

8 位

**XC878CLM**

8 位单片微控制器

用户手册

V1.1 2009-04

**Microcontrollers**

**Edition 2009-04**

**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

## XC878CLM

### 8 位单片微控制器

## 用户手册

### V1.1 2009-04

# Microcontrollers

<b>1</b>	<b>简介</b>	<b>1-1</b>
1.1	特性总结	1-4
1.2	引脚配置	1-6
1.3	引脚定义及功能	1-7
1.4	芯片 ID	1-18
1.5	命名规则	1-19
1.6	保留位、未定义位及未实现位	1-19
1.7	缩写	1-20
<b>2</b>	<b>处理器结构</b>	<b>2-1</b>
2.1	功能概述	2-1
2.2	CPU 寄存器描述	2-2
2.2.1	堆栈指针 (SP)	2-3
2.2.2	数据指针 (DPTR)	2-3
2.2.3	累加器 (ACC)	2-3
2.2.4	B 寄存器	2-3
2.2.5	程序状态字	2-4
2.2.6	扩展操作寄存器 (EO)	2-5
2.2.7	功率控制寄存器 (PCON)	2-6
2.3	存储器扩展	2-7
2.3.1	存储器扩展堆栈	2-7
2.3.2	存储器扩展的注意事项	2-7
2.3.3	存储器扩展寄存器	2-9
2.4	指令时序	2-11
<b>3</b>	<b>存储器结构</b>	<b>3-1</b>
3.1	程序存储器	3-3
3.2	数据存储器	3-3
3.2.1	内部数据存储器	3-3
3.2.2	外部数据存储器	3-4
3.2.2.1	访问外部数据存储器	3-4
3.2.2.2	外部数据存储器接口	3-6
3.3	存储器保护策略	3-8
3.3.1	Flash 存储器保护	3-8
3.3.2	其它控制寄存器	3-11
3.4	特殊功能寄存器	3-12
3.4.1	映射地址扩展	3-12
3.4.1.1	系统控制寄存器 0	3-14
3.4.2	分页地址扩展	3-15
3.4.2.1	分页寄存器	3-17
3.4.3	位寻址	3-18
3.4.4	系统控制寄存器	3-19
3.4.4.1	位保护方案	3-21
3.4.5	XC878 寄存器概览	3-22



3.4.5.1	CPU 寄存器 .....	3-22
3.4.5.2	MDU 寄存器 .....	3-24
3.4.5.3	CORDIC 寄存器 .....	3-25
3.4.5.4	系统控制寄存器 .....	3-25
3.4.5.5	WDT 寄存器 .....	3-28
3.4.5.6	端口寄存器 .....	3-29
3.4.5.7	ADC 寄存器 .....	3-31
3.4.5.8	定时器 T2 比较 / 捕获单元寄存器 .....	3-34
3.4.5.9	定时器 T21 寄存器 .....	3-37
3.4.5.10	CCU6 寄存器 .....	3-38
3.4.5.11	UART1 寄存器 .....	3-42
3.4.5.12	SSC 寄存器 .....	3-42
3.4.5.13	MultiCAN 寄存器 .....	3-43
3.4.5.14	OCDS 寄存器 .....	3-44
3.4.5.15	Flash 寄存器 .....	3-45
3.5	Boot ROM 工作模式 .....	3-45
3.5.1	用户模式 .....	3-47
3.5.2	引导程序加载器模式 .....	3-48
3.5.3	OCDS 模式 .....	3-48
3.5.4	用户 JTAG 模式 .....	3-48
<b>4</b>	<b>闪存存储器 .....</b>	<b>4-1</b>
4.1	Flash 存储器映射 .....	4-2
4.2	Flash 存储块分页 .....	4-2
4.3	工作模式 .....	4-2
4.3.1	读操作 .....	4-3
4.3.2	编程操作 .....	4-3
4.3.3	擦除操作 .....	4-3
4.3.4	整体擦除操作 .....	4-4
4.3.5	中止操作 .....	4-4
4.4	Flash 定时器 .....	4-6
4.5	检错和纠错 .....	4-7
4.6	在系统编程 .....	4-8
4.7	在应用编程 .....	4-9
4.7.1	Flash 编程 .....	4-9
4.7.2	Flash 擦除 .....	4-10
4.7.3	获取芯片信息 .....	4-11
4.8	寄存器映射 .....	4-13
4.9	寄存器描述 .....	4-13
<b>5</b>	<b>中断系统 .....</b>	<b>5-1</b>
5.1	中断结构 .....	5-8
5.1.1	中断结构 1 .....	5-8
5.1.2	中断结构 2 .....	5-9

5.1.2.1	系统控制寄存器 0	5-10
5.2	中断源和中断向量	5-11
5.3	中断优先级	5-13
5.4	中断处理	5-14
5.5	中断响应时间	5-15
5.6	中断寄存器	5-17
5.6.1	中断节点使能寄存器	5-17
5.6.2	外部中断控制寄存器	5-21
5.6.3	中断标志寄存器	5-26
5.6.4	中断优先级寄存器	5-33
5.7	中断标志概览	5-36
<b>6</b>	<b>并行端口</b>	<b>6-1</b>
6.1	基本端口操作	6-2
6.1.1	通用寄存器描述	6-4
6.1.1.1	数据寄存器	6-5
6.1.1.2	方向寄存器	6-6
6.1.1.3	漏极开路控制寄存器	6-6
6.1.1.4	上拉 / 下拉器件寄存器	6-7
6.1.1.5	复用输入和输出功能	6-9
6.1.1.6	驱动能力控制寄存器	6-10
6.2	寄存器地址映射	6-11
6.3	P0 口	6-13
6.3.1	功能	6-13
6.3.2	寄存器描述	6-17
6.4	P1 口	6-20
6.4.1	功能	6-20
6.4.2	寄存器描述	6-24
6.5	P3 口	6-27
6.5.1	功能	6-27
6.5.2	寄存器描述	6-31
6.6	P4 口	6-35
6.6.1	功能	6-35
6.6.2	寄存器描述	6-39
6.7	P5 口	6-42
6.7.1	功能	6-42
6.7.2	寄存器描述	6-46
<b>7</b>	<b>电源、复位和时钟管理</b>	<b>7-1</b>
7.1	内嵌电压调节器的电源系统	7-1
7.2	复位控制	7-3
7.2.1	复位类型	7-3
7.2.1.1	上电复位	7-3
7.2.1.2	硬件复位	7-4

7.2.1.3	看门狗定时器复位 .....	7-4
7.2.1.4	掉电唤醒复位 .....	7-4
7.2.1.5	欠压复位 .....	7-5
7.2.2	模块复位行为 .....	7-6
7.2.3	启动方案 .....	7-7
7.2.4	寄存器描述 .....	7-8
7.3	时钟系统 .....	7-10
7.3.1	时钟产生单元 .....	7-10
7.3.1.1	功能描述 .....	7-11
7.3.2	时钟源控制 .....	7-14
7.3.3	时钟管理 .....	7-15
7.3.4	寄存器描述 .....	7-17
<b>8</b>	<b>省电模式 .....</b>	<b>8-1</b>
8.1	功能描述 .....	8-2
8.1.1	空闲模式 .....	8-2
8.1.2	低速模式 .....	8-2
8.1.3	掉电模式 .....	8-2
8.1.4	外设时钟管理 .....	8-5
8.2	寄存器描述 .....	8-7
<b>9</b>	<b>看门狗定时器 .....</b>	<b>9-1</b>
9.1	功能描述 .....	9-2
9.1.1	模块挂起控制 .....	9-4
9.2	寄存器映射 .....	9-5
9.3	寄存器描述 .....	9-5
<b>10</b>	<b>乘法 / 除法单元 .....</b>	<b>10-1</b>
10.1	功能描述 .....	10-2
10.1.1	除法操作 .....	10-3
10.1.2	归一化 .....	10-3
10.1.3	移位 .....	10-3
10.1.4	忙碌标志 .....	10-3
10.1.5	错误检测 .....	10-4
10.2	中断产生 .....	10-4
10.3	低功耗模式 .....	10-5
10.4	寄存器映射 .....	10-6
10.5	寄存器描述 .....	10-7
10.5.1	操作数和结果寄存器 .....	10-9
10.5.2	控制寄存器 .....	10-11
10.5.3	状态寄存器 .....	10-12
<b>11</b>	<b>CORDIC 协处理器 .....</b>	<b>11-1</b>
11.1	特性 .....	11-2
11.2	功能描述 .....	11-3

11.2.1	CORDIC 协处理器的操作	11-3
11.2.2	中断	11-3
11.2.3	结果数据归一化	11-4
11.2.4	CORDIC 协处理器操作模式	11-4
11.2.4.1	收敛域	11-7
11.2.4.2	溢出考虑	11-7
11.2.5	CORDIC 协处理器数据格式	11-8
11.2.6	CORDIC 协处理器精度	11-8
11.2.7	CORDIC 协处理器性能	11-10
11.3	CORDIC 协处理器内核	11-11
11.3.1	反正切和双曲反正切查找表	11-11
11.3.2	线性函数的仿真查找表	11-12
11.4	低功耗模式	11-13
11.5	寄存器映射	11-14
11.6	寄存器描述	11-15
11.6.1	控制寄存器	11-15
11.6.2	状态和数据控制寄存器	11-17
11.6.3	数据寄存器	11-18
<b>12</b>	<b>串行接口</b>	<b>12-1</b>
12.1	UART	12-2
12.1.1	UART 模式	12-2
12.1.1.1	模式 0, 8 位移位寄存器, 波特率固定	12-2
12.1.1.2	模式 1, 8 位 UART, 波特率可变	12-3
12.1.1.3	模式 2, 9 位 UART, 波特率固定	12-5
12.1.1.4	模式 3, 9 位 UART, 波特率可变	12-5
12.1.2	多处理器通信	12-7
12.1.3	UART 寄存器描述	12-7
12.1.4	波特率产生	12-10
12.1.4.1	固定时钟	12-10
12.1.4.2	专用波特率发生器	12-11
12.1.4.3	定时器 T1	12-22
12.1.5	端口控制	12-23
12.1.6	低功耗模式	12-24
12.1.7	寄存器映射	12-25
12.2	本地互连网络 (LIN)	12-26
12.2.1	LIN 协议	12-26
12.2.2	LIN 报文头发送	12-28
12.2.2.1	和主机自动同步	12-28
12.2.2.2	LIN 波特率检测	12-29
12.3	高速同步串行接口	12-31
12.3.1	基本操作	12-32
12.3.1.1	工作模式选择	12-32

12.3.1.2	全双工工作 .....	12-33
12.3.1.3	半双工工作 .....	12-35
12.3.1.4	连续传送 .....	12-36
12.3.1.5	端口控制 .....	12-37
12.3.1.6	波特率产生 .....	12-39
12.3.1.7	检错机制 .....	12-41
12.3.2	中断 .....	12-43
12.3.3	低功耗模式 .....	12-45
12.3.4	寄存器映射 .....	12-45
12.3.5	寄存器描述 .....	12-46
12.3.5.1	配置寄存器 .....	12-46
12.3.5.2	波特率定时器重载寄存器 .....	12-50
12.3.5.3	发送和接收缓存寄存器 .....	12-51
<b>13</b>	<b>定时器 .....</b>	<b>13-1</b>
13.1	定时器 T0 和定时器 T1 .....	13-2
13.1.1	定时器基本操作 .....	13-2
13.1.2	定时器工作模式 .....	13-2
13.1.2.1	模式 0 .....	13-4
13.1.2.2	模式 1 .....	13-5
13.1.2.3	模式 2 .....	13-6
13.1.2.4	模式 3 .....	13-7
13.1.3	端口控制 .....	13-8
13.1.4	寄存器映射 .....	13-9
13.1.5	寄存器描述 .....	13-10
13.2	定时器 T2 和定时器 T21 .....	13-14
13.2.1	定时器基本操作 .....	13-14
13.2.2	自动重载模式 .....	13-14
13.2.2.1	禁止递增 / 递减计数 .....	13-14
13.2.2.2	使能递增 / 递减计数 .....	13-15
13.2.3	捕获模式 .....	13-17
13.2.4	计数时钟 .....	13-18
13.2.5	外部中断功能 .....	13-19
13.2.6	端口控制 .....	13-19
13.2.7	低功耗模式 .....	13-20
13.2.8	模式挂起控制 .....	13-21
13.2.9	寄存器映射 .....	13-22
13.2.10	寄存器描述 .....	13-23
<b>14</b>	<b>T2CCU .....</b>	<b>14-1</b>
14.1	特性 .....	14-1
14.2	对定时器 T2 的要求 .....	14-3
14.3	T2CCU 的捕获 / 比较定时器 .....	14-3
14.3.1	同步定时器启动 .....	14-4

14.3.2	实现灵活计数频率的定时器级联	14-4
14.3.3	CCT 定时器实现死区时间生成	14-5
14.4	T2CCU 捕获 / 比较通道	14-8
14.5	捕获 / 比较操作	14-10
14.5.1	比较功能	14-10
14.5.1.1	比较模式 0	14-12
14.5.1.2	比较模式 1	14-13
14.5.1.3	并行比较模式	14-14
14.5.1.4	中断和比较功能组合使用	14-17
14.5.2	捕获功能	14-18
14.6	T2CCU 寄存器	14-19
14.6.1	寄存器映射	14-20
14.6.2	T2CCU 寄存器描述	14-21
<b>15</b>	<b>捕获 / 比较单元 6</b>	<b>15-1</b>
15.1	功能描述	15-3
15.1.1	定时器 T12	15-3
15.1.1.1	定时器配置	15-4
15.1.1.2	计数规则	15-4
15.1.1.3	切换规则	15-4
15.1.1.4	T12 的比较模式	15-5
15.1.1.5	0% 和 100% 占空比	15-7
15.1.1.6	死区时间产生	15-7
15.1.1.7	捕获模式	15-8
15.1.1.8	单次模式	15-9
15.1.1.9	类磁滞控制模式	15-9
15.1.2	定时器 T13	15-11
15.1.2.1	定时器配置	15-11
15.1.2.2	比较模式	15-12
15.1.2.3	单次模式	15-12
15.1.2.4	T13 与 T12 同步	15-12
15.1.3	调制控制	15-14
15.1.4	强制中断处理	15-16
15.1.5	多通道模式	15-18
15.1.6	霍尔传感器模式	15-20
15.1.6.1	霍尔序列采样	15-20
15.1.6.2	无刷直流控制	15-20
15.1.7	中断产生	15-24
15.1.8	低功率模式	15-25
15.1.9	模块挂起控制	15-26
15.1.10	端口连接	15-27
15.1.11	定时器同步启动	15-31
15.2	寄存器映射	15-32

<b>15.3</b>	<b>寄存器描述</b>	<b>15-34</b>
15.3.1	系统寄存器	15-36
15.3.2	定时器 T12 相关的寄存器	15-39
15.3.3	定时器 T13 相关寄存器	15-48
15.3.4	捕获 / 比较控制寄存器	15-51
15.3.5	全局调制控制寄存器	15-63
15.3.6	多通道调制控制寄存器	15-68
15.3.7	中断控制寄存器	15-73
<b>16</b>	<b>MultiCAN</b>	<b>16-1</b>
16.1	MultiCAN 内核功能描述	16-3
16.1.1	模块结构	16-3
16.1.2	时钟控制	16-6
16.1.3	CAN 节点控制	16-7
16.1.3.1	位时序单元	16-7
16.1.3.2	位流处理器	16-8
16.1.3.3	错误处理单元	16-8
16.1.3.4	CAN 帧计数器	16-9
16.1.3.5	CAN 节点中断	16-9
16.1.4	报文对象列表结构	16-11
16.1.4.1	基本知识	16-11
16.1.4.2	未被分配的报文对象列表	16-12
16.1.4.3	与 CAN 节点连接	16-12
16.1.4.4	列表命令面板	16-13
16.1.5	CAN 节点分析特性	16-16
16.1.5.1	分析模式	16-16
16.1.5.2	回环模式	16-16
16.1.5.3	位时序分析	16-17
16.1.6	报文验收滤波	16-18
16.1.6.1	接收验收滤波	16-18
16.1.6.2	发送验收滤波	16-18
16.1.7	报文后处理	16-20
16.1.7.1	报文中断	16-20
16.1.7.2	报文挂起	16-21
16.1.8	报文对象数据处理	16-23
16.1.8.1	帧接收	16-23
16.1.8.2	帧发送	16-26
16.1.9	报文对象功能	16-29
16.1.9.1	标准报文对象	16-29
16.1.9.2	单一数据传送模式	16-29
16.1.9.3	单次发送试验	16-29
16.1.9.4	报文对象 FIFO 结构	16-29
16.1.9.5	接收 FIFO	16-32

16.1.9.6	发送 FIFO	16-33
16.1.9.7	网关模式	16-34
16.1.9.8	外来远程请求	16-36
16.1.10	访问仲裁器	16-37
16.1.11	端口控制	16-38
16.1.12	低功耗模式	16-39
16.2	寄存器描述	16-40
16.2.1	全局模块寄存器	16-43
16.2.2	CAN 节点寄存器	16-53
16.2.3	报文对象寄存器	16-69
16.2.4	MultiCAN 访问仲裁寄存器	16-87
<b>17</b>	<b>模数转换器</b>	<b>17-1</b>
17.1	结构概述	17-2
17.2	时钟方案	17-3
17.2.1	转换时序	17-4
17.3	低功耗模式	17-7
17.4	功能描述	17-8
17.4.1	请求源仲裁器	17-8
17.4.2	转换启动模式	17-10
17.4.3	通道控制	17-10
17.4.4	顺序请求源	17-11
17.4.4.1	概述	17-11
17.4.4.2	请求源控制	17-13
17.4.5	并行请求源	17-14
17.4.5.1	概述	17-14
17.4.5.2	请求源控制	17-15
17.4.5.3	外部触发	17-15
17.4.5.4	软件控制	17-16
17.4.5.5	自动扫描	17-16
17.4.6	待读模式	17-17
17.4.7	转换结果的产生	17-17
17.4.7.1	概述	17-17
17.4.7.2	极限检查	17-19
17.4.7.3	数据压缩滤波	17-20
17.4.7.4	结果寄存器的读取	17-21
17.4.8	中断	17-22
17.4.8.1	事件中断	17-23
17.4.8.2	通道中断	17-24
17.4.9	外部触发输入	17-26
17.5	ADC 模块初始化步骤	17-28
17.6	寄存器映射	17-30
17.7	寄存器描述	17-32



17.7.1	基本功能寄存器 .....	17-32
17.7.2	优先级和仲裁寄存器 .....	17-34
17.7.3	外部触发控制寄存器 .....	17-36
17.7.4	通道控制寄存器 .....	17-37
17.7.5	输入综合控制寄存器 .....	17-38
17.7.6	顺序请求源寄存器 .....	17-39
17.7.7	并列请求源寄存器 .....	17-46
17.7.8	结果寄存器 .....	17-50
17.7.9	中断寄存器 .....	17-56
<b>18</b>	<b>片上调试支持 .....</b>	<b>18-1</b>
18.1	特性 .....	18-1
18.2	功能描述 .....	18-1
18.3	调试 .....	18-2
18.3.1	调试事件 .....	18-3
18.3.1.1	硬件断点 .....	18-3
18.3.1.2	软件断点 .....	18-4
18.3.1.3	外部断点 .....	18-4
18.3.1.4	NMI 模式的优先级高于调试模式 .....	18-5
18.3.2	调试动作 .....	18-5
18.3.2.1	调用监控程序 .....	18-5
18.3.2.2	激活 MBC 引脚 .....	18-5
18.4	调试挂起控制 .....	18-6
18.5	寄存器描述 .....	18-7
18.5.1	监控器工作寄存器 2 .....	18-8
18.5.2	输入选择寄存器 .....	18-8
18.6	JTAG ID .....	18-10
<b>19</b>	<b>引导程序加载器 .....</b>	<b>19-1</b>
19.1	UART 和 LIN BSL 模式 .....	19-2
19.1.1	通信协议 .....	19-3
19.1.1.1	UART 传送块结构 .....	19-3
19.1.1.2	LIN 传送块结构 .....	19-4
19.1.1.3	返回主机的回应码 .....	19-6
19.1.2	通过 UART 实现引导程序加载器 .....	19-8
19.1.2.1	通信结构 .....	19-9
19.1.2.2	模式选择 .....	19-9
19.1.2.3	激活模式 0 和模式 2 .....	19-10
19.1.2.4	激活模式 1、3 和 15 .....	19-11
19.1.2.5	激活模式 6 .....	19-12
19.1.2.6	激活模式 10 .....	19-12
19.1.2.7	激活模式 16 .....	19-13
19.1.3	通过 LIN 实现引导程序加载器 .....	19-14
19.1.3.1	通信结构 .....	19-15

19.1.3.2	模式选择 .....	19-16
19.1.3.3	激活模式 0, 2 和 8 .....	19-17
19.1.3.4	激活模式 1, 3 和 9 .....	19-19
19.1.3.5	激活模式 6 .....	19-19
19.1.3.6	激活模式 10 .....	19-20
19.1.3.7	激活模式 16 .....	19-20
19.1.3.8	发向主机的 LIN 回应协议 .....	19-20
19.1.3.9	快速 LIN BSL .....	19-21
19.1.3.10	用户自定义的 LIN BSL 参数 .....	19-21
19.2	MultiCAN BSL 模式 .....	19-23
19.2.1	通信协议 .....	19-23
19.2.2	CAN 报文对象的定义 .....	19-24
19.2.3	用户定义的 MultiCAN BSL .....	19-25
19.3	其它 BSL 模式 .....	19-26

<b>XC878</b>	
版本信息：	2009-04 V 1.1
先前版本：	V 1.0
页	内容（对上一版本的主要修正）
页 16-38	更新表 16-4：端口线 P1.4/RXDC1_3 的 NPCR1.RXSEL 值从 001 <sub>B</sub> 改为 011 <sub>B</sub> 。
页 19-25	更新表 19-7：改变 MultiCAN BSL 参数的地址；01 <sub>H</sub> 对应的值从 6 MHz 改为 5 MHz。
页 4-11	更新表 4-4：所需堆栈大小从 4 改为 5。
页 1-3	更新表 1-2 中 3.3 V 汽车专用衍生产品的工作温度。
页 17-3	当 $f_{ADC}$ 改为 24 MHz 时，位 CTC 可选用 00 <sub>B</sub> ，见章节 17.2。
页 7-11	更新导致时钟失锁的情况的描述。
页 8-5	修改退出掉电模式时第 5 步的描述并添加一条注解。
页 12-15	BCON 寄存器的复位值从 00 <sub>H</sub> 改为 20 <sub>H</sub> 。
页 12-36	更新图 12-14：去除数据和时钟线的互连。
页 13-17	更新图 13-7，去除 RC2 到 TF2 的连接。
页 3-6	增加章节 3.2.2.2 描述新功能“外部数据存储器接口”。
页 3-4	更新章节 3.2.2 和章节 3.2.2.1 有关访问外部接口的内容。
页 1-7	更新表 1-3：增加外部接口的引脚定义。
页 6-20, 页 6-27, 页 6-35, 页 6-42	更新表 6-6, 表 6-8, 表 6-10 和表 6-12 增加外部接口的输入 / 输出功能。
页 7-12 页 7-13	更新章节 7.3.1.1 中选择片外振荡器作为 PLL 输入时钟或系统时钟时所执行的步骤。
页 12-26	添加注解说明 UART1 不支持 LIN 分隔域和同步字节硬件检测。
页 12-29	在波特率检测初始化序列中添加清除 LIN 状态标志和复位分隔域检测等步骤。
页 12-41	更新章节 12.3.1.7 中有关波特率出错的注解。
页 1-2	为表 1-1 补充 LIN BSL 可用性注解。
页 9-7	更新复位值的脚注。

**期待您的指正**

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

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



## 1 简介

XC878 是高性能 8 位微控制器 XC800 家族的新成员，其设计基于和工业标准 8051 处理器兼容的 XC800 内核。

XC878 片内集成 CAN 控制器并支持 LIN 协议，具备高级互联功能。片内 CAN 模块可执行网络协议所需的大多数功能（CAN 帧屏蔽、滤波和缓存功能），从而减轻了 CPU 负荷。

XC878 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。XC878 的存储器保护策略为用户知识产权（IP）提供读保护，为 Flash 提供编程和擦除保护以防止数据遭无意破坏。

Flash 结构支持在应用编程（IAP），在程序执行期间允许用户程序修改 Flash 的内容。通过位于 Boot ROM 中的引导程序加载（BSL）可实现在系统编程（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），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、线性或双曲函数以实现向量旋转和转换；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。

XC878 片内还集成了振荡器和电压调节器，可由 3.3 或 5.0V 的单电源供电。XC878 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。

XC878 的功能单元如图 1-1 所示。

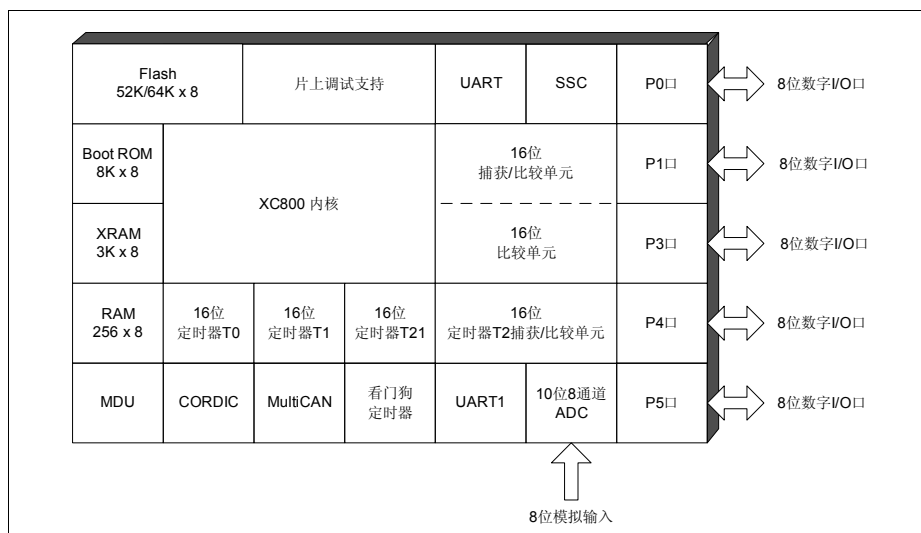


图 1-1 XC878 功能单元

XC878 系列包含多种具有不同配置、不同程序存储器容量、不同供电电压以及不同温度和应用范围（汽车级或工业级应用）的器件，为不同的应用需求提供了经济合算的解决方案。

XC878 的各种器件配置总结见表 1-1。封装形式为 LQFP-64。

表 1-1 器件配置

器件名称	CAN 模块	LIN BSL 支持	MDU 模块
XC878	无	无	无
XC878C	有	无	无
XC878L	无	有	无
XC878M	无	无	有
XC878CM	有	无	有

注：对于支持 LIN BSL 的衍生产品，只使用 LIN BSL（无论 UART 模块是否可用）。

根据器件类型、程序存储器容量、电源电压、工作温度和应用范围（汽车级或工业级应用）的不同，上表中的每种器件又被细分为多种产品类型，总结见表 1-2。

**表 1-2          产品类型**

产品类型	器件类型	程序存储器 (KB)	电源电压 (V)	温度范围 (°C)	应用范围
SAF-XC878-13FFI 5V	Flash	52	5.0	-40 - 85	工业级
SAF-XC878M-13FFI 5V	Flash	52	5.0	-40 - 85	工业级
SAF-XC878CM-13FFI 5V	Flash	52	5.0	-40 - 85	工业级
SAF-XC878-16FFI 5V	Flash	64	5.0	-40 - 85	工业级
SAF-XC878M-16FFI 5V	Flash	64	5.0	-40 - 85	工业级
SAF-XC878CM-16FFI 5V	Flash	64	5.0	-40 - 85	工业级
SAF-XC878-13FFI 3V3	Flash	52	3.3	-40 - 85	工业级
SAF-XC878M-13FFI 3V3	Flash	52	3.3	-40 - 85	工业级
SAF-XC878CM-13FFI 3V3	Flash	52	3.3	-40 - 85	工业级
SAF-XC878-16FFI 3V3	Flash	64	3.3	-40 - 85	工业级
SAF-XC878M-16FFI 3V3	Flash	64	3.3	-40 - 85	工业级
SAF-XC878CM-16FFI 3V3	Flash	64	3.3	-40 - 85	工业级
SAX-XC878-13FFA 5V	Flash	52	5.0	-40 - 105	工业级
SAX-XC878L-13FFA 5V	Flash	52	5.0	-40 - 105	汽车级
SAX-XC878C-13FFA 5V	Flash	52	5.0	-40 - 105	汽车级
SAX-XC878CM-13FFA 5V	Flash	52	5.0	-40 - 105	汽车级
SAX-XC878-16FFA 5V	Flash	64	5.0	-40 - 105	汽车级
SAX-XC878L-16FFA 5V	Flash	64	5.0	-40 - 105	汽车级
SAX-XC878C-16FFA 5V	Flash	64	5.0	-40 - 105	汽车级
SAX-XC878CM-16FFA 5V	Flash	64	5.0	-40 - 105	汽车级
SAF-XC878-16FFA 3V3	Flash	64	3.3	-40 - 85	汽车级
SAF-XC878L-16FFA 3V3	Flash	64	3.3	-40 - 85	汽车级

除非特别声明，本手册中的 XC878 泛指 XC878 系列的所有器件。

## 1.1 特性总结

XC878 的主要特性总结如下：

- 高性能 XC800 内核
  - 标准 8051 处理器兼容
  - 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
  - 256B RAM
  - 3 KB XRAM
  - 52/64 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 3.3V 或 5.0V 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
- 内核电压压降检测
- 片内振荡器（OSC）和锁相环（PLL）产生时钟
  - 时钟丢失检测
- 省电模式
  - 低速模式
  - 空闲模式
  - 掉电模式，可通过 RXD 或 EXINT0 唤醒系统
  - 每个外设均由时钟门控制
- 可编程 16 位看门狗定时器（WDT）
- 五个端口
  - 多达 40 个数字 I/O 引脚
  - 8 个专用模拟输入引脚，用作模数转换器的输入
- 8 通道、10 位模数转换单元（ADC）
- 四个 16 位定时器
  - 定时器 T0 和定时器 T1（T0 和 T1）
  - 定时器 T2 和定时器 T21（T2 和 T21）
- 专门用于算术运算的乘法 / 除法单元（MDU）
- CORDIC 协处理器，支持三角函数、双曲和线性函数计算
- 具有 2 个节点、32 个报文对象的 MultiCAN 模块
- 两个捕获 / 比较单元
  - 产生脉宽调制（PWM）信号的捕获 / 比较单元（CCU6）
  - 产生不同数字信号的定时器 T2 捕获 / 比较单元（T2CCU）
- 两个全双工串行接口（UART 和 UART1）
- 同步串行通道（SSC）
- 片上调试支持
  - 1 KB 监控 RAM（8KB Boot ROM 的一部分）
  - 64 B 监控 RAM
- PG-LQFP-64 引脚封装



- 温度范围  $T_A$ :
  - SAF (-40 至 85 °C)
  - SAX (-40 至 105 °C)

XC878 框图如图 1-2 所示。

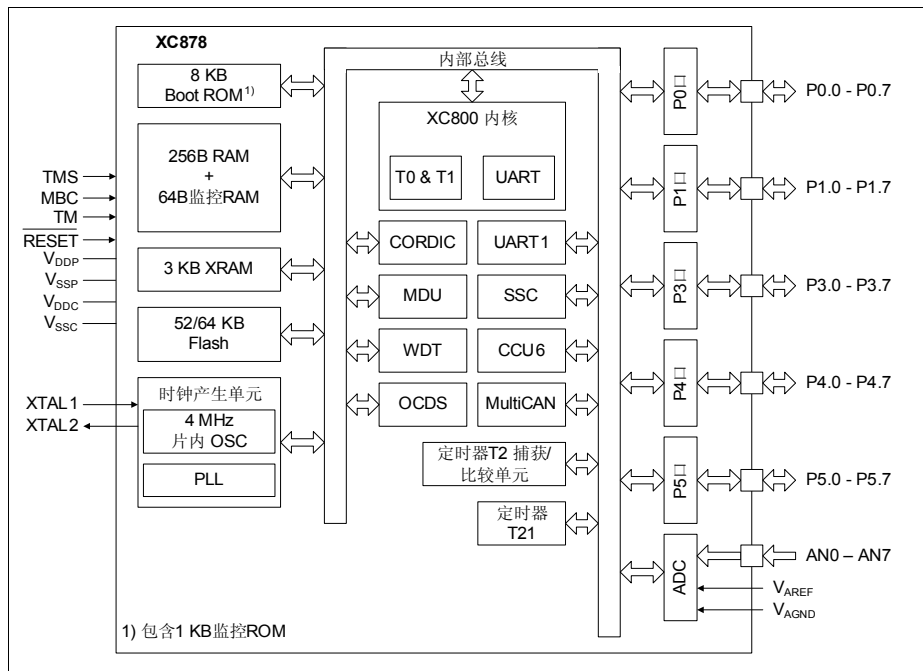


图 1-2 XC878 框图

## 1.2 引脚配置

基于 PG-LQFP-64 封装的 XC878 的引脚配置如图 1-3 所示。

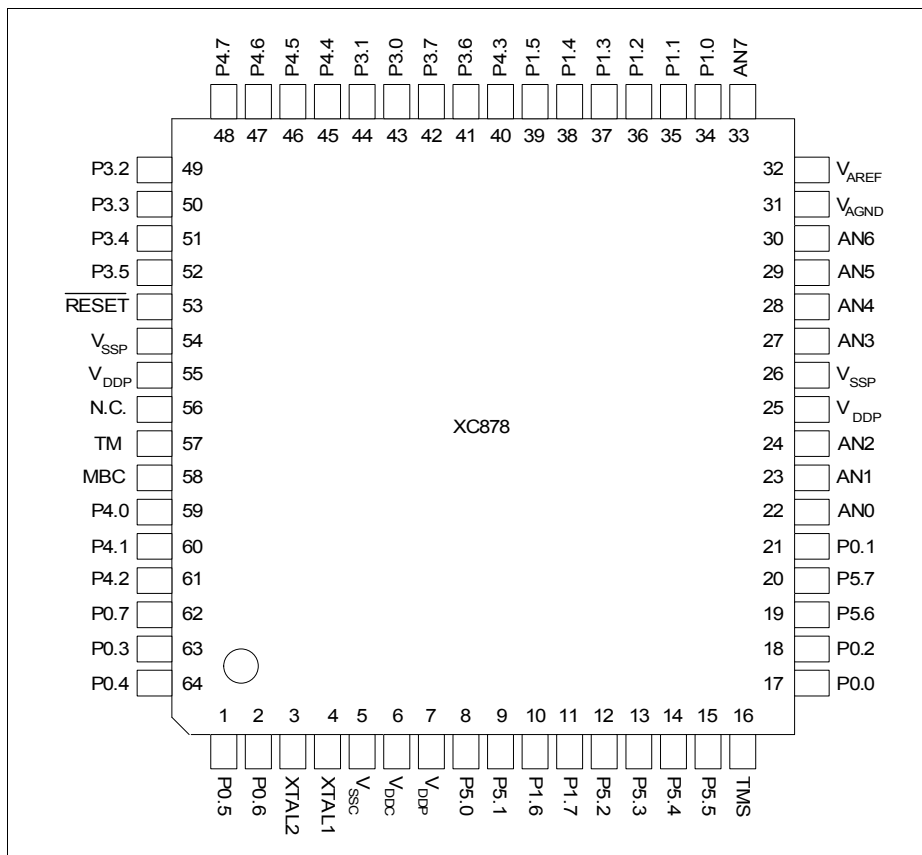


图 1-3 XC878 引脚配置，PG-LQFP-64 封装（顶视图）

## 1.3 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）

**表 1-3** 列出 XC878 的引脚功能及缺省状态。

**表 1-3 引脚定义和功能**

符号	引脚编号 (LQFP-64)	类型	复位 状态	引脚功能
<b>P0</b>		输入 / 输出		<b>P0 口</b> P0 口是 8 位通用双向 I/O 口。它还可用作 JTAG、CCU6、UART、UART1、T2CCU、T21、MultiCAN、SSC 和外部接口的功能引脚。
P0.0	17		高阻	TCK_0 JTAG 时钟输入 T12HR_1 CCU6 定时器 T12 硬件运行输入 CC61_1 捕获 / 比较通道 1 输入 / 输出 CLKOUT_0 时钟输出 RXDO_1 UART 发送数据输出
P0.1	21		高阻	TDI_0 JTAG 串行数据输入 T13HR_1 CCU6 定时器 T13 硬件运行输入 RXD_1 UART 接收数据输入 RXDC1_0 MultiCAN 节点 1 接收输入 COUT61_1 捕获 / 比较通道 1 输出 EXF2_1 定时器 T2 外部标志输出
P0.2	18		上拉	CTRAP_2 CCU6 强制中断输入 TDO_0 JTAG 串行数据输出 TXD_1 UART 发送数据输出 / 时钟输出 TXDC1_0 MultiCAN 节点 1 发送输出
P0.3	63		高阻	SCK_1 SSC 时钟输入 / 输出 COUT63_1 捕获 / 比较通道 3 输出 RXDO1_0 UART1 发送数据输出 A17 地址线 17 输出
P0.4	64		高阻	MTSR_1 SSC 主机发送输出 / 从机接收输入 CC62_1 捕获 / 比较通道 2 输入 / 输出 TXD1_0 UART1 发送数据输出 / 时钟输出 A18 地址线 18 输出

表 1-3 引脚定义和功能

符号	引脚编号 (LQFP-64)	类型	复位 状态	引脚功能
P0.5	1		高阻	MRST_1 SSC 主机接收输入 / 从机发送输出 EXINT0_0 外部中断输入 0 T2EX1_1 定时器 T21 外部触发输入 RXD1_0 UART1 接收数据输入 COUT62_1 捕获 / 比较通道 2 输出 A19 地址线 19 输出
P0.6	2		上拉	T2CC4_1 比较输出通道 4 WR 外部数据写控制输出
P0.7	62		上拉	CLKOUT_1 时钟输出 T2CC5_1 比较输出通道 5 RD 外部数据读控制输出

表 1-3 引脚定义和功能

符号	引脚编号 (LQFP-64)	类型	复位 状态	引脚功能
<b>P1</b>		输入 / 输出		<b>P1 口</b> P1 口是 8 位通用双向 I/O 口。它还可用作 JTAG、CCU6、UART、T0、T1、T2CCU、T21、MultiCAN、SSC 和外部接口的功能引脚。
P1.0	34		PU	RXD_0      UART 接收数据输入 T2EX_0      定时器 T2 外部触发输入 RXDC0_0    MultiCAN 节点 0 接收输入 A8            地址线 8 输出
P1.1	35		PU	EXINT3_0    外部中断输入 3 T0_1          定时器 T0 输入 TXD_0        UART 发送数据输出 / 时钟输出 TXDC0_0    MultiCAN 节点 0 发送输出 A9            地址线 9 输出
P1.2	36		PU	SCK_0        SSC 时钟输入 / 输出 A10           地址线 10 输出
P1.3	37		PU	MTSR_0      SSC 主机发送输出 / 从机接收输入  SCK_2        SSC 时钟输入 / 输出 TXDC1_3    MultiCAN 节点 1 发送输出 A11           地址线 11 输出
P1.4	38		PU	MRST_0      SSC 主机接收输入 / 从机发送输出  EXINT0_1    外部中断输入 0 RXDC1_3    MultiCAN 节点 1 接收输入 MTSR_2      SSC 主机发送输出 / 从机接收输入  A12           地址线 12 输出
P1.5	39		PU	CCPOS0_1    CCU6 霍尔输入 0 EXINT5_0    外部中断输入 5 T1_1          定时器 T1 输入 MRST_2      SSC 主机接收输入 / 从机发送输出  EXF2_0       定时器 T2 外部标志输出 RXDO_0      UART 发送数据输出

表 1-3            引脚定义和功能

符号	引脚编号 (LQFP-64)	类型	复位 状态	引脚功能
P1.6	10		PU	CCPOS1_1    CCU6 霍尔输入 1 T12HR_0    CCU6 定时器 T12 硬件运行输入 EXINT6_0    外部中断输入 6 RXDC0_2    MultiCAN 节点 0 接收输入 T21_1        定时器 T21 输入
P1.7	11		PU	CCPOS2_1    CCU6 霍尔输入 2 T13HR_0    CCU6 定时器 T13 硬件运行输入 T2_1         定时器 T2 输入 TXDC0_2    MultiCAN 节点 0 发送输出 P1.5 和 P1.6 可用作 SSC 的软件片选输出。

**表 1-3 引脚定义和功能**

符号	引脚编号 (LQFP-64)	类型	复位 状态	引脚功能
<b>P3</b>		I/O		<b>P3 口</b> P3 口是 8 位通用双向 I/O 口。它还可用作 CCU6、UART1、T2CCU、T21、MultiCAN 和外部接口的功能引脚。
P3.0	43		Hi-Z	CCPOS1_2 CCU6 霍尔输入 1 CC60_0 捕获 / 比较通道 0 输入 / 输出 RXDO1_1 UART1 发送数据输出 t T2CC0_1/ T2CCU 捕获 / 比较通道 0/ EXINT3_2 外部中断输入 3
P3.1	44		Hi-Z	CCPOS0_2 CCU6 霍尔输入 0 CC61_2 捕获 / 比较通道 1 输入 / 输出 COUT60_0 捕获 / 比较通道 0 输出 TXD1_1 UART1 发送数据输出 / 时钟输出
P3.2	49		Hi-Z	CCPOS2_2 CCU6 霍尔输入 2 RXDC1_1 MultiCAN 节点 1 接收输入 RXD1_1 UART1 接收数据输入 CC61_0 捕获 / 比较通道 1 输入 / 输出 T2CC1_1/ T2CCU 捕获 / 比较通道 1/ EXINT4_2 外部中断输入 4
P3.3	50		Hi-Z	COUT61_0 捕获 / 比较通道 1 输出 TXDC1_1 MultiCAN 节点 1 发送输出 T2CC2_1/ T2CCU 捕获 / 比较通道 2/ EXINT5_2 外部中断输入 5 A13 地址线 13 输出
P3.4	51		Hi-Z	CC62_0 捕获 / 比较通道 2 输入 / 输出 RXDC0_1 MultiCAN 节点 0 接收输入 T2EX1_0 定时器 T21 外部触发输入 T2CC3_1/ T2CCU 捕获 / 比较通道 3/ EXINT6_3 外部中断输入 6 A14 地址线 14 输出
P3.5	52		Hi-Z	COUT62_0 捕获 / 比较通道 2 输出 EXF21_0 定时器 T21 外部标志输出 TXDC0_1 MultiCAN 节点 0 发送输出 A15 地址线 15 输出
P3.6	41		PU	CTRAP_0 CCU6 强制中断输入

表 1-3 引脚定义和功能

符号	引脚编号 (LQFP-64)	类型	复位 状态	引脚功能
P3.7	42		Hi-Z	EXINT4_0 外部中断输入 4 COUT63_0 捕获 / 比较通道 3 输出 A16 地址线 16 输出



**表 1-3 引脚定义和功能**

符号	引脚编号 (LQFP-64)	类型	复位 状态	引脚功能
<b>P4</b>		I/O		<b>P4 口</b> P4 口是 8 位通用双向 I/O 口。它还可用作 CCU6、T0、T1、T2CCU、T21、MultiCAN 和外部接口的功能引脚。
P4.0	59		Hi-Z	RXDC0_3 CC60_1 T2CC0_0/ EXINT3_1 D0 MultiCAN 节点 0 接收输入 捕获 / 比较通道 0 输出 T2CCU 捕获 / 比较通道 0/ 外部中断输入 3 数据线 0 输入 / 输出
P4.1	60		Hi-Z	TXDC0_3 COUT60_1 T2CC1_0/ EXINT4_1 D1 MultiCAN 节点 0 发送输出 捕获 / 比较通道 0 输出 T2CCU 捕获 / 比较通道 1/ 外部中断输入 4 数据线 1 输入 / 输出
P4.2	61		PU	EXINT6_1 T21_0 D2 外部中断输入 6 定时器 T21 输入 数据线 2 输入 / 输出
P4.3	40		Hi-Z	T2EX_1 EXF21_1 COUT63_2 D3 定时器 T2 外部触发输入 定时器 T21 外部标志输出 捕获 / 比较通道 3 输出 数据线 3 输入 / 输出
P4.4	45		Hi-Z	CCPOS0_3 T0_0 CC61_4 T2CC2_0/ EXINT5_1 D4 CCU6 霍尔输入 0 定时器 T0 输入 捕获 / 比较通道 1 输出 T2CCU 捕获 / 比较通道 2/ 外部中断输入 5 数据线 4 输入 / 输出
P4.5	46		Hi-Z	CCPOS1_3 T1_0 COUT61_2 T2CC3_0/ EXINT6_2 D5 CCU6 霍尔输入 1 定时器 T1 输入 捕获 / 比较通道 1 输出 T2CCU 捕获 / 比较通道 3/ 外部中断输入 6 数据线 5 输入 / 输出

表 1-3            引脚定义和功能

符号	引脚编号 (LQFP-64)	类型	复位 状态	引脚功能
P4.6	47		Hi-Z	CCPOS2_3    CCU6 霍尔输入 2 T2_0            定时器 T2 输入 CC62_2        捕获 / 比较通道 2 输出 T2CC4_0       比较输出通道 4 D6              数据线 6 输入 / 输出
P4.7	48		Hi-Z	CTRAP_3      CCU6 强制中断输入 COUT62_2     捕获 / 比较通道 2 输出 T2CC5_0       比较输出通道 5 D7              数据线 7 输入 / 输出

**表 1-3 引脚定义和功能**

符号	引脚编号 (LQFP-64)	类型	复位 状态	引脚功能
<b>P5</b>		I/O		<b>P5 口</b> P5 口是 8 位通用双向 I/O 口。它还可用作 UART、UART1、T2CCU、JTAG 和外部接口的功能引脚。
P5.0	8		PU	EXINT1_1 外部中断输入 1 A0 地址线 0 输出
P5.1	9		PU	EXINT2_1 外部中断输入 2 A1 地址线 1 输出
P5.2	12		PU	RXD_2 UART 接收数据输入 T2CC2_2/ T2CCU 捕获 / 比较通道 2/ EXINT5_3 外部中断输入 5 A2 地址线 2 输出
P5.3	13		PU	CCPOS0_0 CCU6 霍尔输入 0 EXINT1_0 外部中断输入 1 T12HR_2 CCU6 定时器 T12 硬件运行输入 CC61_3 捕获 / 比较通道 1 输入 TXD_2 UART 发送数据输出 / 时钟输出 T2CC5_2 比较输出通道 5 A3 地址线 3 输出
P5.4	14		PU	CCPOS1_0 CCU6 霍尔输入 1 EXINT2_0 外部中断输入 2 T13HR_2 CCU6 定时器 T13 硬件运行输入 CC62_3 捕获 / 比较通道 2 输入 RXDO_2 UART 发送数据输出 T2CC4_2 比较输出通道 4 A4 地址线 4 输出
P5.5	15		PU	CCPOS2_0 CCU6 霍尔输入 2 CTRAP_1 CCU6 强制中断输入 CC60_3 捕获 / 比较通道 0 输入 TDO_1 JTAG 串行数据输出 TXD1_2 UART1 发送数据输出 / 时钟输出 T2CC0_2/ T2CCU 捕获 / 比较通道 0/ EXINT3_3 外部中断输入 3 A5 地址线 5 输出

表 1-3            引脚定义和功能

符号	引脚编号 (LQFP-64)	类型	复位 状态	引脚功能	
P5.6	19		PU	TCK_1 RXD01_2 T2CC1_2/ EXINT4_3 A6	JTAG 时钟输入 UART1 发送数据输出 T2CCU 捕获 / 比较通道 1/ 外部中断输入 4 地址线 6 输出
P5.7	20		PU	TDI_1 RXD1_2 T2CC3_2/ EXINT6_4 A7	JTAG 串行数据输入 UART1 接收数据输入 T2CCU 捕获 / 比较通道 3/ 外部中断输入 6 地址线 7 输出

表 1-3 引脚定义和功能

符号	引脚编号 (LQFP-64)	类型	复位 状态	引脚功能
$V_{DDP}$	7, 25, 55	—	—	<b>I/O 口电源（3.3 或 5.0V）</b> EVR 和模拟模块也使用该电源，所有引脚必须连接至电源。
$V_{SSP}$	26, 54	—	—	<b>I/O 口地</b> 所有引脚必须连接至地。
$V_{DDC}$	6	—	—	<b>内核电源监控（2.5V）</b>
$V_{SSC}$	5	—	—	<b>内核电源地</b>
$V_{AREF}$	32	—	—	<b>ADC 参考电压</b>
$V_{AGND}$	31	—	—	<b>ADC 参考地</b>
<b>AN0</b>	22	I	Hi-Z	模拟输入 0
<b>AN1</b>	23	I	Hi-Z	模拟输入 1
<b>AN2</b>	24	I	Hi-Z	模拟输入 2
<b>AN3</b>	27	I	Hi-Z	模拟输入 3
<b>AN4</b>	28	I	Hi-Z	模拟输入 4
<b>AN5</b>	29	I	Hi-Z	模拟输入 5
<b>AN6</b>	30	I	Hi-Z	模拟输入 6
<b>AN7</b>	33	I	Hi-Z	模拟输入 7
<b>XTAL1</b>	4	I	Hi-Z	片外 OSC 输入 （需要反馈电阻，通常不连接）
<b>XTAL2</b>	3	O	Hi-Z	片外 OSC 输出 （需要反馈电阻，通常不连接）
<b>TMS</b>	16	I	PD	JTAG 测试模式选择
<b>RESET</b>	53	I	PU	复位输入
<b>MBC</b>	58	I	PU	监控器 & 引导程序加载器控制
<b>TM</b>	57	—	—	测试模式（需要外部下拉器件）
<b>NC</b>	56	—	—	不连接

## 1.4 芯片 ID

每种 XC878 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、衍生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 4.7.3](#)；
- 引导程序加载（BSL）模式 10，见 [章节 19.1.2.6](#) 或 [章节 19.1.3.6](#)。

## 1.5 命名规则

本手册使用下面的规则来命名 XC878 的组成单元：

- XC878 的功能单元用大写表示。例如：“SSC 可和移位寄存器通信”。
- 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。
- 阴影位的功能在本手册的其它章节中予以说明。
- 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

## 1.6 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 1-4 定义。

表 1-4 位功能描述

位功能	描述
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。
未定义位	位域中的一些位元组合起来标记为“保留”，表示尚未定义 XC878 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”。

## 1.7 缩写

表 1-5 列出本手册中使用的缩写。

表 1-5 缩写

缩写	说明
ADC	模数转换器
ALU	算术 / 逻辑单元
BSL	引导程序加载器
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
ECC	纠错码
EVR	嵌入式电压调节器
FDR	分数分频器
GPIO	通用输入 / 输出



表 1-5 缩写

缩写	说明
IAP	在应用编程
I/O	输入 / 输出
ISP	在系统编程
JTAG	联合测试行动组
LIN	本地互连网络
MDU	乘 / 除单元
NMI	非可屏蔽中断
OCDS	片上调试支持
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
SFR	特殊功能寄存器
SPI	串行外设接口
SSC	同步串行通道
T2CCU	定时器 T2 捕获 / 比较单元
UART	通用异步收发器
WDT	看门狗定时器

## 2 处理器结构

XC878 具有与标准 8051 兼容的高性能 8 位中央处理单元（CPU）。标准 8051 处理器的机器周期由 12 个时钟周期组成，而 XC878 CPU 的机器周期由 2 个时钟组成，从而可快速、无等待的访问 ROM、RAM 和 Flash 存储器。XC878 的指令集由 45% 的单字节、41% 的双字节和 14% 的三字节指令组成。

XC878 CPU 所提供的调试特性包括：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（SFR）的读写操作。

### 特性

- 2 个时钟的机器周期结构（快速无等待存储器访问）
- 程序存储器下载选择
- 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
- 专用调试模式和调试信号
- 2 个 16 位定时器（定时器 T0 和定时器 T1）
- 全双工串行接口（UART）

### 2.1 功能概述

图 2-1 所示为 CPU 的功能框图。CPU 由指令译码器、运算单元和程序控制单元组成。每条指令由指令译码器译码，所产生的内部信号用来控制 CPU 内部各个模块。这些内部信号控制数据的传送和算术逻辑运算单元（ALU）的操作。

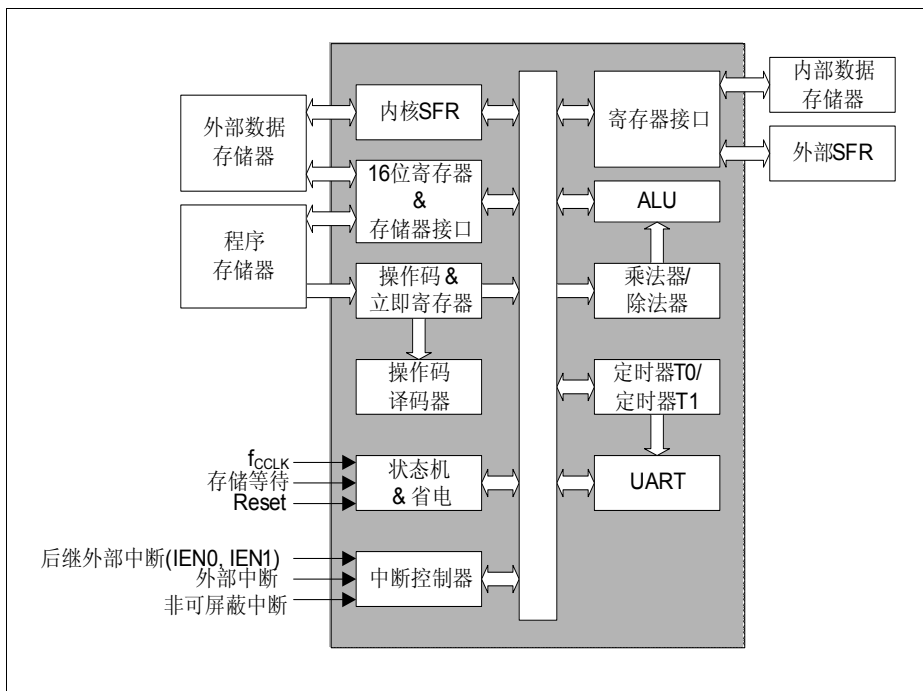


图 2-1 CPU 功能框图

处理器的运算单元具有强大的数据处理能力，包括 ALU、ACC 寄存器、B 寄存器和程序状态字寄存器 PSW。

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 and 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 还包括一个布尔处理机，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。

## 2.2 CPU 寄存器描述

CPU 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>。

### 2.2.1 堆栈指针（SP）

堆栈指针寄存器中存放堆栈指针 SP。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器中；在执行 RET 和 RETI 指令时，从存储器中恢复 PC 的值。也可以通过 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。

### 2.2.2 数据指针（DPTR）

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，用于外部数据存储器访问（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。

### 2.2.3 累加器（ACC）

大多数 ALU 操作中，该寄存器存放其中一个操作数。

### 2.2.4 B 寄存器

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。

## 2.2.5 程序状态字

程序状态字（PSW）寄存器中存放着多个状态位，指示 CPU 的当前状态。

### PSW

#### 程序状态字寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>CY</b>	<b>AC</b>	<b>F0</b>	<b>RS1</b>	<b>RS0</b>	<b>OV</b>	<b>F1</b>	<b>P</b>
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>RS1, RS0</b>	4:3	rw	<b>寄存器组（Register Bank）选择位</b> 由这两位进行 Bank 选择。 00 选择 Bank 0，数据地址 00 <sub>H</sub> -07 <sub>H</sub> 01 选择 Bank 1，数据地址 08 <sub>H</sub> -0F <sub>H</sub> 10 选择 Bank 2，数据地址 10 <sub>H</sub> -17 <sub>H</sub> 11 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。

### 2.2.6 扩展操作寄存器（EO）

指令集中还包括一条指令 **MOVC@ (DPTR++), A**，可对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>**，和软件断点指令 **TRAP** 的操作码相同（见表 2-1）。位 **EO.TRAP\_EN** 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>** 执行软件断点指令 **TRAP**，该指令将 CPU 切换至调试模式执行断点处理。

#### EO

#### 扩展操作寄存器

复位值：00<sub>H</sub>

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

符号	位	类型	描述
DPSEL0	0	rw	数据指针选择 0 选择 DPTR0 1 选择 DPTR1
TRAP_EN	4	rw	TRAP 使能位 0 选择 MOVC @ (DPTR++),A 1 选择软件 TRAP 指令
0	[3:1], [7:5]	r	保留 读操作返回 0，应写入 0。

### 2.2.7 功率控制寄存器（PCON）

CPU 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，CPU 时钟停止，但定时器、串口和中断控制器仍以半速时钟工作。掉电模式下，整个 CPU 的时钟全部停止。

**PCON**  
功率控制寄存器 复位值：00<sub>H</sub>

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 不进入空闲模式 1 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>0</b>	1, [6:4]	r	保留 读操作返回 0，应写入 0。

## 2.3 存储器扩展

8051 系统中可寻址程序存储器或外部数据存储器的标准大小（或称为一个 Bank）为 64KB。XC800 内核支持高达 1MB 的存储器扩展，通过使用存储管理单元（MMU）和存储器扩展堆栈来实现。MMU 增加了一组存储器扩展寄存器（MEX1、MEX2 和 MEX3），通过不同的寻址模式控制对扩展存储空间的访问。存储器扩展堆栈用于对 MEX1 的值进行“压栈”和“出栈”操作。

始终从 4 位当前 Bank（CB）寄存器位域指向的 64KB 存储段中读取相应的程序代码。执行长跳转（LJMP）和调用指令时，由 4 位下一个 Bank（NB）位域更新 CB。CB 和 NB 共同构成 MEX1 寄存器。编程人员只需在跳转或调用指令之前将新 Bank 编号写入 NB 即可。

始终从中断 Bank（IB）寄存器位域指向的 64KB 存储段中执行中断服务程序。此外，还可从当前 Bank 之外的其它 Bank 处执行存储器常数读取（位于代码段）和外部数据访问。这些 Bank 由位于 MEX2 和 MEX3 寄存器中的存储器常数 Bank 指针（MCB）和 XRAM Bank 指针（MXB）选择。

### 2.3.1 存储器扩展堆栈

存储器扩展模式下的中断和调用将用到存储器扩展堆栈，该堆栈和标准堆栈同步更新。

通过 SFR 存储器扩展堆栈指针 MEXSP 寻址存储器扩展堆栈。该读 / 写寄存器提供的堆栈深度最多为 128 字节（位 7 始终为 0）。每次执行调用指令之前 SFR 加 1；执行返回指令之后 SFR 减 1。MEXSP 的缺省复位值为 7F<sub>H</sub>，从而首次加 1 后选择栈底。系统不指示堆栈溢出。

### 2.3.2 存储器扩展的注意事项

以下指令可修改指向的 64KB 存储段：MOVC、MOVX、LJMP、LCALL、ACALL、RET 和 RETI。

不过，相对跳转（如 SJMP）、间接跳转（JMP@A+DPTR）和 2KB 存储段内的绝对跳转（AJMP）不会改变当前 Bank。换言之，这些指令不会改变当前有效的 64KB 存储段（Bank）。

#### 转移常数指令（MOVC）

MOVC 指令可访问当前 Bank（CB19-CB16）或“存储器常数”Bank（MEX3 和 MEX2 中的 MCB19-MCB16）中的数据字节。由 MEX2 中的位 MCM（MEX2.7）进行 Bank 选择。

#### 转移外部数据指令（MOVX）

MOVX 指令可访问当前 Bank 或“数据存储器”Bank（由 MEX3 中的 MXB19-MXB16 定义）中的数据。由 MEX3 中的位 MXM（MEX3.3）进行 Bank 选择。



### 长跳转指令（LJMP）

若需要跳转到扩展存储区的另一个 Bank，必须在执行 LJMP 指令之前将 MEX1 中的下一个 Bank 选择位 NB19 – NB16（MEX1.3 – MEX1.0）设置为合适的 Bank 地址。代码执行到 LJMP 时，NB19 – NB16 被复制到 MEX1 的当前 Bank 选择位 CB19 – CB16（MEX1.7 – MEX1.4），在下一个取指周期到来时出现在地址总线上。

*注： 下一个 Bank 选择位（NB19 – NB16）不被跳转指令修改。*

### 调用指令（LCALL 和 ACALL）

一旦执行 LCALL 指令，MMU 将进行以下操作：

1. 存储器扩展堆栈指针加 1。
2. MEX1 寄存器的内容出现在数据总线上。
3. MEXSP 寄存器中的位 [6:0] 出现在地址总线上。
4. 将存储器扩展堆栈读和写信号设置为进行写操作。
5. 对存储器扩展堆栈执行写操作。
6. 将下一个 Bank 选择位 NB19 – NB16（MEX1.3 – MEX1.0）复制到 CB19 – CB16（MEX1.7 – MEX1.4）。

### 返回指令（RET 和 RETI）

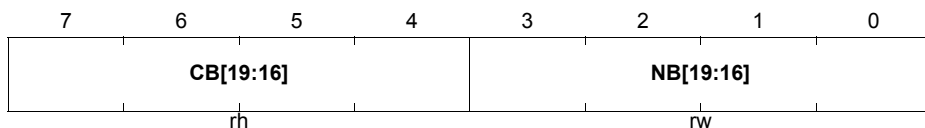
从子程序返回时，MMU 将进行以下操作：

1. MEXSP 寄存器中的位 [6:0] 出现在地址总线上。
2. 将存储器扩展堆栈读和写信号设置为进行读操作。
3. 对存储器扩展堆栈执行读操作。
4. 将存储器扩展堆栈的数据写入 MEX1 寄存器。
5. 存储器扩展堆栈指针减 1。

### 2.3.3 存储器扩展寄存器

#### MEX1

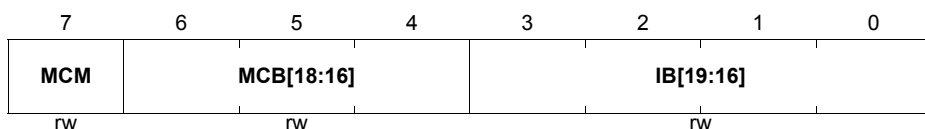
存储器扩展寄存器 1

复位值: 00<sub>H</sub>


符号	位	类型	描述
<b>NB[19:16]</b>	[3:0]	rw	下一个 <b>Bank</b> 编号
<b>CB[19:16]</b>	[7:4]	rh	当前 <b>Bank</b> 编号

#### MEX2

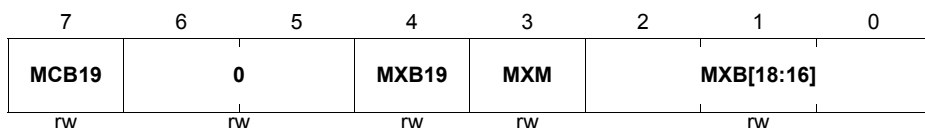
存储器扩展寄存器 2

复位值: 00<sub>H</sub>


符号	位	类型	描述
<b>IB[19:16]</b>	[3:0]	rw	中断 <b>Bank</b> 编号
<b>MCB[18:16]</b>	[6:4]	rw	存储器常数 <b>Bank</b> 编号（结合 MEX3.7）
<b>MCM</b>	7	rw	存储器常数模式 0     MOV <sub>C</sub> 访问当前 <b>Bank</b> 中的数据 1     MOV <sub>C</sub> 访问存储器常数 <b>Bank</b> 中的数据

#### MEX3

存储器扩展寄存器 3

复位值: 00<sub>H</sub>


符号	位	类型	描述
<b>MXB[19:16]</b>	4, [2:0]	rw	<b>XRAM Bank 编号</b>
<b>MXM</b>	3	rw	<b>XRAM Bank 选择</b> 0 MOVX 访问当前 Bank 中的数据 1 MOVX 访问 XRAM Bank 中的数据
<b>MCB19</b>	7	rw	<b>存储器常数 Bank 编号 MSB</b>
<b>0</b>	[6:5]	rw	<b>保留</b> 读操作返回 0，应写入 0。

**MEXSP**
**存储器扩展堆栈指针寄存器**
**复位值：7F<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>0</b>	<b>MXSP</b>						
r	rwh						

符号	位	类型	描述
<b>MXSP</b>	[6:0]	rwh	<b>存储器扩展堆栈指针</b> 它提供的堆栈深度最多为 128 字节。在执行调用指令之前自动加 1；执行返回指令之后自动减 1。
<b>0</b>	7	r	<b>保留</b> 读操作返回 0，应写入 0。

## 2.4 指令时序

对于无等待状态的存储器访问，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。执行一条指令需要一个、两个或者四个机器周期。通常在一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读取。

可以在任一拍 (P1 或 P2) 访问 Flash，指令执行时间延长一个机器周期（插入一个等待状态）。

**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)** 给出单字节单周期 (1 机器周期) 指令的时序图。该时序图表示下一个操作码 (C1P2) 从无需等待状态的存储器中读取，指令在一个周期内完成。

**图 2-2 (b)** 给出双字节单周期 (1 机器周期) 指令的时序图。该时序图表示第二个字节 (C1P1) 和下一个操作码 (C1P2) 从无需等待状态的存储器中读取，指令在一个周期内完成。

**图 2-2 (c)** 给出单字节双周期 (2 机器周期) 指令的时序图。该时序图表示下一个操作码 (C2P2) 从无需等待状态的存储器中读取，指令在两个周期内完成。

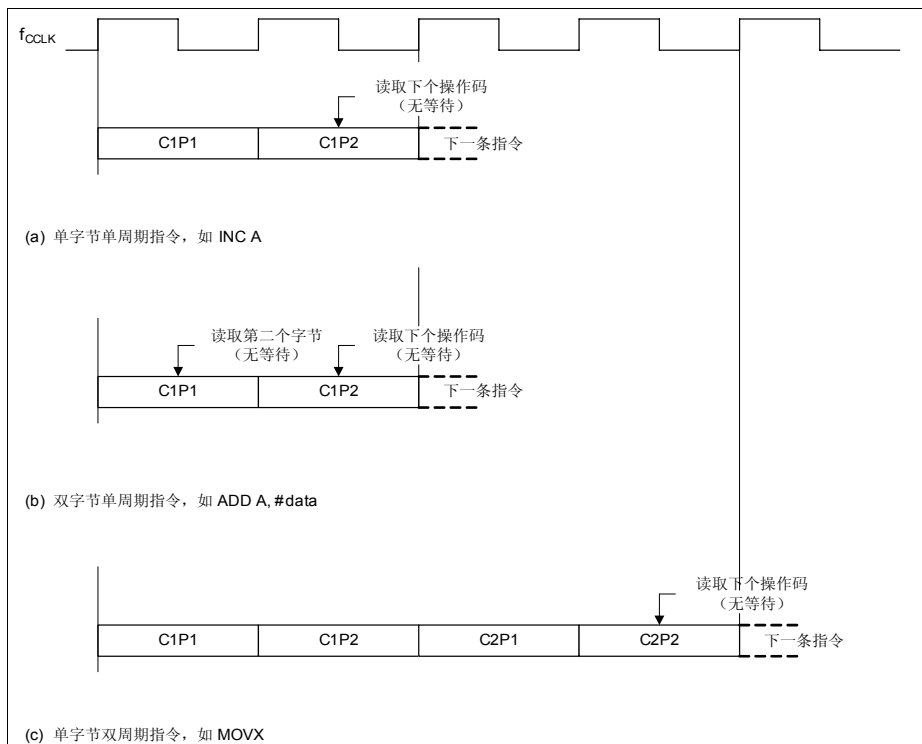


图 2-2 CPU 指令时序

## 处理器结构

指令长度为 1 个、2 个或 3 个字节，参见表 2-1 “字节数” 一行。XC878 执行每条指令时，所需时间包括：

- 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）（XC878 CPU 流水线结构）

注：XC878 CPU 在执行当前指令的同时预取下一条指令的操作码。

**表 2-1** 列出执行每条指令所需的时钟周期数，该值适用于从无需等待状态的快速存储器（如 Boot ROM 和 XRAM）中读取操作数和操作码的情况。表格最后一列给出标准 8051 处理器的指令时钟数，以便和 XC878 CPU 的性能进行比较。XC878 CPU 执行指令的速度是标准 8051 处理器的 6 倍。

**表 2-1 CPU 指令周期**

助记符	十六进制码	字节	$f_{\text{CCLK}}$ 时钟数	
			XC878 (无等待状态)	8051
算术运算				
ADD A,Rn	28-2F	1	2	12
ADD A,dir	25	2	2	12
ADD A,@Ri	26-27	1	2	12
ADD A,#data	24	2	2	12
ADDC A,Rn	38-3F	1	2	12
ADDC A,dir	35	2	2	12
ADDC A,@Ri	36-37	1	2	12
ADDC A,#data	34	2	2	12
SUBB A,Rn	98-9F	1	2	12
SUBB A,dir	95	2	2	12
SUBB A,@Ri	96-97	1	2	12
SUBB A,#data	94	2	2	12
INC A	04	1	2	12
INC Rn	08-0F	1	2	12
INC dir	05	2	2	12
INC @Ri	06-07	1	2	12
DEC A	14	1	2	12
DEC Rn	18-1F	1	2	12

表 2-1 CPU 指令周期

助记符	十六进制码	字节	$f_{\text{CCLK}}$ 时钟数	
			XC878 (无等待状态)	8051
DEC dir	15	2	2	12
DEC @Ri	16-17	1	2	12
INC DPTR	A3	1	4	24
MUL AB	A4	1	8	48
DIV AB	84	1	8	48
DA A	D4	1	2	12
逻辑运算				
ANL A,Rn	58-5F	1	2	12
ANL A,dir	55	2	2	12
ANL A,@Ri	56-57	1	2	12
ANL A,#data	54	2	2	12
ANL dir,A	52	2	2	12
ANL dir,#data	53	3	4	24
ORL A,Rn	48-4F	1	2	12
ORL A,dir	45	2	2	12
ORL A,@Ri	46-47	1	2	12
ORL A,#data	44	2	2	12
ORL dir,A	42	2	2	12
ORL dir,#data	43	3	4	24
XRL A,Rn	68-6F	1	2	12
XRL A,dir	65	2	2	12
XRL A,@Ri	66-67	1	2	12
XRL A,#data	64	2	2	12
XRL dir,A	62	2	2	12
XRL dir,#data	63	3	4	24
CLR A	E4	1	2	12
CPL A	F4	1	2	12
SWAP A	C4	1	2	12
RL A	23	1	2	12

**表 2-1 CPU 指令周期**

助记符	十六进制码	字节	$f_{\text{CCLK}}$ 时钟数	
			<b>XC878</b> (无等待状态)	<b>8051</b>
RLC A	33	1	2	12
RR A	03	1	2	12
RRC A	13	1	2	12
<b>数据传送</b>				
MOV A,Rn	E8-EF	1	2	12
MOV A,dir	E5	2	2	12
MOV A,@Ri	E6-E7	1	2	12
MOV A,#data	74	2	2	12
MOV Rn,A	F8-FF	1	2	12
MOV Rn,dir	A8-AF	2	4	24
MOV Rn,#data	78-7F	2	2	12
MOV dir,A	F5	2	2	12
MOV dir,Rn	88-8F	2	4	24
MOV dir,dir	85	3	4	24
MOV dir,@Ri	86-87	2	4	24
MOV dir,#data	75	3	4	24
MOV @Ri,A	F6-F7	1	2	12
MOV @Ri,dir	A6-A7	2	4	24
MOV @Ri,#data	76-77	2	2	12
MOV DPTR,#data	90	3	4	24
MOVC A,@A+DPTR	93	1	4	24
MOVC A,@A+PC	83	1	4	24
MOVX A,@Ri	E2-E3	1	4	24
MOVX A,@DPTR	E0	1	4	24
MOVX @Ri,A	F2-F3	1	4	24
MOVX @DPTR,A	F0	1	4	24
PUSH dir	C0	2	4	24
POP dir	D0	2	4	24
XCH A,Rn	C8-CF	1	2	12



表 2-1 CPU 指令周期

助记符	十六进制码	字节	$f_{\text{CCLK}}$ 时钟数	
			XC878 (无等待状态)	8051
XCH A,dir	C5	2	2	12
XCH A,@Ri	C6-C7	1	2	12
XCHD A,@Ri	D6-D7	1	2	12
位操作				
CLR C	C3	1	2	12
CLR bit	C2	2	2	12
SETB C	D3	1	2	12
SETB bit	D2	2	2	12
CPL C	B3	1	2	12
CPL bit	B2	2	2	12
ANL C,bit	82	2	4	24
ANL C,/bit	B0	2	4	24
ORL C,bit	72	2	4	24
ORL C,/bit	A0	2	4	24
MOV C,bit	A2	2	2	12
MOV bit,C	92	2	4	24
控制转移				
ACALL addr11	11->F1	2	4	24
LCALL addr16	12	3	4	24
RET	22	1	4	24
RETI	32	1	4	24
AJMP addr 11	01->E1	2	4	24
LJMP addr 16	02	3	4	24
SJMP rel	80	2	4	24
JC rel	40	2	4	24
JNC rel	50	2	4	24
JB bit,rel	20	3	4	24
JNB bit,rel	30	3	4	24
JBC bit,rel	10	3	4	24

**表 2-1 CPU 指令周期**

助记符	十六进制码	字节	$f_{\text{CCLK}}$ 时钟数	
			<b>XC878</b> (无等待状态)	<b>8051</b>
JMP @A+DPTR	73	1	4	24
JZ rel	60	2	4	24
JNZ rel	70	2	4	24
CJNE A,dir,rel	B5	3	4	24
CJNE A,#d,rel	B4	3	4	24
CJNE Rn,#d,rel	B8-BF	3	4	24
CJNE @Ri,#d,rel	B6-B7	3	4	24
DJNZ Rn,rel	D8-DF	2	4	24
DJNZ dir,rel	D5	3	4	24
<b>其它指令</b>				
NOP	00	1	2	12
<b>附加指令</b>				
MOVC @(DPTR++),A	A5	1	4	—
TRAP	A5	1	2	—

### 3 存储器结构

XC878 的 CPU 可寻址以下五个地址空间：

- 52/64 KB Flash 程序存储器
- 8 KB Boot ROM 程序存储器
- 256 B 内部 RAM 数据存储器
- 3 KB 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

图 3-1 所示为内嵌 64 KB Flash 的存储器地址空间分配。

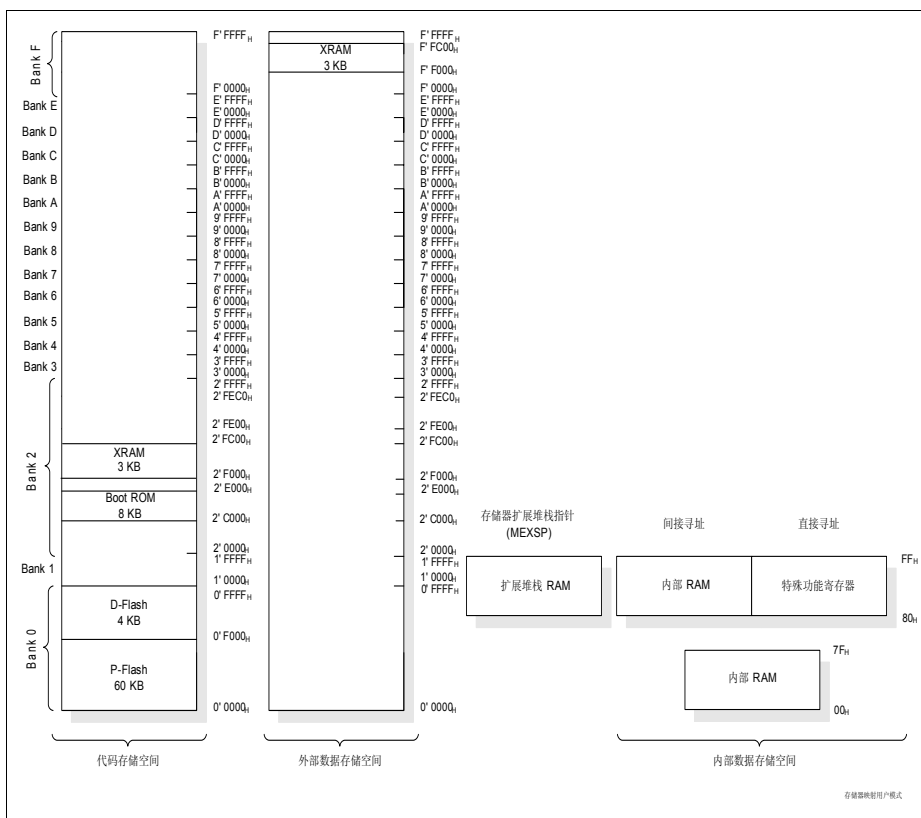


图 3-1 内嵌 64K Flash XC878 的存储空间映射（用户模式）

图 3-2 所示为内嵌 52 KB Flash 的存储器地址空间分配。

## 存储器结构

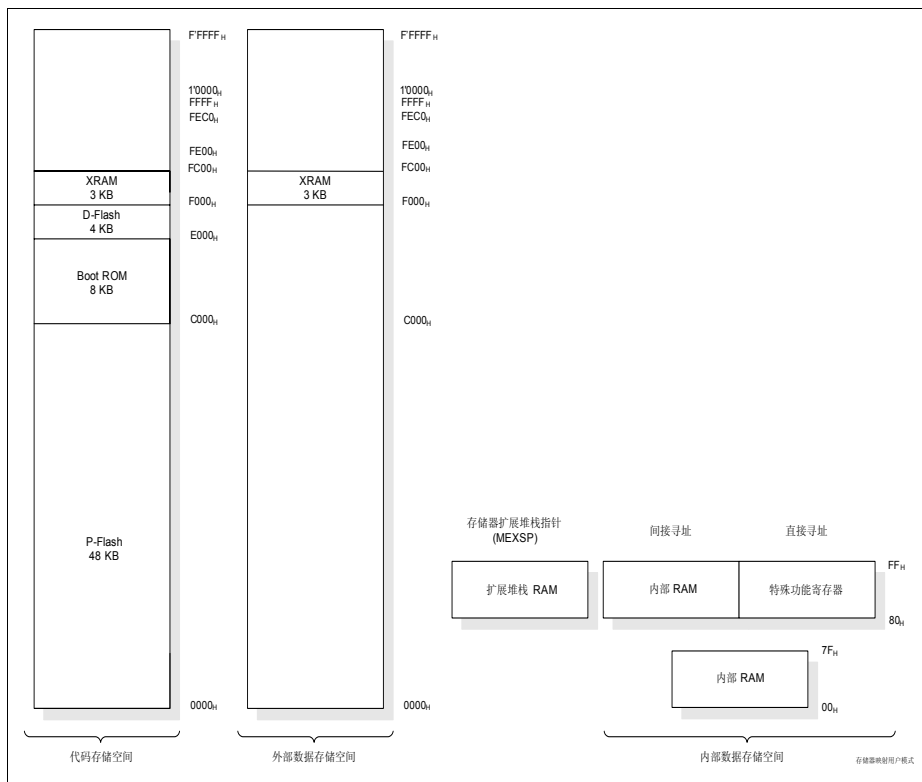


图 3-2 内嵌 52K Flash XC878 的存储空间映射（用户模式）

### 3.1 程序存储器

使用一个专用接口将 CPU 和程序存储器直接相连（而无需任何引脚连接），CPU 的性能得到优化。这意味着在每个时钟的上升沿均能读取代码。在 XC878 中，通过 4 位当前 Bank 指针（CB）或 4 位存储器常数 Bank 指针（MC）实现程序存储器扩展，由位 MCM 控制 Bank 选择。从 CB 或 MC 指向的 64KB 存储段中读取数据，具体描述见[章节 2.3](#)。从 CB 或 MC 指向的 64KB 存储段中读取程序代码。

### 3.2 数据存储器

数据存储器由内部存储器和外部存储器组成。“内部”和“外部”用来区分寄存器存储器和使用“MOVX”指令访问的数据存储器，它们并不是指外部数据存储器位于片外。XC878 中包含一个 3 KB 的片上外部数据存储器 XRAM。内部数据存储区使用 8 位地址寻址。片上 XRAM 通过“MOVX”指令使用 8 位或 16 位间接地址寻址，并利用最多 4 位进行扩展存储器 Bank 选择。

#### 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三个不同区域：256B RAM、128B 特殊功能寄存器（SFR）区和 128 B 扩展存储器堆栈 RAM。RAM 的高 128 B、SFR 和 128 B 扩展存储器堆栈 RAM 共用相同的地址段，通过不同的寻址方式访问。RAM 的低 128 B 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 只能通过直接寻址方式访问；128 B 扩展存储器堆栈 RAM 不可被直接访问，其地址由寄存器 MEXSP 给出。

RAM 地址段  $20_H$  到  $2F_H$  中的 16 个字节可位寻址；地址段  $30_H$  到  $7F_H$  可用作暂存寄存器或堆栈。

### 3.2.2 外部数据存储器的

3 KB 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 中的值定义。因此，在寄存器 XADDRH 中设置 XRAM 高位地址的写指令必须位于 MOVX 指令之前。

访问片上 XRAM 和外部数据存储器时，必须还要选择相应的存储 Bank。根据位 MXM 的设置选择 4 位 XRAM Bank 指针（位于 MEX3.MXB 中）或 4 位当前 Bank 指针（位于 MEX1.CB 中）。从 CB 或 MXB 指向的 64KB 存储段中读取数据，具体描述见 [章节 2.3](#)。

#### 3.2.2.1 访问外部数据存储器

##### 使用 DPTR 访问外部数据存储器（16 位寻址模式）

可通过 16 位寄存器 DPTR 对外部数据存储器间接寻址。这些指令有：

- MOVX A, @DPTR （读）
- MOVX @DPTR, A （写）

若地址指向片上 XRAM，将会访问片上自带的 XRAM。位于该地址段的外部数据存储器不会被访问，因为此时不产生外部总线周期。在执行这些指令之前，必须正确设置相应的存储 Bank。

##### 使用寄存器 R0/R1 访问外部数据存储器（8 位寻址模式）

还可使用寄存器 R0 或 R1 通过 8 位地址对外部数据存储器 and 片上 XRAM 间接寻址。这些指令有：

- MOVX A, @Ri （读）
- MOVX @Ri, A （写）

当前寄存器 Bank 中 R0 或 R1 的值为低字节地址端口提供 8 位地址。对于外部 I/O 扩展译码或较小的 RAM 阵列而言，这 8 位地址已足够使用。对于较大的 RAM 阵列，可使用任意数目的端口引脚来输出高位地址。这些引脚应由一条输出指令控制，该指令须位于 MOVX 指令之前。

为了区分片上 XRAM 访问和外部数据存储器访问，必须初始化寄存器 XADDRH。执行 8 位访问时，地址的高位字节由寄存器 XADDRH 中的值决定。若 XADDRH 的值指向地址段 F0<sub>H</sub>-FB<sub>H</sub>，并已选择正确的存储 Bank，则访问片上 XRAM（地址范围为 F000<sub>H</sub>-FBFF<sub>H</sub>），执行 MOVX 指令时从寄存器 XADDRH 中读取地址的高位字节。因此，在寄存器 XADDRH 中设置 XRAM 高位地址的写指令必须位于 MOVX 指令之前。位于该地址段的外部数据存储器不会被访问，因为此时不产生外部总线周期（数据、地址和控制线保持高电平）。

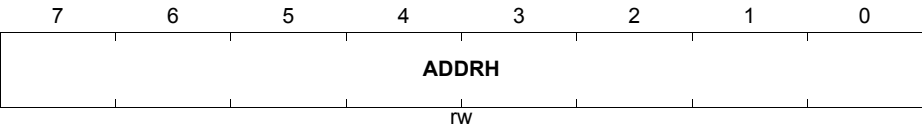
若超出该地址范围，MOVX 将不访问片上 XRAM。若外部数据存储器对应的端口引脚（读、写、地址引脚等）允许用作复用功能，则可访问外部数据存储器。

**存储器结构**

在执行这些指令之前，必须选择正确的存储 Bank。

**XADDRH**  
 片上 **XRAM** 的高位地址

复位值: **F0<sub>H</sub>**



符号	位	类型	描述
<b>XADDRH</b>	7:0	rw	片上 <b>XRAM</b> 的高位地址 在 XC878 中，片上 <b>XRAM</b> 的高位地址为从 F0 <sub>H</sub> - FB <sub>H</sub> 。 若写入该寄存器的值超出该范围，则不访问片上 <b>XRAM</b> 。此时可访问外部数据存储器。

3.2.2.2 外部数据存储接口

XC878 提供了一个外部接口用于扩展数据存储器。外部接口的标准连接如图 3-3 所示。XC878 只支持外部数据访问、不支持外部代码访问。此外，该接口只支持无需等待状态的存储器访问。若实际应用只访问片上 XRAM，外部接口端口可用作其它复用功能或通用 I/O。访问片上 XRAM 时不产生外部总线周期。

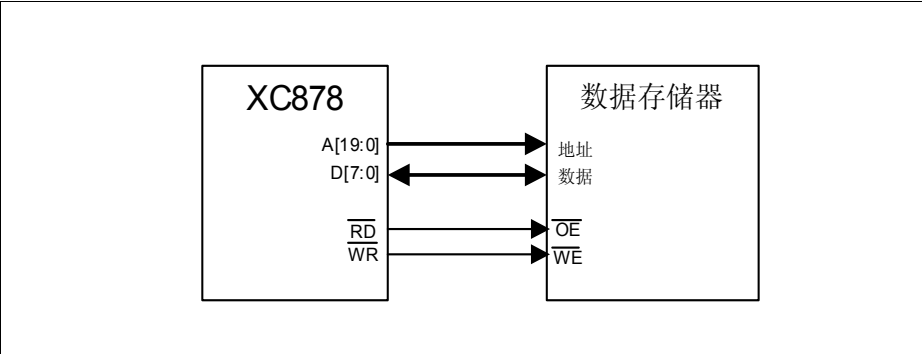


图 3-3 外部接口连接

由寄存器 EINTCON 控制外部接口。若通过 GLOBEN 使能外部接口，相应的端口引脚被使用作外部存储器的地址（若由位 ENAx/ENAH 使能）、数据和控制线，此时不考虑端口复用选择 ALTSEL 的设置。8 位数据总线的方向可由全局位 DDIR 控制，此时和端口方向 DIR 的设置无关。相应的端口控制位 ALTSEL 和 DIR 的内容保持不变、不会由于寄存器 EINTCON 的设置而改变。上拉 / 下拉选择和使能、驱动能力、漏极开路等其它端口设置由标准端口控制寄存器控制。有关端口控制寄存器的详细描述请参见第 6 章。

EINTCON

外部接口控制寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
ENA19	ENA18	ENA17	ENA16	ENAH	DDIRW	DDIR	GLOBEN
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
GLOBEN	0	rw	外部接口全局使能
			0 禁用外部接口
			1 使能外部接口



## 存储器结构

符号	位	类型	描述
<b>DDIR</b>	1	rw	<b>外部数据方向选择</b> 0 设置数据端口为输入口 1 设置数据端口为输出口 可在实际应用中动态切换。
<b>DDIRW</b>	2	rw	<b>仅使能设置位 DDIR</b> 0 EINTCON 中的所有位可写 1 只有位 DDIR 可写。寄存器中的其它位保持不变。
<b>ENAH</b>	3	rw	<b>使能外部地址的高位字节</b> 0 禁用地址线 A8 - A15 1 使能地址线 A8 - A15
<b>ENAx(x = 16-19)</b>	[7:4]	rw	<b>单独使能外部地址 Ax</b> 0 禁用地址线 Ax 1 使能地址线 Ax

### 3.3 存储器保护策略

XC878 的存储器保护策略包括：

- 基本保护：用户可通过 **boot** 选项禁止对所有存储器的外部访问。OCDS 功能和代码下载至 XRAM 和 Flash 的功能将被禁用。
- 读保护：用户可保护 Flash 中的内容不被读取。这是对基本保护的补充。
- Flash 编程和擦除保护：用户可保护 Flash 中的内部不被无意擦除和编程。该特性和读保护共同使用。

通过引导程序加载 (BSL) 模式 6 设定一个有效密码 (16 位非 1 值) 来激活这些保护方案。

#### 3.3.1 Flash 存储器保护

只要密码有效，对器件（包括 Flash）的所有外部访问均被禁止。

为了提供进一步 Flash 保护，可使能 Flash 硬件保护以实现二级读保护并使能编程和擦除保护。

Flash 硬件保护有两种模式：

- 模式 0：只保护 P-Flash；不保护 D-Flash。
- 模式 1：P-Flash 和 D-Flash 均被保护。

每种保护模式的选择及限制条件总结见[表 3-1](#)。

表 3-1 Flash 保护模式

Flash 保护	Flash 保护模式		
无硬件保护	硬件保护		
硬件保护模式	-	0	1
激活	通过 BSL 模式 6 设置一个有效密码		
选择	密码的位 13 = 0	密码的位 13 = 1 密码的 MSB = 0	密码的位 13 = 1 密码的 MSB = 1
P-Flash 的内容可由哪里的指令读出	任意 P-Flash 中的读指令	P-Flash 中的读指令	P-Flash 或 D-Flash 中的读指令
外部对 P-Flash 的访问	不可能	不可能	不可能
P-Flash 编程和擦除	可能	仅在密码的 (MSB-1) 被置 1 的情况下才可能	仅在密码的 (MSB-1) 被置 1 的情况下才可能
D-Flash 的内容可由哪里的指令读出	任意 P-Flash 中的读指令	任意 P-Flash 中的读指令	P-Flash 或 D-Flash 中的读指令
外部对 D-Flash 的访问	不可能	不可能	不可能
D-Flash 编程	可能	可能	可能，当密码的 (MSB-1) 被置 1 时
D-Flash 擦除	可能	满足以下条件时可能： <ul style="list-style-type: none"> <li>每次擦除之前 MISC_CON.DFLASHEN 被置 1；或</li> <li>密码的 (MSB-1) 被置 1</li> </ul>	可能，当密码的 (MSB-1) 被置 1 时

Flash 硬件保护模式 0 下，只有当寄存器 MISC\_CON 中的位 DFLASHEN 置 1 时，才能对 D-Flash 进行擦除操作。每次擦除操作结束时，DFLASHEN 由硬件自动清零。因此，在每次擦除 D-Flash 之前，必须置位 DFLASHEN。在 D-Flash 在系统擦除过程中，由引导程序加载 (BSL) 程序进行 DFLASHEN 的设置；每次进行 D-Flash 在应用擦除之前，必须由用户应用程序来置位 DFLASHEN。该额外的步骤用于防止对 D-Flash 内容的无意破坏。

用户可编程密码的格式如表 3-2 所示。

表 3-2 用户可编程密码的位域

位	大小	作用	值
15	1 位	Flash 硬件保护模式选择位	0 选择 Flash 硬件保护模式 0
			1 选择 Flash 硬件保护模式 1
14	1 位	Flash 存储器编程和擦除选择位	0 禁止对 Flash 编程和擦除
			1 允许对 Flash 编程和擦除
13	1 位	Flash 硬件保护使能位	0 Flash 硬件保护不被激活
			1 Flash 硬件保护将被激活
12	1 位	选择 Flash 未受保护时将被擦除的区域	0 Flash 未受保护时只擦除 P-Flash
			1 Flash 未受保护时擦除 P-Flash 和 D-Flash
11:0	12 位	用户定义的密码	用户设置该密码只需遵循一个条件：该密码的位 8 到位 11 必须至少包含一个 0。

用于使能 Flash 保护的 BSL 模式 6 也可用来禁止 Flash 保护。此时必须由用户提供密码。禁止 Flash 保护时需要密码匹配。密码匹配时，被保护的 P-Flash 和 D-Flash 的内容连同所设定的密码均被自动擦除。若密码有效，下次复位后 Flash 硬件保护被重新使能或禁止。对于其它保护策略，无需复位。

尽管任何保护机制都不是完全可靠的，XC878 的存储器保护策略仍为通用微控制器提供了高级别的保护。

### 3.3.2 其它控制寄存器

MISC\_CON 寄存器中的使能位 DFLASHEN 用于控制擦除 D-Flash。若禁止 Flash 硬件保护或使能保护模式 1，位 DFLASHEN 不起作用。

#### MISC\_CON

其它控制寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
ADCETR0 _MUX	ADCETR1 _MUX			0			DFLASH- EN
r	r			r			rwh

符号	位	类型	描述
DFLASHEN	0	rwh	<p>使能 D-Flash 擦除</p> <p>0 不能擦除 D-Flash</p> <p>1 可以擦除 D-Flash</p> <p>每次擦除 D-Flash 之后，该位由硬件复位。</p> <p><i>注： 对该位的多余设置对XC878的系统操作没有相反的影响。</i></p>
0	[5:1]	r	<p>保留</p> <p>读操作返回 0，应写入 0。</p>

### 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 80H - FFH。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 80H 到 FFH，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 8FH 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 3-4 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。

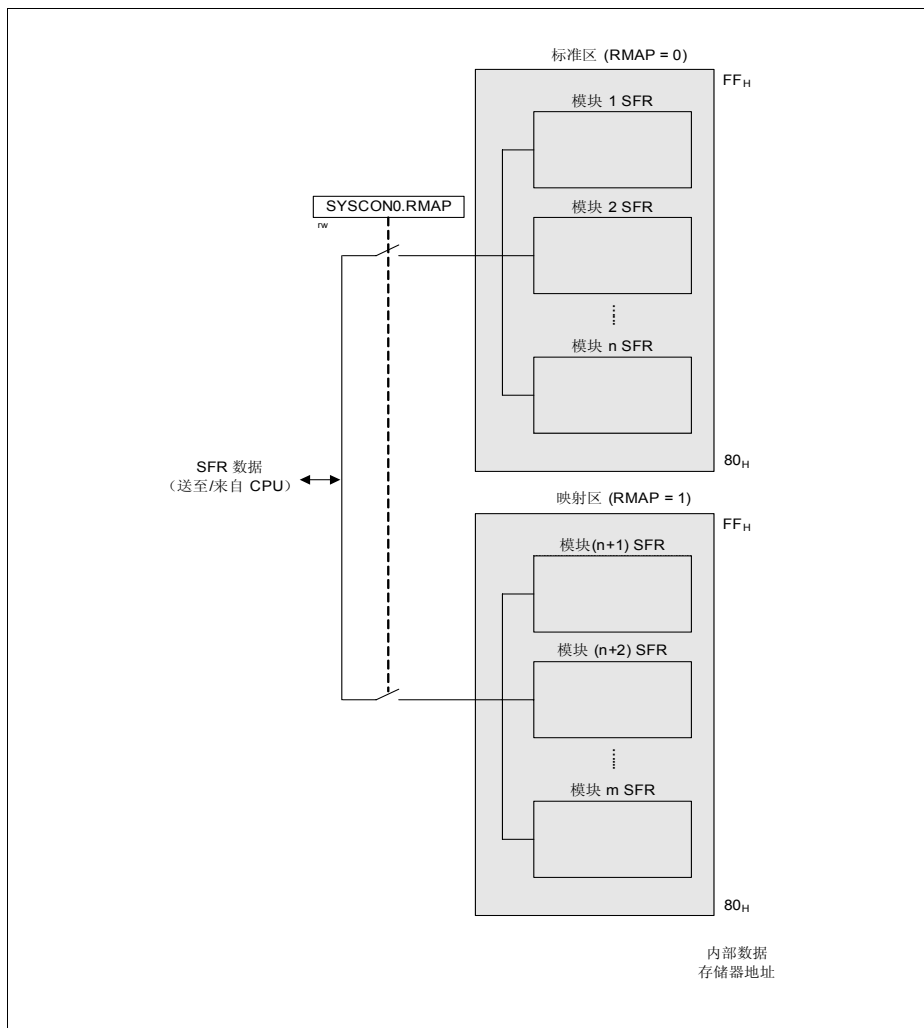


图 3-4 映射地址扩展

3.4.1.1 系统控制寄存器 0

SYSCON0 寄存器中的控制位用于 SFR 映射和中断结构 2 模式选择。

SYSCON0

系统控制寄存器 0

复位值：04<sub>H</sub>

7	6	5	4	3	2	1	0
	0		IMODE	0	1	0	RMAP
	r		rw	r	r	r	rw

符号	位	类型	描述
RMAP	0	rw	SFR 映射控制 0 禁止访问标准 SFR 区 1 使能访问标准 SFR 区
1	2	r	保留 读操作返回 1，应写入 1。
0	[7:5], 3,1	r	保留 读操作返回 0，应写入 0。

注： RMAP 位必须由 ANL 或 ORL 指令清零/置位。



### 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 XC878 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 3-5 所示。

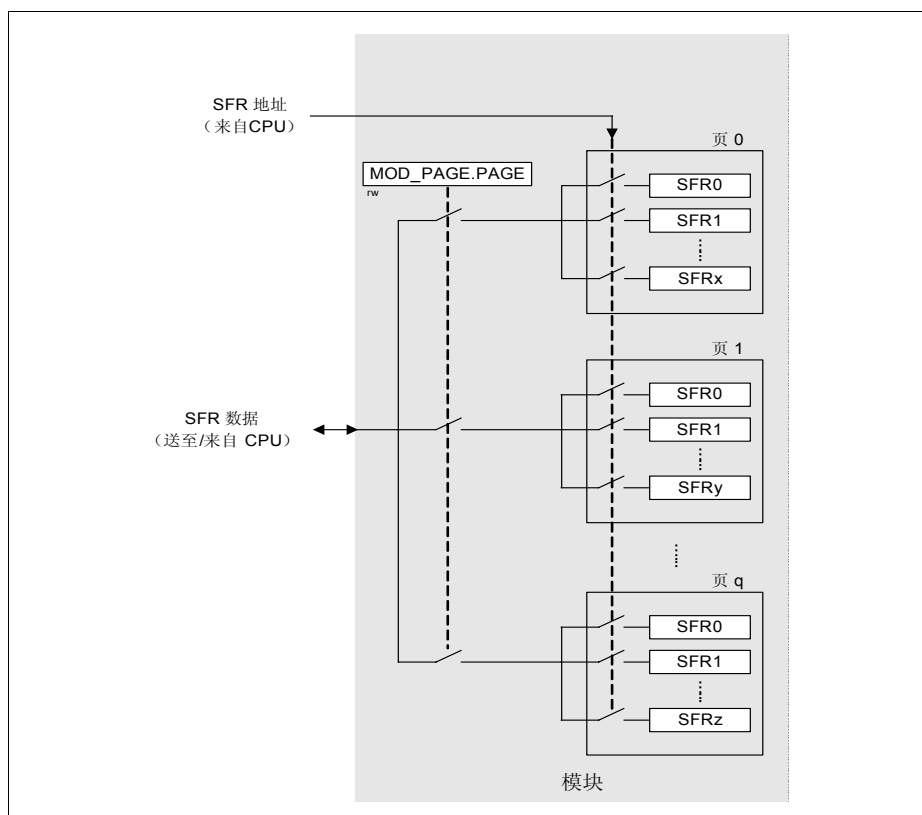


图 3-5 分页地址扩展

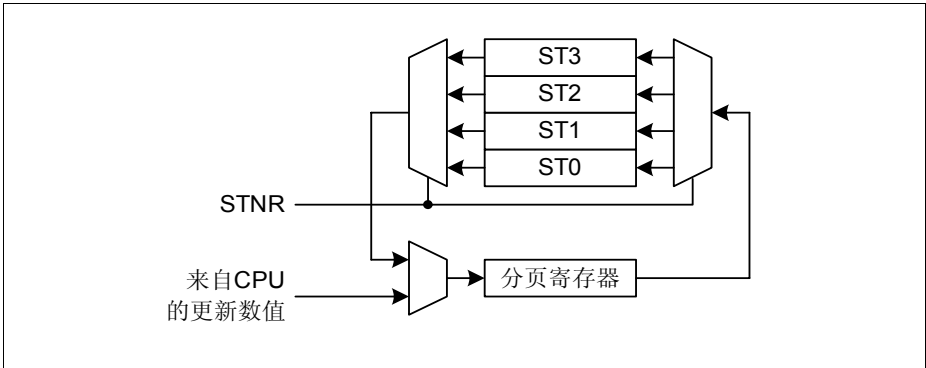
要访问位于（不同于当前页的）其它页上的寄存器，必须退出当前页。这需要重新设置分页寄存器的位域 PAGE，只有这样才能访问到所要的 SFR。

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 **STx** ( $x = 0 - 3$ ) 来保存和恢复当前页的设置。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 **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 **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 3-6 分页机制的页信息存储**

通过这种分页机制，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。

**XC878** 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
- 定时器 T2 捕获 / 比较单元（T2CCU）

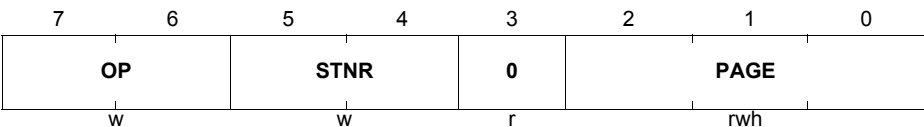
### 3.4.2.1 分页寄存器

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
PAGE	[2:0]	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。
STNR	[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 STx 中。 若 OP = 11 <sub>B</sub> ， PAGE 的内容被 STx 覆盖。写入 PAGE 的值不予理睬。  00 选择 ST0 01 选择 ST1 10 选择 ST2 11 选择 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 中的内容被保存在 STNR 指定的位域 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 STx 中的值覆盖。
0	3	r	<b>保留</b> 读操作返回 0，应写入 0。

### 3.4.3 位寻址

以格式  $1XXXX000_B$  (如:  $80_H$ ,  $88_H$ ,  $90_H$ , ... $F0_H$ ,  $F8_H$ ) 为地址的 SFR 均为可位寻址寄存器。

### 3.4.4 系统控制寄存器

系统控制 SFR 用来控制整个系统的功能，例如中断、可变波特率的产生、时钟管理、位保护方案、振荡器（OSC）和锁相环（PLL）控制。这些 SFR 位于标准 SFR 区（RMAP = 0），组织成两页。SCU\_PAGE 寄存器位于地址单元 BF<sub>H</sub>，包含分页值和页控制信息。

#### SCU\_PAGE

系统控制分页寄存器

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
PAGE	[2:0]	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。
STNR	[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 STx 中。 若 OP = 11 <sub>B</sub> ， PAGE 的内容被 STx 覆盖。写入 PAGE 的值不予理睬。  00 选择 ST0 01 选择 ST1 10 选择 ST2 11 选择 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 中的内容被保存在 STNR 指定的位域 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 STx 中的值覆盖。

## 存储器结构

符号	位	类型	描述
0	3	r	保留 读操作返回 0，应写入 0。

### 3.4.4.1 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：**N**、**P** 和 **K** 分频位、**NDIV**、**PDIV** 和 **KDIV**、**PLL** 输出旁路控制 **PLLBYP**、**XTAL** 掉电控制 **XPD**、振荡器源选择 **OSCSS**、**PLL** 掉电控制 **PLLPD**、看门狗定时器使能位 **WDTEN**、掉电模式和低速模式使能位 **PD** 和 **SD**。

#### PASSWD

口令寄存器

复位值：**07<sub>H</sub>**

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

符号	位	类型	描述
MODE	[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。
PROTECT_S	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。
PASS	[7:3]	w	<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 XC878 寄存器概览

XC878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 3.4.5.15](#)。

注：可位寻址SFR的地址以黑体标出。

#### 3.4.5.1 CPU 寄存器

CPU内核SFR可从标准存储器区和映射存储区访问(RMAP = 0或1)。

**表 3-3 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>	<b>SP</b> 堆栈指针寄存器	复位值: 07 <sub>H</sub>	位域									
		类型	rw									
82 <sub>H</sub>	<b>DPL</b> 数据指针寄存器, 低位字节	复位值: 00 <sub>H</sub>	位域									
		类型	rw	rw	rw	rw	rw	rw	rw	rw		
83 <sub>H</sub>	<b>DPH</b> 数据指针寄存器, 高位字节	复位值: 00 <sub>H</sub>	位域									
		类型	rw	rw	rw	rw	rw	rw	rw	rw		
87 <sub>H</sub>	<b>PCON</b> 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>	<b>TCON</b> 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>	<b>TMOD</b> 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>	<b>TL0</b> 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>	<b>TL1</b> 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>	<b>TH0</b> 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>	<b>TH1</b> 定时器 T1 寄存器, 高位字节	复位值: 00 <sub>H</sub>	位域	VAL								
		类型	rwh									
94 <sub>H</sub>	<b>MEX1</b> 存储器扩展寄存器 1	复位值: 00 <sub>H</sub>	位域	CB				NB				
		类型	r				rw					
95 <sub>H</sub>	<b>MEX2</b> 存储器扩展寄存器 2	复位值: 00 <sub>H</sub>	位域	MCM	MCB			IB				
		类型	rw	rw			rw					
96 <sub>H</sub>	<b>MEX3</b> 存储器扩展寄存器 3	复位值: 00 <sub>H</sub>	位域	MCB1 9	0		MXB1 9	MXM	MXB			
		类型	rw	rw		rw	rw	rw				
97 <sub>H</sub>	<b>MEXSP</b> 存储器扩展堆栈指针寄存器	复位值: 7F <sub>H</sub>	位域	0	MXSP							
		类型	r	rwh								



表 3-3 CPU 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
98 <sub>H</sub>	<b>SCON</b> 复位值: 00 <sub>H</sub> 串行通道控制寄存器	位域	SM0	SM1	SM2	REN	TB8	RB8	TI	RI
		类型	rw	rw	rw	rw	rw	rwh	rwh	rwh
99 <sub>H</sub>	<b>SBUF</b> 复位值: 00 <sub>H</sub> 串行数据缓冲寄存器	位域	VAL							
		类型	rwh							
A2 <sub>H</sub>	<b>EO</b> 复位值: 00 <sub>H</sub> 扩展操作寄存器	位域	0			TRAP EN	0			DPSE L0
		类型	r			rw	r			rw
A8 <sub>H</sub>	<b>IEN0</b> 复位值: 00 <sub>H</sub> 中断使能寄存器 0	位域	EA	0	ET2	ES	ET1	EX1	ET0	EX0
		类型	rw	r	rw	rw	rw	rw	rw	rw
B8 <sub>H</sub>	<b>IP</b> 复位值: 00 <sub>H</sub> 中断优先级寄存器	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> 复位值: 00 <sub>H</sub> 中断优先级寄存器, 高位字节	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> 复位值: 00 <sub>H</sub> 程序状态寄存器	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> 复位值: 00 <sub>H</sub> 累加寄存器	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> 复位值: 00 <sub>H</sub> 中断使能寄存器 1	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> 复位值: 00 <sub>H</sub> B 寄存器	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> 复位值: 00 <sub>H</sub> 中断优先级寄存器 1	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> 复位值: 00 <sub>H</sub> 中断优先级寄存器 1, 高位字节	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 可从映射存储器区访问（RMAP = 1）。

表 3-4 MDU 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 1										
B0 <sub>H</sub>	<b>MDUSTAT</b> MDU 状态寄存器	复位值: 00 <sub>H</sub> 位域 类型	0					BSY	IERR	IRDY
			r					rh	rwh	rwh
B1 <sub>H</sub>	<b>MDUCON</b> MDU 控制寄存器	复位值: 00 <sub>H</sub> 位域 类型	IE	IR	RSEL	STAR T	OPCODE			
			rw	rw	rw	rwh	rw			
B2 <sub>H</sub>	<b>MD0</b> MDU 数据寄存器 0	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rw							
B2 <sub>H</sub>	<b>MR0</b> MDU 结果寄存器 0	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rh							
B3 <sub>H</sub>	<b>MD1</b> MDU 数据寄存器 1	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rw							
B3 <sub>H</sub>	<b>MR1</b> MDU 结果寄存器 1	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rh							
B4 <sub>H</sub>	<b>MD2</b> MDU 数据寄存器 2	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rw							
B4 <sub>H</sub>	<b>MR2</b> MDU 结果寄存器 2	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rh							
B5 <sub>H</sub>	<b>MD3</b> MDU 数据寄存器 3	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rw							
B5 <sub>H</sub>	<b>MR3</b> MDU 结果寄存器 3	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rh							
B6 <sub>H</sub>	<b>MD4</b> MDU 数据寄存器 4	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rw							
B6 <sub>H</sub>	<b>MR4</b> MDU 结果寄存器 4	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rh							
B7 <sub>H</sub>	<b>MD5</b> MDU 数据寄存器 5	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rw							
B7 <sub>H</sub>	<b>MR5</b> MDU 结果寄存器 5	复位值: 00 <sub>H</sub> 位域 类型	DATA							
			rh							

### 3.4.5.3 CORDIC 寄存器

CORDIC SFR 从映射存储器区访问（RMAP = 1）。

表 3-5 CORDIC 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 1										
9A <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							
9B <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							
9C <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							
9D <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							
9E <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							
9F <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 3-6 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>	SYS <sub>CON</sub> 0 系统控制寄存器 0	复位值: 04 <sub>H</sub>	位域	0		IMOD E	0	1	0	RMAP
		类型	r		rw	r	r	r	rw	
RMAP = 0										
BF <sub>H</sub>	SCU <sub>PAGE</sub> 系统控制分页寄存器	复位值: 00 <sub>H</sub>	位域	OP		STNR	0	PAGE		
		类型	w		w	r	rwh			
RMAP = 0, 页 0										

表 3-6 SCU 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
B3 <sub>H</sub>	<b>MODISEL</b> 复位值: 00 <sub>H</sub> 外设输入选择寄存器	位域	0	URRI H	JTAGT DIS	JTAGT CKS	EXINT2 IS	EXINT 1IS	EXINT 0IS	URRI S
		类型	r	rw	rw	rw	rw	rw	rw	rw
B4 <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 6	EXINT 5	EXINT 4	EXINT3	EXINT 2	EXINT 1	EXINT 0
		类型	r	rwh	rwh	rwh	rwh	rwh	rwh	rwh
B5 <sub>H</sub>	<b>IRCON1</b> 复位值: 00 <sub>H</sub> 中断请求寄存器 1	位域	0	CANS RC2	CANS RC1	ADCS R1	ADCSR 0	RIR	TIR	EIR
		类型	r	rwh	rwh	rwh	rw	rw	rw	rw
B6 <sub>H</sub>	<b>IRCON2</b> 复位值: 00 <sub>H</sub> 中断请求寄存器 2	位域	0			CANS RC3	0			CANS RC0
		类型	r			rwh	r			rwh
B7 <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	
B8 <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		EXINT6		EXINT5		EXINT4	
		类型	r		rw		rw		rw	
B8 <sub>H</sub>	<b>NMICON</b> 复位值: 00 <sub>H</sub> NMI 控制寄存器	位域	0	NMI ECC	NMI VDDP	0	NMI OCDS	NMI FLASH	NMI PLL	NMI WDT
		类型	r	rw	rw	r	rw	rw	rw	rw
B9 <sub>H</sub>	<b>NMISR</b> 复位值: 00 <sub>H</sub> NMI 状态寄存器	位域	0	FNMI ECC	FNMI VDDP	0	FNMI OCDS	FNMI FLASH	FNMI PLL	FNMI WDT
		类型	r	rwh	rwh	r	rwh	rwh	rwh	rwh
BD <sub>H</sub>	<b>BCON</b> 复位值: 20 <sub>H</sub> 波特率控制寄存器	位域	BGSEL		NDOV EN	BRDIS	BRPRE			R
		类型	rw		rw	rw	rw			rw
BE <sub>H</sub>	<b>BG</b> 复位值: 00 <sub>H</sub> 波特率定时器 / 重载寄存器	位域	BR_VALUE							
		类型	rwh							
E9 <sub>H</sub>	<b>FDCON</b> 复位值: 00 <sub>H</sub> 分数分频器控制寄存器	位域	BGS	SYNE N	ERRS YN	EOFS YN	BRK	NDOV	FDM	FDEN
		类型	rw	rw	rwh	rwh	rwh	rwh	rw	rw
EA <sub>H</sub>	<b>FDSTEP</b> 复位值: 00 <sub>H</sub> 分数分频器重载寄存器	位域	STEP							
		类型	rw							
EB <sub>H</sub>	<b>FDRES</b> 复位值: 00 <sub>H</sub> 分数分频器结果寄存器	位域	RESULT							
		类型	rh							
RMAP = 0, 页 1										
B3 <sub>H</sub>	<b>ID</b> 复位值: 49 <sub>H</sub> ID 寄存器	位域	PRODID					VERID		
		类型	r					r		
B4 <sub>H</sub>	<b>PMCON0</b> 复位值: 80 <sub>H</sub> 功率模式控制寄存器 0	位域	VDDP WARN	WDT RST	WKRS	WK SEL	SD	PD	WS	
		类型	rh	rwh	rwh	rw	rw	rwh	rw	

## 存储器结构

表 3-6 SCU 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
B5 <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	0	CDC_ DIS	CAN_ DIS	MDU_ DIS	T2CCU_ DIS	CCU_ DIS	SSC_ DIS	ADC_ DIS
		类型	r	rw	rw	rw	rw	rw	rw	rw
B6 <sub>H</sub>	<b>OSC_CON</b> 复位值: XX <sub>H</sub> OSC 控制寄存器	位域	PLLRD RES	PLLBY P	PLLPD	0	XPD	OSC SS	EORD RES	EXTO SCR
		类型	rw	rw	rw	r	rw	rw	rw	rw
B7 <sub>H</sub>	<b>PLL_CON</b> 复位值: 18 <sub>H</sub> PLL 控制寄存器	位域	NDIV						PLL R	PLL_ OCK
		类型	rw						rh	rh
BA <sub>H</sub>	<b>CMCON</b> 复位值: 10 <sub>H</sub> 时钟控制寄存器	位域	KDIV		0	FCCF G	CLKREL			
		类型	rw		r	rw	rw			
BB <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	w					rh	rw	
BE <sub>H</sub>	<b>COCON</b> 复位值: 00 <sub>H</sub> 时钟输出控制寄存器	位域	COUTS		TLEN	0	COREL			
		类型	rw		rw	r	rw			
E9 <sub>H</sub>	<b>MISC_CON</b> 复位值: 00 <sub>H</sub> 其它控制寄存器	位域	ADCE TR0 MUX	ADCE TR1 MUX	0					DFLA SHEN
		类型	rw	rw	r					rw
EA <sub>H</sub>	<b>PLL_CON1</b> 复位值: 20 <sub>H</sub> PLL 控制寄存器 1	位域	NDIV			PDIV				
		类型	rw			rw				
EB <sub>H</sub>	<b>CR_MISC</b> 复位值: 00 <sub>H</sub> 或 01 <sub>H</sub> 复位状态寄存器	位域	CCCF G	MDUC CFG	CCUC CFG	T2CCF G	0			HDRS T
		类型	rw	rw	rw	rw	r			rw
RMAP = 0, 页 3										
B3 <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							
B4 <sub>H</sub>	<b>IRCON3</b> 复位值: 00 <sub>H</sub> 中断请求寄存器 3	位域	0		CANS RC5	CCU6 SR1	0		CANS RC4	CCU6 SR0
		类型	r		rw	rw	r		rw	rw
B5 <sub>H</sub>	<b>IRCON4</b> 复位值: 00 <sub>H</sub> 中断请求寄存器 4	位域	0		CANS RC7	CCU6 SR3	0		CANS RC6	CCU6 SR2
		类型	r		rw	rw	r		rw	rw
B6 <sub>H</sub>	<b>MODIEN</b> 复位值: 07 <sub>H</sub> 外设中断使能寄存器	位域	0			CM5E N	CM4EN	RIREN	TIREN	EIREN
		类型	r			rw	rw	rw	rw	rw
B7 <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	EXINT6IS			UR1RIS		T21EXI S	0	
		类型	rw			rw		rw	r	
BA <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			T2EXI S	T21IS	T2IS	T1IS	T0IS
		类型	r			rw	rw	rw	rw	rw

表 3-6 SCU 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
BB <sub>H</sub>	<b>PMCON2</b> 复位值: 00 <sub>H</sub> 功率模式控制寄存器 2	位域	0						UART 1_DIS	T21_D IS
		类型	r						rw	rw
BD <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		CCTS USP	T21SU SP	T2SUS P	T13SU SP	T12SU SP	WDT S USP
		类型	r		rw	rw	rw	rw	rw	rw
BE <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	0		CIS		SIS		MIS	
		类型	r		rw		rw		rw	
EA <sub>H</sub>	<b>MODPISEL4</b> 复位值: 00 <sub>H</sub> 外设输入选择寄存器 4	位域	0		EXINT5IS		EXINT4IS		EXINT3IS	
		类型	r		rw		rw		rw	

### 3.4.5.5 WDT 寄存器

WDT SFR 从映射存储区访问 (RMAP = 1)。

表 3-7 WDT 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 1										
BB <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	WDTP R	0	WDTE N	WDTR S	WDTI N
		类型	r		rw	rh	r	rw	rw	rw
BC <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							
BD <sub>H</sub>	<b>WDTWINB</b> 复位值: 00 <sub>H</sub> 看门狗窗界计数寄存器	位域	WDTWINB							
		类型	rw							
BE <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							
BF <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							

### 3.4.5.6 端口寄存器

端口 SFR 从标准存储器区访问（RMAP = 0）。

表 3-8 端口寄存器概览

地址	寄存器名		位	7	6	5	4	3	2	1	0
RMAP = 0											
B2 <sub>H</sub>	<b>PORT_PAGE</b> 端口分页寄存器	复位值: 00 <sub>H</sub>	位域	OP		STNR		0	PAGE		
			类型	w		w		r	rwh		
RMAP = 0, 页 0											
80 <sub>H</sub>	<b>P0_DATA</b> P0 口数据寄存器	复位值: 00 <sub>H</sub>	位域	P7	P6	P5	P4	P3	P2	P1	P0
			类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
86 <sub>H</sub>	<b>P0_DIR</b> P0 口方向寄存器	复位值: 00 <sub>H</sub>	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_DATA</b> P1 口数据寄存器	复位值: 00 <sub>H</sub>	位域	P7	P6	P5	P4	P3	P2	P1	P0
			类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
91 <sub>H</sub>	<b>P1_DIR</b> P1 口方向寄存器	复位值: 00 <sub>H</sub>	位域	P7	P6	P5	P4	P3	P2	P1	P0
			类型	rw	rw	rw	rw	rw	rw	rw	rw
92 <sub>H</sub>	<b>P5_DATA</b> P5 口数据寄存器	复位值: 00 <sub>H</sub>	位域	P7	P6	P5	P4	P3	P2	P1	P0
			类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
93 <sub>H</sub>	<b>P5_DIR</b> P5 口方向寄存器	复位值: 00 <sub>H</sub>	位域	P7	P6	P5	P4	P3	P2	P1	P0
			类型	rw	rw	rw	rw	rw	rw	rw	rw
B0 <sub>H</sub>	<b>P3_DATA</b> P3 口数据寄存器	复位值: 00 <sub>H</sub>	位域	P7	P6	P5	P4	P3	P2	P1	P0
			类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
B1 <sub>H</sub>	<b>P3_DIR</b> P3 口方向寄存器	复位值: 00 <sub>H</sub>	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>P4_DATA</b> P4 口数据寄存器	复位值: 00 <sub>H</sub>	位域	P7	P6	P5	P4	P3	P2	P1	P0
			类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
C9 <sub>H</sub>	<b>P4_DIR</b> P4 口方向寄存器	复位值: 00 <sub>H</sub>	位域	P7	P6	P5	P4	P3	P2	P1	P0
			类型	rw	rw	rw	rw	rw	rw	rw	rw
RMAP = 0, 页 1											
80 <sub>H</sub>	<b>P0_PUDSEL</b> P0 口上拉 / 下拉选择寄存器	复位值: FF <sub>H</sub>	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> P0 口上拉 / 下拉使能寄存器	复位值: C4 <sub>H</sub>	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> P1 口上拉 / 下拉选择寄存器	复位值: FF <sub>H</sub>	位域	P7	P6	P5	P4	P3	P2	P1	P0
			类型	rw	rw	rw	rw	rw	rw	rw	rw
91 <sub>H</sub>	<b>P1_PUDEN</b> P1 口上拉 / 下拉使能寄存器	复位值: FF <sub>H</sub>	位域	P7	P6	P5	P4	P3	P2	P1	P0
			类型	rw	rw	rw	rw	rw	rw	rw	rw
92 <sub>H</sub>	<b>P5_PUDSEL</b> P5 口上拉 / 下拉选择寄存器	复位值: FF <sub>H</sub>	位域	P7	P6	P5	P4	P3	P2	P1	P0
			类型	rw	rw	rw	rw	rw	rw	rw	rw

表 3-8 端口寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
93 <sub>H</sub>	<b>P5_PUDEN</b> 复位值: FF <sub>H</sub> P5 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
B0 <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
B1 <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
C8 <sub>H</sub>	<b>P4_PUDSEL</b> 复位值: FF <sub>H</sub> P4 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>P4_PUDEN</b> 复位值: 04 <sub>H</sub> P4 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, 页 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	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
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	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	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	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	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	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
92 <sub>H</sub>	<b>P5_ALTSEL0</b> 复位值: 00 <sub>H</sub> P5 口复用功能选择寄存器 0	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
93 <sub>H</sub>	<b>P5_ALTSEL1</b> 复位值: 00 <sub>H</sub> P5 口复用功能选择寄存器 1	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
B0 <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	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
B1 <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	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
C8 <sub>H</sub>	<b>P4_ALTSEL0</b> 复位值: 00 <sub>H</sub> P4 口复用功能选择寄存器 0	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>P4_ALTSEL1</b> 复位值: 00 <sub>H</sub> P4 口复用功能选择寄存器 1	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
RMAP = 0, 页 3										
80 <sub>H</sub>	<b>P0_OD</b> 复位值: 00 <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
86 <sub>H</sub>	<b>P0_DS</b> 复位值: FF <sub>H</sub> P0 口驱动能力控制寄存器	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: 00 <sub>H</sub> P1 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



表 3-8 端口寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
91 <sub>H</sub>	<b>P1_DS</b> 复位值: FF <sub>H</sub> P1 口驱动能力控制寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
92 <sub>H</sub>	<b>P5_OD</b> 复位值: 00 <sub>H</sub> P5 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
93 <sub>H</sub>	<b>P5_DS</b> 复位值: FF <sub>H</sub> P5 口驱动能力控制寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
B0 <sub>H</sub>	<b>P3_OD</b> 复位值: 00 <sub>H</sub> P3 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
B1 <sub>H</sub>	<b>P3_DS</b> 复位值: FF <sub>H</sub> P3 口驱动能力控制寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
C8 <sub>H</sub>	<b>P4_OD</b> 复位值: 00 <sub>H</sub> P4 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
C9 <sub>H</sub>	<b>P4_DS</b> 复位值: FF <sub>H</sub> P4 口驱动能力控制寄存器	位域	P7	P6	P5	P4	P3	P2	P1	P0
		类型	rw	rw	rw	rw	rw	rw	rw	rw

### 3.4.5.7 ADC 寄存器

ADC SFR 从标准存储器区访问 (RMAP = 0)。

表 3-9 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	rwh		
RMAP = 0, 页 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		0			
		类型	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_LCBR 复位值: B7 <sub>H</sub> 边界检测寄存器	位域	BOUND1				BOUND0			
		类型	rw				rw			
CE <sub>H</sub>	ADC_INPCR0 复位值: 00 <sub>H</sub> 输入综合寄存器 0	位域	STC							
		类型	rw							
CF <sub>H</sub>	ADC_ETRCR 复位值: 00 <sub>H</sub> 外部触发控制寄存器	位域	SYNE N1	SYNE N0	ETRSEL1			ETRSEL0		
		类型	rw	rw	rw			rw		

表 3-9 ADC 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 0, 页 1										
CA <sub>H</sub>	ADC_CHCTR0 复位值: 00 <sub>H</sub> 通道控制寄存器 0	位域	0	LCC			0	RESRSEL		
		类型	r	rw			r	rw		
CB <sub>H</sub>	ADC_CHCTR1 复位值: 00 <sub>H</sub> 通道控制寄存器 1	位域	0	LCC			0	RESRSEL		
		类型	r	rw			r	rw		
CC <sub>H</sub>	ADC_CHCTR2 复位值: 00 <sub>H</sub> 通道控制寄存器 2	位域	0	LCC			0	RESRSEL		
		类型	r	rw			r	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			0	RESRSEL		
		类型	r	rw			r	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			0	RESRSEL		
		类型	r	rw			r	rw		
CF <sub>H</sub>	ADC_CHCTR5 复位值: 00 <sub>H</sub> 通道控制寄存器 5	位域	0	LCC			0	RESRSEL		
		类型	r	rw			r	rw		
D2 <sub>H</sub>	ADC_CHCTR6 复位值: 00 <sub>H</sub> 通道控制寄存器 6	位域	0	LCC			0	RESRSEL		
		类型	r	rw			r	rw		
D3 <sub>H</sub>	ADC_CHCTR7 复位值: 00 <sub>H</sub> 通道控制寄存器 7	位域	0	LCC			0	RESRSEL		
		类型	r	rw			r	rw		
RMAP = 0, 页 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		0	VF	DRC	CHNR		
		类型	rh		r	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		0	VF	DRC	CHNR		
		类型	rh		r	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		0	VF	DRC	CHNR		
		类型	rh		r	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		0	VF	DRC	CHNR		
		类型	rh		r	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, 页 3										
CA <sub>H</sub>	ADC_RESRA0L 复位值: 00 <sub>H</sub> 结果寄存器 0, 累加读取低位字节	位域	RESULT			VF	DRC	CHNR		
		类型	rh			rh	rh	rh		

表 3-9 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_RESRA0H 复位值: 00 <sub>H</sub> 结果寄存器 0, 累加读取高位字节	位域	RESULT							
		类型	rh							
CC <sub>H</sub>	ADC_RESRA1L 复位值: 00 <sub>H</sub> 结果寄存器 1, 累加读取低位字节	位域	RESULT			VF	DRC	CHNR		
		类型	rh			rh	rh	rh		
CD <sub>H</sub>	ADC_RESRA1H 复位值: 00 <sub>H</sub> 结果寄存器 1, 累加读取高位字节	位域	RESULT							
		类型	rh							
CE <sub>H</sub>	ADC_RESRA2L 复位值: 00 <sub>H</sub> 结果寄存器 2, 累加读取低位字节	位域	RESULT			VF	DRC	CHNR		
		类型	rh			rh	rh	rh		
CF <sub>H</sub>	ADC_RESRA2H 复位值: 00 <sub>H</sub> 结果寄存器 2, 累加读取高位字节	位域	RESULT							
		类型	rh							
D2 <sub>H</sub>	ADC_RESRA3L 复位值: 00 <sub>H</sub> 结果寄存器 3, 累加读取低位字节	位域	RESULT			VF	DRC	CHNR		
		类型	rh			rh	rh	rh		
D3 <sub>H</sub>	ADC_RESRA3H 复位值: 00 <sub>H</sub> 结果寄存器 3, 累加读取高位字节	位域	RESULT							
		类型	rh							
RMAP = 0, 页 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			DRCT R
		类型	rw	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			DRCT R
		类型	rw	rw	r	rw	r			rw
CC <sub>H</sub>	ADC_RCR2 复位值: 00 <sub>H</sub> 结果控制寄存器 2	位域	VFCT R	WFR	0	IEN	0			DRCT R
		类型	rw	rw	r	rw	r			rw
CD <sub>H</sub>	ADC_RCR3 复位值: 00 <sub>H</sub> 结果控制寄存器 3	位域	VFCT R	WFR	0	IEN	0			DRCT R
		类型	rw	rw	r	rw	r			rw
CE <sub>H</sub>	ADC_VFCR 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
RMAP = 0, 页 5										
CA <sub>H</sub>	ADC_CHINFR 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>	ADC_CHINCR 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>	ADC_CHINSR 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 4	CHINS 3	CHINS 2	CHINS 1	CHINS 0
		类型	w	w	w	w	w	w	w	w

表 3-9 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_CHINPR</b> 复位值: 00 <sub>H</sub> 通道中断节点指针寄存器	位域	CHINP 7	CHINP 6	CHINP 5	CHINP 4	CHINP 3	CHINP 2	CHINP 1	CHINP 0
		类型	rw	rw	rw	rw	rw	rw	rw	rw
CE <sub>H</sub>	<b>ADC_EVINFR</b> 复位值: 00 <sub>H</sub> 事件中断标志寄存器	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
D3 <sub>H</sub>	<b>ADC_EVINPR</b> 复位值: 00 <sub>H</sub> 事件中断节点指针寄存器	位域	EVINP 7	EVINP 6	EVINP 5	EVINP 4	0		EVINP 1	EVINP 0
		类型	rw	rw	rw	rw	r		rw	rw
RMAP = 0, 页 6										
CA <sub>H</sub>	<b>ADC_CRCR1</b> 复位值: 00 <sub>H</sub> 转换请求控制寄存器 1	位域	CH7	CH6	CH5	CH4	0			
		类型	rwh	rwh	rwh	rwh	r			
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	0			
		类型	rwh	rwh	rwh	rwh	r			
CC <sub>H</sub>	<b>ADC_CMR1</b> 复位值: 00 <sub>H</sub> 转换请求模式寄存器 1	位域	Rsv	LDEV	CLRP ND	SCAN	ENSI	ENTR	0	ENGT
		类型	r	w	w	rw	rw	rw	r	rw
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	Rsv	0	EMPT Y	EV	0		FILL	
		类型	r	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		w		

### 3.4.5.8 定时器 T2 比较 / 捕获单元寄存器

定时器 T2 比较 / 捕获单元 SFR 从标准存储器区访问 (RMAP = 0)。

表 3-10 T2CCU 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 0										
C7 <sub>H</sub>	<b>T2_PAGE</b> 复位值: 00 <sub>H</sub> T2CCU 分页寄存器	位域	OP		STNR		0	PAGE		
		类型	w		w		r	rwh		
RMAP = 0, 页 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		EXEN 2	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	T2RE GS	T2RH EN	EDGE SEL	PREN	T2PRE			DCEN
		类型	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	EXF2E N
		类型	r						rw	rw
RMAP = 0, 页 1										
C0 <sub>H</sub>	<b>T2CCU_CCEN</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较使能寄存器	位域	CCM3		CCM2		CCM1		CCM0	
		类型	rw		rw		rw		rw	
C1 <sub>H</sub>	<b>T2CCU_CCTBSEL</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较时基选择寄存器	位域	CASC	CCTT OV	CCTB 5	CCTB 4	CCTB 3	CCTB 2	CCTB 1	CCTB 0
		类型	rw	rwh	rw	rw	rw	rw	rw	rw
C2 <sub>H</sub>	<b>T2CCU_CCTRELL</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较定时器重载寄存器低位	位域	CCTREL							
		类型	rw							
C3 <sub>H</sub>	<b>T2CCU_CCTRELH</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较定时器重载寄存器高位	位域	CCTREL							
		类型	rw							
C4 <sub>H</sub>	<b>T2CCU_CCTL</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较定时器寄存器低位	位域	CCT							
		类型	rwh							
C5 <sub>H</sub>	<b>T2CCU_CCTH</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较定时器寄存器高位	位域	CCT							
		类型	rwh							
C6 <sub>H</sub>	<b>T2CCU_CCTCON</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较定时器控制寄存器	位域	CCTPRE				CCTO VF	CCTO VEN	TIMSY N	CCTS T
		类型	rw				rwh	rw	rw	rw

表 3-10 T2CCU 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 0, 页 2										
C0 <sub>H</sub>	<b>T2CCU_COSHDW</b> 复位值: 00 <sub>H</sub> T2CCU 比较映射寄存器	位域	ENSH DW	TXOV	COOU T5	COOU T4	COOU T3	COOU T2	COOU T1	COOU T0
		类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
C1 <sub>H</sub>	<b>T2CCU_CC0L</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较寄存器 0 低位	位域	CCVALL							
		类型	rwh							
C2 <sub>H</sub>	<b>T2CCU_CC0H</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较寄存器 0 高位	位域	CCVALH							
		类型	rwh							
C3 <sub>H</sub>	<b>T2CCU_CC1L</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较寄存器 1 低位	位域	CCVALL							
		类型	rwh							
C4 <sub>H</sub>	<b>T2CCU_CC1H</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较寄存器 1 高位	位域	CCVALH							
		类型	rwh							
C5 <sub>H</sub>	<b>T2CCU_CC2L</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较寄存器 2 低位	位域	CCVALL							
		类型	rwh							
C6 <sub>H</sub>	<b>T2CCU_CC2H</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较寄存器 2 高位	位域	CCVALH							
		类型	rwh							
RMAP = 0, 页 3										
C0 <sub>H</sub>	<b>T2CCU_COC0N</b> 复位值: 00 <sub>H</sub> T2CCU 比较控制寄存器	位域	CCM5	CCM4	CM5F	CM4F	POLB	POLA	COMOD	
		类型	rw	rw	rwh	rwh	rw	rw	rw	
C1 <sub>H</sub>	<b>T2CCU_CC3L</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较寄存器 3 低位	位域	CCVALL							
		类型	rwh							
C2 <sub>H</sub>	<b>T2CCU_CC3H</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较寄存器 3 高位	位域	CCVALH							
		类型	rwh							
C3 <sub>H</sub>	<b>T2CCU_CC4L</b> 复位值: 00 <sub>H</sub> 2CCU 捕获 / 比较寄存器 4 低位	位域	CCVALL							
		类型	rwh							
C4 <sub>H</sub>	<b>T2CCU_CC4H</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较寄存器 4 高位	位域	CCVALH							
		类型	rwh							
C5 <sub>H</sub>	<b>T2CCU_CC5L</b> 复位值: 00 <sub>H</sub> 2CCU 捕获 / 比较寄存器 5 低位	位域	CCVALL							
		类型	rwh							
C6 <sub>H</sub>	<b>T2CCU_CC5H</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较寄存器 5 高位	位域	CCVALH							
		类型	rwh							
RMAP = 0, 页 4										
C2 <sub>H</sub>	<b>T2CCU_CCTDTCL</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较定时器死区时间控制寄存器低位	位域	DTM							
		类型	rw							
C3 <sub>H</sub>	<b>T2CCU_CCTDTCH</b> 复位值: 00 <sub>H</sub> T2CCU 捕获 / 比较定时器死区时间控制寄存器高位	位域	DTRE S	DTR2	DTR1	DTR0	DTLEV	DTE2	DTE1	DTE0
		类型	rwh	rh	rh	rh	rw	rw	rw	rw

### 3.4.5.9 定时器 T21 寄存器

定时器 T21 SFR 从映射存储器区访问（RMAP = 1）。

**表 3-11 T21 寄存器概览**

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 1										
C0 <sub>H</sub>	<b>T21_T2CON</b> 复位值: 00 <sub>H</sub> 定时器 T2 控制寄存器	位域	TF2	EXF2	0		EXEN 2	TR2	C/ $\overline{T}$ 2	CP/ RL2
		类型	rwh	rwh	r		rw	rwh	rw	rw
C1 <sub>H</sub>	<b>T21_T2MOD</b> 复位值: 00 <sub>H</sub> 定时器 T2 模式寄存器	位域	T2RE GS	T2RH EN	EDGE SEL	PREN	T2PRE			DCEN
		类型	rw	rw	rw	rw	rw	rw	rw	rw
C2 <sub>H</sub>	<b>T21_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>T21_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>T21_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>T21_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>T21_T2CON1</b> 复位值: 03 <sub>H</sub> 定时器 T2 控制寄存器 1	位域	0						TF2EN	EXF2E N
		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 3-12 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 复位值: 00 <sub>H</sub> CCU6 分页寄存器	位域	OP		STNR		0	PAGE		
		类型	w		w		r	rwh		
RMAP = 0, 页 0										
9A <sub>H</sub>	CCU6_CC63SRL 复位值: 00 <sub>H</sub> 通道 CC63 捕获 / 比较 映射寄存器, 低位字节	位域	CC63SL							
		类型	rw							
9B <sub>H</sub>	CCU6_CC63SRH 复位值: 00 <sub>H</sub> 通道 CC63 捕获 / 比较 映射寄存器, 高位字节	位域	CC63SH							
		类型	rw							
9C <sub>H</sub>	CCU6_TCTR4L 复位值: 00 <sub>H</sub> 定时器控制寄存器 4, 低位字节	位域	T12 STD	T12 STR	0		DT RES	T12 RES	T12R S	T12R R
		类型	w	w	r		w	w	w	w
9D <sub>H</sub>	CCU6_TCTR4H 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>	CCU6_MCMOUTSL 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	MCMPS						
		Type	w	r	rw					
9F <sub>H</sub>	CCU6_MCMOUTSH 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>	CCU6_ISRL 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>	CCU6_ISRH 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>	CCU6_CMPMODIFL 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>	CCU6_CMPMODIFH 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>	CCU6_CC60SRL 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>	CCU6_CC60SRH 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							



表 3-12 CCU6 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
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, 页 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							
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_T12DTCH</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							

表 3-12 CCU6 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
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, 页 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
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 / 比较中断状态位置寄存器, 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 / 比较中断状态位置寄存器, 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					

表 3-12 CCU6 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
FD <sub>H</sub>	CCU6_MODCTR <sub>H</sub> 复位值: 00 <sub>H</sub> 调制控制寄存器, 高位字节	位域	ECT1 30	0	T13MODEN					
		类型	rw	r	rw					
FE <sub>H</sub>	CCU6_TRPCTR <sub>L</sub> 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>	CCU6_TRPCTR <sub>H</sub> 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, 页 3										
9A <sub>H</sub>	CCU6_MCMOUT <sub>L</sub> 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>	CCU6_MCMOUT <sub>H</sub> 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>	CCU6_IS <sub>L</sub> 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
9D <sub>H</sub>	CCU6_IS <sub>H</sub> 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_PISEL0 <sub>L</sub> 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_PISEL0 <sub>H</sub> 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_T12 <sub>L</sub> 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_T12 <sub>H</sub> 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_T13 <sub>L</sub> 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_T13 <sub>H</sub> 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_CMPSTAT <sub>L</sub> 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_CMPSTAT <sub>H</sub> 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 UART1 寄存器

UART1 SFR 从映射存储器区访问（RMAP = 1）。

表 3-13 UART1 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 1										
C8 <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	rwh
C9 <sub>H</sub>	SBUF 串行数据缓冲寄存器 复位值：00 <sub>H</sub>	位域	VAL							
		类型	rwh							
CA <sub>H</sub>	BCON 波特率控制寄存器 复位值：00 <sub>H</sub>	位域	0				BRPRE			R
		类型	r				rw			rw
CB <sub>H</sub>	BG 波特率定时器 / 重载寄存器 复位值：00 <sub>H</sub>	位域	BR_VALUE							
		类型	rwh							
CC <sub>H</sub>	FDCON 分数分频器控制寄存器 复位值：00 <sub>H</sub>	位域	0					NDOV	FDM	FDEN
		类型	r					rwh	rw	rw
CD <sub>H</sub>	FDSTEP 分数分频器重载寄存器 复位值：00 <sub>H</sub>	位域	STEP							
		类型	rw							
CE <sub>H</sub>	FDRES 分数分频器结果寄存器 复位值：00 <sub>H</sub>	位域	RESULT							
		类型	rh							
CF <sub>H</sub>	SCON1 串行通道控制寄存器 1 复位值：07 <sub>H</sub>	位域	0					NDOVEN	TIEN	RIEN
		类型	r					rw	rw	rw

### 3.4.5.12 SSC 寄存器

SSC SFR 从标准存储器区访问（RMAP = 0）。

表 3-14 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	rwh	rwh	rwh	rwh

表 3-14 SSC 寄存器概览 (cont'd)

地址	寄存器名	位	7	6	5	4	3	2	1	0
AC <sub>H</sub>	<b>SSC_TBL</b> 复位值: 00 <sub>H</sub> 发送缓冲寄存器, 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> 接收缓冲寄存器, 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							

### 3.4.5.13 MultiCAN 寄存器

MultiCAN SFR 从标准存储器区访问 (RMAP = 0)。

表 3-15 CAN 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 0										
D8 <sub>H</sub>	<b>ADCON</b> 复位值: 00 <sub>H</sub> CAN 地址 / 数据控制寄存器	位域	V3	V2	V1	V0	AUAD		BSY	RWEN
		类型	rw	rw	rw	rw	rw		rh	rw
D9 <sub>H</sub>	<b>ADL</b> 复位值: 00 <sub>H</sub> CAN 地址寄存器, 低位字节	位域	CA9	CA8	CA7	CA6	CA5	CA4	CA3	CA2
		类型	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh
DA <sub>H</sub>	<b>ADH</b> 复位值: 00 <sub>H</sub> CAN 地址寄存器, 高位字节	位域	0				CA13	CA12	CA11	CA10
		类型	r				rwh	rwh	rwh	rwh
DB <sub>H</sub>	<b>DATA0</b> 复位值: 00 <sub>H</sub> CAN 数据寄存器 0	位域	CD							
		类型	rwh							
DC <sub>H</sub>	<b>DATA1</b> 复位值: 00 <sub>H</sub> CAN 数据寄存器 1	位域	CD							
		类型	rwh							
DD <sub>H</sub>	<b>DATA2</b> 复位值: 00 <sub>H</sub> CAN 数据寄存器 2	位域	CD							
		类型	rwh							
DE <sub>H</sub>	<b>DATA3</b> 复位值: 00 <sub>H</sub> CAN 数据寄存器 3	位域	CD							
		类型	rwh							

### 3.4.5.14 OCDS 寄存器

OCDS SFR 从映射存储区访问（RMAP = 1）。

表 3-16 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: 8U <sub>H</sub> 监控模式控制寄存器 2	位域	STMO DE	EXBC	DSUS P	MBCO N	ALTDI	MMEP	MMOD E	JENA
		类型	rw	rw	rw	rwh	rw	rwh	rh	rh
EA <sub>H</sub>	<b>MEXTCR</b> 复位值: 0U <sub>H</sub> 存储器扩展控制寄存器	位域	0				BANKBPx			
		类型	r				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							
EC <sub>H</sub>	<b>MMWR2</b> 复位值: 00 <sub>H</sub> 监控工作寄存器 2	位域	MMWR2							
		类型	rw							
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	0	MSTE P	MRAM S_P	MRAM S	TRF	RRF
		类型	w	rwh	r	rw	w	rwh	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	MBCA M	MBCIN	EXBF	SWBF	HWB3 F	HWB2 F	HWB1 F	HWB0 F
		类型	rw	rwh	rwh	rwh	rwh	rwh	rwh	rwh
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	MMUI E_P	MMUI E	RRIE_ P	RRIE
		类型	rwh	rwh	rwh	rh	w	rw	w	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							
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							

### 3.4.5.15 Flash 寄存器

Flash SFR 从映射存储区访问（RMAP = 1）。

表 3-17 Flash 寄存器概览

地址	寄存器名	位	7	6	5	4	3	2	1	0
RMAP = 1										
D1 <sub>H</sub>	<b>FCON</b> P-Flash 控制寄存器 复位值: 10 <sub>H</sub>	位域	0	FBSY	YE	1	NVST R	MAS1	ERAS E	PROG
		类型	r	rh	rwh	r	rw	rw	rw	rw
D2 <sub>H</sub>	<b>EECON</b> D-Flash 控制寄存器 复位值: 10 <sub>H</sub>	位域	0	EEBS Y	YE	1	NVST R	MAS1	ERAS E	PROG
		类型	r	rh	rwh	r	rw	rw	rw	rw
D3 <sub>H</sub>	<b>FCS</b> Flash 控制和状态寄存器 复位值: 80 <sub>H</sub>	位域	1	SBEIE	FTEN	0	EEDE RR	EESE RR	FDER R	FSER R
		类型	r	rw	rwh	r	rwh	rwh	rwh	rwh
D4 <sub>H</sub>	<b>FEAL</b> Flash 错误地址寄存器, 低位字节 复位值: 00 <sub>H</sub>	位域	ECCEADDR							
		类型	rh							
D5 <sub>H</sub>	<b>FEAH</b> Flash 错误地址寄存器, 高位字节 复位值: 00 <sub>H</sub>	位域	ECCEADDR							
		类型	rh							
D6 <sub>H</sub>	<b>FTVAL</b> Flash 定时器值寄存器 复位值: 78 <sub>H</sub>	位域	MODE	OFVAL						
		类型	rw	rw						
DD <sub>H</sub>	<b>FCS1</b> Flash 控制和状态寄存器 1 复位值: 00 <sub>H</sub>	位域	0							EEAB ORT
		类型	r							rwh

## 3.5 Boot ROM 工作模式

复位后 CPU 开始工作, 始终先执行存储器映射结构 0 中的 Boot ROM 代码。在存储器映射结构 0 中, Boot ROM 占据程序存储器的地址段 0'0000<sub>H</sub>-0'1FFF<sub>H</sub>、0'C000<sub>H</sub>-0'DFFF<sub>H</sub> 和 2'C000<sub>H</sub>-2'DFFF<sub>H</sub>, 此时程序存储器的其它地址空间被禁用。Boot ROM 启动进程在切换到存储器结构 1 之前, 首先跳转到地址 2'C00X<sub>H</sub>上, 如图 3-7 和图 3-8 所示。因此, 原先占据地址段 0'0000<sub>H</sub>-0'1FFF<sub>H</sub>、0'C000<sub>H</sub>-0'DFFF<sub>H</sub> 和 2'C000<sub>H</sub>-2'DFFF<sub>H</sub> 的 Boot ROM (根据 Flash 存储器的大小) 将只被映射到地址段 2'C000<sub>H</sub>-2'DFFF<sub>H</sub> 或 0'C000<sub>H</sub>-0'DFFF<sub>H</sub>, 此外, 程序存储器的其它地址空间被使能 (XRAM、P-Flash 和 D-Flash)。存储器映射结构切换之后, 其余的 Boot ROM 启动进程将从 2'C00X<sub>H</sub>/0'C00X<sub>H</sub>继续执行。这包括检查引脚 MBC、TMS 和 P0.0 的锁存值, 以进入选定的 Boot ROM 工作模式。Boot ROM 工作模式的选择请参见章节 7.2.3。本手册中给出的 XC878 存储器结构是存储器映射结构切换之后的存储器结构图, 即存储器映射结构 1, 在该映射结构下进入 Boot ROM 的不同工作模式。

注: TM 引脚需要一个外部下拉器件以确保正确进入 Boot ROM 工作模式。

## 存储器结构

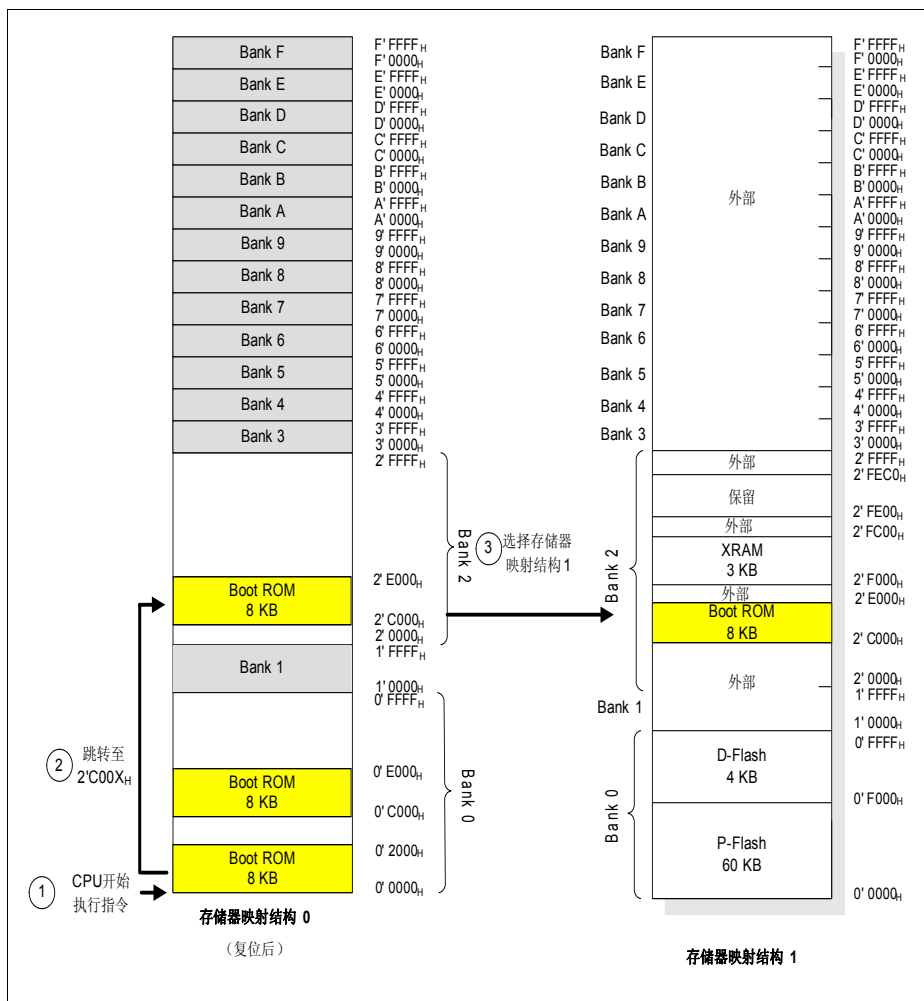


图 3-7 内嵌 64K Flash 的存储器映射结构切换



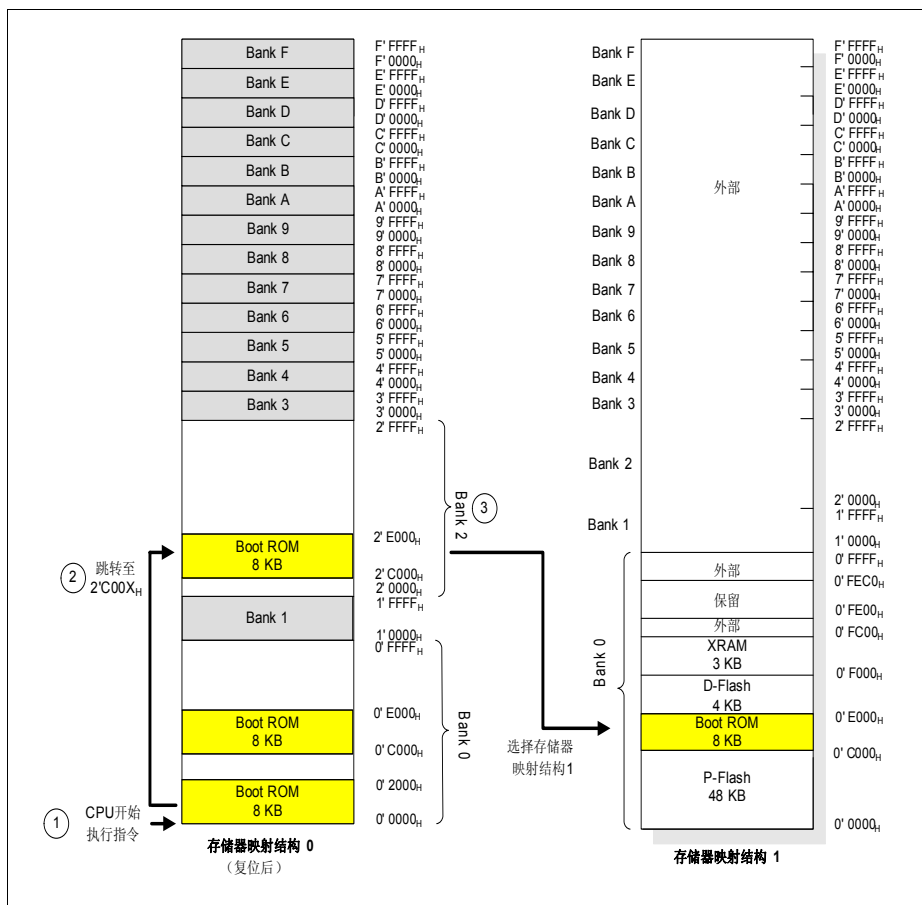


图 3-8 内嵌 52K Flash 的存储器映射结构切换

### 3.5.1 用户模式

如果 (MBC, TMS, P0.0) = (1, 0, x)，Boot ROM 将跳转到程序存储器的地址 0000<sub>H</sub> 处，执行 Flash 存储器中的用户程序。这是 XC878 的正常模式（片上振荡器和 PLL 不旁路）。

但是，如果程序存储器地址 0'0000<sub>H</sub> 中的值为 FF<sub>H</sub>，表明 Flash 存储器还未经用户程序编程，此时将进入引导程序加载（BSL）模式进行 Flash 编程。

注：用户应该避免在程序存储器地址 0000<sub>H</sub> 中写入 FF<sub>H</sub>，从而避免无意进入 BSL 模式。

### 3.5.2 引导程序加载器模式

如果  $(MBC, TMS, P0.0) = (0, 0, x)$ ，将执行存放在 Boot ROM 中的引导程序加载器程序，可以对 XRAM 和 Flash 存储器（若片内包括该存储器）编程、擦除和执行。不同的 BSL 工作模式请参见第 19 章。

### 3.5.3 OCDS 模式

如果  $(MBC, TMS, P0.0) = (0, 1, 0)$ ，将进入 OCDS 模式进行程序调试。首先初始化 OCDS 硬件，接下来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。

### 3.5.4 用户 JTAG 模式

如果  $(MBC, TMS, P0.0) = (1, 1, x)$ ，Boot ROM 将跳转到程序存储器的地址  $0000_H$  处，执行 Flash 存储器中的用户代码，和章节 3.5.1 所描述的正常工作模式相似。该模式下主 JTAG 端口可被自动配置以允许热插拔。

## 4 闪存存储器

XC878 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。

### 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
- 支持擦除中止操作
- D-Flash 和 P-Flash 的最小编程宽度分别为 1 个和 2 个字节
- 最小擦除宽度为 1 页
- 每次读取 1 个字节
- 1 个 CCLK 周期的读取时间
- Flash 出厂时为擦除状态（读返回全 1）

## 4.1 Flash 存储器映射

XC878 产品系列中的 Flash 器件可包含 64KB 或 52KB 嵌入式 Flash 存储器。每种 Flash 器件由程序 Flash (P-Flash) 和数据 Flash (D-Flash) 组成。64KB Flash 器件由 60KB P-Flash 和 4KB D-Flash 构成；52KB Flash 器件由 48KB P-Flash 和 4KB D-Flash 构成。这两种器件中 P-Flash 和 D-Flash 的地址映射归纳见表 4-1 所示。

表 4-1 Flash 存储器映射

器件	描述	大小	地址范围
64-Kbyte Flash	P-Flash	60 KBytes	0'0000 <sub>H</sub> - 0'FFFF <sub>H</sub>
	D-Flash	4 KBytes	0'F000 <sub>H</sub> - 0'FFFF <sub>H</sub>
52-Kbyte Flash	P-Flash	48 KBytes	0'0000 <sub>H</sub> - 0'BFFF <sub>H</sub>
	D-Flash	4 KBytes	0'E000 <sub>H</sub> - 0'FFFF <sub>H</sub>

P-Flash 占据程序存储器中以 0000<sub>H</sub> 为起始地址的地址段，复位和中断向量位于该地址指向的存储器单元。4KB D-Flash 映射到程序存储器的不同地址段（这和 XC878 中 Flash 器件的类型有关）。

## 4.2 Flash 存储块分页

XC878 器件中包含两种 Flash 存储块，即程序 Flash (P-Flash) 和数据 Flash (D-Flash) 存储块。P-Flash 由 120 页构成，每页包含 8 条字线，每条字线包含 64 个字节。D-Flash 由 64 页构成，每页包含 2 条字线，每条字线包含 32 个字节。这两种 Flash 均可用于存储代码和数据。标记 "D" 并不表示 D-Flash 被映射到数据存储区，也不表示它只能用来存储数据。使用该标记旨在区分每种 Flash 的不同页宽和字线宽度。

每种 Flash 存储块内部为分页结构，从而具有灵活的擦除能力。最小擦除宽度始终为完整的一页。D-Flash 存储块内部被划分的更小，从而具有扩展的擦除和重新编程能力。

举例来说，用户程序可为 Flash 存储块中的每一页建立缓存机制。每组数据双份复制保存在大小相同、分开的页面中，这样可确保在真正的数据组被破坏或擦除时可使用备份数据。

此外，用户还可实现 EEPROM 仿真，把 D-Flash 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。当达到页面顶部时，所有真正的数据（代表 EEPROM 的数据）被复制到下一页的底部，上一页被擦除。用这种多方复制的循环机制来仿真 EEPROM，大大提高了 Flash 存储器的耐受能力。为了加速数据查询，RAM 可用于存放指向有效数据组的指针。

## 4.3 工作模式

XC878 提供五种 Flash 工作模式：STANDBY、读取、编程、擦除和整体擦除，适用于 P-Flash 和 D-Flash。读操作由硬件控制，编程、擦除和整体擦除操作均由软件控制。通常，Flash 的工作模式由 BSL 和 Flash 编程 / 擦除子程序控制（见章节 4.7）。不过，用

## 闪存存储器

户也可选择编写子程序来执行这些 Flash 操作（见 [章节 4.3.1- 章节 4.3.5](#)）。Flash 操作需要不同的时间间隔，可通过 Flash 定时器产生（见 [章节 4.4](#)）。

在擦除或编程过程中，不允许对同一 Flash 存储段进行读操作。若用户人为中断编程或擦除周期，数据可能无法被正确编程或擦除。可通过 [章节 4.3.5](#) 所描述的程序退出 D-Flash 编程和擦除操作以确保 Flash 的正确操作。

当设置  $PMCON0.PD = 1$  以进入系统掉电模式，在系统进入掉电状态之前，Flash 由硬件控制自动进入待命状态。

### 4.3.1 读操作

读操作主要由硬件控制。从总线的角度来看，从 Flash 中读数据和从程序存储器中取指相似。一旦来自 CPU 的读信号有效，Flash 将进入读模式，数据将在下一个时钟周期送至 CPU，不需等待状态。

### 4.3.2 编程操作

每个编程周期将数据写入 Flash 阵列。通过指令 "MOVC @(DPTR++),A" 执行写操作。数据首先存放在数据缓存中，由 ECC 模块编码后写入 Flash 阵列。在 XC878 中，P-Flash 的编程宽度分别为 2 字节的整数倍；D-Flash 的编程宽度为 1 个字节。

在一个编程周期内可按如下步骤进行 Flash 编程：

1. 置位  $FCON.PROG$  (P-Flash) 或  $EECON.PROG$  (P-Flash) 以指示一个编程周期开始。
2. 执行 MOVC 指令将一个伪数据写入待访问地址所处字线的任意地址单元中。
3. 延迟最少 5  $\mu s$  ( $T_{vns}$ )。
4. 置位  $FCON/EECON.NVSTR$  以开启电荷泵驱动高电压。
5. 延迟最少 10  $\mu s$  ( $T_{pgs}$ )。
6. 执行 MOVC 指令访问 Flash。 $FCON/EECON.YE$  在下一个时钟周期由硬件置位 ( $YE$  需要保持 40 ns)。
7. 延迟最少 20  $\mu s$ 、但不超过 40  $\mu s$  ( $T_{prog}$ )。
8. 分别清除  $FCON/EECON.YE$  和  $FCS.FTEN$ 。
9. 重复步骤 6-8 继续执行同一行上的数据编程。
10. 清除  $FCON/EECON.PROG$ 。
11. 延迟最少 5  $\mu s$  ( $T_{nvh}$ )
12. 清除  $FCON/EECON.NVSTR$ 。
13. 延迟最少 1  $\mu s$  ( $T_{rcv}$ )。

注： 步骤 1-12 的编程序列必须在 4 ms 内完成。

### 4.3.3 擦除操作

每个擦除周期将 Flash 页面擦除到 "1"。

可按如下步骤擦除 Flash 页面：

1. 置位 FCON/EECON.ERASE 并清除 FCON/EECON.MAS1 以开始新的页面擦除。
2. 执行 MOV<sub>C</sub> 指令将一个伪数据写入待擦除页面中的任意地址单元。
3. 延迟最少 5  $\mu$ s ( $T_{vns}$ )。
4. 置位 FCON/EECON.NVSTR 以开启电荷泵驱动高电压。
5. 延迟最少 20 ms ( $T_{erase}$ )。
6. 清除 FCON/EECON.ERASE。
7. 延迟最少 5  $\mu$ s ( $T_{nvh}$ )。
8. 清除 FCON/EECON.NVSTR。
9. 延迟最少 1  $\mu$ s ( $T_{rcv}$ )。

注：若 P-Flash 中的一页刚被擦除后立即对其中的某个单元进行读操作，寄存器 FCS 中的 FDERR 将被置位，ECC 错误可导致产生中断。为了防止这种情况的发生，应通过位 NMICON.NMIECC 禁止产生 ECC 中断。

#### 4.3.4 整体擦除操作

整体擦除操作会把 P-Flash 或 D-Flash 存储块中的所有存储单元擦除到 "1"。若 P-Flash 被整体擦除之后立即进行读操作，ECC 错误可导致产生中断。为了防止这种情况的发生，应通过位 NMICON.NMIECC 禁止产生 ECC 中断。

可按如下步骤进行整体擦除：

1. 置位 FCON/EECON.ERASE 和 FCON/EECON.MAS1 以开始新的整体擦除周期。
2. 执行 MOV<sub>C</sub> 指令将一个伪数据写入待擦除页面中的任意地址单元。
3. 延迟最少 5  $\mu$ s ( $T_{vns}$ )。
4. 置位 FCON/EECON.NVSTR 以开启电荷泵驱动高电压。
5. 延迟最少 200 ms ( $T_{me}$ )。
6. 清除 FCON/EECON.ERASE。
7. 延迟最少 100  $\mu$ s ( $T_{nvhl}$ )。
8. 清除 FCON/EECON.NVSTR 和 FCON/EECON.MAS1。
9. 延迟最少 1  $\mu$ s ( $T_{rcv}$ )。

#### 4.3.5 中止操作

对 D-Flash 的编程和擦除（包括整体擦除）操作允许由软件中止。为了确保正确操作 Flash，用户软件必须遵循图 4-1 所示的中止流程。中止操作只适用于 D-Flash。一旦寄存器 FCS1 中的位 EEABORT 被置位，读取、编程和擦除操作被中止。

为了确保通过 Flash 定时器产生中止操作所需的正确的时序间隔，软件需先清除 FTEN、之后重新置位 FTEN，这样将把 Flash 定时器中的两个计数器复位到零。

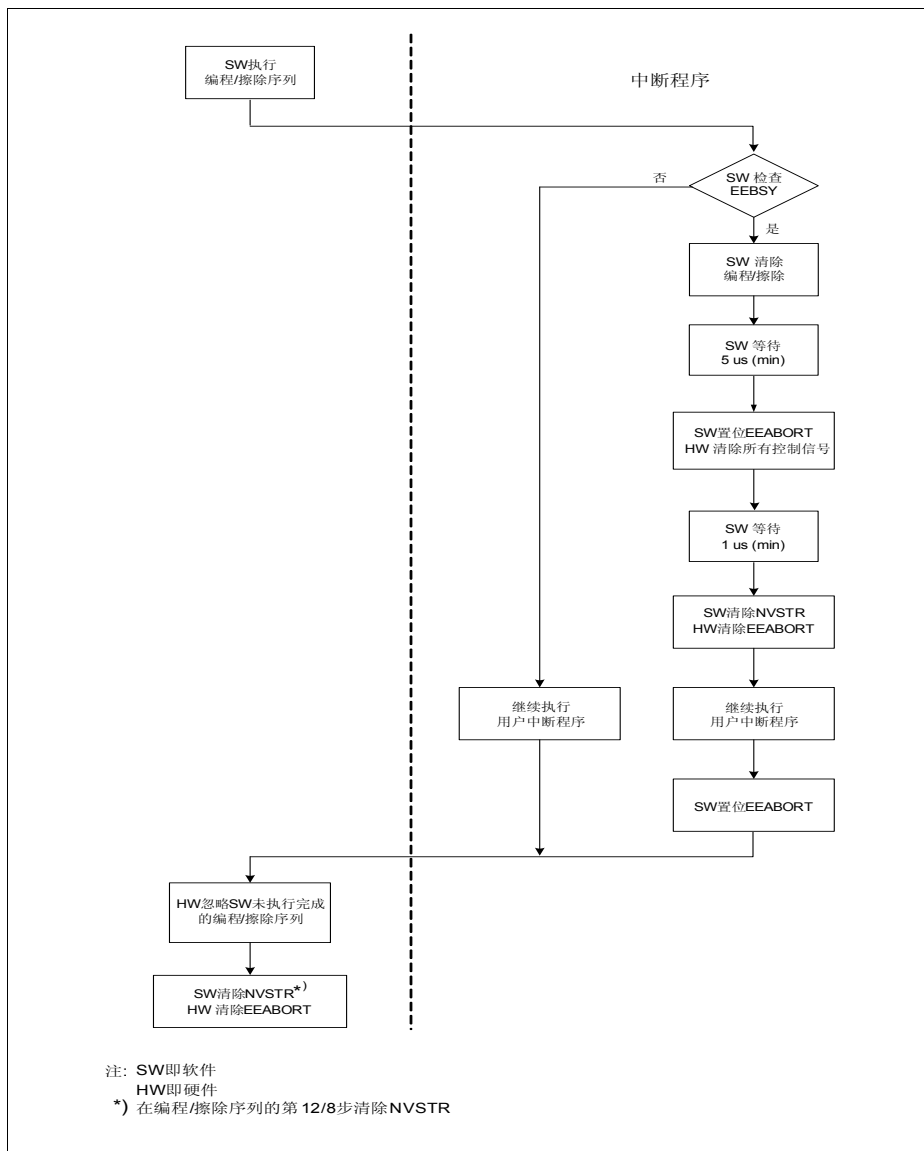


图 4-1 D-Flash 编程 / 擦除中止流程

## 4.4 Flash 定时器

Flash 定时器用于产生编程和擦除所需的时间间隔（如编程建立时间、擦除时间等）。

Flash 定时器有两种工作模式：编程定时器模式和擦除定时器模式，由位 `FTVAL.MODE` 控制模式选择。在编程定时器模式下（`FTVAL.MODE = 0`），使用一个 7 位计数器递增计数，计数到由 `FTVAL.OFVAL` 指定的溢出值后（相应的溢出标志 `FCS.FTOV` 置位），计数器复位到零。若 Flash 定时器未被禁用，则继续重新计数。

在擦除定时器模式下（`FTVAL.MODE = 1`），附加使用一个 12 位计数器。使能 Flash 定时器工作时，该计数器递增计数。每次计数溢出后、在复位到零并继续重新计数之前产生一个信号控制 7 位计数器加 1。只有当 7 位计数器计数到指定的溢出值后才置位定时器溢出信号和标志。

通过置位 / 清除 `FCS.FTEN` 使能 / 禁止 Flash 定时器工作。一旦 Flash 定时器被禁用，7 位和 12 位计数器均被复位并保持为 0。

Flash 定时器的波形示例见 图 4-2。

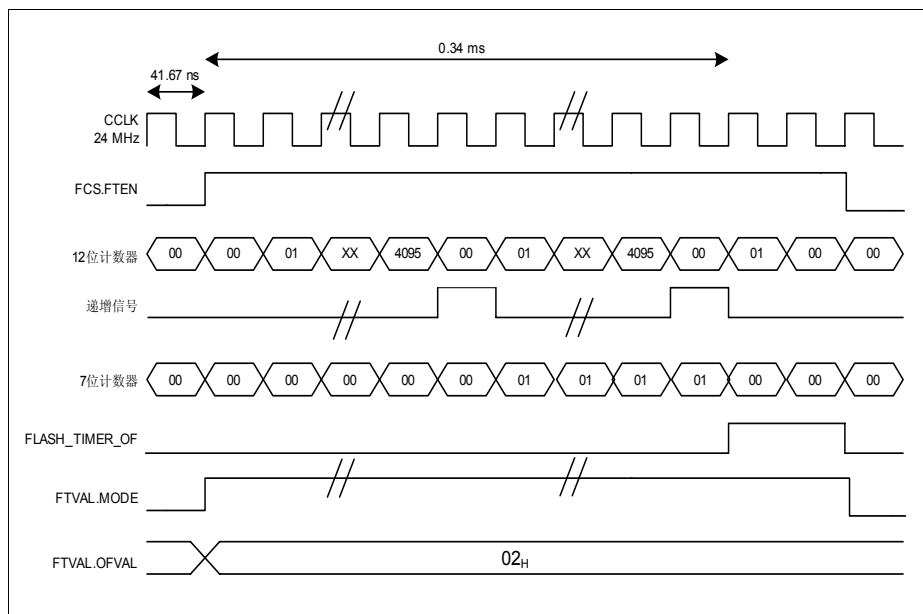


图 4-2 Flash 定时器时序图



## 4.5 检错和纠错

CPU 送出的 8 位数据在存入 Flash 之前采用纠错码 (ECC) 编码。ECC 由修改的汉明码产生, 存放于数据的 MSB 之前。从 Flash 中读取数据时, 数据首先被译码从而实现了动态检错和纠错。

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

可通过寄存器 FCS 中的一位错误标志和两位错误标志区分可纠正的 1 位错误 (结果有效) 和不可纠正的 2 位错误 (结果无效)。

这两种情况均可产生 ECC 非可屏蔽中断 (NMI) 事件; 寄存器 NMISR 中的位 FNMIECC 被置位。若 NMICON.NMIECC 被使能, 则触发一个 NMI 中断。若要产生一位错误中断, 还必须置位寄存器 FCS 中的使能位 SBEIE。

ECC 出错处的 16 位地址被保存在系统控制寄存器 FEAL 和 FEAH 中 (寄存器描述见[章节 4.9](#)), 中断服务程序可访问这两个寄存器以确定出错的位置。

## 4.6 在系统编程

通过位于 **Boot ROM** 中的引导程序加载（**BSL**）实现了 **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 **MBC** 和 **TMS** 引脚的锁存值均为 **0**，选择进入 **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 **P-Flash** 和 **D-Flash** 存储块。

可用的工作模式包括：

- 将用户程序从主机传送至 **Flash** 中
- 执行 **Flash** 中的用户程序
- 进行 **P-Flash** 或 **D-Flash** 页擦除
- 进行 **P-Flash** 和 **D-Flash** 整体擦除

## 4.7 在应用编程

在一些实际应用中，在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、擦除或整体擦除。在开始执行编程、擦除或整体擦除操作之前，Flash 子程序会先执行一些检查、并进行初始化。只有当这些 Flash 操作完成之后才能继续执行用户程序。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。

*注：正在执行某条字线上的 Flash 用户程序时，该字线不能被编程、擦除或整体擦除。例如，存放在 P-Flash 页 1 字线 0 中的用户程序可编程或擦除该页中字线 0 之外的其它字线。*

### 4.7.1 Flash 编程

每次调用 Flash 编程子程序，可将 2 - 64 个字节编程到 P-Flash 的字线（WL）上，将 1 - 32 个字节编程到 D-Flash 的字线（WL）上。

调用该子程序前，用户必须确保待编程的内容按升序依次存放在 XRAM 中，由 DPTR1 指定存放的起始地址。还需由 R1 指示编程到一条字线中的字节数。此外，输入 DPTR0 必须包含有效的 Flash 偶地址（受保护 Flash 存储块的地址视为无效）。否则，PSW.CY 将被置位，不执行编程操作。在每个编程周期内，不允许跨字线编程。举例来说，若要将 4 个字节编程到 P-Flash 中，起始地址不允许为 003EH。

若在调用子程序之前有效输入（见表 4-2）已可用，微控制器将首先执行初始化序列（包括在 IRAM 缓存中设定 Flash 定时器），之后执行编程操作（详见章节 4.3.2）。只有当 Flash 编程子程序结束之后才能继续执行用户程序。

表 4-2      Flash 编程子程序

子程序	DFF6 <sub>H</sub> :FLASH_PROGRAM
输入	DPTR0: Flash 编程起始地址
	DPTR1: XRAM 中存放待编程 Flash 数据的起始地址
	R1: 编程字节数
	EA: 0
输出	PSW.CY: C = 0 将 XRAM 中 R1 个字节编程到 P-Flash 或 D-Flash <sup>1)</sup> 的正确单元中。 C = 1 以下原因可导致无法对 Flash 编程: 1. DPTR0 包含无效的 P-Flash 或 D-Flash 地址。 2. DPTR0 包含 P-Flash 奇地址。 3. R1 为 0 或超过 Flash 字线长度。 4. R1 在 P-Flash 编程过程中为奇数。 5. 待编程的 Flash 存储块跨字线。
所需堆栈大小	9
使用资源	ACC, B, C, R0, R3, R4, R5, IRAM(37 <sub>H</sub> - 3E <sub>H</sub> )

<sup>1)</sup> 一旦调用编程子程序，P-Flash 或 D-Flash 的编程地址由该子程序自动确定。

4.7.2      Flash 擦除

每次调用 Flash 擦除子程序，可进行页擦除（擦除 P-Flash 或 D-Flash 存储块中的某一页）或整体擦除（擦除 P-Flash 或 D-Flash 存储块）。调用该子程序前，ACC 寄存器中必须包含擦除类型，输入 DPTR0 必须包含待擦除页面或存储块中有效的 Flash 地址。同样的，不应擦除受保护 Flash 存储块。

若在调用子程序之前有效输入（见表 4-3）已可用，微控制器将首先执行初始化序列，之后执行擦除操作（详见章节 4.3.3）。只有当 Flash 擦除子程序结束之后才能继续执行用户程序。

表 4-3      Flash 擦除子程序

子程序	DFF9 <sub>H</sub> : FLASH_ERASE
输入	DPTR0: 待擦除 Flash 页面或存储块中的任意地址
	ACC: 擦除类型（0: 页擦除; 1: 整体擦除）
	EA: 0
	MISC_CON.DFLASHEN <sup>1)</sup> : 1
输出	PSW.CY: C = 0 擦除 P-Flash 或 D-Flash <sup>2)</sup> 页面或存储块 C = 1 Flash 地址无效导致无法进行擦除操作
所需堆栈大小	9
使用资源	ACC, B, C, R0, R3, R4, R5, IRAM(37 <sub>H</sub> - 3C <sub>H</sub> )

1) 当 Flash 保护模式 0 被使能，每次擦除 D-Flash 存储块之前需要置位 DFLASHEN。

2) 一旦调用擦除子程序，P-Flash 或 D-Flash 的地址单元由该子程序自动确定。

4.7.3      获取芯片信息

用户调用获取芯片信息子程序，可读出包含芯片相关信息的 4 字节数据。在 XC878 中，这 4 个字节提供芯片 ID，用于识别特定的产品型号。

表 4-4      获取芯片信息子程序

子程序	DFE1 <sub>H</sub> : GET_CHIP_INFO
输入	ACC: 00 <sub>H</sub> = 芯片 ID 其它 = 保留
	R1: IRAM 中存放 4 字节返回数据的起始地址
	MEX2.MCM: 0

表 4-4 获取芯片信息子程序

输出	PSW.CY: C = 0 输入有效 (ACC = 00 <sub>H</sub> ) 时, 按如下方式填充 IRAM 缓存: 字节 1 存入 R1 (MSB) 字节 2 存入 R1+1 字节 3 存入 R1+2 字节 4 存入 R1+3 (LSB) C = 1 输入无效 (ACC ≠ 00 <sub>H</sub> ) 导致 IRAM 缓存被清除
所需堆栈大小	5
使用资源	ACC, R1

### 4.8 寄存器映射

Flash 模块的寄存器包括：两个分别用于 P-Flash 和 D-Flash 的控制寄存器、一个状态寄存器、一个 D-Flash 控制寄存器、一个 Flash 定时器寄存器和两个存放 ECC 出错地址（完整的 16 位地址）的地址寄存器。可从映射 SFR 区访问这些寄存器。

Flash 模块的寄存器地址归纳见[表 4-5](#)。

**表 4-5 SFR 地址列表**

地址	寄存器
D1 <sub>H</sub>	FCON
D2 <sub>H</sub>	EECON
D3 <sub>H</sub>	FCS
D4 <sub>H</sub>	FEAL
D5 <sub>H</sub>	FEAL
D6 <sub>H</sub>	FTVAL
DD <sub>H</sub>	FCS1

### 4.9 寄存器描述

#### FCON

P-Flash 控制寄存器

复位值：10<sub>H</sub>

7	6	5	4	3	2	1	0
0	FBSY	YE	1	NVSTR	MAS1	ERASE	PROG
r	rh	rwh	r	rwh	rwh	rw	rw

符号	位	类型	描述
PROG	0	rw	编程位 0 不选择编程操作。 1 选择编程操作。
ERASE	1	rw	擦除位 0 不选择擦除操作。 1 选择擦除操作。
MAS1	2	rwh	整体擦除位 0 不选择整体擦除操作。 1 选择整体擦除操作。

**闪存存储器**

符号	位	类型	描述
<b>NVSTR</b>	3	rwh	<b>非易失存储位</b> 0 用于驱动高电压的电荷泵关闭。 1 用于驱动高电压的电荷泵开启。
<b>YE</b>	5	rwh	<b>Y 地址使能位</b> YE 位只能由硬件置位，但可由软件和硬件清零。 0 禁用 FYE。 1 使能 FYE。
<b>FBSY</b>	6	rh	<b>P-Flash 忙碌位</b> 0 Flash 当前未进行编程 / 擦除操作。 1 Flash 当前正进行编程 / 擦除操作。
<b>1</b>	4	r	<b>保留</b> 读取返回 1；应写入 1。
<b>0</b>	7	r	<b>保留</b> 读取返回 0；应写入 0。

**EECON**
**D-Flash 控制寄存器**
**复位值：10<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>0</b>	<b>EEBSY</b>	<b>YE</b>	<b>1</b>	<b>NVSTR</b>	<b>MAS1</b>	<b>ERASE</b>	<b>PROG</b>
r	rh	rwh	r	rwh	rwh	rw	rw

符号	位	类型	描述
<b>PROG</b>	0	rw	<b>编程位</b> 0 不选择编程操作。 1 选择编程操作。
<b>ERASE</b>	1	rw	<b>擦除位</b> 0 不选择擦除操作。 1 选择擦除操作。
<b>MAS1</b>	2	rwh	<b>整体擦除位</b> 0 不选择整体擦除操作。 1 选择整体擦除操作。
<b>NVSTR</b>	3	rwh	<b>非易失存储位</b> 0 用于驱动高电压的电荷泵关闭。 1 用于驱动高电压的电荷泵开启。



## 闪存存储器

符号	位	类型	描述
<b>YE</b>	5	rwh	<b>Y 地址使能位</b> YE 位只能由硬件置位，但可由软件和硬件清零。 0 禁用 FYE。 1 使能 FYE。
<b>EEBSY</b>	6	rh	<b>D-Flash 忙碌位</b> 0 Flash 当前未进行编程 / 擦除操作。 1 Flash 当前正进行编程 / 擦除操作。
<b>1</b>	4	r	<b>保留</b> 读取返回 1；应写入 1。
<b>0</b>	7	r	<b>保留</b> 读取返回 0；应写入 0。

## FCS

## Flash 控制和状态寄存器

复位值：80<sub>H</sub>

7	6	5	4	3	2	1	0
<b>1</b>	<b>SBEIE</b>	<b>FTEN</b>	<b>0</b>	<b>EEDERR</b>	<b>EESERR</b>	<b>FDERR</b>	<b>FSERR</b>
r	rw	rwh	r	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>FSERR</b>	0	rwh	<b>Flash 一位错误标志</b> FSERR 由硬件置位、软件复位。 0 数据中未出现一位错误。 1 数据中已出现一位错误。
<b>FDERR</b>	1	rwh	<b>Flash 两位错误标志</b> FDERR 由硬件置位、软件复位。 0 数据中未出现两位错误。 1 数据中已出现两位错误。
<b>EESERR</b>	2	rwh	<b>D-Flash 一位错误标志</b> EESERR 由硬件置位、软件复位。 0 数据中未出现一位错误。 1 数据中已出现一位错误。
<b>EEDERR</b>	3	rwh	<b>D-Flash 两位错误标志</b> EEDERR 由硬件置位、软件复位。 0 数据中未出现两位错误。 1 数据中已出现两位错误。

## 闪存存储器

符号	位	类型	描述
<b>FTEN</b>	5	rwh	<b>Flash 定时器使能位</b> FTEN 位可由软件和硬件置位、但只能由软件复位。 0 禁用 Flash 定时器。 1 使能 Flash 定时器。
<b>SBEIE</b>	6	rw	<b>一位错误中断使能</b> 0 禁止产生一位错误中断。 1 允许产生一位错误中断。
<b>0</b>	4	r	<b>保留</b> 读取返回 0；应写入 0。
<b>1</b>	7	r	<b>保留</b> 读取返回 1；应写入 1。

**FTVAL**

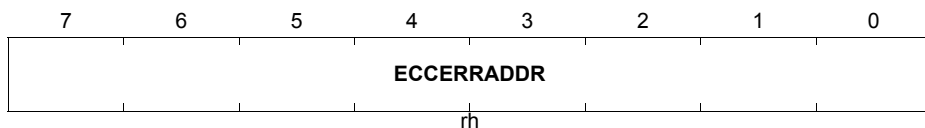
## Flash 定时器值寄存器

复位值：78<sub>H</sub>

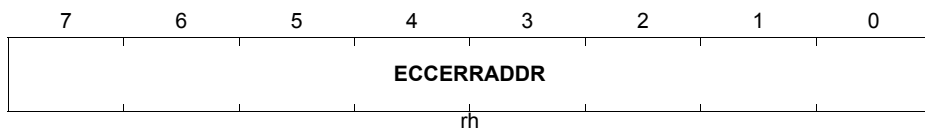
7	6	5	4	3	2	1	0
<b>MODE</b>	<b>OFVAL</b>						
rw	rw						

符号	位	类型	描述
<b>OFVAL</b>	[6:0]	rw	<b>溢出值</b> OFVAL 中存放 Flash 定时器中 7 位计数器的溢出值。 该值缺省为 78 <sub>H</sub> ，每隔大约 5 μs 产生一个上升沿信号（工作频率为 24MHz）。
<b>MODE</b>	7	rw	<b>模式</b> 0 选择编程定时器模式。 1 选择擦除定时器模式。

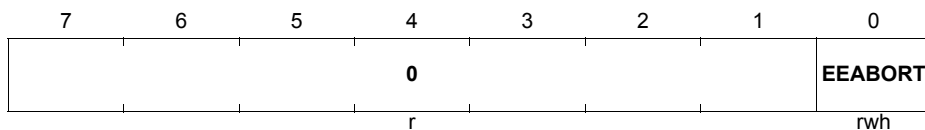
注： OFVAL 的缺省值是基于 24MHz 的内核工作频率计算得出的。若工作频率不是 24MHz，则必须计算新的溢出值并写入 OFVAL。

**FEAL**
**Flash 出错地址寄存器, 低位字节**
**复位值: 00<sub>H</sub>**


符号	位	类型	描述
<b>ECCERRADDR</b>	[7:0]	rh	<b>ECC 出错地址 [7:0]</b>

**FEAH**
**Flash 出错地址寄存器, 高位字节**
**复位值: 00<sub>H</sub>**


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

**FCS1**
**Flash 控制和状态寄存器 1**
**复位值: 00<sub>H</sub>**


符号	位	类型	描述
<b>EEABORT</b>	0	rwf	<b>D-Flash 编程 / 擦除中止</b> EEABORT 由软件置位、通过清除 EECON.NVSTR 被复位。 0     编程 / 擦除操作不被中止。 1     编程 / 擦除操作被中止。 <i>注： 一旦 EEABORT 被置位，读取、编程和擦除操作被中止。</i>
<b>0</b>	4	r	<b>保留</b> 读取返回 0；应写入 0。

## 5 中断系统

XC800 内核支持一个非可屏蔽中断（NMI）和 14 个可屏蔽中断。除了支持标准中断功能（例如，可配置的中断优先级和中断屏蔽功能）之外，XC878 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，从而增加了所支持的中断源数目，附加的状态寄存器用来检测和确定中断源。

XC878 支持 14 个中断向量，共分为 4 级中断优先级。其中，12 个中断向量分配给片上外设 定时器 T0、定时器 T1 和 UART 各分配一个专用中断向量；T2CCU、定时器 T21、UART1、MDU、SSC、MultiCAN、CORDIC、模数转换器和捕获 / 比较单元共享其余 8 个中断向量，外部中断 2 至 6 也共享其中两个中断向量；外部中断 0 和 1 各分配一个专用中断向量。

非可屏蔽中断 NMI 和常规中断类似，区别仅在于 NMI 具有最高优先级（高于其它常规中断）来处理重要的系统事件。XC878 系统中，下列五种事件可产生 NMI：

- WDT 已发出预警
- PLL 与外部晶振失锁
- Flash 定时器已溢出
- VDDP 低于预警电压（若外部电压为 5.0V，对应的 VDDP 预警电压为 4.0V）
- Flash 纠错码（ECC）出错

图 5-1 到图 5-5 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 5-6 给出 NMI 请求源总览。



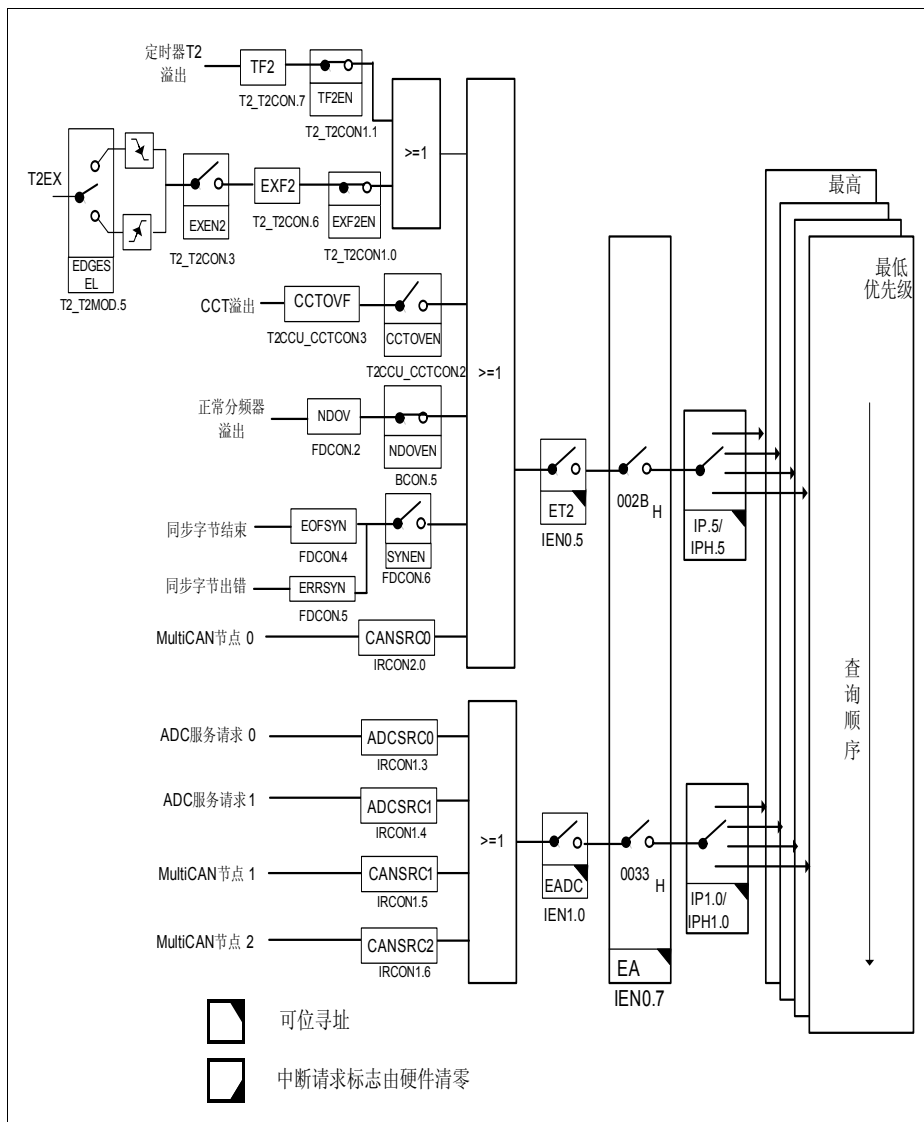
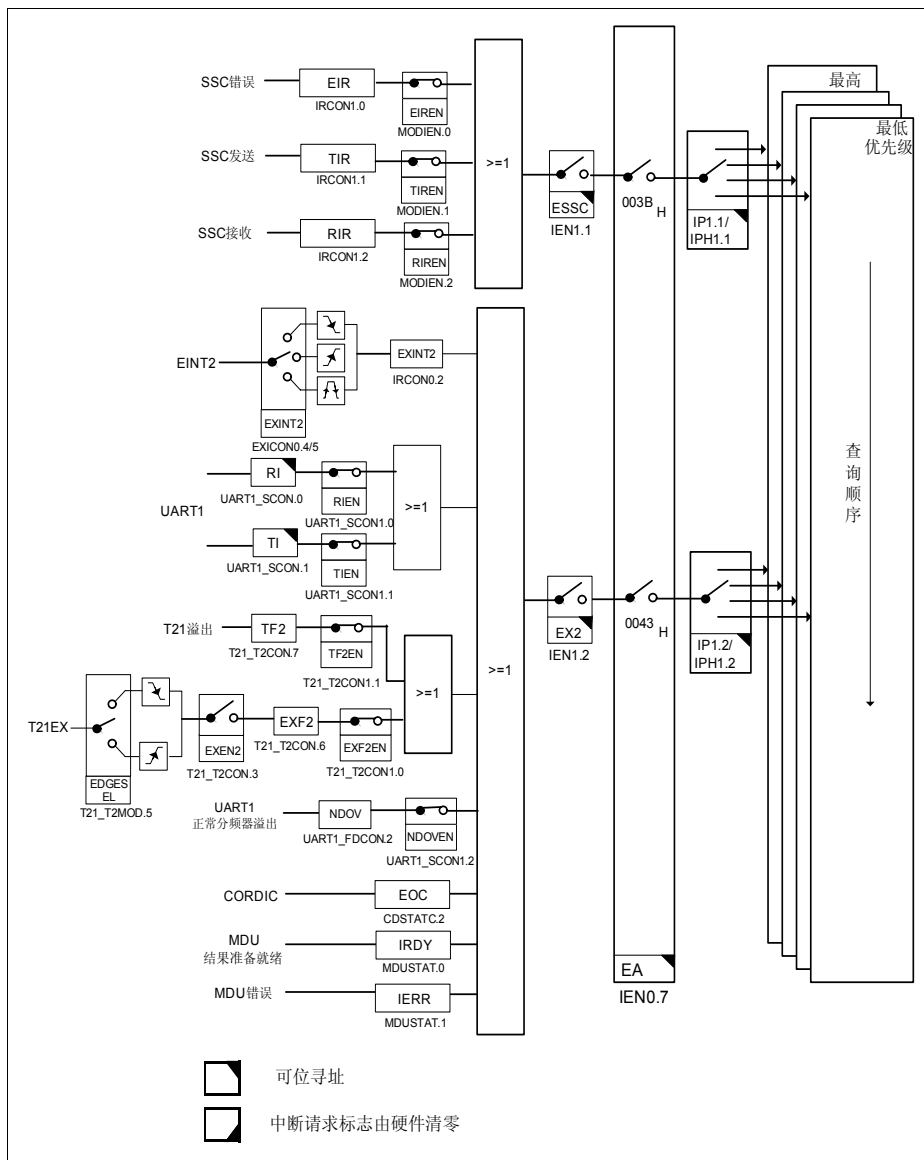


图 5-2 中断请求源 (第二部分)





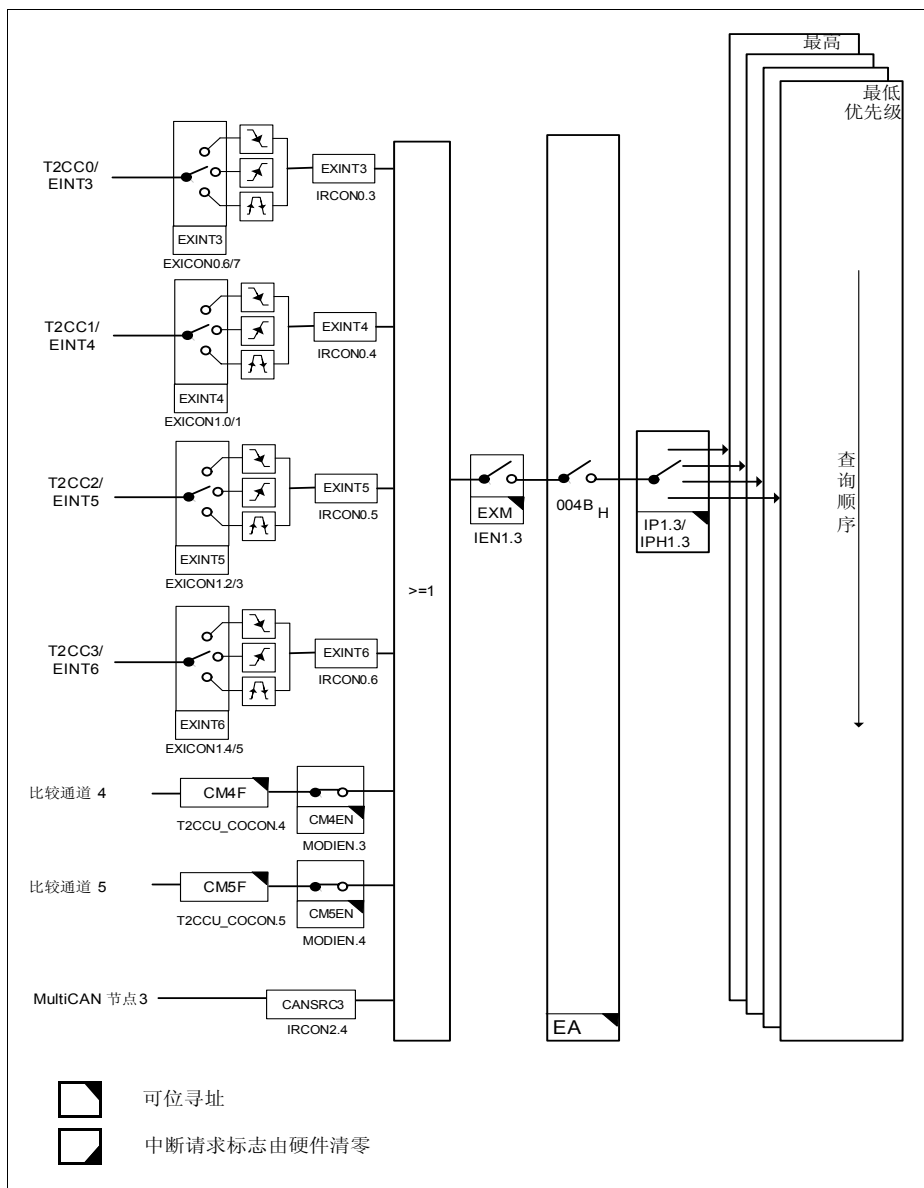


图 5-4 中断请求源（第四部分）

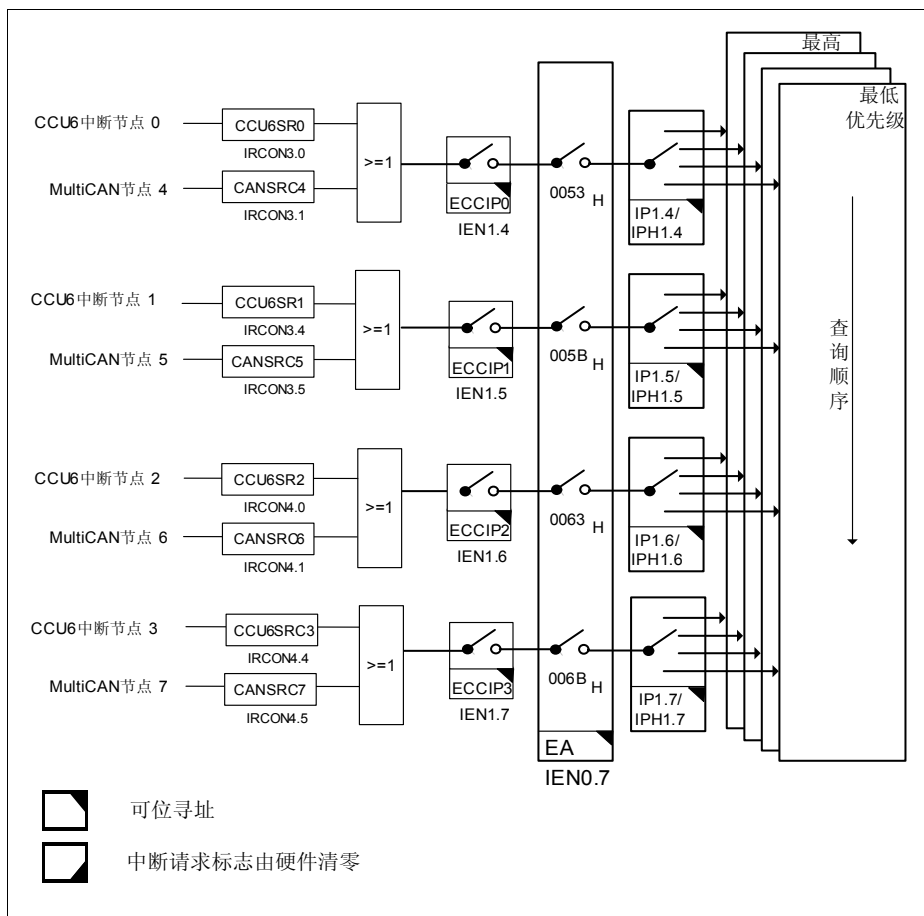


图 5-5 中断请求源（第五部分）

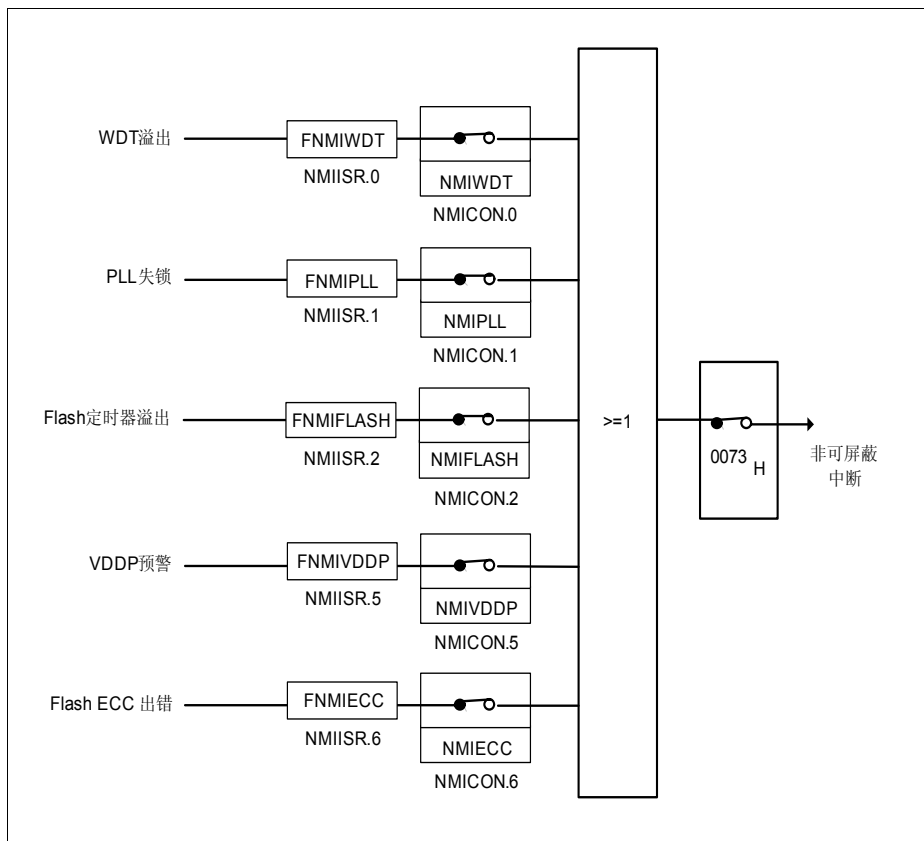


图 5-6 非可屏蔽中断请求源

## 5.1 中断结构

一个中断事件源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提供附加的使能 / 禁止位。

总的来说，XC878 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 / 中断源）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。

### 5.1.1 中断结构 1

中断结构 1 如图 5-7 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。仅当相应的中断节点被使能时，挂起中断请求才有效，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。

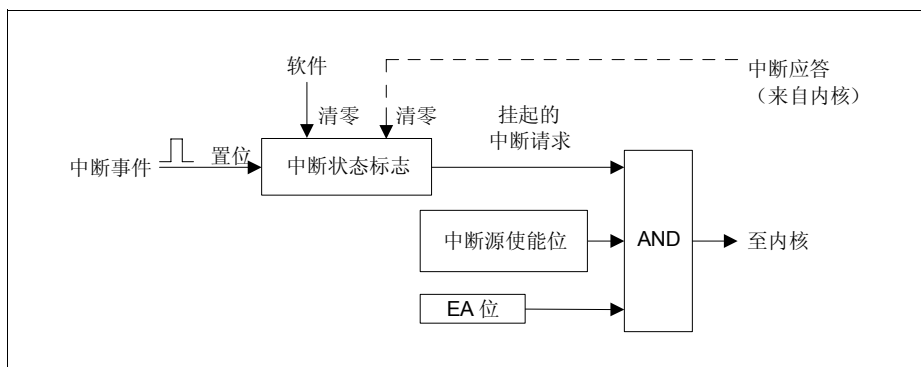


图 5-7 中断结构 1

XC878 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，相应的中断状态标志 TF0、TF1、IE0 和 IE1（位于寄存器 TCON 中）由内核清零。若某个中断节点被禁止（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 UART 模块，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。

### 5.1.2 中断结构 2

图 5-8 所示的中断结构 2 适用于定时器 T2、UART1、LIN、外部中断 2 到 6、ADC、SSC、CCU6、Flash、MDU 和 MultiCAN 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。另外还要通过寄存器 SYSCON0 中的附加控制位 IMODE 来选择处理中断事件的模式（有两种模式供选择）。

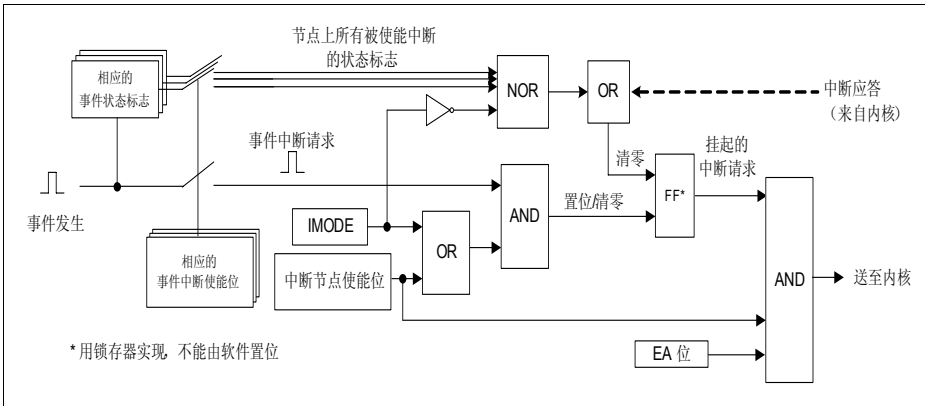


图 5-8 中断结构 2

若 IMODE=1，某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。若 IMODE=0，产生的中断事件将置位中断状态标志，不过，只有当该事件被使能并且中断节点被使能时，才会激活挂起的中断请求。考虑产生中断事件但其中断节点被禁止的情况（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，若 IMODE = 1，先前激活的挂起中断请求此时将产生中断请求送至内核；若 IMODE = 0，则不产生中断请求。

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；清零方式取决于 IMODE 的设置。

若 IMODE=1，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。请注意这和中断结构 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。若 IMODE=0，只有当中断节点使能位被清零，才能间接清零挂起的中断请求。

因此，当 IMODE=0 时，中断节点使能位具有双重功能：使能 / 禁止产生挂起中断请求、对已产生的挂起中断请求清零（通过复位使能位完成）。

注：中断结构2适用于NMI，此时不需要控制位EA，"中断节点使能位"由所有NMICON位的逻辑或（OR）取代。因此当IMODE=1时，NMI节点不可被屏蔽；当IMODE=0时，清零所有NMICON位，NMI挂起中断请求可被清零。

5.1.2.1 系统控制寄存器 0

SYSCON0 寄存器中的控制位用于 SFR 映射和中断结构 2 模式选择。

**SYSCON0**  
系统控制寄存器 0 复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
	0		IMODE	0	1	0	RMAP
	r		rw	r	r	r	rw

符号	位	类型	描述
IMODE	4	rw	中断结构 2 模式选择 0 选择中断结构 2 模式 0 1 选择中断结构 2 模式 1
1	2	r	保留 读操作返回 1；应写入 1。
0	1, 3, [7:5]	r	保留 读操作返回 0；应写入 0。

注： 应使用 ANL 或 ORL 指令清零/置位 IMODE。

## 5.2 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。XC878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 5-1](#)。

**表 5-1 中断向量地址**

中断节点	中断向量地址	XC878 中断源分配	使能位	SFR
NMI	0073 <sub>H</sub>	看门狗定时器 NMI	NMIWDT	NMICON
		PLL NMI	NMIPLL	
		Flash 定时器 NMI	NMIFLASH	
		VDDP 预警 NMI	NMIVDDP	
		Flash ECC NMI	NMIECC	
XINTR0	0003 <sub>H</sub>	外部中断 0	EX0	IEN0
XINTR1	000B <sub>H</sub>	定时器 0	ET0	
XINTR2	0013 <sub>H</sub>	外部中断 1	EX1	
XINTR3	001B <sub>H</sub>	定时器 1	ET1	
XINTR4	0023 <sub>H</sub>	UART	ES	
XINTR5	002B <sub>H</sub>	T2CCU	ET2	
		UART 分数分频器 (正常分频器溢出)		
		MultiCAN 节点 0		
		LIN		

**表 5-1 中断向量地址**

中断节点	中断向量地址	XC878 中断源分配	使能位	SFR
XINTR6	0033 <sub>H</sub>	MultiCAN 节点 1 和 2	EADC	IEN1
		ADC[1:0]		
XINTR7	003B <sub>H</sub>	SSC	ESSC	
XINTR8	0043 <sub>H</sub>	外部中断 2	EX2	
		T21		
		CORDIC		
		UART1		
		UART1 分数分频器 (正常分频器溢出)		
		MDU[1:0]		
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		
		T2CCU		
		MultiCAN 节点 3		
XINTR10	0053 <sub>H</sub>	CCU6 INP0	ECCIP0	
		MultiCAN 节点 4		
XINTR11	005B <sub>H</sub>	CCU6 INP1	ECCIP1	
		MultiCAN 节点 5		
XINTR12	0063 <sub>H</sub>	CCU6 INP2	ECCIP2	
		MultiCAN 节点 6		
XINTR13	006B <sub>H</sub>	CCU6 INP3	ECCIP3	
		MultiCAN 节点 7		



### 5.3 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 5-2。

表 5-2 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 中断	2
外部中断 1	3
定时器 T1 中断	4
UART 中断	5
T2CCU、UART 正常分频器溢出、LIN、MultiCAN 中断	6
ADC、MultiCAN 中断	7
SSC 中断	8
外部中断 2、定时器 T21、UART1、UART1 正常分频器溢出、CORDIC、MDU 中断	9
外部中断 [6:3]、MultiCAN 中断	10
CCU6 中断节点指针 0、MultiCAN 中断	11
CCU6 中断节点指针 1、MultiCAN 中断	12
CCU6 中断节点指针 2、MultiCAN 中断	13
CCU6 中断节点指针 3、MultiCAN 中断	14

## 5.4 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前一周期的 **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** 或 **IP**、**IPH/IP1**、**IP1H** 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** 或 **IP**、**IPH/IP1**、**IP1H** 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。请注意，如果中断标志有效（置位），但由于上述的条件之一而未被响应；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），并将被响应中断源的中断向量地址重新装入 **PC**，中断向量地址总结见[表 5-1](#)。

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。

## 5.5 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 5-9 所示。

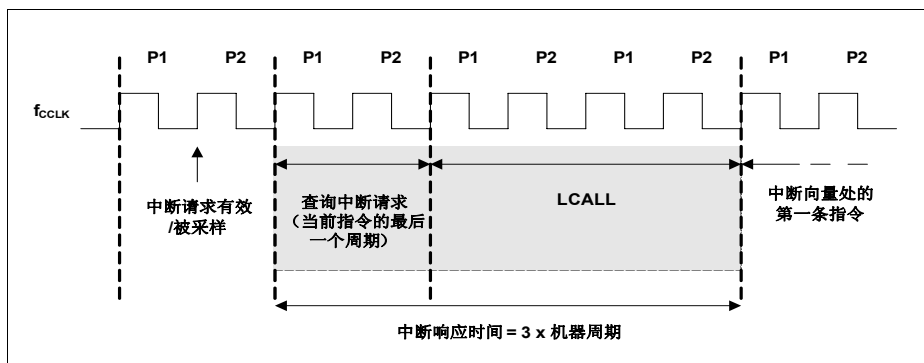


图 5-9 最短中断响应时间

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；
2. 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 5-10。
3. 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 5-11。

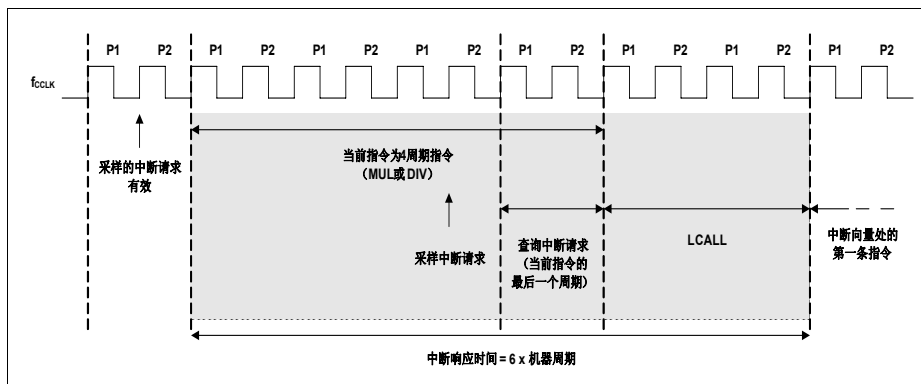


图 5-10 条件 2 下的中断响应时间

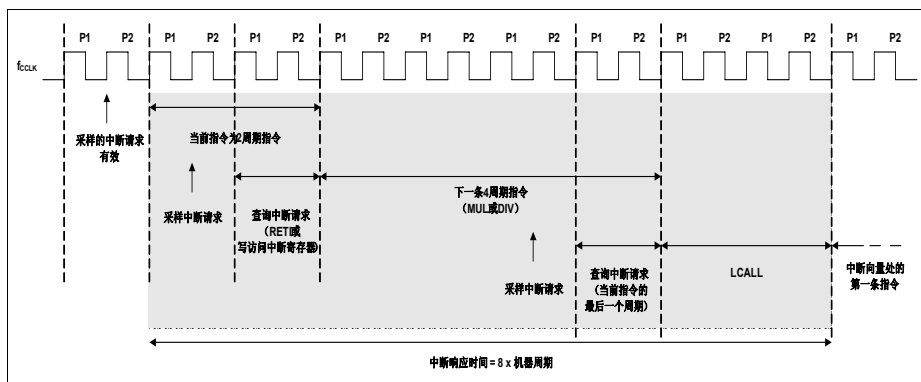


图 5-11 条件 3 下的中断响应时间

因此，在单中断系统中，如果不考虑等待状态，中断响应时间大于等于三个机器周期、小于九个机器周期。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 5-10和图 5-11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）。

## 5.6 中断寄存器

中断寄存器具有以下功能：中断节点使能、外部中断控制、中断标志和中断优先级设置。

### 5.6.1 中断节点使能寄存器

通过置位或清零中断使能寄存器 IEN0 或 IEN1 中各控制位，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 IEN0、IEN1 和 NMICON 的使能位全被清零，即所有中断源缺省被禁止。

**IEN0**  
中断使能寄存器 0 复位值：00<sub>H</sub>

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

## 中断系统

符号	位	类型	描述
<b>ET2</b>	5	rw	中断节点 <b>XINTR5</b> 使能 0 禁止 XINTR5 1 使能 XINTR5
<b>EA</b>	7	rw	全局中断屏蔽 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	保留 读操作返回 0，应写入 0。

**IEN1**

## 中断使能寄存器 1

复位值：00<sub>H</sub>

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	中断节点 <b>XINTR6</b> 使能 0 禁止 XINTR6 1 使能 XINTR6
<b>ESSC</b>	1	rw	中断节点 <b>XINTR7</b> 使能 0 禁止 XINTR7 1 使能 XINTR7
<b>EX2</b>	2	rw	中断节点 <b>XINTR8</b> 使能 0 禁止 XINTR8 1 使能 XINTR8
<b>EXM</b>	3	rw	中断节点 <b>XINTR9</b> 使能 0 禁止 XINTR9 1 使能 XINTR9
<b>ECCIP0</b>	4	rw	中断节点 <b>XINTR10</b> 使能 0 禁止 XINTR10 1 使能 XINTR10
<b>ECCIP1</b>	5	rw	中断节点 <b>XINTR11</b> 使能 0 禁止 XINTR11 1 使能 XINTR11

## 中断系统

符号	位	类型	描述
<b>ECCIP2</b>	6	rw	<b>中断节点 XINTR12 使能</b> 0 禁止 XINTR12 1 使能 XINTR12
<b>ECCIP3</b>	7	rw	<b>中断节点 XINTR13 使能</b> 0 禁止 XINTR13 1 使能 XINTR13

**NMICON**
**NMI 控制寄存器**

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>0</b>	<b>NMIECC</b>	<b>NMIVDDP</b>	<b>0</b>	<b>NMIOCD</b>	<b>NMIFLASH</b>	<b>NMIPLL</b>	<b>NMIWDT</b>
r	rw	rw	r	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>NMIPLL</b>	1	rw	<b>PLL 失锁 NMI 使能</b> 0 禁止 PLL 失锁 NMI 1 使能 PLL 失锁 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>NMIOCD</b>	3	rw	<b>OCDS NMI 使能</b> 0 禁止 OCDS NMI 1 保留
<b>NMIVDDP</b>	5	rw	<b>VDDP 预警 NMI 使能</b> 0 禁止 V <sub>DDP</sub> NMI 1 使能 V <sub>DDP</sub> NMI <i>注：当外部电源为3.3V，用户必须禁止NMIVDDP。</i>
<b>NMIECC</b>	6	rw	<b>ECC NMI 使能</b> 0 禁止 ECC NMI 1 使能 ECC NMI

符号	位	类型	描述
0	4, 7	r	保留 读操作返回 0，应写入 0。



### 5.6.2 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 XC878。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，若进行跳变沿检测，TCON.IT0/1 必须设置为下降沿触发。若检测到有效的跳变沿事件，内部将产生两个 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，以确保该跳变被识别。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。

所有外部中断均可选择从两个引脚输入，由寄存器 MODPISEL 和 MODPISEL1 中的位 EXINTxIS 控制。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

复位值：F0<sub>H</sub>

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 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 边沿检测旁路。中断请求信号直接送入内核

## 中断系统

符号	位	类型	描述
<b>EXINT2</b>	[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
<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

复位值: 3F<sub>H</sub>

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。

MODPISEL

外设输入选择寄存器

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
0	URRISH	JTAGTDIS	JTAGTCKS	EXINT2IS	EXINT1IS	EXINT0IS	URRIS
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
EXINT0IS	1	rw	外部中断 0 输入选择 0 选择外部中断输入 EXINT0_0 1 选择外部中断输入 EXINT0_1
EXINT1IS	2	rw	外部中断 1 输入选择 0 选择外部中断输入 EXINT1_0 1 选择外部中断输入 EXINT1_1
EXINT2IS	3	rw	外部中断 2 输入选择 0 选择外部中断输入 EXINT2_0 1 选择外部中断输入 EXINT2_1
0	7	r	保留 读操作返回 0，应写入 0。

MODPISEL1

外设输入选择寄存器 1

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
EXINT6IS			UR1RIS		T21EXIS	0	
r			rw		rw	r	

符号	位	类型	描述
<b>EXINT6IS</b>	[7:5]	rw	<b>外部中断 6 输入 /T2CCU 捕获 / 比较通道 3 选择</b> 000 选择外部中断输入 EXINT6_0 001 选择外部中断输入 EXINT6_1 010 选择外部中断输入 EXINT6_2/T2CCU 捕获 / 比较通道 T2CC3_0 011 选择外部中断输入 EXINT6_3/T2CCU 捕获 / 比较通道 T2CC3_1 100 选择外部中断输入 EXINT6_4/T2CCU 捕获 / 比较通道 T2CC3_2 101 保留 110 保留 111 保留
<b>0</b>	[1:0]	r	<b>保留</b> 读操作返回 0，应写入 0。

## MODPISEL4

外设输入选择寄存器 4

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>0</b>	<b>EXINT5IS</b>	<b>EXINT4IS</b>	<b>EXINT3IS</b>				
r	rw	rw	rw				

符号	位	类型	描述
<b>EXINT3IS</b>	[1:0]	rw	<b>外部中断 3 输入 /T2CCU 捕获 / 比较通道 0 选择</b> 00 选择外部中断输入 EXINT3_0 01 选择外部中断输入 EXINT3_1/T2CCU 捕获 / 比较通道 T2CC0_0 10 选择外部中断输入 EXINT3_2/T2CCU 捕获 / 比较通道 T2CC0_1 11 选择外部中断输入 EXINT3_3/T2CCU 捕获 / 比较通道 T2CC0_2

## 中断系统

符号	位	类型	描述
<b>EXINT4IS</b>	[3:2]	rw	<b>外部中断 4 输入 /T2CCU 捕获 / 比较通道 1 选择</b> 00 选择外部中断输入 EXINT4_0 01 选择外部中断输入 EXINT4_1/T2CCU 捕获 / 比较通道 T2CC1_0 10 选择外部中断输入 EXINT4_2/T2CCU 捕获 / 比较通道 T2CC1_1 11 选择外部中断输入 EXINT4_3/T2CCU 捕获 / 比较通道 T2CC1_2
<b>EXINT5IS</b>	[5:4]	rw	<b>外部中断 5 输入 /T2CCU 捕获 / 比较通道 2 选择</b> 00 选择外部中断输入 EXINT5_0 01 选择外部中断输入 EXINT5_1/T2CCU 捕获 / 比较通道 T2CC2_0 10 选择外部中断输入 EXINT5_2/T2CCU 捕获 / 比较通道 T2CC2_1 11 选择外部中断输入 EXINT5_3/T2CCU 捕获 / 比较通道 T2CC2_2
<b>0</b>	[7:6]	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

复位值：00<sub>H</sub>

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>IT0</b>	0	rw	<b>外部中断 0 电平 / 边沿触发控制标志</b> 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
<b>IT1</b>	2	rw	<b>外部中断 1 电平 / 边沿触发控制标志</b> 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

### 5.6.3 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。

#### IRCON0

中断请求寄存器 0

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	EXINT6	EXINT5	EXINT4	EXINT3	EXINT2	EXINT1	EXINT0
r	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>EXINTx</b> (x = 0 - 1)	[1:0]	rwh	<b>外部中断 0/1 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 这些标志由外部中断输入上的跳变沿（上升沿 / 下降沿 / 任意沿）置位，这些标志位不直接控制送入内核的中断信号；而是由相应的 TCON 标志来控制送入内核的中断请求 - 查询并清零标志 TCON 足以控制外部中断请求。
<b>EXINT2</b>	2	rwh	<b>外部中断 2 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>EXINTy</b> (y = 3 - 6)	[6:3]	rwh	<b>外部中断 y 或 T2CCU 捕获 / 比较通道 z（z = 0-3）的中断标志</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>0</b>	7	r	<b>保留</b> 读操作返回 0，应写入 0。

# IRCON1

## 中断请求寄存器 1

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	CANSRC2	CANSRC1	ADCSR1	ADCSR0	RIR	TIR	EIR
r	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位	类型	描述
EIR	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
TIR	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
RIR	2	rwh	<b>SSC 接收中断标志</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
ADCSR0	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
ADCSR1	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
CANSRC1	5	rwh	<b>MultICAN 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
CANSRC2	6	rwh	<b>MultICAN 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

## 中断系统

符号	位	类型	描述
<b>0</b>	7	r	保留 读操作返回 0，应写入 0。

**IRCON2**

## 中断请求寄存器 2

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
	<b>0</b>		<b>CANSRC3</b>		<b>0</b>		<b>CANSRC0</b>
	r		rwh		r		rwh

符号	位	类型	描述
<b>CANSRC0</b>	0	rwh	<b>MultiCAN 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>CANSRC3</b>	3	rwh	<b>MultiCAN 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
<b>0</b>	[3:1], [7:5]	r	保留 读操作返回 0，应写入 0。

**IRCON3**

## 中断请求寄存器 3

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>0</b>		<b>CANSRC5</b>	<b>CCU6SR1</b>		<b>0</b>	<b>CANSRC4</b>	<b>CCU6SR0</b>
r		rwh	rwh		r	rwh	rwh

符号	位	类型	描述
<b>CCU6SR0</b>	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



## 中断系统

符号	位	类型	描述
<b>CANSRC4</b>	1	rwh	<b>MultiCAN 中断标志 4</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>CCU6SR1</b>	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
<b>CANSRC5</b>	5	rwh	<b>MultiCAN 中断标志 5</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>	[3:2], [7:6]	r	<b>保留</b> 读操作返回 0，应写入 0。

## IRCON4

## 中断请求寄存器 4

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	CANSRC7	CCU6SR3	0	CANSRC6	CCU6SR2		
r	rwh	rwh	r	rwh	rwh		

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>CANSRC6</b>	1	rwh	<b>MultiCAN 中断标志 6</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>CCU6SR3</b>	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>CANSRC7</b>	5	rwh	<b>MultiCAN 中断标志 7</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>	[3:2], [7:6]	r	<b>保留</b> 读操作返回 0，应写入 0。

## TCON

### 定时器控制寄存器

复位值：00<sub>H</sub>

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

串行通道控制寄存器

复位值: 00<sub>H</sub>

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

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>0</b>	<b>FNMIIECC</b>	<b>FNMI VDDP</b>	<b>0</b>	<b>FNMI OCDS</b>	<b>FNMI FLASH</b>	<b>FNMIPLL</b>	<b>FNMIWDT</b>
r	rwh	rwh	r	rwh	r	rwh	rwh

符号	位	类型	描述
<b>FNMIWDT</b>	0	rwh	<b>WDT NMI 标志</b> 0 未产生 WDT NMI 1 已产生 WDT 预警
<b>FNMIPLL</b>	1	rwh	<b>PLL NMI 标志</b> 0 未产生 PLL NMI 1 PLL 已与外部晶振失锁
<b>FNMIFLASH</b>	2	rwh	<b>Flash 定时器 NMI 标志</b> 0 未产生 Flash 定时器 NMI 1 已产生 Flash 定时器 NMI
<b>FNMIOCDS</b>	3	rwh	<b>OCDS NMI 标志</b> 0 未产生 OCDS NMI 1 保留

符号	位	类型	描述
<b>FNMI VDDP</b>	5	rwh	<b>VDDP 预警 NMI 标志</b> 0 未产生 $V_{DDP}$ NMI 1 已产生 $V_{DDP}$ 预警（若外部电压为 5.0V，对应的 $V_{DDP}$ 预警电压 4.0V）
<b>FNMI ECC</b>	6	rwh	<b>ECC NMI 标志</b> 0 未产生 ECC 错误 1 ECC 已出错
<b>0</b>	4,7	r	<b>保留</b> 读操作返回 0，应写入 0。

NMISR 寄存器只能由软件清零，或在上电复位 / 硬件复位 / 压降复位之后被置为缺省值。在其它复位方式下，如 WDT 复位或掉电唤醒复位时，该寄存器内容保持不变，从而使系统能够确定产生前次 NMI 的中断源。

5.6.4 中断优先级寄存器

每个中断源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，共同确定该中断源的优先级（4 级优先级供选择），见表 5-3。

表 5-3 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 总是具有最高优先级（高于优先级 3），故不需要通过表 5-3 进行优先级选择。

IP

中断优先级寄存器

复位值：00<sub>H</sub>

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

复位值：00<sub>H</sub>

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

复位值：04<sub>H</sub>

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
PCCIP2	6	rw	中断节点 XINTR12 的优先级低位

# 中断系统

符号	位	类型	描述
<b>PCCIP3</b>	7	rw	中断节点 <b>XINTR13</b> 的优先级低位

## IPH1

中断优先级寄存器 1，高位字节

复位值：04<sub>H</sub>

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

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

## 5.7 中断标志概览

中断事件的中断标志存放在不同的 SFR 中。表 5-4 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 5-4 中断请求标志的位置**

中断源	中断请求标志	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
定时器 T21 溢出	TF2	T21_T2CON
定时器 T21 外部事件	EXF2	T21_T2CON
LIN 同步字节结束	EOFSYN	FDCON
LIN 同步字节出错	ERRSYN	FDCON
UART 接收	RI	SCON
UART 发送	TI	SCON
UART 正常分频器溢出	NDOV	FDCON
UART1 接收	RI	UART1_SCON
UART1 发送	TI	UART1_SCON
UART1 正常分频器溢出	NDOV	UART1_FDCON
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/T2CC0	EXINT3	IRCON0
外部中断 4/T2CC1	EXINT4	IRCON0
外部中断 5/T2CC2	EXINT5	IRCON0
外部中断 6/T2CC3	EXINT6	IRCON0
T2CC4	CM4F	T2CCU_COCON
T2CC5	CM5F	T2CCU_COCON
T2CCU CCT 溢出	CCTOVF	T2CCU_CCTCON
MDU 结果准备就绪	IRDY	MDUSTAT
MDU 出错	IERR	MDUSTAT
ADC 服务请求 0	ADCSR0	IRCON1



表 5-4 中断请求标志的位置

中断源	中断请求标志	SFR
ADC 服务请求 1	ADCSR1	IRCON1
SSC 错误	EIR	IRCON1
SSC 发送	TIR	IRCON1
SSC 接收	RIR	IRCON1
CORDIC 中断	EOC	CDSTATC
MultiCAN 中断 0	CANSRC0 <sup>1)</sup>	IRCON2
MultiCAN 中断 1	CANSRC1 <sup>1)</sup>	IRCON1
MultiCAN 中断 2	CANSRC2 <sup>1)</sup>	IRCON1
MultiCAN 中断 3	CANSRC3 <sup>1)</sup>	IRCON2
MultiCAN 中断 4	CANSRC4 <sup>1)</sup>	IRCON3
MultiCAN 中断 5	CANSRC5 <sup>1)</sup>	IRCON3
MultiCAN 中断 6	CANSRC6 <sup>1)</sup>	IRCON4
MultiCAN 中断 7	CANSRC7 <sup>1)</sup>	IRCON4
CCU6 节点 0 中断	CCU6SR0	IRCON3
CCU6 节点 1 中断	CCU6SR1	IRCON3
CCU6 节点 2 中断	CCU6SR2	IRCON4
CCU6 节点 3 中断	CCU6SR3	IRCON4
WDT NMI	FNMIWDT	NMISR
PLL NMI	FNMIPLL	NMISR
FLASH 定时器溢出 NMI	FNMIFLASH	NMISR
V <sub>DDP</sub> 预警 NMI	FNMIVDDP	NMISR
Flash ECC NMI	FNMI ECC	NMISR

<sup>1)</sup> MultiCAN 产生的各中断可分配给不同的 MultiCAN 中断节点 [7:0]，由 MultiCAN 寄存器 NIPRx/MOIPRn 进行选择。

## 6 并行端口

XC878 有 40 个端口引脚，组织成 P0、P1、P3、P4 和 P5 五个并行端口。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。这些端口均为双向口，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）。引脚设置为输出口时，可选择漏极开路模式。

### 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
- 驱动能力可配置
- 通过数字输入输出口传送数据（通用输入 / 输出）
- 片上外设的输入 / 输出口（复用输入输出功能）

## 6.1 基本端口操作

XC878 双向口引脚的结构框图如图 6-1 所示。每个引脚由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。通过设定控制寄存器，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。

每个双向口可配置成输入或输出。由寄存器 **Px\_DIR** ( $x=0, 1, 3, 4$  或  $5$ ) 控制输入输出模式之间的切换、开启或关闭输出和输入驱动器。任何时刻端口只能被设置成输入或输出模式之一。

输入模式下（复位后的缺省模式），输出驱动器关闭（高阻）。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\_DATA** 中读取。

输出模式下，输出驱动器被激活，将输出复用器送出的值驱动至端口引脚。在输出驱动器中，通过控制寄存器 **Px\_OD**，可将每个端口切换至漏极开路模式或正常模式（推挽模式）。输出驱动器的驱动能力还可通过寄存器 **Px\_DS** 来配置。

输出驱动器的前级输出复用器使输出口可用作多种功能。如果引脚用作通用输出口，软件控制输出复用器选通数据寄存器 **Px\_DATA**，软件可对 **Px\_DATA** 置位或清零，从而直接影响引脚的状态；如果引脚用作片上外设的输出口，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 **Px\_ALTSEL0** 和 **Px\_ALTSEL1** 进行复用功能选择。当引脚用作复用功能时，必须在寄存器 **Px\_DIR** 中相应设置引脚方向。

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。

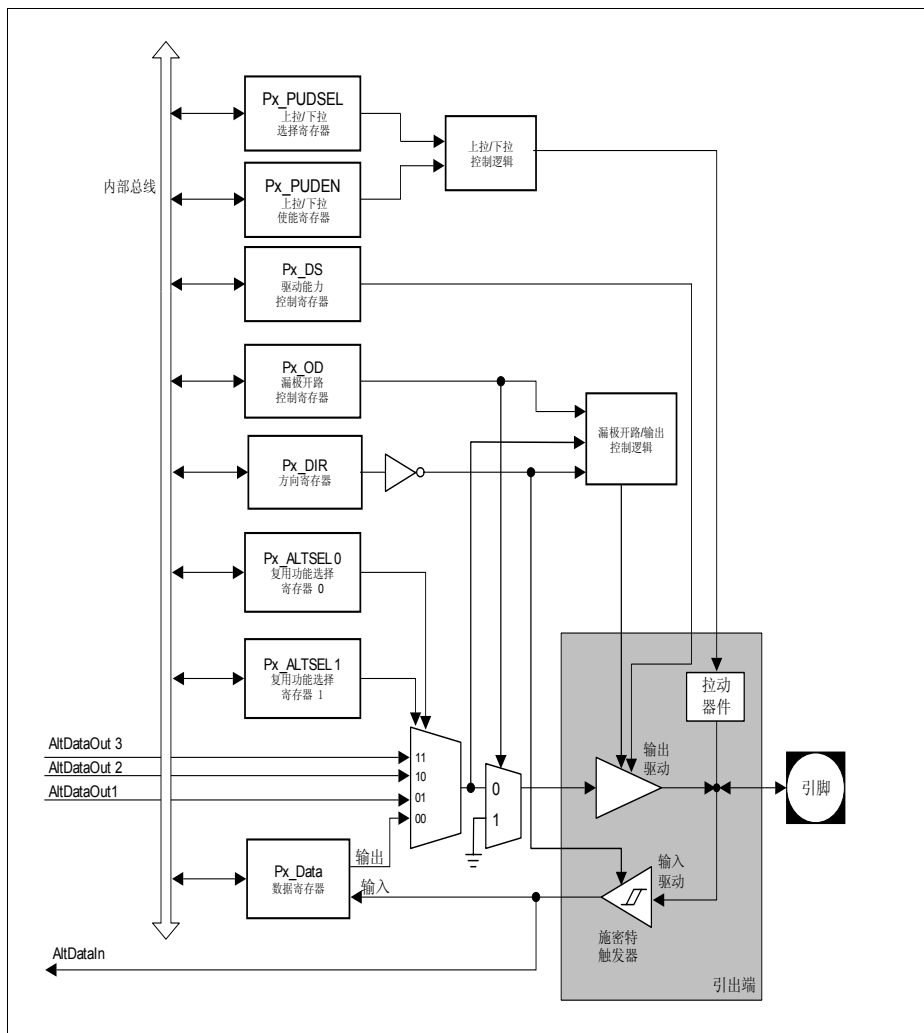


图 6-1 双向口基本结构

### 6.1.1 通用寄存器描述

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。

每个端口寄存器的具体描述请参见[章节 6.3](#)至[章节 6.7](#)。本节仅对不同端口的寄存器予以概述。

**表 6-1 端口寄存器**

寄存器缩写名	寄存器全名	具体描述请参见
Px_DATA	端口 x 数据寄存器	<a href="#">页 6-5</a>
Px_DIR	端口 x 方向寄存器	<a href="#">页 6-6</a>
Px_OD	端口 x 漏极开路控制寄存器	<a href="#">页 6-6</a>
Px_PUDSEL	端口 x 上拉 / 下拉选择寄存器	<a href="#">页 6-7</a>
Px_PUDEN	端口 x 上拉 / 下拉使能寄存器	<a href="#">页 6-7</a>
Px_ALTSEL0	端口 x 复用功能选择寄存器 0	<a href="#">页 6-9</a>
Px_ALTSEL1	端口 x 复用功能选择寄存器 1	<a href="#">页 6-9</a>
Px_DS	端口 x 驱动能力控制寄存器	<a href="#">页 6-10</a>

6.1.1.1 数据寄存器

如果引脚用作通用输出口，输出数据写入数据寄存器 Px\_DATA 中。如果引脚用作通用输入口，端口引脚的锁存值可从寄存器 Px\_DATA 中读出。

注： 如果无外部驱动源，用作输入的端口引脚将会锁存有效的内部上拉 / 下拉设置；并用该有效的上拉 / 下拉值更新寄存器 Px\_DATA。

Px\_DATA  
端口 x 数据寄存器

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

符号	位	类型	描述
Pn (n = 0 - 7)	n	rwh	端口 x 引脚 n 的数据值 0 端口 x 引脚 n 的数据值 = 0 1 端口 x 引脚 n 的数据值 = 1

位 Px\_DATA.n 只有在相应引脚设置为输出 (Px\_DIR.n = 1) 时可被写入，在引脚设置为输入 (Px\_DIR.n = 0) 时不能写入。当引脚设置为 GPIO、且方向被切换 / 设置为输出时，Px\_DATA.n 的内容输出到指定引脚上。读取 Px\_DATA 将返回寄存器的值，而不是相应 Px\_DATA 引脚上的状态。

### 6.1.1.2 方向寄存器

双向端口引脚的方向由相应的方向寄存器 Px\_DIR 控制。

#### Px\_DIR

端口 x 方向寄存器

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	双向口：端口 x 引脚 n 方向控制 0 设置为输入引脚 1 设置为输出引脚

### 6.1.1.3 漏极开路控制寄存器

每个输出引脚均可工作在漏极开路模式。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 / 下拉器件的设置或者处于三态模式；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	端口 x 引脚 n 漏极开路模式选择 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

### 6.1.1.4 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。每个引脚可分别选择上拉 / 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
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 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
P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	rw	rw



并行端口

符号	位	类型	描述
<b>Pn</b> <b>(n = 0 - 7)</b>	n	rw	端口 x 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

### 6.1.1.5 复用输入和输出功能

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可从寄存器读取数字输入值、或从引脚直接读取数字输入值。

由输出复用器来选择复用输出功能（最多可选择 4 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

由寄存器 Px\_ALTSEL0 和 Px\_ALTSEL1 选择复用输出功能。

#### Px\_ALTSELn(n = 0 - 1)

端口 x 复用功能选择寄存器

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> <b>(n = 0 - 7)</b>	n	rw	<b>引脚输出功能</b> 配置 Px_ALTSEL0.Pn 和 Px_ALTSEL1.Pn 选择引脚用作 GPIO 或复用功能： 00 GPIO 功能 10 复用输出功能 1 01 复用输出功能 2 11 复用输出功能 3

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

### 6.1.1.6 驱动能力控制寄存器

可由寄存器 Px\_DS 控制端口的驱动能力，可选择强驱动或弱驱动。

#### Px\_DS

端口 x 驱动能力控制寄存器

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	端口 x 引脚 n 的驱动能力控制 0 弱驱动 1 强驱动

6.2 寄存器地址映射

端口 SFR 在标准存储器区（RMAP = 0），由四页构成。PORT\_PAGE 寄存器位于地址单元 B2<sub>H</sub>，包含了页信息和分页控制信息。

端口 SFR 的地址见表 6-2。

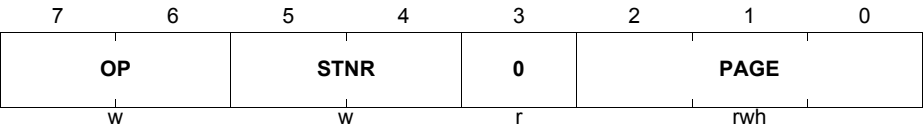
表 6-2 页 0-3 的 SFR 地址列表

地址	页 0	页 1	页 2	页 3
80H	P0_DATA	P0_PUDSEL	P0_ALTSEL0	P0_OD
86H	P0_DIR	P0_PUDEN	P0_ALTSEL1	P0_DS
90H	P1_DATA	P1_PUDSEL	P1_ALTSEL0	P1_OD
91H	P1_DIR	P1_PUDEN	P1_ALTSEL1	P1_DS
92H	P5_DATA	P5_PUDSEL	P5_ALTSEL0	P5_OD
93H	P5_DIR	P5_PUDEN	P5_ALTSEL1	P5_DS
B0H	P3_DATA	P3_PUDSEL	P3_ALTSEL0	P3_OD
B1H	P3_DIR	P3_PUDEN	P3_ALTSEL1	P3_DS
C8H	P4_DATA	P4_PUDSEL	P4_ALTSEL0	P4_OD
C9H	P4_DIR	P4_PUDEN	P4_ALTSEL1	P4_DS

PORT\_PAGE

端口分页寄存器

复位值：00<sub>H</sub>



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

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 $OP=10_B$ , PAGE 的内容在被新值覆盖之前保存在 STx 中。 若 $OP = 11_B$ , PAGE 的内容被 STx 的内容覆盖。写入 PAGE 的值被忽略。 00 选择 ST0 01 选择 ST1 10 选择 ST2 11 选择 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 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 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。

### 6.3 P0 口

P0 口是一个 8 位通用双向口。

P0 口寄存器归纳见表 6-3。

表 6-3 P0 口寄存器

寄存器缩写名	寄存器全名
P0_DATA	P0 口数据寄存器
P0_DIR	P0 口方向寄存器
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_DS	P0 口驱动能力控制寄存器

#### 6.3.1 功能

P0 口输入和输出功能归纳见表 6-4。

由寄存器 P0\_ALTSEL0 和 P0\_ALTSEL1 选择复用输出功能。

表 6-4 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_DATA.P0	—
		ALT1	TCK_0	JTAG
		ALT2	T12HR_1	CCU6
		ALT3	CC61_1	CCU6
	输出	GPO	P0_DATA.P0	—
		ALT1	CLKOUT_0	时钟输出
		ALT2	CC61_1	CCU6
		ALT3	RXDO_1	UART

表 6-4 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_DATA.P1	–
		ALT1	TDI_0	JTAG
		ALT2	T13HR_1	CCU6
		ALT3	RXD_1	UART
		ALT4	RXDC1_0	MultiCAN
	输出	GPO	P0_DATA.P1	–
		ALT1	EXF2_1	T2CCU
		ALT2	COUT61_1	CCU6
		ALT3	–	–
P0.2	输入	GPI	P0_DATA.P2	–
		ALT1	–	–
		ALT2	CTRAP_2	CCU6
		ALT3	–	–
	输出	GPO	P0_DATA.P2	–
		ALT1	TDO_0	JTAG
		ALT2	TXD_1	UART
		ALT3	TXDC1_0	MultiCAN
P0.3	输入	GPI	P0_DATA.P3	–
		ALT1	SCK_1	SSC
		ALT2	–	–
		ALT3	–	–
	输出	GPO	P0_DATA.P3	–
		ALT1	SCK_1	SSC
		ALT2	COUT63_1	CCU6
		ALT3	RXDO1_0	UART1
		EXT_INT	A17	外部接口

表 6-4 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
P0.4	输入	GPI	P0_DATA.P4	–
		ALT1	MTSR_1	SSC
		ALT2	–	–
		ALT3	CC62_1	CCU6
	输出	GPO	P0_DATA.P4	–
		ALT1	MTSR_1	SSC
		ALT2	CC62_1	CCU6
		ALT3	TXD1_0	UART1
		EXT_INT	A18	外部接口
P0.5	输入	GPI	P0_DATA.P5	–
		ALT1	MRST_1	SSC
		ALT2	EXINT0_0	外部中断 0
		ALT3	T2EX1_1	定时器 T21
		ALT4	RXD1_0	UART1
	输出	GPO	P0_DATA.P5	–
		ALT1	MRST_1	SSC
		ALT2	COUT62_1	CCU6
		ALT3	–	–
		EXT_INT	A19	外部接口
P0.6	输入	GPI	P0_DATA.P6	–
		ALT1	–	–
		ALT2	–	–
		ALT3	–	–
	输出	GPO	P0_DATA.P6	–
		ALT1	T2CC4_1	T2CCU
		ALT2	–	–
		ALT3	–	–
		EXT_INT	WR	外部接口



**表 6-4 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_DATA.P7	—
		ALT1	—	—
		ALT2	—	—
		ALT3	—	—
	输出	GPO	P0_DATA.P7	—
		ALT1	CLKOUT_1	SCU
		ALT2	T2CC5_1	T2CCU
		ALT3	—	—
		EXT_INT	RD	外部接口

### 6.3.2 寄存器描述

#### P0\_DATA

##### P0 口数据寄存器

复位值：00<sub>H</sub>

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 (缺省值) 1 P0 口引脚 n 的数据值 = 1

#### P0\_DIR

##### P0 口方向寄存器

复位值：00<sub>H</sub>

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 方向控制</b> 0 设置为输入引脚 (缺省值) 1 设置为输出引脚

#### P0\_OD

##### P0 口漏极开路控制寄存器

复位值：00<sub>H</sub>

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

并行端口

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	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（缺省值） 1 漏极开路模式；只有 0 可被有效输出

**P0\_PUDSEL**
**P0 口上拉 / 下拉选择寄存器**

 复位值：FF<sub>H</sub>

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 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**

 复位值：C4<sub>H</sub>

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 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（缺省值）

并行端口

**P0\_ALTSEL<sub>n</sub>(n = 0 - 1)**
**P0 口复用功能选择寄存器**

 复位值: **00<sub>H</sub>**

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>P<sub>n</sub></b> <b>(n = 0 - 7)</b>	n	rw	<b>引脚输出功能</b> 配置 P0_ALTSEL0.P <sub>n</sub> 和 P0_ALTSEL1.P <sub>n</sub> 选择引脚用作 GPIO 或复用功能: 00 GPIO 功能 (缺省值) 10 复用输出功能 1 01 复用输出功能 2 11 复用输出功能 3

**P0\_DS**
**P0 口驱动能力控制寄存器**

 复位值: **FF<sub>H</sub>**

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>P<sub>n</sub></b> <b>(n = 0 - 7)</b>	n	rw	<b>P0 口引脚 n 的驱动能力控制</b> 0 弱驱动 1 强驱动 (缺省值)

## 6.4 P1 口

P1 口是一个 8 位通用双向口。

P1 口寄存器归纳见[表 6-5](#)。

**表 6-5 P1 口寄存器**

寄存器缩写名	寄存器全名
P1_DATA	P1 口数据寄存器
P1_DIR	P1 口方向寄存器
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_DS	P1 口驱动能力控制寄存器

### 6.4.1 功能

P1 口输入和输出功能归纳见[表 6-6](#)。

由寄存器 P1\_ALTSEL0 和 P1\_ALTSEL1 选择复用输出功能。

**表 6-6 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_DATA.P0	—
		ALT1	RXD_0	UART
		ALT2	T2EX_0	T2CCU
		ALT3	RXDC0_0	MultiCAN
	输出	GPO	P1_DATA.P0	—
		ALT1	—	—
		ALT2	—	—
		ALT3	—	—
		EXT_INT	A8	外部接口

表 6-6 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
P1.1	输入	GPI	P1_DATA.P1	–
		ALT1	–	–
		ALT2	EXINT3_0	外部中断 3
		ALT3	T0_1	定时器 T0
	输出	GPO	P1_DATA.P1	–
		ALT1	–	–
		ALT2	TXD_0	UART
		ALT3	TXDC0_0	MultiCAN
		EXT_INT	A9	外部接口
P1.2	输入	GPI	P1_DATA.P2	–
		ALT1	SCK_0	SSC
		ALT2	–	–
		ALT3	–	–
	输出	GPO	P1_DATA.P2	–
		ALT1	SCK_0	SSC
		ALT2	–	–
		ALT3	–	–
		EXT_INT	A10	外部接口
P1.3	输入	GPI	P1_DATA.P3	–
		ALT1	MTSR_0	SSC
		ALT2	SCK_2	SSC
		ALT3	–	–
	输出	GPO	P1_DATA.P3	–
		ALT1	MTSR_0	SSC
		ALT2	SCK_2	SSC
		ALT3	TXDC1_3	MultiCAN
		EXT_INT	A11	外部接口

表 6-6 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_DATA.P4	–
		ALT1	MRST_0	SSC
		ALT2	EXINT0_1	外部中断 0
		ALT3	RXDC1_3	MultiCAN
		ALT4	MTSR_2	SSC
	输出	GPO	P1_DATA.P4	–
		ALT1	MRST_0	SSC
		ALT2	MTSR_2	SSC
		ALT3	–	–
		EXT_INT	A12	外部接口
P1.5	输入	GPI	P1_DATA.P5	–
		ALT1	CCPOS0_1	CCU6
		ALT2	EXINT5_0	外部中断 5
		ALT3	T1_1	定时器 T1
		ALT4	MRST_2	SSC
	输出	GPO	P1_DATA.P5 <sup>1)</sup>	–
		ALT1	EXF2_0	T2CCU
		ALT2	RXDO_0	UART
		ALT3	MRST_2	SSC
P1.6	输入	GPI	P1_DATA.P6	–
		ALT1	CCPOS1_1	CCU6
		ALT2	T12HR_0	CCU6
		ALT3	EXINT6_0	外部中断 6
		ALT4	RXDC0_2	MultiCAN
		ALT5	T21_1	定时器 T21
	输出	GPO	P1_DATA.P6 <sup>2)</sup>	–
		ALT1	–	–
		ALT2	–	–

表 6-6 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
P1.7	输入	GPI	P1_DATA.P7	–
		ALT1	CCPOS2_1	CCU6
		ALT2	T13HR_0	CCU6
		ALT3	T2_1	T2CCU
	输出	GPO	P1_DATA.P7	–
		ALT1	–	–
		ALT2	–	–
		ALT3	TXDC0_2	MultiCAN

<sup>1)</sup> P1.5 可用作 SSC 的软件片选功能。

<sup>2)</sup> P1.6 可用作 SSC 的软件片选功能。



## 6.4.2 寄存器描述

### P1\_DATA

#### P1 口数据寄存器

复位值：00<sub>H</sub>

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>P1 口引脚 n 的数据值</b> 0 P1 口引脚 n 的数据值 = 0 （缺省值） 1 P1 口引脚 n 的数据值 = 1

### P1\_DIR

#### P1 口方向寄存器

复位值：00<sub>H</sub>

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>P1 口引脚 n 方向控制</b> 0 设置为输入引脚 （缺省值） 1 设置为输出引脚

### P1\_OD

#### P1 口漏极开路控制寄存器

复位值：00<sub>H</sub>

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

并行端口

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rw	<b>P1 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（缺省值） 1 漏极开路模式；只有 0 可被有效输出

**P1\_PUDSEL**
**P1 口上拉 / 下拉选择寄存器**

 复位值：FF<sub>H</sub>

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>P1 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）

**P1\_PUDEN**
**P1 口上拉 / 下拉使能寄存器**

 复位值：FF<sub>H</sub>

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>P1 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（缺省值）

并行端口

**P1\_ALTSELn(n = 0 - 1)**
**P1 口复用功能选择寄存器**

 复位值: **00<sub>H</sub>**

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>引脚输出功能</b> 配置 P1_ALTSEL0.Pn 和 P1_ALTSEL1.Pn 选择引脚用作 GPIO 或复用功能: 00 GPIO 功能 (缺省值) 10 复用输出功能 1 01 复用输出功能 2 11 复用输出功能 3

**P1\_DS**
**P1 口驱动能力控制寄存器**

 复位值: **FF<sub>H</sub>**

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>P1 口引脚 n 的驱动能力控制</b> 0 弱驱动 1 强驱动 (缺省值)

## 6.5 P3 口

P3 口是一个 8 位通用双向口。

P3 口寄存器归纳见[表 6-7](#)。

**表 6-7 P3 口寄存器**

寄存器缩写名	寄存器全名
P3_DATA	P3 口数据寄存器
P3_DIR	P3 口方向寄存器
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
P3_DS	P3 口驱动能力控制寄存器

### 6.5.1 功能

P3 口输入和输出功能归纳见[表 6-8](#)。

由寄存器 P3\_ALTSEL0 和 P3\_ALTSEL1 选择复用输出功能。

**表 6-8 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_DATA.P0	—
		ALT1	CC60_0	CCU6
		ALT2	CCPOS1_2	CCU6
		ALT3	T2CC0_1/EXINT3_2	T2CCU/ 外部中断 3
	输出	GPO	P3_DATA.P0	—
		ALT1	CC60_0	CCU6
		ALT2	T2CC0_1	T2CCU
		ALT3	RXDO1_1	UART1

**表 6-8 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_DATA.P1	–
		ALT1	–	–
		ALT2	CCPOS0_2	CCU6
		ALT3	CC61_2	CCU6
	输出	GPO	P3_DATA.P1	–
		ALT1	COUT60_0	CCU6
		ALT2	CC61_2	CCU6
		ALT3	TXD1_1	UART1
P3.2	输入	GPI	P3_DATA.P2	–
		ALT1	CC61_0	CCU6
		ALT2	CCPOS2_2	CCU6
		ALT3	RXDC1_1	MultiCAN
		ALT4	RXD1_1	UART1
		ALT5	T2CC1_1/EXINT4_2	T2CCU/ 外部中断 4
	输出	GPO	P3_DATA.P2	–
		ALT1	CC61_0	CCU6
		ALT2	T2CC1_1	T2CCU
		ALT3	–	–
P3.3	输入	GPI	P3_DATA.P3	–
		ALT1	T2CC2_1/EXINT5_2	T2CCU/ 外部中断 5
		ALT2	–	–
		ALT3	–	–
	输出	GPO	P3_DATA.P3	–
		ALT1	COUT61_0	CCU6
		ALT2	T2CC2_1	T2CCU
		ALT3	TXDC1_1	MultiCAN
		EXT_INT	A13	外部接口

表 6-8 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
P3.4	输入	GPI	P3_DATA.P4	—
		ALT1	CC62_0	CCU6
		ALT2	T2EX1_0	定时器 T21
		ALT3	RXDC0_1	MultiCAN
		ALT4	T2CC3_1/EXINT6_3	T2CCU/ 外部中断 6
	输出	GPO	P3_DATA.P4	—
		ALT1	CC62_0	CCU6
		ALT2	T2CC3_1	T2CCU
		ALT3	—	—
		EXT_INT	A14	外部接口
P3.5	输入	GPI	P3_DATA.P5	—
		ALT1	—	—
		ALT2	—	—
		ALT3	—	—
	输出	GPO	P3_DATA.P5	—
		ALT1	COUT62_0	CCU6
		ALT2	EXF21_0	定时器 T21
		ALT3	TXDC0_1	MultiCAN
P3.6	输入	GPI	P3_DATA.P6	—
		ALT1	CTRAP_0	CCU6
		ALT2	—	—
		ALT3	—	—
	输出	GPO	P3_DATA.P6	—
		ALT1	—	—
		ALT2	—	—
		ALT3	—	—

**表 6-8 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
P3.7	输入	GPI	P3_DATA.P7	—
		ALT1	—	—
		ALT2	EXINT4_0	外部中断 4
		ALT3	—	—
	输出	GPO	P3_DATA.P7	—
		ALT1	COUT63_0	CCU6
		ALT2	—	—
		ALT3	—	—
		EXT_INT	A16	外部接口

## 6.5.2 寄存器描述

### P3\_DATA

#### P3 口数据寄存器

 复位值: 00<sub>H</sub>

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>
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rwh	<b>P3 口引脚 n 的数据值</b> 0 P3 口引脚 n 的数据值 = 0 (缺省值) 1 P3 口引脚 n 的数据值 = 1

### P3\_DIR

#### P3 口方向寄存器

 复位值: 00<sub>H</sub>

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>P3 口引脚 n 方向控制</b> 0 设置为输入引脚 (缺省值) 1 设置为输出引脚

### P3\_OD

#### P3 口漏极开路控制寄存器

 复位值: 00<sub>H</sub>

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



并行端口

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	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（缺省值） 1 漏极开路模式；只有 0 可被有效输出

### P3\_PUDSEL

**P3 口上拉 / 下拉选择寄存器**

复位值：BF<sub>H</sub>

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>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）

注：位时引脚 P3.6 下拉器件被激活。在 Boot ROM 启动过程中，下拉器件被置为无效，引脚 P3.6 变为三态。

### P3\_PUDEN

**P3 口上拉 / 下拉使能寄存器**

复位值：40<sub>H</sub>

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>P3 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（缺省值）

并行端口

## P3\_ALTSEL0

P3 口复用功能选择 0 寄存器

复位值：00<sub>H</sub>

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>引脚输出功能</b> 配置 P3_ALTSEL0.Pn 和 P3_ALTSEL1.Pn 选择引脚用作 GPIO 或复用功能： 00 GPIO 功能（缺省值） 10 复用输出功能 1 01 复用输出功能 2 11 复用输出功能 3

## P3\_ALTSEL1

P3 口复用功能选择 1 寄存器

复位值：00<sub>H</sub>

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>引脚输出功能</b> 配置 P3_ALTSEL0.Pn 和 P3_ALTSEL1.Pn 选择引脚用作 GPIO 或复用功能： 00 GPIO 功能（缺省值） 10 复用输出功能 1 01 复用输出功能 2 11 复用输出功能 3

并行端口

## P3\_DS

P3 口驱动能力控制寄存器

复位值: FF<sub>H</sub>

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>P3 口引脚 n 的驱动能力控制</b> 0 弱驱动 1 强驱动 (缺省值)

## 6.6 P4 口

P4 口是一个 8 位通用双向口。

P4 口寄存器归纳见表 6-9。

表 6-9 P4 口寄存器

寄存器缩写名	寄存器全名
P4_DATA	P4 口数据寄存器
P4_DIR	P4 口方向寄存器
P4_OD	P4 口漏极开路控制寄存器
P4_PUDSEL	P4 口上拉 / 下拉选择寄存器
P4_PUDEN	P4 口上拉 / 下拉使能寄存器
P4_ALTSEL0	P4 口复用功能选择寄存器 0
P4_ALTSEL1	P4 口复用功能选择寄存器 1
P4_DS	P4 口驱动能力控制寄存器

### 6.6.1 功能

P4 口输入和输出功能归纳见表 6-10。由寄存器 P4\_ALTSEL0 和 P4\_ALTSEL1 选择复用输出功能。

表 6-10 P4 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
P4.0	输入	GPI	P4_DATA.P0	—
		ALT1	—	—
		ALT2	—	—
		ALT3	RXDC0_3	MultiCAN
		ALT4	T2CC0_0/EXINT3_1	T2CCU/ 外部中断 3
		EXT_INT	D0	外部接口
	输出	GPO	P4_DATA.P0	—
		ALT1	CC60_1	CCU6
		ALT2	T2CC0_0	T2CCU
		ALT3	—	—
		EXT_INT	D0	外部接口

**表 6-10 P4 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
P4.1	输入	GPI	P4_DATA.P1	—
		ALT1	T2CC1_0/EXINT4_1	T2CCU/ 外部中断 4
		ALT2	—	—
		ALT3	—	—
		EXT_INT	D1	外部接口
	输出	GPO	P4_DATA.P1	—
		ALT1	COU60_1	CCU6
		ALT2	T2CC1_0	T2CCU
		ALT3	TXDC0_3	MultiCAN
		EXT_INT	D1	外部接口
P4.2	输入	GPI	P4_DATA.P2	—
		ALT1	T21_0	定时器 T21
		ALT2	EXINT6_1	外部中断 6
		ALT3	—	—
		EXT_INT	D2	外部接口
	输出	GPO	P4_DATA.P2	—
		ALT1	—	—
		ALT2	—	—
		ALT3	—	—
		EXT_INT	D2	外部接口
P4.3	输入	GPI	P4_DATA.P3	—
		ALT1	T2EX_1	T2CCU
		ALT2	—	—
		ALT3	—	—
		EXT_INT	D3	外部接口
	输出	GPO	P4_DATA.P3	—
		ALT1	EXF21_1	定时器 T21
		ALT2	COU63_2	CCU6
		ALT3	—	—
		EXT_INT	D3	外部接口

表 6-10 P4 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
P4.4	输入	GPI	P4_DATA.P4	—
		ALT1	CCPOS0_3	CCU6
		ALT2	T0_0	定时器 T0
		ALT3	T2CC2_0/EXINT5_1	T2CCU/ 外部中断 5
		EXT_INT	D4	外部接口
	输出	GPO	P4_DATA.P4	—
		ALT1	CC61_4	CCU6
		ALT2	T2CC2_0	T2CCU
		ALT3	—	—
		EXT_INT	D4	外部接口
P4.5	输入	GPI	P4_DATA.P5	—
		ALT1	CCPOS1_3	CCU6
		ALT2	T1_0	定时器 T1
		ALT3	T2CC3_0/EXINT6_2	T2CCU/ 外部中断 6
		EXT_INT	D5	外部接口
	输出	GPO	P4_DATA.P5	—
		ALT1	COU61_2	CCU6
		ALT2	T2CC3_0	T2CCU
		EXT_INT	D5	外部接口

表 6-10 P4 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
P4.6	输入	GPI	P4_DATA.P6	—
		ALT1	CCPOS2_3	CCU6
		ALT2	T2_0	定时器 T2
		ALT3	—	—
		EXT_INT	D6	外部接口
	输出	GPO	P4_DATA.P6	—
		ALT1	CC62_2	CCU6
		ALT2	T2CC4_0	T2CCU
		ALT3	—	—
		EXT_INT	D6	外部接口
P4.7	输入	GPI	P4_DATA.P7	—
		ALT1	CTRAP_3	CCU6
		ALT2	—	—
		ALT3	—	—
		EXT_INT	D7	外部接口
	输出	GPO	P4_DATA.P7	—
		ALT1	COUT62_2	CCU6
		ALT2	T2CC5_0	T2CCU
		ALT3	—	—
		EXT_INT	D7	外部接口

## 6.6.2 寄存器描述

### P4\_DATA

#### P4 口数据寄存器

复位值：00<sub>H</sub>

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>
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rwh	<b>P4 口引脚 n 的数据值</b> 0 P4 口引脚 n 的数据值 = 0 （缺省值） 1 P4 口引脚 n 的数据值 = 1

### P4\_DIR

#### P4 口方向寄存器

复位值：00<sub>H</sub>

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>P4 口引脚 n 方向控制</b> 0 设置为输入引脚 （缺省值） 1 设置为输出引脚

### P4\_OD

#### P4 口漏极开路控制寄存器

复位值：00<sub>H</sub>

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



并行端口

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rw	<b>P4 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（缺省值） 1 漏极开路模式；只有 0 可被有效输出

## P4\_PUDSEL

**P4 口上拉 / 下拉选择寄存器**

复位值：FF<sub>H</sub>

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>P4 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）

## P4\_PUDEN

**P4 口上拉 / 下拉使能寄存器**

复位值：04<sub>H</sub>

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>P4 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（缺省值）

并行端口

**P4\_ALTSELn(n = 0 - 1)**
**P4 口复用功能选择寄存器**

 复位值: **00<sub>H</sub>**

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>引脚输出功能</b> 配置 P4_ALTSEL0.Pn 和 P4_ALTSEL1.Pn 选择引脚用作 GPIO 或复用功能: 00 GPIO 功能 (缺省值) 10 复用输出功能 1 01 复用输出功能 2 11 复用输出功能 3

**P4\_DS**
**P4 口驱动能力控制寄存器**

 复位值: **FF<sub>H</sub>**

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>P4 口引脚 n 的驱动能力控制</b> 0 弱驱动 1 强驱动 (缺省值)

## 6.7 P5 口

P5 口是一个 8 位通用双向口。

P5 口寄存器归纳见[表 6-11](#)。

**表 6-11 P5 口寄存器**

寄存器缩写名	寄存器全名
P5_DATA	P5 口数据寄存器
P5_DIR	P5 口方向寄存器
P5_OD	P5 口漏极开路控制寄存器
P5_PUDSEL	P5 口上拉 / 下拉选择寄存器
P5_PUDEN	P5 口上拉 / 下拉使能寄存器
P5_ALTSEL0	P5 口复用功能选择寄存器 0
P5_ALTSEL1	P5 口复用功能选择寄存器 1
P5_DS	P5 口驱动能力控制寄存器

### 6.7.1 功能

P5 口输入和输出功能归纳见[表 6-12](#)。

由寄存器 P5\_ALTSEL0 和 P5\_ALTSEL1 选择复用输出功能。

**表 6-12 P5 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
P5.0	输入	GPI	P5_DATA.P0	—
		ALT1	—	—
		ALT2	EXINT1_1	外部中断 1
		ALT3	—	—
	输出	GPO	P5_DATA.P0	—
		ALT1	—	—
		ALT2	—	—
		ALT3	—	—
		EXT_INT	A0	外部接口

**表 6-12 P5 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
P5.1	输入	GPI	P5_DATA.P1	—
		ALT1	—	—
		ALT2	EXINT2_1	外部中断 2
		ALT3	—	—
	输出	GPO	P5_DATA.P1	—
		ALT1	—	—
		ALT2	—	—
		ALT3	—	—
		EXT_INT	A1	外部接口
P5.2	输入	GPI	P5_DATA.P2	—
		ALT1	RXD_2	UART
		ALT2	T2CC2_2/EXINT5_3	T2CCU/ 外部中断 5
		ALT3	—	—
	输出	GPO	P5_DATA.P2	—
		ALT1	T2CC2_2	T2CCU
		ALT2	—	—
		ALT3	—	—
P5.3	输入	EXT_INT	A2	外部接口
		GPI	P5_DATA.P3	—
		ALT1	CCPOS0_0	CCU6
		ALT2	EXINT1_0	外部中断 1
		ALT3	T12HR_2	CCU6
	输出	ALT4	CC61_3	CCU6
		GPO	P5_DATA.P3	—
		ALT1	T2CC5_2	T2CCU
		ALT2	TXD_2	UART
		ALT3	—	—
		EXT_INT	A3	外部接口

表 6-12 P5 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
P5.4	输入	GPI	P5_DATA.P4	–
		ALT1	CCPOS1_0	CCU6
		ALT2	EXINT2_0	外部中断 2
		ALT3	T13HR_2	CCU6
		ALT4	CC62_3	CCU6
	输出	GPO	P5_DATA.P4	–
		ALT1	T2CC4_2	T2CCU
		ALT2	RXDO_2	UART
		ALT3	–	–
		EXT_INT	A4	外部接口
P5.5	输入	GPI	P5_DATA.P5	–
		ALT1	CCPOS2_0	CCU6
		ALT2	T2CC0_2/EXINT3_3	T2CCU/ 外部中断 3
		ALT3	CTRAP_1	CCU6
		ALT4	CC60_3	CCU6
	输出	GPO	P5_DATA.P5	–
		ALT1	TDO_1	JTAG
		ALT2	TXD1_2	UART1
		ALT3	T2CC0_2	T2CCU
		EXT_INT	A5	外部接口
P5.6	输入	GPI	P5_DATA.P6	–
		ALT1	TCK_1	JTAG
		ALT2	T2CC1_2/EXINT4_3	T2CCU/ 外部中断 4
		ALT3	–	–
	输出	GPO	P5_DATA.P6	–
		ALT1	–	–
		ALT2	RXDO1_2	UART1
		ALT3	T2CC1_2	T2CCU
		EXT_INT	A6	外部接口

表 6-12 P5 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
P5.7	输入	GPI	P5_DATA.P7	—
		ALT1	TDI_1	JTAG
		ALT2	RXD1_2	UART1
		ALT3	T2CC3_2/EXINT6_4	T2CCU/ 外部中断 6
	输出	GPO	P5_DATA.P7	—
		ALT1	T2CC3_2	T2CCU
		ALT2	—	—
		ALT3	—	—
		EXT_INT	A7	外部接口

## 6.7.2 寄存器描述

### P5\_DATA

#### P5 口数据寄存器

复位值：00<sub>H</sub>

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>P5 口引脚 n 的数据值</b> 0 P5 口引脚 n 的数据值 = 0 （缺省值） 1 P5 口引脚 n 的数据值 = 1

### P5\_DIR

#### P5 口方向寄存器

复位值：00<sub>H</sub>

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>P5 口引脚 n 方向控制</b> 0 设置为输入引脚 （缺省值） 1 设置为输出引脚

### P5\_OD

#### P5 口漏极开路控制寄存器

复位值：00<sub>H</sub>

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

并行端口

符号	位	类型	描述
<b>Pn</b> (n = 0 - 7)	n	rw	<b>P5 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（缺省值） 1 漏极开路模式；只有 0 可被有效输出

**P5\_PUDSEL**
**P5 口上拉 / 下拉选择寄存器**

 复位值：FF<sub>H</sub>

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>P5 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）

**P5\_PUDEN**
**P5 口上拉 / 下拉使能寄存器**

 复位值：FF<sub>H</sub>

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>P5 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（缺省值）



并行端口

**P5\_ALTSEL<sub>n</sub>(n = 0 - 1)**
**P5 口复用功能选择寄存器**

 复位值: **00<sub>H</sub>**

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>P<sub>n</sub></b> <b>(n = 0 - 7)</b>	n	rw	<b>引脚输出功能</b> 配置 P5_ALTSEL0.P <sub>n</sub> 和 P5_ALTSEL1.P <sub>n</sub> 选择引脚用作 GPIO 或复用功能: 00 GPIO 功能 (缺省值) 10 复用输出功能 1 01 复用输出功能 2 11 复用输出功能 3

**P5\_DS**
**P5 口驱动能力控制寄存器**

 复位值: **FF<sub>H</sub>**

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>P<sub>n</sub></b> <b>(n = 0 - 7)</b>	n	rw	<b>P5 口引脚 n 的驱动能力控制</b> 0 弱驱动 1 强驱动 (缺省值)

## 7 电源、复位和时钟管理

XC878 提供了一系列在紧急条件下（如，欠压）保证系统安全性能的特性。

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以被分别关闭，为不同的省电模式降低功耗。

XC878 时钟系统的核心是时钟产生单元（CGU），利用锁相环（PLL）和振荡器（OSC）单元产生主时钟频率。从主时钟得到的相位同步时钟信号被分配到整个系统中。时钟分频因子可编程设置，从而将主频降至更低的频率以减少功耗。

### 7.1 内嵌电压调节器的电源系统

XC878 微控制器需要两种不同电平电源：

- 嵌入式电压调节器（EVR）和端口需 3.3V 或 5.0V 供电
- 内核、存储器、片内振荡器和外设需 2.5V 供电

XC878 电源系统如图 7-1 所示。由外部电源引脚提供 3.3V 或 5.0V 电源；由 EVR 产生 2.5V 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。

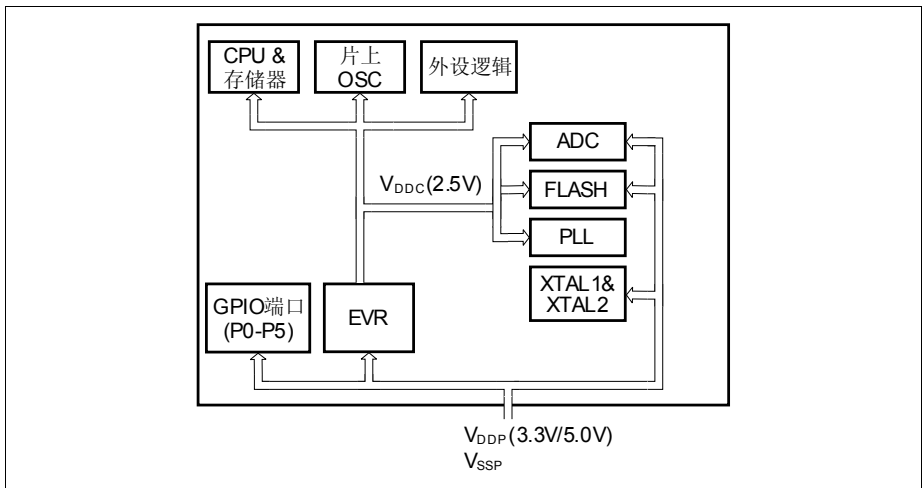


图 7-1 XC878 电源系统

#### EVR 特性：

- 输入电压 ( $V_{DDP}$ )：3.3V/5.0V
- 输出电压 ( $V_{DDC}$ )：2.5V  $\pm$  7.5%
- 掉电模式下提供低功率电压调节器
- $V_{DDP}$  预警检测

---

电源、复位和时钟管理

- $V_{DDC}$  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，在低功耗模式下为系统供电。

EVR 包含  $V_{DDC}$  和  $V_{DDP}$  检测器。 $V_{DDC}$  检测器有一个用于欠压检测的阈值电压（1.8V）。当  $V_{DDC}$  低于 1.8V，欠压复位被激活，微控制器进入复位状态。在正常工作模式和掉电模式下均可使能欠压检测。

如果外接电源为 5.0V， $V_{DDP}$  则有一个 4.0 V 的预警阈值电压。当  $V_{DDP}$  低于 4.0 V 时， $V_{DDP}$  的 NMI 中断标志位  $NMISR.FNMIVDDP$  被置位，如果  $V_{DDP}$  的 NMI 中断被使能（ $NMICON.NMIVDDP$ ），将向 CPU 发送 NMI 中断请求。如果外接电源为 3.3V， $V_{DDP}$  检测器关闭。在掉电模式下，除欠压检测器之外，所有检测器均关闭。

EVR 还有一个  $V_{DDC}$  上电复位（POR）检测器，以保证系统正确上电。POR 的检测电压为 1.8V。在正常工作模式和掉电模式下均采用监控功能。上电过程中， $V_{DDC}$  超过 1.8V 后，EVR 的复位通常需再延迟一段时间，典型值为 300  $\mu$ s。

## 7.2 复位控制

XC878 有五种复位方式：上电复位、硬件复位、看门狗定时器复位，掉电唤醒复位和欠压复位。

XC878 首次上电时，必须定义某些引脚（见**表 7-2**）的状态以使器件能够正确启动。复位过程结束时，引脚采样值被锁存以选择进入期望的启动方式，在下次上电复位或硬件复位之前该值不能被修改，从而保证了器件正常工作时状态稳定。

正常工作模式或掉电模式下可以硬件复位。复位输入引脚 **RESET** 用于硬件复位。

检测到系统出现故障时，看门狗定时器（WDT）也能复位器件。

另一种需被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。

$V_{DDC}$  电压跌至 1.8V 以下时触发欠压复位。

### 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

XC878 电源的典型连接如**图 7-2**所示。对于  $I_{DD\max} = 100\text{mA}$  的电压调节器， $V_{DDP}$  的电容值为 10  $\mu\text{F}$ 。 $V_{DDC}$  的最大电容值为 220 nF。

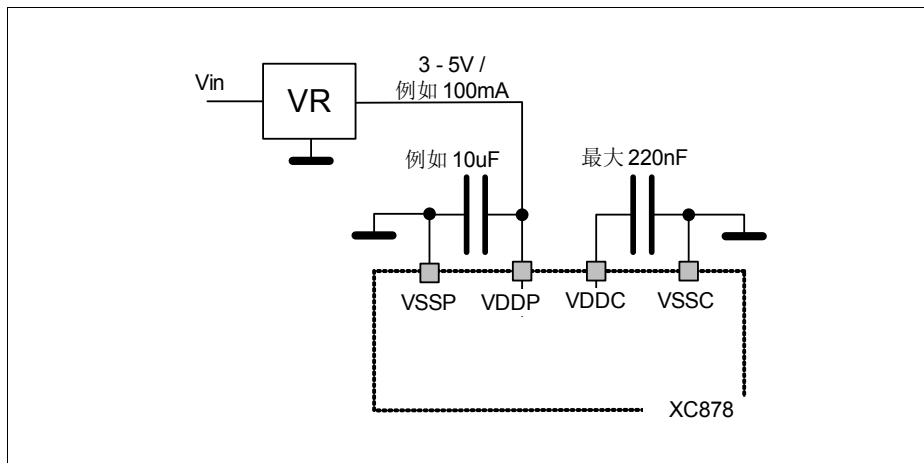


图 7-2 电源电路示例

## 电源、复位和时钟管理

系统启动时，PLL 输出被关闭。一旦 EVR 稳定之后，假如振荡器工作，PLL 与其连接，紧接着进行锁相检测、以确保 PLL 开始正常工作。接下来，等到系统时钟一稳定，则 CPU 复位信号被释放并开始执行用户程序。

复位操作将锁存引脚 MBC、TMS 和 P0.0 的状态值，该值用来选择启动模式（见[章节 7.2.3](#)）。正确的复位将引导系统进入指定状态。从地址 0000<sub>H</sub> 开始执行程序。

上电复位过程如[图 7-3](#) 所示。

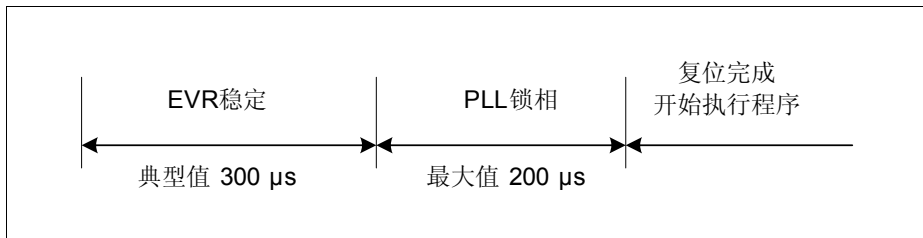


图 7-3 上电复位

### 7.2.1.2 硬件复位

当复位输入引脚 **RESET** 被拉低后，开始执行外部硬件复位过程。**RESET** 信号解除后，复位过程和上电复位过程相同，如[图 7-3](#) 所示。通过 **RESET** 引脚进行硬件复位将终止空闲模式或掉电模式。

复位操作将锁存引脚 MBC、TMS 和 P0.0 的状态值，该状态值用于选择启动模式（参见[章节 7.2.3](#)）。

### 7.2.1.3 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 **PMCON0.WDTRST** 用来指示 WDT 的复位状态。

对于 WDT 复位，因为 EVR 和系统时钟已稳定，因此 WDT 复位比其它类型复位所需的时间短。

### 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，XC878 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 XC878 处于掉电模式，有三种唤醒方式供选择：

- 通过 RXD 唤醒
- 通过 EXINT0 唤醒
- 唤醒 RXD 或 EXINT0 唤醒

---

## 电源、复位和时钟管理

由控制位 `PMCON0.WS` 来选择唤醒方式。掉电唤醒可以带复位或不带复位，由 `PMCON0.WKSEL` 选择。位 `PMCON0.WKRS` 指示唤醒状态（带复位或无复位）。

一旦 XC878 被唤醒，掉电唤醒复位过程和上电复位过程相似。

除以上三种唤醒方式之外，还可以通过 `RESET` 引脚硬件复位退出掉电模式。

### 7.2.1.5 欠压复位

在正常工作模式和掉电模式下，当内核供电电压  $V_{DDC}$  跌至阈值电压  $V_{DDC\_TH}$  (1.8V) 以下时，EVR 中的  $V_{DDC}$  检测器侦测到欠压，欠压将引起器件复位。一旦发生欠压复位，复位过程和上电复位过程相同，如图 7-3 所示。电压下降过程中所有端口处于三态。

## 7.2.2 模块复位行为

**表 7-1** 给出不同复位类型对 XC878 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 复位对器件功能的影响**

模块 / 功能	唤醒复位	WDT 复位	硬件复位	上电复位	压降复位
<b>CPU 核</b>	■	■	■	■	■
<b>外设</b>	■	■	■	■	■
<b>片内静态 RAM</b>	不受影响, 可靠	不受影响, 可靠	不受影响, 可靠	受影响, 不可靠	受影响, 不可靠
<b>OSC, PLL</b>	■	不受影响	■	■	■
<b>端口引脚</b>	■	■	■	■	■
<b>EVR</b>	电压调节器开启	不受影响	不受影响	■	■
<b>FLASH</b>	■	■	■	■	■
<b>NMI</b>	■	■	■	■	■

### 7.2.3 启动方案

XC878 复位时，（一旦复位过程结束）系统必须识别配置类型，根据配置类型启动不同的工作模式。因此，激活特定模式和状态所需的启动信息由外部输入引脚配置。上电复位或硬件复位后，由引脚 MBC、TMC 和 P0.0 共同选择不同的启动模式。XC878 系统可用的启动模式归纳见表 7-2。

表 7-2 XC878 启动选择 <sup>1)</sup>

MBC	TMS	P0.0	模式类型	PC 初始值
1	0	x	用户模式；片内 OSC/PLL 未旁路（正常）	0000 <sub>H</sub>
1	1	x	用户（JTAG）模式 <sup>2)</sup> ；片内 OSC/PLL 未旁路（正常）	0000 <sub>H</sub>
0	0	x	BSL 模式（LIN 模式） <sup>3)</sup> ，UART/MultiCAN 模式 <sup>4)5)</sup> 和其它 BSL 模式 <sup>6)</sup> ；片内 OSC/PLL 未旁路	0000 <sub>H</sub>
0	1	0	OCDS 模式；片内 OSC/PLL 未旁路（正常）	0000 <sub>H</sub>

- <sup>1)</sup> 进行启动选择时，除引脚 MBC、TMS 和 P0.0 外，TM 引脚也需要外部下拉。
- <sup>2)</sup> 带标准 JTAG 引脚（TCK、TDI、TDO）用于热插拔的正常用户模式。
- <sup>3)</sup> 若器件编程为 LIN 模式，始终采用 LIN BSL 而不采用 UART/MultiCAN。
- <sup>4)</sup> 对于带有 MultiCAN 的产品，由固件（基于协议）对 UART 或 MultiCAN BSL 进行译码。若产品不带 MultiCAN 和 LIN，则使用 UART BSL。
- <sup>5)</sup> MultiCAN BSL 模式下，固件将时钟源切换至 XTAL，片上振荡器被旁路。采用此方式避免了片上振荡器频率不变的特性，允许其它的频率时钟输入，因此确保实现精确的波特率检测（尤其对于高数据率的应用来讲）。
- <sup>6)</sup> 其它 BSL 为用户定义的 BSL 模式。用户 BSL 代码存放在 Flash 存储器中。如果 AltBSL 密码有效，则进入该引导加载模式。

注：启动选择只能使用 UART 和 JTAG 引脚的缺省设置。



## 7.2.4 寄存器描述

表 7-3 寄存器 PMCON0 的复位值

复位源	复位值
上电复位 / 硬件复位 / 压降复位	1000 0000 <sub>B</sub>
WDT 复位	1100 0000 <sub>B</sub>
掉电唤醒复位	1010 0000 <sub>B</sub>

表 7-4 寄存器 CR\_MISC 的复位值

复位源	复位值
上电复位 / 压降复位	0000 0000 <sub>B</sub>
硬件复位	0000 0001 <sub>B</sub>

### PMCON0

功率模式控制寄存器

复位值：见表 7-3

7	6	5	4	3	2	1	0
<b>VDDP WARN</b>	<b>WDTRST</b>	<b>WKRS</b>	<b>WKSEL</b>	<b>SD</b>	<b>PD</b>	<b>WS</b>	
rh	rwh	rwh	rw	rw	rwh	rw	

符号	位	类型	描述
<b>WS</b>	[1:0]	rw	<b>唤醒源选择</b> 00 无唤醒源 01 选择唤醒源 RXD（下降沿触发） 10 选择唤醒源 EXINT0（下降沿触发） 11 选择唤醒源 RXD（下降沿触发）或 EXINT0（下降沿触发）
<b>WKSEL</b>	4	rw	<b>唤醒复位选择位</b> 0 无复位唤醒 1 带复位唤醒
<b>WKRS</b>	5	rwh	<b>唤醒指示位</b> 0 未发生唤醒 1 已唤醒 该位只能由硬件置位，软件复位。

## 电源、复位和时钟管理

符号	位	类型	描述
<b>WDTRST</b>	6	rwh	<b>WDT 复位指示位</b> 0 未发生 WDT 复位 1 WDT 已复位 该位只能由硬件置位，软件复位。
<b>VDDPWARN</b>	7	rh	<b>V<sub>DDP</sub> 预警阈值指示位</b> 0 低于 V <sub>DDP</sub> 预警阈值电压 1 高于 V <sub>DDP</sub> 预警阈值电压

## CR\_MISC

时钟和复位其它控制寄存器

复位值：见表 7-4

7	6	5	4	3	2	1	0
<b>CCCFCG</b>	<b>MDUCCFCG</b>	<b>CCUCCFCG</b>	<b>T2CCFCG</b>		<b>0</b>		<b>HDRST</b>
rw	rw	rw	rw		r		rwh

符号	位	类型	描述
<b>HDRST</b>	7	rwh	<b>硬件复位指示位</b> 0 未发生硬件复位 1 已发生硬件复位 该位只能由硬件置位、软件复位。
<b>0</b>	[3:1]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 7.3 时钟系统

XC878 的时钟系统执行以下功能：

- 获取并缓冲输入的时钟信号，产生主时钟
- 将相位同步的时钟信号分配到整个系统中
- 系统主时钟分频（到较低频率）用于省电模式

### 7.3.1 时钟产生单元

XC878 系统的时钟产生单元（CGU）由振荡器电路和锁相环（PLL）构成。XC878 的振荡器可以是片内振荡器（4 MHz）、也可以是片外振荡器（2 MHz -20 MHz）。如果不特别指明，本手册中的“振荡器”统指片内和片外振荡器。复位后，缺省使用片内振荡器；可由软件设定选择片外振荡器。PLL 可将来自振荡器电路的低频外部时钟信号转换成高速内部时钟，使系统发挥最大性能。

CGU 框图如图 7-4 所示。

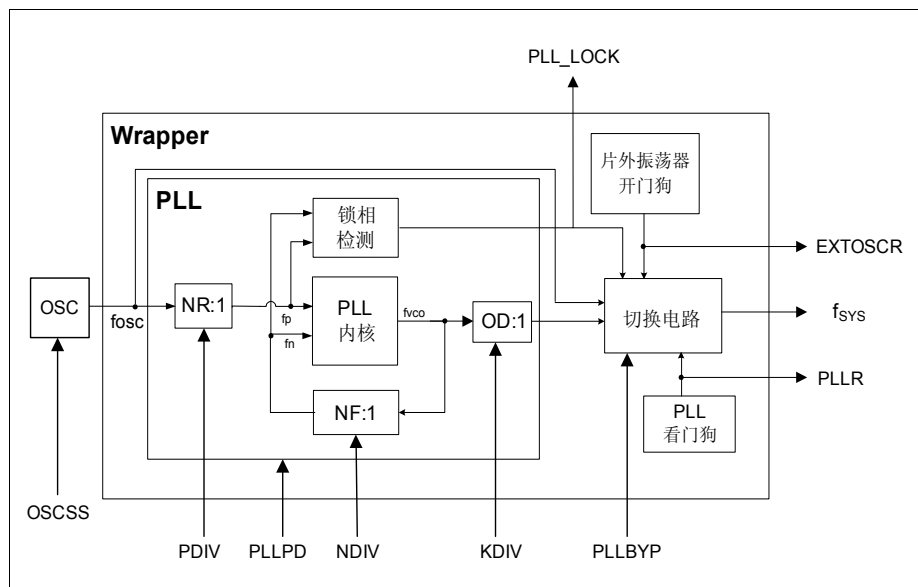


图 7-4 CGU 框图

### 7.3.1.1 功能描述

系统上电时，PLL 输出被关闭。EVR 稳定之后，假如振荡器工作，PLL 与其连接，接下来进行的锁相检测将确保 PLL 开始正常工作。一旦复位结束，位 LOCK 将被置 1 以指示 PLL 已锁相。

#### 失锁操作

出现以下某种或多种情况时，可能发生失锁：

- PLL 与振荡器失锁；或
- PLL 输出频率低于 25 MHz；或
- 选择片外振荡器作为 PLL 的输入时钟源时，片外振荡器的频率低于 1MHz；或
- 设置 OSC\_CON.PLLBYP = 1 和 OSC\_CON.OSCSS = 1 选择片外振荡器频率作为系统频率时，片外振荡器频率低于 1MHz

在失锁阶段，可令以下某个信号无效：

- PLL 锁相状态，PLL\_CON.PLL\_LOCK
- PLL 运行状态，PLL\_CON.PLLR
- 片外振荡器运行状态，OSC\_CON.EXTOSCR

如果 PLL 与 OSC 失锁，置位 PLL 失锁 NMI 中断标志 NMISR.FNMIPLL。若 PLL NMI 被使能 (NMICON.NMIPLL)，将向 CPU 发送 NMI 中断请求。此外，可根据应用的不同分别复位 PLL\_LOCK 标志、PLL\_R 标志和 EXTOSCR 标志。PLL 失锁时，系统时钟自动切换为旁路 PLL 输出、使用片内振荡器作为时钟源。可以利用片内振荡器时钟执行紧急程序。

在下次上电复位、硬件复位或成功执行锁相恢复之前，XC878 始终保持失锁状态。若失锁恢复失败，必须采取紧急措施，例如可由用户关闭系统。

检测到 PLL 失锁时，若 PLL 不是系统时钟源 (PLL\_BYP = 1) 或片外振荡器不用作系统时钟，则只复位锁相标志 (PLL\_LOCK = 0)，无其它操作，从而允许动态切换 PLL 的参数。

#### 失锁恢复

若 PLL 与 OSC 失锁，则自动切换至使用片内振荡器 (4 MHz) 作为系统时钟源。必须检查 EXTOSCR、PLL\_R 和 PLL\_LOCK 标志，只有当这些标志都置位时才能继续正常工作。为了确保成功执行失锁恢复，必须执行下列步骤。

当选择片外振荡器作为 PLL 的输入时钟源时：

1. 置位 EORDRES 重新启动片外振荡器看门狗
  2. 置位 PLLRDRES 重新启动 PLL 看门狗
  3. 旁路 PLL 输出 (PLL\_BYP = 1)
  4. 选择 PLL 掉电模式 (PLL\_PD = 1)
  5. 选择片内振荡器作为振荡时钟源 (OSC\_SS = 0)
  6. 等待 65 个片内振荡器时钟周期
- 若在 65 个片内振荡器时钟周期后 OSC\_CON.EXTOSCR 被置位，则

1. 选择片外振荡器作为振荡时钟源（ $OSCSS = 1$ ）
2. 重新编程  $NDIV$ 、 $PDIV$  和  $KDIV$
3. 切换到 PLL 正常工作模式（ $PLLPD = 0$ ）
4. 等待直到  $PLL\_LOCK$  和  $PLL_R$  标志被置位
5. 禁止 PLL 输出旁路（ $PLLBYP = 0$ ），继续正常工作

当选择片内振荡器作为 PLL 的输入时钟源时：

1. 置位  $PLLRDRES$  重新启动 PLL 看门狗
2. 旁路 PLL 输出（ $PLLBYP = 1$ ）
3. 选择 PLL 掉电模式（ $PLLPD = 1$ ）
4. 重新编程  $NDIV$ 、 $PDIV$  和  $KDIV$
5. 切换到 PLL 正常工作模式（ $PLLPD = 0$ ）
6. 等待直到  $PLL\_LOCK$  和  $PLL_R$  标志被置位
7. 禁止 PLL 输出旁路（ $PLLBYP = 0$ ），继续正常工作

当选择片外振荡器作为系统时钟时：

1. 置位  $EORDRES$  重新启动片外振荡器看门狗
2. 选择片内振荡器作为振荡时钟源（ $OSCSS = 0$ ）
3. 等待 65 个片内振荡器时钟周期
  - 若在 65 个片内振荡器时钟周期后  $EXTOSCR$  被置位，则
1. 选择片外振荡器作为振荡时钟源（ $OSCSS = 1$ ）

## 改变 PLL 的参数

要改变 PLL 的参数，必须执行下列步骤：

1. 旁路 PLL 输出（ $PLLBYP = 1$ ）
2. 选择 PLL 掉电模式（ $PLLPD = 1$ ）
3. 编程  $NDIV$ 、 $PDIV$  和  $KDIV$
4. 切换到 PLL 正常工作模式（ $PLLPD = 0$ ）
5. 置位  $PLLRDRES$  重新启动 PLL 看门狗
6. 等待直到  $PLL\_LOCK$  和  $PLL_R$  标志被置位
7. 禁止 PLL 输出旁路（ $PLLBYP = 0$ ）

## 选择片外振荡器作为 PLL 的输入时钟源

选择片外振荡器作为 PLL 的输入时钟源时，必须执行下列步骤：

1. 选择片内振荡器用作系统时钟（ $OSCSS = 0$ ）
2. 旁路 PLL 输出（ $PLLBYP = 1$ ）
3. 选择 PLL 掉电模式（ $PLLPD = 1$ ）
4. 片外振荡器上电（ $XPDI = 0$ ）
5. 等待 1.5ms 直至片外振荡器稳定（应根据不同片外振荡器相应调整延迟时间）
6. 置位  $EORDRES$  重新启动片外振荡器看门狗
7. 等待 65 个片内振荡器时钟周期
  - 若 65 个片内振荡器时钟周期之后  $EXTOSCR$  置位，则：

8. 置位 OSCSS 选择片外振荡器作为振荡时钟源，外部振荡器用作系统时钟
9. 编程 NDIV、PDIV 和 KDIV
10. 切换到 PLL 正常工作模式（PLLDP = 0）
11. 置位 PLLRDRES 重新启动 PLL 看门狗
12. 等待直到 PLL\_LOCK 和 PLLR 标志被置位
13. 禁止 PLL 输出旁路（PLLBYP = 0）

### 使用直接驱动（PLL 旁路）模式，选择片外振荡器作为系统时钟

选择片外振荡器作为系统时钟时，必须执行下列步骤：

1. 选择片内振荡器用作系统时钟（OSCSS = 0）
2. 旁路 PLL 输出（PLLBYP = 1）
3. 选择 PLL 掉电模式（PLLDP = 1）
4. 片外振荡器上电（XPD = 0）
5. 等待 1.5ms 直至片外振荡器稳定（应根据不同片外振荡器相应调整延迟时间）
6. 置位 EORDRES 重新启动片外振荡器看门狗
7. 等待 65 个片内振荡器时钟周期
  - 若 65 个片内振荡器时钟周期之后 EXTOSCR 置位，则：
8. 置位 OSCSS 选择片外振荡器作为振荡时钟源，外部振荡器用作系统时钟

使用片内振荡器时，为了尽可能减低功耗，可置位 XPD 关闭片外振荡器电路。

### 7.3.2 时钟源控制

时钟系统提供了两种方式产生系统时钟：

#### 直接驱动（PLL 旁路）

PLL 旁路时，系统时钟频率等于片外时钟频率。

(7.1)

$$f_{\text{SYS}} = f_{\text{OSC}}$$

#### PLL 模式

CPU 时钟等于振荡器时钟除以因子 NR（PDIV）、乘以 NF（NDIV），再除以 OD 因子（KDIV）。该模式下，PLL 输出一定不能被旁路。系统正常工作时使用 PLL 模式。

(7.2)

$$f_{\text{SYS}} = f_{\text{OSC}} \times \frac{\text{NF}}{\text{NR} \times \text{OD}}$$

正常运行时，系统工作在 PLL 模式。

对于 XC878 系统，要获得所需的系统频率  $f_{\text{sys}}$ ，可通过位 NDIV、PDIV 和 KDIV 分别选择 NF、NR 和 OD 的值。但选择这些参数时必须满足以下条件：

- $100 \text{ MHz} < f_{\text{VCO}} < 175 \text{ MHz}$
- $800 \text{ kHz} < f_{\text{OSC}} / (2 * \text{NR}) < 8 \text{ MHz}$

表 7-5 举例说明在不同的振荡器输入下，如何选择参数以获得144 MHz的典型系统频率。

表 7-5            系统频率 ( $f_{sys} = 144 \text{ MHz}$ )

振荡器	fosc	N	P	K	fsys
片内	4 MHz	72	2	1	144 MHz
片外	8 MHz	72	4	1	144 MHz
	6 MHz	72	3	1	144 MHz
	4 MHz	72	2	1	144 MHz

### 7.3.3        时钟管理

时钟管理子模块从基本时钟产生系统需要的所有时钟信号，包括：

- 产生系统内的所有时钟
- 基本的时钟降频电路
- 时钟使能 / 禁止集中控制电路

从系统频率 $f_{sys}$ 产生所有时钟如图 7-5所示。正常工作模式下，不同模块的典型频率如下：

- CPU 时钟：CCLK、SCLK = 24 MHz
- 快速时钟：FCLK = 48 MHz
- 外设时钟：PCLK = 24 MHz
- Flash 接口时钟：CCLK2 = 96 MHz 和 CCLK = 24 MHz

XC878 中，MultiCAN、MDU、CORDIC、CCU6 和 T2CCU 这五个模块的工作频率可以为 24 MHz 或 48 MHz，由寄存器 CMCON 和 CR\_MISC 进行选择。

此外，引脚 P0.0 或 P0.7 还可用作时钟输出（CLKOUT）。如果位 COUTS = 00，输出时钟来自片内振荡器输出频率；如果位 COUTS = 01，由位域 COREL 选择时钟输出频率。在该选择下，可通过翻转锁存（位 TLEN 设置为 1）将输出时钟频率进一步 2 分频，使得输出频率的占空比为 50%。如果位 COUTS = 10，选择 PCLK 时钟。

空闲模式下，只有 CPU 时钟 CCLK 被禁止；掉电模式下，CCLK、SCLK、FCLK、CCLK2 和 PCLK 全部被禁止；若使能低速模式，CPU 及外设时钟经分频处理，分频因子可由位域 CMCON.CLKREL 设定。



## 电源、复位和时钟管理

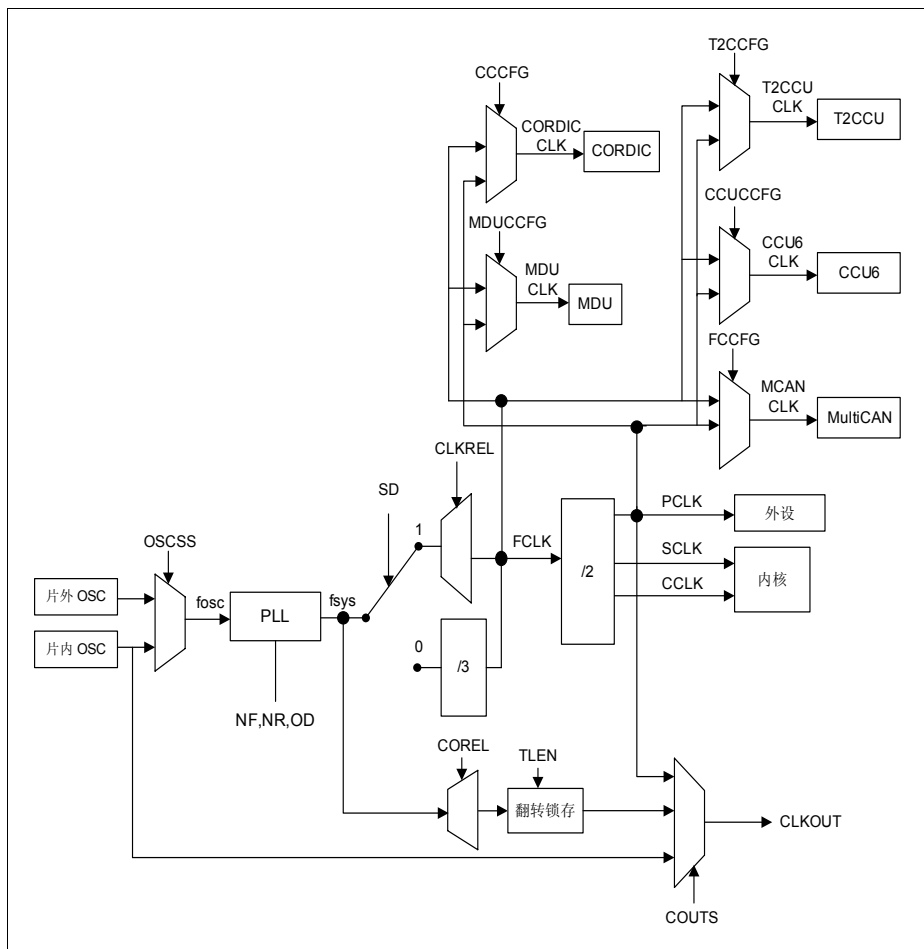


图 7-5 由  $f_{sys}$  产生时钟

### 7.3.4 寄存器描述

#### OSC\_CON

#### OSC 控制寄存器

复位值：0X00 1000<sub>B</sub>

7	6	5	4	3	2	1	0
PLL RDRES	PLLBYP	PLLPD	0	XPD	OSCSS	EORDRES	EXTOSCR
rwh	rwh	rw	r	rw	rwh	rwh	rh

符号	位	类型	描述
EXTOSCR	0	rh	<b>片外振荡器运行状态位</b> 该位指示片外振荡器看门狗的状态。 0 片外振荡器未工作 1 片外振荡器正在工作
EORDRES	1	rwh	<b>片外振荡器看门狗复位</b> 0 无操作 1 复位并重启片外振荡器看门狗 该位将自动复位为 0。
OSCSS	2	rwh	<b>振荡器选择</b> 0 选择片内 OSC 1 选择片外 OSC 该位为受保护位，复位后根据启动配置由硬件更新。 PLL 失锁期间切换到片内振荡器工作时，该位不被硬件修改，仍保留原先的设置。 <i>注： 为了顺利切换时钟源，从片内振荡器切换至片外振荡器之前，片外振荡器的时钟输出必须保持稳定。</i>
XPD	3	rw	<b>XTAL 掉电控制</b> 0 XTAL 未掉电 1 XTAL 掉电 该位为受保护位。 <i>注： 在置位 OSCSS 之前，必须确保 XPD 为 0。</i>
PLLPD	5	rw	<b>PLL 掉电控制</b> 0 PLL 未掉电 1 PLL 掉电 该位为受保护位。

## 电源、复位和时钟管理

符号	位	类型	描述
<b>PLLBYN</b>	6	rwh	<b>PLL 输出旁路控制</b> 0 PLL 输出不被旁路 1 PLL 输出被旁路 复位后根据启动配置由硬件更新该位。
<b>PLLPD</b>	7	rw	<b>PLL 看门狗复位</b> 0 无操作 1 复位并重启 PLL 看门狗 该位将自动复位为 0。
<b>0</b>	4	r	<b>保留</b> 读操作返回 0，应写入 0。

**PLL\_CON**
**PLL 控制寄存器**

复位值：0001 1000<sub>B</sub>

7	6	5	4	3	2	1	0
NDIV						PLL <sub>R</sub>	PLL <sub>LOCK</sub>
rw						rh	rh

符号	位	类型	描述
<b>PLL_LOCK</b>	0	rh	<b>PLL 锁相状态标志</b> 0 PLL 相位未锁定 1 PLL 相位锁定
<b>PLL<sub>R</sub></b>	1	rh	<b>PLL 运行状态标志</b> 0 PLL 未工作 1 PLL 正在工作
<b>NDIV</b>	[7:2]	rw	<b>PLL NF 分频（连同 PLL_CON1[7:5]）</b> 00000000 NF = 2 00000001 NF = 3 00000010 NF = 4 ..... 001000110 NF = 72（缺省值） ..... 111111101 NF = 511 111111110 NF = 512 111111111 NF = 513 NDIV 是受保护位。当保护机制（见 <a href="#">章节 3.4.4.1</a> ）有效时，该位不能被直接写入。

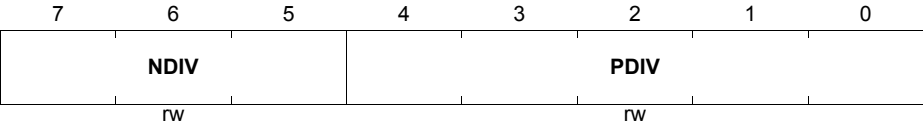
电源、复位和时钟管理

注： 寄存器PLL\_CON的复位值是**0001 1000<sub>B</sub>**。复位后一个时钟周期，如果PLL锁相，LOCK将被置1，于是将会观察到PLL\_CON的值**0001 1001<sub>B</sub>**。

PLL\_CON1

PLL 控制寄存器 1

复位值: 0010 0000<sub>B</sub>

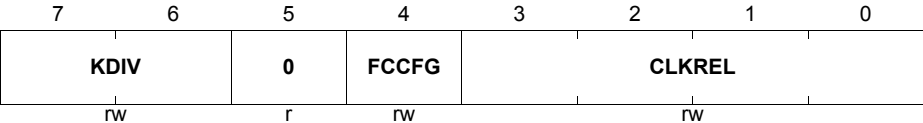


符号	位	类型	描述
PDIV	[4:0]	rw	<b>PLL NR 分频</b> 00000 NR = 2 （缺省值） 00001 NR = 3 00010 NR = 4 ..... 11101 NR = 31 11110 NR = 32 11111 NR = 33 PDIV 是受保护位。当保护机制（见 <a href="#">章节 3.4.4.1</a> ）有效时，该位不能被直接写入。
NDIV	[7:5]	rw	<b>PLL NF 分频控制的 MSB</b> 连同 PLL_CON.NDIV 构成 9 位分频因子。
0	0	r	<b>保留</b> 读操作返回 0，应写入 0。

CMCON

时钟控制寄存器

复位值: 0001 0000<sub>B</sub>



符号	位	类型	描述
<b>CLKREL</b>	[3:0]	rw	<p>时钟分频</p> <p>0000 <math>f_{SYS}/6</math></p> <p>0001 <math>f_{SYS}/12</math></p> <p>0010 <math>f_{SYS}/18</math></p> <p>0011 <math>f_{SYS}/24</math></p> <p>0100 <math>f_{SYS}/36</math></p> <p>0101 <math>f_{SYS}/48</math></p> <p>0110 <math>f_{SYS}/72</math></p> <p>0111 <math>f_{SYS}/96</math></p> <p>1000 <math>f_{SYS}/144</math></p> <p>1001 <math>f_{SYS}/192</math></p> <p>1010 <math>f_{SYS}/288</math></p> <p>1011 <math>f_{SYS}/384</math></p> <p>1100 <math>f_{SYS}/576</math></p> <p>1101 <math>f_{SYS}/768</math></p> <p>1110 <math>f_{SYS}/1152</math></p> <p>注： 只有当 <b>PMCON0.SD</b> 置位时，<b>CLKREL</b> 的改变才生效。</p> <p>注： 以上的时钟分频因子均包含了固定分频因子 2， 见 <a href="#">表 7-5</a>。</p>
<b>FCCFG</b>	4	rw	<p><b>MultiCAN 时钟配置</b></p> <p>0 配置 <math>f_{MCAN}</math> 以 PCLK 频率工作</p> <p>1 配置 <math>f_{MCAN}</math> 以 PCLK 的倍频工作</p>
<b>KDIV</b>	[7:6]	rw	<p><b>PLL OD 分频</b></p> <p>00 OD = 1 （缺省值）</p> <p>01 OD = 2</p> <p>10 OD = 2</p> <p>11 OD = 4</p> <p>KDIV 是被保护位。保护机制（见<a href="#">章节 3.4.4.1</a>）有效时，该位不能被直接写入。</p>
<b>0</b>	5	r	<p><b>保留</b></p> <p>读操作返回 0， 应写入 0。</p>

## COCON

### 时钟输出控制寄存器

复位值：0000 0000<sub>B</sub>

7	6	5	4	3	2	1	0
COUTS		TLEN	0	COREL			
rw		rw	r	rw			

符号	位	类型	描述
COREL	[3:0]	rw	<b>时钟输出分频</b> 0000 $f_{SYS}/3$ 0001 $f_{SYS}/6$ 0010 $f_{SYS}/9$ 0011 $f_{SYS}/12$ 0100 $f_{SYS}/15$ 0101 $f_{SYS}/18$ 0110 $f_{SYS}/21$ 0111 $f_{SYS}/24$ 1000 $f_{SYS}/30$ 1001 $f_{SYS}/36$ 1010 $f_{SYS}/39$ 1011 $f_{SYS}/42$ 1100 $f_{SYS}/48$ 1101 $f_{SYS}/54$ 1110 $f_{SYS}/60$
TLEN	5	rw	<b>翻转锁存使能</b> 只有当 COUTS 设置为 01 <sub>B</sub> 时该位才可用。 0 翻转锁存被禁止。由位域 COREL 选择时钟输出频率 1 翻转锁存被使能。时钟输出频率为位域 COREL 所选择的频率的 1/2，占空比为 50%
COUTS	[7:6]	rw	<b>时钟输出源选</b> 00 选择片内振荡器输出频率 01 由位域 COREL 选择时钟输出频率 10 选择 PCLK 输出频率 11 保留
0	4	r	<b>保留</b> 读操作返回 0，应写入 0。

**电源、复位和时钟管理**

注：在看门狗定时器复位期间，寄存器 **OSC\_CON**、**PLL\_CON**、**PLL\_CON1**、**CMCON** 和 **COCON** 不被复位。

**CR\_MISC**
**时钟和复位其它控制寄存器**
**复位值：0000 000X<sub>B</sub>**

7	6	5	4	3	2	1	0
<b>CCCFG</b>	<b>MDUCCFG</b>	<b>CCUCCFG</b>	<b>T2CCFG</b>	<b>0</b>			<b>HDRST</b>
rw	rw	rw	rw	r			rwh

符号	位	类型	描述
<b>T2CCFG</b>	4	rw	<b>定时器 T2 捕获 / 比较单元时钟配置</b> 0 配置 $f_{T2CCU}$ 以 PCLK 频率工作 1 配置 $f_{T2CCU}$ 以 PCLK 的倍频工作
<b>CCUCCFG</b>	5	rw	<b>CCU6 时钟配置</b> 0 配置 $f_{CCU6}$ 以 PCLK 频率工作 1 配置 $f_{CCU6}$ 以 PCLK 的倍频工作
<b>MDUCCFG</b>	6	rw	<b>MDU 时钟配</b> 0 配置 $f_{MDU}$ 以 PCLK 频率工作 1 配置 $f_{MDU}$ 以 PCLK 的倍频工作
<b>CCCFG</b>	7	rw	<b>CORDIC 时钟配置</b> 0 配置 $f_{CORDIC}$ 以 PCLK 频率工作 1 配置 $f_{CORDIC}$ 以 PCLK 的倍频工作
<b>0</b>	[3:1]	r	<b>保留</b> 读操作返回 0，应写入 0。

## 8 省电模式

XC878 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
- 降低某些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，缺省选择进入有效模式（正常工作模式）（见图 8-1），系统以主频运行。可由软件选择从有效模式进入不同的省电模式，包括：

- 空闲模式
- 低速模式
- 掉电模式

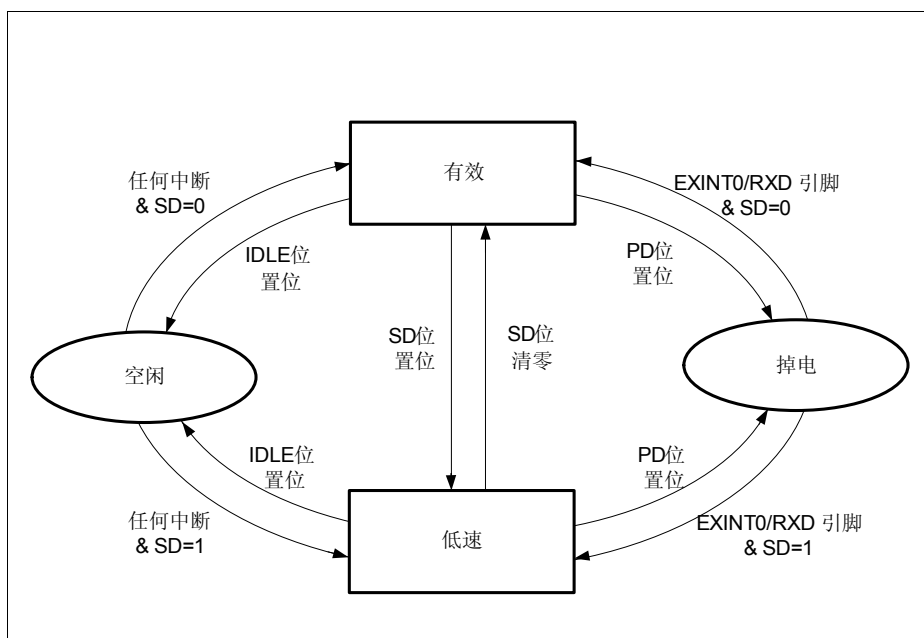


图 8-1 省电模式之间的转换



## 8.1 功能描述

本节描述了不同的省电模式、各模式的操作以及如何进入和退出这些模式。

### 8.1.1 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，从而打断了空闲模式。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，所有其它寄存器均保留进入空闲模式之前的数据；端口引脚保持空闲模式被激活时的逻辑状态。

通过将位 PCON.IDLE 置 1，软件请求进入空闲模式。

下列任何一个条件发生时，系统将返回有效模式：

- 激活任何被使能的中断可结束空闲模式。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。
- 外部硬件复位信号（RESET）被拉低。

### 8.1.2 低速模式

低速模式通过降低器件内部时钟来降低功耗。

SFR PMCON0 中的 SD 置 1 将激活低速模式。位域 CMCON.CLKREL 用来选择不同的低速频率。CPU 和外设均在该低频时钟下工作。对 SD 清零将终止低速模式。

执行以下步骤，低速模式可以和空闲模式组合使用：

1. 置位 PMCON0.SD 激活低速模式
2. 置位 PCON.IDLE 激活空闲模式

有两种方式可结束该组合模式：

- 激活任何被使能的中断可结束空闲模式。CPU 恢复运行，响应中断。执行完 RETI 指令，将执行 IDLE 置 1 指令之后的下一条指令。但低速模式仍保持使能。如果要结束低速模式，必须在相应的中断服务程序中对 SD 清零，或在程序中任何不再需要该模式处对 SD 清零。
- 通过硬件复位也可结束该组合模式。

### 8.1.3 掉电模式

掉电模式下，振荡器和 PLL 均被关闭，包括 Flash 在内的所有存储器均进入待命模式。主电压调节器关闭，但低功率电压调节器继续工作。因此，微控制器的所有功能均被终止，只有 Flash、片内 RAM、XRAM 和 SFR 的内容被保留。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，用户必须注意，掉电模式下端口外部不可悬空，可使用外部上拉/下拉或将端口设置为输出。对于 Flash 存储器，用户必须注意，在进入掉电模式之前要完成 flash 的编程或擦除操作。否则，一旦出现紧急的掉电请求，编程或擦除

## 省电模式

操作可能需要被中止。在这种情况下，当前的操作可能失败。图 8-2 给出在紧急掉电情况下 Flash 操作被中止的流程、此时触发 NMI 中断。

掉电模式下时钟关闭，故不能由中断或 WDT 唤醒；只有接收到外部唤醒信号或复位信号时才能被掉电唤醒。

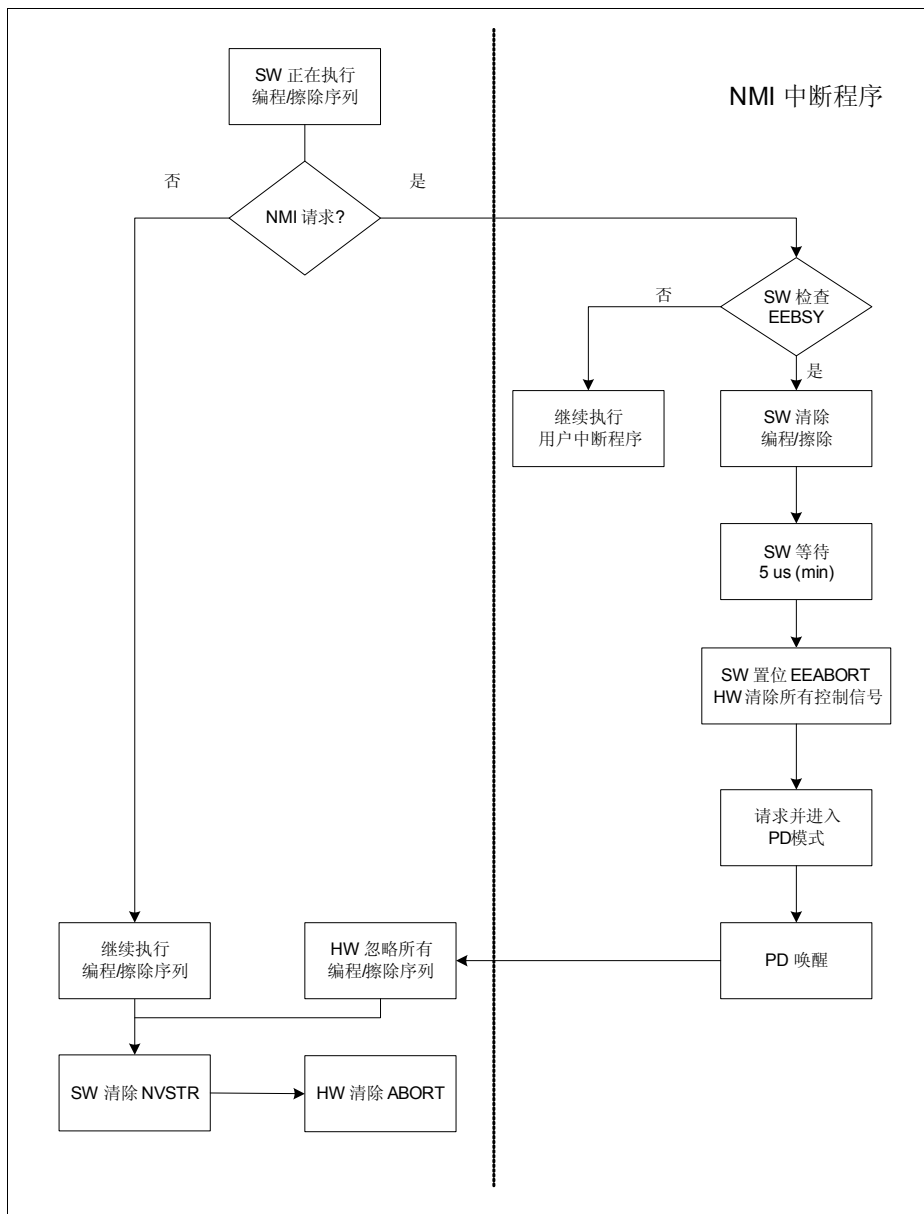


图 8-2 掉电情况下 Flash 操作中止流程

## 进入掉电模式

将 `PMCON0.PD` 置 1，软件请求进入掉电模式。

位 `PMCON0.PD` 置 1 后，必须插入三个 `NOP` 指令，从而可确保从掉电模式唤醒之后第一个指令（三个 `NOP` 指令之后）能够被正确执行。

如果采用外部掉电唤醒，进入掉电模式之前软件必须把 **XC878** 的外部环境准备好，从而在适当条件下触发这些信号。唤醒电路用来检测唤醒信号，激活系统上电。掉电过程中该电路始终保持有效，不依赖于任何时钟。下列引脚的下降沿可触发系统退出掉电模式：

- 引脚 `EXINT0`
- 引脚 `RXD`
- 引脚 `RXD` 或 `EXINT0`

可由寄存器 `PMCON0` 中的位 `WS` 选择唤醒源；由 `PMCON0.WKSEL` 选择带复位唤醒或无复位唤醒。系统进入掉电模式之前必须选定唤醒源和唤醒类型。

## 退出掉电模式

如果通过硬件复位退出掉电模式，器件处于硬件复位状态。

如果系统进入掉电模式前已选定唤醒源和唤醒类型，可由 `EXINT0/RXD` 引脚退出掉电模式。

由位 `MODPISEL.URRIS` 和 `MODPISEL.URRISH` 从三个 `RXD` 输入中选择一个唤醒源；由位 `MODPISEL.EXINT0IS` 从两个 `EXINT0` 输入中选择一个唤醒源。

如果进入掉电模式前 `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。

如果进入掉电模式前 `WKSEL` 已清 0，系统将执行快速唤醒序列。端口引脚继续保持掉电模式下的有效状态，直到执行程序改变引脚值。

无复位掉电唤醒执行下列步骤：

1. 掉电模式下，`EXINT0/RXD` 引脚必须保持高电平。
2. 当 `EXINT0/RXD` 引脚拉低至少保持 100 ns 之后退出掉电模式。
3. 主电压调节器开启，大约需要 300  $\mu$ s 达到稳定。
4. 启动片内振荡器和 PLL（和进入掉电模式之前时钟源的选择无关）。片内振荡器通常需要大约 8  $\mu$ s 达到稳定；检测到片内振荡器时钟达到稳定标称频率后，PLL 在 200  $\mu$ s 内完成锁相。
5. 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。只有在 CPU 恢复运行之后，`EXINT0` 满足产生中断的条件时，才会产生中断。

### 8.1.4 外设时钟管理

功耗降低的程度取决于哪些外设在工作（运行中的外设的个数）。实现某特定功能时，不必要的外设可通过门控电路关闭其输入时钟，禁止该外设工作。例如，在空闲模式下，如果所有定时器被停止、ADC、CCU6、MDU、MultiCAN 和串行接口均不工作，则最大

## 省电模式

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，哪些外设应被停止。

置位寄存器 **PMCON1** 和 **PMCON2** 中的相应控制位，可分别禁止（关闭时钟）外设单元 **ADC**、**SSC**、**CCU6**、**CORDIC**、**MDU**、**MultiCAN**、**UART1**、**T2CCU** 和定时器 **T21**。此外，复位 **GLOBCTR.ANON** 可禁止 **ADC** 的模拟部分。该特性将终止  $f_{\text{ADC}}$  时钟产生，无需进行模数转换时功耗降低。

使用片内振荡器时，为了降低功耗，应当置位 **OSC\_CON.XPD** 使 **XTAL** 掉电。不过，在使用片外振荡器时，不能关闭片内振荡器。

## 8.2 寄存器描述

### PMCON0

#### 功率模式控制寄存器 0

复位值：80<sub>H</sub><sup>1)</sup>

7	6	5	4	3	2	1	0
<b>VDDPWARN</b>	<b>WDRST</b>	<b>WKRS</b>	<b>WKSEL</b>	<b>SD</b>	<b>PD</b>	<b>WS</b>	
rh	rwh	rwh	rw	rw	rwh	rw	

<sup>1)</sup> 看门狗定时器复位对应复位值 C0<sub>H</sub>；掉电唤醒复位对应复位值 A0<sub>H</sub>。

符号	位	类型	描述
<b>WS</b>	[1:0]	rw	<b>唤醒源选择</b> 00 未选择唤醒 01 选择唤醒源 RXD（下降沿触发） 10 选择唤醒源 EXINT0（下降沿触发） 11 选择唤醒源 RXD（下降沿触发）或 EXINT0（下降沿触发）
<b>PD</b>	2	rwh	<b>掉电使能位</b> 该位置位，芯片将进入掉电模式，由唤醒电路来复位芯片。PD 是受保护位。保护机制（见 <a href="#">章节 3.4.4.1</a> ）有效时，该位不能被直接写入。
<b>SD</b>	3	rw	<b>低速模式使能位</b> 该位置位，芯片将进入低速模式，由用户来复位芯片。SD 是受保护位。保护机制有效时，该位不能被直接写入。
<b>WKSEL</b>	4	rw	<b>唤醒复位选择</b> 0 无复位唤醒 1 带复位唤醒
<b>WKRS</b>	5	rwh	<b>唤醒指示位</b> 该位只能由硬件置位，软件复位。 0 未发生唤醒 1 已唤醒

## PCON

功率控制寄存器

复位值：00<sub>H</sub>

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 0 不进入空闲模式 1 进入空闲模式
<b>0</b>	1, [6:4]	r	保留 读操作返回 0，应写入 0。

## MODPSEL

外设输入选择寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>0</b>	<b>URRISH</b>	<b>JTAGTDIS</b>	<b>JTAGTCK</b> <b>S</b>	<b>EXINT2IS</b>	<b>EXINT1IS</b>	<b>EXINT0IS</b>	<b>URRIS</b>
r	rw	rw	rw	rw	rw	r	rw

符号	位	类型	描述
<b>URRISH, URRIS</b>	6, 0	rw	<b>UART 接收输入选择</b> 00 选择 UART 接收输入 RXD_0 01 选择 UART 接收输入 RXD_1 10 选择 UART 接收输入 RXD_2 11 保留
<b>EXINT0IS</b>	1	rw	<b>外部中断 0 输入选择</b> 0 选择外部中断输入 EXINT0_0 1 选择外部中断输入 EXINT0_1
<b>0</b>	7	r	保留 读操作返回 0，应写入 0。

# PMCON1

功率模式控制寄存器 1

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	CDC_DIS	CAN_DIS	MDU_DIS	T2CCU_DIS	CCU_DIS	SSC_DIS	ADC_DIS
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
ADC_DIS	0	rw	<b>ADC 禁止请求位，高有效</b> 0 ADC 正常工作（缺省状态） 1 ADC 被关闭
SSC_DIS	1	rw	<b>SSC 禁止请求位，高有效</b> 0 SSC 正常工作（缺省状态） 1 SSC 被关闭
CCU_DIS	2	rw	<b>CCU 禁止请求位，高有效</b> 0 CCU 正常工作（缺省状态） 1 CCU 被关闭
T2CCU_DIS	3	rw	<b>T2CCU 禁止请求位，高有效</b> 0 T2CCU 正常工作（缺省状态） 1 T2CCU 被关闭
MDU_DIS	4	rw	<b>MDU 禁止请求位，高有效</b> 0 MDU 正常工作（缺省状态） 1 MDU 被关闭
CAN_DIS	5	rw	<b>CAN 禁止请求位，高有效</b> 0 CAN 正常工作（缺省状态） 1 CAN 被关闭
CDC_DIS	6	rw	<b>CORDIC 禁止请求位，高有效</b> 0 CORDIC 正常工作（缺省状态） 1 CORDIC 被关闭
0	7	r	保留 读操作返回 0，应写入 0。



## PMCON2

### 功率模式控制寄存器 2

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0						UART1_DIS	T21_DIS
r						rw	rw

符号	位	类型	描述
T21_DIS	0	rw	<b>定时器 T21 禁止请求位，高有效</b> 0 定时器 T21 正常工作（缺省状态） 1 定时器 T21 被关闭
UART1_DIS	1	rw	<b>UART1 禁止请求位，高有效</b> 0 UART1 正常工作（缺省状态） 1 UART1 被关闭
0	[7:2]	r	<b>保留</b> 读操作返回 0，应写入 0。

## ADC\_GLOBCTR

### 全局控制寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
ANON	DW	CTC		0			
rw	rw	rw		r			

符号	位	类型	描述
ANON	7	rw	<b>模拟部分开启控制</b> 该位使能 ADC 的模拟部分并定义其工作模式。 0 模拟部分关闭，不能进行模数转换。为了使功耗最小，内部模拟电路处于掉电模式，终止 $f_{\text{ADCI}}$ 时钟产生。 1 模拟部分开启，可进行模数转换。模拟部分的自动掉电功能被禁止。
0	3:0	r	<b>保留</b> 读操作返回 0，应写入 0。

# OSC\_CON

## OSC 控制寄存器

复位值: 0X00 1000<sub>B</sub>

7	6	5	4	3	2	1	0
PLLRDRES	PLLBYP	PLLPD	0	XPD	OSCSS	EORDRES	EXTOSCR
rwh	rwh	rw	r	rw	rwh	rwh	rh

符号	位	类型	描述
XPD	3	rw	<b>XTAL 掉电控制</b> 0 XTAL 不掉电 1 XTAL 掉电 XPD 是受保护位。保护机制有效时，该位不能被直接写入。
PLLPD	5	rw	<b>PLL 掉电控制</b> 0 PLL 不掉电 1 PLL 掉电 PLLPD 是受保护位。保护机制有效时，该位不能被直接写入。
0	4	r	<b>保留</b> 读操作返回 0，应写入 0。

## 9 看门狗定时器

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 必须在这个时间间隔内响应 WDT 以避免引发 XC878 系统复位。因此，WDT 服务程序可确保系统能够正常运行；确保系统能够在用户规定的时间周期内退出偶然出错状况。

WDT 缺省被禁止。

在调试模式下，看门狗定时器 WDT 缺省被挂起，停止计数（其调试挂起位缺省被置位，即 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
- 输入频率可选： $f_{\text{PCLK}}/2$  或  $f_{\text{PCLK}}/128$



## 看门狗定时器

- WDT 的高位字节重载值 WDTREL 可由寄存器 WDTREL 编程设定。  
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：

$$P_{WDT} = \frac{2^{(1+\langle WDTIN \rangle * 6)} * (2^{16} - WDTREL * 2^8)}{f_{CLK}} \quad (9.1)$$

如果 WDT 的窗界刷新特性被使能，若 WDTWINB 大于 WDTREL，则溢出周期  $P_{WDT}$  缩短，见图 9-2。用 WDTWINB 替换上面公式中的 WDTREL 即可计算  $P_{WDT}$ 。为了使该窗界刷新特性有用，WDTWINB 不能小于 WDTREL。

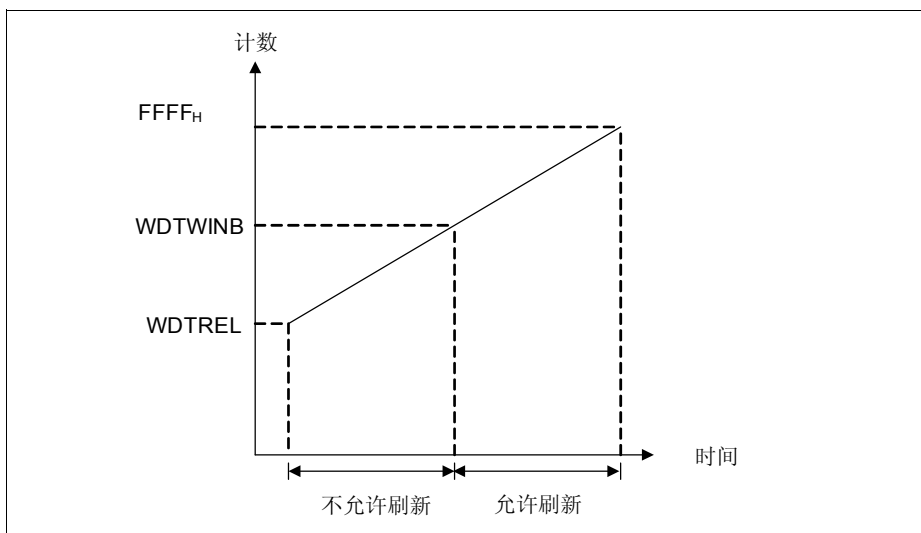


图 9-2 WDT 时序示意图

表 9-1 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 9-1 WDT 溢出周期范围

WDTREL 的重载值	$f_{WDT}$ 预分频					
	2 (WDTIN = 0)			128 (WDTIN = 1)		
	24 MHz	16 MHz	12 MHz	24 MHz	16 MHz	12 MHz
FF <sub>H</sub>	21.3 μs	32.0 μs	42.67 μs	1.37 ms	2.05 ms	2.73 ms
7F <sub>H</sub>	2.75 ms	4.13 ms	5.5 ms	176 ms	264 ms	352 ms
00 <sub>H</sub>	5.46 ms	8.19 ms	10.92 ms	350 ms	524 ms	699 ms

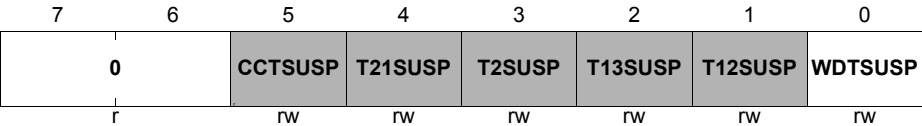
看门狗定时器

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

9.1.1 模块挂起控制

进入调试模式 WDT 缺省被挂起。清零 SFR MODSUSP 中的 WDTSUSP，允许 WDT 在调试模式工作。

**MODSUSP**  
模块挂起控制寄存器 复位值：01<sub>H</sub>



符号	位	类型	描述
WDTSUSP	0	rw	WDT 调试挂起位 0 WDT 不被挂起 1 WDT 被挂起
0	7:6	r	保留 读操作返回 0，应写入 0。

## 9.2 寄存器映射

WDT 操作由五个 SFR 控制，这些 SFR 位于映射 SFR 区。

表 9-2 列出这些 SFR 的地址。

表 9-2 SFR 地址列表

地址	寄存器
BB <sub>H</sub>	WDTCON
BC <sub>H</sub>	WDTREL
BD <sub>H</sub>	WDTWINB
BE <sub>H</sub>	WDTL
BF <sub>H</sub>	WDTH

## 9.3 寄存器描述

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。可位寻址 WDT 控制寄存器 WDTCON 用来控制 WDT 的操作，该寄存器还选择输入时钟的预分频因子。寄存器 WDTREL 规定定时器高位字节的重载值。

### WDTREL

WDT 重载寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
WDTREL							
rw							

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

### WDTCON

WDT 控制寄存器

复位值：00<sub>H</sub>

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

## 看门狗定时器

符号	位	类型	描述
<b>WDTIN</b>	0	rw	<b>看门狗定时器输入频率选择</b> 0 输入频率为 $f_{PCLK}/2$ 1 输入频率为 $f_{PCLK}/128$
<b>WDTRS</b>	1	rwh	<b>WDT 刷新启动控制</b> 高有效。该位置位将开始刷新 WDT，该位由硬件自动清零。
<b>WDTEN</b>	2	rw	<b>WDT 使能</b> WDTEN 是受保护位。保护机制（见 <a href="#">章节 3.4.4.1</a> ）有效时，该位不能被直接写入。 0 禁止 WDT 1 使能 WDT
<b>WDTPR</b>	4	rh	<b>WDT 预警模式标志</b> 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。 0 正常模式（复位后的缺省模式） 1 WDT 工作在预警模式
<b>WINBEN</b>	5	rw	<b>WDT 窗界控制</b> 0 禁止 WDT 窗界特性（缺省状态） 1 使能 WDT 窗界特性
<b>0</b>	3, [7:6]	r	<b>保留</b> 读操作返回 0，应写入 0。

## WDTL

WDT 寄存器，低位字节

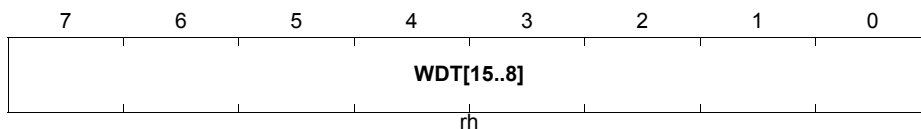
复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
WDT[7..0]							
rh							

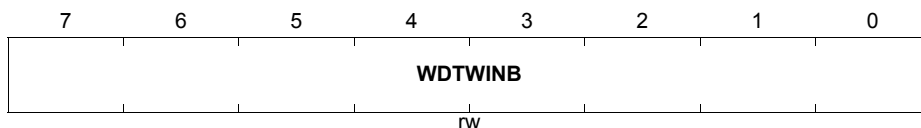
符号	位	类型	描述
<b>WDT[7..0]</b>	7:0	rh	<b>WDT 的当前值</b>



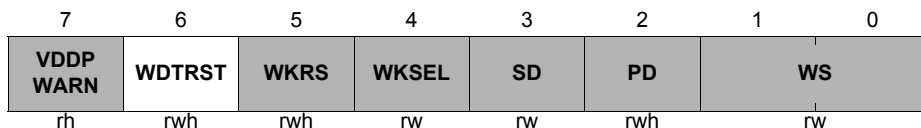
## 看门狗定时器

**WDTH**
**WDT 寄存器, 高位字节**
**复位值: 00<sub>H</sub>**


符号	位	类型	描述
WDTH[15..8]	7:0	rh	WDT 的当前值

**WDTWINB**
**WDT 窗界计数寄存器**
**复位值: 00<sub>H</sub>**


符号	位	类型	描述
WDTWINB	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 WDRST 将生效。 WDTWINB 与 WDTH 匹配。

**PMCON0**
**功率模式控制寄存器 0**
**复位值: 80<sub>H</sub><sup>1)</sup>**


<sup>1)</sup> 看门狗定时器复位对应复位值 C0<sub>H</sub>; 掉电唤醒复位对应复位值 A0<sub>H</sub>。

## 看门狗定时器

符号	位	类型	描述
WDTRST	6	rwh	<b>WDT 复位指示</b> 0 未发生 WDT 复位 1 WDT 已复位

# 10 乘法 / 除法单元

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 XC878 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：12 个寄存器用于数据处理； 1 个寄存器控制 MDU 操作； 1 个寄存器存储状态标志。和其它任何用于外设控制的寄存器一样，这些寄存器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
- 32 位无符号归一化操作
- 32 位算术 / 逻辑移位操作
- 2 个可选的工作频率：外设时钟或外设时钟的两倍

**表 10-1** 给出各种操作中计算所需的时钟周期数。

**表 10-1 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
有符号 16 位 /16 位	16 位	16 位	17
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)

## 10.1 功能描述

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 10-1）：

### 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。

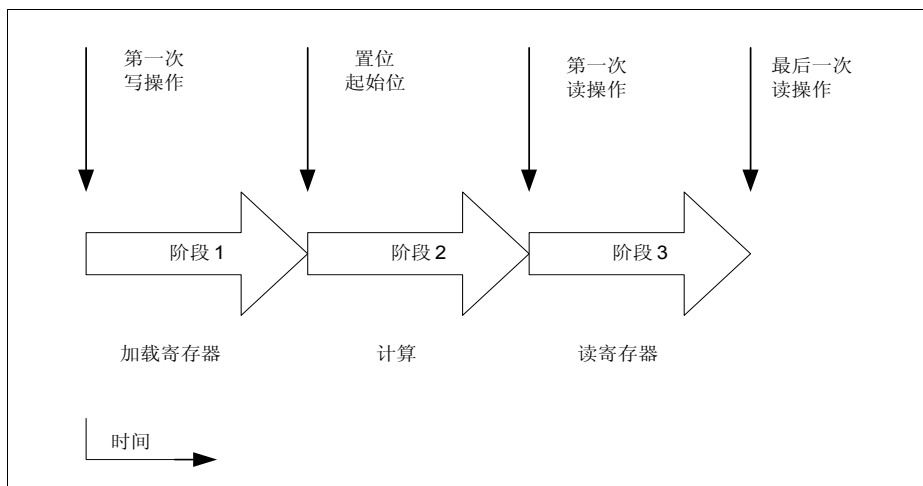


图 10-1 MDU 的操作阶段

### 10.1.1 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  $q = D/d$

且  $r = D \bmod d$

那么  $D = q*d + r$

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  $\text{sign}(r) = \text{sign}(D)$ 。

### 10.1.2 归一化

MDU 支持多达 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）中的无符号整数变量进行归一化。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 中无符号整数变量前面所有的 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（ $= 2^5 - 1$ ）。

### 10.1.3 移位

MDU 支持多达 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  $0101_B$  和  $1010_B$  进行一次算术右移，相应的结果为  $0010_B$  和  $1101_B$ 。

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：模块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。*

### 10.1.4 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。

如果需要执行新的计算，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。

### 10.1.5 错误检测

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 10.2](#)）。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；操作码错误在 MDUCON.START 置 1 后可被检测到。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。因此该结果不应使用。*

## 10.2 中断产生

MDU 的中断结构如[图 10-2](#)所示。MDU 可能产生两种中断事件，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 被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 个时钟周期的高电平。

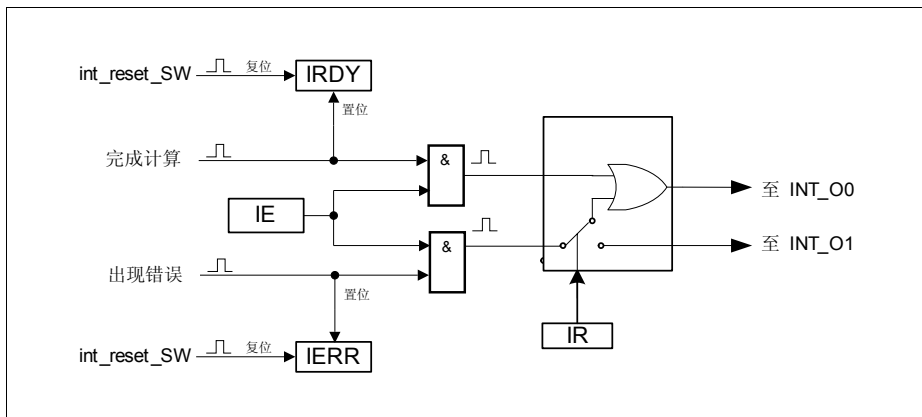


图 10-2 中断产生

### 10.3 低功耗模式

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。该模式通过置位寄存器 PMCON1 中的位 MDU\_DIS 来实现。外设时钟管理的具体内容请参见[章节 8.1.4](#)。

#### PMCON1

功率模式控制寄存器 1

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	CDC_DIS	CAN_DIS	MDU_DIS	T2CCU_DIS	CCU_DIS	SSC_DIS	ADC_DIS
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
MDU_DIS	4	rw	MDU 禁止请求位，高电平有效 0 MDU 处于正常工作模式（缺省） 1 请求禁止 MDU
0	7	r	保留 读操作返回 0，应写入 0。

## 10.4 寄存器映射

表 10-2 列出 MDU 的寄存器和相应地址：

表 10-2 MDU 寄存器

SFR	地址	寄存器名称
MDUCON	B1 <sub>H</sub> (映射)	MDU 控制寄存器
MDUSTAT	B0 <sub>H</sub> (映射)	MDU 状态寄存器
MD0/MR0	B2 <sub>H</sub> (映射)	MDU 数据 / 结果寄存器 0
MD1/MR1	B3 <sub>H</sub> (映射)	MDU 数据 / 结果寄存器 1
MD2/MR2	B4 <sub>H</sub> (映射)	MDU 数据 / 结果寄存器 2
MD3/MR3	B5 <sub>H</sub> (映射)	MDU 数据 / 结果寄存器 3
MD4/MR4	B6 <sub>H</sub> (映射)	MDU 数据 / 结果寄存器 4
MD5/MR5	B7 <sub>H</sub> (映射)	MDU 数据 / 结果寄存器 5

MD<sub>x</sub> 和 MR<sub>x</sub> 寄存器共用相同的地址。由于不能对 MR<sub>x</sub> 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 MD<sub>x</sub> 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 MD<sub>x</sub> 还是寄存器 MR<sub>x</sub>。缺省读取寄存器 MR<sub>x</sub>。



## 10.5 寄存器描述

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 10-3 和表 10-4。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 10-3 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	Control
MD5	M'orH	D'orH	D'orH	-

**表 10-4 MRx 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
MR0	PrL	QuoL	QuoL	ResultL
MR1	Pr	Quo	QuoH	Result
MR2	Pr	Quo	-	Result
MR3	PrH	QuoH	-	ResultH
MR4	M'orL	RemL	RemL	Control
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

## 乘法 / 除法单元

- ...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，规定移位方向、并保存已经执行的移位操作的次数。

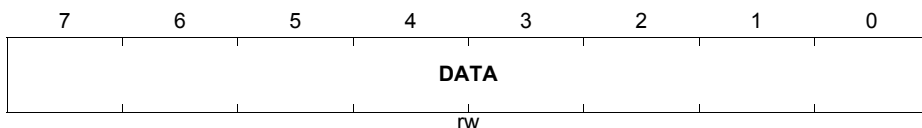
## 10.5.1 操作数和结果寄存器

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。

MDx (x = 0 - 5)

MDU 操作数寄存器

复位值: 00<sub>H</sub>

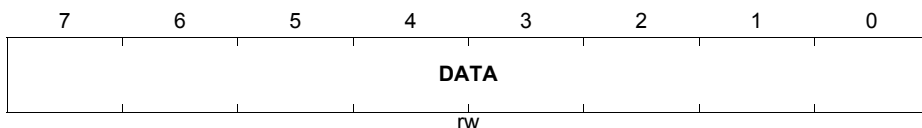


符号	位	类型	描述
DATA	7:0	rw	操作数值 见表 10-3。

MRx (x = 0 - 5)

MDU Result Register

复位值: 00<sub>H</sub>

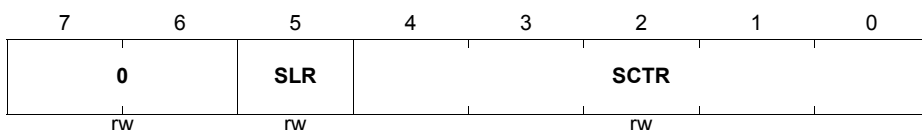


符号	位	类型	描述
DATA	7:0	rh	结果值 见表 10-4。

MD4

移位输入控制寄存器

复位值: 00<sub>H</sub>



**乘法 / 除法单元**

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 选择左移位 1 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。

**MR4**
**移位输出控制寄存器**
**复位值: 00<sub>H</sub>**

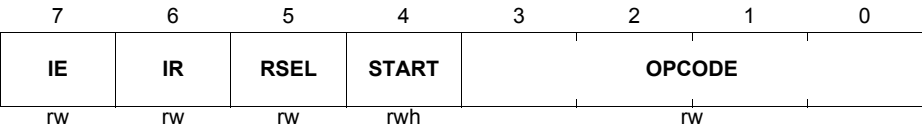
7	6	5	4	3	2	1	0
0			SCTR				
rh			rh				

符号	位	类型	描述
<b>SCTR</b>	4:0	rh	<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。

10.5.2 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。

MDUCON  
MDU 控制寄存器 复位值：00<sub>H</sub>



符号	位	类型	描述
OPCODE	3:0	rw	操作代码 0000 无符号 16 位乘法 0001 无符号 16 位 /16 位除法 0010 无符号 32 位 /16 位除法 0011 32 位逻辑左 / 右移 0100 有符号 16 位乘法 0101 有符号 16 位 /16 位除法 0110 有符号 32 位 /16 位除法 0111 32 位算术左 / 右移 1000 32 位归一化操作 其它：保留
START	4	rwh	启动位 START 由软件置位、硬件复位。 0 未开始执行操作 1 开始执行操作
RSEL	5	rw	读选择 0 读取 MRx 寄存器 1 读取 MDx 寄存器
IR	6	rw	中断线选择 0 两个中断源各有专用的中断线 1 两个中断源共用一条中断线 INT_OO
IE	7	rw	中断使能 0 禁止中断 1 使能中断

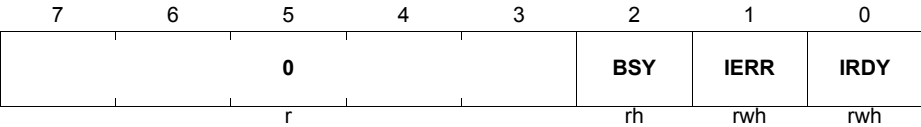
注： 在计算阶段，忙碌标志 MDUSTAT.BSY 置位时，不允许对 MDUCON 进行写操作。

注： 若 MDUCON.START 置 1，向 MDUCON 寄存器的操作代码位域写入保留值将会导致出错。

10.5.3 状态寄存器

寄存器 MDUSTAT 包含 MDU 的状态位。

**MDUSTAT**  
**MDU 状态寄存器** 复位值：00<sub>H</sub>



符号	位	类型	描述
IRDY	0	rwh	计算结果准备就绪中断 IRDY 位由硬件置位、软件复位。 0 成功执行操作后不触发中断 1 成功执行操作后触发中断
IERR	1	rwh	错误中断 IERR 位由硬件置位、软件复位。 0 出现错误时不触发中断 1 出现错误时触发中断
BSY	2	rh	忙碌标志 0 MDU 未进行计算 1 MDU 仍在进行计算
0	7:3	r	保留 读操作返回 0，应写入 0。

## 11 CORDIC 协处理器

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：

$$x_{i+1} = x_i - m \cdot d_i \cdot y_i \cdot 2^{-i} \quad (11.1)$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i} \quad (11.2)$$

$$z_{i+1} = z_i - d_i \cdot e_i \quad (11.3)$$

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

## 11.1 特性

- 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
- 2 种可选的工作频率：外设时钟或外设时钟的两倍
- 可通过门控时钟信号关闭模块
- 16 位数据访问宽度
  - 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



## 11.2 功能描述

以下各节将描述 CORDIC 协处理器的功能。

### 11.2.1 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 **CD\_STATC** 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。

### 11.2.2 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）。

### 11.2.3 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：  
 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 11-1 所示。

表 11-1 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

### 11.2.4 CORDIC 协处理器操作模式

表 11-2 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：

$$x_{i+1} = x_i - m \cdot d_i \cdot y_i \cdot 2^{-i} \tag{11.4}$$

$$y_{i+1} = y_i + d_i \cdot x_i \cdot 2^{-i} \tag{11.5}$$

$$z_{i+1} = z_i - d_i \cdot e_i \tag{11.6}$$

表 11-2 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
	$d_i = \text{sign}(z_i), z_i \rightarrow 0$	$d_i = -\text{sign}(y_i), y_i \rightarrow 0$
圆函数 $m = 1$ $e_i = \text{atan}(2^{-i})$	$X_{\text{final}} = K[X \cos(Z) - Y \sin(Z)] / \text{MPS}$ $Y_{\text{final}} = K[Y \cos(Z) + X \sin(Z)] / \text{MPS}$ $Z_{\text{final}} = 0$ 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: $\text{acos}(w) = \text{atan}[\sqrt{1-w^2} / w]$ $\text{asin}(w) = \text{atan}[w / \sqrt{1-w^2}]$
线性函数 $m = 0$ $e_i = 2^{-i}$	$X_{\text{final}} = X / \text{MPS}$ $Y_{\text{final}} = [Y + X Z] / \text{MPS}$ $Z_{\text{final}} = 0$	$X_{\text{final}} = X / \text{MPS}$ $Y_{\text{final}} = 0$ $Z_{\text{final}} = Z + Y / X$
	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 11-2 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： $\ln(w) = 2 \operatorname{atanh}[(w-1) / (w+1)]$ $\sqrt{(w+0.25)^2 - (w-0.25)^2}$ $\operatorname{acosh}(w) = \ln[w + \sqrt{(1-w^2)}]$ $\operatorname{asinh}(w) = \ln[w + \sqrt{(1+w^2)}]$

## 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 11-2 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 11.2.4.1。计算结果精度的详细描述，参见章节 11.2.6。
- 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  $\cosh(Z) + \sinh(Z)$  的正确结果。

### 11.2.4.1 收敛域

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；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。

**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 11-2** 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 11.2.6**。

### 11.2.4.2 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。

### 11.2.5 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 11.2.3](#)。

### 11.2.6 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 11.3.1](#)和[章节 11.3.2](#)。

若输入数据值在规定的定义域内（见[表 11-2](#)），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 11-3** 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 11-3 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% 1 : 49.1683% $X \leq 1$ 对应的 ND	0 : 55.8702% 1 : 44.1298% $Z \leq 1$ 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% 1 : 48.8579% 2 : 0.3681% 3 : 0.0023% 4 : 0.0002% $X \leq 4$ 对应的 ND	0 : 51.2011% 1 : 48.4944% 2 : 0.3024% 3 : 0.0020% 4 : 0.0001% $Y \leq 4$ 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% 1 : 33.0830% $X \leq 1$ 对应的 ND	0 : 88.5676% 1 : 11.4322% 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% 1 : 30.2859% $X \leq 1$ 对应的 ND	0 : 62.4055% 1 : 37.1965% 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% 1 : 34.5438% 2 : 17.9254% 3 : 11.6747% 4 : 1.3162% $X \leq 4$ 对应的 ND	0 : 58.3062% 1 : 41.6938% $Z \leq 1$ 对应的 ND

表 11-3 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 ≤ 7 对应的 ND
	X ≤ 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。

### 11.2.7 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。



### 11.3 CORDIC 协处理器内核

CORDIC 协处理器包含存放 X、Y 和 Z 数值（2 补码格式）的数据寄存器。3 个移位寄存器以迭代次数移位 X 和 Y 寄存器中的值并产生线性函数的仿真 LUT。此外，每个圆函数和双曲函数各有 1 个查找表（LUT）。通过多路数据选择器选出所需操作模式的 LUT 数据，然后以正确的符号位和 Z 寄存器数据相加。atan LUT 包含预计算的  $\text{atan}(2^{-i})$  数值； $\text{atanh}$  LUT 包含预计算的  $\text{atanh}(2^{-i})$  数值，查找表中存放迭代计数 i 对应的 2 补码格式的查找值。仿真 LUT 实际是一个移位寄存器，通过移位产生所需数据。有限状态机（FSM）每次切换到启动模式开始进行新的计算时重新加载移位寄存器。CORDIC 协处理器 FSM 控制计算流程。

#### 11.3.1 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（ $\text{atanh}$  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； $\text{atanh}$  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 11-4。

表 11-4  $\text{atan}(2^{-i})$  预计算值

迭代次数	$\text{atan}(2^{-i})$ 预计算值 (16 进制)	迭代次数	$\text{atan}(2^{-i})$ 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

- $\text{atanh}$  LUT 数据格式为 S19，见表 11-5。

表 11-5  $\text{atanh}(2^{-i})$  预计算值

迭代次数	$\text{atanh}(2^{-i})$ 预计算值 (16 进制)	迭代次数	$\text{atanh}(2^{-i})$ 预计算值 (16 进制)
i = 0	-	i = 8	28C
i = 1	16618	i = 9	146

表 11-5       $\operatorname{atanh}(2^{-i})$  预计算值

迭代次数	$\operatorname{atanh}(2^{-i})$ 预计算值 (16 进制)	迭代次数	$\operatorname{atanh}(2^{-i})$ 预计算值 (16 进制)
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

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。

11.3.2      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。

11.4 低功耗模式

如果完全不需要 CORDIC 功能，可关闭其时钟输入禁止该模块工作，最大程度降低功耗，该模式通过置位寄存器 PMCON1 中的位 CDC\_DIS 来实现。外设时钟管理的具体内容请参见[章节 8.1.4](#)。

PMCON1

功率模式控制寄存器 1

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	CDC_DIS	CAN_DIS	MDU_DIS	T2CCU_DIS	CCU_DIS	SSC_DIS	ADC_DIS
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
CDC_DIS	6	rw	禁止 CORDIC 请求位，高有效 0 CORDIC 正常工作（缺省） 1 请求禁止 CORDIC
0	7	r	保留 读操作返回 0，应写入 0。

### 11.5 寄存器映射

CORDIC 协处理器寄存器位于映射 SFR 区。[表 11-6](#) 给出这些 SFR 的地址列表。

注： 本节描述的所有 CORDIC 协处理器寄存器在引用时都需添加模块名前缀“CD\_”。

表 11-6 CORDIC 协处理器寄存器

寄存器缩写名	地址 (HEX)	复位值 (HEX)	寄存器全名
CD_CORDXL	9A	00	CORDIC X 数据低字节
CD_CORDXH	9B	00	CORDIC X 数据高字节
CD_CORDYL	9C	00	CORDIC Y 数据低字节
CD_CORDYH	9D	00	CORDIC Y 数据高字节
CD_CORDZL	9E	00	CORDIC Z 数据低字节
CD_CORDZH	9F	00	CORDIC Z 数据高字节
CD_STATC	A0	00	CORDIC 状态和数据控制寄存器
CD_CON	A1	62	CORDIC 控制寄存器

## 11.6 寄存器描述

### 11.6.1 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

复位值：62<sub>H</sub>

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	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。
MODE	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
ROTVEC	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
ST_MODE	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

CORDIC 协处理器

符号	位	类型	描述
<b>X_USIGN</b>	5	rw	<p><b>圆函数向量模式下 X 结果数据格式</b></p> <p>DMAP = 0, 读取 X 结果数据时, X 数据格式如下:</p> <p>0     有符号数, 2 补码</p> <p>1     无符号数 (缺省)</p> <p>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。</p> <p><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></p> <p><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></p>
<b>MPS</b>	7:6	rw	<p><b>X 和 Y 幅值调节因子</b></p> <p>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。</p> <p>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。</p> <p>00   除以 1</p> <p>01   除以 2 (缺省)</p> <p>10   除以 4</p> <p>11   保留, 保存最近的 MPS 设置</p>

CORDIC 协处理器

11.6.2 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** 复位值: 00<sub>H</sub>

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	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。该位读取后被清零、或当启动一次新的 CORDIC 运算时被清零。
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	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：</i> 映射数据寄存器和 <b>KEEP</b> 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。</p> <p><i>注：</i> 如果在多步运算中 <b>KEEPX</b> 被置位，X 的最终结果精度会降低，不能保证具有<a href="#">章节 11.2.6</a>给出的精度。</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>

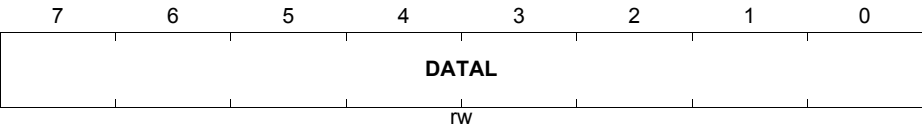
11.6.3 数据寄存器

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 (x = X, Y or Z)

CORDIC x 数据低字节

复位值：00<sub>H</sub>





**CORDIC 协处理器**

符号	位	类型	描述
<b>DATAL</b>	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 (x = X, Y or Z)**
**CORDIC x 数据高字节**
**复位值: 00<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>DATAH</b>							
rw							

符号	位	类型	描述
<b>DATAH</b>	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

## 12 串行接口

XC878有三个与外部器件通信的串行接口，包括两个通用异步收发器（UART和UART1）和一个高速同步串行接口（SSC）。此外，可用 UART 模块来支持本地互连网络（LIN）协议。

### UART 和 UART1 特性

- 全双工异步模式
  - 8 位或 9 位的数据帧，最低有效位（LSB）在先
  - 固定或可变的波特率
- 接收缓存
- 多处理器通信
- 数据发送或接收完成产生中断

### LIN 特性

- 主模式和从模式操作

### SSC 特性

- 主模式和从模式操作
  - 全双工和半双工操作
- 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）在先
  - 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）

## 12.1 UART

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。

注：一般情况下术语“UART”代表串行端口，用于 UART 和 UART1 模块的描述。如果“UART”后紧跟单词“模块”，表达为“UART 模块”，则代表第一个 UART 模块。

### 12.1.1 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 12-1 所示。

表 12-1 UART 的工作模式

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	$f_{PCLK}/64$ or $f_{PCLK}/32^{1)}$
1	1	模式 3: 9 位移位 UART	可变

<sup>1)</sup> UART1 模块的波特率固定为  $f_{PCLK}/64$ 。

#### 12.1.1.1 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（设备）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。在后面的 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。

### 12.1.1.2 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 11111111B 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 12.1.2](#)）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 12-1](#)所示。

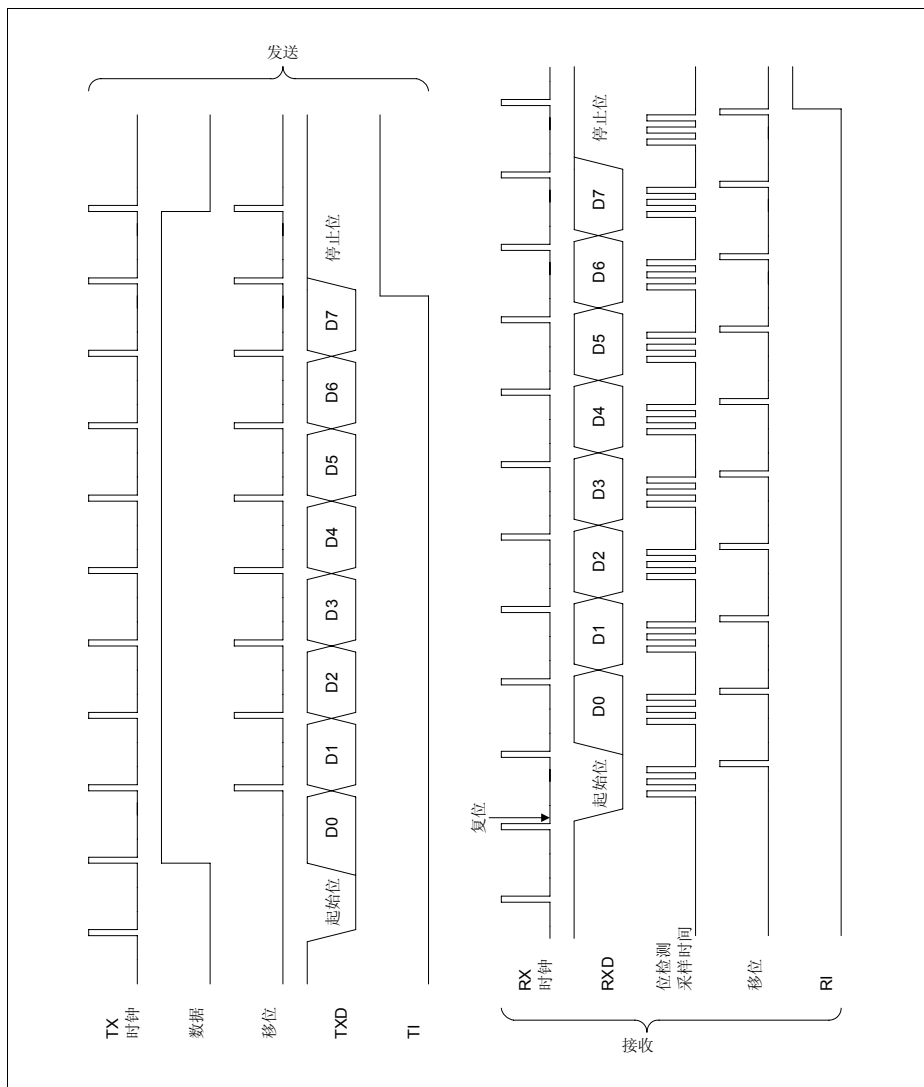


图 12-1 串行接口，模式 1，时序图

### 12.1.1.3 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 12.1.2](#)）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 模块的波特率为  $f_{PCLK}/32$  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，该位用于双倍频选择；UART1 模块的波特率固定为  $f_{PCLK}/64$ 。

### 12.1.1.4 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 12-2](#)所示。

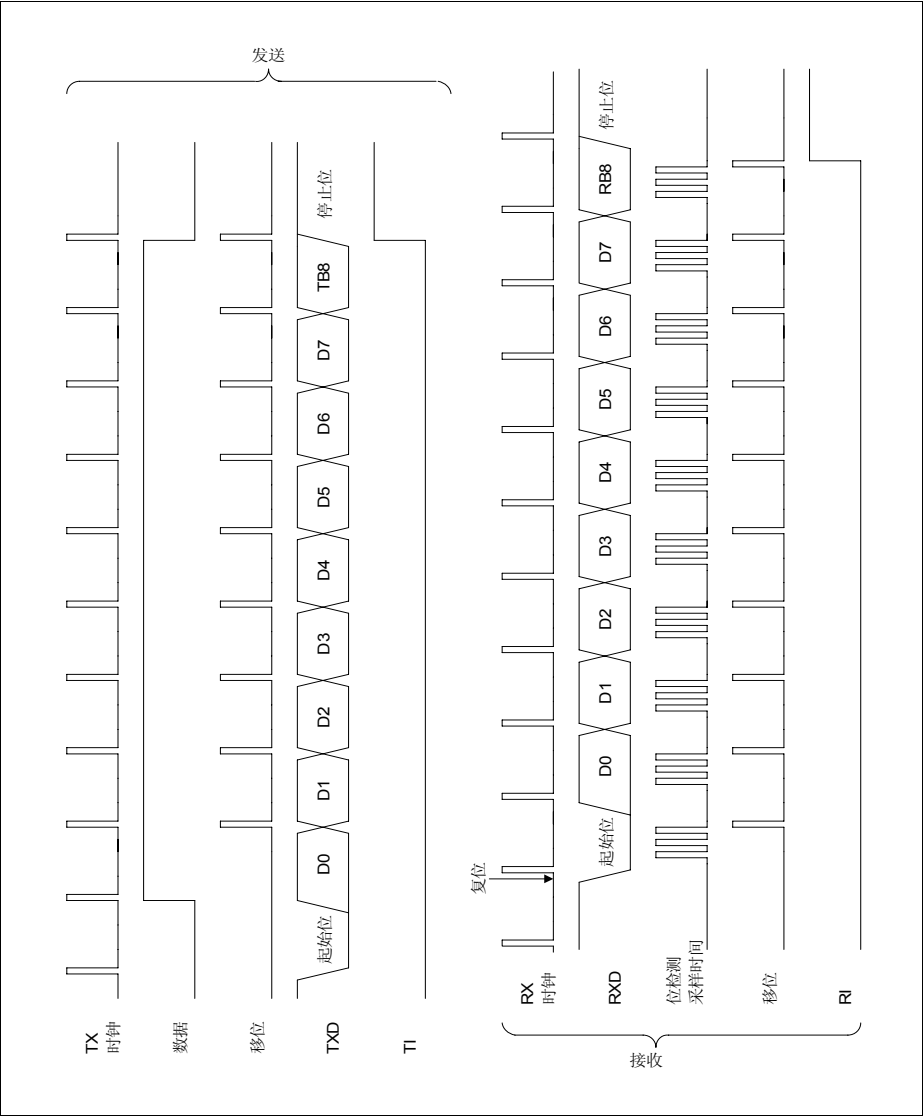


图 12-2 串行接口，模式 2 和 3，时序图

### 12.1.2 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。

当主机需要发送一数据块给数台从机之一时，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。

### 12.1.3 UART 寄存器描述

每个 UART 模块（UART 和 UART1）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。

UART1 模块有一个附加的 SFR SCON1，用来使能或禁止接收、发送和分频器溢出中断。例如：置位 RIEN 为 1，将使能 UART1 模块的接收中断。

#### SBUF

串行数据缓存寄存器

复位值：00<sub>H</sub>

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

复位值：00<sub>H</sub>

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>接收中断标志</b> 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。
<b>TI</b>	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。
<b>RB8</b>	2	rwh	<b>串口接收的第 9 位</b> 模式 2 和 3 中，该位是接收到的第 9 位数据；模式 1 中该位是接收到的停止位；模式 0 中，未使用该位。
<b>TB8</b>	3	rw	<b>串口发送的第 9 位</b> 模式 2 和 3 中，该位是发送的第 9 位数据。
<b>REN</b>	4	rw	<b>串口接收使能位</b> 0 禁止串口接收 1 使能串口接收
<b>SM2</b>	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。
<b>SM1, SM0</b>	6 7	rw	<b>串口工作模式选择</b> 00 模式 0：8 位移位寄存器，固定波特率 ( $f_{PCLK}/2$ ) 01 模式 1：8 位 UART，可变波特率 10 模式 2：9 位 UART，固定波特率 ( $f_{PCLK}/32$ 或 $f_{PCLK}/64$ ) 11 模式 3：9 位 UART，可变波特率

串行接口

# SCON1

串行通道控制寄存器 1

复位值：07<sub>H</sub>

7	6	5	4	3	2	1	0
0					NDOVEN	TIEN	RIEN
r					rw	rw	rw

符号	位	类型	描述
RIEN	0	rw	接收中断使能位 0 禁止接收中断 1 使能接收中断
TIEN	1	rw	发送中断使能位 0 禁止发送中断 1 使能发送中断
NDOVEN	2	rw	正常分频器溢出使能位 0 禁止正常分频器溢出中断 1 使能正常分频器溢出中断
0	[7:3]	r	保留 读操作返回 0；应写入 0。

### 12.1.4 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 12.1.4.1）

在模式 1 和模式 3 中，可变的波特率由：

- 专用波特率发生器产生（见章节 12.1.4.2）

此外，UART 模块的波特率还可由：

- 定时器 T1 产生（见章节 12.1.4.3）

由 UART 模块寄存器 FDCON 中的位 BGS 来选择不同的波特率产生方式。

#### 12.1.4.1 固定时钟

UART 模块在模式 0 和模式 2 中的波特率固定不变；在模式 0 中的波特率只能为  $f_{PCLK}/2$ ，在模式 2 中的波特率可为  $f_{PCLK}/64$  或  $f_{PCLK}/32$ ，这取决于 PCON（功率控制寄存器）中最高位（SMOD）的设置，在模式 1、2 和 3 中，该位作为波特率双倍频选择。在模式 1 和模式 3 中，只有由定时器 T1 提供的可变波特率才和 SMOD 有关，由专用波特率发生器提供的波特率和 SMOD 的设置无关。

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，专用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。

### PCON

功率控制寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
SMOD		0		GF1	GF0	0	IDLE
r/w		r		r/w	r/w	r	r/w

符号	位	类型	描述
SMOD	7	r/w	波特率双倍频使能 0 模式 1、2 和 3 的波特率禁止使用双倍频 1 模式 2 的波特率双倍频；只有由定时 T1 产生波特率时，模式 1 和 3 的波特率才双倍频
0	1, [6:4]	r	保留 读操作返回 0；应写入 0。

## 模式 2 中的波特率

对于 UART 模块，模式 2 的波特率和寄存器 PCON 中 SMOD 的设置有关。如果 SMOD = 0（复位值），波特率为输入时钟  $f_{PCLK}$  的 1/64，如果 SMOD = 1，波特率为输入时钟  $f_{PCLK}$  的 1/32。

(12.1)

$$\text{模式2的波特率} = \frac{2^{\text{SMOD}}}{64} \times f_{PCLK}$$

对于 UART1 模块，模式 2 的波特率为输入时钟  $f_{PCLK}$  的 1/64。

### 12.1.4.2 专用波特率发生器

每个 UART 模块都有一个专用波特率发生器，由可编程 8 位重载值和分频器组（即，预分频器和分数分频器）构成，在输入时钟  $f_{PCLK}$  的基础上产生很宽范围的波特率。

波特率定时器递减计数。如果分数分频器被使能（FDCON.FDEN = 1），则由分数分频器的输出（ $f_{MOD}$ ）为波特率定时器提供时钟；如果分数分频器被禁止（FDEN = 0），则由预分频器的输出（ $f_{DIV}$ ）为波特率定时器提供时钟。分数分频器在用于波特率产生时，必须被设置为分数分频模式（FDCON.FDM = 0）。可通过波特率运行控制位 BCON.R 来启动或停止波特率定时器工作。每次定时器下溢为串行通道产生一个时钟脉冲，同时将寄存器 BG 中的 8 位重载值重新载入定时器。

分数分频器工作在正常分频模式下时（FDEN = 1 和 FDM = 1），波特率定时器停止工作，控制位 BCON.R 失效。

寄存器 BG 是具有双重功能的波特率发生器 / 重载寄存器。若 BCON.R = 1，读取 BG 返回定时器的计数值；写入 BG 则将其内容自动重载到波特率定时器中。若当 BCON.R = 0 时，对 BG 执行写操作，自动重载操作将被延迟，在 BCON.R 置位之后的第一个指令周期才进行自动重载。

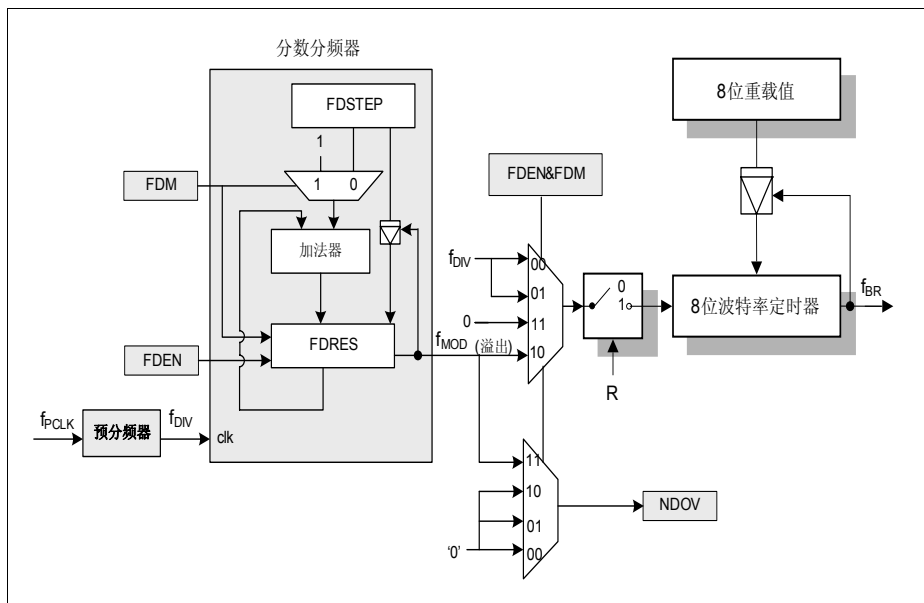


图 12-3 波特率发生器电路

波特率 ( $f_{BR}$ ) 的值取决于以下参数:

- 输入时钟  $f_{PCLK}$
- 寄存器 BCON 中位域 BRPRE 定义的预分频因子 ( $2^{BRPRE}$ )
- 寄存器 FDSTEP 定义的分数分频器 ( $STEP/256$ )  
(只有在分数分频器被使能, 并且工作在分数分频模式时才使用该参数)
- 寄存器 BG 定义的波特率定时器的 8 位重载值 ( $BR\_VALUE$ )

下面给出波特率的计算公式, 公式 (12.2) 对应无分数分频的情况; 公式 (12.3) 对应带有分数分频的情况:

(12.2)

$$\text{波特率} = \frac{f_{PCLK}}{16 \times 2^{BRPRE} \times (BR\_VALUE + 1)} \quad \text{其中 } 2^{BRPRE} \times (BR\_VALUE + 1) > 1$$

(12.3)

$$\text{波特率} = \frac{f_{PCLK}}{16 \times 2^{BRPRE} \times (BR\_VALUE + 1)} \times \frac{STEP}{256}$$

## 串行接口

可以产生的最大波特率为 $f_{PCLK}/32$ 。因此，24MHz 模块时钟所能产生的最大波特率为 0.75 MBaud。

标准 LIN 协议支持的最大波特率为 20 kHz，波特率的精度并不严格，分数分频器可以被禁止。只有预分频器用于自动波特率计算。LIN 快速模式可支持的波特率范围为 20 kHz 到 57.6 kHz，高波特率需要使用分数分频器以获得更高的精度。

**表 12-2** 列出在分频器被禁止、模块时钟为 24MHz 的情况下，各种典型波特率、对应的参数设置以及与理想波特率相比较的波特率偏差。

**表 12-2 UART 典型波特率（分数分频器被禁止）**

波特率	预分频因子 ( $2^{BRPRE}$ )	重载值 (BR_VALUE + 1)	偏差
19.2 kBaud	1 (BRPRE=000 <sub>B</sub> )	78 (4E <sub>H</sub> )	0.17 %
9600 Baud	1 (BRPRE=000 <sub>B</sub> )	156 (9C <sub>H</sub> )	0.17 %
4800 Baud	2 (BRPRE=001 <sub>B</sub> )	156 (9C <sub>H</sub> )	0.17 %
2400 Baud	4 (BRPRE=010 <sub>B</sub> )	156 (9C <sub>H</sub> )	0.17 %

利用分数分频器可产生精度更高的波特率（偏差更低），**表 12-3** 列出采用不同时钟频率产生 57.6 kHz 波特率时，所对应的波特率偏差。分数分频器被使能（分数分频模式），相应的参数设置如下表所示：

**表 12-3 UART 偏差（分数分频器使能）**

$f_{PCLK}$	预分频因子 ( $2^{BRPRE}$ )	重载值 (BR_VALUE + 1)	STEP	偏差
26.67 MHz	1	20 (14 <sub>H</sub> )	177 (B1 <sub>H</sub> )	+0.03 %
24 MHz	1	6 (6 <sub>H</sub> )	59 (3B <sub>H</sub> )	+0.03 %
16 MHz	1	4 (4 <sub>H</sub> )	59 (3B <sub>H</sub> )	+0.03 %
13.33 MHz	1	10 (A <sub>H</sub> )	177 (B1 <sub>H</sub> )	+0.03 %
12 MHz	1	3 (3 <sub>H</sub> )	59 (3B <sub>H</sub> )	+0.03 %
8 MHz	1	2 (2 <sub>H</sub> )	59 (3B <sub>H</sub> )	+0.03 %
6.67 MHz	1	5 (5 <sub>H</sub> )	177 (B1 <sub>H</sub> )	+0.03 %
6 MHz	1	6 (6 <sub>H</sub> )	236 (EC <sub>H</sub> )	+0.03 %

## 分数分频器

8 位分数分频器的输入时钟  $f_{DIV}$  经  $1/n$ 、或  $n/256$  分频后，产生输出时钟  $f_{MOD}$ ，该时钟用作波特率定时器的输入。分数分频器有两种工作模式：

- 分数分频模式
- 正常分频模式

## 分数分频模式

将寄存器 **FDCON** 中的位 **FDM** 清零，选择工作在分数分频模式。一旦分数分频器被使能（**FDEN** = 1），分数分频器的输出时钟  $f_{MOD}$  源自输入时钟  $f_{DIV}$  经  $n/256$  分频，参数  $n$  由寄存器 **FDSTEP** 中的位域 **STEP** 定义，取值在 0 到 255 之间。

在分数分频模式下，输出时钟脉冲  $f_{MOD}$  和 **FDRES.RESULT**+**FDSTEP.STEP** 之和有关；如果相加结果大于 **FF<sub>H</sub>** 而导致溢出，产生一个  $f_{MOD}$  脉冲。

分数分频模式下，平均输出频率由下面公式计算得出：

(12.4)

$$f_{MOD} = f_{DIV} \times \frac{STEP}{256} \quad \text{其中 } STEP = 0 - 255$$

图 12-4 所示为在分数分频模式下，重载值 **STEP** = 8D<sub>H</sub>（分频因子为  $141/256 = 0.55$ ）的时序图。

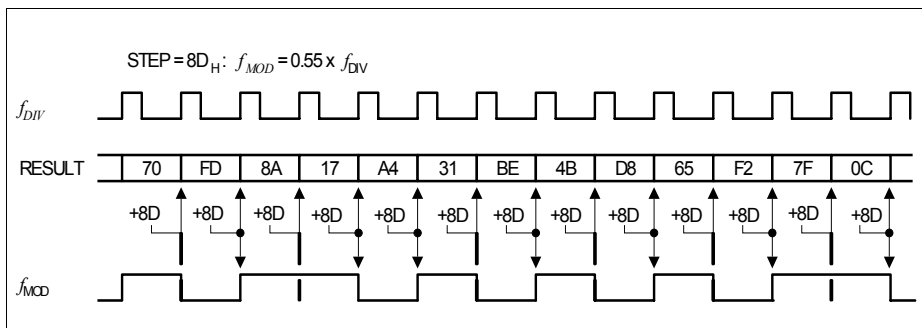


图 12-4 分数分频模式时序图

注：在分数分频模式下， $f_{MOD}$  的最大误差为一个  $f_{DIV}$  时钟周期。

通常，分数分频模式可用于产生比正常分频模式精度更高的平均输出时钟频率。

## 正常分频模式

将寄存器 **FDCON** 中的位 **FDM** 置 1，分数分频器工作在正常分频模式，与此同时禁止波特率产生（见图 12-3）。一旦分数分频器被使能（**FDEN=1**），它被用作一个 8 位自动重载定时器（和波特率产生无关），从重载值开始递增计数，每个输入时钟脉冲计数一次。寄存器 **FDRES** 中的位域 **RESULT** 代表定时器的值，寄存器 **FDSTEP** 的位域 **STEP** 定义重载值。定时器每次溢出，将置位溢出标志（**FDCON.NDOV**），如果位 **NDOVEN** 置位，则产生中断请求。**UART** 模块对应的中断使能位位于 **BCON** 寄存器，**UART1** 对应的中断使能位位于 **SCON1** 寄存器。输出时钟  $f_{MOD}$  为输入时钟  $f_{DIV}$  的  $1/n$ ，其中  $n$  定义为  $256 - STEP$ 。

正常分频模式下的输出频率可由下面的公式计算得到：

(12.5)

$$f_{MOD} = f_{DIV} \times \frac{1}{256 - STEP}$$

图 12-5 给出正常分频模式下、重载值  $STEP = FD_H$  时的操作。为了使  $f_{MOD} = f_{DIV}$ ，**STEP** 必须被编程为  $FF_H$ 。

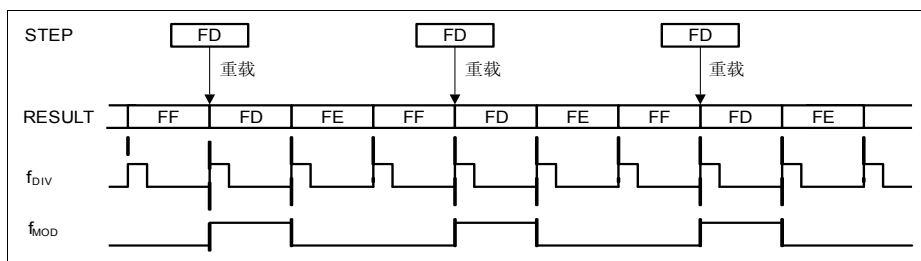


图 12-5 正常分频模式时序图

## 波特率发生器寄存器

**UART** 和 **UART1** 模块中的波特率发生器都对应有五个 SFR: **BG**、**BCON**、**FDCON**、**FDSTEP** 和 **FDRES**。这些寄存器的功能描述如下。

寄存器 **BCON** 包含波特率发生器的控制位和预分频因子选择位。

## BCON

### 波特率控制寄存器

复位值:  $20_H$ 

7	6	5	4	3	2	1	0
<b>BGSEL</b>		<b>NDOVEN</b>	<b>BRDIS</b>	<b>BRPRE</b>		<b>R</b>	
rw		rw	rw	rw		rw	



符号	位	类型	描述
<b>R</b>	0	rw	<b>波特率发生器运行控制</b> 0 禁止波特率发生器工作 1 使能波特率发生器工作 <i>注： R=0 时 BR_VALUE 只能写入。</i>
<b>BRPRE</b>	[3:1]	rw	<b>预分频选择</b> 000 $f_{DIV} = f_{PCLK}$ 001 $f_{DIV} = f_{PCLK}/2$ 010 $f_{DIV} = f_{PCLK}/4$ 011 $f_{DIV} = f_{PCLK}/8$ 100 $f_{DIV} = f_{PCLK}/16$ 101 $f_{DIV} = f_{PCLK}/32$ 其它：保留
<b>BRDIS</b>	4	rw	<b>禁止分隔 / 同步检测</b> 0 分隔 / 同步检测被使能 1 分隔 / 同步检测被禁止
<b>NDOVEN</b>	5	rw	<b>正常分频器溢出使能位</b> 0 禁止正常分频器溢出中断 1 使能正常分频器溢出中断
<b>BGSEL</b>	[7:6]	rw	<b>检测波特率范围选择</b> 对应不同的 BGSEL 值，要检测的波特率范围由下面的公式定义： $f_{PCLK}/(2184 \cdot 2^{BGSEL}) < \text{波特率范围} < f_{PCLK}/(72 \cdot 2^{BGSEL})$ 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 12-4。

注： 位 BRDIS、BGSEL 和 NDOVEN 只适用于 UART 模块，UART1 模块中不使用。因此，在 UART1 模块中，这些位应该始终写 0，写 1 不起作用。UART1 的 BCON 的复位值为 00<sub>H</sub>。

注： 请注意 UART1 的 NDOVEN 位于 SCON1 寄存器中。

表 12-4 对应不同输入频率时 BGSEL 位域定义

$f_{PCLK}$	BGSEL	检测波特率范围选择 $f_{PCLK}/(2184*2^{BGSEL})$ 到 $f_{PCLK}/(72*2^{BGSEL})$
24 MHz	00 <sub>B</sub>	11 kHz to 333.3 kHz
	01 <sub>B</sub>	5.5 kHz to 166.6 kHz
	10 <sub>B</sub>	2.8 kHz to 83.3 kHz
	11 <sub>B</sub>	1.4 kHz to 41.6 kHz
12 MHz	00 <sub>B</sub>	5.5 kHz to 166.6 kHz
	01 <sub>B</sub>	2.8 kHz to 83.3 kHz
	10 <sub>B</sub>	1.4 kHz to 41.6 kHz
	11 <sub>B</sub>	0.7 kHz to 20.8 kHz
2 MHz	00 <sub>B</sub>	0.92 kHz to 27.7 kHz
	01 <sub>B</sub>	0.46 kHz to 13.8 kHz
	10 <sub>B</sub>	0.23 kHz to 6.9 kHz
	11 <sub>B</sub>	0.12 kHz to 3.4 kHz

当  $f_{PCLK} = 24$  MHz 时，可检测到的波特率范围为 1.4 kHz 到 333.3 kHz。为了提高波特率检测精度，可参考以下范例进行 BGSEL 选择：

- 如果波特率落在 1.4 kHz 到 2.8 kHz 之间，选择 BGSEL 为 “11<sub>B</sub>”
- 如果波特率落在 2.8 kHz 到 5.5 kHz 之间，选择 BGSEL 为 “10<sub>B</sub>”
- 如果波特率落在 5.5 kHz 到 11 kHz 之间，选择 BGSEL 为 “01<sub>B</sub>”
- 如果波特率在 11 kHz 到 333.3 kHz 之间，选择 BGSEL 为 “00<sub>B</sub>”。如果波特率为 20 kHz，BGSEL 的可选值为：“00<sub>B</sub>”，“01<sub>B</sub>”，“10<sub>B</sub>”，“11<sub>B</sub>”。然而建议用户选择 “00<sub>B</sub>”，这样可得到更高的检测精度。

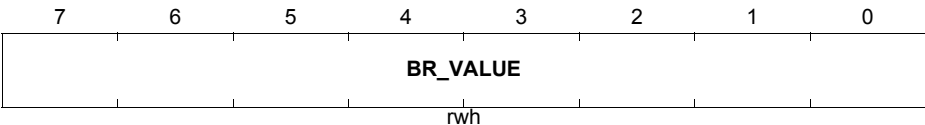
当系统工作在低速模式时，也可检测波特率。标准 LIN 波特率检测所需最小  $f_{PCLK}$  为 2 MHz，此时可检测的波特率范围是 0.12 kHz 到 27.7 kHz。

寄存器 BG 包含波特率定时器的 8 位重载值。

## BG

波特率定时器 / 重载寄存器

复位值：00<sub>H</sub>



符号	位	类型	描述
BR_VALUE	[7:0]	rwh	<b>波特率定时器 / 重载值</b> 读操作将返回波特率定时器的 8 位计数值；写操作将载入波特率定时器 / 重载值。 <i>注： R=0 时 BG 只能写入。</i>

## 串行接口

寄存器 FDCON 包含分数分频器的控制和状态位，以及用于支持 LIN 协议的状态标志（见[章节 12.2.1](#)）。

### FDCON

分数分频器控制寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
BGS	SYNEN	ERRSYN	EOFSYN	BRK	NDOV	FDM	FDEN
rw	rw	rwh	rwh	rwh	rwh	rw	rw

符号	位	类型	描述
FDEN	0	rw	分数分频器使能位 0 分数分频器被禁止，只使用预分频器 1 分数分频器被使能
FDM	1	rw	分数分频器模式选择 0 选择分数分频模式 1 选择正常分频模式
NDOV	2	rwh	正常分频模式下的溢出标志 该位由硬件置位，只能由软件清零。 0 未产生中断请求 1 产生中断请求
BRK	3	rwh	分隔域标志 该位由硬件置位，只能由软件清零。 0 未检测到分隔域 1 检测到分隔域
EOFSYN	4	rwh	同步字节结束标志 该位由硬件置位，只能由软件清零。 0 未检测到同步字节结束 1 检测到同步字节结束
ERRSYN	5	rwh	同步字节出错标志 该位由硬件置位，只能由软件清零。 0 在同步字节中未检测到错误 1 在同步字节中检测到错误
SYNEN	6	rw	同步字节结束和同步字节错误中断使能 0 禁止同步字节结束和同步字节错误中断 1 使能同步字节结束和同步字节错误中断

串行接口

符号	位	类型	描述
BGS	7	rw	波特率发生器选择
			0 选择专用波特率发生器
			1 选择定时器 T1

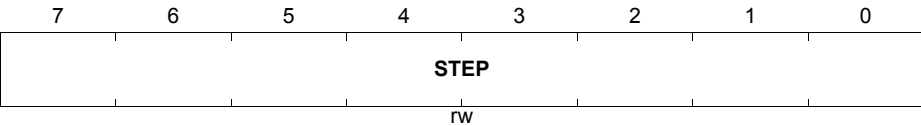
注： 位 3 至位 7 只用于 UART 模块， UART1 模块中不使用。因此， UART1 模块中 FDCON 寄存器中位 3 至位 7 应该始终写入 0， 写入 1 不起作用。

寄存器 FDDSTEP 中包含分数分频器的 8 位 STEP 值。

FDDSTEP

分数分频器重载寄存器

复位值： 00<sub>H</sub>

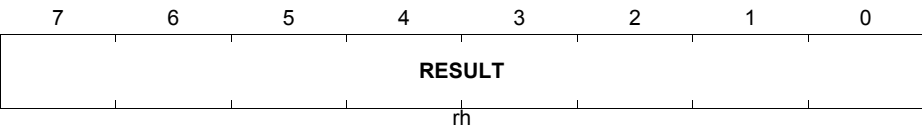


符号	位	类型	描述
STEP	[7:0]	rw	<b>STEP 值</b> 在正常分频模式下， STEP 代表 RESULT 的重载值。 在分数分频模式下， STEP 定义在每个输入时钟周期，和 RESULT 相加的 8 位值。

寄存器 FDRES 包含分数分频器的 8 位 RESULT 值。

**FDRES**  
 分数分频器结果寄存器

复位值: 00<sub>H</sub>



符号	位	类型	描述
RESULT	[7:0]	rh	结果值（RESULT 值） 在正常分频模式下，RESULT 用作重载计数器（和 +1）。 在分数分频模式下，该位域包含 RESULT+STEP 相加的结果。 如果 FDEN 位从 “0” 变到 “1”，RESULT 中加载 FF <sub>H</sub> 。

### 12.1.4.3 定时器 T1

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 SMOD 取值共同决定，计算公式如下：

(12.6)

$$\text{模式 1、3 的波特率} = \frac{2^{\text{SMOD}} \times f_{\text{PCLK}}}{32 \times 2 \times (256 - \text{TH1})}$$

对于给定波特率，定时器 T1 高位字节取值可由下式推导：

(12.7)

$$\text{TH1} = 256 - \frac{2^{\text{SMOD}} \times f_{\text{PCLK}}}{32 \times 2 \times \text{模式 1、3 的波特率}}$$

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。

注： 定时器 T1 不能用来产生 UART1 模块的可变波特率。

## 12.1.5 端口控制

可选择从三个不同的输入 RXD\_0、RXD\_1 和 RXD\_2 将数据移入 UART 模块；UART 模块由位 MODPISSEL.URRIS 和 MODPISSEL.URRISH 进行接收输入选择；UART1 模块由位 MODPISSEL1.UR1RIS 进行输入选择。

### MODPISSEL

外设输入选择寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	URRISH	JTAGTDIS	JTAGTCK S	EXINT2IS	EXINT1IS	EXINT0IS	URRIS
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
URRISH, URRIS	6,0	rw	<b>UART 接收输入选择 [6,0]</b> 00 选择 UART 接收输入 RXD_0 01 选择 UART 接收输入 RXD_1 10 选择 UART 接收输入 RXD_2 11 保留
0	7	r	<b>保留</b> 读操作返回 0，应写入 0。

### MODPISSEL1

外设输入选择寄存器 1

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
EXINT6IS			UR1RIS		T21EXIS	0	
rw			rw		rw	r	

符号	位	类型	描述
UR1RIS	[4:3]	rw	<b>UART1 接收输入选择</b> 00 选择 UART1 接收输入 RXD_0 01 选择 UART1 接收输入 RXD_1 10 选择 UART1 接收输入 RXD_2 11 保留
0	[1:0]	r	<b>保留</b> 读操作返回 0，应写入 0。



12.1.6 低功耗模式

如果完全不需要 UART1 模块功能，可关闭其输入时钟禁止该模块工作，最大程度降低功耗。该模式通过置位寄存器 PMCON2 中的 UART1\_DIS 来实现，描述如下。外设时钟管理的具体描述请参见[章节 8.1.4](#)。

PMCON2

功率模式控制寄存器 1

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0						UART1_DIS	T21_DIS
r						rw	rw

符号	位	类型	描述
UART1_DIS	1	rw	UART1 模块禁止请求位，高有效 0 UART1 模块正常工作（缺省状态） 1 请求禁止 UART1 模块
0	[7:2]	r	保留 读操作返回 0，应写入 0。

注： UART 模块不支持低功耗模式。

### 12.1.7 寄存器映射

前面提到的所有 UART1 模块寄存器，在本手册其它章节中引用时需添加模块名前缀“UART1\_”，如寄存器 UART1\_SCON；不过，所有 UART 模块寄存器不需添加前缀。

UART 模块中的寄存器 SCON 和 SBUF 既可从标准 SFR 区（非映射）、也可从映射 SFR 区访问，UART 模块中的其它 SFR 均位于标准 SFR 区中 SCU 的第 0 页；UART1 模块中的所有 SFR 均位于映射 SFR 区。

**表 12-5** 列出这些 SFR 的地址。

**表 12-5      UART 模块 SFR 地址列表**

UART 模块		UART1 模块	
地址	寄存器	地址	寄存器
98 <sub>H</sub>	SCON	C8 <sub>H</sub>	SCON
99 <sub>H</sub>	SBUF	C9 <sub>H</sub>	SBUF
BD <sub>H</sub>	BCON	CA <sub>H</sub>	BCON
BE <sub>H</sub>	BG	CB <sub>H</sub>	BG
E9 <sub>H</sub>	FDCON	CC <sub>H</sub>	FDCON
EA <sub>H</sub>	FDSTEP	CD <sub>H</sub>	FDSTEP
EB <sub>H</sub>	FDRES	CE <sub>H</sub>	FDRES
		CF <sub>H</sub>	SCON1

## 12.2 本地互连网络（LIN）

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。

**注：** 仅 UART 具有 LIN 波特率检测特性。要使用 UART1 进行 LIN 通信，必须用软件实现分隔和同步字节的检测。

### 12.2.1 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、单主控制器 / 多从设备模式、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 12-6 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

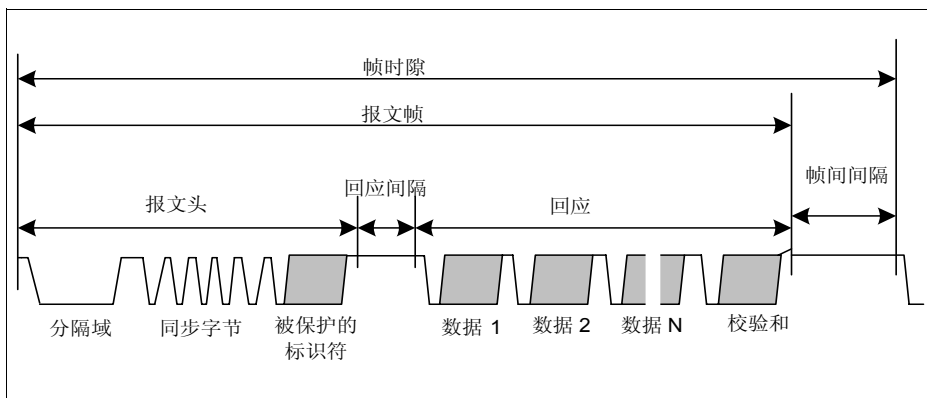


图 12-6 LIN 的帧结构

每个字节域串行发送，如图 12-7 所示。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）。

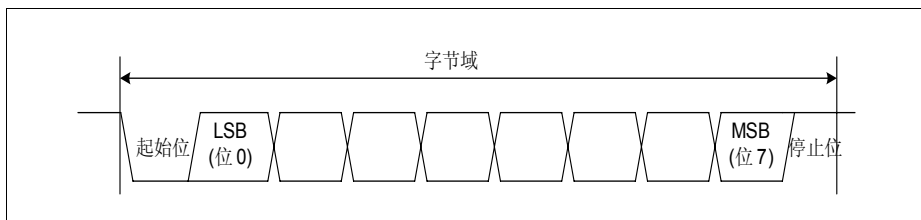


图 12-7 字节域结构

分隔域用作新的一帧的起始信令。只有该域不遵循图 12-7 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 12-8 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。

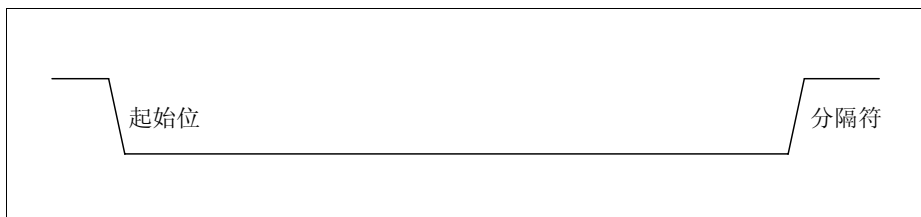


图 12-8 分隔域

同步字节是确定时间基准的特定序列，值为  $55_{\mu}$ ，如图 12-9 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 / 同步序列检测将中断正在进行的数据传送，开始新一帧数据的处理。

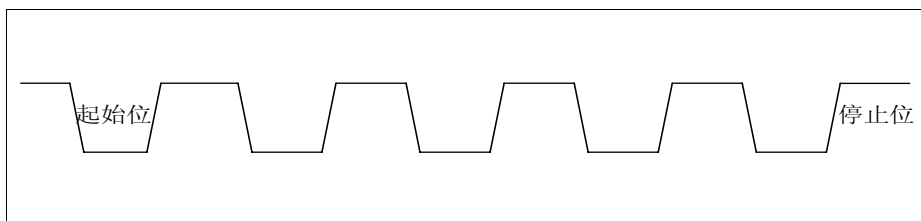


图 12-9 同步字节域

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，随后跟校验字节

### 12.2.2 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。

#### 12.2.2.1 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：初始化接收接口用于接收，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 12.2.2.2](#)，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。*

### 12.2.2.2 LIN 波特率检测

LIN 波特率检测特性通过定时器 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 FDCON.BRK、FDCON.EOFSYN 和 FDCON.ERRSYN。
- 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  $f_{T2}$ 。

LIN 的波特率检测如图 12-10 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

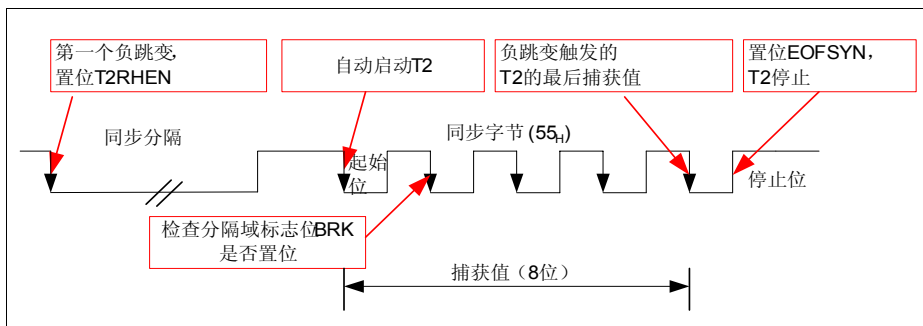


图 12-10 LIN 自动波特率检测

第 1 个下降沿到来时：

- 置位定时器 T2 外部启动使能位（T2MOD.T2RHEN）。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 位的位时间
- 检查分隔域标志位 FDCON.BRK

如果分隔域标志 FDCON.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（FDCON.EOFSYN）置位，定时器 T2 停止工作。T2 重载 / 捕获寄

---

串行接口

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 FDCON.BRK 未置位, 或者
- 同步字节错误标志 FDCON.ERRSYN 被置位, 或者
- 分隔域标志 FDCON.BRK 被置位, 但同步字节结束标志 FDCON.EOFSYN 和同步字节错误标志 FDCON.ERRSYN 未被置位。

### 12.3 高速同步串行接口

同步串行接口（SSC）支持全双工和半双工同步通信。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，从而能够和串行外设接口（SPI）兼容器件、或使用其它同步串行接口的器件进行通信。

通过 TXD 和 RXD 线发送和接收数据，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。数据发送、接收双缓存。

SSC 框图图 12-11 所示。

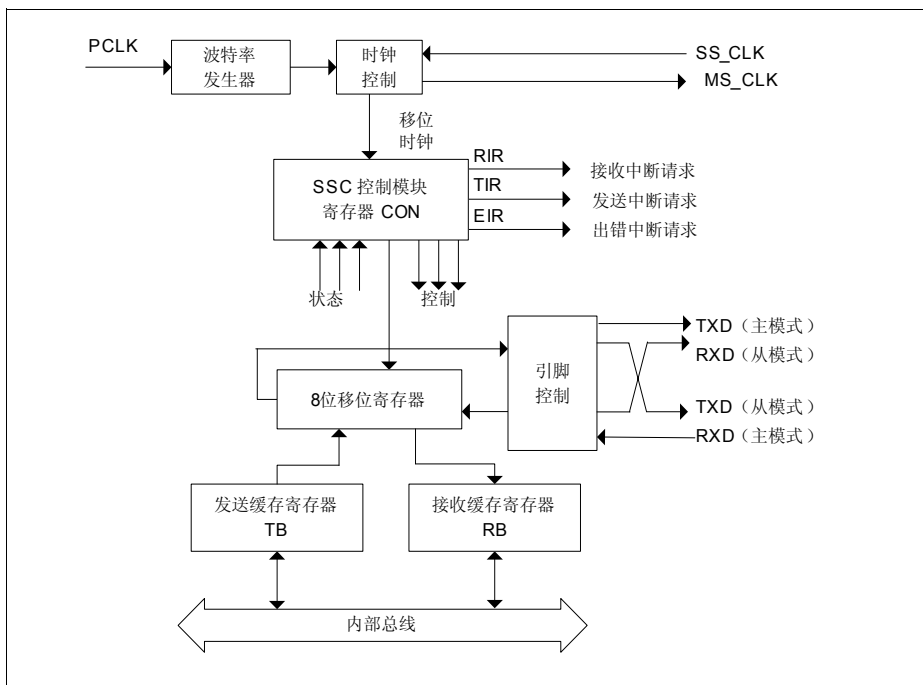


图 12-11 同步串行通道 SSC 框图



## 12.3.1 基本操作

### 12.3.1.1 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），访问控制位
- 工作工程中（CON.EN=1 使能 SSC），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被同步，发送接收同时发生，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），移位寄存器一空即将 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，忙碌标志 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。设定的数据位（2...8 位）传送完成之后，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 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
- 可根据模块时钟设定波特率范围
- 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。

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 **SPI** 接口兼容通信。

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 存放在寄存器的第 0 位。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 选择用上升沿或下降沿发送 / 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 12-12）。

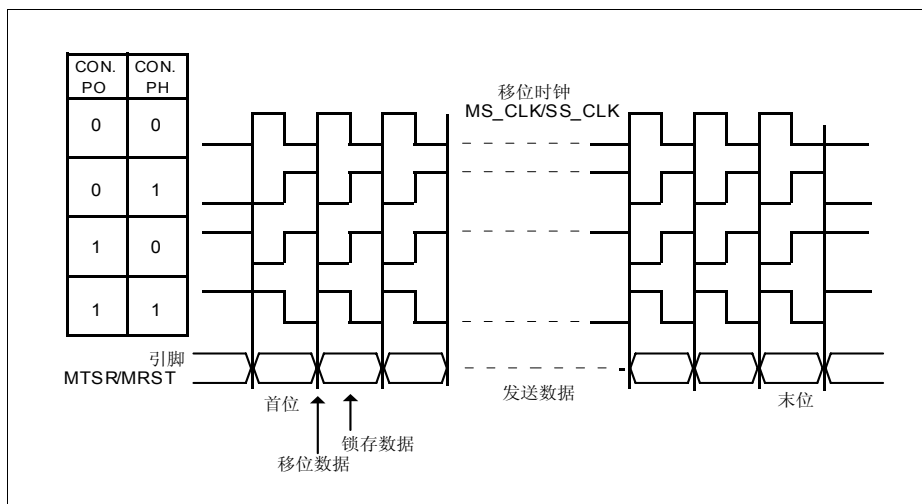


图 12-12 串行时钟相位和极性选择

对器件进行初始化用于串行通信时，必须选择一个器件作为主机、所有其它器件作为从机。

### 12.3.1.2 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。只有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。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。

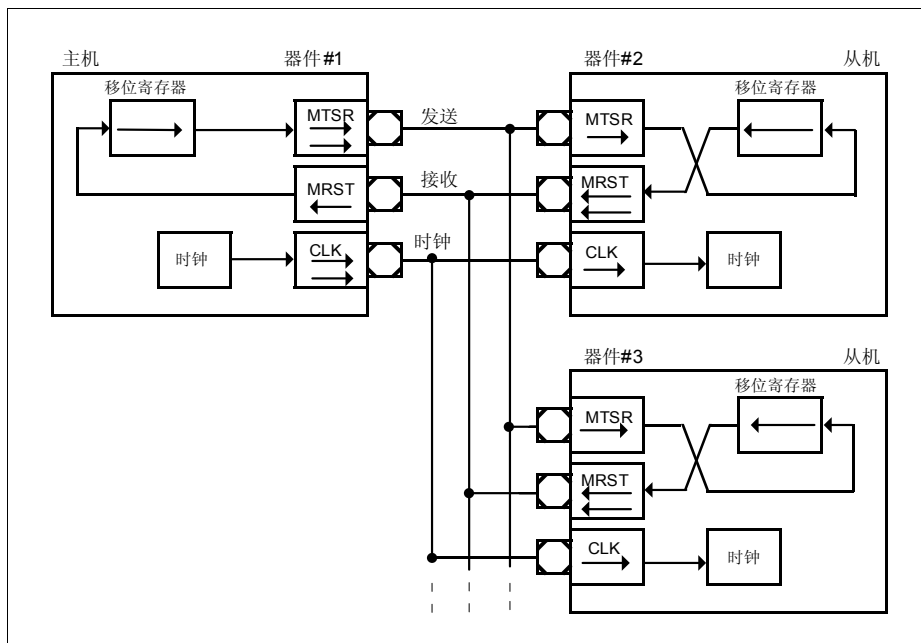


图 12-13 SSC 全双工设置

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 12-13 所示。数据传输时，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）接收线上的数据传输冲突：

- 只有一个从机驱动接收线，即开启该从机 **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，其它从机只可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指令。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指令。
- 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指令。

执行完必要的 **SSC** 的初始化，可使能串行接口。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**，启动数据传输。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

## 串行接口

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 **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。

随着传送开始，忙碌标志 **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。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。如果没有更多数据需要传送（**TB** 空），同时将对 **CON.BSY** 清零。该标志由硬件控制，不应由软件修改。

当被配置为从机器件工作时，一旦发送缓存寄存器的内容复制到移位寄存器中，**SSC** 会立刻在输出引脚上输出所选中的数据首位（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**。

*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，以免产生不希望的时钟跳变，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；由相关的 **ALTSEL** 寄存器准备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。*

### 12.3.1.3 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。数据交换线和每个器件的 **MTSR** 和 **MRST** 引脚连接，移位时钟线和 **SCLK** 引脚连接。

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。

和全双工模式相同，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
- 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遭破坏。

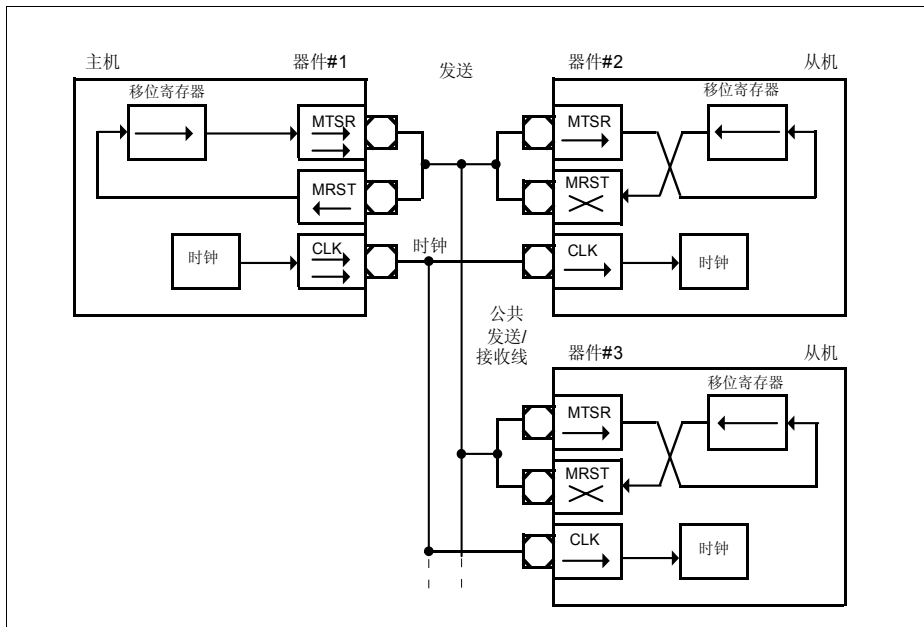


图 12-14 SSC 半双工设置

#### 12.3.1.4 连续传送

当发送中断请求标志被置位，说明发送缓存寄存器 **TB** 已空、并准备好可装入新发送数据。如果当前发送结束时已将新数据装入 **TB**，该数据立刻被移入移位寄存器，没有任何附加延时开始发送下一轮数据。在数据线上两帧数据无间隔连续传送。例如，传送两个字节看上去和传送一个字的效果相同。该特性用来和每次需要传送 **8** 位以上数据的器件接口通信，这只需软件规定数据帧的总长度。该特性还可用来和字节宽器件和字宽器件接口通信。

注：该特性只允许传送数据宽度为所设定基本数据宽度的整数倍，因为在运行过程中可能会禁止/使能 SSC 来重新设定基本数据宽度。

12.3.1.5 端口控制

SSC 使用三条线和外部通信，如图 12-15 所示。引脚 SCLK 用作时钟线；引脚 MRST 和 MTSR 用作串行数据的输入 / 输出线。

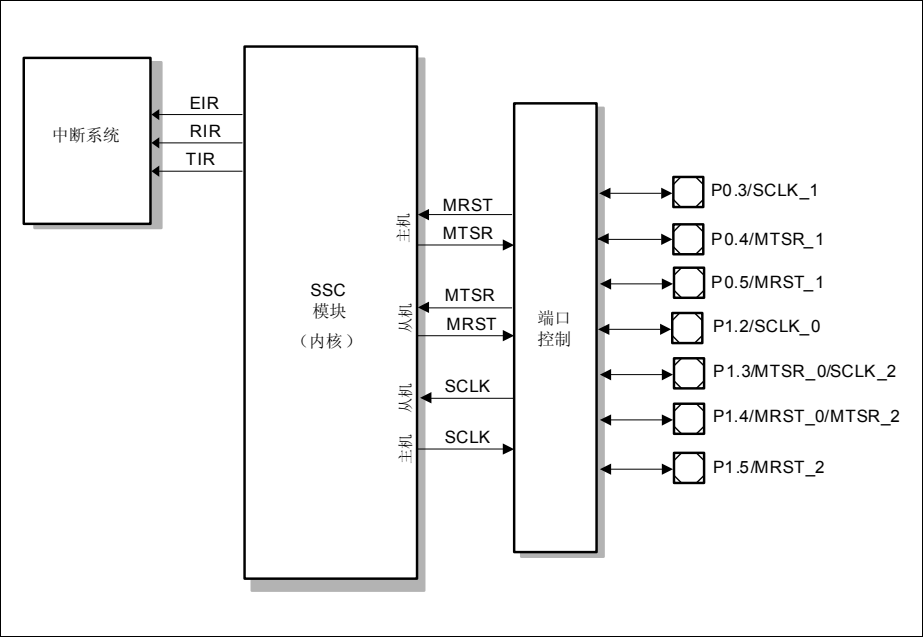


图 12-15 SSC 模块 I/O 接口

SSC 的 I/O 线操作取决于所选择的工作模式（主模式或从模式），端口线的方向也取决于工作模式。模式切换时 SSC 会自动使用正确的内核端口输出或输入线。

由于 SSC I/O 线和通用 I/O 口的双向线连接，使用软件 I/O 控制来控制分配给这些线的端口引脚。端口寄存器必须设置成复用输入输出功能。在主模式和从模式之间切换时，端口寄存器必须重设。在不同接收输入之间进行切换时，需要使用 MODPISEL3 寄存器。

MODPISEL3

端口输入选择寄存器 3

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	CIS	SIS	MIS				
r	rw	rw	rw				

符号	位	类型	描述
<b>MIS</b>	[1:0]	rw	<b>主模式接收输入选择</b> 00 选择接收输入 0 01 选择接收输入 1 10 选择接收输入 2 11 保留
<b>SIS</b>	[3:2]	rw	<b>从模式接收输入选择</b> 00 选择接收输入 0 01 选择接收输入 1 10 选择接收输入 2 11 保留
<b>CIS</b>	[5:4]	rw	<b>从模式时钟输入选择</b> 00 选择时钟输入 0 01 选择时钟输入 1 10 选择时钟输入 2 11 保留
<b>0</b>	[7:6]	r	<b>保留</b> 读操作返回 0，应写入 0。

### 12.3.1.6 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 12-16 所示。

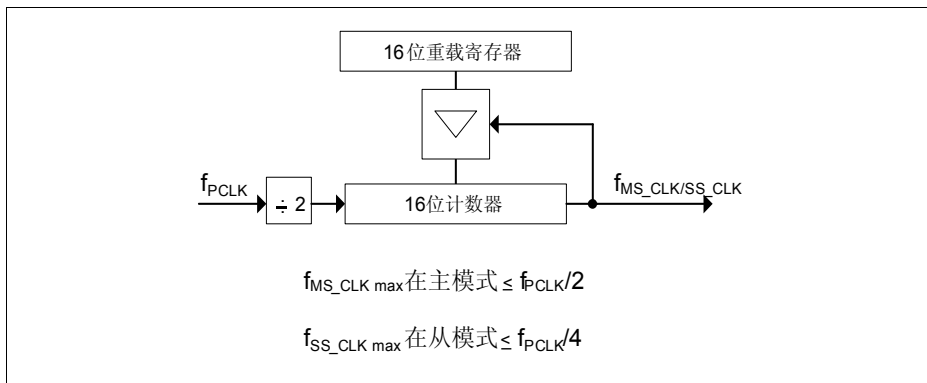


图 12-16 SSC 波特率发生器

波特率发生器的输入时钟来自模块时钟  $f_{PCLK}$ 。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 BR。

**注：** SSC 使能时绝不能对 BR 进行写操作。

下面的公式在给定重载值时计算波特率，或给定波特率计算重载值：

$$\text{波特率} = \frac{f_{PCLK}}{2 \times (<BR> + 1)} \quad \text{BR} = \frac{f_{PCLK}}{2 \times \text{波特率}} - 1$$

<BR> 代表重载寄存器的值（16 位无符号整数），波特率等于  $f_{MS\_CLK/SS\_CLK}$ ，如图 12-16 所示。

使用 24 MHz 的模块时钟，可获得的最大波特率为：主模式 12 MBaud (<BR>=0000<sub>H</sub>)，从模式 6 MBaud (<BR>=0001<sub>H</sub>)。

**表 12-6** 列出一些典型的波特率、相应的重载值以及波特率偏差（假设模块时钟为 24 MHz）。

**表 12-6 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
0000 <sub>H</sub>	12 MBaud （仅限主模式）	0.0%
0001 <sub>H</sub>	6 MBaud	0.0%



**表 12-6      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
0008 <sub>H</sub>	1.3 MBaud	0.0%
000B <sub>H</sub>	1 MBaud	0.0%
000F <sub>H</sub>	750 kBaud	0.0%
0011 <sub>H</sub>	666.7 kBaud	0.0%
0013 <sub>H</sub>	600 kBaud	0.0%
0017 <sub>H</sub>	500 kBaud	0.0%
002C <sub>H</sub>	266.7 kBaud	0.0%
003B <sub>H</sub>	200 kBaud	0.0%
0059 <sub>H</sub>	133.3 kBaud	0.0%
0077 <sub>H</sub>	100 kBaud	0.0%
FFFF <sub>H</sub>	183.11 Baud	0.0%

### 12.3.1.7 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，对应的错误标志位被 1 可被置位，激活出错中断请求线（EIR）产生出错中断请求（见图 12-17）。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）错误标志清零，以防止重复产生中断请求。

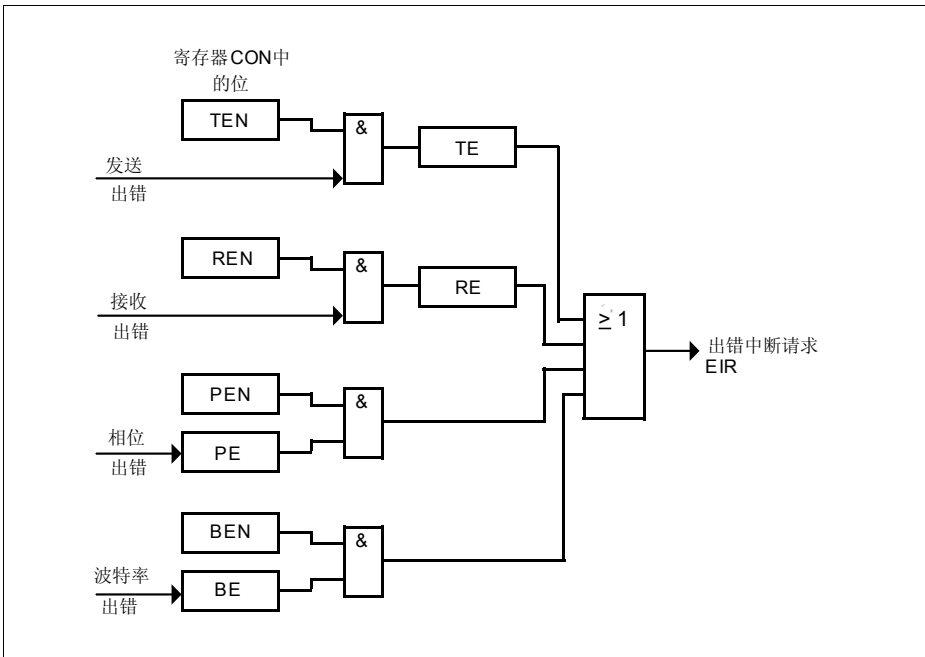


图 12-17 SSC 出错中断控制

**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 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 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“**FFF<sub>H</sub>**”装入从机发送缓存寄存器中。

注： 带有上拉/下拉输出驱动器、未被选中发送数据的从机，通常会关闭它的输出驱动器。但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： 错误状态标志 **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。

### 12.3.2 中断

SSC 中有三个中断源可触发中断：TIR、RIR 和 EIR。

SSC 的中断类型总结见表 12-7。

**表 12-7 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

SSC 模块中的三类中断源使用 [章节 5.1.2](#) 中的中断结构。寄存器 MODIEN 中的 3 位用于使能或禁止这些中断事件。

#### MODIEN

外设中断使能寄存器

复位值：07<sub>H</sub>

7	6	5	4	3	2	1	0
0			CM5EN	CM4EN	RIREN	TIREN	EIREN
r			rw	rw	rw	rw	rw

符号	位	类型	描述
EIREN	0	rw	<b>SSC 错误中断使能位</b> 0 禁止错误中断 1 使能错误中断
TIREN	1	rw	<b>SSC 发送中断使能位</b> 0 禁止发送中断 1 使能发送中断

符号	位	类型	描述
<b>RIREN</b>	2	rw	<b>SSC 接收中断使能位</b> 0 禁止接收中断 1 使能接收中断
<b>0</b>	[7:5]	r	<b>保留</b> 读操作返回 0，应写入 0。

12.3.3 低功耗模式

如果完全不需要 SSC 功能，可关闭其输入时钟最大限度的降低功耗。该模式通过置位寄存器 PMCON1 中的 SSC\_DIS 位来实现。外设时钟管理的具体描述请参见[章节 8.1.4](#)。

PMCON1

功率模式控制寄存器 1

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	CDC_DIS	CAN_DIS	MDU_DIS	T2CCU_DIS	CCU_DIS	SSC_DIS	ADC_DIS
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
SSC_DIS	1	rw	SSC 禁止请求位，高有效 0 SSC 处于正常工作状态（缺省值） 1 请求禁止 SSC
0	7	r	保留 读操作返回 0，应写入 0。

12.3.4 寄存器映射

内核 SFR 地址见[表 12-8](#)。

表 12-8 SFR 地址列表

地址	寄存器名称
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

## 12.3.5 寄存器描述

本章中描述的所有 SSC 寄存器名称，在本手册其它章节中引用时需添加模块名前缀“SSC\_”，如 SSC\_CONL。

### 12.3.5.1 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**

**CONL**

控制寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>LB</b>	<b>PO</b>	<b>PH</b>	<b>HB</b>	<b>BM</b>			
rw	rw	rw	rw	rw			

符号	位	类型	描述
<b>BM</b>	[3:0]	rw	<b>数据宽度选择</b> 0000 保留，不使用该组合 0001 - 0111 传送数据宽度为 2...8 位 (<BM>+1) <i>注： BM[3] 固定为 0。</i>
<b>HB</b>	4	rw	<b>报文头控制</b> 0 先发送 / 接收 LSB 1 先发送 / 接收 MSB
<b>PH</b>	5	rw	<b>时钟相位控制</b> 0 在时钟的前沿移位输出发送数据，后沿锁存接收数据 1 在时钟的前沿锁存接收数据，后沿移位输出发送数据
<b>PO</b>	6	rw	<b>时钟极性控制</b> 0 时钟线低电平空闲，时钟前沿为低到高跳变 1 时钟线高电平空闲，时钟前沿为高到低跳变

**串行接口**

符号	位	类型	描述
<b>LB</b>	7	rw	<b>回环控制</b> 0 正常输出 1 接收输入和发送输出相连（半双工模式）

**CONH**
**控制寄存器，高位字节**
**复位值：00<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>EN</b>	<b>MS</b>	<b>0</b>	<b>AREN</b>	<b>BEN</b>	<b>PEN</b>	<b>REN</b>	<b>TEN</b>
rw	rw	r	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>TEN</b>	0	rw	<b>发送出错中断使能</b> 0 禁止发送出错中断 1 使能发送出错中断
<b>REN</b>	1	rw	<b>接收出错使能</b> 0 禁止接收出错中断 1 使能接收出错中断
<b>PEN</b>	2	rw	<b>相位出错使能</b> 0 禁止相位出错中断 1 使能相位出错中断
<b>BEN</b>	3	rw	<b>波特率出错使能</b> 0 禁止波特率出错中断 1 使能波特率出错中断
<b>AREN</b>	4	rw	<b>自动复位使能</b> 0 波特率出错时无自动复位操作 1 波特率出错时 <b>SSC</b> 被自动复位
<b>MS</b>	6	rw	<b>主机选择</b> 0 从模式。从 <b>SCLK</b> 接收移位时钟 1 主模式。产生移位时钟并从 <b>SCLK</b> 输出
<b>EN</b>	7	rw	<b>使能位 = 0</b> 禁止发送和接收。访问控制位。
<b>0</b>	5	r	<b>保留</b> 读操作返回 0，应写入 0。



**CON.EN = 1: 工作模式**
**CONL**

控制寄存器，低位字节

 复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
0				BC			
r				rh			

符号	位	类型	描述
<b>BC</b>	[3:0]	rh	位计数域 0001 - 1111 每次移位时更新移位计数器
<b>0</b>	[7:4]	r	保留 读操作返回 0，应写入 0。

**CONH**

控制寄存器，高位字节

 复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>EN</b>	<b>MS</b>	<b>0</b>	<b>BSY</b>	<b>BE</b>	<b>PE</b>	<b>RE</b>	<b>TE</b>
rw	rw	r	rh	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>TE</b>	0	rwh	发送出错标志 0 发送未出错 1 从机发送缓存寄存器未更新即开始传送数据
<b>RE</b>	1	rwh	接收出错标志 0 接收未出错 1 接收缓存寄存器的内容在读出之前，新数据已接收完毕
<b>PE</b>	2	rwh	相位出错标志 0 相位未出错 1 接收数据在采样时钟沿附近改变

## 串行接口

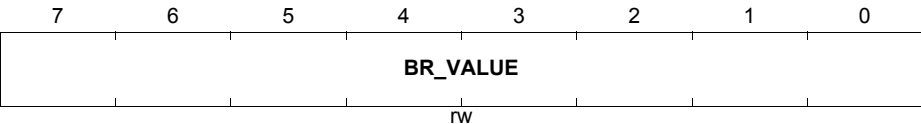
符号	位	类型	描述
<b>BE</b>	3	rwh	<b>波特率出错标志</b> 0 波特率未出错 1 从机的实际波特率超过期望波特率的 2 倍或低于 0.5 倍
<b>BSY</b>	4	rh	<b>忙碌标志</b> 传送过程中被置位。
<b>MS</b>	6	rw	<b>主机选择位</b> 0 从模式。从 SCLK 接收移位时钟 1 主模式。产生移位时钟并从 SCLK 输出
<b>EN</b>	7	rw	<b>使能位 = 1</b> 发送和接收被使能。存取状态标志和主机 / 从机控制。
<b>0</b>	5	r	<b>保留</b> 读操作返回 0，应写入 0。

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。

### 12.3.5.2 波特率定时器重载寄存器

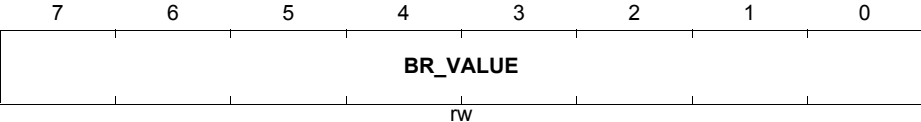
SSC 的波特率定时器重载寄存器 BR 中包含波特率定时器的 16 位重载值。

**BRL**  
波特率定时器重载寄存器，低位字节 复位值：00<sub>H</sub>



符号	位	类型	描述
<b>BR_VALUE</b>	[7:0]	rW	<b>波特率定时器 / 重载寄存器的值 [7:0]</b> 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。

**BRH**  
波特率定时器重载寄存器，高位字节 复位值：00<sub>H</sub>



符号	位	类型	描述
<b>BR_VALUE</b>	[7:0]	rW	<b>波特率定时器 / 重载寄存器的值 [15:8]</b> 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。

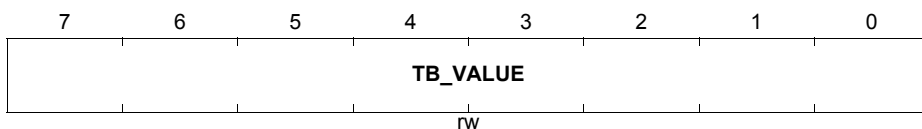
### 12.3.5.3 发送和接收缓存寄存器

SSC 的发送缓存寄存器 TB 中存放发送数据。

#### TBL

发送缓存寄存器，低位字节

复位值：00<sub>H</sub>



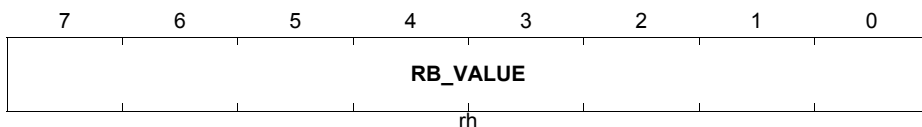
符号	位	类型	描述
<b>TB_VALUE</b>	[7:0]	rw	发送数据寄存器的值 TB_VALUE 是将发送的数据值，发送时 TB 中未被选中的位不予理睬。

SSC 的接收缓存寄存器 RB 中存放接收数据。

#### RBL

接收缓存寄存器，低位字节

复位值：00<sub>H</sub>



符号	位	类型	描述
<b>RB_VALUE</b>	[7:0]	rh	接收数据寄存器的值 RB 中存放接收数据 RB_VALUE，RB 中未被选中的位无效，不予理睬。

## 13 定时器

XC878 提供了 4 个 16 位定时器：定时器 T0、定时器 T1、定时器 T2 和定时器 T21，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。

### 定时器 T0 和定时器 T1 工作特性：

- 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

### 定时器 T2 和定时器 T21 工作特性：

- 递增 / 递减计数可选择
- 16 位自动重载模式
- 单通道、16 位捕获模式

注：定时器 T2 请参考 T2CCU 模块（定时器 T2 捕获 / 比较单元）中的 16 位通用定时器。T2CCU 中 CCU 操作的详情，请参考第 14 章。

## 13.1 定时器 T0 和定时器 T1

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 2 个 PCLK），T0 和 T1 加 1。用作计数器时，定时器 T0 和定时器 T1 对应的外部引脚 T0 或 T1 上每次发生 1 到 0 的跳变时（下降沿），T0 或 T1 加 1。

### 13.1.1 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（使定时器工作）；通过输入选择位 TMOD 来决定定时器的输入来自内部系统时钟或者外部引脚。

注：本章中的“x”（如：TCON.TRx）表示 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>μ</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。进入中断服务程序后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 13.1.2.4](#)。

### 外部控制

定时器的开启 / 关闭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，直接将信号送入内核。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。

若 TMOD.GATEx 被清零，定时器将恢复由软件控制。

### 13.1.2 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 13-1](#)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 13-1 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 级联构成一个 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。

### 13.1.2.1 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 13-1 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。

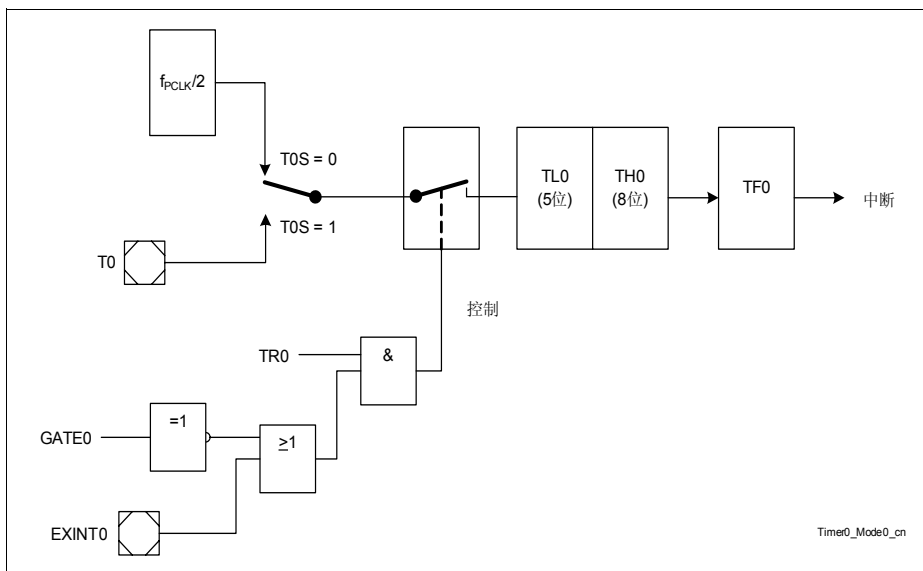


图 13-1 定时器 T0，模式 0：13 位定时器 / 计数器



### 13.1.2.2 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 13-2 所示。

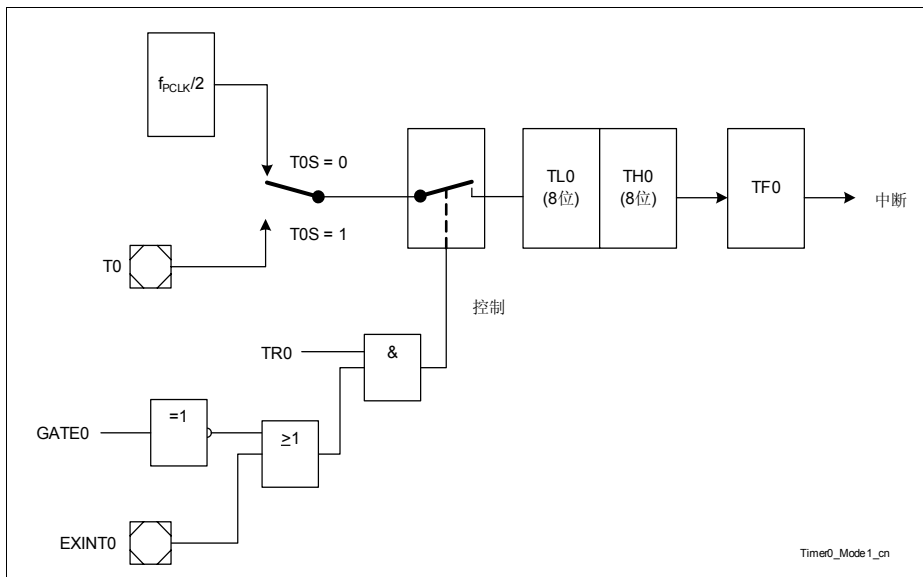


图 13-2 定时器 T0，模式 1：16 位定时器 / 计数器



### 13.1.2.4 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 13-4 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。

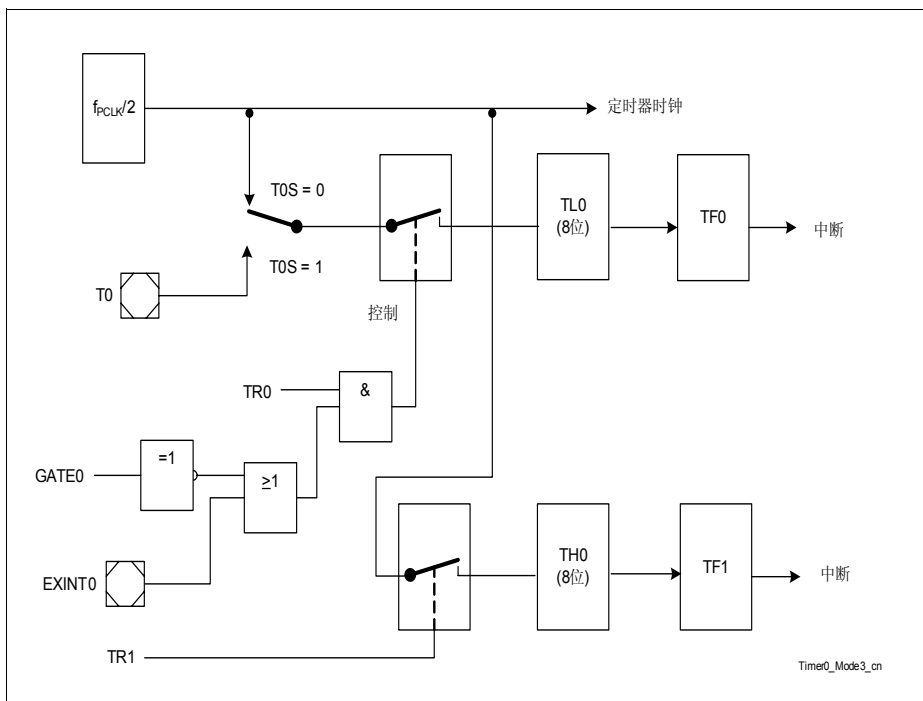


图 13-4 定时器 T0，模式 3：两个 8 位定时器 / 计数器

13.1.3 端口控制

定时器 T0 和定时器 T1 用作事件计数器时，将分别对外部输入引脚 T0 和 T1 上 1 到 0 的跳变进行计数。对于定时器 T0，有两个不同的外部输入引脚 T0\_0 和 T0\_1 可供选择；对于定时器 T1，有两个不同的外部输入引脚 T1\_0 和 T1\_1 可供选择。输入引脚选择通过 MODPISEL2.T0IS 和 MODPISEL2.T1IS 来设定。

MODPISEL2

外设输入选择寄存器 2

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
	0		T2EXIS	T21IS	T2IS	T1IS	T0IS
	r		rw	rw	rw	rw	rw

符号	位	类型	描述
T0IS	0	rw	定时器 T0 输入选择 0 选择定时器 T0 的输入 T0_0 1 选择定时器 T0 的输入 T0_1
T1IS	1	rw	定时器 T1 输入选择 0 选择定时器 T1 的输入 T1_0 1 选择定时器 T1 的输入 T1_1
0	[7:5]	r	保留 读操作返回 0，应写入 0。

### 13.1.4 寄存器映射

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 13-2** 列出这些 SFR 的地址。

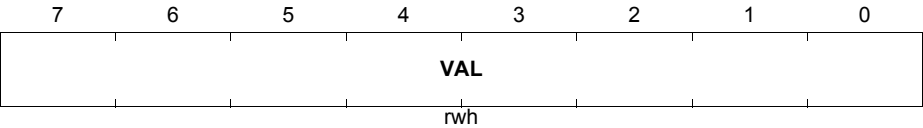
**表 13-2 寄存器映射**

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

### 13.1.5 寄存器描述

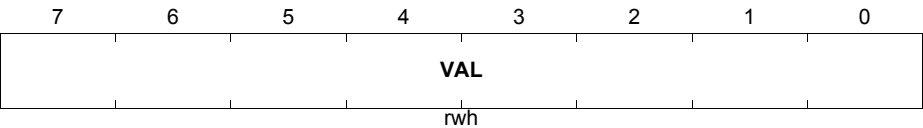
根据选择的工作模式，将定时器 T0 和定时器 T1 的低位字节（TL0，TL1）和高位字节寄存器（TH0，TH1）组成一个定时器。寄存器 TCON 控制定时器 T0 和定时器 T1 的操作。寄存器 TMOD 中包含定时器 T0 和定时器 T1 的工作模式选择控制位。IEN0 寄存器中包含定时器 T0 和定时器 T1 的中断使能控制位。

**TLx (x = 0 - 1)**  
**定时器 x 寄存器，低位字节** **复位值：00<sub>H</sub>**



符号	位	类型	描述
<b>TLx.VAL(x = 0, 1)</b>	[7:0]	rwh	<b>定时器 T0/T1 低位字节寄存器</b> 00 TLx 保存 5 位预分频因子 01 TLx 保存 16 位定时器 / 计数器的低 8 位 10 TLx 保存 8 位定时器值 11 TL0 保存 8 位定时器值；未使用 TL1。

**THx (x = 0 - 1)**  
**定时器 x 寄存器，高位字节** **复位值：00<sub>H</sub>**



符号	位	类型	描述
<b>THx.VAL(x = 0, 1)</b>	[7:0]	rwh	<b>定时器 T0/T1 高位字节寄存器</b> 00 THx 保存 8 位预分频因子 01 THx 保存 16 位定时器 / 计数器的高 8 位 10 THx 保存 8 位重载值 11 TH0 保存 8 位定时器值；未使用 TH1

## 定时器

### TCON

定时器 T0/T1 控制寄存器

复位值: 00<sub>H</sub>

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 定时器关闭 1 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。当主机调用中断服务程序时溢出标志由硬件清零。
TR1	6	rw	定时器 T1 运行控制 0 定时器关闭 1 定时器运行 <i>注：若定时器工作在模式 3，该控制位也影响 TH0。</i>
TF1	7	rwh	定时器 T1 溢出标志 定时器 T1 <sup>1)</sup> 溢出时由硬件置位。当主机调用中断服务程序时溢出标志有硬件清零。

<sup>1)</sup> 若定时器 T0 工作在模式 3，TF1 由 TH0 置位。

### TMOD

定时器模式寄存器

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
GATE1	T1S	T1M	GATE0	T0S	T0M		
rw	rw	rw	rw	rw	rw		

**定时器**

符号	位	类型	描述
<b>T0M[1:0]</b>	[1:0]	rw	<b>模式选择位</b> 00 13 位定时器（M8048 兼容模式） 01 16 位定时器 10 8 位自动重载定时器 11 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>T1M[1:0]</b>	[5:4]	rw	<b>模式选择位</b> 00 13 位定时器（M8048 兼容模式） 01 16 位定时器 10 8 位自动重载定时器 11 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>T0S</b>	2	rw	<b>定时器 T0 选择</b> 0 输入来自内部系统时钟 1 输入来自 T0 引脚
<b>GATE0</b>	3	rw	<b>定时器 T0 门控标志</b> 0 仅当 TCON.TR0 = 1 时定时器 T0 才运行（软件控制） 1 仅当引脚 EXINT0 = 1（硬件控制）且 TCON.TR0 = 1 时定时器 T0 才运行
<b>T1S</b>	6	rw	<b>定时器 T1 选择</b> 0 输入来自内部系统时钟 1 输入来自 T1 引脚
<b>GATE1</b>	7	rw	<b>定时器 T1 门控标志</b> 0 仅当 TCON.TR1 = 1 时定时器 T1 才运行（软件控制） 1 仅当引脚 EXINT1 = 1（硬件控制）且 TCON.TR1 = 1 时定时器 T1 才运行



IEN0

中断使能寄存器

复位值: 00<sub>H</sub>

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

符号	位	类型	描述
ET0	1	rw	定时器 T0 溢出中断使能 0 禁止定时器 T0 中断 1 使能定时器 T0 中断
ET1	3	rw	定时器 T1 溢出中断使能 0 禁止定时器 T1 中断 1 使能定时器 T1 中断 <i>注: 当定时器 T0 工作在模式 3 时, 该中断表明定时器 T0 的 TH0 寄存器溢出。</i>

## 13.2 定时器 T2 和定时器 T21

定时器 T2（指的是 T2CCU 模块中的定时器 T2）和定时器 T21 是功能相同的 16 位通用定时器。定时器 T2 工作在 T2CCU 时钟频率  $f_{T2CCU}$ ，该频率为 PCLK 频率或者 PCLK 频率的 2 倍。定时器 T21 工作在 PCLK 频率。这两个定时器均可工作在两种操作模式：16 位自动重载模式和 16 位单通道捕获模式。每种模式下，它们都可用作定时器或计数器功能。用作定时器时，定时器 T2 以  $f_{T2CCU}/12$  的输入时钟（若预分频被禁止）计数，定时器 T21 以  $f_{PCLK}/12$  的输入时钟计数；用作计数器时，对引脚 T2 上（对应于定时器 T2）和引脚 T21 上（对应于定时器 T21）的 1 到 0 的跳变计数。计数器模式下，定时器 T2 的最大计数精度为  $f_{T2CCU}/24$ （若预分频被禁止），定时器 T21 的最大计数精度为  $f_{PCLK}/24$ 。

*注：以下各节将描述定时器 T2 的功能，除非特别声明，这些描述对定时器 T21 也适用。T2CCU 模块中的 CCU 功能的描述，请参考第 14 章。*

### 13.2.1 定时器基本操作

定时器 T2 可通过位 TR2 由硬件或软件启动。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置位运行位 TR2。只能由软件复位 TR2 终止定时器 T2 工作。

### 13.2.2 自动重载模式

寄存器 T2CON 中的  $CP/\overline{RL}2$  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，开始新一轮计数循环。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。

#### 13.2.2.1 禁止递增 / 递减计数

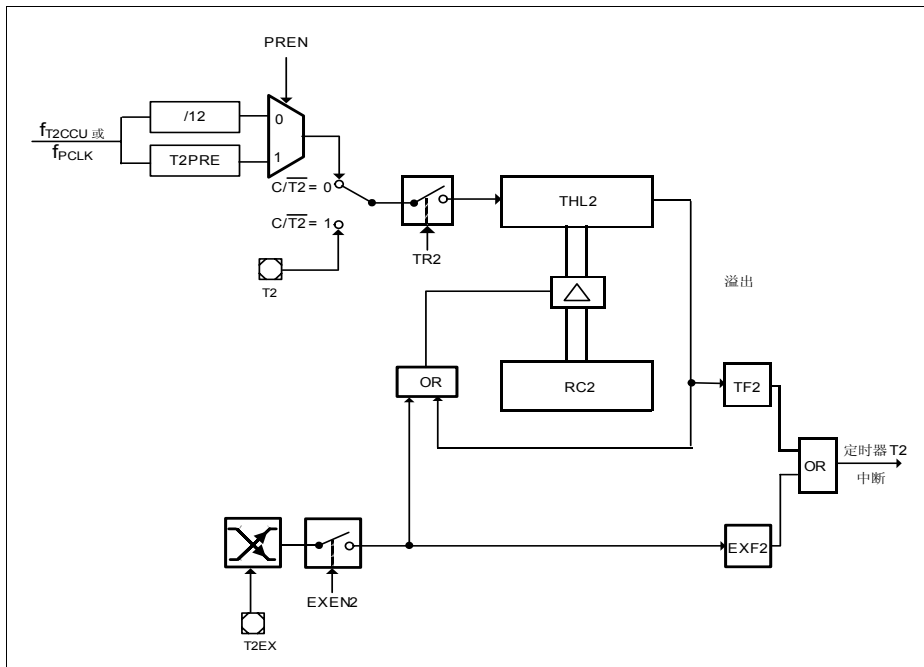
如果 DCEN = 0，递增 / 递减计数选择被禁止，因此定时器只能递增计数。工作原理如图 13-5 所示。

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 FFFFH 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循环，定时器同上一轮计数循环一样，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。如果寄存器 T2CON1 中的相关使能位 EXF2EN/TF2EN 置位，这两种情况均会产生中断，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，跳变沿选择由位 T2REGS 决定。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，跳变沿由位 EDGSEL 选择。



**图 13-5 自动重载模式 (DCEN = 0)**

### 13.2.2.2 使能递增 / 递减计数

若  $\text{DCEN} = 1$ , 可进行递增 / 递减计数选择。计数方向由输入引脚  $\text{T2EX}$  的电平决定。工作原理如图 13-6 所示。

引脚 T2EX 的逻辑电平为 1 时, 定时器 T2 递增计数, 定时器递增计数至最大值  $FFFF_H$  后溢出, 置位 TF2, RC2 寄存器中的 16 位重载值重新装入定时器寄存器。然后开始新一轮计数循环, 定时器同上一轮计数循环一样, 从重载值开始递增计数。在溢出前该重载值已由软件设置。

引脚 T2EX 的逻辑电平为 0 时, 定时器 T2 递减计数。定时器递减计数并当 THL2 的值和寄存器 RC2 中的值相等时发生下溢。下溢后置位 TF2, 并将值 FFFF<sub>H</sub> 重新载入定时器寄存器 THL2 中。然后开始新一轮计数循环, 定时器同上一轮计数循环一样递减计数。

如果位 T2RHEN 置位，定时器 T2 由引脚 T2EX 的上升沿（T2REGS = 1）启动，开始递增计数；或由 T2EX 的下降沿（T2REGS = 0）启动，开始递减计数。

该模式下，定时器 T2 产生上溢或下溢出时 EXF2 翻转，但该标志不会产生中断请求。

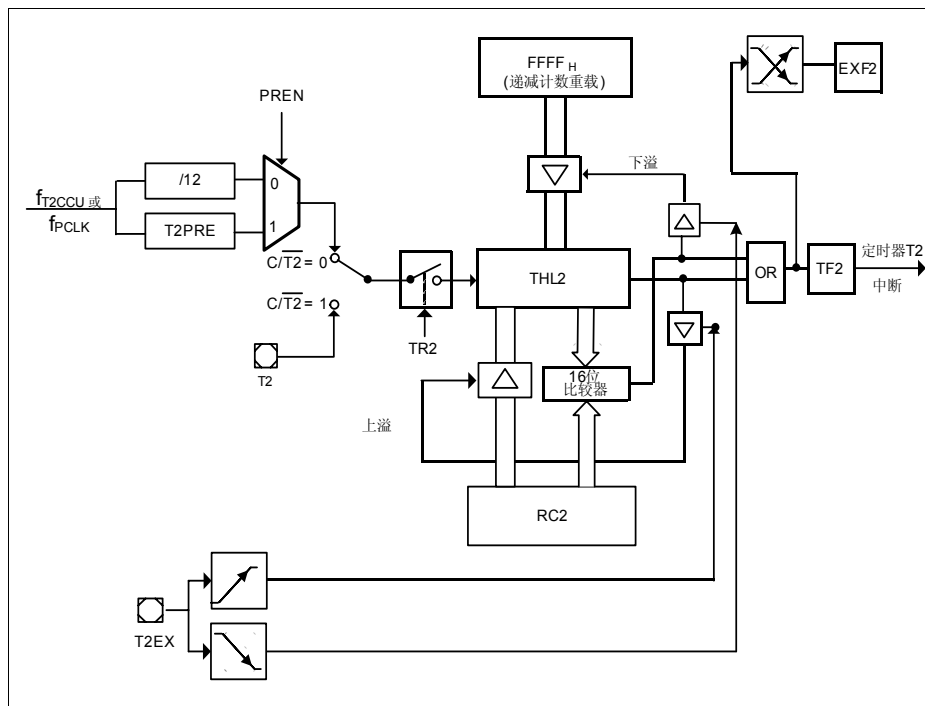


图 13-6 自动重载模式 (DCEN = 1)

### 13.2.3 捕获模式

将寄存器 T2CON 中的 CP/RL2 和 EXEN2 置位，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 FFFF<sub>H</sub> 后溢出，溢出后置位 TF2、并将 0000<sub>H</sub> 重新装入定时器寄存器。TF2 置位会向 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 T2CCU 时钟周期对外部输入采样。如果在一个 T2CCU 时钟周期采样值为低（高）电平、下一个 T2CCU 时钟周期采样值为高（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，从而确保总能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 / 正跳变将触发捕获操作，跳变沿由位 EDGESEL 选择。

执行完捕获操作后，EXF2 置位、可用来产生中断请求。定时器 T2 的捕获功能如图 13-7 所示。

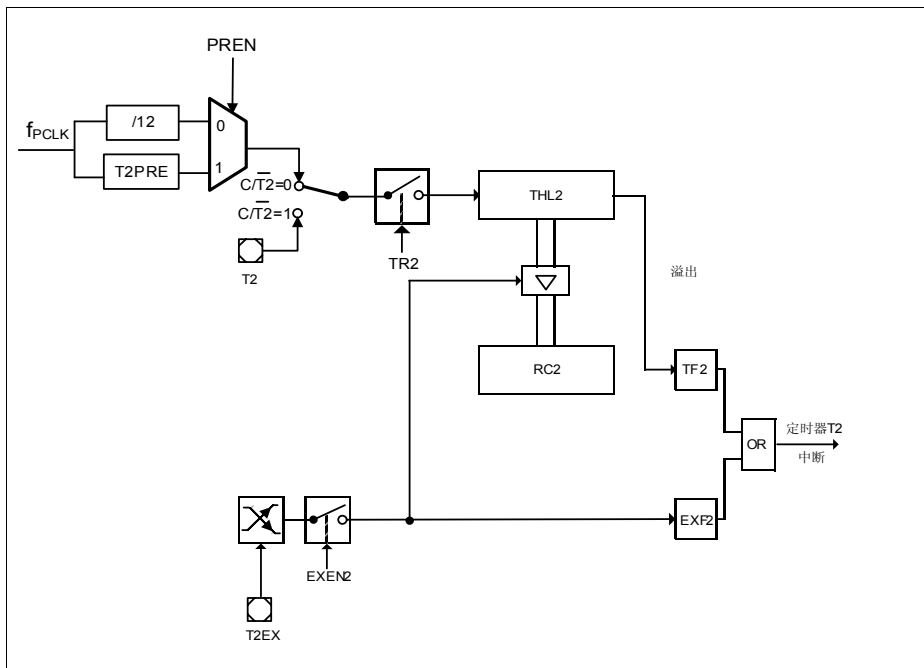


图 13-7 捕获模式

### 13.2.4 计数时钟

自动重载模式下的计数时钟由寄存器 **T2CON** 中的位  $\overline{C/T2}$  来选择，如果  $\overline{C/T2} = 0$ ，使用计数时钟  $f_{T2CCU}/12$ （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** 上的负跳变（1 到 0）进行计数。计数器在连续两个 **T2CCU** 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** 控制。

注： **C501** 兼容特性要求计数精度至少为 24 个时钟周期。

### 13.2.5 外部中断功能

当定时器/计数器功能被禁止时 ( $TR2 = 0$ )，只要 **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。

### 13.2.6 端口控制

定时器 **T2** 和定时器 **T21** 用作事件计数器时，将分别对外部输入引脚 **T2** 和 **T21** 上 1 到 0 的跳变进行计数。对于定时器 **T2**，有两个不同的外部输入引脚 **T2\_0** 和 **T2\_1** 可供选择；对于定时器 **T21**，有两个不同的外部输入引脚 **T21\_0** 和 **T21\_1** 可供选择。输入引脚选择通过 **MODPISEL2.T2IS** 和 **MODPISEL2.T21IS** 来设定。另外，各有两个外部输入源可与定时器 **T2** 的 **T2EX** 引脚和定时器 **T21** 的 **T21EX** 引脚相连，由 **MODPISEL2.T2EXIS** 和 **MODPISEL1.T21EXIS** 来设定。

#### MODPISEL2

外设输入选择寄存器 2

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0			T2EXIS	T21IS	T2IS	T1IS	T0IS
r			rw	rw	rw	rw	rw

符号	位	类型	描述
T2IS	2	rw	定时器 T2 输入选择 0 选择定时器 T2 的输入 T2_0 1 选择定时器 T2 的输入 T2_1
T21IS	3	rw	定时器 T21 输入选择 0 选择定时器 T21 的输入 T21_0 1 选择定时器 T21 的输入 T21_1
T2EXIS	4	rw	T2 外部输入选择 0 选择定时器 T2 的输入 T2EX_0 1 选择定时器 T2 的输入 T2EX_1
0	[7:5]	r	保留 读操作返回 0；应写入 0。

## MODPISEL1

外设输入选择寄存器 1

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
EXINT6IS			UR1RIS		T21EXIS	0	
rw			rw		rw	r	

符号	位	类型	描述
T21EXIS	2	rw	T21 外部输入选择 0 选择 T21 的输入 T21EX_0 1 选择 T21 的输入 T21EX_1
0	[1:0]	r	保留 读操作返回 0；应写入 0。

### 13.2.7 低功耗模式

如果完全不需要定时器 T2（以及捕获 / 比较操作）和定时器 T21 的功能，可关闭其输入时钟最大限度降低功耗，该模式通过置位寄存器 PMCON1 中的位 T2CCU\_DIS、寄存器 PMCON2 中的位 T21\_DIS 来实现。外设时钟管理的具体描述请参见[章节 8.1.4](#)。

## PMCON1

功率模式控制寄存器 1

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	CDC_DIS	CAN_DIS	MDU_DIS	T2CCU_DIS	CCU_DIS	SSC_DIS	ADC_DIS
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
T2CCU_DIS	3	rw	定时器 T2 捕获 / 比较单元禁止请求位，高有效 0 T2CCU 正常工作（缺省值） 1 请求禁止 T2CCU
0	7	r	保留 读操作返回 0；应写入 0。



## PMCON2

### 功率模式控制寄存器 2

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0						UART1_DIS	T21_DIS
r						rw	rw

符号	位	类型	描述
T21_DIS	0	rw	定时器 T21 禁止请求位，高有效 0 定时器 T21 正常工作（缺省值） 1 请求禁止定时器 T21
0	[7:2]	r	保留 读操作返回 0，应写入 0。

### 13.2.8 模式挂起控制

当 OCDS 进入监控模式（见[章节 18.3](#)），可通过置位 SFR MODSUSP 中的模块挂起控制位 T2SUSP，终止定时器 T2 计数。

定时器挂起时，其设置和配置不会改变。

## MODSUSP

### 模式挂起控制寄存器

复位值：01<sub>H</sub>

7	6	5	4	3	2	1	0
0	CCTSUSP	T21SUSP	T2SUSP	T13SUSP	T12SUSP	WDTSUSP	
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
T2SUSP	3	rw	T2 调试挂起位 0 定时器 T2 不被挂起 1 定时器 T2 被挂起
T21SUSP	4	rw	T21 调试挂起位 0 定时器 T21 不被挂起 1 定时器 T21 被挂起
0	[7:6]	r	保留 读操作返回 0，应写入 0。

### 13.2.9 寄存器映射

定时器 T2 和定时器 T21 所包含的 SFR 相同。

以下各节中描述的所有定时器 T2 的寄存器名称，在本手册其它章节中引用时需添加模块名前缀“T2\_”，例如 T2\_T2CON；同样，定时器 T21 需添加模块名前缀“T21\_”，例如 T21\_T2CON。

定时器 T2 的 SFR 位于标准（非映射）SFR 区的页 0；定时器 T21 的 SFR 位于映射 SFR 区。T2 和 T21 中相同功能 SFR 位于相同的地址单元。表 13-3 列出这些寄存器的地址。

**表 13-3 SFR 地址列表**

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

### 13.2.10 寄存器描述

寄存器 T2MOD 用来设置定时器 T2 工作在不同模式。

#### T2MOD

定时器 T2 模式寄存器

复位值：00<sub>H</sub>

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 禁止递增 / 递减计数功能 1 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 预分频位</b> 选择定时器 T2 的输入时钟（来自 T2CCU 时钟） 000 $f_{T2} = f_{T2CCU}$ 001 $f_{T2} = f_{T2CCU}/2$ 010 $f_{T2} = f_{T2CCU}/4$ 011 $f_{T2} = f_{T2CCU}/8$ 100 $f_{T2} = f_{T2CCU}/16$ 101 $f_{T2} = f_{T2CCU}/32$ 110 $f_{T2} = f_{T2CCU}/64$ 111 $f_{T2} = f_{T2CCU}/128$ <i>注： T2CCU 时钟为 PCLK 或 PCLK 频率的 2 倍。 定时器 T21 的时钟为 PCLK。</i>
PREN	4	rw	<b>预分频使能</b> 0 禁止预分频功能， 12 分频器生效 1 使能预分频功能（见位 T2PRE）， 12 分频器旁路
EDGESEL	5	rw	<b>捕获模式 / 重载模式的边沿选择</b> 0 选择引脚 T2EX 的下降沿 1 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 禁止定时器 T2 外部启动 1 使能定时器 T2 外部启动

# 定时器

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 选择引脚 T2EX 的下降沿 1 选择引脚 T2EX 的上升沿

寄存器 T2CON 控制定时器 T2 的工作模式。此外还包含中断状态标志。

## T2CON

### 定时器 T2 控制寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>TF2</b>	<b>EXF2</b>	<b>0</b>	<b>EXEN2</b>	<b>TR2</b>	<b>C/T2</b>	<b>CP/RL2</b>	
rw	rw	r	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>CP/RL2</b>	0	rw	<b>捕获 / 重载选择</b> 0 计数器溢出或引脚 T2EX 发生负 / 正跳变时重载（当 EXEN2 = 1）。 1 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）。由位 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 选择定时器功能 1 对引脚 T2 的负跳变计数
<b>TR2</b>	2	rw	<b>定时器 T2 启动 / 停止控制</b> 0 停止定时器 T2 1 启动定时器 T2
<b>EXEN2</b>	3	rw	<b>定时器 T2 外部使能控制</b> 0 禁止外部事件 1 捕获 / 重载模式下使能外部事件
<b>EXF2</b>	6	rw	<b>定时器 T2 外部事件标志</b> 捕获 / 重载模式下，若 EXEN2 = 1，在引脚 T2EX 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 DCEN = 1 时，不产生送入内核的中断请求。</i>

**定时器**

符号	位	类型	描述
<b>TF2</b>	7	rwh	<b>定时器 T2 上溢 / 下溢标志</b> 定时器 T2 上溢 / 下溢时该位置位。必须由软件清零。
<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。

**T2CON1**
**定时器 T2 控制寄存器 1**
**复位值：03<sub>H</sub>**

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 禁止外部中断 1 使能外部中断
<b>TF2EN</b>	1	rw	<b>上溢 / 下溢中断使能</b> 0 禁止上溢 / 下溢中断 1 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。

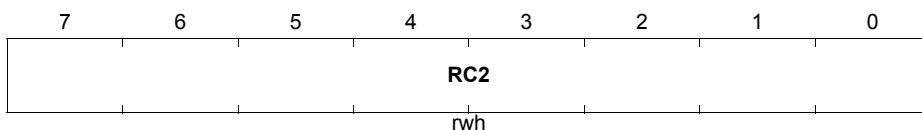
## 定时器

根据所选的工作模式不同，RC2 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。

### RC2L

定时器 T2 重载 / 捕获寄存器，低位字节

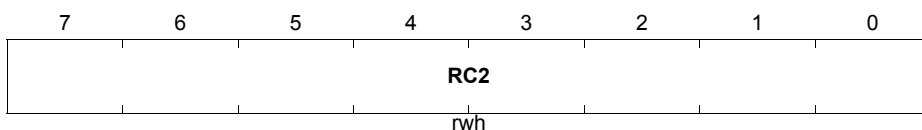
复位值：00<sub>H</sub>



符号	位	类型	描述
RC2	[7:0]	rwh	<b>重载 / 捕获值 [7:0]</b> 若 CP/RL2 = 0，定时器溢出时将该寄存器的值装入定时器寄存器。 若 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装入该寄存器。

### RC2H

定时器 T2 重载捕获寄存器，高位字节 复位值：00<sub>H</sub>



符号	位	类型	描述
RC2	[7:0]	rwh	<b>重载 / 捕获值 [15:8]</b> 若 CP/RL2 = 0，定时器溢出时将该寄存器的值装入定时器寄存器。 若 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装入该寄存器。

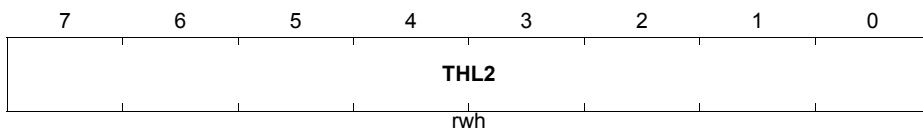
## 定时器

寄存器 T2 保存定时器 T2 的当前 16 位计数值。

### T2L

定时器 T2 寄存器，低位字节

复位值：00<sub>H</sub>

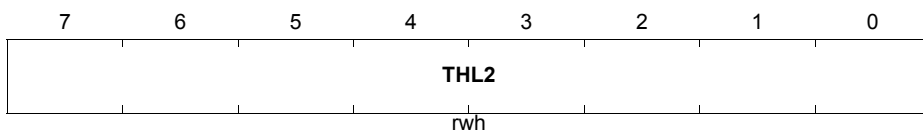


符号	位	类型	描述
<b>THL2</b>	[7:0]	rwh	定时器 T2 计数值 [7:0] 该寄存器存放定时器的当前计数值。

### T2H

定时器 T2 寄存器，高位字节

复位值：00<sub>H</sub>



符号	位	类型	描述
<b>THL2</b>	[7:0]	rwh	定时器 T2 计数值 [15:8] 该寄存器存放定时器的当前计数值。

## 14 T2CCU

T2CCU 模块的框图见图 14-1。该模块由标准定时器 T2 和一个捕获 / 比较单元 (CCU) 组成。捕获 / 比较定时器 (CCT) 是 CCU 的组成部分。可通过 CCU 分别为每路 16 位捕获 / 比较通道选择以定时器 T2 或 CCU 的捕获 / 比较定时器 (CCT) 作为时间基准。定时器 T2 和 CCT 的精度都为 16 位。T2CCU 的时钟频率  $f_{T2CCU}$  可设置为 PCLK 频率或者 PCLK 频率的 2 倍。

T2CCU 可用于各种数字信号产生和事件捕获：如脉冲产生、脉宽调制、脉宽测量等等。目标应用包括各种汽车控制以及工业应用（如频率产生、数模转换、进程控制等等）。

### 14.1 特性

T2CCU 具有如下特性：

- 可单独为每路通道选择定时器 T2 或捕获 / 比较定时器作为时间基准
- 捕获 / 比较定时器与定时器 T2 级联，可实现非常灵活的计数频率
- 通过触发溢出事件，捕获 / 比较定时器可立即被“复位”
- 16 位精度；最大频率 = 2 x 外设时钟频率
- 总共 6 路比较通道
- 总共 4 路捕获通道，与比较通道复用
- 每个比较寄存器都配有一个映射寄存器
  - 软件控制或定时器溢出触发映射传送
- 比较模式 0：发生比较匹配时，比较输出信号从被动态电平变为主动态电平，定时器溢出使得该信号返回到被动态电平。
  - 可通过寄存器位定义通道组 A 和 B 的主动态电平
  - 比较模式 0 支持 0% 到 100% 占空比
- 比较模式 1：下一次比较匹配的比较输出信号电平完全由软件控制
- 与通道 0 并行的比较模式
- 捕获模式 0：捕获 T2CC0 到 T2CC3 四个引脚上的任何外部事件（上升沿 / 下降沿 / 任意沿）
- 捕获模式 1：向相应的通道捕获寄存器低位字节写的操作触发捕获操作
- 4 个捕获通道可分别设置为捕获模式 0 或 1
- 由 CCT 定时器溢出事件和通道 5 比较匹配事件触发 ADC 转换请求。该特性的详细描述见章节 17.4.9。



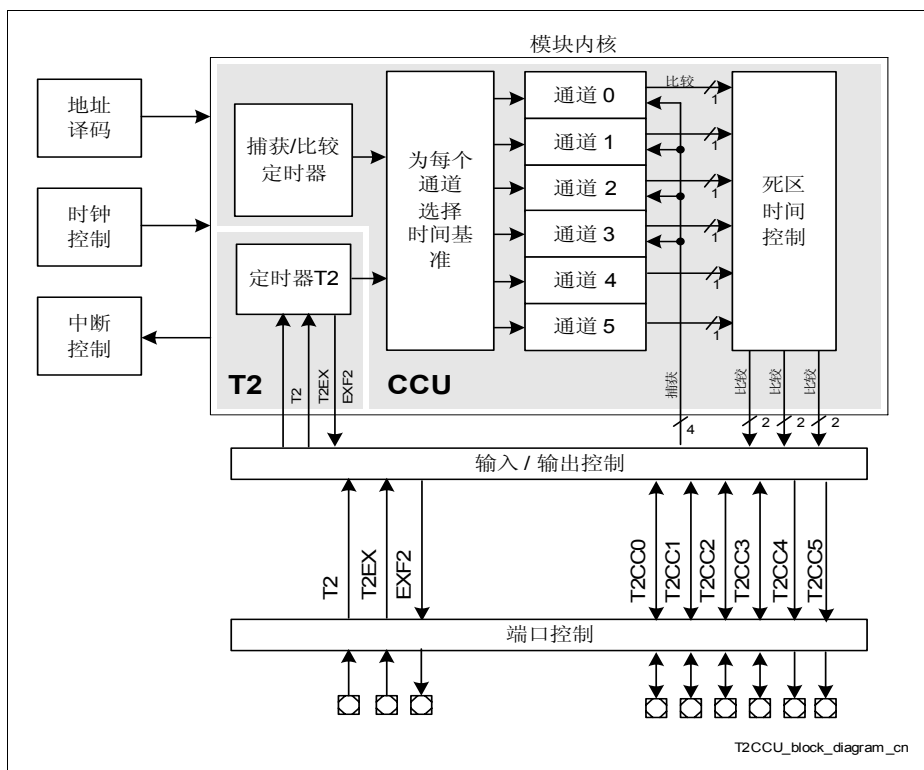


图 14-1 T2CCU 框图

## 14.2 对定时器 T2 的要求

缺省情况下，所以 T2CCU 通道选择 16 位定时器 T2 作为时间基准。通过设置位 TR2 或从外部通过引脚 T2EX 启动定时器 T2 的定时器功能（C/T2）。在启动之前，定时器 T2 溢出后的重载值必须初始化到寄存器 RC2H/L 中。定时器 T2 的递增 / 递减计数器功能必须被禁止（DCEN = 0），因此定时器 T2 仅用作递增计数器。可以根据应用需要设置 T2PRE 来选择定时器工作频率。

定时器 T2 支持的工作频率有： $f_{T2CCU}$ 、 $f_{T2CCU}/2$ 、 $f_{T2CCU}/4$ 、 $f_{T2CCU}/8$ 、 $f_{T2CCU}/16$ 、 $f_{T2CCU}/32$ 、 $f_{T2CCU}/64$  或  $f_{T2CCU}/128$ 。因此，如果 T2CCU 的输入时钟为 PCLK 频率，在最快情况下，16 位定时器每个机器周期递增两次；在最慢情况下，每 64 个机器周期递增一次。

定时器溢出时，定时器溢出标志 TF2 被置位，并产生一个中断（如使能）。

随时都可通过清除位 TR2 使定时器 T2 停止工作。

注： [章节 13.2](#) 详细描述定时器 T2 的操作。

## 14.3 T2CCU 的捕获 / 比较定时器

捕获 / 比较定时器（CCT）是 T2CCU 模块中的捕获 / 比较单元（CCU）的组成部分，专门用作 T2CCU 捕获 / 比较操作的时间基准。

基本上 CCT 是一个自由运行的计数器，可通过设置 T2CCU\_CCTCON.CCTST 来启动。当该位置位时，CCT 从 CCTH/L 定时器寄存器中的值开始计数，增至 FFFFH 之后溢出，随后定时器将被重载（重载值初始化到寄存器 CCTRELH/L），并从该值开始递增计数。CCT 的工作频率可为： $f_{T2CCU}$ 、 $f_{T2CCU}/2$ 、 $f_{T2CCU}/4$ 、 $f_{T2CCU}/8$ 、 $f_{T2CCU}/16$ 、 $f_{T2CCU}/32$ 、 $f_{T2CCU}/64$ 、 $f_{T2CCU}/128$ 、 $f_{T2CCU}/256$ 、 $f_{T2CCU}/512$ 、 $f_{T2CCU}/1024$  或  $f_{T2CCU}/2048$ 。因此如果 T2CCU 的输入时钟为 PCLK 频率，在最快的情况下，CCT 每个机器周期递增两次；在最慢的情况下，每 1024 个机器周期递增一次。

定时器溢出时，定时器溢出标志 CCTOVF 被置位，如果由 CCTOVEN 使能，则产生一个中断。

无论何时，即使 CCT 正在运行，CCT 定时器寄存器 CCTH/L 都可被写入以（重新）初始化定时器值。当 CCT 不运行时，写入 CCTL/H 的操作将更新定时器的相应字节。然而，当定时器运行时，CCTH 必须首先被写入。一旦 CCTL 被写入，则 CCTH 和 CCTL 中的值将更新定时器，定时器从该值开始计数。

在任何时刻都可通过清除位 CCTST 停止 CCT。定时器寄存器将保存最新的定时器计数值。

### 触发定时器溢出

CCT 定时器的一个特殊特性是可由触发的定时器溢出事件立即“复位”定时器。通过“复位”，CCT 定时器被重载，并从重载寄存器中的值开始计数，基准定时器溢出之后，和定时器通道的工作模式相关的所有操作都将被执行。通过置位 T2CCU\_CCTBSEL.CCTTOV，可以异步触发 CCT 定时器溢出事件以完成上述操作。比较模式 0 通道操作的例子见 [图 14-2](#)。比较模式 0 的详细操作请参考 [章节 14.5.1.1](#)。

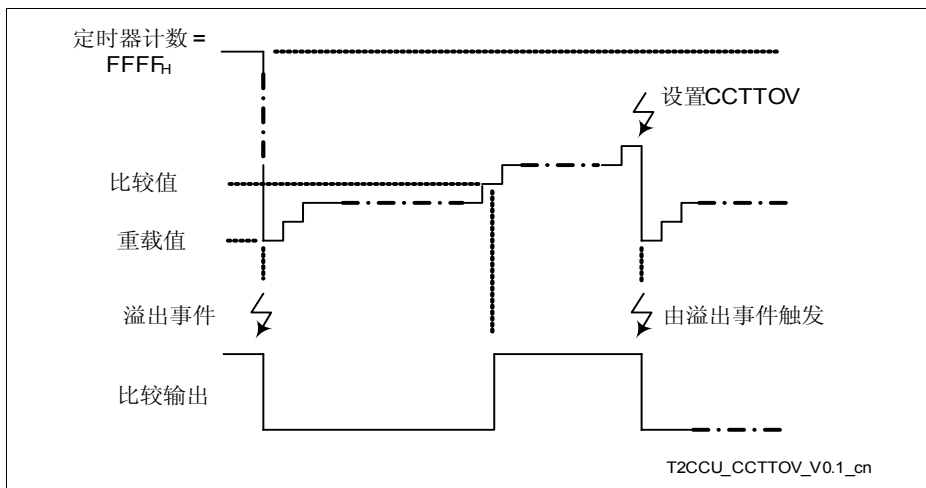


图 14-2 定时器溢出事件触发的、工作在比较模式 0 的通道操作例子

### 14.3.1 同步定时器启动

如上面的章节所述，可以单独启动 / 停止定时器：由 TR2 位控制定时器 T2；由 CCTST 位控制捕获/比较定时器。如果用户想要同步启动定时器，则启动过程需要满足下述条件：

1. 确保 CCT 定时器还未运行
2. 清除位 TR2（停止定时器 T2）并确保定时器被正确初始化
3. 置位 TIMSYN 使能定时器的同步启动
4. 置位 CCTST

第 3 步第 4 步可合并到一个设置 SFR T2CCU\_CCTCON 的写操作中。

同步定时器启动是针对 CCT 定时器还未运行的情况而定义的。当 T2CCU\_CCTCON 位 TIMSYN = 1 且 CCTST = 0，置位 CCTST 的操作还置位定时器 T2 的 T2CON.TR2 位。因此，两个定时器同时启动。若在定时器 T2 已运行的情况下置位 CCTST，定时器 T2 不受影响并继续正常的计数操作。

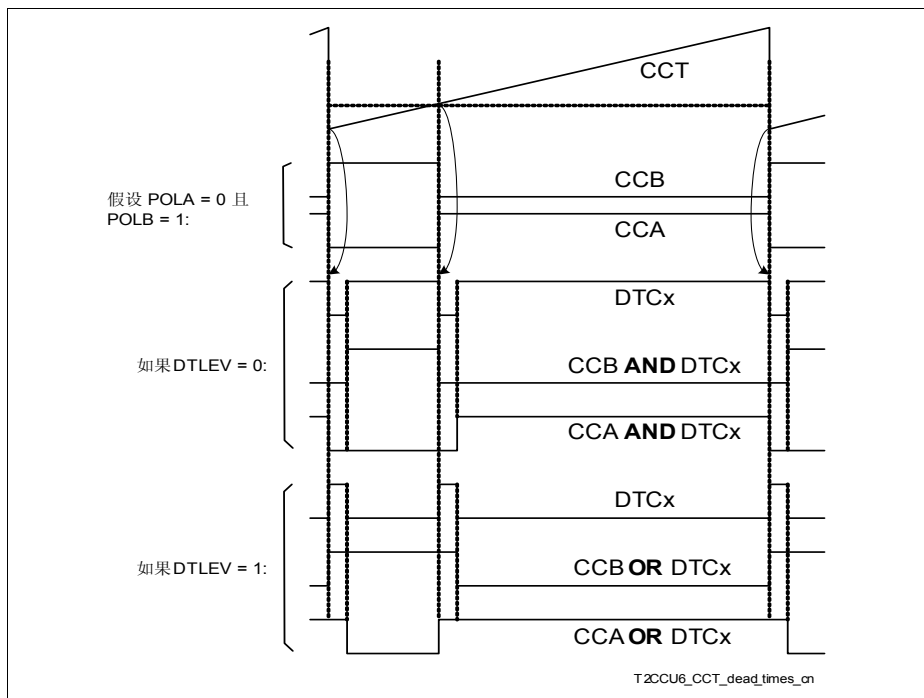
如需要停止定时器，可通过分别清除各定时器控制位来实现。

### 14.3.2 实现灵活计数频率的定时器级联

对于那些要求基准定时器以灵活计数频率运行的应用来说，允许定时器 T2 和 CCT 定时器进行级联。如果位 T2CCU\_CCTBSEL.CASC 被置位，CCT 定时器对定时器 T2 的溢出速率进行计数。请参考图 14-3，其中定时器 T2 被配置为定时器。

定时操作过程中，T2 的重载值可以被重置，同时 T2 仍可作为其它通道捕获或比较操作的时间基准。





**图 14-4 带死区时间产生的 PWM 信号（图中仅画出一对通道）**

在大多数情况下，相互连接的功率开关的切换特性不对称，即打开和关闭开关所需时间不同。功率器件的开启时间小于关闭时间时通常会引发问题，这将导致变频器桥臂短路，从而可能使整个系统瘫痪。为了用硬件解决这个问题，T2CCU 中提供了一个可编程死区时间计数器，将切换信号的被动状态到主动状态的跳变沿延迟（主动状态到被动状态的跳变沿不被延迟）。见图 14-4。

三组通道对的死区时间产生逻辑（见图 14-5）相似。每次 CCT 定时器溢出或发生比较匹配事件时触发对应的死区时间计数器（8 位递减计数器，计数时钟为 CCT 计数频率）并使 DTCx 翻转为主动态电平。触发脉冲（DTCx\_r1）会重载死区时间计数器，重载值被编程到位域 T2CCU\_CCTDTCL.DTM 中。重载操作只有在死区时间特性被使能（T2CCU\_CCTDTCH.DTEx 被置位）且计数器为零的情况下才能进行。

递减计数（还未达到零）时，控制信号 DTCx 保持在主动态电平。当计数器值达到零时，死区时间计数器停止计数且信号 DTCx 再次跳变至被动态电平。因此 DTCx 和 CCT 产生的调制信号 CCA 和 CCB 经过一些逻辑操作，可以使比较输出信号有一个被动到主动沿的延迟，见图 14-4。DTCx 的主动态电平及信号的耦合操作可以通过设置位域 T2CCU\_CCTDTCH.DTLEV（定义开关的被动态电平）进行配置。



## 14.4 T2CCU 捕获 / 比较通道

T2CCU 支持下述工作模式：

- 比较：多达 6 路 PWM 输出，最多 65536 个步长
- 捕获：多达 4 个高速捕获输入

每路通道的捕获 / 比较精度取决于定时器基准工作频率。

**图 14-6** 给出一路通道的定时器 T2 和捕获 / 比较单元 (CCU) 的功能框图。I/O 引脚通常分配用作通用 IO 端口的复用功能。如**表 14-1** 所示，T2CC0、1、2 和 3 的 I/O 始终分别与外部输入中断 EXINT3、4、5、6 共用同一个引脚。对应的外部中断请求标志置位，用于捕获 / 比较功能。定时器溢出时，对应的定时器溢出标志也置位。使用寄存器 MODPSEL1 和 MODPSEL4 选择 T2CC0、1、2 和 3 的输入，见**页 5-23** 和**页 5-24**。

**表 14-1 T2CCU 的捕获 / 比较 I/O**

引脚	功能
T2CC0 / EXINT3	通道 0 的比较输出 / 捕获输入
T2CC1 / EXINT4	通道 1 的比较输出 / 捕获输入
T2CC2 / EXINT5	通道 2 的比较输出 / 捕获输入
T2CC3 / EXINT6	通道 3 的比较输出 / 捕获输入
T2CC4	通道 4 的比较输出
T2CC5	通道 5 的比较输出

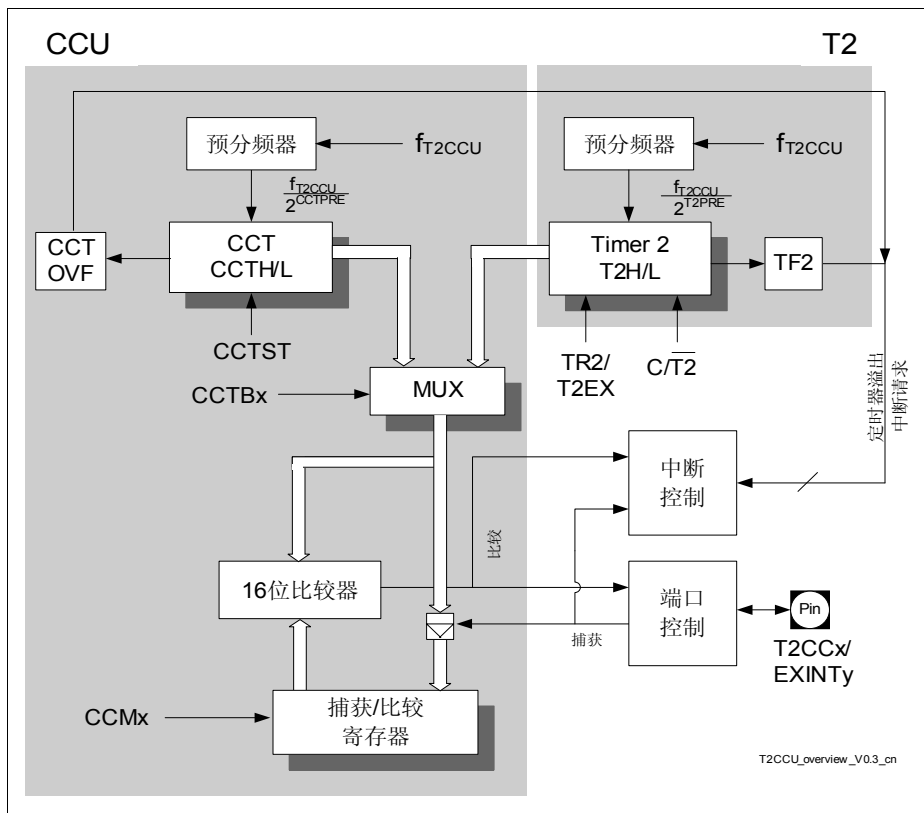


图 14-6 通道 x 的定时器 T2 捕获 / 比较功能



## 14.5 捕获 / 比较操作

T2CCU 总共有可分别使能的 6 路比较通道和 4 路捕获通道。捕获通道可单独选择捕获模式 0 或 1。每个通道的捕获 / 比较功能可在寄存器 T2CCU\_CCEN 中选择。

进行比较操作时，可通过寄存器 T2CCU\_COCON 配置所有比较通道的比较模式。比较通道还进一步分成两组 - A 组和 B 组，A 组由通道 CC0、CC2、CC4 组成，B 组由通道 CC1、CC3 和 CC5 组成，可分别为各通道配置比较模式。因而两组通道比较匹配之后直到定时器溢出（主动态电平）的输出电平可分别配置。

如果选择捕获模式，用于捕获的外部事件可通过外部中断 SFR 配置。可单独为每个通道选择上升沿、下降沿或任意沿为有效的触发沿。

*注：若要实现预先设定的操作，需要用户在使能任何 T2CCU 捕获或比较模式之前，正确初始化所有与 T2CCU 相关的 SFR。*

### 14.5.1 比较功能

使能比较功能之后才能有效驱动 T2CCU 比较输出。当通过设置通道 CCMx 位使能比较功能时，根据通道 SFR 位的设置，对应的比较输出被锁存（初始值）。

比较模式下，保存在比较寄存器中的 16 位值与定时器内容进行比较。当定时器计数值与比较寄存器值匹配时，相应通道的输出信号翻转，并请求中断（如被使能）。

比较寄存器的内容可被当作是“时间戳”，定时器计数到此值时，相应的输出以预先定义的方式（正跳变或者负跳变）改变。该“时间戳”的变化可改变引脚上方波输出信号的波形。随着周期信号占空比的变化，可使用该特性实现脉宽调制以及持续控制各类方波的产生。T2CCU 的两种比较模式加一个并行比较模式可涵盖各种可能的应用。

通过位 T2CCU\_COCON.COMOD 选择有效的比较模式，并适用于所有使能的比较通道。在所有比较模式下，在内部比较信号被激活的那个 T2CCU 时钟周期内，新的输出值即可到达端口引脚。

只要比较器的输入相等，比较匹配信号保持高电平（至少一个 T2CCU 时钟周期）。所有比较匹配操作的有效沿都是比较匹配信号的上升沿。

### 比较寄存器的映射寄存器

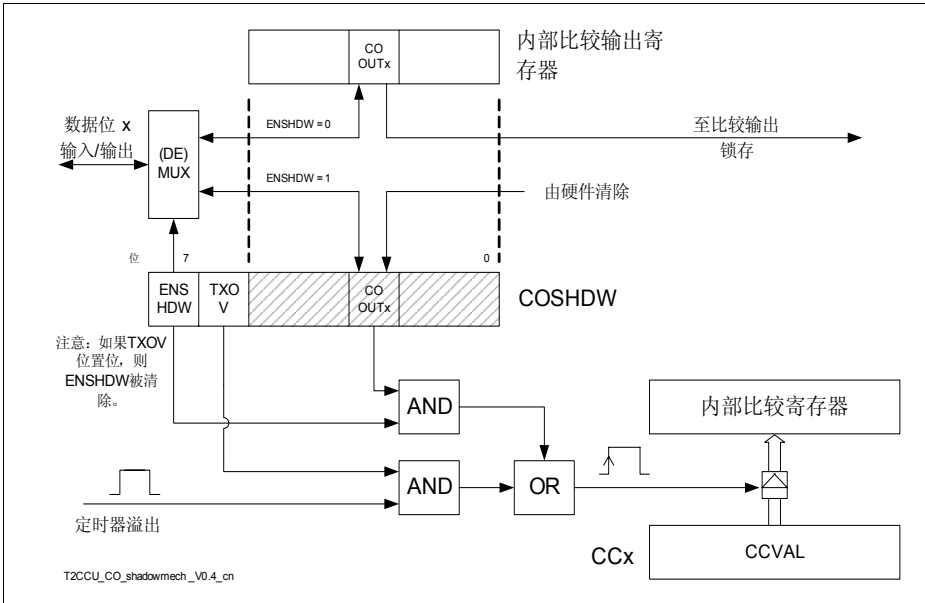
用户代码不能直接访问 T2CCU 的比较寄存器。通过相应 SFR T2CCU\_CCx 的映射传送实现通道比较值的更新。用户可从下面两个触发事件中选择一个来启动向内部比较寄存器进行映射传送。

1. 通过置位 SFR 位 T2CCU\_COSHDW.TXOV = 1，在对应的基准定时器溢出时，启动比较通道的映射传送。在此情况下，比较寄存器的更新通过硬件控制，因此与任何中断服务延迟无关。
2. 通过置位 SFR 位 T2CCU\_COSHDW.ENSHDW = 1，当用户置位对应 SFR 位 COOUTx 时，则启动每个通道映射传送。被置位之后，在下一个 T2CCU 时钟周期，硬件将自动清除 COOUTx。

在上述两种情况下，事件脉冲信号的上升沿是比较寄存器映射传送的有效边沿。更新 16 位比较寄存器需要两条指令，这样会导致毛刺的产生，因此映射传送机制的优势在于可实现无毛刺的脉宽调制。

需要注意：不允许同时置位 TXOV 和 ENSHDW。如果 TXOV 被置位，使能定时器溢出触发映射传送，内部硬件将会清除 ENSHDW 禁止软件启动映射传送。

图 14-7 给出映射传送机制。



## 映射的比较输出位

为了使能 SFR T2CCU\_COSHWDW 位 COOUTx 的双重功能，位 ENSHDW 控制对这些位的访问。当 ENSHDW = 1 时，SFR 位 COOUTx 能够直接被访问，用于控制比较值的映射传送。当 ENSHDW = 0 时，写或读访问 COOUTx 位实际上访问的是内部比较输出寄存器，而不是 COSHWDW 位。该内部比较输出寄存器的对应位控制比较模式 1 和并行比较模式下比较通道的输出极性。

图 14-7 给出位 COOUTx 的双重功能。

### 14.5.1.1 比较模式 0

选择比较模式 0 时，置位 CCMx 使能比较功能，发生下述情况时，比较输出被锁存：

- 如果基准定时器的值等于或大于比较寄存器的值，比较输出被锁存至主动态电平，或者
- 如果基准定时器的值小于比较寄存器的值，比较输出被锁存至被动态电平。

通过 SFR 位 T2CCU\_COCON.POLA 或 POLB 定义比较 A 组或 B 组的每路通道的主动态电平。

工作在比较模式 0 下，当定时器和比较寄存器内容匹配时，输出信号从被动态电平变为主动态电平。定时器溢出之后，比较输出信号返回到被动态电平。

比较模式 0 特别适合产生脉宽调制输出信号，随后这些调制信号被用于通过滤波网络的数模转换或受控器件（如 DC 或 AC 感应电机）。模式 0 还可用于产生预设周期和占空比的输出时钟。该应用几乎不占用 CPU 时间，一旦完成设置，无需 CPU 干预，自动生成输出时钟。

比较模式 0 的功能框图如图 14-8 所示。在该功能下，比较输出信号由 SFR 位 T2CCU\_COCON.POLA/B 的设置和事件（定时器溢出和比较匹配）直接进行控制。

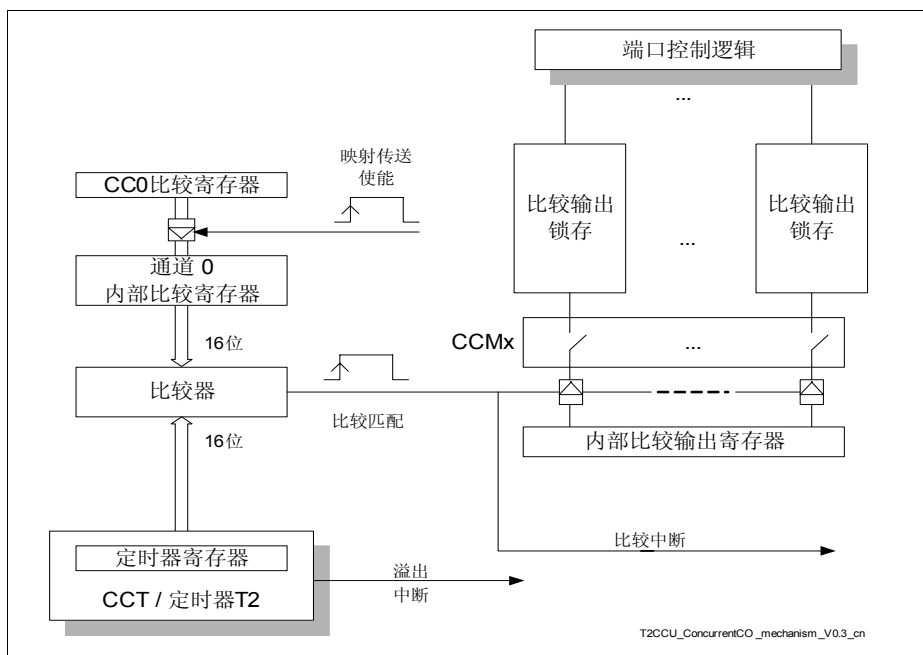


图 14-8 比较模式 0 的功能框图

图 14-9 给出当 POLA/B 设置为 ‘0’ 的通道输出。定时器溢出或比较匹配可触发中断，并在中断服务程序中完成更新下一次的比较值等操作。

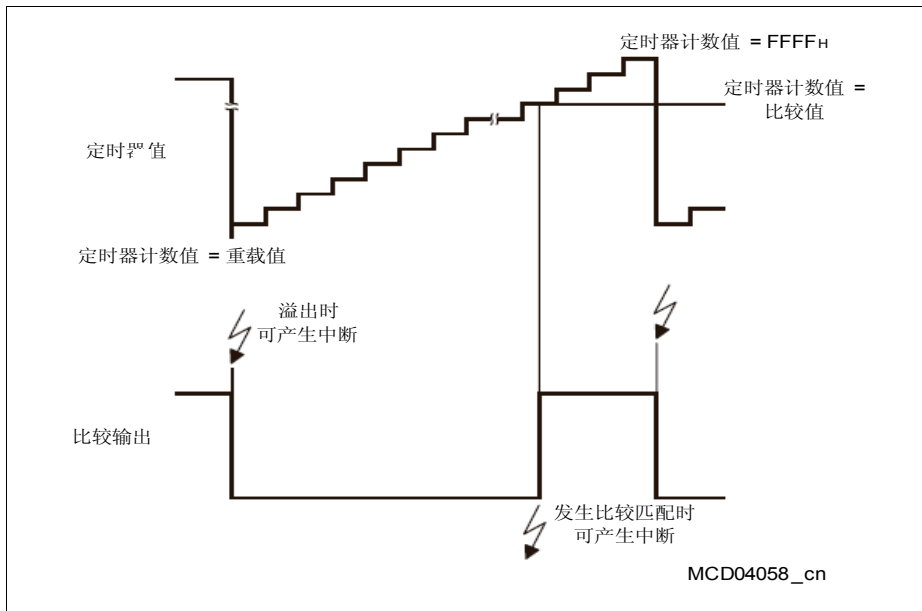


图 14-9 比较模式 0 的通道输出 (POLA/B = 0)

比较模式 0 中，产生占空比从 0% 到 100% 之间的 PWM 信号

PWM 信号的主动态电平是指从比较匹配直到下一次定时器溢出之间的比较输出信号电平。

比较通道设置为 T2CCU\_COCON.POLA/B = 0 (主动态电平 = 高) 时，为了支持 100% 占空比的信号，16 位通道 x 比较值必须设置为等于定时器溢出重载值。此时，定时器溢出时，内部硬件将不会使输出信号变为被动态电平。输出保持主动态电平，直到比较值不再等于定时器重载值时的下一次定时器溢出。

比较通道设置为 T2CCU\_COCON.POLA/B = 0 时，为了支持 0% 占空比的信号，16 位通道 x 比较值必须设置为小于定时器溢出重载值。此时，定时器重载值一定不能等于 0。比较匹配事件将不会出现，比较输出保持在被动态电平 (低电平)。

上述机制同样适用于 T2CCU\_COCON.POLA/B = 1 的比较通道。

### 14.5.1.2 比较模式 1

选择比较模式 1 时，置位 CCMx 使能比较功能，内部比较输出寄存器中位 COOUTx 的值将被锁存至比较输出端口。

工作在比较模式 1 下，软件自适应的决定输出信号的跳变。该模式通常用于产生非固定周期（不同于标准 PWM 产生的固定周期信号）、但对高精度和无抖动有精确控制的输出信号。在比较模式 1 中，下一次比较匹配的比较输出信号电平完全由软件控制。

如果使能比较模式 1，软件设置内部比较输出寄存器的 COOUTx（一个或多个）（在此情况下，位 T2CCU\_COSHDW. ENSHDW 必须被清除）时，在发生下次比较匹配事件之前，输出引脚不送出新值。因此，当通道定时器计数值与通道内部比较寄存器值匹配时，用户可为每路输出通道单独选择是否翻转输出信号（1-0 或 0-1）或者是否保持同样的电平。

图 14-10 给出比较模式 1 的功能框图。在该功能下，只有当发生下一次比较匹配事件时，内部比较输出寄存器的位 COOUTx 的值才被送至比较输出锁存。

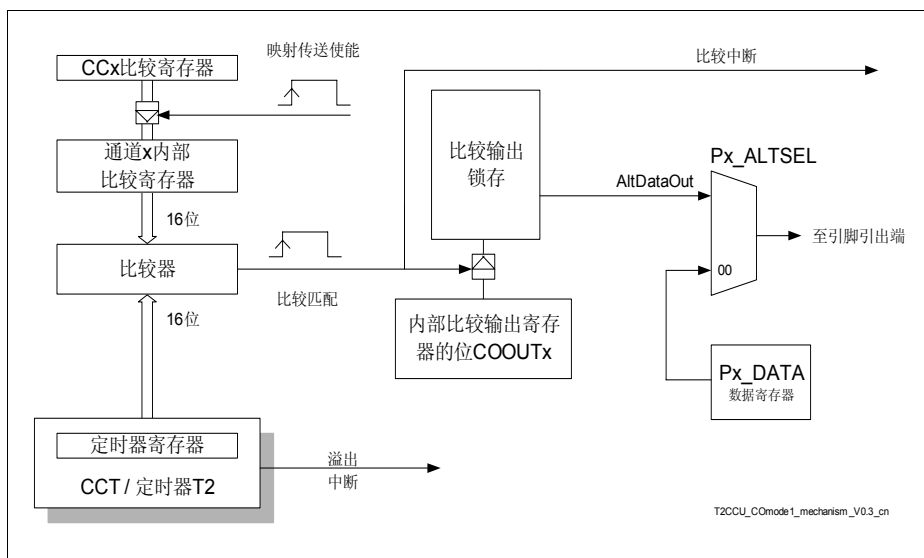


图 14-10 比较模式 1 的功能框图

### 14.5.1.3 并行比较模式

选择并行比较模式时，置位 CCMx 使能比较功能，内部比较输出寄存器中位 COOUTx 的值被锁存到比较输出端口。

并行比较是一种高效的比较模式，特别适用于需要产生多个同步输出信号的应用。这些应用包括：多相步进电机控制、汽车引擎点火线圈控制等。这些应用通常需要在精确预定的时刻将预设的位序列送至输出端口。

图 14-11 给出并行比较模式的功能框图。为了能够正确工作，比较通道 0 必须被使能和初始化。

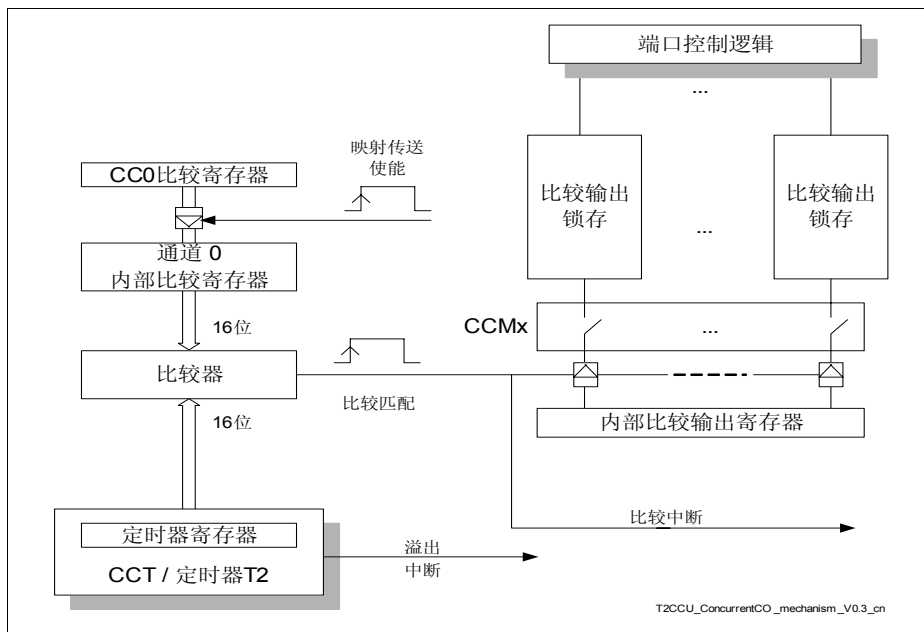


图 14-11 并行比较模式的功能框图

图 14-12 给出两个例子，说明如何使用一个序列表在端口上产生四个方波。在定时器到达设定的计数值之前，序列被写入到内部比较输出寄存器中（`T2CCU_COSHDW.ENSHDW = 0`）。（未来的）定时器计数值（定时器到达该值时，序列被输出至端口）必须被加载到比较映射寄存器 `T2CCU_CC0H` 和 `T2CCU_CC0L`，并传送到内部比较寄存器。为了管理波形的产生，其中一个例子使用一个列出比较值的时间进度表；对于较短的定时器周期，另一个例子使用固定的比较值（如 `FFF0H`）。通过这些方式，用户可控制每个通道的输出在预定的时间点上是否翻转。

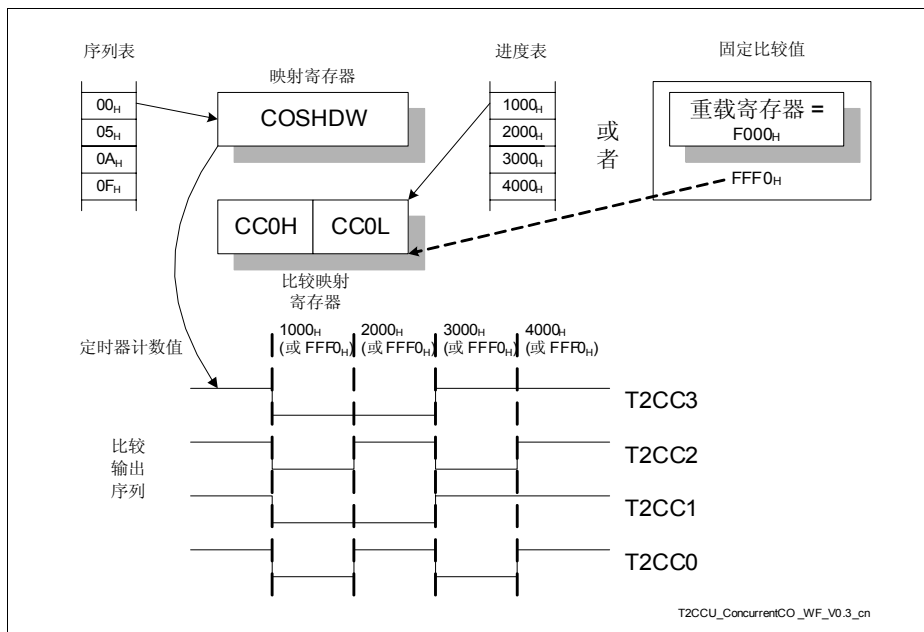


图 14-12 使用并行比较模式产生波形两例（比较值时间进度表或固定比较值）

#### 14.5.1.4 中断和比较功能组合使用

##### 保留的外部中断

比较通道 T2CCU\_CC0-3 中的每一路与一个外部中断共享一个端口引脚，作为引脚的复用功能。当比较模式被使能，且相应的端口引脚被设置为输出模式用作比较输出时，端口引脚的输入驱动器被禁用。因此，相应的外部中断输入不能用作任何用途。

不过，外部中断的中断功能仍可被使能并用于比较匹配中断。此时，相应的外部中断线直接和内部比较匹配信号相连。

##### 比较匹配中断

IENT1.EXM 用于使能所有通道的比较匹配中断（包括通道 4 和通道 5）和捕获中断。当定时器计数值等于通道寄存器值时，发生通道比较匹配。除了控制相应通道的比较输出，相应的比较匹配标志 IRCON.EXINTx 或 T2CCU\_COMOD.CMxF 被置位用于指示中断请求。为此，必须在相应的寄存器 EXICON 中将比较匹配的中断触发边沿设置为上升沿。比较匹配中断服务程序可被用于设置下次通道比较事件的参数，如改变比较寄存器的内容或确定下次“比较”匹配的比较端口输出电平。

使用比较匹配中断的主要优点是避免了在达到匹配之前，无意改写比较 / 比较输出寄存器。在不知道实际定时器计数值时，CPU 写访问比较寄存器就会发生上述无意改写的情况。

*注： 另一种选择是，可使用定时器溢出中断代替比较匹配中断。*



### 14.5.2 捕获功能

在捕获模式下，可通过捕获 / 比较寄存器 **CC0-3** 测量外部信号脉冲宽度。每个 16 位寄存器都可用于锁存相应通道的基准定时器当前的 16 位计数值。该功能提供两种捕获模式。

在模式 0 下，引脚 **T2CC0** 至 **T2CC3** 上的任何外部事件（上升沿 / 下降沿 / 任意沿）都将通道定时器的内容锁存到相应的通道捕获寄存器中。由相应的外部中断设置（每个捕获输入与一个外部中断共享一个引脚）决定捕获边沿。在每个 **CCLK** 时钟对外部输入进行采样。当输入采样在一个 **CCLK** 为低（高）电平，而在下一个 **CCLK** 为高（低）电平时，则识别到一次跳变。相应的外部中断请求标志（**EXINT3** 至 **EXINT6**）被置位。如果中断被使能，CPU 将执行相应的中断服务程序。在下一个（识别到跳变之后的）**T2CCU** 时钟周期，定时器的计时值被锁存到相应的捕获寄存器中。

在模式 1 下，写访问通道捕获寄存器的低位字节的操作将触发捕获操作。软件能够在“运行中”读取通道定时器的内容。对于该功能来讲，写入到捕获寄存器中的内容无关紧要。写指令之后，定时器内容将被锁存到通道捕获寄存器中。在该工作模式下，将不产生中断请求。

图 14-13 给出捕获 / 比较通道的捕获模式。

可通过 **SFR T2CCU\_CCEN**（捕获 / 比较使能寄存器）单独设置每个捕获通道的工作模式。这意味着，与比较模式不同，可同时选择一个捕获通道工作在模式 0，而另一个捕获通道工作在模式 1。

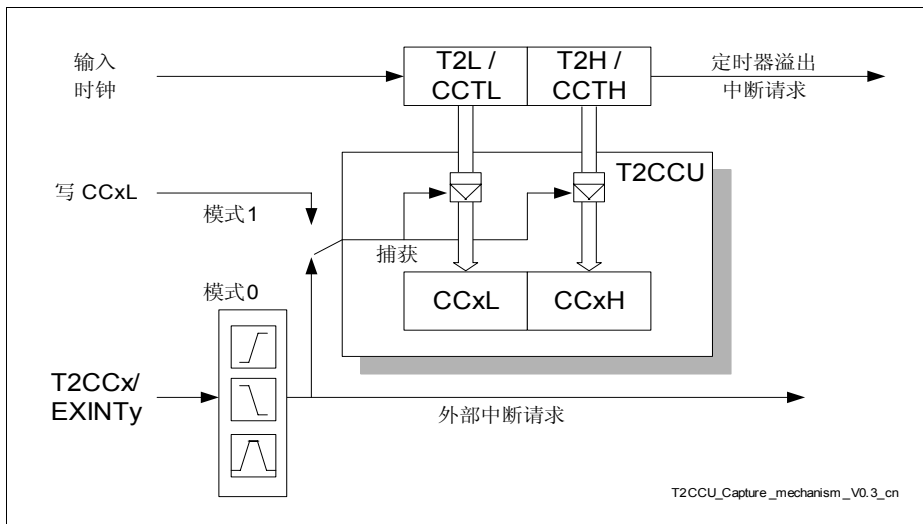


图 14-13 捕获模式功能框图

## 14.6 T2CCU 寄存器

本节描述所有的T2CCU特殊功能寄存器。与定时器T2和中断相关的寄存器也一并列出，参见表 14-2。

注： 引用所有T2CCU寄存器时，需要添加模块名前缀“T2CCU\_”。

表 14-2 Registers Overview 寄存器概述

名称	描述
<b>T2CCU 特殊功能寄存器</b>	
CCTBSEL	捕获 / 比较时间基准选择寄存器
CCTRELL	捕获 / 比较定时器重载寄存器低位
CCTRELH	捕获 / 比较定时器重载寄存器高位
CCTL	捕获 / 比较定时器寄存器低位
CCTH	捕获 / 比较定时器寄存器高位
CCTCON	捕获 / 比较定时器控制寄存器
CCTDTCL	捕获 / 比较定时器死区时间控制寄存器低位
CCTDTCH	捕获 / 比较定时器死区时间控制寄存器高位
CCEN	捕获 / 比较使能寄存器
COSHDW	比较映射寄存器
COCON	比较控制寄存器
CC0L	捕获 / 比较寄存器 0 低位
CC0H	捕获 / 比较寄存器 0 高位
CC1L	捕获 / 比较寄存器 1 低位
CC1H	捕获 / 比较寄存器 1 高位
CC2L	捕获 / 比较寄存器 2 低位
CC2H	捕获 / 比较寄存器 2 高位
CC3L	捕获 / 比较寄存器 3 低位
CC3H	捕获 / 比较寄存器 3 高位
CC4L	比较寄存器 4 低位
CC4H	比较寄存器 4 高位
CC5L	比较寄存器 5 低位
CC5H	比较寄存器 5 高位
<b>与定时器 T2 相关的特殊功能寄存器</b>	
T2_T2CON	定时器 T2 控制寄存器

**表 14-2      Registers Overview 寄存器概述**

名称	描述
T2_T2CON1	定时器 T2 控制寄存器 1
T2_T2MOD	定时器 T2 模式寄存器
T2_RC2L	定时器 T2 重载寄存器，低字节
T2_RC2H	定时器 T2 重载寄存器，高字节
T2_T2L	定时器 T2，低字节
T2_T2H	定时器 T2，高字节
<b>与中断有关的特殊功能寄存器</b>	
IEN0	中断使能寄存器 0
IEN1	中断使能寄存器 1
EXICON0	外部中断控制寄存器 0
EXICON1	外部中断控制寄存器 1
IRCON0	中断请求寄存器 0

### 14.6.1      寄存器映射

T2CCU 支持局部地址扩展机制。定时器 T2 的 SFR 位于页 0，T2CCU 的 SFR 位于页 1 至页 3。局部地址扩展的详情请参考第 4 章。

SFR 的地址（非映射）归纳见表 14-3。

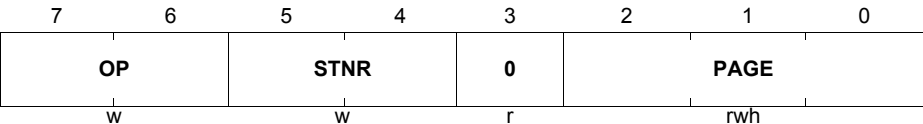
**表 14-3      页 0-4 的 SFR 地址**

地址	页 0	页 1	页 2	页 3	页 4
C0 <sub>H</sub>	T2CON	CCEN	COSHDW	COCON	
C1 <sub>H</sub>	T2MOD	CCTBSEL	CC0L	CC3L	
C2 <sub>H</sub>	RC2L	CCTRELL	CC0H	CC3H	CCTDTCL
C3 <sub>H</sub>	RC2H	CCTRELH	CC1L	CC4L	CCTDTCH
C4 <sub>H</sub>	T2L	CCTL	CC1H	CC4H	
C5 <sub>H</sub>	T2H	CCTH	CC2L	CC5L	
C6 <sub>H</sub>	T2CON1	CCTCON	CC2H	CC5H	

注： T2CCU 中定时器 T2 的 SFR（位于页 0 的）必须通过分页机制进行寻址。

位于 C7<sub>H</sub> 处的 SFR T2\_PAGE 包含页面值和页控制信息。

**T2\_PAGE**  
**T2CCU 页面寄存器** 复位值: 00<sub>H</sub>



符号	位	类型	描述
<b>PAGE</b>	[2:0]	rwh	<b>页信息</b> 写入时，该值表示新页地址。 读出时，该值表示当前有效页地址 = addr [y:x+1]
<b>STNR</b>	[5:4]	w	<b>保存编号</b> 该值指示在哪个保存位域上执行由 OP 定义的操作。 如果 OP = 10 <sub>B</sub> , PAGE 的内容在被新值覆盖之前保存在 STx 中。 如果 OP = 11 <sub>B</sub> , PAGE 的内容被 STx 的内容覆盖。写入 PAGE 的值被忽略。  00 选择 ST0 01 选择 ST1 10 选择 ST2 11 选择 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 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 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。

### 14.6.2 T2CCU 寄存器描述

下面描述 T2CCU 内核的特殊功能寄存器。

## T2CCU\_CCTBSEL

T2CCU 捕获 / 比较时间基准选择寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
CASC	CCTTOV	CCTB5	CCTB4	CCTB3	CCTB2	CCTB1	CCTB0
rw	rwh	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>CCTBx</b> (x = 0-5)	[5:0]	rw	通道 x 时间基准选择 0 选择定时器 T2 1 选择捕获 / 比较定时器 (CCT)
<b>CCTTOV</b>	6	rwh	触发 CCT 定时器溢出事件 0 无操作 1 立即触发 CCT 溢出事件 置位之后, 该位保持一个 T2CCU 时钟周期, 然后被硬件清除。 读取该位始终返回 0。
<b>CASC</b>	7	rw	定时器级联 0 定时器 T2 与 CCT 定时器独立工作 1 定时器 T2 与 CCT 定时器级联

## T2CCU\_CCTRELL

T2CCU 捕获 / 比较定时器重载寄存器低位

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
CCTREL[7:0]							
rw							

符号	位	类型	描述
<b>CCTREL[7:0]</b>	7:0	rw	捕获 / 比较定时器低位字节重载值 定时器溢出时, 该寄存器的内容被加载到 CCT 定时器低位字节寄存器。

## T2CCU\_CCTRELH

T2CCU 捕获 / 比较定时器重载寄存器高位

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
CCTREL[15:8]							
rw							

符号	位	类型	描述
CCTREL[15:8]	7:0	rw	捕获 / 比较定时器低位字节值 定时器溢出时, 该寄存器的内容被加载到 CCT 定时器高位字节寄存器。

## T2CCU\_CCTL

T2CCU 捕获 / 比较定时器寄存器低位

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
CCT[7:0]							
rwh							

符号	位	类型	描述
CCT[7:0]	7:0	rwh	捕获 / 比较定时器低位字节值 该寄存器给出捕获 / 比较定时器计数值的低位字节。 当 CCT 不运行时, CCTL 写操作将更新定时器低位字节。 当 CCT 运行时, 为了 (重新) 初始化, 必须先写入 CCTH。写入 CCTL 时, CCTH 和 CCTL 的值会立即更新定时器内容, 并从该值开始计数。

## T2CCU\_CCTH

T2CCU 捕获 / 比较定时器寄存器高位

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
CCT[15:8]							
rwh							

符号	位	类型	描述
<b>CCT[15:8]</b>	7:0	rwh	<b>捕获 / 比较定时器高位字节值</b> 该寄存器表示捕获 / 比较定时器计数值的高位字节。 当 CCT 不运行时，CCTL 写操作将更新定时器高位字节。 当 CCT 运行时，为了（重新）初始化，必须先写入 CCTH。写入 CCTL 时，CCTH 和 CCTL 的值会立即更新定时器内容，并从该值开始计数。

### T2CCU\_CCTCON

T2CCU 捕获 / 比较定时器控制寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>CCTPRE</b>				<b>CCTOVF</b>	<b>CCTOVEN</b>	<b>TIMSYN</b>	<b>CCTST</b>
rw				rwh	rw	rw	rw

符号	位	类型	描述
<b>CCTST</b>	0	rw	<b>捕获 / 比较定时器启动 / 停止控制</b> 0 停止捕获 / 比较定时器 1 启动捕获 / 比较定时器 如果 TIMSYN = 1 且 CCTST 为 0，同时启动定时器 T2（如果定时器 T2 还未运行）。 <i>注：在 CCTST 已经为 1 的情况下对其置位将不会影响正在运行的 CCT 定时器操作。</i>
<b>TIMSYN</b>	1	rw	<b>使能定时器同步启动</b> 0 禁止定时器同步启动 1 使能定时器同步启动
<b>CCTOVEN</b>	2	rw	<b>捕获 / 比较定时器溢出中断使能</b> 置位该位使能 CCT 溢出中断。
<b>CCTOVF<sup>1)</sup></b>	3	rwh	<b>捕获 / 比较定时器溢出标志</b> CCT 溢出时硬件置位该位。 必须由软件清除该位。

## T2CCU

符号	位	类型	描述
<b>CCTPRE</b>	[7:4]	rw	<p><b>捕获 / 比较定时器预分频因子选择位</b></p> <p>选择 CCT 的输入时钟，该时钟由外设时钟分频得到。</p> <p>0000 <math>f_{CCT} = f_{T2CCU}</math></p> <p>0001 <math>f_{CCT} = f_{T2CCU}/2</math></p> <p>0010 <math>f_{CCT} = f_{T2CCU}/4</math></p> <p>0011 <math>f_{CCT} = f_{T2CCU}/8</math></p> <p>0100 <math>f_{CCT} = f_{T2CCU}/16</math></p> <p>0101 <math>f_{CCT} = f_{T2CCU}/32</math></p> <p>0110 <math>f_{CCT} = f_{T2CCU}/64</math></p> <p>0111 <math>f_{CCT} = f_{T2CCU}/128</math></p> <p>1000 <math>f_{CCT} = f_{T2CCU}/256</math></p> <p>1001 <math>f_{CCT} = f_{T2CCU}/512</math></p> <p>1010 <math>f_{CCT} = f_{T2CCU}/1024</math></p> <p>1011 <math>f_{CCT} = f_{T2CCU}/2048</math></p> <p>其它值 保留</p> <p><i>注： 当 CCTST = 1 时，建议用户不要改变 CCTPRE 的值。</i></p>

<sup>1)</sup> 软件和硬件同时访问该标志位时，硬件操作优先级更高。

## T2CCU\_CCTDTCL

T2CCU 捕获 / 比较定时器死区时间控制寄存器低位

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
DTM							
rw							

符号	位	类型	描述
<b>DTM</b>	[7:0]	rw	<p><b>死区时间</b></p> <p>该 8 位值定义所选择的输出从被动态切换到主动态之间的可编程延迟。从主动态切换到被动态不延迟。</p>



## T2CCU\_CCTDTCH

T2CCU 捕获 / 比较死区时间控制寄存器高位

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
DTRES	DTR2	DTR1	DTR0	DTLEV	DTE2	DTE1	DTE0
rwh	rh	rh	rh	rw	rw	rw	rw

符号	位	类型	描述
<b>DTE<sub>x</sub></b> ( <b>x = 0-2</b> )	[2:0]	rw	<b>通道对 x 死区时间使能位</b> 0 禁止产生死区时间 1 使能通道对 x 产生死区时间。（根据比较状态）从被动态切换至主动态输出时产生延迟，延迟值由位域 DTM 设定。  <i>注：只有在使能相应通道工作在比较模式 0 且 CCT 作为基准定时器时，该位才应被置位。</i>
<b>DTLEV</b>	3	rw	<b>死区时间电平</b> 该位定义死区时间计数期间所有 DTC <sub>x</sub> 的主动态电平。 0 低电平（开关为高主动态） 1 高电平（开关为低主动态）
<b>DTR<sub>x</sub></b> ( <b>x = 4-6</b> )	[6:4]	rh	<b>死区时间逻辑功能运行指示位</b> 0 相应通道死区时间计数器的值为 0 1 相应通道死区时间计数器的值非 0
<b>DTRES</b>	7	rwh	<b>复位死区时间计数器</b> 0 无操作 1 三个死区时间计数器被复位至 0 一旦置位，该位保持一个 T2CCU 时钟周期，之后被硬件清除。 读取该位始终返回 0。

## T2CCU\_CCEN

T2CCU 捕获 / 比较使能寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
CCM3	CCM2	CCM1	CCM0				
rw	rw	rw	rw				

符号	位	类型	描述
<b>CCM0</b>	[1:0]	rw	<b>通道 0 捕获 / 比较使能</b> 00 禁止捕获 / 比较 01 引脚 T2CC0 上的有效沿触发捕获操作 10 使能比较功能 11 寄存器 CC0L 的写操作触发捕获
<b>CCM1</b>	[3:2]	rw	<b>通道 1 捕获 / 比较使能</b> 00 禁止捕获 / 比较 01 引脚 T2CC1 上的有效沿触发捕获操作 10 使能比较功能 11 寄存器 CC1L 的写操作触发捕获
<b>CCM2</b>	[5:4]	rw	<b>通道 2 捕获 / 比较使能</b> 00 禁止捕获 / 比较 01 引脚 T2CC2 上的有效沿触发捕获操作 10 使能比较功能 11 寄存器 CC2L 的写操作触发捕获
<b>CCM3</b>	[7:6]	rw	<b>通道 3 捕获 / 比较使能</b> 00 禁止捕获 / 比较 01 引脚 T2CC3 上的有效沿触发捕获操作 10 使能比较功能 11 寄存器 CC3L 的写操作触发捕获

注： 只有使能比较功能时， T2CCU 输出才能被有效驱动。

## T2CCU\_COSHDW

### T2CCU 比较映射寄存器

复位值： 00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>ENSHDW</b>	<b>TXOV</b>	<b>COOUT5</b>	<b>COOUT4</b>	<b>COOUT3</b>	<b>COOUT2</b>	<b>COOUT1</b>	<b>COOUT0</b>
rw	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
<b>COOUTx</b> (x = 0-5)	[5:0]	rwh	<p><b>比较通道 x 输出</b> <b>ENSHDW = 0</b></p> <p>任何对位 COOUTx 的访问实际上访问的是内部 16 位比较输出寄存器的对应位。 写访问 COOUTx 将改变内部比较输出寄存器的对应位。 读取位 COOUTx 返回内部比较输出寄存器的对应位的值。 在比较模式 1 或并行比较模式，通道发生有效比较匹配时，内部比较输出寄存器的值输出到对应 T2CCx 引脚。</p> <p><i>注：在初始化内部比较输出寄存器时，同时清除位 ENSHDW 和置位 / 清除位 COOUTx 有效。</i></p> <p><b>ENSHDW = 1</b></p> <p>在这种情况下，访问位 COOUTx 访问的是实际 SFR 位。 如果 COOUTx 被置位，寄存器 CCx 的内容将被传送到对应的通道 x 内部比较寄存器。 置位该位时，COOUTx 保持一个 T2CCU 时钟周期，之后由硬件自动清除。</p> <p><i>注：在启动映射传送时，同时置位 ENSHDW 和位 COOUTx（并清除位 TXOV）有效。</i></p>
<b>TXOV</b>	6	rw	<p><b>定时器溢出时使能映射传送</b></p> <p>置位该位使能所有比较通道的映射传送操作：当对应的基准定时器溢出时，将 T2CCU_CCx 中的映射值传送到相应的内部比较寄存器中。</p>
<b>ENSHDW</b>	7	rwh	<p><b>使用 COOUTx 使能映射传送</b></p> <p>清除该位，使能通过 COOUTx 位访问内部比较输出寄存器。 置位该位使用 COOUTx 位将 T2CCU_CCx 中的映射值传送到相应的通道 x 内部比较寄存器中。</p> <p><i>注：置位 TXOV 时，位 ENSHDW 将被内部硬件清除。为了保证 ENSHDW 被置位，用户必须确保 TXOV 被清除。</i></p>

## T2CCU\_COCON

## T2CCU 比较控制寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
CCM5	CCM4	CM5F	CM4F	POLB	POLA	COMOD	
rw	rw	rwh	rwh	rw	rw	rw	

符号	位	类型	描述
COMOD	[1:0]	rw	比较模式控制（比较通道） 00 选择比较模式 0 01 选择比较模式 1 10 选择并行比较模式 其它值 保留
POLA	2	rw	通道组 A 比较主动态电平 在比较模式 0，该位定义属于 A 组的比较通道输出的主动态电平。 0 比较匹配时输出高电平 1 比较匹配时输出低电平
POLB	3	rw	通道组 B 的比较主动态电平 在比较模式 0，该位定义属于 B 组的比较通道输出的主动态电平。 0 比较匹配时输出高电平 1 比较匹配时输出低电平
CM4F <sup>1)</sup>	4	rwh	比较通道 4 中断标志 比较通道 4 发生比较匹配事件时，该位被置位。 必须由软件清除该位。
CM5F <sup>1)</sup>	5	rwh	比较通道 5 中断标志 比较通道 5 发生比较匹配事件时，该位被置位。 必须由软件清除该位。
CCM4	6	rw	通道 4 比较使能 0 禁止比较 1 使能比较 参见下面的注释。
CCM5	7	rw	通道 5 比较匹配 0 禁止比较 1 使能比较 参见下面的注释。

1) 软件和硬件同时访问标志位时，硬件优先级更高。

注： 通过位 **COMOD** 设置比较模式（如果选择比较模式 **0**，还要设置 **POLA/B** 选择主动态电平）之后，应分两步使能比较通道 **4** 和 **5**。这样操作是为了确保启动相应的比较通道时，可根据所选择的模式产生期望的比较输出。

注： 比较通道 **4** 和 **5** 的中断使能分别由 **SCU SFR MODIEN.CM4EN** 和 **CM5EN** 控制。

## T2CCU\_CCxL

T2CCU 捕获 / 比较寄存器 x 低位

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
CCVALL							
rwh							

符号	位	类型	描述
CCVALL	[7:0]	rwh	<p><b>通道 x 捕获 / 比较低位字节值</b></p> <p><b>捕获</b> 读取时，CCVALL 返回最近捕获值的低位字节。</p> <p><b>比较</b> CCVALL 保存写入的低位字节。CCx 作为通道 x 内部比较寄存器的映射寄存器。只有相应的 T2CCU_COSHDW.COOUTx 被置位（ENSHDW = 1），或通道基准定时器溢出时，写入值才传送至内部比较寄存器。</p> <p>读出时，始终返回最近写入的值。</p> <p>注： 对于 T2CCU_CC4 和 T2CCU_CC5，仅支持比较功能。</p>

## T2CCU\_CCxH

T2CCU 捕获 / 比较寄存器 x 高位

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
CCVALH							
rwh							

符号	位	类型	描述
CCVALH	[7:0]	rwh	<p><b>通道 x 捕获 / 比较高位字节值</b></p> <p><u>捕获</u> 读取时，CCVALH 返回最近捕获值的高位字节。</p> <p><u>比较</u> CCVALH 保存写入的高位字节。CCx 作为通道 x 内部比较寄存器的映射寄存器。只有相应的 T2CCU_COSHDW.COOUTx 被置位 (ENSHDW = 1)，或通道基准定时器溢出时，写入值才传送至内部比较寄存器。读出时，始终返回最近写入的值。</p> <p><i>注： 对于 T2CCU_CC4 和 T2CCU_CC5，仅支持比较功能。</i></p>

## 15 捕获 / 比较单元 6

捕获 / 比较单元 6 (CCU6) 中有两个独立的定时器 (T12, T13)，可用来产生脉宽调制 (PWM) 信号，尤其适用于交流电机的控制。CCU6 还支持块切换和多相电机的特殊控制模式。CCU6 框图如图 15-1。

定时器 T12 的 3 路通道可工作在捕获和 / 或比较模式。定时器 T13 只能工作在比较模式。多通道控制单元产生的输出序列可由 T12 和 / 或 T13 调制。调制源可选择并组合使用。

### 定时器 T12 特性:

- 3 路捕获 / 比较通道，每路可用作捕获或比较通道
- 支持三相 PWM 产生 (6 路输出，每路信号对应上桥臂和下桥臂开关)
- 16 位精度
- 两个可选的工作频率：外设时钟和外设时钟的两倍
- 单通道死区时间控制，避免功率级短路
- 同步刷新 T12/T13 寄存器
- 产生中间对齐和边沿对齐 PWM
- 支持单次模式
- 支持多中断请求源
- 类磁滞控制模式

### 定时器 T13 特性:

- 单比较通道，单输出
- 16 位精度，最大计数频率 =  $2 \times$  外设时钟
- 可与 T12 同步
- 周期 - 匹配和比较 - 匹配产生中断
- 支持单次模式

### 附加特性:

- 实现无刷 DC 驱动电机的块切换
- 利用霍尔序列产生位置检测
- 块切换的自动转速测量
- 综合错误处理
- 通过外部信号 (CTRAPP) 快速紧急终止，无需 CPU 干预
- 多通道 AC 电机控制模式
- 输出电平可选，与功率级适配

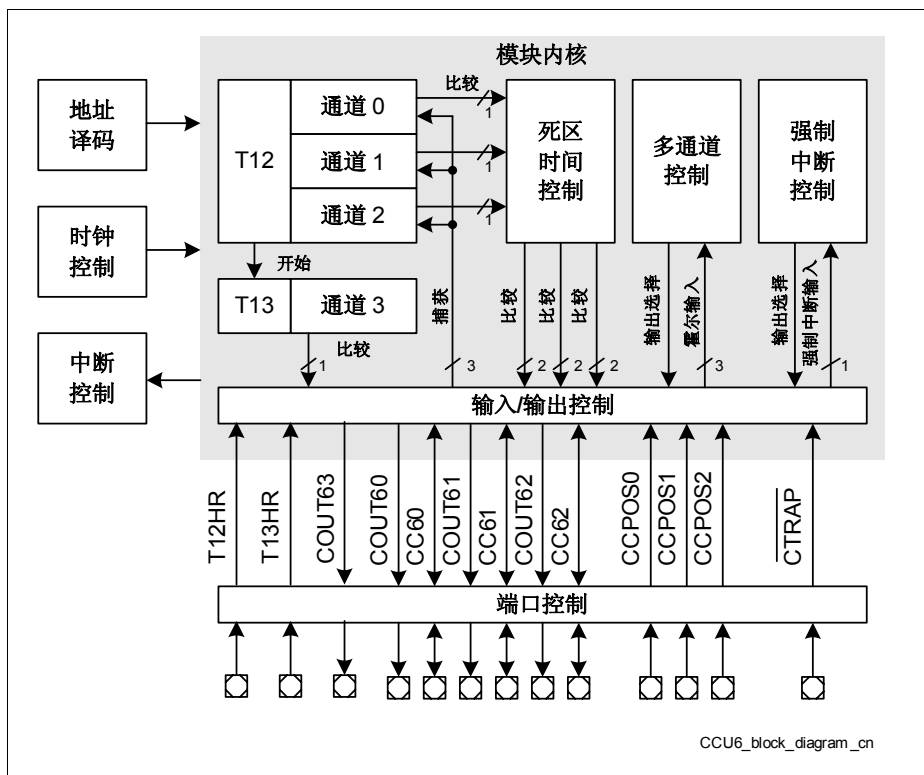


图 15-1 CCU6 框图





### 15.1.1.1 定时器配置

寄存器 T12 代表定时器 T12 的计数值。只有在定时器 T12 停止时才可写入；定时器 T12 工作时写操作无效。寄存器 T12 始终可由软件读取。

边沿对齐模式下，T12 只能递增计数；中间对齐模式下，T12 可递增和递减计数。

由硬件或软件控制位 T12R 以启动和停止定时器 T12。

- 位域 T12RSEL 定义了引脚 T12HR 上的事件：上升沿、下降沿、或二者任一，由硬件置位定时器运行控制位 T12R。
- 若位域 T12RSEL = 00<sub>B</sub>，禁止由外部事件设置 T12R；定时器的运行位只能由软件控制。软件置位 T12RS 或 T12RR 来置位或复位 T12R。
- 在单次模式下，根据位 T12SSC 定义的功能由硬件复位 T12R。若 T12SSC = 1，在下列情况硬件复位 T12R：
  - 边沿对齐模式下 T12 计数至周期值
  - 中间对齐模式下 T12 递减计数至 1

可置位 T12RES 将寄存器 T12 复位至 0。T12RES 的设置不影响运行位 T12R。

### 15.1.1.2 计数规则

以 T12 时钟作为输入，按如下计数规则定义计数序列：

#### T12 工作在边沿对齐模式（位 CTM = 0）

计数方向设置为递增计数（CDIR = 0）。检测到周期匹配时计数器复位至 0，若 STE12 = 1 定时器 T12 发生映射寄存器传送。

#### T12 工作在中间对齐模式下（位 CTM = 1）

- 递减计数检测到 1- 匹配时，计数方向设置为递增计数（CDIR = 0）
- 递增计数检测到周期 - 匹配时，计数方向设置为递减计数（CDIR = 1）
- 如果 STE12 = 1，在下列条件下发生映射寄存器传送：
  - 递增计数时检测到周期 - 匹配
  - 递减计数时检测到 1- 匹配

定时器 T12 不工作时预分频器复位，从而保证可重复产生时序和延迟。

### 15.1.1.3 切换规则

三路比较通道并行工作。计数方向不同，比较 - 匹配的含义不同。为了使 PWM 和输出电平无关，引入两个不同的状态用于比较：主动态和被动态。用这两个状态产生所需的 PWM，该调制信号是 T13 的控制、强制中断控制和多通道控制的逻辑组合。如果将主动态视为 1、被动态视为 0，比较状态之间进行逻辑与（AND）组合。

- 主动态 AND 主动态 = 主动态
- 主动态 AND 被动态 = 被动态
- 被动态 AND 被动态 = 被动态

根据检测到的比较 - 匹配改变比较状态，由 CC6xST 表示。T12 的比较状态定义如下：

- 如果计数值低于比较值，为被动态
- 如果计数值高于比较值，为主动态

从而得到以下的比较状态切换规则：

- 递增计数时，计数至比较值时，置位为主动态
- 递减计数时，计数至比较值时，复位为被动态
- 递增计数时，0- 匹配但不是比较 - 匹配时，复位为被动态
- 递增计数时，0- 匹配同时也是比较 - 匹配时，置位为主动态

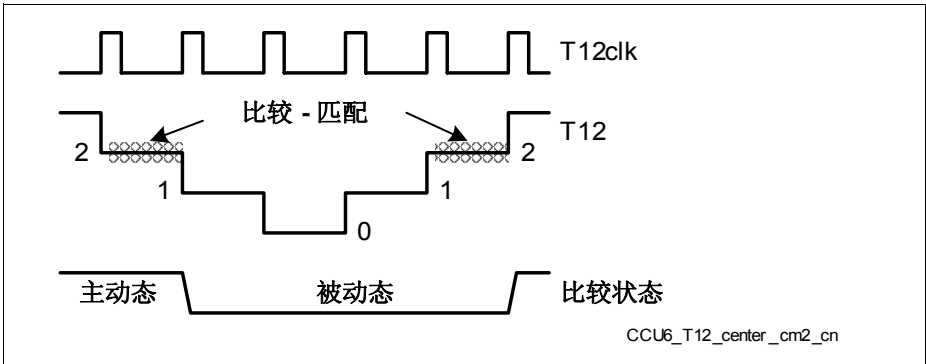


图 15-3 比较值 = 2 的比较状态

只有当定时器工作时才考虑切换规则。因此当定时器停止工作时，对定时器寄存器的写操作不会引起比较操作。

#### 15.1.1.4 T12 的比较模式

比较模式下，寄存器 CC6xR (x = 0 - 2) 是 T12 的实际比较寄存器。CC6xR 中存放的值和 T12 的计数值进行比较（三路通道并行工作）。寄存器 CC6xR 只可由软件读取；修改 CC6xR 时，通过映射寄存器将新值从寄存器 CC6xSR 传送到 CC6xR 中。

寄存器 T12PR 包含定时器的周期值。周期值和 T12 的实际计数值比较，根据定义的计数规则执行相应的计数动作。

图 15-4 以中间对齐模式下无死区时间的比较模式为例。位 CC6xST 指示对应通道发生比较或捕获事件。下列事件将置位 CC6xST（若 CC6xST 当前为 0）：

- 软件置位（MCC6xS）
- 若 T12 工作且 T12 的置位事件被使能，（T12 计数值大于比较值）比较置位事件
- 捕获置位事件

下列事件将复位 CC6xST（若 CC6xST 当前为 1）：

- 软件复位（MCC6xR）

## 捕获 / 比较单元 6

- 若 T12 工作且 T12 的复位事件被使能，(T12 计数值低于当前值时) 比较复位事件 (包括单次模式下 T12 计数到周期值后复位)
- 类磁滞控制模式下的复位事件

位 CC6xPS 代表被动态选择位。定时器 T12 的两条输出线 (CC6x, COUT6x) 可选择在 CC6xST 为 0 时 (CC6xPS = 0) 或在 CC6xST 为 1 时 (CC6xPS = 1) 处于被动态。

输出为被动态时所驱动的输出电平由位域 PSL 中的对应位来定义。

图 15-5 给出了对于不同应用时 CC6xPS/COUT6xPS 和 PSL 的设置。以中间对齐模式带有死区时间的比较模式为例。

只有在定时器 T12 工作时才可能通过硬件修改比较状态位。因此，可用 T12R 作为硬件修改的使能 / 禁止。

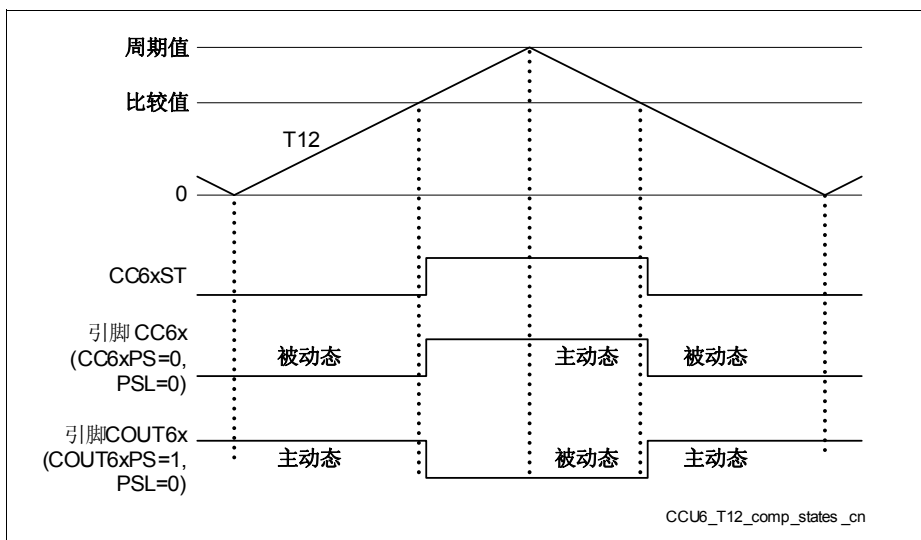


图 15-4 定时器 T12 的比较状态

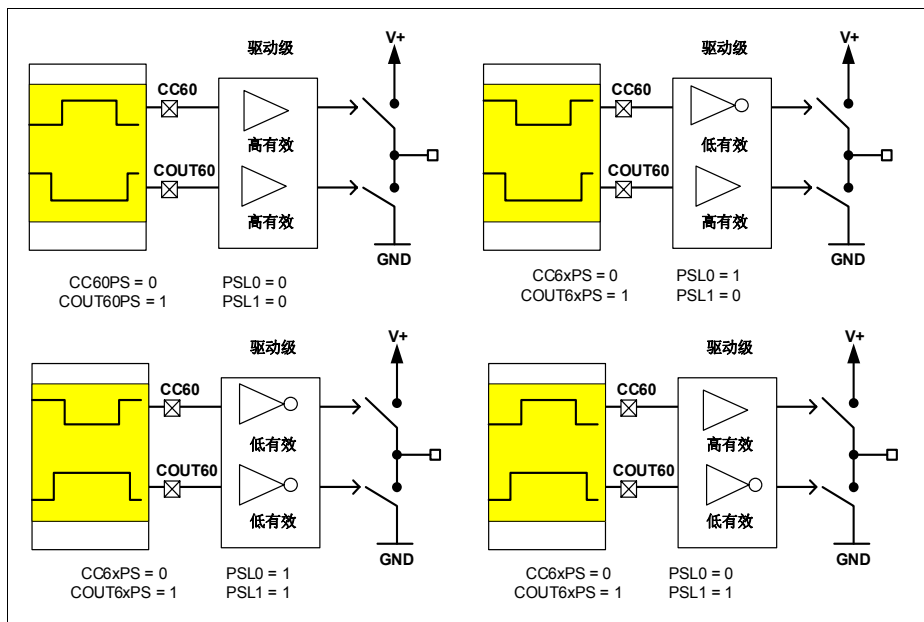


图 15-5 CC6xPS/COU6xPS 和 PSL 的不同设置

对于类磁滞控制模式 ( $MSEL6x = 1001_B$ ) (见 [章节 15.1.1.9](#))，只有在相应的输入  $CCPOSx = 1$  (无效) 时才可能设置比较状态位。

如果选择霍尔传感器模式 ( $MSEL6x = 1000_B$ ) (见 [章节 15.1.6](#))，由定时器 T12 来修改比较通道 1 和 2 的比较状态位，指示已经超过设置时间。

只有当  $CC6xST$  被复位时才可产生置位；只有当被置位时，才可复位。因此，置位和复位  $CC6xST$  的触发事件必须逻辑组合。置位和复位信号的逻辑或可触发死区时间计数器重载 (见 [图 15-6](#))。只有改变  $CC6xST$  时才能触发重载，产生正确的、带有死区时间的 PWM 信号；边沿对齐和中间对齐模式下占空比在 0% 到 100% 之间的 PWM 信号。

### 15.1.1.5 0% 和 100% 占空比

这些计数规则和切换规则可确保产生的 PWM 的占空比 (占空比 = 有效时间 / PWM 总周期) 范围在 0% 到 100% 之间。为了获得 0% 的占空比 (比较状态永远无效)，比较值必须设置为  $T12P+1$  的；比较值为 0 将会使 PWM 的占空比为 100% (比较状态始终有效)。

### 15.1.1.6 死区时间产生

大多数情况下，相互连接的功率开关的切换行为不对称，即打开和关闭开关所需时间不同。功率器件的开启时间小于关闭时间时通常会引发问题，这将导致反相器桥臂短路，从

而可能使整个系统瘫痪。为了用硬件解决这个问题，CCU6 中提供了一个可编程死区时间计数器，可延迟切换信号从被动沿跳变至主动沿的时间（主动沿到被动沿不延迟）。

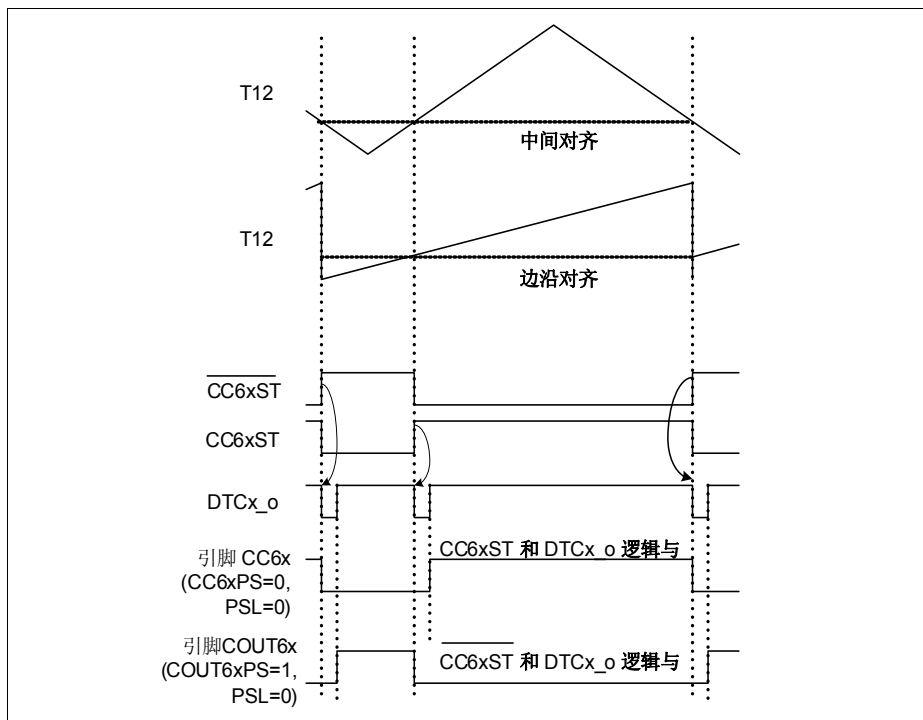


图 15-6 带有死区时间产生的 PWM 信号

寄存器 T12DTC 控制定时器 T12 比较通道的死区时间产生。每路通道可独立由位 DTE<sub>x</sub> 使能 / 禁止产生死区时间。如果允许产生死区时间，位域 DTM（以 T12CLK 为计数时钟的 8 位递减计数器）定义了从被动态跳变至主动态的延迟值。只有当死区时间计数器计数至 0 时才可被重载。

每一路通道和各自的死区时间计数器、触发和使能信号一起工作，三路通道之间工作相互独立。位域 DTM 的值对三路通道均有效。

### 15.1.1.7 捕获模式

捕获模式下，根据位域 MSEL6<sub>x</sub>，位 CC6xST 指示被选中的捕获事件的产生。

- MSEL6<sub>x</sub> = 01XX<sub>B</sub>，双寄存器捕获模式（见表 15-6）
- MSEL6<sub>x</sub> = 101X<sub>B</sub> 或 11XX<sub>B</sub>，多输入捕获模式（见表 15-8）

## 捕获 / 比较单元 6

可选择引脚 CC6x 或 CCPOSx 的上升沿和 / 或下降沿作为捕获事件，将定时器 T12 的内容传送到寄存器 CC6xR 和 CC6xSR 中。要工作在捕获模式，捕获引脚必须设置为输入引脚。

有多种方式保存寄存器中的捕获值。例如，双寄存器捕获模式，定时器的值保存到通道映射寄存器 CC6xSR 中；前次保存在该寄存器中的值同时被复制到通道寄存器 CC6xR 中。于是软件可检查新捕获的值，同时仍有可能读取前次的捕获值。

*注：捕获模式下，根据映射传送规则可请求映射传送，内容未改变的捕获 / 比较寄存器除外。*

### 15.1.1.8 单次模式

当位 T12SSC 被置 1 时，选择定时器 T12 工作在单次模式。单次模式下，定时器 T12 计数到周期值后自动停止工作。图 15-7 示出在边沿对齐和中间对齐模式下，定时器周期结束时的操作。如果 T12SSC 被置位时检测到周期事件结束，位 T12R 和所有 CC6xST 位被复位。

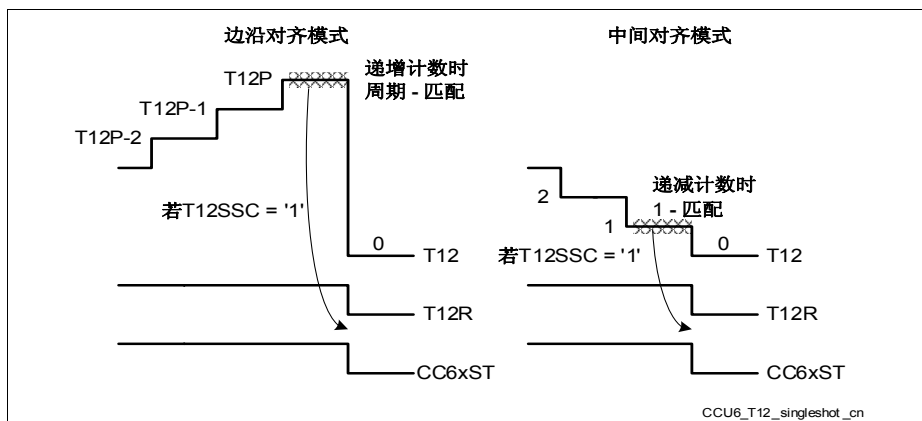


图 15-7 T12 单次模式结束

### 15.1.1.9 类磁滞控制模式

类磁滞控制模式 ( $MSEL6x = 1001_B$ ) 提供了这样的可能性：如果输入 CCPOSx 变为 0，通过复位 CC6xST 可关闭 PWM 输出。这可用作简单的电机控制特性，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 时才有可能置位 CC6xST。图 15-8 给出类磁滞控制的例子。

可利用该模式将和时序相关的行为引入到类磁滞控制器中。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，切换频率和占空比可不断改变。

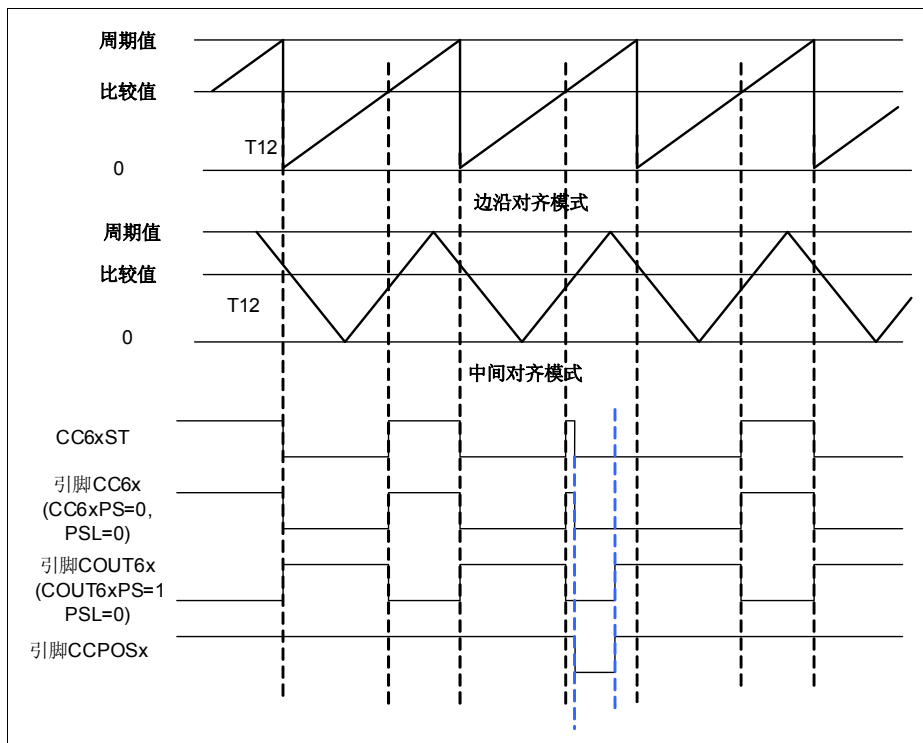


图 15-8 类磁滞控制模式



### 15.1.2 定时器 T13

定时器 T13 和定时器 T12 相似，差别仅在于在比较模式下 T13 只有一路比较通道。计数器只能递增计数（和 T12 的边沿对齐模式相似）。定时器 T13 的输入时钟范围可从  $f_{CCU6}$  至  $f_{CCU6}/128$ ，由位域 T13CLK 设置。为了支持更高的时钟频率，T13PRE = 1 时，可使用附加因子为 1/256 的预分频器，用作 T13 预分频。

周期 - 匹配时，T13 的映射传送由位 STE13 使能。T13 映射传送时，寄存器 CC63SR 的内容被传送到寄存器 CC63R 中。这两个寄存器均可由软件读取，但只有映射寄存器可由软件写入。

位 CC63PS、T13IM 和 PSL63 具有映射位。T13 映射传送时，这些映射位的内容被传送给实际使用位。写操作写入映射位；读操作读取实际使用位的值。

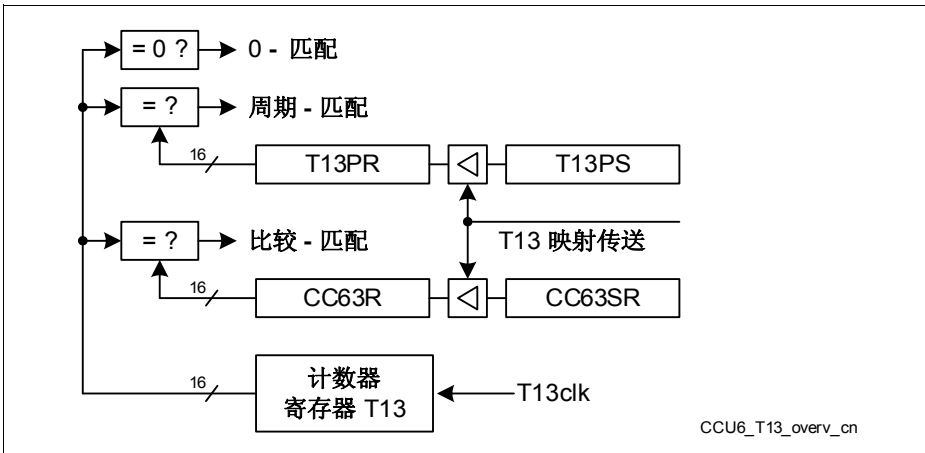


图 15-9 T13 功能概述

定时器 T13 计数和定时器 T12 在边沿对齐模式下的计数和切换规则相同。定时器 T13 的功能如图 15-9 所示。

#### 15.1.2.1 定时器配置

寄存器 T13 代表定时器 T13 的计数值。只有在定时器 T13 停止时才可写入；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 只支持边沿对齐模式（递增计数）。

由硬件或软件控制 T13R 以启动和停止定时器 T13。

- 软件置位 T13RS 或 T13RR 以置位或复位 T13R。
- 在单次模式下，若位 T13SSC = 1，当 T13 递增计数至周期值时，硬件复位 T13R。
- 位域 T13TEC 和 T13TED 选择触发事件置位 T13R，用于和不同的 T12 比较事件同步。

可通过置位 T13RES 将寄存器 T13 复位至 0。对 T13RES 的设置不会影响运行位 T13R。

### 15.1.2.2 比较模式

寄存器 CC63R 是 T13 的实际比较寄存器。CC63R 中保存的值和 T13 的计数值进行比较。寄存器 CC63R 只可由软件读取；修改 CC63R 时，通过映射寄存器将新值从寄存器 CC63SR 传送到 CC63R 中。相应的映射寄存器 CC63SR 可由软件读写。

寄存器 T13PR 包含了定时器 T13 的周期值。周期值和 T13 的实际计数值比较，根据定义的计数规则执行相应的计数动作。

位 CC63ST 指示相应通道发生比较事件。下列事件将置位 CC63ST（若 CC63ST 当前为 0）：

- 软件置位（MCC63S）
- 若 T13 工作且 T13 的置位事件被使能，（T13 计数值大于比较值的）比较置位事件

下列事件将复位 CC63ST（若 CC63ST 当前为 1）：

- 软件复位（MCC63R）
- 若 T13 工作且 T13 的复位事件被使能，（T13 计数值低于比较值的）比较复位事件（包括单次模式下 T13 计数到周期值后复位）

可用 T13 的 PWM 信号调制其它输出信号。为了使 COUT63 和内部的调制信号无关，可由位 T13IM 和 COUT63PS 独立选择比较状态。

### 15.1.2.3 单次模式

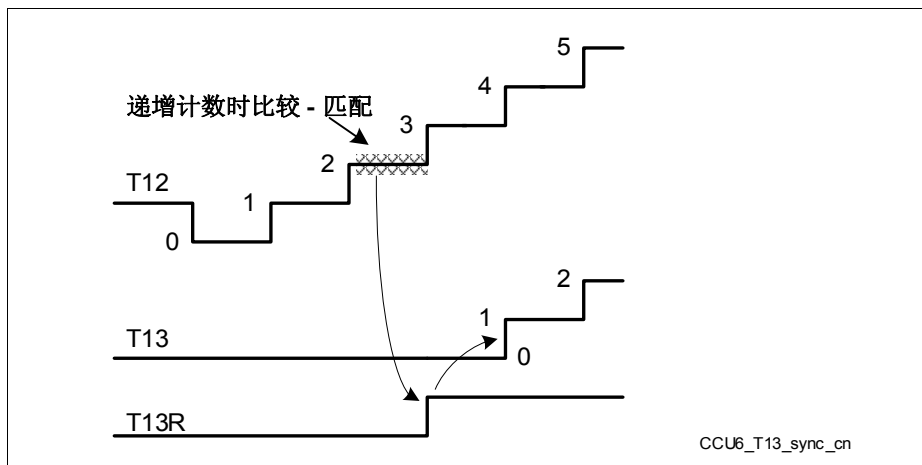
当位 T13SSC 置 1 时，选择定时器 T13 工作在单次模式。单次模式下，定时器 T13 计数到周期结束时自动停止工作。如果 T13SSC 被置位时检测到周期事件结束，位 T13R 和 CC63ST 被复位。

### 15.1.2.4 T13 与 T12 同步

定时器 T13 可由 T12 的事件同步。这些事件包括：

- 通道 0 的 T12 比较事件
- 通道 1 的 T12 比较事件
- 通道 2 的 T12 比较事件
- 通道 0、1 或 2 上的任意 T12 比较事件
- T12 周期 - 匹配
- T12 0- 匹配（递增计数时）
- 输入 CCPOSx 的任意边沿

位域 T13TEC 和 T13TED 选择触发事件，用于启动定时器 T13。该事件硬件置位 T13R，T13 开始计数。该特性和单次模式相结合，可产生一个 T12 事件之后的延迟，该延迟时间可编程设定。



**图 15-10 T13 与 T12 同步**

**图 15-10** 所示为 T13 与 T12 的事件同步，图中选择的 T12 事件为递增计数时的比较 - 匹配（比较值 = 2）。T12 和 T13 的时钟可不同（使用其它分频因子）。为了简化，图中举例的 T12CLK 和 T13CLK 相等。

### 15.1.3 调制控制

调制控制部分将不同的调制源逻辑组合（CC6x\_T12\_o 和 COUT6x\_T12\_o 是由 CC6xPS/COUT6xPS 配置后的输出信号；MOD\_T13\_o 是 T13 反相调制（T13IM）之后的输出信号）。每种调制源可被独立使能用于输出。此外，强制中断功能也归入调制控制，在强制中断状态下（若被使能）禁止相应输出线上的调制。

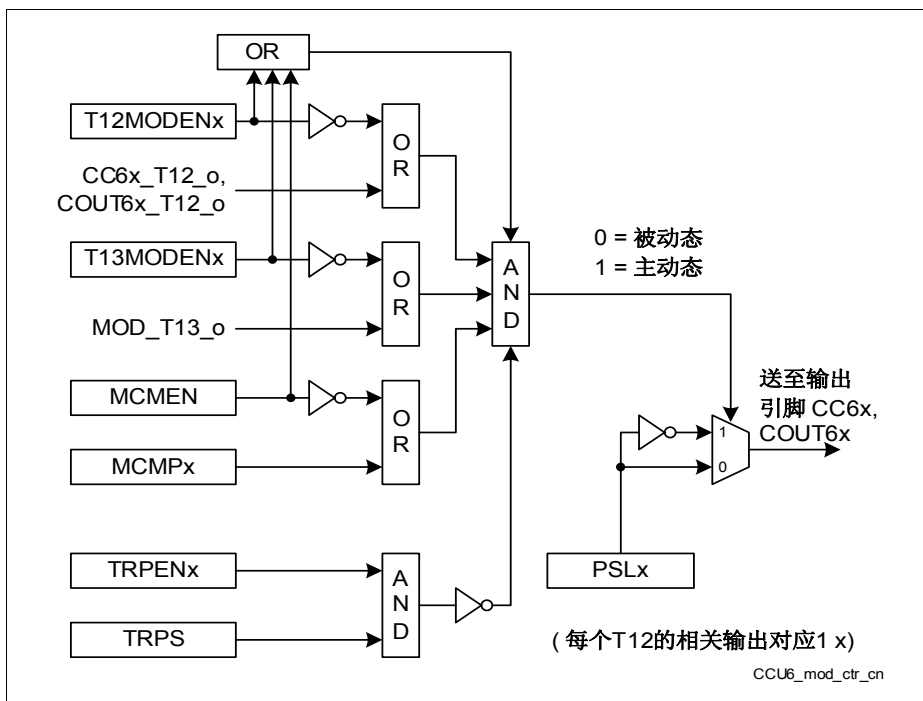


图 15-11 T12 相关输出的调制控制

图 15-11 给出 6 条与 T12 相关的输出线（由“x”表示）：

- T12MODENx 允许用定时器 T12 产生的 PWM 序列进行调制
- T13MODENx 允许用定时器 T13 产生的 PWM 序列进行调制
- MCMPx 选择多通道序列
- TRPENx 使能强制中断功能
- PSLx 定义了输出为被动态时的输出电平

如图 15-12 所示，T13 相关输出 COUT63 的调制控制部分将 T13 的输出信号（COUT63\_T13\_o 是由 COUT63PS 配置后的输出信号）、使能位 ECT13O 和强制中断功能逻辑组合。由 PSL63 选择被动态的输出电平。

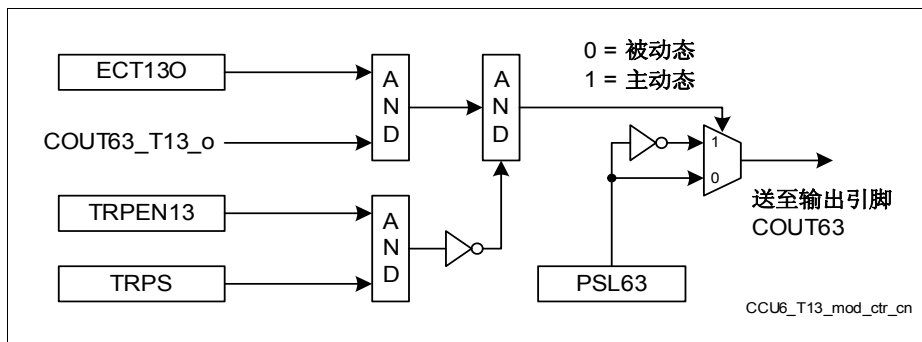


图 15-12 T13 相关输出 COUT63 的调制控制

图 15-13 给出 CC60 和 COUT60 调制控制的例子。

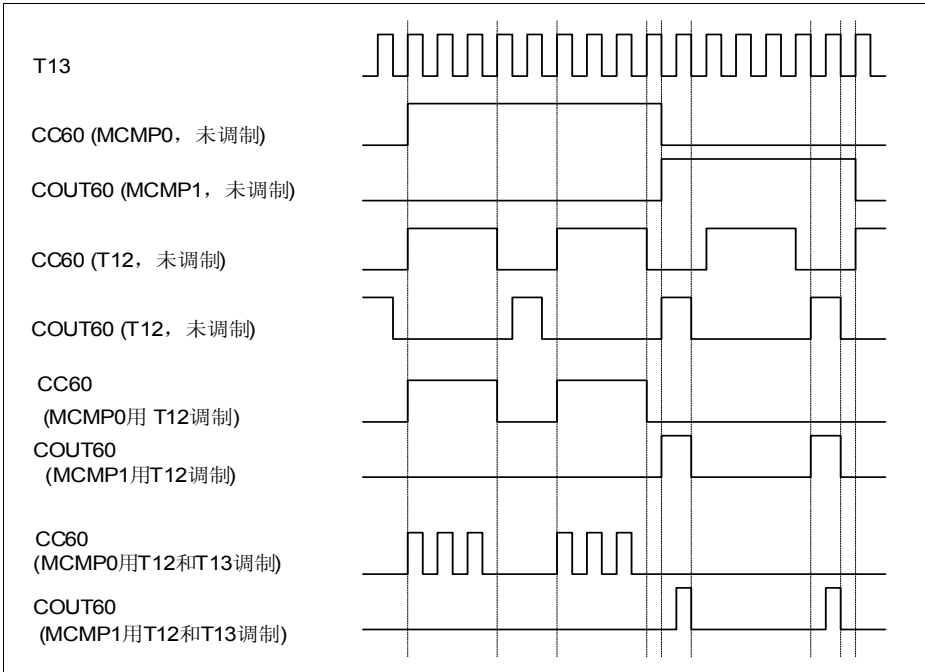


图 15-13 CC60 和 COUT60 的调制控制举例

#### 15.1.4 强制中断处理

强制中断功能使 PWM 输出能够响应输入引脚  $\overline{\text{CTRAP}}$  的状态。如果强制中断输入变为有效（如发生急停时），该功能可用于关闭功率器件。

在强制中断状态下，所选择的输出强制为被动态，无有效调制。若强制中断功能由位 TRPPEN 使能， $\overline{\text{CTRAP}}$  输入信号被拉低，由硬件控制系统立刻进入强制中断状态。也可由软件置位 TRPF（强制中断输入标志）进入该状态，并使得 TRPS = 1（强制中断状态指示标志）。当输入无效，软件控制退出该强制中断状态，并和下列事件同步：

- （如果 TRPM2 = 0）， $\overline{\text{CTRAP}}$  无效后 TRPF 自动复位
- （如果 TRPM2 = 1）， $\overline{\text{CTRAP}}$  无效后 TRPF 必须由软件复位
- TRPF 复位后与 T12 的 PWM 同步  
（T12 边沿对齐模式的周期 - 匹配或中间对齐模式递减计数时 1 - 匹配）
- TRPF 复位后与 T13 的 PWM 同步  
（T13 周期 - 匹配）
- 不与 T12 或 T13 同步

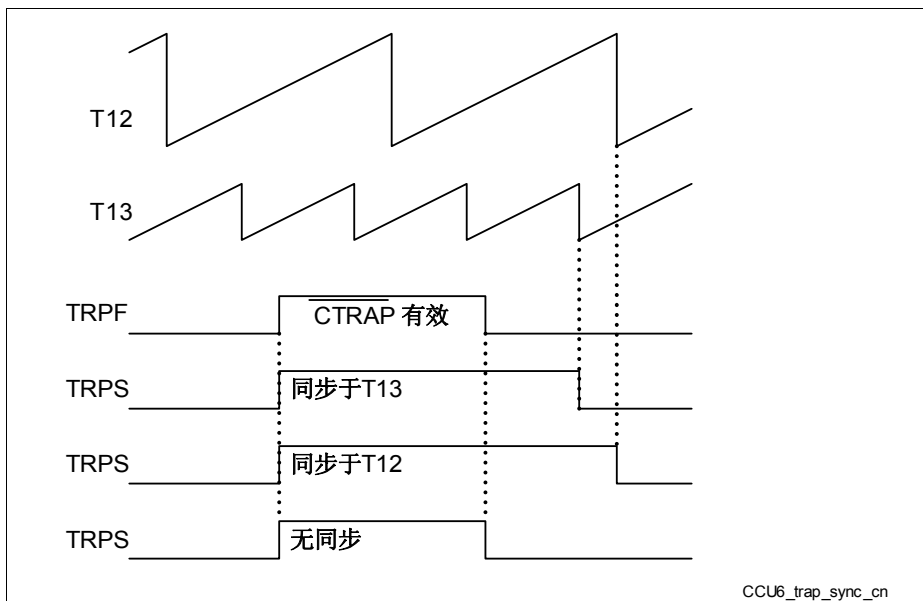


图 15-14 强制中断状态的同步 (TRPM2 = 0)





捕获 / 比较单元 6

---

- 递增计数时 T12 周期 - 匹配 (T12pm)
- 递减计数时 T12 1- 匹配 (T12om)
- T13 周期 - 匹配 (T13pm)
- 通道 1 的 T12 比较匹配 (T12c1cm)
- 正确霍尔事件

可能的硬件同步事件有:

- 递增计数时 T12 0- 匹配 (T12zm)
- T13 0- 匹配 (T13zm)

## 15.1.6 霍尔传感器模式

在无刷直流电机中，下一个多通道状态值由霍尔输入序列决定。霍尔序列（CURH）和调制序列（MCMP）之间关联性强。电机类型不同，驱动电机的调制序列有所不同。因此，灵活定义霍尔序列和相应调制序列之间的关联关系是有好处的。CCU6 具备这样的特性，通过存放实际霍尔序列（CURHS）、下次期望的霍尔序列（EXPHS）以及输出序列（MCMPS）的寄存器来实现。每次产生正确的霍尔事件时，新的霍尔序列和相应的输出序列（取自预定义的表）可由软件载入寄存器 MCMOUTS 中。也可在 MCMOUTS 中写入 STRHP = 1 将映射寄存器内容载入。有相位延迟时（由 T12 通道 1 产生），发生多通道模式映射传送（由位 STR 指示）时可载入新序列。

### 15.1.6.1 霍尔序列采样

以模块时钟  $f_{CCU6}$  采样（CCPOSx 引脚上的）霍尔序列。霍尔传感器模式下（模式 MSEL6x = 1000<sub>B</sub>），死区时间计数器 DTC0 可用作硬件噪声滤波器来抑制霍尔输入的毛刺。霍尔事件到来时，DTC0 被重载并开始计数，在被检测到的事件和采样点之间产生一个延迟。计数器计数至 1 时，采样 CCPOSx 输入信号（无噪声和毛刺），将采样值和当前的霍尔序列（CURH）以及期望霍尔序列（EXPH）进行比较。若采样序列和当前序列相同，这意味着 CCPOSx 的跳变沿由噪声毛刺造成，故不触发任何动作（隐含延迟导致的噪声滤波）。若采样序列和下一个期望序列相同，CCPOSx 的跳变沿为正确的霍尔事件，位 CHE 置位引起中断。

如果要求多通道模式和霍尔序列比较独立于定时器 T12 工作，由 DTC0 产生延迟可被旁路。这种情况下，定时器 T12 可用作其它用途。

位域 HSYNC 定义了触发事件，触发采样霍尔输入序列，并和当前以及期望的霍尔序列相比较。也可写入 SWHC = 1 软件触发霍尔比较。硬件采样触发源包括：

- 输入 CCPOSx (x = 0 - 2) 上的任意边沿
- T13 比较 - 匹配
- T13 周期 - 匹配
- T12 周期 - 匹配（递增计数）
- T12 1- 匹配（递减计数）
- T12 通道 0 比较匹配（递增计数）
- T12 通道 0 比较匹配（递减计数）

正确的霍尔事件可作为寄存器 MCMOUTS 的传送请求事件。映射传送将新的 CURH 序列和下一个 EXPH 序列从 MCMOUTS 复制到 MCMOUT 中。如果霍尔输入采样既非当前序列也非期望序列，位 WHE（错误霍尔事件）置位，从而引起中断并设置空闲模式对 MCMP 清零（调制输出无效）。从空闲模式重新启动工作，必须由软件（位 STRHP 和位域 SWSEL/SWSYN）激活 MCMOUTS 的传送请求。

### 15.1.6.2 无刷直流控制

对于无刷直流电机，有一种特殊的模式（MSEL6x = 1000<sub>B</sub>），是由霍尔输入（CCPOSx）的变化来触发的。在该模式下，T12 的通道 0 用作捕获功能，通道 1 和 2 用作比较功能（无输出调制），多通道模块和可能的 T13 调制一起触发输出切换。

## 捕获 / 比较单元 6

检测到有效的霍尔事件跳变沿，T12 的计数值被捕获到通道 0（代表电机的实际转速）并将 T12 复位。通道 1 的定时器计数至比较值时，触发位域 MCMP 的映射传送、切换到下个多通道状态。该触发事件可和某些条件相结合，用来实现噪声滤波（正确的霍尔事件）并使下个多通道状态与调制源同步（以避免输出毛刺）。如果不使用霍尔传感器，而用无传感器反电势技术，用通道 1 的比较功能产生位置输入到输出切换之间的相位延迟。通道 2 的比较值可用作超时触发（中断），指示电机的目标转速远远低于期望转速（由异常的负载变化所致）。该模式下，必须禁止 T12 调制（T12MODENx = 0）。

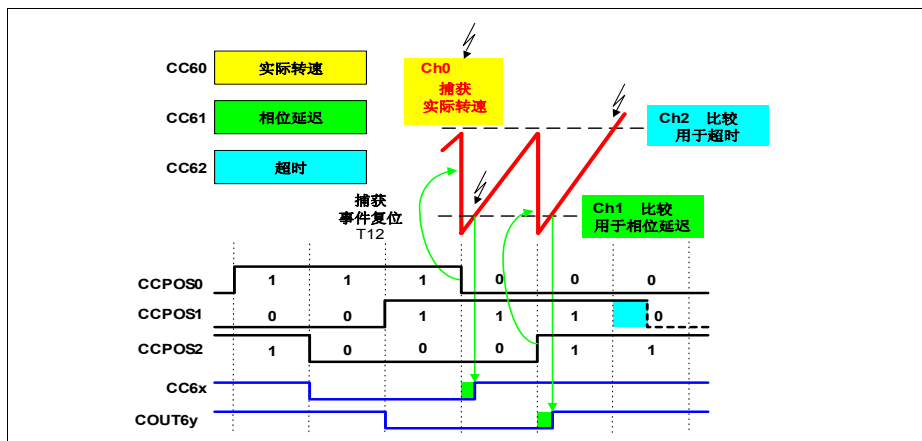


图 15-16 定时器 T12 无刷直流模式 (所有 MSEL6x = 1000<sub>B</sub>)

**表 15-1** 列出一个无刷直流电机控制中块切换的例子。如果输入信号组合 CCPOS0-CCPOS2 改变状态，输出 CC6x 和 COUT6x 相应被设置为新的状态

**图 15-17** 示出左转模式的块切换；**图 15-18** 示出右转模式的块切换。这些图直接根据表 15-1 得到。

表 15-1 块切换控制标

模式	CCPOS0- CCPOS2 输入			CC60 - CC62 输出			COUT60 - COUT62 输出		
	CCP OS0	CCP OS1	CCP OS2	CC60	CC61	CC62	COUT 60	COUT 61	COUT 62
左转， 0° 相移	1	0	1	无效	无效	有效	无效	有效	无效
	1	0	0	无效	无效	有效	有效	无效	无效
	1	1	0	无效	有效	无效	有效	无效	无效
	0	1	0	无效	有效	无效	无效	无效	有效
	0	1	1	有效	无效	无效	无效	无效	有效
	0	0	1	有效	无效	无效	无效	有效	无效
右转	1	1	0	有效	无效	无效	无效	有效	无效
	1	0	0	有效	无效	无效	无效	无效	有效
	1	0	1	无效	有效	无效	无效	无效	有效
	0	0	1	无效	有效	无效	有效	无效	无效
	0	1	1	无效	无效	有效	有效	无效	无效
	0	1	0	无效	无效	有效	inactive	有效	无效
低速	X	X	X	无效	无效	无效	有效	有效	有效
空闲 <sup>1)</sup>	X	X	X	无效	无效	无效	无效	无效	无效

<sup>1)</sup> 假如霍尔输入的采样值既不是当前的霍尔序列也不是期望的霍尔序列，位 WHE（错误霍尔事件）被置位，它可引起中断并置位空闲模式对 MCMP 清零（调制输出无效）。

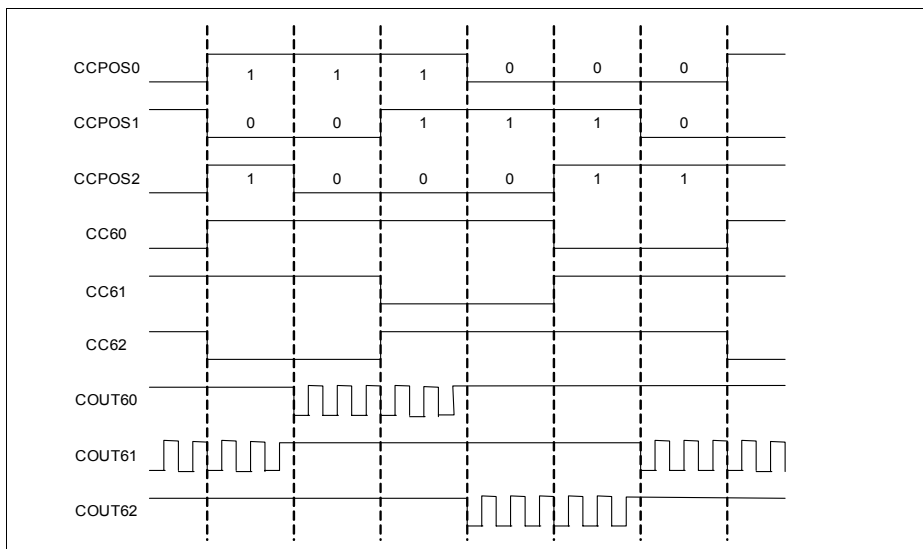


图 15-17 左转模式下的块切换

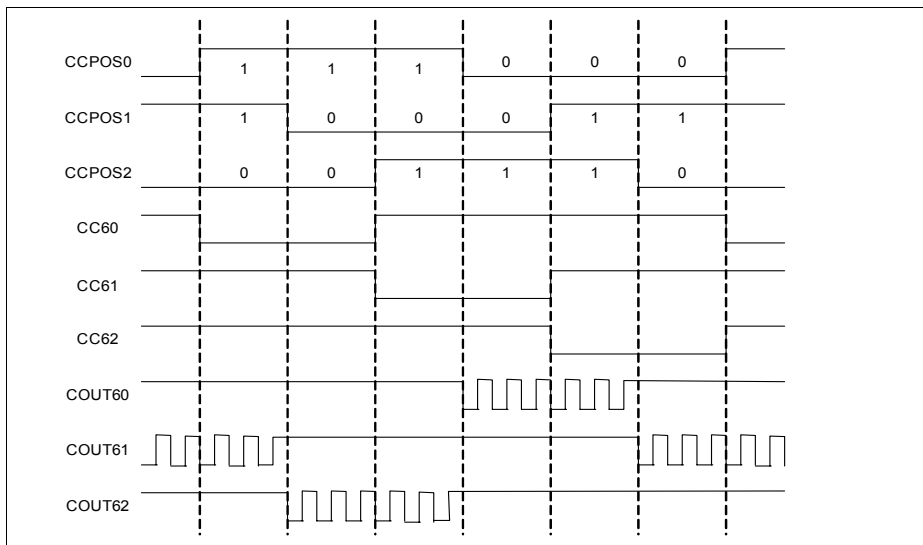


图 15-18 右转模式下的块切换

### 15.1.7 中断产生

可由中断事件触发中断，或软件置位寄存器 **IS** 的相应中断位产生中断。中断的产生和寄存器 **IS** 中的中断标志无关。寄存器 **IS** 只可读；对该寄存器的写操作不会改变其内容。对寄存器 **ISS** 或寄存器 **ISR** 的写操作可分别置位或复位寄存器 **IS** 中的各状态位。

如果寄存器 **IEN** 中的相关中断使能位有效，将产生中断。设置中断节点指针寄存器 **INP**，**CCU6** 模块的中断源可被映射到 4 条中断输出线上。

### 15.1.8 低功率模式

如果完全不需要 CCU6 功能，可关闭其时钟输入最大限度的降低功耗。该模式通过置位寄存器 PMCON1 的位 CCU\_DIS 来实现。外设时钟管理的具体内容请参见[章节 8.1.4](#)。

**PMCON1**  
功率模式控制寄存器 1 复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	CDC_DIS	CAN_DIS	MDU_DIS	T2CCU_DIS	CCU_DIS	SSC_DIS	ADC_DIS
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
CCU_DIS	2	rw	<b>CCU6 禁止请求位，高有效</b> 0    CCU6 处于正常工作状态（缺省值） 1    请求禁止 CCU6
0	7	r	<b>保留</b> 读操作返回 0；应写入 0。

15.1.9 模块挂起控制

当进入 OCDS 监控器模式时（见[章节 18.3](#)），可通过设置 CCU6 模块 SFR MODSUSP 中的模块挂起位 T12SUSP 和 T13SUSP，终止定时器 T12 和定时器 T13 计数。

MODSUSP

模块挂起控制寄存器

复位值：01<sub>H</sub>

7	6	5	4	3	2	1	0
0		CCTSUSP	T21SUSP	T2SUSP	T13SUSP	T12SUSP	WDTSUSP
r		rw	rw	rw	rw	rw	rw

符号	位	类型	描述
T12SUSP	1	rw	定时器 T12 调试挂起位 0 定时器 T12 不会被挂起 1 定时器 T12 将被挂起
T13SUSP	2	rw	定时器 T13 调试挂起位 0 定时器 T13 不会被挂起 1 定时器 T13 将被挂起
0	[7:6]	r	保留 读操作返回 0；应写入 0。



## 15.1.10 端口连接

**表 15-2** 给出满足 CCU6 I/O 功能要求的位和位域设置。该表也给出外设输入选择寄存器的值。

**表 15-2 CCU6 I/O 控制选择**

端口线	PISEL 寄存器位	I/O 控制寄存器位	I/O
P3.6/CTRAP_0	ISTRP = 00 <sub>B</sub>	P3_DIR.P6 = 0	输入
P5.5/CTRAP_1	ISTRP = 01 <sub>B</sub>	P5_DIR.P5 = 0	输入
P0.2/CTRAP_2	ISTRP = 10 <sub>B</sub>	P0_DIR.P2 = 0	输入
P4.7/CTRAP_3	ISTRP = 11 <sub>B</sub>	P4_DIR.P7 = 0	输入
P5.3/CCPOS0_0	ISPOS0 = 00 <sub>B</sub>	P5_DIR.P3 = 0	输入
P1.5/CCPOS0_1	ISPOS0 = 01 <sub>B</sub>	P1_DIR.P5 = 0	输入
P3.1/CCPOS0_2	ISPOS0 = 10 <sub>B</sub>	P3_DIR.P1 = 0	输入
P4.4/CCPOS0_3	ISPOS0 = 11 <sub>B</sub>	P4_DIR.P4 = 0	输入
P5.4/CCPOS1_0	ISPOS1 = 00 <sub>B</sub>	P5_DIR.P4 = 0	输入
P1.6/CCPOS1_1	ISPOS1 = 01 <sub>B</sub>	P1_DIR.P6 = 0	输入
P3.0/CCPOS1_2	ISPOS1 = 10 <sub>B</sub>	P3_DIR.P0 = 0	输入
P4.5/CCPOS1_3	ISPOS1 = 11 <sub>B</sub>	P4_DIR.P5 = 0	输入
P5.5/CCPOS2_0	ISPOS2 = 00 <sub>B</sub>	P5_DIR.P5 = 0	输入
P1.7/CCPOS2_1	ISPOS2 = 01 <sub>B</sub>	P1_DIR.P7 = 0	输入
P3.2/CCPOS2_2	ISPOS2 = 10 <sub>B</sub>	P3_DIR.P2 = 0	输入
P4.6/CCPOS2_3	ISPOS2 = 11 <sub>B</sub>	P4_DIR.P6 = 0	输入
P3.0/CC60_0	ISCC60 = 00 <sub>B</sub>	P3_DIR.P0 = 0	输入
	–	P3_DIR.P0 = 1	输出
		P3_ALTSEL0.P0 = 1	
		P3_ALTSEL1.P0 = 0	
P4.0/CC60_1	–	P4_DIR.P0 = 1	输出
		P4_ALTSEL0.P0 = 1	
		P4_ALTSEL1.P0 = 0	
P5.5/CC60_3	ISCC60 = 11 <sub>B</sub>	P5_DIR.P5 = 0	输入
P3.1/COU60_0	–	P3_DIR.P1 = 1	输出
		P3_ALTSEL0.P1 = 1	
		P3_ALTSEL1.P1 = 0	

**表 15-2 CCU6 I/O 控制选择**

端口线	PISEL 寄存器位	I/O 控制寄存器位	I/O
P4.1/COUT60_1	—	P4_DIR.P1 = 1	输出
		P4_ALTSEL0.P1 = 1	
		P4_ALTSEL1.P1 = 0	
P3.2/CC61_0	ISCC61 = 00 <sub>B</sub>	P3_DIR.P2 = 0	输入
	—	P3_DIR.P2 = 1	输出
		P3_ALTSEL0.P2 = 1	
		P3_ALTSEL1.P2 = 0	
P0.0/CC61_1	ISCC61 = 01 <sub>B</sub>	P0_DIR.P0 = 0	输入
	—	P0_DIR.P0 = 1	输出
		P0_ALTSEL0.P0 = 0	
		P0_ALTSEL1.P0 = 1	
P3.1/CC61_2	ISCC61 = 10 <sub>B</sub>	P3_DIR.P1 = 0	输入
	—	P3_DIR.P1 = 1	输出
		P3_ALTSEL0.P1 = 0	
		P3_ALTSEL1.P1 = 1	
P5.3/CC61_3	ISCC61 = 11 <sub>B</sub>	P5_DIR.P3 = 0	输入
P4.4/CC61_4	—	P4_DIR.P4 = 1	输出
		P4_ALTSEL0.P4 = 1	
		P4_ALTSEL1.P4 = 0	
P3.3/COUT61_0	—	P3_DIR.P3 = 1	输出
		P3_ALTSEL0.P3 = 1	
		P3_ALTSEL1.P3 = 0	
P0.1/COUT61_1	—	P0_DIR.P1 = 1	输出
		P0_ALTSEL0.P1 = 0	
		P0_ALTSEL1.P1 = 1	
P4.5/COUT61_2	—	P4_DIR.P5 = 1	输出
		P4_ALTSEL0.P5 = 1	
		P4_ALTSEL1.P5 = 0	

表 15-2 CCU6 I/O 控制选择

端口线	PISEL 寄存器位	I/O 控制寄存器位	I/O
P3.4/CC62_0	ISCC62= 00 <sub>B</sub>	P3_DIR.P4 = 0	输入
	—	P3_DIR.P4 = 1	输出
		P3_ALTSEL0.P4 = 1	
		P3_ALTSEL1.P4 = 0	
P0.4/CC62_1	ISCC62 = 01 <sub>B</sub>	P0_DIR.P4 = 0	输入
	—	P0_DIR.P4 = 1	输出
		P0_ALTSEL0.P4 = 0	
		P0_ALTSEL1.P4 = 1	
P4.6/CC62_2	—	P4_DIR.P6 = 1	输出
		P4_ALTSEL0.P6 = 1	
		P4_ALTSEL1.P6 = 0	
P5.4/CC62_3	ISCC62 = 11 <sub>B</sub>	P5_DIR.P4 = 0	输入
P3.5/COUT62_0	—	P3_DIR.P5 = 1	输出
		P3_ALTSEL0.P5 = 1	
		P3_ALTSEL1.P5 = 0	
P0.5/COUT62_1	—	P0_DIR.P5 = 1	输出
		P0_ALTSEL0.P5 = 0	
		P0_ALTSEL1.P5 = 1	
P4.7/COUT62_2	—	P4_DIR.P7 = 1	输出
		P4_ALTSEL0.P7 = 1	
		P4_ALTSEL1.P7 = 0	
P3.7/COUT63_0	—	P3_DIR.P7 = 1	输出
		P3_ALTSEL0.P7 = 1	
		P3_ALTSEL1.P7 = 0	
P0.3/COUT63_1	—	P0_DIR.P3 = 1	输出
		P0_ALTSEL0.P3 = 0	
		P0_ALTSEL1.P3 = 1	
P4.3/COUT63_2	—	P4_DIR.P3 = 1	输出
		P4_ALTSEL0.P3 = 0	
		P4_ALTSEL1.P3 = 1	

表 15-2 CCU6 I/O 控制选择

端口线	PISEL 寄存器位	I/O 控制寄存器位	I/O
P1.6/T12HR_0	IST12HR = 00 <sub>B</sub>	P1_DIR.P6 = 0	输入
P0.0/T12HR_1	IST12HR = 01 <sub>B</sub>	P0_DIR.P0 = 0	输入
P5.3/T12HR_2	IST12HR = 10 <sub>B</sub>	P5_DIR.P3 = 0	输入
P1.7/T13HR_0	IST13HR = 00 <sub>B</sub>	P1_DIR.P7 = 0	输入
P0.1/T13HR_1	IST13HR = 01 <sub>B</sub>	P0_DIR.P1 = 0	输入
P5.4/T13HR_2	IST13HR = 10 <sub>B</sub>	P5_DIR.P4 = 0	输入

### 15.1.11 定时器同步启动

定时器 T12 和 T13 能够与 T2CCU 模块中的 CCT 定时器同步启动。通过选择 T12HR\_3 和 T13HR\_3 分别作为 T12HR 输入和 T13HR 输入的方式使能同步启动。在 XC878 中，T2CCU 模块的 CCTST 信号与 T12HR\_3 和 T13HR\_3 相连。如果通过寄存器 TCTR2H 中的位 T12RSEL 选择 T12HR\_3 上的上升沿作为有效边沿，T2CCU\_CCTCON 寄存器中的 CCTST 从 0 到置位的操作将置位 T12R。T13 也可和 T12 以及 CCT 同步启动，如表 15-3 所述。

**表 15-3 选择定时器同步启动**

CCU 定时器输入	PISEL 寄存器位	定时器运行选择
T12HR_3	IST12HR = 11 <sub>B</sub>	TCTR2H.T12RSEL
T13HR_3	IST13HR = 11 <sub>B</sub>	TCTR2H.T13RSEL

## 15.2 寄存器映射

CCU6 的 SFR 在标准存储区 (RMAP = 0)，由四页构成。CCU6\_PAGE 寄存器位于地址 A3<sub>H</sub> 处，包含分页信息和页控制信息。

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添加模块名前缀 "CCU6\_"，例如 CCU6\_CC63SRL。

内核 SFR 地址（非映射）列于表 15-4.

表 15-4 页 0-3 的 SFR 地址列表

地址	页 0	页 1	页 2	页 3
9A <sub>H</sub>	CC63SRL	CC63RL	T12MSELL	MCMOUTL
9B <sub>H</sub>	CC63SRH	CC63RH	T12MSELH	MCMOUTH
9C <sub>H</sub>	TCTR4L	T12PRL	IENL	ISL
9D <sub>H</sub>	TCTR4H	T12PRH	IENH	ISH
9E <sub>H</sub>	MCMOUTSL	T13PRL	INPL	PISEL0L
9F <sub>H</sub>	MCMOUTSH	T13PRH	INPH	PISEL0H
A4 <sub>H</sub>	ISRL	T12DTCL	ISSL	PISEL2
A5 <sub>H</sub>	ISRH	T12DTCH	ISSH	
A6 <sub>H</sub>	CMPMODIFL	TCTR0L	PSLR	
A7 <sub>H</sub>	CMPMODIFH	TCTR0H	MCMCTR	
FA <sub>H</sub>	CC60SRL	CC60RL	TCTR2L	T12L
FB <sub>H</sub>	CC60SRH	CC60RH	TCTR2H	T12H
FC <sub>H</sub>	CC61SRL	CC61RL	MODCTRL	T13L
FD <sub>H</sub>	CC61SRH	CC61RH	MODCTRH	T13H
FE <sub>H</sub>	CC62SRL	CC62RL	TRPCTRL	CMPSTATL
FF <sub>H</sub>	CC62SRH	CC62RH	TRPCTRH	CMPSTATH

### CCU6\_PAGE

CCU6 分页寄存器

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>	[2:0]	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 $page = addr[y:x+1]$ .
<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 $OP = 10_B$ ， <b>PAGE</b> 的内容在被新值覆盖之前保存在 <b>STx</b> 中。 若 $OP = 11_B$ ， <b>PAGE</b> 的内容被 <b>STx</b> 覆盖。写入 <b>PAGE</b> 的值不予理睬。  00 选择 <b>ST0</b> 01 选择 <b>ST1</b> 10 选择 <b>ST2</b> 11 选择 <b>ST3</b>
<b>OP</b>	[7:6]	w	<b>操作</b> 0X 手动保存页模式， <b>STNR</b> 的值被忽略， <b>PAGE</b> 被直接写入。 10 带有自动页保存的新页设置。当前写入 <b>PAGE</b> 中的内容被保存的同时，上次写入 <b>PAGE</b> 中的内容被保存在 <b>STNR</b> 指定的位域 <b>STx</b> 中 11 自动恢复页操作。对写入 <b>PAGE</b> 的内容不予理睬， <b>PAGE</b> 的内容由 <b>STNR</b> 指定的位域 <b>STx</b> 中的值覆盖。
<b>0</b>	3	r	<b>保留</b> 读操作返回 0；应写入 0。

## 15.3 寄存器描述

**表 15-5** 列出所有 CCU6 模块的相关寄存器。

对于所有的 CCU6 寄存器：只写位（由 "w" 表示）读取值始终为 0。当硬件和软件同时请求修改某位时，软件占优。

**表 15-5 Registers Overview 寄存器概览**

寄存器缩写名	寄存器全名	描述见
<b>系统寄存器</b>		
PISEL0L	端口输入选择寄存器 0，低位字节	<a href="#">页 15-36</a>
PISEL0H	端口输入选择寄存器 0，高位字节	<a href="#">页 15-37</a>
PISEL2	端口输入选择寄存器 2	<a href="#">页 15-38</a>
<b>定时器 T12 寄存器</b>		
T12L	定时器 T12 计数寄存器，低位字节	<a href="#">页 15-43</a>
T12H	定时器 T12 计数寄存器，高位字节	<a href="#">页 15-44</a>
T12PRL	定时器 T12 周期寄存器，低位字节	<a href="#">页 15-44</a>
T12PRH	定时器 T12 周期寄存器，高位字节	<a href="#">页 15-45</a>
CC6xRL	通道 CC6x 捕获 / 比较寄存器，低位字节	<a href="#">页 15-45</a>
CC6xRH	通道 CC6x 捕获 / 比较寄存器，高位字节	<a href="#">页 15-45</a>
CC6xSRL	通道 CC6x 捕获 / 比较映射寄存器，低位字节	<a href="#">页 15-46</a>
CC6xSRH	通道 CC6x 捕获 / 比较映射寄存器，高位字节	<a href="#">页 15-46</a>
T12DTCL	定时器 T12 死区时间控制寄存器，低位字节	<a href="#">页 15-47</a>
T12DTCH	定时器 T12 死区时间控制寄存器，高位字节	<a href="#">页 15-47</a>
<b>定时器 T13 寄存器</b>		
T13L	定时器 T13 计数寄存器，低位字节	<a href="#">页 15-48</a>
T13H	定时器 T13 计数寄存器，高位字节	<a href="#">页 15-48</a>
T13PRL	定时器 T13 周期寄存器，低位字节	<a href="#">页 15-49</a>
T13PRH	定时器 T13 周期寄存器，高位字节	<a href="#">页 15-49</a>
CC63RL	通道 CC63 捕获 / 比较寄存器，低位字节	<a href="#">页 15-50</a>
CC63RH	通道 CC63 捕获 / 比较寄存器，高位字节	<a href="#">页 15-50</a>
CC63SRL	通道 CC63 捕获 / 比较映射寄存器，低位字节	<a href="#">页 15-50</a>
CC63SRH	通道 CC63 捕获 / 比较映射寄存器，高位字节	<a href="#">页 15-51</a>
<b>CCU6 控制寄存器</b>		
CMPSTATL	比较状态寄存器，低位字节	<a href="#">页 15-51</a>



表 15-5 Registers Overview 寄存器概览

寄存器缩写名	寄存器全名	描述见
<b>CMPSTATH</b>	比较状态寄存器, 高位字节	页 15-52
<b>CMPMODIFL</b>	比较状态修改寄存器, 低位字节	页 15-53
<b>CMPMODIFH</b>	比较状态修改寄存器, 高位字节	页 15-55
<b>TCTR0L</b>	定时器控制寄存器 0, 低位字节	页 15-56
<b>TCTR0H</b>	定时器控制寄存器 0, 高位字节	页 15-57
<b>TCTR2L</b>	定时器控制寄存器 2, 低位字节	页 15-59
<b>TCTR2H</b>	定时器控制寄存器 2, 高位字节	页 15-60
<b>TCTR4L</b>	定时器控制寄存器 4, 低位字节	页 15-61
<b>TCTR4H</b>	定时器控制寄存器 4, 高位字节	页 15-62
<b>调制控制寄存器</b>		
<b>MODCTRL</b>	调制控制寄存器, 低位字节	页 15-63
<b>MODCTRH</b>	调制控制寄存器, 高位字节	页 15-64
<b>TRPCTRL</b>	强制中断控制寄存器, 低位字节	页 15-65
<b>TRPCTRH</b>	强制中断控制寄存器, 高位字节	页 15-67
<b>PSLR</b>	被动态电平寄存器	页 15-68
<b>MCMOUTSL</b>	多通道模式输出映射寄存器, 低位字节	页 15-69
<b>MCMOUTSH</b>	多通道模式输出映射寄存器, 高位字节	页 15-69
<b>MCMOUTL</b>	多通道模式输出寄存器, 低位字节	页 15-70
<b>MCMOUTH</b>	多通道模式输出寄存器, 高位字节	页 15-72
<b>MCMCTR</b>	多通道模式控制寄存器	页 15-72
<b>T12MSELL</b>	定时器 T12 捕获 / 比较模式选择寄存器, 低位字节	页 15-40
<b>T12MSELH</b>	定时器 T12 捕获 / 比较模式选择寄存器, 高位字节	页 15-42
<b>中断控制寄存器</b>		
<b>ISL</b>	中断状态寄存器, 低位字节	页 15-73
<b>ISH</b>	中断状态寄存器, 高位字节	页 15-74
<b>ISSL</b>	中断状态置位寄存器, 低位字节	页 15-76
<b>ISSH</b>	中断状态置位寄存器, 高位字节	页 15-77
<b>ISRL</b>	中断状态复位寄存器, 低位字节	页 15-78
<b>ISRH</b>	中断状态复位寄存器, 高位字节	页 15-79

表 15-5 Registers Overview 寄存器概览

寄存器缩写名	寄存器全名	描述见
IENL	中断使能寄存器, 低位字节	页 15-80
IENH	中断使能寄存器, 高位字节	页 15-81
INPL	中断节点指针寄存器, 低位字节	页 15-83
INPH	中断节点指针寄存器, 高位字节	页 15-84

### 15.3.1 系统寄存器

寄存器 PISEL0 和 PISEL2 用来选择模块的实际输入信号, 使器件引脚功能可根据应用需求进行配置。输出引脚由端口寄存器选择。

#### PISEL0L

端口输入选择寄存器 0, 低位字节

复位值: 00<sub>H</sub>

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> 该位域定义了用作 CC60 捕获输入信号的端口引脚。 00 选择 CC60_0 输入引脚 01 保留 10 保留 11 选择 CC60_3 输入引脚。
ISCC61	3:2	rw	<b>CC61 的输入选择</b> 该位域定义了用作 CC61 捕获输入信号的端口引脚。 00 选择 CC61_0 输入引脚 01 选择 CC61_1 输入引脚 10 选择 CC61_2 输入引脚 11 选择 CC61_3 输入引脚
ISCC62	5:4	rw	<b>CC62 的输入选择</b> 该位域定义了用作 CC62 捕获输入信号的端口引脚。 00 选择 CC62_0 输入引脚 01 选择 CC62_1 输入引脚 10 保留 11 选择 CC62_3 输入引脚

符号	位	类型	描述
<b>ISTRP</b>	7:6	rw	<b>CTRAP 的输入引脚</b> 该位域定义了用作 <b>CTRAP</b> 输入信号的端口引脚。 00 选择 <b>CTRAP_0</b> 输入引脚 01 选择 <b>CTRAP_1</b> 输入引脚 10 选择 <b>CTRAP_2</b> 输入引脚 11 选择 <b>CTRAP_3</b> 输入引脚

## PISEL0H

端口输入选择寄存器 0，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>IST12HR</b>		<b>ISPOS2</b>		<b>ISPOS1</b>		<b>ISPOS0</b>	
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> 该位域定义了用作 <b>CCPOS0</b> 捕获输入信号的端口引脚。 00 选择 <b>CCPOS0_0</b> 输入引脚 01 选择 <b>CCPOS0_1</b> 输入引脚 10 选择 <b>CCPOS0_2</b> 输入引脚 11 选择 <b>CCPOS0_3</b> 输入引脚
<b>ISPOS1</b>	3:2	rw	<b>CCPOS1 的输入选择</b> 该位域定义了用作 <b>CCPOS1</b> 捕获输入信号的端口引脚 00 选择 <b>CCPOS1_0</b> 输入引脚 01 选择 <b>CCPOS1_1</b> 输入引脚 10 选择 <b>CCPOS1_2</b> 输入引脚 11 选择 <b>CCPOS1_3</b> 输入引脚
<b>ISPOS2</b>	5:4	rw	<b>CCPOS2 的输入选择</b> 该位域定义了用作 <b>CCPOS2</b> 捕获输入信号的端口引脚。 00 选择 <b>CCPOS2_0</b> 输入引脚 01 选择 <b>CCPOS2_1</b> 输入引脚 10 选择 <b>CCPOS2_2</b> 输入引脚 11 选择 <b>CCPOS2_3</b> 输入引脚

## 捕获 / 比较单元 6

符号	位	类型	描述
<b>IST12HR</b>	7:6	rw	<b>T12HR 的输入选择</b> 该位域定义了用作 T12HR 输入信号的端口引脚。 00 选择 T12HR_0 的输入引脚 01 选择 T12HR_1 的输入引脚 10 选择 T12HR_2 的输入引脚 11 选择 T12HR_3 的输入引脚

**PISEL2**
**端口输入选择寄存器 2**
**复位值: 00<sub>H</sub>**

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> 该位域定义了用作 T13HR 捕获输入信号的端口引脚。 00 选择 T13HR_0 的输入引脚 01 选择 T13HR_1 的输入引脚 10 选择 T13HR_2 的输入引脚 11 选择 T13HR_3 的输入引脚
<b>0</b>	7:2	r	<b>保留</b> 读操作返回 0；应写入 0。

### 15.3.2 定时器 T12 相关的寄存器

由定时器 T12 产生 3 路通道的 PWM 序列。定时器 T12 的相关寄存器（在定义好的条件下）同时刷新以保证三路 PWM 通道的一致性。

定时器 T12 支持捕获和比较模式，这两种模式可独立选择用于 3 路通道 CC60、CC61 和 CC62。

寄存器 T12MSEL 中包含用于选择定时器 T12 三路通道捕获/比较功能的控制位。表 15-6、表 15-7 和表 15-8 定义并解释可选的捕获/比较模式。请参考下面的寄存器描述选择合适的工作模式。

**表 15-6 双寄存器捕获模式**

描述
0100 在输入引脚 CC6n 的下降沿将 T12 的内容保存在 CC6nR 中；在输入引脚 CC6n 的下降沿将 T12 的内容保存在 CC6nSR 中。
0101 在输入引脚 CC6n 的上升沿将 CC6nSR 中的内容复制到 CC6nR 中；T12 的实际计数值立即保存在映射寄存器 CC6nSR 中。该特性可用于测量 CC6n 引脚上连续两个上升沿之间的时间。COUT6n 是 I/O 引脚。
0110 在输入引脚 CC6n 的下降沿将 CC6nSR 中的内容复制到 CC6nR 中；T12 的实际计数值立即保存在映射寄存器 CC6nSR 中。该特性可用于测量 CC6n 引脚上连续两个下降沿之间的时间。COUT6n 是 I/O 引脚。
0111 在输入引脚 CC6n 的任意沿将 CC6nSR 中的内容复制到 CC6nR 中；T12 的实际计数值立即保存在映射寄存器 CC6nSR 中。该特性可用于测量 CC6n 引脚上连续两个跳变沿之间的时间。COUT6n 是 I/O 引脚。

**表 15-7 组合 T12 模式**

描述
1000 霍尔传感器模式： 通道 0 为捕获模式，通道 1 和 2 为比较模式。检测到有效的霍尔事件，将 T12 的内容捕获到 CC60 中（作为实际转速的参考值）；CC61 可产生霍尔事件和输出切换之间的相位延迟；如果期望的霍尔事件出现得太晚，CC62 可作为超时触发。若使用霍尔信号，MSEL0，MSEL1 和 MSEL2 必须设置为 1000 <sub>B</sub> 。该模式下，在检测到有效的霍尔事件后定时器 T12 的内容被捕获至 CC60 并对 T12 进行复位；为了避免噪声影响，当检测到霍尔输入的跳变沿时，启动通道 0 的死区时间计数器，计数到 000001 <sub>B</sub> 时，采样霍尔输入并完成序列比较。
1001 带有死区时间的类磁滞控制模式： CCPOSx 输入信号的下降沿用来复位 CC6nST。因此输出信号立刻被切换到被动态，当 CCPOSx 信号为高且 CC6nST 由比较事件置位后，输出切换回主动态（带有死区时间）。

表 15-8      多输入捕获模式

描述	
1010	在输入引脚 CC6n 的上升沿将 T12 的定时器值保存在 CC6nR 中；在输入引脚 CCPOSx 的下降沿将 T12 的定时器值保存在 CC6nSR 中。
1011	在输入引脚 CC6n 的下降沿将 T12 的定时器值保存在 CC6nR 中；在输入引脚 CCPOSx 的上升沿将 T12 的定时器值保存在 CC6nSR 中。
1100	在输入引脚 CC6n 的上升沿将 T12 的定时器值保存在 CC6nR 中；在输入引脚 CCPOSx 的上升沿将 T12 的定时器值保存在 CC6nSR 中。
1101	在输入引脚 CC6n 的下降沿将 T12 的定时器值保存在 CC6nR 中；在输入引脚 CCPOSx 的下降沿将 T12 的定时器值保存在 CC6nSR 中。
1110	在输入引脚 CC6n 的任意沿将 T12 的定时器值保存在 CC6nR 中；在输入引脚 CCPOSx 的任意（升 / 降）沿将 T12 的定时器值保存在 CC6nSR 中。
1111	保留（无捕获或比较动作）

T12MSELL

T12 捕获 / 比较模式选择寄存器，低位字节 复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
MSEL61				MSEL60			
rw				rw			

符号	位	类型	描述
<b>MSEL60, MSEL61</b>	3:0, 7:4	rw	<p><b>捕获 / 比较模式选择</b></p> <p>该位域选择定时器 T12 三路捕获 / 比较通道的工作模式。根据以下定义，每路通道（n = 0, 1, 2）可独立设定为比较或捕获模式：</p> <p>0000 禁止比较输出；引脚 CC6n 和 COUT6n 可用作 I/O 引脚。无捕获操作。</p> <p>0001 比较结果从引脚 CC6n 输出，COUT6n 可用作 I/O 引脚。无捕获操作。</p> <p>0010 比较结果从引脚 COUT6n 输出，CC6n 可用作 I/O 引脚。无捕获操作。</p> <p>0011 比较结果从引脚 COUT6n 和 CC6n 输出。无捕获操作。</p> <p>01XX 双寄存器捕获模式，见表 15-6。</p> <p>1000 霍尔传感器模式，见表 15-7。 为了使能霍尔跳变沿检测，必须将所有三个 MSEL6x 设置为霍尔传感器模式。</p> <p>1001 类磁滞模式，见表 15-7。</p> <p>101X 多输入捕获模式，见表 15-8。</p> <p>11XX 多输入捕获模式，见表 15-8。</p>

**T12MSELH**
**T12 捕获 / 比较模式选择寄存器，高位字节**
**复位值：00<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>D BYP</b>	<b>HSYNC</b>			<b>MSEL62</b>			
<b>rw</b>	<b>rw</b>			<b>rw</b>			

符号	位	类型	描述
<b>MSEL62</b>	3:0	rw	<p><b>捕获 / 比较模式选择</b></p> <p>该位域选择定时器 T12 三路捕获 / 比较通道的工作模式。根据以下定义，每路通道（n = 0, 1, 2）可独立编程设定为比较或捕获模式：</p> <p>0000 比较输出被禁止；引脚 CC6n 和 COUT6n 可用作 I/O 引脚。无捕获操作。</p> <p>0001 比较结果从引脚 CC6n 输出，COUT6n 可用作 I/O 引脚。无捕获操作。</p> <p>0010 比较结果从引脚 COUT6n 输出，CC6n 可用作 I/O 引脚。无捕获操作。</p> <p>0011 比较结果从引脚 COUT6n 和 CC6n 输出。无捕获操作。</p> <p>01XX 双寄存器捕获模式，见表 15-6。</p> <p>1000 霍尔传感器模式，见表 15-7。 为了使能霍尔跳变沿检测，必须将 MSEL6x 全部设置为霍尔传感器模式。</p> <p>1001 类磁滞模式，见表 15-7。</p> <p>101X 多输入捕获模式，见表 15-8。</p> <p>11XX 多输入捕获模式，见表 15-8。</p>



符号	位	类型	描述
<b>HSYNC</b>	6:4	rw	<b>霍尔同步</b> 位域 HSYNC 定义了采样霍尔输入序列、并和当前及期望的霍尔序列位域进行比较的触发源。在所有模式下，可由软件对 SWHC 写入 1 触发采样。 000 任一输入引脚 CCPOSx (x = 0, 1, 2) 的任意升降沿触发采样。 001 T13 比较 - 匹配触发采样。 010 T13 周期 - 匹配触发采样。 011 关闭硬件触发霍尔采样。 100 T12 (递增计数时) 的周期 - 匹配触发采样。 101 T12 (递减计数时) 的 1- 匹配触发采样。 110 T12 通道 0 (递增计数时) 的比较 - 匹配触发采样。 111 T12 通道 0 (递减计数时) 的比较 - 匹配触发采样。
<b>DBYP</b>	7	rw	<b>延迟旁路</b> 位域 DBYP 决定采样霍尔输入序列的触发源 (由 HSYNC 选择) 是否利用定时器 T12 的死区时间计数器 DTC0 产生附加延迟; 或者延迟被旁路。 0 延迟旁路无效。在触发源信号有效之后死区时间计数器 DTC0 产生附加延迟。 1 延迟旁路有效。采样霍尔序列不用死区时间计数器 DTC0 产生延迟。

注：捕获模式下，CC6x 输入的所有跳变沿将置位寄存器 IS 中的相应中断状态标志。为了监控多输入捕获模式下 CCPOSx 输入端所选择的捕获事件，检测到选中事件时置位相应通道的 CC6xST。必须由软件复位中断状态位和 CC6xST。

寄存器 T12 代表定时器 T12 的计数值。定时器 T12 停止时，可写访问该寄存器。T12 运行期间，对该寄存器的写操作不予理睬。寄存器 T12 始终可由软件读取。

边沿对齐模式下，T12 递增计数。中间对齐模式下，T12 可递增递减计数。

### T12L

定时器 T12 计数寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
T12CVL							
rwh							

符号	位	类型	描述
<b>T12CVL</b>	7:0	rwh	定时器 <b>T12</b> 计数值，低位字节 该寄存器存放定时器 <b>T12</b> 低 8 位计数值。

### T12H

定时器 **T12** 计数寄存器，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>T12CVH</b>							
rwh							

符号	位	类型	描述
<b>T12CVH</b>	7:0	rwh	定时器 <b>T12</b> 的计数值，高位字节 该寄存器存放定时器 <b>T12</b> 的高 8 位计数值。

注：一旦定时器 **T12** 停止工作，内部时钟分频器被复位以确保可重复产生同样的时序和延迟。

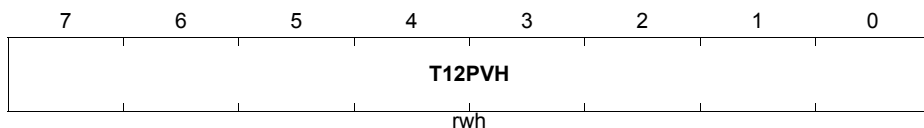
### T12PRL

定时器 **T12** 周期寄存器，低位字节

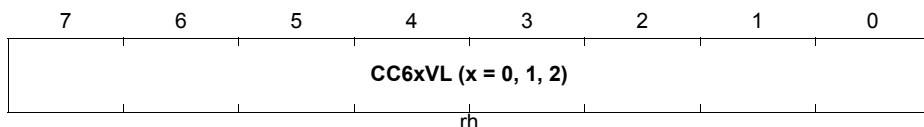
复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>T12PVL</b>							
rwh							

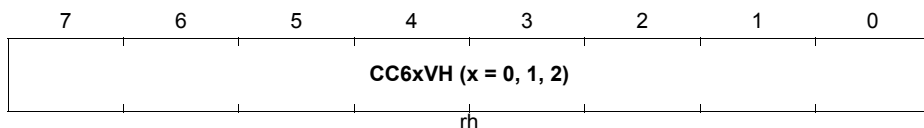
符号	位	类型	描述
<b>T12PVL</b>	7:0	rwh	定时器 <b>T12</b> 的周期值，低位字节 <b>T12PV</b> 定义了用来产生周期 - 匹配的 <b>T12</b> 的计数值。计数到该值后，定时器 <b>T12</b> 被置 0（边沿对齐模式）或改变其计数方向为递减计数（中间对齐模式）。

**T12PRH**
**定时器 T12 周期寄存器，高位字节**
**复位值：00<sub>H</sub>**


符号	位	类型	描述
<b>T12PVH</b>	7:0	rwh	<b>定时器 T12 周期值，高位字节</b> T12PV 定义了用来产生周期 - 匹配的 T12 的计数值。计数到该值后，定时器 T12 被置 0（边沿对齐模式）或改变其计数方向为递减计数（中间对齐模式）。

**CC6xRL (x = 0, 1, 2)**
**通道 CC6x 捕获 / 比较寄存器，低位字节**
**复位值：00<sub>H</sub>**


符号	位	类型	描述
<b>CC6xVL (x = 0, 1, 2)</b>	7:0	rh	<b>通道 x 捕获 / 比较值，低位字节</b> 比较模式下，位域 CC6xV 存放和 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, 1, 2)**
**通道 CC6x 捕获 / 比较寄存器，高位字节**
**复位值：00<sub>H</sub>**


**捕获 / 比较单元 6**

符号	位	类型	描述
<b>CC6xVH</b> <b>(x = 0, 1, 2)</b>	7:0	rh	<b>通道 x 捕获 / 比较值, 高位字节</b> 比较模式下, 位域 CC6xV 存放和 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。

**CC6xSRL (x = 0, 1, 2)**
**通道 CC6x 捕获 / 比较映射寄存器, 低位字节**
**复位值: 00<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>CC6xSL(x = 0, 1, 2)</b>							
rwh							

符号	位	类型	描述
<b>CC6xSL</b> <b>(x = 0, 1, 2)</b>	7:0	rwh	<b>通道 x 捕获 / 比较值, 低位字节</b> 比较模式下, 映射传送时位域 CC6xS 的内容被传送到位域 CC6xV 中; 捕获模式下, T12 的捕获值可从这些寄存器读出。

**CC6xSRH (x = 0, 1, 2)**
**通道 CC6x 捕获 / 比较映射寄存器, 高位字节**
**复位值: 00<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>CC6xSH (x = 0, 1, 2)</b>							
rwh							

符号	位	类型	描述
<b>CC6xSH</b> <b>(x = 0, 1, 2)</b>	7:0	rwh	<b>通道 x 捕获 / 比较值映射寄存器, 高位字节</b> 比较模式下, 映射传送时位域 CC6xS 的内容被传送到位域 CC6xV 中; 捕获模式下, T12 的捕获值可从这些寄存器读出。

### T12DTCL

定时器 T12 死区时间控制寄存器，低位字节

复位值：00<sub>H</sub>

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 位域 DTM 决定从被动态切换至主动态输出时的可编程延迟。主动态切换至被动态无延迟。

### T12DTCH

定时器 T12 死区时间控制寄存器，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	DTR2	DTR1	DTR0	0	DTE2	DTE1	DTE0
r	rh	rh	rh	r	rw	rw	rw

符号	位	类型	描述
DTE <sub>x</sub> (x = 0, 1, 2)	2:0	rw	死区时间使能位 位 DTE0...DTE2 使能和禁止定时器 T12 每路比较通道 (0, 1, 2) 死区时间产生。 0 禁止产生死区时间。(根据实际比较状态) 从被动态切换至主动态相应输出时无延迟。 1 使能产生死区时间。(根据实际比较状态) 从被动态切换至主动态相应输出时产生延迟，延迟值由位域 DTM 设定。
DTR <sub>x</sub> (x = 0, 1, 2)	6:4	rh	死区时间逻辑功能运行指示位 位 DTR0...DTR2 指示定时器 T12 每路比较通道 (0, 1, 2) 产生死区时间的状态。 0 相应通道死区时间计数器的值为 0 1 相应通道死区时间计数器的值非 0
0	3, 7	r	保留 读操作返回 0，应写入 0。

捕获 / 比较单元 6

注：死区时间计数器以和 T12 时钟相同的频率计数。该结构可在中间对齐和边沿对齐 PWM 模式下产生对称的死区时间。若 CC6x 的占空比为 50%，COUT6x 的开启时间为：0.5 \* 周期值 - 死区时间。

注：死区时间计数器由位 DTRES 复位（不由位 T12RES 复位）。

15.3.3 定时器 T13 相关寄存器

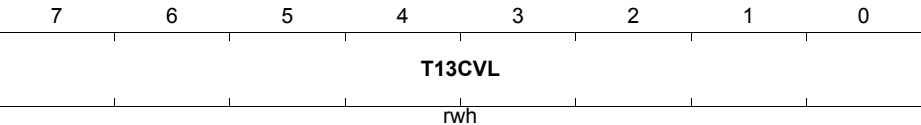
由定时器 T13 产生单路通道 PWM 序列。定时器 T13 的相关寄存器（在定义好的条件下）同时刷新以保证 PWM 信号的一致性。定时器 T13 可由定时器 T12 事件同步。

定时器 T13 仅支持（比较）通道 CC63 上的比较模式。

寄存器 T13 代表定时器 T13 的计数值。定时器 T13 停止时，可写访问该寄存器。T13 运行期间，对该寄存器的写操作不予理睬。寄存器 T13 始终可由软件读取。

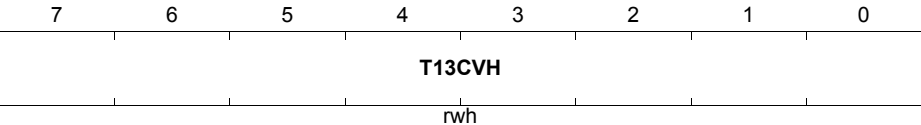
T13 仅支持边沿对齐模式（递增计数）。

**T13L**  
定时器 T13 计数寄存器，低位字节 复位值：00<sub>H</sub>



符号	位	类型	描述
T13CVL	7:0	rwh	定时器 T13 的计数值，低位字节 该寄存器代表定时器 T13 的低 8 位计数值。

**T13H**  
定时器 T13 计数寄存器，高位字节 复位值：00<sub>H</sub>



符号	位	类型	描述
<b>T13CVH</b>	7:0	rwh	定时器 <b>T13</b> 的计数值，高位字节 该寄存器代表定时器 <b>T13</b> 的高 8 位计数值。

注：一旦定时器 **T13** 停止工作，内部时钟分频被复位以确保可重复产生同样的时序和延迟。

### T13PRL

定时器 **T13** 周期寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>T13PVL</b>							
rwh							

符号	位	类型	描述
<b>T13PVL</b>	7:0	rwh	定时器 <b>T13</b> 的周期值，低位字节 <b>T13PV</b> 定义了用来产生周期 - 匹配的 <b>T13</b> 的计数值。 计数到该值后，定时器 <b>T13</b> 被置 0。

### T13PRH

定时器 **T13** 周期寄存器，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>T13PVH</b>							
rwh							

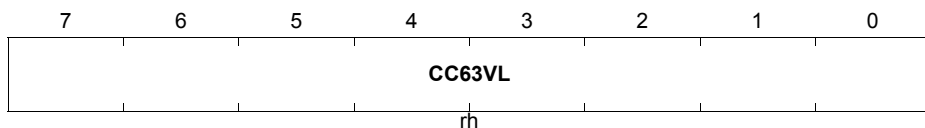
符号	位	类型	描述
<b>T13PVH</b>	7:0	rwh	定时器 <b>T13</b> 的周期值，高位字节 <b>T13PV</b> 定义了用来产生周期 - 匹配的 <b>T13</b> 的计数值。 计数到该值后，定时器 <b>T13</b> 被置 0。

捕获 / 比较单元 **6**

**CC63RL**

通道 **CC63** 捕获 / 比较寄存器，低位字节

复位值: **00<sub>H</sub>**

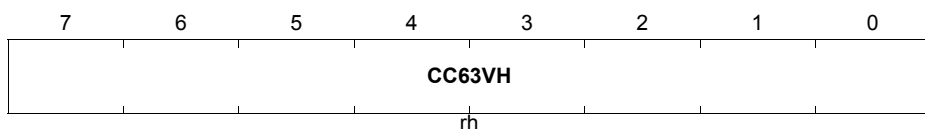


符号	位	类型	描述
<b>CC63VL</b>	7:0	rh	通道 <b>CC63</b> 的比较值，低位字节 位域 <b>CC63V</b> 存放和 <b>T13</b> 计数值进行比较的值。

**CC63RH**

通道 **CC63** 捕获 / 比较寄存器，高位字节

复位值: **00<sub>H</sub>**

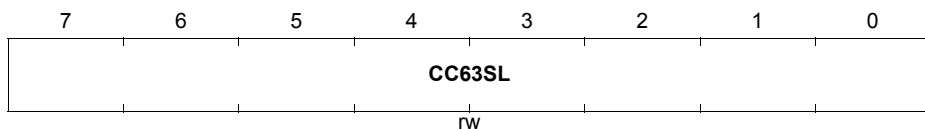


符号	位	类型	描述
<b>CC63VH</b>	7:0	rh	通道 <b>CC63</b> 的比较值，高位字节 位域 <b>CC63V</b> 存放和 <b>T13</b> 计数值进行比较的值。

**CC63SRL**

通道 **CC63** 捕获 / 比较映射寄存器，低位字节

复位值: **00<sub>H</sub>**





符号	位	类型	描述
<b>CC63SL</b>	7:0	rw	通道 <b>CC63</b> 比较值映射寄存器，低位字节 映射传送时位域 CC63S 的内容被传送到位域 CC63V 中。

### CC63SRH

通道 **CC63** 捕获 / 比较映射寄存器，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>CC63SH</b>							
rw							

符号	位	类型	描述
<b>CC63SH</b>	7:0	rw	通道 <b>CC63</b> 比较值映射寄存器，高位字节 映射传送时位域 CC63S 的内容被传送到位域 CC63V 中。

### 15.3.4 捕获 / 比较控制寄存器

寄存器 **CMPSTAT** 中包含监控当前捕获和比较状态的状态位；以及决定比较通道主动态/被动态的控制位。

### CMPSTATL

比较状态寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	<b>CC 63ST</b>	<b>CC POS 2</b>	<b>CC POS 1</b>	<b>CC POS 0</b>	<b>CC 62ST</b>	<b>CC 61ST</b>	<b>CC 60ST</b>
r	rh	rh	rh	rh	rh	rh	rh

**捕获 / 比较单元 6**

符号	位	类型	描述
<b>CC6xST</b> (x = 0, 1, 2, 3)	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 和 T12 相关；CC63ST 和 T13 相关。 0 比较模式下，定时器计数值小于比较值；捕获模式下，从软件复位该位后还未检测到所选择的跳变沿。 1 比较模式下，定时器计数值大于或等于比较值；捕获模式下，检测到选择的跳变沿。 这些位根据 T12 和 T13 的切换规则置位和复位。
<b>CCPOSx</b> (x = 0, 1, 2)	3, 4, 5	rh	<b>霍尔序列采样值</b> 位 CCPOSx 指示和当前以及期望的霍尔序列进行比较的输入霍尔序列的采样值。事件 hcrdy（霍尔比较准备就绪）发生时，进行霍尔采样。 0 输入 CCPOSx 的采样值为 0 1 输入 CCPOSx 的采样值为 1
<b>0</b>	7	r	<b>保留</b> 读操作返回 0；应写入 0。

**CMPSTATH**

比较状态寄存器，高位字节

 复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>T13</b> <b>IM</b>	<b>C</b> <b>OUT63PS</b>	<b>C</b> <b>OUT62PS</b>	<b>CC</b> <b>62PS</b>	<b>C</b> <b>OUT61PS</b>	<b>CC</b> <b>61PS</b>	<b>C</b> <b>OUT60PS</b>	<b>CC</b> <b>60PS</b>
rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh

符号	位	类型	描述
<b>CC6xPS</b> <b>(x = 0, 1, 2)</b>	0, 2, 4	rwh	<b>比较输出的被动态选择</b> 位 CC6xPS 和 COUT6xPS 选择相应比较通道的被动态。被动态时输出引脚驱动被动电平（由寄存器 PSLR 定义的）。位 CC6xPS 和 COUT6xPS（x = 0, 1, 2）和 T12 相关，位 COUT63PS 和 T13 相关。 0      CC6xST 为 0 时相应的比较输出驱动被动电平 1      CC6xST 为 1 时相应的比较输出驱动被动电平 这些位具有映射位，与 T12 和 T13 捕获 / 比较寄存器同时更新。读操作读取实际使用位的值，写操作写入映射位。 捕获模式下不使用这些位。
<b>COUT6xPS</b> <b>(x = 0, 1, 2, 3)</b>	1, 3, 5, 6		
<b>T13IM</b>	7	rwh	<b>T13 反相调制</b> 位 T13IM 将 T13 信号反相用于调制 CC6x 和 COUT6x（x = 0, 1, 2）信号。 0      T13 输出不反相 1      T13 输出反相用于进一步调制 该位具有映射位，与 T13 的比较和周期寄存器同时更新。读操作读取实际使用位的值，写操作写入映射位。

寄存器 CMPMODIF 中包含允许软件修改捕获 / 比较状态的控制位。

CMPMODIFL

比较状态修改寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	MCC 63S		0		MCC 62S	MCC 61S	MCC 60S
r	w		r		w	w	w

符号	位	类型	描述
<b>MCC6xS</b> <b>(x = 0, 1, 2, 3)</b>	0, 1, 2, 6	w	<b>捕获 / 比较状态修改位 (置位)</b> 用软件置位相应的 CC6xST。 例如在相应比较定时器停止工作的情况下，该特性允许用户软件编程分别修改输出线的状态。通过一条数据写指令即可实现 CC6xST 的位操作。 下面给出写访问修改捕获 / 比较状态位时，位 CC6xST 的状态： MCC6xR, MCC6xS = 0,0 未改变 CC6xST 0,1 置位 CC6xST 1,0 复位 CC6xST 1,1 保留 (翻转)
<b>0</b>	5:3,7	r	<b>保留</b> 读操作返回 0；应写入 0。

# CMPMODIFH

比较状态修改寄存器，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	MCC 63R		0		MCC 62R	MCC 61R	MCC 60R
r	w		r		w	w	w

符号	位	类型	描述
MCC6xR (x = 0, 1, 2, 3)	0, 1, 2, 6	w	<p>捕获 / 比较状态修改位（复位）</p> <p>用软件复位相应的 CC6xST。</p> <p>例如在相应比较定时器停止工作的情况下，该特性允许用户软件编程分别修改输出线的状态。通过一条数据写指令即可实现 CC6xST 的位操作。</p> <p>下面给出写访问修改捕获 / 比较状态位时，位 CC6xST 的状态：</p> <p>MCC6xR, MCC6xS =</p> <p>0,0 未改变 CC6xST</p> <p>0,1 置位 CC6xST</p> <p>1,0 复位 CC6xST</p> <p>1,1 保留（翻转）</p>
0	5:3,7	r	<p>保留</p> <p>读操作返回 0；应写入 0。</p>

寄存器 TCTR0 控制定时器 T12 和 T13 的基本功能。

**TCTR0L**

定时器控制寄存器 0，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>CTM</b>	<b>CDIR</b>	<b>STE12</b>	<b>T12R</b>	<b>T12 PRE</b>	<b>T12CLK</b>		
rw	rh	rh	rh	rw	rw		

符号	位	类型	描述
<b>T12CLK</b>	2:0	rw	<b>定时器 T12 输入时钟选择</b> 根据等式 $f_{T12} = f_{CCU} / 2^{<T12CLK>}$ ，选择来自外设时钟的定时器 T12 的输入时钟。 000 $f_{T12} = f_{CCU}$ 001 $f_{T12} = f_{CCU} / 2$ 010 $f_{T12} = f_{CCU} / 4$ 011 $f_{T12} = f_{CCU} / 8$ 100 $f_{T12} = f_{CCU} / 16$ 101 $f_{T12} = f_{CCU} / 32$ 110 $f_{T12} = f_{CCU} / 64$ 111 $f_{T12} = f_{CCU} / 128$
<b>T12PRE</b>	3	rw	<b>定时器 T12 预分频控制位</b> 为了支持更高的时钟频率，允许额外的分频因子 1/256 用于 T12 分频。 0 禁止 T12 的额外分频 1 使能 T12 的额外分频
<b>T12R</b>	4	rh	<b>定时器 T12 运行控制位</b> T12R 启动和停止定时器 T12 工作。软件通过置位 T12RS 或 T12RR 来置位 / 复位 T12R；或根据 T12SSC 定义的功能硬件复位 T12R。 0 定时器 T12 停止 1 定时器 T12 在运行 （由 T12SSC，T12RR 或 T12RS）同时置位 / 复位 T12R 将不起作用，T12R 保持不变。

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 使能或禁止将 T12 的周期值、比较值、被动态选择位和被动态电平从映射寄存器中映射传送到实际寄存器中。映射传送后硬件对 STE12 清零。 递增计数时 T12 的映射传送事件为周期 - 匹配；递减计数时为 1 - 匹配。 0 禁止映射寄存器传送 1 使能映射寄存器传送
CDIR	6	rh	<b>定时器 T12 的计数方向</b> 根据 T12 的计数规则对该位置位或复位 0 T12 递增计数 1 T12 递减计数
CTM	7	rw	<b>T12 工作模式</b> 0 边沿对齐模式： T12 始终递增计数，计数至周期值后从 0 开始重新继续计数。 1 中间对齐模式： 检测到周期 - 匹配时 T12 递减计数，检测到 1 - 匹配时 T12 递增计数

TCTR0H

定时器控制寄存器 0，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	STE 13	T13R	T13 PRE	T13CLK			
r	rh	rh	rw	rw			

符号	位	类型	描述
<b>T13CLK</b>	2:0	rw	<p><b>定时器 T13 输入时钟选择</b>  根据等式 <math>f_{T13} = f_{CCU} / 2^{&lt;T13CLK&gt;}</math>，选择来自外设时钟的定时器 T13 的输入时钟。</p> <p>000 <math>f_{T13} = f_{CCU}</math>  001 <math>f_{T13} = f_{CCU} / 2</math>  010 <math>f_{T13} = f_{CCU} / 4</math>  011 <math>f_{T13} = f_{CCU} / 8</math>  100 <math>f_{T13} = f_{CCU} / 16</math>  101 <math>f_{T13} = f_{CCU} / 32</math>  110 <math>f_{T13} = f_{CCU} / 64</math>  111 <math>f_{T13} = f_{CCU} / 128</math></p>
<b>T13PRE</b>	3	rw	<p><b>定时器 T13 预分频控制位</b>  为了支持更高的时钟频率，允许额外的分频因子 1/256 用于 T13 分频。</p> <p>0 禁止 T13 的额外分频  1 使能 T13 的额外分频</p>
<b>T13R</b>	4	rh	<p><b>定时器 T13 运行控制位</b>  T13R 启动和停止定时器 T13 工作。软件通过置位 T13RS 或 T13RR 来置位 / 复位 T13R；或根据 T13SSC、T13TEC 和 T13TED 定义的功能硬件置位 / 复位 T13R。</p> <p>0 定时器 T13 停止工作  1 定时器 T13 在运行  （由 T13SSC，T13TEC，T13RR 或 T13RS）同时置位 / 复位 T13R 将不起作用，T13R 保持不变。</p>
<b>STE13</b>	5	rh	<p><b>定时器 T13 映射传送使能</b>  如果检测到 T13 映射传送事件时，位 STE13 使能或禁止将 T13 的周期值、比较值、被动态选择位和被动态电平从映射寄存器中映射传送到实际寄存器中。映射传送后硬件对 STE13 清零。  T13 的映射传送事件为周期 - 匹配。</p> <p>0 禁止映射寄存器传送  1 使能映射寄存器传送</p>
<b>0</b>	7:6	r	<p><b>保留</b>  读操作返回 0；应写入 0。</p>



## 捕获 / 比较单元 6

注：只有当定时器 T12 不工作时 ( $T12R = 0$ )，对位域 T12CLK 或位 T12PRE 的写入才有效。只有当定时器 T13 不工作时 ( $T13R = 0$ )，对位域 T13CLK 或位 T13PRE 的写入才有效。

寄存器 TCTR2 控制定时器 T12 和 T13 的单次模式和同步功能。两个定时器均可工作在单次模式；该模式下，定时器在一个计数周期之后自动停止计数，此时计数值为 0。单次模式和 T13 与 T12 同步相结合，可在定义好的 T12 的 PWM 事件之后产生延迟时间可编程的事件。例如，该特性可在规定的延迟之后（以避免切换噪声引起的问题）、用来触发与某个 PWM 事件同步的模数转换操作。

### TCTR2L

定时器控制寄存器 2，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	T13 TED		T13 TEC			T13 SSC	T12 SSC
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 禁止单次模式，对 T12R 无硬件操作。 1 使能单次模式，在下列情况下硬件复位 T12R： 在边沿对齐模式下，T12 计数至周期值。在中间对齐模式下，T12 递减计数时计数至 1 复位 T12R 的同时复位 CC6xST (x = 0, 1, 2)。
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 对 T13R 无硬件操作 1 使能单次模式，T13 计数至周期值后硬件置位 T13R 复位 T13R 的同时复位 CC63ST。

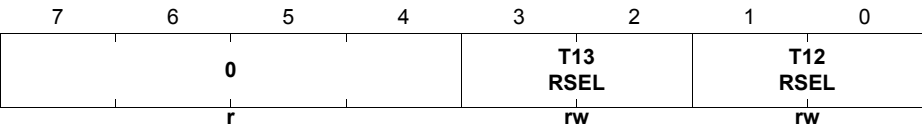
符号	位	类型	描述
T13TEC	4:2	rw	<b>T13 触发事件控制</b> 根据下列组合，位域 T13TEC 选择启动 T13 的触发事件（自动置位 T13R 和 T12 的比较信号同步）： 000 无操作 001 通道 0 的 T12 比较事件触发置位 T13R 010 通道 1 的 T12 比较事件触发置位 T13R 011 通道 2 的 T12 比较事件触发置位 T13R 100 通道 0, 1 或 2 的任意 T12 比较事件触发置位 T13R 101 T12 的周期 - 匹配置位 T13R T12 110 T12 的 0- 匹配（递增计数时）置位 T13R 111 输入 CCPOSx 的任意边沿置位 T13R
T13TED	6:5	rw	<b>T13 触发事件方向</b> 检测到由 T13TEC 定义的触发事件，位域 T13TED 给出控制自动置位 T13R 的附加信息。 00 无操作 01 T12 递增计数时 10 T12 递减计数时 11 和 T12 的计数方向无关
0	7	r	<b>保留</b> 读操作返回 0；应写入 0。

举例说明：  
如果 T12 的任意比较事件将启动定时器 T13（T13TEC = 100<sub>B</sub>），触发事件的方向可设定为：  
- 递增计数 >> 只有当 T12 递增计数时，T12 通道 0, 1, 2 的比较 - 匹配触发 T13R  
- 递减计数 >> 只有当 T12 递减计数时，T12 通道 0, 1, 2 的比较 - 匹配触发 T13R  
- 和位 CDIR 无关 >> 每次 T12 通道 0, 1, 2 比较匹配触发 T13R  
定时器的计数方向取自位 CDIR 的值。因此，如果 T12 工作在边沿模式（只递增计数），只有当位域 T13TED = 01<sub>B</sub> 或 11<sub>B</sub> 时 T13 可自动被启动。

TCTR2H

定时器控制寄存器 2，高位字节

复位值：00<sub>H</sub>



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 定义了信号 T12HR 的事件，可硬件置位运行位 T12R。 00 禁止由外部事件设置 T12R 01 检测到 T12HR 的上升沿时置位 T12R 10 检测到 T12HR 的下降沿时置位 T12R 11 检测到 T12HR 的任意沿时置位 T12R
<b>T13RSEL</b>	3:2	rw	<b>定时器 T13 外部运行选择</b> 位域 T13RSEL 定义了信号 T13HR 的事件，可硬件置位运行位 T13R。 00 禁止由外部事件设置 T13R 01 检测到 T13HR 的上升沿时置位 T13R 10 检测到 T13HR 的下降沿时置位 T13R 11 检测到 T13HR 的任意沿时置位 T13R
<b>0</b>	7:4	r	<b>保留</b> 读操作返回 0；应写入 0。

寄存器 TCTR4 通过独立的置位和复位条件，软件控制运行位 T12R 和 T13R。此外，定时器（运行中时）可被复位，位 STE12 和 STE13 可由软件控制。

#### TCTR4L

定时器控制寄存器 4，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>T12 STD</b>	<b>T12 STR</b>	<b>0</b>	<b>DT RES</b>	<b>T12 RES</b>	<b>T12 RS</b>	<b>T12 RR</b>	
<b>w</b>	<b>w</b>	<b>r</b>	<b>w</b>	<b>w</b>	<b>w</b>	<b>w</b>	

符号	位	类型	描述
<b>T12RR</b>	0	w	<b>定时器 T12 运行复位</b> 对该位置位将复位 T12R。 0 T12R 不受影响 1 T12R 被清零，T12 停止计数
<b>T12RS</b>	1	w	<b>定时器 T12 运行置位</b> 置位该位将置位 T12R。 0 T12R 不受影响 1 T12R 被置位，T12 计数

## 捕获 / 比较单元 6

符号	位	类型	描述
<b>T12RES</b>	2	w	<b>定时器 T12 复位</b> 0 对 T12 无影响 1 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 对 T12R 无影响。
<b>DTRES</b>	3	w	<b>死区时间计数器复位</b> 0 对死区时间计数器无影响 1 三路死区事件计数通道复位为 0
<b>T12STR</b>	6	w	<b>定时器 T12 映射传送请求</b> 0 无操作 1 STE12 被置位，允许映射传送
<b>T12STD</b>	7	w	<b>定时器 T12 映射传送禁止</b> 0 无操作 1 STE12 被复位，不能触发映射传送
<b>0</b>	5:4	r	<b>保留</b> 读操作返回 0；应写入 0。

**TCTR4H**

定时器控制寄存器 4，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>T13 STD</b>	<b>T13 STR</b>		<b>0</b>		<b>T13 RES</b>	<b>T13 RS</b>	<b>T13 RR</b>
w	w		r		w	w	w

符号	位	类型	描述
<b>T13RR</b>	0	w	<b>定时器 T13 运行复位</b> 对该位置位将复位 T13R。 0 T13R 不受影响 1 T13R 被清零，T13 停止计数
<b>T13RS</b>	1	w	<b>定时器 T13 运行置位</b> 对该位置位将置位 T13R。 0 T13R 不受影响 1 T13R 被置位，T13 计数
<b>T13RES</b>	2	w	<b>定时器 T13 复位</b> 0 对 T13 无影响 1 T13 计数寄存器被复位为 0。根据切换规则切换输出信号。复位 T13RES 对 T13R 无影响。

符号	位	类型	描述
<b>T13STR</b>	6	w	定时器 T13 映射传送请求 0 无操作 1 STE13 被置位，允许映射传送
<b>T13STD</b>	7	w	定时器 T13 映射传送禁止 0 无操作 1 没有触发映射传送，STE13 被复位。
<b>0</b>	5:3	r	保留 读操作返回 0；应写入 0。

注：同时向同一位的复位和置位控制位写 1 将不起作用，相应位将保持不变。

### 15.3.5 全局调制控制寄存器

寄存器 MODCTR 中的控制位使能由定时器 T12 和 T13 产生的 PWM 序列调制相应的输出信号。此外，可使能多通道模式用作输出信号的附加调制源。

#### MODCTRL

调制控制寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>MCMEN</b>	<b>0</b>	<b>T12MODEN</b>					
<b>rw</b>	<b>r</b>	<b>rw</b>					

符号	位	类型	描述
<b>T12MODEN</b>	5:0	rw	<b>T12 调制使能</b> 置位这些位将使能由 T12 产生的 PWM 序列调制相应的比较通道。这些位和下列输出信号对应： 位 0 CC60 的调制 位 1 COUT60 的调制 位 2 CC61 的调制 位 3 COUT61 的调制 位 4 CC62 的调制 位 5 COUT62 的调制 调制使能特性定义如下： 0 禁止由 T12 产生的 PWM 序列调制相应的输出信号 1 使能由 T12 产生的 PWM 序列调制相应的输出信号

## 捕获 / 比较单元 6

符号	位	类型	描述
<b>MCMEN</b>	7	rw	<b>多通道模式使能</b> 0 禁止根据位域 MCMP 产生的多通道序列调制相应的输出信号 1 使能根据位域 MCMP 产生的多通道序列调制相应的输出信号
<b>0</b>	6	r	<b>保留</b> 读操作返回 0，应写入 0。

**MODCTRH**

调制控制寄存器，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>ECT 130</b>	<b>0</b>	<b>T13MODEN</b>					
rw	r	rw					

符号	位	类型	描述
<b>T13MODEN</b>	5:0	rw	<b>T13 调制使能</b> 置位这些位将使能由 T13 产生的 PWM 序列调制相应的比较通道。这些位和下列输出信号对应： 位 0 CC60 的调制 位 1 COUT60 的调制 位 2 CC61 的调制 位 3 COUT61 的调制 位 4 CC62 的调制 位 5 COUT62 的调制 调制使能特性定义如下： 0 禁止由 T13 产生的 PWM 序列调制相应的输出信号 1 使能由 T13 产生的 PWM 序列调制相应的输出信号
<b>ECT130</b>	7	rw	<b>定时器 T13 比较输出使能</b> 0 引脚复用功能 COUT63 被禁止 1 引脚复用功能 COUT63 被使能，用于输出 T13 产生的 PWM 信号。
<b>0</b>	6	r	<b>保留</b> 读操作返回 0；应写入 0。

## 捕获 / 比较单元 6

寄存器 TRPCTR 控制强制中断功能。TRPCTR 中包含每路输出信号独立的使能位以及强制中断条件下选择不同操作的控制位。强制中断条件指输入引脚 CTRAP 为低电平，由（寄存器 IS 中的）位 TRPF 监控（反相电平）。当 TRPF = 1（强制中断输入有效）时，（寄存器 IS 中的）强制中断状态位 TRPS 被置 1。

### TRPCTRL

强制中断控制寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
		0			TRP M2	TRP M1	TRP M0
		r			rw	rw	rw

符号	位	类型	描述
TRPM0, TRPM1	1:0	rw	<p><b>强制中断模式控制位 1, 0</b></p> <p>这两位定义了强制中断条件再次无效时，输出信号如何退出强制中断状态。</p> <p>退出强制中断状态时，和驱动 PWM 的定时器同步可避免不需要的短脉冲。TRPM0 和 TRPM1 的组合引发如下操作：</p> <p>00 当检测到 T12（递增计数）的 0 - 匹配时退出强制中断状态（根据 TRPM2 返回正常操作）（和 T12 同步）</p> <p>01 当检测到 T13 的 0- 匹配时退出强制中断状态（根据 TRPM2 返回正常操作）（和 T13 同步）</p> <p>10 保留</p> <p>11 不和 T12 或 T13 同步，立刻退出强制中断状态（根据 TRPM2 返回正常操作）</p>
TRPM2	2	rw	<p><b>强制中断模式控制位 2</b></p> <p>0 一旦输入 CTRAP 变为无效，立刻退出强制中断状态（返回正常操作 = 位 TRPS = 0）。如果输入引脚 CTRAP 变为 1，位 TRPF 由硬件自动清零。如果 TRPF 为 0 并且检测到（由 TRPM0 和 TRPM1 决定的）同步条件时，位 TRPS 由硬件自动清零。</p> <p>1 一旦输入 CTRAP 变为无效，软件复位 TRPF（TRPF 不被硬件清零）立刻退出强制中断状态（返回正常操作 = TRPF = 0）。如果 TRPF 为 0 并且检测到（由 TRPM0 和 TRPM1 决定的）同步条件时，位 TRPS 由硬件自动清零。</p>

## 捕获 / 比较单元 6

符号	位	类型	描述
0	7:3	r	保留 读操作返回 0；应写入 0。



**TRPCTRH**

强制中断控制寄存器，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>TRP PEN</b>	<b>TRP EN 13</b>	<b>TRPEN</b>					
rw	rw	rw					

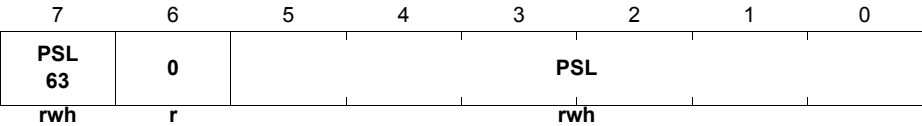
符号	位	类型	描述
<b>TRPEN</b>	5:0	rw	<b>强制中断使能控制</b> 置位这些位将使能下列对应输出信号的强制中断功能： 位 0 CC60 的强制中断功能 位 1 COUT60 的强制中断功能 位 2 CC61 的强制中断功能 位 3 COUT61 的强制中断功能 位 4 CC62 的强制中断功能 位 5 COUT62 的强制中断功能 强制中断功能的使能特性定义如下： 0 禁止相应输出信号的强制中断功能。输出状态和位 TRPS 无关。 1 使能相应输出信号的强制中断功能。位 TRPS = 1 时，输出信号被设置为被动态。
<b>TRPEN13</b>	6	rw	<b>定时器 T13 的强制中断使能控制</b> 0 禁止 T13 的强制中断功能。即使当 TRPS = 1，（如果选择 T13 且被使能）定时器 T13 仍产生 PWM。 1 使能 T13 的强制中断功能。位 TRPS = 1 时定时器 T13 的 PWM 输出信号被设置为被动态。
<b>TRPPEN</b>	7	rw	<b>强制中断引脚使能</b> 0 禁止基于输入引脚 <u>CTR</u> AP 的强制中断功能。只能软件置位 TRPF 产生强制中断。 1 使能基于输入引脚 <u>CTR</u> AP 的强制中断功能。软件置位 TRPF 或由 <u>CTR</u> AP = 0 产生强制中断。

寄存器 PSLR 定义了模块输出引脚驱动的被动态电平。输出端口处于被动态时，驱动的输出引脚电平值为被动态电平值。输出引脚处于主动态时，驱动与被动态电平反相的主动态电平。被动态电平可使被驱动的输出电平和所连接的功率级的驱动极性（反相或不反相）相匹配。

PSLR

被动态电平寄存器

复位值：00<sub>H</sub>



符号	位	类型	描述
PSL	5:0	rwh	<b>比较输出的被动态电平</b> 该位域定义了模块输出为被动态时，所驱动的被动态电平值。 位 0 输出 CC60 的被动态电平 位 1 输出 COUT60 的被动态电平 位 2 输出 CC61 的被动态电平 位 3 输出 COUT61 的被动态电平 位 4 输出 CC62 的被动态电平 位 5 输出 COUT62 的被动态电平 电平值定义如下： 0 被动态电平为 0 1 被动态电平为 1
PSL63	7	rwh	<b>输出 COUT63 的被动态电平</b> 该位定义了输出引脚 COUT63 的被动态电平值 0 被动态电平为 0 1 被动态电平为 1
0	6	r	<b>保留</b> 读操作返回 0；应写入 0。

注： 位域 **PSL** 具有映射寄存器，从而在更新输出电平值时不会产生不期望的脉冲。在 **T12** 映射传送时更新该位域。读操作读取实际使用位的值，写操作写入映射位。

注： 位域 **PSL63** 具有映射寄存器，从而在更新输出电平值时不会产生不期望的脉冲。在 **T13** 映射传送时更新该位。读操作读取实际使用位的值，写操作写入映射位。

15.3.6 多通道调制控制寄存器

寄存器 **MCMOUTS** 中包含多通道模式输出状态的控制位。此外，可选择合适的信号用于由霍尔传感器控制的块切换。该寄存器为寄存器 **MCMOUT** 的映射寄存器（可写），指示当前的有效信号。

## MCMOUTSL

多通道模式输出映射寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
STR MCM	0	MCMPS					
w	r	rw					

符号	位	类型	描述
MCMPS	5:0	rw	多通道 PWM 序列映射 MCMPS 为位域 MCMP 的映射位域。由寄存器 MCMCTR 定义的传送条件触发多通道映射传送。
STRMCM	7	w	MCMP 的映射传送请求 该位置位将立刻用写入位域 MCMPS 的值更新位域 MCMP。该功能允许用软件触发更新 MCMP。读取该位时始终为 0。 0 由定义的硬件动作更新位域 MCMP。对位域 MCMPS 的写操作不会修改位域 MCMP 的值。 1 由位域 MCMPS 的写入值更新位域 MCMP。
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，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
STR HP	0	CURHS			EXPHS		
w	r	rw			rw		

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。当检测到霍尔输入引脚 CCPOS <sub>x</sub> (x = 0, 1, 2) 的跳变沿时，将该位域内容传送到位域 EXPH 中。

符号	位	类型	描述
<b>CURHS</b>	5:3	rw	<b>当前霍尔序列映射</b> 位域 CURHS 是位域 CURH 的映射位域。当检测到霍尔输入引脚 CCPOSx (x = 0, 1, 2) 的跳变沿时, 将该位域内容传送到位域 CURH 中。
<b>STRHP</b>	7	w	<b>霍尔序列映射传送请求</b> 该位置位将立刻用写入位域 CURHS 和 EXPHS 中的值更新位域 CURH 和 EXPH。该功能允许用软件触发更新。读取该位时始终为 0。 0 根据定义的硬件动作更新位域 CURH 和 EXPH。对位域 CURHS 和 EXPHS 的写操作不会修改位域 CURH 和 EXPH 的值。 1 位域 CURH 和 EXPH 由写入位域 CURHS 和 EXPHS 的值更新。
<b>0</b>	6	r	<b>保留</b> 读操作返回 0, 应写入 0。 .

寄存器 MCMOUT 规定了当前所使用的多通道控制位。

## MCMOUTL

多通道模式输出寄存器, 低位字节

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
0	R	MCMP					
r	rh	rh					

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 的值通过映射传送从位域 MCMPS 写入。MCMP 中存放着多通道模式的输出序列。如果寄存器 MODCTR 中的 MCMEN 使能该模式，下列输出信号的输出状态可被修改：</p> <p>位 0 输出 CC60 的多通道状态</p> <p>位 1 输出 COUT60 的多通道状态</p> <p>位 2 输出 CC61 的多通道状态</p> <p>位 3 输出 COUT61 的多通道状态</p> <p>位 4 输出 CC62 的多通道状态</p> <p>位 5 输出 COUT62 的多通道状态</p> <p>多通道序列可将相关的输出设置为被动态。</p> <p>0 输出为被动态。T12 或 T13 产生的 PWM 不起调制作用。</p> <p>1 可输出 T12 或 T13 产生的 PWM 序列（根据寄存器 MODCTR）</p> <p>IDLE = 1 时，位域 MCMP 被清零。</p>
<b>R</b>	6	rh	<p><b>提示标志</b></p> <p>该提示标志指明所选择的触发源已请求从位域 MCMPS 到 MCMP 的映射传送。映射传送后且 MCMEN = 0，该位被清零。</p> <p>0 未发生从 MCMPS 到 MCMP 的映射传送请求。</p> <p>1 所选的触发源已请求从 MCMPS 到 MCMP 的映射传送，但由于所选的同步条件还未满足，还未执行映射传送。</p>
<b>0</b>	7	r	<p><b>保留</b></p> <p>读操作返回 0；应写入 0。</p>

## MCMOUTH

多通道模式输出控制器，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0		CURH			EXPH		
r		rh			rh		

符号	位	类型	描述
EXPH	2:0	rh	<b>期望霍尔序列</b> EXPH 的值通过映射传送从位域 EXPHS 写入。每次检测到霍尔输入引脚跳变之后都进行比较，从而检测是否出现了下次期望的霍尔序列或者错误序列。如果霍尔输入引脚的当前霍尔序列和位域 EXPH 相等，置位 CHE（正确霍尔事件）并（当位 ENCHE 使能中断时）产生中断请求。如果霍尔输入引脚的当前霍尔序列和位域 CURH 和 EXPH 都不相等，置位 WHE（错误霍尔事件）并（当位 ENWHE 使能中断时）产生中断请求。
CURH	5:3	rh	<b>当前霍尔序列</b> 位域 CURH 的值通过映射传送从位域 CURHS 写入。每次检测到霍尔输入引脚跳变之后进行比较，从而检测是否出现了下次期望的霍尔序列或者错误序列。如果当前的霍尔输入序列和位域 CURH 相等，霍尔输入引脚上检测到的跳变无效（如毛刺）。
0	7:6	r	<b>保留</b> 读操作返回 0；应写入 0。

注：寄存器 EXPH 和 CURH 中的位域和输入引脚 CCPOS<sub>x</sub>（ $x = 0, 1, 2$ ）上的霍尔序列按照下面次序对应：（EXPH.2, EXPH.1, EXPH.0），（CURH.2, CURH.1, CURH.0），（CCPOS2, CCPOS1, CCPOS0）。

寄存器 MCMCTR 中包含多通道模式的控制位。

## MCMCTR

多通道模式控制寄存器

复位值：00<sub>H</sub>

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 从以下的触发请求源中选择一个（下次多通道事件），触发从 MCMP5 到 MCMP 的映射传送。触发请求保存在提示标志 R 中，直到映射传送完成；映射传送时标志 R 被自动清零。映射传送和位域 SWSYN 所选择的事件同步发生。 000 不产生触发请求 001 检测到 CCPOSx 上的正确霍尔事件 010 检测到 T13 周期 - 匹配（递增计数） 011 检测到 T12 1 - 匹配（递减计数） 100 检测到 T12 通道 1 比较 - 匹配（相位延迟功能） 101 检测到 T12 周期 - 匹配（递增计数）；否则保留，不产生触发请求
SWSYN	5:4	rw	<b>切换同步</b> 位域 SWSYN 触发先前已被请求的从 MCMP5 到 MCMP 的映射传送（标志 R 由 SWSEL 所选择的事件置位）。该特性可使输出和（T12 或 T13 产生的）PWM 同步。 00 由映射请求触发事件直接引起映射传送 01 T13 0 - 匹配触发映射传送 10 T12 0 - 匹配（递增计数）触发映射传送 11 保留，无操作
0	3, 6, 7	r	<b>保留</b> 读操作返回 0，应写入 0。

注： 只有位 MCMEN = 1 时，才能使能由硬件产生映射传送请求。

15.3.7 中断控制寄存器

ISL

捕获 / 比较中断状态寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
T12 PM	T12 OM	ICC 62F	ICC 62R	ICC 61F	ICC 61R	ICC 60F	ICC 60R
rh	rh	rh	rh	rh	rh	rh	rh

## 捕获 / 比较单元 6

符号	位	类型	描述
<b>ICC6xR</b> (x = 0, 1, 2)	0, 2, 4	rh	<b>捕获, 比较 - 匹配上升沿标志</b> 比较模式下, T12 递增计数时检测到比较 - 匹配; 捕获模式下, 在 CC6x 的输入上检测到上升沿。 0 该位复位后事件还未发生 1 已检测到上述事件
<b>ICC6xF</b> (x = 0, 1, 2)	1, 3, 5	rh	<b>捕获, 比较 - 匹配下降沿标志</b> 比较模式下, T12 递减计数时检测到比较 - 匹配; 捕获模式下, 在 CC6x 的输入上检测到下降沿。 0 该位复位后事件还未发生 1 已检测到上述事件
<b>T12OM</b>	6	rh	<b>定时器 T12 1- 匹配标志</b> 0 该位复位后还未检测到定时器 T12 (递减计数时) 的 1 - 匹配 1 已检测到定时器 T12 (递减计数时) 的 1 - 匹配
<b>T12PM</b>	7	rh	<b>定时器 T12 周期 - 匹配标志</b> 0 该位复位后还未检测到定时器 T12 (递增计数时) 的周期 - 匹配 1 已检测到定时器 T12 (递增计数时) 的周期 - 匹配

## ISH

捕获 / 比较中断状态寄存器, 高位字节

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>STR</b>	<b>IDLE</b>	<b>WHE</b>	<b>CHE</b>	<b>TRP S</b>	<b>TRP F</b>	<b>T13 PM</b>	<b>T13 CM</b>
rh	rh	rh	rh	rh	rh	rh	rh

符号	位	类型	描述
<b>T13CM</b>	0	rh	<b>定时器 T13 比较匹配标志</b> 0 该位复位后还未检测到定时器 T13 的比较 - 匹配 1 已检测到定时器 T13 的比较 - 匹配
<b>T13PM</b>	1	rh	<b>定时器 T13 周期 - 匹配标志</b> 0 该位复位后还未检测到定时器 T13 的周期 - 匹配 1 检测到定时器 T13 的周期 - 匹配



符号	位	类型	描述
TRPF	2	rh	<p><b>强制中断标志</b></p> <p>若 <math>TRPPEN = 1</math> 且 <math>CTRAP = 0</math> 时硬件置位强制中断标志 TRPF，软件也可对该位置位。如果 <math>TRPM2 = 0</math>，输入 CTRAP 拉高时（<math>TRPPEN = 1</math>）硬件复位 TRPF；如果 <math>TRPM2 = 1</math>，必须软件复位 TRPF 退出强制中断状态。</p> <p>0 还未检测到强制中断条件</p> <p>1 已检测到强制中断条件（输入 <math>CTRAP = 0</math> 或由软件控制）</p>
TRPS	3	rh	<p><b>强制中断状态</b></p> <p>0 强制中断状态无效。</p> <p>1 强制中断状态有效。位 TRPF = 1 时置位 TRPS。根据寄存器 TRPCTR 所选择的模式复位 TRPS。</p> <p>在强制中断状态，所选的输出设为被动态。被动态时所驱动的逻辑电平由寄存器 PS LR 中的相应位定义。如果强制中断条件不再有效，但所选的同步条件还未发生时可出现 <math>TRPS = 1</math> 且 <math>TRPF = 0</math>。</p>
CHE	4	rh	<p><b>正确霍尔事件</b></p> <p>在每个有效的霍尔跳变沿，将 EXPH 的内容和引脚 CCPOSx 上的序列进行比较，若二者相等，置位 CHE。</p> <p>0 该位复位后还未检测到跳变至正确的（期望）霍尔事件</p> <p>1 已检测到跳变至正确的（期望）霍尔事件</p>
WHE	5	rh	<p><b>错误霍尔事件</b></p> <p>在每个有效的霍尔跳变沿，将 EXPH 的内容和引脚 CCPOSx 序列进行比较，若 CURH 及 EXPH 与 CCPOSx 均不相等，置位 WHE（错误的霍尔事件）。</p> <p>0 该位复位后还未检测到跳变至错误的（非期望）霍尔事件</p> <p>1 已检测到跳变至错误的（非期望）霍尔事件</p>
IDLE	6	rh	<p><b>空闲状态</b></p> <p>该位和 WHE（错误霍尔事件）一起被置位，IDLE 必须由软件复位。</p> <p>0 无操作</p> <p>1 位域 MCMP 被清零并保持复位状态，所选输出设置为被动态</p>

符号	位	类型	描述
<b>STR</b>	7	rh	<b>多通道模式映射传送请求</b> 多通道模式下，当发生从 MCMOUTS 到 MCMOUT 的映射传送时该位被置位。 0 未发生映射传送 1 已发生映射传送

注：并非寄存器 IS 中所有位均可产生中断，寄存器 IS 中还包含状态位，这些位的置位和复位操作和其它位相似。

注：中断的产生和寄存器 IS 中的值无关，例如：相应位已被置位，将产生中断（若被使能）。检测到寄存器 IS 中相应位（硬件或软件产生）的置位条件时，触发产生中断。

注：比较模式（及霍尔模式）下，只有在定时器运行（TxR = 1）时才能产生和定时器相关的中断。捕获模式下，定时器 T12 不工作也产生捕获中断。

寄存器 ISS 包含软件产生 CCU6 中断请求所需要的置位控制位。

## ISSL

捕获 / 比较中断状态置位寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>S</b> <b>T12</b> <b>PM</b>	<b>S</b> <b>T12</b> <b>OM</b>	<b>S</b> <b>CC</b> <b>62F</b>	<b>S</b> <b>CC</b> <b>62R</b>	<b>S</b> <b>CC</b> <b>61F</b>	<b>S</b> <b>CC</b> <b>61R</b>	<b>S</b> <b>CC</b> <b>60F</b>	<b>S</b> <b>CC</b> <b>60R</b>
<b>W</b>	<b>W</b>	<b>W</b>	<b>W</b>	<b>W</b>	<b>W</b>	<b>W</b>	<b>W</b>

符号	位	类型	描述
<b>SCC60R</b>	0	w	<b>置位捕获，比较 - 匹配上升沿标志</b> 0 无操作 1 将置位寄存器 IS 中的 ICC60R
<b>SCC60F</b>	1	w	<b>置位捕获，比较 - 匹配下降沿标志</b> 0 无操作 1 将置位寄存器 IS 中的 ICC60F
<b>SCC61R</b>	2	w	<b>置位捕获，比较 - 匹配上升沿标志</b> 0 无操作 1 将置位寄存器 IS 中的 ICC61R
<b>SCC61F</b>	3	w	<b>置位捕获，比较 - 匹配下降沿标志</b> 0 无操作 1 将置位寄存器 IS 中的 ICC61F

符号	位	类型	描述
<b>SCC62R</b>	4	w	置位捕获，比较匹配上升沿标志 0 无操作 1 将置位寄存器 IS 中的 ICC62R
<b>SCC62F</b>	5	w	置位捕获，比较匹配下降沿标志 0 无操作 1 将置位寄存器 IS 中的 ICC62F
<b>ST12OM</b>	6	w	置位定时器 T12 1- 匹配标志 0 无操作 1 将置位寄存器 IS 中的 T12OM
<b>ST12PM</b>	7	w	置位定时器 T12 周期 - 匹配标志 0 无操作 1 将置位寄存器 IS 中的 T12PM

注： 如果硬件置位相应的状态标志引起中断，软件置位具有相同的作用。

## ISSH

捕获 / 比较中断状态置位寄存器，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>S STR</b>	<b>S IDLE</b>	<b>S WHE</b>	<b>S CHE</b>	<b>S WHC</b>	<b>S TRPF</b>	<b>S T13 PM</b>	<b>S T13 CM</b>
w	w	w	w	w	w	w	w

符号	位	类型	描述
<b>ST13CM</b>	0	w	置位定时器 T13 比较匹配标志 0 无操作 1 将置位寄存器 IS 中的位 T13CM
<b>ST13PM</b>	1	w	置位定时器 T13 周期 - 匹配标志 0 无操作 1 将置位寄存器 IS 中的位 T13PM
<b>STRPF</b>	2	w	置位强制中断标志 0 无操作 1 将置位寄存器 IS 中的 TRPF 和 TRPS
<b>SWHC</b>	3	w	软件霍尔比较 0 无操作 1 触发霍尔比较动作

符号	位	类型	描述
<b>SCHE</b>	4	w	置位正确霍尔事件标志 0 无操作 1 将置位寄存器 IS 中的位 CHE
<b>SWHE</b>	5	w	置位错误霍尔事件标志 0 无操作 1 将置位寄存器 IS 中的 WHE
<b>SIDLE</b>	6	w	置位 IDLE 标志 0 无操作 1 将置位寄存器 IS 中的 IDLE
<b>SSTR</b>	7	w	置位 STR 标志 0 无操作 1 将置位寄存器 IS 中的 STR

寄存器 ISR 中包含中断请求的复位控制位，用来对相应标志进行软件复位。

## ISRL

捕获 / 比较中断状态复位寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
R T12 PM	R T12 OM	R CC 62F	R CC 62R	R CC 61F	R CC 61R	R CC 60F	R CC 60R
w	w	w	w	w	w	w	w

符号	位	类型	描述
<b>RCC60R</b>	0	w	复位捕获，比较匹配上升沿标志 0 无操作 1 将复位寄存器 IS 中的 ICC60R
<b>RCC60F</b>	1	w	复位捕获，比较匹配下降沿标志 0 无操作 1 将复位寄存器 IS 中的 ICC60F
<b>RCC61R</b>	2	w	复位捕获，比较匹配上升沿标志 0 无操作 1 将复位寄存器 IS 中的 ICC61R
<b>RCC61F</b>	3	w	复位捕获，比较匹配下降沿标志 0 无操作 1 将复位寄存器 IS 中的 ICC61F

## 捕获 / 比较单元 6

符号	位	类型	描述
<b>RCC62R</b>	4	w	复位捕获，比较匹配上升沿标志 0 无操作 1 将复位寄存器 IS 中的 ICC62R
<b>RCC62F</b>	5	w	复位捕获，比较匹配下降沿标志 0 无操作 1 将复位寄存器 IS 中的 ICC62F
<b>RT12OM</b>	6	w	复位定时器 T12 1- 匹配标志 0 无操作 1 将复位寄存器 IS 中的 T12OM
<b>RT12PM</b>	7	w	复位定时器 T12 周期 - 匹配标志 0 无操作 1 将复位寄存器 IS 中的 T12PM

**ISRH**

捕获 / 比较中断状态复位寄存器，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>R STR</b>	<b>R IDLE</b>	<b>R WHE</b>	<b>R CHE</b>	<b>0</b>	<b>R TRPF</b>	<b>R T13 PM</b>	<b>R T13 CM</b>
w	w	w	w	r	w	w	w

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 无操作 1 将复位寄存器 IS 中的 T13CM
<b>RT13PM</b>	1	w	复位定时器 T13 周期 - 匹配标志 0 无操作 1 将复位寄存器 IS 中的 T13PM
<b>RTRPF</b>	2	w	复位强制中断标志 0 无操作 1 将复位寄存器 IS 中的 TRPF （当输入 CTRAP = 0 且 TRPPEN = 1 时，该位不起作用
<b>RCHE</b>	4	w	复位正确霍尔事件标志 0 无操作 1 将复位寄存器 IS 中的 CHE

## 捕获 / 比较单元 6

符号	位	类型	描述
<b>RWHE</b>	5	w	<b>复位错误霍尔事件标志</b> 0 无操作 1 将复位寄存器 IS 中的 WHE
<b>RIDLE</b>	6	w	<b>复位 IDLE 标志</b> 0 无操作 1 将复位寄存器 IS 中的 IDLE
<b>RSTR</b>	7	w	<b>复位 STR 标志</b> 0 无操作 1 将复位寄存器 IS 中的 STR
<b>0</b>	3	r	<b>保留</b> 读操作返回 0；应写入 0。

## IENL

捕获 / 比较中断使能寄存器，低位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>EN T12 PM</b>	<b>EN T12 OM</b>	<b>EN CC 62F</b>	<b>EN CC 62R</b>	<b>EN CC 61F</b>	<b>EN CC 61R</b>	<b>EN CC 60F</b>	<b>EN CC 60R</b>
<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>	<b>rw</b>

符号	位	类型	描述
<b>ENCC60R</b>	0	rw	<b>通道 0 捕获，比较匹配上升沿中断使能</b> 0 当寄存器 IS 中的 ICC60R 被置位时，不产生中断 1 当寄存器 IS 中的 ICC60R 被置位时产生中断。 由位域 INPCC60 选择被激活的中断线
<b>ENCC60F</b>	1	rw	<b>通道 0 捕获，比较匹配下降沿中断使能</b> 0 当寄存器 IS 中的 ICC60F 被置位时，不产生中断 1 当寄存器 IS 中的 ICC60F 被置位时产生中断。 由位域 INPCC60 选择被激活的中断线
<b>ENCC61R</b>	2	rw	<b>通道 1 捕获，比较匹配上升沿中断使能</b> 0 当寄存器 IS 中的 ICC61R 被置位时，不产生中断 1 当寄存器 IS 中的 ICC61R 被置位时产生中断。 由位域 INPCC61 选择被激活的中断线

## 捕获 / 比较单元 6

符号	位	类型	描述
<b>ENCC61F</b>	3	rw	<b>通道 1 捕获，比较匹配下降沿中断使能</b> 0 当寄存器 IS 中的 ICC61F 被置位时，不产生中断 1 当寄存器 IS 中的 ICC61F 被置位时产生中断。 由位域 INPCC61 选择被激活的中断线
<b>ENCC62R</b>	4	rw	<b>通道 2 捕获，比较匹配上升沿中断使能</b> 0 当寄存器 IS 中的 ICC62R 被置位时，不产生中断 1 当寄存器 IS 中的 ICC62R 被置位时产生中断。 由位域 INPCC62 选择被激活的中断线
<b>ENCC62F</b>	5	rw	<b>通道 2 捕获，比较匹配下降沿中断使能</b> 0 当寄存器 IS 中的 ICC62F 被置位时，不产生中断 1 当寄存器 IS 中的 ICC62F 被置位时产生中断。 由位域 INPCC62 选择被激活的中断线
<b>ENT12OM</b>	6	rw	<b>T12 1- 匹配中断使能</b> 0 当寄存器 IS 中的 T12OM 被置位时，不产生中断 1 当寄存器 IS 中的 T12OM 被置位时产生中断。 由位域 INPT12 选择被激活的中断线
<b>ENT12PM</b>	7	rw	<b>T12 周期 - 匹配中断使能</b> 0 当寄存器 IS 中的 T12PM 被置位时，不产生中断 1 当寄存器 IS 中的 T12PM 被置位时产生中断。 由位域 INPT12 选择被激活的中断线

**IENTH**

捕获 / 比较中断使能寄存器，高位字节

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>EN STR</b>	<b>EN IDLE</b>	<b>EN WHE</b>	<b>EN CHE</b>	<b>0</b>	<b>EN TRPF</b>	<b>EN T13 PM</b>	<b>EN T13 CM</b>
rw	rw	rw	rw	r	rw	rw	rw

符号	位	类型	描述
<b>ENT13CM</b>	0	rw	<b>T13 比较 - 匹配中断使能</b> 0 当寄存器 IS 中的 T13CM 被置位时，不产生中断 1 当寄存器 IS 中的 T13CM 被置位时产生中断。由位域 INPT13 选择被激活的中断线
<b>ENT13PM</b>	1	rw	<b>T13 周期 - 匹配中断使能</b> 0 当寄存器 IS 中的 T13PM 被置位时，不产生中断 1 当寄存器 IS 中的 T13PM 被置位时产生中断。由位域 INPT13 选择被激活的中断线
<b>ENTRPF</b>	2	rw	<b>强制中断标志中断使能</b> 0 当寄存器 IS 中的 TRPF 被置位时，不产生中断 1 当寄存器 IS 中的 TRPF 被置位时产生中断。由位域 INPERR 选择被激活的中断线
<b>ENCHE</b>	4	rw	<b>正确霍尔事件中断使能</b> 0 当寄存器 IS 中的 CHE 被置位时，不产生中断 1 当寄存器 IS 中的 CHE 被置位时产生中断。由位域 INPCHE 选择被激活的中断线
<b>ENWHE</b>	5	rw	<b>错误霍尔事件中断使能</b> 0 当寄存器 IS 中的 WHE 被置位时，不产生中断 1 当寄存器 IS 中的 WHE 被置位时产生中断。由位域 INPERR 选择被激活的中断线
<b>ENIDLE</b>	6	rw	<b>空闲使能</b> 该位使能检测到错误霍尔事件（位 WHE 被置位）后自动进入空闲状态（位 IDLE 被置位）。空闲状态下位域 MCMP 自动被清零。 0 检测到错误霍尔事件时，位 IDLE 不自动被置位 1 检测到错误霍尔事件时，位 IDLE 自动被置位
<b>ENSTR</b>	7	rw	<b>多通道模式映射传送中断使能</b> 0 当寄存器 IS 中的 STR 被置位时，不产生中断 1 当寄存器 IS 中的 STR 被置位时产生中断。由位域 INPCHE 选择被激活的中断线
<b>0</b>	3	r	<b>保留</b> 读操作返回 0；应写入 0。



# INPL

捕获 / 比较中断节点指针寄存器，低位字节 复位值：40<sub>H</sub>

7	6	5	4	3	2	1	0
INP CHE		INP CC62		INP CC61		INP CC60	
rw		rw		rw		rw	

符号	位	类型	描述
INPCC60	1:0	rw	<b>通道 0 中断的中断节点指针</b> 该位域定义了当位 ICC60R（若由位域 ENCC60R 使能）或 ICC60F（若由位域 ENCC60F 使能）被置位时所激活的中断输出线。 00 选择中断输出线 SR0 01 选择中断输出线 SR1 10 选择中断输出线 SR2 11 选择中断输出线 SR3
INPCC61	3:2	rw	<b>通道 1 中断的中断节点指针</b> 该位域定义了当位 ICC61R（若由位域 ENCC61R 使能）或 ICC61F（若由位域 ENCC61F 使能）被置位时所激活的中断输出线。 00 选择中断输出线 SR0 01 选择中断输出线 SR1 10 选择中断输出线 SR2 11 选择中断输出线 SR3
INPCC62	5:4	rw	<b>通道 2 中断的中断节点指针</b> 该位域定义了当位 ICC62R（若由位域 ENCC62R 使能）或 ICC62F（若由位域 ENCC62F 使能）被置位时所激活的中断输出线。 00 选择中断输出线 SR0 01 选择中断输出线 SR1 10 选择中断输出线 SR2 11 选择中断输出线 SR3

## 捕获 / 比较单元 6

符号	位	类型	描述
<b>INPCHE</b>	7:6	rw	<b>CHE 中断的中断节点指针</b> 该位域定义了当位 CHE（若由位域 ENCHE 使能）或 STR（若由位域 ENSTR 使能）被置位时所激活的中断输出线。 00 选择中断输出线 SR0 01 选择中断输出线 SR1 10 选择中断输出线 SR2 11 选择中断输出线 SR3

**INPH**

捕获 / 比较中断节点指针寄存器，高位字节

复位值：39<sub>H</sub>

7	6	5	4	3	2	1	0
0		<b>INP T13</b>		<b>INP T12</b>		<b>INP ERR</b>	
r		rw		rw		rw	

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> 该位域定义了当位 TRPF（若由位域 ENTRPF 使能）或 WHE（若由位域 ENWHE 使能）被置位时所激活的中断输出线。 00 选择中断输出线 SR0 01 选择中断输出线 SR1 10 选择中断输出线 SR2 11 选择中断输出线 SR3
<b>INPT12</b>	3:2	rw	<b>定时器 T12 中断的中断节点指针</b> 该位域定义了当位 T12OM（若由位域 ENT12OM 使能）或 T12PM（若由位域 ENT12PM 使能）被置位时所激活的中断输出线。 00 选择中断输出线 SR0 01 选择中断输出线 SR1 10 选择中断输出线 SR2 11 选择中断输出线 SR3

符号	位	类型	描述
<b>INPT13</b>	5:4	rw	<b>定时器 T13 中断的中断节点指针</b> 该位域定义了当位 T13CM（若由位域 ENT13CM 使能）或 T13PM（若由位域 ENT13PM 使能）被置位时所激活的中断输出线。 00 选择中断输出线 SR0 01 选择中断输出线 SR1 10 选择中断输出线 SR2 11 选择中断输出线 SR3
<b>0</b>	7:6	r	<b>保留</b> 读操作返回 0；应写入 0。

## 16 MultiCAN

控制器局域网络 MultiCAN 模块包含两个全功能 CAN 节点，这两个节点可独立工作或者通过网关功能交换数据和远程帧。根据 CAN V2.0B active 规范处理 CAN 帧的发送和接收。每个 CAN 节点都可以接收和发送带 11 位标识符的标准帧和带 29 位标识符的扩展帧。

两个 CAN 节点共用一套报文对象，每个报文对象可被独立分配给两个 CAN 节点之一。除了可用于存储接收帧和发送帧，报文对象还可组合起来在 CAN 节点之间构建网关或者建立 FIFO 缓存器。

可将报文对象组织为双链表结构，每个 CAN 节点都有自己的报文对象列表。CAN 节点仅将帧储存在分配给该 CAN 节点列表的报文对象中，且仅发送属于该报文对象列表中的报文。功能强大、由命令驱动列表控制器执行所有报文对象列表操作。

CAN 节点位时序由模块时钟 ( $f_{CAN}$ ) 控制，可编程的数据率高达 1 Mbit/s。外部总线收发器通过一对接收和发送引脚与 CAN 节点相连。

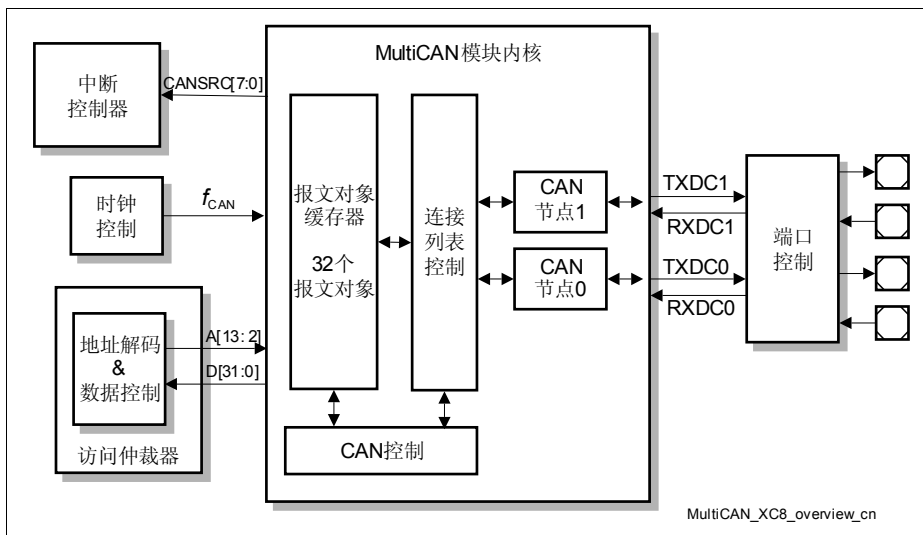


图 16-1 MultiCAN 模块总览

### 特性

- 与 ISO 11898 兼容
- 根据 CAN V2.0 B active 技术规范确定 CAN 功能
- 每个 CAN 节点都有专用控制寄存器
- 数据传送速率高达 1 Mbit/s
- 具有灵活、功能强大的报文传送控制和错误处理能力
- 具有先进的 CAN 总线位时序分析和由帧计数器实现的波特率检测功能
- 全功能 CAN：一套（共 32 个）报文对象可被独立地

- 指派（分配）给任一 CAN 节点
- 配置为发送或接收对象
- 处理 11 位标识符的标准帧或者 29 位标识符的扩展帧
- 通过帧计数器用时间戳识别
- 设置为远程监控模式
- 先进的验收滤波功能
  - 为每个报文对象提供单独的验收屏蔽寄存器，对接收帧进行验收滤波
  - 报文对象可以配置为只接收标准帧或者扩展帧，或者两者都接收
  - 可将报文对象分成四级优先级组，用于发送和接收
  - 根据 CAN 仲裁规则，由帧标识符、IDE 位和 RTR 位选择，或根据其在列表中的次序选择最先发送的报文
- 先进的报文对象功能
  - 报文对象可以组合成任意容量的 FIFO 报文缓存器，容量仅受报文对象总数的限制
  - 报文对象可以互连构成网关，在两个不同 CAN 总线之间自动进行帧传送。单网关可以将任意两个 CAN 节点连接起来。可以定义任意数目的网关
- 先进的数据管理
  - 报文对象可组织为双链列表
  - 在任意时刻、即使在 CAN 节点全速工作期间，都可以进行列表重组
  - 功能强大，由命令驱动列表控制器管理列表的结构并确保列表的一致性
  - 报文 FIFO 基于列表结构，在 CAN 工作期间易于重新划分其大小
  - 静态分配命令与基于非列表的 TwinCAN 应用兼容
- 先进的中断处理
  - 多达 8 条中断输出线，中断请求可单独发至其中之一
  - 报文后处理通知功能可以灵活组成具有 64 个通知位的专用寄存器

## 16.1 MultiCAN 内核功能描述

本节介绍 MultiCAN 模块的功能。

### 16.1.1 模块结构

图 16-2 给出 MultiCAN 模块的总体结构。

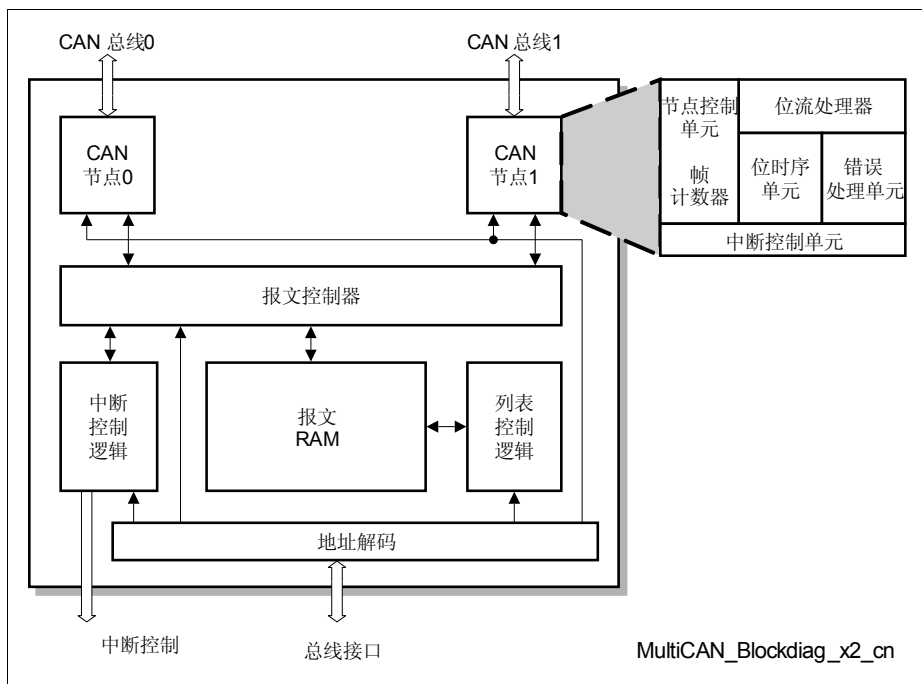


图 16-2 MultiCAN 框图

### CAN 节点

每个 CAN 节点由多个子单元组成。

- 位流处理器

位流处理器根据 ISO 11898 标准, 进行数据帧、远程帧、错误帧和过载帧处理, 还包括串行数据流和输入 / 输出寄存器之间的转换。

- 位时序单元

考虑到传播延迟和相移误差, 位时序单元根据用户的设置确定位时间长度和采样点位置, 位时序单元还执行再同步操作。

- **错误处理单元**

错误处理单元用来管理接收和发送错误计数器。根据两个计数器的内容，可使 CAN 节点进入到“错误激活”，“错误认可”或“总线关闭”状态。

- **节点控制单元**

节点控制单元协调 CAN 节点的操作：

- 使能 / 禁止 CAN 节点传送
- 使能 / 禁止并产生可引发中断请求的特定节点事件（如 CAN 总线错误、帧传送成功等等）
- 帧计数器管理

- **中断控制单元**

中断控制单元根据 CAN 节点中可能出现的不同情况，控制相应中断的产生。

## 报文控制器

报文控制器处理 CAN 节点之间的 CAN 帧交换，也处理存储在报文 RAM 中的报文对象。该单元执行下列操作：

- 进行接收验收滤波，确定用来存储接收 CAN 帧的正确报文对象
- 进行发送验收滤波，单独为每一个 CAN 节点确定要最先发送的报文对象
- 根据报文对象的状态 / 控制位等，在报文对象和 CAN 节点之间进行内容传送
- 处理 FIFO 缓存和网关功能
- 集合报文挂起通知位

## 列表控制器

列表控制器执行所有双链报文对象列表的修改操作。仅允许列表控制器对列表结构进行修改。由一个用户命令接口（命令面板）请求报文对象的分配 / 再分配操作，然后由列表控制器状态机自动执行所请求的命令。

## 中断控制

通用中断结构见 图 16-3。中断事件触发中断的产生，中断脉冲的产生独立于中断状态寄存器中的中断标志。通过软件向中断标志写 0 对其复位。

如果中断使能寄存器中相应的中断使能位被使能，该操作可在模块的 8 条中断输出线 CANSRCm 之一上产生一个中断脉冲。如果多个中断源连接在同一个中断节点指针上（在中断节点指针寄存器中），这些请求被合并为同一条命令线。

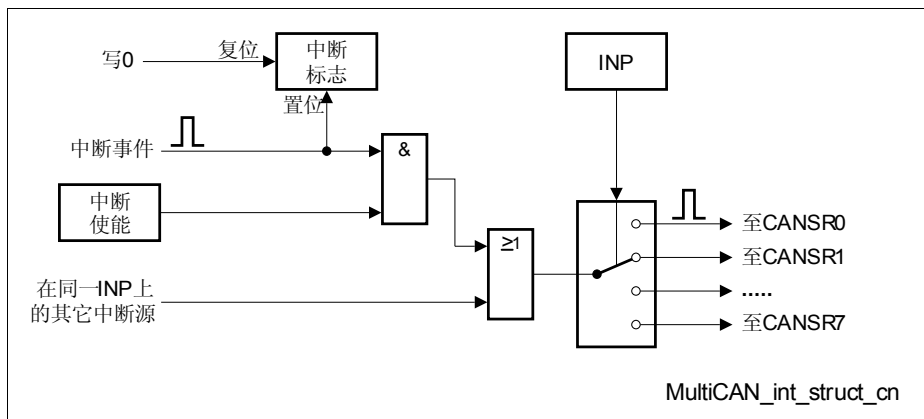


图 16-3 基本中断结构图



## 16.1.2 时钟控制

**表 16-1** 给出有效 CAN 节点波特率为 1 Mbit/s 时，所需的最小工作频率  $f_{CAN}$ ，单位为 MHz。如果需要的波特率小于 1 Mbit/s，所需频率值依照线性比例确定（例如需要的最大波特率为 500 kbit/s，频率值为表中所给值的 50%。）

这些值对应于 CPU 以最大执行能力访问 MultiCAN 模块，可以是舍入的结果。

**表 16-1 最小工作频率 [MHz]**

被分配的报文对象个数 <sup>1)</sup>	仅一个 CAN 节点有效	两个 CAN 节点都有效
16 个报文对象	12	19
32 个报文对象	15	23

1) 只考虑那些已经分配给 CAN 节点的报文对象，未分配的报文对象对最小工作频率没有影响。

### 16.1.3 CAN 节点控制

每个 CAN 节点都可单独配置并运行，而不依赖于其它 CAN 节点。每个 CAN 节点都配有一套独立的 SFR 寄存器，对该节点进行控制和监控。

注： 在下面的描述中，“x”代表节点号，“n”代表报文对象编号。

#### 16.1.3.1 位时序单元

根据 ISO 11898 标准，一个 CAN 位时间可被划分为不同的时间段（见图 16-4）。每个时间段包含几个基本时间单元  $t_q$ ，由位域 NBTRx.BRP 和 NBTRx.DIV8 调整  $t_q$  的大小，这两个位域还可用来控制由模块时钟  $f_{CAN}$  驱动的波特率预分频器。

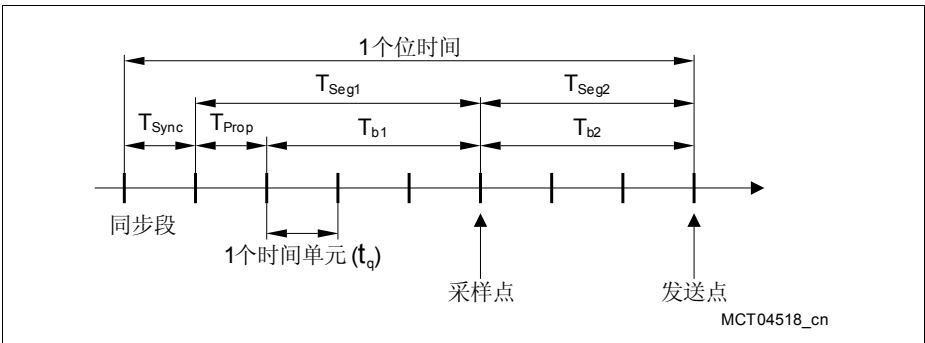


图 16-4 标准 CAN 总线位时序

在同步段 ( $T_{Sync}$ ) 进行发射和接收时间基准之间的相位同步。同步段长度总为 1 个  $t_q$ 。传播段 ( $T_{Prop}$ ) 考虑 CAN 总线上的发射输出驱动器和收发电路的物理传播延迟。对于工作冲突检测机制来讲， $T_{Prop}$  必须等于所有传播延迟量的总和舍入到  $t_q$  的整数倍之后，再乘以 2 所得的值。采样点之前和之后的相位缓冲段 1 和 2 ( $T_{b1}$ ,  $T_{b2}$ ) 用于补偿同步段中检测到的发射和接收之间的时钟相位失配。

重新同步阶段所允许的时间单元的最大值由位域 NBTRx.SJW 定义。传播段和相位缓冲段 1 结合成参数  $T_{Seg1}$ ，由 NBTRx.TSEG1 中的值定义，ISO 标准要求其最小值为 3 个时间单元。参数  $T_{Seg2}$  由 NBTRx.TSEG2 的值定义，包含了相位缓冲段 2，ISO 标准要求其最小值为 2 个时间单元。根据 ISO 标准，CAN 位时间，是  $T_{Sync}$ ， $T_{Seg1}$  和  $T_{Seg2}$  的总和，必须不少于 8 个时间单元。

位时间计算：

$$\begin{aligned}
 t_q &= (BRP + 1) / f_{CAN} && \text{如果 DIV8} = 0 \\
 &= 8 \times (BRP + 1) / f_{CAN} && \text{如果 DIV8} = 1 \\
 T_{Sync} &= 1 \times t_q \\
 T_{Seg1} &= (TSEG1 + 1) \times t_q && (\text{最小: } 3 t_q)
 \end{aligned}$$

$$T_{\text{Seg2}} = (T\text{SEG2} + 1) \times t_q \quad (\text{最小: } 2 t_q)$$

$$\text{位时间} = T_{\text{Sync}} + T_{\text{Seg1}} + T_{\text{Seg2}} \quad (\text{最小: } 8 t_q)$$

为了补偿不同 CAN 控制器之间的时钟相移，CAN 控制器必须在从隐性到显性总线电平的任意边沿进行同步。如果硬同步被使能（在一帧的开始），在同步段重新开始位时间。否则，重新同步跳转宽度  $T_{\text{SJW}}$  定义最大数目的时间单元，重新同步操作可能缩短或拉长位时间。SJW 的值由位域 NBTRx.SJW 定义。

$$T_{\text{SJW}} = (\text{SJW} + 1) \times t_q$$

$$T_{\text{Seg1}} \geq T_{\text{SJW}} + T_{\text{prop}}$$

$$T_{\text{Seg2}} \geq T_{\text{SJW}}$$

$f_{\text{CAN}}$  的最大相对容差由相位缓冲段和重新同步跳转宽度决定。

$$df_{\text{CAN}} \leq \min(T_{b1}, T_{b2}) / 2 \times (13 \times \text{位时间} - T_{b2}) \quad \text{且}$$

$$df_{\text{CAN}} \leq T_{\text{SJW}} / 20 \times \text{位时间}$$

必须在复位 NCRx.INIT 之前，即在使能 CAN 节点工作之前，将有效的位时序写入到寄存器 NBTR 中。仅在位 NCRx.CCE 置位（使能配置更改）的情况下，才可以对寄存器 NBTRx 进行写操作。

### 16.1.3.2 位流处理器

根据报文缓存中的报文对象，由位流处理器产生通过 CAN 总线发送的数据帧和远程帧。该处理器控制 CRC 产生器，且给新的远程帧或数据帧添加校验和信息。在加入‘帧起始位’和‘帧结束域’之后，位流处理器开始 CAN 总线仲裁过程，且当发现总线空闲时连续进行帧发送。进行数据发送的同时，位流控制器连续地监控 I/O 线。如果（总线仲裁阶段或应答时隙之外）在 I/O 线上的电平和发送移位寄存器当前送出位的逻辑状态之间检测到失配，产生一个‘最近错误’中断请求，错误码由位域 NSRx.LEC 给出。

通过检验相关的 CRC 域，验证正在接收的帧的数据一致性。当检测到错误时，产生一个‘最近错误’中断请求，错误码由位域 NSRx.LEC 给出。此外，产生一个错误帧并将其发送到 CAN 总线上。将一个无错误的帧分解成标识符和数据部分之后，接收到的信息被传送到报文缓存，执行远程帧和数据帧处理，中断产生和状态处理。

### 16.1.3.3 错误处理单元

CAN 节点 x 的错误处理单元负责对 CAN 器件进行故障界定。该单元有两个计数器，接收错误计数器 NECNTx.REC 和发送错误计数器 NECNTx.TEC，这两个计数器根据来自位流处理器的命令相应地增加和减少。如果在发送操作的同时，位流处理器自身检测到了一个错误，发送错误计数器增加 8。当一个外部 CAN 节点因为错误帧产生而报告一个

错误情况，节点计数器增加 1。对于错误分析、出错的报文传送方向和识别到传送错误的节点，由相关的 CAN 节点 x 的寄存器 NECNTx 指示。根据错误计数器中的值，CAN 节点被设置为如下状态：“错误激活”，“错误认可”和“总线关闭”。

如果两个错误计数器的值都低于“错误认可”界限 128，CAN 处于错误激活状态。如果至少两者之一等于或者大于 128，CAN 节点处于错误认可状态。

如果发送错误计数器的值等于或者大于“总线关闭”界限 256，“总线关闭”状态被激活，并由标志符 NSRx.BOFF 报告该状态。器件一直保持在该状态，直到完成“总线关闭”恢复过程为止。此外，当至少一个错误计数器等于或大于位域 NECNTx.EWRNLVL 中定义的错误警告值时，NSRx.EWRN 被置位。如果两个错误计数器的值再次降到错误警告值以下，NSRx.EWRN 被复位。

### 16.1.3.4 CAN 帧计数器

每个 CAN 节点都配有帧计数器，用来计数 CAN 帧的发送 / 接收，或者获得关于 CAN 节点何时开始帧发送 / 接收的时刻信息。CAN 帧计数 / 位时间计数的操作由一个 16 位计数器完成，该计数器由 NFCRx 控制。由位域 NFCRx.CFSEL 确定帧计数器的工作模式。

- **帧计数模式：**  
在成功地发送和 / 或接收一个 CAN 帧之后，帧计数器增加。新计数值保存在位域 NFCRx.CFC 中，并被复制到与传送有关的报文对象的位域 MOIPRn.CFCVAL 中。
- **时间戳模式：**  
在新的位时间开始时，帧计数器增加。当一帧的发送 / 接收开始时，帧计数器值被捕获并存储到位域 NFCRx.CFC 中。成功传送该帧之后，捕获值被复制到与传送有关的报文对象的位域 MOIPRn.CFCVAL 中。
- **位时序模式：**  
用于波特率检测和位时序分析（见[章节 16.1.5.3](#)）。

### 16.1.3.5 CAN 节点中断

每个 CAN 节点都配有 4 个中断源，用于产生中断请求：

- 成功发送 / 接收一帧
- 带最近错误码的 CAN 协议错误
- 出现警告情况：发送 / 接收错误计数器达到警告界限，总线关闭状态改变，出现列表长度错误，或者出现列表对象错误
- 帧计数器溢出

除了由硬件产生中断，也可通过寄存器 MITR 由软件触发中断。向位域 MITR.IT 的位 n 写 1，将在相应的中断输出线 CANSRCm 上产生中断请求信号。当写访问位域 MITR.IT 时，如果多于一位被置位，则导致相应的多条中断输出线 CANSRCm 被同时激活。

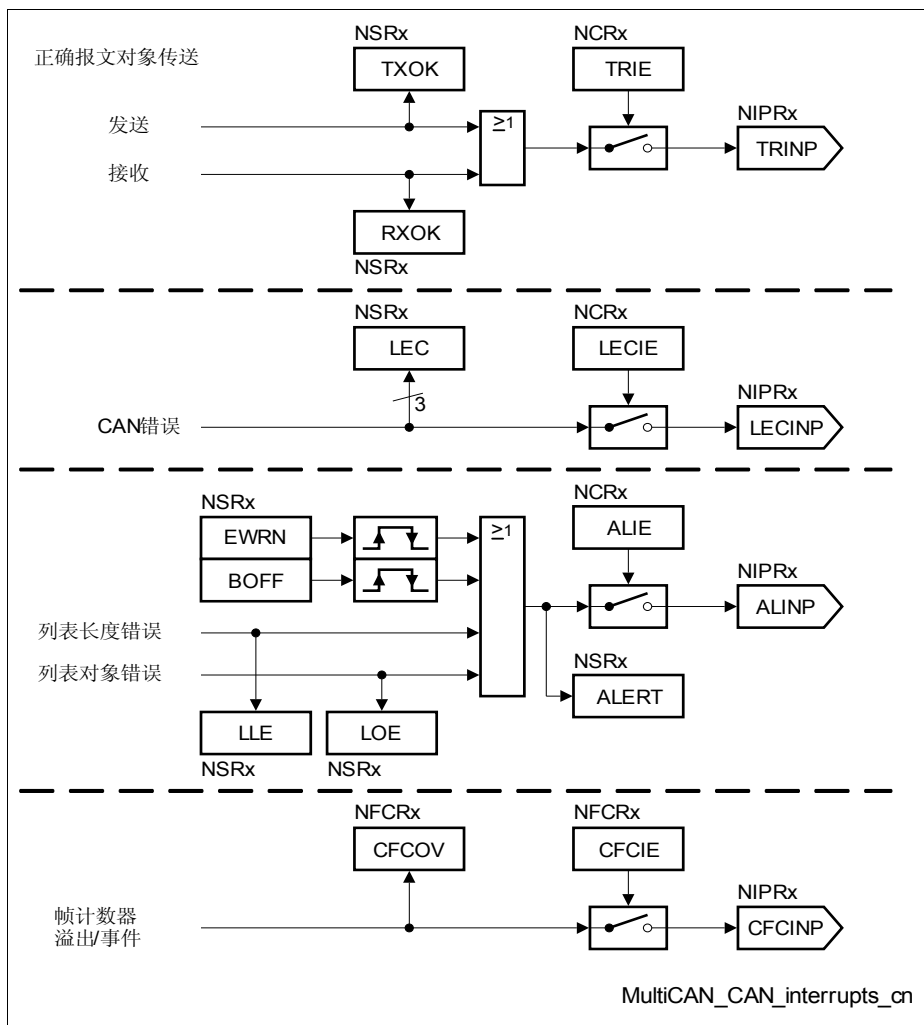


图 16-5 CAN 节点中断

## 16.1.4 报文对象列表结构

本节描述 MultiCAN 模块的报文对象列表结构。

### 16.1.4.1 基本知识

MultiCAN 模块的报文对象被组织为双链列表，在该列表结构中，每个报文对象都有两个指针，一个指向列表中前一个报文对象，另一个指向列表中的后一个报文对象。MultiCAN 模块提供 8 个列表，每个报文对象分配给其中之一。在图 16-6 的例子中，3 个报文对象 3、5、16 分配给了编号为 2 的列表（列表寄存器 LIST2）。

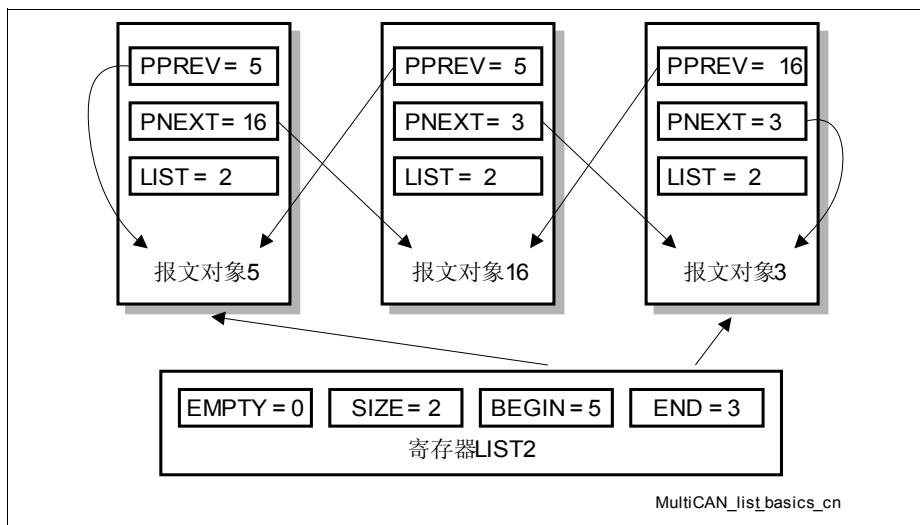


图 16-6 列表中分配报文对象示例

位域 LIST.BEGIN 指向列表中的第一个元素（例中的报文对象 5），位域 LIST.END 指向列表中的最后一个元素（例中的报文对象 3）。列表中的元素个数由位域 LIST.SIZE 给出（SIZE = 列表中的元素数 - 1，因此例子中元素数为 3，SIZE = 2）。位 LIST.EMPTY 指示该列表是否为空（因为列表 2 不为空，所以例中 EMPTY = 0）。

每个报文对象  $n$  有一个指针 MOCTRn.PNEXT，用来指向列表中的下一个报文对象，同时还有一个指针 MOCTRn.PPREV，用来指向列表中的前一个报文对象。第一个报文对象的 PPREV 指向其本身，因为第一个报文对象之前没有报文对象（例中的报文对象 5 为第一个报文对象，由 PPREV = 5 表示）。最后一个报文对象的 PNEXT 也指向其本身，因为最后一个报文对象之后没有报文对象（例中报文对象 3 为列表的最后一个报文对象，由 PNEXT = 3 表示）。

位域 MOCTRn.LIST 指示报文对象当前所属的列表编号，例中的报文对象都分配给了列表 2。因此，所有的分配给列表 2 的报文对象的 LIST 位域都设置为 LIST = 2。

#### 16.1.4.2 未被分配的报文对象列表

列表 0 具有特殊的意义：它是那些未被分配的报文对象的列表。如果一个元素属于列表 0 ( $\text{MOCTRn.LIST} = 0$ )，那就称其未被分配。如果一个元素属于一个编号不等于 0 的列表 ( $\text{MOCTRn.LIST} > 0$ )，称该报文对象被分配。

复位之后，所有的报文对象都是未分配的。这意味着，其  $\text{MOCTRn.LIST} = 0$ ，即都被分配给了列表 0。在由复位引起的报文对象初始分配之后，所有未被分配的报文对象的列表按照报文对象编号来排序（在报文对象  $n$  之前是报文对象  $n-1$ ，之后是报文对象  $n+1$ ）。

#### 16.1.4.3 与 CAN 节点连接

每个 CAN 节点都精确地和唯一的报文对象列表相关联。一个 CAN 节点仅传送分配给该节点列表的报文对象，见图 16-7。一个 CAN 节点接收到的帧也只能存储到属于该节点的报文对象中，而一个 CAN 节点也只能在分配给该节点的报文对象中选择要发送的帧，如图中垂直箭头所示。

列表的个数（8 个）多于节点个数（2 个）。这意味着一些列表没有和 CAN 节点相关联。分配给了没有和 CAN 节点相连的列表的报文对象不能直接从 CAN 节点接收报文，也不可以发送报文。

FIFO 和网关机制和报文对象的编号相关，而并不和某个特定的列表直接相关。用户必须小心处理使得作为 FIFO/ 网关目标对象属于所期望的列表。该机制允许对那些不属于该 CAN 节点的列表进行处理。

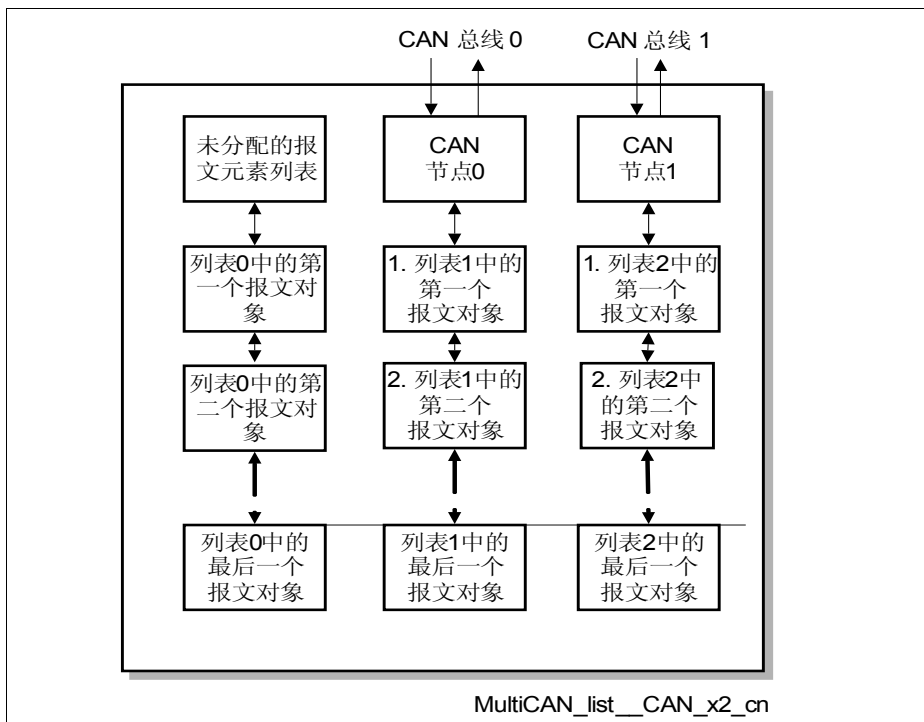


图 16-7 和 CAN 节点相连的报文对象

#### 16.1.4.4 列表命令面板

不能通过对列表寄存器 `LIST`，寄存器 `MOSTATn` 中的位域 `PPREV`，`PNEXT` 和 `LIST` 进行写操作的方式直接修改列表结构，因为这些寄存器是只读寄存器。列表结构的管理只限于通过 **MultiCAN** 模块内部的列表控制器执行。用户通过一个命令面板来控制列表控制器，向列表控制器发出列表分配命令。列表控制器有两个基本作用：

1. 确保所有修改列表结构的操作最终得到一个一致的列表结构
2. 为用户提供灵活性

列表控制器和相关的命令面板使得程序员只需注重列表的最终特性，这些最终特性以 **CAN** 节点的报文对象分配情况，以及分配给同一个列表的报文对象的次序关系为主要特征。由列表控制器完成列表构建（重构）过程。

表 16-2 给出列表命令的总体描述，表 16-7 给出面板命令的详细描述。



表 16-2 面板命令概览

命令名	描述
无操作	不启动新的命令
初始化列表	运行初始化序列，所有报文对象的 CTRL 和 LIST 位域复位
静态分配	将报文对象分配给一个列表
动态分配	将未被分配的对象列表中的第一个报文对象分配给选中的列表
静态插入目标对象之前	将一个报文对象（源对象）从其当前所属的列表中除去，并将其插入到给定目标对象所属的列表结构中，位置在给定目标对象之前。
动态插入目标对象之前	将一个新报文对象插入给定目标对象之前
静态插入目标对象之后	将报文对象（源对象）从其当前所属的列表中除去，并将其插入到给定目标对象所属的列表中，位置在给定目标之后
动态插入目标对象之后	将一个新报文对象插入到给定目标对象之后

通过向位域 PANCTR.PANCMD 写相应命令码的方式启动一个面板命令。在写命令码之前必须将相关的命令参数写入位域 PANCTR.PANAR1 和 PANCTR.PANAR2 中或者将用一个 32 位写访问指令将命令参数和命令码一起写入到寄存器 PANCTR。

写入一个有效的命令码，面板控制寄存器中的 PANCTR.BUSY 标志会被置位且对面板控制寄存器的下一步写访问被忽略。BUSY 标志一直有效，且控制面板保持锁定状态直到所请求的命令执行完毕。一次复位操作之后，列表控制器构建列表 0。在该操作期间，BUSY 被置位，并且禁止对 CAN RAM 的其它访问。当 BUSY 无效时，CAN RAM 才可能被再次访问。

*注：复位后，由列表控制器对 CAN RAM 进行自动初始化，以确保每个报文对象的列表指针指向正确。用位 PANCTR.BUSY 变为无效来指示 CAN RAM 初始化结束。*

在动态分配命令的情况下，从未被分配的报文对象的列表中取报文元素，位 PANCTR.RBUSY 和 BUSY (RBUSY = BUSY = 1) 将同时被置位。这表明列表控制器将用以下的方法对位域 PANCTR.PANAR1 和 PANCTR.PANAR2 进行更新：

1. 从未被分配的元素中取出的报文对象的编号被写入到位域 PANAR1
2. 如果 ERR (PANAR2 位 7) 置为 1，未被分配的单元列表为空且中止该命令。如果 ERR 为 0，未被分配的单元列表不为空且将成功执行该命令。

在列表控制器开始真正的分配过程之前写入动态分配命令的结果。一旦结果可用，RBUSY 再次变为无效 (RBUSY = 0)，而 BUSY 仍然保持有效，直到命令完成为止。这个特性允许用户在进行列表分配的过程中建立新的报文对象。在进行列表操作的同时，对报文对象的访问不受限制。然而，对位于 RAM 内部的寄存器资源的任何访问都会将正在进行的分配过程延迟一个访问周期。

一旦完成该命令，**BUSY** 标志变为无效 (**BUSY = 0**)，对面板控制寄存器的写访问再次被使能。此外，“无操作”命令码自动被写入到位域 **PANCTR.PANCMD** 中。当 **BUSY = 0** 时，可在任意时刻启动一个新命令。

寄存器 **PANCTR** 中的位 (**BUSY** 和 **RBUSY** 除外) 都可以由用户写入。如果命令面板被用在独立的 (可相互中断的) 中断程序中，这个特性允许对 **PANCTR** 进行保存和恢复。如果情况如此，那么任何使用命令面板的任务 (和可能利用命令面板中断另一个) 应该查询 **BUSY** 标志直到该标志符变为无效，且在发布命令之前将整个 **PANCTR** 寄存器保存到一个存储器地址处。在中断服务程序的结束部分，应该从该存储器地址恢复 **PANCTR**。

在将分配给有效的 **CAN** 节点列表的报文对象移动到另一个列表或者同一个列表的另一个位置之前，报文对象 **n** 的位 **MOCTRn.MSGVAL** (“报文有效”) 必须清零。

### 16.1.5 CAN 节点分析特性

本节描述 MultiCAN 模块的 CAN 节点分析特性。

#### 16.1.5.1 分析模式

CAN 分析模式可对 CAN 通信进行监控，而不影响 CAN 总线逻辑状态。置位 NPCRx.CALM，可选择 CAN 分析模式。

在 CAN 分析模式下，CAN 节点的发送引脚永久保持为隐性电平。CAN 节点可以接收帧（数据，远程和错误帧），但是不允许发送。接收到的数据 / 远程帧不会被应答（也就是说，应答间隙为隐性）但是只要其它任何一个节点应答了该帧，该帧将被接收和保存在匹配报文对象中。分析模式提供完整的报文对象功能，但是将不执行发送请求。

#### 16.1.5.2 回环模式

MultiCAN 模块具有回环模式，无需访问外部 CAN 总线就可以进行 MultiCAN 模块的在系统测试和 CAN 驱动软件的开发。

回环特性由一个内部 CAN 总线（位于 MultiCAN 模块内）和每个节点的总线选择开关构成。通过该开关，控制每个节点和内部 CAN 总线连接（回环模式被激活），或者和外部 CAN 总线相连，分别和该 CAN 节点的发送引脚或接收引脚相连（正常操作）。当前未选中的 CAN 总线被驱动为隐性，这意味着发送引脚保持为 1，处于回环模式下，CAN 节点忽略接收引脚。

置位 NPCRx.LBM 来选择回环模式。所有处于回环模式的 CAN 节点可以用内部总线进行通信，而不影响其它不处于回环模式的 CAN 节点的正常工

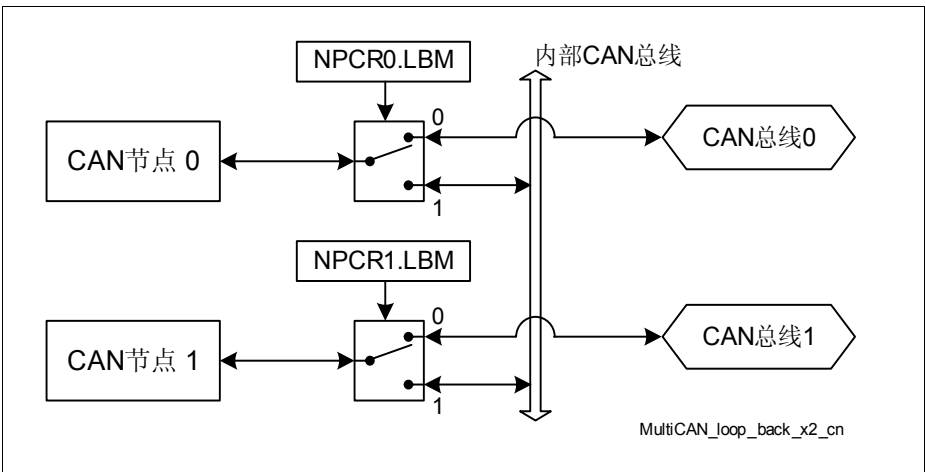


图 16-8 回环模式

### 16.1.5.3 位时序分析

每个 CAN 节点都通过 CAN 帧计数器分析模式执行详细的位时序分析。帧计数器的位时序分析功能可以用于 CAN 波特率自动检测和 CAN 网络的时序分析。

通过设置  $\text{NFCRx.CFMODE} = 10_{\text{B}}$ ，选择位时序分析功能。位时序分析不影响 CAN 节点的操作。位时序的测量结果被写入到位域  $\text{NFCRx.CFC}$ 。在位时序分析模式下，每当位域  $\text{NFCRx.CFC}$  被更新，位  $\text{NFCRx.CFCOV}$  都被置位，用来指示 CFC 更新事件。如果  $\text{NFCRx.CFCIE}$  被置位，可以产生一个中断请求（见图 16-5）。

#### 自动波特率检测

对于自动波特率检测，必须测量在 CAN 总线上观察到的连续显性沿之间的时间。如果位域  $\text{NFCRx.CFSEL} = 000_{\text{B}}$ ，则自动执行该测量。监测 CAN 接收输入线上每一个显性沿，该边沿和最近的显性沿之间的时间（测量以  $f_{\text{CAN}}$  时钟周期数为单位）被保存在位域  $\text{NFCRx.CFC}$  中。

#### 同步分析

如果  $\text{NFCRx.CFSEL} = 010_{\text{B}}$ ，则进行位时间同步监测。测量第一个显性沿和采样点之间的时间，并将其存储在位域  $\text{NFCRx.CFC}$  中。位时序同步偏移可以从该时间得出，因为采样点之后的第一个跳变沿触发同步，且在连续的采样点之间仅有一个同步。

例如，以测量到的波特率接收第一个 CAN 帧时，可利用同步分析对波特率进行精细调整。

#### 驱动延迟测量

当  $\text{NFCRx.CFSEL} = 011_{\text{B}}$ （显性到显性）且  $\text{NFCRx.CFSEL} = 100_{\text{B}}$ （隐性到隐性）时，测量发送沿和相应的接收沿之间的延迟。这些延迟指示将一个新的位值送到 CAN 总线上进行物理实现所需要的时间。

## 16.1.6 报文验收滤波

本节描述 MultiCAN 模块的报文验收滤波功能。

### 16.1.6.1 接收验收滤波

当从 CAN 节点接收 CAN 帧时，在成功接收之后，接收帧存储在唯一的一个报文对象中。只有满足下面六个条件的报文对象才具备接收 CAN 帧的资格：

- 这个报文对象被分配给了要接收该帧的 CAN 节点的报文列表中
- 位 MOSTATn.MSGVAL 被置位
- 位 MOSTATn.RXEN 被置位
- 位 MOSTATn.DIR 等于接收帧的位 RTR  
如果位 MOSTATn.DIR = 1（发送对象），报文对象仅验收远程帧。如果位 MOSTATn.DIR = 0（接收对象），报文对象仅验收数据帧。
- 如果位 MOAMRn.MIDE = 1，用下列方法评估接收帧 IDE 位：如果 MOARn.IDE = 1，接收帧 IDE 位必须置位（表示扩展标识符）；如果 MOARn.IDE = 0，接收帧 IDE 必须清零（表示标准标识符）。  
如果 MOAMRn.MIDE = 1，不需要注意接收帧 IDE 位。在这种情况下，带有标准和扩展帧的报文对象都可以通过验收。
- 如寄存器 MOAMRn 中验收屏蔽所限定的那样，接收帧的标识符和存储在寄存器 MOARn 中的标识符相匹配。这意味着接收报文对象标识符的每一位都和位域 MOARn.ID 相等，除了那些对应于位域 MOAMRn.AM 被清零的验收屏蔽位，这些标识符位对接收无影响。

满足所有这六条评判标准的报文中，具有最高接收优先级的报文对象赢得接收验收滤波并且被选中用来储存接收帧。所有其它报文对象失去接收验收滤波。

下面给出报文对象的优先级机制：

如果满足下面的两个条件，那么报文对象 a（MOa）比报文对象 b（MOb）优先级更高（见页 16-83）：

1. MOa 具有比 MOb 更高的优先级。这意味着，2 位优先级位域 MOARa.PRI 必须等于或者小于位域 MOARb.PRI。
2. 如果两个报文对象具有相同的优先级。（MOARa.PRI = MOARb.PRI）列表次序中 MOb 在 MOa 之后，这意味着，在列表中从 MOa 开始，通过顺序方式可到达 MOb。

### 16.1.6.2 发送验收滤波

报文发送请求是通过在报文所属的报文对象中设置发送请求实现的。如果同一个 CAN 节点有多于一个报文对象具有有效的发送请求，只在其中选择一个报文对象发送，因为在一条 CAN 总线上同时仅可发送一个报文对象。

在 CAN 节点上，满足下述四个条件的报文对象才具备发送的资格：

1. 报文对象分配给了 CAN 节点的报文对象列表
2. 位 MOSTATn.MSGVAL 被置位
3. 位 MOSTATn.TXRQ 被置位

#### 4. 位 MOSTATn.TXEN0 和 MOSTATn.TXEN1 被置位

由优先级机制判定最先发送哪一个报文对象。作如下假设：报文对象 **a**（**MOa**）和报文对象 **b**（**MOb**）都具备发送资格，在列表中 **MOb** 在 **MOa** 之后。这就是说，在列表中从 **MOa** 开始，通过顺序向前的方法可到达 **MOb**。

如果两个报文对象属于不同的优先级（位域 **MOARn.PRI** 中的值不同），**MOAR.PRI** 值较低的报文对象有较高优先级，将最先被发送。

如果两个报文对象具有相同的优先级（寄存器 **MOARn** 的 **PRI** 位域的值相等），如果满足下面的条件中任何一个，那么 **MOa** 比 **MOb** 的发送优先级高。

- **PRI** = 10<sub>B</sub> 且按照 CAN 仲裁规则，**MOa** 具有比 **MOb** 更高或者相同的优先等级（见表 16-13）
- **PRI** = 01<sub>B</sub> 或 **PRI** = 11<sub>B</sub>（由列表次序决定优先级）

有发送资格的报文中发送优先级最高的报文对象赢得发送验收滤波，将最先被发送。所有其它的报文对象失去当前一轮的发送验收滤波机会。在下面几轮的验收滤波中他们都将得到新的机会，再次进行验收滤波。

优先级规则 1 到 2 对于正常 CAN 操作同样有效。

## 16.1.7 报文后处理

在一个报文对象成功发送或者接收一帧后，可通知 CPU 在该报文对象上进行报文后处理。MultiCAN 模块的后处理包括两个单元：

1. 报文中断触发报文后处理
2. 报文挂起寄存器将报文挂起中断收集到一个公共结构中，用于报文后处理。

### 16.1.7.1 报文中断

当接收到的帧存储到一个报文对象中或者成功发送了一帧，可以发出报文中断。对于每个报文对象，可以产生发送和接收中断并将其发送至 8 条 CAN 中断输出线之一上（见图 16-9）。在由 FIFO 或者网关操作所引起的帧存储事件之后，也出现一个接收中断。不管相应的报文中断是否使能，在一次成功的发送 / 接收之后，状态位 MOSTATn.TXPND 和 MOSTATn.RXPND 总是被置位。

具有报文对象 FIFO 已满中断条件。如果位域 MOFCRn.OVIE 被置位，根据实际报文对象类型，FIFO 已满中断将被激活。

根据接收 FIFO 基本对象（MOFCRn.MMC = 0001<sub>B</sub>）的情况，根据发送中断节点指针 MOIPRn.TXINP 的设置，将 FIFO 已满中断请求被送至中断输出线 CANSRCm。

根据发送 FIFO 基本对象（MOFCRn.MMC = 0010<sub>B</sub>）的情况，根据接收中断节点指针 MOIPRn.RXINP 的设置，将 FIFO 已满中断请求被送至中断输出线 CANSRCm。

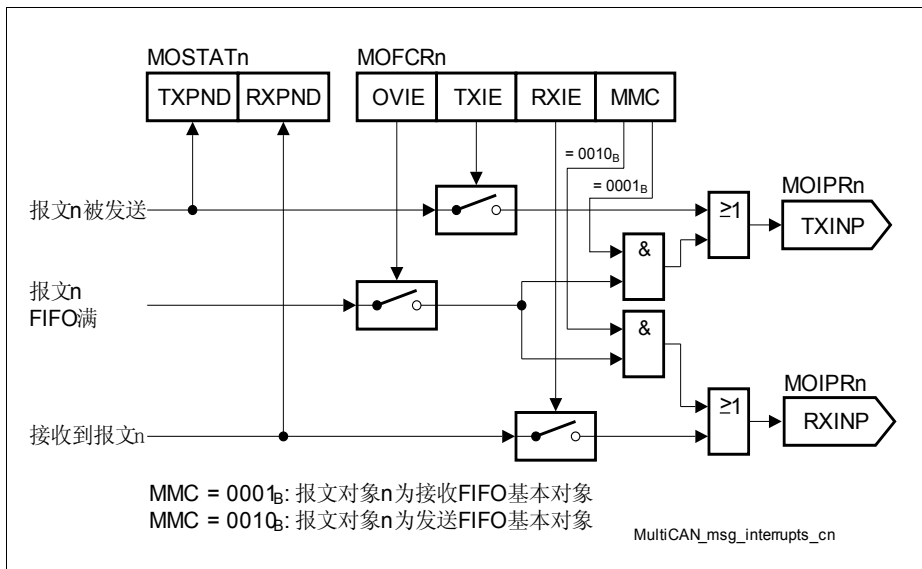


图 16-9 报文中断请求输出线选择

### 16.1.7.2 报文挂起

产生一个报文中断请求的操作将在报文挂起寄存器中置位一个报文挂起位。共有 2 个报文挂起寄存器 MSPNDk (k = 1-0)，每个寄存器有 32 个可用挂起位，因而总共有 64 个挂起位。图 16-10 给出了这些报文挂起位的分配。

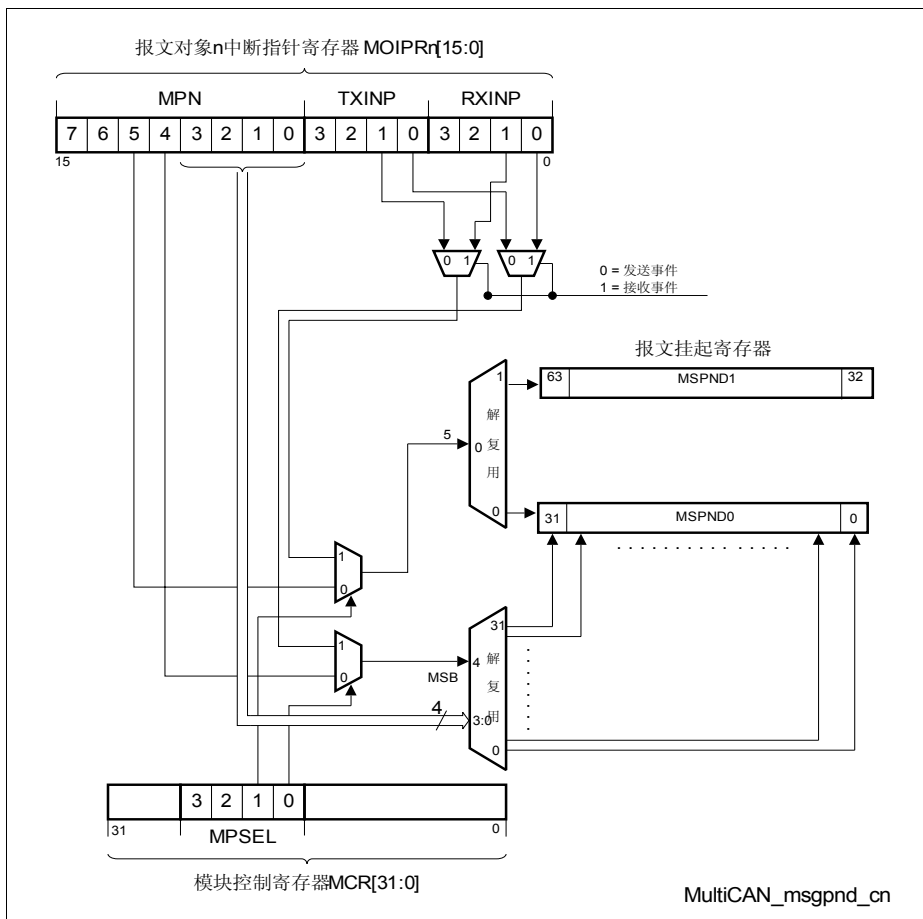


图 16-10 报文挂起位分配

挂起位的位置由两个解复用器定义，通过其选择出 MSPNDk 寄存器编号 k (1 位解复用器)，和在寄存器 MSPNDk 中的位置 (5 位解复用器)。



### 分配情况 1

在这种分配情况下，位域  $\text{MCR.MPSEL} = 0000_{\text{B}}$ 。位置选择由两部分组成：

- $\text{MOIPRn.MPN}$  位 5 ( $\text{MPN}[5]$ ) 选择报文挂起寄存器  $\text{MSPNDk}$  (用于置位挂起位)  $k [k = 1-0]$ 。
- $\text{MOIPRn.MPN}$  的低五位 ( $\text{MPN}[4:0]$ ) 选择  $\text{MSPNDk}$  中要置位的位置 (31-0)。

### 分配情况 2

这种分配情况下，挂起位的分配也需要考虑位域  $\text{MCR.MPSEL}$  的值。位域  $\text{MCR.MPSEL}$  允许挂起位的分配以这样的方法进行，根据接收和发送的情况不同，使用不同目标位置，允许将中断请求节点指针 (接收指针  $\text{MOIPRn.RXINP}$  或者发送指针  $\text{MOIPRn.TXINP}$ ) 包含在内。如果  $\text{MPSEL} = 1111_{\text{B}}$ ，位置选择操作按照如下方法进行：

- 在发送事件中， $\text{TXINP}$  的位 1 定义挂起寄存器  $\text{MSPNDk}$  的编号  $k$ ，在接收事件中， $\text{RXINP}$  的位 1 定义寄存器  $\text{MSPNDk}$  的编号  $k$ 。
- 在相应  $\text{MSPNDk}$  中的位置 (31-0)，由  $\text{TXINP}$  或  $\text{RXINP}$  的最低位和  $\text{MPN}$  的低四位共同选择。

### 一般提示

报文挂起寄存器  $\text{MSPNDk}$  可由软件写入。被写入 1 的位不变，而被写入 0 的位清零。这个特性允许用一个寄存器写操作对单独的  $\text{MSPNDk}$  位清零。因此，当 MultiCAN 模块 (硬件) 置位另一个挂起位时，允许软件同时写访问该寄存器，避免了访问冲突。

每个报文挂起寄存器  $\text{MSPNDk}$  和一个报文指针寄存器  $\text{MSIDk}$  对应。 $\text{MSIDk}$  用来指示报文挂起寄存器  $k$  中所有置 1 的位中位置编号最低的位。 $\text{MSIDk}$  寄存器是只读寄存器，当与其对应报文挂起寄存器  $k$  值改变时，它将立即被更新。

### 16.1.8 报文对象数据处理

本节描述 MultiCAN 模块的报文对象数据处理功能。

#### 16.1.8.1 帧接收

接收一个报文之后，根据图 16-11 中的方案将其存储在报文对象中。MultiCAN 模块不仅将接收到的数据复制到报文对象中，而且还提供其它先进特性以实现 MultiCAN 和 CPU 之间一致的数据交换。

#### MSGVAL

帧接收期间，仅当  $\text{MOSTATn.MSGVAL} = 1$ ，才在报文对象中存储信息。如果位 MSGVAL 被 CPU 复位，MultiCAN 模块将停止所有正在进行的对该报文对象的写访问，因此，可由 CPU 随后的写操作重新配置报文对象，而且不被 MultiCAN 模块打扰。

#### RTSEL

在 CAN 操作期间，当 CPU 重新设置报文对象（例如，清零 MSGVAL，修改报文对象和再次置位 MSGVAL），将发生下列操作：

1. 该报文对象赢得了接收验收滤波
2. CPU 清零 MSGVAL，以重新设置该报文对象
3. 重新设置报文对象后，CPU 再次置位 MSGVAL
4. 到达接收帧尾。因为 MSGVAL 被置位，接收到的数据保存在该报文对象中，产生一个报文中断请求，处理网关和 FIFO 操作，等等。

在重新设置报文对象之后（完成上述 3 步之后），可能不需要存储下面接收的数据。可以通过位  $\text{MOCTRn.RTSEL}$ （“接收 / 发送选择”）来实现上述功能。该位允许报文对象与正在进行的帧接收断开。

当一个报文对象赢得了接收验收滤波，MultiCAN 模块置位 RTSEL 位，用来指示即将进行帧传送。MultiCAN 模块检查 RTSEL 是否因为成功接收一帧而置位，以验证这个报文对象是否仍然准备就绪，可以接收帧。只有在  $\text{RTSEL} = 1$  时，接收到的帧被保存在报文对象中（并进行下面的操作，如报文中断，FIFO 网关操作，标志符更新）。

在 CAN 操作期间，当一个报文对象失效时（MSGVAL 被复位），再次置位 MSGVAL（和置位 MSGVAL 的写操作最接近的）之前，应当清零 RTSEL，以防止将一帧存储在属于报文对象的旧的上下文中。因此，应该按照下边的步骤重新设置报文对象：

1. 清零位 MSGVAL
2. 当  $\text{MSGVAL} = 0$  时，重新设置报文对象
3. 清零位 RTSEL，再次置位 MSGVAL

#### RXEN

位  $\text{MOSTATn.RXEN}$  用于使能该报文对象的帧接收操作。仅当  $\text{RXEN} = 1$  时，报文对象才可以从 CAN 总线上接收 CAN 报文。只有在接收验收滤波期间，MultiCAN 模块才评估

RXEN 的值。接收验收滤波之后，忽略 RXEN，且该位对报文对象中接收报文的实际存储没有后续影响。

位 RXEN 使能所谓的报文对象“软逐步淘汰”：清零 RXEN 之后，报文对象中赢得了验收滤波的当前接收的 CAN 报文仍然存储在报文对象中，但是对于后续的报文，该报文对象将不再进行接收验收滤波。

### **RXUPD, NEWDAT 和 MSGLST**

位 MOSTATn.RXUPD（“接收更新”）用来指示正在进行的帧储存过程。对象更新开始时 RXUPD 置位，结束时清零 RXUPD（包含帧存储和标志符更新）。

在储存接收帧的（标识符，IDE 位，DLC 和数据帧的数据位域）之后，置位 MOSTATn.NEWDAT（“新数据”）。如果再次置位之前 NEWDAT 已经被置位，置位 MOSTATn.MSGLST（“报文丢失”）指示数据的丢失情况。

RXUPD 和 NEWDAT 标志符有助于在 CAN 操作期间保持从报文对象中读取的帧数据的一致性。推荐用户按照下列步骤执行：

1. 清零 NEWDAT 位
2. 从报文对象中读取报文内容（标识符，数据等等）
3. 检查 NEWDAT 和 RXUPD 是否都被清零，如果不是，返回到第一步
4. 因第 3 步成功完成，那么报文对象的内容是一致的，也就是说，报文对象的内容读操作期间没有被 MultiCAN 模块更新

对于数据帧和远程帧接收，位 RXUPD, NEWDAT 和 MSGLST 行为特征相同。

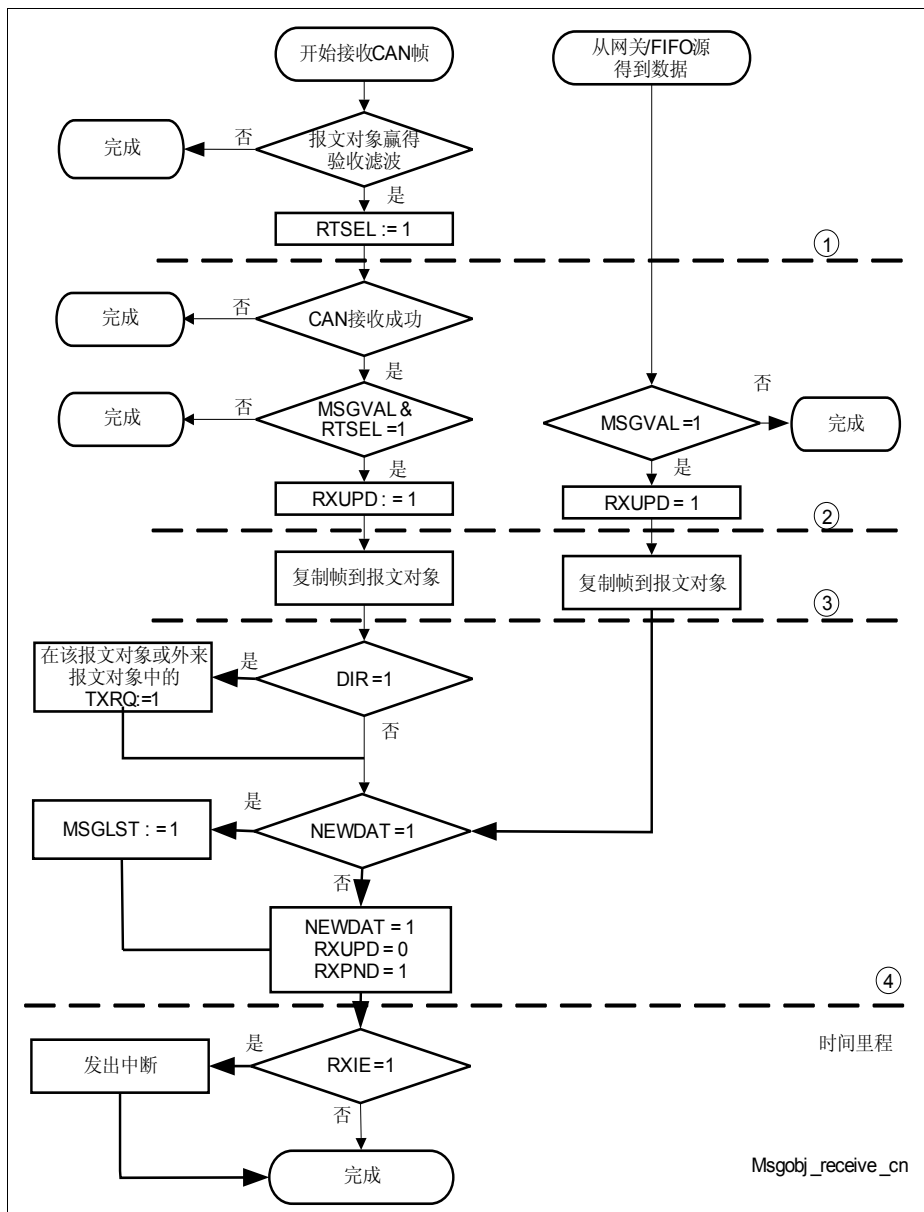


图 16-11 报文对象的接收

### 16.1.8.2 帧发送

帧发送的过程如图 16-12 所示。将要发送的报文内容（标识符，IDE 位，RTR = DIR 位，DLC，和数据帧的数据域）复制到指定的 CAN 节点的内部发送缓存中，同时服务和监控一些状态标志符，用来控制数据处理的一致性。

在发送验收滤波之后，开始报文对象发送过程，远程和数据帧发送过程相同。

#### MSGVAL, TXRQ, TXEN0, TXEN1

仅在寄存器 MOSTATn 的下面 4 位：MSGVAL（“报文有效”）、TXRQ（“发送请求”）、TXEN0（“发送使能 0”）、TXEN1（“发送使能 1”）都置位的情况下，才能发送报文。尽管对于发送过程来说，这些位作用相同，但是它们的定义不同：

表 16-3 报文发送中的位定义

位	描述
MSGVAL	<b>报文有效</b> 该位是报文对象的主控制位
TXRQ	<b>发送请求</b> 为标准发送请求位。只要一个报文对象要进行发送，就必须置位该位。在一次成功发送结束时，除非该报文还有新数据（由 NEWDAT = 1 指示）要发送，否则 TXRQ 被硬件清零。 当置位 MOFCRn.STT（“单次发送试验”），且当报文对象的内容被复制到 CAN 节点的发送帧缓存中时，TXRQ 已经被清零。 接收到的远程请求（在远程帧接收之后）置位 TXRQ，用来请求数据帧发送操作。
TXEN0	<b>发送使能 0</b> 该位可以由软件暂时清零，当报文对象向数据域写新内容时，用来禁止该报文对象的发送。这样可以避免发送由新值和旧值混合在一起的内容不一致的报文帧。 当 TXEN0 = 0 时，仍接受远程请求，但是数据帧的发送被挂起，直到软件再次置位发送使能（置位 TXEN0）。
TXEN1	<b>发送使能 1</b> 发送 FIFO 中，用该位选择发送有效的报文对象。 对于不是发送 FIFO 元素的报文对象，TXEN1 可被永久置 1 或者用作第二独立的发送使能位。

#### RTSEL

在发送验收滤波之后，当已经验证一个报文对象就是下一个要被发送的报文对象时，置位 MOCTRn.RTSEL（“接收 / 发送选择”）。

当报文对象被复制到内部发送缓存中时，检查位 RTSEL，仅在 RTSEL = 1 时，才发送报文。在成功发送报文之后，再次检查位 RTSEL，只有 RTSEL = 1，才会执行报文后处理。

要完成一个有效的报文对象重构，需要执行下面的步骤：

1. 位 MSGVAL 清零；
2. 当 MSGVAL = 0 时，重构报文对象；
3. 清零 RTSEL，置位 MSGVAL

清零 RTSEL 确保报文对象与一个正在进行 / 预定的发送断开。同时确保在报文对象再次变为有效之后，在旧的报文对象上下文之间不出现报文对象的处理（复制报文至发送缓存包括清零 NEWDAT，清零 TXRQ，时间戳更新，报文中断等等），总是在一个新的上下文中进行报文对象处理。

### NEWDAT

当报文对象的内容被传送到 CAN 节点的内部发送缓存中之后，硬件清零位 MOSTATn.NEWDAT（新数据），指示发送报文对象中的数据不再是新值。

当帧发送成功且 NEWDAT 仍然为 0（也就是说，其间没有新数据被复制到报文对象中），硬件自动清零 TXRQ（发送请求）。

然而，如果软件再次置位 NEWDAT（因为要发送新的一帧），不清零 TXRQ，以使能新数据发送。

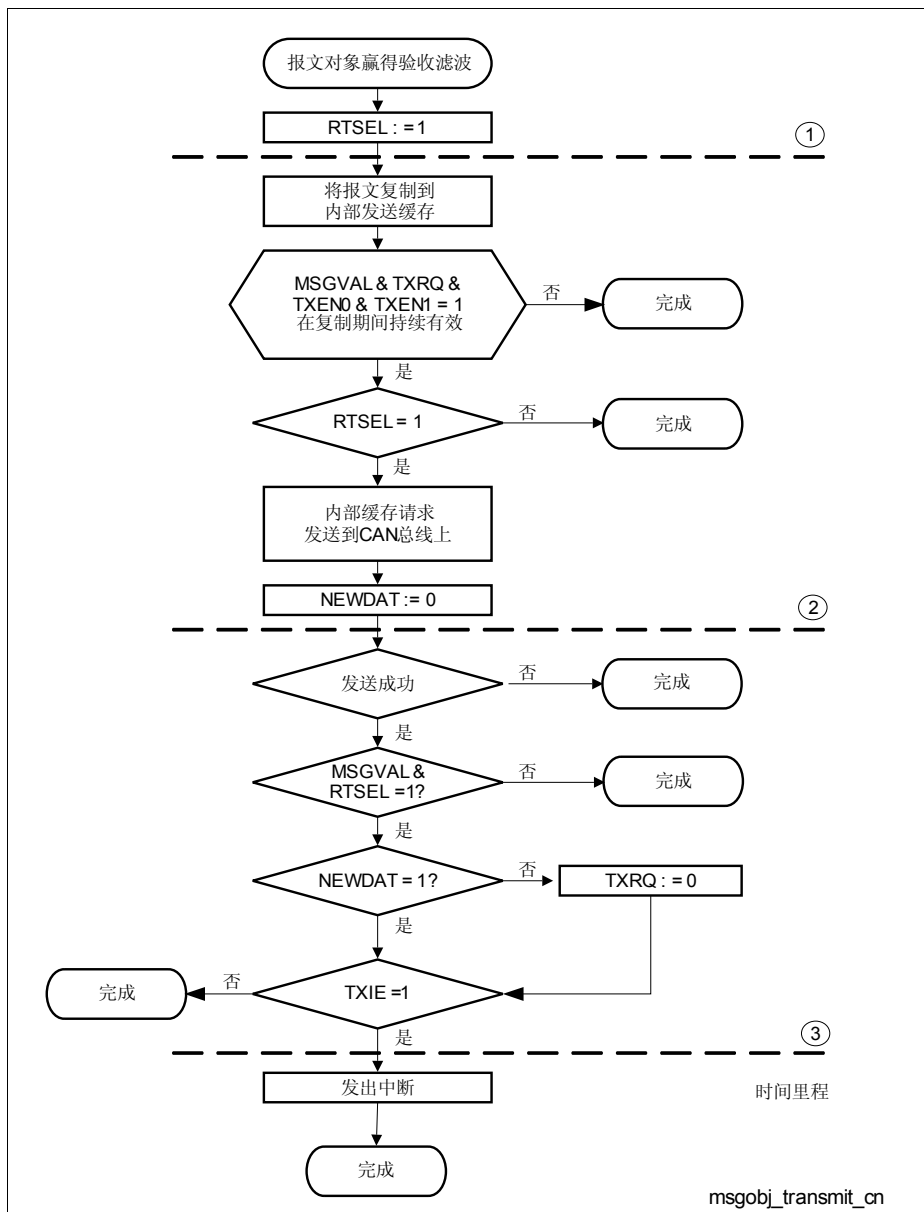


图 16-12 报文对象的发送

### 16.1.9 报文对象功能

本节描述 MultiCAN 模块的报文对象功能。

#### 16.1.9.1 标准报文对象

当位域  $\text{MOFCRn.MMC} = 0000_{\text{B}}$ ，选择标准报文对象。标准报文对象可根据前面各节描述的基本规则发送和接收 CAN 帧。还可以使用如单一数据传送模式或者单次发送试验等附加功能（见下面的章节），且可独立选择这些附加功能。

#### 16.1.9.2 单一数据传送模式

在 CAN 总线上广播数据为了避免发送重复信息，采用单一数据传送模式。通过位  $\text{MOFCRn.SDT}$  选择该模式。

##### 报文接收

当存储在报文对象中的接收报文被新接收到的报文覆盖，第一个报文的内容丢失，且被接收到的新值取代（由  $\text{MSGLST} = 1$  指示）。

单一数据传送模式（ $\text{SDT} = 1$ ），在存储了接收到的数据帧之后，硬件自动清零位  $\text{MSGVAL}$ 。该操作阻止了后续的报文接收。

在接收远程帧之后，位  $\text{MSGVAL}$  不会被自动清零。

##### 报文发送

当接收到一系列远程请求时，报文对象发送几个数据帧来回应远程请求。如果报文对象中的数据在两次发送期间未被更新，同样的数据在 CAN 总线上发送次数可以多于一次。

在单一数据传送模式（ $\text{SDT} = 1$ ），因为在成功发送一个数据帧之后，自动清零  $\text{MSGVAL}$ ，从而避免了数据被再次发送。

在发送远程帧之后，位  $\text{MSGVAL}$  不会被自动清零。

#### 16.1.9.3 单次发送试验

如果置位  $\text{MOFCRn.STT}$ ，当报文对象的帧内容已经被复制到 CAN 节点的内部发送缓存中时，那么发送请求被清零（ $\text{TXRQ} = 0$ ）。因此，当因 CAN 总线错误而使得发送失败的话，不会再次尝试发送报文对象。

#### 16.1.9.4 报文对象 FIFO 结构

CPU 高负荷的情况下，及时的处理一串 CAN 帧可能比较困难。如果必须在短时间内发送或者接收多个报文，就可能发生这种情况。

因此，在 MultiCAN 模块中使用 FIFO 缓存结构来避免丢失接收到的报文，并减少发送报文的建立时间。FIFO 结构也可以用来自动接收或者发送一串 CAN 报文，当整个 CAN 帧序列接收 / 发送结束后，FIFO 产生单报文中断。



也可以构造成几个并行 FIFO，FIFO 的个数和容量仅受可用报文对象个数的限制。无论何时，即使是在 CAN 操作期间，都可以建立，重构其容量和取消 FIFO 结构。

FIFO 基本结构见图 16-13。FIFO 由一个基本报文对象和 n 个从属报文对象组成。从属报文对象由一个列表结构链接在一起（和报文对象列表相似）。基本对象可以分配给任意一个列表。尽管图 16-13 所示的基本对象是在从属对象之外的一部分，其实也可以将基本对象结合到从属对象链表中的任何位置。这意味着：基本对象也是从属对象（这点在网关模式中是不可能的）。报文对象的绝对数目对于 FIFO 的操作没有影响。

不需要将基本对象和从属对象分配给同一个列表。只有从属对象必须分配给同一个列表（因为它们是链接在一起的）。不管基本对象被分配给和从属对象一样的列表或者另一个列表，由寄存器 MOFGPRn 中的几个指针（BOT，CUR 和 TOP）将基本对象和从属对象链接起来。

最小的 FIFO 为一个报文对象，它既是 FIFO 基本对象也是 FIFO 从属对象（这种情况用处不大）。可能的最大 FIFO 结构将包含 MultiCAN 模块的所有报文对象。只能实现容量限制在上述两者之间的 FIFO 结构

In 在 FIFO 基本对象中定义 FIFO 的边界。基本对象的位域 MOFGPRn.BOT 指向（包括编号）FIFO 结构中最底部的从属报文对象。位域 MOFGPRn.TOP 指向（包括编号）FIFO 结构中顶端的从属报文对象。位域 MOFGPRn.CUR 指向（包括编号）实际上被 MultiCAN 模块选中进行报文传送的从属对象。当在该对象中出现了报文传送，将 CUR 设置为列表中下一个从属对象。（CUR = 当前对象的 PNEXT）。如果 CUR 等于 TOP（到达了 FIFO 顶部），CUR 的下次更新将使得 CUR = BOT（从 FIFO 顶部绕回到底部）。这种机制代表了一种环形 FIFO 结构，位域 BOT 和 TOP 建立了 FIFO 中第一个和最后一个元素之间的连接。

基本对象的位域 MOFGPRn.SEL 可被用于监测。这个特性允许在列表中定义一个从属对象，不管何时 CUR 指针到达了 SEL 指针的值，就会在其上产生一个报文中断。因此，SEL 允许检测预先定义的报文传送序列的结束或者当 FIFO 已满时，发出一个警告中断。

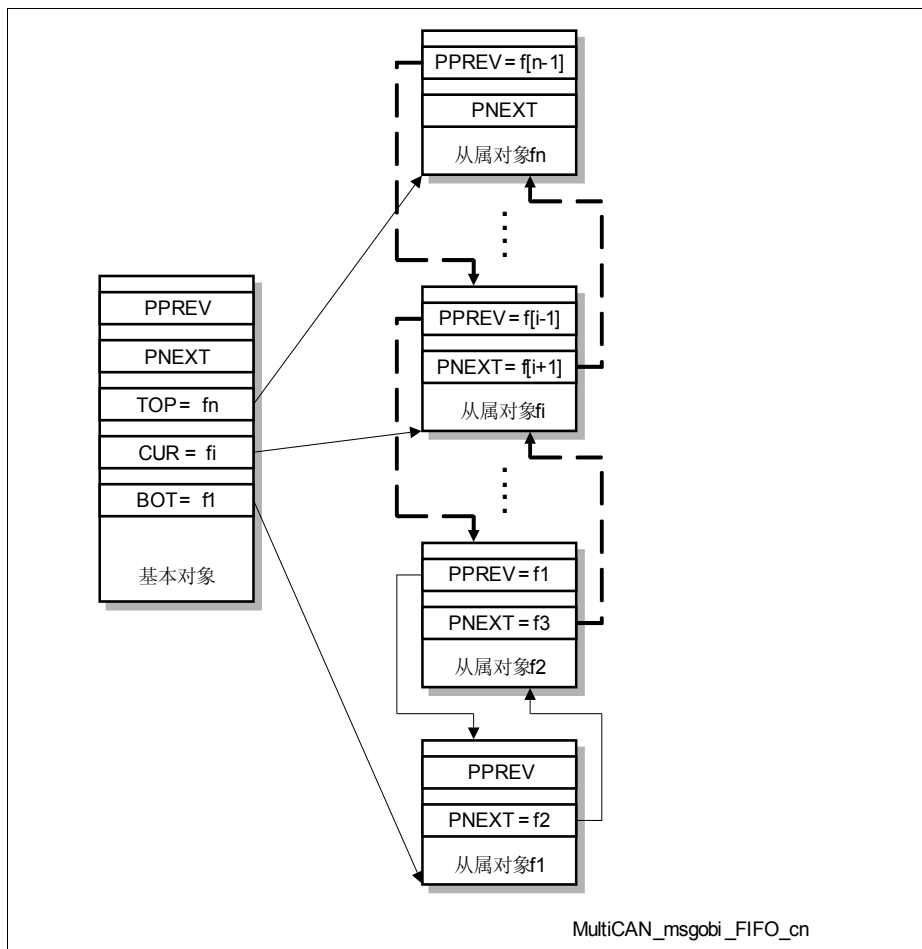


图 16-13 FIFO 结构图 (1 个基本对象, n 个从属对象)

### 16.1.9.5 接收 FIFO

接收 FIFO 结构用于缓存接收到的远程或数据帧。

在 FIFO 基本对象中设置 MOFCRn.MMC = 0001<sub>B</sub>, 选择接收 FIFO。MMC 码自动指定该报文对象为 FIFO 基本对象。FIFO 从属报文对象的报文模式和接收 FIFO 操作无关。

当 FIFO 基本对象从其所属的 CAN 节点接收一帧时, 该帧并不存储在基本对象中, 而是存储在基本对象 MOFGPRn.CUR 指针所选择的从属报文对象中, 就像是由从属报文对象直接接收该帧一样。然而, MOFCRn.MMC = 0000<sub>B</sub> 隐含地假设 FIFO 从属对象执行的是标准报文传送。忽略 FIFO 从属对象实际的报文模式 (MMC 设置)。从属对象不对接收到的帧进行验收滤波, 即不检查匹配标识符, IDE 位和 DIR 位。

接收一个 CAN 帧之后, 基本对象的当前指针 CUR 被设置为 FIFO 结构中的下一个报文对象的编号。这个报文对象将用来储存下一个接收报文。

如果 FIFO 基本对象的位域 MOFCRn.OVIE (“溢出中断使能”) 置位, 当前指针 MOFGPRn.CUR 变成和 MOFGPRn.SEL 相等, 将产生一个 FIFO 溢出中断请求。在接收到的帧存储到从属报文对象中之后, 立即在基本对象的中断节点 TXINP 上产生中断请求。如果 TXIE 被置位, 仍然产生发送中断。

仅在 MSGVAL = 1 时, CAN 报文保存在 FIFO 基本对象和从属对象中。

### 16.1.9.6 发送 FIFO

发送 FIFO 结构用于缓存要发送的一串数据或远程帧。

在 FIFO 基本对象中，通过设置  $MOFCRn.MMC = 0010_B$ ，选择发送 FIFO。分配给发送 FIFO 的从属对象需要明确地设置它们的位域  $MOFCRn.MMC = 0011_B$ ，这点与接收 FIFO 不同。所有从属对象中的 CUR 指针都必须指回到发送 FIFO 基本对象（由软件初始化）。

除了基本对象的 CUR 指针选择的报文对象，其它所有报文对象的  $MOSTATn.TXEN1$ （发送使能 1）都必须用软件清零。CUR 选择的报文对象（从属对象）的  $TXEN1$  必须被置位。CUR（基本对象的）可被初始化为任意一个 FIFO 从属对象。

当给 FIFO 的报文对象设置为有效以启动 FIFO 操作时，必须首先将基本报文对象设置有效标记（ $MSGVAL = 1$ ）。

操作期间，在取消一个发送 FIFO 之前，其从属报文对象必须设置为无效标记（ $MSGVAL = 0$ ）。

发送 FIFO 用所有的 FIFO 元素的  $MOCTRn.TXEN1$  位选择实际要发送的报文。发送验收滤波评估每个报文对象的  $TXEN1$ ，且只有其  $TXEN1$  置位的报文对象可以赢得发送验收滤波。当 FIFO 报文对象已经发送一帧报文，硬件除了执行标准的发送后处理（清零  $TXRQ$ ，发送中断等等），还对  $TXEN1$  位清零，移动 CUR 指针使其指向下一个要发送的报文对象。硬件自动置位下一个报文对象的  $TXEN1$ 。因此， $TXEN1$  就像一个令牌，沿着发送 FIFO 结构移动，用来选择有效的报文元素。

如果置位 FIFO 基本对象的位域  $MOFCRn.OVIE$ （“溢出中断使能”）且当前指针 CUR 和  $MOFGPRn.SEL$  相等，将产生 FIFO 溢出中断请求。对接收到的帧进行后处理之后，在基本对象的中断节点  $RXINP$  上产生中断请求。如果置位  $RXIE$ ，仍然产生发送 FIFO 基本对象的接收中断。

### 16.1.9.7 网关模式

网关模式在两个独立 CAN 总线之间建立自动信息传送，而不需要和 CPU 相互作用。

网关模式在报文对象一级上工作。在网关模式下，信息的传送是在两个报文对象之间进行，结果这两个报文对象所属的 CAN 节点之间也进行了信息传送。网关可建立在任意一对 CAN 节点之间，可以建立网关的个数和可用于构造网关结构的报文对象的个数一样多。

通过设置网关源对象 **s** 的位域 **MOFCRs.MMC = 0100<sub>B</sub>**，选择网关模式。由源报文对象 **s** 的 **MOFGPRd.CUR** 指针选择网关目标报文对象 **d**。目标对象只需要设为有效（其 **MSGVAL = 1**）。所有其它设置都与源和目标对象之间的信息传送无关。

当一个 CAN 帧已经被接收并保存在源对象中时（见图 16-14），除了一些由 MultiCAN 模块执行的附加操作之外，网关源对象和标准报文对象的操作相似。

1. 如果位 **MOFCRs.DLCC** 被置位，将数据长度码 **MOFCRs.DLC** 从网关源对象复制到网关目标对象。
2. 如果位 **MOFCRs.IDC** 被置位，标识符 **MOARs.ID** 和标识符扩展部分 **MOARs.IDE** 从网关源对象复制到网关目标对象。
3. 如果位 **MOFCRs.DATC** 被置位，保存在两个数据寄存器 **MODATALs** 和 **MODATAHs** 的数据被从源对象复制到目标对象。即使 **MOFCRs.DLC** 指示数据少于 8 个字节，所有 8 字节数据都将被复制。
4. 如果位 **MOFCRs.GDFS** 被置位，网关目标对象中的发送请求标志符 **MOSTATd.TXRQ** 被置位。
5. 网关目标对象接收挂起位 **MOSTATd.RXPND** 和新数据位 **MOSTATd.NEWDAT** 都被置位。
6. 如果其 **MOSTATd.RXIE** 被置位，网关目标对象产生报文中断请求。
7. 根据页 16-29 中描述的 FIFO 规则，网关源对象的当前对象指针 **MOFGPRs.CUR** 移动到下一个目标对象。设置 **MOFGPRs.TOP = MOFGPRs.BOT = MOFGPRs.CUR = 目标对象**，可得到单（静态）目标对象网关。

网关源对象和目标对象之间和 FIFO 基本对象和 FIFO 从属对象之间的链接方式相同。这就意味着，可以生成具有 FIFO 结构的目标网关，图 16-13 中左边的报文对象为网关源对象，右边的报文对象为网关目标对象。

网关对接收数据帧（源对象为接收对象，也就是说，**DIR = 0**）的操作和对接收远程帧的操作相同（源对象为发送对象）。

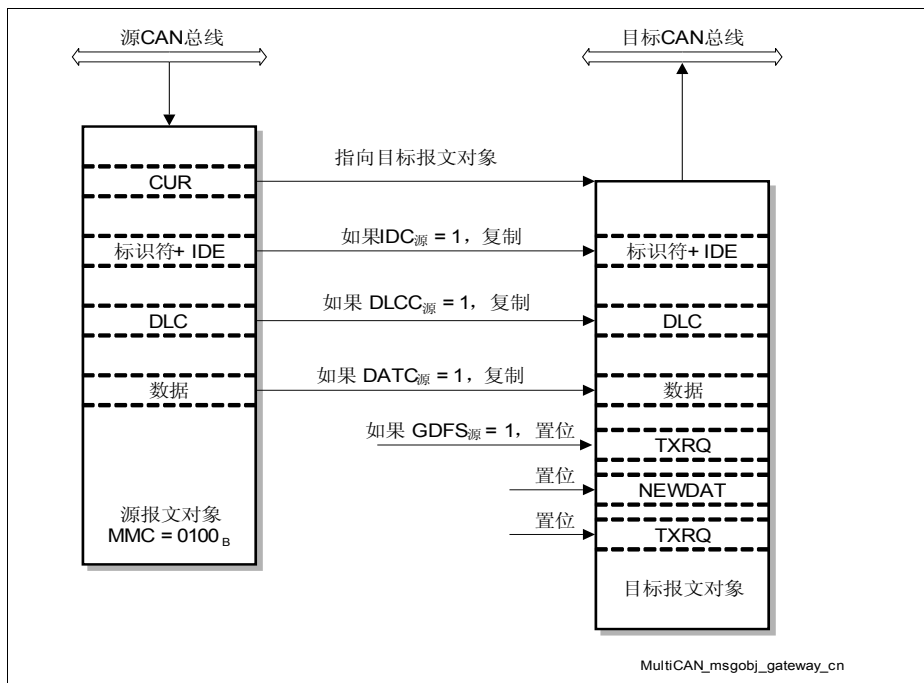


图 16-14 从源对象到目标对象的网关传送

### 16.1.9.8 外来远程请求

当在 CAN 节点上接收到一个远程帧并将其存储在报文对象中时，置位发送请求以触发对该请求的应答（数据帧的发送）或者自动发出第二个请求。如果存储远程请求的报文对象的外来远程请求使能位 **MOFCRn.FRREN** 被清零，同一个报文对象中的 **MOSTATn.TXRQ** 被置位。

如果 **FRREN** 被置位，指针 **MOFGPRn.CUR** 指向的报文对象中的 **TXRQ** 被置位。然而，**CUR** 的值并未被该特性改变。

虽然外来远程请求特性和所选择的报文模式无关，但是对于从网关目标总线上接收一个远程请求之后，在源总线上发出一个远程请求的场合，外来远程请求特性尤其有用。根据网关目标对象中 **FRREN** 设置，目标方具有两个用于处理远程请求的特性（假设源对象接收，目标对象发送，也就是说  $DIR_{源} = 0$  且  $DIR_{目标} = 1$ ）：

#### 网关目标对象中 **FRREN = 0**

1. 网关目标对象接收到一个远程帧
2. 网关目标对象中的 **TXRQ** 被自动置位
3. 保存在目标对象中、带当前数据的数据帧被发送到目标总线上

#### 网关目标对象中 **FRREN = 1**

1. 网关目标对象接收到一个远程帧
2. 网关源对象中的 **TXRQ** 被自动置位（必须是目标对象 **CUR** 指针所指向的）
3. 源对象（接收对象）在源 CAN 总线上发出远程请求
4. 远程请求的接收器以数据帧回应源对象总线上
5. 数据帧存储在源对象中
6. 数据帧复制到目标对象（网关操作）
7. 目标对象的 **TXRQ** 置位（假设  $GDFS_{源} = 1$ ）
8. 保存在目标对象中的新数据被发送到目标总线上，作为目标总线初始远程请求的回应。

### 16.1.10 访问仲裁器

MultiCAN 需要包含最多 16K 字节的 SFR 内核地址范围，比 XC878 所能提供的地址范围要大得多。为满足这个需要，内嵌译码机制的地址扩展单元“访问仲裁器”，对 MultiCAN 内核的 SFR 进行译码。不直接由 CPU 指令本身控制地址线，而是在访问 MultiCAN 内核之前，由编程的寄存器位实现对地址线的控制。

为了对 MultiCAN 内核寄存器的地址进行译码，最少需要 14 条地址线。因为 MultiCAN 寄存器 32 位宽（4 字节），那么地址线 A[1:0] 不需要译码，并被绑定为“00”。地址线 A[13:2] 的实现是由编程寄存器位 CAN\_ADL 中的位 CA2 到 CA9 和 CAN\_ADH 中的位 CA10 到 CA13 实现的。在访问 MultiCAN 寄存器之前，需要对地址寄存器进行编程。

访问仲裁器和 MultiCAN 内核之间的数据总线为 32 位（D[31:0]）。在访问仲裁器中实现了 4 个数据寄存器 CAN\_DATA<sub>n</sub>（n = 3-0）。对 MultiCAN 内核的每个寄存器的读和写操作通过这 4 个数据寄存器进行。

当写访问 MultiCAN 内核时，通过配置寄存器 CAN\_ADCON 中的位 V<sub>n</sub>（n = 3-0）将寄存器 CAN\_DATA<sub>n</sub>（n = 3-0）中的数据设为有效或者无效。在写操作中，仅发送有效数据（字节）。寄存器位 V<sub>n</sub>（n = 3-0）对于读操作没有影响。在读操作期间，将从 MultiCAN 内核读 32 位数据。

寄存器位 CAN\_ADCON.BSY 用于指示发送是否完成。当 BSY 寄存器位被置位时，数据寄存器和地址寄存器不接受任何读 / 写访问。对于 MultiCAN 内核的读 / 写操作只发生在写 CAN\_ADCON 寄存器时。对 MultiCAN 内核的读 / 写操作由位 CAN\_ADCON.RWEN 定义。读 CAN\_ADCON 寄存器对于向 / 从 MultiCAN 内核读 / 写数据的操作无影响。每个向 MultiCAN 的读 / 写操作只读 / 写数据一次。

此外，通过配置位域 CAN\_ADCON.AUAD，还可以实现地址自动增加 / 减少的附加功能。地址可以自动增加 / 减少 1 或 8（当对报文对象编程时，该特性非常有用）。如果使能该功能，在一次读 / 写过程完成之后，地址指针自动指向下一个寄存器地址。地址寄存器 CAN\_ADL 和 CAN\_ADH 也反映地址指针指向的地址。对下一个寄存器的下次读 / 写操作可立即进行，无需再次向地址寄存器 CAN\_ADL 和 CAN\_ADH 写地址。

### MultiCAN 内核的写过程

- 将 MultiCAN 内核寄存器的地址写到 CAN\_ADL 和 CAN\_ADH 寄存器中
- 向 CAN\_DATA0/CAN\_DATA1/CAN\_DATA2/CAN\_DATA3 寄存器写数据
- 写寄存器 CAN\_ADCON，包括设置数据寄存器的有效位和设置寄存器位 RWEN 为 1
- 向 MultiCAN 内核只写一次有效数据。寄存器位 BSY 将变为 1
- 档寄存器位 BSY 变为 0，传送完成

### MultiCAN 内核的读过程

- 将 MultiCAN 内核寄存器的地址写到 CAN\_ADL 和 CAN\_ADH 寄存器
- 写寄存器 CAN\_ADCON，设置寄存器位 RWEN 为 0
- 从 MultiCAN 内核寄存器只读一次 32 位数据，寄存器位 BSY 将变为 1
- 当寄存器位 BSY 变为 0，发送完成
- 从 CAN\_DATA0/CAN\_DATA1/CAN\_DATA2/CAN\_DATA3 寄存器读数据



注： 仅当位 *BSY* 为 0 时，才能读 / 写地址寄存器和数据寄存器。

### 16.1.11 端口控制

端口逻辑控制 MultiCAN 模块和端口 I/O 线之间的相互连接。除了 I/O 控制选择之外，CAN 节点接收输入线的选择由其端口控制寄存器 NPCRx（x = 1-0）中的位域 RXSEL 配置。

表 16-4 给出对于 CAN I/O 线所需要的 I/O 功能，对应位和位域的编程。

表 16-4 CAN I/O 控制选择

端口线	PISEL 寄存器位	寄存器位输入 / 输出控制	I/O
P1.0/RXDC0_0	NPCR0.RXSEL = 000 <sub>B</sub>	P1_DIR.P0 = 0 <sub>B</sub>	输入
P1.1/TXDC0_0	–	P1_DIR.P1 = 1 <sub>B</sub>	输出
		P1_ALTSEL0.P1 = 1 <sub>B</sub>	
		P1_ALTSEL1.P1 = 1 <sub>B</sub>	
P3.4/RXDC0_1	NPCR0.RXSEL = 001 <sub>B</sub>	P3_DIR.P4 = 0 <sub>B</sub>	输入
P3.5/TXDC0_1	–	P3_DIR.P5 = 1 <sub>B</sub>	输出
		P3_ALTSEL0.P5 = 1 <sub>B</sub>	
		P3_ALTSEL1.P5 = 1 <sub>B</sub>	
P1.6/RXDC0_2	NPCR0.RXSEL = 010 <sub>B</sub>	P1_DIR.P6 = 0 <sub>B</sub>	输入
P1.7/TXDC0_2	–	P1_DIR.P7 = 1 <sub>B</sub>	输出
		P1_ALTSEL0.P7 = 1 <sub>B</sub>	
		P1_ALTSEL1.P7 = 1 <sub>B</sub>	
P4.0/RXDC0_3	NPCR0.RXSEL = 011 <sub>B</sub>	P4_DIR.P0 = 0 <sub>B</sub>	输入
P4.1/TXDC0_3	–	P4_DIR.P1 = 1 <sub>B</sub>	输出
		P4_ALTSEL0.P1 = 1 <sub>B</sub>	
		P4_ALTSEL1.P1 = 1 <sub>B</sub>	
P0.1/RXDC1_0	NPCR1.RXSEL = 000 <sub>B</sub>	P0_DIR.P1 = 0 <sub>B</sub>	输入
P0.2/TXDC1_0	–	P0_DIR.P2 = 1 <sub>B</sub>	输出
		P0_ALTSEL0.P2 = 1 <sub>B</sub>	
		P0_ALTSEL1.P2 = 1 <sub>B</sub>	
P3.2/RXDC1_1	NPCR1.RXSEL = 001 <sub>B</sub>	P3_DIR.P2 = 0 <sub>B</sub>	输入
P3.3/TXDC1_1	–	P3_DIR.P3 = 1 <sub>B</sub>	输出
		P3_ALTSEL0.P3 = 1 <sub>B</sub>	
		P3_ALTSEL1.P3 = 1 <sub>B</sub>	

表 16-4 CAN I/O 控制选择

端口线	PISEL 寄存器位	寄存器位输入 / 输出控制	I/O
P1.4/RXDC1_3	NPCR1.RXSEL = 011 <sub>B</sub>	P1_DIR.P4 = 0 <sub>B</sub>	输入
P1.3/TXDC1_3	—	P1_DIR.P3 = 1 <sub>B</sub>	输出
		P1_ALTSEL0.P3 = 1 <sub>B</sub>	
		P1_ALTSEL1.P3 = 1 <sub>B</sub>	

16.1.12 低功耗模式

如果完全不需要 MultiCAN 功能，可关闭其输入时钟最大限度的降低功耗。该模式通过置位寄存器 PMCON1 中的位 CAN\_DIS 来实现。外设时钟管理的具体内容请参见 [章节 8.1.4](#)。

PMCON1

功率模式控制寄存器 1

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	UART1_DIS	CAN_DIS	MDU_DIS	T2_DIS	CCU_DIS	SSC_DIS	ADC_DIS
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
CAN_DIS	5	rw	<b>CAN 禁止请求位，高有效</b> 0 CAN 正常工作（缺省状态） 1 请求禁止 CAN
0	7	r	<b>保留</b> 读操作返回 0；应写入 0。

## 16.2 寄存器描述

以下各节中所描述的所有 MultiCAN 寄存器，在本手册其它章节中引用时需添加模块名前缀“CAN”。

### MultiCAN 内核寄存器概览

MultiCAN 内核寄存器包括三部分：

- 全局模块寄存器
- 节点寄存器，针对每个 CAN 节点  $x$
- 报文对象寄存器，针对每个报文对象  $n$

**表 16-5 Registers Overview 寄存器概述 - MultiCAN 内核寄存器**

寄存器缩写名	寄存器全名	偏移地址 <sup>1)</sup>	描述见
<b>全局模块寄存器</b>			
LISTm	列表寄存器 m	$0100_H + m \times 4_H$	页 16-49
MSPNDk	报文挂起寄存器 k	$0120_H + k \times 4_H$	页 16-50
MSIDk	报文指针寄存器 k	$0140_H + k \times 4_H$	页 16-51
MSIMASK	报文指针屏蔽寄存器	$01C0_H$	页 16-52
PANCTR	面板控制寄存器	$01C4_H$	页 16-43
MCR	模块控制寄存器	$01C8_H$	页 16-47
MITR	模块中断触发寄存器	$01CC_H$	页 16-48
<b>节点寄存器</b>			
NCRx	节点 x 控制寄存器	$0200_H + x \times 100_H$	页 16-53
NSRx	节点 x 状态寄存器	$0204_H + x \times 100_H$	页 16-56
NIPRx	节点 x 中断指针寄存器	$0208_H + x \times 100_H$	页 16-59
NPCRx	节点 x 端口控制寄存器	$020C_H + x \times 100_H$	页 16-61
NBTRx	节点 x 位时序寄存器	$0210_H + x \times 100_H$	页 16-62
NECNTx	节点 x 错误计数寄存器	$0214_H + x \times 100_H$	页 16-64
NFCRx	节点 x 帧计数器寄存器	$0218_H + x \times 100_H$	页 16-65
<b>报文对象寄存器</b>			
MOFCRn	报文对象 n 功能控制寄存器	$1000_H + n \times 20_H$	页 16-77
MOFGPRn	报文对象 n FIFO/ 网关指针寄存器	$1004_H + n \times 20_H$	页 16-80

**表 16-5 Registers Overview 寄存器概述 - MultiCAN 内核寄存器**

寄存器缩写名	寄存器全名	偏移地址 <sup>1)</sup>	描述见
MOIPRn	报文对象 n 中断指针寄存器	$1008_H + n \times 20_H$	页 16-75
MOAMRn	报文对象 n 验收屏蔽寄存器	$100C_H + n \times 20_H$	页 16-81
MODATALn	报文对象 n 数据寄存器低位	$1010_H + n \times 20_H$	页 16-85
MODATAHn	报文对象 n 数据寄存器高位	$1014_H + n \times 20_H$	页 16-86
MOARN	报文对象 n 仲裁寄存器	$1018_H + n \times 20_H$	页 16-82
MOCTRn	报文对象 n 控制寄存器	$101C_H + n \times 20_H$	页 16-69
MOSTATn	报文对象 n 状态寄存器		页 16-72

1) 参数 m, k, x 和 n 的有效范围是:  $m = 7-0$ ,  $k = 1-0$ ,  $x = 1-0$ ,  $n = 31-0$

## MultiCAN 访问仲裁寄存器概览

表 16-6 给出 MultiCAN 访问仲裁 SFR 的地址（非映射）。

**表 16-6 MultiCAN 寄存器映射**

寄存器名称	物理地址	描述见
CAN_DATA3	DE <sub>H</sub> （非映射）	页 16-89
CAN_DATA2	DD <sub>H</sub> （非映射）	页 16-89
CAN_DATA1	DC <sub>H</sub> （非映射）	页 16-89
CAN_DATA0	DB <sub>H</sub> （非映射）	页 16-88
CAN_ADH	DA <sub>H</sub> （非映射）	页 16-88
CAN_ADL	D9 <sub>H</sub> （非映射）	页 16-88
CAN_ADCON	D8 <sub>H</sub> （非映射）	页 16-87

图 16-15 给出 MultiCAN 内核寄存器映射地址图。

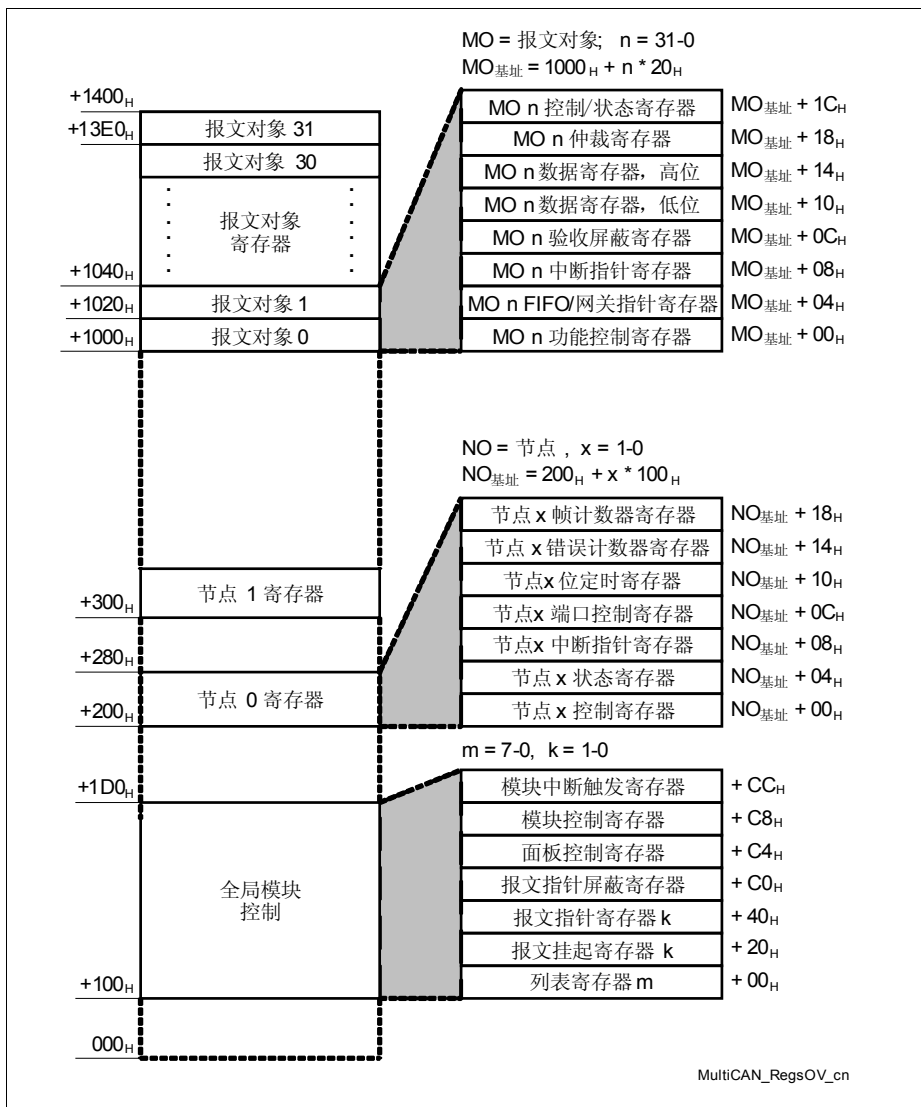


图 16-15 MultiCAN 内核寄存器地址映射图

## 16.2.1 全局模块寄存器

所有列表操作，如列表结构内的报文对象的分配、解除分配、再分配，都是通过命令面板执行的。不可能由软件通过向报文对象和列表寄存器进行写操作的方式直接修改列表结构。

通过向寄存器 PANCTR 的位域写命令码和命令参数启动一个新命令。

### PANCTR

面板控制寄存器

复位值：0000 0301<sub>H</sub>

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PANAR2								PANAR1							
rwh								rwh							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0						RBUSY	BUSY	PANCMD							
r						rh	rh	rwh							

符号	位	类型	描述
PANCMD	[7:0]	rwh	<b>面板命令</b> 将面板命令码写入该位域，可启动一个新命令。在一个面板命令结束时，NOP（无操作）命令码被自动写入该位域。PANCMD 编码定义见表 16-7。
BUSY	8	rh	<b>面板忙标志</b> 0 面板已完成命令，准备好接受新命令 1 面板操作正在进行
RBUSY	9	rh	<b>结果忙标志</b> 0 列表控制器没有安排更新 PANAR1 和 PANAR2 1 正在运行列表命令（BUSY = 1），将向 PANAR1 和 PANAR2 写结果，但结果目前还不可用
PANAR1	[23:16]	rwh	<b>面板参数 1</b> 见表 16-7。
PANAR2	[31:24]	rwh	<b>面板参数 2</b> 见表 16-7。
0	[15:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 面板命令

面板操作命令由一个命令码（PANCMD）和两个面板参数（PANAR1， PANAR2）组成。带返回值的命令将返回值传递至位域 PANAR1。返回错误标志符的命令将错误标志符传递至面板控制寄存器位 31，也就是 PANAR2 位 7。

**表 16-7 面板命令**

PANCMD	PANAR2	PANAR1	命令描述
00 <sub>H</sub>	—	—	<b>无操作</b> 向 PANCMD 写 00 <sub>H</sub> 无任何影响。不启动新命令。
01 <sub>H</sub>	<b>结果：</b> 位 7: ERR 位 6-0: 未定义	—	<b>列表初始化</b> 运行初始化序列，复位所有报文对象的位域 CTRL 和 LIST。列表寄存器 LIST[7:0] 设置至复位值。这将解除所有报文对象的分配。 初始化命令需要所有 CAN 节点 (x = 0-1) 的位 NCRx.INIT 和位 NCRx.CCE 被置位。 PANAR2 的位 7（ERR）报告该操作是否成功： 0     初始化成功 1     并非全部 NCRx.INIT 和 NCRx.CCE 都被置位。因此不执行初始化操作。 每次复位 MultiCAN 模块，都将自动执行列表命令表明命令初始化操作，但是所有报文对象被复位的情况例外。
02 <sub>H</sub>	<b>参数：</b> 列表指针	<b>参数：</b> 报文对象编号	<b>静态分配</b> 为报文对象分配列表。将报文对象从当前所属的列表中除去，并添加到 PANAR2 给出的列表的最后。 该命令还可用于解除报文对象的分配。在这种情况下，目标列表是未被分配报文元素的列表（PANAR2 = 0）。

表 16-7 面板命令

PANCMD	PANAR2	PANAR1	命令描述
03 <sub>H</sub>	<b>参数:</b> 列表指针 <b>结果:</b> 位 7: ERR 位 6-0: 未定义	<b>结果:</b> 报文对象编号	<b>动态分配</b> 将未分配报文对象列表中的第一个报文对象分配给所选择的列表。报文对象附加到列表的最后。报文对象的编号返回给 PANAR1。 <b>ERR 位（PANAR2 位 7）</b> 报告操作是否成功: 0 成功 1 未被分配的报文对象列表为空，因此未执行该操作
04 <sub>H</sub>	<b>参数:</b> 目标对象编号	<b>参数:</b> 源对象编号	<b>静态插入到目标对象之前</b> 将一个报文对象（源对象）从当前所属的列表中除去，并插入到给定目标对象所属的列表结构中，位置在给定目标对象之前。 源对象因而成为目标对象的前一个报文对象。
05 <sub>H</sub>	<b>参数:</b> 目标对象编号 <b>结果:</b> 位 7: ERR 位 6-0: 未定义	<b>结果:</b> 被插入的对象编号	<b>动态插入到给定目标对象之前</b> 在给定目标对象之前插入新的一个报文对象。新报文对象从未被分配的报文元素列表中取出（选择第一个元素）。新报文对象的编号作为 PANAR1 的结果。 <b>ERR 位（PANAR2 位 7）</b> 报告操作是否成功: 0 成功 1 未被分配的报文对象列表为空，因此未执行该操作。
06 <sub>H</sub>	<b>参数:</b> 目标对象编号	<b>参数:</b> 源对象编号	<b>静态插入到目标对象之后</b> 将一个报文对象（源对象）从当前所属的列表中除去，并插入到给定目标对象所属的列表结构中，位置在给定的目标对象之后。 源对象因而成为了目标对象的后一个报文对象。



表 16-7      面板命令

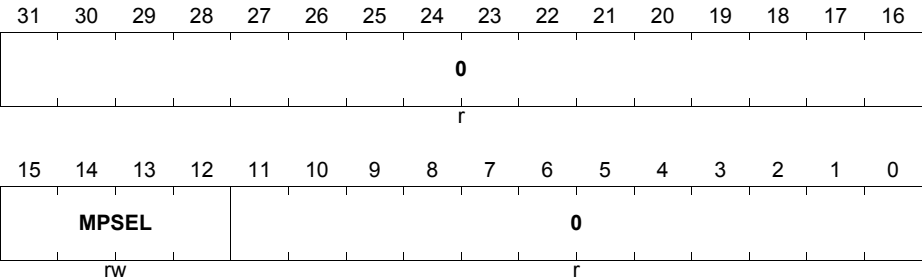
PANCMD	PANAR2	PANAR1	命令描述
07 <sub>H</sub>	<b>参数:</b> 目标对象编号 <b>结果:</b> 位 7: ERR 位 6-0: 未定义	<b>结果:</b> 被插入的对 象编号	<b>动态插入到给定目标对象之后</b> 在给定目标对象之后插入新报文对象。 从未分配的报文元素列表中取出新报文 对象（选择第一个元素）。新报文对象 的编号作为 PANAR1 的结果。 <b>ERR 位（PANAR2 位 7）</b> 报告操作是否 成功。 0      成功 1      未被分配的报文对象列为空，因此 未执行该操作。
08 <sub>H</sub> - FF <sub>H</sub>	—	—	保留

模块控制寄存器 MCR 包含用来定义 MultiCAN 模块操作的基本设置。

MCR

模块控制寄存器

复位值: 0000 0000<sub>H</sub>



符号	位	类型	描述
MPSEL	[15:12]	rw	报文挂起选择器 在报文接收 / 发送之后，位域 MPSEL 可用于选择报文挂起位位置，由 MOIPRn 寄存器位域 RXINP, TXINP, 和 MPN 共同选择。详见 页 16-21 的 图 16-10。
0	[31:16], [11:0]	r	保留 读操作返回 0；应写入 0。



## 列表指针和列表寄存器

每个 CAN 节点都有一个用来定义已经被分配的报文对象列表。还有一个未被分配的报文对象列表。此外，还有一个不与 CAN 节点相连的报文对象通用列表可供使用。列表寄存器的分配方法如下：

- LIST0 定义所有未被分配的报文对象的列表
- LIST1 定义 CAN 节点 0 的列表
- LIST2 定义 CAN 节点 1 的列表
- LIST[7:3] 不和 CAN 节点相联系（自由列表）

### LIST0

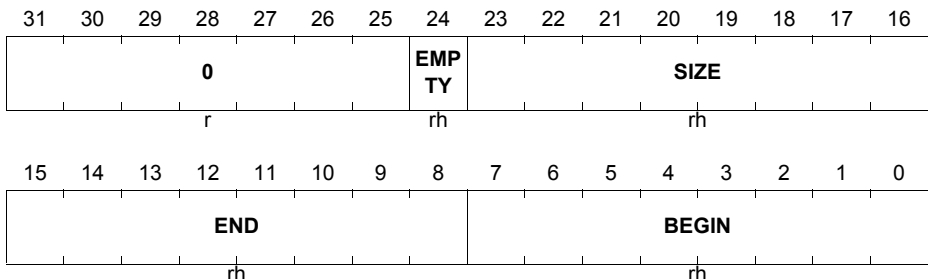
列表寄存器 0

复位值：001F 1F00<sub>H</sub>

LISTm (m = 1-7)

列表寄存器 m

复位值：0100 0000<sub>H</sub>



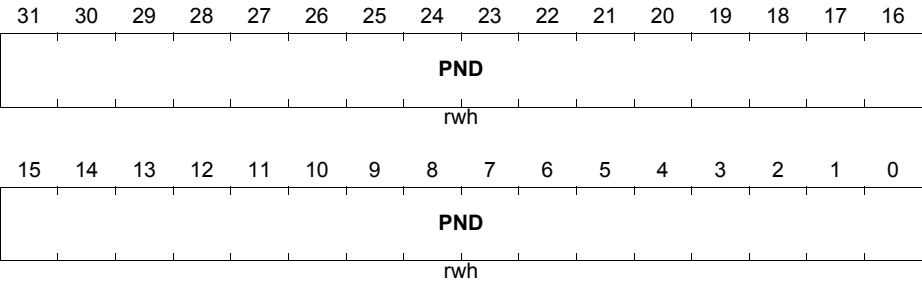
符号	位	类型	描述
<b>BEGIN</b>	[7:0]	rh	列表始端 BEGIN 指出列表 m 中第一个报文对象编号。
<b>END</b>	[15:8]	rh	列表末端 END 指出列表 m 中最后一个报文对象编号。
<b>SIZE</b>	[23:16]	rh	列表大小 SIZE 指示列表 m 中元素个数。 SIZE = 列表中的元素个数 - 1
<b>EMPTY</b>	24	rh	列表为空指示 0 至少给列表 m 分配了一个报文对象 1 未给列表 m 分配报文对象，列表 m 为空
<b>0</b>	[31:25]	r	保留 读操作返回 0；应写入 0。

# 报文通知

当报文对象  $n$  产生一个发送 / 接收报文的 interrupt 请求时, 该请求将被发送至报文对象  $n$  的位域  $MOIPRn.TXIPND$  或者  $MOIPRn.RXIPND$  所选择的 interrupt 输出线上。因为报文对象个数多于 interrupt 输出线个数, 因此典型的一个 interrupt 程序可处理多于一个报文对象请求。因此, MultiCAN 模块执行一个优先级选择机制, 在报文对象集合中选择具有最高优先级的报文对象。

报文挂起寄存器  $MSPNDk$  包含列表  $m$  的挂起 interrupt 通知。

**MSPNDk ( $k = 0-1$ )**  
**报文挂起寄存器  $k$**  复位值: 0000 0000<sub>H</sub>



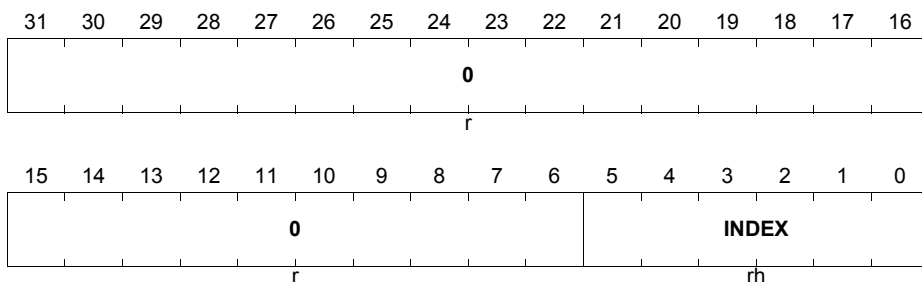
符号	位	类型	描述
PND	[31:0]	rwh	<b>报文挂起</b> 当出现一个报文中断时, 报文对象置位 MSPND 寄存器其中一位, 该位的位置由报文对象的 IPR 寄存器 MPN[4:0] 位域给出。寄存器选择 $k$ 由 MPN 位 5 给出。 寄存器由软件清零 (写 0), 写 1 无影响。

每个报文挂起寄存器和一个报文指针寄存器相关联。报文指针寄存器给出一组挂起位中位置最低的有效（置位的）挂起位。

### MSIDk (k = 0-1)

#### 报文指针寄存器 k

复位值: 0000 0020<sub>H</sub>



符号	位	类型	描述
<b>INDEX</b>	[5:0]	rh	<b>报文挂起指针</b> INDEX 的值由具有下列特性 MSPNDk 挂起位的位置 i 给出: 1. MSPNDk[i] & IM[i] = 1 2. i = 0 或 MSPNDk[i-1:0] & IM[i-1:0] = 0 如果 MSPNDk 中没有满足上述条件的位, 那么 INDEX 被读为 100000 <sub>B</sub> 。 因而, INDEX 给出 MSPNDk 中的第一个挂起位的位置, 只考虑那些被报文指针屏蔽寄存器选中的位。
<b>0</b>	[31:6]	r	<b>保留</b> 读操作返回 0; 应写入 0。

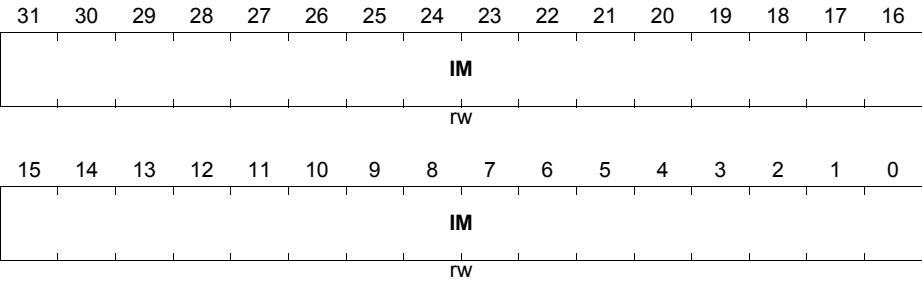
MultiCAN

报文指针屏蔽寄存器 MSIMASK 为计算报文挂起指针选择屏蔽位。所有的报文挂起寄存器及相关的报文指针寄存器共用 MSIMASK 寄存器。

MSIMASK

报文指针屏蔽寄存器

复位值: 0000 0000<sub>H</sub>



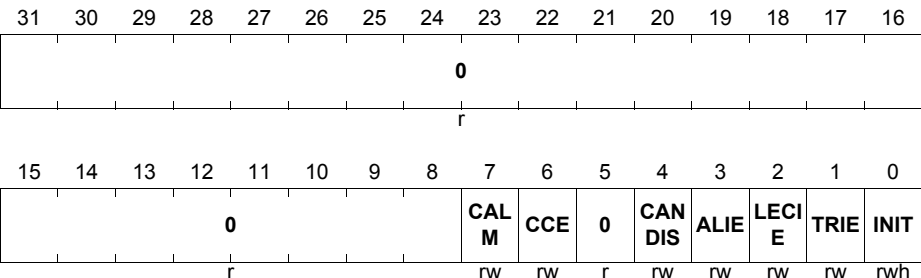
符号	位	类型	描述
IM	[31:0]	rw	报文指针屏蔽 只有 MSPNDk 中那些对应指针屏蔽位被置位的位才用于计算报文指针。

### 16.2.2 CAN 节点寄存器

MultiCAN 模块的每个 CAN 节点配有 CAN 节点寄存器。它们包含直接和 CAN 节点操作相关的信息，这些信息在节点间共享。

节点控制寄存器 NCRx 包含用来确定 CAN 节点操作的基本设置。

**NCRx (x = 0-1)**  
**节点 x 控制寄存器** **复位值: 0000 0001<sub>H</sub>**



符号	位	类型	描述
INIT	0	rwh	<b>节点初始化</b>
			<p><b>0</b>    复位 INIT 用于使能该节点，使其可参加 CAN 通信。</p> <p>如果 CAN 节点处于总线关闭状态，那么继续进行总线关闭的恢复操作（该操作不依赖于 INIT 位）。总线关闭恢复序列结束时，允许该节点参加 CAN 通信。</p> <p>如果 CAN 节点不处于总线关闭状态，在允许该节点参加 CAN 通信之前，必须要检测到 11 个连续的隐性位。</p> <p><b>1</b>    置位该位将终止该 CAN 节点的 CAN 通信。取消任何正在进行的帧传送，发送线变为隐性。如果该节点处于总线关闭状态，那么继续运行总线关闭恢复序列，如果在成功完成总线关闭恢复序列之后，位 INIT 仍然为 1，即在检测到 128 序列 11 个连续隐性位（11 × 1），CAN 节点退出总线关闭状态，但只要 INIT 为 1，总线仍然无效。</p> <p>当 CAN 节点进入总线关闭状态，位 INIT 自动置位。</p>



符号	位	类型	描述
<b>TRIE</b>	1	rw	<b>传送中断使能</b> TRIE 使能 CAN 节点 x 传送中断。节点 x 成功接收或发送 CAN 帧之后产生中断请求。 0 禁止传送中断 1 使能传送中断 通过位域 NIPRx.TRINP 选择由该中断激活的中断输出线。
<b>LECIE</b>	2	rw	<b>LEC 指示错误中断使能</b> LECIE 使能 CAN 节点 x 的最近错误码中断。每次更新 NSRx.LEC 位域使 LEC > 0 (CAN 协议错误) 都会产生中断请求。 0 禁止最近错误码中断 1 使能最近错误码中断 通过 NIPRx.LECINP 选择由该类型中断激活的中断输出线。
<b>ALIE</b>	3	rw	<b>中断警报使能</b> ALIE 使能 CAN 节点 x 警报中断。下列事件之一将产生警报中断： <ul style="list-style-type: none"> <li>• NSRx.BOFF 改变</li> <li>• NSRx.EWRN 改变</li> <li>• 列表长度错误，该错误也置位 NSRx.LLE</li> <li>• 列表对象错误，该错误也置位 NSRx.LOE</li> <li>• 硬件置位 INIT</li> </ul> 0 禁止警报中断 1 使能警报中断 在由 CAN 节点中断指针寄存器位 ALINP 选择的中断节点上产生中断请求。
<b>CANDIS</b>	4	rw	<b>禁止 CAN 节点</b> 置位该位将禁止该 CAN 节点。首先 CAN 节点一直等待，直到总线空闲或总线关闭状态。然后 INIT 被自动置位，如果位 ALIE 被置位，那么产生一个警报中断。
<b>CCE</b>	6	rw	<b>配置改变使能</b> 0 位时序寄存器、端口控制寄存器和错误计数器寄存器可以被读取，忽略所有试图修改这些寄存器的操作。 1 位时序寄存器、端口控制寄存器和错误计数器寄存器可读也可写。

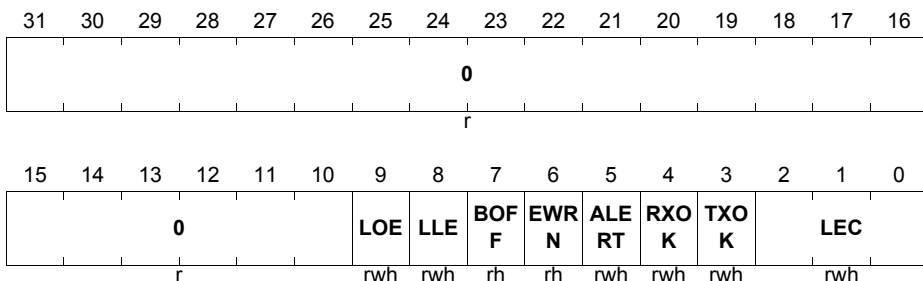
符号	位	类型	描述
<b>CALM</b>	7	rw	<b>CAN 分析模式</b> 如果该位被置位，那么 CAN 节点工作在分析模式。这就意味着可以接收报文，但不能发送报文。对于帧接收并不向 CAN 总线发送应答。有效错误标志以隐性而不是显性发送。发送线连续地保持为隐性 (1) 电平。只有当位 INIT 置位时，才能对位 CALM 写入。
<b>0</b>	[31:8], 5	r	<b>保留</b> 读操作返回 0；应写入 0。

节点状态寄存器 NSRx 报告 CAN 帧传送的成功或者错误信息。

### NSRx (x = 0-1)

#### 节点 x 状态寄存器

复位值: 0000 0000<sub>H</sub>



符号	位	类型	描述
LEC	[2:0]	rwh	<b>最近错误码</b> 该位域给出最近一次 CAN 错误的类型。位域编码的具体内容见 <b>表 16-8</b> 。
TXOK	3	rwh	<b>报文发送成功</b> 0 自从该标志最近一次被复位，无成功的发送操作 1 已经成功发送了一个报文（无错误并且得到至少另外一个节点应答） TXOK 必须由软件复位（写 0），写 1 无任何影响。
RXOK	4	rwh	<b>报文接收成功</b> 0 自从该标志最近一次被复位，无成功的接收操作 1 已经成功接收了一个报文 RXOK 必须由软件复位（写 0）。写 1 无任何影响。
ALERT	5	rwh	<b>警报警告</b> 出现下列事件之一将置位 ALERT（如果 NCRx.ALIE 被置位，这些事件也会触发警报中断）： <ul style="list-style-type: none"> <li>位 NSRx.BOFF 的改变</li> <li>位 NSRx.EWRN 的改变</li> <li>列表长度错误，该错误还置位 NSRx.LLE</li> <li>列表对象错误，该错误还置位 NSRx.LOE</li> <li>硬件已经置位 INIT</li> </ul> ALERT 必须由软件复位（写 0），写 1 无影响。

符号	位	类型	描述
<b>EWRN</b>	6	rh	<b>错误警告状态</b> 0 没有超过警告界限 1 错误计数器 NECNTx.REC 或 NECNTx.TEC 之一达到了警告界限 NECNTx.EWRNLVL.
<b>BOFF</b>	7	rh	<b>总线关闭状态</b> 0 CAN 控制器不处于总线关闭状态 1 CAN 控制器处于总线关闭状态
<b>LLE</b>	8	rwh	<b>列表长度错误</b> 0 自从该标志最近一次被复位，未出现列表长度错误 1 报文验收滤波过程中，检测到一个列表长度错误，属于这个 CAN 节点的列表 中的元素个数和列表终止指针给出的列表大小不同 LLE 必须由软件复位（写 0），写 1 无任何影响。
<b>LOE</b>	9	rwh	<b>列表对象错误</b> 0 自从该标志最近一次被复位，未出现列表对象错误。 1 报文验收滤波过程中，检测到一个列表 对象错误。已经检测到在报文对象控制寄存器中带有错误列表指针的报文对象 LOE 必须由软件复位（写 0），写 1 无任何影响。
<b>0</b>	[31:10]	r	<b>保留</b> 读操作返回 0；应写入 0。

## LEC 位域编码

表 16-8 LEC 位域编码

LEC 值	含义
000 <sub>B</sub>	<b>无错误：</b> 在 CAN 总线上的最近一个报文中没有检测到错误。
001 <sub>B</sub>	<b>填充错误：</b> 在接收到报文的一部分中，一个序列出现了多于 5 个相等的位，而这是不允许的。
010 <sub>B</sub>	<b>格式错误：</b> 接收到的报文帧“固定格式部分”出现格式错误。
011 <sub>B</sub>	<b>应答错误：</b> 发送出去的报文未被另一个 CAN 节点应答。

表 16-8      LEC 位域编码

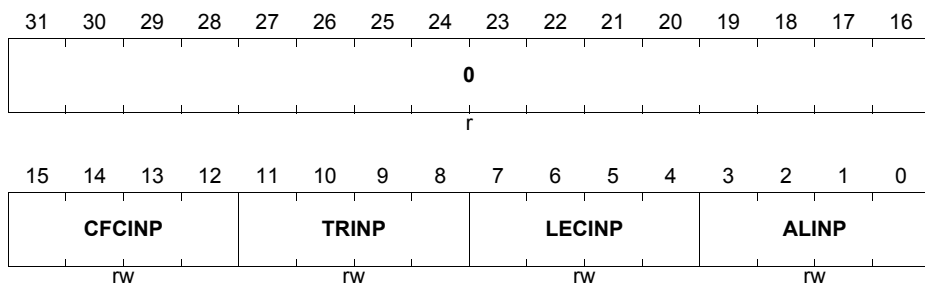
LEC 值	含义
100 <sub>B</sub>	<b>位 1 错误:</b> 在发送报文过程中，CAN 节点试着在仲裁域和应答时隙之外发送隐性电平（1），但是监测到的总线值为显性。
101 <sub>B</sub>	<b>位 0 错误:</b> 该编码指示两种不同情况： 1. 在报文发送过程中（或者应答位，有效错误标志，过载标志），CAN 节点试着发送显性电平（0），但监测到的总线值是隐性。 2. 总线关闭恢复期间，每次监测到 11 个连续的隐性位，将置位该编码。CPU 可以用这个编码指示总线未被连续地扰乱。
110 <sub>B</sub>	<b>CRC 错误:</b> 收到的报文 CRC 校验和不正确。
111 <sub>B</sub>	<b>CPU 写 LEC:</b> CPU 向 LEC 写 111 <sub>B</sub> ，则实际值为 111 <sub>B</sub> ；CPU 向 LEC 写其它值时，写入值被忽略。

NIPR 寄存器中的四个中断指针分别为每种 CAN 节点中断选择中断输出线（共 8 条中断输出线）。CAN 节点中断的具体内容见页 16-9。

### NIPRx (x = 0-1)

节点 x 中断指针寄存器

复位值：0000 0000<sub>H</sub>

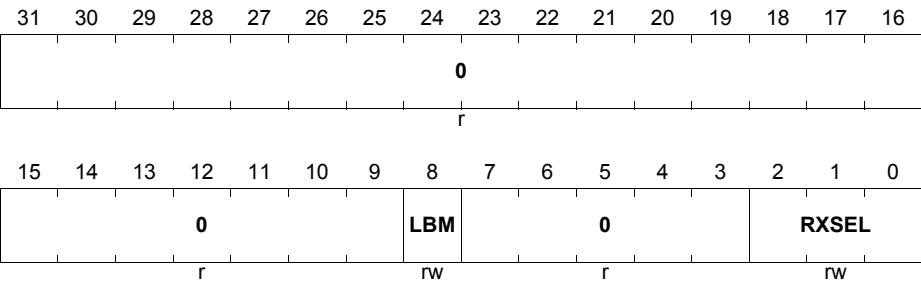


符号	位	类型	描述
<b>ALINP</b>	[3:0]	rw	<b>中断节点指针警报</b> ALINP 为 CAN 节点 x 警报中断选择中断输出线 CANSRCm (m = 0-7) 0000 <sub>B</sub> 选择中断输出线 CANSRC0 0001 <sub>B</sub> 选择中断输出线 CANSRC1 ... 0111 <sub>B</sub> 选择中断输出线 CANSRC7 1000 <sub>B</sub> -1111 <sub>B</sub> 保留
<b>LECINP</b>	[7:4]	rw	<b>最近错误码中断节点指针</b> LECINP 为 CAN 节点 x 的最近错误码中断选择中断输出线 CANSRCm (m = 0-7)。 0000 <sub>B</sub> 选择中断输出线 CANSRC0 0001 <sub>B</sub> 选择中断输出线 CANSRC1 ... 0111 <sub>B</sub> 选择中断输出线 CANSRC7 1000 <sub>B</sub> -1111 <sub>B</sub> 保留

符号	位	类型	描述
<b>TRINP</b>	[11:8]	rw	<b>传送完成中断节点指针</b> TRINP 为 CAN 节点 x 传送完成中断选择中断输出线 CANSRCm (m = 0-7)。 0000 <sub>B</sub> 选择中断输出线 CANSRC0 0001 <sub>B</sub> 选择中断输出线 CANSRC1 ...            ... 0111 <sub>B</sub> 选择中断输出线 CANSRC7 1000 <sub>B</sub> -1111 <sub>B</sub> 保留
<b>CFCINP</b>	[15:12]	rw	<b>帧计数器中断节点指针</b> CFCINP 为 CAN 节点 x 的帧计数器溢出中断选择中断输出线 CANSRCm (m = 0-7)。 0000 <sub>B</sub> 选择中断输出线 CANSRC0 0001 <sub>B</sub> 选择中断输出线 CANSRC1 ...            ... 0111 <sub>B</sub> 选择中断输出线 CANSRC7 1000 <sub>B</sub> -1111 <sub>B</sub> 保留
<b>0</b>	[31:16]	r	<b>保留</b> 读操作返回 0；应写入 0。

节点端口控制寄存器 NPCRx 用于配置 CAN 总线发送 / 接收端口。只有在置位 NCRx.CCE 时，才能向 NPCRx 写入。

**NPCRx (x = 0-1)**  
 节点 x 端口控制寄存器 复位值: 0000 0000<sub>H</sub>



符号	位	类型	描述
<b>RXSEL</b>	[2:0]	rw	<b>接收选择</b> RXSEL 从 8 条可能的接收输入线中选择一个。只能通过所选择的输入执行 CAN 接收操作。 <i>注：在 XC878 中，仅给出可用的 RXSEL 组合（见页 16-38）。</i>
<b>LBM</b>	8	rw	<b>回环模式</b> 0 禁止回环模式 1 使能回环模式。该节点和一个内部（虚拟）回环总线相连。所有处于回环模式的 CAN 节点和该虚拟 CAN 总线相连，因此，它们之间可进行内部通信。在回环模式下，外部发送线被强制为隐性电平
<b>0</b>	[7:3], [31:9]	r	<b>保留</b> 读操作返回 0；应写入 0。

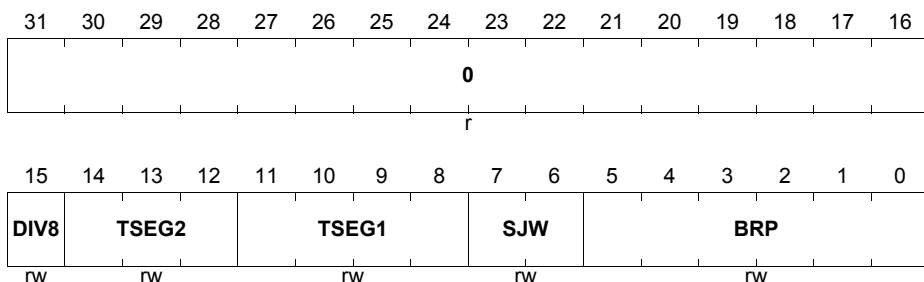


节点位时序寄存器NBTRx设置CAN传送位时序所需要的全部参数。仅当置位NCRx.CCE时，才可写入NBTRx。

### NBTRx (x = 0-1)

#### 节点 x 位时序寄存器

复位值：0000 0000<sub>H</sub>



符号	位	类型	描述
<b>BRP</b>	[5:0]	rw	<b>波特率预分频</b> 如果 DIV8 = 0，一个位时间的持续时间为 (BRP + 1) 个时钟周期 如果 DIV8 = 1，一个位时间的持续时间为 8 × (BRP + 1) 个时钟周期
<b>SJW</b>	[7:6]	rw	<b>(重新) 同步跳跃宽度</b> 用于重新同步的时间为 (SJW + 1) 个时间单元
<b>TSEG1</b>	[11:8]	rw	<b>采样点之前的时间段</b> 用户定义的同步段结束和采样点之前的额定时间为 (TSEG1 + 1) 个时间单元，它包括它包括传播段（将信号传播延迟考虑在内）。重新同步可将该时间段拉长。 TSEG1 的有效值在 2 到 15 之间。
<b>TSEG2</b>	[14:12]	rw	<b>采样点之后的时间段</b> 用户定义的采样点和下一个同步段开始之间的额定时间为 (TSEG2 + 1) 个时间单元。重新同步可将该时间段延缩短。 TSEG2 的有效值在 1 到 7 之间。
<b>DIV8</b>	15	rw	<b>预分频时钟 8 分频</b> 0 一个时间单元持续 (BRP+1) 个时钟周期 1 一个时间单元持续 8 × (BRP+1) 个时钟周期

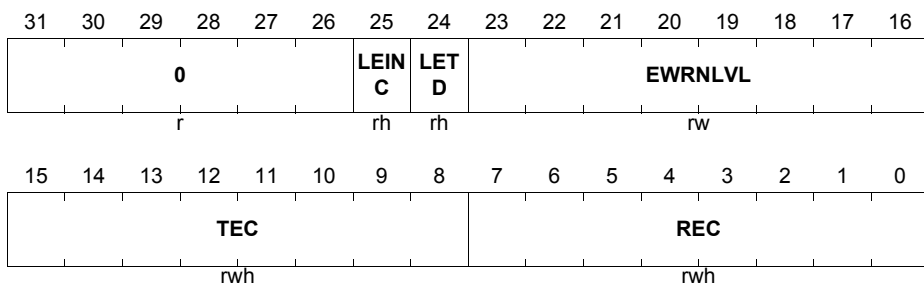
符号	位	类型	描述
0	[31:16]	r	保留 读操作返回 0；应写入 0。

节点错误计数器寄存器 **NECNTx** 包含 CAN 接收和发送错误计数器以及用于错误分析的一些附加位。只有置位 **NCRx.CCE** 时，才可向 **NECNTx** 写入。

### NECNTx (x = 0-1)

#### 节点 x 错误计数器寄存器

复位值：0060 0000<sub>H</sub>

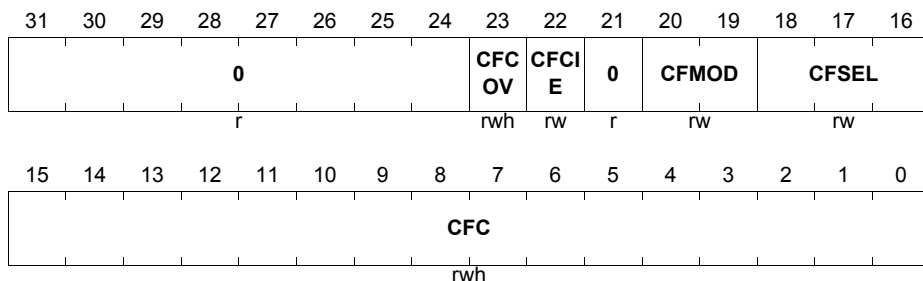


符号	位	类型	描述
REC	[7:0]	rwh	接收错误计数器 位域 REC 包含 CAN 节点 x 接收错误计数器的值。
TEC	[15:8]	rwh	发送错误计数器 位域 TEC 包含 CAN 节点 x 发送错误计数器的值。
EWRNLVL	[23:16]	rw	错误警告级别 位域 EWRNLVL 定义置位相应错误警告位 NSRx.EWRN 的门限值（错误警告级别，默认值 96）。
LETD	24	rh	最近错误的传送方向 0 最近错误（REC 值已增加）出现于 CAN 节点 x 执行接收操作时 1 最近错误（TEC 值已增加）出现于 CAN 节点 x 执行发送操作时
LEINC	25	rh	最近错误导致的错误计数器增量 0 最近错误使得错误计数器加 1 1 最近错误使得错误计数器加 8
0	[31:26]	r	保留 读操作返回 0；应写入 0。

节点帧计数器寄存器 **NFCRx** 包含帧计数器的实际值及帧计数器的控制和状态位。

**NFCRx (x = 0-1)**

节点 x 帧计数器寄存器

复位值: 0000 0000<sub>H</sub>


符号	位	类型	描述
CFC	[15:0]	rwh	<b>CAN 帧计数器</b> 在帧计数模式 (CFMOD = 00 <sub>B</sub> )，该位域包含帧计数器值； 在时间戳模式 (CFMOD = 01 <sub>B</sub> )，该位域包含捕获的位时间计数值，捕获新帧的开始。 在所有的位时序分析模式 (CFMOD = 10 <sub>B</sub> )，CFC 总是给出 $f_{CAN}$ 时钟周期数 (测量结果) 减 1 的值。例如： CFSEL = 000 <sub>B</sub> ，测量模式下，CFC 的值为 34 意味着在接收输入上的最近两个显性沿之间的时间为 35 个 $f_{CAN}$ 时钟周期。

符号	位	类型	描述
<b>CFSEL</b>	[18:16]	rw	<b>CAN 帧计数选择</b> 该位域为选定的帧计数器模式选择帧计数器的功能。 <b>帧计数模式</b> 位 0 如果置位 CFSEL 的位 0，那么每次 CAN 总线上接收一个外来帧（即该帧和报文对象不匹配），CFC 增加。 位 1 如果置位 CFSEL 的位 1，那么每次 CAN 总线上接收到和报文对象匹配的报文帧，CFC 增加。 位 2 如果置位 CFSEL 的位 2，那么 CAN 节点成功发送一帧，CFC 增加。 <b>时间戳模式</b> 000 <sub>B</sub> 新的位时间开始时，帧计数器（内部地）增加。检测到新一帧的 SOF 位期间采样该值。采样值保存到 CFC 位域中。 <b>位时序模式</b> 可用的位时序测量模式见表 16-9。如果 CFCIE 置位，那么 CFC 更新操作将在节点 x（x 为 CAN 节点编号）上产生中断请求。
<b>CFMOD</b>	[20:19]	rw	<b>CAN 帧计数器模式</b> 该位域决定帧计数器的工作模式。 00 <sub>B</sub> 帧计数模式：接收和发送报文帧时，帧计数器增加 01 <sub>B</sub> 时间戳模式：帧计数器用于计算位时间 10 <sub>B</sub> 位时序模式：帧计数器用于位时序分析 11 <sub>B</sub> 保留
<b>CFCIE</b>	22	rw	<b>CAN 帧计数中断使能</b> CFCIE 使能 CAN 节点 x 的 CAN 帧计数器溢出中断。 0 禁止 CAN 帧计数器溢出中断 1 使能 CAN 帧计数器溢出中断 位域 NIPRx.CFCINP 为该类型中断选择中断输出线。
<b>CFCOV</b>	23	rwh	<b>CAN 帧计数器溢出标志</b> 帧计数器溢出（从 FFFF <sub>H</sub> 计数至 0000 <sub>H</sub> ）时，标志 CFCOV 被置位。在位时序分析模式，更新 CFC 将引起 CFCOV 置位。如果 CFCIE = 1，将产生一个中断请求。 0 自从该标志最近一次被复位，未出现溢出 1 自从该标志最近一次被复位，已出现溢出 CFCOV 必须由软件复位。
<b>0</b>	21, [31:24]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 位时序分析模式

**表 16-9 位时序分析模式（CFMOD = 10）**

CFSEL	测量
000 <sub>B</sub>	只要在接收输入端上监测到一个显性沿（从 1 跳变到 0），则将该显性沿和最近的显性沿之间的时间（以时钟周期为单位）保存到 CFC 中。
001 <sub>B</sub>	只要在接收输入端上监测到一个隐性沿（从 0 到 1 的跳变），则将该隐性沿和最新的显性沿之间的时间（以时钟周期为单位）保存到 CFC 中。
010 <sub>B</sub>	只要接收到作为发送显性沿结果的一个显性沿，则将两个沿之间的时间（时钟周期数）保存到 CFC 中。
011 <sub>B</sub>	只要接收到作为发送隐性沿结果的一个隐性沿，则将两个沿之间的时间（时钟周期数）保存到 CFC 中。
100 <sub>B</sub>	只要在接收输入端上监测到用于同步的显性沿，则将该沿和最近采样点之间的时间（以时钟周期为单位）保存到 CFC 中。
101 <sub>B</sub>	在每个采样点，将新位起始时间和前一位起始时间之间的时间（以时钟周期为单位）保存在 CFC[11:0] 中，附加信息写入到 CFC[15:12] 中： CFC[15]：实际位时间的发送值 CFC[14]：实际位时间的接收采样值 CFC[13:12]：CAN 总线信息（见表 16-10）
111 <sub>B</sub>	保留，不要使用该组合。

**表 16-10 CAN 总线状态信息**

CFSEL	CAN 总线状态
00 <sub>B</sub>	<b>没有位</b> CAN 总线空闲，执行位填充（解填充）或者位于下面的帧段： SOF、保留位、SRR、CRC、分隔符、EOF 前 6 位、IFS
01 <sub>B</sub>	<b>新位</b> 该编码代表新帧段的首位。 当前位是下列帧段的首位： 标准 ID（仅发送）的位 10（MSB）、RTR、IDE、DLC（MSB）、每个数据字节的位 7（MSB）以及 ID 扩展的首位。

表 16-10      CAN 总线状态信息

CFC[13:12]	CAN 总线状态
10 <sub>B</sub>	<p><b>位</b></p> <p>该编码代表帧段长度大于 1 中的位 （不是由新位表示的那些帧段的首位）。</p> <p>当前位在下列帧段内处理：</p> <p>ID 位 （除了用于发送的标准 ID 的首位和扩展 ID 的首位）， DLC （3 LSB）和每个数据字节的位 6-0。</p>
11 <sub>B</sub>	<p><b>完成</b></p> <p>当前位位于下面的帧段：</p> <p>应答时隙、 EOF 末位、激活 / 认可错误帧、过载帧</p> <p>连续的两个或更多完成码表示一个错误帧。</p>

### 16.2.3 报文对象寄存器

报文对象控制寄存器 MOCTRn 和报文对象状态寄存器 MOSTATn 位于报文对象地址段的同一个偏移地址上（偏移地址为 1C<sub>H</sub>）。寄存器 MOCTRn 是只写寄存器，可由软件对与 CAN 传送相关的控制位进行置位 / 复位。

#### MOCTR0

报文对象 0 控制寄存器

复位值：0100 0000<sub>H</sub>

#### MOCTR31

报文对象 31 控制寄存器

复位值：1F1E 0000<sub>H</sub>

#### MOCTRn (n = 1-30)

报文对象 n 控制寄存器

复位值：((n+1)\*01000000<sub>H</sub>)+((n-1)\*00010000<sub>H</sub>)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				SET DIR	SET TXE N1	SET TXE N0	SET TXR Q	SET RXE N	SET RTS EL	SET MSG VAL	SET MSG LST	SET NEW DAT	SET RXU PD	SET TXP ND	SET RXP ND
W				W	W	W	W	W	W	W	W	W	W	W	W

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				RES DIR	RES TXE N1	RES TXE N0	RES TXR Q	RES RXE N	RES RTS EL	RES MSG VAL	RES MSG LST	RES NEW DAT	RES RXU PD	RES TXP ND	RES RXP ND
W				W	W	W	W	W	W	W	W	W	W	W	W

符号	位	类型	描述
RESRXPND	0	w	接收挂起复位 / 置位 这两位控制 RXPND 的置位 / 复位条件（见表 16-11）。
SETRXPND	16	w	
RESTXPND	1	w	发送挂起复位 / 置位 这两位控制 TXPND 的置位 / 复位条件（见表 16-11）。
SETTXPND	17	w	
RESRXUPD	2	w	接收更新复位 / 置位 这两位控制 RXUPD 的置位 / 复位条件（见表 16-11）。
SETRXUPD	18	w	
RESNEWDAT	3	w	新数据复位 / 置位 这两位控制 NEWDAT 的置位 / 复位条件（见表 16-11）。
SETNEWDAT	19	w	



符号	位	类型	描述
<b>RESMSGLST</b> <b>SETMSGLST</b>	4 20	w w	报文丢失复位 / 置位 这两位控制 MSGLST 的置位 / 复位条件（见 <a href="#">表 16-11</a> ）。
<b>RESMSGVAL</b> <b>SETMSGVAL</b>	5 21	w w	报文有效复位 / 置位 这两位控制 MSGVAL 的置位 / 复位条件（见 <a href="#">表 16-11</a> ）。
<b>RESRTSEL</b> <b>SETRTSEL</b>	6 22	w w	选中的接收 / 发送复位 / 置位 这两位控制 RTSEL 的置位 / 复位条件（见 <a href="#">表 16-11</a> ）。
<b>RESRXEN</b> <b>SETRXEN</b>	7 23	w w	接收使能复位 / 置位 这两位控制 RXEN 的置位 / 复位条件（见 <a href="#">表 16-11</a> ）。
<b>RESTXRQ</b> <b>SETTXRQ</b>	8 24	w w	发送请求复位 / 置位 这两位控制 TXRQ 的置位 / 复位条件（见 <a href="#">表 16-11</a> ）。
<b>RESTXEN0</b> <b>SETTXEN0</b>	9 25	w w	发送使能 0 复位 / 置位 这两位控制 TXEN0 的置位 / 复位条件（见 <a href="#">表 16-11</a> ）。
<b>RESTXEN1</b> <b>SETTXEN1</b>	10 26	w w	发送使能 1 复位 / 置位 这两位控制 TXEN1 的置位 / 复位条件（见 <a href="#">表 16-11</a> ）。
<b>RESDIR</b> <b>SETDIR</b>	11 27	w w	报文方向复位 / 置位 这两位控制 DIR 的置位 / 复位条件（见 <a href="#">表 16-11</a> ）。
<b>0</b>	[15:12], [31:28]	w	保留 应当写入 0

**表 16-11** 寄存器 MOCTRn 中各位的置位 / 复位条件

RESy 位 <sup>1)</sup>	SETy 位	写操作
写 0	写 0	不改变该位
	无写操作	
无写操作	写 0	
写 1	写 1	
写 1	写 0	复位该位
	无写操作	

表 16-11 寄存器 MOCTRn 中各位的置位 / 复位条件

RESy 位 <sup>1)</sup>	SETy 位	写操作
写 0	写 1	置位该位
无写操作		

1) 参数 “y” 代表该位名称的第二部分 (“RXPND”, “TXPND”, ..., 直至 “DIR”)。

寄存器 MOSTATn 是只读寄存器，指示报文对象列表的状态信息，如当前报文对象的前一个和后一个报文对象的编号及该报文对象所属的列表编号。

### MOSTAT0

报文对象 0 状态寄存器

复位值：0100 0000<sub>H</sub>

### MOSTAT31

报文对象 31 状态寄存器

复位值：1F1E 0000<sub>H</sub>

MOSTATn (n = 1-30)

报文对象 n 状态寄存器

复位值：((n+1)\*01000000<sub>H</sub>)+((n-1)\*00010000<sub>H</sub>)

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
PNEXT								PPREV							
rh								rh							
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
LIST				DIR	TX EN1	TX EN0	TX RQ	RX EN	RTS EL	MSG VAL	MSG LST	NEW DAT	RX UPD	TX PND	RX PND
rh				rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh	rh

符号	位	类型	描述
RXPND	0	rh	<b>接收挂起</b> 0 没有接收到 CAN 报文 1 报文对象 n 已经接收到报文，直接接收或者通过网关复制操作间接接收。 RXPND 不能被硬件清零，必须由软件复位。
TXPND	1	rh	<b>发送挂起</b> 0 没有发送 CAN 报文 1 报文对象 n 通过 CAN 总线成功发送 CAN 报文 TXPND 不能被硬件清零，必须由软件复位。
RXUPD	2	rh	<b>接收更新</b> 0 未进行接收更新 1 正在进行报文标识符、DLC 和报文数据的更新

符号	位	类型	描述
<b>NEWDAT</b>	3	rh	<b>新数据</b> 0 自从最近一次该标志被复位，报文对象 n 未被更新 1 报文对象 n 已经被更新 接收到的 CAN 帧保存到报文对象 n 之后，硬件置位 NEWDAT。 当报文对象 n 开始进行 CAN 发送时，硬件清零 NEWDAT。 在新的发送数据保存到报文对象 n 中之后，NEWDAT 必须由软件置位，在当前进行的发送结束时，防止 TXRQ 被自动复位。
<b>MSGLST</b>	4	rh	<b>报文丢失</b> 0 未丢失报文 1 由于在 NEWDAT 已经置位的情况下，对 NEWDAT 再次置位，导致 CAN 报文丢失。
<b>MSGVAL</b>	5	rh	<b>报文有效</b> 0 报文对象 n 无效 1 报文对象 n 有效 有效的报文对象才能参与 CAN 传送。
<b>RTSEL</b>	6	rh	<b>选中的接收 / 发送</b> 0 未选中报文对象 n 进行接收 / 发送操作 1 选中报文对象 n 进行接收 / 发送操作 <b>帧接收：</b> 当报文对象已经被确认，用来存储当前接收到的 CAN 帧时，由硬件置位 RTSEL。在将接收到的帧最终保存到该报文对象 n 之前，检查 RTSEL 的置位情况。因而，CPU 可以通过软件清零 RTSEL 的方法禁止预定的发送操作。 <b>帧发送：</b> 当已经确认该报文对象 n 为下一个发送的报文对象时，RTSEL 被硬件置位。在报文对象实际建立传送之前，检查 RTSEL 是否仍然置位，位 NEWDAT 被清零。因帧发送成功而验证报文对象 n 之前，也要检查 RTSEL 是否仍然置位。 仅当报文对象 n 的上下文改变时，才需要检查 RTSEL，可以避免和正在进行的帧发送冲突。在所有其它情况下，可以忽略 RTSEL。RTSEL 对报文验收滤波没有影响。RTSEL 不能被硬件清零

符号	位	类型	描述
<b>RXEN</b>	7	rh	<b>接收使能</b> 0 未使能报文对象 <b>n</b> 接收帧 1 使能报文对象 <b>n</b> 接收帧 仅为接收验收滤波评估 <b>RXEN</b> 的值。
<b>TXRQ</b>	8	rh	<b>发送请求</b> 0 未请求报文对象 <b>n</b> 发送报文 1 请求报文对象 <b>n</b> 在 CAN 总线上发送报文 仅在 <b>TXRQ</b> 、 <b>TXEN0</b> 、 <b>TXEN1</b> 和 <b>MSGVAL</b> 置位的情况下，发送请求才有效。如果匹配的远程帧被正确接收，硬件置位 <b>TXRQ</b> 。如果报文对象 <b>n</b> 被成功发送且 <b>NEWDAT</b> 没有被软件再次置位，硬件复位 <b>TXRQ</b> 。
<b>TXEN0</b>	9	rh	<b>发送使能 0</b> 0 未使能报文对象 <b>n</b> 进行帧发送 1 使能报文对象 <b>n</b> 进行帧发送 只有位 <b>TXEN0</b> 和 <b>TXEN1</b> 都置位的情况下，报文对象 <b>n</b> 才能进行发送。 用户可以清零 <b>TXEN0</b> ，来禁止当前更新的报文发送操作，或者禁止远程帧的自动回应。
<b>TXEN1</b>	10	rh	<b>发送使能 1</b> 0 未使能报文对象 <b>n</b> 进行帧发送 1 使能报文对象 <b>n</b> 进行帧发送 只有在 <b>TXEN0</b> 和 <b>TXEN1</b> 都置位的情况下，报文对象 <b>n</b> 才能发送。 <b>MultiCAN</b> 模块用 <b>TXEN1</b> 在发送 FIFO 中选择有效报文对象。
<b>DIR</b>	11	rh	<b>报文方向</b> 0 接收对象选择： <b>TXRQ</b> = 1，安排发送带有报文对象 <b>n</b> 的标识符的远程帧。接收标识符匹配的数据帧，报文存储在报文对象 <b>n</b> 中。 1 发送对象选择： 如果 <b>TXRQ</b> = 1，安排用报文对象 <b>n</b> 发送数据帧。接收标识符匹配的远程帧，置位 <b>TXRQ</b> 。
<b>LIST</b>	[15:12]	rh	<b>列表分配</b> <b>LIST</b> 指示报文对象 <b>n</b> 所属的报文列表编号。当报文对象的列表分配被命令面板修改时，硬件更新 <b>LIST</b> 。

符号	位	类型	描述
PPREV	[23:16]	rh	指向前一个报文对象的指针 PPREV 保存报文列表结构中报文对象 n 的前一个报文对象的编号。
PNEXT	[31:24]	rh	指向下一个报文对象的指针 PNEXT 保存报文列表结构中报文对象 n 的下一个报文对象的编号。

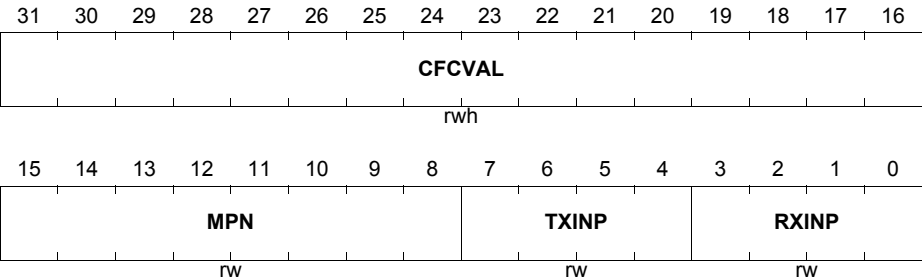
表 16-12      MOSTATn 复位值

报文对象	PNEXT	PPREV	复位值
0	1	0	0100 0000 <sub>H</sub>
1	2	0	0200 0000 <sub>H</sub>
2	3	1	0301 0000 <sub>H</sub>
3	4	2	0402 0000 <sub>H</sub>
...	...	...	...
28	29	27	1D1B 0000 <sub>H</sub>
29	30	28	1E1C 0000 <sub>H</sub>
30	31	29	1F1D 0000 <sub>H</sub>
31	31	30	1F1E 0000 <sub>H</sub>

报文对象中断指针寄存器 MOIPRn 保存报文中断指针，报文挂起编号和报文对象 n 的帧计数器的值。

MOIPRn (n = 0-31)

报文对象 n 中断指针寄存器 复位值: 0000 0000<sub>H</sub>



符号	位	类型	描述
<b>RXINP</b>	[3:0]	rw	<b>接收中断节点指针</b> RXINP 为报文对象 $n$ 接收中断事件选择中断输出线 CANSRC $m$ ( $m = 0-7$ )。RXINP 也可用于报文挂起位选择 (见 <a href="#">页 16-21</a> )。 0000 <sub>B</sub> 选择中断输出线 CANSRC0 0001 <sub>B</sub> 选择中断输出线 CANSRC1 ...            ... 0110 <sub>B</sub> 选择中断输出线 CANSRC6 0111 <sub>B</sub> 选择中断输出线 CANSRC7 1000 <sub>B</sub> -1111 <sub>B</sub> 保留
<b>TXINP</b>	[7:4]	rw	<b>发送中断节点指针</b> TXINP 为报文对象 $n$ 发送中断事件选择中断输出线 CANSRC $m$ ( $m = 0-7$ )。TXINP 也可用于报文挂起位选择 (见 <a href="#">页 16-21</a> )。 0000 <sub>B</sub> 选择中断输出线 CANSRC0 0001 <sub>B</sub> 选择中断输出线 CANSRC1 ...            ... 0110 <sub>B</sub> 选择中断输出线 CANSRC6 0111 <sub>B</sub> 选择中断输出线 CANSRC7 1000 <sub>B</sub> -1111 <sub>B</sub> 保留
<b>MPN</b>	[15:8]	rw	<b>报文挂起编号</b> 该位域用来选择报文挂起寄存器中的挂起位位置，根据报文对象 $n$ 接收 / 发送中断置位。
<b>CFCVAL</b>	[31:16]	rwh	<b>CAN 帧计数器值</b> 当报文保存到报文对象 $n$ 中，或者报文对象 $n$ 成功发送报文时，复制 CAN 帧计数器值 NFCRx.CFC 到 CFCVAL 中。

报文对象功能寄存器 MOFCRn 包含选择和控制报文对象功能的控制位，还包含 CAN 数据长度码。

### MOFCRn (n = 0-31)

报文对象 n 功能控制寄存器

复位值: 0000 0000<sub>H</sub>

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
0				DLC				STT	SDT	RMM	FRR EN	0	OVIE	TXIE	RXIE
rw				rwh				rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0				DAT C	DLC C	IDC	GDF S	0				MMC			
rw				rw	rw	rw	rw	rw				rw			

符号	位	类型	描述
MMC	[3:0]	rw	<b>报文模式控制</b> MMC 控制报文对象 n 的报文模式。 0000 <sub>B</sub> 标准报文对象 0001 <sub>B</sub> 接收 FIFO 基本对象 0010 <sub>B</sub> 发送 FIFO 基本对象 0011 <sub>B</sub> 发送 FIFO 从属对象 0100 <sub>B</sub> 网关源对象 其它值 保留
GDFS	8	rw	<b>网关数据帧发送</b> 0 目标对象中的 TXRQ 不变 1 数据帧从网关源对象传送到目标对象之后，网关目标对象 TXRQ 置位。 仅适用于网关源对象；在其它节点中被忽略。
IDC	9	rw	<b>标识符复制</b> 0 网关源对象的标识符未被复制 1 网关源对象的标志符（在接收到的帧保存到源对象中之后）被复制到网关目标对象中。 仅适用于网关源对象；在其它节点中被忽略。



符号	位	类型	描述
<b>DLCC</b>	10	rw	<b>数据长度码复制</b> 0 不复制数据长度码 1 网关源对象中的数据长度码（在接收到的帧保存到源对象中之后）被复制到网关目标对象 仅适用于网关源对象；在其它节点中被忽略。
<b>DATC</b>	11	rw	<b>数据复制</b> 0 不复制数据域 1 网关源对象中寄存器 MODATALn 和 MODATAHn 中的数据域（在接收到的帧保存到源对象中之后）被复制到网关目标对象 仅适用于网关源对象；在其它节点中被忽略。
<b>RXIE</b>	16	rw	<b>接收中断使能</b> RXIE 使能报文对象 n 的报文接收中断。该中断在接收 CAN 报文之后产生（与报文是直接接收还是通过网关操作接收无关）。 0 禁止报文接收中断 1 使能报文接收中断 由位域 MOIPRn.RXINP 选择被该类型中断激活的中断输出线。
<b>TXIE</b>	17	rw	<b>发送中断使能</b> TXIE 使能报文对象 n 的报文发送中断。在 CAN 报文发送之后产生该中断。 0 禁止报文发送中断 1 使能报文发送中断 由位域 MOIPRn.TXINP 选择被该类型中断激活的中断输出线。
<b>OVIE</b>	18	rw	<b>溢出中断使能</b> OVIE 使能报文对象 n FIFO 已满中断。当前报文报文对象指针（CUR）达到 FIFO/ 网关指针寄存器的 SEL 中的值时，产生该中断。 0 禁止 FIFO 已满中断 1 使能 FIFO 已满中断 如果报文对象 n 是接收 FIFO 基本报文对象，位域 MOIPRn.TXINP 选择被该类型中断激活的中断输出线。 如果报文对象 n 作为发送 FIFO 基本报文对象，由位域 MOIPRn.RXINP 选择被该类型中断激活的中断输出线。 对于所有其它报文对象模式，OVIE 无影响。

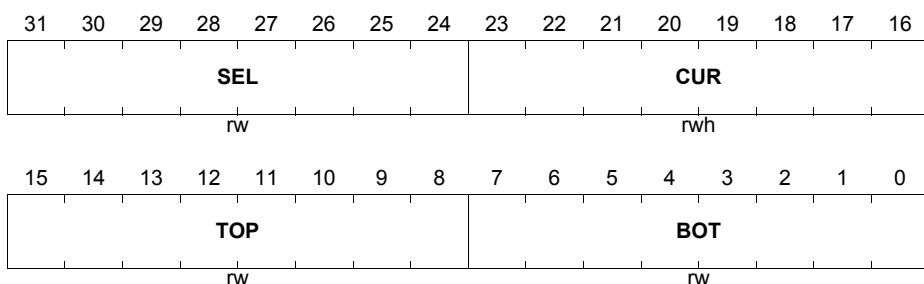
符号	位	类型	描述
<b>FRREN</b>	20	rw	<b>外来远程请求使能</b> 指定由报文对象 <b>n</b> 中、或有指针 <b>CUR</b> 指定的外来报文对象置位 <b>TXRQ</b> 。 0 报文对象 <b>n</b> 接收到一个匹配的远程帧时，置位 <b>TXRQ</b> 1 由 <b>CUR</b> 指定的报文对象接收到一个匹配的远程帧时，置位 <b>TXRQ</b>
<b>RMM</b>	21	rw	<b>发送对象远程监控</b> 0 禁止远程监控： 报文对象 <b>n</b> 的标识符、 <b>IDE</b> 位和 <b>DLC</b> 位在接收匹配远程帧时保持不变 1 使能远程监控： 远程帧的标识符、 <b>IDE</b> 位和 <b>DLC</b> 位被复制到发送对象 <b>n</b> 中，以监控接收到的远程帧 位 <b>RMM</b> 仅适用于发送对象，对于接收对象无影响。
<b>SDT</b>	22	rw	<b>单一数据传送</b> 如果 <b>SDT</b> = 1 且报文对象 <b>n</b> 不是 <b>FIFO</b> 基本对象，那么当该报文对象参与一次成功的数据传送（接收或者发送）时，复位该报文对象的 <b>MSGVAL</b> 。 如果 <b>SDT</b> = 1 且报文对象 <b>n</b> 是 <b>FIFO</b> 基本对象，那么当指向当前报文对象 <b>CUR</b> 的指针达到 <b>FIFO/ 网关</b> 指针寄存器 <b>SEL</b> 中的值时，复位 <b>MSGVAL</b> 。 当 <b>SDT</b> = 0 时，位 <b>MSGVAL</b> 不受影响。
<b>STT</b>	23	rw	<b>单次发送试验</b> 如果该位被置位，那么在报文对象 <b>n</b> 开始发送时 <b>TXRQ</b> 清零。若发送失败，不再次执行发送操作。
<b>DLC</b>	[27:24]	rwh	<b>数据长度码</b> 该位域为报文对象 <b>n</b> 定义数据字节个数。 <b>DLC</b> 有效值从 0 到 8。 <b>DLC</b> > 8 将使得数据长度为 8 个字节。在接收或者发送 <b>CAN</b> 帧时，不截断 <b>DLC</b> 码。
<b>0</b>	[7:4], [15:12], 19, [31:28]	rw	<b>保留</b> 复位之后读操作返回 0；读操作返回最近写入值；应写入 0。

报文对象 FIFO/ 网关指针寄存器 MOFGPRn 包含一组用于 FIFO 和网关操作的报文对象连接指针。

### MOFGPRn (n = 0-31)

报文对象 n FIFO/ 网关指针寄存器

复位值: 0000 0000<sub>H</sub>



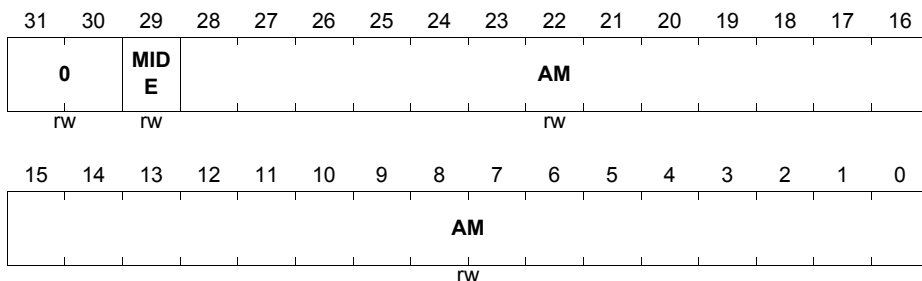
符号	位	类型	描述
<b>BOT</b>	[7:0]	rw	<b>底端指针</b> 位域 BOT 指出 FIFO 结构中的第一个元素。
<b>TOP</b>	[15:8]	rw	<b>顶端指针</b> 位域 TOP 指出 FIFO 结构中的最后一个元素。
<b>CUR</b>	[23:16]	rwh	<b>当前报文对象指针</b> 位域 CUR 指出 FIFO/ 网关结构中实际目标报文对象。 在 FIFO/ 网关操作之后，由列表结构中的下一个报文对象的报文编号（由报文控制寄存器的 PNEXT 给出）更新 CUR，直到到达 FIFO 的顶端的元素（由 TOP 给出），此时将它复位到 FIFO 底端的元素（由 BOT 给出）。
<b>SEL</b>	[31:24]	rw	<b>报文对象选择指针</b> 位域 SEL 为第二指针（软件），作为 FIFO 结构中硬件指针 CUR 的补充，SEL 用于监控目的（FIFO 中断产生）。

报文对象  $n$  验收屏蔽寄存器 MOAMRn 包含报文对象验收滤波屏蔽位。

### MOAMRn ( $n = 0-31$ )

报文对象  $n$  验收屏蔽寄存器

复位值: 3FFF FFFF<sub>H</sub>



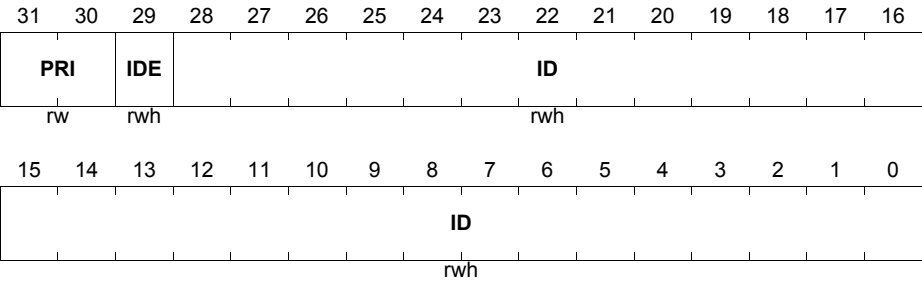
符号	位	类型	描述
<b>AM</b>	[28:0]	rw	<b>报文标识符验收屏蔽</b> 位域 AM 共 29 位，可对接收到的标准标识符（AM[28:18]）或者扩展标识符（AM[28:0]）进行验收滤波。对于标准标识符，“不考虑”位 AM[17:0]。
<b>MIDE</b>	29	rw	<b>报文 IDE 位的验收屏蔽位</b> 0 报文对象 $n$ 接收验收标准和扩展帧 1 报文对象 $n$ 仅接收验收 IDE 位匹配的报文帧
<b>0</b>	[31:30]	rw	<b>保留</b> 复位之后读操作返回 0；读操作返回最近写入值；应写入 0。

报文对象 n 仲裁寄存器 MOARn 包含报文对象 CAN 标识符。

MOARn (n = 0-31)

报文对象 n 仲裁寄存器

复位值: 0000 0000<sub>H</sub>



符号	位	类型	描述
ID	[28:0]	rwh	报文对象 n 的 CAN 标识符 标准报文 (ID[28:18]) 或者扩展报文 (ID[28:0]) 的标识符。对于标准标识符, “不考虑” 位 ID[17:0]。
IDE	29	rwh	报文对象 n 的扩展标识符位 0 报文对象 n 处理带 11 位标识符的标准帧 1 报文对象 n 处理带 29 位标识符的扩展帧

符号	位	类型	描述
PRI	[31:30]	rw	<p><b>优先级分类</b></p> <p>PRI 给报文对象 n 分配 4 个优先级 0、1、2、3。低 PRI 值的优先级较高。总是由 PRI 值较低的报文对象赢得接收和发送验收滤波。基于标识符 / 屏蔽和列表次序的验收滤波只在相同优先级的报文对象之间执行。</p> <p>PRI 还确定发送验收滤波的方法：</p> <p>00<sub>B</sub> 保留</p> <p>01<sub>B</sub> 发送验收滤波基于列表次序。这意味着，只有列表中文本对象 n 之前的所有报文对象都没有有效的发送请求（(MSGVAL &amp; TXEN0 &amp; TXEN1 = 1）时，才考虑发送报文对象 n。</p> <p>10<sub>B</sub> 发送验收滤波基于 CAN 标识符。这意味着，仅在列表中没有具有更高优先级标识符 + IDE + DIR（根据 CAN 仲裁规则）的其它报文对象时，才考虑由报文对象 n 发送（见表 16-13）。</p> <p>11<sub>B</sub> 发送验收滤波基于列表中的次序（与 PRI = 01<sub>B</sub> 相似）。</p>

基于 CAN 仲裁规则的报文对象发送优先级

表 16-13 基于 CAN 仲裁规则的报文对象发送优先级

所选报文对象 A 和 B 的设置 (A 的发送优先级高于 B)	注解
A.MOAR[28:18] < B.MOAR[28:18] (A 的 11 位标准标识符小于 B 的 11 位标准标识符)	带较低标准标识符的报文具有较高优先级。 MOAR[28] 为标准标识符的最高有效位 (MSB)。MOAR[18] 为标准标识符的最低有效位 (LSB)。
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = 0 (发送标准帧) B.MOAR.IDE = 1 (发送扩展帧)	具有相等的标准标识符时，标准帧比扩展帧优先级高。
A.MOAR[28:18] = B.MOAR[28:18] A.MOAR.IDE = B.MOAR.IDE = 0 A.MOCTR.DIR = 1 (发送数据帧) B.MOCTR.DIR = 0 (发送远程帧)	具有相等的标识符时，标准数据帧比标准远程帧优先级高。

表 16-13      基于 CAN 仲裁规则的报文对象发送优先级

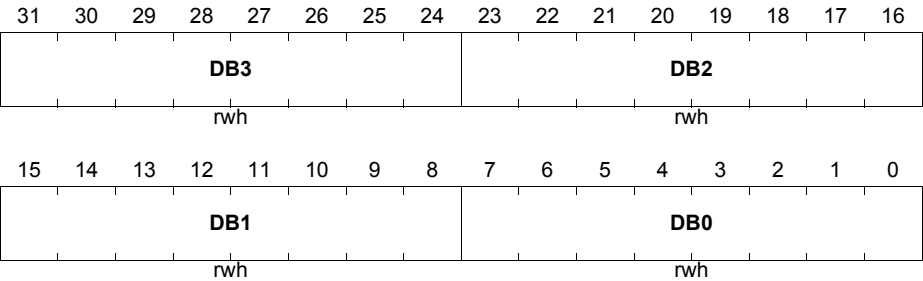
所选报文对象 A 和 B 的设置 (A 的发送优先级高于 B)	注解
A.MOAR[28:0] = B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 A.MOCTR.DIR = 1 （发送数据帧） B.MOCTR.DIR = 0 （发送远程帧帧）	具有相等的标识符时，扩展数据帧比扩展远程帧优先级高。
A.MOAR[28:0] < B.MOAR[28:0] A.MOAR.IDE = B.MOAR.IDE = 1 (29 位标识符)	带有较低标识符的扩展帧有较高的优先级。 MOAR[28] 为整个标识符的最高有效位（MSB）（MOAR[28:18] 为标准标识符，MOAR[17:0] 为标识符扩展部分）。 MOAR[0] 为整个标识符的最低有效位（LSB）。

报文对象  $n$  数据寄存器低位 MODATAL $n$  包含报文对象  $n$  的低四字节数据。未用到的数据字节接收时设置为 0，发送时忽略。

MODATAL $n$  ( $n = 0-31$ )

报文对象  $n$  数据寄存器低位

复位值: 0000 0000<sub>H</sub>

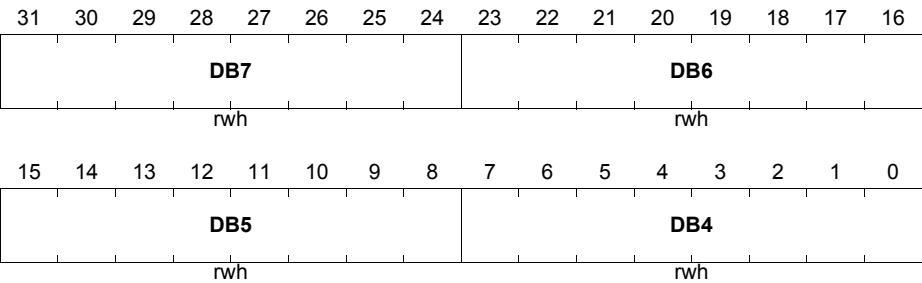


符号	位	类型	描述
DB0	[7:0]	rwh	报文对象 $n$ 的数据字节 0
DB1	[15:8]	rwh	报文对象 $n$ 的数据字节 1
DB2	[23:16]	rwh	报文对象 $n$ 的数据字节 2
DB3	[31:24]	rwh	报文对象 $n$ 的数据字节 3



数据寄存器高位 MODATAHn 包含报文对象 n 的高四字节数据。未用到的数据字节接收时设置为 0，发送时忽略。

**MODATAHn (n = 0-31)**  
 报文对象 n 数据寄存器高位 复位值: 0000 0000<sub>H</sub>



符号	位	类型	描述
DB4	[7:0]	rwh	报文对象 n 的数据字节 4
DB5	[15:8]	rwh	报文对象 n 的数据字节 5
DB6	[23:16]	rwh	报文对象 n 的数据字节 6
DB7	[31:24]	rwh	报文对象 n 的数据字节 7

## 16.2.4 MultiCAN 访问仲裁寄存器

### CAN\_ADCON

CAN 地址 / 数据控制寄存器

复位值: 0000 0000<sub>B</sub>

7	6	5	4	3	2	1	0
V3	V2	V1	V0	AUAD		BSY	RWEN
rw	rw	r	r	rw		rh	rw

符号	位	类型	描述
RWEN	0	rw	读 / 写使能 0 使能读操作 1 使能写操作
BSY	1	rh	数据发送忙 0 已经完成数据发送 1 正在进行数据发送
AUAD	[3:2]	rw	地址自动增加 / 减少 00 不进行地址增加 / 减少 01 当前地址自动增加 (+1) 10 当前地址自动减少 (-1) 11 当前地址自动增加 (+8)
V0	4	rw	CAN 数据 0 有效 0 CAN_DATA0 寄存器中的数据对于发送无效 1 CAN_DATA0 寄存器中的数据对于发送有效
V1	5	rw	CAN 数据 1 有效 0 CAN_DATA1 寄存器中的数据对于发送无效 1 CAN_DATA1 寄存器中的数据对于发送有效
V2	6	rw	CAN 数据 2 有效 0 CAN_DATA2 寄存器中的数据对于发送无效 1 CAN_DATA2 寄存器中的数据对于发送有效
V3	7	rw	CAN 数据 3 有效 0 CAN_DATA3 寄存器中的数据对于发送无效 1 CAN_DATA3 寄存器中的数据对于发送有效

### CAN\_ADL

CAN 地址寄存器低位

复位值：0000 0000<sub>B</sub>

7	6	5	4	3	2	1	0
CA9	CA8	CA7	CA6	CA5	CA4	CA3	CA2
rwh							

符号	位	类型	描述
CAn (n = 2 到 9)	n-2	rwh	CAN 地址位 n

### CAN\_ADH

CAN 地址寄存器高位

复位值：0000 0000<sub>B</sub>

7	6	5	4	3	2	1	0
				CA13	CA12	CA11	CA10
r				rwh	rwh	rwh	rwh

符号	位	类型	描述
CA10	0	rwh	CAN 地址位 10
CA11	1	rwh	CAN 地址位 11
CA12	2	rwh	CAN 地址位 12
CA13	3	rwh	CAN 地址位 13
0	[7:4]	r	保留；读操作返回 0；应写入 0。

### CAN\_DATA0

CAN 数据寄存器 0

复位值：0000 0000<sub>B</sub>

7	6	5	4	3	2	1	0
CD[7:0]							
rwh							

**MultiCAN**

符号	位	类型	描述
<b>CD</b>	[7:0]	rwh	<b>CAN 数据字节 0</b>

**CAN\_DATA1**
**CAN 数据寄存器 1**

 复位值: **0000 0000<sub>B</sub>**

7	6	5	4	3	2	1	0
CD[15:8]							
rwh							

符号	位	类型	描述
<b>CD</b>	[7:0]	rwh	<b>CAN 数据字节 1</b>

**CAN\_DATA2**
**CAN 数据寄存器 2**

 复位值: **0000 0000<sub>B</sub>**

7	6	5	4	3	2	1	0
CD[23:16]							
rwh							

符号	位	类型	描述
<b>CD</b>	[7:0]	rwh	<b>CAN 数据字节 2</b>

**CAN\_DATA3**
**CAN 数据寄存器 3**

 复位值: **0000 0000<sub>B</sub>**

7	6	5	4	3	2	1	0
CD[31:24]							
rwh							

符号	位	类型	描述
<b>CD</b>	[7:0]	rwh	<b>CAN 数据字节 3</b>

## 17 模数转换器

XC878 内含一个带有八路模拟输入选择的高性能 10 位模数转换器（ADC）。ADC 采用逐次逼近技术，最多可转换 8 种不同模拟通道的电压电平。

### 特性

- 逐次逼近
- 8 或 10 位精度
- 八路模拟通道
- 四个独立的结果寄存器
- 针对低速 CPU 访问时的结果数据保护（待读模式）
- 单次转换模式
- 自动扫描功能
- 转换结果的极限检查
- 数据压缩滤波（最多积累 2 个转换结果）
- 两个独立的、优先级可编程的转换请求源
- 转换请求触发方式可选择
- 灵活的中断产生方式，中断服务节点可配置
- 采样时间可编程
- 时钟分频器可编程
- 运行中的转换的取消 / 重启特性
- 集成的采样和保持电路
- 偏移误差补偿
- 低功耗模式

## 17.1 结构概述

ADC 模块由两部分、即模拟和数字部分组成，各部分内含独立的功能模块。

模拟部分包括：

- 模拟输入多路选择  
(选择转换通道)
- 模拟转换级  
(例如，作为 ADC 组成部分的电容网络和比较器)
- 模拟转换级的数字控制部分  
(控制模数转换过程并产生转换结果)

数字部分定义和控制 ADC 模块的总体功能，包括：

- 数字数据和转换请求的处理  
(控制转换触发机制并处理转换结果)
- 器件内部数据总线的总线接口  
(控制中断和寄存器访问)

ADC 模块的框图如图 17-1 所示。模拟输入通道  $x$  ( $x = 0 - 7$ ) 对应端口引脚  $ANx$ 。

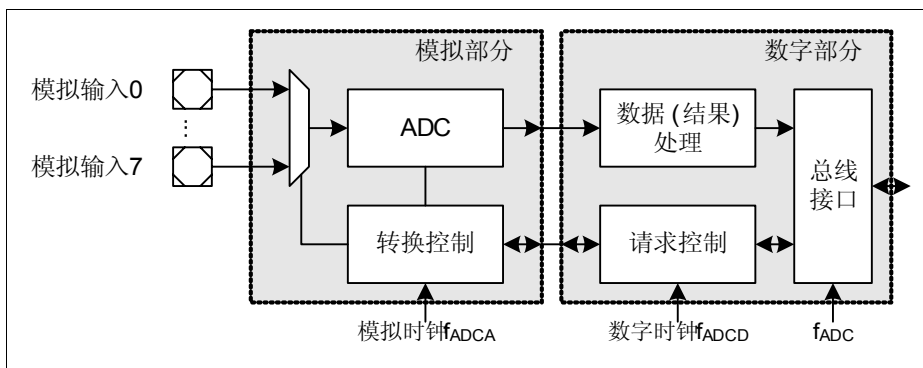


图 17-1 ADC 组成单元

## 17.2 时钟方案

由模块时钟  $f_{ADC}$  产生 ADC 模块模拟和数字部分所需的各种时钟信号：

- $f_{ADCA}$  为模拟部分的输入时钟
- $f_{ADCI}$  为模拟部分的内部时钟（定义转换时间和采样时间的时间基准），该时钟信号基于输入时钟  $f_{ADCA}$ ，在模拟单元内产生，为模拟单元提供正确占空比的时钟信号。
- $f_{ADCD}$  为数字部分的输入时钟。为仲裁器（定义仲裁周期的长短）和其它数字控制单元（如寄存器和中断产生）提供时钟。

预分频因子由寄存器 GLOBCTR 中的位域 CTC 来选择。ADC 无需工作在最大性能时，预分频因子可选择 32。

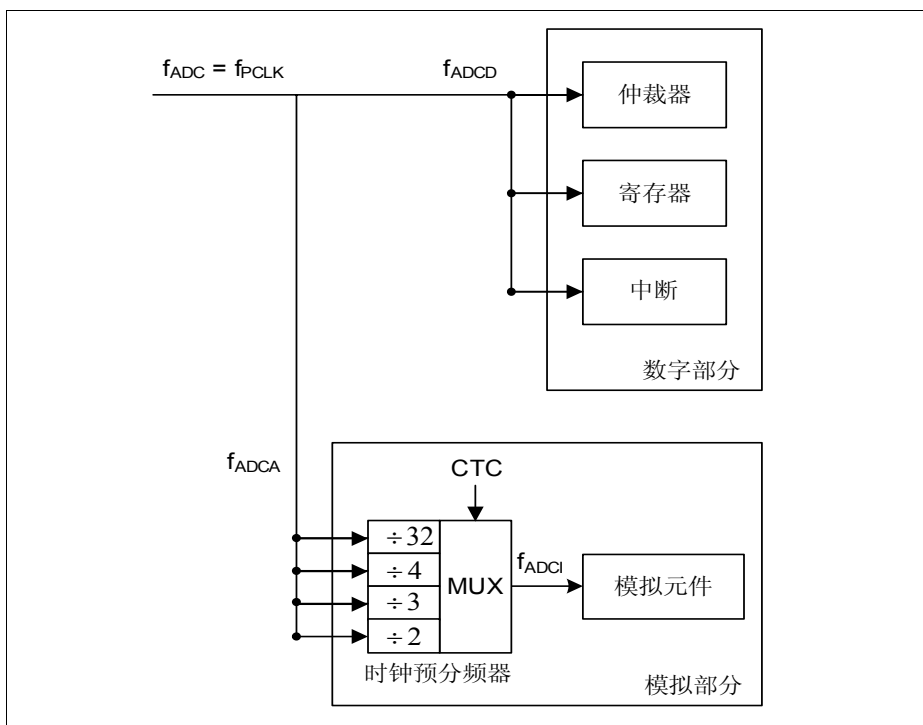


图 17-2 时钟方案

模块时钟  $f_{ADC} = 24 \text{ MHz}$  对应的模拟时钟  $f_{ADCI}$  频率选择见 [表 17-1](#)。

表 17-1  $f_{\text{ADCI}}$  频率选择

模块时钟 $f_{\text{ADC}}$	CTC	预分频因子	模拟时钟 $f_{\text{ADCI}}$
24 MHz	00 <sub>B</sub>	÷ 2	12 MHz
	01 <sub>B</sub>	÷ 3	8 MHz
	10 <sub>B</sub>	÷ 4	6 MHz
	11 <sub>B</sub> (缺省值)	÷ 32	750 kHz

低速模式下,  $f_{\text{ADC}}$  可减至 12 MHz、6 MHz 等等。有一点必须注意, 若低速模式下  $f_{\text{ADC}}$  太低, 可能因电容器电荷损失而导致模数转换错误增多。

## 17.2.1 转换时序

模数转换过程包括以下阶段:

- 同步阶段 ( $t_{\text{SYN}}$ )
- 采样阶段 ( $t_{\text{S}}$ )
- 转换阶段
- 结果写入阶段 ( $t_{\text{WR}}$ )

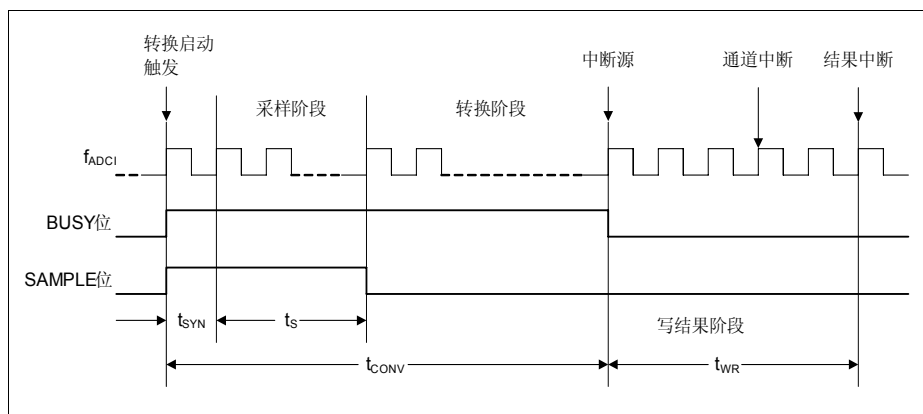


图 17-3 转换时序

### 采样阶段 $t_{\text{SYN}}$

转换启动触发器 (数字部分) 和开始采样 (模拟部分) 之间需要一个  $f_{\text{ADCI}}$  周期进行同步, 转换启动触发器将置位忙标志位 (BUSY) 和采样位 (SAMPLE)。



### 采样阶段 $t_s$

该阶段采样模拟输入电压。内部电容器阵列连接到选中的模拟输入通道上，加载需转换的模拟电压。模拟电压送至内部的电压比较器。开始采样后，寄存器 **GLOBSTR** 中的 **SAMPLE** 和 **BUSY** 标志被置位。该阶段的持续时间对所有模拟输入通道都一样，由寄存器 **INPCR0** 中的位域 **STC** 控制：

$$t_s = (2 + \text{STC}) \times t_{\text{ADCI}} \quad (17.1)$$

### 转换阶段

在该阶段，采用二进制权重电容网络的逐次逼近技术，将模拟电压转换为 8 位或 10 位的数字量。转换开始时，**SAMPLE** 标志复位（表明采样结束），**BUSY** 标志保持有效。只有在转换结束且相应的转换请求源（触发转换的请求源）产生中断时 **BUSY** 标志才复位。

### 结果写入阶段 $t_{\text{WR}}$

转换阶段结束时，在执行完极限检查、三个  $f_{\text{ADCI}}$  周期之后产生相应（转换通道）的通道中断。一旦转换结果写入目标结果寄存器，即产生结果中断。

**总转换时间  $t_{\text{CONV}}$** 

总的转换时间（同步 + 采样 + 电荷重新分配） $t_{\text{CONV}}$  由下面的公式给出：

$$t_{\text{CONV}} = t_{\text{ADC}} \times (1 + r \times (3 + n + \text{STC})) \quad (17.2)$$

其中

$r = \text{CTC} + 2$  对应于  $\text{CTC} = 00_{\text{B}}, 01_{\text{B}}$  或  $10_{\text{B}}$ ,

$r = 32$  对应于  $\text{CTC} = 11_{\text{B}}$ ,

$\text{CTC}$  = 转换时间控制

$\text{STC}$  = 采样时间控制

$n = 8$  或  $10$ （分别对应于 8 位和 10 位转换）

$$t_{\text{ADC}} = 1 / f_{\text{ADC}}$$

举例如下：

$\text{STC} = 00_{\text{H}}$ ,

$\text{CTC} = 01_{\text{B}}$ ,

$f_{\text{ADC}} = 26.7 \text{ MHz}$ ,

$n = 10$ ,

$$t_{\text{CONV}} = t_{\text{ADC}} \times (1 + 3 \times (3 + 10 + 0)) = 1.5 \mu\text{s}$$

### 17.3 低功耗模式

为了降低功耗，在无需进行模数转换时，可将 ADC 模块部分关闭或完全关闭：  
 ADC 模块的模拟部分可通过复位 ANON 位来关闭，这会终止时钟  $f_{\text{ADCI}}$  的产生、从而降低功耗。只有再次使能模拟部分 (ANON = 1) 才可能进行转换。唤醒时间大约为 100 ns。

禁止 ADC 模拟部分的寄存器描述见 [章节 17.7.1](#)。

如果完全不需要 ADC 功能，可通过关闭输入时钟 ( $f_{\text{ADC}}$ ) 将其完全关闭，从而最大程度降低功耗。通过置位寄存器 PMCON1 中的 ADC\_DIS 来实现，具体描述如下。外设时钟管理的具体内容请参见 [章节 8.1.4](#)。

**PMCON1**  
 功率模式控制寄存器 1 (B5<sub>H</sub>) 复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
0	CDC_DIS	CAN_DIS	MDU_DIS	T2CCU_DIS	CCU_DIS	SSC_DIS	ADC_DIS
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
ADC_DIS	0	rw	<b>ADC 禁止请求，高有效</b> 0 <sub>B</sub> ADC 正常工作（缺省状态） 1 <sub>B</sub> 请求禁止 ADC
0	7	r	<b>保留</b> 读操作返回 0；应写入 0。

## 17.4 功能描述

ADC 模块的功能包括：

- 带有独立寄存器的两个不同的（串行或者并行）转换请求源。请求源用来触发由外部事件（与 PWM 信号同步）、序列方案等引起的转换。
- 仲裁器规则地扫描请求源，以确定下次对优先级最高的通道进行转换。每个请求源的优先级可单独设定，其高度灵活性可满足不同应用领域的需求。
- 八路通道中每路的控制寄存器定义了每种模拟输入的行为（例如中断行为、结果寄存器指针、通道级别指针等）。
- 输入综合控制寄存器给出一般的通道控制信息（采样时间）。
- 四个结果寄存器（并非每路模拟输入通道对应一个结果寄存器）用来保存转换结果并控制数据压缩。
- 转换结果的抽取阶段，将新的转换值累加到目标结果寄存器所保存的转换结果上，从而即使 CPU 低频工作，仍可快速、连续进行转换而且没有数据损失的危险。

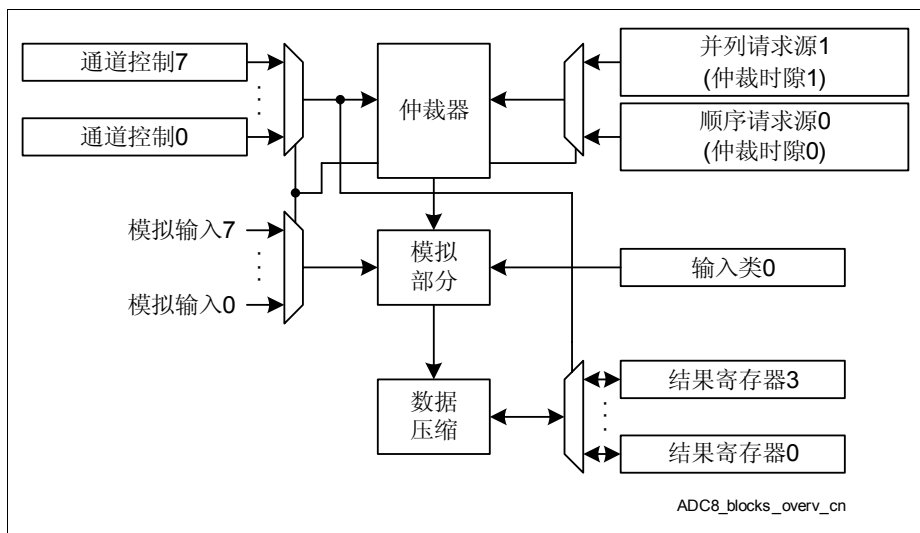


图 17-4 ADC 框图

### 17.4.1 请求源仲裁器

仲裁器可工作在两种模式，由 ARBM 进行选择：

- 持续仲裁模式：  
该模式下，即使没有挂起的转换请求，仲裁器仍连续查询转换请求。
- 由挂起转换请求启动仲裁：  
该模式下，至少有一个转换请求挂起时，仲裁器才开始查询转换请求。

## 模数转换器

一旦开始仲裁，仲裁器查询两个转换请求源（仲裁时隙  $x$  的请求源  $x$ ,  $x = 0 - 1$ ）以确定优先级最高的转换模拟通道。每个仲裁时隙，仲裁器查询每一个请求源请求挂起信号（REQPND）和通道编号有效信号（REQCHNRV）。所有仲裁时隙的总和称为一个仲裁周期。仲裁时隙在参与仲裁之前必须被使能（ASENx = 1）。

每个请求源由位 PRIOx 设定请求源优先级。仲裁器从请求源 0（仲裁时隙 0）开始，检查请求源是否有转换请求挂起（REQPND = 1）。如果发现不止一个同优先级的请求源有挂起请求，则选择对首先发现的请求源所指定的通道进行转换。仲裁器还要检查 REQCHNRV 信号，只有当 REQCHNRV = 1（且 REQPND = 1）时才可进行转换。如果两个请求源设定的优先级相同，由于请求源 0 和仲裁时隙 0 相连，则先对请求源 0 所指定的通道进行转换。

一个完整的仲裁周期  $t_{ARB}$  固定为：

$$t_{ARB} = 4 * t_{ADCD} \quad (17.3)$$

中断优先级和仲裁控制的寄存器描述请参见[章节 17.7.2](#)。

### 17.4.2 转换启动模式

每个仲裁周期结束时，仲裁器应已确定了优先级最高、并有挂起请求的请求源。仲裁器会保存仲裁结果，即通道编号、采样时间以及目标结果寄存器，为下面的工作做准备。如果模拟部分处于空闲状态，则立即开始转换。如果模拟部分正在进行转换，仲裁结果和当前转换的优先级进行比较。如果当前转换的优先级更高或者同级，将继续完成当前转换，该转换结束后立即开始下一个转换。只要模拟部分一空闲并且仲裁器送出转换请求，就开始进行转换。

如果新的转换请求（和当前转换相比）具有更高的优先级，有两种转换启动模式（由位 CSMx 来选择， $x = 0 - 1$ ）：

- 等待启动：  
该模式下，当前转换正常完成，该转换结束后立即处理已挂起的转换请求。转换尽快启动。
- 取消 - 插入 - 重复：  
该模式下，一旦发现更高优先级的新转换请求，则立即中止当前转换，并尽快开始新的转换。被中止的转换请求重新保存在申请该转换的请求源中，将参与下轮仲裁。请求源有效时（包括挂起转换或正在处理的转换）优先级不能由软件改变。转换运行的最后 3 个时钟周期内转换不能被中止。

转换启动控制的相关寄存器描述参见[章节 17.7.2](#)。

### 17.4.3 通道控制

每路通道均有各自的控制信息，规定转换结果将保存在哪个目标结果寄存器中（见[章节 17.7.4](#)）。所有通道唯一的公共控制信息为采样时间，由输入综合控制寄存器定义（见[章节 17.7.5](#)）。

## 17.4.4 顺序请求源

顺序请求源就是转换请求顺序进行。支持的转换请求的数量由顺序缓存队列的长度决定（几级队列）。

顺序请求源寄存器描述见[章节 17.7.6](#)。

### 17.4.4.1 概述

在仲裁时隙 0 的顺序请求源请求对通道 0 至 7 之间的通道顺序转换。队列寄存器保存请求转换的通道编号及一些附加控制信息，从而转换顺序不受通道次序的限制，可自由编程设定。附加控制信息（在转换请求通道完成转换时）使能请求源中断和自动重新添加过程。

顺序请求源由 4 级队列寄存器，一级备份寄存器（QBUR0）和一个模式控制寄存器（QMR0）组成。最新请求的转换被中止后，备份寄存器保存其信息。如果备份寄存器中含有被中止的转换请求（ $V = 1$ ），在处理队列中的转换请求之前，先处理该转换请求；也就是说，只有当备份寄存器中的位 V 被清零后才可开始进行队列中的请求转换。如果备份寄存器中的位 V 没有被置位，当转换开始时队列寄存器 0 中的位 V 被复位。如果备份寄存器或队列寄存器中含有有效的转换请求（ $V = 1$ ），请求源可参与转换请求源仲裁。

*注：4 级队列寄存器中，只有队列 0 寄存器可以读取，其它队列寄存器为内部寄存器。*

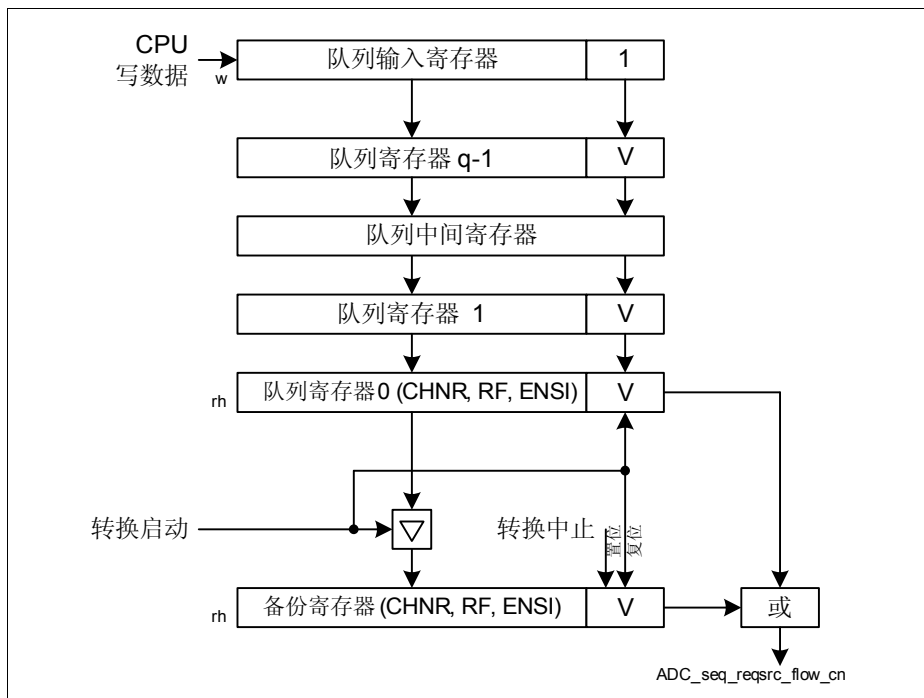


图 17-5 多级寄存器队列

激活自动重新添加特性 ( $RF = 1$ )，允许在成功执行转换后（转换开始），在队列中自动重新插入该挂起请求。否则，挂起请求一旦执行完后将被丢弃。如果使能自动重新添加特性，软件不应在队列输入寄存器写入数据。

顺序转换请求的入口寄存器为只写队列输入寄存器 ( $QINR0$ )。如果队列寄存器为空 ( $V = 0$ )，写入数据将被保存（位  $V$  变  $1$ ），否则写操作被忽略。若被请求的转换开始之后被中止，其设置保存在备份寄存器中（位  $V$  变为  $1$ ）。

顺序请求源寄存器的描述见[章节 17.7.6](#)。



#### 17.4.4.2 请求源控制

如果请求源请求的转换和外部触发事件无关 ( $EXTR = 0$ )，若有效位  $V = 1$ ，通过对信号  $REQPND$  和  $REQCHNRV$  置 1 直接请求转换。在这种情况下，若  $V = 0$  则不请求转换。门控机制使用户能够由位  $ENGT$  来使能 / 禁止转换请求。

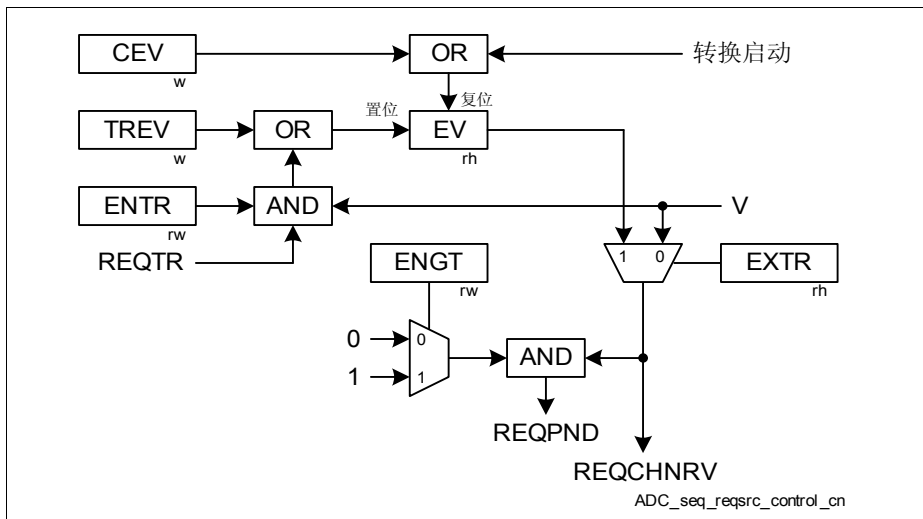


图 17-6 顺序请求源控制

如果转换和外部触发事件有关 ( $EXTR = 1$ )，将考虑信号  $REQTR$  ( $ENTR = 1$ ) 或者软件控制写入  $TREV = 1$ 。这两个操作都会置位事件标志  $EV$ 。事件标志  $EV = 1$  表明已发生外部事件，可请求转换（只有当转换请求有效  $V = 1$  时  $EV$  才可被置位）。在这种情况下，信号  $REQCHNRV$  来自位  $EV$ 。

在队列备份寄存器中，位  $EXTR$  始终视为 0。如果由队列控制的某转换开始后，被一个更高优先级的转换所中止，被中止的转换不需要等待新的触发事件可重新开始。

### 17.4.5 并行请求源

并行请求源并行产生一个或多个通道转换请求。始终按照预先定义的顺序（高路通道在低路通道之前）依次处理转换请求。

并行源寄存器描述见[章节 17.7.7](#)。

#### 17.4.5.1 概述

在仲裁时隙 1 的并行请求源将并行产生一路或多路通道 4 至通道 7 之间的转换请求。转换请求始终按照预先定义的次序（高路通道在低路通道之前）一个接一个（在分开的仲裁周期中）处理。

并行请求源由转换请求控制寄存器（CRCR1）、转换请求挂起寄存器（CRPR1）和转换请求模式寄存器（CRMR1）组成。当选定的加载事件（LED）发生时，转换请求控制寄存器中的内容被复制（覆盖）到转换请求挂起寄存器中。事件类型定义了请求源的行为以及触发方式。

如果转换挂起寄存器的值不为 0，送至仲裁器的转换请求可能被激活。值为 1 的最高挂起位指定了转换通道编号。要参与请求源仲裁，信号 REQPNDR 和 REQCHNRV 必须为 1。

并行请求源寄存器的描述参见[章节 17.7.7](#)。

### 17.4.5.2 请求源控制

所有的转换请求挂起位逻辑或得到中间信号 PND，用来产生 REQCHNRV 和 REQPND。信号 PND 由位 ENGT 控制，使用户能够使能 / 禁止转换请求。见图 17-7。

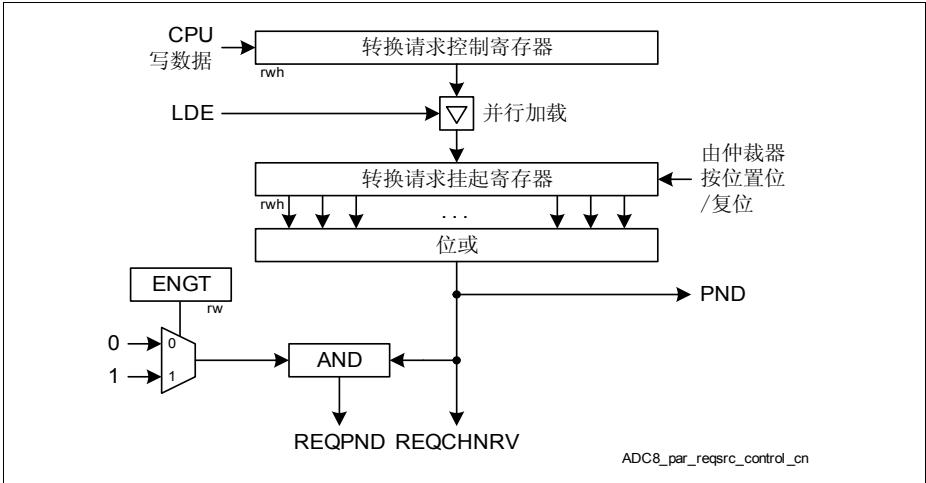


图 17-7 并行请求源控制

并行加载的加载事件包括：

- 输入线 REQTR 上的外部触发，见章节 17.4.5.3。
- 在转换请求控制寄存器的特定地址上的写操作，见章节 17.4.5.4。
- 在请求源模式寄存器中写入 LDEV = 1，见章节 17.4.5.4。
- 请求源内部操作（转换完成且 PND = 0 时的自动扫描模式），见章节 17.4.5.5。

转换请求控制 / 挂起寄存器中的每一位（位 x, x = 4 - 7）对应一路模拟输入通道。每位所在的位置直接对应通道编号。转换请求挂起寄存器中的各位可由仲裁器按位置位或复位：

- 当仲裁器指示某通道开始转换，转换请求挂起寄存器中的相应位自动复位。
- 当仲裁器指示某通道转换被中止，转换请求挂起寄存器中的相应位自动置位。

转换完成且 PND = 0 时，可产生相应的请求源中断（如果中断被使能）。这些规则仅适用于请求源已触发转换的情况。

### 17.4.5.3 外部触发

并行请求源（以及顺序请求源）的转换请求可和外部触发事件同步。对于并行请求源，通过将加载事件耦合到请求触发输入 REQTR 上实现。

#### 17.4.5.4 软件控制

并列请求源的加载事件还可以由软件控制，有两种方式：

- 可从两个不同的地址（**CRCR1** 和 **CRPR1**）上对转换请求控制寄存器写入。访问 **CRCR1** 时，写操作仅改变该寄存器中的位值。访问 **CRPR1** 时，写操作的下一个时钟周期将产生加载事件。只用一条 **MOVE** 指令该自动加载事件即可启动转换。在这种情况下，即将转换的通道信息在 **MOVE** 指令中作为参数给出。
- 可由软件对位 **LDEV** 写入 1 触发加载事件。在这种情况下，加载事件不包含任何要转换通道的信息，始终以转换请求控制寄存器中的通道信息作为目标转换通道。从而能够在另一个地址上对转换请求控制寄存器写入、而不触发加载事件。

#### 17.4.5.5 自动扫描

并列请求源具有自动扫描功能。如果使能自动扫描模式，若并列请求源触发了转换，转换完成且 **PND = 0** 时产生加载事件。无需外部触发或软件操作，该自动重载特性可持续扫描通道 4 至通道 7，查询是否有挂起的转换请求。

### 17.4.6 待读模式

待读模式适用于所有的转换请求源，使 CPU 能够独立处理每次的转换结果而没有丢失数据的危险。如果结果寄存器中的转换结果还未被 CPU 读出就已被新转换结果覆盖，会产生数据丢失。

待读模式下，如果目标结果寄存器已含有有效数据（由被置位的有效标志指示），特定通道上请求源产生的转换请求将被禁止（不可能进行转换）。直到目标结果寄存器中的有效标志被清零（数据无效），才可开始对请求通道进行模数转换。置位 WFR（参见[章节 17.7.8](#)）可使能结果寄存器的待读模式。

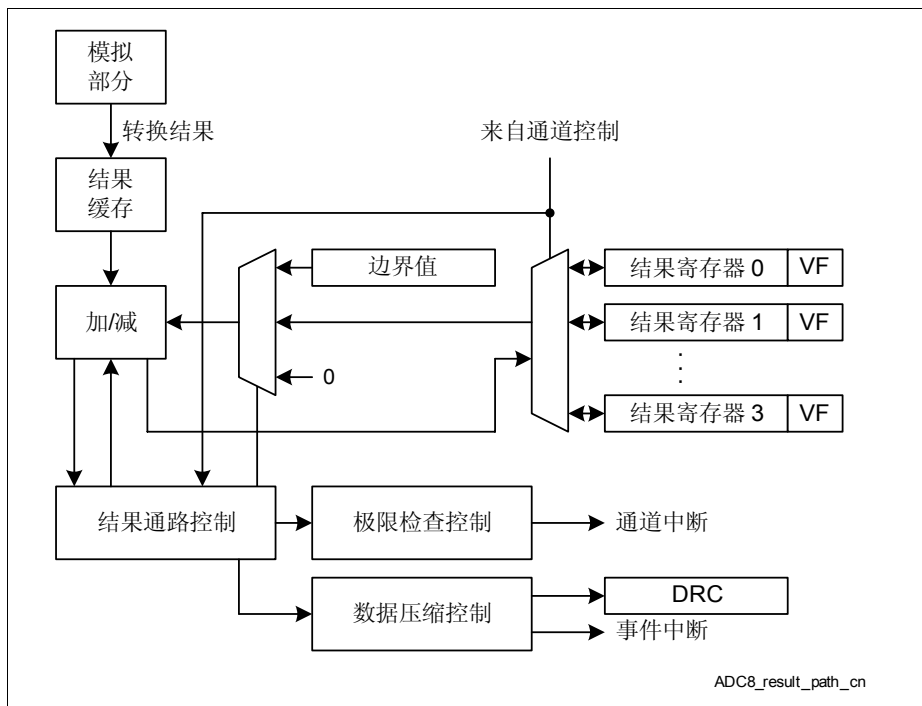
### 17.4.7 转换结果的产生

结果产生部分处理转换结果保存、数据压缩、极限检查和中断产生。

#### 17.4.7.1 概述

ADC 结果的产生由以下几部分组成：

- 极限检查单元，将转换结果和两个选定的边界值（BOUND0 和 BOUND1）比较。根据极限检查结果可产生通道中断。
- 数据压缩滤波器，累加转换结果。将新的转换结果累加到选定的结果寄存器中已保存的转换结果上。
- 四个结果寄存器，保存转换结果。转换结果可由软件从结果寄存器中读出。为每路输入通道分别选择保存转换结果的结果寄存器。



**图 17-8 结果通道**

结果寄存器描述参见[章节 17.7.8](#)。

### 17.4.7.2 极限检查

极限检查和数据压缩滤波基于同一个加 / 减结构。产生的转换结果先和 BOUND0 比较，再和 BOUND1 比较。极限检查单元可根据结果标志（“小于”比较）产生通道中断。当数据压缩滤波的有效结果保存到选定的结果寄存器中，极限检查生效。

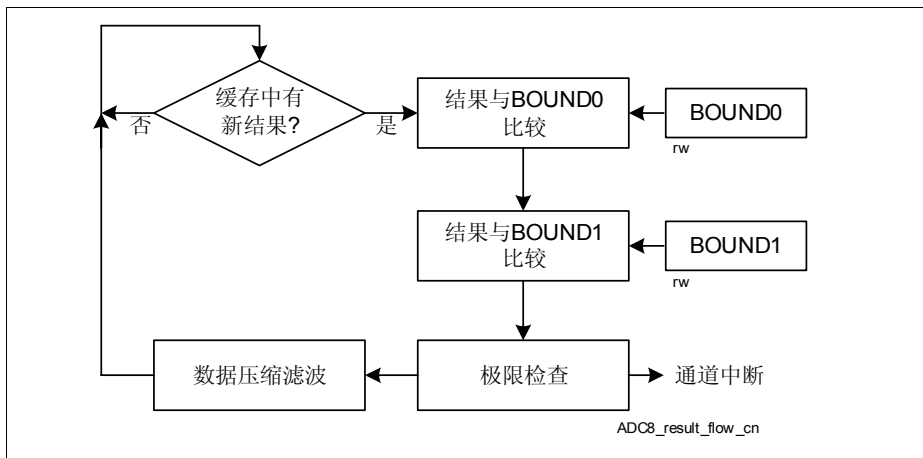


图 17-9 极限检查流程

### 17.4.7.3 数据压缩滤波

每个结果寄存器均可被控制使能或禁止数据压缩滤波。数据压缩模块可累加转换结果，用于抗混迭滤波或结果平均。

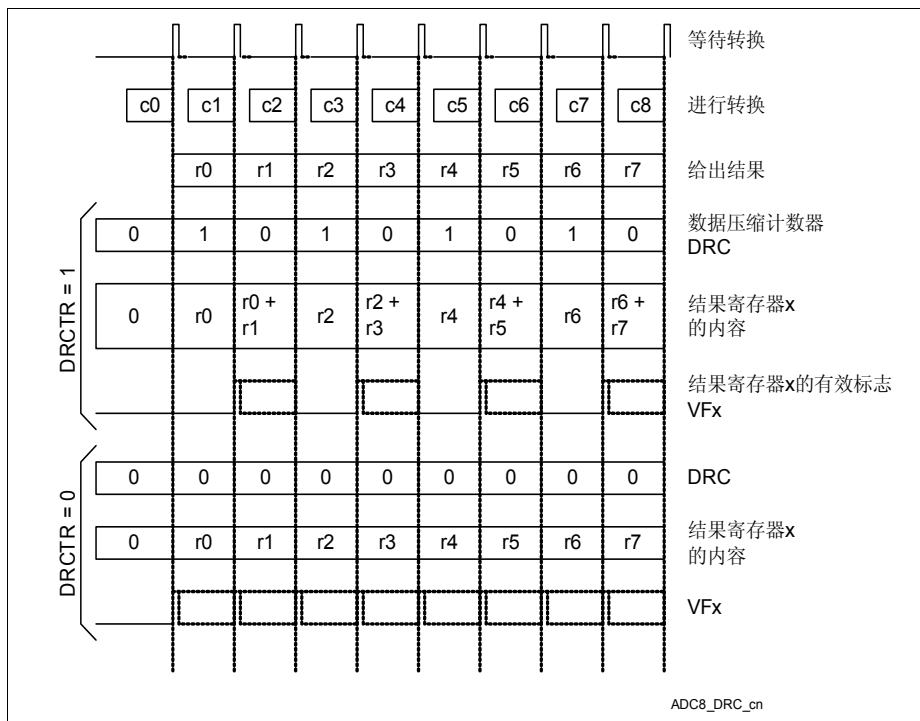


图 17-10 数据压缩流程

如果 DRC 为 0 且有新转换结果产生，（由结果控制寄存器中位 DRCTR 所决定的）重载值重新装入 DRC，将 DRC 为 0 时的转换结果叠加到结果寄存器已保存的转换结果上（取代结果寄存器中的前次内容）。于是完整的转换结果被保存在结果寄存器中。如果重载值为 0（数据压缩滤波被禁止），每次的转换结果都被保存在结果寄存器中，从而产生结果事件、置位结果寄存器的有效标志（VF）。如果重载值为 1（数据压缩滤波被使能），将两次的转换结果累加，此时不产生结果事件、不置位结果寄存器的有效标志 VF。

如果 DRC 为 1 且有新转换结果产生，数据压缩滤波将新的转换结果叠加到结果寄存器已保存的转换结果上，DRC 减 1。叠加后完整的结果保存在结果寄存器中。此时产生结果事件、置位结果寄存器的有效标志。

使能或禁止数据压缩滤波，有可能使到达结果寄存器的通路的周期相同。此外，由于最多可叠加两个转换结果（一个 10 位结果相加两次最大得到 11 位结果），因此可避免结果寄存器溢出。



### 17.4.7.4 结果寄存器的读取

为了适用于广泛的应用领域，结果寄存器  $x$  ( $x = 0 - 3$ ) 的内容可从不同的地址上以不同方式读取（见图 17-11）：

- 正常读取 RESR $x$ L/H：  
此方式读取 8 位或 10 位转换结果。
- 累加读取 RESR $x$ L/H：  
此方式读取累加的 9 位或 11 位转换结果。

所有（有累加或无累加）的转换结果均保存在结果寄存器中，可从具有不同的数据排列及数据宽度的寄存器 RESR $x$ L/H 或 RESR $x$ L/H 中读取。

当使能数据压缩滤波（DRCTR = 1），用户应从 RESR $x$ L/H 中读取数据，因为该寄存器中保存着 9 位（R8:R0）或 11 位（R10:R0）的累加转换结果。读取 RESR $x$ L/H 给出附加（无 MSB）的累加结果。

当禁止数据压缩滤波（DRCTR = 0），用户可从 RESR $x$ L/H 或 RESR $x$ L/H 中读取 8 位或 10 位的转换结果。尤其对于（无累加的）8 位转换，用一条指令即可从 RESR $x$ H 中读出结果。因此，用户可根据应用要求选择不同的结果读取方式。

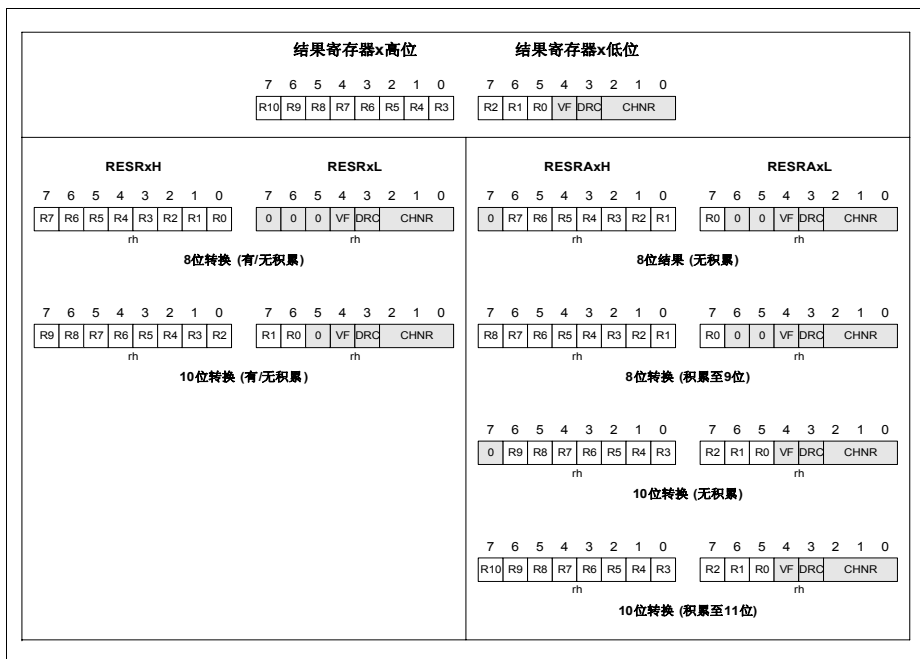


图 17-11 结果寄存器读取

### 17.4.8 中断

ADC 模块提供了两条可由不同的中断源激活的中断服务请求输出线 SR [1:0]。

ADC 的中断结构支持两种中断源：

- 事件中断：由转换请求源事件（请求源中断）或结果寄存器事件（结果中断）激活。
- 通道中断：任何输入通道转换完成时该中断激活。根据极限检查控制位使能通道中断，每路输入通道独立设定。

送入中断压缩单元的所有中断脉冲逻辑或之后送给每条 SR 输出线。

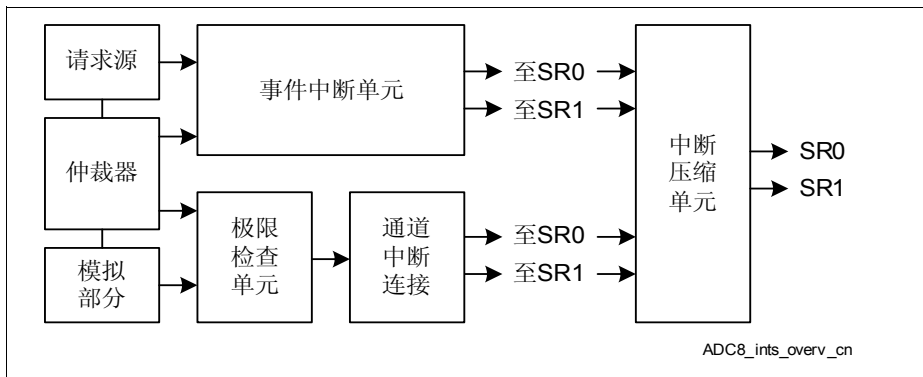


图 17-12 中断概览

中断寄存器的描述参见[章节 17.7.9](#)。

### 17.4.8.1 事件中断

请求源和结果寄存器可产生事件中断。事件中断使能位存放在请求源寄存器的位 **ENSI** 和结果控制寄存器的位 **IEN** 中。可由每个事件的中断节点指针（**EVINP**）选择中断服务请求输出线。

当请求通道转换完毕后产生请求源事件：

- 事件 0：顺序请求源 0（仲裁时隙 0）的请求源事件
- 事件 1：并列请求源 1（仲裁时隙 1）的请求源事件

根据数据压缩控制（参见[章节 17.4.7.3](#)）产生结果寄存器事件：

- 事件 4：结果寄存器 0 的结果寄存器事件
- 事件 5：结果寄存器 1 的结果寄存器事件
- 事件 6：结果寄存器 2 的结果寄存器事件
- 事件 7：结果寄存器 3 的结果寄存器事件

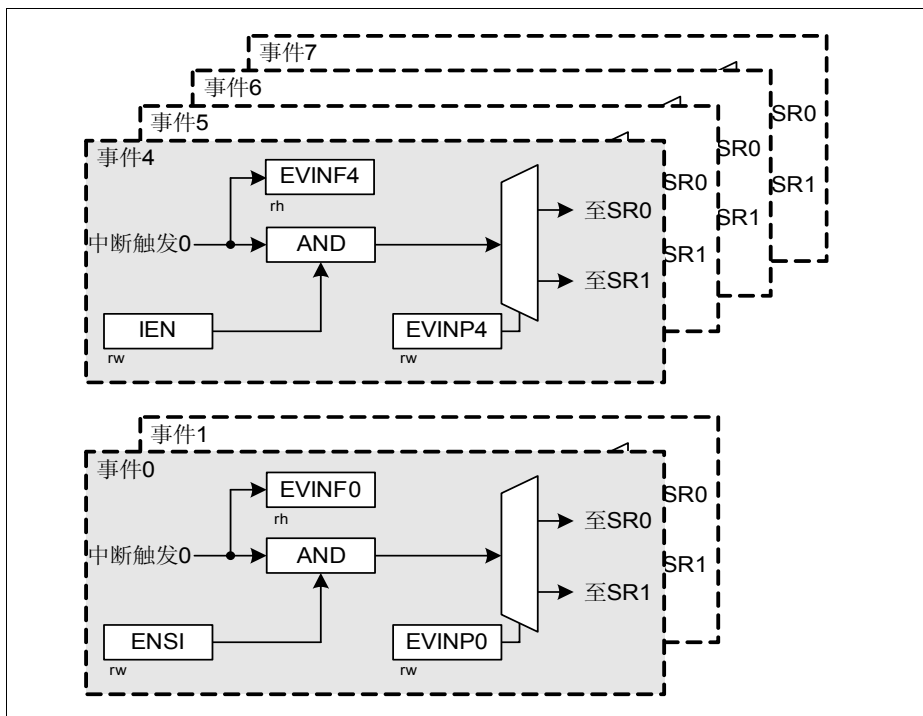


图 17-13 事件中断结构

### 17.4.8.2 通道中断

当转换完成并满足所选择的极限检查条件时产生通道中断。因此一次只有一路通道中断可以被激活。将每路通道的转换结果和两个可选边界相比较，根据极限检查的结果触发中断。

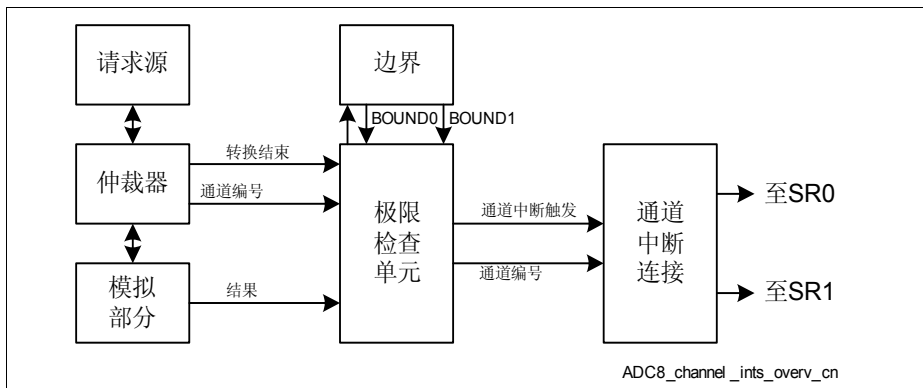


图 17-14 通道中断概览

极限检查单元将转换结果和两个边界（BOUND0 和 BOUND1）进行比较。转换结果空间被划分为 3 个区域：

- 区 I：转换结果小于两个边界值。
- 区 II：转换结果在两个边界值之间，或者与其中一个边界值相等。
- 区 III：转换结果大于两个边界值。

转换完成后，可根据下列条件（由极限检查控制位域 LCC 选择）触发通道中断：

- LCC = 000：不触发中断、禁止通道中断。
- LCC = 001：转换结果不在区 I 时产生通道中断。
- LCC = 010：转换结果不在区 II 时产生通道中断。
- LCC = 011：转换结果不在区 III 时产生通道中断。
- LCC = 100：始终产生通道中断（和边界无关）。
- LCC = 101：转换结果在区 I 时产生通道中断。
- LCC = 110：转换结果在区 II 时产生通道中断。
- LCC = 111：转换结果在区 III 时产生通道中断。

产生通道中断后，由特定通道的中断节点指针 CHINPx（x = 0 - 7）来选择中断服务请求输出线（SR[1:0]），见图 17-15。

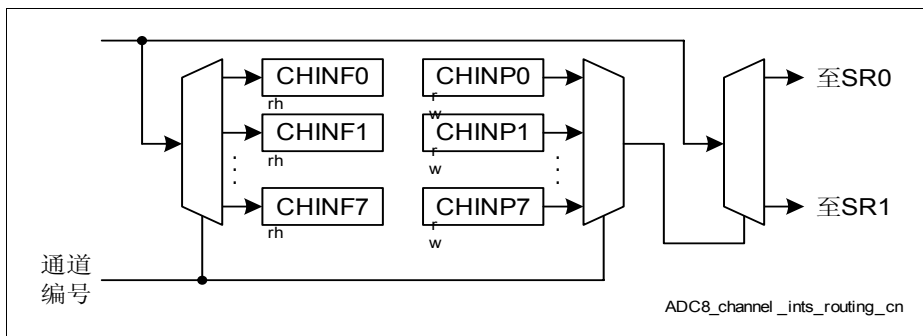


图 17-15 通道中断连接

### 17.4.9 外部触发输入

顺序请求源和并列请求源各有一个请求触发输入  $REQTRx$  ( $x = 0 - 1$ )，通过该外部触发输入可启动转换请求。根据位域  $ETRSELx$  的值，多路选择器从八路外部触发输入 ( $ETR0$  至  $ETR7$ ) 中选择一路送至  $REQTRx$ 。若外部触发请求和 ADC 同步，可不通过同步模块（旁路），该选择由位  $SYNENx$  控制。

外部触发控制寄存器的描述参见[章节 17.7.9](#)。

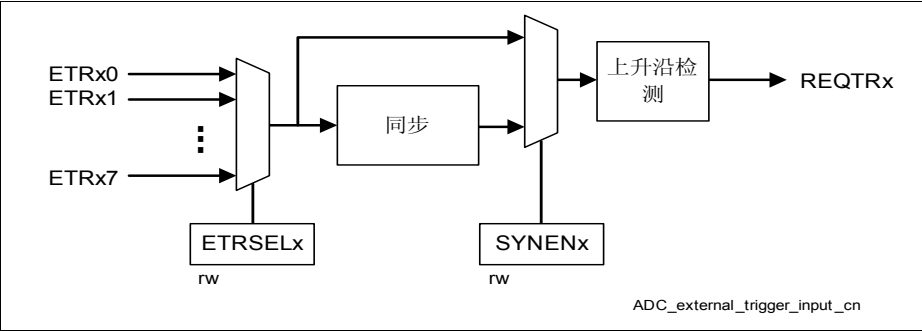


图 17-16 外部触发输入

ADC 的外部触发输入由 CCU6 或 T2CCU 模块产生的事件驱动，总结见[表 17-2](#)。ETR0/1 的输入由用户通过寄存器 MISC\_CON 中的对应位进行选择。

表 17-2 外部触发输入源

外部触发输入	CCU6/T2CCU 事件
ETR0	定时器 T13 周期 - 匹配或 CCT 溢出
ETR1	定时器 T13 比较 - 匹配或 T2CC5 比较匹配
ETR2	定时器 T12 周期 - 匹配
ETR3	通道 0 定时器 T12 比较 - 匹配
ETR4	通道 1 定时器 T12 比较 - 匹配
ETR5	通道 2 定时器 T12 比较 - 匹配
ETR6	多通道模式的映射传送事件
ETR7	多通道模式的正确霍尔事件

MISC\_CON

其它控制寄存器

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
ADCETR0 _MUX	ADCETR1 _MUX			0			DFLASHEN
w	rw			r			rwh

符号	位	类型	描述
ADCETR1 _MUX	6	rwh	<b>ADC 外部触发输入 1 事件选择</b> 0 <sub>B</sub> 来自 CCU6 的 T13CM (缺省设置) 1 <sub>B</sub> 来自 T2CCU 的 CM5F
ADCETR0 _MUX	7	w	<b>ADC 外部触发输入 0 事件选择</b> 0 <sub>B</sub> 来自 CCU6 的 T13PM (缺省设置) 1 <sub>B</sub> 来自 T2CCU 的 CCTOVF
0	[5:1]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 17.5 ADC 模块初始化步骤

下列步骤旨在为用户提供初始化 ADC 模块的一般步骤。根据应用需求的不同，一些步骤可相应改变或省略：

- 配置全局控制功能：
  - 选择转换宽度（GLOBCTR.DW）
  - 选择模拟时钟  $f_{\text{ADCI}}$  分频因子（GLOBCTR.CTC）
- 配置仲裁控制功能：
  - 选择请求源  $x$  的优先级（PRAR.PRIO $x$ ）
  - 选择请求源  $x$  的转换启动模式（PRAR.CSM $x$ ）
  - 使能仲裁时隙  $x$ （PRAR.ASEN $x$ ）
  - 选择仲裁模式（PRAR.ARB $M$ ）
- 配置通道控制信息：
  - 选择通道  $x$  极限检查控制（CHCTR $x$ .LCC）
  - 选择通道  $x$  目标结果寄存器（CHCTR $x$ .RESRSEL）
  - 选择所有通道的采样时间（INPCR0.STC）
- 配置结果控制信息：
  - 使能 / 禁止结果寄存器  $x$  数据压缩（RCR $x$ .DRCTR）
  - 使能 / 禁止结果寄存器  $x$  事件中断（RCR $x$ .IEN）
  - 使能 / 禁止结果寄存器  $x$  待读模式（RCR $x$ .WFR）
  - 使能 / 禁止结果寄存器  $x$  读操作复位有效标志（RCR $x$ .VFCTR）
- 配置中断控制功能：
  - 选择通道  $x$  的中断节点指针（CHINPR.CHINP $x$ ）
  - 选择事件  $x$  的中断节点指针（EVINPR.EVINP $x$ ）
- 配置极限检查边界：
  - 选择所有通道的极限检查边界（LCBR.BOUND0, LCB $R$ .BOUND1）
- 配置外部触发控制功能：
  - 选择请求源  $x$  外部触发输入（ETRCR.ETRSEL $x$ ）
  - 使能 / 禁止请求源  $x$  外部触发输入同步功能（ETRCR.SY $NENx$ ）
- 设置顺序请求源：
  - 使能转换请求（QMR0.ENG $T$ ）
  - 使能 / 禁止外部触发（QMR0.ENT $R$ ）
- 设置并列请求源：
  - 使能转换请求（CRMR1.ENG $T$ ）
  - 使能 / 禁止外部触发（CRMR1.ENT $R$ ）
  - 使能 / 禁止请求源中断（CRMR1.ENS $I$ ）
  - 使能 / 禁止自动扫描（CRMR1.SCAN）
- 开启模拟部分：
  - 置位 GLOBCTR.ANON（等待 100 ns）
- 启动顺序请求：
  - 将（REQCHNR, RF, ENSI 和 EXTR 等信息）写入到 QINR0
  - 用 [章节 17.4.4.2](#) 描述的任何方式产生挂起的转换请求
- 启动并列请求：



## 模数转换器

- 将要转换的通道编号写入到 **CRCR1**（无加载事件）或 **CRPR1**（自动加载事件）
- 用 **章节 17.4.5.2** 描述的任何方式产生加载事件（若加载事件还未产生），用来触发挂起的转换请求。
- 等到 **ADC** 完成转换：
  - 请求源中断指示由该请求源请求的转换已完成
  - 通道中断指示相应的通道转换已完成（并已执行极限检查）
  - 结果中断指示相应结果寄存器中（有 / 无累加）的结果已准备就绪，可供读取。
- 读 **ADC** 结果

## 17.6 寄存器映射

以下描述的所有 ADC 寄存器，在本手册其它章节中应用是需要添加模块名前缀“ADC\_”，例如 ADC\_GLOBCTR。

ADC SFR 的地址见表 17-3 和表 17-4。

**表 17-3 页 0 - 3 的 SFR 地址列表**

地址	页 0	页 1	页 2	页 3
CA <sub>H</sub>	GLOBCTR	CHCTR0	RESR0L	RESRA0L
CB <sub>H</sub>	GLOBSTR	CHCTR1	RESR0H	RESRA0H
CC <sub>H</sub>	PRAR	CHCTR2	RESR1L	RESRA1L
CD <sub>H</sub>	LCBR	CHCTR3	RESR1H	RESRA1H
CE <sub>H</sub>	INPCR0	CHCTR4	RESR2L	RESRA2L
CF <sub>H</sub>	ETRCR	CHCTR5	RESR2H	RESRA2H
D2 <sub>H</sub>	—	CHCTR6	RESR3L	RESRA3L
D3 <sub>H</sub>	—	CHCTR7	RESR3H	RESRA3H

**表 17-4 页 4 - 7 的 SFR 地址列表**

地址	页 4	页 5	页 6	页 7
CA <sub>H</sub>	RCR0	CHINFR	CRCR1	—
CB <sub>H</sub>	RCR1	CHINCR	CRPR1	—
CC <sub>H</sub>	RCR2	CHINSR	CRMR1	—
CD <sub>H</sub>	RCR3	CHINPR	QMR0	—
CE <sub>H</sub>	VFRCR	EVINFR	QSR0	—
CF <sub>H</sub>	—	EVINCR	Q0R0	—
D2 <sub>H</sub>	—	EVINSR	QBUR0/QINR0	—
D3 <sub>H</sub>	—	EVINPR	—	—

# 模数转换器

ADC 的 SFR 位于标准存储器区（RMAP = 0），由七页构成。ADC\_PAGE 寄存器位于地址单元 D1<sub>H</sub>，包含页信息和分页控制信息。

## ADC\_PAGE

ADC 分页寄存器

(D1<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
PAGE	[2:0]	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。
STNR	[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。 如果 OP = 10 <sub>B</sub> ， PAGE 的内容在被新值覆盖之前保存在 STx 中。 如果 OP = 11 <sub>B</sub> ， PAGE 的内容被 STx 覆盖。写入 PAGE 的值不予理睬。 00 <sub>B</sub> 选择 ST0 01 <sub>B</sub> 选择 ST1 10 <sub>B</sub> 选择 ST2 11 <sub>B</sub> 选择 ST3
OP	[7:6]	w	<b>操作</b> 0X <sub>B</sub> 手动保存页模式，STNR 的值被忽略，PAGE 被直接写入 10 <sub>B</sub> 带有自动保存页的新页设置。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 STx 中。 11 <sub>B</sub> 自动恢复页。对写入 PAGE 的内容不予理睬，PAGE 的内容由 STNR 指定的位域 STx 中的值覆盖。
0	3	r	<b>保留</b> 读操作返回 0；应写入 0。

# 17.7 寄存器描述

本节描述和 ADC 模块功能相关的所有寄存器。

## 17.7.1 基本功能寄存器

寄存器 GLOBCTR 控制模拟部分工作和转换时间。

**GLOBCTR**  
全局控制寄存器 (CA<sub>H</sub>) 复位值: 30<sub>H</sub>

7	6	5	4	3	2	1	0
<b>ANON</b>	<b>DW</b>	<b>CTC</b>				<b>0</b>	
rw	rw	rw				r	

符号	位	类型	描述
<b>CTC</b>	[5:4]	w	<b>转换时间控制</b> 该位域定义了内部模拟时钟 $f_{ADCI}$ 的分频因子。该时钟提供了模拟转换和采样时间的内部时间基准。 00 <sub>B</sub> $f_{ADCI} = 1/2 \times f_{ADCA}$ 01 <sub>B</sub> $f_{ADCI} = 1/3 \times f_{ADCA}$ 10 <sub>B</sub> $f_{ADCI} = 1/4 \times f_{ADCA}$ 11 <sub>B</sub> $f_{ADCI} = 1/32 \times f_{ADCA}$ (缺省值)
<b>DW</b>	6	rw	<b>数据宽度</b> 该位定义了转换精度。 0 <sub>B</sub> 结果为 10 位宽 (缺省值) 1 <sub>B</sub> 结果为 8 位宽
<b>ANON</b>	7	rw	<b>模拟部分开启控制</b> 该位使能 ADC 的模拟部分并定义其工作模式。 0 <sub>B</sub> 模拟部分关闭, 不能进行转换。 为了使功耗最低, 内部模拟电路处于掉电模式, 停止产生 $f_{ADCI}$ 。 1 <sub>B</sub> 模拟部分开启, 可以进行转换。模拟部分的自动掉电功能被关闭。
<b>0</b>	[3:0]	r	<b>保留</b> 读操作返回 0; 应写入 0。

## 模数转换器

寄存器 GLOBSTR 指示当前的转换状态。

**GLOBSTR**

全局状态寄存器

(CB<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
0		CHNR			0	SAMPLE	BUSY
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> 正在进行转换
<b>SAMPLE</b>	1	rh	<b>采样阶段</b> 该位指示正在对模拟输入信号采样。 0 <sub>B</sub> 模拟部分未处于采样阶段 1 <sub>B</sub> 模拟部分正处于采样阶段
<b>CHNR</b>	[5:3]	rh	<b>通道编号</b> 该位域指示正在对哪路模拟输入通道进行转换。开始新的转换时位域更新。
<b>0</b>	2, [7:6]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 17.7.2 优先级和仲裁寄存器

寄存器 PRAR 定义请求源的优先级和转换启动模式，并控制使能 / 禁止在仲裁时隙进行转换请求处理。

### PRAR

优先级和仲裁寄存器

(CC<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
ASEN1	ASEN0	0	ARBM	CSM1	PRI01	CSM0	PRI00
rw	rw	r	rw	rw	rw	rw	rw

符号	位	类型	描述
PRI00	0	rw	<b>请求源 0 的优先级</b> 该位定义了顺序请求源 0 的优先级。 0 <sub>B</sub> 低优先级 1 <sub>B</sub> 高优先级
CSM0	1	rw	<b>请求源 0 的转换启动模式</b> 该位定义了顺序请求源 0 的转换启动模式。 0 <sub>B</sub> 选择等待开始模式 1 <sub>B</sub> 选择取消 - 插入 - 重复模式
PRI01	2	rw	<b>请求源 1 的优先级</b> 该位定义了并列请求源 1 的优先级。 0 <sub>B</sub> 低优先级 1 <sub>B</sub> 高优先级
CSM1	3	rw	<b>请求源 1 的转换启动模式</b> 该位定义了并列请求源 1 的转换启动模式。 0 <sub>B</sub> 选择等待开始模式 1 <sub>B</sub> 选择取消 - 插入 - 重复模式
ARBM	4	rw	<b>仲裁模式</b> 该位定义了选择哪种仲裁模式 0 <sub>B</sub> 持续仲裁 (缺省值) 1 <sub>B</sub> 由挂起的转换请求启动仲裁

**模数转换器**

符号	位	类型	描述
<b>ASENx</b> <b>(x = 0 - 1)</b>	[7:6]	rw	<p><b>使能仲裁时隙 x</b></p> <p>这两位分别使能仲裁周期中的一个仲裁时隙。  <b>ASEN0</b> 使能仲裁时隙 0； <b>ASEN1</b> 使能仲裁时隙 1。            如果仲裁时隙被禁止，该时隙内转换请求源所产生的挂起转换请求不参与仲裁。</p> <p><b>0<sub>B</sub></b>    禁止相应仲裁时隙。不请求转换，与时隙 x 相连的请求源的挂起转换请求不参与仲裁。如果仲裁器不能连续运行（<b>ARBM = 1</b>），与仲裁时隙 x 相连的请求源的转换请求一定不能有效。在禁止仲裁时隙之前，必须清除相应请求源的转换请求。</p> <p><b>1<sub>B</sub></b>    使能相应仲裁时隙。挂起的请求源请求进行转换操作。</p>
<b>0</b>	5	r	<p><b>保留</b></p> <p>读操作返回 0；应写入 0。</p>

### 17.7.3 外部触发控制寄存器

寄存器 ETRCR 选择外部触发输入信号源、使能与外部触发输入同步。

#### ETRCR

外部触发控制寄存器

(CF<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
SYNEN1	SYNEN0	ETRSEL1			ETRSEL0		
rw	rw	rw			rw		

符号	位	类型	描述
<b>ETRSELx</b> (x = 0 - 1)	[2:0], [5:3]	rw	<b>请求源 x 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 ETRx0 001 <sub>B</sub> 选择触发输入 ETRx1 010 <sub>B</sub> 选择触发输入 ETRx2 011 <sub>B</sub> 选择触发输入 ETRx3 100 <sub>B</sub> 选择触发输入 ETRx4 101 <sub>B</sub> 选择触发输入 ETRx5 110 <sub>B</sub> 选择触发输入 ETRx6 111 <sub>B</sub> 选择触发输入 ETRx7
<b>SYNENx</b> (x = 0 - 1)	[7:6]	rw	<b>同步使能</b> 0 <sub>B</sub> 产生外部触发输入 REQTRx 的过程中不执行同步操作 1 <sub>B</sub> 产生外部触发输入 REQTRx 的过程中执行同步操作



## 17.7.4 通道控制寄存器

通道控制寄存器用来选择目标结果寄存器、控制极限检查机制。寄存器 CHCTR<sub>x</sub> 对输入通道  $x$  进行相应设置。

CHCTR<sub>x</sub> ( $x = 0 - 7$ )

通道控制寄存器  $x$

( $CA_H + x * 1$ )

复位值: 00<sub>H</sub>

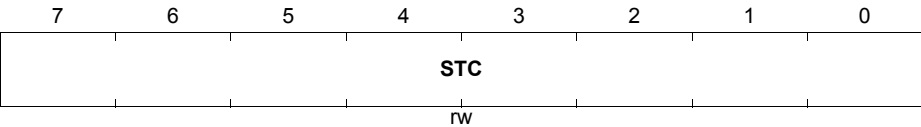
7	6	5	4	3	2	1	0
0	LCC			0	RESRSEL		
r	rw			r	rw		

符号	位	类型	描述
RESRSEL	[1:0]	rw	<b>结果寄存器选择</b> 该位域用来选择由哪个结果寄存器保存通道转换的结果。 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
LCC	[6:4]	rw	<b>极限检查控制</b> 该位域定义了极限检查机制。 编码参见 <a href="#">章节 17.4.8.2</a> 。
0	[3:2], 7	r	<b>保留</b> 读操作返回 0；应写入 0。

17.7.5 输入综合控制寄存器

寄存器 INPCR0 控制输入通道的采样时间。

**INPCR0**  
输入综合控制寄存器 0 (CE<sub>H</sub>) 复位值: 00<sub>H</sub>



符号	位	类型	描述
STC	[7:0]	rw	<b>采样时间控制</b> 该位域定义了附加采样时间，以 $f_{\text{ADCl}}$ 时钟周期为单位。 通过设置该采样值，可扩展两个模拟时钟周期的采样时间。

## 17.7.6 顺序请求源寄存器

这些寄存器中包含顺序请求源 0 的控制位和状态位。

寄存器 QMR0 控制将顺序请求源设定成期望模式。

### QMR0

队列模式寄存器

(CD<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>CEV</b>	<b>TREV</b>	<b>FLUSH</b>	<b>CLRV</b>	<b>0</b>	<b>ENTR</b>	<b>0</b>	<b>ENGT</b>
w	w	w	w	r	rw	r	rw

符号	位	类型	描述
<b>ENGT</b>	0	rw	<b>门控使能</b> 该位使能请求源的门控功能。 0 <sub>B</sub> 门控线始终为 0。请求源关闭。 1 <sub>B</sub> 门控线始终为 1。请求源开启。
<b>ENTR</b>	2	rw	<b>外部触发使能</b> 该位使能外部触发功能。如果该位被使能，寄存器 Q0R0 或 QBUR0 中至少有一个 V 位被置位时，检测到外部触发输入 REQTR 的上升沿时置位 EV。 0 <sub>B</sub> 禁止外部触发 1 <sub>B</sub> 使能外部触发
<b>CLRV</b>	4	w	<b>V 位清零</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 寄存器 Q0R0 或 QBUR0 中的位 V 复位。如果 QBUR0.V = 1，那么复位 QBUR0.V。如果 QBUR0.V = 0，那么 Q0R0.V 复位。
<b>FLUSH</b>	5	w	<b>队列清空</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 队列寄存器中的所有 V 位和 EV 位全部被复位。队列中无有效请求输入。
<b>TREV</b>	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。若请求源等到了触发事件，启动转换请求。
<b>CEV</b>	7	w	<b>事件清零</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 位 EV 被清零

## 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。

寄存器 QSR0 中包含指示顺序请求源状态的标志位。

### QSR0

队列状态寄存器

(CE<sub>H</sub>)

复位值：20<sub>H</sub>

7	6	5	4	3	2	1	0
<b>Rsv</b>	<b>0</b>	<b>EMPTY</b>	<b>EV</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>FILL</b>
r	r	rh	rh	r	r	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 <b>EMPTY</b> = 1，表示队列中无有效的请求源。 00 <sub>B</sub> 如果 <b>EMPTY</b> = 0，表示队列中有 1 个有效请求源 01 <sub>B</sub> 如果 <b>EMPTY</b> = 0，表示队列中有 2 个有效请求源 10 <sub>B</sub> 如果 <b>EMPTY</b> = 0，表示队列中有 3 个有效请求源 11 <sub>B</sub> 如果 <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> 该位指示当 <b>V</b> = 1 时检测到有事件发生。一旦该位被置位，转换开始后该位自动复位。 0 <sub>B</sub> 未检测到事件发生 1 <sub>B</sub> 检测到事件发生
<b>EMPTY</b>	5	rh	<b>队列已空</b> 该位指示顺序请求源队列中是否包含有效的请求输入。如果对列已满 ( <b>EMPTY</b> = 0)，则忽略新的请求输入。 0 <sub>B</sub> 队列中已有 “ <b>FILL</b> +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>Rsv</b>	7	r	<b>保留</b> 读操作返回 1；应写入 0。 <i>注：复位后该位被立即初始化为 0，但随后很快被硬件更新为 1（且保持为 1）。</i>

## 模数转换器

符号	位	类型	描述
0	[3:0], 6	r	保留 读操作返回 0；应写入 0。

寄存器 Q0R0 用来监控顺序请求源的当前状态。

# Q0R0

队列 0 寄存器 0

(CF<sub>H</sub>)

复位值: 00<sub>H</sub>

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	<b>请求转换通道编号</b> 该位域给出将被请求或正被请求的转换通道编号。
V	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。当一个有效输入被写入队列输入寄存器 QINR0 时（或者由中间队列寄存器执行更新），V 被置位。 0 <sub>B</sub> 数据无效 1 <sub>B</sub> 数据有效
RF	5	rh	<b>重填</b> 该位指示当挂起请求处理完毕后（转换开始）被丢弃还是被自动添加到请求队列的顶端。 0 <sub>B</sub> 转换开始后丢弃请求 1 <sub>B</sub> 转换开始后请求重新添加到队列中
ENSI	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，中断激活。 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
EXTR	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
0	3	r	<b>保留</b> 读操作返回 0；应写入 0。

## 模数转换器

寄存器 QBUR0 和 QINR0 共用一个寄存器地址。该寄存器地址上的读操作将对应给出 QBUR0 寄存器的 "rh" 位；相同寄存器地址上的写操作将对应 QINR0 寄存器的 "w" 位。

寄存器 QBUR0 监控被中止的顺序请求源的状态。

### QBUR0

队列备份寄存器 0

(D2<sub>H</sub>)

复位值：00<sub>H</sub>

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	<b>请求转换通道编号</b> 开始执行由 Q0R0 请求的转换时，由位域 Q0R0.REQCHNR 更新该位域。
V	4	rh	<b>请求通道编号有效</b> 该位指示 REQCHNR、RF、ENSI 和 EXTR 是否有效。当正在进行的转换被中止时置位 V。转换开始后该位复位。 0 <sub>B</sub> 备份寄存器中不含有效数据，因为该数据所描述的转换未被中止 1 <sub>B</sub> 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
RF	5	rh	<b>重填</b> 开始执行由 Q0R0 请求的转换时，该位由位 Q0R0.RF 更新。
ENSI	6	rh	<b>请求源中断使能</b> 开始执行由 Q0R0 请求的转换时，该位由位 Q0R0.ENSI 更新。
EXTR	7	rh	<b>外部触发</b> 开始执行由 Q0R0 请求的转换时，该位由位 Q0R0.EXTR 更新。
0	3	r	<b>保留</b> 读操作返回 0；应写入 0。



寄存器 QINR0 是顺序请求源的入口寄存器。

# QINR0

队列输入寄存器 0

(D2<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
EXTR	ENSI	RF	0			REQCHNR	
w	w	w	r			w	

符号	位	类型	描述
REQCHNR	[2:0]	w	请求转换通道编号 该位域定义了被请求的通道编号。
RF	5	w	重填 该位定义了重新添加功能。
ENSI	6	w	请求源中断使能 该位定义了请求源中断功能。
EXTR	7	w	外部触发 该位定义了外部触发功能。
0	[4:3]	r	保留 读操作返回 0；应写入 0。

### 17.7.7 并行请求源寄存器

这些寄存器中包含并行请求源 1 的控制位和状态位。

出现加载事件时，将寄存器 **CRCR1** 的内容复制到挂起寄存器（**CRPR1**）中。可从两个地址访问该寄存器（一个读地址；两个写地址）。第一个读写操作的地址为 **CRCR1** 的地址；第二个写操作的地址对应 **CRPR1** 的地址。对 **CRPR1** 的写操作将使数据写入 **CRCR1** 并在一个时钟周期后自动产生加载事件。

**CRCR1**  
转换请求控制寄存器 1 (CA<sub>H</sub>) 复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>CH7</b>	<b>CH6</b>	<b>CH5</b>	<b>CH4</b>			<b>0</b>	
rwh	rwh	rwh	rwh			r	

符号	位	类型	描述
<b>CHx</b> <b>(x = 4 - 7)</b>	x	rwh	<b>通道位 x</b> 每一位对应一路模拟通道，由寄存器中各位的位置来定义通道编号 <b>x</b> 。加载事件发生时，转换请求挂起寄存器中的相应位 <b>x</b> 由该位覆盖。 0 <sub>B</sub> 并行请求源未请求转换模拟通道 <b>x</b> 1 <sub>B</sub> 并行请求源将请求转换模拟通道 <b>x</b>
<b>0</b>	[3:0]	r	<b>保留</b> 读操作返回 0；应写入 0。

## 模数转换器

寄存器 **CRPR1** 中的控制位用来请求进行相应模拟通道的转换。该寄存器只读；对该地址的写操作将导致数据写入 **CRCR1** 并在一个时钟周期后自动产生加载事件。

### CRPR1

转换请求挂起寄存器 1

(CB<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>CHP7</b>	<b>CHP6</b>	<b>CHP5</b>	<b>CHP4</b>			<b>0</b>	
rwh	rwh	rwh	rwh			r	

符号	位	类型	描述
<b>CHPx</b> (x = 4 - 7)	x	rwh	<b>通道挂起位 x</b> 写操作: 对该地址执行写操作的目标寄存器为 <b>CRCR1</b> 。 读操作: 每一位对应一路模拟通道, 由寄存器中各位的位置来定义通道编号 <b>x</b> 。(开始转换时) 仲裁器自动复位、或 (转换被中止时) 再次置位相应的模拟通道。 0 <sub>B</sub> 并行请求源未请求转换模拟通道 x 1 <sub>B</sub> 并行请求源未请求转换模拟通道 x
<b>0</b>	[3:0]	r	<b>保留</b> 读操作返回 0；应写入 0。

注: 可从该寄存器读取的位通常为“rh”, 不可由写操作直接修改。写操作将修改 **CRCR1** 寄存器中的各位 (故这些位为 “rwh”), 并在一个时钟周期后产生加载事件。

寄存器 CRMR1 将并列请求源设定为期望的模式。

## CRMR1

转换请求模式寄存器 1

(CC<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>Rsv</b>	<b>LDEV</b>	<b>CLRPND</b>	<b>SCAN</b>	<b>ENSI</b>	<b>ENTR</b>	<b>0</b>	<b>ENGT</b>
r	w	w	rw	rw	rw	r	rw

符号	位	类型	描述
<b>ENGT</b>	0	rw	<b>门控使能</b> 该位使能请求源的门控功能。 0 <sub>B</sub> 门控线始终为 0。请求源关闭。 1 <sub>B</sub> 门控线始终为 1。请求源开启。
<b>ENTR</b>	2	rw	<b>外部触发使能</b> 该位使能外部触发功能。如果该位被使能，检测到外部触发输入 REQTR 的上升沿时产生加载事件。 0 <sub>B</sub> 禁止外部触发事件 1 <sub>B</sub> 使能外部触发事件
<b>ENSI</b>	3	rw	<b>请求源中断使能</b> 该位使能请求源中断。当该请求源的最后一个挂起请求执行完毕后（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>自动扫描使能</b> 该位使能自动扫描功能。如果该功能被使能，模数转换（由该请求源请求的）完毕且 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>挂起位清零</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 复位寄存器 CRPR1 中各位
<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

**模数转换器**

符号	位	类型	描述
<b>Rsv</b>	<b>7</b>	r	<b>保留</b> 读操作返回 <b>1</b> ；应写入 <b>0</b> 。 <i>注： 复位后该位被立即初始化为 0，但随后很快被硬件更新为 1（且保持为 1）。</i>
<b>0</b>	<b>1</b>	r	<b>保留</b> 读操作返回 <b>0</b> ；应写入 <b>0</b> 。

### 17.7.8 结果寄存器

结果寄存器中存放转换结果，也可存放引起结果寄存器最新更新的通道编号。结果寄存器可从不同的地址上以不同的方式读取。根据所选择的读取地址，可从结果寄存器中读取以下位域。有关转换结果的数据排列和宽度的详细描述，参见[章节 17.4.7.4](#)。

#### 正常读取 RESRx

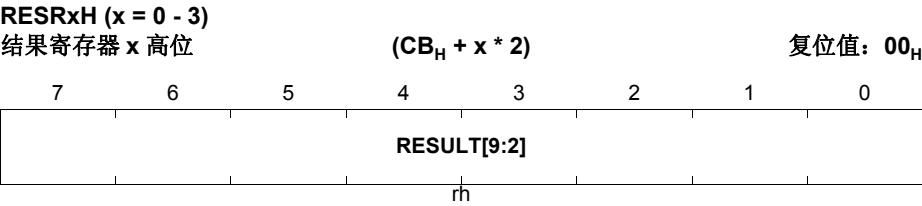
该方式给出 8 位或 10 位的转换结果以及 3 位的通道编号。假如位 RCRx.VFCTR 被置位，读命令访问该寄存器的高位字节时，相应的有效标志被清零。

**RESRxL (x = 0 - 3)**  
**结果寄存器 x 低位** **(CA<sub>H</sub> + x \* 2)** **复位值: 00<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>RESULT[1:0]</b>	<b>0</b>	<b>VF</b>	<b>DRC</b>			<b>CHNR</b>	
rh	r	rh	rh			rh	

符号	位	类型	描述
<b>CHNR</b>	[2:0]	rh	<b>通道编号</b> 该位域包含引起结果寄存器最新更新的通道编号。
<b>DRC</b>	3	rh	<b>数据压缩计数器</b> 该位指示转换结果还需要累加几次，才能产生最终的数据压缩结果。 0 <sub>B</sub> 最终转换结果已产生，可从结果寄存器中取出。该位域为 0 时有效标志自动置位。 1 <sub>B</sub> 必须再累加一次转换结果才可从结果寄存器中读取最终转换值。当该位域为 1 时有效标志自动复位。
<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 <sub>B</sub> 结果寄存器 x 中不含有效数据 1 <sub>B</sub> 结果寄存器 x 中含有效数据
<b>RESULT[1:0]</b>	[7:6]	rh	<b>转换结果</b> 该位域存放转换结果或数据压缩滤波的结果。
<b>0</b>	5	r	<b>保留</b> 读操作返回 0；应写入 0。

模数转换器



符号	位	类型	描述
RESULT[9:2]	[7:0]	rh	转换结果 该位域存放转换结果或数据压缩滤波的结果。

**累加读取 RESRax**

该方式给出累加的 9 位或 11 位的转换结果以及 3 位的通道编号。假如位 RCRx.VFCTR 被置位，读命令访问该寄存器的高位字节时，相应的有效标志被清零。

**RESRaxL (x = 0 - 3)**
**结果寄存器 x, 累加读取低位**
**(CA<sub>H</sub> + x \* 2)**
**复位值: 00<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>RESULT[2:0]</b>			<b>VF</b>	<b>DRC</b>	<b>CHNR</b>		
rh			rh	rh	rh		

符号	位	类型	描述
<b>CHNR</b>	[2:0]	rh	<b>通道编号</b> 该位域包含引起结果寄存器最新更新的通道编号。
<b>DRC</b>	3	rh	<b>数据压缩计数器</b> 该位指示转换结果还需要累加几次，才能产生最终的数据压缩结果。 0 <sub>B</sub> 最终转换结果已产生，可从结果寄存器中取出。该位域为 0 时有效标志自动置位。 1 <sub>B</sub> 必须再累加一次转换结果才可从结果寄存器中读取最终转换值。当该位域为 1 时有效标志自动复位。
<b>VF</b>	4	rh	<b>结果寄存器 x 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 中不含有效数据 1 <sub>B</sub> 结果寄存器 x 中含有效数据
<b>RESULT[2:0]</b>	[7:5]	rh	<b>转换结果</b> 该位域存放转换结果或数据压缩滤波的结果。

**RESRaxH (x = 0 - 3)**
**结果寄存器 x, 累加读取高位**
**(CB<sub>H</sub> + x \* 2)**
**复位值: 00<sub>H</sub>**

7	6	5	4	3	2	1	0
<b>RESULT[10:3]</b>							
rh							



## 模数转换器

符号	位	类型	描述
RESULT[10:3]	[7:0]	rh	转换结果 该位域存放转换结果或数据压缩滤波的结构。

## 模数转换器

对寄存器 VFCR 中的某位写 1 时，将对寄存器 RESRx/RESRAx 中的相应有效标志清零。如果硬件事件触发置位 VFx 操作，且 VFCx = 1，VFx 被清零（软件占优）。

### VFCR

有效标志清零寄存器

(CE<sub>H</sub>)

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0				VFC3	VFC2	VFC1	VFC0
r				w	w	w	w

符号	位	类型	描述
VFCx(x = 0 - 3)	x	w	结果寄存器 x 的有效标志清零 0 <sub>B</sub> 无操作 1 <sub>B</sub> 复位 VFCx
0	[7:4]	r	保留 读操作返回 0；应写入 0。

结果控制寄存器 RCRx 中包含控制结果寄存器行为的控制位以及状态监控位。

### 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>

7	6	5	4	3	2	1	0
VFCTR	WFR	0	IEN		0		DRCTR
rw	rw	r	rw		r		rw

符号	位	类型	描述
DRCTR	0	rw	数据压缩控制 该位决定几个转换结果累加用于数据压缩。它决定了位 DRC 的重载值。 0 <sub>B</sub> 禁止数据压缩滤波。DRC 的重载值为 0，故每个转换结果即为累加结果 1 <sub>B</sub> 使能数据压缩滤波。DRC 的重载值为 1，故 2 个转换结果累加作为最终结果

## 模数转换器

符号	位	类型	描述
<b>IEN</b>	4	rw	<b>中断使能</b> 该位使能和结果寄存器 x 相关的事件中断。DRC 被置 0 时（减 1 或重载后）产生事件中断。 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>待读模式</b> 该位使能结果寄存器 x 的待读模式 0 <sub>B</sub> 禁止待读模式 1 <sub>B</sub> 使能待读模式
<b>VFCTR</b>	7	rw	<b>有效标志控制</b> 该位使能对结果寄存器 x 的有效标志复位（通过读取高位字节实现） 0 <sub>B</sub> 读取 RESR <sub>x</sub> H/RESR <sub>x</sub> H 时不改变 VF（缺省值） 1 <sub>B</sub> 读取 RESR <sub>x</sub> H/RESR <sub>x</sub> H 时复位 VF
<b>0</b>	[3:1], 5	r	<b>保留</b> 读操作返回 0；应写入 0。

## 17.7.9 中断寄存器

寄存器 CHINFR 监控被激活的通道中断标志。

### CHINFR

通道中断标志寄存器

(CA<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
CHINF7	CHINF6	CHINF5	CHINF4	CHINF3	CHINF2	CHINF1	CHINF0
rh	rh	rh	rh	rh	rh	rh	rh

符号	位	类型	描述
CHINF <sub>x</sub> (x = 0 - 7)	x	rh	通道 x 中断标志 通道 x 的中断标志 0 <sub>B</sub> 通道 x 未产生通道中断 1 <sub>B</sub> 通道 x 已产生通道中断

对寄存器 CHINCR 中的某位写 1 时，将对寄存器 CHINFR 中的相应通道中断标志清零。如果一个硬件事件触发置位 CHINF<sub>x</sub>，同时 CHINC<sub>x</sub> = 1，CHINF<sub>x</sub> 被清零（软件占优）。

### CHINCR

通道中断清零寄存器

(CB<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
CHINC7	CHINC6	CHINC5	CHINC4	CHINC3	CHINC2	CHINC1	CHINC0
w	w	w	w	w	w	w	w

符号	位	类型	描述
CHINC <sub>x</sub> (x = 0 - 7)	x	w	通道 x 中断标志清零 0 <sub>B</sub> 无操作 1 <sub>B</sub> 复位 CHINF <sub>x</sub>

## 模数转换器

对寄存器 **CHINSR** 中的某位写 1 时，置位寄存器 **CHINFR** 中的相应通道中断标志，并产生中断脉冲。

### **CHINSR**

通道中断置位寄存器

(CC<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>CHINS7</b>	<b>CHINS6</b>	<b>CHINS5</b>	<b>CHINS4</b>	<b>CHINS3</b>	<b>CHINS2</b>	<b>CHINS1</b>	<b>CHINS0</b>
W	W	W	W	W	W	W	W

符号	位	类型	描述
<b>CHINSx</b> (x = 0 - 7)	x	W	通道 x 中断标志置位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位 <b>CHINFR.x</b> 并产生中断脉冲

寄存器 **CHINPR** 定义发生通道中断时，被激活的中断服务请求输出线 **SRx** (x = 0 或 1)。

### **CHINPR**

通道中断节点指针寄存器

(CD<sub>H</sub>)

复位值: 00<sub>H</sub>

7	6	5	4	3	2	1	0
<b>CHINP7</b>	<b>CHINP6</b>	<b>CHINP5</b>	<b>CHINP4</b>	<b>CHINP3</b>	<b>CHINP2</b>	<b>CHINP1</b>	<b>CHINP0</b>
rW	rW	rW	rW	rW	rW	rW	rW

符号	位	类型	描述
<b>CHINPx</b> (x = 0 - 7)	x	rW	通道 x 中断节点指针 该位决定在通道 x 产生中断时，哪条 SR 线被激活。 0 <sub>B</sub> 线 SR0 被激活 1 <sub>B</sub> 线 SR1 被激活

寄存器 EVINFR 监控被激活的事件中断标志。

## EVINFR

事件中断标志寄存器

(CE<sub>H</sub>)

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
EVINF7	EVINF6	EVINF5	EVINF4	0		EVINF1	EVINF0
rh	rh	rh	rh	r		rh	rh

符号	位	类型	描述
<b>EVINF<sub>x</sub></b> ( <b>x = 0 - 1, 4 - 7</b> )	[1:0], [7:4]	rh	<b>事件 x 的中断标志</b> 该位监控事件 x 的中断状态。 0 <sub>B</sub> 事件 x 未产生事件中断 1 <sub>B</sub> 事件 x 已产生事件中断
<b>0</b>	[3:2]	r	<b>保留</b> 读操作返回 0；应写入 0。

对寄存器 EVINCR 中的某位写 1 时，将对寄存器 EVINFR 中的相应事件中断标志清零。如果硬件事件触发置位 EVINF<sub>x</sub>，同时 EVINC<sub>x</sub> = 1，EVINF<sub>x</sub> 被清零（软件占优）。

## EVINCR

事件中断标志清零寄存器

(CF<sub>H</sub>)

复位值：00<sub>H</sub>

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
<b>EVINC<sub>x</sub></b> ( <b>x = 0 - 1, 4 - 7</b> )	[1:0], [7:4]	w	<b>事件 x 中断标志清零</b> 0 <sub>B</sub> 无操作 1 <sub>B</sub> 复位 EVINFR.x
<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

对寄存器 EVINSR 中的某位写 1 时，置位寄存器 EVINFR 中的相应通道中断标志，并产生中断脉冲（如果中断被使能）。

### EVINSR

事件中中断标志置位寄存器

(D2<sub>H</sub>)

复位值：00<sub>H</sub>

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
EVINSx (x = 0 - 1, 4 - 7)	[1:0], [7:4]	w	事件 x 中断标志置位 0 <sub>B</sub> 无操作 1 <sub>B</sub> 置位 EVINFR.x
0	[3:2]	r	保留 读操作返回 0；应写入 0。

寄存器 EVINPR 定义了发生事件中中断时被激活的中断请求输出线 SRx（x = 0 或 1）。

### EVINPR

事件中中断节点指针寄存器

(D3<sub>H</sub>)

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
EVINP7	EVINP6	EVINP5	EVINP4	0		EVINP1	EVINP0
rw	rw	rw	rw	r		rw	rw

符号	位	类型	描述
EVINPx (x = 0 - 1, 4 - 7)	[1:0], [7:4]	rw	事件 x 中断节点指针 该位决定产生事件 x 中断时，哪条 SRx 线被激活。 0 <sub>B</sub> 线 SR0 被激活 1 <sub>B</sub> 线 SR1 被激活
0	[3:2]	r	保留 读操作返回 0；应写入 0。

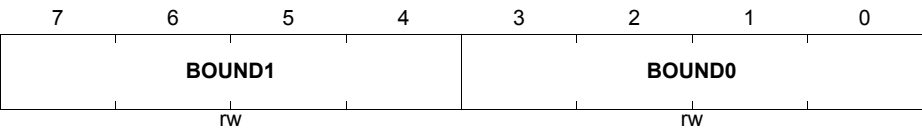
模数转换器

寄存器 **LCBR** 定义了用于进行极限检查的比较值（边界）的高四位。将四个 "0"（8 位转换）或六个 "0"（10 位转换）串联在位域 **BOUND0** 和 **BOUND1** 定义的数据之后构成比较值，和转换结果进行比较。例如：**BOUND1** 的复位值（**B<sub>H</sub>**）所对应的 8 位比较值为 **B0<sub>H</sub>**，10 位比较值为 **2C0<sub>H</sub>**。

**LCBR**  
 边界检查寄存器

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

复位值 **B7<sub>H</sub>**



符号	位	类型	描述
<b>BOUNDx</b> <b>(x = 0 - 1)</b>	[3:0], [7:4]	rw	<b>极限检查的边界</b> 该位域定义了用于进行极限检查的比较值的高四位。 极限检查的结果用来产生中断。



## 18 片上调试支持

片上调试支持（OCDS）提供了基于 XC800 系统进行软件开发与调试所需的基本功能。

OCDS 设计基于以下原则：

- 利用 XC800 核内置的调试功能
- 增加最少的硬件开销
- 监控程序支持大部分操作
- 利用标准接口与主机（调试器）通信

### 18.1 特性

OCDS 支持如下调试特性：

- 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
- 由 JTAG 通过激活专用引脚的方式处理外部断点
- 程序代码单步调试

### 18.2 功能描述

OCDS 的功能框图如 图 18-1 所示。监控器模式控制（MMC）模块是 OCDS 系统的核心，连接 OCDS 的控制信号，支持总体调试功能。MMC 主要通过调试接口和 XC800 内核进行通信，并接收复位和时钟信号。

MMC 根据内核的存储器地址和控制信号，可对专用附加存储器，即监控器 ROM（存储固件代码）和监控器 RAM（存储工作数据且可用作监控器堆栈）进行恰当的访问。

通过 JTAG<sup>1)</sup> 接口访问 OCDS 系统，JTAG 接口是测试和调试专用接口，通常在应用中不使用。MBC 引脚专门用于外部配置与调试控制。

1) JTAG 端口引脚可分配给 P0 口（首选调试端口）或 P1 和 P2 口（次选调试端口）或 P5 口（第三可选调试端口）在和 OCDS 系统连接时，用户必须将 JTAG 引脚（TCK 和 TDI）设置为输入口。

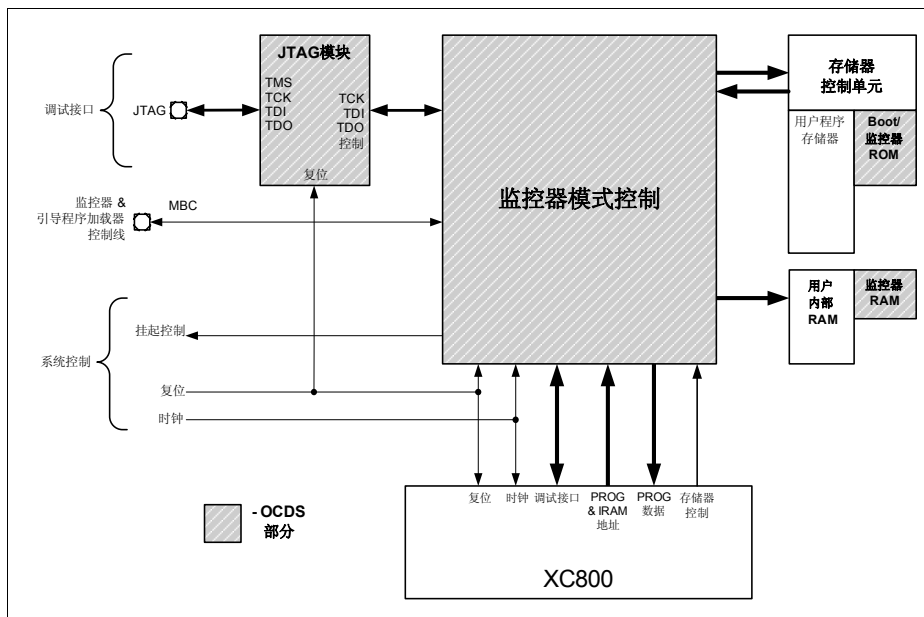


图 18-1 XC878 OCDS 框图

注：只有当XC878从OCDS模式下启动之后，这里描述的所有调试功能才可用。  
启动配置选择的具体内容，参见[章节 7.2.3](#)。

注意：只要OCDS有效，应用软件决不能修改扩展操作（EO）寄存器中的位TRAP\_EN！

### 18.3 调试

片上调试系统的功能分两部分描述。第一部分描述调试事件的产生；第二部分描述产生调试事件时相应的调试动作。

- 调试事件：
  - [硬件断点](#)
  - [软件断点](#)
  - [外部断点](#)
- 调试事件动作
  - [调用监控程序](#)
  - [激活MBC引脚](#)

XC878调试操作基于OCDS硬件和专用软件（即监控程序）的紧密配合。

### 18.3.1 调试事件

OCDS 系统可识别多种不同的调试事件，通常称其为断点。

根据断点事件处理的时间，断点可划分为以下三种类型：

- 指令执行前断点  
断点刚好发生在断点指令（即引起该断点的指令）执行之前。因此，该断点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。
- 指令执行后断点  
在断点指令（即引起该断点的指令）执行之后断点立即发生。因此在发生相应的调试动作时，该断点指令已被执行。
- 立即断点  
这类事件和 XC878 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”地执行调试动作。

#### 18.3.1.1 硬件断点

通过监控 XC878 系统的特定地址总线产生硬件断点。将与硬件断点类型相关的总线和寄存器中设定的断点地址进行连续比较。

硬件断点可分为两类：

- 根据监控的地址总线
  - **指令地址断点**  
监控程序存储器地址（PROGA）
  - **IRAM 地址断点**  
监控用于读/写操作的内部数据存储器地址（SOURCE\_A、DESTIN\_A）
- 根据比较方式
  - 点断点  
只和一个地址值比较；只在和该值匹配时产生断点事件
  - 区间断点  
与两个地址值比较；监控的地址值在两个设定的地址值之间（闭区间）时产生断点事件。

#### 指令地址断点

当断点地址和将要执行的指令的首字节地址匹配（即从程序存储器中取回的指令操作码的起始地址）时，产生这类指令指针（IP）断点。

*注：对于 2 字节和 3 字节指令，第二和第三个指令字节的地址不产生断点。*

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

XC878 OCDS 支持指令地址的点断点和区间断点（参见页 18-4）。

#### 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“指令执行后断点”，在断点指令执行之后立即执行相应调试动作。

XC878 OCDS 只支持 IRAM 地址的区间断点。

OCDS 能够区分对 IRAM 读操作产生断点和写操作产生断点。

### 硬件断点设置

OCDS 最多允许设置 4 个硬件断点。XC878 的程序存储器地址为 20 位宽，内部数据存储器地址（读和写操作）为 8 位宽。在指令地址上设置断点时，**BANKBPx** 定义 4 位组地址，**HWBPx** 定义选定的组内的 16 位地址；在 IRAM 地址上设置断点时，**HWBP2/3L** 和 **HWBP2/3H** 定义 8 位 IRAM 地址范围。

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 = **BANKBP0 & HWBP0** 和  
指令地址 = **BANKBP1 & HWBP1** 或
  - 一个区间断点  
**BANKBP0 & HWBP0** <= 指令地址 <= **BANKBP1 & HWBP1**
- 断点 2
  - 一个点断点，指令地址 = **BANKBP2 & 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 = **BANKBP3 & 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。

#### 18.3.1.2 软件断点

这类断点使用 XC800 专用的（非 8051 标准）TRAP 指令，扩展操作（EO）寄存器中 **TRAP\_EN** 置 1 的同时，该指令由 CPU 译码。

当读取 TRAP 指令时，产生一个“指令执行前断点”，执行相应的断点动作。

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。但只能在可写存储器（RAM/Flash）地址段内设置断点。

*注：为了在调试事件之后继续执行用户程序，外部调试器必须恢复当前软件断点所在地址的原代码。*

#### 18.3.1.3 外部断点

这类调试事件属于“立即断点”，可通过下面两种方式产生：

- 通过 JTAG 接口请求 - 利用一个特定序列，连接到 JTAG 接口上的外部设备可以中断 XC878 上正在运行的用户程序，启动调试进程。
- 在 XC878 运行过程中，将专用监控器和启动程序加载器控制线（MBC）拉低 - 用于响应外部的异步事件。

### 18.3.1.4 NMI 模式的优先级高于调试模式

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：

1. 内核在服务 NMI 时，不响应外部断点。  
若在执行 NMI 中断服务程序的过程中产生外部断点请求，只有在执行 RETI 指令之后该请求才能被接受。
2. 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，中断服务程序将作为代码执行，只有在执行 RETI 之后才能再次激活监控模式。

CPU 处于 NMI 模式时，硬件断点和软件断点照常处理。

## 18.3.2 调试动作

系统可根据当前配置以两种方式来响应调试事件。

### 18.3.2.1 调用监控程序

XC878 带有片上监控程序，产品出厂时保存在非易失监控器 ROM 中（见图 18-1）。识别到调试事件时，激活该监控程序是 OCDS 的主要和基本响应。

OCDS 硬件确保监控器始终安全启动，并在调试过程中完全独立于当前的系统状态。此外，监控器工作时产生的中断请求将不会干扰固件的正常执行。

一旦监控器开始工作，它将运行在自己的堆栈和数据存储器中（见图 18-1 中的监控器 RAM），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，XC878 中的 OCDS 调试完全是非破坏性的。

XC878 监控器包括以下功能：

- 通过 JTAG 接口与外部调试器通信
- 读 / 写访问任意存储器地址和包括指令指针和密码保护位在内的特殊功能寄存器（SFR）
- 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以及 JTAG 通信协议的详细描述。

### 18.3.2.2 激活 MBC 引脚

OCDS 可通过将 MBC 引脚拉低来响应调试事件。

该功能有两种可选择的设置：

- 启动监控程序时的附加动作 - 在这种情况下，MBC 引脚有效最多可持续 77 个系统时钟（SCLK）周期
- CPU 暂时挂起时 OCDS 的唯一动作 - MBC 引脚被驱动为低 4 个 SCLK 周期，这是对程序流的最快响应（断点匹配）

## 18.4 调试挂起控制

XC878 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，它还支持一个附加特性：调试期间的模块挂起特性。

只要器件处于监控器模式（即，用户软件不运行、处于中断状态）并且片上软件（监控器或者启动代码）使能调试挂起功能，OCDS 将向以下计数器模块发送一个信号，这些模块包括：

- 看门狗定时器（WDT）
- 定时器 T2
- 捕获 / 比较单元 6（CCU6）中的定时器 T12 和 T13

模块挂起控制寄存器（MODSUSP）中保存这些定时器的控制位。当某位被置位，在监控器模式有效的情况下，相应的定时器被终止。

这是一种非常有用的特性，尤其对看门狗定时器而言：当不执行用户软件时，该特性使得 XC878 不会被 WDT 复位信号无意地复位，相应地，也不会服务看门狗定时器。

将其它定时器模块挂起对于调试操作来说也是有意义的：一旦不运行应用程序，在断点操作期间，终止定时器有助于更为完全地“冻结”器件状态。

必须要注意：XC878 中的所有调试控制位（OCDS 中的全局使能和 SCU 中的各模块使能位）复位之后必须为 0，即缺省情况下，断点不会使任何模块挂起。但通常，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，监控器将被激活。因此，可以通过调试器根据需要设置挂起控制，之后才启动调试进程。

*注：调试挂起的详细内容请参见各模块中的模块挂起控制章节。*

## 18.5 寄存器描述

从编程人员的角度看，C868-1R，C868-1S 中的 OCDS 由 10 个寄存器地址表示（见表 18-1），这些寄存器均位于映射 SFR 区。

表 18-1 OCDS 直接寻址寄存器

寄存器缩写名	地址 (映射)	寄存器全名
MMCR	F1 <sub>H</sub>	监控器模式控制寄存器
MMCR2	E9 <sub>H</sub>	监控器模式控制寄存器 2
MMSR	F2 <sub>H</sub>	监控器模式状态寄存器
MMBPCR	F3 <sub>H</sub>	监控器模式断点控制寄存器
MMICR	F4 <sub>H</sub>	监控器模式中断控制寄存器
MMDR	F5 <sub>H</sub>	监控器模式数据寄存器
HWBPSR	F6 <sub>H</sub>	硬件断点选择寄存器
HWBPDR	F7 <sub>H</sub>	硬件断点数据寄存器
MEXTCR	EA <sub>H</sub>	存储器扩展控制寄存器
MMWR1	EB <sub>H</sub>	监控器工作寄存器 1
MMWR2	EC <sub>H</sub>	监控器工作寄存器 2

此外有 12 个可间接访问的 OCDS 寄存器，分为两组：

- 8 个硬件断点寄存器，可通过 HWBPSR（寄存器选择）和 HWBPDR（数据）访问这些寄存器。

表 18-2 硬件断点寄存器（8/16 位地址）

寄存器缩写名	寄存器全名
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

- 4 个硬件断点组寄存器，可通过 HWBPSR（寄存器选择）和 MEXTCR（数据）访问这些寄存器。

表 18-3      硬件断点组寄存器

寄存器缩写名	寄存器全名
BANKBP0	硬件断点 0 组寄存器
BANKBP1	硬件断点 1 组寄存器
BANKBP2	硬件断点 2 组寄存器
BANKBP3	硬件断点 3 组寄存器

OCDS 寄存器是片上监控程序的专用寄存器，用户不应对这些寄存器进行写操作。这些寄存器中的大部分或某些位 / 位域被保护，用户软件无法写入，只能在 C868-1R，C868-1S 的两种模式下由固件来修改。

- 启动模式 - 复位后执行启动代码，还未开始执行用户代码
- 监控器模式 - 正在运行监控程序，用户代码被中断

因此，若用户程序无意访问了 OCDS 寄存器，将不会干扰正常的调试功能。

18.5.1      监控器工作寄存器 2

如果 XC878 不从 OCDS 模式下启动，且没有外部器件和 JTAG 接口连接，即不能进行调试操作时，只有寄存器 **MMWR2** 可被用作通用寄存器。

**MMWR2**

监控器工作寄存器 2							映射 SFR (EC <sub>H</sub> )	复位值: 00 <sub>H</sub>
7	6	5	4	3	2	1	0	
MMWR2								
rw								

符号	位	类型	描述
MMWR2	7:0	rw	工作寄存器 2 监控程序的工作地址 2

18.5.2      输入选择寄存器

位 MODPISEL.JTAGTCKS 用来从两个 TCK 输入中选择一个；位 MODPISEL.JATGTDIS 用来从两个 TDI 输入中选择一个。



# MODPISEL

外设输入选择寄存器

复位值：00<sub>H</sub>

7	6	5	4	3	2	1	0
0	URRISH	JTAGTDIS	JTAGTCKS	EXINT2IS	EXINT1IS	EXINT0IS	URRIS
r	rw	rw	rw	rw	rw	rw	rw

符号	位	类型	描述
JTAGTCKS	4	rw	<b>JTAG TCK 输入选择</b> 0 选择 JTAG TCK 输入 TCK_0 1 选择 JTAG TCK 输入 TCK_1
JTAGTDIS	5	rw	<b>JTAG TDI 输入选择</b> 0 选择 JTAG TDI 输入 TDI_0 1 选择 JTAG TDI 输入 TDI_1
0	7	r	<b>保留</b> 读操作返回 0；应写入 0。

## 18.6 JTAG ID

该寄存器是 JTAG 模块内的只读寄存器，用于识别和 JTAG 接口相连的器件。当指令寄存器中的命令为 IDCODE（操作码 04H）时，该寄存器中的内容可被读出；复位后情况相同。

XC878 器件的 JTAG ID 寄存器总结见表 18-4。

**表 18-4 JTAG ID 总结**

器件类型	器件名称	JTAG ID
Flash	XC878*-13FF	1018 3083 <sub>H</sub>
	XC878*-16FF	1018 2083 <sub>H</sub>

注：星号（\*）表示表 1-1 中的器件配置信息。表 1-2 给出 XC878 系列中可用的器件详细编号。

# 19 引导程序加载器

XC878 支持引导程序加载器（BSL）模式，在硬件复位期间，可通过引脚配置进入该模式，如表 19-1 所示。BSL 模式的主要用途是通过串行接口简便、快速地编程 / 擦除 Flash 和 XRAM。XC878 支持 4 种 BSL 模式：

- UART BSL
- LIN BSL
- MultiCAN BSL
- 其它 BSL

如果器件设置为 UART/ MultiCAN，则根据最初的报文头帧，进入对应的 BSL 模式（UART 或 MultiCAN）。

注： 只有 UART 模块支持 UART BSL 模式； UART1 模块不支持该模式。

注： 对于 BSL 模式，只能使用 UART 接收 / 发送引脚和 MultiCAN 节点 0（P1.0/P1.1）的缺省设置。

表 19-1 进入 BSL 模式的引脚配置

MBC <sup>1)</sup>	TMS <sup>1)</sup>	模式 / 注释
0	0	BSL 模式（LIN 模式 <sup>2)</sup> ，UART/ MultiCAN 模式 <sup>3)4)</sup> 和其它 BSL 模式 <sup>5)</sup> ； 片上 OSC/PLL 旁路

<sup>1)</sup> 锁存的引脚值。

<sup>2)</sup> 如果器件设置为 LIN，始终使用 LIN BSL 而不是 UART/MultiCAN。

<sup>3)</sup> 对于提供 MultiCAN 模块的各型号产品，由固件根据协议解码 UART BSL 或 MultiCAN BSL。如果产品不提供 MultiCAN 和 LIN 功能，则使用 UART BSL。

<sup>4)</sup> MultiCAN BSL 模式下，固件将时钟源切换至 XTAL，片上振荡器被旁路。采用此方式避免了片上振荡器频率不变的特性，允许其它频率的时钟输入，因此确保实现精确的波特率检测（尤其对于高数据率的应用来讲）。

<sup>5)</sup> 其它 BSL 为用户定义的 BSL 模式。用户 BSL 代码位于 Flash 存储器中。如果 AltBSL 密码有效，则进入到该引导加载模式。如果密码无效，则根据产品型号进入到 LIN 或 UART/MultiCAN BSL 模式。

章节 19.1 描述 UART 和 LIN BSL 模式，章节 19.2 描述 MultiCAN BSL 模式，章节 19.3 描述其它 BSL 模式。

## 19.1 UART 和 LIN BSL 模式

UART 和 LIN BSL 模式包含 3 个功能部分，由以下 3 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。主机控制并发送一个特殊的报文头信息，用于选择一种通信模式，具体描述见 [表 19-2](#)。
- **阶段 III**：回应主机，指示传送成功 / 失败，具体描述见 [章节 19.1.1.3](#)。

**表 19-2 UART 和 LIN BSL 模式的串行通信模式**

模式	描述
<b>0</b> (00 <sub>H</sub> )	从主机向 XRAM <sup>1)</sup> 传送用户程序
<b>1</b> (01 <sub>H</sub> )	从起始地址开始执行 XRAM 中的用户程序 <sup>2)</sup>
<b>2</b> (02 <sub>H</sub> )	从主机向 P-Flash 传送用户程序 <sup>1)</sup>
<b>3</b> (03 <sub>H</sub> )	执行 P-Flash 中的用户程序 <sup>2)</sup>
<b>6</b> (06 <sub>H</sub> )	Flash 保护模式使能 / 禁止方案 <sup>1)</sup>
<b>8</b> (08 <sub>H</sub> )	从主机向 XRAM 传送用户程序 <sup>1)3)</sup>
<b>9</b> (09 <sub>H</sub> )	执行 XRAM 中的用户程序 <sup>2)3)</sup>
<b>10</b> (0A <sub>H</sub> )	获取 4 字节芯片信息
<b>15</b> (0F <sub>H</sub> )	进入 OCDS UART 模式 <sup>2)</sup>
<b>16</b> (10 <sub>H</sub> )	擦除 / 整体擦除 P-Flash 或 D-Flash 存储器 <sup>1)</sup>

<sup>1)</sup> 微控制器将返回阶段 I/II，并等待来自主机的下一条命令。

<sup>2)</sup> 退出 BSL 模式，未建立串行通信。

<sup>3)</sup> 只有 LIN BSL 模式支持模式 8 和模式 9。与模式 0 和模式 1 相似。

UART BSL 和 LIN BSL 模式下的基本串行通信协议（如传送块结构、返回主机的各种回应码）见 [章节 19.1.1](#)。通过 UART 和 LIN 协议实现的 BSL 模式的具体内容分别见 [章节 19.1.2](#) 和 [章节 19.1.3](#)。

### 19.1.1 通信协议

一旦确定了波特率，主机向微控制器发送一个信息块，用来选择期望的通信模式。UART 的块结构见**章节 19.1.1.1**；LIN 的块结构见**章节 19.1.1.2**。微控制器通过发送特定的回应码来回应主机，具体描述见**章节 19.1.1.3**。

#### 19.1.1.1 UART 传送块结构

一个 UART 传送块由 3 部分组成：

块类型 (1字节)	数据区域 (XX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域。已实现的块类型有：
    - 00<sub>H</sub>**           **“报文头”类型**  
 报文头块长度固定为 8 个字节。在报文头块的数据区域中包含用于选择不同模式的特定信息。
    - 01<sub>H</sub>**           **“数据”类型**  
 在模式 0 和模式 2 中，数据块用来传送程序代码。程序代码放在数据块的数据区域中。<sup>1)</sup>
    - 02<sub>H</sub>**           **“传送结束”（EOT）类型**  
 在模式 0 和模式 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 块的数据区域中。<sup>1)</sup>
  - 数据区域：**报文头块中的数据长度为 6 字节；数据块和 EOT 块中的数据长度不能超过 96 字节。
  - 校验和：**块类型和数据区域的 XOR（异或）校验和由主机发送。BSL 程序计算接收字节的校验和（块类型和数据区域）、并与接收到的校验和进行比较。
- <sup>1)</sup> 由报文头块中的块\_长度定义数据块和 EOT 块的长度。

### 19.1.1.2 LIN 传送块结构

一个 LIN 传送块的长度为 9 字节（固定），由 4 部分组成：

NAD (1字节)	块类型 (1字节)	数据区域 (6字节)	校验和 (1字节)
--------------	--------------	---------------	--------------

- NAD:** 用于诊断的节点地址，指定有效从节点的地址。  
**01<sub>H</sub> 至 7E<sub>H</sub>** 有效从节点地址  
**80<sub>H</sub> 至 FF<sub>H</sub>** 有效从节点地址  
**7F<sub>H</sub>** 广播地址（用于主节点向所有从节点广播）  
**00<sub>H</sub>** 无效从节点地址（为进入休眠模式命令保留）
  - 块类型:** 传送块的类型，它决定如何解读数据区域，具体描述见[章节 19.1.1.1](#)。  
**00<sub>H</sub>** “报文头”类型  
**01<sub>H</sub>** “数据”类型  
**02<sub>H</sub>** “传送结束”（EOT）类型
  - 数据区域:** 固定长度为 6 个字节。对于报文头块，1 个字节指示所选模式，5 个字节代表模式数据；对于数据块和 EOT 块，数据区域包含程序代码。
  - 校验和:** 编程校验和或 LIN 校验和是对 NAD、块类型和数据区域进行相加操作得到的带进位<sup>1)</sup>的 8 位和（不取反和取反）。
- <sup>1)</sup> 带进位的 8 位和等同于将所有值相加、每次结果大于或等于 256 时减去 255（这和模 -255 或模 -256 不同）。

LIN 诊断帧一直使用经典校验和，即只对数据字节进行校验和计算。它用来和 LIN 1.3 从节点通信。经典校验和是对所有数据字节进行相加操作得到的带进位的 8 位和再取反的结果。

通过 LIN 非标准校验和，也被称为编程校验和，来区分 XC878 编程 LIN 帧和正常 LIN 帧，该校验和使得 LIN 总线上的其它从节点（非编程）可忽略该编程帧。XC878 既支持 LIN 经典校验和，也支持编程校验和。编程校验和是对所有 8 个数据字节进行相加操作得到的带进位的 8 位和。

表 19-3 以数据 4A<sub>H</sub>、55<sub>H</sub>、93<sub>H</sub> 和 E5<sub>H</sub> 为例，给出如何进行编程校验和计算。

表 19-3      LIN 帧 - 编程校验和

数据和	HEX	结果	进位	带进位的和
4A <sub>H</sub>	4A <sub>H</sub>	4A <sub>H</sub>	0	4A <sub>H</sub>
(4A <sub>H</sub> ) + 55 <sub>H</sub>	9F <sub>H</sub>	9F <sub>H</sub>	0	9F <sub>H</sub>
(9F <sub>H</sub> ) + 93 <sub>H</sub>	0132 <sub>H</sub>	32 <sub>H</sub>	1	33 <sub>H</sub>
(33 <sub>H</sub> ) + E5 <sub>H</sub>	0118 <sub>H</sub>	18 <sub>H</sub>	1	19 <sub>H</sub>

编程校验和为 19<sub>H</sub>。编程校验和取反产生标准 LIN 校验和（经典校验和（即 E6<sub>H</sub>））。支持编程校验和与 LIN 校验和，在相应的模式中给出。

### 19.1.1.3 返回主机的回应码

微控制器需发送回应码，通知主机是否已成功接收块信息。

**表 19-4** 列出在每种通信模式下，微控制器接收到报文头块、数据块或 EOT 块之后可能返回的各种回应码。

**表 19-4 不同块类型的回应码**

模式	报文头块	数据块	EOT 块
0, 8	应答、块类型错误、校验和错误、保护错误	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, 9	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、保护错误	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	-	-
6	应答、块类型错误、校验和错误、保护错误	-	-
10	应答、块类型错误、校验和错误	-	-
15	应答、块类型错误、校验和错误	-	-
16	应答、块类型错误、校验和错误、保护错误	-	-

如果正确接收到一个信息块，发送应答码（55<sub>H</sub>）。如果失败，可能是块类型错误、或校验和错误。块类型错误由两种情况引起：（i）微控制器接收的块类型和已实现的块类型不同；（ii）微控制器接收到的传送块是以错误的序列传送的。在这两种错误情况下，BSL 程序将等待主机再次发送正确的信息块。

若 Flash 保护模式 0 或模式 1 被使能，Flash 编程和擦除操作受到限制，微控制器将向主机发出保护错误码，说明 Flash 被保护，因而 Flash 不能被编程或擦除。在这种错误情况下，BSL 程序再次等待主机发送下一个报文头块。

**表 19-5** 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 19-5 回应码的定义**

回应码	值	说明			
		块类型	BSL 模式	原因 / 含义	更正操作
应答	55 <sub>H</sub>	报文头	1, 3, 9, 15	发送回应码之后立刻执行所请求操作	-
			6, 16	所请求操作已成功执行完毕	
		EOT	0, 2, 8		
		所有其它类型		成功接收块信息。在模式 10 随后将发送 4 字节数据。准备好接收下一个信息块。	
块类型错误	FF <sub>H</sub>	报文头	2	P-Flash 起始地址为奇数。	P-Flash 地址应当为偶数。
				超过了块长度	将块长度减少到规定的限度内。
		数据, EOT		P-Flash 代码长度为奇数。	P-Flash 代码长度应当为偶数。
		所有其它		块类型未定义或通信结构无效	重新发送有效的块信息
校验和错误	FE <sub>H</sub>	所有所有类型		计算的校验和与接收的校验和不匹配	重新发送块信息
保护错误	FD <sub>H</sub>	报文头	0, 2, 8, 16	使能存储器外部访问保护, 即 FPASSWD 有效	-
			6	禁止保护的操作失败, 即 FPASSWD 与用户密码不匹配	-

### 19.1.2 通过 UART 实现引导程序加载器

一旦进入 **UART BSL**，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：等待来自主机的测试字节（80<sub>H</sub>）
- 步骤 3：和主机波特率同步
- 步骤 4：向主机发送应答字节（55<sub>H</sub>）
- 步骤 5：进入阶段 II

一旦进入 **UART BSL**，就要建立波特率。在下次硬件复位之前，主机和微控制器之间将按该波特率进行通信。

微控制器的串行端口设置为模式 1（8 位 **UART**，波特率可变），定时器 T2 配置为自动重载模式（16 位定时器）进行波特率测量。PC 主机发送测试字节（80<sub>H</sub>）启动同步过程。定时器接收到起始位（0）后启动，接收到测试字节的最后一位（1）后停止。然后 **UART BSL** 程序计算实际的波特率、设置 **PRE** 和 **BG** 的值，激活波特率发生器。同步过程完成之后，微控制器向主机发送应答字节（55<sub>H</sub>）。所支持的波特率范围从 4800 波特到 38400 波特。

如果同步失败，主机不会正确接收到微控制器发送的应答码。在这种情况下，在主机一方，主机软件会向用户显示信息，请求用户重复同步过程。回应码的具体描述见[章节 19.1.1.3](#)。

在微控制器一方，**UART BSL** 程序不能确定同步是否正确。它始终会在发送应答字节之后进入到阶段 II。因此，如果同步失败，必须复位微控制器，重新启动微控制器再次进行同步。

19.1.2.1 通信结构

报文头块、数据块、EOT 块和回应码有两种传送流，见图 19-1。模式 0 和 2 采用一种传送流，其它模式采用另一种传送流。只在模式 0 和模式 2 下传送数据块和 EOT 块。

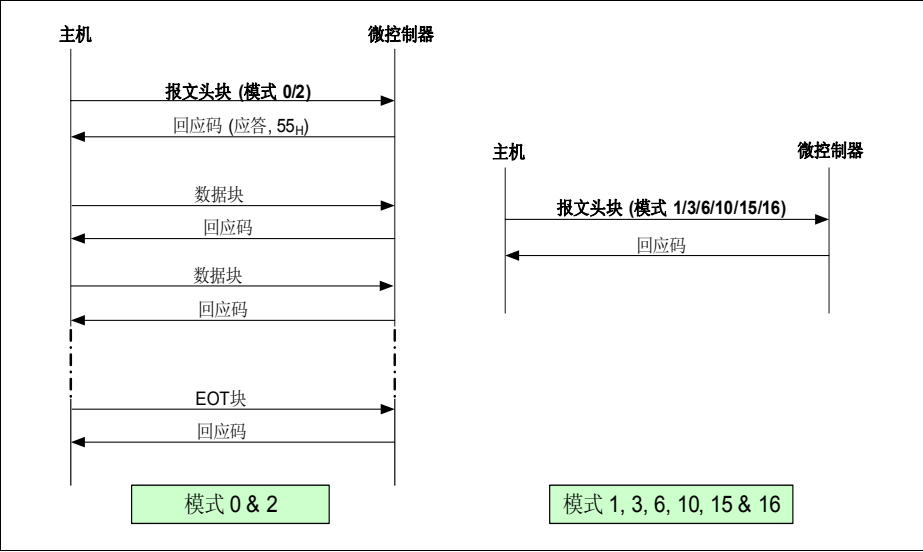


图 19-1 UART BSL 模式的通信结构

19.1.2.2 模式选择

当 UART BSL 程序进入阶段 II，首先等待一个来自主机、包含模式选择信息的 8 字节的报文头块，其结构如下：

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

描述：

- **00<sub>H</sub>：**块类型，标明该块为**报文头块**
- **模式：**选择的模式，支持模式 0-6，见表 19-2。
- **模式数据：**5 字节特殊信息，以激活相应的模式
- **校验和：**报文头块的校验和，所有 7 字节的 XOR（异或）

19.1.2.3 激活模式 0 和模式 2

模式 0 和模式 2<sup>1)</sup> 用于从主机分别向微控制器的 XRAM 和 Flash 传送用户程序。报文头块的结构如下：

报文头块

00 <sub>H</sub> (报文头块)	00 <sub>H</sub> /02 <sub>H</sub> (模式 0/2)	模式数据				校验和
		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)	

模式数据描述：

**起始地址高位，低位：**20 位起始地址中的 16 位地址<sup>2)</sup>，决定将接收到的程序复制到 XRAM/Flash<sup>3)</sup> 中的什么位置。

**块\_长度：**数据或 EOT 块的总长度（块类型、数据区域和校验和）<sup>4)5)</sup>

**未使用：**2 个字节，这些字节未使用，在模式 0/2 中被忽略。

在成功接收到报文头块之后，微控制器进入模式 0/2，在该阶段，主机通过数据块和 EOT 块向微控制器传送程序代码，数据块和 EOT 块的结构如下。

数据块

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

描述：

**程序代码：**程序代码长度为 (块\_长度 - 2) 个字节，块\_长度由前面的报文头决定。

*注：* 不允许空数据块。

1) 当模式 2 被激活时，可改变 XRAM 的内容。  
2) 应当根据 Flash 产品型号动态的分配高 4 位或组。  
3) 在每个编程周期内，不应当跨越字节边界。P-Flash 编程时，起始地址必须为有效的偶数地址。  
4) 在报文头块中定义块\_长度之后，随后的数据或 EOT 块必须和该长度一致。要重新定义块\_长度，必须在新  
的报文头块中完成。  
5) 模式 0 的块\_长度为 3 - 128 字节。模式 2 中，编程 D-Flash 时，块\_长度为 3 - 34 字节，编程 P-Flash 时，  
块\_长度始终为 4 - 66 字节。

## EOT 块

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

描述:

**最后\_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\_代码长度)。这些字节未使用, 可设置为任意值。

### 19.1.2.4 激活模式 1、3 和 15

模式 1 用于执行 XRAM 中起始地址为 0F'000<sub>H</sub> 和 2F'000<sub>H</sub> (取决于产品的 Flash 类型) 的用户程序。模式 3 用于执行 Flash 中起始地址为 0'0000<sub>H</sub> 的用户程序。模式 15 用于使系统进入 OCDS UART 模式。报文头块结构如下:

#### 报文头块

00 <sub>H</sub> (报文头块)	01 <sub>H</sub> /03 <sub>H</sub> /0F <sub>H</sub> (模式 1/3/15)	模式数据	校验和 (1字节)
		未使用 (5字节)	

模式数据描述:

**未使用:** 未使用这 5 个字节, 在模式 1/3/15 下将被忽略。

对于模式 1、3 和 15, 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 BSL 模式, 跳转至 XRAM 地址 (模式 1)、Flash 的地址单元 0'0000<sub>H</sub> (模式 3)、和 / 或开始与 OCDS UART 调试器进行通信 (模式 15)。

### 19.1.2.5 激活模式 6

模式 6<sup>1)</sup> 通过给定的用户密码来使能或禁止 Flash 保护。该模式的报文头块结构如下：

#### 报文头块

00 <sub>H</sub> (报文头块)	06 <sub>H</sub> (模式 6)	模式数据(5 字节)		校验和
		用户密码 (2 字节)	未使用 (3 字节)	

模式数据描述：

**用户密码：**该字节由用户设置，用来使能或禁止 Flash 保护模式。用户密码的具体描述请参见 [章节 3.3.1](#)。传送 16 位用户密码时遵循网络字节顺序，这意味着先传送高位字节。

**未使用：**未使用这四个字节、在模式 6 中忽略不用。

在模式 6 中，报文头块是唯一由主机发出的传送块。当用户想要 (i) 使能 Flash 保护；(ii) 禁止 Flash 保护时，使用该模式。

当 Flash 还未被保护时，微控制器将根据用户密码的 3 MSB 使能各种 Flash 保护机制。一旦接收到报文头块，且微控制器确定该用户密码为程序密码，并用于以后的操作中，则所选择的 Flash 保护模式将被激活。

当 Flash 已经被保护时，如果用户密码字节和程序密码匹配，微控制器将使所有的 Flash 保护模式失效。**被保护的 P-Flash 模块将被擦除，并根据用户密码的位 12 决定是否擦除 D-Flash 模块。**上述操作完成后，程序密码被复位。对于 [章节 3.3.1](#) 中描述的 Flash 硬件保护的情况，下一次上电或硬件复位后，Flash 保护模式将失效。

### 19.1.2.6 激活模式 10

模式 10 (0A<sub>H</sub>) 用来获取 4 字节数据。4 字节数据的内容由报文头块中的选择字节决定。该模式的报文头块结构如下：

#### 报文头块

00 <sub>H</sub> (报文头块)	0A <sub>H</sub> (模式 10)	模式数据 (5 字节)		校验和
		未使用 (4 字节)	选择 (1 字节)	

模式数据描述：

**选择** 该字节将决定发送给主机的 4 字节数据。只有选择字节 =00<sub>H</sub> 用于返回芯片 ID，用于识别特定的衍生产品型号。

1) 当模式 6 被激活时，可改变 XRAM 中的内容。

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

在模式 10 中，报文头块是唯一由主机发出的传送块。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 字节的芯片信息。如果接收到无效选择字节，微控制器将返回 4 字节的 00<sub>H</sub>。

### 19.1.2.7 激活模式 16

模式模式 16（10<sub>H</sub>）用于擦除 P-Flash 或 D-Flash 中的一页，或整体擦除所有 P-Flash 或 D-Flash 存储器模块。如果代码保护被使能，则该模式不可用并返回保护错误。通过模式数据域中的擦除 / 整体擦除字节来选择擦除操作的类型。

#### 报文头块

00 <sub>H</sub> (报文头块)	10 <sub>H</sub> (模式 16)	模式数据 (5 字节)				校验和
		起始地址, 高位 (1 字节)	起始地址, 低位 (1 字节)	擦除/ 整体擦除 (1 字节)	未使用 (2 字节)	

模式数据描述:

**起始地址高位, 低位**擦除操作目标页或模块中的任意 20 位起始地址<sup>1)</sup>的低 16 位。

**擦除 / 整体擦除:** 0（擦除一页）或 1（整体擦除）。

**未使用:** 2 字节，这些字节未被使用，在模式 16 中将被忽略。

1) 应当根据 Flash 产品型号动态的分配高 4 位或组。

### 19.1.3 通过 LIN 实现引导程序加载器

标准 LIN 协议可以支持的最大波特率为 20 kHz。不过，XC878 器件的增强特性可支持高达 57.6 kHz 的波特率。使用标准 LIN 协议所实现的 LIN BSL 支持 20 kHz 及 20 kHz 以下的波特率；使用 UART 协议（单线 UART）所实现的快速 LIN BSL 支持 20 kHz 至 57.6 kHz 之间的波特率，具体内容请参见[章节 19.1.3.9](#)。

通过设置模式 0、模式 2 或模式 8 下的快速编程选项，LIN BSL 支持快速编程。具体内容请参见[章节 19.1.3.3](#)。

LIN BSL 特性如下：

- 根据接收到的每个 LIN 帧，与通信联系方的传送速度（波特率）重新同步
- 使用诊断帧（主机请求和从节点回应）
- 用户将 NAD 预设在 P-Flash 模块的最高地址中（如果该值未设定或无效，则使用缺省的广播 NAD）
- 如果接收到的第一帧是无效 LIN 帧，将其保存到 XRAM 中、跳转到用户模式
- 支持编程校验和、LIN 校验和
- 使用单线 UART（LIN）上的 BSL 模式协议可实现快速 LIN BSL

在进入阶段 II 和 III 之前，始终先执行重新同步和波特率的建立（阶段 I），从而可支持不同的波特率。当接收到主机请求报文头时，进入阶段 II，否则进入阶段 III（从机回应报文头）。主机请求报文头的保护 ID 为  $3C_H$ ，从机回应报文头的保护 ID 为  $7D_H$ 。只有在接收到从机回应报文头之后，微控制器才会回应主机。命令和回应 LIN 帧被识别为诊断 LIN 帧，具有标准的 8 字节数据结构（非 2 字节或 4 字节数据结构）。

一旦进入 LIN BSL，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）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：等待一个来自主机的 LIN 帧
- 步骤 3：和主机波特率同步
- 步骤 4：进入阶段 II（主机请求帧）或
- 阶段 III（从机回应帧）

**注：** 对于每一个主机请求报文头和从机回应报文头 LIN 帧，都需要完成重新同步和波特率建立的过程。

一个报文头 LIN 帧由以下各部分组成：

- 同步（SYN）分隔（13 个位时间，低电平）
- 同步（SYN）字节（ $55_H$ ）
- 保护 ID 域（ $3C_H$  或  $7D_H$ ）

同步分隔域用于指示新帧的开始，必须至少保持 13 位的显性值。在分隔域开始时，若检测到引脚 T2EX 上的负跳变，定时器 T2 外部启动使能位（T2MOD.T2RHEN）被置位。从而当引脚 T2EX 上再次出现负跳变时，定时器 T2 自动启动。最后，查询同步字节结束标志（FDCON.EOFSYN）。该标志被置位时，定时器 T2 被停止。传送时间（8 位）被捕获到 T2 重载 / 捕获寄存器（RC2H/L）中。随后 LIN BSL 程序计算实际的波特率，设置 PRE 和 BG 值、激活波特率发生器。LIN 波特率检测如[图 19-2](#)所示。



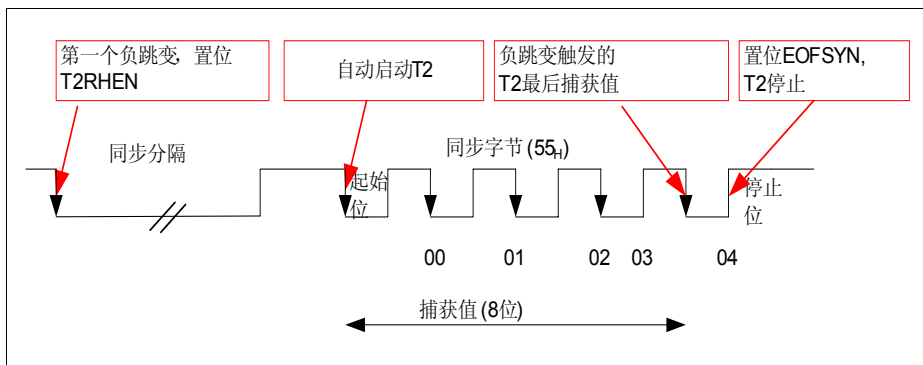


图 19-2 报文头 LIN 的自动波特率检测

### 19.1.3.1 通信结构

PC 主机和微控制器之间 3 个阶段的传送操作如图 19-3 所示，图 19-4 给出主机请求报文头、从机回应报文头、命令和回应 LIN 帧的结构。

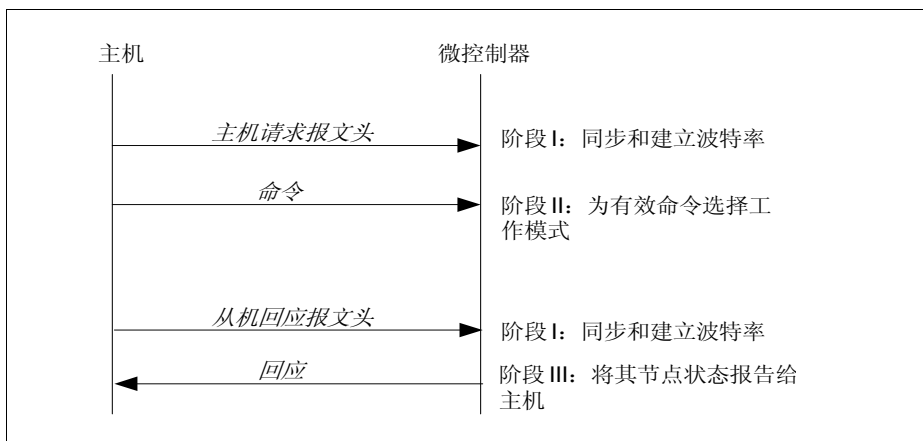


图 19-3 LIN BSL - 阶段 I, II 和 III

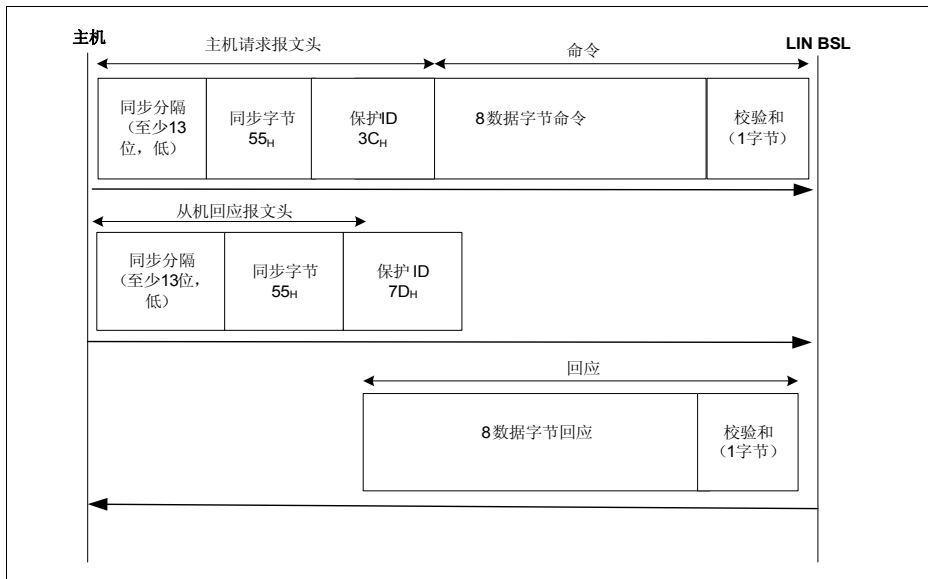


图 19-4 LIN BSL 帧

### 19.1.3.2 模式选择

当 LIN BSL 程序进入阶段 II，首先等待一个来自主机、包含模式选择信息的 9 字节报文头块，结构如下：

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

描述：

- NAD:** 用于诊断的节点地址
- 00<sub>H</sub>:** 块类型，标明该块为**报文头块**
- 模式:** 选择工作模式。支持模式 0、1、2、3、6、8、9、10、15 和 16，见表 19-2
- 模式数据:** 5 字节特殊信息，用于激活相应的模式
- 校验和:** 报文头块的编程校验和或 LIN 校验和

注：模式 8 和 9 支持 LIN 校验和，其它模式支持编程校验和。

19.1.3.3 激活模式 0, 2 和 8

模式 0、模式 8 和模式 2<sup>1)</sup> 分别用于从主机向微控制器的 XRAM 和 Flash 传送用户程序。报文头块的结构如下：

报文头块

NAD (1字节)	00 <sub>H</sub> (报文头 块)	00 <sub>H</sub> /02 <sub>H</sub> /08 <sub>H</sub> (模式 0/2/8)	模式数据					校验和
			起始地 址，高位 (1 字节)	起始地 址，低位 (1 字节)	数据 计数 (1字节)	未使用 (2字节)	Fast_ Prog (1字节)	

模式数据描述：

**起始地址高位，低位：**20 位起始地址<sup>2)</sup> 的的低 16 位，决定将接收到的程序代码复制到 XRAM/Flash<sup>3)</sup> 中的什么位置。

1) 当模式 2 被激活时，可改变 XRAM 的内容。  
2) 应当根据 Flash 产品型号动态的分配高 4 位或组。  
3) 在每个编程周期内，不允许跨越字线边界。P-Flash 编程时，起始地址必须为有效的偶数地址。

## 引导程序加载器

**数据计数：**数据块<sup>1)</sup>的个数。范围从 0 到 255。

**未使用：**2 个字节，这些字节未使用，在模式 0/2/8 下被忽略。

**Fast\_Prog：**指示进入快速 LIN BSL 的字节。

- 01<sub>H</sub>：进入快速 LIN BSL
- 其它值：忽略，不进入快速 LIN BSL

*注：此处未采用 UART BSL 中的块-长度，因为诊断 LIN 帧具有标准的 8 字节数据结构，随后是校验和。*

使用命令 LIN 帧（报文头块）进入快速 LIN BSL 模式时，不应当接收其它主机请求报文头和命令 LIN 帧（数据块或 EOT 块）。微控制器将接收从机回应报文头 LIN 帧并发出回应 LIN 帧以应答接收到的正确报文头块，使用 UART BSL 协议进入到快速 LIN BSL，见[章节 19.1.3.9](#)。

在成功接收到报文头块之后，微控制器进入模式 0/2/8，在该阶段，主机通过数据块和 EOT 块向微控制器传送程序代码，数据块和 EOT 块的结构如下。

### 数据块

NAD (1字节)	数据块 01 <sub>H</sub>	程序代码 (6字节)	校验和 (1 字节)
--------------	------------------------	---------------	---------------

描述：

**程序代码：**每个数据块中程序代码的固定长度为 6 字节。

*注：不允许空的数据块。*

### EOT 块

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

描述：

**最后\_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（有效数据）

**未使用** 长度为 (LIN\_块\_长度<sup>2)</sup>-4-最后\_代码长度)。这些字节未使用，可设置为任意值。

微控制器将数据块中的有效数据（6 字节）传送到一个缓存中，并对接收到的数据字节计数。一旦计数到缓存的最大容量，微控制器将这些数据编程到 Flash/XRAM 中。如果

1) 每个数据块被嵌入到主机请求帧中。

2) LIN\_块\_长度始终为 9 字节，包括 NAD 和校验和。

**引导程序加载器**

在达到缓存最大容量之前，接收到 EOT 块，则将 EOT 中存放的数据字节也编程到 Flash/XRAM 中。

**19.1.3.4 激活模式 1, 3 和 9**

模式1、模式9和模式3分别用于执行微控制器XRAM/Flash中起始地址为0'F000<sub>H</sub>/2'F000<sub>H</sub>（根据具体产品中的 Flash 容量）和 0'0000<sub>H</sub> 的用户程序。在该模式下，报文头块的结构如下：

**报文头块**

NAD (1 字节)	00 <sub>H</sub> (报文头块)	01 <sub>H</sub> /03 <sub>H</sub> /09 <sub>H</sub> (模式 1/3/9)	模式数据	校验和 (1字节)
			未使用 (5 字节)	

模式数据描述：

**未使用：**未使用这 5 个字节，在模式 1/3/9 下将被忽略。

对于模式 1、3、9，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 BSL 模式，跳转至 XRAM 的地址单元 0'F000<sub>H</sub>/2'F000<sub>H</sub>（模式 1 和模式 9）和 / 或 Flash 的地址单元 0'0000<sub>H</sub>（模式 3）。

**19.1.3.5 激活模式 6**

模式 6<sup>1)</sup> 通过给定的用户密码使能或禁止 Flash 保护。该模式的报文头块结构如下：

**报文头块**

NAD (1 字节)	00 <sub>H</sub> (报文头块)	06 <sub>H</sub> (模式 6)	模式数据 (5字节)		校验和 (1 字节)
			用户密码 (2字节)	未使用 (3 字节)	

模式数据描述参见 [章节 19.1.2.5](#)。

1) 模式 6 被激活时，可改变 XRAM 中的内容。

### 19.1.3.6 激活模式 10

模式 10 (0A<sub>H</sub>) 用于获取 4 字节数据，其内容由报文头块中的选择字节决定。该模式的报文头结构如下：

#### 报文头块

NAD (1 字节)	00 <sub>H</sub> (报文头块)	0A <sub>H</sub> (模式 10)	模式数据 (5 字节)		校验和 (1 字节)
			未使用 (4 字节)	选择 (1 字节)	

模式数据描述可参见[章节 19.1.2.6](#)。

### 19.1.3.7 激活模式 16

模式 16 (10<sub>H</sub>) 用于擦除 P-Flash 或 D-Flash 中的一页，或整体擦除所有 P-Flash 或 D-Flash 存储器模块。通过模式数据域中的擦除 / 整体擦除字节来选择擦除操作的类型。

#### 报文头块

NAD (1 字节)	00 <sub>H</sub> (报文头块)	10 <sub>H</sub> (模式 16)	模式数据 (5 字节)				校验和 (1 字节)
			起始地址， 高位 (1 字节)	起始地址， 低位 (1 字节)	擦除/ 整体擦除 (1 字节)	未使用 (2 字节)	

模式数据描述可参见[章节 19.1.2.7](#)。

### 19.1.3.8 发向主机的 LIN 回应协议

当主机发出一个从机回应报文头 LIN 帧后，微控制器发送一个回应块指示其状态。回应块的长度为 9 字节（固定），由四部分组成：

NAD (1 字节)	回应 (1 字节)	未使用 (6 字节)	校验和 (1 字节)
---------------	--------------	---------------	---------------

- **NAD:** 用于诊断的节点地址，指明有效从节点的地址
- **回应:** 应答字节或错误状态指示字节，见[章节 19.1.1.3](#)
- **未使用:** 这 6 个字节被忽略、设置为 00<sub>H</sub>
- **校验和:** LIN 校验和是对 NAD、回应、未使用字节进行相加操作得到的带进位的 8 位和。不管何种模式，所有的回应均采用 LIN 校验和。



**表 19-6**      用户定义的参数

地址	用户定义的参数	规则 / 范围	缺省值
0'BFFE <sub>H</sub> <sup>1)</sup> 0'EF <sub>H</sub> <sup>2)</sup>	NAD	01 <sub>H</sub> – 0FF <sub>H</sub> (00 <sub>H</sub> 被保留)	7F <sub>H</sub>
0'BFFF <sub>H</sub> <sup>1)</sup> 0'EFF <sub>H</sub> <sup>2)</sup>	NAD	-	-

<sup>1)</sup> 对应于具有 52 KB Flash 存储器的器件。

<sup>2)</sup> 对应于具有 64 KB Flash 存储器的器件。

**注：** 对于支持 LIN BSL 的器件，必须确保在保护器件之前设定有效的 NAD。若 NAD 错误则不能授权访问器件。



## 19.2 MultiCAN BSL 模式

只有当 Flash 未保护时才可进入 MultiCAN BSL 模式，否则将进入用户模式，执行存储器地址 0000<sub>H</sub> 处的代码。MultiCAN BSL 协议分为两部分：硬件初始化和软件通信。

在硬件初始化部分，配置 XC878 使用片外振荡器和 CAN 节点 0 进行通信。采用片外振荡器以确保 CAN 实际应用（此时要求振荡器的频率偏差小于 1.5 %）的系统性能最佳。XC878 支持 4 种振荡器频率，用户可在 P-Flash 模块的最高地址中设置选择该频率值的参数。用户定义的参数描述请参见章节 19.2.3。

在软件通信部分，分为三个主要阶段，即自动波特率检测、应答和数据接收阶段。这三个阶段包含了 CAN 报文对象<sup>1)</sup> 的发送和接收操作。

一旦进入 MultiCAN BSL 模式，首先进行自动波特率检测，主机向微控制器发送主机命令报文，微控制器由此确定当前 CAN 网络的波特率并相应设置 CAN 节点的波特率，从而使能通信通道。在应答阶段，微控制器向主机发送应答报文，建立通信通道。通信通道建立之后，进入数据接收阶段。主机发送数据报文对象，将程序代码下载到 XRAM 中，从 XRAM 中执行这些代码。XC878 中，可用于执行程序的 XRAM 为 2 KB。

为了简化 MultiCAN BSL 的实现，作如下假设：

- 主机和 XC878 是 CAN 网络中的唯一 CAN 节点（点对点连接）
- 在该模式中，使用 XC878 CAN 节点 0（P1.0/P1.1）
- XC878 期望接收报文标识符为 555<sub>H</sub> 的标准 CAN 帧。

### 19.2.1 通信协议

在该模式下，使用标准 CAN 数据帧（11 位标识符）报文对象实现数据交换，见图 19-6。XC878 忽略带有其它标识符的报文对象。用标准 CAN 报文的数据域来实现通信协议。

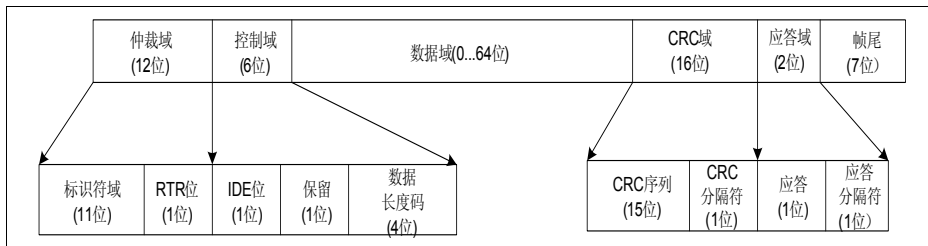


图 19-6 标准 CAN 帧格式

由主机启动通信过程，主机连续发送主机命令报文对象，直到接收到微控制器发出的应答报文对象。

波特率被确定且主机接收到应答之后，主机发送数据报文对象，激活 MultiCAN BSL 工作模式。此后接收到的所有报文中的数据字节将被依次写入 XRAM 中。内部 XRAM 的容量为 2 KB，从而最多可保存 2048 条 8 位指令。

1) CAN 报文对象请参考 BOSCH CAN 规范 V 2.0B 中标准数据帧的定义。

引导程序加载器

一旦所有报文接收完毕，CAN 模块被重新初始化。引导程序加载器终止操作并跳转到地址 0'F000<sub>H</sub>/2'F000<sub>H</sub>（即第一条被载入的指令），开始执行用户代码。此时执行从主机载入到 XRAM 的程序。

注： 引导程序加载器假定所有报文数据均有效。主机将依次发送代码/数据，长度为 8 个代码/数据字节的整数倍。用户可发送的最大报文个数为 256。

19.2.2 CAN 报文对象的定义

主机命令报文对象

在自动波特率检测阶段，主机发送主机命令报文，用于自动波特率检测。因为总线上没有其它节点（点对点连接），主机将连续发送报文。主机发送该命令报文并等待微控制器作出应答。

主机命令报文数据域包含 8 个字节的信息，用于使能 BSL 模式。前两个数据字节，字节 0 和 1 中包含 0x5555。随后两个字节，字节 2 和 3 中包含微控制器返回给主机的应答报文标识符。字节 4 和 5 包含要接收的报文的个数。最后 2 个字节，字节 6 和 7 中包含主机将发送给 XC878 器件的数据报文的标识符。

报文标识符为 555<sub>H</sub>，数据长度码设置为 8。

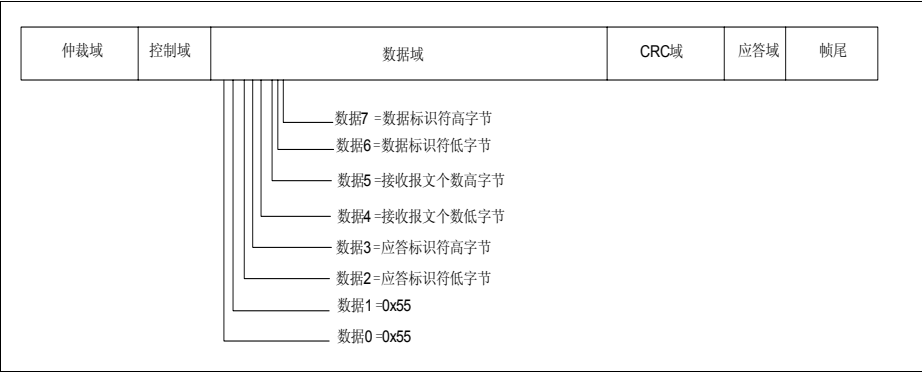


图 19-7 主机命令报文格式

应答报文对象

在应答阶段，在成功确定 CAN 网络波特率之后，微控制器向主机发送应答报文。使用的报文标识符由所接收到主机报文（数据字节 2 和 3）确定。数据长度码设置为 4。

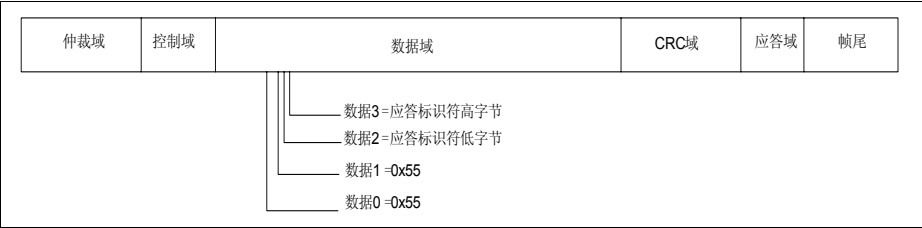


图 19-8 应答报文格式

数据报文对象

在数据接收阶段，主机发送带有标识符的数据报文，该标识符由主机命令报文中的数据字节 6 和 7 给出。数据域包含 BSL 模式所需的用户代码 / 数据。接收到的数据被载入 XRAM 中。

19.2.3 用户定义的 MultiCAN BSL

OSC 值，用于指定（和器件相连的）片外振荡器的频率，该参数由用户设定，存放在 P-Flash 模块的最高地址中。

**表 19-7** 给出 Flash 未被保护时，用户定义的参数的存放地址、设定值以及缺省值。为了保证参数的有效性，需要将真实值和其取反值一起编程。通过检查参数的取反值、真实值和 01<sub>H</sub> 相加的结果是否为 00<sub>H</sub>，可验证该数据是否有效。

表 19-7 用户定义的 MultiCAN BSL 参数

地址 <sup>1)</sup>	参数	取值
0'EFFE <sub>H</sub>	OSC	指示外部振荡器频率的字节 00 <sub>H</sub> : 4 MHz 01 <sub>H</sub> : 5 MHz 02 <sub>H</sub> : 8 MHz 03 <sub>H</sub> : 12 MHz 其它值: 8 MHz （缺省值）
0'EFFF <sub>H</sub>	OSC	指示外部振荡器频率的字节 FF <sub>H</sub> : 4 MHz FE <sub>H</sub> : 5 MHz FD <sub>H</sub> : 8 MHz FC <sub>H</sub> : 12 MHz 其它值: 8 MHz （缺省值）

<sup>1)</sup>表中给出的地址假定器件含有 64 KB Flash 存储器。对于含有 52 KB Flash 存储器的器件，所用 OSC 和 OSC 地址分别为 0'BFFE<sub>H</sub> 和 0'BFFF<sub>H</sub>。

## 19.3 其它 BSL 模式

其它 BSL 模式为一种用户定义的 BSL 代码的模式。BSL 代码编程到 Flash 存储器中，起始地址由 JumpAddrH（跳转地址高位字节）和 JumpAddrL（跳转地址低位字节）给出。通过表 19-1 中的有效 AltBSL 密码进入到 BSL 模式中。这些参数由用户指定，保存在 P-Flash 模块的最高地址处。

用户需要按照表 19-8 中的格式编程 AltBSL 密码、JumpAddrH 和 JumpAddrL。为了确保参数的有效性，AltBSL 密码值和其反向值一起编程。另外，需要将 LJMP 操作码 (02<sub>H</sub>) 编程到表 19-8 给出的地址中。

表 19-8 与未保护的 Flash 相关的用户自定义参数

地址	用户自定义值	规则 / 范围
0'BFFC <sub>H</sub> <sup>1)</sup> 0'EFFC <sub>H</sub> <sup>2)</sup>	AltBSL 密码	00 <sub>H</sub> – 0FF <sub>H</sub>
0'BFFD <sub>H</sub> <sup>1)</sup> 0'EFFD <sub>H</sub> <sup>2)</sup>	AltBSL 密码	-
0'BFFA <sub>H</sub> <sup>1)</sup> 0'EFFA <sub>H</sub> <sup>2)</sup>	JumpAddrH	00 <sub>H</sub> – 0FF <sub>H</sub> <sup>3)</sup>
0'BFFB <sub>H</sub> <sup>1)</sup> 0'EFBB <sub>H</sub> <sup>2)</sup>	JumpAddrL	00 <sub>H</sub> – 0FF <sub>H</sub> <sup>3)</sup>
0'BFF9 <sub>H</sub> <sup>4)</sup> 0'EFF9 <sub>H</sub> <sup>4)</sup>	02 <sub>H</sub> (LJMP 操作码)	-

1) 对应于 52 KB Flash 存储器的器件。

2) 对应于 64 KB Flash 存储器的器件。

3) 根据 Flash 器件信号，用户需要确保 Flash 地址有效。

4) 因为 P-Flash 的起始地址应该为偶数，用户需要确保地址 0xFF8<sub>H</sub> 也被编程。

如果设置的 AltBSL 密码无效，则根据衍生器件型号芯片进入到 LIN 或 UART/MultiCAN BSL 模式。

[www.infineon.com](http://www.infineon.com)