

8 位

**XC83x**

8 位单片微控制器

用户手册

V1.1 2010-09

**Microcontrollers**

**Edition 2010-09**

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

**© 2010 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.

# 8 位

## XC83x

8 位单片微控制器

### 用户手册

V1.1 2010-09

## Microcontrollers

XC83x 用户手册

版本信息：V1.1 2010-09

先前的版本：V1.0

页	内容（对上一版本的主要修正）
<b>1-3</b>	删除 XC83x 的引脚配置和封装信息，相关信息可在数据手册中查找。
<b>4-7</b>	将保存域 STx 更名为 MOD_STx。
<b>5-3, 7-11</b>	读取 Flash 时需插入等待状态的个数（1/0 个）和 CPU 的时钟频率有关。
<b>22-2, 22-43</b>	删除两次转换之间的建立时间，相关信息可在数据手册中查找。
<b>22-20</b>	在寄存器 ADC_CRMR1 中添加位域 ENGT 的描述。
<b>24-25, 24-26</b>	在图 24-6 和 24-7 中添加 LPF 增益。
<b>24-39</b>	更新图 24-10。

期待您的指正

本文档如有不当，错误以及遗漏指出，敬请批评指正，以便我们不断改进文档的质量。  
 请将您的建议（以及相关资料发送至：

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





<b>1</b>	<b>简介</b>	1-1 [1]
1.1	XC83x 特性总结	1-2 [1]
1.2	引脚定义及功能	1-3 [1]
1.3	芯片 ID	1-11 [1]
1.4	命名规则	1-12 [1]
1.5	保留位、未定义位及未实现位	1-12 [1]
1.6	缩写	1-13 [1]
<b>2</b>	<b>XC800 内核</b>	2-1 [1]
2.1	概述	2-1 [1]
2.2	XC800 内核功能模块	2-1 [1]
2.3	CPU SFR	2-3 [1]
2.3.1	堆栈指针 (SP, 81 <sub>H</sub> )	2-3 [1]
2.3.2	数据指针 (DPTR, 82-3 <sub>H</sub> )	2-3 [1]
2.3.3	累加器 (ACC, E0 <sub>H</sub> )	2-3 [1]
2.3.4	B 寄存器 (F0 <sub>H</sub> )	2-3 [1]
2.3.5	程序状态字 (PSW, D0 <sub>H</sub> )	2-3 [1]
2.3.6	扩展操作寄存器 (EO, A2 <sub>H</sub> )	2-5 [1]
2.3.7	功率控制寄存器 (PCON, 87 <sub>H</sub> )	2-6 [1]
2.3.8	中断寄存器	2-6 [1]
2.4	内核外设 SFR	2-7 [1]
2.4.1	定时器寄存器	2-7 [1]
2.4.2	UART 寄存器	2-7 [1]
2.5	指令时序	2-8 [1]
<b>3</b>	<b>存储器结构</b>	3-1 [1]
3.1	程序存储器	3-2 [1]
3.2	数据存储器	3-2 [1]
3.2.1	内部数据存储器	3-3 [1]
3.2.2	外部数据存储器	3-3 [1]
3.3	存储器保护策略	3-4 [1]
3.4	特殊功能寄存器	3-4 [1]
3.4.1	映射地址扩展	3-4 [1]
3.4.1.1	系统控制寄存器 0	3-6 [1]
3.4.2	分页地址扩展	3-7 [1]
3.4.2.1	分页寄存器	3-9 [1]
3.4.3	位寻址	3-10 [1]
3.4.4	位保护方案	3-11 [1]
3.4.5	XC83x 寄存器概览	3-13 [1]
3.4.5.1	CPU 寄存器	3-13 [1]
3.4.5.2	MDU 寄存器	3-14 [1]
3.4.5.3	CORDIC 寄存器	3-16 [1]
3.4.5.4	系统控制寄存器	3-16 [1]
3.4.5.5	端口寄存器	3-19 [1]

3.4.5.6	ADC 寄存器 .....	3-21 [1]
3.4.5.7	LEDSCU 寄存器 .....	3-25 [1]
3.4.5.8	RTC 寄存器 .....	3-26 [1]
3.4.5.9	定时器 T2 寄存器 .....	3-28 [1]
3.4.5.10	CCU6 寄存器 .....	3-28 [1]
3.4.5.11	SSC 寄存器 .....	3-33 [1]
3.4.5.12	IIC 寄存器 .....	3-33 [1]
3.4.5.13	OCDS 寄存器 .....	3-34 [1]
<b>4</b>	<b>闪存存储器 .....</b>	<b>4-1 [1]</b>
4.1	Flash 存储器地址映射 .....	4-2 [1]
4.2	Flash Bank 分区 .....	4-3 [1]
4.3	字线地址 .....	4-5 [1]
4.4	工作模式 .....	4-9 [1]
4.5	检错和纠错 .....	4-9 [1]
4.5.1	Flash 出错地址寄存器 .....	4-11 [1]
4.6	在系统编程 .....	4-12 [1]
4.7	在应用编程 .....	4-13 [1]
4.7.1	Flash 编程 .....	4-14 [1]
4.7.1.1	非后台 Flash 编程子程序 .....	4-14 [1]
4.7.1.2	后台 Flash 编程子程序 .....	4-14 [1]
4.7.2	Flash 擦除 .....	4-14 [1]
4.7.2.1	非后台 Flash 擦除子程序 .....	4-15 [1]
4.7.2.2	后台 Flash 擦除子程序 .....	4-15 [1]
4.7.3	中止后台 Flash 擦除 .....	4-15 [1]
4.7.4	Flash 读模式状态 .....	4-17 [1]
<b>5</b>	<b>Boot 及系统启动 .....</b>	<b>5-1 [1]</b>
5.1	用户 ID .....	5-2 [1]
5.2	Boot ROM 工作模式 .....	5-4 [1]
5.2.1	用户模式（量产） .....	5-4 [1]
5.2.2	用户模式（诊断） .....	5-4 [1]
5.2.3	引导程序加载模式 .....	5-4 [1]
5.2.4	OCDS 模式 .....	5-4 [1]
5.2.4.1	备选用户 BSL 模式 .....	5-4 [1]
<b>6</b>	<b>UART 引导程序加载 .....</b>	<b>6-1 [1]</b>
6.1	阶段 I: 和主机自动串行同步 .....	6-1 [1]
6.1.1	概述 .....	6-2 [1]
6.1.2	BG 和 PRE 的计算 .....	6-3 [1]
6.2	阶段 II: 串行通信协议和模式 .....	6-4 [1]
6.2.1	串行通信协议 .....	6-4 [1]
6.2.1.1	传送块结构 .....	6-4 [1]
6.2.1.2	传送块类型 .....	6-4 [1]

6.2.1.3	返回主机的回应码	6-5 [1]
6.2.2	工作模式选择	6-7 [1]
6.2.2.1	接收报文头块	6-7 [1]
6.2.2.2	模式 0: 将用户程序写入 XRAM	6-7 [1]
6.2.2.3	模式 1: 执行 XRAM 中的用户程序	6-9 [1]
6.2.2.4	模式 2: 将用户程序写入 FLASH	6-9 [1]
6.2.2.5	模式 3: 执行 FLASH 中的用户程序	6-11 [1]
6.2.2.6	模式 4: 擦除 Flash 扇区中的用户程序	6-11 [1]
6.2.2.7	模式 6: 设置 4 字节 USER_ID	6-12 [1]
6.2.2.8	模式 A: 获取 4 字节信息	6-12 [1]
<b>7</b>	<b>系统控制单元</b>	<b>7-1 [1]</b>
7.1	带有嵌入式电压调节器的电源系统	7-1 [1]
7.1.1	低压工作条件	7-3 [1]
7.1.2	EVR 寄存器	7-4 [1]
7.2	复位控制	7-6 [1]
7.2.1	复位类型	7-6 [1]
7.2.1.1	上电复位	7-6 [1]
7.2.1.2	看门狗定时器复位	7-6 [1]
7.2.1.3	软复位	7-7 [1]
7.2.1.4	掉电唤醒复位	7-7 [1]
7.2.1.5	欠压复位	7-7 [1]
7.2.2	模块复位行为	7-7 [1]
7.2.3	复位控制寄存器	7-9 [1]
7.3	时钟系统及控制	7-11 [1]
7.3.1	振荡器看门狗	7-12 [1]
7.3.2	时钟丢失检测及恢复	7-12 [1]
7.3.3	32.768 kHz 振荡器时钟的启动控制	7-13 [1]
7.3.4	CCU 寄存器	7-14 [1]
7.4	功率管理	7-16 [1]
7.4.1	功能说明	7-17 [1]
7.4.1.1	空闲模式	7-17 [1]
7.4.1.2	掉电模式	7-17 [1]
7.4.1.3	外设时钟管理	7-21 [1]
7.4.2	功率管理寄存器	7-23 [1]
7.5	SCU 寄存器映射	7-25 [1]
<b>8</b>	<b>看门狗定时器</b>	<b>8-1 [1]</b>
8.1	概述	8-1 [1]
8.2	系统信息	8-2 [1]
8.2.1	复位的影响	8-2 [1]
8.2.2	时钟配置	8-2 [1]
8.2.3	中断事件及分配	8-2 [1]
8.2.4	模块挂起控制	8-3 [1]

8.3	功能说明	8-4 [1]
8.4	寄存器说明	8-6 [1]
8.4.1	看门狗定时器寄存器	8-6 [1]
<b>9</b>	<b>中断系统</b>	9-1 [1]
9.1	中断源	9-1 [1]
9.1.1	中断源和中断向量	9-8 [1]
9.1.2	中断源和中断优先级	9-9 [1]
9.2	中断结构	9-10 [1]
9.2.1	中断结构 1	9-10 [1]
9.2.2	中断结构 2	9-11 [1]
9.3	中断处理	9-12 [1]
9.4	中断响应时间	9-13 [1]
9.5	寄存器说明	9-15 [1]
9.5.1	中断节点使能寄存器	9-16 [1]
9.5.2	外部中断控制寄存器	9-19 [1]
9.5.3	中断标志寄存器	9-23 [1]
9.5.4	中断优先级寄存器	9-29 [1]
9.6	中断标志概览	9-31 [1]
<b>10</b>	<b>调试系统</b>	10-1 [1]
10.1	概述	10-1 [1]
10.1.1	调试系统的组成部分	10-1 [1]
10.2	产品信息	10-1 [1]
10.2.1	引脚配置	10-2 [1]
10.2.2	时钟配置	10-2 [1]
10.2.3	中断事件和中断节点分配	10-2 [1]
10.2.4	调试挂起控制	10-2 [1]
10.2.5	JTAG ID	10-4 [1]
10.3	调试系统的功能概述	10-4 [1]
10.3.1	识别调试事件	10-4 [1]
10.3.2	激活监控程序	10-5 [1]
10.3.3	调试挂起控制	10-6 [1]
10.3.4	运行监控程序	10-6 [1]
10.3.5	返回用户程序	10-6 [1]
10.3.6	单步执行	10-6 [1]
10.4	断点产生模块	10-7 [1]
10.4.1	产生硬件断点	10-7 [1]
10.4.1.1	指令地址断点	10-7 [1]
10.4.1.2	IRAM 地址断点	10-8 [1]
10.4.1.3	跟踪暂停地址的改变	10-8 [1]
10.4.2	处理外部暂停	10-9 [1]
10.4.3	处理软件断点	10-9 [1]
10.5	不进入监控模式的断点响应方式	10-9 [1]

10.5.1	触发 NMI 请求	10-9 [1]
10.6	NMI 请求及控制	10-10 [1]
10.6.1	来自 OCDS 的 NMI 请求	10-10 [1]
10.6.2	OCDS NMI 服务程序处理	10-10 [1]
10.6.3	OCDS 对 NMI 的控制	10-11 [1]
10.7	NMI 模式的优先级高于调试模	10-11 [1]
10.8	寄存器说明	10-11 [1]
10.8.1	控制和状态寄存器	10-13 [1]
10.8.2	硬件断点寄存器	10-14 [1]
10.8.3	监控工作寄存器	10-17 [1]
<b>11</b>	<b>并行端口</b>	11-1 [1]
11.1	基本端口操作	11-2 [1]
11.1.1	通用寄存器说明	11-7 [1]
11.1.1.1	寄存器映射	11-7 [1]
11.1.1.2	寄存器概览	11-8 [1]
11.2	P0 口	11-16 [1]
11.2.1	功能	11-16 [1]
11.2.2	寄存器说明	11-22 [1]
11.3	P1 口	11-26 [1]
11.3.1	功能	11-27 [1]
11.3.2	寄存器说明	11-30 [1]
11.4	P2 口	11-35 [1]
11.4.1	功能	11-36 [1]
11.4.2	寄存器说明	11-39 [1]
11.5	P3 口	11-41 [1]
11.5.1	功能	11-41 [1]
11.5.2	寄存器说明	11-43 [1]
<b>12</b>	<b>乘法 / 除法单元</b>	12-1 [1]
12.1	概述	12-1 [1]
12.2	系统信息	12-2 [1]
12.2.1	时钟配置	12-2 [1]
12.2.2	中断事件及分配	12-3 [1]
12.3	功能说明	12-3 [1]
12.3.1	除法操作	12-4 [1]
12.3.2	归一化	12-4 [1]
12.3.3	移位	12-5 [1]
12.3.4	带有一次左移操作的乘法	12-5 [1]
12.3.5	带有一次右移操作的除法	12-5 [1]
12.3.6	忙碌标志	12-5 [1]
12.3.7	检错	12-6 [1]
12.4	中断产生	12-6 [1]
12.5	寄存器说明	12-7 [1]

12.5.1	操作数和结果寄存器 .....	12-8 [1]
12.5.2	控制寄存器 .....	12-10 [1]
12.5.3	状态寄存器 .....	12-12 [1]
<b>13</b>	<b>CORDIC 协处理器 .....</b>	<b>13-1 [1]</b>
13.1	概述 .....	13-1 [1]
13.2	系统信息 .....	13-2 [1]
13.2.1	时钟配置 .....	13-2 [1]
13.2.1.1	外设管理控制寄存器 1 .....	13-2 [1]
13.2.2	中断事件及分配 .....	13-3 [1]
13.3	功能描述 .....	13-4 [1]
13.3.1	CORDIC 协处理器的操作 .....	13-4 [1]
13.3.2	结果数据归一化 .....	13-4 [1]
13.3.3	CORDIC 协处理器操作模式 .....	13-5 [1]
13.3.3.1	收敛域 .....	13-8 [1]
13.3.3.2	溢出考虑 .....	13-8 [1]
13.3.4	CORDIC 协处理器数据格式 .....	13-9 [1]
13.3.5	CORDIC 协处理器精度 .....	13-9 [1]
13.3.6	CORDIC 协处理器性能 .....	13-11 [1]
13.4	中断 .....	13-11 [1]
13.5	CORDIC 协处理器硬件 .....	13-12 [1]
13.5.1	反正切和双曲反正切查找表 .....	13-12 [1]
13.5.2	线性函数的仿真查找表 .....	13-13 [1]
13.6	寄存器描述 .....	13-14 [1]
13.6.1	控制寄存器 .....	13-14 [1]
13.6.2	状态和数据控制寄存器 .....	13-16 [1]
13.6.3	数据寄存器 .....	13-17 [1]
<b>14</b>	<b>定时器 T0 和 T1 .....</b>	<b>14-1 [1]</b>
14.1	概述 .....	14-1 [1]
14.2	系统信息 .....	14-1 [1]
14.2.1	引脚 .....	14-1 [1]
14.2.2	时钟配置 .....	14-2 [1]
14.2.3	中断事件和分配 .....	14-2 [1]
14.3	定时器基本操作 .....	14-3 [1]
14.4	定时器工作模式 .....	14-4 [1]
14.4.1	模式 0 .....	14-5 [1]
14.4.2	模式 1 .....	14-6 [1]
14.4.3	模式 2 .....	14-7 [1]
14.4.4	模式 3 .....	14-8 [1]
14.5	寄存器描述 .....	14-9 [1]
14.5.1	定时器 T0 和 T1 寄存器 .....	14-9 [1]
<b>15</b>	<b>定时器 T2 .....</b>	<b>15-1 [1]</b>

<b>15.1</b>	<b>概述</b>	<b>15-1 [1]</b>
<b>15.2</b>	<b>系统信息</b>	<b>15-1 [1]</b>
15.2.1	引脚	15-1 [1]
15.2.2	时钟配置	15-3 [1]
15.2.3	中断事件和分配	15-4 [1]
15.2.4	IP 互连	15-4 [1]
15.2.5	模块挂起控制	15-4 [1]
<b>15.3</b>	<b>基本定时器操作</b>	<b>15-6 [1]</b>
<b>15.4</b>	<b>自动重载模式</b>	<b>15-6 [1]</b>
15.4.1	禁止递增 / 递减计数	15-6 [1]
15.4.2	使能递增 / 递减计数	15-7 [1]
<b>15.5</b>	<b>捕获模式</b>	<b>15-9 [1]</b>
<b>15.6</b>	<b>计数时钟</b>	<b>15-10 [1]</b>
<b>15.7</b>	<b>外部中断功能</b>	<b>15-11 [1]</b>
<b>15.8</b>	<b>寄存器描述</b>	<b>15-12 [1]</b>
15.8.1	模式寄存器	15-13 [1]
15.8.2	控制寄存器	15-14 [1]
15.8.3	定时器 T2 重载 / 捕获寄存器	15-16 [1]
15.8.4	定时器 T2 计数寄存器	15-17 [1]
<b>16</b>	<b>实时时钟</b>	<b>16-1 [1]</b>
<b>16.1</b>	<b>概述</b>	<b>16-1 [1]</b>
<b>16.2</b>	<b>系统信息</b>	<b>16-1 [1]</b>
16.2.1	引脚	16-1 [1]
16.2.2	中断事件和分配	16-1 [1]
16.2.3	模块挂起控制	16-2 [1]
<b>16.3</b>	<b>振荡器</b>	<b>16-2 [1]</b>
<b>16.4</b>	<b>基本定时器操作</b>	<b>16-2 [1]</b>
<b>16.5</b>	<b>实时时钟模式</b>	<b>16-2 [1]</b>
16.5.1	模式 0: 使用 32.768 kHz 晶振时钟的计时模式	16-3 [1]
16.5.2	模式 1: 使用 75 kHz 振荡器时钟的周期性唤醒模式	16-4 [1]
16.5.3	模式 2: 使用 1 Hz 外部时钟的计时模式	16-5 [1]
16.5.4	模式 3: 使用外部时钟的定时器模式	16-6 [1]
<b>16.6</b>	<b>省电模式选择</b>	<b>16-7 [1]</b>
<b>16.7</b>	<b>寄存器描述</b>	<b>16-8 [1]</b>
16.7.1	实时时钟寄存器	16-9 [1]
16.7.2	振荡器寄存器	16-21 [1]
<b>17</b>	<b>UART</b>	<b>17-1 [1]</b>
<b>17.1</b>	<b>概述</b>	<b>17-1 [1]</b>
<b>17.2</b>	<b>系统信息</b>	<b>17-1 [1]</b>
17.2.1	引脚	17-1 [1]
17.2.2	时钟配置	17-3 [1]
17.2.3	中断事件和分配	17-3 [1]

17.3	UART 模式	17-3 [1]
17.3.1	模式 0, 8 位移位寄存器, 波特率固定	17-4 [1]
17.3.2	模式 1, 8 位 UART, 波特率可变	17-4 [1]
17.3.3	模式 2, 9 位 UART, 波特率固定	17-6 [1]
17.3.4	模式 3, 9 位 UART, 波特率可变	17-6 [1]
17.4	多处理器通信	17-8 [1]
17.5	波特率产生	17-9 [1]
17.5.1	固定时钟	17-9 [1]
17.5.2	UART 波特率发生器	17-9 [1]
17.5.3	定时器 1	17-12 [1]
17.6	UART 中的 LIN 支持	17-13 [1]
17.6.1	LIN 协议	17-13 [1]
17.6.2	LIN 报文头发送	17-15 [1]
17.6.2.1	和主机自动同步	17-15 [1]
17.6.2.2	分隔 / 同步域检测逻辑的初始化	17-15 [1]
17.6.2.3	波特率范围选择	17-16 [1]
17.6.2.4	LIN 波特率检测	17-17 [1]
17.7	寄存器描述	17-19 [1]
17.7.1	UART 寄存器	17-20 [1]
17.7.2	波特率产生控制和状态寄存器	17-21 [1]
17.7.3	波特率发生器定时器 / 重载寄存器	17-24 [1]
18	IIC 总线	18-1 [1]
18.1	概述	18-1 [1]
18.2	系统信息	18-1 [1]
18.2.1	引脚	18-1 [1]
18.2.1.1	输出引脚配置	18-2 [1]
18.2.2	时钟配置	18-2 [1]
18.2.3	中断事件及其分配	18-3 [1]
18.3	状态码	18-4 [1]
18.4	波特率产生	18-5 [1]
18.5	时钟同步	18-6 [1]
18.6	总线仲裁	18-6 [1]
18.7	软件复位	18-6 [1]
18.8	操作模式	18-6 [1]
18.8.1	主机发送	18-6 [1]
18.8.2	主机接收	18-10 [1]
18.8.3	从机发送	18-11 [1]
18.8.4	从机接收	18-12 [1]
18.9	寄存器描述	18-13 [1]
18.9.1	从机地址寄存器	18-14 [1]
18.9.2	数据寄存器	18-15 [1]
18.9.3	控制寄存器	18-16 [1]



18.9.4	状态寄存器 .....	18-17 [1]
18.9.5	波特率控制寄存器 .....	18-19 [1]
18.9.6	软件复位寄存器 .....	18-19 [1]
<b>19</b>	<b>高速串行同步接口 .....</b>	<b>19-1 [1]</b>
19.1	概述 .....	19-1 [1]
19.2	系统信息 .....	19-2 [1]
19.2.1	引脚 .....	19-2 [1]
19.2.2	时钟配置 .....	19-6 [1]
19.2.3	中断事件和分配 .....	19-7 [1]
19.3	一般操作 .....	19-8 [1]
19.3.1	工作模式选择 .....	19-8 [1]
19.3.2	全双工工作 .....	19-9 [1]
19.3.3	半双工工作 .....	19-11 [1]
19.3.4	连续传送 .....	19-12 [1]
19.3.5	波特率产生 .....	19-13 [1]
19.3.6	检错机制 .....	19-14 [1]
19.4	中断 .....	19-16 [1]
19.5	寄存器描述 .....	19-17 [1]
19.5.1	配置寄存器 .....	19-18 [1]
19.5.2	波特率定时器重载寄存器 .....	19-22 [1]
19.5.3	发送缓存寄存器 .....	19-22 [1]
19.5.4	接收缓存寄存器 .....	19-23 [1]
<b>20</b>	<b>LED 和触摸感应控制器 .....</b>	<b>20-1 [1]</b>
20.1	概述 .....	20-1 [1]
20.2	系统信息 .....	20-2 [1]
20.2.1	引脚配置 .....	20-2 [1]
20.2.2	时钟配置 .....	20-4 [1]
20.2.3	中断事件和中断节点分配 .....	20-4 [1]
20.2.4	IP 互连 .....	20-5 [1]
20.2.5	调试挂起控制 .....	20-5 [1]
20.3	引脚上 LED 和触摸感应功能的时分复用控制 .....	20-5 [1]
20.4	LED 驱动 .....	20-8 [1]
20.4.1	LED 引脚分配和电流驱动能力 .....	20-9 [1]
20.5	触摸盘感应 .....	20-10 [1]
20.5.1	手指感应 .....	20-11 [1]
20.6	引脚上 LED 和触摸感应功能时分复用 .....	20-13 [1]
20.7	功能使能和控制的注意事项 .....	20-14 [1]
20.8	LEDTSCU 时序计算 .....	20-15 [1]
20.9	LEDTSCU 引脚控制 .....	20-17 [1]
20.10	中断 .....	20-18 [1]
20.11	寄存器描述 .....	20-19 [1]
20.11.1	全局控制和状态 .....	20-20 [1]

20.11.2	功能控制寄存器 .....	20-23 [1]
<b>21</b>	<b>捕获 / 比较单元 6 (CCU6) .....</b>	<b>21-1 [1]</b>
21.1	简介 .....	21-1 [1]
21.1.1	特性概述 .....	21-2 [1]
21.1.2	框图 .....	21-3 [1]
21.1.3	寄存器概述 .....	21-4 [1]
21.2	系统信息 .....	21-7 [1]
21.2.1	引脚 .....	21-7 [1]
21.2.2	时钟配置 .....	21-11 [1]
21.2.3	中断事件和分配 .....	21-13 [1]
21.2.4	IP 互连 .....	21-15 [1]
21.2.5	模块挂起控制 .....	21-17 [1]
21.3	定时器 T12 模块 .....	21-17 [1]
21.3.1	T12 概述 .....	21-19 [1]
21.3.2	T12 计数策略 .....	21-21 [1]
21.3.2.1	时钟选择 .....	21-21 [1]
21.3.2.2	边沿对齐 / 中心对齐模式 .....	21-21 [1]
21.3.2.3	单次模式 .....	21-24 [1]
21.3.3	T12 的比较模式 .....	21-25 [1]
21.3.3.1	比较通道 .....	21-25 [1]
21.3.3.2	通道状态位 .....	21-25 [1]
21.3.3.3	类磁滞控制模式 .....	21-30 [1]
21.3.4	比较模式输出路径 .....	21-31 [1]
21.3.4.1	死区时间产生 .....	21-31 [1]
21.3.4.2	状态选择 .....	21-33 [1]
21.3.4.3	输出调制和电平选择 .....	21-34 [1]
21.3.5	T12 捕获模式 .....	21-36 [1]
21.3.6	T12 映射寄存器传送 .....	21-40 [1]
21.3.7	定时器 T12 工作模式选择 .....	21-41 [1]
21.3.8	T12 相关寄存器 .....	21-42 [1]
21.3.8.1	T12 计数寄存器 .....	21-42 [1]
21.3.8.2	周期寄存器 .....	21-42 [1]
21.3.8.3	捕获 / 比较寄存器 .....	21-44 [1]
21.3.8.4	捕获 / 比较映射寄存器 .....	21-45 [1]
21.3.8.5	死区时间控制寄存器 .....	21-47 [1]
21.3.9	捕获 / 比较控制寄存器 .....	21-49 [1]
21.3.9.1	通道状态位 .....	21-49 [1]
21.3.9.2	T12 模式控制寄存器 .....	21-53 [1]
21.3.9.3	定时器控制寄存器 .....	21-55 [1]
21.4	定时器 T13 模块 .....	21-63 [1]
21.4.1	T13 模块概述 .....	21-63 [1]
21.4.2	T13 计数策略 .....	21-66 [1]

21.4.2.1	定时器 T13 计数操作	21-66 [1]
21.4.2.2	单次模式	21-67 [1]
21.4.2.3	与 T12 同步	21-68 [1]
21.4.3	T13 比较模式	21-70 [1]
21.4.4	比较模式输出路径	21-72 [1]
21.4.5	T13 映射寄存器传送	21-73 [1]
21.4.6	T13 相关寄存器	21-75 [1]
21.4.6.1	T13 计数寄存器	21-75 [1]
21.4.6.2	周期寄存器	21-77 [1]
21.4.6.3	比较寄存器	21-78 [1]
21.4.6.4	比较映射寄存器	21-78 [1]
21.5	强制中断处理	21-80 [1]
21.6	多通道模式	21-82 [1]
21.7	霍尔传感器模式	21-84 [1]
21.7.1	霍尔序列评估	21-85 [1]
21.7.2	霍尔序列比较逻辑	21-87 [1]
21.7.3	霍尔模式标志位	21-88 [1]
21.7.4	实现无刷直流电机控制的霍尔模式	21-90 [1]
21.8	调制控制寄存器	21-92 [1]
21.8.1	调制控制	21-92 [1]
21.8.2	强制中断控制寄存器	21-94 [1]
21.8.3	被动态电平寄存器	21-97 [1]
21.8.4	多通道模式寄存器	21-98 [1]
21.9	中断处理	21-103 [1]
21.9.1	中断结构	21-103 [1]
21.9.2	中断寄存器	21-105 [1]
21.9.2.1	中断状态寄存器	21-105 [1]
21.9.2.2	中断状态置位寄存器	21-108 [1]
21.9.2.3	状态复位寄存器	21-110 [1]
21.9.2.4	中断使能寄存器	21-112 [1]
21.9.2.5	中断节点指针寄存器	21-115 [1]
21.10	一般模块操作	21-117 [1]
21.10.1	输入选择	21-117 [1]
21.10.2	通用寄存器	21-118 [1]
21.10.2.1	端口输入寄存器	21-118 [1]
21.11	寄存器映射	21-121 [1]
<b>22</b>	<b>模数转换器</b>	<b>22-1 [1]</b>
22.1	系统信息	22-3 [1]
22.1.1	引脚	22-3 [1]
22.1.2	时钟配置	22-3 [1]
22.1.3	中断事件和分配	22-4 [1]
22.1.4	IP 互连	22-4 [1]

22.2	简介和基本结构 .....	22-7 [1]
22.3	电气模型 .....	22-10 [1]
22.4	转换特性和误差定义 .....	22-12 [1]
22.5	一般功能配置 .....	22-13 [1]
22.5.1	一般时钟方案和控制 .....	22-13 [1]
22.6	转换请求产生 .....	22-16 [1]
22.6.1	通道扫描请求源处理 .....	22-18 [1]
22.6.2	队列请求源处理 .....	22-23 [1]
22.6.3	硬件触发选择 .....	22-35 [1]
22.7	请求源仲裁 .....	22-36 [1]
22.7.1	仲裁器时序 .....	22-37 [1]
22.7.2	请求源优先级和转换启动模式 .....	22-38 [1]
22.8	模拟输入通道配置 .....	22-41 [1]
22.8.1	参考选择 .....	22-41 [1]
22.8.2	通道参数 .....	22-44 [1]
22.8.3	极限检查 .....	22-48 [1]
22.8.4	假名特性 .....	22-50 [1]
22.8.5	超出范围比较器 .....	22-53 [1]
22.8.6	转换时序 .....	22-57 [1]
22.8.7	通道事件和中断 .....	22-59 [1]
22.9	转换结果处理 .....	22-60 [1]
22.9.1	转换结果的保存 .....	22-60 [1]
22.10	等待 - 读取模式 .....	22-75 [1]
22.10.1	结果事件和中断 .....	22-76 [1]
22.10.2	数据压缩和滤波 .....	22-76 [1]
22.11	中断请求处理 .....	22-79 [1]
22.12	寄存器映射 .....	22-86 [1]
<b>23</b>	<b>Boot ROM 用户子程序 .....</b>	<b>23-1 [1]</b>
23.1	Flash Bank 读模式状态子程序 .....	23-2 [1]
23.2	获得 4 字节信息 .....	23-3 [1]
23.3	特性设置子程序 .....	23-3 [1]
23.4	UART 自动波特率子程序 .....	23-5 [1]
23.5	UART BSL 子程序 .....	23-6 [1]
23.6	Flash 编程子程序 .....	23-7 [1]
23.7	Flash 擦除子程序 .....	23-9 [1]
23.8	Flash 擦除中止子程序 .....	23-12 [1]
<b>24</b>	<b>ROM Library .....</b>	<b>24-1 [1]</b>
24.1	定点 ROM Library .....	24-2 [1]
24.1.1	P 控制器程序 .....	24-2 [1]
24.1.2	PI 控制器程序 .....	24-3 [1]
24.1.3	PT1_24 控制器程序 .....	24-5 [1]
24.1.4	PT1_32 控制器程序 .....	24-6 [1]

24.1.5	Cartesian-Polar 坐标转换程序	24-7 [1]
24.1.6	Clarke 变换程序	24-8 [1]
24.1.7	向量旋转 (Park 变换) 程序	24-10 [1]
24.1.8	向量逆旋转 (Park 逆变换) 程序	24-11 [1]
24.1.9	Sine/Cosine 程序	24-12 [1]
24.2	LED 和触摸感应控制器 ROM Library	24-14 [1]
24.2.1	SET_LDLINE_CMP 函数 (LED 和触摸感应)	24-15 [1]
24.2.1.1	SET_LDLINE_CMP 函数的输入 (仅使能 LED 功能)	24-17 [1]
24.2.1.2	SET_LDLINE_CMP 函数的输入 (仅使能触摸感应功能)	24-19 [1]
24.2.1.3	SET_LDLINE_CMP 函数的输入 (使能 LED 和触摸感应功能)	24-21 [1]
24.2.2	FINDTOUCHEDPAD 函数 (触摸感应)	24-25 [1]
24.2.2.1	FINDTOUCHEDPAD 函数的输入	24-27 [1]
24.2.2.2	FINDTOUCHEDPAD 函数的输出	24-31 [1]
24.2.2.3	FINDTOUCHEDPAD 函数的 XRAM 参数	24-34 [1]
24.2.2.4	函数的具体实现	24-36 [1]
24.2.3	中断时函数的使用	24-44 [1]
24.3	EEPROM 模拟 ROM Library	24-46 [1]
24.3.1	特性	24-46 [1]
24.3.2	系统要求	24-46 [1]
24.3.3	概念	24-46 [1]
24.3.4	API 说明	24-47 [1]
24.3.4.1	常量定义	24-47 [1]
24.3.4.2	EEPROMInfo 数据结构	24-48 [1]
24.3.4.3	函数	24-48 [1]
24.3.4.4	API 使用示例	24-52 [1]

# 1 简介

XC83x 是高性能 8 位微控制器 XC800 家族的成员，其设计基于和工业标准 8051 处理器兼容的 XC800 内核。XC83x 通过高度集成的片上模块，如片上振荡器和电压调节器（支持 2.5V - 5.0V 的电源供电），为用户提升应用能力提高了巨大的空间。此外，XC83x 内嵌 Flash（闪存）存储器，为系统开发和批量生产提供了很大的灵活性。XC83x 的存储器保护策略为用户知识产权（IP）提供了读保护。

多闪存组（multi-bank）Flash 结构支持在应用编程（IAP），运行某 bank 中的用户程序的同时可编程或擦除另一 bank 中的内容。通过 Boot ROM 中的引导加载程序（BSL）可进行在系统编程（ISP），通过外部主机（如 PC）对嵌入式 Flash 方便的编程或擦除。

其它主要特性包括：捕获 / 比较单元 6（CCU6），产生电机控制专用的脉宽调制信号；10 位模数转换器（ADC），内置带有差分输入通道的超出范围比较器，具有自动扫描和结果累加（用于抗混迭滤波或结果平均）等扩展功能；乘法 / 除法单元（MDU），支持 XC800 内核用于需要大量数学运算的高级电机控制（如磁场定向控制）；CORDIC（坐标旋转数字计算机）协处理器，可快速计算三角函数、线性或双曲线函数以实现向量旋转和转换；LED 和触摸感应控制器（LEDTSCU），通过 LED 矩阵驱动 7 段显示器并同时支持触摸感应功能；实时时钟（RTC），支持低功耗应用下的周期性唤醒；片上调试支持（OCDS）单元，提供基于 XC800 系统进行软件开发与调试所需的基本功能。此外，通过扩展的 UART 特性和 LIN 底层驱动部分使得大多数器件可支持本地互连网络（LIN）的应用。XC83x 还为用户提供了不同的省电模式选择，以满足低功耗应用的需求；其丰富的片上外设功能由特殊功能寄存器（SFR）控制，采用智能分页机制（优化中断处理）来扩展 SFR 的地址范围。

XC83x 的功能单元如图 1-1 所示。



图 1-1 XC83x 功能单元

## 1.1 XC83x 特性总结

XC83x 的主要特性总结如下：

- 高性能 XC800 内核
  - 和标准 8051 内核兼容
  - 2 个时钟的机器周期结构（快速无等待内存访问）
  - 双数据指针
- 片内存储器
  - 8 KB Boot ROM，存放启动固件、Flash 和用户程序及 ROM library
  - 256B RAM；外加 64 字节监控 RAM
  - 256B XRAM
  - 4/8 KB Flash，用于存放程序代码和数据（包括存储器保护策略）
- I/O 口 2.5 - 5.5V 供电；内核逻辑电路 2.5V 供电（由嵌入式电压调节器产生）
- 上电复位产生
- IO 电压和内核电压压降检测
- 片内振荡器（OSC）产生时钟
  - 时钟丢失检测
- 省电模式
  - 空闲模式
  - 掉电模式，可通过实时时钟中断唤醒系统
  - 每个外设均由时钟门控制
- 带有可编程窗界的看门狗定时器（WDT），用于刷新及溢出前报警
- 三个通用 I/O 口
  - 4 个大电流 I/O
  - 2 个大灌电流 I/O
  - 多达 25 个数字 I/O 引脚
  - 多达 8 个数字 / 模拟输入引脚
- 用于磁场定向控制（FOC）的 16 位向量计算单元
  - 用于算术运算的乘法 / 除法单元（MDU）
  - 用于三角函数计算的 CORDIC 单元
- 支持定点控制和 EEPROM 模拟的软件库
- 8 通道、10 位模数转换单元（ADC）
  - 最多支持 7 路差分输入通道
  - 多达 8 路通道，超出范围比较器
- 三个 16 位定时器：定时器 T0、T1 和 T2
- 带有 32.768 kHz 晶振引出脚的实时时钟
- 用于产生 PWM 信号的捕获 / 比较单元（CCU6）
- 全双工串行接口（UART）
- 同步串行通道（SSC）
- IIC 串行接口
- LED 和触摸感应控制器（LEDTSCU）
- 通过单引脚 DAP 接口（SPD）支持片上调试
  - 1 KB 监控 RAM（Boot ROM 的一部分）

## – 64 字节监控 RAM

XC83x 的框图如图 1-2 所示。

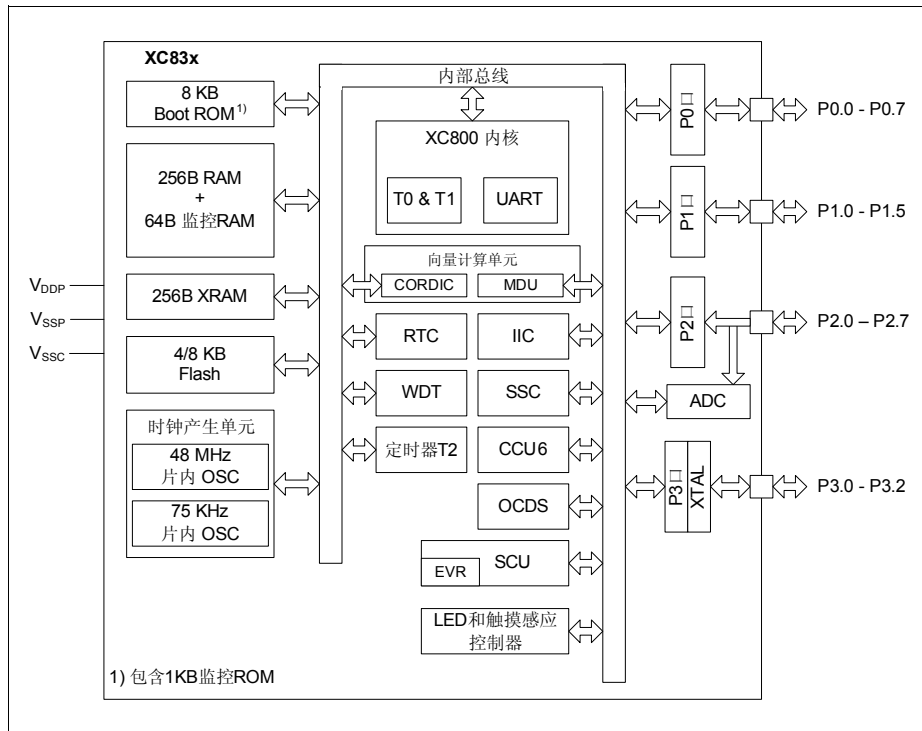


图 1-2 XC83x 框图

## 1.2 引脚定义及功能

复位后所有引脚被配置为具有下列特性之一的输入引脚：

- 仅使能上拉器件 (PU)
- 仅使能下拉器件 (PD)
- 上拉 / 下拉器件均被禁止的高阻态 (Hi-Z)

XC83x 的引脚功能及缺省状态归纳见表 1-1。



**表 1-1 XC83x 的引脚定义和功能**

符号	类型	复位状态	引脚功能
<b>P0</b>	I/O		<b>P0 口</b> P0 口是通用双向 I/O 口。它还可用作 LEDTSCU、定时器 T0、T1 和 T2、SSC、CCU6、IIC、SPD 和 UART 的功能引脚。
P0.0		高阻	T2_0            定时器 T2 输入 T13HR_1       CCU6 定时器 T13 硬件运行输入 MTSR_2        SSC 主机发送输出 / 从机接收输入 MRST_3        SSC 主机接收输入 T12HR_0       CCU6 定时器 T12 硬件运行输入 CCPOS0_0      CCU6 霍尔输入 0 TSIN0          触摸感应输入 0 LINE0          LED 行 0 COUT61_1      捕获 / 比较通道 1 输出
P0.1		高阻	T0_0            定时器 T0 输入 CC61_1         捕获 / 比较通道 1 输入 / 输出 MTSR_3        SSC 从机接收输入 MRST_2        SSC 主机接收输入 / 从机发送输出 T13HR_0       CCU6 定时器 T13 硬件运行输入 CCPOS1_0      CCU6 霍尔输入 1 TSIN1          触摸感应输入 1 LINE1          LED 行 1
P0.2		高阻	T1_0            定时器 T1 输入 CC62_1         捕获 / 比较通道 2 输入 / 输出 SCL_1          IIC 时钟线 CCPOS2_0      CCU6 霍尔输入 2 TSIN2          触摸感应输入 2 LINE2          LED 行 2

**表 1-1 XC83x 的引脚定义和功能**

符号	类型	复位状态	引脚功能	
P0.3		高阻	CC60_1	捕获 / 比较通道 0 输入 / 输出
			SDA_1	IIC 数据线
			CTRAP_0	CCU6 强制中断输入
			TSIN3	触摸感应输入 3
			LINE3	LED 行 3
P0.4		下拉	T2EX_1	定时器 T2 外部触发输入
			SCK_0	SSC 时钟输入 / 输出
			SCL_0	IIC 时钟线
			CTRAP_1	CCU6 强制中断输入
			EXINT1_0	外部中断输入 1
			TSIN4	触摸感应输入 4
			LINE4	LED 行 4
			EXF2_0	定时器 T2 溢出标志
			COL0_1	LED 列 0
			COL3_2	LED 列 3
P0.5		高阻	RXD_0	UART 接收输入
			MTSR_0	SSC 主机发送输出 / 从机接收输入
			MRST_1	SSC 主机接收输入
			EXINT0_0	外部中断输入 0
			TSIN5	触摸感应输入 5
			LINE5	LED 行 5
			COUT62_1	捕获 / 比较通道 2 输出
			TXD_4	UART 发送输出
			COL1_1	LED 列 1
			EXF2_3	定时器 T2 溢出标志

表 1-1 XC83x 的引脚定义和功能

符号	类型	复位状态	引脚功能	
P0.6		上拉	SPD_1	SPD 输入 / 输出
			RXD_1	UART 接收输入
			SDA_0	IIC 数据线
			MTSR_1	SSC 从机接收输入
			MRST_0	SSC 主机接收输入 / 从机发送输出
			EXINT0_1	外部中断输入 0
			T2EX_0	定时器 T2 外部触发输入
			TSIN6	触摸感应输入 6
			LINE6	LED 行 6
			TXD_0	UART 发送输出
			COL2_1	LED 列 2
			COLA_2	LED 列 A
P0.7		高阻	SCL_3	IIC 时钟线
			TSIN7	触摸感应输入 7
			LINE7	LED 行 7
			TXD_5	UART 发送输出 / 2 线 UART BSL 发送输出
			COUT63_0	捕获 / 比较通道 3 输出
			COL3_1	LED 列 3
			COLA_3	LED 列 A
P1	I/O		<b>P1</b> P1 口是通用双向 I/O 口。它还可用作 CCU6、LEDTSCU、SPD、UART 和定时器 T2 的功能引脚。	
P1.0		高阻	SPD_2	SPD 输入 / 输出
			RXD_2	UART 接收输入
			T2EX_2	定时器 T2 外部触发输入
			EXINT0_2	外部中断输入 0
			COL0_0	LED 列 0
			COUT60_0	捕获 / 比较通道 0 输出
			TXD_1	UART 发送输出

**表 1-1 XC83x 的引脚定义和功能**

符号	类型	复位状态	引脚功能	
P1.1		高阻	CC60_0 COL1_0 TXD_2	捕获 / 比较通道 0 输入 / 输出 LED 列 1 UART 发送输出
P1.2		高阻	EXINT4 COL2_0 COUT61_0 COUT63_1	外部中断输入 4 LED 列 2 捕获 / 比较通道 1 输出 捕获 / 比较通道 3 输出
P1.3		高阻	CC61_0 COL3_0 EXF2_2	捕获 / 比较通道 1 输入 / 输出 LED 列 3 定时器 T2 溢出标志
P1.4		高阻	EXINT5 COL4 COUT62_0 COUT63_2	外部中断输入 5 LED 列 4 捕获 / 比较通道 2 输出 捕获 / 比较通道 3 输出
P1.5		高阻	CC62_0 COL5 COLA_1	捕获 / 比较通道 2 输入 / 输出 LED 列 5 LED 列 A
<b>P2</b>	<b>I</b>		<b>P2 口</b> P2 口是通用单向输入口。它还可用作 ADC 和超出范围比较器、CCU6、定时器 T2、SSC 和 UART 的输入引脚。	
P2.0		高阻	CCPOS0_1 T12HR_2 T13HR_2 T2EX_3 T2_1 EXINT0_3 AN0	CCU6 霍尔输入 0 CCU6 定时器 T12 硬件运行输入 CCU6 定时器 T13 硬件运行输入 定时器 T2 外部触发输入 定时器 T2 输入 外部中断输入 0 模拟输入 0 / 超出范围比较器通道 0

**表 1-1 XC83x 的引脚定义和功能**

符号	类型	复位状态	引脚功能
P2.1		高阻	CCPOS1_1 CCU6 霍尔输入 1 RXD_5 UART 接收输入 MTSR_6 SSC 从机接收输入 T0_1 定时器 T0 输入 EXINT1_1 外部中断输入 1 AN1 模拟输入 1 / 超出范围比较器通道 1
P2.2		高阻	CCPOS2_1 CCU6 霍尔输入 2 T12HR_3 CCU6 定时器 T12 硬件运行输入 T13HR_3 CCU6 定时器 T13 硬件运行输入 SCK_3 SSC 时钟输入 / 输出 T1_1 定时器 T1 输入 EXINT2_0 外部中断输入 2 AN2 模拟输入 2 / 超出范围比较器通道 2
P2.3		高阻	CCPOS0_2 CCU6 霍尔输入 0 CTRAP_2 CCU6 强制中断输入 T2_2 定时器 T2 输入 EXINT3 外部中断输入 3 AN3 模拟输入 3 / 超出范围比较器通道 3
P2.4		高阻	T12HR_5 CCU6 定时器 T12 硬件运行输入 T13HR_5 CCU6 定时器 T13 硬件运行输入 T2_3 定时器 T2 输入 AN4 模拟输入 4 / 超出范围比较器通道 4
P2.5		高阻	T12HR_7 CCU6 定时器 T12 硬件运行输入 T13HR_7 CCU6 定时器 T13 硬件运行输入 AN5 模拟输入 5 / 超出范围比较器通道 5

**表 1-1 XC83x 的引脚定义和功能**

符号	类型	复位状态	引脚功能	
P2.6		高阻	SCK_2 EXINT6 AN6	SSC 时钟输入 / 输出 外部中断输入 6 模拟输入 6 / 超出范围比较器通道 6
P2.7		高阻	RXD_6 T2EX_6 MTSR_7 EXINT0_4 AN7	UART 接收输入 定时器 T2 外部触发输入 SSC 从机接收输入 外部中断输入 0 模拟输入 7 / 超出范围比较器通道 7
P3	I/O		<b>P3 口</b> P3 口是通用双向 I/O 口。它还可用作 IIC、LEDTSCU、UART、定时器 T2、SSC、SPD 和 32.768 kHz 晶振的功能引脚。	
P3.0		上拉	SCL_2 SCK_1 EXINT2_1 COL6 XTAL4	IIC 时钟线 SSC 时钟输入 / 输出 外部中断输入 2 LED 列 6 32.768 kHz 外部振荡器输出
P3.1		上拉	RXD_4 RTCCLK MTSR_4 MRST_4 EXINT0_5 COLA_0 XTAL3 EXF2_1	UART 接收输入 RTC 外部时钟输入 SSC 主机发送输出 / 从机接收输入 SSC 主机接收输入 外部中断输入 0 LED 列 A 32.768 kHz 外部振荡器输入 定时器 T2 溢出标志

表 1-1 XC83x 的引脚定义和功能

符号	类型	复位状态	引脚功能
P3.2		上拉	SPD_0      SPD 输入 / 输出 RXD_3      UART 接收输入 / UART BSL 接收输入 SDA_2      IIC 数据线 MTSR_5     SSC 从机接收输入 MRST_5     SSC 主机接收输入 / 从机发送输出 EXINT0_6    外部中断输入 0 T2EX_7      定时器 T2 外部触发输入 TXD_3      UART 发送输出 / 1 线 UART BSL 发送输出
$V_{DDP}$	—	—	I/O 口电源 (2.5 V - 5.5 V)
$V_{DDC}$	—	—	内核电源监控 (2.5 V)
$V_{SSP}/$ $V_{SSC}$	—	—	I/O 口地 / 内核电源地

### 1.3 芯片 ID

每种 XC83x 衍生器件都对应有唯一的芯片 ID，从而便于用户识别不同的器件。根据产品类型、衍生器件类型及器件主要更新信息来分配芯片 ID。

可通过两种方式读取芯片 ID：

- 在应用子程序，见[章节 23.2](#)；
- 引导程序加载（BSL）模式 A，见[章节 6.2.2.8](#)。



## 1.4 命名规则

本手册使用下面的规则来命名 **XC83x** 的组成单元：

- **XC83x** 的功能单元用大写表示。例如：“**SSC** 可用于和移位寄存器通信”。
- 低电平有效的引脚，符号上方加横杠表示。例如：“复位输入引脚 **RESET** 用作硬件复位”。
- 寄存器中的位域和位通常表示为“寄存器名.位域”或“寄存器名.位”。大多数寄存器名包含模块名前缀，用下划线“**\_**”和真正的寄存器名分开。例如“**SSC\_CON**”中“**SSC**”是模块名前缀，“**CON**”才是真正的寄存器名。
- 变量出现在大小写混用中，用来表示一组处理单元或寄存器。例如，寄存器名“**CC6xR**”表示具有变量 **x** (**x** = 0, 1, 2) 的多个“**CC6xR**”寄存器。在首次使用寄存器时给出变量边界（例如“**x** = 0 - 2”），并在必要时重复。
- 数制缺省为十进制。十六进制常数用下标“**H**”表示（如 **C0<sub>H</sub>**）；二进制常数用下标“**B**”表示（如 **11<sub>B</sub>**）。
- 在命名寄存器位域、或者命名一组信号、一组引脚时，表示为“名称 [**A:B**]”，它定义了被命名组从 **B** 到 **A** 的范围。在命名单独的位、信号或引脚时，表示为“名称 [**C**]”，同时给出变量 **C** 的范围（例如 **CFG[2:0]** 和 **TOS[0]**）。
- 单位缩写如下：
  - **MHz** = 兆赫兹
  - **μs** = 微秒
  - **kBaud, kbit** = 每秒 1000 个字符 / 位
  - **MBaud, Mbit** = 每秒 1,000,000 个字符 / 位
  - **Kbyte** = 1024 字节内存
  - **Mbyte** = 1,048,576 字节内存

通常，前缀 **k** 将单位扩大 1000 倍，前缀 **K** 将单位扩大 1024 倍。因此，**Kbyte** 将数值扩大 1024 倍，**kBaud** 将数值扩大 1000 倍。前缀 **M** 将单位扩大 1,000,000 倍或 1048576 倍，**μ** 将单位缩小 0.000001 倍。例如，1 **Kbyte** 是 1024 字节，1 **Mbyte** 是 1024 × 1024 字节，1 **kBaud/kbit** 是每秒 1000 个字符 / 位，1 **MBaud/Mbit** 是每秒 1,000,000 个字符 / 位，1 **MHz** 是 1,000,000 Hz。
- 数据格式定义如下：
  - **Byte** = 8 位

## 1.5 保留位、未定义位及未实现位

定义寄存器位域时，使用下列规则来定义未定义位和未实现位。此外，位和位域读写类型的缩写如 **表 1-2** 定义。

表 1-2 位功能说明

位功能	说明
未实现位	寄存器位域为 “0” 表示该位没有实现功能： 读取这些位域返回 0。 写入这些位域不起作用。 这些位域被保留。软件写入时，应该始终将这些位域设置为 0， 从而和后续产品保持兼容；设置为 1 可能导致不可预测的结果。
Undefined	位域中的一些位元组合起来标记为 “保留”，表示尚未定义 XC83x 在这种位组合下的行为。将寄存器设置为未定义的位组 合，可能导致不可预测的结果。这样的位组合被保留。软件写入 时，必须始终将其设置为位域描述表中提供的合法值。
rw	该位或位域可读写。
r	该位或位域只可读。
w	该位或位域只可写，读取时始终返回 0。
h	该位或位域可由硬件修改（如状态位）。该特性可分别和 “rw” 或 “r” 组合成 “rwh” 和 “rh”。

## 1.6 缩写

表 1-3 列出本手册中使用的缩写。

表 1-3 缩写

缩写	说明
ADC	模数转换器
ALU	算术 / 逻辑单元
BSL	引导程序加载
CAN	控制器局域网
CCU6	捕获 / 比较单元 6
CGU	时钟产生单元
CORDIC	坐标旋转数字计算机
CPU	中央处理单元
DAP	器件访问端口
DNL	差分非线性误差
ECC	纠错码
EVR	嵌入式电压调节器

**表 1-3 缩写**

缩写	说明
FDR	分数分频器
FIFO	先进先出数据缓冲机制
GPIO	通用输入 / 输出
HCPADA	大电流双向端口类型 A
HCPADB	大电流双向端口类型 B
IAP	在应用编程
IIC	内部集成电路总线
I/O	输入 / 输出
INL	积分非线性误差
ISP	在系统编程
JTAG	联合测试行动组
LEDTSCU	LED 和触摸感应控制器单元
LIN	本地互连网络
LSB <sub>n</sub>	最低有效位：以数字格式表示的模拟数值的最小刻度，由 n 位转换精度的最低有效位表示（将测量范围等分为 2 <sup>n</sup> 个区间）
MDU	乘 / 除单元
NMI	非可屏蔽中断
OCDS	片上调试支持
ORC	超出范围比较器
PC	程序计数器
POR	上电复位
PLL	锁相环
PSW	程序状态字
PWM	脉宽调制
RAM	随机存取存储器
ROM	只读存储器
RTC	实时时钟
SCU	系统控制单元
SFR	特殊功能寄存器
SPD	单引脚 DAP
SPI	串行外设接口

表 1-3 缩写

缩写	说明
SSC	同步串行通道
TUE	总不可调整误差
UART	通用异步收发器
WDT	看门狗定时器

## 2 XC800 内核

本章将对 XC800 内核予以说明。

### 2.1 概述

XC800 内核是一个完整的高性能 CPU 内核，其功能与 8051 向上兼容。标准 8051 内核的机器周期由 12 个时钟周期组成，而 XC800 内核的机器周期由 2 个时钟组成。

XC800 内核指令集由 45% 的单字节、41% 的双字节和 14% 的三字节指令组成。每条指令的执行时间为 1、2 或 4 个机器周期。在访问低速存储器时，可通过等待状态延长访问时间（一个等待状态持续一个机器周期，即相当于两个时钟周期）。

XC800 内核可支持的调试特性包括：基本的停止 / 开始，单步执行，断点支持以及对数据存储器、程序存储器和特殊功能寄存器的读写操作。

### 特性

XC800 内核的主要特性如下：

- 2 个时钟的机器周期结构
- 外部数据空间上的片上 XRAM
- 内部数据空间上的 256B IRAM
- 高达 64 KB 代码空间（未完全分配给已实现的存储器）
- 支持同步或异步程序和数据存储器
- 访问低速存储器支持等待状态插入
- 15 个中断源，4 级优先级的中断控制器
- 双数据指针
- 省电模式
- 专用调试模式
- 2 个 16 位定时器（T0 和 T1）
- 全双工串行接口（UART）

### 2.2 XC800 内核功能模块

## XC800 内核

图 2-1 所示为 XC800 内核的功能框图。XC800 内核主要由指令译码器、运算单元、程序控制单元、访问控制单元和中断控制器组成。

指令译码器对每条指令进行译码，所产生的内部信号用于控制内核中的各个模块。这些内部信号控制数据传送和 ALU 操作。

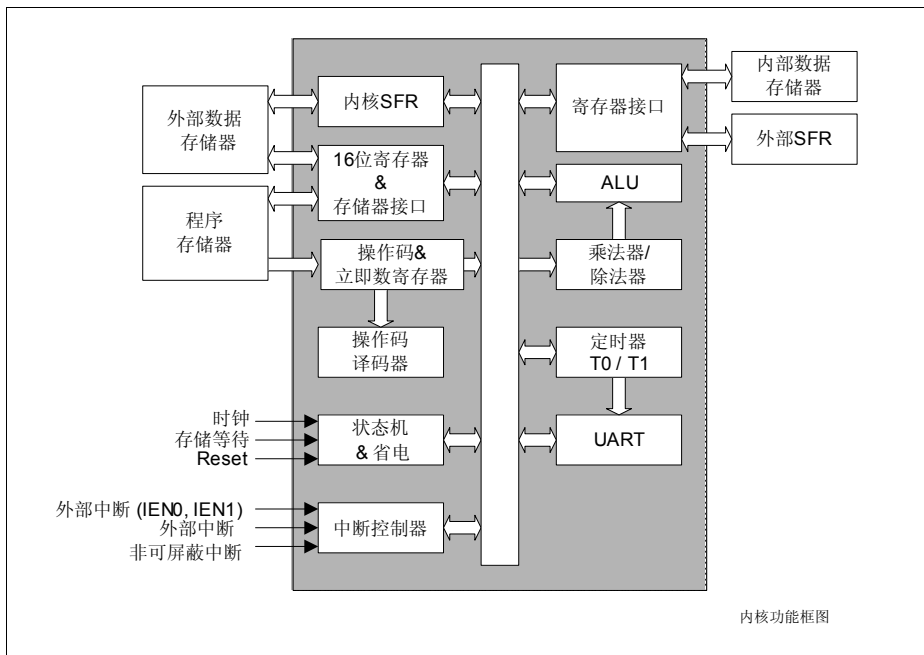


图 2-1 XC800 内核功能框图

处理器的运算单元具有强大的数据处理能力，包括算术逻辑运算单元 (ALU)、A 寄存器、B 寄存器和 PSW 寄存器。来自一个或两个源地址的 8 位数据送至 ALU，在指令译码器的控制下产生 8 位运算结果。ALU 用来完成算术运算和逻辑运算。算术运算包括加、减、乘、除、加 1、减 1、BCD 加法的十进制调整和比较。逻辑运算包括与 (AND)、或 (OR)、异或、补码和循环移位 (右环移、左环移、或 4 位环移 (半字节交换))。ALU 中还包含一个布尔单元，可执行置位、清零、补码、等于 1 跳转、等于 0 跳转、等于 1 跳转并清零，送入 / 取自进位位的位操作。在任意可寻址位 (或该位补码) 和进位标志之间，可执行逻辑与 (AND) 或者逻辑或 (OR) 操作，结果送回进位标志中。

程序控制单元控制程序存储器中指令的执行顺序。16 位程序计数器 (PC) 中保存下一条要执行的指令的地址。条件转移逻辑允许处理器响应内部和外部事件，引起程序执行顺序的改变。

访问控制单元负责选择片上存储资源。中断控制单元负责处理由外设单元产生的中断请求。

## 2.3 CPU SFR

XC800 内核寄存器占用内部数据存储空间的地址段 80<sub>H</sub>-FF<sub>H</sub>。

### 2.3.1 堆栈指针（SP, 81<sub>H</sub>）

堆栈指针寄存器中存放堆栈指针。在执行 LCALL 和 ACALL 指令时，利用 SP 将 PC 的当前值加载到内部数据存储寄存器中；在执行 RET 和 RETI 指令时，从存储器中恢复 PC 的值。还可以通过 PUSH 和 POP 指令在堆栈中保存或从堆栈中恢复数据。压入堆栈前 SP 先自动加 1，出栈后 SP 自动减 1，从而保证了堆栈指针始终指向压入堆栈的最后一个字节，即栈顶。SP 的复位值为 07<sub>H</sub>，所以第一个压入堆栈的数据存储在寄存器 bank 0 之上的地址单元 08<sub>H</sub> 中。SP 的读写操作可由软件控制。编程人员必须确保堆栈的位置和大小不会和其它应用数据重叠。

### 2.3.2 数据指针（DPTR, 82-3<sub>H</sub>）

数据指针（DPTR）存放在寄存器 DPL（数据指针低字节）和 DPH（数据指针高字节）中，构成 16 位地址，用于外部数据存储访问（MOVX A, @DPTR 和 MOVX @DPTR, A）、程序字节转移（MOVC A, @A+DPTR），以及间接程序跳转（JMP @A+DPTR）。数据指针支持两个真正的 16 位操作：载入立即数（MOV DPTR, # data）和加 1（INC DPTR）操作。

### 2.3.3 累加器（ACC, E0<sub>H</sub>）

大多数 ALU 操作中，该寄存器存放其中一个操作数。累加寄存器用 ACC 表示。和累加器相关的指令，其助记符简化为符合“A”。

### 2.3.4 B 寄存器（F0<sub>H</sub>）

乘除法操作时 B 寄存器用来存放第二个操作数，在其它指令中用作暂存寄存器。

### 2.3.5 程序状态字（PSW, D0<sub>H</sub>）

PSW 中存放着多个状态位，指示内核的当前状态。

#### PSW

程序状态字寄存器

(D0<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
CY	AC	F0	RS1	RS0	OV	F1	P
rwh	rwh	rw	rw	rw	rwh	rw	rh

符号	位	类型	描述
<b>P</b>	0	rh	<b>奇偶校验标志</b> 每条指令执行后由硬件置位 / 清零，指示累加器中“1”的个数为奇数 / 偶数，如偶校验。
<b>F1</b>	1	rw	<b>通用标志</b>
<b>OV</b>	2	rwh	<b>溢出标志</b> 用于算术运算指令。
<b>RS0, RS1</b>	3, 4	rw	<b>寄存器组（Register Bank）选择位</b> 由这两位进行 Bank 选择。 00 <sub>B</sub> 选择 Bank 0，数据地址 00 <sub>H</sub> - 07 <sub>H</sub> 01 <sub>B</sub> 选择 Bank 1，数据地址 08 <sub>H</sub> - 0F <sub>H</sub> 10 <sub>B</sub> 选择 Bank 2，数据地址 10 <sub>H</sub> - 17 <sub>H</sub> 11 <sub>B</sub> 选择 Bank 3，数据地址 18 <sub>H</sub> - 1F <sub>H</sub>
<b>F0</b>	5	rw	<b>通用标志</b>
<b>AC</b>	6	rwh	<b>辅助进位标志</b> 用于执行 BCD 操作的指令。
<b>CY</b>	7	rwh	<b>进位标志</b> 用于算术运算指令。



### 2.3.6 扩展操作寄存器 (EO, A2<sub>H</sub>)

指令集中还包括一条指令 **MOVC@(DPTR++), A**，可对由 RAM 实现的程序存储器进行写操作。CPU 初始化时，可用该指令将代码下载到程序存储器中，也可用该指令进行软件更新。该指令将累加器中的内容复制到当前数据指针指向的程序存储器地址中，然后数据指针加 1。

该指令的操作码为 **A5<sub>H</sub>**，和软件暂停指令 **TRAP** 的操作码相同（见表 2-1）。扩展操作（EO）寄存器中的位 **TRAP\_EN** 用于选择操作码 **A5<sub>H</sub>** 所要执行的指令。当 **TRAP\_EN** 为 0（缺省值）时，操作码 **A5<sub>H</sub>** 执行 **MOVC** 指令；当 **TRAP\_EN** 为 1 时，操作码 **A5<sub>H</sub>** 执行软件暂停指令 **TRAP**，该指令将 CPU 切换至调试模式执行断点处理。

寄存器 **EO** 还可用来选择当前的数据指针。

**EO**  
**扩展操作寄存器** (A2<sub>H</sub>) 复位值: 00<sub>H</sub>  
**RMAP: X, PAGE: X**

7	6	5	4	3	2	1	0
	0		TRAP_EN	0	DPSEL2	DPSEL1	DPSEL0
r			rw	r	rw	rw	rw

符号	位	类型	描述
DPSEL0, DPSEL1, DPSEL2	0, 1, 2	rw	<b>数据指针选择</b> 这些位用于选择当前的数据指针。 000 <sub>B</sub> 选择 DPTR0 001 <sub>B</sub> 选择 DPTR1 其它值: 保留
TRAP_EN	4	rw	<b>TRAP 使能位</b> 0 <sub>B</sub> 选择 <b>MOVC @(DPTR++),A</b> 1 <sub>B</sub> 选择软件 <b>TRAP</b> 指令
0	[7:5]	r	<b>保留</b> 读操作返回 0，应写入 0。

### 2.3.7 功率控制寄存器（PCON, 87<sub>H</sub>）

PCON 寄存器用于控制是否进入空闲模式以及 UART 模式 2 的波特率，另外还提供两个通用标志位。

XC800 内核有两种省电模式：空闲模式和掉电模式。控制寄存器 PCON 可进入空闲模式。空闲模式下，内核时钟关闭，但定时器、串口和中断控制器仍正常工作。在掉电模式 1 下，整个 CPU 的时钟全部停止。

**PCON**  
功率控制寄存器 (87<sub>H</sub>) 复位值：00<sub>H</sub>  
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
<b>SMOD</b>		<b>0</b>		<b>GF1</b>	<b>GF0</b>	<b>0</b>	<b>IDLE</b>
rw		r		rw	rw	r	rw

符号	位	类型	描述
<b>IDLE</b>	0	rw	空闲模式使能位 0 <sub>B</sub> 不进入空闲模式 1 <sub>B</sub> 进入空闲模式
<b>GF0</b>	2	rw	通用标志位 0
<b>GF1</b>	3	rw	通用标志位 1
<b>SMOD</b>	7	rw	2 倍波特率使能位 该位控制 UART 模式 2 中的波特率产生。 0 <sub>B</sub> 1 倍波特率 1 <sub>B</sub> 2 倍波特率
<b>0, 0</b>	1, [6:4]	r	保留 读操作返回 0，应写入 0。

### 2.3.8 中断寄存器

共有 1 个非可屏蔽和 14 个可屏蔽中断节点可用。

有关中断寄存器的详细说明，可参见中断一章的描述。

## 2.4 内核外设 SFR

### 2.4.1 定时器寄存器

内核有两个 16 位定时器 - 定时器 T0 和定时器 T1。

有关定时器寄存器的详细说明，请参见定时器 T0 和 T1 一章的描述。

### 2.4.2 UART 寄存器

UART 使用三个 SFR - PCON、SCON 和 SBUF。

有关 UART 寄存器的详细说明，请参见[章节 2.3.7](#) 和 UART 一章的描述。

## 2.5 指令时序

CPU 的机器周期由两个时钟周期组成，分别用拍 1 (P1) 和拍 2 (P2) 表示，对应 CPU 的两个不同状态。执行指令时，CPU 的状态由机器周期和状态编号 (P1/P2) 共同表示，如：C2P1 表示第二个机器周期中的第一个时钟周期。可在机器周期的任一拍、或两拍访问存储器；只能在 P2 结束时对 SFR 进行写操作。指令长度为 1、2 或 3 个字节，执行时间为 1、2 或 4 个机器周期。通常在当条指令的最后一个机器周期的 P2 结束时进行寄存器内容的更新和下一个操作码的预取。

XC800 内核支持访问低速存储器，这通过插入等待周期来实现。每个等待周期的时长为一个机器周期，即两个时钟周期。例如：对于需要一个 / 两个等待状态的存储器而言，在取操作数 / 操作码的每个字节时，访问时间延长一个机器周期。

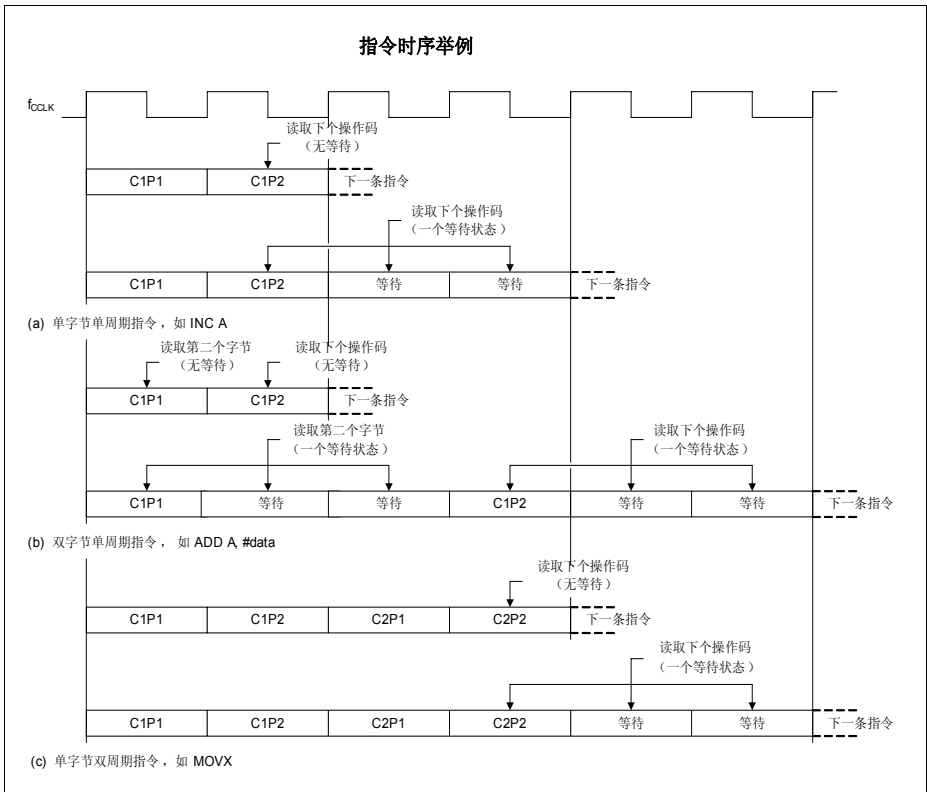
**图 2-2** 给出由 CPU 状态和节拍表示的取指 / 执行时序。每条指令从 C1P1 开始执行，双字节指令第二个字节的读取从 C1P1 开始。

**图 2-2 (a)** 给出两个单字节单周期 ( $1 \times$  机器周期) 指令的时序图。第一个时序图表示下一个操作码 (C1P2) 从无等待状态的存储器读取，指令在一个周期内完成；第二个时序图表示操作码从 Flash 中读取，插入一个等待周期，相同的指令需要两个机器周期完成 (指令时间延长)。

**图 2-2(b)** 给出两个双字节单周期 ( $1 \times$  机器周期) 指令的时序图。第一个时序图表示第二个字节 (C1P1) 和下一个操作码 (C1P2) 从无等待状态的存储器中读取，指令在一个周期内完成；第二个时序图表示每次读取 Flash 时插入一个等待周期 (共插入两个等待周期)，相同的指令需要三个机器周期完成 (指令时间延长)。

**图 2-2 (c)** 给出两个单字节双周期 ( $2 \times$  机器周期) 指令的时序图。第一个时序图表示下一个操作码 (C2P2) 从无等待状态的存储器中读取，指令在两个周期内完成；第二个时序图表示操作码从需要  $1/2$  个等待周期的低速存储器中读取，插入一个等待周期，相同的指令需要三个机器周期完成 (指令时间延长)。

**注：** 对于执行时间超过两个机器周期的指令，在访问需要  $1/2$  个等待周期的低速存储器时，指令的执行周期不一定会延长。不过，需要保证每条指令的执行周期一致。可参见表 2-1。



**图 2-2 CPU 指令时序**

执行每条指令所需的时间包括：

- 译码 / 执行取回的操作码
- 读取操作数（当指令长度超过一个字节时）
- 预取下条指令的首字节（操作码）（CPU 流水线结构）

注：XC800 内核在执行当前指令的同时预取下一条指令的操作码。

**表 2-1** 列出 XC800 内核支持的所有指令。指令长度为 1、2 或 3 个字节（具体见“字节”一列）。每条指令的执行时间为 1、2 或 4 个机器周期（无等待状态）。表中给出执行每条指令所需的机器周期数，共有两个值：第一个值对应从快速存储器（如 Boot ROM 和 XRAM）读取操作数和操作码的情况；第二个值对应从（需要 1/2 个等待状态的）低速存储器（如 Flash）读取操作数和操作码（以及有时访问数据），需要插入等待周期的情况。一个机器周期包含两个 CCLK 时钟周期。

表 2-1 指令一览表

助记符	十六进制 码	字节	机器周期 (无等待状态)	机器周期 (1 个等待状态 <sup>1)</sup> )
算术运算				
ADD A,Rn	28-2F	1	1	2
ADD A,dir	25	2	1	3
ADD A,@Ri	26-27	1	1	2
ADD A,#data	24	2	1	3
ADDC A,Rn	38-3F	1	1	2
ADDC A,dir	35	2	1	3
ADDC A,@Ri	36-37	1	1	2
ADDC A,#data	34	2	1	3
SUBB A,Rn	98-9F	1	1	2
SUBB A,dir	95	2	1	3
SUBB A,@Ri	96-97	1	1	2
SUBB A,#data	94	2	1	3
INC A	04	1	1	2
INC Rn	08-0F	1	1	2
INC dir	05	2	1	3
INC @Ri	06-07	1	1	2
DEC A	14	1	1	2
DEC Rn	18-1F	1	1	2
DEC dir	15	2	1	3
DEC @Ri	16-17	1	1	2
INC DPTR	A3	1	2	2
MUL AB	A4	1	4	4
DIV AB	84	1	4	4
DA A	D4	1	1	2
逻辑运算				
ANL A,Rn	58-5F	1	1	2
ANL A,dir	55	2	1	3
ANL A,@Ri	56-57	1	1	2

表 2-1 指令一览表

助记符	十六进制 码	字节	机器周期 (无等待状态)	机器周期 (1 个等待状态 <sup>1)</sup> )
ANL A,#data	54	2	1	3
ANL dir,A	52	2	1	3
ANL dir,#data	53	3	2	5
ORL A,Rn	48-4F	1	1	2
ORL A,dir	45	2	1	3
ORL A,@Ri	46-47	1	1	2
ORL A,#data	44	2	1	3
ORL dir,A	42	2	1	3
ORL dir,#data	43	3	2	5
XRL A,Rn	68-6F	1	1	2
XRL A,dir	65	2	1	3
XRL A,@Ri	66-67	1	1	2
XRL A,#data	64	2	1	3
XRL dir,A	62	2	1	3
XRL dir,#data	63	3	2	5
CLR A	E4	1	1	2
CPL A	F4	1	1	2
SWAP A	C4	1	1	2
RL A	23	1	1	2
RLC A	33	1	1	2
RR A	03	1	1	2
RRC A	13	1	1	2
数据传送				
MOV A,Rn	E8-EF	1	1	2
MOV A,dir	E5	2	1	3
MOV A,@Ri	E6-E7	1	1	2
MOV A,#data	74	2	1	3
MOV Rn,A	F8-FF	1	1	2
MOV Rn,dir	A8-AF	2	2	4
MOV Rn,#data	78-7F	2	1	3

表 2-1 指令一览表

助记符	十六进制 码	字节	机器周期 (无等待状态)	机器周期 (1 个等待状态 <sup>1)</sup> )
MOV dir,A	F5	2	1	3
MOV dir,Rn	88-8F	2	2	4
MOV dir,dir	85	3	2	5
MOV dir,@Ri	86-87	2	2	4
MOV dir,#data	75	3	2	5
MOV @Ri,A	F6-F7	1	1	2
MOV @Ri,dir	A6-A7	2	2	4
MOV @Ri,#data	76-77	2	1	3
MOV DPTR,#data	90	3	2	5
MOVC A,@A+DPTR	93	1	2	3 or 4 <sup>2)</sup>
MOVC A,@A+PC	83	1	2	3 or 4 <sup>2)</sup>
MOVX A,@Ri	E2-E3	1	2	3
MOVX A,DPTR	E0	1	2	3
MOVX @Ri,A	F2-F3	1	2	3
MOVX @DPTR,A	F0	1	2	3
PUSH dir	C0	2	2	4
POP dir	D0	2	2	4
XCH A,Rn	C8-CF	1	1	2
XCH A,dir	C5	2	1	3
XCH A,@Ri	C6-C7	1	1	2
XCHD A,@Ri	D6-D7	1	1	2
位操作				
CLR C	C3	1	1	2
CLR bit	C2	2	1	3
SETB C	D3	1	1	2
SETB bit	D2	2	1	3
CPL C	B3	1	1	2
CPL bit	B2	2	1	3
ANL C,bit	82	2	2	4
ANL C,/bit	B0	2	2	4



**表 2-1 指令一览表**

助记符	十六进制 码	字节	机器周期 (无等待状态)	机器周期 (1 个等待状态 <sup>1)</sup> )
ORL C,bit	72	2	2	4
ORL C,/bit	A0	2	2	4
MOV C,bit	A2	2	1	3
MOV bit,C	92	2	2	4
<b>控制转移<sup>3)</sup></b>				
ACALL addr11	11->F1	2	2	4
LCALL addr16	12	3	2	5
RET	22	1	2	2 or 3 <sup>2)</sup>
RETI	32	1	2	2 or 3 <sup>2)</sup>
AJMP addr 11	01->E1	2	2	4
LJMP addr 16	02	3	2	5
SJMP rel	80	2	2	4
JC rel	40	2	2	4
JNC rel	50	2	2	4
JB bit,rel	20	3	2	5
JNB bit,rel	30	3	2	5
JBC bit,rel	10	3	2	5
JMP @A+DPTR	73	1	2	2 or 3 <sup>2)</sup>
JZ rel	60	2	2	4
JNZ rel	70	2	2	4
CJNE A,dir,rel	B5	3	2	5
CJNE A,#d,rel	B4	3	2	5
CJNE Rn,#d,rel	B8-BF	3	2	5
CJNE @Ri,#d,rel	B6-B7	3	2	5
DJNZ Rn,rel	D8-DF	2	2	4
DJNZ dir,rel	D5	3	2	5
<b>其它指令</b>				
NOP	00	1	1	2
<b>附加指令</b>				

表 2-1 指令一览表

助记符	十六进制 码	字节	机器周期 (无等待状态)	机器周期 (1 个等待状态 <sup>1)</sup> )
MOVC @(DPTR++),A	A5	1	2	2 or 3 <sup>2)</sup>
TRAP	A5	1	1	—

- 1) 访问只需 1 个等待状态的低速存储器时，有些指令可能不需插入等待周期（通常操作码无需在下个周期进行读取）。
- 2) 这和所访问的存储器的类型（无等待状态、1/2 个等待状态）有关。
- 3) 对于控制转移指令而言，跳转目标不同，指令的执行时间可能不同。

### 3 存储器结构

XC83x 的 CPU 可寻址以下五个地址空间：

- 8 KB Boot ROM 程序存储器
- 256 B 内部 RAM 数据存储器
- 256 XRAM 存储器  
(XRAM 可作为程序存储器或外部数据存储器进行读写)
- 128 B SFR 区
- 4 或 8 KB Flash 程序存储器  
(8 KB 产品中包含一个 64 B 的用户 BSL Flash 扇区)

图 3-1 所示为 8 KB Flash 器件的存储器地址空间分配。

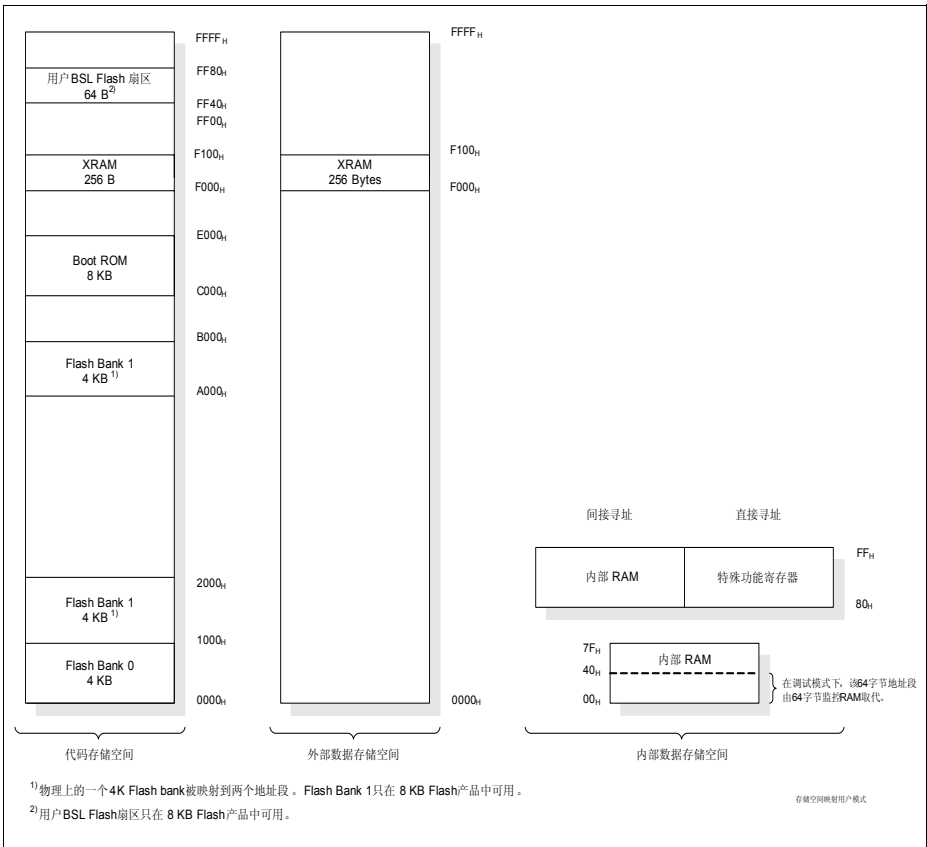


图 3-1 8 KB Flash 器件的存储空间映射

图 3-2 所示为 4 KB Flash 器件的存储器地址空间分配。

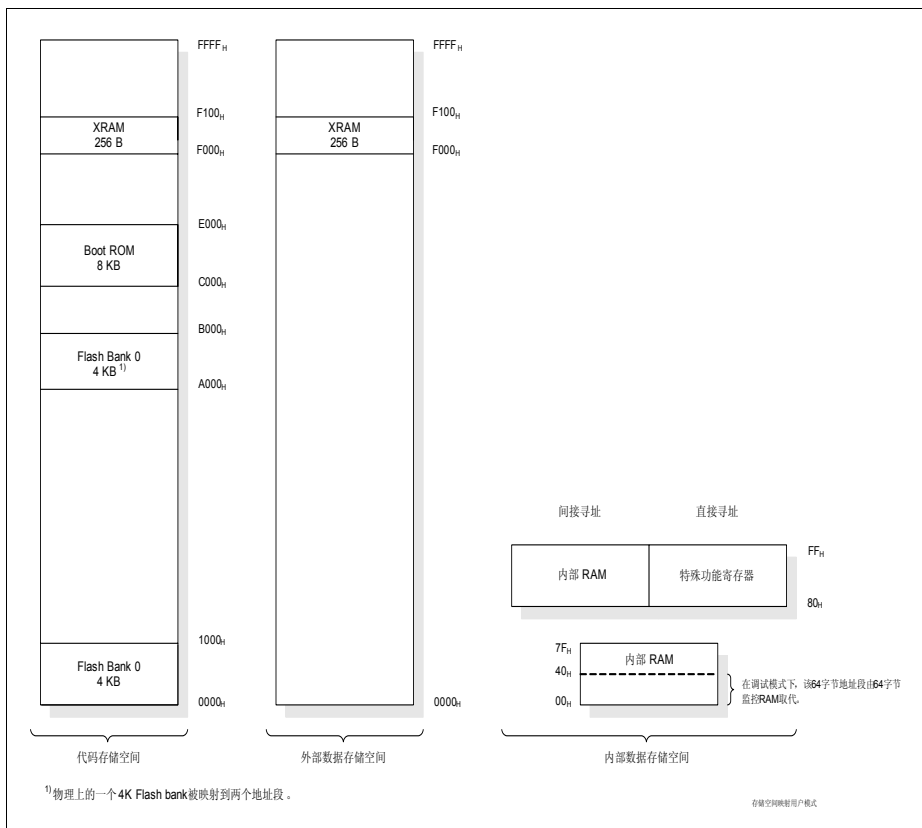


图 3-2 4 KB Flash 器件的存储空间映射

### 3.1 程序存储器

理论上讲，程序存储区的大小为 64 KB。不过，只允许对已定义的程序存储区进行访问（如存储空间映射图所示）。在 XC83x 中，定义的程序存储区由片上存储器占用。

### 3.2 数据存储器

数据存储空间由内部和外部数据存储空间组成。访问内部和外部数据空间时使用的指令操作码不同。在 XC83x 中，片上 XRAM 位于外部数据空间，通过指令 MOVX 进行访问。XC83x 不支持片外存储器访问。内部数据空间由内部 RAM（IRAM）和特殊功能寄存器（SFR）占用，采用不同的寻址方式（直接或间接寻址）访问。

### 3.2.1 内部数据存储器

内部数据存储器被划分为物理上分开的两个不同区域：256 字节 RAM 和 128 字节特殊功能寄存器（SFR）区。RAM 的高 128 字节和 SFR 共用相同的地址段，通过不同的寻址方式访问。RAM 的低 128 字节可通过直接寻址或寄存器间接寻址方式访问；RAM 的高 128 B 只能通过寄存器间接寻址方式访问；SFR 通过直接寻址方式访问。

RAM 地址段 20<sub>H</sub> 到 2F<sub>H</sub> 中的 16 个字节可位寻址；地址段 30<sub>H</sub> 到 7F<sub>H</sub> 可用作暂存寄存器或堆栈。

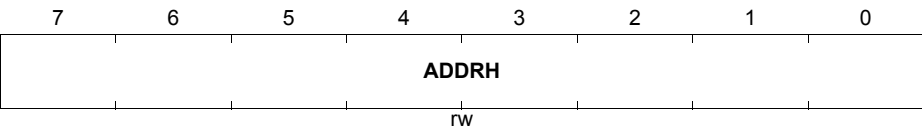
### 3.2.2 外部数据存储器

256 字节 XRAM 被映射到外部数据存储区和程序存储区。可通过 MOVX 和 MOVC 指令访问。

访问 XRAM 的 MOVX 指令使用 8 位或 16 位间接地址。DPTR 寄存器用作 16 位寻址；寄存器 R0 或 R1 用于构成 8 位地址。8 位寻址时，XRAM 地址的高位字节由寄存器 XADDRH 定义。因此，向寄存器 XADDRH 中设置 XRAM 高位地址的写指令必须位于 MOVX 指令之前。

必须在访问 XADDRH 寄存器之前设置 SCU\_PAGE.PAGE。

**XADDRH**  
 片上 XRAM 的高位地址 (F2<sub>H</sub>) 复位值：F0<sub>H</sub>  
 RMAP: 0, PAGE: 3



符号	位	类型	描述
XADDRH	[7:0]	rw	片上 XRAM 的高位地址 在 XC83x 中，该值为 F0 <sub>H</sub> 。

### 3.3 存储器保护策略

XC83x 中的存储器保护策略禁止所有外部访问，从而确保 Flash 中的关键数据和用户 IP 不被非法读取。

通过启动模式索引 / 列表 (BMI) 实现存储器保护：一旦设置 BMI 进入用户模式，只要该 BMI 不被用户程序擦除，则不允许进入其它启动模式。

因此，加载和执行外部程序的启动选择将被阻止，只允许执行从地址 0000<sub>H</sub> 开始的用户程序。

### 3.4 特殊功能寄存器

特殊功能寄存器 (SFR) 占据内部数据存储器的地址段 80<sub>H</sub> - FF<sub>H</sub>。除程序计数器之外，所有的寄存器均位于该 SFR 区。这些 SFR 包括为 CPU 和片上外设提供接口的指针和寄存器。由于内部数据存储区中只可存放 128 个 SFR，小于所需的寄存器总数，因此需要采用地址扩展机制来增加可寻址 SFR 的数目。地址扩展机制包括：

- 映射
- 分页

#### 3.4.1 映射地址扩展

在系统级通过映射进行地址扩展。SFR 区被扩展为两部分：标准（非映射）SFR 区和映射 SFR 区。两个 SFR 区占据相同的地址段 80<sub>H</sub> 到 FF<sub>H</sub>，将可寻址 SFR 的数目扩展至 256 个。选择扩展地址区不由 CPU 指令直接控制，而是由位于地址单元 8F<sub>H</sub> 上的系统控制寄存器 SYSCON0 中的位 RMAP 来控制。置位 SYSCON0 中的 RMAP，控制访问映射 SFR 区；对 RMAP 清零，控制访问标准 SFR 区。SFR 区选择如图 3-3 所示。

只要 RMAP 被置位，即可访问映射 SFR 区。该位不可由硬件自动清零。因此，在访问标准 / 映射寄存器之前，必须由软件分别对 RMAP 清零 / 置位。

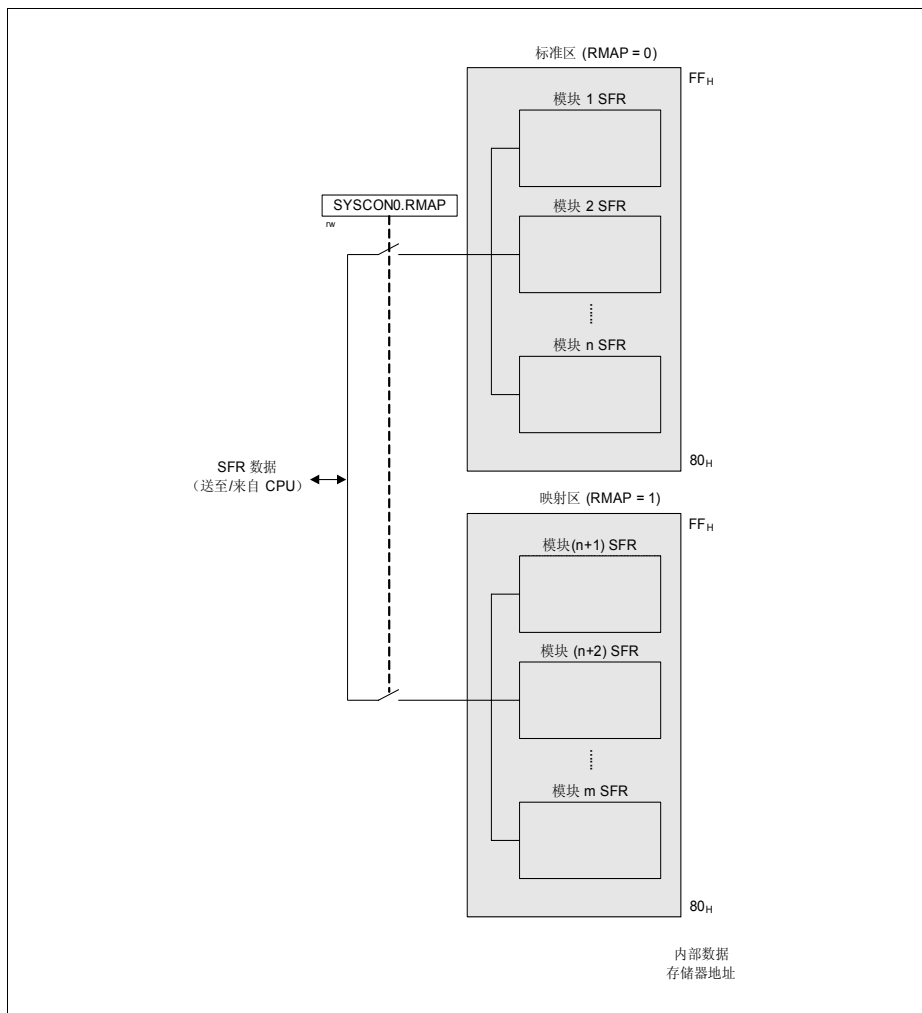
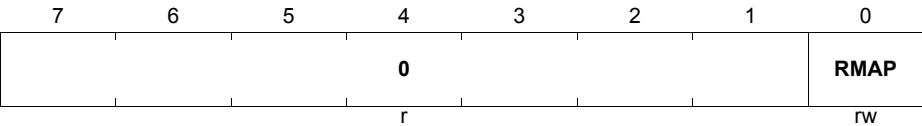


图 3-3 映射地址扩展

3.4.1.1 系统控制寄存器 0

SYSCON0 寄存器用于 SFR 映射控制。

**SYSCON0**  
**系统控制寄存器 0** (8F<sub>H</sub>) **复位值: 00<sub>H</sub>**  
**RMAP: X, PAGE: X**



符号	位	类型	描述
RMAP	0	rw	SFR 映射控制 0 访问非映射 (标准) SFR 区 1 访问映射 SFR 区
0	[7:1]	r	保留 读操作返回 0，应写入 0。



### 3.4.2 分页地址扩展

在模块级通过分页进一步进行地址扩展。映射地址扩展使 XC83x 的 SFR 数目达到 256 个，但即使这样，SFR 的数目仍小于片上外设所需的 SFR 总数。为了满足对 SFR 的需求，某些外设采用内嵌局部地址扩展机制，增加可寻址 SFR 的数目。选择扩展地址区不由 CPU 指令直接控制，而是由模块分页寄存器 MOD\_PAGE 中的位域 PAGE 来控制。因此，在访问和目标模块相关的 SFR 之前，必须先设置位域 PAGE。根据具体要求，每个模块中可能包含的页数不同，每页上 SFR 的个数不同。除了要正确设置 RMAP 值来选择 SFR 区之外，用户还必须确保选择有效的 PAGE 指向所需 SFR。页选择如图 3-4 所示。

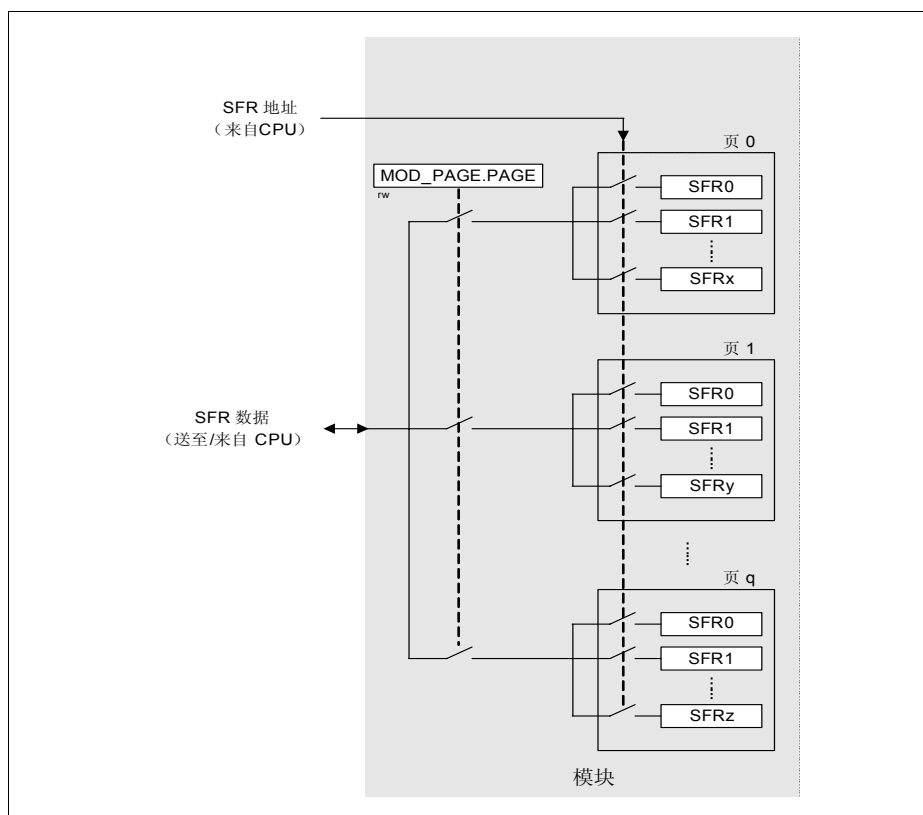


图 3-4 分页地址扩展

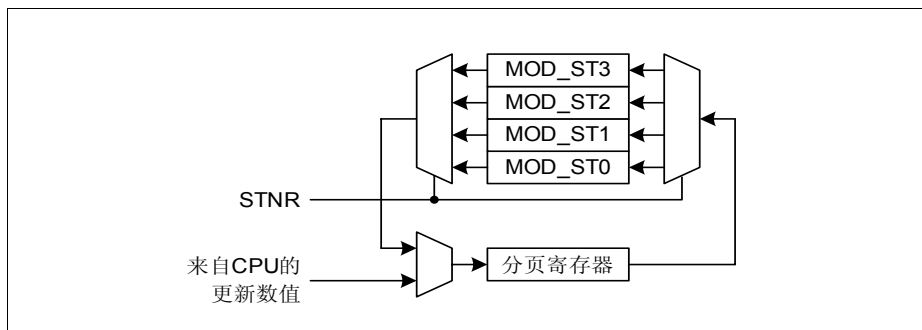
要访问位于（不同于当前页的）其它页上的寄存器，必须退出当前页。这需要重新设置分页寄存器的位域 PAGE，只有这样才能访问到所要的 SFR。

如果在访问分页寄存器和模块寄存器之间开始执行某个中断服务程序，且中断需要访问位于另一页上的寄存器，则需要保存当前页设置，然后设置新页，最后恢复原先页设置。

## 存储器结构

可以用保存域 **MOD\_STx** ( $x = 0 - 3$ ) 来保存和恢复当前页的设置。所有支持该局部地址扩展机制的外设各自均带有一组保存域。例如，**ADC** 的保存域为 **ADC\_STx**。同时指出应使用哪些保存域和新页值，用一次写操作即可完成：

- **PAGE** 中的内容在被新值覆盖之前保存在 **MOD\_STx** 中  
(在中断服务程序开始时保存当前页设置，并设置新页编号)；或
- 用 **MOD\_STx** 中的内容覆盖 **PAGE** 的内容，对写入 **PAGE** 的值不予理睬  
(在中断服务程序结束时恢复中断发生之前的页设置)



**图 3-5 分页机制的页信息存储**

通过这种分页机制，很多中断服务程序（或其它程序）无需读出并保存上次使用的页信息、即可改变页设置。仅用写操作使系统更加简单、高速。这种机制显著改善了小中断服务程序的性能。

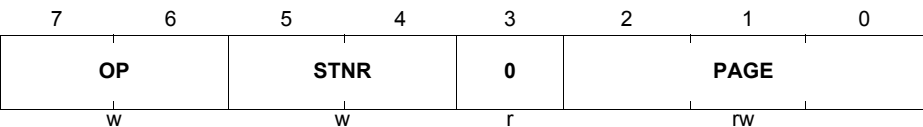
**XC83x** 的以下外设 / 寄存器支持局部地址扩展：

- 并行端口
- 模数转换器（ADC）
- 捕获比较单元 6（CCU6）
- 系统控制寄存器

### 3.4.2.1 分页寄存器

分页寄存器定义如下：

**MOD\_PAGE**  
 模块 MOD 分页寄存器 复位值：00<sub>H</sub>



符号	位	类型	描述
<b>PAGE</b>	[2:0]	rw	<b>页信息</b> 写入时，该值表示新页的值。 读出时，该值表示当前有效页的值。
<b>STNR</b>	[5:4]	w	<b>保存编号</b> 该编号指示在哪个保存位域上执行由 OP 确定的操作。 若 OP = 10 <sub>B</sub> ， PAGE 的内容在被新值覆盖之前保存在 MOD_STx 中。 若 OP = 11 <sub>B</sub> ， PAGE 的内容被 MOD_STx 覆盖。写入 PAGE 的值不予理睬。  00 选择 MOD_ST0 01 选择 MOD_ST1 10 选择 MOD_ST2 11 选择 MOD_ST3
<b>OP</b>	[7:6]	w	<b>操作</b> 0X 手动保存页模式，STNR 的值被忽略，PAGE 被直接写入。 10 带有自动页保存的新页设置。当前写入 PAGE 中的内容被保存的同时，上次写入 PAGE 中的内容被保存在 STNR 指定的位域 MOD_STx 中。 11 自动恢复页操作。对写入 PAGE 的内容不予理睬，PAGE 的内容由 STNR 指定的位域 MOD_STx 中的值覆盖。

**存储器结构**

符号	位	类型	描述
<b>0</b>	<b>3</b>	<b>r</b>	<b>保留</b> 读操作返回 0，应写入 0。

**3.4.3 位寻址**

以格式 1XXXX000<sub>B</sub>（如：80<sub>H</sub>, 88<sub>H</sub>, 90<sub>H</sub>, ..., F0<sub>H</sub>, F8<sub>H</sub>）为地址的 SFR 均为可位寻址寄存器。

### 3.4.4 位保护方案

位保护方案通过 **PASSWD** 寄存器来防止某位由软件直接写入（即该位被保护）。当位域 **MODE** 为 **11<sub>B</sub>**，位域 **PASS** 中写入 **10011<sub>B</sub>**，将开放所有被保护位的访问权限；位域 **PASS** 中写入 **10101<sub>B</sub>**，会关闭所有被保护位的访问权限。在上述两种情况下，即使向 **PASSWD** 寄存器写入 **98<sub>H</sub>** 或 **A8<sub>H</sub>**，位域 **MODE** 的值不改变。向位域 **PASS** 写入 **11000<sub>B</sub>** 时，才能改变位域 **MODE** 的值。例如，向寄存器 **PASSWD** 写 **D0<sub>H</sub>** 会禁止位保护方案。

请注意：如果未写入“关闭访问权限”口令，权限最多开放 **32** 个 **CCLK** 时钟周期。如果在 **32** 个 **CCLK** 时钟周期结束前再次写入“开放访问权限”口令，将重新计数 **32** 个 **CCLK** 周期。被保护位包括：软件复位请求位 **SWRQ**；看门狗定时器使能位 **WDTEN**；RTC 时钟计数寄存器 **CNT0-5** 和掉电使能位 **PD**。

- **RSTCON.SWRQ**：软件复位请求位
- **PMCON0.PD**：掉电使能位
- **WDTCON.WDTEN**：WDT 使能位
- **RTC\_CNTx (x = 0 - 5)**：RTC 时钟计数寄存器中的所有位

必须在访问 **PASSWD** 寄存器之前设置 **SCU\_PAGE.PAGE**。

#### **PASSWD**

口令寄存器

复位值：**07<sub>H</sub>**

**RMAP: 0, PAGE: 1**

7	6	5	4	3	2	1	0
PASS					PROTECT _S	MODE	
wh					rh	rw	

符号	位	类型	描述
<b>MODE</b>	[1:0]	rw	<b>位保护方案控制位</b> <b>00</b> 保护方案禁止 - 允许直接访问被保护位。 <b>11</b> 保护方案使能 - 向位域 <b>PASS</b> 写入密码以开启和关闭对被保护位的访问（缺省值）。 其它：保护方案使能 这两位不能直接写入。要在 <b>11<sub>B</sub></b> 和 <b>00<sub>B</sub></b> 之间切换时，必须将位域 <b>PASS</b> 设置为 <b>11000<sub>B</sub></b> ，只有这样 <b>MODE[1:0]</b> 的值才能被写入。
<b>PROTECT_S</b>	2	rh	<b>位保护信号状态位</b> 该位表明保护状态。 <b>0</b> 软件可以写入所有被保护位。 <b>1</b> 软件不能写入任何被保护位。

**存储器结构**

符号	位	类型	描述
<b>PASS</b>	[7:3]	wh	<b>口令位</b> 位保护方案只能识别以下三个序列。 <b>11000<sub>B</sub></b> 使能设置位域 <b>MODE</b> 。 <b>10011<sub>B</sub></b> 开放所有被保护位的写权限。 <b>10101<sub>B</sub></b> 关闭所有被保护位的写权限。

### 3.4.5 XC83x 寄存器概览

XC83x中的SFR按功能单元分组。SFR内容（位）总结见 [章节 3.4.5.1](#)至[章节 3.4.5.13](#)。

注：可位寻址SFR的地址以黑体标出。

#### 3.4.5.1 CPU 寄存器

CPU SFR 可从标准存储器区和映射存储区访问（RMAP = 0 或 1）。

**表 3-1 CPU 寄存器概览**

地址	寄存器名		位	7	6	5	4	3	2	1	0
RMAP = 0 或 1											
81 <sub>H</sub>	SP 堆栈指针寄存器	复位值: 07 <sub>H</sub>	位域	SP							
			类型	rw							
82 <sub>H</sub>	DPL 数据指针寄存器, 低位字节	复位值: 00 <sub>H</sub>	位域	DPL7	DPL6	DPL5	DPL4	DPL3	DPL2	DPL1	DPL0
			类型	rw	rw	rw	rw	rw	rw	rw	rw
83 <sub>H</sub>	DPH 数据指针寄存器, 高位字节	复位值: 00 <sub>H</sub>	位域	DPH7	DPH6	DPH5	DPH4	DPH3	DPH2	DPH1	DPH0
			类型	rw	rw	rw	rw	rw	rw	rw	rw
87 <sub>H</sub>	PCON 功率控制寄存器	复位值: 00 <sub>H</sub>	位域	SMOD	0			GF1	GF0	0	IDLE
			类型	rw	r			rw	rw	r	rw
88 <sub>H</sub>	TCON 定时器控制寄存器	复位值: 00 <sub>H</sub>	位域	TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
			类型	rwh	rw	rwh	rw	rwh	rw	rwh	rw
89 <sub>H</sub>	TMOD 定时器模式寄存器	复位值: 00 <sub>H</sub>	位域	GATE 1	T1S	T1M		GATE 0	T0S	T0M	
			类型	rw	rw	rw		rw	rw	rw	
8A <sub>H</sub>	TL0 定时器 T0 寄存器, 低位字节	复位值: 00 <sub>H</sub>	位域	VAL							
			类型	rwh							
8B <sub>H</sub>	TL1 定时器 T1 寄存器, 低位字节	复位值: 00 <sub>H</sub>	位域	VAL							
			类型	rwh							
8C <sub>H</sub>	TH0 定时器 T0 寄存器, 高位字节	复位值: 00 <sub>H</sub>	位域	VAL							
			类型	rwh							
8D <sub>H</sub>	TH1 定时器 T1 寄存器, 高位字节	复位值: 00 <sub>H</sub>	位域	VAL							
			类型	rwh							
98 <sub>H</sub>	SCON 串行通道控制寄存器	复位值: 00 <sub>H</sub>	位域	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
			类型	rw	rw	rw	rw	rw	rwh	rwh	rw
99 <sub>H</sub>	SBUF 串行数据缓存寄存器	复位值: 00 <sub>H</sub>	位域	VAL							
			类型	rwh							
A2 <sub>H</sub>	EO 扩展操作寄存器	复位值: 00 <sub>H</sub>	位域	0			TRAP_EN	0	DPSE L2	DPSE L1	DPSE L0
			类型	r			rw	r	rw	rw	rw
A8 <sub>H</sub>	IEN0 中断使能寄存器 0	复位值: 00 <sub>H</sub>	位域	EA	0	ET2	ES	ET1	EX1	ET0	EX0
			类型	rw	r	rw	rw	rw	rw	rw	rw

表 3-1 CPU 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
B8 <sub>H</sub>	<b>IP</b> 中断优先级寄存器 复位值: 00 <sub>H</sub>	位域	0		PT2	PS	PT1	PX1	PT0	PX0
		类型	r		rw	rw	rw	rw	rw	rw
B9 <sub>H</sub>	<b>IPH</b> 中断优先级寄存器, 高位字节 复位值: 00 <sub>H</sub>	位域	0		PT2H	PSH	PT1H	PX1H	PT0H	PX0H
		类型	r		rw	rw	rw	rw	rw	rw
D0 <sub>H</sub>	<b>PSW</b> 程序状态字寄存器 复位值: 00 <sub>H</sub>	位域	CY	AC	F0	RS1	RS0	OV	F1	P
		类型	rwh	rwh	rw	rw	rw	rwh	rw	rh
E0 <sub>H</sub>	<b>ACC</b> 累加寄存器 复位值: 00 <sub>H</sub>	位域	ACC7	ACC6	ACC5	ACC4	ACC3	ACC2	ACC1	ACC0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
E8 <sub>H</sub>	<b>IEN1</b> 中断使能寄存器 1 复位值: 00 <sub>H</sub>	位域	ECCIP 3	ECCIP 2	ECCIP 1	ECCIP 0	EXM	EX2	ESSC	EADC
		类型	rw	rw	rw	rw	rw	rw	rw	rw
F0 <sub>H</sub>	<b>B</b> B 寄存器 复位值: 00 <sub>H</sub>	位域	B7	B6	B5	B4	B3	B2	B1	B0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
F8 <sub>H</sub>	<b>IP1</b> 中断优先级寄存器 1 复位值: 00 <sub>H</sub>	位域	PCCIP 3	PCCIP 2	PCCIP 1	PCCIP 0	PXM	PX2	PSSC	PADC
		类型	rw	rw	rw	rw	rw	rw	rw	rw
F9 <sub>H</sub>	<b>IPH1</b> 中断优先级寄存器 1, 高位字节 复位值: 00 <sub>H</sub>	位域	PCCIP 3H	PCCIP 2H	PCCIP 1H	PCCIP 0H	PXMH	PX2H	PSSC H	PADC H
		类型	rw	rw	rw	rw	rw	rw	rw	rw

### 3.4.5.2 MDU 寄存器

MDU SFR 可从标准存储器区访问 (RMAP = 0)。

表 3-2 MDU 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0	
RMAP = 0											
B0 <sub>H</sub>	MDUSTAT MDU 状态寄存器	复位值: 00 <sub>H</sub>	位域	0					BSY	IERR	IRDY
			类型	r					rh	rwh	rwh
B1 <sub>H</sub>	MDUCON MDU 控制寄存器	复位值: 00 <sub>H</sub>	位域	IE	IR	RSEL	STAR T	OPCODE			
			类型	rw	rw	rw	rwh	rw			
B2 <sub>H</sub>	MD0 MDU 数据寄存器 0	复位值: 00 <sub>H</sub>	位域	DATA							
			类型	rw							
B2 <sub>H</sub>	MR0 MDU 结果寄存器 0	复位值: 00 <sub>H</sub>	位域	DATA							
			类型	rh							
B3 <sub>H</sub>	MD1 MDU 数据寄存器 1	复位值: 00 <sub>H</sub>	位域	DATA							
			类型	rw							
B3 <sub>H</sub>	MR1 MDU 结果寄存器 1	复位值: 00 <sub>H</sub>	位域	DATA							
			类型	rh							



表 3-2 MDU 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
B4 <sub>H</sub>	<b>MD2</b> MDU 数据寄存器 2 复位值: 00 <sub>H</sub>	位域	DATA							
		类型	rw							
B4 <sub>H</sub>	<b>MR2</b> MDU 结果寄存器 2 复位值: 00 <sub>H</sub>	位域	DATA							
		类型	rh							
B5 <sub>H</sub>	<b>MD3</b> MDU 数据寄存器 3 复位值: 00 <sub>H</sub>	位域	DATA							
		类型	rw							
B5 <sub>H</sub>	<b>MR3</b> MDU 结果寄存器 3 复位值: 00 <sub>H</sub>	位域	DATA							
		类型	rh							
B6 <sub>H</sub>	<b>MD4</b> MDU 数据寄存器 4 复位值: 00 <sub>H</sub>	位域	DATA							
		类型	rw							
B6 <sub>H</sub>	<b>MR4</b> MDU 结果寄存器 4 复位值: 00 <sub>H</sub>	位域	DATA							
		类型	rh							
B7 <sub>H</sub>	<b>MD5</b> MDU 数据寄存器 5 复位值: 00 <sub>H</sub>	位域	DATA							
		类型	rw							
B7 <sub>H</sub>	<b>MR5</b> MDU 结果寄存器 5 复位值: 00 <sub>H</sub>	位域	DATA							
		类型	rh							

### 3.4.5.3 CORDIC 寄存器

CORDIC SFR 从标准存储器区访问（RMAP = 0）。

表 3-3 CORDIC 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 1										
BA <sub>H</sub>	<b>CD_CORDXL</b> 复位值: 00 <sub>H</sub> CORDIC X 数据寄存器, 低位字节	位域	DATAL							
		类型	rw							
BB <sub>H</sub>	<b>CD_CORDXH</b> 复位值: 00 <sub>H</sub> CORDIC X 数据寄存器, 高位字节	位域	DATAH							
		类型	rw							
BC <sub>H</sub>	<b>CD_CORDYL</b> 复位值: 00 <sub>H</sub> CORDIC Y 数据寄存器, 低位字节	位域	DATAL							
		类型	rw							
BD <sub>H</sub>	<b>CD_CORDYH</b> 复位值: 00 <sub>H</sub> CORDIC Y 数据寄存器, 高位字节	位域	DATAH							
		类型	rw							
BE <sub>H</sub>	<b>CD_CORDZL</b> 复位值: 00 <sub>H</sub> CORDIC Z 数据寄存器, 低位字节	位域	DATAL							
		类型	rw							
BF <sub>H</sub>	<b>CD_CORDZH</b> 复位值: 00 <sub>H</sub> CORDIC Z 数据寄存器, 高位字节	位域	DATAH							
		类型	rw							
A0 <sub>H</sub>	<b>CD_STATC</b> 复位值: 00 <sub>H</sub> CORDIC 状态和数据控制 寄存器	位域	KEEP Z	KEEP Y	KEEP X	DMAP	INT_E N	EOC	ERRO R	BSY
		类型	rw	rw	rw	rw	rw	rwh	rh	rh
A1 <sub>H</sub>	<b>CD_CON</b> 复位值: 00 <sub>H</sub> CORDIC 控制寄存器	位域	MPS		X_USI GN	ST_M ODE	ROTV EC	MODE		ST
		类型	rw		rw	rw	rw	rw		rwh

### 3.4.5.4 系统控制寄存器

系统控制 SFR 从标准存储器区访问（RMAP = 0）。

表 3-4 SCU 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 0 或 1										
8F <sub>H</sub>	SYSCON0 复位值: 00 <sub>H</sub> 系统控制寄存器 0	位域	0							RMAP
		类型	r							rw
RMAP = 0										
F1 <sub>H</sub>	SCU_PAGE 复位值: 00 <sub>H</sub> 系统控制分页寄存器	位域	OP		STNR		0	PAGE		
		类型	w		w		r	rw		
RMAP = 0, PAGE 0										

**表 3-4 SCU 寄存器概览 (cont'd)**

地址	寄存器名	位	7	6	5	4	3	2	1	0
EE <sub>H</sub>	<b>NMICON</b> 复位值：00 <sub>H</sub> NMI 控制寄存器	位域	NMI XTAL CLK	NMI ECC	NMI VDDP	NMI VDDC	NMI OCDS	NMI FLASH	NMI OSC CLK	NMI WDT
		类型	rw	rw	rw	rw	rw	rw	rw	rw
EF <sub>H</sub>	<b>EXICON0</b> 复位值：F0 <sub>H</sub> 外部中断控制寄存器 0	位域	EXINT3		EXINT2		EXINT1		EXINT0	
		类型	rw		rw		rw		rw	
F2 <sub>H</sub>	<b>IRCON0</b> 复位值：00 <sub>H</sub> 中断请求寄存器 0	位域	0		EXINT 5	EXINT 4	EXINT 3	EXINT 2	0	
		类型	r		rwh	rwh	rwh	rwh	r	
F3 <sub>H</sub>	<b>IRCON1</b> 复位值：00 <sub>H</sub> 中断请求寄存器 1	位域	0			ADCS R1	ADCS R0	RIR	TIR	EIR
		类型	r			rwh	rwh	rwh	rwh	rwh
F4 <sub>H</sub>	<b>EXICON1</b> 复位值：3F <sub>H</sub> 外部中断控制寄存器 1	位域	0				EXINT5		EXINT4	
		类型	r				rw		rw	
F5 <sub>H</sub>	<b>IRCON2</b> 复位值：00 <sub>H</sub> 中断请求寄存器 2	位域	0			CCU6 SR1	0			CCU6 SR0
		类型	r			rwh	r			rwh
F6 <sub>H</sub>	<b>IRCON3</b> 复位值：00 <sub>H</sub> 中断请求寄存器 3	位域	0			CCU6 SR3	0			CCU6 SR2
		类型	r			rwh	r			rwh
F7 <sub>H</sub>	<b>NMISR</b> 复位值：00 <sub>H</sub> NMI 状态寄存器	位域	FNMI XTAL CLK	FNMI ECC	FNMI VDDP	FNMI VDDC	FNMI OCDS	FNMI FLASH	FNMI OSC CLK	FNMI WDT
		类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
RMAP = 0, PAGE 1										
EE <sub>H</sub>	<b>SDCON</b> 复位值：34 <sub>H</sub> 电源检测控制寄存器	位域	0		VDDP TH	VDDC TH	VDDP BOBP	VDDP BOA	VDDP PW	VDDC PW
		类型	r		rh	rh	rw	rw	rw	rw
EF <sub>H</sub>	<b>PMCON1</b> 复位值：00 <sub>H</sub> 功率模式控制寄存器 1	位域	IIC_ DIS	LTS_ DIS	CDC_ DIS	MDU_ DIS	T2_ DIS	CCU_ DIS	SSC_ DIS	ADC_ DIS
		类型	rw	rw	rw	rw	rw	rw	rw	rw
F2 <sub>H</sub>	<b>PASSWD</b> 复位值：07 <sub>H</sub> 口令寄存器	位域	PASS					PROT ECT_ S	MODE	
		类型	wh					rh	rw	
F3 <sub>H</sub>	<b>PMCON0</b> 复位值：01 <sub>H</sub> 功率模式控制寄存器 0	位域	0			WK SEL	PD MODE		PD	EWS
		类型	r			rw	rw		rwh	rw
F4 <sub>H</sub>	<b>OSC_CON</b> 复位值：30 <sub>H</sub> OSC 控制寄存器	位域	0	INT OSC_ ST	XPD	XTAL 2L	XTAL OWD RST	75K OSC 2L	48M OSC 2L	RC OWD RST
		类型	r	rh	rh	rh	rwh	rh	rh	rwh
F5 <sub>H</sub>	<b>ID</b> 复位值：UU <sub>H</sub> ID 寄存器	位域	PRODID					VERID		
		类型	r					r		

**表 3-4 SCU 寄存器概览 (cont'd)**

地址	寄存器名	位	7	6	5	4	3	2	1	0
F6 <sub>H</sub>	<b>WDTCON</b> 复位值: 00 <sub>H</sub> 看门狗定时器控制寄存器	位域	0		WINB EN	WDT PR	0	WDT EN	WDT RS	0
		类型	r		rw	rh	r	rw	rwh	r
F7 <sub>H</sub>	<b>RSTCON</b> 复位值: 00 <sub>H</sub> 复位控制寄存器	位域	SWRQ	0				SOFT RS	WDT RST	WKR S
		类型	rwh	r				rwh	rwh	rwh
RMAP = 0, PAGE 3										
EE <sub>H</sub>	<b>MODPISEL3</b> 复位值: 00 <sub>H</sub> 外设输入选择寄存器 3	位域	IST13HR1			IST12HR1			CTR APIS	
		类型	rw			rw			rw	
F2 <sub>H</sub>	<b>XADDRH</b> 复位值: F0 <sub>H</sub> 片上 XRAM 高位地址	位域	ADDRH							
		类型	rw							
F3 <sub>H</sub>	<b>MODPISEL</b> 复位值: 00 <sub>H</sub> 外设输入选择寄存器	位域	CIS		SIS			MIS		
		类型	rw		rw			rw		
F4 <sub>H</sub>	<b>MODPISEL1</b> 复位值: 00 <sub>H</sub> 外设输入选择寄存器 1	位域	EXINT 2IS	EXINT 1IS	EXINT0IS			URRIS		
		类型	rw	rw	rw			rw		
F5 <sub>H</sub>	<b>MODPISEL2</b> 复位值: 00 <sub>H</sub> 外设输入选择寄存器 2	位域	0	T0IS	T1IS	T2IS		T2EXIS		
		类型	r	rw	rw	rw		rw		
F6 <sub>H</sub>	<b>MODSUSP</b> 复位值: 01 <sub>H</sub> 模块挂起控制寄存器	位域	0		LTS SUSP	RTC SUSP	T2SUS P	T13SU SP	T12SU SP	WDT SUSP
		类型	r		rw	rw	rw	rw	rw	rw
F7 <sub>H</sub>	<b>MODIEN</b> 复位值: 07 <sub>H</sub> 外设中断使能寄存器	位域	CCU6 SR3 EN	CCU6 SR2 EN	0			RIREN	TIREN	EIREN
		类型	rw	rw	r			rw	rw	rw
RMAP = 0, PAGE 4										
F3 <sub>H</sub>	<b>WDTREL</b> 复位值: 00 <sub>H</sub> 看门狗定时器重载寄存器	位域	WDTREL							
		类型	rw							
F4 <sub>H</sub>	<b>WDTWINB</b> 复位值: 00 <sub>H</sub> 看门狗窗界计数器寄存器	位域	WDTWINB							
		类型	rw							
F5 <sub>H</sub>	<b>WDTL</b> 复位值: 00 <sub>H</sub> 看门狗定时器寄存器, 低位字节	位域	WDT							
		类型	rh							
F6 <sub>H</sub>	<b>WDTH</b> 复位值: 00 <sub>H</sub> 看门狗定时器寄存器, 高位字节	位域	WDT							
		类型	rh							
RMAP = 0, PAGE 5										
F2 <sub>H</sub>	<b>BCON</b> 复位值: 00 <sub>H</sub> 波特率控制寄存器	位域	BGSEL	0	BRDIS	BRPRE			R	
		类型	rw	r	rw	rw			rw	
F3 <sub>H</sub>	<b>BGL</b> 复位值: 00 <sub>H</sub> 波特率定时器 / 重载寄存器, 低位字节	位域	BR_VALUE			FDSEL				
		类型	rwh			rw				

表 3-4 SCU 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
F4 <sub>H</sub>	<b>BGH</b> 复位值: 00 <sub>H</sub> 波特率定时器 / 重载寄存器, 高位字节	位域	BR_VALUE							
		类型	rwh							
F5 <sub>H</sub>	<b>LINST</b> 复位值: 00 <sub>H</sub> LIN 状态寄存器	位域	BGS	SYNE N	ERRS YN	EOFS YN	BRK	0		
		类型	rw	rw	rwh	rwh	rwh	r		
F6 <sub>H</sub>	<b>FEAL</b> 复位值: 00 <sub>H</sub> Flash 错误地址寄存器, 低位字节	位域	ECCERRADDR							
		类型	rh							
F7 <sub>H</sub>	<b>FEAH</b> 复位值: 00 <sub>H</sub> Flash 错误地址寄存器, 高位字节	位域	ECCERRADDR							
		类型	rh							

### 3.4.5.5 端口寄存器

端口 SFR 从标准存储器区访问 (RMAP = 0)。

表 3-5 端口寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 0										
8E <sub>H</sub>	<b>PORT_PAGE</b> 复位值: 00 <sub>H</sub> 端口分页寄存器	位域	OP		STNR		0	PAGE		
		类型	w		w		r	rw		
RMAP = 0, PAGE 0										
80 <sub>H</sub>	<b>P0_DATAOUT</b> 复位值: FF <sub>H</sub> P0 口数据输出寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
86 <sub>H</sub>	<b>P0_DATAIN</b> 复位值: UU <sub>H</sub> P0 口数据输入寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rh	rh	rh	rh	rh	rh	rh	rh
90 <sub>H</sub>	<b>P1_DATA</b> 复位值: 3F <sub>H</sub> P1 口数据寄存器	位域	0		P5	P4	P3	P2	P1	P0
		类型	r		rw	rw	rw	rw	rw	rw
91 <sub>H</sub>	<b>P1_DATAIN</b> 复位值: UU <sub>H</sub> P1 口数据输入寄存器	位域	0		P5	P4	P3	P2	P1	P0
		类型	r		rh	rh	rh	rh	rh	rh
92 <sub>H</sub>	<b>P1_OCD</b> 复位值: 00 <sub>H</sub> P1 口过流检测寄存器	位域	P3_O CEN	P2_O CEN	P1_O CEN	P0_O CEN	P3_O CF	P2_O CF	P1_O CF	P0_O CF
		类型	rw	rw	rw	rw	rwh	rwh	rwh	rwh
94 <sub>H</sub>	<b>P2_DATAIN</b> 复位值: UU <sub>H</sub> P2 口数据输入寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rh	rh	rh	rh	rh	rh	rh	rh
C8 <sub>H</sub>	<b>P3_DATAOUT</b> 复位值: 07 <sub>H</sub> P3 口数据输出寄存器	位域	0					P2	P1	P0
		类型	r					rw	rw	rw
C9 <sub>H</sub>	<b>P3_DATAIN</b> 复位值: UU <sub>H</sub> P3 口数据输入寄存器	位域	0					P2	P1	P0
		类型	r					rw	rw	rw
RMAP = 0, PAGE 1										

**表 3-5 端口寄存器概览 (cont'd)**

地址	寄存器名	位	7	6	5	4	3	2	1	0
80 <sub>H</sub>	<b>P0_PUDSEL</b> 复位值: EF <sub>H</sub> P0 口上拉 / 下拉选择寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
86 <sub>H</sub>	<b>P0_PUDEN</b> 复位值: C4 <sub>H</sub> P0 口上拉 / 下拉使能寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
90 <sub>H</sub>	<b>P1_PUDSEL</b> 复位值: 3F <sub>H</sub> P1 口上拉 / 下拉选择寄存器	位域	0		P5	P4	P3	P2	P1	P0
		类型	r		rw	rw	rw	rw	rw	rw
91 <sub>H</sub>	<b>P1_PUDEN</b> 复位值: 00 <sub>H</sub> P1 口上拉 / 下拉使能寄存器	位域	0		P5	P4	P3	P2	P1	P0
		类型	r		rw	rw	rw	rw	rw	rw
92 <sub>H</sub>	<b>P1_OCPED</b> 复位值: 00 <sub>H</sub> P1 口过流检测使能寄存器	位域	0				P3	P2	P1	P0
		类型	r				rw	rw	rw	rw
93 <sub>H</sub>	<b>P2_PUDSEL</b> 复位值: FF <sub>H</sub> P2 口上拉 / 下拉选择寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
94 <sub>H</sub>	<b>P2_PUDEN</b> 复位值: 00 <sub>H</sub> P2 口上拉 / 下拉使能寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
C8 <sub>H</sub>	<b>P3_PUDSEL</b> 复位值: BF <sub>H</sub> P3 口上拉 / 下拉选择寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
C9 <sub>H</sub>	<b>P3_PUDEN</b> 复位值: 40 <sub>H</sub> P3 口上拉 / 下拉使能寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
RMAP = 0, PAGE 2										
80 <sub>H</sub>	<b>P0_ALTSEL0</b> 复位值: 00 <sub>H</sub> P0 口复用功能选择寄存器 0	位域	0	P6	P5	P4	P3	P2	P1	P0
		类型	r	rw	rw	rw	rw	rw	rw	rw
85 <sub>H</sub>	<b>P0_ALTSEL2</b> 复位值: 00 <sub>H</sub> P0 口复用功能选择寄存器 2	位域	0	P6	P5	P4	0			
		类型	r	rw	rw	rw	r			
86 <sub>H</sub>	<b>P0_ALTSEL1</b> 复位值: 00 <sub>H</sub> P0 口复用功能选择寄存器 1	位域	0	P6	P5	P4	P3	P2	P1	P0
		类型	r	rw	rw	rw	rw	rw	rw	rw
90 <sub>H</sub>	<b>P1_ALTSEL0</b> 复位值: 00 <sub>H</sub> P1 口复用功能选择寄存器 0	位域	0		P5	P4	P3	P2	P1	P0
		类型	r		rw	rw	rw	rw	rw	rw
91 <sub>H</sub>	<b>P1_ALTSEL1</b> 复位值: 00 <sub>H</sub> P1 口复用功能选择寄存器 1	位域	0		P5	P4	P3	P2	P1	P0
		类型	r		rw	rw	rw	rw	rw	rw
C8 <sub>H</sub>	<b>P3_ALTSEL0</b> 复位值: 00 <sub>H</sub> P3 口复用功能选择寄存器 0	位域	0				P2		P1	P0
		类型	r				rw		rw	rw
C9 <sub>H</sub>	<b>P3_ALTSEL1</b> 复位值: 00 <sub>H</sub> P3 口复用功能选择寄存器 1	位域	0				P2		P1	P0
		类型	r				rw		rw	rw
RMAP = 0, PAGE 3										
80 <sub>H</sub>	<b>P0_OD</b> 复位值: FF <sub>H</sub> P0 口漏级开路输出控制寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
90 <sub>H</sub>	<b>P1_OD</b> 复位值: 3F <sub>H</sub> P1 口漏级开路输出控制寄存器	位域	0		P5	P4	P3	P2	P1	P0
		类型	r		rw	rw	rw	rw	rw	rw

**表 3-5 端口寄存器概览 (cont'd)**

地址	寄存器名	位	7	6	5	4	3	2	1	0
92 <sub>H</sub>	<b>P1_SLEW</b> 复位值: 00 <sub>H</sub> P1 口翻转速度控制寄存器	位域	0				P3	P2	P1	P0
		类型	r				rw	rw	rw	rw
94 <sub>H</sub>	<b>P2_EN</b> 复位值: 00 <sub>H</sub> P2 口使能寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
C8 <sub>H</sub>	<b>P3_OD</b> 复位值: 07 <sub>H</sub> P3 口漏级开路输出控制寄存器	位域	0					P2	P1	P0
		类型	r					rw	rw	rw

### 3.4.5.6 ADC 寄存器

ADC SFR 从标准存储器区访问 (RMAP = 0)。

**表 3-6 ADC 寄存器概览**

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 0										
D1 <sub>H</sub>	ADC_PAGE 复位值: 00 <sub>H</sub> ADC 分页寄存器	位域	OP		STNR		0	PAGE		
		类型	w		w		r	rw		
RMAP = 0, PAGE 0										
CA <sub>H</sub>	ADC_GLOBCTR 复位值: 30 <sub>H</sub> 全局控制寄存器	位域	ANON	DW	CTC		ORCIE N	CLCIE N	0	
		类型	rw	rw	rw		rw	rw	r	
CB <sub>H</sub>	ADC_GLOBSTR 复位值: 00 <sub>H</sub> 全局状态寄存器	位域	0			CHNR		0	SAMP LE	BUSY
		类型	r			rh		r	rh	rh
CC <sub>H</sub>	ADC_PRAR 复位值: 00 <sub>H</sub> 优先级和仲裁寄存器	位域	ASEN 1	ASEN 0	0	ARBM	CSM1	PRI01	CSM0	PRI00
		类型	rw	rw	r	rw	rw	rw	rw	rw
CD <sub>H</sub>	ADC_LCBR0 复位值: 70 <sub>H</sub> 边界检测寄存器 0	位域	BOUND0							
		类型	rw							
CE <sub>H</sub>	ADC_INPCR0 复位值: 00 <sub>H</sub> 输入综合寄存器 0	位域	0				STC			
		类型	r				rw			
CF <sub>H</sub>	ADC_LCBR1 复位值: B0 <sub>H</sub> 边界检测寄存器 1	位域	BOUND1							
		类型	rw							
D2 <sub>H</sub>	ADC_LORE 复位值: 00 <sub>H</sub> 锁存超出范围事件寄存器	位域	LORE 7	LORE 6	LORE 5	LORE 4	LORE 3	LORE 2	LORE 1	LORE 0
		类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
D3 <sub>H</sub>	ADC_ENORC 复位值: 00 <sub>H</sub> 使能超出范围比较器寄存器	位域	ENOR C7	ENOR C6	ENOR C5	ENOR C4	ENOR C3	ENOR C2	ENOR C1	ENOR C0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
RMAP = 0, PAGE 1										
CA <sub>H</sub>	ADC_CHCTR0 复位值: 00 <sub>H</sub> 通道控制寄存器 0	位域	BFEN	LCC			REFSEL		RESRSEL	
		类型	rw	rw			rw		rw	

表 3-6 ADC 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
CB <sub>H</sub>	ADC_CHCTR1 复位值: 00 <sub>H</sub> 通道控制寄存器 1	位域	BFEN	LCC			REFSEL		RESRSEL	
		类型	rw	rw			rw		rw	
CC <sub>H</sub>	ADC_CHCTR2 复位值: 00 <sub>H</sub> 通道控制寄存器 2	位域	BFEN	LCC			REFSEL		RESRSEL	
		类型	rw	rw			rw		rw	
CD <sub>H</sub>	ADC_CHCTR3 复位值: 00 <sub>H</sub> 通道控制寄存器 3	位域	0	LCC			REFSEL		RESRSEL	
		类型	r	rw			rw		rw	
CE <sub>H</sub>	ADC_CHCTR4 复位值: 00 <sub>H</sub> 通道控制寄存器 4	位域	0	LCC			REFSEL		RESRSEL	
		类型	r	rw			rw		rw	
CF <sub>H</sub>	ADC_CHCTR4 复位值: 00 <sub>H</sub> 通道控制寄存器 5	位域	0	LCC			REFSEL		RESRSEL	
		类型	r	rw			rw		rw	
D2 <sub>H</sub>	ADC_CHCTR4 复位值: 00 <sub>H</sub> 通道控制寄存 6	位域	0	LCC			REFSEL		RESRSEL	
		类型	r	rw			rw		rw	
D3 <sub>H</sub>	ADC_CHCTR4 复位值: 00 <sub>H</sub> 通道控制寄存器 7	位域	0	LCC			REFSEL		RESRSEL	
		类型	r	rw			rw		rw	
RMAP = 0, PAGE 2										
CA <sub>H</sub>	ADC_RESR0L 复位值: 00 <sub>H</sub> 结果寄存器 0, 低位字节	位域	RESULT				VF	DRC	CHNR	
		类型	rh				rh	rh	rh	
CB <sub>H</sub>	ADC_RESR0H 复位值: 00 <sub>H</sub> 结果寄存器 0, 高位字节	位域	RESULT							
		类型	rh							
CC <sub>H</sub>	ADC_RESR1L 复位值: 00 <sub>H</sub> 结果寄存器 1, 低位字节	位域	RESULT				VF	DRC	CHNR	
		类型	rh				rh	rh	rh	
CD <sub>H</sub>	ADC_RESR1H 复位值: 00 <sub>H</sub> 结果寄存器 1, 高位字节	位域	RESULT							
		类型	rh							
CE <sub>H</sub>	ADC_RESR2L 复位值: 00 <sub>H</sub> 结果寄存器 2, 低位字节	位域	RESULT				VF	DRC	CHNR	
		类型	rh				rh	rh	rh	
CF <sub>H</sub>	ADC_RESR2H 复位值: 00 <sub>H</sub> 结果寄存器 2, 高位字节	位域	RESULT							
		类型	rh							
D2 <sub>H</sub>	ADC_RESR3L 复位值: 00 <sub>H</sub> 结果寄存器 3, 低位字节	位域	RESULT				VF	DRC	CHNR	
		类型	rh				rh	rh	rh	
D3 <sub>H</sub>	ADC_RESR3H 复位值: 00 <sub>H</sub> 结果寄存器 3, 高位字节	位域	RESULT							
		类型	rh							
RMAP = 0, PAGE 4										
CA <sub>H</sub>	ADC_RCR0 复位值: 00 <sub>H</sub> 结果控制寄存器 0	位域	VFCT R	WFR	0	IEN	0	DLPF	0	DRCT R
		类型	rw	rw	r	rw	r	rw	r	rw
CB <sub>H</sub>	ADC_RCR1 复位值: 00 <sub>H</sub> 结果控制寄存器 1	位域	VFCT R	WFR	0	IEN	0	DLPF	0	DRCT R
		类型	rw	rw	r	rw	r	rw	r	rw



**表 3-6 ADC 寄存器概览 (cont'd)**

地址	寄存器名	位	7	6	5	4	3	2	1	0
CC <sub>H</sub>	<b>ADC_RCR2</b> 复位值: 00 <sub>H</sub> 结果控制寄存器 2	位域	VFCT R	WFR	0	IEN	0	DLPF	0	DRCT R
		类型	rw	rw	r	rw	r	rw	r	rw
CD <sub>H</sub>	<b>ADC_RCR3</b> 复位值: 00 <sub>H</sub> 结果控制寄存器 3	位域	VFCT R	WFR	0	IEN	0	DLPF	0	DRCT R
		类型	rw	rw	r	rw	r	rw	r	rw
CE <sub>H</sub>	<b>ADC_VFCR</b> 复位值: 00 <sub>H</sub> 有效标志清零寄存器	位域	0				VFC3	VFC2	VFC1	VFC0
		类型	r				w	w	w	w
CF <sub>H</sub>	<b>ADC_ALR0</b> 复位值: 00 <sub>H</sub> 假名寄存器 0	位域	0					ALIAS0		
		类型	r					rw		
D2 <sub>H</sub>	<b>ADC_CNF</b> 复位值: 00 <sub>H</sub> 配置超出范围比较寄存器	位域	CNF7	CNF6	CNF5	CNF4	CNF3	CNF2	CNF1	CNF0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
D3 <sub>H</sub>	<b>ADC_ETRCR</b> 复位值: 00 <sub>H</sub> 外部触发控制寄存器	位域	0		ETRSEL1			ETRSEL0		
		类型	r		rw			rw		
RMAP = 0, PAGE 5										
CA <sub>H</sub>	<b>ADC_CHINFR</b> 复位值: 00 <sub>H</sub> 通道中断标志寄存器	位域	CHINF 7	CHINF 6	CHINF 5	CHINF 4	CHINF 3	CHINF 2	CHINF 1	CHINF 0
		类型	rh	rh	rh	rh	rh	rh	rh	rh
CB <sub>H</sub>	<b>ADC_CHINCR</b> 复位值: 00 <sub>H</sub> 通道中断清零寄存器	位域	CHINC 7	CHINC 6	CHINC 5	CHINC 4	CHINC 3	CHINC 2	CHINC 1	CHINC 0
		类型	w	w	w	w	w	w	w	w
CC <sub>H</sub>	<b>ADC_CHINSR</b> 复位值: 00 <sub>H</sub> 通道中断置位寄存器	位域	CHINS 7	CHINS 6	CHINS 5	CHINS 5	CHINS 3	CHINS 2	CHINS 1	CHINS 0
		类型	w	w	w	w	w	w	w	w
CE <sub>H</sub>	<b>ADC_EVINFR</b> 复位值: 00 <sub>H</sub> 事件中中断标志寄存器	位域	EVINF 7	EVINF 6	EVINF 5	EVINF 4	0		EVINF 1	EVINF 0
		类型	rh	rh	rh	rh	r		rh	rh
CF <sub>H</sub>	<b>ADC_EVINCR</b> 复位值: 00 <sub>H</sub> 事件中中断清零标志寄存器	位域	EVINC 7	EVINC 6	EVINC 5	EVINC 4	0		EVINC 1	EVINC 0
		类型	w	w	w	w	r		w	w
D2 <sub>H</sub>	<b>ADC_EVINSR</b> 复位值: 00 <sub>H</sub> 事件中中断置位标志寄存器	位域	EVINS 7	EVINS 6	EVINS 5	EVINS 4	0		EVINS 1	EVINS 0
		类型	w	w	w	w	r		w	w
RMAP = 0, PAGE 6										
CA <sub>H</sub>	<b>ADC_CRCCR1</b> 复位值: 00 <sub>H</sub> 转换请求控制寄存器 1	位域	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
		类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
CB <sub>H</sub>	<b>ADC_CRPR1</b> 复位值: 00 <sub>H</sub> 转换请求挂起寄存器 1	位域	CHP7	CHP6	CHP5	CHP4	CHP3	CHP2	CHP1	CHP0
		类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
CC <sub>H</sub>	<b>ADC_CMRM1</b> 复位值: 00 <sub>H</sub> 转换请求模式寄存器 1	位域	0	LDEV	CLRP ND	SCAN	ENSI	ENTR	0	ENGT
		类型	r	w	w	rw	rw	rw	r	rw

表 3-6 ADC 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
CD <sub>H</sub>	<b>ADC_QMR0</b> 复位值: 00 <sub>H</sub> 队列模式寄存器 0	位域	CEV	TREV	FLUS H	CLRV	0	ENTR	0	ENGT
		类型	w	w	w	w	r	rw	r	rw
CE <sub>H</sub>	<b>ADC_QSR0</b> 复位值: 20 <sub>H</sub> 队列状态寄存器 0	位域	0		EMPT Y	EV	0		FILL	
		类型	r		rh	rh	r		rh	
CF <sub>H</sub>	<b>ADC_Q0R0</b> 复位值: 00 <sub>H</sub> 队列 0 寄存器 0	位域	EXTR	ENSI	RF	V	0		REQCHNR	
		类型	rh	rh	rh	rh	r		rh	
D2 <sub>H</sub>	<b>ADC_QBUR0</b> 复位值: 00 <sub>H</sub> 队列备份寄存器 0	位域	EXTR	ENSI	RF	V	0		REQCHNR	
		类型	rh	rh	rh	rh	r		rh	
D2 <sub>H</sub>	<b>ADC_QINR0</b> 复位值: 00 <sub>H</sub> 队列输入寄存器 0	位域	EXTR	ENSI	RF	0			REQCHNR	
		类型	w	w	w	r			rh	

### 3.4.5.7 LEDSCU 寄存器

LEDSCU SFR 从标准存储器区访问（RMAP = 0）。

**表 3-7 LEDSCU 寄存器概览**

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 0										
97 <sub>H</sub>	<b>LTS_GLOBCTL0</b> 复位值: 00 <sub>H</sub> 全局控制寄存器 0	位域	LD_EN	TS_EN	CLK_PS					
		类型	rw	rw	rw					
D4 <sub>H</sub>	<b>LTS_COMPARE</b> 复位值: 00 <sub>H</sub> 时间片比较映射寄存器	位域	SHD_CMP							
		类型	rw							
D5 <sub>H</sub>	<b>LTS_LDLINE</b> 复位值: 00 <sub>H</sub> LED 行值映射寄存器	位域	SHD_LINE							
		类型	rw							
D6 <sub>H</sub>	<b>LTS_LDTSTCTL</b> 复位值: 00 <sub>H</sub> LED 和触摸感应控制寄存器	位域	NR_LEDCOL			COL LEV	NR_PADT			TSO EXT
		类型	rw			rw	rw			rw
D7 <sub>H</sub>	<b>LTS_TSCTL</b> 复位值: 00 <sub>H</sub> 触摸感应控制寄存器	位域	TS CTR OVL	TS CTRR	TS CTR SAT	E PULL	PADT SW	PADT		
		类型	rw	rw	rw	rw	rw	rwh		
D8 <sub>H</sub>	<b>LTS_GLOBCTL1</b> 复位值: 00 <sub>H</sub> 全局控制寄存器 1	位域	TSF	ITS _EN	TFF	ITF _EN	CLK SEL	FNCOL		
		类型	rwh	rw	rwh	rw	rw	rh		
D9 <sub>H</sub>	<b>LTS_TSVAL</b> 复位值: 00 <sub>H</sub> 触摸感应计数值寄存器	位域	TSCTRVAL							
		类型	rwh							

### 3.4.5.8 RTC 寄存器

RTC SFR 从标准存储器区访问（RMAP = 0）。

表 3-8 RTC 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 0										
95 <sub>H</sub>	<b>RTC_RTCON</b> 复位值：00 <sub>H</sub> 实时时钟控制寄存器	位域	SFRT C	CRFT C	ESRT C	ECRT C	RTCC T	RTM		RTCC
		类型	rwh	rwh	rw	rw	rwh	rw		rw
96 <sub>H</sub>	<b>RTC_RTCON1</b> 复位值：02 <sub>H</sub> 实时时钟控制寄存器 1	位域	0							RTYR
		类型	r							rw
E1 <sub>H</sub>	<b>RTC_CNT0</b> 复位值：00 <sub>H</sub> 时钟计数寄存器 0 模式 0	位域	0		MILLISECONDS					
		类型	r		rwh					
E1 <sub>H</sub>	<b>RTC_CNT0</b> 复位值：00 <sub>H</sub> 时钟计数寄存器 0 模式 1 和 3	位域	CNT_VAL							
		类型	rwh							
E2 <sub>H</sub>	<b>RTC_CNT1</b> 复位值：00 <sub>H</sub> 时钟计数寄存器 1 模式 0 和 2	位域	0		SECONDS					
		类型	r		rwh					
E2 <sub>H</sub>	<b>RTC_CNT1</b> 复位值：00 <sub>H</sub> 时钟计数寄存器 1 模式 1 和 3	位域	CNT_VAL							
		类型	rwh							
E3 <sub>H</sub>	<b>RTC_CNT2</b> 复位值：00 <sub>H</sub> 时钟计数寄存器 2 模式 0 和 2	位域	0		MINUTES					
		类型	r		rwh					
E3 <sub>H</sub>	<b>RTC_CNT2</b> 复位值：00 <sub>H</sub> 时钟计数寄存器 2 模式 1 和 3	位域	CNT_VAL							
		类型	rwh							
E4 <sub>H</sub>	<b>RTC_CNT3</b> 复位值：00 <sub>H</sub> 时钟计数寄存器 3 模式 0 和 2	位域	0			HOURS				
		类型	r			rwh				
E4 <sub>H</sub>	<b>RTC_CNT3</b> 复位值：00 <sub>H</sub> 时钟计数寄存器 3 模式 1 和 3	位域	CNT_VAL							
		类型	rwh							
E5 <sub>H</sub>	<b>RTC_CNT4</b> 复位值：00 <sub>H</sub> 时钟计数寄存器 4 模式 0 和 2	位域	DAYS							
		类型	rwh							
E6 <sub>H</sub>	<b>RTC_CNT5</b> 复位值：00 <sub>H</sub> 时钟计数寄存器 5 模式 0 和 2	位域	0						DAYS	
		类型	r						rw	
E7 <sub>H</sub>	<b>RTC_RTCCR0</b> 复位值：00 <sub>H</sub> 实时时钟比较 / 捕获寄存器 0 模式 0	位域	0		CC_MSECS					
		类型	r		rwh					
E7 <sub>H</sub>	<b>RTC_RTCCR0</b> 复位值：00 <sub>H</sub> 实时时钟比较 / 捕获寄存器 0 模式 1 和 3	位域	CC_VAL							
		类型	rwh							
E9 <sub>H</sub>	<b>RTC_RTCCR1</b> 复位值：00 <sub>H</sub> 实时时钟比较 / 捕获寄存器 1 模式 0 和 2	位域	0		CC_SECONDS					
		类型	r		rwh					

表 3-8 RTC 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
E9 <sub>H</sub>	RTC_RTCCR1 复位值: 00 <sub>H</sub> 实时时钟比较 / 捕获寄存器 1 模式 1 和 3	位域	CC_VAL							
		类型	rwh							
EA <sub>H</sub>	RTC_RTCCR2 复位值: 00 <sub>H</sub> 实时时钟比较 / 捕获寄存器 2 模式 0 和 2	位域	0		CC_MINUTES					
		类型	r		rwh					
EA <sub>H</sub>	RTC_RTCCR2 复位值: 00 <sub>H</sub> 实时时钟比较 / 捕获寄存器 2 模式 1 和 3	位域	CC_VAL							
		类型	rwh							
EB <sub>H</sub>	RTC_RTCCR3 复位值: 00 <sub>H</sub> 实时时钟比较 / 捕获寄存器 3 模式 0 和 2	位域	0			CC_HOURS				
		类型	r			rwh				
EB <sub>H</sub>	RTC_RTCCR3 复位值: 00 <sub>H</sub> 实时时钟比较 / 捕获寄存器 3 模式 1 和 3	位域	CC_VAL							
		类型	rwh							
EC <sub>H</sub>	RTC_RTCCR4 复位值: 00 <sub>H</sub> 实时时钟比较 / 捕获寄存器 4 模式 0 和 2	位域	CC_DAYS							
		类型	rwh							
ED <sub>H</sub>	RTC_RTCCR5 复位值: 00 <sub>H</sub> 实时时钟比较 / 捕获寄存器 5 模式 0 和 2	位域	0							CC_D AYS
		类型	r							rw

### 3.4.5.9 定时器 T2 寄存器

定时器 T2 SFR 从标准存储器区访问（RMAP = 0）。

表 3-9 T2 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 0										
C0 <sub>H</sub>	<b>T2_T2CON</b> 复位值: 00 <sub>H</sub> 定时器 T2 控制寄存器	位域	TF2	EXF2	0		EXEN2	TR2	C_T2	CP_RL2
		类型	rwh	rwh	r		rw	rwh	rw	rw
C1 <sub>H</sub>	<b>T2_T2MOD</b> 复位值: 00 <sub>H</sub> 定时器 T2 模式寄存器	位域	T2REGS	T2RHEN	EDGESEL	PREN	T2PRE			DCEN
		类型	rw	rw	rw	rw	rw	rw	rw	rw
C2 <sub>H</sub>	<b>T2_RC2L</b> 复位值: 00 <sub>H</sub> 定时器 T2 重载 / 捕获寄存器, 低位字节	位域	RC2							
		类型	rwh							
C3 <sub>H</sub>	<b>T2_RC2H</b> 复位值: 00 <sub>H</sub> 定时器 T2 重载 / 捕获寄存器, 高位字节	位域	RC2							
		类型	rwh							
C4 <sub>H</sub>	<b>T2_T2L</b> 复位值: 00 <sub>H</sub> 定时器 T2 寄存器, 低位字节	位域	THL2							
		类型	rwh							
C5 <sub>H</sub>	<b>T2_T2H</b> 复位值: 00 <sub>H</sub> 定时器 T2 寄存器, 高位字节	位域	THL2							
		类型	rwh							
C6 <sub>H</sub>	<b>T2_T2CON1</b> 复位值: 03 <sub>H</sub> 定时器 T2 控制寄存器 1	位域	0						TF2EN	EXF2EN
		类型	r						rw	rw

### 3.4.5.10 CCU6 寄存器

CCU6 SFR 从标准存储器区访问（RMAP = 0）。

表 3-10 CCU6 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0	
RMAP = 0											
A3 <sub>H</sub>	CCU6_PAGE CCU6 分页寄存器	复位值：00 <sub>H</sub>	位域	OP		STNR		0	PAGE		
			类型	w		w		r	rw		
RMAP = 0, PAGE 0											
9A <sub>H</sub>	CCU6_CC63SRL 通道 CC63 捕获 / 比较 映射寄存器， 低位字节	复位值：00 <sub>H</sub>	位域	CC63SL							
			类型	rw							
9B <sub>H</sub>	CCU6_CC63SRH 通道 CC63 捕获 / 比较 映射寄存器， 高位字节	复位值：00 <sub>H</sub>	位域	CC63SH							
			类型	rw							
9C <sub>H</sub>	CCU6_TCTR4L 定时器控制寄存器 4，低位字节	复位值：00 <sub>H</sub>	位域	T12 STD	T12 STR	0		DT RES	T12 RES	T12R S	T12R R
			类型	w	w	r		w	w	w	w

**表 3-10 CCU6 寄存器概览 (cont'd)**

地址	寄存器名	位	7	6	5	4	3	2	1	0
9D <sub>H</sub>	<b>CCU6_TCTR4H</b> 复位值: 00 <sub>H</sub> 定时器控制寄存器 4, 高位字节	位域	T13 STD	T13 STR	0			T13 RES	T13R S	T13R R
		类型	w	w	r			w	w	w
9E <sub>H</sub>	<b>CCU6_MCMOUTSL</b> 复位值: 00 <sub>H</sub> 多通道模式输出映射寄存器, 低位字节	位域	STRM CM	0	MCMPS					
		类型	w	r	rw					
9F <sub>H</sub>	<b>CCU6_MCMOUTSH</b> 复位值: 00 <sub>H</sub> 多通道模式输出映射寄存器, 高位字节	位域	STRH P	0	CURHS			EXPHS		
		类型	w	r	rw			rw		
A4 <sub>H</sub>	<b>CCU6_ISR1</b> 复位值: 00 <sub>H</sub> 捕获 / 比较中断状态复位寄存器, 低位字节	位域	RT12 PM	RT12 OM	RCC6 2F	RCC6 2R	RCC6 1F	RCC6 1R	RCC6 0F	RCC6 0R
		类型	w	w	w	w	w	w	w	w
A5 <sub>H</sub>	<b>CCU6_ISRH</b> 复位值: 00 <sub>H</sub> 捕获 / 比较中断状态复位寄存器, 高位字节	位域	RSTR	RIDLE	RWHE	RCHE	0	RTRPF	RT13 PM	RT13 CM
		类型	w	w	w	w	r	w	w	w
A6 <sub>H</sub>	<b>CCU6_CMPMODIFL</b> 复位值: 00 <sub>H</sub> 比较状态修改寄存器, 低位字节	位域	0	MCC6 3S	0			MCC6 2S	MCC6 1S	MCC6 0S
		类型	r	w	r			w	w	w
A7 <sub>H</sub>	<b>CCU6_CMPMODIFH</b> 复位值: 00 <sub>H</sub> 比较状态修改寄存器, 高位字节	位域	0	MCC6 3R	0			MCC6 2R	MCC6 1R	MCC6 0R
		类型	r	w	r			w	w	w
FA <sub>H</sub>	<b>CCU6_CC60SRL</b> 复位值: 00 <sub>H</sub> 通道 CC60 捕获 / 比较 映射寄存器, 低位字节	位域	CC60SL							
		类型	rwh							
FB <sub>H</sub>	<b>CCU6_CC60SRH</b> 复位值: 00 <sub>H</sub> 通道 CC60 捕获 / 比较 映射寄存器, 高位字节	位域	CC60SH							
		类型	rwh							
FC <sub>H</sub>	<b>CCU6_CC61SRL</b> 复位值: 00 <sub>H</sub> 通道 CC61 捕获 / 比较 映射寄存器, 低位字节	位域	CC61SL							
		类型	rwh							
FD <sub>H</sub>	<b>CCU6_CC61SRH</b> 复位值: 00 <sub>H</sub> 通道 CC61 捕获 / 比较 映射寄存器, 高位字节	位域	CC61SH							
		类型	rwh							
FE <sub>H</sub>	<b>CCU6_CC62SRL</b> 复位值: 00 <sub>H</sub> 通道 CC62 捕获 / 比较 映射寄存器, 低位字节	位域	CC62SL							
		类型	rwh							
FF <sub>H</sub>	<b>CCU6_CC62SRH</b> 复位值: 00 <sub>H</sub> 通道 CC62 捕获 / 比较 映射寄存器, 高位字节	位域	CC62SH							
		类型	rwh							
RMAP = 0, PAGE 1										
9A <sub>H</sub>	<b>CCU6_CC63RL</b> 复位值: 00 <sub>H</sub> 通道 CC63 捕获 / 比较寄存器, 低位字节	位域	CC63VL							
		类型	rh							
9B <sub>H</sub>	<b>CCU6_CC63RH</b> 复位值: 00 <sub>H</sub> 通道 CC63 捕获 / 比较寄存器, 高位字节	位域	CC63VH							
		类型	rh							
9C <sub>H</sub>	<b>CCU6_T12PRL</b> 复位值: 00 <sub>H</sub> 定时器 T12 周期寄存器, 低位字节	位域	T12PVL							
		类型	rwh							

表 3-10 CCU6 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
9D <sub>H</sub>	<b>CCU6_T12PRH</b> 复位值: 00 <sub>H</sub> 定时器 T12 周期寄存器, 高位字节	位域	T12PVH							
		类型	rwh							
9E <sub>H</sub>	<b>CCU6_T13PRL</b> 复位值: 00 <sub>H</sub> 定时器 T13 周期寄存器, 低位字节	位域	T13PVL							
		类型	rwh							
9F <sub>H</sub>	<b>CCU6_T13PRH</b> 复位值: 00 <sub>H</sub> 定时器 T13 周期寄存器, 高位字节	位域	T13PVH							
		类型	rwh							
A4 <sub>H</sub>	<b>CCU6_T12DTCL</b> 复位值: 00 <sub>H</sub> 定时器 T12 死区时间 控制寄存器, 低位字节	位域	DTM							
		类型	rw							
A5 <sub>H</sub>	<b>CCU6_T12DTC</b> 复位值: 00 <sub>H</sub> 定时器 T12 死区时间 控制寄存器, 高位字节	位域	0	DTR2	DTR1	DTR0	0	DTE2	DTE1	DTE0
		类型	r	rh	rh	rh	r	rw	rw	rw
A6 <sub>H</sub>	<b>CCU6_TCTR0L</b> 复位值: 00 <sub>H</sub> 定时器控制寄存器 0, 低位字节	位域	CTM	CDIR	STE1 2	T12R	T12 PRE	T12CLK		
		类型	rw	rh	rh	rh	rw	rw		
A7 <sub>H</sub>	<b>CCU6_TCTR0H</b> 复位值: 00 <sub>H</sub> 定时器控制寄存器 0, 高位字节	位域	0		STE1 3	T13R	T13 PRE	T13CLK		
		类型	r		rh	rh	rw	rw		
FA <sub>H</sub>	<b>CCU6_CC60RL</b> 复位值: 00 <sub>H</sub> 通道 CC60 捕获 / 比较寄存器, 低位字节	位域	CC60VL							
		类型	rh							
FB <sub>H</sub>	<b>CCU6_CC60RH</b> 复位值: 00 <sub>H</sub> 通道 CC60 捕获 / 比较寄存器, 高位字节	位域	CC60VH							
		类型	rh							
FC <sub>H</sub>	<b>CCU6_CC61RL</b> 复位值: 00 <sub>H</sub> 通道 CC61 捕获 / 比较寄存器, 低位字节	位域	CC61VL							
		类型	rh							
FD <sub>H</sub>	<b>CCU6_CC61RH</b> 复位值: 00 <sub>H</sub> 通道 CC61 捕获 / 比较寄存器, 高位字节	位域	CC61VH							
		类型	rh							
FE <sub>H</sub>	<b>CCU6_CC62RL</b> 复位值: 00 <sub>H</sub> 通道 CC62 捕获 / 比较寄存器, 低位字节	位域	CC62VL							
		类型	rh							
FF <sub>H</sub>	<b>CCU6_CC62RH</b> 复位值: 00 <sub>H</sub> 通道 CC62 捕获 / 比较寄存器, 高位字节	位域	CC62VH							
		类型	rh							
RMAP = 0, PAGE 2										
9A <sub>H</sub>	<b>CCU6_T12MSELL</b> 复位值: 00 <sub>H</sub> T12 捕获 / 比较模式选择寄存器, 低位字节	位域	MSEL61				MSEL60			
		类型	rw				rw			
9B <sub>H</sub>	<b>CCU6_T12MSELH</b> 复位值: 00 <sub>H</sub> T12 捕获 / 比较模式选择寄存器, 高位字节	位域	DBYP	HSYNC			MSEL62			
		类型	rw	rw			rw			
9C <sub>H</sub>	<b>CCU6_IENL</b> 复位值: 00 <sub>H</sub> 捕获 / 比较中断使能寄存器, 低位字节	位域	ENT1 2 PM	ENT1 2 OM	ENCC 62F	ENCC 62R	ENCC 61F	ENCC 61R	ENCC 60F	ENCC 60R
		类型	rw	rw	rw	rw	rw	rw	rw	rw



**表 3-10 CCU6 寄存器概览 (cont'd)**

地址	寄存器名	位	7	6	5	4	3	2	1	0
9D <sub>H</sub>	<b>CCU6_IENH</b> 复位值: 00 <sub>H</sub> 捕获 / 比较中断使能寄存器, 高位字节	位域	EN STR	EN IDLE	EN WHE	EN CHE	0	EN TRPF	ENT1 3PM	ENT1 3CM
		类型	rw	rw	rw	rw	r	rw	rw	rw
9E <sub>H</sub>	<b>CCU6_INPL</b> 复位值: 40 <sub>H</sub> 捕获 / 比较中断节点指针寄存器, 低位字节	位域	INPCHE		INPCC62		INPCC61		INPCC60	
		类型	rw		rw		rw		rw	
9F <sub>H</sub>	<b>CCU6_INPH</b> 复位值: 39 <sub>H</sub> 捕获 / 比较中断节点指针寄存器, 高位字节	位域	0		INPT13		INPT12		INPERR	
		类型	r		rw		rw		rw	
A4 <sub>H</sub>	<b>CCU6_ISSL</b> 复位值: 00 <sub>H</sub> 捕获 / 比较中断状态置位寄存器, 低位字节	位域	ST12 PM	ST12 OM	SCC6 2F	SCC6 2R	SCC6 1F	SCC6 1R	SCC6 0F	SCC6 0R
		类型	w	w	w	w	w	w	w	w
A5 <sub>H</sub>	<b>CCU6_ISSH</b> 复位值: 00 <sub>H</sub> 捕获 / 比较中断状态置位寄存器, 高位字节	位域	SSTR	SIDLE	SWHE	SCHE	SWH C	STRP F	ST13 PM	ST13 CM
		类型	w	w	w	w	w	w	w	w
A6 <sub>H</sub>	<b>CCU6_PSLR</b> 复位值: 00 <sub>H</sub> 被动态电平寄存器	位域	PSL63	0	PSL					
		类型	rwh	r	rwh					
A7 <sub>H</sub>	<b>CCU6_MCMCTR</b> 复位值: 00 <sub>H</sub> 多通道模式控制寄存器	位域	0		SWSYN		0	SWSEL		
		类型	r		rw		r	rw		
FA <sub>H</sub>	<b>CCU6_TCTR2L</b> 复位值: 00 <sub>H</sub> 定时器控制寄存器 2, 低位字节	位域	0	T13TED		T13TEC			T13 SSC	T12 SSC
		类型	r	rw		rw			rw	rw
FB <sub>H</sub>	<b>CCU6_TCTR2H</b> 复位值: 00 <sub>H</sub> 定时器控制寄存器 2, 高位字节	位域	0				T13RSEL		T12RSEL	
		类型	r				rw		rw	
FC <sub>H</sub>	<b>CCU6_MODCTRL</b> 复位值: 00 <sub>H</sub> 调制控制寄存器, 低位字节	位域	MCM EN	0	T12MODEN					
		类型	rw	r	rw					
FD <sub>H</sub>	<b>CCU6_MODCTRH</b> 复位值: 00 <sub>H</sub> 调制控制寄存器, 高位字节	位域	ECT1 3O	0	T13MODEN					
		类型	rw	r	rw					
FE <sub>H</sub>	<b>CCU6_TRPCTRL</b> 复位值: 00 <sub>H</sub> 强制中断控制寄存器, 低位字节	位域	0					TRPM 2	TRPM 1	TRPM 0
		类型	r					rw	rw	rw
FF <sub>H</sub>	<b>CCU6_TRPCTRH</b> 复位值: 00 <sub>H</sub> 强制中断控制寄存器, 高位字节	位域	TRPP EN	TRPE N13	TRPEN					
		类型	rw	rw	rw					
RMAP = 0, PAGE 3										
9A <sub>H</sub>	<b>CCU6_MCMOUTL</b> 复位值: 00 <sub>H</sub> 多通道模式输出寄存器, 低位字节	位域	0	R	MCMP					
		类型	r	rh	rh					
9B <sub>H</sub>	<b>CCU6_MCMOUTH</b> 复位值: 00 <sub>H</sub> 多通道模式输出寄存器, 高位字节	位域	0		CURH			EXPH		
		类型	r		rh			rh		
9C <sub>H</sub>	<b>CCU6_ISL</b> 复位值: 00 <sub>H</sub> 捕获 / 比较中断状态寄存器, 低位字节	位域	T12 PM	T12 OM	ICC62 F	ICC62 R	ICC61 F	ICC61 R	ICC60 F	ICC60 R
		类型	rh	rh	rh	rh	rh	rh	rh	rh

**表 3-10 CCU6 寄存器概览 (cont'd)**

地址	寄存器名	位	7	6	5	4	3	2	1	0
9D <sub>H</sub>	CCU6_ISH 复位值: 00 <sub>H</sub> 捕获 / 比较中断状态寄存器, 高位字节	位域	STR	IDLE	WHE	CHE	TRPS	TRPF	T13 PM	T13 CM
		类型	rh	rh	rh	rh	rh	rh	rh	rh
9E <sub>H</sub>	CCU6_PISEL0L 复位值: 00 <sub>H</sub> 端口输入选择寄存器 0, 低位字节	位域	ISTRP		ISCC62		ISCC61		ISCC60	
		类型	rw		rw		rw		rw	
9F <sub>H</sub>	CCU6_PISEL0H 复位值: 00 <sub>H</sub> 端口输入选择寄存器 0, 高位字节	位域	IST12HR		ISPOS2		ISPOS1		ISPOS0	
		类型	rw		rw		rw		rw	
A4 <sub>H</sub>	CCU6_PISEL2 复位值: 00 <sub>H</sub> 端口输入选择寄存器 2	位域	0						IST13HR	
		类型	r						rw	
FA <sub>H</sub>	CCU6_T12L 复位值: 00 <sub>H</sub> 定时器 T12 计数寄存器, 低位字节	位域	T12CVL							
		类型	rwh							
FB <sub>H</sub>	CCU6_T12H 复位值: 00 <sub>H</sub> 定时器 T12 计数寄存器, 高位字节	位域	T12CVH							
		类型	rwh							
FC <sub>H</sub>	CCU6_T13L 复位值: 00 <sub>H</sub> 定时器 T13 计数寄存器, 低位字节	位域	T13CVL							
		类型	rwh							
FD <sub>H</sub>	CCU6_T13H 复位值: 00 <sub>H</sub> 定时器 T13 计数寄存器, 高位字节	位域	T13CVH							
		类型	rwh							
FE <sub>H</sub>	CCU6_CMPSTATL 复位值: 00 <sub>H</sub> 比较状态寄存器, 低位字节	位域	0	CC63 ST	CC POS2	CC POS1	CC POS0	CC62 ST	CC61 ST	CC60 ST
		类型	r	rh	rh	rh	rh	rh	rh	rh
FF <sub>H</sub>	CCU6_CMPSTATH 复位值: 00 <sub>H</sub> 比较状态寄存器, 高位字节	位域	T13IM	COUT 63PS	COUT 62PS	CC62 PS	COUT 61PS	CC61 PS	COUT 60PS	CC60 PS
		类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

### 3.4.5.11 SSC 寄存器

SSC SFR 从标准存储器区访问（RMAP = 0）。

表 3-11 SSC 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 0										
AA <sub>H</sub>	<b>SSC_CONL</b> 复位值: 00 <sub>H</sub> 控制寄存器, 低位字节 编程模式	位域	LB	PO	PH	HB	BM			
		类型	rw	rw	rw	rw	rw			
AA <sub>H</sub>	<b>SSC_CONL</b> 复位值: 00 <sub>H</sub> 控制寄存器, 低位字节 工作模式	位域	0				BC			
		类型	r				rh			
AB <sub>H</sub>	<b>SSC_CONH</b> 复位值: 00 <sub>H</sub> 控制寄存器, 高位字节 编程模式	位域	EN	MS	0	AREN	BEN	PEN	REN	TEN
		类型	rw	rw	r	rw	rw	rw	rw	rw
AB <sub>H</sub>	<b>SSC_CONH</b> 复位值: 00 <sub>H</sub> 控制寄存器, 高位字节 工作模式	位域	EN	MS	0	BSY	BE	PE	RE	TE
		类型	rw	rw	r	rh	rw	rw	rw	rw
AC <sub>H</sub>	<b>SSC_TBL</b> 复位值: 00 <sub>H</sub> 发送缓存寄存器, 低位字节	位域	TB_VALUE							
		类型	rw							
AD <sub>H</sub>	<b>SSC_RBL</b> 复位值: 00 <sub>H</sub> 接收缓存寄存器, 低位字节	位域	RB_VALUE							
		类型	rh							
AE <sub>H</sub>	<b>SSC_BRL</b> 复位值: 00 <sub>H</sub> 波特率定时器重载寄存器, 低位字节	位域	BR_VALUE							
		类型	rw							
AF <sub>H</sub>	<b>SSC_BRH</b> 复位值: 00 <sub>H</sub> 波特率定时器重载寄存器, 高位字节	位域	BR_VALUE							
		类型	rw							

### 3.4.5.12 IIC 寄存器

IIC SFR 从标准存储器区访问（RMAP = 0）。

表 3-12 IIC 寄存器概览

地址	寄存器名		位	7	6	5	4	3	2	1	0
RMAP = 0											
DA <sub>H</sub>	IIC_ADDR 从机地址寄存器	复位值：00 <sub>H</sub>	位域	SLA							GCE
			类型	rw							rw
DB <sub>H</sub>	IIC_DATA 数据字节寄存器	复位值：00 <sub>H</sub>	位域	DATA							
			类型	rw							
DC <sub>H</sub>	IIC_CNTR 控制寄存器	复位值：00 <sub>H</sub>	位域	IEN	ENAB	STA	STP	IFLG	AAK	0	
			类型	rw	rw	rwh	rwh	rwh	rw	r	
DD <sub>H</sub>	IIC_STAT 状态寄存器	复位值：F8 <sub>H</sub>	位域	STAT					0		
			类型	rh					r		

表 3-12 IIC 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
DD <sub>H</sub>	<b>IIC_BRCR</b> 复位值: 00 <sub>H</sub> 波特率控制寄存器	位域	0	BRP				PREDIV		
		类型	w	w				w		
DE <sub>H</sub>	<b>IIC_ADDRX</b> 复位值: 00 <sub>H</sub> 扩展从机地址寄存器	位域	SLAX							
		类型	rw							
DF <sub>H</sub>	<b>IIC_SRST</b> 复位值: UU <sub>H</sub> 软件复位寄存器	位域	SRST							
		类型	w							

### 3.4.5.13 OCDS 寄存器

OCDS SFR 从映射存储区访问 (RMAP = 1)。

表 3-13 OCDS 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 1										
E9 <sub>H</sub>	<b>MMCR2</b> 复位值: 1U <sub>H</sub> 监控模式控制寄存器 2	位域	STMO DE	0	DSUS P	MBCO N	ALTDI	MMEP	MMOD E	JENA
		类型	rw	rw	rw	rw	rw	rw	rh	rh
F1 <sub>H</sub>	<b>MMCR</b> 复位值: 00 <sub>H</sub> 监控模式控制寄存器	位域	MEXIT _P	MEXIT	MMNM IE	MSTE P	MRAM S_P	MRAM S	TRF	RRF
		类型	w	rw	rw	rw	w	rw	rh	rh
F2 <sub>H</sub>	<b>MMSR</b> 复位值: 00 <sub>H</sub> 监控模式状态寄存器	位域	0		EXBF	SWBF	HWB3 F	HWB2 F	HWB1 F	HWB0 F
		类型	rw		rw	rw	rw	rw	rw	rw
F3 <sub>H</sub>	<b>MMBPCR</b> 复位值: 00 <sub>H</sub> 断点控制寄存器	位域	SWBC	HWB3C		HWB2C		HWB1 C	HWB0C	
		类型	rw	rw		rw		rw	rw	
F4 <sub>H</sub>	<b>MMICR</b> 复位值: 00 <sub>H</sub> 监控模式中断控制寄存器	位域	DVEC T	DRET R	COMR ST	MSTS EL	FMIR W	FNMIR R	NMIR WE	NMIR RE
		类型	rw	rw	rw	rh	rw	rw	rw	rw
F5 <sub>H</sub>	<b>MMDR</b> 复位值: 00 <sub>H</sub> 监控模式数据传送寄存器 接收	位域	MMRR							
		类型	rh							
F5 <sub>H</sub>	<b>MMDR</b> 复位值: 00 <sub>H</sub> 监控模式数据传送寄存器 发送	位域	MMTR							
		类型	w							
F6 <sub>H</sub>	<b>HWBPSR</b> 复位值: 00 <sub>H</sub> 硬件断点选择寄存器	位域	0			BPSEL _P	BPSEL			
		类型	r			w	rw			
F7 <sub>H</sub>	<b>HWBPDR</b> 复位值: 00 <sub>H</sub> 硬件断点数据寄存器	位域	HWBPxx							
		类型	rw							
EB <sub>H</sub>	<b>MMWR1</b> 复位值: 00 <sub>H</sub> 监控工作寄存器 1	位域	MMWR1							
		类型	rw							

表 3-13        OCDS 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
EC <sub>H</sub>	MMWR2        复位值 : 00 <sub>H</sub> 监控工作寄存器 2	位域	MMWR2							
		类型	rw							

## 4 闪存存储器

XC83x 内嵌用户可编程的非易失性闪存（Flash）存储器，能够快速、可靠的存储用户代码和数据。Flash 由嵌入式电压调节器（EVR）提供的 2.5V 电压供电，不需要额外的编程或擦除电压。Flash 存储器的分区特性使每个扇区都可被独立擦除。

### 特性

- 通过 UART 进行在系统编程（ISP）
- 在应用编程（IAP）
- 纠错码（ECC）可动态纠正一位错误
- 后台编程和擦除操作，使 CPU 负荷最小
- 支持擦除中止操作
- 最小编程宽度为 32 字节
- 最小擦除宽度为 1 个扇区
- 每次读取 1 个字节
- 1 或  $3 \times \text{CCLK}$  周期的读取时间（这取决于等待状态为 0 还是 1）<sup>1)</sup>
- Flash 出厂时为擦除状态（读返回全零）

1) CPU 工作在 24 MHz 时插入一个等待状态。CPU 工作在 8 MHz 时无需插入等待状态。

## 4.1 Flash 存储器地址映射

XC83x 提供两种不同大小 Flash 的器件，其程序存储器映射如图 4-1 所示。

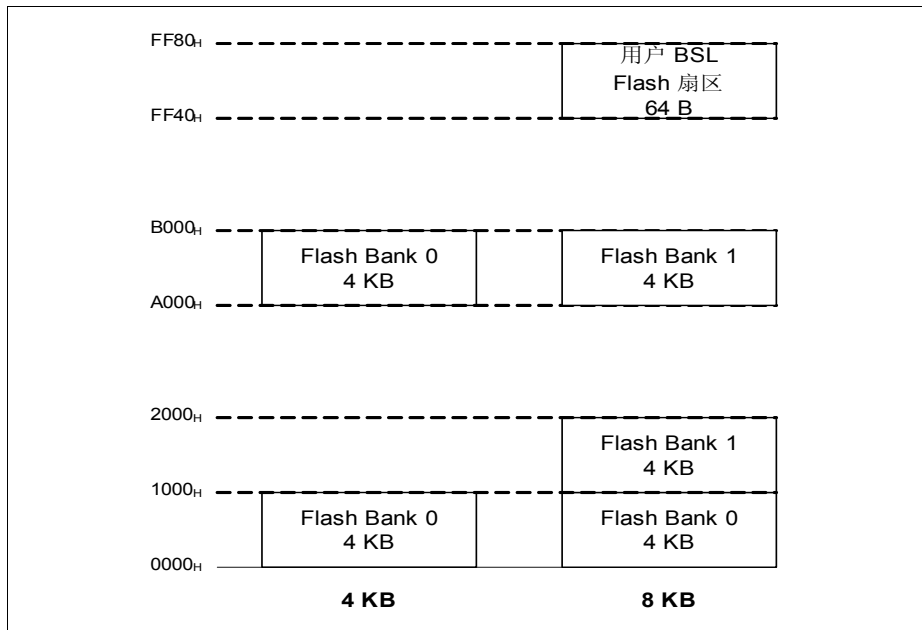


图 4-1 Flash 存储器映射

对于 8 KB 的 Flash 器件，有两个 4 KB 的 Flash bank（Bank 0 和 Bank 1）可用。Bank 0 被映射到程序存储区的地址段 0000<sub>H</sub> - 0FFF<sub>H</sub>；Bank 1 被映射到程序存储区的两个地址段 1000<sub>H</sub> - 1FFF<sub>H</sub> 和 A000<sub>H</sub> - AFFF<sub>H</sub>。这意味着，访问地址 1000<sub>H</sub> 和 A000<sub>H</sub> 将指向相同的物理单元。

*注：在 8 KB 的 Flash 器件中，包含一个独立的 64 字节用户 BSL Flash 扇区，它由两条字线组成，支持系统启动后进入备选 User BSL 模式。若不使用该特性，该扇区还可用于存储代码或数据。*

对于 4 KB 的 Flash 器件，只有一个 4 KB 的 Flash bank（Bank 0）可用，它被映射到程序存储区的两个地址段 0000<sub>H</sub> - 0FFF<sub>H</sub> 和 A000<sub>H</sub> - AFFF<sub>H</sub>。

Flash bank 的双重地址映射（8 KB Flash 器件中的 Bank 1 和 4 KB Flash 器件中的 Bank 0）便于软件编程。通常而言，位于低位地址段（0000<sub>H</sub> - 0FFF<sub>H</sub> 和 1000<sub>H</sub> - 1FFF<sub>H</sub>）的 Flash bank 应存储程序代码；位于高位地址段（A000<sub>H</sub> - AFFF<sub>H</sub>）的 Flash bank 应存储数据。

### 4.2 Flash Bank 分区

XC83x Flash 器件包含一个或两个 4 KB 的 Flash bank，其分区如图 4-2 所示。每个 Flash bank 可用于存储代码和数据。

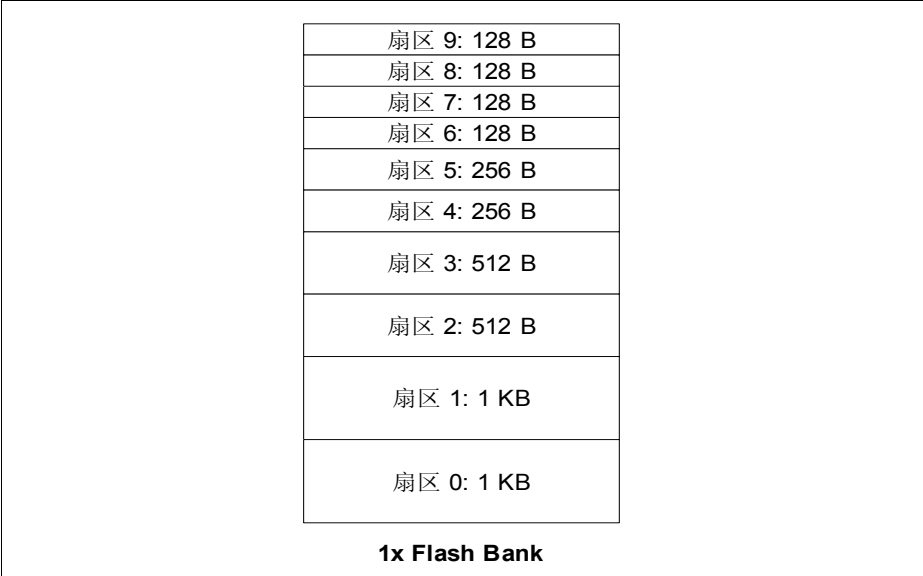


图 4-2 Flash Bank 分区

每个 4 KB Flash bank 的扇区划分：

- 两个 1 KB 扇区
- 两个 512B 扇区
- 两个 256B 扇区
- 四个 128B 扇区

每个 Flash bank 内部的扇区结构具备灵活的擦除能力。最小擦除宽度始终为一个完整的扇区。各个扇区可被分别擦除、或者同时擦除。和标准电可擦除可编程ROM（EEPROM）相反，Flash 存储器单元擦除后值全为 0。

Flash bank 被划分为更多的物理扇区，从而具有扩展的擦除和重新编程能力。每种扇区的个数均为偶数，从而能够更加灵活、更好的适应于广泛的应用领域。

举例说明，用户可编程为每个扇区实现缓存机制。每组数据双份复制保存在大小相同、独立的扇区中，以保证当真正的数据组被破坏或擦除时可使用备份数据。

用户还可实现 EEPROM 模拟，把 Flash bank 当作环形堆栈存储器使用。最新值的更新始终设定在真正区域的顶部。当达到扇区顶部时，所有真正的数据（代表 EEPROM 的数据）被复制到下一个扇区的底部，上一个扇区被擦除。用这种多方复制的循环机制来



---

## 闪存存储器

模拟 EEPROM，大大提高了 Flash 存储器的耐受能力。为了加速数据查询，RAM 可存放指向有效数据组的指针。

### 4.3 字线地址

8 KB Flash 器件中的 Bank 0 和 Bank 1 的字线（WL）地址如图 4-3 和图 4-4 所示；4 KB Flash 器件中的 Bank 0 的字线地址如图 4-5 示。

Flash Bank 0			
字节 31	字节 2	字节 1	字节 0
0FFF <sub>H</sub>	0FE2 <sub>H</sub>	0FE1 <sub>H</sub>	0FE0 <sub>H</sub>
...	...	...	...
0F9F <sub>H</sub>	0F82 <sub>H</sub>	0F81 <sub>H</sub>	0F80 <sub>H</sub>
0F7F <sub>H</sub>	0F62 <sub>H</sub>	0F61 <sub>H</sub>	0F60 <sub>H</sub>
...	...	...	...
0F1F <sub>H</sub>	0F02 <sub>H</sub>	0F01 <sub>H</sub>	0F00 <sub>H</sub>
0EFF <sub>H</sub>	0EE2 <sub>H</sub>	0EE1 <sub>H</sub>	0EE0 <sub>H</sub>
...	...	...	...
0E9F <sub>H</sub>	0E82 <sub>H</sub>	0E81 <sub>H</sub>	0E80 <sub>H</sub>
0E7F <sub>H</sub>	0E62 <sub>H</sub>	0E61 <sub>H</sub>	0E60 <sub>H</sub>
...	...	...	...
0E1F <sub>H</sub>	0E02 <sub>H</sub>	0E01 <sub>H</sub>	0E00 <sub>H</sub>
0DFF <sub>H</sub>	0DE2 <sub>H</sub>	0DE1 <sub>H</sub>	0DE0 <sub>H</sub>
...	...	...	...
0D1F <sub>H</sub>	0D02 <sub>H</sub>	0D01 <sub>H</sub>	0D00 <sub>H</sub>
0CFF <sub>H</sub>	0CE2 <sub>H</sub>	0CE1 <sub>H</sub>	0CE0 <sub>H</sub>
...	...	...	...
0C1F <sub>H</sub>	0C02 <sub>H</sub>	0C01 <sub>H</sub>	0C00 <sub>H</sub>
0BFF <sub>H</sub>	0BE2 <sub>H</sub>	0BE1 <sub>H</sub>	0BE0 <sub>H</sub>
...	...	...	...
0A3F <sub>H</sub>	0A22 <sub>H</sub>	0A21 <sub>H</sub>	0A20 <sub>H</sub>
0A1F <sub>H</sub>	0A02 <sub>H</sub>	0A01 <sub>H</sub>	0A00 <sub>H</sub>
09FF <sub>H</sub>	09E2 <sub>H</sub>	09E1 <sub>H</sub>	09E0 <sub>H</sub>
...	...	...	...
083F <sub>H</sub>	0822 <sub>H</sub>	0821 <sub>H</sub>	0820 <sub>H</sub>
081F <sub>H</sub>	0802 <sub>H</sub>	0801 <sub>H</sub>	0800 <sub>H</sub>
07FF <sub>H</sub>	07E2 <sub>H</sub>	07E1 <sub>H</sub>	07E0 <sub>H</sub>
...	...	...	...
045F <sub>H</sub>	0442 <sub>H</sub>	0441 <sub>H</sub>	0440 <sub>H</sub>
043F <sub>H</sub>	0422 <sub>H</sub>	0421 <sub>H</sub>	0420 <sub>H</sub>
041F <sub>H</sub>	0402 <sub>H</sub>	0401 <sub>H</sub>	0400 <sub>H</sub>
03FF <sub>H</sub>	03E2 <sub>H</sub>	03E1 <sub>H</sub>	03E0 <sub>H</sub>
...	...	...	...
005F <sub>H</sub>	0042 <sub>H</sub>	0041 <sub>H</sub>	0040 <sub>H</sub>
003F <sub>H</sub>	0022 <sub>H</sub>	0021 <sub>H</sub>	0020 <sub>H</sub>
001F <sub>H</sub>	0002 <sub>H</sub>	0001 <sub>H</sub>	0000 <sub>H</sub>
WL 地址			

图 4-3 8KB Flash 器件 Bank 0 字线地址

Flash Bank 1				Flash Bank 1 (备选地址映射)			
字节 31	字节 2	字节 1	字节 0	字节 31	字节 2	字节 1	字节 0
1FF <sub>H</sub>	1FE <sub>H</sub>	1FE <sub>H</sub>	1FE <sub>0</sub>	AFF <sub>H</sub>	AFE <sub>H</sub>	AFE <sub>H</sub>	AFE <sub>0</sub>
...	...	...	...	...	...	...	...
1F9 <sub>H</sub>	1F8 <sub>H</sub>	1F8 <sub>H</sub>	1F8 <sub>0</sub>	AF9 <sub>H</sub>	AF8 <sub>H</sub>	AF8 <sub>H</sub>	AF8 <sub>0</sub>
1F7 <sub>H</sub>	1F6 <sub>H</sub>	1F6 <sub>H</sub>	1F6 <sub>0</sub>	AF7 <sub>H</sub>	AF6 <sub>H</sub>	AF6 <sub>H</sub>	AF6 <sub>0</sub>
...	...	...	...	...	...	...	...
1F1 <sub>H</sub>	1F0 <sub>H</sub>	1F0 <sub>H</sub>	1F0 <sub>0</sub>	AF1 <sub>H</sub>	AF0 <sub>H</sub>	AF0 <sub>H</sub>	AF0 <sub>0</sub>
1EF <sub>H</sub>	1EE <sub>H</sub>	1EE <sub>H</sub>	1EE <sub>0</sub>	AEE <sub>H</sub>	AEE <sub>H</sub>	AEE <sub>H</sub>	AEE <sub>0</sub>
...	...	...	...	...	...	...	...
1E9 <sub>H</sub>	1E8 <sub>H</sub>	1E8 <sub>H</sub>	1E8 <sub>0</sub>	AE9 <sub>H</sub>	AE8 <sub>H</sub>	AE8 <sub>H</sub>	AE8 <sub>0</sub>
1E7 <sub>H</sub>	1E6 <sub>H</sub>	1E6 <sub>H</sub>	1E6 <sub>0</sub>	AE7 <sub>H</sub>	AE6 <sub>H</sub>	AE6 <sub>H</sub>	AE6 <sub>0</sub>
...	...	...	...	...	...	...	...
1E1 <sub>H</sub>	1E0 <sub>H</sub>	1E0 <sub>H</sub>	1E0 <sub>0</sub>	AE1 <sub>H</sub>	AE0 <sub>H</sub>	AE0 <sub>H</sub>	AE0 <sub>0</sub>
1DF <sub>H</sub>	1DE <sub>H</sub>	1DE <sub>H</sub>	1DE <sub>0</sub>	ADE <sub>H</sub>	ADE <sub>H</sub>	ADE <sub>H</sub>	ADE <sub>0</sub>
...	...	...	...	...	...	...	...
1D1 <sub>H</sub>	1D0 <sub>H</sub>	1D0 <sub>H</sub>	1D0 <sub>0</sub>	AD1 <sub>H</sub>	AD0 <sub>H</sub>	AD0 <sub>H</sub>	AD0 <sub>0</sub>
1CF <sub>H</sub>	1CE <sub>H</sub>	1CE <sub>H</sub>	1CE <sub>0</sub>	ACF <sub>H</sub>	ACE <sub>H</sub>	ACE <sub>H</sub>	ACE <sub>0</sub>
...	...	...	...	...	...	...	...
1C1 <sub>H</sub>	1C0 <sub>H</sub>	1C0 <sub>H</sub>	1C0 <sub>0</sub>	AC1 <sub>H</sub>	AC0 <sub>H</sub>	AC0 <sub>H</sub>	AC0 <sub>0</sub>
1BF <sub>H</sub>	1BE <sub>H</sub>	1BE <sub>H</sub>	1BE <sub>0</sub>	ABF <sub>H</sub>	ABE <sub>H</sub>	ABE <sub>H</sub>	ABE <sub>0</sub>
...	...	...	...	...	...	...	...
1A3 <sub>H</sub>	1A2 <sub>H</sub>	1A2 <sub>H</sub>	1A2 <sub>0</sub>	AA3 <sub>H</sub>	AA2 <sub>H</sub>	AA2 <sub>H</sub>	AA2 <sub>0</sub>
1A1 <sub>H</sub>	1A0 <sub>H</sub>	1A0 <sub>H</sub>	1A0 <sub>0</sub>	AA1 <sub>H</sub>	AA0 <sub>H</sub>	AA0 <sub>H</sub>	AA0 <sub>0</sub>
19F <sub>H</sub>	19E <sub>H</sub>	19E <sub>H</sub>	19E <sub>0</sub>	A9F <sub>H</sub>	A9E <sub>H</sub>	A9E <sub>H</sub>	A9E <sub>0</sub>
...	...	...	...	...	...	...	...
183 <sub>H</sub>	182 <sub>H</sub>	182 <sub>H</sub>	182 <sub>0</sub>	A83 <sub>H</sub>	A82 <sub>H</sub>	A82 <sub>H</sub>	A82 <sub>0</sub>
181 <sub>H</sub>	180 <sub>H</sub>	180 <sub>H</sub>	180 <sub>0</sub>	A81 <sub>H</sub>	A80 <sub>H</sub>	A80 <sub>H</sub>	A80 <sub>0</sub>
17F <sub>H</sub>	17E <sub>H</sub>	17E <sub>H</sub>	17E <sub>0</sub>	A7F <sub>H</sub>	A7E <sub>H</sub>	A7E <sub>H</sub>	A7E <sub>0</sub>
...	...	...	...	...	...	...	...
145 <sub>H</sub>	144 <sub>H</sub>	144 <sub>H</sub>	144 <sub>0</sub>	A45 <sub>H</sub>	A44 <sub>H</sub>	A44 <sub>H</sub>	A44 <sub>0</sub>
143 <sub>H</sub>	142 <sub>H</sub>	142 <sub>H</sub>	142 <sub>0</sub>	A43 <sub>H</sub>	A42 <sub>H</sub>	A42 <sub>H</sub>	A42 <sub>0</sub>
141 <sub>H</sub>	140 <sub>H</sub>	140 <sub>H</sub>	140 <sub>0</sub>	A41 <sub>H</sub>	A40 <sub>H</sub>	A40 <sub>H</sub>	A40 <sub>0</sub>
13F <sub>H</sub>	13E <sub>H</sub>	13E <sub>H</sub>	13E <sub>0</sub>	A3F <sub>H</sub>	A3E <sub>H</sub>	A3E <sub>H</sub>	A3E <sub>0</sub>
...	...	...	...	...	...	...	...
105 <sub>H</sub>	104 <sub>H</sub>	104 <sub>H</sub>	104 <sub>0</sub>	A05 <sub>H</sub>	A04 <sub>H</sub>	A04 <sub>H</sub>	A04 <sub>0</sub>
103 <sub>H</sub>	102 <sub>H</sub>	102 <sub>H</sub>	102 <sub>0</sub>	A03 <sub>H</sub>	A02 <sub>H</sub>	A02 <sub>H</sub>	A02 <sub>0</sub>
101 <sub>H</sub>	100 <sub>H</sub>	100 <sub>H</sub>	100 <sub>0</sub>	A01 <sub>H</sub>	A00 <sub>H</sub>	A00 <sub>H</sub>	A00 <sub>0</sub>
WL 地址				WL 地址			

**图 4-4 8KB Flash 器件 Bank1 字线地址**

## 闪存存储器

Flash Bank 0				Flash Bank 0 (备选地址映射)			
字节 31	字节 2	字节 1	字节 0	字节 31	字节 2	字节 1	字节 0
0FF <sub>FH</sub>	0FE <sub>2H</sub>	0FE <sub>1H</sub>	0FE <sub>0H</sub>	AFF <sub>FH</sub>	AFE <sub>2H</sub>	AFE <sub>1H</sub>	AFE <sub>0H</sub>
...	...	...	...	...	...	...	...
09F <sub>FH</sub>	0F8 <sub>2H</sub>	0F8 <sub>1H</sub>	0F8 <sub>0H</sub>	AF9 <sub>FH</sub>	AF8 <sub>2H</sub>	AF8 <sub>1H</sub>	AF8 <sub>0H</sub>
07F <sub>FH</sub>	0F6 <sub>2H</sub>	0F6 <sub>1H</sub>	0F6 <sub>0H</sub>	AF7 <sub>FH</sub>	AF6 <sub>2H</sub>	AF6 <sub>1H</sub>	AF6 <sub>0H</sub>
...	...	...	...	...	...	...	...
01F <sub>FH</sub>	0F0 <sub>2H</sub>	0F0 <sub>1H</sub>	0F0 <sub>0H</sub>	AF1 <sub>FH</sub>	AF0 <sub>2H</sub>	AF0 <sub>1H</sub>	AF0 <sub>0H</sub>
0EF <sub>FH</sub>	0EE <sub>2H</sub>	0EE <sub>1H</sub>	0EE <sub>0H</sub>	AEE <sub>FH</sub>	AEE <sub>2H</sub>	AEE <sub>1H</sub>	AEE <sub>0H</sub>
...	...	...	...	...	...	...	...
0E9 <sub>FH</sub>	0E8 <sub>2H</sub>	0E8 <sub>1H</sub>	0E8 <sub>0H</sub>	AE9 <sub>FH</sub>	AE8 <sub>2H</sub>	AE8 <sub>1H</sub>	AE8 <sub>0H</sub>
0E7 <sub>FH</sub>	0E6 <sub>2H</sub>	0E6 <sub>1H</sub>	0E6 <sub>0H</sub>	AE7 <sub>FH</sub>	AE6 <sub>2H</sub>	AE6 <sub>1H</sub>	AE6 <sub>0H</sub>
...	...	...	...	...	...	...	...
0E1 <sub>FH</sub>	0E0 <sub>2H</sub>	0E0 <sub>1H</sub>	0E0 <sub>0H</sub>	AE1 <sub>FH</sub>	AE0 <sub>2H</sub>	AE0 <sub>1H</sub>	AE0 <sub>0H</sub>
0DF <sub>FH</sub>	0DE <sub>2H</sub>	0DE <sub>1H</sub>	0DE <sub>0H</sub>	ADF <sub>FH</sub>	ADE <sub>2H</sub>	ADE <sub>1H</sub>	ADE <sub>0H</sub>
...	...	...	...	...	...	...	...
0D1 <sub>FH</sub>	0D0 <sub>2H</sub>	0D0 <sub>1H</sub>	0D0 <sub>0H</sub>	AD1 <sub>FH</sub>	AD0 <sub>2H</sub>	AD0 <sub>1H</sub>	AD0 <sub>0H</sub>
0CF <sub>FH</sub>	0CE <sub>2H</sub>	0CE <sub>1H</sub>	0CE <sub>0H</sub>	ACF <sub>FH</sub>	ACE <sub>2H</sub>	ACE <sub>1H</sub>	ACE <sub>0H</sub>
...	...	...	...	...	...	...	...
0C1 <sub>FH</sub>	0C0 <sub>2H</sub>	0C0 <sub>1H</sub>	0C0 <sub>0H</sub>	AC1 <sub>FH</sub>	AC0 <sub>2H</sub>	AC0 <sub>1H</sub>	AC0 <sub>0H</sub>
0BF <sub>FH</sub>	0BE <sub>2H</sub>	0BE <sub>1H</sub>	0BE <sub>0H</sub>	ABF <sub>FH</sub>	ABE <sub>2H</sub>	ABE <sub>1H</sub>	ABE <sub>0H</sub>
...	...	...	...	...	...	...	...
0A3 <sub>FH</sub>	0A2 <sub>2H</sub>	0A2 <sub>1H</sub>	0A2 <sub>0H</sub>	AA3 <sub>FH</sub>	AA2 <sub>2H</sub>	AA2 <sub>1H</sub>	AA2 <sub>0H</sub>
0A1 <sub>FH</sub>	0A0 <sub>2H</sub>	0A0 <sub>1H</sub>	0A0 <sub>0H</sub>	AA1 <sub>FH</sub>	AA0 <sub>2H</sub>	AA0 <sub>1H</sub>	AA0 <sub>0H</sub>
09F <sub>FH</sub>	09E <sub>2H</sub>	09E <sub>1H</sub>	09E <sub>0H</sub>	A9F <sub>FH</sub>	A9E <sub>2H</sub>	A9E <sub>1H</sub>	A9E <sub>0H</sub>
...	...	...	...	...	...	...	...
083 <sub>FH</sub>	082 <sub>2H</sub>	082 <sub>1H</sub>	082 <sub>0H</sub>	A83 <sub>FH</sub>	A82 <sub>2H</sub>	A82 <sub>1H</sub>	A82 <sub>0H</sub>
081 <sub>FH</sub>	080 <sub>2H</sub>	080 <sub>1H</sub>	080 <sub>0H</sub>	A81 <sub>FH</sub>	A80 <sub>2H</sub>	A80 <sub>1H</sub>	A80 <sub>0H</sub>
07F <sub>FH</sub>	07E <sub>2H</sub>	07E <sub>1H</sub>	07E <sub>0H</sub>	A7F <sub>FH</sub>	A7E <sub>2H</sub>	A7E <sub>1H</sub>	A7E <sub>0H</sub>
...	...	...	...	...	...	...	...
045 <sub>FH</sub>	044 <sub>2H</sub>	044 <sub>1H</sub>	044 <sub>0H</sub>	A45 <sub>FH</sub>	A44 <sub>2H</sub>	A44 <sub>1H</sub>	A44 <sub>0H</sub>
043 <sub>FH</sub>	042 <sub>2H</sub>	042 <sub>1H</sub>	042 <sub>0H</sub>	A43 <sub>FH</sub>	A42 <sub>2H</sub>	A42 <sub>1H</sub>	A42 <sub>0H</sub>
041 <sub>FH</sub>	040 <sub>2H</sub>	040 <sub>1H</sub>	040 <sub>0H</sub>	A41 <sub>FH</sub>	A40 <sub>2H</sub>	A40 <sub>1H</sub>	A40 <sub>0H</sub>
03F <sub>FH</sub>	03E <sub>2H</sub>	03E <sub>1H</sub>	03E <sub>0H</sub>	A3F <sub>FH</sub>	A3E <sub>2H</sub>	A3E <sub>1H</sub>	A3E <sub>0H</sub>
...	...	...	...	...	...	...	...
005 <sub>FH</sub>	004 <sub>2H</sub>	004 <sub>1H</sub>	004 <sub>0H</sub>	A05 <sub>FH</sub>	A04 <sub>2H</sub>	A04 <sub>1H</sub>	A04 <sub>0H</sub>
003 <sub>FH</sub>	002 <sub>2H</sub>	002 <sub>1H</sub>	002 <sub>0H</sub>	A03 <sub>FH</sub>	A02 <sub>2H</sub>	A02 <sub>1H</sub>	A02 <sub>0H</sub>
001 <sub>FH</sub>	000 <sub>2H</sub>	000 <sub>1H</sub>	000 <sub>0H</sub>	A01 <sub>FH</sub>	A00 <sub>2H</sub>	A00 <sub>1H</sub>	A00 <sub>0H</sub>
WL 地址				WL 地址			

图 4-5 4KB Flash 器件 Bank0 字线地址

## 闪存存储器

对于 8 KB Flash 器件，WL 地址计算如下：

$$0000_H + 20_H \times n, 0 \leq n \leq 127 \quad \text{Flash Bank 0} \quad (4.1)$$

$$1000_H/A000_H + 20_H \times n, 0 \leq n \leq 127 \quad \text{Flash Bank 1} \quad (4.2)$$

对于 4 KB Flash 器件，WL 地址计算如下：

$$0000_H/A000_H + 20_H \times n, 0 \leq n \leq 127 \quad \text{Flash Bank 0} \quad (4.3)$$

Flash bank 每次只能编程一条字线，每条字线的最大 / 最小编程宽度为 32 个字节。编程之前，用户必须首先用 "MOV" 指令将数据写入 IRAM，写入的字节个数等于编程宽度；然后通过引导程序加载 (BSL) 程序 (见 [章节 4.6](#)) 或 Flash 编程子程序 (见 [章节 4.7.1](#)) 将这些 IRAM 数据传送到 Flash bank 对应的写缓存中。一旦数据写入缓存，内嵌的编程和擦除状态机使充电泵电压逐渐上升。一旦完成充电过程，写缓存中易失的数据内容被存储到所选字线的非易失 Flash 单元中，根据 [图 4-3](#)、[图 4-4](#) 和 [图 4-5](#) 所示的字线地址选择字线。数据填充 IRAM 是非常必要的 (字节个数等于编程宽度)，否则上一次存储在写缓存中的内容将被保留并被编程到字线上。

由于 Flash 单元能承受两个门干扰，同一条字线在擦除前可编程两次。这意味着如果需要编程的数据字节的个数小于 32 字节 (最小编程宽度)，用户可选择先编程这些数据字节 ( $x$ ； $x$  可以是 1-31 之间的任意整数)，之后再编程其余字节 ( $32-x$ )。不过，由于 Flash 的最小编程宽度始终为 32 个字节，因此在每个编程周期中未被编程的字节必须写入全 0。

[图 4-6](#) 示例用 16 个数据字节对同一条字线编程两次。第一个编程周期，低 16 字节写入有效数据，不含有效数据的高 16 字节写入全 0；第二个编程周期恰恰相反，高 16 字节写入有效数据，已含有效数据的低 16 字节写入全 0。

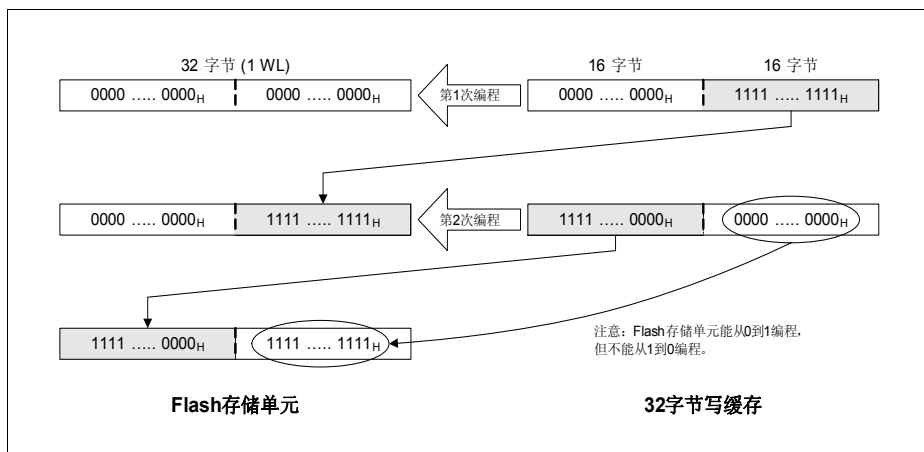


图 4-6 Flash 编程

## 4.4 工作模式

每个 Flash bank 的工作模式如图 4-7 所示。

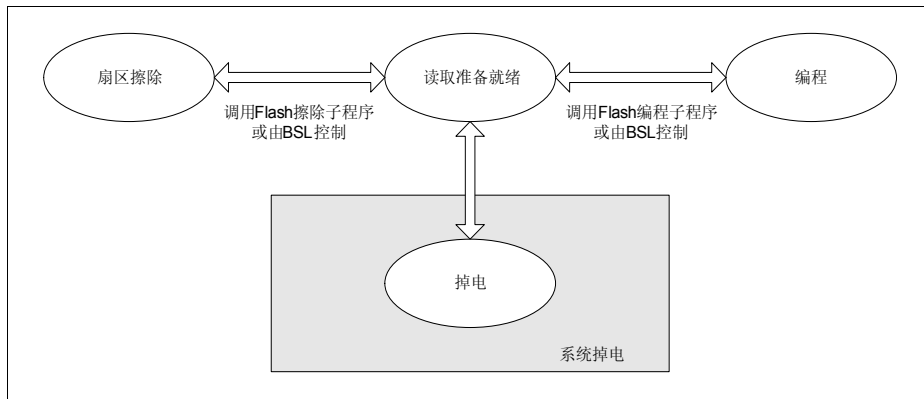


图 4-7 Flash 工作模式

通常，Flash 的工作模式由 BSL 和 Flash 编程或擦除子程序控制（见章节 4.7）。

Flash bank 在进入编程或扇区擦除模式之前必须处在读取准备就绪模式。该模式下，Flash bank 的 32 字节写缓存可被写入，存储器单元的内容可由 CPU 读出。编程模式下，32 字节写缓存中的数据被编程到目标字线的 Flash 存储单元中。

每个 Flash bank 的工作模式由专用的状态机控制，以保证不同模式之间的正确转移，这就避免了 Flash 的内容被无意破坏、并且有合理的较低软件开销。状态机还确保了 Flash bank 在编程或擦除过程中被封锁（不可读取）。在任何时候，Flash bank 只能处于读取准备就绪、编程或者扇区擦除模式。不过，在读取一个 Flash bank 中的内容的同时，可对另一个 Flash bank 编程或擦除。

当用户设置  $PMCON0.PD = 1$  进入系统掉电模式，Flash bank 由硬件自动带入掉电状态。系统从掉电状态唤醒后，Flash bank 进入读取准备就绪模式，允许 CPU 访问。

## 4.5 检错和纠错

CPU 送出的 8 位数据存储到 Flash 中之前，采用纠错码（ECC）编码。读取时，从 Flash 中取出数据并译码，实现动态检错和纠错。

纠错算法（汉明码）能够：

- 检查并纠正所有的 1 位错误
- 检查所有的 2 位错误，但无法纠正

无法区分可纠正的 1 位错误（结果有效）和不可纠正的 2 位错误（结果无效），两种情况均产生 ECC 非屏蔽中断（NMI）事件；寄存器 NMISR 中的位 FNMI ECC 被置位；若 NMICON.NMIECC 被使能，则触发一个 NMI 中断。ECC 出错的 16 位 Flash 地址被

---

**闪存存储器**

保存在系统控制 SFR FEAL 和 FEAH 中，中断服务程序可访问这些寄存器，从而确定出错的 Flash bank /Flash 扇区。

### 4.5.1 Flash 出错地址寄存器

FEAL 和 FEAH 寄存器中存放 ECC 出错的 16 位 Flash 地址。必须在访问这些寄存器之前设置 SCU\_PAGE.PAGE。

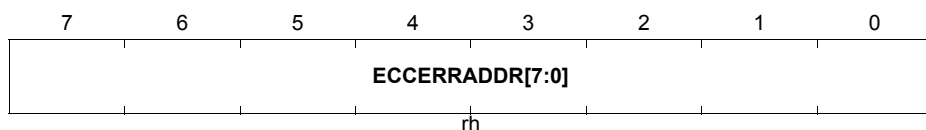
#### FEAL

Flash 出错地址寄存器，低位字节

(F6<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 5



符号	位	类型	描述
ECCERRADDR	[7:0]	rh	ECC 出错地址 [7:0]

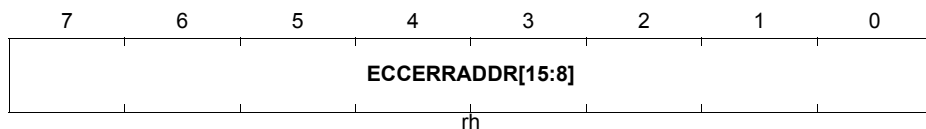
#### FEAH

Flash 出错地址寄存器，高位字节

(F7<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 5



符号	位	类型	描述
ECCERRADDR	[7:0]	rh	ECC 出错地址 [15:8]



## 4.6 在系统编程

基于 Boot ROM 的引导程序加载器 (BSL) 支持 Flash 在系统编程 (ISP)，可将用户程序加载到应用板上的空白微控制器中；对以前已编程的微控制器，无需从应用板上卸下、即可直接擦除并重新编程。该特性使嵌入式设计易于使用且功能多样。

微控制器的串行接口 (UART) 支持 ISP，通过常用的 RS-232 串行电缆和 PC 主机相连。上电复位或硬件复位后，若 BMI 和 BMI 值和设定值匹配，则选择进入相应的 BSL 模式。BSL 程序首先和串行通信端 (PC 主机) 的传输速率 (波特率) 自动同步。BSL 和主机之间的通信通过传送协议完成。主机按照规定的块结构向微控制器按块发送信息，BSL 程序返回一个应答或出错字节以回应接收数据。用户可编程、擦除或执行 Flash bank 的内容。

可用的工作模式包括：

- 将用户程序从主机传送到 Flash 中
- 执行 Flash 中的用户程序
- 从相同或不同的 Flash bank 中擦除 Flash 扇区
- 所有 Flash 扇区全部擦除

## 4.7 在应用编程

在一些实际应用中，程序执行期间可能需要修改 Flash 的内容。系统支持在应用（IAP）编程，Flash 用户程序通过调用 Boot ROM 中的特殊子程序，可对 Flash 存储器进行编程或擦除（参见图 4-8 和第 23 章）。在启动编程或者擦除操作之前，Flash 子程序将首先执行一些检查并执行初始化序列。通过 "MOVC" 指令读取 Flash 的内容，检查编程或擦除是否成功是非常必要的。其它特殊子程序包括：中止 Flash 擦除操作和检查 Flash bank 的读取准备就绪状态。

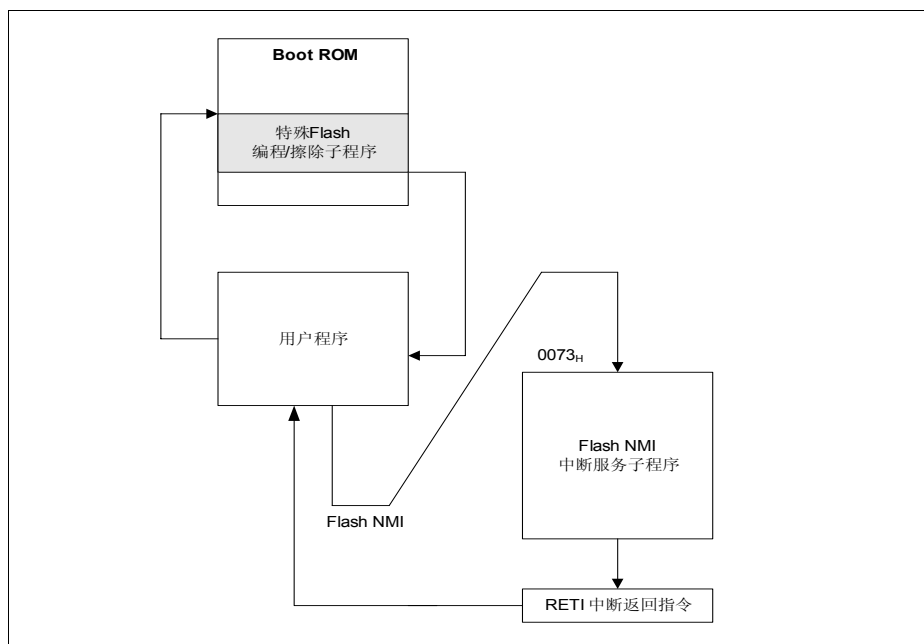


图 4-8 Flash 编程 / 擦除流程

### 4.7.1 Flash 编程

每次调用 Flash 编程子程序，可将 32 字节的数据编程到：

- Flash bank 选中的字线（WL）上
- 64 字节用户 BSL Flash 扇区中<sup>1)</sup>

调用该子程序前，用户必须确保 32 字节的字线内容按升序依次存放在 IRAM 中，由当前寄存器 bank 中的 R5 指定存放的起始地址。此外，输入 DPTR 必须包含有效的 Flash WL 地址，SFR NMISR 中的所有 NMI 标志必须为 0。否则，PSW.CY 将被置位，不执行编程操作。

Boot ROM 中提供了两种 Flash 编程子程序：

- 非后台 Flash 编程子程序（见 [章节 4.7.1.1](#)）
- 后台 Flash 编程子程序（见 [章节 4.7.1.2](#)）

#### 4.7.1.1 非后台 Flash 编程子程序

非后台 Flash 编程子程序在 Flash 编程结束之前不允许继续执行用户程序。

可使用该子程序对正在执行用户代码的 Flash bank 进行编程。不能同时编程和读取 Flash。还可使用该子程序对定义中断向量的 Flash bank 进行编程，因为在 Flash 编程期间不能处理中断。

对于该子程序，SFR NMISR 中的 FNMIFLASH 标志自动清零，因此无须用户处理。

#### 4.7.1.2 后台 Flash 编程子程序

后台 Flash 编程子程序允许用户程序从上次中止处继续向后执行，直至产生 Flash NMI 事件。随后 FNMIFLASH 标志置位，若 SFR NMICON 中的控制位 NMIFLASH 使能该中断，则产生 Flash NMI 送入 CPU，进入 Flash NMI 中断服务程序。此时，Flash bank 处于读取准备就绪模式，即编程结束。

该子程序可用于以下两种情况：

- 正在执行用户代码的 Flash bank 和待编程的 Flash bank 是不同的 bank
- XRAM 调用该子程序

*注：应避免 Flash bank 1 调用该子程序来编程用户 BSL Flash 扇区（反之亦然）的情况，此时应使用非后台 Flash 编程子程序。*

### 4.7.2 Flash 擦除

每次调用 Flash 擦除子程序，允许用户选择擦除：

- 一个扇区；或
- 64 字节用户 BSL Flash 扇区<sup>2)</sup>

1) 仅在 8KB Flash 器件中可用。

2) 仅在 8KB Flash 器件中可用。

- 相同或不同 Flash bank 中的多个扇区<sup>1)</sup>

调用该子程序前，用户必须确保当前寄存器 bank 中的 R4 - R7 已作相应设置。选择擦除某扇区时，R4 - R7 中对应该扇区的控制位必须置 1。不被擦除的扇区对应的控制位置 0。若不提供输入或输入无效，PSW.CY 将被置位，不执行擦除操作。

Boot ROM 中提供了两种 Flash 擦除子程序：

- 非后台 Flash 擦除子程序（见章节 4.7.2.1）
- 后台 Flash 擦除子程序（见章节 4.7.2.2）

#### 4.7.2.1 非后台 Flash 擦除子程序

非后台 Flash 擦除子程序在 Flash 擦除结束之前不允许继续执行用户程序。

可使用该子程序对正在执行用户代码的 Flash bank 进行擦除。不能同时擦除和读取 Flash。还可使用该子程序对定义中断向量的 Flash bank 进行擦除，因为在 Flash 擦除期间不能处理中断。

对于该子程序，SFR NMISR 中的 FNMIFLASH 标志自动清零，因此无须用户处理。

*注：由于 Flash 擦除操作结束后程序将返回用户代码，因此，用户切勿大意一并擦除了自己的用户程序。*

#### 4.7.2.2 后台 Flash 擦除子程序

后台 Flash 擦除子程序允许用户程序从上次中止处继续向后执行，直至产生 Flash NMI 事件。随后 FNMIFLASH 标志置位，若控制位 NMIFLASH 使能该中断，则产生 Flash NMI 送入 CPU，进入 Flash NMI 中断服务程序。此时，Flash bank 处于读取准备就绪模式，即擦除结束。

该子程序可用于以下两种情况：

- 正在执行用户代码的 Flash bank 和待擦除的 Flash bank 是不同的 bank
- XRAM 调用该子程序

*注：应避免 Flash bank 1 调用该子程序来擦除用户 BSL Flash 扇区（反之亦然）的情况，此时应使用非后台 Flash 擦除子程序。*

#### 4.7.3 中止后台 Flash 擦除

一次完整的 Flash bank 的擦除操作大约需要 100 ms，在此期间，不能对该 Flash bank 执行读取和编程操作。XC83x 扩展出后台 Flash 擦除中止功能：正在执行的擦除操作可以被中止，从而使具有更高优先级的任务得以执行，例如从该 Flash bank 读取关键数据、或将关键数据编程到该 Flash bank 中。因此，所选 Flash bank 扇区上的擦除操作可以被中止，从而允许读取或编程其它扇区。为了尽量减少中止操作对 Flash 数据保持 / 可重复擦写次数的影响，保证数据的可靠性，必须注意以下几点：

- 擦除操作启动之后，在 5 ms 之内不能中止擦除操作。

<sup>1)</sup> 可能包含用户 BSL Flash 扇区。

## 闪存存储器

- 每一个扇区上最多允许 2 次连续的中止擦除（两次之间没有完整的擦除操作）。
- 在单次或者连续两次擦除操作被中止之后，由于相关扇区的数据被破坏，需要通过用户程序进行完整的擦除操作（大约 100 ms）。
- 从Flash可重复擦写次数<sup>1)</sup>的角度，一次擦除中止操作视作一次编程/擦除操作来计数。
- 在 Flash 的使用寿命内，每个 Flash 扇区的擦除中止操作最多为 2500 次。

擦除操作开始之后，5 ms 内不能调用 Flash 擦除中止子程序，请用户务必严格遵守该要求。否则，不能中止擦除操作。一旦该子程序执行完毕，用户可调用 Flash 读模式状态子程序，检查中止擦除操作是否已结束。当所选的 Flash bank 已处于读模式，则表明中止擦除操作已完成。

1) Flash 的详细参数信息请参考 XC83x 数据手册。

#### 4.7.4 Flash 读模式状态

除 Flash 编程和擦除子程序之外，还有一个子程序用于检查 Flash bank 是否已作好被读取的准备。

调用该子程序之前，用户必须确保将输入 R7 配置为选中的 Flash bank。Flash 处于读取准备就绪模式时，进位标志清零，否则该标志置位。若子程序的输入无效，进位标志也置位。

执行中止后台 Flash 擦除子程序时，该子程序尤其有用：调用中止擦除子程序时，用户可调用该子程序，检查是否已成功完成擦除操作、Flash 是否已处于等待读取模式。

## 5 Boot 及系统启动

根据 Boot 模式索引 / 列表（BMI）值进入不同的 boot 模式，如用户模式、引导程序加载（BSL）模式和片上调试（OCDS）模式。有关每种 boot 模式的详细描述，请参见[章节 5.2](#)。

可通过 BSL 模式 6 或用户程序 BR\_PROG\_USER\_ID 编程设定 BMI/BMI 值。用户需要设置 BMI 和 BMI 以进入不同的 boot 模式。UART BSL 模式为系统的缺省模式。引入 BMI 用以检查 BMI 值是否有效：当 BMI + BMI + 1 = 0 时 BMI 值有效。每种模式对应的 BMI 值归纳见[表 5-1](#)。

**表 5-1 Boot 模式索引 / 列表**

Boot 模式 <sup>1)</sup>	BMI	BMI
UART BSL 模式	00 <sub>H</sub>	FF <sub>H</sub>
UART BSL 模式 <sup>2)</sup>	00 <sub>H</sub>	00 <sub>H</sub>
备选用户 BSL 模式 <sup>3)</sup>	01 <sub>H</sub>	FE <sub>H</sub>
用户模式（量产）	10 <sub>H</sub>	EF <sub>H</sub>
使用 SPD 引脚 SPD_0 的用户模式（诊断）	50 <sub>H</sub>	AF <sub>H</sub>
使用 SPD 引脚 SPD_1 的用户模式（诊断）	52 <sub>H</sub>	AD <sub>H</sub>
使用 SPD 引脚 SPD_2 的用户模式（诊断）	54 <sub>H</sub>	AB <sub>H</sub>
使用 SPD 引脚 SPD_0 的 OCDS 模式	60 <sub>H</sub>	9F <sub>H</sub>
使用 SPD 引脚 SPD_1 的 OCDS 模式	62 <sub>H</sub>	9D <sub>H</sub>
使用 SPD 引脚 SPD_2 的 OCDS 模式	64 <sub>H</sub>	9B <sub>H</sub>
保留	其它	其它

1) SPD：单引脚 DAP；

2) 当 BMI 为 00、BMI 为 FF 时，可进入 UART BSL 模式。

3) 直接跳转至 DFlash Bank 1 的地址单元 FF40H 处并执行代码。仅 8 KB Flash 产品支持该 boot 模式。

BMI 和 BMI 是用户 ID 寄存器 USER\_ID 的一部分。USER\_ID 是 4 字节数据，其中存放特定的用户信息，具体描述见[章节 5.1](#)。用户不但可通过编程 USER\_ID 设置 BMI 值，还可选择系统的工作频率（8 MHz 或 24 MHz）。在执行用户程序之前选择 boot 模式。

若器件已工作在用户模式（量产），可通过用户程序中的特定代码修改 BMI 值使其进入另一种 boot 模式。只有满足设定的用户条件时，才能执行该代码。在该特定代码中，用户可使用以下指令修改 BMI 值：

- LJMP 至用户程序 BR\_UART\_BSL，进入 BSL 模式（使用 BSL 模式 6）
- LCALL 用户程序 BR\_PROG\_USER\_ID，编程 USER\_ID 进入其它 boot 模式

出于保护代码的目的，用户可在执行该代码之前擦除所有 Flash 内容。有关这些用户程序的详细说明，可参见 Boot ROM 用户程序一章的内容。

## Boot 及系统启动

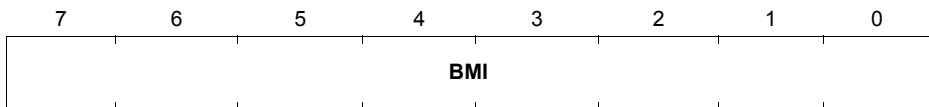
- 注： 更新 **BMI** 之后将执行软复位。在软复位之前关闭电源可能会导致 **BMI** 更新出错。系统无需其它复位操作即可使用新 **BMI** 值。
- 注： 建议用户在修改 **BMI** 的过程中禁用看门狗定时器 (**WDT**)。若在修改 **BMI** 的过程中发生 **WDT** 复位，可能会导致 **BMI** 更新出错。除此之外，还必须保持供电稳定。错误的 **BMI** 可使系统进入死锁状态。
- 注： 无意执行 **BR\_PROG\_USER\_ID** 用户程序将影响 **BMI**。无效的 **BMI** 可能导致器件无法正常启动。因此，必须设置一定的用户条件以防止无意跳转到该程序。

在 **OCDS** 模式和用户模式 (诊断) 下，将根据 **BMI** 和  $\overline{\text{BMI}}$  的值使能单引脚 **DAP (SPD)** 的调试接口。

### 5.1 用户 ID

**USER\_ID** 是 4 字节数据，其中存放特定的用户信息 (如启动选择 **BMI**)。用户可通过 **BR\_GET\_4\_BYTES\_INFO** 用户程序或 **BSL** 模式 **A** 访问 **USER\_ID**。用户可通过 **BR\_FEATURE\_SETTING** 用户程序或 **BSL** 模式 **6** 设置 **USER\_ID**。有关 **BSL** 模式的详细描述，请参见 **UART** 引导程序加载一章的内容。

#### **USER\_ID**, 字节 0



符号	位	描述
<b>BMI</b>	[7:0]	<b>Boot 模式索引 / 列表</b> 选择进入不同的 boot 模式：用户模式、UART BSL 模式和 OCDS 模式，详见表 5-1。

#### **USER\_ID**, 字节 1





## Boot 及系统启动

符号	位	描述
<b>BMI</b>	[15:8]	<b>BMI 值取反</b> BMI 值取反。通过 $\overline{\text{BMI}}$ 检查 BMI 是否有效。当 $\text{BMI} + \overline{\text{BMI}} + 1 = 0$ 时，BMI 有效。

### USER\_ID, 字节 2

23	22	21	20	19	18	17	16
			0				

符号	位	描述
<b>0</b>	[23:16]	保留

### USER\_ID, 字节 3

31	30	29	28	27	26	25	24
CLKMODE_SEL	PERIPHERALS_EN			0			

符号	位	描述
<b>0</b>	[29:24]	保留
<b>PERIPHERALS_EN</b>	30	<b>外设使能位</b> 0 禁用寄存器 PMCON1 中定义的所有外设 1 使能寄存器 PMCON1 中定义的所有外设
<b>CLKMODE_SEL</b>	31	<b>时钟模式选择</b> 0 8 MHz 工作模式（Flash 读取无需等待） 1 24 MHz 工作模式（Flash 读取需一个等待状态）

## 5.2 Boot ROM 工作模式

复位后 CPU 开始工作，始终先执行程序存储器地址段  $0000_H - 1FFF_H$  中的 Boot ROM 代码。Boot ROM 启动进程将首先切换到地址段  $C000_H - DFFF_H$ ，随后从  $C00X_H$  继续执行其余的 Boot ROM 启动进程。本手册中给出的 XC83x 存储器结构是 Boot ROM 地址切换之后的存储器结构图，在该结构下进入 Boot ROM 的不同工作模式。

### 5.2.1 用户模式（量产）

若选择用户模式（量产），Boot ROM 将跳转到程序存储器的地址  $0000_H$  处，执行 Flash 存储器中的用户程序。在该模式下，外部访问 Flash 时其内容受保护。这是 XC83x 的正常工作模式。

### 5.2.2 用户模式（诊断）

若选择用户模式（诊断），Boot ROM 将跳转到程序存储器的地址  $0000_H$  处，执行 Flash 存储器中的用户程序。和章节 5.2.1 所描述的用户模式（量产）相似，区别在于：该模式下，指定的 SPD 端口被自动配置为允许热插拔。

### 5.2.3 引导程序加载模式

若选择引导程序加载（BSL）模式，将执行 Boot ROM 中的引导加载程序，可以对 XRAM 和 Flash 存储器编程、擦除和执行。有关不同 BSL 工作模式的具体描述，请参见 UART BSL 一章的内容。

### 5.2.4 OCDS 模式

若选择 OCDS 模式，将进入调试模式对程序进行调试。首先初始化 OCDS 硬件，随后跳转到程序存储器的地址  $0000_H$  处，执行 Flash 存储器中的用户代码，开始进行调试。OCDS 模式下，内部数据存储器的最低 64 字节（地址  $00_H - 3F_H$ ）可选择映射至 64 字节的监控 RAM 或内部数据 RAM。

#### 5.2.4.1 备选用户 BSL 模式

备选用户 BSL 模式允许用户执行自订制的 BSL 代码。首先必须将该代码写入 64 字节用户 BSL Flash 扇区中（位于程序存储器的地址段  $FF40_H - FF7F_H$ ）。

若选择备选用户 BSL 模式，Boot ROM 将跳转至程序存储器的地址单元  $FF40_H$ ，执行用户 BSL 代码。

*注：该模式仅在 8 KB Flash 产品中可用。*

## 6 UART 引导程序加载

XC83x 支持 UART 引导程序加载（BSL）模式，可通过设置相应的 BMI 值进入该模式（参见 Boot 及系统启动一章的描述）。该模式的主要目的是：通过 UART 便捷的编程 / 擦除 Flash 和 XRAM。

UART BSL 模式 包含两个功能部分，由以下两个阶段来表示：

- **阶段 I：** 建立串行连接，并与串行通信联系方（主机）的传送速度（波特率）自动同步。
- **阶段 II：** 和主机进行串行通信。主机控制串行通信，发送一个特殊的报文头信息选择工作模式。可选择的工作模式有：
  - **模式 0：** 将用户程序写入 XRAM
  - **模式 1：** 执行 XRAM 中的用户程序
  - **模式 2：** 将用户程序写入 FLASH<sup>1)</sup>
  - **模式 3：** 执行 FLASH 中的用户程序
  - **模式 4：** 擦除 Flash 扇区中的用户程序<sup>1)</sup>
  - **模式 6：** 设置 4 字节 USER\_ID
  - **模式 A：** 获取 4 字节信息

其它模式执行完毕之后，微控制器将返回阶段 II 并等待来自主机的下一条命令。**模式 1、模式 3 和 模式 6** 除外。

阶段 II 选用的串行通讯方式与阶段 I 接收到的字节有关（在波特率 0x0055 之后），通过该字节选择**单引脚（0xAA）**或**双引脚（0x55）** UART。UART 自动波特率检测程序 通过 LIN 波特率检测特性来计算波特率。UART BLS 模式的端口设置见**表 6-1**。

**表 6-1            UART BSL 模式的端口设置**

器件	UART_INIT_ID	单 / 双引脚	使用的引脚
XC83x	0xAA	单引脚	P3.2 用作 RXD P3.2 用作 TXD
	0x55	双引脚	P3.2 用作 RXD P0.7 用作 TXD

该串行传送为异步模式，串行参数为 8N1（8 位数据、无奇偶校验、1 位停止位）。由于微控制器可在阶段 I 和主机自动同步，因此主机可在一定范围内灵活的改变波特率。

### 6.1 阶段 I：和主机自动串行同步

一旦进入 UART BSL 模式，按照以下步骤建立串行连接、并和串行通信联系方（主机）的传送速度（波特率）进行自动同步：

- **步骤 1：** 初始化串行接口用于接收，初始化定时器用于波特率测量
- **步骤 2：** 等待主机发送的同步分隔域（13 位低电平）和同步字节（0x55）

1) 包括用户 BSL Flash 扇区，仅限于 XC83x-8K 器件。

## UART 引导程序加载

- 步骤 3: 和主机波特率同步
- 步骤 4: 读取 UART\_INIT\_ID (0xAA 或 0x55)，确定该传送方式为单引脚还是双引脚 UART 并初始化相应的引脚
- 步骤 5: 向主机发送应答字节 (55<sub>H</sub>)
- 步骤 6: 进入阶段 II

### 6.1.1 概述

可通过波特率检测特性检测主机波特率，由定时器 T2 来实现。必须执行以下初始化操作：

- 微控制器的串行端口设置为模式 1（8 位 UART，波特率可变）
- 检测波特率范围选择（BCON.BGSEL）定义为 00（缺省值）
- 在引脚 T2EX 的下降沿捕获定时器 T2 数据寄存器的值
- 使能定时器 T2 的外部事件（引脚 T2EX 发生下降沿跳变时，置位标志位 EXF2）
- $f_{T2} = f_{PCLK} / 4$ （T2PRE=010）

UART BSL 的自动波特率检测包括：

- 同步分隔域（13 个位时间，低电平）
- 同步字节（55<sub>H</sub>）
- UART\_INIT\_ID

分隔域用于指示新的一帧开始，它必须至少保持 13 位显性值。在分隔域开始处检测到引脚 T2EX 下降沿跳变时，置位定时器 T2 外部启动使能位（T2MOD.T2RHEN）。当引脚 T2EX 再次发生下降沿跳变时，自动启动定时器 T2 工作。最后查询同步字节结束标志（FDCON.EOFSYN）。若该标志置位，定时器 T2 停止工作。T2 重载 / 捕获寄存器（RC2H/L）中存放 8 位的位时间。随后，自动波特率检测程序计算实际波特率，设置 PRE 和 BG 并激活波特率发生器。以上操作完成之后，在发送应答字节之前（0x55），读取 UART\_INIT\_ID 以确定该传送方式为单引脚还是双引脚 UART。波特率检测的原理如图 6-1 所示。

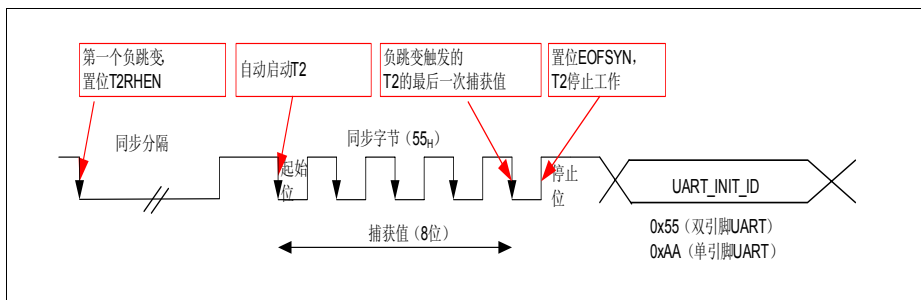


图 6-1 自动波特率检测

### 6.1.2 BG 和 PRE 的计算

建立自动波特率检测时，必须预先载入 **BG** 和 **PRE** 的值。由于共有两个未知值，因此需要两个计算公式。首先，波特率 (**baud**) 和重载值 (**BG**) 的关系和片上外设频率 ( $f_{PCLK}$ ) 有关：

(6.1)

$$\text{baud} = \frac{f_{PCLK}}{16 \left( \text{PRE}_x \left( \text{BRVALUE} + \frac{n}{32} \right) \right)}$$

其次，波特率 (**baud**) 和定时器 **T2** 捕获值 (**T2**) 的关系和 **T2** 的计数频率 ( $f_{T2}$ ) 以及接收位的个数 ( $N_b$ ) 有关：

(6.2)

$$\text{baud} = \frac{f_{T2} \times N_b}{T2}$$

联合公式 (6.1) 和公式 (6.2),  $N_b = 8, f_{T2} = f_{PCLK} / 4$  ( $T2PRE=010$ ) ,  $PRE=1$

(6.3)

$$\frac{f_{PCLK}}{16 \left( \text{PRE}_x \left( \text{BRVALUE} + \frac{n}{32} \right) \right)} = \frac{\frac{f_{PCLK}}{4} \times 8}{T2}$$

简化公式 (6.3) 得到：

(6.4)

$$\left( \text{BRVALUE} + \frac{n}{32} \right) = \frac{T2}{32}$$

捕获 8 位的位时间便于公式用汇编语言实现。除 32 的操作可简单的通过 5 位右移实现。在先前的 **Acropolis** 家族产品中，移位操作会导致小数位损失，从而降低了波特率的准确性。因此，在 **XC83x** 中，实际的移位操作（除法）不必由固件完成，将完全由硬件实现。通过该方式可保留除法结果的小数位，从而提高了波特率检测的准确性。因此，将 8 位的位时间捕获到 **R2CH** 和 **R2CL** 中之后，直接将数值分别编程到 **BGH** 和 **BGL** 中。

设置 **BG** 和 **PRE** 之后，使能波特率发生器工作，随后的数据帧将采用该波特率。接收 **UART\_INIT\_ID** 确定传送方式是单引脚还是双引脚 **UART** 之后，**UART BSL** 程序向主机发送一个应答字节 (**55<sub>H</sub>**)。若该字节接收正确，则可确保双方的串行接口以相同的波特率工作。

## 6.2 阶段 II：串行通信协议和模式

和主机成功同步之后，UART BSL 程序进入阶段 II。在该阶段，微控制器和主机通信，主机选定期望的工作模式。通信协议的详细说明如下：

### 6.2.1 串行通信协议

主机和 UART BSL 程序之间的通信基于简单的传送协议来完成。主机向微控制器按块发送信息。所有的传送块均遵守特定的块结构。通信几乎是单向的，也就是说，主机负责发送传送块，UART BSL 程序仅仅通过返回应答或错误字节对主机作出回应，微控制器本身不发送任何传送块。

不过，也有例外情况 - 在某些模式，微控制器需要向主机发送应答或错误字节之外的其它数据。

#### 6.2.1.1 传送块结构

一个传送块由 3 部分组成：

块类型 (1 字节)	数据区 (X 字节)	校验和 (1 字节)
---------------	---------------	---------------

- **块类型：**传送块的类型，它决定如何解读**数据区**。已实现的块类型有：
  - 00<sub>H</sub> 类型 “报文头”
  - 01<sub>H</sub> 类型 “数据”
  - 02<sub>H</sub> 类型 “传送结束” (EOT)
- **数据区：**一组数据字节。模式 0 和 2 的**数据区**长度不能超过 96/97 字节，这取决于它是数据块还是 EOT。
- **校验和：****块类型**和**数据区**的 XOR 校验和。

每次串行通信时，由主机决定传送块的数量以及相应的长度。出于安全的考虑，每个传送块的最后一个字节是**块类型**和**数据区**的**校验和**。主机将**块类型**和**数据区**的所有字节进行 XOR 产生校验和。UART BSL 程序每次接收到一个传送块后，计算接收字节（**块类型**和**数据区**）的校验和并与接收到的校验和进行比较。

注： 若待写入 Flash 的字节长度小于 1WL，PC 主机必须用 00<sub>H</sub> 填充其余地址单元，传送数据的长度为 32n 个字节（n=1-3）。

#### 6.2.1.2 传送块类型

共有三种**传送块**类型，这和**块类型**的取值有关。表 6-2 给出这些块类型的通用信息，详细说明将在后面的相应章节中给出。

表 6-2 传送块类型

传送块	传送块类型	说明
报文头块	00 <sub>H</sub>	报文头块的长度固定为 8 个字节。该传送块的 <b>数据区</b> 中包含着特殊信息，用于选择不同的工作模式。
数据块	01 <sub>H</sub>	数据块的长度和报文头块中给出的特殊信息有关。该传送块适用于工作模式 0 和 2，用于传送程序代码。程序代码放在数据块的 <b>数据区</b> 。
EOT 块	02 <sub>H</sub>	EOT 块的长度和报文头块中给出的特殊信息有关。在工作模式 0 和 2 中，EOT 块是数据传送的最后一个块。要传送的最后的程序代码放在 EOT 块的 <b>数据区</b> 。

### 6.2.1.3 返回主机的回应码

微控制器应发送回应码，通知主机是否已成功接收块信息。若接收正确，发送应答码（55<sub>H</sub>）。若接收失败，有两种错误的可能。一种是块类型错误，此时 UART BSL 将向主机返回块类型错误码（0FF<sub>H</sub>）。两种情况可导致块类型错误：一种情况是微控制器接收的块类型和已实现的块类型不同；另一种情况是微控制器接收的传送块顺序有误。比如：在工作模式 0 下，若接收到报文头块之后，紧接着接收到的不是数据块，而是另一个报文头块，微控制器则把这种情况视为块类型错误。另一种错误是校验和错误。若校验和比较失败，UART BSL 程序向主机返回校验和错误码（0FE<sub>H</sub>），拒绝接收。出现这两种错误时，UART BSL 程序将等待主机重新发送正确的块信息。

由于 XC83x 中没有 Flash 保护，因此不存在 Flash 保护错误。UART BSL 不负责检查 XRAM/Flash 的地址是否有效，由用户来确保 Flash/XRAM 的 DPTR 有效。

微控制器返回主机的回应码类型归纳见表 6-3。

表 6-3 回应码类型

通信状态	返回主机的回应码
成功	55 <sub>H</sub>
块类型错误	0FF <sub>H</sub>
校验和错误	0FE <sub>H</sub>

表 6-4 列出微控制器在接收到报文头块、数据块或 EOT 块之后可能返回的各种回应码。

表 6-4            不同块类型的回应码

模式	报文头块	数据块	EOT 块
0	应答，块类型错误，校验和错误	应答，块类型错误，校验和错误	应答，块类型错误，校验和错误
1	应答，块类型错误，校验和错误		
2	应答，块类型错误，校验和错误	应答，块类型错误，校验和错误	应答，块类型错误，校验和错误
3	应答，块类型错误，校验和错误		
4	应答，块类型错误，校验和错误		
6	应答，块类型错误，校验和错误		
A	应答，块类型错误，校验和错误		
E	应答，块类型错误，校验和错误		

回应码的定义见表 6-5，表中列出导致回应码出错的各种原因，并给出相应的更正操作。

表 6-5            回应码的定义

回应码	值	说明			更正操作
		块类型	BSL 模式	原因 / 含义	
应答	55 <sub>H</sub>	报文头	1, 3	发送回应码之后立刻执行所请求操作。	
			6, A	所请求操作已成功执行完毕。	
		EOT	0, 2, 4		
		所有其它类型		成功接收块信息。在模式 A 随后将发送 ID。做好接收下一个信息块的准备。	
块类型错误	FF <sub>H</sub>	所有其它类型		块类型未定义或传送块顺序有误。	重新发送有效的块信息
校验和错误	FE <sub>H</sub>	所有类型		计算的校验和与接收的校验和不匹配。	重新发送有效的块信息



### 6.2.2 工作模式选择

当 UART BSL 程序进入阶段 II，它首先等待主机发送的一个 8 字节、包含工作模式选择信息的报文头块。UART BSL 程序根据该信息选择并激活期望的工作模式。若微控制器接收到一个错误的报文头块，UART BSL 程序将向主机发送校验和错误或块类型错误码，等待新的报文头块。在这种情况下，主机将重新发送报文头块或发布消息。

#### 6.2.2.1 接收报文头块

每次进行数据通信时，报文头块始终是主机发送的第一个传送块。报文头块中包含工作模式编号以及相关模式的特殊信息（参见“模式数据”）。其通用结构如下：

块类型 00 <sub>H</sub> (报文头块)	数据区		校验和 (1 字节)
	模式 (1 字节)	模式数据 (5 字节)	

说明：

- 00<sub>H</sub>：块类型，标明该信息块为**报文头块**
- 模式：工作模式。已实现的工作模式有：
  - 00<sub>H</sub> (模式 0：将用户程序写入 XRAM)
  - 01<sub>H</sub> (模式 1：执行 XRAM 中的用户程序)
  - 02<sub>H</sub> (模式 2：将用户程序写入 FLASH)
  - 03<sub>H</sub> (模式 3：执行 FLASH 中的用户程序)
  - 04<sub>H</sub> (模式 4：擦除 Flash 扇区中的用户程序)
  - 06<sub>H</sub> (模式 6：设置 4 字节 USER\_ID)
  - 0A<sub>H</sub> (模式 A：获取 4 字节信息)
- 模式数据：5 字节的特殊信息，激活相应工作模式的必备信息
- 校验和：报文头块的校验和

#### 6.2.2.2 模式 0：将用户程序写入 XRAM

模式 0 用于主机向微控制器的 XRAM 传送用户程序。

报文头块的结构如下：

报文头块

00H (报文头)	00H (模式0)	数据区				校验和 (1字节)
		起始地址 高位 (1字节)	起始地址 低位 (1字节)	块长度 (1字节)	未使用 (2字节)	

模式数据说明:

**起始地址高位, 低位:** 16 位起始地址, 决定将接收到的程序复制到 XRAM 中的什么位置。

**块长度:** 每次传送的数据块或 EOT 块的总长度。

**未使用:** 2 字节, 这些字节未使用, 在模式 0 中被忽略。

注: **块长度**是指报文头块之后的传送块 (数据块或 EOT 块) 的总长度 (块类型、数据区和校验和)。每次传送的数据块和 EOT 块的长度应该相同。

注: 数据块 (在报文头块之后, 发送 1-255 个数据块, 最后发送 1 个 EOT 块) 的最大长度为  $96+2 = 98$  个字节。

注: EOT 块 (在报文头块之后, 每次只发送 1 个 EOT 块) 的最大长度为  $= 96 + 3 = 99$  个字节。

微控制器在成功接收报文头块之后进入模式 0, 主机向微控制器传送程序代码, 即传送数据块和 EOT 块, 说明如下:

注: 不允许传送空数据块。

数据块

01H (数据块) (1字节)	程序代码 ( (块长度-2) 个字节)	校验和 (1字节)
-----------------------	------------------------	--------------

说明:

**程序代码:** 程序代码的长度为 (块长度-2) 个字节, 块长度由前面的报文头决定。

### EOT 块

02 <sub>H</sub> (EOT 块) (1 字节)	最后的代码 长度 (1 字节)	程序代码	未使用	校验和 (1 字节)
--------------------------------------	-----------------------	------	-----	---------------

说明:

**最后的代码长度:** 该字节指示该 EOT 块中所含程序代码的长度。

**程序代码:** 要发送给微控制器的最后的程序代码。

**未使用:** 长度为 (块长度 -3- 最后的代码长度)。这些字节未使用, 可设置为任意值。

### 6.2.2.3 模式 1: 执行 XRAM 中的用户程序

模式 1 用于执行 XRAM 中起始地址为 0F'000<sub>H</sub> 的用户程序。报文头块的结构如下:

#### 报文头块

00 <sub>H</sub> (报文头块)	01 <sub>H</sub> (模式 1)	数据区	校验和 (1 字节)
		未使用	

模式数据说明:

**未使用:** 未使用这 5 个字节, 在模式 1 下被忽略。

对于工作模式 1, 报文头块是主机发出的唯一传送块, 无需进一步的串行通信。微控制器随后将退出 UART BSL 模式, 跳转至 XRAM 的地址单元 0F000<sub>H</sub>。

### 6.2.2.4 模式 2: 将用户程序写入 FLASH

模式 2 用于主机向微控制器的 Flash 传送用户程序。

报文头块的结构如下:

#### 报文头块

00 <sub>H</sub> (报文头)	02 <sub>H</sub> (模式 2)	数据区				校验和 (1 字节)
		起始地址 高位 (1 字节)	起始地址 低位 (1 字节)	块长度 (1 字节)	未使用 (2 字节)	

UART 引导程序加载

模式数据说明:

**起始地址高位, 低位 :** 16 位起始地址, 决定将接收到的程序复制到 Flash 中的什么位置。该地址必须有效并和页地址对齐。

**块长度:** 每次传送的数据块或 EOT 块的长度。每次主机可发送最小 1WL (32 字节) 和最大 3WL (96 字节)。若要发送数据块, 最大块长度为 98 (=96+2) 个字节; 若只发送 EOT 块, 最大块长度为 99 (=96+3) 个字节。

**未使用:** 2 字节, 这些字节未使用, 在模式 2 中被忽略。

*注:* 若数据从非分页地址处开始, PC 主机将用 00<sub>H</sub> 填充前面的地址单元并提供该页地址的**起始地址**。比如: 若数据从 0F82<sub>H</sub> 开始, PC 主机将用 00<sub>H</sub> 填充地址 0F80<sub>H</sub> 和 0F81<sub>H</sub> 并向微控制器提供**起始地址** 0F80<sub>H</sub>。若数据仅有 8 个字节, PC 主机将用 00<sub>H</sub> 填充剩余的地址单元, 传送 32n 个数据字节。

*注:* **块长度**是指报文头块之后的传送块 (数据块或 EOT 块) 的总长度 (块类型、数据区和校验和)。由于数据区长度是 32 字节的整数倍, 因此, 加上块类型和校验和字节, **块长度**为 32n+2 个字节。

*注:* 可通过该模式编程用户 BSL Flash 扇区, 这仅限于 XC83x-8K 器件。

微控制器在成功接收报文头块之后进入模式 2, 主机向微控制器传送程序代码, 即传送数据块和 EOT 块, 说明如下:

*注:* 不允许传送空数据块。

数据块

01 <sub>H</sub> (数据块) (1字节)	程序代码 ( (块长度-2) 个字节)	校验和 (1字节)
-----------------------------------	------------------------	--------------

说明:

**程序代码:** 程序代码的长度为 (块长度 -2) 个字节, 块长度由前面的报文头决定。

EOT 块

02 <sub>H</sub> (EOT 块) (1字节)	最后的代码 长度 (1字节)	程序代码	未使用	校验和 (1字节)
-------------------------------------	----------------------	------	-----	--------------

说明:

**最后的代码长度:** 该字节指示该 EOT 块中所含程序代码的长度。

## UART 引导程序加载

注： 若发送的是数据块，该字节应为 0。若该字节不为 0，附加的不期望字节将被写入。

程序代码：要发送给微控制器的最后的程序代码。

未使用：长度为（块长度 -3- 最后的代码长度）。这些字节应写入 0。

### 6.2.2.5 模式 3：执行 FLASH 中的用户程序

模式 3 用于执行 Flash 中起始地址为 0000<sub>H</sub> 的用户程序。报文头块的结构如下：

#### 报文头块

00 <sub>H</sub> (报文头块)	03 <sub>H</sub> (模式 3)	数据区	校验和 (1 字节)
		未使用	

模式数据说明：

未使用：未使用这 5 个字节，在模式 3 下被忽略。

对于工作模式 3，报文头块是主机发出的唯一传送块，无需进一步的串行通信。微控制器随后将退出 UART BSL 模式，跳转至 Flash 的地址单元 0000<sub>H</sub>。

### 6.2.2.6 模式 4：擦除 Flash 扇区中的用户程序

模式 4 用于擦除 Flash Bank 0<sup>1)</sup>和 Flash Bank 1<sup>2)</sup>中的不同扇区。它还支持擦除用户 BSL Flash 扇区<sup>2)</sup>。

#### 报文头块

00 <sub>H</sub> (报文头块)	04 <sub>H</sub> (模式 4)	数据区					校验和 (1 字节)
		Flash_B0_SectorL (1 字节)	Flash_B0_SectorH (1 字节)	Flash_B1_SectorL (1 字节)	Flash_B1_SectorH (1 字节)	未使用 (1 字节)	

模式数据说明：

**Flash\_B0\_SectorL**：Flash Bank 0 的扇区 0-7 分别由该字节的位 0-7 代表。若某位为 1，则表示选中相应的扇区。比如：若该字节为 0x12，表明将删除 Flash Bank 0 的扇区 1 和 4。

1) 适用于 XC83x-4K 和 XC83x-8K 器件。

2) 仅限于 XC83x-8K 器件。

## UART 引导程序加载

**Flash\_B0\_SectorH** : Flash Bank 0 的扇区 8-9 分别由该字节的位 0-1 代表。若某位为 1, 则表示选中相应的扇区。比如: 若该字节为 0x01, 表明将删除 Flash Bank 0 的扇区 8。

**Flash\_B1\_SectorL** : Flash Bank 1 的扇区 0-7 分别由该字节的位 0-7 代表。若某位为 1, 则表示选中相应的扇区。比如: 若该字节为 0x12, 表明将删除 Flash Bank 1 的扇区 1 和 4。

**Flash\_B0\_SectorH** : Flash Bank 1 的扇区 8-9 分别由该字节的位 0-1 代表。若某位为 1, 则表示选中相应的扇区。比如: 若该字节为 0x05, 表明将删除 Flash Bank 1 的扇区 8 和用户 BSL Flash 扇区。

注: 不需要/未选中的位应清零。

注: *Flash\_B1\_SectorL* 和 *Flash\_B1\_SectorH* 仅在 XC83x-8K 器件中才可用。对于其它配置, 这两个字节被忽略、不使用。

### 6.2.2.7 模式 6: 设置 4 字节 USER\_ID

模式 6 用于设置用户 ID, 即 USER\_ID (4 字节)。报文头块的结构如下:

#### 报文头块

00 <sub>H</sub> (报文头块)	06 <sub>H</sub> (模式6)	数据区					校验和 (1字节)
		USER_ID_ 3 (1字节)	USER_ID_ 2 (1字节)	USER_ID_ 1 (1字节)	USER_ID_ 0 (1字节)	未使用 (1字节)	

#### 模式数据说明:

**User\_ID**: 这 4 个字节由用户定义, 用于 BMI 和初始化设置。

**未使用**: 未使用该字节, 在模式 6 下被忽略。

设置 **USER\_ID** 之后, 向主机返回应答响应 (指示操作成功), 将触发**软件复位**从而进入设定的启动模式。

注: 需谨慎操作和执行该模式。应最后执行该模式: 用户应首先写入 *Flash* 代码, 然后写入 *XRAM* 代码 (如有需要), 最后设置 **USER\_ID**。

注: 不提供擦除 **USER\_ID** 的选项。若要擦除 **USER\_ID**, 只需向 4 字节 **USER\_ID** 中写入 0x00 即可。

### 6.2.2.8 模式 A: 获取 4 字节信息

模式 A 用于获取 4 字节数据, 该数据的内容由报文头块中的选择字节决定。报文头块的结构如下:

报文头块

00 <sub>H</sub> (报文头块)	0A <sub>H</sub> (模式A)	数据区		校验和 (1字节)
		未使用 (4字节)	选择 (1字节)	

模式数据说明:

**选择:** 该字节决定将要发送给主机的 4 字节数据。00<sub>H</sub> - 02<sub>H</sub> 为有效选择。

00<sub>H</sub> - 芯片 ID (MSB 字节 1... LSB 字节 4)

01<sub>H</sub> - USER\_ID (MSB 字节 1... LSB 字节 4)

在模式 A 中，报文头块是主机发出的唯一传送块。如果成功接收报文头块，微控制器将向主机返回应答字节，随后是 4 字节数据。如果接收到无效的选择字节，微控制器将返回 4 字节 00<sub>H</sub>。USER\_ID 是用户 ID，4 字节 USER\_ID\_INFO 的顺序为: USER\_ID\_4, USER\_ID\_3, USER\_ID\_2 和 USER\_ID\_1。

## 7 系统控制单元

XC83x 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：

- 嵌入式电压调节器（EVR）（见 [页 7-1 上的章节 7.1](#)）
- 复位控制（见 [页 7-6 上的章节 7.2](#)）
- 时钟系统和控制（见 [页 7-11 上的章节 7.3](#)）
- 功率管理（见 [页 7-16 上的章节 7.4](#)）

XC83x 提供了一系列在紧急条件下（如欠压）确保系统性能的特性。

XC83x 时钟系统的核心是时钟产生单元（CGU），通过 48 MHz 振荡器产生主时钟频率。从主时钟得到的相位同步时钟信号被分配到整个系统中。

### 7.1 带有嵌入式电压调节器的电源系统

内核、存储器和外设的供电电压由嵌入式电压调节器（EVR）来调节；EVR 内部的检测电路可确保供电电压在规定的工作电压范围内。EVR 的主电压调节器和低功率电压调节器可被分别关闭，从而在不同的省电模式下降低功耗。

XC83x 微控制器需要两种不同电平的电源：

- 嵌入式电压调节器（EVR）和端口需 2.5 V - 5.5 V 供电
- 内核、存储器、片内振荡器和外设需 2.5 V 供电

XC83x 的电源系统如 [图 7-1](#) 所示。必须由外部电源引脚提供 2.5 V - 5.5 V 的电源。由 EVR 产生 2.5 V 电源。内嵌 EVR 有助于降低整个芯片的功耗及应用板设计的复杂度。



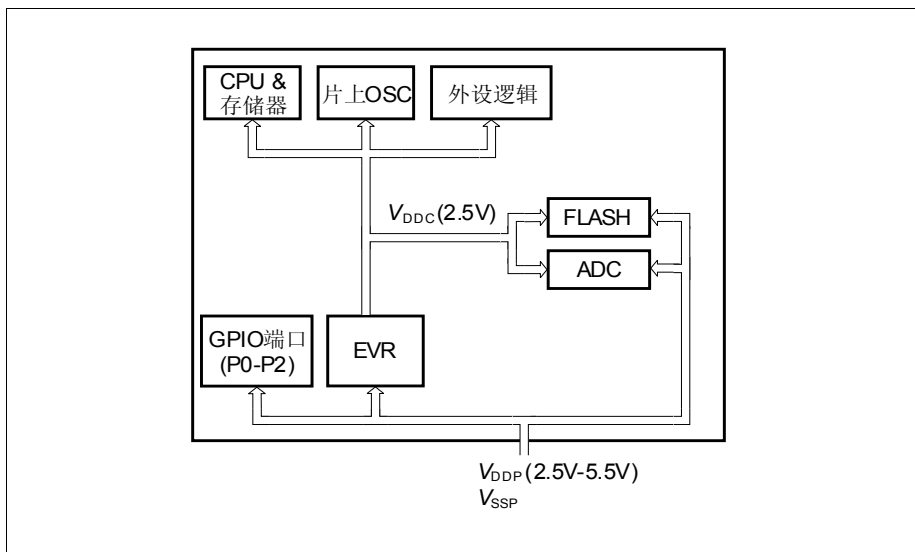


图 7-1 XC83x 电源系统

### EVR 特性:

- 输入电压 ( $V_{DDP}$ ): 3.0 V - 5.5 V @ 全电压工作条件
- 在低负载的正常工作模式或掉电模式下, 输入电压 ( $V_{DDP}$ ) 降至 2.5 V @ 低压工作条件
- 输出电压 ( $V_{DDC}$ ): 2.5 V +/- 7.5%
- 掉电模式下提供低功率电压调节器
- $V_{DDC}$  和  $V_{DDP}$  预警检测
- $V_{DDC}$  欠压检测

EVR 由一个主电压调节器和一个低功率电压调节器组成。正常工作模式下, 两个电压调节器均被使能。掉电模式下, 主电压调节器关闭, 低功率电压调节器继续工作, 以低功耗为系统供电。

在全电压工作条件下, EVR 的输入电压范围是 3.0 V - 5.5V; 在低压工作条件下, EVR 的输入电压范围降至 2.5 V。在低压工作条件下, XC83x 只工作在带有有限负载的正常工作模式或掉电模式。

EVR 包含  $V_{DDC}$  和  $V_{DDP}$  检测器。进行  $V_{DDC}$  检测时有两个阈值电压: 预警电压 (2.4 V) 和欠压复位电压 (2.3 V)。当  $V_{DDC}$  低于 2.4 V 时,  $V_{DDC}$  NMI 标志位 NMISR.FNMIVDDC 置位, 若允许产生  $V_{DDC}$  NMI 中断 (NMICON.NMIVDDC), 则向 CPU 发送该 NMI 中断请求。SDCON.VDDCPW 缺省设置禁用预警检测, 在低压工作条件下, 如此设置尤其必要。若允许进行预警检测, 该位必须置 1。若  $V_{DDC}$  低于 2.3 V, 将会激活欠压复位, 微控制器进入复位状态。掉电模式下,  $V_{DDC}$  低于 1.5 V 时产生欠压复位。

进行  $V_{DDP}$  检测时也有两个阈值电压：预警电压（3.6 V）和欠压复位电压（2.9 V）。当  $V_{DDP}$  低于 3.6 V 时， $V_{DDP}$  NMI 标志位 NMISR.FNMIVDDP 置位，若允许产生  $V_{DDP}$  NMI 中断（NMICON.NMIVDDP），则向 CPU 发送该 NMI 中断请求。若  $V_{DDP}$  低于 2.9 V，将会激活欠压复位，微控制器进入复位状态。正常工作模式下，特别是低压工作条件下，可通过寄存器 SDCON 中的位 VDDPPW 和 VDDPBOA 禁用这两种阈值检测。掉电模式下自动关闭这两种检测，不过，若 VDDPBOPD 置 1， $V_{DDP}$  低于 3.6 V 时产生欠压复位。

$V_{DDP}$  和  $V_{DDC}$  除欠压阈值电平之外， $V_{DDP}$  低于 2.4 V 时 EVR 进入复位模式。若通过位 VDDPBOA 和 VDDPBOPD 禁用正常工作模式或掉电模式下的  $V_{DDP}$  欠压检测，可发生上述情况。

### 7.1.1 低压工作条件

在  $2.5\text{ V} < V_{DDP} < 3.0\text{ V}$  的低压工作条件下，工作电流必定低于一组值，这和 EVR 的驱动能力有关，具体极限值可参见数据手册。根据  $V_{DDP}$  的实际电压，电流消耗须低于规定值。若条件不满足，则可能触发欠压复位。主要模块的电流消耗情况可参见数据手册。

注：XC83x 的全电压工作条件为  $3\text{ V} < V_{DDP} < 5.5\text{ V}$ 。

### 7.1.2 EVR 寄存器

SDCON 用于使能或禁用各检测器。此外，该寄存器还给出  $V_{DDP}$  和  $V_{DDC}$  阈值电平的状态。必须在访问该寄存器之前设置 SCU\_PAGE.PAGE。

#### SDCON

电压检测控制寄存器

(EE<sub>H</sub>)

复位值：34<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	VDDPTH	VDDCTH	VDDPBOPD	VDDPBOA	VDDPPW	VDDCPW	
r	rh	rh	rw	rw	rw	rw	

符号	位	类型	描述
VDDCPW	0	rw	<b><math>V_{DDC}</math> 预警检测使能</b> 0 禁止 $V_{DDC}$ 预警检测。 1 使能 $V_{DDC}$ 预警检测。 <i>注： 只有当该位被置 1 时，才能触发 VDDC 预警标志和 NMI。</i>
VDDPPW	1	rw	<b><math>V_{DDP}</math> 预警检测使能</b> 0 禁止 $V_{DDP}$ 预警检测。 1 使能 $V_{DDP}$ 预警检测。 <i>注： 只有当该位被置 1 时，才能触发 VDDP 预警标志和 NMI。</i>
VDDPBOA	2	rw	<b>使能正常工作模式下的 <math>V_{DDP}</math> 欠压检测</b> 0 禁止正常工作模式下的 $V_{DDP}$ 欠压检测。 1 使能正常工作模式下的 $V_{DDP}$ 欠压检测。
VDDPBOPD	3	rw	<b>使能掉电模式下的 <math>V_{DDP}</math> 欠压检测</b> 0 禁止掉电模式下的 $V_{DDP}$ 欠压检测。 1 使能掉电模式下的 $V_{DDP}$ 欠压检测。
VDDCTH	4	rh	<b><math>V_{DDC}</math> 阈值指示</b> 0 低于 $V_{DDC}$ 预警阈值电平。 1 高于 $V_{DDC}$ 预警阈值电平。 <i>注： 该位不受位 VDDCPW 的影响。</i>
VDDPTH	5	rh	<b><math>V_{DDP}</math> 阈值指示</b> 0 低于 $V_{DDP}$ 预警阈值电平。 1 高于 $V_{DDP}$ 预警阈值电平。 <i>注： 该位不受位 VDDPPW 的影响。</i>

符号	位	类型	描述
0	[7:6]	r	保留 读取返回 0；应写入 0。

## 7.2 复位控制

XC83x 有五种复位方式：上电复位、看门狗定时器复位、软复位、掉电唤醒复位和欠压复位。

当 XC83x 首次上电、或输入电压低于阈值电压导致欠压时，系统复位之后，必须先达到合适的电压阈值才能开始工作。CPU 随后开始执行 Boot ROM 固件。

检测到系统出现故障时，看门狗定时器（WDT）也能复位器件。

另一种需要检测的复位是器件掉电模式下的复位（唤醒复位）。上电复位后静态 RAM 的内容未被定义；而掉电唤醒复位后，静态 RAM 的内容完好保留。

若使能和  $V_{DDC}$  和  $V_{DDP}$  的欠压检测， $V_{DDC}$  电压跌至 2.3 V 以下或  $V_{DDP}$  电压跌至 2.9 V 以下时触发欠压复位。

应用软件可根据需要触发软复位。

### 7.2.1 复位类型

#### 7.2.1.1 上电复位

电源电压  $V_{DDP}$  用于芯片上电。上电后首先复位 EVR 模块，包括：

1. 启动主电压调节器和低功率电压调节器
2. 当  $V_{DDP}$  和  $V_{DDC}$  达到  $V_{DDP}$  和  $V_{DDC}$  检测器的阈值电压时，EVR 复位变为无效

系统启动后，使用 48 MHz 的振荡器产生系统频率，器件以 8 MHz 频率工作。EVR 稳定之后，一旦 48 MHz 的振荡器达到稳定，在释放系统的复位信号之前 Flash 必须进入读取准备就绪模式。用户模式下，可由 Boot ROM 中的启动固件将系统时钟切换到用户可选的时钟模式（8 MHz 或 24 MHz）。由引脚 P0.4 指示器件复位（复位指示引脚）。

系统复位后，启动固件开始工作，它将对 USER\_ID 中的信息，如 boot 模式（用户模式、BSL 模式、OCDS 模式）和系统时钟频率进行译码。在进入选中的 boot 模式之前，启动固件将根据译码值进行必要的设置。

注：  $V_{DDP}$  未上电时，任一 GPIO 引脚流入的电流决不能使  $V_{DDP}$  的电压高于 0.3 - 0.5 V。

#### 7.2.1.2 看门狗定时器复位

看门狗定时器复位是一种内部复位。看门狗定时器（WDT）中的计数器必须被周期刷新或清零。如果 WDT 没有被及时、正确刷新，它将向 CPU 发送 NMI 中断请求，在预先定义的一个超时周期之后对系统复位。位 RSTCON.WDTRST 用来指示 WDT 的复位状态。

对于 WDT 复位而言，由于 EVR 和 48 MHz 振荡器已稳定工作，因此 WDT 复位比其它类型复位所需的时间短。

### 7.2.1.3 软复位

软复位是一种内部复位，软件置位 RSTCON 寄存器中的软复位请求位 SWRQ 触发该复位。应用软件可根据需要触发软复位。位 RSTCON.SOFTRS 用来指示软复位状态。

### 7.2.1.4 掉电唤醒复位

掉电模式下，低功率电压调节器仍然工作，XC83x 仍有供电。如果进入掉电状态的时间恰当，所有重要的系统状态将由软件保留在 Flash 中。

如果 XC83x 处于掉电模式，有三种唤醒方式供选择：

- 由 RTC 唤醒事件唤醒（和掉电模式的类型有关）
- 由 EXINT0 唤醒
- 由 RTC 时钟源出错唤醒

通过控制位 PMCON0.EWS 选择由 EXINT0 引脚唤醒系统。根据掉电模式的类型不同，RTC 唤醒事件和 RTC 时钟出错也可用作唤醒源。掉电唤醒可以带复位或不带复位，这由 PMCON0.WKSEL 选择。位 RSTCON.WKRS 指示唤醒状态（带复位或无复位）。

由于 EVR 只需较短的时间即可达到稳定，因此掉电唤醒复位比上电复位所需的时间短。

### 7.2.1.5 欠压复位

正常工作模式下，当内核电压跌至以下某电压时，EVR 中的  $V_{DDC}$  检测器检测到欠压：

- 阈值电压  $V_{DDC\_TH}$ （约 2.3 V）或
- 若使能检测器工作，阈值电压  $V_{DDP\_TH}$ （约 2.9 V）或
- 若禁止检测器工作，阈值电压  $V_{DDP\_TH}$ （约 2.4 V）

欠压将引发器件复位。掉电模式下，由 EVR 中的 POR 监控  $V_{DDC}$ 。当  $V_{DDC}$  跌至 1.5 V 以下时发生复位。

一旦发生欠压复位，该复位过程和上电复位过程相同。

### 7.2.2 模块复位行为

**表 7-1** 给出不同复位类型对 XC83x 各功能单元的影响。符号 "■" 表示该功能被复位至相应的缺省值。

**表 7-1 复位对模块 / 器件功能的影响**

模块 / 功能	上电复位	欠压复位	唤醒复位	软复位	WDT 复位
CPU	■	■	■	■	■
SCU	■	■	■ 指示位除外	■ 指示位除外	■ 指示位除外
外设	■	■	■	■	■
调试系统	■	■	■	■	■

表 7-1      复位对模块 / 器件功能的影响

模块 / 功能	上电复位	欠压复位	唤醒复位	软复位	WDT 复位
端口控制	■	■	■	■	■
启动 FW	初始化	初始化	初始化	初始化	初始化
片内静态 RAM	受影响，不可靠	受影响，不可靠	不受影响，可靠	不受影响，可靠	不受影响，可靠
Flash	■	■	■	■	■
EVR	■	■	■	■	■
时钟系统	■	■	■	■	■

### 7.2.3 复位控制寄存器

RSTCON 寄存器中存放唤醒事件、WDT 复位和软复位的指示位。发生这些事件时，RSTCON 的相应复位值归纳见**表 7-2**。

#### RSTCON

复位控制寄存器

(F7<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
SWRQ		0			SOFTRS	WDTRST	WKRS
rwh		r			rwh	rwh	rwh

符号	位	类型	描述
WKRS	0	rwh	<b>唤醒指示位</b> 0 未发生唤醒 1 已发生唤醒 该位只能由硬件置位，软件清零。
WDTRST	1	rwh	<b>WDT 复位指示位</b> 0 未发生 WDT 复位 1 已发生 WDT 复位 该位只能由硬件置位，软件清零。
SOFTRS	2	rwh	<b>软复位指示位</b> 0 未发生软复位 1 已发生软复位 该位只能由硬件置位，软件清零。
SWRQ	7	rwh	<b>软复位请求</b> 0 无操作 1 请求软复位 该位由硬件自动清零。 SWRQ 是受保护位。保护方案激活时，不能直接对该位进行写操作。
0	[6:3]	r	<b>保留</b> 读取返回 0；应写入 0。



表 7-2      寄存器 RSTCON 的复位值

复位源	复位值
掉电唤醒复位	0000 0001 <sub>B</sub>
WDT 复位	0000 0010 <sub>B</sub>
软复位	0000 0100 <sub>B</sub>
上电复位 / 欠压复位	0000 0000 <sub>B</sub>



正常工作模式下，CPU 时钟（CCLK, SCLk）和外设时钟（PCLK）有两种频率选择：8 MHz 和 24 MHz。通过 Boot ROM 中的用户程序 BR\_CLKMODE\_SETTING 切换时钟。若用户程序的输入参数 CLKMODE 设置为 0，选择 8 MHz；设置为 1，选择 24 MHz。在修改 CPU 时钟和 PCLK 时钟的过程中，建议禁用所有中断以避免对 Flash 进行访问，否则可能导致失败的 Flash 操作。若 CPU 工作在 24 MHz，读取 Flash 时需插入一个等待状态；CPU 工作在 8 MHz 时，无需插入等待状态。

空闲模式下，只关闭 CPU 时钟 CCLK。掉电模式下，CCLK, SCLK, FPCLK, SPCLK, CCLKn 和 PCLK 均关闭。

除 48 MHz 片内振荡器之外，XC83x 还内置一个 75 kHz 振荡器。当 48 MHz 时钟丢失时，该 75 kHz 振荡器用作二级系统时钟，它还可用作 WDT 的输入时钟。

此外，XC83x 需要一个 32.768 kHz 的片外振荡器（经振荡器引出脚连接），从而通过实时时钟模式 0 支持计时功能。

### 7.3.1 振荡器看门狗

XC83x 中有三种振荡器看门狗：48 MHz 振荡器看门狗（48 MHz OWD）、75 kHz 振荡器看门狗（75 kHz OWD）和 32.768 kHz 片外振荡器看门狗（32.768 kHz OWD）。48 MHz OWD 监控 48 MHz 时钟源，低于 43 MHz（典型值）的输入频率被检测为“振荡器频率过低”。75 kHz OWD 监控 75 kHz 时钟源，低于 35 kHz（典型值）被检测为“频率过低”。置位 OSC\_CON.RCOWDRST 可重新启动 48 MHz 和 75 kHz 振荡器检测。13 个 75 kHz 的时钟周期之后（约 180 us），检测状态输出才有效。第三种振荡器看门狗监控 32.768 kHz 的片外时钟源。

复位后缺省禁用 48 MHz 和 75 kHz OWD。在使能 OWD 之前，用户必须检查标志位 OSC\_CON.INTOSC\_ST 的状态。若标志位 INTOSC\_ST 为 1，75 kHz 振荡器应已稳定工作。随后，可通过置位 OSC\_CON.RCOWDRST 使能 OWD 功能。建议当 48MOSC2L 和 75KOSC2L 均为 0 时继续执行用户程序。用户还可选择在等待 48MOSC2L 和 75KOSC2L 稳定之前执行其它操作。当 48MOSC2L 或 75KOSC2L 为 1 时，若控制位 NMICON.NMIOSCCLK 使能中断，则会产生振荡器时钟丢失 NMI。

当 32.768 kHz 振荡器的频率低于 16 kHz（典型值）时，相应的振荡器看门狗可检测到时钟丢失。

### 7.3.2 时钟丢失检测及恢复

器件正常工作期间，48 MHz 振荡器看门狗检测到低于 43 MHz 的标称输入频率时，发生时钟丢失。在这种情况下，若控制位 NMICON.NMIOSCCLK 使能中断，将产生 NMI 中断。此时，振荡器状态标志 48MOSC2L 置 1，由 75 kHz 振荡器提供系统时钟，可使用该振荡器频率执行紧急程序。

下次上电复位或成功执行时钟恢复之前，XC83x 始终保持时钟丢失状态。可通过置位 OSC\_CON.RCOWDRST 重新启动检测以进行时钟恢复。检测到振荡器频率稳定且高于 43 MHz 之后，位 48MOSC2L 清零，系统时钟将自动切换到 48 MHz 的时钟源。

检测到 75 kHz 振荡器的时钟丢失时，75 kHz 振荡器看门狗状态标志 75KOSC2L 置 1。若控制位 NMICON.NMIOSCCLK 使能中断，将产生 NMI 中断。此时，将不会切换到 48

MHz 振荡器。不过，48 MHz 振荡器看门狗不起作用，不能监控该主时钟。可使用 48 MHz 时钟执行紧急程序。

*注： 时钟丢失状态下以 75 kHz 作为系统频率时，可以读取 Flash。不过不允许编程和擦除 Flash。*

*注： 48 MHz 或 75 kHz 振荡器发生时钟丢失时，通过 NMICON.NMIOSCCLK 使能时钟丢失 NMI 中断。*

当 32.768 kHz 振荡器看门狗检测到低于 16 kHz 的标称输入频率时，若控制位 NMICON.NMIXTALCLK 使能中断，将产生 NMI 中断，同时片外振荡器状态标志 XTAL2L 将置 1。此时，RTC 模块若工作在计时模式，则会导致时钟丢失。建议用户终止 RTC 操作或为 RTC 选择其它时钟源（因为计时模式将无法正常工作）。

### 7.3.3 32.768 kHz 振荡器时钟的启动控制

在 XC83x 中，使用 32.768 kHz 的片外时钟进行计时。必须执行以下步骤使能该时钟：

- OSC\_CON.XPD 置 0 以使能 32.768 kHz 振荡器引出脚。
- 等待 2 s 至振荡稳定。
- 使能 32.768 kHz OWD 之前，确保 75 kHz 振荡器达到稳定（OSC\_CON.INTOSC\_ST = 1）。
- OSC\_CON.XTALOWDRST 置 1 以重新启动 32.768 kHz OWD。
- 当 XTAL2L 为 0 时，选择实时时钟模式 0（计时模式）。

### 7.3.4 CCU 寄存器

复位（任意类型）之后，时钟控制单元的所有寄存器复位到缺省值。

寄存器 OSC\_CON 控制 48 MHz、75 kHz 和 32.768 kHz 振荡器看门狗。

**OSC\_CON**  
**OSC 控制寄存器** (F4<sub>H</sub>) 复位值: 30<sub>H</sub>  
**RMAP: 0, PAGE: 1**

7	6	5	4	3	2	1	0
0	INTOSC_ST	XPD	XTAL2L	XTALOWD_RST	75KOSC_2L	48MOSC_2L	RCOWD_RST
r	rh	rw	rh	rwh	rh	rh	rwh

符号	位	类型	描述
RCOWDRST	0	rwh	<b>48 MHz 和 75 kHz 振荡器看门狗复位</b> 置位该位将重新启动振荡器检测。该位将自动复位到 0，因此读取始终返回 0。 0 不起作用 1 重新启动 48 MHz 和 75 kHz 振荡器看门狗。 48MOSC2L 和 75KOSC2L 标志将保持最近一次的值，180 us 后才显示更新值。
48MOSC2L	1	rh	<b>48 MHz 振荡器频率过低标志</b> 振荡器看门狗监控 $f_{SYS}$ （等于 $f_{OSC}$ ）。由片内 48 MHz 振荡器产生系统频率。 0 $f_{SYS}$ 高于阈值。 1 $f_{SYS}$ 低于阈值。
75KOSC2L	2	rh	<b>75 kHz 振荡器频率过低标志</b> 振荡器看门狗监控 75 kHz 振荡器频率。 0 75 kHz 振荡器频率高于阈值。 1 75 kHz 振荡器频率低于阈值。
XTALOWDRST	3	rwh	<b>32.768 kHz 片外振荡器看门狗复位</b> 置位该位将重新启动振荡器检测。该位将自动复位到 0，因此读取始终返回 0。 0 不起作用 1 重新启动 32.768 kHz 振荡器看门狗。 XTAL2L 标志将保持最近一次的值，250 us 后才显示更新值。

系统控制单元

符号	位	类型	描述
<b>XTAL2L</b>	4	rh	<b>32.768 kHz 片外振荡器频率过低标志</b> 振荡器看门狗监控 32.768 kHz 片外振荡器频率。 0 片外振荡器频率高于阈值。 1 片外振荡器频率低于阈值。
<b>XPD</b>	5	rw	<b>32.768 kHz 振荡器引出脚关闭控制</b> 0 32.768 kHz 振荡器引出脚未关闭。 1 32.768 kHz 振荡器引出脚关闭。GPIO 功能可用。
<b>INTOSC_ST</b>	6	rh	<b>片内振荡器稳定指示</b> 0 48MHz 和 75 kHz 振荡器不稳定。 1 48MHz 和 75 kHz 振荡器稳定。
<b>0</b>	7	r	<b>保留</b> 读取返回 0；应写入 0。

注： OSC\_CON 寄存器的复位值为 0011 0110<sub>B</sub>。复位后一个时钟周期，若两个振荡器均工作，位 48MOSC2L 和 75KOSC2L 将清零，随后该寄存器的复位值为 0011 0000<sub>B</sub>。

## 7.4 功率管理

XC83x 通过以下方式实现了多种省电模式，从而可灵活的降低系统功耗：

- 终止 CPU 时钟
- 终止系统某个单元的时钟
- 降低部分外设单元的时钟频率
- 具有快速重启能力的全系统掉电

复位后，缺省进入正常工作模式（见 图 7-3），系统以主频运行。在正常工作模式下，系统可运行在 24 MHz 或 8 MHz。可由软件选择进入不同的省电模式，包括：

- 空闲模式
- 掉电模式 1
- 掉电模式 2
- 掉电模式 3
- 掉电模式 4

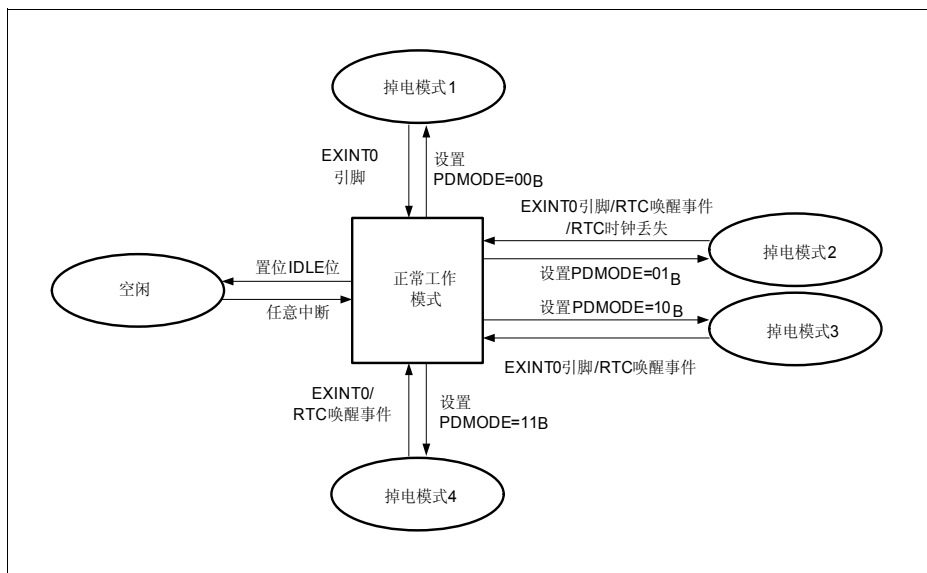


图 7-3 省电模式之间的转换（无复位）

在 XC83x 中，以 3.0 V - 5.5 V 正常  $V_{DDP}$  电压为系统供电时，所有功能在正常工作模式和空闲模式下均有效；供电电压降低时，用户需要的功能保持有效（只要有效电流低于极限值）。掉电模式下，当  $V_{DDP}$  低至 2.5 V 时，[章节 7.4.1.2](#) 中描述的特定模块仍须继续工作，但系统性能可能下降。

## 7.4.1 功能说明

### 7.4.1.1 空闲模式

空闲模式通过终止内核时钟来降低功耗。

空闲模式下，振荡器继续工作，但内核时钟关闭，内核停止工作。输入时钟未关闭的外设仍继续工作。系统进入空闲模式前用户应关闭 WDT；否则 WDT 在溢出时将产生内部复位，从而打断空闲模式。空闲模式下，CPU 的状态被完整保存：堆栈指针、程序计数器、程序状态字、累加器以及所有其它寄存器均保留进入空闲模式之前的数值；端口引脚保持空闲模式被激活时的逻辑状态。

软件通过设置 PCON.IDLE = 1 请求进入空闲模式。

激活任意被使能的中断可使系统退出空闲模式，返回正常工作模式。CPU 恢复运行，响应中断。执行完 RETI 指令，CPU 将返回执行 IDLE 置 1 指令之后的下一条指令。

### 7.4.1.2 掉电模式

为了达到不同的省电等级，XC83x 支持四种掉电模式：掉电模式 1、2、3 和 4。通常，在这两种模式下，48 MHz 振荡器和 Flash 存储器进入掉电状态，主电压调节器关闭，只有低功率电压调节器继续工作。因此，微控制器的大部分功能被终止，但 Flash、片内 RAM、XRAM 和 SFR 的内容被保留。

**表 7-3** 给出掉电模式下部分模块的行为。在掉电模式 2 下，RTC 以周期唤醒模式工作，从而器件可在特定的时间段之后退出掉电模式，该功能在低功耗应用中十分有用。在低压工作条件下（主电压可低至 2.5 V），那些在掉电模式下运行的模块仍保持工作，不过系统性能可能下降。其余模块在这两种掉电模式下均关闭。**表 7-4** 给出每种掉电模式的唤醒源。其中一种唤醒源是通过设置 EWS = 1、由 EXINT0 引脚接收外部唤醒信号。由寄存器 EXICON0 中的位 EXINT0 选择触发唤醒信号的跳变沿。

注：EXICON0.EXINT0 = 11<sub>B</sub> 不能用于掉电唤醒。

**表 7-3 掉电模式下的模块行为<sup>1)</sup>**

模块	掉电模式 1	掉电模式 2 <sup>2)</sup>	掉电模式 3 <sup>2)</sup>	掉电模式 4 <sup>2)</sup>
RTC	N	Y	Y	Y
75 kHz OSC	N	Y	N	Y
32.768 kHz OSC PAD	N	Y	Y	N
75 KHz OWD	N	N	N	N
32.768 kHz OWD	N	Y	N	N

1) N 代表掉电模式下模块关闭；Y 代表掉电模式下模块可工作。

2) 掉电模式下不支持 RTC 模式 2 和 3。一旦进入掉电模式，RTCCCLK 引脚将被关闭。



表 7-4 可用的掉电唤醒源

模块	掉电模式 1	掉电模式 2	掉电模式 3	掉电模式 4
RTC 唤醒	不可用	可用	可用	可用
Clock 出错	不可用	可用	不可用	不可用
EXINT0 <sup>1)</sup>	可用	可用	可用	可用

1) 通过位 PMCON0.EWS 选择 EXINT0 唤醒源。

## 掉电模式 1

该模式下，所有外设模块、CPU、由 32.768 kHz XTAL 引出脚或 75 kHz 振荡器提供输入时钟的 RTC 均被关闭，32.768 kHz 和 75 kHz 振荡器看门狗也被关闭。时钟关闭后，不能由中断或实时时钟唤醒系统，只能通过设置位 EWS = 1、由 EXINT0 引脚上的外部唤醒信号唤醒系统。系统进入掉电模式之前，必须选定唤醒源和唤醒类型。

进入掉电模式 1 的步骤如下：

- 设置 PMCON0.PDMODE = 00<sub>B</sub> 选择掉电模式 1。
- 设置 PMCON0.PD = 1 关闭包括 48 MHz 和 75 kHz 振荡器在内的所有模块。

位 PMCON0.PD 置 1 后必须插入两条 NOP 指令，从而可确保从掉电模式唤醒之后第一个指令（两条 NOP 指令之后）能够被正确执行。

## 掉电模式 2

该模式下，除表 7-3 列出的模块之外，其它所有模块均被关闭。RTC 以 32.768 kHz XTAL 引出脚提供的时钟工作在计时模式（RTC 模式 0），保持实时计数。此外，在该模式下，可使能 75 kHz 和 32.768 kHz 振荡器看门狗（OWD）以监控片外振荡器的状态。可通过 RTC 唤醒事件唤醒系统退出掉电模式 2，还可通过设置位 PMCON0.EWS = 1、由 EXINT0 引脚上的外部唤醒信号唤醒系统。若 32.768 kHz 的片外振荡器时钟丢失，OSC\_CON.XTAL2L 标志置 1 时退出掉电模式 2。

进入掉电模式 2 的步骤如下：

- 设置 OSC\_CON.XPD = 0 使能 32.768 kHz 振荡器 pad。
- 等待 2 s 至振荡器稳定。
- 使能 32.768 kHz OWD 之前，确保 75 kHz 振荡器达到稳定（OSC\_CON.INTOSC\_ST = 1）。
- 设置 OSC\_CON.XTALOWDRST=1 重新启动 32.768 kHz OWD。
- 当 XTAL2L 为 0 时，选择 RTC 模式 0（计时模式）。
- 设置 RTCON.RTCC = 1 确保使能 RTC。
- 设置 WDTCON.WDTEN = 0 禁用 WDT 模块。
- 设置 PMCON0.PDMODE = 01<sub>B</sub> 选择掉电模式 2。
- 设置 PMCON0.PD = 1 进入掉电模式。

位 PMCON0.PD 置 1 后必须插入两条 NOP 指令，从而可确保从掉电模式唤醒之后第一个指令（两条 NOP 指令之后）能够被正确执行。

*注：在进入掉电模式之前，必须通过软件设置禁用 WDT 定时器。若掉电模式下 WDT 发生溢出，器件将无法唤醒。*

### 掉电模式 3

掉电模式 3 和掉电模式 2 的区别在于：该模式下，32.768 kHz 振荡器看门狗和 75 kHz 振荡器被关闭，无法监控片外振荡器的状态。若片外振荡器稳定，可通过 RTC 唤醒事件唤醒系统退出掉电模式 3。唤醒时间必须长于进入掉电模式所需的时间。此外，还可通过设置位 PMCON0.EWS = 1、由 EXINT0 引脚上的外部唤醒信号唤醒系统。

进入掉电模式 3 的步骤如下：

- 设置 OSC\_CON.XPD = 0 使能 32.768 kHz 振荡器 pad。
- 等待 2 s 至振荡器稳定。
- 选择 RTC 模式 0（计时模式）。
- 设置 RTCON.RTCC = 1 确保使能 RTC。
- 设置 PMCON0.PDMODE = 10<sub>B</sub> 选择掉电模式 3。
- 设置 PMCON0.PD = 1 进入掉电模式。

*注：以上步骤不包括使能 32.768 kHz OWD，可在工作模式下监控片外振荡器时使能该 OWD。进入掉电模式 3 时，OWD 和 75 kHz 振荡器将由硬件自动关闭。*

位 PMCON0.PD 置 1 后必须插入两条 NOP 指令，从而可确保从掉电模式唤醒之后第一个指令（两条 NOP 指令之后）能够被正确执行。

### 掉电模式 4

该模式下，除表 7-3 列出的模块之外，其它所有模块均被关闭。RTC 以 75 kHz 的频率工作在周期唤醒模式下，保持实时计数。不过，由于 75 kHz 振荡器看门狗关闭，因此无法监控 75 kHz 振荡器的状态。可通过 RTC 唤醒事件唤醒系统退出掉电模式 4，还可通过设置位 PMCON0.EWS = 1、由 EXINT0 引脚上的外部唤醒信号唤醒系统。

进入掉电模式 4 的步骤如下：

- 设置 RTCON.RTCC = 1 确保使能 RTC 工作。
- 设置 WDTCON.WDTEN = 0 禁用 WDT 模块。
- 设置 PMCON0.PDMODE = 11<sub>B</sub> 选择掉电模式 4。
- 设置 PMCON0.PD = 1 进入掉电模式。

位 PMCON0.PD 置 1 后必须插入两条 NOP 指令，从而可确保从掉电模式唤醒之后第一个指令（两条 NOP 指令之后）能够被正确执行。

在各种掉电模式下，端口引脚保持掉电模式被激活时的逻辑状态。对于数字端口而言，一旦芯片进入掉电模式，所有未用作唤醒源的端口的输入 / 输出驱动器禁用，从而可降低漏电流。用户若想在输出引脚上保持“1”或“0”电平，在进入掉电模式之前应使能上拉 / 下拉器件。系统唤醒后，将重新使能端口引脚。

*注：在进入掉电模式之前，必须通过软件设置禁用 WDT 定时器。若掉电模式下 WDT 发生溢出，器件将无法唤醒。*

## 退出掉电模式

可通过以下方式退出掉电模式：

- EXINT0 引脚上检测到由 EXICON0.EXINT0 选定的信号跳变
- RTC 产生的唤醒事件请求
- RTC 时钟源出错

注：  $EXICON0.EXINT0 = 11_B$  不能用于掉电唤醒。

由位 PMCON0.EWS 使能 EXINT0 唤醒源。可由位 MODPSEL1.EXINT0IS 选择 EXINT0 的输入引脚。由位 PMCON0.WKSEL 选择带复位唤醒或无复位唤醒。

若进入掉电模式前 WKSEL 已置 1，系统将执行和上电复位类似的复位序列。因此，所有端口引脚处于复位状态，该状态一直保持到执行程序将其改变。

若进入掉电模式前 WKSEL 已清 0，系统将执行快速唤醒序列。端口引脚保持掉电模式下的有效状态，直至执行程序改变引脚值。

由 EXINT0 触发的无复位掉电唤醒的步骤如下：

1. 掉电模式下，EXINT0 引脚必须保持无效电平。
2. EXINT0 引脚变有效至少保持 100 ns 之后退出掉电模式。
3. 主电压调节器开启，大约需要 150  $\mu$ s 达到稳定。
4. 片上振荡器开启，通常需要 10  $\mu$ s 达到稳定。
5. 随后 FLASH 进入读取准备就绪模式。和正常复位下的时长 160  $\mu$ s 不同，该段时间可忽略。
6. CPU 恢复运行。内核将返回执行 PD 置 1 指令之后的下一条指令。

注： 唤醒源 EXINT0 不会引发中断，即使进入掉电模式之前 EXINT0 被使能。其它唤醒源也一样。只有在 CPU 恢复运行之后，EXINT0 满足中断产生条件时，才会产生中断。

在低压工作条件下，退出掉电模式进入正常工作模式时，有效电流必须低于数据手册规定的极限值。若该条件不满足，则在唤醒之后立即发生欠压复位。

无复位掉电唤醒后，可由唤醒事件的状态标志指示唤醒源。此外，可根据需要置位 OSC\_CON.RCOWDRST 重新使能振荡器看门狗。不过，在使能振荡器看门狗工作之前，必须检查标志位 OSC\_CON.INTOSC\_ST 的状态，确保 48 MHz 和 75 kHz 振荡器已稳定工作，详细说明请参见[章节 7.3.1](#)。

掉电模式 1 下，RTC 关闭。一旦退出掉电模式，RTC 要由用户重新使能。

### 7.4.1.3 外设时钟管理

功耗降低的程度取决于哪些外设在工作（运行外设的个数）。在实现某种特定功能时，对于那些不必要的外设，可关闭其时钟输入禁止该外设工作。例如，在空闲模式下，若所有定时器被停止，ADC、CCU6、MDU 和串行接口均不工作，则最大程度降低了系统功耗。不过，用户必须慎重决定在正常工作模式和空闲模式下，哪些外设应继续工作，那些外设应被停止。

置位寄存器 PMCON1 中的相应控制位，可分别禁止（关闭时钟）外设单元 ADC、SSC、CCU6、MDU、LEDTSU、IIC 和定时器 T2。此外，复位 GLOBCTR.ANON 可禁止 ADC 的模拟部分，该特性将终止 ADC 模拟时钟的产生，无需进行模数转换时功耗降低。

使用片上振荡器以节省功耗时，应置位 OSC\_CON.XPD 关闭 XTAL。使用片外振荡器时，可通过置位 OSC\_CON.OSCPD 关闭片上振荡器。

**PMCON1**  
**外设管理控制寄存器 1** (EF<sub>H</sub>) 复位值: FF<sub>H</sub>  
**RMAP: 0, PAGE: 1**

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	CDC_DIS	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
ADC_DIS	0	rw	<b>ADC 禁用请求，高有效</b> 0    ADC 正常工作 1    请求禁用 ADC（缺省状态）
SSC_DIS	1	rw	<b>SSC 禁用请求，高有效</b> 0    SSC 正常工作 1    请求禁用 SSC（缺省状态）
CCU_DIS	2	rw	<b>CCU 禁用请求，高有效</b> 0    CCU 正常工作 1    请求禁用 CCU（缺省状态）
T2_DIS	3	rw	<b>T2 禁用请求，高有效</b> 0    T2 正常工作 1    请求禁用 T2（缺省状态）
MDU_DIS	4	rw	<b>MDU 禁用请求，高有效</b> 0    MDU 正常工作 1    请求禁用 MDU（缺省状态）

---

**系统控制单元**

符号	位	类型	描述
<b>CDC_DIS</b>	5	rw	<b>CORDIC 禁用请求，高有效</b> 0     CORDIC 正常工作 1     请求禁用 CORDIC （缺省状态）
<b>LTS_DIS</b>	6	rw	<b>LEDTSCU 禁用请求，高有效</b> 0     LEDTSCU 正常工作 1     请求禁用 LEDTSCU （缺省状态）
<b>IIC_DIS</b>	7	rw	<b>IIC 禁用请求，高有效</b> 0     IIC 正常工作 1     请求禁用 IIC （缺省状态）

## 7.4.2 功率管理寄存器

### PMCON0

功率模式控制寄存器 0

(F3<sub>H</sub>)

复位值: 01<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
	<b>0</b>		<b>WKSEL</b>		<b>PDMODE</b>	<b>PD</b>	<b>EWS</b>
	r		rw		rw	rwh	rw

符号	位	类型	描述
<b>EWS</b>	0	rw	外部中断 0 唤醒源选择 0 不选择外部中断 0 唤醒 1 选择外部中断 0 唤醒
<b>PD</b>	1	rwh	掉电模式使能，高有效 置位该位将使芯片进入由 PDMODE 选择的掉电模式。由唤醒电路复位该位。 PD 是受保护位。保护机制有效时，该位不能被直接写入。有关保护机制的详细说明，参见 <a href="#">章节 3.4.4</a> 。
<b>PDMODE</b>	[3:2]	rw	掉电模式选择 00 选择掉电模式 1 01 选择掉电模式 2 10 选择掉电模式 3 11 选择掉电模式 4
<b>WKSEL</b>	4	rw	唤醒复位选择位 0 无复位唤醒 1 带复位唤醒
<b>0</b>	[7:5]	r	保留 读取返回 0；应写入 0。

**PCON** [ 注：该寄存器位于 XC800 内核中 ]

功率控制寄存器 [ 不可位寻址 ]

(87<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
<b>SMOD</b>		<b>0</b>		<b>GF1</b>	<b>GF0</b>	<b>0</b>	<b>IDLE</b>
rw		r		rw	rw	r	rw

符号	位	类型	描述
IDLE	0	rw	空闲模式使能
			0 不进入空闲模式 1 进入空闲模式

## 7.5 SCU 寄存器映射

系统控制 SFR 用于控制系统的整体功能（如中断、可变波特率产生、时钟管理、位保护方案及振荡器）。这些 SFR 位于标准 SFR 区（RMAP = 0），组织成 8 页。SCU\_PAGE 寄存器位于地址单元 F1<sub>H</sub>，其中包含分页值和页控制信息。

**SCU\_PAGE**  
**SCU 分页寄存器**  
**RMAP: 0, PAGE: X** **(F1<sub>H</sub>)** **复位值: 00<sub>H</sub>**

7	6	5	4	3	2	1	0
OP		STNR		0	PAGE		
w		w		r	rwh		

符号	位	类型	描述
<b>PAGE</b>	[3:0]	rwh	<b>页信息</b> 写入时，该值表示新页的值。读出时，该值表示当前有效页的值 = addr [y:x+1]。
<b>STNR</b>	[5:4]	w	<b>保存编号</b> 该编号指示在哪个保存位域上执行由 OP 确定的操作。 若 OP = 10 <sub>B</sub> ， PAGE 的内容在被新值覆盖之前保存在 SCU_STx 中。 若 OP = 11 <sub>B</sub> ， PAGE 的内容被 SCU_STx 覆盖。写入 PAGE 的值不予理睬。 00 选择 SCU_ST0 01 选择 SCU_ST1 10 选择 SCU_ST2 11 选择 SCU_ST3
<b>OP</b>	[7:6]	w	<b>操作</b> 0X 手动保存页模式，STNR 的值被忽略，PAGE 被直接写入。 10 带有自动页保存的新页设置。当前写入 PAGE 中的内容被保存的同时，上次写入 PAGE 中的内容被保存在 STNR 指定的位域 SCU_STx 中。 11 自动恢复页操作。对写入 PAGE 的内容不予理睬，PAGE 的内容由 STNR 指定的位域 SCU_STx 中的值覆盖。



## 系统控制单元

系统控制 SFR 的地址归纳见表 7-5，寄存器 SYSCON0 未在表中列出，可从标准（非映射）和映射地址单元 8F<sub>H</sub> 访问该寄存器。

**表 7-5 SCU 页 0 - 7 的 SFR 地址列表**

地址	页 0	页 1	页 2	页 3
F2 <sub>H</sub>	IRCON0	PASSWD		XADDRH
F3 <sub>H</sub>	IRCON1	PMCON0		MODPISEL
F4 <sub>H</sub>	EXICON1	OSC_CON		MODPISEL1
F5 <sub>H</sub>	IRCON2	ID		MODPISEL2
F6 <sub>H</sub>	IRCON3	WDTCON		MODSUSP
F7 <sub>H</sub>	NMISR	RSTCON		MODIEN
EE <sub>H</sub>	NMICON	SDCON		MODPISEL3
EF <sub>H</sub>	EXICON0	PMCON1		
地址	页 4	页 5	页 6	页 7
F2 <sub>H</sub>		BCON		
F3 <sub>H</sub>	WDTREL	BGL		
F4 <sub>H</sub>	WDTWINB	BGH		
F5 <sub>H</sub>	WDTL	LINST		
F6 <sub>H</sub>	WDTH	FEAL		
F7 <sub>H</sub>		FEAH		

## 8 看门狗定时器

### 8.1 概述

看门狗定时器 (WDT) 为检测软硬件故障以及故障恢复提供了高度可靠和安全的方式。要在用户预设的时间间隔内定期复位 WDT。CPU 必须在该时间间隔内服务 WDT 以避免引发 XC83x 系统复位。因此，WDT 服务程序可确保系统能够正常运行，确保系统可在用户设定的时间之后退出意外出错状况。

WDT 缺省被禁用。

调试模式下，看门狗定时器 WDT 缺省被挂起，停止计数（其调试挂起位缺省置位，即  $\text{MODSUSP.WDTSUSP} = 1$ ）。因此，在调试过程中不需要刷新 WDT。

#### 特性

- 16 位看门狗定时器
- 定时器高 8 位的重载值可编程设定
- 窗界可编程设定
- 输入频率由 75 kHz 片内振荡器（二级时钟源）产生

## 8.2 系统信息

本节给出 WDT 的相关系统信息。

### 8.2.1 复位的影响

必须周期刷新或清零 WDT 的定时器，否则定时器将溢出，触发看门狗复位。由寄存器 RSTCON 中的标志位 WDTRST 指示是否发生 WDT 复位。

在访问 RSTCON 寄存器之前必须先对 SCU\_PAGE 寄存器中的位域 PAGE 进行设置。

RSTCON

复位控制寄存器

RMAP: 0, PAGE: 1

(F7<sub>H</sub>)

复位值: 00<sub>H</sub><sup>1)</sup>

7	6	5	4	3	2	1	0
SWRQ	0				SOFTRS	WDTRST	WKRS
rwh	r				rwh	rwh	rwh

1) 发生 WDT 复位时，该寄存器的复位值为 02<sub>H</sub>。

符号	位	类型	描述
WDTRST	1	rwh	看门狗定时器复位指示位 0 未发生看门狗复位 1 已发生看门狗复位 该位只能由硬件置位、软件清零。
0	[6:3]	r	保留 读取返回 0；应写入 0。

### 8.2.2 时钟配置

WDT 以 75 kHz 振荡器产生的时钟工作。

### 8.2.3 中断事件及分配

表 8-1 给出 WDT 中断的非可屏蔽中断节点分配。

表 8-1 WDT 事件的非可屏蔽中断节点控制

事件	中断节点使能位	中断节点标志位	向量地址
WDT 溢出	NMICON.NMIWDT	NMISR.FNMIWDT	73 <sub>H</sub>

### 8.2.4 模块挂起控制

进入调试模式后，WDT 的定时器缺省被挂起。在该模式下，也可通过清零 MODSUSP. WDTSUSP 允许 WDT 工作。寄存器 MODSUSP 的定义可参见[章节 10.2.4](#)。

### 8.3 功能说明

看门狗定时器（WDT）是以 75 kHz 时钟递增计数的 16 位定时器，它由两个 8 位定时器串联组成。服务 WDT 时，可由用户预设定时器高 8 位的值，从而能够修改 WDT 的失效时间；每次服务 WDT 时复位定时器的低 8 位。WDT 单元的框图如图 8-1 所示。

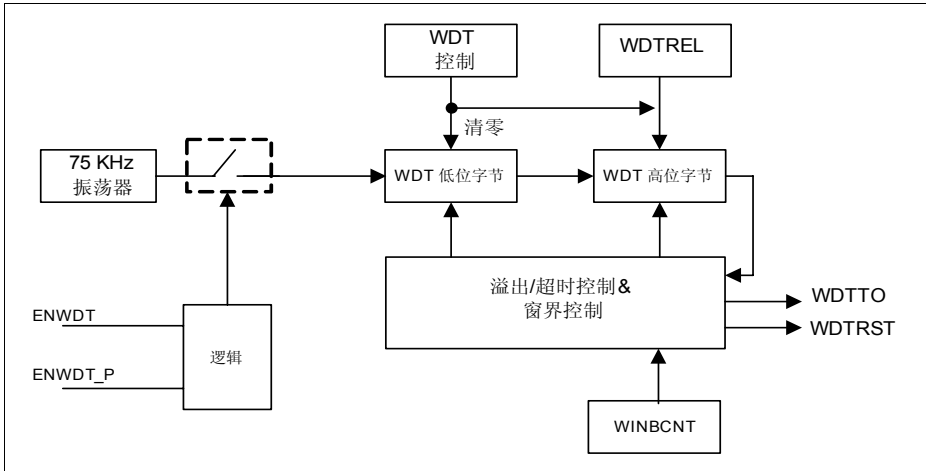


图 8-1 WDT 框图

若置位 WDTEEN 使能 WDT 工作，定时器将从用户设置的初始值开始递增计数。CPU 必须在计数器溢出之前服务 WDT，服务即执行刷新操作（WDTRS 置 1），把用户设置的初始值重新装入定时器，继续正常工作。

定时器溢出之前若 WDT 未被服务，则认为系统出错、正常运行模式被终止，将产生 WDT NMI 请求（FNMIWDT），随后进入预警模式。预警持续 30<sub>H</sub> 个计数周期。在预警阶段，WDT 刷新操作不予理睬，也不能关闭 WDT，将立即进行 XC83x 复位（WDRST）且该复位过程不能被终止。由寄存器 RSTCON 中的标志位 WDRST 指示是否发生 WDT 复位。若刷新和溢出同时发生，WDT 将不会进入预警状态。

必须周期刷新 WDT 以保证定时器不会溢出。刷新 WDT 意味着：对定时器的低位字节清零，将位域 WDTREL 中预先设置的值重新载入高位字节。刷新 WDT 还将清零 WDRST。

WDT 有一个“可编程窗界”，在窗界内不能刷新看门狗，窗界内的刷新操作视为无效，此时虽不会产生 NMI 请求，但 WDT 会激活 WDRST。窗界取值在 0000<sub>H</sub> 到 WDTWINB 和 00<sub>H</sub> 串联组成的值之间。可由 WINBEN 使能该特性。

WDT 被刷新后，定时器从 ( $\text{WDTREL} \times 2^8$ ) 开始继续递增计数。WDT 的溢出周期可由重载值 WDTREL 编程设定，WDTREL 是 WDT 的高位字节，在寄存器 WDTREL 中设置该重载值。

# 看门狗定时器

从 WDT 被刷新到下次溢出之间的溢出周期  $P_{WDT}$  可由下面的公式决定：

$$P_{WDT} = \frac{(2^{16} - WDTREL \times 2^8)}{f_{PCLK}} \tag{8.1}$$

当使能 WDT 的窗界刷新特性，若 WDTWINB 大于 WDTREL，则溢出周期  $P_{WDT}$  缩短，见图 8-2。用 WDTWINB 替换上面公式中的 WDTREL 即可计算相应的  $P_{WDT}$ 。为了使该窗界刷新特性真正有用，WDTWINB 不能小于 WDTREL。

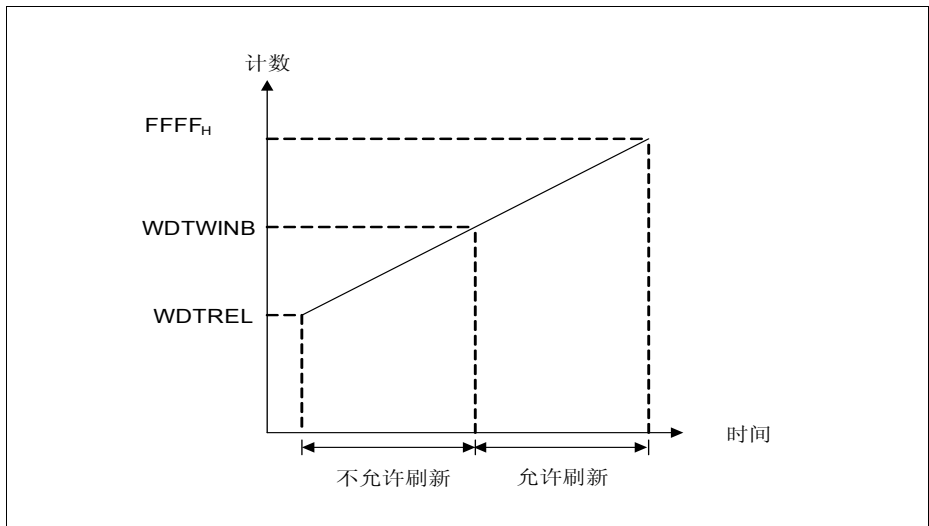


图 8-2 WDT 时序示意图

表 8-2 列出不同输入时钟所对应的 WDT 溢出周期取值。数值舍入到 3 位有效。

表 8-2 WDT 溢出周期范围

WDTREL 中的重载值	75 kHz 输入频率
FF <sub>H</sub>	3.4 ms
7F <sub>H</sub>	440 ms
00 <sub>H</sub>	874 ms

注：出于安全的考虑，建议用户在每次刷新 WDT 之前对 WDTCON 重写。

注：若置位 SFR MODSUSP 中的 WDTSUSP，当 OCDS 进入监控模式时，WDT 可被调试挂起。

### 8.4 寄存器说明

共由 5 个 SFR 控制 WDT 的操作，可从映射 SFR 区访问这些寄存器。  
 这些 SFR 的地址归纳见表 8-3。

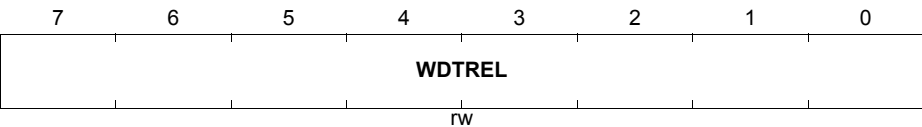
表 8-3 寄存器映射

地址	页	寄存器
F6 <sub>H</sub>	1	WDTCON
F3 <sub>H</sub>	4	WDTREL
F4 <sub>H</sub>	4	WDTWINB
F5 <sub>H</sub>	4	WDTL
F6 <sub>H</sub>	4	WDTH

#### 8.4.1 看门狗定时器寄存器

WDT 的当前计数值存放在不可位寻址的只读寄存器 WDTH 和 WDTL 中。WDT 控制寄存器 WDTCON（可位寻址）用于控制 WDT 的操作，还用来选择输入时钟的预分频因子。寄存器 WDTREL 定义定时器高位字节的重载值。WDTWINB 定义看门狗窗界计数值。

**WDTREL**  
**WDT 重载寄存器** (F3<sub>H</sub>) **复位值: 00<sub>H</sub>**  
**RMAP: 0, PAGE: 4**



符号	位	类型	描述
WDTREL	[7:0]	rw	WDT 的重载值 (载入 WDT 的高位字节)

## 看门狗定时器

## WDTCON

WDT 控制寄存器

(F6<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0		WINBEN	WDTPR	0	WDTEN	WDTRS	0
r		rw	rh	r	rw	rwh	r

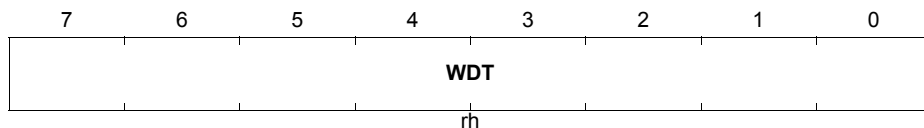
符号	位	类型	描述
WDTRS	1	rwh	<b>WDT 刷新开始控制</b> 高有效。该位置位将开始刷新 WDT。该位由软件置位后由硬件自动清零。
WDTEN	2	rw	<b>WDT 使能</b> 0 禁用 WDT 1 使能 WDT WDTEN 是受保护位。保护机制有效时，该位不能被直接写入。 <i>注： 预警模式下 (WDTPR = 1)，清零 WDTEN 无效。</i>
WDTPR	4	rh	<b>WDT 预警模式标志</b> 0 正常工作模式（复位后的缺省模式） 1 WDT 工作在预警模式 检测到 WDT 出错时该位置 1，WDT 发出 NMI 强制中断并进入预警模式。预警周期结束后芯片复位。
WINBEN	5	rw	<b>WDT 窗界使能控制</b> 0 禁用 WDT 窗界特性（缺省状态） 1 使能 WDT 窗界特性
0	0, 3, [7:6]	r	<b>保留</b> 读操作返回 0，应写入 0。



**看门狗定时器**
**WDTL**

看门狗定时器，低位字节

**(F5<sub>H</sub>)**

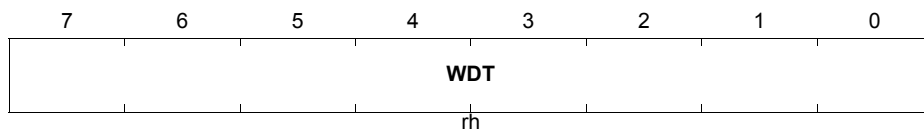
 复位值: **00<sub>H</sub>**
**RMAP: 0, PAGE: 4**


符号	位	类型	描述
WDT	[7:0]	rh	WDT 的当前值

**WDTH**

看门狗定时器，高位字节

**(F6<sub>H</sub>)**

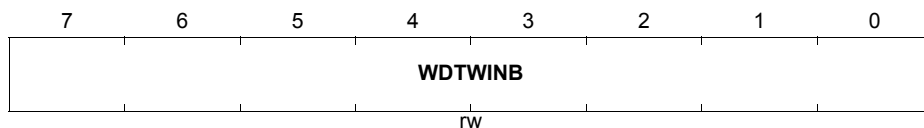
 复位值: **00<sub>H</sub>**
**RMAP: 0, PAGE: 4**


符号	位	类型	描述
WDT	[7:0]	rh	WDT 的当前值

**WDTWINB**

WDT 窗界计数寄存器

**(F4<sub>H</sub>)**

 复位值: **00<sub>H</sub>**
**RMAP: 0, PAGE: 4**


**看门狗定时器**

符号	位	类型	描述
<b>WDTWINB</b>	[7:0]	rw	<b>WDT 窗界计数值</b> 该值可编程设定。窗界取值在 0000 <sub>H</sub> 到 (WDTWINB, 00 <sub>H</sub> ) 之间，在窗界内 WDT 不能被刷新，否则将产生 WDT 复位 (WDTRST 置位)。WDTWINB 与 WDTW 匹配。

## 9 中断系统

XC800 内核支持 1 个非可屏蔽中断（NMI）和 14 个可屏蔽中断。除了内核支持的标准中断功能（例如，可配置的中断优先级和中断屏蔽功能）之外，XC83x 中断系统还提供了扩展中断功能，例如：将每个中断向量映射给多个中断源以增加所支持的中断源个数，附加的状态寄存器可用来检测和确定中断源。

### 9.1 中断源

XC83x 支持 14 个中断向量，共分为 4 级中断优先级。其中 10 个中断向量分配给片上外设：定时器 T0、定时器 T1、UART 和 SSC 各分配 1 个专用中断向量；定时器 T2、ADC、LIN、LEDTSCU、P1 口 HCPADA 和捕获 / 比较单元共享 6 个中断向量。此外，将 4 个中断向量分配给外部中断、MDU、CORDIC、RTC 和 IIC：外部中断 0 和 1 各分配一个专用中断向量，外部中断 2 和 MDU、CORDIC、IIC 共享 1 个中断向量。RTC 和外部中断 [6:3] 共享 1 个中断向量。

非可屏蔽中断（NMI）具有最高中断优先级。在 XC83x 系统中，以下事件可产生 NMI：

- 看门狗定时器，溢出前预警
- 48 MHz 和 75 kHz 振荡器，振荡器时钟丢失
- 32.768 kHz XTAL 振荡器，XTAL 时钟丢失
- Flash 定时器，操作（如擦除）结束
- OCDS，发生用户 IIRAM 事件
- Flash ECC 出错
- $V_{DDP}$  预警
- $V_{DDC}$  预警

图 9-1, 图 9-2, 图 9-3, 图 9-4 和图 9-5 给出常规中断源和中断节点总览（包括相应的控制和状态标志）。图 9-6 给出 NMI 请求源总览。

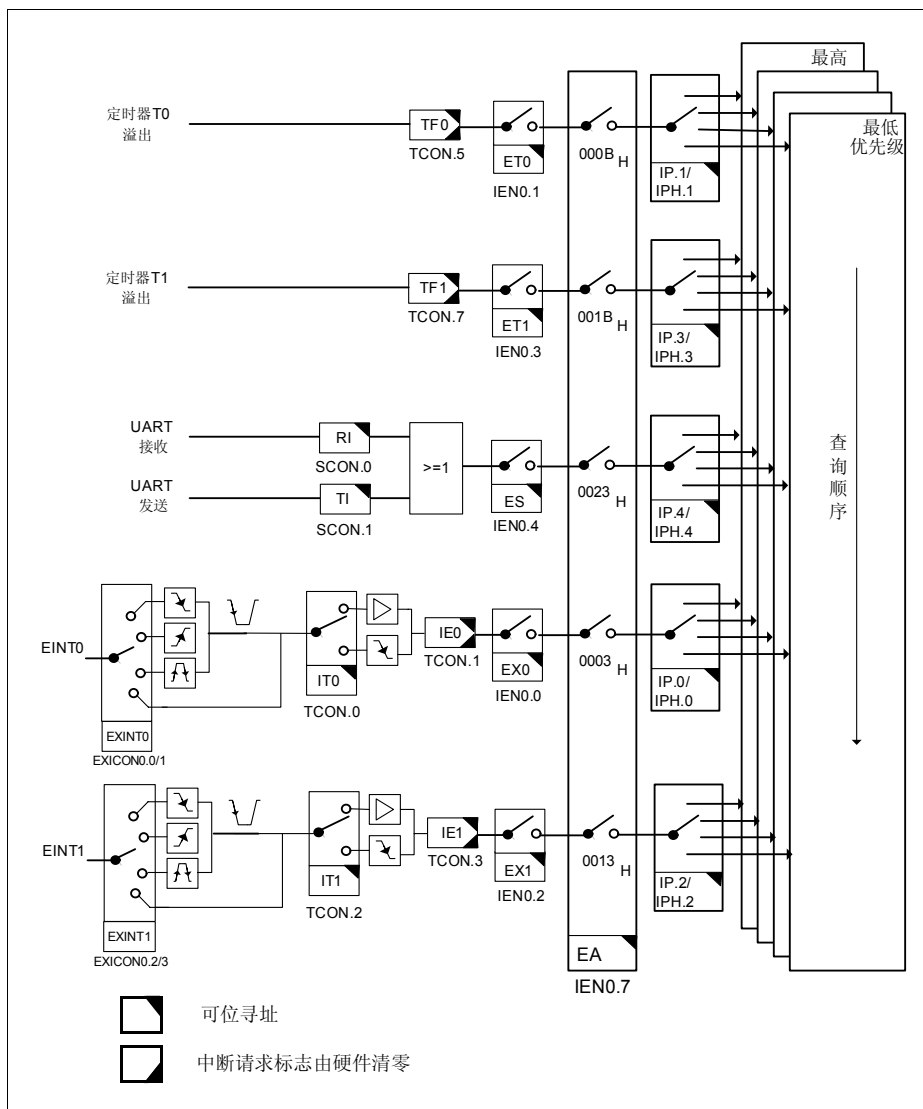


图 9-1 中断请求源（第 1 部分）

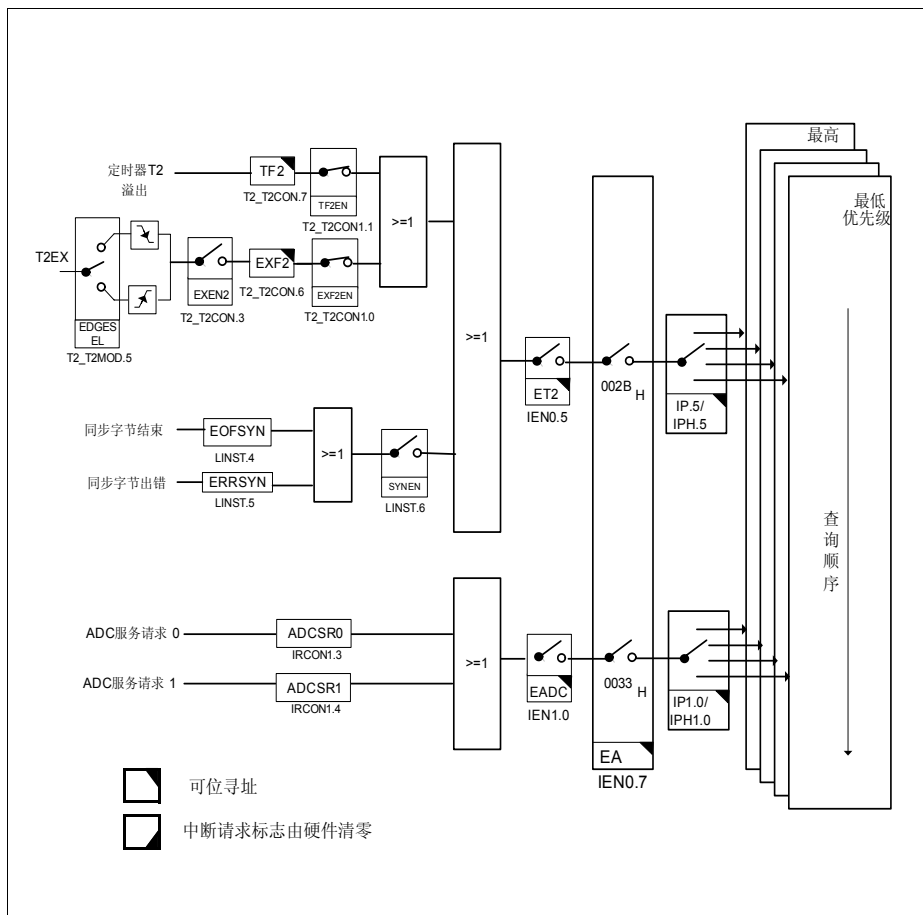


图 9-2 中断请求源（第 2 部分）

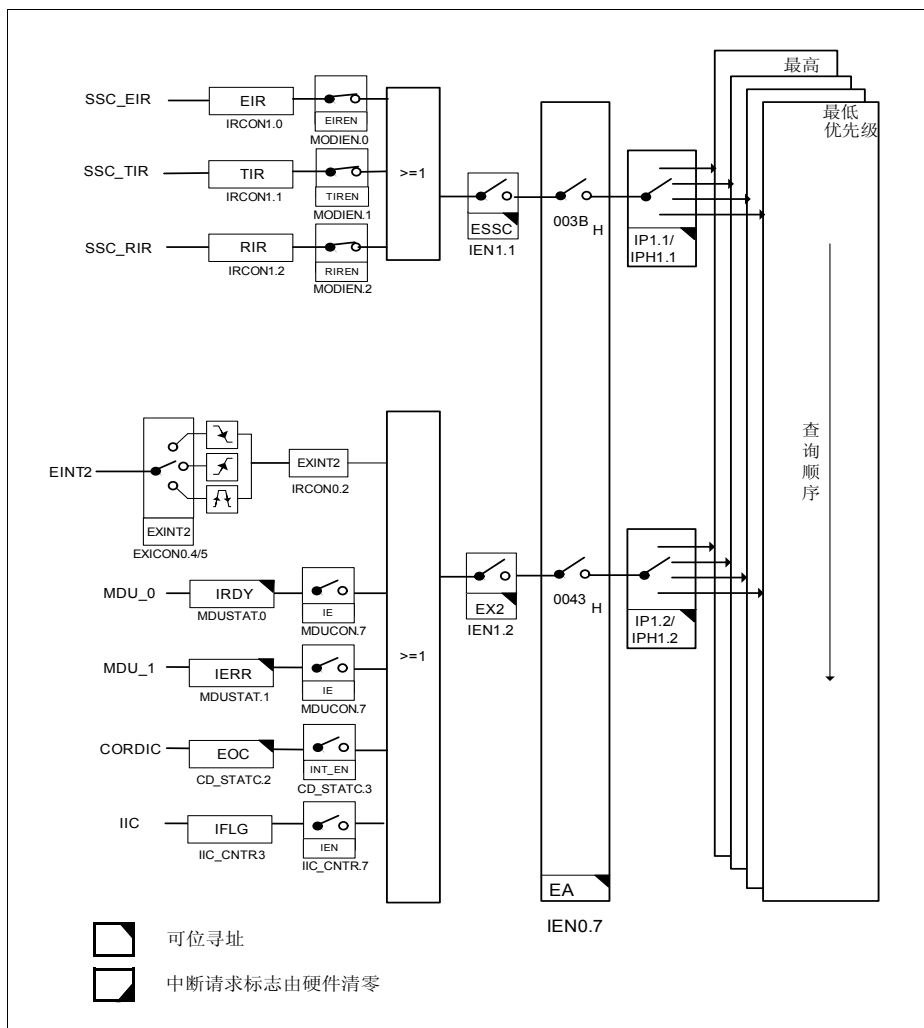


图 9-3 中断请求源（第 3 部分）



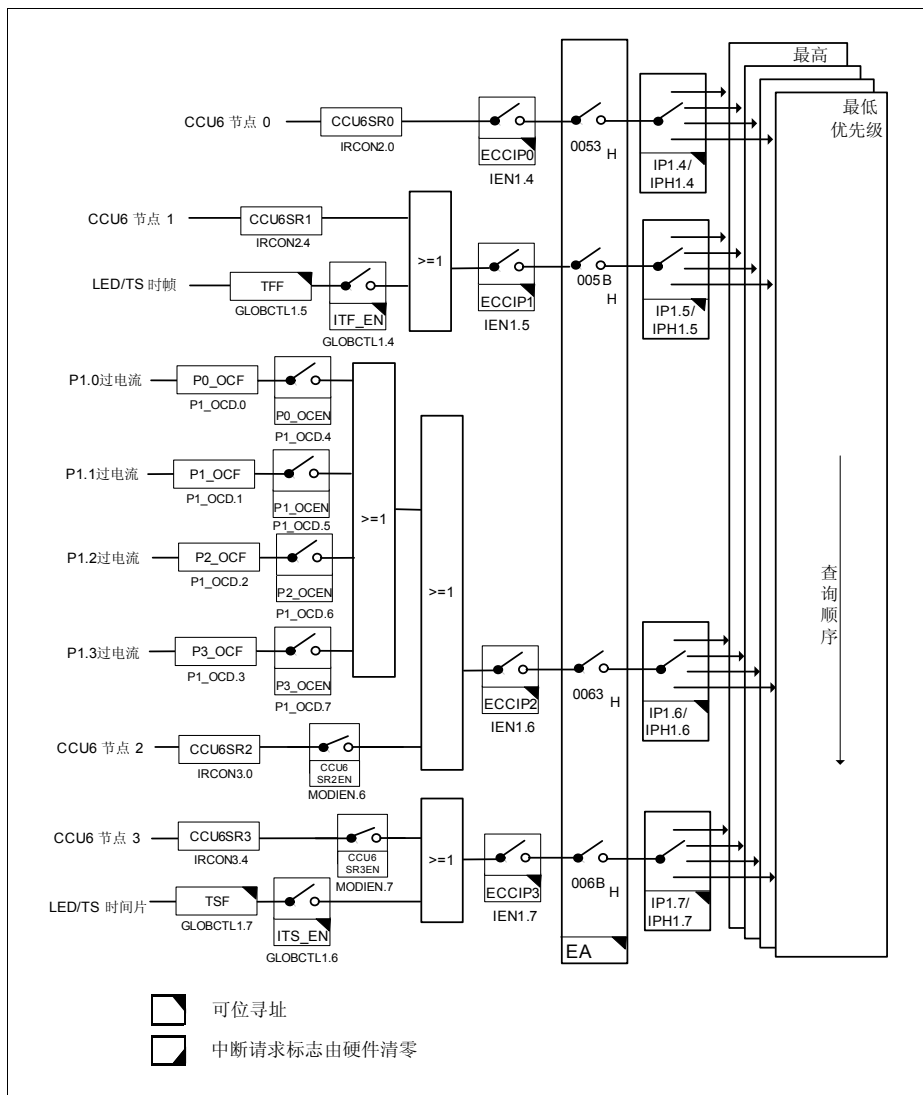


图 9-5 中断请求源（第 5 部分）



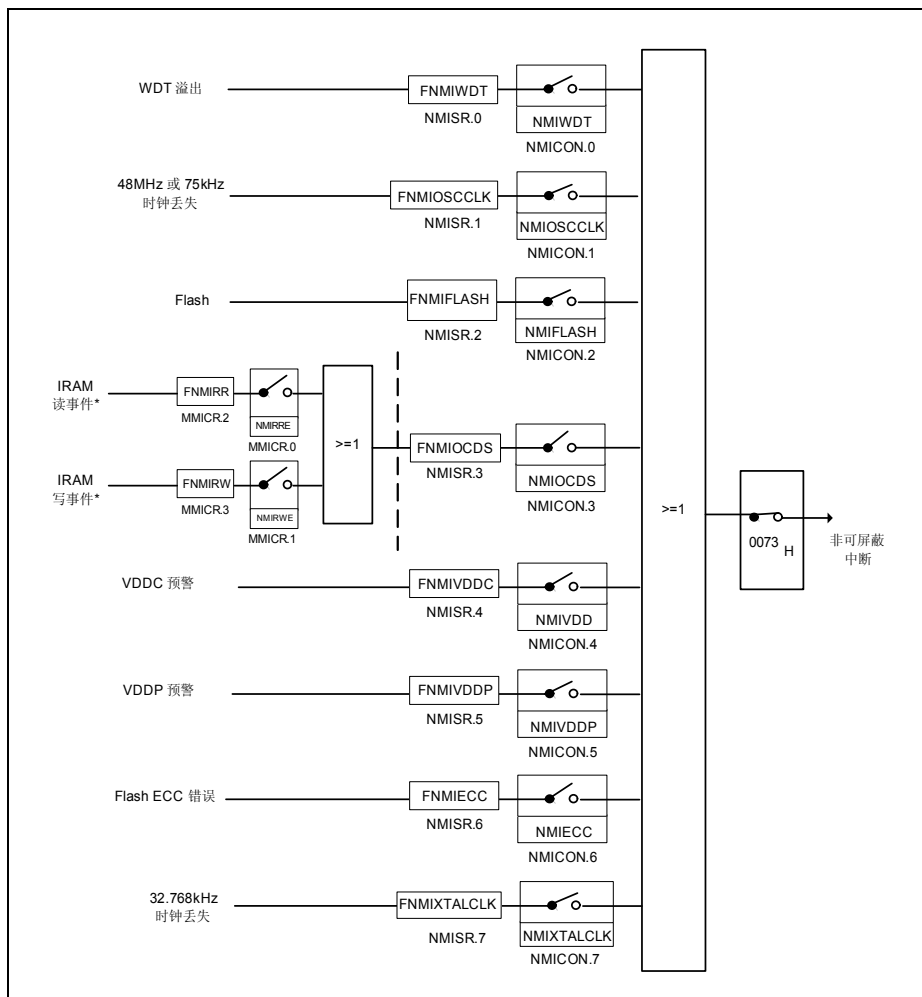


图 9-6 非可屏蔽中断请求源

### 9.1.1 中断源和中断向量

每个中断事件源对应有一个所属中断节点的中断向量地址。通过访问该中断向量来服务相应的中断节点请求。通过使能位可单独使能或禁止每个中断节点的中断服务。XC83x 中分配给各中断源的中断向量地址和对应的中断节点使能位归纳见[表 9-1](#)。

**表 9-1 中断向量地址**

中断节点	中断向量地址	XC83x 中断源分配	使能位	SFR
NMI	0073 <sub>H</sub>	看门狗定时器 NMI	NMIWDT	NMICON
		48 MHz 和 75 kHz 时钟 NMI	NMIOSCCLK	
		32.768 kHz XTAL 时钟 NMI	NMIXTALCLK	
		Flash 操作完成 NMI	NMIFLASH	
		OCDS NMI	NMIOCDS	
		ECC 出错 NMI	NMIECC	
		VDDP 预警 NMI	NMIVDDP	
		VDDC 预警 NMI	NMIVDDC	
XINTR0	0003 <sub>H</sub>	外部中断 0	EX0	IEN0
XINTR1	000B <sub>H</sub>	定时器 T0	ET0	
XINTR2	0013 <sub>H</sub>	外部中断 1	EX1	
XINTR3	001B <sub>H</sub>	定时器 T1	ET1	
XINTR4	0023 <sub>H</sub>	UART	ES	
XINTR5	002B <sub>H</sub>	定时器 T2	ET2	
		LIN		

表 9-1 中断向量地址

中断节点	中断向量地址	XC83x 中断源分配	使能位	SFR
XINTR6	0033 <sub>H</sub>	ADC	EADC	IEN1
XINTR7	003B <sub>H</sub>	SSC	ESSC	
XINTR8	0043 <sub>H</sub>	外部中断 2	EX2	
		CORDIC		
		MDU		
		IIC		
XINTR9	004B <sub>H</sub>	外部中断 3	EXM	
		外部中断 4		
		外部中断 5		
		外部中断 6		
		RTC 中断		
XINTR10	0053 <sub>H</sub>	CCU6 SR0	ECCIP0	
XINTR11	005B <sub>H</sub>	CCU6 SR1	ECCIP1	
		LEDTSU 时帧		
XINTR12	0063 <sub>H</sub>	CCU6 SR2	ECCIP2	
		P1 口 HCPADA		
XINTR13	006B <sub>H</sub>	CCU6 SR3	ECCIP3	
		LEDTSU 时间片		

### 9.1.2 中断源和中断优先级

当前被服务的中断只能响应具有更高优先级的中断，而不能响应同级或低优先级中断。因此，具有最高优先级的中断不响应其它任何中断。

若 CPU 同时接收到两个或更多不同优先级的中断请求，它会首先响应最高优先级的请求。若 CPU 同时接收到多个相同优先级的中断请求，则由内部查询顺序决定首先响应哪个中断请求。因此，每级优先级内又含有由查询顺序决定的次级优先级结构，见表 9-2。

表 9-2 同级内的优先级结构

中断源	优先级
非可屏蔽中断（NMI）	（最高）
外部中断 0	1
定时器 T0	2

表 9-2 同级内的优先级结构

中断源	优先级
外部中断 1	3
定时器 T1	4
UART	5
定时器 T2 / LIN	6
ADC 和 ORC	7
SSC	8
外部中断 2 / CORDIC/ MDU / IIC	9
EXINT[6:3] / 实时时钟	10
CCU6 中断节点指针 0	11
CCU6 中断节点指针 1 / LED 和触摸感应	12
CCU6 中断节点指针 2/P1 口 HCPADA	13
CCU6 中断节点指针 3 / LED 和触摸感应	14

## 9.2 中断结构

中断事件可由片上外设或由外部产生。中断事件检测由相关的片上外设来控制。可通过中断状态标志确定发生了哪种中断事件，该特性对于共用一个中断节点的多个中断源尤其有用。每个中断节点（NMI 除外）有一个全局使能 / 禁止位。在大多数情况下，还为特定的中断事件（NMI 事件）提供附加的使能 / 禁止位。某事件的中断使能位若被禁用，即使发生该中断事件，也不会产生中断请求。

总的来说，XC83x 有两种中断结构，主要区别在于（由事件触发）产生中断请求（每个直接送至内核的中断请求对应一个中断向量 / 节点）的方式、以及中断请求被清零的方式不同。

这两种中断结构的共同点是都具有中断屏蔽位 EA，用于全局使能或者禁止所有送至内核的中断请求（NMI 除外）。复位 EA（清零）只会屏蔽送至内核的挂起中断请求，而不会阻止捕获送入的中断请求。

注：NMI 节点和其它中断节点类似，区别在于前者没有 EA 位。NMI 节点是不可屏蔽的。

### 9.2.1 中断结构 1

中断结构 1 如图 9-7 所示，中断事件将会置位中断状态标志，同时产生一个挂起的中断请求。只有当相应的中断节点被使能时，挂起的中断请求才有效，中断内核。一旦中断节点被服务（中断被应答），其挂起的中断请求（由中断状态标志表示）可由硬件（内核）自动清零。

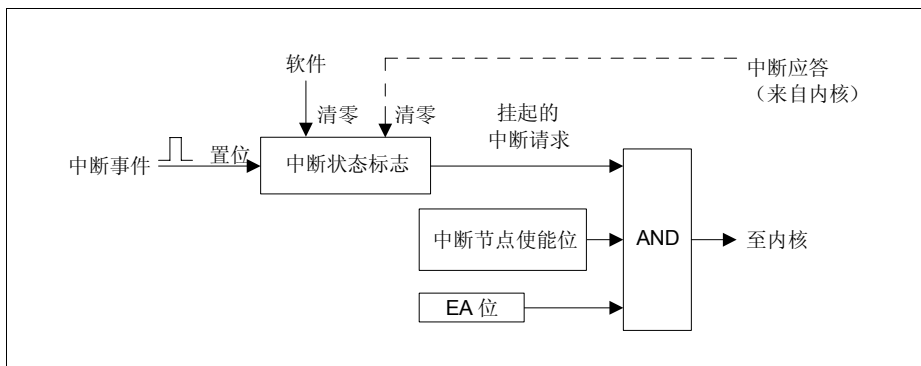


图 9-7 中断结构 1

XC83x 系统中，一旦中断源定时器 T0、定时器 T1、外部中断 0 和外部中断 1（各自对应一个专用中断节点）产生的挂起中断请求被服务，寄存器 TCON 中相应的中断状态标志 TF0、TF1、IE0 和 IE1 由内核清零。若某个中断节点被禁用（例如，使用软件查询），由于内核未被中断（从而也不会产生中断应答），其相应的中断状态标志必须由软件清除。对于具有专用中断节点的 UART 模块而言，即使其挂起的中断请求被服务，寄存器 SCON 中的中断状态标志 RI 和 TI 也不能由内核清零。UART 模块的中断状态标志（挂起的中断请求）只能由软件清零。

## 9.2.2 中断结构 2

图 9-8 所示的中断结构 2 适用于定时器 T2、LIN、外部中断 2 到 6、ADC、SSC、IIC、RTC、LEDTSU、P1 口 HCPADA、CCU6 和 MDU 中断源。该结构中，中断状态标志不会直接驱动挂起的中断请求。

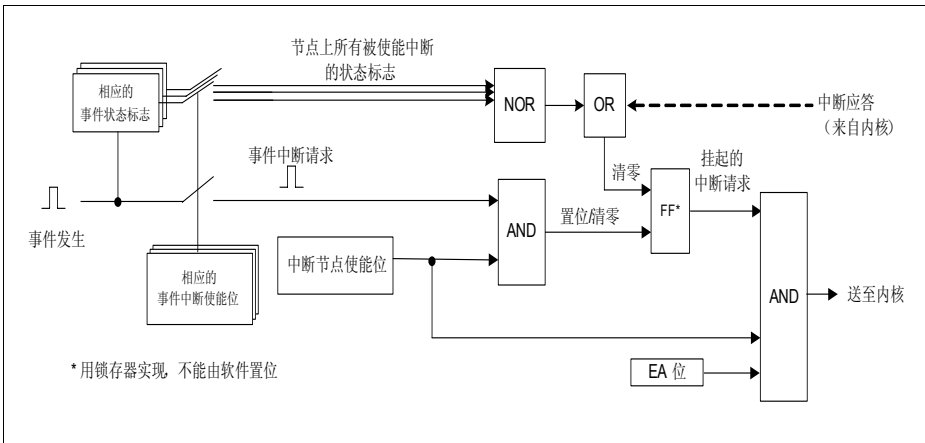


图 9-8 中断结构 2

中断源产生的中断事件将置位相应的中断状态标志，同时，若该事件被使能，将激活挂起的中断请求送至内核。考虑产生中断事件但其中断节点被禁用的情况（假设全局中断使能位 **EA** 置位）。一段时间过后使能中断节点时，先前激活的挂起中断请求此时将产生中断请求送至内核。**NMI** 节点比较特殊，因为它是非可屏蔽中断。

有效的挂起中断请求会中断内核，一旦中断节点被服务（中断被应答），该请求被硬件（内核）自动清零；中断状态标志保持置位，必须由软件清零。挂起的中断请求也可由软件清零；只有将节点上被使能中断的状态标志全部清零，才能间接清零挂起的中断请求。需要注意：这和中断结构 1 的方式有所不同，中断结构 1 只需复位节点的中断状态标志，即可直接清零挂起的中断请求。

总而言之，在中断结构 2 中，中断处理的优先级由高到低排列如下：1) **CPU** 应答中断请求，执行中断服务程序将清除挂起的中断请求；2) 任一被使能中断的事件发生时，将置位挂起的中断请求；3) 清除所有被使能中断的状态标志将清除挂起的中断请求。

## 9.3 中断处理

在每个机器周期的 **P2** 对中断请求信号进行采样，在下一个机器周期对采样到的中断请求进行查询。如果某中断节点请求标志在上一周期的 **P2** 已经有效，查询周期将发现该请求，中断系统将产生一个 **LCALL** 指令调用相应的中断服务程序。由硬件生成的 **LCALL** 指令在下列任意一种条件下都会推迟执行：

1. 正在处理同级或更高优先级的中断。
2. 当前周期（查询周期）不是正在执行指令的最后一个周期。
3. 正在执行的指令是 **RETI** 或访问寄存器 **IEN0/IEN1**、**IP/IPH** 或 **IP1/IPH1** 的写指令。

上述任意一种情况都会推迟执行 **LCALL** 进入中断服务程序。条件 2 保证了正在执行的指令在进入中断服务程序之前可执行完成；条件 3 保证了如果正在执行的指令是 **RETI** 或是

对寄存器 IEN0/IEN1、IP/IPH 或 IP1/IPH1 的写操作时，进入中断服务程序之前必须至少再执行一条指令，该延迟保证了中断状态的改变可被 CPU 监测到。

中断查询在每个机器周期重复执行，查询到的值为前一机器周期 P2 的采样值。需要注意，若某中断标志有效（置位），但由于上述的条件之一而导致它的节点中断请求未被响应；或当服务该中断节点时，该中断标志已不再有效，相应的中断源不再被响应。换言之，如果中断请求标志曾经有效、但未能被 CPU 及时响应，该标志将不被记忆。每个查询周期仅查询挂起的中断请求。

处理器执行硬件生成的 LCALL 指令，调用相应的中断服务程序响应中断请求。有些情况下，由硬件清除中断标志；有些情况下，必须由用户程序清除中断标志。由硬件生成的 LCALL 会将程序计数器（PC）的值压入堆栈（但不保存程序状态字 PSW 的内容），并将被响应中断节点的中断向量地址重新装入 PC，中断向量地址总结见表 9-1。

中断服务程序执行到 RETI 指令时，程序返回继续执行调用中断后的下一条指令。RETI 指令通知处理器中断服务程序已执行完毕，然后从栈顶弹出两个字节重新装入 PC，继续执行被中断的程序。需要注意的是，RETI 指令非常重要，它通知处理器用户程序已离开当前的中断优先级。简单的 RET 指令也可以返回到被中断的程序，但这样会使中断控制系统假定中断服务程序仍在执行。在这种情况下，同级或低优先级中断请求就无法被响应。

## 9.4 中断响应时间

对于某中断节点上的（不同中断源产生的）中断事件，其相应的中断请求信号将在每个机器周期的 P2 被采样，在下一个机器周期之前该值不会被电路查询。如果中断请求有效并且响应中断的条件成立，下一条将执行硬件生成的调用指令，进入中断服务程序。调用指令本身占用两个机器周期，因此，从中断请求有效到开始执行中断服务程序中的第一条指令至少需要三个完整的机器周期，如图 9-9 所示。

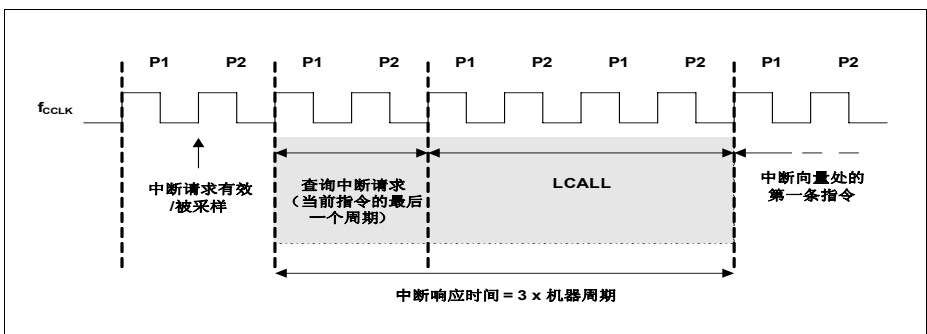


图 9-9 最短中断响应时间

如果下列任意一种情况出现，中断请求则需要更长的响应时间：

1. 如果一个同级或更高优先级的中断正在处理中，则附加的等待时间取决于其它中断服务程序所需时间。

## 中断系统

- 如果正在执行的指令还没有执行到它的最后一个周期，则附加的等待时间不会超过三个机器周期，因为最长的指令（**MUL** 和 **DIV** 指令）仅四个机器周期，见 [图 9-10](#)。
- 如果正在执行的指令是 **RETI** 指令或访问寄存器 **IEN0**、**IEN1**、或 **IP(H)**、**IP1(H)** 的写指令，则附加的等待时间不会超过五个机器周期（最多再用一个机器周期完成当前指令，如果指令是 **MUL** 或 **DIV**，再加四个机器周期完成下条指令），见 [图 9-11](#)。

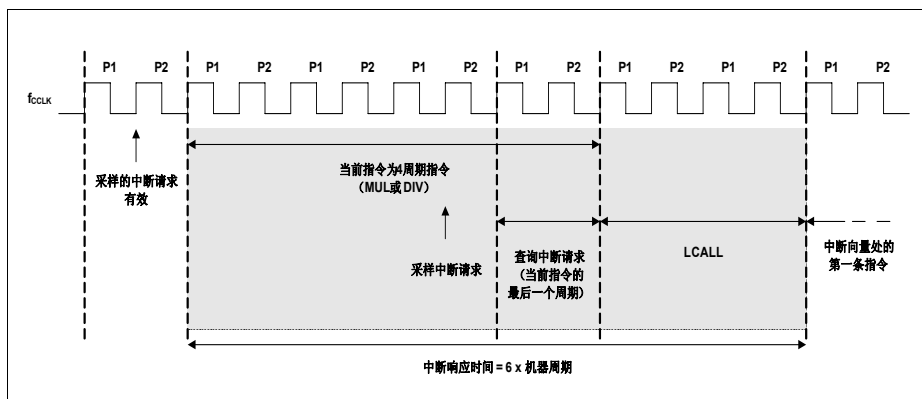


图 9-10 条件 2 下的中断响应时间

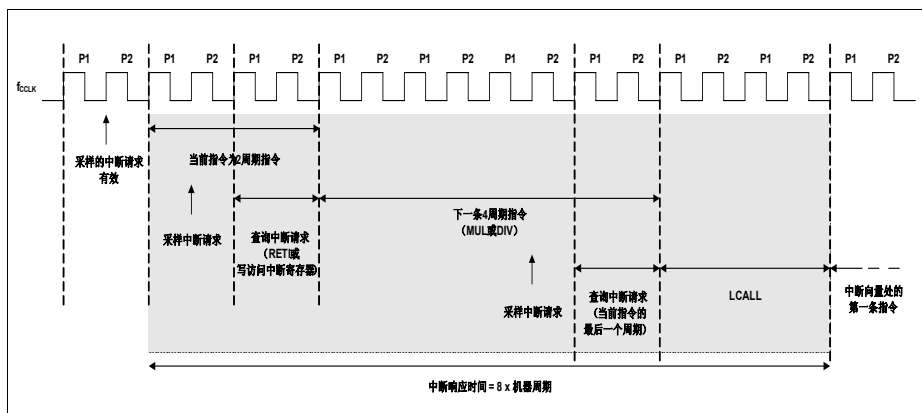


图 9-11 条件 3 下的中断响应时间

因此，在单中断系统中，中断响应时间始终大于等于三个机器周期、小于九个机器周期（不考虑等待状态）。当考虑等待状态时，中断响应时间将会延长，这取决于中断响应期间（[图 9-10](#) 和 [图 9-11](#) 的阴影部分）执行的用户指令占用的时间（硬件生成的 **LCALL** 除外）。



### 9.5 寄存器说明

中断 SFR 具有以下功能：中断节点使能、外部中断控制、中断标志和中断优先级设置。  
中断 SFR 及相应的地址归纳见表 9-3。

表 9-3 寄存器映射

地址	寄存器
RMAP = 0 或 1	
A8 <sub>H</sub>	IEN0
E8 <sub>H</sub>	IEN1
B8 <sub>H</sub>	IP
B9 <sub>H</sub>	IPH
F8 <sub>H</sub>	IP1
F9 <sub>H</sub>	IPH1
88 <sub>H</sub>	TCON
98 <sub>H</sub>	SCON
RMAP = 0	
EE <sub>H</sub> (SCU 页 0)	NMICON
EF <sub>H</sub> (SCU 页 0)	EXICON0
F4 <sub>H</sub> (SCU 页 0)	EXICON1
F4 <sub>H</sub> (SCU 页 3)	MODPISEL1
F2 <sub>H</sub> (SCU 页 0)	IRCON0
F3 <sub>H</sub> (SCU 页 0)	IRCON1
F5 <sub>H</sub> (SCU 页 0)	IRCON2
F6 <sub>H</sub> (SCU 页 0)	IRCON3
F7 <sub>H</sub> (SCU 页 0)	NMISR

### 9.5.1 中断节点使能寄存器

通过置位或清零中断使能寄存器 IEN0 和 IEN1 中的各控制位，每个中断节点可分别被使能或禁止。寄存器 IEN0 中还包含全局中断屏蔽位（EA），该位被清零可立即禁止所有挂起的中断请求。

多个 NMI 中断源共用 NMI 中断向量，每个中断源可通过寄存器 NMICON 分别被使能或禁止。

复位后，寄存器 IEN0、IEN1 和 NMICON 中的使能位全被清零，即所有中断节点缺省被禁止。

**IEN0**  
**中断使能寄存器 0** (A8<sub>H</sub>) 复位值 : 00<sub>H</sub>  
**RMAP: X, PAGE: X**

7	6	5	4	3	2	1	0
<b>EA</b>	<b>0</b>	<b>ET2</b>	<b>ES</b>	<b>ET1</b>	<b>EX1</b>	<b>ET0</b>	<b>EX0</b>
rw	r	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>EX0</b>	0	rw	中断节点 <b>XINTR0</b> 使能 0 禁止 XINTR0 1 使能 XINTR0
<b>ET0</b>	1	rw	中断节点 <b>XINTR1</b> 使能 0 禁止 XINTR1 1 使能 XINTR1
<b>EX1</b>	2	rw	中断节点 <b>XINTR2</b> 使能 0 禁止 XINTR2 1 使能 XINTR2
<b>ET1</b>	3	rw	中断节点 <b>XINTR3</b> 使能 0 禁止 XINTR3 1 使能 XINTR3
<b>ES</b>	4	rw	中断节点 <b>XINTR4</b> 使能 0 禁止 XINTR4 1 使能 XINTR4
<b>ET2</b>	5	rw	中断节点 <b>XINTR5</b> 使能 0 禁止 XINTR5 1 使能 XINTR5

## 中断系统

符号	位	类型	描述
<b>EA</b>	7	rw	<b>全局中断屏蔽</b> 0 所有挂起的中断请求（NMI 除外）都被禁止 1 挂起的中断请求未被禁止
<b>0</b>	6	r	<b>保留</b> 读操作返回 0，应写入 0。

**IEN1**

中断使能寄存器 1

(E8<sub>H</sub>)

复位值 : 00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
<b>ECCIP3</b>	<b>ECCIP2</b>	<b>ECCIP1</b>	<b>ECCIP0</b>	<b>EXM</b>	<b>EX2</b>	<b>ESSC</b>	<b>EADC</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>EADC</b>	0	rw	<b>中断节点 XINTR6 使能</b> 0 禁止 XINTR6 1 使能 XINTR6
<b>ESSC</b>	1	rw	<b>中断节点 XINTR7 使能</b> 0 禁止 XINTR7 1 使能 XINTR7
<b>EX2</b>	2	rw	<b>中断节点 XINTR8 使能</b> 0 禁止 XINTR8 1 使能 XINTR8
<b>EXM</b>	3	rw	<b>中断节点 XINTR9 使能</b> 0 禁止 XINTR9 1 使能 XINTR9
<b>ECCIP0</b>	4	rw	<b>中断节点 INTR10 使能</b> 0 禁止 XINTR10 1 使能 XINTR10
<b>ECCIP1</b>	5	rw	<b>中断节点 XINTR11 使能</b> 0 禁止 XINTR11 1 使能 XINTR11
<b>ECCIP2</b>	6	rw	<b>中断节点 XINTR12 使能</b> 0 禁止 XINTR12 1 使能 XINTR12

符号	位	类型	描述
<b>ECCIP3</b>	7	rw	<b>中断节点 XINTR13 使能</b> 0 禁止 XINTR13 1 使能 XINTR13

在访问 NMICON 寄存器之前必须先对 SCU\_PAGE 寄存器中的位域 PAGE 进行设置。

**NMICON**
**NMI 控制寄存器**
**(EE<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: 0**

7	6	5	4	3	2	1	0
<b>NMI-XTALCLK</b>	<b>NMIECC</b>	<b>NMIVDDP</b>	<b>NMIVDDC</b>	<b>NMIOCDs</b>	<b>NMI-FLASH</b>	<b>NMI-OSCCLK</b>	<b>NMIWDT</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>NMIWDT</b>	0	rw	<b>WDT NMI 使能</b> 0 禁止 WDT NMI 1 使能 WDT NMI
<b>NMIOSCCLK</b>	1	rw	<b>48 MHz 或 75 kHz 振荡器时钟丢失 NMI 使能</b> 0 禁止 48 MHz 或 75 kHz 振荡器时钟丢失 NMI 1 使能 48 MHz 或 75 kHz 振荡器时钟丢失 NMI
<b>NMIFLASH</b>	2	rw	<b>Flash 定时器 NMI 使能</b> 0 禁止 Flash 定时器 NMI 1 使能 Flash 定时器 NMI
<b>NMIOCDs</b>	3	rw	<b>OCDS NMI 使能</b> 0 禁止 OCDS NMI 1 使能 OCDS NMI
<b>NMIVDDC</b>	4	rw	<b>VDDC 预警 NMI 使能</b> 0 禁止 VDDC 预警 NMI 1 使能 VDDC 预警 NMI
<b>NMIVDDP</b>	5	rw	<b>VDDP 预警 NMI 使能</b> 0 禁止 VDDP 预警 NMI 1 使能 VDDP 预警 NMI
<b>NMIECC</b>	6	rw	<b>ECC 出错 NMI 使能</b> 0 禁止 ECC 出错 NMI 1 使能 ECC 出错 NMI

符号	位	类型	描述
NMIXTALCLK	7	rw	<b>32.768 kHz XTAL 时钟丢失 NMI 使能</b>
			0 禁止 32.768 kHz XTAL 时钟丢失 NMI
			1 使能 32.768 kHz XTAL 时钟丢失 NMI

注： 外部供电电压不超过 3.3 V 时, 用户必须禁用 NMIVDDP。

## 9.5.2 外部中断控制寄存器

共有 7 个外部中断，EXT\_INT[6:0]，从端口送入 XC83x。外部中断引脚的分配情况归纳见 [表 9-4](#)。外部中断可由信号的正沿、负沿、或任意沿触发。寄存器 EXICON0 和 EXICON1 指定触发外部中断的有效跳变沿。在外部中断中，外部中断 0 和外部中断 1 可直接送入内核、无需跳变沿检测。送入内核的信号可通过 TCON 寄存器中的位 IT0 和 IT1 进一步设定为由低电平或负跳变激活。不过，若在 SCU 中进行跳变沿检测，TCON.IT0/1 必须设置为下降沿触发。若 SCU 检测到有效的跳变沿事件，将产生两个 CCLK 周期的低脉冲用于内核检测。

若外部中断由正（负）沿触发，外部中断请求引脚必须至少保持一个 CCLK 周期的低（高）电平，接着至少保持一个 CCLK 周期的高（低）电平，从而确保该跳变被识别。如果外部中断 0 和外部中断 1 的边沿检测被旁路，外部中断请求引脚必须至少保持两个 CCLK 周期的高或低电平。

外部中断 2 到 6 和其它中断源共用两个中断节点。因此，除相应的中断节点使能之外，外部中断 2 到 6 中的任一个可被单独禁用，它们在复位后缺省设置为禁用。

**表 9-4 外部中断 [6:0] 引脚功能及选择**

引脚	功能	说明	由以下位 / 位域选择
P0.5	EXINT0_0	外部中断输入 0	MODPISEL1.EXINT0IS = 000 <sub>B</sub>
P0.6	EXINT0_1		MODPISEL1.EXINT0IS = 001 <sub>B</sub>
P1.0	EXINT0_2		MODPISEL1.EXINT0IS = 010 <sub>B</sub>
P2.0	EXINT0_3		MODPISEL1.EXINT0IS = 011 <sub>B</sub>
P2.7	EXINT0_4		MODPISEL1.EXINT0IS = 100 <sub>B</sub>
P3.1	EXINT0_5		MODPISEL1.EXINT0IS = 101 <sub>B</sub>
P3.2	EXINT0_6		MODPISEL1.EXINT0IS = 110 <sub>B</sub>
P0.4	EXINT1_0	外部中断输入 1	MODPISEL1.EXINT1IS = 0 <sub>B</sub>
P2.1	EXINT1_1		MODPISEL1.EXINT1IS = 1 <sub>B</sub>
P2.2	EXINT2_0	外部中断输入 2	MODPISEL1.EXINT2IS = 0 <sub>B</sub>
P3.0	EXINT2_1		MODPISEL1.EXINT2IS = 1 <sub>B</sub>
P2.3	EXINT3	外部中断输入 3	—

表 9-4 外部中断 [6:0] 引脚功能及选择

引脚	功能	说明	由以下位 / 位域选择
P1.2	EXINT4	外部中断输入 4	—
P1.4	EXINT5	外部中断输入 5	—
P2.6	EXINT5	外部中断输入 6	—

在访问 EXICON0、EXICON1 和 MODPISEL1 寄存器之前必须先对 SCU\_PAGE 寄存器中的位域 PAGE 进行设置。

注：多个外部中断支持复用输入引脚。当切换输入时，应当考虑相关引脚上的有效沿 / 电平触发选择和相应电平，以防止产生不需要的中断。

### EXICON0

外部中断控制寄存器 0

(EF<sub>H</sub>)

复位值：F0<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
EXINT3		EXINT2		EXINT1		EXINT0	
rw		rw		rw		rw	

符号	位	类型	描述
EXINT0	1:0	rw	<b>外部中断 0 触发选择</b> 00 下降沿触发中断 01 上升沿触发中断 10 上升沿和下降沿均触发中断 11 SCU 中的边沿检测旁路。中断请求信号直接送入内核
EXINT1	3:2	rw	<b>外部中断 1 触发选择</b> 00 下降沿触发中断 01 上升沿触发中断 10 上升沿和下降沿均触发中断 11 SCU 中的边沿检测旁路。中断请求信号直接送入内核
EXINT2	5:4	rw	<b>外部中断 2 触发选择</b> 00 下降沿触发中断 01 上升沿触发中断 10 上升沿和下降沿均触发中断 11 外部中断 2 被禁止

**中断系统**

符号	位	类型	描述
<b>EXINT3</b>	7:6	rw	<b>外部中断 3 触发选择</b> 00 下降沿触发中断 01 上升沿触发中断 10 上升沿和下降沿均触发中断 11 外部中断 3 被禁止

**EXICON1**

外部中断控制寄存器 1

**(F4<sub>H</sub>)**

 复位值: **3F<sub>H</sub>**
**RMAP: 0, PAGE: 0**

7	6	5	4	3	2	1	0
0	EXINT6		EXINT5		EXINT4		
r	rw		rw		rw		

符号	位	类型	描述
<b>EXINT4</b>	1:0	rw	<b>外部中断 4 触发选择</b> 00 下降沿触发中断 01 上升沿触发中断 10 上升沿和下降沿均触发中断 11 外部中断 4 被禁止
<b>EXINT5</b>	3:2	rw	<b>外部中断 5 触发选择</b> 00 下降沿触发中断 01 上升沿触发中断 10 上升沿和下降沿均触发中断 11 外部中断 5 被禁止
<b>EXINT6</b>	5:4	rw	<b>外部中断 6 触发选择</b> 00 下降沿触发中断 01 上升沿触发中断 10 上升沿和下降沿均触发中断 11 外部中断 6 被禁止
<b>0</b>	7:6	r	<b>保留</b> 读操作返回 0，应写入 0。

## MODISEL1

外设输入选择寄存器 1

(F4<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
EXINT2IS	EXINT1IS	EXINT0IS			URRIS		
rw	rw	rw			rw		

符号	位	类型	描述
EXINT0IS	[5:3]	rw	<b>外部中断 0 输入选择</b> 000 选择外部中断输入 EXINT0_0 001 选择外部中断输入 EXINT0_1 010 选择外部中断输入 EXINT0_2 011 选择外部中断输入 EXINT0_3 100 选择外部中断输入 EXINT0_4 101 选择外部中断输入 EXINT0_5 110 选择外部中断输入 EXINT0_6 111 不使用
EXINT1IS	6	rw	<b>外部中断 1 输入选择</b> 0 选择外部中断输入 EXINT1_0 1 选择外部中断输入 EXINT1_1
EXINT2IS	7	rw	<b>外部中断 2 输入选择</b> 0 选择外部中断输入 EXINT2_0 1 选择外部中断输入 EXINT2_1

## TCON

定时器和计数器控制 / 状态寄存器

(88<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
rwh	rw	rwh	rw	rwh	rw	rwh	rw



符号	位	类型	描述
IT0	0	rw	外部中断 0 电平 / 边沿触发控制 0 选择低电平触发外部中断 0 1 选择下降沿触发外部中断 0
IT1	2	rw	外部中断 1 电平 / 边沿触发控制 0 选择低电平触发外部中断 1 1 选择下降沿触发外部中断 1

### 9.5.3 中断标志寄存器

不同中断源的中断标志存放在多个特殊功能寄存器（SFR）中。若软件和硬件同时访问某标志位，硬件访问占优。在访问 IRCONx 寄存器之前必须先对 SCU\_PAGE 寄存器中的位域 PAGE 进行设置。

#### IRCON0

中断请求寄存器 0

(F2<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	EXINT6	EXINT5	EXINT4	EXINT3	EXINT2	0	
r	rwh	rwh	rwh	rwh	rwh	r	

符号	位	类型	描述
EXINTx (x = 2 - 6)	[6:2]	rwh	外部中断 x 的中断标志 该位由硬件置位，只能由软件清零。 0 未产生中断事件 1 已产生中断事件
0	[1:0], 7	r	保留 读操作返回 0，应写入 0。

**IRCON1**

中断请求寄存器 1

(F3<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
	<b>0</b>		<b>ADCSR1</b>	<b>ADCSR0</b>	<b>RIR</b>	<b>TIR</b>	<b>EIR</b>
	r		rwh	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>EIR</b>	0	rwh	<b>SSC 出错中断标志</b> 该位由硬件置位，只能由软件清零。 0 未产生中断事件。 1 已产生中断事件
<b>TIR</b>	1	rwh	<b>SSC 发送中断标志</b> 该位由硬件置位，只能由软件清零。 0 未产生中断事件 1 已产生中断事件
<b>RIR</b>	2	rwh	<b>SSC 发送中断标志</b> 该位由硬件置位，只能由软件清零。 0 未产生中断事件 1 已产生中断事件
<b>ADCSR0</b>	3	rwh	<b>ADC 中断标志 0</b> 该位由硬件置位，只能由软件清零。 0 未产生中断事件 1 已产生中断事件
<b>ADCSR1</b>	4	rwh	<b>ADC 中断标志 1</b> 该位由硬件置位，只能由软件清零。 0 未产生中断事件 1 已产生中断事件
<b>0</b>	[7:5]	r	<b>保留</b> 读操作返回 0，应写入 0。

**IRCON2**

中断请求寄存器 2

(F5<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
	0		CCU6SR1		0		CCU6SR0
	r		rwh		r		rwh

符号	位	类型	描述
CCU6SR0	0	rwh	<b>CCU6 中断标志 0</b> 该位由硬件置位，只能由软件清零。 0 未产生中断事件 1 已产生中断事件
CCU6SR1	4	rwh	<b>CCU6 中断标志 1</b> 该位由硬件置位，只能由软件清零。 0 未产生中断事件 1 已产生中断事件
0	[3:1], [7:5]	r	<b>保留</b> 读操作返回 0，应写入 0。

**IRCON3**

中断请求寄存器 3

(F6<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
	0		CCU6SR3		0		CCU6SR2
	r		rwh		r		rwh

符号	位	类型	描述
CCU6SR3	4	rwh	<b>CCU6 中断标志 3</b> 该位由硬件置位，只能由软件清零。 0 未产生中断事件 1 已产生中断事件

## 中断系统

符号	位	类型	描述
<b>CCU6SR2</b>	0	rwh	<b>CCU6 中断标志 2</b> 该位由硬件置位，只能由软件清零。 0 未产生中断事件 1 已产生中断事件
<b>0</b>	[3:1], [7:5]	r	<b>保留</b> 读操作返回 0，应写入 0。

### TCON

定时器和计数器控制 / 状态寄存器

(88<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
<b>TF1</b>	<b>TR1</b>	<b>TF0</b>	<b>TR0</b>	<b>IE1</b>	<b>IT1</b>	<b>IE0</b>	<b>IT0</b>
rwh	rw	rwh	rw	rwh	rw	rwh	rw

符号	位	类型	描述
<b>IE0</b>	1	rwh	<b>外部中断 0 标志</b> 检测到外部中断 0 事件时，该标志由硬件置位。 处理器响应中断转入中断服务程序后，该标志由硬件清零，也可由软件清零。
<b>IE1</b>	3	rwh	<b>外部中断 1 标志</b> 检测到外部中断 1 事件时，该标志由硬件置位。 处理器响应中断转入中断服务程序后，该标志由硬件清零，也可由软件清零。
<b>TF0</b>	5	rwh	<b>定时器 T0 溢出标志</b> 定时器 T0 溢出时，该标志由硬件置位。处理器响应中断转入中断服务程序后，该标志由硬件清零，也可由软件清零。
<b>TF1</b>	7	rwh	<b>定时器 T1 溢出标志</b> 定时器 T1 溢出时，该标志由硬件置位。处理器响应中断转入中断服务程序后，该标志由硬件清零，也可由软件清零。

## SCON

UART 控制 / 状态寄存器

(98<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
<b>SM0</b>	<b>SM1</b>	<b>SM2</b>	<b>REN</b>	<b>TB8</b>	<b>RB8</b>	<b>TI</b>	<b>RI</b>
rw	rw	rw	rw	rw	rwh	rwh	rwh

符号	位	类型	描述
<b>RI</b>	0	rwh	<b>串行接口接收中断标志</b> 完成一个串行数据字节的接收后, 该标志由硬件置位; 必须由软件清零。
<b>TI</b>	1	rwh	<b>串行接口发送中断标志</b> 完成一个串行数据字节的发送后, 该标志由硬件置位; 必须由软件清零。

## NMISR

NMI 状态寄存器

(F7<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
<b>FNMI XTALCLK</b>	<b>FNMI ECC</b>	<b>FNMI VDDP</b>	<b>FNMI VDDC</b>	<b>FNMI OCDS</b>	<b>FNMI FLASH</b>	<b>FNMI OSCCLK</b>	<b>FNMIWDT</b>
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>FNMIWDT</b>	0	rwh	<b>WDT NMI 标志</b> 0 未发生 WDT NMI 1 已发生 WDT 预警
<b>FNMIOSCCLK</b>	1	rwh	<b>48 MHz 或 75 kHz 振荡器时钟 NMI 标志</b> 0 未发生 48 MHz 或 75 kHz 振荡器 NMI 1 已发生 48 MHz 或 75 kHz 时钟丢失
<b>FNMIFLASH</b>	2	rwh	<b>Flash 操作完成 NMI 标志</b> 0 未发生 Flash NMI 1 Flash 操作已完成

## 中断系统

符号	位	类型	描述
<b>FNMIOCDs</b>	3	rwh	<b>OCDS NMI 标志</b> 0 未发生 OCDS NMI 1 监控模式下已发生 JTAG 接收或用户中断请求
<b>FNMIVDDC</b>	4	rwh	<b>VDDC 预警 NMI 标志</b> 0 未发生 $V_{DDC}$ NMI 1 已发生 $V_{DDC}$ 预警（跌至 2.3V 以下）
<b>FNMIVDDP</b>	5	rwh	<b>VDDP 预警 NMI 标志</b> 0 未发生 $V_{DDP}$ NMI 1 已发生 $V_{DDP}$ 预警（若外部电压为 5.0V，对应的 $V_{DDP}$ 预警电压 4.0V）
<b>FNMIECC</b>	6	rwh	<b>ECC NMI 标志</b> 0 未发生 ECC 错误 1 ECC 已出错
<b>FNMIXTALCLK</b>	7	rwh	<b>32.768 kHz XTAL 振荡器时钟 NMI 标志</b> 该位由硬件置位，只能由软件清零。 0 未发生 32.768 kHz XTAL 时钟丢失 NMI 1 已发生 32.768 kHz XTAL 时钟丢失

注： *NMISR* 寄存器只能由软件清零，或在上电复位之后被置为缺省值。在其它复位方式下，该寄存器的值保持不变，从而可检查引发前次 NMI 的中断源。

### 9.5.4 中断优先级寄存器

每个中断节点的优先级可分别编程设定（共有四级优先级可用）。用两对中断优先级寄存器来设定每个中断向量的优先级。第一对中断优先级寄存器是 IP 和 IPH；第二对中断优先级寄存器是 IP1 和 IPH1。

每对中断优先级寄存器中的对应位相组合，共同确定该中断节点的优先级（4 级优先级供选择），见表 9-5。

表 9-5 中断优先级选择

IPH.x / IPH1.x	IP.x / IP1.x	优先级
0	0	优先级 0（最低）
0	1	优先级 1
1	0	优先级 2
1	1	优先级 3（最高）

注：由于 NMI 具有最高优先级（高于优先级3），故不需要通过表 9-5 进行优先级选择。

IP  
中断优先级寄存器 (B8<sub>H</sub>) 复位值: 00<sub>H</sub>  
RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
0	PT2	PS	PT1	PX1	PT0	PX0	
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
PX0	0	rw	中断节点 XINTR0 的优先级低位
PT0	1	rw	中断节点 XINTR1 的优先级低位
PX1	2	rw	中断节点 XINTR2 的优先级低位
PT1	3	rw	中断节点 XINTR3 的优先级低位 3
PS	4	rw	中断节点 XINTR4 的优先级低位
PT2	5	rw	中断节点 XINTR5 的优先级低位
0	[7:6]	r	保留 读操作返回 0，应写入 0。

**IPH**
**中断优先级寄存器，高位字节**
**(B9<sub>H</sub>)**
**复位值：00<sub>H</sub>**
**RMAP: X, PAGE: X**

7	6	5	4	3	2	1	0
0		PT2H	PSH	PT1H	PX1H	PT0H	PX0H
r		rw	rw	rw	rw	rw	rw

符号	位	类型	描述
PX0H	0	rw	中断节点 XINTR0 的优先级高位
PT0H	1	rw	中断节点 XINTR1 的优先级高位
PX1H	2	rw	中断节点 XINTR2 的优先级高位
PT1H	3	rw	中断节点 XINTR3 的优先级高位
PSH	4	rw	中断节点 XINTR4 的优先级高位
PT2H	5	rw	中断节点 XINTR5 的优先级高位
0	[7:6]	r	保留 读操作返回 0，应写入 0。

**IP1**
**中断优先级寄存器 1**
**(F8<sub>H</sub>)**
**复位值：04<sub>H</sub>**
**RMAP: X, PAGE: X**

7	6	5	4	3	2	1	0
PCCIP3	PCCIP2	PCCIP1	PCCIP0	PXM	PX2	PSSC	PADC
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
PADC	0	rw	中断节点 XINTR6 的优先级低位
PSSC	1	rw	中断节点 XINTR7 的优先级低位
PX2	2	rw	中断节点 XINTR8 的优先级低位
PXM	3	rw	中断节点 XINTR9 的优先级低位
PCCIP0	4	rw	中断节点 XINTR10 的优先级低位
PCCIP1	5	rw	中断节点 XINTR11 的优先级低位



符号	位	类型	描述
PCCIP2	6	rw	中断节点 XINTR12 的优先级低位
PCCIP3	7	rw	中断节点 XINTR13 的优先级低位

## IPH1

中断优先级寄存器 1，高位字节

(F9<sub>H</sub>)

复位值：04<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
PCCIP3H	PCCIP2H	PCCIP1H	PCCIP0H	PXMH	PX2H	PSSCH	PADCH
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
PADCH	0	rw	中断节点 XINTR6 的优先级高位
PSSCH	1	rw	中断节点 XINTR7 的优先级高位
PX2H	2	rw	中断节点 XINTR8 的优先级高位
PXMH	3	rw	中断节点 XINTR9 的优先级高位
PCCIP0H	4	rw	中断节点 XINTR10 的优先级高位
PCCIP1H	5	rw	中断节点 XINTR11 的优先级高位
PCCIP2H	6	rw	中断节点 XINTR12 的优先级高位
PCCIP3H	7	rw	中断节点 XINTR13 的优先级高位

## 9.6 中断标志概览

中断事件的中断标志存放在不同的 SFR 中。表 9-6 列出各中断标志所属的 SFR。中断标志的具体信息将在相关外设章节中给出。

表 9-6 中断标志的位置

中断事件	中断标志	SFR
定时器 T0 溢出	TF0	TCON
定时器 T1 溢出	TF1	TCON
定时器 T2 溢出	TF2	T2_T2CON
定时器 T2 外部事件	EXF2	T2_T2CON
UART 接收	RI	SCON

**表 9-6 中断标志的位置**

中断事件	中断标志	SFR
UART 发送	TI	SCON
LIN 同步字节结束	EOFSYN	LINST
LIN 同步字节出错	ERRSYN	LINST
外部中断 0	IE0	TCON
外部中断 1	IE1	TCON
外部中断 2	EXINT2	IRCON0
外部中断 3	EXINT3	IRCON0
外部中断 4	EXINT4	IRCON0
外部中断 5	EXINT5	IRCON0
外部中断 6	EXINT6	IRCON0
RTC 比较 / 唤醒中断	CFRTC	RTC_RTCON
RTC 秒定时中断	SFRTC	RTC_RTCON
ADC 服务请求 0	ADCSR0	IRCON1
ADC 服务请求 1	ADCSR1	IRCON1
CORDIC 计算结束	EOC	CD_STATC
MDU 结果准备就绪	IRDY	MDU_MDUSTAT
MDU 出错	IERR	MDU_MDUSTAT
SSC 错误	EIR	IRCON1
SSC 发送	TIR	IRCON1
SSC 接收	RIR	IRCON1
IIC 中断	IFLG	CNTR
P1.0 HCPADA 过电流中断	P0_OCF	P1_OCD
P1.1 HCPADA 过电流中断	P1_OCF	P1_OCD
P1.2 HCPADA 过电流中断	P2_OCF	P1_OCD
P1.3 HCPADA 过电流中断	P3_OCF	P1_OCD
CCU6 节点 0 中断	CCU6SR0 <sup>1)</sup>	IRCON2
CCU6 节点 1 中断	CCU6SR1 <sup>1)</sup>	IRCON2
CCU6 节点 2 中断	CCU6SR2 <sup>1)</sup>	IRCON3
CCU6 节点 3 中断	CCU6SR3 <sup>1)</sup>	IRCON3
LEDTSCU 时间片中断	TSF	LTS_GLOBCTL1

**表 9-6 中断标志的位置**

中断事件	中断标志	SFR
LEDTSCU 时帧中断	TFF	LTS_GLOBCTL1
WDT NMI	FNMIWDT	NMISR
48 MHz 和 75 kHz 振荡器 NMI	FNMIOSCLK	NMISR
32.768 kHz XTAL 振荡器 NMI	FNMIXTALCLK	NMISR
Flash 操作完成 NMI	FNMIFLASH	NMISR
OCDS NMI	FNMIOCDS	NMISR
VDDC 预警 NMI	FNMIVDDC	NMISR
VDDP 预警 NMI	FNMIVDDP	NMISR
Flash ECC NMI	FNMIIECC	NMISR

1) 可通过 CCU6 寄存器 INPL/INPH 将每个 CCU6 中断分配给任意一个 CCU6 中断节点 [3:0]。

## 10 调试系统

本章将简要介绍 XC800 调试系统，着重对 OCDS 单元（提供硬件支持调试功能）予以说明。

### 10.1 概述

调试系统由片上调试支持（OCDS）单元和 ROM 中的调试监控程序构成，它为基于 XC800 的控制系统提供基本调试功能，通过调试接口由外部工具直接控制。

#### 特性

OCDS 支持以下调试特性：

- 在程序存储器的指令地址上和指定的地址范围内设置断点
- 在内部 RAM 地址段设置断点
- 在 Flash/RAM 代码区可设置数量不限的软件断点
- 通过调试接口处理外部暂停
- 程序代码单步调试
- 出现 IIRAM 读 / 写断点时用户处理 OCDS NMI

#### 10.1.1 调试系统的组成部分

下面将简要介绍调试系统的组成部分：

##### 调试接口

可通过调试接口访问器件（如读写数据）。XC83x 支持的调试接口有：

- 单引脚 DAP（SPD）
- 1 或 2 引脚 UART

SPD 接口代表单引脚器件访问端口，它是英飞凌微控制器的标准化端口，仅使用单引脚 DAP1 进行串行数据读写。

UART 是标准接口，但不支持 OCDS 硬件命令。

##### 监控程序

OCDS 监控程序（或 OCDS 固件）存放在 Boot ROM 中。调试器发出命令后，监控程序执行指令以访问可寻址的地址单元（如器件的内存和 SFR）。

##### 片上调试支持单元（OCDS）

OCDS 硬件是调试系统的核心，它通过与 XC800 CPU 的接口来控制芯片的调试。

### 10.2 产品信息

本节给出和 OCDS 相关的产品信息。

## 10.2.1 引脚配置

**表 10-1** 列出 XC83x 中的调试引脚功能。

**表 10-1 XC83x 引脚功能和选择**

引脚	功能	说明	由 ... 选择
P3.2	SPD_0	单引脚调试访问端口：数据	根据 BMI 值进行选择，详见第 5 章的描述。
P0.6	SPD_1		
P1.0	SPD_2		

## 10.2.2 时钟配置

OCDS 以 CPU 频率 8 MHz 或 24 MHz 工作。

## 10.2.3 中断事件和中断节点分配

发生暂停事件时，可以不进入监控模式，而是转入执行 OCDS NMI 服务程序。在这种配置下，OCDS NMI 引发的操作完全由用户程序控制，后面将对该使用方法进行详细说明。

OCDS 模块产生的中断事件以及相应的中断使能位和标志位归纳见 **表 10-2**。

**表 10-2 OCDS 中断事件**

事件	事件中中断使能位	事件标志位
IRAM 读事件	MMICR.NMIRRE	MMICR.FNMIRR
IRAM 写事件	MMICR.NMIRWE	MMICR.FNMIRW

**表 10-3** 给出 OCDS 中断事件的中断节点分配情况。

**表 10-3 OCDS 中断事件的中断节点控制**

事件	中断节点使能位	中断节点标志位	中断向量地址
IRAM 读事件	NMICON.NMIOCDS	NMISR.FNMIOCDS	73 <sub>H</sub>
IRAM 写事件			

## 10.2.4 调试挂起控制

监控模式有效时，用户可选择允许某些模块挂起。可使能调试挂起的模块有：

- 看门狗定时器
- CCU6 的定时器 T12
- CCU6 的定时器 T13
- 定时器 T2

- RTC 定时器
- LEDTSCU 计数器

若调试挂起，在监控模式下，这些模块的功能被冻结（比如定时器停止计数）。寄存器 MODSUSP 用于控制调试挂起功能。在访问 MODSUSP 寄存器之前必须先对 SCU\_PAGE 寄存器中的位域 PAGE 进行设置。

## MODSUSP

模块挂起控制寄存器

(F6<sub>H</sub>)

复位值：01<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	LTSSUSP	RTCSUSP	T2SUSP	T13SUSP	T12SUSP	WDTSUSP	
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
WDTSUSP	0	rw	<b>SCU WDT 调试挂起位</b> 0 WDT 不被挂起 1 WDT 将被挂起
T12SUSP	1	rw	<b>定时器 T12 调试挂起位</b> 0 捕获 / 比较单元中的定时器 T12 不被挂起 1 捕获 / 比较单元中的定时器 T12 将被挂起 此外，挂起后 T12 PWM 输出置为无效电平、捕获输入禁用。
T13SUSP	2	rw	<b>定时器 T13 调试挂起位</b> 0 捕获 / 比较单元中的定时器 T13 不被挂起 1 捕获 / 比较单元中的定时器 T13 将被挂起 此外，挂起后 T13 PWM 输出置为无效电平、捕获输入禁用。
T2SUSP	3	rw	<b>定时器 T2 调试挂起位</b> 0 定时器 T2 不被挂起 1 定时器 T2 将被挂起
RTCSUSP	4	rw	<b>RTC 调试挂起位</b> 0 RTC 不被挂起 1 RTC 将被挂起
LTSSUSP	5	rw	<b>LED 和触摸感应计数器调试挂起位</b> 0 LED 和触摸感应计数器不被挂起 1 LED 和触摸感应计数器将被挂起

符号	位	类型	描述
0	[7:6]	r	保留 读取返回 0；应写入 0。

## 10.2.5 JTAG ID

XC83x 器件的 JTAG ID 见 表 10-4。

表 10-4 JTAG ID

器件类型	器件名	JTAG ID
Flash	XC83x-2FGI	101BA083 <sub>H</sub>
	XC83x-2FRI	
	XC83x-1FRI	101BB083 <sub>H</sub>

## 10.3 调试系统的功能概述

XC800 调试操作基于 OCDS 硬件和 监控程序（存放在 ROM 中）的紧密配合。

下面将详细说明相关的操作。

### 10.3.1 识别调试事件

在发生调试事件时，OCDS 负责启动相应的监控程序。

XC83x 支持的调试事件（断点）有以下几种类型：

#### 硬件断点

最多可设置 4 个硬件断点，发生以下情况时引发暂停：

- 从指定的程序存储器（IP）地址单元中获取代码 - 在用户程序的一条指令上设置暂停
- 从指定的程序存储器（IP）地址段上获取代码 - 在用户程序的多条连续指令上设置暂停
- 对指定的内部 RAM 地址单元进行读或写操作（可选） - 数据移出 / 移入地址单元时引发暂停

在指令地址上设置断点时，HWBPx 定义 16 位地址；在 IRAM 地址上设置断点时，HWBP2/3L 和 HWBP2/3H 定义 8 位 IRAM 地址区间。

所支持的断点设置包括：

- 断点 0
- 断点 1
  - 两个点断点
    - 指令地址 = HWBP0 和
    - 指令地址 = HWBP1 或

- 一个区间断点  
HWBP0 <= 指令地址 <= HWBP1
- 断点 2
  - 一个点断点, 指令地址 = HWBP2, 或
  - 一个区间断点, HWBP2L <= IRAM 读地址 <= HWBP2H
- 断点 3
  - 一个点断点, 指令地址 = HWBP3, 或
  - 一个区间断点, HWBP3L <= IRAM 写地址 <= HWBP3H

若将区间断点的高低地址设置为同一地址值, 将会产生点断点。

### 软件断点

这类断点使用 XC800 专用的 (非 8051 标准) TRAP 指令, 同时还将扩展操作 (EO) 寄存器中的位 TRAP\_EN 置 1。

读取 TRAP 指令时, 产生一个断点, 执行相应的暂停动作。

实际上, 软件断点的行为与指令地址上的点断点相似, 区别在于前者由程序代码产生, 而后者由专门的比较逻辑实现。

在用户程序中可用软件断点替换原先的指令代码, 数量不限。但只能在已实现的 RAM/Flash 的地址上设置断点。

### 外部暂停

下面的情况可引发外部暂停:

- 由 OCDS 命令通过调试接口请求: 外部调试器可激活监控程序并中断运行在目标 XC800 上的用户程序。

### 10.3.2 激活监控程序

一旦识别到调试事件且已正确设置 OCDS 寄存器, OCDS 的主要响应即激活监控程序 (位于 ROM 中)。

OCDS 确保始终安全启动监控程序, 并在调试过程中完全独立于当前的系统状态。此外, 监控程序工作时产生的中断请求将不会干扰监控程序正常执行。

一旦监控程序开始工作, 它将运行在自己的堆栈和数据存储器中, 从而确保当控制返回用户程序时, 所有的 CPU 和内存资源不被改变。因此, OCDS 调试完全是非破坏性的。

监控程序可提供的功能包括:

- 通过调试接口与外部调试器通信
- 读/写任意存储器地址和特殊功能寄存器 (SFR), 包括指令指针和密码保护位
- 配置 OCDS 以及设置/删除断点
- 单指令执行 (单步执行)

*注: 本手册不提供监控程序功能的详细描述。*



### 10.3.3 调试挂起控制

XC83x OCDS 除了支持基本调试功能 - 设置断点和暂停用户软件的执行，它还支持调试期间的模块挂起。

只要器件处于监控模式（即，用户软件不运行、处于暂停状态）且片上软件使能调试挂起功能，模块可在调试期间挂起。

这是一种非常有用的特性，尤其对看门狗定时器而言：它可避免调试期间 WDT 意外复位。通常，为定时器模块提供调试挂起功能。暂停期间终止计数器工作可将器件的状态完全“冻结”。

通常，复位之后禁用调试挂起控制位（OCDS 中的全局使能和 SCU 中的各模块使能位），即缺省情况下，暂停不会使任何模块挂起。通常而言，若要进行调试，器件会从 OCDS 模式下启动，在运行任何用户程序之前将激活监控程序。调试器应提供挂起控制的用户配置。

### 10.3.4 运行监控程序

由于监控程序运行在自己的堆栈和数据存储器中，因此内核的状态被完整保存，以便返回用户程序时恢复该状态。

一旦开始执行监控程序，它可读写所有系统资源，数据可与调试器进行通信。

### 10.3.5 返回用户程序

接收到返回用户程序的请求后，监控程序根据需要设置内核寄存器（包括指令指针 IP）的状态：可以和发生断点前的状态相同，也可以由调试器在暂停阶段改变部分状态。随后跳转到 IP 指向的地址单元，系统返回用户模式。

### 10.3.6 单步执行

为了实现单步执行，OCDS 开始的操作和返回用户程序时相同，但同时置位标志位 MMCR.MSTEP。通过这种方式，在重新进入监控之前，只执行一条用户指令。

## 10.4 断点产生模块

只有在监控模式下才能产生断点。

**注意：** 为了能够通过监控模式进行调试，应用程序不应改变扩展操作（EO）寄存器中的位 **TRAP\_EN**。

断点可划分为以下三种类型：

- **指令执行前暂停**  
暂停刚好发生在暂停指令（即引发该断点的指令）执行之前。因此，该暂停指令是用户程序中将要执行的下一条指令，但只在相应的调试动作发生后执行。
- **指令执行后暂停**  
在断点指令（即引发该断点的指令）执行之后暂停立即发生。因此在发生相应的调试动作时，该断点指令已被执行。
- **立即暂停**  
这类事件和 XC83x 内部的指令执行不同步，在这种情况下不存在“引起调试事件的指令”。调试事件一出现，OCDS“尽可能快的”执行调试动作。

### 10.4.1 产生硬件断点

该模块围绕一组比较器构成，专门用于识别两类硬件断点：

#### 10.4.1.1 指令地址断点

只有当断点地址和指令首字节（即指令操作码）地址匹配时，才会产生这类 IP 断点。对于 2 字节和 3 字节指令，指令的第二和第三字节的地址不产生暂停。

IP 断点属于“指令执行前暂停”，因此只有在相应调试动作执行之后才执行该暂停指令。

#### 地址比较器、控制位和标志位

用于产生 IP 断点的比较器说明如下：

- **CMP0**  
两个 **HWBP0** 寄存器（A 侧）和 16 位程序存储器地址总线（B 侧）之间的 16 位比较器。  
产生两个输出信号：
  - $A=B$  - （IP= **HWBP0** 时暂停）
  - $A<B$  - 见下面的说明。
- **CMP1**  
两个 **HWBP1** 寄存器（A 侧）和 16 位程序存储器地址总线（B 侧）之间的 16 位比较器。  
产生两个输出信号：
  - $A=B$  - （IP= **HWBP1** 时暂停）
  - $A>=B$  的同时 CMP0 的  $A<B$  - （**HWBP0**  $=<IP<=$  **HWBP1**，产生中断）
- **CMPIP2**  
两个 **HWBP2** 寄存器（A 侧）和 16 位程序存储器地址总线（B 侧）之间的 16 位比较器。  
产生一个输出信号：
  - $A=B$  - （IP = **HWBP2** 时暂停）

## • CMPIP3

两个 **HWBP3** 寄存器 (A 侧) 和 16 位程序存储器地址总线 (B 侧) 之间的 16 位比较器。产生一个输出信号:

- $A=B$  - (IP = **HWBP3** 时暂停)

### 10.4.1.2 IRAM 地址断点

对内部数据存储器 (IRAM) 执行读写操作的地址位于给定地址区间内时, 产生这类 IRAM 地址断点。

IRAM 断点属于“指令执行后时暂停”, 在断点指令执行之后立即执行相应调试动作。XC83x OCDS 只支持 IRAM 地址的区间断点。

OCDS 能够区分读操作断点和写操作断点。

读断点和写断点的具体处理不同。

### 地址比较器、控制位和标志位

用于产生内存访问断点的比较器说明如下:

## • CMPRL2, CMPRH2

两个 8 位比较器:

- **HWBP2L** 寄存器 (A 侧) 和 8 位 IRAM 源地址总线 (B 侧) 之间的比较器 **CMPRL2**, 产生  $A \leq B$ ;
- **HWBP2H** 寄存器 (A 侧) 和 8 位 IRAM 源地址总线 (B 侧) 之间的比较器 **CMPRH2**, 产生  $A \geq B$ 。

因此:

**HWBP2L** = <IRAM 读地址 <= **HWBP2H** 时暂停。

## • CMPWL3, CMPWH3

两个 8 位比较器:

- **HWBP3L** 寄存器 (A 侧) 和 8 位 IRAM 目的地址总线 (B 侧) 之间的比较器 **CMPWL3**, 产生  $A \leq B$ ;
- **HWBP3H** 寄存器 (A 侧) 和 8 位 IRAM 目的地址总线 (B 侧) 之间的比较器 **CMPWH3**, 产生  $A \geq B$ 。因此:

**HWBP3L** = <IRAM 写地址 <= **HWBP3H** 时暂停。

### 10.4.1.3 跟踪暂停地址的改变

硬件断点产生机制能够确保一旦满足暂停条件, 立即强制 XC800 内核进入调试模式。此时程序计数器的值冻结, 该值对应内核若未转入调试模式、正常工作时将要读取的指令的地址。

进入调试模式时, 监控程序将确定引发暂停的事件并将相关信息 (如 IRAM 读 / 写) 送至调试器。

### 10.4.2 处理外部暂停

下面将说明如何产生并处理外部暂停请求：

#### 1. 请求来源

- a) 通过 SPD 发送调试器请求（专用命令），且 **MMODE\_r=1**, **RRF\_r=0** 及 数据字节 = **A5<sub>H</sub>**，同时：

**MMCR2.MMODE=0** 且 **MMCR2.ALTDI=0**

#### 2. 内核停止执行当前命令，

内核进入调试模式

*注：在执行 NMI 中断服务程序时不处理外部暂停。*

*详细说明参见[章节 10.7](#)。*

### 10.4.3 处理软件断点

由于软件断点属于“指令执行前暂停”，因此只有在相应调试动作执行之后才执行断点上的指令。

内核一旦取指 **TRAP** 指令，软件断点生效，从而进入调试模式。

**注意：** *用户程序切勿清除 **EO** 寄存器中的位 **TRAP\_EN**。*

OCDS 有两种响应方式：

- 识别到暂停事件时，进入监控模式
- 对软件暂停不予理睬

当上述条件不满足时（如软件断点被禁用），发生这种情况。

执行到禁用的软件断点时，用户程序将在该指令（**TRAP** 指令）之后暂停 4 个时钟周期。

## 10.5 不进入监控模式的断点响应方式

若监控模式禁用，产生暂停事件时，OCDS 系统不启动监控程序（**MM\_no\_entry**），而是触发另一种动作。

### 10.5.1 触发 NMI 请求

OCDS 系统不启动监控程序，而是请求 OCDS NMI，详见[章节 10.6.1](#)。

只能通过软件清除此处置位的标志位 **FNMI RR/FNMIRW** - 必须由用户 NMI 服务程序执行该操作。

## 10.6 NMI 请求及控制

OCDS 硬件可产生 NMI 请求并控制一般的 NMI 处理。

### 10.6.1 来自 OCDS 的 NMI 请求

OCDS 产生中断请求从 `ocds_nmi_o` 上输出。该信号送至 XC800 SCU（系统控制单元）内部的中断管理模块，置位 NMI 状态寄存器中的相应标志位。若 NMI 控制寄存器 NMICON 使能 OCDS NMI 中断，则产生非可屏蔽中断请求送至内核。

访问内部 RAM 中给定的地址区间时（读 / 写操作分别配置），OCDS 可激活 NMI 请求。该高级特性可完全由用户程序控制，使得可通过部分 OCDS 来支持一个通用的、用户可访问的功能。

实现该特性的基础是：

- 断点产生模块可识别 IRAM 地址区间的读 / 写访问（参见 [章节 10.4.1.2](#)）
- 断点匹配的同时若 MMCR2.MBCON=1，不启动监控固件，实际上不发生调试。

*注：需要置位 EO.TRAP\_EN 使能产生断点。*

执行 IRAM 访问时，可通过以下步骤配置、触发并处理 OCDS NMI 请求（参见 [章节 10.5.1](#)）：

1. 首先必须设置 MMCR2.MBCON=1
2. 用户程序必须执行以下步骤：
  - a) 置位 MMICR 中的 NMI 使能位 NMIRRE/NMIRWE（分别对应 IRAM 读 / 写访问）
  - b) 将监测区间的低地址写入 HWBP2L/HBWP3L（分别对应 IRAM 读 / 写访问）
  - c) 将监测区间的高地址写入 HWBP2H/HBWP3H（分别对应 IRAM 读 / 写访问）
3. 完成上述置位操作之后，若 MBCON=1，一旦读 / 写访问到配置的 IRAM 区间，OCDS 将触发 NMI 请求。
4. OCDS 产生 NMI 请求后，置位寄存器 MMICR 中的相应标志位 FNMIRR/FNMIRW - 参见 [章节 10.8.1](#) 中的寄存器说明和 [章节 10.5.1](#) 中的功能定义。  
若 NMICON.NMIOCDS=1（该寄存器位于 SCU 中）使能 OCDS NMI 中断，该请求被送至内核，随后跳转到 **XC800 程序存储器的标准单元**，激活 **NMI 中断服务请求**，该服务程序：
  - a) 通过评估 NMISR.FNMIOCDS（SCU 中）和 MMICR.FNMIRR/FNMIRW（OCDS 中）的值，可识别该 NMI 请求由 OCDS 产生以及触发该请求的 IRAM 访问类型。
  - b) 该服务程序结束前必须清除标志位 FNMIRR/FNMIRW，从而可正确响应后面出现的 NMI。
5. 所有设置 MMBPCR 以尝试配置 / 激活断点进行调试的操作都将复位内部 `nmi_r*e/nmi_r*e_l` 标志，从而在进行 IRAM 访问时禁止产生 NMI 请求。

*注意：用户程序将支持 OCDS NMI 的所有软件处理（服务程序）！*

### 10.6.2 OCDS NMI 服务程序处理

IRAM 读 / 写访问触发 OCDS NMI 时，为了使硬件正确工作，需要对 OCDS NMI 服务程序进行特殊的处理以设立正确的 RETI 地址。

建议按照以下程序处理：

```
NMI_OCDS_MAIN: ;OCDS NMI 服务程序
    POP RETADD_H ; 获取存储在 SP 中的返回地址
    POP RETADD_L
    ;/-- 处理 OCDS NMI 的用户程序 ---//
    ;// 用户程序开始
    ...
    ;// 用户程序结束
    MOV A,RETADD_L
    CJNE A,#00H,DEC_LOW_BYTE ; 低位地址字节下溢?
    DEC RETADD_H
DEC_LOW_BYTE:
    DEC RETADD_L ; 返回地址减 1 个地址单元
    PUSH RETADD_L ; 将正确的返回地址回存到 SP 中
    PUSH RETADD_H
    RETI
```

### 10.6.3 OCDS 对 NMI 的控制

控制的目的在于支持正确的 NMI 行为：

- 只要系统仍处于监控模式，在进入监控模式之前已挂起的所有 NMI 请求均不受影响，即：这些请求被安全保存，系统返回用户模式时送至内核。

## 10.7 NMI 模式的优先级高于调试模

当内核处于 NMI 模式（内核接受 NMI 中断请求之后、执行 RETI 指令之前，即：在执行 NMI 中断服务程序的过程中），调试功能被阻滞 / 限制：

- 内核在服务 NMI 时，不响应外部暂停（参见 [章节 10.4.2](#)）。  
若在执行 NMI 中断服务程序的过程中产生外部暂停请求，只有在执行 RETI 指令之后该请求才能被接受。
- 可在 NMI 中断服务程序中插入断点，但之后不能进行单步调试。  
若请求单步调试，中断服务程序将连续执行，只有在执行 RETI 之后才能重新激活监控模式。

CPU 处于 NMI 模式时，硬件断点和软件断点照常处理。

## 10.8 寄存器说明

从映射 SFR 区访问 OCDS SFR，这些（直接寻址）SFR 和相应的地址归纳见 [表 10-5](#)。

**表 10-5 寄存器映射（直接寻址）**

地址	寄存器
F4 <sub>H</sub>	MMICR
F6 <sub>H</sub>	HWBPSR
F7 <sub>H</sub>	HWBPDR
EC <sub>H</sub>	MMWR2

此外还有多个硬件断点寄存器，可通过 **HWBPSR**（寄存器选择）和 **HWBPDR**（数据）间接访问，请参见**表 10-6**和**表 10-7**（详细说明见**章节 10.8**）。

**表 10-6 硬件断点寄存器（间接访问）**

寄存器	说明
HWBP0L	硬件断点 0 低位寄存器
HWBP0H	硬件断点 0 高位寄存器
HWBP1L	硬件断点 1 低位寄存器
HWBP1H	硬件断点 1 高位寄存器
HWBP2L	硬件断点 2 低位寄存器
HWBP2H	硬件断点 2 高位寄存器
HWBP3L	硬件断点 3 低位寄存器
HWBP3H	硬件断点 3 高位寄存器

**表 10-7 HWBPSR [3:0]: 选择硬件断点寄存器**

BPSEL	选择的寄存器	BPSEL	选择的寄存器
0xxx	保留 - 不选择		
1000	HWBP0L	1001	HWBP0H
1010	HWBP1L	1011	HWBP1H
1100	HWBP2L	1101	HWBP2H
1110	HWBP3L	1111	HWBP3H

## 10.8.1 控制和状态寄存器

### MMICR

监控模式中断控制寄存器

(F4<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 1, PAGE: X

7	6	5	4	3	2	1	0
				<b>FNMIW</b>	<b>FNMIW</b>	<b>NMIWE</b>	<b>NMIWE</b>
				rwh	rwh	rw	rw

符号	位	类型	描述
<b>NMIWE</b>	0	rw	读访问 IRAM 触发 NMI (参见 <a href="#">章节 10.6.1</a> ): 0 禁用 1 使能
<b>NMIWE</b>	1	rw	写访问 IRAM 触发 NMI (参见 <a href="#">章节 10.6.1</a> ): 0 禁用 1 使能
<b>FNMIW</b>	2	rwh	IRAM 读访问 NMI 标志
<b>FNMIW</b>	3	rwh	IRAM 写访问 NMI 标志

位 **FNMIW** / **FNMIW**:

- 读 / 写 IRAM 时 OCDS 发出 NMI 请求, 此时硬件置位相应的标志位 (参见[章节 10.5.1](#)和[章节 10.6.1](#))
- 软件只能复位相应的标志位





**HWBP0L**

硬件断点 0 低位寄存器

通过 HWBPDR 设置

 复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
HWBP0L							

符号	位	类型	描述
HWBP0L	[7:0]		比较地址 HWBP0 的低位字节

**HWBP0H**

硬件断点 0 高位寄存器

通过 HWBPDR 设置

 复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
HWBP0H							

符号	位	类型	描述
HWBP0H	[7:0]		比较地址 HWBP0 的高位字节

**HWBP1L**

硬件断点 1 低位寄存器

通过 HWBPDR 设置

 复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
HWBP1L							

符号	位	类型	描述
HWBP1L	[7:0]		比较地址 HWBP1 的低位字节

**HWBP1H**

硬件断点 1 高位寄存器

通过 HWBPDR 设置

 复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
HWBP1H							

符号	位	类型	描述
HWBP1H	[7:0]		比较地址 HWBP1 的高位字节

**HWBP2L**

硬件断点 2 低位寄存器

通过 HWBPDR 设置

 复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
HWBP2L							

符号	位	类型	描述
HWBP2L	[7:0]		比较地址 HWBP2 的低位字节

**HWBP2H**

硬件断点 2 高位寄存器

通过 HWBPDR 设置

 复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
HWBP2H							

符号	位	类型	描述
HWBP2H	[7:0]		比较地址 HWBP2 的高位字节

### HWBP3L

硬件断点 3 低位寄存器      通过 HWBPDR 设置      复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
HWBP3L							

符号	位	类型	描述
HWBP3L	[7:0]		比较地址 HWBP3 的低位字节

### HWBP3H

硬件断点 3 高位寄存器      通过 HWBPDR 设置      复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
HWBP3H							

符号	位	类型	描述
HWBP3H	[7:0]		比较地址 HWBP3 的高位字节

### 10.8.3 监控工作寄存器

若 XC83x 不从 OCDS 模式启动, 且没有外部器件和接口连接 - 即不进行调试操作时, 寄存器 **MMWR2** 可用作通用寄存器。

#### MMWR2

监控工作寄存器 2      (EC<sub>H</sub>)      复位值: 00<sub>H</sub>

RMAP: 1, PAGE: X

7	6	5	4	3	2	1	0
MMWR2							

rW

符号	位	类型	描述
<b>MMWR2</b>	[7:0]	rw	监控程序的工作地址 2

# 11 并行端口

XC83x 的引脚被组织成 P0 - P3 四个并行端口。XC83x 中使用的端口引脚类型归纳见表 11-1。

每个引脚带有一对可分别被使能或禁止的内部上拉 / 下拉器件。P0、P1 和 P3 口为双向口，可用作通用输入 / 输出（GPIO）或片上外设的输入 / 输出（复用输入输出功能）。引脚设置为输出时，可选择漏极开路模式。P1 口是大电流双向端口，可用于电机和 LED 应用。P2 口是单向输入口，可用作通用输入、片上外设的输入以及 ADC 的模拟输入。

**表 11-1 XC83x 中的端口类型**

端口引脚	端口类型
P0.1 - P0.7	标准双向端口
P1.0 - P1.3	大电流双向 A 类端口（HCPADA） <sup>1)</sup>
P1.4 - P1.5	大电流双向 B 类端口（HCPADB） <sup>2)</sup>
P2.0 - P2.7	输入端口
P3.0 - P3.2	标准双向端口

<sup>1)</sup> HCPADA 用于直接驱动电机和 LED。

<sup>2)</sup> HCPADB 仅用于 LED 应用，不用于直接驱动电机。

## 大电流双向 A 类端口（HCPADA）特性：

- 引脚方向可配置
- 上拉 / 下拉器件可配置
- 漏极开路模式可配置
- 通过数字输入输出传送数据（通用输入 / 输出）
- 片上外设的输入 / 输出（复用输入输出功能）
- 在省电模式下输入和输出驱动器禁用
- 支持单个 2 相步进电机（即：使用 4 个 HCPADA 端口引脚）
- 支持 LED 驱动
- 翻转速度控制
- 过电流保护使能控制
- 过电流检测指示 & 关闭
- 发生过电流事件时可触发 TRAP 功能

## 大电流双向 B 类端口（HCPADB）特性：

- 引脚方向可配置
- 上拉 / 下拉器件可配置
- 漏极开路模式可配置
- 通过数字输入输出传送数据（通用输入 / 输出）

- 片上外设的输入 / 输出 (复用输入输出功能)
- 在省电模式下输入和输出驱动器禁用
- 支持 LED 驱动

#### 标准双向口特性:

- 引脚方向可配置
- 上拉 / 下拉器件可配置
- 漏极开路模式可配置
- 通过数字输入输出传送数据 (通用输入 / 输出)
- 片上外设的输入 / 输出 (复用输入输出功能)
- 在省电模式下输入和输出驱动器禁用

#### 输入口特性:

- 输入驱动器可配置
- 上拉 / 下拉器件可配置
- 通过数字输入接收数据 (通用输入)
- 片上外设的输入 (复用输入功能)
- ADC 模块的模拟输入
- 在省电模式下输入驱动器禁用

### 11.1 基本端口操作

XC83x 双向口引脚的结构框图如图 11-1 所示。每个引脚通过一组控制位和数据位来配置, 从而引脚使用具有很大的灵活性。通过设置控制寄存器, 每个引脚可分别被配置成输入或输出; 还可配置成带有或者不带内部上拉 / 下拉器件的漏极开路引脚。

标准双向口的输入和输出驱动器在正常工作模式下使能, 掉电模式下禁用。不过, 在掉电模式下, 用作掉电唤醒的端口引脚的驱动器使能工作。

输出模式下, 输出驱动器将复用器送出的值驱动至端口引脚。在输出模式下, 可通过寄存器 `Px_OD` ( $x = 0, 1$ ) 将每个引脚切换至漏极开路模式或正常模式 (推挽模式)。

输出驱动器的前级输出复用器使输出可用作多种功能。若引脚用作通用输出, 软件控制复用器选通数据寄存器 `Px_DATA`, 软件可对 `Px_DATA` 中的各位位置位或清零, 从而直接影响引脚的状态。若引脚用作片上外设的输出, 输出复用器会选通复用功能输出线 (`AltDataOut`), 将其送至输出驱动电路。由寄存器 `P0_ALTSEL0`、`P0_ALTSEL1`、`P0_ALTSEL2`、`P1_ALTSEL0` 和 `P1_ALTSEL1` 进行复用功能选择。

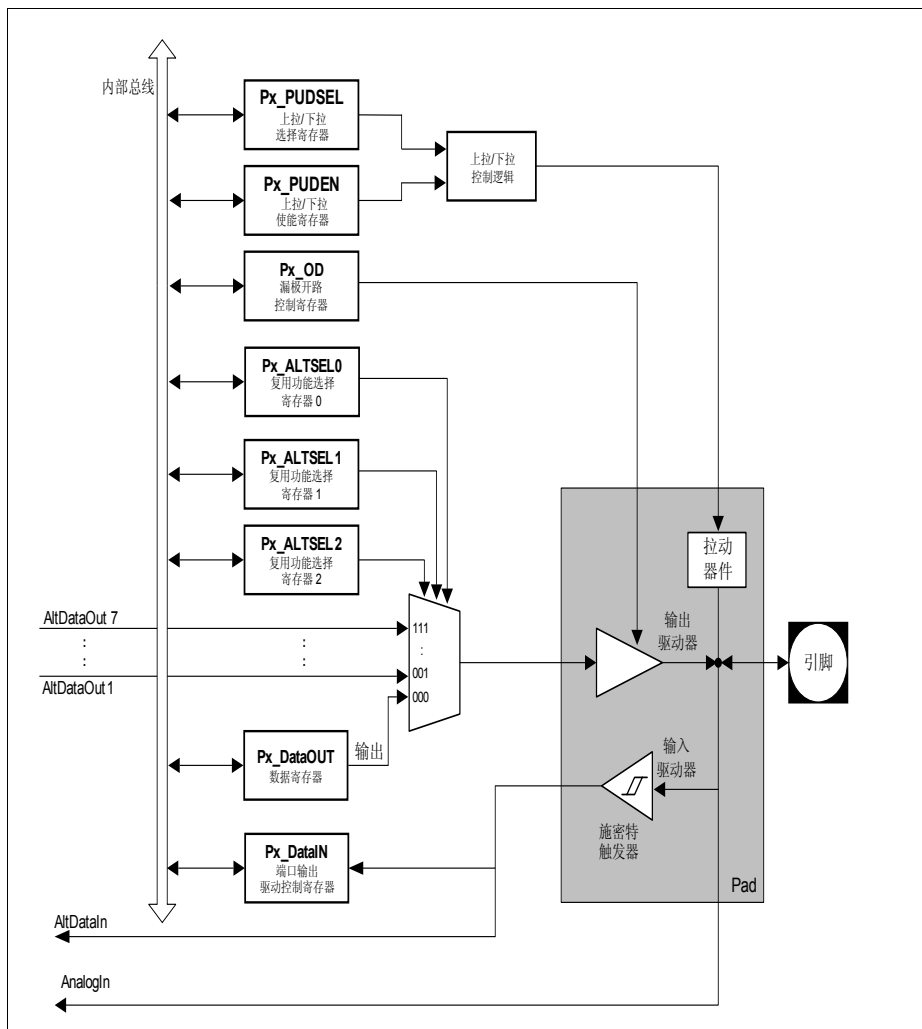


图 11-1 标准双向口和大电流双向 B 类端口的内部结构

将引脚配置为输入模式（复位后的缺省模式）时，必须通过以下步骤将输出驱动器切换至高阻：

- 通过寄存器 Px\_DATAOUT 向端口引脚写 "1"，将输出驱动器置为高阻
- 通过寄存器 Px\_OD 使能漏极开路功能
- 通过寄存器 Px\_ALTSELx 选择用作 GPIO 功能



在输入模式下，若复用输出线确保保持高电平，则不必设置用作 GPIO 功能。

引脚上的实际电压值由施密特触发器译成逻辑 0 或 1，可从寄存器 **Px\_DATAIN** 中读取。每个引脚还可编程设置激活内部的弱上拉或下拉器件。寄存器 **Px\_PUDEN** 使能或禁止拉动器件，寄存器 **Px\_PUDSEL** 选择激活上拉或下拉器件。

以下代码给出如何将所有 P0 口引脚配置为输入模式：

```
ANL    SYSCON0, #0xFE
MOV    PORT_PAGE, #0x00
MOV    P0_DATAOUT, #0xFF      ; 置位 P0_DATAOUT
MOV    PORT_PAGE, #0x03
MOV    P0_OD, #0xFF          ; 使能漏极开路
MOV    PORT_PAGE, #0x02
MOV    P0_ALTSEL0, #0x00      ; 复用功能选择 GPIO
MOV    P0_ALTSEL1, #0x00      ; Px_ALTSELx 位于 PORT_PAGE = 0x02 中
MOV    P0_ALTSEL2, #0x00
```

大电流双向 A 类端口引脚的结构框图如图 11-2 所示。除了具有标准双向端口引脚的特性之外，该类引脚还具有直接驱动电机的特性。可通过 **Px\_OCPEN** 控制过电流保护。使能过电流保护功能时，通过禁用端口内部的输入和输出驱动器，可为引脚提供过载自我保护。从过电流状态下恢复时，状态标志 **Px\_OCF** 需要由软件清零。一旦该标志被清零，输入和输出驱动器将被使能。发生过电流时，寄存器 **Px\_OCD** 中的位 **Px\_OCF** 置 1<sub>B</sub>。若 **Px\_OCD** 中的位 **Px\_OCEN** 置 1<sub>B</sub>，过电流事件可触发中断。此外，可通过寄存器 **Px\_SLEW** 选择端口的翻转速度。

若设置 **CCU6\_PISEL0L.ISTRP[7:6]** 为 01<sub>B</sub> 或 10<sub>B</sub>，发生过电流事件时，可触发 **CCU6** 模块的 **TRAP** 功能。

大电流双向 B 类端口没有过电流保护和检测功能，其引脚的结构框图 and 标准双向口类似，如图 11-1 所示。该类端口具有高灌电流以支持 **LED** 应用。

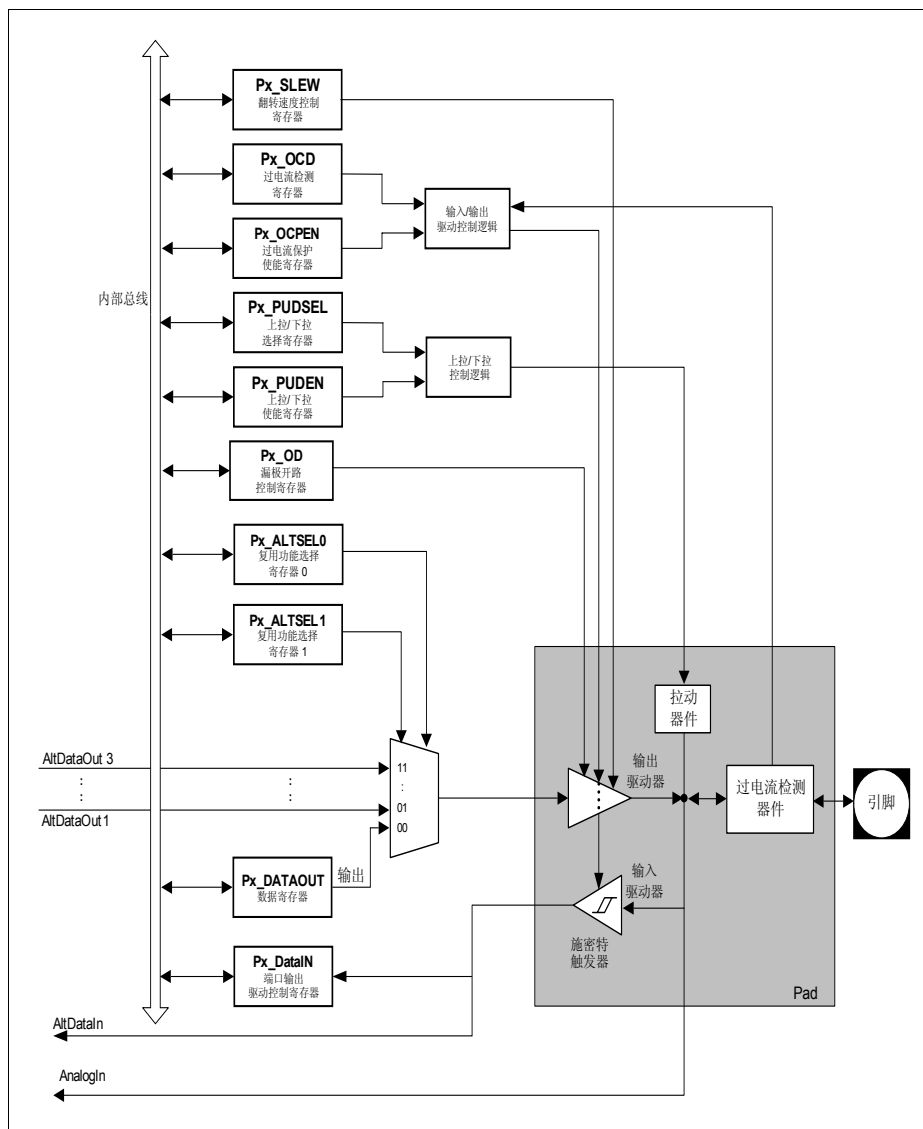


图 11-2 大电流双向 A 类端口的基本结构

## 并行端口

单向输入口引脚结构如图 11-3 所示。每一个 P2 口引脚只能工作在输入模式。通过寄存器 P2\_EN 禁止 / 使能输入驱动器。输入驱动器被使能时，引脚上的实际电压值由施密特触发器译成逻辑 0 或 1，可从寄存器 P2\_DATAIN 读出。每个引脚还可设置为激活内部的弱上拉或下拉器件。寄存器 P2\_PUDEN 使能或禁止拉动器件，寄存器 P2\_PUDSEL 选择激活上拉还是下拉器件。模拟输入（AnalogIn）不经过数字电路和施密特触发器，直接进入 ADC 的输入通道。

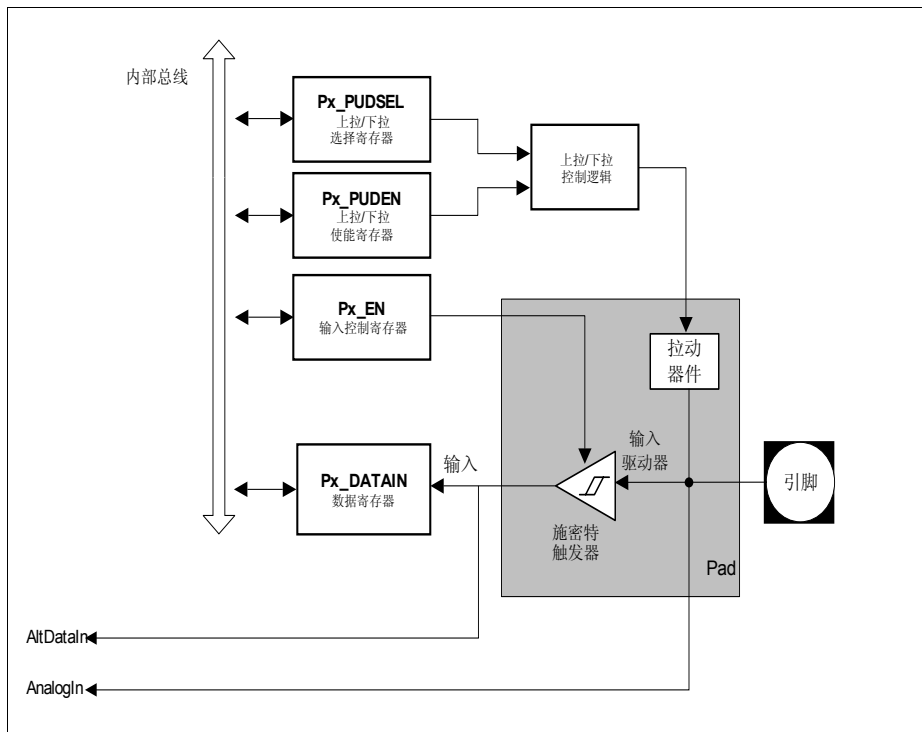


图 11-3 单向输入口的基本机构

### 11.1.1 通用寄存器说明

本节将对 XC83x 中的端口 SFR 进行详细说明。

#### 11.1.1.1 寄存器映射

端口 SFR 位于标准存储区（RMAP = 0），由四页构成。PORT\_PAGE 寄存器位于地址单元 8E<sub>H</sub>，包含页信息和分页控制信息。

端口 SFR 的地址归纳见表 11-2。

表 11-2 页 0-3 的 SFR 地址列表

地址	页 0	页 1	页 2	页 3
80 <sub>H</sub>	P0_DATAOUT	P0_PUDSEL	P0_ALTSEL0	P0_OD
86 <sub>H</sub>	P0_DATAIN	P0_PUDEN	P0_ALTSEL1	
85 <sub>H</sub>			P0_ALTSEL2	
90 <sub>H</sub>	P1_DATAOUT	P1_PUDSEL	P1_ALTSEL0	P1_OD
91 <sub>H</sub>	P1_DATAIN	P1_PUDEN	P1_ALTSEL1	
92 <sub>H</sub>	P1_OCD	P1_OCPEN		P1_SLEW
93 <sub>H</sub>		P2_PUDSEL		
94 <sub>H</sub>	P2_DATAIN	P2_PUDEN		P2_EN
C8 <sub>H</sub>	P3_DATAOUT	P3_PUDSEL	P3_ALTSEL0	P3_OD
C9 <sub>H</sub>	P3_DATAIN	P3_PUDEN	P3_ALTSEL1	

#### PORT\_PAGE

端口分页寄存器

(8E<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
OP		STNR		0	PAGE		
w		w		r	rwh		

符号	位	类型	描述
PAGE	[2:0]	rwh	<p>页信息</p> <p>写入时，该值表示新页地址。</p> <p>读出时，该值表示当前的有效页 = addr [y:x+1]。</p>

符号	位	类型	描述
<b>STNR</b>	[5:4]	w	<b>保存编号</b> 该编号指示在哪个保存位域上执行由 OP 确定的操作。 若 OP = 10 <sub>B</sub> , PAGE 的内容在被新值覆盖之前保存在 PORT_STx 中。 若 OP = 11 <sub>B</sub> , PAGE 的内容被 PORT_STx 的内容覆盖。写入 PAGE 的值被忽略。 00 选择 PORT_ST0 01 选择 PORT_ST1 10 选择 PORT_ST2 11 选择 PORT_ST3
<b>OP</b>	[7:6]	w	<b>操作</b> 0X 手动保存页模式, STNR 的值被忽略, PAGE 被直接写入。 10 带有自动页保存的新页设置。当前写入 PAGE 中的内容被保存的同时, 上次写入 PAGE 中的内容被保存在 STNR 指定的位域 PORT_STx 中。 11 自动恢复页操作。写入 PAGE 的内容被忽略, PAGE 的内容由 STNR 指定的位域 PORT_STx 中的值覆盖。
<b>0</b>	3	r	<b>保留</b> 读操作返回 0, 应写入 0。

### 11.1.1.2 寄存器概览

每个并行口的控制位和数据位存放在一组 8 位寄存器中。具有相同含义和功能的位放在同一个寄存器中。这些寄存器将端口设置为通用 I/O 口或复用功能输入 / 输出口。

P0 和 P2 口仅实现了表 11-3 列出的部分寄存器。每个端口寄存器的具体描述可参见章节 11.2 至章节 11.5。本节仅对不同端口的寄存器予以概述。

表 11-3 端口寄存器

寄存器缩写名	寄存器全名	具体说明参见
Px_DATAOUT	端口 x 数据输出寄存器	页 11-9
Px_DATAIN	端口 x 数据输入寄存器	页 11-10
Px_OD	端口 x 漏极开路控制寄存器	页 11-10
Px_PUDSEL	端口 x 上拉 / 下拉选择寄存器	页 11-11

表 11-3 端口寄存器

寄存器缩写名	寄存器全名	具体说明参见
Px_PUDEN	端口 x 上拉 / 下拉使能寄存器	页 11-11
Px_ALTSEL0	端口 x 复用功能选择寄存器 0	页 11-12
Px_ALTSEL1	端口 x 复用功能选择寄存器 1	页 11-12
Px_ALTSEL2	端口 x 复用功能选择寄存器 2	页 11-12
Px_OCD	端口 x 过电流检测寄存器	页 11-13
Px_OCPEN	端口 x 过电流保护使能寄存器	页 11-14
Px_SLEW	端口 x 翻转速度控制寄存器	页 11-15
Px_EN	端口 x 输入控制寄存器	页 11-15

注：并非每个端口均实现了上表列出的所有寄存器。

## 端口数据输出寄存器

若引脚用作通用输出口，输出数据写入端口 x 的寄存器 Px\_DATAOUT 中。

读取 Px\_DATAOUT 将返回寄存器的值，而不是 Px\_DATAOUT 引脚上的状态。

## Px\_DATAOUT

### 端口 x 数据输出寄存器

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rw	<b>端口 x 引脚 n 的数据值</b> 0 端口 x 引脚 n 的数据值 = 0 1 端口 x 引脚 n 的数据值 = 1 （缺省值）

## 端口数据输入寄存器

若引脚用作通用输入口，可从寄存器 **Px\_DATAIN** 中读取引脚上的值。数据寄存器 **Px\_DATAIN** 中始终保存端口引脚的锁存值，即使它被分配用作输出引脚。

### **Px\_DATAIN**

#### 端口 **x** 数据输入寄存器

7	6	5	4	3	2	1	0
<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

符号	位	类型	描述
<b>Pn</b> <b>(n = 0 - 7)</b>	n	rh	<b>端口 <b>x</b> 引脚 <b>n</b> 的数据值</b> 0 端口 <b>x</b> 引脚 <b>n</b> 的数据值 = 0 1 端口 <b>x</b> 引脚 <b>n</b> 的数据值 = 1

## 漏极开路控制寄存器

工作在输出模式的每个引脚均可切换到漏极开路模式。如果用 **1** 驱动，不会激活驱动器，引脚的输出状态取决于内部上拉 / 下拉器件的设置；如果用 **0** 驱动，驱动器的下拉晶体管导通。

漏极开路模式由寄存器 **Px\_OD** 控制。

### **Px\_OD**

#### 端口 **x** 漏极开路控制寄存器

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> <b>(n = 0 - 7)</b>	n	rw	<b>端口 <b>x</b> 引脚 <b>n</b> 漏极开路模式选择</b> 0 正常模式；0 和 1 可有效输出 1 漏极开路模式；只有 0 可有效输出（缺省选择）

## 上拉 / 下拉器件寄存器

端口引脚可选择使用内部上拉或下拉器件，从而具备以下输入特性：

- 三态
- 带有弱上拉的高阻态
- 带有弱下拉的高阻态

和以下输出特性：

- 推挽（选择上拉 / 下拉）
- 带有内部上拉的漏极开路输出
- 带有外部上拉的漏极开路输出

上拉 / 下拉器件可由寄存器 **Px\_PUDSEL** 和 **Px\_PUDEN** 控制。寄存器 **Px\_PUDSEL** 选择上拉 / 下拉类型；寄存器 **Px\_PUDEN** 使能或禁止上拉 / 下拉器件。可分别为每个引脚选择上拉 / 下拉器件。

### **Px\_PUDSEL**

端口 **x** 上拉 / 下拉选择寄存器

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rw	端口 <b>x</b> 引脚 <b>n</b> 上拉 / 下拉选择 0 选择下拉器件 1 选择上拉器件

### **Px\_PUDEN**

端口 **x** 上拉 / 下拉使能寄存器

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rw	端口 <b>x</b> 引脚 <b>n</b> 上拉 / 下拉使能 0 禁用上拉或下拉器件 1 使能上拉或下拉器件



## 复用输入功能

一个端口引脚上的复用输入功能的数目不限。每个 I/O 引脚的端口控制逻辑通过寄存器提供不同输入通路的数字输入值。有关端口输入选择寄存器的具体说明可参见相应外设章节的内容。

## 复用输出功能

由输出复用器选择复用输出功能（最多可选择 8 条输出线）。输出复用器可由以下寄存器控制：

- 寄存器 Px\_ALTSEL0
- 寄存器 Px\_ALTSEL1
- 寄存器 Px\_ALTSEL2

由寄存器 Px\_ALTSEL0、Px\_ALTSEL1 和 Px\_ALTSEL2 选择复用输出功能。对于不超过 4 种复用功能的端口，通过 Px\_ALTSEL2 进行设置。

*注：应将 Px\_ALTSEL0.Pn、Px\_ALTSEL1.Pn 和 Px\_ALTSEL2.Pn 设置成已实现的复用输出功能。*

### Px\_ALTSELn (n = 0, 1, 2)

#### 端口 x 复用功能选择寄存器

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rw	<b>引脚输出功能</b> 配置 Px_ALTSEL0.Pn、Px_ALTSEL1.Pn 和 Px_ALTSEL2.Pn 选择引脚用作 GPIO 或复用功能： 000 GPIO 功能 001 复用输出功能 1 010 复用输出功能 2 011 复用输出功能 3 100 复用输出功能 4 101 复用输出功能 5 110 复用输出功能 6 111 复用输出功能 7

### 过电流检测寄存器

每个端口引脚均配有过电流检测电路，检测到发生过电流时，该电路将置位过电流检测寄存器中相应的 Px\_OCF。若 Px\_OCEN 置 1，同时会产生中断。

### Px\_OCD 端口 x 过电流检测寄存器

7	6	5	4	3	2	1	0
<b>P3_OCEN</b>	<b>P2_OCEN</b>	<b>P1_OCEN</b>	<b>P0_OCEN</b>	<b>P3_OCF</b>	<b>P2_OCF</b>	<b>P1_OCF</b>	<b>P0_OCF</b>
rw	rw	rw	rw	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>Pn_OCF</b> (n = 0 - 3)	n	rwh	大电流端口 x 引脚 n 过电流检测 0 未产生过电流。向该位写 0 将清除锁存值。 1 检测到过电流。
<b>Pn_OCEN</b> (n = 0 - 3)	n	rw	大电流端口 x 引脚 n 过电流中断使能 0 禁止产生过电流中断。 1 使能产生过电流中断。

### 过电流保护使能寄存器

若使能过电流保护寄存器，当过电流检测寄存器的值被锁存时（即表明已产生过电流），大电流端口的输入和输出驱动器将被自动禁用。

**Px\_OCPEN**  
端口 x 过电流保护使能寄存器

7	6	5	4	3	2	1	0
0				P3	P2	P1	P0
r				rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> <b>(n = 0 - 3)</b>	n	rw	大电流端口 x 引脚 n 过电流保护使能 0 禁用过电流保护。 1 使能过电流保护。

### 翻转速度控制寄存器

翻转寄存器控制每个大电流端口的翻转速度。

#### Px\_SLEW

端口 x 翻转速度控制寄存器

7	6	5	4	3	2	1	0
		0		P3	P2	P1	P0
		r		rW	rW	rW	rW

符号	位	类型	描述
Pn (n = 0 - 3)	n	rW	大电流端口 x 引脚 n 翻转速度控制 0 选择陡变沿。 1 选择缓变沿。

### 输入控制寄存器

输入控制寄存器控制每个端口引脚的输入驱动器。该寄存器用于 P2 口。

#### Px\_EN

端口 x 输入控制寄存器

7	6	5	4	3	2	1	0
P7	P6	P5	P4	P3	P2	P1	P0
rW	rW	rW	rW	rW	rW	rW	rW

符号	位	类型	描述
Pn (n = 0 - 7)	n	rW	端口 x 引脚 n 的输入驱动器控制 0 使能输入驱动器 1 禁用输入驱动器

## 11.2 P0 口

P0 口是一个 8 位通用双向口，其每个引脚均采用标准双向 pad。P0 口寄存器归纳见表 11-4。

**表 11-4 P0 口寄存器**

寄存器缩写名	寄存器全名
P0_DATAIN	P0 口数据输入寄存器
P0_DATAOUT	P0 口数据输出寄存器
P0_OD	P0 口漏极开路控制寄存器
P0_PUDSEL	P0 口上拉 / 下拉选择寄存器
P0_PUDEN	P0 口上拉 / 下拉使能寄存器
P0_ALTSEL0	P0 口复用功能选择寄存器 0
P0_ALTSEL1	P0 口复用功能选择寄存器 1
P0_ALTSEL2	P0 口复用功能选择寄存器 2

### 11.2.1 功能

P0 口输入和输出功能归纳见表 11-5。

**表 11-5 P0 口 I/O 功能**

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P0.0	输入	GPI	P0_DATAIN.P0	
		ALT1	T2_0	定时器 T2
		ALT2	T13HR_1	CCU6
		ALT3	MTSR_2	SSC
		ALT4	MRST_3	SSC
		ALT5	T12HR_0	CCU6
		ALT6	CCPOS0_0	CCU6
		ALT7	TSIN0	LEDTSCU
	输出	GPO	P0_DATAOUT.P0	
		ALT1	LINE0/TSIN0	LEDTSCU
		ALT2	MTSR_2	SSC
		ALT3	COUT61_1	CCU6

**表 11-5 P0 口 I/O 功能**

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P0.1	输入	GPI	P0_DATAIN.P1	
		ALT1	T0_0	定时器 T0
		ALT2	CC61_1	CCU6
		ALT3	MTSR_3	SSC
		ALT4	MRST_2	SSC
		ALT5	T13HR_0	CCU6
		ALT6	CCPOS1_0	CCU6
		ALT7	TSIN1	LEDTSCU
	输出	GPO	P0_DATAOUT.P1	
		ALT1	LINE1/TSIN1	LEDTSCU
		ALT2	MRST_2	SSC
		ALT3	CC61_1	CCU6
P0.2	输入	GPI	P0_DATAIN.P2	
		ALT1	T1_0	定时器 T1
		ALT2	CC62_1	CCU6
		ALT3	SCL_1	IIC
		ALT4	–	
		ALT5	–	
		ALT6	CCPOS2_0	CCU6
		ALT7	TSIN2	LEDTSCU
	输出	GPO	P0_DATAOUT.P2	
		ALT1	LINE2/TSIN2	LEDTSCU
		ALT2	SCL_1	IIC
		ALT3	CC62_1	CCU6

表 11-5 P0 口 I/O 功能

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P0.3	输入	GPI	P0_DATAIN.P3	
		ALT1	–	
		ALT2	CC60_1	CCU6
		ALT3	SDA_1	IIC
		ALT4	–	
		ALT5	–	
		ALT6	CTRAP_0	CCU6
		ALT7	TSIN3	LEDTSCU
	输出	GPO	P0_DATAOUT.P3	
		ALT1	LINE3/TSIN3	SSC
		ALT2	SDA_1	IIC
		ALT3	CC60_1	CCU6
P0.4	输入	GPI	P0_DATAIN.P4	
		ALT1	T2EX_1	定时器 T2
		ALT2	SCL_0	IIC
		ALT3	SCK_0	SSC
		ALT4	–	
		ALT5	EXINT1_0	SCU
		ALT6	CTRAP_1	CCU6
		ALT7	TSIN4	LEDTSCU
	输出	GPO	P0_DATAOUT.P4	
		ALT1	LINE4/TSIN4	LEDTSCU
		ALT2	SCK_0	SSC
		ALT3	EXF2_0	定时器 T2
		ALT4	SCL_0	IIC
		ALT5	COL0_1	LEDTSCU
		ALT6	COL3_2	LEDTSCU
		ALT7	COLA_4	LEDTSCU

**表 11-5 P0 口 I/O 功能**

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P0.5	输入	GPI	P0_DATAIN.P5	
		ALT1	RXD_0	UART
		ALT2	MTSR_0	SSC
		ALT3	MRST_1	SSC
		ALT4	EXINT0_0	SCU
		ALT5	—	
		ALT6	TSIN5	LEDTSCU
	输出	GPO	P0_DATAOUT.P5	
		ALT1	LINE5/TSIN5	LEDTSCU
		ALT2	MTSR_0	SSC
		ALT3	COUT62_1	CCU6
		ALT4	TXD_4	UART
		ALT5	COL1_1	LEDTSCU
		ALT6	EXF2_3	定时器 T2



**表 11-5 P0 口 I/O 功能**

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P0.6	输入	GPI	P0_DATAIN.P6	
		ALT1	RXD_1	UART
		ALT2	SDA_0	IIC
		ALT3	MTSR_1	SSC
		ALT4	MRST_0	SSC
		ALT5	EXINT0_1	SCU
		ALT6	T2EX_0	定时器 T2
		ALT7	TSIN6	LEDTSCU
		其它	SPD_1	SPD
	输出	GPO	P0_DATAOUT.P6	
		ALT1	LINE6/TSIN6	LEDTSCU
		ALT2	MRST_0	SSC
		ALT3	TXD_0	UART
		ALT4	SDA_0	IIC
		ALT5	COL2_1	LEDTSCU
		ALT6	COLA_2	LEDTSCU
		其它	SPD_1	SPD

**表 11-5 P0 口 I/O 功能**

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P0.7	输入	GPI	P0_DATAIN.P7	
		ALT1	–	
		ALT2	SCL_3	IIC
		ALT3	–	
		ALT4	–	
		ALT5	–	
		ALT6	–	
		ALT7	TSIN7	LED/TS
	输出	GPO	P0_DATAOUT.P7	
		ALT1	LINE7/TSIN7	LED/TS
		ALT2	TXD_5	UART
		ALT3	COUT63_0	CCU6
		ALT4	SCL_3	IIC
		ALT5	COL3_1	LED/TS
		ALT6	COLA_3	LED/TS

## 11.2.2 寄存器说明

### P0\_DATAIN

P0 口数据输入寄存器

 (86<sub>H</sub>)

 复位值: XX<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
<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

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rh	<b>P0 口引脚 n 的数据值</b> 0 P0 口引脚 n 的数据值 = 0 1 P0 口引脚 n 的数据值 = 1

### P0\_DATAOUT

P0 口数据输出寄存器

 (80<sub>H</sub>)

 复位值: FF<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rw	<b>P0 口引脚 n 的数据值</b> 0 P0 口引脚 n 的数据值 = 0 1 P0 口引脚 n 的数据值 = 1

### P0\_OD

P0 口漏极开路控制寄存器

 (80<sub>H</sub>)

 复位值: FF<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> <b>(n = 0 - 7)</b>	n	rw	<b>P0 口引脚 n 漏极开路模式选择</b> 0 正常模式；0 和 1 可被有效输出 1 漏极开路模式；只有 0 可被有效输出（缺省值）

**P0\_PUDSEL**
**P0 口上拉 / 下拉选择寄存器**
**(80<sub>H</sub>)**
**复位值：EF<sub>H</sub>**
**RMAP: 0, PAGE: 1**

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> <b>(n = 0 - 7)</b>	n	rw	<b>P0 口引脚 n 上拉 / 下拉选择</b> 0 选择下拉器件 1 选择上拉器件（缺省值）

**P0\_PUDEN**
**P0 口上拉 / 下拉使能寄存器**
**(86<sub>H</sub>)**
**复位值：50<sub>H</sub>**
**RMAP: 0, PAGE: 1**

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> <b>(n = 0 - 7)</b>	n	rw	<b>P0 口引脚 n 上拉 / 下拉使能</b> 0 禁用上拉或下拉器件 1 使能上拉或下拉器件

**注：** P0.4 缺省使能下拉器件。在启动程序中，由固件切换到使能上拉器件。

并行端口

**P0\_ALTSEL0**
**P0 口复用功能选择寄存器 0**
**(80<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rw	见表 11-6

**P0\_ALTSEL1**
**P0 口复用功能选择寄存器 1**
**(86<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rw	见表 11-6

**P0\_ALTSEL2**
**P0 口复用功能选择寄存器 2**
**(85<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>			<b>0</b>	
rw	rw	rw	rw			r	

符号	位	类型	描述
<b>Pn</b> <b>(n = 4 - 7)</b>	n	rw	见表 11-6
<b>0</b>	[3:0]	r	保留 读取返回 0；应写入 0。

**表 11-6      位 P0\_ALTSEL2.Pn、P0\_ALTSEL1.Pn 和 P0\_ALTSEL0.Pn 的功能**

<b>P0_ALTSEL2.Pn</b>	<b>P0_ALTSEL1.Pn</b>	<b>P0_ALTSEL0.Pn</b>	<b>功能</b>
0	0	0	GPIO（缺省）
0	0	1	选择复用功能 1
0	1	0	选择复用功能 2
0	1	1	选择复用功能 3
1	0	0	选择复用功能 4
1	0	1	选择复用功能 5
1	1	0	选择复用功能 6
1	1	1	选择复用功能 7

### 11.3 P1 口

P1 口是一个通用双向口。在 XC83x 中，P1.0、P1.1、P1.2 和 P1.3 采用大电流双向 A 类端口；P1.4 和 P1.5 采用大电流双向 B 类端口。P1 口寄存器归纳见[表 11-7](#)。

**表 11-7 P1 口寄存器**

寄存器缩写名	寄存器全名
P1_DATAOUT	P1 口数据输出寄存器
P1_DATAIN	P1 口数据输入寄存器
P1_OD	P1 口漏极开路控制寄存器
P1_PUDSEL	P1 口上拉 / 下拉选择寄存器
P1_PUDEN	P1 口上拉 / 下拉使能寄存器
P1_ALTSEL0	P1 口复用功能选择寄存器 0
P1_ALTSEL1	P1 口复用功能选择寄存器 1
P1_OCD	P1 口过流检测寄存器
P1_OCPEN	P1 口过流保护使能寄存器
P1_SLEW	P1 口翻转速度控制寄存器

### 11.3.1 功能

P1 口输入和输出功能归纳见[表 11-8](#)。

**表 11-8 P1 口 I/O 功能**

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P1.0	输入	GPI	P1_DATAIN.P0	
		ALT 1	RXD_2	UART
		ALT 2	T2EX_2	定时器 T2
		ALT 3	–	
		ALT 4	–	
		ALT 5	EXINT0_2	SCU
		ALT 6	–	
		其它	SPD_2	SPD
	输出	GPO	P1_DATAOUT.P0	
		ALT 1	COL0_0	LEDTSCU
		ALT 2	COU60_0	CCU6
		ALT 3	TXD_1	UART
		其它	SPD_2	SPD
P1.1	输入	GPI	P1_DATAIN.P1	
		ALT 1	–	
		ALT 2	–	
		ALT 3	–	
		ALT 4	–	
		ALT 5	–	
		ALT 6	CC60_0	CCU6
	输出	GPO	P1_DATAOUT.P1	
		ALT 1	COL1_0	LEDTSCU
		ALT 2	CC60_0	CCU6
		ALT 3	TXD_2	UART



表 11-8 P1 口 I/O 功能

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P1.2	输入	GPI	P1_DATAIN.P2	
		ALT 1	–	
		ALT 2	–	
		ALT 3	–	
		ALT 4	–	
		ALT 5	EXINT4	SCU
		ALT 6	–	
	输出	GPO	P1_DATAOUT.P2	
		ALT 1	COL2_0	LEDTSU
		ALT 2	COUT61_0	CCU6
		ALT 3	COUT63_1	CCU6
P1.3	输入	GPI	P1_DATAIN.P3	
		ALT 1	–	
		ALT 2	–	
		ALT 3	–	
		ALT 4	–	
		ALT 5	–	
		ALT 6	CC61_0	CCU6
	输出	GPO	P1_DATAOUT.P3	
		ALT1	COL3_0	LEDTSU
		ALT2	CC61_0	CCU6
		ALT3	EXF2_2	定时器 T2

表 11-8 P1 口 I/O 功能

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P1.4	输入	GPI	P1_DATAIN.P4	
		ALT 1	–	
		ALT 2	–	
		ALT 3	–	
		ALT 4	–	
		ALT 5	EXINT5	SCU
		ALT 6	–	
	输出	GPO	P1_DATAOUT.P4	
		ALT1	COL4	LEDTSCU
		ALT2	COUT62_0	CCU6
		ALT3	COUT63_2	CCU6
P1.5	输入	GPI	P1_DATAIN.P5	
		ALT 1	–	
		ALT 2	–	
		ALT 3	–	
		ALT 4	–	
		ALT 5	–	
		ALT 6	CC62_0	CCU6
	输出	GPO	P1_DATAOUT.P5	
		ALT1	COL5	LEDTSCU
		ALT2	CC62_0	CCU6
		ALT3	COLA_1	LEDTSCU

### 11.3.2 寄存器说明

#### P1\_DATAIN

P1 口数据输入寄存器

(91<sub>H</sub>)

复位值: XX<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	P5	P4	P3	P2	P1	P0	
r	rh	rh	rh	rh	rh	rh	rh

符号	位	类型	描述
Pn (n = 0 - 5)	n	rh	P1 口引脚 n 的数据值 0 P1 口引脚 n 的数据值 = 0 1 P1 口引脚 n 的数据值 = 1
0	[7:6]	r	保留 读取返回 0；应写入 0。

#### P1\_DATAOUT

P1 口数据输出寄存器

(90<sub>H</sub>)

复位值: 3F<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	P5	P4	P3	P2	P1	P0	
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
Pn (n = 0 - 5)	n	rw	P1 口引脚 n 的数据值 0 P1 口引脚 n 的数据值 = 0 1 P1 口引脚 n 的数据值 = 1
0	[7:6]	r	保留 读取返回 0；应写入 0。

并行端口

# P1\_OD

P1 口漏极开路控制寄存器

(90<sub>H</sub>)

复位值：3F<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	P5	P4	P3	P2	P1	P0	
r	rW	rW	rW	rW	rW	rW	rW

符号	位	类型	描述
Pn (n = 0 - 5)	n	rW	P1 口引脚 n 漏极开路模式选择 0 正常模式；0 和 1 可被有效输出 1 漏极开路模式；只有 0 可被有效输出（缺省值）
0	[7:6]	r	保留 读取返回 0；应写入 0。

# P1\_PUDSEL

P1 口上拉 / 下拉选择寄存器

(90<sub>H</sub>)

复位值：3F<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	P5	P4	P3	P2	P1	P0	
r	rW	rW	rW	rW	rW	rW	rW

符号	位	类型	描述
Pn (n = 0 - 5)	n	rW	P1 口引脚 n 上拉 / 下拉选择 0 选择下拉器件 1 选择上拉器件（缺省值）
0	[7:6]	r	保留 读取返回 0；应写入 0。

**P1\_PUDEN**
**P1 口上拉 / 下拉使能寄存器**
**(91<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: 1**

7	6	5	4	3	2	1	0
0		P5	P4	P3	P2	P1	P0
r		rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>P<sub>n</sub></b> <b>(n = 0 - 5)</b>	n	rw	<b>P1 口引脚 n 上拉 / 下拉使能</b> 0 禁用上拉或下拉器件 (缺省值) 1 使能上拉或下拉器件
<b>0</b>	[7:6]	r	保留 读取返回 0；应写入 0。

**P1\_ALTSELx (x = 0 - 1)**
**P1 口复用功能选择寄存器**
**(90<sub>H</sub> + x)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
0		P5	P4	P3	P2	P1	P0
r		rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>P<sub>n</sub></b> <b>(n = 0 - 5)</b>	n	rw	见 <b>表 11-9</b>
<b>0</b>	[7:6]	r	保留 读取返回 0；应写入 0。

**表 11-9 位 P1\_ALTSEL0.P<sub>n</sub> 和 P1\_ALTSEL1.P<sub>n</sub> 的功能**

P1_ALTSEL1.P <sub>n</sub>	P1_ALTSEL0.P <sub>n</sub>	功能
0	0	GPIO (缺省)
0	1	选择复用功能 1

表 11-9 位 P1\_ALTSEL0.Pn 和 P1\_ALTSEL1.Pn 的功能

P1_ALTSEL1.Pn	P1_ALTSEL0.Pn	功能
1	0	选择复用功能 2
1	1	选择复用功能 3

## P1\_SLEW

P1 口翻转速度控制寄存器

(92<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
0				P3	P2	P1	P0
r				rw	rw	rw	rw

符号	位	类型	描述
Pn (n = 0 - 3)	n	rw	大电流 P1 口引脚 n 翻转速度控制 0 选择陡变沿。 1 选择缓变沿。
0	[7:4]	r	保留 读取返回 0；应写入 0。

## P1\_OCPEN

P1 口过电流保护使能寄存器

(92<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
0				P3	P2	P1	P0
r				rw	rw	rw	rw

符号	位	类型	描述
Pn (n = 0 - 3)	n	rwh	大电流 P1 口引脚 n 过电流保护使能 0 禁用过电流保护。 1 使能过电流保护。
0	[7:4]	r	保留 读取返回 0；应写入 0。

并行端口

## P1\_OCD

P1 口过电流检测寄存器

(92<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
P3_OCEN	P2_OCEN	P1_OCEN	P0_OCEN	P3_OCF	P2_OCF	P1_OCF	P0_OCF
rw	rw	rw	rw	rwh	rwh	rwh	rwh

符号	位	类型	描述
Pn_OCF (n = 0 - 3)	n	rwh	大电流 P1 口引脚 n 过电流检测 0 未产生过电流。向该位写 0 将清除锁存值。 1 检测到过电流。
Pn_OCEN (n = 0 - 3)	y+4	rw	大电流 P1 口引脚 n 过电流中断使能 0 禁止产生过电流中断。 1 使能产生过电流中断。

## 11.4 P2 口

P2 口是一个通用单向输入口。在 XC83x 中，P2 口包含 P2.0 - P2.7 共 8 个引脚。P2 寄存器归纳见[表 11-10](#)。

**表 11-10 P2 口寄存器**

寄存器缩写名	寄存器全名
P2_DATAIN	P2 口数据寄存器
P2_EN	P2 口使能寄存器
P2_PUDSEL	P2 口上拉 / 下拉选择寄存器
P2_PUDEN	P2 口上拉 / 下拉使能寄存器



## 11.4.1 功能

P2 口输入和输出功能归纳见[表 11-11](#)。

**表 11-11 P2 口输入功能**

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P2.0	输入	GPI	P2_DATAIN.P0	
		ALT 1	CCPOS0_1	CCU6
		ALT 2	T12HR_2	CCU6
		ALT 3	T13HR_2	CCU6
		ALT 4	T2EX_3	定时器 T2
		ALT 5	T2_1	定时器 T2
		ALT 6	EXINT0_3	SCU
		ANALOG	AN0	ADC
P2.1	输入	GPI	P2_DATAIN.P1	
		ALT 1	CCPOS1_1	CCU6
		ALT 2	RXD_5	UART
		ALT 3	MTSR_6	SSC
		ALT 4	–	
		ALT 5	T0_1	定时器 T0
		ALT 6	EXINT1_1	SCU
		ANALOG	AN1	ADC
P2.2	输入	GPI	P2_DATAIN.P1	
		ALT 1	CCPOS2_1	CCU6
		ALT 2	T12HR_3	CCU6
		ALT 3	T13HR_3	CCU6
		ALT 4	SCK_3	SSC
		ALT 5	T1_1	定时器 T1
		ALT 6	EXINT2_0	SCU
		ANALOG	AN2	ADC

**表 11-11 P2 口输入功能**

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P2.3	输入	GPI	P2_DATAIN.P3	
		ALT 1	CCPOS0_2	CCU6
		ALT 2	CTRAP_2	CCU6
		ALT 3	–	
		ALT 4	–	
		ALT 5	T2_2	定时器 T2
		ALT 6	EXINT3	SCU
		ANALOG	AN3	ADC
P2.4	输入	GPI	P2_DATAIN.P4	
		ALT 1	–	
		ALT 2	T12HR_5	CCU6
		ALT 3	T13HR_5	CCU6
		ALT 4	–	
		ALT 5	T2_3	
		ALT 6	–	
		ANALOG	AN4	ADC
P2.5	输入	GPI	P2_DATAIN.P5	
		ALT 1	–	
		ALT 2	T12HR_7	CCU6
		ALT 3	T13HR_7	CCU6
		ALT 4	–	
		ALT 5	–	
		ALT 6	–	
		ANALOG	AN5	ADC

**表 11-11 P2 口输入功能**

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P2.6	输入	GPI	P2_DATAIN.P6	
		ALT 1	–	
		ALT 2	–	
		ALT 3	–	
		ALT 4	SCK_2	SSC
		ALT 5	–	
		ALT 6	EXINT6	SCU
		ANALOG	AN6	ADC
P2.7	输入	GPI	P2_DATAIN.P7	
		ALT 1	–	
		ALT 2	RXD_6	UART
		ALT 3	T2EX_6	定时器 T2
		ALT 4	MTSR_7	SSC
		ALT 5	–	
		ALT 6	EXINT0_4	SCU
		ANALOG	AN7	ADC

## 11.4.2 寄存器说明

### P2\_DATAIN

P2 口数据寄存器

(94<sub>H</sub>)

复位值: XX<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
<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

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rh	<b>P2 口引脚 n 的数据值</b> 0 P2 口引脚 n 的数据值 = 0 1 P2 口引脚 n 的数据值 = 1

### P2\_EN

P2 口使能寄存器

(94<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rw	<b>P2 口引脚 n 的输入驱动器控制</b> 0 使能输入驱动器 (缺省值) 1 禁用输入驱动器

### P2\_PUDSEL

P2 口上拉 / 下拉选择寄存器

(93<sub>H</sub>)

复位值: FF<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> <b>(n = 0 - 7)</b>	n	rw	<b>P2 口引脚 n 上拉 / 下拉选择</b> 0 选择下拉器件 1 选择上拉器件 (缺省值)

**P2\_PUDEN**
**P2 口上拉 / 下拉使能寄存器**
**(94<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: 1**

7	6	5	4	3	2	1	0
<b>P7</b>	<b>P6</b>	<b>P5</b>	<b>P4</b>	<b>P3</b>	<b>P2</b>	<b>P1</b>	<b>P0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>Pn</b> <b>(n = 0 - 7)</b>	n	rw	<b>P2 口引脚 n 上拉 / 下拉使能</b> 0 禁用上拉或下拉器件 (缺省值) 1 使能上拉或下拉器件

## 11.5 P3 口

P3 口是一个通用双向口。P3 口寄存器归纳见[表 11-12](#)。

**表 11-12 P3 口寄存器**

寄存器缩写名	寄存器全名
P3_DATAIN	P3 口数据输入寄存器
P3_DATAOUT	P3 口数据输出寄存器
P3_OD	P3 口漏极开路控制寄存器
P3_PUDSEL	P3 口上拉 / 下拉选择寄存器
P3_PUDEN	P3 口上拉 / 下拉使能寄存器
P3_ALTSEL0	P3 口复用功能选择寄存器 0
P3_ALTSEL1	P3 口复用功能选择寄存器 1

### 11.5.1 功能

P3 口输入和输出功能归纳见[表 11-13](#)。

**表 11-13 P3 口 I/O 功能**

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P3.0	输入	GPI	P3_DATAIN.P0	
		ALT1	—	
		ALT2	SCL_2	IIC
		ALT3	SCK_1	SSC
		ALT4	—	
		ALT5	EXINT2_1	SCU
		ALT6	—	
	输出	GOP	P3_DATAOUT.P0	
		ALT1	COL6	LED/TS
		ALT2	SCK_1	SSC
		ALT3	SCL_2	IIC
		ANALOG	XTAL4	XTAL 振荡器

表 11-13 P3 口 I/O 功能

端口引脚	I/O	功能选择	连接信号	来自 / 送至模块
P3.1	输入	GPI	P3_DATAIN.P1	
		ALT1	RXD_4	UART
		ALT2	RTCCLK	
		ALT3	MTSR_4	SSC
		ALT4	MRST_4	SSC
		ALT5	EXINT0_5	SCU
		ALT6	—	
		ANALOG	XTAL3	XTAL 振荡器
	输出	GOP	P3_DATAOUT.P1	
		ALT1	COLA_0	LED/TS
		ALT2	MTSR_4	SSC
		ALT3	EXF2_1	定时器 T2
P3.2	输入	GPI	P3_DATAIN.P2	
		ALT1	RXD_3	UART
		ALT2	SDA_2	IIC
		ALT3	MTSR_5	SSC
		ALT4	MRST_5	SSC
		ALT5	EXINT0_6	SCU
		ALT6	T2EX_7	定时器 T2
		其它	SPD_0	SPD
	输出	GOP	P3_DATAOUT.P2	
		ALT1	TXD_3	UART
		ALT2	MRST_5	SSC
		ALT3	SDA_2	IIC
		其它	SPD_0	SPD

## 11.5.2 寄存器说明

### P3\_DATAIN

P3 口数据输入寄存器

(C9<sub>H</sub>)

复位值: XX<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0					P2	P1	P0
r					rh	rh	rh

符号	位	类型	描述
Pn (n = 0 - 2)	n	rh	P3 口引脚 n 的数据值 0 P3 口引脚 n 的数据值 = 0 1 P3 口引脚 n 的数据值 = 1
0	[7:3]	r	保留 读取返回 0；应写入 0。

### P3\_DATAOUT

P3 口数据输出寄存器

(C8<sub>H</sub>)

复位值: 07<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0					P2	P1	P0
r					rw	rw	rw

符号	位	类型	描述
Pn (n = 0 - 2)	n	rw	P3 口引脚 n 的数据值 0 P3 口引脚 n 的数据值 = 0 1 P3 口引脚 n 的数据值 = 1
0	[7:3]	r	保留 读取返回 0；应写入 0。



并行端口

### P3\_OD

P3 口漏极开路控制寄存器

(C8<sub>H</sub>)

复位值：07<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
		0			P2	P1	P0
		r			rw	rw	rw

符号	位	类型	描述
Pn (n = 0 - 2)	n	rw	<b>P3 口引脚 n 漏极开路模式选择</b> 0 正常模式；0 和 1 可被有效输出 1 漏极开路模式；只有 0 可被有效输出（缺省值）
0	[7:3]	r	<b>保留</b> 读取返回 0；应写入 0。

### P3\_PUDSEL

P3 口上拉 / 下拉选择寄存器

(C8<sub>H</sub>)

复位值：07<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
		0			P2	P1	P0
		r			rw	rw	rw

符号	位	类型	描述
Pn (n = 0 - 2)	n	rw	<b>P3 口引脚 n 上拉 / 下拉选择</b> 0 选择下拉器件 1 选择上拉器件（缺省值）
0	[7:3]	r	<b>保留</b> 读取返回 0；应写入 0。

并行端口

### P3\_PUDEN

P3 口上拉 / 下拉使能寄存器

(C9<sub>H</sub>)

复位值: 07<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
		0			P2	P1	P0
		r			rw	rw	rw

符号	位	类型	描述
Pn (n = 0 - 2)	n	rw	P3 口引脚 n 上拉 / 下拉使能 0 禁用上拉或下拉器件 1 使能上拉或下拉器件 (缺省值)
0	[7:3]	r	保留 读取返回 0；应写入 0。

### P3\_ALTSELx(x = 0 - 1)

P3 口复用功能选择寄存器

(C8<sub>H</sub>+x)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
		0			P2	P1	P0
		r			rw	rw	rw

符号	位	类型	描述
Pn (n = 0 - 2)	n	rw	见表 11-14
0	[7:3]	r	保留 读取返回 0；应写入 0。

表 11-14 位 P3\_ALTSEL0.Pn 和 P3\_ALTSEL1.Pn 的功能

P3_ALTSEL1.Pn	P3_ALTSEL0.Pn	功能
0	0	GPIO
0	1	选择复用功能 1

表 11-14 位 P3\_ALTSEL0.Pn 和 P3\_ALTSEL1.Pn 的功能

P3_ALTSEL1.Pn	P3_ALTSEL0.Pn	功能
1	0	选择复用功能 2
1	1	选择复用功能 3

## 12 乘法 / 除法单元

### 12.1 概述

乘法 / 除法单元 (MDU) 具有快速 16 位乘法、16 位和 32 位除法功能以及移位和归一化特性。集成 MDU 的 XC83x 内核支持需要进行快速数学计算的实时控制应用。

MDU 共使用 14 个寄存器，其中 12 个寄存器用于数据处理；1 个寄存器用于控制 MDU 操作；1 个寄存器用于存储状态标志。和其它用于外设控制的寄存器一样，这些寄存器被映射为特殊功能寄存器。MDU 操作不依赖于 CPU、与 CPU 同时工作。

#### 特性

- 快速有符号 / 无符号 16 位乘法
- 快速有符号 / 无符号 32 位除以 16 位和 16 位除以 16 位操作
- 带有一次左移操作的有符号 16 位乘法（支持定点数计算，格式为 Q15）
- 带有一次右移操作的有符号 32 位除以 16 位除法（支持定点数计算，格式为 Q15）
- 32 位无符号归一化操作
- 32 位算术 / 逻辑移位操作

表 12-1 给出各种操作中计算所需的时钟周期数。

表 12-1 MDU 操作特性参数

操作	结果	余数	计算所需时钟周期数
有符号 32 位 /16 位	32 位	16 位	33
带有一次右移操作的 有符号 32 位 /16 位	32 位	16 位	33
有符号 16 位 /16 位	16 位	16 位	17
有符号 16 位 × 16 位	32 位	–	16
带有一次左移操作的 有符号 16 位 × 16 位	32 位	–	16
无符号 32 位 /16 位	32 位	16 位	32
无符号 16 位 /16 位	16 位	16 位	16
无符号 16 位 /16 位	32 位	–	16
32 位归一化	–	–	移位次数 +1（最大 32）
32 位左移 / 右移	–	–	移位次数 +1（最大 32）

## 12.2 系统信息

本节给出 MDU 的相关系统信息。

### 12.2.1 时钟配置

MDU 由 FPCLK 提供时钟，它以 48 MHz 的固定频率工作。

如果完全不需要 MDU 功能，可关闭其输入时钟禁止该模块工作，最大限度降低功耗。这可通过置位寄存器 PMCON1 中的位 MDU\_DIS 来实现。

在访问 PMCON1 寄存器之前必须先对 SCU\_PAGE 寄存器中的位域 PAGE 进行设置。

**PMCON1**  
**外设管理控制寄存器 1** (EF<sub>H</sub>) 复位值: FF<sub>H</sub>  
**RMAP: 0, PAGE: 1**

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	CDC_DIS	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rW	rW	rW	rW	rW	rW	rW	rW

符号	位	类型	描述
MDU_DIS	4	rw	<b>MDU 禁用请求位，高电平有效</b> 0 MDU 正常工作 1 请求禁用 MDU（缺省）

## 12.2.2 中断事件及分配

由 MDU 产生的中断事件以及相应的事件中断使能位和标志位归纳见表 12-2。

表 12-2 MDU 中断事件

中断事件	中断事件使能位	事件标志位
计算结束	MDUCON.IE	MDUSTAT.IRDY
出错		MDUSTAT.IERR

表 12-3 给出 MDU 中断源的中断节点分配。

表 12-3 MDU 事件的中断节点控制

中断事件	中断节点使能位	中断节点标志位	向量地址
计算结束	IEN1.EX2	—	43 <sub>H</sub>
出错			

## 12.3 功能说明

可将 MDU 视作一个特殊的乘法、除法、归一化和移位操作的协处理器。其操作可分为 3 个阶段（见图 12-1）：

### 阶段 1：加载 MDx 寄存器

在该阶段，CPU 将操作数载入 MDU 操作数寄存器（MDx）。

向位域 MDUCON.OPCODE 写入代表所需操作的 4 位操作码，选择 MDU 要执行的计算类型。

### 阶段 2：执行计算

只有在启动位 MDUCON.START 置位的情况下，才进入该阶段，紧接着忙碌标志置位。下一个周期启动位被自动清零。

在该阶段，MDU 独立工作，其操作和 CPU 并行进行。该阶段结束时，计算结果存储在 MDU 结果寄存器（MRx）中。

### 阶段 3：从 MRx 寄存器中读取结果

在该最后阶段，CPU 从 MRx 寄存器中取出计算结果。进入下一个计算阶段时 MRx 寄存器将被重写。

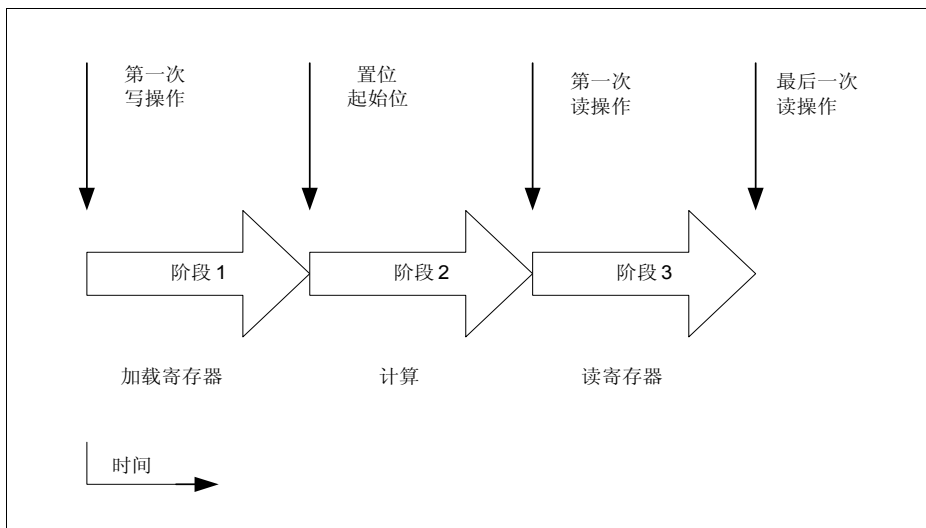


图 12-1 MDU 的操作阶段

### 12.3.1 除法操作

MDU 支持截断除法操作，这是现代处理器中的通行做法，同时符合 ISO C99 标准。截断除法操作中除法和取模运算的关系如下：

如果  $q = D \div d$   
 且  $r = D \bmod d$   
 则  $D = q * d + r$   
 且  $|r| < |d|$

其中“D”是被除数、“d”是除数、“q”是商、“r”是余数。

截断除法操作将商向零舍入（负向舍入），余数的符号总是和被除数的符号一致，即  $\text{sign}(r) = \text{sign}(D)$ 。

### 12.3.2 归一化

MDU 支持高达 32 位无符号数的归一化操作。

对保存在 MD0（最低有效字节）到 MD3（最高有效字节）中的无符号 32 位变量进行归一化。该特性主要用来支持使用浮点算术运算的应用。在归一化过程中，通过左移操作

将寄存器 MD0 到 MD3 中无符号 32 位变量前面所有的 0 除去。当 MSB（最高有效位）为 1 时整个操作完成。

归一化后，位域 MR4.SCTR 中包含进行左移操作的次数，该值将来可作为指数。归一化操作中最多移位次数为 31（ $= 2^5 - 1$ ）。

### 12.3.3 移位

MDU 支持高达 32 位无符号和有符号数的逻辑和算术移位。

逻辑移位时，从寄存器 MD3 的左端或者 MD0 的右端移入 0；算术左移等同于逻辑左移，但是算术右移时，符号位从 MD3 的左端移入。举例如下，如果对数据 0101<sub>B</sub> 和 1010<sub>B</sub> 进行一次算术右移，相应的结果为 0010<sub>B</sub> 和 1101<sub>B</sub>。

对于任何移位操作，寄存器位 MD4.SLR 指定移位方向，MD4.SCTR 指定移位次数。

*注：MDU 不检测因算术左移操作而引起的溢出。用户必须确保算术左移的结果始终在 MDU 边界值之内。*

### 12.3.4 带有一次左移操作的乘法

带有一次左移操作的乘法和有符号 16 位乘法运算相似，不同在于前者在完成乘法运算后，MDU 会对乘积进行一位的算数左移。

MDU 的这种操作有助于 Q15<sup>1)</sup> 格式的有符号定点数的乘法运算。两个 Q15 数据的乘积为一个 Q1.30 定点数。对该乘积进行一位算数左移，将产生一个 Q31 数，取 32 位数的高 16 位即可得到格式为 Q15 的乘积。

### 12.3.5 带有一次右移操作的除法

带有一次右移操作的除法和 32 位除以 16 位的运算相似，不同在于前者在进行除法运算之前先完成一位的算数右移。

MDU 的这种操作有助于 Q15 格式的有符号 16 位和 16 位定点数的除法运算。一般情况下，要除一个 Q15 的定点数得到一个 Q15 的商，在进行 32 位除以 16 位的运算之前，需要先将除数由  $2^{15}$  的因子按比例放大。

用户程序可将 16 位被除数写入 32 位操作数（数据）寄存器的高两字节，然后选择带有一次右移操作的除法运算来完成同样的功能。

### 12.3.6 忙碌标志

忙碌标志指示 MDU 正在进行计算。MDUSTAT.BSY 在计算开始时置位，在阶段 2 结束时清零。错误标志被置位时，忙碌标志也被清零。

若需要执行新的计算，首先查询忙碌标志，只有当其不被置位时，才能置位启动位，开始新的计算。若忙碌标志仍被置位，对启动位的任何写操作将被忽略。

1) Qx.y 是定点数的一种表示形式，其中的 x 表示这个定点数位数的整数部分（缺省为 0），y 则代表小数部分。符号位不在这种表示形式的考虑范围内。



### 12.3.7 检错

错误标志 MDUSTAT.IERR 指示计算过程出现了错误。当出现下列任一种情况时，由硬件置位该标志：

- 除数为 0
- 将保留操作码写入 MDUCON 寄存器

置位错误标志将中止当前操作、触发中断（见[章节 12.4](#)）。除数为 0 的操作不会立即置位错误标志，而是在除法操作的计算阶段结束时置位错误标志；MDUCON.START 置 1 后可检测到操作码错误。除数为 0 引起的错误会将一个饱和值载入 MRx 寄存器中。

*注：错误标志被置位时，MDU 模块不能保证任何结果的准确性。因此不应使用该结果。*

### 12.4 中断产生

MDU 的中断结构如[图 12-2](#)所示。MDU 可产生两种中断事件，每种中断事件置位一个中断标志。中断标志由软件清零。

阶段 2 结束时，中断标志 MDUSTAT.IRDY 由硬件置位，指示成功完成了一次计算。可从寄存器 MRx 中读取结果。中断线 INT\_O0 直接分配给该中断源。

计算过程出现错误时，也能产生中断，置位中断标志 MDUCON.IERR 来指示发生错误中断。除数为 0 时，只有在计算阶段结束时才置位 MDUCON.IERR。一旦 MDUCON.IERR 被置位，将中止任何正在执行的计算。除数为 0 时，会将一个饱和值载入 MRx 寄存器中。由位 MDUCON.IR 选择分配给该错误中断源的中断线。

只有当中断使能位 MDUCON.IE 置位且发生相应的中断事件时，才会产生中断。中断请求信号将保持 2 个 CCLK 时钟周期的高电平。

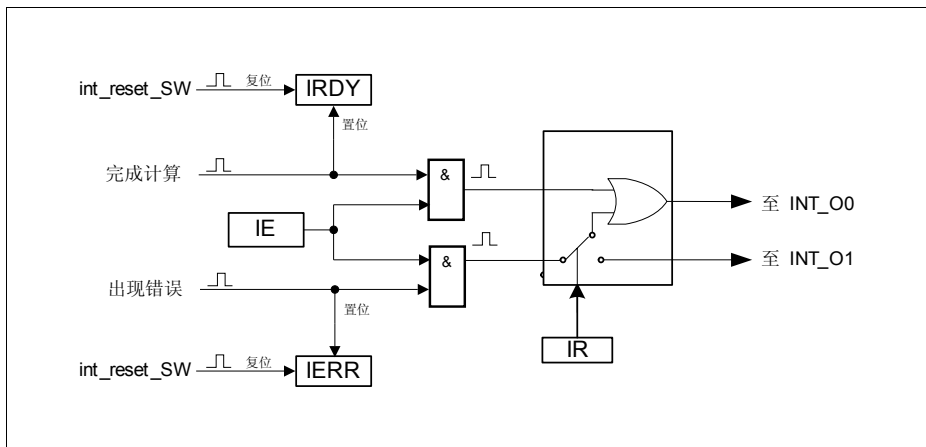


图 12-2 中断产生

## 12.5 寄存器说明

从标准（非映射）SFR 区访问 MDU SFR，这些 SFR 和相应的地址归纳见[表 12-4](#)。

**表 12-4 寄存器映射**

SFR	地址	寄存器名
MDU_MDUCON	B1 <sub>H</sub>	MDU 控制寄存器
MDU_MDUSTAT	B0 <sub>H</sub>	MDU 状态寄存器
MDU_MD0/MR0	B2 <sub>H</sub>	MDU 数据 / 结果寄存器 0
MDU_MD1/MR1	B3 <sub>H</sub>	MDU 数据 / 结果寄存器 1
MDU_MD2/MR2	B4 <sub>H</sub>	MDU 数据 / 结果寄存器 2
MDU_MD3/MR3	B5 <sub>H</sub>	MDU 数据 / 结果寄存器 3
MDU_MD4/MR4	B6 <sub>H</sub>	MDU 数据 / 结果寄存器 4
MDU_MD5/MR5	B7 <sub>H</sub>	MDU 数据 / 结果寄存器 5

MDx 和 MRx 寄存器共用相同的地址。由于不能对 MRx 寄存器进行写操作，因此，对这些地址进行的写操作都视作在对 MDx 寄存器进行写操作。

执行读操作时，需要通过位 MDUCON.RSEL 来选择是读取寄存器 MDx 还是寄存器 MRx。缺省读取寄存器 MRx。

MDU 的 14 个 SFR 包括一个控制寄存器 MDUCON、一个状态寄存器 MDUSTAT 和两组数据寄存器：MD0 到 MD5（包含操作数）和 MR0 到 MR5（包含计算结果）。

操作类型不同，每个 MDx 和 MRx 寄存器的作用不同，归纳见[表 12-5](#)和[表 12-6](#)。例如在乘法运算中，16 位乘数的低字节必须写入 MD4，高字节写入 MD5。

**表 12-5 MDx 寄存器**

寄存器	寄存器在操作中的作用			
	16 位乘法	32/16 位除法	16/16 位除法	归一化和移位
MD0	M'andL	D'endL	D'endL	OperandL
MD1	M'andH	D'end	D'endH	Operand
MD2	–	D'end	–	Operand
MD3	–	D'endH	–	OperandH
MD4	M'orL	D'orL	D'orL	控制
MD5	M'orH	D'orH	D'orH	–

表 12-6      MRx 寄存器

Register	寄存器在操作中的作用			
	16 位乘法	32/16 位除法	16/16 位除法	归一化和移位
MR0	PrL	QuoL	QuoL	ResultL
MR1	Pr	Quo	QuoH	Result
MR2	Pr	Quo	–	Result
MR3	PrH	QuoH	–	ResultH
MR4	M'orL	RemL	RemL	控制
MR5	M'orH	RemH	RemH	–

缩写:

- D'end: 被除数, 除法运算的第一个操作数
- D'or: 除数, 除法运算的第二个操作数
- M'and: 被乘数, 乘法运算的第一个操作数
- M'or: 乘数, 乘法运算的第二个操作数
- Pr: 积, 乘法运算的结果
- Rem: 余数
- Quo: 商, 除法运算的结果
- ...L: 该字节为 16 位或 32 位操作数的低有效字节
- ...H: 该字节为 16 位或 32 位操作数的高有效字节

MDx 寄存器具有映射寄存器, 在计算开始时将数据从实际寄存器锁存到映射寄存器中, 从而使得在进行当前计算的同时, 可以将下一组操作数写入 MDx 寄存器。

操作中未用到的 MDx 和 MRx 寄存器对用户来说是未被定义的。对于归一化和移位操作, 寄存器 MD4 和 MR4 用作移位输入和输出控制寄存器, 它们规定移位方向并保存已执行的移位操作的次数。

### 12.5.1      操作数和结果寄存器

MDx 和 MRx 寄存器用来存储操作数和计算结果。MD4 和 MR4 还用做移位和归一化操作的输入输出控制寄存器。

**MDU\_MDx (x = 0 - 5)**

MDU 数据寄存器 x [ 操作数 ]

 $(B2_H + x * 1)$ 

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
DATA							
rw							

符号	位	类型	描述
DATA	[7:0]	rw	操作数 见 表 12-5。

**MDU\_MRx (x = 0 - 5)**

MDU 数据寄存器 x [ 结果 ]

 $(B2_H + x * 1)$ 

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
DATA							
rh							

符号	位	类型	描述
DATA	[7:0]	rh	计算结果 见 表 12-6。

**MDU\_MD4**

移位输入控制寄存器

 $(B6_H)$ 

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0		SLR		SCTR			
rw		rw		rw			

符号	位	类型	描述
<b>SCTR</b>	[4:0]	rw	<b>移位计数器</b> 写入 SCTR 的计数值决定一次移位操作要执行的移位次数。
<b>SLR</b>	5	rw	<b>移位方向</b> 0 <sub>B</sub> 选择左移位 1 <sub>B</sub> 选择右移位
<b>0</b>	[7:6]	rw	<b>保留</b> 应写入 0，读取将返回未定义的数据。

#### MDU\_MR4

移位输出控制寄存器

(B6<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0			SCTR				
rh			rh				

符号	位	类型	描述
<b>SCTR</b>	[4:0]	rw	<b>移位计数器</b> 一次归一化操作之后， SCTR 中包含所执行的归一化移位操作的次数。
<b>0</b>	[7:5]	rh	<b>保留</b> 应写入 0，读取将返回未定义的数据。

### 12.5.2 控制寄存器

寄存器 MDUCON 选择操作类型并控制相应操作的启动。

#### MDU\_MDUCON

MDU 控制寄存器

(B1<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
IE	IR	RSEL	START	OPCODE			
rw	rw	rw	rw	rw			

符号	位	类型	描述
<b>OPCODE</b>	[3:0]	rw	<b>操作码</b> 0000 <sub>B</sub> 无符号 16 位乘法 0001 <sub>B</sub> 无符号 16 位 /16 位除法 0010 <sub>B</sub> 无符号 32 位 /16 位除法 0011 <sub>B</sub> 32 位逻辑左 / 右移 0100 <sub>B</sub> 有符号 16 位乘法 0101 <sub>B</sub> 有符号 16 位 /16 位除法 0110 <sub>B</sub> 有符号 32 位 /16 位除法 0111 <sub>B</sub> 32 位算术左 / 右移 1000 <sub>B</sub> 32 位归一化操作 1001 <sub>B</sub> 带有一次左移操作的有符号 16 位乘法 1010 <sub>B</sub> 带有一次右移操作的有符号 32 位 /16 位除法 其它：保留
<b>START</b>	4	rwh	<b>启动位</b> START 由软件置位、硬件复位。 0 <sub>B</sub> 未开始执行操作 1 <sub>B</sub> 开始执行操作
<b>RSEL</b>	5	rw	<b>读选择</b> 0 <sub>B</sub> 读取 MRx 寄存器 1 <sub>B</sub> 读取 MDx 寄存器
<b>IR</b>	6	rw	<b>中断线选择</b> 0 <sub>B</sub> 两个中断源各有专用的中断线 1 <sub>B</sub> 两个中断源共用一条中断线 INT_O0
<b>IE</b>	7	rw	<b>中断使能</b> 0 <sub>B</sub> 禁止中断 1 <sub>B</sub> 使能中断

注： 在计算阶段，当忙碌标志 **MDUSTAT.BSY** 置位时，不允许对 **MDUCON** 进行写操作。

注： 若 **MDUCON.START** 置 1，向 **MDUCON** 寄存器的操作码位域写入保留值将会导致出错。

### 12.5.3 状态寄存器

寄存器 MDUSTAT 中存放 MDU 的状态标志。

#### MDU\_MDUSTAT

MDU 状态寄存器

(B0<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
		0			BSY	IERR	IRDY
		r			rh	rwh	rwh

符号	位	类型	描述
IRDY	0	rwh	<b>计算结果准备就绪中断</b> IRDY 位由硬件置位、软件复位。 0 <sub>B</sub> 成功执行操作后不触发中断 1 <sub>B</sub> 成功执行操作后触发中断
IERR	1	rwh	<b>错误中断</b> IERR 位由硬件置位、软件复位。 0 <sub>B</sub> 出现错误时不触发中断 1 <sub>B</sub> 出现错误时触发中断
BSY	2	rh	<b>忙碌标志</b> 0 <sub>B</sub> MDU 未进行计算 1 <sub>B</sub> MDU 仍在进行计算
0	[7:3]	r	<b>保留</b> 读操作返回 0，应写入 0。

## 13 CORDIC 协处理器

本章将对 CORDIC 协处理器予以说明。

### 13.1 概述

这里仅对 CORDIC 协处理器的特性予以说明，不涉及 CORDIC 算法的理论背景。

CORDIC 算法是计算三角（圆）函数、线性函数、双曲函数及其它相关函数的非常有用的收敛算法。该算法不仅支持欧几里德平面中的向量旋转，还支持线性和双曲面中的向量旋转。

CORDIC 算法是一个迭代过程，其截断误差是固有误差。每次计算进行 16 次迭代、内核数据宽度至少为 20 位时，CORDIC 协处理器可达到更高的精度。和其它复杂算法相比，该算法的主要优点在于硬件开销较低。

一般 CORDIC 算法对应如下 CORDIC 方程。因子  $m$  控制向量旋转并选择圆函数（三角函数）、线性和双曲函数相对应的角度设置：

$$x_{i+1} = x_i - m \cdot d_i \cdot y_i \cdot 2^{-i} \quad (13.1)$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i} \quad (13.2)$$

$$z_{i+1} = z_i - d_i \cdot e_i \quad (13.3)$$

其中：

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

$m = 0$  线性函数， $e_i = 2^{-i}$

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

在本文档中，CORDIC 数据使用下列术语：

- 结果数据：CORDIC 计算结束时，最终的结果数据（位 BSY 不再有效）
- 计算数据：CORDIC 迭代的中间数据或最后一次迭代的数据
- 初始数据：第一次 CORDIC 迭代所使用的数据，通常是用户初始化数据

### 特性

CORDIC 协处理器的主要特性如下：

- 操作模式
  - 支持所有 CORDIC 操作模式，可用来解圆（三角）函数、线性（乘－加，除－加）函数和双曲函数
  - 所有操作模式都集成有相应的查找表（LUT）
- 圆函数向量模式：求解角度和幅值时，X 和 Y 的初始值可扩展在  $[-2^{15}, (2^{15}-1)]$  之间取值
- 圆函数旋转模式：Z 的初始值可扩展在  $[-2^{15}, (2^{15}-1)]$  之间取值，对应求解三角函数时角度在  $[-\pi, ((2^{15}-1)/2^{15})\pi]$  之间取值
- 可通过门控时钟信号关闭模块
- 16 位数据访问宽度



## CORDIC 协处理器

- X、Y 内核数据包含 24 位数据外加 2 位溢出位
- Z 内核数据包含 20 位数据外加 1 位溢出位
- KEEP 位，上次的计算结果保留在内核寄存器中用于新的计算
- 每次计算进行 16 次迭代：从置位启动位（ST）到置位计算结束（EOC）标志，最多需要 41 个时钟周期，其中不包括读写数据字节占用的时间。
- 数据处理采用 2 补码
  - 只有一种例外情况：用户可将 X 结果数据设定为无符号数
- X 和 Y 数据通常被当作是整数或有理数，X 和 Y 的数据格式必须一致
- LUT 入口数据是 20 位有符号整数
  - atan 和 atanh LUT 入口数据是角度值的整数表示（S19），按比例因子调整后由整数  $[-2^{15}, (2^{15}-1)]$  代表角度  $[-\pi, ((2^{15}-1)/2^{15})\pi]$
  - 圆函数和双曲函数可访问的 Z 的结果数据是整数，数据格式为 S15
- 线性函数的仿真 LUT
  - 数据格式为 1 位整数和 15 位小数位（1.15）
  - 线性函数可访问的 Z 的结果数据是有理数，固定数据格式 S4.11（有符号 4Q16）
- 截断误差
  - 因为 LSB 被截断，CORDIC 的计算结果可能返回一个近似值
  - CORDIC 计算结果精度高（尤其在圆函数模式中）
- 中断
  - 计算结束产生中断
  - 中断使能和相应的中断标志

## 13.2 系统信息

本节给出 CORDIC 协处理器的相关系统信息。

### 13.2.1 时钟配置

CORDIC 协处理器由 FPCLK 提供时钟，它以 48 MHz 的固定频率工作。

如果完全不需要 CORDIC 协处理器功能，可关闭其输入时钟禁止该模块工作，最大限度降低功耗。这可通过置位寄存器 PMCON1 中的位 CDC\_DIS 来实现。

#### 13.2.1.1 外设管理控制寄存器 1

##### PMCON1

外设管理控制寄存器 1

(EF<sub>H</sub>)

复位值：FF<sub>H</sub>

RMAP:0, PAGE:1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	CDC_DIS	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	rw	rw	rw	rw	rw	rw

CORDIC 协处理器

符号	位	类型	描述
CDC_DIS	5	rw	<b>CORDIC 禁用请求位，高电平有效</b> 0 CORDIC 正常工作 1 请求禁用 CORDIC（缺省）

### 13.2.2 中断事件及分配

由 CORDIC 协处理器产生的中断事件以及相应的事件中断使能位和标志位归纳见表 13-1。

表 13-1 CORDIC 协处理器 中断事件

中断事件	中断事件使能位	事件标志位
计算结束	STATC.INT_EN	STATC.EOC

表 13-2 给出 CORDIC 协处理器中断源的中断节点分配。

表 13-2 CORDIC 协处理器事件的中断节点控制

中断事件	中断节点使能位	中断节点标志位	向量地址
计算结束	IEN1.EX2	—	43 <sub>H</sub>

### 13.3 功能描述

以下各节将描述 CORDIC 协处理器的功能。

#### 13.3.1 CORDIC 协处理器的操作

CORDIC 协处理器可工作在旋转模式或向量模式下，用来计算圆（三角）函数、线性（乘 - 加，除 - 加）函数或者双曲函数。通过 **CD\_CON** 控制寄存器进行工作模式选择。

清零寄存器 **CD\_CON** 中的 **KEEP** 位，可初始化内核数据寄存器。如果 **ST\_MODE = 1**，置位 **ST** 将启动新的计算；否则，在缺省状态 **ST\_MODE = 0** 下，对寄存器 **CD\_CORDXL** 执行写操作之后会自动启动新的计算。每次计算的迭代次数固定为 16 次，计算过程中 **BSY** 置位，指示处于忙碌状态。计算结束后该忙碌标志由硬件清零。

CORDIC 计算的第一步（假如相应的 **KEEP** 未被置位），将数据寄存器 **CD\_CORDxL** 和 **CD\_CORDxH** 中的初始值载入内核数据寄存器中。在计算过程中，内核数据寄存器中总是保存最新的中间数据；计算完成后，它们保存结果数据。

数据寄存器 **CD\_CORDxL** 和 **CD\_CORDxH** 是映射寄存器，可在运算过程中被写入而不会影响正常运算。只有在有效设置 **ST** 或当 **ST\_MODE = 0** 时，对 **X** 的低字节 **CD\_CORDXL** 执行写操作之后（假如相应的 **KEEP** 未被置位），映射寄存器中的值才能被传送到内核数据寄存器中。必须在一次计算结束后（**BSY** 不再有效）、新的计算开始之前读取结果数据。若位 **CD\_STATC.DMAP = 0**，将直接从内核数据寄存器中读取结果数据。内核数据直接送到总线上，因而作为映射寄存器的数据寄存器在该操作期间不会被覆盖（改写）。若位 **CD\_STATC.DMAP = 1**，将读取映射寄存器的内容，尽管只能读出用户定义的初始化数据。

每次计算结束后，**CD\_STATC.BSY** 返回 0，计算结束（**EOC**）标志被置位，若 **INT\_EN = 1**，中断被使能，中断请求信号被激活。检查 **X**、**Y** 和 **Z** 的结果数据，若有数据溢出，**ERROR** 位被置位。开始新的计算、或在读取 **ERROR** 之后，该标志位被自动清零。

开始新的计算时，内核数据寄存器不会再保存前一次的计算结果，而将始终保存计算的初始值、或上一次 CORDIC 迭代的（中间）结果。

在运算过程中、**BSY** 置位时，置位 **ST** 无效。必须在 **BSY** 失效后一段时间、位 **ST** 再次被置位才能启动新的计算。同样地，在运算期间（由位 **BSY** 指示）改变工作模式的操作无效。

#### 13.3.2 结果数据归一化

CORDIC 协处理器在所有操作模式下都将返回归一化的 **X** 和 **Y** 结果数据，计算公式如下：

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

另一方面，对于 **Z** 结果数据的解释有所不同，这由所使用的 CORDIC 函数决定：

## CORDIC 协处理器

对于**线性函数**，CORDIC 计算得到的 Z 数据无需额外处理，从而可直接当作结果数据。可访问的 Z 的结果数据为实数，格式为有符号 4Q16。

对于**圆函数**和**双曲函数**，可访问的 Z 的结果数据是一个归一化的整数值，由整数 $[-2^{15}, (2^{15}-1)]$  代表角度范围  $[-\pi, ((2^{15}-1)/2^{15})\pi]$ 。CORDIC 协处理器按照如下比例关系调整 Z 数据：

$$\text{输入 Z 的初始值} = \text{实数 Z 的初始值 (以弧度为单位)} \times \frac{32768}{\pi}$$

$$\text{实数 Z 结果数据 (以弧度为单位)} = \text{Z 结果数据} \times \frac{\pi}{32768}$$

CORDIC 计算结果数据包含一个固有的、由旋转模式或者向量模式引起的增益因子 K。每种 CORDIC 函数对应的 K 值不同，如**表 13-3** 所示。

**表 13-3 CORDIC 函数结果数据的固有增益因子**

函数	增益 K 的近似值
圆函数	1.64676
双曲函数	0.828
线性函数	1

### 13.3.3 CORDIC 协处理器操作模式

**表 13-4** 给出 CORDIC 协处理器的一般操作模式。该表中的 X、Y 和 Z 代表初始值； $X_{\text{final}}$ 、 $Y_{\text{final}}$  和  $Z_{\text{final}}$  代表计算结束，BSY 失效后的最终结果数据。

CORDIC 方程如下：

$$x_{i+1} = x_i - m \cdot d_i \cdot y_i \cdot 2^{-i} \quad (13.4)$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i} \quad (13.5)$$

$$z_{i+1} = z_i - d_i \cdot e_i \quad (13.6)$$

**表 13-4 CORDIC 协处理器操作模式和对应的结果数据**

函数	旋转模式	向量模式
	$d_i = \text{sign}(z_i), z_i \rightarrow 0$	$d_i = -\text{sign}(y_i), y_i \rightarrow 0$

## CORDIC 协处理器

表 13-4 CORDIC 协处理器操作模式和对应的结果数据

函数	旋转模式	向量模式
<b>圆函数</b> $m = 1$ $e_i = \text{atan}(2^{-i})$	$X_{\text{final}} = K[X \cos(Z) - Y \sin(Z)] / \text{MPS}$ $Y_{\text{final}} = K[Y \cos(Z) + X \sin(Z)] / \text{MPS}$ $Z_{\text{final}} = 0$ 其中 $K \approx 1.64676$	$X_{\text{final}} = K \sqrt{X^2 + Y^2} / \text{MPS}$ $Y_{\text{final}} = 0$ $Z_{\text{final}} = Z + \text{atan}(Y / X)$ 其中 $K \approx 1.64676$
	解 $\cos(Z)$ 和 $\sin(Z)$ , 设置 $X = 1 / K$ , $Y = 0$ 可用定义域: 由于采用了预处理逻辑, 所以 $X$ 、 $Y$ 和 $Z$ 可全域取值。	解向量幅值 ( $\sqrt{x^2 + y^2}$ ), 设置 $X = x / K$ , $Y = y / K$ 可用定义域: 由于采用了预处理和后处理逻辑, 所以 $X$ 和 $Y$ 可全域取值。  解 $\text{atan}(Y / X)$ , 设置 $Z = 0$ 可用定义域: $X$ 和 $Y$ 全域取值, $X = 0$ 除外。
	关系: $\tan(v) = \sin(v) / \cos(v)$	关系: $\text{acos}(w) = \text{atan}[\sqrt{1-w^2} / w]$ $\text{asin}(w) = \text{atan}[w / \sqrt{1-w^2}]$
<b>线性函数</b> $m = 0$ $e_i = 2^{-i}$	$X_{\text{final}} = X / \text{MPS}$ $Y_{\text{final}} = [Y + X Z] / \text{MPS}$ $Z_{\text{final}} = 0$	$X_{\text{final}} = X / \text{MPS}$ $Y_{\text{final}} = 0$ $Z_{\text{final}} = Z + Y / X$
	解 $X \cdot Z$ , 设置 $Y = 0$ 可用定义域: $ Z  \leq 2$	解 $Y / X$ , 设置 $Z = 0$ 可用定义域: $ Y / X  \leq 2$ , $X > 0$

表 13-4 CORDIC 协处理器操作模式和对应的结果数据

函数	旋转模式	向量模式
双曲函数 $m = -1$ $e_i = \operatorname{atanh}(2^{-i})$	$X_{\text{final}} = k[X \cosh(Z) - Y \sinh(Z)] / \text{MPS}$ $Y_{\text{final}} = k[Y \cosh(Z) + X \sinh(Z)] / \text{MPS}$ $Z_{\text{final}} = 0$ 其中 $k \approx 0.828$	$X_{\text{final}} = k \sqrt{X^2 - Y^2} / \text{MPS}$ $Y_{\text{final}} = 0$ $Z_{\text{final}} = Z + \operatorname{atanh}(Y / X)$ 其中 $k \approx 0.828$
	解 $\cosh(Z)$ 、 $\sinh(Z)$ 和 $e^Z$ ， 设置 $X = 1 / k$ ， $Y = 0$ 可用定义域： $ Z  \leq 1.11\text{rad}$ ， $Y = 0$	解 $\sqrt{x^2 - y^2}$ ， 设置 $X = x / k$ ， $Y = y / k$ 可用定义域： $ y  <  x $ ， $X > 0$  解 $\operatorname{atanh}(Y / X)$ ，设置 $Z = 0$ 可用定义域： $ \operatorname{atanh}(Y / X)  \leq 1.11\text{rad}$ ， $X > 0$ .
	关系： $\tanh(v) = \sinh(v) / \cosh(v)$ $e^v = \sinh(v) + \cosh(v)$ $w^t = e^{t \ln(w)}$	关系： $\ln(w) = 2 \operatorname{atanh}[(w-1) / (w+1)]$ $\sqrt{w} = \sqrt{(w+0.25)^2 - (w-0.25)^2}$ $\operatorname{acosh}(w) = \ln[w + \sqrt{1 - w^2}]$ $\operatorname{asinh}(w) = \ln[w + \sqrt{1 + w^2}]$

## 使用提示

- 要求解相关函数，用户必须确保 CORDIC 数据的初始值 X、Y 和 Z 有意义（在收敛域之内），从而确保结果收敛。表 13-4 中的可用定义域给出 CORDIC 算法所支持的收敛域，除去使函数无意义的数据范围。关于收敛域的详细描述，参见章节 13.3.3.1。计算结果精度的详细描述，参见章节 13.3.5。
- 必须考虑对函数定义域的限制。例如，解  $\operatorname{atan}(Y/X)$  时，设置初始值  $X = 0$  无意义。不符合函数定义域的值将导致错误的 CORDIC 计算结果。
- 所有的数据输入均按 2 补码来处理。只有一种情况例外：用户可选择将 X 结果数据（仅 X 数据）按无符号数读取。
- 只有在圆函数向量模式下的 X 结果数据（仅 X 数据）才会始终为正、且大于初始值。因此，用户可能希望将 X 结果数据的 MSB 位用作数据位而非符号位。置位  $X\_USIGN = 1$ ，X 结果数据将被当作无符号数处理。
- 对于圆函数和双曲函数，由于对应的 LUT 固定，Z 值始终被当作有符号整数 S19 处理（数据访问格式为 S15）。LUT 中包含  $\operatorname{atan}(2^{-i})$ ， $i = 0, 1, 2, \dots, 15$  和  $\operatorname{atanh}(2^{-i})$ ， $i = 0, 1, 2, \dots, 15$  按比例因子调整后的整数值（S19），从而可用整数  $[-2^{19}, (2^{19}-1)]$  代表角度  $[-\pi, ((2^{19}-1)/2^{19})\pi]$ 。因此，对于圆函数和双曲函数，限定 Z 数据（不考虑收敛域）代表的角度范围  $[-\pi, ((2^{15}-1)/2^{15})\pi]$ ，超出该范围的 Z 计算结果将导致溢出错误。
- 对于线性函数，Z 数据始终被当作有符号小数 S4.15 处理（数据访问格式为 S4.11（有符号数 4Q16））。仿真 LUT 实际上是一个移位寄存器，其中存放  $2^{-i}$  的实数值，

## CORDIC 协处理器

数据格式为 1.15。因此，不管收敛域如何，从逻辑上来讲，Z 数据只对幅值小于 16 的数据有用。溢出错误由位 CD\_STATC.ERROR 指示。

- MPS 设置对 Z 数据无影响。用户必须确保选择恰当的 Z 初始值，以防止结果数据溢出和结果不正确。
- CORDIC 协处理器的设计要满足：当用户正确设置  $MPS > 1$  时，X 和 Y 数据不会出现内部溢出；且可完成结果数据的读取操作。然而，在这些情况下，MPS 设置越高，由于 LSB 位的丢失导致结果精度越低。
- 于在双曲函数旋转模式下，Y 的初始值必须设置为 0，因此结果精度受到限制。也就是说，在一次计算中，CORDIC 协处理器不能返回  $\cosh(Z) + i \sinh(Z)$  的正确结果。

### 13.3.3.1 收敛域

本用户手册不对 CORDIC 收敛的固有局限性进行理论性阐述。

为了保证结果数据收敛，要根据使用的操作模式，对幅值或初始数据值以及相应可用的数据格式加以限制。考虑到 CORDIC 结果数据的收敛性，下面给出通常应用中的可用定义域。

**旋转模式：**Z 数据必须收敛于 0。Z 数据初值必须等于或者小于  $\sum d_i \cdot e_i$ ，其中  $e_i$  随着迭代次数 i 递增而递减。换言之， $|Z| \leq LUT$  总和。对于圆函数来说，这意味着  $|Z| \leq$  代表 1.74 弧度的整数倍；对于线性函数来说， $|Z| \leq 2$ ；对于双曲函数来说， $|Z| \leq$  代表 1.11 弧度的整数倍。

**向量模式：**Y 数据必须收敛于 0。X 和 Y 的初始值受限于 Z 函数（由 LUT 决定）。对于圆函数来说，意味着  $|\tan(Y/X)| \leq 1.74$  弧度；对于线性函数来说， $|Y/X| \leq 2$ ；对于双曲函数来说， $|\tanh(Y/X)| \leq 1.11$  弧度。在向量模式中，还要求  $X > 0$ 。

CORDIC 协处理器的操作模式通常受到这些收敛域的限制，不过，圆函数旋转模式和圆函数向量模式可通过额外的预（后）处理逻辑以支持更宽的输入范围。

**圆函数旋转模式：**支持 Z 全域取值  $[-2^{15}, (2^{15}-1)]$ ，代表角度  $[-\pi, ((2^{15}-1)/2^{15})\pi]$ 。除了考虑溢出之外，对于 X 和 Y 初始值没有限制，而且可以通过设置 MPS 来消除溢出的可能性。

**圆函数向量模式：**支持 X 和 Y 全域取值  $[-2^{15}, (2^{15}-1)]$ ；Z 的初始值必须满足  $|Z| \leq \pi/2$  以避免可能出现的 Z 结果数据溢出。

*注：需要考虑对函数定义域的限制，以使计算结果有意义，例如，除数为 0 是无意义的。表 13-4 给出了各函数的“可用定义域”，兼顾 CORDIC 收敛性和函数的取值范围。*

*注：输入值可能在收敛域之内，但这并不能保证 CORDIC 结果数据的精度固定不变。CORDIC 协处理器的精度讨论请参见章节 13.3.5。*

### 13.3.3.2 溢出考虑

除了考虑收敛域，还必须考虑对输入数据幅值的限制，以防止结果数据溢出。

对于所有操作模式，CORDIC 协处理器处理数据溢出的方式都相同。可通过正确设置 MPS 来防止 X 和 Y 数据溢出，在一定程度上 MPS 值由 CORDIC 协处理器的操作模式和应用数据决定。

## CORDIC 协处理器

MPS 设置对 Z 数据无影响。对于圆函数和双曲函数，任何超出范围  $[-\pi, ((2^{15}-1)/2^{15})\pi]$  的角度都不能由 Z 表示，否则将导致 Z 数据产生溢出错误。请注意，由于 Z 的内核数据取值范围为  $[-\pi, ((2^{19}-1)/2^{19})\pi]$ ，按比例因子调整后由  $[-2^{19}, (2^{19}-1)]$  表示，因此始终按照这种方式对 Z 的读写数据归一化。对于线性函数，Z 是一个实数值，Z 的幅值一定不能超过 4 位整数。

### 13.3.4 CORDIC 协处理器数据格式

CORDIC 协处理器的初始数据 X、Y 和 Z 输入是 2 补码格式。结果数据也是 2 补码格式。

只有圆函数向量模式中的 X 结果数据有例外情况。X 结果数据的缺省数据格式为 2 补码，不过用户可通过设置位 `CD_CON.X_USIGN = 1`，将 X 结果数据按无符号数读取。这样可防止潜在的 X 数据结果溢出（包括对 MPS 的设置），因为此时 MSB 位是数据位。请注意，只有工作在圆函数向量模式时，置位 `X_USIGN = 1` 才有效，该操作产生的结果数据始终为正、且大于初始值。

一般来说，X 和 Y 输入数据可以是整数或有理数（小数）。但在任何计算中，X 和 Y 的数据格式必须保持一致。此外，数据是小数时，X 和 Y 必须具有相同个数的小数位。

对于圆函数或者双曲函数，Z 数据始终是归一化的整数；对于线性函数，可访问的 Z 数据是一个实数，其输入和结果数据格式为 S4.11（有符号 4Q16），是带 11 位小数位的小数。

数据归一化的详细描述请参见[章节 13.3.2](#)。

### 13.3.5 CORDIC 协处理器精度

每次 CORDIC 计算的迭代次数固定为 16 次，从迭代 0 开始。双曲函数例外，它从迭代 1 开始，按照设定的步骤进行迭代。可寻址的数据寄存器宽度为 16 位，而用于计算的内部内核 X 和 Y 数据寄存器宽度为 26 位（24 位数据位加上 2 位溢出位）；内部内核 Z 数据寄存器宽度为 21 位（20 位数据位加上 1 位溢出位）。有关 LUT 数据格式的详细描述，请参见[章节 13.5.1](#)和[章节 13.5.2](#)。

若输入数据值在规定的定义域内（见[表 13-4](#)），则可确保 CORDIC 协处理器每次计算的结果都收敛，尽管在每种操作模式下，各种数据格式所造成的结果数据精度并不固定。精度通过期望数据和结果数据之间的幅值差来衡量。“归一化误差”（ND）是一个通用术语，表示结果数据和期望数据之间的幅值偏差。计算该误差时将输入 / 结果数据当作整数值。若数据为有理数，则必须对误差幅值进行说明。例如，线性函数向量模式下 Z 的数据格式为 S4.11 - ND = 1（01<sub>B</sub>）意味着实际数据和期望数据之间的幅值误差不大于  $|2^{-11} + 2^{-11}|$ ；ND = 2（10<sub>B</sub>）意味着误差不超过  $|2^{-10} + 2^{-11}|$ ；ND = 3（11<sub>B</sub>）意味着误差不超过  $|2^{-11} + 2^{-10} + 2^{-11}|$ ；ND = 4（100<sub>B</sub>）意味着误差不超过  $|2^{-9} + 2^{-11}|$ ，依此类推。考虑到可能的截断误差，始终加上一个数值  $2^{-11}$ 。

**表 13-5** 列出一 次计算中归一化误差的概率。对于每一种 CORDIC 协处理器工作模式，输入设置总要满足特定条件（即在可用定义域内，可能还有附加条件），该概率值是大约一百万次不同输入设置对应的仿真结果。

使用有理数（小数）可以很容易的增加每种操作模式的精度。不过这仅适用于 X 和 Y 数据（X 和 Y 数据必须始终保持相同的数据格式），而 Z 的数据格式根据 LUT 的定义是固



# CORDIC 协处理器

定不变的。显然，对于给定的输入  $X$  和  $Y$ （和  $Z$ ），无论它们是整数还是有理数，希望计算结果应始终返回一个固定值，差别仅在于如何对输入和结果数据予以解释，也就是说，数据有没有小数位和有几个小数位。如果  $X$  和  $Y$  是整数，CORDIC 计算结果和期望数据之间的误差绝对不会比  $X$  和  $Y$  是有理数时的计算误差小。因此，应根据对应模式下的最大 ND 值，尽可能将  $X$  和  $Y$  设置为有理数并谨慎选择小数位位数。

**表 13-5 一次计算的归一化误差**

模式	$X$ 归一化误差	$Y$ 或者 $Z$ 归一化误差
圆函数向量模式	输入条件：可用定义域，且 $[(1.64676/2) \cdot \sqrt{X^2 + Y^2}] \geq 600$	
	0 : 50.8317%	0 : 55.8702%
	1 : 49.1683%	1 : 44.1298%
	$X \leq 1$ 对应的 ND	$Z \leq 1$ 对应的 ND
圆函数旋转模式	输入条件：可用定义域（ $X$ 、 $Y$ 和 $Z$ 全域取值）	
	0 : 50.7715%	0 : 51.2011%
	1 : 48.8579%	1 : 48.4944%
	2 : 0.3681%	2 : 0.3024%
	3 : 0.0023%	3 : 0.0020%
	4 : 0.0002%	4 : 0.0001%
	$X \leq 4$ 对应的 ND	$Y \leq 4$ 对应的 ND
线性函数向量模式	输入条件：可用定义域（ $ Y/X  \leq 2$ ， $X > 0$ ）	
	0 : 66.9170%	0 : 88.5676%
	1 : 33.0830%	1 : 11.4322%
	$X \leq 1$ 对应的 ND	2 : 0.0002% $Z \leq 2$ 对应的 ND
线性函数旋转模式	输入条件：可用定义域（ $ Z  \leq 2$ ）	
	0 : 69.7141%	0 : 62.4055%
	1 : 30.2859%	1 : 37.1965%
	$X \leq 1$ 对应的 ND	2 : 0.3980% $Y \leq 2$ 对应的 ND
双曲函数向量模式	输入条件：可用定义域（ $ Y  <  X $ ， $X > 0$ ， $ \operatorname{atanh}(Y/X)  \leq 1.11\text{rad}$ ）	
	0 : 34.5399%	0 : 58.3062%
	1 : 34.5438%	1 : 41.6938%
	2 : 17.9254%	$Z \leq 1$ 对应的 ND
	3 : 11.6747%	
	4 : 1.3162% $X \leq 4$ 对应的 ND	

表 13-5 一次计算的归一化误差

模式	X 归一化误差	Y 或者 Z 归一化误差
双曲函数旋转模式	输入条件：可用定义域 ( $ Z  \leq 1.11\text{rad}$ , $Y = 0$ )	
	0 : 14.9401%	0 : 40.4787%
	1 : 31.6474%	1 : 40.6711%
	2 : 23.7692%	2 : 11.9209%
	3 : 14.8353%	3 : 4.6940%
	4 : 7.4881%	4 : 1.7290%
	5 : 4.3398%	5 : 0.4453%
	6 : 2.4387%	6 : 0.0607%
	7 : 0.5267%	7 : 0.0003%
	8 : 0.0146%	$Y \leq 7$ 对应的 ND
	$X \leq 8$ 对应的 ND	

注： 不能保证多步计算的最终结果具有上面给出的精度/ 误差，举例来说，如果一个运算包含 2 次 CORDIC 计算，第二次计算要使用第一次的计算结果（通过置位相应的 KEEP 位使能该特性），由于近似和误差的累积，不能保证有上述精度。

### 13.3.6 CORDIC 协处理器性能

CORDIC 计算所需时间随着计算精度的增加而线性增加。增加迭代次数可得到更高的精度，这需要增加数据参数的宽度。

CORDIC 使用桶形移位寄存器进行数据移位。一次计算的迭代次数固定为 16 次，从开始计算到 EOC 标志置位总共不超过 41 个时钟周期。需要注意的是，只有在 一个时钟周期之后，ERROR 标志才有效。这样完整的计算时序也适用于那些包含附加数据处理的操作模式，也适用于需要进行重复迭代、以及外加一个时钟用于模式建立的双曲函数操作模式。

注： 上述时序不包含软件向 6 个数据寄存器加载初始数据，从 6 个数据寄存器读取最终结果所需要的时间。

### 13.4 中断

计算结束（EOC）是 CORDIC 协处理器的唯一中断源。若设置 CD\_STATC.INT\_EN = 1 使能中断，在 CORDIC 计算结束时中断请求信号被激活，由中断标志 CD\_STATC.EOC 指示该状态。如果 EOC 标志没有被软件清零，它会一直保持置位状态，直到读取 Z 结果数据的低字节（DMAP = 0）时，才被硬件清零。

处理 EOC 数据时，必须检查数据以确保 ERROR 标志没有被置位（该标志指示数据溢出）。

### 13.5 CORDIC 协处理器硬件

本节给出 CORDIC 协处理器查找表的相关信息。

#### 13.5.1 反正切和双曲反正切查找表

正切查找表（atan LUT）和双曲反正切查找表（atanh LUT）中的数据宽度分别为 20 位和 21 位。Atan LUT 的每个入口数据由 1 个符号位（MSB）和 19 个整数位组成；atanh LUT 的每个入口数据由 1 个重复位（MSB）、1 个符号位和 19 个整数位组成。

LUT 的内容是：

- atan LUT 数据格式为 S19，见表 13-6。

表 13-6 atan(2<sup>-i</sup>) 预计算值

迭代次数	atan(2 <sup>-i</sup> ) 预计算值 (16 进制)	迭代次数	atan(2 <sup>-i</sup> ) 预计算值 (16 进制)
i = 0	20000	i = 8	28C
i = 1	12E40	i = 9	146
i = 2	9FB4	i = 10	A3
i = 3	5111	i = 11	51
i = 4	28B1	i = 12	29
i = 5	145D	i = 13	14
i = 6	A2F	i = 14	A
i = 7	518	i = 15	5

- atanh LUT 数据格式为 S19，见表 13-7。

表 13-7 atanh(2<sup>-i</sup>) 预计算值

迭代次数	atanh(2 <sup>-i</sup> ) 预计算值 (16 进制)	迭代次数	atanh(2 <sup>-i</sup> ) 预计算值 (16 进制)
i = 0	-	i = 8	28C
i = 1	16618	i = 9	146
i = 2	A681	i = 10	A3
i = 3	51EA	i = 11	51
i = 4	28CC	i = 12	29
i = 5	1461	i = 13	14
i = 6	A30	i = 14	A
i = 7	518	i = 15	5

---

**CORDIC 协处理器**

Z 数据是实际角度的归一化表示。按比例因子调整后，使得  $[-\pi, ((2^{19}-1)/2^{19})\pi]$  等同于  $[-2^{19}, (2^{19}-1)]$ 。因为数据访问宽度为 16 位，因此 LSB 最低 4 位被截断。因此，从用户的角度来看，角度  $[-\pi, ((2^{15}-1)/2^{15})\pi]$  由  $[-2^{15}, (2^{15}-1)]$  来代表。

### 13.5.2 线性函数的仿真查找表

线性函数的仿真 LUT 实际是一个移位寄存器。仿真 LUT 的数据格式为 1Q16，1 个整数位（MSB），随后 15 个小数位。

对于线性函数来说，Z 为实数，内部 Z 数据的格式为有符号 4Q20。外部读取数据时，最低 4 位小数部分被截断，最终计算结果为 1 个符号位、4 个整数位、后跟 11 个小数位。

### 13.6 寄存器描述

从标准（非映射）SFR 区访问 CORDIC 协处理器的特殊功能寄存器，这些 SFR 和对应的地址归纳见 表 13-8。

表 13-8 地址映射

地址	寄存器
BA <sub>H</sub>	CD_CORDXL
BB <sub>H</sub>	CD_CORDXH
BC <sub>H</sub>	CD_CORDYL
BD <sub>H</sub>	CD_CORDYH
BE <sub>H</sub>	CD_CORDZL
BF <sub>H</sub>	CD_CORDZH
A0 <sub>H</sub>	CD_STATC
A1 <sub>H</sub>	CD_CON

#### 13.6.1 控制寄存器

CD\_CON 寄存器用于控制 CORDIC 协处理器的操作。若 CD\_STATC.BSY 置位，对该寄存器进行的写操作无效。

**CD\_CON**  
**CORDIC 控制寄存器** (A1<sub>H</sub>) **复位值: 62<sub>H</sub>**  
**RMAP:0, PAGE:X**

7	6	5	4	3	2	1	0
MPS		X_USIGN	ST_MODE	ROTVEC	MODE		ST
rw		rw	rw	rw	rw		rwh

符号	位	类型	功能描述
ST	0	rwh	<b>启动计算</b> 如果 ST_MODE = 1，置位 ST 启动 CORDIC 计算。只有在 BSY 未置位的情况下，该位才有效。对该寄存器进行写操作时，可同时设置该位和其它位。开始计算时，该位由硬件清零。

## CORDIC 协处理器

符号	位	类型	功能描述
<b>MODE</b>	2:1	rw	<b>操作模式</b> 00 线性函数模式 01 圆函数模式（缺省模式） 10 保留 11 双曲函数模式
<b>ROTVEC</b>	3	rw	<b>旋转向量模式选择</b> 0 向量模式（缺省模式） 1 旋转模式
<b>ST_MODE</b>	4	rw	<b>启动方式</b> 0 对 X 低字节寄存器 CD_CORDXL 进行写操作之后自动启动计算（缺省方式） 1 只有当位 ST 置位后才启动计算
<b>X_USIGN</b>	5	rw	<b>圆函数向量模式下 X 结果数据格式</b> DMAP = 0, 读取 X 结果数据时, X 数据格式如下: 0 有符号数, 2 补码 1 无符号数（缺省） 该位置位时, X 结果数据的 MSB 位被当作数据位、而非符号位来处理。 <i>注： 只有在圆函数向量模式下该位才有效。在所有其它模式下, X 始终按作 2 补码数据处理。</i> <i>注： 在圆函数向量模式下 X_USIGN = 1 才有意义, 因为结果数据总是为正并且总是大于初始值。</i>
<b>MPS</b>	7:6	rw	<b>X 和 Y 幅值调节因子</b> 在计算的最后一次迭代之后, X 和 Y 的计算值除以该因子, 产生最终结果数据。 正确设置 MPS, 对于避免相应内核数据寄存器的结果溢出是非常重要的。 00 除以 1 01 除以 2（缺省） 10 除以 4 11 保留, 保存最近的 MPS 设置

CORDIC 协处理器

13.6.2 状态和数据控制寄存器

CD\_STATC 寄存器可位寻址，通常用于指示 CORDIC 协处理器的状态，该寄存器还包含数据控制位和中断控制位。

**CD\_STATC**  
**CORDIC 状态和数据控制寄存器** (A0<sub>H</sub>) 复位值: 00<sub>H</sub>  
**RMAP:0, PAGE:X**

7	6	5	4	3	2	1	0
KEEPZ	KEEPY	KEEPX	DMAP	INT_EN	EOC	ERROR	BSY
rw	rw	rw	rw	rw	rwh	rh	rh

符号	位	类型	功能描述
BSY	0	rh	<b>CORDIC 协处理器忙碌指示</b> 该位被置位时，指示计算正在进行。ST 位被置位一个时钟之后该标志被置位。计算结束时，该位被清零。
ERROR	1	rh	<b>错误标志</b> 若 X、Y 或 Z 的计算结果溢出，CORDIC 计算结束后该位置位。当启动一次新的 CORDIC 运算时该位被清零。
EOC	2	rwh	<b>计算结束标志</b> 当 CORDIC 计算完成、BSY 变为无效后该位被置位。 除非由软件清零，否则该位一直保持置位，直到读取 Z 结果数据（DMAP = 0）的低字节时，才被硬件自动清零。
INT_EN	3	rw	<b>中断使能</b> 该位置位将使能 CORDIC 协处理器中断。
DMAP	4	rw	<b>数据映射</b> 0 从内核数据寄存器读取（结果）数据（缺省） 1 从映射数据寄存器读取（初始）数据

CORDIC 协处理器

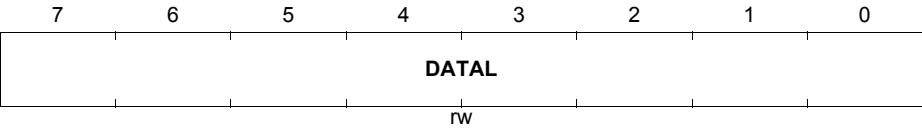
符号	位	类型	功能描述
KEEPX	5	rw	<p><b>X 的最终结果用作新计算的初始数据</b></p> <p>若该位置位，新的计算将使用前一次计算的结果作为新计算的初始数据。也就是说，在开始新的计算时，内核数据寄存器的内容将不被映射数据寄存器的内容改写。在最初计算加载 X 的初始值时，该位应该被清零。</p> <p><i>注： 映射数据寄存器和 KEEP 位无关，它始终保存最近写入的初始值，直到软件对其进行下一次改写。</i></p> <p><i>注： 如果在多步运算中 KEEPX 被置位，X 的最终结果精度会降低，不能保证具有<a href="#">章节 13.3.5</a>给出的精度。</i></p>
KEEPY	6	rw	<p><b>Y 的最终结果用作新计算的初始数据</b></p> <p>&lt; 见 KEEPX 的描述 &gt;</p>
KEEPZ	7	rw	<p><b>Z 的最终结果用作新计算的初始数据</b></p> <p>&lt; 见 KEEPX 的描述 &gt;</p>

13.6.3 数据寄存器

数据寄存器用于初始化 X、Y 和 Z 参数。可以读出 CORDIC 计算的结果数据（DMAP = 0）；也可读出映射寄存器中的初始数据（DMAP = 1）。无论读取时 DMAP 如何设置，这些数据寄存器始终保存最近写入的初始值，直到用户软件对其进行下一次写操作、或者复位操作。

CD\_CORDXL

CORDIC X 数据低字节 (BA<sub>H</sub>) 复位值: 00<sub>H</sub>  
RMAP:0, PAGE:X





CORDIC 协处理器

符号	位	类型	功能描述
DATAL	7:0	rw	<b>数据低字节</b> 对该字节进行写操作始终写入相应映射数据寄存器的低字节中。在 CORDIC 计算过程中可以写入新数据。  读取时： DMAP=0: 从内核数据字节读取结果数据 DMAP=1: 从映射数据字节读取初始值

CD\_CORDXH

CORDIC X 数据高字节  
RMAP:0, PAGE:X

(BB<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
DATAH							
rw							

符号	位	类型	功能描述
DATAH	7:0	rw	<b>数据高字节</b> 对该字节进行写操作始终写入相应映射数据寄存器的高字节中。在 CORDIC 计算过程中可以写入新数据。  读取时： DMAP=0: 从内核数据字节读取结果数据 DMAP=1: 从映射数据字节读取初始值

CD\_CORDYL

CORDIC Y 数据低字节  
RMAP:0, PAGE:X

(BC<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
DATAL							
rw							

CORDIC 协处理器

符号	位	类型	功能描述
DATAL	7:0	rw	<b>数据低字节</b> 对该字节进行写操作始终写入相应映射数据寄存器的低字节中。在 CORDIC 计算过程中可以写入新数据。  读取时： DMAP=0: 从内核数据字节读取结果数据 DMAP=1: 从映射数据字节读取初始值

CD\_CORDYH

CORDIC Y 数据高字节  
RMAP:0, PAGE:X

(BD<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
DATAH							
rw							

符号	位	类型	功能描述
DATAH	7:0	rw	<b>数据高字节</b> 对该字节进行写操作始终写入相应映射数据寄存器的高字节中。在 CORDIC 计算过程中可以写入新数据。  读取时： DMAP=0: 从内核数据字节读取结果数据 DMAP=1: 从映射数据字节读取初始值

CD\_CORDZL

CORDIC Z 数据低字节  
RMAP:0, PAGE:X

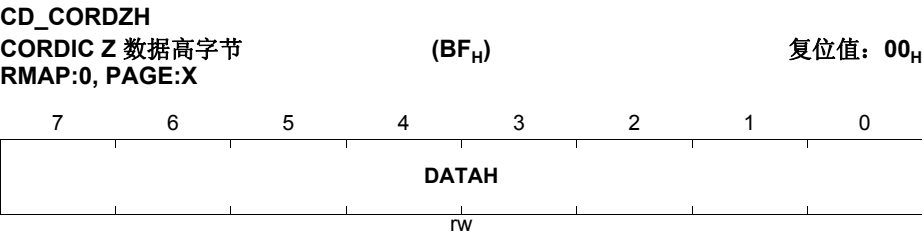
(BE<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
DATAL							
rw							

CORDIC 协处理器

符号	位	类型	功能描述
DATAL	7:0	rw	<p><b>数据低字节</b></p> <p>对该字节进行写操作始终写入相应映射数据寄存器的低字节中。在 CORDIC 计算过程中可以写入新数据。</p> <p>读取时： DMAP=0: 从内核数据字节读取结果数据 DMAP=1: 从映射数据字节读取初始值</p>



符号	位	类型	功能描述
DATAH	7:0	rw	<p><b>数据高字节</b></p> <p>对该字节进行写操作始终写入相应映射数据寄存器的高字节中。在 CORDIC 计算过程中可以写入新数据。</p> <p>读取时： DMAP=0: 从内核数据字节读取结果数据 DMAP=1: 从映射数据字节读取初始值</p>

# 14 定时器 T0 和 T1

## 14.1 概述

定时器 T0 和定时器 T1 均可用作定时器或计数器。用作定时器时，每个机器周期（即每两个输入时钟周期或两个 PCLK），T0 和 T1 加 1。用作计数器时，定时器 T0 和定时器 T1 对应的外部输入引脚 T0 或 T1 上每次发生 1-0 的跳变时（下降沿），T0 或 T1 加 1。定时器 T0 和 T1 在许多定时应用中非常有用，例如可用来测量事件之间的时间间隔、对事件计数、产生规律时间间隔的信号。特别是定时器 T1 还可用作片上串口的波特率发生器。

### 特性：

- 四种工作模式：
  - 模式 0：13 位定时器 / 计数器
  - 模式 1：16 位定时器 / 计数器
  - 模式 2：带有自动重载的 8 位定时器 / 计数器
  - 模式 3：两个 8 位定时器 / 计数器

## 14.2 系统信息

本节提供与定时器 T0 和 T1 相关的系统信息。

### 14.2.1 引脚

定时器 T0 和 T1 可用作事件计数器，对外部输入引脚 T0 和 T1 上 1-0 的跳变进行计数。可分别从 T0\_0 和 T0\_1/T1\_0 和 T1\_1 中选择定时器 T0/T1 的事件源。XC83x 中的定时器 T0 和定时器 T1 的引脚分配见 [表 14-1](#)。通过 SFR 位 MODPISEL2.T0IS 和 MODPISEL2.T1IS 选择相应引脚。

表 14-1 XC83x 中定时器 T0 和 T1 的引脚分配

引脚	功能	描述	由 ... 选择
P0.1	T0_0	定时器 T0 输入	MODPISEL2.T0IS = 0 <sub>B</sub>
P2.1	T0_1		MODPISEL2.T0IS = 1 <sub>B</sub>
P0.2	T1_0	定时器 T1 输入	MODPISEL2.T1IS = 0 <sub>B</sub>
P2.2	T1_1		MODPISEL2.T1IS = 1 <sub>B</sub>

访问 MODPISEL2 寄存器之前，必须编程设置 SCU\_PAGE 寄存器中的 PAGE 位域。

定时器 T0 和 T1

**MODISEL2**  
 外设输入选择寄存器 2 (F5<sub>H</sub>) 复位值: 00<sub>H</sub>  
 RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	T0IS	T1IS	T2IS	T2EXIS			
r	rw	rw	rw	rw			

符号	位	类型	描述
T1IS	5	rw	定时器 T1 输入选择 0 选择定时器 T1 输入 T1_0 1 选择定时器 T1 输入 T1_1
T0IS	6	rw	定时器 T0 输入选择 0 选择定时器 T0 输入 T0_0 1 选择定时器 T0 输入 T0_1
0	7	r	保留 读操作返回 0；应写入 0。

14.2.2 时钟配置

定时器 T0 和 T1 的工作频率 PCLK 可为 8 MHz 或 24 MHz。

14.2.3 中断事件和分配

表 14-2 列出定时器 T0 和 T1 的中断事件源以及对应的中断节点分配。

表 14-2 定时器 T0 和 T1 事件的中断节点分配

事件	中断节点使能位	中断节点标志位	向量地址
定时器 T0 溢出	IEN0.ET0	TCON.TF0	0B <sub>H</sub>
定时器 T1 溢出	IEN0.ET1	TCON.TF1	1B <sub>H</sub>

### 14.3 定时器基本操作

特殊功能寄存器 TCON 和 TMOD 控制两个定时器的操作。置位控制位 TCON.TRx 来使能相应的定时器（允许定时器工作）；通过输入选择位 TMOD 来选择定时器的输入来自内部系统时钟或者外部引脚。

注：本章中 'x'（如 TCON.TRx）代表 0 或 1。

每个定时器由两个 8 位寄存器，TLx（低位字节）和 THx（高位字节）组成，其复位缺省值为 00<sub>h</sub>。TCON.TRx 置位或清零不影响定时器寄存器的内容。

#### 定时器溢出

定时器溢出时，定时器溢出标志 TCON.TFx 置位，并在中断使能控制位 IEN0.ETx 置位时产生中断。进入中断服务程序时，溢出标志被自动清零。

定时器 T0 工作在模式 3 时，定时器 T1 的控制位 TR1、TF1 和 ET1 为 TH0 保留，具体内容请参见[章节 14.4.4](#)。

#### 外部控制

定时器的启动 / 停止不仅可以完全由软件控制，还可以由外部端口控制。使用外部端口控制时，必须首先设置 SFR EXICON0 旁路 EXINTx 边沿检测电路，信号直接送入内核。若定时器被使能（TCON.TRx = 1）且 TMOD.GATEx 置位，只有当内核外部中断信号 EXINTx = 1 时定时器才能工作，这样便于进行脉宽测量。不过，当定时器 T1 工作在模式 3 时外部控制不适用。

如果 TMOD.GATEx 被清零，定时器恢复完全由软件控制。

### 14.4 定时器工作模式

定时器 T0 和定时器 T1 完全兼容，均可设定为四种不同的工作模式，如表 14-3 所示。由寄存器 TMOD 中的位域 TxM 选择定时器的工作模式。

两个定时器在模式 0、1 和 2 时独立工作；在模式 3 时具有特定功能。

表 14-3 定时器 T0 和定时器 T1 工作模式

模式	功能描述
0	<b>13 位定时器 / 计数器</b> 该定时器实质上是一个带有 32 预分频的 8 位定时器 / 计数器。包含此工作模式仅仅是为了和 Intel 8048 器件兼容。
1	<b>16 位定时器 / 计数器</b> 定时器寄存器 TLx 和 THx 级联组成一个 16 位定时器 / 计数器。
2	<b>具有自动重载的 8 位定时器 / 计数器</b> 定时器寄存器 TLx 溢出时，THx 中用户定义的 8 位数据会自动重新载入 TLx。
3	<b>定时器 T0 作为两个独立的 8 位定时器 / 计数器工作</b> 定时器寄存器 TL0 和 TH0 用作两个独立的 8 位定时器 / 计数器。即使定时器 T1 被使能，它仍然停止计数、保持原先的计数值。

## 定时器 T0 和 T1

### 14.4.1 模式 0

在模式 0 下，定时器 T0 或定时器 T1 是带有 32 预分频的 8 位定时器 / 计数器，模式 0 的操作如图 14-1 所示。

该模式下，定时器寄存器配置为 13 位寄存器。当计数值从全“1”翻转为全“0”时溢出，定时器溢出标志  $TFx$  置位，该标志可请求中断。当  $TRx = 1$  且  $GATEx = 0$  或  $EXINTx = 1$  ( $GATEx = 1$  允许定时器由外部输入  $EXINTx$  控制，以便用于脉宽测量) 时，定时器计数输入使能。 $TRx$  是寄存器 TCON 中的控制位； $GATEx$  位于寄存器 TMOD 中。

13 位寄存器由  $THx$  的 8 位和  $TLx$  的低 5 位组成， $TLx$  的高 3 位不确定，应将其忽略。置位运行标志 ( $TRx$ ) 不对该寄存器清零。

模式 0 的操作对于定时器 T0 和定时器 T1 相同。

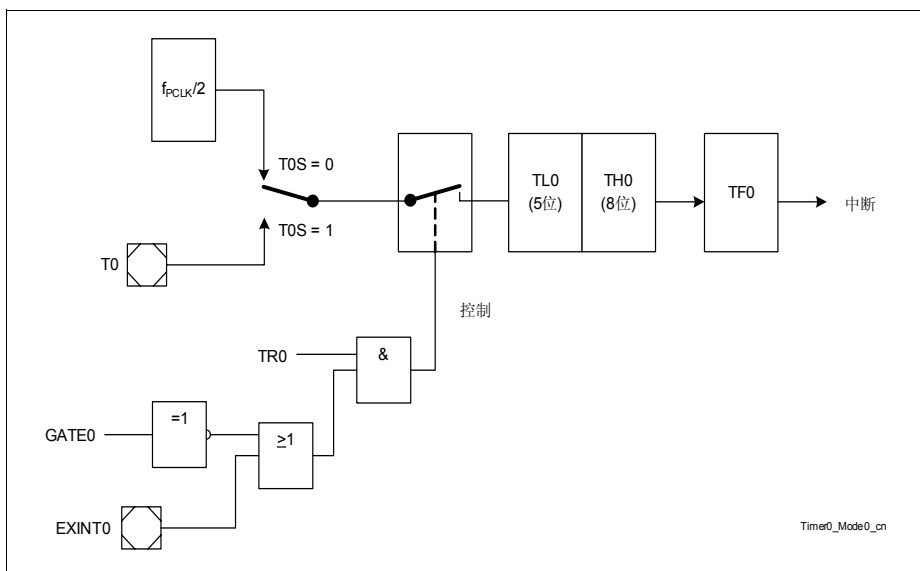


图 14-1 定时器 T0，模式 0：13 位定时器 / 计数器



## 定时器 T0 和 T1

### 14.4.2 模式 1

模式 1 和模式 0 的操作基本相同，唯一区别是模式 1 下定时器寄存器中的 16 位全部有用。模式 1 的操作如图 14-2 所示。

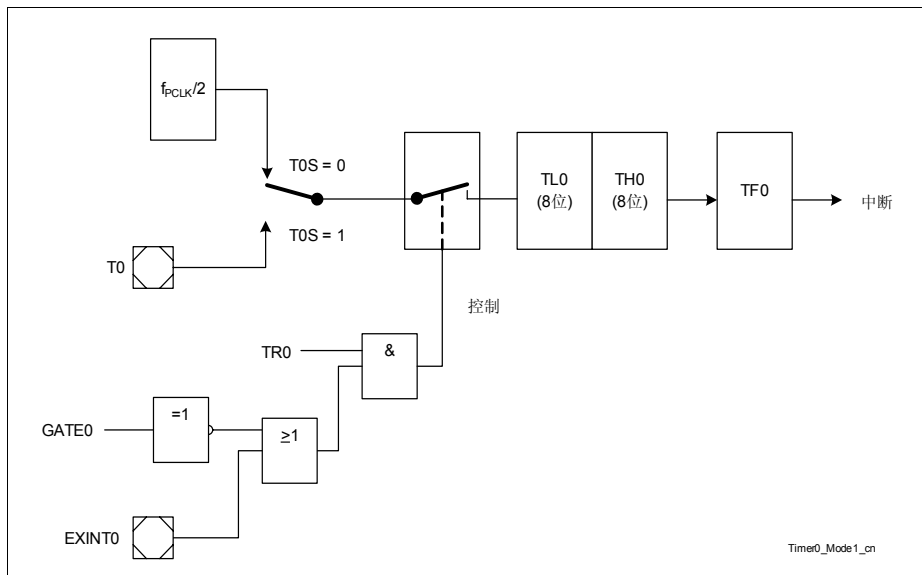


图 14-2 定时器 T0，模式 1：16 位定时器 / 计数器



## 定时器 T0 和 T1

### 14.4.4 模式 3

在模式 3 下，定时器 T0 和定时器 T1 的操作不同。定时器 T0 的 TL0 和 TH0 是两个独立的 8 位计数器；定时器 T1 只保持其原先计数值，效果如同设置 TR1 = 0。

定时器 T0 在模式 3 的逻辑结构如图 14-4 所示。TL0 使用定时器 T0 的控制位 GATE0、TR0 和 TF0，而 TH0 锁定为定时器功能（计数机器周期），占用定时器 T1 的控制位 TR1 和 TF1，TH0 溢出时将置位 TF1，并在 ET1 置位时产生中断。

模式 3 可用于额外需要一个 8 位定时器的场合。当定时器 T0 工作在模式 3 且 TR1 置位时，切换到模式 3 以外的任何其它模式可开启定时器 T1、切换到模式 3 时定时器 T1 关闭。

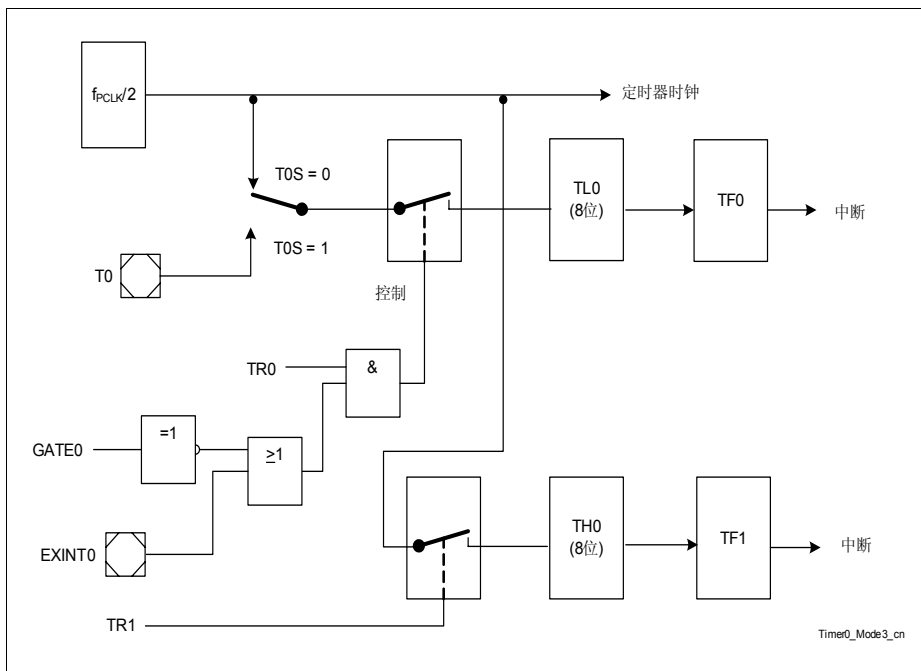


图 14-4 定时器 T0，模式 3：两个 8 位定时器 / 计数器

### 14.5 寄存器描述

通过 7 个 SFR 控制定时器 T0 和定时器 T1 的操作。可从标准（非映射）SFR 区和映射 SFR 区对其进行访问。

表 14-4 列出这些 SFR 的地址。

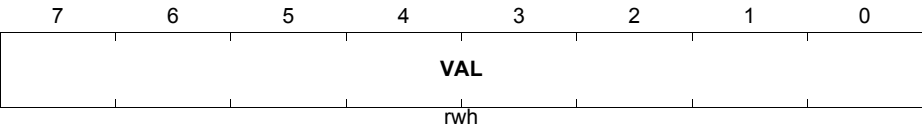
表 14-4 寄存器映射

地址	寄存器
88 <sub>H</sub>	TCON
89 <sub>H</sub>	TMOD
8A <sub>H</sub>	TL0
8B <sub>H</sub>	TL1
8C <sub>H</sub>	TH0
8D <sub>H</sub>	TH1
A8 <sub>H</sub>	IEN0

#### 14.5.1 定时器 T0 和 T1 寄存器

根据选择的工作模式，定时器 T0 和定时器 T1 的低位字节（TL0，TL1）和高位字节寄存器（TH0，TH1）可组合成一个定时器。寄存器 TCON 控制定时器 T0 和定时器 T1 的操作。寄存器 TMOD 中包含定时器 T0 和定时器 T1 的工作模式选择控制位。IEN0 寄存器中包含定时器 T0 和定时器 T1 的中断使能控制位。

**TLx (x = 0 - 1)**  
 定时器 x，低位字节 (8A<sub>H</sub> + x) 复位值：00<sub>H</sub>  
**RMAP: X, PAGE: X**



符号	位	类型	描述
VAL	[7:0]	rwh	定时器 T0/T1 低位字节寄存器 00 <sub>B</sub> TLx 保存 5 位预分频值 01 <sub>B</sub> TLx 保存 16 位定时器值的低 8 位 10 <sub>B</sub> TLx 保存 8 位定时器值 11 <sub>B</sub> TL0 保存 8 位定时器值；未使用 TL1。

# 定时器 T0 和 T1

THx (x = 0 - 1)

定时器 x, 高位字节

(8C<sub>H</sub> + x)

复位值: 00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
VAL							
rwh							

符号	位	类型	描述
VAL	[7:0]	rwh	定时器 T0/1 高位字节寄存器 00 <sub>B</sub> THx 保存 8 位定时器值 01 <sub>B</sub> THx 保存 16 位定时器值的高 8 位 10 <sub>B</sub> THx 保存 8 位重载值 11 <sub>B</sub> TH0 保存 8 位定时器值; 未使用 TH1

TCON

定时器 T0/1 控制寄存器

(88<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
TF1	TR1	TF0	TR0	IE1	IT1	IE0	IT0
rwh	rw	rwh	rw	rwh	rw	rwh	rw

符号	位	类型	描述
TR0	4	rw	定时器 T0 运行控制 0 <sub>B</sub> 定时器停止 1 <sub>B</sub> 定时器运行
TF0	5	rwh	定时器 T0 溢出标志 定时器 T0 溢出时由硬件置位。 当 CPU 调用中断服务程序时溢出标志由硬件清零。

定时器 T0 和 T1

符号	位	类型	描述
TR1	6	rw	<b>定时器 T1 运行控制</b> 0 <sub>B</sub> 定时器停止 1 <sub>B</sub> 定时器运行 <i>注： 如果定时器 T0 工作在模式 3， T1 的运行控制位也影响 TH0。</i>
TF1	7	rwh	<b>定时器 T1 溢出标志</b> 定时器 T1 溢出时，由硬件置位 <sup>1)</sup> 当 CPU 调用中断服务程序时溢出标志由硬件清零。

1) 如果定时器 T0 工作在模式 3， TF1 改由 TH0 置位。

TMOD

定时器模式寄存器

(89<sub>H</sub>)

复位值： 00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
GATE1	T1S	T1M	GATE0	T0S	T0M		
rw	rw	rw	rw	rw	rw		

符号	位	类型	描述
T0M	[1:0]	rw	<b>模式选择位</b> 00 <sub>B</sub> 13 位定时器 (M8048 兼容模式) 01 <sub>B</sub> 16 位定时器 10 <sub>B</sub> 8 位自动重载定时器 11 <sub>B</sub> 定时器 T0 被分成两部分。 8 位定时器 TL0 由标准定时器 T0 的控制位控制； 8 位定时器 TH0 由标准定时器 T1 的控制位控制。 TL1 和 TH1 的内容保持不变 (定时器 T1 停止工作)。
T0S	2	rw	<b>T0 输入选择</b> 0 <sub>B</sub> 输入来自内部系统时钟 1 <sub>B</sub> 输入来自 T0 引脚

# 定时器 T0 和 T1

符号	位	类型	描述
<b>GATE0</b>	3	rw	<b>T0 门控标志</b> 0 <sub>B</sub> 仅在 TCON.TR0 = 1 时定时器 T0 才运行（软件控制）。 1 <sub>B</sub> 仅当 EXINT0 引脚 = 1（硬件控制）且 TCON.TR0 置位时，定时器 T0 才运行
<b>T1M</b>	[5:4]	rw	<b>模式选择位</b> 00 <sub>B</sub> 13 位定时器（M8048 兼容模式） 01 <sub>B</sub> 16 位定时器 10 <sub>B</sub> 8 位自动重载模式 11 <sub>B</sub> 定时器 T0 被分成两部分。8 位定时器 TL0 由标准定时器 T0 的控制位控制；8 位定时器 TH0 由标准定时器 T1 的控制位控制。TL1 和 TH1 的内容保持不变（定时器 T1 停止工作）。
<b>T1S</b>	6	rw	<b>T1 输入选择</b> 0 <sub>B</sub> 输入来自内部系统时钟 1 <sub>B</sub> 输入来自 T1 引脚
<b>GATE1</b>	7	rw	<b>T1 门控标志</b> 0 <sub>B</sub> 仅当 TCON.TR1 = 1 时定时器才运行（软件控制） 1 <sub>B</sub> 仅当 EXINT0 引脚 = 1（硬件控制）且 TCON.TR1 置位时，定时器才运行。

## IEN0

中断使能寄存器

(A8<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
<b>EA</b>	<b>0</b>	<b>ET2</b>	<b>ES</b>	<b>ET1</b>	<b>EX1</b>	<b>ET0</b>	<b>EX0</b>
rw	r	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>ET0</b>	1	rw	<b>T0 溢出中断使能</b> 0 <sub>B</sub> 禁止定时器 T0 中断 1 <sub>B</sub> 使能定时器 T0 中断

定时器 T0 和 T1

符号	位	类型	描述
ET1	3	rw	<p><b>T1 溢出中断使能</b></p> <p>0<sub>B</sub> 禁止定时器 T1 中断</p> <p>1<sub>B</sub> 使能定时器 T1 中断</p> <p><i>注：</i> 当定时器 T0 工作在模式 3，该中断指示定时器 T0 的 TH0 寄存器的溢出。</p>



# 15 定时器 T2

## 15.1 概述

定时器 T2 为 16 位通用定时器，其功能与 C501 产品家族中的定时器 T2 兼容。每种工作模式下，定时器 T2 可用作定时器或计数器。用作定时器时，定时器 T2 以 PCLK/12 的输入时钟（若预分频被禁止）计数；用作计数器时，对引脚 T2 上 1-0 的跳变计数。计数器模式下，定时器 T2 的最大计数精度为 PCLK/24（若预分频被禁止）。

### 特性

- 16 位自动重载模式
- 可选择递增或递减计数
- 单通道、16 位捕获模式

## 15.2 系统信息

本节提供与定时器 T2 相关的系统信息。

### 15.2.1 引脚

T2 可用作事件计数器，对外部输入引脚 T2 上 1-0 的跳变计数。外部输入可从四个源中选择：T2\_0、T2\_1、T2\_2 和 T2\_3。通过 SFR 位 MODPSEL2.T2IS 进行选择。另外，定时器 T2 的 T2EX 引脚有 8 个源可供选择。通过 MODPSEL2.T2EXIS 进行选择。这些源中的 4 个为外部引脚，其中 2 个可被来自 ADC 模块的超出范围 0（ORC0）事件和超出范围 1（ORC1）事件触发。

XC82x 的定时器 T2 引脚分配见表 15-1。

表 15-1 XC83x 中定时器 T2 引脚功能

引脚 / 源	功能	描述	由 ... 选择
P0.0	T2_0	定时器 T2 输入	MODPSEL2.T2IS = 00 <sub>B</sub>
P2.0	T2_1		MODPSEL2.T2IS = 01 <sub>B</sub>
P2.3	T2_2		MODPSEL2.T2IS = 10 <sub>B</sub>
P2.4	T2_3		MODPSEL2.T2IS = 11 <sub>B</sub>

定时器 T2

表 15-1 XC83x 中定时器 T2 引脚功能

引脚 / 源	功能	描述	由 ... 选择
P0.6	T2EX_0	定时器 T2 外部触发输入	MODPISEL2.T2EXIS = 000 <sub>B</sub>
P0.4	T2EX_1		MODPISEL2.T2EXIS = 001 <sub>B</sub>
P1.0	T2EX_2		MODPISEL2.T2EXIS = 010 <sub>B</sub>
P2.0	T2EX_3		MODPISEL2.T2EXIS = 011 <sub>B</sub>
ORC0 事件	T2EX_4		MODPISEL2.T2EXIS = 100 <sub>B</sub>
ORC1 事件	T2EX_5		MODPISEL2.T2EXIS = 101 <sub>B</sub>
P2.7	T2EX_6		MODPISEL2.T2EXIS = 110 <sub>B</sub>
P3.2	T2EX_7		MODPISEL2.T2EXIS = 111 <sub>B</sub>
P0.4	EXF2_0	定时器 T2 溢出标志	P0_ALTSEL0.P4 = 1 <sub>B</sub> P0_ALTSEL1.P4 = 1 <sub>B</sub> P0_ALTSEL2.P4 = 0 <sub>B</sub>
P3.1	EXF2_1		P3_ALTSEL0.P1 = 1 <sub>B</sub> P3_ALTSEL1.P1 = 1 <sub>B</sub>
P1.3	EXF2_2		P1_ALTSEL0.P3 = 1 <sub>B</sub> P1_ALTSEL1.P3 = 1 <sub>B</sub>
P0.5	EXF2_3		P0_ALTSEL0.P5 = 0 <sub>B</sub> P0_ALTSEL1.P5 = 1 <sub>B</sub> P0_ALTSEL2.P5 = 1 <sub>B</sub>

访问 MODPISEL2 寄存器之前，必须编程设置 SCU\_PAGE 寄存器中的 PAGE。

## MODPISEL2

外设输入选择寄存器 2

(F5<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	T0IS	T1IS	T2IS	T2EXIS			
r	rW	rW	rW	rW			

## 定时器 T2

符号	位	类型	描述
<b>T2EXIS</b>	[2:0]	rw	<b>定时器 T2 外部输入选择</b> 000 选择定时器 T2 输入 T2EX_0 001 选择定时器 T2 输入 T2EX_1 010 选择定时器 T2 输入 T2EX_2 011 选择定时器 T2 输入 T2EX_3 100 选择定时器 T2 输入 T2EX_4 101 选择定时器 T2 输入 T2EX_5 110 选择定时器 T2 输入 T2EX_6 111 选择定时器 T2 输入 T2EX_7 <i>注： T2EX_4 和 T2EX_5 分别由超出范围 0 事件和超出范围 1 事件触发。</i>
<b>T2IS</b>	[4:3]	rw	<b>定时器 T2 输入选择</b> 00 选择定时器 T2 输入 T2_0 01 选择定时器 T2 输入 T2_1 10 选择定时器 T2 输入 T2_2 11 选择定时器 T2 输入 T2_3
<b>0</b>	7	r	<b>保留</b> 读操作返回 0；应写入 0。

### 15.2.2 时钟配置

定时器 T2 工作频率 PCLK 为 8 MHz 或 24 MHz。

如果完全不需要定时器 T2 的功能，可通过门控关闭其时钟输入的方式禁止该模块，从而最大限度的降低功耗。可通过如下所述的寄存器 PMCON1 中的 T2\_DIS 来禁止该模块。

#### PMCON1

外设管理控制寄存器 1

(EF<sub>H</sub>)

复位值：FF<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	CDC_DIS	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
T2_DIS	3	rw	T2 禁止请求，高有效 0 T2 正常工作 1 请求禁止 T2（缺省设置）

### 15.2.3 中断事件和分配

表 15-2 列出定时器 T2 的中断事件源以及对应事件中断使能位和标志位。

表 15-2 定时器 T2 中断事件

事件	中断事件使能位	事件标志位
定时器 T2 溢出	T2_T2CON1.TF2EN	T2_T2CON.TF2
定时器 T2 外部事件	T2_T2CON1.EXF2EN	T2_T2CON.EXF2

表 15-3 给出每个定时器 T2 中断请求源的中断节点分配。

表 15-3 定时器 T2 事件的中断节点控制

事件	中断节点使能位	中断节点标志位	向量地址
定时器 T2 溢出	IEN0.ET2	—	2B <sub>H</sub>
定时器 T2 外部事件			

### 15.2.4 IP 互连

在XC83x中，T2EX\_4输入和 T2EX\_5输入可分别由来自ADC模块的超出范围0（ORC0）事件和超出范围 1（ORC1）事件触发，见表 15-4。发生超出范围事件时，ORCx 将产生一个送至 T2EX 输入的上升沿信号。为了检测 T2EX 输入上的上升沿，T2\_T2MODE 寄存器中的 EDGESEL 或 T2REGS 必须置 1（取决于所使用的引脚）。另外，T2EX\_4 或 T2EX\_5 输入必须通过 MODPISEL2 寄存器进行设置，见表 15-1。

表 15-4 定时器 T2 互连

连接至其它功能模块 / 信号	定时器 T2 功能 / 信号
超出范围 0 (o): ORC0	T2 外部触发 (i): T2EX_4
超出范围 1 (o): ORC1	T2 外部触发 (i): T2EX_5

### 15.2.5 模块挂起控制

当片上调试支持（OCDS）处于监控模式（MMCR2.MMODE = 1）且调试挂起信号有效（MMCR2.DSUSP = 1）时，XC83x 的 T2 模块中的定时器 / 计数器可根据寄存器

---

定时器 T2

MODSUSP 中的对应模块挂起位的设置被挂起。挂起时，由于其输入时钟被关闭，定时器停止计数。整个模块仍有时钟输入，因此仍可访问模块寄存器。请参考[章节 10.2.4](#) 中的寄存器 MODSUSP 的定义。

### 15.3 基本定时器操作

定时器 T2 可通过位 TR2 由硬件或者软件启动。可置位 TR2 由软件启动定时器 T2；若位 T2RHEN 置位，定时器 T2 可由硬件启动。位 T2REGS 定义引脚 T2EX 上的事件：由下降沿或上升沿来硬件置位运行位 TR2。只能由软件复位 TR2 停止定时器 T2 工作。

### 15.4 自动重载模式

寄存器 T2CON 中的 CP/RL2 置 0 时，选择自动重载模式。该模式下，定时器 T2 计数至溢出后，将一个 16 位的初始值重新装入定时器寄存器中，开始新一轮计数过程。置位寄存器 T2CON 的 TF2 表示计数溢出，同时向 CPU 发送中断请求（若中断被使能）。溢出标志 TF2 必须由软件清零。

根据寄存器 T2MOD 中 DCEN 控制位的设置，自动重载模式可进一步分为两种类型。

#### 15.4.1 禁止递增 / 递减计数

若 DCEN = 0，递增 / 递减计数选择被禁止，定时器只能递增计数。工作原理如图 15-1 所示。

若寄存器 T2CON 中的 EXEN2 = 0，一旦将 T2CON 寄存器中的 TR2 置位，定时器开始递增计数，计数至最大值  $FFFF_H$  后溢出并置位 TF2，将寄存器 RC2 中的 16 位重载值重新装入定时器寄存器。在溢出前该重载值已由软件设置。然后开始新一轮计数过程，定时器同上一轮计数过程一样，从重载值开始递增计数。

若 EXEN2 = 1，一旦将 TR2 置位，定时器开始递增计数至最大值  $FFFF_H$ 。计数溢出或输入引脚 T2EX 的负 / 正跳变（由寄存器 T2MOD 中的位 EDGESEL 选择）均会引起 16 位重载，将寄存器 RC2 的内容重新装入定时器寄存器。若由溢出引起重载，溢出标志 TF2 置位。若由引脚 T2EX 的负 / 正跳变引起重载，寄存器 T2CON 中的 EXF2 置位。若由 TCON1 寄存器中的相关使能位 EXF2EN/TF2EN 使能，这两种情况均会产生中断并送至 CPU，定时器进入下一轮计数过程。EXF2 标志和 TF2 相似，必须由软件清零。

如果位 T2RHEN 置位，由引脚 T2EX 上的第一个下降沿 / 上升沿启动定时器 T2，由位 T2REGS 选择跳变沿。如果位 EXEN2 置位，引脚 T2EX 上的下降沿 / 上升沿启动定时器 T2 的同时置位 EXF2。引脚 T2EX 上接下来的负 / 正跳变将触发重载，由位 EDGESEL 选择跳变沿。

*注：计数器模式下，如果由 T2EX 触发重载的同时检测到计数时钟 T2，先进行重载操作（重载操作比计数操作优先）。在接下来的 T2 计数时钟计数器值加 1。*







## 15.5 捕获模式

置位寄存器 T2CON 中的  $\overline{\text{CP/RL2}}$  和 EXEN2，进入 16 位捕获模式。该模式下，递减计数功能必须保持禁止。定时器是一个 16 位递增计数的定时器，计数至最大值  $\text{FFFF}_H$  后溢出，溢出后置位 TF2、并将  $0000_H$  重新装入定时器寄存器。置位 TF2 会向 CPU 发送中断请求。

此外，在引脚 T2EX 的下降沿 / 上升沿（由 T2MOD.EDGESEL 选择），将定时器寄存器（THL2）的值捕获到寄存器 RC2 中。每个 PCLK 时钟周期对外部输入采样。如果在一个 PCLK 周期采样值为低（高）电平、下一个 PCLK 周期采样值为高（低）电平，将识别到一次信号沿跳变。如果在计数器加 1 的同时检测到捕获信号，计数器先加 1 然后执行捕获操作，从而确保始终捕获到定时器寄存器的最新值。

如果 T2RHEN 置位，由引脚 T2EX 的第一个下降沿 / 上升沿启动定时器 T2，跳变沿选择由位 T2REGS 决定。如果位 EXEN2 置位，引脚 T2EX 的下降沿 / 上升沿（由位 EDGESEL 选择跳变沿）启动定时器 T2 的同时置位 EXF2。引脚 T2EX 上接下来的负 / 正跳变将触发捕获，由位 EDGESEL 选择跳变沿。

执行完捕获操作，EXF2 置位、可用来产生中断请求。定时器 T2 的捕获功能如图 15-3 所示。

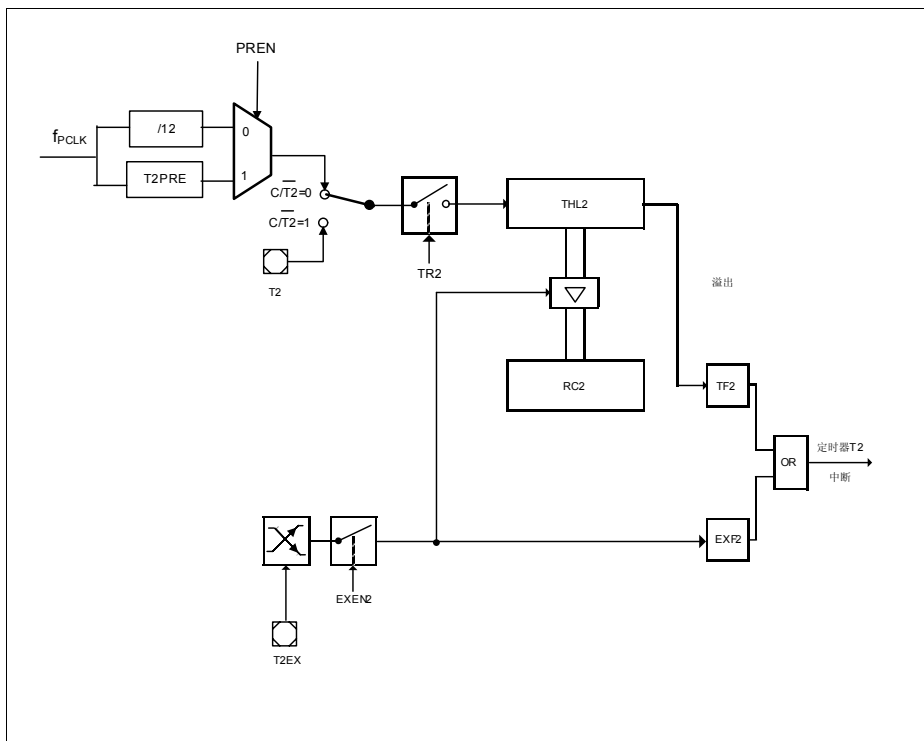


图 15-3 捕获模式

## 15.6 计数时钟

自动重载模式下的计数时钟由寄存器 T2CON 中的  $\overline{C/T2}$  位来选择。如果  $\overline{C/T2} = 0$ ，使用计数时钟  $f_{PCLK}/12$ （若预分频器被禁止）进行计数。

如果  $\overline{C/T2} = 1$ ，定时器 T2 用作计数器，对输入引脚 T2 上的 1-0 的跳变进行计数。计数器在连续两个 PCLK 时钟周期对引脚 T2 采样。如果在第一个 PCLK 周期采样为 1、下一个 PCLK 周期采样为 0，则计数器加 1。因此，输入引脚上的电平至少应该在 1 个时钟周期内保持稳定。

若 T2RHEN 置位，定时器 T2 可由引脚 T2EX 上的下降沿 / 上升沿启动工作，跳变沿选择由位 T2REGS 决定。

**注：** C501 兼容特性要求计数精度至少为 24 个时钟周期。

## 15.7 外部中断功能

当定时器 / 计数器功能被禁止时 ( $TR2 = 0$ )，只要定时器 T2 保持使能 ( $PMCON1.T2\_DIS = 0$ )，仍可通过引脚 T2EX 上的外部事件产生定时器 T2 中断。为了实现该功能，必须置位寄存器 T2CON 中的控制位 EXEN2。从而根据 CP/ RL2 的设置，使得 T2EX 上的任意跳变产生空的加载或捕获操作。

通过禁止定时器 / 计数器功能，T2EX 还可用来提供边沿触发的（上升沿或下降沿）的外部中断功能，位 EXF2 作为外部中断标志。

### 15.8 寄存器描述

下面各节中描述的所有定时器 T2 寄存器名称在本文档其它章节引用时，需要增加模块名前缀 “T2\_”，如 T2\_T2CON。

定时器 T2 SFR 位于标准（非映射）SFR 区域。[表 15-5](#) 列出这些地址。

**表 15-5 寄存器映射**

地址	寄存器
C0 <sub>H</sub>	T2CON
C1 <sub>H</sub>	T2MOD
C2 <sub>H</sub>	RC2L
C3 <sub>H</sub>	RC2H
C4 <sub>H</sub>	T2L
C5 <sub>H</sub>	T2H
C6 <sub>H</sub>	T2CON1

## 定时器 T2

### 15.8.1 模式寄存器

寄存器 T2MOD 用于配置定时器 T2 的各种工作模式。

#### T2\_T2MOD

定时器 T2 模式寄存器

(C1<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
T2REGS	T2RHEN	EDGESEL	PREN	T2PRE		DCEN	
rW	rW	rW	rW	rW		rW	

符号	位	类型	描述
DCEN	0	rW	<b>递增 / 递减计数器使能</b> 0 <sub>B</sub> 禁止递增 / 递减计数功能 1 <sub>B</sub> 使能递增/递减计数且由引脚T2EX控制计数方向（递增 = 1，递减 = 0）
T2PRE	[3:1]	rW	<b>定时器 T2 预分频选择</b> 选择定时器 T2 的输入时钟（来自外设时钟） 000 <sub>B</sub> $f_{T2} = f_{PCLK}$ 001 <sub>B</sub> $f_{T2} = f_{PCLK} / 2$ 010 <sub>B</sub> $f_{T2} = f_{PCLK} / 4$ 011 <sub>B</sub> $f_{T2} = f_{PCLK} / 8$ 100 <sub>B</sub> $f_{T2} = f_{PCLK} / 16$ 101 <sub>B</sub> $f_{T2} = f_{PCLK} / 32$ 110 <sub>B</sub> $f_{T2} = f_{PCLK} / 64$ 111 <sub>B</sub> $f_{T2} = f_{PCLK} / 128$
PREN	4	rW	<b>预分频使能</b> 0 <sub>B</sub> 禁止预分频功能，2 或 12 分频器生效 1 <sub>B</sub> 使能预分频功能（见 T2PRE 位），2 或 12 分频器被旁路。
EDGESEL	5	rW	<b>捕获模式 / 重载模式的边沿选择</b> 0 <sub>B</sub> 选择引脚 T2EX 的下降沿 1 <sub>B</sub> 选择引脚 T2EX 的上升沿
T2RHEN	6	rW	<b>定时器 T2 外部启动使能</b> 0 <sub>B</sub> 禁止定时器 T2 外部启动 1 <sub>B</sub> 使能定时器 T2 外部启动

## 定时器 T2

符号	位	类型	描述
<b>T2REGS</b>	7	rw	<b>定时器 T2 外部启动边沿选择</b> $0_B$ 选择引脚 T2EX 的下降沿 $1_B$ 选择引脚 T2EX 的上升沿

### 15.8.2 控制寄存器

控制寄存器 T2CON 控制定时器 T2 的工作模式以及中断产生，此外还包含中断状态标志。

#### T2\_T2CON

定时器 T2 控制寄存器

(C0<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
<b>TF2</b>	<b>EXF2</b>	<b>0</b>	<b>0</b>	<b>EXEN2</b>	<b>TR2</b>	<b>C_T2</b>	<b>CP_RL2</b>
rw	rw	r	r	rw	rw	rw	rw

符号	位	类型	描述
<b>CP_RL2</b>	0	rw	<b>捕获 / 重载模式选择</b> $0_B$ 计数器溢出或引脚 T2EX 发生负 / 正跳变时重载 (当 EXEN2 = 1)。 $1_B$ 引脚 T2EX 发生负 / 正跳变时, 捕获定时器 T2 数据寄存器的内容 (当 EXEN2 = 1)。由 EDGESEL 选择由引脚 T2EX 的负跳变还是正跳变触发操作。
<b>C_T2</b>	1	rw	<b>定时器或计数器选择</b> $0_B$ 选择定时器功能 $1_B$ 对引脚 T2 上的负跳变计数
<b>TR2</b>	2	rw	<b>定时器 T2 启动 / 停止控制</b> $0_B$ 停止定时器 T2 $1_B$ 启动定时器 T2
<b>EXEN2</b>	3	rw	<b>定时器 T2 外部使能控制</b> $0_B$ 禁止外部事件 $1_B$ 捕获 / 重载模式下使能外部事件

## 定时器 T2

符号	位	类型	描述
<b>EXF2</b>	6	rwh	<b>定时器 T2 外部事件标志</b> 捕获 / 重载模式下, 若 <b>EXEN2 = 1</b> , 在引脚 <b>T2EX</b> 上有负 / 正跳变时该位由硬件置位, 该位必须由软件清零。 <i>注: 自动重载模式下位 <b>DCEN = 1</b> 时, 不产生送入内核的中断请求。</i>
<b>TF2</b>	7	rwh	<b>定时器 T2 上溢 / 下溢标志</b> 定时器 T2 上溢或下溢时该标志置位。必须由软件清除该位。
<b>0</b>	[5:4]	r	<b>保留</b> 读操作返回 0; 应写入 0。

寄存器 T2CON1 用于使能外部中断和溢出中断。

### T2\_T2CON1

定时器 T2 控制寄存器 1

(C6<sub>H</sub>)

复位值: 03<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
			0			<b>TF2EN</b>	<b>EXF2EN</b>
			r			rw	rw

符号	位	类型	描述
<b>EXF2EN</b>	0	rw	<b>外部中断使能</b> 0 <sub>B</sub> 禁止外部中断 1 <sub>B</sub> 使能外部中断
<b>TF2EN</b>	1	rw	<b>上溢 / 下溢中断使能</b> 0 <sub>B</sub> 禁止上溢 / 下溢中断 1 <sub>B</sub> 使能上溢 / 下溢中断
<b>0</b>	[7:2]	r	<b>保留</b> 读操作返回 0; 应写入 0。

## 定时器 T2

### 15.8.3 定时器 T2 重载 / 捕获寄存器

根据所选的工作模式不同，RC2 寄存器中存放定时器寄存器溢出时的 16 位重载值；或存放捕获到的定时器当前计数值。

#### T2\_RC2L

定时器 T2 重载 / 捕获寄存器，低位字节 (C2<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
RC2							
rwh							

符号	位	类型	描述
RC2	[7:0]	rwh	<b>重载 / 捕获值 [7:0]</b> 如果 CP/RL2 = 0，定时器溢出时将该值装入到定时器寄存器。 如果 CP/RL2 = 1，当 EXEN2 = 1，引脚 T2EX 发生负 / 正跳变时，定时器的当前计数值被捕获到该寄存器。

#### T2\_RC2H

定时器 T2 重载 / 捕获寄存器，高位字节 (C3<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
RC2							
rwh							

符号	位	类型	描述
RC2	[7:0]	rwh	<b>捕获 / 重载值 [15:8]</b> 如果 CP/RL2 = 0，定时器溢出时将该值装入到定时器寄存器。 如果 CP/RL2 = 1，当 EXEN2 = 1，引脚 T2EX 发生负 / 正跳变时，定时器的当前计数值被捕获到该寄存器。



## 定时器 T2

### 15.8.4 定时器 T2 计数寄存器

寄存器 T2L 和 T2H 保存定时器 T2 的当前 16 位计数值。

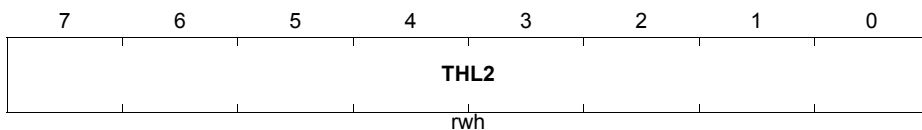
#### T2\_T2L

定时器 T2 寄存器，低位字节

(C4<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: X



符号	位	类型	描述
THL2	[7:0]	rwh	定时器 T2 计数值 [7:0] 这些位用来指示当前 16 位定时器值。

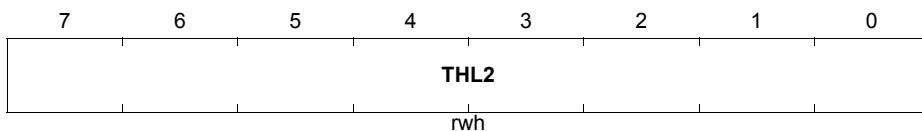
#### T2\_T2H

定时器 T2 寄存器，高位字节

(C5<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: X



符号	位	类型	描述
THL2	[7:0]	rwh	定时器 T2 计数值 [15:8] 这些位指示当前 16 位定时器值。

# 16 实时时钟

## 16.1 概述

XC83x 还提供一个实时时钟（RTC）外设模块，一旦启动，RTC 可独立工作，而与微控制器其余部分的状态无关。实时时钟的时钟源可为：由 32.768 kHz 振荡器引出脚提供的 32.768 kHz 片外振荡器、75 kHz 片内振荡器或来自 RTCCLK 引脚的外部时钟输入。32.768 kHz 频率可被分频用以记录从某个起始时刻开始的时间段，即为实时时钟。

### 特性

- 使用 32.768 kHz 晶振时钟的计时模式
- 使用 75 kHz 片内振荡器的周期唤醒模式
- 从掉电模式中唤醒

## 16.2 系统信息

本节提供与实时时钟相关的系统信息。

### 16.2.1 引脚

RTC 时钟源可以是片上或外部时钟源（通过 RTCCLK 引脚输入）。

XC83x 的 RTCCLK 引脚分配见表 16-1。

表 16-1 XC83x 中 RTC 的引脚功能

引脚	功能	描述	由 ... 选择
P3.1	RTCCLK	RTC 外部时钟输入	–
P3.1	XTAL3	32.768 kHz 片外振荡器输入	OSC_CON.XPD = 0 <sub>B</sub>
P3.0	XTAL4	32.768 kHz 片外振荡器输出	

### 16.2.2 中断事件和分配

表 16-2 列出 RTC 的中断事件源以及相应的中断使能位和标志位。

表 16-2 RTC 中断事件

事件	事件中中断使能位	事件中中断标志位
RTC 比较 / 唤醒	RTC_RTCON.ECRTC	RTC_RTCON.CFRTC
RTC 秒定时	RTC_RTCON.ESRTC	RTC_RTCON.SFRTC

表 16-3 给出每个 RTC 中断源的中断节点分配。

表 16-3      RTC 事件的中断节点控制

事件	中断节点使能位	中断节点标志位	向量地址
RTC 比较 / 唤醒	IEN1.EXM	—	4B <sub>H</sub>
RTC 秒定时			

### 16.2.3      模块挂起控制

当片上调试支持（OCDS）处于监控模式（MMCR2.MMODE = 1）且调试挂起信号有效（MMCR2.DSUSP = 1）时，可根据寄存器 MODSUSP 相关位的设置挂起 XC83x 中 RTC 模块的定时器 / 计数器，该寄存器的描述见[章节 10.2.4](#)。

被挂起时，因其计数器输入时钟被门控禁止，仅定时器停止计数。模块本身仍有时钟输入，因此仍可访问模块寄存器。

### 16.3      振荡器

32.768 kHz 片外振荡器和 75 kHz 片内振荡器均可用作 RTC 的时钟源。32.768 kHz 振荡器的输入来自片外石英晶振，该晶振和 32.768 kHz 振荡器引出脚的两个输入引脚 XTAL3 和 XTAL4 相连。75 kHz 振荡器除可用作 RTC 的输入时钟外，还可用作振荡器看门狗的时钟，用以监控 32.768 kHz 片外振荡器。

一旦上电，这些振荡器开始工作，而与微控制器的状态无关：即使器件进入空闲模式或掉电模式 2、3 和 4，振荡器仍继续工作。如有需要，32.768 kHz 振荡器引出脚也可被软件关闭。

这些振荡器以及整个 RTC，即使在电源为 2.5V - 5.5 V 的某些掉电模式期间仍能正常工作。

### 16.4      基本定时器操作

RTC 为递增计数的 41 位定时器。该模块由一组 4 到 6 个计数寄存器组成，这些寄存器统称为 CNT 寄存器，给出当前计数值或 RTC 的当前时间。启动 RTC 之前，可向 CNT 写入任意值。启动 RTC 时，该写入值被当作初始值。另一组寄存器 RTCCR，由 4 到 6 个寄存器组成，可用于中断产生，还可将器件从掉电模式中唤醒。捕获事件被触发时，RTCCR 寄存器还可用于保存捕获值。置位 RTCON 寄存器中的 RTCC 为 1，可启动实时时钟。该位用于使能 RTC 定时器的输入时钟。

### 16.5      实时时钟模式

在 XC83x 中，实时时钟可工作在四种模式。模式 0 和模式 4 为计时模式，用以维持实时系统时钟，以小时、分钟和秒来指示当前时间。在这两种模式下，使用 32.768 kHz 片外振荡器或直接以片外 1 Hz 时钟作为 RTC 的时钟输入。模式 1 为使用 41 位计数器的周期唤醒模式。该模式下，使用片内 75 kHz 振荡器作为时钟输入。模式 3 和模式 1 相似，以外部时钟作为 32 位计数器的时钟输入。在 XC83x 中，上电后缺省选择模式 1。用户需要确保只有当 RTC 停止工作时，才能切换其工作模式。

### 16.5.1 模式 0：使用 32.768 kHz 晶振时钟的计时模式

RTC 模式 0 具有计时功能。在该模式下，即使器件进入空闲和掉电模式 2 和 3，RTC 仍继续工作。

启动 RTC 之前，可向 RTC 的 CNT 寄存器写入用户期望的、以天、小时、分钟、秒、微妙表示的任意时间范围。当 RTCON 寄存器的 RTCC 位置 1 启动定时器时，写入值作为 RTC 定时器的初始值。该位还使能 RTC 定时器的输入时钟，如图 16-1 所示。当写入 CNT 的值超出规定的范围，比如向 CNT1 写入 70 秒或向 CNT2 写入 80 分钟，CNT 将被复位到 0。读取这些寄存器将返回 0。CNT 寄存器受到位保护机制（详见 SCU 章节的描述）的保护。只有在当停止模式下（RTCC = 0）保护机制被禁止时，才能更新初始计数值。

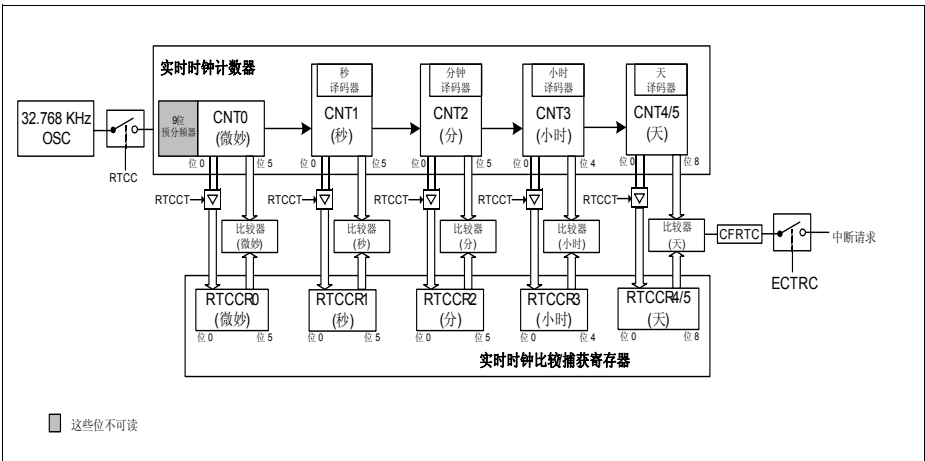


图 16-1 实时时钟模式 0

实时时钟的低 9 位（用作后缀 32 位计数器 CNT 的预分频器）设为初始值 00000000<sub>B</sub>。每个输入时钟周期定时器加 1。采用 32.768 kHz 的输入频率时，1 秒的时间相当于 15 位定时器计数溢出。在这种情况下，寄存器 CNT5-CNT0 中分别存放天、小时、分、秒和微秒的实时数值。寄存器 CNT0 中有 6 位用于存放实时时钟的微秒时段；寄存器 CNT1 和 CNT2 中各有 6 位分别用于存放实时时钟的秒和分钟时段，计数 60 次之后，秒和分钟计数器将复位到 0。器件处于工作模式时，每过 1 秒，标志位 SFRTC 置 1。若 RTCON 寄存器中的 ESRTC 置 1，同时可产生中断。寄存器 CNT3 存放实时时钟的小时值，计数 24 次之后寄存器复位到 0。寄存器 CNT4 和 CNT5 用于存放实时时钟的天数值，它们在复位之前可计数到 1 年。1 年可能是 365 或 366 天。通过寄存器 RTCON1 中的位 RTYR 选择是否为闰年。

RTC 正常工作时，RTCCR 寄存器的内容将和 RTC 计数器（CNT）的内容进行比较。RTCCR 的值可由用户任意设置。若 ECTRC 置 1，在工作模式下，二者内容相等时将产生中断，寄存器 RTCON 中的位 CFRTC 将置位。当器件处于掉电模式 2 和 3 时，这将

## 实时时钟

产生一个从软件掉电状态唤醒的操作。可通过 **CFRTC** 标志监控实时时钟唤醒请求，但必须由用户软件清除该标志。在这种情况下，实时时钟不受影响、继续工作。此时的唤醒操作与通过 **EXINT0** 引脚从掉电模式唤醒的操作相似。

*注： 不带复位从掉电模式唤醒之后，不会产生唤醒中断，与 **ECRTC** 位的状态无关。不过唤醒后，**CFRTC** 标志必须置 1。*

*注： 建议用户在设置寄存器 **RTCCR** 之后读取该值以确保其有效。若该值超出规定的范围，将导致比较操作失败。*

如果满足下列条件，实时时钟的模式 0 可产生唤醒请求使 **XC83x** 退出掉电模式：

- **XC83x** 处于掉电模式 2 或 3
- 掉电模式被使能（**PMCON0** 寄存器中的位 **PD = 1**），且
- 已选择唤醒模式类型（**PMCON0** 中的位 **WKSEL**），
- 维持工作电源电平（包括压降情况在内）

通过置位 **RTCON** 寄存器中的 **RTCCT** 位为 1，能够触发捕获事件。2 个 CPU 时钟之后，捕获的 **CNT** 值将改写 **RTCCR** 寄存器先前的内容。一旦触发捕获事件，有必要更新实际比较值。在 **XC83x** 中，推荐用户触发捕获事件，以读取 **RTC** 计数值（**CNT**）。由于一次只能读取 8 位数据，当 **RTC** 运行时，有可能读取到错误的 32 位实时值。

当 **RTCC** 位设置为 0 时，实时时钟停止计数且 **CNT** 寄存器保存最新值。随后置位该位会从先前停止的计数值开始，启动一个新的计数过程。

*注： 当捕获事件和比较事件出现在同一个时钟周期内时，这两个事件能够同时发生。*

### 16.5.2 模式 1：使用 75 kHz 振荡器时钟的周期性唤醒模式

**图 16-2** 给出 **RTC** 模式 1 的框图，该模式为上电复位的缺省模式。该模式下，**RTC** 由一个 41 位通用定时器组成，75 kHz 振荡器为该定时器提供输入时钟。

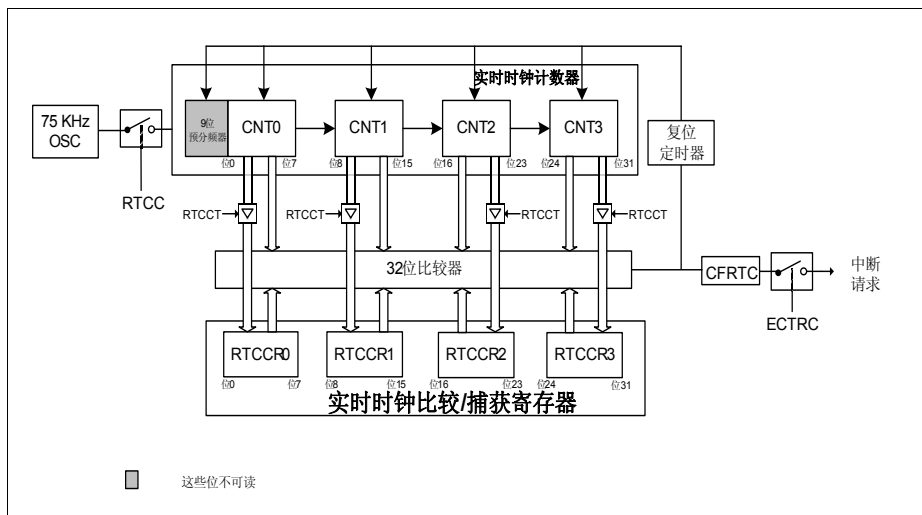


图 16-2 实时时钟模式 1

在掉电模式 4 下，可通过在 RTCCR 寄存器中设置计数值（该计数值等于唤醒时间的长度），每隔一段固定的时间（周期性）对系统唤醒一次。和模式 0 相似，定时器持续工作，直至 CNT 的计数值和 RTCCR 的值匹配，随后置位 CFRTC 标志。若 ECRTC 被使能，工作模式下将产生中断。与此同时，41 位定时器将复位并从 0 开始重新计数。此外，计数操作始终继续，达到 1 分钟、1 小时或 1 天时计数值不复位。模式 1 的其余操作和模式 0 类似，区别在于：该模式下，SFRTC 标志不置位。

### 16.5.3 模式 2：使用 1 Hz 外部时钟的计时模式

图 16-3 给出 RTC 模式 2 的框图。模式 2 和模式 0 的功能类似，区别在于：该模式下，9 位预分频器和 CNT0 被旁路。由 RTCCLK 引脚提供的 1 Hz 外部时钟用作 CNT1-CNT5 的输入时钟。和模式 0 相似，定时器持续工作，直至 CNT 的计数值（CNT1-CNT5）和 RTCCR 的值（RTCCR1-RTCCR5）匹配，随后置位 CFRTC 标志。若 ECRTC 被使能，将产生中断。在计数器开始工作之前，用户必须将 RTCCLK 引脚设置为输入模式并和 1 Hz 外部时钟相连。

在 XC83x 中，RTC 模式 2 不能用于将系统从掉电模式中唤醒。

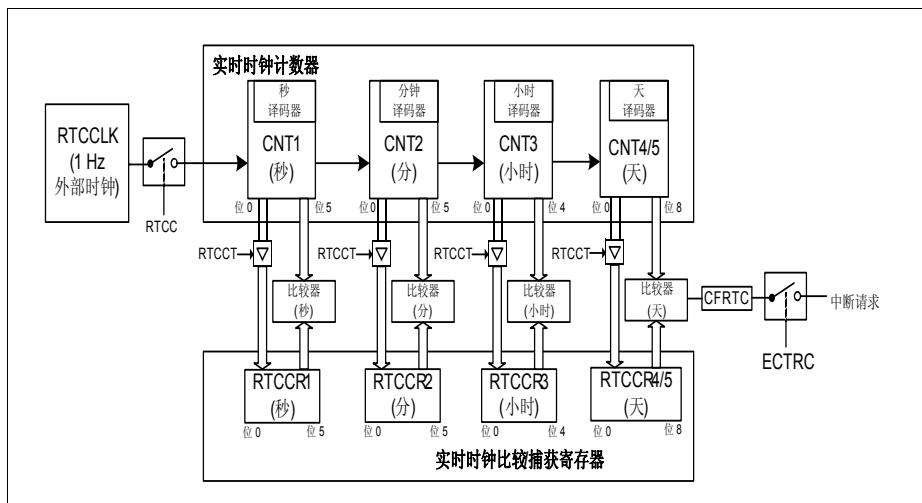


图 16-3 实时时钟模式 2

### 16.5.4 模式 3：使用外部时钟的定时器模式

图 16-4 给出 RTC 模式 3 的框图。该模式下，实时时钟由一个 32 位通用定时器组成。除了 9 位预分频器被旁路外，和模式 1 的功能相同。外部时钟通过 RTCCLK 引脚输入到定时器。在计数器开始工作之前，用户必须将 RTCCLK 引脚设置为输入模式并和外部时钟相连。

在 XC83x 中，RTC 模式 3 不能用于将系统从掉电模式中唤醒。

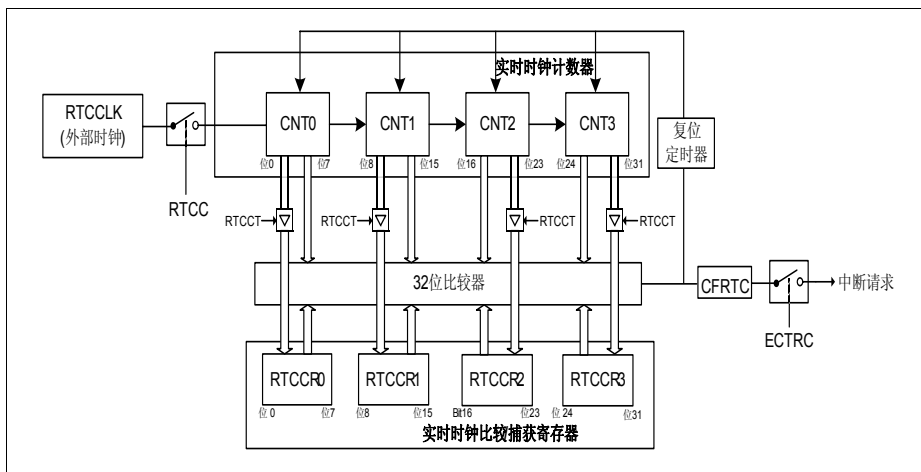


图 16-4 实时时钟模式 3

## 16.6 省电模式选择

一旦启动，实时时钟将持续计数，直到位 `RTCON.RTCC` 被清除。实时时钟不受 XC83x 空闲模式的影响，在掉电模式下也能继续计数（掉电模式 1 除外）。另外，若 `OSC_CON.XTAL2L` 或 `OSC_CON.75KOSC2L` 状态被置 1，实时时钟将不会自动停止。掉电模式 2、3 和 4 下，如果满足下述条件，实时时钟将继续运行：

- 实时时钟的时钟源可用，
- $V_{DDP} > 2.5\text{ V}$

软件掉电模式 1 下，实时时钟停止工作。



# 16.7 寄存器描述

共有 14 个 SFR 用于控制实时时钟的操作。可通过标准（非映射）SFR 区域访问这些寄存器，寄存器描述见下文。

表 16-4 列出这些 SFR 的地址。

表 16-4 寄存器映射

地址	寄存器
95 <sub>H</sub>	RTCON
96 <sub>H</sub>	RTCON1
E1 <sub>H</sub>	CNT0
E2 <sub>H</sub>	CNT1
E3 <sub>H</sub>	CNT2
E4 <sub>H</sub>	CNT3
E5 <sub>H</sub>	CNT4
E6 <sub>H</sub>	CNT5
E7 <sub>H</sub>	RTCCR0
E9 <sub>H</sub>	RTCCR1
EA <sub>H</sub>	RTCCR2
EB <sub>H</sub>	RTCCR3
EC <sub>H</sub>	RTCCR4
ED <sub>H</sub>	RTCCR5

## 16.7.1 实时时钟寄存器

### RTC\_RTCON

实时时钟控制寄存器

(95<sub>H</sub>)

复位值：02<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
SFRTC	CFRTC	ESRTC	ECRTC	RTCCT	RTM		RTCC
rwh	rwh	rw	rw	rwh	rw		rw

符号	位	类型	描述
RTCC	0	rw	<b>实时时钟启动 / 停止控制</b> 0 <sub>B</sub> 停止实时时钟操作 1 <sub>B</sub> 启动实时时钟操作
RTM	[2:1]	rw	<b>实时时钟模式</b> 00 <sub>B</sub> 模式 0；选择使用 32.768 kHz XTAL 时钟的计时醒式 01 <sub>B</sub> 模式 1；选择使用 75 kHz 振荡器时钟的周期唤醒模式 10 <sub>B</sub> 模式 2；选择使用 1 Hz 外部时钟的计时模式（外部时钟由 RTCCLK 引脚输入） 11 <sub>B</sub> 模式 3；选择使用外部时钟的定时器模式（外部时钟由 RTCCLK 引脚输入） <i>注：切换工作模式的同时会切换时钟源。切换工作模式时，RTC 必须停止工作。</i> <i>注：切换时钟源时（从模式 1 下的 75 kHz 振荡器时钟切换到模式 0 下的 32.768 kHz、或模式 2 和 3 下的外部时钟），晶振时钟输出或外部时钟源必须保持稳定。</i>
RTCCT	3	rwh	<b>实时时钟捕获事件触发</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 立即触发捕获事件 一旦置位，该位保持两个 PCLK 时钟周期，然后被硬件清除。 读取该位始终返回 0。
ECRTC	4	rw	<b>实时时钟比较中断使能</b> 0 <sub>B</sub> 禁止实时时钟比较中断 1 <sub>B</sub> 使能实时时钟比较中断

**实时时钟**

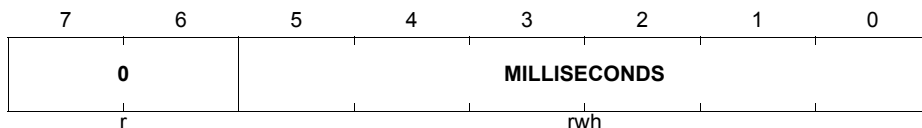
符号	位	类型	描述
<b>ESRTC</b>	5	rw	<b>实时时钟秒定时中断使能</b> $0_B$ 模式 0 下，禁止每秒产生实时时钟中断 $1_B$ 模式 0 下，使能每秒产生实时时钟中断 <i>注：每秒都产生中断的功能仅适用于模式 0 和模式 2。</i>
<b>CFRTC</b>	6	rwh	<b>实时时钟比较标志</b> 发生比较匹配时，该位被硬件置位，且只能由软件清除。 仅在 XC83x 处于掉电模式 2、3 或 4 时，才产生唤醒请求。
<b>SFRTC</b>	7	rwh	<b>实时时钟秒定时标志</b> 每秒该位被硬件置位，且只能被软件清除。该标志仅适用于模式 0 和模式 2。

**RTC\_RTCON1**
**实时时钟控制寄存器 1**
**(96<sub>H</sub>)**
**复位值：00<sub>H</sub>**
**RMAP: 0, PAGE: X**

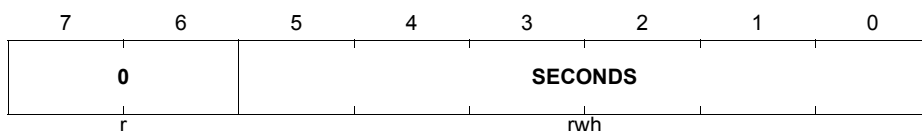
7	6	5	4	3	2	1	0
			0				<b>RTYR</b>
			r				rw

符号	位	类型	描述
<b>RTYR</b>	0	rw	<b>实时时钟年份类型</b> $0_B$ 非闰年；365 天 $1_B$ 闰年；366 天
<b>0</b>	[7:1]	r	<b>保留</b> 读取返回 0；应写入 0。

在模式 0 下，使用 CNT0 - CNT5 来表示时间，它们分别对应实时时钟的微秒、秒、分、小时和天。在模式 2 下，寄存器 CNT1-CNT5 可用，和模式 0 的功能相同。

**RTC\_CNT0 [ 模式 0 ]**
**计数时钟寄存器 0**
**(E1<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**


符号	位	类型	描述
<b>MILLISECONDS</b>	[5:0]	rwh	<b>实时时钟微秒值 [5:0]</b> 这些位代表当前定时器的微秒值。计数到 3F <sub>H</sub> 之后，该寄存器复位到 0。与此同时，CNT1.secs 计数加 1。
<b>0</b>	[7:6]	r	<b>保留</b> 读取返回 0；应写入 0。

**RTC\_CNT1 [ 模式 0 和模式 2 ]**
**计数时钟寄存器 1**
**(E2<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**


符号	位	类型	描述
<b>SECONDS</b>	[5:0]	rwh	<b>实时时钟秒值 [5:0]</b> 这些位代表当前定时器的秒值。模式 0 下，计数到 60 <sub>D</sub> 之后，该寄存器复位到 0。与此同时，CNT2.minutes 计数加 1。
<b>0</b>	[7:6]	r	<b>保留</b> 读取返回 0；应写入 0。

# 实时时钟

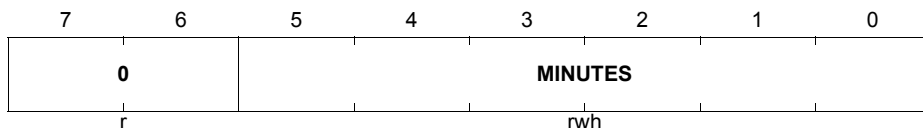
## RTC\_CNT2 [ 模式 0 和模式 2 ]

计数时钟寄存器 2

(E3<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X



符号	位	类型	描述
MINUTES	[5:0]	rwh	实时时钟分钟值 [5:0] 这些位代表当前定时器的分钟值。计数到 60 <sub>D</sub> 之后，该寄存器复位到 0。与此同时，CNT3.hours 计数加 1。
0	[7:6]	r	保留 读取返回 0；应写入 0。

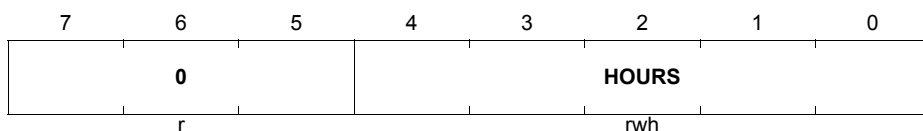
## RTC\_CNT3 [ 模式 0 和模式 2 ]

计数时钟寄存器 3

(E4<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X



符号	位	类型	描述
HOURS	[4:0]	rwh	实时时钟小时值 [4:0] 这些位代表当前定时器的小时值。计数到 24 <sub>D</sub> 之后，该寄存器复位到 0。与此同时，寄存器CNT4 和 CNT5 中的天数加 1。
0	[7:5]	r	保留 读取返回 0；应写入 0。

**RTC\_CNT4 [ 模式 0 和模式 2 ]**

计数时钟寄存器 4

(E5<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
DAYS							
rwh							

符号	位	类型	描述
DAYS	[4:0]	rwh	实时时钟天数 [7:0] 这些位代表当前定时器天数的低 8 位。

**RTC\_CNT5 [ 模式 0 和模式 2 ]**

计数时钟寄存器 5

(E6<sub>H</sub>)

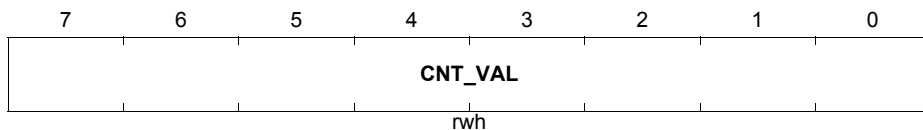
复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

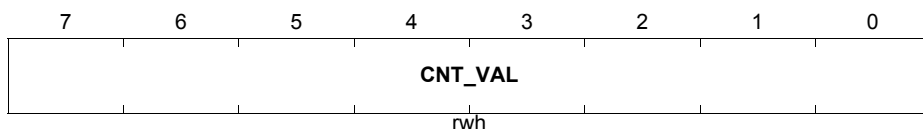
7	6	5	4	3	2	1	0
0							DAYS
r							rwh

符号	位	类型	描述
DAYS	0	rwh	实时时钟天数 [8] 这些位代表当前定时器天数的 MSB。
0	[7:1]	r	保留 读取返回 0；应写入 0。

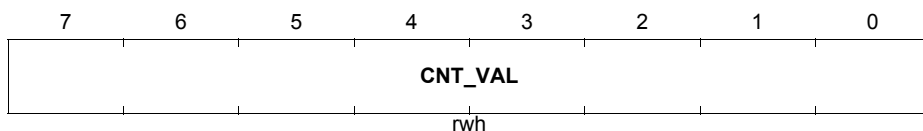
模式 1 下，寄存器 CNT0 - CNT3 代表 41 位实时时钟定时器的高 32 位；模式 3 下，CNT0 - CNT3 代表 32 位定时器。

**RTC\_CNT0 [ 模式 1 和模式 3]**
**计数时钟寄存器 0**
**(E1<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**


符号	位	类型	描述
<b>CNT_VAL</b>	[7:0]	rwh	<b>实时时钟计数值 [7:0]</b> 这些位代表当前实时时钟定时器的计数值 [7:0]。

**RTC\_CNT1 [ 模式 1 和模式 3]**
**计数时钟寄存器 1**
**(E2<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**


符号	位	类型	描述
<b>CNT_VAL</b>	[7:0]	rwh	<b>实时时钟计数值 [15:8]</b> 这些位代表当前实时时钟计数器的计数值 [15:8]。

**RTC\_CNT2 [ 模式 1 和模式 3]**
**计数时钟寄存器 2**
**(E3<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**


符号	位	类型	描述
<b>CNT_VAL</b>	[7:0]	rwh	<b>实时时钟计数值 [23:16]</b> 这些位代表当前实时时钟定时器的计数值 [23:16]。

**RTC\_CNT3 [ 模式 1 和模式 3]**
**计数时钟寄存器 3**
**(E4<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**

7	6	5	4	3	2	1	0
<b>CNT_VAL</b>							
rwh							

符号	位	类型	描述
<b>CNT_VAL</b>	[7:0]	rwh	<b>实时时钟计数值 [31:23]</b> 这些位代表当前实时时钟定时器的计数值 [31:23]。

在模式 0 下，实时时钟定时器的微秒、秒、分、小时和天数值和 RTCCR 寄存器中的值进行比较，从而可产生比较或唤醒事件。位 RTCCT 置 1 触发捕获事件时，RTCCR 寄存器的值将在 2 个 CPU 时钟周期之后被捕获到的 CNT 值改写。一旦触发捕获事件，有必要对实际比较值进行更新。RTCCR 寄存器由 6 个寄存器 RTCCR0 - RTCCR5 组成。

在模式 2 下，RTCCR0 寄存器不可用于比较操作。

**RTC\_RTCCR0 [ 模式 0 ]**
**实时时钟比较 / 捕获寄存器 0**
**(E7<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**

7	6	5	4	3	2	1	0
<b>0</b>	<b>CC_MSECS</b>						
r	rwh						

符号	位	类型	描述
<b>CC_MSECS</b>	[5:0]	rwh	<b>比较 / 捕获微秒值 [5:0]</b> 这些位代表比较 / 捕获操作下当前定时器的微秒值，取值范围为 0 <sub>H</sub> - 3F <sub>H</sub> 。



符号	位	类型	描述
<b>0</b>	[7:6]	r	<b>保留</b> 读取返回 0；应写入 0。

**RTC\_RTCCR1 [ 模式 0 和模式 2]**
**实时时钟比较 / 捕获寄存器 1**
**(E9<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**

7	6	5	4	3	2	1	0
<b>0</b>	<b>CC_SECONDS</b>						
r	rwh						

符号	位	类型	描述
<b>CC_SECONDS</b>	[5:0]	rwh	<b>比较 / 捕获秒值 [5:0]</b> 这些位代表比较 / 捕获操作下当前定时器的秒值，取值范围为 0 <sub>D</sub> - 59 <sub>D</sub> ，超出该范围的值被视为 0 <sub>D</sub> 。
<b>0</b>	[7:6]	r	<b>保留</b> 读取返回 0；应写入 0。

**RTC\_RTCCR2 [ 模式 0 和模式 2]**
**实时时钟比较 / 捕获寄存器 2**
**(EA<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**

7	6	5	4	3	2	1	0
<b>0</b>	<b>CC_MINUTES</b>						
r	rwh						

符号	位	类型	描述
<b>CC_MINUTES</b>	[5:0]	rwh	<b>比较 / 捕获分钟值 [5:0]</b> 这些位代表比较 / 捕获操作下当前定时器的分钟值，取值范围为 0 <sub>D</sub> - 59 <sub>D</sub> ，超出该范围的值被视为 0 <sub>D</sub> 。
<b>0</b>	[7:6]	r	<b>保留</b> 读取返回 0；应写入 0。

**RTC\_RTCCR3 [ 模式 0 和模式 2]**
**实时时钟比较 / 捕获寄存器 3**
**(EB<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**

7	6	5	4	3	2	1	0
0			CC_HOURS				
r			rwh				

符号	位	类型	描述
CC_HOURS	[4:0]	rwh	比较 / 捕获小时值 [4:0] 这些位代表比较 / 捕获操作下当前定时器的小时值，取值范围为 0 <sub>D</sub> - 23 <sub>D</sub> ，超出该范围的值被视为 0 <sub>D</sub> 。
0	[7:5]	r	保留 读取返回 0；应写入 0。

**RTC\_RTCCR4 [ 模式 0 和模式 2]**
**实时时钟比较 / 捕获寄存器 4**
**(EC<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**

7	6	5	4	3	2	1	0
CC_DAYS							
rwh							

符号	位	类型	描述
CC_DAYS	[7:0]	rwh	比较 / 捕获天数 [7:0] 这些位代表比较 / 捕获操作下当前定时器的天数的低 8 位。

**RTC\_RTCCR5 [ 模式 0 和模式 2]**

实时时钟比较 / 捕获寄存器 5

(ED<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0							CC_DAYS
r							rwh

符号	位	类型	描述
CC_DAYS	0	rwh	比较 / 捕获天数值 [8] 这些位代表比较 / 捕获操作下当前定时器的天数值的 MSB。
0	[7:1]	r	保留 读取返回 0；应写入 0。

在模式 1 和模式 3 下，4 个比较和捕获寄存器 RTCCR0 - RTCCR3 用于保存 32 位比较值，当 CNT 寄存器中的计数值与该比较值匹配时，将产生比较事件。当置位 RTCCT 触发捕获事件时，RTCCR 寄存器的内容将在 2 个 CPU 时钟周期之后被捕获到的 CNT 值改写。一旦触发捕获事件，有必要对实际比较值进行更新。

**RTC\_RTCCR0 [ 模式 1 和模式 3]**

实时时钟比较 / 捕获寄存器 0

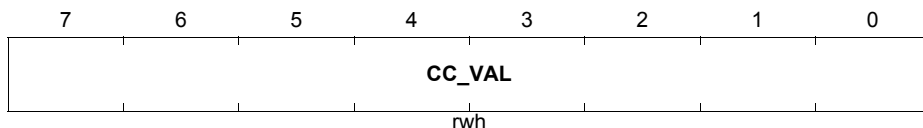
(E7<sub>H</sub>)

复位值: 00<sub>H</sub>

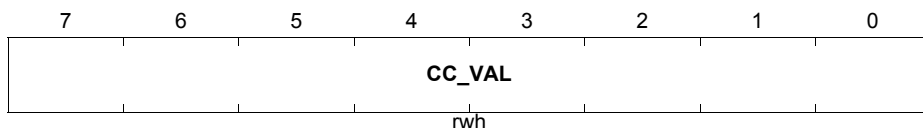
RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
CC_VAL							
rwh							

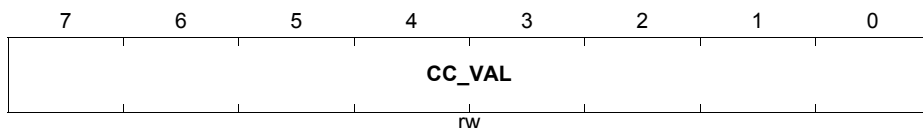
符号	位	类型	描述
CC_VAL	[7:0]	rwh	比较 / 捕获值 [7:0] 这些位代表 32 位比较 / 捕获值的 [7:0]，当前计数值与比较值匹配时，将产生比较中断。

**RTC\_RTCCR1 [ 模式 1 和模式 3]**
**实时时钟比较 / 捕获寄存器 1**
**(E9<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**


符号	位	类型	描述
CC_VAL	[7:0]	rwh	<b>比较 / 捕获值 [15:8]</b> 这些位代表 32 位比较 / 捕获值的 [15:8]。当前计数值与该值匹配时，将产生比较中断。

**RTC\_RTCCR2 [ 模式 1 和模式 3]**
**实时时钟比较 / 捕获寄存器 2**
**(EA<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**


符号	位	类型	描述
CC_VAL	[7:0]	rwh	<b>比较 / 捕获值 [23:16]</b> 这些位代表 32 位比较 / 捕获值的 [23:16]。当前计数器值与该值匹配时，将产生比较事件。

**RTC\_RTCCR3 [ 模式 1 和模式 3]**
**实时时钟比较 / 捕获寄存器 3**
**(EB<sub>H</sub>)**
**复位值: 00<sub>H</sub>**
**RMAP: 0, PAGE: X**


符号	位	类型	描述
CC_VAL	[7:0]	rwh	<b>比较 / 捕获值 [31:23]</b> 这些位代表 32 位比较 / 捕获值的 [31:23]。当前计数值与该值匹配时，将产生一个比较中断。

### 16.7.2 振荡器寄存器

XTAL 振荡器看门狗监控 32.768 kHz 片外振荡器的操作，检测振荡器频率是否低于阈值。寄存器 OSC\_CON 控制 32.768 kHz 振荡器关闭、控制 XTAL 振荡器看门狗的操作并监控其状态。该寄存器位于 SCU 中。

**OSC\_CON**  
**OSC 控制寄存器** (F4<sub>H</sub>) 复位值: 30<sub>H</sub>  
**RMAP: 0, PAGE: 1**

7	6	5	4	3	2	1	0
0	INTOSC_ST	XPD	XTAL2L	XTALOWD_RST	75KOSC_2L	48MOSC_2L	RCOWD_RST
r	rh	rw	rh	rwh	rh	rh	rwh

符号	位	类型	描述
XTALOWDRST	3	rwh	<b>32.768 kHz 片外振荡器看门狗复位</b> 置位该位将把 XTAL2L 状态标志复位到 1 并重新启动振荡器检测。该位将自动复位到 0，因此读取始终返回 0。 0 不起作用 1 将 XTAL2L 标志复位到 1 并重新启动 32.768 kHz 振荡器看门狗。
XTAL2L	4	rh	<b>32.768 kHz 片外振荡器频率过低标志</b> 振荡器看门狗监控 32.768 kHz 片外振荡器频率。 0 片外振荡器频率高于阈值。 1 片外振荡器频率低于阈值。
XPD	5	rw	<b>32.768 kHz XTAL 关闭控制</b> 0 32.768 kHz XTAL 未关闭。 1 32.768 kHz XTAL 关闭。

## 17 UART

### 17.1 概述

UART 是一个全双工异步接收 / 发送器，可以同时接收和发送数据。它还有接收缓存功能，第一个字节从接收寄存器中读出之前，可以开始接收第二个字节。但是如果第二个字节接收完毕时第一个字节仍未被读出，其中一个字节将会丢失。

#### UART 特性:

- 全双工异步模式
  - 8 位或 9 位数据帧，LSB 在先
  - 固定或可变波特率
- 接收缓存
- 多处理器通信
- 数据发送或接收完毕产生中断

### 17.2 系统信息

本节提供与 UART 相关的系统信息。

#### 17.2.1 引脚

模式 0 下（串行端口用作一个移位寄存器），数据从 RXD 移入，RXDO 移出，TXD 线用来提供移位时钟，外部器件根据该移位时钟移入和移出数据。在模式 1，2 和 3，端口用作 UART，通过 TXD 发送数据，从 RXD 接收数据。

*注：XC83x 不支持模式 0 操作，因此 RXDO 线不和任何通用 I/O 端口相连。*

UART I/O 引脚通常被分配用作通用 I/O 端口的其它功能。寄存器 MODPISEL1 中的位 URRIS 用来选择 RXD 输入功能。XC83x UART 引脚分配见表 17-1。

表 17-1 XC83x 中 UART 引脚功能

引脚	功能	描述	由 ... 选择
P0.5	RXD_0	UART 接收数据输入 0	MODPISEL1.URRIS = 000 <sub>B</sub>
P0.6	RXD_1	UART 接收数据输入 1	MODPISEL1.URRIS = 001 <sub>B</sub>
P1.0	RXD_2	UART 接收数据输入 2	MODPISEL1.URRIS = 010 <sub>B</sub>
P3.2	RXD_3	UART 接收数据输入 3	MODPISEL1.URRIS = 011 <sub>B</sub>
P3.1	RXD_4	UART 接收数据输入 4	MODPISEL1.URRIS = 100 <sub>B</sub>
P2.1	RXD_5	UART 接收数据输入 5	MODPISEL1.URRIS = 101 <sub>B</sub>
P2.7	RXD_6	UART 接收数据输入 6	MODPISEL1.URRIS = 110 <sub>B</sub>

表 17-1 XC83x 中 UART 引脚功能

引脚	功能	描述	由 ... 选择
P0.6	TXD_0	UART 发送数据输出 0	P0_ALTSEL0.P6 = 1 <sub>B</sub> P0_ALTSEL1.P6 = 1 <sub>B</sub> P0_ALTSEL2.P6 = 0 <sub>B</sub>
P1.0	TXD_1	UART 发送数据输出 1	P1_ALTSEL0.P0 = 1 <sub>B</sub> P1_ALTSEL1.P0 = 1 <sub>B</sub>
P1.1	TXD_2	UART 发送数据输出 2	P1_ALTSEL0.P1 = 1 <sub>B</sub> P1_ALTSEL1.P1 = 1 <sub>B</sub>
P3.2	TXD_3	UART 发送数据输出 3	P3_ALTSEL0.P2 = 1 <sub>B</sub> P3_ALTSEL1.P2 = 1 <sub>B</sub>
P0.5	TXD_4	UART 发送数据输出 4	P0_ALTSEL0.P5 = 0 <sub>B</sub> P0_ALTSEL1.P5 = 0 <sub>B</sub> P0_ALTSEL2.P5 = 1 <sub>B</sub>
P0.7	TXD_5	UART 发送数据输出 5	P0_ALTSEL0.P7 = 0 <sub>B</sub> P0_ALTSEL1.P7 = 1 <sub>B</sub> P0_ALTSEL2.P7 = 0 <sub>B</sub>

访问 MODPISEL1 寄存器之前，必须编程设置 SCU\_PAGE 寄存器中的 PAGE 位域。

### MODPISEL1

外设输入选择寄存器 1

(F4<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
EXINT2IS0	EXINT1IS	EXINT0IS			URRIS		
r	rw	rw			rw		

符号	位	类型	描述
URRIS	[2:0]	rw	<b>UART 接收输入选择</b> 000 选择 UART 接收输入 RXD_0 001 选择 UART 接收输入 RXD_1 010 选择 UART 接收输入 RXD_2 011 选择 UART 接收输入 RXD_3 100 选择 UART 接收输入 RXD_4 101 选择 UART 接收输入 RXD_5 110 选择 UART 接收输入 RXD_6 111 保留



### 17.2.2 时钟配置

UART 的工作频率 PCLK 为 8 MHz 或 24 MHz。

### 17.2.3 中断事件和分配

**表 17-2** 列出 UART 的中断事件源以及对应的事件中断使能位和标志位。

**表 17-2 UART 中断事件**

事件	事件中断使能位	事件标志位
收到数据	—	SCON.RI
发送数据		SCON.TI

**表 17-3** 给出每个 UART 中断源的中断节点分配。

**表 17-3 UART 事件的中断节点控制**

事件	中断节点使能位	中断节点标志位	向量地址
收到数据	IEN0.ES	—	23 <sub>H</sub>
发送数据			

## 17.3 UART 模式

UART 有四种工作模式。模式 0，UART 用作一个 8 位移位寄存器；模式 1，UART 是一个 8 位串行口；模式 2 和模式 3，UART 是一个 9 位串行口。模式 2 和模式 3 的唯一区别在于波特率不同，模式 2 的波特率固定不变，模式 3 的波特率可变。可根据专用波特率发生器的下溢速率、或定时器 T1 的溢出速率来设置可变的波特率。

设置 SM0 和 SM1 选择不同的工作模式，如 **表 17-4** 所示。

**表 17-4 UART 模式**

SM0	SM1	工作模式	波特率
0	0	模式 0: 8 位移位寄存器	$f_{PCLK}/2$
0	1	模式 1: 8 位移位 UART	可变
1	0	模式 2: 9 位移位 UART	$f_{PCLK}/64$ 或 $f_{PCLK}/32$
1	1	模式 3: 9 位移位 UART	可变

### 17.3.1 模式 0，8 位移位寄存器，波特率固定

模式 0 下，串行端口用作一个 8 位移位寄存器。数据从 RXD 移入，RXDO 移出，TXD 线用来提供移位时钟，外部器件根据该移位时钟移入和移出数据。

写入 SBUF 激活发送过程。一个机器周期之后，数据已被写入发送移位寄存器中，且第 9 位写入 1。在接下来的 7 个机器周期中，发送移位寄存器中的内容每右移一位，左边移入一个 0，当数据字节的最高有效位 MSB 移到输出位置时，移位寄存器中还包含一个 1，左边是一个全零序列。在置位 TI 之前，控制模块执行最后一次移位操作。

在  $REN = 1$  和  $RI = 0$  的条件下，启动接收操作。在接收周期的开始， $11111110_B$  写入接收移位寄存器。在随后的每一个机器周期，移位寄存器的内容左移一位，将（同一个机器周期内）RXD 线上的采样值从右边移入。当第一个字节中（初始字节  $11111110_B$ ）的 0 到达移位寄存器的最左端时，控制模块执行最后一次移位操作，加载 SBUF 并置位 RI。

输入时钟为  $f_{PCLK}$  时，传送波特率固定为  $f_{PCLK}/2$ ，即每个机器周期传送一位。

### 17.3.2 模式 1，8 位 UART，波特率可变

模式 1 中 UART 是 8 位串行口。数据帧由 1 位起始位（0）、8 位数据位和 1 位停止位（1）组成，以可变波特率从 TXD 端发送或从 RXD 端接收。

写入 SBUF 激活发送过程。数据送入发送移位寄存器中，第 9 位装入“1”（和模式 0 相同）。在 16 分频计数器下次翻转后的机器周期的 P1 时刻，起始位复制到 TXD 上，一个位时间后，数据有效。再经过一个位时间，数据开始右移输出，从左边填入 0。当 MSB 移至移位寄存器的输出位置时，控制器执行最后一次移位并置位 TI。

在 RXD 端检测到负跳变时启动接收（采样速率为波特率的 16 倍），16 分频计数器复位，并将  $11111111_B$  写入接收移位寄存器。如果检测到有效起始位（0）（基于“三中取二”采样），则将其移入寄存器，接着移入 8 位数据。如果接收到的第一位不是有效起始位，控制器将重新回去检测 RXD 的负跳变。当起始位移至寄存器的最左端时，控制器作最后一次移位，然后将 8 位数据装入 SBUF，将停止位装入 RB8（SCON.2），置位 RI。上述操作只有满足以下条件时才发生： $RI = 0$  并且  $SM2 = 0$ （见章节 17.4）或接收到的停止位 = 1。如果这些条件都不满足，所接收的数据就会丢失。

模式 1 发送 / 接收的相关时序如图 17-1 所示。

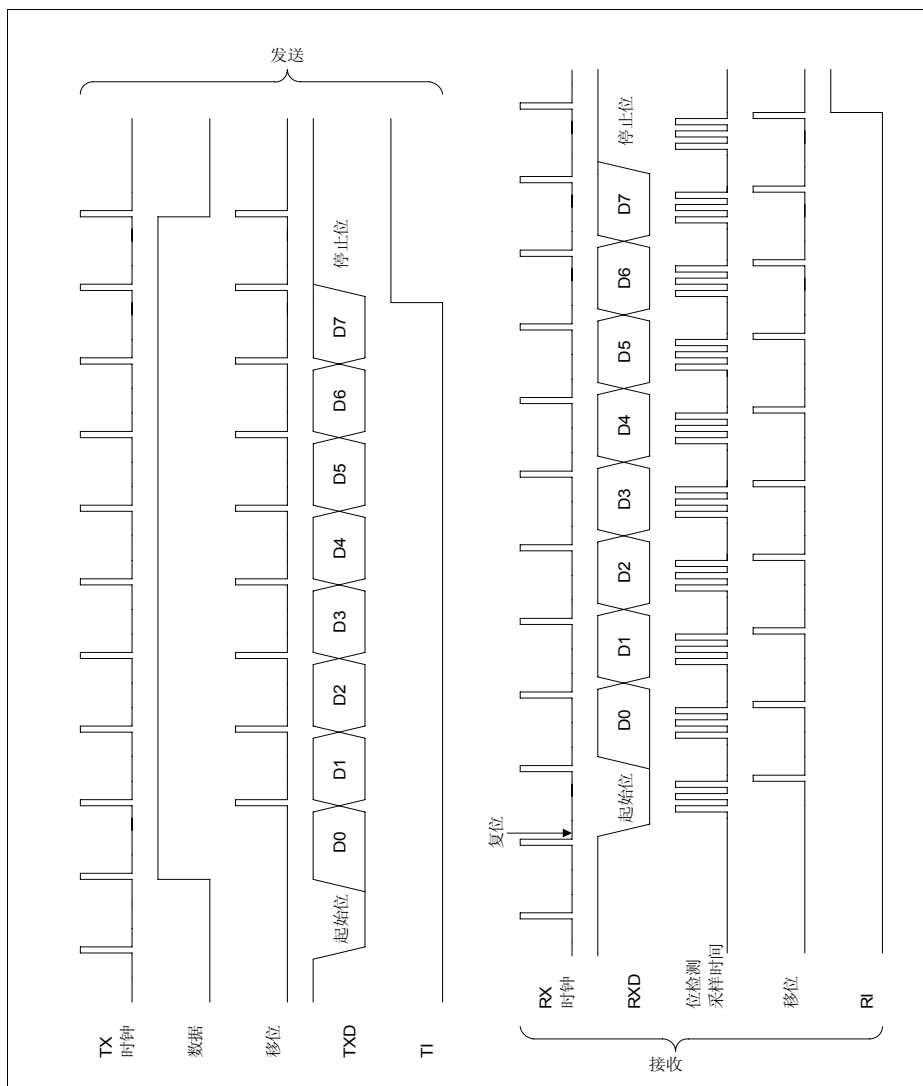


图 17-1 串行接口，模式 1，时序图

### 17.3.3 模式 2，9 位 UART，波特率固定

模式 2 中 UART 是 9 位串行口。由 1 位起始位（0）、8 位数据位、1 位可编程数据位（第 9 位）和一位停止位（1）组成的数据帧发送到 TXD 端或从 RXD 端接收，发送时第 9 位取自 TB8（SCON.3）；接收时，第 9 位存入 RB8（SCON.2）。

写入 SBUF 激活发送过程。数据被送入发送移位寄存器，TB8 被复制到第 9 位。在 16 分频计数器下次翻转后的机器周期的 P1 时刻，起始位被复制到 TXD 上，一个位时间之后数据有效。再经过一个位时间，数据开始右移输出。第一次移位，停止位（1）从左边移入，此后移入 0。当 TB8 移至输出位置时，控制器执行最后一次移位并置位 TI。

在 RXD 端检测到负跳变时启动接收（采样率为波特率的 16 倍），16 分频计数器复位，并将 1111 1111<sub>B</sub> 写入接收移位寄存器。如果检测到有效起始位（0）（基于“三中取二”），则将其移入寄存器，接着移入 8 位数据。如果接收的第一位不是有效起始位，控制器将重新回去检测 RXD 的负跳变。当起始位移至寄存器的最左端时，控制器执行最后一次移位，然后将 8 位数据装入 SBUF，第 9 位数据装入 RB8（SCON.2），置位 RI。上述操作只有满足以下条件时才发生：RI = 0 并且 SM2 = 0（见[章节 17.4](#)）或第 9 位 = 1。如果这些条件都不满足，所接收的数据就会丢失。

UART 的波特率为  $f_{PCLK}/32$  或  $f_{PCLK}/64$ ，波特率取决于寄存器 PCON（功率控制寄存器）中最高位（SMOD）的设置，该位用于倍频选择。

### 17.3.4 模式 3，9 位 UART，波特率可变

除了波特率可变之外，模式 3 和模式 2 完全相同。

所有模式下，数据的发送可由任何一条以 SBUF 为目的寄存器的指令启动；如果 REN = 1，由接收到有效起始位启动数据接收操作。

数据帧发送或接收完成时，串行接口产生中断请求。相应的中断请求标志分别为 TI 或 RI。如果不使用串行中断（串行中断被禁止），TI 和 RI 也可用来查询串行接口。

模式 2 和 3 发送 / 接收的相关时序如[图 17-2](#)所示。

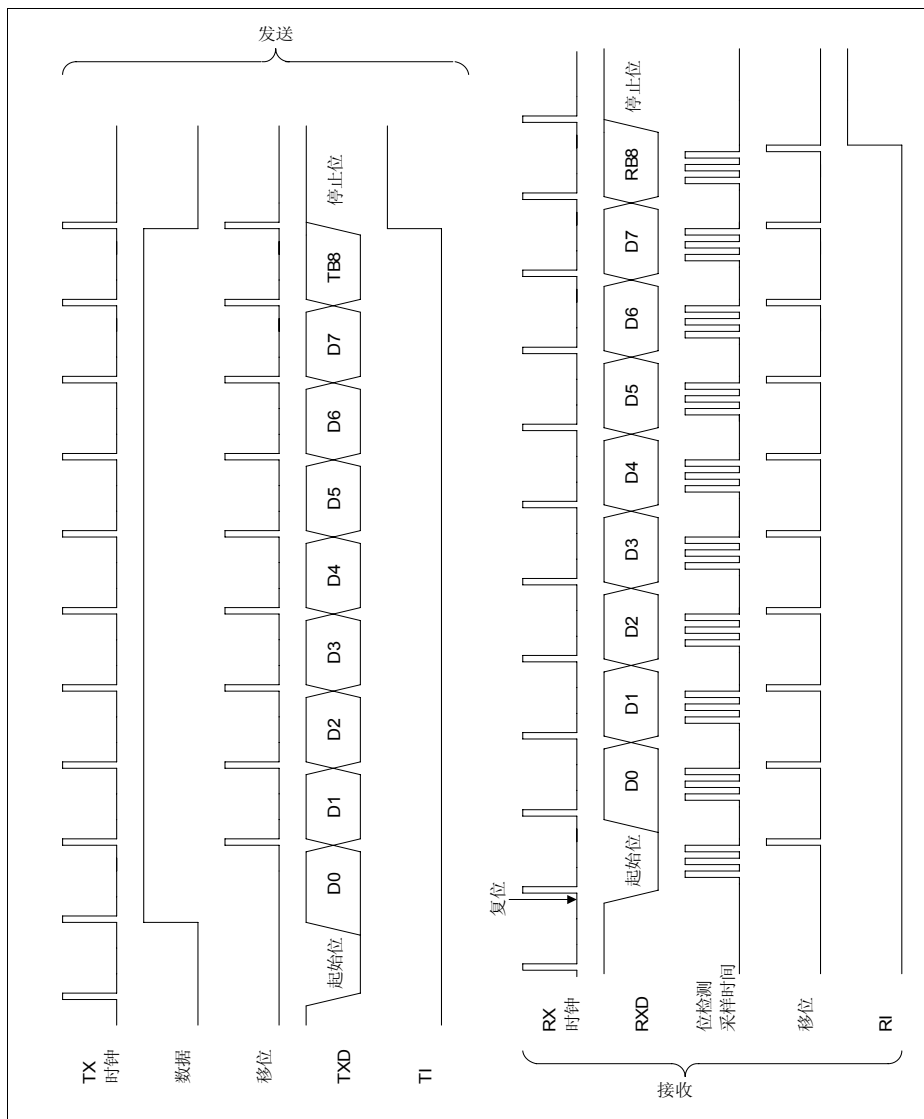


图 17-2 串行接口，模式 2 和 3，时序图

## 17.4 多处理器通信

模式 2 和模式 3 还可专门用于多处理器通信，此时地址字节的第 9 位 = 1、数据字节的第 9 位 = 0。在这些模式下接收 9 位数据，第 9 位存入 RB8。始终以停止位来结束通信。串口可编程设定为：接收到停止位后，只有当 RB8 = 1 串口中断才被激活。

可通过置位寄存器 SCON 中的位 SM2 使能该特性。多处理器系统中使用该特性的一种方法描述如下。

当主机需要向数台从机之一发送数据块时，主机首先发送一个地址字节对目标从机进行识别，地址字节和数据字节通过第 9 位区别：地址字节的第 9 位为 1；数据字节的第 9 位为 0。SM2 = 1 时，数据字节不会使从机产生中断，而地址字节将中断所有从机，因而每个从机都会检查接收到的字节，判断是否被寻址。被寻址的从机将 SM2 清零，准备接收主机发送的数据；未被寻址的从机的 SM2 保持为 1，对主机发送的数据不予理睬。

位 SM2 对模式 0 无影响。模式 1 中 SM2 可被用来检验停止位的有效性。模式 1 接收时，如果 SM2 = 1，只有接收到有效的停止位才可产生接收中断。

## 17.5 波特率产生

根据不同的操作模式，有几种产生串口波特率时钟的方式：

在模式 0 和模式 2 中，波特率固定，所以它们使用：

- 固定时钟（见章节 17.5.1）

在模式 1 和模式 3 中，可变波特率可由以下两种方式产生：

- UART 波特率发生器产生（见章节 17.5.2），或者
- 定时器 T1 产生（见章节 17.5.3）

由寄存器 LINST 中的位 BGS 选择可变波特率产生方式。

必须对“波特率时钟”和“波特率”加以区分。串行接口需要的时钟是波特率的 16 倍，用于内部同步。因此，UART 波特率发生器和定时器 T1 必须为串口提供“波特率时钟”，它被 16 分频即得到真正的“波特率”。缩写  $f_{PCLK}$  指输入时钟频率。

### 17.5.1 固定时钟

模式 0 和模式 2 的波特率固定：在模式 0 中波特率只能为  $f_{PCLK}/2$ ；在模式 2 中波特率可在  $f_{PCLK}/64$  或  $f_{PCLK}/32$  中选择，取决于 PCON 寄存器中的位 SMOD。

位 SMOD 用作倍频选择器，如公式 (17.1) 所示。如果 SMOD = 0（复位之后的值），波特率为输入时钟频率  $f_{PCLK}$  的 1/64；如果 SMOD = 1，波特率为  $f_{PCLK}$  的 1/32。

(17.1)

$$\text{模式2的波特率} = \frac{2^{\text{SMOD}}}{64} \times f_{PCLK}$$

### 17.5.2 UART 波特率发生器

UART 波特率发生器可为 UART 模式 1 和模式 3 产生可变的波特率。波特率发生器具有可编程设定的 11 位重载值、3 位预分频器和 5 位分数分频器。

波特率发生器的时钟为输入时钟  $f_{PCLK}$  经过预分频处理得到的  $f_{DIV}$ 。波特率定时器递减计数，可通过波特率运行控制位 BCON.R 启动或停止波特率定时器。每次定时器下溢都会为串行通道提供一个时钟脉冲。每次下溢时，将保存在重载寄存器 BG 的 BR\_VALUE 中的 11 位重载值加载到波特率定时器。两次下溢之间的时间取决于分数分频器的“n”值，由 BGL.FD\_SEL 从 32 个值中进行选择，定时器计数次数比 BR\_VALUE 规定的值多 1。由位 BCON.BRPRE 选择预分频器。

寄存器 BG 为双功能波特率发生器 / 重载寄存器。读取 BG 返回定时器的内容，写 BG（低位字节）始终更新重载寄存器。

仅当 BCON.R = 0 时，才能写访问 BG。BCON.R 复位之后的下一个指令周期用重载寄存器的内容自动重载定时器。当 BCON.R 置位时，任何写访问 BG 的操作都将被忽略。

波特率发生器的波特率取决于以下各位和寄存器值：

- 输入时钟  $f_{PCLK}$
- BCON.BRPRE 的值
- BG.FD\_SEL 的值
- 11 位重载值 BG.BR\_VALUE

图 17-3 给出波特率发生器的简化框图。

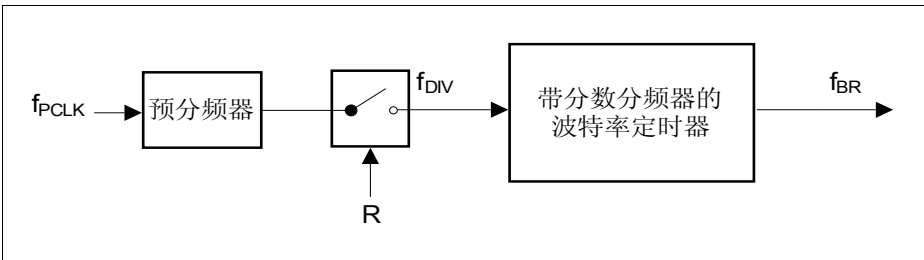


图 17-3 波特率发生器简化框图

根据下面的公式计算最终的波特率。

(17.2)

$$\text{波特率} = \frac{f_{PCLK}}{16 \times \text{PRE} \times \left( \text{BR\_VALUE} + \frac{n}{32} \right)}$$

PRE 的值（预分频器）由位域 BCON.BRPRE 选择。BR\_VALUE 代表重载值，被当作无符号 11 位整数。n/32 由分数分频器的位域 BG.FDSEL 进行选择。

可产生的最大波特率限定为  $f_{PCLK}/32$ 。因此，8 MHz 和 24 MHz 的模块时钟，对应的最大可用波特率分别为 0.25 MBaud 和 0.75 MBaud。

表 17-5 和表 17-6 列出各种常见波特率、对应的参数设置以及与理想波特率之间的偏差。

表 17-5 UART 典型波特率 ( $f_{PCLK} = 8 \text{ MHz}$ )

波特率 ( $f_{PCLK} = 8 \text{ MHz}$ )	PRE	重载值 (BR_VALUE)	分数分频值 (FD_NUM)	BG 寄存器 <sup>1)</sup>	偏差
115.2 kBaud	1 (BRPRE = 000)	4 (4 <sub>H</sub> )	11 (B <sub>H</sub> )	008B <sub>H</sub>	-0.08%
20 kBaud	1 (BRPRE = 000)	25 (19 <sub>H</sub> )	0 (0 <sub>H</sub> )	0320 <sub>H</sub>	0.00%
19.2 kBaud	1 (BRPRE = 000)	26 (1A <sub>H</sub> )	1 (1 <sub>H</sub> )	0341 <sub>H</sub>	+0.04%
9600 Baud	2 (BRPRE = 001)	26 (1A <sub>H</sub> )	1 (1 <sub>H</sub> )	0341 <sub>H</sub>	+0.04%



**表 17-5      UART 典型波特率 ( $f_{\text{PCLK}} = 8 \text{ MHz}$ )**

波特率 ( $f_{\text{PCLK}} = 8 \text{ MHz}$ )	PRE	重载值 (BR_VALUE)	分数分频值 (FD_NUM)	BG 寄存器 <sup>1)</sup>	偏差
4800 Baud	4 (BRPRE = 010)	26 (1A <sub>H</sub> )	1 (1 <sub>H</sub> )	0341 <sub>H</sub>	+0.04%
2400 Baud	8 (BRPRE = 011)	26 (1A <sub>H</sub> )	1 (1 <sub>H</sub> )	0341 <sub>H</sub>	+0.04%

1) 16 位 BG 寄存器的值由 11 位 BR\_VALUE 和 5 位 FD\_NUM 级联得到。

**表 17-6      UART 典型波特率 ( $f_{\text{PCLK}} = 24 \text{ MHz}$ )**

波特率 ( $f_{\text{PCLK}} = 24 \text{ MHz}$ )	PRE	重载值 (BR_VALUE)	分数分频值 (FD_NUM)	BG 寄存器 <sup>1)</sup>	偏差
115.2 kBaud	1 (BRPRE = 000)	13 (0D <sub>H</sub> )	1 (01 <sub>H</sub> )	01A1 <sub>H</sub>	-0.08%
20 kBaud	1 (BRPRE = 000)	75 (4B <sub>H</sub> )	0 (00 <sub>H</sub> )	0960 <sub>H</sub>	+0.00%
19.2 kBaud	1 (BRPRE = 000)	78 (4E <sub>H</sub> )	4 (04 <sub>H</sub> )	09C4 <sub>H</sub>	+0.00%
9600 Baud	2 (BRPRE = 001)	78 (4E <sub>H</sub> )	4 (04 <sub>H</sub> )	09C4 <sub>H</sub>	+0.00%
4800 Baud	4 (BRPRE = 010)	78 (4E <sub>H</sub> )	4 (04 <sub>H</sub> )	09C4 <sub>H</sub>	+0.00%
2400 Baud	8 (BRPRE = 011)	78 (4E <sub>H</sub> )	4 (04 <sub>H</sub> )	09C4 <sub>H</sub>	+0.00%

1) 16 位 BG 寄存器的值由 11 位 BR\_VALUE 和 5 位 FD\_NUM 级联得到。

### 17.5.3 定时器 1

UART 模块工作在模式 1 和模式 3 时，可用定时器 T1 产生可变的波特率。理论上，该定时器可工作在任何一种模式下。但实际上，定时器 T1 必须设置为自动重载模式（定时器 T1 模式 2），适当设置定时器的高位字节以产生所需的波特率。波特率由定时器 T1 的溢出速率和 PCON 寄存器中的 SMOD 取值共同决定，计算公式如下：

(17.3)

$$\text{模式1, 3的波特率} = \frac{2^{\text{SMOD}} \times f_{\text{PCLK}}}{32 \times 2 \times (256 - \text{TH1})}$$

对于给定波特率，定时器 T1 高位字节取值可由下式推导：

(17.4)

$$\text{TH1} = 256 - \frac{2^{\text{SMOD}} \times f_{\text{PCLK}}}{32 \times 2 \times \text{模式1, 3的波特率}}$$

**注：** 如果定时器 T0 工作在模式 3，定时器 T1 既不指示溢出也不会产生中断；当定时器 T0 使用其控制位和溢出标志时，定时器 T1 暂停。因此，UART 模块的波特率由定时器 T0 而不是由定时器 T1 决定。用户应当避免在模式 3 下使用定时器 T0 和定时器 T1 来产生波特率。

## 17.6 UART 中的 LIN 支持

UART 模块支持本地互连网络（LIN）协议的主机和从机操作。LIN 波特率检测特性由检测分隔和同步字节的硬件逻辑组成，通过定时器 T2 来检测 LIN 总线波特率，从而使 UART 模块与 LIN 的波特率同步、以进行数据发送和接收。

### 17.6.1 LIN 协议

LIN 是一种机动车内部网络互连的整体通信概念。该通信协议基于 SCI（UART）数据格式、单主机 / 多从机概念、无需固定的时间基准即可实现各节点时钟同步。LIN 具有吸引力的一大特性是：从节点无需通过石英晶振或陶瓷晶振即可实现自同步，大大降低了硬件平台的成本。因此，需要计算并返回每一个报文帧的波特率。

LIN 的报文帧结构如图 17-4 所示。由以下部分组成：

- 报文头，由分隔域（13 个位时间，低电平）、同步字节（55<sub>H</sub>）和 ID 标识符域组成
- 回应时间
- 数据字节（根据 UART 协议）
- 校验和

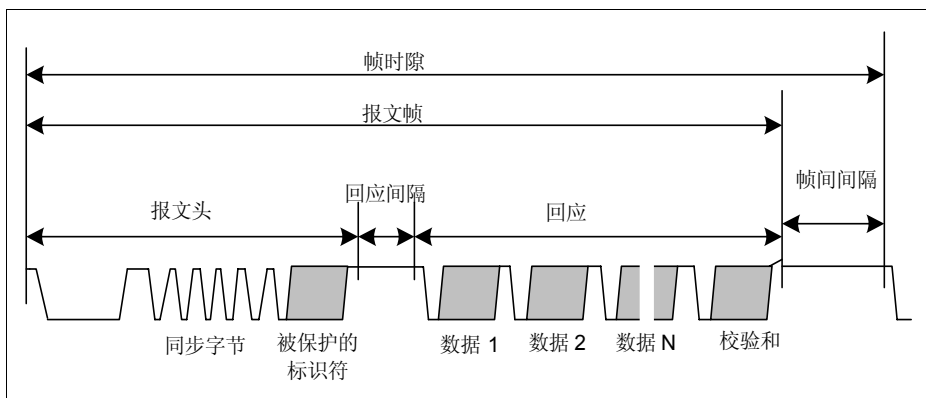
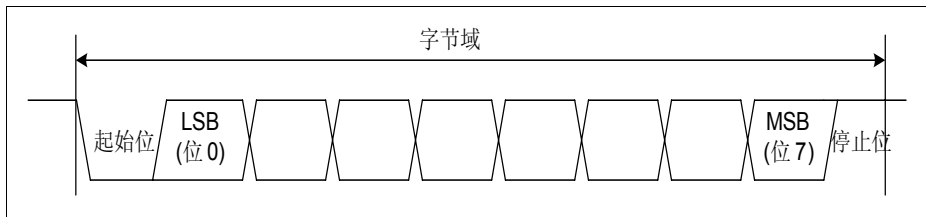


图 17-4 LIN 帧结构

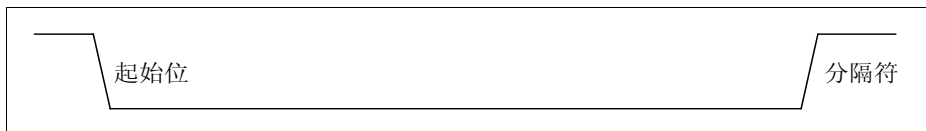
每个字节域串行发送，如图 17-5 所示。首先发送 LSB；最后发送 MSB。起始位编码为值“0”（显性值），停止位编码为值“1”（隐性值）。



**图 17-5 字节域结构**

分隔域用作新帧的起始信令。只有该域不遵循图 17-5 所示的字节域结构。分隔域始终由主机任务（主机模式）产生，它必须至少保持包括起始位在内的 13 位显性值，随后为分隔符，如图 17-6 所示。分隔符必须至少保持 1 个标称位时间。

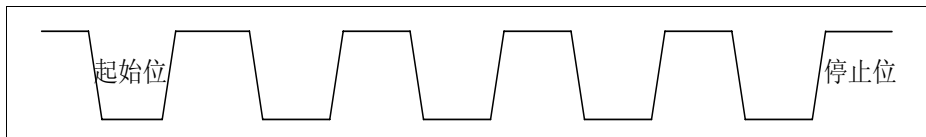
从节点将使用的分隔域检测阈值为 11 个标称位时间。



**图 17-6 分隔域**

同步字节是确定时间基准的特定序列，值为 55<sub>μ</sub>，如图 17-7 所示。

即使从机希望接收到字节域（假定字节域相互分开），它始终要检测分隔域 / 同步序列。如果在检测阶段接收到字节域，分隔域 / 同步序列检测将中止正在进行的数据传送，开始新一帧数据的处理。



**图 17-7 同步字节域**

当主机发送了正确的 ID 标识符，相应的目标从机将接收和发送数据：

1. 从机等待同步分隔
2. 从机和同步字节同步
3. 从机侦听标识符（ID）
4. 从机根据 ID 标识符来确定接收或发送数据，或不进行任何操作
5. 发送数据时，从机发送 2、4 或 8 个数据字节，随后跟着校验字节

## 17.6.2 LIN 报文头发送

LIN 报文头发送只适用于主模式。在 LIN 通信中，主机任务决定在何时、将哪帧传送到总线上；还决定由哪个从机来接收每帧数据。主机和从机之间所需的握手信息由主机的报文头提供。

报文头由分隔序列、同步序列及随后的标识符组成。这三个域中，只有分隔域不能按正常的 8 位 UART 数据格式传送。分隔域必须包含 13 位或更多的显性值以保证从机能正确同步。

在 LIN 通信中，开始传送被保护的标识符时要求从机已被同步。为了实现同步，每一帧以分隔域开始、随后紧跟同步字节。该序列是唯一的，它为任何从机任务提供了足够的信息来检测新帧的开始、并在标识符开始时使从机同步。

### 17.6.2.1 和主机自动同步

一旦进入 LIN 通信，建立主从机连接，串行通信端（主机）的传送速度（波特率）用户软件按照下述步骤进行自动同步：

步骤 1：初始化接口用于接收，初始化定时器用于波特率测量

步骤 2：等待来自主机的 LIN 帧

步骤 3：和主机波特率同步

步骤 4：进入主机请求帧或从机回应帧

在下一节，[章节 17.6.2.2](#)，给出微控制器建立 LIN 波特率检测需要的提示。

**注：** 每个主机请求报文头或从机回应报文头 LIN 帧都要进行重新同步和波特率建立操作。

### 17.6.2.2 分隔 / 同步域检测逻辑的初始化

LIN 波特率检测特性使用定时器 T2 来检测 LIN 总线波特率。初始化过程如下：

- 微控制器的串行端口设置为模式 1（8 位 UART，可变波特率）。
- 由位域 BCON.BGSEL 提供波特率范围。
- 翻转 BCON.BRDIS（在清零之前置 1）以初始化分隔 / 同步检测逻辑。
- 清除所有状态标志 LINST.BRK、LINST.EOFSYN 和 LINST.ERRSYN 到 0。
- 将 T2 设置为捕获模式，由引脚 T2EX 的下降沿触发捕获操作。位 T2MOD.EDGESEL 缺省设置为 0 且位 T2CON.CP/RL2 设置为 1。
- 使能定时器 T2 外部事件。T2CON.EXEN2 设置为 1。（当引脚 T2EX 出现负跳变时，置位 EXF2 标志）。
- 可由位域 T2MOD.T2PRE 设置  $f_{T2}$ 。

**注：** 推荐用户收到每个完整的 LIN 帧之后还翻转 BCON.BRDIS 位，以避免噪声环境中错误的分隔域检测（即 LIN 总线上的毛刺）。

### 17.6.2.3 波特率范围选择

分隔 / 同步位域检测逻辑支持的分隔域中的位的最大个数由 [公式 \(17.5\)](#) 定义：

(17.5)

$$\text{最大位数} = \text{波特率} \times \frac{4095}{\text{采样频率}}$$

采样频率由 [公式 \(17.6\)](#) 给出：

(17.6)

$$\text{采样频率} = \frac{\text{PCLK}}{8 \times 2^{\text{BGSEL}}}$$

如果分隔域中的位超过最大值，内部计数器将溢出，导致波特率检测错误。因此，对于需要检测的波特率范围，必须选择合适的 BGSEL 值。

不同 BGSEL 设置所定义的波特率范围见表 17-7。

表 17-7 对应不同输入频率时 BGSEL 位域定义

$f_{\text{PCLK}}$	BGSEL	可检测的波特率范围 $f_{\text{PCLK}}/(2184 \cdot 2^{\text{BGSEL}})$ 至 $f_{\text{PCLK}}/(72 \cdot 2^{\text{BGSEL}})$
24 MHz	00 <sub>B</sub>	11 kHz 至 333.3 kHz
	01 <sub>B</sub>	5.5 kHz 至 166.7 kHz
	10 <sub>B</sub>	2.8 kHz 至 83.3 kHz
	11 <sub>B</sub>	1.4 kHz 至 41.7 kHz
8 MHz	00 <sub>B</sub>	3.7 kHz 至 111.1 kHz
	01 <sub>B</sub>	1.8 kHz 至 55.6 kHz
	10 <sub>B</sub>	0.92 kHz 至 27.8 kHz
	11 <sub>B</sub>	0.46 kHz 至 13.9 kHz

每个 BGSEL 值支持一定范围的波特率检测。如果所使用的波特率范围落在所定义的范围之外，则可能不能正确检测波特率。

当  $f_{\text{PCLK}} = 24 \text{ MHz}$  时，可检测的波特率范围为 1.4 kHz 至 333.3 kHz。选择 BGSEL 值时，可参考以下范例：

- 如果波特率落在 1.4 kHz 到 2.8 kHz 之间，选择的 BGSEL 为 "11<sub>B</sub>"。
- 如果波特率落在 2.8 kHz 到 5.5 kHz 之间，选择的 BGSEL 为 "10<sub>B</sub>"。
- 如果波特率落在 5.5 kHz 到 11 kHz 之间，选择的 BGSEL 为 "01<sub>B</sub>"。
- 如果波特率落在 11 kHz 到 333.3 kHz 之间，选择的 BGSEL 值为 "00<sub>B</sub>"。如果波特率为 20 kHz，则可选择的 BGSEL 值为 "00<sub>B</sub>"，"01<sub>B</sub>"，"10<sub>B</sub>"，和 "11<sub>B</sub>"。然而，为了得到更高的检测精度，建议用户选择 "00<sub>B</sub>"。

当  $f_{\text{PCLK}} = 8 \text{ MHz}$  时，仍可检测波特率，此时可检测波特率范围为 0.46 kHz 至 111.1 kHz。

### 17.6.2.4 LIN 波特率检测

LIN 的波特率检测如图 17-8 所示，LIN 帧的报文头包括：

- 同步分隔域（13 个位时间，低电平）
- 同步字节（55<sub>H</sub>）
- 被保护 ID 域

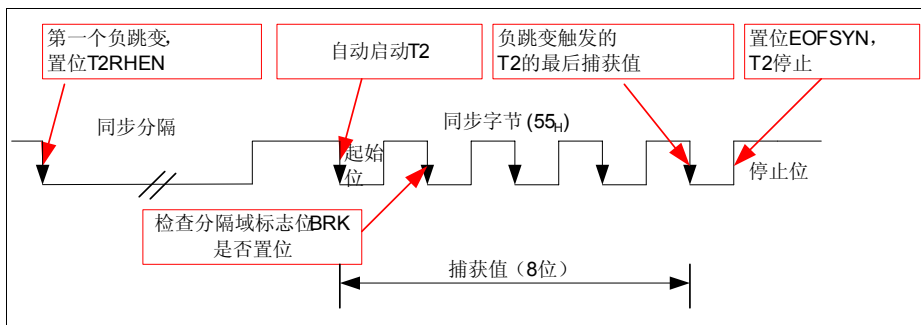


图 17-8 LIN 自动波特率检测

第 1 个下降沿到来时:

- 置位定时器 T2 外部启动使能位 (T2MOD.T2RHEH)。缺省选择引脚 T2EX 的下降沿触发定时器 T2 外部启动 (位 T2MOD.T2REGS 为 0)

第 2 个下降沿到来时:

- 硬件启动定时器 T2

第 3 个下降沿到来时:

- 定时器 T2 捕获同步字节 2 位的位时序
- 检查分隔域标志位 LINST.BRK

如果分隔域标志 LINST.BRK 被置位, 软件可以继续捕获同步字节的 4/6/8 位。最后, 同步字节结束标志 (LINST.EOFSYN) 置位, 定时器 T2 停止工作。T2 重载 / 捕获寄存器 (RC2H/L) 存有 2/4/6/8 位的位时间。然后, LIN 子程序计算实际波特率。如果 UART 模块使用波特率发生器产生波特率, 则需设置 PRE 和 BG 的值。

在第 3 个下降沿之后, 如果检测到下面的条件, 软件可以退出当前操作并继续检测下一个 LIN 帧的报文头:

- 分隔域标志 LINST.BRK 未置位, 或者
- 同步字节错误标志 LINST.ERRSYN 被置位, 或者
- 分隔域标志 LINST.BRK 被置位, 但同步字节结束标志 LINST.EOFSYN 和同步字节错误标志 LINST.ERRSYN 未被置位。



## 17.7 寄存器描述

除了 SCON 和 SBUF 寄存器既可从标准（非映射）SFR 区，也可从映射 SFR 区访问外，其余 UART SFR 都位于标准 SFR 区域的 SCU 页 5。访问这些寄存器之前，必须编程设置 SCU\_PAGE 寄存器的 PAGE 位域。

表 17-8 列出 SFR 的地址。

表 17-8 寄存器映射

地址	寄存器
98 <sub>H</sub>	SCON
99 <sub>H</sub>	SBUF
F2 <sub>H</sub>	BCON
F3 <sub>H</sub>	BGL
F4 <sub>H</sub>	BGH
F5 <sub>H</sub>	LINST

## 17.7.1 UART 寄存器

UART 有两个特殊功能寄存器（SFR）：控制寄存器 SCON 和数据寄存器 SBUF。复位时，两个寄存器均返回到 00<sub>H</sub>。SFR SCON 是串口控制和状态寄存器，该寄存器包含模式选择位、发送和接收的第 9 位（TB8 和 RB8）和串口中断位（TI 和 RI）

SBUF 是串行接口的发送和接收缓存寄存器。数据写入 SBUF 将会加载到发送移位寄存器并启动数据发送。该寄存器既用来发送也用来接收数据。发送数据写入该寄存器；接收数据从该寄存器读出，但两条数据通路相互独立。

读取 SBUF 将会访问物理上分开的接收寄存器。

### SBUF

串行数据缓存寄存器

(99<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
VAL							
rwh							

符号	位	类型	描述
VAL	[7:0]	rwh	串行接口缓存寄存器

### SCON

串行通道控制寄存器

(98<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
SM0	SM1	SM2	REN	TB8	RB8	TI	RI
rw	rw	rw	rw	rw	rwh	rwh	rwh

符号	位	类型	描述
RI	0	rwh	接收中断标志 模式 0 中，第 8 位接收结束时，由硬件置位；模式 1、2 和 3 中，接收到停止位的中间时刻由硬件置位。必须由软件清零。

UART

符号	位	类型	描述
TI	1	rwh	<b>发送中断标志</b> 模式 0 中，第 8 位发送结束时，由硬件置位；模式 1、2 和 3 中，开始发送停止位时由硬件置位。必须由软件清零。
RB8	2	rwh	<b>串口接收的位 9</b> 模式 2 和 3 中，该位是接收到的第 9 位数据。 模式 1，该位是接收到的停止位。 模式 0 中，未使用该位。
TB8	3	rw	<b>串口发送的位 9</b> 模式 2 和 3 中，该位是发送的第 9 位数据。
REN	4	rw	<b>串口接收使能</b> 0 <sub>B</sub> 禁止串口接收 1 <sub>B</sub> 使能串口接收
SM2	5	rw	<b>模式 2 和 3 中串口多处理器通信使能位</b> 模式 2 或 3 中，若 SM2 被置 1，如果接收到的第 9 位数据（RB8）为 0，RI 不会被激活；模式 1 中，若 SM2 被置 1，如果没有接收到有效的停止位（RB8），RI 不会被激活。模式 0 中，SM2 应为 0。
SM1	6	rw	<b>串口工作模式选择</b> 见 SM0 的定义。
SM0	7	rw	<b>串口工作模式选择 [SM0, SM1]</b> 00 <sub>B</sub> 模式 0: 8 位移位寄存器，固定波特率 ( $f_{\text{PCLK}}/2$ ) 01 <sub>B</sub> 模式 1: 8 位 UART，可变波特率 10 <sub>B</sub> 模式 2: 9 位 UART，固定波特率 ( $f_{\text{PCLK}}/64$ 或 $f_{\text{PCLK}}/32$ ) 11 <sub>B</sub> 模式 3: 9 位 UART，可变波特率

## 17.7.2 波特率产生控制和状态寄存器

### PCON

功率控制寄存器

(87<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: X, PAGE: X

7	6	5	4	3	2	1	0
SMOD		0		GF1	GF0	0	IDLE
rw		r		rw	rw	r	rw

符号	位	类型	描述
SMOD	7	rw	波特率倍频使能 $0_B$ 模式 1、2 和 3 的波特率禁止倍频 $1_B$ 模式 2 的波特率倍频；只有由定时 T1 产生波特率时，模式 1 和 3 的波特率才倍频
0	1,[6:4]	r	保留 读操作返回 0；应写入 0。

### BCON

波特率控制寄存器

(F2<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
BGSEL		0	BRDIS	BRPRE		R	
rw		r	rw	rw		rw	

符号	位	类型	描述
R	0	rw	波特率发生器运行控制位 $0_B$ 禁止波特率发生器 $1_B$ 使能波特率发生器 <i>注： 仅在 R = 0 时，才能写入 BR_VALUE。</i>
BRPRE	[3:1]	rw	预分频选择 选择由外设时钟分频得到的输入时钟 $f_{DIV}$ $000_B f_{DIV} = f_{PCLK}$ $001_B f_{DIV} = f_{PCLK}/2$ $010_B f_{DIV} = f_{PCLK}/4$ $011_B f_{DIV} = f_{PCLK}/8$ $100_B f_{DIV} = f_{PCLK}/16$ $101_B f_{DIV} = f_{PCLK}/32$ 其它值：保留
BRDIS	4	rw	禁止波特率检测 $0_B$ 使能分隔 / 同步检测 $1_B$ 禁止分隔 / 同步检测

符号	位	类型	描述
<b>BGSEL</b>	[7:6]	rw	<b>选择要检测的波特率范围</b> 对应不同的 BGSEL 值，所检测的波特率范围由下面的公式给出： $\frac{f_{\text{pclk}}}{(2184 \cdot 2^{\text{BGSEL}})} < \text{波特率范围} < \frac{f_{\text{pclk}}}{(72 \cdot 2^{\text{BGSEL}})}$ 其中 BGSEL = 00 <sub>B</sub> , 01 <sub>B</sub> , 10 <sub>B</sub> , 11 <sub>B</sub> . 对应不同输入频率，位域 BGSEL 的取值见表 17-6。

## LINST

LIN 状态寄存器

(F5<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
<b>BGS</b>	<b>SYNEN</b>	<b>ERRSYN</b>	<b>EOFSYN</b>	<b>BRK</b>		<b>0</b>	
rw	rw	rwh	rwh	rwh		r	

符号	位	类型	描述
<b>BRK</b>	3	rwh	<b>分隔域标志</b> 该位由硬件置位，只能由软件清除。 0 <sub>B</sub> 未检测分隔域 1 <sub>B</sub> 检测到分隔域
<b>EOFSYN</b>	4	rwh	<b>同步字节结束中断标志</b> 该位由硬件置位，只能由软件清除。 0 <sub>B</sub> 未检测到同步字节结束 1 <sub>B</sub> 检测到同步字节结束
<b>ERRSYN</b>	5	rwh	<b>同步字节错误中断标志</b> 该位由硬件置位，只能由软件清除。 0 <sub>B</sub> 未在同步字节中检测到错误 1 <sub>B</sub> 在同步字节中检测到错误
<b>SYNEN</b>	6	rw	<b>同步字节结束和同步字节错误中断使能</b> 0 <sub>B</sub> 禁止同步字节结束和同步字节错误中断 1 <sub>B</sub> 使能同步字节结束和同步字节错误中断
<b>BGS</b>	7	rw	<b>波特率发生器选择</b> 0 <sub>B</sub> 选择 UART 波特率发生器 1 <sub>B</sub> 选择定时器 T1 作为波特率发生器

### 17.7.3 波特率发生器定时器 / 重载寄存器

波特率定时器 / 重载寄存器 **BG** 的低位字节和高位字节包含波特率定时器的 11 位重载值和 5 位分数分频器的设置。

读取 **BG** 的低位字节返回波特率定时器的低 3 位和 **FD\_SEL** 的设置，读取高位字节返回波特率定时器的高 8 位。

写入寄存器 **BG** 将重载值载入到波特率定时器并选择分数分频因子，在 **BCON.R** 置位之后的第一个指令周期进行加载操作。

仅在 **R = 0** 时，才能写入 **BG**。

#### BGL

波特率定时器 / 重载寄存器，低位字节

(F3<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
BR_VALUE			FD_SEL				
rwh			rw				

符号	位	类型	描述
FD_SEL	[4:0]	rw	<b>分数分频器选择</b> 选择分数分频器 $n/32$ ，其中 $n$ 为 <b>FD_SEL</b> 的值，范围从 0 到 31。 例如，将 0001 <sub>B</sub> 写入到 <b>FD_SEL</b> 选择的分数分频因子为 $1/32$ 。 <i>注：如果 BR_VALUE = 000<sub>H</sub>，则分数分频器无效</i>
BR_VALUE	[7:5]	rwh	<b>波特率定时器 / 重载值</b> 11 位波特率定时器 / 重载值的低 3 位。 参见 BGH 寄存器的描述。

#### BGH

波特率定时器 / 重载寄存器，高位字节

(F4<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
BR_VALUE							
rwh							

符号	位	类型	描述
BR_VALUE	[7:0]	rwh	波特率定时器 / 重载值 11 位波特率定时器 / 重载值的高 8 位。 当 11 位 BR_VALUE 为 000 <sub>H</sub> 时，旁路波特率定时器。

# 18 IIC 总线

## 18.1 概述

IIC 模块为微控制器和 IIC 总线之间提供符合 IIC 总线协议的通信接口。  
 IIC 可工作在主机模式或从机模式。主机模式下，由 IIC 模块执行仲裁，从而可用于多主机系统中。IIC 通信数据率可高达 400 KBaud，支持 7 位寻址和 10 位寻址。

特性：

- 主机模式或从机模式工作
- 支持多主机系统
- 执行仲裁和时钟同步
- 支持 7 位和 10 位寻址模式
- 产生的波特率可选，可高达 400 KBaud （快速模式）
- 通过中断服务程序或轮询实现的灵活控制

## 18.2 系统信息

本节提供与 IIC 相关的系统信息。

### 18.2.1 引脚

XC83x 中的 IIC 引脚分配见 [表 18-1](#)。

**表 18-1 XC83x 中的 IIC 引脚功能**

引脚	功能	描述	由 ... 选择
P0.4	SCL_0	IIC 时钟输入 / 输出	P0_ALTSEL0.P4 = 0 <sub>B</sub> P0_ALTSEL1.P4 = 0 <sub>B</sub> P0_ALTSEL2.P4 = 1 <sub>B</sub>
P0.2	SCL_1	IIC 时钟输入 / 输出	P0_ALTSEL0.P2 = 0 <sub>B</sub> P0_ALTSEL1.P2 = 1 <sub>B</sub>
P3.0	SCL_2	IIC 时钟输入 / 输出	P3_ALTSEL0.P0 = 1 <sub>B</sub> P3_ALTSEL1.P0 = 1 <sub>B</sub>
P0.7	SCL_3	IIC 时钟输入 / 输出	P0_ALTSEL0.P7 = 0 <sub>B</sub> P0_ALTSEL1.P7 = 0 <sub>B</sub> P0_ALTSEL2.P7 = 1 <sub>B</sub>
P0.6	SDA_0	IIC 数据输入 / 输出	P0_ALTSEL0.P6 = 0 <sub>B</sub> P0_ALTSEL1.P6 = 0 <sub>B</sub> P0_ALTSEL2.P6 = 1 <sub>B</sub>



表 18-1 XC83x 中的 IIC 引脚功能

引脚	功能	描述	由 ... 选择
P0.3	SDA_1	IIC 数据输入 / 输出	P0_ALTSEL0.P3 = 0 <sub>B</sub> P0_ALTSEL1.P3 = 1 <sub>B</sub>
P3.2	SDA_2	IIC 数据输入 / 输出	P3_ALTSEL0.P2 = 1 <sub>B</sub> P3_ALTSEL1.P2 = 1 <sub>B</sub>

对于选中的同一组引脚，由于 SCL 和 SDA 的输入和输出功能设置相同，仅需通过 GPIO 模块中的 Px\_ALTSELn 寄存器进行输入输出设置。不需要单独的输入选择控制（PISEL）。

如果通过 Px\_ALTSELn 寄存器选择的输入 / 输出多于一组，那么信号编号低的具有的优先级较高。例如：如果输出信号 SDA\_0 和 SDA\_1 同时被选中，则输出仅来自 SDA\_0。

### 18.2.1.1 输出引脚配置

分配给 IIC 线的引脚驱动器提供漏极开路输出。这点可确保 IIC 模块未上电时，不会给 IIC 总线带来任何负载。因此，有必要在 IIC 总线连线上添加外部上拉电阻（100 KBaud 工作时大约为 10 KΩ，400 KBaud 工作时大约为 2 KΩ）。

在建立通讯连接之前，用于 IIC 总线通信的所有引脚必须切换到输出，必须使能这些引脚的复用功能。

如果不由 IIC 模块驱动，这些引脚必须关闭其驱动器（即漏极开路输出驱动 1）。由于外部上拉器件的作用，空闲时相应的总线电平为 1。

### 18.2.2 时钟配置

IIC 的工作频率 PCLK 为 8 MHz 或 24 MHz。

如果完全不需要 IIC 功能，可通过关闭其时钟输入的方式整个的禁止该模块，以最大程度降低功耗。可通过置位下述寄存器 PMCON1 中的 IIC\_DIS 位来实现。

在访问 PMCON1 寄存器之前，必须编程设置 SCU\_PAGE 寄存器中的 PAGE 位域。

#### PMCON1

外设管理控制寄存器 1

(EF<sub>H</sub>)

复位值: FF<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	CDC_DIS	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
IIC_DIS	7	rw	IIC 禁止请求，高电平有效。 0 IIC 正常工作 1 请求禁止 IIC（缺省设置）

### 18.2.3 中断事件及其分配

**表 18-2** 列出来自 IIC 模块的中断事件源，以及对应的中断使能和标志位。

**表 18-2 IIC 中断事件**

事件	事件中断使能位	事件标志位
IIC 中断	CNTR.IEN	CNTR.IFLG

**表 18-3** 给出 IIC 中断源的中断节点分配情况。

**表 18-3 IIC 事件的中断节点控制**

事件	中断节点使能位	中断节点标志位	向量地址
IIC 中断	IEN1.EX2	—	43 <sub>H</sub>

## 18.3 状态码

由 STAT 寄存器中的 5 位状态码来定义 IIC 的状态。

当 STAT 中的状态码为 F8<sub>H</sub> 时，表示没有可用的相关状态信息，不产生中断且 CNTR 寄存器中的 IFLG 位不置位。所有其他状态码与定义的 IIC 状态一一对应。当进入其中任意一个状态时，对应的状态码出现在寄存器 STAT 中且 CNTR 寄存器中的 IFLG 位置位。当 IFLG 位被清除时，状态码返回到 F8<sub>H</sub>。

**表 18-4 状态码**

STAT寄存器值	状态
00 <sub>H</sub>	总线错误
08 <sub>H</sub>	发送起始条件（START condition）
10 <sub>H</sub>	发送重复起始条件（Repeated START condition）
18 <sub>H</sub>	发送地址和写位，收到应答（ACK）位
20 <sub>H</sub>	发送地址和写位，收到非应答位
28 <sub>H</sub>	主机模式下发送数据字节，收到应答位
30 <sub>H</sub>	主机模式下发送数据字节，收到非应答位
38 <sub>H</sub>	在地址或数据字节中丢失仲裁
40 <sub>H</sub>	发送地址和读位，收到应答位
48 <sub>H</sub>	发送地址和读位，收到非应答位
50 <sub>H</sub>	主机模式下收到数据字节，发送应答位
58 <sub>H</sub>	主机模式下收到数据字节，发送非应答位
60 <sub>H</sub>	收到从机地址和写位，发送应答位
68 <sub>H</sub>	由于收到主机，从机地址和写位而丢失地址仲裁，发送应答位
70 <sub>H</sub>	收到广播呼叫地址，发送应答位
78 <sub>H</sub>	由于收到主，从地址和读位而丢失地址仲裁，收到应答位
80 <sub>H</sub>	收到从机地址之后，收到数据字节，发送应答位
88 <sub>H</sub>	收到从机地址之后，收到数据字节，发送非应答位
90 <sub>H</sub>	收到广播呼叫地址之后，收到数据字节，发送应答位
98 <sub>H</sub>	收到广播呼叫地址之后，收到数据字节，发送非应答位
A0 <sub>H</sub>	从机模式下，收到停止条件（STOP condition）或重复起始条件
A8 <sub>H</sub>	收到从机地址和读位，发送应答位
B0 <sub>H</sub>	由于收到主机，从机地址和读位而丢失地址仲裁，收到应答位
B8 <sub>H</sub>	从机模式下发送数据字节，收到应答位

表 18-4 状态码

STAT寄存器值	状态
C0 <sub>H</sub>	从机模式下发送数据字节，收到非应答位
C8 <sub>H</sub>	从机模式下发送最后一个字节，收到应答位
D0 <sub>H</sub>	发送第二个地址字节和写位，收到应答位
D8 <sub>H</sub>	发送第二个地址字节和写位，收到非应答位
E0 <sub>H</sub>	未使用
E8 <sub>H</sub>	未使用
F0 <sub>H</sub>	未使用
F8 <sub>H</sub>	没有相关的状态信息， IFLG = 0

如果 IIC 总线上出现非法情况，则进入到总线错误状态（IIC 状态码为 00<sub>H</sub>）。为了从该状态中恢复，CNTR 寄存器中的 STP 位必须置位且 IFLG 位必须被清除。IIC 模块返回到空闲状态（状态码为 F8<sub>H</sub>）：将不在 IIC 总线上发送停止条件。

要请求恢复发送，STA 位置 1 的同时置位 STP 位。然后 IIC 将从总线错误中恢复过来，发送一个起始条件。

## 18.4 波特率产生

为了支持大范围的输入时钟频率（ $f_{IIC}$ ），IIC 提供两个分频器，用于在主机模式产生 IIC 总线上所需要的波特率。两个分频器由 BRCR 寄存器中的 PREDIV 和 BRP 位定义。

按照下述公式计算波特率：

(18.1)

$$f_{OSCL} = \frac{f_{PCLK}}{2^{PREDIV} \times (BRP + 1) \times 10}$$

表 18-5 给出常用的波特率以及所需的分频器设置。

表 18-5 波特率选择

$f_{PCLK}$	波特率 = 100 KBaud		波特率 = 400 KBaud	
	PREDIV	BRP+1	PREDIV	BRP+1
8 MHz	1 (1 <sub>H</sub> )	4 (4 <sub>H</sub> )	1 (1 <sub>H</sub> )	1 (1 <sub>H</sub> )
24 MHz	1 (1 <sub>H</sub> )	12 (C <sub>H</sub> )	1 (1 <sub>H</sub> )	3 (3 <sub>H</sub> )

根据下面的公式计算 IIC 总线的采样频率：

(18.2)

$$f_{\text{SAMP}} = \frac{f_{\text{PCLK}}}{2^{\text{PREDIV}}}$$

为了确保能够正确检测到总线上的起始和停止条件，IIC 模块对 IIC 总线进行采样的频率必须至少是总线上的频率最快的主机总线时钟的 10 倍。在标准模式下，采样频率因此至少为 1 MHz，在快速模式下，至少为 4 MHz，以确保其它总线主机正确操作。

## 18.5 时钟同步

当 IIC 主机模式工作时，如果由 IIC 总线上的另一个器件驱动时钟线，IIC 模块将其时钟与 IIC 总线时钟同步。时钟的高电平周期将由产生最短高电平时钟周期的器件决定。时钟的低电平周期将由产生最长低电平时钟周期的器件决定。

IIC 主机模式工作时，如果与一个时钟速度慢的从机通信，从机可将 SCL 线保持低电平，直到其准备好接受下一位为止的方式来拉长每个位周期。IIC 将按照上述方法自动重新同步时钟。

当 IIC 从机模式工作时，将在传送每个字节之后将 SCL 线保持低电平，直到寄存器 CNTR 中的 IFLG 被清除为止。

## 18.6 总线仲裁

主机模式下，IIC 将检查每个发送的逻辑 1，在 IIC 总线上是否表现为逻辑 1。如果总线上的另一个器件驳回并将 SDA 线拉低，则仲裁丢失。如果发送数据字节或非应答位期间仲裁丢失，则 IIC 将返回到空闲状态。如果发送地址期间仲裁丢失，IIC 将切换到从机模式，从而 IIC 可识别自己的从机地址或者广播呼叫地址。

## 18.7 软件复位

通过向寄存器 SRST（地址 DF<sub>H</sub>）写任意值的方式，IIC 模块将被软件复位。该操作使得 IIC 返回到空闲状态（STAT 设置为 F8<sub>H</sub>），并将 CNTR 寄存器中的 STP，STA 和 IFLG 位置为 0。

## 18.8 操作模式

IIC 模块的所有操作模式需要将 CNTR 寄存器中的 ENAB 位置 1。

### 18.8.1 主机发送

主机发送模式下，IIC 向从机接收器发送数个字节。

**IIC 总线**

将寄存器 CNTR 中的 STA 位置 1，进入到主机发送模式。然后 IIC 将测试 IIC 总线状态，当总线空闲时，IIC 将发送一个起始条件。发送起始条件之后，IFLG 位将置位，STAT 寄存器中的状态码将为 08<sub>H</sub>。在该中断被响应之前，必须将 7 位从机地址或 10 位从机地址的第一部分加载到 DATA 寄存器中，其 LSB 被清 0（也就是说，加上写位）以指示发送模式。此时 IFLG 位应当被清除以促使继续进行传送。

发送 7 位从机地址（或 10 位地址的第一部分）加上写位之后，IFLG 将再次被置位。STAT 寄存器中的状态码可能是下述几种情况：

**表 18-6 主机发送模式下，发送地址之后的状态码**

状态码	IIC 状态	CPU 响应	下一个 IIC 动作
18 <sub>H</sub>	地址 + 写位， 收到应答位	<b>对应于 7 位地址：</b>  向 DATA 写数据字节，清除 IFLG  或者置位 STA，清除 IFLG  或者置位 STP，清除 IFLG  或者置位 STA 和 STP，清除 IFLG  <b>对应于 10 位地址：</b>  将扩展地址字节写入到 DATA，清除 IFLG	发送数据字节，接收应答位  发送重复起始条件  发送停止条件  发送停止条件，然后发送起始条件  发送扩展数据字节
20 <sub>H</sub>	地址 + 写位， 收到非应答位	与状态码 18 <sub>H</sub> 相同	与状态码 18 <sub>H</sub> 相同
38 <sub>H</sub>	仲裁丢失	清除 IFLG  或者置位 STA，清除 IFLG	返回到空闲状态  当总线空闲时，发送起始条件
68 <sub>H</sub>	仲裁丢失 收到 SLA + 写位， 发送应答位	清除 IFLG， AAK = 0  或者清除 IFLG， AAK = 1	接收数据字节，发送非应答位  接收数据字节，发送应答位

表 18-6 主机发送模式下，发送地址之后的状态码

状态码	IIC 状态	CPU 响应	下一个 IIC 动作
78 <sub>H</sub>	仲裁丢失， 收到广播呼叫地址， 发送应答位	与状态码 68 <sub>H</sub> 相同	与状态码 68 <sub>H</sub> 相同
B0 <sub>H</sub>	仲裁丢失， 接收到 SLA + 读位， 发送应答位	向 DATA 写字节，清除 IFLG， AAK = 0	发送最后一个字节，接收 应答位
		或者向 DATA 写字节，清 除 IFLG， AAK = 1	发送数据字节，接收应答 位

如果使用 10 位寻址，那么 10 位地址的第一部分 + 写位被成功发送之后，状态码将为 18<sub>H</sub> 或 20<sub>H</sub>。

中断被响应之后，发送地址第二部分，STAT 寄存器将包含下述状态码之一：

表 18-7 主机发送模式下，发送第二部分地址字节（10 位寻址）之后的状态码

状态码	IIC 状态	CPU 响应	下一个 IIC 动作
38 <sub>H</sub>	仲裁丢失	清除 IFLG	返回到空闲状态
		或者置位 STA，清除 IFLG	当总线空闲时，发送起始 条件
68 <sub>H</sub>	仲裁丢失， 收到 SLA + 写位，发 送应答位	清除 IFLG， AAK = 0	接收数据字节，发送非应 答位
		或者清除 IFLG， AAK = 1	接收数据字节，发送应答 位
B0 <sub>H</sub>	仲裁丢失， 接收 SLA + 读位， 发送应答位	向 DATA 写字节，清除 IFLG， AAK = 0	发送最后一个字节，接收 应答位
		或者向 DATA 写字节，清 除 IFLG， AAK = 1	发送数据字节，接收应答 位

**表 18-7 主机发送模式下，发送第二部分地址字节（10 位寻址）之后的状态码**

状态码	IIC 状态	CPU 响应	下一个 IIC 动作
D0 <sub>H</sub>	发送第二个地址字节 + 写位，收到应答位	向 DATA 写字节，清除 IFLG  或者置位 STA，清除 IFLG  或者置位 STP，清除 IFLG  或者置位 STA 和 STP，清除 IFLG	发送数据字节，接收应答位  发送重复起始条件  发送停止条件  发送停止条件，然后发送起始条件
D8 <sub>H</sub>	发送第二个地址字节 + 写位，收到非应答位	与状态码 D0 <sub>H</sub> 相同	与状态码 D0 <sub>H</sub> 相同

如果已经发送了重复起始条件，状态码将为 10<sub>H</sub>，而不是 08<sub>H</sub>。

发送每个数据字节后，IFLG 将置位，下述三种状态码之一将出现在 STAT 寄存器中。

**表 18-8 主机发送模式下，发送数据之后的状态码**

状态码	IIC 状态	CPU 响应	下一个 IIC 动作
28 <sub>H</sub>	发送数据字节收到应答位	向 DATA 写字节，清除 IFLG  或者置位 STA，清除 IFLG  或者置位 STP，清除 IFLG  或者置位 STA 和 STP，清除 IFLG	发送数据字节，接收应答位  发送重复起始条件  发送停止条件  发送停止条件，然后发送起始条件
30 <sub>H</sub>	发送数据字节收到非应答位	与状态码 28 <sub>H</sub> 相同	与状态码 28 <sub>H</sub> 相同
38 <sub>H</sub>	仲裁丢失	清除 IFLG  或者置位 STA，清除 IFLG	返回到空闲状态  总线空闲时，发送起始条件

当所有字节被发送完毕之后，应当向 CNTR 寄存器的 STP 位写 1。然后 IIC 将发送停止条件，清除 STP 位并返回到空闲状态（状态码为 F8<sub>H</sub>）。



## 18.8.2 主机接收

主机接收模式下，IIC 将接收来自从机发送器的数个字节。

发送起始条件之后，IFLG 位将被置位，STAT 寄存器中的状态码将为 08<sub>H</sub>。此时从机地址（或者 10 位从机地址的第一部分）应当被加载到 DATA 寄存器中，LSB 设置为 1 以指示读操作。此时 IFLG 位将被清除到 0 以促使传送继续进行。

当 7 位从机地址（或者 10 位从机地址的第一部分）和读位被发送之后，IFLG 位将被再次置位。STAT 寄存器中的状态码可能为下述几种情况之一：

**表 18-9 主机接收模式下，发送地址之后的状态码**

状态码	IIC 状态	CPU 响应	下一个 IIC 动作
40 <sub>H</sub>	发送地址 + 读位，收到应答位	清除 IFLG，AAK = 0  或者清除 IFLG，AAK = 1	接收数据字节，发送非应答位  接收数据字节，发送应答位
48 <sub>H</sub>	发送地址 + 读位，收到应答位	置位 STA，清除 IFLG  或者置位 STP，清除 IFLG  或者置位 STA 和 STP，清除 IFLG	发送重复起始条件  发送停止条件  发送停止条件，然后发送起始条件
38 <sub>H</sub>	仲裁丢失	清除 IFLG  或者置位 STA，清除 IFLG	返回到空闲状态  当总线空闲时，发送起始条件
68 <sub>H</sub>	仲裁丢失，收到 SLA + 写位，发送应答位	清除 IFLG，AAK = 0  或者清除 IFLG，AAK = 1	接收数据字节，发送非应答位  接收数据字节，发送应答位
78 <sub>H</sub>	仲裁丢失，收到广播呼叫地址，发送应答位	与状态码 68 <sub>H</sub> 相同	与状态码 68 <sub>H</sub> 相同
B0 <sub>H</sub>	仲裁丢失，收到 SLA + 读位，发送应答位	向 DATA 写字节，清除 IFLG，AAK = 0  或者向 DATA 写字节，清除 IFLG，AAK = 1	发送最后一个字节，接收应答位  发送数据字节，接收应答位

## IIC 总线

如果使用 10 位寻址，使用 10 位地址 + 写位的从机首先被寻址。然后主机在 10 位地址的第一部分之后，紧接着再发送一个重启，但是 + 读位 - 之后状态码将为 40<sub>H</sub> 或 48<sub>H</sub>。由从机来记住其在重启之前被选中。

如果已发送重复起始条件，状态码将为 10<sub>H</sub>，而不是 08<sub>H</sub>。

收到每个数据字节之后，IFLG 将被置位，STAT 寄存器中的状态码将为下面三种情况之一：

**表 18-10 主机接收模式下，收到数据之后的状态码**

状态码	IIC 状态	CPU 响应	下一个 IIC 动作
50 <sub>H</sub>	收到数据字节 发送应答位	读取 DATA，清除 IFLG， AAK = 0	接收数据字节，发送非应 答位
		或者读取 DATA，清除 IFLG，AAK = 1	接收数据字节，发送应答 位
58 <sub>H</sub>	收到数据字节， 发送非应答位	读取 DATA，置位 STA， 清除 IFLG	发送重复起始条件
		或者读取 DATA，置位 STP，清除 IFLG	发送停止条件
		或者读取 DATA，置位 STA 和 STP，清除 IFLG	发送停止条件，然后发送 起始条件
38 <sub>H</sub>	在发送非应答位期间 仲裁丢失	清除 IFLG	返回到空闲状态
		或者置位 STA，清除 IFLG	当总线空闲时，发送起始 条件

收到所有字节之后，应当发送非应答位，然后应当向 CNTR 寄存器中的 STP 位写 “1” 的方式置位该位。IIC 将发送一个停止条件，清除 STP 位并返回到空闲状态（状态码为 F8<sub>H</sub>）。

### 18.8.3 从机发送

从机发送模式下，数个字节被发送至主机接收器。IIC 模块为了回应，CNTR 寄存器中的 AAK 位需要被置位。

起始条件之后，当 IIC 收到自己的从机地址和读位，IIC 将进入到从机发送模式。然后 IIC 将发送一个应答位并置位 CNTR 寄存器中的 IFLG 位。STAT 寄存器中的状态码将为 A8<sub>H</sub>。

注意：如果 IIC 具有扩展从机地址（由 ADDR[7:3] 中的 11110<sub>B</sub> 指示），则该从机将首先被选中，然后一个重启之后紧接着另一个地址字节。如果该地址字节与保存在 ADDR 中

的值匹配，接收到该地址字节之后 IIC 将发送一个应答位。此时产生一个中断，IFLG 将被置位，状态码将为  $A8_H$ 。主机不再发送第二个地址字节：需要由从机记忆在重启之前，该从机被选中。

发送地址和接收从机地址以及读位过程中，如果仲裁丢失，则可从主机模式直接进入到了从机发送模式。STAT 寄存器中的状态码将为  $B0_H$ 。

然后需要发送的数据字节应当加载到 DATA 寄存器中且 IFLG 被清除。当 IIC 已经发送一个字节并接收到应答位，IFLG 将被置位，STAT 寄存器中的状态码将为  $B8_H$ 。一旦要发送的最后一个字节被加载到 DATA 寄存器中，当 IFLG 被清除，AAK 位也应当被清除。发送最后一个字节之后，IFLG 将被置位，且 STAT 寄存器中的状态码为  $C8_H$ 。IIC 返回到空闲模式（状态码为  $F8_H$ ）。要想再次进入从机模式，AAK 位必须先被置 1。

发送一个字节之后，如果未接收到应答位，IFLG 将被置位，STAT 寄存器中的状态码为  $C0_H$ 。IIC 将返回到空闲状态。

应答位之后，如果检测到停止条件，IIC 将返回到空闲状态。

#### 18.8.4 从机接收

在从机接收模式下，从主机发送器接收数个数据字节。

起始条件之后，当 IIC 接收到自身的从机地址和写位（LSB = 0）之后，IIC 将进入从机接收模式。然后 IIC 将发送一个应答位并置位 CNTR 寄存器中的 IFLG 位：此时 STAT 寄存器中的状态码为  $60_H$ 。当 IIC 接收到广播呼叫地址  $00_H$  时，IIC 也会进入到从机接收模式（如果 ADDR 寄存器中的 GCE 位被置位）。此时状态码将为  $70_H$ 。

注意：当 IIC 具有扩展从机地址（由 ADDR[7:3] 的值为  $11110_B$  表示），IIC 将在收到第一个地址字节之后发送一个应答位，但是将不产生中断，IFLG 将不被置位且状态码将不改变。仅在收到第二个地址字节之后，IIC 才产生中断，置位 IFLG 并且状态码变为  $70_H$ 。

发送地址和从机地址 + 写位过程中（或广播呼叫地址，如果 ADDR 寄存器中的 GCE 置 1），主机模式下如果仲裁丢失，则可直接进入到从机接收模式。然后 STAT 寄存器中的状态码将为  $68_H$ （如果接收到从机地址）或者  $78_H$ （如果接收到广播呼叫地址）。为了继续进行数据传送，IFLG 必须被清除至 0。

如果 CNTR 寄存器中的 AAK 位置为 1，那么接收到每个字节之后，发送一个应答位（SDA 上的低电平）且 IFLG 位被置位：STAT 寄存器将继续包含状态码  $80_H$ （或者  $90_H$ ，如果是在广播呼叫地址之后进入到从机接收模式）。可从 DATA 寄存器中读取接收到的数据字节且 IFLG 位必须被清除，以允许继续进行传送操作。应答位之后，检测到停止条件或重复起始条件时，IFLG 位被置位且 STAT 寄存器将包含状态码  $A0_H$ 。

如果传送过程中，AAK 位被清除，IIC 将在接收到下一个字节之后，发送一个非应答位（SDA 上的高电平），并且置位 IFLG 位，STAT 寄存器将包含状态码  $88_H$ （或者  $98_H$ ，如果是在接收到广播呼叫地址之后进入到从机接收模式）。当 IFLG 被清除至 0，IIC 将返回到空闲模式（状态码为  $F8_H$ ）。

## 18.9 寄存器描述

可从标准（非映射）SFR 区域访问 IIC 特殊功能寄存器。表 18-11 列出 IIC 寄存器及其地址。

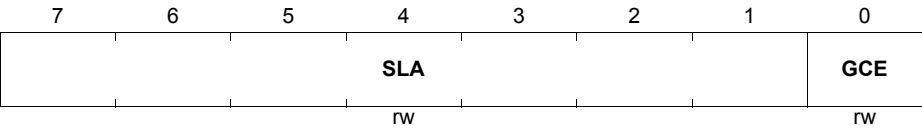
**表 18-11 寄存器映射**

地址	寄存器	描述
DA <sub>H</sub>	IIC_ADDR	从机地址寄存器
DB <sub>H</sub>	IIC_DATA	数据字节寄存器
DC <sub>H</sub>	IIC_CNTR	控制寄存器
DD <sub>H</sub>	IIC_STAT	状态寄存器（只读）
DD <sub>H</sub>	IIC_BRCCR	时钟控制寄存器（只写）
DE <sub>H</sub>	IIC_ADDRX	扩展从机地址寄存器
DF <sub>H</sub>	IIC_SRST	软件复位寄存器

18.9.1 从机地址寄存器

ADDR 寄存器包含广播呼叫地址选择使能位和器件地址（7 位寻址模式下的全部地址和 10 位寻址的地址的第一部分）。10 位寻址模式下，地址第二部分保存在 ADDR<sub>X</sub> 寄存器中。

IIC\_ADDR  
从机地址寄存器 (DA<sub>H</sub>) 复位值: 00<sub>H</sub>  
RMAP: 0, PAGE: X



符号	位	类型	描述
GCE	0	rw	广播呼叫地址使能 0 <sub>B</sub> 禁止广播呼叫地址的选项 1 <sub>B</sub> 使能广播呼叫地址的选项
SLA	[7:1]	rw	从机地址 从机模式工作时，对于 7 位寻址，SLA 包含 IIC 的 7 位地址。 对于 10 位寻址，最高 5 位 SLA[6:2] 固定为 11110 <sub>B</sub> ，而 SLA[1:0] 包含 10 位从地址的高两位。 剩下的低 8 位地址位于 IIC_ADDR <sub>X</sub> 寄存器中。

ADDRX 寄存器包含 10 位寻址模式的低 8 位从机地址。

### IIC\_ADDRX

扩展从机地址寄存器

(DE<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X



符号	位	类型	描述
SLAX	[7:0]	rw	扩展从机地址 从机模式下，对于 10 位寻址方式，SLAX 包含低 8 位地址。

## 18.9.2 数据寄存器

DATA 寄存器包含数据字节或者要发送的从机地址或者刚接收到的数据字节。

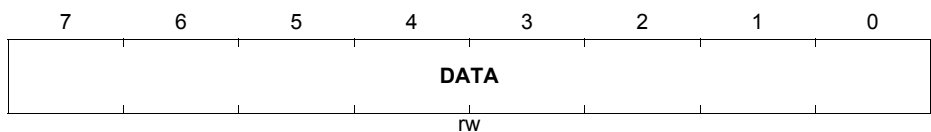
### IIC\_DATA

数据寄存器

(DB<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X



符号	位	类型	描述
DATA	[7:0]	rw	数据字节 要发送或接收到的数据

### 18.9.3 控制寄存器

CNTR 寄存器用于配置 IIC 并产生起始条件和停止条件，该寄存器还包含中断状态标志。

#### IIC\_CNTR

控制寄存器

(DC<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
IEN	ENAB	STA	STP	IFLG	AAK	0	
rw	rw	rwh	rwh	rwh	rw	r	

符号	位	类型	描述
AAK	2	rw	<b>应答控制</b> 0 <sub>B</sub> 在主机模式或从机模式收到数据字节时，发送一个非应答位。 1 <sub>B</sub> 如果出现下述情况之一，发送一个应答位： 1) 接收到匹配的 7 位或 10 位从机地址； 2) 接收到广播呼叫地址； 3) 主机或从机模式下接收到一个数据字节。 <i>注： 在从机发送器模式下，如果 AAK 位被清零，则认为 DATA 寄存器中的字节为最后一个字节。</i>
IFLG	3	rwh	<b>中断标志</b> IFLG 位被硬件置位，只能由软件清除。 0 <sub>B</sub> 未发生中断事件 1 <sub>B</sub> 已发生中断事件
STP	4	rwh	<b>停止主机模式</b> 当 STP 位被置为 1，IIC 将发送一个停止条件。 如果 IIC 处于从机模式，STP 位被置位，则不会在 IIC 总线上发送停止条件，但是 IIC 的操作与已经接收到停止条件的情况相同。 如果 STA 和 STP 位都被置位，IIC 将在发送起始条件之前，先发送停止条件（如果工作在主机模式） 0 <sub>B</sub> IIC 将不发送任何停止条件 1 <sub>B</sub> IIC 将在 IIC 总线上发送一个停止条件 <i>注： 发送停止条件之后，STP 位被自动清除；向该位写 0 无效。</i>

## IIC 总线

符号	位	类型	描述
<b>STA</b>	5	rwh	<b>启动主机模式</b> 当 STA 位被置 1 时，IIC 将进入到主机模式并且一旦 IIC 总线空闲，就发送一个起始条件。 如果 IIC 已经处于主机模式，且已经发送一个或多个字节，将发送一个重复起始条件。 如果从机模式下 IIC 仍然被访问，IIC 将完成从机模式下的数据传送并且一旦 IIC 总线被释放，立即进入到主机模式。 0 <sub>B</sub> IIC 不进入主机模式 1 <sub>B</sub> IIC 进入到主机模式并在 IIC 总线上发送一个起始条件 <i>注：发出起始条件之后，STA 位被自动清除；向该位写 0 无效。</i>
<b>ENAB</b>	6	rw	<b>IIC 使能</b> 0 <sub>B</sub> 禁止 IIC 1 <sub>B</sub> 使能 IIC
<b>IEN</b>	7	rw	<b>中断使能</b> 0 <sub>B</sub> 禁止中断 1 <sub>B</sub> 使能中断
<b>0</b>	[1:0]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 18.9.4 状态寄存器

只读的 STAT 寄存器用来保存 IIC 的 5 位状态码。

#### IIC\_STAT

状态寄存器 [ 读模式 ]

(DD<sub>H</sub>)

复位值：F8<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
STAT					0		
r					r		

符号	位	类型	描述
<b>STAT</b>	[7:3]	r	<b>状态码</b> 5 位状态码（见表 18-4）。



符号	位	类型	描述
0	[2:0]	r	保留 读操作返回 0；应写入 0。

### 18.9.5 波特率控制寄存器

只写寄存器 BRCR 用来控制 IIC 的采样频率和主机模式下的 IIC 的波特率。

#### IIC\_BRCCR

波特率控制寄存器 [ 只写模式 ]

(DD<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0	BRP				PREDIV		
W	W				W		

符号	位	类型	描述
PREDIV	[2:0]	w	用于生成波特率的预分频器 000 <sub>B</sub> 使用预分频因子 1 001 <sub>B</sub> 使用预分频因子 2 010 <sub>B</sub> 使用预分频因子 4 011 <sub>B</sub> 使用预分频因子 8 100 <sub>B</sub> 使用预分频因子 16 101 <sub>B</sub> 使用预分频因子 32 110 <sub>B</sub> 使用预分频因子 64 111 <sub>B</sub> 使用预分频因子 128
BRP	[6:3]	w	波特率预分频器 决定 IIC 主机模式波特率
0	7	w	保留 应当写入 0。

### 18.9.6 软件复位寄存器

SRST 寄存器为 IIC 提供软件复位。

#### IIC\_SRST

软件复位寄存器

(DF<sub>H</sub>)

复位值: XX<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
SRST							
W							

符号	位	类型	描述
SRST	[7:0]	w	软件复位 向 SRST 位域写任意值将触发 IIC 软件复位。

## 19 高速串行同步接口

### 19.1 概述

同步串行接口（SSC）支持全双工和半双工串行同步通信。串行时钟信号由 SSC 内部的 16 位波特率发生器产生（主模式），或从外部主机接收得到（从模式）。数据宽度、移位方向、时钟极性和相位均可编程设定，从而能够和 SPI 兼容器件、或使用其它同步串行接口的器件进行通信。数据发送和数据接收双缓存。

#### 特性

- 主模式和从模式操作
  - 全双工或半双工操作
- 数据发送、接收双缓存
- 灵活的数据格式
  - 数据位个数可编程：2 位至 8 位
  - 移位方向可编程：最低有效位（LSB）或最高有效位（MSB）先移位
  - 时钟极性可编程：移位时钟低电平空闲或高电平空闲
  - 时钟 / 数据相位可编程：在移位时钟的前沿或后沿进行数据移位
- 波特率可变
- 与串行外设接口（SPI）兼容
- 中断产生
  - 发送缓存寄存器已空
  - 接收缓存寄存器已满
  - 出错状况（接收、相位、波特率、发送错误）

**图 19-1** 给出 SSC 的框图。

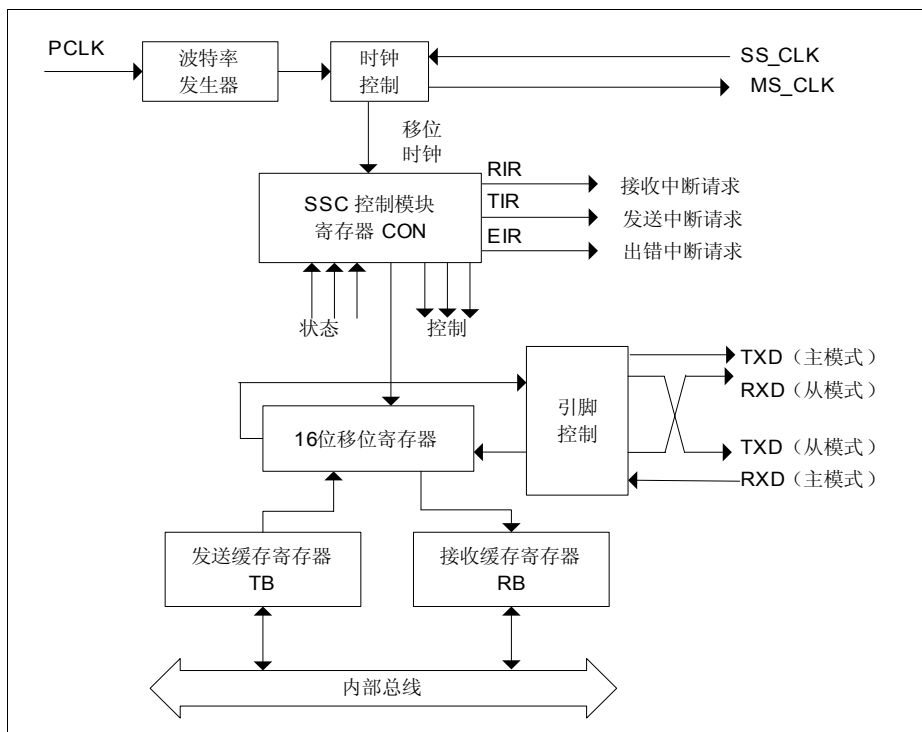


图 19-1 SSC 框图

## 19.2 系统信息

本节给出与 SSC 相关的系统信息。

### 19.2.1 引脚

XC83x 中 SSC 引脚分配见[表 19-1](#)。

**表 19-1 XC83x 中的 SSC 引脚功能**

引脚	功能	描述	由 ... 选择
P0.4	SCK_0	SSC 时钟输入 / 输出	输入: MODPISEL.CIS = 00 <sub>B</sub> 输出: P0_ALTSEL0.P4 = 0 <sub>B</sub> P0_ALTSEL1.P4 = 1 <sub>B</sub> P0_ALTSEL2.P4 = 0 <sub>B</sub>
P3.0	SCK_1	SSC 时钟输入 / 输出	输入: MODPISEL.CIS = 01 <sub>B</sub> 输出: P3_ALTSEL0.P0 = 0 <sub>B</sub> P3_ALTSEL1.P0 = 1 <sub>B</sub>
P2.6	SCK_2	SSC 时钟输入	输入: MODPISEL.CIS = 10 <sub>B</sub>
P2.2	SCK_3	SSC 时钟输入	输入: MODPISEL.CIS = 11 <sub>B</sub>
P0.5	MTSR_0	SSC 主机发送输出 / 从机接收输入	输入: MODPISEL.SIS = 000 <sub>B</sub> 输出: P0_ALTSEL0.P5 = 0 <sub>B</sub> P0_ALTSEL1.P5 = 1 <sub>B</sub> P0_ALTSEL2.P5 = 0 <sub>B</sub>
P0.6	MTSR_1	SSC 从机接收输入	输入: MODPISEL.SIS = 001 <sub>B</sub>
P0.0	MTSR_2	SSC 主机发送输出 / 从机接收输入	输入: MODPISEL.SIS = 010 <sub>B</sub> 输出: P0_ALTSEL0.P0 = 0 <sub>B</sub> P0_ALTSEL1.P0 = 1 <sub>B</sub>
P0.1	MTSR_3	SSC 从机接收输入	输入: MODPISEL.SIS = 011 <sub>B</sub>
P3.1	MTSR_4	SSC 主机发送输出 / 从机接收输入	输入: MODPISEL.SIS = 100 <sub>B</sub> 输出: P3_ALTSEL0.P1 = 0 <sub>B</sub> P3_ALTSEL1.P1 = 1 <sub>B</sub>

**表 19-1 XC83x 中的 SSC 引脚功能**

引脚	功能	描述	由 ... 选择
P3.2	MTSR_5	从机接收输入	输入: MODPISEL.SIS = 101 <sub>B</sub>
P2.1	MTSR_6	从机接收输入	输入: MODPISEL.SIS = 110 <sub>B</sub>
P2.7	MTSR_7	从机接收输入	输入: MODPISEL.SIS = 111 <sub>B</sub>
P0.6	MRST_0	SSC 主机接收输入 / 从机发送输出	输入: MODPISEL.MIS = 000 <sub>B</sub> 输出: P0_ALTSEL0.P6 = 0 <sub>B</sub> P0_ALTSEL1.P6 = 1 <sub>B</sub> P0_ALTSEL2.P6 = 0 <sub>B</sub>
P0.5	MRST_1	SSC 主机接收输入	输入: MODPISEL.MIS = 001 <sub>B</sub>
P0.1	MRST_2	SSC 主机接收输入 / 从机发送输出	输入: MODPISEL.MIS = 010 <sub>B</sub> 输出: P0_ALTSEL0.P1 = 0 <sub>B</sub> P0_ALTSEL1.P1 = 1 <sub>B</sub>
P0.0	MRST_3	SSC 主机接收输入	输入: MODPISEL.MIS = 011 <sub>B</sub>
P3.1	MRST_4	SSC 主机接收输入	输入: MODPISEL.MIS = 100 <sub>B</sub>
P3.2	MRST_5	SSC 主机接收输入 / 从机发送输出	输入: MODPISEL.MIS = 101 <sub>B</sub> 输出: P3_ALTSEL0.P2 = 0 <sub>B</sub> P3_ALTSEL1.P2 = 1 <sub>B</sub>

SSC 使用三根线与外部器件通信。引脚 **SCK** 用作时钟线，引脚 **MRST**（主机接收 / 从机发送）和 **MTSR**（主机发送 / 从机接收）用作串行数据的输入 / 输出线。

每条输入线可有多多个不同输入源，由 **MODPISEL** 寄存器从中选择输入信号。与此相似，每个输出线也可与多个信号相连，由对应端口的 **ALTSELx** 寄存器进行选择。

访问 **MODPISEL** 寄存器之前，必须编程设置 **SCU\_PAGE** 寄存器的 **PAGE** 位域。

## 高速串行同步接口

## MODISEL

外设输入选择寄存器

(F3<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
CIS		SIS			MIS		
rw		rw			rw		

符号	位	类型	描述
MIS	[2:0]	rw	<b>主机模式接收输入选择</b> 000 选择 SSC 主机接收输入 0 001 选择 SSC 主机接收输入 1 010 选择 SSC 主机接收输入 2 011 选择 SSC 主机接收输入 3 100 选择 SSC 主机接收输入 4 101 选择 SSC 主机接收输入 5 110 保留 111 保留
SIS	[5:3]	rw	<b>从机模式接收输入选择</b> 000 选择 SSC 从机接收输入 0 001 选择 SSC 从机接收输入 1 010 选择 SSC 从机接收输入 2 011 选择 SSC 从机接收输入 3 100 选择 SSC 从机接收输入 4 101 选择 SSC 从机接收输入 5 110 选择 SSC 从机接收输入 6 111 选择 SSC 从机接收输入 7
CIS	[7:6]	rw	<b>从机模式时钟输入选择</b> 00 选择 SSC 从机时钟输入 0 01 选择 SSC 从机时钟输入 1 10 选择 SSC 从机时钟输入 2 11 选择 SSC 从机时钟输入 3



### 19.2.2 时钟配置

SSC 的工作频率 PCLK 可为 8 MHz 或 24 MHz。

如果完全不需要 SSC 功能，可通过关闭其时钟输入的方式禁止该模块，从而最大程度的降低功耗。

访问 PMCON1 寄存器之前，必须编程设定 SCU\_PAGE 中的 PAGE 位域。

**PMCON1**  
 外设管理控制寄存器 1 (EF<sub>H</sub>) 复位值: FF<sub>H</sub>  
 RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	CDC_DIS	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
SSC_DIS	1	rw	SSC 禁止请求，高有效 0 SSC 正常工作 1 请求禁止 SSC（缺省设置）

### 19.2.3 中断事件和分配

**表 19-2** 列出 SSC 的中断事件源以及对应的中断节点分配。

*注： 所有 SSC 中断使能位和标志位都在中断节点中设置。*

**表 19-2 SSC 事件的中断节点控制**

事件	中断节点使能位	中断节点标志位	向量地址
传送开始	MODIEN.TIREN; IEN1.ESSC	IRCON1.TIR	3B <sub>H</sub>
传送结束	MODIEN.RIREN; IEN1.ESSC	IRCON1.RIR	3B <sub>H</sub>
出现错误	MODIEN.EIREN; IEN1.ESSC	IRCON1.EIR	3B <sub>H</sub>

SSC 模块的三个中断使用中断结构 2，由寄存器 MODIEN 使能这些中断。访问 MODIEN 之前，必须编程设置 SCU\_PAGE 中的 PAGE 位域。

**MODIEN**  
外设中断使能寄存器 (F7<sub>H</sub>) 复位值：07<sub>H</sub>  
RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
CCU6SR3 EN	CCU6SR2 EN		0		RIREN	TIREN	EIREN
rw	rw		r		rw	rw	rw

符号	位	类型	描述
EIREN	0	rw	<b>SSC 出错中断使能</b> 0 禁止出错中断 1 使能出错中断
TIREN	1	rw	<b>SSC 发送中断使能</b> 0 禁止发送中断 1 使能发送中断
RIREN	2	rw	<b>SSC 接收中断使能</b> 0 禁止接收中断 1 使能接收中断
0	[5:3]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 19.3 一般操作

SSC 支持全双工和半双工同步通信，波特率可达 12 MBaud (@ 24 MHz 模块时钟)。串行时钟信号可由 SSC 自身产生（主模式）或从外部主机接收（从模式）。数据宽度、移位方向、时钟极性和相位均可编程设定，从而能够和串行外设接口（SPI）兼容器件进行通信。数据发送和接收双缓存。SSC 内的 16 位波特率发生器可为 SSC 提供单独的串行时钟信号。

SSC 可灵活配置，因此其可和其它同步串行接口通信，用作主机 / 从机或进行多主通信的互连或者可与流行的兼容 SPI 接口进行通信。因此，SSC 可与移位寄存器（I/O 扩展），外设（如 EEPROM 等等）或其它控制器（联网）通信。SSC 支持全双工和半双工通信。通过 TXD 和 RXD 线发送或接收数据，通常这两条线分别和引脚 MTSR（主机发送 / 从机接收）和 MRST（主机接收 / 从机发送）相连。时钟信号从 MS\_CLK（主机串行移位时钟）输出或从 SS\_CLK（从机串行移位时钟）输入；这两条时钟线通常和引脚 SCLK 相连。

### 19.3.1 工作模式选择

控制寄存器 CON 控制串行通道 SSC 的工作模式。该寄存器具有双重功能：

- 编程过程中（CON.EN=0 禁止 SSC），可通过该寄存器访问控制位
- 工作工程中（CON.EN=1 使能 SSC），可通过该寄存器访问状态标志

SSC 移位寄存器经引脚控制逻辑与发送线和接收线相连。串行数据的发送和接收同步进行，且发送和接收操作同时发生，即接收数据和发送数据的位数相同。发送数据写入发送缓存寄存器（TB），移位寄存器一旦为空，则立即将 TB 的内容移入。SSC 主机（CON.MS=1）立刻开始发送；SSC 从机（CON.MS=0）将等待有效的移位时钟。开始传送时，忙标志 CON.BSY 被置位，发送中断请求线（TIR）被激活以指示寄存器 TB 可重新装入。传送完设定的数据位（2...8 位）之后，移位寄存器的内容移入接收缓存寄存器（RB），接收中断请求线（RIR）被激活。如果没有进一步的数据发送（TB 为空），同时对 CON.BSY 清零。CON.BSY 由硬件控制，不应由软件修改。

**注：** SSC 开始发送数据，在发送数据写入 TB 至少两个时钟周期之后置位 CON.BSY。因此，不建议通过查询 CON.BSY 指示传送的开始和结束；建议由中断服务程序（若中断被使能）或中断标志位 IRCON1.TIR 和 IRCON1.RIR（若中断被禁止）指示传送的开始和结束。

**注：** 在给定的时间内每次只有一个 SSC 用作主机。

串行数据的传送可用很多方式设定：

- 数据宽度规定为 2 到 8 位
- 可先传送 LSB 或先传送 MSB
- 移位时钟可为低电平空闲或高电平空闲
- 可在移位时钟信号的前沿或后沿对数据移位
- 波特率范围从 183.11 Baud 到 12 Mbaud (@24 MHz 模块时钟)
- 可产生移位时钟（MS\_CLK）或接收移位时钟（SS\_CLK）

这些特性使 SSC 广泛适用于需要串行数据传送的应用领域。

## 高速串行同步接口

数据宽度选择支持从 2 位到 8 位“字符”任意长度数据帧的传送。先传送 LSB（CON.HB=0）可使 SSC 和 SSC 器件在同步模式下通信；或和其它串行接口、如 8051 的串行接口通信；先传送 MSB（CON.HB=1）可使 SSC 与兼容 SPI 接口进行通信。

无论如何设定数据宽度、选择先传送 LSB 还是 MSB，寄存器 TB 和 RB 中的传送数据始终右对齐，数据的 LSB 存放在寄存器的位 0。内部移位寄存器逻辑将重排数据。TB 中未选中的数据位被忽略；RB 中未选中的数据位无效、接收服务程序对其不予理睬。

时钟控制使 SSC 的收发行为可适用于多种不同的串行接口。用一个特定移位时钟沿（上升或下降）移出发送数据；同时用另一个移位时钟沿锁存接收数据。位 CON.PH 选择用时钟前沿或后沿发送 / 接收。位 CON.PO 选择空闲状态下移位时钟的电平。因此，对于高电平空闲时钟，时钟前沿为时钟下降沿，即 1 到 0 的跳变（见图 19-2）。

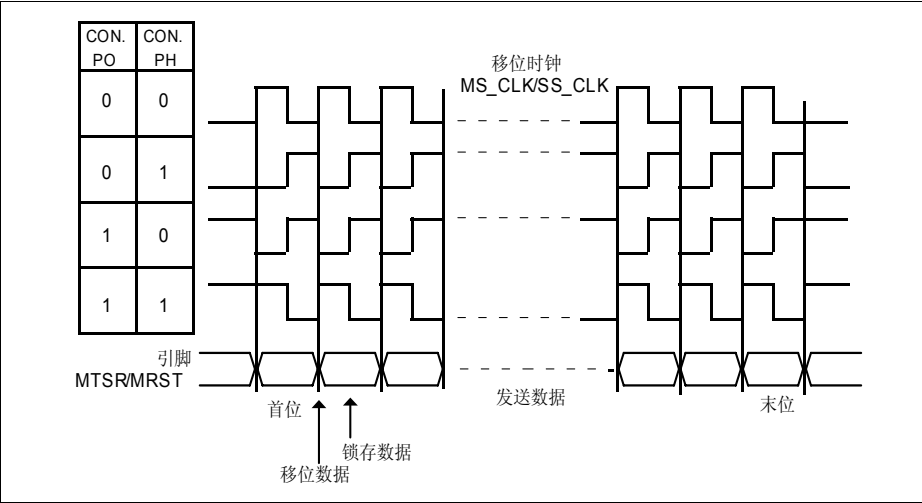


图 19-2 串行时钟相位和极性选择

### 19.3.2 全双工工作

由三条线将各个器件相互连接。始终由主机来定义这三条线：和主机数据输出线 TXD 相连的是发送线；和主机数据输入线 RXD 相连的是接收线；移位时钟线是 MS\_CLK 或 SS\_CLK。由被选作主机的器件产生移位时钟、并从 MS\_CLK 线上将其输出。由于所有从机会接收该时钟，它们的引脚 SCLK 必须切换到输入模式。主机移位寄存器的输出与外部发送线相连，这些线依次连接至从机的移位寄存器输入。为了使能主机接收从机移出的数据，从机移位寄存器的输出与外部接收线相连。外部连接是硬件连线，这些引脚的功能和方向由各个器件是主机操作还是从机操作决定。

注：图中所示的移位方向对于先传送 LSB 和先传送 MSB 的操作都适用。

初始化器件时，必须选定一个主机，其他所有器件都设置为从机操作。初始化过程包括选择器件 SSC 的工作模式以及设置对应端口线的功能。

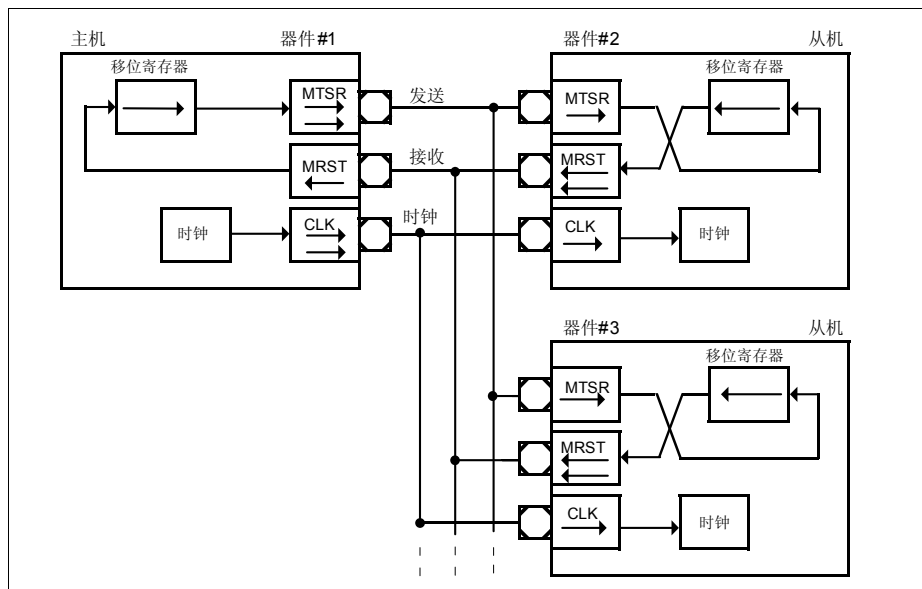


图 19-3 SSC 全双工配置

所有从机器件的数据输出引脚 **MRST** 都连接到同一条接收线上，如图 19-3 所示。数据传送时，每个从机从移位寄存器移出数据。有两种方式避免（不同从机同时传送数据引起）接收线上的数据传送冲突：

- 只有一个从机驱动接收线，即使能该从机 **MRST** 引脚驱动器。必须将所有其它从机的 **MRST** 引脚设置为输入，从而只有一个从机可将数据输出到主机的接收线上，其它从机只能接收来自主机的数据。主机通过两种方式选择期望从其获取数据的从机器件：通过单独的选择线或向该从机发送特殊命令。被选中的从机将 **MRST** 引脚切换到输出，直到它收到一个取消信号或命令。这种方式仅适用于引脚能够禁止其输出驱动器的情况。
- 从机在 **MRST** 引脚上采用漏极开路输出，形成线与连接。这种情况下接收线需要外部上拉。所有未选中向主机发送数据的从机只发送“1”，从而避免了选中的从机发送到接收线上的数据遭破坏。因为这一高电平不能被有效驱动到接收线上，只能由上拉器件保持；选中的从机发送“0”时，可有效的将接收线拉低。主机通过两种方式选择期望从其获取数据的从机器件：通过单独的选择线或向该从机发送特殊命令。

执行完必要的 **SSC** 初始化，可启用串行接口。主机器件的时钟线此时进入设定的时钟极性。开始传送之前数据线上的值为 0 或 1。传送结束后，数据线将始终保持最后发送的数据位的逻辑电平。

当串行接口使能，主机器件将发送数据写入 **TB**，启动数据传送。该数据被复制到移位寄存器中（假定此时寄存器已空），在波特率发生器产生的下一个时钟，将选定的发送数据的首位送到 **TXD** 线上（只有 **CON.EN=1** 才可开始发送）。根据选择的时钟相位，在

## 高速串行同步接口

MS\_CLK 线上产生时钟脉冲。同时，在相反的时钟沿，主机将输入线 RXD 上检测到的数据位锁存并移入移位寄存器，发送数据和接收数据进行了“交换”。由于时钟线和所有从机相连，从机移位寄存器将和主机移位寄存器同步移位 - 移出寄存器中的数据，移入输入线上检测到的数据。在预设个数的时钟脉冲（由数据宽度选择）之后，主机发送的数据全部保存在所有从机移位寄存器中；同时主机移位寄存器中保存选中从机发送的数据。主机和所有从机移位寄存器的内容被复制到接收缓存寄存器 RB，接收中断请求线 RIR 被激活。

当发送缓存寄存器的内容复制到从机的移位寄存器中，从机器件立刻在 RXD 上输出选中的数据首位（传送数据的 MSB 或 LSB）。直到 SS\_CLK 的第一个时钟沿出现时才置位 CON.BSY。从机器件将不会象主机那样等待波特率发生器的下一个时钟，原因是：根据所选择的时钟相位，由主机产生的首个时钟沿可能已经用于移出首个数据位。因此，在此刻从机的首个数据位必须已经有效。

*注：无论有效数据是否被发送或接收，SSC 的发送和接收总是同时发生。*

*注：在初始化主机 CLK 引脚时需多加注意，以免产生不期望的时钟跳变，该跳变可能会干扰其它器件。在由相关的方向控制寄存器将时钟引脚切换到输出之前，必须由控制寄存器 CON 选择时钟输出电平；通过相关的 ALTSEL 寄存器设置该引脚复用功能输出选择；或将时钟空闲电平装入输出锁存器中。*

### 19.3.3 半双工工作

在半双工模式下，一条数据线既用作数据接收又用作数据发送。半双工模式有两种端口配置方式。第一种选择是使用所有三个引脚，但是数据交换线与每个器件的 MTSR 和 MRST 引脚连接，移位时钟线和 SCLK 引脚连接。第二种选择是使用两个引脚，要求 MTSR 和 MRST 线选择同一个 GPIO 引脚，因此建立通信仅需要单根数据交换线，移位时钟线和前一种方式相同，与 SCLK 相连。

主机器件产生移位时钟控制数据的传送，同时从机器件接收主机发送的数据。由于所有的发送和接收引脚均连接到同一条数据交换线上，串行数据可在任意两个器件之间传送。

和全双工模式相似，通过两种方式避免数据交换线上的数据传送冲突：

- 只有发送器件可开启其发送引脚驱动器，此方式仅适用于引脚能够禁止其输出驱动器的情况。
- 不发送数据的器件采用漏极开路输出并只发送“1”

因为数据输入和输出彼此相连，发送器件将从输入引脚（主机器件对应 MRST，从机对应 MTSR）读回它发送的数据。若接收数据和发送数据不一致，通过这种方法可以检测到公共数据交换线上数据是否遭破坏。

使用三个引脚的半双工模式端口配置见 [图 19-4](#)。



### 19.3.5 波特率产生

串行通道 SSC 自带 16 位专用波特率发生器，具有 16 位重载功能，使波特率的产生和定时器无关。SSC 波特率发生器如图 19-5 所示。

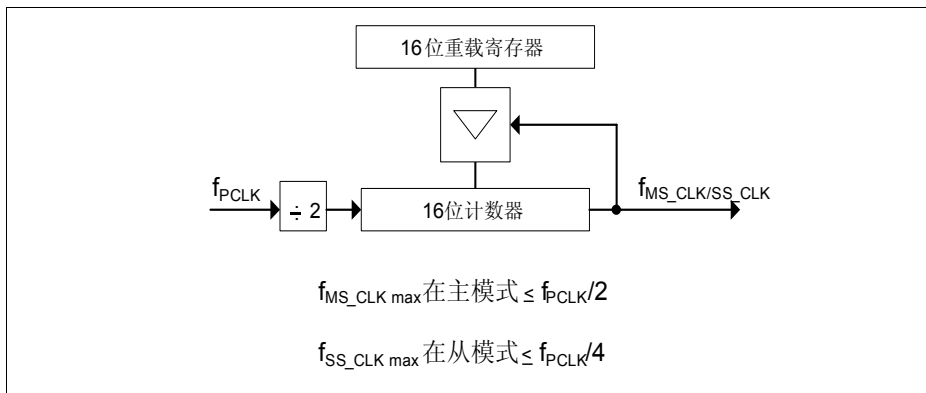


图 19-5 SSC 波特率发生器

波特率发生器的输入时钟为模块时钟  $f_{\text{hw\_clk}}$ ，定时器递减计数。寄存器 BR 具有双重功能，是波特率发生器 / 重载寄存器。SSC 被使能时，读取 BR 返回定时器的计数值；SSC 被禁止时，读取 BR 返回设定的重载值。该模式下，期望的重载值可写入到 BR。

注：SSC 被使能时，一定不能写访问 BR。

下面给出给定重载值时计算波特率，或给定波特率计算重载值的公式：

$$\text{波特率} = \frac{f_{\text{hw\_clk}}}{2 \cdot (\text{BR} + 1)} \quad \text{BR} = \frac{f_{\text{hw\_clk}}}{2 \cdot \text{波特率}} - 1 \quad (19.1)$$

<BR> 代表重载寄存器的值，被当作无符号 16 位整数，波特率等于  $f_{\text{MS\_CLK/SS\_CLK}}$ 。

模块时钟为 24 MHz 时，可达到的最大波特率：主模式下为 12 MBaud (<BR> = 0000<sub>H</sub>) 或从模式下为 6 MBaud (<BR> = 0001<sub>H</sub>)。

表 19-3 列出一些常用的波特率以及所需的重载值和偏差，假设模块时钟为 24 MHz。

表 19-3 SSC 典型波特率 ( $f_{\text{hw\_clk}} = 24 \text{ MHz}$ )

重载值	波特率 (= $f_{\text{MS\_CLK/SS\_CLK}}$ )	偏差
0000 <sub>H</sub>	12 MBaud (仅适用于主模式)	0.0%
0001 <sub>H</sub>	6 MBaud	0.0%
0005 <sub>H</sub>	2 MBaud	0.0%
000B <sub>H</sub>	1 MBaud	0.0%



表 19-3      **SSC 典型波特率 ( $f_{hw\_clk} = 24\text{ MHz}$ )**

重载值	波特率 ( $= f_{MS\_CLK/SS\_CLK}$ )	偏差
0017 <sub>H</sub>	500 kBaud	0.0%
0077 <sub>H</sub>	100 kBaud	0.0%
FFFF <sub>H</sub>	183.11 Baud	0.0%

### 19.3.6      检错机制

SSC 能够检测四种出错情况。主从模式下均可检测接收出错和相位出错；仅在从模式下检测发送出错和波特率出错。当检测到错误时，对应的出错标志位被 1 可被置位，如被使能，激活出错中断请求线（EIR）产生出错中断请求（见 图 19-6）。错误中断处理器会检查出错标志以确定出错原因。出错标志不会被自动复位，中断被响应后必须由软件清零。出错中断使能位被置位时，可由中断服务程序来响应出错情况；出错中断使能位未被置位时，可由软件查询出错情况。

注： 出错中断处理器必须对相关（被使能的）出错标志清零，以防止重复产生中断请求。

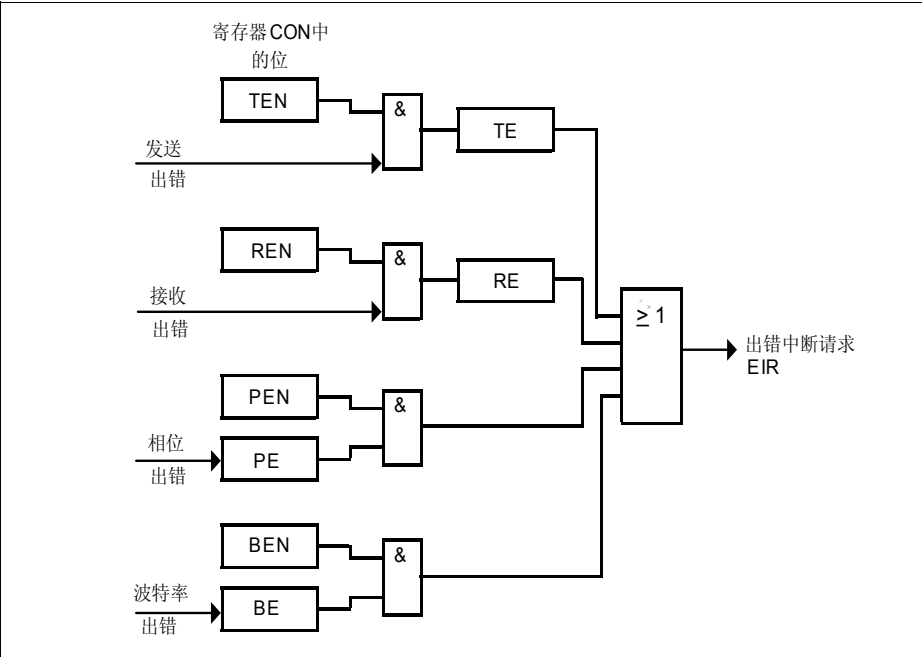


图 19-6      **SSC 错误中断控制**

## 高速串行同步接口

**接收出错**（主模式或从模式）：当新的数据帧已接收完毕，但上个数据仍未从接收缓存寄存器 RB 中读出时检测到接收出错。使能 CON.REN 时，置位出错标志 CON.RE 和出错中断请求线 EIR。接收缓存寄存器 RB 中的旧数据将被新数据覆盖且不可恢复。

**相位出错**（主模式或从模式）：以模块时钟频率采样引脚 MRST（主模式）或引脚 MTSR（从模式）上的输入数据，在移位时钟信号 SCLK 锁存时钟沿的一个周期之前和两个周期之后若数据改变则检测到相位出错。此时置位出错标志 CON.PE，CON.PEN 被使能时，置位出错中断请求线 EIR。

*注： 当同时接收和发送数据时，如果波特率设置为  $f_{hw\_clk}/2$ ，出现相位错误。*

**波特率出错**（从模式）：当输入时钟信号和设定的波特率之间偏差超过 100% 时检测到波特率出错，即实际波特率超过期望波特率的两倍或不到期望波特率的一半。此时置位出错标志 CON.BE，CON.BEN 被使能时，置位 EIR。使用该检错功能要求将从机的波特率发生器设置成可产生和主机器件相同的波特率。该特性可检测时钟线上错误的、多余的或丢失的脉冲（在某帧之内）。

*注： 如果发生该错误且位 CON.AREN = 1，将自动复位 SSC。检测到过少或过多的时钟脉冲时会自动复位，重新初始化 SSC。*

*注： 如果通信停止，任何传送之后都可出现该错误。这种情况的出现是由于 SSC 模块支持背对背的多传送。为了处理这种情况，在完成一次传送之后，波特率检测器期望在下一个时钟周期立即进行新的传送。*

**发送出错**（从模式）：主机启动传送（SS\_CLK 已有效），但从机发送缓存寄存器 TB 在上次传送后仍未更新，检测到发送出错。使能 CON.TEN 时，置位出错标志 CON.TE 和 EIR。如果发送缓存器未被更新即开始传送，从机将移出移位寄存器中“旧”的内容，它通常是上次传送时接收到的数据。如果未选中该从机发送数据，这将可能导致半双工模式下发送 / 接收线上数据被破坏（漏极开路输出设置）。该模式要求未被选中发送数据的从机只发送“1”；也就是说，传送前必须将“FFFF<sub>H</sub>”装入从机发送缓存寄存器中。

*注： 为了避免可能的数据冲突或数据错误识别，建议在任何传送之前始终先加载从机发送缓存寄存器。*

可通过控制寄存器 CON 中的出错状态标志确定出错中断请求（接收、相位、波特率或发送出错）产生的原因。

*注： 出错状态标志 CON.TE, CON.RE, CON.PE 和 CON.BE 在进入出错中断服务程序时不能被自动复位，必须由软件清零。*

### 19.4 中断

SSC 的三个中断可通过置位 / 清除 SFR SCU\_MODIEN 中的对应位来单独使能或禁止。  
各中断的详细描述见 [章节 19.3](#)，总结见 [表 19-4](#)。

**表 19-4 SSC 中断源**

中断	信号	描述
发送开始	TIR	指示发送缓存寄存器可重新装入新数据
发送结束	RIR	设定位数的数据已发送完成并移入接收缓存寄存器
接收出错	EIR	当新的数据帧已接收完毕，但上个数据仍未从接收缓存寄存器中读出时，产生该中断
相位出错	EIR	在移位时钟信号 SCLK 锁存时钟沿的一个周期之前和两个周期之后数据改变，产生该中断
波特率出错 (仅限从模式)	EIR	当输入时钟信号和设定的波特率之间偏差超过 100% 时，产生该中断
发送出错 (仅限从模式)	EIR	主机启动发送，但从机的 TB 在上次传送后仍未更新，产生该中断

## 19.5 寄存器描述

可从标准（非映射）SFR 区域访问 SSC 特殊功能寄存器。SFR 的地址见表 19-5。

**表 19-5 寄存器映射**

地址	寄存器
AA <sub>H</sub>	CONL
AB <sub>H</sub>	CONH
AC <sub>H</sub>	TBL
AD <sub>H</sub>	RBL
AE <sub>H</sub>	BRL
AF <sub>H</sub>	BRH

## 19.5.1 配置寄存器

串行通道 SSC 的工作模式由控制寄存器 CON 控制。该寄存器包含模式和检错选择控制位，以及用于错误识别的出错状态标志位。根据位 EN 的设置，使能控制功能或状态标志，使能主 / 从模式控制。

**CON.EN = 0: 编程模式**

### SSC\_CONL

控制寄存器低位 [ 编程模式 ]

(AA<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
LB	PO	PH	HB			BM	
rw	rw	rw	rw			rw	

符号	位	类型	描述
BM	[3:0]	rw	<b>数据宽度选择</b> 0000 <sub>B</sub> 保留。不要使用该组合。 0001 <sub>B</sub> - 0111 <sub>B</sub> 传送数据宽度为 2 ... 8 位 (<BM>+1) <i>注: BM 位域的最高位始终为 0。因此传送和接收数据的最大值为 8 位。允许的数据宽度为 2...8 位 (&lt;BM&gt;+1)</i>
HB	4	rw	<b>报文头控制</b> 0 <sub>B</sub> 先发送 / 接收 LSB 1 <sub>B</sub> 先发送 / 接收 MSB
PH	5	rw	<b>时钟相位控制</b> 0 <sub>B</sub> 在时钟的前沿移位输出发送数据，后沿锁存接收数据 1 <sub>B</sub> 在时钟的前沿锁存接收数据，后沿移位输出发送数据
PO	6	rw	<b>时钟极性控制</b> 0 <sub>B</sub> 时钟线低电平空闲，时钟前沿为低到高跳变 1 <sub>B</sub> 时钟线高电平空闲，时钟前沿为高到低跳变
LB	7	rw	<b>回环控制</b> 0 <sub>B</sub> 正常输出 1 <sub>B</sub> 接收输入和发送输出相连（半双工模式）

# SSC\_CONH

控制寄存器高位 [ 编程模式 ]

(AB<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
EN	MS	0	AREN	BEN	PEN	REN	TEN
rw	rw	r	rw	rw	rw	rw	rw

符号	位	类型	描述
TEN	0	rw	发送出错中断使能 0 <sub>B</sub> 禁止发送出错中断 1 <sub>B</sub> 使能发送出错中断
REN	1	rw	接收出错使能 0 <sub>B</sub> 禁止接收出错中断 1 <sub>B</sub> 使能接收中断出错
PEN	2	rw	相位出错使能 0 <sub>B</sub> 禁止相位出错中断 1 <sub>B</sub> 使能接收中断出错
BEN	3	rw	波特率出错使能 0 <sub>B</sub> 禁止波特率出错中断 1 <sub>B</sub> 使能波特率出错中断
AREN	4	rw	自动复位使能 0 <sub>B</sub> 波特率出错时, 无需附加动作 1 <sub>B</sub> 波特率出错时, SSC 自动复位。
MS	6	rw	主机选择 0 <sub>B</sub> 从模式, 从 SCLK 接收移位时钟 1 <sub>B</sub> 主模式, 产生移位时钟并输出至 SCLK
EN	7	rw	使能位 = 0 禁止发送和接收。访问控制位。
0	5	r	保留 读操作返回 0 ; 应写入 0。

CON.EN = 1: 工作模式

## SSC\_CONL

控制寄存器低位 [ 工作模式 ]

(AA<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
0				BC			
r				rh			

符号	位	类型	描述
BC	[3:0]	rh	位计数域 每次移位操作都更新移位计数器。 <i>注: 该位域不能被写入。</i>
0	[7:4]	r	保留 读操作返回 0 ; 应写入 0。

## SSC\_CONH

控制寄存器高位 [ 工作模式 ]

(AB<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
EN	MS	0	BSY	BE	PE	RE	TE
rw	rw	r	rh	rwh	rwh	rwh	rwh

符号	位	类型	描述
TE	0	rwh	发送出错标志 0 <sub>B</sub> 发送未出错 1 <sub>B</sub> 从机发送缓存寄存器未被更新即开始传送数据
RE	1	rwh	接收出错标志 0 <sub>B</sub> 接收未出错 1 <sub>B</sub> 接收缓存寄存器的内容在读出之前, 新数据已接收完毕

## 高速串行同步接口

符号	位	类型	描述
PE	2	rwh	<b>相位出错标志</b> $0_B$ 相位未出错 $1_B$ 接收数据在采样时钟沿附近改变
BE	3	rwh	<b>波特率出错标志</b> $0_B$ 波特率未出错 $1_B$ 从机的实际波特率超过期望波特率的 2 倍或低于 0.5 倍
BSY	4	rh	<b>忙标志</b> 传送过程中被置位。 <i>注： 该位不能被写入。</i>
MS	6	rw	<b>主机选择位</b> $0_B$ 从模式。从 SCLK 接收移位时钟 $1_B$ 主模式。产生移位时钟并从 SCLK 输出
EN	7	rw	<b>使能位 = 1</b> 发送和接收被使能。访问状态标志和主机 / 从机控制。
0	5	r	<b>保留</b> 读操作返回 0，应写入 0。

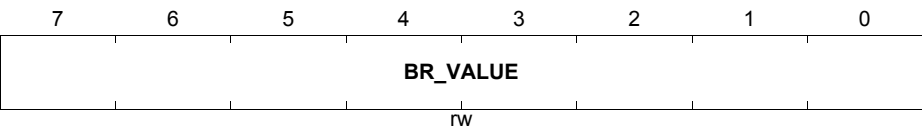
*注： 在访问 CON 之前，先由 CON.EN 的状态决定访问 CON 的目标（控制位或标志位）。也就是说，在编程模式下（CON.EN = 0）CON 中写入 C057<sub>H</sub> 将初始化 SSC（CON.EN 为 0），然后启动 SSC 工作（CON.EN = 1）。对 CON 进行写操作时，必须对保留位写入 0。*



### 19.5.2 波特率定时器重载寄存器

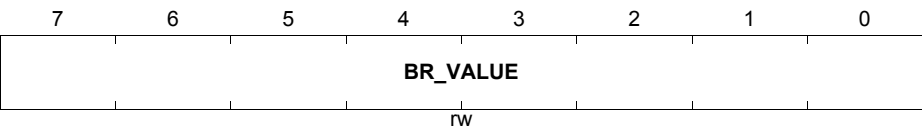
SSC 的波特率定时器重载寄存器 BR 中包含波特率定时器的 16 位重载值。

**SSC\_BRL**  
 波特率定时器重载寄存器低位 (AE<sub>H</sub>) 复位值: 00<sub>H</sub>  
 RMAP: 0, PAGE: X



符号	位	类型	描述
BR_VALUE	[7:0]	rw	波特率定时器 / 重载寄存器值 读取 BR 返回波特率定时器的 16 位计数值；写入 BR 将 BR_VALUE 装入波特率定时器重载寄存器中。

**SSC\_BRH**  
 波特率定时器重载寄存器高位 (AF<sub>H</sub>) 复位值: 00<sub>H</sub>  
 RMAP: 0, PAGE: X



符号	位	类型	描述
BR_VALUE	[7:0]	rw	波特率定时器 / 重载寄存器值 读取 BR 返回波特率定时器的 16 位计数值；写入 BR 将 BR_VALUE 装入波特率定时器重载寄存器中。

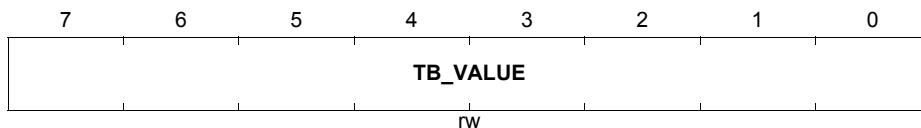
### 19.5.3 发送缓存寄存器

SSC 的发送缓存寄存器 TB 中存放发送数据。

**SSC\_TBL**

发送缓存寄存器低位

**(AC<sub>H</sub>)**

 复位值: **00<sub>H</sub>**
**RMAP: 0, PAGE: X**


符号	位	类型	描述
<b>TB_VALUE</b>	[7:0]	rw	发送数据寄存器值 TB_VALUE 将发送的数据值，发送时 TB 中未被选中的位不予理睬。

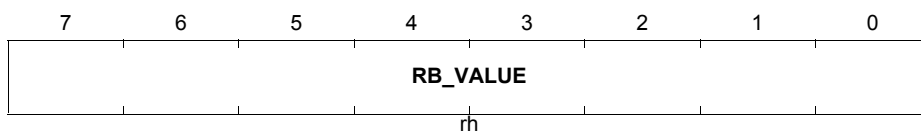
**19.5.4 接收缓存寄存器**

SSC 的接收缓存寄存器 RB 中存放接收数据。

**SSC\_RBL**

接收缓存寄存器低位

**(AD<sub>H</sub>)**

 复位值: **00<sub>H</sub>**
**RMAP: 0, PAGE: X**


符号	位	类型	描述
<b>RB_VALUE</b>	[7:0]	rh	接收数据寄存器值 RB 中存放接收数据 RB_VALUE，RB 中未被选中的位无效，不予理睬。

## 20 LED 和触摸感应控制器

本章将对 LEDTSCU 模块予以详细说明。

### 20.1 概述

LED 和触摸感应控制单元（LEDTSCU）提供多达 8 个行引脚和 8 个列引脚，在单个引脚上实现 LED 驱动和触摸盘（touchpad）感应的时分复用控制。

LEDTSCU 提供优化的硬件控制用于列使能和功能选择；需由软件控制每个列时间片的行使能。

#### 特性

LEDTSCU 支持 LED 驱动功能，具有以下特性：

- 多达 8 列 LED 选择；若触摸感应功能使能，最多 7 列 LED 选择；
- 每列 LED 的时间片长度和时间片内的有效时间可配置
- 每个时间片（每列）可驱动多达 8 个 LED，共阴极或共阳极
- 对每个列时间片的行序列进行映射传送
- 每个时间片产生中断
- 通过 SFR 控制行引脚和列引脚

LEDTSCU 支持按盘感应功能，具有以下特性：

- 多达 8 个按盘输入
- 可由软件控制 pad turn（按盘时帧），也可完全由硬件通过循环的方式控制 pad turn
- 按盘振荡控制电路可灵活配置（如振荡频率和有效时间控制）
- 8 位计数器：对引脚上的振荡周期进行计数
- 和 LED 功能共用时间片长度配置，时间片内的有效时间可配置
- 每个时间片和时帧产生中断
- 触摸感应功能下的引脚硬件控制

**注：** 本章在提到 LED 或触摸感应引脚时，多处用“引脚 COL[x]”，“TSIN[x]”表示。它们实际上是由用户通过引脚复用功能选择位域 ALTSEL 进行配置的，详见[章节 20.9](#)的描述。

## 20.2 系统信息

本节给出 LEDTSCU 的相关系统信息。

### 20.2.1 引脚配置

**表 20-1** 说明如何使能 / 选择特定的 LED 或触摸感应引脚功能。

**表 20-1 XC83x 引脚功能和选择**

引脚	功能	描述	通过以下设置选择
P0.0	TSIN0/LINE0	触摸感应输入 0/ LED 行 0	P0_ALTSEL0.P0 = 1 <sub>B</sub> P0_ALTSEL1.P0 = 0 <sub>B</sub>
P0.1	TSIN1/LINE1	触摸感应输入 1/ LED 行 1	P0_ALTSEL0.P1 = 1 <sub>B</sub> P0_ALTSEL1.P1 = 0 <sub>B</sub>
P0.2	TSIN2/LINE2	触摸感应输入 2/ LED 行 2	P0_ALTSEL0.P2 = 1 <sub>B</sub> P0_ALTSEL1.P2 = 0 <sub>B</sub>
P0.3	TSIN3/LINE3	触摸感应输入 3/ LED 行 3	P0_ALTSEL0.P3 = 1 <sub>B</sub> P0_ALTSEL1.P3 = 0 <sub>B</sub>
P0.4	TSIN4/LINE4	触摸感应输入 4/ LED 行 4	P0_ALTSEL0.P4 = 1 <sub>B</sub> P0_ALTSEL1.P4 = 0 <sub>B</sub> P0_ALTSEL2.P4 = 0 <sub>B</sub>
P0.5	TSIN5/LINE5	触摸感应输入 5/ LED 行 5	P0_ALTSEL0.P5 = 1 <sub>B</sub> P0_ALTSEL1.P5 = 0 <sub>B</sub> P0_ALTSEL2.P5 = 0 <sub>B</sub>
P0.6	TSIN6/LINE6	触摸感应输入 6/ LED 行 6	P0_ALTSEL0.P6 = 1 <sub>B</sub> P0_ALTSEL1.P6 = 0 <sub>B</sub> P0_ALTSEL2.P6 = 0 <sub>B</sub>
P0.7	TSIN7/LINE7	触摸感应输入 7/ LED 行 7	P0_ALTSEL0.P7 = 1 <sub>B</sub> P0_ALTSEL1.P7 = 0 <sub>B</sub> P0_ALTSEL2.P7 = 0 <sub>B</sub>
P1.0	COL0_0	LED 列 0	P1_ALTSEL0.P0 = 1 <sub>B</sub> P1_ALTSEL1.P0 = 0 <sub>B</sub>
P0.4	COL0_1	LED 列 0	P0_ALTSEL0.P4 = 1 <sub>B</sub> P0_ALTSEL1.P4 = 0 <sub>B</sub> P0_ALTSEL2.P4 = 1 <sub>B</sub>
P1.1	COL1_0	LED 列 1	P1_ALTSEL0.P1 = 1 <sub>B</sub> P1_ALTSEL1.P1 = 0 <sub>B</sub>

**LED 和触摸感应控制器**
**表 20-1 XC83x 引脚功能和选择**

引脚	功能	描述	通过以下设置选择
P0.5	COL1_1	LED 列 1	P0_ALTSEL0.P5 = 1 <sub>B</sub> P0_ALTSEL1.P5 = 0 <sub>B</sub> P0_ALTSEL2.P5 = 1 <sub>B</sub>
P1.2	COL2_0	LED 列 2	P1_ALTSEL0.P2 = 1 <sub>B</sub> P1_ALTSEL1.P2 = 0 <sub>B</sub>
P0.6	COL2_1	LED 列 2	P0_ALTSEL0.P6 = 1 <sub>B</sub> P0_ALTSEL1.P6 = 0 <sub>B</sub> P0_ALTSEL2.P6 = 1 <sub>B</sub>
P1.3	COL3_0	LED 列 3	P1_ALTSEL0.P3 = 1 <sub>B</sub> P1_ALTSEL1.P3 = 0 <sub>B</sub>
P0.7	COL3_1	LED 列 3	P0_ALTSEL0.P7 = 1 <sub>B</sub> P0_ALTSEL1.P7 = 0 <sub>B</sub> P0_ALTSEL2.P7 = 1 <sub>B</sub>
P0.4	COL3_2	LED 列 3	P0_ALTSEL0.P4 = 0 <sub>B</sub> P0_ALTSEL1.P4 = 1 <sub>B</sub> P0_ALTSEL2.P4 = 1 <sub>B</sub>
P1.4	COL4	LED 列 4	P1_ALTSEL0.P4 = 1 <sub>B</sub> P1_ALTSEL1.P4 = 0 <sub>B</sub>
P1.5	COL5	LED 列 5	P1_ALTSEL0.P5 = 1 <sub>B</sub> P1_ALTSEL1.P5 = 0 <sub>B</sub>
P3.0	COL6	LED 列 6	P3_ALTSEL0.P0 = 1 <sub>B</sub> P3_ALTSEL1.P0 = 0 <sub>B</sub>
P3.1	COLA_0	触摸感应外部上拉 / LED 列 A	P3_ALTSEL0.P1 = 1 <sub>B</sub> P3_ALTSEL1.P1 = 0 <sub>B</sub>
P1.5	COLA_1	触摸感应外部上拉 / LED 列 A	P1_ALTSEL0.P5 = 1 <sub>B</sub> P1_ALTSEL1.P5 = 1 <sub>B</sub>
P0.6	COLA_2	触摸感应外部上拉 / LED 列 A	P0_ALTSEL0.P6 = 0 <sub>B</sub> P0_ALTSEL1.P6 = 1 <sub>B</sub> P0_ALTSEL2.P6 = 1 <sub>B</sub>
P0.7	COLA_3	触摸感应外部上拉 / LED 列 A	P0_ALTSEL0.P7 = 0 <sub>B</sub> P0_ALTSEL1.P7 = 1 <sub>B</sub> P0_ALTSEL2.P7 = 1 <sub>B</sub>
P0.4	COLA_4	触摸感应外部上拉 / LED 列 A	P0_ALTSEL0.P4 = 1 <sub>B</sub> P0_ALTSEL1.P4 = 1 <sub>B</sub> P0_ALTSEL2.P4 = 1 <sub>B</sub>

## 20.2.2 时钟配置

系统为 LED&TSCU 内核配有两种时钟源：FPCLK（48 MHz）和 SPCLK（8 MHz）。内核主要以 FPCLK（48 MHz）时钟工作。LEDTS 计数器可由用户配置，从而以 FPCLK 或 SPCLK 的不同分频时钟工作。触摸感应计数器及相应的逻辑以 FPCLK 对按键振荡周期进行计数和处理。

若彻底不需要 LEDTSCU 功能，可关闭其时钟输入以禁用该模块，从而最大程度降低功耗，通过置位寄存器 PMCON1 中的位 LTS\_DIS 来实现（描述见下页）。

在访问 PMCON1 寄存器之前必须先对 SCU\_PAGE 寄存器中的位域 PAGE 进行设置。

### PMCON1

外设管理控制寄存器 1

(EF<sub>H</sub>)

复位值：FF<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	CDC_DIS	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
LTS_DIS	6	rw	LEDTSCU 禁用请求，高有效 0 LEDTSCU 正常工作 1 请求禁用 LEDTSCU（缺省设置）
0	5	r	保留 读取返回 0；应写入 0。

## 20.2.3 中断事件和中断节点分配

LEDTSCU 模块产生的中断事件以及相应的中断使能位和标志位归纳见表 20-2。

表 20-2 LEDTSCU 中断事件

事件	事件中中断使能位	事件标志位
时间片开始	GLOBCTL1.ITS_EN	GLOBCTL1.TSF
时帧开始	GLOBCTL1.ITF_EN	GLOBCTL1.TFF

表 20-3 给出 LEDTSCU 中断事件的中断节点分配情况。

表 20-3 LEDTSCU 中断事件的中断节点控制

事件	中断节点使能位	中断节点标志位	中断向量地址
时间片开始	IEN1.ECCIP3	–	6B <sub>H</sub>
时帧开始	IEN1.ECCIP1	–	5B <sub>H</sub>

## 20.2.4 IP 互连

LEDTSCU 和其他外设的硬件互连使得无需软件干预、自动化的程度更高。

表 20-4 LEDTSCU 互连

LEDTSCU 功能 / 信号	所连接的其它模块功能 / 信号
比较匹配（输出）：LEDTS_CM	ADC 外部触发（输入）：REQTR0G, REQTR1G
时间片中断（输出）：LEDTS_TSI	ADC 外部触发（输入）：REQTR0H, REQTR1H

## 20.2.5 调试挂起控制

调试监控模式有效时，LEDTSCU 定时器 / 计数器、LEDTS 计数器和 TS 计数器可被（共同）使能挂起操作。可通过寄存器 MODSUSP 配置调试挂起控制，具体描述参见 [章节 10.2.4](#)。

调试挂起后，这些计数器停止计数（保持最后的计数值）。

## 20.3 引脚上 LED 和触摸感应功能的时分复用控制

单个引脚支持 LED 和触摸感应功能的时分复用控制。硬件提供 LED 模式或触摸感应模式的功能使能控制。

在每个时帧内，最多 8 个时间片可配置 – 最多有 1 个用于触摸感应功能、8 个用于 LED 功能。若使能触摸感应，最后一个时间片用于该功能，此时 pad turn 有效的引脚所对应的按键振荡电路将被使能。若使能 LED 功能，时帧内的其余时间片用于该功能（一次使能一列 LED）。

由于每个时帧内只有一个时间片可用于触摸感应功能，缺省状态下，硬件会根据用户配置的触摸感应输入的个数（pad turns）自动使能每个按键振荡器（pad\_turn\_x）并以循环的方式感应各个引脚。此外，也可以由用户通过软件来控制 pad turn 有效。若禁用触摸感应功能，在最后一个时间片内 pad turn 无效。

一个时帧由多个时间片组成，时间片的长度和个数可配置。若使能触摸感应功能的 pad turn 硬件自动控制，多个时帧组成一个时段，时段内应包括所有 pad\_turn。LED 和 / 或触摸感应功能对应的时间片的长度相同，通过 LEDTS 计数器设置，详见 [章节 20.4](#)，[章节 20.8](#) 和 [图 20-2](#)。

## LED 和触摸感应控制器

若使能时间片中断，LEDTS 计数器的 8LSB 溢出时触发该中断，从而启动每个新时间片。也可使能时帧中断，整个 LEDTS 计数器溢出时触发该中断。

为了使 LED 列激活时间和 / 或触摸感应振荡计数时间更加灵活，每个时间片内的列使能和按盘振荡使能的占空比均可调。

**图 20-1** 给出 LED 矩阵加按盘的示例。该示例的配置为 8 X 4 LED 矩阵，有 4 个 pad turn 由硬件依次使能（共有 6 个按盘）。该示例中，四个时帧构成一个时段。

产生时间片中断后，软件可以：

- 设置下个时间片的行值
- 设置下个时间片的比较值

可根据 [章节 20.7](#) 中 **位域 FNCOL 的说明** 以确定当前有效的时间片。

一次时帧中断指示一个触摸输入线已被感应，从而应用软件可以：

- 启动触摸感应分析程序并更新状态
- 使能 LED 显示更新



## LED 和触摸感应控制器

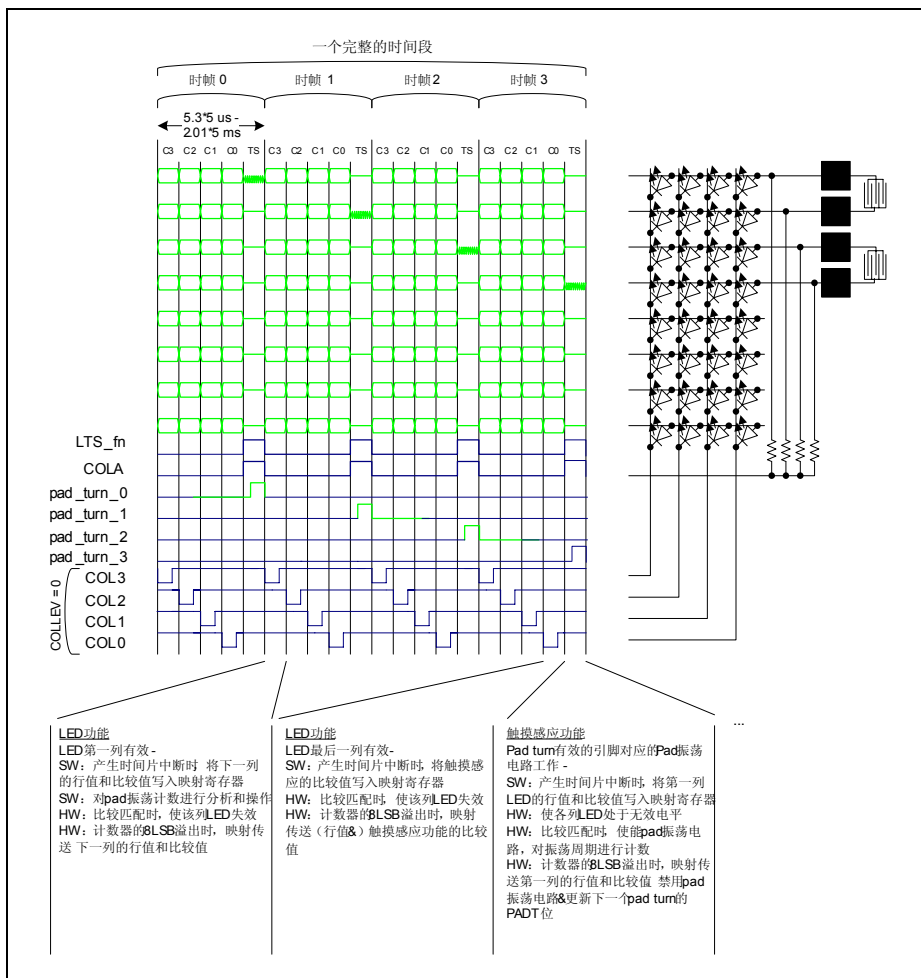


图 20-1 引脚上时分复用 LEDTSCU 功能 (示例)

## 20.4 LED 驱动

LED 驱动的硬件控制主要是 LED 列选择 – 一次使能一列。需要通过基于中断的软件处理（中断服务程序）来使能选中列工作。最多支持 8 列 LED，每列最多可使能 8 个 LED。

该模块可灵活调整 LED 的亮度以满足不同类型 LED 的要求。通过 LEDTS 计数器的比较寄存器来调整每个时间片内列使能的占空比。从时间片开始到发生比较匹配事件期间，选中列被使能。若 LED 列使能的占空比为 100%，比较值应设置为  $FF_H$ 。若比较值设置为  $00_H$ ，选中列在该时间片内将始终保持无效电平。通过映射传送更新每个时间片的比较寄存器，每个新时间片开始时自动进行映射传送。

此外，还可通过映射传送更新每列 LED（时间片）的 LED 使能控制（LED 行序列）。每个新时间片开始时自动进行该映射传送。

首次启动 LEDTS 计数器工作时（由 CLK\_PS 使能输入时钟），激活第一列的行序列和比较值的映射传送。

当 LEDTS 计数器的 8LSB 溢出时，可产生时间片中断，用于指示新的时间片开始。

LED 控制电路如图 20-2 所示，该图同时示出按键振荡器使能控制。控制电路的时钟源可选：FPCLK（48 MHz）或 SPCLK（8 MHz）。可通过一个 6 位分频器灵活配置 LEDTS 计数器的计数率，计数器溢出时一个时帧结束。在由多个时间片（个数可配置）组成的时帧内，各列 LED 被依次激活。在时帧内的最后一个时间片激活触摸感应功能（若使能该功能）。每个时帧由长度相等的多个时间片组成，时间片长度可配置（5.3 us 到 2.01 ms 之间）。

当位 CLK\_PS 从 0 设置为其它值且 LED 或触摸感应功能中至少有一种功能被使能时，LEDTS 计数器开始工作。两种功能均被禁用时 LEDTS 计数器不工作。计数器工作期间，为了避免过度设置功能使能从而干扰硬件控制，只有在 CLK\_PS = 0 时才能修改位 TS\_EN 和 LD\_EN。不过，当 CLK\_PS 从 0 设置为 1、或从 1 设置为 0 时，可对 SFR GLOBCTL0 进行一次写操作来置位 TS\_EN 和 LD\_EN。

计数器启动后，根据所使能的功能从复位值 / 重载值开始计数：1) 使能 LED 功能时，计数个数为 LED 列数（位域 NR\_LEDCOL）；2) 使能触摸感应功能时，在时帧最后增加一个时间片。计数器始终递增计数，从 7FFH 溢出后从重载值（等于复位值）开始重新计数。在每个时帧内，LED 列使能的顺序始终从最高使能列开始（从高向低）。下面以四列 LED 举例说明：

- 从 COL3 开始，
- 随后是 COL2，
- 随后是 COL1，
- 最后是 COL0。

若不使能触摸感应功能，COLA 始终用于控制 LED 功能的最后一列 LED，举例说明：

- 从 COL2 开始，
- 随后是 COL1，
- 随后是 COL0，
- 最后是 COLA。

LED 和触摸感应控制器

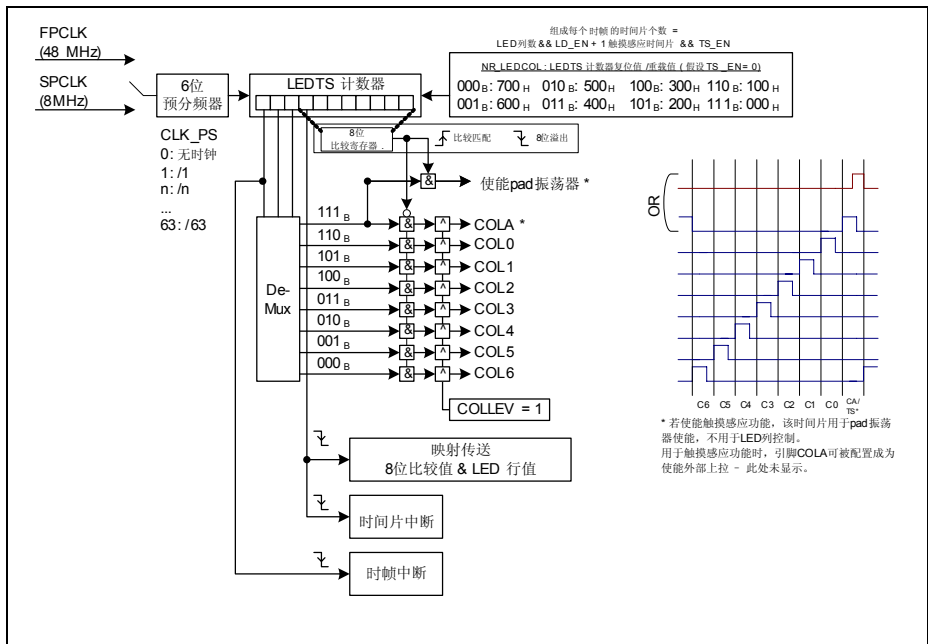


图 20-2 LED 功能控制电路（包括按键振荡器使能控制）

有关时间片以及 LEDTSCU 相关的时序计算，请参见[章节 20.8](#)。

### 20.4.1 LED 引脚分配和电流驱动能力

一个时间片内只能使能一个 LED 列引脚，最多控制 8 个 LED。COL[x] 的引脚分配可灵活配置以满足不同的应用需要。该器件支持大电流引脚，有关 LED 引脚的电流驱动能力，可参见产品数据手册。例如：某些引脚的灌电流为 40 mA。这意味着，若一列直接驱动 8 个 LED，每个 LED 的驱动能力限于 5 mA。

在任何情况下，一次直接驱动（拉或灌）所有 8 个 LED 的总电流绝不能超过所规定的各引脚的最大电流（ $\Sigma I_M$ ）。若需要更高的 LED 驱动能力，将要用到外部晶体管。

## 20.5 触摸盘感应

图 20-3 引脚振荡控制电路如图 20-3 所示，该部分已与标准 GPIO 焊盘集成在一起。每个输入引脚 TSIN[x] 对应一个 pad turn (pad\_turn\_x)，在触摸感应时间片内，pad turn 有效期间 TS 计数器工作。

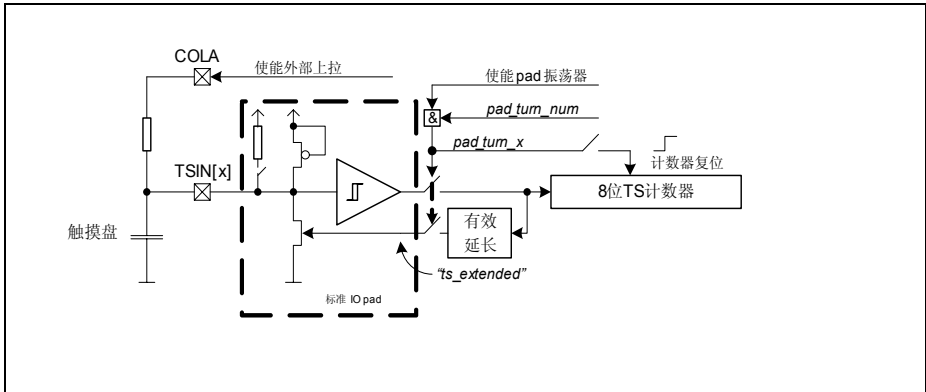


图 20-3 触摸感应振荡器控制电路

8 位 TS 计数器对振荡周期进行计数。只有当 pad turn 无效时才能设置该计数器。每次激活新 pad turn 时，可使能 TS 计数器自动复位（到 00<sub>H</sub>）。位 TSCTROVF 指示计数器已溢出。此外，也可按下面的方式配置：计数器在当前的时间片内计数饱和后停止工作，即不溢出 & 停留在计数值 FF<sub>H</sub>。在这种情况下，只有当新 pad turn 有效时 TS 计数器才重新开始工作。

该模块支持引脚低电平有效延长（时间可配置），从而可根据用户系统的需要来调整按盘振荡。对按盘振荡的放电阶段进行延长，可延长一个或四个 FPCLK 周期。图 20-4 对该功能予以说明。

在使能振荡电路工作的有效时间内，有效触摸感应引脚 TSIN[x] 的按盘配置由硬件控制，具体参见章节 20.9。特别是，当用户系统采用外部电阻上拉时，可选择禁止内部弱上拉。在整个触摸感应时间片内，COLA 保持高电平，从而使与 COLA 相连的外部电阻上拉有效。这样的配置可在一定程度上灵活调整按盘的振荡频率以满足用户系统的需要。

在使能的 LINE[x]/TSIN[x] 引脚上，触摸感应和 LED 功能时分复用。在触摸感应时间片内，对于其它 pad turn 无效的 TSIN 引脚，其对应的 LINE 输出保持有效。软件应负责将行位置 1 以避免从引脚 COLA 灌入电流。

时帧内的最后一个时间片用于触摸感应功能，有关时间片分配和配置的详细说明，请参见章节 20.3 和章节 20.4。

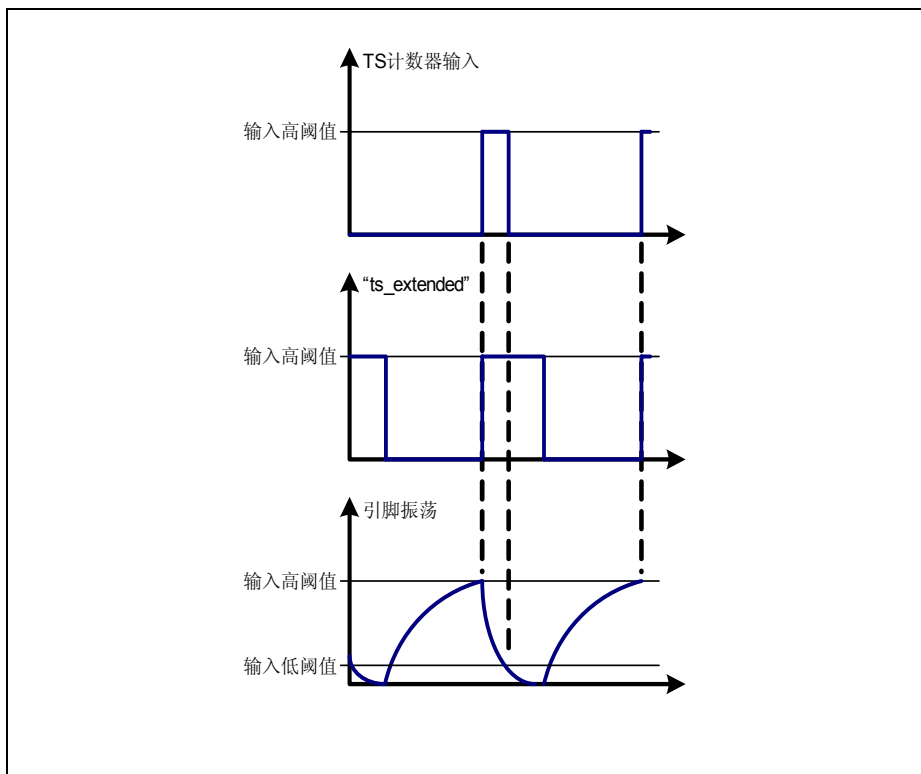


图 20-4 引脚低电平延长功能

对于 Pad turn 有效的引脚，其按键振荡电路使能工作的时间长短可由用户配置。通过一个比较值来调整时间片内按键振荡电路使能工作（TS 计数器工作）的占空比。只有当发生比较匹配时，按键振荡电路才使能开始工作，直到时间片结束时停止。若时间片内按键振荡使能的占空比为 100%，比较值应设为 00H；比较值设为 FFH 时，按键振荡电路不工作。

可根据需要使能时间片中断和 / 或时帧中断用于触摸感应控制。

### 20.5.1 手指感应

将手指放在触摸盘上时，按键的电容增大，相应的振荡频率将降低。手指不触摸时，按键的振荡频率通常介于 0.25 MHz - 0.5 MHz 之间；手指触摸时，按键的振荡频率通常介于 0.125 MHz - 0.25 MHz 之间。正如之前的描述，用户系统可在一定程度上灵活调整按键振荡：1) 引脚低电平有效延长时间可配置；2) 用户可选择采用外部电阻上拉。时间片的长度可配置介于 5.3  $\mu$ s (0.18 MHz) - 2.01 ms (496 Hz) 之间，还可通过软件配

---

**LED 和触摸感应控制器**

置 **pad turn** 的有效时间（在时间片内可调，通过比较功能实现），并设置一个振荡计数阈值用以检测手指控制是否有效。

## 20.6 引脚上 LED 和触摸感应功能时分复用

在引脚上时分复用 LED 和触摸感应功能时，需要注意：

- 若使能触摸感应功能，LED 最多有 7 列。
- 在触摸感应功能下，COLA 输出高电平使得可通过外部电阻上拉（若引脚使能该特性）。
- 在触摸感应时间片内，建议设置 LED 行输出高电平。
- 在 LED 时间片内，COLA 输出低电平。若相连的 LED 行输出高电平，将会向 COLA 灌入电流。
- 每个 TSIN 引脚的有效电容大小和与之相连的器件以及应用板的设计关系紧密。应该对实际应用中的所有触摸盘进行校正以实现稳健的触摸检测。

## 20.7 功能使能和控制的注意事项

建议用户在设置 SFR **LTS\_GLOBCTL0** 启动 LED 和 / 或 TS 功能之前，首先设置所有与 LEDTSCU 相关的寄存器中的配置位。

注：只有当计数器不工作，即 **CLK\_PS = 0** 时，才能修改和 LED/ 触摸感应功能相关的 SFR（特别是影响 LEDTS 计数器配置的 SFR）。详见 [章节 20.11](#) 中 SFR 的描述。

### 仅使能 LED 功能

仅使能 LED 功能：置位 LD\_EN，清除 TS\_EN。

复位后进行初始化：

```
MOV LTS_GLOBCTL0, #0b10XXXXXX
    ; 置位 LD_EN 并启动 LEDTS 计数器以分频时钟计数
    ; (CLK_PS != 0)
```

在工作期间重新配置：

```
MOV LTS_GLOBCTL0, #0x00; 清零时钟分频因子以终止 LEDTS 计数器工作
MOV LTS_GLOBCTL0, #0b10XXXXXX
```

### 仅使能触摸感应功能

仅使能触摸感应功能：清除 LD\_EN，置位 TS\_EN。

复位后进行初始化：

```
MOV LTS_GLOBCTL0, #0b01XXXXXX
    ; 置位 TS_EN 并启动 LEDTS 计数器以分频时钟计数
    ; (CLK_PS != 0)
```

在工作期间重新配置：

```
MOV LTS_GLOBCTL0, #0x00; 清零时钟分频因子以终止 LEDTS 计数器工作
MOV LTS_GLOBCTL0, #0b01XXXXXX
```

### 使能 LED 和触摸感应两种功能

使能两种功能：置位 LD\_EN 和 TS\_EN。

复位后进行初始化：

```
MOV LTS_GLOBCTL0, #0b11XXXXXX
    ; 置位 LD_EN 和 TS_EN 并启动 LEDTS 计数器以分频时钟计数
    ; (CLK_PS != 0)
```

在工作期间重新配置：

```
MOV LTS_GLOBCTL0, #0x00; 清零时钟分频因子以终止 LEDTS 计数器工作
MOV LTS_GLOBCTL0, #0b11XXXXXX
```



### 位域 FNCOL 的说明

在每个时间片内，由软件更新下个时间片将使用的行值和比较值。位域 FNCOL 给出前一个时间片内的有效功能 / 列。软件可根据该信息决定当前时间片内的有效功能 / 列，并为下一个时间片准备（待映射传送的）数值。

下面的示例中一个时帧内包含六个时间片：五列 LED+ 触摸感应功能。

**表 20-5 FNCOL 位域的说明**

FNCOL	当前时间片的有效功能 / 列	下个时间片的有效功能 / 列（由 SW 通过映射寄存器进行准备）
111 <sub>H</sub>	LED COL[4]	LED COL[3]
010 <sub>H</sub>	LED COL[3]	LED COL[2]
011 <sub>H</sub>	LED COL[2]	LED COL[1]
100 <sub>H</sub>	LED COL[1]	LED COL[0]
101 <sub>H</sub>	LED COL[0]	触摸感应 TSIN[PADT]
110 <sub>H</sub>	触摸感应 TSIN[PADT]	LED COL[4]

### 20.8 LEDTSCU 时序计算

LEDTSCU 时序计算的公式如下：

计数率（CR）：

$$CR = (fCLK) \div (PREscaler) \tag{20.1}$$

时间片长度（TSD）：

$$TSD = 2^8 \div (CR) \tag{20.2}$$

时帧长度：

$$TimeFrameDuration = (Number\ of\ time\ slice) \times TSD \tag{20.3}$$

LED 驱动的有效时间：

$$LED\ Drive\ Active\ Duration = TSD \times Compare\_VALUE \div 2^8 \tag{20.4}$$

触摸感应的有效时间:

(20.5)

$$\text{Touch-sense Drive Active Duration} = \text{TSD} \times (2^8 - \text{Compare\_VALUE}) \div 2^8$$

### 20.9 LEDTSCU 引脚控制

用户可通过 GPIO-SFR ALTSEL 为 LEDTSCU 功能灵活的分配引脚：

- COL[x] （用于 LED 列控制）
- LINE[x]/TSIN[x] （用于 LED 行控制或触摸感应）

根据用户配置选择激活哪个 COL[x] 和 / 或 LINE[x]/TSIN[x] 时，可参考[章节 20.4](#)的描述。

引脚用作 LED 功能时，用户必须通过 GPIO-SFR ALTSEL 对其进行配置。引脚用作触摸感应功能时，LEDTSCU 为 pad turn 有效的引脚提供硬件控制的同时，还需通过 ALTSEL 选择 TSIN （和 COLA）功能，可参见[表 20-6](#) 和 [图 20-5](#)。

**表 20-6 LEDTSCU 引脚控制信号**

功能	LD/TS_en	LTS_fn	引脚	引脚控制
LED 列	LD_EN = 1	0 = LED	使能 COL[x] ； COL[ 其余 ] 处于无效电平。 若 TS_EN = 1， COLA = 0。	由 GPIO SFR 设置
LED 行	LD_EN = 1	0 = LED	LINE[x] = 从 LDLINE 映射传送的值	由 GPIO SFR 设置
触摸感应	TS_EN = 1	1 = 触摸感应	使能 TSIN[x] 启动振荡电路。 所有其它 TSIN 引脚输出行值。  除 COLA = 1 之外， COL[ 其余 ] 处于无效电平。	由硬件控制 pad_turn_x <sup>1)</sup> ： - 使能上拉 （可由位 EPULL 禁止该控制） - 使能漏极开路

1) 对于 pad turn 无效的其它引脚，则无需硬件控制，仍采用 GPIO SFR 设置。

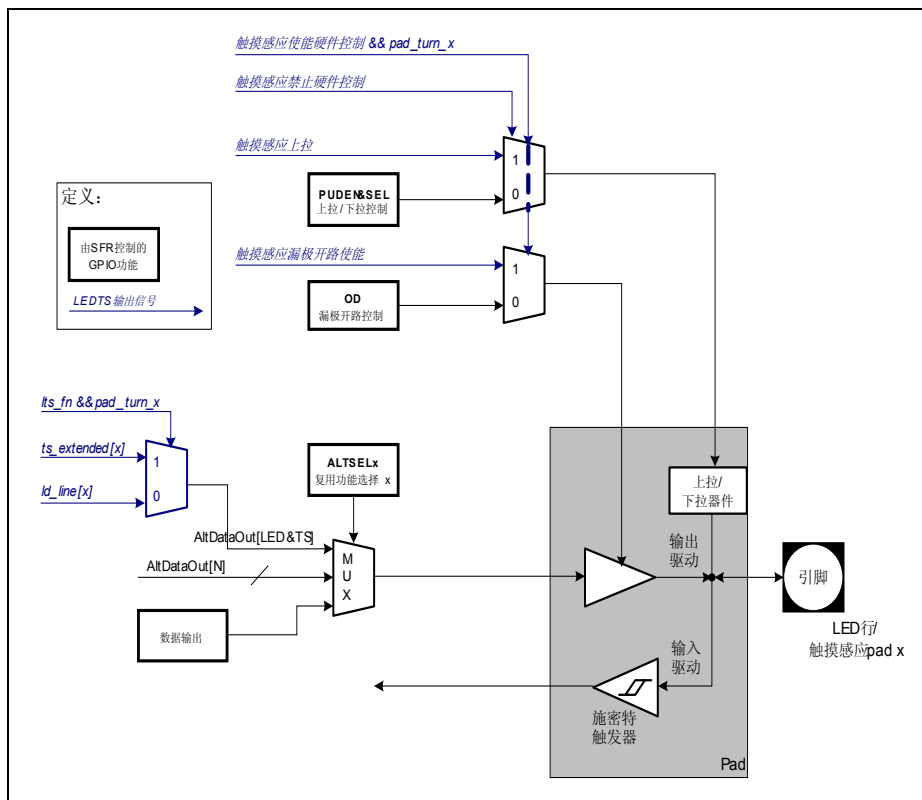


图 20-5 触摸感应功能中按盘输入引脚的硬件控制

## 20.10 中断

LEDTSCU 内核可触发两种中断：1) 时间片事件；2) 时帧事件。发生这两种事件、或 CLK\_PS 从 0 设置为其它值时，无论相应的中断是否被使能，标志位均置位。若中断被使能，时间片 / 时帧事件（以及 CLK\_PS 从 0 设置为其它值的操作）将激活相应的中断请求。

### 20.11 寄存器描述

从标准（非映射）SFR 区访问 **LEDTSCU** 的特殊功能寄存器。LEDTSCU SFR 和相应的地址归纳见**表 20-7**。

**表 20-7 寄存器映射**

地址	寄存器
97 <sub>H</sub>	LTS_GLOBCTL0
D4 <sub>H</sub>	LTS_COMPARE
D5 <sub>H</sub>	LTS_LDLINE
D6 <sub>H</sub>	LTS_LDTSCTL
D7 <sub>H</sub>	LTS_TSCTL
D8 <sub>H</sub>	LTS_GLOBCTL1
D9 <sub>H</sub>	LTS_TSVAL

## LED 和触摸感应控制器

## 20.11.1 全局控制和状态

LEDTSCU 有三个全局控制和状态寄存器。

## LTS\_GLOBCTL0

全局控制寄存器 0

(97<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
LD_EN	TS_EN	CLK_PS					
rw	rw	rw					

符号	位	类型	描述
CLK_PS	[5:0]	rw	<b>LEDTS 计数器时钟分频因子</b> 根据设置对输入时钟 FPCLK 或 SPCLK 分频: 0 <sub>D</sub> 无时钟 1 <sub>D</sub> 1 分频 n <sub>D</sub> n 分频 63 <sub>D</sub> 63 分频 只有当至少使能触摸感应或 LED 其中一种功能时, 才能将该位 (从 0) 设置为其它值。LEDTS 计数器基于所使能的功能 (以及 NR_LEDCOL) 从复位值 / 重载值开始以输入时钟计数, 详见 <a href="#">章节 20.4</a> 的描述。 将该位清零时, LEDTS 计数器停止计数并复位。
TS_EN <sup>1)</sup>	6	rw	<b>触摸感应功能使能</b> 当 CLK_PS 从 0 设置为其它值时, 置位该位以使能触摸感应功能。
LD_EN	7	rw	<b>LED 功能使能</b> 当 CLK_PS 从 0 设置为其它值时, 置位该位以使能 LED 功能。

1) 只有当 CLK\_PS = 0 时才能修改该位。

## LED 和触摸感应控制器

## LTS\_GLOBCTL1

全局控制寄存器 1

RMAP: 0, PAGE: X

(D8<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
TSF	ITS_EN	TFF	ITF_EN	CLKSEL	FNCOL		
rwh	rw	rwh	rw	rw	rh		

符号	位	类型	描述
FNCOL	[2:0]	rh	先前的有效功能 /LED 列状态 指示前一个时间片的有效功能 /LED 列状态。当 LEDTS 计数器的 8LSB 溢出、开始下个新时间片时，对该位进行更新。由内部 DE-MUX 的锁存值控制，请参见图 20-2。
CLKSEL <sup>1)</sup>	3	rw	LEDTS 计数器时钟输入选择 0 <sub>B</sub> 选择 48 MHz 1 <sub>B</sub> 选择 8 MHz
ITF_EN	4	rw	使能时帧中断 0 <sub>B</sub> 禁止 1 <sub>B</sub> 使能
TFF	5	rwh	时帧中断标志 每次开始一个新时帧（以及 CLK_PS 从 0 设置为其它值）时，该位置位。由软件清零。
ITS_EN	6	rw	使能时间片中断 0 <sub>B</sub> 禁止 1 <sub>B</sub> 使能
TSF	7	rwh	时间片中断标志 每次开始一个新时间片（以及 CLK_PS 从 0 设置为其它值）时，该位置位。由软件清零。

1) 只有当 CLK\_PS = 0 时才能修改该位。

**LTS\_COMPARE**

时间片比较映射寄存器

**(D4<sub>H</sub>)**

 复位值: **00<sub>H</sub>**
**RMAP: 0, PAGE: X**

7	6	5	4	3	2	1	0
<b>SHD_CMP</b>							
rw							

符号	位	类型	描述
<b>SHD_CMP</b>	[7:0]	rw	待映射传送的时间片比较值 每次开始一个新时间片时，该值被映射传送到时间片比较寄存器中。 映射传送和 CLK_PS 的设置有关。



## LED 和触摸感应控制器

## 20.11.2 功能控制寄存器

这些寄存器用于控制 LED 驱动和触摸感应功能。

LTS\_LDTSCCTL

LED 和触摸感应控制寄存器

(D6<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
NR_LEDCOL			COLLEV	NR_PADT			TSOEXT
rw			rw	rw			rw

符号	位	类型	描述
TSOEXT	0	rw	引脚低电平有效延长 0 <sub>B</sub> 延长 1 FPCLK 1 <sub>B</sub> 延长 4 FPCLK
NR_PADT <sup>1)</sup>	[3:1]	rw	触摸感应 Pad Turn 的个数 该位域定义触摸感应输入的个数（和 pad turn 的个数相等），用于 pad turn 使能的硬件控制。 0 <sub>D</sub> 1 n <sub>D</sub> n+1
COLLEV	4	rw	LED 列的有效电平 0 <sub>B</sub> 低有效 1 <sub>B</sub> 高有效
NR_LEDCOL	[7:5]	rw	LED 列数 定义 LED 列数。 000 <sub>B</sub> 1 列 LED 001 <sub>B</sub> 2 列 LED 010 <sub>B</sub> 3 列 LED 011 <sub>B</sub> 4 列 LED 100 <sub>B</sub> 5 列 LED 101 <sub>B</sub> 6 列 LED 110 <sub>B</sub> 7 列 LED 111 <sub>B</sub> 8 列 LED（若 TS_EN = 1，LED 最大列数 = 7）  注：从最高列开始依次使能各列 LED。若不使能触摸感应功能，在最后一个时间片激活 COLA。

1) 只有当 CLK\_PS = 0 时才能修改该位。

## LED 和触摸感应控制器

## LTS\_LDLINE

LED 行值映射寄存器

(D5<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
SHD_LINE							
rw							

符号	位	类型	描述
SHD_LINE	[7:0]	rw	待映射传送的 LED 行值 每次开始一个新时间片时，该值被映射传送给各行 LED。 映射传送和 CLK_PS 的设置有关。

## LTS\_TSCTL

触摸感应控制寄存器

(D7<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
TSCTROV F	TSCTRR	TSCTRSA T	EPULL	PADTSW	PADT		
rwh	rw	rw	rw	rw	rwh		

符号	位	类型	描述
PADT	[2:0]	rwh	触摸感应 Pad Turn （输入引脚） 该位为下一个或当前有效的 pad turn （触摸感应输入引脚）。若 PADTSW = 0，在触摸感应时间片结束时该位由硬件自动更新。始终可软件设置该位。 0 <sub>D</sub> TSIN0 n <sub>D</sub> TSIN[n]
PADTSW <sup>1)</sup>	3	rw	触摸感应 Pad Turn 软件控制 0 <sub>B</sub> 硬件以循环的方式从 TSIN0 开始自动使能各触摸感应输入引脚。 1 <sub>B</sub> 禁止硬件控制，只能软件控制。通过位 PADT 配置有效的触摸感应输入。

LED 和触摸感应控制器

符号	位	类型	描述
EPULL	4	rw	<p><b>使能引脚 COLA 的外部上拉配置</b></p> <p>该位置位时，禁止对有效的触摸感应输入引脚进行内部上拉。</p> <p>0<sub>B</sub> 在触摸感应时间片内的置位期间，使能硬件控制 TSIN[x] 引脚内部上拉。这种设置下，未指定将 COLA 分配给任何引脚。</p> <p>1<sub>B</sub> 使能外部上拉：在整个触摸感应时间片内引脚 COLA 输出 1。</p> <p><i>注： 无论该位如何设置，引脚 COLA 在整个触摸感应时间片内始终输出 1。</i></p>
TSCTRSAT	5	rw	<p><b>TS 计数器饱和</b></p> <p>0<sub>B</sub> 禁用。</p> <p>1<sub>B</sub> 使能。TS 计数器在当前时间片内计数到 FF<sub>H</sub> 后停止工作。当新 pad turn 有效时，计数器由比较匹配事件触发重新开始工作。</p>
TSCTRR	6	rw	<p><b>TS 计数器自动复位</b></p> <p>0<sub>B</sub> 禁止 TS 计数自动复位。</p> <p>1<sub>B</sub> 使能 TS 计数器自动复位，当新 pad turn 有效时，计数器由比较匹配事件触发自动复位到 00<sub>H</sub>。</p>
TSCTROVF	7	rwh	<p><b>TS 计数器溢出指示</b></p> <p>该位指示 TS 计数器是否已溢出。当新 pad turn 有效时，该位由比较匹配事件触发清零。</p> <p>0<sub>B</sub> 未发生溢出。</p> <p>1<sub>B</sub> TS 计数器至少已溢出一一次。</p>

1) 只有当 CLK\_PS = 0 时才能修改该位。

LTS\_TSVAL

触摸感应计数器值

(D9<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
TSCTRVAL							
rwh							

## LED 和触摸感应控制器

符号	位	类型	描述
TSCTRVAL	[7:0]	rwh	<b>TS 计数器的计数值</b> 该位域指示 TS 计数器的计数值。只有当 pad turn 无效时才能设置该寄存器。 可使能该计数器在新 pad turn 有效时自动复位。

## 21 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 CCU 模块的器件的重要特性。

本章结构如下：

- 简介（见[章节 21.1](#)）
  - 包括寄存器概览（见[章节 21.1.3](#)）
- 系统信息（见[章节 21.2](#)）
- 定时器 T12 的操作（见[章节 21.3](#)）
  - 包括与 T12 相关的寄存器（见[章节 21.3.8](#)）
  - 和捕获 / 比较控制寄存器（见[章节 21.3.9](#)）
- 定时器 T13 的操作（见[章节 21.4](#)）
  - 包括与 T13 相关的寄存器（见[章节 21.4.6](#)）
- 强制中断处理（见[章节 21.5](#)）
- 多通道模式（见[章节 21.6](#)）
- 霍尔传感器模式（见[章节 21.7](#)）
- 调制控制寄存器（见[章节 21.8](#)）
- 中断处理（见[章节 21.9](#)）
  - 包括中断寄存器（见[章节 21.9.2](#)）
- 一般模块操作（见[章节 21.10](#)）
  - 包括通用寄存器（见[章节 21.10.2](#)）

### 21.1 简介

CCU6 单元由带有三个捕获 / 比较通道的定时器 T12 模块和带有一个比较通道的定时器 T13 模块组成。T12 的各通道既能独立产生 PWM 信号或接受捕获触发，各通道也可共同产生驱动 AC 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**。

### 21.1.1 特性概述

本节给出 CCU6 的各组成模块及其主要特性。

#### 定时器 T12 模块特性

- 三个捕获 / 比较通道，可单独用作捕获或比较通道
- 支持三相 PWM 产生（6 个输出，每相两个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 T3 模块特性

- 一个独立的比较通道（具有一个输出）
- 16 位精度，最大计数频率 = 外设时钟频率
- T13 寄存器同步更新
- 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

#### 附加的特定功能

- 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（CTRAP）控制快速急停，无需 CPU 干预
- 多通道 AC 驱动的控制模式
- 输出电平可选，以配合功率器件

## 捕获 / 比较单元 6 (CCU6)

### 21.1.2 框图

定时器 T12 的三个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。

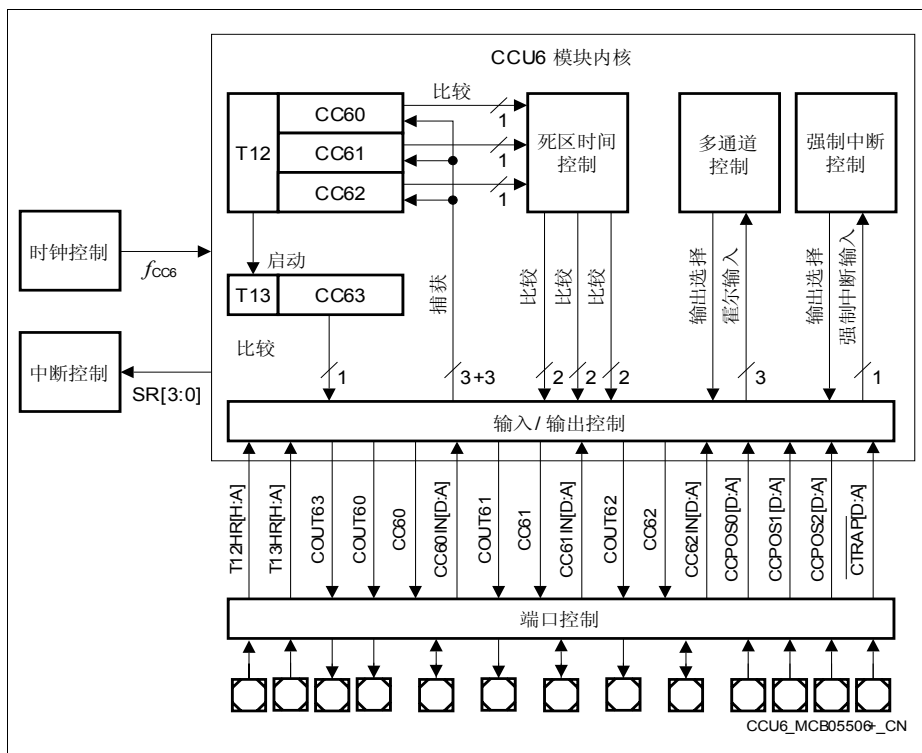


图 21-1 CCU6 框图

捕获 / 比较单元 6 (CCU6)

21.1.3 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。其中 x 代表模块编号。

**表 21-1** 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。8 位短地址在该模块中不可用。

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
T12L	CMPSTATL	ISL	PISEL0L
T12H	CMPSTATH	ISH	PISEL0H
T12PRL	CMPMODIFL	ISSL	PISEL2
T12PRH	CMPMODIFH	ISSH	
T12DTCL	T12MSELL	ISRL	
T12DTCH	T12MSELH	ISRH	
CC60RL	TCTR0L	INPL	
CC60RH	TCTR0H	INPH	
CC60SRL	TCTR2L	IENL	
CC60SRH	TCTR2H	IENH	
CC61RL	TCTR4L		
CC61RH	TCTR4H		
CC61SRL			
CC61SRH			
CC62RL			
CC62RH			
CC62SRL			
CC62SRH			
	调制控制寄存器		
	MODCTRL		
	MODCTRH		
	TRPCTRL		
	TRPCTRH		
	PSLR		
	MCMCTR		
	MCMOUTSL		
	MCMOUTSH		
	MCMOUTL		
	MCMOUTH		
T13相关寄存器			
T13L			
T13H			
T13PRL			
T13PRH			
CC63RL			
CC63RH			
CC63SRL			
CC63SRH			

图 21-2 CCU6 寄存器



## 捕获 / 比较单元 6 (CCU6)

表 21-1 CCU6 模块寄存器总结

寄存器 缩写名	描述	偏移 地址	复位值	见 页
<b>通用寄存器</b>				
<b>PISEL0L</b>	端口输入选择寄存器低位	9E <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-118</a>
<b>PISEL0H</b>	端口输入选择寄存器高位	9F <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-119</a>
<b>PISEL2</b>	端口输入选择寄存器 2	A4 <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-120</a>

## 定时器 T12 相关寄存器

<b>T12L</b>	定时器 T12 计数寄存器低位	FA <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-42</a>
<b>T12H</b>	定时器 T12 计数寄存器高位	FB <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-42</a>
<b>T12PRL</b>	定时器 T12 周期寄存器低位	9C <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-43</a>
<b>T12PRH</b>	定时器 T12 周期寄存器高位	9D <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-43</a>
<b>T12DTCL</b>	定时器 T12 死区时间控制寄存器低位	A4 <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-47</a>
<b>T12DTCH</b>	定时器 T12 死区时间控制寄存器高位	A5 <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-47</a>
<b>CC60RL</b>	通道 CC60 捕获 / 比较寄存器低位	FA <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-44</a>
<b>CC60RH</b>	通道 CC60 捕获 / 比较寄存器高位	FB <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-44</a>
<b>CC61RL</b>	通道 CC61 捕获 / 比较寄存器低位	FC <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-44</a>
<b>CC61RH</b>	通道 CC61 捕获 / 比较寄存器高位	FD <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-44</a>
<b>CC62RL</b>	通道 CC62 捕获 / 比较寄存器低位	FE <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-44</a>
<b>CC62RH</b>	通道 CC60 捕获 / 比较寄存器高位	FF <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-44</a>
<b>CC60SRL</b>	通道 CC60 捕获 / 比较映射寄存器低位	FA <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-45</a>
<b>CC60SRH</b>	通道 CC60 捕获 / 比较映射寄存器高位	FB <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-45</a>
<b>CC61SRL</b>	通道 CC61 捕获 / 比较映射寄存器低位	FC <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-45</a>
<b>CC61SRH</b>	通道 CC61 捕获 / 比较映射寄存器高位	FD <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-45</a>
<b>CC62SRL</b>	通道 CC62 捕获 / 比较映射寄存器低位	FE <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-45</a>
<b>CC62SRH</b>	通道 CC62 捕获 / 比较映射寄存器高位	FF <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-45</a>

## 捕获 / 比较控制寄存器

<b>CMPSTATL</b>	比较状态寄存器低位	FE <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-49</a>
<b>CMPSTATH</b>	比较状态寄存器高位	FF <sub>H</sub>	00 <sub>H</sub>	<a href="#">页 21-50</a>

## 捕获 / 比较单元 6 (CCU6)

表 21-1 CCU6 模块寄存器总结

寄存器缩写名	描述	偏移地址	复位值	见页
<b>CMPMODIFL</b>	比较状态修改寄存器低位	A6 <sub>H</sub>	00 <sub>H</sub>	页 21-51
<b>CMPMODIFH</b>	比较状态修改寄存器高位	A7 <sub>H</sub>	00 <sub>H</sub>	页 21-52
<b>T12MSELL</b>	T12 捕获 / 比较模式选择寄存器低位	9A <sub>H</sub>	00 <sub>H</sub>	页 21-53
<b>T12MSELH</b>	T12 捕获 / 比较模式选择寄存器高位	9B <sub>H</sub>	00 <sub>H</sub>	页 21-53
<b>TCTR0L</b>	定时器控制寄存器 0 低位	A6 <sub>H</sub>	00 <sub>H</sub>	页 21-55
<b>TCTR0H</b>	定时器控制寄存器 0 高位	A7 <sub>H</sub>	00 <sub>H</sub>	页 21-56
<b>TCTR2L</b>	定时器控制寄存器 2 低位	FA <sub>H</sub>	00 <sub>H</sub>	页 21-58
<b>TCTR2H</b>	定时器控制寄存器 2 高位	FB <sub>H</sub>	00 <sub>H</sub>	页 21-59
<b>TCTR4L</b>	定时器控制寄存器 4 低位	9C <sub>H</sub>	00 <sub>H</sub>	页 21-61
<b>TCTR4H</b>	定时器控制寄存器 4 高位	9D <sub>H</sub>	00 <sub>H</sub>	页 21-62

## 定时器 T13 相关寄存器

<b>T13L</b>	定时器 T13 计数寄存器低位	FC <sub>H</sub>	00 <sub>H</sub>	页 21-75
<b>T13H</b>	定时器 T13 计数寄存器高位	FD <sub>H</sub>	00 <sub>H</sub>	页 21-75
<b>T13PRL</b>	定时器 T13 周期寄存器低位	9E <sub>H</sub>	00 <sub>H</sub>	页 21-77
<b>T13PRH</b>	定时器 T13 周期寄存器高位	9F <sub>H</sub>	00 <sub>H</sub>	页 21-77
<b>CC63RL</b>	定时器 T13 比较寄存器低位	9A <sub>H</sub>	00 <sub>H</sub>	页 21-78
<b>CC63RH</b>	定时器 T13 比较寄存器高位	9B <sub>H</sub>	00 <sub>H</sub>	页 21-78
<b>CC63SRL</b>	定时器 T13 比较映射寄存器低位	9A <sub>H</sub>	00 <sub>H</sub>	页 21-79
<b>CC63SRH</b>	定时器 T13 比较映射寄存器高位	9B <sub>H</sub>	00 <sub>H</sub>	页 21-79

## 调制控制寄存器

<b>MODCTRL</b>	调制控制寄存器低位	FC <sub>H</sub>	00 <sub>H</sub>	页 21-92
<b>MODCTRH</b>	调制控制寄存器高位	FD <sub>H</sub>	00 <sub>H</sub>	页 21-93
<b>TRPCTRL</b>	强制中断控制寄存器低位	FE <sub>H</sub>	00 <sub>H</sub>	页 21-94
<b>TRPCTRH</b>	强制中断控制寄存器高位	FF <sub>H</sub>	00 <sub>H</sub>	页 21-95
<b>PSLR</b>	被动态电平等寄存器	A6 <sub>H</sub>	00 <sub>H</sub>	页 21-97
<b>MCMOUTSL</b>	多通道模式输出映射寄存器低位	9E <sub>H</sub>	00 <sub>H</sub>	页 21-99

表 21-1 CCU6 模块寄存器总结

寄存器缩写名	描述	偏移地址	复位值	见页
MCMOUTSH	多通道模式输出映射寄存器高位	9F <sub>H</sub>	00 <sub>H</sub>	页 21-99
MCMOUTL	多通道模式输出寄存器低位	9A <sub>H</sub>	00 <sub>H</sub>	页 21-100
MCMOUTH	多通道模式输出寄存器高位	9B <sub>H</sub>	00 <sub>H</sub>	页 21-101
MCMCTR	多通道模式控制寄存器	A7 <sub>H</sub>	00 <sub>H</sub>	页 21-98

### 中断状态和节点寄存器

ISL	中断状态寄存器低位	9C <sub>H</sub>	00 <sub>H</sub>	页 21-105
ISH	中断状态寄存器高位	9D <sub>H</sub>	00 <sub>H</sub>	页 21-106
ISL	中断状态置位寄存器低位	A4 <sub>H</sub>	00 <sub>H</sub>	页 21-108
ISSH	中断状态置位寄存器高位	A5 <sub>H</sub>	00 <sub>H</sub>	页 21-108
ISRL	中断状态复位寄存器低位	A4 <sub>H</sub>	00 <sub>H</sub>	页 21-110
ISRH	中断状态复位寄存器高位	A5 <sub>H</sub>	00 <sub>H</sub>	页 21-110
INPL	中断节点指针寄存器低位	9E <sub>H</sub>	40 <sub>H</sub>	页 21-115
INPH	中断节点指针寄存器高位	9F <sub>H</sub>	39 <sub>H</sub>	页 21-115
IENL	中断使能寄存器低位	9C <sub>H</sub>	0000 <sub>H</sub>	页 21-112
IENH	中断使能寄存器高位	9D <sub>H</sub>	0000 <sub>H</sub>	页 21-113

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，CCU6 模块不予响应。

## 21.2 系统信息

本节提供与 CCU6 相关的系统信息。

### 21.2.1 引脚

XC83x 的中的 CCU6 引脚分配见表 21-2。

表 21-2 CCU6 引脚功能和选择

引脚	功能	描述	由 ... 选择
捕获输入信号			

**捕获 / 比较单元 6 (CCU6)**
**表 21-2 CCU6 引脚功能和选择**

引脚	功能	描述	由 ... 选择
P1.1	CC60_0	通道 CC60 上的捕获事件输入信号	CCU6_PISEL0L.ISCC60 = 00 <sub>B</sub>
P0.3	CC60_1		CCU6_PISEL0L.ISCC60 = 01 <sub>B</sub>
P1.3	CC61_0	通道 CC61 上的捕获事件输入信号	CCU6_PISEL0L.ISCC61 = 00 <sub>B</sub>
P0.1	CC61_1		CCU6_PISEL0L.ISCC61 = 01 <sub>B</sub>
P1.5	CC62_0	通道 CC62 上的捕获事件输入信号	CCU6_PISEL0L.ISCC62 = 00 <sub>B</sub>
P0.2	CC62_1		CCU6_PISEL0L.ISCC62 = 01 <sub>B</sub>
强制中断输入信号			
P0.3	CTRAP_0	CTRAP 输入信号	MODPISEL3.CTRAPIS = 00 <sub>B</sub> CCU6_PISEL0L.ISTRP = 00 <sub>B</sub>
P0.4	CTRAP_1		MODPISEL3.CTRAPIS = 01 <sub>B</sub> CCU6_PISEL0L.ISTRP = 00 <sub>B</sub>
P2.3	CTRAP_2		MODPISEL3.CTRAPIS = 10 <sub>B</sub> CCU6_PISEL0L.ISTRP = 00 <sub>B</sub>
霍尔输入信号			
P0.0	CCPOS0_0	CCPOS0 输入信号	CCU6_PISEL0H.ISPOS0 = 00 <sub>B</sub>
P2.0	CCPOS0_1		CCU6_PISEL0H.ISPOS0 = 01 <sub>B</sub>
P2.3	CCPOS0_2		CCU6_PISEL0H.ISPOS0 = 10 <sub>B</sub>
P0.1	CCPOS1_0	CCPOS1 输入信号	CCU6_PISEL0H.ISPOS1 = 00 <sub>B</sub>
P2.1	CCPOS1_1		CCU6_PISEL0H.ISPOS1 = 01 <sub>B</sub>
P0.2	CCPOS2_0	CCPOS2 输入信号	CCU6_PISEL0H.ISPOS2 = 00 <sub>B</sub>
P2.2	CCPOS2_1		CCU6_PISEL0H.ISPOS2 = 01 <sub>B</sub>
定时器输入信号			
P0.0	T12HR_0	T12HR 输入信号	MODPISEL3.IST12HR1 = 000 <sub>B</sub> CCU6_PISEL0H.IST12HR = 00 <sub>B</sub>
P2.0	T12HR_2		MODPISEL3.IST12HR1 = 010 <sub>B</sub> CCU6_PISEL0H.IST12HR = 00 <sub>B</sub>
P2.2	T12HR_3		MODPISEL3.IST12HR1 = 011 <sub>B</sub> CCU6_PISEL0H.IST12HR = 00 <sub>B</sub>
P2.4	T12HR_5		MODPISEL3.IST12HR1 = 101 <sub>B</sub> CCU6_PISEL0H.IST12HR = 00 <sub>B</sub>
P2.5	T12HR_7		MODPISEL3.IST12HR1 = 111 <sub>B</sub> CCU6_PISEL0H.IST12HR = 00 <sub>B</sub>

**捕获 / 比较单元 6 (CCU6)**
**表 21-2 CCU6 引脚功能和选择**

引脚	功能	描述	由 ... 选择
P0.1	T13HR_0	T13HR 输入信号	MODPISEL3.IST13HR1 = 000 <sub>B</sub> CCU6_PISEL2.IST13HR = 00 <sub>B</sub>
P0.0	T13HR_1		MODPISEL3.IST13HR1 = 001 <sub>B</sub> CCU6_PISEL2.IST13HR = 00 <sub>B</sub>
P2.0	T13HR_2		MODPISEL3.IST13HR1 = 010 <sub>B</sub> CCU6_PISEL2.IST13HR = 00 <sub>B</sub>
P2.2	T13HR_3		MODPISEL3.IST13HR1 = 011 <sub>B</sub> CCU6_PISEL2.IST13HR = 00 <sub>B</sub>
P2.4	T13HR_5		MODPISEL3.IST13HR1 = 101 <sub>B</sub> CCU6_PISEL2.IST13HR = 00 <sub>B</sub>
P2.5	T13HR_7		MODPISEL3.IST13HR1 = 111 <sub>B</sub> CCU6_PISEL2.IST13HR = 00 <sub>B</sub>

**比较输出信号**

P1.1	CC60_0	通道 CC60 比较输出	P1_ALTSEL0.P1 = 0 <sub>B</sub> P1_ALTSEL1.P1 = 1 <sub>B</sub>
P0.3	CC60_1		P0_ALTSEL0.P3 = 1 <sub>B</sub> P0_ALTSEL1.P3 = 1 <sub>B</sub>
P1.0	COUT60_0		P1_ALTSEL0.P0 = 0 <sub>B</sub> P1_ALTSEL1.P0 = 1 <sub>B</sub>
P1.3	CC61_0	通道 CC61 比较输出	P1_ALTSEL0.P3 = 0 <sub>B</sub> P1_ALTSEL1.P3 = 1 <sub>B</sub>
P0.1	CC61_1		P0_ALTSEL0.P1 = 1 <sub>B</sub> P0_ALTSEL1.P1 = 1 <sub>B</sub>
P1.2	COUT61_0		P1_ALTSEL0.P2 = 0 <sub>B</sub> P1_ALTSEL1.P2 = 1 <sub>B</sub>
P0.0	COUT61_1		P0_ALTSEL0.P0 = 1 <sub>B</sub> P0_ALTSEL1.P0 = 1 <sub>B</sub>

## 捕获 / 比较单元 6 (CCU6)

表 21-2 CCU6 引脚功能和选择

引脚	功能	描述	由 ... 选择
P1.5	CC62_0	通道 CC62 比较输出	P1_ALTSEL0.P5 = 0 <sub>B</sub> P1_ALTSEL1.P5 = 1 <sub>B</sub>
P0.2	CC62_1		P0_ALTSEL0.P2 = 1 <sub>B</sub> P0_ALTSEL1.P2 = 1 <sub>B</sub>
P1.4	COUT62_0		P1_ALTSEL0.P4 = 0 <sub>B</sub> P1_ALTSEL1.P4 = 1 <sub>B</sub>
P0.5	COUT62_1		P0_ALTSEL0.P5 = 1 <sub>B</sub> P0_ALTSEL1.P5 = 1 <sub>B</sub> P0_ALTSEL2.P5 = 0 <sub>B</sub>
P0.7	COUT63_0	通道 CC63 比较输出	P0_ALTSEL0.P7 = 1 <sub>B</sub> P0_ALTSEL1.P7 = 1 <sub>B</sub> P0_ALTSEL2.P7 = 0 <sub>B</sub>
P1.2	COUT63_1		P1_ALTSEL0.P2 = 1 <sub>B</sub> P1_ALTSEL1.P2 = 1 <sub>B</sub>
P1.4	COUT63_2		P1_ALTSEL0.P4 = 1 <sub>B</sub> P1_ALTSEL1.P4 = 1 <sub>B</sub>

## MODISEL3

外设输入选择寄存器 3

(EE<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
IST13HR1			IST12HR1			CTRAPIS	
rw			rw			rw	

符号	位	类型	描述
CTRAPIS	[1:0]	rw	<b>CCU6 CTRAP 输入选择</b> 0 选择 CCU6 CTRAP 输入引脚 CTRAP_0 01 选择 CCU6 CTRAP 输入引脚 CTRAP_1 10 选择 CCU6 CTRAP 输入引脚 CTRAP_2 11 选择通道 3 超出范围事件 (ORC) 输入引脚 CTRAP_3

捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
IST12HR1	[4:2]	rw	<b>CCU6 T12HR 端口引脚输入选择</b> 000 选择 T12HR 输入引脚 T12HR_0 001 选择通道 0 超出范围事件 T12HR_1 010 选择 T12HR 输入引脚 T12HR_2 011 选择 T12HR 输入引脚 T12HR_3 100 选择通道 2 超出范围事件输入 T12HR_4 101 选择 T12HR 输入引脚 T12HR_5 110 选择通道 4 超出范围事件输入 T12HR_6 111 选择 T12HR 输入引脚 T12HR_7 注: 要选择由端口引脚触发 <span style="float:right">T12HR,</span> CCU6_PISEL0H.IST12HR 必须设置为 00 <sub>B</sub> 。
IST13HR1	[7:5]	rw	<b>CCU6 T13HR 端口引脚输入选择</b> 000 选择 T13HR 输入引脚 T13HR_0 001 选择 T13HR 输入引脚 T13HR_1 010 选择 T13HR 输入引脚 T13HR_2 011 选择 T13HR 输入引脚 T13HR_3 100 选择通道 2 超出范围事件输入引脚 T13HR_4 101 选择 T13HR 输入引脚 T13HR_5 110 选择通道 4 超出范围事件输入引脚 T13HR_6 111 选择 T13HR 输入引脚 T13HR_7 注: 要选择由端口引脚触发 <span style="float:right">13HR,</span> CCU6_PISEL2.IST13HR 必须设置为 00 <sub>B</sub> 。

访问 MODPISEL3 寄存器之前, 必须编程设置 SCU\_PAGE 寄存器中的 PAGE 位域。

## 21.2.2 时钟配置

CCU6 内核工作时钟 FPCLK 频率固定为 48 MHz。

如果完全不需要 CCU6 功能, 可通过门控关闭其时钟输入的方式整个的禁止 CCU6 该模块, 从而最大程度的降低功耗。通过下述寄存器 PMCON1 中的 CCU\_DIS 来完成该设置。

访问 PMCON1 寄存器之前, 必须编程设置 SCU\_PAGE 寄存器 PAGE。

### PMCON1

外设管理控制寄存器 1

(EF<sub>H</sub>)

复位值: FF<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	CDC_DIS	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	rw	rw	rw	rw	rw	rw

## 捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
CCU_DIS	2	rw	<b>CCU6 禁止请求，高有效</b> 0 CCU6 正常工作 1 请求禁止 CCU6 (缺省设置)



捕获 / 比较单元 6 (CCU6)

21.2.3 中断事件和分配

表 21-3 列出 CCU6 的中断源以及对应的中断事件使能位和标志位。

表 21-3 CCU6 中断事件

事件	事件中断使能位	事件标志位	服务请求输出
见 章节 21.9	见 IENL, IENH	见 ISL, ISH	SR0, SR1, SR2, SR3

表 21-4 给出 CCU6 每个中断源的中断节点分配情况。

表 21-4 CCU6 事件中断节点控制

事件	中断节点使能位	中断节点标志位	向量地址
SR0	IEN1.ECCIP0	IRCON2.CCU6SR0	53 <sub>H</sub>
SR1	IEN1.ECCIP1	IRCON2.CCU6SR1	5B <sub>H</sub>
SR2	IEN1.ECCIP2 MODIEN.CCU6SR2EN	IRCON3.CCU6SR2	63 <sub>H</sub>
SR3	IEN1.ECCIP3 MODIEN.CCU6SR3EN	IRCON3.CCU6SR3	6B <sub>H</sub>

除了表 21-3 中的中断使能位，位 MODIEN.CCU6SR2EN 和 MODIEN.CCU6SR3EN 必须设置为 1<sub>B</sub>，以使能 SR2 和 SR3 中断服务请求。访问 MODIEN 寄存器之前，必须编程设置寄存器 SCU\_PAGE 中的 PAGE 位。

注： CCU6 模块的 SR2 和 SR3 事件用作模块间互连的输出信号，可用来触发 CC6x 输入： T12HR 输入和 T13HR 输入。如果该类型触发模式不需要中断服务，可禁止 MODIEN 中的控制位。然而，SR2 和 SR3 事件要用作互连输出信号，必须使能事件中断使能位 IENL 和 IENH。

MODIEN

外设中断使能寄存器

(F7<sub>H</sub>)

复位值： 07<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
CCU6SR3 EN	CCU6SR2 EN		0		RIREN	TIREN	EIREN
rw	rw		r		rw	rw	rw

**捕获 / 比较单元 6 (CCU6)**

符号	位	类型	描述
<b>CCU6SR2EN</b>	6	rw	<b>CCU6 SR2 使能</b> 0 禁止 CCU6 SR2 中断服务请求 1 使能 CCU6 SR2 中断服务请求
<b>CCU6SR3EN</b>	7	rw	<b>CCU6 SR3 使能</b> 0 禁止 CCU6 SR3 中断服务请求 1 使能 CCU6 SR3 中断服务请求
<b>0</b>	[5:3]	r	<b>保留；</b> 读操作返回 0；应写入 0。

## 捕获 / 比较单元 6 (CCU6)

### 21.2.4 IP 互连

CCU6 可和其它外设相互连接，从而无需软件干预即可实现高水平的自动化。

**表 21-5 CCU6 输出连接关系**

CCU6 功能 / 信号	连接至其它模块功能 / 信号
<b>定时器 T12 输出信号</b>	
T12 周期匹配 (o): T12PM	ADC 请求源 0/1 触发 2 输入 (i): REQTR0C, REQTR1C
<b>定时器 T13 输出信号</b>	
T13 比较匹配 (o): T13CM	ADC 请求源 0/1 触发 4 输入 (i): REQTR0E, REQTR1E
T13 周期匹配 (o): T13PM	ADC 请求源 0/1 触发 3 输入 (i): REQTR0D, REQTR1D
<b>CCU6 服务请求输出信号</b>	
服务请求输出 SR2 (o): CCU6_SR2	ADC 请求源 0/1 触发 0 输入 (i): REQTR0A, REQTR1A
服务请求输出 SR3 (o): CCU6_SR3	ADC 请求源 0/1 触发 1 输入 (i): REQTR0B, REQTR1B
<b>其它信号</b>	
MCM 映射传送 (o): MCM_ST	ADC 请求源 0/1 触发 5 输入 (i): REQTR0F, REQTR1F

**表 21-6 CCU6 输入连接关系**

CCU6 功能 / 信号	连接至其它模块功能 / 信号	由 ... 选择
<b>捕获输入</b>		
CCU6 输入 (i): CC60	CCU6 SR2 输出 (o): CCU6_SR2	CCU6_PISEL0L.ISCC60 = 10 <sub>B</sub>
	ADC 边界事件 0 (o): ADC_BF0	CCU6_PISEL0L.ISCC60 = 11 <sub>B</sub>
CCU6 输入 (i): CC61	CCU6 SR2 输出 (o): CCU6_SR2	CCU6_PISEL0L.ISCC61 = 10 <sub>B</sub>
	ADC 边界事件 1 (o): ADC_BF1	CCU6_PISEL0L.ISCC61 = 11 <sub>B</sub>

**捕获 / 比较单元 6 (CCU6)**
**表 21-6 CCU6 输入连接关系**

CCU6 功能 / 信号	连接至其它模块功能 / 信号	由 ... 选择
CCU6 输入 (i): CC62	CCU6 SR2 输出 (o): CCU6_SR2	CCU6_PISEL0L.ISCC62 = 10 <sub>B</sub>
	ADC 边界事件 2 (o): ADC_BF2	CCU6_PISEL0L.ISCC62 = 11 <sub>B</sub>
CCU6 输入 (i): T12HR	CCU6 SR2 输出 (o): CCU6_SR2	CCU6_PISEL0H.IST12HR = 01 <sub>B</sub>
	CCU6 SR3 输出 (o): CCU6_SR3	CCU6_PISEL0H.IST12HR = 10 <sub>B</sub>
	ADC 通道事件 (o): ADC_CHEV	CCU6_PISEL0H.IST12HR = 11 <sub>B</sub>
	ORC 事件 0 (o): ORCEVENT0	MODPISEL3.IST12HR1 = 001 <sub>B</sub> CCU6_PISEL0H.IST12HR = 00 <sub>B</sub>
	ORC 事件 2 (o): ORCEVENT2	MODPISEL3.IST12HR1 = 100 <sub>B</sub> CCU6_PISEL0H.IST12HR = 00 <sub>B</sub>
	ORC 事件 4(o): ORCEVENT4	MODPISEL3.IST12HR1 = 110 <sub>B</sub> CCU6_PISEL0H.IST12HR = 00 <sub>B</sub>
CCU6 输入 (i): T13HR	CCU6 SR2 输出 t (o): CCU6_SR2	CCU6_PISEL2.IST13HR = 01 <sub>B</sub>
	CCU6 SR3 输出 (o): CCU6_SR3	CCU6_PISEL2.IST13HR = 10 <sub>B</sub>
	ADC 通道事件 (o): ADC_CHEV	CCU6_PISEL2.IST13HR = 11 <sub>B</sub>
	ORC 事件 2 (o): ORCEVENT2	MODPISEL3.IST13HR1 = 100 <sub>B</sub> CCU6_PISEL2.IST13HR = 00 <sub>B</sub>
	ORC 事件 4(o): ORCEVENT4	MODPISEL3.IST13HR1 = 110 <sub>B</sub> CCU6_PISEL2.IST13HR = 00 <sub>B</sub>
CCU6 输入 (i): CTRAP	ORC 事件 3 (o): ORCEVENT3	MODPISEL3.CTRAPIS = 11 <sub>B</sub> CCU6_PISEL0L.ISTRP = 00 <sub>B</sub>
	P1 过流事件 (o): P1_OCD	CCU6_PISEL0L.ISTRP = 01 <sub>B</sub> /10 <sub>B</sub>
	ADC 通道事件 0 (o): ADC_CHEV0	CCU6_PISEL0L.ISTRP = 11 <sub>B</sub>
CCU6 输入 (i): CCPOS0	ADC 边界事件 0 (o): ADC_BF0	CCU6_PISEL0H.ISPOS0 = 11 <sub>B</sub>

表 21-6 CCU6 输入连接关系

CCU6 功能 / 信号	连接至其它模块功能 / 信号	由 ... 选择
CCU6 输入 (i): CCPOS1	ADC 边界事件 1 (o): ADC_BF1	CCU6_PISEL0H.ISPOS1 = 11 <sub>B</sub>
CCU6 输入 (i): CCPOS2	ADC 边界事件 2 (o): ADC_BF2	CCU6_PISEL0H.ISPOS2 = 11 <sub>B</sub>

### 21.2.5 模块挂起控制

当 OCDS 处于监控模式 (MMCR2.MMODE = 1)，且调试挂起信号有效 (MMCR2.DSUSP = 1) 时，根据寄存器 MODSUSP 中的相关模块挂起位的设置，XC83x 的 CCU6 模块中的定时器 T12 和定时器 T13 可被挂起。该寄存器的定义见[章节 10.2.4](#)。被挂起时，由于计数器的输入时钟关闭，定时器停止计数。模块仍有时钟输入，因此可访问模块寄存器。

## 21.3 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 个通道寄存器相连，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磁滞比较模式。

本节提供下述内容：

- T12 模块概述 (见[章节 21.3.1](#))
- 计数策略 (见[章节 21.3.2](#))
- 比较模式 (见[章节 21.3.3](#))
- 比较模式输出通路 (见[章节 21.3.4](#))
- 捕获模式 (见[章节 21.3.5](#))
- 映射传送 (见[章节 21.3.6](#))
- T12 工作模式选择 (见[章节 21.3.7](#))
- T12 计数寄存器描述 (见[章节 21.3.8](#))

## 捕获 / 比较单元 6 (CCU6)

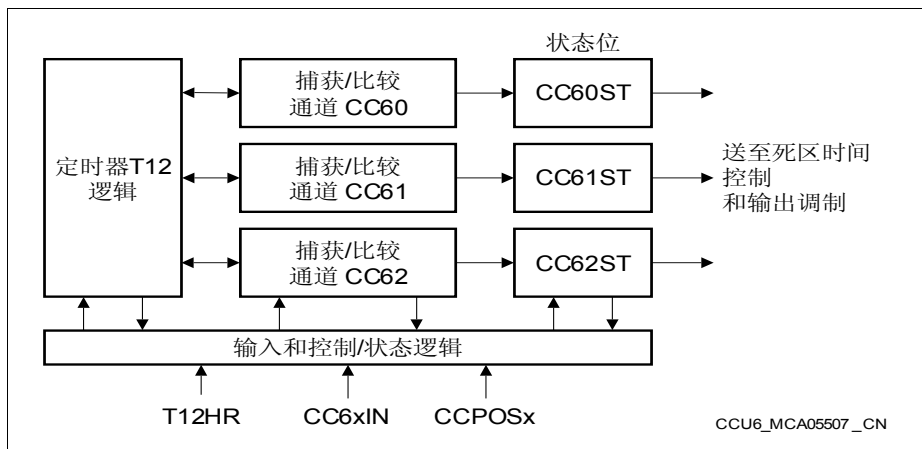


图 21-3 定时器 T12 模块总览

### 21.3.1 T12 概述

图 21-4 为定时器 T12 的内部结构框图。由寄存器 **TCTR0L**, **TCTR0H**, **TCTR2L**, **TCTR2H**, **TCTR4L**, **TCTR4H**, **PISEL0L** 和 **PISEL0H** 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) 和一个可选的 1/256 分频器, 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 21-7)。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。

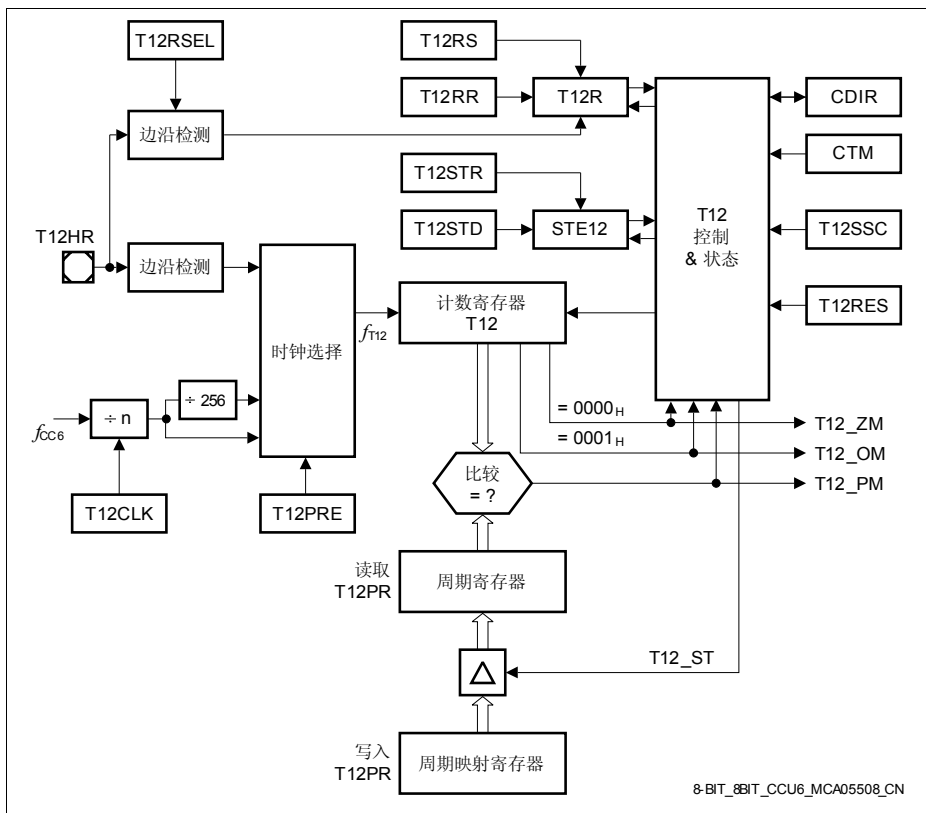


图 21-4 定时器 T12 逻辑和周期比较器

T12 计数寄存器 **T12L**, **T12H** 通过比较器和周期寄存器 **T12PRL**, **T12PRH** 相连, 该寄存器的内容决定 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\_PM

## 捕获 / 比较单元 6 (CCU6)

(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 得到比较器的当前周期值，对映射周期寄存器进行写访问则写入新周期值。“T12 映射传送”控制信号 T12\_ST 控制将新周期值从映射寄存器传送到 T12PR 中的操作（见**章节 21.3.6**），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 21-5**）或中心对齐模式（见**图 21-6**），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 21-7**和**图 21-8**）。

运行位 T12R 控制 T12 定时器的启动和停止，可由寄存器 **TCTR4L**，**TCTR4H** 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（**TCTR2H.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。



### 21.3.2 T12 计数策略

本节描述 T12 的时钟和计数功能。

#### 21.3.2.1 时钟选择

定时器 T12 的输入时钟  $f_{T12}$  来自经过预分频处理（预分频因子可编程设定）的模块时钟  $f_{CC6}$  和一个可选的 1/256 分频器。所得预分频因子列于表 21-7 中。定时器 T12 不工作时（**TCTR0L.T12R** = 0），预分频器复位，从而保证产生可重复的时序和延迟。

表 21-7 定时器 T12 输入频率选择

T12CLK	所得输入时钟 $f_{T12}$ 预分频器关闭 (T12PRE = 0)	所得输入时钟 $f_{T12}$ 预分频器开启 (T12PRE = 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>	$f_{CC6} / 128$	$f_{CC6} / 32768$

#### 21.3.2.2 边沿对齐 / 中心对齐模式

在**边沿对齐模式**下（CTM = 0），定时器 T12 始终递增计数（CDIR = 0）。计数至由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）。

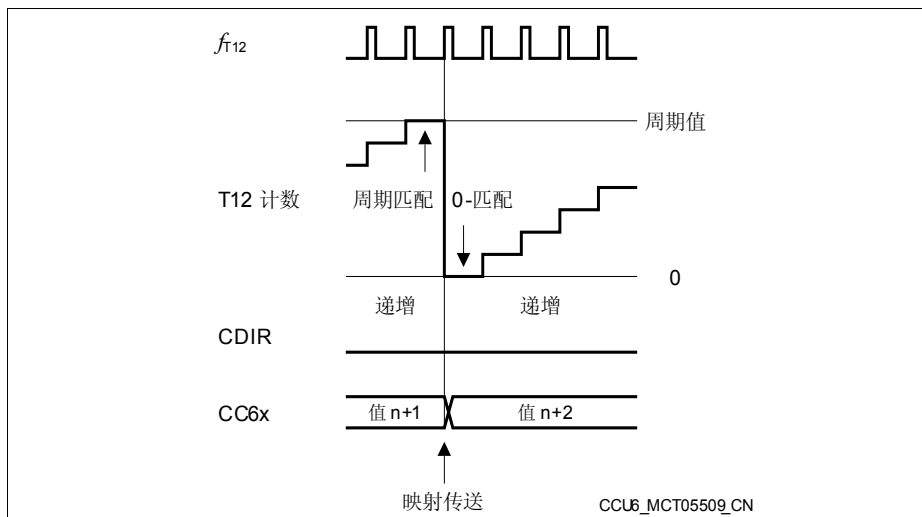


图 21-5 边沿对齐模式下 T12 的操作

因此边沿对齐模式下，定时器的周期值由下式给出：

$$T12_{PER} = \langle \text{周期值} \rangle + 1; \text{以 } T12 \text{ 时钟频率 } (f_{T12}) \text{ 计数} \quad (21.1)$$

在**中心对齐模式**下 (CTM = 1)，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)，下一个计数脉冲来到时，计数方向控制位变为递增计数。

因此中心对齐模式下，定时器的周期由下式给出：

$$T12_{PER} = (\langle \text{周期值} \rangle + 1) \times 2; \text{以 } T12 \text{ 时钟频率 } (f_{T12}) \text{ 计数} \quad (21.2)$$

- 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。

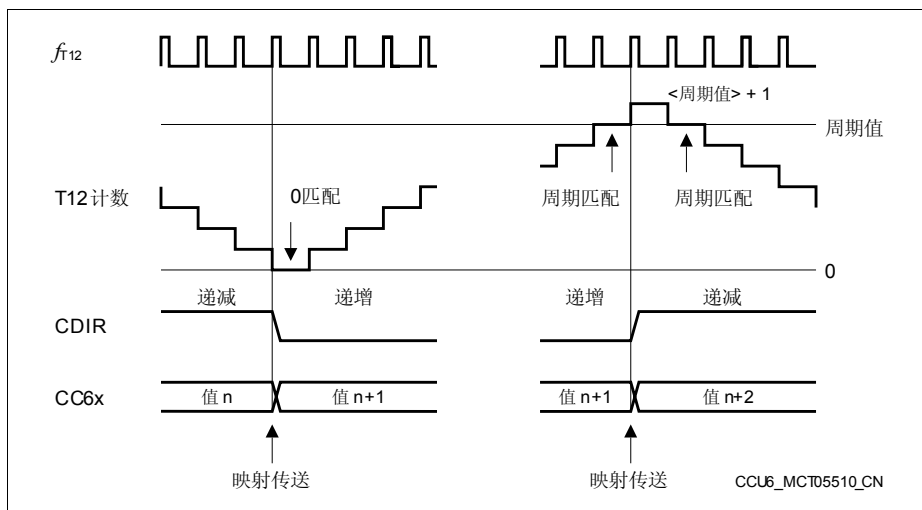


图 21-6 中心对齐模式下 T12 的操作

注：在 1- 匹配或周期匹配后的下一个定时器时钟，位 CDIR 改变。因此，定时器以原先的计数方向再继续计数一个时钟后才真正改变计数方向（见图 21-6）。

## 捕获 / 比较单元 6 (CCU6)

### 21.3.2.3 单次模式

单次模式下，定时器运行位 **T12R** 被硬件清除。如果 **T12SSC = 1**，当前定时器周期结束之后定时器 **T12** 停止运行。

边沿对齐模式下，当定时器 **T12** 计数到周期值后被清零至  $0000_H$ ，**T12R** 被清除，（见图 21-7）。

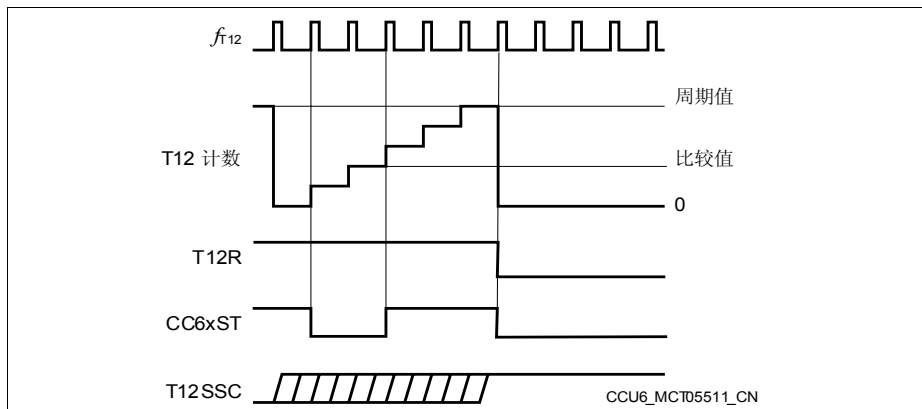


图 21-7 边沿对齐模式下的单次操作

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 21-8）。

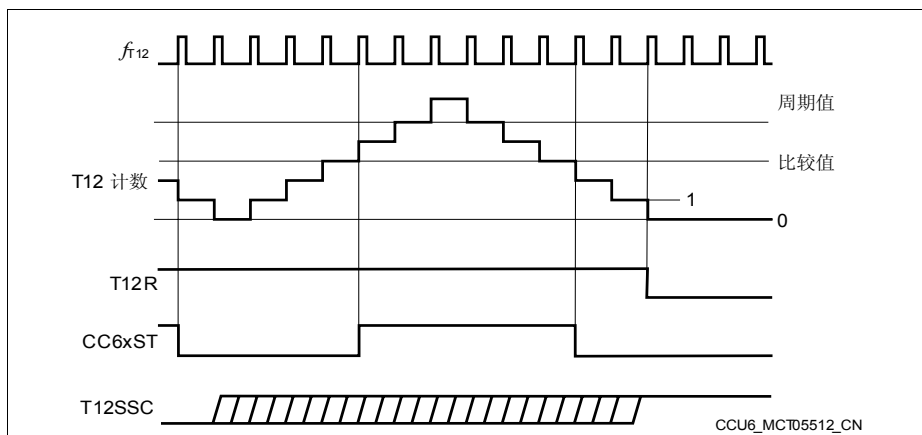


图 21-8 中心对齐模式下的单次操作

## 捕获 / 比较单元 6 (CCU6)

### 21.3.3 T12 的比较模式

定时器 T12 的三个独立的捕获 / 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 21.3.5](#)。

#### 21.3.3.1 比较通道

比较模式下（见[图 21-9](#)），三个比较通道 CC60, CC61 和 CC62 可产生三相 PWM 序列。

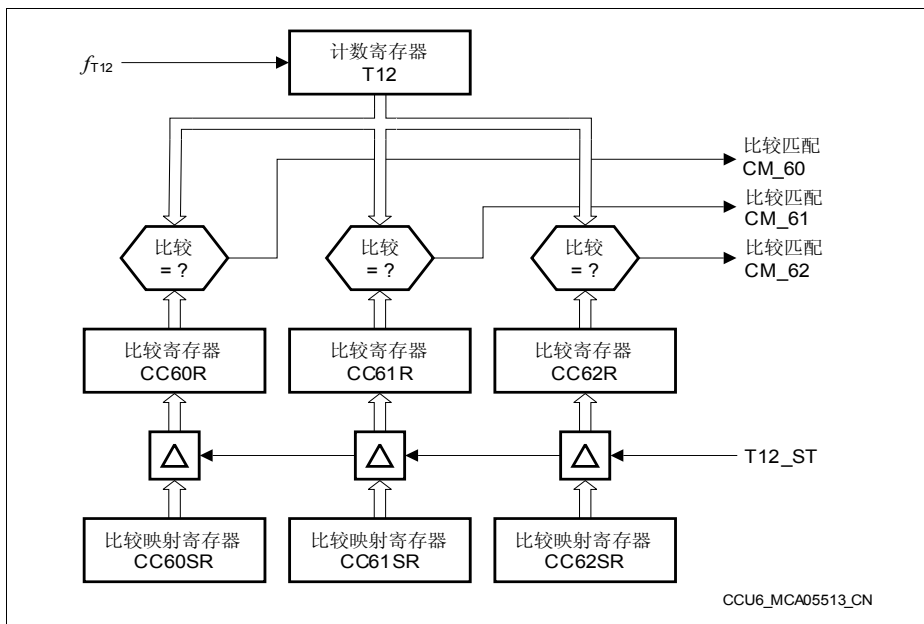


图 21-9 T12 通道比较器

每个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。每个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。

#### 21.3.3.2 通道状态位

每个（比较）通道都有对应的状态位，CMPSTATL.CC6xST，用来保存比较（或捕获）操作的状态（见[图 21-10](#)）。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。

## 捕获 / 比较单元 6 (CCU6)

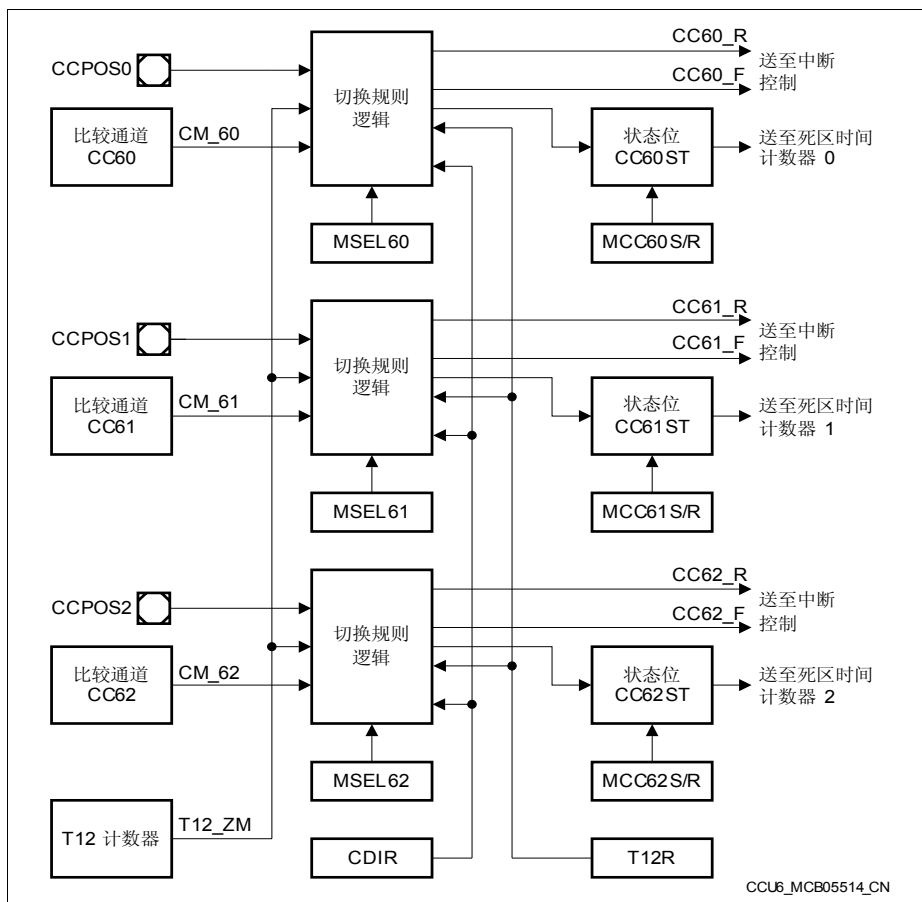


图 21-10 比较模式下的比较状态位框图

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 **T12MSELL**.MSEL6x。

此外，可通过软件设定寄存器 **CMPMODIFL**，**CMPMODIFH** 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)。

## 捕获 / 比较单元 6 (CCU6)

递增计数检测到比较匹配时，则指示发生比较中断事件 **CC6x\_R**；递减计数检测到比较中断事件时，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。

只有当定时器 **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 21-11 和图 21-12 所示）：

状态位 **CC6xST** 被置位到 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** 被复位到 0：

- **T12** 递减计数到比较匹配后，下一个 **T12** 时钟 ( $f_{T12}$ ) 复位 **CC6xST**（即，中心对齐模式下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**。

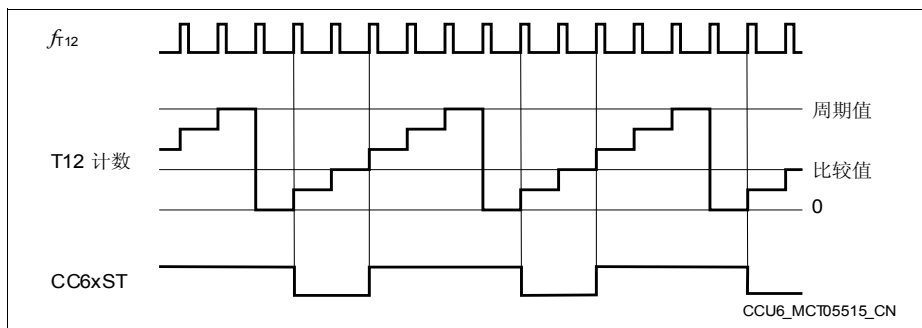


图 21-11 边沿对齐模式下的比较操作

图 21-13 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。

## 捕获 / 比较单元 6 (CCU6)

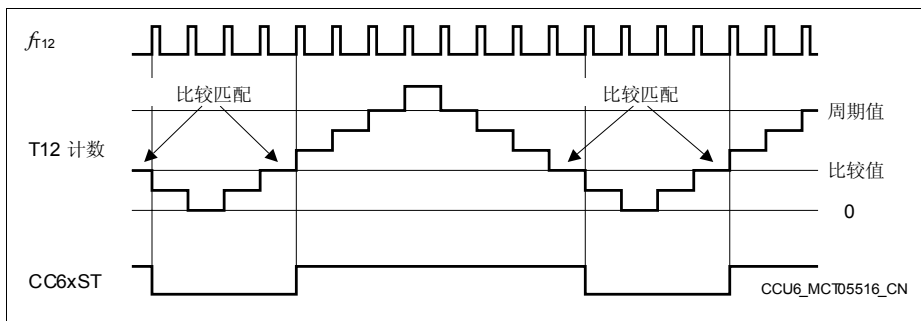


图 21-12 中心对齐模式下的比较操作

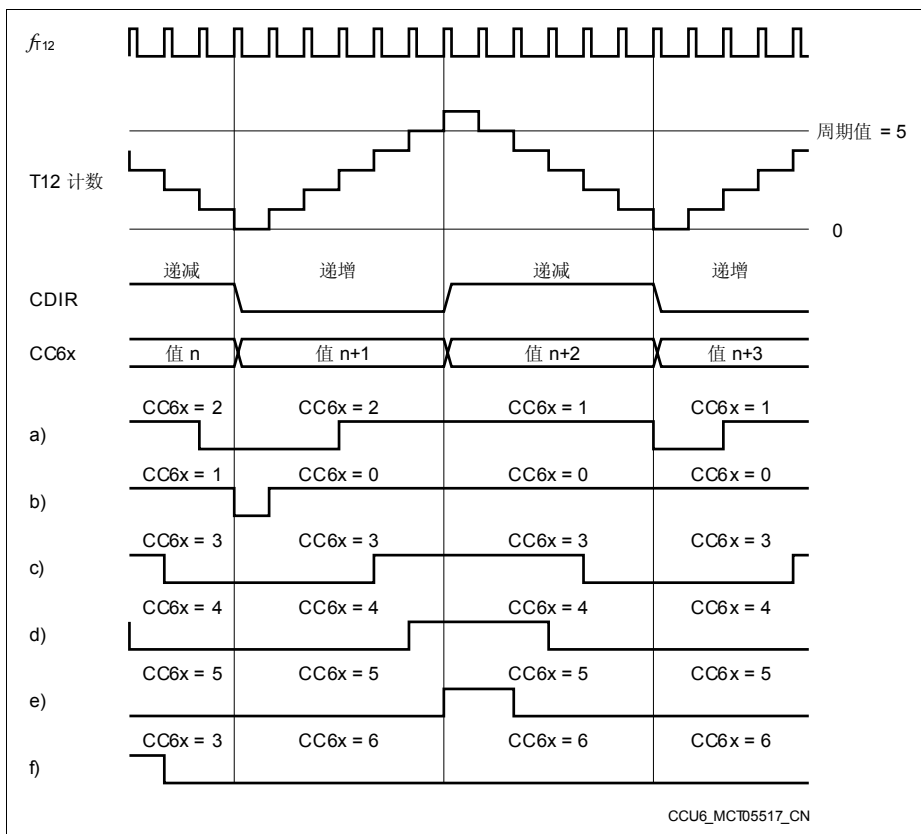


图 21-13 比较波形举例

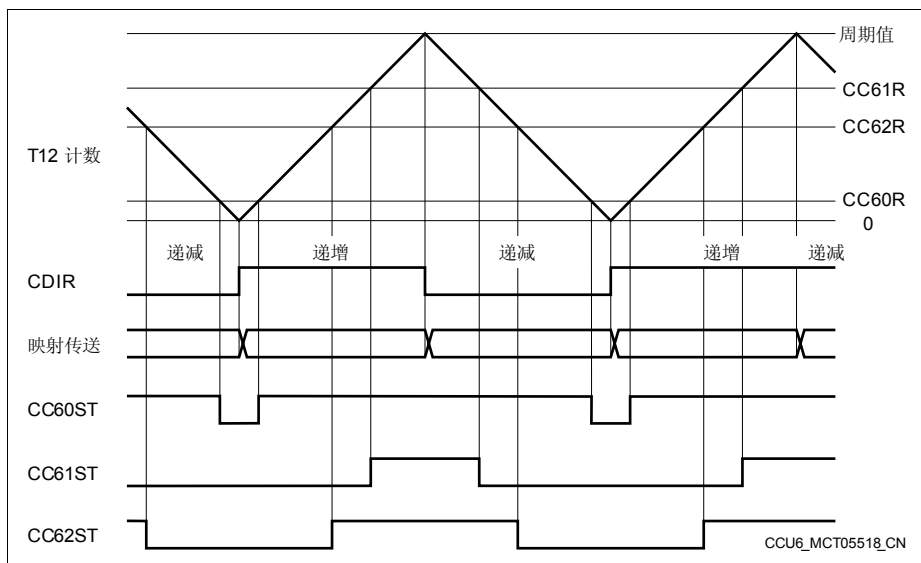


## 捕获 / 比较单元 6 (CCU6)

例 b) 图示说明 跳变到占空比为 100% 的情况。先使用比较值  $0001_H$ ，然后改为  $0000_H$ 。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  $0001_H$  产生）。接下来的定时器周期，状态位 CC6xST 始终保持在 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  $< \text{周期值} > + 1$ ，状态位 CC6ST 始终保持为 0。

**图 21-14** 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 21-14** 三通道比较波形

### 21.3.3.3 类磁滞控制模式

类磁滞控制模式 (**T12MSELL.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 21-10](#) 状态位逻辑和 [图 21-15](#) 输出路径)。

使用  $f_{CC6}$  来评估 **CCPOSx** 输入。

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, 可按照某个固定时间基准接通输出; 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, 才切断输出以避免损坏系统。

### 21.3.4 比较模式输出路径

图 21-15 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 21.3.4.3。

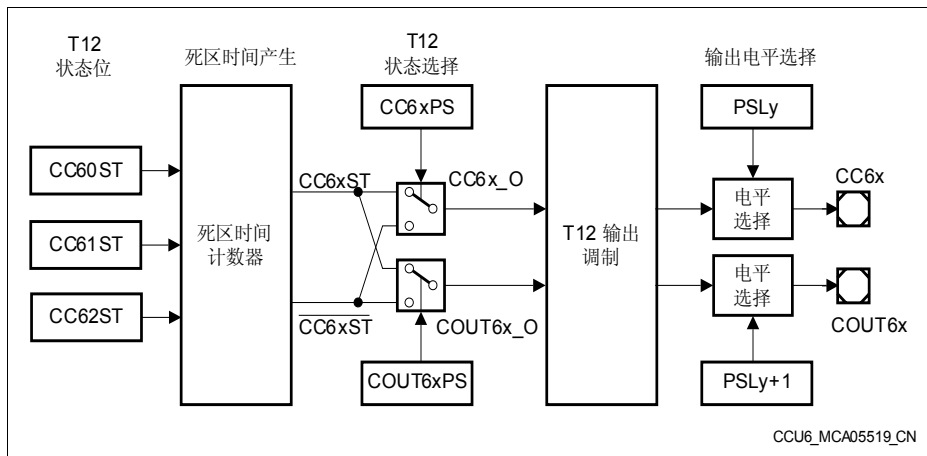


图 21-15 比较模式简化输出路径框图

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，外部功率开关关闭。

#### 21.3.4.1 死区时间产生

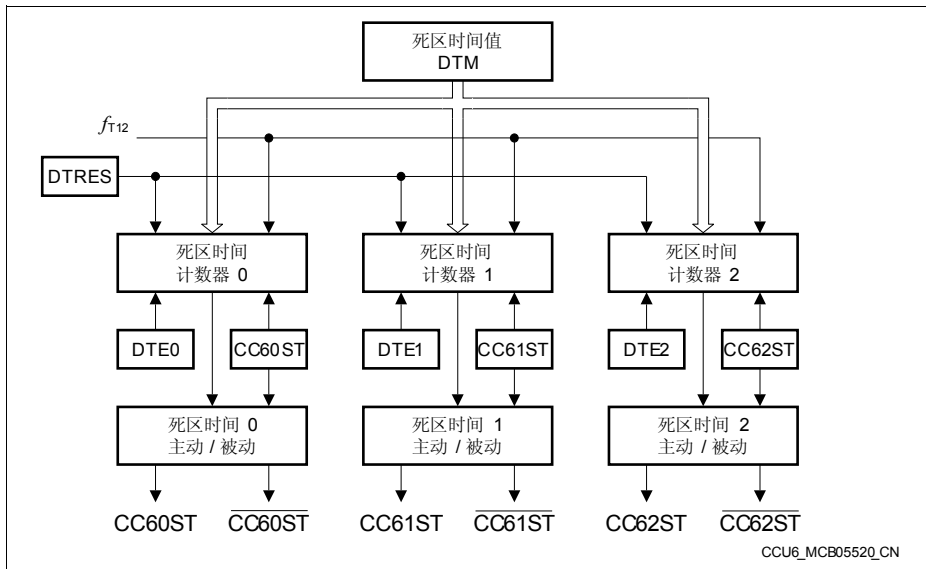
功率逆变器中某一相的上下桥臂开关信号（信号互补）由同一个比较通道产生。例如，若  $T12$  计数值大于比较值时，上桥臂的控制开关导通（状态位 = 1），那么，当  $T12$  计数值小于比较值时，下桥臂的控制开关导通（状态位 = 0）。

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）。

$T12$  三个通道的死区时间产生模块的结构相似，如图 21-16 所示。由寄存器  $T12DTCL$ 、 $T12DTCH$  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

## 捕获 / 比较单元 6 (CCU6)

定。对于 T12 三个通道来说，死区时间的长度一致。写 **TCTR4L.DTRES = 1**，则将所有死区时间设置为被动。



**图 21-16 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 **CC6<sub>x</sub>ST** 的改变无效。这就避免了状态位 **CC6<sub>x</sub>ST** 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 **CC6<sub>x</sub>ST** 边沿。

根据状态位 **CC6<sub>x</sub>ST**，死区时间产生模块为每个比较通道输出一个直接信号 **CC6<sub>x</sub>ST** 和一个反向信号 **CC6<sub>x</sub>ST**，由相关的死区计数器的结果屏蔽每个信号（波形见图 21-17）。



### 21.3.4.3 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 **MODCTRL**，**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 21-18），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** 的反向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 21-18 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。

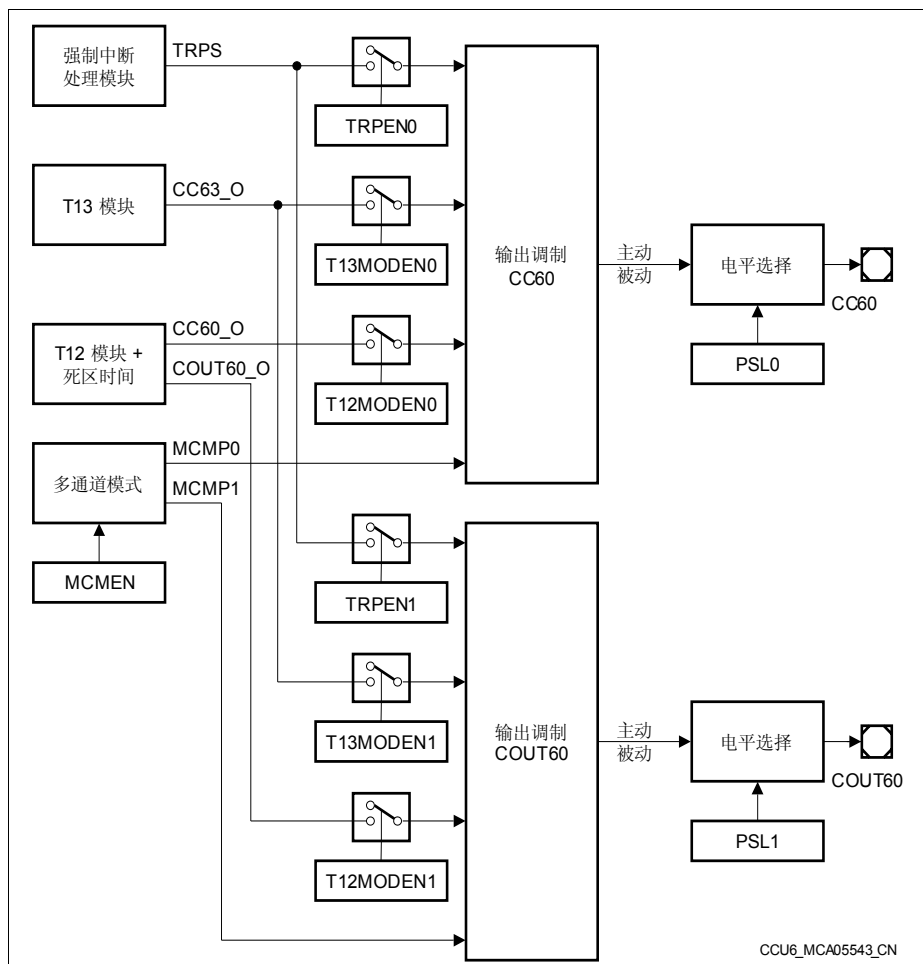


图 21-18 比较通道 CC60 的输出调制

## 捕获 / 比较单元 6 (CCU6)

### 21.3.5 T12 捕获模式

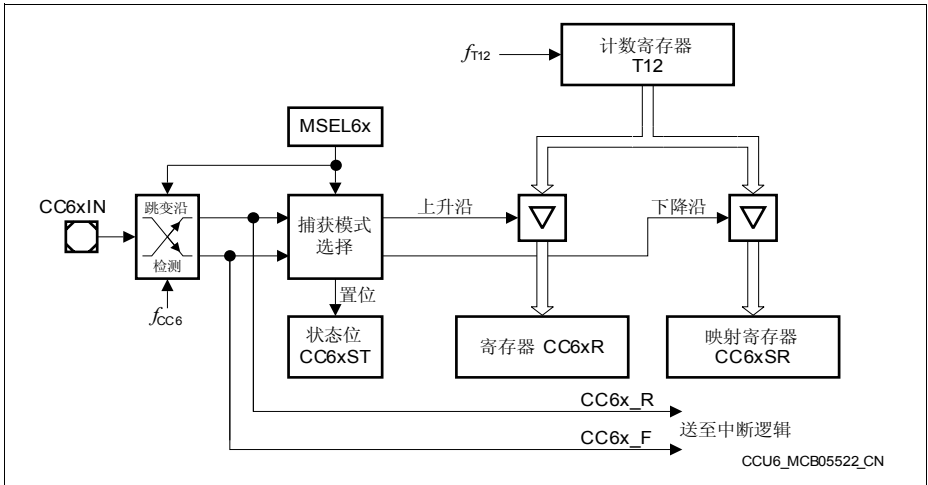
T12 模块的三个通道均可响应外部信号 CC6xIN，用于捕获 T12 的时间信息。  
捕获模式下，当在输入 CC6xIN 上检测到上升沿时，则发生中断事件 CC6x\_R，在输入 CC6xIN 上检测到下降沿时，则发生中断事件 CC6x\_F。

CCU6 单元提供了多种不同的捕获模式。所有捕获模式下，均使用各通道的两个寄存器（实际寄存器以及映射寄存器）。通过 **T12MSELL.MSEL6x** 位域分别为各通道选择捕获模式。

**表 21-8 捕获模式总览**

MSEL6x	模式	信号	有效边沿	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 21-19 说明捕获模式 1 的操作。**检测到输入信号 CC6xIN 的上升沿（0-1 的跳变）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（1-0 的跳变）时，将定时器 T12 的内容捕获到 CC6xSR 中。



**图 21-19 捕获模式 1 框图**

捕获模式 2、3 和 4 如图 21-20 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



## 捕获 / 比较单元 6 (CCU6)

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 21-8。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。

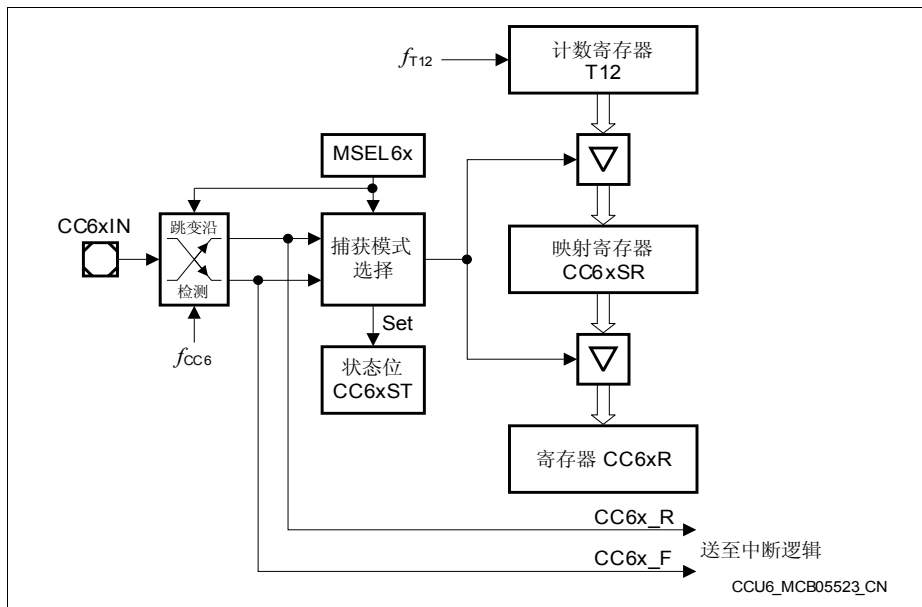


图 21-20 捕获模式 2, 3 和 4 框图

## 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 和信号 CCPOSx 触发捕获。

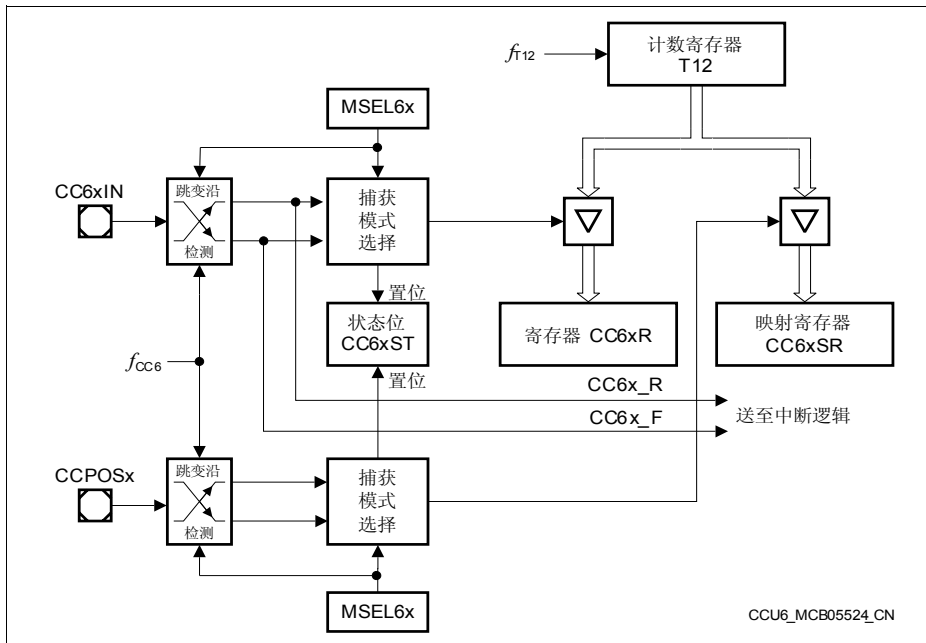


图 21-21 多输入捕获模式框图

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 21-9。

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 21.9）。

表 21-9 多输入捕获模式总览

MSEL6x	模式	信号	有效边沿	T12 保存到
1010 <sub>B</sub>	5	CC6xIN	上升沿	CC6xR
		CCPOSx	下降沿	CC6xSR

捕获 / 比较单元 6 (CCU6)

表 21-9 多输入捕获模式总览

MSEL6x	模式	信号	有效边沿	T12 保存到
1011 <sub>B</sub>	6	CC6xIN	下降沿	CC6xR
		CCPOSx	上升沿	CC6xSR
1100 <sub>B</sub>	7	CC6xIN	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	任意边沿	CC6xR
		CCPOSx	任意边沿	CC6xSR
1111 <sub>B</sub>	—	保留 (无捕获或比较操作)		

### 21.3.6 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 **TCTR0L.STE12** (向只写位 **TCTR4L.T12STR** 写入 1 将置位 STE12 ; 向只写位 **TCTR4L.T12STD** 写入 1 将清除 STE12)。

图 21-22 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。

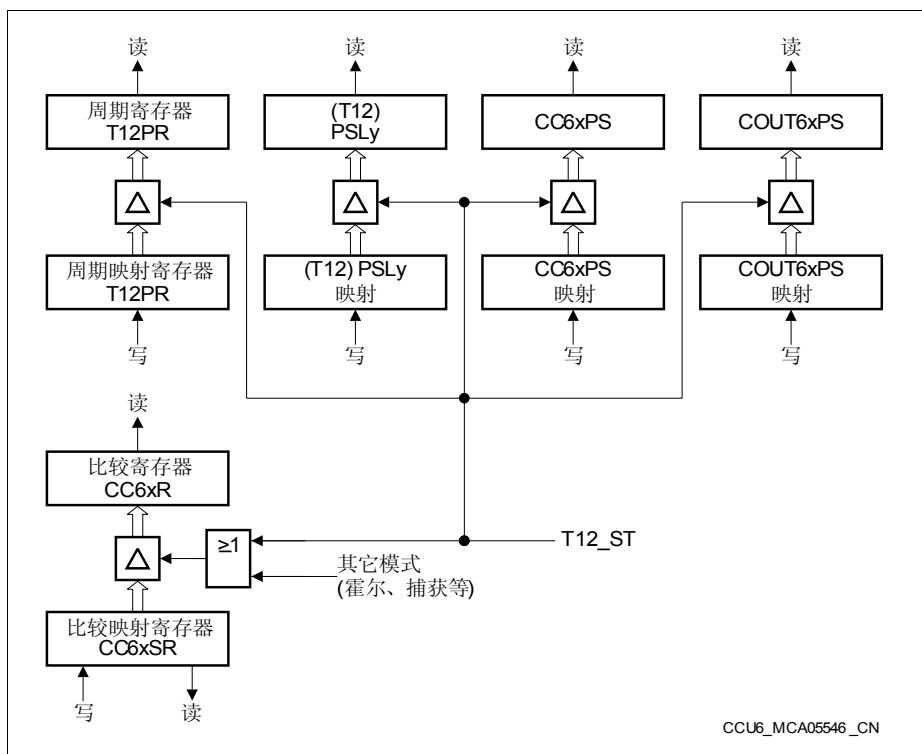


图 21-22 T12 映射寄存器概览

## 捕获 / 比较单元 6 (CCU6)

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
- STE12 = 1 且递增计数时检测到周期匹配，或
- STE12 = 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。

### 21.3.7 定时器 T12 工作模式选择

由位域 **TCTR4L.MSEL6x** 和 **T12MSELL.MSEL6x** 定义 T12 各通道的工作模式。

**表 21-10 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>	捕获 / 比较模式关闭
0001 <sub>B</sub> , 0010 <sub>B</sub> , 0011 <sub>B</sub>	比较模式，见 <a href="#">章节 21.3.3</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 21.3.5</a>
1000 <sub>B</sub>	霍尔传感器模式，见 <a href="#">章节 21.7</a> 为确保能够正确使用该模式，三个位域 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 21.3.3.3</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 21.3.5</a>

由定时器控制寄存器 **TCTR0L**, **TCTR0H** 和 **TCTR2L**, **TCTR2H** 控制定时器的时钟和计数策略。通过写寄存器 **TCTR4L**, **TCTR4H** 可触发特定操作。

## 21.3.8 T12 相关寄存器

### 21.3.8.1 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。

#### T12L

定时器 T12 计数寄存器低位 (FA<sub>H</sub>) 复位值: 00<sub>H</sub>  
RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
T12CVL							
rwh							

符号	位	类型	描述
T12CVL	[7:0]	rwh	定时器 T12 计数值 该寄存器表示 T12 的 16 位计数值的低 8 位。

#### T12H

定时器 T12 计数寄存器高位 (FB<sub>H</sub>) 复位值: 00<sub>H</sub>  
RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
T12CVH							
rwh							

符号	位	类型	描述
T12CVH	[7:0]	rwh	定时器 T12 计数值 该寄存器表示 T12 的 16 位计数值的高 8 位。

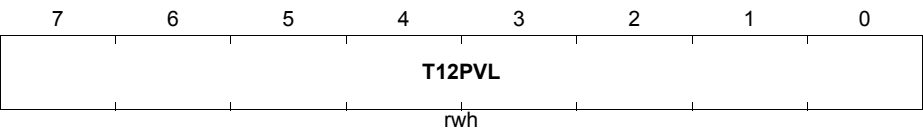
### 21.3.8.2 周期寄存器

寄存器 T12PRL/H 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

捕获 / 比较单元 6 (CCU6)

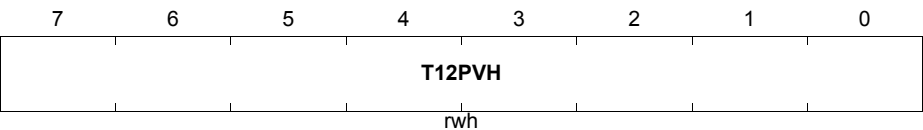
传送。软件读操作读出当前用于周期比较的周期值；而写操作将周期值写入映射寄存器中。映射寄存器结构便于用户同步更新 T12 的所有相关值。

**T12PRL**  
**定时器 T12 周期寄存器低位** (9C<sub>H</sub>) **复位值: 00<sub>H</sub>**  
**RMAP: 0, PAGE: 1**



符号	位	类型	描述
T12PVL	[7:0]	rwh	<b>定时器 T12 周期值</b> T12PVL 定义导致周期匹配的周期值的低 8 位。计数到该值时，定时器被置为 0（边沿对齐模式）或者将计数方向改为递减（中心对齐模式）。

**T12PRH**  
**定时器 T12 周期寄存器高位** (9D<sub>H</sub>) **复位值: 00<sub>H</sub>**  
**RMAP: 0, PAGE: 1**



符号	位	类型	描述
T12PVH	[7:0]	rwh	<b>定时器 T12 周期值</b> T12PVH 定义导致周期匹配的周期值的高 8 位。计数到该值时，定时器 T12 置为 0（边沿对齐模式）或将计数方向改为递减（中心对齐模式）。

捕获 / 比较单元 6 (CCU6)

21.3.8.3 捕获 / 比较寄存器

比较模式下，寄存器 CC6xRL/H ( $x = 0 - 2$ ) 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（三个通道同时进行）。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 CC6xRL/H 中。

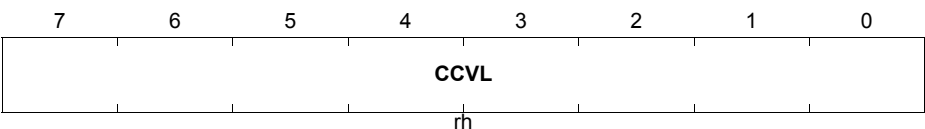
CC6xRL ( $x = 0-2$ )

通道 CC6x 捕获 / 比较寄存低位

$$(FA_H + x * 2)$$

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 1



符号	位	类型	描述
CCVL	[7:0]	rh	捕获 / 比较值 比较模式下，位域 CCV 中存放着和 T12 计数值进行比较的值；捕获模式下，T12 的捕获值可从这些寄存器读出。

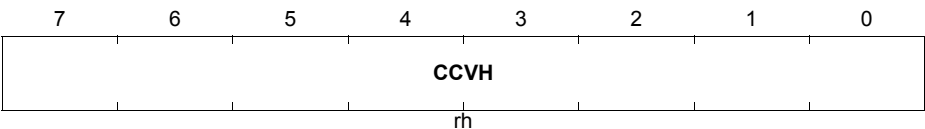
CC6xRH ( $x = 0-2$ )

通道 CC6x 捕获 / 比较寄存器高位

$$(FB_H + x * 2)$$

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 1



符号	位	类型	描述
CCVH	[7:0]	rh	捕获 / 比较值 比较模式下，位域 CCV 中存放着和 T12 计数值进行比较的值；捕获模式下，T12 捕获值可从这些寄存器读出。



捕获 / 比较单元 6 (CCU6)

21.3.8.4 捕获 / 比较映射寄存器

寄存器 CC6xRL/H 可由软件读取，若要修改该寄存器的值，则需要将映射寄存器 CC6xSRL/H 中的修改值映射传送到 CC6xRL/H 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 CC6xSRL/H 中。

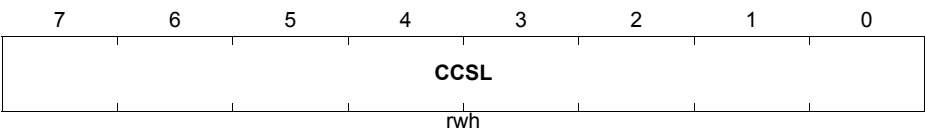
CC6xSRL (x=0-2)

CC6x 捕获 / 比较映射寄存器低位

$$(FA_H + x * 2)$$

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 0



符号	位	类型	描述
CCSL	[7:0]	rh	通道 x 捕获 / 比较值映射寄存器 比较模式下，映射传送期间，位域 CCS 的内容被传送到对应通道的位域 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）。

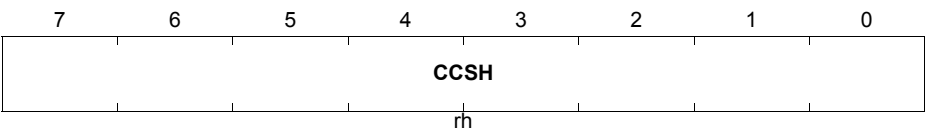
CC6xSRH (x=0-2)

通道 CC6x 捕获 / 比较寄存器高位

$$(FB_H + x * 2)$$

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 0



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）。

捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
<b>CCSH</b>	[7:0]	rh	<b>通道 x 捕获 / 比较值的映射寄存器</b> 比较模式下，映射传送期间，位域 CCS 的内容被传送到对应通道的位域 CCV 中；捕获模式下，T12 的捕获值可从这些寄存器读出。

捕获 / 比较单元 6 (CCU6)

### 21.3.8.5 死区时间控制寄存器

寄存器 T12DTCL/H 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。

#### T12DTCL

定时器死区时间控制寄存器低位

(A4<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
DTM							
rw							

符号	位	类型	描述
DTM	[7:0]	rw	死区时间值 位域 DTM 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。

#### T12DTCH

定时器 T12 死区时间控制寄存器

(A5<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	DTR2	DTR1	DTR0	0	DTE2	DTE1	DTE0
r	rh	rh	rh	r	rw	rw	rw

## 捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
<b>DTE2, DTE1, DTE0</b>	2, 1, 0	rw	<b>死区时间产生使能位</b> 位 DTE0..DTE2 使能和禁止定时器 T12 各比较通道 (0, 1, 2) 的死区时间产生。 0 <sub>B</sub> 死区时间计数器 x 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）。 1 <sub>B</sub> 死区时间计数器 x 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 DTM 内的编程设定延迟）。
<b>DTR2, DTR1, DTR0</b>	6, 5, 4	rw	<b>死区时间运行指示位</b> 位 DTR0...DTR2 指示定时器 T12 各比较通道 (0, 1, 2) 死区时间产生的状态。 0 <sub>B</sub> 死区时间计数器 x 当前处于被动状态 1 <sub>B</sub> 死区时间计数器 x 当前处于主动状态
<b>0</b>	7, 3	r	<b>保留；</b> 读操作返回 0；应写入 0。

捕获 / 比较单元 6 (CCU6)

21.3.9 捕获 / 比较控制寄存器

21.3.9.1 通道状态位

比较状态寄存器 CMPSTATL/H 中的状态位指示当前的捕获和比较状态，控制位决定比较通道处于主动态还是被动态。

**CMPSTATL**  
比较状态寄存器低位 (FE<sub>H</sub>) 复位值: 00<sub>H</sub>  
RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	CC63ST	CCPOS2	CCPOS1	CCPOS0	CC62ST	CC61ST	CC60ST
r	rh	rh	rh	rh	rh	rh	rh

符号	位	类型	描述
CC60ST, CC61ST, CC62ST, CC63ST <sup>1)</sup>	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。
CCPOS60, CCPOS61, CCPOS62	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，该值被采样 0 <sub>B</sub> 采样到的输入 CCPOS6x 为 0 1 <sub>B</sub> 采样到的输入 CCPOS6x 为 1
0	7	r	<b>保留；</b> 读操作返回 0；应写入 0。

1) 根据 T12, T13 的切换规则置位或清除这些状态位。

## 捕获 / 比较单元 6 (CCU6)

**CMPSTATH**

比较状态寄存器高位

(FF<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
<b>T13IM</b>	<b>COUT63PS</b>	<b>COUT62PS</b>	<b>CC62PS</b>	<b>COUT61PS</b>	<b>CC61PS</b>	<b>COUT60PS</b>	<b>CC60PS</b>
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>CC60PS, CC61PS, CC62PS, COUT60PS, COUT61PS, COUT62PS, COUT63PS</b> 1)	0, 2, 4, 1, 3, 5, 6	rwh	<b>比较输出的被动态选择</b> 位 CC6xPS, COUT6xPS 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, COUT6xPS (x = 0, 1, 2) 与 T12 相关, 位 CC63PS 与 T13 相关。 0 <sub>B</sub> CC6xST 为 0 时对应比较信号处于被动电平 1 <sub>B</sub> CC6xST 为 1 时对应比较信号处于被动电平 捕获模式下, 不使用这些位。
<b>T13IM</b> <sup>2)</sup>	15	rwh	<b>T13 反向调制控制</b> 位 T13IM 将 T13 信号反向用于调制 CC6x 和 COUT6x (x = 0, 1, 2) 信号。 0 <sub>B</sub> T13 输出 CC63_O 等于 <u>CC63ST</u> 1 <sub>B</sub> T13 输出 CC63_O 等于 <u>CC63ST</u>

- 1) 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。
- 2) 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 CMPMODIFL/H 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

### CMPMODIFL

比较状态修改寄存器低位

(A6<sub>H</sub>)

复位值 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	MCC63S		0		MCC62S	MCC61S	MCC60S
r	w		r		w	w	w

符号	位	类型	描述
MCC60S, MCC61S, MCC62S, MCC63S	0, 1, 2, 6	w	<b>捕获 / 比较状态修改位</b> 这些位用于软件置位（MCC6xS）或清除（MCC6xR <sup>1)</sup> ）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，允许通过一个数据写操作对 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
0	[5:3], 7	r	<b>保留；</b> 读操作返回 0；应写入 0。

1) 该位域保存在比较状态修改寄存器高位中。

捕获 / 比较单元 6 (CCU6)

CMPMODIFH

比较状态修改寄存器高位

(A7<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
0	MCC63R		0		MCC62R	MCC61R	MCC60R
r	w		0		rw	w	w

符号	位	类型	描述
MCC60R, MCC61R, MCC62R, MCC63R	0, 1, 2, 6	w	<p><b>捕获 / 比较状态寄存器修改位</b></p> <p>这些位用于软件置位 (MCC6xS<sup>1)</sup>) 或清除 (MCC6xR) 对应的 CC6xST。</p> <p>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，允许通过一个数据写操作对 CC6xST 位进行控制。</p> <p>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：</p> <p>[MCC6xR, MCC6xS] =</p> <p>00<sub>B</sub> 位 CC6xST 不变</p> <p>01<sub>B</sub> 位 CC6xST 置位</p> <p>10<sub>B</sub> 位 CC6xST 被清除</p> <p>11<sub>B</sub> 保留</p>
0	[5:3], 7	r	<p><b>保留；</b></p> <p>读操作返回 0；应写入 0。</p>

1) 该位域保存在比较状态修改寄存器低位中。

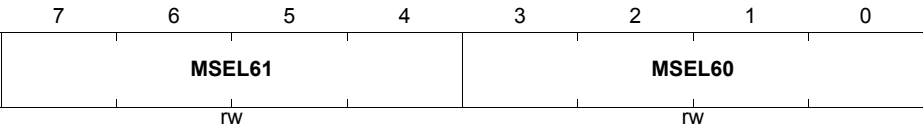


捕获 / 比较单元 6 (CCU6)

### 21.3.9.2 T12 模式控制寄存器

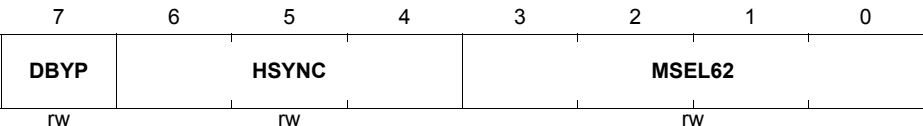
寄存器 T12MSELL/H 用于控制定时器 T12 模块中的三个通道对应的捕获 / 比较功能。

**T12MSELL**  
**T12 模式选择寄存器低位** (9A<sub>H</sub>) 复位值: 00<sub>H</sub>  
**RMAP: 0, PAGE: 2**



符号	位	类型	描述
<b>MSEL60, MSEL61</b>	[3:0], [7:4]	rw	<b>捕获 / 比较模式选择</b> 这些位域为 T12 的三个捕获 / 比较通道选择相应的工作模式。每个通道 (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)。编码见 <a href="#">表 21-10</a> 。

**T12MSELH**  
**T12 模式选择寄存器高位** (9B<sub>H</sub>) 复位值: 00<sub>H</sub>  
**RMAP: 0, PAGE: 2**



符号	位	类型	描述
<b>MSEL62</b>	[3:0]	rw	<b>捕获 / 比较模式选择</b> 这些位域为 T12 的三个捕获 / 比较通道选择相应的工作模式。每个通道 (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)。编码见 <a href="#">表 21-10</a> 。
<b>HSYNC</b>	[6:4]	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 21-15</a> 。

捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
DBYP	7	rw	<p><b>延迟旁路</b></p> <p>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。</p> <p>0<sub>B</sub> 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。</p> <p>1<sub>B</sub> 旁路有效 不使用死区时间计数器 0 产生延迟。</p>

### 21.3.9.3 定时器控制寄存器

寄存器 TCTR0L/H 控制 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。

#### TCTR0L

定时器控制寄存器 0 低位

(A6<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
CTM	CDIR	STE12	T12R	T12PRE	T12CLK		
rw	rh	rh	rh	rw	rw		

符号	位	类型	描述
T12CLK	[2:0]	rw	<p><b>定时器 T12 输入时钟选择</b></p> <p>根据等式 <math>f_{T12} = f_{CC6} / 2^{&lt;T12CLK&gt;}</math> 选择定时器 T12 的输入时钟。</p> <p>000<sub>B</sub> <math>f_{T12} = f_{CC6}</math>            001<sub>B</sub> <math>f_{T12} = f_{CC6} / 2</math>            010<sub>B</sub> <math>f_{T12} = f_{CC6} / 4</math>            011<sub>B</sub> <math>f_{T12} = f_{CC6} / 8</math>            100<sub>B</sub> <math>f_{T12} = f_{CC6} / 16</math>            101<sub>B</sub> <math>f_{T12} = f_{CC6} / 32</math>            110<sub>B</sub> <math>f_{T12} = f_{CC6} / 64</math>            111<sub>B</sub> <math>f_{T12} = f_{CC6} / 128</math></p>
T12PRE	3	rw	<p><b>定时器 T12 预分频使能位</b></p> <p>为了支持更高时钟频率，可使能分频因子为 1/256 的 T12 附加预分频器。</p> <p>0<sub>B</sub> T12 附加预分频器被禁止            1<sub>B</sub> T12 附加预分频器被使能</p>
T12R	4	rh	<p><b>定时器 T12 运行位<sup>1)</sup></b></p> <p>T12R 用于启动和终止定时器 T12。置位 T12RS/T12RR 可软件置位 / 复位 T12R。或根据位域 T12SSC 定义的功能硬件复位 T12R。</p> <p>0<sub>B</sub> 定时器 T12 被终止            1<sub>B</sub> 定时器 T12 正在运行</p>

捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
STE12	5	rh	<b>定时器 T12 映射传送使能</b> 如果检测到 T12 映射传送事件，位 STE12 用于使能或禁止 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到实际寄存器的映射传送。映射传送之后硬件清除位 STE12。 递增计数时，T12 映射传送事件为周期匹配；递减计数时，映射传送事件为 1- 匹配。 0 <sub>B</sub> 映射寄存器传送被禁止 1 <sub>B</sub> 映射传送被使能
CDIR	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。 0 <sub>B</sub> T12 递增计数 1 <sub>B</sub> T12 递减计数
CTM	7	rw	<b>T12 的工作模式</b> 0 <sub>B</sub> 边沿对齐模式： T12 始终递减计数，到达周期值之后，从 0 开始继续计数。 1 <sub>B</sub> 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。

1) 由 ( T12SSC, T12RR 或 T12RS) 同时置位 / 复位 T12R 的操作无效，位 T12R 保持不变。

TCTR0H

定时器控制寄存器 0 高位

(A7<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
0	STE13	T13R	T13PRE			T13CLK	
r	rh	rh	rw			rw	

## 捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
<b>T13CLK</b>	[2:0]	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 $000_B \quad f_{T13} = f_{CC6}$ $001_B \quad f_{T13} = f_{CC6} / 2$ $010_B \quad f_{T13} = f_{CC6} / 4$ $011_B \quad f_{T13} = f_{CC6} / 8$ $100_B \quad f_{T13} = f_{CC6} / 16$ $101_B \quad f_{T13} = f_{CC6} / 32$ $110_B \quad f_{T13} = f_{CC6} / 64$ $111_B \quad f_{T13} = f_{CC6} / 128$
<b>T13PRE</b>	3	rw	<b>定时器 T13 预分频使能位</b> 为了支持更高时钟频率，可使能分频因子为 1/256 的 T13 附加预分频器。 $0_B \quad$ T13 的附加预分频器被禁止 $1_B \quad$ T13 的附加预分频器被使能
<b>T13R</b>	4	rh	<b>定时器 T13 运行位<sup>1)</sup></b> T13R 用于启动和终止定时器 T13。置位 T13RS/T13RR 可软件置位 / 复位 T13R。或根据位域 T13SSC, T13TEC 和 T13TED 定义的功能由硬件置位 / 清除 T13R。 $0_B \quad$ 定时器 T13 被终止 $1_B \quad$ 定时器 T13 正在运行
<b>STE13</b>	5	rh	<b>定时器 T13 映射传送使能</b> 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。 T13 映射传送事件为周期匹配。 T13 映射传送事件为周期匹配。 $0_B \quad$ 禁止映射寄存器传送 $1_B \quad$ 使能映射寄存器传送
<b>0</b>	[7: 6]	r	<b>保留；</b> 读操作返回 0；应写入 0。

1) (由 T13SSC, T13TEC, T13RR 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 TCTR2L/H 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。

**TCTR2L**  
**定时器控制寄存器 2 低位** (FA<sub>H</sub>) **复位值: 00<sub>H</sub>**  
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
0	T13TED		T13TEC			T13SSC	T12SSC
r	rw		rw			rw	rw

符号	位	类型	描述
T12SSC	0	rw	<b>定时器 T12 单次模式控制</b> 该位控制 T12 的单次模式。 0 <sub>B</sub> 单次模式被禁止，硬件不控制 T12R。 1 <sub>B</sub> 单次模式被使能，如果满足以下条件，位 T12R 被硬件清除： - 边沿对齐模式下，T12 达到周期值 - 中心对齐模式下，递减计数时，T12 计数至 1。 清除位 T12R 的同时，位 CC6xST (x=0, 1, 2) 也被清除
T13SSC	1	rw	<b>定时器 T13 单次模式控制</b> 该位控制 T13 的单次模式。 0 <sub>B</sub> 单次模式被禁止，硬件不控制 T13R 1 <sub>B</sub> 单次模式被使能，如果 T13 达到其周期值，位 T13R 被硬件清除。清除 T13R 的同时，位 CC63ST 也被清除。

## 捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
<b>T13TEC</b>	[4:2]	rw	<b>T13 触发事件控制</b> 根据下面的组合，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）。 000 <sub>B</sub> 无操作 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> 由输入 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>0</b>	7	r	<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。

**TCTR2H**

定时器控制寄存器 2 高位

(FB<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0				T13RSEL		T12RSEL	
r				rw		rw	

**捕获 / 比较单元 6 (CCU6)**

符号	位	类型	描述
<b>T12RSEL</b>	[1:0]	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>	[3:2]	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。 10 <sub>B</sub> 在 T13HR 上检测到下降沿时，位 T13R 被置位。 11 <sub>B</sub> 在 T13HR 上检测到任意信号沿时，位 T13R 被置位。
<b>0</b>	[7:4]	r	<b>保留；</b> 读操作返回 0；应写入 0。



## 捕获 / 比较单元 6 (CCU6)

寄存器 TCTR4L/H 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。

### TCTR4L

定时器控制寄存器 4 低位

(9C<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
T12STD	T12STR	0	DTRES	T12RES	T12RS	T12RR	
w	w	r	w	w	w	w	

符号	位	类型	描述
T12RR	0	w	定时器 T12 运行位复位 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
T12RS	1	w	定时器 T12 运行位置位 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
T12RES	2	w	定时器 T12 复位 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。
DTRES	3	w	死区时间计数器复位 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
T12STR	6	w	定时器 T12 映射传送请求 0 <sub>B</sub> 无操作 1 <sub>B</sub> STE12 被置位，使能映射传送
T12STD	7	w	定时器 T12 映射传送禁止 0 <sub>B</sub> 无操作 1 <sub>B</sub> STE12 被清除，不触发映射传送
0	[5:4]	r	保留； 读操作返回 0；应写入 0。

注：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。

## 捕获 / 比较单元 6 (CCU6)

**TCTR4H**

定时器控制寄存器 4 高位

(9D<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
<b>T13STD</b>	<b>T13STR</b>		<b>0</b>		<b>T13RES</b>	<b>T13RS</b>	<b>T13RR</b>
w	w		r		w	w	w

符号	位	类型	描述
<b>T13RR</b>	0	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>	1	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>	2	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>T13STR</b>	6	w	<b>定时器 T13 映射传送请求</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> STE13 被置位, 使能映射传送
<b>T13STD</b>	7	w	<b>定时器 T13 映射传送禁止</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> STE13 被复位, 不触发映射传送
<b>0</b>	[5:3]	r	<b>保留;</b> 读操作返回 0; 应写入 0。

注: 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。

## 21.4 定时器 T13 模块

定时器 T13 和定时器 T12 类似，但只有一个比较通道。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。

- T13 概述（见章节 21.4.1）
- 计数策略（见章节 21.4.2）
- 比较模式（见章节 21.4.3）
- 比较输出通路（见章节 21.4.4）
- 映射寄存器传送（见章节 21.4.5）
- T13 计数寄存器描述（见章节 21.4.6）

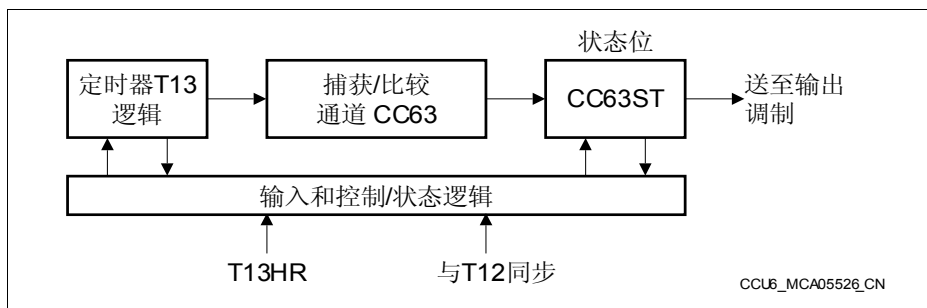


图 21-23 定时器 T13 模块框图

### 21.4.1 T13 模块概述

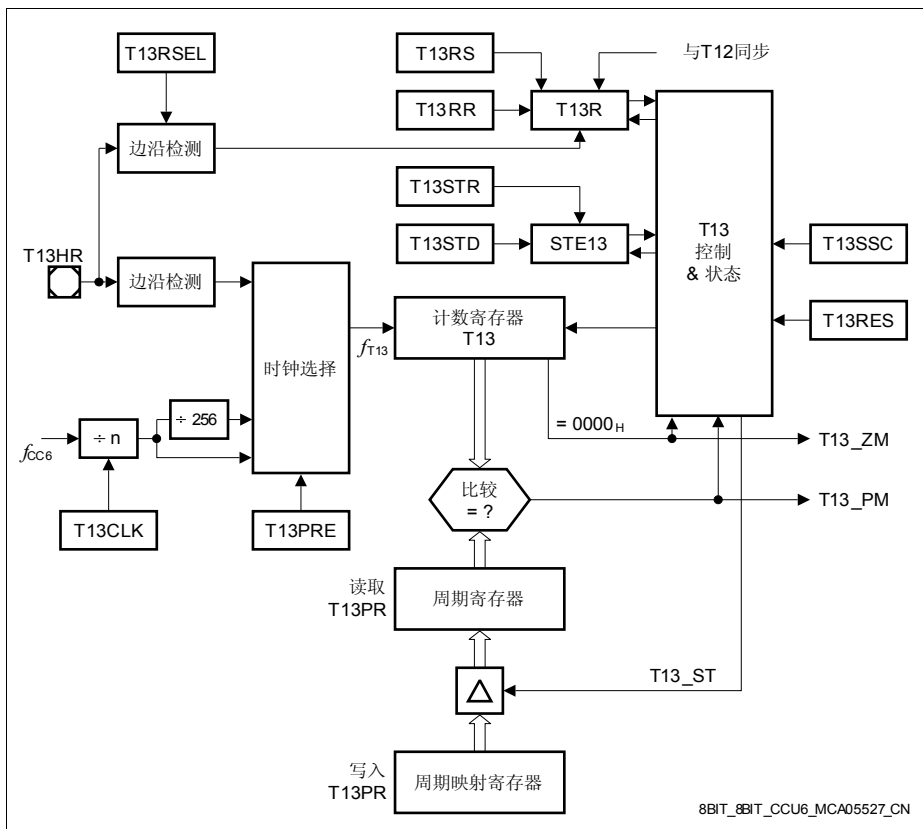
图 21-24 所示为定时器 T13 的内部结构框图。由寄存器 **TCTR0L**, **TCTR0H**, **TCTR2L**, **TCTR2H**, **TCTR4L**, **TCTR4H** 和 **PISEL2** 控制定时器 T13 模块。

定时器 T13 的输入时钟  $f_{T13}$  来自经过预分频处理的（预分频因子可编程设定）和 1/256 分频（可选）的模块时钟  $f_{CC6}$ 。T13 只能递增计数（类似于 T12 的边沿对齐模式）。

计数器寄存器 **T13L**, **T13H** 通过比较器和周期寄存器 **T13PRL**, **T13PRH** 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  $0000_H$ 。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**。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 21.4.5）。

信号 **T13\_ZM** 指示计数器的值是否等于  $0000_H$ 。

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（图 21-24）。



**图 21-24 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 **TCTR4L**, **TCTR4H** 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，即 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。

---

## 捕获 / 比较单元 6 (CCU6)

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。写操作写入映射寄存器；读操作读取实际周期寄存器的值。

## 21.4.2 T13 计数策略

本节描述 T13 的时钟和计数功能。

### 21.4.2.1 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：

$$T13_{PER} = \text{周期值} + 1; \text{以 } T13 \text{ 时钟频率 } (f_{T13}) \text{ 计数} \quad (21.3)$$

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 21-25 所示。

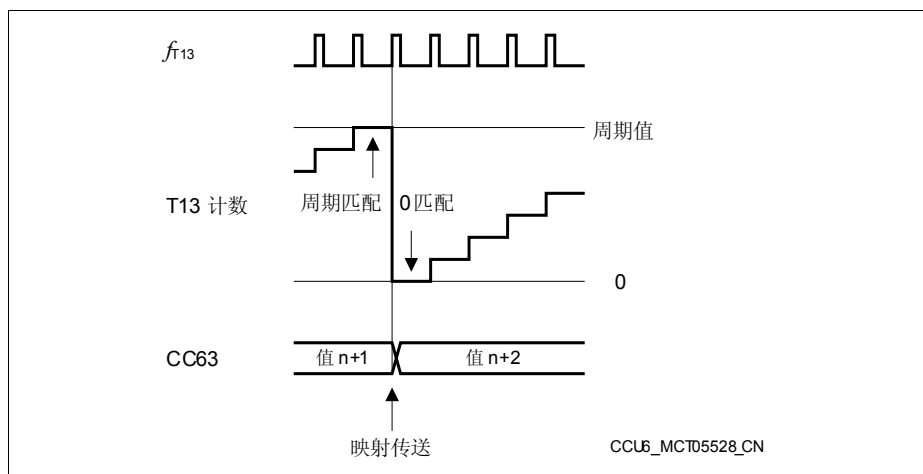


图 21-25 T13 计数序列

## 捕获 / 比较单元 6 (CCU6)

### 21.4.2.2 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。

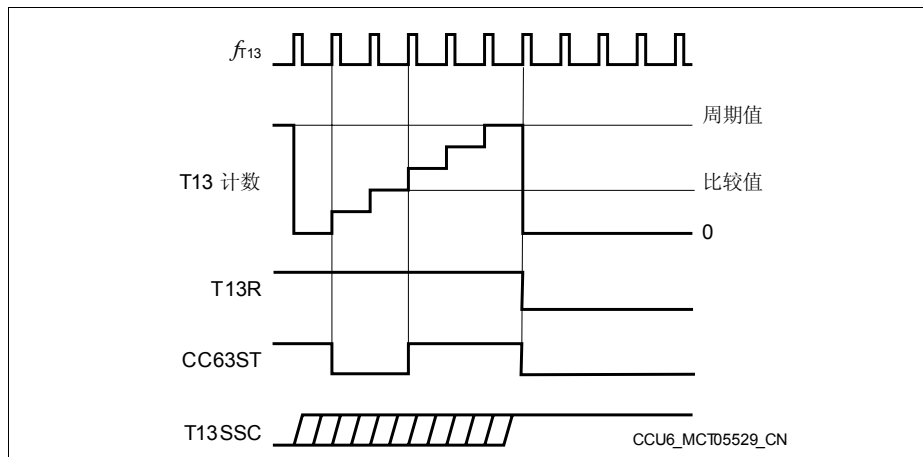
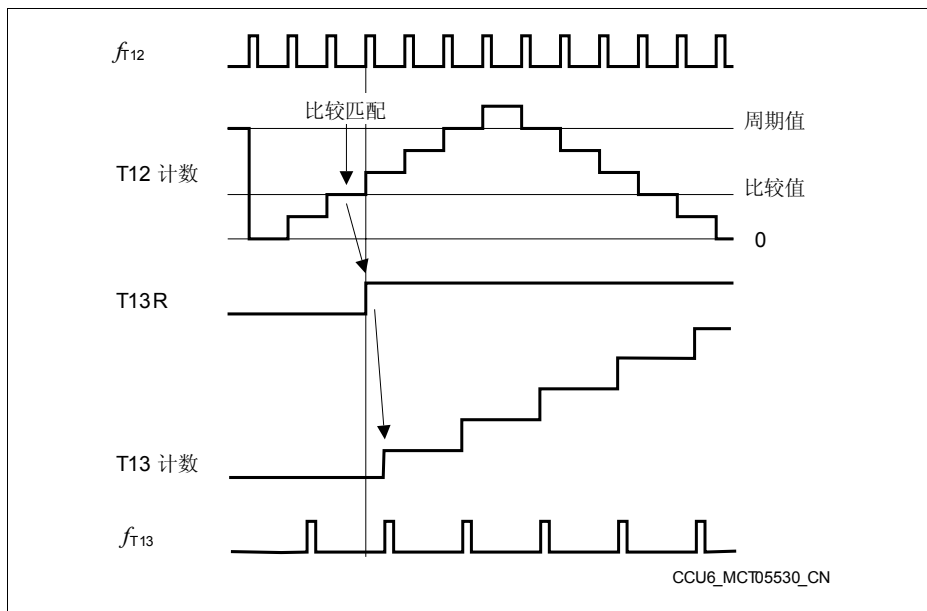


图 21-26 定时器 T13 的单次操作

### 21.4.2.3 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 21-27** 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 21-27 T13 与 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 21-11 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 21-12）。



表 21-11 T12 触发事件选择

T13TEC	事件选择
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 21-12 T12 触发事件附加条件

T13TED	事件选择附加条件
00 <sub>B</sub>	保留，无操作
01 <sub>B</sub>	T12 递增计数时 (CDIR = 0) 选定的触发事件有效
10 <sub>B</sub>	T12 递减计数时 (CDIR = 1) 选定的触发事件有效
11 <sub>B</sub>	无论 T12 递增 / 递减计数，选定的触发事件均有效

### 21.4.3 T13 比较模式

定时器 T13 的单个比较通道可执行（和 T13 的计数值有关的）比较操作。

图 21-23 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 CC63RL, CC63RH 和对应映射寄存器 CC63SRL, CC63SRH）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 CMPSTATL.CC63ST 用于保存比较操作的状态，图 21-28 给出状态位的逻辑操作。

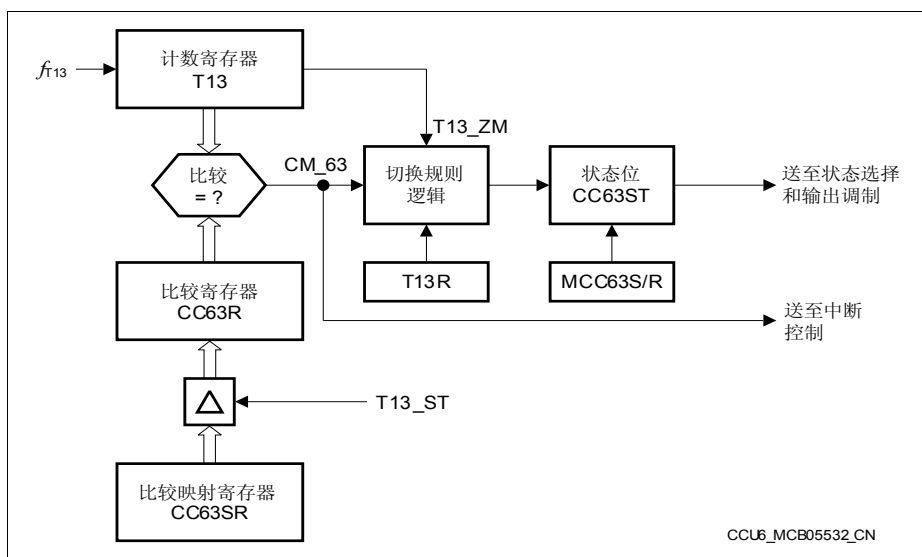


图 21-28 T13 状态位框图

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，可通过设置置位控制位 MCC63S 和复位控制位 MCC63R（位于寄存器 CMPMODIFL, CMPMODIFH 中）分别置位或复位 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：

## 捕获 / 比较单元 6 (CCU6)

- 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** 被复位至 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。

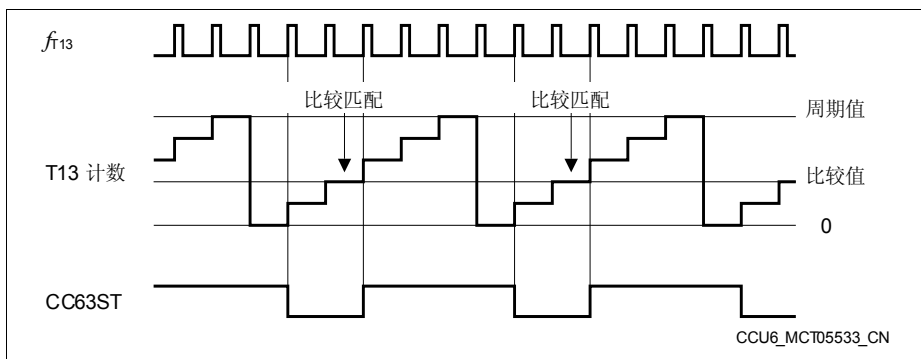


图 21-29 T13 比较操作

## 21.4.4 比较模式输出路径

图 21-30 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 21.3.4.3。

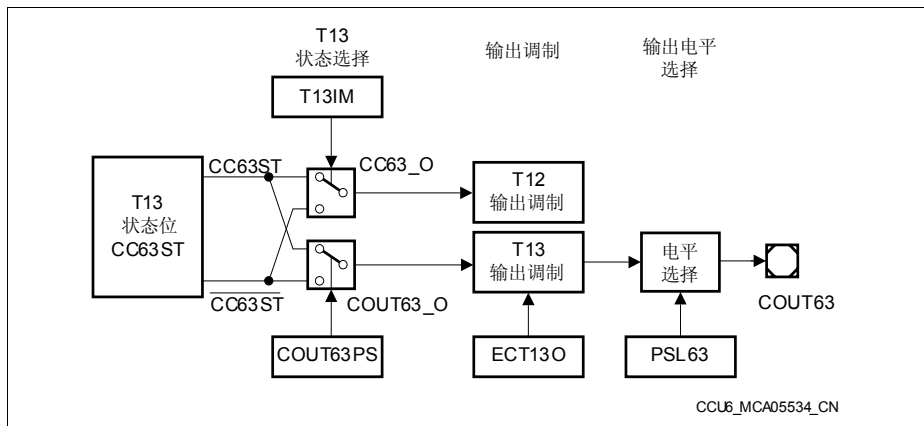


图 21-30 通道 CC63 输出通路

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 21-31）：

- T13 相关比较信号 COUT63\_O 由 T13 状态选择输出，由位 MODCTRH.ECT13O 使能
- 强制中断状态 TRPS 具有单独的使能位 TRPCTRH.TRPEN13。

如果调制输入信号 COUT63\_O 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。

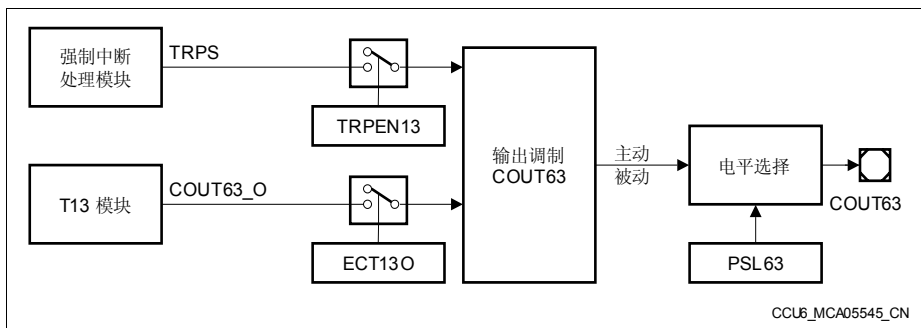


图 21-31 T13 输出调制

### 21.4.5 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 **TCTR0H.STE13** (向只写位 **TCTR4H.T13STR** 写入 1 将置位 STE13；向只写位 **TCTR4H.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。

## 捕获 / 比较单元 6 (CCU6)

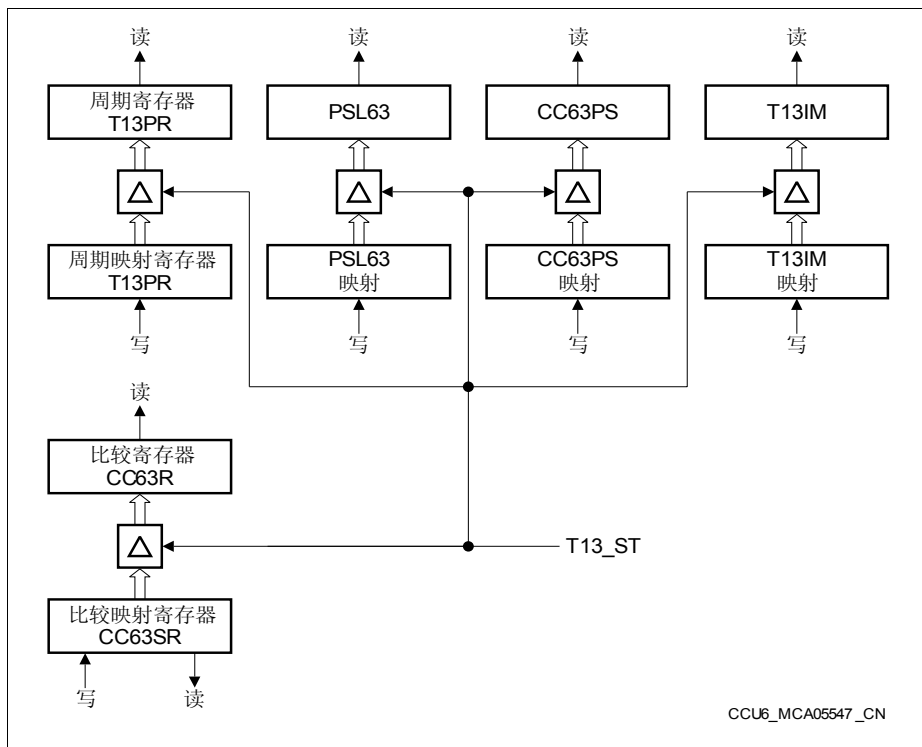


图 21-32 T13 映射寄存器概览

## 21.4.6 T13 相关寄存器

### 21.4.6.1 T13 计数寄存器

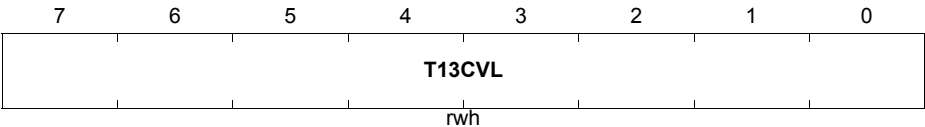
基于定时器 T13 产生单个通道脉宽调制 (PWM) 信号的序列。相关的定时器 T13 寄存器可同时更新 (以精心定义的条件) 以确保 PWM 信号的一致性。T13 还可和多个定时器 T12 事件同步。

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)。

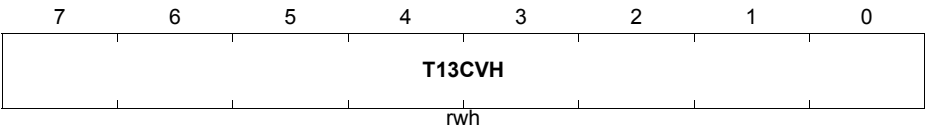
**T13L**  
 定时器 T13 计数寄存器低位 (FC<sub>H</sub>) 复位值: 00<sub>H</sub>  
 RMAP: 0, PAGE: 3



符号	位	类型	描述
T13CVL	[7:0]	rwh	定时器 T13 计数值 定时器 T13 的 16 位计数值的低 8 位。

*注: 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。*

**T13H**  
 定时器 T13 计数寄存器高位 (FD<sub>H</sub>) 复位值: 00<sub>H</sub>  
 RMAP: 0, PAGE: 3



符号	位	类型	描述
T13CVH	[7:0]	rwh	定时器 T13 计数值 定时器 T13 的 16 位计数值的高 8 位。

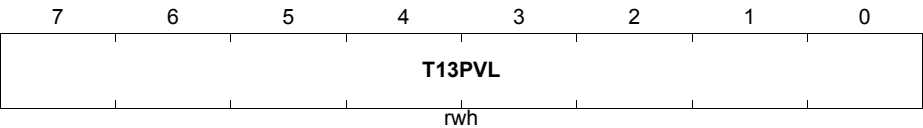
捕获 / 比较单元 6 (CCU6)

注： 定时器停止时，内部时钟分频器复位以确保产生可重复的时序和延迟。

21.4.6.2 周期寄存器

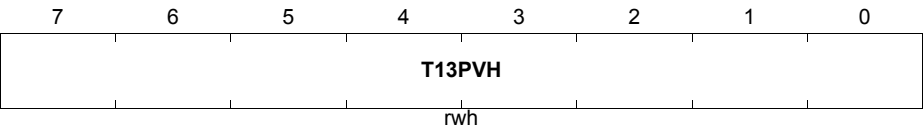
寄存器 T13PR 中存放着定时器 T13 的周期值。周期值和 T13 的当前计数值进行比较，由计数规则决定计数器的后续操作。该寄存器对应有一个映射寄存器，由位 STE13 控制映射传送。读操作读取当前用于比较操作的周期值；写操作写入映射寄存器；映射寄存器结构便于用户同时更新 T13 的所有相关值。

**T13PRL**  
 定时器 T13 周期寄存器低位 (9E<sub>H</sub>) 复位值：00<sub>H</sub>  
 RMAP: 0, PAGE: 1



符号	位	类型	描述
<b>T13PVL</b>	[7:0]	rwh	<b>T13 周期值</b> T13PVL 定义 T13 周期值的低 8 位。计数到周期值时，定时器 T13 被置为 0。

**T13PRH**  
 定时器 T13 周期寄存器高位 (9F<sub>H</sub>) 复位值：00<sub>H</sub>  
 RMAP: 0, PAGE: 1



符号	位	类型	描述
<b>T13PVH</b>	[7:0]	rwh	<b>T13 周期值</b> T13PVH 定义 T13 的周期值的高 8 位。计数到周期值时，定时器 T13 置为 0。



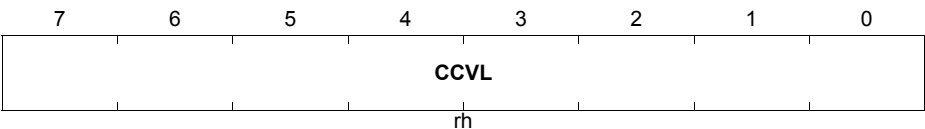
捕获 / 比较单元 6 (CCU6)

21.4.6.3 比较寄存器

寄存器 CC63RL/H 寄存器是 T13 的实际比较寄存器。CC63RL/H 中的值和 T13 计数值进行比较。状态位 CC63ST 位于寄存器 **CMPSTATL** 内。

**CC63RL**

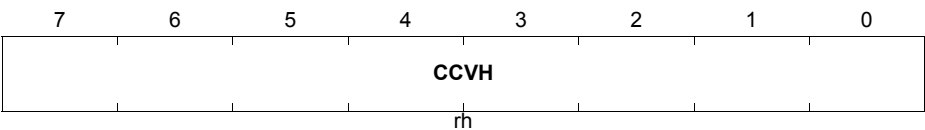
**T13 比较寄存器低位** (9A<sub>H</sub>) **复位值: 00<sub>H</sub>**  
**RMAP: 0, PAGE: 1**



符号	位	类型	描述
CCVL	[7:0]	rh	通道 <b>CC63</b> 比较值 该位域包含 CCV 的低 8 位值，使用该值和 T13 计数值进行比较。

**CC63RH**

**T13 比较寄存器高位** (9B<sub>H</sub>) **复位值: 00<sub>H</sub>**  
**RMAP: 0, PAGE: 1**



符号	位	类型	描述
CCVH	[7:0]	rh	通道 <b>CC63</b> 比较值 该位域包含 CCV 的高 8 位值，使用该值和 T13 计数值进行比较。

## 捕获 / 比较单元 6 (CCU6)

### 21.4.6.4 比较映射寄存器

寄存器 CC63RL/H 只能由软件读取，通过将映射寄存器 CC63SRL/H 中的值映射传送的方式修改 CC63RL/H 的值。映射寄存器 CC63SRL/H 可软件读 / 写。

#### CC63SRL

##### T13 比较映射寄存器

(9A<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
CCSL							
rw							

符号	位	类型	描述
CCSL	[7:0]	rw	通道 CC63 比较值映射寄存器 映射传送过程中，CCSL 的内容被传送至 CCV 的低 8 位。

#### CC63SRH

##### T13 比较映射寄存器高位

(9B<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
CCSH							
rw							

符号	位	类型	描述
CCSH	[7:0]	rw	通道 CC63 比较值映射寄存器 映射传送过程中，CCSH 的内容被传送至 CCV 的高 8 位。

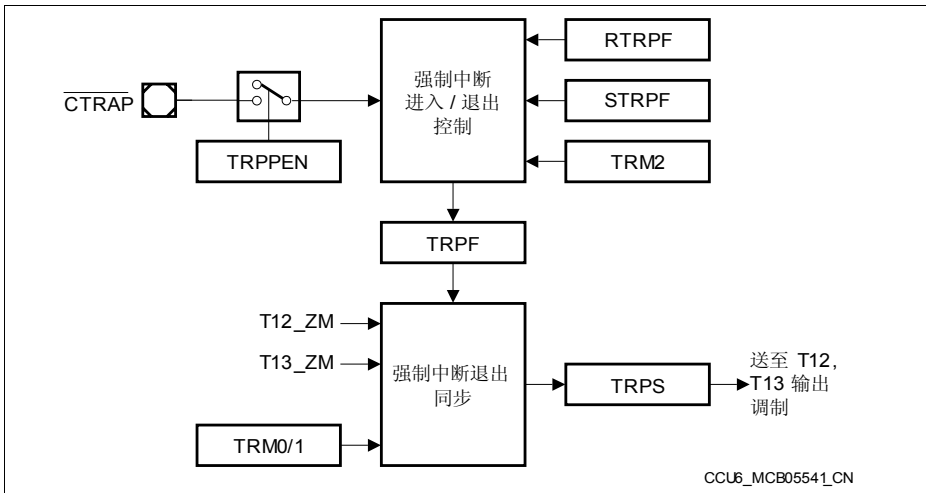
## 21.5 强制中断处理

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 **TRPCTRL**，**TRPCTR** 中的位控制。强制中断标志 **TRPF** 和 **TRPS** 位于寄存器 **ISH** 中，通过对寄存器 **ISSH** 和 **ISRH** 进行写操作可软件置位 / 清除这些标志。**图 21-33** 给出强制中断功能的概述。

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（ $\text{CTRAP} = 0$ ）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** 的六个输出和 **T13** 的单个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（ $\text{CTRAP} = 1$  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 21-33 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（ $\text{CTRAP} = 1$ ），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 **TRPCTRL/H** 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

# 捕获 / 比较单元 6 (CCU6)

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 21-34 给出相关操作。

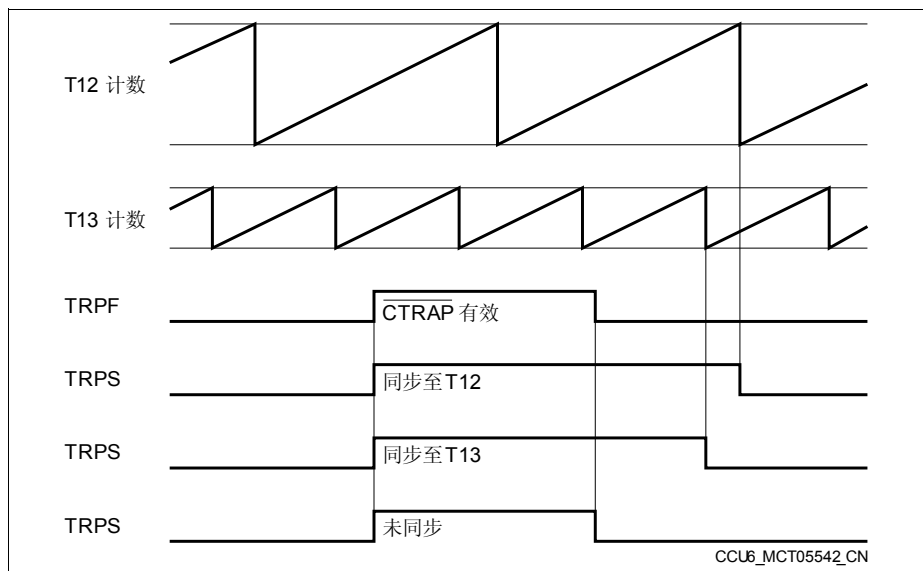


图 21-34 强制中断状态同步 (TRM2 = 0)

## 21.6 多通道模式

多通道模式可用一条指令调制全部六个 T12 的相关输出信号。位域 **MCMOUTL.MCMP** 规定哪些输出有效。若多通道模式被使能（位 **MODCTRL.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 **MCMOUTSL.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。

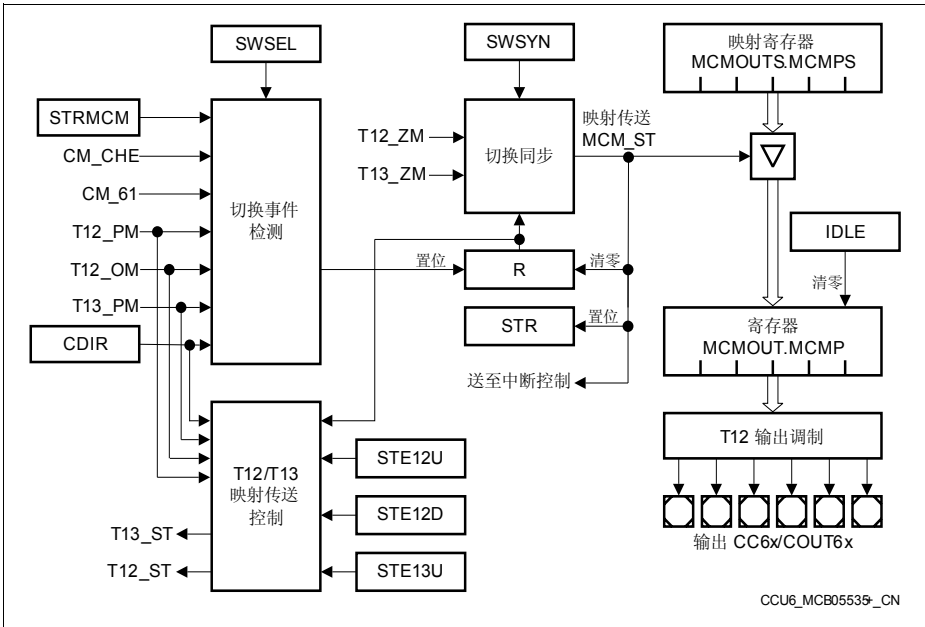


图 21-35 多通道模式框图

图 21-35 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 **MCMP** 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 **MCMP** 到 **MCMP** 的映射传送，位 **ISH.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。

## 捕获 / 比较单元 6 (CCU6)

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 MCMP5 且设置映射传送请求位 STRMCM = 1。对应于所有 SWSSEL 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见表 21-13 和表 21-14。

**表 21-13 多通道模式切换事件选择**

SWSSEL	选择的事件（见寄存器 MCMCTR）
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 21-14 多通道模式切换同步**

SWSYN	同步事件（见寄存器 MCMCTR）
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

## 21.7 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 MCMOUTL，MCMOUTH 中存放着实际使用的值；对应的映射寄存器 MCMOUTSL，MCMOUTSH 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

### 21.7.1 霍尔序列评估

跳变沿检测模块以模块时钟  $f_{CC6}$  永久的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的监控霍尔输入，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。

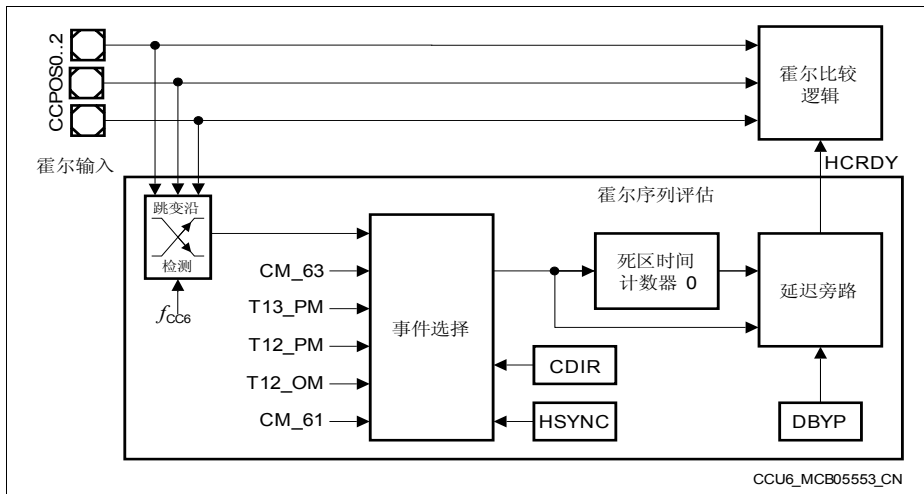


图 21-36 霍尔序列评估

如果评估信号 HCRDY（霍尔比较准备就绪，见图 21-37）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 21-36 给出霍尔序列评估事件和噪声滤波逻辑。

表 21-15 总结出可选的触发输入信号。



表 21-15 霍尔传感器模式触发事件选择

HSYNC	事件选择 (见寄存器 T12MSELL, T12MSELH)
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)

## 21.7.2 霍尔序列比较逻辑

图 21-37 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 CMPSTATL, CMPSTATH 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。

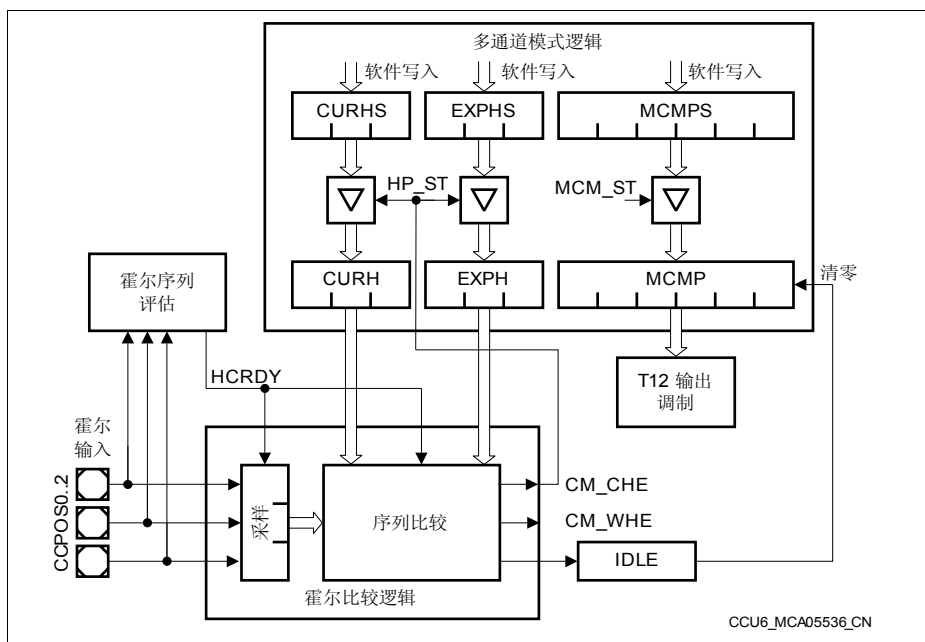


图 21-37 霍尔序列比较逻辑

- 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。

向 MCMOUTS.STRHP 写 1 (对于 EXPH 和 CURH 来讲) 或 MCMOUTS.STRMCMP = 1 (对于 MCMP 来讲), 也可实现映射传送。

### 21.7.3 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 **ISH.CHE** (正确的霍尔事件)。也可通过软件置位寄存器 **ISSH.SCHE** 对该标志位置位。若寄存器 **IENTH.ENCHE** 被使能, CHE 的置位信号还可向 CPU 发送中断请求。位域 **INPL.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 **ISRH.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 21-35)。

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。

## 捕获 / 比较单元 6 (CCU6)

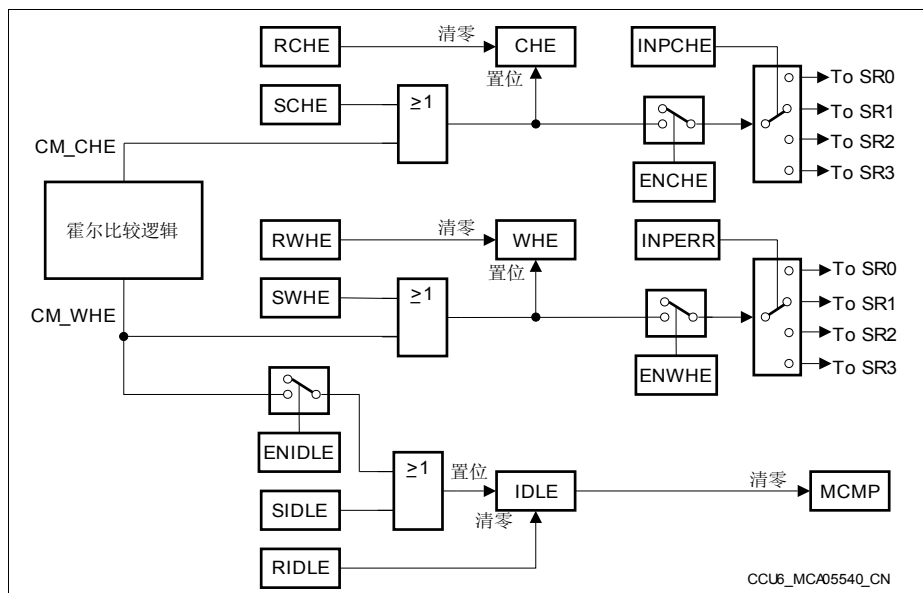


图 21-38 霍尔模式标志

### 21.7.4 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 三个通道的位域 **T12MSELL/T12MSELH.MSELx** 全部设定为 1000<sub>B</sub>，选定该模式。该模式下，如图 21-39 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。

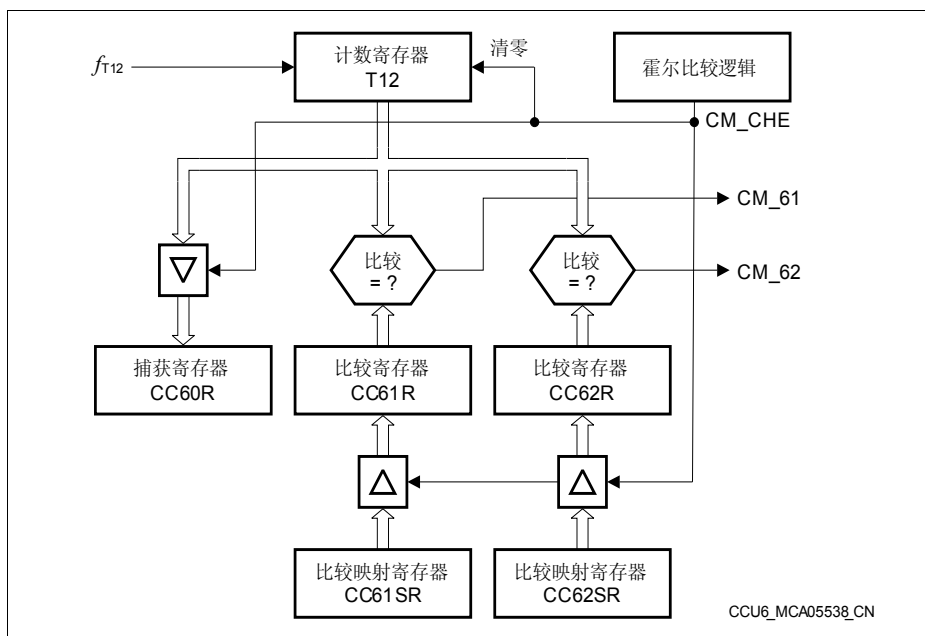
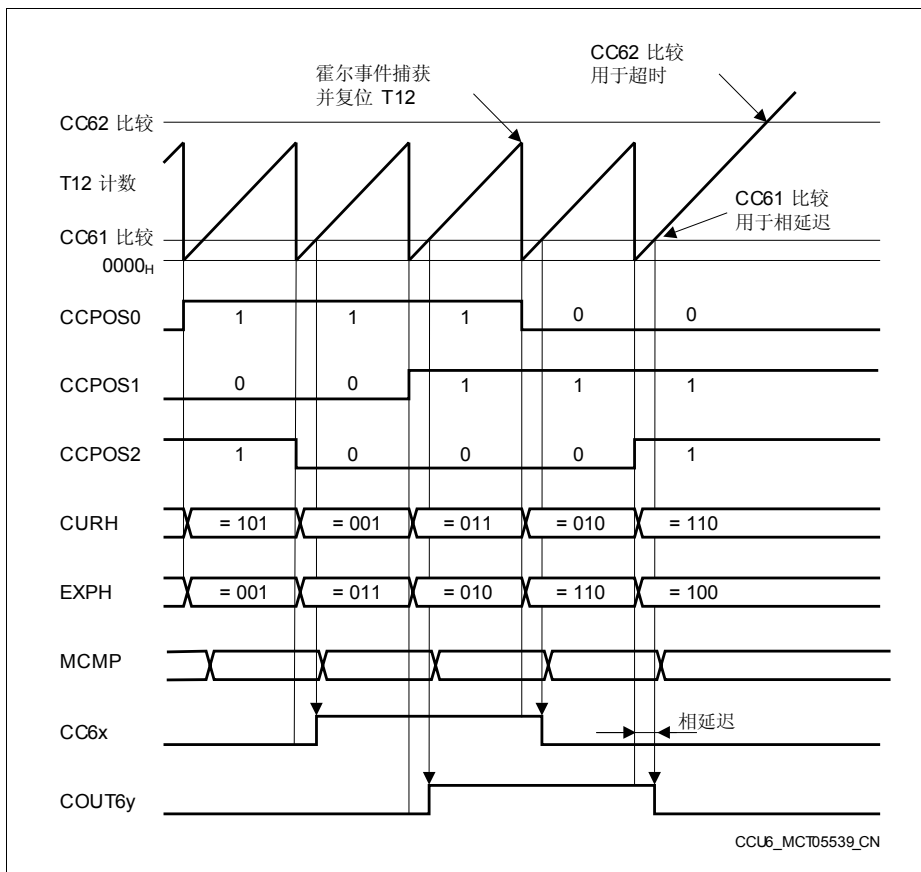


图 21-39 霍尔传感器模式下的 T12 模块

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。

## 捕获 / 比较单元 6 (CCU6)


图 21-40 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, 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 21.6)。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。

## 21.8 调制控制寄存器

### 21.8.1 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。

#### MODCTRL

调制控制寄存器低位

(FC<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
MCMEN	0	T12MODEN					
rw	r	rw					

符号	位	类型	描述
T12MODEN	[5:0]	rw	<b>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 序列调制相应输出信号
MCMEN	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信号
0	6	r	<b>保留；</b> 读操作返回 0；应写入 0。

## 捕获 / 比较单元 6 (CCU6)

## MODCTRH

调制控制寄存器高位

(FD<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
ECT13O	0	T13MODEN					
rw	r	rw					

符号	位	类型	描述
T13MODEN	[5:0]	rw	<b>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。 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 序列调制相应输出信号
ECT13O	7	rw	<b>定时器 T13 比较输出使能</b> 0 <sub>B</sub> 输出 COUT63 处于被动态 1 <sub>B</sub> 输出 COUT63 被使能, 输出 T13 产生的 PWM 信号。
0	6	r	<b>保留;</b> 读操作返回 0; 应写入 0。



捕获 / 比较单元 6 (CCU6)

### 21.8.2 强制中断控制寄存器

寄存器 TRPCTRL/H 控制强制中断功能。其中存放着各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，CTRAP 输入引脚为低电平，由位 ISH.TRPF 监控（反向电平）。当 TRPF = 1（强制中断输入有效），强制中断状态位 IS.TRPS 被设置为 1。

#### TRPCTRL

强制中断控制寄存器低位

(FE<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
		0			TRPM2	TRPM1	TRPM0
		r			rw	rw	rw

符号	位	类型	描述
TRPM1, TRPM0	1, 0	rw	<p><b>强制中断模式控制位 1, 0</b></p> <p>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。</p> <p>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。</p> <p>[TRPM1, TRPM0] 的组合导致：</p> <p>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）。</p> <p>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）。</p> <p>10<sub>B</sub> 保留</p> <p>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）。</p>

## 捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
TRPM2	2	rw	<b>强制中断模式控制位 2</b> 该位定义强制中断输入条件 ( $\overline{\text{CTRAP}} = 0$ 和 $\text{TRPPEN} = 1$ ) 不再有效时 (或者 $\text{CTRAP} = 1$ 或者 $\text{TRPPEN} = 0$ )，如何清除强制中断标志。 $0_B$ 自动模式： 如果强制中断输入条件不再有效，由硬件自动清除位 TRPF $1_B$ 手动模式： 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 $\text{ISR.RTRPF} = 1$ ，软件清除 TRPF。
0	[7:3]	r	<b>保留；</b> 读操作返回 0；应写入 0。

## TRPCTRH

强制中断控制寄存器高位

(FF<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
TRPPEN	TRPEN13				TRPEN		
rw	rw				rw		

符号	位	类型	描述
TRPEN	[5:0]	rw	<b>强制中断使能控制</b> 置位各控制位可使能下述对应输出信号的强制中断功能： $\text{TRPEN0} = \text{TRPCTR.8}$ 对应输出 CC60 $\text{TRPEN1} = \text{TRPCTR.9}$ 对应输出 COUT60 $\text{TRPEN2} = \text{TRPCTR.10}$ 对应输出 CC61 $\text{TRPEN3} = \text{TRPCTR.11}$ 对应输出 COUT61 $\text{TRPEN4} = \text{TRPCTR.12}$ 对应输出 CC62 $\text{TRPEN5} = \text{TRPCTR.13}$ 对应输出 COUT62 $0_B$ 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。 $1_B$ 相应输出信号的强制中断功能被使能。当 $\text{IS.TRPS} = 1$ 时，输出被设置为被动态。

捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
TRPEN13	6	rw	<p><b>T13 输出的强制中断使能控制</b></p> <p>0<sub>B</sub> 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>1<sub>B</sub> 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>
TRPPEN	7	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 CTRAP 上检测到下降沿，才产生中断。</p> <p>0<sub>B</sub> 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位 TRPF 产生。</p> <p>1<sub>B</sub> 根据输入引脚 <math>\overline{\text{CTRAP}}</math> 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 CTRAP=0 产生。</p>

### 21.8.3 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向。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（反向或不反向）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** (A6<sub>H</sub>) 复位值: 00<sub>H</sub>  
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
PSL63	0	PSL					
rwh	r	rwh					

符号	位	类型	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	r	<b>保留；</b> 读操作返回 0；应写入 0。

## 捕获 / 比较单元 6 (CCU6)

### 21.8.4 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

(A7<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0	SWSYN			0	SWSEL		
r	rw			r	rw		

符号	位	类型	描述
SWSEL	[2:0]	rw	<b>切换选择</b> 位域 SWSEL 选择从 MCMP <sub>S</sub> 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。 000 <sub>B</sub> 不产生触发请求 001 <sub>B</sub> 检测到正确霍尔事件 (CM_CHE) 010 <sub>B</sub> 检测到 T13 周期匹配 (递增计数时) 011 <sub>B</sub> T12 1- 匹配 (递减计数时) 100 <sub>B</sub> T12 通道 1 比较匹配 (相位延迟功能) 101 <sub>B</sub> 检测到 T12 周期匹配 (递增计数时) 110 <sub>B</sub> 保留, 不产生触发请求 111 <sub>B</sub> 保留, 不产生触发请求
SWSYN	[5:4]	rw	<b>切换同步</b> 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。 00 <sub>B</sub> 直接; 触发事件立即引发映射传送 01 <sub>B</sub> T13 0- 匹配触发映射传送 10 <sub>B</sub> T12 0- 匹配 (递增计数时) 触发映射传送 11 <sub>B</sub> 保留; 无操作。
0	[7:6], 3	r	<b>保留;</b> 读操作返回 0; 应写入 0。

## 捕获 / 比较单元 6 (CCU6)

寄存器 MCMOUTSL/H 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。

### MCMOUTSL

多通道模式输出映射寄存器低位

(9E<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
STRMCM	0	MCMPS					
w	r	rw					

符号	位	类型	描述
MCMPS	[5:0]	rw	多通道 PWM 序列映射位域 MCMPS 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。
STRMCM	7	w	MCMPS 的映射传送请求 置位 STRMCM = 1 立即激活 MCM_ST，用位域 MCMPS 的值更新位域 MCMP。 读取该位时始终返回 0。 读取该位始终返回 0。 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位域 MCMP 被更新
0	6	r	保留； 读操作返回 0；应写入 0。

### MCMOUTSH

多通道模式输出映射寄存器高位

(9F<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
STRHP	0	CURHS			EXPHS		
rw	r	rw			rw		

捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
EXPHS	[2:0]	rw	期望霍尔序列映射位域 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。
CURHS	[5:3]	rw	当前霍尔序列映射位域 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。
STRHP	7	w	霍尔序列映射传送请求 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
0	6	r	保留； 读操作返回 0；应写入 0。

MCMOUTL  
多通道模式输出寄存器低位

(9A<sub>H</sub>) 复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0	R	MCMF					
r	rw	rw					

捕获 / 比较单元 6 (CCU6)

符号	位	类型	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> <p>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。</p>
<b>0</b>	6	r	<p><b>保留；</b></p> <p>读操作返回 0；应写入 0。</p>

**MCMOUTH**

多通道模式输出寄存器高位

(9B<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0		CURH			EXPH		
r		rw			rw		



## 捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
<b>EXPH</b>	[10:8]	rh	<p><b>期望霍尔序列</b>  由映射传送 HP_ST 用位域 EXPHS 的值更新位域 EXPH。  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。  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>  由映射传送 HP_ST 用位 CURHS 的值更新位域 CURH。  如果 HCRDY = 1，为了检测毛刺，CURH 和 CCPOSx 输入采样进行比较。  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。  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>	[7:6]	r	<p><b>保留；</b>  读操作返回 0；应写入 0。</p>

## 21.9 中断处理

本节描述 CCU6 模块的中断处理。

### 21.9.1 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 21-41）。

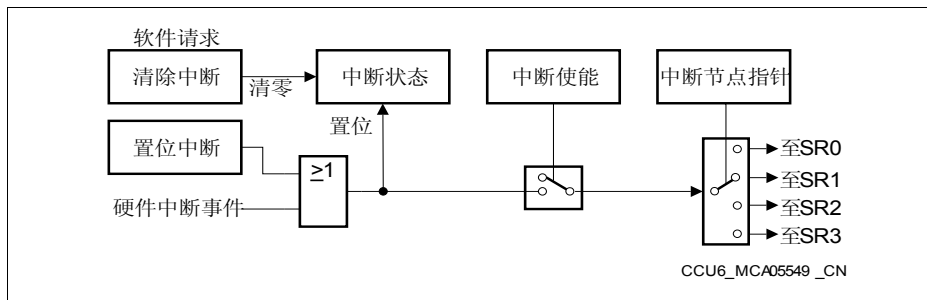


图 21-41 一般中断结构

CCU6 模块的可用中断事件如图 21-42 所示。

## 捕获 / 比较单元 6 (CCU6)

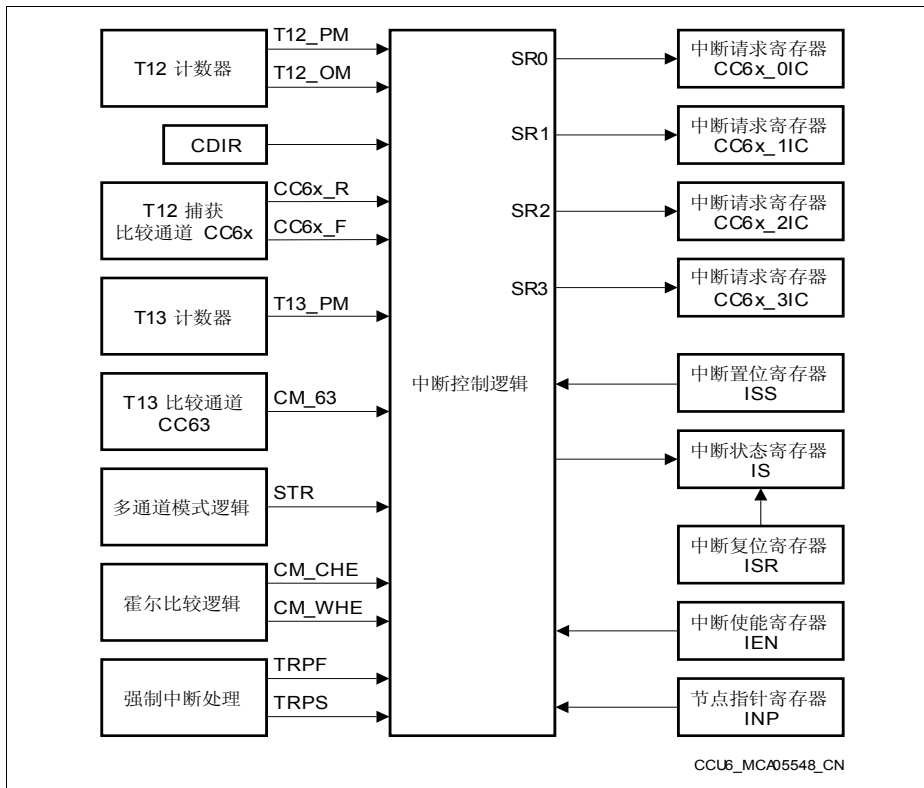


图 21-42 中断源和中断事件

## 21.9.2 中断寄存器

### 21.9.2.1 中断状态寄存器

寄存器 ISL/H 保存各中断请求位。ISL/H 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 ISSL/H（置位对应位）或寄存器 ISRL/H（清除对应位）可软件置位或清除寄存器 ISL/H 中的各位。

中断的产生和寄存器 ISL/H 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 ISL/H 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 ISL/H 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）。*

#### ISL

中断状态寄存器低位

(9C<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
<b>T12PM</b>	<b>T12OM</b>	<b>ICC62F</b>	<b>ICC62R</b>	<b>ICC61F</b>	<b>ICC61R</b>	<b>ICC60F</b>	<b>ICC60R</b>
rh	rh	rh	rh	rh	rh	rh	rh

符号	位	类型	描述
<b>ICC60R,</b> <b>ICC61R,</b> <b>ICC62R</b>	0, 2, 4	rh	<p><b>捕获，比较匹配上升沿标志</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>

## 捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
ICC60F, ICC61F, ICC62F	1, 3, 5	rh	<b>捕获, 比较匹配下降沿标志位</b> 该位指示已经检测到 CC6x_F 事件。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 CC6xIN 发生下降沿跳变。则通过该位指示已经检测到 CC6x_F 事件。 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
T12OM	6	rh	<b>定时器 T12 1- 匹配标志</b> 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。 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
T12PM	7	rh	<b>定时器 T12 周期 - 匹配标志</b> 该位指示递增计数时 (T12_PM 且 CDIR = 0), 检测到 T12 周期匹配事件。 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

## ISH

中断状态寄存器高位

(9D<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
STR	IDLE	WHE	CHE	TRPS	TRPF	T13PM	T13CM
rh	rh	rh	rh	rh	rh	rh	rh

符号	位	类型	描述
T13CM	0	rh	<b>定时器 T13 比较匹配标志</b> 该位指示已经检测到定时器 T13 比较匹配 (CM_63)。 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
T13PM	1	rh	<b>定时器 T13 周期匹配标志</b> 该位指示已经检测到定时器 T13 周期匹配 (T13_PM)。 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

## 捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
TRPF	2	rh	<b>强制中断标志</b> 该位指示检测 / 已经检测到强制中断情况（输入 $\overline{\text{CTRAP}} = 0$ 或由软件设定）。如果 $\text{TRM2} = 0$ ，输入 $\overline{\text{CTRAP}} = 1$ 或 $\text{TRPPEN} = 0$ 时，硬件自动清除 TRPF。 如果 $\text{TRM2} = 1$ ，必须由软件设置 $\text{RTRPF} = 1$ 以清除该标志。 $0_{\text{B}}$ 还未检测到强制中断情况 $1_{\text{B}}$ 检测 / 已经检测到强制中断情况
TRPS	3	rh	<b>强制中断状态<sup>1)</sup></b> 该位指示实际强制中断状态。如果 $\text{TRPF} = 1$ 则该位置位；根据寄存器 $\text{TRPCTR}$ 中选定的模式清除该位。 $0_{\text{B}}$ 强制中断状态无效 $1_{\text{B}}$ 强制中断状态有效
CHE	4	rh	<b>正确霍尔事件标志</b> 该位指示已经检测到正确霍尔事件（ $\text{CM\_CHE}$ ） $0_{\text{B}}$ 还未检测到上述事件 $1_{\text{B}}$ 已检测到上述事件
WHE	5	rh	<b>错误霍尔事件标志</b> 该位指示已经检测到错误霍尔事件（ $\text{CM\_WHE}$ ） $0_{\text{B}}$ 还未检测到上述事件 $1_{\text{B}}$ 已检测到上述事件
IDLE	6	rh	<b>空闲状态标志位</b> 若空闲状态被使能（ $\text{ENIDLE} = 1$ ），IDLE 位和位 WHE 同时置位；该位必须由软件清除。 $0_{\text{B}}$ 无操作 $1_{\text{B}}$ 位 MCMP 被清除，选中的输出被设置为被动态
STR	7	rh	<b>多通道模式映射传送请求</b> 该位指示已经发生了从 MCMP5 到 MCMP 的映射传送（ $\text{MCM\_ST}$ ）。 $0_{\text{B}}$ 还未检测到上述事件 $1_{\text{B}}$ 已检测到上述事件

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  $\text{PSLR}$  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  $\text{TRPS} = 1$  和  $\text{TRPF} = 0$  的情况。

## 捕获 / 比较单元 6 (CCU6)

### 21.9.2.2 中断状态置位寄存器

通过寄存器 ISSL/H 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 ISSL/H 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。

#### ISSL

中断状态置位寄存器低位

(A4<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
ST12PM	ST12OM	SCC62F	SCC62R	SCC61F	SCC61R	SCC60F	SCC60R
W	W	W	W	W	W	W	W

符号	位	类型	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 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 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 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T12PM 将被置位

#### ISSH

中断状态置位寄存器高位

(A5<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
SSTR	SIDLE	SWHE	SCHE	SWHC	STRPF	ST13PM	ST13CM
W	W	W	W	W	W	W	W

**捕获 / 比较单元 6 (CCU6)**

符号	位	类型	描述
<b>ST13CM</b>	0	w	置位定时器 <b>T13</b> 比较匹配标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T13CM 将被置位
<b>ST13PM</b>	1	w	置位定时器 <b>T13</b> 周期匹配标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T13PM 将被置位
<b>STRPF</b>	2	w	置位强制中断标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 TRPF 和 TRPS 将被置位
<b>SWHC</b>	3	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>	4	w	置位正确霍尔事件标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 CHE 将被置位
<b>SWHE</b>	5	w	置位错误霍尔事件标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 WHE 将被置位
<b>SIDLE</b>	6	w	置位 <b>IDLE</b> 标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 IDLE 将被置位
<b>SSSTR</b>	7	w	置位 <b>STR</b> 标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 STR 将被置位



## 捕获 / 比较单元 6 (CCU6)

### 21.9.2.3 状态复位寄存器

通过寄存器 ISRL/H 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。

#### ISRL

中断状态复位寄存器低位

(A4<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
RT12PM	RT12OM	RCC62F	RCC62R	RCC61F	RCC61R	RCC60F	RCC60R
W	W	W	W	W	W	W	W

符号	位	类型	描述
RCC60R, RCC61R, RCC62R	0, 2, 4	W	复位捕获, 比较匹配上升沿标志位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 CC6xR 将被复位
RCC60F, RCC61F, RCC62F	1, 3, 5	W	复位捕获, 比较匹配下降沿标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 CC6xF 将被复位
RT12OM	6	W	复位定时器 T12 1- 匹配标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T12OM 将被复位
RT12PM	7	W	复位定时器 T12 周期匹配标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T12PM 将被复位

#### ISRH

中断状态寄存器高位

(A5<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
RSTR	RIDLE	RWHE	RCHE	0	RTRPF	RT13PM	RT13CM
W	W	W	W	r	W	W	W

## 捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
<b>RT13CM</b>	0	w	复位定时器 T13 比较匹配标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T13CM 将被复位
<b>RT13PM</b>	1	w	复位定时器 T13 周期匹配标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 T13PM 将被复位
<b>RTRPF</b>	2	w	复位强制中断标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 TRPF 将被复位 (当输入 $\overline{\text{CTRAP}} = 0$ 且 TRPPEN = 1 时, 则不考虑。)
<b>RCHE</b>	4	w	复位正确霍尔事件标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 CHE 将被复位
<b>RWHE</b>	5	w	复位错误霍尔事件标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 WHE 将被复位
<b>RIDLE</b>	6	w	复位 IDLE 标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 IDLE 将被复位
<b>RSTR</b>	7	w	复位 STR 标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 STR 将被复位
<b>0</b>	3	r	保留; 读操作返回 0; 应写入 0。

## 捕获 / 比较单元 6 (CCU6)

## 21.9.2.4 中断使能寄存器

寄存器 IENL/H 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。

IENL

中断使能寄存器低位

(9C<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
ENT12PM	ENT12OM	ENCC62F	ENCC62R	ENCC61F	ENCC61R	ENCC60F	ENCC60R
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
ENCC60R, ENCC61R, ENCC62R	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。
ENCC60F, ENCC61F, ENCC62F	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。
ENT12OM	6	rw	<b>定时器 T12 1- 匹配中断使能</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。
ENT12PM	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。

## 捕获 / 比较单元 6 (CCU6)

**IENH**

中断使能寄存器高位

(9D<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
<b>ENSTR</b>	<b>ENIDLE</b>	<b>ENWHE</b>	<b>ENCHE</b>	<b>0</b>	<b>ENTRPF</b>	<b>ENT13PM</b>	<b>ENT13CM</b>
rw	rw	rw	rw	r	rw	rw	rw

符号	位	类型	描述
<b>ENT13CM</b>	0	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>	1	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。 1 <sub>B</sub> 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>	2	rw	<b>强制中断标志中断使能</b> 0 <sub>B</sub> 如果出现寄存器 IS 中的位 TRPF 置位情况, 不产生中断。 1 <sub>B</sub> 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>	4	rw	<b>正确霍尔事件中断使能</b> 0 <sub>B</sub> 如果出现寄存器 IS 中的位 CHE 置位情况, 不产生中断。 1 <sub>B</sub> 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。

## 捕获 / 比较单元 6 (CCU6)

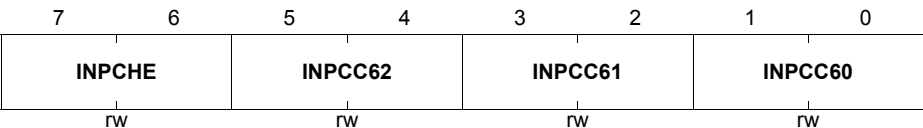
符号	位	类型	描述
<b>ENWHE</b>	5	rw	<b>错误霍尔事件中断使能</b> <b>0<sub>B</sub></b> 如果出现寄存器 IS 中的位 WHE 置位情况，不产生中断。 <b>1<sub>B</sub></b> 如果出现寄存器 IS 中的位 WHE 置位情况，则产生中断请求。位域 INPERR 所选择的服务请求输出被激活。
<b>ENIDLE</b>	6	rw	<b>使能 IDLE</b> 由该位使能检测到错误霍尔事件之后（位 WHE 被置位），自动进入到空闲状态（位 IDLE 将被置位）。空闲状态期间，位域 MCMP 被自动清除。 <b>0<sub>B</sub></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>	7	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，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>	3	r	<b>保留；</b> 读操作返回 0；应写入 0。

捕获 / 比较单元 6 (CCU6)

21.9.2.5 中断节点指针寄存器

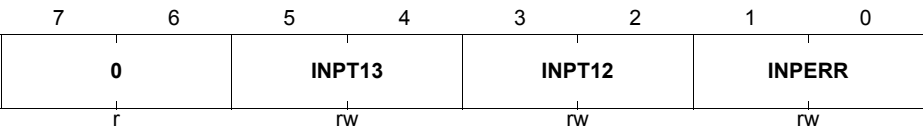
寄存器 INPL/H 保存中断节点指针，允许用户进行灵活中断处理。如果发生相应中断事件且使能该事件产生中断，由这些位域定义哪个服务请求输出将被激活。

**INPL**  
**中断节点指针寄存器低位** (9E<sub>H</sub>) **复位值: 40<sub>H</sub>**  
**RMAP: 0, PAGE: 2**



符号	位	类型	描述
INPCC60, INPCC61, INPCC62	[1:0], [3:2], [5:4]	rw	<b>通道 CC6x 中断的中断节点指针</b> 该位域定义由中断请求源 CC6xR (如果由位 ENCC6xR 使能) 或 CC6xF (如果由位 ENCC6xF 使能) 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
INPCHE	[7:6]	rw	<b>CHE 中断的中断节点指针</b> 该位域定义位 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。

**INPH**  
**中断节点指针寄存器高位** (9F<sub>H</sub>) **复位值: 39<sub>H</sub>**  
**RMAP: 0, PAGE: 2**



## 捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
<b>INPERR</b>	[1:0]	rw	<b>错误中断的中断节点指针</b> 该位域定义位 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>	[3:2]	rw	<b>定时器 T12 中断的中断节点指针</b> 该位域定义位 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>	[5:4]	rw	<b>定时器 T13 中断的中断节点指针</b> 该位域定义位 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>	[7:6]	r	<b>保留；</b> 读操作返回 0；应写入 0。

## 21.10 一般模块操作

本节提供下述信息：

- 输入选择 (见 [章节 21.10.1](#))
- 通用寄存器描述 (见 [章节 21.10.2](#))

### 21.10.1 输入选择

通过编程设置端口输入选择寄存器 **PISEL0L**, **PISEL0H** 和 **PISEL2**。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。



捕获 / 比较单元 6 (CCU6)

21.10.2 通用寄存器

21.10.2.1 端口输入寄存器

寄存器 PISEL0L 和 PISEL0H 中的位域为模块输入选择实际的输入信号。

**PISEL0L**  
 端口输入选择寄存器低位 (9E<sub>H</sub>) 复位值: 00<sub>H</sub>  
 RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
ISTRP		ISCC62		ISCC61		ISCC60	
rw		rw		rw		rw	

符号	位	类型	描述
ISCC60	[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> 输入信号 CC60_0 01 <sub>B</sub> 输入信号 CC60_1 10 <sub>B</sub> CCU6 SR2 事件 11 <sub>B</sub> ADC 通道 0 边界极限检查事件
ISCC61	[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> 输入信号 CC61_0 01 <sub>B</sub> 输入信号 CC61_1 10 <sub>B</sub> CCU6 SR2 事件 11 <sub>B</sub> ADC 通道 1 边界极限检查事件
ISCC62	[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> 输入信号 CC62_0 01 <sub>B</sub> 输入信号 CC62_1 10 <sub>B</sub> CCU6 SR2 事件 11 <sub>B</sub> ADC 通道 2 边界极限检查事件

## 捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
<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。 MODPISEL3.CPTRAPIS 同时用于选择触发 CTRAP 的 8 个触发源之一。 00 <sub>B</sub> 选择输入引脚 <u>CTRAP_0</u> , <u>CTRAP_1</u> , <u>CTRAP_2</u> 和 <u>CTRAP_3</u> 。寄存器 MODPISEL3 中的位 CTAPIS 用于进一步从这四个输入引脚中选择一个。 01 <sub>B</sub> 选择 P1 过流检测输出 10 <sub>B</sub> 上述选项 00 或 01 中的任意输入源被选中, 用来触发 CTRAP 11 <sub>B</sub> 选择 ADC 通道事件

## PISEL0H

端口输入选择寄存器 0 高位

(9F<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
ISTR12HR	ISPOS2	ISPOS1	ISPOS0				
rw	rw	rw	rw				

符号	位	类型	描述
<b>ISPOS0</b>	[1:0]	rw	<b>CCPOS0 的输入选择</b> 该位域用于选择 CCPOS0 的输入信号。 00 <sub>B</sub> 选择输入 CCPOS0_0 01 <sub>B</sub> 选择输入 CCPOS0_1 10 <sub>B</sub> 选择输入 CCPOS0_2 11 <sub>B</sub> ADC 通道 0 边界极限检查事件
<b>ISPOS1</b>	[3:2]	rw	<b>CCPOS1 的输入选择</b> 该位域用于选择 CCPOS1 的输入信号 00 <sub>B</sub> 选择输入 CCPOS1_0 01 <sub>B</sub> 选择输入 CCPOS1_1 10 <sub>B</sub> 未使用 11 <sub>B</sub> ADC 通道 1 边界极限检查事件

捕获 / 比较单元 6 (CCU6)

符号	位	类型	描述
<b>ISPOS2</b>	[5:4]	rw	<b>CCPOS2 的输入选择</b> 该位域用来选择 CCPOS2 输入信号。 00 <sub>B</sub> 输入信号 CCPOS2_0 01 <sub>B</sub> 输入信号 CCPOS2_1 10 <sub>B</sub> 未使用 11 <sub>B</sub> ADC 通道 2 边界极限检查事件
<b>IST12HR</b>	[7:6]	rw	<b>T12HR 的输入选择</b> 该位域用来选择 T12HR 的输入信号。 00 <sub>B</sub> 来自 T12HR[7:0] <sup>1)</sup> 的任意输入 01 <sub>B</sub> CCU6 SR2 输出 10 <sub>B</sub> CCU6 SR3 输出 11 <sub>B</sub> ADC 通道事件

1) SCU 模块中的 MODIPISEL3 寄存器中的 IST12HR1 用于在 T12HR[7:0] 中选择一个。

**PISEL2**

端口输入选择寄存器 2

(A4<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 3

7	6	5	4	3	2	1	0
0						IST13HR	
r						rw	

符号	位	类型	描述
<b>IST13HR</b>	[1:0]	rw	<b>T13HR 的输入选择</b> 该位域用来选择 T13HR 输入的信号。 00 <sub>B</sub> 来自 T13HR[7:0] <sup>1)</sup> 的输入信号 01 <sub>B</sub> CCU6 SR2 输出 10 <sub>B</sub> CCU6 SR3 输出 11 <sub>B</sub> ADC 通道事件
<b>0</b>	[7:2]	r	<b>保留；</b> 读操作返回 0；应写入 0。

1) SCU 模块中的 MODIPISEL3 寄存器中的 IST13HR1 用于在 T13HR[7:0] 中选择一个。

捕获 / 比较单元 6 (CCU6)

### 21.11 寄存器映射

CCU6 SFR 位于标准存储器区 (RMAP = 0)，由 4 页组成。CCU6\_PAGE 寄存器位于地址 A3<sub>H</sub> 处，包含分页值和页控制信息。

#### CCU6\_PAGE

CCU6 分页寄存器

(F1<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: X

7	6	5	4	3	2	1	0
OP		STNR		0	PAGE		
w		w		r	rwh		

符号	位	类型	描述
PAGE	[3:0]	rwh	<b>分页位</b> 写入时，该值表示新页的地址。读出时，该值表示当前有效页的值 =addr[y:x+1]
STNR	[5:4]	w	<b>保存编号</b> 该编号指明在哪个保存位域上执行 OP 定义的操作。 若 OP = 10 <sub>B</sub> , PAGE 的内容在被新值覆盖之前保存在 CCU6_STx 中。 若 OP = 11 <sub>B</sub> , PAGE 的内容被 CCU6_STx 覆盖。写入 PAGE 的值不予理睬。 00 选择 CCU6_ST0 01 选择 CCU6_ST1 10 选择 CCU6_ST2 11 选择 CCU6_ST3
OP	[7:6]	w	<b>操作</b> 0X 手动保存页模式，STNR 的值被忽略，PAGE 被直接写入 10 带有自动页保存的新页设置。当前写入 PAGE 中的内容被保存的同时，上次写入 PAGE 的内容被保存在 STNR 规定的位域 CCU6_STx 中。 11 自动恢复页。对写入 PAGE 的内容不予理睬，PAGE 由 STNR 规定的位域 CCU6_STx 中的内容覆盖

内核 SFR 的地址（非映射）列表见表 21-16。

**捕获 / 比较单元 6 (CCU6)**

本章中描述的所有 CCU6 寄存器名称，在本手册其它章节中引用时需加上模块名前缀“CCU6\_”，例如 CCU6\_CC63SRL。

**表 21-16 CCU6 页 0 – 3 的 SFR 地址列表**

地址	页 0	页 1	页 2	页 3
9AH	<b>CC63SRL</b>	<b>CC63RL</b>	<b>T12MSELL</b>	<b>MCMOUTL</b>
9BH	<b>CC63SRH</b>	<b>CC63RH</b>	<b>T12MSELH</b>	<b>MCMOUTH</b>
9CH	<b>TCTR4L</b>	<b>T12PRL</b>	<b>IENL</b>	<b>ISL</b>
9DH	<b>TCTR4H</b>	<b>T12PRH</b>	<b>IENH</b>	<b>ISH</b>
9EH	<b>MCMOUTSL</b>	<b>T13PRL</b>	<b>INPL</b>	<b>PISEL0L</b>
9FH	<b>MCMOUTSH</b>	<b>T13PRH</b>	<b>INPH</b>	<b>PISEL0H</b>
A4H	<b>ISRL</b>	<b>T12DTCL</b>	<b>ISSL</b>	<b>PISEL2</b>
A5H	<b>ISRH</b>	<b>T12DTCH</b>	<b>ISSH</b>	
A6H	<b>CMPMODIFL</b>	<b>TCTR0L</b>	<b>PSLR</b>	
A7H	<b>CMPMODIFH</b>	<b>TCTR0H</b>	<b>MCMCTR</b>	
FAH	CC60SRL	CC60RL	<b>TCTR2L</b>	<b>T12L</b>
FBH	CC60SRH	CC60RH	<b>TCTR2H</b>	<b>T12H</b>
FCH	CC61SRL	CC61RL	<b>MODCTRL</b>	<b>T13L</b>
FDH	CC61SRH	CC61RH	<b>MODCTRH</b>	<b>T13H</b>
FEH	CC62SRL	CC62RL	<b>TRPCTRL</b>	<b>CMPSTATL</b>
FFH	CC62SRH	CC62RH	<b>TRPCTRH</b>	<b>CMPSTATH</b>

## 22 模数转换器

XC83x 的模数转换器模块（ADC）采用逐次逼近技术将模拟输入值（电压）转换成离散数字值。

ADC 模块具有一个输入通道个数可选的 ADC 内核（ADC0）。

输入通道可选且仲裁方式灵活。

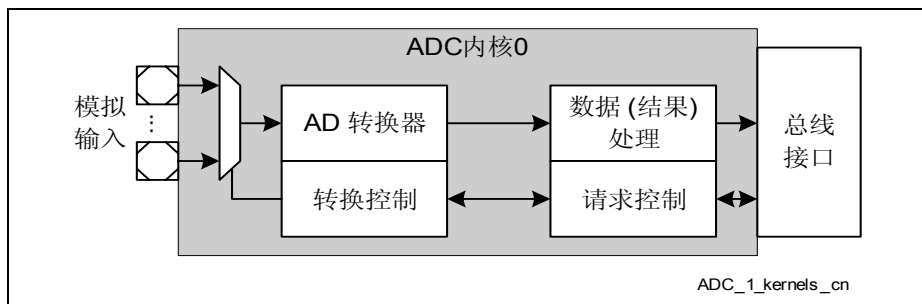


图 22-1 ADC 模块框图

本节给出 ADC 模块特性概述，在后面的章节中按照下述方式介绍 ADC 的一般结构：

- “简介和基本结构” 页 22-7
- “一般功能配置” 页 22-13
- “转换请求产生” 页 22-16
- “请求源仲裁” 页 22-36
- “模拟输入通道配置” 页 22-41
- “转换结果处理” 页 22-60
- “中断请求处理” 页 22-79
- “寄存器映射” 页 22-86

下述特性给出 ADC 内核的功能：

- 输入电压范围：0 V - 模拟电源电压 ( $V_{DDP} = 3.0 \text{ V}$  至  $5.5 \text{ V}$ )
- 每个通道都具有三个可选的内部参考电压源，支持比例式测量和不同的信号范围：
  - 内部  $V_{DDP}$  和  $V_{SSP}$
  - 内部  $1.2V_{ref}$  和  $CH0$ ，用作 ADC 电压参考地<sup>1)</sup>
  - 内部  $1.2V_{ref}$  和  $V_{SSP}$ <sup>1)</sup>
- 多达 8 个模拟输入通道
- 转换速度和采样时间可调整，以适应于各种传感器和参考
- 转换时间低至  $1 \mu\text{s}$ （取决于结果宽度和采样时间）
- 灵活的请求源选择和仲裁
  - 单通道转换（单次或重复）
  - 可配置的自动扫描转换（单次或重复）
  - 可编程的任意转换序列（单次或重复）
  - 可由软件、定时器事件或者外部事件触发转换操作
  - 等待 - 启动模式，实现最大的吞吐量，或
  - 取消 - 插入 - 重复模式，用来减少转换延迟
- 功能强大的结果处理
  - 结果宽度可选：8 位和 10 位
  - 4 个独立的结果寄存器
  - 可配置的极限检查，边界值可编程设定
  - 通过将可选个数的转换结果进行累加的方式减少数据率
  - 通过平均转换结果实现的一阶数字低通滤波器
- 基于可选事件的灵活中断产生
- 支持各种省电模式
- 附加特性
  - 超出范围（ORC）电压比较器，检测每个可触发其它模块的输入通道的电压
  - 可配置的极限检查器，能够触发其它模块

1) 使用该模式，两次转换之间需要建立时间。具体参数请参见数据手册。

## 22.1 系统信息

本节提供与 ADC 相关的系统信息。

### 22.1.1 引脚

XC83x 中的 ADC 模块的引脚分配见表 22-1。

表 22-1 ADC 引脚功能和选择

引脚	功能	描述	由 ... 选择
P2.0	CH0	模拟输入通道 0	P2_EN.P0 = 1 <sub>B</sub>
P2.1	CH1	模拟输入通道 1	P2_EN.P1 = 1 <sub>B</sub>
P2.2	CH2	模拟输入通道 2	P2_EN.P2 = 1 <sub>B</sub>
P2.3	CH3	模拟输入通道 3	P2_EN.P3 = 1 <sub>B</sub>
P2.4	CH4	模拟输入通道 4	P2_EN.P4 = 1 <sub>B</sub>
P2.5	CH5	模拟输入通道 5	P2_EN.P5 = 1 <sub>B</sub>
P2.6	CH6	模拟输入通道 6	P2_EN.P6 = 1 <sub>B</sub>
P2.7	CH7	模拟输入通道 7	P2_EN.P7 = 1 <sub>B</sub>

### 22.1.2 时钟配置

ADC 内核的工作频率 FPCLK 固定为 48 MHz。见章节 22.5.1。

如果完全不需要 ADC 功能，可通过门控关闭其时钟输入的方式整个的禁止该模块，从而最大程度的降低功耗。通过置位下述的 PMCON1 寄存器中的 ADC\_DIS 来实现。

必须在访问 PMCON1 寄存器之前编程设置 SCU\_PAGE 寄存器中的位域 PAGE。

#### PMCON1

外设管理控制寄存器 1

(EF<sub>H</sub>)

复位值: FF<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
IIC_DIS	LTS_DIS	CDC_DIS	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
ADC_DIS	0	rw	ADC 禁止请求，高有效
			0 ADC 正常工作
			1 请求禁止 ADC（缺省设置）



### 22.1.3 中断事件和分配

**表 22-2** 列出 ADC 的中断事件源以及相应的中断使能位和标志位。

**表 22-2 ADC 中断事件**

事件	事件中断使能位	事件标志位	服务请求输出
请求源事件中断	ADC_Q0R0.ENSEI	ADC_EVINFR.EVINFx	SR0
	ADC_QBUR0.ENSEI		
通道事件中断	ADC_GLOBCTR.CLCIEN	ADC_CHINFR.CHINFx	SR0
结果事件中断	ADC_RCRx (x = 0 - 3).IEN	ADC_EVINFR.EVINFx	SR0
超出范围比较器事件中断	ADC_GLOBCTR.ORCIEN	ADC_LORE.LOREx	SR1

**表 22-3** 给出每个 ADC 中断源的中断节点分配。

**表 22-3 ADC 事件的中断节点控制**

事件	中断节点使能位	中断节点标志位	向量地址
SR0	IEN1.EADC	IRCON1.ADCSR0	33 <sub>H</sub>
SR1		IRCON1.ADCSR1	

### 22.1.4 IP 互连

ADC 可与其它外设互连，无需软件干预即可实现高水平的自动化。**表 22-4** 描述 ADC 输出、通道事件和超出范围比较器事件与 CCU6 和定时器 T2 输入的连接关系。**表 22-5** 描述 CCU6 和 LEDTSCU 输出与 ADC 外部请求触发输入的连接关系。

**表 22-4 ADC 输出连接关系**

ADC 功能 / 信号	连接至其它模块的输入	由 ... 选择
<b>通道事件</b>		
ADC 通道事件 (o): ADC_CHEV	CCU6 输入 (i): T12HR	CCU6_PISEL0H.IST12HR = 11 <sub>B</sub>
	CCU6 输入 (i): T13HR	CCU6_PISEL2.IST13HR = 11 <sub>B</sub>
ADC 通道事件 0 (o): ADC_CHEV0	CCU6 输入 (i): CTRAP	CCU6_PISEL0L.ISTRP = 11 <sub>B</sub>
ADC 通道事件 1(o): ADC_CHEV1	未连接	-
ADC 通道事件 2 (o): ADC_CHEV2	未连接	-

**表 22-4 ADC 输出连接关系**

ADC 功能 / 信号	连接至其它模块的输入	由 ... 选择
ADC 边界事件 0 (o): ADC_BF0	CCU6 输入 (i): CC60	CCU6_PISEL0L.ISCC60 = 11 <sub>B</sub>
	CCU6 输入 (i): CCPOS0	CCU6_PISEL0H.ISPOS0 = 11 <sub>B</sub>
ADC 边界事件 1 (o): ADC_BF1	CCU6 输入 (i): CC61	CCU6_PISEL0L.ISCC61 = 11 <sub>B</sub>
	CCU6 输入 (i): CCPOS1	CCU6_PISEL0H.ISPOS1 = 11 <sub>B</sub>
ADC 边界事件 2 (o): ADC_BF2	CCU6 输入 (i): CC62	CCU6_PISEL0L.ISCC62= 11 <sub>B</sub>
	CCU6 输入 (i): CCPOS2	CCU6_PISEL0H.ISPOS2= 11 <sub>B</sub>

**超出范围比较器 (ORC)**

ORC 事件 0 (o): ORCEVENT0	CCU6 输入 (i): T12HR	MODPISEL3.IST12HR1 = 001 <sub>B</sub> CCU6_PISEL0H.IST12HR = 00 <sub>B</sub>
	定时器 T2 输入 (i): T2EX	MODPISEL2.T2EXIS = 100 <sub>B</sub>
ORC 事件 1 (o): ORCEVENT1	定时器 T2 输入 (i): T2EX	MODPISEL2.T2EXIS = 101 <sub>B</sub>
ORC 事件 2 (o): ORCEVENT2	CCU6 输入 (i): T12HR	MODPISEL3.IST12HR1 = 100 <sub>B</sub> CCU6_PISEL0H.IST12HR = 00 <sub>B</sub>
	CCU6 输入 (i): T13HR	MODPISEL3.IST13HR1 = 100 <sub>B</sub> CCU6_PISEL2.IST13HR = 00 <sub>B</sub>
ORC 事件 3 (o): ORCEVENT3	CCU6 输入 (i): CTRAP	MODPISEL3.CTRAPIS = 11 <sub>B</sub> CCU6_PISEL0L.ISTRP = 00/10 <sub>B</sub>
ORC 事件 4 (o): ORCEVENT4	CCU6 输入 (i): T12HR	MODPISEL3.IST12HR1 = 110 <sub>B</sub> CCU6_PISEL0H.IST12HR = 00 <sub>B</sub>
	CCU6 输入 (i): T13HR	MODPISEL3.IST13HR1 = 110 <sub>B</sub> CCU6_PISEL2.IST13HR = 00 <sub>B</sub>

**表 22-5 ADC 输入连接关系**

ADC 功能 / 信号	连接至其它模块功能 / 信号
<b>请求源 x (x = 0,1)</b>	
请求源 x 触发 0 输入 (i): REQTRxA	CCU6 服务请求输出 SR2 (o):CCU6_SR2
请求源 x 触发 1 输入 (i): REQTRxB	CCU6 服务请求输出 SR3 (o): CCU6_SR3
请求源 x 触发 2 输入 (i): REQTRxC	CCU6 T12 周期 - 匹配 (o): T12PM
请求源 x 触发 3 输入 (i): REQTRxD	CCU6 T13 周期 - 匹配 (o): T13PM
请求源 x 触发 4 输入 (i): REQTRxE	CCU6 T13 比较 - 匹配 (o): T13CM
请求源 x 触发 5 输入 (i): REQTRxF	CCU6 MCM 映射传送 (o): MCM_ST

表 22-5      ADC 输入连接关系

ADC 功能 / 信号	连接至其它模块功能 / 信号
请求源 x 触发 6 输入 (i): REQTRxG	LEDTSKU 比较 - 匹配 (o): LEDTS_CM
请求源 x 触发 7 输入 (i): REQTRxH	LEDTSKU 时间片断中断 (o): LEDTS_TSI

## 22.2 简介和基本结构

ADC 具有一组可根据给定应用的需要进行配置的功能单元。由这些单元构成从输入信号到数字结果之间的通路。

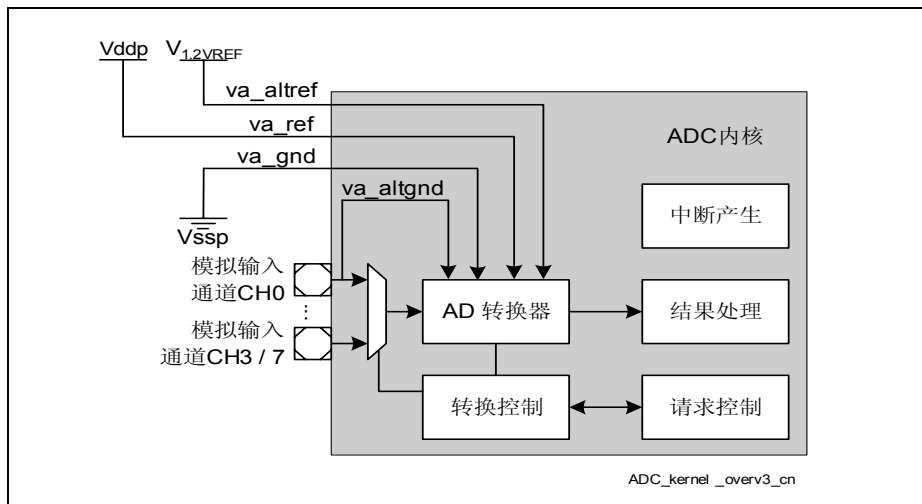


图 22-2 ADC 内核框图

### 请求源控制

可由两个请求源产生并行转换请求：

- 线性序列请求源请求按照可配置的顺序自动扫描多达 8 个通道
- 任意序列源请求对多达 4 个可任意选择的通道排成队列

每个请求源可（单次或重复）请求所选择的转换序列，每个请求源可单独使能或可由外部事件触发，如 PWM 边沿、定时器信号或引脚跳变。

仲裁器可从不同请求源中决断出并行转换请求。具有较高优先级的请求或者取消正在运行的低优先级请求（取消 - 插入 - 重复模式）或者在当前运行的转换结束后立即进行转换（等待 - 启动模式）。如果目标结果寄存器还未被读取，转换可被推迟（等待 - 读取模式）。

### 输入通道选择

模拟输入复用器从多达 8 个（CH0 - CH7）模拟输入中选择一个进行转换。两个请求源可选择线性序列或任意序列。这些请求源的优先级可配置。

**注：** 由于引脚的限制，并非所有模拟输入通道在所有封装的器件中都可用。详情请参考 [章节 22.1](#) 中的关于具体实现的描述。

## 转换控制

转换参数（如采样阶段的持续时间）可在输入级中配置。

因此输入通道可根据与 ADC 相连的传感器类型（或其它模拟源）进行调整。

## 模拟 / 数字转换器

将选中的输入通道转换成数字值，先对选中的输入通道上的电压进行采样，然后产生选定位数的结果位。

## 结果处理

每个模拟输入通道的转换结果可直接送至 4 个结果寄存器中的一个。可由一组通道或单个通道使用一个结果寄存器。

等待 - 读取模式通过阻止新的转换，直到先前的结果被读出为止的方式，避免了因结果被覆盖而带来的数据损失。

数据压缩（例如：用于数字抗混叠滤波）自动将两次转换结果相加，之后才向 CPU 发送中断请求。

而且，结果寄存器可被级联，组成 FIFO 结构，用来保存数个转换结果而不会覆盖先前的数据。从而可允许 CPU 延迟读取 ADC 中的结果数据。

ADC 中实现了一个数字一阶低通滤波器，在将结果写入到结果寄存器之前，该滤波器连续的对转换结果进行滤波。

## 中断产生

下述 ADC 事件可向 CPU 发出中断请求：

- **请求源事件**指示对应请求源的转换序列完成。该事件可用于触发新序列的建立。
- **通道事件**指示某个通道完成转换。该事件可和极限检查相结合，仅当转换结果在定义的范围时，才产生中断。
- **结果事件**指示相应结果寄存器中有新的结果数据可用。如果数据压缩或数字低通滤波器模式有效，仅在一个完整的累加过程之后才产生结果事件。
- **超出范围比较器事件**指示 ADC 输入通道上检测到的电压高于或低于  $V_{ddp}$ 。

所有中断请求都被分配给两个中断节点。

## 附加特性

支持下列两种检测和触发机制，确保快速响应而无需 CPU 干预。

**超出范围比较器（ORC）**内嵌到每个 ADC 通道中，当电压超出范围时，将触发其它模块或中断。当输入通道上的电压升高至高于  $V_{ddp}$  电平或者当输入通道的电压降低至低于  $V_{ddp}$  电平时，则出现输入超出范围的情况。

超出范围比较器与其它模块（如 CCU6，定时器）相连，能够触发其它模块的启动或停止。所有超出范围比较器事件被分配给一个中断节点。

---

模数转换器

**可配置极限检查器**将转换结果与编程设置的边界值比较，当满足触发条件时，触发其它模块或中断。

可配置的极限检查器与其它模块（如 CCU6）相连，从而可用于触发其它模块的启动或停止。

## 22.3 电气模型

对模拟输入电压的转换将由两个连续的阶段组成：

- 采样阶段：输入电压被采样并保存  
输入信号通路为该阶段的简化模型。
- 转换阶段：保存的电压值被转换成数字结果  
参考电压通路为该阶段的简化模型。

### 输入信号通路

XC83x 的 ADC 模块使用开关电容  $C_{AIN}$  进行采样（小的寄生电容用每个输入引脚上的小电容表示）。采样过程中，选中的输入  $CHx$  通过输入复用器（用理想开关和串联电阻  $R_{AIN}$  的模型表示）与电容  $C_{AIN}$  相连。

在采样阶段， $CHx$  的开关闭合，从而输入电压  $V_{AINx}$  连接到电容  $C_{AIN}$  上。

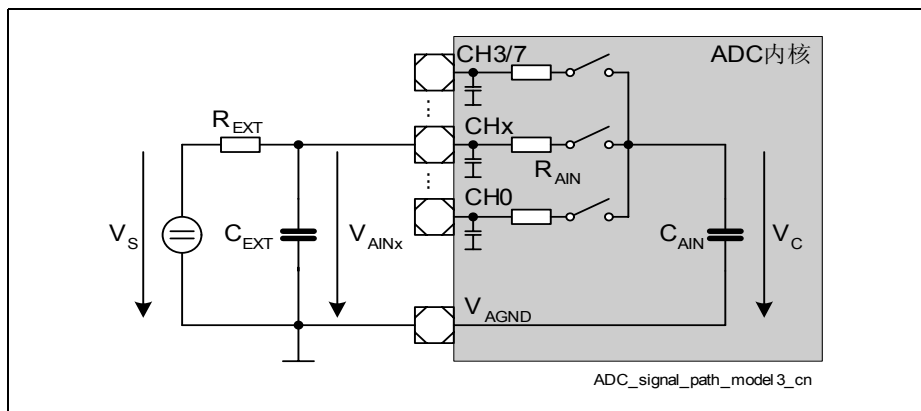


图 22-3 信号通路模型

**图 22-3** 给出模拟输入信号通路的简化模型。带有内部阻抗  $R_{EXT}$  的模拟电压源（电压值为  $V_S$ ）用来代表要转换的模拟输入。

在采样阶段，对应的开关闭合，向电容  $C_{AIN}$  充电。由于 RC 网络的低通效应，实际要转换的电压  $V_C$  不会立即跟随  $V_S$ 。采样阶段的长度主要取决于模拟电压源  $R_{EXT}$  的值和需要的转换精度。

为了减少  $R_{EXT}$  的影响并滤除输入噪声，推荐用户在每个 ADC 的模拟输入引脚上引入一个快速外部隔直流电容  $C_{EXT}$ 。因此主要由  $C_{EXT}$  提供采样阶段的电荷。与不加隔直流电容相比，该结构明显缩短了采样阶段的时间，因为用来定义采样时间的低通时常数主要由  $R_{AIN}$  和  $C_{AIN}$  的值决定。

另外，电容  $C_{AIN}$  自动预先充电至大约标准参考电压  $V_{AREF}$  的一半，以使得采样开始时， $V_{AINx}$  和  $V_C$  之间的平均电压差最小。由于参数的变化和寄生效应的影响， $C_{AIN}$  上的预先充电的电压通常小于  $V_{AREF} / 2$ 。

另一方面， $C_{EXT}$  和  $C_{AIN}$  之间电荷的重新分配导致采样过程中  $V_{AINx}$  的电压发生改变。为

## 模数转换器

了使得该电压变化低于  $1 \text{ LSB}_n$ ，推荐使用的外部隔直流电容  $C_{\text{EXT}}$  至少在  $2^n \times C_{\text{AIN}}$  的范围内。

对于同一个模拟输入通道来讲，在两次采样之间， $R_{\text{EXT}}$  和  $C_{\text{EXT}}$  组成的低通滤波器应当允许  $V_{\text{AINx}}$  跟随  $V_S$ 。

请注意：尤其对于高温应用来讲，ADC 的模拟输入结构可能导致漏电流并由于  $R_{\text{EXT}}$  上的压降而引入误差。如果输入电压电平接近模拟电源地  $V_{\text{SS}}$  或模拟电源  $V_{\text{DDP}}$ ，ADC 输入漏电流增加。推荐使用的工作范围是输入电压大约为  $V_{\text{DDP}}$  的 3% 和 97% 之间，以减少相应 ADC 通道的输入漏电流。

另外，漏电流还受到相邻模拟输入的过载情况的影响。过载期间，输入上施加的电压超过电源电压，此时内嵌的保护电路限制输入电压。这使得过载电流通过保护电路传递（通过一个耦合因子）至相邻输入上。



## 22.4 转换特性和误差定义

ADC 的转换特性描述模拟输入电压和  $2^n$  离散数字结果值 ( $n$  位精度) 之间的关系。每个数字结果值 (范围是 0 到  $2^n-1$  之间) 代表的输入电压范围由参考电压范围除以  $2^n$  来定义。该范围 (被称为量化步长或代码宽度) 代表 ADC 的最小刻度 (被称为  $LSB_n$ )。对于每个转换结果, 数字结果的离散特性产生的系统固有的量化不确定性为  $\pm 0.5 LSB_n$ 。理想的转换曲线中, 当模拟输入达到  $0.5 LSB_n$  时, 产生第一个数字跳变 (0 到 1 之间) 量化步长在整个输入电压范围内等量分配。

低于或高于参考电压模拟输入电压使得数字结果饱和至数字结果 0 或者  $2^n-1$ 。

实际转换曲线表现为在一定程度上偏离理想转换曲线:

- **偏移误差**指以误差最低的代码得到的真实转换曲线和理想转换曲线之间的偏差。误差最低的代码指的是对两种曲线来讲, 所有可能的代码中曲线拟合度最佳的情形。
- **增益误差**指的是实际转换曲线与理想转换曲线的斜率之间的偏差。同样是指所有代码中两种曲线拟合度最佳的情形。
- **微分非线性误差 (DNL)** 指的是实际代码宽度和理想代码宽度之间的偏差 (两个相邻数字转换结果对应的模拟输入电压之间的差别)。-1  $LSB_n$  的 DNL 值指示代码丢失。
- **积分非线性误差 (INL)** 实际转换曲线和调整过的理想转换曲线之间的偏差 (与实际转换曲线相同的偏移和增益误差, 但是代码宽度是等量的)。
- **总的不可调整误差 (TUE)** 描述了对于给定的测量范围来讲, 实际转换结果与理想转换特性之间的最大偏差。由于上面提到的这些误差可相互补偿, TUE 的值通常比单个误差的总和要小得多。

TUE 还包含生产工艺的差别和内部噪声的影响 (如果切换噪声由系统产生, 通常会导导致 TUE 值的增加)。

## 22.5 一般功能配置

对于每个通道、转换源等来讲很多参数都是单独选择的，但是一些配置则适用于整个 ADC 内核。

### 22.5.1 一般时钟方案和控制

ADC 内核的各部分由时钟信号驱动，时钟信号是基于用来访问 ADC 模块的、总线上的时钟  $f_{\text{ADC}}$  产生的。XC83x 的 ADC 与系统时钟相连，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，用来定义转换长度和采样时间。  
见 [章节 22.8.6](#)。
- 仲裁器使用数字时钟  $f_{\text{ADCD}}$ ，并用于定义仲裁周期的持续时间。
- 所有其它数字结构（如中断等）由模块时钟  $f_{\text{ADC}}$  直接驱动。

时序参数被编程到寄存器 [ADC\\_GLOBCTR](#) 中。

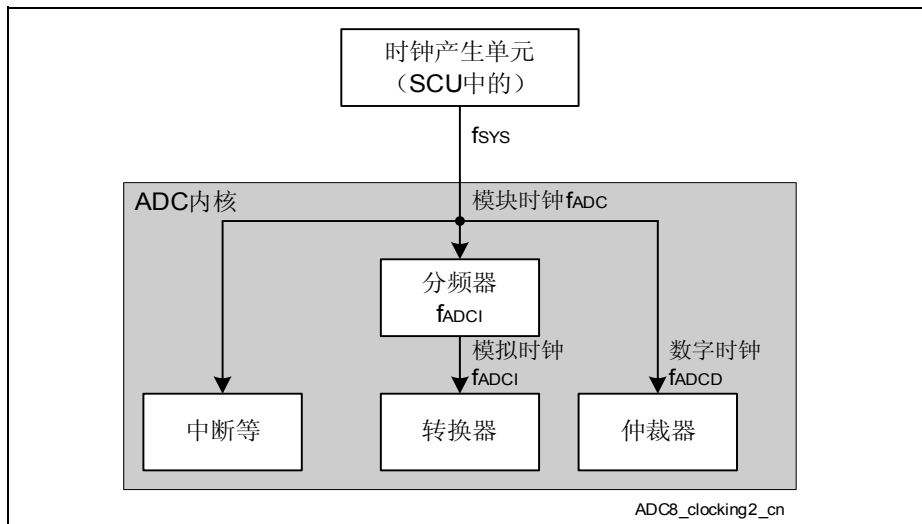


图 22-4 时钟方案

**注：** 转换运行过程中，如果 ADC 的时钟降至一个最小值以下或者停止，转换结果可能被破坏。为了得到正确的转换结果， $f_{\text{ADCI}}$  的频率一定不能超过所定义的范围。详情请参考相关数据手册。

全局控制寄存器定义转换器单元的基本时序参数和基本操作模式，包含下列各位：

- 使能 / 禁止模拟转换器
- 定义结果精度，8/10 位宽
- 定义  $f_{\text{ADCI}}$ （模拟部分的内部时钟频率）的分频因子 CTC
- 使能 / 禁止超出范围比较器中断

- 使能 / 禁止通道极限检查中断

## ADC\_GLOBCTR

全局控制寄存器

(CA<sub>H</sub>)

复位值：30<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
ANON	DW	CTC	ORCIEN	CLCIEN	0		
rw	rw	rw	rw	rw	rw	r	

符号	位	类型	描述
CLCIEN	2	rw	<b>使能通道极限检查中断</b> 该位使能与极限检查相关的通道事件中断，见图 22-15。如果缓存中有新的转换结果且结果触发极限检查单元，当使能位置为 1 时，产生通道事件中断。 0 <sub>B</sub> 禁止事件中断 1 <sub>B</sub> 使能事件中断
ORCIEN	3	rw	<b>使能超出范围比较器中断</b> 0 <sub>B</sub> 禁止超出范围比较器中断 1 <sub>B</sub> 使能超出范围比较器中断
CTC	[5:4]	rw	<b>转换时间控制</b> 该位域定义分频得到内部模拟时钟 $f_{ADCI}$ 的分频因子。该时钟提供用来计算转换和采样时间的内部时间基准。 00 <sub>B</sub> $f_{ADCI} = 1/3 \times f_{ADCA}$ 01 <sub>B</sub> $f_{ADCI} = 1/4 \times f_{ADCA}$ 10 <sub>B</sub> 保留 11 <sub>B</sub> $f_{ADCI} = 1/6 \times f_{ADCA}$ （缺省设置）
DW	6	rw	<b>数据宽度</b> 该位域定义转换结果的位数。 0 <sub>B</sub> 结果为 10 位宽（缺省设置） 1 <sub>B</sub> 结果为 8 位宽
ANON	7	rw	<b>模拟部分开启</b> 该位使能 ADC 模块的模拟部分并定义其工作模式。 0 <sub>B</sub> ADC 模块的模拟部分关闭，不可能进行转换。为了使功耗最小，内部模拟电路处于掉电状态，且不产生 $f_{ADCI}$ 。 1 <sub>B</sub> ADC 模块的模拟部分开启，可进行转换。模拟部分的自动掉电功能被禁止。

## 模数转换器

符号	位	类型	描述
<b>0</b>	[1:0]	r	<b>保留</b> 读操作返回 0；应写入 0。

全局状态寄存器指示转换的当前状态。

**ADC\_GLOBSTR**

全局状态寄存器

(CB<sub>H</sub>)

复位值：00<sub>H</sub>
**RMAP: 0, PAGE: 0**

7	6	5	4	3	2	1	0
0			<b>CHNR</b>		0	<b>SAMPLE</b>	<b>BUSY</b>
r			rh		r	rh	rh

符号	位	类型	描述
<b>BUSY</b>	0	rh	<b>模拟部分忙</b> 该位指示转换正在进行。 0 <sub>B</sub> 模拟部分空闲 1 <sub>B</sub> 当前正在进行转换
<b>SAMPLE</b>	1	rh	<b>采样阶段</b> 该位指示模拟输入信号当前是否被采样。 0 <sub>B</sub> 模拟部分未进行采样 1 <sub>B</sub> 模拟部分正在进行采样
<b>CHNR</b>	[5:3]	rh	<b>通道编号</b> 该位域指示当前正在转换哪个模拟输入通道。新的转换操作启动时，该信息被更新。 <i>注：位 5 仅适用于具有 8 个转换通道的器件。对于未实现的通道，这些位应当被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。</i>
<b>0</b>	2, [7:6]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 22.6 转换请求产生

ADC 内核的转换请求单元自动处理转换请求的产生。两个请求源可产生转换请求。由仲裁器对并行请求源进行决断，选择下一个要转换的通道。

发生触发事件时，请求源请求对某个模拟通道或通道序列进行转换。

- **软件触发**直接激活对应的请求源
- **外部触发**用外部事件，如来自定时器的 PWM 信号或来自端口引脚的触发脉冲，来同步触发请求源。

应用软件选择触发源、要转换的通道以及请求源的优先级。无需外部触发，就可由软件直接激活请求源。

仲裁器规律的扫描挂起的转换请求，并选择具有最高优先级的转换请求。然后转换请求被前送至转换器，启动所请求通道的转换操作。

每个转换源都可工作在单次模式或连续模式：

- **单次模式**，触发一次之后，请求编程设置的转换（序列）。随后的转换（序列）必须再一次被触发。
- **连续模式**，触发一次之后，自动重复请求编程设置的转换（序列）。

对于每个请求源来讲，根据 8 个可选的触发输入（REQTRx[H:A]）中的一个产生外部触发。

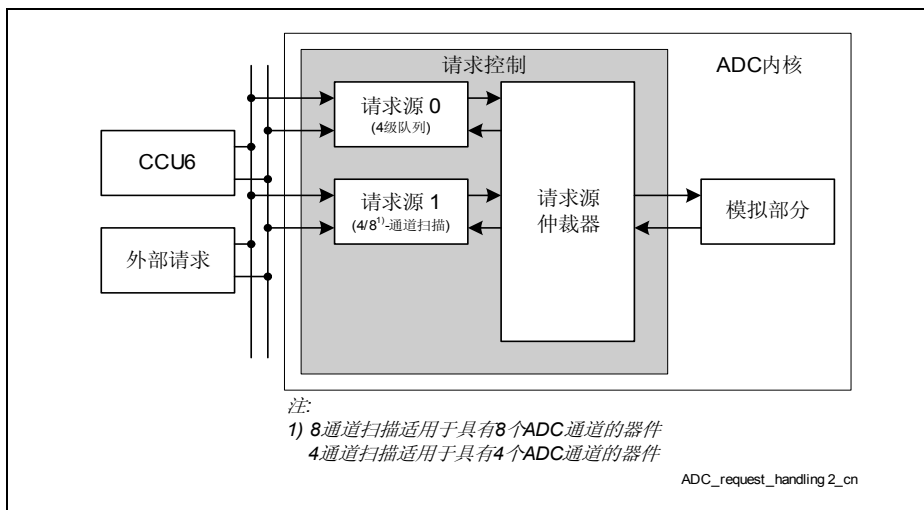


图 22-5 转换请求单元

有两类请求源可用：

- **通道扫描源**可发出一组连续序列的输入通道转换请求。在被使能所有通道中，该序列从具有最高通道编号的通道开始，按照通道编号降序排列。对于被使能的扫描序列，

每个通道按照顺序依次进行转换。

一个扫描请求源连续的或按照一个规律的时间基准转换一系列输入通道。例如，如果设置的优先级较低，可在后台任务中扫描一些输入通道，以更新那些对时间要求不严格的信息。请求源 1 是通道扫描源。

- **队列请求源**可按照任意的次序发出输入通道转换请求。该序列中的队列顺序可随意编程。可支持通道扫描源无法实现的、特定应用需要的转换序列。而且，还支持在一个序列中对同一个通道进行多次转换。

队列请求源连续的或根据规律的时间基准转换一系列输入通道。例如，如果设置为中等优先级，可在发生特定事件时（例如，与 PWM 同步），转换一些输入通道。与此同时，具有较低优先级的请求源挂起。

请求源 0 是一个 4 级队列源。

## 22.6.1 通道扫描请求源处理

每个模拟输入通道可包含在扫描序列中或者被排除在扫描序列之外（参见寄存器 **ADC\_CRCR1** 中的位）。编程设置的寄存器值保持不变（当前正在进行的扫描序列不会改变这些值）。在所有被使能进行通道扫描的通道中，从编号最高的通道开始，从高到低依次进行转换。

发生加载事件时，转换请求参数被传送至挂起位（见寄存器 **ADC\_CRCR1**）。挂起转换请求指示正在进行的扫描序列中将要转换的输入通道。由扫描请求源触发每次转换操作，并自动清除对应的挂起位。如果扫描源触发的最后一次转换结束且所有的挂起位都被清除，则认为当前的扫描序列的转换结束并产生一个请求源事件。

如果至少一个挂起位置位，则仅由仲裁器发出转换请求。

如果仲裁器因为更高优先级的请求而中止扫描请求源所触发的转换，对应的挂起位被自动置位。此措施保证不会丢失被中止的转换，而是在下一轮仲裁过程中进行处理。

触发单元根据选中的外部（ADC 模块之外的）触发信号产生加载事件。例如，定时器单元可发出一个与 PWM 事件同步转换的请求信号。

加载事件启动扫描序列，可由软件或通过选中的硬件触发产生加载事件。

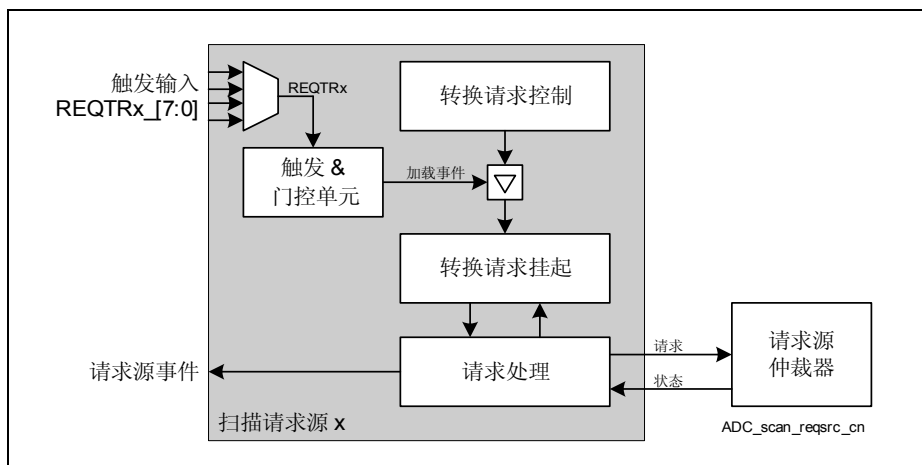


图 22-6 扫描请求源

## 扫描请求源操作

通过执行下述动作来**配置扫描请求源**：

- 通过编程设置 **ADC\_CRCR1**，选择扫描序列中的输入通道
- 如果想要由硬件触发转换，通过编程设置 **ADC\_ETRCR** 选择合适的触发输入，并编程设置 **ADC\_CRMR1** 来使能触发
- 通过编程设置寄存器 **ADC\_CRMR1** 来定义加载事件操作（挂起位处理、自动扫描模式）  
位 **LDM = 0** 的加载事件将被从 **ADC\_CRCR1** 的内容复制到 **ADC\_CRPR1**（覆盖模式）。该操作启动新的扫描序列并中止先前扫描序列中的任何挂起转换。  
位 **LDM = 1** 的加载事件将 **ADC\_CRCR1** 和 **ADC\_CRPR1** 的内容进行或操作（组合模式）。该操作启动的扫描序列将包括先前的扫描序列中的挂起转换。
- 使能对应的仲裁时隙（1），接受来自通道扫描源的转换请求（见寄存器 **ADC\_PRAR**）。

通过产生加载事件来**启动通道扫描序列**：

- 如果选择并使能硬件触发，在选中的输入信号上（例如，来自定时器或输入引脚）产生所配置的跳变。
- 通过置位 **ADC\_CRMR1.LDEV = 1** 产生软件加载事件。
- 通过直接向 **ADC\_CRPR1** 的挂起位写扫描序列的方式产生加载事件。序列被复制到 **ADC\_CRCR1** 并自动产生加载事件。

在此情况下，可用一次数据写操作来定义并启动一个扫描序列。

*注：如果使能自动扫描，完成扫描序列之后，每次发生请求源事件时，自动产生加载事件。该方式连续的重复预定义的扫描序列（自动扫描）。*

通过执行下述步骤**停止或中止正在进行的扫描序列**：

- 如果使能外部门控，将门控信号切换至预定义的无效电平。该方式不会修改转换挂起位，仅阻止向仲裁器发出转换请求。
- 禁止仲裁器中的对应的仲裁时隙（1）。该方式不会修改转换挂起位的内容，仅阻止仲裁器接受来自请求处理模块的转换请求。
- 通过清除位域 **ENGT = 0<sub>B</sub>** 来禁止通道扫描源。通过置位 **ADC\_CRMR1.CLRPND = 1** 来清除挂起请求位。



## 扫描请求源事件和中断

如果扫描序列中的最后一个转换完成（所有挂起位 = 0），则发生一个扫描请求源事件。可根据图 22-7 中的结构，由请求源事件产生请求源事件中断。如果检测到请求源事件，则寄存器 **ADC\_EVINFR** 中对应的指示标志被置位。通过向寄存器 **ADC\_EVINCR** 中的对应位写 1，可由软件清除指示标志。

每次检测到相关请求源事件时（且由 **CRM Rx. ENSI** 使能的情况下），服务请求输出 **SRx** 被激活。

请求源事件和结果事件共享同一个寄存器。由寄存器 **ADC\_EVINFR** 中的位来保存请求源事件：

- 事件 1：通道扫描源 1 的请求源事件（仲裁时隙 1 中）

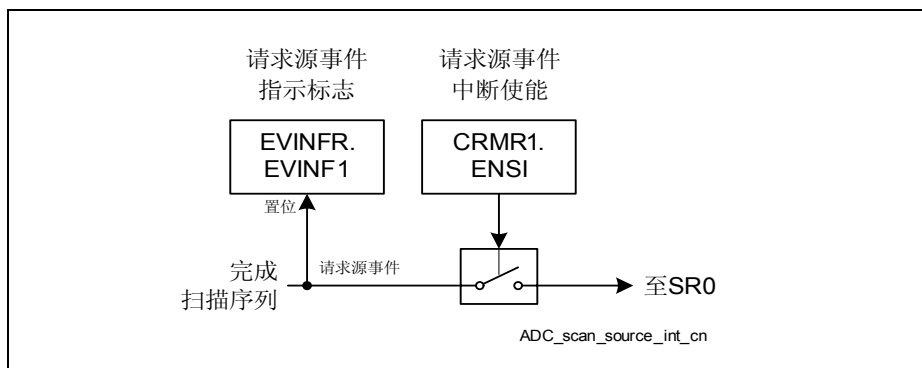


图 22-7 扫描请求源的中断产生

转换请求模式寄存器中的位用来设置需要的转换模式。

## ADC\_CRM Rx1

转换请求模式寄存器 1

(CC<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
0	LDEV	CLRPND	SCAN	ENSI	ENTR	0	ENGT
r	w	w	rw	rw	rw	r	rw

**模数转换器**

符号	位	类型	描述
<b>ENGT</b>	0	rw	<b>使能门控功能</b> 该位用于使能请求源的门控功能。 0 <sub>B</sub> 门控线恒为 0，请求源关闭。 1 <sub>B</sub> 门控线恒为 1，请求源开启。
<b>ENTR</b>	2	rw	<b>使能外部触发</b> 该位用于使能外部触发。如果使能，在外部触发输入 REQTR 上检测到上升沿时，则发生加载事件。 0 <sub>B</sub> 禁止外部触发 1 <sub>B</sub> 使能外部触发
<b>ENSI</b>	3	rw	<b>使能请求源中断</b> 该位用于使能请求源中断。完成该请求源的最后—个挂起转换时（当 PND = 0），产生中断。 0 <sub>B</sub> 禁止请求源中断 1 <sub>B</sub> 使能请求源中断
<b>SCAN</b>	4	rw	<b>使能自动扫描</b> 该位用于使能自动扫描功能。如果使能，当转换（该请求源请求的）完成且 PND = 0 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>CLRPND</b>	5	w	<b>清除挂起位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 寄存器 CRPR1 中的位被复位
<b>LDEV</b>	6	w	<b>产生加载事件</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 产生加载事件
<b>0</b>	7,[1:0]	r	<b>保留</b> 读操作返回 0；应写入 0。

模数转换器

转换请求控制寄存器 1 选择请求源 1（通道扫描源）请求的、需要转换的通道。当发生加载事件时，这些位被用来更新挂起寄存器 CRPR1。

注： 写访问寄存器 CRPR1 还更新寄存器 CRCR1 并产生加载事件。

**ADC\_CRCR1**  
 转换请求控制寄存器 1  
 RMAP: 0, PAGE: 6

(CA<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位	类型	描述
CHx (x = 0 - 7)	x	rwh	<b>通道位 x</b> 每位对应一个模拟通道，通道编号 x 由该位在寄存器中的位置定义。当发生加载事件时，转换请求挂起寄存器中的对应位 x 将被该位覆盖。 0 <sub>B</sub> 并行请求源将不请求模拟通道 x 进行转换。 1 <sub>B</sub> 并行请求源将请求模拟通道 x 进行转换。 <i>注意： 位 4-7 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道，这些位应被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。</i>

## 模数转换器

转换请求挂起寄存器指示请求源 1（通道扫描源）中的哪个通道请求转换操作。发生加载事件时，由挂起寄存器 **CRCR1** 中的内容更新这些位。

注：写寄存器 **CRPR1** 还会更新 **CRCR1** 并产生加载事件。

**ADC\_CRPR1**  
**转换请求挂起寄存器 1** (CB<sub>H</sub>) 复位值：00<sub>H</sub>  
**RMAP: 0, PAGE: 6**

7	6	5	4	3	2	1	0
<b>CHP7</b>	<b>CHP6</b>	<b>CHP5</b>	<b>CHP4</b>	<b>CHP3</b>	<b>CHP2</b>	<b>CHP1</b>	<b>CHP0</b>
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>CHPx</b> (x = 0 - 7)	x	rwh	<b>通道挂起位 x</b> 写方式： 写该位实际上指向的是寄存器 <b>CRCR1</b> 读方式： 每位对应一个模拟通道；通道编号 x 由该位在寄存器中的位置定义。 仲裁器自动复位（转换启动时）或者再次置位（转换被中止时）相应模拟通道。 0 <sub>B</sub> 并行请求源未请求模拟通道 x 进行转换操作 1 <sub>B</sub> 并行请求源请求模拟通道 x 进行转换操作 <i>注意：位 4-7 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道，这些位应当被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。</i>
<b>0</b>	[7:4]	r	<b>保留</b> 读操作返回 0；应写入 0。

注：可从该寄存器位置读取的位通常是 ‘rh’ 类型。不能直接由写操作修改这些位。写操作可修改 **CRCR1** 中的位（这就是该寄存器类型被标志为 ‘rwh’ 的原因）并在一个时钟周期之后引发加载事件。

### 22.6.2 队列请求源处理

队列请求源支持由任意通道组成的短转换序列（与使能通道按照固定次序进行转换的扫描请求源不同）。编程设置的序列保存在队列缓存中（基于一个 FIFO 机制）。通过队列输入给出所请求通道编号，其中队列级 0 用来定义下一个要转换的通道。

仅当一个有效的请求源被保存在队列级 0 中时，才能由仲裁器发出一个转换请求。



## 队列请求源的特性

XC83x 的 ADC 内核提供具有缓存能力的队列请求源：

- 队列请求源 0 提供 4 级缓存且可处理多达 4 个输入通道请求源。支持特定应用的短转换序列，尤其适用于要求多次转换同一个通道的、对时间要求苛刻的序列。

## 队列请求源的操作

执行下述动作来**配置队列请求源**：

- 向队列输入寄存器 **ADC\_QINR0** 写请求源来定义序列。因为如果使能重填特性，则不允许软件写 **QINRx**，因此必须在使能队列请求源之前初始化整个序列。
- 如果需要硬件触发，通过编程设置 **ADC\_ETRCR** 来选择合适的触发输入。  
编程设置寄存器 **ADC\_QMR0** 的位域 **ENGT** 来使能触发
- 使能对应的仲裁时隙（0），接受来自队列请求源的转换请求（见寄存器 **ADC\_PRAR**）。

通过产生触发事件来**启动队列序列**：

- 如果选择并使能硬件触发，在选中的输入信号（如来自定时器或输入引脚）上产生配置的跳变。
- 通过设置 **QMRx.TREV = 1** 产生软件触发事件
- 向空队列写新的队列请求。这使得一个（新的）有效队列请求被前送至队列级 0 并启动转换请求（如果由 **QMRx.ENGT** 使能且无需等待外部触发）。

*注：如果重填机制被激活，处理过的请求源被自动重载到队列中。通过此方式连续的重复对应的序列（自动扫描）。  
在此情况下，当队列源正在运行时，不要写队列输入。  
向整个被重填的队列进行写操作时，该写操作被忽略。*

通过执行下述动作来**停止或中止队列序列**：

- 禁止仲裁器中的对应仲裁时隙（0）。此动作不修改队列请求源，但是阻止仲裁器接受来自请求处理模块的请求。
- 通过清除位域 **ENGT = 0<sub>B</sub>** 来禁止队列请求源：
  - 通过置位 **QMRx.CLRV = 1** 的方式，使下一个挂起的队列请求失效。  
如果备份级中包含一个有效请求源，需要使该请求源失效，否则应该使队列级 0 失效。
  - 通过置位 **QMRx.FLUSH = 1** 删去队列中的所有请求源。

## 队列请求源事件和中断

当转换完成时，则发生队列源请求源事件。可根据图 22-9 的结构，由请求源事件产生请求源事件中断。如果检测到请求源事件，则置位 **ADC\_EVINFR** 寄存器中的相应的指示位。向寄存器 **ADC\_EVINCR** 的对应位写 1 的方式可软件清除中断指示标志。正常序列转换的情况下中断使能位来自队列级 0；被中止之后，重复转换的情况下中断使能位来自备份级。

每次检测到相关的请求源事件时（如果分别由 **Q0Rx.ENS**，或 **QBURx.ENS** 使能），服务请求输出线 **SRx** 被激活。

请求源事件和结果事件共享同一个寄存器。请求源事件保存在寄存器 **ADC\_EVINFR** 中。

- 事件 0：队列源 0 的请求源事件（仲裁时隙 0 中的）

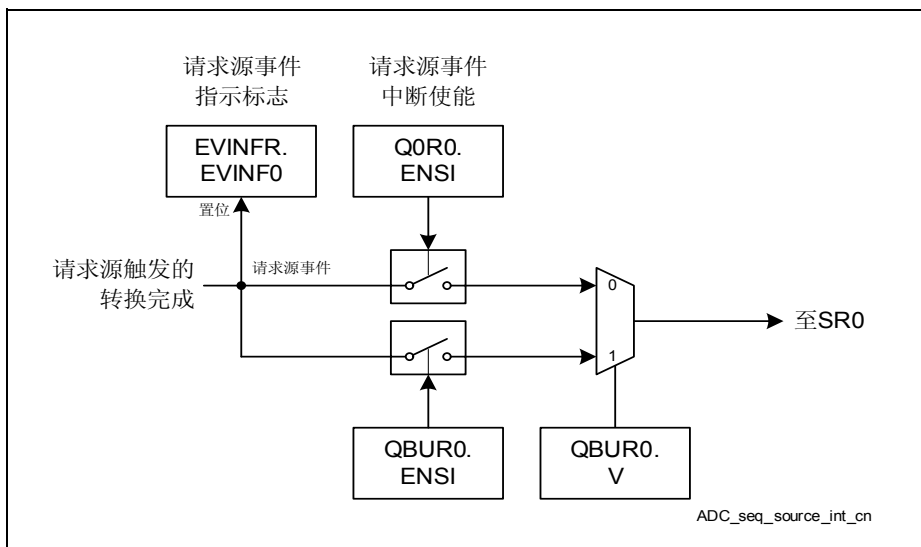


图 22-9 队列请求源的中断产生



队列模式寄存器用于配置队列请求源的工作模式。

# ADC\_QMR0

队列模式寄存器

(CD<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
CEV	TREV	FLUSH	CLR V	0	ENTR	0	ENGT
w	w	w	w	r	rw	r	rw

符号	位	类型	描述
ENGT	0	rw	<b>门控使能</b> 该位用于使能请求源的门控功能。 0 <sub>B</sub> 门控线一直为 0，该请求源被关闭。 1 <sub>B</sub> 门控线一直为 1，该请求源被开启。
ENTR	2	rw	<b>使能外部触发</b> 该位用于使能外部触发功能。如果被使能，当寄存器 Q0R0 或 QBUR0 中至少一个 V 位置位时，且如果在外部触发输入 REQTR 上检测到上升沿，则位 EV 被置位。 0 <sub>B</sub> 禁止外部触发 1 <sub>B</sub> 使能外部触发
CLR V	4	w	<b>清除 V 位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 寄存器 Q0R0 或 QBUR0 中的 V 位被复位。如果 QBUR0.V = 1，则 QBUR0.V 被复位；如果 QBUR0.V = 0，则 Q0R0.V 被复位。
FLUSH	5	w	<b>清空队列</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 队列寄存器中的所有位 V 和位 EV 被复位。队列中不再包含有效请求源。
TREV	6	w	<b>触发事件</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 软件产生触发事件。如果请求源等待触发事件，则启动转换请求。
CEV	7	w	<b>清除事件位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 EV 被清除

模数转换器

符号	位	类型	描述
0	1, 3	r	保留 读操作返回 0；应写入 0。

注： 通过 **QMR.CLRV** 或 **QMR.FLUSH** 软件修改队列内容之前，必须完成该队列相关的所有硬件动作。因此，必须禁止仲裁间隙且软件必须等待至少两个仲裁周期（以确保该请求源不再是仲裁获胜者）。然后，要检查位 **ADC\_GLOBCTR.BUSY** 以确保该请求源触发的转换不再运行。然后软件可读取 **QBURx** 和 **Q0Rx** 并可以开始修改队列内容。

## 模数转换器

队列状态寄存器指示队列请求源的当前状态。填充级别和空队列信息请参考队列的中间级（如果有的话）和队列寄存器 0。被中止的转换保存在备份级中，不由这些位指示（参见 QBURx.V）。

### ADC\_QSR0

队列状态寄存器

(CE<sub>H</sub>)

复位值：20<sub>H</sub>

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
0	EMPTY	EV	0				FILL
r	rh	rh	r				rh

符号	位	类型	描述
<b>FILL</b>	[1:0]	rh	<b>填充级别</b> 该位指示在顺序请求源队列中有多少个有效请求源。每次有新请求写入 QINR0 中，该位域加 1；每完成一次转换，该位域减 1。如果填充级别达到最大值，则忽略新的转换请求。如果位 <b>EMPTY</b> = 1，表示队列中没有有效的请求源。 00 <sub>B</sub> 如果位 <b>EMPTY</b> = 0，表示队列中有 1 个有效请求源 01 <sub>B</sub> 如果位 <b>EMPTY</b> = 0，表示队列中有 2 个有效请求源 10 <sub>B</sub> 如果位 <b>EMPTY</b> = 0，表示队列中有 3 个有效请求源 11 <sub>B</sub> 如果位 <b>EMPTY</b> = 0，表示队列中有 4 个有效请求源
<b>EV</b>	4	rh	<b>事件检测</b> 当 V = 1 时，该位指示检测到有事件发生。一旦该位被置位，转换启动后该位自动复位。 0 <sub>B</sub> 还未检测到事件 1 <sub>B</sub> 已经检测到事件
<b>EMPTY</b>	5	rh	<b>队列为空</b> 该位指示顺序请求源是否包含有效请求源。如果队列已满（ <b>EMPTY</b> = 0），则新的请求源被忽略。 0 <sub>B</sub> 队列中已有“FILL+1”个有效请求源 1 <sub>B</sub> 队列为空，队列中没有有效的请求源。
<b>0</b>	[3:2], [7:6]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 模数转换器

队列输入寄存器是队列请求源转换请求的入口寄存器。

**ADC\_QINR0**

队列输入寄存器 0

(D2<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
<b>EXTR</b>	<b>ENSI</b>	<b>RF</b>	<b>0</b>			<b>REQCHNR</b>	
W	W	W	r			W	

符号	位	类型	描述
<b>REQCHNR</b>	[2:0]	w	<b>请求通道编号</b> 该位域用来定义所请求的通道编号。 <i>注意：位 2 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道，这些位应被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。</i>
<b>RF</b>	5	w	<b>重填</b> 该位定义重填功能。
<b>ENSI</b>	6	w	<b>使能请求源中断</b> 该位定义请求源中断功能。
<b>EXTR</b>	7	w	<b>外部触发</b> 该位定义外部触发功能。
<b>0</b>	[4:3]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 模数转换器

队列寄存器 0 用来监控挂起请求的状态（队列级别 0）。

**ADC\_Q0R0**

队列 0 寄存器 0

(CF<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
<b>EXTR</b>	<b>ENSI</b>	<b>RF</b>	<b>V</b>	<b>0</b>	<b>REQCHNR</b>		
rh	rh	rh	rh	r	rh		

符号	位	类型	描述
<b>REQCHNR</b>	[2:0]	rh	<b>请求通道编号</b> 该位域指示将要或当前请求转换的通道编号。 <i>注意：位 2 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道，这些位应被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。</i>
<b>V</b>	4	rh	<b>请求通道编号有效</b> 该位指示 REQCHNR、RF、ENSI 和 EXTR 中的数据是否有效。当有效请求源写入到队列输入寄存器 QINR0 中（或者由中间队列寄存器更新 QINR0）时，位 V 被置位。 0 <sub>B</sub> 数据无效 1 <sub>B</sub> 数据有效
<b>RF</b>	5	rh	<b>重填</b> 该位指示当挂起请求被执行后（转换启动）是被丢弃还是被自动重填到请求队列的顶端。 0 <sub>B</sub> 转换启动后丢弃请求 1 <sub>B</sub> 转换启动后将请求重填到队列中
<b>ENSI</b>	6	rh	<b>请求源中断使能</b> 该位指示转换结束后是否产生请求源中断。若转换结束且 ENSI = 1，则中断触发被激活。 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>EXTR</b>	7	rh	<b>外部触发</b> 该位定义转换请求是否和外部触发事件有关。事件标志（位 EV）指示是否已发生外部事件并可请求转换。 0 <sub>B</sub> 位 EV 不用来启动转换请求 1 <sub>B</sub> 位 EV 用来启动转换请求

## 模数转换器

符号	位	类型	描述
0	3	r	保留 读操作返回 0；应写入 0。

队列备份寄存器用来监控中止的队列请求的状态。

# ADC\_QBUR0

队列备份寄存器 0

(D2<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 6

7	6	5	4	3	2	1	0
EXTR	ENSI	RF	V	0	REQCHNR		
rh	rh	rh	rh	r	rh		

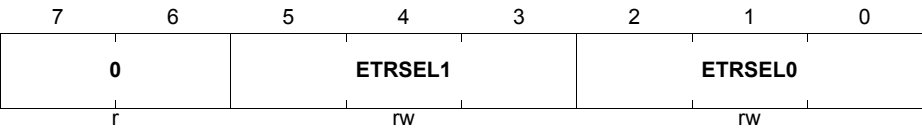
符号	位	类型	描述
REQCHNR	[2:0]	rh	<p><b>请求通道编号</b></p> <p>开始执行由 Q0R0 请求的转换时，由位域 Q0R0.REQCHNR 更新该位域。</p> <p><i>注意：位 2 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道，该位应被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。</i></p>
V	4	rh	<p><b>请求通道编号有效</b></p> <p>该位指示 REQCHNR，RF，ENSI 和 EXTR 中的数据是否有效。如果正在运行的转换被中止，则位 V 置位。当转换启动时，该位被复位。</p> <p>0B 备份寄存器中不包含有效数据，因为该数据所描述的转换操作未被中止。</p> <p>1B 数据有效。在处理 Q0R0 请求的转换之前，先请求执行被中止的转换。</p>
RF	5	rh	<p><b>重填</b></p> <p>开始执行由 Q0R0 请求的转换时，该位由位 Q0R0.RF 更新。</p>
ENSI	6	rh	<p><b>使能请求源中断</b></p> <p>开始执行由 Q0R0 请求的转换时，该位由位 Q0R0.ENSI 更新。</p>
EXTR	7	rh	<p><b>外部触发</b></p> <p>开始执行由 Q0R0 请求的转换时，该位由位 Q0R0.EXTR 更新。</p>
0	3	r	<p><b>保留</b></p> <p>读操作返回 0；应写入 0。</p>

注： 寄存器 **QBURx** 与寄存器 **QINRx** 共用一个地址。  
读操作返回 **QBURx** 寄存器中的状态位；写操作的目标是寄存器 **QINRx** 中的控制位。

### 22.6.3 硬件触发选择

可由软件和硬件信号激活请求源。硬件触发可为来自多个外设模块的信号或端口输入。  
外部触发控制寄存器中的位用来选择外部触发输入信号源。

**ADC\_ETRCR**  
外部触发控制寄存器 (D3<sub>H</sub>) 复位值：00<sub>H</sub>  
RMAP: 0, PAGE: 4



符号	位	类型	描述
ETRSEL0, ETRSEL1	[2:0], [5:3]	rw	<b>请求源 x 的外部触发选择</b> 该位域用来选择外部触发输入信号。 000 <sub>B</sub> 选择触发输入 REQTRxA 001 <sub>B</sub> 选择触发输入 REQTRxB 010 <sub>B</sub> 选择触发输入 REQTRxC 011 <sub>B</sub> 选择触发输入 REQTRxD 100 <sub>B</sub> 选择触发输入 REQTRxE 101 <sub>B</sub> 选择触发输入 REQTRxF 110 <sub>B</sub> 选择触发输入 REQTRxG 111 <sub>B</sub> 选择触发输入 REQTRxH
0	[7:6]	r	<b>保留</b> 读操作返回 0；应写入 0。



## 22.7 请求源仲裁

请求源仲裁器规律的依次查询请求源，寻找挂起的请求源。在仲裁周期内给每个请求源分配特定的时隙，称之为仲裁时隙。

根据应用的需要，在具有来自多个源的并行请求源的情况下，可通过寄存器 **ADC\_PRAR** 由用户配置每个请求源的优先级，从而仲裁器可选择下一个要转换的通道。

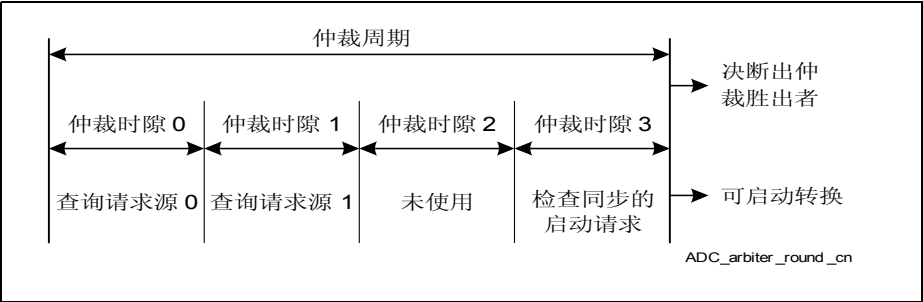
被禁止的或未使用的仲裁时隙被认为是空的，不参与仲裁。复位之后，所有时隙都被禁止，要参与仲裁过程，则必须使能这些时隙（通过寄存器 **ADC\_PRAR**）。

**图 22-10** 给出仲裁序列的总结。仲裁周期由每个可用请求源所对应的仲裁时隙组成。始终在最后一个时隙评估同步请求源，且同步请求源比所有其它请求源优先级都要高一些。为了调整与其它产品的时序（XC83x 不需此操作），可在仲裁周期中插入附加的时隙。每个仲裁周期结束时，仲裁器已经确定了具有最高优先级的转换请求。

如果在仲裁周期内启动一次转换，该仲裁周期将不会给出仲裁获胜者。

在 XC83x 中，下述请求源可用：

- 仲裁时隙 0：**4 级顺序源**，任意次序的 4 级序列
- 仲裁时隙 1：**4/8 通道扫描源**，预先定义次序的序列
- 最有一个仲裁时隙：**同步源**，可和来自另一个 ADC 内核的转换请求同步（始终以最高优先级处理同步的从 ADC 内核中的请求）



**图 22-10 仲裁周期**

## 22.7.1 仲裁器时序

仲裁器的时序（即仲裁周期）由一轮仲裁中仲裁时隙个数以及每个仲裁时隙的持续时间共同决定。

仲裁周期由 4 个仲裁时隙组成。

仲裁时隙的持续时间可编程设定  $t_{\text{Slot}} = \frac{1}{f_{\text{ADC}}}$ 。

从而仲裁周期的持续时间为  $t_{\text{ARB}} = N \times t_{\text{Slot}}$ （ $N$  = 仲裁时隙个数）。

仲裁周期引入一个时序最小单元的概念，用来检测输入的转换请求信号以及启动相关转换的最早时间点。该时序最小单元可引入一个最大为一个仲裁周期的抖动。通过使一轮仲裁周期最小的方式可减小这个抖动。

为了在转换请求的触发事件（如由定时器单元或外部事件触发的）和相关转换的启动操作之间，实现可重复产生的反应时间（没有抖动的固定延迟），主要有下述两种选择。对两种选择来讲，在触发事件发生之前，转换器必须空闲且一定不能有挂起的其它转换请求，空闲状态的持续时间至少是一个仲裁周期。

- 如果位 **ADC\_PRAR.ARB** = 0，**仲裁器持续运行**。  
转换触发的产生必须和仲裁器的时序同步，输入的触发应当精确的为仲裁器最小时间单元的  $n$  倍（ $n = 1, 2, 3, \dots$ ）。为了具有灵活性，仲裁时隙的持续时间可编程设置为时钟周期  $f_{\text{ADC}}$  的几倍。
- 如果位 **ADC\_PRAR.ARB** = 1，当没有挂起的转换请求时，**一个仲裁周期之后，仲裁器停止**。如果至少有一个使能的请求源则再次启动仲裁器。转换请求的触发不需要与仲裁器的时序同步。

### 22.7.2 请求源优先级和转换启动模式

每个请求源的优先级都可配置，因此仲裁器可决断来自不同请求源的并行请求。选择具有最高优先级的请求源进行转换。这些可配置的优先级（见寄存器 **ADC\_PRAR**）可适应给定应用的需要。

**转换启动模式**决定如何处理在仲裁中获胜的转换请求。

优先级和仲裁寄存器用来定义请求源的优先级，每个请求源的启动模式以及使能 / 禁止仲裁时隙以决定是否接受某转换请求。

*注： 只有当请求源被禁止，且该源请求的正在进行的转换完成之后，才能改变其优先级和转换启动模式设置。*

**ADC\_PRAR**  
 优先级和仲裁寄存器 (CC<sub>H</sub>) 复位值: 00<sub>H</sub>  
 RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
<b>ASEN1</b>	<b>ASEN0</b>	<b>0</b>	<b>ARBM</b>	<b>CSM1</b>	<b>PRIO1</b>	<b>CSM0</b>	<b>PRIO0</b>
rw	rw	r	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>PRIO0</b>	0	rw	<b>请求源 0 的优先级</b> 该位定义顺序请求源 0 的优先级。 0 <sub>B</sub> 低优先级 1 <sub>B</sub> 高优先级
<b>CSM0</b>	1	rw	<b>请求源 0 的转换启动模式</b> 该位定义顺序请求源 0 的转换启动模式。 0 <sub>B</sub> 选择等待 - 启动模式 1 <sub>B</sub> 选择取消 - 插入 - 重复模式
<b>PRIO1</b>	2	rw	<b>请求源 1 的优先级</b> 该位定义并行请求源 1 的优先级。 0 <sub>B</sub> 低优先级 1 <sub>B</sub> 高优先级
<b>CSM1</b>	3	rw	<b>请求源 1 的转换启动模式</b> 该位定义并行请求源 1 的转换启动模式。 0 <sub>B</sub> 选择等待 - 启动模式 1 <sub>B</sub> 选择取消 - 插入 - 重复模式

## 模数转换器

符号	位	类型	描述
<b>ARBM</b>	4	rw	<b>仲裁模式</b> 该位决定选择哪种仲裁模式。 $0_B$ 持续仲裁（缺省设置） $1_B$ 由挂起的转换请求启动仲裁
<b>ASEN0, ASEN1</b>	6, 7	rw	<b>使能仲裁时隙 x</b> 每位使能一个仲裁时隙。ASEN0 使能仲裁时隙 0； ASEN1 使能仲裁时隙 1。 如果禁止仲裁时隙，在仲裁中忽略与该时隙相连的请求源的挂起转换请求。 $0_B$ 禁止对应的仲裁时隙 $1_B$ 使能对应的仲裁时隙
<b>0</b>	5	r	<b>保留</b> 读操作返回 0；应写入 0。

注： 如果仲裁器不应持续运行（ $ARBM = 1$ ），与仲裁时隙 x 相连的请求源一定不能有效。因此在禁止某仲裁时隙之前，应当清除相关请求源的转换请求。

## 转换启动模式

仲裁器选出下一个要转换的请求之后，对该通道的处理取决于转换器的当前状态：

- 转换器当前空闲：立即启动在仲裁中获胜的转换请求
- 当前转换的优先级相同或更高：完成当前转换，随后启动在仲裁中获胜的转换请求
- 当前转换的优先级较低：接下来的动作由用户配置：
  - **等待-启动模式**：完成当前转换，然后启动在仲裁中获胜的转换请求。该模式提供的吞吐量最大，但是对于高优先级的转换来讲，会产生一个延迟 / 抖动。例子，见图 22-11：

转换 A 在 (t1) 时刻请求而在 (t2) 时刻启动。转换 B 在 (t3) 时刻请求，但是只有在完成转换 A 之后才能在 (t4) 时刻启动。

- **取消-插入-重复模式**：当前转换被中止，中止之后 ( $1 \dots 3 f_{\text{ADCl}}$  个时钟周期)，启动在仲裁中获胜的转换请求。

从对应的请求源中恢复被中止的转换请求并在下一个仲裁周期中参与仲裁。该模式对较高优先级的转换来讲带来的抖动 / 延迟最小，但是减少了整个的吞吐量。

例子见图 22-11：

转换 A 在 (t6) 时刻请求，在 (t7) 时刻启动。转换 B 在 (t8) 时刻请求，而在 (t9) 时刻启动，转换 A 被中止但是再次被请求。当转换 B 完成时 (t10)，转换 A 被重新启动。

例外情况：等待读取模式有效时，如果两个请求使用同一个结果寄存器，（见章节 22.10）则当前转换不能被中止。

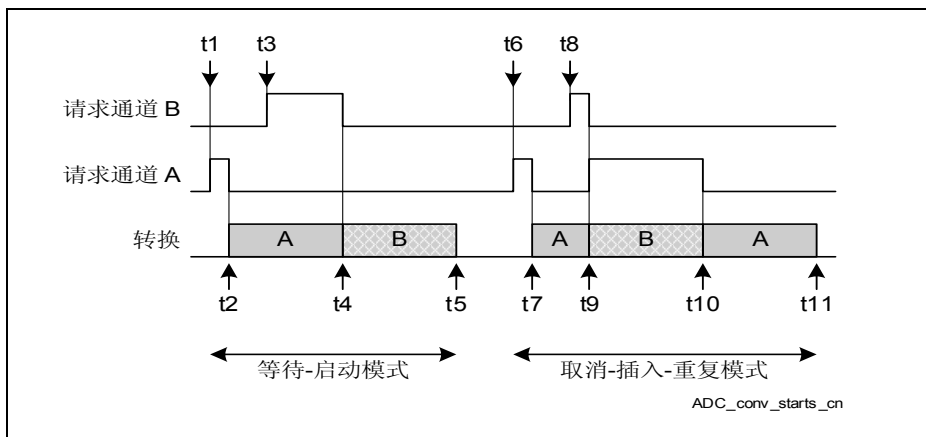


图 22-11 转换启动模式

在寄存器 **ADC\_PRAR** 中为每个请求源单独配置转换启动模式，并且该启动模式适用于该源请求的所有通道。在该例中，通道 A 由一个优先级较低请求源发出，而通道 B 的转换由一个优先级较高的请求源发出。

## 22.8 模拟输入通道配置

每个模拟输入通道都有数个可配置的参数，用来控制该通道的转换操作。某个通道赢得仲裁之后，其配置参数则适用于转换器。通道控制寄存器（[页 22-44](#) 上的 CHCTR<sub>x</sub>）定义下述参数：

- **转换参数：**输入级定义采样时间和数据宽度。所有通道使用相同的输入级。
- **参考选择：**三类内部参考可选，范围从内部的 1.2V 和 V<sub>DDP</sub> 3.3/5V。参考选择决定 ADC 的转换模式，即单端模式或者差分模式，见[章节 22.8.1](#)。  
请注意低参考电压得到的最小电压间隔较小，后果是 TUE 因为噪声的影响而增加。
- **结果目标：**转换结果可被保存到 4 个结果寄存器之一。
- **通道事件处理：**通道事件可用来限制结果落在规定区域之内或之外（极限检查）。

除了一般的通道控制，ADC 内核还支持一种特殊机制（被称为假名特性，见[章节 22.8.4](#)）将转换请求重新定向到另一个通道编号。

### 22.8.1 参考选择

ADC 的转换结果始终参考一个参考电压。如果模拟输入电压等于参考电压，则得到最大的数字结果值（满量程）。为了用满量程的数字值代表多个测量范围，用户可在下面三种转换模式中进行选择：

- 单端测量，ADC 参考内部连接至 V<sub>DDP</sub> 和 V<sub>SSP</sub>，见[图 22-12](#)。
- 类差分测量，ADC 参考连接至内部 1.2V 参考电压和来自 CH0 的参考地，见[图 22-13](#)。
- 单端测量，ADC 参考连接至内部 1.2V 参考电压和 V<sub>SSP</sub>，见[图 22-14](#)。

单端 ADC 转换的情况下，参考连接至 V<sub>DDP</sub> 和 V<sub>SSP</sub> 电压，ADC 参考电压内部连接至 V<sub>DDP</sub> 和 V<sub>SSP</sub> 电压，见[图 22-12](#)。

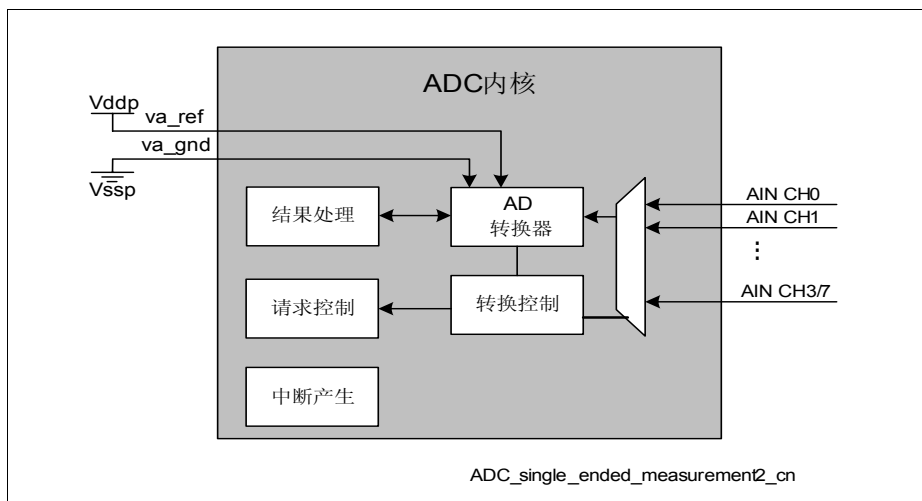
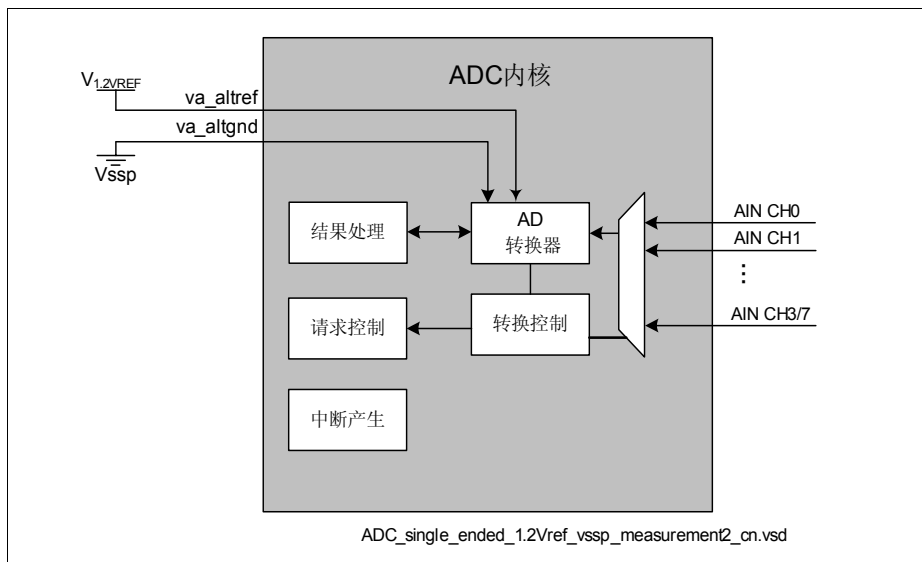


图 22-12 参考连接至 V<sub>DDP</sub> 和 V<sub>SSP</sub> 的单端测量





**图 22-14 使用内部 1.2V 参考电压和 Vssp 的单端测量模式**

通过编程设置 **ADC\_CHCTRx.REFSEL** 来为每个输入通道选择参考。需要注意：使用该模式时，两次转换之前需要建立时间，具体参数请参见数据手册。



## 22.8.2 通道参数

可由对应的通道控制寄存器来配置每个模拟输入通道。采样时间和结果宽度通过输入级进行选择。

通道控制寄存器为每个输入通道选择控制参数，还可用来选择目标结果寄存器以及内部参考电压，控制极限检查机制以及边界标志。

### ADC\_CHCTR<sub>x</sub> (x = 0 - 2)

通道 x 控制寄存器

(CA<sub>H</sub> + x \* 1)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 1

7	6	5	4	3	2	1	0
BFEN	LCC			REFSEL		RESRSEL	
rw	rw			rw		rw	

符号	位	类型	描述
RESRSEL	[1:0]	rw	<b>结果寄存器选择</b> 该位域定义该通道的转换结果存放在哪个结果寄存器中。 00 <sub>B</sub> 选择结果寄存器 0 01 <sub>B</sub> 选择结果寄存器 1 10 <sub>B</sub> 选择结果寄存器 2 11 <sub>B</sub> 选择结果寄存器 3
REFSEL	[3:2]	rw	<b>参考输入选择</b> 该位域定义该通道使用的参考源 00 <sub>B</sub> 使用参考 V <sub>DDP</sub> 和 V <sub>SSP</sub> 进行模数转换（见图 22-12） 01 <sub>B</sub> 使用内部 1.2V 和 CH0 参考地进行模数转换（见图 22-13） <sup>1)</sup> 10 <sub>B</sub> 使用内部 1.2V 和 V <sub>ssp</sub> 进行模数转换（见图 22-14） <sup>1)</sup> 11 <sub>B</sub> 保留；不要使用该组合。

## 模数转换器

符号	位	类型	描述
<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行为，见 <a href="#">章节 22.8.3</a> 。 000 <sub>B</sub> 不进行极限检查 001 <sub>B</sub> 结果落在区域 I 之外 010 <sub>B</sub> 结果落在区域 II 之外 011 <sub>B</sub> 结果落在区域 III 之外 100 <sub>B</sub> 始终进行极限检查（不考虑边界） 101 <sub>B</sub> 结果落在区域 I 之内 110 <sub>B</sub> 结果落在区域 II 之内 111 <sub>B</sub> 结果落在区域 III 之内
<b>BFEN</b>	7	rw	<b>使能边界标志</b> 该位用作边界标志 ADC_BF<2:0> 信号的门控控制 0 <sub>B</sub> 禁止对应通道的边界标志信号 1 <sub>B</sub> 使能对应通道的边界标志信号

1) 使用该模式时，两次转换之间需要建立时间。具体参数请参见数据手册。

**ADC\_CHCTR<sub>x</sub> (x = 3 - 5)**

通道 x 控制寄存器

(CA<sub>H</sub> + x \* 1)

复位值: 00<sub>H</sub>

**ADC\_CHCTR<sub>x</sub> (x = 6 - 7)**

通道 x 控制寄存器

(CC<sub>H</sub> + x \* 1)

复位值: 00<sub>H</sub>

**RMAP: 0, PAGE: 1**

7	6	5	4	3	2	1	0
0	LCC			REFSEL		RESRSEL	
r	rw			rw		rw	

符号	位	类型	描述
<b>RESRSEL</b>	[1:0]	rw	<b>结果寄存器选择</b> 该位域定义该通道的转换结果存放在哪个结果寄存器中。 00 <sub>B</sub> 选择结果寄存器 0 01 <sub>B</sub> 选择结果寄存器 1 10 <sub>B</sub> 选择结果寄存器 2 11 <sub>B</sub> 选择结果寄存器 3

## 模数转换器

符号	位	类型	描述
REFSEL	[3:2]	rw	<b>参考输入选择</b> 该位域定义该通道使用的参考源 00 <sub>B</sub> 使用参考 $V_{DDP}$ 和 $V_{SSP}$ 进行模数转换（见图 22-12） 01 <sub>B</sub> 使用内部 1.2V 和 CH0 参考地进行模数转换（见图 22-13） <sup>1)</sup> 10 <sub>B</sub> 使用内部 1.2V 和 $V_{SSP}$ 进行模数转换（见图 22-14） <sup>1)</sup> 11 <sub>B</sub> 保留；不要使用该组合。
LCC	[6:4]	rw	<b>极限检查控制</b> 该位域定义极限检查机制的行为，见章节 22.8.3。 000 <sub>B</sub> 不进行极限检查 001 <sub>B</sub> 结果落在区域 I 之外 010 <sub>B</sub> 结果落在区域 II 之外 011 <sub>B</sub> 结果落在区域 III 之外 100 <sub>B</sub> 始终进行极限检查（不考虑边界） 101 <sub>B</sub> 结果落在区域 I 之内 110 <sub>B</sub> 结果落在区域 II 之内 111 <sub>B</sub> 结果落在区域 III 之内
0	7	r	<b>保留</b> 读操作返回 0；应写入 0。

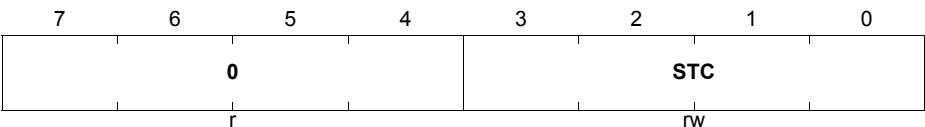
1) 使用该模式时，两次转换之间需要建立时间。具体参数请参见数据手册。

**注：** ADC\_CHCTR4 - 7 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道，这些位被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。

模数转换器

输入级用来定义采样阶段的长度以及转换精度。  
 缺省设置选择最小的采样阶段长度 2 个  $f_{\text{ADCl}}$  周期。  
 输入级寄存器用来为每个输入级选择采样时间和精度。

**ADC\_INPCR0**  
 输入级 0 寄存器 (CE<sub>H</sub>) 复位值: 00<sub>H</sub>  
 RMAP: 0, PAGE: 0



符号	位	类型	描述
STC	[3:0]	rw	<b>采样时间控制</b> 该位域定义采样时间的附加长度，以 $f_{\text{ADCl}}$ 时钟周期的倍数为单位 编程设置的采样时间被延长 2 个模拟时钟周期
0	[7:4]	r	<b>保留</b> 读操作返回 0；应写入 0。

### 22.8.3 极限检查

极限检查机制自动用两个边界值（边界 A 和边界 B）与每个转换结果进行比较。对每个通道来讲，用户从一组可编程设置的值（ADC\_LCBR0 和 ADC\_LCBR1）中间选择边界值。

根据两次比较的结果产生通道中断。通过相关通道控制寄存器的位域 LCC 选择产生中断的条件。

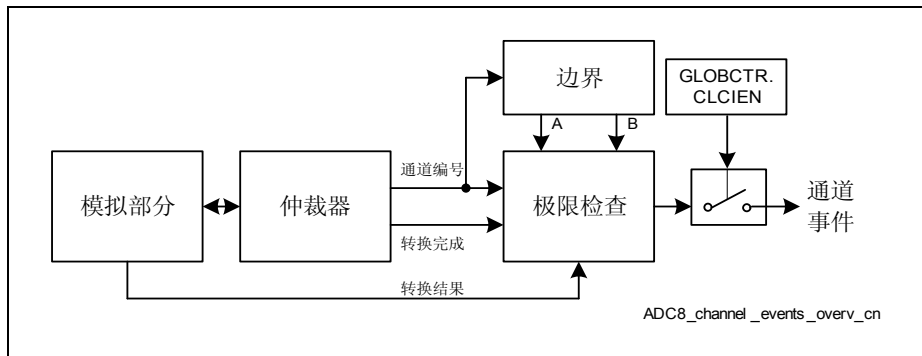


图 22-15 通道事件产生

两个可选择的边界值将转换结果的范围划分成三个区域：

- 区域 I：转换结果低于或等于两个边界
- 区域 II：转换结果高于一个边界并低于 / 等于另一个边界
- 区域 III：转换结果高于两个边界

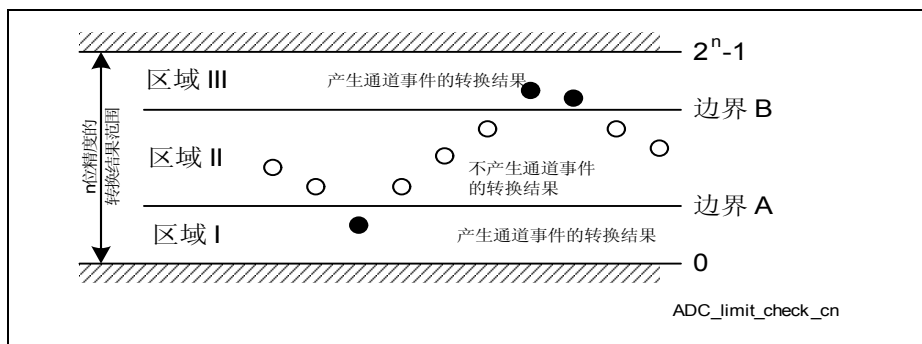


图 22-16 极限检查

极限检查示例显示仅当转换结果落在区域 II（LCC = 010<sub>B</sub>）所定义的正常范围之外时，才产生通道事件。

## 模数转换器

如果仅需要两个区域，则边界 A 和边界 B 使用相同的边界寄存器。在此情况下，区域 II 为空，仅有两个结果范围可用。此情况下应避免  $LCC = x10_B$  的设置。

极限检查的典型应用是监控任务（温度、压力、电流等等），此情况下结果的真实值不如其所在范围重要。只要测量值在规定的有效范围内，则不需要任何 CPU 动作。只有当转换结果落在有效范围之外，才应当产生中断来指示该紧急情况（温度过高，压力不足等等）。

如果想要监控的模拟输入信号是作为自动扫描序列的一部分而由一个规律的时间基准来自动触发的情况下，CPU 负荷被减至最小。从而在正常情况下，带来的 CPU 负荷为 0。

*注：过流保护的情况下，可使用通道事件来禁止 PWM 产生，以减少电流（在 XC83x 中，ADC 模块的中断输出线与 CCU6x 单元的对应输入相连，从而在没有 CPU 干预的情况下，允许迅速响应中断情况。*

### 边界标志控制

可通过配置极限检查机制来自动控制边界标志。当转换结果位于区域 III 时，可置位边界标志；当转换结果位于区域 I 时，清除边界标志。

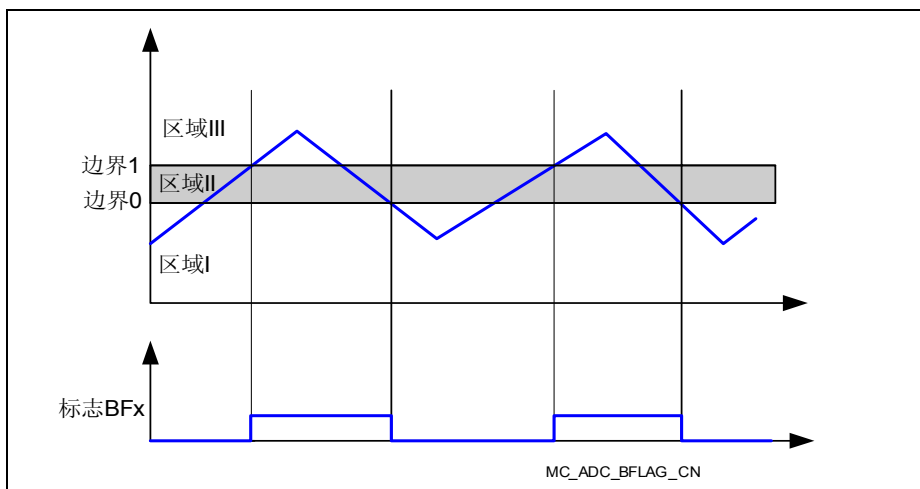


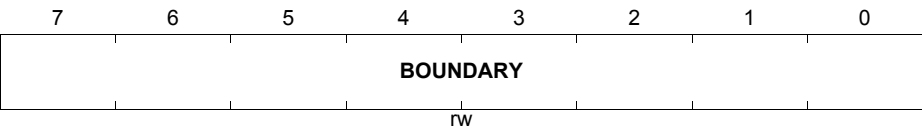
图 22-17 边界标志控制

边界值的不同定义出边界标志的置位 / 清除的滞后现象。

在用来监控线性霍尔元件的三个通道上使用该特性可产生馈送至 CCU6x 单元的三个霍尔位置输入的信号。

极限检查边界寄存器用来定义极限检查单元的比较值（边界值）。

**ADC\_LCBR0**  
**极限检查边界寄存器 0** (CD<sub>H</sub>) **复位值: 70<sub>H</sub>**  
**ADC\_LCBR1**  
**极限检查寄存器 1** (CF<sub>H</sub>) **复位值: B0<sub>H</sub>**  
**RMAP: 0, PAGE: 0**



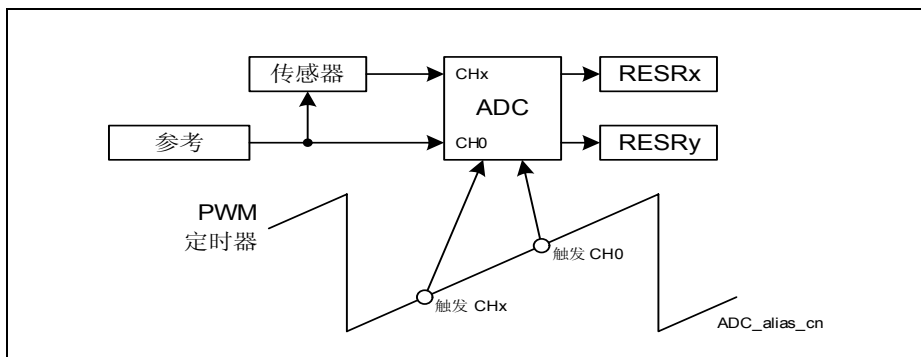
符号	位	类型	描述
<b>BOUNDARY</b>	[7:0]	rw	<b>极限检查的边界值</b> 使用该值和实际转换结果进行比较。

### 22.8.4 假名特性

假名特性将来自通道 CH0 的请求重新定向到另一个通道。这意味着将使用另一个通道的参数而不是 CH0 的参数转换来自 CH0 的通道输入。重新定向特性可用作下列场合：

- 同一个信号可被测量两次，两个结果（原始的和重新定向的）可分别保存在结果寄存器中。从而不依赖于 CPU 的中断延迟，该特性确保两次转换依次快速进行而且避免了数据丢失。  
 传感器信号被连接至一个输入（而不是两个）。该特性可在低成本应用中节约输入引脚，并减少误差计算中需要考虑的输入漏电流。
- 即使模拟输入 CH0 被用作备用参考地（见图 22-18），仍可使用 CH0 的内部触发和数据处理特性。
- 两次转换（对同一信号）的通道设置可使用不同的边界值、中断等等。
- 如果已经建立了队列请求源，来自通道 CH0 的转换请求可轻松重新定向至其它输入通道而无需清空整个队列。

**图 22-18** 给出上述特性的示例，传感器信号被连接至一个输入通道（CH<sub>x</sub>），但是触发两次转换（通道 CH<sub>x</sub> 和 CH0）。假名特性将转换请求从 CH0 重新定向至 CH<sub>x</sub>，但是还使用 CH0 的设置进行转换。尽管测量同一个模拟输入（CH<sub>x</sub>），两次转换结果可被保存并从结果寄存器中恢复出来：RESR<sub>x</sub>（触发的 CH<sub>x</sub> 转换）和 RESR0（触发的 CH0 转换）。另外，还可使能或禁止并选择不同的中断或极限边界。



**图 22-18 假名特性**

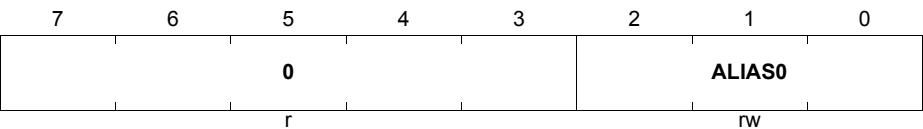
在典型的低成本 AC 驱动应用中，仅使用一个公共电流传感器来确定相电流。根据使用的 PWM 序列，测量值具有不同的意义且必须精确定位采样点在 PWM 周期中的位置。



模数转换器

假名寄存器指定代替 CH0 的通道编号，即 CH0 被请求时，实际上将使用另一个通道。编程设置的假名通道编号控制模拟输入复用器（ADC 的）。原始通道编号控制所有其它内部动作并同步转换请求。

**ADC\_ALR0**  
**假名寄存器 0** (CF<sub>H</sub>) **复位值: 00<sub>H</sub>**  
**RMAP: 0, PAGE: 4**



符号	位	类型	描述
<b>ALIAS0</b>	[2:0]	rw	<b>CH0 转换请求的假名通道编号</b> 该位域指示用来代替 CH0 的通道编号。并且使用通道 CH0 的设置完成转换。该位域的缺省设置为 CH0。 <i>注意：位 2 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道，该位应被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。</i>
<b>0</b>	[7:3]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 22.8.5 超出范围比较器

每个 ADC 通道都内嵌超出范围比较器机制，用来检测高于或低于  $V_{ddp}$  的电压。使用超出范围比较器之前，应当通过置位 **ADC\_ENORC.ENORCx** 来使能，并通过设置位 **ADC\_CNf.CNFx** 来决定检测电压是高于或低于  $V_{ddp}$ （见图 22-19）。

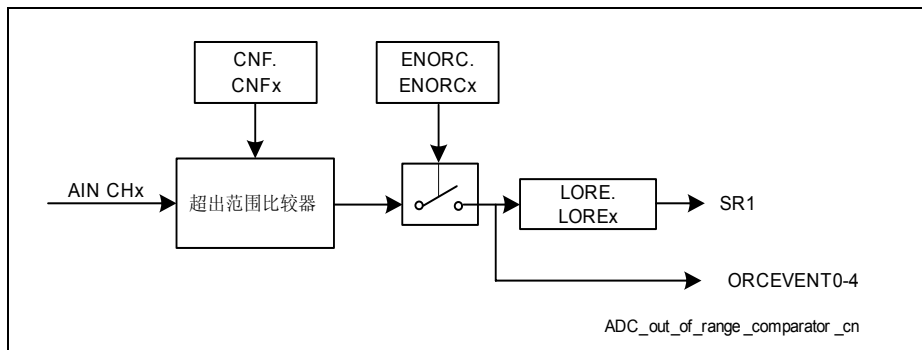


图 22-19 超出范围比较器

当发生电压超出范围事件时，事件被锁存到 **ADC\_LORE.LOREx** 中，通过 **ADC\_SR1** 置位中断请求并通过与 **ORCEVENT0-4** 连接来触发其它模块。仅 **ADC\_CH0-4** 可用来触发其它模块。

## 模数转换器

这些寄存器中的位域用来使能 ADC 通道的超出范围比较器。

**ADC\_ENORC**

使能超出范围比较器寄存器

(D3<sub>H</sub>)

复位值：00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
<b>ENORC7</b>	<b>ENORC6</b>	<b>ENORC5</b>	<b>ENORC4</b>	<b>ENORC3</b>	<b>ENORC2</b>	<b>ENORC1</b>	<b>ENORC0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>ENORCx</b> (x = 0 - 7)	x	rw	<b>使能超出范围比较器 x</b> 由该位使能对应模拟输入通道的超出范围比较器。 0 <sub>B</sub> 禁止超出范围比较器 1 <sub>B</sub> 使能超出范围比较器 <i>注意：位 4-7 仅适用于具有 8 个模拟 ADC 通道的器件。对于未实现的通道，这些位应当被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。</i>

模数转换器

该寄存器中的位用来选择：当检测到的每个输入 ADC 通道电压是高于或是低于 Vddp 时触发超出范围比较器。

**ADC\_CNF**  
**配置超出范围比较器寄存器 (D2<sub>H</sub>)**  
**RMAP: 0, PAGE: 4**  
**复位值: 00<sub>H</sub>**

7	6	5	4	3	2	1	0
CNF7	CNF6	CNF5	CNF4	CNF3	CNF2	CNF1	CNF0
rW	rW	rW	rW	rW	rW	rW	rW

符号	位	类型	描述
<b>CNF<sub>x</sub></b> <b>(x = 0 - 7)</b>	x	rW	<b>超出范围比较器标志 x</b> 该位选择由 CH <sub>x</sub> 上升沿还是下降沿触发超出范围比较器标志寄存器。 0 <sub>B</sub> 下降沿触发超出范围事件寄存器 1 <sub>B</sub> 上升沿触发超出范围事件寄存器 <i>注：位 4-7 仅适用于具有 8 个模拟 ADC 通道的器件。对于未实现的通道，这些位应当被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。</i>

## 模数转换器

这些寄存器中的位域指示对应模拟输入通道是否已发生电压超出范围事件。如果某模拟输入通道发生电压超出范围事件，则该寄存器中的对应位锁存至 1。向该寄存器写 0 可清除事件标志。

### ADC\_LORE

锁存超出范围事件寄存器

(D2<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 0

7	6	5	4	3	2	1	0
LORE7	LORE6	LORE5	LORE4	LORE3	LORE2	LORE1	LORE0
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>LORE<sub>x</sub></b> ( <b>x = 0 - 7</b> )	x	rwh	<p><b>锁存超出范围事件 x</b></p> <p>该位用来指示哪个模拟输入通道发生了电压超出范围事件。LORE 位可由硬件置位且只能由软件清除。</p> <p>0<sub>B</sub> 通道 x 未发生电压超出范围事件。向该位写 ‘0’ 可清除该标志。</p> <p>1<sub>B</sub> 通道 x 已发生电压超出范围事件</p> <p><i>注意: 位 4-7 仅适用于具有 8 个模拟 ADC 通道的器件。对于未实现的通道, 这些位应被当作保留位: 类型为 ‘r’, 读操作返回 0; 应写入 0。</i></p>

## 22.8.6 转换时序

转换所需的总转换时间取决于下述几个可由用户定义的因子：

- ADC 转换时钟频率，其中  $f_{\text{ADCI}} = f_{\text{ADC}} / (\text{CTC} + 3)$
- 所选择的采样时间，其中  $t_{\text{S}} = (2 + \text{STC}) \times t_{\text{ADCI}}$   
(STC = 附加采样时间，参见 [章节 22.5.1](#))
- 所选择的结果宽度 N (8/10 位)
- 以模块时钟速度进行的同步阶段

转换时间由采样时间、转换步骤和同步阶段组成，可通过下面的公式计算：

$$t_{\text{CN}} = t_{\text{ADC}} \times (1 + r \times (3 + n + \text{STC}))$$

$$r = \text{CTC} + 3,$$

CTC = 转换时间控制 ([ADC\\_GLOBCTR.CTC](#))

STC = 采样时间控制 ([ADC\\_INPCR0.STC](#))

n = 8 或 10 (分别对应 8 位和 10 位转换)

$$t_{\text{ADC}} = 1 / f_{\text{ADC}}$$

触发模数转换的频率还取决于下面几个可配置的因子：

- 可选择的转换时间，由输入级的设置来定义
- 被取消的、且需要重复进行的转换带来的延迟
- 可配置的仲裁周期时间
- 外部触发信号的频率，如被使能

例 1:

当 N=10 位, STC=0000<sub>B</sub>,  $f_{\text{ADC}}=48\text{Mhz}$ , CTC=01<sub>B</sub>

$$f_{\text{ADCI}} = (48\text{Mhz} / (1 + 3)) = 12\text{Mhz} = 83.33\text{ns} \quad (22.1)$$

$$t_{\text{sample}} = (2 + 0) \times 83.33\text{ns} = 166.67\text{ns} \quad (22.2)$$

表 22-6      采样时间

采样时间	ADC_INPCR0.STC (二进制)
$2 \times t_{\text{ADCI}}$	0000 <sub>B</sub>
$3 \times t_{\text{ADCI}}$	0001 <sub>B</sub>
...	
$17 \times t_{\text{ADCI}}$	1111 <sub>B</sub>

$$t_{\text{conv}} = \left( \frac{1}{48\text{MHz}} \right) \times (1 + (1 + 3) \times (3 + 10 + 0)) = 1.104\mu\text{s} \tag{22.3}$$

## 22.8.7 通道事件和中断

发生通道事件时，可产生如图 22-20 中所示结构的通道事件中断。如果检测到通道事件，则置位寄存器 **ADC\_CHINFR** 中对应的中断指示标志。向寄存器 **ADC\_CHINCR** 中的对应位写 1，可软件清除指示标志。

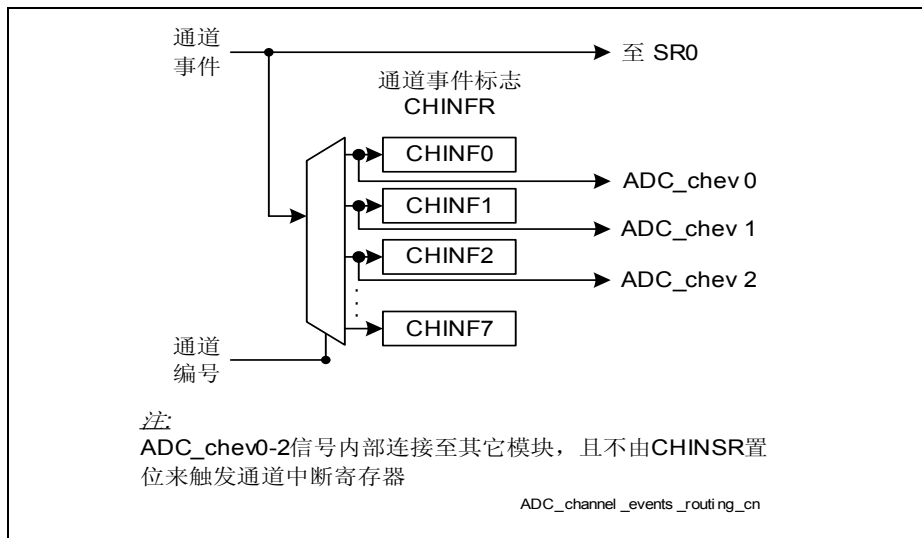


图 22-20 通道事件中断产生



## 22.9 转换结果处理

每个模拟输入通道的转换结果可保存在 4 个转换结果寄存器中的一个（由相关通道控制寄存器 CHCTR<sub>x</sub> 中的位域 RESRSEL 选择）。该结构为不同组的通道提供保存转换结果的不同位置。用户可根据应用的需要（数据压缩、自动扫描、假名特性等等），来分配转换结果，使 CPU 的负荷最小并且更加能容忍较长的中断延迟。

### 22.9.1 转换结果的保存

每个结果寄存器都有一个对应的数据有效标志（VF<sub>x</sub>）。该标志指示何时“新的”有效数据被保存到对应的结果寄存器且可供读取。

根据结果寄存器读取方式（见下文），当结果被读出时，相应的有效标志被自动清除或保持置位。

- 自动清除有效标志为结果的产生和恢复提供简单的握手。该方式还支持等待 - 读取模式。
- 有效标志保持置位。无需应用给出握手信号就能给出结果值的方式来支持调试。

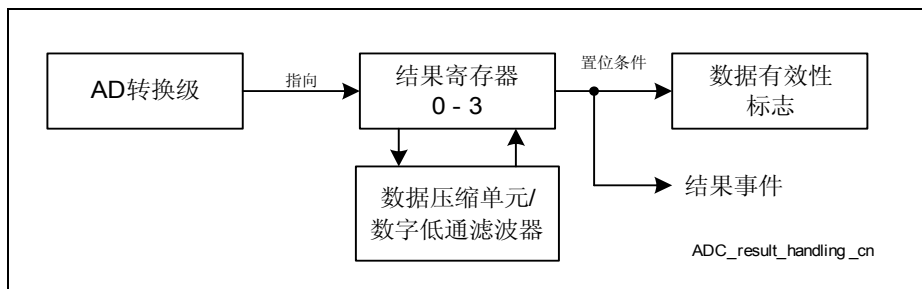


图 22-21 转换结果保存

转换结果的处理由以下功能组成：

- 转换结果保存到用户可配置的寄存器中
- 等待 - 读取模式，多个通道共用同一个结果寄存器时，避免数据损失（见 [章节 22.10](#)）
- 结果事件中中断（见 [章节 22.10.1](#)）
- 数据压缩或抗混叠滤波和数字低通滤波器（见 [章节 22.10.2](#)）

可在每个结果寄存器中叠加多达 4 个结果值。此方式减少了 ADC 产生中断的频率。

- 标准应用读取方式 (RESRxL/H):
  - **8 位转换模式, 禁止数据压缩和低通滤波器** (即 `ADC_GLOBCTR.DW=1, RCRx.DRCTR=0, RCRx.DLPF=0`)
 

选择该模式时, 目标结果寄存器为 `ADC_RESRxL (x=0-3)` 和 `ADC_RESRxH (x=0-3)` 中, 见图 22-22。

在 `ADC_RESRxL (x=0-3)`, 其中位 2-0 指示触发结果事件的通道编号; 寄存器 `ADC_RESRxH (x=0-3)`, 位 7-0 返回转换结果。读取结果时自动清除有效标志。只有不进行数据压缩时, 该读取方式才有用。
  - **10 位转换模式, 禁止数据压缩和低通滤波器** (即 `ADC_GLOBCTR.DW=0, RCRx.DRCTR=0, RCRx.DLPF=0`)
 

选择该模式时, 目标结果寄存器为 `ADC_RESRxL (x=0-3)` 和 `ADC_RESRxH (x=0-3)`, 见图 22-23。

在 `ADC_RESRxL (x=0-3)`, 其中位 2-0 指示触发结果事件的通道编号, 位 7-5 返回转换结果的低 3 位。在 `ADC_RESRxH (x=0-3)` 中, 位 6-0 返回转换结果的位 9-3。读取结果时自动清除有效标志。只有不进行数据压缩时, 该读取方式才有用。
- 累加应用读取方式 (RESRxL/H):
  - **8 位转换模式, 使能数据压缩并禁止数字低通滤波器** (即 `ADC_GLOBCTR.DW=1, RCRx.DRCTR=1, RCRx.DLPF=0`)
 

选择该模式时, 目标结果寄存器为 `ADC_RESRxL (x=0-3)` 和 `ADC_RESRxH (x=0-3)`, 见图 22-24。

第一次转换之后, 在 `ADC_RESRxL (x=0-3)` 结果寄存器中, 位 2-0 指示触发结果事件的通道编号。在 `ADC_RESRxH (x=0-3)` 中, 位 7-0 返回转换结果。该方式允许一次读取结果。读取结果时自动清除有效标志。

第二次转换时, 在 `ADC_RESRxL (x=0-3)` 结果寄存器中, 位 2-0 指示触发结果事件的通道编号, 位 7 为累加结果的最低位。在 `ADC_RESRxH (x=0-3)` 中, 位 7-0 返回累加结果的位 8-1。
  - **10 位转换模式, 使能数据压缩, 禁止数字低通滤波器** (即 `ADC_GLOBCTR.DW=0, RCRx.DRCTR=1, RCRx.DLPF=0`)
 

选择该模式时, 目标结果寄存器为 `ADC_RESRxL (x=0-3)` 和 `ADC_RESRxH (x=0-3)`, 见图 22-25。

第一次转换之后, 在 `ADC_RESRxL (x=0-3)` 结果寄存器中, 位 2-0 指示触发结果事件的通道编号, 位 7-5 返回转换结果的位 2-0。在 `ADC_RESRxH (x=0-3)` 中, 位 6-0 返回转换结果的位 9-3。读取结果时, 自动清除有效标志。

第二次转换之后, 在 `ADC_RESRxL (x=0-3)` 结果寄存器中, 位 2-0 指示触发通道事件的通道编号, 位 7-5 返回累加结果的位 2-0。在 `ADC_RESRxH (x=0-3)` 中, 位 7-0 返回累加结果的位 10-3。
- 数字低通滤波应用读取方式 (RESRxL/H):
  - **10 位转换模式, 禁止数据压缩, 使能数字低通滤波器** (即 `ADC_GLOBCTR.DW=0, RCRx.DRCTR=0, RCRx.DLPF=1`) 该模式仅适用于 10 位转换, 且使用数字低通滤波器时, 一定不能使能数据压缩特性, 原因是 ADC 模块中只有一个加法器。选择该模式时, 目标结果寄存器为 `ADC_RESRxL (x=0-3)` 和 `ADC_RESRxH (x=0-3)`,

见图 22-26。

在 ADC\_RESRxL (x=0-3) 中, 位 2-0 指示触发结果事件的通道编号, 位 7-3 返回滤波结果的位 4-0。在 ADC\_RESRxH (x=0-3) 中, 位 7-0 返回滤波结果的位 12-5。

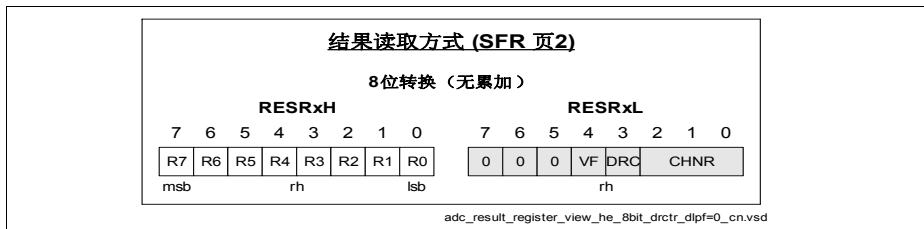


图 22-22 8 位, RCRx.DRCTR=0, RCRx.DLPF=0, 结果寄存器读取方式

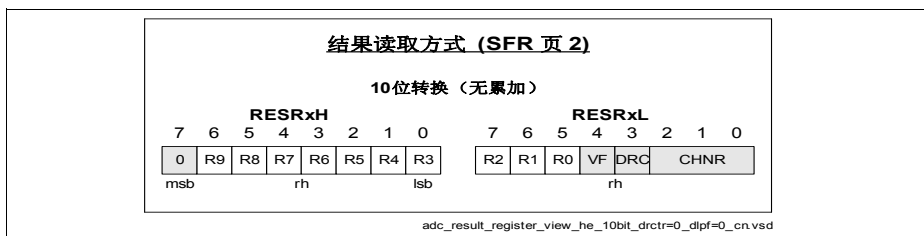


图 22-23 10 位, RCRx.DRCTR=0, RCRx.DLPF=0, 结果寄存器读取方式

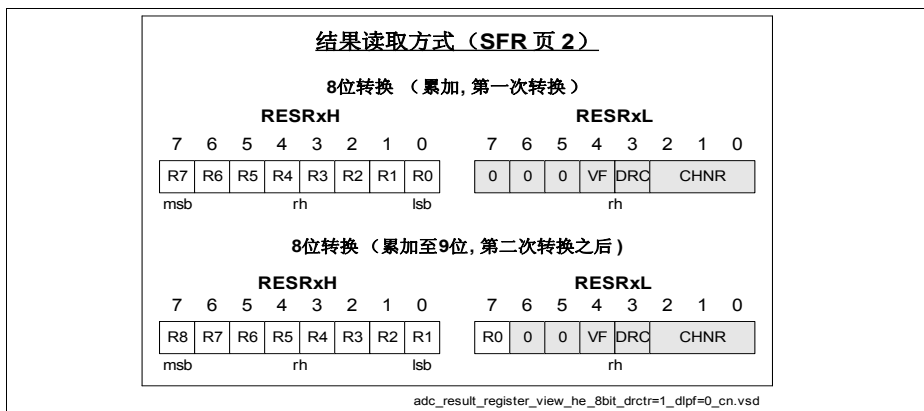
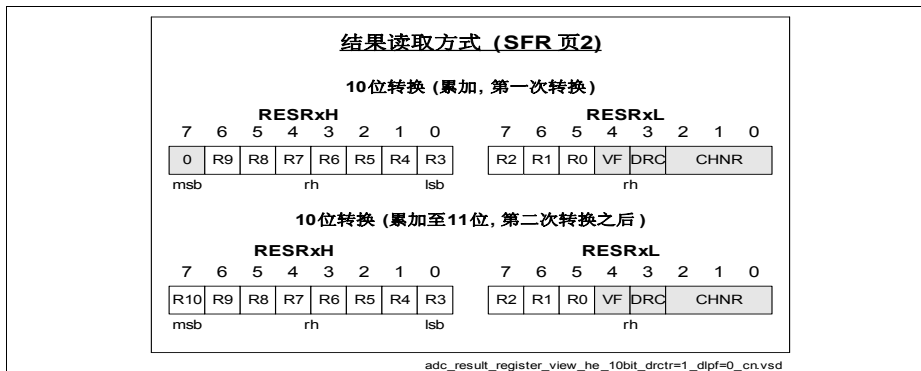
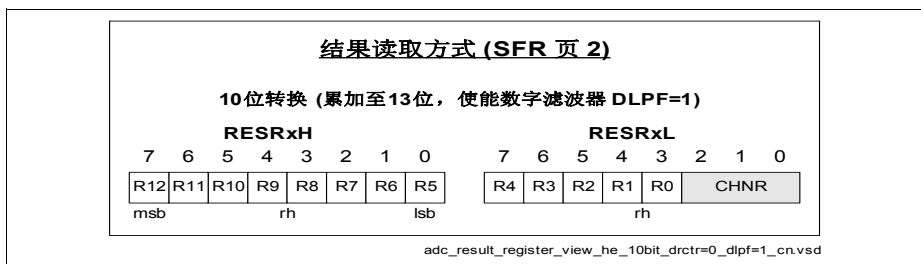


图 22-24 8 位, RCRx.DRCTR=1, RCRx.DLPF=0, 结果寄存器读取方式



**图 22-25      10 位, RCRx.DRCTR=1, RCRx.DLPF=0, 结果寄存器读取方式**



**图 22-26      10 位, RCRx.DRCTR=0, RCRx.DLPF=1, 结果寄存器读取方式**

所有转换结果保存在结果寄存器中, 结果寄存器中的内容在保存在同一个页地址上, 根据所选择的模式, 对齐数据。

## 模数转换器

标准读取方式由结果寄存器ADC\_RESRxL和ADC\_RESRxH组成，给出转换结果和相关通道编号。

读取寄存器RESRx时，对应的有效标志被清除（从应用的角度看）。

**ADC\_RESRxL (x = 0 - 2)**

结果寄存器 x 低位，标准 8 位或累加 8 位的第一次转换( $CA_H + x * 2$ ) 复位值: 00<sub>H</sub>

**ADC\_RESR3L**

结果寄存器 3 低位，标准 8 位或累加 8 位的第一次转换(D2<sub>H</sub>) 复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
0			VF	DRC	CHNR		
r			rh	rh	rh		

符号	位	类型	描述
CHNR	[2:0]	rh	<b>通道编号</b> 该位域包含最新更新寄存器的通道编号 <i>注意：位 2 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道，这些位应被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。</i>
DRC	3	rh	<b>数据压缩计数器</b> 该位指示要得到数据压缩的最终结果还需要累加多少个转换结果。 0 <sub>B</sub> 结果寄存器中的最终结果可用。当该位被置 0 时，有效标志被自动置位。 1 <sub>B</sub> 要得到最终结果，还需要累加一个转换结果。当该位被置 1 时，有效标志被自动复位。
VF	4	rh	<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。 0 <sub>B</sub> 结果寄存器 x 不包含有效数据 1 <sub>B</sub> 结果寄存器 x 包含有效数据
0	[7:5]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 模数转换器

**ADC\_RESRxH (x = 0 - 2)**

结果寄存器 x 高位，标准 8 位或累加 8 位的第一次转换( $CB_H + x * 2$ )

复位值：00<sub>H</sub>
**ADC\_RESR3H**

结果寄存器 3 高位，标准 8 位或累加 8 位的第一次转换(D3<sub>H</sub>)

复位值：00<sub>H</sub>
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
RESULT[7:0]							
rh							

符号	位	类型	描述
RESULT[7:0]	[7:0]	rh	转换结果 该位包含转换结果或数据压缩滤波的结果。

**ADC\_RESRxL (x = 0 - 2)**

结果寄存器 x 低位，标准读取 10 位转换( $CA_H + x * 2$ )

复位值：00<sub>H</sub>
**ADC\_RESR3L**

结果寄存器 3 低位，标准读取 10 位转换 (D2<sub>H</sub>)

复位值：00<sub>H</sub>
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
RESULT[2:0]			VF	DRC	CHNR		
rh			rh	rh	rh		

符号	位	类型	描述
CHNR	[2:0]	rh	通道编号 该位包含最新更新结果寄存器的通道编号。 <i>注意：位 2 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道，这些位应被当作保留位：类型为 ‘r’，读操作返回 0；应写入 0。</i>

**模数转换器**

符号	位	类型	描述
<b>DRC</b>	3	rh	<b>数据压缩计数器</b> 该位指示要得到数据压缩的最终结果还需要累加多少个转换结果。 $0_B$ 结果寄存器中的最终结果可用。当该位被置 0 时，有效标志被自动置位。 $1_B$ 要得到最终结果，还需要累加一个转换结果。当该位被置 1 时，有效标志被自动复位。
<b>VF</b>	4	rh	<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。 $0_B$ 结果寄存器 x 不包含有效数据 $1_B$ 结果寄存器 x 包含有效数据
<b>RESULT[2:0]</b>	[7:5]	rh	<b>转换结果</b> 该位域包含转换结果或数据压缩滤波的结果。

**ADC\_RESRxH (x = 0 - 2)**

 结果寄存器 x 高位，标准读取 10 位转换( $CB_H + x * 2$ )

 复位值:  $00_H$ 
**ADC\_RESR3H**

 结果寄存器 3 高位，标准读取 10 位转换 ( $D3_H$ )

 复位值:  $00_H$ 
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
0	RESULT[9:3]						
r	rh						

符号	位	类型	描述
<b>RESULT[9:3]</b>	[6:0]	rh	<b>转换结果</b> 该位域包含转换结果或数据压缩滤波的结果。
<b>0</b>	7	r	<b>保留</b> 读操作返回 0；应写入 0。

## 模数转换器

**ADC\_RESRxL (x = 0 - 2)**

结果寄存器 x 低位，累加读取 8 位的第二次转换( $CA_H + x * 2$ )

复位值: 00<sub>H</sub>
**ADC\_RESR3L**

结果寄存器 3 低位，累加读取 8 位的第二次转换(D2<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
R0	0	VF	DRC	CHNR			
rh	r	rh	rh	rh			

符号	位	类型	描述
CHNR	[2:0]	rh	<b>通道编号</b> 该位包含最新更新结果寄存器的通道编号。 <i>注意: 位 2 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道, 这些位应被当作保留位: 类型为 ‘r’, 读操作返回 0; 应写入 0。</i>
DRC	3	rh	<b>数据压缩计数器</b> 该位指示要得到数据压缩的最终结果还需要累加多少个转换结果。 0 <sub>B</sub> 结果寄存器中的最终结果可用。当该位被置 0 时, 有效标志被自动置位。 1 <sub>B</sub> 要得到最终结果, 还需要累加一个转换结果。当该位被置 1 时, 有效标志被自动复位。
VF	4	rh	<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。 0 <sub>B</sub> 结果寄存器 x 不包含有效数据 1 <sub>B</sub> 结果寄存器 x 包含有效数据
0	[6:5]	r	<b>保留</b> 读操作返回 0; 应写入 0。
RESULT[0]	7	rh	<b>转换结果</b> 该位域包含转换结果或数据压缩滤波的结果。



## 模数转换器

**ADC\_RESRxH (x = 0 - 2)**

结果寄存器 x 高位，累加读取 8 位的第二次转换( $CB_H + x * 2$ )

复位值: 00<sub>H</sub>
**ADC\_RESR3H**

结果寄存器 3 高位，累加读取 8 位的第二次转换( $D3_H$ )

复位值: 00<sub>H</sub>
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
RESULT[8:1]							
rh							

符号	位	类型	描述
RESULT[8:1]	[7:0]	rh	转换结果 该位域包含转换结果或数据压缩滤波的结果。

**ADC\_RESRxL (x = 0 - 2)**

结果寄存器 x 低位，累加读取 10 位的第一次转换( $CA_H + x * 2$ )

复位值: 00<sub>H</sub>
**ADC\_RESR3L**

结果寄存器 3 低位，累加读取 10 位的第一次转换( $D2_H$ )

复位值: 00<sub>H</sub>
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
RESULT[2:0]			VF	DRC	CHNR		
rh			rh	rh	rh		

符号	位	类型	描述
CHNR	[2:0]	rh	通道编号 该位包含最新更新结果寄存器的通道编号。 <i>注意: 位 2 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道, 这些位应被当作保留位: 类型为 ‘r’, 读操作返回 0; 应写入 0。</i>

**模数转换器**

符号	位	类型	描述
<b>DRC</b>	3	rh	<b>数据压缩计数器</b> 该位指示要得到数据压缩的最终结果还需要累加多少个转换结果。 $0_B$ 结果寄存器中的最终结果可用。当该位被置 0 时，有效标志被自动置位。 $1_B$ 要得到最终结果，还需要累加一个转换结果。当该位被置 1 时，有效标志被自动复位。
<b>VF</b>	4	rh	<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。 $0_B$ 结果寄存器 x 不包含有效数据 $1_B$ 结果寄存器 x 包含有效数据
<b>RESULT[2:0]</b>	[7:5]	rh	<b>转换结果</b> 该位域包含转换结果或数据压缩滤波的结果。

**ADC\_RESRxH (x = 0 - 2)**

 结果寄存器 x 高位，累加读取 10 位的第一次转换( $CB_H + x * 2$ )

 复位值: **00<sub>H</sub>**
**ADC\_RESR3H**

 结果寄存器 3 高位，累加读取 10 位的第一次转换( $D3_H$ )

 复位值: **00<sub>H</sub>**
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
<b>0</b>	<b>RESULT[9:3]</b>						
r	rh						

符号	位	类型	描述
<b>RESULT[9:3]</b>	[6:0]	rh	<b>转换结果</b> 该位域包含转换结果或数据压缩滤波的结果。
<b>0</b>	7	r	<b>保留</b> 读操作返回 0；应写入 0。

## 模数转换器

**ADC\_RESRxL (x = 0 - 2)**

结果寄存器 x 低位，累加读取 10 位的第二次转换( $CA_H + x * 2$ )

复位值: 00<sub>H</sub>
**ADC\_RESR3L**

结果寄存器 3 低位，累加读取 10 位的第二次转换(D2<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 2

7	6	5	4	3	2	1	0
RESULT[2:0]			VF	DRC	CHNR		
rh			rh	rh	rh		

符号	位	类型	描述
CHNR	[2:0]	rh	<b>通道编号</b> 该位包含最新更新结果寄存器的通道编号。 <i>注意: 位 2 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道, 这些位应被当作保留位: 类型为 ‘r’, 读操作返回 0; 应写入 0。</i>
DRC	3	rh	<b>数据压缩计数器</b> 该位指示要得到数据压缩的最终结果还需要累加多少个转换结果。 0 <sub>B</sub> 结果寄存器中的最终结果可用。当该位被置 0 时, 有效标志被自动置位。 1 <sub>B</sub> 要得到最终结果, 还需要累加一个转换结果。当该位被置 1 时, 有效标志被自动复位。
VF	4	rh	<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。 0 <sub>B</sub> 结果寄存器 x 不包含有效数据 1 <sub>B</sub> 结果寄存器 x 包含有效数据
RESULT[2:0]	[7:5]	rh	<b>转换结果</b> 该位域包含转换结果或数据压缩滤波的结果。

## 模数转换器

**ADC\_RESRxH (x = 0 - 2)**

结果寄存器 x 高位，累加读取 10 位的第二次转换( $CB_H + x * 2$ )

复位值: 00<sub>H</sub>
**ADC\_RESR3H**

结果寄存器 3 高位，累加读取 10 位的第二次转换( $D3_H$ )

复位值: 00<sub>H</sub>
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
RESULT[10:3]							
rh							

符号	位	类型	描述
RESULT[10:3]	[7:0]	rh	转换结果 该位域包含转换结果或数据压缩滤波的结果。

**ADC\_RESRxL (x = 0 - 2)**

结果寄存器 x 低位，低通滤波读取 10 位转换( $CA_H + x * 2$ )

复位值: 00<sub>H</sub>
**ADC\_RESR3L**

结果寄存器 3 低位，低通滤波读取 10 位转换( $D2_H$ )

复位值: 00<sub>H</sub>
**RMAP: 0, PAGE: 2**

7	6	5	4	3	2	1	0
RESULT[4:0]				CHNR			
rh				rh			

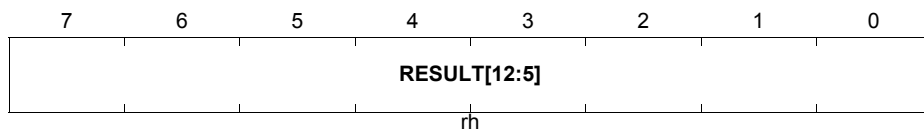
符号	位	类型	描述
CHNR	[2:0]	rh	通道编号 该位包含最新更新结果寄存器的通道编号。 <i>注意: 位 2 仅适用于具有 8 个 ADC 通道的器件。对于未实现的通道, 这些位应被当作保留位: 类型为 'r', 读操作返回 0; 应写入 0。</i>
RESULT[4:0]	[7:3]	rh	转换结果 该位域保存数字低通滤波的结果。

**模数转换器**
**ADC\_RESRxH (x = 0 - 2)**

 结果寄存器 x 高位, 低通滤波读取 10 位 ( $CB_H + x * 2$ )

 复位值: 00<sub>H</sub>
**ADC\_RESR3H**

 结果寄存器 3 高位, 低通滤波读取 10 位 ( $D3_H$ )

 复位值: 00<sub>H</sub>
**RMAP: 0, PAGE: 2**


符号	位	类型	描述
<b>RESULT[12:5]</b>	[7:0]	rh	<b>转换结果</b> 该位域保存数字低通滤波的结果。

结果控制寄存器控制结果寄存器的行为并监控其状态。

### ADC\_RCRx (x = 0 - 3)

结果控制寄存器 x

(CA<sub>H</sub> + x \* 1)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 4

7	6	5	4	3	2	1	0
<b>VFCTR</b>	<b>WFR</b>	<b>0</b>	<b>IEN</b>	<b>0</b>	<b>DLPF</b>	<b>0</b>	<b>DRCTR</b>
rw	rw	r	rw	r	rw	r	rw

符号	位	类型	描述
<b>DRCTR<sup>1)</sup></b>	0	rw	<b>数据压缩控制</b> 该位定义进行数据压缩需要累加多少个转换结果。 该位定义 DRC 的重载值。 0 <sub>B</sub> 禁止数据压缩滤波。DRC 的重载值为 0，因此转换结果不累加。 1 <sub>B</sub> 使能数据压缩滤波。DRC 的重载值为 1，因此由两个转换结果累加得到最终结果。
<b>DLPF<sup>1)</sup></b>	2	rw	<b>数字低通滤波器控制</b> 该位定义转换结果是否经过数字低通滤波处理。 0 <sub>B</sub> 禁止数字低通滤波器 1 <sub>B</sub> 使能数字低通滤波器，当该位第 1 次从 0 变为 1 时，先前的结果寄存器被清除。见 <a href="#">页 22-77</a>
<b>IEN</b>	4	rw	<b>中断使能</b> 该位使能与结果寄存器 x 相关的事件中断。当 DRC 置为 0 时，产生事件中断（递减之后或重载之后）。 0 <sub>B</sub> 禁止事件中断 1 <sub>B</sub> 使能事件中断
<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>VFCTR</b>	7	rw	<b>有效标志控制</b> 该位使能复位结果寄存器 x 的有效标志（通过读取高位字节）。 0 <sub>B</sub> 读访问 RESRxH/RESRAxH 时，VF 不变（缺省设置） 1 <sub>B</sub> 读访问 RESRxH/RESRAxH 时，VF 被复位

模数转换器

符号	位	类型	描述
0	1,3,5	r	保留 读操作返回 0；应写入 0。

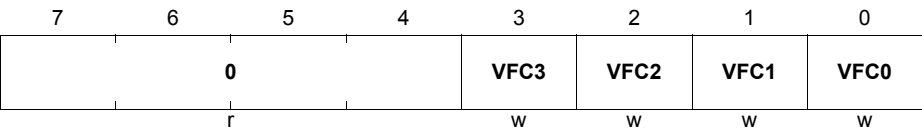
1) 10 位转换时，可使用数据压缩滤波或数字低通滤波。然而，因为只有一个加法器，因此不能同时开启这两种滤波器。因此同时使能这两个滤波器时，DRCTR=1 和 DLPF=1，结果与 DRCTR=0 且 DLPF=1，仅使能数字低通滤波相同。

8 位转换时，仅能使数据压缩滤波器。选择 8 位转换模式时，使能数字低通滤波器（DLPF=1）的结果与禁止数字低通滤波器（DLPF=0）的结果相同。

模数转换器

有效标志清除寄存器用于清除相应结果寄存器中的有效标志。

**ADC\_VFCR**  
有效标志清除寄存器 (CE<sub>H</sub>) 复位值: 00<sub>H</sub>  
RMAP: 0, PAGE: 4



符号	位	类型	描述
VFCx(x = 0 - 3)	x	w	清除结果寄存器 x 的有效标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 结果寄存器 RESRxL 中的 VF 被复位
0	[7:4]	r	保留 读操作返回 0；应写入 0。

22.10 等待 - 读取模式

在 CPU 读取先前的数据之前，等待 - 读取模式用来防止结果寄存器被新的转换结果覆盖。例如，自动扫描转换序列或其它对“时序要求不苛刻”的序列有可能使用同一个结果寄存器。然而，结果来自不同的输入通道，因此覆盖会破坏先前的转换结果<sup>1)</sup>。

等待 - 读取模式自动挂起该通道的转换，直到当前结果被读出才启动该通道的转换操作。因此可由硬件或软件触发来请求一次转换或转换序列，只有在先前的转换结果被读取之后，才启动下一次转换操作。根据 CPU 读取先前转换结果的能力（中断延迟）自动调整转换序列的启动。

如果结果寄存器使能等待 - 读取模式（相应结果控制寄存器中的位 WFR = 1），当目标结果寄存器包含有效数据时（由有效标志 VFx = 1 指示）或者当前运行的转换使用同一个结果寄存器时，则请求源不产生转换请求。

如果两个请求源使用同一个结果寄存器且选择了等待 - 读取模式，则高优先级的请求源不能中断在其请求转换之前已经启动的低优先级请求源。取消 - 插入 - 重复模式在此情况下不可用。如果高优先级请求使用另一个结果寄存器，则可取消低优先级的转换并在随后重复该转换。

1) 使用不同的结果寄存器的单个通道的重复转换不会破坏其它结果，而是更新它们先前的结果值。通过此方式，结果寄存器中始终是可用的、实际的信号值。



### 22.10.1 结果事件和中断

根据图 22-27 的结构，由结果事件产生结果事件中断。如果检测到结果事件，置位寄存器 **ADC\_EVINFR** 中的指示标志。还可通过向 **ADC\_EVINSR** 中的对应位写 1 的方式置位这些标志，写 0 无效。可通过向寄存器 **ADC\_EVINCR** 写 1 的方式软件清除这些指示标志。

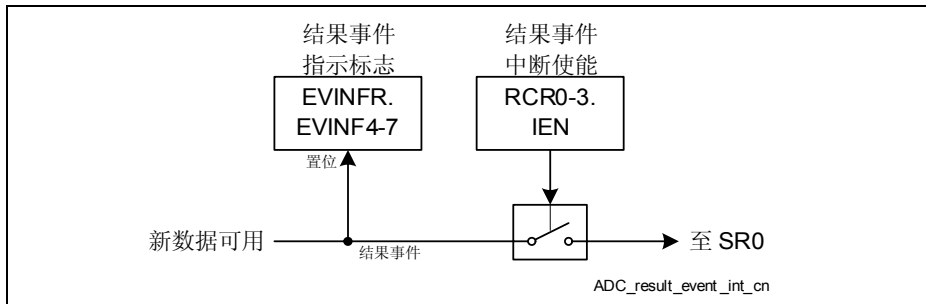


图 22-27 结果事件中断产生

请求源事件中断和结果事件中断共用同一个服务请求输出线 **SR0**。

结果事件和请求源事件共用同一组寄存器。结果事件位于 **ADC\_EVINFR** 的下列位置：

- 事件 4：结果寄存器 0 的结果事件
- 事件 5：结果寄存器 1 的结果事件
- ...
- 事件 7：结果寄存器 3 的结果事件

### 22.10.2 数据压缩和滤波

在产生结果中断之前，数据压缩自动累加一组转换结果。采用此方式可去除一些来自输入信号的噪声并减少 CPU 因读取 **ADC** 转换数据所产生的负荷。

标准数据压缩模式可累加任意结果寄存器中的结果值。

数字低通滤波器通过平均先前结果和当前结果的方式去除测量到的瞬态噪声电压，对结果值进行预处理。

#### 标准数据压缩模式

数据压缩模式可用作数字滤波器，进行抗混叠或数据抽取用途。累加最多两个转换结果并产生最终结果。

每个结果寄存器可单独使其数据压缩特性，由寄存器 **ADC\_RCRx (x = 0 - 3)** 中的 **DRCTR** 进行控制。数据压缩计数器指示累加操作的实际状态。

注：累加的转换操作之间可插入目标为其它结果寄存器的转换操作。

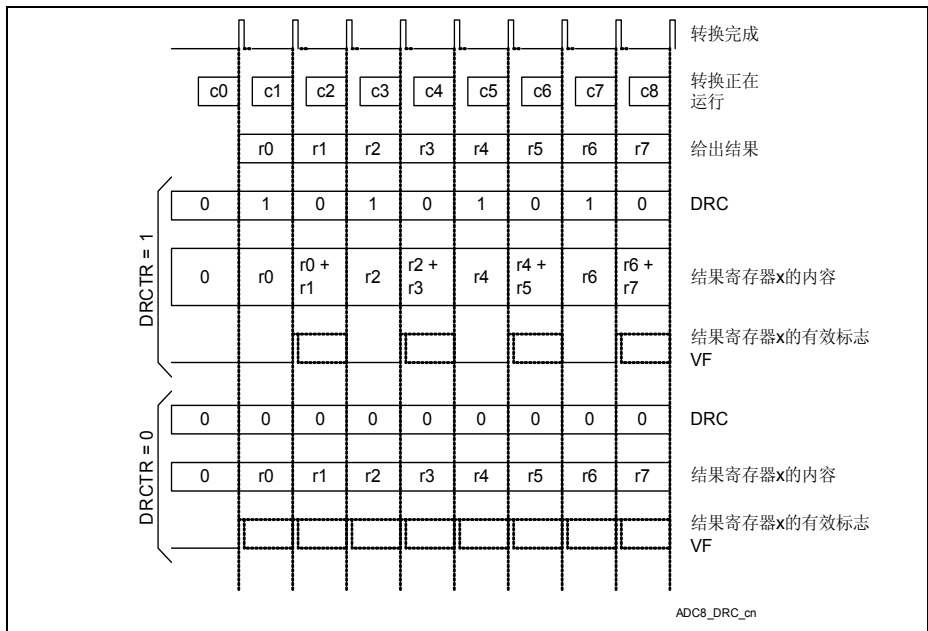


图 22-28 标准数据压缩滤波器

本例给出 2 个累加转换结果的数据压缩序列。8 个转换结果（r0 ... r7）累加之后得到 4 个最终结果。

转换完成时，数据被保存到结果寄存器中，如果使能数据压缩模式，由数据压缩计数器 DRC 控制数据处理：

- 如果 DRC = 0 （例中的 t3, t5, t7, t9），转换结果被保存到寄存器中。用位域 RCRx.DRCR 中的内容加载 DRC。（即累加开始）
- 如果 DRC > 0 （例中的 t2, t4, t6, t8），转换结果和结果寄存器中的值相加 DRC 减 1
- 如果 DRC 变为 0，不管是从 1 减为 0（例中的 t2, t4, t6, t8）还是从 DRCR 加载的，相应结果寄存器中的有效位被置位，且发生结果寄存器事件。最终结果必须在下一个数据压缩序列启动之前（例中 t3, t5, t7 或 t9 之前）被读取。读取结果时自动清除有效标志。

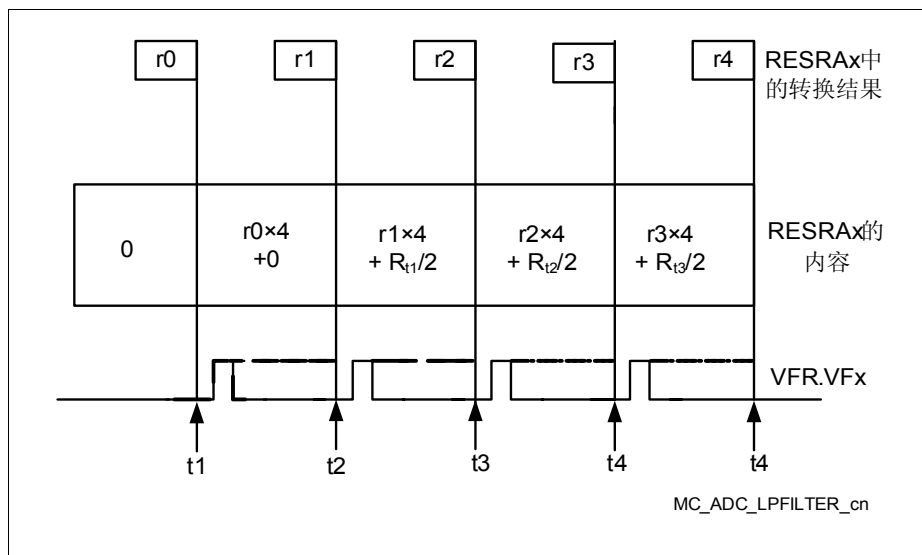
### 数字低通滤波模式

数据压缩逻辑还可以构成一个低通滤波器，将放大的结果值（因子 4）与一个缩小的先前值（因子为 0.5）相加。

# 模数转换器

每个结果寄存器可单独使能低通滤波模式，由寄存器 **ADC\_RCRx (x = 0 - 3)** 中的 LPFEN 控制。此时不使用数据压缩计数器 DRC。

注意：对于低通滤波模式，结果数据必须来自一个专用的通道。



**图 22-29 低通滤波器操作**

每个结果乘以 4 并和先前寄存器值的 1/2 相加。相加之后有效标志被激活。

## 22.11 中断请求处理

可由几种类型的事件产生中断。每个 ADC 内核提供两个连接至中断节点的服务请求输出信号（ADCx\_SR[1:0]）。四类事件可产生中断请求：

- **请求源事件：**指示完成了请求源所请求的转换序列。对于扫描源来讲，完成仲裁之后的一组预定的通道转换之后，产生中断事件。对于顺序请求源来讲，用户可决定在转换序列中的何处产生请求源事件。  
请求源事件指示转换序列已经达到一个预定的状态，软件可访问相关的一组结果。
- **通道事件：**指示转换已经完成。可选择，仅在转换结果在编程定义的范围内才产生通道事件。  
通道事件尤其可用来指示模拟输入值在额定工作范围之内或之外。从而可减少 CPU 的背景任务负荷，即如果满足或超出特定转换结果范围时才需要产生中断。
- **结果事件：**指示结果寄存器中有一个新的有效结果。通常，该事件触发 CPU 的读操作。可选择是否使能数据压缩或数字低通滤波器，如使能则以较低的速率产生结果事件。
- **超出范围比较器事件：**指示在模拟输入通道上检测到的电压高于或低于 Vddp。

每个 ADC 事件有专用的标志指示且可由软件清除。如果使能某事件中断，每次发生该事件都会产生中断，与相应的指示标志无关。此方式可确保高效的处理 ADC 事件（无需清除指示标志，ADC 事件也可产生中断）。

请求源事件，通道事件和结果事件共用一个服务请求输出 ADC\_SR0，超出范围比较器事件使用服务请求输出 ADC\_SR1。

*注：如果全部被使能，一次转换可能引发三个中断，每个中断一种类型。*

*在此情况下，ADC 模块首先触发请求源事件中断，然后是通道事件中断，紧接着是结果事件中断（所有中断都发生在数个  $f_{\text{ADC}}$  时钟周期内）。*

模数转换器

事件指示标志寄存器 ADC\_EVINFR 监控检测到的请求源事件（标志 EVINF0-EVINF1）和结果事件（标志 EVINF4-EVINF7）。

**ADC\_EVINFR**  
**事件中断标志寄存器** (CE<sub>H</sub>) 复位值: 00<sub>H</sub>  
**RMAP: 0, PAGE: 5**

7	6	5	4	3	2	1	0
<b>EVINF7</b>	<b>EVINF6</b>	<b>EVINF5</b>	<b>EVINF4</b>	<b>0</b>		<b>EVINF1</b>	<b>EVINF0</b>
rh	rh	rh	rh	r		rh	rh

符号	位	类型	描述
<b>EVINF0, EVINF1, EVINF4, EVINF5, EVINF6, EVINF7</b>	0, 1, 4, 5, 6, 7	rh	<b>事件中断 x 的标志</b> 该位监控事件中断 x 的状态 0 <sub>B</sub> 事件中断 x 未发生。写 0 清除该寄存器 1 <sub>B</sub> 事件中断 x 已发生。写 1 置位该寄存器并产生中断脉冲（如被使能）
<b>0</b>	[3:2]	r	<b>保留</b> 读操作返回 0；应写入 0。

模数转换器

当事件中断置位标志寄存器 **ADC\_EVINSR** 中的某位被写入时，则置位 **ADC\_EVINFR** 中的对应位并产生中断请求。

**ADC\_EVINSR**  
**事件中断清除标志寄存器** (D2<sub>H</sub>) 复位值: 00<sub>H</sub>  
**RMAP: 0, PAGE: 5**

7	6	5	4	3	2	1	0
EVINS7	EVINS6	EVINS5	EVINS4	0		EVINS1	EVINS0
w	w	w	w	r		w	w

符号	位	类型	描述
EVINS0, EVINS1, EVINS4, EVINS5, EVINS6, EVINS7	0, 1, 4, 5, 6, 7	w	置位事件 x 中断标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 EVINFR.x 被置位
0	[3:2]	r	保留 读操作返回 0；应写入 0。

注： 向事件中断置位标志寄存器 **ADC\_EVINSR** 写 1 则置位 **ADC\_EVINFR** 中对应位并产生中断请求，写 0 无效。

模数转换器

向事件指示位清除寄存器 **ADC\_EVINCR** 写 1 会清除寄存器 **ADC\_EVINFR** 中的对应事件指示标志 **EVINF<sub>x</sub>**。当相应位置被写 1 时，如果检测到请求源或结果事件，标志 **EVINF<sub>x</sub>** 被清除。

**ADC\_EVINCR**  
 事件中断清除标志寄存器 (CF<sub>H</sub>) 复位值: 00<sub>H</sub>  
 RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
EVINC7	EVINC6	EVINC5	EVINC4	0		EVINC1	EVINC0
W	W	W	W	r		W	W

符号	位	类型	描述
EVINC0, EVINC1, EVINC4, EVINC5, EVINC6, EVINC7	0, 1, 4, 5, 6, 7	w	清除事件 x 的中断标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 EVINFR.x 复位
0	[3:2]	r	保留 读操作返回 0；应写入 0。

通道事件指示标志寄存器 CHINFR 监控检测到的通道 0 ... 7 的通道事件。

**ADC\_CHINFR**  
 通道中断标志寄存器  
 RMAP: 0, PAGE: 5

(CA<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>CHINF7</b>	<b>CHINF6</b>	<b>CHINF5</b>	<b>CHINF4</b>	<b>CHINF3</b>	<b>CHINF2</b>	<b>CHINF1</b>	<b>CHINF0</b>
rh	rh	rh	rh	rh	rh	rh	rh

符号	位	类型	描述
<b>CHINF<sub>x</sub></b> <b>(x = 0 - 7)</b>	x	rh	通道 <b>x</b> 的中断标志 该位监控通道 <b>x</b> 中断的状态 0 <sub>B</sub> 通道 <b>x</b> 未发生通道中断 1 <sub>B</sub> 通道 <b>x</b> 已发生通道中断 <i>注意: 位 4-7 仅适用于具有 8 个 ADC 通道的器件。                      对于未实现的通道, 这些位应被当作保留位: 类型为 ‘r’, 读操作返回 0; 应写入 0。</i>



## 模数转换器

向 CHINSR 寄存器中的某位写 1 会置位相应的中断请求位并产生中断请求。写 0 无效。

## ADC\_CHINSR

通道中断置位寄存器

(CC<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
CHINS7	CHINS6	CHINS5	CHINS4	CHINS3	CHINS2	CHINS1	CHINS0
W	W	W	W	W	W	W	W

符号	位	类型	描述
<b>CHINSx</b> (x = 0 - 7)	x	w	<b>通道 x 中断标志置位</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位 CHINFR.x 并产生中断脉冲 <i>注意: 位 4-7 仅适用于具有 8 个 ADC 通道的器件。            对于未实现的通道, 这些位应被当作保留位: 类型为 ‘r’, 读操作返回 0; 应写入 0。</i>

模数转换器

向寄存器 CHINCR 中的某位写 1 则清除寄存器 **ADC\_CHINFR** 中的对应通道事件指示标志 CHINFx。当相应位置被写 1 时，如果检测到通道事件，标志 CHINFx 被清除。

**ADC\_CHINCR**

通道中断清除寄存器

(CB<sub>H</sub>)

复位值: 00<sub>H</sub>

RMAP: 0, PAGE: 5

7	6	5	4	3	2	1	0
<b>CHINC7</b>	<b>CHINC6</b>	<b>CHINC5</b>	<b>CHINC4</b>	<b>CHINC3</b>	<b>CHINC2</b>	<b>CHINC1</b>	<b>CHINC0</b>
W	W	W	W	W	W	W	W

符号	位	类型	描述
<b>CHINCx</b> (x = 0 - 7)	x	W	清除通道 x 的中断标志 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 CHINFR.x 复位 <i>注意: 位 4-7 仅适用于具有 8 个 ADC 通道的器件。 对于未实现的通道, 这些位应被当作保留位: 类型为 'r', 读操作返回 0; 应写入 0。</i>

## 22.12 寄存器映射

ADC SFR 位于标准存储器区 (RMAP = 0)，由 6 页组成。ADC\_PAGE 寄存器的地址为 D1<sub>H</sub>，包含分页值和页控制信息。

**ADC\_PAGE**  
**ADC 分页寄存器** (F1<sub>H</sub>) 复位值: 00<sub>H</sub>  
**RMAP: 0, PAGE: X**

7	6	5	4	3	2	1	0
OP		STNR		0	PAGE		
w		w		r	rwh		

符号	位	类型	描述
PAGE	[3:0]	rwh	<b>分页位</b> 写入时，该值表示新页的值。读出时，该值指示当前有效页的值 = addr [y:x+1]
STNR	[5:4]	w	<b>保存编号</b> 该编号指明在哪个保存位域上执行 OP 定义的操作。 若 OP = 10 <sub>B</sub> ， PAGE 的内容在被新值覆盖之前保存在 ADC_STx 中 若 OP = 11 <sub>B</sub> ， PAGE 的内容被 ADC_STx 的内容覆盖。写入 PAGE 的值不予理睬 00 选择 ADC_ST0 01 选择 ADC_ST1 10 选择 ADC_ST2 11 选择 ADC_ST3
OP	[7:6]	w	<b>操作</b> 0X 手动保存页模式，STNR 的值被忽略，PAGE 被直接写入 10 带有自动页保存的新页设置。当前写入 PAGE 中的内容被保存的同时，上次写入 PAGE 的内容被保存在 STNR 规定的位域 ADC_STx 中 11 自动恢复页。对写入 PAGE 的内容不予理睬，PAGE 由 STNR 规定的位域 ADC_STx 中的内容覆盖。

内核 SFR 的地址列于表 22-7 和表 22-8 中。

本章中描述的所有 ADC 寄存器名称，在本手册其他章节中引用时需加上模块名前缀 "ADC\_"，例如 ADC\_GLOBCTR。

**表 22-7 页 0 – 3 的 SFR 地址列表**

地址	页 0	页 1	页 2	页 3
CAH	<a href="#">ADC_GLOBCTR</a>	CHCTR0	RESR0L	
CBH	<a href="#">ADC_GLOBSTR</a>	CHCTR1	RESR0H	
CCH	<a href="#">ADC_PRAR</a>	CHCTR2	RESR1L	
CDH	<a href="#">ADC_LCBR0</a>	CHCTR3	RESR1H	
CEH	<a href="#">ADC_INPCR0</a>	CHCTR4	RESR2L	
CFH	<a href="#">ADC_LCBR1</a>	CHCTR5	RESR2H	
D2H	<a href="#">ADC_LORE</a>	CHCTR6	RESR3L	
D3H	<a href="#">ADC_ENORC</a>	CHCTR7	RESR3H	

**表 22-8 页 4 – 7 的 SFR 地址列表**

地址	页 4	页 5	页 6	页 7
CAH	RCR0	<a href="#">ADC_CHINFR</a>	<a href="#">ADC_CRCCR1</a>	
CBH	RCR1	<a href="#">ADC_CHINCR</a>	<a href="#">ADC_CRPR1</a>	
CCH	RCR2	<a href="#">ADC_CHINSR</a>	<a href="#">ADC_CRMRI</a>	
CDH	RCR3		<a href="#">ADC_QMR0</a>	
CEH	<a href="#">ADC_VFRCR</a>	<a href="#">ADC_EVINFR</a>	<a href="#">ADC_QSR0</a>	
CFH	<a href="#">ADC_ALR0</a>	<a href="#">ADC_EVINCR</a>	<a href="#">ADC_Q0R0</a>	
D2H	<a href="#">ADC_CNF</a>	<a href="#">ADC_EVINSR</a>	<a href="#">ADC_QBUR0/ ADC_QINR0</a>	
D3H	<a href="#">ADC_ETRCR</a>			

## 23 Boot ROM 用户子程序

Boot ROM 用户子程序提供一组非常有用的、可供用户应用调用的子程序。[表 23-1](#) 给出用户应用能够调用的子程序列表，这些子程序是由英飞凌开发的并且作为 Boot ROM 功能的组成部分。

**表 23-1      Boot ROM 用户子程序列表**

地址	名称	描述
0xDFDE	<a href="#">BR_FLASH_READ_MODE_STATUS</a>	检查选中的 Flash Bank 的读模式状态
0xDFE1	<a href="#">BR_GET_4_BYTES_INFO</a>	读取 4 字节的芯片 ID 编号和用户 ID 编号 (USER_ID)
0xDFE4	<a href="#">BR_PROG_USER_ID</a>	编程设置 4 字节 USER_ID (BR_FEATURE_SETTING 的选项 1)
0xDFE4	<a href="#">BR_CLKMODE_SETTING</a>	初始化 CLKMODE (BR_FEATURE_SETTING 的选项 0)
0xDFE7	<a href="#">BR_AUTO_BAUD</a>	自动检测 UART 波特率
0xDFEA	<a href="#">BR_UART_BSL</a>	重新进入 UART BSL 模式
0xDFED	<a href="#">BR_FLASH_BACKGROUND_PROGRAM</a>	后台编程代码 / 数据
0xDFF0	<a href="#">BR_FLASH_BACKGROUND_ERASE</a>	后台擦除 Flash
0xDFF3	<a href="#">BR_FLASH_BACKGROUND_ERASE_ABORT</a>	Flash 擦除中止 (后台)
0xDFF6	<a href="#">BR_FLASH_PROGRAM</a>	将代码 / 数据编程到 Flash 中
0xDFF9	<a href="#">BR_FLASH_ERASE</a>	擦除 Flash

可从 XRAM 或 Flash 中调用擦除和编程子程序。[表 23-2](#) 描述如何使用和调用这些子程序。

**表 23-2      调用 Flash 用户子程序**

名称	器件	从何处调用	目标
<a href="#">BR_FLASH_PROGRAM</a>  <a href="#">BR_FLASH_ERASE</a>	XC83x - 8K	XRAM	Flash Bank 0 或 Flash Bank 1
		Flash Bank 0/1	Flash Bank 0/1
	XC83x - 4K	Flash Bank 0	Flash Bank 0
		XRAM	Flash Bank 0

Boot ROM 用户子程序

表 23-2      调用 Flash 用户子程序 (cont'd)

名称	器件	从何处调用	目标
BR_FLASH_ERASE	XC83x - 8K	XRAM 或 Flash Bank 0/1	同时对 Flash Bank 0 和 Flash Bank 1 进行操作
BR_FLASH_BACKGROUND_PROGRAM  BR_FLASH_BACKGROUND_ERASE	XC83x - 8K	XRAM	Flash Bank 0 或 Flash Bank 1
		Flash Bank 0	Flash Bank 1
		Flash Bank 1	Flash Bank 0
BR_FLASH_BACKGROUND_ERASE_ABORT	XC83x - 4K	XRAM	Flash Bank 0

23.1      Flash Bank 读模式状态子程序

读模式状态子程序允许检查 Flash Bank 的准备好 - 可供读取的状态。当用户使用 BR\_FLASH\_BACKGROUND\_ERASE\_ABORT 子程序时，该读模式状态子程序会非常有用。调用擦除中止子程序之后，用户可调用该读模式状态子程序，用来检查擦除操作是否被成功中止且 Flash 是否已经进入读模式。

调用该子程序之前，用户必须确保将输入 R7 配置为所选择的 Flash Bank。当 Flash 处于读模式时，借位标志被清除；否则，借位标志被置位。对于错误的输入，借位标志也被置位。

表 23-3      Flash Bank 读模式状态子程序说明

子程序	BR_FLASH_READ_MODE_STATUS
输入	当前寄存器组的 R7: 所选择的 Flash Bank 00 <sub>H</sub> Flash Bank 0 01 <sub>H</sub> Flash Bank 1 <sup>1)</sup> 其它: 保留 (无效选项)
输出	C = 0: 所选择的 Flash Bank 处于读模式 C = 1: 所选择的 Flash Bank 不处于读模式 C = 1:I 所选择的选项无效
所需堆栈大小	6
使用 / 破坏的资源	A

1) 仅适用于 XC83x - 8K 器件。对于其它器件配置，视其为无效选项。

### 23.2 获得 4 字节信息

该子程序允许工具链软件或者甚至是用户代码读出器件的芯片 ID 编号（见[章节 1.3](#)）或用户 ID 编号（USER\_ID；见[章节 5.1](#)）。

**表 23-4 BR\_GET\_4\_BYTES\_INFO 子程序说明**

子程序	BR_GET_4_BYTES_INFO
输入	<p>当前寄存器组的 R7：选项  00<sub>H</sub> 选项 0 芯片 ID 编号  01<sub>H</sub> 选项 1 用户 ID 编号（USER_ID）  其它：保留（无效选项）</p> <p>当前寄存器组的 R5：  4 字节 IRAM 缓存指针</p>
输出	<p>选项 0 对应的 IRAM 缓存填充方式：  [R5]: 芯片 ID 编号（MSB）  [R5+1]: 芯片 ID 编号  [R5+2]: 芯片 ID 编号  [R5+3]: 芯片 ID 编号（LSB）</p> <p>选项 1 对应的 IRAM 缓存填充方式：  [R5]: USER_ID 编号（MSB）  [R5+1]: USER_ID 编号  [R5+2]: USER_ID 编号  [R5+3]: USER_ID 编号（LSB）</p> <p>无效选项对应的 IRAM 缓存填充方式：  [R5]: 0x00  [R5+1]: 0x00  [R5+2]: 0x00  [R5+3]: 0x00</p> <p>C = 0；读取成功  C = 1；读取不成功（选项无效）</p>
所需缓存大小	8
使用 / 破坏的资源	<p>A</p> <p>当前寄存器组的 R1 和 R7（2 字节）</p>

### 23.3 特性设置子程序

Boot ROM 中的特性设置子程序用于初始化一些设置。当前仅提供两个选项：CLKMODE 设置（BR\_CLKMODE\_SETTING）和编程 USER\_ID（BR\_PROG\_USER\_ID）。

Boot ROM 用户子程序

CLKMODE 设置子程序将频率设置为 8 MHz 或 24 MHz，而编程 USER\_ID 子程序将 4 字节用户 ID (USER\_ID) 信息编程到器件中。这 4 字节用户 ID 编号将由：BMI，BMI（用于确定 Boot 模式）和一些如时钟频率设置和被使能的外设等的初始化信息。一旦编程完成，该 BR\_PROG\_USER\_ID 子程序将发出一个软件复位。因此除了器件复位之外，没有其它的操作成功的指示，且器件将从编程设置的启动模式开始启动。该 BR\_PROG\_USER\_ID 子程序将检查 NMISR 是否为 0x00。如果是，该子程序将清除 NMICON 和 EA 并继续编程 USER\_ID。调用该子程序之后无法返回，因此在调用该程序之前，用户应当留意看门狗服务程序并禁止 WDT。

表 23-5 CLKMODE 设置子程序

子程序	BR_CLKMODE_SETTING (BR_FEATURE_SETTING 一选项 0)
输入	当前寄存器组的 R7: 选项 00 <sub>H</sub> 选项 0 - BR_CLKMODE_SETTING 其它: 保留 (无效选项)
	当前寄存器组的 R5: 00 - 8 MHz 80 - 24 MHz 其它 —保留
输出	C = 0: 选择了正确选项，相应的设置 CLK 模式 C = 1:l 选择了无效选项
所需要的堆栈大小	9
使用 / 破坏的资源	A

表 23-6 编程 USER\_ID 子程序的说明说明

子程序	BR_PROG_USER_ID (BR_FEATURE_SETTING 一选项 1)
-----	---



表 23-6      编程 USER\_ID 子程序的说明说明

输入	当前寄存组的 R7: 选项 01 <sub>H</sub> 选项 1-BR_PROG_USER_ID 其它: 保留 (无效选项)
	当前寄存器组的 R5: 用来存放 4 字节用户 ID 编号的 IRAM 起始地址 [R5]: USER_ID_3 [R5+1]:USER_ID_2 [R5+2]:USER_ID_1 [R5+3]:USER_ID_0
	4 字节用户 ID 编号 (IRAM 中的)
	SFR NMISR = 00 <sub>H</sub>
	堆栈指针 (SP) 设置: 0x07 <= SP <= 0x60 或者 0xC0 <= SP <= 0xE0
输出 <sup>1)</sup>	C = 1: 编程用户 ID 失败 C = 1: 由于 NMISR 非 00 <sub>H</sub> , 退出子程序—错误
所需堆栈大小	10
使用 / 破坏的资源	DPTR, A
	当前寄存器组的 R0, R1, R3, R5, R6 和 R7 (6 字节)
	IRAM 地址 0x80 - 0xBF (64 字节)

1) 只有当编程 USER\_ID 失败之后, 才退出子程序并可观察到输出。成功编程 USER\_ID 时, 将触发一个软件复位并按照 BMI 编程的启动模式启动。

### 23.4      UART 自动波特率子程序

从主机接收到一个同步字节之后, 该子程序允许由应用软件配置 UART 设置。一旦调用该子程序, 则接收同步字节、波特率计算和发送回应字节过程中用户应用将停滞。自动波特率的协议描述见 [图 6-1](#), 自动波特率所使用的端口引脚的定义见 [表 6-1](#)。

用户需要注意所使用的时钟频率和并根据所需要的自动波特率来编程 BCON.BGSEL SFR 位域。

- 注: 该子程序不修改时钟频率和 BGSEL SFR 位域。可在调用该子程序之前, 根据所需要的波特率和时钟频率修改这些位域。
- 注: 一旦进入到该用户子程序, 设置 SPD 的操作不被禁止。用户必须确保 SPD 和自动波特率所使用的端口引脚不会发生冲突。
- 注: 为了不干扰自动波特率的检测和计算, 强烈建议用户在调用该子程序之前, 禁止所有中断。

表 23-7 BR\_AUTO\_BAUD 子程序说明

子程序	BR_AUTO_BAUD
输入	相应的设置 BCON.BGSEL SFR 位域
输出	该子程序配置端口引脚（至输入 / 输出，RXD/TXD）并根据接收到的 005555 <sub>H</sub> ，或 0055AA <sub>H</sub> 自动波特率值配置 UART 波特率。随后，如果成功完成初始化，该子程序发送回应符号 55 <sub>H</sub> （应答码）。
所需堆栈大小	11
使用的 / 破坏的资源	端口相关的 SFR, A, SBUF, SCON, MODPISEL1, MODPISEL2, TCON, TMOD, TH0, TL0, TR0, BCON, LINST, T2_T2CON, T2_RC2H/L, T2_T2MOD, T2_T2H/L IRAM 地址 0x7F

## 23.5 UART BSL 子程序

该子程序允许应用软件从各种模式，如编程、擦除 XRAM、擦除 Flash 等模式中跳回到 UART BSL 模式。在该子程序中将完成自动波特率检测。用户需要留意所使用的时钟频率并根据需要的自动波特率设置 BCON.BGSEL SFR 位域。UART\_BSL 模式特性的详细信息，请参考引导程序加载器一章。

**注：** 该子程序不修改时钟频率和 BGSEL SFR 位域。在调用该子程序之前，需要由用户根据需要的波特率和时钟频率修改这些位域。

**注：** 由于该子程序始终等待任何的 UART BSL 报文头，因此该程序不具备返回能力，因此需要使用一个 LJMP 指令（而不是 LCALL）。

**注：** 进入到该子程序之后，SPD 设置被禁止。且一旦 SPD 被禁止，不能再次被使能，除非通过复位的方式来解决。

表 23-8 BR\_UART\_BSL 子程序说明

子程序	BR_UART_BSL
输入	IEN0.EA = 0 NMICON = 0x00 相应的设置 BCON.BGSEL SFR 位域 堆栈指针（SP）设置： 0x07 <= SP <= 0x60 或者 SP = 0xE0
输出	-

表 23-8 BR\_UART\_BSL 子程序说明

所需堆栈大小	-
使用 / 破坏的资源	-

注： 根据选择的以及执行的 UART BSL 模式，部分 IRAM 地址上的内容可能被破坏。

### 23.6 Flash 编程子程序

每次调用 Flash 编程子程序允许编程：

- 将 32 个数据字节（WL 对齐）写入到所选择的 Flash 字线
- 将 32 个数据字节（WL 对齐）写入到所选择的用户 BSL Flash 扇区字线<sup>1)</sup>

调用该子程序之前，用户必须确保要编程的 Flash 内容已经被缓存到同样大小的 IRAM 缓存中。为用户两种类型的 Flash 编程子程序。可从 XRAM 或 Flash 中调用擦除和编程子程序。表 23-2 描述如何使用和调用这些子程序。

第一类子程序（不支持后台编程）**BR\_FLASH\_PROGRAM**：将一直等待，直到编程操作完成，才允许继续执行用户程序。对于正在执行的用户代码保存在想要编程的 Flash Bank 的情况来说，该类子程序对用户来说非常必要，尤其是对于仅有一个 Flash bank 的情况来讲。Flash 不能同时处于编程模式和读模式。对于用户想要编程的 Flash bank 还用于保存中断向量的情况，Flash Bank 处于编程模式时，中断处理器无法访问该 Flash Bank，因此无法处理中断，此时该类子程序也非常有用。对于第一种类型的编程子程序，Boot ROM 将用来控制代码，FNMIFLASH 标志被自动清除且因此无需用户处理该标志。

表 23-9 FLASH\_PROGRAM 子程序说明

子程序	BR_FLASH_PROGRAM
输入	想要编程的 Flash WL 对齐地址 当前寄存器组的 R6: DPH 当前寄存器组的 R7: DPL <sup>1)</sup>
	当前寄存器组的 R5: 存放 32 字节 Flash 数据的 IRAM 起始地址
	32 字节 Flash 数据
	所有中断，包括 NMI 在内必须被禁止（IEN0.EA=0，NMICON=00 <sub>H</sub> ）
	SFR NMISR = 00 <sub>H</sub>

1) 仅 XC83x - 8K 器件支持该操作。

## Boot ROM 用户子程序

**表 23-9 FLASH\_PROGRAM 子程序说明 (cont'd)**

输出	PSW.CY: 0 = Flash 编程成功 1 = Flash 编程不成功
	DPTR 增加 20 <sub>H</sub> <sup>2)</sup>
所需要的堆栈大小	12
使用的 / 破坏的资源	DPTR, A
	当前寄存器组的 R0, R2, R6 和 R7 (4 字节)

1) 对于对齐的字线地址, DPL 的低 5 位为 0, 例如: 00<sub>H</sub>, 20<sub>H</sub>, 40<sub>H</sub>, 60<sub>H</sub>, 80<sub>H</sub>, A0<sub>H</sub>, C0<sub>H</sub> 和 E0<sub>H</sub>。

2) 当 PSW.CY 为 0 时, DPTR 仅增加 20<sub>H</sub>。

第二类子程序 (支持后台编程) **BR\_FLASH\_BACKGROUND\_PROGRAM**, 允许在调用编程子程序之后, 继续执行用户程序代码。用户将一直等待, 直到产生 Flash NMI 事件; 寄存器 NMISR 中的位 FNMIFLASH 置位, 且如果由 NMIFLASH 使能, 为了进入 Flash NMI 服务子程序, 触发一个发送给 CPU 的 NMI。此时 Flash bank 处于准备好 - 可供读取模式, 即编程操作已经完成。

**表 23-10 FLASH\_BACKGROUND\_PROGRAM 子程序说明**

子程序	BR_FLASH_BACKGROUND_PROGRAM
输入	想要编程的 Flash WL 对齐地址: 当前寄存器组的 R6: DPH 当前寄存器组的 R7: DPL <sup>1)</sup>
	当前寄存器组的 R5: 存放 32 字节 Flash 数据的 IRAM 起始地址
	32 字节 Flash 数据
	Flash NMI (NMICON.NMIFLASH) 被使能 (1) 或禁止 (0)
	SFR NMISR = 00 <sub>H</sub>
输出	PSW.CY: 0 = 正在进行 Flash 编程 1 = Flash 编程不成功
	DPTR 增加 20 <sub>H</sub> <sup>2)</sup>
所需堆栈大小	8
使用的 / 破坏的资源	DPTR, A
	当前寄存器组的 R0, R2, R6 和 R7 (4 字节)

1) 对于对齐的字线地址, DPL 的低 5 位为 0, 例如: 00<sub>H</sub>, 20<sub>H</sub>, 40<sub>H</sub>, 60<sub>H</sub>, 80<sub>H</sub>, A0<sub>H</sub>, C0<sub>H</sub> 和 E0<sub>H</sub>。

2) 当 PSW.CY 为 0 时, DPTR 仅增加 20<sub>H</sub>。

## 23.7 Flash 擦除子程序

每次调用 Flash 擦除子程序允许擦除所选择 Flash Bank 的扇区。无论何时，可选择擦除多个 Flash 扇区<sup>1)2)</sup>。该子程序还支持擦除用户 BSL Flash 扇区<sup>1)</sup>。用户 BSL Flash 扇区的擦除操作可与 Flash Bank0/1 Flash 扇区的擦除一起完成。

调用该子程序之前，用户必须确保相应的设置输入。为用户提供两类 Flash 擦除子程序。可从 XRAM 或 Flash 中调用擦除和编程用户子程序。表 23-2 描述如何使用和调用这些子程序。

1) 仅适用于 XC83x - 8K 器件。

2) 适用于 **BR\_FLASH\_ERASE** 用户子程序，可从 Flash 或 XRAM 中调用该子程序。同时擦除两个 Flash bank 时，需要选择 **BR\_FLASH\_BACKGROUND\_ERASE** 用户子程序，此时必须从 XRAM 中调用该程序。

Boot ROM 用户子程序

第一类子程序（不支持后台擦除） **BR\_FLASH\_ERASE**，将等待直到擦除操作完成，才能继续执行用户程序。如果查询错误时发现 FNMIVDDP， FNMIVDD 或 FNMIPLL 标志被置位，则该子程序将被中止。该类子程序在下述情况下非常必要：如果用户想要擦除的 Flash Bank 刚好保存着正在执行的用户代码，尤其当器件仅有一个 Flash Bank 时。Flash 不能同时处于擦除模式和读取模式。对于用户想要擦除的 Flash bank 还用于保存中断向量的情况，Flash Bank 处于擦除模式时，中断处理器无法访问该 Flash Bank，因此无法处理中断，此时该类子程序也非常有用。对于第一类的擦除子程序，Boot ROM 将控制代码， FNMIFLASH 标志被自动清除，因此无需要用户处理该标志。

注： 擦除操作完成后，由于程序将返回到用户代码，用户需要注意不要无意的擦除了自己的用户代码。

注： 当为该子程序提供无效的 / 无输入时， C 标志将被置位，将退出子程序。不能完成其他任何操作。

表 23-11 FLASH\_ERASE 子程序说明

子程序	BR_FLASH_ERASE
输入 <sup>1)</sup>	<p>当前寄存器组的 R7: Flash Bank 0 中的、所选择的想要擦除的扇区 LSB 代表扇区 0， MSB 代表扇区 7</p> <p>当前寄存器组的 R6: Flash Bank 0 中的、所选择的想要擦除的扇区 LSB 代表扇区 8， 位 1 代表扇区 9</p> <p>当前寄存器组的 R5<sup>2)</sup>: Flash Bank 1 中的、所选择的想要擦除的扇区 LSB 代表扇区 0， MSB 代表扇区 7</p> <p>当前寄存器组的 R4<sup>2)</sup>: Flash Bank1 中的、所选择的想要擦除的扇区 LSB 代表扇区 8， 位 1 代表扇区 9， 位 2 代表用户 BSL Flash 扇区</p> <p>SFR NMISR = 00<sub>H</sub></p> <p>所有中断，包括 NMI 在内都应当被禁止（IEN0.EA=0， NMICON=00<sub>H</sub>）</p>
输出	<p>PSW.CY:</p> <p>0 = Flash 擦除操作成功</p> <p>1 = Flash 擦除操作不成功</p> <p>1 = 给出的输入无效 / 无输入</p>
所需要的堆栈大小	12
使用 / 破坏的资源	<p>DPTR, A</p> <p>当前寄存器组的 R2 和 R7（2 字节）</p>

## Boot ROM 用户子程序

- 1) 如果不选择擦除 Bank 中的某个 / 某几个扇区，对应输入应当设置为 0。
- 2) 仅适用于 XC83x-8K 器件。对于其它器件配置，该字节被忽略。

第二类子程序（支持后台擦除） **BR\_FLASH\_BACKGROUND\_ERASE**，允许在调用擦除子程序之后，继续执行用户程序代码。用户将一直等待，直到产生 Flash NMI 事件；寄存器 NMISR 中的位 FNMIFLASH 置位，且如果由 NMIFLASH 使能，为了进入 Flash NMI 服务子程序，触发一个发送给 CPU 的 NMI。此时 Flash bank 处于准备好 - 可供读取模式，即擦除操作已经完成。

*注： 当为该子程序提供无效 / 无输入时，C 标志将被置位，将退出子程序。不能完成其他任何操作。*

**表 23-12 FLASH\_BACKGROUND\_ERASE 子程序说明**

子程序	BR_FLASH_BACKGROUND_ERASE
输入 <sup>1)</sup>	当前寄存器组的 R7: Flash Bank 0 中的、想要擦除的扇区 LSB 代表扇区 0， MSB 代表扇区 7
	当前寄存器组的 R6: Flash Bank 0 中的、想要擦除的扇区 LSB 代表扇区 8， 位 1 代表扇区 9
	当前寄存器组的 R5 <sup>2)</sup> : Flash Bank1 中的、想要擦除的扇区 LSB 代表扇区 0， MSB 代表扇区 7
	当前寄存器组的 R4 <sup>2)</sup> : Flash Bank1 中的、想要擦除的扇区 LSB 代表扇区 9， 位 1 代表扇区 9， 位 2 代表用户 BSL Flash 扇区
	SFR NMISR = 00 <sub>H</sub>
	Flash NMI（NMICON.NMIFLASH）被使能（1）或被禁止（0）
输出	PSW.CY: 0 = 正在进行 Flash 擦除操作 1 = Flash 擦除操作不成功 1 = 给出的输入无效 / 无输入
所需堆栈大小	8
使用 / 破坏的资源	DPTR, A
	当前寄存器组的 R2（1 字节）

- 1) 如果不选择擦除 Bank 中的某个 / 某几个扇区，对应输入应当设置为 0。

2) 仅适用于 XC83x-8K 器件。对于其它器件配置，该字节被忽略。

### 23.8 Flash 擦除中止子程序

Flash Bank 上的每个完整的擦除操作大约需要 100 ms，在此期间不能对 Flash 进行读取或编程操作。对于 XC83x 来讲，已经提供了预防措施，允许正在进行的擦除操作（类型 2，[BR\\_FLASH\\_BACKGROUND\\_ERASE](#)）被中断转而执行具有更高优先级的任务，如从 Flash 读取 / 向 Flash 编程关键数据的操作。因此，Flash Bank 中所选择扇区的擦除操作可被中止，以允许读取或编程其它扇区。为了将擦除中止操作对 Flash 数据保持特性 / 编程 / 擦除周期次数的影响减至最低，并且为了保证数据可靠性，对每个 Flash Bank 进行操作时，必须注意以下几点：

- 擦除操作开始 5 ms 之后才能被中止
- 每个扇区最多允许两次连续的（其间没有完整的擦除操作）擦除中止操作。
- 单次或两次连续的擦除中止操作之后，需要一个由用户程序启动的、完整的擦除操作（大约需要 100 ms），由于相关的数据已经被破坏。
- 对于特定的编程 / 擦除次数，每个被中止的擦除操作被当作一次编程 / 擦除操作。
- 整个芯片生命周期期间，每个 D-Flash 扇区允许的最大擦除中止次数为 2500。

Flash 擦除操作启动之后 5 ms 之内，不能调用 Flash 擦除中止子程序。这是一个严格的要求，用户必须遵守。否则，擦除操作不能被中止。一旦退出该子程序，用户可调用 [BR\\_FLASH\\_READ\\_MODE\\_STATUS](#) 用户子程序检查擦除中止操作是否已经完成。当所选择的扇区已经进入读模式，则只是擦除中止操作已经完成。

表 23-13 Flash 擦除中止子程序

子程序	BR_FLASH_BACKGROUND_ERASE_ABORT
输入	Flash Bank 处于擦除模式 <sup>1)</sup>
输出	C = 0: 正在进行擦除中止操作 C = 1: 未启动 Flash 擦除中止操作
所需堆栈大小	3
使用的 / 破坏的资源	A

1) 对于 XC83x-8K 器件来讲，Flash Bank 0 和 Flash Bank 1 可同时处于擦除模式，因而两个 Flash Bank 的擦除操作也可一起被中止。



## 24 ROM Library

除用户程序之外，我们还为用户开发自己的应用提供了一组实用的 ROM library 程序。XC83x 可提供的 ROM library 有：

- **定点 ROM Library**,
- **LED 和触摸感应控制器 ROM Library**,
- **EEPROM 模拟 ROM Library**

用户应用程序可调用表 24-1 中列出的 ROM Library 函数。后面各节将对不同的 ROM library 予以详细说明。

**表 24-1 XC83x ROM Library 函数和相应的地址**

地址	函数	说明
0xD649	_PI_CONTROLLER_G16	定点 ROM Library
0xD64F	_PI_CONTROLLER_G256	定点 ROM Library
0xD655	_P_CONTROLLER_G16	定点 ROM Library
0xD65B	_P_CONTROLLER_G256	定点 ROM Library
0xD75B	_PT1_24	定点 ROM Library
0xD78A	_PT1_32	定点 ROM Library
0xD7CE	_CLARKE	定点 ROM Library
0xD802	_CARTESIAN_POLAR_POLL	定点 ROM Library
0xD805	_CARTESIAN_POLAR_RET	定点 ROM Library
0xD81E	_VECTOR_ROTATION	定点 ROM Library
0xD821	_INV_VECTOR_ROTATION	定点 ROM Library
0xD859	_SINCOSINE_POLL	定点 ROM Library
0xD85C	_SINCOSINE_RET	定点 ROM Library
0xDFCC	FINDTOUCHEDPAD	LEDTS ROM Library
0xDFCF	SET_LDLINE_CMP	LEDTS ROM Library
0xDFD2	INITEEPROM	EEPROM 模拟 ROM Library
0xDFD5	FIXEEPROM	EEPROM 模拟 ROM Library
0xDFD8	READEEPROM	EEPROM 模拟 ROM Library
0xDFDB	WRITEEEPROM	EEPROM 模拟 ROM Library

## 24.1 定点 ROM Library

定点 library 中包含多个存储于 ROM、可由用户访问的程序。以下各节将对这些程序予以详细说明。

表中的“执行周期”未包括调用指令“LCALL”。在 Flash 中执行该指令需要 10 个时钟周期；在 XRAM 中执行该指令需要 8 个时钟周期。

### 24.1.1 P 控制器程序

P 控制器程序用于实现一种比例控制算法，该算法的定义如下：

(24.1)

$$Y(k) = G \times Kp \times X(k)$$

其中：

- Y(k)：P 控制器的输出
- G：增益因子（16 或 256）
- Kp：比例增益
- X：误差 = 参考值 - 实际值
- k：时间或瞬时时间

表 24-2 P 控制器程序说明

子程序	_P_controller_G16 或 _P_controller_G256
地址	0xD655: _P_controller_G16 0xD65B: _P_controller_G256
C 函数原型	int P_controller_G16(int Ref_val, int Actual_val, char idata *Kp) int P_controller_G256(int Ref_val, int Actual_val, char idata *Kp)
输入	R7, R6 (MSB), Ref_val = 16 位参考值
	R5, R4 (MSB), Acutal_val = 16 位实际值
	R3 = 指向 Kp_H 的 idata 指针
输出	R7, R6 (MSB) = 16 位 Y(k) 值
执行周期	_P_controller_G16: 138 cclks _P_controller_G256: 112 cclks
用到的资源	PSW, A, MDU
	当前寄存器 bank 中的 R0, R4, R5

代码示例：

```
// 全局变量
data int Speed;
data int Speed_ref;
```

```
data int Speed_kp;
data int P_Speed;

// 程序
Speed = 16288
Speed_ref = 4000;
Speed_kp = 4096;
P_Speed = P_controller_G16(Speed_ref, Speed, (char idata *)
&Speed_kp);
//P_speed = 0x7F40
```

### 24.1.2 PI 控制器程序

PI 控制器程序用于实现一种比例积分控制算法，该算法的定义如下：

(24.2)

$$Y(k) = G \times Kp \times X(k) + Y(k-1) + Ki \times X(k)$$

其中：

- Y(k)：PI 控制器的输出
- Y(k-1)：PI 控制器前一次的输出
- G：增益因子（16 或 256）
- Kp：比例增益
- Ki：积分增益
- X：误差 = 参考值 - 实际值
- k：时间或瞬时时间

**表 24-3 PI 控制器程序说明**

子程序	_PI_controller_G16 或 _PI_controller_G256
地址	0xD649: _PI_controller_G16 0xD64F: _PI_controller_G256
C 函数原型	int PI_controller_G16(int Ref_val, int Actual_val, PI_param *pPI) int PI_controller_G256(int Ref_val, int Actual_val, PI_param *pPI)
输入	R7, R6 (MSB), Ref_val = 16 位参考值
	R5, R4 (MSB), Acutal_val = 16 位实际值
	R3 = 指向 Y(k-1)_H 的 idata 指针

表 24-3 PI 控制器程序说明

备注	要求如下的结构体： Struct{ Y(k-1)[HLh] Kp[HL] Ki[HL] PI_SAT_HH //24 位饱和值 0xHHFFFF 的高字节 }
输出	R7, R6 (MSB) = 16 位 Y(k) 值
执行周期	_PI_controller_G16: 248 cclks _PI_controller_G256: 226 cclks
用到的资源	PSW, A, MDU
	当前寄存器 bank 中的 R0, R2, R4, R5

代码示例：

```
// 定义结构体
typedef struct pi_param{
    char        yn[3];
    int          kp;
    int          ki;
    unsigned char  PI_SAT_HH;
}PI_param;

// 全局变量
data int Speed;
data int Speed_ref;
data int PI_Speed;
idata PI_param Speed_control;

// 程序
Speed = 2048;
Speed_ref = 4000;
Speed_control.yn[0] = 0x01;
Speed_control.yn[1] = 0x4C;
Speed_control.yn[2] = 0x97;
Speed_control.kp = 4096;
Speed_control.ki = 32;
Speed_control.PI_SAT_HH = 0x1F;
PI_Speed = PI_controller_G16(Speed_ref, Speed, &Speed_control);
//PI_Speed = 0x108E
```

### 24.1.3 PT1\_24 控制器程序

PT1\_24 控制器程序是一个数字低通滤波器，其实现是通过以下方式定义的：

(24.3)

$$Y(k) = Y(k-1) + 2^{-Z} \times [X(k) - Y(k-1)]$$

其中：

- Y(k): PT1\_24 控制器的输出
- Y(k-1): PT1\_24 控制器前一次的输出
- Z: 分频因子 ( $1/2^Z$ )
- X(k): 16 位输入
- k: 时间或瞬时时间

表 24-4 PT1\_24 控制器程序说明

子程序	_PT1_24
地址	0xD75B: _PT1_24
C 函数原型	int PT1_24(int X, char Z, char idata *pY)
输入	R7, R6 (MSB) = 16 位 X 值
	R5 = 8 位 Z 值，右移位次数 ( $1/2^Z$ )
	R3 = 指向 Y(k-1)_H 的 idata 指针
备注	Y(k-1) = 24 位 [HLh]
	X 必须介于 0xC000 - 0x3FFF 之间
输出	R7, R6 (MSB) = 16 位 Y(k) 值
执行周期	76 cclk
用到的资源	PSW, A, MDU
	当前寄存器 bank 中的 R0

代码示例：

```
// 声明变量
data char Speed[3];
data int result;
data int angle_new;
data int angle_old;

// 程序
Speed[0] = 0xF8;
Speed[1] = 0xF3;
```

```
Speed[2] = 0xCB;
angle_new = 0x1800;
angle_old = 0x0800;
result = PT1_24(angle_new - angle_old, 5, &speed);
//result = 0xF9AC;
```

### 24.1.4 PT1\_32 控制器程序

PT1\_32 控制器程序是一个积分器，其实现是通过以下方式定义的：

(24.4)

$$Y(k) = Y(k-1) + Z1 \times X(k) - Z2 \times Y(k-1)$$

其中：

- Y(k): PT1\_32 控制器的输出
- Y(k-1): PT1\_32 控制器前一次的输出
- Z: 分频因子 ( $1/2^Z$ )
- X(k): 16 位输入
- k: 时间或瞬时时间

表 24-5 PT1\_32 控制器程序说明

子程序	_PT1_32
地址	0xD78A: _PT1_32
C 函数原型	int PT1_32(int X, int Z2, int idata *pY)
输入	R7, R6 (MSB) = 16 位 X 值
	R5, R4 (MSB) = 16 位 Z2 值
	R3 = 指向 Y(k-1)_H 的 idata 指针
	MDU_MD4 = Z1_L
	MDU_MD5 = Z1_H
备注	Y(k-1) = 32 位 [HLh]
	Z1 必须介于 0x0000 - 0x3FFF 之间，或 X 必须介于 0xC000 - 0x3FFF 之间
输出	R7, R6 (MSB) = 16 位 Y(k) 值
执行周期	118 cclk
用到的资源	PSW, A, MDU
	当前寄存器 bank 中的 R0

**代码示例:**

```
// 全局变量
data int    result;
data int    Integrator[2];

// 程序
Integrator[0] = 0xFF12;
Integrator[1] = 0xE828;
MDU_MD4 = 0x1F; //Low byte Z1
MDU_MD5 = 0x00; //High byte Z1
result = PT1_32(X, 0x109, &Integrator);
//result = 0xFF15
```

### 24.1.5 Cartesian-Polar 坐标转换程序

该程序用于将笛卡尔坐标转换为极坐标，定义如下：

(24.5)

$$R = K \times \left( \frac{\sqrt{X^2 + Y^2}}{4} \right)$$

(24.6)

$$\theta = \text{atan}\left(\frac{Y}{X}\right)$$

其中：

- R: 距离原点的径向距离
- $\theta$ : 相对 x 轴的反时针夹角
- K: ~ 1.64676 倍的比例因子
- X: x 坐标
- Y: y 坐标

**表 24-6 Cartesian-Polar 坐标转换程序说明**

子程序	_Cartesian_Polar_poll 或 _Cartesian_Polar_ret
地址	0xD802: _Cartesian_Polar_poll 0xD805: _Cartesian_Polar_ret

表 24-6 Cartesian-Polar 坐标转换程序说明

<b>C 函数原型</b>	void Cartesian_Polar_poll(int X, int Y) - 程序在返回之前等待 CD_BSY 标志。一旦程序返回，可从 CORDX 和 CORDZ 中读取结果。  void Cartesian_Polar_ret(int X, int Y) - 一旦 CORDIC 开始工作，程序返回，不等待 CD_BSY 标志。用户将检查 CD_BSY 标志或在读取 CORDX 和 CORDY 中的结果之前等待足够长的时间。
<b>输入</b>	R7, R6 (MSB) = 16 位 X 值 R5, R4 (MSB) = 16 位 Y 值
<b>输出</b>	CORDX = R 和 CORDZ = 0
<b>执行周期</b>	_Cartesian_Polar_poll: 58 cclk _Cartesian_Polar_ret: 38 cclk
<b>用到的资源</b>	PSW, A, CORDIC

代码示例：

```
// 要得到 R 值，设置 X = x/1.64676, Y = y/1.6476
// 即：R = Sqrt[sq(X) + sq(Y)]/4
// 若 X = 300, Y = 400,
//X = 300/1.64676 = 182.18 = 0xB7
//Y = 400/1.64676 = 242.90 = 0xF3
```

```
// 程序
int R_result, angle_result;

Cartesian_Polar_ret(0xB7, 0xF3);
while(CD_BSY); // 等待至转换结束
R_result = CD_CORDXLH;
angle_result = CD_CORDZLH;
//R_result = 0x007D, angle_result = 0x25B3
```

### 24.1.6 Clarke 变换程序

该程序用于将三相系统变换为两相正交系统，其实现是通过以下方式定义的：



$$(24.7)$$

$$I\_alpha = \frac{I\_phaseA}{2}$$

$$(24.8)$$

$$I\_beta = \left( \frac{PhaseA + 2 \times PhaseB}{\sqrt{3} \times 2} \right)$$

其中：

- Y(k): PT1\_24 控制器的输出
- Y(k-1): PT1\_24 控制器前一次的输出
- Z: 分频因子 ( $1/2^Z$ )
- X(k): 16 位输入
- k: 时间或瞬时时间

**表 24-7      Clarke 变换程序说明**

子程序	_Clarke
地址	0xD7CE: _Clarke
C 函数原型	void Clarke(int idata *_I_phaseAB, int idata *_I_alphabeta)
输入	R7 = 指向 I_phaseA_H 的 idata 指针 R5 = 指向 I_alpha_H 的 idata 指针
备注	char idata *_I_phaseAB IRAM 中的地址分配: I_phaseA_H (MSB), I_phaseA_L, I_phaseB_H, I_phaseB_L  char idata *_I_alphabeta IRAM 中的地址分配: I_alpha_H (MSB), I_alpha_L, I_beta_H, I_beta_L
输出	I_alpha 和 I_beta 的结果保存在第 2 个输入参数的地址单元中
执行周期	84 cclk
用到的资源	PSW, A, MDU 当前寄存器 bank 中的 R0, R1

代码示例：

```
// 声明变量
data int I_phaseA _at_ (0x36);
data int I_phaseB _at_ (0x38);
data int I_alpha _at_ (0x4A);
```

```
data int I_beta      _at_ (0x4C);

// 程序
I_phaseA = -2048;
I_phaseB = 8192;
Clarke(&I_phaseA, &I_alpha);
//I_alpha = 0xFC00, I_beta = 0x102A
```

### 24.1.7 向量旋转（Park 变换）程序

该程序用于将 Clarke 变换产生的两相（参见[章节 24.1.6](#)）旋转一个角度  $\theta$ ：

(24.9)

$$I_d = \frac{(I_{\alpha} \times \cos(\theta) + I_{\beta} \times \sin(\theta))}{2}$$

(24.10)

$$I_q = \frac{(-I_{\alpha} \times \sin(\theta) + I_{\beta} \times \cos(\theta))}{2}$$

其中：

- Y(k): PT1\_24 控制器的输出
- Y(k-1): PT1\_24 控制器前一次的输出
- Z: 分频因子 ( $1/2^2$ )
- X(k): 16 位输入
- k: 时间或瞬时时间

**表 24-8 向量旋转程序说明**

子程序	_Vector_Rotation
地址	0xD81E: _Vector_Rotation
C 函数原型	void Vector_Rotation(int Angle, int idata *pInput, int idata *pOutput)
输入	R7, R6 (MSB) = 16 位角度值 R5 = 指向 I_alpha_H 的 idata 指针 R3 = 指向 Iq_H 的 idata 指针
备注	char idata *pInput IRAM 中的地址分配: I_alpha_H (MSB), I_alpha_L, I_beta_H, I_beta_L

表 24-8 向量旋转程序说明

	char idata *pOutput IRAM 中的地址分配: Iq_H (MSB), Iq_L, Id_H, Id_L
输出	Iq 和 Id 的结果保存在第 3 个输入参数的地址单元中
执行周期	92 cclk
用到的资源	PSW, A, B, CORDIC
	当前寄存器 bank 中的 R0

代码示例:

```
// 声明变量
data int I_alpha _at_ (0x4A);
data int I_beta _at_ (0x4C);
data int I_q _at_ (0x4E);
data int I_d _at_ (0x50);
data int angle;

// 程序
I_alpha = -4096;
I_beta = 8192;
// 若角度为 126°, 将其转换为弧度单位
angle = 126*32768/180;
Vector_Rotation(angle, &I_alpha, &I_q);
//I_q = 0xFB2C, I_d = 0x1D0F
```

### 24.1.8 向量逆旋转 (Park 逆变换) 程序

该程序用于实现向量逆旋转 (向量旋转参见 [章节 24.1.7](#)), 将 Id 和 Iq 变换回 I\_alpha 和 I\_beta, 旋转角度为  $\theta$ :

$$I_{\alpha} = \frac{(I_d \times \cos(\theta) - I_q \times \sin(\theta))}{4} \quad (24.11)$$

$$I_{\beta} = \frac{(I_d \times \sin(\theta) + I_q \times \cos(\theta))}{4} \quad (24.12)$$

其中:

- Y(k): PT1\_24 控制器的输出

- Y(k-1): PT1\_24 控制器前一次的输出
- Z: 分频因子 ( $1/2^Z$ )
- X(k): 16 位输入
- k: 时间或瞬时时间

表 24-9 向量逆旋转程序说明

子程序	_Inv_Vector_Rotation
地址	0xD821: _Inv_Vector_Rotation
C 函数原型	void Inv_Vector_Rotation(int Angle, int idata *pInput, int idata *pOutput)
输入	R7, R6 (MSB) = 16 位角度值 R5 = 指向 Iq_H 的 idata 指针 R3 = 指向 I_alpha_H 的 idata 指针
备注	char idata *pInput IRAM 中的地址分配: Iq_H (MSB), Iq_L, Id_H, Id_L  char idata *pOutput IRAM 中的地址分配: I_alpha_H (MSB), I_alpha_L, I_beta_H, I_beta_L
输出	Iq 和 Id 的结果保存在第 3 个输入参数的地址单元中
执行周期	88 cclk
用到的资源	PSW, A, B, CORDIC 当前寄存器 bank 中的 R0

### 24.1.9 Sine/Cosine 程序

该程序用于实现 Sine/Cosine 函数:

(24.13)

$$Y_o = \frac{K \times [Y \times \cos(z) + X \times \sin(z)]}{2}$$

其中:

- Yo: 输出值
- K: ~ 1.64676 倍的比例因子
- Y: cos(z) 的增益因子
- X: sin(z) 的增益因子
- z: 角度 (以弧度为单位时:  $z * 32768/180^\circ$ )

表 24-10 Sine/Cosine 程序说明

子程序	<code>_SinCos_poll</code> 或 <code>_SinCos_ret</code>
地址	<code>0xD859: _SinCos_poll</code> <code>0xD85C: _SinCos_ret</code>
C 函数原型	<code>int SinCos_poll(int X, int Y, int Z)</code> - 程序将结果返回 R7 和 R6 之前等待 CD_BSY 标志。  <code>void SinCos_retI(int X, int Y, int Z)</code> - 一旦 CORDIC 开始工作，程序返回，不等待 CD_BSY 标志。用户将检查 CD_BSY 标志或在读取 CORDY 中的结果之前等待足够长的时间。
输入	<code>R7, R6 (MSB) = 计算 Sine 的 16 位 X 值。计算 Cosine 时输入为 0。</code> <code>R5, R4 (MSB) = 计算 Cosine 的 16 位 Y 值。计算 Sine 时输入为 0。</code> <code>R3, R2 (MSB) = 16 位 Z 值</code>
输出	<code>R7, R6 (MSB) = Yo, 适用于程序 _SinCos_poll</code> <code>CORDY = Yo, CD_BSY 标志置位之后适用于程序 _SinCos_ret</code>
执行周期	<code>_SinCos_poll: 70 cclk</code> <code>_SinCos_ret: 42 cclk</code>
用到的资源	PSW, A, CORDIC

代码示例:

```
// 计算 Sine 值示例
// 第 1 步：设置 Y = 0。公式变为：
//Yo = K * [X * sin(z)]/2 = 1.64676 * [X * sin(z)]/2
// 第 2 步：使增益因子为 1 (Yo = sin(z))，
// 设置 X = 2/1.64676 = 1.214506
// 将 x 扩大 10000 倍以获得更好的精度，
//X = 12145.06 = 0x2F72
// 第 3 步：将 z 转换为弧度单位。若角度为 45°，
//z = 45° * 32768/180° = 8192 = 0x2000

// 程序
int sin_out;

sin_out = SinCos_poll(0x2F72, 0, 0x2000);
//sin_out = 0x1B9F, 缩小 10000 倍即得到结果 0.7071
```

## 24.2 LED 和触摸感应控制器 ROM Library

LEDTS ROM Library 中包含两个存储于 ROM、可由用户访问的程序。以下各节将对这些程序予以详细说明。

- [SET\\_LDLINE\\_CMP 函数（LED 和触摸感应）](#)
- [FINDTOUCHEDPAD 函数（触摸感应）](#)
- [中断时函数的使用](#)

LEDTS 的调用函数归纳见表 24-11。

表 24-11 LEDTS ROM Library 程序列表

地址	函数	说明
0xD0CF	<a href="#">SET_LDLINE_CMP</a>	设置 SFR LTS_LDLINE 和 LTS_COMPARE（仅使能 LED 功能）
0xD0CF	<a href="#">SET_LDLINE_CMP</a>	设置 SFR LTS_LDLINE 和 LTS_COMPARE（仅使能触摸感应功能）
0xD0CF	<a href="#">SET_LDLINE_CMP</a>	设置 SFR LTS_LDLINE 和 LTS_COMPARE（使能 LED 和触摸感应功能）
0xD0CC	<a href="#">FINDTOUCHEDPAD</a>	取平均，计算 LowTrip 并评估按盘是否被触摸

发生时间片或时帧中断时，调用这些函数，详见[章节 24.2.3](#)的描述。

### 24.2.1 SET\_LDLINE\_CMP 函数（LED 和触摸感应）

该函数根据用户的输入参数<sup>1)</sup>为 LED 和 / 或触摸感应功能设置 SFR LTS\_LDLINE 和 LTS\_COMPARE（亮度 / 振荡窗）。该函数读取 SFR LTS\_GLOBCTL1.FNCOL 的值，从而选择将哪些数据同步写入 SFR LTS\_LDLINE 以实现 LED 显示<sup>2)</sup>。同样的，该函数还为各列 LED 或触摸感应功能的各个 pad turn 更新 SFR LTS\_COMPARE 的值。图 24-1 示出该函数的使用概况；图 24-2 给出该函数中 SFR 的设置。

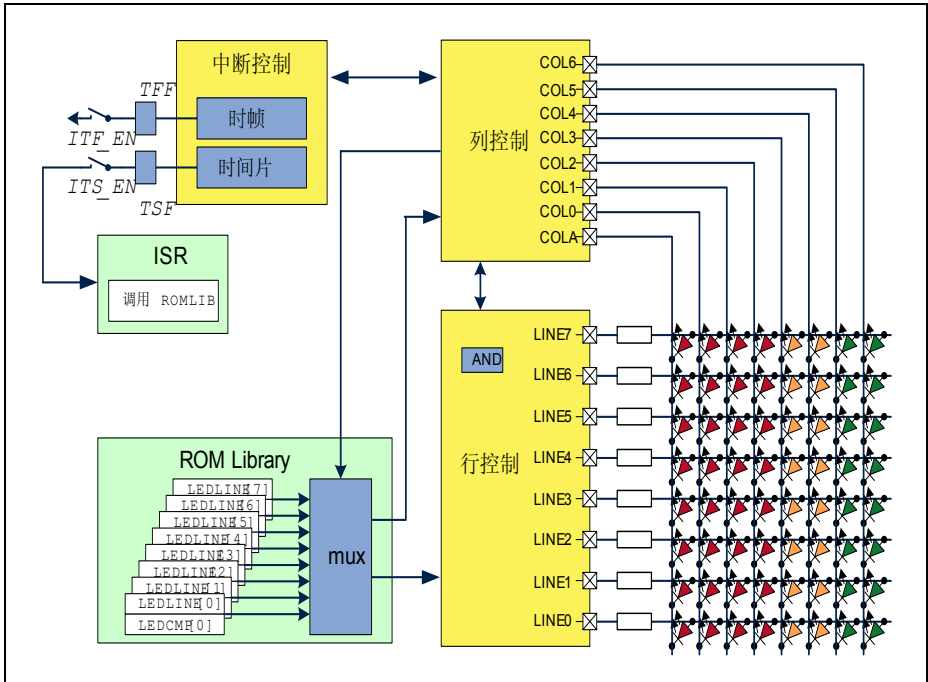


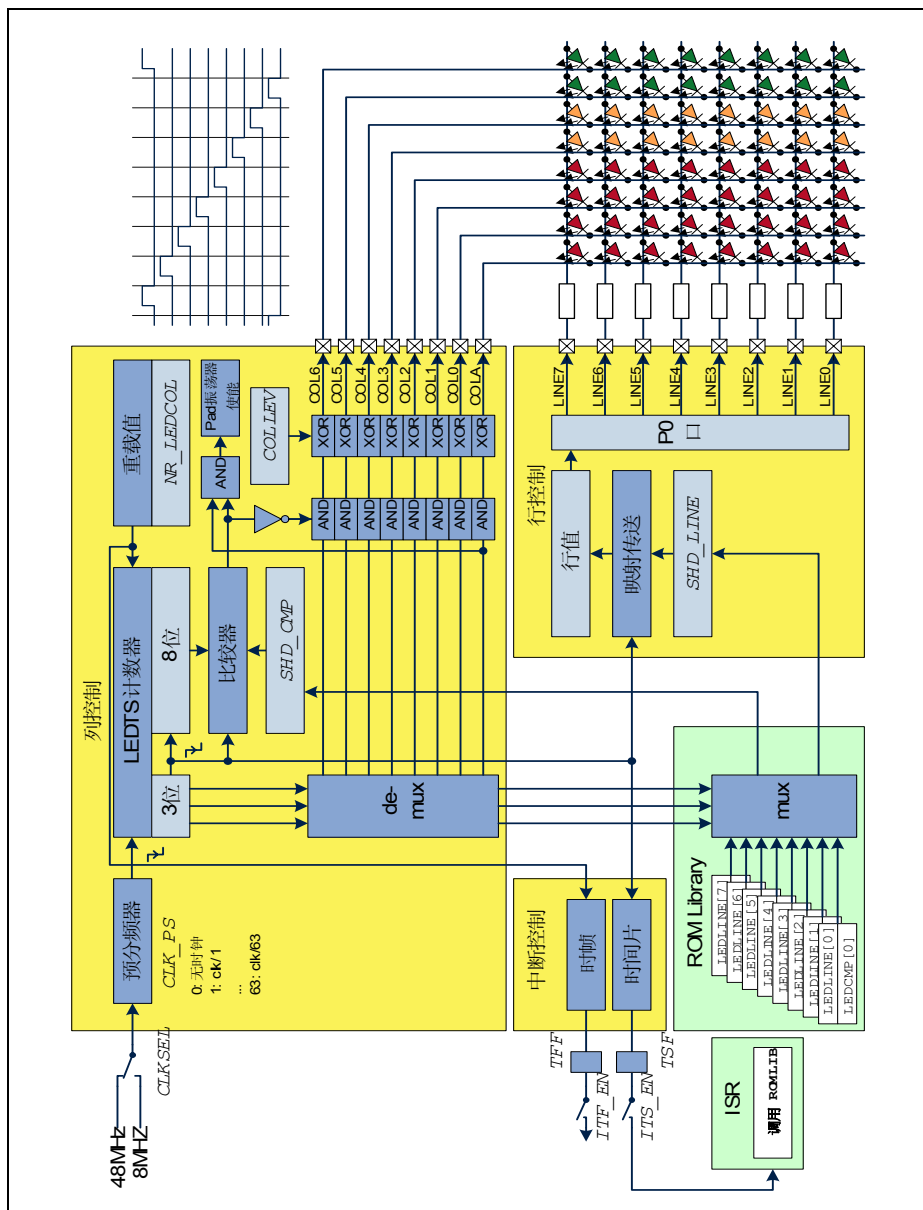
图 24-1 SET\_LDLINE\_CMP 函数概览

每次发生时间片或时帧中断时可调用该函数。有关如何调用该函数的详细描述，可参见章节 24.2.3。

章节 24.2.1.1，章节 24.2.1.2 和章节 24.2.1.3 分别给出仅 LED 功能使能、仅触摸感应功能使能以及 LED 和触摸感应功能均使能情况下的输入。

1) 该函数决定当前时间片内的有效功能 / 列并为下一个时间片更新行值和（映射传送的）比较值，这些操作用户不可见。

2) 用作触摸感应功能时，可将 LTS\_LDLINE 设置为 0xFF。



**图 24-2 SET\_LDLINE\_CMP 函数 - SFR 设置**



### 24.2.1.1 SET\_LDLINE\_CMP 函数的输入（仅使能 LED 功能）

仅使能 LED 功能时，相应的输入参数归纳见表 24-12。使能 2 列、4 列、6 列或 8 列 LED 时的输入参数示例见图 24-3。

**表 24-12 LDLINE & COMPARE 设置说明（仅使能 LED 功能）**

子程序	SET_LDLINE_CMP
地址	DFCF <sub>H</sub>
输入	当前寄存器 bank 中的 R7: LDLINE 参数的起始 IRAM 地址
	当前寄存器 bank 中的 R5: COMPARE 参数的起始 IRAM 地址
	写入 IRAM R7... R7+(y-1) 的 LDLINE 参数 <sup>1)</sup> : @R7 = COLA 的 LDLINE 参数 @R7+1 = COL0 的 LDLINE 参数 ... @R7+(y-1) = COL(y-2) 的 LDLINE 参数
	写入 IRAM R5...R5+(y-1) 的 COMPARE 参数 <sup>1)</sup> @R5 = COLA 的 COMPARE 参数 @R5+1 = COL0 的 COMPARE 参数 ... @R5+(y-1) = COL(y-2) 的 COMPARE 参数
输出	SFR LTS_LDLINE 设置完成 SFR LTS_COMPARE 设置完成
所需堆栈大小	2
用到的资源	A, R0, R1, R2, R3, R4

1) 和使能 LED 的列数有关；y = 使能 LED 的列数。

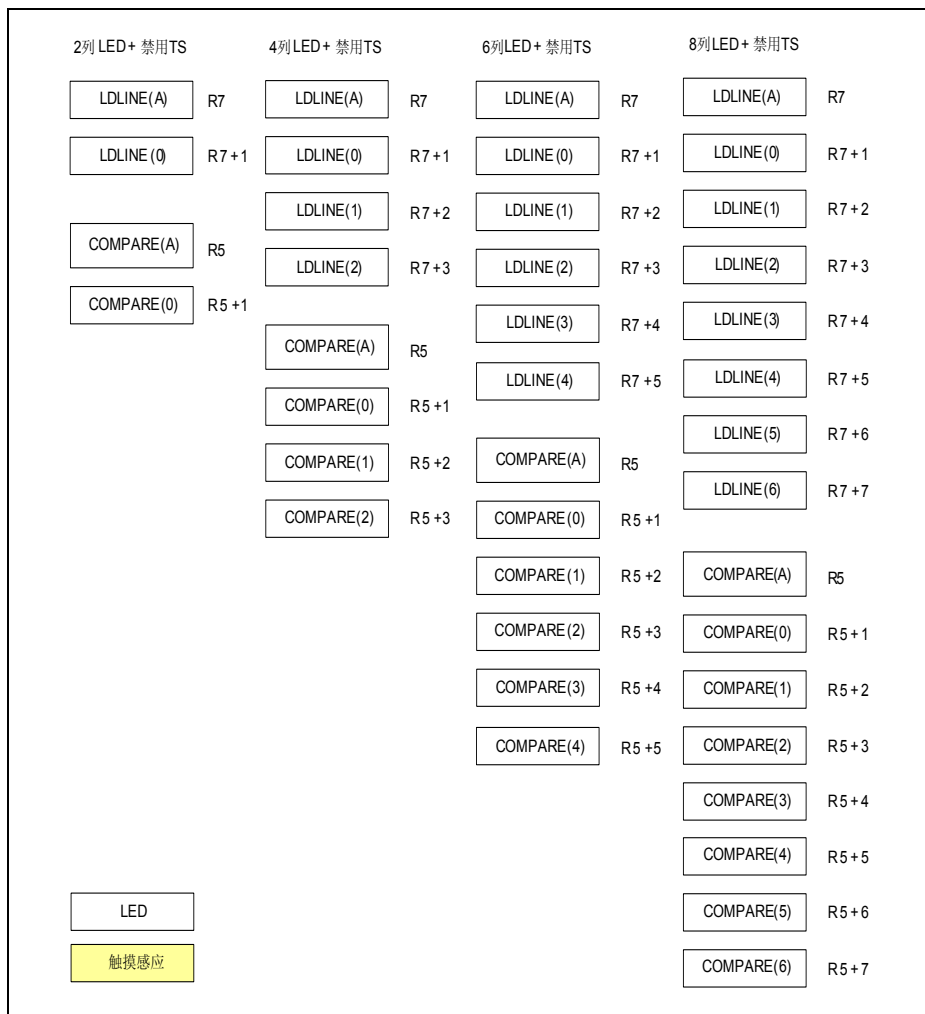


图 24-3 输入参数示例（仅使能 LED 功能）

### 24.2.1.2 SET\_LDLINE\_CMP 函数的输入（仅使能触摸感应功能）

仅使能触摸感应功能时，相应的输入参数归纳见表 24-13（采用公共比较参数）和表 24-14（采用不同的比较参数）。输入参数示例见图 24-4。

用户可以灵活的选择比较参数，也就是说，可为所有 pad turn 选择一个公共比较参数，或为每个 pad turn 分别选择不同的比较参数。这也为不同的按盘提供了更好的灵敏度。若 CMP\_OPTION（位于地址 R5）设置为 0xFF，表示选择不同的比较参数，则可从 IRAM 地址 R5+1 开始递增读取每个 pad turn 的不同比较参数<sup>1)</sup>。若 CMP\_OPTION 不设置为 0xFF，表示所有 pad turn 共用一个比较参数，该值写入地址 R5 中<sup>2)</sup>。

在准备触摸感应功能的比较值时，函数中已考虑到量化效应。各比较值将和 2, 4, 8, 16 ... 进行 XRL 操作，将最终的结果更新到 SFR LTS\_COMPARE 中。量化效应提供了更好的灵敏度，由于所有操作均由函数完成，因此它对用户而言完全不可见。

**表 24-13 LDLINE & 公共 COMPARE 设置说明（仅使能触摸感应功能）**

子程序	SET_LDLINE_CMP
地址	DFCF <sub>H</sub>
输入	<p>当前寄存器 bank 中的 R7: LDLINE 参数的起始 IRAM 地址</p> <p>当前寄存器 bank 中的 R5: CMP_OPTION &amp; COMPARE 参数的起始 IRAM 地址</p> <p>写入 IRAM 地址 R7 的 LDLINE 参数: @R7 = COLA (用作触摸感应) 的 LDLINE 参数</p> <p>写入 IRAM 地址 R5 的 COMPARE 参数: @R5 = 所有 pad turn 的公共 COMPARE 参数<sup>1)</sup></p>
输出	<p>SFR LTS_LDLINE 设置完成</p> <p>SFR LTS_COMPARE 设置完成</p>
所需堆栈大小	2
用到的资源	A, R0, R1, R2, R3, R4

1) 该公共比较参数不能设置为 0xFF。

**表 24-14 LDLINE & 不同 COMPARE 设置说明（仅使能触摸感应功能）**

子程序	SET_LDLINE_CMP
地址	DFCF <sub>H</sub>

1) 见表 24-14。

2) 见表 24-13。

**表 24-14      LDLINE & 不同 COMPARE 设置说明（仅使能触摸感应功能）**

输入	当前寄存器 bank 中的 R7: LDLINE 参数的起始 IRAM 地址
	当前寄存器 bank 中的 R5: CMP_OPTION & COMPARE 参数的起始 IRAM 地址
	写入 IRAM 地址 R7 的 LDLINE 参数: @R7 = COLA ( 用作触摸感应 ) 的 LDLINE 参数
	写入 IRAM 地址 R5...R5+z 的 COMPARE 参数 <sup>1)</sup> @R5 = CMP_OPTION (0xFF value) @R5+1 = PADT0 的 COMPARE 参数 @R5+2 = PADT1 的 COMPARE 参数 ...
	@R5+z = PADT(z-1) 的 COMPARE 参数
输出	SFR LTS_LDLINE 设置完成 SFR LTS_COMPARE 设置完成
所需堆栈大小	2
用到的资源	A, R0, R1, R2, R3, R4

1) 和使能 pad turn 的个数有关； z = 使能 pad turn 的个数。

### 24.2.1.3 SET\_LDLINE\_CMP 函数的输入（使能 LED 和触摸感应功能）

LED 和触摸感应功能均使能时，相应的输入参数归纳见表 24-15（采用公共比较参数）和（表 24-16 采用不同的比较参数）。输入参数示例见图 24-4 和图 24-5。

用户可以灵活的选择比较参数，也就是说，可为所有 pad turn 选择一个公共比较参数，或为每个 pad turn 分别选择不同的比较参数。这也为不同的按盘提供了更好的灵敏度。若 CMP\_OPTION（位于地址 R5）设置为 0xFF，表示选择不同的比较参数，则可从 IRAM 地址 R5+1 开始递增读取每个 pad turn 的不同比较参数<sup>1)</sup>。若 CMP\_OPTION 不设置为 0xFF，表示所有 pad turn 共用一个比较参数，该值写入地址 R5 中<sup>2)</sup>。

在准备触摸感应功能的比较值时，函数中已考虑到量化效应。各比较值将和 2, 4, 8, 16 ... 进行 XRL 操作，将最终的结果更新到 SFR LTS\_COMPARE 中。量化效应提供了更好的灵敏度，由于所有操作均由函数完成，因此它对用户而言完全不可见。

**表 24-15 LDLINE & 公共 COMPARE 设置说明（LEDTS）**

子程序	SET_LDLINE_CMP
地址	DFCF <sub>H</sub>
输入	<p>当前寄存器 bank 中的 R7: LDLINE 参数的起始 IRAM 地址</p> <p>当前寄存器 bank 中的 R5: COMPARE 参数的起始 IRAM 地址</p> <p>写入 IRAM R7... R7+y 的 LDLINE 参数<sup>1)</sup>. @R7 = COLA（用作触摸感应功能）的 LDLINE 参数 @R7+1 = COL0 的 LDLINE 参数 ... @R7+y = COL(y-1) 的 LDLINE 参数</p> <p>写入 IRAM R5... R5+y 的 COMPARE 参数<sup>1)</sup> @R5 = pad turn (COLA) 的公共 COMPARE 参数<sup>2)</sup> @R5+1 = COL0 的 COMPARE 参数 ... @R5+y = COL(y-1) 的 COMPARE 参数</p>
输出	<p>SFR LTS_LDLINE 设置完成</p> <p>SFR LTS_COMPARE 设置完成</p>
所需堆栈大小	2
用到的资源	A, R0, R1, R2, R3, R4

1) 和使能 LED 的列数有关；y = 使能 LED 的列数。触摸感应功能使能时，最多可使用 7 列 LED。.

2) 该公共比较参数不能设置为 0xFF。

1) 见表 24-16。

2) 见表 24-15。

**表 24-16 LDLINE & 不同 COMPARE 设置说明 (LEDTS)**

子程序	SET_LDLINE_CMP
地址	DFCF <sub>H</sub>
输入	<p>当前寄存器 bank 中的 R7: LDLINE 参数的起始 IRAM 地址</p> <p>当前寄存器 bank 中的 R5: CMP_OPTION &amp; COMPARE 参数的起始 IRAM 地址</p> <p>写入 IRAM 地址 R7...R7+y 的 LDLINE 参数: @R7 = COLA ( 用作触摸感应功能 ) 的 LDLINE 参数 @R7+1 = COL0 的 LDLINE 参数 ... @R7+y = COL(y-1) 的 LDLINE 参数</p> <p>写入 IRAM R5... R5+(y+z) 的 COMPARE 参数<sup>1)</sup> @R5 =CMP_OPTION (0xFF) @R5+1 = COL0 的 COMPARE 参数 @R5+2 = COL1 的 COMPARE 参数 ... @R5+y = COL(y-1) 的 COMPARE 参数 @R5+(y+1) = PADT0 的 COMPARE 参数 @R5+(y+2) = PADT1 的 COMPARE 参数 ... @R5+(y+z) = PADT(z-1) 的 COMPARE 参数</p>
输出	<p>SFR LTS_LDLINE 设置完成</p> <p>SFR LTS_COMPARE 设置完成</p>
所需堆栈大小	2
用到的资源	A, R0, R1, R2, R3, R4

1) 和使能 LED 的列数和使能 pad turn 的个数有关； y = 使能 LED 的列数， z = 使能 pad turn 的个数。

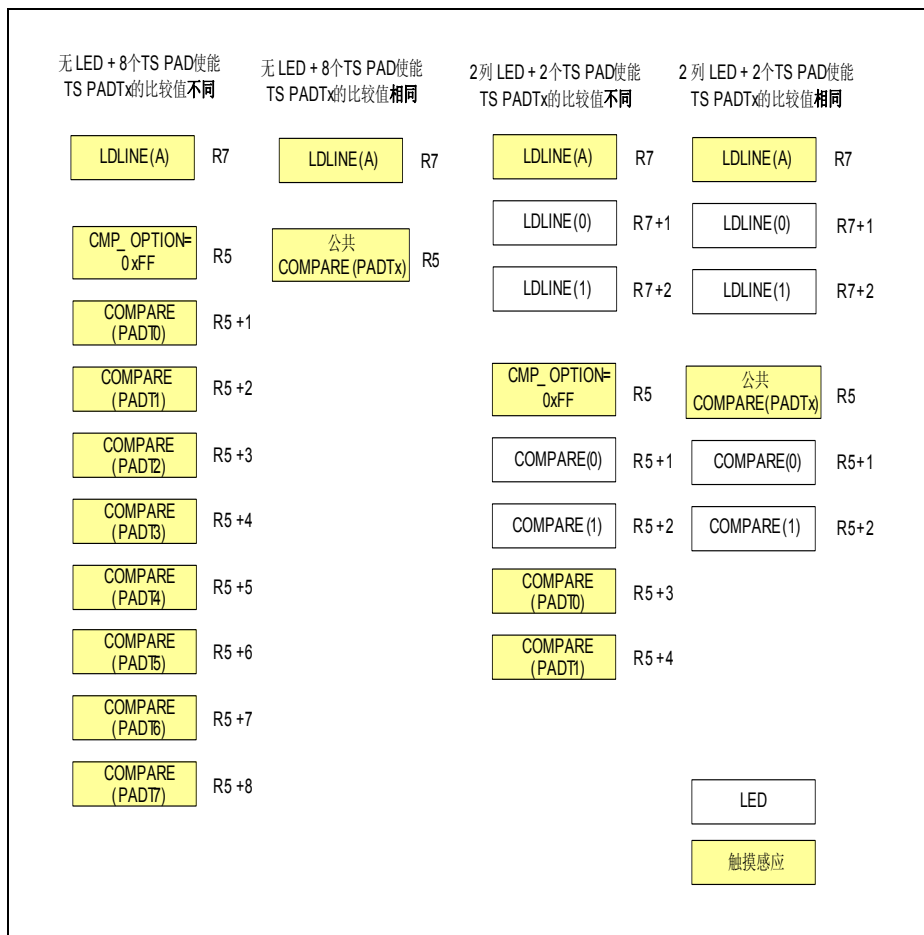


图 24-4 输入参数示例 1 (LED 和触摸感应功能均使能)

ROM Library

4列 LED + 5个TS PAD使能 TS PADTx的比较值不同		4列 LED + 5个TS PAD使能 TS PADTx的比较值相同		7列 LED + 8个TS PAD使能 TS PADTx的比较值不同		7列LED + 8个TS PAD使能 TS PADTx的比较值相同	
LDLINE (A)	R7	LDLINE (A)	R7	LDLINE (A)	R7	LDLINE (A)	R7
LDLINE (0)	R7 + 1	LDLINE (0)	R7 + 1	LDLINE (0)	R7 + 1	LDLINE (0)	R7 + 1
LDLINE (1)	R7 + 2	LDLINE (1)	R7 + 2	LDLINE (1)	R7 + 2	LDLINE (1)	R7 + 2
LDLINE (2)	R7 + 3	LDLINE (2)	R7 + 3	LDLINE (2)	R7 + 3	LDLINE (2)	R7 + 3
LDLINE (3)	R7 + 4	LDLINE (3)	R7 + 4	LDLINE (3)	R7 + 4	LDLINE (3)	R7 + 4
CMP_OPTION=0xFF	R5	公共 COMPARE (PADTx)	R5	LDLINE (4)	R7 + 5	LDLINE (4)	R7 + 5
COMPARE (0)	R5 + 1	COMPARE (0)	R5 + 1	LDLINE (5)	R7 + 6	LDLINE (5)	R7 + 6
COMPARE (1)	R5 + 2	COMPARE (1)	R5 + 2	LDLINE (6)	R7 + 7	LDLINE (6)	R7 + 7
COMPARE (2)	R5 + 3	COMPARE (2)	R5 + 3	CMP_OPTION=0xFF	R5	公共 COMPARE (PADTx)	R5
COMPARE (3)	R5 + 4	COMPARE (3)	R5 + 4	COMPARE (0)	R5 + 1	COMPARE (0)	R5 + 1
COMPARE (PADT0)	R5 + 5			COMPARE (1)	R5 + 2	COMPARE (1)	R5 + 2
COMPARE (PADT1)	R5 + 6			COMPARE (2)	R5 + 3	COMPARE (2)	R5 + 3
COMPARE (PADT2)	R5 + 7			COMPARE (3)	R5 + 4	COMPARE (3)	R5 + 4
COMPARE (PADT3)	R5 + 8			COMPARE (4)	R5 + 5	COMPARE (4)	R5 + 5
COMPARE (PADT4)	R5 + 9			COMPARE (5)	R5 + 6	COMPARE (5)	R5 + 6
				COMPARE (6)	R5 + 7	COMPARE (6)	R5 + 7
				COMPARE (PADT0)	R5 + 8		
				COMPARE (PADT1)	R5 + 9		
				COMPARE (PADT2)	R5 + 10		
				COMPARE (PADT3)	R5 + 11		
				COMPARE (PADT4)	R5 + 12		
				COMPARE (PADT5)	R5 + 13		
				COMPARE (PADT6)	R5 + 14	LED	
				COMPARE (PADT7)	R5 + 15	触摸感应	

图 24-5 输入参数示例 2（LED 和触摸感应功能均使能）



### 24.2.2 FINDTOUCHEDPAD 函数（触摸感应）

触摸感应的概念是基于将按键看作是 pad 和地之间的一个电容。手指触摸按键时，电容将改变。可通过按键、电阻和 I/O pad 组成的 RC 振荡回路来测量电容变化。由一个 8 位计数器对预设时间段内的振荡操作进行计数（即构成一个频率计数器）。每个 pad turn 均使用该频率计数器（LTS\_TSVAL）。用户可决定采样次数以累加构成一个总频率计数器（TOTAL\_TSCTRL/H）。

可由函数 FINDTOUCHEDPAD 成功检测按键是否被触摸，该函数包括以下特性：

- 找到平均值
- 找到 LowTrip（trip point）
- 找到哪个按键被触摸并产生状态标志

图 24-6 示出该函数的使用概况；图 24-7 给出该函数中 SFR 的设置。

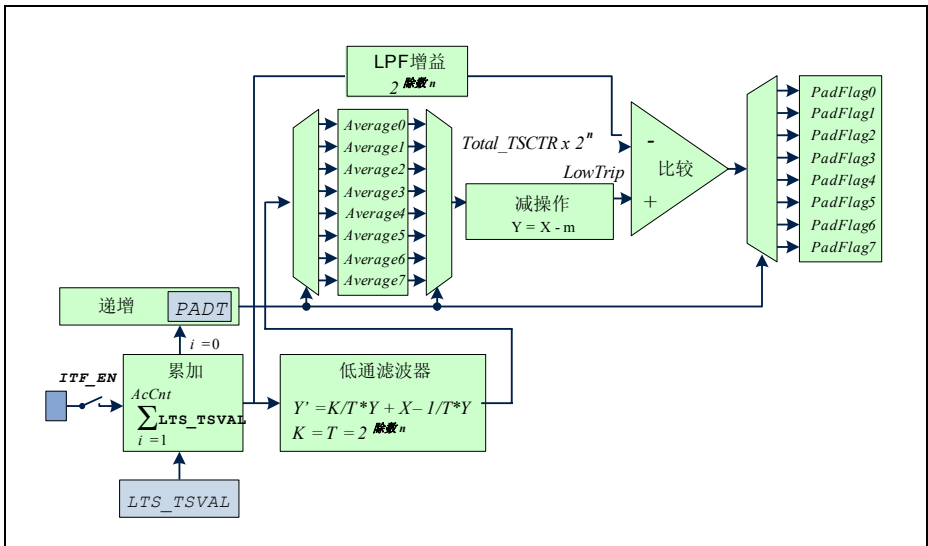


图 24-6 FINDTOUCHEDPAD 函数概览

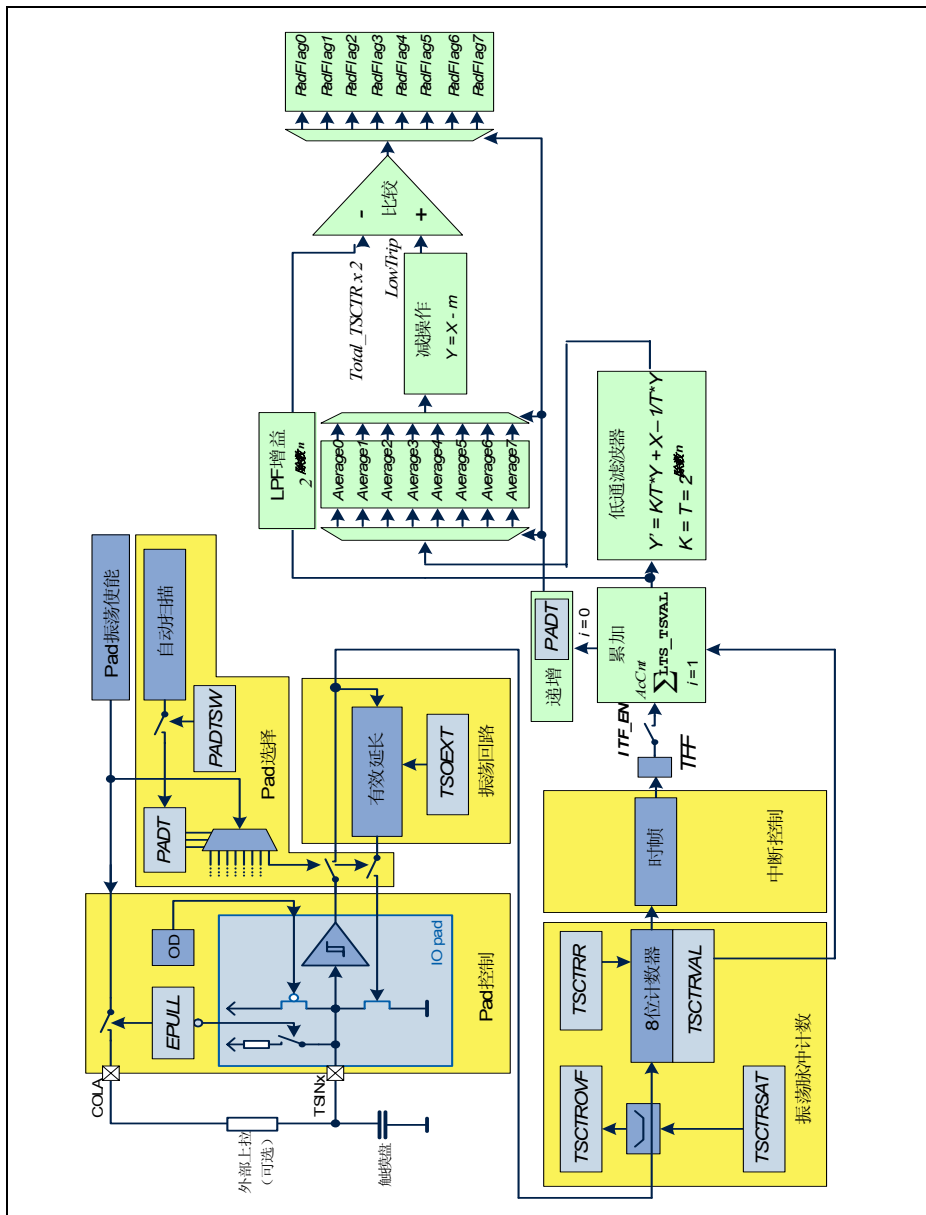


图 24-7 FINDTOUCHEDPAD 函数 - SFR 设置

### 24.2.2.1 FINDTOUCHEDPAD 函数的输入

该函数计算每个 pad turn 的连续平均值以消除 pad 计数频率上的所有毛刺、生成一个可用于计算 trip point 的稳定值。该平均值的计算是基于采样次数的累加值（输入：累加计数器）并包含了低通滤波器增益（输入：除数 n）。平均值减去某数值（输入：减数 m）得到 LowTrip（trip point）。可根据 Low Trip 判断按键是否被触摸。由状态标志（输出：PadResult, PadFlag 和 PadError）指示通过函数检测到的各种结果。当至少有一个 pad 标志被置位（PADFlag 状态 != 0x00）时，则跳过平均值和 Low Trip 计算。

用户可选择每个 pad turn 使用公共的或不同的 trip point 值，这由 IRAM 地址单元 0x32 中的减法选项（subtraction option）来决定。若减法选项为 0x00，则选择公共减数 m 值，在 IRAM 地址 0x34 中初始化该 m 值；若选择不同的减数 m 值，则将减法选项设置为存放不同 m 值的起始地址。用户还可选择公共或不同的振荡窗周期（LTS\_COMPARE），具体参见[章节 24.2.1](#)。

该函数的详细说明列于[表 24-17](#)，其参数说明还可参见[章节 24.2.2.2](#)，函数的具体实现见[章节 24.2.2.4](#) 的描述。

**表 24-17 找到被触摸 touchpad 程序说明**

子程序	FINDTOUCHEDPAD
地址	DFCC <sub>H</sub>
输入	IRAM 地址上的输入参数或 SFR 输入设置 LTS_GLOBCTL0.TS_EN = 1 <sup>1)</sup> 使能触摸感应功能和其相应设置 LTS_TSCTL.PADTSW = 0 用户使用该函数时必须禁止硬件控制 <sup>2)</sup>
IRAM 地址 0x30	<b>累加计数值</b> 用于计算平均值所需累加的采样次数。该值可介于 1-255 之间。
IRAM 地址 0x31	<b>ShortCount</b> 该值和 PadDown 计数器（PDC）的计数值相比较，用以指示置位 PadResult 标志的有效时段。PDC < ShortCount 指示有效时段。一旦检测到某按键被触摸，立即将 PDC 初始化为 0xFF，每个周期 PDC 减 1。
IRAM 地址 0x32	<b>减法选项或减数 m 的地址<sup>3)</sup></b> 该减法选项为每个按键选择公共的或者不同的减数值。若选项为 0x00，则选择公共减数 m 值，该值存放在 IRAM 地址 0x34 中；若选项不为 0x00，则为各按键选择不同的减数值，此时该地址单元中存放不同减数 m 的起始地址。

表 24-17 找到被触摸 touchpad 程序说明

IRAM 地址 0x33	<b>除数 n</b> 用于计算平均值的低通滤波器增益 ( $2^n$ )，n 可取 1, 2, 3, 4, 5, 6, 7, 8，所对应的低通滤波器增益为 2, 4, 8... 256。
IRAM 地址 0x34	<b>减数 m 值：</b> 平均值减去该 m 值计算得到 LowTrip (Trip point) 值。 <b>公共减数 m 值<sup>4)</sup></b> 或
或 0x32 上用户定义的 IRAM 地址 2	<ul style="list-style-type: none"> <li>• 减数 m 值的地址<sup>5)</sup>：PADT0 的减数 m 值</li> <li>• 减数 m 值的地址 +1：PADT1 的减数 m 值 1</li> <li>• 减数 m 值的地址 +2：PADT2 的减数 m 值</li> <li>• <b>减数 m 值的地址 +z：PADTz 的减数 m 值</b></li> </ul> 其中 z 代表被使能 pad turn 的个数
IRAM 地址 0x2D	<b>PadError 状态</b> 若该位为 1，函数将退出相应的 pad turn，不进行任何分析。当位 PadFlag 为 0 时，用户应清除该状态以便进一步分析。
IRAM 地址 0x2E	<b>PadResult 状态</b> 若该位为 1，函数将退出相应的 pad turn，不进行任何分析。当位 PadFlag 为 0 时，用户应清除该状态以便进一步分析。
输出	IRAM 地址上的输出参数：
IRAM 地址 0x2D	<b>PadError 状态</b> 位 0 指示 PADT0 的错误状态 位 1 指示 PADT1 的错误状态 .. 位 7 指示 PADT7 的错误状态 该字节 / 位指示按盘被触摸的时间过长。该字节 / 位不由函数清零，必须由用户清零。只有当 PadFlag 为 0 时才能访问该状态。
IRAM 地址 0x2E	<b>PadResult 状态</b> 位 0 指示 PADT0 的结果状态 位 1 指示 PADT1 的结果状态 .. 位 7 指示 PADT7 的结果状态 7 该字节 / 位指示按盘在有效时段内被触摸。该字节 / 位不由函数清零，必须由用户清零。只有当 PadFlag 为 0 时才能访问该状态。

表 24-17 找到被触摸 touchpad 程序说明

IRAM 地址 0x2F	<b>PadFlag 状态</b> 位 0 指示 PADT0 的标志状态 位 1 指示 PADT1 的标志状态 .. 位 7 指示 PADT7 的标志状态 该字节 / 位为 1 指示按盘正被触摸并在进行分析。分析完毕后，该字节 / 位由函数清零。该字节 / 位由函数置位或清零，用户绝不能访问该位。当相应的 PadFlag 为 0 时，用户可读取 PadError 和 PadResult 的状态。
所需堆栈大小	2
用到的资源	A, R0, R1, R3, R4, R5, R6, R7
	IRAM 地址 0x2D - 0x2F (3 字节)
	XRAM 地址 0xF0EC - 0xF0FF (最多 20 字节) <sup>6)</sup>

- 1) 使能触摸感应控制以及触摸感应 pad turn 的个数和中断标志（时帧 / 时间片）。
- 2) 该函数将控制 pad turn 的个数并通过 LTS\_TSCTL.PADT 进行配制。
- 3) 若用户将该值设置为 0x34（举例），则：IRAM 地址 0x34 上的减数 m 值用于 PADT0；IRAM 地址 0x35 上的减数 m 值用于 PADT1；IRAM 地址 0x36 上的减数 m 值用于 PADT2，依此类推。
- 4) 若所有按盘使用公共的减数 m 值，在 IRAM 地址 0x32 上将减法选项设置为 0x00。
- 5) 在 IRAM 地址 0x32 上定义减数 m 值的起始地址。
- 6) 该函数保存用于计算平均值、Low trip 值和决定触摸状态的参数。切勿修改这些单元中的数据以保证触摸感应算法有效，详见章节 24.2.2.3 的描述。

### Notes:

- 该函数使用固定的 IRAM 地址段 0x2D - 0x34 和 XRAM 地址段 0xF0EC - 0xF0FF。这些地址段不能被其它用户程序设置。检测到出错时，用户可对其清零以重新开始计算。
- 软件必须顺序使能触摸感应 pad turn（如：PADT0, PADT1...）。用户不能使能 3 个 pad turn 并将其定义为 PADT0、PADT4 和 PADT6，必须将它们定义 PADT0、PADT1 和 PADT2。
- pad turn 的保持时间为（累加计数值 +1<sup>1)</sup>），当 TSCTR\_COUNTER<sup>2)</sup>为 0x00 时，函数使 pad turn 加 1。一旦最高使能 pad turn 被置位，函数重新返回 PADT0。例如，若累加计数器定义为 0x04，使能 2 个触摸感应 pad turn，PADTx 的顺序将为 PADT1, PADT1, PADT1, PADT1, PADT0, PADT0, PADT0, PADT0, PADT0, PADT1...。
- 通常在产生时帧中断时调用该函数，但用户也可在产生时间片中断时调用该函数。

1) 附加一个计数周期用于计算平均值。

2) TSCTR\_COUNTER 是存放在 XRAM 中的一个参数，它用于保存用户自定义的输入参数 - 累加计数值。当 LTS\_GLOBCTL1.FNCOL = 0x07（触摸感应功能）每次调用 FINDTOUCHEDPAD 函数时，该值减 1。

---

**ROM Library**

- 该函数会检查 `LTS_GLOBCTL1.FNCOL` 是否为 `0x07`，因此，产生时间片中断时，用户可在函数 `SET_LDLINE_CMP` 之前安全的调用该函数，两个函数之间无需额外的检查代码。该函数在执行其余部分之前将首先检查该时间片中断是否对应于触摸感应功能。若 `TS_GLOBCTL1.FNCOL` 不是 `0x07`，函数将退出。可参见图 24-13。

### 24.2.2.2 FINDTOUCHEDPAD 函数的输出

该函数检测按键是否被触摸并确定其状态，函数输出为 PadFlag、PadResult 和 PadError。

该函数检查按键是否被触摸（当前的计数值是否小于 trip point）并在 PadFlag 状态寄存器中设置相应的标志位。PadFlag 首次被置位时，由软件实现的 Pad 递减计数器（PDC）将被初始化为 0xFF，每个周期<sup>1)</sup>PDC 减 1 直至按键被释放，或直至 PDC 已递减至 0x00。若 PadFlag 状态被置位，则意味着某个按键正在被评估，此时，PadResult 和 PadError 应被忽略。一旦评估完成，PadFlag 状态寄存器中的相应标志位将被清零，结果由 PadResult 和 PadError 状态寄存器给出。

按键是否被触摸分为两种情况：

- 已触摸（PadFlag = 1）
- 未触摸（PadFlag = 0）

被触摸按键分为三种结果：

- 有效触摸（PadResult = 1）
- 无效触摸（短暂触摸 - 忽略）
- 过长时间触摸（PadError = 1）

一旦按键被释放（PDC 的计数值高于 trip point），将 PDC 和预先设定的 ShortCount（输入）阈值进行比较。若高于该阈值，PadFlag 被清零，不再执行其它操作。该情况被视为无效触摸。

若 PDC 低于该阈值但 >0，清除 PadFlag 并置位 PadResult 中相应的标志位，这意味着已识别到成功的触摸操作。但一旦 PDC 递减到 0，则清除 PadFlag 并置位 PadError 中相应的标志位，这意味着触摸时间过长。

由于这些操作占用多个时间片时间，用户程序在读取 PadError 或 PadResult 之前必须检查确保 PadFlag 已清零。PadFlag 非零说明仍在对按键进行分析。

#### PadFlag(F)

Pad Flag 状态（IRAM 地址 0x2F）

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
PadTurn7	PadTurn6	PadTurn5	PadTurn4	PadTurn3	PadTurn2	PadTurn1	PadTurn0
rw	rw	rw	rw	rw	rw	rw	rw

1) 所有使能按键的累加计数周期（AEPACP）。见公式 (24.23)。

## ROM Library

符号	位	类型	描述
<b>PadTurn0</b>	0	rw	<b>Pad Turn 0 的 PadFlag</b> 0B 未触摸按盘 1B 已触摸按盘
<b>PadTurn1</b>	1	rw	<b>Pad Turn 1 的 PadFlag</b> 0B 未触摸按盘 1B 已触摸按盘
<b>PadTurn2</b>	2	rw	<b>Pad Turn 2 的 PadFlag</b> 0B 未触摸按盘 1B 已触摸按盘
<b>PadTurn3</b>	3	rw	<b>Pad Turn 3 的 PadFlag</b> 0B 未触摸按盘 1B 已触摸按盘
<b>PadTurn4</b>	4	rw	<b>Pad Turn 4 的 PadFlag</b> 0B 未触摸按盘 1B 已触摸按盘
<b>PadTurn5</b>	5	rw	<b>Pad Turn 5 的 PadFlag</b> 0B 未触摸按盘 1B 已触摸按盘
<b>PadTurn6</b>	6	rw	<b>Pad Turn 6 的 PadFlag</b> 0B 未触摸按盘 1B 已触摸按盘
<b>PadTurn7</b>	7	rw	<b>Pad Turn 7 的 PadFlag</b> 0B 未触摸按盘 1B 已触摸按盘

**PadResult(R)**
**PadResult 状态 (IRAM 地址 0x2E)**
**复位值: 00<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>PadTurn7</b>	<b>PadTurn6</b>	<b>PadTurn5</b>	<b>PadTurn4</b>	<b>PadTurn3</b>	<b>PadTurn2</b>	<b>PadTurn1</b>	<b>PadTurn0</b>
rw	rw	rw	rw	rw	rw	rw	rw



## ROM Library

符号	位	类型	描述
<b>PadTurn0</b>	0	rw	<b>Pad Turn 0 的 PadResult</b> 0B 触摸按盘无效 1B 触摸按盘有效
<b>PadTurn1</b>	1	rw	<b>Pad Turn 1 的 PadResult</b> 0B 触摸按盘无效 1B 触摸按盘有效
<b>PadTurn2</b>	2	rw	<b>Pad Turn 2 的 PadResult</b> 0B 触摸按盘无效 1B 触摸按盘有效
<b>PadTurn3</b>	3	rw	<b>Pad Turn 3 的 PadResult</b> 0B 触摸按盘无效 1B 触摸按盘有效
<b>PadTurn4</b>	4	rw	<b>Pad Turn 4 的 PadResult</b> 0B 触摸按盘无效 1B 触摸按盘有效
<b>PadTurn5</b>	5	rw	<b>Pad Turn 5 的 PadResult</b> 0B 触摸按盘无效 1B 触摸按盘有效
<b>PadTurn6</b>	6	rw	<b>Pad Turn 6 的 PadResult</b> 0B 触摸按盘无效 1B 触摸按盘有效
<b>PadTurn7</b>	7	rw	<b>Pad Turn 7 的 PadResult</b> 0B 触摸按盘无效 1B 触摸按盘有效

**PadError(E)**
**PadError 状态 (IRAM 地址 0x2D)**
**复位值: 00<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>PadTurn7</b>	<b>PadTurn6</b>	<b>PadTurn5</b>	<b>PadTurn4</b>	<b>PadTurn3</b>	<b>PadTurn2</b>	<b>PadTurn1</b>	<b>PadTurn0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>PadTurn0</b>	0	rw	<b>Pad Turn 0 的 PadError</b> 0B 触摸按盘无误 1B 触摸按盘有误（长按）
<b>PadTurn1</b>	1	rw	<b>Pad Turn 1 的 PadError</b> 0B 触摸按盘无误 1B 触摸按盘有误（长按）
<b>PadTurn2</b>	2	rw	<b>Pad Turn 2 的 PadError</b> 0B 触摸按盘无误 1B 触摸按盘有误（长按）
<b>PadTurn3</b>	3	rw	<b>Pad Turn 3 的 PadError</b> 0B 触摸按盘无误 1B 触摸按盘有误（长按）
<b>PadTurn4</b>	4	rw	<b>Pad Turn 4 的 PadError</b> 0B 触摸按盘无误 1B 触摸按盘有误（长按）
<b>PadTurn5</b>	5	rw	<b>Pad Turn 5 的 PadError</b> 0B 触摸按盘无误 1B 触摸按盘有误（长按）
<b>PadTurn6</b>	6	rw	<b>Pad Turn 6 的 PadError</b> 0B 触摸按盘无误 1B 触摸按盘有误（长按）
<b>PadTurn7</b>	7	rw	<b>Pad Turn 7 的 PadError</b> 0B 触摸按盘无误 1B 触摸按盘有误（长按）

### 24.2.2.3 FINDTOUCHEDPAD 函数的 XRAM 参数

FINDTOUCHEDPAD 函数计算每个 pad turn 的平均值并检测每次触摸为有效、无效或过长时间触摸（由输出状态指示）。该函数保存的参数用于跟踪按盘被触摸的时间长短（PDC）、LTS\_TSVAL 的累加值以及所有 pad turn 的平均值 AverageL/H，函数使用并更新这些参数，用户切勿对其进行修改。表 24-18 列出这些参数、相应的地址以及参数说明。

**表 24-18 存放在 XRAM 中的参数**

参数	地址	说明
Pad 递减计数器 (PDC)	0xF0FF	PDC <sup>1)</sup> 用于跟踪按键被触摸的时间长短。一旦检测到触摸操作, PDC 被初始化为 0xFF, 每个周期 <sup>2)</sup> PDC 减 1。按键被释放后, 将 PDC 和输入值 ShortCount 和 / 或 0x00 比较以确定该次触摸为有效、无效或过长时间模式。
TOTAL_TSCTR (高字节)	0xF0FE	TOTAL_TSCTRL/H <sup>3)</sup> 是累加触摸感应计数值 (LTS_TSVAL)。该值将用于计算平均值并和 LowTrip 值 <sup>4)</sup> 进行比较。
TOTAL_TSCTR (低字节)	0xF0FD	
TSCTR_Counter	0xF0FC	TSCTR_Counter <sup>5)</sup> 基于输入值 “累加计数器” 跟踪 LTS_TSVAL 的累加值。
Average0 (高字节)	0xF0FB	pad turn 0 (PADT0) 的平均值 <sup>6)</sup>
Average0 (低字节)	0xF0FA	
Average1 (高字节)	0xF0F9	pad turn 1 (PADT1) 的平均值 <sup>6)</sup>
Average1 (低字节)	0xF0F8	
Average2 (高字节)	0xF0F7	pad turn 2 (PADT2) 的平均值 <sup>6)</sup>
Average2 (低字节)	0xF0F6	
Average3 (高字节)	0xF0F5	pad turn 3 (PADT3) 的平均值 <sup>6)</sup>
Average3 (低字节)	0xF0F4	
Average4 (高字节)	0xF0F3	pad turn 4 (PADT4) 的平均值 <sup>6)</sup>
Average4 (低字节)	0xF0F2	
Average5 (高字节)	0xF0F1	pad turn 5 (PADT5) 的平均值 <sup>6)</sup>
Average5 (低字节)	0xF0F0	
Average6 (高字节)	0xF0EF	pad turn 6 (PADT6) 的平均值 <sup>6)</sup>
Average6 (低字节)	0xF0EE	
Average7 (高字节)	0xF0ED	pad turn 7 (PADT7) 的平均值 <sup>6)</sup>
Average7 (低字节)	0xF0EC	

1) 可修改 PDC 来调整检测周期以产生 PadError 或 PadResult 结果。

2) 该周期指所有使能按键的累加计数周期。参见公式 (24.23)。

3) TSCTR\_Counter 不为 0x00 时, 它意味着累加未结束, 当前的 TOTAL\_TSCTRL/H 数值还不是总触摸感应计数值。

4) 具体实现请参见章节 24.2.2.4。

5) TSCTR\_Counter 为 0x00 时, TOTAL\_TSCTRL/H 的数值 即和 LowTrip 比较, 从而确定是否检测到触摸操作或按键是否被释放。PadFlag 为 0 后, 发生下一次时间片 / 时帧中断时, 该值将用于计算相应的平均值。

6) 始终更新平均值，PadFlag 不为 0x00 的情况除外（检测到触摸操作并在处理中）。有关平均值的计算，参见 [章节 24.2.2.4](#) 的描述。用于存放平均值的 XRAM 地址个数和使能 pad turn 的个数有关，8 个平均值最多 16 个地址。若 pad turn 未使能，则不使用相应的地址。

#### 24.2.2.4 函数的具体实现

函数中用到的主要公式包括：如何得到 Total\_TSCTRL/H，如何计算平均值，如何计算 LowTrip（Trip point）以及如何进行比较以判断按键是否被触摸。使用 SFR LTS\_TSVAL 的值进行计算。

在输入参数 “AccumulatorCounter（累加计数值）” 中定义需要累加的总采样次数。所有采样的总数值（TOTAL\_TSCTRL/H）是 SFR LTS\_TSVAL 数值的累加结果。

$$\text{TOTAL\_TSCTRL/H}(x) = \sum_{i=1}^{\text{AccumulatorCounter}} \text{LTS\_TSVAL}(i) \quad (24.14)$$

其中  $i$  是用户定义的输入参数 - 累加计数值（采样次数），该值可介于 1-255 之间。

平均值的计算见 [公式 \(24.15\)](#)。

$$\text{AVERAGEL/H}(x) = \text{AVERAGEL/H}(x-1) + \text{TOTAL\_TSCTRL/H}(x) - \frac{\text{AVERAGEL/H}(x-1)}{2^n} \quad (24.15)$$

其中  $n$  是用户定义的输入参数 - 除数  $n$ ，该值可取 1 - 8。

平均值减去用户定义的输入参数 - 减数  $m$ ，即得到 LowTrip（Trip point）的值，具体计算见 [公式 \(24.16\)](#)。

$$\text{LOWTRIPL/H}(x) = \text{AVERAGEL/H}(x) - \text{SUBTRACTION}_m \quad (24.16)$$

将计算得到的 LowTrip 值和当前累加的 LTS\_TSVAL 值，即 TOTAL\_TSCTRL/H \*  $2^n$  进行比较。

当前累加的 LTS\_TSVAL 小于 LowTrip 时，则识别按键正被触摸，见 [公式 \(24.17\)](#)：

$$\text{TOTAL\_TSCTRL/H}(x) \times 2^n < \text{LOWTRIPL/H}(x) \quad (24.17)$$

当前累加的 LTS\_TSVAL 大于 LowTrip 时，则识别按盘未被触摸或已被释放（触摸完毕），见公式 (24.18):

$$(24.18)$$

$$TOTAL\_TSCTRL/H(x) \times 2^n \geq LOWTRIP/L/H(x)$$

一旦识别到按盘正被触摸，计数器 PDC 初始化到 0xFF 并开始递减计数直至触摸完毕。该情况下，将 PDC 和 ShortCount（输入参数）进行比较，从而判断触摸是否有效。

### 内部状态

由于触摸感应功能的时分复用特性，该函数被调用时可处于不同的状态，其状态可由该函数本身或由用户程序来修改。共有 5 种内部状态：空闲、按盘被触摸、按盘被释放、Error 以及 Result 状态。图 24-8 示出该函数所处的不同状态以及状态之间的转移。

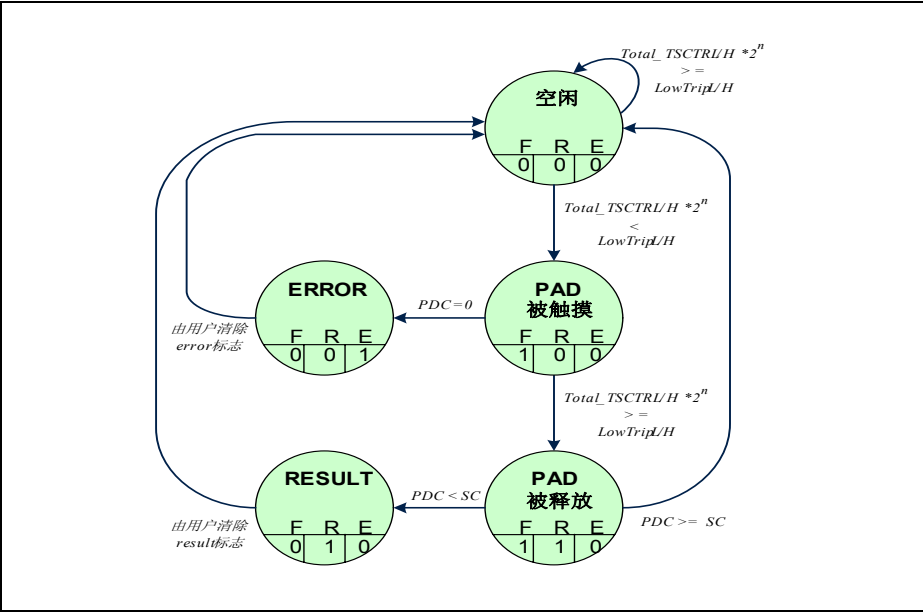


图 24-8 状态图

进入某状态的操作:

空闲:

进入: 检查  $Total\_TSCTRL/H \times 2^n < LowTripL/H$  是否成立

退出: 若成立，置位相应的 PadFlag 位，启动计数器（PDC）工作，PDC 预设值为 0xFF。

按盘被触摸:

进入: PDC 递减计数, 检查 PDC 是否等于 0, 检查  $\text{Total\_TSCTRL}/H * 2^n < \text{LowTripL}/H$  point 是否成立

退出: 根据检查的结果置位相应的 PadError 标志、清除 PadFlag 标志或置位 PadResult 标志

按盘被释放:

进入: 检查  $\text{PDC} \geq \text{short count}$  是否成立

退出: 若  $\text{PDC} \geq \text{short count}$ , 清除所有标志; 若  $\text{PDC} < \text{short count}$ , 清除 PadFlag 标志

Pad Result:

进入: 检查 PadResult 标志是否由用户清零, 若未清零, 退出

退出: 空闲

Pad Error:

进入: 检查 PadError 标志是否由用户清零, 若未清零, 退出

退出: 空闲

### Pad 递减计数器 - PDC

函数一旦检测到按盘被触摸, 则将 PDC 初始化为 0xFF。每个周期<sup>1)</sup>函数使 PDC 减 1, 直至按盘被释放<sup>2)</sup>。对于两个按盘被触摸的情况, 在它们的分析阶段, PDC 仅递减一次 (可检测出 PDC 之前已递减过)。图 24-9 示出 PDC 如何工作。PDC 是一个软件递减计数器, 以周期<sup>1)</sup>作为计数时钟。

1) 所有使能按盘的累加计数周期 (AEPACP)。见公式 (24.23)。

2) 或直至 PDC 递减到 0x00。

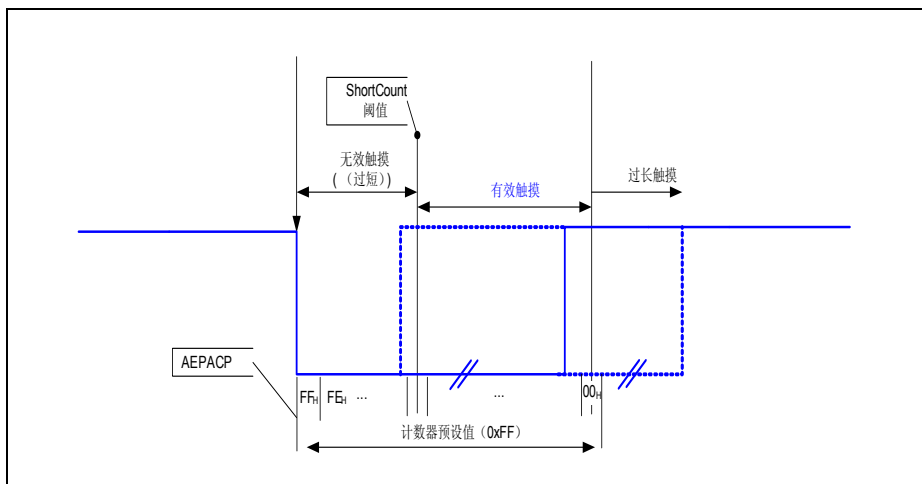


图 24-9 Pad 递减计数器 - PDC

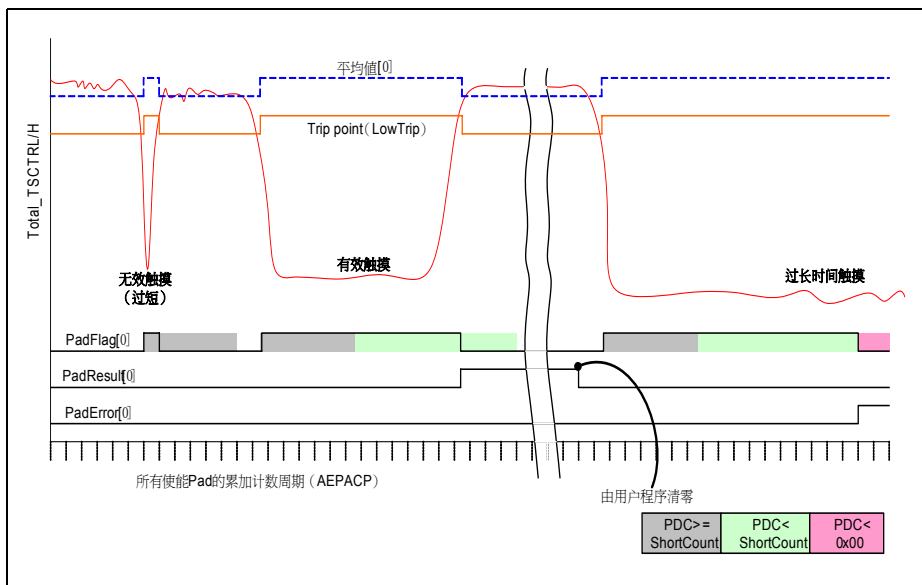


图 24-10 时序图

**图 24-10** 以 Pad turn 0 为例说明 PadFlag、PadResult、PadError 和 PDC 的状态及相互关系。Trip point (LowTrip) 可以固定, 或如上图所示, 一旦 padflag 被置位时可增加一定的滞后 (通过改变用户输入减数 m 来实现)。

第一个 PadFlag 被置位时, 向 PDC 载入其超时值。在随后的 Pad turn 中, 该计数器递减计数直至到 0, 或直至所有的 PadFlag 位被清零后 PDC 置 0。若某个 Pad turn 上次保持置位的 PadFlag 被清零, 退出该函数时 PDC 置 0。若按盘的触摸时间过短 (short count) 或在 PDC 超时之前按盘被释放, 可发生这样的情况。

有一种容易被忽略的情况, 即在触摸另一个按盘之前用户未清除上一个按盘的 PadError 或 PadResult 标志。在这种情况下 (例如: 上一个被触摸的按盘是个组合按盘, 下一个被触摸的按盘是个单一按盘, 其按盘独立于上一个组合按盘), 会有两个以上的 PadResult 标志置位。

PDC 每个周期 (所有使能按盘的累加计数周期) 递减一次, 即使两个 PadFlag 位同时置位。

### 时间片、时帧和周期的定义

下面给出时间片、时帧和周期的定义。周期的概念包括硬件控制方案的周期和软件控制 ROM Library 方案的周期 (单个按盘累加计数周期和所有使能按盘累加计数周期)。触摸感应功能使能时, 时间片、时帧和周期的计算由以下公式给出, 它们之间的关系可参见 **图 24-11** 的示例。

时间片长度 (TSD) 的定义见 **公式 (24.19)**:

$$\text{Time Slice Duration} = \frac{\text{Prescaler} \times 256}{f_{\text{CLK}}} \quad (24.19)$$

其中 prescaler 为 LEDTS 计数器时钟的预分频因子 (LTS\_GLOBCLK.CLK\_PS), 输入时钟  $f_{\text{CLK}}$  为 8 MHz 或 24 MHz, 这和 USER\_ID 的设置有关。

时帧长度 (TFD) 的定义见 **公式 (24.20)**:

$$\text{Time Frame Duration} = (\text{Number of Time Slice(s)}) \times \text{TSD} \quad (24.20)$$



硬件控制方案中周期的定义见公式 (24.21):

(24.21)

$$\text{Period}_{(\text{Hardware-Controlled Scheme})} = (\text{Number of Touch-sense input(s) TSIN}[x]) \times \text{TFD}$$

软件控制ROM Library方案中, 单个按盘累加计数周期 (SPACP) 的定义见公式 (24.22):

(24.22)

$$\text{Period}_{(\text{SPAC})} = (\text{AccumulatorCounter} + 1) \times \text{TFD}$$

其中累加计数值为函数 FINDTOUCHEDPAD 的用户自定义输入参数。

软件控制 ROM Library 方案中, 所有使能按盘累加计数周期 (AEPACP) 的定义见公式 (24.23):

(24.23)

$$\text{Period}_{(\text{AEPAC})} = \text{Period}_{(\text{SPAC})} \times (\text{Number of Touch-sense input(s) TSIN}[x])$$

## ROM Library

按键被触摸时，PDC 每个 AEPACP 减 1。按键被释放后，检查 PDC 的值。若  $PDC < ShortCount$ ，则触摸有效；若 PDC 已递减至 0x00，则触摸时间过长。最短和最有效 pad 检测周期的定义分别见 公式 (24.24) 和 公式 (24.25)。

(24.24)

$$\text{Minimum Valid Pad Detection Period (VPDP)} = (0xFF - ShortCount + 1) \times AEPACP$$

(24.25)

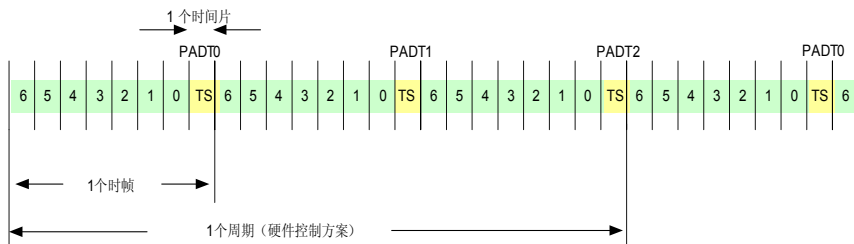
$$\text{Maximum Valid Pad Detection Period (VPDP)} = 0xFF \times AEPACP$$

用户可引入 ErrorCount 来缩短最长有效 pad 检测周期（以获得更快输出 PadError=1），见 公式 (24.26)。

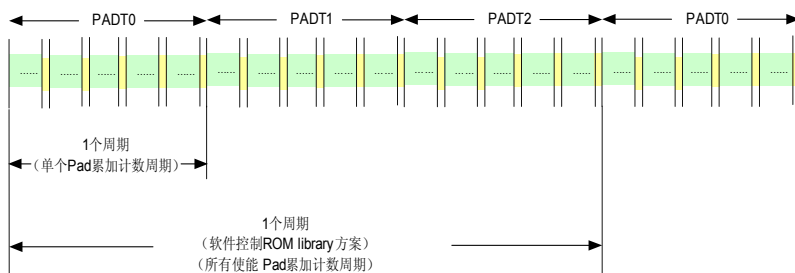
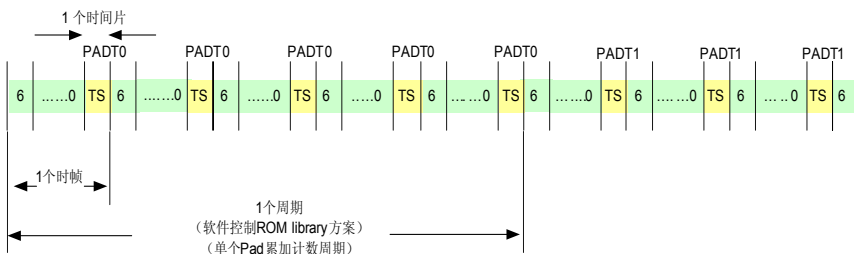
(24.26)

$$\text{Maximum Valid Pad Detection Period (VPDP)} = (0xFF - ErrorCount + 1) \times AEPACP$$

举例：使能7列LED和3个触摸感应Pad Turn



举例：使能7列LED和3个触摸感应Pad Turn，累加计数值 = 0x04



某个Pad被触摸时，PDC在一个周期内（所有使能PADDx累加计数周期）递减1次，  
在组合Pad的情况下（2个Pad被触摸），PDC在一个周期内（所有使能PADDx累加计数周期）仍递减1次

图 24-11 软件控制 ROM Library 方案中的周期定义

### 24.2.3 中断时函数的使用

根据 LED 或 / 和触摸感应功能的使能情况，产生时间片或 时帧中断时可调用这些函数（前提是函数的输入参数已准备就绪）。图 24-12和图 24-13示出在各种情况下用户如何调用相应的函数。

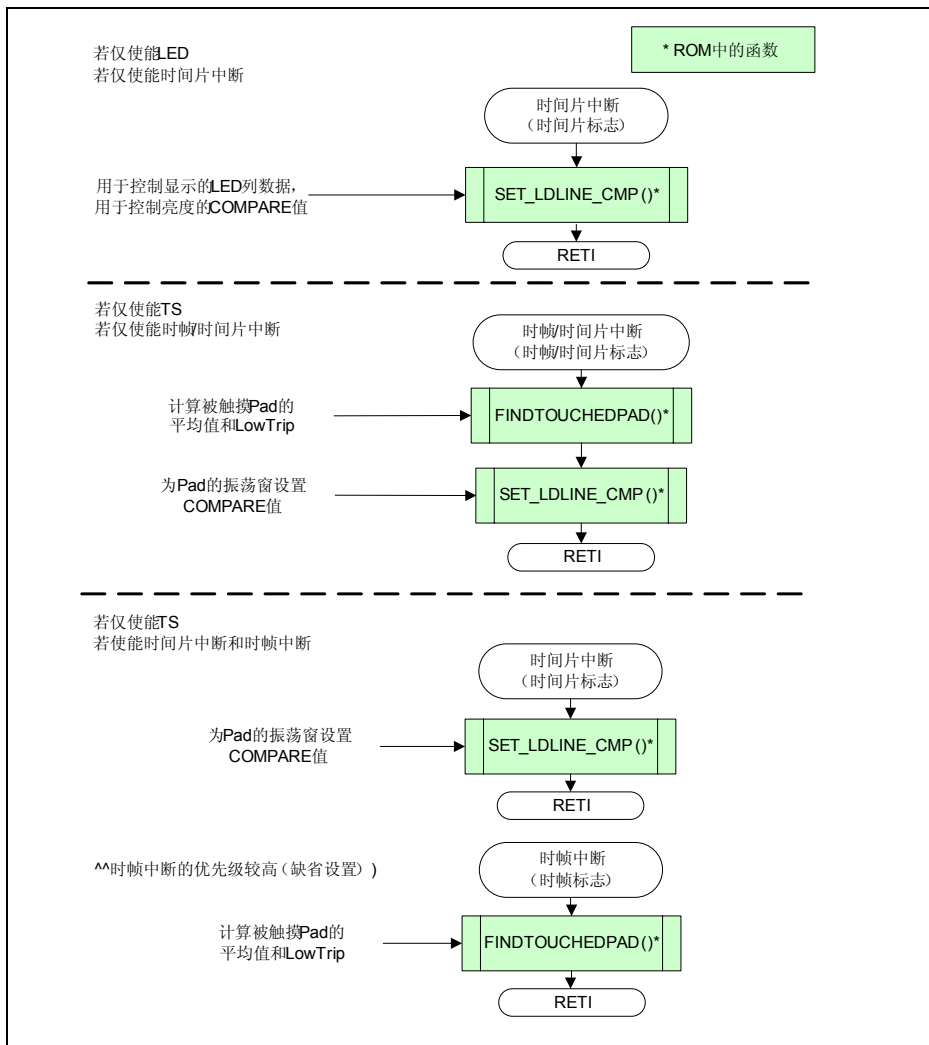


图 24-12 中断服务程序中函数的调用 (i)

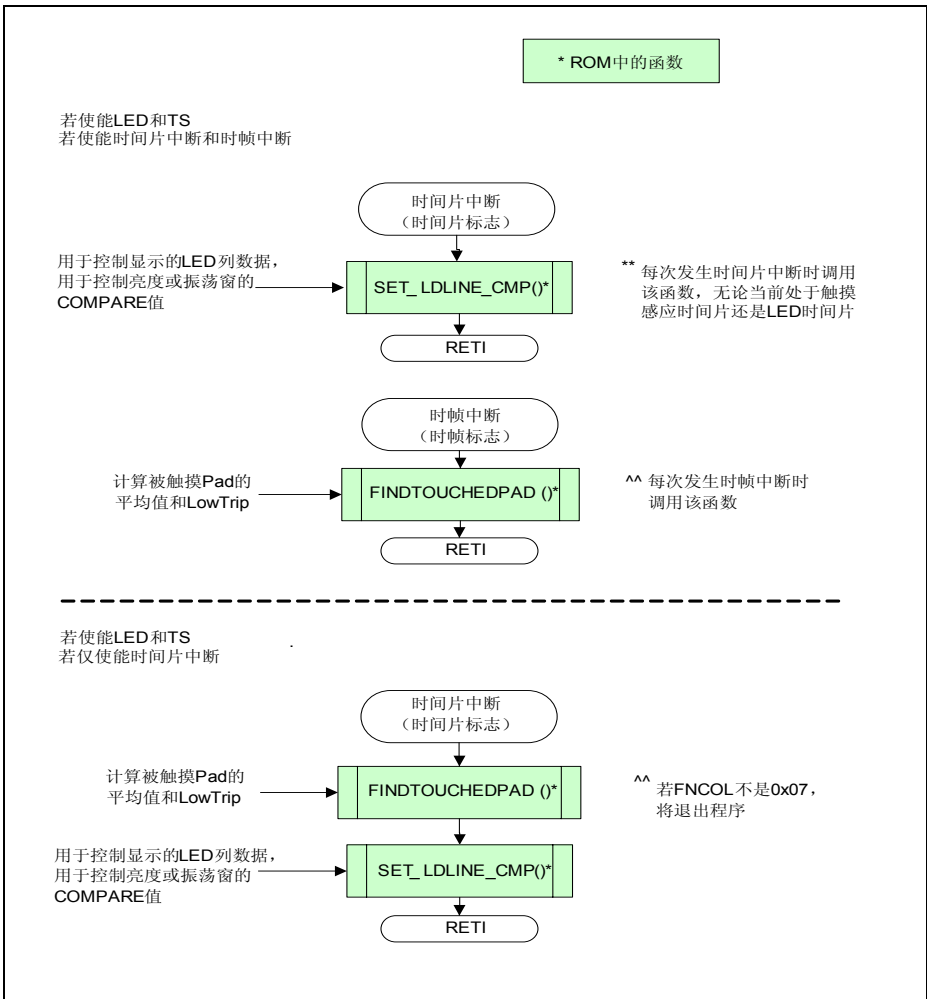


图 24-13 中断服务程序中函数的调用 (ii)

## 24.3 EEPROM 模拟 ROM Library

XC83x 通过 ROM library 为用户提供了 EEPROM 模拟功能。使用片上 Flash 模拟 EEPROM。ROM library 提供了一种访问模拟 EEPROM 的程序框架。

针对该程序框架的说明，可参阅应用指南“AP0805710 XC866/XC886/XC888 EEPROM 模拟”。

### 24.3.1 特性

EEPROM 模拟 ROM library 具有以下特性：

- 提供初始化、修改和读 / 写模拟 EEPROM 的函数
- 支持模拟 EEPROM 的大小：31、62、93 和 124 字节
- 高达 160 万次的耐擦写周期 @ 2 年的数据保持时间

### 24.3.2 系统要求

ROM library 有以下要求：

- 使用 Flash 地址段 AE00<sub>H</sub> - AFFF<sub>H</sub> 共 512 个字节进行 EEPROM 模拟<sup>1)</sup>
- 支持 KEIL C51 工具链（小存储器模型，大端，寄存器调用规则）
- 只支持基于查询的 Flash 操作<sup>2)</sup>
- 不支持编程退出检测

### 24.3.3 概念

使用 Flash 中的 512 个字节（分为两个逻辑扇区）来模拟 EEPROM。每次仅有一个逻辑扇区为“有效扇区”，即该扇区内包含着最新数据。用户程序在模拟 EEPROM 之前必须先进行相应的初始化，这包括调用初始化函数并指定模拟方案。初始化函数将更新 [页 24-48 “EEPROMInfo 数据结构”](#) 中的信息（ROM library 函数将使用该信息）。

ROM library 支持 4 种模拟方案，即模式 0（Mode\_0）、模式 1（Mode\_1）、模式 2（Mode\_2）和模式 3（Mode\_3）。每种模式代表一种模拟 EEPROM 的大小。

用户程序通过逻辑地址访问模拟 EEPROM。有效逻辑地址的数目和模拟 EEPROM 的大小有关。每个逻辑地址代表 32 个数据字节，其中 31 个用户数据字节和 1 个状态字节，这对应一次 Flash 写操作的大小，即 Flash 字线（WL）大小。

逻辑地址提供了一种方便的抽象，使用户程序无需处理 Flash 的实际物理地址。

ROM library 通过状态字节来管理 EEPROM 模拟。向每条 Flash 字线的最后一个字节中写入状态字节。用户程序可安全的忽略该字节。

1) 对于 8K 的器件而言，使用 Flash 地址段 1E00<sub>H</sub> - 1FFF<sub>H</sub> 来模拟 EEPROM；对于 4K 的器件而言，使用 Flash 地址段 0E00<sub>H</sub> - 0FFF<sub>H</sub> 来模拟 EEPROM。

2) 使用 Boot ROM flash 用户程序（非后台），包括 NMI 在内的所有中断必须禁用。

表 24-19      EEPROM 模拟方案

模拟方案	模拟 EEPROM 大小（字节）	有效逻辑地址
Mode_0	31 数据 + 1 状态	WL_0
Mode_1	62 数据 + 2 状态	WL_0 & WL_1
Mode_2	93 数据 + 3 状态	WL_0, WL_1 & WL_2
Mode_3	124 数据 + 4 状态	WL_0, WL_1, WL_2 & WL_3

表 24-20      逻辑地址和数据字节的关系

逻辑地址	数据字节
WL_0	0 – 30
WL_1	31 – 61
WL_2	62 – 92
WL_3	93 – 124

初始化函数还可进行一些基本的检错操作。它可检测 Flash 擦除操作期间是否发生退出、以及模拟 EEPROM 是否首次被使用。InitEEPROM() 假定：在首次调用该函数时，用于模拟 EEPROM 的 Flash 区段已被擦除。

用户程序使用逻辑地址通过读写函数访问模拟 EEPROM。每次读操作将读取 31 个用户数据字节和 1 个状态字节，这些字节将保存在由调用函数指定的 IRAM 地址中。

类似的，每次写操作只能写入 31 个用户数据字节和 1 个状态字节，这些待写入的字节应保存在由调用函数指定的 IRAM 地址中。状态字节将由写函数自动更新。

一旦“有效扇区”中填满数据，随后的写操作将指向下一个逻辑扇区。下一个逻辑扇区将成为“有效扇区”，上一个“有效扇区”被收回。收回操作包括：在擦除上一个“有效扇区”之前，将其中的所有有效数据写入当前的“有效扇区”中。

要更新少于 31 个字节的数据时，用户程序必须先从存放这些数据的逻辑地址上读出该数据。一旦读出的数据被保存在 IRAM 中，用户程序将会更新 IRAM 中的相关数据，随后调用写函数将新数据写入模拟 EEPROM。

### 24.3.4      API 说明

模拟 EEPROM 的操作由一个数据结构和四个函数控制。此外还给出一些常量的定义以便于使用。

#### 24.3.4.1    常量定义

函数为多个常量定义了有意义的名称，这样可提高代码的可读性。

#### EEPROM 模拟方案

- #define MODE\_0 32
- #define MODE\_1 64
- #define MODE\_2 96
- #define MODE\_3 128

逻辑地址

- #define WL\_0 0
- #define WL\_1 1
- #define WL\_2 2
- #define WL\_3 3

#### 24.3.4.2 EEPROMInfo 数据结构

EEPROM 模拟 API 采用一个称为 *EEPROMInfo* 的数据结构来管理 EEPROM 模拟操作。

```
typedef struct EEPROMInfo{  
    unsigned int ActiveSector;  
    unsigned int WriteAddress;  
    unsigned char DataSize;  
};data EEPROMInfo;
```

*EEPROMInfo* 结构中包含 3 个变量: *ActiveSector*、*WriteAddress* 和 *DataSize*。该结构必须由用户程序声明并保存在间接 IRAM 地址中。

*ActiveSector* 存放当前“有效扇区”的地址, *ReadEEPROM()* 将使用该变量以定位当前模拟的数据组。*WriteAddress* 存放数据写入的起始地址, 用户程序调用 *WriteEEPROM()* 时将使用该变量。调用 *InitEEPROM()* 和 *WriteEEPROM()* 时更新这两个变量。*DataSize* 存放 EEPROM 模拟的数据组大小。

*EEPROMInfo* 不能由用户直接修改, 它只能由 EEPROM 模拟 API 提供的函数来修改。

#### 24.3.4.3 函数

EEPROM 模拟 API 提供以下 4 个函数:

- InitEEPROM
- FixEEPROM
- WriteEEPROM
- ReadEEPROM

在访问模拟 EEPROM 时必须使用这些函数。

#### InitEEPROM



表 24-21 InitEEPROM

函数	InitEEPROM
函数原型	unsigned char InitEEPROM(unsigned char mode, EEPROMInfo *config)
输入	mode - 模拟方案，即模拟 EEPROM 的大小，有效值为 32, 64, 96 和 128 config - 指向 <i>EEPROMInfo</i> 数据结构的指针
返回	status - 模拟 EEPROM 的状态。输入到 FixEEPROM() 0x0 - 模拟 EEPROM 的状态 OK 0x1 - 扇区 8 & 9 需要擦除 0x2 - 扇区 6 & 7 需要擦除 0x3 - 扇区 6 - 9 需要擦除
最大堆栈大小	4 字节

InitEEPROM() 将根据选定的模拟方案初始化 EEPROMInfo 数据结构。在使用其它 API 函数之前至少需要调用一次该函数。InitEEPROM() 假定：在首次调用该函数时，用于模拟 EEPROM 的 Flash 区段处于擦除状态。

InitEEPROM() 将会更新 EEPROM 配置，指示当前的“有效扇区”和随后调用 WriteEEPROM() 函数时数据将要写入的 Flash 地址。

包含有效状态字节的扇区被选作“有效扇区”。若两个逻辑扇区均包含有效的状态字节，包含有效状态字节数目最少的逻辑扇区被选作“有效扇区”。此外，InitEEPROM() 还将返回错误状态，请求对包含最多有效状态字节的扇区进行擦除。不过会有一种不期望的情况：在 WriteEEPROM() 回收操作期间突然断电。

以 124 字节 EEPROM 模拟为例：若在 WriteEEPROM() 回收操作期间突然断电，此时并非 124 字节的有效信息全部被复制到新扇区中。

InitEEPROM() 还将检查擦除操作是否意外退出并返回状态字节指示需要重新初始化的扇区。

## FixEEPROM

表 24-22 FixEEPROM

函数	FixEEPROM
函数原型	void FixEEPROM(unsigned char sector_status, unsigned char idata *buffer)

表 24-22      FixEEPROM

输入	sector_status - 由 InitEEPROM() 返回的状态
	buffer - 指向 IRAM 中 32 字节缓存的指针
返回	无
最大堆栈大小	4 字节 + 12 字节（Boot ROM 用户程序）

FixEEPROM() 将根据 InitEEPROM() 返回的状态字节来修正无效扇区。它将擦除相应的 Flash 扇区并设置状态字节以指示该扇区已被擦除。

需要通过指向 IRAM 中 32 字节缓存的指针来实现 Flash 写操作。在调用该函数之前，要确保 NMISR SFR 中的标志位被清零。

## WriteEEPROM

表 24-23 WriteEEPROM

函数	WriteEEPROM
函数原型	void WriteEEPROM(unsigned char address, char idata * src, EEPROMInfo *config)
输入	address - 数据写入的逻辑地址
	src - 指向 IRAM 中 32 字节待写入数据的指针
	config - 指向 <i>EEPROMInfo</i> 数据结构的指针
返回	无
最大堆栈大小	6 字节 + 12 字节 (Boot ROM 用户程序)

WriteEEPROM() 根据 EEPROM 的配置进行写操作，将数据写入有效的 Flash 地址单元并更新 EEPROM 配置。一次写入 32 字节 (31 个数据字节 + 1 个状态字节)。

此外，当上一个“有效扇区”填满数据后，该函数将“收回”该扇区。在调用该函数之前，要确保 NMISR SFR 中的标志位被清零。

## ReadEEPROM

表 24-24 ReadEEPROM

函数	ReadEEPROM
函数原型	unsigned char ReadEEPROM(unsigned char address, char idata * dst, EEPROMInfo *config)
输入	address - 读取数据的逻辑地址
	dst - 指向 IRAM 中保存读取数据的 32 字节缓存的指针
	config - 指向 <i>EEPROMInfo</i> 数据结构的指针
输出	status - 读操作的结果
	0x00 - 读操作成功
	0xFF - 读操作失败
最大堆栈大小	2 字节

ReadEEPROM() 将在当前的“有效扇区”中根据给定的“地址”寻找最新数据，它将始终返回 32 字节的数据，即 31 个数据字节 +1 个状态字节。

“地址”取值介于 0-3 之间。若数据组大小为 32 字节，则仅地址 0 有效；若数据组大小为 128 字节，则地址 0-3 均有效，地址 2 将返回字节 64-95，字节 95 为状态字节。

若无法找到地址，函数将返回错误代码。

#### 24.3.4.4 API 使用示例

下面以 C 语言举例说明如何使用 EEPROM 模拟 API，该举例示出如何初始化 Flash 区段并给出一些错误处理的方式。

```
#include "EEPROM.h"

void main(void)
{
    /*EEPROM 模拟所需的数据结构 */
    char idata buffer[32];
    EEPROMInfo config;
    unsigned char eeprom_status;

    /* 初始化并检查模拟 EEPROM 的完整性 */
    eeprom_status = InitEEPROM(MODE_3, &config);
    if(0 != eeprom_status){
        if( 0x0 == config.ActiveSector){
            /* 检测到擦除操作退出或首次使用 EEPROM*/
            /* 初始化无效扇区 */
            FixEEPROM(eeprom_status, buffer);
        }else{
            /* 在收回操作期间突然发生编程或擦除退出 */

            /* 根据用户的实际应用，可执行其它操作 */

            /* 初始化无效扇区。数据将被擦除 */
            FixEEPROM(eeprom_status, buffer);
        }

        /* 检查 EEPROM 扇区是否处于有效状态 */
        eeprom_status = InitEEPROM(MODE_3, &config);
    }
}
```

```
if (0 != eeprom_status){
    /*EEPROM 出现致命错误 */
    while(1);
}

}

/* 用户程序 */

/* 读取和写入模拟 EEPROM 举例 */
/* 读取当前数据 */
ReadEEPROM(WL_1, &buffer, &config);
/* 更新待写入 EEPROM 的数据 */
buffer[20] += 1;
/* 写回更新数据 */
WriteEEPROM(WL_1, &buffer, &config);
while(1); /* 等待断电 */
}
```

[www.infineon.com](http://www.infineon.com)