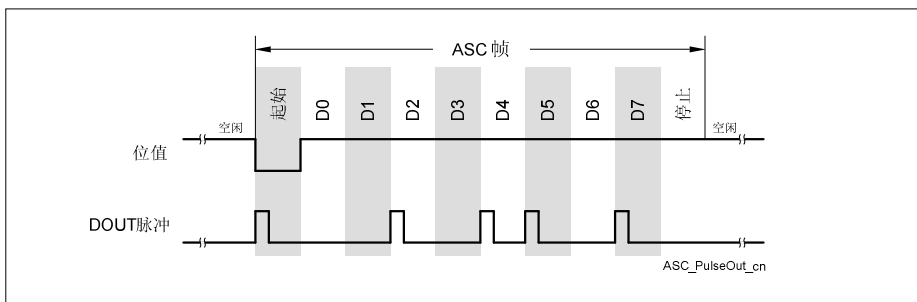


图 21-30 脉冲输出示例

### 21.3.3.6 自动映射机制

在传送每帧数据时，通过自动映射机制使得协议控制寄存器 PCRL 和 PCRH、以及位域 SCTRH.FLE 的值保持不变（每次开始传送数据帧时发生映射操作）。可在任意时刻向这些寄存器设置新值，下次发送数据帧时新设置生效。在数据帧传送期间，所应用（映射）的设置不改变，尽管在开始传送之后已写入新值。

每次开始传送数据字时，位域 SCTRH.WLE 和 CTRL.SDIR 被自动映射。因此，一帧数据中可包含不同长度的数据字。建议只有在不传送数据帧时才修改 CTRL.SDIR，从而避免软硬件冲突。

需要注意：数据字的起始点对于发送器和接收器可不同。为了确保操作正确，建议只有当发送器和接收器都处于空闲状态时才修改 SCTRH.WLE。若接收器和接收器使用同一个数据信号（如在 LIN 总线系统中），检测到 RSI 事件之后，SCTRH.WLE 可在数据传送过程中修改。

### 21.3.3.7 数据帧结束控制

每帧数据中的数据位个数由位域 SCTRH.FLE 决定。为了使连续发送帧具有不同的帧长设置，该位域可由硬件修改。通过 TCSRL.FLEMD = 1 使能自动更新机制（此时，位 TCSRL.WLEMD、SELMD 和 WAMD 必须清零）。

若允许自动更新，当开始传送 ASC 帧时，发送控制信息 TCI 自动更新位域 TCSRL.FLEMD（一帧可由 1-32 位数据组成）。TCI 的值对应被写入的 TBUFxx（无附加数据缓存）或 INxx（带有附加数据缓存）的地址单元。采用该机制，向 TBUF07（或 IN07）写入数据字，会产生具有 8 位数据的 ASC 帧。

### 21.3.3.8 模式控制

在 ASC 模式下，支持以下内核模式：

- 运行模式 0/1：  
执行所设定的操作，对数据传送无影响。
- 终止模式 0：  
清除位 PSR.TXIDLE，不启动新的发送，正常完成当前的发送。不修改位 PSR.RXIDLE，仍可能接收数据。退出终止模式 0 时，根据 PCR.IDM 的设置置位 TXIDLE。
- 终止模式 1：  
清除位 PSR.TXIDLE，不启动新的发送，正常完成当前的发送。清除位 PSR.RXIDLE，不能接收数据，正常完成当前的接收。  
退出终止模式 1 时，根据 PCR.IDM 的设置置位 TXIDLE 和 RXIDLE。



### 21.3.3.9 禁用 ASC 模式

为了在关闭 ASC 模式时不造成数据损坏，此时接收器和发送器必须处于空闲状态。这一点可通过寄存器 KSCFG 请求终止模式 1 来确保。等到数据帧结束后可关闭 ASC 模式。

### 21.3.3.10 协议中断事件

在 ASC 模式下，可产生以下协议相关事件（并导致协议中断）。和发送器相关的事件包括冲突检测和发送帧结束事件；和接收器相关的事件包括同步分隔符检测、接收噪声检测、格式错误检查以及接收帧结束事件。

需要注意：为了监控新的事件，寄存器 PSR 不能由硬件自动清零，必须由软件清零。

- 冲突检测：  
该中断指示发送数据（DOUT）和 DX1 的输入采样值不匹配。具体描述见 [页 21-99](#)。
- 发送帧结束：  
该中断指示发送器已完成一帧的传送。在停止位之后 PSR.TFF 置位。不执行发送操作时。
- 接收帧结束：  
该中断指示接收器已完成一帧的接收。在停止位之后 PSR.RFF 置位。不执行接收操作时，可改变 DIN 信号到端口引脚的分配。
- 同步分隔符检测：  
LIN 网络使用该中断指示已接收到同步分隔符（在 LIN 帧的开头）。
- 接收噪声检测：  
该中断指示某一位的输入采样值和两个时间单元之前的输入采样值不相等。
- 帧格式错误：  
ASC 协议中停止位的值定义为 1 电平。若停止位的采样值为 0，该中断指示出现格式错误。

### 21.3.3.11 数据传送中断处理

数据传送中断指示和 ASC 帧处理相关的事件：

- 发送缓存中断 TBI：  
开始发送数据字的第一位数据之后置位 PSR.TBIF。此时是新数据字可写入 TBUF 的最早时刻。  
一旦产生发送缓存事件，TCSRL.TDV 被清零，新数据可载入发送缓存。
- 发送移位中断 TSI：

开始发送数据字的最后一位数据之后置位 PSR.TSIF。

- 接收开始中断 RSI:

在数据字的第一位数据的采样点之后置位 PSR.RSIF 指示。

- 接收中断 RI 和备选中断 AI:

若数据字不带奇偶校验位（奇偶校验机制禁用或者该数据字不是数据帧的最后一个字），在该数据字的最后一位的采样点之后置位 PSR.RIF。

若数据字带有奇偶校验位（奇偶校验机制使能或者该数据字是数据帧的最后一个字），若未检测到奇偶校验错误，在奇偶校验位的采样点之后置位 PSR.RIF；若检测到奇偶校验错误，置位 PSR.AIF。

由 RBUFSR.SOF = 1 指示已接收到一帧数据中的第一个数据字。

WA = 0 时置位 PSR.RIF 可产生接收中断；WA = 1 时置位 PSR.AIF 可产生备选中断。

### 21.3.3.12 协议相关参数和错误

协议相关参数（RBUFSR.PAR）和协议相关错误（RBUFSR.PERR）这两个标志位存放在接收缓存状态寄存器中，指示每个接收数据字的相关信息。

在 ASC 模式下，由协议相关参数来监控接收到的奇偶校验位，由协议相关错误来指示校验结果（0 = 接收到的奇偶校验位和计算出的奇偶校验值相等）。该信息只应用于每帧接收数据的最后一个数据字，若数据字不是一帧的最后一个数据字、或奇偶校验被禁用时，这两个标志位为 0。

### 21.3.3.13 接收缓存处理

若接收缓存 FIFO 可用（CCFG.RB = 1）并被使能（RBDTRH.SIZE > 0），在 ASC 模式下建议设置 RBCTRH.RCIM = 11<sub>B</sub>。若 OUTRH.RCI[0] = 1，表明该数据字是新一帧数据中的第一个数据字。若 OUTRH.RCI[4] = 1，表明产生奇偶校验错误。OUTRH.RCI[3]给出接收到的奇偶校验位的值。

在 RCI 模式下，可产生标准接收缓存事件和备选接收缓存事件（RBCTRH.RNM = 1）：

- 标准接收缓存事件指示可从 OUTRL 读取接收到的、无奇偶校验错误的数字字
- 备选接收缓存事件指示可从 OUTRL 读取接收到的、有奇偶校验错误的数字字

### 21.3.3.14 同步分隔符检测

接收器始终检查 DIN 信号上是否出现多个连续的 0 电平，其个数为每帧所设定的数据位个数（SCTRH.FLE）加 2（无奇偶校验）或加 3（带有奇偶校验）。若在该事件之后检测到采样数据位为 0 电平，位 PSR.SBD 置位。若 PCRL.SBIEN = 1，可产生协议

中断。每次检测到 DIN 信号发生下降沿跳变时，从 0 开始重新计数。在 LIN 总线系统中，从控制器将利用该特性检测同步分隔符（主控器件不检查同步分隔符）。

例如，对于无奇偶校验的 8 位数据 ASC 帧，若连续 10 位 0 电平之后下一位采样电平为 0（代表自第一个下降沿之后的第 11 位数据的采样值），置位 PCRL.SBD。

### 21.3.3.15 传送状态指示

若 PCRH.CTR[16]（接收状态使能 RSTEN）置位，接收状态可由标志位 PSR[9] = BUSY 监控。在这种情况下，接收一帧数据的整个过程中（从起始位到停止位），位 BUSY 置位。

若 PCRH.CTR[17]（发送状态使能 TSTEN）置位，发送状态可由标志位 PSR[9] = BUSY 监控。在这种情况下，发送一帧数据整个的过程中（从起始位到停止位），位 BUSY 置位。

若 RSTEN 和 TSTEN 均置位，标志位 BUSY 指示接收状态和发送状态逻辑或的结果。若这两位均清零，BUSY 不随传送状态而改变（状态变化被忽略）。

### 21.3.4 ASC 协议寄存器

ASC 模式下，寄存器 PCRH、PCRL 和 PSR 用于处理 ASC 相关信息。

#### 21.3.4.1 ASC 协议控制寄存器

ASC 模式下的寄存器 PCRL/PCRH 定义如下：

#### PCRL

协议控制寄存器 L [ASC 模式] (40<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PL			SP					FFI EN	FEI EN	RNI EN	CD EN	SBI EN	IDM	STP B	SMD
rw			rw					rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
SMD	0	rw	<p>采样模式</p> <p>该位域定义 ASC 接收器的采样模式。对于所选择的数据输入信号，一个位时间内可只采样一次、或采样三次（连续的时间单元）。若选择采样三次，则把多数判决的结果移入接收移位寄存器。</p> <p>0<sub>B</sub> 每一个位时间采样一次。采样当前输入值。</p>

符号	位序号	类型	功能描述
			1 <sub>B</sub> 每一个位时间采样三次并执行多数判决。
<b>STPB</b>	1	rw	<b>停止位</b> 该位定义 ASC 帧内停止位的个数。 0 <sub>B</sub> 1 位停止位。 1 <sub>B</sub> 2 位停止位。
<b>IDM</b>	2	rw	<b>空闲检测模式</b> 该位定义是否关闭空闲检测、或基于帧长进行空闲检测。 0 <sub>B</sub> 关闭总线空闲检测，位 PSR.TXIDLE 和 PSR.RXIDLE 自动置位从而允许在数据传送之前不检查输入值。 1 <sub>B</sub> 出现 SCTR.H.FLE 加 2（无奇偶校验）或加 3（带有奇偶校验）个连续的被动位时间之后，总线视为空闲。
<b>SBIEN</b>	3	rw	<b>同步分隔符中断使能</b> 若检测到同步分隔符，该位使能产生协议中断。自动检测始终有效，因此，位 SBD 的设置可和 SBIEN 无关。 0 <sub>B</sub> 禁止产生中断。 1 <sub>B</sub> 允许产生中断。
<b>CDEN</b>	4	rw	<b>冲突检测使能</b> 该位使能发送器对冲突检测做出响应。 0 <sub>B</sub> 禁止冲突检测。 1 <sub>B</sub> 若检测到发送冲突，发送器终止数据发送，输出 1，置位 PSR.COL 并产生协议中断。为了能够重新发送数据，PSR.COL 必须由软件清零。
<b>RNIEN</b>	5	rw	<b>接收噪声检测中断使能</b> 若检测到接收噪声，该位使能产生协议中断。自动检测始终有效，因此，位 PSR.RNS 的设置可和 PCRL.RNIEN 无关。 0 <sub>B</sub> 禁止产生中断。

符号	位序号	类型	功能描述
			1 <sub>B</sub> 允许产生中断。
<b>FEIEN</b>	6	rw	<b>格式错误中断使能</b> 若检测到格式错误，该位使能产生协议中断。自动检测始终有效，因此，位 PSR.FER0/FER1 的设置可和 PCRL.FEIEN 无关。 0 <sub>B</sub> 禁止产生中断。 1 <sub>B</sub> 允许产生中断。
<b>FFIEN</b>	7	rw	<b>数据帧结束中断使能</b> 接收器接收/发送器发送最后一帧之后，该位使能产生协议中断。自动检测始终有效，因此，位 PSR.RFF 或 PSR.TFF 的设置可和 PCRL.FFIEN 无关。 0 <sub>B</sub> 禁止产生中断。 1 <sub>B</sub> 允许产生中断。
<b>SP</b>	[12:8]	rw	<b>采样点</b> 该位域定义数据位的采样点。采样点切勿超出设定的位时序（PCRL.SP ≤ BRGL.DCTQ）。
<b>PL</b>	[15:13]	rw	<b>脉冲长度</b> 该位域定义 0 数据位的长度，以时间单元计数（从时间单元 0 开始计数）。电平为 0 的数据位可产生一个长度小于位时间的 0 脉冲（如用于 IrDA）。PL 不能改变位时间的长度，只能改变输出信号上 0 脉冲的长度。 脉冲长度一定不能大于设定的位时序（PCRL.PL ≤ BRGL.DCTQ）。 发送器会用到该位域的设置；接收器不使用。 000 <sub>B</sub> 脉冲长度和位长度相等（0 不缩短）。 001 <sub>B</sub> 0 数据位的脉冲长度为 2 个时间单元。 010 <sub>B</sub> 0 数据位的脉冲长度为 3 个时间单元。 ... .. 111 <sub>B</sub> 0 数据位的脉冲长度为 8 个时间单元。

**PCRH**

**协议控制寄存器 H [ASC 模式] (42<sub>H</sub>) 复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>M CLK</b>	<b>0</b>													<b>TST EN</b>	<b>RST EN</b>
rw	r													rw	rw

符号	位序号	类型	功能描述
<b>RSTEN</b>	0	rw	<b>接收状态使能</b> 该位使能根据接收状态修改标志位 PSR[9] = BUSY。 0 <sub>B</sub> 不根据接收状态修改标志位 PSR[9]。 1 <sub>B</sub> 在接收一帧数据的整个过程中，标志位 PSR[9]置位。
<b>TSTEN</b>	1	rw	<b>发送状态使能</b> 该位使能根据发送状态修改标志位 PSR[9] = BUSY。 0 <sub>B</sub> 不根据发送状态修改标志位 PSR[9]。 1 <sub>B</sub> 在发送一帧数据的整个过程中，标志位 PSR[9]置位。
<b>0</b>	[14:2]	r	<b>保留</b> 读操作返回 0；ASC 模式下不修改该位域。
<b>MCLK</b>	15	rw	<b>主控时钟使能</b> 该位使能产生主控时钟 MCLK。 0 <sub>B</sub> 禁止产生 MCLK，MCLK 信号为 0。 1 <sub>B</sub> 允许产生 MCLK。

**21.3.4.2 ASC 协议状态寄存器**

ASC 模式下的寄存器 PSR 定义如下。PSR 不被硬件清零。

向寄存器 PSR 写 1 时，PSR 中相应的标志位被清零；向 PSR 写 1 将置位相应的标志位，但不会引发其它操作（不产生中断）。写 0 不起作用。在使能新协议之前，PSR 标志位应由软件清零。

**PSR**

**协议状态寄存器 [ASC 模式]**

**(44<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>AIF</b>	<b>RIF</b>	<b>TBIF</b>	<b>TSIF</b>	<b>DLIF</b>	<b>RSIF</b>	<b>BU SY</b>	<b>TFF</b>	<b>RFF</b>	<b>FER 1</b>	<b>FER 0</b>	<b>RNS</b>	<b>COL</b>	<b>SBD</b>	<b>RX IDLE</b>	<b>TX IDLE</b>
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位序号	类型	功能描述
<b>TXIDLE</b>	0	rwh	<b>发送空闲</b> 该位指示发送线（DX1）是否已空闲。只有当 TXIDLE 置位时才能开始发送数据帧。 0 <sub>B</sub> 发送线（DX1）还未空闲。 1 <sub>B</sub> 发送线（DX1）已空闲，可以发送数据帧。
<b>RXIDLE</b>	1	rwh	<b>接收空闲</b> 该位指示接收线（DX0）是否已空闲。只有当 RXIDLE 置位时才能开始接收数据帧。 0 <sub>B</sub> 接收线（DX0）还未空闲。 1 <sub>B</sub> 接收线（DX0）已空闲，可以接收数据帧。
<b>SBD</b>	2	rwh	<b>同步分隔符检测<sup>1)</sup></b> 若检测到设定数目的连续 0 电平值时（称为同步分隔符，用于 LIN 总线系统），该位置位。 0 <sub>B</sub> 还未检测到同步分隔符。 1 <sub>B</sub> 已检测到同步分隔符。
<b>COL</b>	3	rwh	<b>冲突检测<sup>1)</sup></b> 若检测到发送冲突（且 PCRL.CDEN = 1），该位置位。 0 <sub>B</sub> 还未检测到发送冲突，可发送数据帧。 1 <sub>B</sub> 已检测到发送冲突，不能发送数据帧。
<b>RNS</b>	4	rwh	<b>接收噪声检测<sup>1)</sup></b> 若检测到接收噪声，该位置位。 0 <sub>B</sub> 还未检测到接收噪声。 1 <sub>B</sub> 已检测到接收噪声。

符号	位序号	类型	功能描述
<b>FER0</b>	5	rwh	<b>停止位 0 的格式错误<sup>1)</sup></b> 若停止位 0 的采样值为 0，该位置位（称为格式错误 0）。 0 <sub>B</sub> 还未检测到格式错误 0。 1 <sub>B</sub> 已检测到格式错误 0。
<b>FER1</b>	6	rwh	<b>停止位 1 的格式错误<sup>1)</sup></b> 若停止位 1 的采样值为 0，该位置位（称为格式错误 1）。 0 <sub>B</sub> 还未检测到格式错误 1。 1 <sub>B</sub> 已检测到格式错误 1。
<b>RFF</b>	7	rwh	<b>接收帧结束<sup>1)</sup></b> 若接收器已接收到最后一个停止位，该位置位。 0 <sub>B</sub> 接收数据帧未结束。 1 <sub>B</sub> 接收数据帧结束。
<b>TFF</b>	8	rwh	<b>发送帧结束<sup>1)</sup></b> 若发送器已送出最后一个停止位，该位置位。 0 <sub>B</sub> 发送数据帧未结束。 1 <sub>B</sub> 发送数据帧结束。
<b>BUSY</b>	9	rwh	<b>传送状态 BUSY</b> 该位指示接收状态（若 PCRH.RSTEN = 1）、或发送状态（若 PCRH.TSTEN = 1）、或二者逻辑或的结果（若 PCRH.RSTEN = PCRH.TSTEN = 1）。 0 <sub>B</sub> 未进行数据传送。 1 <sub>B</sub> 正在进行数据传送。
<b>RSIF</b>	10	rwh	<b>接收开始指示标志</b> 0 <sub>B</sub> 未发生接收开始事件。 1 <sub>B</sub> 已发生接收开始事件。
<b>DLIF</b>	11	rwh	<b>数据丢失指示标志</b> 0 <sub>B</sub> 未发生数据丢失事件。



符号	位序号	类型	功能描述
			1 <sub>B</sub> 已发生数据丢失事件。
<b>TSIF</b>	12	rwh	<b>发送移位指示标志</b> 0 <sub>B</sub> 未发生发送移位事件。 1 <sub>B</sub> 已发生发送移位事件。
<b>TBIF</b>	13	rwh	<b>发送缓存指示标志</b> 0 <sub>B</sub> 未发生发送缓存事件。 1 <sub>B</sub> 已发生发送缓存事件。
<b>RIF</b>	14	rwh	<b>接收指示标志</b> 0 <sub>B</sub> 未发生接收事件。 1 <sub>B</sub> 已发生接收事件。
<b>AIF</b>	15	rwh	<b>备选接收指示标志</b> 0 <sub>B</sub> 未发生备选接收事件。 1 <sub>B</sub> 已发生备选接收事件。

1) 该状态位可产生协议中断（见[页 21-21](#)）。中断状态标志的一般描述请参见中断章节。

### 21.3.5 硬件 LIN 支持

为了支持 LIN 协议，对于主控制器，应设置  $TCSRL.FLEMD = 1$ ；对于从控制器， $TCSRL.FLEMD$  应清零、必须设置一帧固定由 8 位数据组成（ $SCTRH.FLE = 7_H$ ）。必须关闭主控和从控制器的奇偶校验（ $CCR.PM = 00_B$ ），数据传输时移位 LSB 在先（ $SCTRL.SDIR = 0$ ），具有 1 位停止位（ $PCRL.STPB = 0$ ）。

本地互连网络（LIN）数据交换协议所包含的多个符号可由 ASC 模式处理。每个 LIN 符号代表完整的一帧 ASC 数据。LIN 总线为单主控 - 多从控总线系统（准确定义请参阅 LIN 协议规范）。

完整 LIN 帧包含以下符号：

- 同步分隔符：

主控制器发送一个同步分隔符以指示开始新的一帧。它包含至少 13 位连续的 0 电平、以及至少 1 位 1 电平（对应 1 个停止位）。因此必须向 **TBUF11**（或 **IN11**）写入 0（从而一帧由 **SOF** 和随后的 12 位 0 电平数据组成）。

从控制器检测 11 位连续的 0 电平（执行同步分隔符检测）。若检测到该事件，置位 **PSR.SBD**，可产生协议中断。此外，接收数据值 0 存放在接收缓存中，指示产生格式错误。

若必须调整从控制器的波特率以和主控制器匹配，必须在开始下一个符号之前设置  $BRGL.TMEN = 1$ （使能波特率测量）、 $DX0CR.CM = 10_H$ （下降沿有效）和  $DX1CR.CM = 00_H$ 。

- 同步字节：

主控制器通过向  $TBUF07$ （或  $IN07$ ）写入数据值  $55_H$  发送该符号。从控制器接收到该符号后可不执行其它操作（并丢弃该数据），也可以利用数据的下降沿进行波特率测量。位  $PSR.TSIF = 1$ （并可选择产生中断）指示检测到下降沿并把该下降沿和前一个下降沿之间的时间捕获到  $BRGH.PDIV$  中。在第 2、3、4 和第 5 次激活  $TSIF$  之后读取有效的捕获值。在该符号内第 5 次激活  $TSIF$  之后，必须关闭波特率检测（ $BRGL.TMEN = 0$ ），可设置  $BRGH.PDIV =$  先前的捕获值 / ( $2 \times$  每位中的时间单元数)（假定  $BRGL.PCTQ = 00_B$ ）。

测量波特率时为了避免  $PDIV$  溢出，在设置分数分频器的预分频值时，必须确保  $PDIV$  的目标值小于  $1024 / (2 \times \text{每位中的时间单元数})$ 。由于该操作会使  $PDIV$  的目标值较小，波特率测量精度因而受到限制。因此，建议执行以下操作：

- 为了测量波特率，设置分数分频器的分频值 = ( $2 \times$  每位中的时间单元数)
- 重新写入  $FDRL.DM$  的当前值，重新启动分数分频器
- 设置  $BRGL.TMEN = 1$  开启波特率测量功能（需要注意：使能波特率测量时，不可能进行同步分隔符检测）
- 恢复分数分频器的原先设置
- 设置  $BRGL.TMEN = 0$  关闭波特率测量功能
- $BRGL.PDIV$  中的测量结果此时可直接用作波特率设置

- 其它符号：

LIN 帧中的其它符号可由  $ASC$  数据帧处理，无需特殊操作。

若 LIN 主控制器逐帧发送 LIN 数据帧，输入  $DX2$  可连接到外部定时器以触发发送操作（例如，同步分隔符已准备就绪，但只有在满足触发时才开始传送）。需要注意： $ASC$  接收器在进行波特率测量时，同一  $USIC$  通道中的  $ASC$  发送器不能发送数据。

## 21.4 同步串行通道 (SSC)

同步串行通道  $SSC$  提供类似  $SPI$  的数据传送功能。 $SSC$  可处理主控制器和从控制器（至少一个）之间同步数据帧的发送和接收操作。若  $CCFG.SSC = 1$ （ $SSC$  模式可用），由  $CCR.MODE = 0001_B$  选择  $SSC$  模式。

本章包含以下内容：

- 信号描述（见 [页 21-112](#)）
- 操作  $SSC$  的一般性描述（见 [页 21-118](#)）
- 主控模式操作（见 [页 21-120](#)）

- 从控模式操作（见页 21-125）
- 协议寄存器（见页 21-127）
- SSC 时序（见页 21-132）

### 21.4.1 信号描述

同步 SSC 数据传送由移位时钟控制发送和/或接收数据信号的同步传送、决定何时数据有效（定义发送和采样点）。

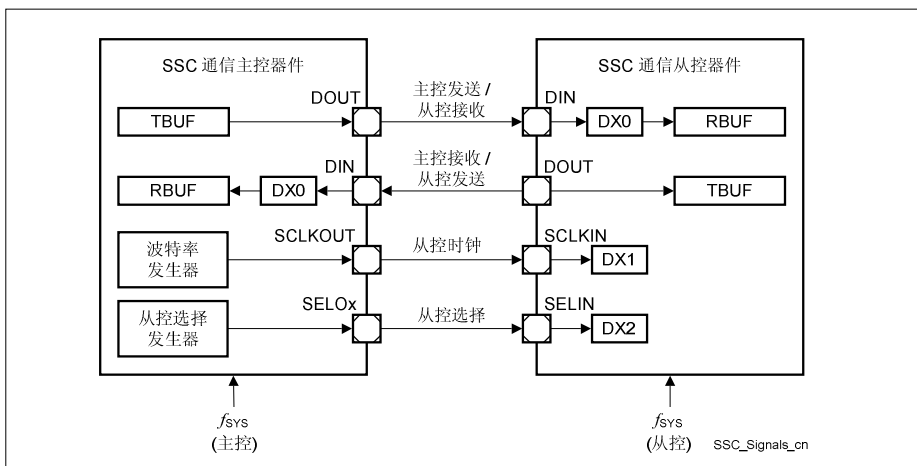


图 21-31 全双工通信的 SSC 信号连接

为了指示数据发送的开始和结束、分别寻址多个从控器件，SSC 模块支持从控选择信号的处理（可选，并非必须使用）。主控模式有多达 8 个不同的从控选择输出信号（SELO<sub>x</sub>, x = 0-7）；从控模式有 1 个从控选择输入信号 SELIN。在多数应用中，从控选择信号低有效。

工作在主控模式下的器件控制数据帧的开始和结束、控制移位时钟和从控选择信号的产生。这包含移位时钟的波特率设置、以及移位时钟和从控选择输出信号之间的延迟。若多个 SSC 模块相连，一次只能有一个 SSC 主控器件，可以有多个从控器件。从控器件接收移位时钟和从控选择信号。关于输入级 DX0、DX1 和 DX2 的设置，参见页 21-32。

表 21-8 SSC 通信信号

SSC 模式	接收数据	发送数据	移位时钟	从控选择
主控	MRST <sup>1)</sup> , 输入 DIN, 由 DX0 处理	MTSR <sup>2)</sup> , 输出 DOUT	输出 SCLKOUT	输出 SELOx
从控	MTSR, 输入 DIN, 由 DX0 处理	MRST, 输出 DOUT	输入 SCLKIN, 由 DX1 处理	输入 SELIN, 由 DX2 处理

- 1) MRST = 主控接收从控发送, 也称作 MISO = 主控输入从控输出  
2) MTSR = 主控发送从控接收, 也称作 MOSI = 主控输出从控输入

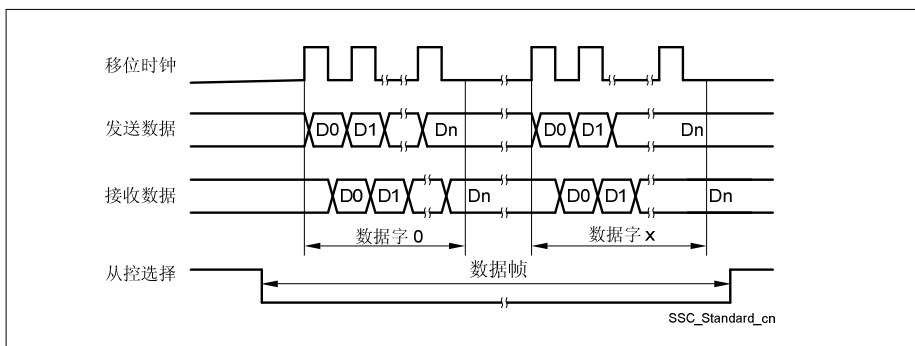


图 21-32 4 线标准通信信号

#### 21.4.1.1 发送和接收数据信号

半双工模式下, 只用一条数据线进行主控和从控器件之间的单向数据传送。在这种情况下, MRST 和 MTSR 相连, 根据数据方向一个作为输入、另一个作为输出。用户必须谨慎处理数据方向以避免数据冲突 (例如: 对于漏极开路驱动的线条与连接, 发送全“1”的虚设数据; 或使能/禁止推挽输出驱动)。全双工模式下, 通过两个独立的数据信号 MTST 和 MRST 进行主控和从控器件之间的双向数据传送, 如图 21-31 所示。

接收数据输入信号 DIN 由输入级 DX0 处理。在主控模式 (由 MRST 表示) 和从控模式下 (由 MTSR 表示), 数据输入信号 DIN 从输入引脚获取。DOUT (数据输出) 信号的极性由位域 SCTRL.DOCFG (数据输出配置) 控制。

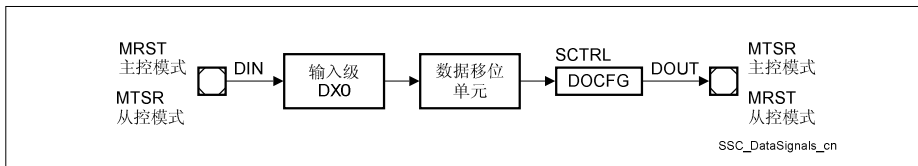


图 21-33 SSC 数据信号

#### 21.4.1.2 移位时钟信号

移位时钟信号由输入级 DX1 处理。在从控模式下，SCLKIN 来自外部主控制器件，因此必须将 DX1 连接到输入引脚上。输入级可将接收到的输入信号反相，从而根据数据移位单元的功能（上升沿发送数据，下降沿接收数据）相应调整 SCLKIN 的信号极性。

在主控模式下，由内部波特率发生器产生移位时钟。波特率发生器的输出信号 SCLK 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 被外部从控器件使用。

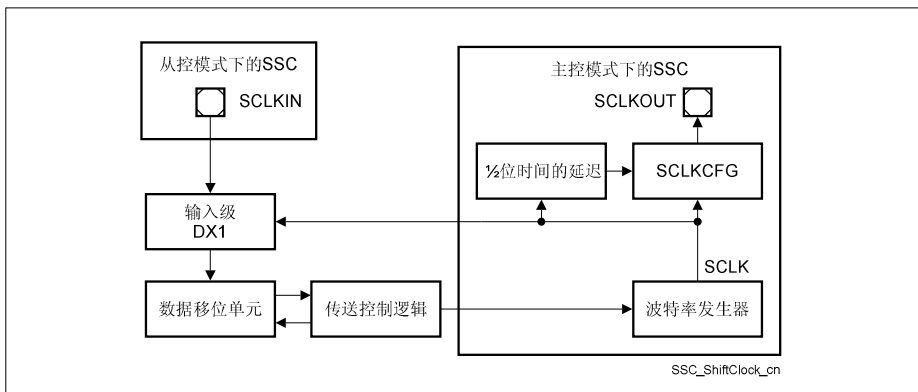


图 21-34 SSC 移位时钟信号

由于 SSC 支持多种不同的应用，因此在主控模式下，有多种配置 SCLKOUT 的方式。通过位域 BRGH.SCLKCFG（移位时钟配置）可实现 4 种可能的配置，如图 21-35 所示。

- 无延迟，无极性翻转（SCLKCFG = 00<sub>B</sub>，SCLKOUT 等于 SCLK）：

SCLKOUT 的无效电平为 0，此时不传送数据。在 SCLKOUT 的第一个上升沿发送数据帧中的第一位数据；在 SCLKOUT 的第一个下降沿接收数据帧中的第一位数据。在 SCLKOUT 的最后一个上升沿发送数据帧中的最后一位数据；在

SCLKOUT 的最后一个下降沿接收数据帧中的最后一位数据。该设置可用于主控和从控模式。它和内部数据移位单元的操作相一致。

- 无延迟，极性翻转（SCLKCFG = 01<sub>B</sub>）：

SCLKOUT 的无效电平为 1，此时不传送数据。在 SCLKOUT 的第一个下降沿发送数据帧中的第一位数据；在 SCLKOUT 的第一个上升沿接收数据帧中的第一位数据。在 SCLKOUT 的最后一个下降沿发送数据帧中的最后一位数据；在 SCLKOUT 的最后一个上升沿接收数据帧中的最后一位数据。该设置可用于主控和从控模式。

- SCLKOUT 延迟 1/2 个移位时钟周期，无极性翻转（SCLKCFG = 10<sub>B</sub>）：

SCLKOUT 的无效电平为 0，此时不传送数据。

在 SCLKOUT 第一个上升沿之前的 1/2 移位时钟周期发送数据帧中的第一位数据。由于延迟的结果，将在 SCLKOUT 的下降沿发送后续的数据位；在 SCLKOUT 最后一个上升沿之前的 1/2 移位时钟周期发送数据帧中的最后一位数据。在 SCLKOUT 第一个下降沿之前的 1/2 移位时钟周期接收数据帧中的第一位数据。由于延迟的结果，将在 SCLKOUT 的上升沿接收后续的数据位；在 SCLKOUT 最后一个下降沿之前的 1/2 移位时钟周期接收数据帧中的最后一位数据。

该设置只可用于主控模式，不可用于从控模式（所连接的从控器件必须在 SCLKOUT 的第一个跳变沿之前提供第一位数据，比如，一旦由从控选择信号寻址到某从控器件，该器件立即提供第一位数据）。

- SCLKOUT 延迟 1/2 个移位时钟周期，极性翻转（SCLKCFG = 11<sub>B</sub>）：

SCLKOUT 的无效电平为 1，此时不传送数据。

在 SCLKOUT 第一个下降沿之前的 1/2 移位时钟周期发送数据帧中的第一位数据。由于延迟的结果，将在 SCLKOUT 的上升沿发送后续的数据位；在 SCLKOUT 最后一个下降沿之前的 1/2 移位时钟周期发送数据帧中的最后一位数据。在 SCLKOUT 第一个上升沿之前的 1/2 移位时钟周期接收数据帧中的第一位数据。由于延迟的结果，将在 SCLKOUT 的下降沿接收后续的数据位；在 SCLKOUT 最后一个上升沿之前的 1/2 移位时钟周期接收数据帧中的最后一位数据。

该设置只可用于主控模式，不可用于从控模式（所连接的从控器件必须在 SCLKOUT 的第一个跳变沿之前提供第一位数据，比如，一旦由从控选择信号寻址到某从控器件，该器件立即提供第一位数据）。

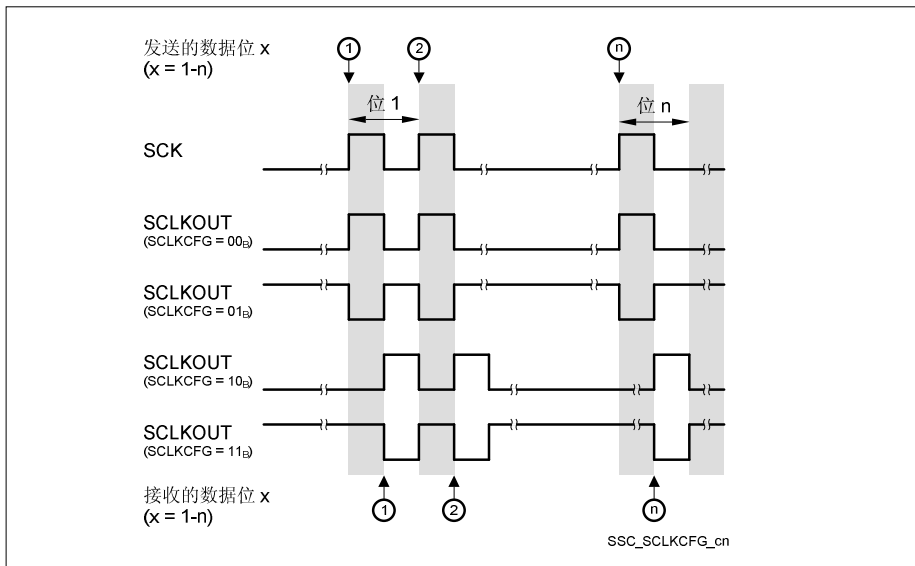


图 21-35 SSC 主控模式下 SCLKOUT 的配置

### 21.4.1.3 从控选择信号

从控选择信号由输入级 DX2 处理。在从控模式下，输入信号 SELIN 来自外部主控器件。输入级可将接收到的输入信号反相，从而根据数据移位单元的功能（模块内部信号可看作高有效，因此，只有当数据移位单元的从控选择输入电平为 1 时，才能进行数据传送。否则忽略移位时钟脉冲、不传送数据）相应调整 SELIN 的信号极性。若输入信号 SELIN 低有效，应在 DX2 输入级中进行极性翻转。

在主控模式下，由内部从控选择发生器产生主从选择信号 MSLS。为了能够独立寻址不同的外部从控器件，可通过 SELCFG 配置内部 MSLS 信号、以最多 8 个 SELOx 信号的形式输出。

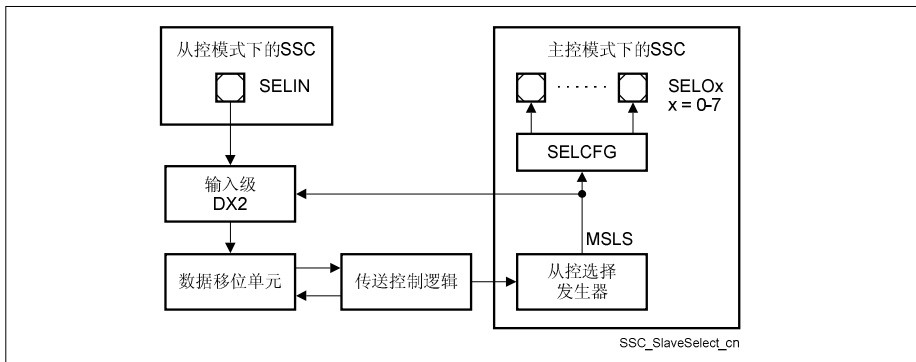


图 21-36 SSC 从选择信号

通过协议控制寄存器 PCRL 和 PCRH 控制 SELCFG。关于 MSLS 信号产生的具体描述，请参见[章节 21.4.3.2](#)。

- PCRL.SELCTR 选择直接选择模式或编码选择模式
- PCRL.SELINV 翻转 SELOx 的输出极性
- PCRH.SELO[7:0]代表每条 SELOx 输出线的值

SELCFG 支持以下 SELOx 输出信号的配置：

- 直接选择模式（SELCTR = 1）：

每条 SELOx ( $x = 0-7$ ) 输出线可直接和一个外部从控器件相连。若位域 SELO 中位  $x$  的值为 0，SELOx 输出恒无效。当内部信号 MSLS 有效（见[章节 21.4.3.2](#)）且位域 SELO 中位  $x$  的值为 1，SELOx 输出有效。若在传送一帧数据的过程中位域 SELO 中多个位的值为 1，则可同时寻址多个外部从控器件，其个数由可用的 SELOx 输出数所限制。

- 编码选择模式（SELCTR = 0）：

为了增加外部从控器件的个数，SELOx 输出线 ( $x = 0-7$ ) 可用作外部地址译码器的地址。这些线只在开始传送新的数据帧时改变，它们和 MSLS 无其它关系。信号 SELO0 可用作外部地址译码器的使能信号。当 MSLS 有效（在传送一帧数据的过程中）且位域 SELO 中位 0 的值为 1，信号 SELO0 有效。此外，在编码选择模式下，SELO0 相对 MSLS 延迟一个  $f_{sys}$  时钟周期，从而在启用地址译码器之前使其它 SELOx 输出线稳定。



## **21.4.2 操作 SSC**

本节给出操作 SSC 的一般性描述，不特定针对主控或从控模式。

### **21.4.2.1 自动映射机制**

在传送每帧数据时 (= MSLS 有效)，通过自动映射机制使得波特率控制寄存器 BRGL 和 BRGH、位域 SCTRH.FLE 以及协议控制寄存器 PCRL 和 PCRH 的值保持不变。可在任意时刻向这些寄存器设置新值，下次传送数据帧时新设置生效。在数据帧传送期间，所应用（映射）的设置不改变，尽管在开始传送之后已写入新值。

每次开始传送数据字时，位域 SCTRH.WLE 和 SCTRL.SDIR 被自动映射。因此，一帧数据中可包含不同长度的数据字。建议只有在不在传送数据帧时才修改 SCTRL.SDIR，从而避免软硬件冲突。

**需要注意：**数据字的起始点对于发送器（第一位发送数据）和接收器（第一位接收数据）不同。为了确保操作正确，建议在修改 SCTRH.WLE 之前查询接收开始中断 RSI 的状态。若 TCSRL.WLEMD = 1，建议在产生接收开始中断之后更新 TCSRL 和 TBUFxx。

### **21.4.2.2 模式控制**

在 SSC 模式下，支持以下内核模式：

- 运行模式 0/1：  
执行所设定的操作，对数据传送无影响。
- 终止模式 0/1：  
发送缓存中的内容视为无效。尽管 TCSRL.TDV 当前为 0，它不能被终止模式修改。

在主控模式下，正常完成当前的数据字传送，但不启动新数据字传送（终止条件不被视为帧结束条件）。从控模式下，正常完成当前的数据字传送。从控器件处于终止模式时，若外部主控器件启动数据字传送，从控器件不发送有效数据、而是发送被动的数据。为了避免从控器件发送被动的数据，若主控器件不响应从控器件的终止模式，建议不要将 SSC 从控器件设置成该模式。

### **21.4.2.3 禁用 SSC 模式**

为了在关闭 SSC 模式时不造成数据损坏，此时接收器和发送器必须处于空闲状态。这一点可通过寄存器 KSCFG 请求终止模式 1 来确保。通过 KSCFG.ACK = 1 对终止模式 1 作出应答之后，可关闭 SSC 模式。

#### 21.4.2.4 数据帧控制

SSC 数据帧可由多个连续的数据字组成，数据字之间由字内延迟分隔。若无字内延迟，多个数据字看似构成一个更长的数据字，长度和数据帧相等。一帧数据内各数据字的长度大多数都相同，但也会有不相同的情况。可通过 **SCTRH.WLE** 读取每个新数据字的字长信息；在每次开始传送新数据帧时，通过 **SCTRH.FLE** 读取帧长信息。

可通过两种不同的方式定义 SSC 数据帧的长度：

- 通过每帧中数据位的个数：

若已定义了每一帧数据中数据位的个数（帧长 **FLE**），则不必通过从控选择信号来指示一帧的起始和结束。若在某个数据字内达到设定的数据位个数，数据帧视为已结束、最后这个数据字中剩余的数据位被忽略、不被传送。

该方法可应用于最多包含 **63** 位数据的数据帧。

- 通过从控选择信号：

若每一帧数据中数据位的个数未知，由从控选择信号来指示一帧的起始和结束。若在某个数据字内检测到从控选择信号失效，数据帧视为已结束、最后这个数据字中剩余的数据位被忽略、不被传送。

该方法必须应用于帧长超过 **63** 位的数据帧（**FLE** 的编程极限值）。使用从控选择信号的好处是可以明确定义数据流中各数据帧的开始和结束。此外，从控选择信号可用于单独寻址从控器件。

#### 21.4.2.5 奇偶校验模式

SSC 模式不支持奇偶校验，必须设置 **CCR.PM = 00<sub>B</sub>**。

#### 21.4.2.6 传送模式

在 SSC 模式下，必须设置 **SCTRL.TRM = 01<sub>B</sub>** 以允许数据传送。设置 **SCTR.TRM = 00<sub>B</sub>** 将立即禁止并终止数据传送。

#### 21.4.2.7 数据传送中断处理

数据传送中断指示和 SSC 帧处理相关的事件：

- 发送缓存中断 **TBI**：

开始发送数据字的第一位数据之后置位 **PSR.TBIF**。

- 发送移位中断 **TSI**：

开始发送数据字的最后一位数据之后置位 **PSR.TSIF**。

- 接收开始中断 **RSI**：

在接收到数据字的第一位数据之后置位 **PSR.RSIF** 指示。

发生该事件之后，位 **TCSRL.TDV** 被清零，新数据可载入发送缓存。

- 接收中断 RI:

始终由 RBUFSR.SOF = 0 指示接收到多字数据帧中的第二个、第三个以及后续的数据字。若 RBUFSR.SOF = 0，接收到最后一位数据后置位 PSR.RIF。

位 RBUFSR.SOF 指示接收到的数据字是一帧数据中的第一个数据字还是后续的数据字。SSC 模式下，它决定是否产生备选中断或接收终端。

- 备选中断 AI:

始终由 RBUFSR.SOF = 1 指示接收到数据帧中的第一个数据字（对多字帧和单字帧均适用）。SSC 模式下，它将置位 PSR.AIF。

### 21.4.2.8 协议相关参数和错误

协议相关参数 (RBUFSR.PAR) 和协议相关错误 (RBUFSR.PERR) 这两个标志位存放在接收缓存状态寄存器中，指示每个接收数据字的相关信息。

在 SSC 模式下，RBUFSR.PAR 始终为 0。由协议相关错误标志 RBUFSR.PERR 指示是否开始接收新的数据帧 (0 = 接收到的数据字不是一帧中的第一个数据字；1 = 接收到的数据字是新一帧中的第一个数据字)。

### 21.4.2.9 接收缓存处理

若接收缓存 FIFO 可用 (CCFG.RB = 1) 并被使能 (RBDTRH.SIZE > 0)，在 SSC 模式下建议设置 RBCTRH.RCIM = 01<sub>B</sub>。若 OUTRH.RCI[4] = 1，表明该数据字是新一帧数据中的第一个数据字。OUTRH.RCI[3:0]给出接收到的数据的长度。

在 RCI 模式下，可产生标准接收缓存事件和备选接收缓存事件 (RBCTRH.RNM = 1)：

- 标准接收缓存事件指示可从 OUTRL 读取接收到的、一帧中的非首个数据字
- 备选接收缓存事件指示可从 OUTRL 读取接收到的、一帧中的第一个数据字

### 21.4.3 主控模式下操作 SSC

在主控模式下操作 SSC 时，必须考虑以下因素：

- 选择 SSC 模式：

建议在 CCR.MODE = 0000<sub>B</sub> 时配置 SSC 的所有参数（在运行期间这些参数不改变）。必须设置 SCTRL.TRM = 01<sub>B</sub>。必须在 CCR.MODE = 0000<sub>B</sub> 时配置输入级，从而避免输入信号上出现的未预计跳变，随后可通过 CCR.MODE = 0001<sub>B</sub> 使能 SSC 模式。

- 引脚连接：

通过 DX0CR.INSW = 1 建立输入级 DX0 和接收数据输入引脚（信号 DIN）之间的连接；配置发送数据输出引脚（信号 DOUT）。

- 波特率产生：  
必须选择期望的波特率设置，包括分数分频器和波特率发生器设置。必须设置 **DX1CR.INSW = 0**，将波特率发生器的输出 **SCLK** 直接用作数据移位单元的输入。配置移位时钟输出引脚（信号 **SCLKOUT**）。
- 从控选择的产生：  
必须通过 **PCRL.MSLSEN = 1** 并编程时间单元计数器以使能从控选择延迟的产生。必须设置 **DX2CR.INSW = 0**，将从控选择发生器输出 **MSLS** 用作数据移位单元的输入。如果需要，配置从控选择输出引脚（信号 **SELOx**）。
- 数据格式配置：  
根据应用需要，通过寄存器 **SCTRL** 和 **SCTRH** 设置字长、帧长和移位方向。

#### 21.4.3.1 波特率产生

SSC 模式下的波特率（决定一位数据的长度）由 **SCLK** 信号的频率决定（1 个  $f_{SCLK}$  时钟周期代表一位数据）。SSC 波特率的产生不涉及任何时间单元计数器。

在标准 SSC 应用中，可选的 **MCLK** 输出信号和 **SCLK** 之间相位无关，可被禁止（**BRGL.PPPEN = 0**）。在这种情况下，**SCLK** 信号直接来自协议输入频率  $f_{PIN}$ 。在特殊情况下 **MCLK** 和 **SCLK** 之间需要有固定的相位关系（比如使用 **MCLK** 作为外部器件的时钟参考），必须使用附加的 2 分频电路（**BRGL.PPPEN = 1**）。

分频因子可编程，通过位域 **BRGH.PDIV** 设置。

$$f_{SCLK} = \frac{f_{PIN}}{2} \times \frac{1}{PDIV+1} \quad \text{若 } PPPEN = 0$$

$$f_{SCLK} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV+1} \quad \text{若 } PPPEN = 1 \quad (21.8)$$

#### 21.4.3.2 MSLS 产生

从控选择信号指示一帧数据的起始和结束，通信主控器件还利用该信号分别选择期望的从控器件。在开始传送一帧的数据部分之前，通信主控器件的从控选择输出变为有效，有效时间长度可编程（前沿延迟  $T_{ld}$ ），从控器件利用该时间进行通信准备。在传送最后一位数据之后，从控选择信号仍保持一段时间有效，该时间长度可编程（后沿延迟  $T_{ld}$ ），用以满足从控保持时间的需要，之后从控选择信号再次变为无效。若数据帧背靠背连续传送，从控选择信号失效和下次激活之间的最小时间可编程（下一帧延迟  $T_{nf}$ ）。若数据帧由多个数据字组成，数据字之间的延迟也可编程（字内延迟  $T_{iw}$ ）。

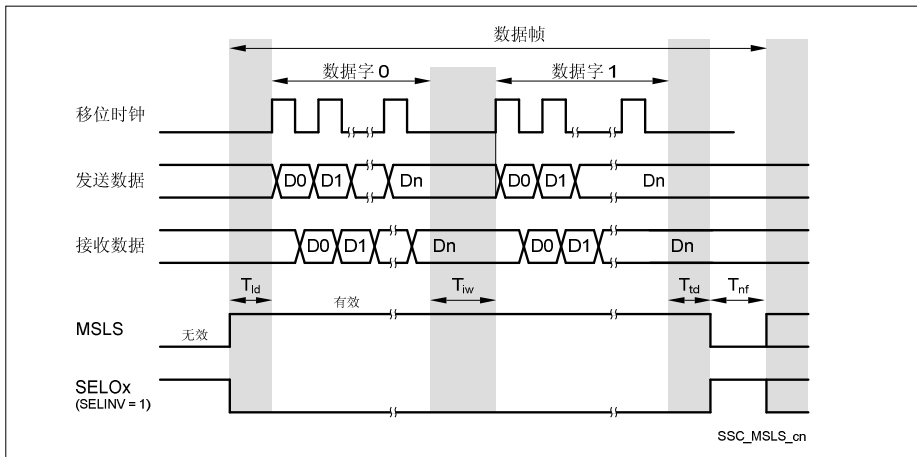


图 21-37 SSC 主控模式下 MSLS 的产生

在 SSC 主控模式下，从控选择延迟的定义如下：

- 前沿延迟  $T_{ld}$ :  
前沿延迟从有效数据可用开始，此时内部信号 MSLS 变为有效。在前沿延迟结束之后，波特率发生器产生 SCLK 的第一个移位时钟沿（上升沿）。
- 后沿延迟  $T_{td}$ :  
后沿延迟从一帧数据的最后一个 SCLK 周期结束开始。后沿延迟结束之后，内部信号 MSLS 变为无效。
- 字内延迟  $T_{iw}$ :  
该延迟可选，可由 PCRH.TIWEN 使能/禁止。若禁止字内延迟（ $TIWEN = 0$ ），当前数据字的最后一位和同一帧中下一个数据字的第一位之间无间隔。若使能字内延迟（ $TIWEN = 1$ ），字内延迟从每个数据字的最后一个 SCLK 周期结束开始。字内延迟结束之后，开始产生下一个数据字的第一个 SCLK 周期。在此过程中，不产生移位时钟脉冲，信号 MSLS 保持有效。通信方有时间来“消化”前一个数据字或准备下一个数据字。
- 下一帧延迟  $T_{nf}$ :  
下一帧延迟从后沿延迟结束开始。在此过程中，不产生移位时钟脉冲，信号 MSLS 保持无效。下一帧延迟结束之后，数据帧视为结束。

### 21.4.3.3 自动从控选择更新

若 SSC 的帧长和字长由位域 SCTR.H.FLE 和 SCTR.H.WLE 定义，可利用发送控制信息 TCI 来更新从控选择设置 PCR.H.CTR[23:16]，从而控制 SELOx 选择输出。通过 TCSR.L.SELMD = 1 使能自动更新机制（位 TCSR.L.WLEMD、FLEMD 和 WAMD 必须清零）。在这种情况下，一帧中第一个数据字的 TCI 定义整个帧的从控选择设置（见 [页 21-101](#)）。

### 21.4.3.4 从控选择延迟的产生

基于时间单元产生从控选择延迟。时间单元的长度（由  $f_{CTQIN}$  时钟周期决定）和每种延迟中时间单元的个数可编程。

在标准 SSC 应用中，前沿延迟  $T_{ld}$  和后沿延迟  $T_{ld}$  主要用于确保输入和输出线上信号稳定、并将输入级的建立和保持时间考虑在内。这两种延迟的长度相等（多数情况下小于一个位时间），可由同一组位域设定。

- **BRGL.CTQSEL**  
定义用于产生  $T_{ld}$  和  $T_{ld}$  的时间单元的输入频率  $f_{CTQIN}$
- **BRGL.PCTQ**  
定义用于产生  $T_{ld}$  和  $T_{ld}$  的时间单元的长度（ $f_{CTQIN}$  经 1、2、3 或 4 分频）
- **BRGL.DCTQ**  
定义用于产生  $T_{ld}$  和  $T_{ld}$  的时间单元的个数

字内延迟  $T_{iw}$  和下一帧延迟  $T_{nf}$  用于处理接收到的数据或准备下一个数据。这两种延迟的长度相等（多数情况下小于一个位时间），可由同一组位域设定。

- **PCRL.CTQSEL1**  
定义用于产生  $T_{nf}$  和  $T_{iw}$  的时间单元的输入频率  $f_{CTQIN}$
- **PCRL.PCTQ1**  
定义用于产生  $T_{nf}$  和  $T_{iw}$  的时间单元的长度（ $f_{CTQIN}$  经 1、2、3 或 4 分频）
- **PCRL.DCTQ1**  
定义用于产生  $T_{nf}$  和  $T_{iw}$  的时间单元的个数
- **PCR.H.TIWEN**  
使能/禁止字内延迟  $T_{iw}$

每种延迟由时间单元的长度和时间单元的个数决定，通过位域 CTQSEL/CTQSEL1、PCTQ/DCTQ 和 PCTQ1/DCTQ1 设定。为了能够灵活设定延迟长度，输入频率可有多种选择（如基于位时间或协议相关分频器较高的频率输入）。延迟时间定义如下：

$$T_{ld} = T_{ld} = \frac{(PCTQ+1) \times (DCTQ+1)}{f_{CTQIN}}$$

$$T_{iw} = T_{nf} = \frac{(PCTQ1+1) \times (DCTQ1+1)}{f_{CTQIN}} \quad (21.9)$$

#### 21.4.3.5 协议中断事件

在 SSC 模式下，可产生以下协议相关事件并导致协议中断。这些事件和一帧数据的起始和结束相关。一旦开始传送一帧数据，则可对下一帧数据进行新的设置；一帧数据结束后，可修改 SSC 的引脚连接。

需要注意：为了监控新的事件，寄存器 PSR 不能由硬件自动清零，必须由软件清零。

- **MSLS 中断：**

该中断指示在主控模式下（MSLS 产生使能），已开始传送一帧数据（MSLS 激活）或一帧数据已结束（MSLS 失效）。内部 MSLS 信号的任何改变都会置位 PSR.MSLSEV。若 PCRL.MSLSIEN = 1，可产生协议中断。可通过 PSR.MSLS 读取 MSLS 信号的状态，从而在检测到该中断时执行恰当的操作。

- **DX2T 中断：**

该中断监控 DX2 级输入信号的跳变（尽管该信号不用作从控选择输入）。

检测到 DX2 输入信号发生跳变时（跳变沿可编程），置位 PSR.DX2TEV。若 PCRL.DX2TIEN = 1，可产生协议中断。可通过 PSR.DX2S 读取所选输入信号的状态，从而在检测到该中断时执行恰当的操作。

#### 21.4.3.6 数据帧结束控制

主控制件的 MSLS 发生器需要帧长信息。可基于每帧中数据位的个数来控制数据帧结束（通过 SCTR.H.FLE < 63 选择该机制），此外，还可通过以下机制控制数据帧结束。建议设置 SCTR.H.FLE = 63（若同时使用多种数据帧结束控制机制，首先发现的结束条件将结束数据帧）。

- **基于软件控制的数据帧开始指示 TCSRL.SOF：**

若由软件处理不带数据 FIFO 的 TBUF 数据，可使用该机制。若 SOF 置位，TBUF 中的有效内容被看作是新一帧中的第一个字。TBUF 的内容在传送到发送移位寄存器之前，位 SOF 必须置位，因此，建议在向 TBUF 写入数据之前设置 SOF。若要开始传送新的数据帧（之前插入 T<sub>id</sub>），必须先结束当前数据字的传送、并插入从控选择延迟 T<sub>id</sub> 和 T<sub>nf</sub>。

若由软件处理 SOF，必须设置 TCSRL.WLEMD = 0。在这种情况下，所有 TBUF[31:0]地址单元表现为相同的行为（数据传送不使用 TCI 信息）。

- **基于软件控制的数据帧结束指示 TCSRL.EOF：**

若由软件处理不带数据 FIFO 的 TBUF 数据，可使用该机制。若 EOF 置位，TBUF 中的有效内容被看作是新一帧中的最后一个字。TBUF 的内容在传送到发

送移位寄存器之前，位 EOF 必须置位，因此，建议在向 TBUF 写入数据之前设置 EOF。TBUF 中的数据字完全送出之后，插入从控选择延迟  $T_{id}$  和  $T_{nf}$ 。随后可开始传送新的数据帧（之前插入  $T_{id}$ ）。

若由软件处理 EOF，必须设置 TCSRL.WLEMD = 0。在这种情况下，所有 TBUF[31:0]地址单元表现为相同的行为（数据传送不使用 TCI 信息）。

- 基于软件控制的和地址相关的数据帧结束：

若由软件处理不带数据 FIFO 的 TBUF 数据，可使用该机制。若位 TCSRL.WLEMD = 1，被写入的 TBUF[31:0]的地址单元可用作发送控制信息 TCI[4:0]，用于更新每个数据字的 SCTRH.WLE (=TCI[3:0]) 和 TCSRL.EOF (=TCI[4])。被写入的 TBUF[31:0]的地址定义了字长和数据帧结束信息（TBUF[31:16]导致数据帧结束）。

例如，若向 TBUF[07]写入发送数据，则对应一个 8 位的数据字，此时数据帧未结束；若向 TBUF[31]写入发送数据，则对应一个 16 位的数据字，随后紧跟  $T_{id}$ 、MSLS 失效和  $T_{nf}$ 。

若 TCSRL.WLEMD = 1，向 TBUF 写入数据之后，切勿软件设置位 TCSRL.EOF、TCSRL.SOF 和 SCTRH.WLE。此外，建议清除 TCSRL.SELMD、FLEMD 和 WAMD。

- 基于 FIFO 的和地址相关的数据帧结束：

若使用数据 FIFO 来保存发送数据，可使用该机制。其行为和前一种机制相似，区别仅在于将发送数据写入 FIFO 输入单元 IN[31:0]、而不是 TBUF[31:0]。在这种情况下，切勿软件操作任何 TBUF 单元。

- 和 TBUF 相关的数据帧结束：

若位 PCRL.FEM = 0，发送一个数据字之后若 TBUF 中不包含有效数据，则认为数据帧结束（TCSRL.TDV = 0 或处于终止模式）。在这种情况下，软件必须确保运行模式下在发送一帧数据的过程中 TBUF 不空。若 PCRL.FEM = 1，当发送缓存正在等待新数据（再次 TCSRL.TDV = 1）或在退出终止模式之前，信号 MSLS 始终保持有效。

- 软件控制数据帧结束：

可由软件结束数据帧，通过向寄存器 PSCR 中的相关位置 1 以清除 PSR.MSLS 来实现。该写操作将立刻清除 PSR.MSLS，但在完成当前数据字的传送、并插入从控选择延迟  $T_{id}$  和  $T_{nf}$  之后，信号 MSLS 才变为无效。

#### 21.4.4 从控模式下操作 SSC

在从控模式下操作 SSC 时，必须考虑以下因素：

- 选择 SSC 模式：

建议在 CCR.MODE = 0000<sub>B</sub> 时配置 SSC 的所有参数（在运行期间这些参数不改变）。必须设置 SCTRL.TRM = 01<sub>B</sub>。必须在 CCR.MODE = 0000<sub>B</sub> 时配置输



入级，从而避免输入信号上出现的未预计跳变，随后可通过  $\text{CCR.MODE} = 0001_{\text{B}}$  使能 SSC 模式。

- 引脚连接：

通过  $\text{DX0CR.INSW} = 1$  建立输入级 DX0 和所选接收数据输入引脚（信号 DIN）之间的连接；配置发送数据输出引脚（信号 DOUT）。

通过  $\text{DX1CR.INSW} = 1$  建立输入级 DX1 和所选移位时钟输入引脚（信号 SCLKIN）之间的连接。

通过  $\text{DX2CR.INSW} = 1$  建立输入级 DX2 和所选从控选择输入引脚（信号 SELIN）之间的连接。若未使用从控选择输入信号，DX2 必须向数据移位单元发送 1 电平以允许数据接收和发送。若某从控器件未被选中（DX2 级向数据移位单元发送 0 电平）而接收到移位时钟脉冲，则不接收送入的数据，DOUT 信号输出由  $\text{SCTRL.PDL}$  定义的被动数据电平。

- 波特率产生：

不需波特率发生器，可通过分数分频器关闭。

- 从控选择的产生：

不需产生从控选择延迟，可被关闭。不需使用寄存器  $\text{PCRL/PCRH}$  中的位/位域  $\text{MSLSEN}$ 、 $\text{SELCTR}$ 、 $\text{SELINV}$ 、 $\text{CTQSEL1}$ 、 $\text{PCTQ1}$ 、 $\text{DCTQ1}$ 、 $\text{MSLSIEN}$ 、 $\text{SELO}[7:0]$  和  $\text{TIWEN}$ ，可设置为 0。

#### 21.4.4.1 协议中断

在 SSC 模式下，可产生以下协议相关事件并导致协议中断。这些事件和一帧数据的起始和结束相关。一旦开始传送一帧数据，则可对下一帧数据进行新的设置；一帧数据结束后，可修改 SSC 的引脚连接。

需要注意：为了监控新的事件，寄存器  $\text{PSR}$  不能由硬件自动清零，必须由软件清零。

- MSLS 事件：

MSLS 产生被关闭，该事件不可用。

- DX2T 事件：

从控选择输入信号 SELIN 由 DX2 级处理，所选信号的跳变沿可产生协议中断。该中断可指示已开始或/和已完成一帧数据的传送。

检测到 DX2 输入信号发生跳变时（跳变沿可编程），置位  $\text{PSR.DX2TEV}$ 。若  $\text{PCRL.DX2TIEN} = 1$ ，可产生协议中断。可通过  $\text{PSR.DX2S}$  读取所选输入信号的状态，从而在检测到该中断时执行恰当的操作。

#### 21.4.4.2 数据帧结束控制

从控模式下，可由以下方式决定帧长。从控器件必须参照外部从控选择信号、或参照接收到的数据位个数。

- 从控器件已预知帧长，无从控选择：  
在这种情况下，可将 SCTRH.FLE 设置为已知值（若不超过 63）。达到所设定的帧长时，认为当前传送的数据字结束。
- 从控器件不知帧长，无从控选择：  
在这种情况下，从控器件的软件必须基于数据字决定数据帧是否结束。可将 SCTRH.FLE 设置为字长 SCTRH.WLE 或其最大值以关闭从控内部帧长评估。
- 由从控选择信号 SELIN 寻址从控器件：  
若由主控器件发送的从控选择信号寻址从控器件，则由该信号给出数据帧的开始和结束信息。  
在这种情况下，应将 SCTRH.FLE 设置为其最大值以关闭从控内部帧长评估。

#### 21.4.5 SSC 协议寄存器

SSC 模式下，寄存器 PCRH、PCRL 和 PSR 用于处理 SSC 相关信息。

##### 21.4.5.1 SSC 协议控制寄存器

SSC 模式下的寄存器 PCRL/PCRH 定义如下：

##### PCRL

协议控制寄存器 L [SSC 模式] (40<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DX2 TIEN	MSL SIEN	0	DCTQ1				PCTQ1		CTQSEL1	FEM	SEL INV	SEL CTR	MSL SEN		
rw	rw	r	rw				rw		rw	rw	rw	rw	rw	rw	

符号	位序号	类型	功能描述
MSLSEN	0	rw	<b>MSLS 使能</b> 该位使能/禁止产生主从选择信号 MSLS。若 SSC 是传送从控器件，则从引脚上读取 SLS 信息，不需内部产生 MSLS。若 SSC 是传送主控器件，它必须提供 MSLS 信号。 0 <sub>B</sub> 禁止产生 MSLS (MSLS = 0)。

符号	位序号	类型	功能描述
			<p>该设置用于 SSC 从控模式。</p> <p>1<sub>B</sub> 使能产生 MSLS。</p> <p>该设置用于 SSC 主控模式。</p>
<b>SELCTR</b>	1	rw	<p><b>选择控制</b></p> <p>该位选择 SELO[7:0]输出的操作模式。</p> <p>0<sub>B</sub> 编码选择模式。</p> <p>1<sub>B</sub> 直接选择模式。</p>
<b>SELINV</b>	2	rw	<p><b>信号反相</b></p> <p>该位定义 SELO[7:0]输出和主从选择信号 MSLS 之间的极性关系。</p> <p>0<sub>B</sub> SELO 输出和 MSLS 信号的极性相同（高有效）。</p> <p>1<sub>B</sub> SELO 输出和 MSLS 信号的极性相反（低有效）。</p>
<b>FEM</b>	3	rw	<p><b>数据帧结束模式</b></p> <p>若发送缓存中不包含有效数据，该位定义是否将其看作是数据帧结束的条件。</p> <p>0<sub>B</sub> 若当前数据帧中的最后一位数据已送出、且发送缓存 TBUF 不包含新数据（TDV = 0），该帧视为结束。</p> <p>1<sub>B</sub> 当无新数据可用、且未达到其它数据帧结束条件时，MSLS 信号仍保持有效。在这种情况下，对于多字组成的数据帧，软件可接受发送数据有延迟（不自动使 MSLS 失效）。</p>
<b>CTQSEL1</b>	[5:4]	rw	<p><b>输入频率选择</b></p> <p>该位域定义用于产生从控选择延迟 <math>T_{nf}</math> 和 <math>T_{iw}</math> 的输入频率 <math>f_{CTQIN}</math>。</p> <p>00<sub>B</sub> <math>f_{CTQIN} = f_{PDIV}</math></p> <p>01<sub>B</sub> <math>f_{CTQIN} = f_{PPP}</math></p> <p>10<sub>B</sub> <math>f_{CTQIN} = f_{SCLK}</math></p> <p>11<sub>B</sub> <math>f_{CTQIN} = f_{MCLK}</math></p>
<b>PCTQ1</b>	[7:6]	rw	<p><b>用于产生 <math>T_{nf}</math> 和 <math>T_{iw}</math> 的分频因子 PCTQ1</b></p>

符号	位序号	类型	功能描述
			该位域定义用于产生字内延迟和下一帧延迟的分频因子 PCTQ1（取值范围 = 0 - 3）。 $T_{iw} = T_{nf} = 1 / f_{CTQIN} \times (PCTQ1+1) \times (DCTQ1+1)$
<b>DCTQ1</b>	[12:8]	rw	<b>用于产生 <math>T_{nf}</math> 和 <math>T_{iw}</math> 的分频因子 DCTQ1</b> 该位域定义用于产生字内延迟和下一帧延迟的分频因子 DCTQ1（取值范围 = 0 - 3）。 $T_{iw} = T_{nf} = 1 / f_{CTQIN} \times (PCTQ1+1) \times (DCTQ1+1)$
<b>MSLSIEN</b>	14	rw	<b>MSLS 中断使能</b> 若 MSLS 信号的状态改变（由 PSR.MSLSEV = 1 指示），该位使能/禁止产生协议中断。 0 <sub>B</sub> 检测到 MSLS 信号改变时不产生协议中断。 1 <sub>B</sub> 检测到 MSLS 信号改变时产生协议中断。
<b>DX2TIEN</b>	15	rw	<b>DX2T 中断使能</b> 若 DX2T 信号被激活（由 PSR.DX2TEV = 1 指示），该位使能/禁止产生协议中断。 0 <sub>B</sub> 若 DX2T 被激活，不产生协议中断。 1 <sub>B</sub> 若 DX2T 被激活，产生协议中断。
<b>0</b>	13	r	<b>保留</b> 读操作返回 0；应写入 0。

## PCRH

协议控制寄存器 H [SSC 模式]

(42<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>M CLK</b>	<b>0</b>					<b>TIW EN</b>	<b>SELO</b>								
rw	r					rw	rw								

符号	位序号	类型	功能描述
<b>SELO</b>	[7:0]	rw	<b>选择输出</b> 该位域定义 SELO[7:0]输出线的设置。 0 <sub>B</sub> SELO <sub>x</sub> 输出线不能被激活。

符号	位序号	类型	功能描述
			1 <sub>B</sub> SELOx 输出线可以被激活。 (根据由 SELCTR 选择的模式)
<b>TIWEN</b>	8	rw	<b>使能字内延迟 T<sub>iw</sub></b> 该位使能/禁止发送数据字之后产生字内延迟 T <sub>iw</sub> 。 0 <sub>B</sub> 同一帧的数据字之间无延迟。 1 <sub>B</sub> 在同一帧的数据字之间插入字内延迟 T <sub>iw</sub> 。
<b>MCLK</b>	15	rw	<b>主控时钟使能</b> 该位使能/禁止产生主控时钟输出信号 MCLK，和主控或从控模式无关。 0 <sub>B</sub> 禁止产生 MCLK，输出 MCLK = 0。 1 <sub>B</sub> 使能产生 MCLK。
<b>0</b>	[14:9]	r	<b>保留</b> 读操作返回 0；在 SSC 模式下不修改。

#### 21.4.5.2 SSC 协议状态寄存器

SSC 模式下的寄存器 PSR 定义如下。PSR 不被硬件清零。

向寄存器 PSCR 写 1 时，PSR 中相应的标志位被清零；向 PSR 写 1 将置位相应的标志位，但不会引发其它操作（不产生中断）。写 0 不起作用。在使能新协议之前，PSR 标志位应由软件清零。

#### PSR

协议状态寄存器 [SSC 模式]

(44<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>AIF</b>	<b>RIF</b>	<b>TBIF</b>	<b>TSIF</b>	<b>DLIF</b>	<b>RSIF</b>	<b>0</b>						<b>DX2 TEV</b>	<b>MSL SEV</b>	<b>DX2 S</b>	<b>MSL S</b>
rw	rw	rw	rw	rw	rw	r						rw	rw	rw	rw

符号	位序号	类型	功能描述
<b>MSLS</b>	0	rw	<b>MSLS 的状态</b> 该位指示 MSLS 信号的当前状态。该位必须由软件清零以终止正在传送的数据帧。

符号	位序号	类型	功能描述
			<p>0<sub>B</sub> 内部信号 MSLS 失效 (0)。</p> <p>1<sub>B</sub> 内部信号 MSLS 有效 (1)。</p>
<b>DX2S</b>	1	rwh	<p><b>DX2S 的状态</b></p> <p>该位指示 DX2S 信号的当前状态，该信号可用作从控选择输入 SELIN。</p> <p>0<sub>B</sub> DX2S 为 0。</p> <p>1<sub>B</sub> DX2S 为 1。</p>
<b>MSLSEV</b>	2	rwh	<p><b>MSLS 事件检测<sup>1)</sup></b></p> <p>该位指示自 MSLSEV 被清零后 MSLS 信号的状态已改变。结合 MSLS 的状态位，可监控 MSLS 信号被激活/失效。</p> <p>0<sub>B</sub> MSLS 信号还未改变其状态。</p> <p>1<sub>B</sub> MSLS 信号已改变其状态。</p>
<b>DX2TEV</b>	3	rwh	<p><b>DX2T 事件检测<sup>1)</sup></b></p> <p>该位指示自 DX2TEV 被清零后 DX2T 触发信号是否被激活。</p> <p>0<sub>B</sub> DX2T 信号还未被激活。</p> <p>1<sub>B</sub> DX2T 信号已被激活。</p>
<b>0</b>	[9:4]	r	<p><b>保留</b></p> <p>读操作返回 0；在 SSC 模式下不修改。</p>
<b>RSIF</b>	10	rwh	<p><b>接收开始指示标志</b></p> <p>0<sub>B</sub> 未发生接收开始事件。</p> <p>1<sub>B</sub> 已发生接收开始事件。</p>
<b>DLIF</b>	11	rwh	<p><b>数据丢失指示标志</b></p> <p>0<sub>B</sub> 未发生数据丢失事件。</p> <p>1<sub>B</sub> 已发生数据丢失事件。</p>
<b>TSIF</b>	12	rwh	<p><b>发送移位指示标志</b></p> <p>0<sub>B</sub> 未发生发送移位事件。</p> <p>1<sub>B</sub> 已发生发送移位事件。</p>

符号	位序号	类型	功能描述
<b>TBIF</b>	13	rwh	<b>发送缓存指示标志</b> 0 <sub>B</sub> 未发生发送缓存事件。 1 <sub>B</sub> 已发生发送缓存事件。
<b>RIF</b>	14	rwh	<b>接收指示标志</b> 0 <sub>B</sub> 未发生接收事件。 1 <sub>B</sub> 已发生接收事件。
<b>AIF</b>	15	rwh	<b>备选接收指示标志</b> 0 <sub>B</sub> 未发生备选接收事件。 1 <sub>B</sub> 已发生备选接收事件。

1) SSC 模式下该状态位可产生协议中断（见[页 21-21](#)）。中断状态标志的一般描述请参见中断章节。

## 21.4.6 SSC 时序

为了确保数据的正确接收和发送，输入和输出信号必须考虑时序问题。除模块内部时序（由输入滤波、事件响应时间等引起）之外，还必须考虑从输入引脚经输入级到模块的时序（ $T_{in}$ ）、从模块经输出驱动到引脚的时序（ $T_{out}$ ）、以及信号线上的传播时间（ $T_{prop}$ ）。

需要注意：由于数字滤波器和同步电路会造成系统延迟，因此  $DXn$  输入级可能会有附加延迟。若使用这些功能，则必须考虑该延迟。

### 21.4.6.1 闭环延迟

系统固有的闭环延迟会限制 SSC 通信的波特率。在一种典型应用中，工作在全双工模式的主控和从控器件通过两条独立的数据线发送和接收数据。一般情况下，所有的发送器使用一个移位时钟沿发送、所有的接收器使用另一个移位时钟沿接收。主控制器件送出发送数据、移位时钟和从控选择信号（可选）。因此，波特率发生器（BRG）和从控选择产生（SSC）是主控制器件的一部分。主控和从控模式下 SSC 模块的帧控制相似，主要区别在于哪个模块产生移位时钟以及从控选择信号（可选）。

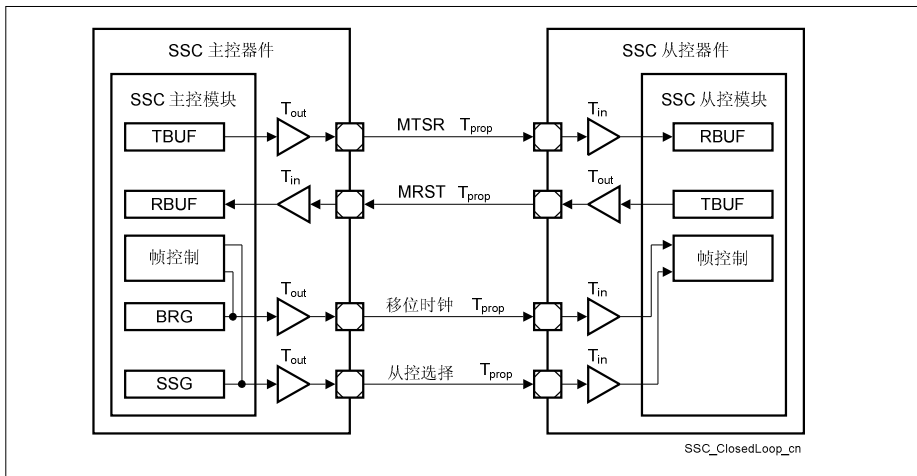


图 21-38 SSC 闭环延迟

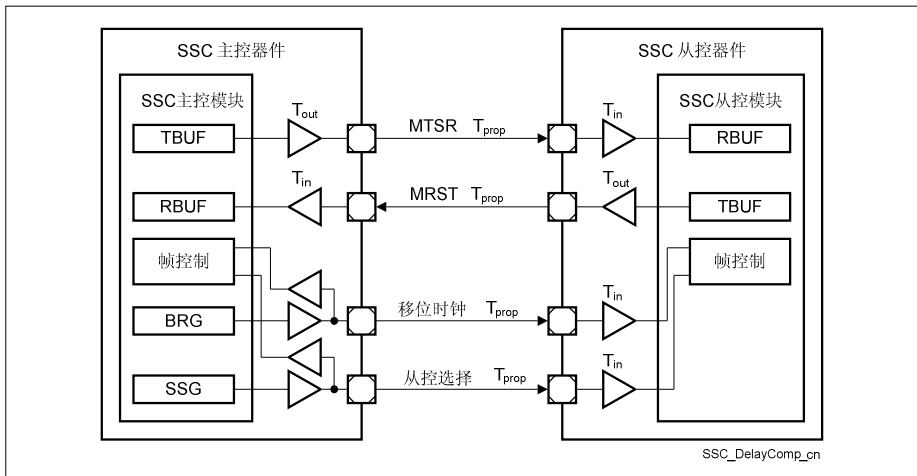
主控和从控 SSC 模块之间的信号通路包括主控输出驱动、主控和从控制器的连线以及从控制器的输入级。从控制器接收到移位时钟沿后，开始接收主控器件发送的数据并向其返回自己的数据。主控器件基于其内部移位时钟沿接收从控制器发送的数据。为了确保主控器件能正确接收数据，当它使用内部的下一个移位时钟沿接收数据时，从控制器的发送数据必须保持稳定（考虑建立和保持时间）（通常 1/2 移位时钟周期）。为了避免数据遭破坏，输入和输出级的累积延迟、信号在信号线上的传播时间、以及发送器/接收器的响应时间都必须考虑在内，尤其对于高波特率的情况。

在给出的举例中，主控 SSC 模块在产生移位时钟和评估接收数据之间的时间间隔为  $(T_{out\_master} + 2 \times T_{prop} + T_{in\_slave} + T_{out\_slave} + T_{in\_master} + \text{模块响应时间} + \text{输入建立时间})$ 。输入通路的延迟主要由引出端上输入级的特性决定；输出通路的延迟由输出驱动延迟及其摆率、驱动器的外部负载和电流驱动能力决定。输入/输出驱动器的具体参数请参阅数据手册。

#### 21.4.6.2 主控模式下的延迟补偿

在主控模式下，可通过延迟补偿达到更高的波特率。如果（至少）移位时钟引脚为双向引脚，可进行延迟补偿。





**图 21-39 带有延迟补偿的 SSC 主控模式**

若主控模式下的移位时钟信号直接取自输出信号的输入功能，主控器件移位时钟输出的延迟被补偿。在这种情况下，在移位时钟通路上，只需考虑主控和从控器件输入延迟的差值，不需考虑整个主控输出延迟和从控输入延迟。

在给出的举例中，主控 SSC 模块产生移位时钟信号和评估接收数据之间的时间间隔将减少 ( $T_{in\_maste} + T_{out\_master}$ )。

尽管工作在主控模式，移位时钟输入和可选的从控选择信号不和数据移位单元内部直连，而是从输入引脚取值 ( $DXnCR.INSW = 1$ )。若移位时钟输出引脚（以及从控选择输出引脚）是双向引脚，延迟补偿不会导致额外的引脚。在这种情况下，由于输入信号和引脚本身的信号电平有关，因此输入信号和其它内部信号无关。

## 21.5 IIC 总线协议 (IIC)

USIC 的 IIC 协议参照 IIC 总线规范 V2.1（飞利浦半导体 2000 年 1 月发布）。和 IIC 总线规范有所不同，USIC 器件假定在所有模式下，总线信号的上升/下降时间最大为 300 ns。关于总线的驱动能力，请参阅 AC/DC 一章中的引出端特性。不支持 CBUS 模式和 HS 模式。

若  $CCFG.IIC = 1$ （IIC 模式可用），由  $CCR.MODE = 0100_B$  选择 IIC 模式。

本章包含以下内容：

- 简介（见[页 21-135](#)）
- 操作 IIC 协议（见[页 21-137](#)）

- 符号时序和编程（见页 21-143）
- 数据流处理（见页 21-146）
- IIC 协议寄存器（见页 21-150）

### 21.5.1 简介

USIC IIC 特性:

- 双线接口，一条用于移位时钟传送和同步（移位时钟 SCL），一条用于数据传送（移位数据 SDA）
- 通信模式：标准模式（100 kBit/s）或快速模式（高达 400 kBit/s）
- 支持 7 位寻址和 10 位寻址
- 主控模式操作：  
IIC 控制总线处理并提供时钟信号。
- 从控模式操作：  
外部主控器件控制总线处理并提供时钟信号。
- 多主控模式操作：  
多个主控器件可同时连接到总线上，通过总线仲裁决定 IIC 模块是主控还是从控器件。主控/从控操作可逐帧改变。
- 有效的帧处理（低软件开销），还允许 PEC 传送
- 功能强大的中断处理（由于引入大量的指示标志位）
- 支持输入延迟补偿

#### 21.5.1.1 信号描述

IIC 的特性为双线连接（SDA 和 SCL）。这些信号的输出驱动必须具备漏极开路特性，从而使得所有 SDA 线和所有 SCL 线可构成线与连接。由于具备这样的结构，输出级驱动的高电平未必会立即在相应输入上产生高电平。因此，每条 SDA 或 SCL 连接必须同时为输入和输出，由于输入功能始终监控信号的电平（在发送时也如此）。

- 移位数据 SDA：由 DX0 处理的输入，输出信号 DOUT
- 移位时钟 SCL：由 DX1 处理的输入，输出信号 SCLKOUT

IIC 总线参与方（模块 IIC A 和 IIC B）的连接如图 21-40 所示。在该示例中，模块 IIC A 的 SDA 和 SCL 的输入和输出信号分别使用独立、分开的引脚。若实际应用中 DOUT 和 DX0 输入不使用同一个引脚（SCLKOUT 和 DX1 类似），可使用这种引脚分配。模块 IIC B 的引脚分配显示 DOUT 和 DX0 输入使用同一个引脚，SCLKOUT 和 DX1 同样也使用同一个引脚。

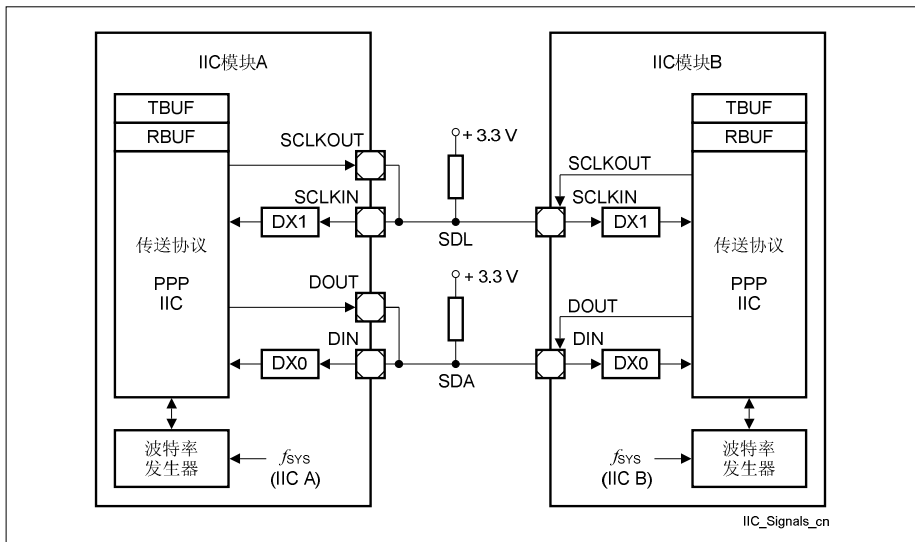


图 21-40 IIC 信号连接

### 21.5.1.2 符号

符号是指 SDA 和 SCL 线上一系列的跳变沿。根据所选择的波特率，符号可包含 10 或 25 个时间单元  $t_q$ 。波特率发生器决定时间单元  $t_q$  的长度；IIC 协议预处理器处理每个符号中的跳变沿序列；用户可根据应用需求对符号序列进行编程。

符号定义如下:

- 总线空闲:

SDA 和 SCL 为高电平。当前无数据传送。

- 数据位符号:

在 SCL 的高电平阶段 SDA 保持稳定。SDA 代表传送数据位的值。通过 SCL 上的时钟脉冲来传送每一位数据。在数据传送期间，只有当 SCL 为低电平时 SDA 才能改变。

- 开始符号:

在 SCL 为高电平的情况下，若信号 SDA 首先为高、随后跳变为低电平，则指示满足开始条件。在总线空闲之后，该开始条件启动数据传送。

- 重新开始符号:

当总线还未空闲时，该开始条件启动在一个数据符号之后进行数据传送。因此，SDA 置高、SCL 为低，随后为开始符号。

- 终止符号:

在 SCL 为高电平的情况下, 若信号 SDA 跳变为高电平, 则指示满足终止条件。该终止条件终止数据传送, 释放总线使其处于空闲状态。在开始和终止条件之间, 可传送任意数目的字节。

### 21.5.1.3 帧格式

通过双线 (SDA、SCL) IIC 总线协议可确保可靠、高效的数据传送。发送方发送 (数据) 字节之后, 接收并检查随后应答域的值。由于 IIC 是线与总线系统, 因此只要有一个器件发送 0, 总线上即为 0, 该值会被所有器件接收。

一个数据字包含 8 个数据位符号、随后是一个应答数据位符号。数据字可被解读为地址信息 (在开始符号之后) 或传送数据 (在地址之后)。

为了能够接收应答信号, 数据位的发送方必须发送 1 (作为应答值) 以释放 SDA 线。应答位可以是主动电平也可以是被动电平, 这取决于接收器的内部状态。

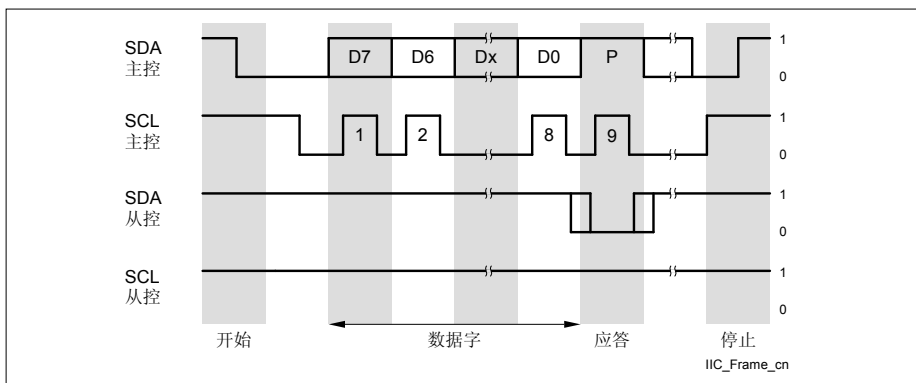


图 21-41 IIC 帧格式举例 (简化格式)

### 21.5.2 操作 IIC

为了执行 IIC 协议, 必须考虑以下因素:

- 选择 IIC 模式:

建议在 CCR.MODE = 0000<sub>B</sub> 时配置 IIC 的所有参数 (在运行期间这些参数不改变)。应设置 SCTRL.TRM = 11<sub>B</sub>。必须在 CCR.MODE = 0000<sub>B</sub> 时配置输入级, 从而避免输入信号上出现的未预计跳变, 随后可通过 CCR.MODE = 0100<sub>B</sub> 使能 IIC 模式。

- 引脚连接:

通过  $DX0CR.INSW = 0$  建立输入级  $DX0$  ( $DX0CR.DPOL = 0$ ) 和移位数据引脚  $SDA$  (信号  $DIN$ ) 之间的连接; 将发送数据输出信号  $DOUT$

( $SCTRL.DOCFG = 00_B$ ) 配置到同一个引脚。若该引脚可用, 它可同时用作输入和输出, 或将输入和输出引脚相连构成  $SDA$  线。

移位时钟线  $SCL$  应用相同的机制。必须将信号  $SCLKOUT$  ( $BRGH.SCLKCFG = 00_B$ ) 和  $DX1$  的输入 ( $DX1CR.DPOL = 0$ ) 相连。

IIC 协议中不使用输入级  $DX2$ 。

若  $DX0/1$  使能数字输入滤波器, 为了正确计算信号时序, 必须考虑该延迟。

用作  $SDA$  和  $SCL$  的引脚必须置为漏极开路模式以支持 IIC 总线的线与结构。

需要注意: 通过  $CCR.MODE = 0100_B$  使能 IIC 模式之前, 还必须正确配置用作 IIC I/O 引脚的 I/O 端口。

- 位时序配置:

在标准模式下 ( $100\text{ kBit/s}$ ), 要求最低模块频率为  $2\text{ MHz}$ ; 在快速模式下 ( $400\text{ kBit/s}$ ), 要求最低模块频率为  $10\text{ MHz}$ 。此外, 若使用数字滤波器来消除小于  $50\text{ ns}$  的毛刺, 滤波器频率需  $20\text{ MHz}$ 。

若另一个 IIC 参与方延长  $SCL$  低电平阶段, 则  $SCL$  高电平阶段会产生最大为  $1/f_{PPP}$  的不确定的时序误差。

具体内容请参阅 [章节 21.5.3](#)。

- 数据格式配置:

数据格式必须配置如下:  $8$  个数据位 ( $SCTRH.WLE = 7$ )、数据流长度不限 ( $SCTRH.FLE = 3FF_H$ )、MSB 移位在先 ( $SCTRL.SDIR = 1$ )。必须禁用奇偶校验 ( $CCP.PM = 00_B$ )。

- 一般提示:

若主控器件发送的地址选中某一从控器件, 则该从控模块变为有效 (接收或发送)。从控器件向主控器件发送数据时使用发送通路。因此, 为了避免冲突, 主控器件切勿请求从本通道定义的从控地址上读取数据。

只有当 IIC 模块参与 IIC 总线通信时, 内置的检错机制才被激活。

若从控器件无法处理过高的频率, 它可延长  $SCL$  信号的低电平阶段。

遵照 IIC 规范进行数据传送时, 只有当  $SCL = 0$  时才能改变移位数据线  $SDA$  (由 IIC 总线规范定义)。

### 21.5.2.1 发送链

在仲裁阶段以及从控器件发送期间, IIC 总线协议需要一种“位内响应”, 发送链环路延迟会限制可达到的最大波特率, 这主要取决于总线特性 (总线负载、模块频率等)。

**图 21-40** 给出从控器件发送的一般信号通路和延迟。主控器件产生移位时钟 **SCL**，输出到总线上，然后送至输入级和输入滤波器。此时，从控器件可检测到跳变沿并相应产生 **SDA** 数据信号。**SDA** 信号经输出级和总线送至主控器件的接收部分，然后经输入级和输入滤波器之后被采样。

必须在 **SCL** 信号改变之前完成上述过程（包括信号达到稳定需要的时间）。在计算以  $f_{\text{SYS}}$  和  $f_{\text{PPP}}$  为函数的波特率时，必须将该通路上的延迟考虑在内。

### 21.5.2.2 字节延展

当器件用作收发器时，若应发送数据字节而发送 **TBUF** 中不包含有效数据，则在下一个应答位结束之后保持 **SCL** = 0。若检测到 **TBUF** 中含有新的有效数据，结束等待周期。

### 21.5.2.3 波特率更新

可以逐帧修改波特率设置。**IIC** 总线空闲时采样（映射）**BRGL/H** 寄存器设置和 **PCR.STIM**。在传送数据帧的过程中可编程新设置，在开始下一帧传送时新设置生效。为了尽量避免因修改波特率设置而造成的一致（必须更新多个寄存器），建议用户在使用 **IIC** 协议时不要修改波特率，尤其对于从控器件。

### 21.5.2.4 主控器件仲裁

在发送地址和数据期间，主控方发送器在 **SCL** 的上升沿检测每一位发送数据是否和 **SDA** 线上读取的数据相等。若相等，下一位数据值可为 0。若不相等（发送值 = 1，读取值 = 0），主控方失去发送仲裁。由状态标志 **PSR.ARL** 指示这种情况，若 **PCRH.ARLIEN** = 1，可产生协议中断。

若已失去发送仲裁，软件必须重新初始化整个帧，从第一个地址字节开始，接下来是主控方重新尝试发送的开始条件。仲裁也用于 **ACK** 位。

### 21.5.2.5 释放 TBUF

若不应答或产生错误时，**TBUF** 的内容变为无效。在这两种情况下，软件必须刷新发送缓存并用合适的值对其重新设置以响应前一事件。

### 21.5.2.6 模式控制

在多主控模式下，只支持运行模式 0 和终止模式 0，切勿编程为其它模式。

- 运行模式 0:

执行所设定的操作。当需要处理新的 **TBUF** 入口时，若 **TCSRL.TDV** = 0（未发现新的有效 **TBUF** 入口），**IIC** 模块等待 **TDV** 变为置位以继续执行操作。

- 运行模式 1:

执行所设定的操作。主控模式下，当需要处理新的 TBUF 入口时，若  $TCSR.LTDV = 0$ （未发现新的有效 TBUF 入口），IIC 模块发送终止条件以结束当前帧。从控模式下，和运行模式 0 的操作相同。

- **终止模式 0:**

位  $TCSR.LTDV$  被内部视为 0（该位本身不被终止模式修改）。在主控模式下，正常完成当前的数据字传送，但不启动新数据字传送（等待 TDV 有效）。若 TDV 被主控和从控器件视为 0，当外部主控制器读取从控器件的数据时，从控器件将强制总线为等待状态。

此外，不可能从等待状态强制产生终止条件。原因如下：必须通过不应答以及随后的终止条件来结束主控制器的读取传送，从而允许从控器件释放其 SDA 线。否则，从控器件可能强制 SDA 线为 0（下一个字节的第一位数据），从而不可能产生终止条件（SDA 线的上升沿跳变）。

为了能够继续执行操作，必须将模式切换回运行模式 0。

- **终止模式 1:**

和终止模式 0 相似，区别在于，主控制器将发送 STOP 条件以结束当前帧。

若在 10 位地址的首字节之后主控制器请求终止模式 1，则送出终止条件。在这种情况下，从控器件将产生一个错误中断。

### 21.5.2.7 IIC 协议中断事件

在 IIC 模式下，可产生以下协议相关事件（并导致协议中断）。

需要注意：为了监控新的事件，寄存器 PSR 不能由硬件自动清零，必须由软件清零。

- **发送缓存事件:**

当发送缓存 TBUF 中的内容已载入发送移位寄存器，发送缓存事件指示标志  $PSR.TBIF$  置位，指示已启动 TBUF 入口所请求的操作。

一旦产生发送缓存事件，位  $TCSR.LTDV$  被清零。在处理上一个 TBUF 入口的同时，可利用该中断对下一个 TBUF 入口进行写操作（由发送器部分处理）。

- **接收事件:**

接收事件指示标志  $PSR.RIF$  指示一个新的数据字节已写入接收缓存 RBUF0/1（新一帧中首个地址字节除外，它由备选接收中断指示）。接收到数据字节时，该标志位置位。在处理新的数据字节的同时，可利用该中断读出接收到的数据（由接收器部分处理）。

- **备选接收事件:**

备选接收事件指示标志 AIF 基于位  $RBUF.SR[9]$ （和  $RBUF[9]$  相同），指示接收到的数据字是新一帧数据中的第一个数据字。

- **协议中断事件:**

IIC 协议相关中断事件包括：接收到符号、检测到帧错误（由 PSR.ERR 统一指示）或出现不期望的/错误的 TDF 编码（由 PSR.WTDF 统一指示）。

- 在正确位置处接收到开始条件（PSR.SCR）
- 在一帧的正确位置处接收到重新开始条件（PSR.RSCR）
- 在一帧的正确位置处传送终止条件（PSR.PCR）
- 主控仲裁丢失（PSR.ARL）
- 从控读取请求（PSR.SRR）
- 接收到无应答（PSR.NACK）
- 开始条件未出现在一帧中期望的位置处（PSR.ERR）
- 终止条件未出现在一帧中期望的位置处（PSR.ERR）
- 作为从控器件，10 位地址在收送时在首地址字节之后被终止条件中断
- 主控模式下的 TDF 从控编码（PSR.WTDF）
- 从控模式下的 TDF 主控编码（PSR.WTDF）
- 发现预留的 TDF 编码（PSR.WTDF）
- 主控模式下，在传送一帧的过程中出现开始条件编码（PSR.WTDF）
- 主控模式下，传送方向变为接收之后（主控读取）的数据字节发送编码（PSR.WTDF）

若 TBUF 中有错误的 TDF 编码，直到 TDF 值被纠错或无效之后错误事件才生效。若使能相关中断，中断处理器在处理其它可能的中断事件之前，应首先检查 PSR.WTDF 并更正 TBUF 或使其失效。

### 21.5.2.8 接收器地址应答

在（重复）开始条件之后，主控器件发送一个从控地址以识别通信的目标器件。该地址可包含一个或两个地址字节（用于 7 位或 10 位寻址）。在一个地址字节之后，被寻址的从控器件必须作出接收应答。

因此，可在器件中编程设定从控地址并和接收到的地址进行比较。假若地址匹配，从控器件将对主控器件作出应答（SDA = 0）。未选中的非目标从控器件将通过不应答（SDA = 1）对主控器件作出响应。若从控器件有能力处理相应的请求，除编程地址匹配之外，必须对另一个地址字节作出应答。地址字节 00<sub>H</sub> 代表广播呼叫地址，可被应答；地址字节 01<sub>H</sub> 代表开始字节的产生，不被应答。

为了能对不同的地址字节进行选择性的应答，USIC IIC 实现了以下控制机制：

- 若位 PCRH.ACK00 置位，对地址字节 00<sub>H</sub> 作出应答。
- 地址字节 01<sub>H</sub> 不被应答。
- 接收到的第一个地址字节的前 7 位和编程设定的从控地址（PCR.SLAD[15:9]）进行比较。若地址匹配，从控器件发送应答。此外，若从控地址设定为 1111



0XX<sub>B</sub>，从控器件等待另一个地址字节，将它和 PCR.SLAD[7:0]进行比较并发送应答，该操作针对 10 位寻址模式。用户必须谨慎处理预留地址（具体内容请参阅 IIC 规范）。只支持地址 1111 0XX<sub>B</sub>。

在以上这些情况下，地址匹配时 PSR.SLSEL 将被置位。该位由（重复）开始条件自动清零。

### 21.5.2.9 接收器处理

选中的从控接收器始终对接收到的每个数据字节作出应答。若接收缓存 RBUF0/1 已满且不能接收更多数据，则寄存器的值被覆盖（此时 PSR.DLI 置位，可产生协议中断）。

在检查器件是否被选中之前，地址接收也使用寄存器 RBUF0/1 来保存地址。接收到的地址不置位 RDV0/1，因此对地址的处理和对接收数据的处理有所不同。

### 21.5.2.10 接收器状态信息

除接收数据字节之外，IIC 协议相关信息也保存在接收缓存的 16 位数据字中。接收到的数据字节位于 RBUF[7:0]，附加信息可通过 RBUF[12:8]监控。该结构无需读取其它寄存器即可识别每个接收数据字节的含义（使用 FIFO 数据缓存也同样）。

- **RBUF[8]:**

接收的应答位的值。该信息还作为协议参数保存在 RBUFSR[8]中。

- **RBUF[9]:**

若该位的值为 1，则指示在（重新）开始条件之后接收到地址，随后接收到新一帧的第一个数据字节；若该位的值为 0，则指示接收到后续的数据字节。该信息还保存在 RBUFSR[9]中，从而可通过不同的中断服务程序处理地址和数据。

- **RBUF[10]:**

若该位的值为 1，则指示工作在从控模式下的器件已接收到数据字节；若该位的值为 0，则指示主控模式下接收到数据字节。

- **RBUF[11]:**

若该位的值为 1，则指示接收缓存中的数据字节不完整/出错，这是由于开始或终止条件在一帧中的位置错误。该位和 PSR 中的帧错误状态位不同，因为 PSR 中的状态位由软件清零（“粘着”位），而 RBUF[11]由每个数据字节顺序评估。

若该位的值为 0，则指示接收到的数据字节正确，和帧错误无关。

- **RBUF[12]:**

若该位的值为 0，则指示已接收到设定的地址；若该位的值为 1，则指示一个广播呼叫地址。

### 21.5.3 符号时序

IIC 的符号时序由产生移位时钟线 SCL 的主控器件决定。标准模式和快速模式下的符号时序不同。

- 100 kBaud 标准模式 (PCR.H.STIM = 0) :  
一个符号由 10 个时间单元  $t_q$  组成。要求最低模块频率  $f_{SYS} = 2 \text{ MHz}$ 。
- 400 kBaud 标准模式 (PCR.H.STIM = 1) :  
一个符号由 25 个时间单元  $t_q$  组成。要求最低模块频率  $f_{SYS} = 10 \text{ MHz}$ 。

只有当发送器和接收器处于空闲状态或 CCR.MODE = 0 时才能修改波特率的设置。通过寄存器 BRGL 定义时间单元  $t_q$  的长度 (由一个  $f_{PCTQ}$  周期给出) :

- BRGL.CTQSEL  
定义用于产生时间单元的输入频率  $f_{CTQIN}$
- BRGL.PCTQ  
定义时间单元的长度 (对  $f_{CTQIN}$  进行 1、2、3、4 分频)
- BRGL.DCTQ  
定义每个符号中时间单元的个数 ( $t_q$  的个数 = DCTQ + 1)

标准设置由 CTQSEL = 00<sub>B</sub> ( $f_{CTQIN} = f_{PDIV}$ ) 和 PPPEN = 0 ( $f_{PPP} = f_{PIN}$ ) 给出。在这些条件下, 频率  $f_{PCTQ}$  的计算公式如下:

$$f_{PCTQ} = f_{PIN} \times \frac{1}{PDIV + 1} \times \frac{1}{PCTQ + 1} \quad (21.10)$$

考虑到信号 SCL 下降沿之后 SDA 需要 300ns 的保持时间, 引入保持延迟  $t_{HDEL}$ 。它还可避免错误检测开始或终止条件。可通过位域 PCR.HDEL 设定该延迟的长度。要将输入采样和输出更新的时间考虑在内, 根据以下约束条件编程 HDEL:

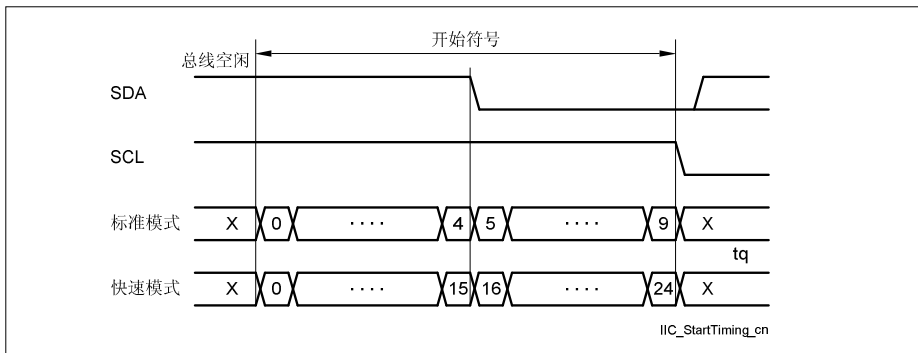
$$HDEL \geq 300ns \times f_{PPP} - (3 \times \frac{f_{PPP}}{f_{SYS}}) + 1 \quad \text{带有数字滤波且 } HDEL_{\min} = 2 \quad (21.11)$$

$$HDEL \geq 300ns \times f_{PPP} - (3 \times \frac{f_{PPP}}{f_{SYS}}) + 2 \quad \text{带有数字滤波且 } HDEL_{\min} = 1$$

若使用数字输入滤波器, 输入信号上出现毛刺时 HDEL 会补偿 2 个滤波器周期的滤波延迟 (应使用  $f_{PPP}$ )。假若 SDA 线上的某位数据恰好在 SCL 的上升沿之前或下降沿之后改变, HDEL 补偿可确保该数据位不会被当作开始或终止条件来处理。

### 21.5.3.1 开始符号

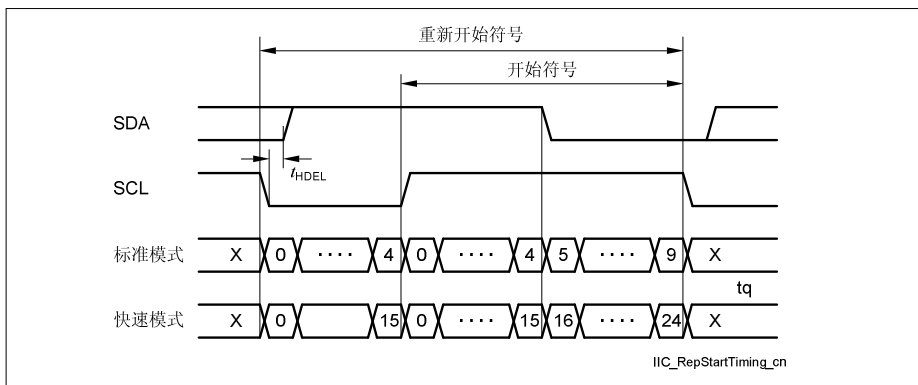
开始符号的时序如图 21-42 所示。



**图 21-42 开始符号时序**

### 21.5.3.2 重新开始符号

在重新开始符号的前半段，SCL 输出低电平（长度为指定的时间单元数），随后输出高电平。检测到 SCL 输入的上升沿之后，产生正常的开始符号，如图 21-43 所示。



**图 21-43 重新开始符号时序**



对于应答传送，USIC IIC 需等待接收器指示已接收到一个完整的字节，这将增加 3 个  $f_{\text{SYS}}$  时钟周期的延迟。必须恰当选择模块的最低输入频率以确保到 SCL 上升沿的 SDA 建立时间。

## 21.5.4 数据流处理

数据流的处理和 IIC 数据帧内的符号序列由 IIC 的发送器部分控制。IIC 是面向字节的总线协议，不过 USIC 数据缓存字最多可包含 16 位数据。除将要发送的数据字节之外（位于 TBUF[7:0]），用于控制 IIC 序列的位域 TDF（发送数据格式）位于 TBUF[10:8]。TDF 编码定义如何发送每个数据字节（IIC 主控或从控）、控制（重复）开始和终止符号的发送。通过该结构，只需写访问 TBUFx 或使用 FIFO 数据缓存即可为 IIC 主控器件定义完整的数据帧（因为不必访问其它控制寄存器）。

若出现错误的或不期望的 TDF 编码（比如在发送缓存的建立过程中由软件出错造成），主控器件将发送终止条件，这将中止当前正在传送的数据帧。从控模块等待有效的 TDF 编码并设置 SCL = 0。软件必须使不期望的 TDF 编码无效并写入有效的 TDF。

需要注意：在多主控总线系统的仲裁阶段，不期望的终止条件可能导致不可预测的总线行为。

### 21.5.4.1 发送数据格式

主控模式下有以下发送数据格式可用：

- 作为主控器件发送数据字（TDF = 000<sub>B</sub>）：  
 主控器件向从控器件发送数据字时使用该格式。发送器发送其数据字节（TBUF[7:0]），接收并检查从控器件返回的应答位。
- 接收数据字节并发送应答 0（TDF = 010<sub>B</sub>）：  
 主控器件读取从控器件的数据字节时使用该格式。主控器件使用 0 电平对传送作出应答、继续传送。
- 接收数据字节并发送应答 1（TDF = 011<sub>B</sub>）：  
 主控器件读取从控器件的数据字节时使用该格式。主控器件使用 1 电平对传送不作应答、终止传送。TBUF[7:0]的内容被忽略。
- 发送开始条件（TDF = 100<sub>B</sub>）：  
 若总线空闲时 TBUF 包含该入口，则将产生开始条件。TBUF[7:0]的内容被看作是发送的第一个地址字节（位 TBUF[7:1]为地址，LSB 为读/写控制）。
- 发送重新开始条件（TDF = 101<sub>B</sub>）：  
 若 TBUF 包含该入口、SCL = 0 且当前无字节传送，若器件是当前的主控器件，它将送出重新开始条件。当前的主控器件是指已设置开始条件（并赢得主控仲裁权）的器件。TBUF[7:0]的内容被当作发送的第一个地址字节（位 TBUF[7:1]为地址，LSB 为读/写控制）。

- 发送终止条件 (TDF = 110<sub>B</sub>) :  
若当前的主控器件已完成最后一个字节的传送 (包括应答), 它将发送一个终止条件 (若该格式存放在 TBUF 中)。TBUF[7:0]的内容被忽略。
- TDF = 111<sub>B</sub>:  
该格式保留不可用。释放 TBUF 入口并置位 PSR 中的错误位 (这可导致协议中断), 此外不执行任何操作。

从控模式下有以下发送数据格式可用 (一帧内的符号由主控器件控制、从控器件只有被主控器件“请求”时才须发送数据):

- 作为从控器件发送数据字 (TDF = 001<sub>B</sub>) :  
从控器件向主控器件发送数据字节时使用该格式。发送器发送其数据字节 (TBUF[7:0]) 并发送应答位 1。

#### 21.5.4.2 有效的主控发送数据格式

根据 IIC 帧格式的定义, 只有一些特定的 TDF 编码序列可用并有效。若在数据帧传送过程中检测到错误的 TDF 编码, 传送被中止、标志位 PCR.WTDF 置位。此外还可产生中断 (若被使能)。当 TDF 编码有误时, 若 USIC IIC 的主控器件仍占用 SDA 线, 则通过终止条件立即中止数据帧传送。但假若从控器件占用 SDA 线 (读传送), 主控器件则必须执行一次无应答的空的读操作, 从而使得在发送终止条件之前从控器件释放 SDA 线。空的读操作所接收的数据字节将保存在 RBUF0/1 中, 但 RDV0/1 不被置位。因此, 空的读操作不会产生接收中断, 数据字节不会保存到接收 FIFO 中。

若当前数据帧中的传送方向已改变 (主控读访问), 则不可能产生发送数据请求 (TDF = 000<sub>B</sub>)、不会被接收 (导致一个错误的 TDF 编码指示)。

**表 21-9      有效的 TDF 编码**

帧位置	有效的 TDF 编码
第一个 TDF 编码 (主控空闲)	开始 (100 <sub>B</sub> )
读传送: 第二个 TDF 编码 (在开始或重新开始之后)	应答接收 (010 <sub>B</sub> ) 或无应答接收 (011 <sub>B</sub> )
写传送: 第二个 TDF 编码 (在开始或重新开始之后)	发送 (000 <sub>B</sub> )、重新开始 (101 <sub>B</sub> )、或终止 (110 <sub>B</sub> )
读传送: 应答之后的第三个以及后续的 TDF 编码	应答接收 (010 <sub>B</sub> ) 或无应答接收 (011 <sub>B</sub> )
读传送: 无应答之后的第三个以及后续的 TDF 编码	重新开始 (101 <sub>B</sub> ) 或终止 (110 <sub>B</sub> )

帧位置	有效的 TDF 编码
写传送：第三个以及后续的 TDF 编码	发送（000 <sub>B</sub> ）、重新开始（101 <sub>B</sub> ）、或终止（110 <sub>B</sub> ）

- **第一个 TDF 编码：**  
通过 TDF 开始编码（100<sub>B</sub>）开始主控传送。所有其它编码被忽略，但不会指示 WTDF 错误。
- **对于读操作，在开始（100<sub>B</sub>）或重新开始（101<sub>B</sub>）之后的 TDF 编码：**  
若开始执行主控读传送（地址字节的 LSB = 1），SDA 的传送方向改变，从控器件将有效驱动数据线。在这种情况下，只有编码 010<sub>B</sub> 和 011<sub>B</sub> 有效。若编码有误需要中止当前的传送，在产生终止条件之前，主控器件必须执行一次空的读操作。
- **对于写操作，在开始（100<sub>B</sub>）或重新开始（101<sub>B</sub>）之后的 TDF 编码：**  
若开始执行主控写传送（地址字节的 LSB = 0），主控器件将占用 SDA 线。在这种情况下，发送（000<sub>B</sub>）、重新开始（101<sub>B</sub>）和终止（110<sub>B</sub>）编码有效。其它编码被视为错误。若编码有误需要中止当前的传送，立即产生终止条件。
- **对于应答前一数据字节的读操作，第三个和后续的的 TDF 编码：**  
若开始执行主控读传送（由地址字节的 LSB 决定），SDA 的传送方向改变，从控器件将有效驱动数据线。为了强制从控器件释放 SDA 线，主控器件必须不应答字节传送。在这种情况下，只有编码 010<sub>B</sub> 和 011<sub>B</sub> 有效。若编码有误需要中止当前的传送，在产生终止条件之前，主控器件必须执行一次空的读操作。
- **对于不应答前一数据字节的读操作，第三个和后续的的 TDF 编码：**  
若开始执行主控读传送（由地址字节的 LSB 决定），SDA 的传送方向改变，从控器件将有效驱动数据线。为了强制从控器件释放 SDA 线，主控器件必须不应答字节传送。在这种情况下，只有重新开始编码（101<sub>B</sub>）和终止编码（110<sub>B</sub>）有效。若编码有误需要中止当前的传送，立即产生终止条件。
- **对于写操作，第三个和后续的的 TDF 编码：**  
若开始执行主控写传送（由地址字节的 LSB 决定），主控器件将占用 SDA 线。在这种情况下，发送（000<sub>B</sub>）、重新开始（101<sub>B</sub>）和终止（110<sub>B</sub>）编码有效。其它编码视为错误。若编码有误需要中止当前的传送，立即产生终止条件。
- **主控器件接收从控器件的无应答之后，自动送出终止条件（除非随后的 TDF 编码请求重新开始）。在这种情况下，所有其它 TDF 编码被忽略。**

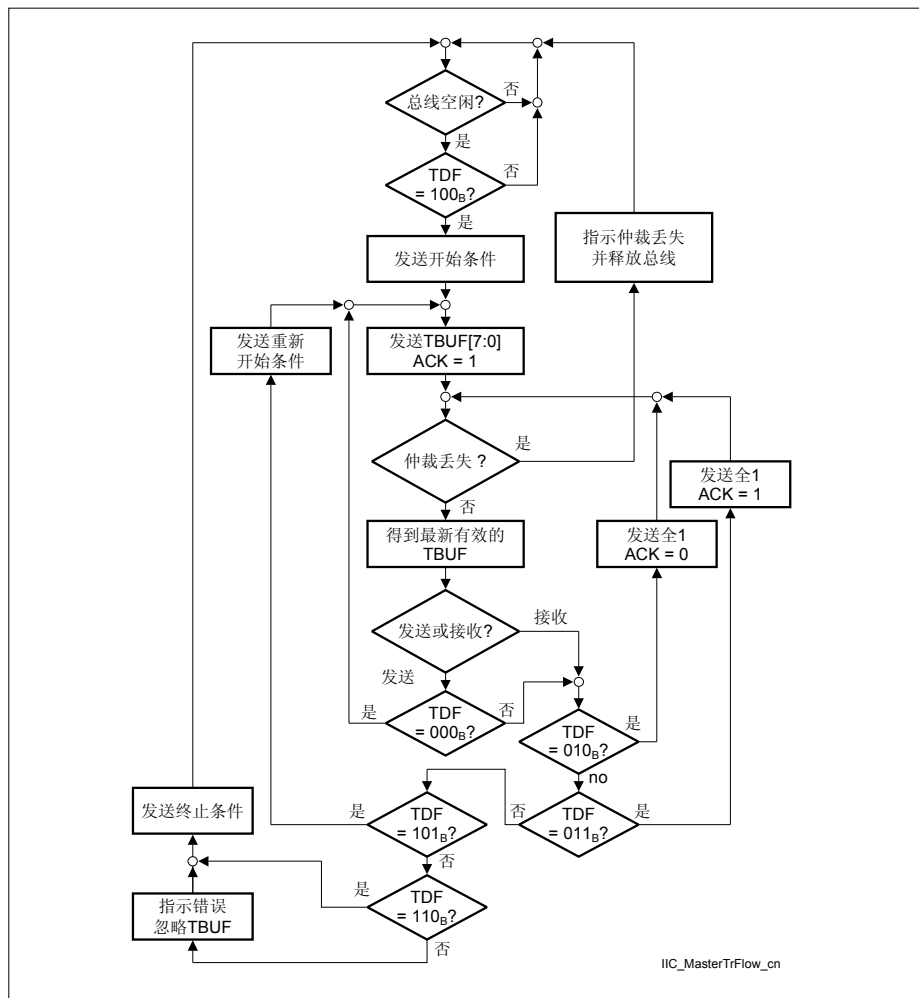


图 21-46 IIC 主控发送



## 21.5.5 IIC 协议寄存器

IIC 模式下，寄存器 PCRH、PCRL 和 PSR 用于处理 IIC 相关信息。

### 21.5.5.1 IIC 协议控制寄存器

IIC 模式下的寄存器 PCRL/PCRH 定义如下：

#### PCRL

协议控制寄存器 L [IIC 模式] (40<sub>H</sub>) 复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
SLAD	[15:0]	rw	<b>从控地址</b> 该位域包含编程的从控地址。将接收到的第一个地址字节中的相应位和 SLAD[15:9]进行比较、检查地址是否匹配。若 SLAD[15:11] = 11110 <sub>B</sub> ，则还要将第二个地址字节和 SLAD[7:0]进行比较。

#### PCRH

协议控制寄存器 H [IIC 模式] (42<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
M CLK	0	HDEL				SAC KDIS	ERR IEN	SRR IEN	ARL IEN	NAC KIEN	PCR IEN	RSC RIEN	SCR IEN	STIM	ACK 00
rw	rw	rw				rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	类型	功能描述
ACK00	0	rw	<b>应答 00<sub>H</sub></b> 该位定义从控器件是否对从控地址 00 <sub>H</sub> 敏感。 0 <sub>B</sub> 从控器件对该地址不敏感。 1 <sub>B</sub> 从控器件对该地址敏感。
STIM	1	rw	<b>符号时序</b>

符号	位序号	类型	功能描述
			<p>该位定义一个符号使用多少个时间单元。</p> <p><b>0<sub>B</sub></b> 一个符号包含 10 个时间单元。该时序适用于标准模式（100 kBaud）。</p> <p><b>1<sub>B</sub></b> 一个符号包含 25 个时间单元。该时序适用于快速模式（400 kBaud）。</p>
<b>SCRIEN</b>	2	rw	<p><b>接收到开始条件中断使能</b></p> <p>若检测到开始条件，该位使能产生协议中断。</p> <p><b>0<sub>B</sub></b> 禁止产生开始条件中断。</p> <p><b>1<sub>B</sub></b> 允许产生开始条件中断。</p>
<b>RSCRIEN</b>	3	rw	<p><b>接收到重新开始条件中断使能</b></p> <p>若检测到重新开始条件，该位使能产生协议中断。</p> <p><b>0<sub>B</sub></b> 禁止产生重新开始条件中断。</p> <p><b>1<sub>B</sub></b> 允许产生重新开始条件中断。</p>
<b>PCRIEN</b>	4	rw	<p><b>接收到终止条件中断使能</b></p> <p>若检测到终止条件，该位使能产生协议中断。</p> <p><b>0<sub>B</sub></b> 禁止产生终止条件中断。</p> <p><b>1<sub>B</sub></b> 允许产生终止条件中断。</p>
<b>NACKIEN</b>	5	rw	<p><b>无应答中断使能</b></p> <p>若主控制器件检测到无应答，该位使能产生协议中断。</p> <p><b>0<sub>B</sub></b> 禁止产生无应答中断。</p> <p><b>1<sub>B</sub></b> 允许产生无应答中断。</p>
<b>ARLIEN</b>	6	rw	<p><b>仲裁丢失中断使能</b></p> <p>若检测到仲裁丢失事件，该位使能产生协议中断。</p> <p><b>0<sub>B</sub></b> 禁止产生仲裁丢失中断。</p> <p><b>1<sub>B</sub></b> 允许产生仲裁丢失中断。</p>
<b>SRRIEN</b>	7	rw	<p><b>从控读取请求中断使能</b></p> <p>若检测到从控读取请求，该位使能产生协议中断。</p> <p><b>0<sub>B</sub></b> 禁止产生从控读取请求中断。</p> <p><b>1<sub>B</sub></b> 允许产生从控读取请求中断。</p>

符号	位序号	类型	功能描述
<b>ERRIEN</b>	8	rw	<b>错误中断使能</b> 若检测到 IIC 错误（由 PSR.ERR 或 PSR.WTDF 指示），该位使能产生协议中断。 0 <sub>B</sub> 禁止产生错误中断。 1 <sub>B</sub> 允许产生错误中断。
<b>SACKDIS</b>	9	rw	<b>从控应答禁止</b> 该位禁止为从控器件产生一个有效的应答信号（有效应答 = 0 电平）。一旦被软件置位，每次出现（重复）开始条件时，该位被自动清零。若在接收到一个字节之后（由中断指示）、开始传送下一个应答位之前该位被置位，则以被动电平发送应答位。这将指示接收器不接收更多的字节。因此，若第一个接收中断用于置位该位，将接收 2 个字节。 0 <sub>B</sub> 允许产生一个有效的从控应答（以 0 电平进行从控应答 = 可接收更多字节）。 1 <sub>B</sub> 禁止产生一个有效的从控应答（以 1 电平进行从控应答 = 停止接收）。
<b>HDEL</b>	[13:10]	rw	<b>硬件延迟</b> 为了将 IIC 协议所规定的 SDA 保持时间考虑在内，该位域定义了硬件延迟，该延迟可对 SCL 信号的内部处理进行补偿（见页 21-143）。
<b>0</b>	14	rw	<b>保留</b> 读操作返回 0；应写入 0。
<b>MCLK</b>	15	rw	<b>主控时钟使能</b> 该位使能产生主控时钟 MCLK（不直接用于 IIC 协议，可用作一般的频率输出）。 0 <sub>B</sub> 禁止产生 MCLK，MCLK 为 0。 1 <sub>B</sub> 使能产生 MCLK。

### 21.5.5.2 IIC 协议状态寄存器

IIC 模式下的寄存器 PSR 定义如下。需要注意：PSR 不被硬件清零。

向寄存器 PSR 写 1 时，PSR 中相应的标志位被清零；向 PSR 写 1 将置位相应的标志位，但不会引发其它操作（不产生中断）。写 0 不起作用。在使能新协议之前，PSR 标志位应由软件清零。

#### PSR

协议状态寄存器 [IIC 模式]

(44<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
AIF	RIF	TBIF	TSIF	DLIF	RSIF	0	ERR	SRR	ARL	N ACK	PCR	R SCR	SCR	W TDF	SL SEL
rwh	rwh	rwh	rwh	rwh	rwh	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位序号	类型	功能描述
SLSEL	0	rwh	<b>从控选择</b> 该位指示该器件已被选作从控器件。 0 <sub>B</sub> 器件未被选作从控器件。 1 <sub>B</sub> 器件被选作从控器件。
WTDF	1	rwh	<b>发现错误的 TDF 编码<sup>1)</sup></b> 该位指示已发现不期望的/错误的 TDF 编码。若 PCRH.ERRIEN = 1，可产生协议中断。 0 <sub>B</sub> 未发现错误的 TDF 编码。 1 <sub>B</sub> 已发现错误的 TDF 编码。
SCR	2	rwh	<b>接收到开始条件<sup>1)</sup></b> 该位指示已在 IIC 总线上检测到开始条件。若 PCRH.SCRIEN = 1，可产生协议中断。 0 <sub>B</sub> 未检测到开始条件。 1 <sub>B</sub> 已检测到开始条件。
RSCR	3	rwh	<b>接收到重新开始条件<sup>1)</sup></b> 该位指示已在 IIC 总线上检测到重新开始条件。若 PCRH.RSCRIEN = 1，可产生协议中断。 0 <sub>B</sub> 未检测到重新开始条件。 1 <sub>B</sub> 已检测到重新开始条件。

符号	位序号	类型	功能描述
<b>PCR</b>	4	rwh	<b>接收到终止条件<sup>1)</sup></b> 该位指示已在 IIC 总线上检测到终止条件。若 PCRH.PCRIEN = 1，可产生协议中断。 0 <sub>B</sub> 未检测到终止条件。 1 <sub>B</sub> 已检测到终止条件。
<b>NACK</b>	5	rwh	<b>接收到无应答<sup>1)</sup></b> 该位指示主控模式下已接收到无应答。从控模式下该位不被置位。若 PCRH.NACKIEN = 1，可产生协议中断。 0 <sub>B</sub> 未接收到无应答。 1 <sub>B</sub> 接收到无应答。
<b>ARL</b>	6	rwh	<b>仲裁丢失<sup>1)</sup></b> 该位指示仲裁已丢失。若 PCRH.ARLIEN = 1，可产生协议中断。 0 <sub>B</sub> 仲裁未丢失。 1 <sub>B</sub> 仲裁已丢失。
<b>SRR</b>	7	rwh	<b>从控读取请求<sup>1)</sup></b> 该位指示已检测到从控读取请求。请求发送缓存中的第一个数据字节可用时该位变为有效。对于后续连续的数据字节，发送缓存送出更多中断。要结束传送时，主控发送器发送终止条件。若 PCRH.SRRIEN = 1，可产生协议中断。 0 <sub>B</sub> 未检测到从控读取请求。 1 <sub>B</sub> 已检测到从控读取请求。
<b>ERR</b>	8	rwh	<b>错误<sup>1)</sup></b> 该位指示已检测到一个 IIC 错误（帧格式或 TDF 编码）。若 PCRH.ERRIEN = 1，可产生协议中断。 0 <sub>B</sub> 未检测到 IIC 错误。 1 <sub>B</sub> 已检测到 IIC 错误。
<b>0</b>	9	r	<b>保留</b> 读操作返回 0；在 IIC 模式下不修改。

符号	位序号	类型	功能描述
<b>RSIF</b>	10	rwh	<b>接收开始指示标志</b> 0 <sub>B</sub> 未发生接收开始事件。 1 <sub>B</sub> 已发生接收开始事件。
<b>DLIF</b>	11	rwh	<b>数据丢失指示标志</b> 0 <sub>B</sub> 未发生数据丢失事件。 1 <sub>B</sub> 已发生数据丢失事件。
<b>TSIF</b>	12	rwh	<b>发送移位指示标志</b> 0 <sub>B</sub> 未发生发送移位事件。 1 <sub>B</sub> 已发生发送移位事件。
<b>TBIF</b>	13	rwh	<b>发送缓存指示标志</b> 0 <sub>B</sub> 未发生发送缓存事件。 1 <sub>B</sub> 已发生发送缓存事件。
<b>RIF</b>	14	rwh	<b>接收指示标志</b> 0 <sub>B</sub> 未发生接收事件。 1 <sub>B</sub> 已发生接收事件。
<b>AIF</b>	15	rwh	<b>备选接收指示标志</b> 0 <sub>B</sub> 未发生备选接收事件。 1 <sub>B</sub> 已发生备选接收事件。

1) IIC 模式下该状态位可产生协议中断（见[页 21-21](#)）。中断状态标志的一般描述请参见中断章节。

## 21.6 IIS 协议

本章说明 USIC 模块如何处理 IIS 协议。该串行协议可用于处理工作在主控模式和从控模式下的两器件之间的同步数据帧的接收和发送。基于 USIC 通信通道的 IIS 连接支持半双工和全双工数据传送。若  $CCFG.IIS = 1$ （IIS 模式可用），由  $CCR.MODE = 0011_B$  选择 IIS 模式。

本章包含以下内容：

- 简介（见[页 21-156](#)）
- 操作 IIS 协议（见[页 21-159](#)）
- 主控模式操作（见[页 21-163](#)）
- 从控模式操作（见[页 21-169](#)）

- 协议寄存器（见页 21-167）

## 21.6.1 简介

IIS 协议是同步串行通信协议，主要用于音频和娱乐信息等应用领域。关于 IIS 的详细信息，请参阅飞利浦 1986 年发布、1996 年 6 月修订的 IIS 规范。

### 21.6.1.1 信号描述

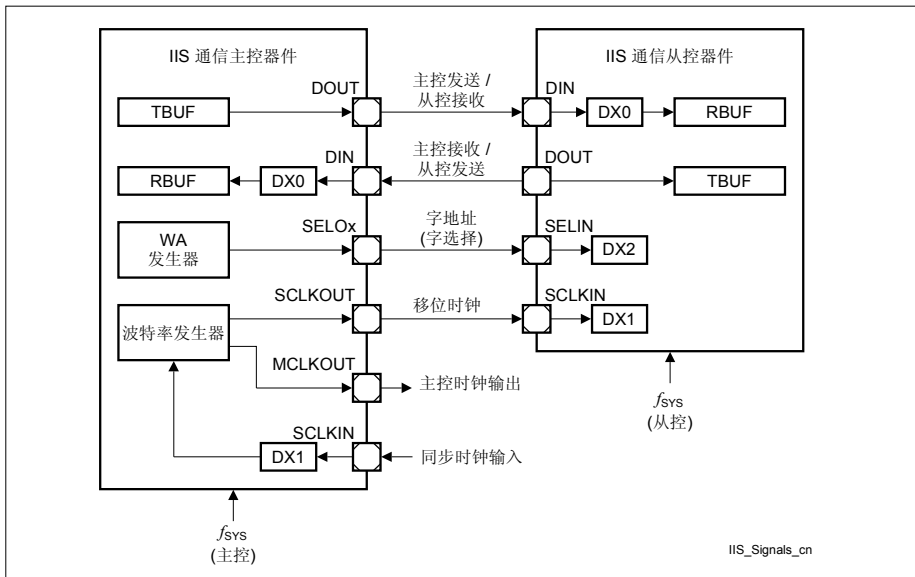
IIS 主控和从控器件之间的连接基于以下这些信号：

- 移位时钟信号 **SCK**，由传送主控器件产生。一旦建立 IIS 连接，则恒生成 **SCK**（即使无有效数据传送）。
- 字地址信号 **WA**（也称作 **WS**），由传送主控器件产生。它指示新数据字的开始和目标音频通道（如左通道/右通道）。若使能 **WA** 产生（对于传送主控器件，**PCR.WAGEN = 1**），字地址输出信号 **WA** 在所有 **SELOx** 输出上可用。**WA** 信号随移位时钟的下降沿同步改变。
- 若 IIS 主控器件用作发送器，它产生主控发送从控接收数据信号。数据随移位时钟的下降沿同步改变。
- 若 IIS 从控器件用作发送器，它产生主控接收从控发送数据信号。数据随移位时钟的下降沿同步改变。

USIC 通信通道的发送器和接收器部分可同时使用，从而建立 IIS 主控和从控器件之间的全双工数据连接。

表 21-10 IIS 的 IO 信号

IIS 模式	接收数据	发送数据	移位时钟	字地址
主控	输入 DIN， 由 DX0 处理	输出 DOUT	输出 SCLKOUT	输出 SELOx
从控	输入 DIN， 由 DX0 处理	输出 DOUT	输入 SCLKIN， 由 DX1 处理	输入 SELIN， 由 DX2 处理



**图 21-47 IIS 信号**

USIC IIS 通信主控器件还有两个附加信号可用：

- 主控时钟输出信号 **MCLKOUT**，和移位时钟有固定的相位关系，支持音频元件过采样。它还可用作与同步 IIS 连接的通信网络的主控时钟输出。
- 同步时钟输入 **SCLKIN**，用于使移位时钟和外部频率同步，从而支持无法直接从通信主控器件系统频率  $f_{sys}$  产生的音频。它还可用作与同步 IIS 连接的通信网络的主控时钟输入。

### 21.6.1.2 协议概览

IIS 连接支持在同一条数据线上传送两种不同的数据帧（比如一种数据帧用于左音频通道、一种数据帧用于右音频通道）。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 USCI 通信通道中，将要发送的数据字被标记为发送至左通道或右通道。同样，接收到的数据字也包含一个标记，用于指示 **WA** 的状态。



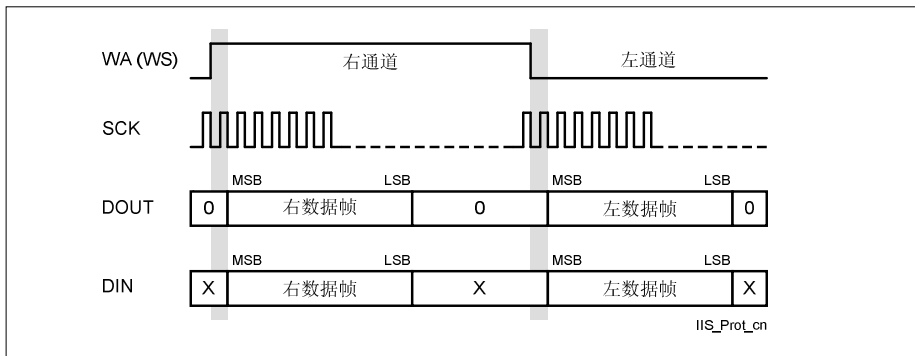


图 21-48 协议概览

### 21.6.1.3 传送延迟

传送延迟特性可使数据传送（发送和接收）延迟进行，延迟时间可编程（以移位时钟周期计数）。

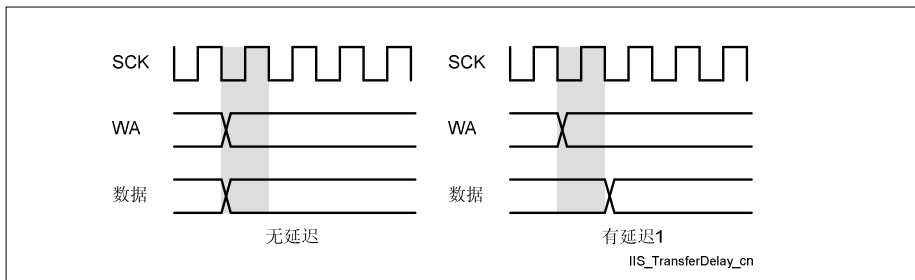
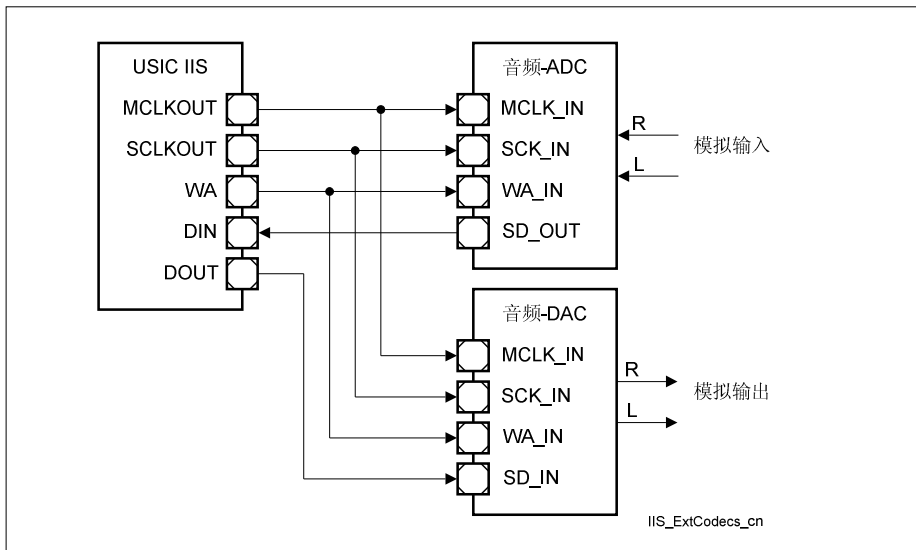


图 21-49 IIS 传送延迟

### 21.6.1.4 外部音频元件的连接

IIS 信号可用于和外部音频器件（如 Codec）或其它音频数据源通信。



**图 21-50 外部音频器件的连接**

在一些应用中（尤其是音频-ADC 或音频-DAC），需要一个和移位时钟信号有固定相位关系的主控时钟信号。MCLKOUT 的频率是 SCLKOUT 移位时钟频率的整数倍。该参数决定了外部器件的过采样率（常用值：256 或 384）。

## 21.6.2 操作 IIS

本节给出操作 IIS 的一般性描述，不特定针对主控或从控模式。

### 21.6.2.1 帧长和字长配置

每次 WA 信号一改变，则将传送一个完整的数据帧（帧长  $\leq$  系统字长），将要传送的数据位的个数由 SCTR.H.FLE 定义。一帧数据可包含多个数据字，数据字长由 SCTR.H.WLE 定义。WA 两次跳变之间的系统字长用 SCLK 的周期数来定义（右通道可用的数据位个数；左通道可用的数据位个数相同）。

若系统字长大于由 SCTR.H.FLE 定义的帧长，额外的数据位以主动电平（SCTRL.PDL）发送；若系统字长小于器件帧长，发送数据中的部分 LSB 不能被传送。

建议将寄存器 TCSRL 中的 WLEMD、FLEMD 和 SELMD 设置为 0。

### 21.6.2.2 自动映射机制

在传送每帧数据时，通过自动映射机制使得波特率和移位控制设置保持不变。可在任意时刻向相应的寄存器设置新值，下次传送数据帧时新设置生效。在数据帧传送期间，所应用（映射）的设置不改变，尽管在开始传送之后已写入新值，该设置被内部“冻结”。

尽管实现了自动映射机制，建议用户只在 IISx 协议被关闭的情况下才改变波特率和移位控制设置。

### 21.6.2.3 模式控制

在 IIS 模式下，支持以下内核模式：

- 运行模式 0/1：  
执行所设定的操作，对数据传送无影响。
- 终止模式 0/1：  
位 PCRL.WAGEN 内部视为 0（位本身不改变）。若 WAGEN = 1，结束当前的系统字，然后停止产生 WA，但 PSR.END 不置位。在进入终止模式之前，结束整个数据帧，包括由 PCRH.TDEL 定义的传送延迟。  
当通过 WAGEN = 1 退出终止模式时，重新开始产生 WA 信号。

### 21.6.2.4 传送延迟

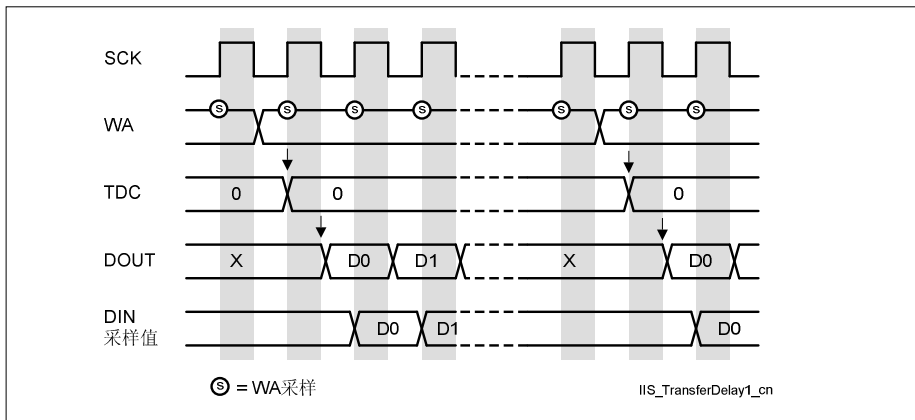
传送延迟可用于使数据传送和事件（如 WA 信号改变）同步。由于事件的输入信号直接被接收器采样（结果是，发送器可在下一个跳变沿使用检测信息），因此，该事件的产生必须和移位时钟 SCK 的下降沿同步（类似于发送数据的改变）。

切勿使用和移位时钟不同步的事件信号。在图 21-49 的示例中，发送主控制器产生事件（信号 WA 改变），该事件和移位时钟 SCK 同步。在 SCK 的上升沿，信号 WA 被采样并检查是否发生改变。若检测到 WA 信号改变，传送延迟计数器 TDC 被自动加载可编程重载值（PCRH.TDEL），否则 TDC 对 SCK 的每个上升沿递减计数，计数到 0 后停止。若 TDC 变为 0，启动数据传送。在以下两种情况下可启动数据传送：

- 检测到事件时 TDC 由 PCRH.TDEL = 0 重载
- 递减计数直至 TDC 为 0

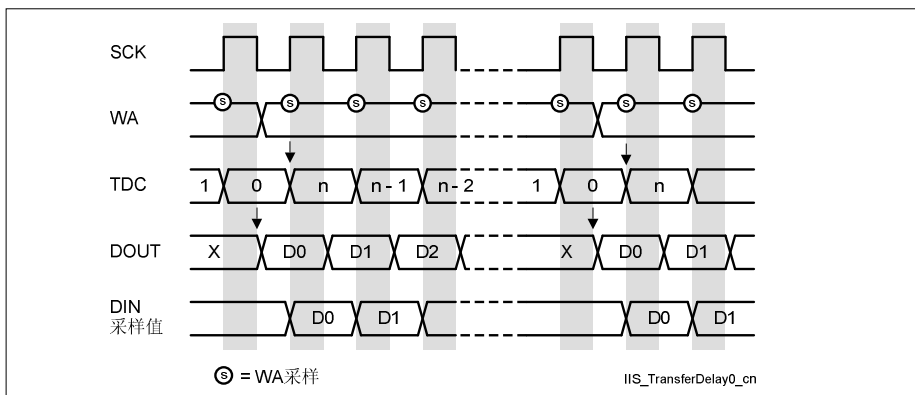
传送延迟计数器位于 IIS 协议预处理器内，软件无法监控。传送延迟（以 SCK 周期数表示）由 PCRH.TDEL + 1 给出。

在图 21-51 的示例中，TDC 的重载值 PCRH.TDEL 为 0。若接收器采样 WA 检测到信号发生改变，计数器 TDC 被重载。若重载值为 0，在 WA 信号发生改变一个移位时钟周期之后开始数据传送。



**图 21-51 具有延迟 1 的传送延迟**

无传送延迟的理想情况如图 21-52 所示。WA 信号改变和数据输出变为有效同时发生。这意味着发送器提前“知道”事件信号将随 **TCLK** 的下一个上升沿而改变。此“预知”的情况可通过检测到 **WA** 改变后插入传送延迟来实现。此延迟应等于系统字长减 1。



**图 21-52 具有延迟 0 的传送延迟**

若检测到传送延迟结束和 **WA** 跳变同时发生，则开始执行传送、并用 **PCR.H.TDEL** 加载延迟计数器。这样可实现无延迟传送。在这种情况下，传送延迟从上次事件开始、到发生新事件时结束。若 **PCR.H.TDEL** 的值大于系统字长，则无传送发生。

### 21.6.2.5 奇偶校验模式

IIS 模式不支持奇偶校验，必须设置 **CCR.PM = 00<sub>B</sub>**。

### 21.6.2.6 传送模式

在 IIS 模式下，必须设置 **SCTRL.TRM = 11<sub>B</sub>** 以允许数据传送。设置 **SCTR.TRM = 00<sub>B</sub>** 将立即禁止并终止数据传送。

### 21.6.2.7 数据传送中断处理

数据传送中断指示和 IIS 帧处理相关的事件：

- 发送缓存中断 **TBI**:  
开始发送数据字的第一位数据之后置位 **PSR.TBIF**。
- 发送移位中断 **TSI**:  
开始发送数据字的最后一位数据之后置位 **PSR.TSIF**。
- 接收开始中断 **RSI**:  
在接收到数据字的第一位数据之后置位 **PSR.RSIF**。  
发生该事件之后，位 **TCSRL.TDV** 被清零，新数据可载入发送缓存。
- 接收中断 **RI** 和备选中断 **AI**:  
**WA = 0** 时，在接收到数据字的最后一位数据之后置位 **PSR.RIF**。  
位 **RBUFSR.SOF** 指示接收到的数据字是否是新一帧的第一个数据字。  
**WA = 1** 时，在接收到数据字的最后一位数据之后置位 **PSR.AIF**。  
位 **RBUFSR.SOF** 指示接收到的数据字是否是新一帧的第一个数据字。

### 21.6.2.8 协议相关参数和错误

为了区分用于左通道或右通道的数据字，IIS 协议预处理器（在 **WA** 跳变之后）采样 **WA** 输入电平并将它作为协议相关错误（尽管它不是错误标志、而是指示标志）存放在接收缓存状态寄存器的位 **RBUFSR[9]** 中。接收到新数据字时，位 **RBUFSR[9]** 决定激活标准接收中断（若 **RBUFSR[9] = 0**）或备选接收中断（若 **RBUFSR[9] = 1**）。接收到的左通道或右通道数据可由不同的中断或 **DMA** 机制处理（若相应事件送至不同的中断节点）。标志位 **PAR** 始终为 **0**。

### 21.6.2.9 发送数据处理

IIS 协议预处理器可区分数据发送至左通道还是右通道。由位 TCSRL.WA 指示缓存中的数据将发送至哪路通道。若 TCSRL.WA = 0，在 WA 发生下降沿跳变之后将发送数据；若 TCSRL.WA = 1，在 WA 发生上升沿跳变之后将发送数据。WA 跳变之后被采样，该采样值用来区分两路通道（参考 PSR.WA）。

若 TCSRL.WAMD = 1，则可由发送控制信息 TCI[4]自动更新位 TCSRL.WA。在这种情况下，写入 TBUF[15:0]的数据（若使用 FIFO 数据缓存，则写入 IN[15:0]）看作是左通道数据；写入 TBUF[31:16]的数据（若使用 FIFO 数据缓存，则写入 IN[31:16]）看作是右通道数据。

### 21.6.2.10 接收缓存处理

若接收缓存 FIFO 可用（CCFG.RB = 1）并被使能（RBDTRH.SIZE > 0），在 IIS 模式下建议设置 RBCTR.H.RCIM = 11<sub>B</sub>。若 OUTRH.RCI[0] = 1，表明该数据字是新一帧数据中的第一个数据字。OUTRH.RCI[4]给出 WA 的采样值，表明是哪路通道。

在 RCI 模式下，可产生标准接收缓存事件和备选接收缓存事件（RBCTR.H.RNM = 1）：

- 标准接收缓存事件指示可从 OUTRL 读取 WA = 0 时启动的数据帧中的数据字
- 备选接收缓存事件指示可从 OUTRL 读取 WA = 1 时启动的数据帧中的数据字

### 21.6.2.11 闭环延迟补偿

由于 IIS 协议的同步信令机制和 SSC 协议相似，因此，在实际应用中必须将闭环延迟考虑在内。IIS 模式下，可通过主控模式下的闭环延迟补偿获得更高的波特率。

详细描述请参阅 SSC 一节。

## 21.6.3 主控模式下操作 IIS

在主控模式下操作 IIS 时，必须考虑以下因素：

- 选择 IIS 模式：

建议在 CCR.MODE = 0000<sub>B</sub>时配置 IIS 的所有参数（在运行期间这些参数不改变）。必须设置 SCTRL.TRM = 11<sub>B</sub>。必须在 CCR.MODE = 0000<sub>B</sub>时配置输入级，从而避免输入信号上出现的多余跳变，随后可通过 CCR.MODE = 0011<sub>B</sub>使能 IIS 模式。

- 数据传送的引脚连接：

通过 DX0CR.INSW = 1 建立输入级 DX0 和接收数据输入引脚（DIN）之间的连接；配置发送数据输出引脚（信号 DOUT）。

支持全双工数据传输的数据移位单元基于同一个 WA 信号，经 DX0 处理的数值被看作是数据位（发送器的接收功能不能被单独禁止）。若接收 IIS 数据，不必配置发送器（无需将 DOUT 分配给引脚）。

- 波特率产生:

必须选择期望的波特率设置，包括分数分频器和波特率发生器设置。必须设置 DX1CR.INSW = 0，将波特率发生器的输出 SCLK 直接用作数据移位单元的输入。用无附加延迟的、SCLKOUT 的反相电平配置移位时钟输出引脚（BRGH.SCLKCFG = 01<sub>B</sub>）。

- 字地址 WA 产生:

必须设置 PCRL.WAGEN = 1，并编程 WA 两次跳变之间的移位时钟周期数以产生 WA。必须设置 DX2CR.INSW = 0，将 WA 发生器的输出用作数据移位单元的输入。如果需要，配置 WA 输出引脚（信号 SELOx）。

- 数据格式配置:

根据应用需要，通过寄存器 SCTRL 和 SCTRH 设置字长、帧长和移位方向。通常，移位 MSB 在先（SCTRL.SDIR = 1）。

可置位 TCSRL.WAMD 从而利用发送控制信息 TCI[4]来区分是在 WA = 0 还是 WA = 1 时发送数据字。

### 21.6.3.1 波特率产生

IIS 模式下的波特率由 SCLK 信号的频率决定（1 个 f<sub>SCLK</sub> 时钟周期代表一位数据）。

若使用分数分频模式产生 f<sub>PIN</sub>，会有一个 f<sub>SYS</sub> 时钟周期的误差，该误差不会在多个 SCLK 周期上累积。因此，可产生所需的平均频率，不过，占空比为 50% 的 SCLK 和 MCLK 信号会出现一个 f<sub>SYS</sub> 时钟周期的误差。

在 IIS 应用中，可选的 MCLK 输出信号和 SCLK 之间相位无关，SCLK 可直接从 f<sub>PIN</sub> 产生（BRGL.PPPEN = 0）。在特殊情况下，MCLK 和 SCLK 之间需要有固定的相位关系（比如使用 MCLK 作为外部器件的时钟参考），必须使用附加的 2 分频电路（BRGL.PPPEN = 1）。由于信号 MCLK 随每个 f<sub>PIN</sub> 时钟周期翻转，因此执行这样的分频操作。此时信号 SCLK 基于 MCLK 产生，如图 21-53 所示。

整数分频因子可编程，通过位域 BRGH.PDIV 设置。

$$f_{SCLK} = \frac{f_{PIN}}{2} \times \frac{1}{PDIV + 1} \quad \text{若 } PPPEN = 0$$

$$f_{SCLK} = \frac{f_{PIN}}{2 \times 2} \times \frac{1}{PDIV + 1} \quad \text{若 } PPPEN = 1 \quad (21.12)$$

注：在 IIS 协议中，主控器件（产生移位时钟和 WA 信号的单元）在 SCK 的下降沿修改数据和 WA 输出的状态。从控发送器也必须在下降沿发送数据。在 SCLK

的上升沿对接收数据采样。必须编程输入级 *DX1* 和 *SCLKOUT* 使移位时钟信号反相从而和内部信号匹配。

### 21.6.3.2 WA 产生

每 *N* 个 *SCLK* 信号周期之后字地址/字选择信号 *WA* 规律的翻转一次。*WA* 两次跳变之间的时间称为系统字长，可通过以下位域设定。

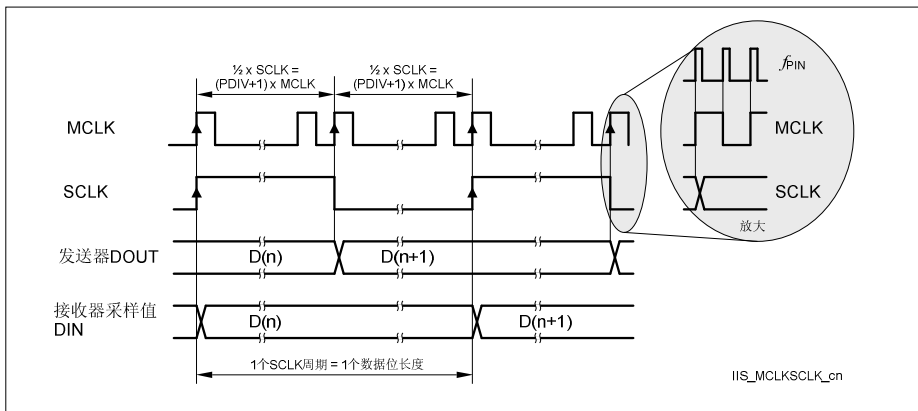
在 *IIS* 主控模式下，系统字长的定义如下：

- **BRGL.CTQSEL** = 10<sub>B</sub>  
基于随 *SCLK* 翻转的 *WA* 信号
- **BRGL.PCTQ**  
定义每个系统字长的 *SCLK* 周期数 *N*
- **BRGL.DCTQ**  
定义每个系统字长的 *SCLK* 周期数 *N*

$$N = (PCTQ + 1) \times (DCTQ + 1) \quad (21.13)$$

### 21.6.3.3 主控时钟输出

主控时钟信号 *MCLK* 可由 *IIS* 的主控器件产生 (**BRGL.PPPEN** = 1)。它专门用于连接外部 *Codec* 器件。可通过 **BRGH.MCLKCFG** 配置信号极性从而产生输出信号 *MCLKOUT*。



**图 21-53 IIS 的 MCLK 和 SCLK**



#### 21.6.3.4 协议中断事件

在 IIS 模式下，可产生以下协议相关事件并导致协议中断。

需要注意：为了监控新的事件，寄存器 PSR 不能由硬件自动清零，必须由软件清零。

- **WA 上升沿/下降沿事件：**  
 WA 生成模块指示这两种事件（由寄存器 PSR 监控）。WA 的下降沿事件置位 PSR.WAFE；WA 的上升沿事件置位 PSR.WARE。若 PCRL.WAFEIEN = 1（针对 WA 下降沿跳变）或 PCRL.WAREIEN = 1（针对 WA 上升沿跳变），可产生协议中断。
- **WA 结束事件：**  
 设置 PCRL.WAGEN = 0 禁止产生 WA 之后，WA 生成模块指示它停止产生 WA 信号。若 PCRL.ENDIEN = 1，可产生协议中断。
- **DX2T 事件：**  
 由 PSR.DX2TEV = 1 指示触发信号 DX2T 有效。若 PCRL.DX2TIEN = 1，可产生协议中断。若使用类似于 SSC 模式中的延迟补偿（详细描述见 SSC 一节），可评估该事件来取代 WA 的上升沿/下降沿事件。

#### 21.6.4 从控模式下操作 IIS

在从控模式下操作 IIS 时，必须考虑以下因素：

- **选择 IIS 模式：**  
 建议在 CCR.MODE = 0000<sub>B</sub> 时配置 IIS 的所有参数（在运行期间这些参数不改变）。必须设置 SCTRL.TRM = 11<sub>B</sub>。必须在 CCR.MODE = 0000<sub>B</sub> 时配置输入级，从而避免输入信号上出现的多余跳变，随后可通过 CCR.MODE = 0011<sub>B</sub> 使能 IIS 模式。
- **数据传送的引脚连接：**  
 通过 DX0CR.INSW = 1 建立输入级 DX0 和接收数据输入引脚（DIN）之间的连接；配置发送数据输出引脚（信号 DOUT）。  
 支持全双工数据传送的数据移位单元基于同一个 WA 信号，经 DX0 处理的数值被看作是数据位（发送器的接收功能不能被单独禁止）。若接收 IIS 数据，不必配置发送器（无需将 DOUT 分配给引脚）。
- **移位时钟的引脚连接：**  
 通过 DX1CR.INSW = 1 且极性翻转（DX1CR.DPOL = 1）建立输入级 DX1 和移位时钟输入引脚（SCLKIN）之间的连接。
- **WA 输入的引脚连接：**  
 通过 DX2CR.INSW = 1 建立输入级 DX2 和 WA 输入引脚（SELIN）之间的连接。

- 波特率产生：  
不需波特率发生器，可由分数分频器关闭。
- WA 产生：  
不需 WA 发生器、可关闭（PCRL.WAGEN = 0）。

#### 21.6.4.1 协议事件和中断

在 IIS 模式下，可产生以下协议相关事件并导致协议中断。

需要注意：为了监控新的事件，寄存器 PSR 不能由硬件自动清零，必须由软件清零。

- WA 上升沿/下降沿/结束事件：  
WA 生成模块被关闭，这些事件不可用。
- DX2T 事件：  
由 PSR.DX2TEV = 1 指示触发信号 DX2T 有效。若 PCRL.DX2TIEN = 1，可产生协议中断。

#### 21.6.5 IIS 协议寄存器

IIS 模式下，寄存器 PCRL、PCRH 和 PSR 用于处理 IIS 相关信息。

##### 21.6.5.1 IIS 协议控制寄存器

IIS 模式下的寄存器 PCRL/PCRH 定义如下：

#### PCRL

协议控制寄存器 L [IIS 模式]

(40H)

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DX2 TIEN	0							END IEN	WAR EIEN	WAF EIEN	0	SEL INV	DT EN	WAG EN	
rw								rw	rw	rw	r	rw	rw	rw	

符号	位序号	类型	功能描述
WAGEN	0	rw	<b>WA 生成使能</b> 该位使能/禁止产生字地址控制输出信号 WA。 0 <sub>B</sub> IIS 可用作从控器件。禁止产生字地址信号， 输出信号 WA 为 0。MCLK 信号的产生和

符号	位序号	类型	功能描述
			<p>PCR.H.MCLK 有关。</p> <p>1<sub>B</sub> IIS 可用作主控制器件。允许产生字地址信号。信号 WA 被使能后从 0 开始。允许产生 MCLK 信号，和 PCR.H.MCLK 无关。WAGEN 被清零后，USIC 模块在接下来的 4 个 WA 周期内停止产生 WA 信号。</p>
<b>DTEN</b>	1	rw	<p><b>数据传送使能</b></p> <p>该位使能/禁止传送 IIS 帧用以应答 WA 的跳变。</p> <p>0<sub>B</sub> WA 输入信号的跳变被忽略，不进行传送。</p> <p>1<sub>B</sub> 允许传送。</p>
<b>SELINV</b>	2	rw	<p><b>选择输出反相</b></p> <p>该位定义 SELOx 输出和内部产生的字地址信号 WA 之间的极性关系。</p> <p>0<sub>B</sub> SELO 输出和 WA 信号的极性相同。</p> <p>1<sub>B</sub> SELO 输出和 WA 信号的极性相反。</p>
<b>WAFEIEN</b>	4	rw	<p><b>WA 下降沿中断使能</b></p> <p>产生 WA 下降沿跳变时，该位使能/禁止产生协议中断。</p> <p>0<sub>B</sub> 产生 WA 下降沿跳变时不激活协议中断。</p> <p>1<sub>B</sub> 产生 WA 下降沿跳变时激活协议中断。</p>
<b>WAREIEN</b>	5	rw	<p><b>WA 上升沿中断使能</b></p> <p>产生 WA 上升沿跳变时，该位使能/禁止产生协议中断。</p> <p>0<sub>B</sub> 产生 WA 上升沿跳变时不激活协议中断。</p> <p>1<sub>B</sub> 产生 WA 上升沿跳变时激活协议中断。</p>
<b>ENDIEN</b>	6	rw	<p><b>结束中断使能</b></p> <p>若清除 PCR.WAGEN 后停止产生 WA（停止产生 WA 之前先完成整个系统字长的处理），该位使能/禁止产生协议中断。</p> <p>0<sub>B</sub> 不激活协议中断。</p> <p>1<sub>B</sub> 激活协议中断。</p>

符号	位序号	类型	功能描述
<b>DX2TIEN</b>	15	rw	<b>DX2T 中断使能</b> 若信号 DX2T 有效（由 PSR.DX2TEV = 1），该位使能/禁止产生协议中断。 0 <sub>B</sub> 若 DX2T 有效，不激活协议中断。 1 <sub>B</sub> 若 DX2T 有效，激活协议中断。
<b>0</b>	3, [14:7]	rw	<b>保留</b> 读操作返回 0；应写入 0。

### PCRH

协议控制寄存器 H [IIS 模式]

(42<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>M CLK</b>	<b>0</b>										<b>TDEL</b>				
rw	rw										rw				

符号	位序号	类型	功能描述
<b>TDEL</b>	[5:0]	rw	<b>传送延迟</b> 当检测到事件发生时，该位域用于定义传送延迟。 若 TDEL = 0，附加延迟功能被关闭，将引入一个移位时钟周期的延迟。
<b>0</b>	[14:6]	r	<b>保留</b> 读操作返回 0；应写入 0。
<b>MCLK</b>	15	rw	<b>主控时钟使能</b> 该位使能产生主控时钟 MCLK（不直接用于 IIC 协议，可用作一般的频率输出）。 0 <sub>B</sub> 禁止产生 MCLK，MCLK 为 0。 1 <sub>B</sub> 使能产生 MCLK。

### 21.6.5.2 IIS 协议状态寄存器

IIS 模式下的寄存器 PSR 定义如下。需要注意：PSR 不被硬件清零。

向寄存器 PSCR 写 1 时，PSR 中相应的标志位被清零；向 PSR 写 1 将置位相应的标志位，但不会引发其它操作（不产生中断）。写 0 不起作用。在使能新协议之前，PSR 标志位应由软件清零。

#### PSR

**协议状态寄存器 [IIS 模式]**

**(44<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>AIF</b>	<b>RIF</b>	<b>TBIF</b>	<b>TSIF</b>	<b>DLIF</b>	<b>RSIF</b>	<b>0</b>		<b>END</b>	<b>WA RE</b>	<b>WA FE</b>	<b>DX2 TEV</b>	<b>0</b>	<b>DX2 S</b>	<b>WA</b>	
rwh	rwh	rwh	rwh	rwh	rwh	r		rwh	rwh	rwh	rwh	r	rwh	rwh	

符号	位序号	类型	功能描述
<b>WA</b>	0	rwh	<b>字地址</b> 检测到 WA 发生跳变后，该位指示 WA 输入信号的采样值/状态。该信息保存在位 RBUFSR[9]中，用于区分接收数据用于左通道还是右通道。 0 <sub>B</sub> WA 采样为 0。 1 <sub>B</sub> WA 采样为 1。
<b>DX2S</b>	1	rwh	<b>DX2S 的状态</b> 该位指示 DX2S 信号的当前状态，该信号可用作字地址信号 WA。 0 <sub>B</sub> DX2S 为 0。 1 <sub>B</sub> DX2S 为 1。
<b>DX2TEV</b>	3	rwh	<b>DX2T 事件检测<sup>1)</sup></b> 该位指示 DX2T 信号是否被激活。在 IIS 从控模式下，若 PCRL.DX2TIEN = 1，DX2T 有效时会产生协议中断。 0 <sub>B</sub> DX2T 信号还未被激活。 1 <sub>B</sub> DX2T 信号已被激活。
<b>WAFE</b>	4	rwh	<b>WA 下降沿事件<sup>1)</sup></b> 该位指示是否已产生 WA 输出信号的下降沿。若

符号	位序号	类型	功能描述
			<p>PCRL.WAFEIEN = 1, 该事件产生协议中断。</p> <p>0<sub>B</sub> 还未产生 WA 的下降沿。</p> <p>1<sub>B</sub> 已产生 WA 的下降沿。</p>
<b>WARE</b>	5	rwh	<p><b>WA 上升沿事件<sup>1)</sup></b></p> <p>该位指示是否已产生 WA 输出信号的上升沿。若 PCRL.WAREIEN = 1, 该事件产生协议中断。</p> <p>0<sub>B</sub> 还未产生 WA 的上升沿。</p> <p>1<sub>B</sub> 已产生 WA 的上升沿。</p>
<b>END</b>	6	rwh	<p><b>WA 产生结束<sup>1)</sup></b></p> <p>该位指示在清除 PCRL.WAGEN 之后是否已停止产生 WA。在清除 WAGEN 之前应由软件清除该位。</p> <p>0<sub>B</sub> 还未停止产生 WA (若 WA 生成模块正在工作且 WAGEN 已被清零)。</p> <p>1<sub>B</sub> 已停止产生 WA (若 WA 生成模块已在工作)。</p>
<b>RSIF</b>	10	rwh	<p><b>接收开始指示标志</b></p> <p>0<sub>B</sub> 未发生接收开始事件。</p> <p>1<sub>B</sub> 已发生接收开始事件。</p>
<b>DLIF</b>	11	rwh	<p><b>数据丢失指示标志</b></p> <p>0<sub>B</sub> 未发生数据丢失事件。</p> <p>1<sub>B</sub> 已发生数据丢失事件。</p>
<b>TSIF</b>	12	rwh	<p><b>发送移位指示标志</b></p> <p>0<sub>B</sub> 未发生发送移位事件。</p> <p>1<sub>B</sub> 已发生发送移位事件。</p>
<b>TBIF</b>	13	rwh	<p><b>发送缓存指示标志</b></p> <p>0<sub>B</sub> 未发生发送缓存事件。</p> <p>1<sub>B</sub> 已发生发送缓存事件。</p>
<b>RIF</b>	14	rwh	<p><b>接收指示标志</b></p> <p>0<sub>B</sub> 未发生接收事件。</p> <p>1<sub>B</sub> 已发生接收事件。</p>

符号	位序号	类型	功能描述
<b>AIF</b>	15	rwh	<b>备选接收指示标志</b> 0 <sub>B</sub> 未发生备选接收事件。 1 <sub>B</sub> 已发生备选接收事件。
<b>0</b>	2, [9:7]	r	<b>保留</b> 读操作返回 0；在 IIS 模式下不修改。

1) IIS 模式下该状态位可产生协议中断（见[页 21-21](#)）。中断状态标志的一般描述请参见中断章节。

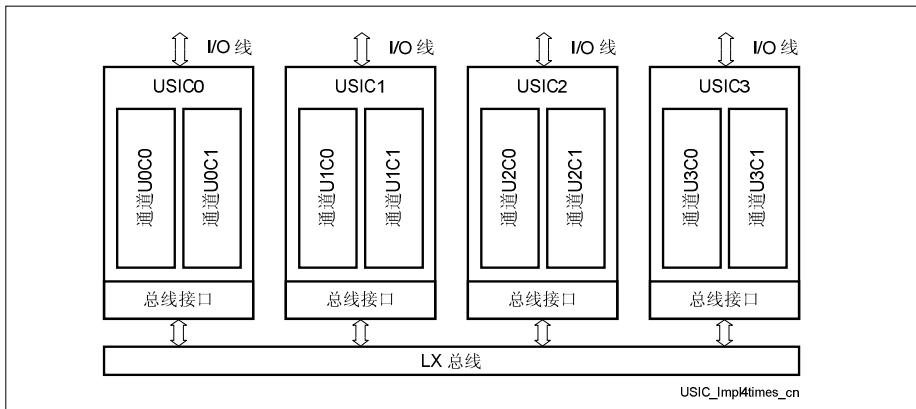
## 21.7 XE16xyM 中 USIC 的实现

本节将描述 XE16xyM 中 USIC 模块的实现。它包含以下内容：

- 模块实现概览（见[页 21-172](#)）
- 通道特性（见[页 21-173](#)）
- 地址映射（见[页 21-174](#)）
- 模块 ID 寄存器（见[页 21-175](#)）
- 中断控制寄存器（见[页 21-176](#)）
- 输入/输出连接（见[页 21-178](#)）
- USIC 模块 0 I/O 线（见[页 21-179](#)）
- USIC 模块 1 I/O 线（见[页 21-182](#)）
- USIC 模块 2 I/O 线（见[页 21-184](#)）
- USIC 模块 3 I/O 线（见[页 21-187](#)）

### 21.7.1 模块实现概览

XE16xyM 器件最多可包含 4 个相同的 USIC 模块（USIC0、USIC1、USIC2 和 USIC3），每个 USIC 模块中有两路通信通道。在各产品的数据手册中详细说明了在该产品中可用的 USIC 模块。



**图 21-54 XE16xyM 中 USIC 模块的结构**

### 21.7.2 通道特性

XE16xyM 中的 USIC 通道支持以下功能：

**表 21-11 USIC 模块特性集**

通道	ASC 协议	LIN 支持	SSC 协议	IIC 协议	IIS 协议	FIFO 缓存入口	SELOx <sup>1)</sup>
U0C0	支持	支持	支持	支持	支持	64 共用	8
U0C1	支持	支持	支持	支持	支持		4
U1C0	支持	支持	支持	支持	支持	64 共用	8
U1C1	支持	支持	支持	支持	支持		5
U2C0	支持	支持	支持	支持	支持	64 共用	6
U2C1	支持	支持	支持	支持	支持		3
U3C0	支持	支持	支持	支持	支持	64 共用	4
U3C1	支持	支持	支持	支持	支持		2

1) 该数字是指 144 引脚封装中最多可用的信号个数。需要注意：这些信号中有些可能相互重叠。因此，并非所有的信号同时可用。



### 21.7.3 地址映射

USIC 通信通道寄存器的基地址归纳见下表。这些寄存器的准确地址为相对地址（由表 21-3 给出）加通道基地址（由表 21-12 给出）。

**表 21-12 寄存器地址空间**

模块	起始地址	结束地址	注
U0C0	204000 <sub>H</sub>	2041FF <sub>H</sub>	标准地址
U0C1	204200 <sub>H</sub>	2043FF <sub>H</sub>	标准地址
U1C0	204800 <sub>H</sub>	2049FF <sub>H</sub>	标准地址
U1C1	204A00 <sub>H</sub>	204BFF <sub>H</sub>	标准地址
U2C0	205000 <sub>H</sub>	2051FF <sub>H</sub>	标准地址
U2C1	205200 <sub>H</sub>	2053FF <sub>H</sub>	标准地址
U3C0	205800 <sub>H</sub>	2059FF <sub>H</sub>	标准地址
U3C1	205A00 <sub>H</sub>	205BFF <sub>H</sub>	标准地址
U0C0A	20B000 <sub>H</sub>	20B1FF <sub>H</sub>	备选地址
U0C1A	20B200 <sub>H</sub>	20B3FF <sub>H</sub>	备选地址
U1C0A	20B400 <sub>H</sub>	20B5FF <sub>H</sub>	备选地址
U1C1A	20B600 <sub>H</sub>	20B7FF <sub>H</sub>	备选地址
U2C0A	20B800 <sub>H</sub>	20B9FF <sub>H</sub>	备选地址
U2C1A	20BA00 <sub>H</sub>	20BBFF <sub>H</sub>	备选地址
U3C0A	20BC00 <sub>H</sub>	20BDFF <sub>H</sub>	备选地址
U3C1A	20BE00 <sub>H</sub>	20BFFF <sub>H</sub>	备选地址

## 21.7.4 模块 ID 寄存器

模块 ID 寄存器指示 USIC 模块的功能和设计阶段。

### USIC0\_IDL

模块 ID 寄存器 L (204008<sub>H</sub>) 复位值: C0XX<sub>H</sub>

### USIC1\_IDL

模块 ID 寄存器 L (204808<sub>H</sub>) 复位值: C0XX<sub>H</sub>

### USIC2\_IDL

模块 ID 寄存器 L (205008<sub>H</sub>) 复位值: C0XX<sub>H</sub>

### USIC3\_IDL

模块 ID 寄存器 L (205808<sub>H</sub>) 复位值: C0XX<sub>H</sub>

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

符号	位序号	类型	功能描述
MOD_REV	[7:0]	r	<b>模块修订版编号</b> MOD_REV 定义模块修订版编号，从 01 <sub>H</sub> 开始编号（第一版）。
MOD_TYPE	[15:8]	r	<b>模块类型</b> 该位域的值为 C0 <sub>H</sub> ，它定义该模块为 32 位模块。

### USIC0\_IDH

模块 ID 寄存器 H (20400A<sub>H</sub>) 复位值: 003A<sub>H</sub>

### USIC1\_IDH

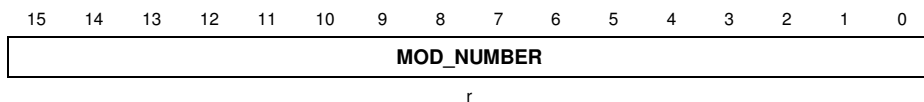
模块 ID 寄存器 H (20480A<sub>H</sub>) 复位值: 003A<sub>H</sub>

### USIC2\_IDH

模块 ID 寄存器 H (20500A<sub>H</sub>) 复位值: 003A<sub>H</sub>

### USIC3\_IDH

模块 ID 寄存器 H (20580A<sub>H</sub>) 复位值: 003A<sub>H</sub>



符号	位序号	类型	功能描述
<b>MOD_NUMBER</b>	[15:0]	r	<b>模块编号值</b> 该位域定义 USIC 模块的 ID 编号（003A <sub>H</sub> = USIC）。

## 21.7.5 中断控制寄存器

每路 USIC 通道提供 4 个服务请求输出 SR[3:0]（这些输出未必需要全部连接到独立的中断寄存器 UxCy\_nIC 上）。服务请求输出到中断控制寄存器的具体分配归纳见**表 21-13**。

每路 USIC 通信通道连接到 3 个专用的中断控制寄存器上（连接到 UxCy\_SR[2:0]，比如一个用于发送，一个用于接收，一个用于协议或错误处理、或用于备选接收事件）。每路通信通道的第四个中断控制寄存器（连接到 UxCy\_SR3）和模块 CC2 共用。

中断控制寄存器位于 SFR 区，在中断一章已对它们进行了详细描述。

**表 21-13 USIC 中断控制寄存器**

服务请求输出线	中断控制寄存器/位
USIC0 通道 0 的 SR0	U0C0_0IC
USIC0 通道 0 的 SR1	U0C0_1IC
USIC0 通道 0 的 SR2	U0C0_2IC

服务请求输出线	中断控制寄存器/位
USIC0 通道 0 的 SR3	SCU_ISSR.4, 和 CC2 共用
USIC0 通道 1 的 SR0	U0C1_0IC
USIC0 通道 1 的 SR1	U0C1_1IC
USIC0 通道 1 的 SR2	U0C1_2IC
USIC0 通道 1 的 SR3	SCU_ISSR.5, 和 CC2 共用
USIC1 通道 0 的 SR0	U1C0_0IC
USIC1 通道 0 的 SR1	U1C0_1IC
USIC1 通道 0 的 SR2	U1C0_2IC
USIC1 通道 0 的 SR3	SCU_ISSR.6, 和 CC2 共用
USIC1 通道 1 的 SR0	U1C1_0IC
USIC1 通道 1 的 SR1	U1C1_1IC
USIC1 通道 1 的 SR2	U1C1_2IC
USIC1 通道 1 的 SR3	SCU_ISSR.7, 和 CC2 共用
USIC2 通道 0 的 SR0	U2C0_0IC
USIC2 通道 0 的 SR1	U2C0_1IC
USIC2 通道 0 的 SR2	U2C0_2IC
USIC2 通道 0 的 SR3	SCU_ISSR.12, 和 CC2 共用
USIC2 通道 1 的 SR0	U2C1_0IC
USIC2 通道 1 的 SR1	U2C1_1IC
USIC2 通道 1 的 SR2	U2C1_2IC
USIC2 通道 1 的 SR3	SCU_ISSR.13, 和 CC2 共用
USIC3 通道 0 的 SR0	U3C0_0IC
USIC3 通道 0 的 SR1	U3C0_1IC
USIC3 通道 0 的 SR2	U3C0_2IC
USIC3 通道 0 的 SR3	SCU_ISSR.2, 和 CC2 共用

服务请求输出线	中断控制寄存器/位
USIC3 通道 1 的 SR0	U3C1_0IC
USIC3 通道 1 的 SR1	U3C1_1IC
USIC3 通道 1 的 SR2	U3C1_2IC
USIC3 通道 1 的 SR3	SCU_ISSR.3, 和 CC2 共用

### 21.7.6 输入/输出连接

USIC 通道的 I/O 连接如图 21-55 所示。以下各表定义了 XE16xyM 器件中 USIC 通道 I/O 线的引脚分配和内部连接。命名规则：UxCy 代表 USIC 模块 x 通道 y。

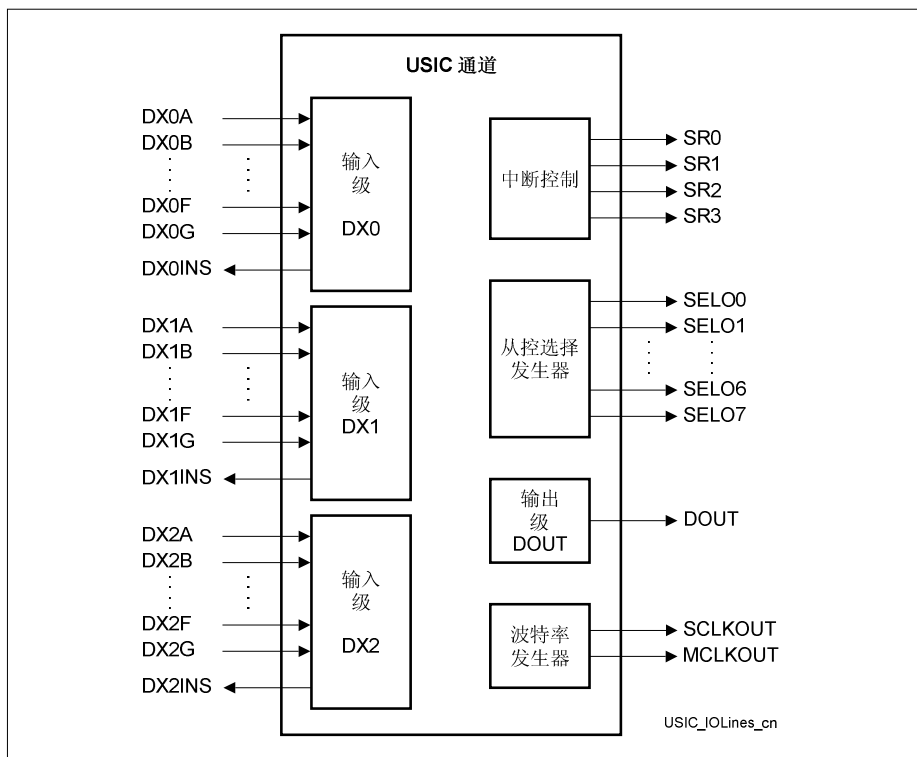


图 21-55 USIC 通道 I/O 线

服务请求输出 SR[3:0]到中断控制寄存器的连接请参见由[页 21-176](#)的[表 21-13](#)。

### 21.7.6.1 USIC 模块 0 的 I/O 线

USIC 模块 0 通道 0 的信号具有前缀 “U0C0\_”，通道 1 的信号具有前缀 “U0C1\_”。

**表 21-14 USIC0 的 I/O 连接**

信号	用作	来自/送至	
		通道 0 – U0C0	通道 1 – U0C1
数据输入			
DX0A	移位数据输入	P10.0	P10.0
DX0B	移位数据输入	P10.1	P10.7
DX0C	移位数据输入	P10.6	P10.14
DX0D	移位数据输入	P7.4	P2.3
DX0E	移位数据输入	P2.3	P2.10
DX0F	移位数据输入	P2.4	P7.3
DX0G	回环数据移位输入	U0C0_DOUT	U0C1_DOUT
时钟输入			
DX1A	移位时钟输入	P10.1	P10.10
DX1B	移位时钟输入	P10.2	P10.5
DX1C	移位时钟输入	P10.8	P10.15
DX1D	移位时钟输入	P2.5	P2.8
DX1E	移位时钟输入	0	P7.4
DX1F	单线 ASC 冲突检测输入	U0C0_DX0INS	U0C1_DX0INS
DX1G	回环移位时钟输入	U0C0_SCLKOUT	U0C1_SCLKOUT
控制输入			
DX2A	移位控制输入	P10.3	P10.3

信号	用作	来自/送至	
		通道 0 – U0C0	通道 1 – U0C1
DX2B	移位控制输入	P10.4	P10.4
DX2C	移位控制输入	P10.10	P2.7
DX2D	移位控制输入	P2.6	0
DX2E	发送数据有效性检验输入	CC2_CC24	RTC_T14INT
DX2F	发送数据有效性检验输入	CCU60_T13_PM	CCU60_T13_PM
DX2G	回环移位控制输入	U0C0_SELO0	U0C1_SELO0

**数据输出**

DOUT	移位数据输出	P2.3	P2.9
		P7.3	P2.10
		P10.1	P7.3
		P10.6	P7.4
		-	P10.0
		-	P10.7
		-	P10.14
		-	P10.15
		-	P4.3
		-	P2.4

**时钟输出**

MCLKOUT	主控时钟输出 (如用于 IIS)	P10.8	P10.9
SCLKOUT	移位时钟输出	P2.5	P2.8
		P10.2	P7.4
		-	P10.5

**控制输出**

SELO0	移位控制输出 0	P2.6	P2.7
-------	----------	------	------

信号	用作	来自/送至	
		通道 0 – U0C0	通道 1 – U0C1
		P10.10	P10.8
		U0C0_DX2G	U0C1_DX2G
SELO1	移位控制输出 1	P2.7	P2.6
SELO2	移位控制输出 2	P2.11	P2.11
SELO3	移位控制输出 3	P2.10	P2.12
		P10.4	-
SELO4	移位控制输出 4	P2.12	-
		P3.4	-
		P10.9	-
SELO5	移位控制输出 5	P3.5	-
SELO6	移位控制输出 6	P3.6	-
SELO7	移位控制输出 7	P3.7	-

**系统相关输出**

DX0INS	ERU 的外部中断输入 (SCU)	ERU_0A2	ERU_0B2
	单线 ASC 冲突检测	U0C0_DX1F	U0C1_DX1F
DX1INS	-	-	-
DX2INS	ERU 的外部中断输入 (SCU)	ERU_0A3	ERU_0B3

**回环输出**

DOUT	回环移位数据输出	U0C0_DX0G	U0C1_DX0G
SCLKOUT	回环移位时钟输出	U0C0_DX1G	U0C1_DX1G
SELO0	回环移位控制输出	U0C0_DX2G	U0C1_DX2G



### 21.7.6.2 USIC 模块 1 的 I/O 线

USIC 模块 1 通道 0 的信号具有前缀 “U1C0\_”，通道 1 的信号具有前缀 “U1C1\_”。

**表 21-15 USIC1 的 I/O 连接**

信号	用作	来自/送至	
		通道 0 – U1C0	通道 1 – U1C1
数据输入			
DX0A	移位数据输入	P0.0	P0.6
DX0B	移位数据输入	P0.1	P0.7
DX0C	移位数据输入	P10.14	ESR1
DX0D	移位数据输入	P2.3	ESR2
DX0E	移位数据输入	ESR0	P6.0
DX0F	移位数据输入	ESR1	CAN1INS
DX0G	回环数据移位输入	U1C0_DOUT	U1C1_DOUT
时钟输入			
DX1A	移位时钟输入	P0.1	P0.5
DX1B	移位时钟输入	P0.2	P0.6
DX1C	移位时钟输入	P0.5	P6.2
DX1D	移位时钟输入	P10.11	0
DX1E	移位时钟输入	P10.12	0
DX1F	单线 ASC 冲突检测输入	U1C0_DX0INS	U1C1_DX0INS
DX1G	回环移位时钟输入	U1C0_SCLKOUT	U1C1_SCLKOUT
控制输入			
DX2A	移位控制输入	P0.3	P0.4
DX2B	移位控制输入	ESR0	ESR1
DX2C	移位控制输入	ESR1	ESR2

信号	用作	来自/送至	
		通道 0 – U1C0	通道 1 – U1C1
DX2D	移位控制输入	P10.6	P6.3
DX2E	发送数据有效性检验输入	CC2_CC25	RTC_T14INT
DX2F	发送数据有效性检验输入	CCU61_T13_PM	CCU61_T13_PM
DX2G	回环移位控制输入	U1C0_SELO0	U1C1_SELO0

**数据输出**

DOUT	移位数据输出	P0.0	P0.6
		P0.1	P0.7
		P10.12	P6.1
		P10.13	P8.2
		P10.15	-

**时钟输出**

MCLKOUT	主控时钟输出 (如用于 IIS)	P1.0	P1.7
SCLKOUT	移位时钟输出	P0.2	P0.5
		P10.11	P6.2

**控制输出**

SELO0	移位控制输出 0	P0.3	P0.4
		P10.6	P6.3
SELO1	移位控制输出 1	P0.4	P0.3
		P10.14	-
SELO2	移位控制输出 2	P0.5	P1.6
		P10.15	-
SELO3	移位控制输出 3	P0.7	P1.5
		P10.13	-
SELO4	移位控制输出 4	P1.0	P1.4

信号	用作	来自/送至	
		通道 0 – U1C0	通道 1 – U1C1
SELO5	移位控制输出 5	P1.1	-
SELO6	移位控制输出 6	P1.2	-
SELO7	移位控制输出 7	P1.3	-

**系统相关输出**

DX0INS	ERU 的外部中断输入 (SCU)	ERU_1A2	ERU_1B2
	单线 ASC 冲突检测	U1C0_DX1F	U1C1_DX1F
DX1INS	ERU 的外部中断输入 (SCU)	ERU_3B0	-
DX2INS	ERU 的外部中断输入 (SCU)	ERU_1A3	ERU_1B3

**回环输出**

DOUT	回环移位数据输出	U1C0_DX0G	U1C1_DX0G
SCLKOUT	回环移位时钟输出	U1C0_DX1G	U1C1_DX1G
SELO0	回环移位控制输出	U1C0_DX2G	U1C1_DX2G

**21.7.6.3 USIC 模块 2 的 I/O 线**

USIC 模块 2 通道 0 的信号具有前缀 “U2C0\_”，通道 1 的信号具有前缀 “U2C1\_”。

**表 21-16 USIC2 的 I/O 连接**

信号	用作	来自/送至	
		通道 0 – U2C0	通道 1 – U2C1
数据输入			
DX0A	移位数据输入	P3.0	P3.6
DX0B	移位数据输入	P3.1	P3.7

信号	用作	来自/送至	
		通道 0 – U2C0	通道 1 – U2C1
DX0C	移位数据输入	P1.5	P1.1
DX0D	移位数据输入	P1.6	P1.2
DX0E	移位数据输入	P9.5	ESR2
DX0F	移位数据输入	P5.8	P5.10
DX0G	回环数据移位输入	U2C0_DOUT	U2C1_DOUT

**时钟输入**

DX1A	移位时钟输入	P3.0	P3.5
DX1B	移位时钟输入	P3.2	P3.6
DX1C	移位时钟输入	P1.7	P1.2
DX1D	移位时钟输入	P9.7	0
DX1E	移位时钟输入	0	0
DX1F	单线 ASC 冲突检测输入	U2C0_DX0INS	U2C1_DX0INS
DX1G	回环移位时钟输入	U2C0_SCLKOUT	U2C1_SCLKOUT

**控制输入**

DX2A	移位控制输入	P3.3	P3.4
DX2B	移位控制输入	P1.4	ESR2
DX2C	移位控制输入	0	ESR1
DX2D	移位控制输入	0	0
DX2E	发送数据有效性检验输入	CC2_CC26	RTC_T14INT
DX2F	发送数据有效性检验输入	CCU62_T13_PM	CCU62_T13_PM
DX2G	回环移位控制输入	U2C0_SELO0	U2C1_SELO0

**数据输出**

DOUT	移位数据输出	P3.0	P3.6
		P3.1	P3.7

信号	用作	来自/送至	
		通道 0 – U2C0	通道 1 – U2C1
		P1.6	P1.1
		P9.4	P10.8
		P9.5	-
		P10.5	-

**时钟输出**

MCLKOUT	主控时钟输出 (如用于 IIS)	-	-
SCLKOUT	移位时钟输出	P3.2	P3.5
		P1.7	P1.2

**控制输出**

SELO0	移位控制输出 0	P3.3	P3.4
SELO1	移位控制输出 1	P3.4	P3.3
SELO2	移位控制输出 2	P3.5	P2.13
SELO3	移位控制输出 3	P3.7	-
SELO4	移位控制输出 4	P1.3	-
SELO5	移位控制输出 5	P1.4	-
SELO6	移位控制输出 6	-	-
SELO7	移位控制输出 7	-	-

**系统相关输出**

DX0INS	ERU 的外部中断输入 (SCU)	ERU_2A2	ERU_2B2
	单线 ASC 冲突检测	U2C0_DX1F	U2C1_DX1F
DX1INS	ERU 的外部中断输入 (SCU)	ERU_2B0	-
DX2INS	ERU 的外部中断输入 (SCU)	ERU_2A3	ERU_2B3

信号	用作	来自/送至	
		通道 0 – U2C0	通道 1 – U2C1
回环输出			
DOUT	回环移位数据输出	U2C0_DX0G	U2C1_DX0G
SCLKOUT	回环移位时钟输出	U2C0_DX1G	U2C1_DX1G
SELO0	回环移位控制输出	U2C0_DX2G	U2C1_DX2G

#### 21.7.6.4 USIC 模块 3 的 I/O 线

USIC 模块 3 通道 0 的信号具有前缀 “U3C0\_”，通道 1 的信号具有前缀 “U3C1\_”。

**表 21-17      USIC3 的 I/O 连接**

信号	用作	来自/送至	
		通道 0 – U3C0	通道 1 – U3C1
数据输入			
DX0A	移位数据输入	P10.3	P2.10
DX0B	移位数据输入	P4.5	P11.4
DX0C	移位数据输入	0	0
DX0D	移位数据输入	0	0
DX0E	移位数据输入	0	0
DX0F	移位数据输入	0	0
DX0G	回环数据移位输入	U3C0_DOUT	U3C1_DOUT
时钟输入			
DX1A	移位时钟输入	P10.14	P11.0
DX1B	移位时钟输入	P4.2	0
DX1C	移位时钟输入	0	0
DX1D	移位时钟输入	0	0

**通用串行接口通道**

信号	用作	来自/送至	
		通道 0 – U3C0	通道 1 – U3C1
DX1E	移位时钟输入	0	0
DX1F	单线 ASC 冲突检测输入	U3C0_DX0INS	U3C1_DX0INS
DX1G	回环移位时钟输入	U3C0_SCLKOUT	U3C1_SCLKOUT

**控制输入**

DX2A	移位控制输入	P10.11	P11.1
DX2B	移位控制输入	P10.2	P11.5
DX2C	移位控制输入	P4.4	0
DX2D	移位控制输入	0	0
DX2E	发送数据有效性检验输入	CC2_CC27	RTC_T14INT
DX2F	发送数据有效性检验输入	CCU63_T13_PM	CCU63_T13_PM
DX2G	回环移位控制输入	U3C0_SELO0	U3C1_SELO0

**数据输出**

DOUT	移位数据输出	P10.4	P2.11
		P4.5	P11.4
		P4.6	P11.2
		-	-
		-	-

**时钟输出**

MCLKOUT	主控时钟输出 (如用于 IIS)	-	-
SCLKOUT	移位时钟输出	P10.14	P11.0
		P4.2	-

**控制输出**

SELO0	移位控制输出 0	P10.11	P11.1
SELO1	移位控制输出 1	P10.2	P11.5

**通用串行接口通道**

信号	用作	来自/送至	
		通道 0 – U3C0	通道 1 – U3C1
SELO2	移位控制输出 2	P4.4	-
SELO3	移位控制输出 3	P4.1	-
SELO4	移位控制输出 4	-	-
SELO5	移位控制输出 5	-	-
SELO6	移位控制输出 6	-	-
SELO7	移位控制输出 7	-	-

**系统相关输出**

DX0INS	ERU 的外部中断输入 (SCU)	-	-
	单线 ASC 冲突检测	U3C0_DX1F	U3C1_DX1F
DX1INS	ERU 的外部中断输入 (SCU)	-	-
DX2INS	ERU 的外部中断输入 (SCU)	-	-

**回环输出**

DOUT	回环移位数据输出	U3C0_DX0G	U3C1_DX0G
SCLKOUT	回环移位时钟输出	U3C0_DX1G	U3C1_DX1G
SELO0	回环移位控制输出	U3C0_DX2G	U3C1_DX2G



## 22 控制器局域网 (MultiCAN) 控制器

本章描述 XE16xyM 的 MultiCAN 控制器，内容包括：

- MultiCAN 内核概述 (见 [章节 22.1](#))
- MultiCAN 内核功能描述 (见 [章节 22.2](#))
- XE16xyM MultiCAN 控制器的具体实现和寄存器 (端口连接和控制、中断控制、地址译码、时钟控制，见 [章节 22.5](#))

注：本章描述的 MultiCAN 寄存器在 XE16xyM 用户手册其它章节引用时，需要添加模块名前缀 “CAN\_”。

### 22.1 MultiCAN 简述

本节描述 XE16xyM 通信模块 MultiCAN 的串行通讯接口 CAN (控制器局域网)。

#### 22.1.1 概述

MultiCAN 模块包含 6 个代表通信接口的独立 CAN 节点。

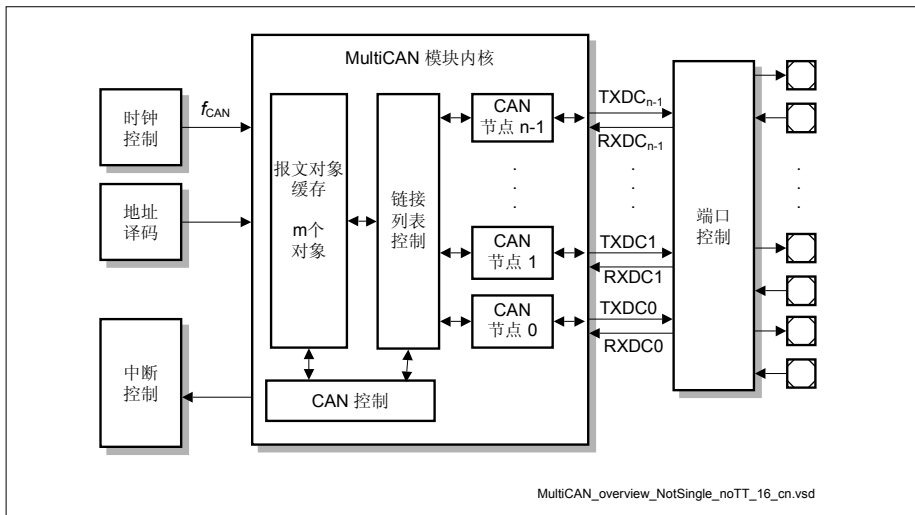


图 22-1 MutliCAN 模块概览

## 22.1.2 CAN 特性

MultiCAN 模块的高性能得益于下述关键特性：

- 根据 CAN V2.0 B active 技术规范确定 CAN 功能（与 ISO 11898 标准兼容）
- 多达 6 个独立的 CAN 节点
- 多达 256 个独立报文对象（所有 CAN 节点共用）
- 每个 CAN 节点都有专用控制寄存器
- 数据传送速率高达 1 Mbit/s，每个 CAN 节点可单独设定数据传送速率
- 具有灵活、功能强大的报文传送控制和错误处理能力
- 全 CAN 功能：报文对象可被单独
  - 分配给 6 个 CAN 节点之一
  - 配置为发送或者接收对象
  - 处理 11 位标识符的标准帧或者 29 位标识符的扩展帧
  - 用于滤波的可编程验收屏蔽寄存器
  - 通过帧计数器进行监控
  - 设置为远程监控模式
- 支持自动网关模式
- 16 个独立的、可编程中断输出
- 用于总线监控的 CAN 分析器模式
- 可选择对 MultiCAN 模块内的 SRAM 进行奇偶校验错误保护

## 22.2 CAN 功能描述

本节描述 CAN 模块的核心特性。

### 22.2.1 命名惯例和定义

**表 22-1** 给出贯穿整个 MultiCAN 规范的一些常数。对于给定的 MultiCAN 具体实现，此处给出最大值。无论如何，在不同的产品中，根据产品配置，部分报文对象或节点可能被禁止。

**表 22-1 固定的模块常数**

常数	值	描述
<b>n_objects</b>	256	报文对象的个数 n_objects 表示可用的报文对象总数。
<b>n_interrupts</b>	16	中断输出线个数 n_interrupts 表示可用的中断输出线总数。
<b>n_pendings</b>	256	报文挂起位个数 n_pendings 表示可用的报文挂起位个数。报文挂起寄存器的个数由 n_pendings/32 给出。
<b>n_lists</b>	8	列表个数 n_lists 表示可用于分配报文的列表总数。
<b>n_nodes</b>	6	可用 CAN 节点个数 n_nodes 代表可用的 CAN 节点总数。由于每个 CAN 节点都有其自身的列表，加上未被分配的元素列表，因此关系表达式： $n\_nodes < n\_lists$ 为真。

### 22.2.2 简介

MultiCAN 模块包含六个全 CAN 功能的节点，这些节点可独立工作或者通过网关功能交换数据和远程帧。根据 CAN V2.0B (active) 规范发送和接收 CAN 帧。每个 CAN 节点都可以接收和发送 11 位标识符的标准帧和 29 位标识符的扩展帧。

### 22.2.2.1 特性概述

所有 CAN 节点共用一套报文对象，每个报文对象可被独立分配给任一 CAN 节点。除了存储接收帧和发送帧外，报文对象可组合起来，在 CAN 节点之间构建网关或者建立 FIFO 缓存器。

可将报文对象组织为双链列表结构，每个 CAN 节点都有自己的报文对象列表。CAN 节点仅将帧储存在分配给该 CAN 节点列表的报文对象中，且仅发送属于该报文对象列表的报文对象中的报文。

功能强大、由命令驱动列表控制器执行所有报文对象列表操作。

CAN 节点位时序都来自外设时钟 (fCAN)，可编程的数据率最高可达 1Mbit/s。由一对接收和发送引脚将每个 CAN 节点和总线收发器连接起来。

#### 特性

- 与 ISO 11898 标准兼容
- 根据 CAN V2.0 B active 技术规范确定 CAN 功能
- 每个 CAN 节点都有专用控制寄存器
- 数据传送速率高达 1 Mbit/s
- 具有灵活、功能强大的报文传送控制和错误处理能力
- 具有先进的 CAN 总线位时序分析和由帧计数器实现的波特率检测功能。
- 全 CAN 功能：一套（共 256 个）报文对象可被单独：
  - 分配（指派）给任一 CAN 节点
  - 配置为发送或者接收对象
  - 处理 11 位标识符的标准帧或者 29 位标识符的扩展帧
  - 由帧计数器计数或分配时间戳
  - 设置为远程监控模式
- 先进的验收滤波功能：
  - 为每个报文对象提供单独的验收屏蔽寄存器，对接收帧进行验收滤波
  - 报文对象可配置为只接收标准帧或者扩展帧，或者两者都接收
  - 可将报文对象分成四级优先级组
  - 根据 CAN 仲裁规则，基于帧标识符、IDE 位和 RTR 位选择最先发送的报文
- 先进的报文对象功能：
  - 报文对象可以组合成任意大小的 FIFO 报文缓存器，该缓存的大小仅受报文对象总数的限制；

**控制器局域网 (MultiCAN)**

- 可以将报文对象连接起来形成网关，在两个不同 CAN 总线之间自动进行帧传送。单网关可以将任意两个 CAN 节点连接起来。可以定义任意数目的网关。
- 先进的数据管理：
  - 报文对象可组织为双链列表；
  - 任意时刻，即使在 CAN 节点全速工作期间，都可以进行列表重组；
  - 功能强大，命令驱动列表控制器管理列表结构并确保列表一致性；
  - 基于列表结构的报文 FIFO，在 CAN 工作期间易于重新划分其大小；
  - 与基于非列表的 TwinCAN 应用兼容的静态分配命令；
- 先进的中断处理
  - 多达 16 条中断输出线，大多数中断请求可单独发至其中之一；
  - 报文后处理通知功能可以灵活组成具有 256 个通知位的专用寄存器。

### 22.2.2.2 模块结构

图 22-2 给出具有 6 个 CAN 节点的 MultiCAN 模块的总结构图。

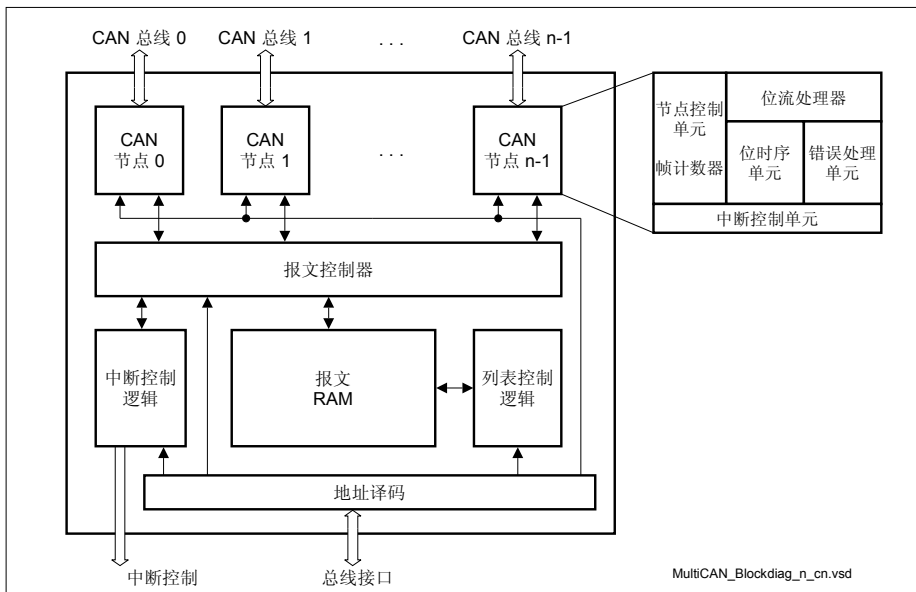


图 22-2 带有多个 CAN 节点的 MutliCAN 模块框图

### CAN 节点

CAN 节点由表 22-2 中描述的多子单元组成：

表 22-2 CAN 节点的子单元

子单元	描述
位流处理器	位流处理器根据 ISO 11898 标准，进行数据帧、远程帧、错误帧和过载帧处理，其中还包括串行数据流和输入/输出寄存器之间的转换。
位时序单元	考虑到传播延迟和相移误差，位时序单元根据用户设置确定位时间长度和采样点位置。位时序单元还执行再同步操作。
错误处理单元	错误处理单元管理接收和发送错误计数器。根据这两个计数器内

**控制器局域网络 (MultiCAN)**

子单元	描述
	容, 使 CAN 节点进入到“错误激活”, “错误认可”或“总线关闭”状态。
<b>节点控制单元</b>	节点控制单元协调 CAN 节点操作: <ul style="list-style-type: none"> <li>• 使能/禁止 CAN 节点传送</li> <li>• 使能/禁止和产生可引发中断请求的特定事件 (如 CAN 总线错误, 帧传送成功等)</li> <li>• 帧计数器的管理</li> </ul>

### 报文控制器

报文控制器处理 CAN 节点之间的 CAN 帧交换, 也处理存储在报文 RAM 中的报文对象。该模块执行下列操作:

- 进行接收验收滤波, 确定用来存储接收 CAN 帧的正确报文对象
- 进行发送验收滤波, 单独为每个 CAN 节点确定要最先发送的报文对象
- 根据报文对象的状态/控制位等, 在报文对象和 CAN 节点之间进行内容传送
- 处理 FIFO 缓存和网关功能
- 集合报文挂起通知位

### 列表控制器

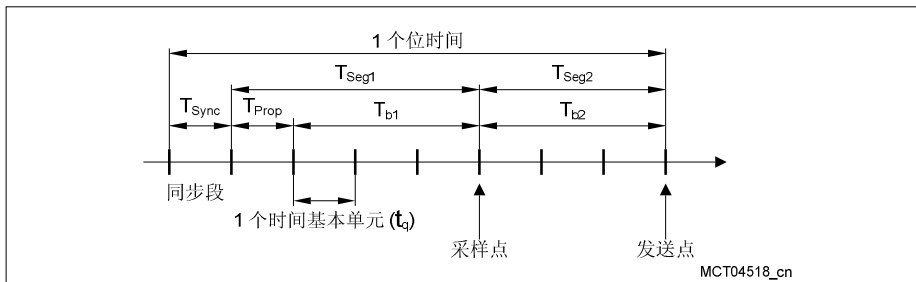
由列表控制器执行双链报文对象列表的所有修改操作。仅允许列表控制器对列表结构进行修改。由一个用户命令接口 (命令面板) 请求报文对象的分配/再分配操作, 然后由列表控制器的状态机自动执行所请求的命令。

## 22.2.3 CAN 节点控制

可单独配置并运行每个 CAN 节点, 而不依赖于其它 CAN 节点。每个 CAN 节点都配有一组 SFR 寄存器, 用于控制和监控 CAN 节点。

### 22.2.3.1 位时序

根据 ISO 11898 标准, CAN 位时间可被划分为不同时间段 (见图 22-3)。每个时间段包含几个基本时间单元  $t_q$ 。由位域 BRP 和位 DIV8 调整  $t_q$  的大小, 这两个参数还控制波特率预分频器 (见位时序寄存器 NBTR)。由 MultiCAN 模块时钟  $f_{CAN}$  驱动波特率预分频器。



**图 22-3 CAN 总线位时序标准**

在同步段 ( $T_{Sync}$ ) 进行发射和接收时间基准之间的相位同步。同步段长度总为  $1t_q$ 。传播段 ( $T_{Prop}$ ) 考虑 CAN 总线上发送输出驱动器和收电路的物理传播延迟。对于工作冲突检测机制来讲,  $T_{Prop}$  必须等于所有传播延迟量的总和舍入到  $t_q$  整数倍之后, 再乘以 2 所得的值。采样点之前和之后的相位缓冲段 1 和 2 ( $T_{b1}$ ,  $T_{b2}$ ) 用于补偿同步段中检测到的发射和接收之间的时钟相位失配。

重新同步阶段允许的时间单元最大值由 CAN 节点位时序寄存器 NBTR 中的位域 SJW 定义。传播段和相位缓冲段 1 结合成参数  $T_{Seg1}$ , 由相应 CAN 节点位时序寄存器 NBTR 中的 TSEG1 定义, ISO 标准要求其最小值为 3 个时间单元。参数  $T_{Seg2}$ , 由 CAN 节点位时序寄存器 NBTR 中的 TSEG2 定义, 包含相位缓冲段 2, ISO 标准要求其最小值为 2 个时间单元。根据 ISO 标准, CAN 位时间, 为  $T_{Sync}$ ,  $T_{Seg1}$  和  $T_{Seg2}$  的总和, 必须不少于 8 个时间单元。

位时间的计算:

$$\begin{aligned} t_q &= (BRP+1)/f_{CAN} && \text{如果 } DIV8 = 0 \\ &= 8 \times (BRP+1)/f_{CAN} && \text{如果 } DIV8 = 1 \\ T_{Sync} &= 1 t_q \\ T_{Seg1} &= (TSEG1+1) \times t_q && (\text{最小: } 3 t_q) \\ T_{Seg2} &= (TSEG2+1) \times t_q && (\text{最小: } 2 t_q) \\ \text{位时间} &= T_{Sync} + T_{Seg1} + T_{Seg2} && (\text{最小: } 8 t_q) \end{aligned}$$

为了补偿不同 CAN 控制器时钟之间的相移, CAN 控制器必须是在从隐性到显性总线电平的任意边沿进行同步。如果硬同步被使能 (在一帧的开始), 在同步段重新开始位时间。否则, 重新同步跳转宽度  $T_{SJW}$  定义最大数目的时间单元, 重新同步操作可能缩短或拉长位时间。SJW 的值编程设置到 CAN 节点位时序寄存器中。

$$\begin{aligned} T_{SJW} &= (SJW + 1) \times t_q \\ T_{Seg1} &\geq T_{SJW} + T_{Prop} \\ T_{Seg2} &\geq T_{SJW} \end{aligned}$$

$f_{CAN}$  的最大相对容差由相位缓冲段和重新同步跳转宽度决定。



$$df_{CAN} \leq \min(T_{b1}, T_{b2})/2 \times (13 \times \text{位时间} - T_{b2}) \text{ 且}$$

$$df_{CAN} \leq T_{SJW}/20 \times \text{位时间}$$

在复位节点控制寄存器中的位 INIT 之前，也就是说，在使能 CAN 节点操作之前，必须将有效 CAN 位时序写入到寄存器 NBTR 中。

仅在相应的节点控制寄存器位 CCE（使能配置更改）置位的情况下，才能对节点位时序寄存器进行写操作。

### 22.2.3.2 CAN 错误处理

CAN 节点的错误处理单元负责对 CAN 器件进行故障界定。该单元有 2 个计数器，接收错误计数器和发送错误计数器（控制寄存器 NECNT 内），这两个计数器根据位流处理器的命令相应的增加和减少。如果在发送操作的同时，位流处理器自身检测到了一个错误，发送错误计数器增加 8。当一个外部 CAN 节点因为错误帧产生而报告一个错误情况，节点计数器增加 1。对于错误分析、出错报文传送方向和识别到传送错误的节点，都由相关的 CAN 节点的控制寄存器 NECNT 指示。根据错误计数器的值，CAN 节点被设置为如下状态：“错误激活”“错误认可”和“总线关闭”。

如果两个错误计数器值都低于错误认可界限 128，CAN 处于错误激活状态。如果至少两者之一等于或者大于 128，CAN 节点处于错误认可状态。

如果发送错误计数器的值等于或者大于“总线关闭”界限 256，“总线关闭”状态被激活，并由 CAN 节点状态寄存器 NSR 中的标志 BOFF 报告该状态。器件一直保持在该状态，直到完成“总线关闭”恢复过程为止。另外，当至少一个错误计数器等于或大于 CAN 节点控制寄存器 NECNT 中的位域 EWRNLVL 中定义的错误警告值时，NSR 状态寄存器内的标志 EWRN 被置位。如果两个错误计数器的值再次降到错误警告值以下，位 EWRN 被复位（见页 22-52）。

### 22.2.3.3 CAN 帧计数器

每个 CAN 节点都配有帧计数器，用来计数 CAN 帧的发送/接收，或者获得关于 CAN 节点何时开始帧发送/接收的时刻信息。CAN 帧计数/位时间计数的操作由 16 位计数器完成，该计数器由相应节点的寄存器 NFCR 控制。由寄存器 NFCR 中的位域 CFSEL 确定帧计数器的工作模式。

- 帧计数模式：成功发送和/或接收一个 CAN 帧之后，帧计数器增加。新计数值被复制到与传送有关的报文对象的中断指针寄存器的位域 CFC 中。
- 时间戳模式：新的位时间开始时，帧计数器增加。当开始发送/接收一帧时，帧计数器值被捕获并存储到寄存器 NFCR 的位域 CFC 中。成功传送该帧之后，捕获值被复制到与传送有关的报文对象的中断指针寄存器的位域 CFC 中。
- 位时序模式：用于波特率检测和位时序分析（见章节 22.2.5.3）

### 22.2.3.4 CAN 节点中断

每个 CAN 节点都有 4 个中断源。

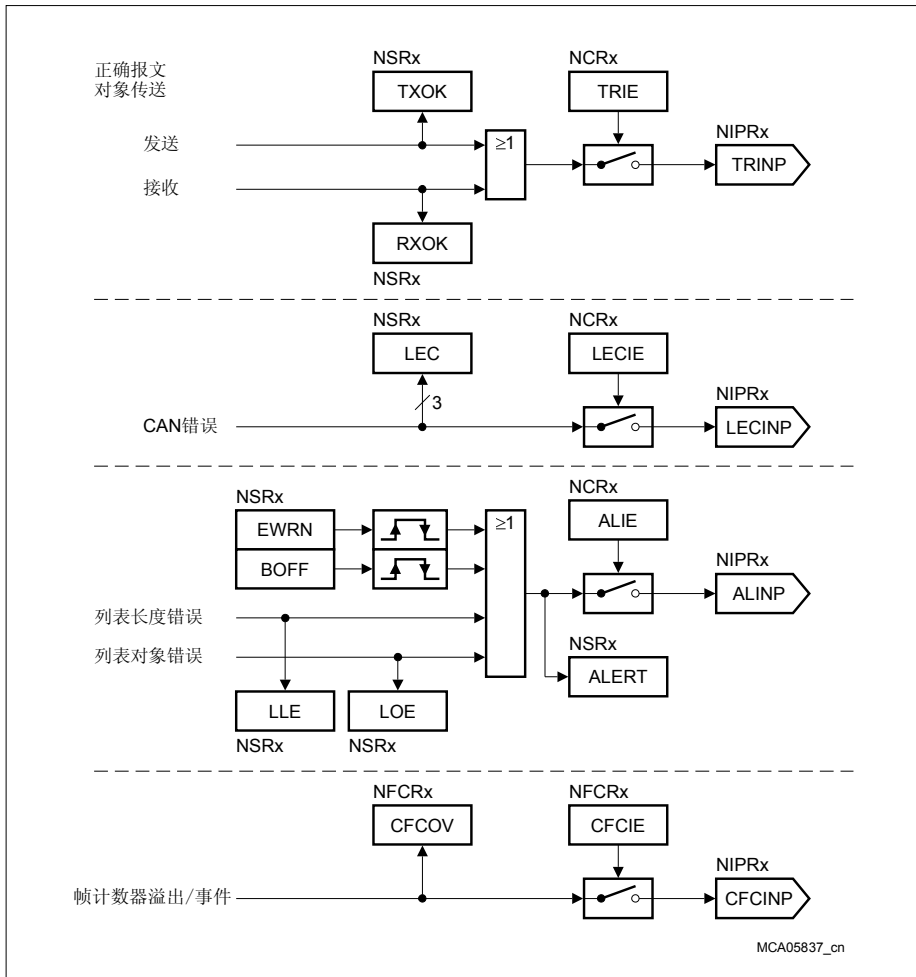


图 22-4 CAN 节点中断

发生下述情况时，CAN 节点可产生中断请求：

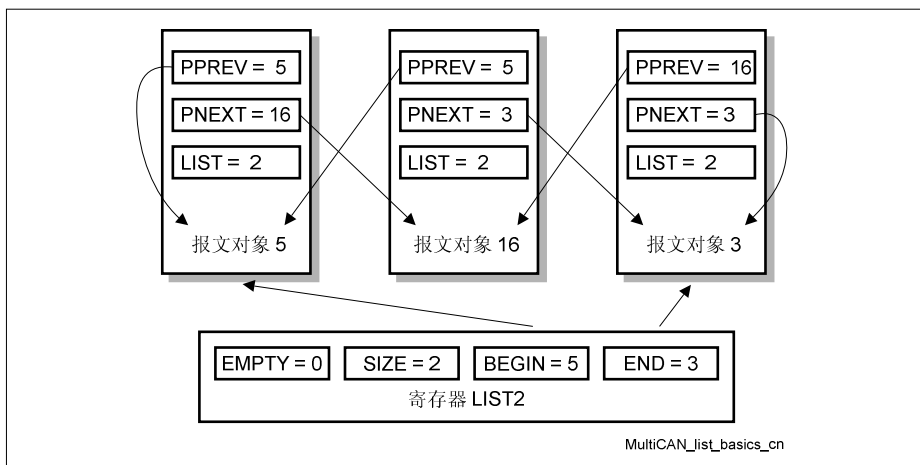
- 成功发送/接收一帧
- 帧计数器溢出（帧计数模式/时间戳模式）或位时序测量事件（位时序模式）
- 与 CAN 节点相关的错误

## 22.2.4 报文对象列表结构

MultiCAN 模块的报文对象被组织为双链列表。在该列表结构中，每个报文对象都有两个指针，一个指向列表中的前一个报文对象，另一个指向列表中的后一个报文对象。

### 22.2.4.1 基本知识

XE16xyM 系列的 MultiCAN 模块提供 8 个不同列表，每个报文对象分配给其中之一。报文对象控制寄存器中的 4 位 LIST 位域指示相关报文对象当前所属列表编号。在图 22-5 的例子中，3 个报文对象分配给了编号为 2 的列表。



**图 22-5 列表中分配报文对象示例**

列表寄存器的位域 **BEGIN** 指向列表的第一个元素（例中的报文对象 5），位域 **END** 指向列表的最后一个元素（例中的报文对象 3）。列表中的元素个数由位域 **SIZE** 给出（#元素 = SIZE + 1，因此例中 SIZE = 2，元素个数为 3）。位 **EMPTY** 指示该列表是否为空（因为列表 2 不为空，所以例中 EMPTY = 0）。

每个报文对象都有一个指针 **PNEXT**（位于报文对象控制寄存器内），用来指向列表中的下一个报文对象，还有一个指针 **PPREV**，用来指向列表中的前一个报文对象。第一个报文对象的 **PPREV** 指向其本身，因为第一个报文对象之前没有报文对象（例中的报文对象 5 为第一个报文对象，由 **PPREV = 5** 表示）。最后一个报文对象的 **PNEXT** 也指向其本身，因为最后一个报文对象之后没有报文对象（例中报文对象 3 为列表的最后一个报文对象，由 **PNEXT = 3** 表示）。

每个报文对象还有一个 4 位 LIST 位域（位于报文对象控制寄存器内）指示报文对象当前所属的列表编号（例中的报文对象都分配给了列表 2，因此 LIST = 2）。

#### 22.2.4.2 未被分配的报文对象列表

列表 0 具有特殊的意义：它是那些未被分配的元素列表。当且仅当一个元素属于列表 0，那就称其未被分配。当且仅当一个元素属于编号不等于 0 的列表，则称该报文对象被分配。

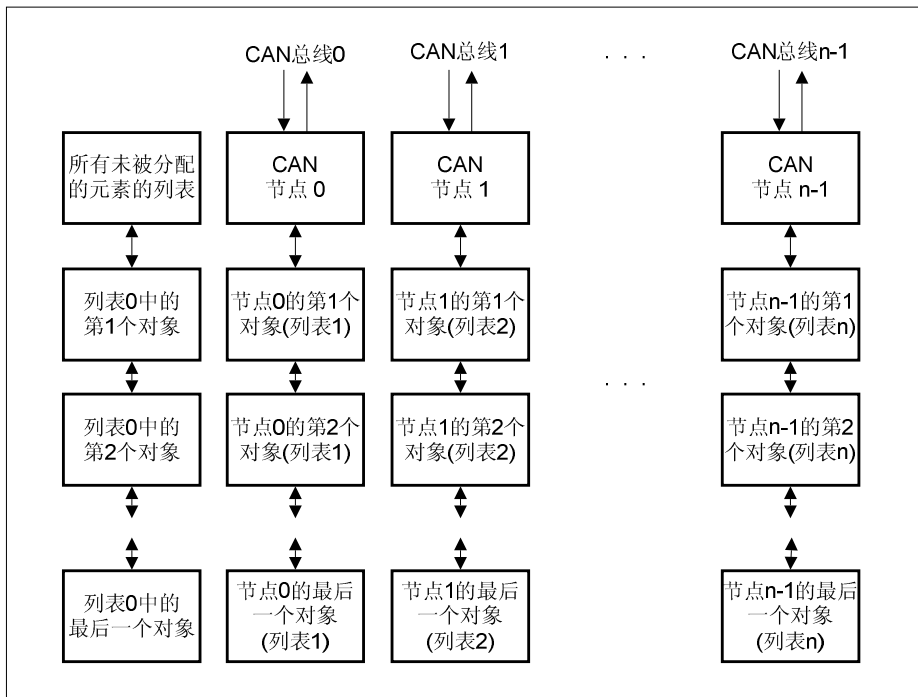
复位之后，所有的报文对象都未被分配。即所有报文对象都属于列表 0。此时，所有未被分配的报文对象的列表按照报文对象编号来排序，即在报文对象  $n$  之前是报文对象  $n-1$ ，之后是报文对象  $n+1$ 。

#### 22.2.4.3 与 CAN 节点连接

一个 CAN 节点和唯一的报文对象列表相关联...

**表 22-3      列表编号**

列表编号	描述
<b>0</b>	未被分配的元素列表
<b>1 至 <math>n\_nodes</math></b>	与 CAN 节点相关的列表。 列表编号 $i$ 属于 CAN 节点 $i-1$ 。
<b><math>n\_nodes+1</math> 至 <math>n\_lists-1</math></b>	自由用户列表，这些列表和 CAN 节点无关。



**图 22-6 和几个 CAN 节点相连的报文对象**

#### 22.2.4.4 列表命令面板

不能通过对列表 (LIST) 寄存器以及报文对象中的位域 PPREV、PNEXT 和 LIST 进行写操作的方式直接修改列表结构, 因为这些寄存器是只读寄存器。列表结构的管理仅限于通过 MultiCAN 模块内部的列表控制器执行。用户通过一个命令面板控制列表控制器, 向列表控制器发出列表分配命令。列表控制器有两个基本作用:

1. 确保所有修改列表结构的操作最终得到一致的列表结构
2. 使得用户在最大程度上方便、灵活的管理列表结构

列表控制器和相关的命令面板使得程序员只需注重列表的最终特性, 这些特性以 CAN 节点报文对象的分配情况, 以及分配给同一个列表的报文对象的次序关系为主要特征。由列表控制器完成列表构建 (重构) 过程。

通过向面板控制寄存器中的位域 PANCMD 写入相应命令码 (见 [页 22-43](#) 上的 [表 22-9 “命令面板”](#)) 的方式启动一个面板命令。在写命令码之前必须将相关的命令参数写入位域 PANAR1 和 PANAR2 中, 或者用一个 32 位写访问操作将命令参数和命令码一起写入到面板控制寄存器中 (只有在 32 位系统环境中才可能实现)。

写入一个有效的命令码之后，面板控制寄存器中的 **BUSY** 标志变为有效 ( $BUSY = 1$ )，且控制面板寄存器被锁定，这意味着对面板控制寄存器的进一步写访问将被忽略。**BUSY** 标志保持有效且控制面板保持锁定状态，直到所请求的命令执行完毕。

当发出的命令为动态分配的情况下，从未被分配的报文对象的列表中取报文元素，那么 **RBUSY** 和 **BUSY** 同时变为有效 ( $RBUSY = BUSY = 1$ )。这表明列表控制器将用以下的方法对位域 **PANAR1** 和 **PANAR2** 进行更新：

1. 从未被分配的报文元素中取出的报文对象的编号被写入到位域 **PANAR1**
2. 由 **PANAR2** 位7 (位7 = **ERR**) 指示错误状态。如果  $ERR = 1$ ，未被分配的元素列表为空且该命令被中止。如果  $ERR = 0$ ，那么未被分配的元素列表不为空且将成功执行该命令。

在列表控制器开始真正的分配过程之前写入动态分配命令的结果。一旦结果可用，**RBUSY** 再次变为无效 ( $RBUSY = 0$ )，而 **BUSY** 仍然保持有效，直到命令完成为止。这个特性允许用户在进行列表分配的过程中建立新的报文对象。进行列表操作的同时，对报文对象的访问不受限制。然而，对位于 **RAM** 内部的寄存器资源的任何访问都会将正在进行的分配过程延迟一个访问周期。

一旦完成该命令，**BUSY** 标志变为无效 ( $BUSY = 0$ )，对面板控制寄存器的写访问再次被使能，且 **NOP** 命令码被自动写入到面板控制寄存器的位域 **CMD** 中。**BUSY** 无效期间，可在任意时刻启动一个新命令。

面板控制寄存器中的所有位域 (**BUSY** 和 **RBUSY** 除外) 都可以由用户写入。如果命令面板需要被用在独立的 (可相互中断的) 中断程序中，这个特性允许对面板控制寄存器进行保存和恢复。如果是这种情况，那么任何使用命令面板的任务 (和可能利用命令面板中断另一个) 应该查询 **BUSY** 标志直到其变为无效，且在发出命令之前将整个 **PANCTR** 寄存器保存到一个存储器地址。在中断服务程序的结束部分，该任务应该从该存储器地址中恢复 **PANCTR**。

在将分配给有效的 **CAN** 节点列表的报文对象移动到另一个列表或者同一个列表的另一个位置之前，报文对象控制寄存器内的位 **MSGVAL** (“报文有效”) 必须被清除。

## **22.2.5 CAN 节点分析模式**

**CAN** 分析模式可监控 **CAN** 节点的通信，而不影响 **CAN** 总线逻辑状态。

### **22.2.5.1 分析模式**

通过置位节点控制寄存器中的位 **CALM**，可单独为每个 **CAN** 节点选择 **CAN** 分析模式。

在 **CAN** 分析模式下，**CAN** 节点的发送引脚始终保持为隐性电平。**CAN** 节点可以接收帧 (数据、远程和错误帧)，但是不允许进行发送操作。有效的错误帧被隐性发送。接收到的数据/远程帧不会被应答 (即应答时隙为隐性)，但是只要任何其它节点应答了该帧，该帧将被接收和保存在匹配报文对象中。

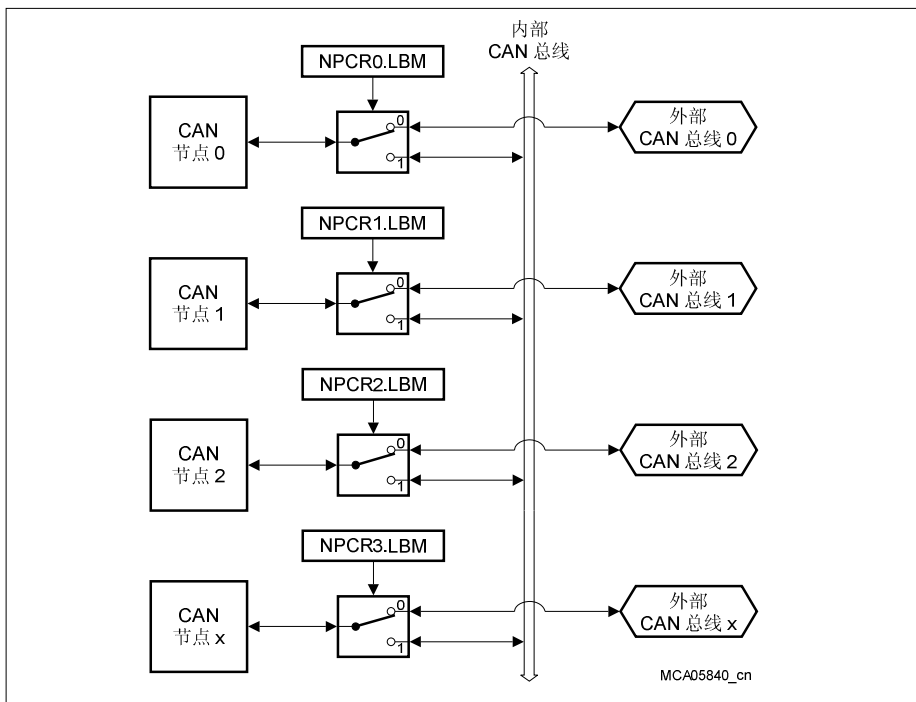
分析模式提供全部的报文对象功能，但是将不执行发送请求。

### 22.2.5.2 回环模式

MultiCAN 模块具有回环模式，能够进行在系统测试，同时无需访问外部 CAN 总线就可以进行 CAN 驱动软件的开发。

回环特性由一条内部 CAN 总线 (MultiCAN 模块内) 和每个 CAN 节点的总线选择开关构成 (见图 22-7)。通过该开关，控制每个 CAN 节点或者和内部 CAN 总线连接 (回环模式被激活)，或者和外部 CAN 总线相连，也就是说和发送和接收引脚相连 (正常操作)。当前未选中的 CAN 总线被驱动为隐性，即发送引脚保持为 1，处于回环模式的 CAN 节点忽略接收引脚。

置位相应的节点端口控制寄存器中的位 LBM 来为每个 CAN 节点选择回环模式。所有处于回环模式的 CAN 节点可以通过内部总线进行通信，而不影响其它不处于回环模式的 CAN 节点的工作。



**图 22-7 CAN 节点的回环模式**

### 22.2.5.3 位时序分析

每个 CAN 节点都通过 CAN 帧计数器的专用分析模式执行详细的位时序分析。帧计数器的位时序分析功能可以用于 CAN 波特率自动检测和 CAN 网络的时序分析。

通过设置 CAN 节点帧计数器寄存器内的位域 CFCMOD = 10<sub>B</sub> (位时序模式)，选择 CAN 节点的位时序分析功能。

位时序分析不影响 CAN 节点操作。

位时序的测量结果被写入到位域 CFC。在位时序分析模式下，只要位域 CFC 被更新，位 CFCOV 也被置位，用来指示 CFC 更新事件。如果 CFCIE 也被置位，则会产生一个中断请求，对于 CAN 节点  $i = 0$  至 5 来说，在中断输出线  $i$  上产生中断请求。

#### 自动波特率检测

对于自动波特率检测，必须测量 CAN 总线上观察到的连续显性沿之间的时间。如果 CAN 节点帧计数器寄存器内的位域 CFSEL = 000<sub>B</sub>，则自动执行该测量。监测 CAN 接收输入线上每个显性沿，该边沿和最近的显性沿之间的时间（测量以  $f_{CAN}$  时钟周期数为单位）被保存在位域 CFC 中。

#### 同步分析

如果 CFSEL = 010<sub>B</sub>，则进行位时间同步监测。测量第一个显性沿和采样点之间的时间，并将其保存在 CFC 中。位时序同步偏移可以从该时间得出，因为采样点之后的第一个边沿触发同步，且在连续的采样点之间仅有一个同步。

在接收第一个 CAN 帧期间测量到波特率之后，同步分析可被用于波特率精细调整。

#### 驱动延迟测量

当 CFSEL = 011<sub>B</sub> (显性到显性) 且 CFSEL = 001<sub>B</sub> (隐性到隐性) 时，测量发送沿和相应的接收沿之间的延迟。这些延迟指示将一个新的位值送到 CAN 总线上进行物理实现所需要的时间。



## 22.2.6 报文验收滤波

报文验收滤波包括接收和发送滤波两部分。

### 22.2.6.1 接收验收滤波

当从 CAN 节点接收 CAN 帧时，成功接收之后，接收到的帧存储在唯一的报文对象中。只有满足以下条件的报文对象才具备接收 CAN 帧的资格：

1. 这个报文对象被分配给接收该帧的 CAN 节点的列表。
2. 报文控制寄存器中的 MSGVAL 被置位。
3. 报文控制寄存器中的 RXEN 被置位。
4. 报文控制寄存器中的 DIR 等于接收帧的位 RTR。如果位 DIR = 1（发送对象），报文对象仅验收远程帧。如果位 DIR = 0（接收对象），报文对象仅验收数据帧。
5. 如果验收屏蔽寄存器（MOAMR）中的位 MIDE = 1，那么接收帧 IDE 位等于仲裁寄存器的（MOAR）中的 IDE 位：如果 MOAR.IDE = 1，报文对象仅验收带扩展标识符的接收帧；如果 MOAR.IDE = 0，那么报文对象仅验收标准帧。如果 MOAMR.MIDE = 0，不需要注意接收帧的 IDE 位。即报文对象可以验收标准帧和扩展帧。
6. 如寄存器 MOAMR 中验收屏蔽所限定的那样，接收帧的标识符和保存在报文对象仲裁寄存器中的标识符相匹配。这意味着接收报文对象标识符的每一位都和验收寄存器内对应的标识符位相等，除了那些对应于位域 MOMAR 中被清零的验收屏蔽位，这些标识符位对接收无影响。图 22-8 给出标识符的检查过程。

下面给出报文对象的优先级排序机制：

只要满足下面的条件，则报文对象 A 比报文对象 B 的接收优先级高：

1. A 具有比 B 更高的优先级。即 A 的 MOAR.PRI 必须小于或者等于 B 的 MOAR.PRI。
2. 如果两个报文对象具有相同的优先级（A 的 PRI = B 的 PRI），列表中 B 在 A 之后，即在列表中从 A 开始，通过顺序方式可到达 B。

满足所有这六条评判标准的报文对象中，具有最高接收优先级的唯一报文对象赢得接收验收滤波，即被选中用来储存接收帧。所有其它报文对象失去接收验收滤波。

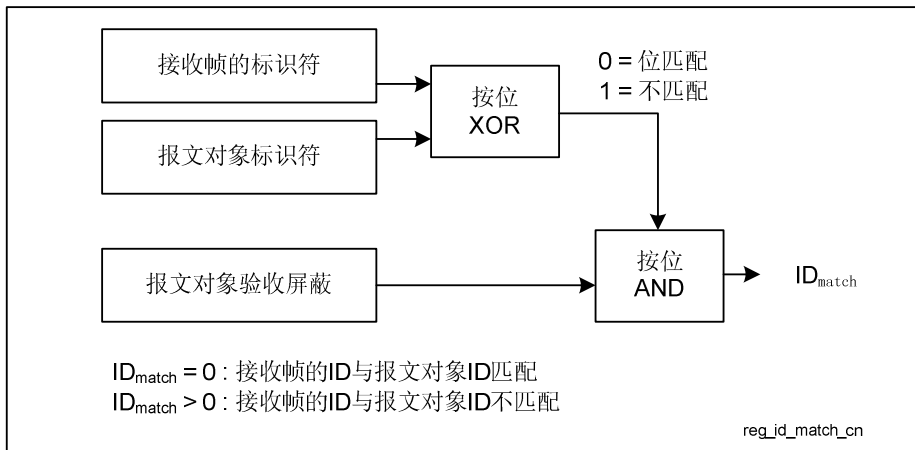


图 22-8 接收报文标识符验收检查

### 22.2.6.2 发送验收滤波

报文发送请求是通过在报文所属的报文对象中设置发送请求实现的。如果同一个 CAN 节点有多于一个报文对象具有有效的发送请求，只在其中选择一个报文对象发送，因为在一条 CAN 总线上同时仅可发送一个报文对象。

在给定 CAN 节点上，满足下述条件的报文对象才具备发送资格（见图 22-9）：

1. 报文对象分配给了 CAN 节点的报文对象列表。
2. 报文对象控制寄存器内的 MSGVAL 被置位。
3. 报文对象控制寄存器内的 TXRQ 被置位。
4. 报文对象控制寄存器内的 TXEN0 和 TXEN1 被置位。

由优先级机制判定最先发送哪一个报文对象。作如下假设：报文对象 A 和报文对象 B 都具备发送的资格，不失其通用性，假设 B 在 A 之后，即列表中从 A 开始，通过顺序方式可到达 B。对这两个报文对象来说，按如下参数定义 CAN 报文  $CAN_A$  和  $CAN_B$ ，其中标识符、IDE 和 RTR 位来自 MOAR.ID，MOAR.IDE 和 MOCTR.DIR。

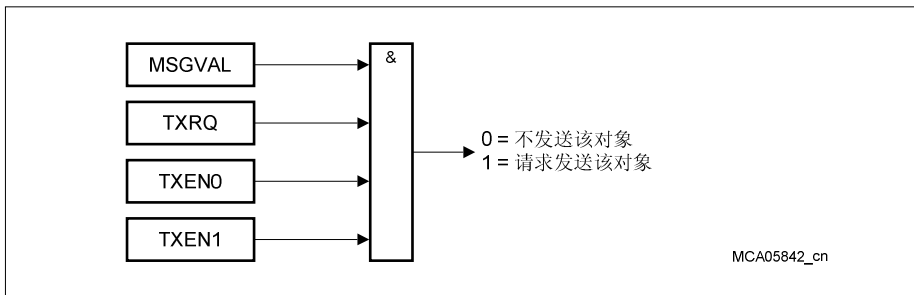
如果报文对象具有不同的优先级（报文对象仲裁寄存器 MOAR 中的 PRI 值不同），则 PRI 值较小的报文对象优先级较高，该报文对象被最先发送。

如果两个报文对象具有相同的优先级（寄存器 MOAR.PRI 位域的值相等），如果满足下面的条件中任何一个，那么报文对象 A 比 B 的发送优先级高。

1.  $PRI = 10$  且按照 CAN 仲裁规则，CAN 报文  $CAN_A$  具有比  $CAN_B$  更高或者相同的优先等级（见表 22-13）。
2.  $PRI = 01$  或者  $PRI = 11$ （由列表次序决定优先级）。

3. PRI = 00 保留。对于发送验收滤波来说, 不考虑 PRI = 00 的发送对象。

有发送资格的报文中发送优先级最高的唯一报文对象赢得发送验收滤波, 也就是说, 将最先被发送。所有其它的报文对象失去当前一轮的发送验收滤波机会。在下面的验收滤波中它们都将得到新的机会, 再次进行验收滤波。



**图 22-9 报文对象有效发送请求**

### 22.2.7 报文后处理接口

在报文对象成功发送或者接收一帧后, 可通知 CPU 在该报文对象上执行报文后处理。MultiCAN 模块的后处理接口包括两个单元:

1. 报文中断触发报文后处理
2. 报文挂起寄存器将挂起报文中断收集到一个公共结构中, 用于报文后处理。

#### 22.2.7.1 报文对象中断

将接收到的帧存储到报文对象中或者成功发送了一帧之后, 可以请求报文中断。对于每个报文对象, 可以将发送和接收中断单独送至可用的中断输出线之一 (见 [图 22-10](#))。除了由报文对象所属 CAN 节点直接存储接收帧的操作之外, 还可由 FIFO 或者网关动作导致的帧存储事件触发接收中断。不管相应的报文中断是否使能, 在一次成功的发送/接收操作之后, TXPND 和 RXPND 总是被置位。



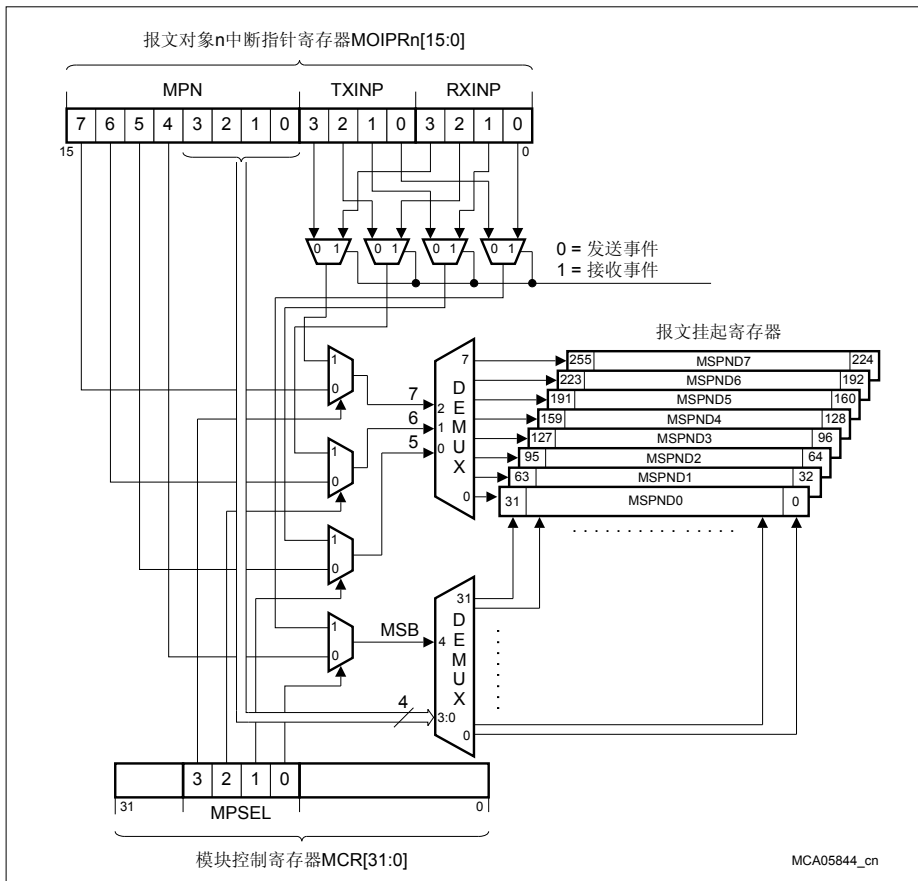


图 22-11 报文挂起位目标位置分配 (发送/接收)

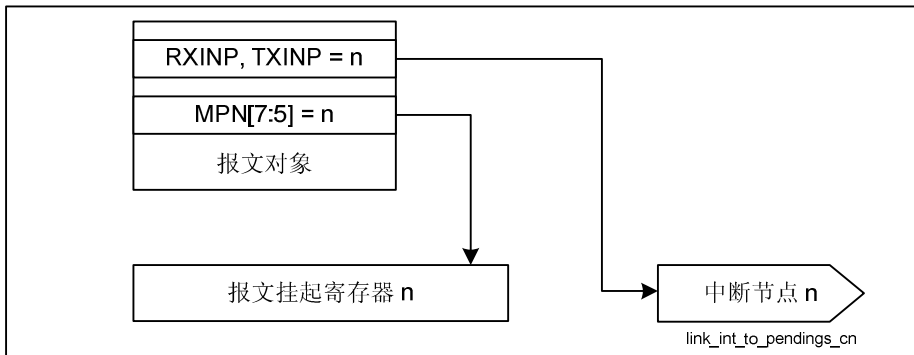


图 22-12 中断输出线和报文挂起寄存器相连的例子

## 22.2.8 报文对象数据处理

本节描述帧发送和帧接收期间进行的操作。

### 22.2.8.1 帧接收

从 CAN 总线接收一个报文之后，根据图 22-13 中的方案将其存储在报文对象中。MultiCAN 模块不仅将接收到的数据复制到报文对象中，而且还提供其它先进特性以实现 MultiCAN 和 CPU 之间一致的数据交换。

#### MSGVAL

报文对象控制寄存器中的位 MSGVAL (“报文有效”) 为报文对象的主控制位。帧接收过程中，当 MSGVAL 置位 (MSGVAL = 1) 时，MultiCAN 模块才在该报文对象中存储信息。

只要 MSGVAL 被 CPU 复位 (MSGVAL = 0)，MultiCAN 模块将停止所有正在进行的对该报文对象的写访问，因此随后 CPU 可用写操作重新配置报文对象，而且不被 MultiCAN 模块打扰。

#### RTSEL

CAN 操作期间，当 CPU 重新设置报文对象 (即清除 MSGVAL，修改报文对象和再次置位 MSGVAL)，将发生下列操作：

1. 该报文对象赢得接收验收滤波
2. CPU清除MSGVAL，以重新设置该报文对象
3. 重新设置报文对象后，CPU再次置位MSGVAL

## **控制器局域网 (MultiCAN)**

4. 到达接收帧尾。因为MSGVAL被置位，接收到的数据保存在该报文对象中，产生一个报文中断请求，处理网关和FIFO动作等等

如果报文对象的上下文已经改变，用户可能不希望存储接收数据，因为旧的报文对象设置已经被用于报文的验收滤波。

位 MOCTR.RTSEL (“接收/发送选择”) 允许报文对象与正在进行的帧接收断开：

当一个报文对象赢得接收验收滤波，MultiCAN 模块置位 RTSEL 位 (RTSEL = 1)，用来指示即将进行帧传送。MultiCAN 模块检查 RTSEL 是否因为成功接收一帧而置位，以验证这个报文对象是否仍然准备就绪，可以接收帧。只有当 RTSEL = 1 时，接收帧被保存在报文对象中（并进行后续动作，如报文中断，FIFO&网关动作，标志位更新）。

在 CAN 操作期间，用户使得报文对象失效时 (MSGCAL -> 0)，再次置位 MSGVAL (和置位 MSGVAL 的写操作最接近的) 之前，用户应当清除 RTSEL，以防止将一帧存储在属于报文对象的旧的上下文中。因此，应该按照下边的步骤重新设置报文对象：

1. 清除位MSGVAL
2. 当MSGVAL = 0 时，重新设置报文对象
3. 清除RTSEL位并置位MSGVAL

## **RXEN**

位 MOCTR.RXEN 用于使能该报文对象的帧接收操作。仅当 RXEN = 1 时，报文对象才能从 CAN 总线上接收 CAN 报文。只有在接收验收滤波期间，MultiCAN 模块才评估 RXEN 的值。接收验收滤波之后，忽略 RXEN，即 RXEN 对报文对象中接收报文的实际存储没有后续影响。

位 RXEN 使能所谓的报文对象“软逐步淘汰”：当用户清除 RXEN 时，当前接收的 CAN 报文仍然存储在已经赢得验收滤波报文对象中，但是对于后续的报文，该报文对象将不再赢得接收验收滤波。

## **RXUPD, NEWDAT 和 MSGLST**

报文对象控制寄存器内的标志 RXUPD (“接收更新”) 用来指示正在进行的帧存储过程。MultiCAN 模块在报文对象更新开始时 RXUPD 置位，结束时清除 RXUPD (包含帧存储和标志位更新)。

在存储接收帧 (标识符, IDE 位, DLC 和数据帧的数据位域) 之后，MultiCAN 模块置位报文对象的 NEWDAT (“新数据”)。如果此时 NEWDAT 已经被置位，置位 MSGLST (“报文丢失”) 指示数据的丢失情况。

CAN 操作期间，CPU 可利用 RXUPD 和 NEWDAT 标志位保持从报文对象中读取的帧数据的一致性。推荐用户按照下列步骤执行：

1. 清除 NEWDAT 位
2. 从报文对象中读取报文内容 (标识符, 数据等等)
3. 读取报文对象控制寄存器并检查是否 NEWDAT 和 RXUPD 都被清除, 如果不是, 返回到第 1 步
4. 因第 3 步成功完成, 那么读取的报文对象内容是一致的, 也就是说, 读操作期间报文对象的内容没有被 MultiCAN 模块更新

对于接收数据和远程帧, 位 RXUPD, NEWDAT 和 MSGLST 工作机制相同。

### 22.2.8.2 帧发送

报文对象发送的过程如图 22-14 所示。除了将要发送的报文内容 (标识符、IDE 位、RTR=DIR 位、DLC 和数据帧的数据域) 复制到报文对象所属的 CAN 节点的内部发送缓存中, 同时还要服务和监控一些状态标志位, 用来控制数据处理的一致性。

对于给定的报文对象来说, 在发送验收滤波之后远程和数据帧的发送过程相同。

### MSGVAL, TXRQ, TXEN0, TXEN1

页 22-22 内的 “MSGVAL” 帧接收操作中对于 MSGVAL 位的描述也适用于发送操作。

仅在报文对象控制寄存器中的 MSGVAL (“报文有效”)、TXRQ (“发送请求”)、TXEN0 (“发送使能 0”) 和 TXEN1 (“发送使能 1”) 都置位 (1) 的情况下, 才能发送报文 (见图 22-9)。尽管对于发送过程来说, 这些位的作用相同, 但是它们的定义不同:

表 22-4 报文发送操作中 MOCTR 中的位置位 (1)

位	描述
MSGVAL	<b>报文有效</b> 该位是报文对象的主控制位
TXRQ	<b>发送请求</b> 为标准发送请求位。只要一个报文对象要进行发送, CPU 就必须置位该位。在一次成功发送结束时, 除非还有新数据 (由 NEWDAT = 1 指示) 要发送, 否则 TXRQ 被自动清除。 当报文对象功能寄存器中的单次发送试验置位时 (STT = 1), 且当报文对象的内容被复制到 CAN 节点的发送帧缓存中时, TXRQ 已经被 MultiCAN 模块清除。 接收到的远程请求 (即在 CAN 总线上接收远程帧之后) 置位 TXRQ,



位	描述
	用来请求相应数据帧发送操作。
<b>TEXN0</b>	<p><b>发送使能 0</b></p> <p>该位可以由 CPU 暂时清除，当报文对象向数据域写新内容时，用来禁止该报文对象的发送。这样可以避免发送由新值和旧值混合在一起的内容不一致的报文帧。</p> <p>当 <b>TXEN0 = 0</b> 时，仍验收远程请求，但是数据帧的发送被挂起，直到 CPU 再次使能发送 (<b>TXEN0 = 1</b>)。</p>
<b>TEXN1</b>	<p><b>发送使能 1</b></p> <p>发送 FIFO 中，用该位选择 FIFO 结构中发送有效的报文对象。</p> <p>对于不是发送 FIFO 的元素的报文对象，<b>TXEN1</b> 可被永久置 1 或者用作第二个、独立的发送使能位。</p>

## RTSEL

当已经确定某个报文对象就是下一个要被发送的报文对象时（由验收滤波决定），MultiCAN 置位 MOCTR.RTSEL（“接收/发送选择”）。

当 MultiCAN 将报文对象被复制到发送缓存中时，检查位 RTSEL，仅在 RTSEL = 1 时，才发送报文。

成功发送报文之后，再次检查位 RTSEL，只有 RTSEL = 1，才能执行报文后处理。

要完成一个有效的报文对象重构，需要执行下面的步骤：

1. 清除 MSGVAL（“报文有效”）；
2. 当 MSGVAL = 0 时，重构报文对象；
3. 清除 RTSEL 并置位 MSGVAL；

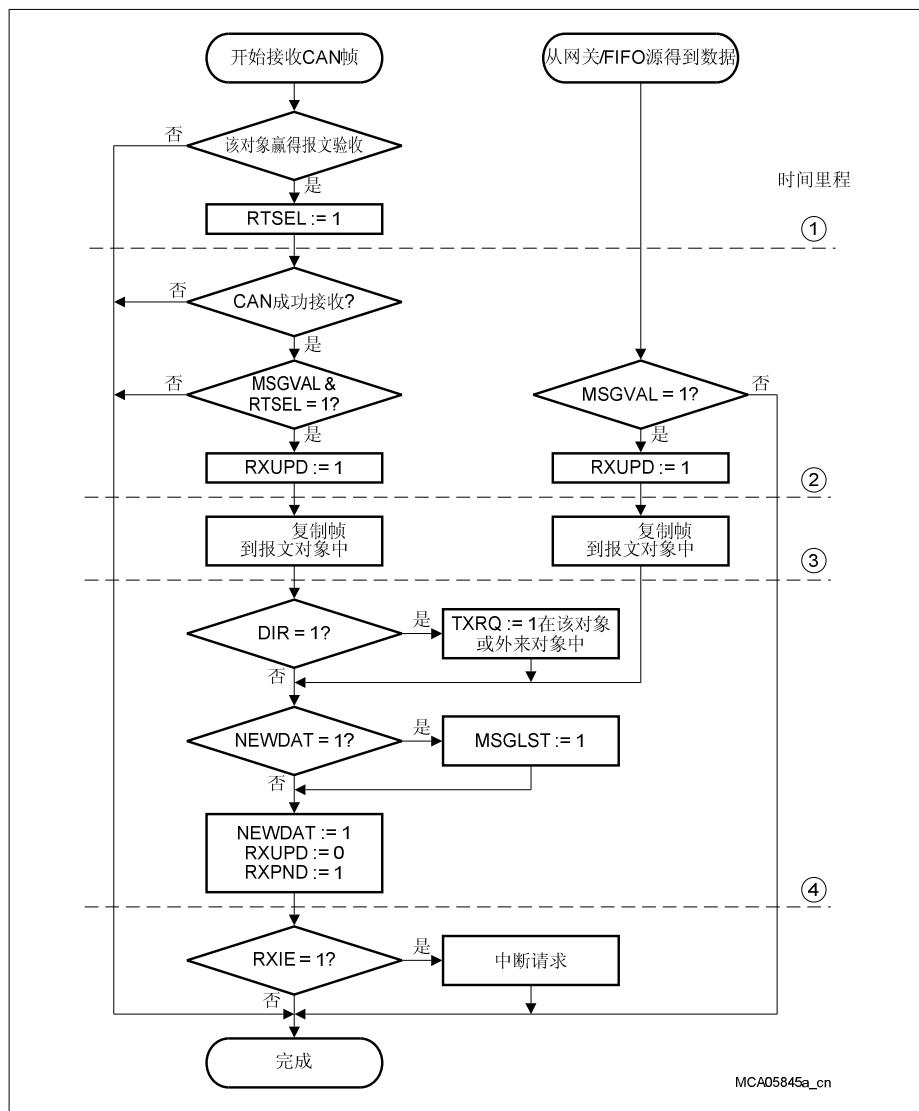
此时清除 RTSEL 是为了确保报文对象与一个正在进行/预定的发送断开，同时确保在报文对象再次变为有效之后，在旧的报文对象上下文之间不进行报文对象的处理（复制报文至发送缓存包括清零 NEWDAT、清除 TXRQ、时间戳更新、报文中断等等），总是在一个新的上下文中进行报文对象处理。

## NEWDAT

当报文对象的内容被传送到 CAN 节点的内部发送缓存中之后，位 NEWDAT（“新数据”）被清除，指示发送报文对象中的数据不再是新值。

当帧发送成功且 NEWDAT 仍然为 0（也就是说，其间没有新数据被复制到报文对象中），TXRQ（“发送请求”）被自动清除。

然而，如果 CPU 再次置位 NEWDAT（因为要发送新的一帧），则不清除 TXRQ，以使能新数据发送。



**图 22-13 MultiCAN 模块报文对象的输出传送**

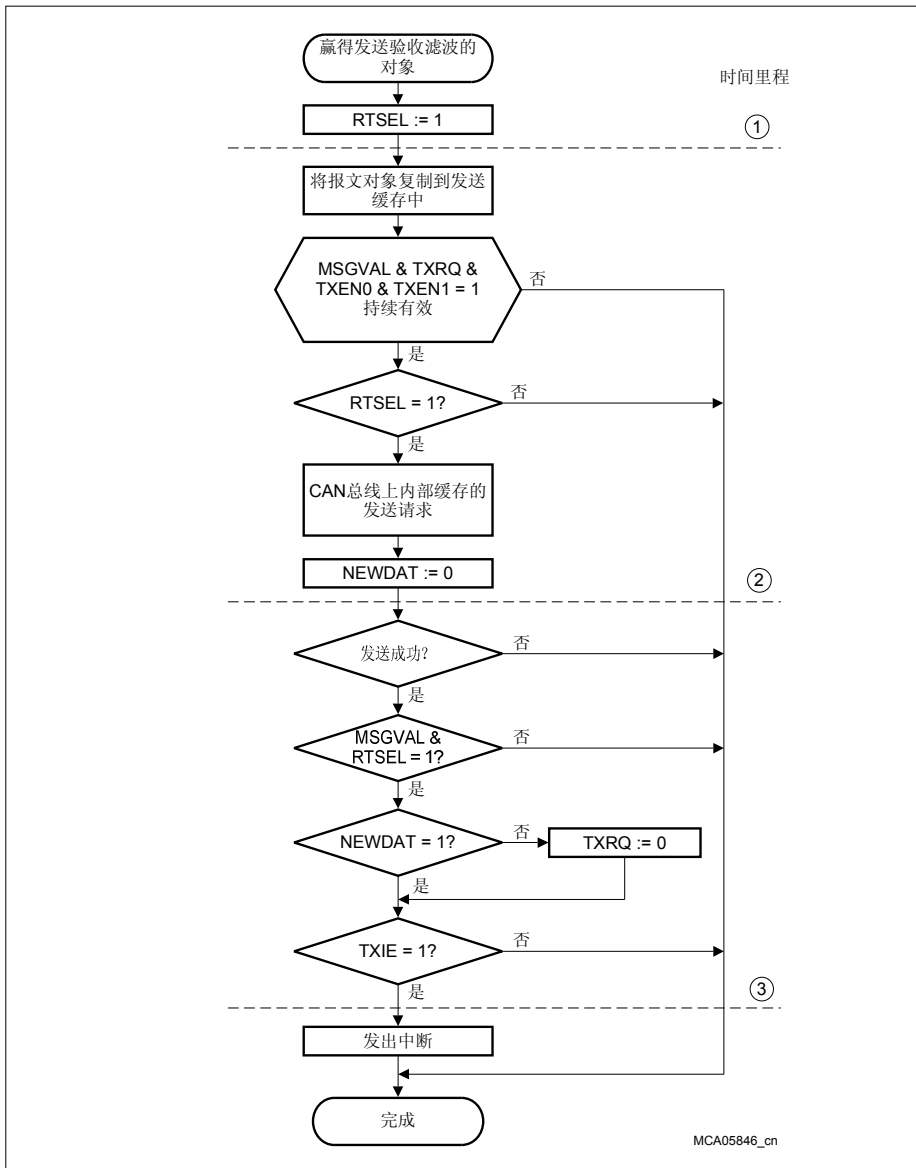


图 22-14 报文对象的发送

## 22.2.9 报文对象功能

本节描述 MultiCAN 模块报文对象的功能。

### 22.2.9.1 标准报文对象模式

当报文对象功能控制寄存器中的  $MMC = 0000_B$ ，选择标准报文对象功能。标准报文对象可根据前面各节描述的基本规则发送和接收 CAN 帧。还可以使用如单数据传送模式或者单次发送试验等附加功能（见下面的章节），用户可单独选择这些附加功能。

### 22.2.9.2 单数据传送模式

对于在 CAN 总线上广播数据且避免发送重复信息的应用中，单数据传送模式是一个有用的特性。通过报文对象功能寄存器中的位 SDT 选择该模式。

#### 报文接收

当存储在报文对象中的接收报文被 CPU 读取之前被新接收到的报文覆盖，则第一个报文的内容丢失被接收到的新值取代（由  $MSGLST = 1$  指示）。

如果  $SDT = 1$ （单数据传送模式有效），存储了接收到的数据帧之后，MultiCAN 控制器自动清除位 MSGVAL。该操作阻止了后续的报文接收。

在接收远程帧之后，位 MSGVAL 不会被自动清除。

#### 报文发送

当接收到一系列远程请求时，报文对象发送几个数据帧来回应远程请求。如果报文对象中的数据在两次发送期间未被更新，同样的数据在 CAN 总线上发送次数可以多于一次。

在单数据传送模式（ $SDT = 1$ ），因为在成功发送一个数据帧之后，自动清除 MSGVAL，从而避免了数据被再次发送。

在发送远程帧之后，位 MSGVAL 不会被自动清除。

### 22.2.9.3 单次发送试验

如果报文对象功能寄存器中的位 STT 置位（ $STT=1$ ），当报文对象的帧内容已经被复制到 CAN 节点的内部发送缓存中时，那么发送请求被清除（ $TXRQ := 0$ ）。因此，当因 CAN 总线错误而发送失败的话，不会再次尝试发送报文对象。

#### 22.2.9.4 报文对象 FIFO 结构

CPU 高负荷的情况下，及时的处理一串 CAN 帧可能比较困难。如果必须在短时间内发送或者接收多个报文，就可能发生这种情况。

因此，在 MultiCAN 模块中使用 FIFO 缓存结构来避免丢失接收到的报文，并减少发送报文的建立时间。FIFO 结构也可以用来自动接收或者发送一串 CAN 报文，当整个 CAN 帧接收/发送序列结束后，FIFO 可产生单报文中断。

也可以根据应用需要构造几个并行 FIFO，FIFO 的个数和大小仅受可用报文对象个数的限制。无论何时，即使是在 CAN 操作期间，都可以建立，重构其大小和取消 FIFO 结构。

FIFO 基本结构见图 22-15。FIFO 由一个基本报文对象（图左侧）和几个从属报文对象（图右侧）组成。从属报文对象由一个列表结构链接在一起。基本对象可以分配给任意一个列表。尽管图 22-15 所示的基本对象是在从属对象之外，其实也可以将基本对象结合到从属对象链表中的任何位置。这意味着：基本对象也是从属对象（这点在网关模式中是不可能的）。报文对象的绝对数目对于 FIFO 的操作没有影响。

不需要将基本对象和从属对象分配给同一个列表。只有从属对象必须分配给同一个列表（因为它们是链接在一起的）。不管基本对象被分配给和从属对象一样的列表或者另一个列表，由指针 BOT、CUR 和 TOP 将基本对象和从属对象链接起来。

最小的 FIFO 为一个报文对象，它既是 FIFO 基本对象也是 FIFO 从属对象（这种情况用处不大）。可能的最大 FIFO 结构将包含 MultiCAN 模块的所有报文对象。只能实现大小限制在上述两者之间的 FIFO 结构。

在 FIFO 基本对象中定义 FIFO 的边界。基本对象的 FIFO/网关指针寄存器中的位域 BOT 指向 FIFO 结构中第一个（最底部）从属报文对象。FIFO/网关指针寄存器中的位域 TOP 指向 FIFO 结构最后一个（顶端）从属报文对象。

FIFO/网关指针寄存器中的位域 CUR 指向实际上被 MultiCAN 模块选中进行报文传送的从属对象。该对象中的报文被传送之后，将 CUR 设置为列表中下一个从属对象。如果 CUR 已经到达了 FIFO 顶部（ $CUR = TOP$ ），CUR 的下次更新将使其从 FIFO 顶部回绕到底部（ $CUR := BOT$ ）。否则，CUR 设置为从属报文对象列表中的下一个报文对象（ $CUR := \text{当前报文对象的 PNEXT}$ ）。这种机制代表了一种循环 FIFO 结构，位域 BOT 和 TOP 建立了 FIFO 中第一个和最后一个元素之间的连接，线性结构不具备这样的特点。

基本对象的 FIFO/网关指针寄存器中的位域 SEL 可被用于监测。这个特性允许在列表中定义任意一个从属对象，如果 CUR 指针到达了 SEL 的值，就会在其上产生一个报文中断。因此，SEL 为用户提供一种非常方便的方法，允许检测报文传送序列（预先定义好的）的结束，或者当 FIFO 已满时向 CPU 发出一个警告中断。

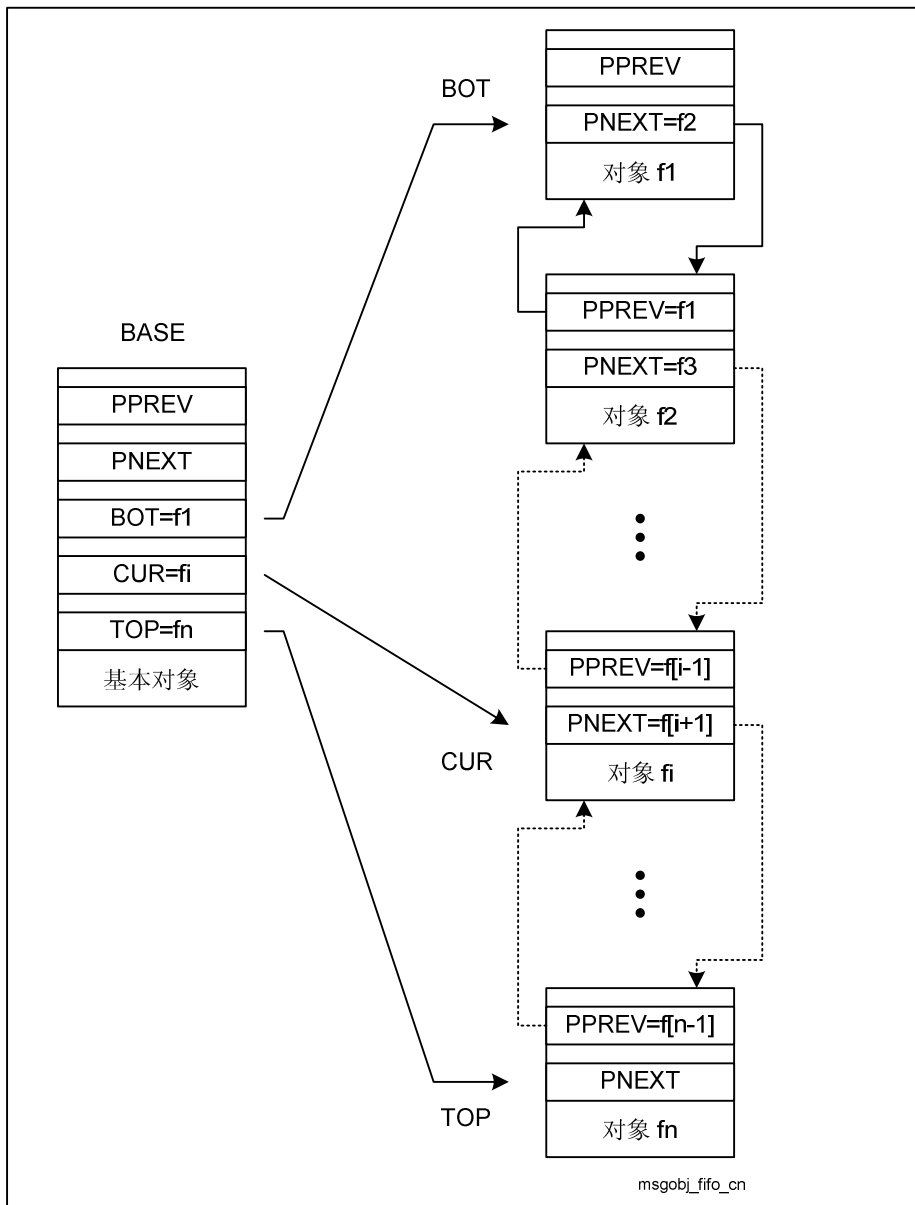


图 22-15 一个 FIFO 基本对象和  $n$  个 FIFO 目的对象 (从属) 的 FIFO 结构

### 22.2.9.5 接收 FIFO

接收 FIFO 结构用于缓存输入（接收）的远程或数据帧。

在 FIFO 基本对象的报文对象功能控制寄存器中设置  $MMC = 0001_B$ ，选择接收 FIFO。MMC 码自动指定该报文对象为 FIFO 基本对象。FIFO 从属报文对象的报文模式和接收 FIFO 操作无关。

当 FIFO 基本对象从其所属的 CAN 节点接收一帧时，该帧并不存储在基本对象中，而是存储在 FIFO 基本对象的 FIFO/网关指针寄存器中的 CUR 指针所选择的从属报文对象中。

由 CUR 选择的报文对象接收 CAN 报文，就像是其直接接收该帧一样。然而， $MMC = 0000_B$  隐含的假设 FIFO 从属对象执行的是标准报文传送。FIFO 从属对象实际的报文模式（MMC 设置）被忽略。从属对象对接收到的帧不进行验收滤波，即不检查匹配标识符，IDE 位和 DIR 位。

FIFO 基本对象接收 CAN 帧之后，MultiCAN 模块将当前指针 CUR 被设置为 FIFO 结构中的下一个报文对象。这个报文对象将用来储存下一个接收到的报文。CUR 的旧值被用于当前传送操作。

如果 FIFO 基本对象的报文对象功能寄存器中的位域 OVIE 置位且指针 CUR 等于 SEL 中存储的值，将产生一个 FIFO 溢出中断请求。在接收帧存储到从属报文对象中之后，立即在中断输出线 TXINP（基本对象的 TXINP）上产生中断请求。如果 TXIE 被置位，仍然产生发送中断。

仅在 FIFO 基本和从属对象中的  $MSGVAL = 1$  时，CAN 报文才保存在 FIFO 从属对象中。

为了避免由从属报文对象直接接收报文（就像是该从属报文对象为独立的报文对象而不是 FIFO 的组成部分），每个从属对象的位 RXEN 必须被清除。只有从属对象所在列表未分配给 CAN 节点的情况下，才“无需考虑”位 RXEN 的设置。

### 22.2.9.6 发送 FIFO

发送 FIFO 结构用于缓存要发送的一串数据或远程帧。发送 FIFO 由一个基本报文对象和一个或多个从属报文对象组成。

在 FIFO 基本对象中，通过报文对象功能控制寄存器中的  $MMC = 0010_B$  选择发送 FIFO。与接收 FIFO 不同，分配给发送 FIFO 的从属对象需要明确的设置它们的位域  $MMC = 0011_B$ 。所有从属对象中的 CUR 指针都必须指回到发送 FIFO 基本对象（由用户进行初始化）。

除了基本对象的 CUR 指针选择的报文对象，所有报文对象的 TXEN1 都必须被清除（由用户进行初始化）。CUR 选择的报文对象（从属对象）的 TXEN1 必须被置位。CUR（基本对象的）可被初始化为任意一个 FIFO 从属对象。

当 FIFO 的报文对象被设置为有效以启动 FIFO 操作时，必须首先将基本报文对象设置为有效（ $MSGVAL := 1$ ）。

操作期间，在取消一个发送 FIFO 之前，必须首先将其从属报文对象设置为无效 (MSGVAL := 0)。

发送 FIFO 用所有的 FIFO 元素的报文对象控制寄存器的位 TXEN1 位选择实际要发送的报文。发送验收滤波评估每个报文对象的 TXEN1，且只有其 TXEN1 置位的报文对象可以赢得发送验收滤波。当 FIFO 报文对象发送一帧报文之后，MultiCAN 除了执行标准的发送后处理（清除 TXRQ，发送中断等等），还清除该报文对象的 TXEN1 位，移动 FIFO 基本对象的 CUR 指针使其指向下一个要发送的报文对象。下一个报文对象的 TXEN1 被自动置位。因此，TXEN1 就像一个令牌，沿着发送 FIFO 结构移动，用来选择有效的 FIFO 元素。

如果 FIFO 基本报文对象的报文对象功能寄存器中的 OVIE 位域被置位且指针 CUR 等于 SEL 中所存储的值，将产生 FIFO 溢出中断请求。对接收帧进行后处理之后，在由 RXINP（基本对象的 RXINP）所定义的中断输出线上产生中断请求。如果置位 RXIE，仍然产生发送 FIFO 基本对象的接收中断。



### 22.2.9.7 网关模式

网关模式在两个独立 CAN 总线之间建立自动信息传送，而不需要和 CPU 相互作用。

网关模式在报文对象一级上工作。在网关模式下，信息的传送在两个报文对象之间进行，从而这两个报文对象所属的 CAN 节点之间也进行了信息传送。可在任意一对 CAN 节点之间建立网关，可以建立的网关个数和可用于构造网关结构的报文对象的个数一样多。

通过设置网关源对象的报文对象功能控制寄存器中的位域  $MMC = 0100_B$ ，选择网关模式。由源对象的 FIFO/网关指针寄存器中的 CUR 指针选择网关目标对象。只需将目标对象设置为有效（其  $MSGVAL = 1$ ）。所有其它设置都与源和目标对象之间的信息传送无关。

CAN 帧被接收并保存在源对象中之后，除了一些由 MultiCAN 模块执行的附加动作之外，网关源对象和标准报文对象的操作相似（见图 22-16）：

1. 如果源对象的报文对象功能寄存器中的位 DLCC 被置位，那么 DLC 编码被从网关源对象复制到网关目标对象。
2. 如果源对象的报文对象功能寄存器中的位 IDC 被置位，那么标识符和 IDE 位被从网关源对象复制到网关目标对象。
3. 如果源对象的报文对象功能寄存器中的位 DATC 被置位，那么数据域被从源对象复制到目标对象。
4. 如果源对象的报文对象功能寄存器中的位 GDFS 被置位，那么网关目标对象的报文对象控制寄存器中的 TXRQ 被置位。
5. 目标对象的报文对象控制寄存器中的 RXPND 和 NEWDAT 都被置位。
6. 如果目标对象的报文对象控制寄存器中的 RXIE 被置位，网关目标对象产生报文中断请求。
7. 根据章节 22.2.9.4 中描述的 FIFO 规则，网关源对象的 FIFO/网关指针寄存器的当前对象指针 CUR 移动到下一个目标对象。通过设置  $TOP = BOT = CUR =$  目标对象，可得到单（静态）目标对象网关。

网关源对象和目标对象之间与 FIFO 基本对象和 FIFO 从属对象之间的链接方式相同。这就意味着，可以生成具有 FIFO 结构的目标网关（见图 22-15），图 22-15 左边的报文对象为网关源对象，右边的报文对象为网关目标对象。

网关对接收数据帧（源对象为接收对象，即  $DIR = 0$ ）的操作和对接收远程帧的操作相同（源对象为发送对象）。

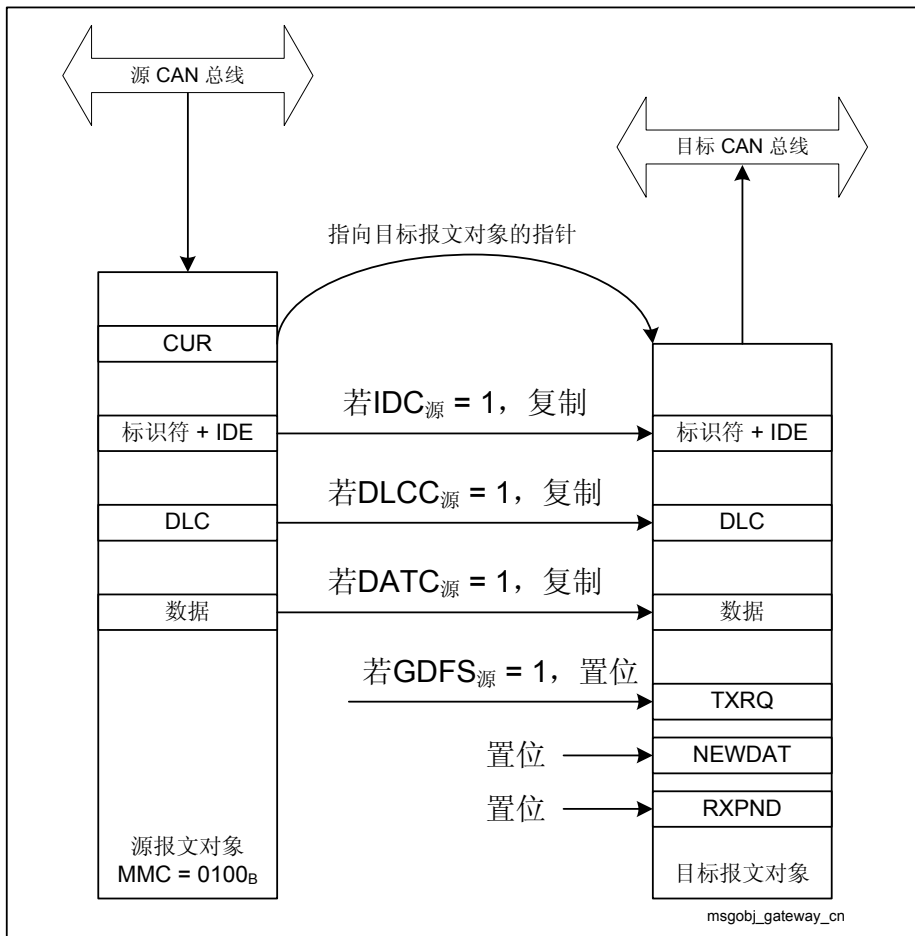


图 22-16 从源对象到目标对象的网关传送

### 22.2.9.8 外来远程请求

当在 CAN 节点上接收到一个远程帧并将其存储在报文对象中时，置位发送请求以触发对该请求的应答（数据帧的发送）或者自动发出第二个请求。如果存储远程请求的报文对象的功能控制寄存器中的位 FRREN 被清除（FRREN = 0），同一个报文对象的控制寄存器中的 TXRQ 被置位。

如果 **FRREN** 被置位 (**FRREN** = 1: 外来远程请求使能), 那么 FIFO/网关指针寄存器内的指针 **CUR** 指向的报文对象中的 **TXRQ** 被置位。然而, **CUR** 的值并未被该特性改变。

虽然外来远程请求特性和所选择的报文模式无关, 但是对于从网关目标总线上接收一个远程请求之后, 在源总线上发出一个远程请求的场合, 外来远程请求特性尤其有用。根据网关目标对象中 **FRREN** 的设置, 有两种方法可用于处理出现在目标方的远程请求 (假设源对象接收, 目标对象发送, 即 **DIR**<sub>源</sub>=0 且 **DIR**<sub>目标</sub>=1):

### 网关目标对象中 **FRREN** = 0

1. 网关目标对象接收到一个远程帧
2. 网关目标对象中的 **TXRQ** 被自动置位。
3. 保存在目标对象中的、带当前数据的数据帧被发送到目标总线上。

### 网关目标对象中 **FRREN** = 1

1. 网关目标对象接收到一个远程帧。
2. 网关源对象中的 **TXRQ** 被自动置位 (必须是目标对象 **CUR** 指针所指向的)。
3. 源对象 (作为接收对象) 在源 **CAN** 总线上发出远程请求。
4. 远程请求的接收器以数据帧回应源对象总线上。
5. 数据帧存储在源对象中。
6. 数据帧被复制到目标对象 (网关动作)。
7. 目标对象 **TXRQ** 置位 (假设 **GDFS**<sub>源</sub> = 1)。
8. 保存在目标对象中的新数据被发送到目标总线上, 作为对目标总线上的初始远程请求的回应。

## 22.3 MultiCAN 内核寄存器

MultiCAN 模块的寄存器组由下面三部分组成：

1. **全局模块寄存器**适用于整个 MultiCAN 模块，因此仅有一组。
2. **CAN 节点寄存器**适用于单个 CAN 节点，因此每个节点具有一组 CAN 节点寄存器。
3. **报文对象寄存器**的总和，用于定义单个报文对象，每个报文对象具有一组报文对象寄存器。

### 22.3.1 寄存器地址映射

表 22-5 给出 MultiCAN 模块的相对于基址的地址映射表。

表 22-5 MultiCAN 模块地址映射（相对于 MultiCAN 基址）

寄存器组	起始地址	总范围
全局模块寄存器	+ 100 <sub>H</sub>	+ 100 <sub>H</sub> 至 + 1FF <sub>H</sub>
CANx 的 CAN 节点寄存器，x = 0-5	+ 200 <sub>H</sub>	+ 200 <sub>H</sub> 至 + 7FF <sub>H</sub>
报文对象寄存器，n = 0-255	+ 1000 <sub>H</sub>	+ 1000 <sub>H</sub> 至 + 2FFF <sub>H</sub>

### 全局模块寄存器

仅有一组全局模块寄存器，表 22-6 列出其相对于全局模块寄存器起始地址的相对地址。

**表 22-6 全局模块寄存器的相对地址**

寄存器	相对地址	寄存器全名
LIST0L	100 <sub>H</sub>	列表寄存器 0 低位
LIST0H	102 <sub>H</sub>	列表寄存器 0 高位
LIST1L	104 <sub>H</sub>	列表寄存器 1 低位
LIST1H	106 <sub>H</sub>	列表寄存器 1 高位
LIST2L	108 <sub>H</sub>	列表寄存器 2 低位
LIST2H	10A <sub>H</sub>	列表寄存器 2 高位
LIST3L	10C <sub>H</sub>	列表寄存器 3 低位
LIST3H	10E <sub>H</sub>	列表寄存器 3 高位
LIST4L	110 <sub>H</sub>	列表寄存器 4 低位
LIST4H	112 <sub>H</sub>	列表寄存器 4 高位
LIST5L	114 <sub>H</sub>	列表寄存器 5 低位
LIST5H	116 <sub>H</sub>	列表寄存器 5 高位
LIST6L	118 <sub>H</sub>	列表寄存器 6 低位
LIST6H	11A <sub>H</sub>	列表寄存器 6 高位
LIST7L	11C <sub>H</sub>	列表寄存器 7 低位
LIST7H	11E <sub>H</sub>	列表寄存器 7 高位
MSPND0L	140 <sub>H</sub>	报文挂起寄存器 0 低位
MSPND0H	142 <sub>H</sub>	报文挂起寄存器 0 高位
MSPND1L	144 <sub>H</sub>	报文挂起寄存器 1 低位
MSPND1H	146 <sub>H</sub>	报文挂起寄存器 1 高位
MSPND2L	148 <sub>H</sub>	报文挂起寄存器 2 低位

**控制器局域网络 (MultiCAN)**

寄存器	相对地址	寄存器全名
MSPND2H	14A <sub>H</sub>	报文挂起寄存器 2 高位
MSPND3L	14C <sub>H</sub>	报文挂起寄存器 3 低位
MSPND3H	14E <sub>H</sub>	报文挂起寄存器 3 高位
MSPND4L	150 <sub>H</sub>	报文挂起寄存器 4 低位
MSPND4H	152 <sub>H</sub>	报文挂起寄存器 4 高位
MSPND5L	154 <sub>H</sub>	报文挂起寄存器 5 低位
MSPND5H	156 <sub>H</sub>	报文挂起寄存器 5 高位
MSPND6L	158 <sub>H</sub>	报文挂起寄存器 6 低位
MSPND6H	15A <sub>H</sub>	报文挂起寄存器 6 高位
MSPND7L	15C <sub>H</sub>	报文挂起寄存器 7 低位
MSPND7H	15E <sub>H</sub>	报文挂起寄存器 7 高位
MSID0	180 <sub>H</sub>	报文指针寄存器 0
MSID1	184 <sub>H</sub>	报文指针寄存器 1
MSID2	188 <sub>H</sub>	报文指针寄存器 2
MSID3	18C <sub>H</sub>	报文指针寄存器 3
MSID4	190 <sub>H</sub>	报文指针寄存器 4
MSID5	194 <sub>H</sub>	报文指针寄存器 5
MSID6	198 <sub>H</sub>	报文指针寄存器 6
MSID7	19C <sub>H</sub>	报文指针寄存器 7
MSIMASKL	1C0 <sub>H</sub>	报文指针屏蔽寄存器低位
MSIMASKH	1C2 <sub>H</sub>	报文指针屏蔽寄存器高位
PANCTRL	1C4 <sub>H</sub>	面板控制寄存器低位
PANCTRH	1C6 <sub>H</sub>	面板控制寄存器高位
MCR	1C8 <sub>H</sub>	模块控制寄存器
MITR	1CC <sub>H</sub>	模块中断触发寄存器

**控制器局域网络 (MultiCAN)**

寄存器	相对地址	寄存器全名
-	+120 <sub>H</sub> ... +13E <sub>H</sub> +148 <sub>H</sub> ... +17E <sub>H</sub> +188 <sub>H</sub> ... +1BE <sub>H</sub> +1CE <sub>H</sub> ... +1FE <sub>H</sub>	保留

### CAN 节点寄存器

CAN 节点寄存器位于连续的 32 位地址内，表 22-7 给出 32 位 CAN 节点寄存器相对于 CAN 节点寄存器模块基址的相对地址，每个 CAN 节点对应一组 CAN 节点寄存器组。

**表 22-7      CAN 节点寄存器的相对地址**

寄存器	相对地址	寄存器全名
NCR	+00 <sub>H</sub>	CAN 节点控制寄存器
NSR	+04 <sub>H</sub>	CAN 节点状态寄存器
NIPR	+08 <sub>H</sub>	CAN 节点中断指针寄存器
NPCR	+0C <sub>H</sub>	CAN 节点端口控制寄存器
NBTRL	+10 <sub>H</sub>	CAN 节点位时序寄存器低位
NBTRH	+12 <sub>H</sub>	CAN 节点位时序寄存器高位
NECNTL	+14 <sub>H</sub>	CAN 节点错误计数器低位
NECNTH	+16 <sub>H</sub>	CAN 节点错误计数器高位
NFCRL	+18 <sub>H</sub>	CAN 节点帧计数器低位
NFCRH	+1A <sub>H</sub>	CAN 节点帧计数器高位
-	+1C <sub>H</sub> 至 +FE <sub>H</sub>	保留

### 报文对象寄存器

报文对象寄存器位于连续的 32 位地址内，表 22-8 给出 32 位报文对象寄存器相对于报文对象寄存器模块基址的相对地址。

**表 22-8 报文对象寄存器的相对地址**

寄存器	相对地址	寄存器全名
MOFCRL	+00 <sub>H</sub>	报文对象功能控制寄存器低位
MOFCRH	+02 <sub>H</sub>	报文对象功能控制寄存器高位
MOFGPRL	+04 <sub>H</sub>	报文对象 FIFO/网关指针寄存器低位
MOFGPRH	+06 <sub>H</sub>	报文对象 FIFO/网关指针寄存器高位
MOIPRL	+08 <sub>H</sub>	报文对象中断指针寄存器低位
MOIPRH	+0A <sub>H</sub>	报文对象中断指针寄存器高位
MOAMRn	+0C <sub>H</sub>	报文对象验收屏蔽寄存器低位
MOAMRn	+0E <sub>H</sub>	报文对象验收屏蔽寄存器高位
MODATALL	+10 <sub>H</sub>	报文对象数据寄存器低位的低半部分
MODATALH	+12 <sub>H</sub>	报文对象数据寄存器低位的高半部分
MODATAHL	+14 <sub>H</sub>	报文对象数据寄存器高位的低半部分
MODATAHH	+16 <sub>H</sub>	报文对象数据寄存器高位的高半部分
MOARL	+18 <sub>H</sub>	报文对象仲裁寄存器低位
MOARH	+1A <sub>H</sub>	报文对象仲裁寄存器高位
MOCTRL MOSTATL	+1C <sub>H</sub>	报文对象控制寄存器低位 报文对象状态寄存器低位
MOCTRH MOSTATH	+1E <sub>H</sub>	报文对象控制寄存器高位 报文对象状态寄存器高位



## 22.3.2 全局 MultiCAN 寄存器

### 22.3.2.1 模块 ID 寄存器

**ID**

**模块 ID 寄存器**

**(08H)**

**复位值: 4001<sub>H</sub>**

16

8 7

0

MOD_NUMBER	MOD_REV
r	r

符号	位序号	类型	功能描述
<b>MOD_REV</b>	[7:0]	r	<b>模块修订版编号</b> 位 7-0 给出模块修订版编号，从 01 <sub>H</sub> 开始（第一版）、02 <sub>H</sub> 、03 <sub>H</sub> 、...直到 FF <sub>H</sub> 。
<b>MOD_NUMBER</b>	[15:8]	r	<b>模块 ID 编号</b> 位 15-8 给出模块 ID 编号。MultiCAN 的模块编号为 40 <sub>H</sub> 。

### 22.3.2.2 命令面板

所有列表操作都是通过命令面板执行的，如列表结构中报文对象的分配，解除分配，重新分配。不能由软件通过向报文对象和列表寄存器进行写操作的方式直接修改列表结构。

通过在面板控制寄存器中设置命令参数和命令码的方式启动新命令。

#### PANCTRH

面板控制寄存器

(1C6<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PANAR2								PANAR1							
rwh								rwh							

符号	位序号	类型	功能描述
PANAR1	[7:0]	rwh	面板参数 1
PANAR2	[15:8]	rwh	面板参数 2

#### PANCTRL

面板控制寄存器

(1C4<sub>H</sub>)

复位值: 0301<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						RBUSY	BUSY	PANCMD							
r						rh	rh	rwh							

符号	位序号	类型	功能描述
PANCMD	[7:0]	rwh	命令面板 将面板命令码写入该位域，可启动一个新命令。在一个面板命令结束时，NOP（无操作）命令码被自动写入该位域。
BUSY	8	rh	面板忙标志位 0 <sub>B</sub> 面板已完成命令，准备好接受新命令 1 <sub>B</sub> 面板操作正在进行
RBUSY	9	rh	结果忙标志位

**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
			<p>0<sub>B</sub> 列表控制器没有更新 PANAR1 和 PANAR2 的计划</p> <p>1<sub>B</sub> 正在运行的列表命令 (BUSY=1) 将向 PANAR1 和 PANAR2 写结果, 但是结果目前还不可用</p>
<b>0</b>	[15:10]	r	<p><b>保留</b></p> <p>读返回为 0; 应该写入 0。</p>

**面板命令**

一个面板操作命令由一个命令码 (写入到 PANCMD) 和多达两个面板参数 (PANAR1, PANAR2) 组成。带返回值的命令将返回值传递至位域 PANAR1。返回错误标志位的命令将错误标志位传递至 PANAR2 的位 7。

**表 22-9 面板命令**

命令码	PANAR2	PANAR1	命令描述
<b>0</b>			<p><b>无操作</b></p> <p>向 PANCMD 写 0 无任何影响。不启动新命令。</p>
<b>1</b>	<p><b>结果:</b></p> <p>位 7 :ERR</p> <p>位 6-0 :未定义</p>		<p><b>列表初始化</b></p> <p>运行初始化序列, 复位所有报文对象的位域 CTRL 和 LIST。列表寄存器 LIST[7:0]设置为其复位值。这将导致所有报文对象的分配被解除。初始化命令需要所有 CAN 节点 0-5 的节点控制寄存器中的位 INIT 和位 CCE 被置位。</p> <p>ERR 位 (PANAR2 的位 7) 报告该操作的是否成功:</p> <p>0 初始化成功</p> <p>1 并非所有的 INIT 和 CCE 都被置位。因此不执行初始化操作</p> <p>每次复位 MultiCAN 模块, 都将自动执行初始化命令, 但是所有报文对</p>

**控制器局域网络（MultiCAN）**

命令码	PANAR2	PANAR1	命令描述
			象被复位的情况例外。
<b>2</b>	<b>参数:</b> 列表指针	<b>参数:</b> 报文对象编号	<b>静态分配</b> 将给定报文对象分配给一个列表。 将报文对象从其当前所属的列表中除去，并将其添加到 PANAR2 给出的列表的最后。  该命令还可用于解除报文对象的分配。在这种情况下，目标列表是未被分配报文元素的列表（PANAR2 = 0）。
<b>3</b>	<b>参数:</b> 列表指针  <b>结果:</b> 位 7 : ERR 位 6-0: 未定义	<b>结果:</b> 报文对象编号	<b>动态分配</b> 将未分配报文对象列表中的第一个报文对象分配给所选择的列表。将该报文对象添加到列表的最后。报文对象的编号返回给 PANAR1。  <b>ERR 位（PANAR2 位 7）报告操作是否成功:</b>  0 <sub>B</sub> 成功  1 <sub>B</sub> 未执行操作，因为未被分配的报文对象列表为空。
<b>4</b>	<b>参数:</b> 目标对象编号	<b>参数:</b> 源对象编号	<b>静态插入到目标对象之前</b> 将一个报文对象（源对象）从其当前所属的列表中除去，并将其插入到给定目标对象所属的列表结构中，位置在给定目标对象之前。  源对象因而成为了目标对象的前一个报文对象。
<b>5</b>	<b>参数:</b> 目标对象编号  <b>结果:</b> 位 7 : ERR 位 6-0: 未定义	<b>结果:</b> 被插入的报文对象编号	<b>动态插入到给定目标对象之前</b> 在给定目标对象之前插入一个新报文对象。新报文对象从未被分配的报文元素列表中取出（选择第一个元素）。新报文对象的编号作为结果返回给 PANAR1。  ERR 位（PANAR2 位 7）报告操作

**控制器局域网络（MultiCAN）**

命令码	PANAR2	PANAR1	命令描述
			是否成功。 0 <sub>B</sub> 成功 1 <sub>B</sub> 未被分配的报文对象列表为空，因此未执行该操作。
<b>6</b>	<b>参数：</b> 目标对象编号	<b>参数：</b> 源对象编号	<b>静态插入到目标对象之后</b> 将一个报文对象（源对象）从其当前所属的列表中除去，并将其插入到给定目标对象所属的列表结构中，位置在给定的目标对象之后。源对象因而成为了目标对象的后一个报文对象。
<b>7</b>	<b>参数：</b> 目标对象编号 <b>结果：</b> 位 7 : ERR 位 6-0: 未定义	<b>结果：</b> 被插入的报文对象编号	<b>动态插入到给定目标对象之后</b> 在给定目标对象之后插入新报文对象。从未分配的报文元素列表中取出新报文对象（选择第一个元素）。新报文对象的编号作为结果保存在 PANAR1 中。 ERR 位（PANAR2 位 7）报告操作是否成功。 0 <sub>B</sub> 成功 1 <sub>B</sub> 未被分配的报文对象列为空，因此未执行该操作。
<b>8 - 255</b>	-	-	<b>保留</b>

### 22.3.2.3 模块控制寄存器

模块控制寄存器包含用于定义 MultiCAN 模块操作的基本设置。

#### MCR

##### 模块控制寄存器

(1C8<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
MPSEL	[15:12]	rw	<b>报文挂起选择</b> 在报文接收/发送之后，位域 MPSEL 可被用于计算报文挂起位位置，由位域 RXINP、TXINP 和 MPN（报文挂起编号）共同选择。 INP ... 报文接收操作则使用 RXINP 报文发送操作则使用 TXINP MPN ...8 位报文挂起编号 报文挂起位的有效位置按照如下公式进行计算： $POS = ( (INP \& MPSEL) << 4 )   (MPN \& (\sim MPSEL << 4) )   (MPN \& = 0x0F_H)$ 如果 MPSEL = 0，则挂起位位置由报文挂起位编号 MPN 给出。 如果 MPSEL = 1111 <sub>B</sub> ，则挂起位位置的高 4 位由中断输出线指针 INP 给出，低 4 位来自 MPN。
0	[11:0]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 22.3.2.4 中断触发寄存器 ITR

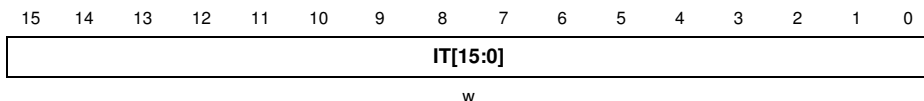
中断触发寄存器 ITR 允许由软件在每条中断输出线上触发中断请求。

#### MITR

模块中断触发寄存器

(1CC<sub>H</sub>)

复位值: 0000<sub>H</sub>



符号	位序号	类型	功能描述
IT[15:0]	[15:0]	w	<b>中断触发</b> 向位 $n$ 写 1 ( $n = 15-0$ )，在中断输出线 $n$ 上产生中断请求；写 0 无任何影响。读位域 IT 始终返回 0。使用一个写操作向 IT 的多个位写 1，则同时能产生多个中断请求。

### 22.3.2.5 列表指针

每个 CAN 节点都有一个分配给该节点的报文对象的列表。另外，还有一个未被分配的报文对象列表。此外，还有一个不与 CAN 节点相联系的报文对象通用列表可供使用。根据 [页 22-12 表 22-3 “列表编号”](#) 给每个列表分配指针。

#### LIST0H

列表寄存器 0 高位

(102<sub>H</sub>)

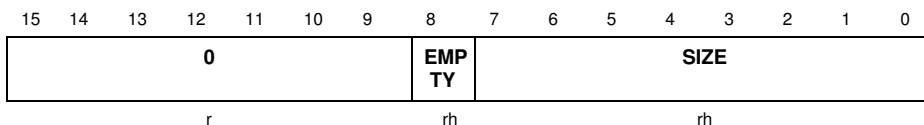
复位值: 007F<sub>H</sub>

LISTyH (y =1-7)

列表寄存器 y 高位

(102<sub>H</sub>+y\*4)

复位值: 0100<sub>H</sub>



符号	位序号	类型	功能描述
SIZE	[7:0]	rh	<b>列表大小</b> 如果列表 I 不为空，则元素个数由下式给定：

控制器局域网络 (MultiCAN)

符号	位序号	类型	功能描述
			#元素 = SIZE + 1 如果列表 I 为空, 则 SIZE = 0。
<b>EMPTY</b>	8	rh	<b>列表为空指示</b> 0 <sub>B</sub> 至少给列表 I 分配了一个报文对象。 1 <sub>B</sub> 未给列表 I 分配报文对象。
<b>0</b>	[15:9]	r	<b>保留</b> 读操作返回 0; 应写入 0。

**LIST0L**

列表寄存器 0 低位

(100<sub>H</sub>)

复位值: 7F00<sub>H</sub>

**LISTxL (x =1-7)**

列表寄存器 x 低位

(100<sub>H</sub>+x\*4)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>END</b>								<b>BEGIN</b>							
rh								rh							

符号	位序号	类型	功能描述
<b>BEGIN</b>	[7:0]	rh	<b>列表始端</b> 指向列表 I 第一个报文对象。
<b>END</b>	[15:8]	rh	<b>列表末端</b> 指向列表 I 最后一个报文对象。



### 22.3.2.6 报文通知

当报文对象因报文发送/接收操作而产生一个中断请求时，该请求将被送至报文对象的位域 TXINP 或 RXINP 所选择的中断输出线上。因为报文对象个数多于中断输出线个数，因此通常一个中断程序可处理来自多个报文对象的请求。因此，MultiCAN 模块实现了一个优先级选择机制，在报文对象集合中选择具有最高优先级的报文对象。报文挂起寄存器包含挂起的中断请求。

#### MSPNDkH (k=0-7)

报文挂起寄存器 k 高位 (142<sub>H</sub>+k\*4<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PND[31:16]															
rwh															

符号	位序号	类型	功能描述
PND[31:16]	[15:0]	rwh	<b>报文挂起</b> 当出现一个报文中断时，报文对象置位 MSPND 寄存器中的一位，该位的位置由报文对象的 IPR 寄存器 MPN[4:0]位域给出。寄存器选择 n 由 MPN 最高位给出。 由软件清除寄存器位（写 0），写 1 无影响。

#### MSPNDkL (k=0-7)

报文挂起寄存器 k 低位 (140<sub>H</sub>+k\*4<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PND[15:0]															
rwh															

符号	位序号	类型	功能描述
PND[15:0]	[15:0]	rwh	<b>报文挂起</b> 与 PND[31:26]的描述相同。

**控制器局域网络 (MultiCAN)**

每个报文挂起寄存器和一个报文指针寄存器相关联。报文指针寄存器给出一组挂起位中位置最低的有效（置位的）挂起位。

**MSIDk (k=0-7)**

**报文指针寄存器 k**

**(180<sub>H</sub>+k\*4<sub>H</sub>)**

**复位值: 0020<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>										<b>INDEX</b>					
r										rh					

符号	位序号	类型	功能描述
<b>INDEX</b>	[5:0]	rh	<b>报文挂起指针</b> INDEX 的值由具有下列特性 MSPNDk 挂起位的位 i 给出： <ol style="list-style-type: none"> <li>MSPNDk[i] &amp; IM[i] = 1</li> <li>i = 0 或者 MSPNDk[i-1:0] &amp; IM[i-1:0] = 0</li> </ol> 如果 MSPNDk 中没有满足上述条件的位，那么 INDEX 被读为 100000 <sub>B</sub> 。 因而，INDEX 给出 MSPNDk 中的第一个挂起位的位置，只考虑那些被报文指针屏蔽寄存器选中的位。
<b>0</b>	[15:6]	r	<b>保留；</b> 读操作返回 0；应写入 0。

**控制器局域网络 (MultiCAN)**

报文指针屏蔽寄存器选择计算报文挂起指针的位。所有的报文挂起寄存器及相关的报文指针寄存器共用报文指针屏蔽寄存器。

**MSIMASKH**

报文指针屏蔽寄存器

(1C2<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
IM[31:16]															
rw															

符号	位序号	类型	功能描述
IM[31:16]	[15:0]	rw	<b>报文指针屏蔽</b> 只有 MSPNDk 中的那些对应指针屏蔽位被置位的位才被用于计算报文指针。

**MSIMASKL**

报文指针屏蔽寄存器

(1C0<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
IM[15:0]	[15:0]	rw	<b>报文指针屏蔽</b> 只有 MSPNDk 中的那些对应指针屏蔽位被置位的位才被用于计算报文指针。

## 22.3.3 CAN 节点寄存器

MultiCAN 模块的每个 CAN 节点包含一组 CAN 节点寄存器。这些寄存器包含直接和 CAN 节点操作的相关信息，这些信息不能在节点间共享。

### 22.3.3.1 节点控制寄存器

节点控制寄存器包含确定 CAN 节点操作的基本设置，以及 CAN 节点与报文对象之间的相互作用。

#### NCRx (x=0-5)

节点 x 控制寄存器

(200H+x\*100H)

复位值: 0001H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0							SUS EN	CAL M	CCE	0	CAN DIS	ALIE	LECI E	TRIE	INIT
r							rw	rw	rw	r	rw	rw	rw	rw	rwh

符号	位序号	类型	功能描述
INIT	0	rwh	<p><b>节点初始化</b></p> <p><b>0<sub>B</sub></b> 复位 INIT 用于使能该节点，使其可参加 CAN 通信。</p> <p>如果 CAN 节点处于总线关闭状态，那么继续进行总线关闭的恢复操作（该操作不依赖于 INIT 位）。总线关闭恢复序列结束时，允许该节点参加 CAN 通信。</p> <p>如果 CAN 节点不处于总线关闭状态，在允许该节点参加 CAN 通信之前，必须要检测到 11 个连续的隐性位。</p> <p><b>1<sub>B</sub></b> 置位该位将终止该 CAN 节点的 CAN 通信。取消任何正在进行的帧传送，发送线变为隐性。</p> <p>如果该节点处于总线关闭状态，那么继续运行总线关闭恢复序列，如果在成功完成总线关闭恢复序列之后，位 INIT 仍然为 1，即在检测到 128 次连续 11 个隐性位 (11×1)，CAN 节点退出总线关闭状态，但只要 INIT 为 1，那么总线仍然是无效的。</p> <p>当 CAN 节点变为总线关闭状态（见 <a href="#">页 22-9</a>），</p>

**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
			INIT 自动置位。
<b>TRIE</b>	1	rw	<b>传送中断使能</b> 如果该位置位，那么因成功接收或发送一个 CAN 帧而产生中断请求。通过 CAN 节点中断指针寄存器中的 TRINP 选择中断输出线。
<b>LECIE</b>	2	rw	<b>LEC 指示错误中断使能</b> 如果该位置位，则每次更新节点状态寄存器 LEC 位域，使得 $LEC > 0$ (CAN 协议错误) 时，都会产生中断请求。通过 CAN 节点中断指针寄存器中的 LECINP 选择中断输出线。
<b>ALIE</b>	3	rw	<b>中断警报使能</b> 如果该位置位，则下列事件之一将产生一个警告中断： <ol style="list-style-type: none"> <li>1) CAN 节点状态寄存器中位 BOFF 改变</li> <li>2) CAN 节点状态寄存器中位 EWRN 改变</li> <li>3) 列表长度错误，该错误也置位 CAN 节点状态寄存器中的位 LLE</li> <li>4) 列表对象错误，该错误也置位 CAN 节点状态寄存器中的位 LOE</li> <li>5) MultiCAN 已经置位 INIT</li> </ol> 在 CAN 节点中断指针寄存器中的位域 ALINP 选择的中断输出线上产生该类型的中断请求。
<b>CANDIS</b>	4	rw	<b>禁止 CAN 节点</b> 置位该位将禁止该 CAN 节点。首先 CAN 节点一直等待，直到总线空闲或总线关闭状态。然后 INIT 被自动置位，如果位 ALIE 被置位，那么产生一个警报中断。
<b>CCE</b>	6	rw	<b>配置改变使能</b> <b>0<sub>B</sub></b> 位时序寄存器，端口控制寄存器和错误计数器寄存器只能被读取，忽略所有试图修改这些寄存器的操作。 <b>1<sub>B</sub></b> 位时序寄存器，端口控制寄存器和错误计数器寄存器可读也可写。

符号	位序号	类型	功能描述
<b>CALM</b>	7	rw	<b>CAN 分析模式</b> 如果该位被置位，那么 CAN 节点工作在分析模式。这就意味着可以接收报文，但不能发送报文。帧接收操作之后不向 CAN 总线发送应答。有效错误标志以隐性而不是显性发送。发送线连续保持为隐性(1)电平。 只有当位 INIT 置位时，才能对位 CALM 写入。
<b>SUSEN</b>	8	rw	<b>挂起模式</b> 该位可设置 CAN 节点通过 OCDS（片上调试支持）进入到挂起模式。 <b>0<sub>B</sub></b> CAN 节点忽略 OCDS 挂起触发 <b>1<sub>B</sub></b> OCDS 挂起触发禁止 CAN 模块：只要 CAN 节点变为总线空闲或总线关闭，位 INIT 被内部强制为 1，CAN 节点被禁止。而位 INIT 的实际值保持不变。 通过调试复位来复位 SUSEN。
<b>0</b>	5, [15:9]	r	<b>保留</b> 读操作返回 0，应写入 0。

### 22.3.3.2 节点状态寄存器

节点状态寄存器报告错误信息以及 CAN 帧传送是否成功。

NSRx (x = 0-5)

节点 x 状态寄存器

(204<sub>H</sub>+x\*100<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0					SUS ACK	LOE	LLE	BOF F	EWR N	ALE RT	RXO K	TXO K	LEC		
r					rh	rwh	rwh	rh	rh	rwh	rwh	rwh	rwh		

符号	位序号	类型	功能描述
LEC	[2:0]	rwh	<b>最近错误码</b> 该位域编码的详细信息见 <a href="#">表 22-10</a> 。
TXOK	3	rwh	<b>报文发送成功</b> 0 <sub>B</sub> 从最近一次该标志被复位开始, 无成功的发送操作。 1 <sub>B</sub> 已经成功发送了一个报文 (无错误并且得到至少另外一个节点应答)。 TXOK 必须由软件复位 (写 0), 写 1 无任何影响。
RXOK	4	rwh	<b>报文接收成功</b> 0 <sub>B</sub> 从最近一次该标志被复位开始, 无成功的接收操作 1 <sub>B</sub> 已经成功接收了一个报文 RXOK 必须由软件复位 (写 0), 写 1 无任何影响。
ALERT	5	rwh	<b>警报/警告</b> 出现下列事件之一将置位 ALERT (如果 ALIE 被置位, 这些事件也会触发报警中断): 1) CAN 节点状态寄存器中的位 BOFF 改变 2) CAN 节点状态寄存器中的位 EWRN 改变 3) 列表长度错误, 该错误也置位 CAN 节点状态寄存器中的位 LLE

**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
			4) 列表对象错误, 该错误也置位 CAN 节点状态寄存器中的位 LOE  5) MultiCAN 模块已经置位 INIT ALERT 必须由软件复位 (写 0), 写 1 无影响。
<b>EWRN</b>	6	rh	<b>错误警告状态</b> 0 <sub>B</sub> 没有超过警告界限。 1 <sub>B</sub> 错误计数器 REC 或 TEC 达到警告界限 EWRNLVL。
<b>BOFF</b>	7	rh	<b>总线关闭状态</b> 0 <sub>B</sub> CAN 控制器不处于总线关闭状态。 1 <sub>B</sub> CAN 控制器处于总线关闭状态。
<b>LLE</b>	8	rwh	<b>列表长度错误</b> 0 <sub>B</sub> 从最近一次该标志被清零开始, 没有出现列表长度错误。 1 <sub>B</sub> 报文验收滤波过程中, 检测到列表长度错误, 属于这个 CAN 节点的列表中的元素个数和列表终止指针给出的列表大小 (SIZE) 不同。 LLE 必须由软件复位 (写 0), 写 1 无任何影响。
<b>LOE</b>	9	rwh	<b>列表对象错误</b> 0 <sub>B</sub> 从最近一次该标志被清零开始, 没有出现列表对象错误。 1 <sub>B</sub> 报文验收滤波过程中, 检测到一个列表 对象错误。已经检测到带有错误列表指针 (报文对象控制寄存器中的) 的报文对象。 LOE 必须由软件复位 (写 0), 写 1 无任何影响。
<b>SUSACK</b>	10	rh	<b>挂起应答</b> 0 <sub>B</sub> CAN 节点不处于挂起模式或挂起请求未被处理, 但是 CAN 节点还未达到总线空闲或总线关闭状态。 1 <sub>B</sub> CAN 节点处于挂起模式: OCDS 挂起请求使 CAN 节点无效 (位 NCR.INIT 内部强制置 1)
<b>0</b>	[15:11]	r	<b>保留</b> ; 读操作返回 0, 应写入 0。



## LEC 位域编码

表 22-10 LEC 位域编码

LCE 值	含义
000 <sub>B</sub>	<u>无错误:</u> 在 CAN 总线上的最近一个报文中没有检测到错误。
001 <sub>B</sub>	<u>填充错误:</u> 在接收到报文中不允许出现连续 5 个相等位的部分, 出现了多于 5 个相等的位。
010 <sub>B</sub>	<u>格式错误:</u> 接收到的报文帧“固定格式部分”出现格式错误。
011 <sub>B</sub>	<u>应答错误:</u> 发送出去的报文未被另一个 CAN 节点应答。
100 <sub>B</sub>	<u>位 1 错误:</u> 在报文发送过程中, CAN 节点试着在仲裁域和应答间隙之外发送隐性电平 (1), 但是监测到的总线值为显性。
101 <sub>B</sub>	<u>位 0 错误:</u> 该编码指示两种不同情况: a) 在报文发送过程中 (或者应答位, 有效错误标志, 过载标志), CAN 节点试着发送显性电平 (0), 但监测到的总线值是隐性。 b) 总线关闭恢复期间, 每次监测到 11 个连续的隐性位, 将置位该编码。CPU 可以用这个编码指示总线未被连续扰乱。
110 <sub>B</sub>	<u>CRC 错误:</u> 收到的报文 CRC 校验和不正确。
111 <sub>B</sub>	<u>CPU 向 LEC 写:</u> 无论何时 CPU 向 LEC 写 111 <sub>B</sub> , LEC 被设置为 111 <sub>B</sub> ; 无论何时 CPU 向 LEC 写其它值, 实际写入值被忽略。

### 22.3.3.3 节点中断指针寄存器

节点中断指针寄存器将 CAN 节点的每个中断请求源与一条中断输出线（共有 16 条）相连。

#### NIPRx (x=0-5)

节点 x 中断指针寄存器

(208<sub>H</sub>+x\*100<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CFCINP</b>				<b>TRINP</b>				<b>LECINP</b>				<b>ALINP</b>			
rw				rw				rw				rw			

符号	位序号	类型	功能描述
<b>ALINP</b>	[3:0]	rw	<b>警报中断节点指针</b> 如果由 ALIE = 1 使能，由 ALINP 选择用来报告“警报中断请求”的中断输出线 INT_Om (m = 0-15) 编号。
<b>LECINP</b>	[7:4]	rw	<b>最近错误码中断节点指针</b> 如果由 LECIE = 1 使能，由 LECINP 选择用来报告“最近错误中断请求”的中断输出线 INT_Om (m = 0-15) 编号。
<b>TRINP</b>	[11:8]	rw	<b>传送完成中断节点指针</b> 如果由 TRIE = 1 使能，由 TRINP 选择用来报告“传送中断请求”的中断输出线 INT_Om (m = 0-15) 编号。
<b>CFCINP</b>	[15:12]	rw	<b>帧计数器中断节点指针</b> 如果由 CFCIE = 1 使能，由 CFCINP 选择用来报告“帧计数器溢出中断请求”的中断输出线 INT_Om (m = 0-15) 编号。

### 22.3.3.4 节点端口控制寄存器

节点端口控制寄存器用于配置 CAN 总线发送/接收端口。只有 NCRx.CCE 置位时，才能向 NPCRx 写入。

#### NPCRx (x = 0-5)

##### 节点 x 端口控制寄存器

(20C<sub>H</sub>+x\*100<sub>H</sub>)

复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
<b>RXSEL</b>	[2:0]	rw	<b>接收选择</b> RXSEL 从 8 条可能的接收输入线中选择一个。只能通过所选择的输入执行 CAN 通信。其它输入被忽略。 参见“接收输入选择”章节。
<b>LBM</b>	8	rw	<b>回环模式</b> 0 <sub>B</sub> 回环模式被禁止 1 <sub>B</sub> 使能回环模式。该节点和一个内部（虚拟）回环 CAN 总线相连。所有处于回环模式的 CAN 节点都和该虚拟 CAN 总线相连，因此，它们之间可进行内部通信。在回环模式下，外部发送线被强制为隐性电平。
<b>0</b>	[7:3], [15:9]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 22.3.3.5 节点位时序寄存器

节点位时序寄存器包含设置 CAN 传送位时序所需的全部参数。仅当置位 NCRx.CCE 时，才可写入 NBTRx。

#### NBTRxH (x=0-5)

节点 x 位时序寄存器高位 (212<sub>H</sub>+x\*100<sub>H</sub>) 复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
FTX	0	rw	快速发送 (仅适用于 TTC) CAN 总线上出现报文传送请求时，在新的位时间开始时发送帧起始 (SOF) 信号。 如果 CAN 总线处于空闲状态，且位 FTX 置位 (FTX = 1)，那么新报文的发送操作立即触发新的位时间。该操作消除了报文的发送触发和发送输出线上实际 SOF 信号之间的可变延迟。当发送触发出现在 CAN 位时间的不同位置时，则出现上述可变延迟。
0	[15:1]	r	保留； 读操作返回 0；应写入 0。

#### NBTRxL (x=0-5)

节点 x 位时序寄存器低位 (210<sub>H</sub>+x\*100<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIV8	TSEG2		TSEG1			SJW		BRP							
rw	rw		rw			rw		rw							

符号	位序号	类型	功能描述
BRP	[5:0]	rw	波特率预分频 如果 DIV8 = 0，一个时间单元等于 (BRP+1) 个

**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
			<p>时钟周期。</p> <p>如果 <math>DIV8 = 1</math>，一个时间单元等于 <math>8 \times (BRP+1)</math> 个时钟周期。</p>
<b>SJW</b>	[7:6]	rw	<p><b>(重新) 同步跳跃宽度</b></p> <p>可用于重新同步的时间为 <math>(SJW+1)</math> 个时间单元</p>
<b>TSEG1</b>	[11:8]	rw	<p><b>采样点前的时间段</b></p> <p>由用户定义的同步段结束和采样点之间的额定时间为 <math>(TSEG1+1)</math> 个时间单元。它包括传播段（将信号传播延迟考虑在内）。重新同步操作可将该时间段拉长。</p> <p>TSEG1 的有效值在 2 到 15 之间。</p>
<b>TSEG2</b>	[14:12]	rw	<p><b>采样点后的时间段</b></p> <p>由用户定义的采样点和下一个同步段开始之间的额定时间为 <math>(TSEG2+1)</math> 个时间单元。重新同步操作可将该时间段缩短。</p> <p>TSEG2 的有效值在 1 到 7 之间。</p>
<b>DIV8</b>	15	rw	<p><b>预分频时钟 8 分频</b></p> <p>0<sub>B</sub> 一个时间单元持续 <math>(BRP+1)</math> 个时钟周期</p> <p>1<sub>B</sub> 一个时间单元持续 <math>8 \times (BRP+1)</math> 个时钟周期</p>

### 22.3.3.6 节点错误计数器寄存器

#### NECNTxH (x = 0-5)

节点 x 错误计数器寄存器高位 (216<sub>H</sub>+x\*100<sub>H</sub>) 复位值: 0060<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						LEIN C	LET D	EWRNLVL							
r						rh	rh	rw							

符号	位序号	类型	功能描述
EWRNLVL	[7:0]	rw	错误警告级别 位域 EWRNLVL 定义用于设置相应错误警告位 EWRN 的门限值 (错误警告级别, 默认值 96)。
LETD	8	rh	最近错误传送方向 0 <sub>B</sub> CAN 接收时出现最近错误 (REC 值已增加) 1 <sub>B</sub> CAN 发送时出现最近错误 (TEC 值已增加)
LEINC	9	rh	最近错误引起错误计数器的增加量 0 <sub>B</sub> 最近错误使错误计数器加 1 1 <sub>B</sub> 最近错误使错误计数器加 8
0	[15:10]	r	保留 读操作返回 0, 应写入 0。

#### NECNTxL (x = 0-5)

节点 x 错误计数器寄存器低位 (214<sub>H</sub>+x\*100<sub>H</sub>) 复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
REC	[7:0]	rwh	接收错误计数器 位域 REC 包含 CAN 节点的接收错误计数器值

符号	位序号	类型	功能描述
<b>TEC</b>	[15:8]	rwh	<b>发送错误计数器</b> 位域 TEC 包含 CAN 节点的发送错误计数器值

### 22.3.3.7 节点帧计数器寄存器

节点帧计数器寄存器包含帧计数器的实际值及帧计数器的控制和状态位。

#### NFCRxH (x=0-5)

节点 x 帧计数器寄存器高位                      **(21A<sub>H</sub>+x\*100<sub>H</sub>)**                      复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0								CFC OV	CFCI E	0	CFMOD		CFSEL		
r								rwh	rw	r	rw		rw		

符号	位序号	类型	功能描述
<b>CFSEL</b>	[2:0]	rw	<b>CAN 帧计数选择</b> 该位域为选定的帧计数模式选择帧计数器的功能。 <b>帧计数模式</b> 位 0    如果 CFSEL 位 0 被置位, 那么每次 CAN 总线上接收到一个外来帧 (也就是说, 该帧和报文对象不匹配) 之后, CFC 增加。 位 1    如果 CFSEL 位 1 被置位, 那么每次 CAN 总线上接收到和报文对象匹配的报文帧之后, CFC 增加。 位 2    如果 CFSEL 位 2 被置位, 那么 CAN 节点成功发送一帧之后, CFC 增加。 <b>时间戳模式</b> 新的位时间开始时, 帧计数器 (内部的) 增加。当总线空闲时, 其值被永久采样到 CFC 位域。将新帧的 SOF 位到来之前的采样值写入到相应报文对象中。报文对象的处理结束时, 采样操作仍继续。 <b>位时序模式</b> 可用的位时序测量模式见表 22-11。如果 CFCIE 置位, 那么 CFC 更新操作将在节点 x (其中 x 为

**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
			CAN 节点编号) 上产生中断请求。
<b>CFMOD</b>	[4:3]	rw	<b>CAN 帧计数器模式</b> 该位域定义帧计数器工作模式 00 <sub>B</sub> 帧计数模式: 接收和发送报文帧时, 帧计数器增加 01 <sub>B</sub> 时间戳模式: 帧计数器用于计数 CAN 位时间 10 <sub>B</sub> 位时序模式: 帧计数器用于位时序分析 <sup>1)</sup> 11 <sub>B</sub> 保留
<b>CFCIE</b>	6	rw	<b>CAN 帧计数中断使能</b> 0 <sub>B</sub> 禁止 CAN 帧计数器溢出中断 1 <sub>B</sub> 使能 CAN 帧计数器溢出中断
<b>CFCOV</b>	7	rwh	<b>CAN 帧计数器溢出标志</b> 帧计数器溢出 (从 FFFF <sub>H</sub> 计数到 0000 <sub>H</sub> ) 时标志 CFCOV 被置位。在位时序分析模式, 更新 CFC 将使得 CFCOV 置位。如果 CFCIE = 1 将产生一个中断请求。 0 <sub>B</sub> 从最近一次该标志被清零开始, 未出现溢出 1 <sub>B</sub> 从最近一次该标志被清零开始, 已出现溢出 CFCOV 必须由软件清除。
<b>0</b>	5, [15:8]	r	<b>保留;</b> 读操作返回 0, 应写入 0。

1) 对于所有的位时序分析模式, NFCRx.CFC 的计数值始终显示为测量值减 1。例如: 在 CFSEL = 000 的模式下, CFC 值为 34, 指示在接收输入的最近两个显性沿之间的时间为 35 个 f<sub>CAN</sub> 时钟周期。



**NFCRxL (x=0-5)**

节点 **x** 帧计数器寄存器低位 (218<sub>H</sub>+x\*100<sub>H</sub>) 复位值: 0000<sub>H</sub>

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

符号	位序号	类型	功能描述
<b>CFC</b>	[15:0]	rwh	<b>CAN 帧计数器</b> 在帧计数模式，该位域包含帧计数器的值。 在时间戳模式下，该位域包含捕获的位时间计数 值，捕获新一帧的开始；

**位时序分析模式和状态**

**表 22-11 位时序分析模式 (CFMOD = 10)**

CFSEL	测量
000	无论何时在接收输入端上监测到一个显性沿（从 1 跳变到 0），该显性沿和最近的显性沿之间的时间被（以时钟周期为单位）保存到 CFC 中。
001	无论何时在接收输入端上监测到一个隐形沿（从 0 跳变到 1）在该隐性沿和最新的显性沿之间的时间（以时钟周期为单位）被保存到 CFC 中。
010	无论何时接收到作为发送显性沿结果的一个显性沿，两个沿之间的时间（时钟周期数）被保存到 CFC 中。
011	无论何时接收到作为发送隐性沿结果的一个隐性沿，两个沿之间的时间（时钟周期数）被保存到 CFC 中。
100	无论何时在接收输入端上监测到用于同步的显性沿，该沿和最近的采样点之间的时间（以时钟周期为单位）被保存到 CFC 中。
101	对每个采样点来说，新位时间开始和前一个位时间开始之间的时间（以时钟周期为单位）保存在 CFC[11:0]中，附加信息写入到 CFC[15:12]中： CFC[15]: 实际位时间的发送值

CFSEL	测量
	CFC[14]: 实际位时间的接收采样值 CFC[13:12]: CAN 总线信息 (见 <a href="#">表 22-12</a> )
110	保留
111	保留

表 22-12 CAN 总线状态信息

CFC[13:12]	CAN 总线状态
00	<b>没有位</b> CAN 总线空闲, 执行位填充 (解填充) 或者处于下面的帧段: SOF、SRR、CRC、分隔符、EOF 最先 6 位以及 IFS。
01	<b>新位</b> 该编码代表新帧段的首位。 当前位是下列帧段的首位: 标准 ID (仅发送) 的位 10 (MSB)、RTR、保留位、IDE、DLC (MSB)、每个数据字节的位 7 (MSB) 以及 ID 扩展的首位。
10	<b>位</b> 该编码代表帧段内长度大于 1 的位 (并不是新位中表示的那些段的首位)。 处理的当前位在下列帧段内: ID 位 (除了用于发送的标准 ID 的首位和扩展 ID 的首位), DLC (3 LSB) 和每个数据字节的位 6-0。
11	<b>完成</b> 当前位是下列帧段内的一位: 应答时隙、EOF 的最后位、激活/认可错误帧、过载帧。 连续的 2 个或更多完成码表示一个错误帧。

## 22.3.4 报文对象寄存器

### 22.3.4.1 报文对象控制寄存器

报文对象控制寄存器包含用于 CAN 传送的控制位和报文对象连接指针。每个控制位都有一个对应位 (CTRL 位域中的)。可通过向 CTRL 中的对应位写 1 的方式置位控制位。可通过直接向控制位写 0 的方式清除控制位。其它任何组合不会改变控制位。复位初始化之后, 指针 PNEXT (MOCTRnH[15:8]的读取值) 指向报文对象 n+1 (PNEXT = n+1), 报文对象 255 的 PNEXT 例外, 该报文对象是初始列表的最后一个报文对象 (PNEXT = 255)。指针 PPREV (MOCTRnH[7:0]的读取值) 最初指向报文对象 n-1 (PPREV = n-1), 报文对象 0 的 PPREV 例外, 该报文对象指示初始列表的开头 (PPREV = 0)。复位初始化意味着最初所有报文对象都属于未被分配的元素列表。

#### MOCTR0H

报文对象 0 控制寄存器高位 (101E<sub>H</sub>)

复位值: 0100<sub>H</sub>

#### MOCTR255H

报文对象 255 控制寄存器高位 (2FFE<sub>H</sub>)

复位值: FFFE<sub>H</sub>

#### MOCTRnH(n = 1-254)

报文对象 n 控制寄存器高位 (101E<sub>H</sub>+n\*20<sub>H</sub>)  
((n+1)\*0100<sub>H</sub>+((n-1)\*0001<sub>H</sub>)

复位值:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				SET DIR	SET TXE N1	SET TXE N0	SET TXR Q	SET RXE N	SET RTS EL	SET MSG VAL	SET MSG LST	SET NEW DAT	SET RXU PD	SET TXP ND	SET RXP ND
w				w	w	w	w	w	w	w	w	w	w	w	w

符号	位序号	类型	功能描述
SETRXPND	0	w	接收挂起置位 该位置位 RXPND
SETTXPND	1	w	发送挂起置位 该位置位 TXPND
SETRXUPD	2	w	接收更新置位 该位置位 RXUPD
SETNEWDAT	3	w	新数据置位 该位置位 NEWDAT

**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
<b>SETMSGLST</b>	4	w	报文丢失置位 该位置位 MSGLST
<b>SETMSGVAL</b>	5	w	报文有效置位 该位置位 MSGVAL
<b>SETRTSEL</b>	6	w	接收/发送选择置位 该位置位 RTSEL
<b>SETRXEN</b>	7	w	接收使能置位 该位置位 RXEN
<b>SETTXRQ</b>	8	w	发送请求置位 该位置位 TXRQ
<b>SETTXEN0</b>	9	w	发送使能 0 置位 该位置位 TXEN0
<b>SETTXEN1</b>	10	w	发送使能 1 置位 该位置位 TXEN1
<b>SETDIR</b>	11	w	报文方向置位 该位置位 DIR
<b>0</b>	[15:12]	w	保留 应写入 0。

**MOCTRnL(n = 0-255)**

报文对象 n 控制寄存器低位 (101C<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>				<b>RES DIR</b>	<b>RES TXE N1</b>	<b>RES TXE N0</b>	<b>RES TXR Q</b>	<b>RES RXE N</b>	<b>RES RTS EL</b>	<b>RES MSG VAL</b>	<b>RES MSG LST</b>	<b>RES NEW DAT</b>	<b>RES RXU PD</b>	<b>RES TXP ND</b>	<b>RES RXP ND</b>
w				w	w	w	w	w	w	w	w	w	w	w	w

符号	位序号	类型	功能描述
<b>RESRXPND</b>	0	w	<b>接收挂起复位</b> 该位复位 RXPND
<b>RESTXPND</b>	1	w	<b>发送挂起复位</b> 该位复位 TXPND
<b>RESRXUPD</b>	2	w	<b>接收更新复位</b> 该位复位 RXUPD
<b>RESNEWDAT</b>	3	w	<b>新数据复位</b> 该位复位 NEWDAT
<b>RESMSGGLST</b>	4	w	<b>报文丢失复位</b> 该位复位 MSGGLST
<b>RESMSGVAL</b>	5	w	<b>报文有效复位</b> 该位复位 MSGVAL
<b>RESRTSEL</b>	6	w	<b>接收/发送选择复位</b> 该位复位 RTSEL
<b>RESRXEN</b>	7	w	<b>接收使能复位</b> 该位复位 RXEN
<b>RESTXRQ</b>	8	w	<b>发送请求复位/置位</b> 该位复位 TXRQ
<b>RESTXEN0</b>	9	w	<b>发送使能 0 复位</b> 该位复位 TXEN0
<b>RESTXEN1</b>	10	w	<b>发送使能 1 复位</b> 该位复位 TXEN1
<b>RESDIR</b>	11	w w	<b>报文方向复位</b> 该位复位 DIR
<b>0</b>	[15:12]	w	<b>保留</b> 应写入 0。

#### 22.3.4.2 报文对象状态寄存器

##### **MOSTAT0H**

报文对象 0 状态寄存器高位 (101E<sub>H</sub>)

复位值: 0100<sub>H</sub>

##### **MOSTAT255H**

报文对象 255 状态寄存器高位 (2FFE<sub>H</sub>)

复位值: FFFE<sub>H</sub>

##### **MOSTATnH(n = 1-254)**

报文对象 n 状态寄存器高位 (101E<sub>H</sub>+n\*20<sub>H</sub>)  
((n+1)\*0100<sub>H</sub>+((n-1)\*0001<sub>H</sub>)

复位值:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PNEXT</b>								<b>PPREV</b>							
rh								rh							

符号	位序号	类型	功能描述
<b>PPREV</b>	[15:8]	rh	指向前一个报文对象的指针 PPREV 保存报文列表结构中报文对象的前一个报文对象的编号。
<b>PNEXT</b>	[7:0]	rh	指向下一个报文对象的指针 PNEXT 保存报文列表结构中报文对象的下一个报文对象的编号。

##### **MOSTATnL (n = 0-255)**

报文对象 n 状态寄存器低位 (101C<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>LIST</b>				<b>DIR</b>	<b>TX EN1</b>	<b>TX EN0</b>	<b>TX RQ</b>	<b>RX EN</b>	<b>RT SEL</b>	<b>MSG VAL</b>	<b>MSG LST</b>	<b>NEW DAT</b>	<b>RX UPD</b>	<b>TX PND</b>	<b>RX PND</b>
rh				rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

符号	位序号	类型	功能描述
<b>RXPND</b>	0	rh	接收挂起 0 <sub>B</sub> 没有接收到 CAN 报文 1 <sub>B</sub> 报文对象 n 已经接收到报文, 直接接收或者通

**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
			<p>过网关复制动作间接接收。</p> <p>RXPND 不能被硬件清零，必须由软件复位。</p>
<b>TXPND</b>	1	rh	<p><b>发送挂起</b></p> <p>0<sub>B</sub> 没有发送 CAN 报文</p> <p>1<sub>B</sub> 报文对象 n 已经通过 CAN 总线成功发送 CAN 报文</p> <p>TXPND 不能被硬件复位，必须由软件复位。</p>
<b>RXUPD</b>	2	rh	<p><b>接收更新</b></p> <p>0<sub>B</sub> 未进行接收更新</p> <p>1<sub>B</sub> 正在更新报文对象的标识符，DLC 和数据</p>
<b>NEWDAT</b>	3	rh	<p><b>新数据</b></p> <p>0<sub>B</sub> 从最近一次该标志被复位开始，报文对象 n 未被更新</p> <p>1<sub>B</sub> 报文对象 n 已经被更新</p> <p>接收到的 CAN 帧保存到报文对象 n 之后，硬件置位 NEWDAT。当报文对象 n 开始进行 CAN 发送时，硬件清除 NEWDAT。在新的发送数据保存到报文对象 n 中之后，NEWDAT 必须由软件置位，防止当前进行的发送结束时 TXRQ 被自动清零。</p>
<b>MSGLST</b>	4	rh	<p><b>报文丢失</b></p> <p>0<sub>B</sub> 未出现报文丢失</p> <p>1<sub>B</sub> 由于在 NEWDAT 已经置位的情况下，对 NEWDAT 再次置位，导致 CAN 报文丢失</p>
<b>MSGVAL</b>	5	rh	<p><b>报文有效</b></p> <p>0<sub>B</sub> 报文对象 n 无效</p> <p>1<sub>B</sub> 报文对象 n 有效</p> <p>有效的报文对象才能参加 CAN 传送。</p>
<b>RTSEL</b>	6	rh	<p><b>接收/发送选择</b></p> <p>0<sub>B</sub> 报文对象 n 未被选中进行接收或发送操作</p> <p>1<sub>B</sub> 报文对象 n 被选中进行接收或发送操作</p> <p><b>帧接收：</b></p>

符号	位序号	类型	功能描述
			<p>当报文对象已经被确认，用来存储当前接收到的 CAN 帧时，由硬件置位 RTSEL。在将接收到的帧最终保存到该报文对象 n 之前，检查 RTSEL 的是否置位。因而，CPU 可以通过软件清除 RTSEL 的方法压制预定的将 CAN 帧发送至报文对象 n 的操作。</p> <p><b>帧发送：</b></p> <p>当已经确认该报文对象 n 为下一个发送的报文对象时，RTSEL 被硬件置位。在报文对象实际建立传送之前，检查 RTSEL 是否仍然置位，且位 NEWDAT 被清除。因帧发送成功而验证报文对象 n 的操作之前，也要检查 RTSEL 是否仍然置位。</p> <p>仅当报文对象 n 的上下文改变时，才需要检查 RTSEL，可以避免和正在进行的帧发送冲突。在所有其它情况下，可以忽略 RTSEL。RTSEL 对报文验收滤波没有影响。RTSEL 不能被硬件清除。</p>
<b>RXEN</b>	7	rh	<p><b>接收使能</b></p> <p>0<sub>B</sub> 禁止报文对象 n 接收帧</p> <p>1<sub>B</sub> 使能报文对象 n 接收帧</p> <p>仅在接收验收滤波时才需要评估 RXEN 的值</p>
<b>TXRQ</b>	8	rh	<p><b>发送请求</b></p> <p>0<sub>B</sub> 报文对象 n 未请求发送报文</p> <p>1<sub>B</sub> 报文对象 n 请求在 CAN 总线上发送报文</p> <p>仅在 TXRQ、TXEN0、TXEN1 和 MSGVAL 置位的情况下，发送请求才有效。匹配的远程帧被正确接收之后，硬件置位 TXRQ。如果报文对象 n 被成功发送且 NEWDAT 没有被软件再次置位，硬件复位 TXRQ。</p>
<b>TXEN0</b>	9	rh	<p><b>发送使能 0</b></p> <p>0<sub>B</sub> 禁止报文对象 n 进行帧发送</p> <p>1<sub>B</sub> 使能报文对象 n 进行帧发送</p> <p>只有位 TXEN0 和 TXEN1 都置位的情况下，报文对象 n 才能进行发送。</p> <p>用户可以清除 TXEN0，来禁止当前更新的报文的</p>



**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
			发送，或者禁止远程帧的自动回应。
<b>TXEN1</b>	10	rh	<b>发送使能 1</b> 0 <sub>B</sub> 禁止报文对象 n 进行帧发送 1 <sub>B</sub> 使能报文对象 n 进行帧发送 只有在 TXEN0 和 TXEN1 都置位的情况下，报文对象 n 才能发送。 MultiCAN 模块用 TXEN1 在发送 FIFO 中选择有效报文对象。
<b>DIR</b>	11	rh	<b>报文方向</b> 0 <sub>B</sub> 接收对象选择： TXRQ=1，安排发送带有报文对象 n 的标识符的远程帧。接收到的标识符匹配的数据帧，报文存储在报文对象 n 中。 1 <sub>B</sub> 发送对象选择： 如果 TXRQ=1，安排用报文对象 n 发送数据帧。接收到标识符匹配的远程帧时，置位 TXRQ。
<b>LIST</b>	[15:12]	rh	<b>列表分配</b> LIST 指示报文对象 n 所属的报文列表编号。当报文对象的列表分配被命令面板修改时，硬件更新 LIST。

### 22.3.4.3 报文对象中断指针寄存器

报文对象中断指针寄存器 MOIPR H/L 保存与报文中断相关的各种指针以及帧计数器值。

#### MOIPRnH (n = 0-255)

报文对象 n 中断指针寄存器高位 (100A<sub>H</sub>+n\*20<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>CFCVAL</b>															
rwh															

符号	位序号	类型	功能描述
<b>CFCVAL</b>	[15:0]	rwh	<b>CAN 帧计数器值</b> 报文保存到报文对象中，或者报文对象成功发送报文之后，CAN 节点帧计数器寄存器 (NFCR) 中的 CAN 帧计数器值 CFC 被复制到 CFCVAL 中。

#### MOIPRnL (n = 0-225)

报文对象 n 中断指针寄存器低位 (1008<sub>H</sub>+n\*20<sub>H</sub>) 复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>MPN</b>								<b>TXINP</b>				<b>RXINP</b>			
rw								rw				rw			

符号	位序号	类型	功能描述
<b>RXINP</b>	[3:0]	rw	<b>接收中断节点指针</b> 为接收中断选择中断输出线 INT_Om (m = 0-15)
<b>TXINP</b>	[7:4]	rw	<b>发送中断节点指针</b> 为发送中断选择中断输出线 INT_Om (m = 0-15)
<b>MPN</b>	[15:8]	rw	<b>报文挂起编号</b> 出现接收/发送中断时，该位域用来选择报文挂起寄存器中的挂起位位置。

#### 22.3.4.4 报文对象功能控制寄存器

报文对象功能控制寄存器高位/低位用于选择和配置报文对象的功能，还用于保存 CAN 数据长度码。

#### MOFCRnH (n = 0-255)

##### 报文对象功能控制寄存器高位

(1002<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OMSC				DLC				STT	SdT	RMM	FRR EN	0	OVIE	TXIE	RXIE
rwh				rwh				rw	rw	rw	rw	r	rw	rw	rw

符号	位序号	类型	功能描述
RXIE	0	rw	<b>接收中断使能</b> 如果 RXIE 置位，则使能 CAN 报文接收操作产生报文中断请求，无论该报文是被直接接收还是通过网关动作接收的。 由 RXINP 为该中断选择中断输出线。
TXIE	1	rw	<b>发送中断使能</b> 如果 TXIE 置位，当报文对象通过 CAN 总线成功发送报文时，使能产生报文中断请求。 由 TXINP 为该中断选择中断输出线。
OVIE	2	rw	<b>溢出中断使能</b> 如果 OVIE = 1，那么当前报文对象指针 (CUR) 达到 FIFO/网关指针寄存器的 SEL 中的值时，产生 FIFO 已满中断。 如果报文对象 n 是接收 FIFO 基本报文对象，由 TXINP 为该中断选择中断输出线。 如果报文对象 n 作为发送 FIFO 基本报文对象，由 RXINP 为该中断选择中断输出线。 对于所有其它报文对象模式，OVIE 无影响。

**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
<b>FRREN</b>	4	rw	<p><b>外来远程请求使能</b></p> <p>指明在该报文对象或指针 CUR 指向的外来报文对象中，TXRQ 是否置位</p> <p>0<sub>B</sub> 报文对象接收到一个匹配的远程帧时，置位其 TXRQ</p> <p>1<sub>B</sub> 由指针 CUR 指定的报文对象接收到一个匹配的远程帧时，置位其 TXRQ</p>
<b>RMM</b>	5	rw	<p><b>发送对象远程监控</b></p> <p>0<sub>B</sub> 远程监控被禁止：报文对象的标识符，IDE 位，DLC 位在接收匹配远程帧时保持不变</p> <p>1<sub>B</sub> 远程监控使能：匹配远程帧的标识符，DLC 和 IDE 位被复制到发送报文对象中，以监控接收的远程帧</p> <p>位 RMM 仅适用于发送报文对象，对于接收报文对象无影响。</p>
<b>SDT</b>	6	rw	<p><b>单数据传送</b></p> <p>如果 SDT = 1 且该报文对象不是 FIFO 基本报文对象，那么该报文对象参加一次成功的数据传送（接收或者发送）之后，复位该报文对象的 MSGVAL。</p> <p>如果 SDT = 1 且该报文对象是 FIFO 基本报文对象，那么当指向当前报文对象 CUR 的指针达到 FIFO/网关指针寄存器 SEL 中的值时，复位 MSGVAL。</p> <p>当 SDT = 0 时，不影响 MSGVAL。</p>
<b>STT</b>	7	rw	<p><b>单次发送试验</b></p> <p>如果该位被置位，那么报文对象开始发送时，TXRQ 被清除。因而发送失败的情况下，不再次执行发送操作。</p>

控制器局域网络 (MultiCAN)

符号	位序号	类型	功能描述
<b>DLC</b>	[11:8]	rwh	<b>数据长度码</b> DLC 的有效值从 0 到 8。 DLC > 8 将使得数据长度为 8 个字节，但是 DLC 编码不会因为接收或发送 CAN 帧的操作而被截断。
<b>0</b>	[15:12]	r	<b>保留；</b> 读操作返回 0；应写入 0。
<b>0</b>	3	r	<b>保留；</b> 读操作返回 0；应写入 0。

MOFCRnL (n = 0-255)

报文对象功能控制寄存器低位

(1000<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>0</b>				<b>DAT C</b>	<b>DLC C</b>	<b>IDC</b>	<b>GDFS</b>	<b>0</b>				<b>MMC</b>			
r				rw	rw	rw	rw	r				rw			

符号	位序号	类型	功能描述
<b>MMC</b>	[3:0]	rw	<b>报文模式控制</b> 位域 MMC 控制报文对象的功能 0000 <sub>B</sub> 标准报文对象 0001 <sub>B</sub> 接收 FIFO 基本报文对象 0010 <sub>B</sub> 发送 FIFO 基本报文对象 0011 <sub>B</sub> 发送 FIFO 从属报文对象 0100 <sub>B</sub> 网关源报文对象 其它位组合 保留
<b>GDFS</b>	8	rw	<b>网关数据帧发送</b> 0 <sub>B</sub> 目标报文对象的 TXRQ 未被置位 1 <sub>B</sub> 数据帧从网关源对象传送到目标对象之后，网

**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
			关目标对象的 TXRQ 置位。 仅适用于网关源对象。
<b>IDC</b>	9	rw	<b>标识符复制</b> 如果 IDC = 1，则网关源对象的标识符（在接收帧被保存到源对象中之后）被复制到网关目标对象中。 仅适用于网关源对象。
<b>DLCC</b>	10	rw	<b>数据长度码复制</b> 如果 DLCC = 1，则网关源对象的数据长度码（在接收帧被保存到源对象中之后）被复制到网关目标对象中。 仅适用于在网关源对象。
<b>DATC</b>	11	rw	<b>数据复制</b> 如果 DATC = 1，则网关源对象（在接收帧被保存到源对象之后）的数据域（寄存器 MODATA0 和 MODATA4）被复制到网关目标对象。 仅适用于在网关源对象。
<b>0</b>	[7:4], [15:12]	r	<b>保留；</b> 读操作返回 0；应写入 0。

#### 22.3.4.5 报文对象 FIFO/网关指针寄存器

报文对象 FIFO/网关指针寄存器 H/L 包含一组用于 FIFO 和网关功能的报文对象连接指针。

#### MOFGPRnH (n = 0-255)

##### 报文对象 n FIFO/网关指针寄存器高位

(1006<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>SEL</b>								<b>CUR</b>							
rw								rwh							

**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
<b>CUR</b>	[7:0]	rwh	<b>当前报文对象指针</b> 当前报文对象指针指向 FIFO/网关结构中实际目标报文对象。  在 FIFO/网关操作之后，由列表结构中的下一个报文对象的报文编号（由报文控制寄存器的 PNEXT 给出）更新 CUR，直到到达 FIFO 的顶端的元素（由 TOP 给出），此时将它复位到 FIFO 底端的元素（由 BOT 给出）。
<b>SEL</b>	[15:8]	rw	<b>报文选择指针</b> 报文选择指针 SEL 为第二指针（软件），作为 FIFO 结构中硬件指针 CUR 的补充，SEL 仅用于监控目的。

**MOFGPRnL (n = 0-255)**

报文对象 n FIFO/网关指针寄存器低位

(1004<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>TOP</b>								<b>BOT</b>							
rw								rw							

符号	位序号	类型	功能描述
<b>BOT</b>	[7:0]	rw	<b>底端指针</b> 底端指针指向 FIFO 结构中的第一个元素。
<b>TOP</b>	[15:8]	rw	<b>顶端指针</b> 顶端指针指向 FIFO 结构中的最后一个元素。

*注：分配给同一个 CAN 节点的报文对象的指针必须置位。禁止指向不属于同一个 CAN 节点连接列表的报文对象。*

#### 22.3.4.6 报文对象验收屏蔽寄存器

寄存器 MOAMR H/L 包含用于报文对象验收滤波的屏蔽位。

##### MOAMRnH (n = 0-255)

报文对象 n 验收屏蔽寄存器高位

(100E<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 3FFF<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0	MID E	AM[28:16]													
r	rw	rw													

符号	位序号	类型	功能描述
AM[28:16]	[12:0]	rw	报文标识符验收屏蔽高位 描述见 MOARnL.AM[15:0]
MIDE	13	rw	报文 IDE 位的验收屏蔽位 0 <sub>B</sub> 该报文对象接收验收标准和扩展帧 1 <sub>B</sub> 该报文对象仅接收 IDE 位匹配的帧
0	[15:14]	r	保留; 读操作返回 0; 应写入 0。

##### MOAMRnL (n = 0-255)

报文对象 n 验收屏蔽寄存器低位

(100C<sub>H</sub>+n\*20<sub>H</sub>)

复位值: FFFF<sub>H</sub>

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

符号	位序号	类型	功能描述
AM[15:0]	[15:0]	rw	报文标识符验收屏蔽 可对带有标准标识符 (AM[28:18]) 或者扩展标识符 (AM[28:0]) 的接收报文进行验收滤波。对于标准标识符, “不考虑” 位 AM[17:0]。



### 22.3.4.7 报文对象仲裁寄存器

寄存器 MOAR H/L 包含报文对象的 CAN 标识符。

**MOARnH (n = 0-255)**

**报文对象 n 仲裁寄存器高位**

**(101A<sub>H</sub>+n\*20<sub>H</sub>)**

**复位值: 0000<sub>H</sub>**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
<b>PRI</b>	<b>IDE</b>	<b>ID[28:16]</b>													
rw	rwh	rwh													

符号	位序号	类型	功能描述
<b>ID[28:16]</b>	[12:0]	rwh	<b>报文对象的 CAN 标识符</b> 标准报文 (ID[28:18]) 或者扩展报文 (ID[28:0]) 的标识符。对于标准标识符, “不考虑” 位 ID[17:0]。
<b>IDE</b>	13	rwh	<b>报文对象的扩展标识符位</b> 0 <sub>B</sub> 带有 11 位标识符的标准帧 1 <sub>B</sub> 带有 29 位标识符的扩展帧
<b>PRI</b>	[15:14]	rw	<b>优先级分类</b> PRI 给报文对象分配 4 个优先级 0、1、2、3。PRI 值低的优先级较高。总是由 PRI 值较低的报文对象赢得接收和发送验收滤波。只有在报文对象优先级相同的情况下, 才执行基于标识符/屏蔽和列表次序的验收滤波。PRI 还决定发送验收滤波的方法: 00 <sub>B</sub> 保留。发送验收滤波时, 不考虑 PRI = 00 的发送对象。 01 <sub>B</sub> 根据列表次序进行发送验收。即只有列表中该报文对象之前的所有报文对象都没有有效发送请求 (MSGVAL & TXRQ & TXEN0 & TXEN1=1) 时, 才考虑发送该报文对象。 10 <sub>B</sub> 根据 CAN 标识符进行发送验收滤波。即仅在列表中没有具有更高优先级标识符 +IDE+DIR (根据 CAN 仲裁规则) 的其它报文对象时, 才考虑发送该报文对象。

**控制器局域网 (MultiCAN)**

符号	位序号	类型	功能描述
			11 <sub>B</sub> 根据列表次序进行发送验收滤波 (与 PRI = 01 <sub>B</sub> 相似)。

**MOARnL (n = 0-255)**

报文对象 n 仲裁寄存器低位

(1018<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ID[15:0]															
rwh															

符号	位序号	类型	功能描述
ID[15:0]	[15:0]	rwh	报文对象的 <b>CAN</b> 标识符低位 标准报文 (ID[28:18]) 或者扩展报文 (ID[28:0]) 的标识符。对于标准标识符, “不考虑” 位 ID[17:0]。

发送优先级

表 22-13 基于 CAN 仲裁规则的发送优先级

所选报文对象 A 和 B 设置 (A 的优先级高于 B)	注
A.MOAR[28:18] < B.MOAR[28:18] (A 的 11 位标准标识符小于 B 的 11 位标准标识符)	带较低标准标识符的报文具有较高优先级。 MOAR[28]为标准标识符的最高有效位 (MSB)。MOAR[18]为标准标识符的最低有 效位 (LSB)。
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = 0 (发送标准帧) B.MOAR.IDE = 1 (发送扩展帧)	具有相等的标准标识符时, 标准帧比扩展帧优 先级高
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = B.MOAR.IDE = 0 A.MOCTR.DIR = 1 (发送数据帧) B.MOCTR.DIR = 0 (发送远程帧)	具有相等的标识符时, 标准数据帧比标准远程 帧优先级高
A.MOAR[28:0] = B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 A.MOCTR.DIR = 1 (发送数据帧) B.MOCTR.DIR = 0 (发送远程帧)	具有相等的标识符时, 扩展数据帧比扩展远程 帧优先级高
A.MOAR[28:0] < B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 (29 位标识符)	带有较低标识符的扩展帧有较高的发送优先 级。MOAR[28]为整个标识符的最高有效位 (MSB) (MOAR[28:18]为标准标识符, MOAR[17:0]为标识符扩展部分)。MOAR[0]为 整个标识符的最低有效位 (LSB)。

#### 22.3.4.8 报文对象数据寄存器

##### MODATAnLH (n = 0-255)

报文对象 n 数据寄存器低位的高半部分

(1012<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB3								DB2							
rwh								rwh							

符号	位序号	类型	功能描述
DB2	[7:0]	rwh	CAN 数据字节 2
DB3	[15:8]	rwh	CAN 数据字节 3

##### MODATAnLL (n = 0-255)

报文对象 n 数据寄存器低位的低半部分

(1010<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB1								DB0							
rwh								rwh							

符号	位序号	类型	功能描述
DB0	[7:0]	rwh	CAN 数据字节 0
DB1	[15:8]	rwh	CAN 数据字节 1

寄存器 MODATAH H/L 包含报文对象 n 的高四字节数据。未使用的数据字节接收时设置为 0，发送时忽略。

### MODATAnHH (n = 0-255)

报文对象 n 数据寄存器高位的高半部分

(1016<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB7								DB6							
rwh								rwh							

符号	位序号	类型	功能描述
DB6	[7:0]	rwh	CAN 数据字节 6
DB7	[15:8]	rwh	CAN 数据字节 7

### MODATAnHL (n = 0-255)

报文对象 n 数据寄存器高位的低半部分

(1014<sub>H</sub>+n\*20<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DB5								DB4							
rwh								rwh							

符号	位序号	类型	功能描述
DB4	[7:0]	rwh	CAN 数据字节 4
DB5	[15:8]	rwh	CAN 数据字节 5

## 22.4 通用控制和状态

下面一节描述 MultiCAN 模块的时钟、调试和中断的相关内容。

### 22.4.1 时钟控制

MultiCAN 功能模块的 CAN 时钟频率  $f_{CAN}$  来自系统时钟  $f_{SYS}$  (=系统总线上的时钟)。模块内的分数分频器 FDIV 产生用于位时序计算的 CAN 时钟频率。对于所有 CAN 节点来说,该频率是相同的。任务调度器位于  $f_{SYS}$  域内。分数分频器控制位域 FDR.DM 可用于使能/禁止时钟的产生。

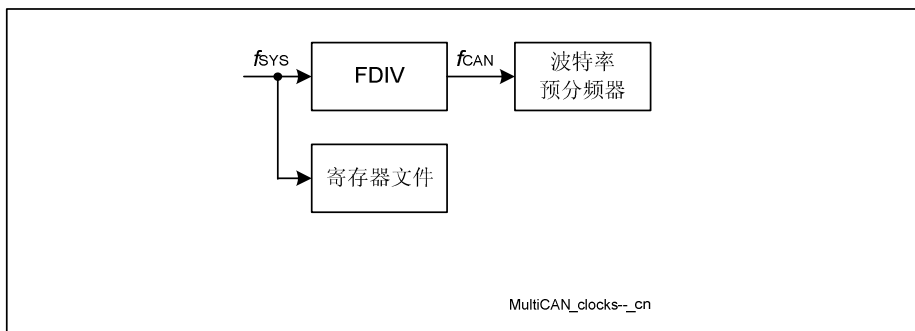


图 22-17 MultiCAN 模块时钟产生

分数分频器 FDIV 的输出  $f_{CAN}$  来自系统时钟  $f_{SYS}$ , 对其进行  $n$  分频得到。寄存器文件位于系统频率域。挂起信号 (用于回应 OCDS 挂起请求的、来自模块的应答信号) 冻结或者复位分数分频器。

*注: 接收输入线包含一个同步部分, 以确保输入数据稳定。该项措施与内部 CAN 状态机一起, 使得 CAN 输入和输出之间的响应时间最小 (至少为 3 个  $f_{SYS}$  时钟周期)。通常输入部分的切换延迟可被忽略, 而需要考虑端口输出驱动器的上升/下降时间 (可编程设置), 尤其对于高波特率的应用场合来说更是如此。*

当有效 CAN 节点要求的波特率为 1 Mbit/s 时 (必须考虑被激活的 CAN 节点的最高 CAN 波特率), 由下表给出 MultiCAN 模块操作 (验收滤波、报文对象处理等等) 的最小频率  $f_{SYS}$  (以 MHz 为单位)。如果需要的波特率更低, 则所需值线性缩小 (例如, 要求的波特率最大 500 kbit/s, 则所需值为表中值的 50%)。

下表假设 CPU (或 PEC) 以最大的处理能力访问 MultiCAN 模块。这些值可能进行了取整。

**表 22-14      最小操作频率[MHz]**

分配的报文对象 MO <sup>1)</sup> 个数	1 个 CAN 节点有效	2 个 CAN 节点有效	3 个 CAN 节点有效	4 个 CAN 节点有效	5 个 CAN 节点有效	6 个 CAN 节点有效
<b>16 MO</b>	12	19	26	33	40	47
<b>32 MO</b>	15	23	30	37	44	52
<b>64 MO</b>	21	28	37	46	53	61
<b>128 MO</b>	40	45	50	55	61	70
<b>144 MO</b>	42	47	52	57	62	70
<b>160 MO</b>	46	51	56	61	66	72
<b>176 MO</b>	50	55	60	66	71	76
<b>192 MO</b>	54	59	65	70	75	80
<b>208 MO</b>	58	64	69	74	79	84
<b>224 MO</b>	63	68	73	78	83	89
<b>240 MO</b>	67	72	77	82	88	93
<b>256 MO</b>	71	76	81	87	92	97

1) 仅需要考虑分配给某个 CAN 节点的报文对象。未被分配的报文对象对最小工作频率无影响。

MultiCAN 的波特率产生基于  $f_{SYS}$ ，需要谨慎选择该频率以得到正确的 CAN 位时序。需要的  $f_{SYS}$  值是 CAN 波特率的整数倍 (n) 乘以每个 CAN 位时间的时间单元个数。例如，每个位时间等于  $20 t_q$  的情况下，为了达到 1 Mbit/s，可能的  $f_{SYS}$  值由公式  $[n \times 20]$  MHz 给出，其中 n 为从 1 开始的整数值。

为了使抖动最小，在进行高波特率通信时建议用户不要使用分数分频器模式。

#### **22.4.2    端口输入控制**

可为每个 CAN 节点选择用作 RXDCAN 输入的引脚。所选引脚（几种可能的设置之一）与 CAN 节点相连。

### 22.4.3 挂起模式

可由 OCDS 触发挂起模式，冻结模块状态并访问寄存器（至少可进行读访问）。下面给出关于挂起模式的两种类型：

- 所有动作立即停止（“硬挂起模式”）：  
一旦挂起线变为有效，立刻关闭模块时钟。由 BPI 的快速关闭特性支持该模式。不支持对于 MultiCAN 模块的写动作，仅有组合的读动作才给出需要的数据（CAN RAM 和 CAN 寄存器不能被访问）

该模式下，所有后续模块动作被禁止，不能与其它器件进行通信的概率很高，CAN 总线被处于硬挂起模式的器件阻滞（例如，如果被挂起的 CAN 节点刚好送出一个显性电平）。退出挂起模式时，不太可能继续正常操作，必须进行复位操作。

- 完成当前动作（“软挂起模式”）：  
完成内部动作之后（例如发送 CAN 帧之后），模块功能被自动关闭（时钟仍然运行！）。因为此特性，通信网络不会因为某个通信参与方的挂起模式而被阻滞。另外，可读/写访问所有寄存器。结果是，调试器能够停止模块动作并修改寄存器。退出挂起模式之后，这些修改才生效。设计该模式是为了当系统复位仍在运行时，能够由 OCDS 修改寄存器或读取这些寄存器，且这些寄存器内容不被挂起模式破坏。

MultiCAN 模块的挂起机制的具体实现允许单独冻结 CAN 节点。为了支持软挂起模式，BPI 的快速关闭特性（硬挂起）一定不能由用户激活。为了支持系统所需的灵活性，可单独使能每个 CAN 节点的软挂起模式。

可使能/禁止整个 MultiCAN 模块的硬挂起特性，可单独使能/禁止每个 CAN 节点的软挂起特性。仅在所有 CAN 节点信号可被挂起时，分数分频器才能禁止 CAN 时钟。无效的 CAN 节点可始终被挂起。

### 22.4.4 中断结构

下图给出 MultiCAN 模块的通用中断结构。中断事件能够触发中断的产生。中断脉冲的产生和中断状态寄存器内的中断标志无关。通过写 0 的方式可软件复位中断标志。

如果由中断使能寄存器内相关的中断使能位使能，可在 MultiCAN 模块的 16 条中断输出线 INT\_Ox 之一上产生中断脉冲。如果多于一个中断源和同一个中断节点指针（在中断节点指针寄存器中）相连，则这些中断请求合并成一个公共请求线。



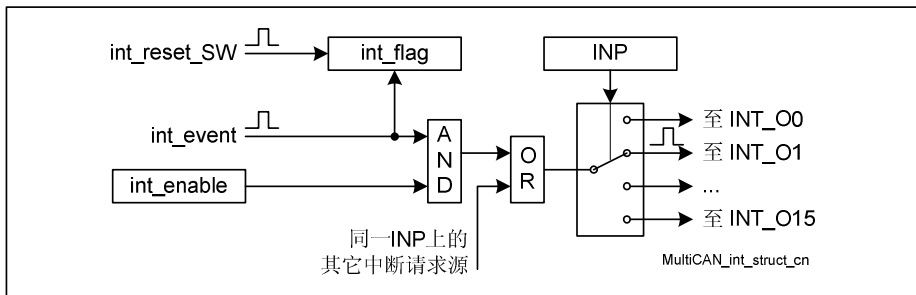


图 22-18 通用中断结构

## 22.5 MultiCAN 模块的实现

本节描述 CAN 模块的接口，包括时钟控制、端口连接、中断控制和地址译码等。

### 22.5.1 CAN 模块接口

图 22-19 给出 XE16xyM 系列中 MultiCAN 模块的具体实现和接口连接。MultiCAN 模块内核的 I/O 线（每个 CAN 节点 2 条）连接到表 22-18 所述的端口上。CAN 模块还与时钟控制、中断控制和地址译码逻辑相连。

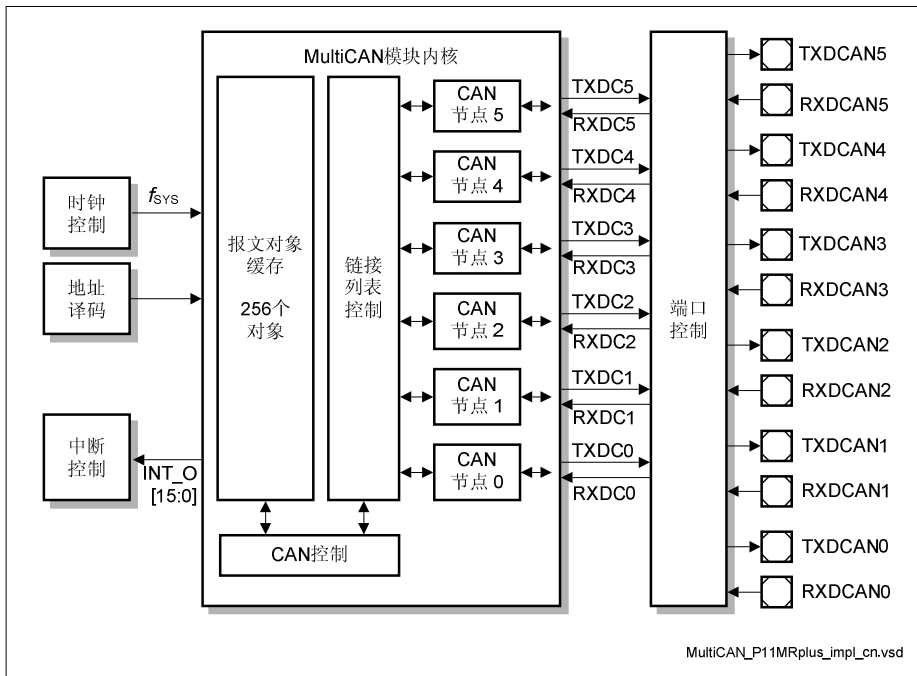


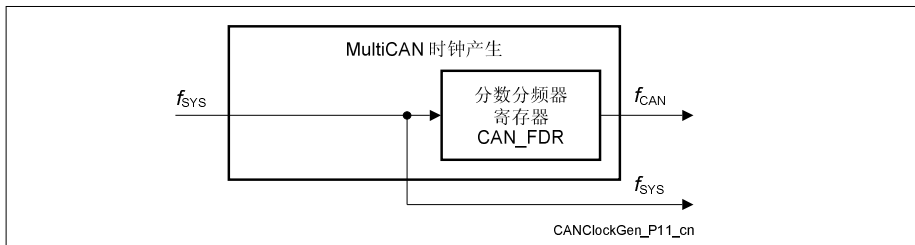
图 22-19 CAN 模块具体实现与互连

MultiCAN 中断控制寄存器  $x$  与 CAN 中断输出线  $INT\_Ox$  相连，其中  $x = 15-0$ 。另外，信号  $INT\_O15$  可被用于启动定时器。

### 22.5.2 模块时钟产生

如图 22-20 所示，由时钟产生单元产生并控制 MultiCAN 模块的时钟信号。时钟产生单元负责使能/禁止控制，时钟频率调整和调试时钟控制。

通过  $CAN\_FDR$  寄存器执行模块定时器时钟  $f_{CAN}$  的频率控制。



**图 22-20 MultiCAN 模块时钟产生**

模块控制时钟  $f_{SYS}$  为 MultiCAN 模块的内部时钟，作为控制用途，如控制逻辑和寄存器操作的时钟。 $f_{SYS}$  的频率与系统时钟频率  $f_{SYS}$  一致。

模块定时器时钟  $f_{CAN}$  为 MultiCAN 模块内部时钟，用作所有和时序相关的操作的输入时钟。

$f_{CAN}$  的频率由如下公式定义：

$$f_{CAN} = f_{SYS} \times \frac{1}{n}, \text{ 其中 } n = 1024 - \text{CAN\_FDR.STEP}$$

$$\text{或者 } f_{CAN} = f_{SYS} \times \frac{n}{1024} \quad \text{其中 } n = 0-1023$$

注：上面的两个公式中，第一个公式适用于正常分频模式 ( $\text{CAN\_FDR.DM} = 01_B$ )，第二个适用于分数分频模式 ( $\text{CAN\_FDR.DM} = 10_B$ )。

注：MultiCAN 分数分频器的输入信号  $ECEN$  被硬线连接至 0。

### 22.5.2.1 分数分频器概述

分数分频器使用一个可编程分频器对输入时钟分频产生输出时钟。分数分频器将输入时钟  $f_{IN}$  进行分频，分频因子为  $1/n$  或者  $n/1024$  ( $n=0-1023$  之间的任意值)，得到输出时钟信号  $f_{OUT}$ 。可由分数分频器寄存器控制位域  $FDR.DM$  使能/禁止时钟产生。

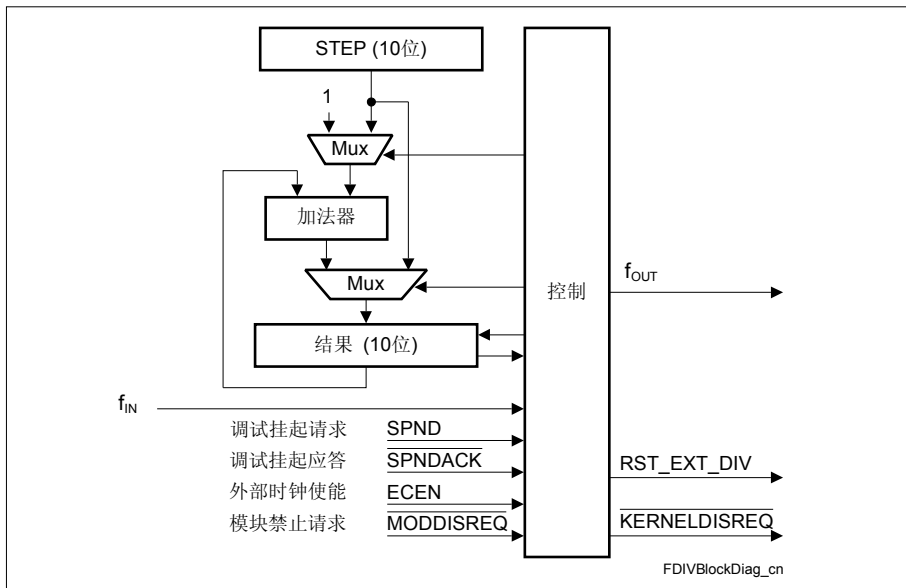


图 22-21 分数分频器框图

分数分频器内的时钟产生还由下面四个输入信号控制。

表 22-15 分数分频器 I/O 线

信号	I/O	描述
SPND	输入	挂起请求 输入由调试系统的挂起请求信号控制。当调试系统向片上模块发出一挂起请求时，该信号变为有效。
SPNDACK		挂起应答 由来自模块内核的禁止应答信号驱动的输入。分数分频器通过 $\overline{\text{KERNELDISREQ}} = 0$ 发出挂起请求之后，模块内核通过激活该信号回应挂起请求。
$\overline{\text{MODDISREQ}}$		模块禁止请求 输入与来自 CLC 逻辑的禁止请求输出相连。该输入上的有效信号激活输出信号 $\overline{\text{KERNELDISREQ}}$ 。
ECEN		外部时钟使能 信号 ECEN 能够用于将分数分频器的时钟产生与外部事件同步。
$\overline{\text{KERNELDISREQ}}$	输出	内核禁止请求 当 $\overline{\text{MODDISREQ}}$ 被激活或者当 SPND 变为有效时，该输出信号变为有效。
RST_EXT_DIV		复位外部分频器 该输出信号用于控制（停止/复位） $f_{\text{OUT}}$ 外部分频器电路。
$f_{\text{OUT}}$		模块时钟使能信号 $f_{\text{OUT}}$ 为模块时钟的使能信号。该模块时钟本身由 $f_{\text{OUT}}$ 使能信号和 $f_{\text{IN}}$ 相与得到。模块时钟频率参考主要指 $f_{\text{OUT}}$ 和 $f_{\text{IN}}$ 相与。

分数分频器有两种工作模式：

- 正常分频模式
- 分数分频模式

### 正常分频模式

正常分频模式下 (FDR.DM = 01<sub>B</sub>)，分数分频器用作重载计数器 (加 1)，从 3FF<sub>H</sub> 至 000<sub>H</sub> 的跳变在 f<sub>OUT</sub> 上产生一个输出时钟脉冲。由 FDR.RESULT 代表计数值，FDR.STEP 定义重载值。

正常分频模式的输出频率由以下公式定义：

$$f_{OUT} = f_{IN} \times \frac{1}{n}, \text{ 其中 } n = 1024 - \text{STEP}$$

为了使 f<sub>OUT</sub> = f<sub>IN</sub>，STEP 必须设置为 3FF<sub>H</sub>。图 22-22 给出正常分频模式下，重载值 FDR.STEP = 3FD<sub>H</sub> 时的分频操作示例。f<sub>OUT</sub> 的时钟频率代表 f<sub>OUT</sub> 使能信号和 f<sub>IN</sub> 相与的结果。

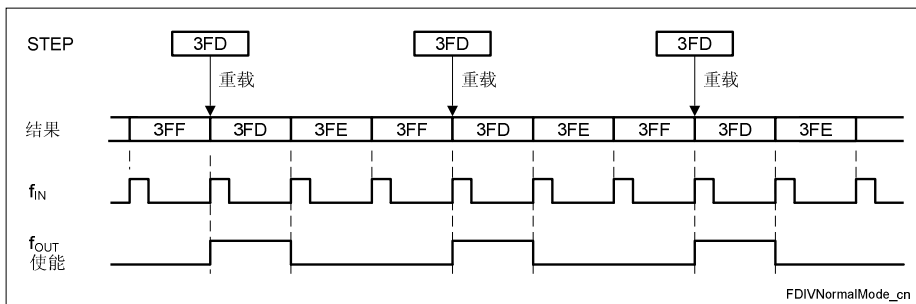


图 22-22 正常分频模式时序

## 分数分频模式

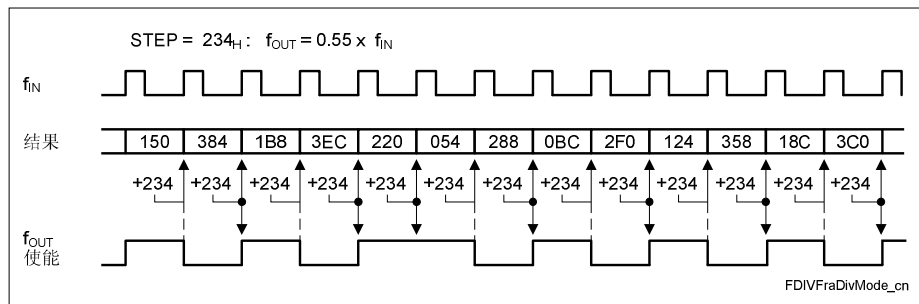
当选择分数分频模式 (FDR.DM = 10<sub>B</sub>) 时, 输入时钟  $f_{IN}$  经过  $n/1024$  分频得到 ( $n$  为从 0 到 1023 之间的任意值) 输出时钟  $f_{OUT}$ 。通常, 分数分频模式所允许设置的平均输出时钟频率比正常分频模式的精度更高。

分数分频模式下, 根据 FDR.RESULT + FDR.STEP 的结果在  $f_{OUT}$  上产生输出时钟脉冲。如果相加的结果超过 3FF<sub>H</sub> 而导致溢出, 则在  $f_{OUT}$  上产生一个脉冲。请注意分数分频模式下, 时钟  $f_{OUT}$  可有一个  $f_{IN}$  时钟周期的最大周期抖动。

分数分频模式下的输出频率由如下公式定义:

$$f_{OUT} = f_{IN} \times \frac{n}{1024} \quad \text{其中 } n = 0-1023$$

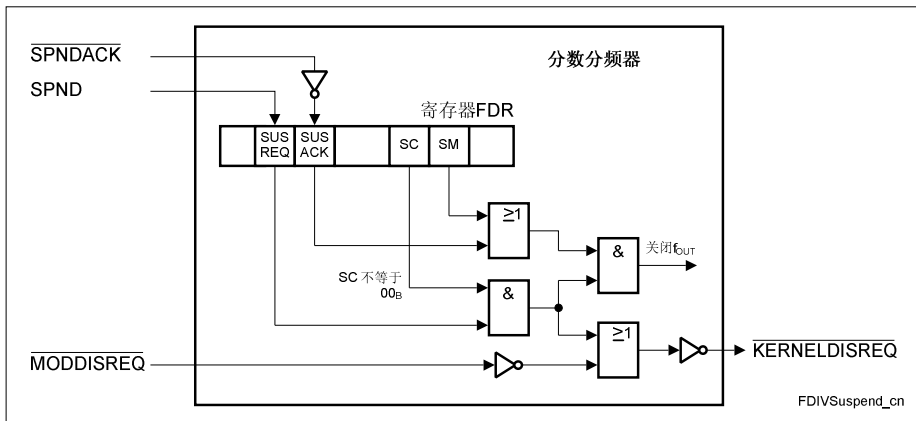
**图 22-23** 给出重载值为 FDR.STEP = 234<sub>H</sub> (= 分频因子 564/1024 = 0.55) 时的分数分频模式操作示例。 $f_{OUT}$  的时钟频率代表  $f_{IN}$  与  $f_{OUT}$  使能信号相与的结果。



**图 22-23 分数分频模式时序**

## 挂起模式控制

可根据输入挂起请求 (SPND) 控制分数分频器的操作。挂起模式下由片上调试控制逻辑激活该输入。挂起模式下, 模块寄存器可读/可写, 但是其它模块内部功能被冻结。由 SPND = 1 请求挂起模式。通过设置 SPNDACK 至 0 (授权挂起模式) 的方式应答挂起模式请求之后且 FDR.SC 不等于 00<sub>B</sub> (时钟输出信号被禁止), 一个  $f_{IN}$  时钟周期之后进入到挂起模式。通过设置位 SM = 1 且 FDR.SC 不等于 00<sub>B</sub> (立即挂起模式), 立即进入到挂起模式。信号 SPND 和  $\overline{SPNDACK}$  的状态被锁存到寄存器 FDR 的两个状态标志 SUSREQ 和 SUSACK 中, 且为了保持在挂起模式下, ( $\overline{SPNDACK}$  或位 SM) 必须保持置位。



**图 22-24 挂起模式配置**

当  $\overline{\text{MODDISREQ}}$  被激活时，内核禁止请求信号  $\overline{\text{KERNELDISREQ}}$  始终有效，与分数分频器逻辑的挂旗模式设置无关。

## 外部时钟使能

软件禁止模块时钟产生之后（设置  $\text{FDR.DISCLK} = 1$ ），通过输入  $\text{ECEN} = 1$ （受硬件控制）可退出禁止状态。当  $\text{FDR.ENHW} = 1$  时，该特性使能。MultiCAN 模块中，信号  $\text{ECEN}$  绑定为 0 电平。

## 寄存器概述

### 分数分频器寄存器

分数分频器包含两个寄存器， $\text{FDRL}$ （低 16 位）和  $\text{FDRH}$ （高 16 位）。

### FDRL

#### 分数分频器寄存器 L

(0C<sub>H</sub>)

复位值: 0000<sub>H</sub>

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



**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
<b>STEP</b>	[9:0]	rw	<b>步长值</b> 正常分频模式下，STEP 包含 RESULT 的重载值。 分数分频模式下，STEP 定义一个 10 位累加值，在每个输入时钟周期将该值累加到 RESULT 上。
<b>SM</b>	11	rw	<b>挂起模式</b> SM 选择授权或者立即挂起模式。 0 选择授权挂起模式 1 选择立即挂起模式
<b>SC</b>	[13:12]	rw	<b>挂起控制</b> SC 定义挂起模式下（位 SUSREQ 和 SUSACK 置位）分数分频器的操作。 00 继续产生时钟 01 停止产生时钟且不产生时钟输出信号。 RESULT 不改变，当向位域 DM 写 01 <sub>B</sub> 或 10 <sub>B</sub> 时的情况除外 10 停止产生时钟且不产生时钟输出信号。将 3FF <sub>H</sub> 加载到 RESULT。 11 与 SC = 10 <sub>B</sub> 情况相同，但是 RST_EXT_DIV 为 1（与位域 DM 无关）。
<b>DM</b>	[15:14]	rw	<b>分频模式</b> DM 选择分数分频模块的功能。 00 关闭分数分频器；不产生输出时钟。 RST_EXT_DIV 为 1。不更新 RESULT（复位之后的缺省设置）。 01 选择正常分频模式 10 选择分数分频模式 11 关闭分数分频器，不产生输出时钟，不更新 RESULT。
<b>0</b>	10	r	<b>保留</b> 读操作返回 0；应写入 0。

## FDRH

分数分频器寄存器 H

(0E<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
DIS CLK	EN HW	SUS REQ	SUS ACK	0	RESULT										
rwh	rw	rh	rh	r	rh										

符号	位序号	类型	功能描述
RESULT	[9:0]	rh	<b>结果值</b> 正常分频模式下, RESULT 用作重载计数器 (加 1)。 分数分频模式下, 该位域包含 RESULT+STEP 的结果。 如果 DM 被设置为 01 <sub>B</sub> 或 10 <sub>B</sub> , 3FF <sub>H</sub> 被加载到 RESULT 中。
SUSACK	12	rh	<b>挂起模式应答</b> 0 挂起模式未被应答 1 挂起模式被应答 当 SUSACK 和 SUSREQ 置位时, 进入到挂起模式。
SUSREQ	13	rh	<b>挂起模式请求</b> 0 未请求挂起模式 1 请求挂起模式 当 SUSREQ 和 SUSACK 置位时, 进入到挂起模式。
ENHW	14	rw	<b>使能硬件时钟控制</b> 0 位 DISCLK 不能被输入信号 ECEN 上的高电平硬件复位。 1 当输入信号 ECEN 为高电平时, 位 DISCLK 可被硬件复位。

**控制器局域网络 (MultiCAN)**

符号	位序号	类型	功能描述
<b>DISCLK</b>	15	rwh	<b>关闭时钟</b> 0 根据位域 DM 的设置，产生 f <sub>OUT</sub> 时钟 1 分数分频器停止。使能信号 f <sub>OUT</sub> 变为无效。不发生改变，写访问位域 DM 的情况除外。
<b>0</b>	其它	r	<b>保留</b> 读操作返回 0；应写入 0。

分数分频器工作模式

表 22-16 分数分频器功能表

模式	SC	DM	RES_EXT_DIV	RESULT	f <sub>OUT</sub>	分数分频器的操作
正常模式	-	00	1	不变	无效	关闭
		01	0	连续更新 <sup>1)</sup>	有效	正常分频器模式
		10				分数分频器模式
		11		不变	无效	关闭
挂起模式	00	00	1	不变	无效	关闭
		01	0	连续更新 <sup>1)</sup>	有效	正常分频器模式
		10				分数分频器模式
		11		不变	无效	关闭
	01	00	1	不变	无效	关闭
		01	0	不变 <sup>1)</sup>		暂停
		10				
		11		不变		关闭
	10	00	1	用 3FF <sub>H</sub> 加载	无效	关闭
		01	0			暂停
		10				
		11				关闭
	11	-	1	用 3FF <sub>H</sub> 加载	无效	关闭

1) 每次写操作 FDR，设置 DM = 01<sub>B</sub> 或 10<sub>B</sub> 时，都会将 RESULT 设置为 3FF<sub>H</sub>。

### 22.5.3 模式控制行为

MultiCAN 模块提供两种用于停止 CAN 通信的机制。

- 挂起模式：

由 OCDS 模块发出挂起模式请求。可单独使能/禁止每个 CAN 节点是否响应挂起请求。挂起模式下，CAN 节点正确完成正在运行的 CAN 帧之后，不启动新的 CAN 帧。

- 立即停止模式：

当器件模式控制请求停止模式时（由 CAN\_KSCCFG 配置），进入立即停止模式。如果请求立即停止模式，CAN 模块立即停止所有 CAN 操作（即使当时正在进行 CAN 帧传送）且将所有的发送输出设置为 1。为了允许进行 CAN 操作，位域 NOMCFG 必须被设置为运行模式。为了支持挂起模式（见上面的描述），位域 SUMCFG 必须被设置为运行模式以防止立即停止模式。

### 22.5.4 模式控制

对于系统控制任务，如省电或用于调试的挂起请求，模式控制的概念允许编程设置不同器件工作条件下的模块行为。对于每个工作模式，MultiCAN 内核行为为可编程设定，通过 SCU 的全局状态控制部分进行请求。MultiCAN 模块提供一个寄存器

**CAN\_KSCCFG**，用来定义下述器件工作模式下模块内核的行为：

- 正常操作：

该工作模式为缺省工作模式。当挂起请求或时钟关闭请求不被响应时，模块处于该模式。模块时钟不会被关闭且 MultiCAN 寄存器可读或写。内核行为由寄存器 KSCCFG.NOMCFG 定义。

- OCDS 挂起模式：

当挂起请求（由调试器发出的）有效时，请求该工作模式。不关闭模块时钟，MultiCAN 寄存器可读或写。由 KSCCFG.SUMCFG=00，且 OCDS 寄存器被正确配置。

- 时钟关闭模式：

为了降低功耗，可请求此工作模式。模块时钟被关闭。

对于 MultiCAN 模块来说，模式控制能够影响下面的内部操作：

- CAN 报文的当前发送：

如果有挂起请求，则可启动 CAN 报文的发送。必须由模式控制使能操作的启动。如果内核模式允许启动（运行模式 0 和 1），则执行发送操作。如果内核模式不允许启动（停止模式 0 和 1），则不启动发送操作。启动请求不是被取消，而是被冻结。如果内核模式再次变为运行模式，则被“冻结”的请求如编程设定的那样被启动。

MultiCAN 内核的行为可被设置为各种器件工作模式（正常操作、挂起模式、时钟关闭模式），如表 22-17 所示。

表 22-17 MultiCAN 内核行为

内核模式	内核行为	编码
运行模式 0	按照规定进行内核操作，对数据传送操作没有影响（运行模式 0 和运行模式 1 下，内核行为相同）	00 <sub>B</sub>
运行模式 1		01 <sub>B</sub>
停止模式 0	完成一些内部操作之后（可能需要数个时钟周期），模块停止。不完成挂起的 CAN 传送操作。器件将外部总线驱动为隐性电平。不可能对寄存器进行读/写访问。	10 <sub>B</sub>
停止模式 1		11 <sub>B</sub>

通常，位域 KSCCFG.NOMCFG 应当配置为运行模式 0，作为标准操作的缺省设置。如果 MultiCAN 内核不应当响应挂起请求（且如处于正常模式那样继续运行），位域 KSCCFG.SUMCFG 的配置必须和 KSCCFG.NOMCFG 的值相同。如果达到特定停止条件时，MultiCAN 内核应当显示出不同的行为和停止操作，停止模式 0 或停止模式 1 的编码必须写入到 KSCCFG.SUMCFG。

一种相似的机制同样适用于时钟关闭模式，也可由位域 KSCCFG.COMCFG 编程设置需要的模块内核行为。

*注：停止模式的选择在很大程度上取决于应用的需要。同一个应用不太可能同时需要不同的停止模式。因此，寄存器 KSCCFG 内的位域应当仅需要一种停止模式类型（停止模式 0 或 1）。停止模式 0 和停止模式 1 不应混合使用，从而避免 MultiCAN 模块从停止模式 0 转换到停止模式 1（反之亦然）。*

请注意当所有配置位域设置为运行模式 0 时，位 KSCCFG.MODEN 只应由软件设置。

## 22.5.5 模块控制寄存器描述

### 22.5.5.1 内核状态配置寄存器

对于不同的器件工作模式，通过内核状态配置寄存器 KSCCFG 选择期望的内核模式。

应用复位可复位位域 KSCCFG.NOMCFG 和位域 KSCCFG.COMCFG。位域 KSCCFG.SUMCFG 可由调试复位进行复位。

注：NOMCFG、SUMCFG 和 COMCFG 的编码描述见[表 22-17](#)。

### CAN\_KSCCFG

内核状态配置寄存器

SFR (FE1E<sub>H</sub>)

复位值: 0000<sub>H</sub>

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
BP COM	0	COMCFG	BP SUM	0	SUMCFG	BP NOM	0	NOMCFG	0	BP MOD EN	MOD EN				
w	r	rw	w	r	rw	w	r	rw	r	w	rw				

符号	位序号	读写类型	功能描述
<b>MODEN</b>	0	rw	<p><b>模块使能</b></p> <p>该位使能模块内核时钟和模块功能。</p> <p>0<sub>B</sub> 模块被立即关闭（无需考虑停止条件）。</p> <p>该模块不响应模块控制操作，模块时钟被关闭。模块不能响应读访问并且忽略写访问（KSCCFG 除外）。</p> <p>1<sub>B</sub> 模块开启，可以正常工作。</p> <p>向 MODEN 写 1 之后，访问其它 MultiCAN 寄存器之前，推荐读取寄存器 KSCCFG 以避免流水线对控制模块的影响。</p> <p>注：该位可被应用复位复位。</p>
<b>BPMODEN</b>	1	w	<p><b>MODEN 的位保护</b></p> <p>该位使能对 MODEN 的写操作。读该位始终返回 0。</p>

**控制器局域网络 (MultiCAN)**

符号	位序号	读写类型	功能描述
			0 <sub>B</sub> MODEN 不改变 1 <sub>B</sub> 由写入值更新 MODEN
<b>NOMCFG</b>	[5:4]	rw	<b>正常操作模式配置</b> 该位域定义适用于正常操作模式的内核模式。 0X <sub>B</sub> 模块被开启 1X <sub>B</sub> 模块被关闭 只有 CR=00 或 11 时，才考虑该位域。 <i>注：该位可被应用复位复位。</i>
<b>BPNO</b>	7	w	<b>NOMCFG 的位保护</b> 该位使能对 NOMCFG 的写操作。读该位始终返回 0。 0 <sub>B</sub> NOMCFG 不改变 1 <sub>B</sub> 由写入值更新 NOMCFG。
<b>SUMCFG</b>	[9:8]	rw	<b>挂起模式配置</b> 该位域定义适用于挂起模式的内核模式。 0X <sub>B</sub> 模块被开启 为了具有软挂起行为，推荐使用该设置。由 OCDS 模块挂起内核。 1X <sub>B</sub> 模块被关闭 只有 CR=01 时，才考虑该位域。 <i>注：该位可被调试复位复位。</i>
<b>BPSUM</b>	11	w	<b>SUMCFG 的位保护</b> 该位使能对 SUMCFG 的写操作。读该位始终返回 0。 0 <sub>B</sub> SUMCFG 不改变 1 <sub>B</sub> 由写入值更新 SUMCFG
<b>COMCFG</b>	[13:12]	rw	<b>时钟关闭模式配置</b> 该位域定义适用于时钟关闭模式的内核模式。 0X <sub>B</sub> 模块被开启



**控制器局域网络 (MultiCAN)**

符号	位序号	读写类型	功能描述
			<b>1X<sub>B</sub></b> 模块被关闭 只有 CR=10 时，才考虑该位域。 <i>注：该位可被应用复位复位。</i>
<b>BPCOM</b>	15	w	<b>COMCFG 的位保护</b> 该位使能对 COMCFG 的写操作。读该位始终返回 0。 <b>0<sub>B</sub></b> COMCFG 不改变 <b>1<sub>B</sub></b> 由写入值更新 COMCFG
<b>0</b>	[3:2], 6, 10, 14	r	<b>保留：</b> 读操作返回 0；应写入 0。

*注：位 BPxxx 的位保护允许使用单次写操作修改部分配置位（无需 CPU 使用读取-修改-回写机制）。*

### 22.5.6 外部信号的连接

下面的表格给出 MultiCAN 信号与 XE16xyM 器件内其它模块或引脚的数字信号连接。

为每个 CAN 节点选择的输入信号 (由位域 NPCRx.RXSEL 位域选择) 被表示为 CANxINS (CAN 节点 x 输入信号, 其中  $x = 5-0$ )。

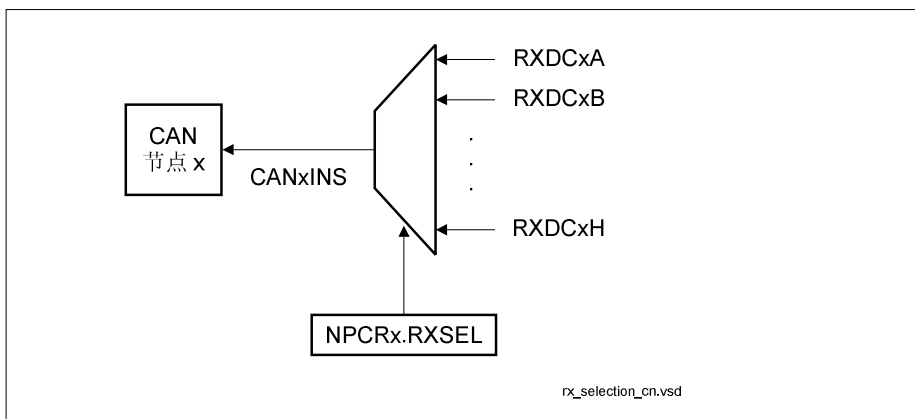


图 22-25 CAN 器件接收输入选择

**表 22-18 XE16xyM 内的 MultiCAN 的信号相互连接**

信号	来自/送至 哪个模块	I/O 至 CAN	可用作...
MultiCAN 节点 0 信号			
RXDC0A	P2.3	I	接收输入 A (NPCR0.RXSEL = 000 <sub>B</sub> )
RXDC0B	P0.3		接收输入 B (NPCR0.RXSEL = 001 <sub>B</sub> )
RXDC0C	P2.0		接收输入 C (NPCR0.RXSEL = 010 <sub>B</sub> )
RXDC0D	P2.6		接收输入 D (NPCR0.RXSEL = 011 <sub>B</sub> )
RXDC0E	ESR1		接收输入 E (NPCR0.RXSEL = 100 <sub>B</sub> )
RXDC0F	P11.0		接收输入 F (NPCR0.RXSEL = 101 <sub>B</sub> )
RXDC0G	1		接收输入 G(NPCR0.RXSEL = 110 <sub>B</sub> )
RXDC0H	0		接收输入 H (NPCR0.RXSEL = 111 <sub>B</sub> )
TXDC0	P0.1	O	发送输出
	P0.2		
	P2.1		
	P2.4		
	P2.5		
	P11.1		
MultiCAN 节点 1 信号			
RXDC1A	P2.4	I	接收输入 A (NPCR1.RXSEL = 000 <sub>B</sub> )
RXDC1B	P0.4		接收输入 B (NPCR1.RXSEL = 001 <sub>B</sub> )
RXDC1C	P2.7		接收输入 C (NPCR1.RXSEL = 010 <sub>B</sub> )
RXDC1D	CAN0INS		接收输入 D (NPCR1.RXSEL = 011 <sub>B</sub> )
RXDC1E	ESR2		接收输入 E (NPCR1.RXSEL = 100 <sub>B</sub> )
RCDC1F	P8.1		接收输入 F (NPCR1.RXSEL = 101 <sub>B</sub> )
RXDC1G	1		接收输入 G (NPCR1.RXSEL = 110 <sub>B</sub> )

**控制器局域网络 (MultiCAN)**

信号	来自/送至 哪个模块	I/O 至 CAN	可用作...
RXDC1H	0		接收输入 H (NPCR1.RXSEL = 111 <sub>B</sub> )
CAN1INS	U1C1_DX0F	O	
TXDC1	P0.6	O	发送输出
	P2.2		
	P2.9		
	P8.2		

**MultiCAN 节点 2 信号**

RXDC2A	P4.3	I	接收输入 A (NPCR2.RXSEL = 000 <sub>B</sub> )
RXDC2B	P10.11		接收输入 B (NPCR2.RXSEL = 001 <sub>B</sub> )
RXDC2C	CAN1INS		接收输入 C (NPCR2.RXSEL = 010 <sub>B</sub> )
RXDC2D	P2.13		接收输入 D (NPCR2.RXSEL = 011 <sub>B</sub> )
RXDC2E	P6.1		接收输入 E (NPCR2.RXSEL = 100 <sub>B</sub> )
RXDC2F	P5.15		接收输入 F (NPCR2.RXSEL = 101 <sub>B</sub> )
RXDC2G	1		接收输入 G (NPCR2.RXSEL = 110 <sub>B</sub> )
RXDC2H	0		接收输入 H (NPCR2.RXSEL = 111 <sub>B</sub> )
TXDC2	P4.1	O	发送输出
	P4.2		
	P10.12		
	P6.0		
	P2.12		

**MultiCAN 节点 3 信号**

RXDC3A	P3.3	I	接收输入 A (NPCR3.RXSEL = 000 <sub>B</sub> )
RXDC3B	P3.0		接收输入 B (NPCR3.RXSEL = 001 <sub>B</sub> )
RXDC3C	P10.14		接收输入 C (NPCR3.RXSEL = 010 <sub>B</sub> )

**控制器局域网络 (MultiCAN)**

信号	来自/送至 哪个模块	I/O 至 CAN	可用作...
RXDC3D	CAN2INS		接收输入 D (NPCR3.RXSEL = 011 <sub>B</sub> )
RXDC3E	P0.5		接收输入 E (NPCR3.RXSEL = 100 <sub>B</sub> )
RXDC3F	-		接收输入 F (NPCR3.RXSEL = 101 <sub>B</sub> )
RXDC3G	1		接收输入 G (NPCR3.RXSEL = 110 <sub>B</sub> )
RXDC3H	0		接收输入 H (NPCR3.RXSEL = 111 <sub>B</sub> )
TXDC3	P3.1	O	发送输出
	P3.2		
	P10.13		
	P0.7		

**MultiCAN 节点 4 信号**

RXDC4A	P3.4	I	接收输入 A (NPCR4.RXSEL = 000 <sub>B</sub> )
RXDC4B	P7.0		接收输入 B (NPCR4.RXSEL = 001 <sub>B</sub> )
RXDC4C	P10.7		接收输入 C (NPCR4.RXSEL = 010 <sub>B</sub> )
RXDC4D	CAN3INS		接收输入 D (NPCR4.RXSEL = 011 <sub>B</sub> )
RXDC4E	P1.7		接收输入 E (NPCR4.RXSEL = 100 <sub>B</sub> )
RXDC4F	-		接收输入 F (NPCR4.RXSEL = 101 <sub>B</sub> )
RXDC4G	1		接收输入 G (NPCR4.RXSEL = 110 <sub>B</sub> )
RXDC4H	0		接收输入 H (NPCR4.RXSEL = 111 <sub>B</sub> )
TXDC4	P3.6	O	发送输出
	P7.1		
	P7.2		
	P10.6		

**MultiCAN 节点 5 信号**

RXDC5A	P1.4	I	接收输入 A (NPCR5.RXSEL = 000 <sub>B</sub> )
--------	------	---	--

**控制器局域网络 (MultiCAN)**

信号	来自/送至 哪个模块	I/O 至 CAN	可用作...
RXDC5B	P11.4		接收输入 B (NPCR5.RXSEL = 001 <sub>B</sub> )
RXDC5C	P2.1		接收输入 C (NPCR5.RXSEL = 010 <sub>B</sub> )
RXDC5D	CAN4INS		接收输入 D (NPCR5.RXSEL = 011 <sub>B</sub> )
RXDC5E	-		接收输入 E (NPCR5.RXSEL = 100 <sub>B</sub> )
RXDC5 [G:F]	1		接收输入[G:F] (NPCR5.RXSEL = 101 <sub>B</sub> 或 110 <sub>B</sub> )
RXDC5H	0		接收输入 H (NPCR5.RXSEL = 111 <sub>B</sub> )
TXDC5	P2.0	O	发送输出
	P7.2		
	P11.3		

**一般 MultiCAN 信号**

INT_O[15:0]	中断控制器	O	中断输出线（服务请求） <sup>1)</sup>
	INT_O15 至 CCU62 和 CCU63		通知定时器启动

1) CAN 中断和其它模块共享中断输出线。详情参见 ISSRx 寄存器描述。

### 22.5.7 MultiCAN 模块寄存器地址映射

在 XE16xyM 中，MultiCAN 模块寄存器位于下述地址范围内：

**表 22-19 寄存器的地址空间**

模块	基址	结束地址	注
CAN	200000 <sub>H</sub>	203FFF <sub>H</sub>	16 KB
CANa	208000 <sub>H</sub>	20AFFF <sub>H</sub>	12 KB

注：MultiCAN 模块的完整详细地址映射描述见 XE16xyM 系统单元用户手册章节“寄存器概览”。

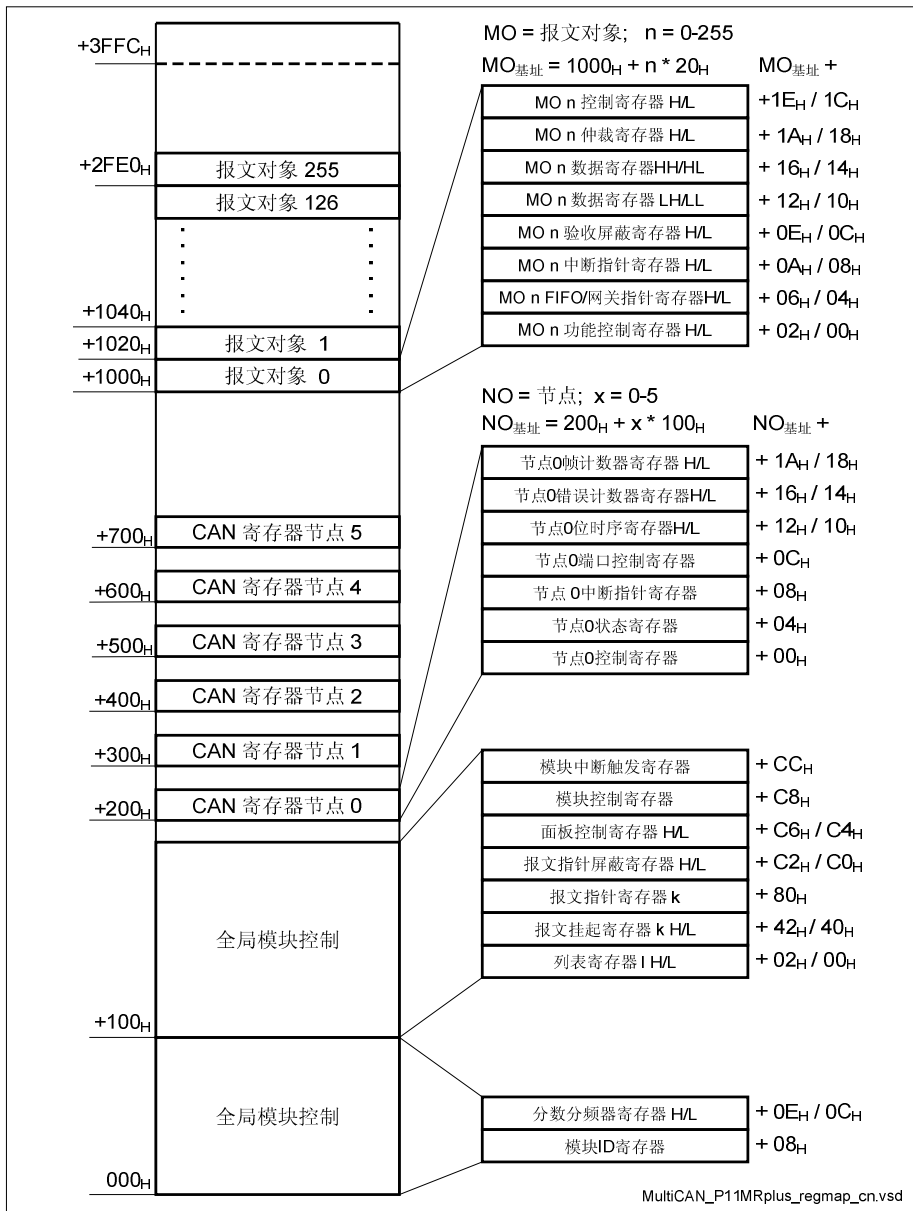
MultiCAN 模块地址范围内，以下寄存器模块位于下列偏移地址处，见图 22-26：

- 0000<sub>H</sub> 时钟控制、分数分频器、ID 一般寄存器
- 0100<sub>H</sub> 全局模块控制寄存器组
- 0200<sub>H</sub> CAN 节点 0 寄存器
- 1000<sub>H</sub> 报文对象存储器（每个报文对象 32 字节）

为了确保每个报文对象内的列表指针正确，复位之后，由列表控制器自动初始化 CAN RAM。该 CAN RAM 初始化的结束由位 PANCTR.BUSY 变为无效指示。结束初始化序列之前，一定不能通过其它任何指令访问 CAN 模块（查询位 PANCTR.BUSY 的指令除外）。

可选择是否使能对 CAN RAM 进行奇偶校验检查。由 SCU 控制该特性。

**控制器局域网络 (MultiCAN)**



**图 22-26 MultiCAN 模块寄存器映射**



## 23 附录：功能和操作更新

XE16xyM 器件是 XE166 家族微控制器的后续产品。该新系列产品提供了许多增强特性。

某些现有特性进行了改进以实现增强特性。因此，对增强特性的控制方式有所不同，这些特性可提供相当出色的功能。

该附录归纳了两种类型的更新：

- **功能更新：**集成新特性以添加新功能。
- **操作更新：**以不同的方式控制现有的特性。

*注：在升级到新的系列产品时，这些提示有助于用户了解新增的功能以避免出错，有助于设计出在两种产品上都能运行的软件。*

### Flash 存储器

XE16xyM 提供额外 64 KB Flash 模块。该 Flash 模块可用于 EEPROM 仿真以及代码和常量存储。因此，最大可用 Flash 容量增加到 832 KB。在 XE16xyM 产品中修改了逻辑扇区。单独的 ECC 状态和强制中断控制支持对 Flash 模块并行编程。程序 Flash 的中断信号被送至 SCU 的中断节点。

### RAM 区

最大可用 SRAM（PSRAM）容量为 32 KB，先前的版本为 64 KB。XE16xyM 还提供 8 KB 备用存储器（SBRAM）。

### MPU

XE16xyM 新增了存储器保护单元（MPU），为应用的不同任务提供相互隔离的专用存储器区。

### MCHK

XE16xyM 新增了存储器检查单元（MCHK），它是一个硬件 CRC32 产生器，面向于高安全性的应用。

### ECC

具有单位检错和纠错（ECC）的纠错机制为 XE16xyM RAM 提供了增强的保护功能。为了保证兼容性，也可选择奇偶校验保护。

## 启动配置

修订通过端口 P10 进行硬件启动配置的内容，以支持附加的、用户可选的启动模式。当  $\overline{\text{TRST}}$  为低电平时该配置被忽略，因此，在这种标准情况下，根本不需要进行配置。

## 通过 DAP 进行调试

除标准 JTAG 调试接口之外，XE16xyM 新增的器件访问端口（DAP）还为用户提供了一个双线调试接口。

## 简化的 EVR 控制

由于片上 EVR 可完全满足 XE16xyM 的供电需求，因此去除引脚 TRef，这样可减少配置操作并提供一个额外的 IO 引脚 P2.13。

## 时钟特性

在 XE16xyM 中，时钟输入引脚 CLKIN1 还可连接至片上 PLL。

振荡器计数器利用位 OSCSTAB 指示振荡器是否稳定工作。

晶体振荡器的增益可编程设置。

## 系统定时器

XE16xyM 新增的系统定时器（STM）支持省电模式下的日历/时钟功能。

## 串行通道

XE16xyM 可提供多达 8 路串行通道，先前的版本可提供 6 路。

由忙碌标志提供 UART 模式下的附加信息。

由帧起始标志提供 SSC 模式下的附加信息。

附加的端口连接选择提高了灵活性。

寄存器 ISSR 中的位 ISS2/ISS3 除了可选择 CAPCOM2 作为中断源外，还可以选择附加的 USIC3 模块作为中断源。

## A/D 转换器

ADC 模块具有以下附加功能：

- 断线检测和复用器测试模式提高了 ADC 的可靠性
- 等间隔采样为滤波器算法提供了优化的输入数据

- 增强的触发/门控功能（由请求源寄存器 RSIRx 代替 PISEL）优化了片上模块之间的相互作用
- 改进了外部复用器的硬件控制

### **MultiCAN**

XE16xyM 中的 MultiCAN 模块提供多达 6 个 CAN 节点（先前的版本为 5 个）和多达 256 个报文对象（先前的版本为 128 个）。附加的端口连接选择提高了灵活性。MultiCAN 模块中的一些错误在本版器件中予以更正。

### **GPT12E**

定时器模块（GPT12E）具有端口输入选择寄存器（PISEL）和附加的端口连接。

### **CCU6x**

CCU6 单元提供了 8 路硬件运行输入（先前的版本仅提供 4 路），改进了与 ADC 模块的内部互连以及映射传送能力。

改进了来自/送至 USIC 模块的触发信号。

附加的端口连接选择提高了灵活性。

### **ERU**

外部请求单元（ERU）的 4 个中断请求线分配给专用的中断节点，它们不再与 CAPCOM2 单元共享这些节点。寄存器 ISSR 中的 ISS0...ISS3 以及 ISS8...ISS11 不再用于选择 ERU 请求。

### **唤醒定时器**

唤醒定时器（WUT）可产生周期性的触发信号。为了实现该功能，本版器件中的时钟源可选、预分频器可编程设置。计数器寄存器 WICR 被重载寄存器 WUTREL 代替。因此，计数器不能由软件直接读取。

### **看门狗定时器**

看门狗定时器（WDT）第二次超时之后，XE16xyM 进入恒复位状态。指示首次超时的状态标志必须由写操作显性清除。服务 WDT 的操作不会自动清除该位。

### **LXBus 地址范围**

扩展 LXBus 的地址范围，增加模块 MultiCAN、USIC0、USIC1 和 USIC2 的备选地址段。

### 其它

可选择附加引脚作为 ESR 逻辑的唤醒触发输入。

快速启动模式支持无 Flash 存储器的器件的操作。

寄存器 ESRDAT 受寄存器安全机制保护。

*注：XE16xyM 中已更正的错误归纳见相应的勘误表。*

## 英飞凌科技中国总部地址及联系方式

---

英飞凌科技（中国）有限公司

地址：上海市浦东张江高科技园区松涛路647弄7-8号

邮编：201203

电话：+86-21-61019000

传真：+86-21-50806204

主页：[www.infineon.com/cn](http://www.infineon.com/cn)

[www.infineon.com](http://www.infineon.com)