

XC164CM

内嵌 C166SV2 核的 16 位单片微控制器
第 1 卷（共 2 卷）：系统单元

Microcontrollers



Never stop thinking

Edition 2006 - 03

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

**© Infineon Technologies AG 2006.
All Rights Reserved.**

Legal Disclaimer

The information given in this document shall in no event be regarded as a guarantee of conditions or characteristics ("Beschaffenheitsgarantie"). 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 your nearest Infineon Technologies Office (www.infineon.com).

Warnings

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

Infineon Technologies Components may only be used in life-support devices or systems 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.

XC164CM

内嵌 C166SV2 核的 16 位单片微控制器
第 1 卷（共 2 卷）：系统单元

Microcontrollers



Never stop thinking

XC164CM, 第 1 卷（共 2 卷）：系统单元

版本信息：V1.2, 2006 - 03

先前的版本: V1.1, 2005-11
 V1.0, 2005-06

页	内容
从版本 V1.1, 2005-11 到版本 V1.2, 2006-02 的主要修正	
9-1	更正 EBC 模块章节标题：“外部” → “LXBus”。
1-2	在衍生器件列表中添加新衍生器件。
9-9	章节“LXBus 访问控制和信号产生”描述重复，删除该部分。
从版本 V1.0, 2005-06 到版本 V1.1, 2005-11 的主要修正	
	更正手册中出现的错误。
	更正端口章节、CPU 章节、体系架构概况章节中对 EBC 外部访问的错误描述，仅执行内部 LXBus 访问。
5-35	删除 EX6IN 和 EX7IN，六个快速外部中断可用。
6-14	添加 RSTCON 寄存器复位值的描述。
6-33	添加 SYSSTAT.OSCSTAB 位。
7-1	更正 P0 口、P3 口。
3-24	添加描述“程序 Flash 编程和安全扇区编程的相互配合”。

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（以及本手册的相关参考资料）发送至：

mcdocu-chinesecomments.XIY@infineon.com



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

目录:

本用户手册分为两卷：“系统单元”和“外设单元”。为了便于用户的使用，目录（以及关键字索引）将两卷中的章节（以及关键字）统一列出，以便用户快速查阅相关内容（卷[1]或卷[2]）。

1	简介	1-1
1.1	16 位微控制器系列产品	1-3
1.2	基本特性	1-5
1.3	缩写	1-10
1.4	命名规则	1-11
2	体系架构概况	2-1
2.1	基本 CPU 概念及优化	2-2
2.1.1	高指令带宽/快速执行	2-5
2.1.2	功能强大的执行单元	2-6
2.1.3	高性能的分支、调用和循环处理	2-7
2.1.4	统一、优化的指令格式	2-8
2.1.5	可编程多优先级中断系统	2-9
2.1.6	系统资源接口	2-10
2.2	片上系统资源	2-11
2.3	片上外设模块	2-14
2.4	时钟产生	2-28
2.5	功率管理特性	2-29
2.6	片上调试支持（OCDS）	2-30
2.7	保护位	2-31
3	存储器结构	3-1
3.1	地址映射	3-3
3.2	特殊功能寄存器区	3-5
3.3	数据存储区	3-9
3.4	程序存储区	3-11
3.5	系统堆栈	3-13
3.6	IO 区	3-14
3.7	存储器边界越界	3-15

3.8	片上程序 Flash 模块	3-16
3.8.1	Flash 工作模式	3-18
3.8.2	命令序列	3-19
3.8.3	纠错与数据完整性	3-25
3.8.4	保护与安全特征	3-28
3.8.5	Flash 状态信息	3-33
3.8.6	操作控制和错误处理	3-36
3.9	程序存储器控制	3-38
3.9.1	地址映射	3-39
3.9.2	Flash 存储器访问	3-40
3.9.3	IMB 控制功能	3-42
4	中央处理器（CPU）	4-1
4.1	CPU 的组成	4-3
4.2	指令读取和程序流控制	4-4
4.2.1	分支检测和分支预测规则	4-6
4.2.2	预测正确的指令流	4-7
4.2.3	预测错误的指令流	4-9
4.3	指令处理流水线	4-11
4.3.1	通用寄存器引起的流水线冲突	4-12
4.3.2	间接寻址模式引起的流水线冲突	4-15
4.3.3	存储器带宽引起的流水线冲突	4-16
4.3.4	CPU-SFR 更新引起的流水线冲突	4-19
4.4	CPU 配置寄存器	4-26
4.5	通用寄存器的使用	4-30
4.5.1	GPR 寻址模式	4-33
4.5.2	上下文切换	4-35
4.6	代码寻址	4-40
4.7	数据寻址	4-42
4.7.1	短寻址模式	4-43
4.7.2	长寻址模式	4-45
4.7.3	间接寻址模式	4-49
4.7.4	DSP 寻址模式	4-51
4.7.5	系统堆栈	4-57

4.8	标准数据处理	4-61
4.8.1	16 位加法器/减法器、阵列移位器和 16 位逻辑单元	4-64
4.8.2	位操作单元	4-65
4.8.3	乘除单元	4-66
4.9	DSP 数据处理（MAC 单元）	4-68
4.9.1	数字的表示和舍入	4-70
4.9.2	16 位×16 位有符号/无符号乘法器和换算器	4-71
4.9.3	级联单元	4-71
4.9.4	一位换算器	4-71
4.9.5	40 位加法/减法器	4-71
4.9.6	数据限制器	4-71
4.9.7	累加移位器	4-72
4.9.8	40 位有符号累加器寄存器	4-73
4.9.9	MAC 单元状态字 MSW	4-74
4.9.10	重复计数器 MRW	4-76
4.10	常数寄存器	4-78
5	中断与强制中断功能	5-1
5.1	中断系统结构	5-2
5.2	中断仲裁与控制	5-4
5.3	中断向量表	5-9
5.4	外围事件控制器通道的操作	5-18
5.4.1	PEC 源指针与目的指针	5-22
5.4.2	PEC 传送控制	5-25
5.4.3	支持数据链的通道链接模式	5-26
5.4.4	PEC 中断控制	5-27
5.5	中断和 PEC 服务请求的优先级排序	5-29
5.6	上下文切换与状态保存	5-31
5.7	中断节点共享	5-34
5.8	外部中断	5-35
5.9	OCDS 请求	5-41
5.10	中断服务请求的响应延迟	5-41
5.11	强制中断功能	5-43
6	通用系统控制功能	6-1

6.1	系统复位	6-2
6.1.1	复位源和复位阶段	6-3
6.1.2	复位后的状态	6-5
6.1.3	特定应用的初始化	6-8
6.1.4	系统启动配置	6-10
6.1.5	复位行为控制	6-12
6.2	时钟产生	6-13
6.2.1	振荡器	6-14
6.2.2	时钟产生和频率控制	6-16
6.2.3	时钟分配	6-23
6.2.4	振荡器看门狗	6-24
6.2.5	中断产生	6-24
6.2.6	外部时钟信号的产生	6-25
6.3	中央系统控制功能	6-29
6.3.1	状态指示	6-32
6.3.2	复位源指示	6-33
6.3.3	外设关闭握手机制	6-34
6.3.4	调试系统控制	6-35
6.3.5	寄存器安全机制	6-37
6.4	功率管理	6-41
6.4.1	功耗降低模式	6-41
6.4.2	降低时钟频率	6-44
6.4.3	灵活的外设管理	6-44
6.5	看门狗定时器（WDT）	6-46
6.6	ID 控制模块	6-51
7	并行端口	7-1
7.1	输入阈值控制	7-2
7.2	输出驱动器控制	7-3
7.3	端口复用功能	7-7
7.4	P1 口	7-8
7.5	P3 口	7-19
7.6	P5 口	7-37
7.7	P9 口	7-42

8	专用引脚	8-1
9	LXBus 控制器（EBC）	9-1
9.1	时序原则	9-2
9.1.1	基本总线周期协议	9-2
9.1.2	总线周期示例：最快访问周期	9-3
9.2	功能描述	9-4
9.2.1	配置寄存器简介	9-4
9.2.2	时序配置寄存器 TCONCS7	9-4
9.2.3	功能配置寄存器 FCONCS7	9-6
9.2.4	地址窗选择寄存器 ADDRSEL7	9-7
9.2.5	对 TwinCAN 的访问控制	9-9
9.2.6	关机控制	9-9
9.3	EBC 寄存器表	9-10
10	引导程序加载器	10-1
10.1	进入引导程序加载模式	10-2
10.2	加载启动代码	10-4
10.3	退出引导程序加载模式	10-4
10.4	选择 BSL 波特率	10-5
11	调试系统	11-1
11.1	简介	11-1
11.2	调试接口	11-2
11.3	OCDS 模块	11-3
11.3.1	调试事件	11-5
11.3.2	调试动作	11-6
11.4	Cerberus	11-7
11.4.1	功能概述	11-7
12	指令集概述	12-1
13	器件规范	13-1

14	通用定时器单元	14-1
14.1	定时器模块 GPT1	14-2
14.1.1	GPT1 核心定时器 T3 的控制	14-4
14.1.2	GPT1 核心定时器 T3 的工作模式	14-8
14.1.3	GPT1 辅助定时器 T2/T4 的控制	14-15
14.1.4	GPT1 辅助定时器 T2/T4 的工作模式	14-18
14.1.5	GPT1 时钟信号控制	14-28
14.1.6	GPT1 定时器寄存器	14-31
14.1.7	GPT1 定时器的中断控制	14-32
14.2	定时器模块 GPT2	14-33
14.2.1	GPT2 核心定时器 T6 的控制	14-35
14.2.2	GPT2 核心定时器 T6 的工作模式	14-39
14.2.3	GPT2 辅助定时器 T5 的控制	14-42
14.2.4	GPT2 辅助定时器 T5 的工作模式	14-45
14.2.5	GPT2 寄存器 CAPREL 工作模式	14-50
14.2.6	GPT2 时钟信号控制	14-55
14.2.7	GPT2 定时器寄存器	14-58
14.2.8	GPT2 定时器和 CAPREL 的中断控制	14-59
14.3	GPT 模块接口	14-60
15	实时时钟	15-1
15.1	确定 RTC 的时间基准	15-2
15.2	RTC 运行控制	15-5
15.3	RTC 工作模式	15-6
15.3.1	48 位定时器操作	15-9
15.3.2	系统时钟操作	15-9
15.3.3	周期性中断的产生	15-10
15.4	RTC 中断产生	15-11
16	模数转换器	16-1
16.1	模式选择	16-4
16.1.1	兼容模式	16-4
16.1.2	增强模式	16-6
16.2	ADC 操作	16-10
16.2.1	固定通道转换模式	16-13

16.2.2	自动扫描转换模式	16-14
16.2.3	等待读取模式	16-15
16.2.4	通道插入模式	16-16
16.3	自动校准	16-19
16.4	转换时间控制	16-20
16.5	ADC 中断控制	16-23
16.6	ADC 模块接口	16-24
17	捕获/比较单元 CAPCOM2	17-1
17.1	CAPCOM2 定时器	17-4
17.2	CAPCOM2 定时器中断	17-9
17.3	捕获/比较通道	17-10
17.4	捕获模式操作	17-13
17.5	比较模式操作	17-14
17.5.1	比较模式 0	17-15
17.5.2	比较模式 1	17-15
17.5.3	比较模式 2	17-18
17.5.4	比较模式 3	17-18
17.5.5	双寄存器比较模式	17-22
17.6	比较输出信号的产生	17-25
17.7	单次事件操作	17-27
17.8	交错和非交错操作	17-29
17.9	CAPCOM2 中断	17-34
17.10	外部输入信号的要求	17-36
17.11	CAPCOM2 单元接口	17-37
18	捕获/比较单元 6 (CAPCOM6)	18-1
18.1	定时器 T12 模块	18-4
18.1.1	定时器 T12 的操作	18-7
18.1.2	T12 的比较模式	18-12
18.1.3	死区时间产生	18-22
18.1.4	T12 捕获模式	18-25
18.1.5	类磁滞控制模式	18-30
18.2	定时器 T13 模块	18-31
18.2.1	定时器 T13 的操作	18-34

18.2.2	T13 的比较模式	18-39
18.3	定时器模块控制	18-44
18.4	多通道模式	18-51
18.5	霍尔传感器模式	18-54
18.5.1	霍尔序列比较逻辑	18-55
18.5.2	霍尔序列采样	18-56
18.5.3	用定时器 T12 模块实现无刷直流电机控制	18-57
18.5.4	霍尔模式标志位	18-59
18.6	强制中断处理	18-65
18.7	输出调制控制	18-69
18.8	映射寄存器传送控制	18-74
18.9	中断产生	18-76
18.10	暂停模式	18-85
18.11	CAPCOM6 单元接口	18-86
19	异步/同步串行接口（ASC）	19-1
19.1	操作概述	19-3
19.2	异步工作	19-5
19.2.1	异步数据帧	19-6
19.2.2	异步发送	19-9
19.2.3	发送 FIFO 操作	19-9
19.2.4	异步接收	19-12
19.2.5	接收 FIFO 操作	19-12
19.2.6	FIFO 透明模式	19-15
19.2.7	IrDA 模式	19-16
19.2.8	异步模式中 RxD/TxD 数据路径选择	19-18
19.3	同步操作	19-19
19.3.1	同步发送	19-20
19.3.2	同步接收	19-20
19.3.3	同步时序	19-20
19.4	波特率产生	19-22
19.4.1	异步模式下的波特率产生	19-22
19.4.2	同步模式下的波特率产生	19-26
19.5	自动波特率检测	19-27

19.5.1	操作概述	19-27
19.5.2	串行帧自动波特率检测	19-28
19.5.3	波特率选择和计算	19-30
19.5.4	自动波特率检测成功后，改写寄存器	19-34
19.6	硬件错误检测功能	19-35
19.7	中断	19-36
19.8	寄存器	19-40
19.9	ASC 模块接口	19-59
20	高速同步串行接口	20-1
20.1	介绍	20-1
20.2	工作概述	20-1
20.2.1	工作模式选择	20-3
20.2.2	全双工操作	20-9
20.2.3	半双工操作	20-11
20.2.4	连续传送	20-13
20.2.5	波特率产生	20-13
20.2.6	检错机制	20-15
20.2.7	SSC 寄存器总结	20-17
20.2.8	端口配置要求	20-18
20.3	SSC 模块接口	20-19
21	TwinCAN 模块	21-1
21.1	内核描述	21-1
21.1.1	概述	21-1
21.1.2	TwinCAN 控制外壳	21-4
21.1.2.1	初始化处理	21-4
21.1.2.2	中断请求压缩器	21-5
21.1.2.3	全局控制和状态逻辑	21-6
21.1.3	CAN 节点控制逻辑	21-7
21.1.3.1	概述	21-7
21.1.3.2	定时控制单元	21-9
21.1.3.3	位流处理器	21-11
21.1.3.4	错误处理逻辑	21-11
21.1.3.5	节点中断处理	21-12

21.1.3.6	报文中断处理	21-13
21.1.3.7	中断指示	21-13
21.1.4	报文处理单元	21-15
21.1.4.1	仲裁和验收屏蔽寄存器	21-16
21.1.4.2	数据帧和远程帧处理	21-17
21.1.4.3	发送报文对象处理	21-18
21.1.4.4	接收报文对象处理	21-20
21.1.4.5	单数据传送模式	21-22
21.1.5	CAN 报文对象缓存（FIFO）	21-23
21.1.5.1	CAN 控制器对缓存区的访问	21-25
21.1.5.2	CPU 对缓存区的访问	21-26
21.1.6	网关报文处理	21-27
21.1.6.1	正常网关模式	21-28
21.1.6.2	带 FIFO 缓存的正常网关模式	21-32
21.1.6.3	共享网关模式	21-35
21.1.7	TwinCAN 模块编程	21-40
21.1.7.1	CAN 节点 A/B 设置	21-40
21.1.7.2	报文对象的初始化	21-40
21.1.7.3	报文传送控制	21-41
21.1.8	回环模式	21-44
21.1.9	单次发送试验功能	21-45
21.1.10	模块时钟要求	21-45
21.2	TwinCAN 寄存器描述	21-46
21.2.1	寄存器映射	21-46
21.2.2	CAN 节点 A/B 寄存器	21-48
21.2.3	CAN 报文对象寄存器	21-63
21.2.4	全局 CAN 控制/状态寄存器	21-80
21.3	XC164CM 模块的实现	21-82
21.3.1	TwinCAN 模块接口	21-82
21.3.2	与 TwinCAN 模块相关的外部寄存器	21-83
21.3.2.1	系统寄存器	21-84
21.3.2.2	端口寄存器	21-85
21.3.2.3	中断寄存器	21-87

21.3.3	寄存器表	21-88
22	寄存器组	22-1
22.1	PD+BUS 外设	22-1
22.2	LXBUS 外设	22-13
	关键字索引	1

1 简介

嵌入式控制应用领域正在快速增长，这对当今微控制器的实时性提出了很高要求。为了实现复杂的控制算法，就必须能够处理大量的数字和模拟输入信号，并在规定的最大响应时间内产生正确的输出信号。此外，嵌入式控制应用通常还要求电路板体积小、功耗低、系统成本低。

因此，嵌入式控制应用要求微控制器具备以下功能：

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

随着嵌入式控制系统的复杂度日益增长，新型高端嵌入式控制系统对微控制器提出了更高要求，它必须具备比传统 8 位微控制器更高的 CPU 性能和更强大的外设功能。因此，英飞凌决定开发 16 位 CMOS 微控制器系列，而不考虑与先前 8 位微控制器系统的兼容性。

16 位微控制器系列的体系架构继续沿用流行的英飞凌 8 位微控制器系列中成熟的硬件和软件概念。

关于本手册

本用户手册描述了英飞凌 XC166 系列 16 位微控制器的功能。这些微控制器的功能基本相同，不过不同型号的产品各自又具备独特特性，描述如下。

本手册的内容涵盖了以下几种衍生器件的所有特性：

表 1-1 XC164xx 衍生器件一览表

衍生器件	xx	特定功能模块及个数		
		TwinCAN	ADC	CAPCOM6
XC164xx-8F	CM	1	1	1
64KB 程序 Flash	GM	1	1	–
2KB DSRAM	SM	–	1	1
2KB PSRAM	TM	–	1	–
2KB DPRAM	KM	1	–	–
2 × ASC, 2 × SSC	LM	–	–	–
1 × GPT12E, 1 × CAPCOM2				
XC164xx-4F	CM	1	1	1
32KB 程序 Flash	GM	1	1	–
0KB DSRAM	SM	–	1	1
2KB PSRAM	TM	–	1	–
2KB DPRAM	KM	1	–	–
2 × ASC, 2 × SSC	LM	–	–	–
1 × GPT12E, 1 × CAPCOM2				

本手册适用于表中列出的衍生器件，并给出不同器件可用的温度范围和封装形式。

为简化起见，这些不同的器件在本手册中统一用 **XC164CM** 表示，各种器件的完整型号名请参阅相应的数据手册。

本手册中的某些章节不适用于现有的、或计划开发的 **XC164CM** 的所有衍生产品（不同的器件集成有不同的片上存储器或片上外设），在这些章节中，将尽可能添加注释予以说明。

1.1 16 位微控制器系列产品

英飞凌开发的 16 位微控制器系列用来满足实时嵌入式控制应用的高性能需求。其体系架构经过优化，具有更高的指令处理能力，对外部激励（中断）的响应时间更短。系统集成了智能的外设子系统，最大程度降低了 CPU 对外设的干预；同时最大程度降低了通过外部总线接口进行通信的需求。该架构具有高度的灵活性，可满足诸如汽车、工业控制或数据通讯等不同应用领域的需求。

16 位微控制器系列的内核采用模块化的开发概念。所有系列产品均采用同一个高效、控制优化的指令集（第二代产品扩展附加指令）。尽管各种新型系列产品的内部存储器容量、工艺、片上外设集、以及/或者 IO 引脚数目不同，相同的指令集使它们的应用变得简单方便。

系统除集成标准片上外设之外，还可通过 XBUS（外部总线的内部标识）直接集成某些特定应用所需的专用外设模块。

嵌入式控制应用的编程规模正在增大，由于高级语言更易编写、调试和维护，因此受到编程人员的青睐。C166 系列从第二代产品开始支持高级语言编程。

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

C165 和 C167 是**第二代产品**。第二代微控制器更加强大，扩展了支持高级语言编程（HLL）的附加指令集，扩展了地址空间、内部 RAM 容量，并增加了外部总线上不同资源的高效管理。

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

将专用外设集成到系统中，可提高系统性能、同时使电路板上的芯片数目最少。英飞凌 16 位微控制器的第二代产品采用 XBUS 概念，来支持专用外设的集成。XBUS 是外部总线的内部标识，通过标准化所需接口，可以简化外设集成。典型应用即 CAN 模块的集成。

C165 系列是 C167 的功能简化版，系统未集成模数转换单元、CAPCOM 单元和 PWM 产生模块，因此封装较小、功耗较低、设计简单。

C164 系列（C167CS 的衍生产品）以及部分 C161 系列是 16 位微控制器的**第三代产品**，进一步增强了灵活的功率管理。功率管理机制有效控制了微控制器在特定状态下的功耗，从而降低了系统总功耗。

XC16x 系列是 16 位微控制器的**第四代产品**。XC166 系列产品显著提高了 16 位微控制器的系统性能：MAC 单元加入 DSP 功能处理数字滤波器算法，从而大大缩短了乘除运算的时间；五级流水线结构；大多数指令为单周期指令；可遍寻地址空间的 PEC 传送，这些特性均提高了系统性能。可通过片上调试支持单元（OCDS）对目标系统进行调试。

XC16x 系列不同产品的片上存储器¹⁾ 类型不同:

- 掩模可编程 ROM
- Flash 存储器
- OTP 存储器
- 无 ROM 存储器

此外，一些产品还集成有特殊功能单元。

英飞凌可为用户提供各种不同封装、不同温度范围和速度级别的产品。

更多的标准产品和专用产品正在规划和开发中。

注：不是所有的衍生产品都有对应各种温度范围、速度级别、封装和不同程序存储器的产品。

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

注：由于目前推出的大多数 XC164CM 系列产品的体系架构和基本特性（如 CPU 内核和片上外设）均相同，因此除非特别指明，本手册中“XC164CM”的描述也适用于其它衍生产品。

¹⁾ 不是所有的衍生产品都具有各种不同类型的片上存储器。

1.2 基本特性

XC164CM 是英飞凌 16 位单片 CMOS 微控制器系列的增强型产品。XC164CM 将功能和性能扩展的 C166SV2 内核、功能强大的片上外设子系统和片上存储器单元完美结合，并且有效降低了系统功耗。

高性能 XC164CM 具有以下主要特性：

五级流水线高性能 16 位 CPU 和 MAC 单元

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

高效的控制指令集

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

功率管理特性

- 用于改善功耗和 EMC 的门控时钟概念
- 通过时钟产生单元可编程降低系统的运行速度

- 灵活的外设管理，每个外设可单独被禁用
- 可通过快速外部中断或片上 RTC 将系统从休眠模式唤醒
- 输出频率可编程

集成的片上存储器

- 2KB 双口 RAM (DPRAM)，用于存储变量、寄存器组和堆栈
- 高达 2KB¹⁾ 的片上高速数据 SRAM (DSRAM)，用于存储变量和堆栈
- 2KB 片上高速程序/数据 SRAM (PSRAM)，用于存储代码和数据
- 64/32 KB 片上 Flash 程序存储器，用于存储指令或常量

注：系统堆栈可位于整个寻址空间的任何区域。

16 级优先级中断系统

- 80 个具有独立中断向量的中断源，15 级优先级（8 个组优先级）
- 最短内部中断响应延迟为 13 个周期
- 快速外部中断
- 外部中断源可编程选择
- 中断向量表可编程（起始地址和相邻向量的间距）

8 通道外围事件控制器 (PEC)

- 由中断驱动的单周期数据传送
- PEC 中断请求优先级可编程设定（从 15 级到 8 级）
- PEC 传送数目选择
（执行设定数目的 PEC 传送之后转入标准 CPU 中断）
- PEC 结束中断的中断优先级可单独设定
- 无需 CPU 响应中断请求时保存和恢复系统状态的开销
- 24 位源和目的指针地址，支持整个地址空间内的数据传送

智能片上外设子系统

- 14 通道模数转换器，转换精度（10 位或 8 位）和转换时间（降至 2.55μs 或 2.15μs）可编程，支持自动扫描模式和转换通道插入

1) 不同衍生产品的片上 DSRAM 容量大小不同，在第 2 页“关于本手册”中列出衍生器件的型号及相应参数信息。

- 1 个捕获/比较单元，带有两个独立的时间基准；在不同的工作模式下可灵活产生 PWM 信号或记录事件；包括 2 个 16 位定时器/计数器，最大精度 f_{SYS}
- 灵活产生 PWM 信号的捕获/比较单元（CAPCOM6）（3/6 捕获/比较通道和 1 个比较通道）
- 两个多功能通用定时器单元：
 - GPT1: 3 个 16 位定时器/计数器，最大精度 $f_{\text{SYS}}/4$
 - GPT2: 2 个 16 位定时器/计数器，最大精度 $f_{\text{SYS}}/2$
- 两个异步/同步串行通道（USART）
 - 带有波特率发生器，可进行奇偶校验错误、帧错误和过载错误检测
 - 带有自动波特率检测，接收/发送 FIFO，支持 IrDA
- 两个高速同步串行通道（SPI 兼容），数据长度和移位方向可编程
- 控制器局域网络（TwinCAN）模块，V2.0B active 版本，两个节点独立工作，或通过网关交换数据，全功能 CAN/基本 CAN
- 具有闹钟中断的实时时钟
- 看门狗定时器，刷新时间间隔可编程
- 引导程序加载器，用于进行灵活的系统初始化
- 系统配置和控制寄存器的保护管理

片上调试支持（OCDS）

- 片上调试控制器和 JTAG 控制器的相关接口
- JTAG 接口和断点接口
- 硬件、软件和外部引脚断点
- 最多 4 个指令指针断点
- 调试事件控制，如调用监控程序、暂停 CPU、或触发数据传送
- 通过 JTAG 接口控制的专用 DEBUG 指令
- 通过 JTAG 接口访问内部寄存器或存储器的任意地址
- 利用 MOV 插入指令实现单步调试和观察点

多达 47 个具有独立位寻址功能的 IO 引脚

- 输入模式具有三态
- 输入阈值电压可选（非所有引脚）
- 推挽或漏极开路输出模式
- 端口驱动控制可编程

- I/O 电压 5V（内核逻辑和晶振输入电压为 2.5V）

不同的温度范围¹⁾

- -40 至 +85°C
- -40 至 +125°C

英飞凌 CMOS 工艺

- 低功耗 CMOS 工艺，通过灵活的功率管理可使系统工作在空闲、休眠和掉电等省电模式

64 引脚超薄塑料方形扁平（TQFP）封装

- PG-TQFP，尺寸 10×10 mm，0.5 mm（19.7 mil）引脚间距，表面焊接技术

完整的开发支持

英飞凌借助第三方为用户提供微控制器的开发工具。目前全球约有 120 个开发工具供应商，从本地的小生产商到拥有各种产品的跨国公司，都在为英飞凌的 C500、C800、XC800、C166、XC166 和 TriCore 微控制器系列提供功能强大的开发工具，这确保了用户可尽早获取各种不同性价比以及高质量的关键开发工具，如编译器、汇编器、仿真器、调试器或在线仿真器等。

在产品开发的初期，英飞凌就已经和开发工具战略伙伴进行合作，以保证嵌入式系统开发人员可获得可靠、易于使用的开发工具，帮助开发人员以最有效的方式和最短的学习时间学会使用英飞凌的微控制器。

英飞凌 16 位微控制器的开发工具环境包括以下工具：

- 编译器（C/C++）
- 宏汇编器、连接器、定位器、库管理工具，格式转换工具
- 体系架构仿真器
- HLL 调试器
- 实时操作系统
- VHDL 芯片模型
- 在线仿真器（基于 bondout 或标准芯片）
- 插入式仿真器
- 仿真和芯片夹紧适配器，芯片插座
- 逻辑分析反汇编器

1) 不是所有的衍生产品都具有全部温度范围的产品。

- 入门开发工具包
- 带有监控程序的评估板
- 工业板（也用于 CAN、FUZZY、PROFIBUS、FORTH 应用）
- 底层驱动软件（CAN、PROFIBUS、LIN）
- 芯片配置代码生成工具（DaVE）

1.3 缩写

本手册中用到的缩写归纳如下：

JTAG	联合测试行动组
ADC	模数转换器
ALE	地址锁存使能
ALU	算术逻辑单元
ASC	异步/同步串行通道
CAN	控制器局域网（License Bosch）
CAPCOM	捕获和比较单元
CISC	复杂指令集计算
CMOS	互补金属氧化物半导体
CPU	中央处理单元
DMU	数据管理单元
EBC	外部总线控制器
ESFR	扩展特殊功能寄存器
Flash	电可擦除非易失存储器
GPR	通用寄存器
GPT	通用定时器单元
HLL	高级语言
IIC	内部集成电路（总线）
IO	输入/输出
LXBus	外部总线的内部标识
OCDS	片上调试支持
OTP	一次性可编程存储器
PEC	外围事件控制器
PLA	可编程逻辑阵列
PLL	锁相环
PMU	程序管理单元
PWM	脉宽调制
RAM	随机访问存储器
RISC	精简指令集计算

ROM	只读存储器
RTC	实时时钟
SFR	特殊功能寄存器
SSC	同步串行通道

1.4 命名规则

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

描述具有规则结构的寄存器（如端口）时，采用数字编号来描述寄存器中的各位，不采用过多相似的位名称，例如 P5 口的第 3 位通常称为 P5.3。

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

ADC_CTR0 表明寄存器 CTR0 是模块 ADC 的组成部分；SYSCON1.CPSYS 表明位域 CPSYS 是寄存器 SYSCON1 的组成部分。

2 体系架构概貌

XC164CM 的内核架构将 RISC 和 CISC 处理器的优点完美结合。其计算和控制功能由于 MAC 单元的 DSP 功能而变得更加完善。XC164CM 将高性能 CPU 与一系列功能强大的外设单元集成起来，有效的联接成一个单片机系统。片上存储器模块带有专用总线和控制单元，用于存储代码和数据。这些特性促成了 XC164CM 微控制器的高性能，使得 XC164CM 不仅可满足当前的应用，还将适用于未来的工程应用。XC164CM 中还采用了一种 LXBus 总线，用来标识外部总线接口。LXBus 总线提供了一种标准化的方法，可方便的将附加的专用外设集成到标准 XC164CM 系列芯片中。

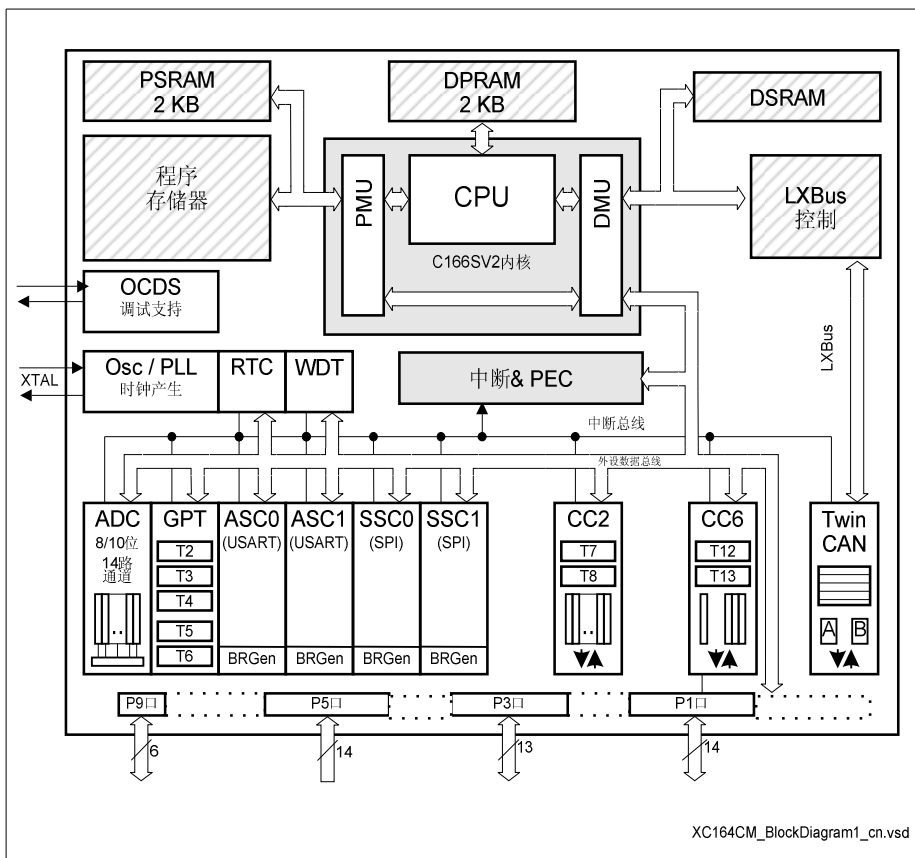


图 2-1 XC164CM 功能框图

不同衍生器件片内集成的程序存储器、DSRAM、以及外设集不同，XC164CM 衍生器件型号及对应参数归纳见[页 1-2](#)。

2.1 基本 CPU 概念及优化

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

XC164CM 中的大多数指令为单周期指令，从而提高了 CPU 性能，同时与 C166 指令保持兼容。此外，XC164CM 出色的 DSP 性能、对不同存储器和外设的并行访问等特性提高了系统的整体性能。

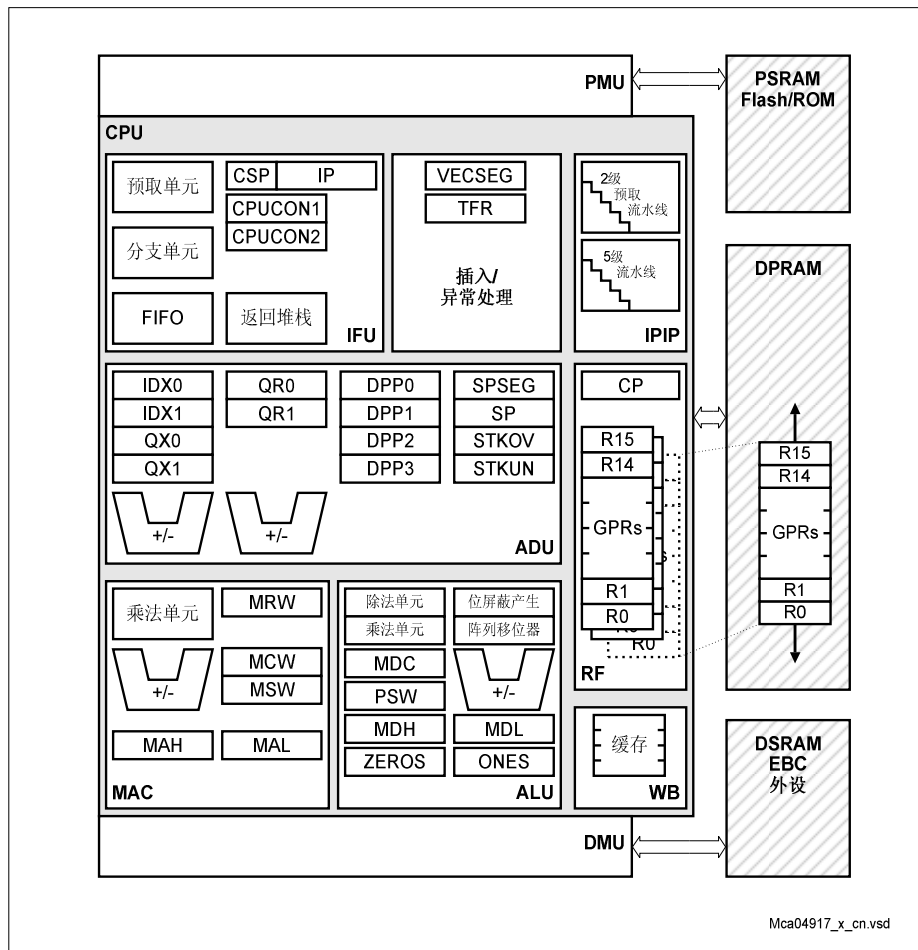


图 2-2 CPU 框图

CPU 特性概述

- 指令与 C166 系列完全向上兼容
- 2 级取指流水线，FIFO 用于指令预取
- 5 级指令执行流水线
- 专用硬件实现流水线上的数据前送（解决数据相依性问题）
- 多种用于数据和指令传送的高带宽总线
- 代码和数据的线性地址空间（冯诺伊曼体系）
- 大多数指令为单时钟周期指令
- 单时钟周期快速乘法运算（16 位 × 16 位）
- 21 个 CPU 时钟周期的快速后台除法运算（32 位/16 位）
- 内嵌的高级 MAC（乘累加）单元：
 - 零延迟的单周期 MAC 指令；MAC 单元内含 16 位 × 16 位乘法器
 - 40 位阵列移位器和 40 位累加器用于溢出处理
 - MAC 指令支持 32 位自动饱和或舍入
 - 直接支持小数运算
 - 每一个 FIR（有限冲激响应滤波器）抽头需要一个 MAC 周期，无循环缓冲管理
- 增强的布尔位操作能力
- 高性能分支、调用和循环处理
- 零周期跳转操作
- 基于寄存器的设计，具有多个可变的寄存器组（存储字节或字操作数）
- 两个附加的快速局部寄存器组，支持快速上下文切换
- 带有上溢/下溢自动检测的系统堆栈
- “快速中断”特性

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

2.1.1 高指令带宽/快速执行

基于硬件的支持，大多数 XC164CM 指令可以在一个时钟周期内执行完成（ $1/f_{CPU}$ ），这些指令包括大多数寻址模式下的算术指令、逻辑指令和转移指令。

一些特殊指令如 **SRST** 或 **PWRDN**，需占用多个机器周期。由于除法指令主要是后台执行，因此可与其它指令并行执行。由于采用预测机制（见**章节 4.2**），被正确预测的分支指令只占用一个周期，或者甚至可以和其它指令并行执行，不单独占用周期（零周期跳转）。

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

两级取指流水线从相应的程序存储器中读取指令并进行预处理：

指令预取：按照预测顺序从 PMU 预取指令。在分支检测单元中进行指令预处理，检测分支。预测逻辑决定是否执行这些分支/跳转指令。

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

五级指令处理流水线执行如下操作：

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

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

存储：读取所有需要的操作数。

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

回写：将结果操作数写回到指定地址。DPRAM 中的操作数通过回写缓存保存。

2.1.2 功能强大的执行单元

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

大多数内部执行模块经过优化，既可执行 8 位运算，也可执行 16 位运算；XC164CM 的指令既可处理字节运算，也为字运算提供了字节的符号扩展；内部总线结构也支持字节或字传送，根据外设请求向外设发送、或从外设读取数据。

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

16 位阵列移位器在一个周期内可实现多位移位，它还支持循环移位和算术移位。

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

地址数据单元（ADU） 包含两个独立的算术单元，用来产生、计算和更新数据访问地址。ADU 执行以下主要任务：

- 标准地址单元支持短寻址、长寻址和间接寻址模式的线性地址运算，还支持数据分页和堆栈处理。
- DSP 地址产生单元中包含一组附加的地址指针和偏移地址寄存器，只用于 CoXXX 指令。

CPU 支持多种功能强大的寻址模式（短寻址、长寻址、间接寻址），用于字、字节和位数据访问。不同的寻址模式格式不同，应用范围不同。

专用位处理指令可对外设进行有效的控制及测试，同时增强了数据处理能力。这些指令可直接访问可位寻址空间中的两个操作数，而无需先将其移入临时标志。逻辑指令只用一条指令就能比较和修改外设的某个控制位。多位移位指令（单周期执行）可避免循环执行单位移位指令（长指令流）。位域指令可通过一条指令来修改某操作数中的多个位元。

2.1.3 高性能的分支、调用和循环处理

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

随后从预测的目标地址中继续预取指令。如果预测正确，即使执行分支指令（即，指令执行是非线性的），随后的指令就可以“无间隙”地进入执行流水线。分支目标预测（见**章节 4.2.1**）使用如下规则：

- **无条件分支：**在这种情况下分支预测几乎不起作用，始终执行跳转、跳转到指定的目标地址。该规则适用于隐含的无条件跳转指令，如 **JMPS**、**CALLR**、或 **RET**，以及含有条件代码“unconditional”的跳转指令（无条件跳转指令），如 **JMPI cc_UC**。
- **固定预测：**常使用分支指令实现循环功能，如跳转返回循环的起始位置。该规则应用于条件跳转指令，如 **JMPR cc_XX** 或 **JNB**。
- **可变预测：**在这种情况下，预测（执行或不执行跳转）被编码在指令中，不同跳转指令的分支预测可编程设定。因此，软件编程人员可优化指令流，跳转到指定的代码段¹⁾。该规则适用于跳转指令 **JMPA** 和 **CALLA**。
- **条件间接分支：**假定始终不执行这些跳转。该规则适用于跳转指令 **JMPI cc_XX, [Rw]** 和 **CALLI cc_XX, [Rw]**。

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

1) 编程工具支持每种预测使用专用助记符（由编程人员决定）；或者支持一般助记符，应用自己的预测规则。

2.1.4 统一、优化的指令格式

为了使流水线的性能最佳，指令集的设计融入了精简指令集计算（RISC）的概念，能够对指令和操作数快速解码，同时减少流水线等待时间。不过，并不排除用户使用一些复杂指令。指令集设计满足了以下目标：

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

使用功能强大的 XC164CM 指令集，CPU 硬件的高性能得以充分发挥。指令集包括以下指令类型：

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

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

2.1.5 可编程多优先级中断系统

XC164CM 提供了 80 个独立的中断节点，分成 16 级优先级，每级内又分 8 个组优先级。大多数中断源和专用的中断节点相连。有些情况下，多个中断源可共用一个中断节点，以节约系统资源。这些中断节点可由多个中断源请求激活，由中断子节点控制寄存器控制。

XC164CM 的下列增强特性保证了多中断源的处理：

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

XC164CM 能够对随机事件快速响应，中断响应时间通常为 13 个时钟周期（在内部程序执行情况下）。快速外部中断输入信号在每个时钟周期都被采样，因此，即使外部信号持续时间极短，仍可被系统识别。

XC164CM 还提供了“硬件强制中断”机制，用以识别并处理在运行过程中出现的异常或错误情况。硬件强制中断会立即引起非屏蔽系统响应，和标准中断服务相似（跳转到指定的中断向量地址上）。由强制中断标志寄存器（TFR）中的标志位来指示是否发生硬件强制中断。除非当前正在处理另一个更高优先级的强制中断服务，否则，硬件强制中断将中断正在执行的任何程序。硬件强制中断服务通常不能被标准中断或 PEC 中断所中断。

软件中断通过“TRAP”指令结合一个强制中断编号来实现。

2.1.6 系统资源接口

XC164CM 通过多总线系统将 CPU 与系统资源联接，这些总线同时传送数据，提升了系统整体性能，避免了由于指令或操作数传送而导致 CPU 停滞。

由于全局寄存器组存放在双口 RAM（DPRAM）中，DPRAM 与 CPU 直接相连。对这些单元的读写操作会影响系统性能，因此，访问 DPRAM 经过精心优化。

程序管理单元（PMU） 控制对片上程序存储器，如 ROM/Flash 和程序/数据 RAM（PSRAM）的访问。

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

数据管理单元（DMU） 控制对片上数据 RAM（DSRAM）、与外设总线连接的片上外设，以及与片上 LXBus 连接的外设的访问。由外部总线控制器（EBC）控制对内部 LXBus 的访问。

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

PMU 和 DMU 紧密配合工作，可进行高速数据互传。互传双向进行：

- **经过 DMU 的 PMU：**从外部存储器读取的代码应该通过 DMU 重新定向到 EBC。不过，用户必须清楚：
 - XC164CM 不支持通过 EBC 访问外部资源
 - 不能从数据 RAM（DSRAM）中读取代码
- **经过 PMU 的 DMU：**可以对 PMU 控制的片上资源进行数据访问，包括以下几种传送类型：
 - 从片上程序 ROM/Flash 读取常数
 - 从片上 PSRAM 读取数据
 - 将数据写入片上 PSRAM（在执行程序之前需要先将数据写入）
 - 编程/擦除片上 Flash 存储器

2.2 片上系统资源

XC164CM 微控制器围绕 CPU 提供了多种功能强大的系统资源。CPU 与这些资源的完美组合使得 XC164CM 微控制器系列产品具有很高的性能。

外围事件控制器（PEC）和中断控制

外围事件控制器可通过传送数据（字或字节）来响应中断请求，该操作只占用一个指令周期，无需保存和恢复系统状态。每个机器周期，中断控制模块会确定各中断源的优先次序。若请求 PEC 服务，则启动 PEC 传送；若请求 CPU 中断服务，则查询保存在 PSW 寄存器中 CPU 的当前优先级，以确定是否正在执行更高优先级的中断。若中断被响应，系统的当前状态压入内部系统堆栈，CPU 转入执行相应的中断服务程序。

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

每进行一次 PEC 服务，PEC 传送计数器减 1，连续传送模式除外。当计数器计数到零时，程序回到 PEC 中断向量位置，执行该标准中断。PEC 传送适用于将寄存器的内容送入/移出存储器。XC164CM 有 8 路 PEC 通道，每路通道均可进行快速、由中断驱动的数据传送。

存储器区域

XC164CM 的存储器空间为冯诺伊曼体系架构。将程序存储器、数据存储器、寄存器和 IO 口组织在同一个线性地址空间内，容量高达 16MB。整个存储器空间可按字节或字访问。片上存储器的特定区域可直接进行位访问。

64KB 片上 Flash 或 ROM 存储器¹⁾ 存储代码或常数。片上 Flash 存储器由四个 8KB 扇区、一个 32KB 扇区¹⁾ 组成。每个扇区可分别进行写保护²⁾、擦除和编程（以 128 字节为单位）。整个 Flash 存储区可读保护。可使用一个密码序列暂时解锁被保护区。Flash 模块既支持 64 位单时钟周期³⁾ 的快速读访问，也具有安全高效的编程和擦除算法。动态纠错为所有读操作提供了极高的数据安全性。

编程 128 字节通常需要 2ms（最长 5ms）；擦除一个扇区通常需要 200ms（最长 500ms）。

ROM 在工厂进行掩膜编程。

注：从片上程序存储器执行程序是所有可能的方案中速度最快的，可使系统性能最高。片上程序存储器的类型由所选衍生器件决定。片上程序存储器还包含 PSRAM。

1) 不同衍生产品的存储器容量不同，在第 2 页“关于本手册”中列出衍生器件的型号及相应参数信息。

2) 每两个 8KB 扇区组合进行写保护。

3) Flash 访问可能需要插入等待状态，这取决于实际的工作频率。

2KB 片上程序 SRAM（PSRAM） 用来存储用户代码或数据。通过 PMU 访问 PSRAM，从而优化了指令读取。

2KB 片上数据 SRAM（DSRAM）¹⁾ 用来存储一般用户数据。通过 DMU 访问 DSRAM，从而优化了数据访问。

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

DPRAM 的高 256 字节可直接位寻址。用作 GPR 时，DPRAM 的任何位置都可位寻址。

CPU 可支配的真正寄存器组由多达 16 个字宽和/或字节宽的全局 GPR 组成，这些 GPR 位于片上 RAM 区。上下文指针（CP）寄存器指定 CPU 每次访问全局寄存器组的基地址。寄存器组的数目只受内部 RAM 空间的限制。为了便于进行参数传递，两个寄存器组可相互重叠。

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

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

特殊功能寄存器 占用三个地址区域：标准特殊功能寄存器区（SFR）占用 512B；扩展特殊功能寄存器区（ESFR）占用 512B；内部 IO 区（XSFR）占用 4KB。SFR 是字宽寄存器，用来控制和监控不同片上单元的功能。未使用的 SFR 地址被保留，用于对 XC166 的后续系列产品进行功能扩展。保留地址区不能被访问、或者将其置零，以保证向上兼容。

1) 不同衍生产品的片上 DSRAM 容量不同，在**第 2 页“关于本手册”**中列出衍生器件的型号及相应参数信息。

表 2-1 XC164CM 存储器映射

地址区间	起始地址	结束地址	区域大小 ¹⁾	备注
Flash 寄存器空间	FF'F000 _H	FF'FFFF _H	4KB	2)
保留 (Acc.trap)	F8'0000 _H	FF'EFFF _H	<0.5MB	减去 Flash 寄存器
保留用作 PSRAM	E0'0800 _H	F7'FFFF _H	<1.5MB	减去 PSRAM
程序 SRAM	E0'0000 _H	E0'07FF _H	2KB	最大
保留用作程序存储	C1'0000 _H	DF'FFFF _H	<2MB	减去 Flash
程序 Flash	C0'0000 _H	C0'FFFF _H	64KB	3)
保留	40'0000 _H	BF'FFFF _H	8MB	—
保留	20'0800 _H	3F'FFFF _H	<2MB	减去 TwinCAN
TwinCAN 寄存器	20'0000 _H	20'07FF _H	2KB	通过 EBC 访问
保留	01'0000 _H	1F'FFFF _H	<2MB	减去段 0
数据 RAM 和 SFR	00'8000 _H	00'FFFF _H	32 KB	部分使用
保留	00'0000 _H	00'7FFF _H	32 KB	—

1) 标有“<”的区域大小略小于标注值，见“备注”列。

2) 访问未定义的寄存器地址将返回强制中断代码（1E9B_H）。

3) 不同衍生产品的存储器容量不同，在**第 2 页“关于本手册”**中列出衍生器件的型号及相应参数信息。

2.3 片上外设模块

XC166 系列将外设和内核明确分离。该结构支持最大数量的并行操作；可以方便的添加或删除外设，而无需改动内核。每个功能模块独立处理各自的数据，并通过公共总线交换信息。通过设置特殊功能寄存器（SFR）来控制外设。SFR 位于标准 SFR 区（00'FE00_H ... 00'FFFF_H）、扩展 ESFR 区（00'F000_H ... 00'F1FF_H），或内部 IO 区（00'E000_H ... 00'EEEE_H）。

通过这些内嵌外设可使 CPU 与外部资源连接，或将原本需要在片外添加的功能集成到片内。

XC164CM 的基本外设包括：

- 两个通用定时器模块（GPT1 和 GPT2）
- 两个异步/同步串行接口（ASC0 和 ASC1）
- 两个高速串行接口（SSC0 和 SSC1）
- 一个看门狗定时器
- 一个捕获/比较单元（CAPCOM2）
- 增强型捕获/比较单元（CAPCOM6）
- 一个 10 位模数转换器（ADC）
- 一个实时时钟（RTC）
- 四个 I/O 端口，共 47 根 I/O 线

LXBus 是外部总线的内部标识，不支持位寻址。通过 EBC 访问与 LXBus 连接的片上资源，如同访问片外资源一样。LXBus 将片上外设连接到 CPU：

- TwinCAN 模块，具有两个 CAN 节点和网关功能

每个外设都包含一组特殊功能寄存器（SFR），用来控制外设的功能、暂时保存中间的数据结果；还包含一组状态标志位，用于监控外设的状态。每个外设可独立选择时钟源，这些时钟信号由主时钟经 2 的整数倍分频产生。

外设接口

片上外设通常有两种接口：一个与 CPU 的接口，一个与外部硬件的接口。CPU 与片上外设的通信通过特殊功能寄存器（SFR）和中断来实现。SFR 用作外设的控制/状态和数据寄存器。中断请求由外设根据特定的事件产生，如操作完成、操作错误等。

片上外设与外部硬件的通信通过并口上的特定引脚实现。此时，端口引脚由片上外设控制（用作输出），或由外部硬件控制（用作输入）。与引脚的通用 I/O 功能相对应，这称为引脚的“复用（输入或输出）功能”。

外设时序

CPU 和外设的内部操作基于主时钟（ f_{MC} ）控制。时钟产生单元通过片上振荡器从晶振产生主时钟、或从外部时钟信号产生主时钟。送入外设的门控时钟信号独立于 CPU 时钟信号。空闲模式下，CPU 时钟停止，外设继续工作。每个状态下 CPU 均可访问外设 SFR。如果软件设定 SFR 和外设修改 SFR 同时发生，软件写操作占优。有关外设时序的信息将在各个外设章节中予以详细说明。

编程提示

- **访问 SFR：**所有 SFR 位于存储器空间的数据页 3。以下寻址方式支持访问 SFR：
 - 间接或直接 **16 位（mem）寻址**必须确保所用的数据页指针（DPP0 ... DPP3）选择数据页 3。
 - 通过外围事件控制器（PEC）访问 SFR 时，用 SRCPx 和 DSTPx 指针寻址，不使用数据页指针。
 - 对标准 SFR 区的**短 8 位（reg）寻址**，不使用数据页指针，该 512B 区域内的寄存器被直接访问。
 - 对扩展 **ESFR 区**的**短 8 位（reg）寻址**，需要切换到 512B 的扩展 SFR 区，该操作通过扩展指令 EXTR、EXTP（R）、EXTS（R）来实现。
- 通过间接或直接 16 位（mem）寻址对字宽 SFR 进行**字节写操作**、或通过 PEC 进行字节传送时，非寻址字节应强制置零。通过短 8 位（reg）寻址对 SFR 进行字节写操作时，只能访问 SFR 的低位字节，高位字节强制置零。因此推荐使用位域指令（BFLDL 和 BFLDH）对 SFR 的任意字节的任意位进行写操作，该指令不会影响非寻址字节和非选定位。
- **保留位：**XC164CM 的 SFR 中有些位标记为“保留”。禁止用户软件对保留位置 1。这些位目前没有使用，可能会用作扩展后续产品的新功能，启用保留位时，“1”代表新功能的有效状态，“0”代表新功能的有效状态。因此将保留位置 0 保证了当前软件的可移植性。进行读操作时，保留位应该被忽略或屏蔽。

捕获/比较单元（CAPCOM2）

CAPCOM2 单元支持多达 16 路通道上时序的产生和控制，最大精度为 1 个系统时间周期（交错模式下最大精度为 8 个系统周期）。CAPCOM2 单元通常用于处理高速 I/O 任务，如脉冲和波形的发生、脉宽调制（PWM）、数模（D/A）转换、软件定时、或记录外部事件时间信息。

CAPCOM2 中有两个 16 位定时器（T7, T8），它们各有相应的重载寄存器，为捕获/比较寄存器组提供两个独立的时间基准。

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

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

CAPCOM2 模块中的 10 个寄存器各自对应一个与之相关的端口引脚，捕获功能下用作输入，指示发生比较事件时用作输出。

表 2-2 比较模式（CAPCOM2）

比较模式	功能
模式 0	仅产生中断的比较模式； 每个定时器周期可产生多个比较中断
模式 1	每次比较匹配时输出引脚翻转； 每个定时器周期可产生多个比较事件
模式 2	仅产生中断的比较模式； 每个定时器周期只产生一次比较中断
模式 3	匹配时引脚置 1；比较定时器溢出时引脚置 0； 每个定时器周期只能产生一次比较事件
双寄存器模式	两个寄存器控制同一引脚； 每次比较匹配时输出引脚翻转； 每个定时器周期可产生多个比较事件
单次事件模式	产生单次信号跳变或脉冲； 可在任何比较模式下使用

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

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

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

捕获/比较单元 CAPCOM6

CAPCOM6 单元支持最多三路 16 位捕获/比较通道、一路独立的 10 位比较通道上时序的产生和控制。

比较模式下，CAPCOM6 单元的每路通道产生两个输出信号，它们极性相反、脉冲翻转区间互不重叠（死区时间控制）。比较通道产生的单路 PWM 输出信号可进一步用于调制捕获/比较输出信号。

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

CAPCOM6 可产生边沿对齐或中心对齐的 PWM 输出信号。这些 PWM 输出信号可连续产生，也可只产生一次（单次模式）。

比较定时器 T12（16 位）和 T13（10 位）的时钟源是经预分频处理的系统时钟。

在电机控制应用中（无刷直流驱动），可由比较定时器 T12、或中断输入口上的霍尔传感序列（块切换）控制产生各种多通道 PWM 信号。霍尔传感器模式为霍尔输入信号提供了噪声滤波、支持自动转速测量。

CAPCOM6 具有强制中断功能，可使系统快速急停、无需 CPU 干预。外部信号（CTR_{AP}）会触发输出引脚切换到选定的逻辑电平以保护相连的功率器件。

通用定时器（GPT12E）单元

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

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

模块 GPT1 中的三个定时器 T2、T3、T4 可被分别设置为四种基本工作模式之一：定时器模式、门控定时器模式、计数器模式和增量接口模式。定时器模式下，定时器的输入时钟来自经过预分频处理的系统时钟（预分频因子可编程设定）；计数器模式下，可用外部事件作为定时器的时钟源。

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

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

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

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

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

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

锁存器的状态可以用作定时器 T5 的输入时钟，可从引脚 T6OUT 输出。定时器 T6 的上溢/下溢信号还可用作 CAPCOM2 定时器 T7 的输入时钟；还可触发重载，将 CAPREL 寄存器的值重新装入定时器 T6 中。

一旦端口引脚（CAPIN）上的外部信号发生跳变，CAPREL 寄存器可捕获定时器 T5 的计数值。捕获操作完成后，可将定时器 T5 清零。这使得 XC164CM 能够测量绝对时间间隔、或者实现倍频，而无需软件干预。

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

实时时钟

XC164CM 的实时时钟（RTC）模块直接由独立时钟源驱动，该时钟信号由片上主振荡器经预分频产生（ $f_{\text{RTC}} = f_{\text{OSCm}}/32$ ）。因此，RTC 的时钟和 XC164CM 的时钟产生模式无关。

RTC 由一组分频模块构成：

- 8:1 分频器（可编程选择开启或关闭）
- 可重载 16 位定时器 T14
- 32 位 RTC 定时器（通过寄存器 RTCH 和 RTCL 访问），由以下定时器级联构成：
 - 可重载 10 位定时器
 - 可重载 6 位定时器
 - 可重载 6 位定时器
 - 可重载 10 位定时器

所有定时器递增计数。每个定时器可各自产生中断请求，所有的中断请求逻辑组合，构成一个公共中断节点请求。此外，T14 可以产生一个独立的中断节点请求。

注：为了保持正确的系统时间，与 RTC 相关的寄存器不受复位影响，即使执行突发的复位操作。

RTC 模块的不同用途包括：

- 系统时钟，决定当前时间和日期
在空闲模式、休眠模式和掉电模式下均可选用
- 周期性中断，提供与 CPU 频率和其它资源无关的系统时间标记，可规律的将系统从空闲模式唤醒
- 48 位定时器，用于长时间间隔的测量（最长时间跨度 >100 年）
- 预定时间的闹钟中断，用于唤醒系统

A/D 转换器

为了进行模拟信号测量，XC164CM 片上集成一个精度为 10 位、带有 14 路输入通道和采样保持电路的模数转换器（ADC）。该 A/D 转换器采用逐次逼近技术。采样时间（电容器充电）和转换时间可编程设定（两种模式），从而与外部电路相匹配。A/D 转换器还可以工作在 8 位转换模式，此时转换时间进一步减小。

A/D 转换器为转换结果寄存器（ADDAT）提供了过载错误检测/保护：如果前一次转换结果还未从结果寄存器读取，而新转换已经完成，将产生中断请求；或者将新转换暂时挂起，直到前一次的结果被读取。

某些应用所需要的模拟输入通道较少，此时其余通道可用作数字输入口。

XC164CM 的 A/D 转换器支持四种转换模式。标准单通道转换模式下，对选定通道上的模拟电平进行一次采样，并转换为数字结果；单通道连续转换模式下，对选定通道上的模拟电平进行重复采样和转换，而无需软件干预；自动扫描模式下，对预先选定的一组通道进行顺序采样和转换；自动扫描连续转换模式下，对预先选定的一组通道进行重复采样和转换。此外，可将一个特定通道的转换插入正在执行的转换序列中，插入通道转换结束后继续执行序列转换（序列不受干扰），这称为通道插入模式。

外围事件控制器（PEC）可将转换结果自动保存在一个存储表中，而无需每次传送数据时转入和退出中断服务程序，降低了软件开销。

每次复位之后，以及在正常工作期间，ADC 会自动执行校准操作。这种自动自校准会根据工作条件的变化（如温度）来不断调整转换器，并补偿工艺误差。

校准周期为 A/D 转换周期的一部分，因此它不会影响 A/D 转换器的正常操作。完成一次转换之后可禁止自动校准，从而缩短总转换时间。

为了去除模拟输入信号中的数字噪声，并避免输入触发噪声，可将模拟输入引脚从数字 IO 口或输入级断开，该操作由软件控制，通过寄存器 P5DIDIS（P5 口数字输入禁止）分别设置每个引脚。

不进行模数转换时，A/D 转换器的自动掉电特性能够最大程度降低系统功耗。

异步/同步串行接口（ASC0/ASC1）

异步/同步串行接口 **ASC0/ASC1（USART）** 可用于和其它微控制器、处理器、终端或外部外设器件进行串行通信，它们与英飞凌 8 位微控制器系列的串口向上兼容，支持全双工异步通信和半双工同步通信。带有分数分频器的专用波特率发生器可精确产生所有的标准波特率，而无需调整晶振频率。

异步模式下，发送或接收 8 位或 9 位数据帧（奇偶校验位可选），数据帧以一个起始位开始，一个或两个停止位结束。进行多处理器通信时，采用数据字节和地址字节区分机制（8 位数据加唤醒位模式）。ASC0/1 支持高达 115.2 kbit/s 的 IrDA 数据传送，IrDA 脉冲宽度固定或者可编程设定。自动波特率检测单元可检测异步数据帧的波特率及模式，并自动初始化波特率发生器及模式控制位。

同步模式下，字节（8 位）的发送或接收与 ASC0/1 产生的移位时钟同步。

始终先移位 LSB。两种模式下，数据的发送和接收都经 FIFO 缓存（8 级发送 FIFO、8 级接收 FIFO）。可选择回环模式用于测试用途。ASC0/1 提供了五个独立的中断向量，分别用于发送缓存、发送、接收、自动波特率检测和错误处理。

为了提高数据传送的可靠性，模块提供了多种硬件错误检测功能。可在发送数据时自动产生奇偶校验位，或在接收时检查奇偶校验位；帧错误检测可以识别出停止位丢失的数据帧；若新字符已接收完成，而前一字符还未从接收缓存寄存器读出，将产生一个过载错误。

特性概述

- 全双工异步工作模式
 - 8 位或 9 位数据帧，LSB 在先，一个或两个停止位，奇偶校验位产生/检查
 - 波特率范围 2.5 Mbit/s 到 0.6 bit/s（@40MHz）
 - 具有地址/数据字节自动检测功能的多处理器通信模式
 - 支持 IrDA 数据传送，数据率最高可达 115.2 kbit/s（@40MHz）
 - 回环功能
 - 自动波特率检测
- 半双工 8 位同步工作模式，波特率范围 5 Mbit/s 到 406.9 bit/s（@40MHz）
- 发送/接收经 FIFO 缓存（8 级发送 FIFO、8 级接收 FIFO）
- 可选择回环功能用于测试
- 中断产生条件：发送缓存已空、最后一位发送完毕、接收缓存已满、出错情况（帧错误、奇偶校验错误、过载错误）、自动波特率检测开始或停止。

高速同步串行通道（SSC0/SSC1）

高速同步串行通道 SSC0/SSC1 支持全双工和半双工同步通信，它们可与串行连接的外设接口，并完全支持 SPI 功能。

SSC0/SSC1 中的专用波特率发生器可产生所有的标准波特率，而无需调整晶振频率。

SSC 发送或接收 2 到 16 位长度的字符，发送/接收和移位时钟同步，该时钟可由 SSC 自身产生（主模式），也可由外部主机产生（从模式）。SSC 可以先移位最低有效位（LSB）或最高有效位（MSB），移位方向、锁存时钟沿（时钟相位）和时钟极性均可编程设定。

可选择回环模式用于测试用途。

SSC0/SSC1 提供了三个独立的中断向量，分别用于发送、接收和错误处理。

为了提高数据传送的可靠性，模块提供了多种硬件错误检测功能。发送错误和接收错误监控数据缓存的操作；相位错误和波特率错误检测错误的串行数据。

特性概述

- 主模式或从模式操作
- 全双工或半双工传送
- 波特率可变：20 Mbit/s 到 305.18 bit/s（@40MHz）
- 灵活的数据格式
 - 数据位个数可编程：2 至 16 位
 - 移位方向可编程：LSB 或 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
- 可选择回环功能用于测试
- 中断产生条件：发送缓存已空、接收缓存已满、出错情况（接收、相位、波特率，发送错误）
- 三个引脚接口，多种灵活的 SSC 引脚配置

片上 TwinCAN 模块

片上集成的 TwinCAN 模块用于控制 CAN 帧的自动发送和接收，遵循 CAN V2.0B（active）规范。片上 TwinCAN 模块可以发送和接收带有 11 位标识符的标准帧和带有 29 位标识符的扩展帧。

两个全功能的 CAN 节点共享 TwinCAN 模块的资源，目的是优化 CAN 总线通信处理，最大程度减轻 CPU 的工作负荷。该模块提供 32 个报文对象，它们可独立分配给两个 CAN 节点中的任意一个，还可以组合构成 FIFO 结构。每个报文对象分别对应有助于滤波的验收屏蔽寄存器。

全 CAN 功能和 FIFO 结构的灵活组合可满足复杂嵌入式控制应用的实时要求。CAN 总线监控功能的增强以及报文对象个数的增加使得 CAN 的总线通信处理更加精确和方便。

网关功能允许在两个独立的 CAN 总线系统之间自动进行数据交换，这减轻了 CPU 负荷，提高了整个系统的实时性。

两个 CAN 节点的位定时都从主时钟得到，可通过编程使数据速率达到 1Mbit/s。每个 CAN 节点有一对接收和发送引脚与外部总线收发器连接。

特性概述：

- CAN 功能符合 CAN V2.0 B active 规范
- 支持的数据速率高达 1Mbit/s
- 具有灵活、功能强大的报文传送控制和错误处理能力
- 每个报文对象具有全 CAN 功能和基本 CAN 功能
- 32 个灵活的报文对象
 - 分配给两个 CAN 节点之一
 - 设置为发送或接收对象
 - 构成 2、4、8、16 或 32 级 FIFO 报文缓存
 - 设置为处理带有 11 位或 29 位标识符的帧
 - 提供用于滤波的可编程验收屏蔽寄存器
 - 通过帧计数器进行监控
 - 可设置为远程监控模式
- 多达 8 个独立的可编程中断节点
- 实现了用于总线监控的 CAN 分析模式

看门狗定时器

看门狗定时器提供了一种故障保险机制，用来避免系统长时间处于故障状态。

看门狗定时器在芯片复位后始终被使能；兼容模式下，在执行 **EINIT** 指令之前，看门狗定时器可被禁止；增强模式下，看门狗定时器在任何时候都可通过指令 **DISWDT** 和 **ENWDT** 禁止和使能。因此，芯片启动进程始终受看门狗定时器监控。看门狗定时器在溢出之前必须由软件刷新。如果硬件或软件发生错误，软件不能及时刷新，看门狗定时器将溢出，产生内部硬件复位，并将 **RSTOUT** 引脚拉低，从而复位外部的硬件电路。

看门狗定时器是一个 16 位定时器，输入时钟基于系统时钟经 2/4/128/256 分频处理。看门狗定时器寄存器的高位字节可设置为预定义的重载值（保存在 **WDTREL** 中），用于调整监控的时间间隔。应用程序每次服务看门狗定时器之后，看门狗定时器的高位字节将被重载，低位字节被清零。看门狗定时器可监控 13 μ s 至 419 ms 的时间间隔（@40 MHz）。

看门狗定时器复位后的缺省时间间隔为 3.28ms（@40 MHz）。

并口

XC164CM 可提供多达 47 条 I/O 线，组成三个输入/输出口和一个输入口。所有端口线均可位寻址，通过方向寄存器分别被设置为输入或输出功能。这些 I/O 口为真正的双向口，用作输入时切换到高阻态；用作输出时，可通过控制寄存器将其设置为推挽输出或漏极开路输出。内部复位时，所有引脚配置为输入。

端口驱动的边沿特性（形状）和驱动特性（输出电流）可通过寄存器 POCONx 设置。

某些端口的输入阈值可选（TTL 或类 CMOS），特殊的类 COMS 输入阈值可利用电压滞环特性抑制噪声。可以分别为各端口的每个字节选择输入阈值。

所有端口线均可选择用作复用输入或输出功能。未用作复用功能的端口可作为通用 IO 口使用。

表 2-3 XC164CM 并口概述

端口	控制	复用功能
P1 口	Pad 驱动	捕获输入或比较输出， 串行接口线， 快速外部中断输入
P3 口	Pad 驱动， 漏极开路， 输入阈值	定时器控制信号，串行接口线， 系统时钟输出 CLKOUT（或 FOUT）
P5 口	-	A/D 转换器的模拟输入通道， 定时器控制信号
P9 口	Pad 驱动， 漏极开路， 输入阈值	捕获输入或比较输出
		CAN 接口线 ¹⁾

1) 可由软件分配。

2.4 时钟产生

时钟产生单元通过可编程片上 PLL 和多种预分频因子，非常灵活的为 XC164CM 提供时钟信号。主时钟 f_{MC} 是参考时钟信号，用作 TwinCAN 的输入时钟、并输出给外部系统。CPU 时钟 f_{CPU} 和系统时钟 f_{SYS} 可以和主时钟相同（1:1）、也可以由主时钟 2 分频（ $f_{SYS} = f_{CPU} = f_{MC}/2$ ）。

片上振荡器可驱动外部晶振或者接收外部时钟信号。振荡器时钟频率可经片上 PLL 倍频（倍频因子可编程设定），或经预分频器分频（分频因子可编程设定）。

旁路模式下（直接驱动或预分频），PLL 可提供一个独立的时钟，以监控片上振荡器产生的时钟信号。该 PLL 时钟和 XTAL1 时钟无关。若未出现期望的振荡器时钟跳变，振荡器看门狗（OWD）会激活 PLL 解锁/OWD 中断节点，并通过 PLL 时钟信号为 CPU 提供紧急时钟。在这种情况下 PLL 以基频工作。

关闭 PLL **可禁止振荡器看门狗**。该操作降低了功耗，但在振荡器时钟丢失的情况下不会产生中断请求。

2.5 功率管理特性

附加功率管理特性（描述如下）扩展了基本省电模式（空闲和掉电）。将这些特性相结合，可以最大程度降低控制器的功耗，从而满足应用的低功耗需求。

- 基本省电模式
- 灵活的时钟产生方式
- 灵活的外设管理（各外设可分别被禁止和使能）
- 通过 RTC 定时器将 CPU 从空闲模式下周期唤醒

以上特性给系统提供了极大的灵活性，在节省功耗（待机时间）和外设操作（系统功能）之间取得很好的平衡。

基本省电模式

XC164CM 可切换到特殊的工作模式（由指令控制），这些模式下系统功耗降低（功能减少）。

- **空闲模式**下 CPU 停止运行，而外设继续工作。
- **休眠模式**和**掉电模式**下所有的时钟信号关闭、所有操作停止运行（RTC 可选择继续运行）。可以通过外部中断信号终止休眠模式。

灵活的时钟产生

灵活的时钟产生系统结合了一系列改进机制（部分用户可控），为 XC164CM 模块提供时钟信号。对于功耗要求严格的工作模式（如待机操作），灵活产生时钟尤其重要。

功率优化的振荡器可以降低产生 XC164CM 时钟信号的功耗。

时钟系统控制内部和外部时钟信号的分配和频率。用户可降低 XC164CM CPU 时钟频率，从而大大降低系统功耗。

外部电路可由频率可编程的时钟输出信号 FOUT 控制。

灵活的外设管理

灵活的外设管理提供了一种机制，可分别使能和禁止各个外设模块。每种省电模式下（如不同的系统工作模式、待机状态等），只有所需模块保持运行。例如，保持通信接口模块运行时，其它外设模块可被关闭，不过，禁用外设的寄存器仍可被访问。

周期唤醒空闲或休眠模式

周期唤醒空闲或休眠模式，不但能够大大降低系统功耗（和附加的功率管理特性相结合），还可以保证系统高度可用。CPU 和选定的外设被周期性激活，扫描外部信号和

事件（扫描频率较低），扫描结束后 CPU 和选定的外设又返回省电模式。这大大降低了系统的平均功耗。空闲/休眠模式还可以由外部中断信号唤醒。

2.6 片上调试支持（OCDS）

片上调试支持系统为用户提供了各种内嵌的调试和仿真功能。在XC164CM上运行的用户软件可以方便的在目标系统中进行调试。

OCDS通过调试接口由外部调试设备进行控制，调试接口由遵循IEEE-1149标准的JTAG接口和断点接口组成。调试器通过一组专用寄存器来控制OCDS，这些寄存器可由JTAG接口访问。此外，OCDS系统还可以由CPU控制（如监控程序）。插入接口允许CPU执行由OCDS产生的指令。

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

观察点的数据传送可以通过 JTAG 接口实现。

调试接口通过 6 个接口信号（4 条 JTAG 线，2 条中断线）与外部电路通信。这些接口信号使用 P3 口的复用功能。

2.7 保护位

XC164CM 提供了一种特殊机制，保护那些可由片上硬件修改的数据位不被软件意外修改（见[章节 4.8.2](#)）。被保护位归纳如下：

表 2-4 XC164CM 的保护位

寄存器	位名称	备注
GPT12E_T3CON	T3OTL	GPT1 定时器输出翻转锁存
GPT12E_T6CON	T6OTL	GPT2 定时器输出翻转锁存
ASC0_CON	REN	ASC0 接收使能标志
ASC1_CON	REN	ASC1 接收使能标志
SSC0_CON	BSY	SSC0 忙碌标志
SSC0_CON	BE, PE, RE, TE	SSC0 错误标志
SSC1_CON	BSY	SSC1 忙碌标志
SSC1_CON	BE, PE, RE, TE	SSC1 错误标志
ADC_CON/ ADC_CTR0	ADST, ADCRQ	ADC 启动标志/插入请求标志
TFR	TFR.15, 14,13, 12	A 类强制中断标志
TFR	TFR.7, 4, 3, 2	B 类强制中断标志
PECISNC	C7IR ... C0IR	所有通道中断请求标志
CC2_SEE	SEE.15 ... SEE.0	单次事件使能位
CC2_OUT	CC15IO ... CC0IO	比较输出位
P1L	P1L.7	用于 CAPCOM2 功能的 P1 口的相关位
P1H	P1H.5- 4, P1H.0	用于 CAPCOM2 功能的 P1 口的相关位
P9	P9.5 ... P9.0	用于 CAPCOM2 功能的 P9 口的所有位
RTC_ISNC	T14IR, CNT3IR ... CNT0IR	中断节点共享请求标志
CC2_CC31-16IC	CC31IR ... CC16IR	CAPCOM2 中断请求标志

寄存器	位名称	备注
CC2_T8-7IC	T7IR, T8IR	CAPCOM2 定时器中断请求标志
CCU6_IC	CIR	CAPCOM6 中断请求标志
CCU6_EIC	EIR	CAPCOM6 错误中断请求标志
CCU6_T12IC	T12IR	CAPCOM6 定时器 T12 中断请求标志
CCU6_T13IC	T13IR	CAPCOM6 定时器 T13 中断请求标志
GPT12E_T6-2IC	T6IR ... T2IR	GPT 定时器中断请求标志
GPT12E_CRIC	CRIR	GPT2 CAPREL 中断请求标志
ADC_CIC	ADCIR	ADC 转换结束中断请求标志
ADC_EIC	ADEIR	ADC 过载中断请求标志
ASC0_T(B)IC	TIR, TBIR	ASC0 发送（缓存）中断请求标志
ASC0_RIC, ASC0_EIC	RIR, EIR	ASC0 接收/错误中断请求标志
ASC0_ABIC	ABIR	ASC0 自动波特率检测中断请求标志
ASC1_T(B)IC	TIR, TBIR	ASC1 发送（缓存）中断请求标志
ASC1_RIC, ASC1_EIC	RIR, EIR	ASC1 接收/错误中断请求标志
ASC1_ABIC	ABIR	ASC1 自动波特率检测中断请求标志
SSC0_TIC, SSC0_RIC	TIR, RIR	SSC0 发送/接收中断请求标志
SSC0_EIC	EIR	SSC0 错误中断请求标志
SSC1_TIC, SSC1_RIC	TIR, RIR	SSC1 发送/接收中断请求标志
SSC1_EIC	EIR	SSC1 错误中断请求标志
PLLIC	PLLIR	PLL/OWD 中断请求标志
EOPIC	EOPIR	PEC 结束中断请求标志
CAN_7IC, CAN_0IC	CAN7IR ... CAN0IR	TwinCAN 中断请求标志

寄存器	位名称	备注
RTC_IC	RTCIR	RTC 中断请求标志
-----	XX16IR ... XX0IR	“未指定节点”中断请求标志

3 存储器结构

XC164CM 的存储器空间按照冯诺伊曼体系结构组织，这意味着代码和数据的访问使用同一个线性地址空间。所有物理上独立的存储区，包括内部 Flash、内部 RAM、内部特殊功能寄存器区（SFR 和 ESFR）、内部 IO 区和外部存储区（只包含由内部 LxBus 访问的片上 TwinCAN 模块），都被映射到一个共同的地址空间。

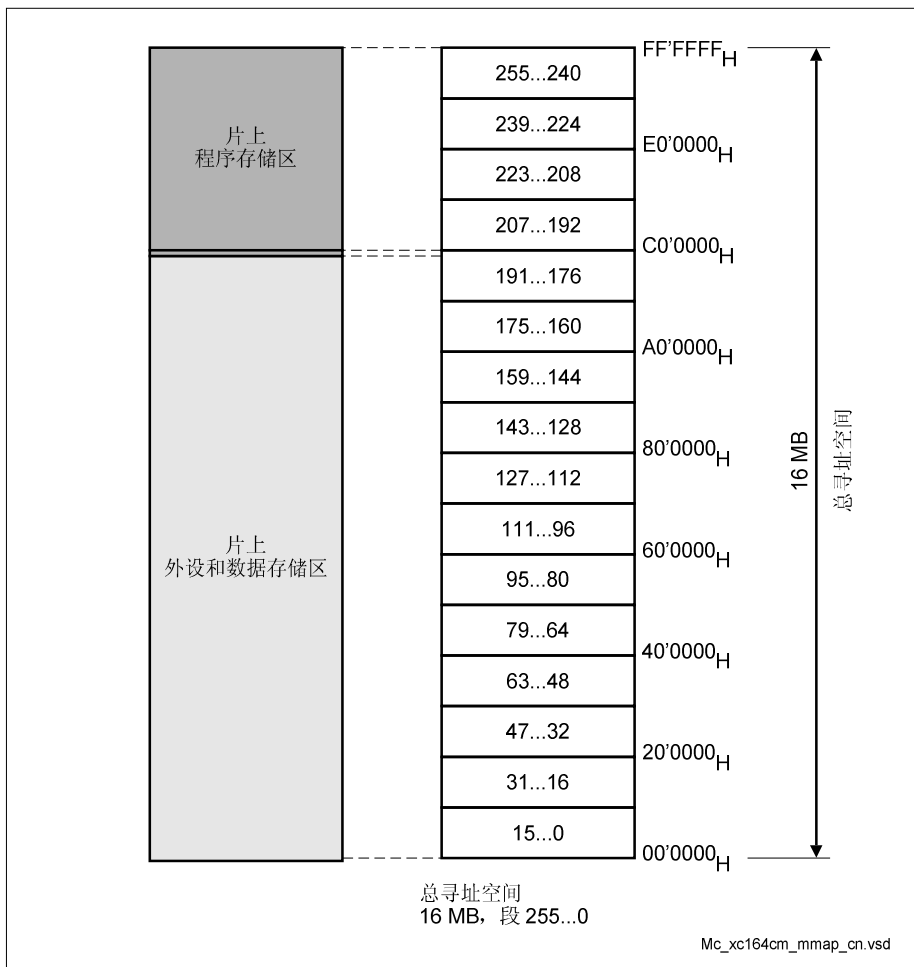


图 3-1 地址空间总览

XC164CM 的可寻址存储器空间为 16MB，分为 256 段，每段 64KB。每段又细分为 4 个数据页，每页 16KB（如图 3-1）。

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

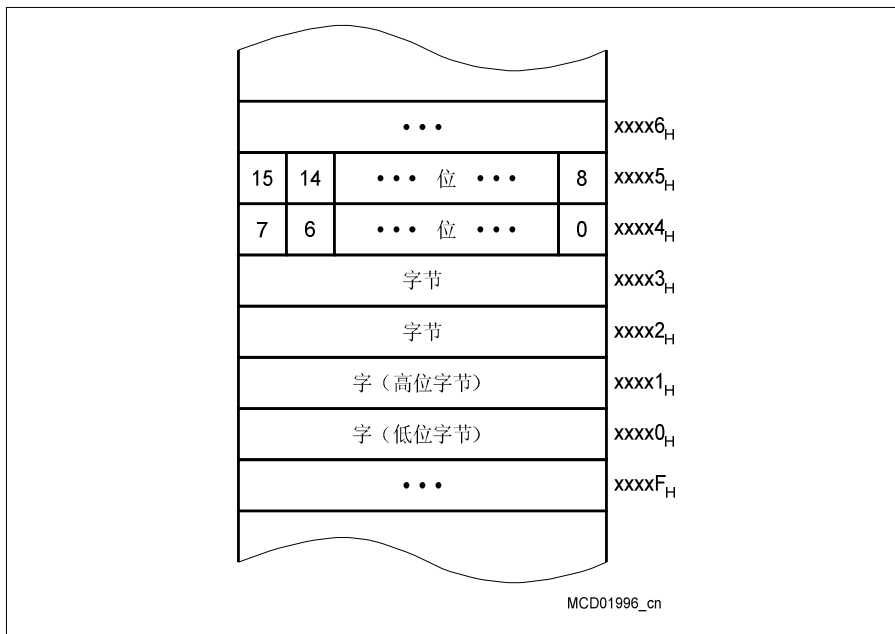


图 3-2 字、字节和位的存储（以字节为单位组织存储器）

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

3.1 地址映射

不同的存储区以及外设寄存器（见**表 3-1**）全部映射到一个连续的地址空间上。每一个区段都可采用相同的方式进行访问。**XC164CM** 的存储映射空间中有一部分保留区域，从而能够和后续的增强型衍生产品向上兼容。

表 3-1 XC164CM 存储器映射

地址区间	起始地址	结束地址	区域大小 ¹⁾	备注
Flash 寄存器空间	FF'F000 _H	FF'FFFF _H	4KB	2)
保留 (Acc.trap)	F8'0000 _H	FF'EFFF _H	<0.5MB	减去 Flash 寄存器
为 PSRAM 保留	E0'0800 _H	F7'FFFF _H	<1.5MB	减去 PSRAM
程序 SRAM	E0'0000 _H	E0'07FF _H	2KB	最大 ³⁾
为程序存储保留	C1'0000 _H	DF'FFFF _H	<2MB	减去 Flash
程序 Flash	C0'0000 _H	C0'FFFF _H	64KB	4)
保留	40'0000 _H	BF'FFFF _H	8MB	—
保留	20'0800 _H	3F'FFFF _H	<2MB	减去 TwinCAN
TwinCAN 寄存器	20'0000 _H	20'07FF _H	2KB	通过 EBC 访问
保留	01'0000 _H	1F'FFFF _H	<2MB	减去段 0
SFR 区	00'FE00 _H	00'FFFF _H	0.5KB	—
双口 RAM	00'F600 _H	00'FDFF _H	2KB	—
为 DPRAM 保留	00'F200 _H	00'F5FF _H	1KB	—
ESFR 区	00'F000 _H	00'F1FF _H	0.5KB	—
XSFR 区	00'E000 _H	00'EFFF _H	4KB	—
保留	00'C800 _H	00'DFFF _H	6KB	—
数据 SRAM ⁴⁾	00'C000 _H	00'C7FF _H	2KB	仅 XC164CM-8F 中包含 DSRAM
为 DSRAM 保留	00'8000 _H	00'BFFF _H	16KB	—
保留	00'0000 _H	00'7FFF _H	32KB	—

- 1) 标有“<”的区域大小略小于标注值，见“备注”列。
- 2) 访问未定义的寄存器地址将返回强制中断代码（1E9B_H）。
- 3) 这是本手册描述的单片机系列所能实现的最大存储空间。
- 4) 不同衍生产品的存储器容量不同，在**第 2 页“关于本手册”**中列出衍生器件的型号及相应参数信息。

3.2 特殊功能寄存器区

特殊功能寄存器（SFR）控制 XC164CM 的系统和外设功能。可通过以下三个专用地址区域访问特殊功能寄存器：

- 512 B SFR 区（位于内部 RAM 上方：00'FFFF_H...00'FE00_H）
- 512 B ESFR 区（位于内部 RAM 下方：00'F1FF_H...00'F000_H）
- 4KB XSFR 区（位于 ESFR 区下方：00'EFFH...00'E000_H）

这种地址组织方式保证了与 C166 系列产品向上兼容。

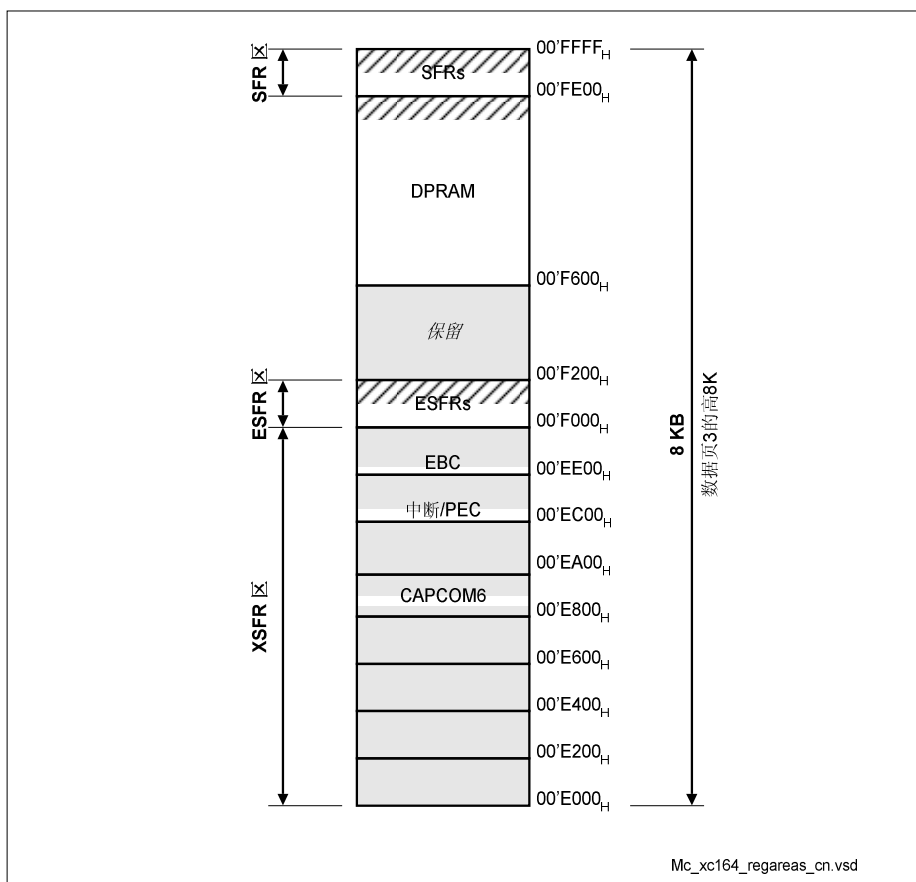


图 3-3 特殊功能寄存器（SFR）映射

注：SFR 区、ESFR 区以及内部 RAM 的高 256 字节均可位寻址（见图 3-3 中用斜杠标注的区域）。

特殊功能寄存器

XC164CM 的 CPU、总线接口、IO 端口、及片上外设功能均通过特殊功能寄存器（SFR）来控制。

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

注：对 SFR 中的一个字节进行写操作时，另外一个未被寻址的字节将被清零。

SFR 区的上半部分（地址分别为 00'FFFF_H...00'FF00_H）和 ESFR 区的上半部分（00'F1FF_H...00'F100_H）寄存器可位寻址，因此通过位寻址可直接修改或检查相应的控制/状态位。

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

ESFR_SWITCH_EXAMPLE:

EXTR #4	； 切换到 ESFR 区执行以下 4 条指令
MOV ODP9, #data16	； ODP9 采用 8 位寄存器寻址
BFLDL DP9, #mask, #data8	； 对位域的位寻址
BSET DP1H.7	； 对位元的位寻址
MOV T8REL, R1	； T8REL 采用 16 位存储器寻址
	； R1 被复制到 ESFR 空间
	； （该访问不需要执行 EXTR）
；-----；	； EXTR #4 指令有效范围至此结束
MOV T8REL, R1	； T8REL 采用 16 位存储器寻址
	； R1 通过 SFR 区访问

为了尽量减少使用 EXTR 指令，ESFR 区中主要存放控制初始化以及模式选择的寄存器。尽可能地将需要被频繁访问的寄存器存放在标准 SFR 区。

注：XC164CM 有专用工具监控 ESFR 区的访问。访问 ESFR 区时会自动插入 EXTR 指令；若 EXTR 指令缺失或是过多会发出警告。

访问 XSFR 区寄存器采用 16 位寻址，不需特别的寻址模式或预防措施。

通用寄存器

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

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

表 3-2 通用寄存器映射到 DPRAM

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

XC164CM 支持快速寄存器组（上下文）切换。DPRAM 中可以同时存在多个全局寄存器组。不过，在给定时刻，只有被上下文指针寄存器（CP）选中的那个全局寄存器组有效。重新选择有效全局寄存器组时，只需更新寄存器 CP 即可。可用一个特殊的上下文切换指令（SCXT）执行寄存器组切换，该指令会自动保存当前 CP，并载入新 CP。所能实现的寄存器组（任意大小）的个数只受限于可用 DPRAM 的容量。

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

PEC 源指针和目的指针

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

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

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

若某个 PEC 通道未被使用，相应的指针单元可用作其它用途。

用于 PEC 数据传送的源指针和目的指针的具体内容，请参阅[章节 5.4](#)。

注：对 PEC 指针中的一个字节进行写操作时，该指针中另外一个未被寻址的字节将被清零。

3.3 数据存储区

XC164CM 中有两个片上 RAM 区，用于数据存储：

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

注：数据也可以存储在 PSRAM 中（见[章节 3.4](#)）。不过，访问数据存储区速度最快。

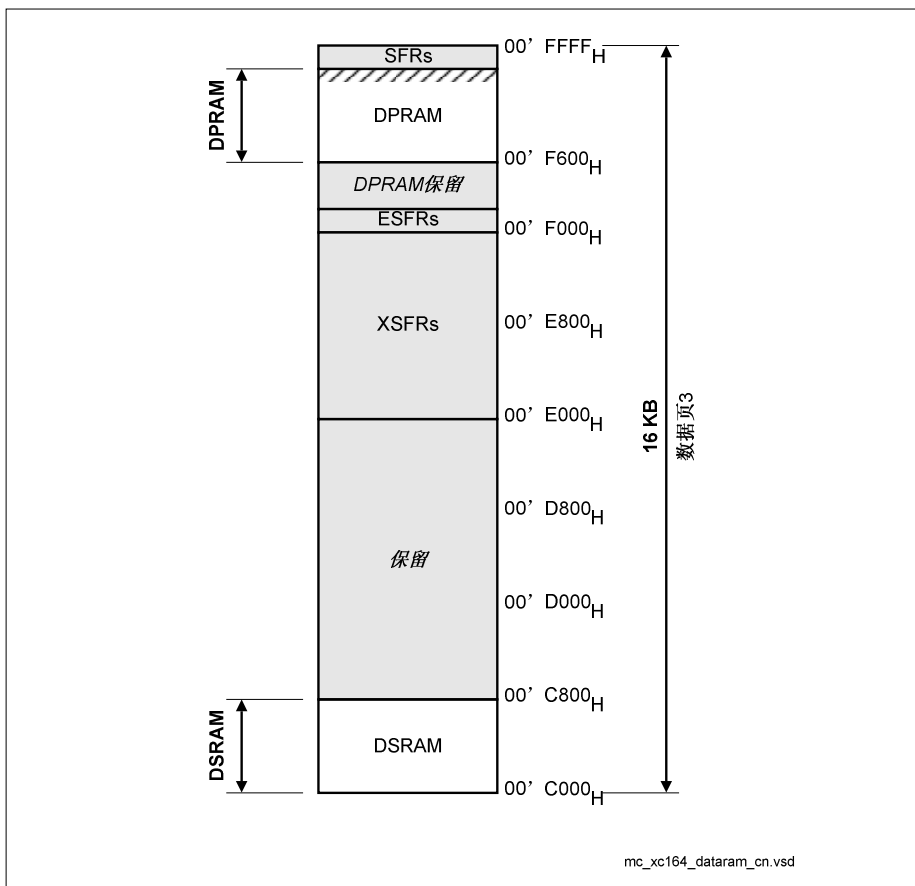


图 3-4 片上数据 RAM 映射

注：不同衍生产品所包含的 DSRAM 容量不同。在**第 2 页“关于本手册”**中列出衍生器件的型号及相应参数信息。

双端口 RAM (DPRAM)

XC164CM 片上集成有 2KB 的 DPRAM (00'F600_H...00'FDFF_H)。若 DPP 寄存器指向数据页 3，DPRAM 中的任意字或字节可通过间接寻址或 16 位长寻址模式访问。对任意字的访问通过寻址偶字节地址实现。DPRAM 中，字数据的最高存储地址为 00'FDFE_H。

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

DPRAM 的高 256B (00'FD00_H...00'FDFF_H) 可位寻址（见**图 3-4** 用斜杠标注的区域）。

注：代码不能在 DPRAM 内执行。

DPRAM 占用 3KB 的专用区域 (00'F200_H...00'FDFF_H)，其中有 1KB 的 DPRAM 区保留待用。

数据 SRAM (DSRAM)¹⁾

XC164CM 片上集成有 2KB 的 DSRAM (00'C000_H...00'C7FF_H)。若 DPP 寄存器指向数据页 3，DSRAM 中的任意字或字节可通过间接寻址或 16 位长寻址模式访问。对任意字的访问通过寻址偶字节地址实现。DSRAM 中，字数据的最高存储地址为 00'C7FE_H。

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

注：代码不能在 DSRAM 内执行。

DSRAM 占用 20KB 的专用区域 (00'8000_H...00'CFFF_H)，目前未使用的 DSRAM 区保留待用。

1) 不同衍生产品所包含的 DSRAM 容量不同，在**第 2 页“关于本手册”**中列出衍生器件的型号及相应参数信息。

3.4 程序存储区

XC164CM 中有两个片上程序存储区，用于代码/ 数据存储：

- **程序 Flash** 存储代码和常量数据。Flash 存储器可通过应用软件（反复）编程。
- **程序 SRAM（PSRAM）** 存储临时代码和其它数据。例如，高级引导程序可以写在 PSRAM 中，继而执行该程序对片上 Flash 存储器进行编程。

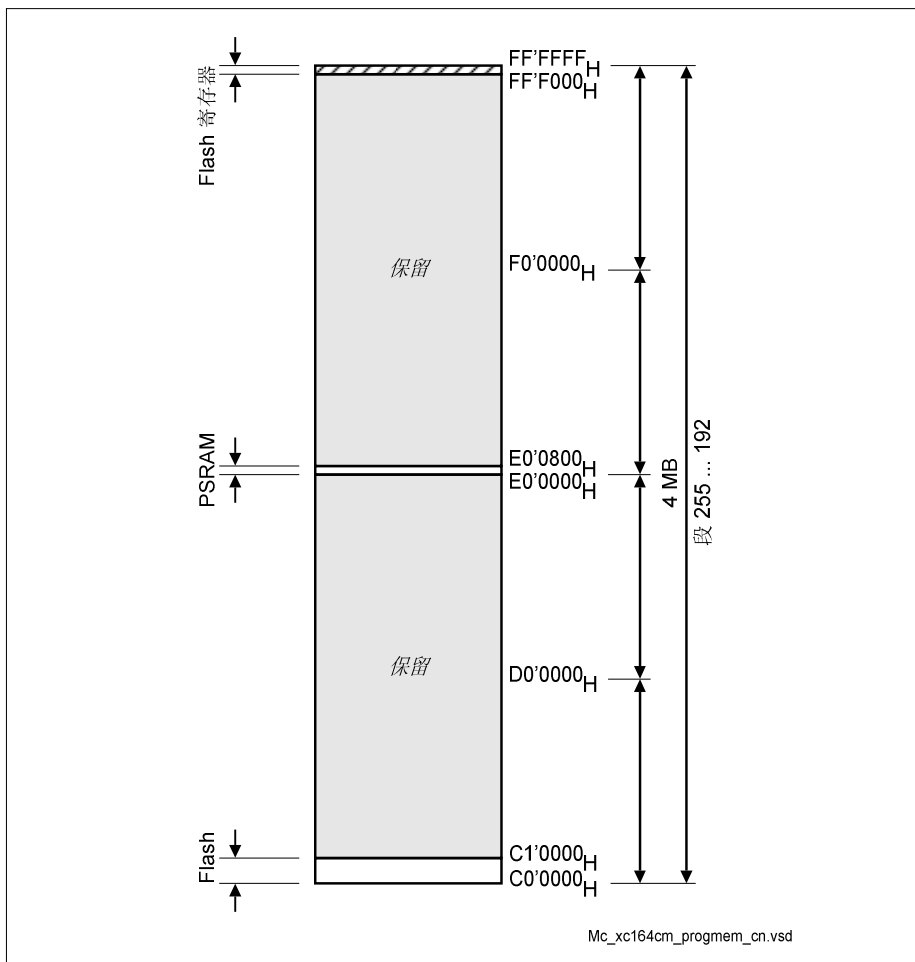


图 3-5 片上程序存储器映射

程序/数据 SRAM（PSRAM）

XC164CM 片上集成有 2KB 的 PSRAM（E0'0000_H...E0'07FF_H）。PSRAM 能够快速执行代码，无起始延时。因此，PSRAM 支持非顺序代码的执行（如通过中断向量表产生非顺序代码）。

若 DPP 寄存器指向数据页 896，PSRAM 中的任意字或字节可通过间接寻址或 16 位长寻址模式访问。对任意字的访问通过寻址偶字节地址实现。PSRAM 中，字数据的最高存储地址为 E0'07FE_H。

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

任何数据都可以保存在 PSRAM 中。不过，由于 PSRAM 是为取指而优化设计的，因此，通过数据存储器存取数据性能更好。

注：PSRAM 不可位寻址。

PSRAM 占用 1.5MB 的专用区域（E0'0000_H...F7'FFFF_H），目前未使用的 PSRAM 区保留待用。

非易失程序存储器（Flash）

XC164CM 片上集成有 64KB¹⁾的程序 Flash（C0'0000_H...C0'FFFF_H）。通过字节选择线选中字和字节，一次始终读取 64 位的代码和数据。若选用的 DPP 寄存器指向对应数据页时，程序存储器中的任意字或字节可通过间接寻址或 16 位长寻址模式访问。对任意字的访问通过寻址偶字节地址实现。程序存储器中，字数据的最高存储地址为 C0'FFFE_H¹⁾。

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

注：该程序存储器不可位寻址。

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

1) 不同衍生产品所包含的程序存储器容量不同，在**第 2 页“关于本手册”**中列出衍生器件的型号及相应参数信息。

3.5 系统堆栈

系统堆栈可位于 XC164CM 存储空间中的任意地址单元。

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

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

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

堆栈指针总是指向最新的系统堆栈入口，而非下一个可用的堆栈地址。

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

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

3.6 IO 区

XC164CM 地址空间中的下列区域属于 IO 区：

- **内部 IO 区** 用于访问片内外设。内部 IO 区分为三个区：
 - SFR 区，地址范围：00'FE00_H...00'FFFF_H（512B）
 - ESFR 区，地址范围：00'F000_H...00'F1FF_H（512B）
 - XSFR 区，地址范围：00'E000_H...00'EFFF_H（4KB）

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

由于控制外设模块和控制存储器的方式不同，因此 IO 区具有一些特殊属性：

- IO 区访问不经过缓存。保存 IO 区的读写结果时不使用回写缓存。
- 不执行不确定的读操作。直到所有不确定因素都确定后才能执行读操作（如条件分支后的预读取操作）
- 禁止数据前送。由于在执行 IO 写操作后，外设的内部状态会发生改变，因此，直到流水线中所有挂起的 IO 写操作完成之后，才能执行 IO 读操作。

3.7 存储器边界越界

XC164CM 的地址空间隐含地划分为大小相同的存储区块（区块内实际使用的存储区域大小不同）和不同的逻辑存储区。跨越这些代码或数据区的边界时，需要特别加以注意，以确保控制器能够执行所要的操作。

地址空间被划分为多种不同的**存储区**，包括 SFR 区、片上程序或数据 RAM 区、片上 Flash、片上 LXBUS 外设（若有集成），以及外部存储区（XC164CM 中无外部存储区）。

访问连续存放在不同存储区的**数据**时不会出现问题。但在执行存放于不同存储区的**代码**时，必须通过分支指令明确地进行存储区切换，系统不支持代码越界连续访问，这样会导致错误的结果。

注：从外部存储区切换到片上 RAM 区发生在段 0 内。

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

取指时，代码段不会自动切换、必须使用指令进行段切换。使用 JMPS、CALLS、RETS 指令可实现代码段的切换。

在较大的顺序执行的程序中，务必在一段的最高代码存储单元中存放无条件分支指令，控制程序跳转到下一个相关段，以避免预取指令时离开当前段。

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

3.8 片上程序 Flash 模块

XC164CM 片上集成有 64KB¹⁾ 的嵌入式 Flash 存储器（起始地址为 C0'0000_H，见 [图 3-5](#)），用于存储代码或常量。Flash 工作电压 5V，无需额外的编程电压。片内电压发生器需要大约 250μs 的电压稳定时间。Flash 阵列由 5 个扇区：4 个 8KB 扇区和 1 个 32KB 扇区组成¹⁾。Flash 不仅读取速度很快，而且编程、擦除算法简单、同时具有写保护功能。64 位的取指操作，在一个访问周期内读取两个双字指令（或四个单字指令），实现了 CPU 的最大性能。

通过纠错码能够动态纠正 Flash 的单个位错误，从而提高了数据的完整性。此外，通过特殊的裕量检测机制可检测并修正有问题的数据位，从而避免它们引发系统故障。

所有的 Flash 操作由命令序列（参考 JEDEC 单电源 Flash 标准）控制。编程和擦除算法由内部 Flash 状态机控制自动执行，从而以较低的软件开销避免了 Flash 的内容被无意破坏。命令序列由连续的写（或读）操作组成，这些操作访问 Flash 或 Flash 寄存器中的虚拟单元。Flash 的虚拟单元具有特定的地址（见命令序列表）。

为使编程效率最优，页面模式（突发脉冲模式）先将 128 个字节快速装入页面缓存中（快速 CPU 操作），然后通过一条存储指令（耗时典型值 2ms²⁾），将缓存中的内容编程到 Flash 中。每个扇区可被单独擦除（耗时典型值 200ms²⁾）。

注：和标准的 EPROM 相反，Flash 存储单元被擦除后全部为“0”。

Flash 模块提供了两种数据/代码保护机制：整个 Flash 阵列的一般读/写保护，和特定扇区³⁾ 写保护。使用密码检测序列可暂时禁止这些硬件保护特性。密码检测序列的加密信息及关键字存储在一个单独的安全扇区内，和用户代码/数据分开（见 [章节 3.8.4](#)）。

专用 Flash 状态寄存器返回 Flash 的全局和特定扇区的状态信息。在任意时刻，都可通过 Flash 状态寄存器查询某操作是否正确执行、以及 Flash 模块的当前状态。

Flash 模块的物理地址范围为 0'0000_H...0'FFFF_H¹⁾，被映射到起始地址为 C0'0000_H 的程序存储器区；单独的安全扇区也被映射到该区域。采用特殊的保护指令避免了访问冲突。

通过自动编程/擦除算法以及页面缓存填充机制实现了在系统编程（编程/擦除 Flash 时，执行存放在 Flash 中的编程子程序，填充页面缓存）。执行编程/擦除操作时，Flash 读访问被禁止。此外，被彻底擦除的 Flash 模块也能够系统在编程。内嵌的引导程序加载器可通过串行接口加载初始化编程子程序，它进而控制对 Flash 模块进行编程。在编程空白 Flash、以及重新编程过程出现问题（如电源故障）、又没有安全程序时，初始化编程非常有用。

1) 不同衍生产品所包含的程序存储器容量不同，在 [第 2 页“关于本手册”](#) 中列出衍生器件的型号及相应参数信息。

2) 具体参数请参阅数据手册。

3) 两个 8KB 的扇区组合成一个可锁定的 16KB 存储区进行写保护。

注：引导模式下，绝对禁止访问受保护的 Flash。在进行任何编程/擦除操作之前，必须采用正确的密码序列暂时禁止保护机制。

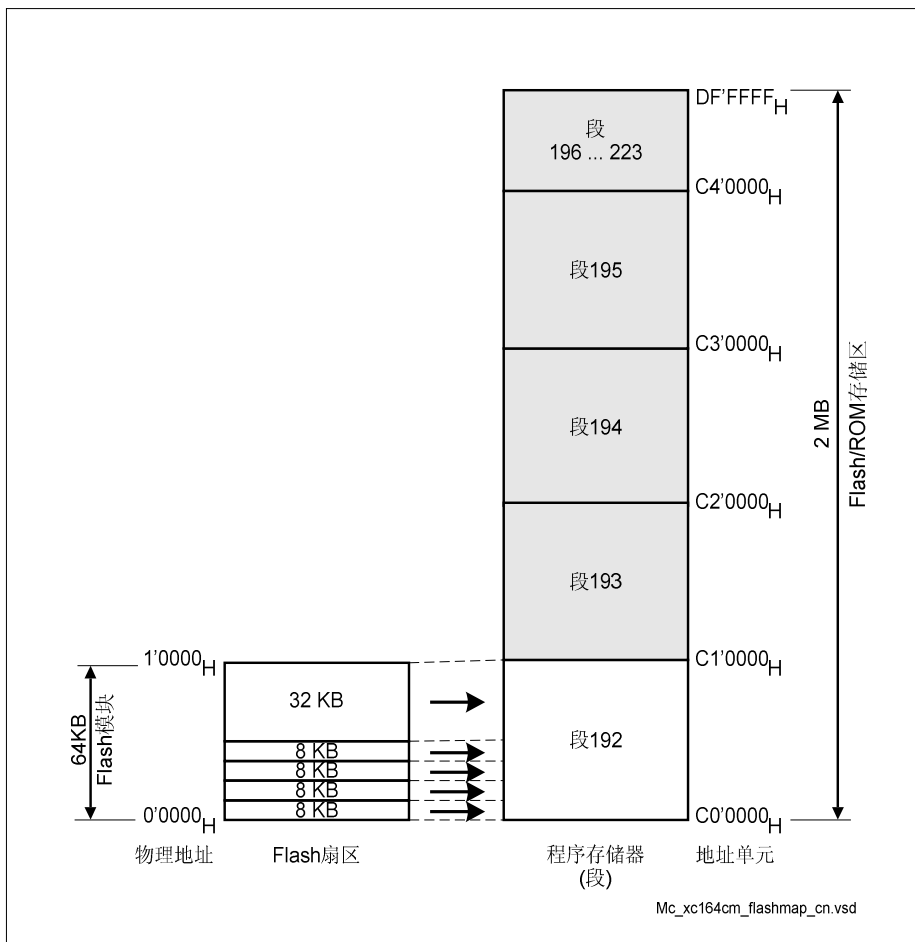


图 3-6 片上 Flash 扇区映射

注：程序存储器的段 192 映射到段 193。

3.8.1 Flash 工作模式

片上 Flash 模块有两种基本的工作模式：

- **标准读模式：**从 Flash 模块中读取代码和数据
- **命令模式：**Flash 模块执行先前定义的命令

标准读模式

标准读模式（正常工作模式）下，Flash 存储器和标准 ROM 一样，可通过任意寻址模式读取 Flash 中的代码和数据。

下列情况下，**Flash 进入标准读模式：**

- 解除系统复位信号之后（电源稳定之后）
- 复位命令执行之后，若没有被激活的编程或擦除操作
- 每个完整的命令（编程、擦除等）执行之后
- 检测到命令序列出错时
- 检测到保护冲突时（对受保护的扇区进行编程或擦除）

注：标准读模式由忙碌状态位 $BUSY = 0$ 指示。

当命令序列的最后一条命令被解码、开始执行 Flash 阵列编程或擦除操作时，**Flash 退出标准读模式**。因此，命令序列中最后一条命令之前的所有命令（特别是加载页面缓存）都可由本身存放在 Flash 模块中的代码执行。

每次对 Flash 进行读操作会执行自动检错，两位错误可被检测并指示、但不可自动修正；单位错误可被检测、指示、并自动修正（见[章节 3.8.3](#)）。

注：裕量检测操作可定位并避免单位错误。

命令模式

除标准读操作之外，所有的 Flash 操作都由命令序列启动，命令序列写入（虚拟）Flash 命令寄存器（位于 Flash 地址空间）。被保护命令还需要四个密码字进行验证。

命令序列中的最后一条命令写入命令寄存器之后，**Flash 进入命令模式**。所有激活 Flash 阵列操作（例如擦除扇区）的命令序列，其执行过程会占用一段时间，因此命令模式将在规定的一段时间内保持有效。当 Flash 处于命令模式（忙碌标志置位）时，对 Flash 阵列的读访问会延迟到 Flash 返回标准读模式之后才执行。

注：命令模式由忙碌状态位 $BUSY = 1$ 指示。

正确执行命令序列之后、或状态寄存器指示有错误发生时，**Flash 退出命令模式**。

不启动 Flash 操作的命令序列（如进入页面模式）会立刻执行，不进入命令模式。

3.8.2 命令序列

除正常的读操作之外，所有的操作都由写入 Flash 状态机的命令序列启动和控制。命令序列中不同的写周期不仅定义了所需命令，还建立了故障保险机制，以保护 Flash 的内容不被无意破坏。出于性能方面的考虑，不直接控制 Flash 阵列操作的命令均为单周期命令；控制 Flash 阵列操作的命令需要多个周期，保护控制命令需要校验 64 位安全校验码（四个密码字）。命令周期可以不连续（允许暂停）。

命令序列和取指操作可同时执行，因此，只要 Flash 模块处于读模式、没在执行编程或擦除操作，命令序列指令也可从片上 Flash 中执行。若用于查询状态寄存器的命令序列保存在 Flash 之外的存储器中，则可在任何状态下（包括在编程、擦除期间）查询状态寄存器的值，否则取指将停滞。

写入错误的地址和数据，或者写入地址和数据的序列错误，期望的操作将被中止，复位 Flash 模块进入读模式，置位状态寄存器中的序列错误标志。

读取状态的命令寻址分离的 Flash 寄存器空间，无需命令序列。寄存器写操作只需一个命令周期执行。

编程操作：CPU 以最快速度装载 128 字节的页面缓存，然后通过命令序列，将缓存中的内容编程到 Flash 中。编程过程分为三步：

- 用进入页面模式命令初始化页面缓存（还定义了目标页面地址）。
- 用连续页面装载命令装载页面缓存（页面缓存的偏移地址自动递增）。
- 通过写页面指令将整个缓存的内容编程到 Flash 中。

擦除操作：可清除选定扇区中、或 256 字节字线上的所有数据位。擦除命令序列包含了目标扇区或字线的地址。

当请求编程/擦除操作后，相应操作会自动执行，无需额外的用户控制。操作状态和操作终止信息均由状态标志位指示。掉电请求只有在编程/擦除操作结束后才有效。电源稳定后的系统复位将中止编程/擦除操作，此时 Flash 状态寄存器将显示操作错误 (OPER)。

以下三个列表归纳了各种指令序列，分别用于：

- 组织 Flash 访问（表 3-3）
- 编程与擦除（表 3-4）
- 保护控制（表 3-5）

注：每个命令序列列出所需的地址 (A=...) 和数据 (D=...)。

组织命令

表 3-3 命令序列定义（组织访问）

周期	复位到读模式	状态清零	读 Flash 状态或裕量	写裕量
1	A = Cx'xxAA _H D = xxF0 _H	A = Cx'xxAA _H D = xxF5 _H	A = RLOC D = <状态>	A = Cx'xxAA _H D = xxFA _H
2	—	—	—	A = FF'F00C _H D = 裕量值

注:

RLOC 代表 Flash 寄存器区的寄存器偏移量（rr），Flash 寄存器区的起始地址为 FF'F000_H（相应的寄存器地址为 FF'F0rr_H）。

<状态>代表返回的状态字。

裕量值代表用于裕量控制的控制字。

表中的虚拟地址（Cx'xxAA_H）必须指向 Flash 存储空间（如 C0'00AA_H）。

为了查询 Flash 模块的忙碌状态位（BUSY），可在命令模式下执行“读 Flash 状态”命令。

复位到读命令退出未完成的命令序列，并清除状态寄存器 FSR 中的错误标志位。除非执行密码检测序列，否则可在命令序列的任意时刻执行复位命令。复位并不会退出命令模式。

状态清零命令将清除错误标志位以及写状态位 PROG、ERASE（硬件控制的标志位不受影响）。只有在读模式下可接受状态清零命令，否则会产生序列错误。

读寄存器命令将返回以下寄存器的内容：

- Flash 状态寄存器 FSR，提供 Flash 的状态信息。
- 保护设置寄存器 PROCON，指示被保护的扇区。
- 裕量控制寄存器 MAR，指示 Flash 的读裕量。

写裕量寄存器命令用于校验 Flash 的操作和用户控制的刷新操作，以识别并纠正有问题的数据位（见[章节 3.8.3](#)）。

编程/擦除命令

表 3-4 命令序列定义（编程和擦除）

周期	进入页面模式 ¹⁾	装载页面数据字 ²⁾	写页面 ³⁾	擦除扇区 ¹⁾	擦除字线 ¹⁾
1	A = Cx'xxAA _H D = xx50 _H	A = Cx'xxF2 _H D = WDAT	A = Cx'xxAA _H D = xxA0 _H	A = Cx'xxAA _H D = xx80 _H	A = Cx'xxAA _H D = xx80 _H
2	A = WLOC D = xxAA _H	—	A = Cx'xx5A _H D = xxAA _H	A = Cx'xx54 _H D = xxAA _H	A = Cx'xx54 _H D = xxAA _H
3	—	—	—	A = SLOC D = xx33 _H	A = WLA D = xx03 _H

- 1) 当保护功能被使能时，该指令序列被拒绝。
 2) 如果写入页面缓存的数据超过 128 字节，超出容量的数据将丢失。
 3) 只有事先已经进入页面模式，才可接受该命令序列。

注：

WLOC 代表（128 字节页面缓存写入）128 字节存储块的首地址单元（最低地址单元），如 C0'AB80_H 或 C0'AC00_H（128 字节宽度）。

WDAT 代表将要保存在缓存中的数据字。

SLOC 代表目标扇区的首地址单元（最低地址单元），如扇区 3 的首地址 C0'6000_H。

WLA 代表将要擦除的 256 字节目标字线的首地址单元（最低地址单元），如最高 256 字节（扇区 3 的顶部字线）的首地址 C0'7F00_H。

表中的虚拟地址（Cx'xx.._H）必须指向 Flash 存储空间（如 C0'00AA_H）。

为了查询 Flash 模块的忙碌状态位（BUSY），可在命令模式下执行“读 Flash 状态”命令。

警告：若在擦除 Flash 前，已对 Flash 页面（128 字节的缓存空间）进行不止一次的写操作，将可能会破坏相邻存储单元的数据！当编程算法没有写在连续的地址单元时，这尤其应当注意。

进入页面模式命令将清除页面缓存并初始化内部缓存指针，准备进行 128 字节页面编程。状态寄存器 FSR 中的位 PAGE 被置位以表明 Flash 处于页面模式。在页面模式下，发出进入页面模式命令将退出当前操作，开始新的页面操作。在退出页面期间，写入页面缓存的数据将会丢失。进入页面模式命令还定义了将被编程的 128 字节页面的首地址。

注：只有在保护机制无效的情况下，进入页面模式命令才有效。

装载页面数据字命令将数据字存放到页面缓存中。页面缓存的偏移地址由内部缓存指针决定，每次加载操作之后缓存指针自动递增。超过缓存容量（128 字节）的数据字将会丢失（无出错提示）。

注：只有在页面模式激活（有效）的情况下，装载页面数据字命令才有效。

写页面命令将 128 字节页面缓存中的内容（包括纠错码）写入（编程）Flash 阵列。编程页面的地址由前面的进入（安全）页面模式命令定义。

写页面命令发出之后，Flash 模块进入命令模式，此时 $PAGE = 0$ ， $PROG = 1$ ， $BUSY = 1$ 。对 Flash 模块的读操作延迟到命令模式结束之后才执行。编程操作自动执行，无需额外的用户控制。

如果对安全页面进行写操作，新的保护配置（包括关键字或保护确认代码）在该命令执行之后立刻生效。

注：只有在页面模式激活（有效）的情况下，写页面命令才有效。

擦除扇区命令将清除所选扇区（见 SLOC）的所有数据位。

擦除扇区命令发出之后，Flash 模块进入命令模式，此时 $ERASE = 1$ ， $BUSY = 1$ 。对 Flash 模块的读操作延迟到命令模式结束之后才执行。擦除操作自动执行，无需额外的用户控制。

注：只有在保护机制无效的情况下，擦除扇区命令才有效。

擦除字线命令将清除所选 256 字节字线上（见 WLA）的所有数据位。

擦除字线命令发出之后，Flash 模块进入命令模式，此时 $ERASE = 1$ ， $BUSY = 1$ 。对 Flash 模块的读操作延迟到命令模式结束之后才执行。擦除操作自动执行，无需额外的用户控制。

注：只有在保护机制无效的情况下，擦除字线命令才有效。

保护控制命令

表 3-5 命令序列定义（保护控制）

周期	禁止读保护	禁止写保护	重新使能保护功能	擦除安全字线 ¹⁾	进入安全页面模式 ¹⁾
1	A = Cx'xx3C _H D = xx00 _H	A = Cx'xx3C _H D = xx00 _H	A = Cx'xx5E _H D = xx5E _H	A = Cx'xxAA _H D = xx80 _H	A = Cx'xxAA _H D = xx55 _H
2	A = Cx'xx54 _H D = PW1	A = Cx'xx54 _H D = PW1	—	A = Cx'xx54 _H D = xxA5 _H	A = SECLOC D = xxAA _H
3	A = Cx'xxAA _H D = PW2	A = Cx'xxAA _H D = PW2	—	A = SECWLA D = xx53 _H	—
4	A = Cx'xx54 _H D = PW3	A = Cx'xx54 _H D = PW3	—	—	—
5	A = Cx'xxAA _H D = PW4	A = Cx'xxAA _H D = PW4	—	—	—
6	A = Cx'xx5A _H D = xx55 _H	A = Cx'xx5A _H D = xx05 _H	—	—	—

1) 当保护功能被使能时，该指令序列被拒绝。

注：当检查第二个或第四个密码字时，不能执行复位到读模式命令。此时，该命令被视为密码字。

注：

SECLOC 代表（128 字节页面缓存写入）安全扇区内 128 字节存储块的首地址单元（最低地址单元），如 C0'0080_H 或 C0'0100_H。

SECWLA 代表将被擦除的 256 字节安全字线的首地址单元（最低地址单元），如 C0'0100_H 表示高 256 字节字线的首地址。

PWn 代表组成 64 位安全校验码中的四个密码字（n=1...4）。

表中的虚拟地址（Cx'xx.._H）必须指向 Flash 存储空间（如 C0'00AA_H）。

为了查询 Flash 模块的忙碌状态位（BUSY），可在命令模式下执行“读 Flash 状态”命令。

禁止读保护命令将暂时禁止一般的 Flash 读保护（和一般的 Flash 写保护），此时 $PRODI = 1$ 。只有在执行重新使能保护命令或下一次复位之后，读保护才会被再次使能。

读保护被禁止时，可执行 Flash 读操作（包括插入的 OCDS 指令）；只要某扇区未被写保护锁定，则可对该扇区进行编程/擦除操作。

注：用确认码锁定保护功能之前，可用该命令序列检验设定的关键字是否正确。关键字出错时，由 Flash 的状态寄存器 FSR 中的标志位 PROER 指示。

这是一个受保护的命令序列。需通过 64 位安全码（用户设定的四个密码字）验证之后，该命令序列才能生效（见[章节 3.8.4](#)）。

禁止扇区写保护命令将暂时禁止所有写保护扇区中特定扇区的写保护特性，此时 $SUL = 1$ 。只有在执行重新使能保护命令或下一次复位之后，写保护才会有效。

写保护被禁止时，只要扇区未被一般的读/写保护锁定，则可对该扇区进行所有操作。

注：用确认码锁定保护特性之前，可用该命令序列检验用户设定的关键字是否正确。关键字出错时，由 Flash 的状态寄存器 FSR 中的标志位 PROER 指示。

这是一个受保护的命令序列。需通过 64 位安全码（用户设定的四个密码字）验证之后，该命令序列才能生效（见[章节 3.8.4](#)）。

重新使能保护命令将立即恢复所有已设定但暂时被禁止的保护特性（一般的读写保护和/或特定扇区写保护）。

擦除安全字线命令将清除所选字线上的所有数据位（见 SECLOC）。

擦除安全字线命令发出之后，Flash 模块进入命令模式，此时 $ERASE = 1$ ， $BUSY = 1$ 。对 Flash 模块的读操作延迟到命令模式结束后才执行。擦除操作自动执行，无需额外的用户控制。

该命令执行之后，新的保护配置（包括关键字和确认码）立刻生效（见[章节 3.8.4](#)）。

注：只有在保护机制无效的情况下，擦除安全字线命令才有效。

进入安全页面模式命令将清除页面缓存并初始化内部缓存指针，准备进行 128 字节安全页面的编程。状态寄存器 FSR 中的位 PAGE 被置位以表明 Flash 处于页面模式。在页面模式下，发出进入安全页面模式命令将退出当前操作，开始新的页面操作。在退出页面期间，写入页面缓存的数据将会丢失。进入安全页面模式命令还定义了将被编程的 128 字节安全页面的首地址。请参阅[章节 3.8.4](#)。

注：只有在保护机制无效的情况下，进入安全页面命令才有效。

程序 Flash 编程和安全扇区编程的相互配合

XC164CM 的片上 Flash 模块通过查询程序 Flash 以及安全扇区的内部状态信息获知其当前状态。擦除或编程操作之后（写页面命令），内部状态信息被更新。复位后，始终选择查询程序 Flash 的状态信息。

为了保证编程正确，必须确保编程操作（程序 Flash/安全扇区）之前的操作总是对同一区域的编程或擦除操作，否则可能会存储错误数据。

对程序 Flash/安全扇区的擦除始终能够正确进行，和之前进行何种操作无关。

换言之：

请确保在对安全扇区进行擦除和编程之间，不对程序 Flash 擦除/编程或复位。

请确保在对程序 Flash 进行擦除和编程之间，不执行安全扇区的擦除/编程。

3.8.3 纠错与数据完整性

通过纠错码（ECC）来增强数据的完整性。在进行 Flash 写操作时，动态生成 ECC，ECC 和相应的数据一同存储到 Flash 阵列中。每次读取 Flash 时，将 64 位数据连同相关的 8 位 ECC 一起取出，对 ECC 进行评估。

单位错误可被检测到、并在运行过程中自动修正。因此单位错误不会影响系统操作。

双位错误可被检测到、并触发访问出错强制中断，从而避免使用错误的指令或数据。

Flash 状态寄存器 FSR 中的专用标志位（SBER，DBER）指示每次读操作的出错情况。

出现双位错误（不可由 ECC 自动纠错）的概率非常低。在检测到单位错误之后，执行数据恢复操作，即可避免双位错误的发生。数据恢复操作包括以下步骤：

- 检测包含错误位的字线
- 暂时保存字线的内容
- 擦除字线
- 重新对擦除字线编程（需要两个写页面操作）

由于在读取字线时，单位错误已由 ECC 自动修正，因此写入临时缓存的字线数据是正确的。采用标准的命令序列进行擦除和编程。

检验操作

检验操作可检测到出错的字线。位 SBER 清零后开始读取 Flash 中的某区域。由于每次可从 Flash 阵列读取 64 位数据，因此每次读操作之后读地址自动加 8，从而可以使所需的读周期最少。读取指定区域之后，位 SBER 指示该区域是否存在单位错误。检验算法可将检查区域逐渐缩小到一条字线，或者顺序检查所有字线。

刷新操作

在出现读错误（并在运行期间执行数据恢复操作）之前检查出问题位，并对它们重新编程（刷新），甚至可以避免单位错误。将检验操作和裕量检测控制相结合可检测出问题位。

裕量检测控制

用 Flash 单元存储的电荷来表征位电平。单元中的电荷发生变化时（比如，对相邻单元进行操作时，电荷耦合会引起电荷变化），相应的位可能被读错。由于电荷在缓慢变化，因此可在某位被读错之前检测到电荷的变化。在这种情况下，也可以采用一些预防性的修正措施（通过软件修正）。

读取 Flash 单元时，采用更严格的裕量比较可检测到问题位（即电荷变化位）。由裕量控制寄存器 MAR 选择裕量值，可通过“读/写裕量”命令序列访问该寄存器（见**表 3-3**）。

将寄存器 MAR 中的位域 MARLEVSEL 设定为 0001_B或 0100_B，对应选择不同的严格裕量。读取某位时，若采用低电平裕量读取返回 1，采用标准裕量读取返回 0，表明此位有问题，称为弱 0；若采用高电平裕量读取返回 0，采用标准裕量读取返回 1 值，表明此位有问题，称为弱 1。通过标准裕量和严格裕量可检测出这些问题位。

注：读操作可以直接跟在 MAR 修改操作之后。

MAR

裕量控制寄存器

SFR (FF'F00C_H)

复位值: 0000_H

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

符号	位序号	读写类型	功能描述
MARWV	7	rw	裕量写入确认 0 复位值。MARWV 不能写 0。 1 每次对寄存器 MAR 进行写操作时，该位必须置位（MARWV=1），与写操作的目的无关。
MARLEVSEL	[3:0]	rw	裕量电平选择 0000 标准读裕量（正常操作） 0001 低电平裕量（用于检验弱 0） 0100 高电平裕量（用于检验弱 1） 其它 保留

注：只有通过写裕量命令才能写入裕量值。每次写入时 MARWV 必须被置位。

3.8.4 保护与安全特征

Flash 模块提供了强大、灵活的保护机制，防止数据和代码被破坏（即被擦除）、被意外修改（即重新编程），同时保护 Flash 不被意外读访问。Flash 模块提供了两种保护机制：

- **特定扇区写保护**保护各特定扇区不被擦除和编程。这对保证启动程序不受破坏非常重要；也可避免由系统故障、甚至是误操作造成代码/数据被修改。
- **一般读/写保护**保护整个程序 Flash 区域不被访问，包括数据读访问、指令读取（即跳转到程序 Flash 区）、OCDS 操作，以及擦除/编程操作。不过可执行命令序列和寄存器访问。

保护特征由用户编程设定。在对 Flash 中的某区域重新编程、或调用外部子程序时，可暂时禁止保护特征。禁止和重新使能保护特征均由软件控制。不过，复位后所有用户设置的保护特征都将自动生效（使能）。

根据应用需求，可将两种保护特征相结合，实现灵活的保护机制，保护 Flash 或部分 Flash 不被非法的编程或擦除。

注：只对程序 Flash 提供保护，程序 SRAM 无保护机制。

密码与安全校验码

所有的保护控制（保护特性的设定、禁止、重新使能）通过命令序列来实现。这些命令序列和编程/擦除命令序列（见表 3-5）相似。暂时中止保护特征的两个命令序列（禁止读保护命令，禁止扇区写保护命令）也受保护，需通过密码（64 位安全校验码）验证之后，命令序列才能生效，从而最大限度地保证了数据不被非法访问。

验证密码时，命令序列中的四个密码字和存放在安全扇区中的四个关键字（组成 64 位安全校验码）进行比较。如检测到任何不匹配，保护机制保持有效，扇区仍被锁定，Flash 状态寄存器指示保护错误（PROER）。这种情况下，只有在下一次系统复位之后，新的禁止扇区写保护或禁止读保护命令才被接受。

安全特征设定

设定安全特征时，将下列数据写入安全扇区（见图 3-7）：

- 安全控制位，选择要设定的安全特征
- 64 位安全校验码（四个关键字）
- 16 位确认码

注：只要保护功能被使能，安全扇区自身也受保护。

安全控制位可通过寄存器 PROCON 进行查询。设定安全控制位时，必须采用相同的位结构。

PROCON

保护控制寄存器

SFR（FF'F004_H）

复位值: xxxxx_H

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
R PRO	-	-	-	-	-	-	-	-	-	-	-	-	SL3	SL2	SL1	SL0
rh	-	-	-	-	-	-	-	-	-	-	-	-	rh	rh	rh	rh

符号	位序号	读写类型	功能描述
RPRO	15	rh	读/写保护设置 0 未设定一般读/写保护 1 设定一般读/写保护
SLn (n = 3...0)	3,2,1,0	rh	扇区锁定位 n 0 未设定扇区 n 写保护 1 设定扇区 n 写保护 <i>注：每两个 8KB 的扇区合并为一个 16KB 存储区，可由 SL0 和 SL1 联合锁定。</i>

注：可读取寄存器 PROCON 检查安全设置。要修改安全设置，必须修改安全扇区。

注：对于 64K Flash 器件（不存在第二个 64K 扇区“5”），若扇区“0”到“4”中的任意一个要被写保护，则不存在的扇区“5”（由位 SL3 控制）必须保持“写保护”状态。

对于 32K Flash 器件（不存在第二个 64K 扇区“5”），若扇区“0”到“3”中的任意一个要被写保护，则不存在的扇区“5”（由位 SL3 控制）必须保持“写保护”状态。这种情况下，位 SL2 与控制无关。

暂时禁止安全特征时，必须正确输入 64 位安全校验码（如，494E'4649'4E45'4F4E_H）。输入四个密码字时，出现任何错误都将中止命令，并“冻结”当前的安全状态，直至系统下一次复位。

16 位确认码（8AFE_H）用于确认所设定的安全特征有效。安全设置生效之前，须通过验证。

安全信息和确认码分别存储在不同的字线上，从而可独立被修改和擦除。

构成安全信息的每一个字节在字线上存储三次，并以零字节结束。因此，每个字（16 位）的存储占用两个双字的空间（见图 3-7 的示例）。每个字节的三重备份保证了极高的数据可靠性。

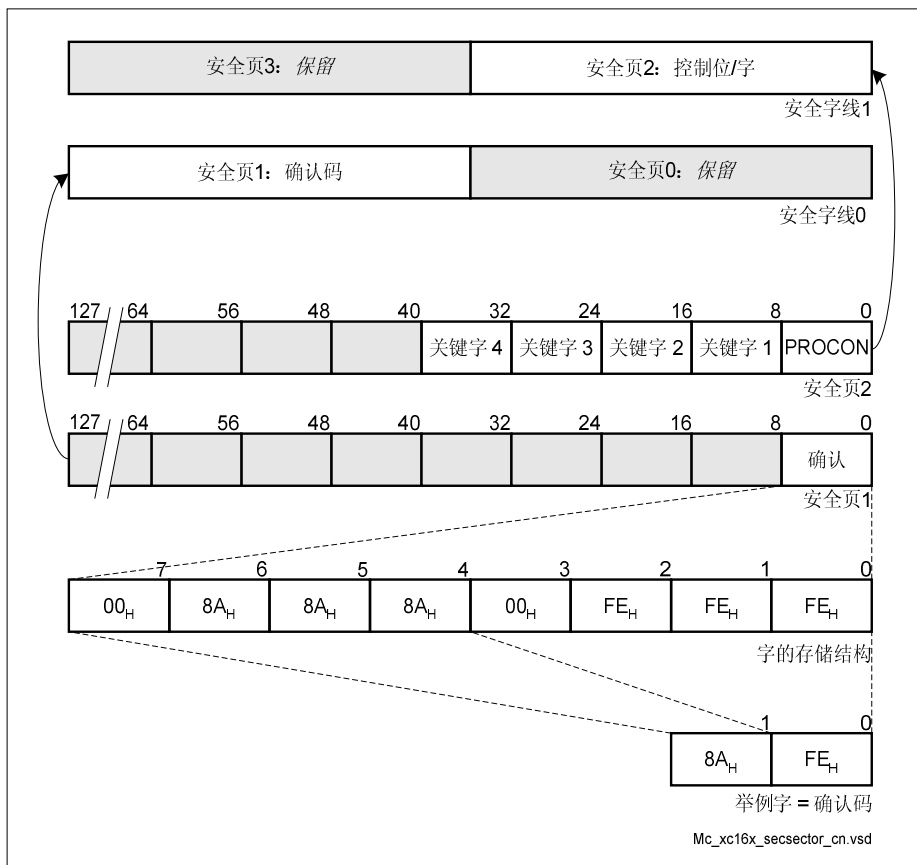


图 3-7 安全扇区结构

一旦要修改安全设置（安全设定、修改、取消），需执行以下步骤：

- 擦除安全字线 0 清除确认码。

此操作将取消所有的保护特征（PROIN = 0）。

- 擦除安全字线 1。
- 将期望的安全设置和关键字写入安全页面 2。
- 验证设定的安全设置和关键字。
- 将确认码写入安全页面 1。

此操作设定了新的保护特征。

依照以上步骤，可以防止如下述情况所造成的死锁。在确认码存在的情况下，设定关键字出错（比如编程过程中的电源问题），此时安全特征会立刻有效而错误的关键字不被识别，从而会导致死锁。

读/写保护控制

代码和数据的读保护分别通过寄存器 IMBCTR 中的控制位 DCF（禁止取指）和 DDF（禁止读数据）来激活。只要禁止标志位（DCF, DDF）被置位并且读保护处于激活状态，读访问将被禁止，寄存器 IMBCTR 中的位 RPA（读保护有效）指示当前读保护是否被激活。访问被保护的 Flash 将返回虚假值 1E9B_H。读保护被禁止（RPA = 0）时，位 DCF 和 DDF 的设置对读访问没有任何影响。

复位后，从片上 Flash 开始执行指令时，位 DCF 和 DDF 被清零，从而可从安全区域执行代码，允许对 Flash 进行所有读写操作。用户可编程置位 DDF，禁止从 Flash 中读取数据，但此时仍可执行指令。

其它任何复位方式（包括引导模式）下，两个标志位均被置位（若保护已设定）。输入 64 位安全校验码，可通过在外部资源中运行的软件暂时禁止读保护。

注：位 DCF 和 DDF 只能由软件置位，不能由软件清零。

注：当读保护有效，执行片上 Flash 的程序时，一定不能置位 DCF。

如果已设定保护特征（RPRO=1）、且当前保护未被禁止（PRODI = 0），则读/写保护有效（RPA = 1）。

读/写保护处理

复位后，位 RPA 指示读/写保护是否被设定。用户可编程暂时禁止读/写保护（此时 RPA = 0）。RPA = 1 时，由 DCF 和 DDF 禁止对 Flash 进行读访问。由于从片上 Flash 启动时，DCF 和 DDF 被清零，用户软件将负责保护处理。

如果读/写保护被使能，为了避免通过调试接口对 Flash 进行未经授权的访问，调试系统将被禁用。除非用户编程使能调试接口，此时即使读/写保护被使能，调试接口仍被暂时激活。

为确保安全的读/写保护，请遵守以下规则：

- 不跳转至（JUMP）或调用（CALL）外部存储器地址
- 不执行从任何接口下载的代码
- 将控制移交至外部存储区之前（不返回），置位 DCF 和 DDF
- 禁用调试系统

注：一旦读/写保护无效，可立即执行外部代码；也可使能调试接口，对被保护器件进行调试。

不过，只有当通过特定的安全密码验证，使读写保护无效，才可执行以上操作。

3.8.5 Flash 状态信息

Flash 状态寄存器 FSR 中存放着 Flash 模块的所有状态信息，包括：

- 工作状态
- 错误情况
- 安全等级

应在命令序列执行前后读取 FSR。不能直接写入 FSR。“状态清零”命令将清零错误标志位和状态标志位 PROG、ERASE。“复位到读”命令将清零错误标志位。

FSR

Flash 状态寄存器

SFR (FF'F000_H)

复位值: 0xxx_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	SUL	-	PRO IN	PRO DI	DB ER	SB ER	PRO ER	SQ ER	-	OP ER	PA GE	ERA SE	PR OG	BU SY
-	-	rh	-	rh	rh	rh	rh	rh	rh	-	rh	rh	rh	rh	rh

符号	位序号	读写类型	功能描述
SUL	13	rh	扇区解锁 0 扇区被保护 1 所有扇区被暂时解锁（检查一般保护）
PROIN	11	rh	保护特性设定 0 未设定安全特性 1 设定一般读/写保护和/或特定扇区写保护（见寄存器 PROCON）
PRODI	10	rh	保护禁止 0 一般读/写保护有效（若该安全特性已被设定） 1 一般读/写保护被暂时禁止
DBER	9	rh	双位错误 （通过“状态清零”，“复位到读”命令序列对该位清零） 0 未产生双位错误 1 检测到一个双位错误（不可纠正）

符号	位序号	读写类型	功能描述
SBER	8	rh	单位错误 （通过“状态清零”，“复位到读”命令序列对该位清零） 0 读访问未出错 1 检测到单位错误并自动纠正
PROER	7	rh	保护错误 （通过“状态清零”，“复位到读”命令序列对该位清零） 0 未检测到保护错误 1 发生保护错误：试图对锁定扇区编程/擦除或者输入的安全校验码无效 ¹⁾
SQER	6	rh	命令序列错误 （通过“状态清零”，“复位到读”命令序列对该位清零） 0 未检测到命令序列错误 1 执行不合理的命令步骤导致状态机操作中 <i>注：执行“复位到读”命令中止命令序列时，SQER 不置位。当 Flash 模块处于忙碌状态（PROG 或者 ERASE 未被清零）时，执行“状态清除”命令，SQER 被置位。</i>
OPER	4	rh	操作错误 （通过“状态清零”，“复位到读”命令序列对该位清零） 0 Flash 操作正确完成或正在进行中 1 Flash 操作未能正确完成（退出）
PAGE	3	rh	页面模式 （通过“复位到读”序列对该位清零） 0 Flash 未工作在页面模式 1 Flash 工作在页面模式，正在填充页面缓存 <i>注：标准读模式下，页面模式也能被激活。</i>
ERASE	2	rh	擦除状态 （通过“状态清零”，“复位到读”命令序列对该位清零） 0 无擦除操作

符号	位序号	读写类型	功能描述
			1 正在进行 Flash 擦除操作
PROG	1	rh	编程状态 （通过“状态清零”，“复位到读”命令序列对该位清零） 0 无编程操作 1 正在进行 Flash 编程（写页面）
BUSY	0	rh	Flash 忙碌 0 就绪 ：Flash 的命令序列执行完毕。模块处于标准读模式 1 忙碌 ：正在执行命令序列或者 Flash 正处于充电状态 ²⁾ 。模块不处于读模式

1) 保护出错后，只有在复位后才接受新的密码序列。

2) 系统复位后，忙碌标志位 BUSY 会持续有效约 250μs，直至 Flash 内部电压稳定。

注：通过分析位 PROG、ERASE 和位 BUSY、OPER，控制软件可确定当前 Flash 的状态：操作正在执行、已经结束、还是已经中止。

3.8.6 操作控制和错误处理

命令序列中的最后一条命令写入命令寄存器之后，开始执行命令序列，由状态标志位（编程状态标志 **PROG**，擦除状态标志 **ERASE**）和忙碌标志位 **BUSY** 指示 Flash 的当前状态。通过查询 **BUSY** 标志位就能够检测命令是否执行完毕。建议用户随后查询错误标志位，以确定是否发生误操作。

推荐采用下述通用步骤执行命令：

- 清除状态位
- 向 Flash 模块写入命令序列
- 查询位 **SQER** 和 **PROER** 位，确保命令序列正确
- 如果出错：通过“状态清零”或者“复位”命令清除标志位，并执行相关操作（如重试）
- 查询位 **PROG** 和 **ERASE**，检查命令是否正确
- 查询 **BUSY** 确定命令是否执行完毕
- 检查错误标志位

错误标志位锁存在状态寄存器 **FSR** 中。只要错误标志置位，则表明出现了相应的错误。因此，有必要使用命令清除这些错误标志。

表 3-6 举例给出软件如何处理出错情况。

表 3-6 出错情况的软件处理

错误类别	出错情况	软件处理
SQER 命令序列错误	寄存器地址错误，命令/扇区/字线地址错误，命令代码错误，非法的命令序列	检查地址或代码，使用正确值重新执行
OPER 操作错误	由于软件复位、看门狗复位、硬件热重启而退出编程或擦除操作	重新执行 Flash 操作（ PROG 和 ERASE 指示出错的操作类型）
PROER 保护错误	对保护扇区进行写操作（进入页面模式），密码错误	保护禁止后重试，复位后重试
SBER 单位错误	纠错码（ECC）已检测到一个单位错误	更新错误字线（见 章节 3.8.3 ）
DBER ¹⁾ 双位错误	纠错码（ECC）已检测到一个双位错误	双位错误触发强制中断

1) 若检测到一个单位错误后即执行刷新操作，就不会发生双位错误（参见**章节 3.8.3**）。

复位与掉电处理

当系统复位，Flash 内部电压稳定之后，Flash 模块复位其状态机，进入标准读模式。内部电压需要逐渐升高（比如掉电之后）或逐渐下降（比如编程、擦除操作被中断之后）。电源是否达到稳定状态由标志位 **BUSY** 指示。电压稳定前的所有访问会延迟到电压稳定之后才执行。

系统进入掉电模式、休眠模式或空闲模式（Flash 关闭）、通过寄存器 **SYSCON3** 禁用 Flash 模块、或执行软件复位，这些操作都会请求 Flash 逐渐降低其内部电压。Flash 命令执行完毕、当前操作（包括编程、擦除操作）结束之后，才会接受请求，继而 CPU 可执行需要的操作。

注：当计算系统从空闲、休眠、或掉电状态中唤醒所需的延迟时间时，必须将电源达到稳定所花费的时间也计算在内。

3.9 程序存储器控制

内部程序存储模块 IMB 由控制存储器访问的接口部分（程序存储器接口 PMI）和以下存储模块组成：

- 内部程序 Flash（包括纠错码 ECC）存储器，起始地址 C0'0000_H
- 2KB 程序 SRAM，起始地址 E0'0000_H

Flash 和程序 SRAM 既可存储程序代码、也可存储数据，代码和数据都可被 CPU 访问。

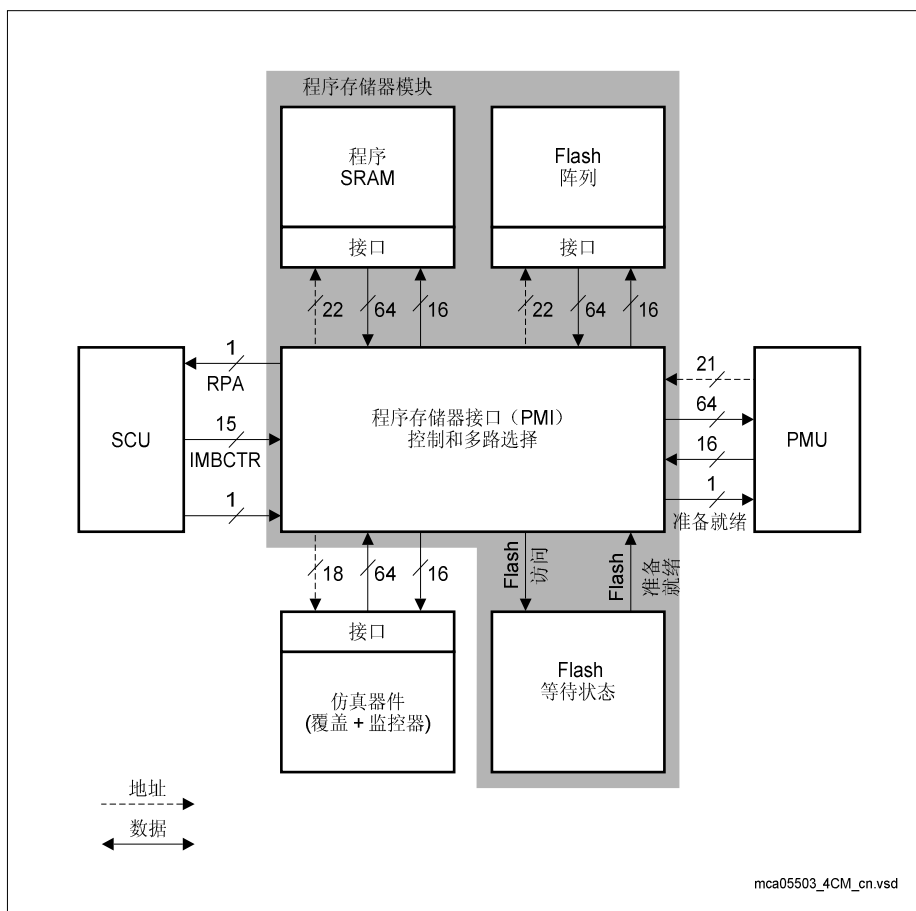


图 3-8 内部程序存储模块 IMB 框图

图 3-8 所示为 IMB 主框图。为简化起见，图中未标出特殊的控制信号。

可根据应用需求设置存储器的访问特性。当程序从片上 **Flash** 存储器或内部 **SRAM** 中运行时，在某些情况下访问延时必须相同。为了解决这个问题，可将 **SRAM** 的访问时间设定为和 **Flash** 时序相同。内部 **SRAM** 最快可支持单周期访问。在 **IMB** 内部的程序存储接口（**PMI**）模块中，包含可编程的等待状态发生逻辑。

SRAM 和 **Flash** 存储器的访问周期数可分别编程设定。

3.9.1 地址映射

程序存储模块的地址映射如 **图 3-9** 所示。

程序 **Flash** 的起始地址始终为 $C0'0000_H$ ，程序 **SRAM** 的起始地址为 $E0'0000_H$ 。

Flash 只读状态寄存器的起始地址为 $FF'F000_H$ ，必须避免对该地址段进行写操作。

若某存储区未明确标明为有效存储/寄存器区，则禁止对其进行访问。

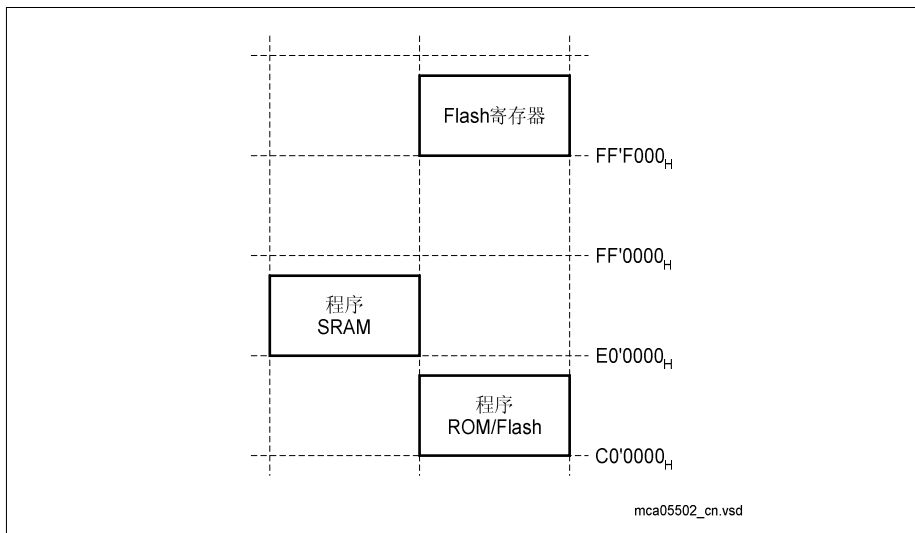


图 3-9 程序存储模块 IMB 地址映射

3.9.2 Flash 存储器访问

PMU/PMI 和 Flash 存储器接口的内部功能结构如图 3-10 所示。对 Flash 的访问分两个阶段：

- Flash 阵列以不超过 50ns 的固定延时送出 CPU 所需数据。第一个访问阶段（1+ WS）包括访问 Flash 阵列的时间，因此必须通过 IMBCTRL 寄存器的位域 WSFLASH 选择相应的等待状态数。

例如：系统以 40 MHz 工作时，时钟周期为 25 ns。因此，访问需要两个周期，必须选择插入一个等待状态（1+1）。

- 纠错（ECC）和 PMU 各需要一个时钟周期。

CPU 在 1 + WS + 2 个时钟周期（若选择 1 个等待状态，则对应 4 个周期）之后获得所需数据。不过，该延时只适用于单次访问（从非线性地址读取）。预取机制使后续访问的第 1 阶段和上一次访问的第 2 阶段重合，因此进行线性访问（如指令读取）时系统性能很高。

Flash 阵列本身只需第 1 阶段，因此可以每 1+WS 个周期对 Flash 访问一次。

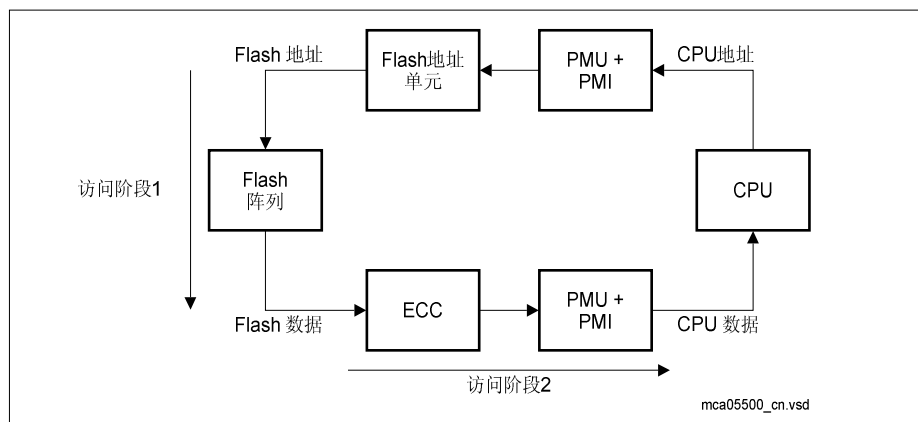


图 3-10 Flash - PMI 结构

Flash 访问举例：1 个等待状态（WS = 1）

第一次访问 Flash（如跳转至 CPU 指定的 Flash 中的第一个地址），需要 4 个时钟周期读取相应的数据（1+1+2）。

随后 Flash 按照 4 -2 -2 -2 -2 -...的时钟周期连续送出数据字。

当 CPU 需要从其它地址（而非预取单元中的地址）读取数据时（比如跳转），Flash 地址单元立刻切换到新地址，开始新的读取序列（4 -...）。

注：如果访问 Flash 所需的时间超过两个周期（多于一个等待状态），预取将没有意义，此时将禁用 Flash 预取功能。

3.9.3 IMB 控制功能

等待状态的产生

等待状态由等待状态单元产生，指示所需数据（或指令字）何时可用。Flash 存储器的地址窗起始地址为 **C0'0000_H**，占用 **2MB** 的地址空间。

复位后，存储器访问的缺省设置为：访问 **Flash** 需要两个时钟周期；访问程序 **SRAM** 需要一个时钟周期。

IMB 控制寄存器

寄存器 **IMBCTR** 可控制 **Flash** 以及其它 **IMB** 存储器模块等待状态的产生。一个等待状态代表一个时钟周期。为了使访问其它存储器的时序与访问 **Flash** 的时序一致，必须插入以时钟周期为单位（取决于时钟频率）的等待状态。也可采用 **Flash** 时序对用户 **PSRAM** 进行访问，比如用于仿真。

寄存器安全机制保护该寄存器不受意外修改。该寄存器只能由硬件复位，不能由软件复位或看门狗复位。

IMBCTR

IMB 控制寄存器

ESFR (F0FE_H/7F_H)

复位值: **xx01_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RPA	-					DDF	DCF	-				WS ROM	WS RAM	WS FLASH	
rh	-					rwh	rwh	-				rw	rw	rw	

符号	位序号	读写类型	功能描述
RPA	15	rh	读保护激活 该位监控 Flash 内部读保护的状态。 0 Flash 内部读保护未激活。位 DCF 和 DDF 无效 1 Flash 内部读保护被激活。位 DCF 和 DDF 有效
DDF	9	rwh	禁止从 Flash 存储器读取数据 该位使能/禁止从内部 Flash 存储器读取数据。该位一旦置位，只能由硬件复位清零。 0 允许从 Flash 存储器中读取数据

符号	位序号	读写类型	功能描述
			1 禁止从 Flash 存储器中读取数据。当 RPA = 0 时，该位无效
DCF	8	rwh	禁止从 Flash 存储器取指令 该位使能/禁止从内部 Flash 存储器读取指令。该位一旦置位，只能由硬件复位清零。 0 允许从 Flash 存储器中读取指令 1 禁止从 Flash 存储器中读取指令。当 RPA = 0 时，该位无效
WSROM	3	rw	访问用户 ROM 的等待状态控制 该位定义了 IMB 中用户 ROM 的读访问特性。用户 ROM 占用的地址空间为 C0'0000 _H ...DF'FFFF _H 。不可对该存储区进行写访问。 0 以最高速度访问用户 ROM，即单时钟周期的读访问时间。 1 对用户 ROM 的读访问操作类似于用户 Flash，需考虑流水线结构和访问时序以获得相同的特性。 <i>注：位 WSROM 只对集成有 ROM 的 XC164CM 系列产品有效。</i>
WSRAM	2	rw	访问程序 RAM 的等待状态控制 该位定义了 IMB 中程序 SRAM 的读访问特性。程序 SRAM 占用的地址空间为 E0'0000 _H ...F7'FFFF _H 。对该存储区的写访问在一个时钟周期内完成。 0 以最高速度访问程序 SRAM，即单时钟周期的读访问时间。 1 对程序 SRAM 的读访问操作类似于用户 Flash，需考虑流水线结构和访问时序以获得相同的特性。
WSFLASH	[1:0]	rw	Flash 存储器的等待状态控制 该位域定义了在读取 Flash 存储器时，需要插入的等待状态个数。Flash 占用的地址空间为 C0'0000 _H ...DF'FFFF _H 。

符号	位序号	读写类型	功能描述
			00 Flash 读访问时无需附加等待状态，即一个时钟周期完成 Flash 读访问。
			01 Flash 读访问时附加一个等待状态，用两个时钟周期完成 Flash 读访问（缺省值）。
			10 Flash 读访问时附加两个等待状态，用三个时钟周期完成 Flash 读访问。
			11 Flash 读访问时附加三个等待状态，用四个时钟周期完成 Flash 读访问。

4 中央处理器（CPU）

中央处理器（CPU）的基本任务是对指令进行读取和解码，为算术逻辑单元（ALU）和乘累加单元（MAC）提供操作数，并在 ALU 和 MAC 中进行操作数运算，保存先前的计算结果。因为 CPU 是 XC164CM 微控制器的核心，它还受外设子系统操作的影响。

XC164CM 实现了 5 级处理流水线（外加 2 级取指令流水线），能够并行处理多达 5 条指令。由于采取了该并行机制，XC164CM 的大多数指令在一个时钟周期内完成。

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

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

注：以下描述中若出现“外部访问”，是指对由 EBC 控制的资源的访问；换言之，是对外部总线和/或内部 LX-bus 的访问。

访问外部存储器期间，CPU 会尽可能保持运行。如果所需的外部数据尚未准备好，或在完成前一次的访问之前，CPU 请求新的外部存储器访问，EBC 将使 CPU 处于保持状态（暂停），直至完成被请求的访问操作。EBC 模块将在第 9 章专门介绍。

XC164CM 的片上外设单元由独立的时钟产生器提供时钟，几乎独立于 CPU 进行工作。CPU 与这些外设之间通过特殊功能寄存器（SFR）来交换数据和控制信息。

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

XC164CM 支持两种基本的中断处理类型：

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

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

与其它片上外设相比，看门狗定时器与 CPU 之间有着更紧密的联系。若看门狗被使能，CPU 要在设定的时间之内（可编程设定）服务（刷新）看门狗。否则，看门狗会复位芯片。因而，在执行错误代码时，看门狗能够防止 CPU 程序跑飞。复位后，看门狗定时器自动开始重新计数。如有需要，可通过软件禁止看门狗定时器。

除了正常工作状态外，CPU 还有以下几种特殊状态：

- **复位状态**：任何类型的复位（硬件、软件、看门狗）都将强制 CPU 进入预先定义的有效工作状态。

- **空闲状态：**CPU 时钟关闭，片上外设的时钟仍然运行。
- **休眠状态：**所有片上时钟都关闭（可选择是否关闭 RTC 时钟），外部中断输入使能。
- **掉电状态：**所有片上时钟都关闭（可选择是否关闭 RTC 时钟），所有输入均无效。

可通过中断（如果处于空闲或休眠模式）或复位（如果处于掉电模式）使 CPU 回到正常工作状态。

可通过特定的系统控制指令使系统进入空闲、休眠、掉电和复位状态。

CPU 内核有一套专用的特殊功能寄存器（CSFR）：

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

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

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

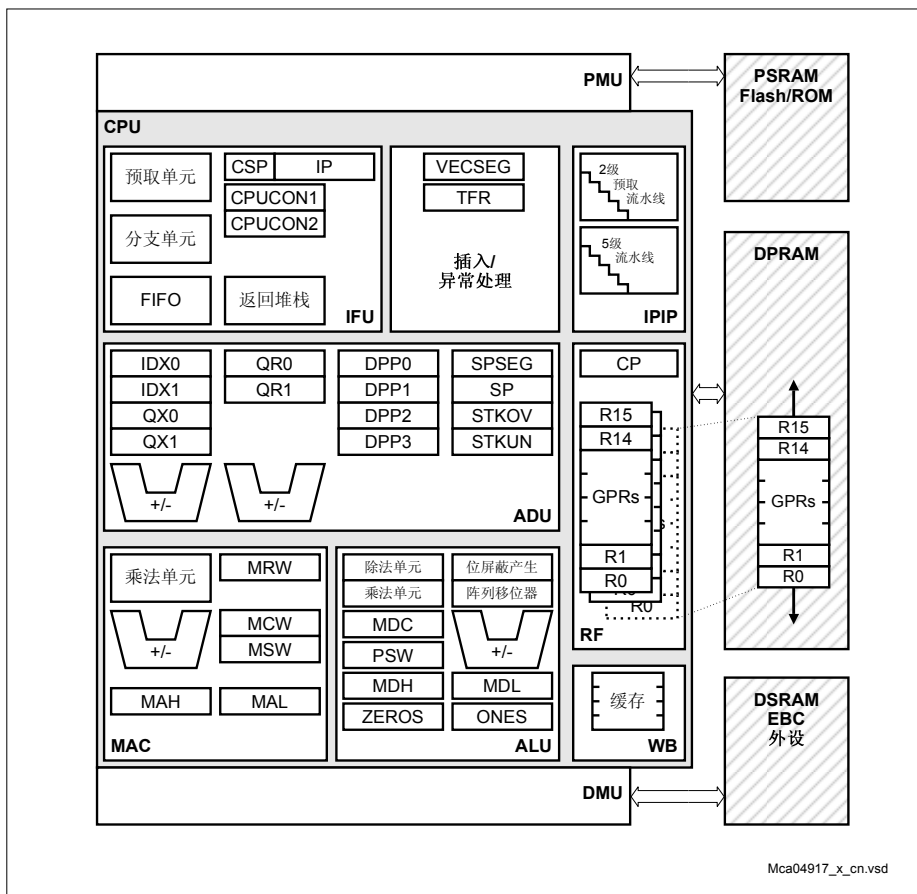
注：若对同一个 CSFR 的软件写请求和硬件修改同时发生，软件写请求占优。

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

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

4.1 CPU 的组成

CPU 的良好性能得益于诸多单元的协调工作（见**图 4-1**），这些单元都针对各自功能进行了优化设计。**预取指单元**和**分支单元**向流水线提供数据，使 CPU 因读取指令而引起的停滞时间最短。**寻址单元**支持多种复杂的寻址模式，从而避免使用附加指令。**算术逻辑单元**和**乘累加单元**用来处理不同长度的数据，执行复杂的运算。三个**存储器接口**和**写缓存**使得由数据传送而引起的 CPU 停滞时间最短。



一般情况下，指令会通过 7 级流水线，每一级处理任务不同（具体见[章节 4.3](#)）：

- 2 级取指流水线将指令从程序存储器中预先读取出来，并存储到指令 FIFO 中
- 5 级处理流水线执行保存在指令 FIFO 中的每一条指令

经过流水线的每一级至少需要一个时钟周期，因此，任何单独的指令都至少需要 5 个时钟周期才能完成。不过由于流水线能并行（同时）处理多达 5 条指令（有分支指令的情况下可以处理 6 条指令），因此，复位后一旦流水线被填满，大部分指令看起来像是在一个时钟周期内完成。

流水线提高了一段时间内的指令平均吞吐量。

4.2 指令读取和程序流控制

指令读取单元（IFU）预先读取指令并对指令进行预处理，从而提供连续的指令流。IFU 可通过 64 位总线从程序管理单元（PMU）中至少同时读取两条指令。预取的指令保存在指令 FIFO 中。

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

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

一旦出现预测指令流不正确的情况，取指流水线被旁路，以减少死循环时间。

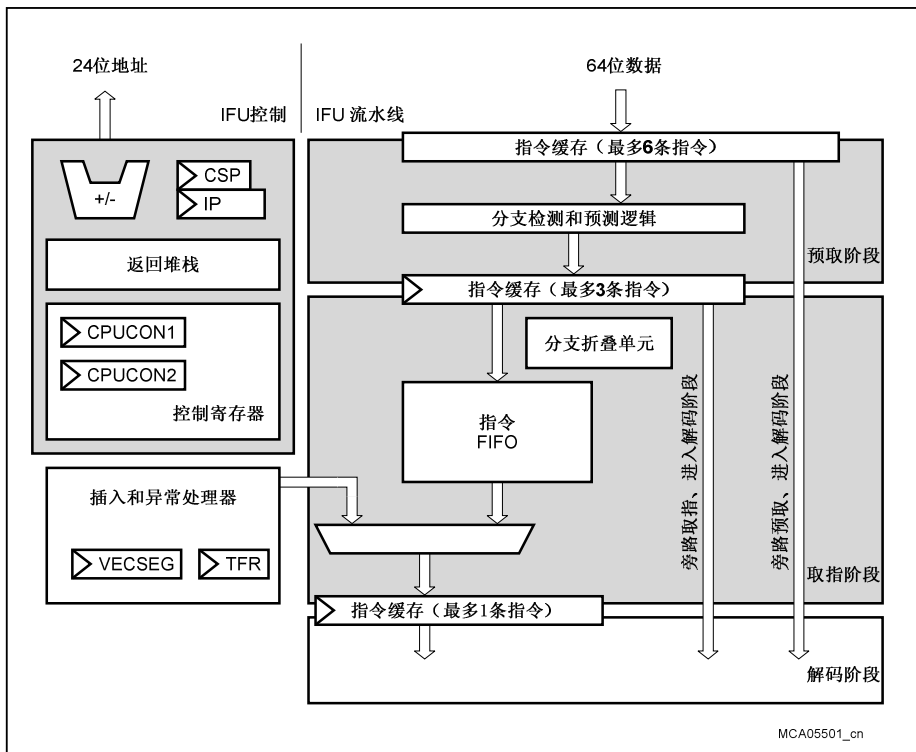


图 4-2 IFU 框图

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

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

注：被错误预测的指令流，其流水线操作将在后续章节中予以描述。

4.2.1 分支检测和分支预测规则

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

表 4-1 分支分类和预测规则

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

1) 对一般的 JMPA 和 CALLA 指令，该位也可根据跳转条件由汇编器自动置位/清零（条件为 cc_Z: “不跳转”，否则：“跳转”）。

4.2.2 预测正确的指令流

假设程序存储器访问无需等待状态¹⁾（0 等待状态），表 4-2 给出指令连续执行的过程。该举例中，大多数指令在一个 CPU 周期内完成， I_{n+6} 指令（多周期指令）占用两个 CPU 周期。图表给出了顺序指令流经过不同流水线阶段的情况。图 4-3 给出相应的程序存储器区段。

从指令 FIFO 中取出指令送入处理流水线的同时，IFU 继续预取指令填充 FIFO。只要 IFU 能够正确预测指令流，指令的执行和预取相互独立。

该举例中采用快速内部程序存储器，预取指令单元读取指令的速度超过处理流水线处理指令的速度。在 T_{n+4} 周期内，FIFO 和预取指令缓存均已被填满，无法继续预取新的指令。PMU 地址保持不变（ T_{n+4} ），直到整个 64 位能够被再次保存（ T_{n+7} ）到 96 位预取指令缓存中。

1) 访问 Flash 存储器可能需要插入等待状态，这取决于实际的工作频率。

表 4-2 预测正确的指令流（顺序执行）

	T_n	T_{n+1}	T_{n+2}	T_{n+3}	T_{n+4}	T_{n+5}	T_{n+6}	T_{n+7}	T_{n+8}
PMU 地址	I_{a+16}	I_{a+24}	I_{a+32}	I_{a+40}	I_{a+40}	I_{a+40}	I_{a+40}	I_{a+48}	I_{a+48}
PMU 数据 64 位	I_{d+1}	I_{d+2}	I_{d+3}	I_{d+4}	I_{d+5}	I_{d+5}	I_{d+5}	I_{d+5}	I_{d+7}
预取指令	I_{n+6}	I_{n+9}	I_{n+12}	I_{n+14}	I_{n+15}	I_{n+15}	I_{n+16}	I_{n+17}	I_{n+18}
96 位缓存	I_{n+13}	I_{n+15}
	I_{n+9}	I_{n+11}		I_{n+19}	I_{n+19}	I_{n+19}	I_{n+19}	I_{n+19}	I_{n+21}
取指	I_{n+5}	I_{n+6}	I_{n+9}	I_{n+12}	I_{n+14}	—	I_{n+15}	I_{n+16}	I_{n+17}
指令缓存		I_{n+7} I_{n+8}	I_{n+10} I_{n+11}	I_{n+13}					
FIFO 内容	I_{n+3}	I_{n+4}	I_{n+5}	I_{n+6}	I_{n+7}	I_{n+7}	I_{n+8}	I_{n+9}	I_{n+10}

	I_{n+5}	I_{n+8}	I_{n+11}	I_{n+13}	I_{n+14}	I_{n+14}	I_{n+15}	I_{n+16}	I_{n+17}
从 FIFO 中 取指	I_{n+4}	I_{n+5}	I_{n+6}	I_{n+7}	I_{n+7}	I_{n+8}	I_{n+9}	I_{n+10}	I_{n+11}
解码	I_{n+3}	I_{n+4}	I_{n+5}	I_{n+6}	I_{n+6}	I_{n+7}	I_{n+8}	I_{n+9}	I_{n+10}
寻址	I_{n+2}	I_{n+3}	I_{n+4}	I_{n+5}	I_{n+6}	I_{n+6}	I_{n+7}	I_{n+8}	I_{n+9}
存储	I_{n+1}	I_{n+2}	I_{n+3}	I_{n+4}	I_{n+5}	I_{n+6}	I_{n+6}	I_{n+7}	I_{n+8}
执行	I_n	I_{n+1}	I_{n+2}	I_{n+3}	I_{n+4}	I_{n+5}	I_{n+6}	I_{n+6}	I_{n+7}
回写	—	I_n	I_{n+1}	I_{n+2}	I_{n+3}	I_{n+4}	I_{n+5}	I_{n+6}	I_{n+6}

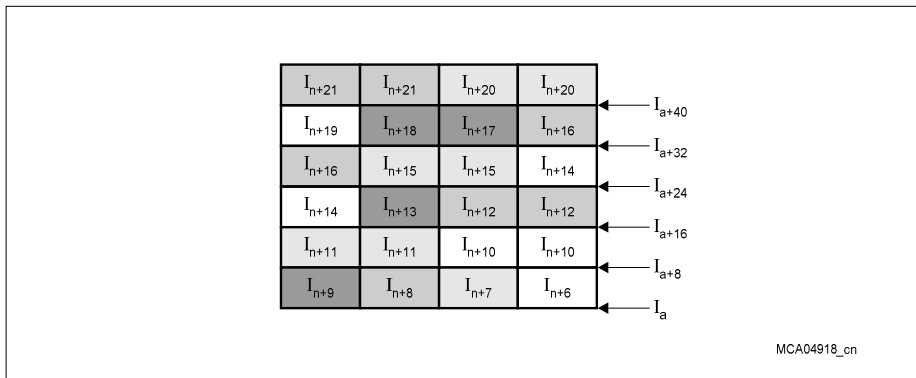


图 4-3 预测正确的指令流对应的程序存储器区段

4.2.3 预测错误的指令流

如果 CPU 检测到 IFU 对指令流预测错误，那么流水线各级将被取消，包含错误预取指令的指令 FIFO 被清空。整个取指过程将从程序的正确位置重新开始。

假设程序存储器访问无需等待状态¹⁾（0 等待状态），表 4-3 给出重新开始执行指令的过程。图 4-4 给出相应的程序存储器区段。

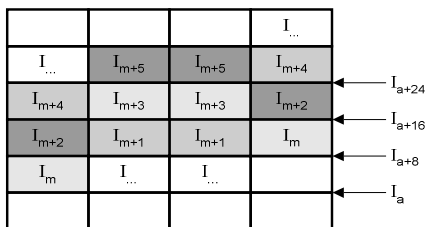
在周期 T_n ，CPU 检测到错误的预测，它将取消流水线操作。在 T_{n+1} ，新地址送给 PMU，在 T_{n+2} 送出第一个数据。但是由于目标指令跨越 64 位存储边界，因此需要在 T_{n+3} 内再次取指，从而得到完整的 32 位指令。在 T_{n+4} ，预取缓存中保存了两个 32 位指令，而第一条指令 I_m 被直接送入解码阶段。

此时，指令预取单元重新开始工作，预取更多的指令。在 T_{n+5} ，指令 I_{m+1} 从取指缓存直接进入解码阶段。取指令队列给出取指缓存中的所有指令以及从指令 FIFO 中读取的指令。在 T_{n+6} ，指令 I_{m+3} 是第一条从 FIFO 中取出的指令。在同一个周期内，指令 I_{m+2} 仍然从取指缓存直接送入解码阶段。

1) 访问 Flash 存储器可能需要插入等待状态，这取决于实际的工作频率。

表 4-3 预测错误的指令流（重新启动执行）

	T_n	T_{n+1}	T_{n+2}	T_{n+3}	T_{n+4}	T_{n+5}	T_{n+6}	T_{n+7}	T_{n+8}
PMU 地址	I_{\dots}	I_a	I_{a+8}	I_{a+16}	I_{a+24}	I_{\dots}	I_{\dots}	I_{\dots}	I_{\dots}
PMU 数据 64 位	I_{\dots}	—	I_d	I_{d+1}	I_{d+2}	I_{d+3}	I_{\dots}	I_{\dots}	I_{\dots}
预取指令 96 位缓存	I_{\dots}	—	—	—	I_m I_{m+1}	I_{m+2} I_{m+3}	I_{m+4} I_{m+5}	I_{\dots}	I_{\dots}
取指 指令缓存	I_{next+2}	—	—	—	—	I_{m+1}	I_{m+2} I_{m+3}	I_{m+4} I_{m+5}	I_{\dots}
从 FIFO 中 取指	—	—	—	—	—	—	I_{m+3}	I_{m+4}	I_{m+5}
解码	I_{next+1}	—	—	—	I_m	I_{m+1}	I_{m+2}	I_{m+3}	I_{m+4}
寻址	I_{next}	—	—	—	—	I_m	I_{m+1}	I_{m+2}	I_{m+3}
存储	I_{branch}	—	—	—	—	—	I_m	I_{m+1}	I_{m+2}
执行	I_n	I_{branch}	—	—	—	—	—	I_m	I_{m+1}
回写	—	I_n	I_{branch}	—	—	—	—	—	I_m



64位宽（4*16）程序存储器

MCA04919_cn

图 4-4 预测错误的指令流对应的程序存储器区段

4.3 指令处理流水线

XC164CM 采用 5 级流水执行每条指令。所有指令都要依次通过指令处理流水线的每一阶段。下面给出指令处理流水线以及 2 级取指流水线的各阶段描述：

第 1 阶段 → 预取指令：该阶段按照预测的顺序从 PMU 预取指令。分支检测单元对这些指令进行预处理，检测分支指令。预测逻辑决定是否执行这些分支/跳转指令。

第 2 阶段 → 取指令：根据分支预测规则，计算出要读取的下一条指令的指针。为了实现零周期的分支指令，分支折叠单元对检测到的分支指令进行预处理，并将其与前一条指令组合。预取的指令保存在指令 FIFO 中。与此同时，从指令 FIFO 中取出的指令送入指令流水线进行处理。

第 3 阶段 → 解码：指令被解码，如有需要，从寄存器文件中读取用于间接寻址的 GPR。

第 4 阶段 → 寻址：计算所有操作数的地址。指令访问系统堆栈时，SP（堆栈指针）自动递增或者递减。

第 5 阶段 → 存储：读取所有需要的操作数。

第 6 阶段 → 执行：对读取的操作数进行 ALU 或 MAC 运算，并更新状态标志。对 CPU-SFR 执行写操作，用作间接寻址指针的 GPR 自动递增/递减。

第 7 阶段 → 回写：所有的外部操作数和内部 DPRAM 中剩余的操作数被写回。内部 SRAM 中的操作数写入回写缓存中。

对于超过一个 CPU 时钟周期的指令，内部会产生一些特定的插入指令为其提供时间。它们被自动插入到流水线的解码阶段，然后像其它标准指令一样经过流水线的其余阶段。还可以利用指令插入来处理程序中断、PEC 传送和 OCE 操作。尽管实际上不需要注意这些内部插入指令，但它们的确有助于流水线的操作。

带宽的限制（不同阶段访问相同的资源）以及指令之间数据的相依性都会降低 CPU（流水线）的性能。XC164CM CPU 具有专用硬件来检测并解决各种类型的数据相依性问题。后续章节将对一些数据相依问题详细说明。

由于 XC164CM 的 CPU 最多可以同时处理五条不同的指令，因此需要使用附加的专用硬件来解决可能存在于不同流水线阶段之间的数据相依问题。该硬件支持操作数读写值的“前送”，可以用时间上最优、且不损失性能的方式解决大多数可能的冲突。不过，也有极少数的情况，程序员需要注意流水线，使用流水线冲突造成的延迟执行其它指令，从而优化 CPU 性能。

注：XC164CM 有一个完全互锁的流水线，这意味着这些冲突不会导致系统出错。只有考虑到性能因素时才需要对指令进行重新排序。

下面的例子给出特殊情况下的流水线操作，并给出通过指令重排提高指令执行性能的基本规则。

4.3.1 通用寄存器引起的流水线冲突

GPR（通用寄存器）是 CPU 的工作寄存器，使用 GPR 的指令之间可能存在很多的数据相依性。XC164CM 采用一个高速五端口的寄存器文件来防止带宽冲突，并使用专用硬件来检测和解决数据相依性。通过特殊的前送总线将 GPR 的值从流水线的的一个阶段送至另一个阶段。大多数情况下，即使存在数据相依性，仍然可以无任何延迟的执行指令。

Conflict_GPRs_Resolved:

I_n ADD R0, R1 ; 计算 R0 的新值
 I_{n+1} ADD R3, R0 ; 再次使用 R0
 I_{n+2} ADD R6, R0 ; 再次使用 R0
 I_{n+3} ADD R6, R1 ; 再次使用 R6
 I_{n+4} ...

表 4-4 解决使用 GPR 引起的流水线数据相依性

阶段	T_n	T_{n+1}	T_{n+2}	$T_{n+3}^{1)}$	$T_{n+4}^{2)}$	$T_{n+5}^{3)}$
解码	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R3, R0}$	$I_{n+2} = \text{ADD R6, R0}$	$I_{n+3} = \text{ADD R6, R1}$	I_{n+4}	I_{n+5}
寻址	I_{n-1}	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R3, R0}$	$I_{n+2} = \text{ADD R6, R0}$	$I_{n+3} = \text{ADD R6, R1}$	I_{n+4}
存储	I_{n-2}	I_{n-1}	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R3, R0}$	$I_{n+2} = \text{ADD R6, R0}$	$I_{n+3} = \text{ADD R6, R1}$
执行	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R3, R0}$	$I_{n+2} = \text{ADD R6, R0}$
回写	I_{n-4}	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R3, R0}$

- 1) 将 R0 从执行阶段前送至存储阶段。
- 2) 将 R0 从回写阶段前送至存储阶段。
- 3) 将 R6 从执行阶段前送至存储阶段。

但是，如果 GPR 用于间接寻址，在解码阶段就已经需要使用地址指针（即 GPR）。这种情况下，该指令会一直停滞在寻址阶段，直到执行完 ALU 操作且将运算结果前送至寻址阶段。

Conflict_GPRs_Pointer_Stall:

I_n ADD R0, R1 ; 计算 R0 的新值
 I_{n+1} MOV R3, [R0] ; 使用 R0 作为地址指针
 I_{n+2} ADD R6, R0
 I_{n+3} ADD R6, R1
 I_{n+4} ...

表 4-5 GPR 用作指针时引起的流水线数据相依性（停滞）

阶段	T_n	T_{n+1}	T_{n+2} ¹⁾	T_{n+3} ²⁾	T_{n+4}	T_{n+5}
解码	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{MOV R3, [R0]}$	I_{n+2}	I_{n+2}	I_{n+2}	I_{n+3}
寻址	I_{n-1}	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{MOV R3, [R0]}$	$I_{n+1} = \text{MOV R3, [R0]}$	$I_{n+1} = \text{MOV R3, [R0]}$	I_{n+2}
存储	I_{n-2}	I_{n-1}	$I_n = \text{ADD R0, R1}$	—	—	$I_{n+1} = \text{MOV R3, [R0]}$
执行	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{ADD R0, R1}$	—	—
回写	I_{n-4}	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{ADD R0, R1}$	—

1) R0 的新值还不可用。

2) R0 从执行阶段前送至寻址阶段（下一周期）。

为了避免指令停滞，可以插入一个多周期指令或者两个单周期指令。这些插入的指令绝对不能更新用于间接寻址的 GPR。

Conflict_GPRs_Pointer_NoStall:

I_n ADD R0, R1 ; 计算 R0 的新值
 I_{n+1} ADD R6, R0 ; R0 未更新，只被读取
 I_{n+2} ADD R6, R1
 I_{n+3} MOV R3, [R0] ; 使用 R0 作为地址指针
 I_{n+4} ...

表 4-6 GPR 用作指针时引起的流水线数据相依性（不停滞）

阶段	T_n	T_{n+1}	T_{n+2}	$T_{n+3}^{1)}$	T_{n+4}	T_{n+5}
解码	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R6, R0}$	$I_{n+2} = \text{ADD R6, R1}$	$I_{n+3} = \text{MOV R3, [R0]}$	I_{n+4}	I_{n+5}
寻址	I_{n-1}	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R6, R0}$	$I_{n+2} = \text{ADD R6, R1}$	$I_{n+3} = \text{MOV R3, [R0]}$	I_{n+4}
存储	I_{n-2}	I_{n-1}	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R6, R0}$	$I_{n+2} = \text{ADD R6, R1}$	$I_{n+3} = \text{MOV R3, [R0]}$
执行	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R6, R0}$	$I_{n+2} = \text{ADD R6, R1}$
回写	I_{n-4}	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{ADD R0, R1}$	$I_{n+1} = \text{ADD R6, R0}$

1) R0 从执行阶段前送至寻址阶段（下一周期）。

4.3.2 间接寻址模式引起的流水线冲突

使用间接寻址模式时，地址产生单元采用一种推测寻址机制。在对地址解码之前，根据历史记录表，选择从哪种存储器区域（DPRAM、DSRAM 等）读取数据。该历史记录表为每个 GPR 提供了一个入口，其中保存着最近一次通过相应 GPR 寻址的存储区的信息。如果对存储区预测错误，必须重新开始进行读访问。

建议用户在使用 GPR 进行间接寻址时，应使其始终指向同一个存储器区域。如果更新后的 GPR 指向不同的存储器区域，那么下次的读操作将会访问错误的存储器区域，从而必须重新进行读操作，这会导致流水线停滞。

Conflict_GPRs_Pointer_WrongHistory:

```

In   ADD R3, [R0]    ; 例如 R0 指向 DPRAM
In+1 MOV R0, R4
...
Ii   MOV DPPX, ...   ; 改变 DPPx
...
Im   ADD R6, [R0]    ; 例如 R0 现在指向 SRAM
Im+1 MOV R6, R1
Im+2 ...
    
```

表 4-7 使用指针引起的流水线数据相依性（有效推测）

阶段	T _n	T _{n+1}	T _{n+2}	T _{n+3}	T _{n+4}	T _{n+5}
解码	I _n = ADD R3, [R0]	I _{n+1} = MOV R0, R4	I _{n+2}	I _{n+3}	I _{n+4}	I _{n+5}
寻址	I _{n-1}	I _n = ADD R3, [R0]	I _{n+1} = MOV R0, R4	I _{n+2}	I _{n+3}	I _{n+4}
存储	I _{n-2}	I _{n-1}	I _n = ADD R3, [R0]	I _{n+1} = MOV R0, R4	I _{n+2}	I _{n+3}
执行	I _{n-3}	I _{n-2}	I _{n-1}	I _n = ADD R3, [R0]	I _{n+1} = MOV R0, R4	I _{n+2}
回写	I _{n-4}	I _{n-3}	I _{n-2}	I _{n-1}	I _n = ADD R3, [R0]	I _{n+1} = MOV R0, R4

表 4-8 使用指针引起的流水线数据相依性（无效推测）

阶段	T_m	T_{m+1}	T_{m+2} ¹⁾	T_{m+3}	T_{m+4}	T_{m+5}
解码	$I_m = \text{ADD R6, [R0]}$	$I_{m+1} = \text{MOV R6, R1}$	$I_{m+1} = \text{MOV R6, R1}$	I_{m+2}	I_{m+3}	I_{m+4}
寻址	I_{m-1}	$I_m = \text{ADD R6, [R0]}$	$I_m = \text{ADD R6, [R0]}$	$I_{m+1} = \text{MOV R6, R1}$	I_{m+2}	I_{m+3}
存储	I_{m-2}	I_{m-1}	—	$I_m = \text{ADD R6, [R0]}$	$I_{m+1} = \text{MOV R6, R1}$	I_{m+2}
执行	I_{m-3}	I_{m-2}	I_{m-1}	—	$I_m = \text{ADD R6, [R0]}$	$I_{m+1} = \text{MOV R6, R1}$
写回	I_{m-4}	I_{m-3}	I_{m-2}	I_{m-1}	—	$I_m = \text{ADD R6, [R0]}$

1) 因为历史记录错误（目标区域已经改变），必须重复访问地址 [R0]。

4.3.3 存储器带宽引起的流水线冲突

如果流水线中的指令同时访问同一存储器区域时，可能会出现存储器带宽冲突。XC164CM 采用了一种特殊的访问机制最大程度的减少冲突。CPU 的 DPRAM 有两个独立的读/写端口，能够无延迟、并行进行读写操作。对 DSRAM 的写操作可以先保存在回写缓存中，直至完成读操作为止。

除 CoXXX 指令以外的所有指令，每个周期只能读一个存储器操作数。因为 DPRAM 有两个独立的读/写端口，读操作和写操作之间不会出现冲突。只有其它的流水线停滞条件才会产生 DPRAM 带宽冲突。DPRAM 是一个同步流水式存储器。寻址阶段的目标地址有效后开始执行读访问，在存储阶段将读取的数据送出。如果在存储阶段，存储器读访问被停滞、处于寻址阶段的随后指令又要开始新的存储器读操作，那么该读访问同样必须延迟。但是，该冲突被现有的流水线停滞所掩盖。

CoXXX 指令是唯一能够在一个周期读取两个存储器操作数的指令。如果三个操作数都在 DPRAM 中，那么两个读操作和一个挂起的写操作之间会出现带宽冲突。在执行滤波器程序时，这种情况对性能影响特别显著。其中的一个操作数应该放在 DSRAM 中，以确保 CoXXX 指令的单周期正确执行。

Conflict_DPRAM_Bandwidth:

```

In    ADD op1, R1
In+1  ADD R6, R0
In+2  CoMAC [IDX0], [R0]
In+3  MOV R3, [R0]
In+4  ...
    
```

表 4-9 存储器（DPRAM）带宽冲突引起的流水线数据相依性

阶段	T _n	T _{n+1}	T _{n+2}	T _{n+3}	T _{n+4} ¹⁾	T _{n+5}
解码	I _n = ADD op1, R1	I _{n+1} = ADD R6, R0	I _{n+2} = CoMAC ...	I _{n+3} = MOV R3, [R0]	I _{n+4}	I _{n+4}
寻址	I _{n-1}	I _n = ADD op1, R1	I _{n+1} = ADD R6, R0	I _{n+2} = CoMAC ...	I _{n+3} = MOV R3, [R0]	I _{n+3} = MOV R3, [R0]
存储	I _{n-2}	I _{n-1}	I _n = ADD op1, R1	I _{n+1} = ADD R6, R0	I _{n+2} = CoMAC ...	I _{n+2} = CoMAC ...
执行	I _{n-3}	I _{n-2}	I _{n-1}	I _n = ADD op1, R1	I _{n+1} = ADD R6, R0	—
回写	I _{n-4}	I _{n-3}	I _{n-2}	I _{n-1}	I _n = ADD op1, R1	I _{n+1} = ADD R6, R0

1) CoMAC 指令由于存储器带宽冲突而停滞。

DSRAM 是一个单端口读/写存储器。为了减少带宽冲突，XC164CM 中使用回写缓存。它有三个数据入口。仅在回写缓存已满、且同时发生读写访问的情况下，当某个缓存入口被回写时，才必须停滞读访问。

Conflict_DSRAM_Bandwidth:

I_n ADD op1, R1
 I_{n+1} ADD R6, R0
 I_{n+2} ADD R6, op2
 I_{n+3} MOV R3, R2
 I_{n+4} ...

表 4-10 存储器（DSRAM）带宽冲突引起的流水线数据相依性

阶段	T_n	T_{n+1}	T_{n+2}	T_{n+3}	T_{n+4} ¹⁾	T_{n+5}
解码	$I_n = \text{ADD op1, R1}$	$I_{n+1} = \text{ADD R6, R0}$	$I_{n+2} = \text{ADD R6, op2}$	$I_{n+3} = \text{MOV R3, R2}$	I_{n+4}	I_{n+4}
寻址	I_{n-1}	$I_n = \text{ADD op1, R1}$	$I_{n+1} = \text{ADD R6, R0}$	$I_{n+2} = \text{ADD R6, op2}$	$I_{n+3} = \text{MOV R3, R2}$	$I_{n+3} = \text{MOV R3, R2}$
存储	I_{n-2}	I_{n-1}	$I_n = \text{ADD op1, R1}$	$I_{n+1} = \text{ADD R6, R0}$	$I_{n+2} = \text{ADD R6, op2}$	$I_{n+2} = \text{ADD R6, op2}$
执行	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{ADD op1, R1}$	$I_{n+1} = \text{ADD R6, R0}$	—
回写	I_{n-4}	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{ADD op1, R1}$	$I_{n+1} = \text{ADD R6, R0}$
回写缓存	充满	充满	充满	充满	充满	充满

1) ADD R6, op2 指令由于存储器带宽冲突而停滞。

4.3.4 CPU-SFR 更新引起的流水线冲突

CPU-SFR 控制 CPU 的功能和行为。对 CSFR 的修改和更新会影响流水线中的指令流。因此，用户必须特别留意保证流水线中的指令使用正确的 CSFR 值。在流水线**执行**阶段后期更新 CSFR。此时，如果没有冲突检测，处于**解码、寻址、存储**阶段的指令将仍旧使用未更新的寄存器值。为了保证操作正确，CPU 会检测冲突情况并停滞流水线。为了提高性能，CPU 区分不同类型的 CPU-SFR。根据以下规则对指令进行重新排序，可以改善流水线中指令流的性能。

共有三种类型的 CPU-SFR：

- 不产生流水线冲突的 CSFR（ONES、ZEROS、MCW）
- 在执行阶段后期更新的 CSFR 结果寄存器，会导致一个周期的停滞
- 影响整个 CPU 或流水线的 CSFR，会导致操作被取消

CSFR 结果寄存器

ALU 和 MAC 单元的 CSFR 结果寄存器 MDH、MDL、MSW、MAH、MAL 和 MRW 在流水线执行阶段的后期被更新。如果处于存储阶段的指令（CoSTORE 除外）要访问这些寄存器，寄存器值不能被前送。该指令必须在存储阶段停滞一个周期。

Conflict_CSFR_Update_Stall:

I_n MUL R0, R1
 I_{n+1} MOV R6, MDL
 I_{n+2} ADD R6, R1
 I_{n+3} MOV R3, [R0]
 I_{n+4} ...

表 4-11 结果 CSFR 引起的流水线数据相依性（停滞）

阶段	T_n	T_{n+1}	T_{n+2}	T_{n+3} ¹⁾	T_{n+4}	T_{n+5}
解码	$I_n = \text{MUL R0, R1}$	$I_{n+1} = \text{MOV R6, MDL}$	$I_{n+2} = \text{ADD R6, R1}$	$I_{n+3} = \text{MOV R3, [R0]}$	$I_{n+3} = \text{MOV R3, [R0]}$	I_{n+4}
寻址	I_{n-1}	$I_n = \text{MUL R0, R1}$	$I_{n+1} = \text{MOV R6, MDL}$	$I_{n+2} = \text{ADD R6, R1}$	$I_{n+2} = \text{ADD R6, R1}$	$I_{n+3} = \text{MOV R3, [R0]}$
存储	I_{n-2}	I_{n-1}	$I_n = \text{MUL R0, R1}$	$I_{n+1} = \text{MOV R6, MDL}$	$I_{n+1} = \text{MOV R6, MDL}$	$I_{n+2} = \text{ADD R6, R1}$
执行	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{MUL R0, R1}$	—	$I_{n+1} = \text{MOV R6, MDL}$
回写	I_{n-4}	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{MUL R0, R1}$	—

1) 此处不能读取 MDL。

通过重排指令，可以将不占用该资源的指令填充到流水线的空隙周期中。

Conflict_CSFR_Update_Resolved:

```

In      MUL R0, R1
In+1    MOV R3, [R0]
In+2    MOV R6, MDL
In+3    ADD R6, R1
In+4    ...
    
```

表 4-12 结果 CSFR 引起的流水线数据相依性（无停滞）

阶段	T _n	T _{n+1}	T _{n+2}	T _{n+3}	T _{n+4} ¹⁾	T _{n+5}
解码	I _n = MUL R0, R1	I _{n+1} = MOV R3, [R0]	I _{n+2} = MOV R6, MDL	I _{n+3} = ADD R6, R1	I _{n+4}	I _{n+5}
寻址	I _{n-1}	I _n = MUL R0, R1	I _{n+1} = MOV R3, [R0]	I _{n+2} = MOV R6, MDL	I _{n+3} = ADD R6, R1	I _{n+4}
存储	I _{n-2}	I _{n-1}	I _n = MUL R0, R1	I _{n+1} = MOV R3, [R0]	I _{n+2} = MOV R6, MDL	I _{n+3} = ADD R6, R1
执行	I _{n-3}	I _{n-2}	I _{n-1}	I _n = MUL R0, R1	I _{n+1} = MOV R3, [R0]	I _{n+2} = MOV R6, MDL
回写	I _{n-4}	I _{n-3}	I _{n-2}	I _{n-1}	I _n = MUL R0, R1	I _{n+1} = MOV R3, [R0]

1) 此时MDL可被读取，无需停滞。

影响整个 CPU 的 CSFR

某些 CSFR 会影响整个 CPU 或存储器操作阶段之前流水线的操作。CPU-SFR CPUCON1、CP、SP、STKUN、STKOV、VECSEG、TFR 和 PSW 被用户修改时，会影响整个 CPU 的功能；CPU-SFR IDX0、IDX1、QX1、QX0、DPP0、DPP1、DPP2 和 DPP3 在被用户修改时，只影响解码、寻址和存储阶段。这种情况下，流水线行为取决于用于修改 CSFR 的指令和寻址模式。

通过“POP CSFR”或使用 `reg, #data16` 寻址模式的指令修改这些 CSFR 寄存器时，XC164CM 在初始化阶段通过一种特殊机制来提高性能。

为了便于进一步解释，修改 CSFR 的指令可被称作“CSFR 修改指令”。“CSFR 修改指令”进入处理流水线后，在解码阶段被检测到。下面列出的指令在解码阶段被保持（所有其它指令不被保持）：

- 使用长寻址模式（mem）的指令
- 使用间接寻址模式（[Rw], [Rw+]...）的指令、JMPI 和 CALLI 除外
- ENWDT、DISWDT、EINIT
- 所有 CoXXX 指令

如果 CPUCON1、CP、SP、STKUN、STKOV、VECSEG、TFR 或 PSW 被修改，并且“CSFR 修改指令”进入执行阶段，那么流水线将被取消。这种修改将影响整个流水线和指令预取，因此需要通过彻底取消和重启机制来确保正确的指令流。如果 IDX0、IDX1、QX1、QX0、DPP0、DPP1、DPP2 或 DPP3 被修改，只有解码、寻址和存储阶段受到影响，不需取消流水线。这种修改不影响已经进入寻址、存储阶段的指令，因为这些指令当时不使用这些资源。其它类型的指令在解码阶段被保持，直到 CSFR 被修改为止。

下面的例子给出流水线停滞的一种情况。指令“MOV IDX1, #12”修改 CSFR，其后的指令“MOV R6, mem”将在解码阶段被保持，直到 IDX1 寄存器被更新。第二个例子给出一个优化的初始化程序。

Conflict_Canceling:

I_n MOV IDX1, #12

I_{n+1} MOV R6, mem

I_{n+2} ADD R6, R1

I_{n+3} MOV R3, [R0]

表 4-13 控制 CSFR 引起的流水线数据相依性（取消流水线）

阶段	T_n	T_{n+1}	T_{n+2}	T_{n+3}	T_{n+4}	T_{n+5}
解码	$I_n = \text{MOV}$ IDX1, #12	$I_{n+1} = \text{MOV}$ R6, mem	$I_{n+1} = \text{MOV}$ R6, mem	$I_{n+1} = \text{MOV}$ R6, mem	$I_{n+1} = \text{MOV}$ R6, mem	$I_{n+2} = \text{ADD}$ R6, R1
寻址	I_{n-1}	$I_n = \text{MOV}$ IDX1, #12	—	—	—	$I_{n+1} = \text{MOV}$ R6, mem
存储	I_{n-2}	I_{n-1}	$I_n = \text{MOV}$ IDX1, #12	—	—	—
执行	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{MOV}$ IDX1, #12	—	—
回写	I_{n-4}	I_{n-3}	I_{n-2}	I_{n-1}	$I_n = \text{MOV}$ IDX1, #12	—

Conflict_Canceling_Optimized:

```

In      MOV IDX1, #12
In+1    MOV MAH, #23
In+2    MOV MAL, #25
In+3    MOV R3, #08
In+4    ...
    
```

表 4-14 控制 CSFR 引起的流水线数据相依性（经过优化）

阶段	T _n	T _{n+1}	T _{n+2}	T _{n+3}	T _{n+4}	T _{n+5}
解码	I _n = MOV IDX1, #12	I _{n+1} = MOV MAH, #23	I _{n+2} = MOV MAL, #25	I _{n+3} = MOV R3, #08	I _{n+4}	I _{n+5}
寻址	I _{n-1}	I _n = MOV IDX1, #12	I _{n+1} = MOV MAH, #23	I _{n+2} = MOV MAL, #25	I _{n+3} = MOV R3, #08	I _{n+4}
存储	I _{n-2}	I _{n-1}	I _n = MOV IDX1, #12	I _{n+1} = MOV MAH, #23	I _{n+2} = MOV MAL, #25	I _{n+3} = MOV R3, #08
执行	I _{n-3}	I _{n-2}	I _{n-1}	I _n = MOV IDX1, #12	I _{n+1} = MOV MAH, #23	I _{n+2} = MOV MAL, #25
回写	I _{n-4}	I _{n-3}	I _{n-2}	I _{n-1}	I _n = MOV IDX1, #12	I _{n+1} = MOV MAH, #23

对于所有其它修改这些 CSFR（影响整个 CPU 的 CSFR）的指令，采用简单的停滞和取消机制来保证正确的指令流。

在流水线的存储阶段，用户对这类 CSFR（影响整个 CPU 的 CSFR）进行的修改可被检测到。随后的指令在寻址和解码阶段被停滞。如果指令进入指令执行阶段，那么整个流水线和 IFU 中的指令 FIFO 被清空，指令流需要彻底重新开始。

Conflict_Canceling_Completely:

```

In      MOV PSW, R4
In+1    MOV R6, R1
In+2    ADD R6, R1
In+3    MOV R3, [R0]
In+4    ...
    
```

表 4-15 控制 CSFR 引起的流水线数据相依性（全部取消）

阶段	T _{n+1}	T _{n+2}	T _{n+3}	T _{n+4}	T _{n+5}	T _{n+6}
解码	I _{n+1} = MOV R6, R1	I _{n+2} = ADD R6, R1	I _{n+2} = ADD R6, R1	—	—	I _{n+1} = MOV R6, R1
寻址	I _n = MOV PSW, R4	I _{n+1} = MOV R6, R1	I _{n+1} = MOV R6, R1	—	—	—
存储	I _{n-1}	I _n = MOV PSW, R4	—	—	—	—
执行	I _{n-2}	I _{n-1}	I _n = MOV PSW, R4	—	—	—
回写	I _{n-3}	I _{n-2}	I _{n-1}	I _n = MOV PSW, R4	—	—

4.4 CPU 配置寄存器

CPU 配置寄存器用于选择 XC164CM CPU 内核的一般特性和功能。通常来说，禁止应用程序修改这些寄存器（特殊情况将在勘误表中声明）。

注：执行 *EINIT* 指令之后，CPU 配置寄存器被寄存器安全机制保护。

CPUCON1

CPU 控制寄存器 1

SFR (FE18H/0CH)

复位值：0007_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	VECSC	WDT CTL	SGT DIS	INTS CXT	BP	ZCJ	
-	-	-	-	-	-	-	-	-	rw	rw	rw	rw	rw	rw	

符号	位序号	读写类型	功能描述
VECSC	[6:5]	rw	中断向量间距 00 中断向量之间的间距为 2 个字 ¹⁾ 01 中断向量之间的间距为 4 个字 10 中断向量之间的间距为 8 个字 11 中断向量之间的间距为 16 个字
WDTCTL	4	rw	看门狗定时器配置 0 仅在初始化结束 ²⁾ 后才可执行 DISWDT 1 始终可以执行 DISWDT/ENWDT（增强 WDT 模式）
SGTDIS	3	rw	分段禁止/使能控制 0 分段使能 1 分段禁止
INTSCXT	2	rw	切换上下文的中断使能 0 切换上下文不可被中断 1 切换上下文可被中断
BP	1	rw	分支预测单元使能 0 禁止分支预测 1 使能分支预测

符号	位序号	读写类型	功能描述
ZCJ	0	rw	零周期跳转使能 0 禁止零周期跳转功能 1 使能零周期跳转功能

- 1) 该缺省值（2 个字）与 C166 系列体系结构中的向量间距兼容。
2) DISWDT（EINIT 指令之后执行）和 ENWDT 指令被内部转变成一个 NOP 指令。

CPUCON2

CPU 控制寄存器 2

SFR（FE1AH/0DH）

复位值：8FBB_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FIFODEPTH				FIFOFED	BYP PF	BYP F	EIO IAEN	STE N	LFIC	OV RUN	RET ST	-	DAID	SL	
rw				rw	rw	rw	rw	rw	rw	rw	rw	-	rw	rw	

符号	位序号	读写类型	功能描述
FIFODEPTH	[15:12]	rw	FIFO 缓存设置 0000 没有 FIFO 0001 1 级 FIFO 1000 8 级 FIFO 1001 保留 1111 保留
FIFOFED	[11:10]	rw	FIFO 填充配置 00 禁止 FIFO 01 每个周期 FIFO 中至多填充 1 条指令 10 每个周期 FIFO 中至多填充 2 条指令 11 每个周期 FIFO 中至多填充 3 条指令
BYP PF	9	rw	预取指旁路控制 0 从预取指到解码之间的旁路通道被禁止 1 从预取指到解码之间的旁路通道被使能
BYP F	8	rw	取指旁路控制

符号	位序号	读写类型	功能描述
			0 从取指到解码之间的旁路通道被禁止 1 从取指到解码之间的旁路通道被使能
EIOIAEN	7	rw	早期 IO 插入应答使能 0 不保证通过破坏性的读取进行插入应答 1 保证通过破坏性的读取进行插入应答
STEN	6	rw	停滞指令使能（用于调试） 0 禁用停滞指令 1 使能停滞指令（详见下面的例子）
LFIC	5	rw	线性跟随器指令缓存 0 禁用线性跟随器指令缓存 1 使能线性跟随器指令缓存
OVRUN	4	rw	流水线控制 0 不允许过载流水线空隙周期 1 允许过载流水线空隙周期
RETST	3	rw	返回堆栈使能 0 禁止返回堆栈 1 使能返回堆栈
DAID	1	rw	Atomic 期间的插入请求控制 0 Atomic 期间，拒绝插入请求 1 Atomic 期间，接受插入请求
SL	0	rw	短循环模式使能 0 禁止短循环模式 1 使能短循环模式

专用的停滞调试指令举例：

STALLAM da, ha, dm, hm ; 操作码: 44 dahadmhm

STALLEW de, he, dw, hw ; 操作码: 45 dehedwhw

; 相应的流水线阶段在“d”周期之后停滞“h”个周期

; (“d”和“h”为 6 位值)

注：一般情况下，这些寄存器不能由应用程序修改（特殊情况在勘误表中声明）。

4.5 通用寄存器的使用

CPU 有多个寄存器组，每个寄存器组由 16 个寄存器 R0、R1、R2、...R15 组成，通常称之为通用寄存器（GPR）。访问这些寄存器只占用一个 CPU 周期。GPR 是算术逻辑运算单元的工作寄存器，也可用作间接寻址的地址指针。

通过 5 端口寄存器文件访问这些寄存器组，可实现对寄存器组的高速访问，满足 CPU 性能要求。寄存器文件分成三个独立的物理寄存器组，对应**两种类型**：

- **两个局部寄存器组**，是寄存器文件的一部分。
- **一个全局寄存器组**，映射到存储器，缓存在寄存器文件中。

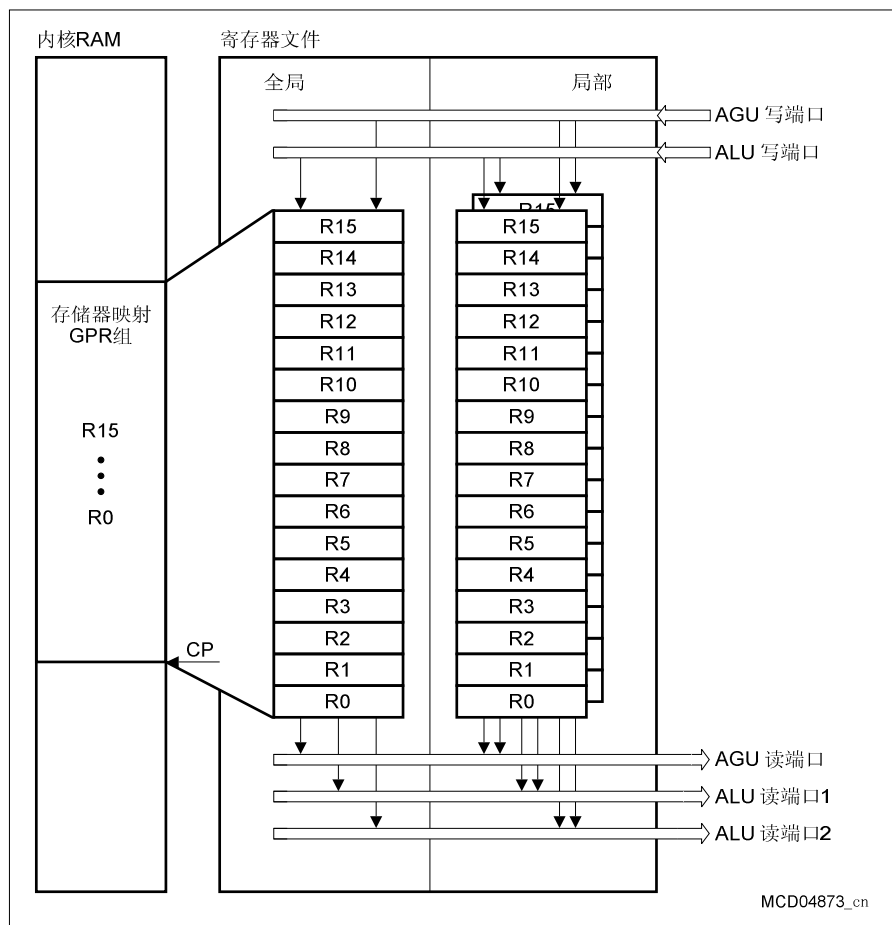


图 4-5 寄存器文件

由 PSW 寄存器中的位域 BANK 选择激活哪一个物理寄存器组。用户可通过修改 PSW 的指令改变所选寄存器组；或通过 RETI 指令、中断或硬件强制中断改变所选寄存器组。发生中断时，通过中断控制器（ITC）中的寄存器 BNKSELx 来选择寄存器组。硬件强制中断始终使用全局寄存器组。

局部寄存器组由专用的物理寄存器组成，全局寄存器组表现为一个缓存。映射到存储器的 GPR 组（全局寄存器组）位于内部 DPRAM 中。每个寄存器组占用 16 个字长的连续地址区段。上下文指针（CP）寄存器决定当前所选寄存器组的基地址。为保证访问速度，DPRAM 中的 GPR 被缓存到 5 端口寄存器文件中（同一时刻，只能缓存一个存储器映射 GPR 组）。如果激活全局寄存器组，在执行后面的指令前，将对缓存进行验证。验证之后，所有对 GPR 的进一步访问会转向全局寄存器组。

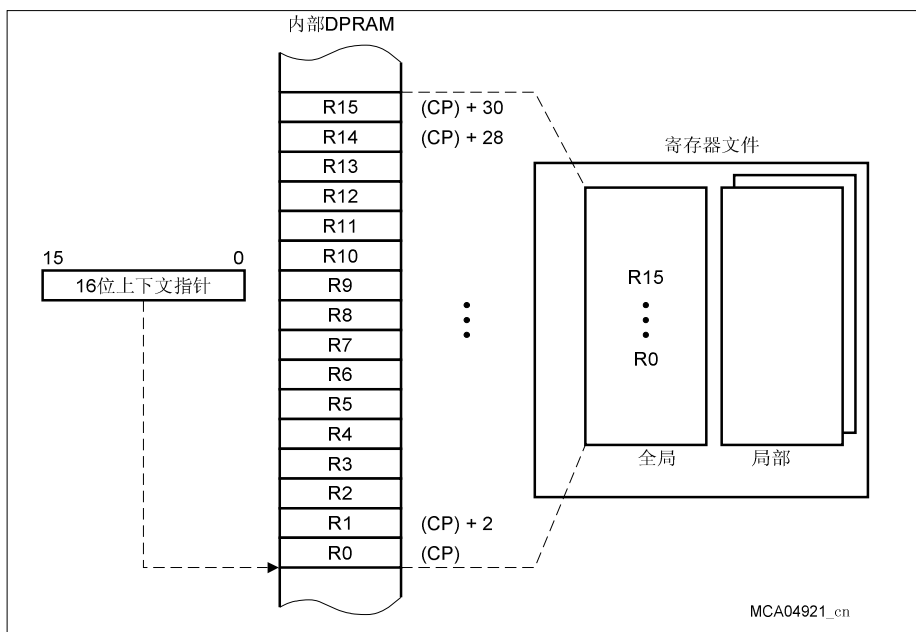


图 4-6 通过寄存器 CP 选择全局寄存器组

4.5.1 GPR 寻址模式

GPR 是工作寄存器，会被频繁地访问。因此有三种可能的方式来访问寄存器组：

- GPR 短寻址（助记符：Rw 或 Rb）
- 寄存器短寻址（助记符：reg 或 bitoff）
- 存储器长寻址（助记符：mem），只适用于全局寄存器组

GPR 短寻址指定当前寄存器组（通过 BANK 位域选择）内的偏移地址。4 位 GPR 短寻址可以访问所有 16 个寄存器，2 位 GPR 短寻址（部分指令采用）则能访问最低的 4 个寄存器。

根据寻址是字寻址（Rw）还是字节寻址（Rb），在物理访问寄存器组之前将 GPR 短地址乘 2（Rw），或者不乘 2（Rb）。从而 GPR 字访问和字节访问都可采用这种寻址方式。

注：用作间接寻址指针的 GPR 总是采用字操作的方式。

寻址局部寄存器组，结果偏移量可以直接使用；寻址全局寄存器组，结果偏移量要与寄存器 CP 的值逻辑相加。CP 指向当前全局寄存器组的基地址。（见图 4-7）。

8 位寄存器短寻址所用的地址范围从 F0_H 到 FF_H。与 4 位 GPR 短寻址方式相同，利用低四位来寻址，忽略高四位。GPR 物理地址的计算和 4 位短 GPR 寻址相同。对于 GPR 位操作，采用相同的方法来计算 GPR 的字地址。一个字中被访问位元的位置由单独的附加 4 位值来确定。

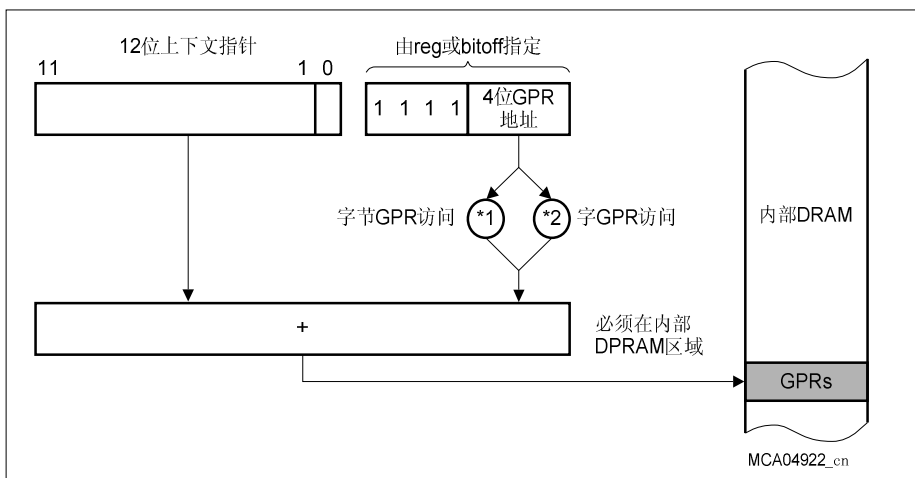


图 4-7 寄存器短寻址模式下 CP 的使用

24 位存储器寻址可直接用于访问位于 DPRAM 中的 GPR（不适用于局部寄存器组）。进行存储器读访问时，命中检测逻辑会检测被寻址的存储器内容是否缓存到全局寄存器组中。如果缓存命中，则开始进行全局寄存器组的读访问，使用从缓存中读取的数据，而忽略从存储器中读取的数据。这会导致一个 CPU 周期的延时（MOV R4, mem [CP ≤ mem ≤ CP+31]）。进行存储器写访问时，命中检测逻辑会提前决定是否缓存命中。不过，地址转换需要额外一个 CPU 周期。该值直接被写入全局寄存器组而无需额外的延迟（MOV mem, R4）。

注：不推荐使用 24 位 GPR 寻址模式，因为读和写访问都需要额外增加一个周期。

表 4-16 访问 GPR 的寻址模式

寄存器 ¹⁾		字节寄存器		短地址 ²⁾		
名称	存储器地址 ³⁾	名称	存储器地址 ³⁾	8 位	4 位	2 位
R0	(CP) + 0	RL0	(CP) + 0	F0 _H	0 _H	0 _H
R1	(CP) + 2	RH0	(CP) + 1	F1 _H	1 _H	1 _H
R2	(CP) + 4	RL1	(CP) + 2	F2 _H	2 _H	2 _H
R3	(CP) + 6	RH1	(CP) + 3	F3 _H	3 _H	3 _H
R4	(CP) + 8	RL2	(CP) + 4	F4 _H	4 _H	---
R5	(CP) + 10	RH2	(CP) + 5	F5 _H	5 _H	---
R6	(CP) + 12	RL3	(CP) + 6	F6 _H	6 _H	---
R7	(CP) + 14	RH3	(CP) + 7	F7 _H	7 _H	---
R8	(CP) + 16	RL4	(CP) + 8	F8 _H	8 _H	---
R9	(CP) + 18	RH4	(CP) + 9	F9 _H	9 _H	---
R10	(CP) + 20	RL5	(CP) + 10	FA _H	A _H	---
R11	(CP) + 22	RH5	(CP) + 11	FB _H	B _H	---
R12	(CP) + 24	RL6	(CP) + 12	FC _H	C _H	---
R13	(CP) + 26	RH6	(CP) + 13	FD _H	D _H	---
R14	(CP) + 28	RL7	(CP) + 14	FE _H	E _H	---
R15	(CP) + 30	RH7	(CP) + 15	FF _H	F _H	---

- 1) 前 8 个 GPR（R7...R0）也可以按字节访问。对 GPR 的一个字节进行写操作，不会影响该 GPR 的另一个字节。
- 2) 短寻址模式适用于所有寄存器组。
- 3) 长寻址模式只适用于映射到存储器的全局 GPR 组。

4.5.2 上下文切换

当操作系统的任务调度程序激活一个新任务，或者在调用或终止某个中断服务子程序时，剩余任务的工作上下文（即寄存器）必须被保存，新任务的工作上下文必须被恢复。可以用两种方式改变 CPU 上下文：

- 切换选用的寄存器组
- 切换全局寄存器的上下文

切换选用的物理寄存器组

通过更新寄存器 PSW 的位域 BANK，可以立即切换工作的寄存器组。可在当前存储器映射的全局寄存器组（缓存在全局寄存器组中）（BANK = 00_B）、局部寄存器组 1（BANK = 10_B）和局部寄存器组 2（BANK = 11_B）之间进行切换。

进行中断服务时，更新中断控制器中的寄存器 BNKSELx 位域 BANK，可以自动执行寄存器组切换。执行 RETI 指令，位域 BANK 将被自动恢复，上下文切换回原寄存器组。

还可以通过修改位域 BANK，在寄存器文件的三个物理寄存器组之间执行切换。由于流水线存在数据相依性，因此，若用户对寄存器 PSW 进行修改，必须取消流水线。

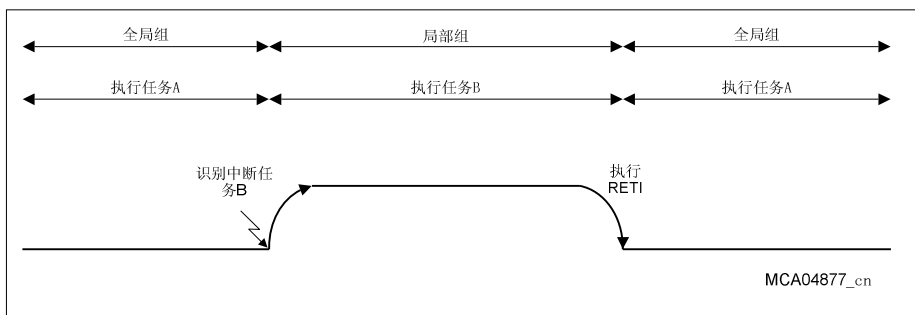


图 4-8 通过改变物理寄存器组进行上下文切换

切换至局部寄存器组之后，可以立即使用新的寄存器组；切换至全局寄存器组之后，在执行下一条指令前，缓存的存储器映射 GPR 必须有效。如果此时全局寄存器组无效（比如上下文切换过程被中断），将自动重复缓存验证过程。

全局寄存器组的上下文切换

通过改变存储器映射 GPR 组的基地址，切换全局寄存器组的内容。基地址由上下文指针（CP）给出。

完成 CP 更新之后，状态机开始保存全局寄存器组的旧内容并加载新内容。保存和加载算法需要 19 个 CPU 周期：其中缓存验证过程占用了 16 个周期；为了避免由完成验证过程而引起流水线冲突，指令执行需要再停滞 3 个周期。上下文切换过程包括两个阶段：

- **保存阶段：**通过插入 8 条 STORE 指令，将全局寄存器组的内容存回到 DPRAM 中。最后一条 STORE 指令之后，全局寄存器组的内容无效。
- **加载阶段：**通过插入 8 条 LOAD 指令，将新的上下文加载到全局寄存器组。最后一条 LOAD 指令之后，全局寄存器组的内容有效。

直到全局寄存器组再次有效之后，才可开始执行代码。在验证过程中可能出现硬件中断。如何完成验证过程取决于该中断所选择的寄存器组类型：

- 如果中断也使用全局寄存器组，在执行中断服务程序之前，需要完成验证过程（见图 4-9）。
- 如果中断使用局部寄存器组，验证过程被中断，立即执行中断服务程序（见图 4-10）。切换回全局寄存器组之后，完成验证过程：
 - 如果中断发生在**保存阶段**，整个验证过程从头开始执行。
 - 如果中断发生在**加载阶段**，只重复加载阶段的操作。

如果局部寄存器组的中断服务程序（图 4-11 中的任务 B）再次被一个使用全局寄存器组的中断（任务 C）打断，在执行任务 C 的代码之前，必须先完成已被挂起的验证过程。这意味着任务 A 的验证过程不会影响任务 B 的中断延迟时间，而会影响任务 C 的中断延迟时间。

注：如果任务 C 立即中断任务 A，将首先完成任务 A 的寄存器组验证过程。最坏的情况，中断延迟在两种情况下相同（见图 4-9 和图 4-11）。

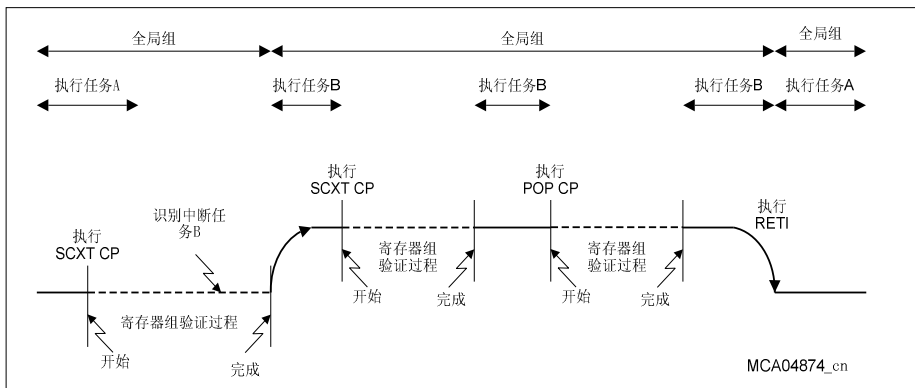


图 4-9 验证过程被使用全局寄存器组的中断程序中中断

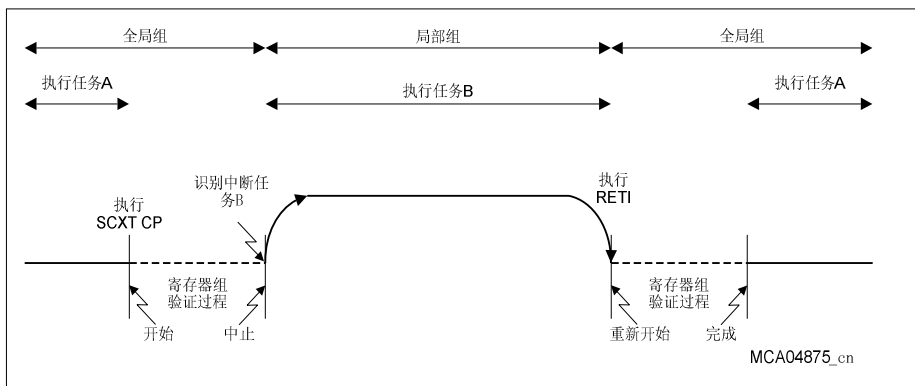


图 4-10 验证过程被使用局部寄存器组的中断程序中中断

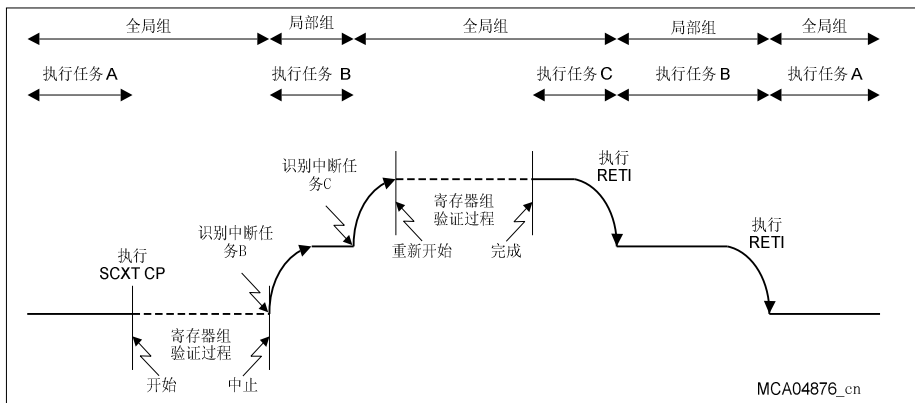


图 4-11 验证过程被使用局部寄存器组和全局寄存器组的中断程序中断

上下文指针（CP）

该寄存器不可位寻址，用来选择当前全局寄存器组上下文。任何能修改 SFR 的指令都能够更新 CP 寄存器。

CP

上下文指针

SFR（FE10_H/08_H）

复位值: FC00_H

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

符号	位序号	读写类型	功能描述
cp	[11:1]	rw	寄存器 CP 可修改部分 指定当前全局（映射到存储器）寄存器组的（字）基地址。如果对 CP[11:9]位写入 000 _B ，则 CP[11:10]由硬件设置为 11 _B 。

注：用户应该确保所设置的 CP 值和 GPR 短地址逻辑相加后的 GPR 物理地址始终在内部 DPRAM 地址范围内。如果不满足该条件，会出现不可预知的结果。设置 CP（基地址）不能小于内部 DPRAM 的起始地址。

XC164CM 可使用一条指令切换存储器映射 GPR 组。切换之后，服务程序在自己独立的上下文内执行。

指令“SCXT CP, #New_Bank”将当前的上下文指针（CP）的值压入系统堆栈，并用“New_Bank”加载 CP，该立即数选定一个新寄存器组。此时服务程序可以使用它“自己的寄存器”。当服务程序终止时，该存储器寄存器组中的内容被保留，即，其内容在下次调用服务程序时依然可用。

在从中断服务程序返回之前（RETI），先前的 CP 从系统堆栈中弹出，使用先前的寄存器组。

注：由于内部指令流水执行，因此，对 CP 寄存器的写操作会停滞指令流，直到真正执行完寄存器文件的上下文切换为止。紧随 CP 寄存器更新指令之后的指令可以使用改变后的 CP 新值。

4.6 代码寻址

XC164CM 提供了 16MB 的可寻址存储空间，该地址空间分为 256 段、每段 64KB。通过一个专用的 24 位代码地址指针从存储器中读取指令。该指针分为两部分：一个 8 位代码段指针 CSP 和一个被称作指令指针（IP）的 16 位偏移指针。CSP 和 IP 直接级联构成 24 位物理存储器地址。

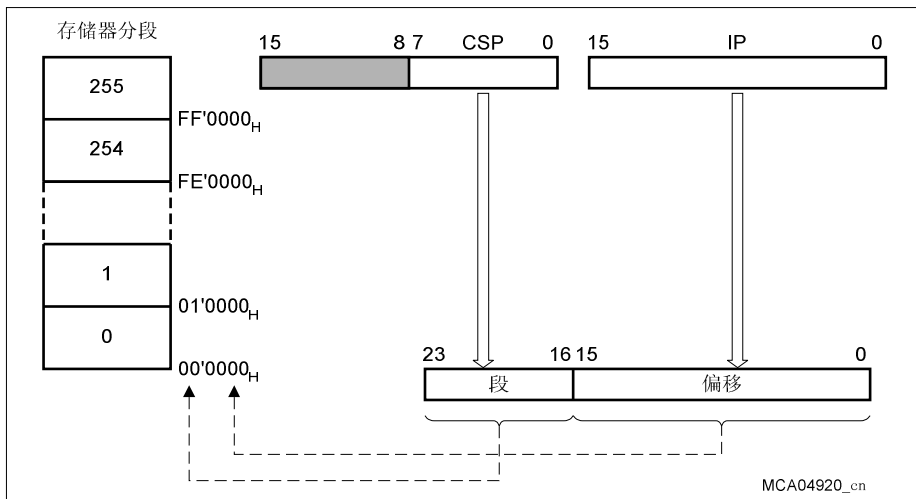


图 4-12 通过代码段指针和指令指针寻址

代码段指针 CSP 选择当前运行指令所在的代码段。寄存器 CSP 的低 8 位用来从 256 个 64KB 大小的代码段中选择一个代码段，高 8 位保留待用。CSP 的复位值由 VECSEG 寄存器指定（见[章节 5.3](#)）。

注：寄存器 CSP 为只读存储器，不能对其进行写访问。

存储器分段模式下（复位后的缺省状态），寄存器 CSP 可通过 J MPS 和 C ALLS 指令直接修改，也可通过 R ET S 和 R ET I 指令由堆栈间接修改。

存储器不分段模式下（置位寄存器 CPUCON1 中的位 S G T D I S），CSP 固定指向用来禁止分段的指令所在的代码段。使用段间调用（C ALLS）或返回（R ET urns）指令不可能修改寄存器 CSP。

在响应中断或强制中断时，寄存器 CSP 会自动载入中断向量表的段地址（由寄存器 VECSEG 指定）。

注：在不分段存储器模式下为了正确执行中断任务，VECSEG 必须选择和 CSP 当前值相同的段，也就是说，向量表必须位于 CSP 指向的段。

CSP

代码段指针

SFR (FE08_H/04_H)

复位值: xxxx_H

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

符号	位序号	读写类型	功能描述
SEGNR	[7:0]	rh	指定当前指令所在的段地址

注：复位之后，将寄存器 **VECSEG** 的值自动载入寄存器 **CSP**。

指令指针IP决定当前读取指令的16位段内地址，代码段由**CSP**寄存器选择。寄存器**IP**未被映射到**XC164CM**的地址空间，因此不能由程序员直接访问。不过，它可通过返回指令由堆栈间接修改。执行分支指令和读取指令之后，**IP**被**CPU**更新。

IP

指令指针

--- (---/---)

复位值: 0000_H

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

符号	位序号	读写类型	功能描述
IP	[15:1]	h	指定当前指令的段内偏移量。 IP 与当前段 < SEGNR >有关。
0	0	-	IP 始终按字操作。

4.7 数据寻址

地址数据单元（ADU）包含两个独立的算术单元：标准地址产生单元（SAGU）和 DSP 地址产生单元（DAGU），用来产生、计算和更新数据访问地址。ADU 主要执行以下任务：

- 标准地址产生（SAGU）
- DSP 地址产生（DAGU）
- 数据分页（SAGU）
- 堆栈处理（SAGU）

SAGU 支持间接寻址模式的线性算术地址运算，还可产生所有其它短寻址和长寻址模式的地址。

DAGU 包括一套附加的地址指针和偏移寄存器，仅和 CoXXX 指令一起使用。

CPU 提供了多种功能强大的用于字、字节和位数据访问的寻址模式（短、长、间接寻址）。不同的寻址模式使用不同的格式，具有不同的寻址范围。

4.7.1 短寻址模式

短寻址模式可访问 GPR、SFR 或可位寻址的存储器空间。这些寻址模式通过一个偏移地址（8/4/2 位）和一个隐含的基地址共同确定 24 位物理地址：

表 4-17 短寻址模式

助记符	基地址 ¹⁾	偏移地址	短地址区域	访问范围
Rw	(CP)	2 × Rw	0 ... 15	GPR（字）
Rb	(CP)	1 × Rb	0 ... 15	GPR（字节）
reg	00'FE00 _H	2 × reg	00 _H ... EF _H	SFR（字，低字节）
	00'F000 _H	2 × reg	00 _H ... EF _H	ESFR（字，低字节）
	(CP)	1× (reg ^ 0F _H)	F0 _H ... FF _H	GPR（字）
	(CP)	1× (reg ^ 0F _H)	F0 _H ... FF _H	GPR（字节）
bitoff	00'FD00 _H	2 × bitoff	00 _H ... 7F _H	RAM 中可位寻址字
	00'FF00 _H	2× (bitoff ^ 7F _H)	80 _H ... EF _H	SFR 中可位寻址字
	00'F100 _H	2× (bitoff ^ 7F _H)	80 _H ... EF _H	ESFR 中可位寻址字
	(CP)	2× (bitoff ^ 0F _H)	F0 _H ... FF _H	GPR 中可位寻址字
bitaddr	可位寻址字 见bitoff	位元位置	0 ... 15	任意位

1) 对通用寄存器（GPR）的访问也可能访问局部寄存器组，此时不使用 CP。

物理地址 = 基地址 + Δ × 短地址

注：访问字节 GPR，Δ 等于 1；访问字 GPR，Δ 等于 2。

Rw, Rb: 用于直接访问当前工作上下文（全局寄存器组或局部寄存器组）中的任意 GPR。指令格式中'Rw'和'Rb'需要 4 位。全局寄存器组的基地址由寄存器 CP 决定。'Rw'和'Rb'分别给出局部寄存器组内或全局寄存器组内（以 CP 为基地址）的 4 位 GPR 字地址和 4 位 GPR 字节地址。

reg: 用于直接访问任意(E)SFR 或当前工作上下文（全局寄存器组或局部寄存器组）中的任意 GPR。指令格式中'reg'值需要 8 位。短地址范围在 00_H 和 EF_H 之间的'reg'始终用来指定(E)SFR 地址。此时，因子'Δ'等于 2，访问标准 SFR 区时，基地址为 00' FE00_H；访问扩展 ESFR 区时，基地址为 00'F000_H。通过'reg'访问 ESFR 区之前，需要用 EXT*R 指令切换基地址。根据操作码不同，通过'reg'可分别对 SFR 的整个字（字操作）或低字节（字节操作）进行寻址。请注意：不能通过'reg'寻址模式访问

SFR 的高字节。短地址范围在 $F0_H$ 和 FF_H 之间的 'reg' 始终用来指定 GPR 地址。此时，只使用 'reg' 的低四位来确定 GPR 的物理地址，因此，它与 'Rw' 和 'Rb' 寻址模式相同。

bitoff: 用于直接访问可位寻址存储空间中的任意字。指令格式中 'bitoff' 需要 8 位。对应不同地址范围的 'bitoff' 选择不同的基地址，以产生所需物理地址（见表 4-17）。通过 'bitoff' 访问 ESFR 区之前，需要用 EXT*R 指令切换基地址。

bitaddr: 可位寻址存储空间中任意位元的地址均由字地址（见 'bitoff'）和该字中的位元位置（'bitpos'）共同决定。因此，在指令格式中 'bitaddr' 需要 12 位。

4.7.2 长寻址模式

长寻址模式指定 24 位地址，因此可访问整个地址空间中的任意字或字节数据。可以采用不同的方式来指定长地址，以产生完整的 24 位地址：

- **使用四个数据页指针之一（DPP 寄存器）：**利用 16 位指针中的位 15...14 来选择一个 DPP，利用位 13...0 指定 14 位数据页偏移地址（见图 4-13）。
- **直接选择使用的数据页：**通过前面的 EXTP（R）指令选择数据页，16 位指针中的位 13...0 指定 14 位数据页偏移地址。

直接选择使用的段：通过前面的 EXTS（R）指令选择段地址，16 位指针指定 16 位段偏移地址。

注：不能执行对奇地址字节的字访问，该操作会引发一个硬件强制中断。

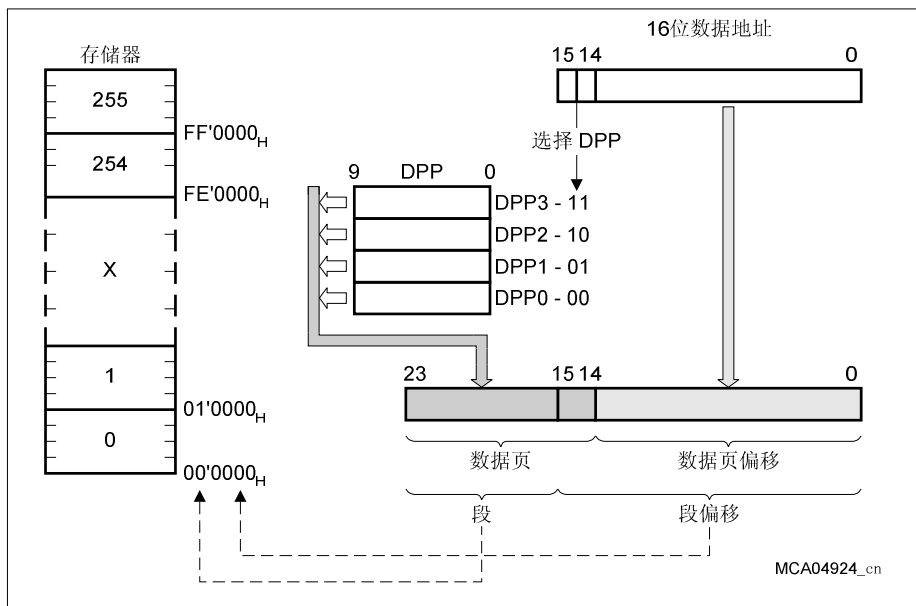


图 4-13 数据页指针寻址

数据页指针 DPP0, DPP1, DPP2, DPP3

这四个不可位寻址的寄存器可以使多达四个不同的数据页同时有效。利用每个DPP寄存器的低10位，从1024个可能的数据页（每个数据页16KB）中选择一个数据页；高6位保留待用。

DPP0

数据页指针0						SFR（FE00 _H /00 _H ）						复位值: 0000 _H				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
-	-	-	-	-	-	DPP0PN										
-	-	-	-	-	-	rw										

DPP1

数据页指针1						SFR（FE02 _H /01 _H ）						复位值: 0001 _H			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	DPP1PN									
-	-	-	-	-	-	rw									

DPP2

数据页指针2						SFR（FE04 _H /02 _H ）						复位值: 0002 _H					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
-	-	-	-	-	-	DPP2PN											
-	-	-	-	-	-	rw											

DPP3

数据页指针3						SFR（FE06 _H /03 _H ）						复位值: 0003 _H			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	DPP3PN									
-	-	-	-	-	-	rw									

符号	位序号	读写类型	功能描述
DPPxPN	[9:0]	rw	DPPx 的数据页编码 通过 DPPx 指定所选用的数据页

DPP 寄存器以 16KB 数据页为单位，能够访问整个存储器空间。只要通过间接或直接 16 位长寻址模式来访问任意存储器地址（通过 EXT 扩展指令的替代访问和 PEC 数据传送除外），都会隐含地使用 DPP 寄存器。复位之后，数据页指针以下述方式初始化：所有间接或直接 16 位长地址都生成同样的 18 位地址，从而可以访问第 0 段内的数据页 3...0，如图 4-13 所示。用户如果不想使用数据分页，则无需其它操作。

将间接或直接 16 位长地址的低 14 位和由其高 2 位所选择的 DPP 寄存器的内容串联起来，即实现了数据分页。所选择的 DPP 寄存器指定 1024 个数据页中的一个。该数据页基地址和 14 位页偏移地址共同组成 24 位的物理地址（即使分段被禁止）。

访问外部数据时，位域 SALSEL 选中的 DPP 寄存器段地址被输出到相应的段地址引脚上。

可通过任何能够修改 SFR 的指令更新 DPP 寄存器。

注：由于内部指令流水执行，因此，对 DPPx 寄存器的写操作会停滞指令流，直到 DPP 被更新为止。紧随 DPP 寄存器更新指令之后的指令可以使用改变后的 DPPx 新值。

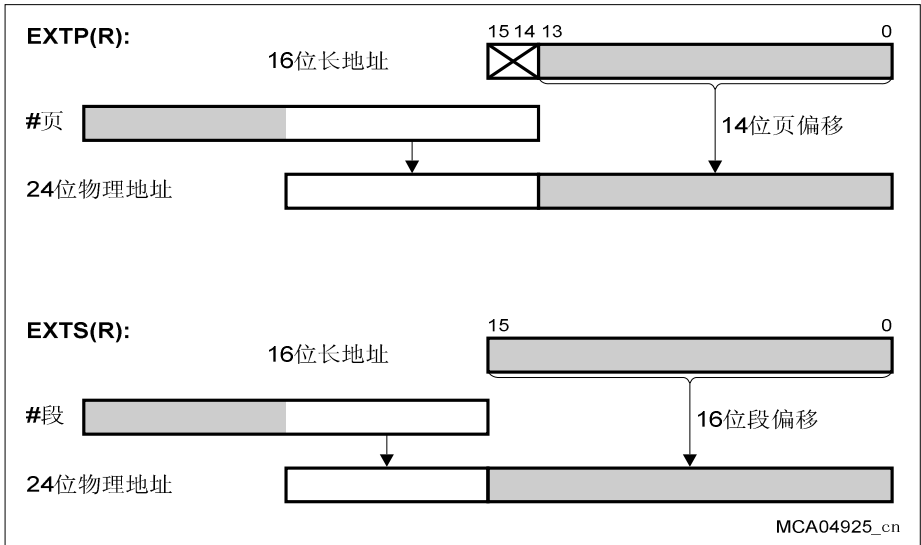


图 4-14 DPP 替代机制（直接选择数据页或段地址）

注：替代页或替代段可由一个常数（#pag, #seg）或通过 GPR 字（Rw）指定。

表 4-18 长寻址模式

助记符	基地址 ¹⁾	偏移量	访问范围
mem	(DPPx)	mem ^ 3FFF _H	任意字或字节
mem	pag	mem ^ 3FFF _H	任意字或字节
mem	seg	mem	任意字或字节

1) 代表与14位偏移地址串联的10位数据页编号，或者与16位偏移地址串联的8位段编号。

4.7.3 间接寻址模式

可以将间接寻址模式看成是短寻址和长寻址模式的组合。也就是说，16 位“长”指针通过 GPR 字的内容间接提供，GPR 由 4 位短地址直接指定（‘Rw’ = 0 ... 15）。

有些间接寻址模式，在计算 16 位长地址之前，给 GPR 的内容加上一个常量；有些间接寻址模式，能以 2 或 1（对应字或字节）递减或递增间接地址指针（GPR 的内容），或者以偏移地址寄存器 QR0 或 QR1 的值递减或递增间接地址指针。

表 4-19 从间接指针产生物理地址

步骤	执行的操作	计算	备注
1	从短地址计算间接指针（GPR 字）的地址	GPR 地址 = 2 × 短地址 [+ (CP)]	见表 4-17
2	根据数据类型（字节操作 Δ=1；字操作 Δ=2），前减间接指针（‘-Rw’）	(GPR 地址) = (GPR 地址) - Δ	可选步骤，仅在寻址模式需要时执行
3	用一个常数调整指针（‘Rw+const16’）	指针 = (GPR 地址) + 常数	可选步骤，仅在寻址模式需要时执行
4	利用计算结果指针计算 24 位物理地址	物理地址 = 页/段 + 指针偏移量	使用 DPP 或页/段替代化机制，见表 4-18
5	根据数据类型（字节操作 Δ=1；字操作 Δ=2），或根据偏移寄存器的值（Δ=QRx） ¹⁾ ，后增/减间接指针（‘Rw ±’）	(GPR 地址) = (GPR 地址) ± Δ	可选步骤，仅在寻址模式需要时执行

1) 后减以及根据 QRx 修改间接指针仅适用于 CoXXX 指令。

注：有些指令只使用 GPR 的最低四个字（R3...R0）作为间接地址指针，这种情况下，GPR 由 2 位短地址指定。

XC164CM 提供了以下几种间接寻址模式：

表 4-20 间接寻址模式

助记符	特性
[Rw]	大多数指令使用任何一个 GPR (R15...R0) 作为间接地址指针；有些指令只使用最低 4 个 GPR (R3...R0) 作为间接地址指针。
[Rw+]	访问之后，间接地址指针自动加 2 或 1（对应字或字节操作）
[-Rw]	访问之前，间接地址指针自动减 2 或 1（对应字或字节操作）
[Rw + #data16]	计算长地址之前，间接地址指针加上指定的 16 位常数
[RW-]	访问之后，间接地址指针自动减 2（对应字操作）
[Rw + QRx]	访问之后，间接地址指针自动加 QRx（对应字操作）
[Rw - QRx]	访问之后，间接地址指针自动减 QRx（对应字操作）

不可位寻址的偏移寄存器 QR0 和 QR1 与 CoXXX 指令一起使用。可能引起的指令流停滞参见[章节 4.3.4](#)。

QR0

偏移寄存器

ESFR (F004_H/02_H)

复位值: 0000_H

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

QR1

偏移寄存器

ESFR (F006_H/03_H)

复位值: 0000_H

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

符号	位序号	读写类型	功能描述
QR	[15:1]	rw	寄存器 QRx 可修改部分 为间接寻址模式指定 16 位字偏移地址（最低位始终为 0）

4.7.4 DSP 寻址模式

除了标准地址产生单元（SAGU），DSP 地址产生单元（DAGU）还提供了一组附加的指针寄存器（IDX0、IDX1）和偏移寄存器（QX0、QX1）。使用指针寄存器 IDX0 和 IDX1，DSP 专用 CoXXX 指令可在一个 CPU 周期内完成。XC164CM 提供了一个独立的算术单元，可并行完成这些专用指针寄存器的更新和 SAGU 中 GPR 指针的修改。DAGU 只支持使用特殊指针寄存器 IDX0 和 IDX1 的间接寻址模式。

这两个地址指针可用于算术操作，也可用于特殊的 CoMOV 指令。产生 24 位物理地址的方式有所不同：

- 对于 **CoMOV** 指令，IDX 指针与 DPP 或选择的数据页/段地址级联，这和长寻址模式相同（总结见图 4-13）。
- 对于算术 **CoXXX** 指令，IDX 指针自动扩展成 24 位存储器地址，指向内部 DPRAM 区，如图 4-15 所示。

IDX0

地址指针																SFR（FF08 _H /84 _H ）																复位值: 0000 _H															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																
																idx																0															
																rw																r															

IDX1

地址指针																SFR（FF0A _H /85 _H ）																复位值: 0000 _H															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																
															idx															0																	
																rw																r															

符号	位序号	读写类型	功能描述
idx	[15:1]	rw	寄存器 IDXx 可修改部分 指定 16 位字地址指针。

注：在初始化IDX寄存器时，可能停滞指令流。正确的操作请参见[章节4.3.4](#)。

有些间接寻址模式，在计算 16 位长地址之前，可进行并行的数据转移操作（示例见图 4-16）；有些间接寻址模式，允许 2 递减或 2 递增间接地址指针（IDXx 内容），或者以偏移地址寄存器 QX0 或 QX1 的值递增或递减间接地址指针（与 IDX 指针联合使用）。

QX0

偏移寄存器																ESFR (F000 _H /00 _H)				复位值: 0000 _H			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
qx																				0			
																rw				r			

QX1

偏移寄存器																ESFR (F002 _H /01 _H)				复位值: 0000 _H			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
qx																				0			
																rw				r			

符号	位序号	读写类型	功能描述
qx	[15:1]	rw	寄存器 QXx 可修改部分 为间接寻址模式指定 16 位字偏移地址。

注：在初始化QX寄存器时，可能停滞指令流。正确的操作参见[章节4.3.4](#)。

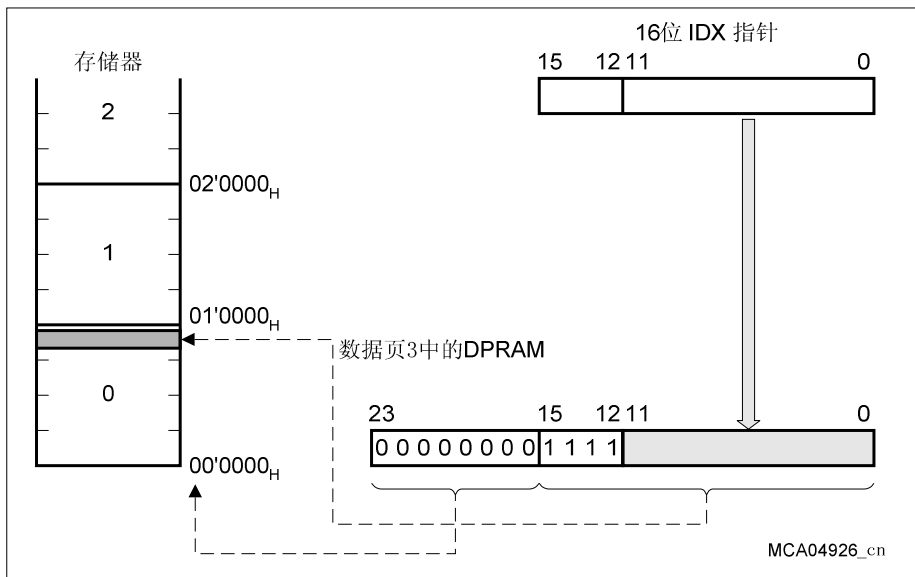


图 4-15 算术 MAC 操作和通过 IDX 指针寻址

表 4-21 从间接指针（IDXx）产生物理地址

步骤	执行的操作	计算	备注
1	确定所使用的 IDXx 指针	---	-
2	为并行数据转移操作计算中间长地址，并以 $\Delta=2$ 或偏移寄存器的值 ($\Delta=QXx$) 递增或递减间接指针 ('IDXx \pm ')	中间地址 = (IDXx 指针) $\pm \Delta$	可选步骤，仅在指令 CoXXXXM 和寻址模式需要时执行
3	计算 16 位长地址	长地址 = (IDXx 指针)	-
4	利用计算结果指针计算 24 位物理地址	物理地址 = 页/段 + 指针偏移量	使用 DPP 或页/段替代机制，见表 4-18 和图 4-15

步骤	执行的操作	计算	备注
5	以 $\Delta=2$ 或偏移量寄存器的值($\Delta=QXx$), 后递增或递减间接指针 (' $IDXx \pm '$)	($IDXx$ 指针) = ($IDXx$ 指针) $\pm \Delta$	可选步骤, 仅在寻址模式需要时执行

XC164CM提供了以下几种间接寻址模式:

表 4-22 DSP 寻址模式

助记符	特殊性
[$IDXx$]	大多数 CoXXX 指令使用 $IDXx$ ($IDX0,IDX1$) 作为间接地址指针
[$IDXx+$]	访问之后, 间接地址指针自动加 2
有并行数据转移	对于 CoXXXM 指令, 保存在间接地址指针中的地址自动前减 2 用于并行转移操作, 指针自身不前减。访问之后, 间接地址指针自动加 2。
[$IDXx-$]	访问之后, 间接地址指针自动减 2
有并行数据转移	对于 CoXXXM 指令, 保存在间接地址指针中的地址自动前加 2 用于并行转移操作, 指针自身不前加。访问之后, 间接地址指针自动减 2。
[$IDXx+QXx$]	访问之后, 间接地址指针自动递增 QXx
有并行数据转移	对于 CoXXXM 指令, 保存在间接地址指针中的地址自动前减 QXx 用于并行转移操作。指针自身不前减。访问之后, 间接地址指针自动递增 QXx 。
[$IDX-QXx$]	访问之后, 间接地址指针自动递减 QXx
有并行数据转移	对于 CoXXXM 指令, 保存在间接地址指针内的地址自动前加 QXx 用于并行转移操作, 指针自身不前加。访问之后, 间接地址指针自动递减 QXx 。

注: 并行数据转移的例子可参见[图 4-16](#)。

CoREG 寻址模式

CoSTORE指令使用特殊的CoREG寻址模式，在一次MAC操作之后立即保存MAC单元寄存器。MAC单元寄存器的地址编码在CoSTORE指令格式中，如表4-23所示。

表 4-23 CoREG 寻址模式编码

助记符	寄存器	www:w 位[31:27]编码
MSW	MAC 单元状态字	00000
MAH	MAC 单元累加器的高位字	00001
MAS	受限的 MAC 单元累加器高位字	00010
MAL	MAC 单元累加器的低位字	00100
MCW	MAC 单元控制字	00101
MRW	MAC 单元重复字	00110

图4-16的例子给出带有并行数据转移操作的CoXXXM指令的复杂操作过程，该操作基于[章节4.7.3（间接寻址模式）](#)和[章节4.7.4（DSP寻址模式）](#)所描述的寻址模式。

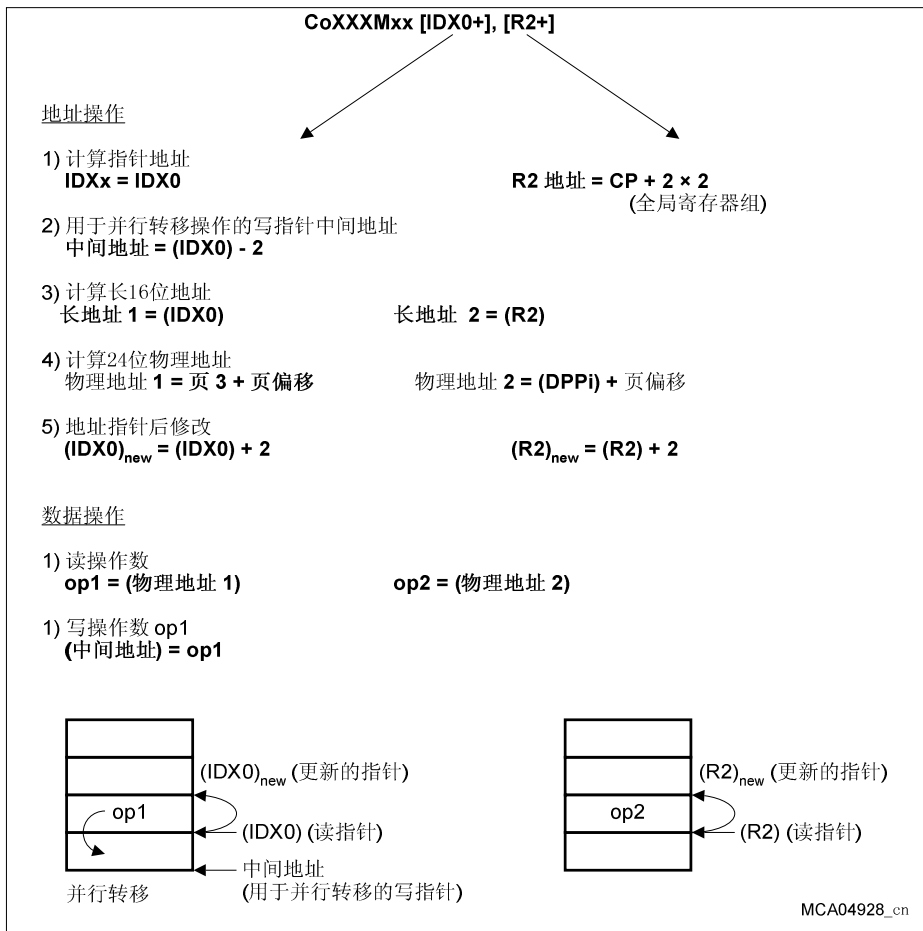


图 4-16 算术 MAC 操作和并行转移

4.7.5 系统堆栈

XC164CM 支持高达 64KB 的系统堆栈。堆栈可以位于片上或者片外存储器。使用 16 位堆栈指针寄存器（SP）对段内的 64KB 进行寻址，堆栈段由堆栈指针段寄存器（SPSEG）选定。还可以用软件实现虚拟堆栈（通常大于 64KB）。堆栈上溢寄存器 STKOV 和堆栈下溢寄存器 STKUN 支持该机制（说明见下文）。

堆栈指针寄存器 SP 和 SPSEG

寄存器 SPSEG（不可位寻址）选择当前堆栈所在的段。寄存器 SPSEG 的低 8 位用来从 256 个 64KB 大小的段中选择一个作为堆栈段，高 8 位保留待用。

堆栈指针 SP（不可位寻址）指向系统堆栈的栈顶（TOS）。数据压栈之前 SP 自动递减；数据出栈后 SP 自动递增。因此，按照从高地址到向低地址的方向生成系统堆栈。

系统堆栈地址由寄存器 SPSEG 的内容和寄存器 SP 的 16 位值共同构成，如图 4-17 所示。

系统堆栈不能跨越 64KB 的段边界。

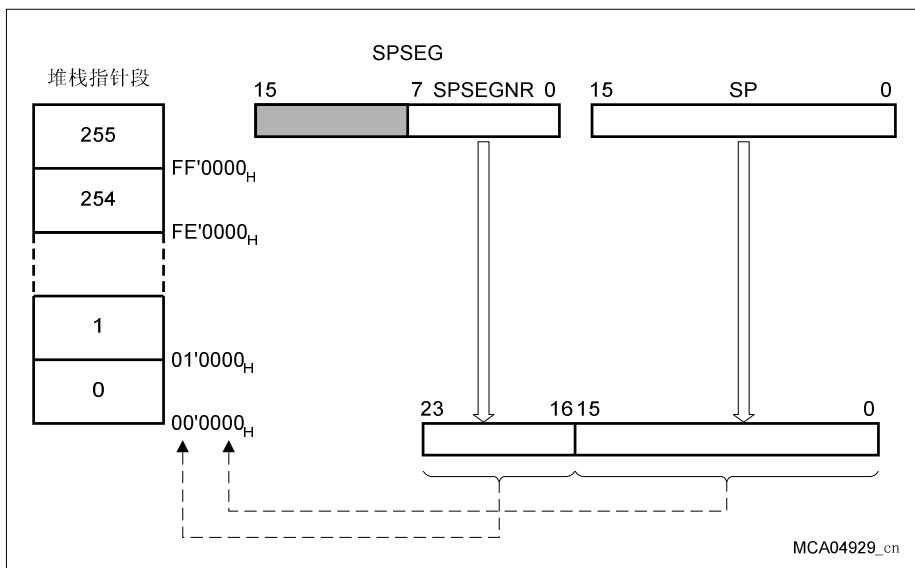


图 4-17 通过堆栈指针寻址

SP

堆栈指针寄存器

SFR (FE12_H/09_H)

复位值: FC00_H

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

符号	位序号	读写类型	功能描述
sp	[15:1]	rwh	寄存器 SP 可修改部分 指定系统堆栈的栈顶。

SPSEG

堆栈指针段

SFR (FF0C_H/86_H)

复位值: 0000_H

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

符号	位序号	读写类型	功能描述
SPSEGNR	[7:0]	rw	堆栈指针段编号 指定堆栈所在的段。

*注：可以由任何能够修改 16 位 SFR 的指令更新寄存器 SPSEG 和 SP。由于内部指令流水执行，对 SPSEG 或 SP 的写操作会使指令流停滞，直到寄存器更新完毕。紧随 SPSEG 或 SP 更新指令之后的指令可以使用改变后的新值。
在改变堆栈指针寄存器的内容时，需要特别的小心。不恰当的修改可能导致错误的系统行为。*

堆栈上溢/下溢指针 STKOV/STKUN

这两个边界寄存器（不可位寻址）用于监视堆栈指针。堆栈指针达到上限或下限时会产生一个强制中断。进行堆栈指针比较时不需要考虑堆栈指针段寄存器SPSEG。系统堆栈不能跨越64KB段边界。

每次对SP隐含写操作（该操作会使SP自动递减）之前（指令CALLA、CALLI、CALLR、CALLS、PCALL、TRAP、SCXT或PUSH），STKOV与SP进行比较。如果SP和STKOV相等，则触发堆栈上溢强制中断。

每次对SP隐含读操作（该操作会使SP自动递增）之前（指令RET、RETS、RETP、RETI或POP），STKUN与SP进行比较，如果SP和STKUN相等，则触发堆栈下溢强制中断。

有两种不同的方法来处理堆栈上溢/下溢强制中断：

- **致命错误指示**将堆栈溢出看成系统错误，执行相关的强制中断服务程序。
在堆栈溢出强制中断情况下，栈底的数据可能已经被执行强制中断服务程序时压栈的状态信息所覆盖。
- **虚拟堆栈控制**可将系统堆栈用作“堆栈缓存”，以支持更大的外部用户堆栈：上溢时清空缓存；下溢时重新填充缓存。

堆栈界限控制范围

通过寄存器 STKOV 和 STKUN 可以检测出堆栈指针（SP）被隐含修改后是否超过了规定的堆栈区，从而实现了堆栈界限控制。

如果通过转移指令或算术指令显性的修改堆栈指针，则 SP 不与 STKOV 和 STKUN 进行比较。这种情况下，若修改后的堆栈指针超出规定的堆栈区，即低于 STKOV 或高于 STKUN，将无法检测到堆栈错误。只有当 SP 被隐含修改时，才进行堆栈上溢/下溢检测。

SP 值可能超出允许的 SP 范围，而不触发强制中断。但是，如果由于隐含修改（例如 PUSH 或 POP）SP 值而导致超出允许的界限，则会触发相应的强制中断。

注：STKOV 和 STKUN 可以由任何能够修改 SFR 的指令进行更新。如果在 ATOMIC/EXT 序列中发生堆栈上溢或下溢，则完成作为序列指令一部分的堆栈操作。在完成所有的 ATOMIC/EXT 序列之后，才发出强制中断。

STKOV

堆栈上溢寄存器

SFR (FE14_H/0A_H)

复位值: FA00_H

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

符号	位序号	读写类型	功能描述
stkov	[15:1]	rw	寄存器 STKOV 可修改部分 指定系统堆栈下限的段内偏移地址。

STKUN

堆栈下溢寄存器

SFR (FE16_H/0B_H)

复位值: FC00_H

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

符号	位序号	读写类型	功能描述
stkun	[15:1]	rw	寄存器 STKUN 可修改部分 指定系统堆栈上限的段内偏移地址。

4.8 标准数据处理

通过16位ALU执行所有的标准算术、移位和逻辑操作。除实现标准功能以外，XC164CM的ALU还包括一个位处理单元和一个乘除单元。大多数内部执行模块经过优化设计，用于执行8位或16位运算。流水线被填充之后，大多数指令在一个CPU周期内完成。每次ALU操作之后，PSW寄存器的状态标志自动更新，指示微控制器的当前状态。这些标志可以用作分支转移的特定条件。XC164CM支持有符号运算和无符号运算，这通过用户可选的分支测试来完成。进入中断或强制中断服务程序时，由CPU自动保存状态标志。另一组标志位表示当前CPU的中断状态。两个独立位（USR0和USR1）用作通用标志。

PSW

处理器状态字

SFR (FF10_H/88_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ILVL			IEN	-	BANK		USR 1	USR 0	MUL IP	E	Z	V	C	N	
rwh			rw	-	rwh		rwh	rwh	r	rwh	rwh	rwh	rwh	rwh	

符号	位序号	读写类型	功能描述
ILVL	[15:12]	rwh	CPU 优先级 0 _H 优先级最低 F _H 优先级最高
IEN	11	rw	全局中断/PEC 使能位 0 禁止中断/PEC 请求 1 使能中断/PEC 请求
BANK	[9:8]	rwh	用于寄存器文件组选择 00 全局寄存器组 01 保留 10 局部寄存器组 1 11 局部寄存器组 2
USR1	7	rwh	通用标志 可被应用程序使用
USR0	6	rwh	通用标志

符号	位序号	读写类型	功能描述
			可被应用程序使用
MULIP	5	r	正在进行乘/除运算 <i>注：始终置 0（MUL/DIV 不可中断），用于和现有的软件兼容</i>
E	4	rwh	表格结束标志 0 源操作数既不是 8000 _H 也不是 80 _H 1 源操作数是 8000 _H 或 80 _H
Z	3	rwh	零标志 0 ALU 结果非零 1 ALU 结果为零
V	2	rwh	溢出标志 0 没有产生溢出 1 产生溢出
C	1	rwh	进位标志 0 没有产生进位/借位 1 产生进位/借位
N	0	rwh	负值结果 0 ALU 结果非负 1 ALU 结果为负

ALU/MAC 状态（N、C、V、Z、E、USR0、USR1）

PSW 中的状态标志（N、C、V、Z、E）指示最近一次执行 ALU 操作之后，ALU 的状态。大多数指令会根据特定规则（这取决于执行 ALU 操作或是数据转移操作）设置这些标志位。

若通过指令显性更新 PSW 寄存器，则不能按照以下的描述来解读状态标志，因为对 PSW 的任何显性设置将取代由 CPU 隐含产生的状态标志。用户读取 PSW 寄存器，将读出前一条指令执行之后的 PSW 状态。

注：复位后 ALU 的所有状态位被清零。

N 标志：对于大多数 ALU 操作，如果结果的最高位为 1，则 N 标志置 1；反之清零。对于整数操作，N 标志可以看成结果的符号位（负数：N=1，正数：N=0）。负数始终用相应正数的 2 补码表示。对于字数据类型，有符号数的范围从-8000_H至

+7FFF_H，对字节数据类型，有符号数的范围从-80_H至+7F_H。对只有一个操作数的布尔位操作，N标志表示指定定位的前一个状态；对有两个操作数的布尔位操作，N标志表示两个指定定位的逻辑异或（XOR）。

C标志：一次加法运算之后，C标志表示指定字或字节数据的最高位是否产生进位。减法或比较操作后，C标志指示借位，即加法进位标志的逻辑“非”。

这意味着，在减法运算中，若指定字或字节数据的最高位**没有**产生进位，则C标志置1（在ALU内部，通过2补码加法来实现减法运算）；当补码加法引起进位时，C标志清零。

对于逻辑操作、乘除ALU操作，C标志始终清零，因为这些操作不能产生进位。

对于移位和循环移位操作，C标志表示最后移出位的值。如果移位计数到0，C标志清零。对优化ALU操作来说，C标志也会被清零，这是因为在对一个操作数归一化的过程中不可能从最高位移出1。

对只有一个操作数的布尔位操作，C标志始终清零。对有两个操作数的布尔位操作，C标志表示两个指定定位的逻辑与（AND）。

V标志：进行加法、减法和2补码运算时，如果字运算结果超过16位有符号数取值范围（-8000_H到+7FFF_H），或者字节运算结果超过8位有符号数取值范围（-80_H到+7F_H），则V标志始终置1。反之V标志清零。请注意，如果V标志指示发生溢出，则整数加、整数减或2补码运算的结果无效。

进行乘除运算时，如果结果不能用一个字数据来表示，则V标志置1；反之清零。请注意，除以零的操作始终导致溢出。与除法结果不同，不论V标志是否置1，乘法结果始终有效。

由于逻辑ALU操作不可能产生一个无效结果，因此V标志始终清零。

V标志还能用作循环右移位和右移位操作的“粘着位”。仅使用C标志，可以估算的由右移位操作所导致的舍入误差最多为1/2 LSB。和V标志一起使用时，C标志可以更精确的估算舍入误差（见[表 4-24](#)）。

对只有一个操作数的布尔位操作，V标志始终被清零；对有两个操作数的布尔位操作，V标志表示两个指定定位的逻辑或（OR）。

表 4-24 右移位舍入误差的估算

C标志	V标志	舍入误差的估算
0	0	无舍入误差
0	1	0<舍入误差<1/2 LSB
1	0	舍入误差 = 1/2 LSB
1	1	舍入误差>1/2 LSB

Z 标志：如果 ALU 操作结果等于 0，通常 Z 标志置 1；反之清零。

对带进位的加法和减法运算，只有当 Z 标志已经为 1，并且当前 ALU 的操作结果也等于零时，Z 标志才被置 1。该机制用于支持多精度计算。

对只有一个操作数的布尔位操作，Z 标志表示指定位前一状态的逻辑“非”。对有两个操作数的布尔位操作，Z 标志表示两个指定位的逻辑或非（NOR）。对于优先化 ALU 操作，Z 标志指示第二个操作数是否为 0。

E 标志：表格结束标志。E 标志可以由执行 ALU 或数据转移操作的指令改变。那些不能合理用来搜索表格的指令将清零 E 标志。在所有其它情况下，E 标志值取决于源操作数的值，以指示是否达到搜索表的末端。如果指令的源操作数的值等于相应指令的数据格式所代表的最小负值（字数据类型为 8000H；字节数据类型为 80H），E 标志置 1；否则清零。

通用控制功能（USR0、USR1、BANK）

寄存器 PSW 中包含一些位专用于通用控制功能。在任务切换和中断时，它们会被自动保存和恢复。

USR0/USR1 标志：在重复执行 MAC 指令时，USR0/USR1 标志可被自动置位。它们还可用作应用程序的通用标志。

BANK：位域 BANK 选择当前工作的寄存器组（局部或全局）。在转入中断服务程序，以及执行 RETI 指令后，硬件自动更新位域 BANK。还可以使用任何能对 PSW 进行写操作的指令修改位域 BANK。

CPU 中断状态（IEN、ILVL）

IEN：中断使能位允许中断被全局使能（IEN = 1）或禁止（IEN = 0）。

ILVL：四位中断优先级位域指定当前 CPU 工作的优先级。转入中断服务程序时，CPU 中断优先级由硬件更新。不过它也可由软件修改，以防止 CPU 响应其它中断。如果分配 CPU 的中断优先级为 15，即具有最高优先级，那么，除硬件强制中断或外部非屏蔽中断之外，当前 CPU 操作不能被中断。详细信息参见第 5 章。

复位后，所有中断被全局禁止，CPU 最初操作的优先级设定为最低（ILVL = 0）。

4.8.1 16 位加法器/减法器、阵列移位器和 16 位逻辑单元

所有的标准算术和逻辑操作由 16 位 ALU 完成。进行字节操作时，ALU 结果的第 6 和第 7 位用来控制状态标志。进行多精度运算时，将运算前面计算结果的“进位”信号送给 ALU。

16 位阵列移位器支持单周期内多位移动，还支持循环和算术移位。

4.8.2 位操作单元

XC164CM提供了许多用于位处理的指令。这些指令可以控制内部RAM中的软件标志位；或通过SFR中的控制位来控制相应的片上外设，或通过端口引脚控制IO功能。

和其它微控制器不同，XC164CM中的指令支持直接访问可位寻址空间中的两个操作数，无需先将它们移入临时单元。多位移位指令可以避免单位移位操作造成的长指令流。这些指令都为单周期指令。

指令BSET、BCLR、BAND、BOR、BXOR、BMOV、BMOVN显性的置位或清零特定位。位域指令BFLDL和BFLDH允许同时处理特定字节中的最多8位。在执行跳转指令时，指令JBC和JNBS隐含的清零或置位特定位。指令JB和JNB（以及和标志位有关的条件跳转指令）根据特定位的状态来决定是否进行跳转。

注：读取未定义的位始终返回'0'，对其进行写操作无效。

处理单个位或一组位的所有指令在内部均采用读-修改-写的过程（访问包含指定位的整个字）。

该方法产生下列结果：

- 读-修改-写方法对受硬件影响的位可能很关键。在这些情况下，硬件可能改变正在进行读-修改-写操作的特定位；此时，回写值会覆盖由硬件产生的新值。可以通过硬件保护或通过特殊编程（见[章节4.3](#)）来解决这个问题。
- 只有位于内部地址区（内部RAM和SFR）的位才能被修改。位操作指令不能用于外部地址区。

SFR区、ESFR区和内部DPRAM的高256字节可以位寻址，位于这些存储区的位元可以直接由位指令进行操作。其它SFR必须按字节/字访问。

注：所有GPR均可位寻址，与通过上下文指针（CP）指定的寄存器组的位置无关。即使GPR位于不可位寻址的RAM区域，它仍然可以位寻址。

保护位在执行读-修改-写序列期间（比如在读-修改-写序列的读和写操作之间，硬件设置一个中断请求标志）不被改变。硬件保护逻辑确保只有那些可由软件修改的位受回写操作的影响。XC164CM的所有保护位归纳见[章节2.7](#)。

注：如果对某一位的硬件修改和软件操作同时发生，软件访问占优、决定该位的最终值。

4.8.3 乘除单元

XC164CM 的乘除单元由两个独立的部分组成：一个是快速 16×16 位乘法器，能在一个 CPU 周期内完成乘法运算；一个是除法器，能在 18 至 21 个 CPU 周期内（取决于数据和除法类型）完成除法运算。执行除法指令需要 4 个 CPU 周期。为了提高性能，除法算法的剩余 17 个 CPU 周期在后台运行，其间可以并行执行其它指令，也可以无延迟、立即开始执行中断任务。如果在除法运算的过程中，某指令（来自于原指令流或中断任务）试图使用除法单元，则该指令被停滞，直到前面的除法运算完成为止。

为避免指令停滞，在中断任务的前 14 个 CPU 周期内不应使用乘除单元。举例来说：转入中断后，需要执行多达 14 条单周期指令以后，才能再次使用乘除指令（最坏情况）。

乘除运算隐含的使用 32 位乘/除寄存器 MD（由两个不可位寻址的数据寄存器 MDH 和 MDL 级联而成）和相应的控制寄存器 MDC。当 ALU 内部执行乘除操作时，CPU 隐含的使用 MDC（可位寻址的 16 位寄存器）。

乘法运算的 32 位结果保存在 MD 中。若进行长除运算，必须在除法运算之前将 32 位被除数保存在 MD 中。除法运算之后，寄存器 MDH 给出 16 位余数，寄存器 MDL 给出商。

MDH

乘/除高位寄存器 SFR (FE0CH/06H) 复位值: 0000H

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

符号	位序号	读写类型	功能描述
mdh	[15:0]	rwh	MD 的高位部分 32 位乘除寄存器 MD 的高 16 位。

MDL

乘/除低位寄存器

SFR (FE0E_H/07_H)

复位值: 0000_H

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

符号	位序号	读写类型	功能描述
mdl	[15:0]	rwh	MD 的低位部分 32 位乘除寄存器 MD 的低 16 位。

只要软件更新 MDH 或 MDL，乘/除控制寄存器（MDC）中的乘/除寄存器正在使用标志位（MDRIU）即置 1。只要软件读取寄存器 MDL，则 MDRIU 标志清零。

MDC

乘/除控制寄存器

SFR (FF0E_H/87_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	MDRIU	-	-	-	-
-	-	-	-	-	-	-	-	-	-	-	r(w)h	-	-	-	-

符号	位序号	读写类型	功能描述
MDRIU	4	rwh	乘/除寄存器正在使用标志位 0 软件读取 MDL 时该位清零 1 软件写入 MDL 或 MDH；或当执行乘法或除法指令时，该位置 1

注：MDRIU 标志指示寄存器 MD（MDL 和 MDH）的使用情况，因此必须新的乘法或除法操作前保存 MD。

4.9 DSP 数据处理（MAC 单元）

在 MAC 单元中执行新型 CoXXX 算术指令。MAC 单元支持单指令周期、非流水线的 32 位加、32 位减、右移左移、16×16 位乘法、累减/加的乘法。MAC 单元包括以下主要部分，如图 4-18 所示：

- 16×16 位有符号/无符号乘法器，结果带符号¹⁾
- 级联单元
- 用于小数计算的换算器（左移一位移位器）
- 40 位加法/减法器
- 40 位有符号累加器
- 数据限制器
- 累加移位器
- 重复计数器

1) ALU 中使用相同的硬件乘法器。

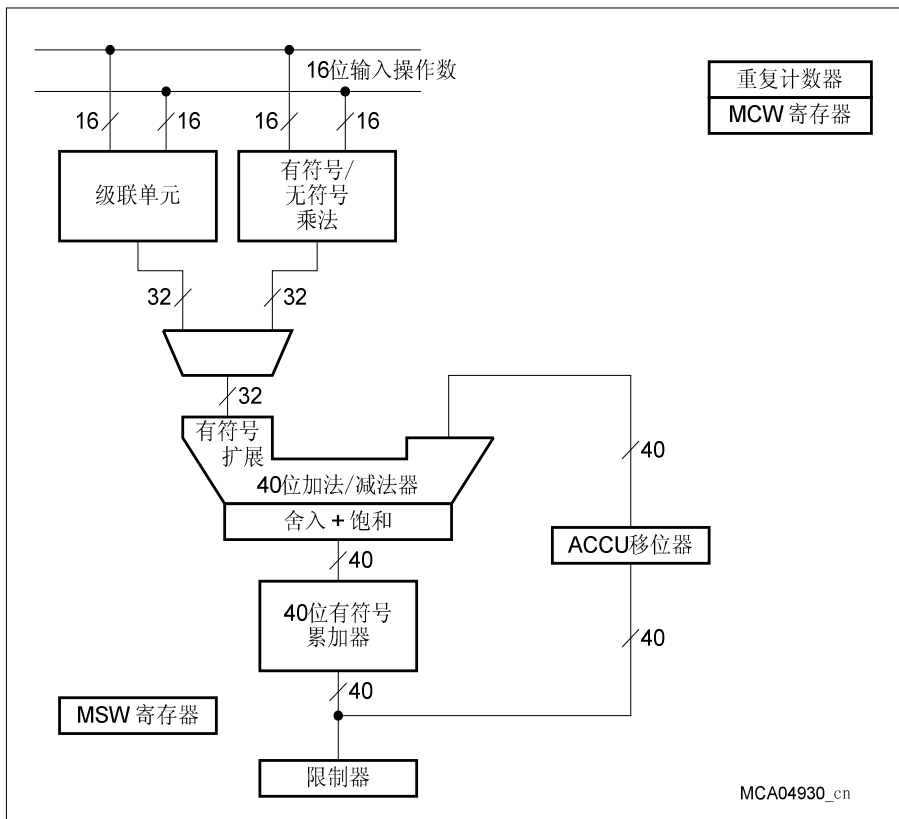


图 4-18 MAC 单元功能框图

MAC 单元的工作寄存器是一个专用 40 位累加寄存器。每次 MAC 操作之后，状态寄存器 MSW 中的一组标志位被自动更新。这些标志可以用作分支转移的特定条件。和 PSW 标志位不同，在进入中断或强制中断程序时，CPU 不会自动保留这些标志位。因此，如果不同的任务和中断共用 MAC 资源，所有 MAC 专用寄存器必须保存在堆栈中。MAC 单元的一般属性由 MAC 控制字 MCW 设定。

MCW

MAC控制字					SFR (FFDC _H /EE _H)								复位值: 0000H			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
-	-	-	-	-	MP	MS	-	-	-	-	-	-	-	-	-	
-	-	-	-	-	rw	rw	-	-	-	-	-	-	-	-	-	

符号	位序号	读写类型	功能描述
MP	10	rw	一位换算控制 0 禁止乘积移位 1 使能有符号乘法运算的乘积移位
MS	9	rw	饱和控制 0 禁止饱和和操作 1 使能饱和至 32 位值

4.9.1 数字的表示和舍入

XC164CM 支持二进制数的 2 补码表示。这种格式下，二进制字的最高位是符号位，正数设为 0，负数设为 1。只有乘法/乘累加指令支持无符号数，这些指令指定每个操作数是无符号数还是有符号数。

2 补码小数格式，用 1.[N-1]格式表示 N 位操作数（1 个符号位，N-1 个小数位）。这种格式可以表示 $-1 \sim +1 \cdot 2^{-(N-1)}$ 之间的数字。寄存器 MCW 中的 MP 置 1 时支持这种格式。

XC164CM 实现了 2 补码舍入。使用这种舍入操作，在截断操作之前（MAL 被清零），舍入点右边一位（即 MAL 的位 15）加 1。

4.9.2 16 位×16 位有符号/无符号乘法器和换算器

乘法器在一个 CPU 周期内并行完成 16×16 有符号/无符号小数和整数乘法。乘法器支持无符号和有符号操作数的乘法，乘积始终用有符号小数或整数格式表示。

乘积送给一位换算器，可用于补偿由两个 16 位 2 补码数相乘产生的附加符号位。

4.9.3 级联单元

级联单元使 MAC 能够在一个 CPU 周期内完成 32 位的算术运算。在 40 位加法/减法器中执行 32 位算术运算之前，级联单元将两个 16 位操作数级联为一个 32 位操作数。40 位加法/减法器的第二个操作数始终是累加器的当前值。级联单元还用来给累加器预载一个 32 位的数值。

4.9.4 一位换算器

一位换算器可以将级联单元的结果或乘法器的输出左移一位。换算器由级联指令或寄存器 MCW 中的位 MP 控制。

如果 MP 位置位，则乘积左移一位，用于补偿由两个 16 位 2 补码数相乘产生的附加符号位。只有当两个输入操作数都为有符号数时，才执行自动移位。

4.9.5 40 位加法/减法器

在进行一系列乘/累加操作期间，40 位加法/减法器允许中间溢出。加法/减法器有两个输入端口（一个 40 位输入口和一个 32 位输入口）。通过 ACCU 移位器将累加器的输出值反馈到加法/减法器的 40 位输入端口上。加法/减法器 32 位输入口的操作数来自一位换算器。32 位操作数是有符号数，在执行加/减法算术前被扩展成 40 位。

加法/减法器的输出送入累加器。每次累加过后，也可以自动对结果舍入和饱和至 32 位值。通过给结果加上 00'0000'8000_H 来实现舍入操作；通过置位寄存器 MCW 中的位 MS 使能自动饱和操作。

当累加器处于溢出饱和模式，且发生溢出时，根据溢出方向和所采用的算术运算，将 32 位数据的最大正值或最大负值载入累加器。饱和操作后累加器的值为 00'7FFF'FFFF_H（正）或 FF'8000'0000_H（负）。

4.9.6 数据限制器

当利用 CoSTORE <destination>., MAS 指令读取累加器时，也可用饱和算法来限制溢出。对 MAC 单元的累加器执行限制操作。如果累加器内容可以用目的操作数的大小表示而不产生溢出，则数据限制器被禁用，操作数不被修改。如果累加器内容用目标操作数的大小来表示会产生溢出，限制器将用一个“限制”数据（见表 4-25）来代替累加器内容。

表 4-25 限制器输出

ME 标志	MN 标志	限制器输出
0	x	不变
1	0	7FFF _H
1	1	8000 _H

注：在这种特殊情况下，累加器和状态寄存器都不受影响。只能通过 CoSTORE 指令读取 MAS。

4.9.7 累加移位器

累加移位器是一个40位输入、40位输出的并行移位器。源累加器移位操作如下：

- 没有移位（不被修改）
- 最多 16 位算术左移
- 最多 16 位算术右移

注意寄存器 MSW 中的 ME、MSV、MSL 受左移影响；因此，如果饱和机制被使能（位 MS），其行为与加法/减法器相似。

注：在饱和使能的情况下执行左移操作，需要特别的注意。通常，如果 MAE 包含有效位，则累加器中的 32 位值将饱和。但是，左移操作有可能将某些有效位移出累加器。这种情况下，40 位结果值将被错误的解释，或者不被饱和，或者被错误地饱和。左移操作的结果还有可能将一个原来的正数饱和成最小的负数，反之亦然。

4.9.8 40 位有符号累加器寄存器

40 位累加器由三个寄存器 MAE、MAH、MAL 级联组成。其中，MAE 8 位宽，MAH 和 MAL16 位宽。MAE 是 40 位累加器的最高有效字节。该字节执行监控功能。MAE 位于 MSW 的低位字节。

当 MAH 被写入时，累加器的值自动调整扩展为带符号的 40 位格式。这意味着 MAL 被清零，对正数（MAH 的最高位为 0），向 MAE 中自动载入 00_H；对负数（MAH 最高位是 1），向 MAE 中自动载入 FF_H，以 2 补码格式表示被扩展的 40 位负数。可以看出，扩展的 40 位值等于未扩展的 32 位值。换句话说，扩展之后，MAE 不包含有效位。通常，当 40 位有符号结果的最高 9 位相同时，会出现这种情况。

累加器操作期间，可能发生溢出，导致无法用 32 位表示计算结果，这时 MAE 会改变。当累加器中的有符号结果已超出 32 位，MSW 中的扩展标志“E”置位。当 40 位有符号结果的最高 9 位不相同，出现这种情况，即 MAE 包含有效位。

大多数 CoXXX 操作使用 40 位累加器寄存器作为源和/或目的操作数。

MAH

累加器高位字								SFR (FE5E _H /2F _H)								复位值: 0000 _H			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
MAH																			
rwh																			

MAL

累加器低位字								SFR (FE5C _H /2E _H)								复位值: 0000 _H			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
MAL																			
rwh																			

符号	位序号	读写类型	功能描述
MAH、MAL	[15:0]	rwh	累加器的高位和低位字 与 MAE 级联构成 40 位累加器。

4.9.9 MAC 单元状态字 MSW

寄存器 MSW 的高字节（可位寻址）给出 MAC 单元的当前状态。寄存器 MSW 的低字节表示 MAC 累加器扩展的高 8 位，与寄存器 MAH、MAL 级联构成 40 位累加器。

MSW

MAC 状态字

SFR（FFDE_H/EF_H）

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	MV	MSL	ME	MSV	MC	MZ	MN	MAE							
-	rwh	rwh	rwh	rwh	rwh	rwh	rwh	rwh							

符号	位序号	读写类型	功能描述
MV	14	rwh	溢出标志 0 没有产生溢出 1 产生溢出
MSL	13	rwh	粘着限制标志 0 结果不饱和 1 结果饱和
ME	12	rwh	MAC 扩展标志 0 MAE 不包含有效位 1 MAE 包含有效位
MSV	11	rwh	粘着溢出标志 0 没有发生溢出 1 发生溢出
MC	10	rwh	进位标志 0 没有产生进位/借位 1 产生进位/借位
MZ	9	rwh	零标志 0 MAC 结果非零 1 MAC 结果为零
MN	8	rwh	负标志 0 MAC 结果为正

符号	位序号	读写类型	功能描述
			1 MAC 结果为负
MAE	[7:0]	rwh	MAC 累加器扩展 40 位累加器的最高 8 位，与寄存器 MAH 和 MAL 级联。

MAC 单元状态（MV、MN、MZ、MC、MSV、ME、MSL）

这些状态标志指示最近一次执行 MAC 操作之后 MAC 的状态。大多数指令会根据特定规则（这取决于执行 MAC 操作或是数据转移操作）设置这些标志位。

若通过指令显性更新 MSW 寄存器，则状态标志不再能代表真实的 MAC 状态。对 MSW 的显性设置将取代由 MAC 单元隐含产生的状态标志。用户读取 MSW 寄存器，将读出前一条指令执行之后的 MSW 状态。可以通过任何能够访问 SFR 的指令来访问寄存器 MSW。

注：复位之后，所有MAC状态位被清零。

MN 标志：对大多数 MAC 操作，如果结果的最高位为 1，则 MN 标志置 1；否则清零。整数操作的情况下，MN 标志可以理解成结果的符号位（负值：MN = 1，正值：MN = 0）。负数始终由相应正数的 2 补码来表示。有符号数的范围从 80'0000'0000_H 到 7F'FFFF'FFFF_H。

MZ 标志：通常如果 MAC 操作结果等于 0，MZ 标志置 1，否则清零。

MC 标志：在一次 MAC 加法运算之后，MC 标志指示累加器扩展字节 MAE 的最高位是否产生了“进位”。在一次 MAC 减法或 MAC 比较运算之后，MC 标志指示是否出现“借位”（表示加法“进位”的逻辑“非”）。这意味着，若执行减法运算时，累加器最高位没有产生“进位”，则 MC 标志置 1。MAC 单元通过 2 补码加法实现减法运算，当补码加法导致“进位”时，MC 标志清零。

左移 MAC 操作，MC 标志表示最后移出位的值；右移位 MAC 操作，MC 标志始终清零。对于算术右移 MAC 操作，如果舍入操作使累加器扩展字节 MAE 的最高位产生“进位”，则置位 MC。

MSV 标志：进行加法、减法、2 补码和舍入运算时，如果 MAC 结果超出 40 位有符号数的取值范围，则 V 标志始终置 1。如果 V 标志指示发生溢出，则 MAC 结果无效。

MSV 标志是一个“粘着位”。一旦置位，其状态不受其它 MAC 操作的影响。只有直接的写操作能够对其清零。

ME 标志：如果累加器扩展字节 MAE 包含有效位，ME 标志置位。也就是说，此时累加器最高 9 位不完全相等。

MSL 标志：如果累加器已执行了自动饱和操作，MSL 标志置位。寄存器 MCW 的位 MS 置位时，使能自动饱和。还可以由限制累加器内容的指令置位 MSL 标志。如果累加器内容被限制，则 MSL 标志置位。

MSL 是一个“粘着位”。一旦置位，其状态不受其它 MAC 操作的影响。只有直接的写操作能够对其清零。

MV 标志：如果加法、减法和累加操作的结果超出有符号数的最大范围（80'0000'0000_H ~ 7F'FFFF'FFFF_H），则 MV 标志置位；否则清零。请注意：如果 MV 标志指示算术溢出，此时整数加、整数减、或累加的结果无效。

4.9.10 重复计数器 MRW

重复计数器 MRW 控制循环的重复次数。该寄存器加载之后，才能被指令-USRx CoXXX 使用。MAC 操作能使该计数器递减。执行-USRx CoXXX 指令时，MRW 在递减之前检测是否计数到 0。如果 MRW 等于零，则 USRx 位置位，MRW 不再递减。可以通过任何能够访问 SFR 的指令对寄存器 MRW 进行访问。

MRW

MAC重复字					SFR (FFDA _H /ED _H)					复位值: 0000 _H					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
REPEAT_COUNT															
rwh															

符号	位序号	读写类型	功能描述
REPEAT_COUNT	[15:0]	rwh	16 位循环计数器

所有 CoXXX 指令在其操作数域中均有一个 3 位宽的重复控制域 'rrr'（位地址 [31:29]），用来控制 MRW 重复计数器。[表 4-26](#) 列出可能的编码。

表 4-26 MAC 重复字控制的编码

‘rrr’ 编码	控制重复计数器
000 _B	常规 CoXXX 指令
001 _B	保留
010 _B	‘-USR0 CoXXX’ 指令， 递减重复计数器，如果 MRW 为 0，置位 USR0
011 _B	‘-USR1 CoXXX’ 指令， 递减重复计数器，如果 MRW 为 0，置位 USR1
1XX _B	保留

注：位 USR0 在先前的体系结构中也被用作通用标志。为防止程序员或编译器使用该标志而产生冲突，使用 ‘-USR0 CoXXX’ 指令时必须非常小心。

下面的例子给出一个 20 次的循环操作。每执行一次 CoMACM 指令，MRW 计数器减 1。

```

MOV  MRW, #19                ; 预载循环计数器
loop01:
-USR1  CoMACM [IDX0+], [R0+]    ; 计算和递减MSW
      ADD R2, #0002H
      JMPA cc_nusr1, loop01      ; 重复循环，直到USR1置位为止

```

注：由于正确预测的 JMPA 指令执行时间为零周期，从而提供了重复指令的功能。

4.10 常数寄存器

这些可位寻址寄存器的所有位由硬件固定为 0 或 1，它们是只读寄存器。寄存器 ZEROS/ONES 中包含全 0 或全 1，可用于位操作或屏蔽码产生。可通过任何能寻址 SFR 的指令访问常数寄存器。

ZEROS

常数0寄存器

SFR (FF1CH/8EH)

复位值: 0000_H

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

ONES

常数1寄存器

SFR (FF1EH/8FH)

复位值: FFFF_H

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

5 中断与强制中断功能

XC164CM 提供了几种中断机制，可对各种中断服务请求作出快速、灵活的响应。这些中断请求由微控制器的内部或外部中断源产生。对各类异常情况的处理方式相似：

- 由中断控制器（ITC）产生中断
- 由外围事件控制器（PEC）发出 DMA 传送
- 由 TRAP 指令、特定系统状态或错误引发强制中断

正常中断处理

CPU 暂时挂起当前执行的程序，转入中断处理程序，服务中断请求设备。当前程序状态（IP、PSW 以及分段模式下的 CSP）被保存到内部系统堆栈中。用户可使用优先级排序机制（共分 16 级优先级），为多个要处理的中断请求指定不同的优先级。

通过外围事件控制器（PEC）进行中断处理

与正常软件控制的中断处理相比，采用集成的外围事件控制器（PEC）可以更快地服务中断请求设备。中断请求触发 PEC，通过 8 路可编程 PEC 服务通道中的一路，在任意两个地址单元之间进行一个字或一个字节的传送。PEC 传送数据期间，暂停 CPU 正常程序的执行，不需要保存内部程序状态信息。PEC 服务和正常中断处理使用相同的优先级排序机制。

强制中断功能

在指令执行期间出现异常情况时，强制中断功能被激活。还可通过非屏蔽中断引脚 **NMI** 外部激活强制中断。XC164CM 提供了几种硬件强制中断功能，用来处理指令执行期间可能出现的错误和异常。硬件强制中断的优先级最高，会被系统立即响应。软件强制中断功能由 TRAP 指令引发，它产生一个软件中断，具有特定的中断向量。对所有类型的强制中断，当前的程序状态都保存到系统堆栈中。

外部中断处理

尽管 XC164CM 未提供专用的中断引脚，但中断系统可和外部中断源连接，并提供多种外部事件（包括标准输入、非屏蔽中断、快速外部中断等）响应机制。除屏蔽中断和复位输入之外，其它中断功能均为端口的复用功能。

5.1 中断系统结构

XC164CM 具有 80 个独立的中断节点，构成 16 级中断优先级，每级内又分 8 个子级（组优先级）。为支持软件设计的模块化和一致性，大多数中断或 PEC 请求源都对应有独立的中断控制寄存器和中断向量。中断控制寄存器包含中断请求标志、中断使能位和相关中断源的中断优先级。每个中断源请求由特定事件激活，这由相应器件所选择的工作模式决定。为了有效的利用资源，多个中断源的中断节点可以合并成一个。这些中断节点可以由多个中断源请求激活，例如串行接口中的各种错误。不过，可以通过串行通道控制寄存器中的特定状态标志来指示错误类型。中断子节点控制寄存器支持中断节点的共享。

XC164CM 具有一个向量化的中断系统。在该系统中，存储空间中的特定向量地址保留用作复位、强制中断和中断服务功能。一旦发生中断请求，CPU 会跳转到相应中断源所对应的向量地址上。这样可以直接指示引起该请求的中断源。B 类硬件强制中断共用同一个中断向量。由强制中断标志寄存器（TFR）中的状态标志来确定引起强制中断的异常事件类型。特殊的软件 TRAP 指令的向量地址由指令中的操作数域指定，对应一个 7 位的强制中断编号。

这些保留的向量地址构成一个跳转表，位于 XC164CM 地址空间由寄存器 VECSEG 选定的某段的低位地址段。跳转表由适当的跳转指令组成，跳转指令将 CPU 的控制权移交给中断或强制中断服务程序。这些服务程序可以位于地址空间的任何位置。跳转表的入口指向所选代码段的最低端。每个入口占用 2、4、8 或 16 个字（由 CPUCON1 寄存器中的位域 VECSC 决定），至少能够存放一条双字指令。相应的中断向量地址可以通过中断向量编号乘以所选的中断向量间距 ($2^{(VECSC+2)}$) 计算得到。

所有挂起的中断请求都要经过仲裁。仲裁胜出的中断请求连同其中断优先级和操作请求一起发送给 CPU。CPU 根据仲裁胜出者所请求的功能（正常中断、PEC、跳转表缓存等）触发相应的操作。

中断全局使能的情况下，如果中断请求源的优先级高于当前 CPU 的优先级，CPU 会接受该操作请求；如果中断请求源的优先级比当前 CPU 的优先级低（或相等），该中断请求挂起。

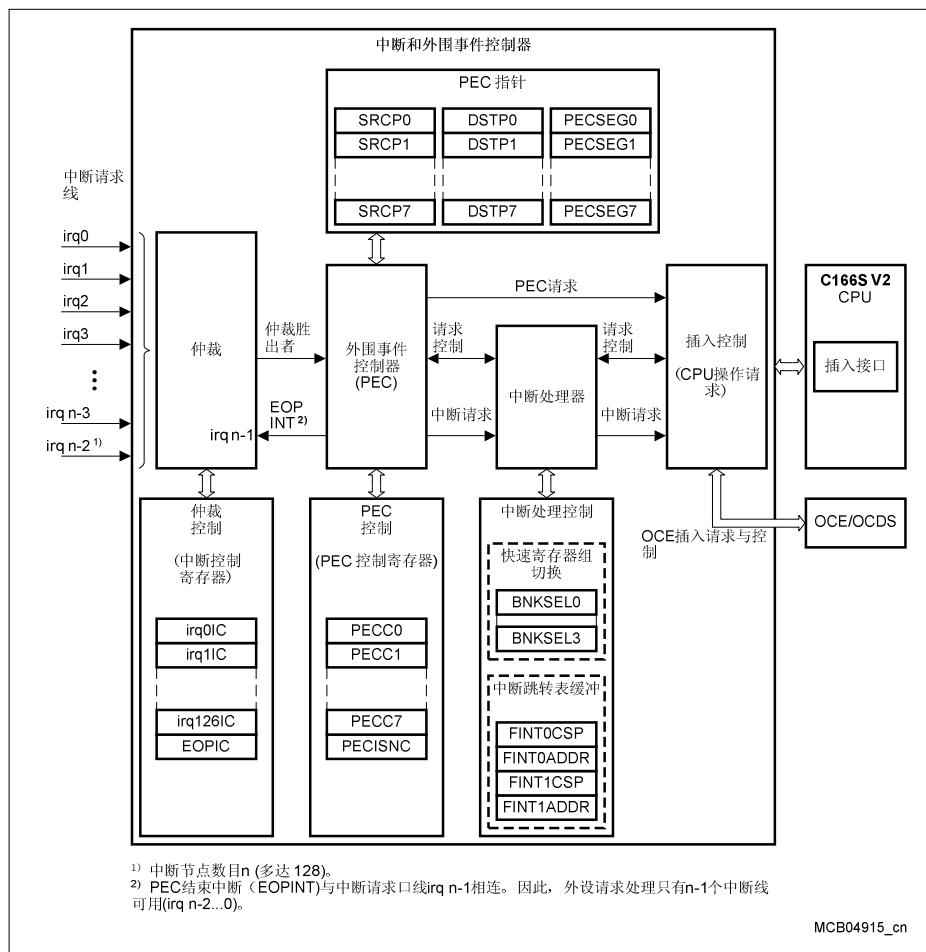


图 5-1 中断和 PEC 控制器框图

5.2 中断仲裁与控制

XC164CM 中断仲裁系统能够处理来自多达 80 个中断源的中断请求。中断请求可由片上外设或外部输入触发。

由寄存器 PSW 的全局中断使能位 (IEN) 和 CPU 优先级选择位域 (ILVL) 全局控制中断的处理。此外，不同的中断源由各自的中断控制寄存器 (...IC) 单独控制。因此，CPU 是否接受中断请求由各自的中断控制寄存器和 PSW 共同决定。PEC 服务由寄存器 PECCx、源指针和目的指针控制，这三者共同确定相应 PEC 服务通道的任务。

中断请求置位相关的中断请求标志 **xxIR**。如果中断请求节点被相关的中断使能位 **xxIE** 使能，那么将在下一个时钟周期、或完成正在执行的仲裁周期之后开始中断仲裁。新的仲裁周期开始时，会对所有挂起的中断请求进行仲裁，与这些中断请求何时产生无关。

图 5-2 给出三级中断优先级排序机制。

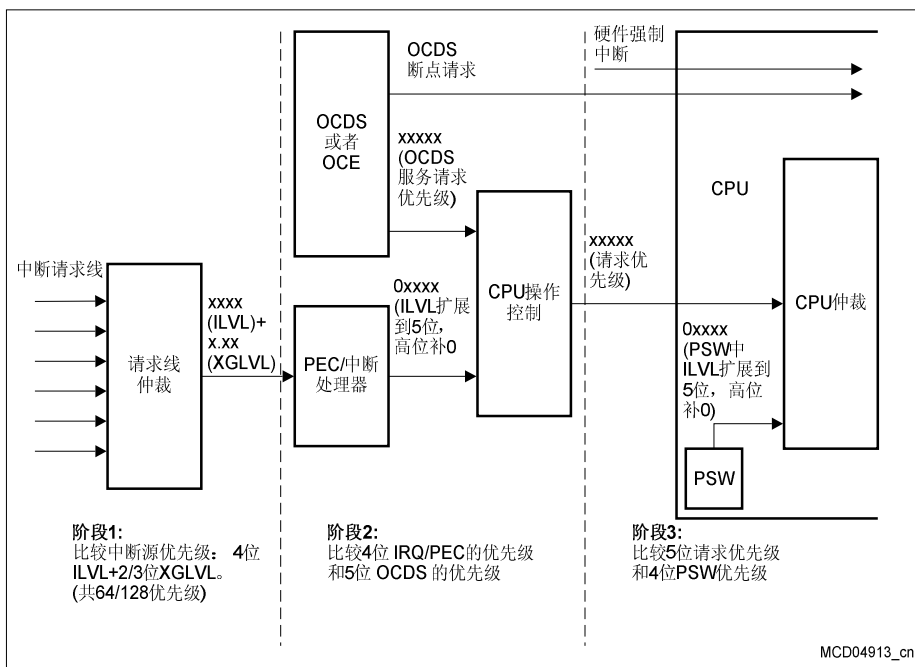


图 5-2 中断仲裁

中断优先级排序可以分为三个阶段：

- 选择一个有效的中断请求
- 将所选择的中断请求的优先级与 OCDS 服务请求的优先级进行比较
- 将最终胜出的中断请求的优先级与 CPU 优先级进行比较

仲裁第一阶段

在仲裁第一阶段，比较所有有效中断请求线的优先级。每个中断请求的优先级由相应寄存器 **xxIC** 中的位域 **ILVL** 决定。扩展组优先级 **XGLVL**（由位域 **GPX** 和 **GLVL** 组合构成）在同一优先级内定义了多达 8 个子优先级。根据组优先级可以区分同一优先级内中断请求的优先级次序，最终可确定一个中断请求胜出。

注：所有被使能且设定有相同中断优先级（ILVL）的中断请求必须具有不同的组优先级。否则会产生错误的中断向量。

仲裁第二阶段

在仲裁第二阶段，将第一阶段胜出的中断请求和 OCDS 服务请求的优先级相比较。OCDS 服务请求无需经过第一阶段，直接进入 CPU 操作控制单元。CPU 操作控制单元将前一阶段胜出的中断请求的 4 位优先级（忽略组优先级）和 OCDS 服务请求的 5 位优先级进行比较。比较前，先将中断请求的 4 位 **ILVL** 扩展为 5 位，最高位补 0（**MSB=0**）。这意味着，只要 OCDS 服务请求的优先级最高位为 1（**MSB=1**），OCDS 请求总会在仲裁第二阶段胜出。但是，当 OCDS 请求和与中断请求发生冲突时，中断请求胜出。

仲裁第三阶段

在仲裁第三阶段，将第二阶段胜出的中断请求与当前 CPU 任务的优先级相比较。只有在中断请求的优先级高于当前 CPU 优先级（由 **PSW** 寄存器中位域 **ILVL** 决定），并且中断和 **PEC** 请求被全局使能（**PSW** 寄存器中的全局中断使能位 **IEN** 置位）的情况下，中断请求才会被 CPU 接受。第二阶段胜出的中断请求的优先级为 5 位，CPU 4 位优先级与之比较时需要先扩展为 5 位，最高位补 0（**MSB=0**）。这意味着，只要中断请求的优先级最高位为 1（**MSB=1**），该请求总会中断 CPU 的当前任务。若中断请求的优先级低于或等于当前 CPU 任务的优先级，请求保持挂起状态。

*注：CPU 缺省的优先级为 0000_B，因此当请求优先级为 0000_B 时，虽然进行仲裁，但该中断请求永远不会被 CPU 接受。然而，每个各自被使能的中断请求（包括所有被 CPU 拒绝的中断请求和优先级为 0000_B 的中断请求）都会将 CPU 从空闲状态唤醒，而与全局中断使能位 **IEN** 无关。*

OCDS 断点请求和硬件强制中断都不经过仲裁机制，直接进入内核（见图 5-2）。

只要有使能的中断请求，就会启动仲裁过程；只要有挂起的中断请求，仲裁就会持续进行。当没有中断挂起时，停止仲裁以节约功耗。

中断控制寄存器

各中断节点由各自的专用中断控制寄存器（xxIC，其中“xx”代表各节点）来控制。所有中断控制寄存器的结构都相同。中断控制寄存器的低 9 位包含了在进行一次优先级仲裁时，相关中断源的所有中断控制和状态信息，高 7 位保留待用。所有的中断控制寄存器都可位寻址，所有位都能由软件读写。因此，只需要一条指令即可对每个中断源进行编程或修改。

xxIC

中断控制寄存器

(E)SFR (yyyy_H/zz_H)

复位值: - 000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	GPX	xxIR	xxIE	ILVL			GLVL		
-	-	-	-	-	-	-	rw	rwh	rw	rw			rw		

符号	位序号	读写类型	功能描述
GPX	8	rw	组优先级扩展 将位域 GLVL 扩展到 3 位组优先级。
xxIR¹⁾	7	rwh	中断请求标志 0 无挂起的中断请求 1 该中断源已产生中断请求
xxIE	6	rw	中断使能控制位 (单独使能/禁止特定的中断源) 0 中断请求被禁止 1 中断请求被使能
ILVL	[5:2]	rw	中断优先级 F _H 最高优先级 0 _H 最低优先级
GLVL	[1:0]	rw	组优先级

符号	位序号	读写类型	功能描述
			（与 GPX 组合，扩展为 3 位组优先级）
			3 _H 最高优先级
		
			0 _H 最低优先级

1) 位 xxIR 具有位保护功能。

指令访问中断控制寄存器时，操作单位是字类型，寄存器的高 7 位（15...9）读取时返回 0，写入时被忽略。建议对这些位写 0。以下介绍的中断控制寄存器的结构适用于每个 xxIC 寄存器（“xx”代表各中断请求源）。

中断请求标志：一旦中断源请求相关的中断服务，中断请求标志由硬件置位。一旦进入中断服务程序或 PEC 服务，请求标志被自动清零。在 PEC 服务时，如果所选 PEC 通道的控制寄存器 PECCx 中位域 COUNT 由 1 减到 0 且 EOPINT 清零，中断请求标志保持置位状态。这样可产生一个具有相同优先级的正常 CPU 中断，以响应 PEC 传送的完成。

注：通过软件修改中断请求标志和通过硬件置位或清零的效果相同。

中断使能控制位：决定中断节点是（使能）否（禁止）参与仲裁。无论中断是否被使能，只要产生中断请求，相关的请求标志就会置位。因此，即使在节点被禁止的情况下，也可以通过查询 xxIR 来判断是否产生中断请求。

注：在这种情况下，xxIR 不会自动清零，必须由软件清零。

中断优先级与组优先级

4 位位域 ILVL 指定中断请求的优先级。ILVL 值越大，优先级越高。因此，0000_B 代表的优先级最低，1111_B 最高。

当多个具有相同优先级的中断请求被激活时，需根据位域 GPX 和 GLVL 级联构成组优先级来进行二级仲裁，选出其中组优先级最高的中断请求。同样，组优先级（位域 GPX 和 GLVL 级联构成）值越大，优先级越高。因此，000_B 代表的组优先级最低，111_B 最高。

注：所有使能且设定有相同中断优先级（ILVL）的中断请求必须具有不同的组优先级。否则会产生错误的中断向量。

进入中断服务程序时，将仲裁胜出的中断源的优先级（高于 CPU 当前优先级）复制到 PSW 寄存器的位域 ILVL 中，在该操作之前先将旧的 PSW 内容压入堆栈。

XC164CM 中断系统能够处理多达 15 层嵌套的优先级不同的中断服务程序（0 优先级不会被仲裁）。

若相关 PEC 通道被使能且配置适当（请参阅**章节 5.4**），优先级为 15...8 的中断请求（即 $ILVL=1XXXX_B$ ）可以由 PEC 服务。优先级为 7...1 的中断请求只能按照正常中断进行处理。

注：CPU 缺省的优先级为 0000_B，因此，优先级为 0 的中断请求无法中断 CPU，不会被 CPU 接受。不过，被单独使能的（和位 IEN 无关）、优先级为 0000_B 的中断请求会终止 CPU 的空闲状态，重新激活 CPU。

寄存器 PSW 的全局中断控制功能

是否接受中断请求取决于 CPU 的当前优先级（PSW 寄存器的位域 ILVL）以及 PSW 寄存器中的全局中断使能控制位 IEN（见**章节 4.8**）。

CPU 优先级位域 ILVL 定义了当前 CPU 操作的优先级。该位域指示当前执行的程序的优先级。进入中断服务程序后，该位域更新为被响应的中断请求的优先级。服务中断请求之前，PSW 被保存到系统堆栈中。CPU 优先级决定了 CPU 能够响应的最低优先级。任何低于或等于 CPU 优先级的请求不予应答。CPU 当前优先级可以通过软件进行修改，用来控制哪些中断请求源能够被应答。PEC 传送不会真正中断 CPU，只是“窃取”一个时钟周期而已。因此，PEC 服务并不会影响 PSW 中的 ILVL 位域。

硬件强制中断将 CPU 优先级切换到最高级（即 15），从而在执行异常强制中断服务程序时，不会响应任何中断和 PEC 请求。

注：TRAP 指令不改变 CPU 的优先级，软件调用强制中断服务程序时，可能被更高优先级的请求中断。

中断使能位 IEN 全局使能或禁止 PEC 操作以及正常 CPU 中断操作。当 IEN 清零时，CPU 不接受任何新的中断请求（见**章节 4.3.4**）。当 IEN 置位时，所有已被分别使能的中断请求源将被全局使能（通过各自中断控制寄存器中的中断使能位来分别使能各中断请求源）。强制中断属于非屏蔽中断，因此不受 IEN 的影响。

注：要产生中断请求，中断源还必须由相关控制寄存器中的中断使能控制位使能。

寄存器组选择位域 BANK 决定 CPU 操作当前使用的寄存器组。当 CPU 进入中断服务程序时，该位域被更新，选择与中断服务请求相关的寄存器组：

- 优先级为 15...12 的请求使用由相应 BNKSEL 寄存器中位域 GPRSELx 选择的寄存器组
- 优先级为 11...1 的请求始终使用全局寄存器组，即 $BANK = 00_B$
- 硬件强制中断始终使用全局寄存器组，即 $BANK = 00_B$
- TRAP 指令不改变当前寄存器组

5.3 中断向量表

XC164CM 具有一个向量化的中断系统。该系统保留了一组特定的存储单元，发生相应的触发事件时，系统会自动访问这些单元。触发事件包括：

- 复位（硬件、软件、看门狗）
- 强制中断（系统出错触发的硬件强制中断或 TRAP 软件强制中断）
- 中断服务请求

一旦接受请求，CPU 将跳转到触发源的相关地址。通过向量位置可直接确认引起请求的中断源，但以下**两种情况例外**：

- B 类硬件强制中断共用一个中断向量。强制中断标志寄存器（TFR）中的状态标志用于确定是哪种异常事件引起了强制中断，详见**章节 5.11**。
- 多个中断请求（例如一个模块内的多个中断请求）可能会共用一个中断节点。此时，将通过附加的标志位来识别请求源，从而可通过软件分别对每个请求源进行处理。详见**章节 5.7**。

这些保留的向量地址在 XC164CM 地址空间中构成一个向量表。向量表中包含着适当的跳转指令，跳转指令将 CPU 的控制移交给中断或强制中断服务程序。这些服务程序可以位于地址空间的任何位置。向量表的位置和结构可以编程设定。

向量段寄存器 VECSEG 定义了向量表所在的段（可以使用保留区之外的所有段）。

寄存器 CPUCON1 中的位域 VECSC 定义了两个相邻向量之间的间距（可以为 2、4、8 或 16 个字）。CPUCON1 寄存器的描述见**章节 4.4**。

每个向量都有一个相对于向量表段基地址（由 VECSEG 决定）的偏移地址。该偏移量可通过向量编号乘以由 VECSC 设定的向量间距计算得到。

表 5-2 列出 XC164CM 中的所有中断请求源或 PEC 服务请求源、对应的中断向量地址、中断向量编号，以及中断控制寄存器。

注：所有当前未被相关模块使用或在实际衍生产品中未和模块连接的中断节点，都可以通过软件置位相应 IR 标志产生软件控制的中断请求。

VECSEG

向量段指针

SFR (FF12_H/89_H)

复位值: 见表 5-1

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

符号	位序号	读写类型	功能描述
vecseg	[7:0]	rwh	向量表的段编号

寄存器 VECSEG 的复位值（即向量表的初始位置）由复位配置决定。表 5-1 列出向量表可能的初始位置。由于向量表提供复位向量，因此对系统而言，必须知道向量表的初始位置。

表 5-1 寄存器 VECSEG 复位值

初始值	复位配置
00C0 _H	从内部程序存储器开始的标准启动
00E0 _H	执行引导程序代码

表 5-2 XC164CM 中断节点

中断源或 PEC 服务请求	控制寄存器	向量地址 ¹⁾	向量编号
EX0IN	CC1_CC8IC	xx'0060 _H	18 _H / 24 _D
EX1IN	CC1_CC9IC	xx'0064 _H	19 _H / 25 _D
EX2IN	CC1_CC10IC	xx'0068 _H	1A _H / 26 _D
EX3IN	CC1_CC11IC	xx'006C _H	1B _H / 27 _D
EX4IN	CC1_CC12IC	xx'0070 _H	1C _H / 28 _D
EX5IN	CC1_CC13IC	xx'0074 _H	1D _H / 29 _D
CAPCOM 寄存器 16	CC2_CC16IC	xx'00C0 _H	30 _H / 48 _D
CAPCOM 寄存器 17	CC2_CC17IC	xx'00C4 _H	31 _H / 49 _D
CAPCOM 寄存器 18	CC2_CC18IC	xx'00C8 _H	32 _H / 50 _D
CAPCOM 寄存器 19	CC2_CC19IC	xx'00CC _H	33 _H / 51 _D
CAPCOM 寄存器 20	CC2_CC20IC	xx'00D0 _H	34 _H / 52 _D
CAPCOM 寄存器 21	CC2_CC21IC	xx'00D4 _H	35 _H / 53 _D
CAPCOM 寄存器 22	CC2_CC22IC	xx'00D8 _H	36 _H / 54 _D
CAPCOM 寄存器 23	CC2_CC23IC	xx'00DC _H	37 _H / 55 _D
CAPCOM 寄存器 24	CC2_CC24IC	xx'00E0 _H	38 _H / 56 _D
CAPCOM 寄存器 25	CC2_CC25C	xx'00E4 _H	39 _H / 57 _D
CAPCOM 寄存器 26	CC2_CC26IC	xx'00E8 _H	3A _H / 58 _D
CAPCOM 寄存器 27	CC2_CC27IC	xx'00EC _H	3B _H / 59 _D
CAPCOM 寄存器 28	CC2_CC28IC	xx'00F0 _H	3C _H / 60 _D
CAPCOM 寄存器 29	CC2_CC29IC	xx'0110 _H	44 _H / 68 _D
CAPCOM 寄存器 30	CC2_CC30IC	xx'0114 _H	45 _H / 69 _D
CAPCOM 寄存器 31	CC2_CC31IC	xx'0118 _H	46 _H / 70 _D
CAPCOM 定时器 T7	CC2_T7IC	xx'00F4 _H	3D _H / 61 _D
CAPCOM 定时器 T8	CC2_T8IC	xx'00F8 _H	3E _H / 62 _D

中断源或 PEC 服务请求	控制寄存器	向量地址 ¹⁾	向量编号
GPT1 定时器 T2	GPT12E_T2IC	xx'0088 _H	22 _H / 34 _D
GPT1 定时器 T3	GPT12E_T3IC	xx'008C _H	23 _H / 35 _D
GPT1 定时器 T4	GPT12E_T4IC	xx'0090 _H	24 _H / 36 _D
GPT2 定时器 T5	GPT12E_T5IC	xx'0094 _H	25 _H / 37 _D
GPT2 定时器 T6	GPT12E_T6IC	xx'0098 _H	26 _H / 38 _D
GPT2 CAPREL 寄存器	GPT12E_CRIC	xx'009C _H	27 _H / 39 _D
A/D 转换完成	ADC_CIC	xx'00A0 _H	28 _H / 40 _D
A/D 过载错误	ADC_EIC	xx'00A4 _H	29 _H / 41 _D
ASC0 发送	ASC0_TIC	xx'00A8 _H	2A _H / 42 _D
ASC0 发送缓存	ASC0_TBIC	xx'011C _H	47 _H / 71 _D
ASC0 接收	ASC0_RIC	xx'00AC _H	2B _H / 43 _D
ASC0 错误	ASC0_EIC	xx'00B0 _H	2C _H / 44 _D
ASC0 自动波特率	ASC0_ABIC	xx'017C _H	5F _H / 95 _D
SSC0 发送	SSC0_TIC	xx'00B4 _H	2D _H / 45 _D
SSC0 接收	SSC0_RIC	xx'00B8 _H	2E _H / 46 _D
SSC0 错误	SSC0_EIC	xx'00BC _H	2F _H / 47 _D
PLL/OWD	PLL_IC	xx'010C _H	43 _H / 67 _D
ASC1 发送	ASC1_TIC	xx'0120 _H	48 _H / 72 _D
ASC1 发送缓存	ASC1_TBIC	xx'0178 _H	5E _H / 94 _D
ASC1 接收	ASC1_RIC	xx'0124 _H	49 _H / 73 _D
ASC1 错误	ASC1_EIC	xx'0128 _H	4A _H / 74 _D
ASC1 自动波特率	ASC1_ABIC	xx'0108 _H	42 _H / 66 _D
PEC 结束子通道	EOPIC	xx'0130 _H	4C _H / 76 _D
CAPCOM6 定时器 T12	CCU6_T12IC	xx'0134 _H	4D _H / 77 _D
CAPCOM6 定时器 T13	CCU6_T13IC	xx'0138 _H	4E _H / 78 _D

中断源或 PEC 服务请求	控制寄存器	向量地址 ¹⁾	向量编号
CAPCOM6 紧急事件	CCU6_EIC	xx'013C _H	4F _H / 79 _D
CAPCOM6	CCU6_IC	xx'0140 _H	50 _H / 80 _D
SSC1 发送	SSC1_TIC	xx'0144 _H	51 _H / 81 _D
SSC1 接收	SSC1_RIC	xx'0148 _H	52 _H / 82 _D
SSC1 错误	SSC1_EIC	xx'014C _H	53 _H / 83 _D
CAN0	CAN_0IC	xx'0150 _H	54 _H / 84 _D
CAN1	CAN_1IC	xx'0154 _H	55 _H / 85 _D
CAN2	CAN_2IC	xx'0158 _H	56 _H / 86 _D
CAN3	CAN_3IC	xx'015C _H	57 _H / 87 _D
CAN4	CAN_4IC	xx'0164 _H	59 _H / 89 _D
CAN5	CAN_5IC	xx'0168 _H	5A _H / 90 _D
CAN6	CAN_6IC	xx'016C _H	5B _H / 91 _D
CAN7	CAN_7IC	xx'0170 _H	5C _H / 92 _D
RTC	RTC_IC	xx'0174 _H	5D _H / 93 _D
未分配节点	---	xx'0040 _H	10 _H / 16 _D
未分配节点	---	xx'0044 _H	11 _H / 17 _D
未分配节点	---	xx'0048 _H	12 _H / 18 _D
未分配节点	---	xx'004C _H	13 _H / 19 _D
未分配节点	---	xx'0050 _H	14 _H / 20 _D
未分配节点	---	xx'0054 _H	15 _H / 21 _D
未分配节点	---	xx'0058 _H	16 _H / 22 _D
未分配节点	---	xx'005C _H	17 _H / 23 _D
未分配节点	---	xx'0078 _H	1E _H / 30 _D
未分配节点	---	xx'007C _H	1F _H / 31 _D
未分配节点	---	xx'0080 _H	20 _H / 32 _D
未分配节点	---	xx'0084 _H	21 _H / 33 _D

中断源或 PEC 服务请求	控制寄存器	向量地址 ¹⁾	向量编号
未分配节点	---	xx'00FC _H	3F _H / 63 _D
未分配节点	---	xx'0100 _H	40 _H / 64 _D
未分配节点	---	xx'0104 _H	41 _H / 65 _D
未分配节点	---	xx'012C _H	4B _H / 75 _D
未分配节点	---	xx'0160 _H	58 _H / 88 _D

1) 寄存器 VECSEG 决定向量表所在的段。

CPUCON1 寄存器中的位域 VECSC 定义了两个相邻向量之间的间距。该表使用 VECSC 的缺省值，即两个相邻向量之间的间距为 4 个字节（两个字）。

表 5-3 列出硬件强制中断的向量地址以及存放在 TFR 寄存器中的相关状态标志。

在同一条指令内可能检测到多个强制中断，该表还列出了这些情况下强制中断服务的优先级。任何复位操作（硬件复位、软件复位指令 SRST 或看门狗定时器溢出复位）之后，程序从位于 xx'0000_H 的复位向量开始执行。复位操作比其它任何系统操作的优先级都高，因此具有最高优先级（强制中断优先级 III）。

软件强制中断的向量地址可以是任意确定的向量地址。通过软件 TRAP 指令执行中断服务程序时，其优先级始终为当前 CPU 优先级，由寄存器 PSW 的位域 ILVL 指示。这意味着所有的硬件强制中断或更高优先级的中断请求都会中断软件 TRAP 指令。

表 5-3 硬件强制中断总结

异常条件	强制中断标志	强制中断向量	向量地址 ¹⁾	向量编号	强制中断优先级
复位功能： <ul style="list-style-type: none"> • 硬件复位 • 软件复位 • 看门狗定时器溢出 	—	RESET RESET RESET	xx'0000H xx'0000H xx'0000H	00H 00H 00H	III III III
A 类硬件强制中断 <ul style="list-style-type: none"> • 非屏蔽中断 • 堆栈上溢 • 堆栈下溢 • 软件断点 	NMI STKOF STKUF SOFTBRK	NMITRAP STOTRAP STUTRAP SBRKTRAP	xx'0008H xx'0010H xx'0018H xx'0020H	02H 04H 06H 08H	II II II II
B 类硬件强制中断 <ul style="list-style-type: none"> • 未定义操作码 • PMI 访问错误 • 受保护指令错误 • 非法字操作数访问 	UNDOPC PACER PRTFLT ILLOPA	BTRAP BTRAP BTRAP BTRAP	xx'0028H xx'0028H xx'0028H xx'0028H	0AH 0AH 0AH 0AH	I I I I
保留	—	—	[2CH-3CH]	[0BH-0FH]	—
软件强制中断 <ul style="list-style-type: none"> • TRAP 指令 	—	—	任意 ¹⁾	任意 [00H-7FH]	当前 CPU 优先级

1) 寄存器 VECSEG 决定向量表所在的段。

CPUCON1 寄存器中的位域 VECSC 定义了两个相邻向量之间的间距。该表使用 VECSC 的缺省值，即两个相邻向量之间的间距为 4 个字节（两个字）。

中断跳转表缓存

通过向量表服务中断请求时，会引发两条连续的跳转指令：一条跳转到中断向量地址的隐含跳转指令和一条跳转到服务程序真实地址的跳转指令。采用中断跳转表缓存（ITC，也称作“快速中断”）可以缩短中断响应时间。该特性通过直接为 CPU 提供中断服务程序的地址来实现，此时无需再使用第二条跳转指令。

ITC 提供两个 24 位指针，CPU 能够直接跳转到相关的服务程序中。系统可以将快速中断分配给两个中断源，中断源优先级必须在 15...12 之间。

两个指针分别存放在一对中断跳转表缓存寄存器（FINTxADDR、FINTxCSP）中。每对寄存器中包含指针的段地址、偏移地址以及所分配的优先级（选择与相关中断节点设定的优先级相同）。

FINT0ADDR

快速中断地址寄存器 0																XSFR (EC02H/--)																复位值: 0000H															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																
ADDR																0																															
rw																r																															

FINT1ADDR

快速中断地址寄存器 1																XSFR (EC06H/--)																复位值: 0000 _H															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0																																
ADDR																0																															
rw																r																															

符号	位序号	读写类型	功能描述
ADDR	[15:1]	rw	中断服务程序地址 定义指向中断服务程序的 24 位地址指针的第 15...1 位。该字偏移量与 FINTxCSP.SEG 串联组成 24 位指针。

FINT0CSP

快速中断控制寄存器 0						XSFR (EC00H/--)						复位值: 0000H					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0		
EN	-	-	GPX	ILVL	GLVL	SEG											
rw	-	-	rw	rw	rw	rw											

FINT1CSP

快速中断控制寄存器 1

XSFR (EC04H/--)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EN	-	-	GPX	ILVL	GLVL	SEG									
rw	-	-	rw	rw	rw	rw									

符号	位序号	读写类型	功能描述
EN	15	rw	快速中断使能 0 禁止使用中断跳转表缓存 1 使能中断跳转表缓存，中断请求对应的向量表入口被旁路，使用缓存指针。
GPX	12	rw	组优先级扩展 与 GLVL 联合使用。
ILVL	[11:10]	rw	中断优先级 该位域选择分配给该指针的中断请求的优先级（15...12）。 00 中断优先级 12（1100 _B ） 01 中断优先级 13（1101 _B ） 10 中断优先级 14（1110 _B ） 11 中断优先级 15（1111 _B ）
GLVL	[9:8]	rw	组优先级 与 GPX 共同确定分配给该指针的中断请求的组优先级。
SEG	[7:0]	rw	中断服务程序段编号 定义指向中断服务程序的 24 位指针的第 23...16 位，与 FINTxADDR 串联组成 24 位指针。

5.4 外围事件控制器通道的操作

XC164CM 外围事件控制器（PEC）提供 8 路 PEC 服务通道，在任意两个地址单元之间实现一个字节或字的传送。PEC 传送可由中断服务请求触发，具有最快的中断响应。在很多情况下，PEC 传送足以满足相应外设（如串行通道等）的需要。

PEC 传送并不改变当前上下文，只是“窃取”CPU 的几个时钟周期，所以不需要像标准中断那样对当前程序状态和上下文进行保存和恢复操作。

每路 PEC 通道由与之对应的一组专用寄存器来控制：

- 每路通道的 24 位源指针
- 每路通道的 24 位目的指针
- 每路通道的通道计数器/控制寄存器（PECCx），选择相应通道的工作模式
- 两个中断控制寄存器，控制数据块传送操作

PECC 寄存器控制相应 PEC 通道执行的操作。

传送大小（位 BWT）控制在每个 PEC 服务周期内传送一个字节或一个字，这将决定传送数据的大小和指针修改的递增步长。

指针修改（位域 INC）控制哪些 PEC 指针会在 PEC 传送后递增。如果指针未修改（INC = 00_B），该通道将始终在同一源地址和同一目的地址之间传送数据。

传送控制（位域 COUNT）控制 PEC 通道在 PEC 传送后是否保持有效。位域 COUNT 通常使能一路 PEC 通道（COUNT > 00_H）。

PECC 寄存器还为 PEC 通道分配中断优先级（位域 PLEV）；控制 PEC 传送完毕后的中断操作（位 EOPINT）。

注：所有使能且分配给 PEC 服务的中断请求源应该使用不同的 PEC 通道。否则，同时发生中断请求时，只执行一次传送。当 COUNT 减至 00_H 时，CPU 会被中断，将产生错误的中断向量。

仅当 PEC 优先级高于 CPU 优先级时，才会执行 PEC 传送。

PECCx

PEC 控制寄存器

SFR (FECyH/6zH, 表 5-4)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	EOP INT	PLEV	CL	INC	BWT	COUNT									
-	rw	rw	rw	rw	rw	rwh									

符号	位序号	读写类型	功能描述
EOPINT	14	rw	PEC 结束中断选择 0 相同 (PEC) 优先级上的 PEC 结束中断 1 通过独立节点 EOPIC 产生 PEC 结束中断
PLEV	[13:12]	rw	PEC 优先级选择 该位域控制 PEC 通道优先级的分配 (见下一节)。
CL	11	rw	通道链接控制 0 PEC 通道各自独立工作 1 PEC 通道成对链接工作 ¹⁾
INC	[10:9]	rw	递增步长控制 (指针修改) ²⁾ 00 不修改指针 01 DSTPx 加 1 或 2 (BWT = 1 或 0) 10 SRCPx 加 1 或 2 (BWT = 1 或 0) 11 DSTPx 和 SRCPx 都加 1 或 2
BWT	8	rw	字/字节传送选择 0 传送一个字 1 传送一个字节
COUNT	[7:0]	rwh	PEC 传送计数 对 PEC 传送次数计数, 控制通道的操作 (见章节 5.4.2)。

1) 功能描述见“支持数据链的通道链接模式”。

2) 仅在当前段内递增/递减指针。

表 5-4 PEC 控制寄存器地址

寄存器	地址	寄存器区	寄存器	地址	寄存器区
PECC0	FEC0 _H / 60 _H	SFR	PECC4	FEC8 _H / 64 _H	SFR
PECC1	FEC2 _H / 61 _H	SFR	PECC5	FECA _H / 65 _H	SFR
PECC2	FEC4 _H / 62 _H	SFR	PECC6	FECC _H / 66 _H	SFR
PECC3	FEC6 _H / 63 _H	SFR	PECC7	FECE _H / 67 _H	SFR

PEC 通道编号从对应的位域 ILVL（LSB）和 GLVL 得到，由 PEC 通道的位域 PLEV 选择通道所能使用的优先级（ILVL）（见**表 5-5**）。因此，当 PLEV = 00_B，中断源优先级设定为 15（ILVL = 1111_B）时，可以选择 PEC 通道组 7...4；当 PLEV = 00_B，中断源优先级设定为 14 时（ILVL=1110_B），可以选择 PEC 通道组 3...0；当 PLEV = 10_B，中断源的优先级设定为 10 时（ILVL=1010_B），可以选择 PEC 通道组 3...0。最终由组优先级确定 PEC 通道编号（组优先级 3...0，即 GPX=0）。

根据 PEC 通道编号对同时发出的 PEC 通道请求进行优先级排序，通道 0 优先级最低，通道 7 优先级最高。

注：所有请求 PEC 服务的中断源必须使用不同的 PEC 通道。否则，将可能激活错误的 PEC 通道。

表 5-5 PEC 通道分配

所选 PEC 通道	组优先级	根据位域 PLEV 决定使用的中断优先级			
		PLEV = 00 _B	PLEV = 01 _B	PLEV = 10 _B	PLEV = 11 _B
7	3	15	13	11	9
6	2				
5	1				
4	0				
3	3	14	12	10	8
2	2				
1	1				
0	0				

表 5-6 给出在中断控制器寄存器和 PEC 通道设置不同时，系统的操作示例。

表 5-6 中断优先级示例

优先级		服务类型		
中断优先级	组优先级	COUNT = 00 _H , PLEV = XX _B	COUNT ≠ 00 _H , PLEV = 00 _B	COUNT ≠ 00 _H , PLEV = 01 _B
1111	111	CPU 中断 优先级 15, 组优先级 7	CPU 中断 优先级 15, 组优先级 7	CPU 中断 优先级 15, 组优先级 7
1111	011	CPU 中断 优先级 15, 组优先级 3	PEC 服务 通道 7	CPU 中断 优先级 15, 组优先级 3
1111	010	CPU 中断 优先级 15, 组优先级 2	PEC 服务 通道 6	CPU 中断 优先级 15, 组优先级 2
1110	010	CPU 中断 优先级 14, 组优先级 2	PEC 服务 通道 2	CPU 中断 优先级 14, 组优先级 2
1101	110	CPU 中断 优先级 13, 组优先级 6	CPU 中断 优先级 13, 组优先级 6	CPU 中断 优先级 13, 组优先级 6
1101	010	CPU 中断 优先级 13, 组优先级 2	CPU 中断 优先级 13, 组优先级 2	PEC 服务 通道 6
0001	011	CPU 中断 优先级 1, 组优先级 3	CPU 中断 优先级 1, 组优先级 3	CPU 中断 优先级 1, 组优先级 3
0001	000	CPU 中断 优先级 1, 组优先级 0	CPU 中断 优先级 1, 组优先级 0	CPU 中断 优先级 1, 组优先级 0
0000	XXX	无服务	无服务	无服务

注：仅在 GPX = 0, COUNT ≠ 0 的情况下，才能实现 PEC 服务。

优先级为 7...1 的中断请求不能触发 PEC 传送，它们只能使用中断服务程序进行处理：和 PECC 寄存器无关，不检查位域 COUNT。

5.4.1 PEC 源指针与目的指针

PEC 通道的源指针和目的指针指定了数据传送的源和目的地址。两个指针都是 24 位指针，由 16 位偏移地址寄存器（SRCPx 或 DSTPx）和 8 位段地址（SRCSEGx 或 DSTSEGx，均存放在寄存器 PECSEGx 中）级联构成。

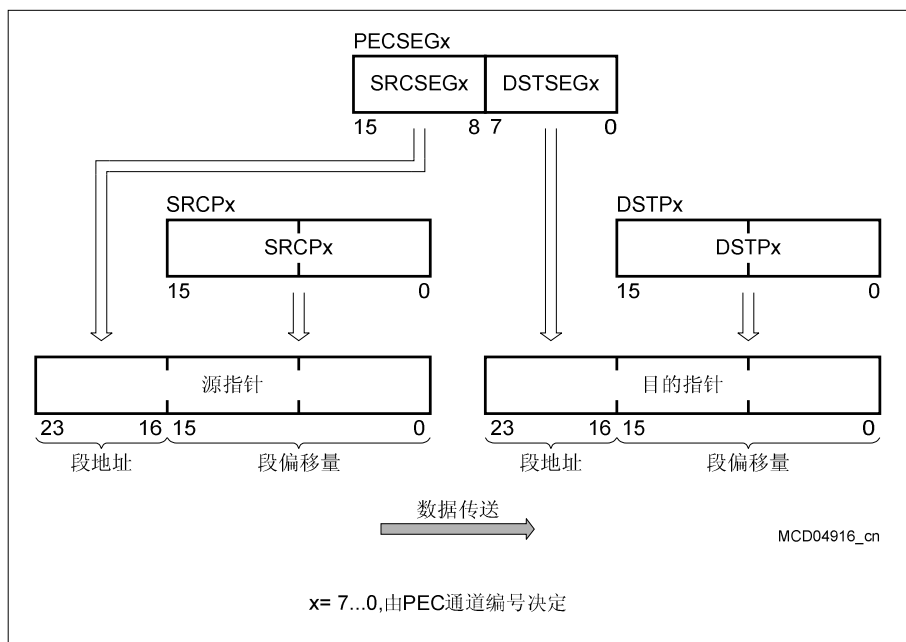


图 5-3 PEC 数据指针

PEC 传送后，PEC 指针由硬件自动增加：只增加偏移地址（SRCPx 和/或 DSTPx），相应的段地址不被硬件修改。因此，指针只能在当前段内增加，而不能超越段边界。当 PEC 指针增加到最大偏移量（字传送：FFFE_H，字节传送：FFFF_H），指针不再增加，保持最大偏移量。这样可以防止相邻段的存储单元被无意修改。

指针饱和的情况下，系统并不产生错误事件。因此，用户在程序设计时必须避免发生这种状况。

注：PEC 数据传送不使用数据页指针 DPP3...DPP0。

未使用的 PEC 指针可用来存储数据。

SRCPx

PEC 源指针

XSFR (ECyyH/--, 表 5-7)

复位值: 0000_H

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

符号	位序号	读写类型	功能描述
srcpx	[15:0]	rwh	PEC 通道 x 的源地址指针偏移量 源地址位 15...0

DSTPx

PEC 目的指针

XSFR (ECyyH/--, 表 5-7)

复位值: 0000_H

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

符号	位序号	读写类型	功能描述
dstpx	[15:0]	rwh	PEC 通道 x 的目标地址指针偏移量 目的地址位 15...0

PECSEGx

PEC 段指针

XSFR (ECyyH/--, 表 5-7)

复位值: 0000_H

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

符号	位序号	读写类型	功能描述
srcsegx	[15:8]	rw	通道 x 源地址的段指针 源地址位 23...16
dstsegx	[7:0]	rw	通道 x 目的地址的段指针 目的地址位 23...16

表 5-7 PEC 数据指针寄存器地址

通道 #	0	1	2	3	4	5	6	7
PECSEGx	EC80 _H	EC82 _H	EC84 _H	EC86 _H	EC88 _H	EC8A _H	EC8C _H	EC8E _H
SRCPx	EC40 _H	EC44 _H	EC48 _H	EC4C _H	EC50 _H	EC54 _H	EC58 _H	EC5C _H
DSTPx	EC42 _H	EC46 _H	EC4A _H	EC4E _H	EC52 _H	EC56 _H	EC5A _H	EC5E _H

注：如果指定 PEC 通道选择字数据传送 (BWT = 0)，那么相应的源指针和目的指针必须包含有效的字地址，即指向偶字节地址。否则，使用该 PEC 通道会引发非法字访问强制中断。

5.4.2 PEC 传送控制

PEC 传送计数位域 COUNT 控制相应 PEC 通道的操作。COUNT 位域的值选择服务请求被激活后 PEC 所要进行的操作。COUNT 可以支持：指定次数的 PEC 传送、无限次的传送或者不执行 PEC 服务。**表 5-8** 给出在 COUNT 先前值不同的情况下，所对应的位域 COUNT 当前值、中断请求标志 IR 及 PEC 通道操作。

表 5-8 位域 COUNT 的影响

COUNT 先前值	COUNT 修改值	服务后的 IR 值	PEC 通道操作和注解
FF _H	FF _H	0	转移一个字/字节 连续传送模式，即不修改 COUNT 值
FE _H ... 02 _H	FD _H ... 01 _H	0	转移一个字/字节，COUNT 减 1
01 _H	00 _H	1	EOPINT = 0 （特定通道中断） 转移一个字节/字 请求标志保持置位状态，触发另一个请求
		0	EOPINT = 1 （独立的 PEC 结束中断） 转移一个字节/字 请求标志清零，置位相应 PEC 子节点请求标志 CxIR ¹⁾
00 _H	00 _H	—	无 PEC 操作！ 激活中断服务程序，而不是激活 PEC 通道服务

1) 如果子节点请求被使能（CxIE = 1），那么置位子节点请求标志 CxIR 也会置位标志 EOPIR。

PEC 传送计数器可通过 PEC 通道执行指定次数的 PEC 请求服务，然后（当 COUNT 值递减到 00_H 时）激活中断服务程序。该中断服务程序或者是特定通道中断（和该 PEC 通道优先级相同），或者是普通的 PEC 结束中断。每次 PEC 传送后，COUNT 值递减（COUNT = FF_H 的情况除外），并对请求标志清零以指示已完成所请求的服务。

当 COUNT 的值为 00_H 时，PEC 通道保持空闲状态，相应的中断服务程序被激活。从而可通过标准中断服务程序来处理所有优先级的服务请求。

COUNT 位域为 FF_H 时，选择**连续传送模式**。这种情况下，COUNT 的值不被修改，相应 PEC 通道服务任何请求，直至 PEC 通道被再次禁止。

执行一次 PEC 传送后、COUNT 由 01_H 减至 00_H 时，请求一个标准中断，用于 PEC 数据块传送结束后的处理（特定通道中断或者普通 PEC 结束中断，见**表 5-8**）。

5.4.3 支持数据链的通道链接模式

在通道链接模式下，每两路 PEC 通道构成一对（通道 0+1、2+3、4+5、6+7），成对的两路通道被依次激活。对偶数通道的请求触发当前有效的 PEC 通道（或数据块传送结束中断），而对奇数通道的请求只触发与之相关的中断节点。当一路通道的传送计数计数到 0 时，控制切换至另一路通道，然后返回。这种模式支持数据链传送，可以将相互独立的数据块传送到同一目的地址（反之亦然）。例如，由多个数据块（前导码、数据等）建立通信帧。

如果一对通道中至少一路通道的链接控制位（PECCx 寄存器中的位 CL）置位，则这一对通道的通道链接模式被使能。链接的通道对由其中偶数通道的优先级设置（中断优先级和组优先级）控制。通道链接模式使能后，偶数通道有效。

如果有效通道的传送计数器由 1 减为 0（之前 COUNT > 0）时，该通道链接控制位 CL 为 1，且另一通道的传送计数器不为 0，那么就**执行通道链接**。在这种情况下，有效通道发送 EOP 中断请求，自动选择激活另一路通道。

注：通道链接总是从偶数通道开始。

如果有效通道的传送计数器由 1 减为 0（之前 COUNT > 0）时，该通道链接控制位 CL 为 0，或者另一路通道的传送计数器为 0，那么就**终止通道链接**。在这种情况下，会触发一个由 EOPINT 选定的中断（特定通道中断或普通 EOP 中断）。

通过置位 CL 和设置传送计数器值（>0），即可使用 PEC 通道链接模式实现数据链传送。由通道链接中断触发该操作重复执行，直至整个序列传送完成。最后一次传送时，中断服务程序应清零相应的位 CL，从而在整个传送结束时，通过最后通道的位 EOPINT 选择产生标准中断或者 PEC 结束中断。

注：为了使能通道链接，两个通道的传送计数器最初都必须设置为非零值。在数据序列传送过程中，只有在传送计数器计数到 0 时，才需要重新设置该通道。

5.4.4 PEC 中断控制

当执行指定次数的 PEC 传送后，相应的 PEC 通道被禁止，取而代之的激活标准的中断服务程序。每个 PEC 通道或者激活相关的特定通道中断节点；或者激活寄存器 PECISNC 中相关的 PEC 子节点请求标志，该请求标志随后将激活寄存器 EOPIC 中共用的节点请求标志（见图 5-4）。

PECISNC

PEC 中断子节点控制寄存器 **SFR (FFA8_H/D4_H)** **复位值: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
C7IR	C7IE	C6IR	C6IE	C5IR	C5IE	C4IR	C4IE	C3IR	C3IE	C2IR	C2IE	C1IR	C1IE	C0IR	C0IE
rwh	rw	rwh	rw	rwh	rw	rwh	rw	rwh	rw	rwh	rw	rwh	rw	rwh	rw

符号	位序号	读写类型	功能描述
CxIR x = 7...0	[2x+1]	rwh	PEC 通道 x 中断请求标志 0 PEC 通道 x 没有挂起的中断请求 1 PEC 通道 x 已经发出一个 PEC 结束中断请求 <i>注：该标志必须由软件清零</i>
CxIE x = 7...0	[2x]	rw	PEC 通道 x 中断使能控制位 （单独使能/禁止特定的中断请求源） 0 禁止通道 x PEC 结束中断请求 1 使能通道 x PEC 结束中断请求 ¹⁾

1) 推荐在置位使能标志（CxIE）前，先清零相应的中断请求标志（CxIR）。否则，以前的请求仍然挂起，不能触发新的中断请求。

EOPIC

PEC 结束中断控制寄存器 **ESFR (F180_H/C0_H)** **复位值: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	GPX	EOP IR	EOP IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw	rw				rw	

注：各控制位的描述请参见通用中断控制寄存器的描述。

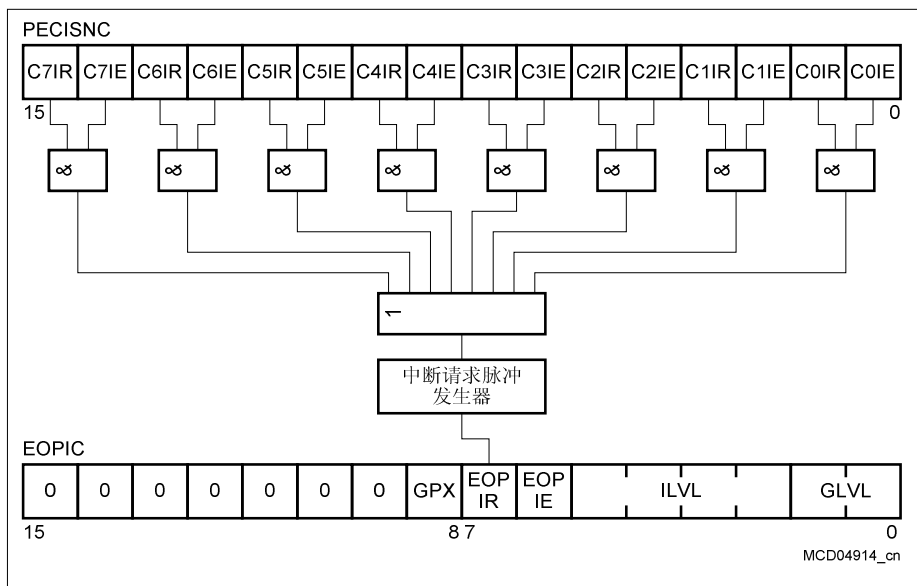


图 5-4 PEC 结束中断子节点

注：中断服务程序必须服务并清零所有有效的请求后才能终止。随后出现的请求将再次置位 **EOPIR**，再次进入中断服务程序。

5.5 中断和 PEC 服务请求的优先级排序

中断请求和 PEC 服务请求被使能时，它们可参与仲裁，仲裁胜出的请求被响应；中断请求和 PEC 服务请求被禁止时，这些请求将被忽略，不进行处理。

可以通过三种机制**使能和禁止中断请求**：

- 控制位
- 中断优先级
- ATOMIC 指令和 EXTended 指令

控制位可分别控制每个中断请求源的“开”、“关”，从而控制是否产生中断请求。控制位 (xxIE) 位于相应的中断控制寄存器中。由 PSW 寄存器中的位 IEN 对所有中断请求进行全局使能或禁止。该控制位是一个“总开关”，它决定是否接受来自任何中断源的请求。

对于要参与仲裁的某个特定的中断请求，其相应中断源的使能位和全局使能位二者都必须置位。

根据**中断优先级**，自动选择一组中断请求予以应答，忽略所有其它请求。仲裁胜出的中断源的优先级和 CPU 当前优先级相比较，只有当它高于 CPU 优先级时，该中断请求才被服务。软件修改 CPU 的优先级后，所有同级或者更低优先级的中断请求将不被响应。优先级为 0 的中断源将被禁止，永远不会被服务。

ATOMIC 和 EXTend 指令在其后 1...4 条指令执行期间，自动禁止所有中断请求。这对旗语操作等场合非常有用，在执行该不可分的指令序列之后，不需要重新使能中断系统。

中断类别管理

一类中断由一组重要程度相同的中断源组成。也就是说，从系统角度讲，它们具有相同的优先级。同类的中断不能相互中断。XC164CM 具有两个相关特性支持该功能：

使用相同的中断优先级 (ILVL)，并为每个成员分配一个专用的组优先级，可以建立最多 8 个成员的中断类。由中断控制器自动生成和处理该功能。

使用相邻的中断优先级 (ILVL) 和相应的组优先级（每个 ILVL 包含 8 个组优先级），可以建立多于 8 个成员的中断类。该中断类中的每个中断服务程序都会将 CPU 的优先级设为该类中的最高优先级。所有同级或更低优先级的请求将被拒绝，即该类中的其它请求不会被接受。

下面的例子建立了 3 类中断，每类中断包含 2 或 3 级中断优先级。在中断类 2 中，优先级为 6 的中断通过将当前 CPU 优先级修改成 8（该类中的最高优先级），可禁止中断类 2 中的所有其它中断源。此时，中断类 1 的中断请求或 PEC 请求仍能得到服务。

通过这种方式，所有的中断源（PEC 请求除外）被分配给 3 类中断优先级，而不是像硬件支持的那样分配给 7 级不同的优先级。

表 5-9 软件控制的中断类（示例）

ILVL (优先级)	组优先级								解释
	7	6	5	4	3	2	1	0	
15									最多 8 路 PEC 服务通道
14									
13									
12	X	X	X	X	X	X	X	X	中断类 1 9 个中断源分布在 2 级中断优先级上
11	X								
10									
9									
8	X	X	X	X	X	X	X	X	中断类 2 17 个中断源分布在 3 级中断优先级上
7	X	X	X	X	X	X	X	X	
6	X								
5	X	X	X	X	X	X	X	X	中断类 3 9 个中断源分布在 2 级中断优先级上
4	X								
3									
2									
1									
0									无服务

5.6 上下文切换与状态保存

仲裁后的中断请求在被真正处理之前，当前工作任务的状态被自动保存到系统堆栈中，这包括要保存 CPU 状态（PSW）和中断返回地址（执行完中断服务程序，返回被中断的任务所在的地址继续运行）。返回地址由指令指针（IP）指定，存储器分段模式下，还与代码段指针（CSP）有关。寄存器 CPUCON1 中的位 SGTDIS 控制如何保存返回地址。

不分段模式下，系统堆栈先保存 PSW、然后是 IP；分段模式下，系统堆栈先保存 PSW、然后是 CSP、接下来是 IP。如果禁止存储器分段，这样可使系统堆栈得到优化利用。

CPU 优先级控制位域（PSW 中的 ILVL）被更新为被响应的中断请求的优先级，于是，CPU 工作在新的优先级上。

寄存器组选择位域（PSW 中的 BANK）相应修改为与中断请求相关的寄存器组。中断请求与寄存器组之间的关联，部分已经被预先定义，部分可以由用户编程。

中断请求被响应后，相应的中断请求标志被清零；将对应的中断向量载入 IP 和 CSP，从中断向量地址中取出中断服务程序的第一条指令（分支指令），进而跳转到真正的服务程序中（使用中断跳转表缓存的情况例外）。所有其它 CPU 资源，如数据页指针、上下文指针等不受影响。

当退出中断服务程序时（执行 RETI 指令），状态信息从系统堆栈中以“后进先出”的次序弹出。弹出时，需要考虑 SGTDIS 的值。

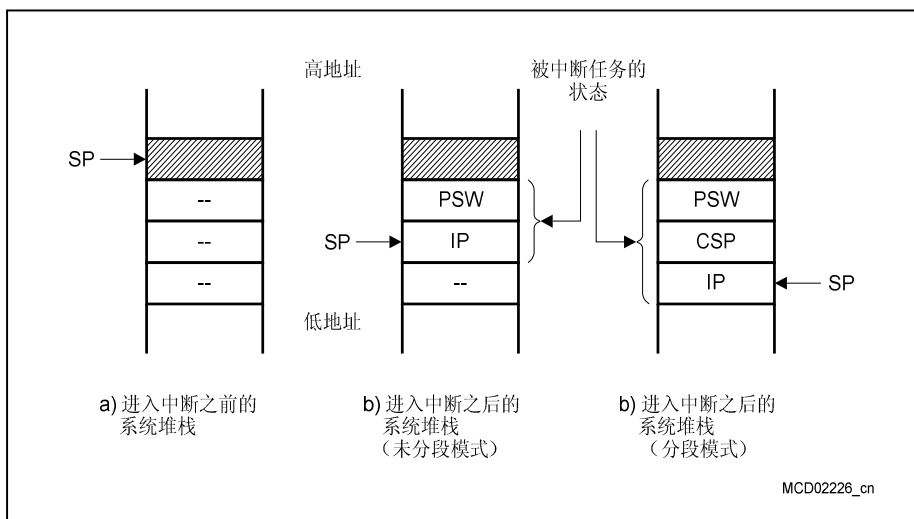


图 5-5 任务状态保存在系统堆栈中

上下文切换

中断服务程序通常要在堆栈中保存中断服务程序要用到的所有寄存器，并在中断返回前将这些寄存器恢复。中断服务程序使用的寄存器越多，保存和恢复所消耗的时间就越多。XC164CM 可以自动或者采用一条指令来切换整个 CPU 通用寄存器组（GPR）。这样，中断服务程序可以用其独立的上下文工作（见[章节 4.5.2](#)）。

XC164CM 内核有两种方式切换上下文：

全局寄存器组的上下文切换：通过一条指令改变上下文指针，进而改变整个 CPU 通用寄存器（GPR）的全局寄存器组的基地址，使中断服务程序可以在自己独立的上下文中执行。指令“SCXT CP, #New_Bank”将上下文指针（CP）的内容压入系统堆栈，将立即数“New_Bank”载入 CP，这样即选定了新的寄存器组。此时中断服务程序可以使用“自己的寄存器”。当服务程序结束时，这个寄存器组被保留，即下次调用时，其内容仍然可用。中断返回（RET1）之前，只需将原先的 CP 从系统堆栈中弹出，恢复使用原先的全局寄存器组。

中断程序所使用的资源（比如数据页指针 DPP）必须被保存和恢复。

注：考虑到流水线的操作，上下文切换时，会有一定的时序限制。

通过改变所选用的寄存器组来切换上下文：自动更新位域 BANK，从两个局部寄存器组或当前全局寄存器组中选择一个，此时服务程序可以直接使用“自己的寄存器”。当服务程序结束时，这个寄存器组被保留。在下次调用时其内容仍然可用。

当切换到全局寄存器组时，由于全局寄存器很可能被多个任务使用，所以中断服务程序必须切换全局寄存器组的上下文指针，以获得一组专用的 GPR。

对于优先级为 15...12 的中断，其目标寄存器组可以预先选定，进而自动切换。寄存器组选择寄存器 BNKSELx 为每一个可能的仲裁优先级提供一个 2 位位域。一旦某中断请求被接受，相应位域值就被复制到 PSW 寄存器的位域 BANK 中，用来选择寄存器组。

表 5-10 给出四个寄存器组选择寄存器中控制位域对应的仲裁优先级分配。

BNKSELx

寄存器组选择寄存器 x

XSFR（表 5-10）

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPRSEL7	GPRSEL6	GPRSEL5	GPRSEL4	GPRSEL3	GPRSEL2	GPRSEL1	GPRSEL0								
rw	rw	rw	rw	rw	rw	rw	rw								

符号	位序号	读写类型	功能描述
GPRSELy (y = 7...0)	[2y+1:2y]	rw	寄存器组选择 00 全局寄存器组 01 保留 10 局部寄存器组 1 11 局部寄存器组 2

表 5-10 寄存器组控制位域对应的优先级分配

寄存器组选择控制寄存器		中断节点优先级		备注
寄存器名	位域	中断优先级	组优先级	
BNKSEL0 (EC20 _H /--)	GPRSEL0...3	12	0...3	低组优先级
	GPRSEL4...7	13	0...3	
BNKSEL1 (EC22 _H /--)	GPRSEL0...3	14	0...3	
	GPRSEL4...7	15	0...3	
BNKSEL2 (EC24 _H /--)	GPRSEL0...3	12	4...7	高组优先级
	GPRSEL4...7	13	4...7	
BNKSEL3 (EC26 _H /--)	GPRSEL0...3	14	4...7	
	GPRSEL4...7	15	4...7	

5.7 中断节点共享

如果多个中断请求互斥或者产生速率很低，那么这些中断请求可以共享中断节点。在这种情况下，如果有多个中断请求被使能，中断处理器将首先确定响应哪个请求源。不过，如果产生中断请求的速率较低，这点时间开销无关紧要。

节点共享有两种方式：1) 通过中断子节点控制寄存器 **ISNC** 控制，该寄存器为每个请求源提供单独的请求标志和使能位；2) 用所有请求源逻辑或（OR）的结果触发公共节点。用于仲裁的中断优先级由节点控制寄存器（...IC）决定。

与节点请求位被自动清零相反，**ISNC** 寄存器中的特定请求标志必须由软件复位。

表 5-11 子节点控制位分配

中断节点	中断请求源	控制
EOPIC	PEC 通道 7...0	PECISNC
RTC_IC	RTC: T14 溢出、CNT0...CNT3 溢出	RTC_ISNC
ASC0_ABIC	ASC0: 自动波特率检测开始，错误请求	逻辑或
ASC1_ABIC	ASC1: 自动波特率检测开始，错误请求	逻辑或

5.8 外部中断

尽管 XC164CM 没有专用的 INTR 输入引脚，但它可利用多条 IO 线作为中断输入，响应外部异步事件。中断功能可以与引脚的主要功能复用，在不需要引脚主要功能的情况下，该引脚可专门用作中断输入。

即使在休眠模式下，**快速外部中断**检测也能提供灵活的唤醒信号。该功能也可以根据外部输入信号产生附加的中断请求。

表 5-12 可用作快速外部中断的引脚

端口引脚	原主要功能	控制寄存器
P1H.5-4/CC25-24IO	CAPCOM 寄存器 25-24 捕获输入	CC25—CC24
P1H.0/CC23IO	CAPCOM 寄存器 23 捕获输入	CC23
P1L.7/CC22IO	CAPCOM 寄存器 22 捕获输入	CC22
P9.5-0/CC21-16IO	CAPCOM 寄存器 21-16 捕获输入	CC21—CC16
P3.2/CAPIN	GPT2 捕获输入引脚	T5CON
P3.7/T2IN	辅助定时器 T2 输入引脚	T2CON
P3.5/T4IN	辅助定时器 T4 输入引脚	T4CON

对于上表中的每个引脚，可以选择由该引脚上的正跳变、负跳变或任意跳变来触发中断或 PEC 服务请求。跳变沿选择在端口引脚相应的外设控制寄存器中定义（对快速外部中断单独控制）。外设必须设定工作在特定的模式，以允许由外部信号产生中断。中断请求的优先级由各外设中断源的中断控制寄存器确定，通过外设中断源的中断向量来服务相应的外部中断请求。

注：为了把上述引脚用作外部中断输入，必须通过相应的端口方向控制寄存器 DPx 中的方向控制位 DPx.y 将上述引脚切换到输入模式。

当端口引脚 CCxIO 用作外部中断输入引脚时，对应的捕获/比较寄存器 CCx 中的位域 CCMODx 必须选择捕获模式。若 CCMODx 设置为 001_B，检测到引脚 CCxIO 上的正跳变时，寄存器 CCxIC 中的中断请求标志 CCxIR 置位；若 CCMODx 设置为 010_B，检测到引脚 CCxIO 上的负跳变时，中断请求标志 CCxIR 置位；若 CCMODx 设置为 011_B，检测到引脚 CCxIO 上的任意跳变时，中断请求标志 CCxIR 置位。在以上三种情况下，无论 CAPCOM 的定时器是否运行，定时器的内容都将被锁存到捕获寄存器 CCx 中。若中断使能位 CCxIE 置位，将产生 PEC 请求或 CCxINT 中断请求。

当 GPT1 模块中相关的辅助定时器 T2 或 T4 配置成捕获模式时，引脚 T2IN 和 T4IN 也可以用作外部中断输入引脚。将控制寄存器 T2CON 或 T4CON 中的模式控制位

域 T2M 或 T4M 设置为 101_B，即可选择捕获模式。外部输入信号的触发有效沿由位域 T2I 或 T4I 决定。若位域 T2I 或者 T4I 设置为 X01_B，T2IN 或 T4IN 引脚上出现正跳变时，寄存器 T2IC 或 T4IC 中的中断请求标志 T2IR 或 T4IR 置位；若位域 T2I 或者 T4I 设置为 X10_B，外部引脚负跳变时，相应的中断请求标志置位；若位域 T2I 或者 T4I 设置为 X11_B，外部引脚任意跳变时，将置位请求标志。在以上三种情况下，引脚 T2IN 或 T4IN 跳变时，内核定时器 T3 的内容被捕获到辅助定时器寄存器 T2 或 T4 中。若中断使能位 T2IE 或 T4IE 置位，将产生 PEC 请求或者 T2INT、T4INT 中断请求。

CAPIN 引脚与定时器输入引脚略有不同，因为 CAPIN 引脚可以用作外部中断输入引脚，而不影响外设功能。当寄存器 T5CON 中的捕获模式使能位 T5SC 被清零时，CAPIN 引脚上的信号跳变只对 CRIC 寄存器中的中断请求标志 CRIR 置位，并不激活寄存器 CAPREL 的捕获功能。

因此，当引脚 CAPIN 用作外部中断输入时，寄存器 CAPREL 仍可用作 GPT2 定时器 T5 的重载寄存器。T5CON 寄存器中的位域 CI 用于选择外部中断输入信号的有效跳变。CI = 01_B 时，中断请求标志在外部正跳变时置位；CI = 10_B 时，中断请求标志在外部负跳变时置位；CI = 11_B，中断请求标志在外部任意跳变时均置位。若中断使能位 CRIE 置位，将产生 CRINT 中断请求或 PEC 请求。

注：非屏蔽中断输入引脚 \overline{NMI} 和复位输入引脚 \overline{RSTIN} 为 CPU 响应外部输入信号提供了另外一种方式。 \overline{NMI} 和 \overline{RSTIN} 是能够引起硬件强制中断的专用输入引脚。

快速外部中断

每个系统时钟周期都会对快速外部中断引脚采样。也就是说，在每个 $1/f_{SYS}$ 时间周期内扫描和检测外部事件。不过，对这些中断请求的仲裁和处理按正常时序进行。

外部中断控制器 EXICON 分别为 8 个快速外部中断选择触发沿（上升沿、下降沿、或上升/下降沿）。

这些快速外部中断使用 CAPCOM 通道 CC13...CC8 的中断节点和中断向量，因此，捕获/比较功能不能使用。

EXICON

外部中断控制寄存器

ESFR (F1C0H/E0H)

复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	EXI5ES	EXI4ES	EXI3ES	EXI2ES	EXI1ES	EXI0ES	-	-	-	-	-	-
-	-	-	-	rw	rw	rw	rw	rw	rw	-	-	-	-	-	-

符号	位序号	读写类型	功能描述
EXIxES x = 5...0	[11:10] ... [1:0]	rw	外部中断 x 跳变沿选择位域 00 禁止快速外部中断：标准模式 01 正跳变时产生中断（上升沿） 10 负跳变时产生中断（下降沿） 11 任意跳变时产生中断（上升沿或下降沿）

外部中断源控制

每个快速外部中断（由 EXICON 寄存器控制）的输入源可以来自多达三个相关端口引脚（标准引脚 EXnIN 或两个复用引脚）。激活复用输入引脚可检测禁用接口的接口线上的跳变。一旦得到触发信号，相应接口可被重新激活，并对检测到的操作作出响应。

通过寄存器 EXISEL0 和 EXISEL1 控制中断输入源选择。除了可以选择三个输入引脚中的一个，还可以选择将两个或三个输入引脚逻辑组合。这样可增加唤醒线的数目，或者定义一个特定信号组合来触发唤醒中断。

EXISEL0

外部中断源寄存器 0 **ESFR (F1DA_H/ED_H)** 复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
EXI3SS				EXI2SS				EXI1SS				EXI0SS			
rw				rw				rw				rw			

EXISEL1

外部中断源寄存器 1 **ESFR (F1D8_H/EC_H)** 复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-				-				EXI5SS				EXI4SS			
-				-				rw				rw			

符号	位序号	读写类型	功能描述
EXIxES x = 5...0	[15:12]	rw	外部中断 x 输入源选择
	...		0000 从相关 EXxIN 引脚输入
	[3:0]		0001 从复用引脚 AltA 输入
			0010 从复用引脚 AltB 输入
			0011 从引脚 EXxIN 与复用引脚 AltA 逻辑或的结果输入
			0100 从引脚 EXxIN 与复用引脚 AltA 逻辑与的结果输入
			0101 从复用引脚 AltA 与复用引脚 AltB 逻辑或的结果输入
			0110 从复用引脚 AltA 与复用引脚 AltB 逻辑与的结果输入
			0111 从引脚 EXxIN、AltA、AltB 三者逻辑或的结果输入
			1XXX 保留，请不要使用该组合

表 5-13 总结了寄存器 EXISEL 中的相关位域（即中断线）与输入引脚的关系。

表 5-13 外部中断节点与中断输入引脚的连接

控制位域	标准引脚 EXnIN	复用引脚 AltA	复用引脚 AltB	中断控制 寄存器	相关接口	备注
EXI0SS	P1H.0	P1H.3	P1H.0	CC8IC	SSC1	—
EXI1SS	P1H.1	P3.1	—	CC9IC	ASC1	—
EXI2SS	P1H.2	P3.11	P3.10	CC10IC	ASC0	—
EXI3SS	P1H.3	P3.13	—	CC11IC	SSC0	—
EXI4SS	P1H.4	P9.2	—	CC12IC	CAN_A	实际接口 引脚可编程 设置
EXI5SS	P1H.5	P9.0	—	CC13IC	CAN_B	

休眠模式期间的外部中断

在休眠模式下，所有外设时钟信号均无效，这也将关闭快速外部中断的标准边沿检测。但为了启动唤醒操作，这些中断输入上的跳变必须被识别。因此，对快速外部中断采用一种不需要时钟信号的特殊边沿检测逻辑（因此在休眠模式也可工作），并带有一个模拟噪声滤波器，可以抑制周期小于 **10ns** 的毛刺（由噪声产生）。至少持续 **100ns** 的输入脉冲才会被系统识别并产生中断请求。

虽然该噪声滤波器将外部唤醒信号的识别延迟了大约 **100ns**，但毛刺抑制保证了在活跃环境下系统休眠/唤醒机制的安全和稳健操作。

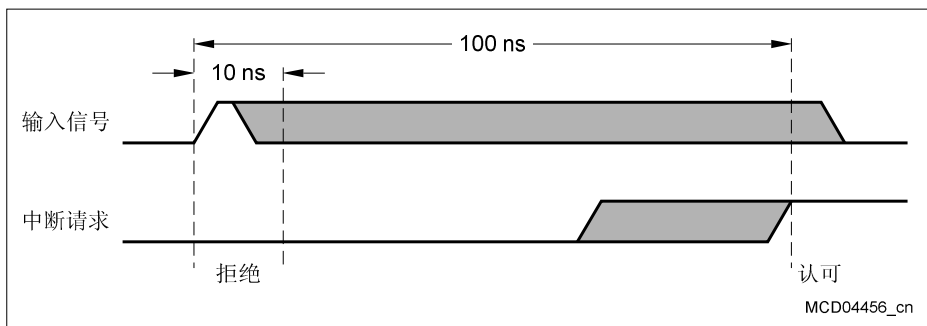


图 5-6 输入噪声滤波操作

外部中断脉冲时序

通过同步逻辑和异步逻辑来处理外部中断输入。系统工作在较高频率时，由同步逻辑支持短中断脉冲的识别；在休眠模式（此时系统时钟不可用）下，由异步逻辑确保中断脉冲的识别。

在下面两种情况下，系统能够可靠地识别出外部中断信号：

- 信号至少保持有效 **100ns**（带毛刺滤波器的异步逻辑），或
- 信号至少保持 **2 个 f_{SYS} 周期**（同步逻辑）

无论先满足以上哪个条件，中断信号都会被识别到。

*注：从休眠模式唤醒后到 PLL 稳定的这段时间间隔内，即便出现新的外部中断脉冲也不要紧。这是因为，如果外部中断信号能够至少持续 **100ns**，那么异步逻辑将能准确地检测到该信号。*

注： \overline{NMI} 输入引脚具有相同的毛刺滤波器功能，且对时序的要求也相同。

5.9 OCDS 请求

OCDS 模块发出高优先级的断点请求或标准服务请求。断点请求直接发送至 CPU（与硬件强制中断请求相似），并进行优先级排序。因此，断点请求不必进行标准中断仲裁过程，具有最高优先级。

标准 OCDS 服务请求与经过仲裁的中断/PEC 请求一并送至 CPU 操作控制单元。优先级较高的请求发送给 CPU，获得 CPU 服务。如果标准 OCDS 请求和中断/PEC 请求的优先级相同，中断/PEC 请求将胜出。

这种方法保证了准确的断点控制，同时对系统操作的影响尽可能小。

CPU 操作控制单元也会将请求应答和拒绝信息从内核送回给相应的请求源。

5.10 中断服务请求的响应延迟

XC164CM 服务请求（中断或 PEC 服务请求）的产生和指令流的执行不同步。因此，这些请求经过仲裁后，被插入到当前的指令流中。这样可以使中断服务请求的处理与当前执行的指令流无关，但也导致了一定的延迟时间。

请求响应延迟时间是指从中断控制器（ITC）的请求信号被激活直到相关指令到达流水线执行阶段的时间。

表 5-14 列出了这一过程的连续步骤。

表 5-14 服务请求响应步骤

步骤描述	中断响应	PEC 响应
通过 3 级请求仲裁后，请求被 CPU 接受（见章节 5.2）	9 个周期	9 个周期
在流水线指令流中插入一条内部指令	4 个周期	4 个周期
从中断向量表中取出的第一条指令进入流水线执行阶段	4 个周期 / 0 ¹⁾	— — —
最小的请求响应延迟	17/13 个周期	13 个周期

1) 可以使用中断跳转表缓存来节省这段时间（见章节 5.3）。

额外的响应延时

因为服务请求要插入到当前指令流中，所以指令流的属性会对中断请求的响应时间有一定影响。

表 5-15 系统逻辑引起的额外延时

延迟原因	中断响应	PEC 响应
中断控制器忙，其正在执行一个仲裁周期	最大 9 个周期	最大 9 个周期
流水线停滞， 因为在插入的指令（PEC 或 ITRAP）之前，已经有两条指令进入流水线。需要将这两条指令执行完毕后才能执行插入的指令。例如，这些指令可能需要向外设或存储器写数据，或者从外设或存储器读数据，或者需要额外的时钟周期才能完成。	$2 \times T_{ACCmax}$	$2 \times T_{ACCmax}$
流水线被取消，因为流水线中该插入指令之前的指令更新了内核 SFRs	4 个周期	4 个周期
写堆栈时的存储器访问（如果堆栈不是采用 DPRAM 和 DSRAM）	$\frac{2}{3} \times T_{ACC}^{1)}$	— — —
读中断向量表时的存储器访问 （中断跳转表缓存除外）	$2 \times T_{ACC}$	— — —

1) 取决于分段模式是否开启。

真正的中断请求响应时间可能还会被延迟，这取决于实际应用中所采用的编程技术。可以采取以下方式减小响应时间：

- 只通过 **JUMP** 指令从中断向量表跳转到真正的中断服务程序。
 可以将对时间要求严格的指令直接放在中断向量表中，之后紧跟一个分支指令，跳转到中断服务程序的剩余部分。由 CPUCON1 寄存器的位域 VECSC 选择两个相邻向量的间距。
- 在执行想要运行的指令前，进行上下文切换（见**章节 5.6**）。
 可以将对时间要求严格的指令编程为“非破坏性”指令，在进行上下文切换之前（用于中断服务程序的剩余部分），执行这些指令。

5.11 强制中断功能

强制中断中断当前执行程序的方式和标准中断相似。不过，强制中断不需经过中断系统的优先级排序，用于需要系统能够立即响应的场合。强制中断功能不可屏蔽，其优先级始终高于其它任何中断请求。

XC164CM 提供了两种不同的强制中断机制：**硬件强制中断**由程序执行期间的事件（比如非法访问或未定义的操作码）触发；**软件强制中断**通过当前指令流中的一条指令触发。

软件强制中断

TRAP 指令引发软件调用中断服务程序。中断向量编号由强制中断指令的操作数域给定，确定即将跳转到向量表中的向量地址。

执行 **TRAP** 指令，与在同一向量地址处发生中断的效果类似。**PSW**、**CSP**（分段模式）和 **IP** 压入内部系统堆栈，并跳转到特定的向量地址。执行强制中断时，将寄存器 **VECSEG** 的值载入强制中断服务程序的 **CSP**。**TRAP** 指令不影响任何中断请求标志。**TRAP** 指令调用的中断服务程序必须由 **RETI** 指令结束（从中断返回），以确保正确的操作。

注：TRAP 指令不会修改 PSW 寄存器中的 CPU 优先级和所选的寄存器组，所以服务程序和调用该服务程序的任务的优先级相同。因此，和硬件事件触发不同，由 TRAP 指令触发的强制中断服务程序使用原有的寄存器组，它可以被其它强制中断或更高优先级的中断请求中断。

硬件强制中断

硬件强制中断由程序执行期间的错误或特定系统状态触发（汇编阶段不能识别指令）。还可能有意地触发硬件强制中断，例如：通过产生一个非法操作码的强制中断来模拟附加指令。XC164CM 能区分 8 种不同的硬件强制中断功能。当检测到硬件强制中断条件时，CPU 跳转到相应的强制中断向量地址。在进入强制中断处理程序之前，要先执行完引发强制中断的那条指令。

硬件强制中断是不可屏蔽的，它始终具有比其它任何 CPU 操作更高的优先级。如果在同一个指令周期内检测到多个强制中断条件发生，具有最高优先级的强制中断得到处理（见 [表 5-3](#)）。

PSW、**CSP**（分段模式下）和 **IP** 压入内部系统堆栈，**PSW** 寄存器中的 **CPU** 优先级设置为最高（即 15 级），禁止所有中断。选择全局寄存器组，指令执行跳转到向量表中的相应强制中断向量地址。必须由 **RETI** 指令来结束强制中断服务程序。

XC164CM 的 8 种硬件强制中断功能分为两类：

A 类强制中断：

- 外部非屏蔽中断（NMI）
- 堆栈上溢强制中断

- 堆栈下溢强制中断
- 软件断点

这些强制中断的优先级相同，但分别对应不同的向量地址。

B 类强制中断：

- 未定义操作码
- 程序存储器访问出错
- 保护错误
- 非法字操作数访问

这些强制中断的优先级相同，并具有相同的向量地址。

强制中断标志寄存器（TFR）可位寻址，从而可使服务程序识别发生了哪种强制中断。每种强制中断功能由单独的请求标志指示。当发生一种硬件强制中断时，寄存器 TFR 中相应的请求标志置 1。

复位功能（硬件、软件、看门狗）可视为同一种类型的强制中断。复位功能的系统优先级最高（强制中断优先级 III）。

A 类强制中断的系统优先级仅低于最高优先级（强制中断优先级 II），B 类强制中断的优先级仅低于 A 类优先级。因此，A 类强制中断能够中断 B 类强制中断。多个 A 类强制中断同时出现时，会在内部实现优先级排序，NMI 强制中断优先级最高，软件断点优先级最低。

若未定义操作码强制中断（B 类强制中断）和 NMI 强制中断（A 类强制中断）同时出现，NMI 和 UNDOPC 标志都被置位，未定义操作码指令的 IP 被压入系统堆栈，执行 NMI 强制中断服务程序。从 NMI 服务程序返回之后，IP 弹出堆栈。由于 UNDOPC 强制中断挂起，IP 会被立即再次压入堆栈。

注：强制中断服务程序必须清除相应的强制中断标志。否则，退出服务程序之后，将会请求新的强制中断。软件置位强制中断请求标志与硬件置位具有相同的效果。

TFR

强制中断标志寄存器

SFR (FFAC_H/D6_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
NMI	STK OF	STK UF	SOFTBRK	-	-	-	-	UNDOPC	-	-	PACER	PRTFLT	ILL OPA	-	-
rwh	rwh	rwh	rwh	-	-	-	-	rwh	-	-	rwh	rwh	rwh	-	-

符号	位序号	读写类型	功能描述
NMI	15	rwh	非屏蔽中断标志 0 未检测到非屏蔽中断 1 在 $\overline{\text{NMI}}$ 引脚上检测到一个负跳变（下降沿）
STKOF	14	rwh	堆栈上溢标志 0 未检测到堆栈上溢事件 1 当前堆栈指针小于寄存器 STKOV 的值
STKOUF	13	rwh	堆栈下溢标志 0 未检测到堆栈下溢事件 1 当前堆栈指针大于寄存器 STKUN 的值
SOFTBRK	12	rwh	软件断点 0 未检测到软件断点事件 1 检测到软件断点事件
UNDOPC	7	rwh	未定义操作码 0 未检测到未定义操作码事件 1 当前被解码的指令不包含有效的 XC164CM 操作码
PACER	4	rwh	程序存储器访问错误 0 未检测到程序存储器访问错误事件 1 检测到非法或错误的程序存储器访问
PRTFLT	3	rwh	保护错误 0 未检测到保护错误事件 1 检测到非法格式的受保护指令

符号	位序号	读写类型	功能描述
ILLOPA	2	rwh	非法字操作数访问 0 未检测到非法字操作数访问事件 1 试图在奇地址进行字操作数访问（读或写）

A 类强制中断

由高优先级的系统 $\overline{\text{NMI}}$ 或特殊 CPU 事件，如软件断点、堆栈上溢、堆栈下溢事件产生 A 类强制中断。A 类强制中断并不用来指示硬件故障。A 类强制中断事件发生之后，将调用专用服务程序来响应该事件。每种 A 类强制中断在向量表中都有各自的向量地址。A 类强制中断不能中断正在执行的 **atomic/extend** 序列和 I/O 访问。因为在完成中断服务程序后，必须保证指令流的正确执行。而被中断的 **extend** 序列指令不能重新启动。除 $\overline{\text{NMI}}$ （异步外部事件）外，所有 A 类强制中断在指令执行流水线中产生。只能在指令执行的存储阶段产生 A 类强制中断事件。因此，在同一 CPU 周期内，A 类强制中断不可能由流水线上两条不同的指令产生。总是要将引起 A 类强制中断事件的指令执行完毕。如果正在执行 **atomic/extend** 序列或 I/O 读访问，则先将该序列执行完毕。一旦指令或序列执行完毕，流水线被取消，最后执行指令的下一条指令的 IP 压入堆栈。因此，发生 A 类强制中断时，堆栈始终保存指令流中未执行指令中的第一条指令的 IP。

注：分支折叠单元允许将跳转指令及前一条指令并行执行。经过预处理的跳转指令与前一条指令相结合，跳转指令与引起 A 类强制中断的指令同时执行。指令流中，未执行指令中的第一条指令的 IP 被压入堆栈。

多个 A 类强制中断同时出现时，会在内部实现优先级排序，NMI 强制中断优先级最高，软件断点优先级最低。

注：两个不同的 A 类强制中断同时发生时，两个强制中断标志都置位。最后执行指令的下一个指令的 IP 被压入堆栈。先执行具有较高优先级的强制中断服务程序。从该服务程序返回后，IP 从堆栈中弹出。由于另一个挂起的 A 类强制中断尚未处理，IP 被立即再次压入堆栈（除非第一个强制中断服务程序已经将 TFR 中与第二个强制中断相对应的强制中断标志清零）。

B 类强制中断

B 类强制中断由不可恢复的硬件故障产生。硬件出错时，CPU 必须立刻启动故障服务程序。B 类强制中断能够中断 **atomic/extend** 序列和 I/O 读访问。B 类强制中断服务程序执行完后，不可能恢复被中断的指令流。

所有 B 类强制中断具有相同的优先级（强制中断优先级 1）。多个 B 类强制中断同时发生时，寄存器 **TFR** 中的相应标志置位，并进入强制中断服务程序。由于所有 B 类强制中断具有相同的向量地址，同时发生的 B 类强制中断的优先级由强制中断服务程序中的软件决定。

访问出错是异步外部事件（对 CPU 而言），所有其它 B 类强制中断在指令执行流水线中产生。只能在指令执行的存储阶段产生 B 类强制中断事件。因此，在同一 CPU 周期内，B 类强制中断不可能由流水线上两条不同的指令产生。总是要将引起 B 类强制中断事件的指令执行完毕。之后，流水线取消，引起强制中断的指令的下一条指令的 IP 压入堆栈。因此，堆栈始终保存指令流中未执行指令中的第一条指令的 IP。

注：分支折叠单元允许将跳转指令及其前一条指令并行执行。经过预处理的跳转指令与前一条指令相结合，跳转指令与引起 B 类强制中断的指令同时执行。指令流中，未执行指令中的第一条指令的 IP 被压入堆栈。

在 B 类强制中断服务程序执行期间，若发生 A 类强制中断，则立即处理 A 类强制中断。而在 A 类强制中断服务程序执行期间，若发生 B 类强制中断，则要等到 A 类强制中断服务程序利用 **RETI** 指令退出后，才能处理 B 类强制中断。在这种情况下，**TFR** 寄存器中保存 B 类强制中断的状态信息，但引起该强制中断的指令 IP 值丢失。

*注：A 类强制中断和 B 类强制中断同时发生时，二者的强制中断标志都置位。引起强制中断的指令的下一条指令 IP 被压入堆栈，执行 A 类强制中断。若正在执行 **atomic/extend** 指令序列或正在读访问 I/O 时，发生两个强制中断，B 类强制中断会中断 **atomic/extend** 操作，无需等待指令序列执行完毕就开始执行 A 类强制中断的服务程序。从服务程序返回之后，IP 从系统堆栈中弹出。由于尚未处理 B 类强制中断，IP 会立刻被再次压入堆栈。此时，不可能恢复被中断的指令流。*

外部 NMI 强制中断

只要在专用外部引脚 $\overline{\text{NMI}}$ （非屏蔽中断）上检测到由高到低的跳变，即置位 TFR 寄存器中的 NMI 标志，CPU 将进入 NMI 强制中断服务程序。

堆栈上溢强制中断

只要堆栈指针与堆栈上溢寄存器 STKOV 相等且堆栈指针将自动减小，即置位 TFR 寄存器中的 STKOF 标志，CPU 将进入堆栈上溢强制中断服务程序。

堆栈上溢恢复时，必须保证堆栈中有足够的空间将当前系统状态保存两次（PSW、IP、分段模式下还包括 CSP），否则应当产生系统复位。

堆栈下溢强制中断

只要堆栈指针与堆栈下溢寄存器 STKUN 相等且堆栈指针将自动增加，即置位 TFR 寄存器中的 STKUF 标志，CPU 将进入堆栈下溢强制中断服务程序。

软件断点强制中断

当 CPU 正在执行 SBRK 指令时，即置位寄存器 TFR 中的 SOFTBRK 标志，CPU 随即进入软件断点调试程序。片上仿真模块可以禁止产生软件断点指令的强制中断标志。这种情况下，该指令只中断指令流，并将该事件通知给调试器，标志位不置位，不执行强制中断服务程序。

未定义操作码强制中断

若当前指令经 CPU 解码不包含有效的 XC164CM 操作码，即置位 TFR 寄存器中的 UNDOPC 标志，CPU 进入未定义操作码强制中断服务程序。引起未定义操作码强制中断的指令像 NOP 指令一样执行。

该特性可以用于仿真未实现的指令。强制中断服务程序可根据堆栈中的 IP 检查出错的指令，从而对未实现指令进行解码。为了恢复处理过程，在执行 RETI 指令之前，必须将未定义指令的指令字节数（由用户决定）加到堆栈中的 IP 值上。

程序存储器访问出错

当检测到程序存储器访问出错时，即置位寄存器 TFR 中的 PACER 标志，CPU 进入 PMI 访问出错强制中断服务程序。出现以下情况时，报告程序存储器访问出错：

- 当 Flash 存储器被禁止时，对其进行访问
- 当 Flash 存储器处于读保护时，从外部对其进行访问
- 读取 Flash 存储器时检测到双位错误
- 访问保留地址（见表 3-1 中的存储器映射）

- 在非仿真模式下，访问监控 RAM

出现访问错误时，会额外产生软件强制中断代码 1E9BH。

保护错误强制中断

在执行特定的被保护指令时，只要指令的操作码没有在指令的第二个字中重复两次，而且跟在操作码后的字节不是操作码的补码，即置位 TFR 寄存器中的 PRTFLT 标志，CPU 进入保护错误强制中断程序。受保护的指令包括 DISWDT、EINIT、IDLE、PWRDN、SRST、ENWDT 和 SRVWDT。引起保护错误强制中断的指令像 NOP 指令一样执行。

非法字操作数访问强制中断

只要试图从奇地址读取或写入字操作数，即置位 TFR 寄存器中的 ILLOPA 标志，CPU 进入非法字操作码访问强制中断服务程序。

6 通用系统控制功能

XC164CM 系统控制单元（SCU）概括了多种中央控制任务和产品特性。这些特性包括：看门狗定时器（WDT）或时钟产生（CGU）等通用功能模块、以及寄存器保护机制或复位产生等系统基本功能。

SCU 提供了以下通用功能：

- **系统复位**由复位控制模块产生，控制芯片的复位和启动行为（内部初始化）。该模块控制复位触发和复位时序，还能通过外部硬件控制 XC164CM 的基本配置。
- **时钟产生单元（CGU）**内嵌片上振荡器和锁相环（PLL）。该模块产生 XC164CM 所需的所有时钟信号，并将它们分配给各个模块，并可指示时钟产生系统的状态。
- **中央系统控制功能**包括所有的中央控制任务，如安全等级选择、休眠模式和掉电模式下的系统行为。根据不同的应用状态，安全等级控制状态机支持不同的安全等级（如保护模式和非保护模式）。
- **看门狗定时器（WDT）**提供了一种防止系统出错的故障保险机制。可检测长时间的系统故障，它在芯片初始化之后始终被使能。WDT 可工作在兼容模式或增强模式。
- **ID 控制模块**包括六个 ID 寄存器，用于提供最重要的芯片参数（芯片制造商、芯片型号及属性）。这些信息可用于自动测试选择。

6.1 系统复位

系统复位功能将 XC164CM 初始化至指定的缺省状态。进入缺省状态需满足以下条件之一：将引脚 **RSTIN**（硬件复位输入）拉低（硬件复位信号有效）、执行 **SRST** 指令（软件复位），或看门狗定时器溢出。

只要上述任何一个条件发生，微控制器则会经过一个内部复位过程进入预定义的缺省状态。软件复位时，挂起的内部保持状态被取消，完成当前的内部访问周期（若存在），结束 **LX-bus** 访问周期，随后，关闭 **IO** 引脚驱动器（高阻态）。硬件复位和看门狗复位会立即取消所有操作。

内部复位过程由几个连续的阶段组成，不同的复位源对应的复位阶段不同。通常情况下，复位可以异步（外部）或同步（内部）触发，总是同步结束。

表 6-1 复位序列

阶段	硬件复位 ¹⁾	看门狗复位	软件复位
1	外部复位阶段 包括外部触发信号解除 （ RSTIN = 1）之前的时间，芯片被异步复位	-----跳过-----	预复位阶段（关机） 包括片上模块正在运行的操作和挂起操作完成之前的时间
2	内部复位阶段 芯片的特定部分（外设系统和/或 CPU ）处于复位状态（复位控制模块除外）。 内部复位阶段的时间由复位事件定时器决定		
3	初始化阶段 根据缺省配置，设置芯片的特定部分（外设系统和/或 CPU ）： <ul style="list-style-type: none"> 内部启动：使用固定的缺省配置 引导程序加载器：程序代码从外部系统加载 		
4	运行（复位阶段终止） 此时开始执行用户软件		

1) 当电源电压超出规定的工作电压范围时，始终处于硬件复位状态，如上电过程。

6.1.1 复位源和复位阶段

系统复位分几个阶段执行，不同的复位触发源对应不同的复位序列（见表 6-1）。

外部复位阶段

当复位输入信号 $\overline{\text{RSTIN}}$ 为低时，硬件复位被异步触发。 $\overline{\text{RSTIN}}$ 上的毛刺抑制输入滤波器可滤除所有短于 10ns 的信号。为了确保能正确识别到 $\overline{\text{RSTIN}}$ 信号，该信号必须至少保持 100ns 的低电平，才可安全通过复位输入滤波器。电压达到稳定之后复位信号还需持续一段时间。

注：外部复位的最短持续时间必须确保硬件配置信号已达到其期望逻辑电平。

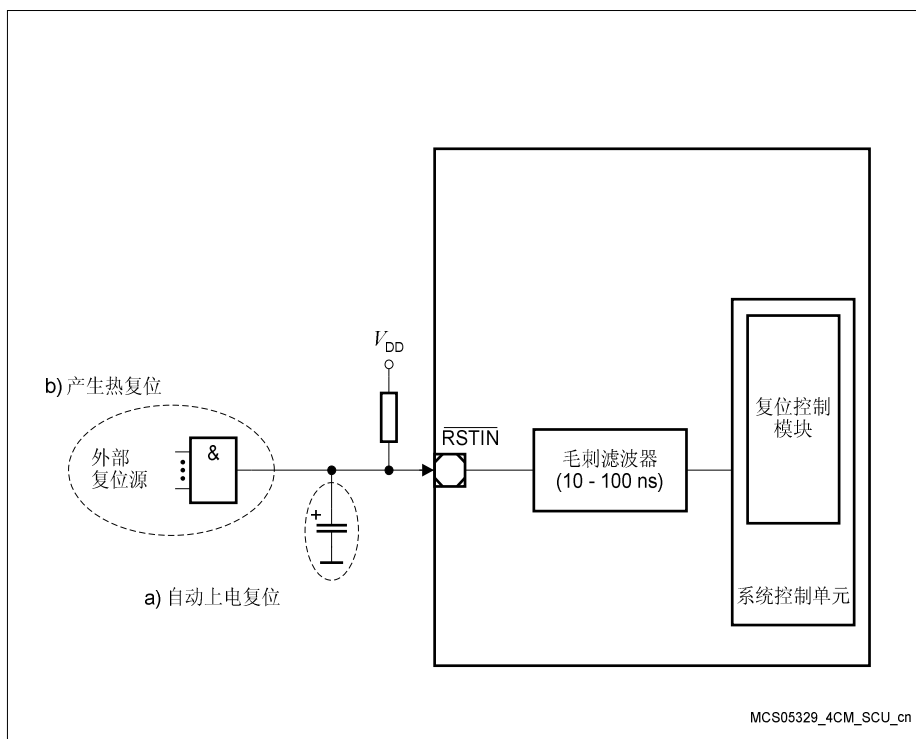


图 6-1 外部复位电路

输入引脚 $\overline{\text{RSTIN}}$ 上的硬件复位可由以下方式触发（见图 6-1）：

- 用一个外部上拉电阻与一个外部电容连接，可产生自动上电复位。
- 用一个外部上拉电阻与一个外部开关连接，可提供手动复位。
- $\overline{\text{RSTIN}}$ 还可以连接到其它逻辑的输出，产生热复位。

注：在外部复位阶段，整个芯片处于复位状态。 $\overline{\text{RSTIN}}$ 电平变高时，外部复位阶段同步结束。

预复位阶段

软件复位触发进入预复位阶段。在预复位阶段，CPU 首先将其流水线（包括所有回写缓存）处理完毕，然后将软件复位请求送至系统控制单元。该复位请求被激活之后，流水线保持清空。

软件复位请求一旦激活，SCU 向具有关机握手机制的工作模块发送关机请求（见章节 6.3.3）。一旦所有模块作出关机应答，预复位阶段完成。

响应关机请求时，EBC 将完成当前运行的总线周期。

内部复位阶段

内部复位开始后，内部复位条件变得有效。即，此时内部复位信号才真正加至模块上。若复位由硬件触发，内部复位条件可能已经有效。

注：复位控制模块（包括看门狗定时器）不被复位。

内部复位阶段的持续时间由寄存器 RSTCON 中的复位时间控制位域 RSTLEN 决定。看门狗定时器（WDT）的低字节用作计数复位时间。进入内部复位阶段后，定时器被清零，并以频率 f_{WDT} 递增计数。

硬件复位后，缺省计数频率为 $f_{\text{WDT}} = f_{\text{SYS}}/2 = f_{\text{MC}}/2$ ；内部复位源不改变当前的时钟设置和位域 WDTIN 的值，因此使用原先选择的 f_{WDT} 计数频率。

内部复位阶段的持续时间可由下面的公式计算得到：

$$t_{\text{RST}} = \frac{2^{(\text{RSTLEN})}}{f_{\text{WDT}}} \quad (6.1)$$

复位结束（初始化阶段）

内部复位阶段结束、将控制移交给软件之前，会发生如下操作：

- 置位寄存器 **SYSSTAT** 中的复位指示标志
- 选择初始化配置和复位起始地址
- 解除内部复位信号
- 若选择从引导程序加载模式启动，则执行相应操作

注：看门狗定时器从 0 继续递增计数。

6.1.2 复位后的状态

复位结束后，XC164CM 中的大多数模块进入预定义的缺省状态，从而确保可重复的初始条件，避免复位后的误动作。

XC164CM 寄存器的复位值

在执行复位序列期间，XC164CM 的寄存器被预设为缺省值。大多数特殊功能寄存器（SFR），包括系统寄存器、外设控制和数据寄存器被清零，因此，复位后所有外设和中断系统关闭或处于空闲状态。也存在少数例外情况，这些寄存器先被预初始化，该初始值可能固定、也可能由输入引脚控制（见[表 6-2](#)）。此外，有些寄存器在硬件复位、软件复位或 WDT 复位后具有特定的行为和复位值，这些寄存器包括：PLLCON、RSTCON、FOCON、SYSCON0、RTC_T14、RTC_T14REL、RTC_RTCL 和 RTC_RTCH，具体内容请参见相关寄存器描述。

表 6-2 复位后的非零寄存器

寄存器名称	初始值	注
DPP1	0001 _H	指向数据页 1
DPP2	0002 _H	指向数据页 2
DPP3	0003 _H	指向数据页 3
CP	FC00 _H	-
STKUN	FC00 _H	-
STKOV	FA00 _H	-
SP	FC00 _H	-
CPUCON1	0007 _H	-
CPUCON2	8FBB _H	-

寄存器名称	初始值	注
ONES	FFFF _H	固定值
PLLCON	27X0 _H	取决于启动模式
RSTCON	00X0 _H	取决于启动模式
VECSEG	00XX _H	取决于启动模式
SYSSTAT	XXXX _H	取决于当前状态
SYSCON3	9FD0 _H	RTC、TwinCAN、Flash、GPT、SSC0、ASC0、ADC 被使能
FCONCS7	0027 _H	-
ADDRSEL7	2000 _H	-
CCU6_INP	3940 _H	-
ADC_CTR0	1000 _H	-
CAN_ACR	0001 _H	-
CAN_BCR	0001 _H	-
RTCCON	8003 _H	-
RTC_T14	UUUU _H	只受 RTC 复位影响（置位寄存器 SYSCON0 中的位 RTCRST 触发 RTC 复位），不受硬件复位、软件复位和看门狗复位的影响。
RTC_T14REL	UUUU _H	
RTC_RTCL	UUUU _H	
RTC_RTCH	UUUU _H	

复位后的操作

内部复位条件解除之后，XC164CM 从选定的程序存储器地址单元中（由配置决定）读取第一条指令。规定在该首地址单元中存放一条跳转指令，跳转到实际的初始化程序（可位于任意地址单元）。

注：如果硬件复位期间激活引导程序加载模式，则 XC164CM 不从程序存储器读取指令。

标准引导程序加载器通过串口 ASC0 获取数据。

复位后的看门狗定时器操作

内部复位结束后，看门狗定时器继续运行，并以当前选择的时钟频率 f_{WDT} 计数。看门狗/软件复位后 f_{WDT} 不会改变，硬件复位后计数频率 $f_{WDT} = f_{SYS}/2 = f_{MC}/2$ 。缺省重载值为 00_H。因此，内部复位完成后（取决于选择的复位时间长度），看门狗将在 2^{16} 个时钟周期后溢出（硬件复位后看门狗将在 2^{17} 个 f_{MC} 周期后溢出），除非它被禁止、被服务或重新编程。如果由看门狗定时器溢出引起系统复位，寄存器 SYSSTAT 中的 WDTR（看门狗定时器复位指示）标志被置 1，指示造成内部复位（跳转到软件初始化程序）的原因。其它类型的复位之后，WDTR 被清零。内部复位完成后，看门狗定时器可以用 DISWDT（禁止看门狗定时器）指令禁止，兼容模式下该指令要在 EINIT 指令之前执行；增强模式下 DISWDT 指令可在任意时刻执行。

复位后的片上 RAM 区域

软件复位和 WDT 复位时，片上 RAM 的大部分内容被保留。只有两种例外情况：

- 一部分 DPRAM 区域（区域 00'FBA0_H ... 00'FC1F_H）可能在初始化阶段（见 [表 6-1](#)）被改变，因此该区域不应存储 WDT/SW 复位后需要保留的数据。
- 在引导程序加载操作期间，串行接收的数据存储在起始地址为 E0'0000_H 的 PSRAM 中。

由于硬件复位可（与内部操作）异步发生，它可能中断当前的写操作，从而无意破坏片上 RAM 的内容。如果在掉电、休眠或空闲模式下硬件复位，并且不存在 PEC 传送，则 RAM 内容被保留。

注：上电硬件复位后，RAM 内容未定义。

复位后的端口

内部复位期间，方向寄存器被清零，XC164CM 的所有端口引脚被设置为输入，引脚驱动切换至高阻态。从而保证了 XC164CM 和外部器件不会把同一引脚驱动成不同的电平。

复位时若激活片上引导程序加载，引脚 TxD0（复用端口功能）在接收到零字节后将切换到输出模式。

所有其它引脚保持高阻态，直到它们被软件或外设操作修改。

6.1.3 特定应用的初始化

复位之后，XC164CM 的各个模块必须根据给定的应用进行相应初始化。初始化操作和 XC164CM 在应用中需要执行的任务，以及一些系统属性（如工作频率、外接电路等）有关。

通常，在 XC164CM 准备运行应用软件之前，必须执行以下初始化：

引导程序加载模式（可选）

引导程序加载模式可用于初始 Flash 编程。

系统堆栈

系统堆栈（容量、堆栈指针、上下限寄存器）的缺省设置可根据应用需求进行调整。复位后，寄存器 SP 和 STKUN 有相同的复位值 00'FC00_H，寄存器 STKOV 的复位值为 00'FA00_H。根据复位初始化缺省设置，DPRAM 中的系统堆栈大小为 256 个字，堆栈从 00'FBFE_H 向下生长。系统堆栈也可移至 DSRAM 区，其容量可根据应用需要改变。

注：内部复位结束后中断系统应保持禁止状态，直到 SP 被初始化。

尽管中断系统被禁止，但强制中断（包括 NMI）有可能发生。

寄存器组

全局寄存器组的位置由上下文指针（CP）决定，在使用通用寄存器（GPR）之前，可修改 CP 以指向应用特定的寄存器组。复位后，寄存器 CP 的值为 FC00_H，即寄存器组从 00'FC00_H 向上生长。

片上 RAM

根据应用需求，用户可能希望在正常执行程序之前，对部分内部可写存储器（DPRAM/DSRAM/PSRAM）进行初始化。通过设定 CP 寄存器选定寄存器组后，就可方便的通过间接寻址方式对所需内部存储器部分进行初始化。

中断系统

复位后，各中断节点和全局中断系统被禁止。若要使能中断请求，各中断节点必须指定各自的中断优先级，并被使能。如果缺省属性不合适，可调整中断向量表。寄存器 VECSEG 定义向量表的地址，寄存器 CPUCON1 中的位域 VECSC 定义中断向量的间距。向量地址中存放着指向各自的特殊处理子程序的指针（快速中断机制除外）。必须通过置位寄存器 PSW 的位 IEN 全局使能中断系统。为了避免由于使用未经初始化的堆

栈指针进行堆栈操作，导致破坏内部存储单元的内容，必须注意：在初始化结束之前不要使能中断系统。

端口

通常情况下，XC164CM 的所有端口在复位后切换至输入状态。复位后，有些引脚可被自动控制，如引导模式下的 TxD 引脚。用作通用 IO 的引脚必须由软件初始化。给定引脚所需的端口模式（输入/输出、漏极开路/推挽、输入阈值等）取决于它所要实现的功能。

外设

复位时，XC164CM 的片上外设模块进入指定的缺省状态（见各外设模块的描述），缺省状态下外设被禁止。若要使用某个外设，必须根据应用需求对其初始化。

外设初始化包括使能相应外设、选择工作模式（如计数器/定时器）、设定工作参数（如波特率）、使能接口引脚（若需要）、为中断节点设定中断优先级等。

进行这些标准初始化之后，可能还要根据应用执行特定的操作，如指定输出引脚的电平、通过接口发送代码、锁存输入电平等。

看门狗定时器

复位之后，看门狗定时器有效，以缺省的周期计数。若要保持看门狗定时器有效，应通过选择适当的预分频值和重载值来设定期望的计数周期，否则看门狗定时器必须在 EINIT 指令之前被禁止。

初始化终止

软件初始化子程序应由 EINIT 指令终止。该指令是一条受保护指令。

执行 EINIT 指令后，有如下结果：

- 禁止 DISWDT 指令执行（除非选择增强模式）
- 将寄存器的安全等级切换至“写保护模式”（见[章节 6.3.5](#)）

6.1.4 系统启动配置

尽管 XC164CM 的大多数可编程特性在初始化阶段或程序执行期间由软件设置，但由于有些特性会在首次执行程序时使用到，因此必须更早进行设置。

内部复位结束时，通过锁存多个引脚上的逻辑电平来完成系统启动配置。复位期间，这些线上的内部上拉电阻有效，从而确保引脚在没有外部驱动时保持无效/缺省电平。要选择特定配置时，外部下拉/上拉电阻可能会取代缺省电平。此外，并非所有引脚必须被控制用作所有模式。

因此，只需最少的外部电路即可实现多种系统启动配置。如果引脚 $\overline{\text{TRST}}$ 锁存低电平，所有四个配置引脚被忽略（见**表 6-3**）。

注：用于启动配置的引脚上的负载必须相对于内部上拉/下拉电阻足够小，以保持缺省电平；或者必须通过外部上拉/下拉电阻保证该电平。

若要取代引脚上的缺省电平，必须通过外部上拉/下拉来实现。

要确保在复位结束时达到有效的目标电平。

有专门的应用指南对此进行详细说明。

一旦达到目标电平，硬件复位可被立即终止。XC164CM 自动进入等待状态，直到振荡器、PLL 以及 Flash 启动并可操作。

硬件复位后，一旦内部复位阶段结束，引脚 P9.4、P9.5、P1H.4、P1H.5 上的电平被锁存。（见**表 6-1**）。根据需要的配置，由外部控制其中的二至四个引脚（见**表 6-3**）。

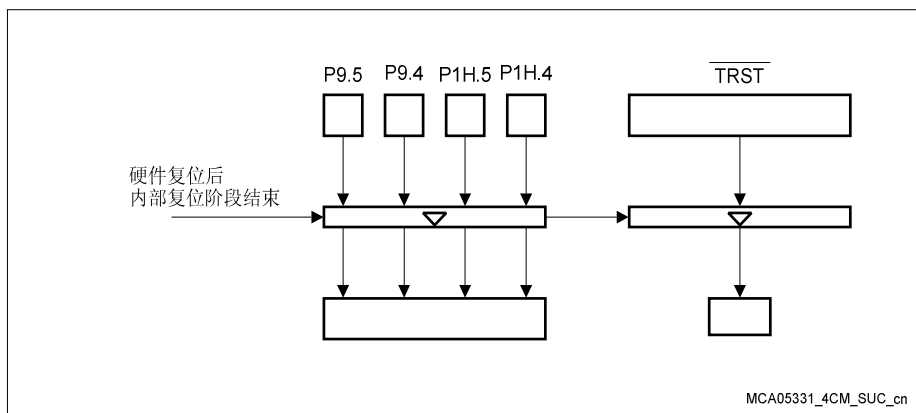


图 6-2 锁存配置

表 6-3 通过外部电路进行基本启动配置

锁存配置	$\overline{\text{TRST}}$	P1H.5	P1H.4	P9.5	P9.4
内部启动（OCDS 禁用）	0	X	X	X	X
内部启动	1	X	X	1	1
通过 ASC0 启动	1	X	X	0	1
自适应模式	1	1	1	0	0
保留	1	所有其它组合			

注：当 $\overline{\text{TRST}} = 1$ 且硬件复位信号有效时，配置引脚的上拉电阻被激活。

XC164CM 的初始时钟产生模式由寄存器 PLLCON 的复位值决定，选择旁路模式：

- 内部启动：PLLCON= 2710_H（预分频因子 2:1，覆盖最大的频率范围）
- 引导程序加载模式：PLLCON= 2700_H（直接驱动达到最大波特率）

用户可在任意时刻改变该启动配置。

片上引导程序加载器可通过串口 ASC0 将启动代码转移至 XC164CM 的内部 PSRAM 中。XC164CM 随后在 PSRAM 中执行装载的启动代码。

缺省：XC164CM 从地址 C0'0000_H 开始读取指令。引导程序加载器关闭。

自适应模式

该模式下，XC164CM 进入被动状态，与复位期间的状态相似。XC164CM 的引脚悬空至三态，或通过内部上拉/下拉电阻使引脚失效。片上振荡器和实时时钟禁止。

该模式使得用户板上的 XC164CM 处于虚拟关闭状态。即使 XC164CM 保留在电路板上，仍可使用仿真器来控制电路板电路。复位之后（不选择自适应模式），板上的 XC164CM 能够恢复对电路板的控制。

缺省：自适应模式关闭。

注：当 XTAL1 由外部时钟发生器驱动时(同时 XTAL2 断开)，该时钟信号还能用于驱动仿真器。

不过，若使用晶振，只有当至少 XC164CM 的 XTAL2 和电路断开时，仿真器的振荡器才能使用该晶振（自适应模式下输出 XTAL2 将被驱动为高电平）。

只有通过外部复位才能激活自适应模式。

6.1.5 复位行为控制

XC164CM 的复位行为由一组控制/状态寄存器进行控制。根据复位源的不同，初始化程序可根据状态信息执行不同的操作。

根据应用需求，通过复位控制寄存器 **RSTCON** 设定内部复位阶段的持续时间。

RSTCON

复位控制寄存器															mem (F1E0H/--)		复位值: 00X0 _H ^{1) 2)}	
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0			
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-		RSTLEN	
-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-			rw

- 1) 该复位值仅对硬件复位有效。
 2) 内部和 ASC 引导程序加载启动之后，该寄存器的复位值为 0010_H。

符号	位序号	读写类型	功能描述
RSTLEN	[2:0]	rw	复位持续时间控制¹⁾ 下一次内部复位阶段的持续时间为 $t_{RST} = t_{WDT} \times 2^{RSTLEN}$ 000 1 t_{WDT} : 硬件复位后持续时间的缺省值 111 128 t_{WDT} : 最大持续时间

1) RSTLEN 总是对下一次复位有效。不过，初始的上电复位由外部硬件控制，持续时间长于任何可配置的复位过程。
 注: **RSTCON** 由寄存器安全机制保护（见 [章节 6.3.5](#)），只能通过长寻址（mem）方式访问。

6.2 时钟产生

XC164CM 微控制器硬件和片上外设的所有操作都由时钟产生单元（CGU）提供的时钟信号控制。

参考时钟分三个阶段产生：

振荡器

片上皮尔斯（Pierce）振荡器（主振荡器）可以与外部晶振及适当的振荡电路配合工作，也可以由外部振荡器或其它时钟源驱动。

时钟产生和频率控制

主振荡器的输入时钟信号驱动微控制器：

- 直接经预分频处理，预分频因子可编程设定（1...60）；可提供相位耦合操作（比例因子 = 1）；或使器件低频工作，降低功耗（比例因子 $\gg 1$ ）。
- 先送入片上锁相环（PLL），在低频输入下实现了最大系统性能。

时钟分配

通过独立的时钟驱动器将时钟信号分配给 CPU 及多个外设模块。器件特定部分的时钟可由预分频时钟信号提供。

注：由一个独立的时钟驱动器将经过预分频处理的主振荡器时钟分配给 RTC 模块，因此 RTC 不受时钟控制功能的影响。

6.2.1 振荡器

XC164CM 的主振荡器是一个功率优化（低功耗）的皮尔斯振荡器，由一个反相器和一个反馈元件组成。引脚 XTAL1 和 XTAL2 将反相器连接到外部晶振。标准的外部振荡电路（见图 6-3）由晶振、两个小终端电容和限制晶振电流的串联电阻（ R_{x2} ）组成。测试电阻（ R_Q ）可临时插入，用于测量振荡器电路的振荡裕度（负电阻）。

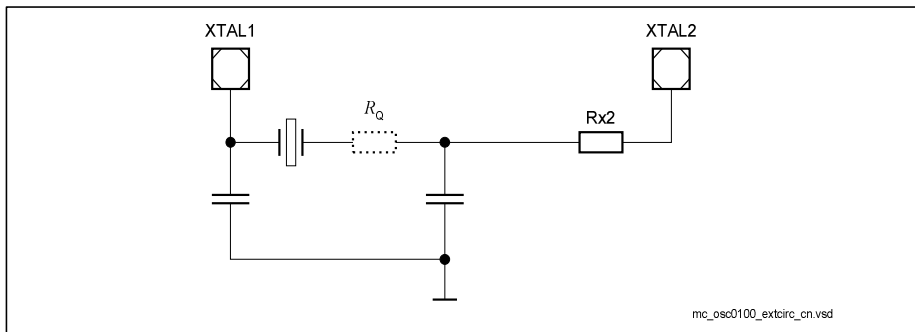


图 6-3 外部（主）振荡器电路

片上振荡器经过优化，可以接收 4 至 16MHz 的外部输入频率。

外部时钟信号（如来自外部振荡器或主器件的时钟）可从 XTAL1 输入。此时皮尔斯振荡器无需产生振荡，而是由输入信号驱动。这种情况下，输入频率的变化范围可从 0 到 50 MHz（请注意：最大可用的输入频率受器件最大时钟频率的限制）。

*注：为了校验振荡器—晶振系统的输入振幅，并确定实际振荡裕度（容许裕量或负电阻），强烈推荐对最终的目标系统进行**振荡器测量**。*

测量技术请参阅振荡器应用指南（请通过产品代理或互联网获取）。

除非 RTC 需要保持运行，否则主振荡器在掉电模式和休眠模式下自动关闭。在只需维持最少系统功能的情况下，关闭主振荡器可进一步降低功耗。

主振荡器增益降低

在硬件复位期间和复位之后，主振荡器的驱动电平（增益）较高，从而确保起振时可以安全启动（强制晶振振荡）。晶振起振由位 **OSCLOCK = 1** 指示。振荡器启动后，一旦达到振荡稳定（振幅超过最大值的 90%），主振荡器增益可以降低。该操作降低了振荡器功耗，这在省电模式下尤其重要。

增益降低由软件控制，在现有软件中可见。置位寄存器 **SYSCON0** 中的位 **OSCGRED**（见 **章节 6.3**），可降低振荡器增益。由于振荡器振幅不能直接测量，因此在使能增益降低之前需要延时大约 2^{15} 个振荡器时钟周期。

由寄存器 **SYSSTAT** 中的位 **OSCSTAB (=1)** 指示已产生 2^{15} 个振荡器时钟周期。

OSCSTAB = 0 时，禁止降低振荡器增益。因此，软件可在任意时刻置位 **OSCGRED**。如果在 **OSCSTAB = 1** 之前置位 **OSCGRED**，增益降低会自动延迟发生。

*注：经过延迟之后（由 **OSCSTAB = 1** 指示），优化的振荡器电路的振幅已超过最大值的 90%。*

*也必须在增益降低模式下（如果应用需要该模式）对振荡器-晶振系统进行**振荡器测量**（容许裕量或负阻）。*

*如果在休眠模式下关闭主振荡器，位 **OSCLOCK** 和 **OSCSTAB** 被清零，振荡器重新启动，从而保证了唤醒后振荡器可安全启动。*

6.2.2 时钟产生和频率控制

时钟产生单元通过可编程片上 PLL 和多种预分频处理，非常灵活的为 XC164CM 提供时钟信号。XC164CM 的内部操作由内部主时钟 f_{MC} 控制。主时钟 f_{MC} 是参考时钟信号，用作 TwinCAN 的输入时钟，并输出给外部系统。

CPU 和 EBC 由 CPU 时钟信号 f_{CPU} 驱动。CPU 时钟频率可以和主时钟相同（ $f_{CPU} = f_{MC}$ ），也可以由主时钟 2 分频： $f_{CPU} = f_{MC}/2$ 。该分频因子由寄存器 SYSCON1 的位 CPSYS 选择。

其它外设使用系统时钟信号 f_{SYS} ，其频率和 CPU 时钟相同（ $f_{SYS} = f_{CPU}$ ）。

振荡器时钟频率可通过片上 PLL 倍频（倍频因子可编程），也可通过预分频器分频（分频因子可编程）。通过这些选择，主时钟可被调整至很宽的频率范围。即使晶振频率较低，通过 PLL 操作也能实现最优性能；振荡器时钟分频可使系统低频运行，大大降低功耗。

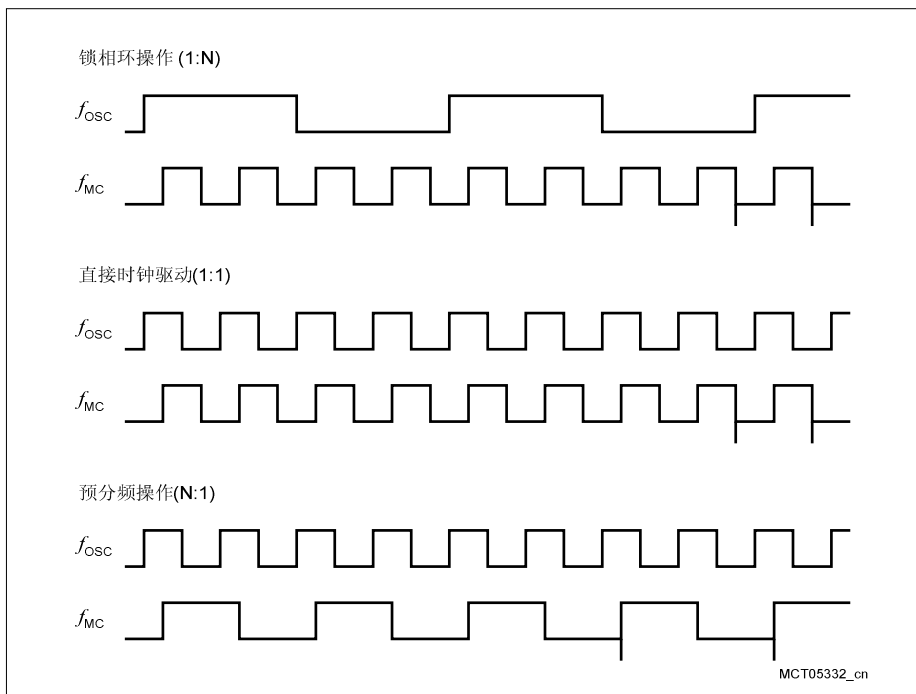


图 6-4 主时钟的产生机制

注：图 6-4 所示的 PLL 操作中 PLL 因子为 1:4，预分频操作中分频因子为 2:1。

时钟产生单元（CGU）包括以下功能，用来产生 XC164CM 所需的时钟信号：

- 根据用户设定的模式和因子从振荡器时钟产生主时钟信号
- 为特定功能区产生时钟信号
- 根据 XC164CM 的工作模式控制振荡器操作
- 如果检测到时钟系统故障，产生中断请求

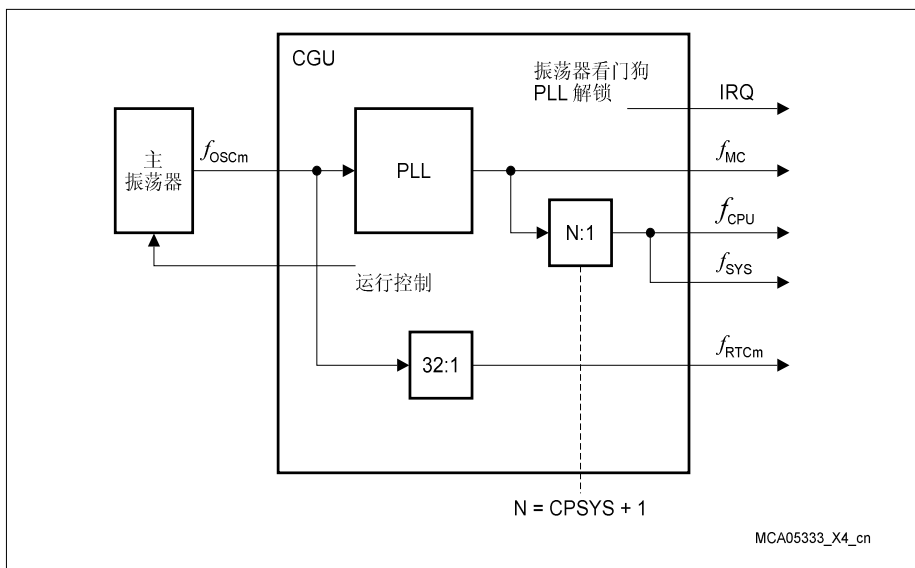


图 6-5 时钟产生单元的基本结构

注：CPU 时钟和系统时钟的分频因子由寄存器 SYSCON1 中的位 CPSYS 选择。

主时钟信号由高度灵活的片上 PLL 产生。PLL 模块可将振荡器时钟信号倍频，倍频因子可编程设定（1:0.15 ... 1:10），从而即使在晶振频率较低的情况下也可实现系统的高性能；旁路模式下，振荡器频率经 1:1 ... 60:1 分频，主时钟或者和振荡器时钟信号直接相连（1:1），或者用于降低系统频率以节约功耗。

主时钟的产生方式及相应的参数通过 PLL 控制寄存器 PLLCON 进行选择。

PLLCON

PLL 控制寄存器

ESFR (F1D0_H/E8_H)

复位值: 27X0_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PLL WRI	PLL CTRL	PLLMUL				PLLVB		PLLIDIV		PLLODIV					
rh	rw	rw				rw		rw		rwh					

符号	位序号	读写类型	功能描述
PLLWRI	15	rh	PLL 写忽略标志 0 寄存器 PLLCON 可写 1 寄存器 PLLCON 的写周期被忽略
PLLCTRL	[14:13]	rw	PLL 运行控制 00 旁路 PLL 时钟倍频, VCO 关闭 01 旁路 PLL 时钟倍频, VCO 运行 10 使用 VCO 时钟, 输入时钟关闭 11 使用 VCO 时钟, 输入时钟连接
PLLMUL	[12:8]	rw	PLL 倍频因子 PLL将输入频率和倍频因子相乘 （有效值: 1'1111 _B ... 0'0111 _B ） ¹⁾ $f_{VCO} = f_{IN} \times (PLLMUL+1)$
PLLVB	[7:6]	rw	PLL VCO 频带选择 ²⁾ 设定值, VCO输出频率, 基频 00 100 ... 150 MHz, 20 ... 80 MHz 01 150 ... 200 MHz, 40 ... 130 MHz 10 200 ... 250 MHz, 60 ... 180 MHz 11 保留
PLLIDIV	[5:4]	rw	PLL 输入分频因子 将振荡器频率调整到规定的PLL输入频率范围内（有效值: 11 _B ... 00 _B ） $f_{IN} = f_{OSC} / (PLLIDIV+1)$
PLLODIV	[3:0]	rwh	PLL 输出分频因子 将PLL输出频率调整到所需的CPU频率（有

符号	位序号	读写类型	功能描述
			效值 1110 _B ... 0000 _B ³⁾ $f_{MC} = f_{VCO} / (PLLODIV + 1)$

- 1) 倍频因子小于 8 (PLLMUL = 7) 时可能影响稳定性。此时，增强的噪声敏感度可能导致不正确的 VCO 频率。
- 2) 所选择的 VCO 频带必须包含期望的 VCO 频率 (8 MHz × 20 = 160 MHz → 频带 01_B)。
- 3) 值 1111_B 保留用作紧急模式操作，不能通过软件进入该模式。

注：PLLCON 受寄存器安全机制保护（见章节 6.3.5）。其复位值取决于系统启动配置。

寄存器 PLLCON 不受软件复位影响；受硬件复位和看门狗复位的影响，这两种复位操作会使其恢复硬件复位值。

状态机根据寄存器 PLLCON 的设定控制时钟的产生。位 PLLWRI = 0 指示状态机准备使用 PLLCON 的新设定值。为了支持监控，寄存器 PLLCON 被写入时，接受新的时钟配置选择；被读取时返回时钟产生的实际状态。

PLL 模块包括倍频逻辑、一组预分频器、锁相检测、以及振荡器看门狗。

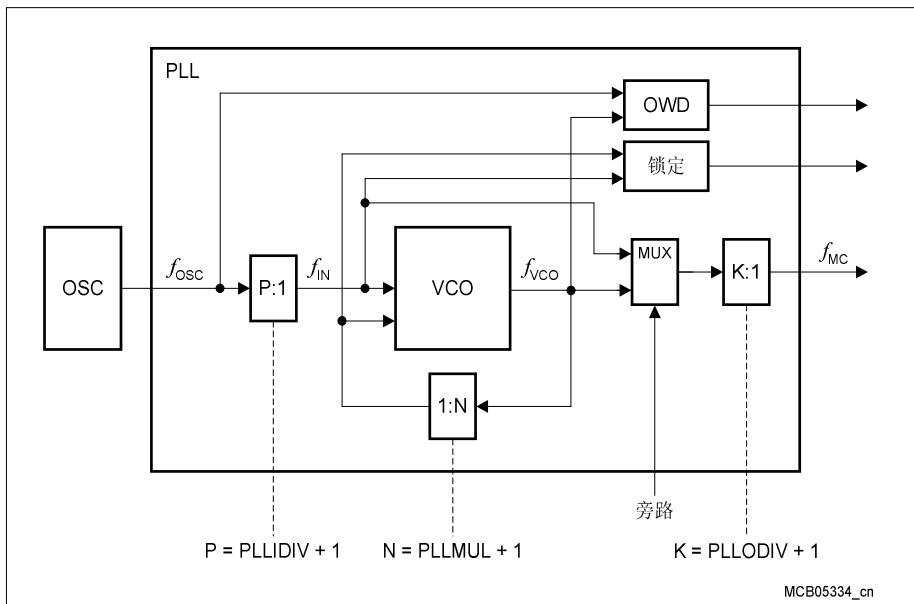


图 6-6 PLL 框图

PLL 操作

设定 PLL 操作之后（PLLCTRL = 11_B），XC164CM 的输入时钟送入片上锁相环电路，片上锁相环电路可将该输入时钟倍频，倍频因子最大为 $F = 10$ ，产生占空比为 50% 的主时钟信号，即 $f_{MC} = f_{osc} \times F$ 。

片上 PLL 电路允许 XC164CM 以低频外部时钟运行，同时仍可提供最佳系统性能。PLL 还提供了故障保险机制，可检测频率偏差，并在外部时钟出错的情况执行紧急操作。

PLL 检测到输入时钟信号丢失时，会产生中断请求。该报警中断表明 PLL 频率不再被锁定，即不再和振荡器频率相关。当输入时钟不稳定、或者当输入时钟完全失效（如晶振损坏）时，会发生这种情况。针对这种情况，同步机制会将 PLL 的输出频率降至 PLL 基频（取决于由位域 PLLVB 选择的 VCO 频带），并选择安全输出分频因子 $K = 16$ 。在缺少外部时钟的情况下，PLL 为 CPU 继续提供基频信号以执行紧急操作。此时主时钟频率 $f_{MCe} = f_{VCObase}/16$ 。

注：在硬件复位期间，选择最低 VCO 频带和分频因子 16。

即使没有外部时钟信号，上电后 PLL 仍可提供稳定的时钟信号（此时 PLL 工作在基频）。一旦外部时钟信号可用，PLL 会立刻开始执行同步操作。外部时钟稳定振荡在规定的频率范围内之后，PLL 锁定该外部时钟。这意味着，PLL 将和该时钟同步，频率为 $F \times f_{osc}$ 。

PLL 电路始终使主时钟和输入时钟同步，该同步操作平稳进行，即主时钟频率不会发生突变。由于外部频率是 PLL 输出频率的 $1/F$ ，输出频率可能略高或略低于期望频率。这种微小的偏差会导致 f_{MC} 抖动，影响主时钟的周期。该抖动对长周期没有影响，对于短周期（1...4 个 CPU 时钟周期），抖动要低于 9%。

时钟信号要经过多个模块处理（见图 6-6）。时钟总倍频因子 F 由以下几部分组成：输入分频因子（P:1）、倍频因子（1:N）、输出分频因子（K:1），因此

$$F = (PLL\text{MUL} + 1) / ((PLL\text{IDIV} + 1) \times (PLL\text{ODIV} + 1))。$$

输入时钟分频器将振荡器时钟频率调整至 PLL 优化的输入频率范围（ $f_{in} = f_{osc} / (PLL\text{IDIV} + 1) = 4 \dots 35 \text{ MHz}$ ）。

PLL 内核将调整后的输入频率（在选定的 VCO 频带内）和可选的倍频因子相乘（ $f_{vco} = f_{in} \times (PLL\text{MUL} + 1)$ ）。必须根据期望的 VCO 频率选择有效的 VCO 频带（PLLVB）。

注：PLL 锁定到一个给定因子以确保产生正确的 CPU 时钟信号时、或者不使用 PLL 产生时钟时，PLL 内核可旁路。

输出时钟分频器根据可编程分频因子对 VCO 输出频率进行分频处理，以产生主时钟信号（ $f_{MC} = f_{vco} / (PLL\text{ODIV} + 1)$ ）。可通过调整该分频因子控制 XC164CM 的工作频率，而无须对 PLL 内核重新设定。

图 6-7 归纳了主时钟的产生步骤。

若要由最低输入时钟频率（4MHz）产生可能的最高主时钟频率（40MHz），则应使用最大倍频因子 F_{\max} 。因此，最大可用倍频因子 $F_{\max} = 10$ 。

应用软件可通过寄存器 PLLCON 和 SYSCON1 选择最佳时钟产生模式。复位后 XC164CM 进入缺省时钟产生模式。

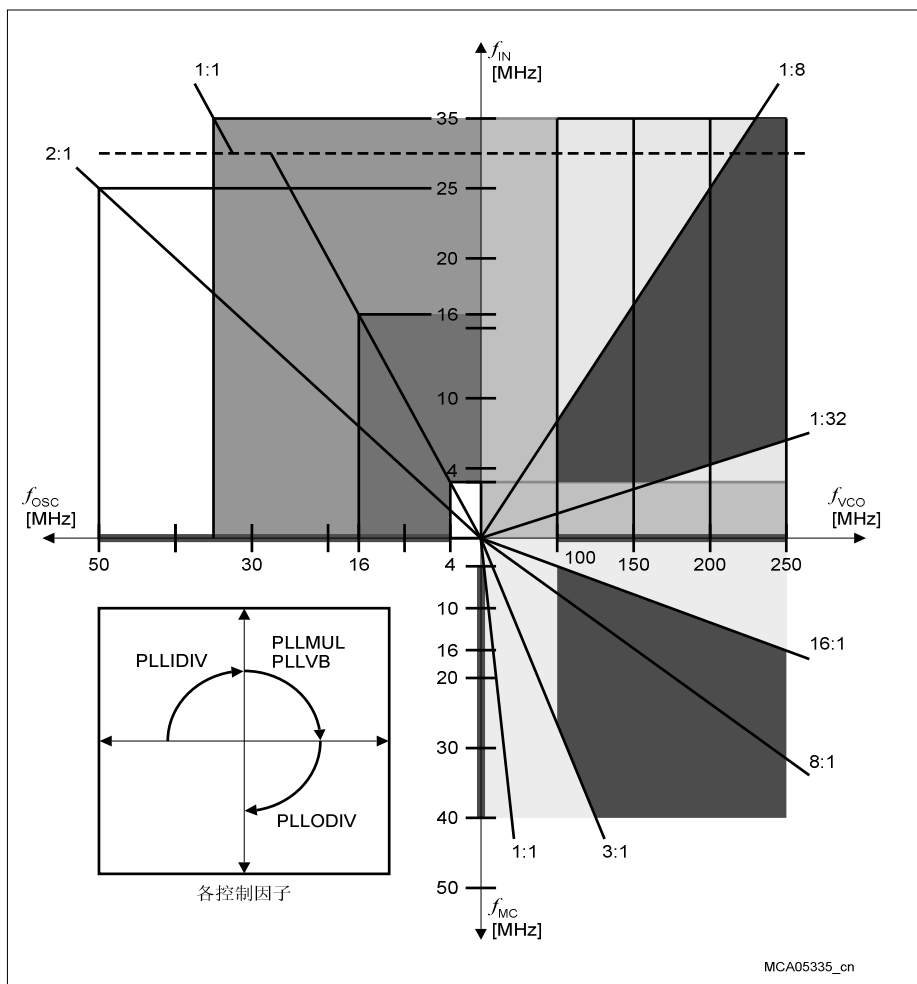


图 6-7 有效时钟频带

旁路操作

PLL 设置为旁路模式时（ $PLLCTRL = 0XB$ ），时钟信号不经过 PLL 内核，内部振荡器信号（输入时钟信号 XTAL1）经输入和输出预分频处理后产生主时钟：

$$f_{MC} = f_{OSC} / ((PLLIDIV + 1) \times (PLLODIV + 1))$$

若这两个预分频因子都设为 1（ $PLLIDIV = PLLODIV = 0$ ），则主时钟 f_{MC} 和 f_{OSC} 相同，因此 f_{MC} 的高低电平时间由输入时钟 f_{OSC} 的占空比决定。

若这两个预分频因子都选择最大值，则可得到最低的主时钟频率：

$$f_{MCmin} = f_{OSC} / ((3 + 1) \times (14 + 1)) = f_{OSC} / 60$$

主时钟占空比

主时钟信号 f_{MC} 最后经输出预分频处理（ $K = PLLODIV + 1$ ）。 f_{MC} 的占空比取决于所选择的输出预分频因子。选择偶数因子，占空比为 50%；选择奇数因子，占空比为 $(K - 1) / (K \times 2)$ 。最坏的情况为 $K = 3$ ，此时占空比为 $(3 - 1) / (3 \times 2) = 2/6 = 33\%$ 。

6.2.3 时钟分配

由多个时钟驱动器将时钟信号分配给相关的控制器模块，时钟域归纳见表 6-4。实时时钟模块 RTC 的时钟源来自经预分频处理的主振荡器时钟，由一个独立的时钟驱动器驱动。

表 6-4 时钟域

时钟域	域时钟	正常工作模式	空闲模式	休眠、掉电模式	连接电路	模块时钟
LXBus	f_{MC}	打开	打开	关闭	TwinCAN	f_{CAN}
					SCU ¹⁾	-
CPU	f_{CPU}	打开	关闭	关闭	CPU、DPRAM、EBC, OCDS、Flash/ROM、PSRAM、DSRAM	-
PDBus	f_{SYS}	打开	打开	关闭	ADC	f_{ADC}
					ASC0、ASC1	f_{ASC}
					CAPCOM2	f_{CC}
					CAPCOM6	f_{CC6}
					GPT12	f_{GPT}
					SSC0、SSC1	f_{SSC}
					端口、RTC ²⁾ 、WDT、SCU (中断控制、寄存器访问) ¹⁾	-
RTC	f_{RTC}	打开	打开	打开/关闭	RTC ²⁾	-

1) 由于时钟产生单元是 SCU 的一部分，因此 SCU 从属于多个时钟域。

2) RTC 从属于 PDBus 时钟域（提供 RTC 工作时钟）。RTC 的计数时钟信号直接来自主振荡器。

注：所有 PDBus 外设的时钟信号均为 f_{SYS} 。不过，在外设章节的描述中，时钟信号根据外设名称来命名。见表 6-4 中“模块时钟”列所示。

6.2.4 振荡器看门狗

XC164CM 提供了一个振荡器看门狗（OWD），监控旁路模式下（此时主时钟信号不由 PLL 内核提供）送入片上振荡器（由外部晶振或外部时钟驱动）输入引脚 XTAL1 上的时钟信号。PLL 提供了一个 VCO 时钟信号（基频），用于监控振荡器时钟的跳变。该 VCO 时钟和 XTAL1 时钟无关。若期望的振荡器时钟跳变未出现，OWD 将激活 PLL 解锁/OWD 中断节点，并为 CPU 提供一个紧急时钟，以代替所选的振荡器时钟。这种情况下，VCO 在选定频带内以基频振荡。紧急时钟频率为 $f_{VCO}/16$ 。

主时钟由 PLL 提供时，若振荡器时钟出错，系统工作在 PLL 基频。

利用这个紧急时钟信号（通常频率较低），CPU 可执行关机序列，使系统进入一个确定的、安全空闲状态；或者在系统性能降低的情况下进行紧急系统操作。

注：在特殊情况下，如果旁路模式选择了较高的预分频因子，紧急时钟频率也要高于原先期望的频率。

关闭 VCO（PLLCTRL = 00_B）可**禁止振荡器看门狗**。此时 VCO 处于空闲状态，不提供时钟信号，主时钟直接从振荡器时钟获得。该操作降低了功耗，但在振荡器时钟丢失的情况下不会产生中断请求。

6.2.5 中断产生

当 PLL 退出锁相状态，或者当 OWD 检测到一个不正确的时钟输入信号时，CGU 发出中断请求。

PLL_IC

PLL 中断控制寄存器

ESFR (F19E_H/CF_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	GPX	PLL IR	PLL IE	ILVL				GLVL	
-	-	-	-	-	-	-	rw	rwh	rw		rw			rw	

注：控制位域的具体解释请参阅通用中断控制寄存器的描述。

6.2.6 外部时钟信号的产生

XC164CM 为外部电路提供了一个时钟信号，可用于驱动片外外设，或者用作时钟参考。可选择两种输出类型：

- CLKOUT 直接输出主时钟信号 f_{MC} ，主要用于时序参考
- FOUT 输出一个频率可编程的时钟信号，用于驱动和控制外部电路

频率可编程的时钟输出信号 f_{OUT} 可由软件控制（与 CLKOUT 相反），从而可适用于不同的外部电路。可编程特性还将功率管理提升到系统级；XC164CM 的外部电路（片外外设等）也会受到影响，即外部电路可以运行在不同的频率，或暂时被完全关闭。

f_{OUT} 时钟信号由一个重载计数器产生，因此可用较小的步长进行选择输出频率。可选择通过翻转锁存器提供占空比为 50% 的时钟信号。

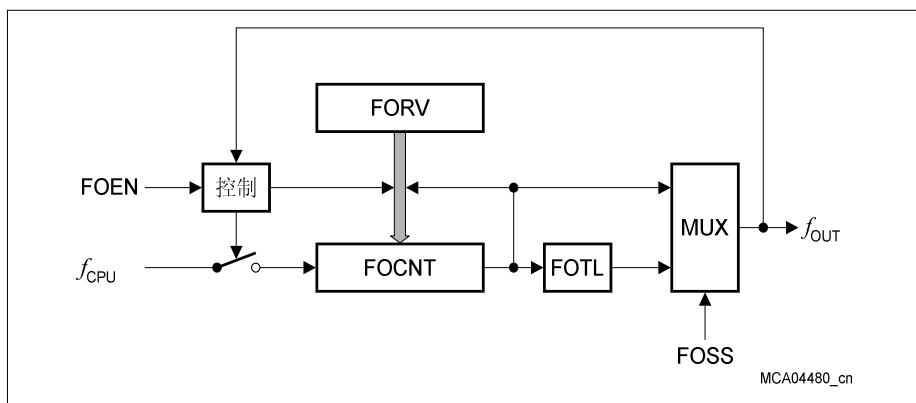


图 6-8 时钟输出信号产生

信号 f_{OUT} 始终提供完整的输出周期（见图 6-9）：

- 当启动 f_{OUT} 时（FOEN → 1），FOCNT 由 FORV 加载
- 当停止 f_{OUT} 时（FOEN → 0）， f_{OUT} 变为（或等于）0 时 FOCNT 停止

寄存器 **FOCON** 控制输出信号的产生（输出信号类型、频率、波形、激活），并指示所有状态信息（计数值，**FOTL**）。

FOCON

频率输出控制寄存器

SFR (FFAA_H/D5_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
FO EN	FO SS	FORV						CLK EN	FO TL	FOCNT					
rw	rw	rw						rw	rh	rh					

符号	位序号	读写类型	功能描述
FOEN	15	rw	频率输出使能 0 当信号 f_{OUT} 为/变低时停止产生频率输出 1 FOCNT 运行, f_{OUT} 送至引脚。 f_{OUT} 发生 0 到 1 的跳变后开始首次重载
FOSS	14	rw	频率输出信号选择 0 翻转锁存输出: 占空比 = 50% 1 重载计数器输出: 占空比取决于 FORV
FORV	[13:8]	rw	频率输出重载值 每次 FOCNT 下溢时, 将重载值复制到 FOCNT 中。
CLKEN	7	rw	CLKOUT 使能 0 CLKOUT 信号被禁止 P3.15 用作通用 IO 或输出 FOUT (缺省) 1 P3.15 输出信号 CLKOUT ($f = f_{MC}$)
FOTL	6	rh	频率输出翻转锁存 每次 FOCNT 下溢时翻转锁存。
FOCNT	[5:0]	rh	频率输出计数器

注: 位域 **FOCNT** 和位 **FOTL** 不可写。这可防止对寄存器 **FOCON** 进行写操作时产生非法的时钟周期, 比如改变了输出频率或停止输出时钟信号。

利用寄存器安全机制, 执行 **EINIT** 后 **FOCON** 被写保护 (见章节 6.3.5)。

注：对所有大于 0 的重装载值 $FORV$ ，输出信号（ $FOSS = 1$ ）在一个 f_{MC} 周期持续时间内为高电平。 $FORV = 0$ ，输出信号与 f_{MC} 相同。

输出频率计算

输出频率可计算如下：

$$f_{OUT} = f_{MC} / ((FORV + 1) \times 2^{(1 - FOSS)})$$

故： $f_{OUTmin} = f_{MC} / 128$ ($FORV = 3F_H$, $FOSS = 0$) ,

$$f_{OUTmax} = f_{MC} / 1$$
 ($FORV = 00_H$, $FOSS = 1$) 。

表 6-6 f_{OUT} 的可选输出频率范围

f_{MC}	f_{OUT} [kHz], $FORV = xx$, $FOSS = 1/0$					$FORV$, $f_{OUT} = 1MHz$	
	00 _H	01 _H	02 _H	3E _H	3F _H	FOSS=0	FOSS=1
4 MHz	4000	2000	1333.33	63.492	62.5	01 _H	03 _H
	2000	1000	666.67	31.746	31.25		
10 MHz	10000	5000	3333.33	158.73	156.25	04 _H	09 _H
	5000	2500	1666.67	79.365	78.125		
12 MHz	12000	6000	4000	190.476	187.5	05 _H	0B _H
	6000	3000	2000	95.238	93.75		
16 MHz	16000	8000	5333.33	253.968	250	07 _H	0F _H
	8000	4000	2666.67	126.984	125		
20 MHz	20000	10000	6666.67	317.46	312.5	09 _H	13 _H
	10000	5000	3333.33	158.73	156.25		
25 MHz	25000	12500	8333.33	396.825	390.625	0B _H (1.04167) 0C _H (0.96154)	18 _H
	12500	6250	4166.67	198.413	195.313		
33 MHz	33000	16500	11000	523.810	515.625	0F _H (1.03125) 10 _H (0.97059)	20 _H
	16500	8250	5500	261.905	257.813		

6.3 中央系统控制功能

大多数控制功能和状态信息是和 XC164CM 的各个外设模块紧密相关的。不过，有些控制功能对整个芯片均有效（而非仅针对某特定模块）。这些功能（包括相关的控制和状态位）是 SCU 的一部分。

SYSCON0

通用系统控制寄存器

ESFR (F1BE_H/DF_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
RTC RST	RTC CM	-	OSC G RED	-	-	-	-	-	-	-	-	-	-	-	-
rwh	rw	-	rw	-	-	-	-	-	-	-	-	-	-	-	-

符号	位序号	读写类型	功能描述
RTCRST	15	rwh	RTC 复位触发 0 无动作 1 RTC 模块复位 ¹⁾ <i>注: RTCRST 被置位后, 经过一个 SCU 时钟周期, RTCRST 返回 0。</i>
RTCCM	14	rw	RTC 时钟模式¹⁾ 0 同步模式: RTC 以系统时钟运行。寄存器可被读写。 1 异步模式 ²⁾ : RTC 以（异步）计数时钟运行。不可进行写操作。
OSCGRED	12	rw	振荡器增益降低控制 0 不降低, 保持原有增益电平 1 降低增益（见 章节 6.2.1 ）

1) RTC 复位之后, 立即进入由位 RTCCM 当前选择的时钟模式。

2) 如果系统时钟低于 $4 \times f_{\text{COUNT}}$, 则要使用异步模式。在休眠模式或掉电模式下, 系统时钟被禁止, 而 RTC 要继续运行, 则工作在异步模式（见 [第 15 章](#)）。

注: SYSCON0 受寄存器安全机制保护（见 [章节 6.3.5](#)）。复位值仅对硬件复位有效。

寄存器 SYSCON1 控制以下功能：

- 系统的主时钟预分频因子
- 空闲/休眠模式下程序 Flash 的状态
- 休眠模式和掉电模式下端口驱动器的状态
- 空闲或休眠模式选择

SYSCON1

系统控制寄存器

ESFR (F1DC_H/EE_H)

复位值:0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	CP SYS	-	-	PF CFG		PD CFG		SLEEP CON	
-	-	-	-	-	-	-	rw	-	-	rw		rw		rw	

符号	位序号	读写类型	功能描述
CPSYS	8	rw	系统时钟预分频因子（见 章节 6.2.2 ） CPU 时钟信号预分频： 0 $f_{CPU} = f_{MC}$ 1 $f_{CPU} = f_{MC}/2$
PFCFG	[5:4]	rw	程序 Flash 配置 ¹⁾ 00 程序 Flash 始终打开（缺省状态） 01 空闲或休眠模式下程序 Flash 关闭 10 保留 11 保留
PDCFG	[3:2]	rw	端口驱动器配置 00 端口驱动器始终打开（缺省状态） 01 空闲或休眠模式下端口驱动器关闭 10 掉电模式下端口驱动器关闭 11 保留
SLEEPCON	[1:0]	rw	休眠模式配置 （执行 IDLE 指令进入该模式） 00 进入普通空闲模式 01 进入休眠模式

符号	位序号	读写类型	功能描述
			10 保留
			11 保留

1) 掉电模式下程序 Flash 始终关闭。

注：SYSCON1 受寄存器安全机制保护（见[章节 6.3.5](#)）。

6.3.1 状态指示

系统状态寄存器 **SYSSTAT** 通过一组标志位指示时钟产生单元的状态、以及最近一次复位的复位源。

SYSSTAT

系统状态标志

mem (F1E4H/--)

复位值: XXXXH

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
OSC LOC K	PLL LOC K	CLK HIX	CLK LOX	OSC STA B	PLL EM	-	-	-	-	-	-	-	HW R	SW R	WDT R
rh	rh	rh	rh	rh	rh	-	-	-	-	-	-	-	rh	rh	rh

符号	位序号	读写类型	功能描述
OSCCLOCK	15	rh	振荡器信号状态位 0 振荡器未锁定 1 振荡器锁定（已计数 2048 个 f _{osc} 时钟周期，因此假定已稳定）
PLLLOCK	14	rh	PLL 信号状态位 0 PLL 未锁相（基频或正在调整） 1 PLL 锁相（稳定输出频率）
CLKHIX	13	rh	超过输入时钟的上限 0 输入时钟频率低于输入时钟的上限 1 输入时钟频率过高
CLKLOX	12	rh	超过输入时钟的下限 0 输入时钟频率高于输入时钟的下限 1 输入时钟频率过低
OSCSTAB	11	rh	振荡器稳定标志 0 振荡器正在启动。 1 振荡器计数器已计数到其上限阈值（2 ¹⁵ 个 f _{osc} 时钟周期）。根据缺省增益值，该标志指示振荡器振幅已达到最大值的 90%。 硬件复位、或当振荡器从关闭状态唤醒之后该

符号	位序号	读写类型	功能描述
			位清零。
PLLEM	10	rh	PLL 紧急模式标志 0 时钟产生未遇到问题 1 时钟产生出错 <i>注：若从休眠模式唤醒后振荡器已锁定，则 PLLEM 被自动清零。否则该位保持置位直到硬件复位。</i>
HWR	2	rh	硬件复位指示标志 0 上一次复位不是硬件复位 1 上一次复位是硬件复位
SWR	1	rh	软件复位指示标志 0 上一次复位不是软件复位 1 上一次复位是软件复位
WDTR	0	rh	看门狗定时器复位指示标志 0 上一次复位不是看门狗定时器复位 1 上一次复位是看门狗定时器复位

注：寄存器 SYSSTAT 的复位值取决于当前有效的状态标志。

6.3.2 复位源指示

寄存器 SYSSTAT 中的复位标志用于指示最近一次复位的触发源。XC164CM 开始运行后，初始化软件会检测这些标志，以确定最近一次的复位是由外部硬件信号（ $\overline{\text{RSTIN}}$ ）触发、还是由软件、或是由看门狗定时器溢出触发。从而可使微控制器系统的初始化和进一步操作适应各自工作环境的需要。例如，在看门狗定时器复位后，可用一个特殊的子程序验证软件的完整性。

各复位指示标志是互斥的。每次复位后，只有一个标志被置位（取决于复位源）。

当输入引脚 $\overline{\text{RSTIN}}$ 采样为低时（信号有效），指示**硬件复位**。

执行指令 SRST 触发复位时，指示**软件复位**。

看门狗定时器溢出触发复位时，指示**看门狗定时器复位**。

6.3.3 外设关闭握手机制

执行软件复位或进入掉电模式时，SCU 请求关闭那些正在运行的、并具有关闭握手机制的外设。外设单元接收到关闭请求后，先完成当前正在执行的操作（若存在），然后响应 SCU 的关闭请求。

这些单元包括：PMU、DMU、EBC、ADC 和程序 Flash。

一旦所有单元响应了关闭请求，关闭握手过程结束。

6.3.4 调试系统控制

调试和仿真电路由两个寄存器控制：

- 仿真控制寄存器 EMUCON 控制基本 OCDS 功能。
- OCE/OCDS 外设挂起使能寄存器 OPSEN 选择由挂起信号中止运行的外设。OPSEN 和寄存器 SYSCON3 的结构相同。

EMUCON

仿真控制寄存器

SFR (FE0AH/05H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	-	OC EN	OCD SIO EN	-
-	-	-	-	-	-	-	-	-	-	-	-	-	rw	rw	-

符号	位序号	读写类型	功能描述
OCEN	2	rw	OCDS/Cerberus 使能 0 OCDS 和 Cerberus 仍处于复位状态 1 OCDS 和 Cerberus 可工作
OCDSIOEN	1	rw	OCDS 断点输入/输出使能 0 禁止 OCDS 断点输入/输出 $\overline{\text{BRKIN}} / \overline{\text{BRKOUT}}$ 1 使能 OCDS 断点输入/输出 $\overline{\text{BRKIN}} / \overline{\text{BRKOUT}}$

注：利用寄存器安全机制，执行 EINIT 指令后，EMUCON 被写保护（见[章节 6.3.5](#)）。

OPSEN

OCE/OCDS 外设挂起使能寄存器

ESFR (FE58_H/2C_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSC 1 SEN	RTC SEN	CAN SEN	-	-	ASC 1 SEN	-	CC6 SEN	CC2 SEN	-	PFM SEN	-	GPT SEN	SSC 0 SEN	ASC 0 SEN	ADC SEN
rw	rw	rw	-	-	rw	-	rw	rw	-	rw	-	rw	rw	rw	rw

符号	位序号	读写类型	功能描述
xxSEN	15...13, 10, 8, 7, 5, 3...0	rw	模块 xx 挂起使能 0 相应模块在挂起时保持有效 1 相应模块在挂起时中止运行

注：利用寄存器安全机制，执行 EINIT 指令后，OPSEN 被写保护（见[章节 6.3.5](#)）。

执行到某断点时，寄存器 OPSEN 选中的片上外设停止运行并进入掉电模式，其行为如同通过寄存器 SYSCON3 将其关闭。这些外设的寄存器此时可被读取，但不能写入；读访问不会触发任何操作（外设已关闭）。

无论何种原因关闭外设（由 SYSCON3 关闭或由 OPSEN 关闭），SYSCON3 始终指示外设的关闭状态。这意味着，执行到某断点处之后，调试器读取寄存器 SYSCON3 的值时，将返回 SYSCON3 和 OPSEN 按位逻辑或的结果。

建议用户将 OPSEN.5 (PFMSEN) 置 0（缺省值）。否则，当执行到某断点时，程序 Flash 不可使用（即无法读取），继续执行程序时 Flash 的内部电压不得不重新升高（即，软件和外设之间的同步丢失）。

6.3.5 寄存器安全机制

XC164CM 中有一些专用寄存器用来控制系统的关键功能和模式。这些寄存器受寄存器安全机制保护，从而可确保重要的系统功能不会被无意修改。

寄存器安全机制控制三种不同的安全等级：

- **写保护模式**（执行 EINIT 指令后进入）。
保护寄存器被锁定、不能进行写访问（只可读）。
- **安全模式**
保护寄存器只能通过特殊的指令序列写入。
- **无保护模式**（复位后进入）
保护无效，寄存器可在任意时刻写入。

注：所选的安全等级适用于 XC164CM 中所有的受保护寄存器（见表 6-8）。

控制安全等级

有两个寄存器用于控制安全等级。安全等级命令寄存器 SCUSCL 接受命令，控制状态机修改安全等级（命令序列由密码保护）。安全等级状态寄存器 SCUSLS（只可读）给出实际的密码、实际的安全等级、以及切换状态机的状态。

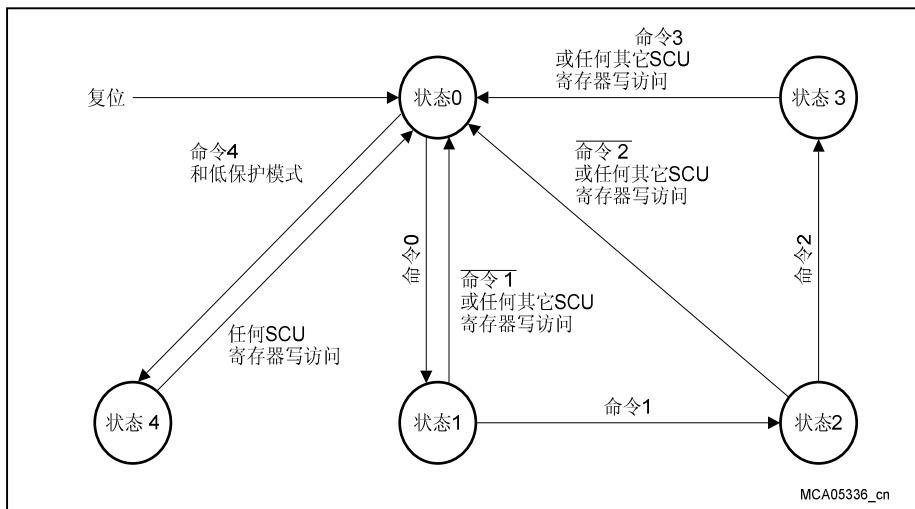


图 6-10 用于安全等级切换的状态机

可使用**两种机制**控制实际的安全等级：

- **修改安全等级**

执行以下命令序列可修改安全等级：

“命令 0-命令 1-命令 2-命令 3”。

该序列可设立新的安全等级和/或新密码。

- **安全模式下的访问**

可用如下方法实现：在写访问之前将“命令 4”写入寄存器 SCUSLC。只有在选择安全模式时，才可实现该快速访问。

对 SCU 的所有寄存器始终可进行读操作，不会影响命令序列。寄存器 SCUSLS 中命令状态机的实际状态始终可被读取。

注：安全模式下写入命令 4 之后，保护机制无效，直到对 SCU 寄存器或 PD 总线上的寄存器进行下一次写访问后保护机制被激活；也就是说，访问该区域之外的寄存器不会重新激活保护机制。

复位后缺省进入无保护状态。只有在执行 EINIT 指令之后才能通过命令序列改变该状态。

SCUSLS

安全等级状态寄存器

ESFR (F0C2H/61H)

复位值: 0000H

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

符号	位序号	读写类型	功能描述
STATE	[15:13]	rh	切换状态机的当前状态 000 等待命令 0 或 命令 4 （缺省状态） 001 等待命令 1 010 等待命令 2 011 等待新的安全等级和密码 100 安全模式下许可的下次访问 101 保留 11X 保留
SL	[12:11]	rh	安全等级 00 无保护模式（复位后的缺省值）

符号	位序号	读写类型	功能描述
			01 安全模式 10 保留 11 写保护模式（执行 EINIT 指令后进入）
PASSWORD	[7:0]	rh	当前安全控制密码 复位后缺省值 = 00 _H

SCUSLC

安全等级命令寄存器

ESFR (F0C0_H/60_H)

复位值: 0000_H

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

符号	位序号	读写类型	功能描述
COMMAND	[15:0]	rw	安全等级控制命令 必须将控制安全等级的命令写入该寄存器 (见 表 6-7)

注：寄存器 SCUSLC 不受安全机制保护，以保证可在任意状态下修改安全等级。

表 6-7 安全等级控制命令

命令	定义	注
命令 0	AAAA _H	-
命令 1	5554 _H	-
命令 2	96 _H <密码取反>	-
命令 3	000 _B <新等级> 000 _B <新密码>	-
命令 4	8E _H <密码取反>	只用于安全模式

注：推荐使用不可分指令序列（atomic sequence）锁定所有命令序列。

程序示例:

```

EXTR #4                      ; 改变安全等级的序列
MOV SCUSLC, #0AAAAH         ; 命令 0
MOV SCUSLC, #05554H         ; 命令 1
MOV SCUSLC, #096FFH         ; 命令 2: 当前密码 = 00H
MOV SCUSLC, #008EDH         ; 命令 3: 等级 = 01, 新密码 = EDH

EXTR #1                      ; 安全模式下的访问序列
MOV SCUSLC, #8E12H          ; 命令 4: 当前密码 = EDH
MOV register, data           ; 命令 4 使能寄存器的写访问
    
```

寄存器安全机制不仅保护 SCU 寄存器，还保护其它模块中的寄存器，这些保护寄存器归纳见**表 6-8**。

表 6-8 受安全机制保护的寄存器

寄存器名称	功能
RSTCON	复位控制
SYSCON0	通用系统控制
SYSCON1	功率管理
PLLCON	时钟产生控制
SYSCON3	外设管理
FOCON	外设管理（CLKOUT/FOUT）
IMBCTR	内部指令存储模块控制
OPSEN	仿真控制
EMUCON	仿真控制
WDTCON	看门狗定时器属性
EXICON	外部中断控制
EXISEL0、EXISEL1	外部中断控制
CPUCON1、CPUCON2	CPU 配置，执行 EINIT 指令后受保护

寄存器名称	功能
TCONCS7	EBC 时序配置（内部 TwinCAN 访问）
FCONCS7	EBC 功能配置（内部 TwinCAN 访问）
ADDRSEL7	EBC 地址窗配置（内部 TwinCAN 访问）

6.4 功率管理

可通过多种机制来降低 XC164CM 的功耗。功耗降低的程度取决于所要实现的系统性能。XC164CM 提供了三种软件控制降低功耗的主要方式：

- 工作在功耗降低模式（空闲、休眠、掉电），使 CPU、端口和控制逻辑无效
- 降低 CPU 频率和系统频率
- 选择需要运行的外设模块（灵活的外设管理）

实际应用（程序员）可根据不同的工作条件选择最佳的功耗节省方式，从而使系统灵活的工作在最大性能、部分性能和待机状态下。

这三种方式可独立使用，从而为各种应用提供了最大的灵活性。

6.4.1 功耗降低模式

XC164CM 提供了三种功耗降低模式，对应不同的功耗等级。通过专用指令进行模式选择：IDLE 指令选择空闲或休眠模式，PWRDN 指令选择掉电模式。

注意：复位后所有功耗降低模式被终止。

空闲模式

空闲模式下，所有被使能的外设（包括看门狗定时器）继续正常工作，只有 CPU 被中止运行。

*注：*由软件禁止的外设在空闲模式下仍然保持禁止状态。

进入空闲模式需要如下条件：执行 IDLE 指令（SYSCON1 中的位域 SLEEPCON 必须为 00_B）、且 IDLE 指令之前的指令已执行完毕。为防止无意进入空闲模式，IDLE 指令为 32 位被保护指令。

空闲模式可被中断请求**终止**，该中断请求来自任何被使能的中断源，其中断使能标志在进入空闲模式前已置位，和全局中断使能位 IEN 的状态无关。

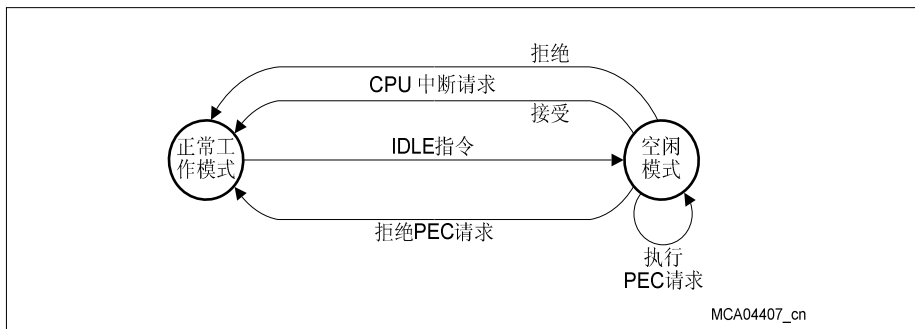


图 6-11 空闲模式和正常工作模式之间的转换

若中断请求源的优先级高于当前 CPU 的优先级，且中断系统被全局使能，则转入执行相应的中断服务程序。中断服务程序执行 RETI（从中断返回）指令后，CPU 继续执行 IDLE 指令之后的程序。反之，若中断请求源的优先级较低、或中断系统被全局禁止，则相应的中断请求不被响应，CPU 立即恢复执行 IDLE 指令之后的程序。

若 PEC 中断请求的优先级高于当前 CPU 的优先级，且中断系统被全局使能，则执行 PEC 数据传送。PEC 数据传送结束后，CPU 保持空闲状态。反之，若 PEC 中断请求的优先级较低、或中断系统被全局禁止，则不能执行 PEC 传送，此时 CPU 不再保持空闲模式，而是恢复执行 IDLE 指令之后的程序。

空闲模式还能通过非屏蔽中断终止，即 $\overline{\text{NMI}}$ 引脚从高电平跳变到低电平。空闲模式被中断请求或 NMI 请求终止后，中断仲裁模块执行一轮优先级排序，以确定最高优先级请求。如果存在 NMI 请求，将进入 NMI 强制中断。

在进入空闲模式之前，若某个中断请求的中断使能标志被置位，则无论 CPU 当前所处的优先级，该中断请求总会终止空闲模式。当检测到 CPU 中断请求时，即使由于当前 CPU 优先级较高、或中断系统被全局禁止（ $\text{IEN} = 0$ ）而不响应该中断，CPU 也不会返回空闲模式。只有当中断系统被全局使能（ $\text{IEN} = 1$ ），且优先级高于当前 CPU 优先级的 PEC 服务被响应并执行后，CPU 才会返回空闲模式。

注：被使能且优先级为 0 的中断请求可终止空闲模式。不过相关的中断向量不会被访问。

看门狗定时器可用于监控空闲模式：若看门狗定时器溢出之前没有中断或 NMI 请求发生，将产生内部看门狗定时器复位。为了防止看门狗定时器在空闲模式下溢出，进入空闲模式之前必须为看门狗定时器设定合理的时间间隔。

休眠模式

休眠模式下，所有内部模块（包括看门狗定时器）的时钟停止。不过，内部 RAM 的内容由 V_{DD} 引脚提供的电压保留。

进入休眠模式需要如下条件：执行 IDLE 指令（SYSCON1 中的位域 SLEEPCON 必须为 01_B），IDLE 指令之前的指令已执行完毕，且所有被使能的中断请求标志目前未被激活。为防止无意进入休眠模式，IDLE 指令为 32 位被保护指令。

休眠模式可通过如下操作**终止**：RTC 中断（若被使能）、外部中断线或复用输入线上的电平跳变（若相应的电平检测被使能）、或 NMI 请求。外部中断电平检测由寄存器 EXICON 控制。唤醒后的操作取决于快速外部中断（触发唤醒的中断）中断使能标志的设置，以及全局中断使能标志 PSW.IEN 的设置。如果唤醒被触发之前，中断使能标志没有被软件置位，则 XC164CM 从休眠模式切换至空闲模式；如果中断使能标志为 1，则 XC164CM 切换至正常工作模式。根据全局中断使能标志以及中断优先级的设置，相应执行中断服务程序或 IDLE 之后的指令，该操作与空闲模式下的操作相同。

注：串口接收线可通过寄存器 EXISELn 与外部中断输入引脚相连。所有外设被停止，因此不能产生中断请求。

休眠模式下的总功耗主要取决于端口驱动中的电流。为了使消耗电流最低，可通过寄存器 SYSCON1 中的控制位将所有引脚的驱动器关闭（引脚切换为高阻态）。若应用要求即使在休眠模式下，也要使一个或多个端口驱动器保持有效，可将其端口驱动器设置为输入。

掉电模式

掉电模式下，CPU 和所有外设都停止工作。内部 RAM 的内容由 V_{DD} 引脚提供的电压保留。

进入掉电模式需要如下条件：执行 PWRDN 指令，PWRDN 指令之前的指令执行完毕， $\overline{\text{NMI}}$ （非屏蔽中断）引脚在执行 PWRDN 指令时被外部拉低。为防止无意进入掉电模式，PWRDN 指令为 32 位被保护指令，并由 $\overline{\text{NMI}}$ 信号附加验证。如果此时 $\overline{\text{NMI}}$ 不为低电平，则不会进入掉电模式，继续执行原有程序。

掉电模式只能通过硬件复位**终止**（不能由内部复位终止）。

掉电模式下的总功耗主要取决于端口驱动中的电流。为了使消耗电流最低，可通过寄存器 SYSCON1 中的控制位将所有引脚的驱动器关闭（引脚切换为高阻态）。若实际应用要求即使在掉电模式下，也要使一个或多个端口驱动器保持有效，可将其端口驱动器设置为输入。

6.4.2 降低时钟频率

XC164CM 的功耗与系统频率线性相关。时钟控制的方法见 [章节 6.2，时钟产生](#)。

6.4.3 灵活的外设管理

XC164CM 的功耗还取决于正在工作的逻辑模块的数目。外设管理可关闭那些在特定系统状态（如特定接口模式或待机状态）下不需要的片上外设，从而减少时钟驱动电路。所有工作中的模块功能不受影响。

注：对被禁止的外设的寄存器进行读操作会返回有效的寄存器内容，对该寄存器进行写操作将无效。

读操作不会触发被禁止外设的任何操作。

若外设被禁止，其相关输出引脚保持被禁止时刻的状态。

通过软件设置寄存器 **SYSCON3** 来灵活的管理外设，寄存器中的各控制位和片上外设模块一一对应。

写入 SYSCON3 请求禁止/激活各外设模块。对寄存器 **SYSCON3** 进行写操作时，请确保所有未定义的位置 1。

读取 SYSCON3 根据关机握手机制（见 [章节 6.3.3](#)）返回外设的真实状态。

SYSCON3

系统控制寄存器 3

ESFR (F1D4H/EAH)

复位值:9FD0_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
SSC 1 DIS	RTC DIS	CAN DIS	-	-	ASC 1 DIS	-	CC6 DIS	CC2 DIS	-	PFM DIS	-	GPT DIS	SSC 0 DIS	ASC 0 DIS	ADC DIS
rw	rw	rw	-	-	rw	-	rw	rw	-	rw	-	rw	rw	rw	rw

符号	位序号	读写类型	功能描述
SSC1DIS	15	rw	同步串行通道 SSC1
RTCDIS	14	rw	实时时钟
CANDIS	13	rw	片上 CAN 模块
ASC1DIS	10	rw	USART ASC1
CC6DIS	8	rw	CAPCOM 单元 6
CC2DIS	7	rw	CAPCOM 单元 2
PFMDIS	5	rw	程序 Flash 模块 ¹⁾

符号	位序号	读写类型	功能描述
GPTDIS	3	rw	通用定时器模块
SSC0DIS	2	rw	同步串行通道 SSC0
ASC0DIS	1	rw	USART ASC0
ADCDIS	0	rw	模数转换器

1) 若程序 Flash 在正常工作模式下被禁止（PMFDIS 置位），下次访问程序 Flash 时，将产生强制中断断代码 1E9B_H，产生“程序存储器访问错误”的强制中断。

寄存器 SYSCON3 的复位值为 9FD0_H，对应程序存储器和一组基本的外设被使能。XC164CM 的其它外设必须在初始化期间被使能后才可使用。

对于某些 XC164CM 衍生产品，复位值 9FD0_H可能会选中一些该产品未集成的外设（见表 1-1）。为了保证统一的软件概念，建议用户在初始化阶段向 SYSCON3 中写入有效值。

注：寄存器 SYSCON3 的外设禁止位分配和器件型号紧密相关，不同产品 SYSCON3 的禁止位分配可能有差别。

利用寄存器安全机制，在执行 EINIT 指令之后，SYSCON3 被写保护（见章节 6.3.5）。

6.5 看门狗定时器（WDT）

为了使系统能够从软件或硬件故障中恢复，XC164CM 提供了看门狗定时器。如果定时器在溢出之前没有被软件及时服务，将会启动一个内部复位序列。如果看门狗定时器被使能，并且在溢出之前规律地被软件服务，那么看门狗定时器能够监视程序的执行，只有当程序运行不正常时才会溢出。如果由硬件故障导致软件错误，看门狗定时器也会溢出。看门狗定时器可防止控制器长时间（由用户规定最长故障时间）处于故障状态。

看门狗定时器有两个寄存器：

- 只读定时器寄存器，包含当前计数值
- 控制寄存器，用于初始化和复位源检测

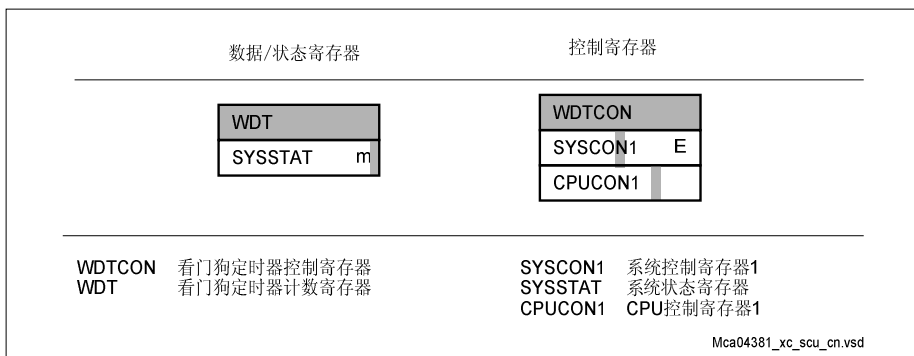


图 6-12 看门狗定时器的相关 SFR 和端口引脚

看门狗定时器是一个 16 位递增计数器，计数时钟是经过预分频处理的系统时钟（ f_{SYS} ）。预分频因子可选择：

- 2 分频（WDTIN = 00_B），或
- 4 分频（WDTIN = 10_B），或
- 128 分频（WDTIN = 01_B），或
- 256 分频（WDTIN = 11_B）

16 位看门狗定时器由两个 8 位定时器级连构成（见图 6-13）。看门狗定时器的高 8 位可在服务看门狗时被预置成用户设定的值，用来改变看门狗的失效时间；低 8 位在每次服务看门狗后被复位。

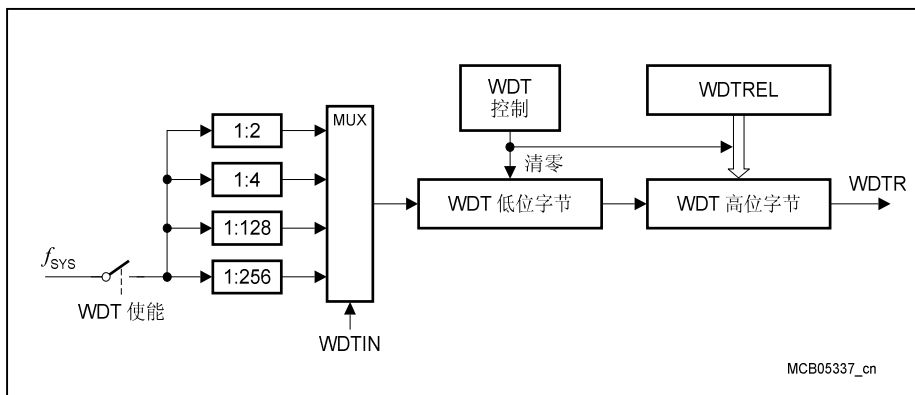


图 6-13 看门狗定时器框图

看门狗定时器的操作

看门狗定时器的当前计数值存放在看门狗定时器寄存器 WDT 中，该寄存器是不可位寻址的只读寄存器。看门狗定时器的操作由可位寻址的看门狗定时器控制寄存器 WDTCON 控制，该寄存器定义定时器高位字节的重载值，选择输入时钟的预分频因子。

任何复位后（注释情况除外），看门狗定时器被使能，并以缺省频率 $f_{\text{WDT}} = f_{\text{SYS}}/2$ 从 0000H 开始递增计数。可通过设定预分频因子（位域 WDTIN）改变输入频率（ $f_{\text{WDT}} = f_{\text{SYS}}/4、128、256$ ）。

看门狗定时器可用指令 DISWDT（禁止看门狗定时器）禁止。DISWDT 指令为 32 位被保护指令。

兼容 WDT 模式下，指令 DISWDT 只能在复位之后、执行 EINIT 或 SRVWDT 指令（这两条指令会禁止 DISWDT 的执行）之前执行。一旦看门狗定时器被禁止，WDT 只能通过复位被使能。

增强 WDT 模式下，看门狗定时器可在任意时刻被禁止和使能（和 EINIT 指令无关），分别通过执行指令 DISWDT 和 ENWDT 进行控制。指令 ENWDT 为 32 位被保护指令。如果看门狗定时器由指令 ENWDT 重新使能，则它被隐性地服务。

WDT 的基本控制模式（兼容/增强）由寄存器 CPUCON1 中的位 WDTCTL 选择。

注：若硬件复位激活引导程序加载模式，看门狗定时器将被禁止。加载软件开始运行后，WDT 被使能。

如果看门狗定时器没有被指令 DISWDT 禁止，那么，即使处于空闲模式，该定时器也将继续递增计数。如果定时器计数到 FFFFH 后仍未被服务，看门狗定时器会溢出，并触发内部复位。此时寄存器 SYSSTAT 中的看门狗定时器复位指示标志（WDTR）将被置位。

注意：看门狗定时器复位不受任何限制。当前所有的数据/代码访问被立刻中止。

为了防止看门狗定时器溢出，它必须由用户软件周期性的服务。服务看门狗定时器的三种操作如下：

- 执行 32 位被保护指令 SRVWDT
- 对寄存器 WDTCON 进行写操作
- 执行指令 ENWDT（增强 WDT 模式）

服务看门狗定时器的操作包括：清零看门狗定时器寄存器 WDT 的低位字节，将位域 WDTREL 中的预设值载入高位字节，WDTREL 是寄存器 WDTCON 的高位字节。服务之后，看门狗定时器从数值 $(\text{WDTREL} \times 2^8)$ 开始继续递增计数。

指令 SRVWDT 的编码方式使错误服务看门狗定时器的几率最小（例如从错误地址读取和执行一个位序列）。如果指令 SRVWDT 与被保护指令的格式不匹配，则进入保护错误强制中断，而不会去执行指令。

WDTCON

WDT 控制寄存器

SFR (FFAE_H/D7_H)

复位值: 0000_H

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

符号	位序号	读写类型	功能描述
WDTREL	[15:8]	rw	看门狗定时器重载值 (用于 WDT 的高位字节)
WDTIN	[1:0]	rw	看门狗定时器输入频率选择 00 $f_{WDT} = f_{SYS}/2$ 01 $f_{WDT} = f_{SYS}/128$ 10 $f_{WDT} = f_{SYS}/4$ 11 $f_{WDT} = f_{SYS}/256$

注: WDTCON 受寄存器安全机制保护 (见 [章节 6.3.5](#))。

看门狗定时器的溢出时间周期可由两种方式编程设定:

- 看门狗定时器的**输入频率**可由寄存器 WDTCON 中的位域 WDTIN 选择 (控制预分频因子), 输入频率可分别为 $f_{SYS}/2$ 、 $f_{SYS}/4$ 、 $f_{SYS}/128$ 或 $f_{SYS}/256$ 。
- WDT 的高位字节**重载值** WDTREL 可通过寄存器 WDTCON 编程设定。

从服务看门狗定时器到下一次溢出之间的周期 P_{WDT} 计算如下:

$$P_{WDT} = \frac{2^{(1 + \langle WDTIN.1 \rangle + \langle WDTIN.0 \rangle \times 6)} \times (2^{16} - \langle WDTREL \rangle \times 2^8)}{f_{SYS}} \quad (6.2)$$

表 6-9 列出在特定的系统时钟下，看门狗定时器可能的周期值（取决于预分频位域 WDTIN 的设置）。

表 6-9 看门狗周期取值范围

系统时钟 f_{SYS}	预分频		WDTREL 中的重载值		
	WDTIN	f_{WDT}	FF _H	7F _H	00 _H
10 MHz	00 _B	$f_{\text{SYS}} / 2$	51.20 μs	6.61 ms	13.11 ms
	10 _B	$f_{\text{SYS}} / 4$	102.4 μs	13.21 ms	26.21 ms
	01 _B	$f_{\text{SYS}} / 128$	3.28 ms	422.7 ms	838.9 ms
	11 _B	$f_{\text{SYS}} / 256$	6.55 ms	845.4 ms	1678 ms
20 MHz	00 _B	$f_{\text{SYS}} / 2$	25.60 μs	3.30 ms	6.55 ms
	10 _B	$f_{\text{SYS}} / 4$	51.20 μs	6.61 ms	13.11 ms
	01 _B	$f_{\text{SYS}} / 128$	1.64 ms	211.4 ms	419.4 ms
	11 _B	$f_{\text{SYS}} / 256$	3.28 ms	422.7 ms	838.9 ms
30 MHz	00 _B	$f_{\text{SYS}} / 2$	17.07 μs	2.20 ms	4.37 ms
	10 _B	$f_{\text{SYS}} / 4$	34.13 μs	4.40 ms	8.74 ms
	01 _B	$f_{\text{SYS}} / 128$	1.09 ms	140.1 ms	279.6 ms
	11 _B	$f_{\text{SYS}} / 256$	2.19 ms	281.8 ms	559.2 ms
40 MHz	00 _B	$f_{\text{SYS}} / 2$	12.80 μs	1.65 ms	3.28 ms
	10 _B	$f_{\text{SYS}} / 4$	25.60 μs	3.30 ms	6.55 ms
	01 _B	$f_{\text{SYS}} / 128$	0.82 ms	105.7 ms	209.7 ms
	11 _B	$f_{\text{SYS}} / 256$	1.64 ms	211.4 ms	419.4 ms

注：建议在每次服务看门狗之前重新设定寄存器 WDTCON，特别当寄存器安全机制被禁止，或软件需要改变看门狗周期时。

6.6 ID 控制模块

为了识别最重要的芯片参数，XC164CM 定义了一组寄存器，提供芯片制造商、芯片类型及芯片属性等信息。这些 ID 寄存器可用于自动测试选择和未知芯片的识别。

注：00'F070_H ... 00'F07E_H 区域内未定义的位置被保留，用来提供未来产品的 ID 特性。

IDMANUF

制造商 ID 寄存器

ESFR (F07E_H/3F_H)

复位值: 1820_H

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

符号	位序号	读写类型	功能描述
MANUF	[15:5]	r	制造商 JEDEC 规范制造商代码。 0C1 _H 英飞凌科技有限公司
MANSEC	[4:0]	r	制造商内部部门 00 _H 标准微控制器

IDCHIP

芯片 ID 寄存器

ESFR (F07C_H/3E_H)

复位值: 23XX_H

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

符号	位序号	读写类型	功能描述
CHIPID	[15:8]	r	器件 ID 识别器件名称（参见列表）。
Revision	[7:0]	r	器件版本编号 识别器件版本号。

IDMEM

程序存储器 ID 寄存器

ESFR (F07A_H/3D_H)

复位值: 3010_H

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

符号	位序号	读写类型	功能描述
Type	[15:12]	r	片上程序存储器类型 识别片上存储器类型: 0 _H 无片上程序存储器 1 _H 掩膜可编程 ROM 2 _H EEPROM 存储器 3 _H Flash 存储器 4 _H OTP 存储器
Size	[11:0]	r	片上程序存储器容量 以 4KB 为单位的存储器容量, 即 存储器大小 = <Size> × 4 KB

IDPROG

编程电压 ID 寄存器

ESFR (F078_H/3C_H)

复位值: 4040_H

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

符号	位序号	读写类型	功能描述
PROGVPP	[15:8]	r	编程 V_{PP} 电压 ¹⁾ 用于编程或擦除片上程序存储器的特殊电源电压 (若有需要)。 公式: $V_{PP} = 20 \times \text{PROGVPP} / 256 \text{ [V]}^2)$
PROGVDD	[7:0]	r	编程 V_{DD} 电压 ¹⁾ 用于编程或擦除片上程序存储器的标准电源

符号	位序号	读写类型	功能描述
			电压。 公式: $V_{DD} = 20 \times \langle \text{PROGVDD} \rangle / 256 \text{ [V]}$

- 1) ROM 系列产品不可编程。因此寄存器 IDPROG 读取返回值为 0000_H。
- 2) XC164CM 无需特殊编程电压， PROGVPP = PROGVDD。

7 并行端口

本章介绍 XC164CM 并行端口的具体实现，包括各端口相关寄存器的定义、各端口引脚复用功能的分配与控制、以及各端口引脚的配置。

XC164CM 的 IO 线由 9 个输入/输出端口和 1 个输入端口组成。

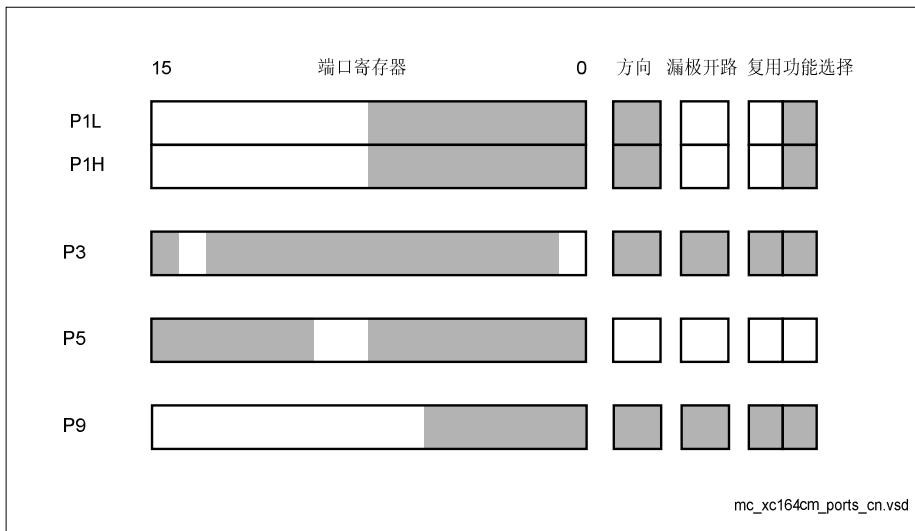


图 7-1 XC164CM 端口总览

7.1 输入阈值控制

XC164CM 的标准输入根据 TTL 电平确定输入信号的状态。为了接收并识别高噪声信号，某些特定端口的引脚可选择类 CMOS 输入阈值代替标准 TTL 阈值。这些特殊阈值高于 TTL 阈值，其电压滞环特性可防止输入信号的电平在接近 TTL 阈值时发生翻转。

端口输入控制寄存器 PICON 为指定端口的每个字节选择输入阈值。每个 8 位端口各由一位控制，每个 16 位端口各由两位控制。

PICON

端口输入控制寄存器 ESFR (F1C4H/E2H) 复位值: 0000H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-						-	-	P9 LIN	-	-	-	P3 HIN	P3 LIN	-	-
-						-	-	rw	-	-	-	rw	rw	-	-

符号	位序号	读写类型	功能描述
PxHIN	3	rw	端口 x 高位字节输入电平选择 0 Px[15-8] 引脚采用标准 TTL 输入电平 1 Px[15-8] 引脚采用特殊阈值输入电平
PxLIN	2, 7	rw	端口 x 低位字节输入电平选择 0 Px[7-0] 引脚采用标准 TTL 输入电平 1 Px[7-0] 引脚采用特殊阈值输入电平

每个引脚的方向和输出方式都可单独设置，和输入阈值的设定无关。输入电压滞环为高噪声信号或缓慢变化的外部信号提供了稳定输入。

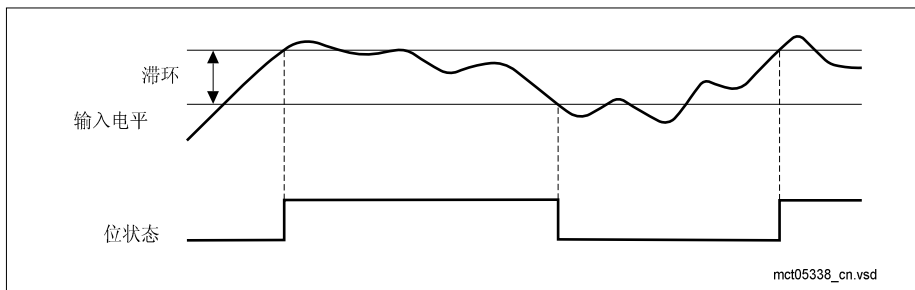


图 7-2 特殊输入阈值的电压滞环

7.2 输出驱动器控制

将引脚切换到输出模式 $DPx.y = "1"$ ，即可激活相应引脚的输出驱动器。驱动至输出引脚的值由端口输出锁存器或相关的复用功能（如地址、外设 IO 等）决定。用户软件可通过以下机制控制输出驱动器的特性：

- **漏极开路模式：**上拉晶体管始终关断。只能有效输出 0 值，需外加上拉电阻。
- **驱动器特性：**可选择驱动能力。
- **边沿特性：**可选择输出信号的上升/下降时间。

漏极开路模式

XC164CM 的某些端口提供漏极开路控制，将端口引脚的输出驱动器从推挽配置切换到漏极开路配置。推挽模式下，端口输出驱动器配有上拉和下拉晶体管，从而可将输出有效驱动至高电平或低电平；漏极开路模式下，上拉晶体管始终被关断，输出驱动器只能将 IO 口有效驱动至低电平。在端口锁存器中写入“1”时，下拉晶体管关闭，输出呈高阻态，此时高电平必须由外部上拉电路产生。利用该特性，可将多条端口线“线与”连接，从而可节省外部连接逻辑电路，也无需额外的软件开销来使能/禁用输出信号。

该特性由漏极开路控制寄存器 $ODPx$ 控制。这些寄存器可按位对每条端口线分别进行漏极开路模式选择。

若控制位 $ODPx.y$ 为“0”（复位缺省值），相应的输出驱动器为推挽模式；若 $ODPx.y$ 为“1”，则选择漏极开路配置。请注意所有 $ODPx$ 寄存器位于 $ESFR$ 区。

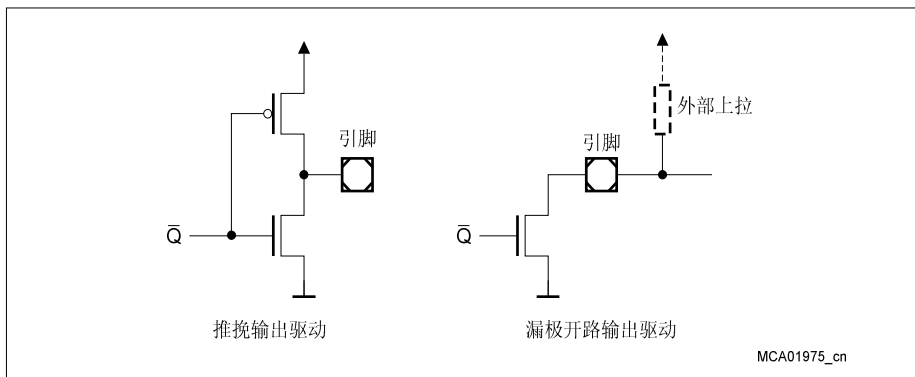


图 7-3 推挽模式和漏极开路模式下的输出驱动器

驱动器特性

该特性定义了驱动器的一般驱动能力，或者在达到目标输出电平后是否减小驱动能力。减小驱动能力可增加输出内阻，减弱由输出线引入/引出的噪声干扰。但在驱动 LED 或功率晶体管时，可能仍需稳定的大驱动电流。

XC164CM 引脚的可控输出驱动器的每个方向（推、挽）都有三种大小不同的晶体管（强、中、弱）。激活/关闭这些晶体管的时间长短决定了端口驱动器的输出特性。

驱动器的驱动能力可灵活选择，从而满足不同应用的需求。

强驱动模式下，中电流和强电流晶体管被激活。该模式下，即使已达到目标电平，驱动器仍提供最大输出电流。

中驱动模式下，只有中电流晶体管被激活，其它晶体管关闭。

弱驱动模式下，只有弱电流晶体管被激活、其它晶体管关闭。弱驱动模式下，由于电流峰值小，电平变化较为平缓（可减小噪声敏感度），但是电平跳变的时间增长（慢沿），跳变时间与容性负载有关。

边沿特性

边沿特性定义了相关输出的上升/下降时间，即输出跳变时间。平缓边沿降低了改变外部容性负载上的电压时的峰值电流。但对于总线接口，可能仍需要陡直边沿（快速变化）。边沿特性作用于前级驱动器，进而控制最后的驱动级输出。

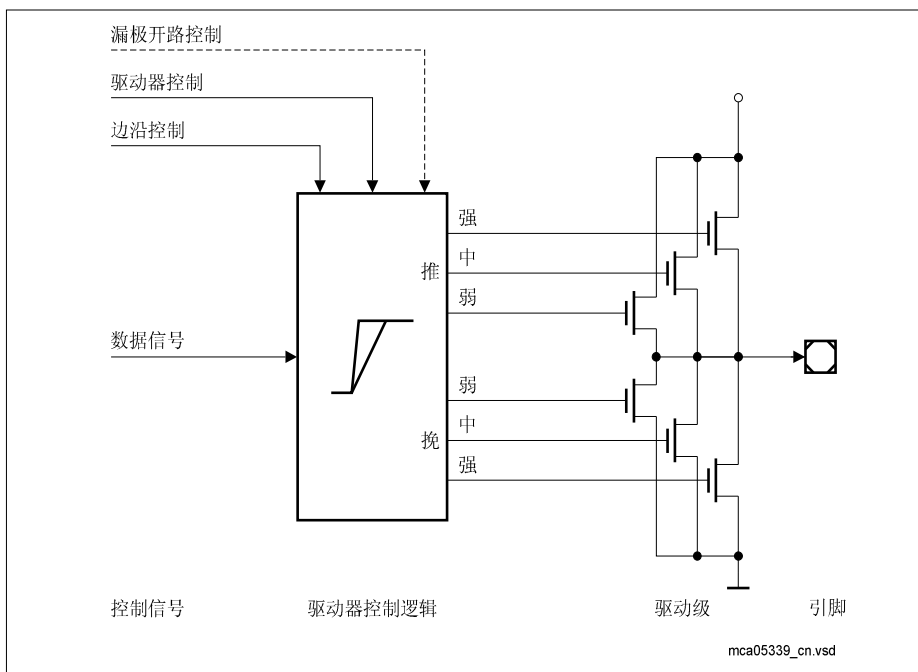


图 7-4 带有边沿控制的三级输出驱动器结构

注：漏极开路模式下，输出引脚的上拉（推）晶体管始终关断。

图 7-4 仅表示输出驱动器的功能结构，并非真正的实现结构。

端口输出控制寄存器 POCONx 提供相应的控制位。4 位控制域用来配置输出驱动能力和边沿特性。一个字端口占用四个 4 位控制域，一个字节端口占用两个 4 位控制域，每个 4 位控制域控制 4 个端口引脚。**表 7-1** 列出定义的 POCON 寄存器、控制位域及端口引脚的分配。

POCON*

端口输出控制寄存器*

ESFR (F0xx_H/yy_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
PDM3N				PDM2N				PDM1N				PDM0N			
rw				rw				rw				rw			

符号	位序号	读写类型	功能描述
PDMxN	[3:0], X = 0 [7:4], X = 1 [11:8], X = 2 [15:12], X = 3	rw	端口驱动模式，半字节 x 设定值，驱动能力¹⁾，边沿形状²⁾ 0000 强驱动，陡边沿模式 0001 强驱动，中等坡度边沿模式 0010 强驱动，平缓边沿模式 0011 弱驱动，标准边沿 ³⁾ 0100 中驱动，标准边沿 ³⁾ 0101 保留，不要使用！ 0110 保留，不要使用！ 0111 保留，不要使用！ 1xxx 保留，不要使用！

1) 定义相应驱动器能向外部电路提供的电流。

2) 定义跳变到新输出电平的切换特性。产生边沿（即输出电平改变）时，对通过驱动器的峰值电流也有影响。

3) 该驱动级别下，不能选择其它边沿形状。

表 7-1 端口输出控制寄存器分配

控制寄存器	地址	控制引脚（由 POCONx.y-z 控制） ¹⁾				注
		.15-12	.11- 8	.7- 4	.3- 0	
POCON9	F094 _H /4A _H	---	---	P9.5-4	P9.3-0	-
POCON3	F08A _H /45 _H	P3.15, P3.13	P3.11-8	P3.7-4	P3.3-1	-
POCON1H	F086 _H /43 _H	---	---	P1H.5-4	P1H.3-0	-
POCON1L	F084 _H /42 _H	---	---	P1L.7-4	P1L.3-0	-

1) x 表示端口号，y-z 表示位域范围。

7.3 端口复用功能

为了使系统最大限度地适用于不同应用中特定的 IO 需求，端口具有可编程复用输入或输出功能。

使用引脚的**复用输出功能**时，引脚方向必须设置为输出（DPx.y = "1"），那些复位后可直接使用、被自动配置的引脚除外。否则，引脚保持高阻态，不受复用输出功能的影响。由软件控制的端口的输出功能通过一或两个特定 ALTSELnPx 寄存器进行选择（n = 0 或 1）。

使用引脚的**复用输入功能**时，引脚方向必须设置为输入（DPx.y = "0"），此时外部器件驱动引脚。复位后，引脚方向缺省为输入。复用输入功能可用于部分外设和外部中断输入。外设的复用输入通过相应的 PISEL 寄存器（如 CAN_PISEL 寄存器）选择。复用外部中断输入通过 SCU 单元中的 EXISEL0 和 EXISEL1 寄存器选择。

所有未用作复用功能的端口线可用作通用 IO。端口引脚用作通用输出口时，在使能输出驱动器之前，先将输出值写入端口锁存，从而避免输出引脚上发生不期望的跳变。该操作对单个引脚和一组引脚均适用。这种情况下，读取这个端口将会返回该端口输出锁存器的值。这一特性可用于测试用途：向端口输出锁存器写数据，然后读取，就可软件触发输入功能。

7.4 P1 口

P1 口由 6 位端口 P1H 和 8 位端口 P1L 组成，分别代表 P1 口的高位字节和低位字节。P1H 和 P1L 可被分别写入（例如通过 PEC 传送）、互不影响。

P1 口用作通用 IO 口时，每条线的方向可通过相应的方向寄存器 DP1H 和 DP1L 进行设置。

P1L

P1 口低位字节寄存器 **SFR (FF04H/82H)** **复位值: 0000H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-								P7	P6	P5	P4	P3	P2	P1	P0
-								rwh	rw	rw	rw	rw	rw	rw	rw

P1H

P1 口高位字节寄存器 **SFR (FF06H/83H)** **复位值: 0000H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-										P5	P4	P3	P2	P1	P0
-										rwh	rwh	rw	rw	rw	rwh

符号	位序号	读写类型	功能描述
P1X.y	[7:0]	rw(h)	P1H 或 P1L 数据寄存器的位 y

注：位 P1L.7, P1H.0, P1H.4-5 在用作 CAPCOM2 输出时，均被位保护。

DP1L

P1L 方向控制寄存器																ESFR (F104 _H /82 _H)				复位值: 0000 _H			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
-								P7	P6	P5	P4	P3	P2	P1	P0								
-								rw	rw	rw	rw	rw	rw	rw	rw								

DP1H

P1H 方向控制寄存器																ESFR (F106 _H /83 _H)				复位值: 0000 _H			
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0								
-										P5	P4	P3	P2	P1	P0								
-										rw	rw	rw	rw	rw	rw								

符号	位序号	读写类型	功能描述
DP1X.y	[7:0]	rw	P1H 或 P1L 方向控制寄存器的位 y 0 端口线 P1X.y 为输入（高阻） 1 端口线 P1X.y 为输出

CAPCOM2 和 SSC1 的复用功能通过寄存器 ALTSEL0P1L 和 ALTSEL0P1H 选择。

ALTSEL0P1L

P1L 复用功能选择寄存器 0																ESFR (F130 _H /98 _H)				复位值: 0000 _H			
15	14	13	12	11	10	9	8	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	位序号	读写类型	功能描述
ALTSEL0P1L.y	[7:0]	rw	P1L 复用功能选择寄存器 0 的位 y 0 相关外设输出未选作引脚的复用功能 1 相关外设输出被选作引脚的复用功能

ALTSEL0P1H

P1H 复用功能选择寄存器 0 ESFR (F120_H/90_H) 复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-										P5	P4	P3	P2	P1	P0
-										rw	rw	rw	rw	rw	rw

符号	位序号	读写类型	功能描述
ALTSEL0 P1H.y	[5:0]	rw	P1H 复用功能选择寄存器 0 的位 y 0 相关外设输出未选作引脚的复用功能 1 相关外设输出被选作引脚的复用功能

P1 口复用功能

P1 口 IO 及复用功能如图 7-5 所示。

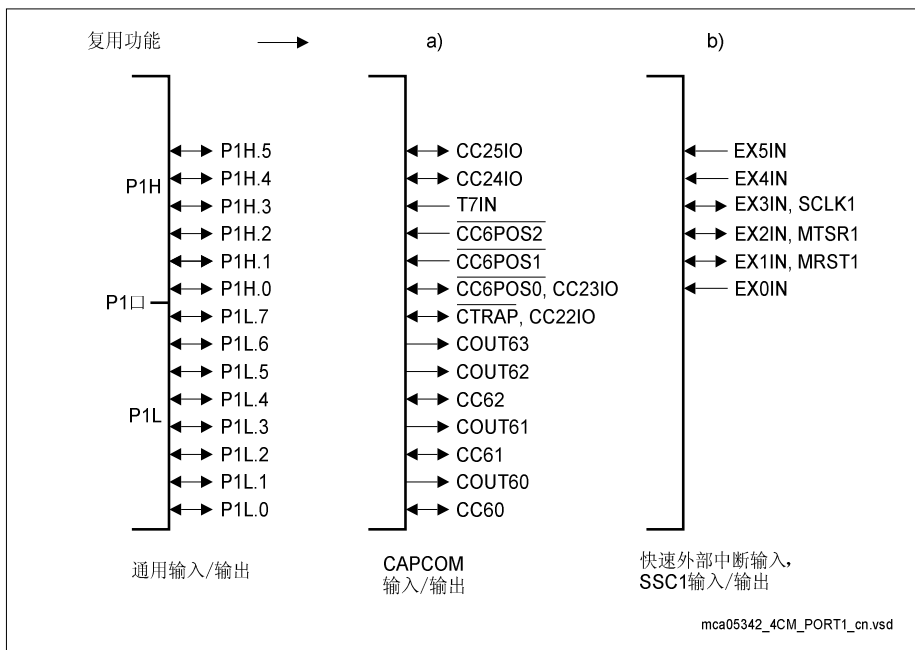


图 7-5 P1 口 IO 及复用功能

表 7-2 给出 P1 口各引脚的功能设置。

注：此处列出的比较输出信号来自 CAPCOM2 单元的 OUT 寄存器。若 CAPCOM 单元直接控制端口锁存器，则多路输出选择器必须选择通用输出。

表 7-2 P1 口功能

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
P1L.0	通用输入	P1L.0	ALTSEL0P1L. P0=0	DP1L.P0=0
	通用输出			DP1L.P0=1
	CC60I 捕获输入	CAPCOM6	-	DP1L.P0=0
	CC60O 比较输出	CAPCOM6	ALTSEL0P1L. P0=1	DP1L.P0=1
P1L.1	通用输入	P1L.1	ALTSEL0P1L. P1=0	DP1L.P1=0
	通用输出			DP1L.P1=1
	COUT60 比较输出	CAPCOM6	ALTSEL0P1L. P1=1	DP1L.P1=1
P1L.2	通用输入	P1L.2	ALTSEL0P1L. P2=0	DP1L.P2=0
	通用输出			DP1L.P2=1
	CC61I 捕获输入	CAPCOM6	-	DP1L.P2=0
	CC61O 比较输出	CAPCOM6	ALTSEL0P1L. P2=1	DP1L.P2=1
P1L.3	通用输入	P1L.3	ALTSEL0P1L. P3=0	DP1L.P3=0
	通用输出			DP1L.P3=1
	COUT61 比较输出	CAPCOM6	ALTSEL0P1L. P3=1	DP1L.P3=1
P1L.4	通用输入	P1L.4	ALTSEL0P1L. P4=0	DP1L.P4=0
	通用输出			DP1L.P4=1
	CC62I 捕获输入	CAPCOM6	-	DP1L.P4=0
	CC62O 比较输出	CAPCOM6	ALTSEL0P1L. P4=1	DP1L.P4=1
P1L.5	通用输入	P1L.5	ALTSEL0P1L. P5=0	DP1L.P5=0
	通用输出			DP1L.P5=1
	COUT62 比较输出	CAPCOM6	ALTSEL0P1L.	DP1L.P5=1

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
			P5=1	
P1L.6	通用输入	P1L.6	ALTSEL0P1L. P6=0	DP1L.P6=0
	通用输出			DP1L.P6=1
	COUT63 比较输出	CAPCOM6	ALTSEL0P1L. P6=1	DP1L.P6=1
P1L.7	通用输入	P1L.7	ALTSEL0P1L. P7=0	DP1L.P7=0
	通用输出			DP1L.P7=1
	CC22I 捕获输入	CAPCOM2	-	DP1L.P7=0
	CC22O 比较输出		ALTSEL0P1L. P7=1	DP1L.P7=1
	$\overline{\text{CTRAP}}$ 强制中断输入	CAPCOM6	-	DP1L.P7=0
P1H.0	通用输入	P1H.0	ALTSEL0P1H. P0=0	DP1H.P0=0
	通用输出			DP1H.P0=1
	CC23I 捕获输入	CAPCOM2	-	DP1H.P0=0
	CC23O 比较输出		ALTSEL0P1H. P0=1	DP1H.P0=1
	$\overline{\text{CC6POS0}}$ 位置 0 输入	CAPCOM6	-	DP1H.P0=0
	快速外部中断输入 EX0IN	-	-	DP1H.P0=0
P1H.1	通用输入	P1H.1	ALTSEL0P1H. P1=0	DP1H.P1=0
	通用输出			DP1H.P1=1
	SSC1 主机接收输入 MRST1	SSC1	-	DP1H.P1=0
	SSC1 从机发送输出 MRST1		ALTSEL0P1H. P1=1	DP1H.P1=1
	$\overline{\text{CC6POS1}}$ 位置 1 输入	CAPCOM6	-	DP1H.P1=0
	快速外部中断输入	-	-	DP1H.P1=0

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
	EX1IN			
P1H.2	通用输入	P1H.2	ALTSEL0P1H. P2=0	DP1H.P2=0
	通用输出			DP1H.P2=1
	SSC1 从机接收输入 MTSR1	SSC1	-	DP1H.P2=0
	SSC1 主机发送输出 MTSR1		ALTSEL0P1H. P2=1	DP1H.P2=1
	$\overline{\text{CC6POS2}}$ 位置 2 输入	CAPCOM6	-	DP1H.P2=0
	快速外部中断输入 EX2IN	-	-	DP1H.P2=0
P1H.3	通用输入	P1H.3	ALTSEL0P1H. P3=0	DP1H.P3=0
	通用输出			DP1H.P3=1
	SSC1 时钟输入 SCLK1	SSC1	-	DP1H.P3=0
	SSC1 时钟输出 SCLK1		ALTSEL0P1H. P3=1	DP1H.P3=1
	快速外部中断输入 EX3IN	-	-	DP1H.P3=0
P1H.x (x=5-4)	通用输入	P1H.x	ALTSEL0P1H. Px=0	DP1H.Px=0
	通用输出			DP1H.Px=1
	CC25-24 捕获输入	CAPCOM2	-	DP1H.Px=0
	CC25-24 比较输出		ALTSEL0P1H. Px=1	DP1H.Px=1
	快速外部中断输入 EXxIN	-	-	DP1H.Px=0

外部中断和输入引脚的连接关系详见表 5-13。

以下各图表示 P1 口各引脚的不同配置。

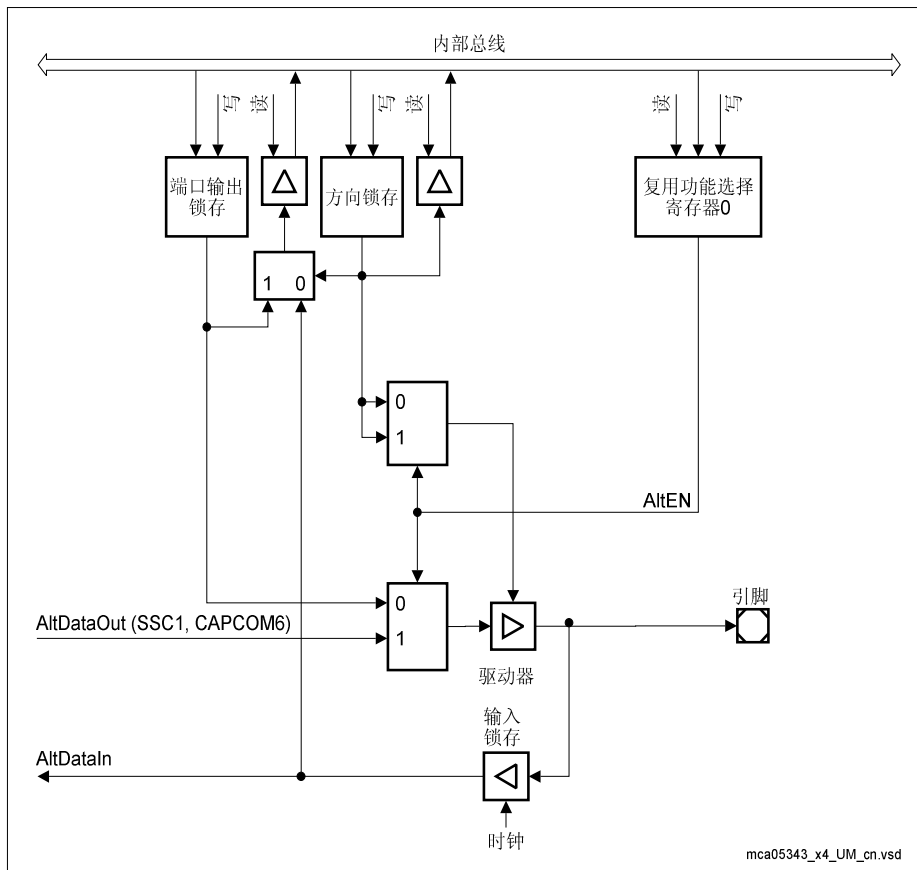


图 7-6 P1L.0-P1L.6、P1H.1-P1H.3 端口配置

表 7-3 P1L.0- P1L.6、P1H.1- P1H.3 复用功能控制

引脚	控制线		寄存器		功能
	AltEN	AltDIR	DP1L/ DP1H	ALTSEL0 P1L/P1H	
P1L.y (y=0-6)	0	-	0 或 1	0	通用 IO
P1H.x (x=1-3)	-		0	-	SSC1, CAPCOM6 捕获输入
	1		1	1	SSC1, CAPCOM6 比较输出

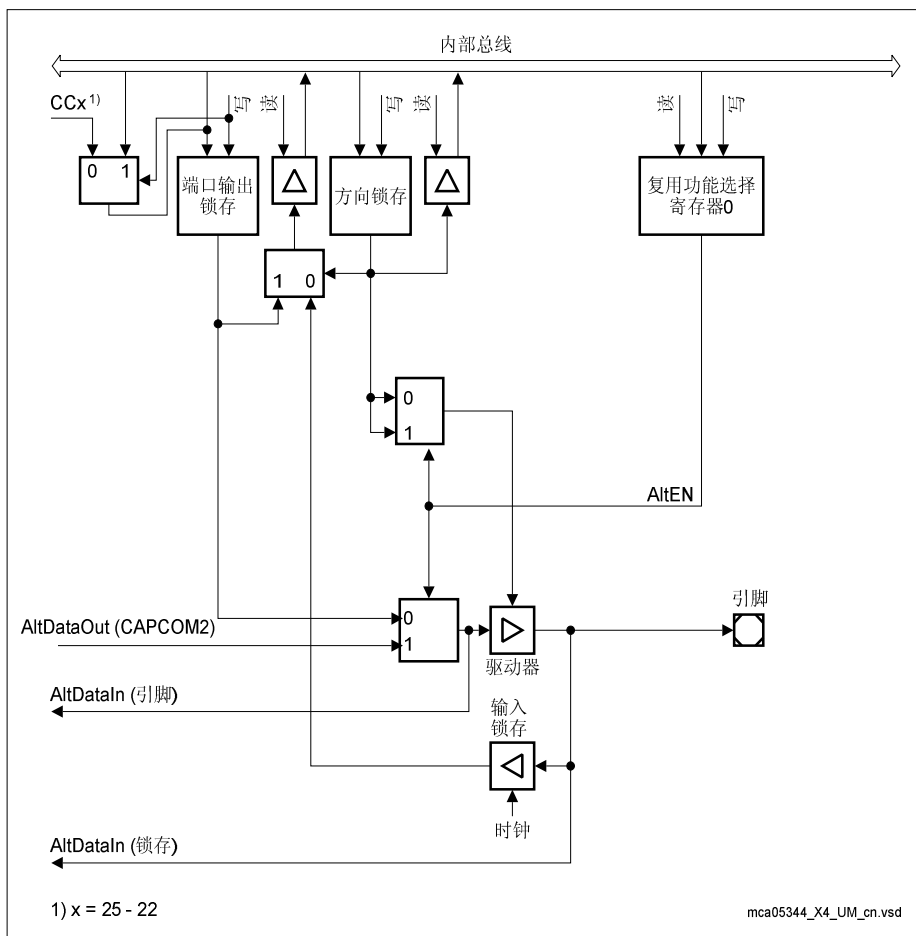


图 7-7 P1L.7、P1H.0、P1H.4、P1H.5 端口配置

表 7-4 P1L.7、P1H.0、P1H.4、P1H.5 复用功能控制

引脚	控制线		寄存器		功能
	AltEN	AltDIR	DP1L/ DP1H	ALTSEL0 P1L/P1H	
P1L.7	0	-	0 或 1	0	通用 IO
P1H.0	-		0	-	CAPCOM2 捕获输入
P1H.x (x = 4, 5)	1		1	1	CAPCOM2 比较输出

7.5 P3 口

P3 口为 13 位端口。用作通用 IO 时，每条线的方向可由相应的方向寄存器 DP3 进行设置。每条线的输出方式可由漏极开路控制寄存器 ODP3 进行设定，选择推挽或者漏极开路模式输出。

P3

P3 口数据寄存器 **SFR (FFC4_H/E2_H)** 复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	-	P13	-	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	-
rW	-	rW	-	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	-

符号	位序号	读写类型	功能描述
P3.y	[11:1], 13,15	rW	P3 口数据寄存器的位 y

DP3

P3 口方向控制寄存器 **SFR (FFC6_H/E3_H)** 复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	-	P13	-	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	-
rW	-	rW	-	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	rW	-

符号	位序号	读写类型	功能描述
DP3.y	[11:1], 13,15	rW	P3 口方向控制寄存器的位 y 0 端口线 P3.y 为输入（高阻） 1 端口线 P3.y 为输出

ODP3

P3 口漏极开路控制寄存器

ESFR (F1C6_H/E3_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	P13	-	P11	P10	P9	P8	P7	P6	P5	P4	P3	P2	P1	-
-	-	rw	-	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	-

符号	位序号	读写类型	功能描述
ODP3.y	[11:1], 13	rw	P3 口漏极开路控制寄存器的位 y 0 端口线 P3.y 以推挽模式输出 1 端口线 P3.y 以漏极开路模式输出

注：引脚 P3.15 不支持漏极开路模式。

SSC0、ASC0、ASC1 和 GPT 的复用功能通过寄存器 ALTSEL0P3 和 ALTSEL1P3 选择。

注：外设输出复用功能的具体选择见表 7-5。

ALTSEL0P3

P3 口复用功能选择寄存器 0

ESFR (F126_H/93_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	P13	-	P11	P10	P9	P8	-	-	P5	-	P3	-	P1	-
-	-	rw	-	rw	rw	rw	rw	-	-	rw	-	rw	-	rw	-

符号	位序号	读写类型	功能描述
ALTSEL0P3.y	1, 3, 5, [11:8], 13	rw	P3 口复用功能选择寄存器 0 的位 y 0 相关外设输出未选作引脚的复用功能 1 相关外设输出被选作引脚的复用功能

ALTSEL1P3

P3 口复用功能选择寄存器 1

ESFR (F128_H/94_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	-	-	-	-	P1	-
-	-	-	-	-	-	-	-	-	-	-	-	-	-	rw	-

符号	位序号	读写类型	功能描述
ALTSEL1 P3.y	1	rw	P3 口复用功能选择寄存器 1 的位 y 0 相关外设输出未选作引脚的复用功能 1 相关外设输出被选作引脚的复用功能

P3 口复用功能

正常工作模式下，P3 口可用作不同的功能，如外部定时器控制线、两个串行接口和 CLKOUT/FOUT。P3 口 IO 和复用功能如图 7-8 所示。

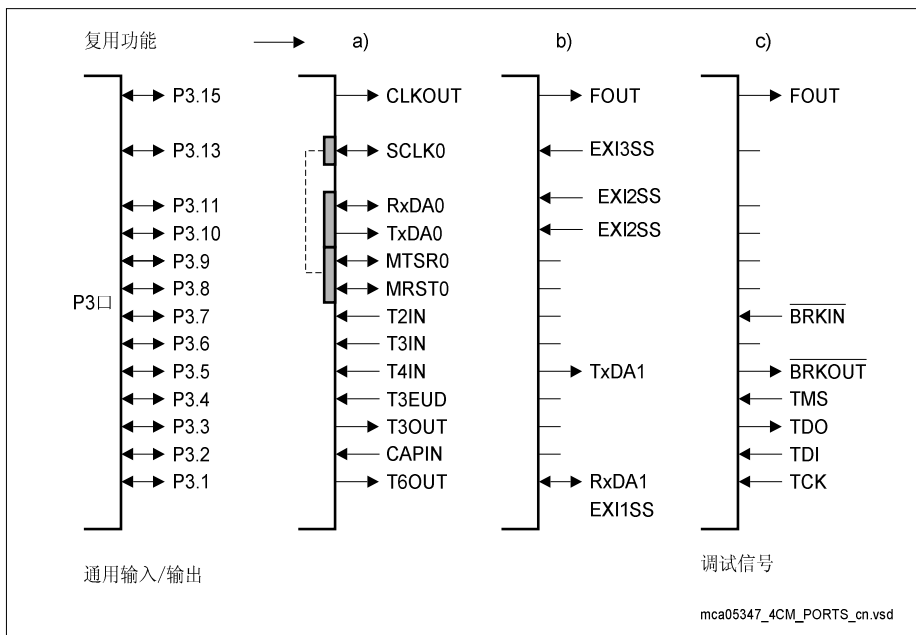


图 7-8 P3 口 IO 及复用功能

选择复用输出功能 – TxDA1、T6OUT、T3OUT、MRST0、MTSR0、TxDA0、RxDA0 和 SCLK0 时，该复用输出功能和端口输出锁存线（通用输出）逻辑与，然后输出。

P3 口各引脚的功能列于表 7-5 中。

表 7-5 P3 口功能

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
P3.0	未实现	-	-	-
P3.1	通用输入	P3.1	ALTSEL0P3. P1=0 且 ALTSEL1P3. P1=0	DP3.P1=0
	通用输出			DP3.P1=1
	定时器 T6 翻转锁存输出, T6OUT	GPT	ALTSEL0P3. P1=0 且 ALTSEL1P3. P1=1 且 P3.P1=1	DP3.P1 = 1
	ASC1 接收输入 RxDA1, 用作输入	ASC1	-	DP3.P1=0
	ASC1 接收输入 RxDA1, 用作输出		ALTSEL0P3. P1=1	DP3.P1 = 1
	JTAG 时钟输入, TCK	OCDS	-	DP3.P1=0
P3.2	通用输入	P3.2	-	DP3.P2=0
	通用输出			DP3.P2 = 1
	GPT12E 捕获输入 CAPIN	GPT		DP3.P2=0
	JTAG 数据输入, TDI	OCDS	-	DP3.P2=0
P3.3	通用输入	P3.3	ALTSEL0P3. P3=0	DP3.P3=0
	通用输出			DP3.P3=1
	定时器 T3 翻转锁存输出, T3OUT	GPT	ALTSEL0P3. P3=1 且 P3.P3=1	DP3.P3=1
	JTAG 数据输出, TDO	OCDS	AltEN1=1	-
P3.4	通用输入	P3.4	-	DP3.P4=0
	通用输出			DP3.P4 = 1

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
	定时器 T3 外部递增/递减计数控制输入，T3EUD	GPT		DP3.P4=0
	JTAG 测试模式选择输入，TMS	OCDS		DP3.P4=0
P3.5	通用输入	P3.5	ALTSEL0P3.P5=0	DP3.P5=0
	通用输出			DP3.P5 =1
	定时器 T4 计数输入，T4IN	GPT	-	DP3.P5=0
	ASC1 发送输出，TxDA1	ASC1	ALTSEL0P3.P5=1	DP3.P5 =1
	调试系统：断点输出 BRKOUT	OCDS	AltEN1=1	-
P3.6	通用输入	P3.6	-	DP3.P6=0
	通用输出			DP3.P6 =1
	定时器 T3 计数输入，T3IN	GPT	-	DP3.P6=0
P3.7	通用输入	P3.7	AltEN1.7=0	DP3.P7=0
	通用输出			DP3.P7 =1
	定时器 T2 计数输入，T2IN	GPT	-	DP3.P7=0
	调试系统：断点输入，BRKIN	OCDS	-	DP3.P7=0
P3.8	通用输入	P3.8	ALTSEL0P3.P8=0	DP3.P8=0
	通用输出			DP3.P8 =1
	SSC0 主机接收输入，MRST0	SSC0	-	DP3.P8=0
	SSC0 从机发送输出，MRST0		ALTSEL0P3.P8=1 且	DP3.P8=1

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
			P3.P8=1	
P3.9	通用输入	P3.9	ALTSEL0P3. P9=0	DP3.P9=0
	通用输出			DP3.P9 =1
	SSC0 从机接收输入 MTSR0	SSC0	-	DP3.P9=0
	SSC0 主机发送输出 MTSR0		ALTSEL0P3. P9=1 且 P3.P9=1	DP3.P9=1
P3.10	通用输入	P3.10	ALTSEL0P3. P10=0	DP3.P10=0
	通用输出			DP3.P10 =1
	ASC0 发送输出 TxDA0	ASC0	ALTSEL0P3. P10=1 且 P3.P10=1	DP3.P10=1
P3.11	通用输入	P3.11	ALTSEL0P3. P11=0	DP3.P11=0
	通用输出			DP3.P11 =1
	ASC0 接收输入 RxDA0，用作输入	ASC0	-	DP3.P11=0
	ASC0 接收输入 RxDA0，用作输出		ALTSEL0P3. P11=1 且 P3.P11=1	DP3.P11 =1
P3.12	未实现			
P3.13	通用输入	P3.13	ALTSEL0P3. P13=0	DP3.P13=0
	通用输出			DP3.P13 =1
	SSC0 从时钟输入， SCLK0	SSC0	-	DP3.P13=0
	SSC0 主时钟输出， SCLK0		ALTSEL0P3. P13=1 且 P3.P13=1	DP3.P13=1
P3.14	未实现			

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
P3.15	通用输入	P3.15	CLKOUT 和 FOUT 均被禁 止	DP3.P15=0
	通用输出			DP3.P15 =1
	系统时钟输出 CLKOUT	-	CLKOUT 被使 能(AltEN1.15)	输出 (AltDIR=1)
	可编程频率输出 FOUT	-	CLKOUT 被禁 止，FOUT 被 使能 (AltEN2.15)	输出 (AltDIR=1)

外部中断和输入引脚的连接关系详见表 5-13。

以下各图表示 P3 口各引脚的不同配置。

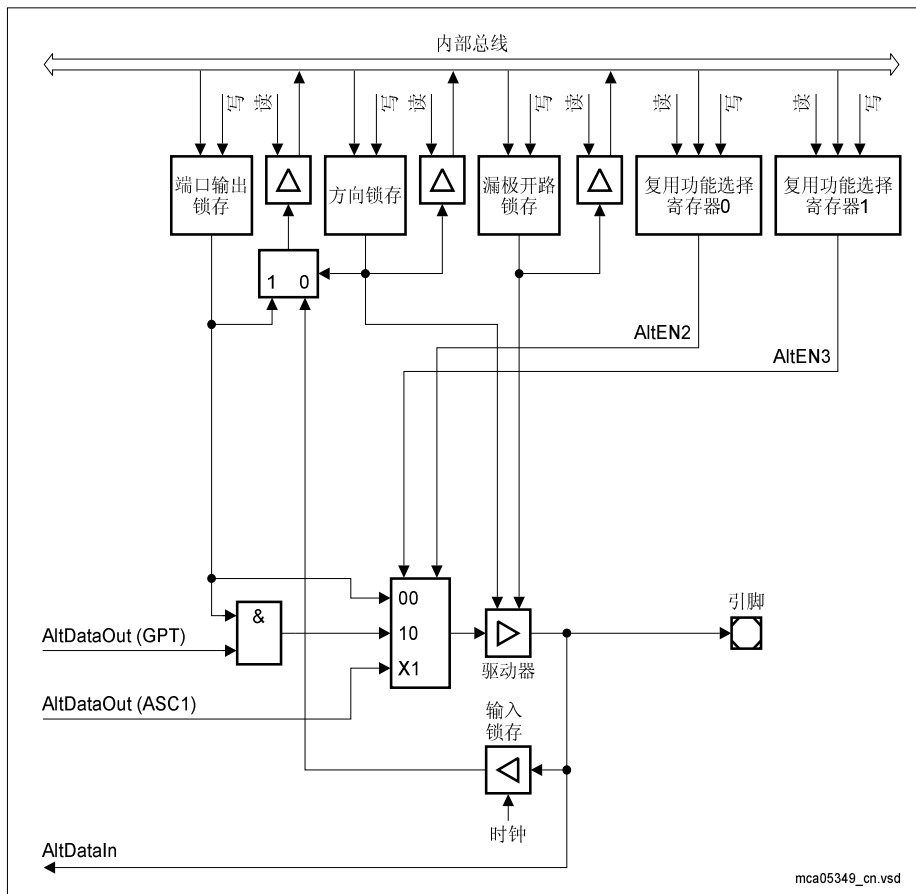


图 7-9 P3.1 端口配置

表 7-6 P3.1 复用功能控制

引脚	控制线				寄存器				功能
	AltEN			AltDIR	DP3L	P3	ALTSEL0 P3	ALTSEL1 P3	
	3	2	1						
P3.1	0	0	-	-	0 或 1	0 或 1	0	0	通用 IO
	1	0			1	1	0	1	GPT 输出
	x	1			1	-	1	-	ASC1 输出

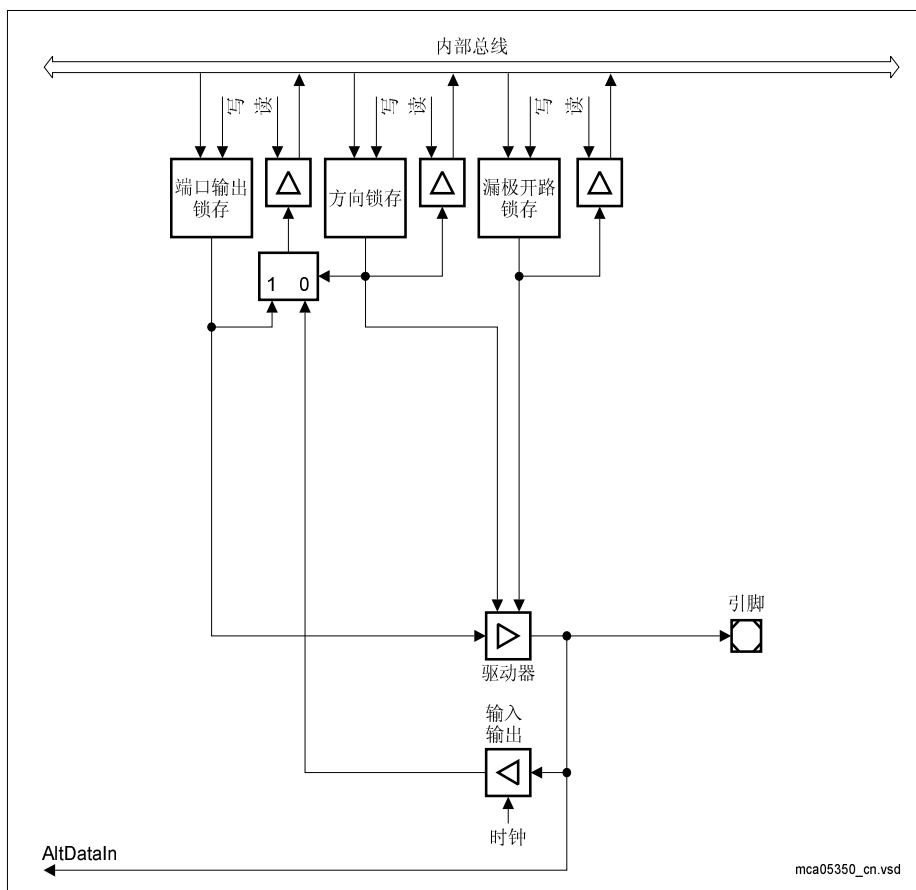


图 7-10 P3.2、P3.4、P3.6 和 P3.7 端口配置

表 7-7 P3.2、P3.4、P3.6 和 P3.7 复用功能控制

引脚	控制线			寄存器		功能
	AltEN		AltDIR	DP3L	ALTSEL0P3	
	2	1				
P3.2, P3.4, P3.6, P3.7	-	-	-	0 或 1	0	通用 IO
				0		GPT 输入

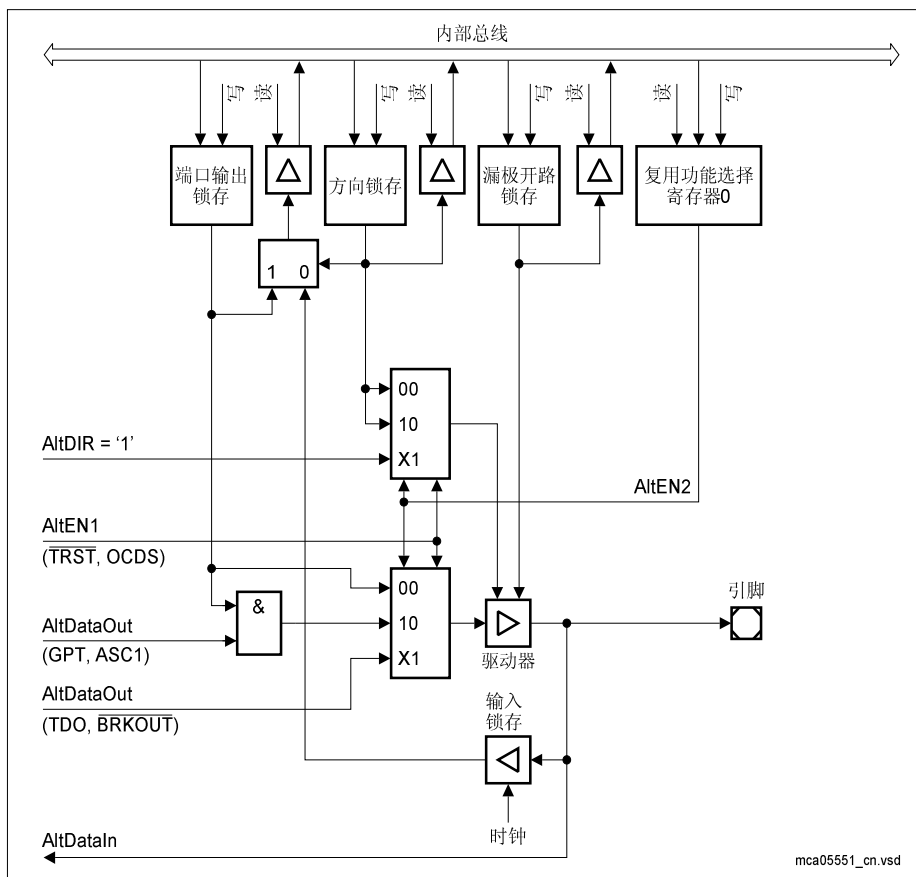


图 7-11 P3.3、P3.5 端口配置

表 7-8 P3.3¹⁾ 和 P3.5 复用功能控制

引脚	控制线			寄存器			功能
	AltEN		Alt DIR	DP3L	P3	ALTSEL0 P3	
	2	1					
P3.3	0	0	-	0 或 1	0 或 1	0	通用 IO
P3.5	1	0		1	1 ²⁾	1	GPT 输出，ASC1 输出
	x	1	1	-	-	-	调试输出

1) P3.3 无复用输入功能。

2) P3.5 复用输出信号不和端口锁存信号逻辑与。

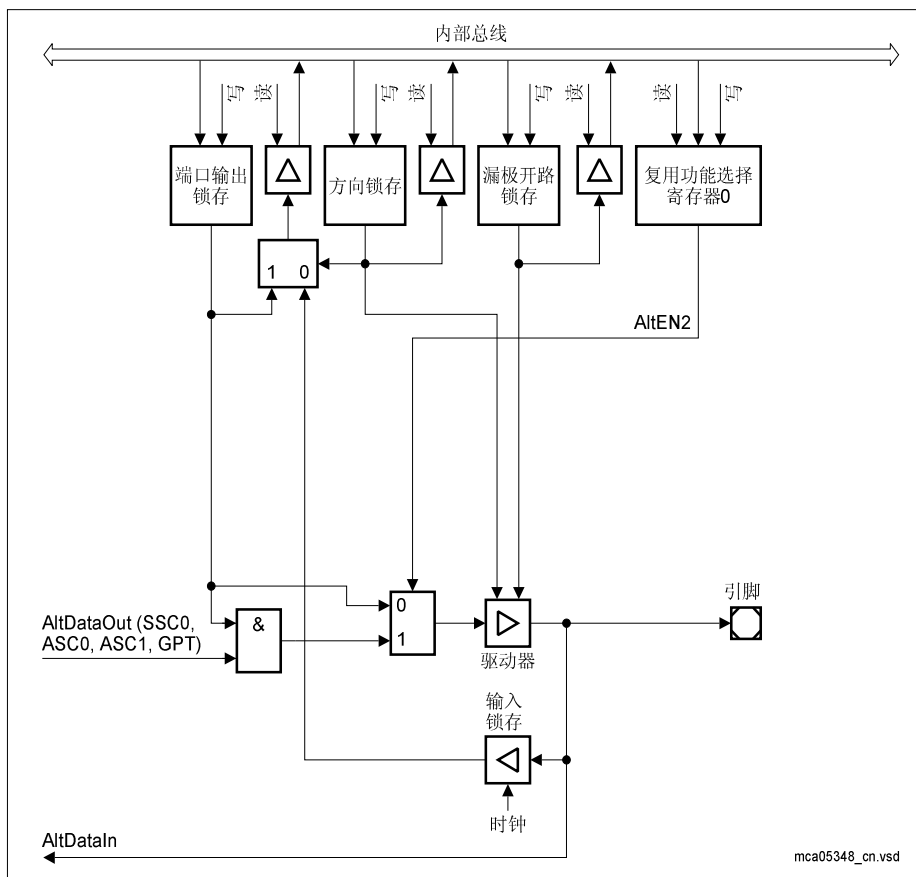


图 7-12 P3.8-P3.11 和 P3.13 端口配置

表 7-9 P3.8-P3.11 和 P3.13 复用功能控制

引脚	控制线			寄存器			功能
	AltEN		Alt DIR	DP3L	P3	ALTSEL0P3	
	2	1					
P3.8 -11 P3.13	0	-	-	0 或 1	0 或 1	0	通用 IO
	1			1	1	1	SSC0、ASC0、ASC1、GPT 输出
	-		-	0	-	-	CAPCOM1、SSC0、ASC0 输入

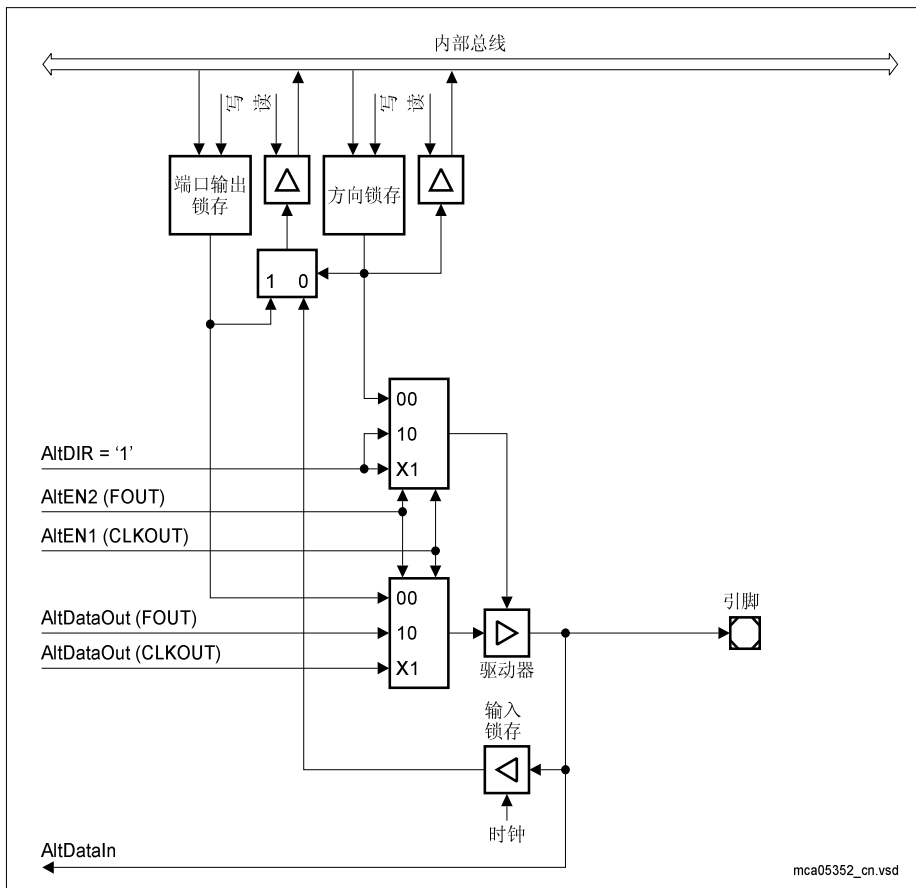


图 7-13 P3.15 端口配置

表 7-10 P3.15 复用功能控制

引脚	控制线			寄存器		功能
	AltEN		Alt DIR	DP3L	ALTSEL0P3	
	2	1				
P3.15	0	0	-	0 或 1	-	通用 IO
	x	1	1	-		CLKOUT
	1	0				FOUT

7.6 P5 口

P5 口为 14 位输入端口。无输出锁存器和方向控制寄存器。写入 P5 的数据将会丢失。

P5

P5 口数据寄存器										SFR (FFA2 _H /D1 _H)					复位值: XXXX _H				
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0				
P15	P14	P13	P12	P11	P10	-	-	P7	P6	P5	P4	P3	P2	P1	P0				
r	r	r	r	r	r	-	-	r	r	r	r	r	r	r	r				

符号	位序号	读写类型	功能描述
P5.y	[15:10], [7:0]	r	P5 口数据寄存器的位 y（只读）

P5 口复用功能

P5 口的每个引脚都与模数转换器的输入复用器连接。高 6 位还可用作 GPT 的复用输入功能。P5 口 IO 和复用功能如图 7-14 所示。

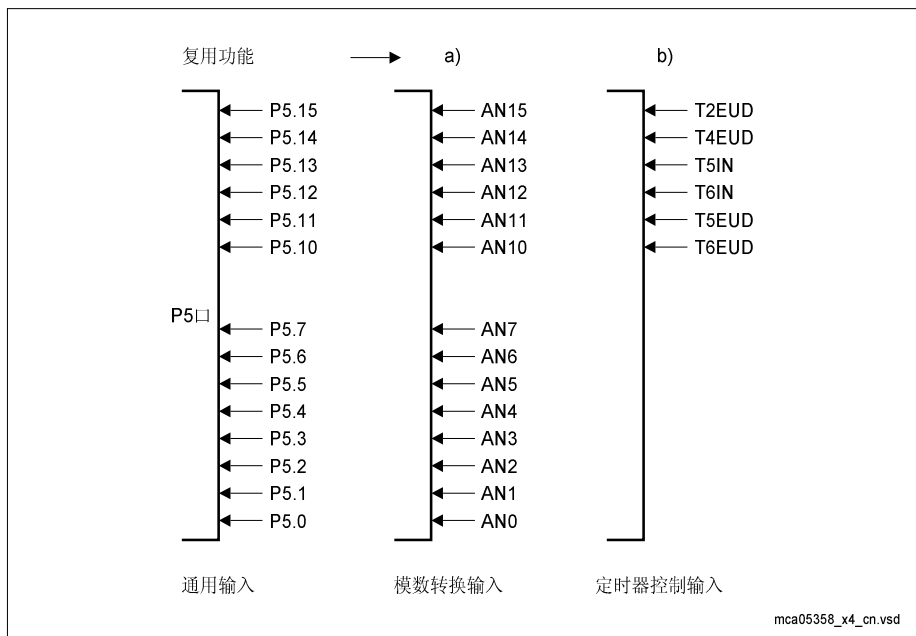


图 7-14 P5 口 IO 及复用功能

P5 口数字输入控制

P5 口引脚可用作数字输入和模拟输入。用作模拟输入时，输入级的施密特触发器必须禁用，通过置位寄存器 P5DIDIS 中的相应位来实现。

复位后，P5 口引脚缺省使能数字输入。

P5DIDIS

P5 口数字输入禁用寄存器

SFR (FFA4_H/D2_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
P15	P14	P13	P12	P11	P10	-	-	P7	P6	P5	P4	P3	P2	P1	P0
rw	rw	rw	rw	rw	rw	-	-	rw	rw	rw	rw	rw	rw	rw	rw

符号	位序号	读写类型	功能描述
P5DIDIS.y	[15:10], [7:0]	rw	P5 口位 y 数字输入控制 0 使能输入级的施密特触发器 1 禁用输入级的施密特触发器，引脚用作模拟输入时需要这样设置

P5 口各引脚的功能列于表 7-11 中。

表 7-11 P5 口功能

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
P5.x (x=7-0)	通用输入	P5.x	P5DIDIS.Px=0	仅输入
	模拟输入通道 ANx	ADC	P5DIDIS.Px=1	
P5.10	通用输入	P5.10	P5DIDIS.P10=0	仅输入
	模拟输入通道 AN10	ADC	P5DIDIS.P10=1	
	定时器 T6 外部递增/递减计数输入，T6EUD	GPT12E	P5DIDIS.P10=0	
P5.11	通用输入	P5.11	P5DIDIS.P11=0	仅输入
	模拟输入通道 AN11	ADC	P5DIDIS.P11=1	

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
	定时器 T5 外部递增/递减计数输入, T5EUD	GPT12E	P5DIDIS.P11=0	
P5.12	通用输入	P5.12	P5DIDIS.P12=0	仅输入
	模拟输入通道 AN12	ADC	P5DIDIS.P12=1	
	定时器 T6 计数输入, T6IN	GPT	P5DIDIS.P12=0	
P5.13	通用输入	P5.13	P5DIDIS.P13=0	仅输入
	模拟输入通道 AN13	ADC	P5DIDIS.P13=1	
	定时器 T5 计数输入, T5IN	GPT	P5DIDIS.P13=0	
P5.14	通用输入	P5.14	P5DIDIS.P14=0	仅输入
	模拟输入通道 AN14	ADC	P5DIDIS.P14=1	
	定时器 T4 外部递增/递减计数输入, T4EUD	GPT	P5DIDIS.P14=0	
P5.15	通用输入	P5.15	P5DIDIS.P15=0	仅输入
	模拟输入通道 AN14	ADC	P5DIDIS.P15=1	
	定时器 T2 外部递增/递减计数输入, T2EUD	GPT	P5DIDIS.P15=0	

P5 口引脚配置如图 7-15 所示。

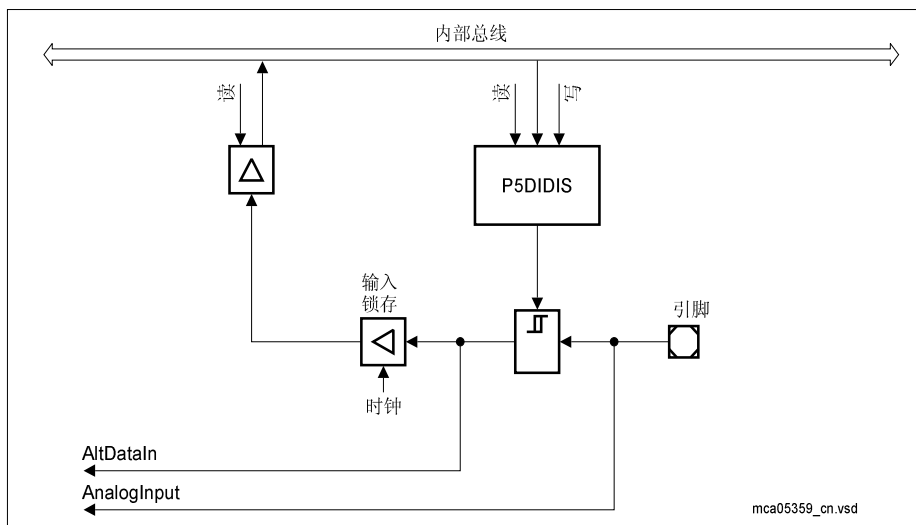


图 7-15 P5 口端口配置

7.7 P9 ☐

P9 口为 6 位端口。用于通用 IO 时，每条线的方向可由相应的方向控制寄存器 DP9 进行设置。每条线的输出方式可由漏极开路控制寄存器 ODP9 进行设定，选择推挽或者漏极开路模式输出。

P9

P9 口数据寄存器

SFR (FF16_H/8B_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-								-	-	P5	P4	P3	P2	P1	P0
-								-	-	rwh	rwh	rwh	rwh	rwh	rwh

符号	位序号	读写类型	功能描述
P9.y	[5:0]	rwh	P9 口数据寄存器的位 y

注：位 P9.0 - P9.5 用作 CAPCOM2 输出时被位保护。

DP9

P9 口方向控制寄存器

SFR (FF18_H / 8C_H)

复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-								-	-	P5	P4	P3	P2	P1	P0
-								-	-	RW	RW	RW	RW	RW	RW

符号	位序号	读写类型	功能描述
DP9.y	[5:0]	rw	P9 口方向控制寄存器的位 y 0 端口线 P9.y 为输入（高阻） 1 端口线 P9.y 为输出

ODP9

P9 口漏极开路控制寄存器 **SFR (FF1A_H/8D_H)** **复位值: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-						-	-	P5	P4	P3	P2	P1	P0		
-							-	-	rw	rw	rw	rw	rw	rw	

符号	位序号	读写类型	功能描述
ODP9.y	[5:0]	rw	P9 口漏极开路控制寄存器的位 y 0 端口线 P9.y 以推挽模式输出 1 端口线 P9.y 以漏极开路模式输出

TwinCAN 和 CAPCOM2 模块的复用功能通过寄存器 ALTSEL0P9 和 ALTSEL1P9 选择。

ALTSEL0P9

P9 口复用功能选择寄存器 0 **ESFR (F138_H/9C_H)** **复位值: 0000_H**

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-						-	-	-	-	P3	-	P1	-		
-							-	-	-	-	rw	-	rw	-	

符号	位序号	读写类型	功能描述
ALTSEL0 P9.y	3,1	rw	P9 口复用功能选择寄存器 0 的位 y 0 相关外设输出未选作引脚的复用功能 1 相关外设输出被选作引脚的复用功能

ALTSEL1P9

P9 口复用功能选择寄存器 1 ESFR (F13A_H/9D_H) 复位值: 0000_H

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-								-	-	P5	P4	P3	P2	P1	P0
-								-	-	rw	rw	rw	rw	rw	rw

符号	位序号	读写类型	功能描述
ALTSEL1 P9.y	[5:0]	rw	P9 口复用功能选择寄存器 1 的位 y 0 相关外设输出未选作引脚的复用功能 1 相关外设输出被选作引脚的复用功能

P9 口复用功能

P9 口引脚可用作 TwinCAN 模块的发送和接收，也可用作 CAPCOM2 输入/输出。

P9 口 IO 和复用功能如图 7-16 所示。

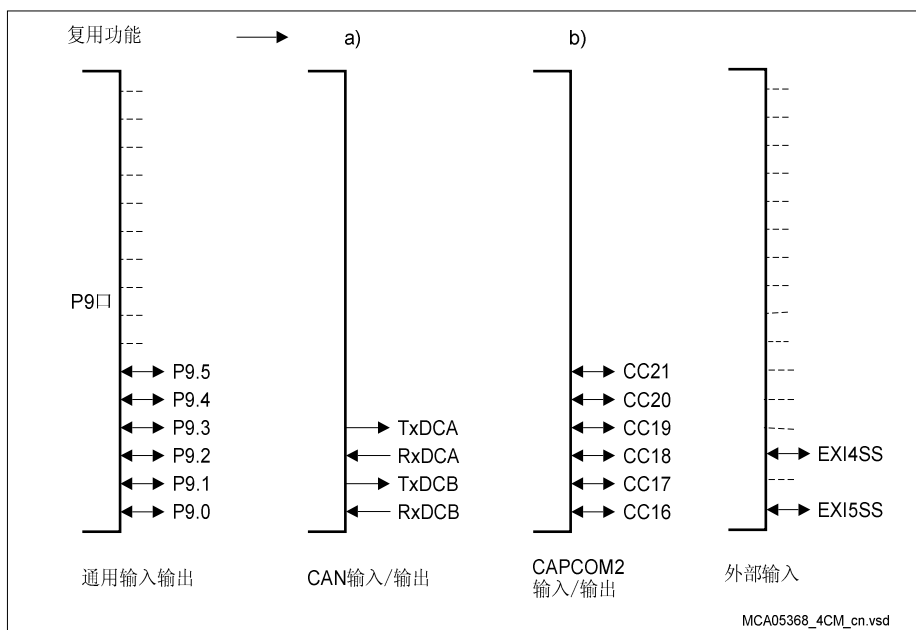


图 7-16 P9 口 IO 及复用功能

P9 口各引脚的功能列于表 7-12 中。

表 7-12 P9 口功能

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
P9.0	通用输入	P9.0	ALTSEL0P9. P0=0 且 ALTSEL1P9. P0=0	DP9.P0=0
	通用输出			DP9.P0 =1
	CC16I 捕获输入	CAPCOM2	-	DP9.P0=0
	CC16O 比较输出		ALTSEL0P9. P0=0 且 ALTSEL1P9. P0=1	DP9.P0 =1
	TwinCAN 接收输入， RxDCA, RXDCB	TwinCAN	-	DP9.P0=0
P9.1	通用输入	P9.1	ALTSEL0P9. P1=0 且 ALTSEL1P9. P1=0	DP9.P1=0
	通用输出			DP9.P1 =1
	CC17I 捕获输入	CAPCOM2	-	DP9.P1=0
	CC17O 比较输出		ALTSEL0P9. P1=0 且 ALTSEL1P9. P1=1	DP9.P1 =1
	TwinCAN 发送输出， TxDCB	TwinCAN	ALTSEL0P9. P1=1 且 ALTSEL1P9. P1=1	DP9.P1=1
P9.2	通用输入	P9.2	ALTSEL0P9. P2=0 且 ALTSEL1P9. P2=0	DP9.P2=0
	通用输出			DP9.P2 =1
	CC18I 捕获输入	CAPCOM2	-	DP9.P2=0

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
	CC18O 比较输出		ALTSEL0P9. P2=0 且 ALTSEL1P9. P2=1	DP9.P2 =1
	TwinCAN 接收输入, RxDCA, RXDCB	TwinCAN	-	DP9.P2=0
P9.3	通用输入	P9.3	ALTSEL0P9. P3=0 且 ALTSEL1P9. P3=0	DP9.P3=0
	通用输出			DP9.P3 =1
	CC19I 捕获输入	CAPCOM2	-	DP9.P3=0
	CC19O 比较输出		ALTSEL0P9. P3=0 且 ALTSEL1P9. P3=1	DP9.P3 =1
	TwinCAN 发送输出, TxDCA	TwinCAN	ALTSEL0P9. P3=1 且 ALTSEL1P9. P3=1	DP9.P3=1
P9.4	通用输入	P9.4	ALTSEL0P9. P4=0 且 ALTSEL1P9. P4=0	DP9.P4=0
	通用输出			DP9.P4 =1
	CC20I 捕获输入	CAPCOM2	-	DP9.P4=0
	CC20O 比较输出		ALTSEL0P9. P4=0 且 ALTSEL1P9. P4=1	DP9.P4 =1
P9.5	通用输入	P9.5	ALTSEL0P9. P5=0 且 ALTSEL1P9. P5=0	DP9.P5=0
	通用输出			DP9.P5 =1
	CC21I 捕获输入	CAPCOM2	-	DP9.P5=0

端口引脚	引脚功能	相关寄存器/ 模块	复用功能	方向控制
	CC21O 比较输出		ALTSEL0P9. P5=0 且 ALTSEL1P9. P5=1	DP9.P5 =1

外部中断和输入引脚的连接关系详见表 5-13。

以下各图表示 P9 口各引脚的不同配置。

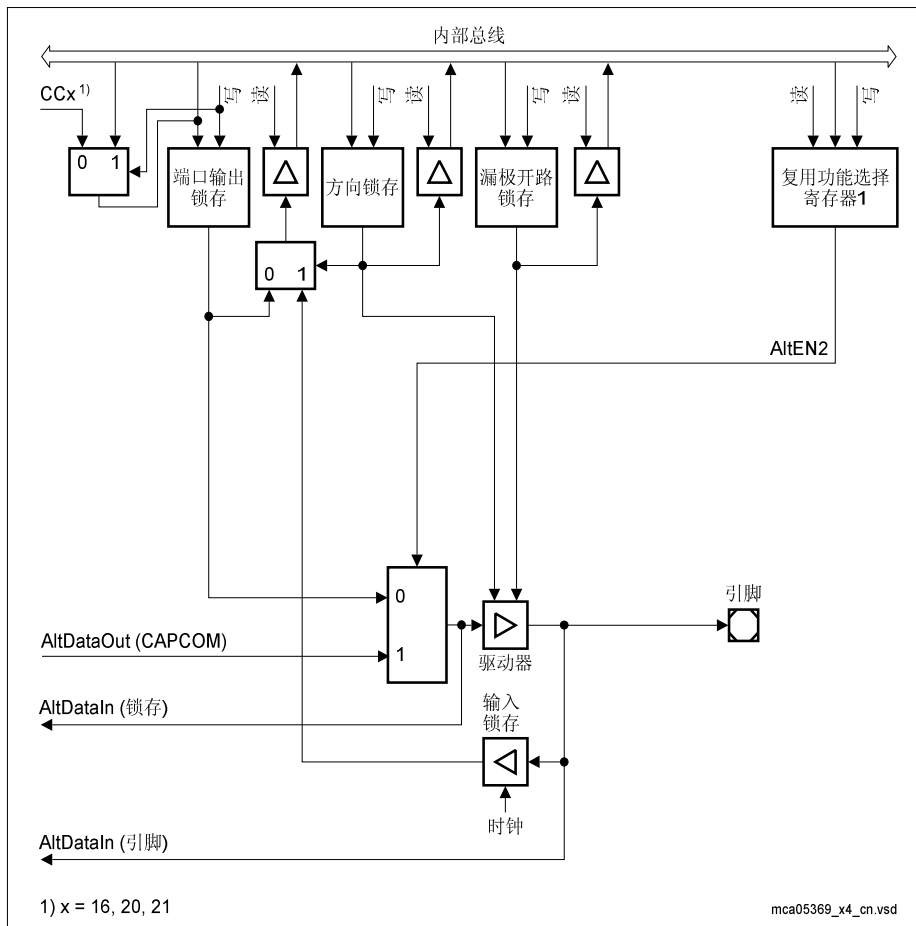


图 7-17 P9.0、P9.2、P9.4 和 P9.5 端口配置

表 7-13 P9.0、P9.2、P9.4 和 P9.5 复用功能控制

引脚	控制线	寄存器			功能
	AltEN2	DP9	ALTSEL1 P9	ALTSEL0 P9	
P9.0,	0	0 或 1	0	-	通用 IO
P9.2,	-	0	-		CAN、CAPCOM2 输入
P9.4, P9.5	1	1	1		CAPCAM2 比较输出

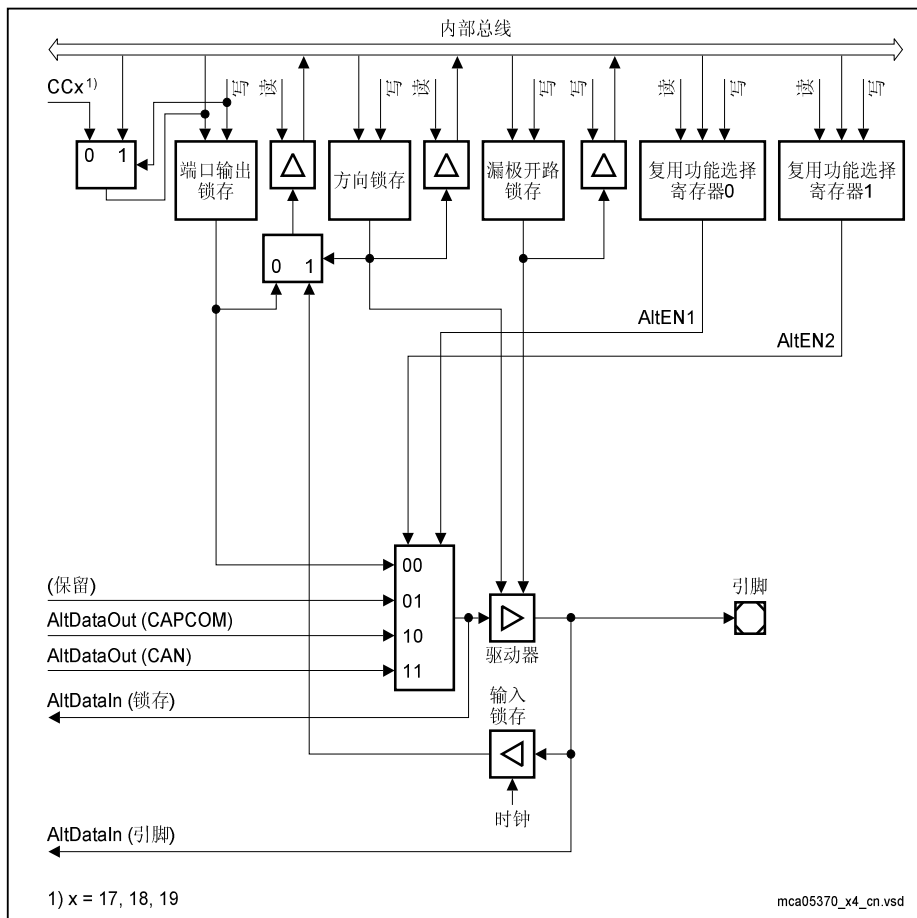


图 7-18 P9.1 和 P9.3 端口配置

表 7-14 P9.1 和 P9.3 复用功能控制

引脚	控制线		寄存器			功能
	AltEN		DP9	ALTSEL1 P9	ALTSEL0 P9	
	2	1				
P9.1, P9.3	0	0	0 或 1	0	0	通用 IO
	-		0	-		CAN、CAPCOM2 输入
	0	1	1	0	1	保留
	1	1	1	1	1	CAN 输出
	1	0	1	1	0	CAPCOM2 比较输出

8 专用引脚

XC164CM 中大多数功能模块的输入/输出或控制信号由并行端口引脚的复用功能来实现。不过也有一些信号需要使用专用引脚，如振荡器、特殊控制信号以及电源。

表 8-1 列出 XC164CM 中的 17 个专用引脚。

表 8-1 XC164CM 专用引脚

引脚	功能
$\overline{\text{NMI}}$	非屏蔽中断输入
XTAL1、XTAL2	振荡器输入/输出（主振荡器）
$\overline{\text{RSTIN}}$	复位输入
$\overline{\text{TRST}}$	调试系统的测试复位输入
V_{AREF} 、 V_{AGND}	模数转换器电源
V_{DDI}	内部逻辑的数字电源（2 个引脚）
V_{DDP}	端口驱动的数字电源（4 个引脚）
V_{SS}	数字电源接地（4 个引脚）

非屏蔽中断输入 $\overline{\text{NMI}}$ 可通过外部信号（如电源失效信号）产生一个高优先级的强制中断。 $\overline{\text{NMI}}$ 也可使 PWRDN 指令生效，将 XC164CM 切换到掉电模式。每个系统时钟周期都对 $\overline{\text{NMI}}$ 引脚进行采样，检查信号有无跳变。

振荡器输入 XTAL1 和输出 XTAL2 将内部主振荡器连接到外部晶振。主振荡器由一个反相器和一个反馈元件组成。标准外部振荡电路（见章节 6.2.1）由晶振、两个小终端电容和限制晶振电流的串联电阻组成。主振荡器用于产生 XC164CM 的基本工作时钟信号。

外部时钟信号可从引脚 XTAL1 输入，使 XTAL2 处于开路状态。

复位输入 $\overline{\text{RSTIN}}$ 使 XC164CM 进入预先定义好的复位状态。上电、硬件故障或手动复位等外部事件可触发复位。

复位配置的描述见章节 6.1.4。

测试复位输入 $\overline{\text{TRST}}$ 使 XC164CM 调试系统进入复位状态。正常工作期间，该输入应保持有效。释放 $\overline{\text{TRST}}$ 引脚的有效状态，使得片上调试系统能够进行调试。

模数转换器电源引脚 V_{AREF} 和 V_{AGND} 为片上 ADC 模块提供独立的电源（参考电压）。当 V_{AREF} 和 V_{AGND} 与 V_{DD} 和 V_{SS} 分开之后，可降低数字逻辑部分对模拟输入信号的耦合噪声，从而提高转换结果的稳定性。

电源引脚 V_{DDI}/V_{DDP} 和 V_{SS} 为 XC164CM 的数字逻辑电路提供电源。 V_{DD}/V_{SS} 的去耦电容应尽量靠近引脚放置。 V_{DDI} 引脚（2.5V）为 XC164CM 内部逻辑模块提供电源， V_{DDP} 引脚（5.0V）为输出端口驱动提供电源。

注：所有 V_{DD} 和 V_{SS} 引脚必须分别连接到各自的电源和地上。

9 LXBus 控制器（EBC）

XC164CM 的外部总线控制器（EBC）只可访问片内 LXBus 模块 TwinCAN。LXBus 是外部总线的内部标识，用来控制对片内外设和模块的访问。

注：XC164CM 的 EBC 不支持对片外器件的访问。

EBC 的功能通过一组配置寄存器来控制。

功能控制寄存器 FCONCS7 通过地址（复用/非复用）、数据（16 位/8 位）和片选使能定义 LXBus 总线周期。时序配置寄存器 TCONCS7 控制总线访问的时序，规定了由不同访问节拍组成的总线周期时序。所有这些参数用于访问由地址选择寄存器 ADDRSEL7 定义的地址区域。

寄存器组（FCONCS7 /TCONCS7 /ADDRSEL7）定义了一个可编程的“地址窗”，使用片选信号 $\overline{CS7}$ 来访问 LXBus 上的片内 TwinCAN 模块。

9.1 时序原则

9.1.1 基本总线周期协议

总线时序定义为 6 个不同的时序节拍（A-F）。这些节拍定义了执行任意访问时所需的所有控制信号。当一个节拍开始时，在给定的输出延迟内输出信号可改变。在该输出延迟过后，控制输出信号在该节拍内保持稳定。每个节拍占用的时钟周期数可编程设定。

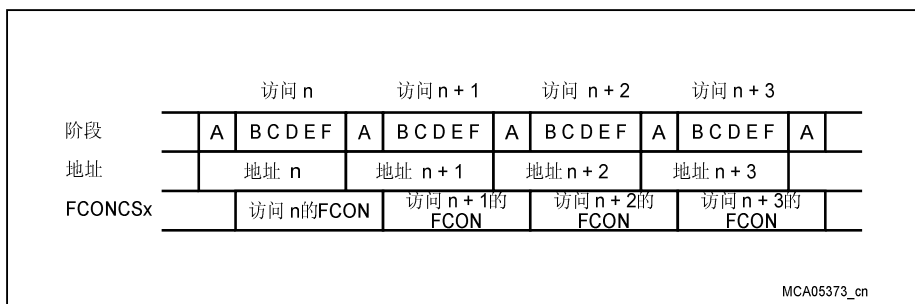


图 9-1 访问序列的节拍示意图

节拍 A 将来自前一周期的数据总线驱动器设置成高阻态（ \overline{CS} 切换后的高阻等待状态）。并非在每个访问周期都会插入节拍 A，只有在改变 \overline{CS} 时才插入。如果前一次访问使用的 \overline{CS} （ \overline{CSx} ）和下一次访问使用的 \overline{CS} （ \overline{CSy} ）不同，则根据前一次 \overline{CS} （ \overline{CSx} ）设置的控制位来执行节拍 A 的周期。

在下一个周期的地址和 ALE 有效后，插入节拍 A。

9.1.2 总线周期示例：最快访问周期

TwinCAN 使用最快非复用总线周期，由信号 READY 控制。

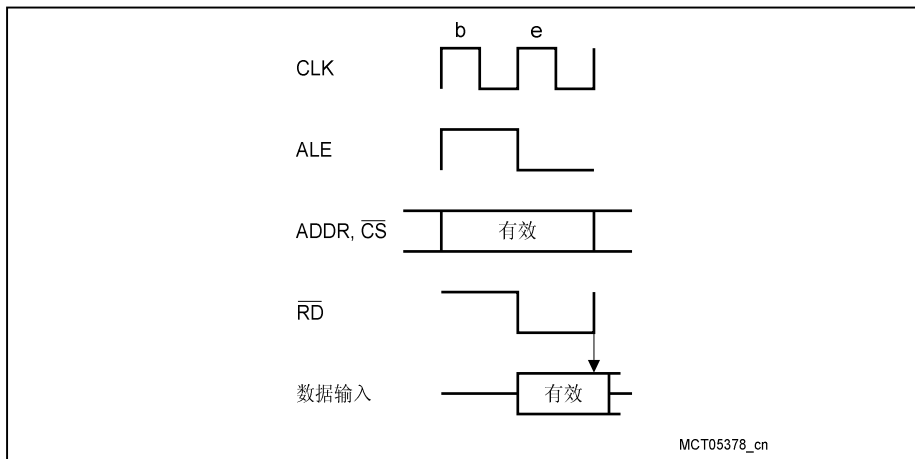


图 9-2 最快读周期，非复用总线

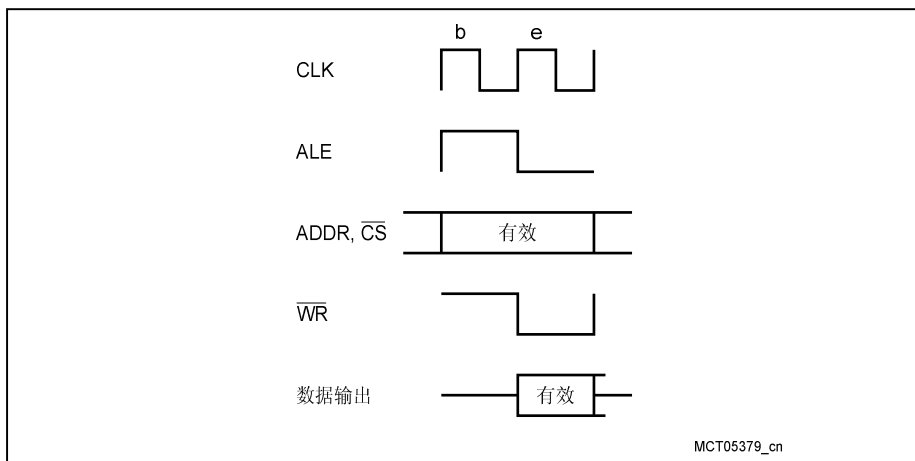


图 9-3 最快写周期，非复用总线

9.2 功能描述

9.2.1 配置寄存器简介

XC164CM EBC 和 XC166 系列其它衍生器件兼容。EBC 占用/预留 128 字节的地址空间。

注：所有 EBC 寄存器受 EINIT 保护机制写保护。因此，每次执行 EINIT 指令之后，这些寄存器不可写。

9.2.2 时序配置寄存器 TCONCS7

时序控制寄存器用来设置总线周期中各节拍占用的周期数。访问地址窗时，该寄存器可以被重新设置，新设置在下一次访问时有效。

TCONCS7

CS7 时序配置寄存器					XSFR (EE48H/--)					复位值: 0000H					
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
	WRPHF		RDPHF		PHE				PHD	PHC		PHB	PHA		
-	rw		rw		rw				rw	rw		rw	rw		

注：TCONCS7 对应片选信号 $\overline{CS7}$ ，用来控制对 LXBus 外设 TwinCAN 的内部访问。

符号	位序号	读写类型	功能描述
WRPHF	[14:13]	rw	写节拍 F 00 0 个时钟周期 11 3 个时钟周期
RDPHF	[12:11]	rw	读节拍 F 00 0 个时钟周期 11 3 个时钟周期
PHE	[10:6]	rw	节拍 E 00000: 1 个时钟周期

符号	位序号	读写类型	功能描述
			11111: 32 个时钟周期
PHD	5	rw	节拍 D 0 0 个时钟周期 1 1 个时钟周期
PHC	[4:3]	rw	节拍 C 00 0 个时钟周期 11 3 个时钟周期
PHB	2	rw	节拍 B 0 1 个时钟周期 1 2 个时钟周期
PHA	[1:0]	rw	节拍 A 00 0 个时钟周期 11 3 个时钟周期

9.2.3 功能配置寄存器 FCONCS7

功能配置寄存器用来控制选定地址窗的总线功能。总线分为 8 位和 16 位总线、复用和非复用模式。此外，还可定义地址窗（及其片选信号 $\overline{CS7}$ ）是否被使能。

FCONCS7

CS7 功能配置寄存器											XSFR (EE4AH/--)		复位值: 0027 _H		
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
-	-	-	-	-	-	-	-	-	-	BTYP	-	RDY MOD	RDY EN	EN CS	
-	-	-	-	-	-	-	-	-	-	rw	-	rw	rw	rw	

注：TCONCS7 对应片选信号 $\overline{CS7}$ ，用来控制对 LXBus 外设 TwinCAN 的内部访问。

符号	位序号	读写类型	功能描述
BTYP	[5:4]	rw	总线类型选择 00 8 位非复用总线 01 8 位复用总线 10 16 位非复用总线 11 16 位复用总线
RDYMOD	2	rw	准备就绪模式 0 异步准备就绪 1 同步准备就绪
RDYEN	1	rw	准备就绪使能 0 访问时序由位域 PHEX 控制 1 访问时序由位域 PHEX 和 READY 信号控制
ENCS	0	rw	片选使能 0 禁止 1 使能

注：根据 ENCS7，片选信号 $\overline{CS7}$ 及其相关寄存器组可被使能，用于控制对 LXBus 外设 TwinCAN 的内部访问。

9.2.4 地址窗选择寄存器 ADDRSEL7

ADDRSEL7

CS7 对应的地址区域/大小 XSFR (EE4EH /--) 复位值: 2000_H

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

符号	位序号	读写类型	功能描述
RGSAD	[15:4]	rw	地址区域起始地址选择
RGSZ	[3:0]	rw	地址区域大小选择（见表 9-1）

地址区域定义

使能的寄存器组 FCONCS7 /TCONCS7/ADDRSEL7 定义了 XC164CM 地址空间中的独立地址区域。在这个地址区域内，LXBus 的访问条件可独立控制，地址区域（窗）由 ADDRSEL7 寄存器定义。地址窗的起始地址定义了寻址窗内存储器/模块时不需要用到的高位地址（见表 9-1）。由 ADDRSEL7.RGSZ 选择地址窗大小，定义了 ADDRSEL7.RGSAD 中的相关位（由“R”标记），位“R”用于选择相应窗口请求地址的高位地址。请求地址中的其它位用来对该地址窗内的存储器寻址。ADDRSEL7.RGSAD 的低位（标记为“x”）被忽略。

表 9-1 由 ADDRSEL7 选定的地址区域和大小

ADDRSEL7		地址窗	
区域大小 RGSZ	RGSAD 的相关位（R）	选定的地址区域	地址窗起始地址 A[23:0] 用 RGSAD 中的 R 位选择
3 ... 0	15 ... 4	大小	A23 ... A0
0000	RRRR RRRR RRRR	4 KB	RRRR RRRR RRRR 0000 0000 0000
0001	RRRR RRRR RRRx	8 KB	RRRR RRRR RRR0 0000 0000 0000
0010	RRRR RRRR RRxx	16 KB	RRRR RRRR RR00 0000 0000 0000
0011	RRRR RRRR Rxxx	32 KB	RRRR RRRR R000 0000 0000 0000
0100	RRRR RRRR xxxx	64 KB	RRRR RRRR 0000 0000 0000 0000
0101	RRRR RRRx xxxx	128 KB	RRRR RRR0 0000 0000 0000 0000
0110	RRRR RRxx xxxx	256 KB	RRRR RR00 0000 0000 0000 0000
0111	RRRR Rxxx xxxx	512 KB	RRRR R000 0000 0000 0000 0000
1000	RRRR xxxx xxxx	1 MB	RRRR 0000 0000 0000 0000 0000
1001	RRRx xxxx xxxx	2 MB	RRR0 0000 0000 0000 0000 0000
1010	RRxx xxxx xxxx	4 MB	RR00 0000 0000 0000 0000 0000
1011	Rxxx xxxx xxxx	8 MB	R000 0000 0000 0000 0000 0000
11xx	xxxx xxxx xxxx	保留	---- ---- ---- ---- ---- ----

注：地址窗起始地址只能由表 9-1 指定边界。

9.2.5 对 TwinCAN 的访问控制

访问 TwinCAN 模块需要使用 LXBus 控制。

访问 TwinCAN 时，使用 $\overline{\text{CS7}}$ 、控制寄存器 ADDRSEL7、TCONCS7 和 FCONCS7。用 $\overline{\text{CS7}}$ 控制选择 LXBus；用 ADDRSEL7 定义地址窗，建议将其设定在“外部 IO 区”（从 20'0000_H 到 3F'0000_H）。所有的外部地址区域中，只有外部 IO 区能够确保在前一个写访问结束后开始执行读访问。

复位后（由启动程序控制），TwinCAN 地址窗缺省设置在 20'0000_H 至 20'0FFF_H（4 KB），对应 ADDRSEL7 的缺省值为 2000_H。该初值可以由用户修改。

总线功能控制寄存器 FCONCS7 的缺省初始值根据 TwinCAN 的应用要求进行选择：16 位非复用总线，访问时序由同步 READY 控制。对应 FCONCS7 的缺省值为 0027_H。

复位后，初始的 LXBus 周期时序（由寄存器 TCONCS7 控制）最短，1 个总线周期占用 2 个时钟周期。但是利用 TwinCAN 的 READY 功能进行等待状态控制时，该最小时序会被延长。该时序控制由 TCONCS7 的复位值（0000_H）控制。

注意：尽管 TwinCAN 寄存器组可编程设定，但强烈建议用户不要修改这些寄存器的缺省值，ADDRSEL7 除外（因为 TwinCAN 是片上外设）。

9.2.6 关机控制

如果系统控制单元（SCU）发送关机请求，必须确保在整个芯片切换至空闲、掉电、休眠或软件复位模式之前，EBC 的所有功能处于非工作状态。除了要完成正在运行的总线周期，还要执行已请求的总线周期。只有当该关机序列结束时，EBC 模块（以及其它模块）才能发送关机应答，芯片进入被请求模式。

EBC 的关机控制：

- 结束所有挂起的周期请求
- 发送关机应答

9.3 EBC 寄存器表

表 9-2 列出所有 XC164CM 中实现的 EBC 配置寄存器，按物理地址排序。这些寄存器都位于 XSFR 区（内部 IO 空间）。

表 9-2 EBC 存储器表（按物理地址排序）

名称	物理地址	描述	复位值 ¹⁾
保留	EE00 _H – EE46 _H	保留，不要使用	-
TCONCS7	EE48 _H	$\overline{\text{CS7}}$ 时序配置寄存器	0000 _H
FCONCS7	EE4A _H	$\overline{\text{CS7}}$ 功能配置寄存器	0027 _H
ADDRSEL7	EE4E _H	$\overline{\text{CS7}}$ 地址大小和区域寄存器	2000 _H
保留	EE50 _H – EEFF _H	保留，不要使用	–

1) 注：为了和将来功能扩展的产品保持兼容，不应用软件对保留（和未列出的）地址进行读写操作。

10 引导程序加载器

XC164CM 的内嵌引导程序加载器（BSL）提供了一种加载启动程序（复位后执行）的机制，通过串行接口实现。在这种情况下，初始化代码不需要存放在外部存储器或内部 ROM /OTP/ Flash 中。

引导程序加载器将代码/数据移入内部 PSRAM，Flash/ROM 存储器不是必需的，不过，它可用来提供查找表、或“核心代码”，即用于 IO 操作、数字运算、系统初始化等的一系列通用子程序。

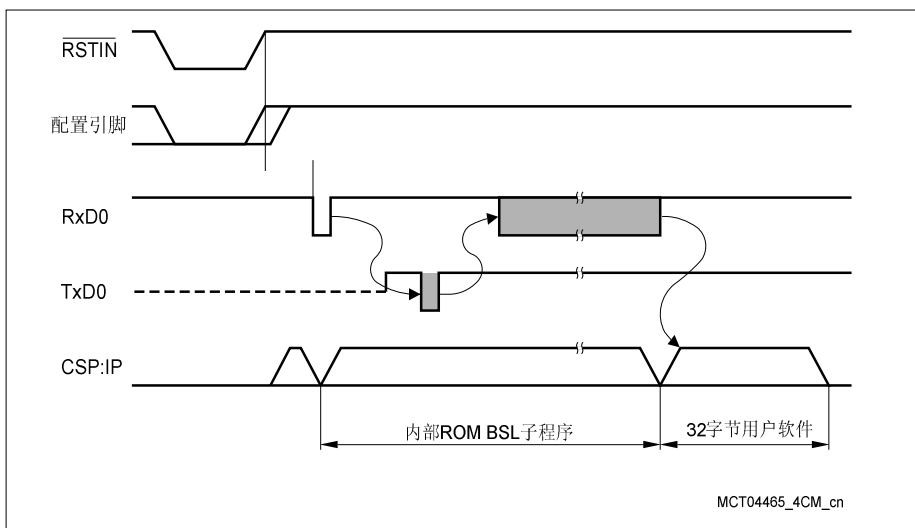


图 10-1 引导程序加载器序列

通过引导程序加载器可将整个应用软件装入无 ROM 存储器的系统；将临时软件装入系统用于测试或校正；还可加载 Flash 器件的编程程序。

BSL 机制不仅可用于标准的系统启动，还可用于系统维护（固件更新）、行尾编程或测试等特殊情况。

10.1 进入引导程序加载模式

硬件复位时，由外部配置触发 XC164CM 进入 BSL 模式：

- 内部复位结束时，引脚 $P9.5 = 0$ ， $P9.4 = 1$ ， $\overline{TRST} = 1$

引导程序加载器代码存储在特殊的 Boot-ROM 中，不需使用 Flash 存储器区。

复位期间用于激活 BSL 的外部硬件可以是一个简单的下拉电阻，每次硬件复位时使用；也可以使用可切换的方式（通过跳线或外部信号），临时使用引导程序加载器。

若每次硬件复位后进入引导程序加载模式，在复位期间可通过一个简单的下拉电阻激活 BSL；若需临时进入引导程序加载模式，可选择通过跳线或外部信号来实现。

发送 ID 字节之后 ASC0 接收器才能被使能。BSL 与主机采用半双工连接方式。

注：进入 BSL 模式的正确复位配置需要将一组相关引脚驱动到指定的逻辑电平（见[章节 6.1.4](#)）。

BSL 模式的初始状态

进入 BSL 模式并进行相应初始化之后，XC164CM 扫描 RxD0 线以接收一个零字节，它由一个起始位，8 个 0 数据位和一个停止位组成。根据零字节的持续时间计算对应于当前 CPU 时钟的波特率因子，并相应初始化串行接口 ASC0，将引脚 TxD0 切换至输出。利用该波特率，BSL 向提供加载数据的主机返回一个 ID 字节。

该 ID 字节指明启动器件的型号。器件代码定义如下：

- 55H: 8xC166
- A5H: C167 先前的版本（停产）
- B5H: C165 先前的版本
- C5H: C167 衍生器件
- D5H: 所有具有 ID 寄存器的器件，包括 XC16x

注：ID 字节 D5H 未直接指明特定的产品型号，相关信息可从 ID 寄存器中获得。

XC164CM 进入 BSL 模式之后，将自动进行以下设置：

寄存器/位域	复位值
看门狗定时器	禁止
P3.10/TxD0	"1"
DP3.10	"1"
ALTSEL0P3.10	"1"
ASC0_BG	XXXXH
ASC0_CON	8811H
GPT12E_T6CON	0880H
GPT12E_T6	XXXXH
PLLCON	2700H

与正常复位不同，BSL 模式下看门狗定时器被禁止，因此引导程序加载序列的操作不受时间限制。引脚 TxD0 设置为输出，从而 XC164CM 可返回 ID 字节。

10.2 加载启动代码

发送 ID 字节之后，BSL 通过 ASC0 接口循环接收数据，每次接收 32 个字节。这些字节依次存入内部 PSRAM 从 E0'0004H 至 E0'0023H 的地址单元中，因此最多可在 PSRAM 区存入 16 条指令。PSRAM 的前两个字中加载了 DISWDT 指令。为了执行加载代码，BSL 使寄存器 VECSEG 指向单元 E0'0000H，即第一条加载的指令¹⁾。引导程序加载器序列由软件复位终止。由于一般的应用程序都可能远远多于 16 条指令，因此，通常由初始的加载程序来再次加载其它代码或数据。第二次接收循环可直接使用经过预先初始化的 ASC0 接口来接收数据，并将数据存入用户指定的任意地址单元。

二级加载代码可能是最终的应用程序代码；也可能是更高级的加载程序，加入传送协议以增强加载代码或数据的完整性；还可能是一段代码，来改变系统配置。

该过程可能会经过多次重复加载，也可能直接执行最终应用程序。

注：不执行从被保护 Flash/ROM 读取数据的操作。

10.3 退出引导程序加载模式

引导程序加载器被激活后，看门狗定时器和调试系统被禁止。从主机接收到第 32 个字节之后、BSL 模式结束时，调试系统被自动释放。若需要激活看门狗定时器，必须执行指令 ENWDT（在执行 EINIT 指令之前）。复位也可重新使能 WDT：

- 软件复位（和外部配置无关）
- 硬件复位，不配置为 BSL 模式

（非 BSL）复位之后，XC164CM 开始执行由 P9 口、P1H 口和 $\overline{\text{TRST}}$ 决定的用户存储器中的程序。

1) 包括执行初始的 DISWDT 指令，确保 2 级加载不被看门狗定时器中止。

10.4 选择 BSL 波特率

串口 ASC0 的波特率通过测量接收到的第一个零字节的时间长度计算得到，从而使 XC164CM 的引导程序加载器工作在很宽的波特率范围内。但是，波特率必须保持在上下限之间以确保正确的数据传送。

$$B_{MC} = \frac{f_{SYS}}{32 \times (ASC0BG + 1)} \quad (10.1)$$

XC164CM 使用定时器 GPT12E_T6 测量起始零字节的长度，该测量的量化不确定性造成和实际波特率的第一次偏差；从定时器的内容计算 ASC0_BG 重载值会造成第二次偏差。方程 (10.2) 给出该关系：

$$ASC0BG = \frac{T6 - 36}{72} \quad T6 = \frac{9}{4} \times \frac{f_{SYS}}{B_{Host}} \quad (10.2)$$

为了确保从主机到 XC164CM 的正确数据传送，ASC0 的内部初始化波特率和主机实际波特率偏差应该低于 2.5%。主机波特率和 XC164CM 波特率的偏差 (F_B ，百分数) 可根据方程 (10.3) 计算：

$$F_B = \left| \frac{B_{Contr} - B_{Host}}{B_{Contr}} \right| \times 100\% \quad F_B \leq 2.5\% \quad (10.3)$$

注：函数 (F_B) 不考虑振荡器和其它支持串口通讯的器件的公差。

该波特率偏差是一个非线性函数，和 CPU 时钟和主机波特率有关。由于较小的波特率预分频因子容易导致较大的量化误差，因此函数 (F_B) 的最大值随主机波特率增加而增大（见图 10-2）。

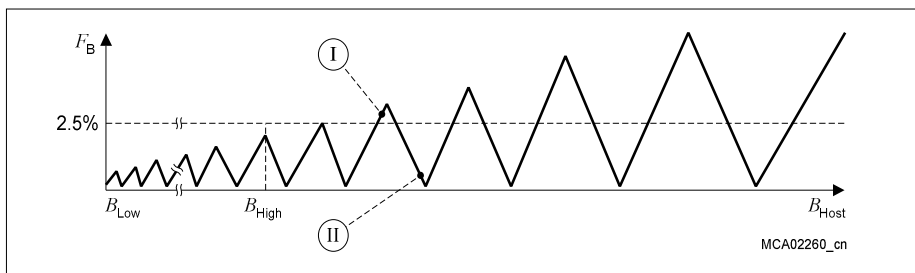


图 10-2 主机和 XC164CM 之间的波特率偏差

测量零字节时，**最小波特率**（图 10-2 中的 B_{Low} ）由定时器 GPT12E_T6 的最大计数能力决定，即取决于系统时钟。最小波特率通过将 GPT12E_T6 的最大计数值 2^{16} 代入波特率公式计算得到。低于 B_{Low} 的波特率将导致 GPT12E_T6 溢出，此时 ASC0 不能被正确初始化，将无法与外部主机进行通讯。

最大波特率（图 10-2 中的 B_{High} ）是未超过偏差极限的最高波特率， B_{Low} 和 B_{High} 之间的所有波特率都低于偏差极限。 B_{High} 表示能与外部主机正常通讯、无需额外测试的最大波特率。

只要实际偏差超过偏差极限，也可采用**更高的波特率**。某个波特率（图 10-2 中标记 I）可能超过偏差极限，而比它更高的波特率（图 10-2 中标记 II）可能低于偏差极限。满足以下三个条件的任何波特率都可被使用：

- 波特率在 ASC0 指定的工作范围内
- 外部主机能够使用该波特率
- 计算偏差低于极限值

表 10-1 引导程序加载器的波特率范围

f_{SYS} [MHz]	10	12	16	20	25	33
B_{MAX}	312,500	375,000	500,000	625,000	781,250	1,031,250
B_{High}	9,600	19,200	19,200	19,200	38,400	38,400
B_{STDmin}	600	600	600	1,200	1,200	1,200
B_{Low}	344	412	550	687	859	1,133

注：进入引导程序加载模式时，缺省配置选择直接驱动模式产生时钟。这种情况下，引导程序加载器以 $f_{SYS} = f_{OSC}$ 工作，对于原先要进行 PLL 操作的低输入频率而言，ASC0 的最大波特率受到限制。

更高级的引导程序序列可将时钟产生模式（通过寄存器 PLLCON）切换至 PLL 模式，以获得更高的程序下载波特率。

11 调试系统

11.1 简介

XC164CM 包含一个片上调试支持（OCDS）系统，通过调试接口引脚由外部设备直接控制，为用户提供了方便的调试功能。

片上调试支持（OCDS）

OCDS 支持各种调试特性，包括设置断点和跟踪存储器位置等。OCDS 的典型应用是在用户系统环境下调试运行在 XC164CM 上的用户软件。

OCDS 系统通过**调试接口**由外部调试设备控制，调试接口包括一个独立的 JTAG 接口和一个断点接口（图 11-1）。调试器通过一组 OCDS 寄存器（可通过 JTAG 接口访问），以及一组特殊调试 IO 指令来管理调试任务。此外，OCDS 系统可以由 CPU 控制（如监控程序）。OCDS 系统与 CPU 内核之间可以通过插入接口进行交互，允许执行 Cerberus 生成的指令，还可以通过断点接口进行交互。

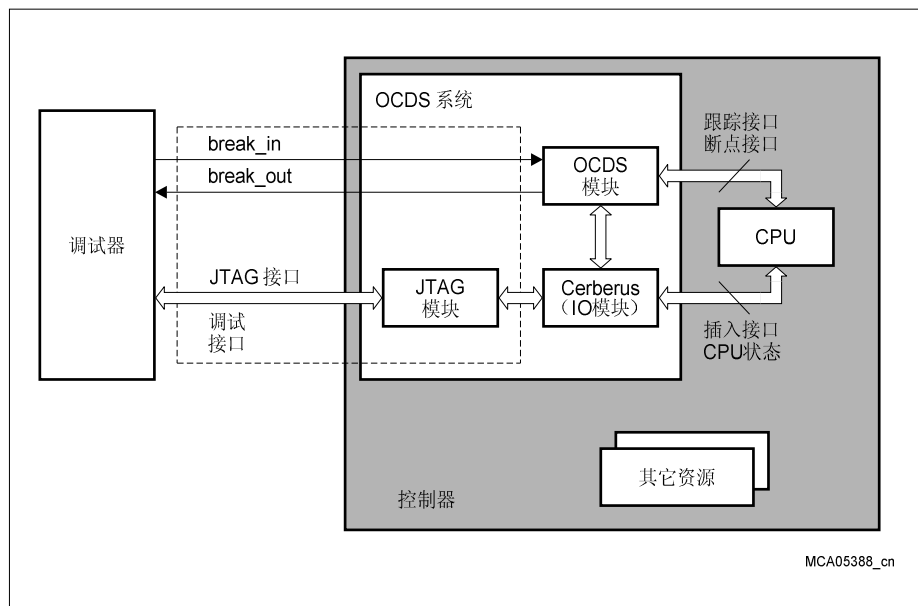


图 11-1 OCDS 总体结构

OCDS 系统功能由**调试接口**、**OCDS 模块**和调试 IO 控制模块（**Cerberus**）组成。调试 IO 控制模块（**Cerberus**）提供调试接口（外部调试器）和内部系统之间交互所必需的全部功能。

OCDS 系统具有以下基本特性：

- 硬件、软件和外部引脚断点
- 对断点做出响应（CPU 暂停、监控程序调用、数据传送和外部信号）
- 对整个地址空间的读/写访问
- 单步调试
- JTAG 接口和断点接口的调试接口引脚
- 插入任意指令
- 通过外部总线传送进行快速存储器跟踪
- 分析和状态寄存器

11.2 调试接口

调试接口是访问 XC164CM 片上调试支持系统（OCDS）资源的通道。通过该调试接口，可以与所有片上、片外的存储器（若存在）和控制寄存器交换数据。

功能和特性

- 用于片上调试支持（OCDS）的独立接口
- 基于 IEEE 1149 JTAG 标准的 JTAG 端口
- 用于外部触发和断点指示的断点接口
- 一般存储器访问功能
- 独立的数据传送通道，比如用来对片上非易失存储器编程

调试接口包括：

- 标准 **JTAG 接口**
- 两个附加的 XC164CM 特定信号– **OCDS 断点接口**

JTAG 接口

JTAG 接口是一个标准化的专用端口，通常用于边界扫描和芯片内部测试。由于这两种应用在器件正常工作期间都被禁止，因此 JTAG 端口是用来进行调试的理想接口。

该接口具有符合 JTAG IEEE.1149 标准的信号：

- **TDI** - 串行数据输入
- **TDO** - 串行数据输出
- **TCK** - JTAG 时钟
- **TMS** - 状态机控制信号
- $\overline{\text{TRST}}$ 复位/模块使能

OCDS 断点接口

在调试器和 XC164CM **OCDS 模块**之间，使用两个附加信号实现了一个直接的异步断点通道：

- $\overline{\text{BRKIN}}$ （断点输入请求）可使调试器异步中断 CPU，并强制 CPU 进入一个预定义的状态/动作。
- $\overline{\text{BRKOUT}}$ （断点输出信号）可以由 OCDS 激活，用于通知外部环境某些预定义的调试事件已经发生，但它不中断 CPU。

11.3 OCDS 模块

OCDS 模块的应用是在用户系统环境下调试运行在 CPU 上的用户软件。这利用外部调试器实现，该调试器通过独立的**调试接口**来控制**OCDS 模块**。

特性

- 硬件、软件和外部引脚断点
- 最多 4 个指令指针断点
- 用于硬件断点的屏蔽比较
- 还可通过监控程序来配置 OCDS
- 支持多 CPU/主机系统
- 通过监控程序或 CPU 暂停进行单步调试
- 暂停模式下，PC 可见（通过 **Cerberus** 插入 IO_READ_IP 指令）

基本概念

片上调试的概念分为两个部分。第一部分包含调试事件的产生，第二部分定义产生调试事件时采取何种动作。

- 调试事件：
 - **硬件断点**
 - **SBRK 指令**的解码
 - 激活**断点输入引脚**
- 调试事件动作：
 - **CPU 暂停模式**
 - **调用监控程序**
 - **触发传送**
 - **激活外部引脚**输出

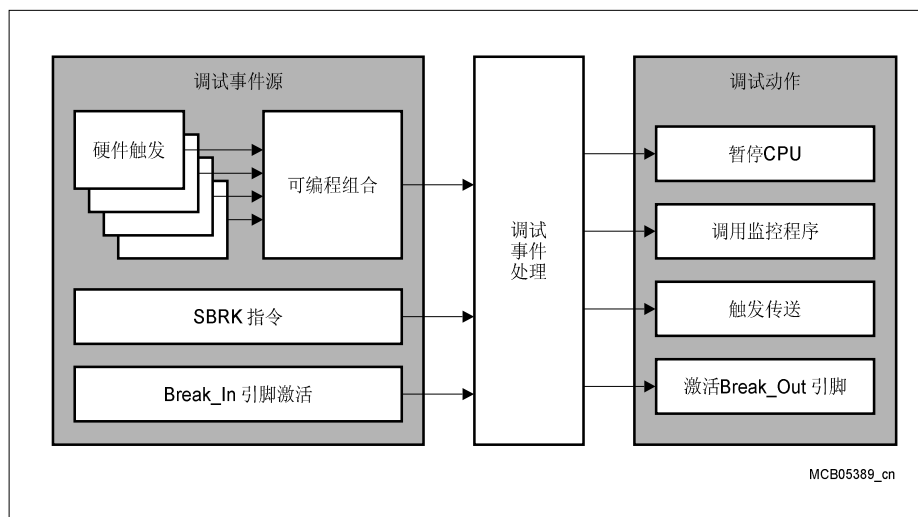


图 11-2 OCDS 概念：框图

11.3.1 调试事件

调试事件可以有几种不同的来源。

硬件断点

硬件断点是一种调试事件，当单个触发信号或多个触发信号的组合与编程设定的条件匹配时，产生调试事件。

可以使用如下的硬件触发源：

表 11-1 硬件触发

触发源	大小
任务标识符	16 位
指令指针	24 位
读数据地址（两个总线被监控）	2 × 24 位
写数据地址	24 位
数据值（读或写）	16 位

SBRK 指令

可以通过 **SBRK** 指令产生一个调试事件。比如，调试器可以利用它在 **RAM** 中临时产生插入码，以此实现**软件断点**。

SBRK（软件断点）指令的操作码定义为 **8C00_H**。当该指令被解码，并进入执行阶段时，整个流水线被取消，包括 **SBRK** 指令本身。因此，实际上 **SBRK** 指令从不被自己“执行”。

下一步的操作取决于 **OCDS** 的设置：

- 如果 **OCDS** 被使能、软件断点也被使能，则 **CPU** 进入**暂停模式**
- 如果 **OCDS** 被禁止，或者软件断点被禁止，则执行**软件断点强制中断**（**SBRKTRAP**），该强制中断是 **A** 类强制中断，中断编号 **08_H**

断点引脚输入

外部的调试断点引脚（**BRKIN**）可使调试器异步中断处理器。

11.3.2 调试动作

当 OCDS 使能、并产生调试事件时，会发生下列动作：

触发传送

调试事件产生时可能发生的一个动作是触发 **Cerberus**：

- 执行数据传送 — 用于在不能被中断的重要子程序中进行存储器数据传送
- 向内核插入一条指令 — 利用该机制，可以将任意一条指令插入 XC164CM 流水线

暂停模式

该动作发生后，OCDS 模块向内核发送一个断点请求。

如果 OCDS 断点优先级高于当前 CPU 优先级，则内核接受该断点请求；如果断点请求被接收，系统进入挂起状态，指令流暂停。

暂停模式仍然可以被更高优先级的用户中断所中断。如果发生这种情况，外部调试系统可以通过调试接口进行读取和更新操作来监控目标系统。

调用监控程序

调试事件发生时可能采取的一个动作是调用监控程序。

快速转入监控程序使得能够定义灵活的调试环境，可以满足调试实时系统的很多需求。通常情况下监控程序具有最高优先级，不能被任何请求源中断。

还有一种可被中断的监控程序。这种情况下，当监控程序（调试器）工作时，仍然可以执行保障安全的重要代码，这给用户带来了极大的灵活性。

激活外部引脚

该动作激活 **OCDS 断点接口** 的外部引脚 $\overline{\text{BRKOUT}}$ 。在不能被中断的重要子程序中，通过外部引脚 **BRKOUT** 向外部环境发出一个信号，指示已经发生了一个特殊的事件。该特性还可用于内部调试硬件与外部调试硬件之间的同步。

11.4 Cerberus

Cerberus 模块提供和控制外部调试器（通过 **调试接口**）、**OCDS 模块**、XC164CM 内部系统之间进行交互所必需的所有操作。

特性

- JTAG 接口用作控制和数据通道
- 一般存储器读/写功能（RW 模式），可以访问整个地址空间
- 读/写通用寄存器（GPR）
- 插入任意指令
- 外部主机控制所有任务
- 在正常运行和暂停模式下，可以获得所有任务的信息
- 任务的优先级可配置
- 完全支持监控程序和外部主机（调试器）之间的通信
- 可选择错误保护
- 内部总线锁定状态的分析寄存器

Cerberus 的目标应用是将 JTAG 接口作为独立的端口，用于片上调试支持（OCDS）。外部调试器通过插入机制能够访问 OCDS 寄存器和任意存储器地址。

11.4.1 功能概述

Cerberus 通过 **JTAG 接口** 由外部调试器控制。调试器提供 **Cerberus IO 指令**，并执行双向数据传送。

Cerberus 有两种主要的工作模式：

读/写操作模式

读/写（RW）模式是 Cerberus 最典型的操作模式。该模式用于读写存储器地址或插入指令。该模式下，内核的插入接口被激活。

该模式下，外部调试器（主机）利用 JTAG 接口可以进行下列操作：

- 从目标系统读写存储器地址（数据传送）
- 插入将由内核执行的任意指令

所有的 **Cerberus IO 指令** 都可以在 RW 模式下使用。在 RW 模式下，专用的 **IO_READ_IP** 指令用于读取在断点状态下 CPU 的 IP 值。

利用插入指令可以访问任何存储器地址（如同 PEC 传送）。通用情况下，可以使用下面的 **Cerberus IO 指令**：

- **IO_READ_WORD, IO_WRITE_WORD**

- **IO_READ_BLOCK, IO_WRITE_BLOCK**
- **IO_WRITE_BYTE**

在执行这些指令期间，主机向/从指定的寄存器/存储单元写入/读出数据，而 Cerberus 本身执行其余的工作：通过向内核插入合适的指令来执行 PEC 传送。

通信操作模式

这种模式下，外部主机（调试器）与运行在 CPU 上的监控程序进行通信。数据传输通过 PDBus+ 寄存器来实现。外部主机是所有任务的发起者，请求监控程序读写数据。

和**读/写操作模式**不同，该模式下 Cerberus 不执行读写请求，它将置位 CPU 可访问寄存器中的请求位，通知监控程序主机想要发送（**IO_WRITE_WORD**）或接收（**IO_READ_WORD**）数据。监控程序必须查询该状态寄存器，相应执行正确的动作。

通信模式是复位后的缺省模式。在通信模式下，只有 **IO_WRITE_WORD** 和 **IO_READ_WORD** 可用。

主机和监控程序直接通过专用数据寄存器来交换数据。为了实现主机（调试器）和监控程序的同步访问，在 Cerberus 状态寄存器中有相关的控制位。

12 指令集概述

本章简要总结 XC164CM 的指令集，按照指令的类型进行分类。本章可使读者基本上了解功能强大、种类繁多的 XC164CM 指令集及其一般使用。

XC166 系列的“**指令集手册**”给出每条指令的**详细描述**，包括操作数类型、状态标志设置、寻址模式、指令长度（字节数）和目标代码格式等；还提供了按照不同准则归类的指令表，便于用户快速查询。

指令类型总结

将指令按照类型分组可以帮助用户区分相似的指令（例如 SHR，ROR），以及特定指令的变体（例如 ADD，ADDB）。便于用户查询 XC164CM 的指令。

注：所使用的助记符参见指令详细描述（指令集手册）。

表 12-1 算术运算指令

两个字或字节加法	ADD	ADDB
带进位的两个字或字节加法	ADDC	ADDCB
两个字或字节减法	SUB	SUBB
带借位的两个字或字节减法	SUBC	SUBCB
16×16 位有符号或无符号数乘法	MUL	MULU
16/16 位有符号或无符号数除法	DIV	DIVU
32/16 位有符号或无符号数除法	DIVL	DIVLU
字或字节的 1 补码	CPL	CPLB
字或字节的 2 补码（负）	NEG	NEGB

表 12-2 逻辑运算指令

两个字或字节按位与	AND	ANDB
两个字或字节按位或	OR	ORB
两个字或字节按位异或	XOR	XORB

表 12-3 比较和循环控制指令

比较两个字或字节	CMP	CMPB
两字比较后，自行增加 1 或 2	CMPI1	CMPI2
两字比较后，自行减少 1 或 2	CMPD1	CMPD2

表 12-4 布尔位操作指令

对字的高字节或低字节中的可屏蔽位域操作	BFLDH	BFLDL
对某位置 “1”	BSET	-
对某位清 “0”	BCLR	-
转移某一位	BMOV	-
对某位取反后转移	BMOVN	-
两位与操作	BAND	-
两位或操作	BOR	-
两位异或操作	BXOR	-
两位比较	BCMP	-

表 12-5 移位和循环移位指令

字右移	SHR	-
字左移	SHL	-
字右环移	ROR	-
字左环移	ROL	-
字算术右移（符号位移位）	ASHR	-

表 12-6 优先化指令

确定归一化字操作数所需的移位周期数（支持浮点数）	PRIOR	-
--------------------------	-------	---

表 12-7 数据转移指令

字或字节的标准数据转移	MOV	MOVB
将字节数据转移到字地址，进行符号或零字节扩展	MOVBS	MOVVBZ

注：数据转移指令可以和很多不同的寻址方式一起使用，包括间接寻址和自动指针增加/减少。

表 12-8 系统堆栈指令

将一个字压入系统堆栈	PUSH	-
将一个字从系统堆栈中弹出	POP	-
将一个字存入系统堆栈，然后用新值刷新旧值（用于寄存器组切换）	SCXT	-

表 12-9 跳转指令

在当前程序段内，有条件跳转至绝对、间接或相对寻址的目标指令	JMPA	JMPI	JMPR
在任意程序段内，无条件跳转至绝对寻址的目标指令	JMPS	-	-
在当前程序段内，根据可选位的状态，有条件跳转至相对寻址的目标指令	JB	JNB	-
在当前程序段内，根据可选位的状态，有条件跳转至相对寻址的目标指令，跳转后对测试位取反（支持旗语操作）	JBC	JNBC	-

表 12-10 调用指令

在当前程序段内，有条件调用绝对或间接寻址的子程序	CALLA	CALLI
在当前程序段内，无条件调用相对寻址的子程序	CALLR	-
在任意程序段内，无条件调用绝对寻址的子程序	CALLS	-
在当前程序段内，无条件调用绝对寻址的子程序，并将选择的寄存器压入系统堆栈	PCALL	-
无条件跳转至代码段<VECSEG>内的中断或强制中断向量跳转表	TRAP	-

表 12-11 返回指令

在当前程序段内，从子程序返回	RET	-
在任何程序段内，从子程序返回	RETS	-
在当前程序段内，从子程序返回，并将选择的寄存器从系统堆栈弹出	RETP	-
从中断服务程序返回	RETI	-

表 12-12 系统控制指令

软件复位 XC164CM	SRST	-
进入空闲或休眠模式	IDLE	-
进入掉电模式	PWRDN	-
服务看门狗定时器	SRVWDT	-
禁止看门狗定时器	DISWDT	-
使能看门狗定时器（只能在 WDT 增强模式下执行）	ENWDT	-
指示初始化子程序结束（在 WDT 兼容模式下，初始化之后，禁止执行 DISWDT 指令）	EINIT	-

表 12-13 其它指令

空操作，需要 2 个字节的存储空间和最短执行时间	NOP	-
定义不可分的指令序列	ATOMIC	-
将"reg"、"bitoff"、"bitaddr" 寻址模式切换至扩展 SFR 区	EXTR	-
直接选择特定的数据页（代替 DPP）替代 DPP 寻址模式，可选择性的切换至 ESFR 区	EXTP	EXTPR
直接选择特定的段（代替 DPP）替代 DPP 寻址模式，可选择性的切换至 ESFR 区	EXTS	EXTSR

*注：ATOMIC 和 EXT*指令支持不可中断的代码序列，例如用于旗语操作。还支持超过当前 DPP 极限（ATOMIC 除外）的数据寻址，该特性对于利用高级语言对具有较大存储器模型的编程非常有用。*

表 12-14 MAC 单元指令

乘（和累加）	CoMUL	CoMAC
加/减	CoADD	CoSUB
右移/左移	CoSHR	CoSHL
算术右移	CoASHR	-
加载累加器	CoLOAD	-
存储 MAC 寄存器	CoSTORE	-
比较值	CoCMP	-
最小/最大	CoMIN	CoMAX
绝对值	CoABS	-
舍入	CoRND	-
数据转移	CoMOV	-
累加器取负	CoNEG	-
空操作	CoNOP	-

受保护指令

XC164CM 指令集中，有一些指令对于控制微控制器的功能十分重要，这些指令被称为受保护指令。这些指令使用最长的 **32 位** 指令格式，而常规指令仅使用最长指令格式的一部分（比如低 **8 位**），其它位用来提供附加信息，如所用到的寄存器。在取指期间数据可能被破坏，使用 **32 位** 受保护双字指令可以增强数据安全性。重要的操作，如软件复位，只有在指令解码没有任何错误的情况下才能执行。这增强了微控制器系统的安全性和可靠性。

13 器件规范

器件规范描述了器件的电气参数，给出器件的直流特性（如输入、输出、电源或电流）和交流特性（如时序特性和时序要求）。

这些直流和交流特性不仅受 XC164CM 内核架构、指令集、内核基本功能以及片上外设的影响、而且会随器件改进、或者标准器件各衍生产品的不同而改变。

本用户手册没有给出这些电气参数，如有需要，请参阅数据手册（不断更新中）。

所有的电气参数都可从最新版的相关数据手册中查阅。

注：在开始新的设计之前，必须先核实器件的特性，从而确保使用最新信息。

图 13-1 所示的 XC164CM 引脚框图给出电源和 IO 引脚的位置排列。所有引脚的详细描述请参阅数据手册。

*注：并非所有的衍生产品都能够支持**图 13-1** 给出的所有复用功能。具体描述请参阅相关产品的数据手册。*

关键字索引

用户可通过关键字方便的查阅其对应 **XC164CM** 架构、功能单元或模块功能的详细描述。检索关键字可帮助用户快速找到 **XC164CM** 相关问题的答案。

本用户手册分为两卷：“系统单元”和“外设单元”。为了便于用户的使用，关键字索引（以及目录）将两卷中的关键字（以及章节）统一列出，以便用户快速查阅相关内容（卷[1]或卷[2]）。

A

Acronyms（缩写），1-10

Adapt Mode（自适应模式），6-11

ADC（模数转换器），2-22, 16-1

ADC_CIC, 16-23

ADC_CON, 16-4

ADC_CON1, 16-5

ADC_CTR0, 16-7

ADC_CTR2, 16-8

ADC_CTR2IN, 16-8

ADC_EIC, 16-23

Address（地址）

Boundaries（边界），3-15

Mapping（映射），3-3

Addressing Modes（寻址模式）

CoREG Addressing Mode（CoREG 寻址模式），4-55

DSP Addressing Modes（DSP 寻址模式），4-51

Indirect Addressing Modes（间接寻址模式），4-49

Long Addressing Modes（长寻址模式），4-45

Short Addressing Modes（短寻址模式），4-43

Alternate Port Functions（复用端口功能），7-7

ALU（算术逻辑运算单元），4-62

Analog/Digital Converter（模数转换器），16-1

Arbitration of conversions（转换仲裁），16-18

ASC（异步/同步串行接口），19-1

ASCx_EIC, 19-37

ASCx_RIC, 19-37

ASCx_TBIC, 19-37

ASCx_TIC, 19-37

Autobaud Detection（自动波特率检测），19-27

Error Detection（错误检测），19-35

Features and Functions（特性和功能），19-1

IrDA Frames（IrDA 帧），19-8

Register（寄存器）

BG, 19-22

RBUF, 19-12, 19-20

TBUF, 19-9, 19-20

Transmit FIFO（发送 FIFO），19-9

ASCx_BG, 19-45

ASCx_CON, 19-42

ASCx_FDV, 19-46

Auto Scan conversion（自动扫描转换），16-14

Autobaud Detection（自动波特率检测），19-27

B

Baudrate（波特率）

ASC0（异步/同步串行接口 0），19-22

Bootstrap Loader（引导程序加载器），10-5

CAN（控制器局域网），21-55

Bit（位）

Handling（处理），4-65

Manipulation Instructions（操作指令），12-2

protected（保护），2-31, 4-65

reserved（保留），2-15

Block Diagram ITC/PEC（中断和 PEC 控制器框图），5-3

BNKSELx, 5-33

Bootstrap Loader（引导程序加载器），6-11, 10-1

Boundaries（边界），3-15

Bus（总线）

ASC（异步/同步串行接口），19-1

CAN（控制器局域网），2-25

SSC（高速同步串行接口），20-1

C

Calibration（校准），16-19

CAN（控制器局域网）

Acceptance filtering（验收滤波），21-16

analysing mode（分析模式），21-7

arbitration（仲裁），21-16

baudrate（波特率），21-55

bit timing（位定时），21-9, 21-55

bus off（总线关闭）

recovery sequence（恢复序列），21-4

status bit（状态位），21-50

- error counters（错误计数器），21-53
- error handling（错误处理），21-11
- error warning level（错误警告级别），21-53
- frame counter/time stamp（帧计数器/时间戳），21-57
- interface（接口），2-25
- single data transfer（单次数据传送），21-22
- CAPCOM12（捕获比较单元 1/2）
 - Capture Mode（捕获模式），17-13
 - Counter Mode（计数器模式），17-8
- CAPCOM2（捕获比较单元 2），2-16
- CAPCOM6（捕获比较单元 6），2-18
- CAPREL, 14-58
- Capture Mode（捕获模式）
 - GPT1（通用定时器单元 1），14-27
 - GPT2（CAPREL），14-50
- Capture/Compare Registers（捕获/比较寄存器），17-10
- CC1_SEE, 17-27
- CC2_DRM, 17-23
- CC2_IOC, 17-29
- CC2_M4-7, 17-10
- CC2_OUT, 17-25
- CC2_SEE, 17-27
- CC2_SEM, 17-27
- CC2_T78CON, 17-5
- CC2_T7IC, 17-9
- CC2_T8IC, 17-9
- CC63R, 18-43
- CC63SR, 18-43
- CC6xR, 18-19
- CC6xSR, 18-20
- CCU6_xIC, 18-84
- CCxIC, 17-34
- Clock（时钟）
 - generation（产生），2-28
 - output signal（输出信号），6-25
- CMPMODIF, 18-50
- CMPSTAT, 18-49
- Command sequences（命令序列），3-19
- Concatenation of Timers（定时器级联），14-23, 14-49
- Configuration（配置）
 - Reset（复位），6-10
- Context（上下文）
 - Pointer Updating（指针更新），4-36
 - Switch（切换），4-35

Switching（切换），5-32
Conversion（转换）
 analog/digital（模拟/数字），16-1
 Arbitration（仲裁），16-18
 Auto Scan（自动扫描），16-14
 timing control（时序控制），16-20
Count direction（计数方向），14-6, 14-37
Counter Mode（计数器模式）（GPT1），14-10, 14-41
Counter（计数器），14-20, 14-47
CP, 4-38
CPU（中央处理器），2-2, 4-1
CPUCON1, 4-26
CPUCON2, 4-27
CRIC, 14-59
CSP, 4-41

D

Data Management Unit（数据管理单元），2-10
Data Page（数据页），4-46
 boundaries（边界），3-15
Data SRAM（数据 SRAM），3-10
Development Support（开发支持），1-8
Direction（方向）
 count（计数），14-6, 14-37
Disable（禁止）
 Interrupt（中断），5-29
Division（除法），4-66
Double-Register Compare（双寄存器比较），17-22
DP1H, 7-9
DP1L, 7-9
DP3, 7-19
DP9, 7-42, 21-86
DPP, 4-46
Driver characteristic (ports)（驱动器特性（端口）），7-4
DSTPx, 5-23
Dual-Port RAM（双端口 RAM），3-10

E

EBC（外部总线控制器）
 Memory Table（存储器表），9-10
Edge characteristic(ports)（边沿特性（端口）），7-5

EMUCON, 6-35
Enable（使能）
 Interrupt（中断），5-29
End of PEC Interrupt Sub Node（PEC 结束中断子节点），5-28
EOPIC, 5-27
Erase command（擦除命令），3-21
Error correction（纠错），3-25
Error Detection（错误检测）
 ASC（异步/同步串行接口），19-35
 SSC（高速同步串行接口），20-15
EXICON, 5-37
EXISEL0, 5-38
EXISEL1, 5-38
External（外部）
 Fast interrupts（快速中断），5-37
 Interrupt pulses（中断脉冲），5-40
 Interrupt source control（中断源控制），5-37
 Interrupts during sleep mode（休眠模式期间的中断），5-39
 Interrupts（中断），5-35

F

Fast external interrupts（快速外部中断），5-37
FINT0ADDR, 5-16
FINT0CSP, 5-16
FINT1ADDR, 5-16
FINT1CSP, 5-17
Flags（标志位），4-61–4-64
Flash（闪存）
 command sequences（命令序列），3-19
 memory mapping（存储器映射），3-16
 memory（存储器），3-12
 waitstates（等待状态），3-41
FOCON, 6-26
Frequency（频率）
 output signal（输出信号），6-25
FSR, 3-33

G

Gated timer mode（门控定时器模式）（GPT1），14-9
Gated timer mode（门控定时器模式）（GPT2），14-40
GPR（通用寄存器），3-7

GPT（通用定时器），2-19
GPT1（通用定时器单元 1），14-2
GPT12E_CAPREL, 14-58
GPT12E_CRIC, 14-59
GPT12E_T2, 14-31
GPT12E_T2CON, 14-15
GPT12E_T2IC, 14-32
GPT12E_T3, 14-31
GPT12E_T3CON, 14-4
GPT12E_T3IC, 14-32
GPT12E_T4, 14-31
GPT12E_T4CON, 14-15
GPT12E_T4IC, 14-32
GPT12E_T5, 14-58
GPT12E_T5CON, 14-42
GPT12E_T5IC, 14-59
GPT12E_T6, 14-58
GPT12E_T6CON, 14-35
GPT12E_T6IC, 14-59
GPT2（通用定时器单元 2），14-33

H

Hardware（硬件）
Traps（强制中断），5-43

I

IDCHIP, 6-51
Idle Mode（空闲模式），6-41
IDMANUF, 6-51
IDMEM, 6-52
IDPROG, 6-52
IDX0, 4-51
IDX1, 4-51
IEN, 18-82
IMB（内部程序存储模块）
 block diagram（框图），3-39
 control functions（功能），3-42
 memories（存储器）
 address map（地址映射），3-39
 wait states（等待状态），3-42
IMBCTR, 3-42
Incremental Interface Mode（增量接口模式）（GPT1），14-11
INP, 18-83

Instruction（指令），12-1
 Bit Manipulation（位操作），12-2
 Pipeline（流水线），4-11
 protected（保护），12-6
Interface（接口）
 ASC（异步/同步串行接口），19-1
 CAN（控制器局域网），2-25
 External Bus（外部总线），9-1
 SSC（高速同步串行接口），20-1
Interrupt Handling（中断处理）
 CAN transfer（CAN 传送），21-5
Interrupt（中断）
 Arbitration（仲裁），5-4
 during sleep mode（休眠模式期间），5-39
 Enable/Disable（使能/禁止），5-29
 External（外部），5-35
 Fast external（快速外部），5-37
 input timing（输入时序），5-40
 Jump Table Cache（跳转表缓存），5-16
 Latency（延迟），5-41
 Node Sharing（节点共享），5-34
 Priority（优先级），5-7
 Processing（处理），5-1
 RTC（实时时钟），15-11
 source control（请求源控制），5-37
 Source（请求源），5-11
 System（系统），2-9, 5-2
 Vectors（向量），5-11
IP, 4-41
IrDA Frames ASC（IrDA 帧 ASC），19-8
IS, 18-77
ISR, 18-81
ISS, 18-80

L

Latency（延迟）
 Interrupt, PEC（中断，外围事件控制器），5-41

M

MAH, 4-73
MAL, 4-73
用户手册

MAR, 3-27
Margin check（裕量检测）, 3-27
MCMCTR, 18-64
MCMOUT, 18-62
MCMOUTS, 18-61
MCW, 4-70
MDC, 4-67
MDH, 4-66
MDL, 4-67
Memory（存储器）, 2-11
 Areas(Data)（区域（数据））, 3-9
 Areas(Program)（区域（程序））, 3-11
 DPRAM（双端口 RAM）, 3-10
 DSRAM（数据 SRAM）, 3-10
 Flash（闪存）, 3-12
 Program Flash（程序闪存）, 3-16
 PSRAM, 3-12
MODCTR, 18-53
MRW, 4-76
MSW, 4-74
Multiplication（乘法）, 4-66

N

NMI（非屏蔽中断）, 5-1, 5-48
Noise filter(Ext.Interrupts)（噪声滤波（外部中断））, 5-40

O

OCDS（片上调试支持）
 Requests（请求）, 5-41
ODP3, 7-20
ODP9, 7-43
ONES, 4-78
Open Drain Mode（漏极开路模式）, 7-3
OPSEN, 6-36
Oscillator（振荡器）
 circuitry（电路）, 6-14
 measurement（测量）, 6-14
 Watchdog（看门狗）, 6-24

P

P1H, 7-8

P1L, 7-8
P3, 7-19
P5, 7-37
P9, 7-42
PEC pointer（PEC 指针）, 3-8
PEC（外围事件控制器）, 2-11, 5-18
 Latency（延迟）, 5-41
 Transfer Count（传送计数）, 5-19
PECCx, 5-19
PECISNC, 5-27
PECSEGx, 5-24
Peripheral（外设）
 Event Controller（事件控制器）, 5-18
 Register Set（寄存器组）, 22-1
 Summary（归纳）, 2-14
Phase Locked Loop（锁相环）, 6-13
PICON, 7-2
Pins（引脚）, 8-1
Pipeline（流水线）, 4-11
PLL（锁相环）, 6-13
PLL_IC, 6-24
PLLCON, 6-18
POCON*, 7-6
Port（端口）, 2-27
Ports（端口）
 Alternate Port Functions（复用端口功能）, 7-7
 Driver characteristic（驱动器特性）, 7-4
 Edge characteristic（边沿特性）, 7-5
Power Management（功率管理）, 2-29
PROCON, 3-29
Program Management Unit（程序管理单元）, 2-10
Programming command（编程命令）, 3-21
Protected（保护）
 Bits（位）, 2-31, 4-65
 features（特性）, 3-28
 Instruction（指令）, 12-6
Protection（保护）
 Commands（命令）, 3-23
PSLR, 18-73
PSW, 4-61

Q

QR0, 4-50
QR1, 4-50
QX0, 4-52
QX1, 4-52

R

RAM
 data SRAM（数据 SRAM）, 3-10
 dual ported（双端口）, 3-10
 program/data（程序/数据）, 3-12
 status after reset（复位后的状态）, 6-7
Real Time Clock（实时时钟）, 2-21, 15-1
Register Areas（寄存器区）, 3-5
Register map（寄存器映射）
 TwinCAN module（TwinCAN 模块）, 21-46
Register Table（寄存器表）
 LXBUS peripherals（LXBUS 外设）, 22-13
 PD+BUS peripherals（PD+BUS 外设）, 22-1
RELH, 15-8
RELL, 15-8
Reserved bits（保留位）, 2-15
Reset（复位）, 6-2
 Configuration（配置）, 6-10
 Source indication（触发源指示）, 6-33
 Values（值）, 6-5
RSTCON, 6-12
RTC（实时时钟）, 2-21
RTC（实时时钟）, 15-1
RTC_CON, 15-5
RTC_IC, 15-12
RTC_ISNC, 15-12
RTCH, 15-7
RTCL, 15-7

S

SCUSLC, 6-39
SCUSLS, 6-38
Security（安全）
 features（特性）, 3-28
Segment（段）

- bounderies（边界），3-15
- Segmentation（分段），4-40
- Self-calibration（自校准），16-19
- Serial Interface（串行接口），2-23, 2-24
 - ASC（异步/同步串行接口），19-1
 - Asynchronous（异步），19-5
 - CAN（控制器局域网），2-25
 - SSC（高速同步串行接口），20-1
 - Synchronous（同步），19-19
- SFR（特殊功能寄存器），3-5
- Sharing（共享）
 - Interrupt Nodes（中断节点），5-34
- Sleep mode（休眠模式），6-43
- Software（软件）
 - Traps（强制中断），5-43
- Source（源）
 - Interrupt（中断），5-11
 - Reset（复位），6-33
- SP, 4-58
- SPSEG, 4-58
- SRAM
 - Data（数据），3-10
- SRCPx, 5-23
- SSC（高速同步串行接口），20-1
 - Baudrate generation（波特率产生），20-13
 - Block diagram（框图），20-3
 - Continuous transfer operation（连续传送操作），20-13
 - Error detection（错误检测），20-15
 - Full duplex operation（全双工操作），20-9
 - General operation（操作概述），20-1
 - Half duplex operation（半双工操作），20-11
 - Interrupts（中断），20-15
- SSCx_CON, 20-4, 20-5
- Stack（堆栈），3-13, 4-57
- Startup Configuration（启动配置），6-10
- STKOV, 4-60
- STKUN, 4-60
- SYSCON0, 6-29
- SYSCON1, 6-30
- SYSCON3, 6-44
- SYSSTAT, 6-32

T

T12, 18-6
T12DTC, 18-24
T12MSEL, 18-21
T12PR, 18-6
T13, 18-33
T13PR, 18-33
T2, 14-31
T2CON, 14-15
T2IC, 14-32
T3, 14-31
T3CON, 14-4
T3IC, 14-32
T4, 14-31
T4CON, 14-15
T4IC, 14-32
T5, 14-58
T5CON, 14-42
T5IC, 14-59
T6, 14-58
T6CON, 14-35
T6IC, 14-59
T7IC, 17-9
T8IC, 17-9
TCTR0, 18-44
TCTR2, 18-46
TCTR4, 18-47
TFR, 5-45
Timer（定时器）, 14-2, 14-33
 Auxiliary Timer（辅助定时器）, 14-15, 14-42
 Concatenation（级联）, 14-23, 14-49
 Core Timer（核心定时器）, 14-4, 14-35
 Counter Mode（计数器模式）（GPT1）, 14-10, 14-41
 Gated Mode（门控模式）（GPT1）, 14-9
 Gated Mode（门控模式）（GPT2）, 14-40
 Incremental Interface Mode（增量接口模式）（GPT1）, 14-11
 Mode（模式）（GPT1）, 14-8
 Mode（模式）（GPT2）, 14-39
Tools（工具）, 1-8
Transmit FIFO（发送 FIFO）, 19-9
Traps（强制中断）, 5-43
TRPCTR, 18-67
TwinCAN
 FIFO（先入先出）

- Base object（基本对象），21-23
- Circular buffer（环形缓存），21-25
- configuration（配置），21-74
- for CAN messages（CAN 报文），21-23
- Gateway control（网关控制），21-74
- Slave objects（从属对象），21-25
- Frames（帧）
 - Counter（计数器），21-8
 - Handling（处理），21-17
- gateway（网关）
 - configuration（配置），21-74
- Gateway（网关）
 - Normal mode（正常模式），21-28
 - Shared mode（共享模式），21-35
 - With FIFO（带 FIFO），21-32
- Initialization（初始化），21-40
- Interrupts（中断）
 - Indication（指示）/ INTID, 21-13, 21-52
 - Node pointer/request compressor（节点指针/请求压缩），21-5
- Loop-back mode（回环模式），21-44
- Message handling（报文处理），21-15
 - FIFO（先入先出），21-23
 - Gateway + FIFO（网关+FIFO），21-32
 - Gateway overview（网关概述），21-27
 - Normal gateway（正常网关），21-28
 - Shared gateway（共享网关），21-35
 - Transfer control（传送控制），21-41
- Message interrupts（报文中断），21-13
- Message object（报文对象）
 - Configuration（配置），21-71
 - Control bits（控制位），21-67
 - Interrupt indication（中断指示），21-13
 - Interrupts（中断），21-13
 - Register description（寄存器描述），21-63
 - Transfer handling（传送处理），21-17
- Node control（节点控制），21-7
- Node interrupts（节点中断），21-12
- Node selection（节点选择），21-71
- Overview（概述），21-1
- register map（寄存器映射），21-46
- Registers（global）寄存器（全局）

- Receive interrupt pending（接收中断挂起），21-80
- Transmit interrupt pending（发送中断挂起），21-81
- Registers（message specific）寄存器（报文专用）
 - Acceptance mask（验收屏蔽），21-66
 - Arbitration（identifier）仲裁（标识符），21-65
 - Configuration（配置），21-71
 - Control（控制），21-67
 - Data（数据），21-64
- Registers（node specific）寄存器（节点专用）
 - Bit timing（位定时），21-55
 - Frame counter（帧计数器），21-57
 - Global interrupt node pointer（全局中断节点指针），21-59
 - Interrupt pending（中断挂起），21-52
 - INTID mask（屏蔽），21-61
 - Status（状态），21-50
- Single transmission（单次发送），21-45
- single-shot mode（单次模式），21-22
- Transfer interrupts（传送中断），21-6
- TwinCAN registers（short names）TwinCAN 寄存器（缩写）
 - ABTRH, 21-55
 - ABTRL, 21-55
 - ACR, 21-48
 - AECNTH, 21-53
 - AECNTL, 21-53
 - AFCRH, 21-57
 - AFCRL, 21-57
 - AGINP, 21-59
 - AIMR4, 21-62
 - AIMRH0, 21-61
 - AIMRL0, 21-61
 - AIR, 21-52
 - ASR, 21-50
 - BBTRH, 21-55
 - BBTRL, 21-55
 - BCR, 21-48
 - BECNTH, 21-53
 - BECNTL, 21-53
 - BFCRH, 21-57
 - BFCRL, 21-57
 - BGINP, 21-59
 - BIMR4, 21-62
 - BIMRH0, 21-61
 - BIMRL0, 21-61
 - BIR, 21-52

BSR, 21-50
MSGAMRHn, 21-66
MSGAMRLn, 21-66
MSGARHn, 21-65
MSGARLn, 21-65
MSGCFGHn, 21-71
MSGCFGLn, 21-71
MSGCTRHn, 21-67
MSGCTRLn, 21-67
MSGDRHn0, 21-63
MSGDRHn4, 21-64
MSGDRLn0, 21-63
MSGDRLn4, 21-64
MSGFGCRHn, 21-74
MSGFGCRLn, 21-74
RXIPNDH, 21-80
RXIPNDL, 21-80
TXIPNDH, 21-81
TXIPNDL, 21-81

V

VECSEG, 5-10

W

Waitstates（等待状态）
Flash（闪存）, 3-41
Watchdog（看门狗）, 2-26, 6-46
after reset（复位后）, 6-7
Oscillator（振荡器）, 6-24
WDT（看门狗定时器）, 6-47
WDTCON, 6-49

Z

ZEROS, 4-78

<http://www.infineon.com>