MC9S08JS16

参考手册

苏州大学飞思卡尔嵌入式系统研发中心翻译

http://sumcu.suda.edu.cn

2009 年 11 月

MC9S08JS16 系列特点

8 位 HCS08 中央处理单元（CPU）

- 48-MHz HCS08CPU(中央处理单元)
- 24-MHz 内部总线频率
- HC08 指令集，增加了 BGND 指令
- 支持高达 32 个中断/复位源

存储器选项

- 高达 16KB 的片上在线可编程 FLASH 存储器，有块保护和安全选项
- 高达 512B 的片上 RAM
- 256B 的 USB RAM

时钟源选项

- 时钟源选项包括晶体，蜂鸣器，外部时钟，或能够 NVM 调整的精确的内部集成时钟
- MCG（通用时钟产生器）-PLL，FLL；带修正的内部参考时钟。

系统保护

- 可选的系统正常操作（COP）复位，使用独立的 1KHz 的内部时钟源或总线时钟运行
- 低压检测复位或中断
- 非法操作符检测复位
- 非法地址检测复位

省电模式

- 增加了两种停止模式

加载 USB

- MASS 位整体擦除闪存阵列
- 部分擦除闪存阵列-擦除除起始的 1KB 外的所有块。
- 编程区

外部设备

- USB-USB 2.0 全速（12Mbps），含专用片上 3.3V 调节器和收发器；支持端口 0 以及多达 6 个附加端点。
- SPI-8 或 16 位可选的串行外设接口模块并有硬件匹配功能模块
- SCI-一个串行通信接口模块，可选的 13 位停止符；全双工反向不归零;LIN 扩展主机停止位产生；LIN 从机扩展停止位检测；活动边沿唤醒
- MTIM-8 位模数计数器，含 8 位预分频因子和溢出中断
- 定时器-一个 2 通道 16 位定时器/脉宽调制（TPM）模块；每个定时器在每个通道上都有可选的输入捕捉，输出比较，PWM。每个定时器在每个通道上都可以配置

为带缓冲，中心 PWM（CPWM）

- KBI-8 引脚键盘中断模块
- RTC-含有二进制或十进制与分频器的实时计数器
- CRC-带 16 位移位寄存器的硬件 CRC 产生器电路；基于多项式 x^16+x^12+x^5+1 的 CRC16-CCITT 校验

输入/输出

- 输入时，每个端口都有软件选择的上拉电阻
- 输出时，每个端口都有软件选择的转换速率控制
- 输出时，每个端口都有软件选择的驱动强度
- 主机复位引脚和上电复位（POR）
- RESET，IRQ，BKGD/MS 引脚在内部上拉以减小系统消耗

封装选择

- 24 引脚 QFN
- 20 引脚 SOIC

版本历史

为了提供最先进的最新资料，我们在万维网上发布的资料是最新修改的。您的印刷本可能是此修订前的版本。若需核查您是否具有最新的可用信息，请参考：http://freescale.com

下面描述了此版本与历史版本的所发生的变化

版本号	版本日期	变化描述
1	8/27/2008	最初发行版
2	12/17/2008	0xFFAE 和 0xFFAF 的寄存器内容发生改变，见表 4-4 以及出厂修正值的描述 KBI 特性部分的删除重复信息 在复位为 0 后，改变 PTASE/PTBSE 寄存器的默认值
3	3/6/2009	更新图 4-4 和图 4-5
4	4/24/2009	增加了 MC9S08JS16L 和 MC9S08JS8L 信息

目　　录

第一章 导言

1.1 芯片概述

MC9S08JS60 系列 MCU 是 HCS08 家族成员中低成本、高性能的 8 位微处理器。家族中所有的 MCU 使用增强型 HCS08 内核，可以根据需要选取使用具有不同的模块，存储器大小，存储器类型和封装类型。

注：

MC9S08JS16/MC9S08JS8 和 MC9S08JS16L/MC9S08JS8L 唯一的区别在于 MC9S08JS16 和 MC9S08JS8 支持 USB 引导程序在电压高于 3.9V 时工作，而 MC9S08JS16L 和 MC9S08JS8L 支持 USB 引导程序在电压等于 3.3V 时工作。

当使用 MC9S08JS16L 和 MC9S08JS8L 的 USB 引导程序功能时，应禁止内部 USB 电压调节器并向 V_{USB33} 引脚输入 3.3V 电源。

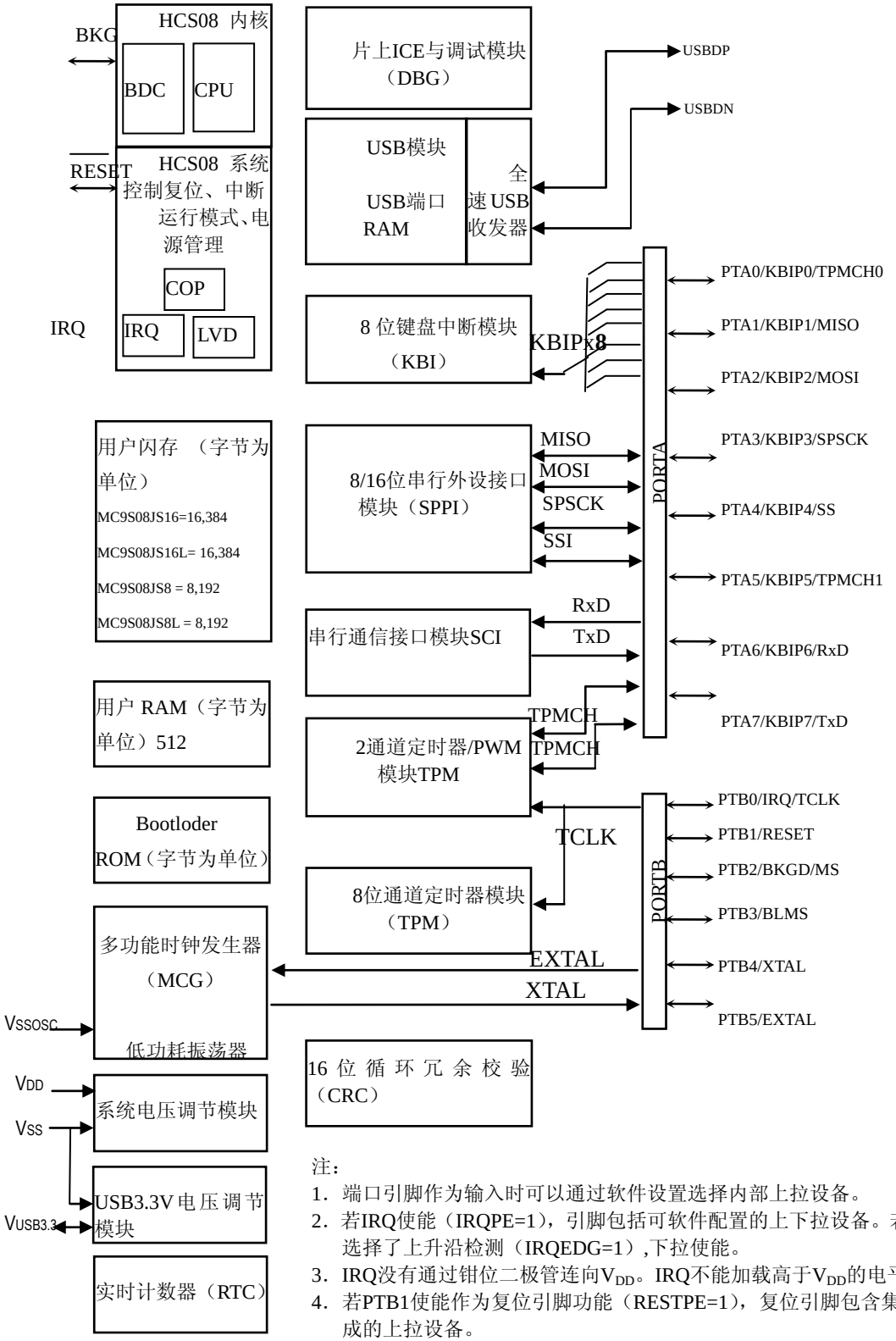
表 1-1 总结了在 MC9S08JS16 系列中芯片的每个封装类型的可用外设。

表1-1 MC9S08JS16系列各MCU封装的特点

特点	MC9S08JS8/ MC9S08JS8L			MC9S08JS16/ MC9S08JS16L		
封装	24引脚 QFN		20引脚 SOIC	24引脚 QFN		20引脚 SOIC
Flash大小（字节）	8192			16384		
RAM大小（字节）	512			512		
USB RAM（字节）	256			256		
ACMP	有			256		
ADC	12通道			有		
IIC	有	8通道	8通道	12通道	8通道	8通道
IRQ	有			有		
KBI	8			有		
SCI	有			8	7	7
SPI	有			有		
MTIM	有			有		
TPM通道	2			2		
USB	有			有		
CRC	有			有		
I/O引脚	14（2个只能输出）			14（2个只能输出）		

1.2 MCU框图

图 1-1 的框图给出了 MC9S08JS16MCU 的结构。



注：

1. 端口引脚作为输入时可以通过软件设置选择内部上拉设备。
2. 若IRQ使能（IRQPE=1），引脚包括可软件配置的上下拉设备。若选择了上升沿检测（IRQEDG=1）,下拉使能。
3. IRQ没有通过钳位二极管连向V_{DD}。IRQ不能加载高于V_{DD}的电平
4. 若PTB1使能作为复位引脚功能（RESTPE=1），复位引脚包含集成的上拉设备。
5. 引脚包含集成上拉设备。
6. 当KBI使能（KBIPEn=1）而且相关引脚被配置为上拉设备使能，那么上升沿（KBEDGn=1）被用于将上拉设备配置为下拉设备。

图 8-1. MC9S08JS16系列框图

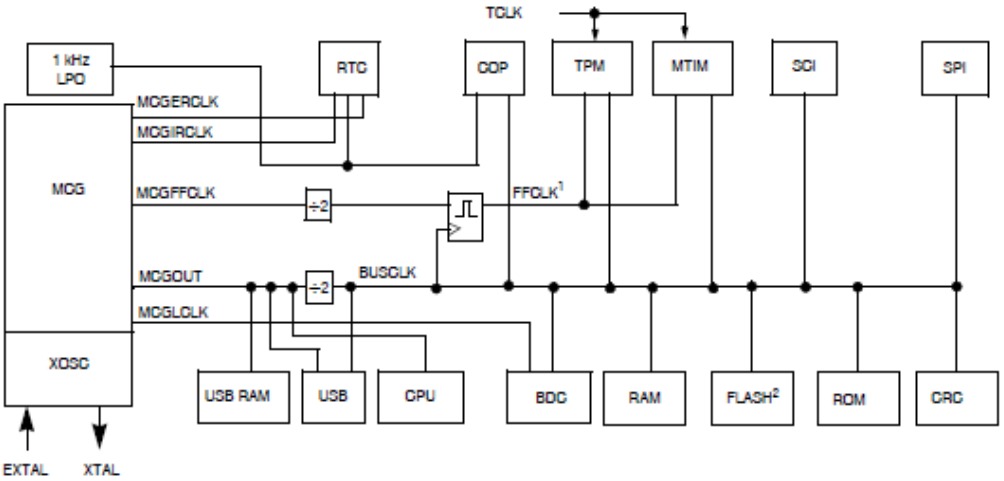
表 1-2 列出了片内模块的功能版本。

表1-3片内模块的版本

模块	版本
中央处理器（CPU）	2
模拟比较器（ACMP）	2
多功能时钟发生器（MCG）	1
实时计数器（RTC）	1
AD转换器（ADC）	1
串行通讯接口（SCI）	4
16位串行外设接口（SPI16）	1
模计时器（MTIM）	1
计时器脉冲宽度调制器（TPM）	3
通用串行总线（USB）	1
循环冗余校验发生器（CRC）	2
调试模块（DBG）	2

1.3 系统时钟分布

TCLK——TPM 和 MTIM 的外部输入时钟源，并且在第十四章，“计时器脉冲宽度调制器（S08TPMV3）”以 TPMCLK 被提到。



1 固定频率时钟（FFCLK）被内部同步到总线时钟，且不能超过总线时钟频率的一半。
2 FLASH和EEPROM对写入和擦除操作有频率要求，见MC9S08JS16系列手册获得详细说明。

图1-2 系统时钟分布框图

MCG 提供了如下时钟源:

- MCGOUG——该时钟源被用于 CPU, USB RAM 和 USB 模块的时钟，除以二即得到外设总线时钟频率（BUSCLK）。MCG 控制寄存器中的控制位决定了连接三个时钟源中的哪一个。
  ——内部参考时钟
  ——外部参考时钟
  ——频率锁相环（FLL）或者 PLL 锁相环输出
  参考第九章，“多功能时钟发生器（S08MCGV1），”参考设置 MCGOUT 时钟的细节。

- MCGLCLK——该时钟源由 MCG 的数控振荡器（DCO）产生。当系统内总线时钟频率较低时可以用高级工具选择这个内部自时钟来加速 BDC 通讯。
- MCGIRCLK——这是内部参考时钟并且可以被选作实时计数器（RTC）的时钟源。第九章，“多功能时钟发生器（S08MCGV1），”详细解释了 MCGIRCLK。第十一章，“实时计数器（S08RTCV1），”给出了更多关于 MCGIRCLK 使用方面的信息。
- MCGERCLK——这是外部参考时钟并且可被选作实时计数器模块的时钟源。9.4.6 节，“外部参考时钟”，详细说明了 MCGERCLK。第十一章，“实时计数器（S08RTCV1）”，给出了关于 MCGERCLK 应用于该模块的详细信息。
- MCGFFCLK——这个时钟源频率与 BUSCLK 同步以后除以 2 用以产生 FFCLK。它可以被选作 TPM 或 MTIM 模块的时钟源。MCGFFCLK 的频率由 MCG 的具体设置决定。参考 9.4.7 节，“固定频率时钟”，获得更多细节。
- LPO 时钟——这个时钟由一个完全独立于 MCG 模块的内部低功耗振荡器产生。LPO 时钟可被选作 RTC 或 COP 模块的时钟源。第十一章，“实时计数器（S08RTCV1），”和 5.4 节，“计算机正常操作监控模块（COP）看门狗，”给出了 LPO 时钟应用于这些模块的详细说明。
- TPMCLK——TPMCLK 是 TPM 或 MTIM 模块的可选外部时钟源。TPMCLK 的频率必须限制在总线频率的 1/4 以内来保证同步。参考第十四章，“计时器脉冲宽度调制器（S08TPMV3），”获得更多信息。

第二章 引脚和连接

2.1 简介

这一章描述了连接到各封装引脚的信号。内容包括引脚布局图，信号性能表，以及对信号的详细描述。

2.2 芯片引脚布局

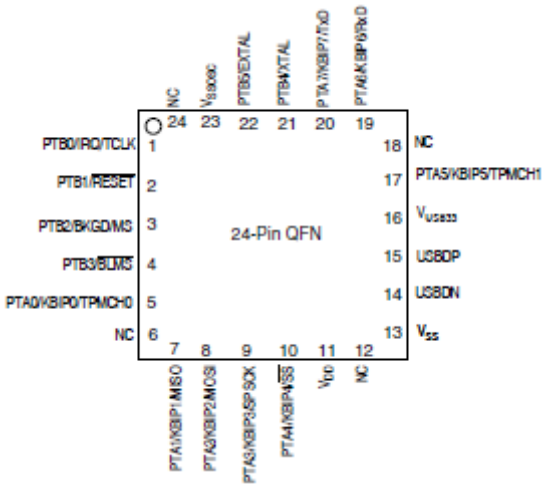


图2-1 MC9S08JS16系列24引脚QFN封装

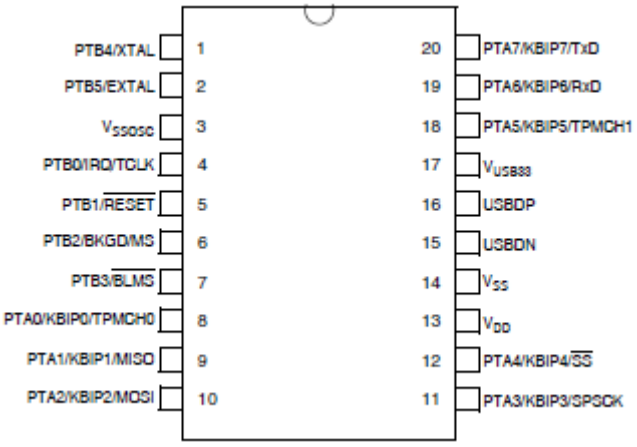
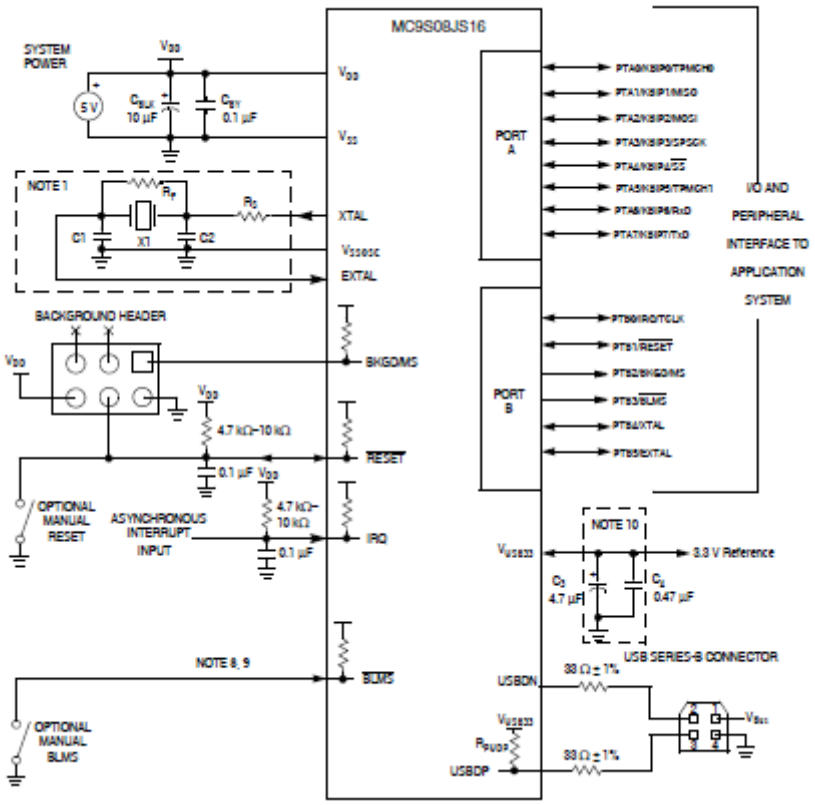


图2-2 MC9S08JS16系列20引脚SOIC封装

2.3 推荐系统连接

图 2-3 显示了几乎所有 MC9S08JS16 系列应用系统通用的引脚连接。



注：
1.如果使用MCG内部时钟则不需要外部晶振电路。USB操作需要外部晶振。
2.XTAL和EXTAL引脚和PTB4，PTB5一样。
3. EMC敏感应用推荐在RESET和IRQ上接RC滤波器。
4 R_{PUDP}只用于全速USB。框图显示了片内调节器和R_{PUDP}被使能时的设置。电压调节器输出用于R_{PUDP}。当在USBDP上接外部上拉电阻时，P_{RUDP}可以选择性被禁止。
5.V_{BUS}是从上游端口提供的5V电压用于USB操作。
6.USBDP和USBDN由3.3V调节器供电。
7.USB操作时推荐一个2MHZ或4MHZ的外部晶振。
8.BLMS引脚有装载限制，不在该引脚上连接任何电容。
9.如果在PTB2/PTB3上游外部上拉电阻，引脚被安排为输出引脚。否则会在引脚上有电流消耗（10K欧姆有0.5mA）。PTB2/PTB3上负载必须小于50PF。
10.如果使用V_{USB33}作为电源，需要一个外部电容。

图2-3 基本系统连接

2.3.1 电源（V_{DD}, V_{SS}, V_{SSOSC}, V_{USB33}）

V_{DD}和V_{SS}是MCU主要电源供应引脚，该电源给所有I/O缓冲电路和一个内部电压调节器供电。内部电压调节器将调节后的低压电源提供给CPU和MCU其它内部电路。

一般情况下，实际应用系统中电源引脚接两个独立的电容。在这种情况下，应该选用一个大容量电解电容器来为全体系统提供大电量存储，如10µF的钽电容，另外一个选用0.1µF的陶瓷旁路电容，尽可能的靠近配对的V_{DD}和V_{SS}电源引脚，用以抑制高频噪声。MC9S08JS16系列都有一个V_{SSOSC}引脚。这个引脚应连接到系统的接地平面或低阻抗连接到主V_{SS}引脚。

V_{USB33}连接内部USB 3.3V电压调节器。第十五章，“通用串行总线设备控制器

（S08USBV1）”，全面介绍了 V_{USB33} 电源引脚。

2.3.2 振荡器（ XTAL ,EXTAL ）

复位过后，MCU 使用内部发生的时钟，这个时钟源由 MCG 模块产生。想了解更多关于 MCG 的信息，请参考第九章，“多功能时钟发生器（S08MCGV1）”。

这个 MCU 的振荡器（XOSC）是一个皮尔斯振荡器，可容纳一个晶体或陶瓷谐振器，除了一个晶体或陶瓷的谐振器，一个外接振荡器还可以被连接到 EXTAL 输入引脚。

RS（使用时）与 RF 是低电感电阻，如碳电阻。线绕电阻和一些金属膜电阻器，电感太大。C1 和 C2 通常应是专门为高频率应用设计的高品质的陶瓷电容器。

RF 是用来提供一个偏置的路径的电阻，来保证在晶振启动时，EXTAL 的输入在其线性范围内且其阻值一般不是很重要。典型的系统 RF 阻值为 1MΩ 至 10MΩ 。阻值较高会对湿度比较敏感，阻值较低会减少增益并有可能阻止启动（在极端情况下）。

C1 和 C2 通常在 5pF 至 25pF 范围之间，选择时要与具体的晶振或谐振器相匹配。在选择 C1 和 C2 时务必要考虑到印刷电路板（PCB）和 MCU 引脚的电容。晶振制造商通常要指定一个负载电容，值等于 C1 和 C2（电容值通常相等）串联的串联值。作为首选的近似值，为每个振荡器引脚（EXTAL 和 XTAL）使用约 10pF 的电容作为引脚和 PCB 的连接电容。

2.3.3 $\overline{\text{RESET}}$ 引脚

经过上电复位（POR），PTB1/$\overline{\text{RESET}}$ 引脚被缺省指定为一个通用输入端口引脚，PTB1。

将 SOPT1 中的 RSTPE 置 1 可以将该引脚设为包含了一个内接上拉设备的 $\overline{\text{RESET}}$ 引脚。之后，该引脚将一直作为 $\overline{\text{RESET}}$ 引脚直到下一个 LVD 或 POR。当被使能时，$\overline{\text{RESET}}$ 引脚可以被用来在该引脚被拉低时使 MCU 从一个外部源中复位。

内部上电复位和低电压复位电路通常使得系统不需要外部复位电路。引脚通常是连接到标准的 6 引脚背景调试连接器，所以一个开发系统可以直接复位 MCU 系统。如果需要，可以通过添加一个简单的接地开关来增加一个外部手动复位装置（拉低复位引脚来强迫复位）。

当有非 POR 复位启动时（无论来自外部信号还是系统内部），$\overline{\text{RESET}}$ 引脚的电平会被拉低大约 66 个总线周期，然后被释放。复位电路解码复位事件并通过置位系统控制复位状态寄存器（ SRS ）相应的位将其记录下来。

注：

加载在内接上拉 $\overline{\text{RESET}}$ 引脚上的电压测量值比 V_{DD} 低。连接到该引脚上的内接门电路被拉至 V_{DD}。

如果要求将 $\overline{\text{RESET}}$ 引脚拉至 V_{DD} 水平，则必须使用外接上拉设备。

注：

在 EMC 敏感的应用中，推荐在复位引脚（允许）上配置一个外部 RC 滤波器。（图 2-3）

2.3.4 背景/模式选择（ BKGD /MS ）

在上电复位（POR）或是背景调试强制复位（见 5.7.3 节，“系统背景调试强制复位寄存器（SBDFR），”获得更多细节）中，PTB2/BKGD/MS 引脚作为一个模式选择引脚。复位后，

该引脚立即作为背景引脚被用于背景调试通信。当作为 BKGD/MS 引脚时（BKGDPE=1），该引脚自动包含一个内部上拉电阻。

当 SOPT1 中的 BKGDPE 被置 1 时，背景调试通信功能被允许。BKGDPE 在任何 MCU 复位后都会被置 1，在使用任何 PTB2/BKGD/MS 引脚备选功能时该位必须被清零。

若这个引脚上没有任何连接，则 MCU 在复位的上升沿进入正常的操作模式。如果调试系统被连接到 6 引脚的标准背景调试接口，它可以在复位上升沿时保持 BKGD/MS 为低电平，强制 MCU 进入背景模式。

BKGD 引脚主要用于背景调试控制器（BDC）通讯使用，遵循自定义协议，该协议规定每传送一个位使用目标 MCU 的 16 个 BDC 时钟周期。目标 MCU 的 BDC 时钟频率可以和总线时钟频率一样，因此不要将任何大的电容和 BKGD/MS 引脚相连，否则会干扰背景串行通信。

虽然 BKGD 引脚是一个伪开漏引脚，但是背景调试通信协议提供了简短的，主动驱动，高加速脉冲以确保快速上升时间。电缆的小电容和内部上拉电阻的绝对值对 BKGD 引脚上的上升沿和下降沿几乎不起任何作用。

注：在 IRQ 或 $\overline{\text{RESET}}$ 功能被允许前，要在功能引脚上允许 GPIO 上拉并且等待 2μs。否则 IRQ 或 $\overline{\text{RESET}}$ 设置可能会失败。

2.3.5引导程序模式选择（$\overline{\text{BLMS}}$）

在上电复位（POR）过程中，CPU 探测作为模式选择引脚的 PTB3/$\overline{\text{BLMS}}$引脚的状态。当逻辑为低且 BKGD/MS 没有被拉低时，CPU 进入引导程序模式。在上电复位（POR）过程中，接到 PTB3/$\overline{\text{BLMS}}$引脚的内部上拉设备自动被使能。复位上升过后，该引脚立刻作为通用输出引脚且内部上拉设备自动被禁止。

2.3.6 USB数据引脚（USBDP,USBDN）

USBDP（D+）和 USBDN（D-）引脚是在 USB 物理层（PHY）模块内接全速数据通讯的模拟输入/输出线路。R_{PUDP} 是 USBDP 引脚的可选上拉电阻。

2.3.7 GPIO及外设端口

MC9S08JS16 系列 MCU 共支持 14 个通用 IO 引脚，包含两个与片内外设复用的单输出引脚，（定时器，串行 I/O，键盘中断，etc）。

当一个引脚被设置为一个通用输出引脚或外设输出引脚时，程序可以设置驱动强度和禁止或允许速率转换控制。当一个引脚被设置为一个通用输入引脚或外设输入引脚时，程序可以允许上拉设备。

要详细了解如何将这些引脚定义为输入/输出引脚，请参考第六章，“并行输入/输出”。要了解哪种情况下以及怎样将这些引脚定义为外设引脚，请参考相关章节。

复位过后，所有的引脚除了单输出引脚（PTB2/BKGD/MS，PTB3/$\overline{\text{BLMS}}$）都被设置为高阻抗通用输入引脚，上拉设备被禁止。

表2-1 引脚功能及封装引脚数

引脚数（封装）		《——最低	优先级	——》最高
24（QFN）	20（SOIC）	端口引脚	Alt 1	Alt 2
1	4	PTB0	IRQ	TCLK
2	5	PTB1		$\overline{\text{RESET}}$
3	6	PTB2	BKGD	MS
4	7	PTB3		$\overline{\text{BLMS}}$
5	8	PTA0	KBIP0	TPMCH0
6	—	NC		
7	9	PTA1	KBIP1	MISO
8	10	PTA2	KBIP2	MOSI
9	11	PTA3	KBIP3	SPSCK
10	12	PTA4	KBIP4	$\overline{\text{SS}}$
11	13			V_{DD}
12	—	NC		
13	14			V_{SS}
14	15			USBDN
引脚数（封装）		《——最低	优先级	——》最高
15	16			USBDP
16	17			V_{USB33}
17	18	PTA5	KBIP5	TPMCH1
18	—	NC		
19	19	PTA6	KBIP6	RxD
20	20	PTA7	KBIP7	TxD
21	1	PTB4	XTAL	
22	2	PTB5	EXTAL	
23	3			V_{SSOSC}
24	—	NC		

注：

当一个可选功能首先被允许时，可能会使该模块有一个杂乱的沿。用户程序必须在中断允许前将所有相关标志位清零。表 2-1 显示了如果有多个模块被允许它们之间的优先级。最高优先级的模块将会控制引脚。在一个较低优先级功能已经允许的情况下选择一个较高优先级的引脚功能将会使低优先级的模块有杂乱的沿。一个模块被允许前所有共享该引脚的模块必须被禁止。

第三章 操作模式

3.1 简介

本章将描述 MC9S08JS60 的操作模式。同时描述进入各种模式、退出各种模式和各种模式中的功能。

3.2 特性

用于代码设计的激活背景模式
等待模式：
CPU 暂停，以节省电力
系统时钟运行
保持全电压调节
停止模式：CPU 和总线时钟停止
STOP2—内部电路部分掉电，RAM 和 USB RAM 的内容保留
STOP3—为快速恢复，所有内部电路不掉电；RAM 和 USB RAM 以及寄存器内容保留

3.3 运行模式

这是 MC9S08JS60 系列在一般情况下的操作模式。在 MCU 离开复位时，若 BKGD/MS 为高电平，则进入此模式。在这种模式下复位后，CPU 从存储器的 0xFFFE:0xFFFF 中取出程序首地址，执行内存中的代码。

3.4 激活背景模式

激活背景模式功能由 HCS08 内核中的背景调试控制器（BDC）管理。在软件开发期间 BDC 连同片内在线仿真器（ICE）调试模块（DBG）一起提供各种方式来分析 MCU 操作（见 5.7.3 节，“系统背景调试强制复位寄存器（SBDFR）”）

进入激活背景模式的五种方式：
在 POR 过程中或紧接着背景调试强制复位之后当 BKGD /MS 引脚为低电平时
当 BKGD 引脚接收到 BACKGROUND 命令时
当 BGND 指令被执行时
当遇到 BDC 断点时
当遇到 DBG 断点时

在进入激活背景模式后，CPU 处于一个暂停状态，等待串行后台命令，而不是执行用户应用程序的指令。

后台命令的两种类型：

非插入命令，定义为可在用户程序运行时发出的命令。当 MCU 在运行模式下时，非插入命令可经由 BKGD 引脚发出。当 MCU 在激活背景模式下也可执行非插入命令。非插入命令包括：

存储器访问命令
存储器访问状态命令

BDC 寄存器访问命令

BACKGROUND 命令

激活背景命令，只有当 MCU 在激活背景模式下才可以执行。激活背景命令，包括：

读/写 CPU 寄存器

单步调试指令

退出激活背景模式，返回用户应用程序（GO）

激活背景模式通常用于 MCU 第一次在运行模式下运行之前，向 Flash 程序存储器写入引导程序或用户程序。当 MC9S08JS60 系列 FREESCALE 半导体出厂时，除非有特别指明，Flash 存储器默认为擦除，因此在运行模式下没有程序可以执行，直到 Flash 存储器初次被编程。激活背景模式也可用于在 Flash 存储器已经写入程序后，对其进行擦除和重新写入程序。

有关激活背景模式的更多信息可见第 17 章“开发支持”。

3.5 等待模式

等待模式由执行 WAIT 指令进入。在执行 WAIT 指令时，CPU 处于一个没有时钟的低功耗状态。当 CPU 的进入等待模式时，CCR 中的 I 位被清零，允许中断。当一个中断请求发生时，CPU 退出等待模式，恢复正常处理，开始堆栈操作，运行相应的中断服务程序。 而 MCU 在等待模式，对可用的背景调试命令有一些限制。当 MCU 在等待模式下，只有 BACKGROUND 命令和存储器访问状态命令可用。存储器访问状态命令不容许存储器访问，但它们会报告一个错误，表明 MCU 处于停止或等待模式。BACKGROUND 命令可以将 MCU 从等待模式唤醒，进入激活背景模式。

3.6 停止模式

当系统选项寄存器中的 STOPE 位置位时，执行到 STOP 指令系统就进入两种停止模式的一种。在两种停止模式下，总线和 CPU 时钟都被暂停。MCG 模块可以被设置为离开内部参考时钟运行。更多的有关信息可见第九章，“多功能时钟发生器（S08MCGV1），”。

一些为低电压（1.8 到 3.6V）操作设计的 HCS08 芯片还包含 STOP1 模式。MC9S08JS16 系列 MCU 不包含 STOP1 模式。

表 3-1 显示了所有影响停止模式选择的控制位以及各种情况下模式的选择。在执行一个 STOP 指令后系统进入一个选定的模式。

表3-1 停止模式选择

STOPE	ENBDM[1]	LVDE	LVDSE	PPDC	停止模式
0	x	x		x	停止模式被禁止，执行STOP指令会导致非法操作码复位
1	1	x		x	进入STOP3[2], BDM允许
1	0	两个位都必须为1		x	进入STOP3,激活电压调节器
1	0	任意一位为0		0	STOP3
1	0	任意一位为0		1	STOP2

1 ENBDM 位于 BDCSCR 中，仅仅能通过 BDC 命令进入，见 14.4.1.1 节，“BDC 状态与控制寄存器（BDCSCR）“。

2 当处于 STOP3 模式且 BDM 允许时，因为内部时钟允许运行，S_{IDD} 会接近 R_{IDD} 水平。

3.6.1 STOP3模式

在表 3-1 所示的条件下，执行 STOP 指令进入 Stop3 模式。所有的内部寄存器和逻辑，RAM 的内容，和 I/O 引脚状态都被保持。

STOP3 可以通过 $\overline{\text{RESET}}$ 方式退出，或由以下某项中断源退出：实时中断（RTC），USB 恢复中断，LVD，IRQ，KBI，或 SCI。

如果 STOP3 是以 $\overline{\text{RESET}}$ 引脚方式退出的，那么获取复位向量后，MCU 会复位，操作会恢复。如果因内部中断源退出，MCU 就获取适当的中断向量。

3.6.1.1 停止模式下LVD使能

当供应电压下降到 LVD 电压值以下时，LVD 系统能够产生中断或复位。如果在 CUP 执行 STOP 指令时 LVD 在停止模式中被允许（SPMSC1 中的 LVDE 和 LVDSE 都被设置为 1），则电压调节器在停止模式过程中保持激活状态。如果在停止模式中允许 LVD 的情况下用户试图进入 SPOP2，MCU 会进入 STOP3 模式。

当 MCGC2 中的 RANGE 位被置 1 时，为了可以在一个外部参考时钟下操作 XOSC，进入 STOP3 模式时 LVD 必须被设置为允许。

注：

为了得到 USB 低挂起电流，在进入 USB 挂起模式之前，我们必须将 MCGC2 中的 ERCLKEN 何 EREFSTEN 置 1，但是在 STOP3 模式中将 LVD 禁止。

3.6.1.2 停止模式下激活的BDM使能

如果 BDCSCR 的 ENBDM 位被置位，则从运行模式进入激活背景模式被允许。第 17 章“开发支持”中描述了该寄存器。当 CPU 执行 STOP 指令时，如果 ENBDM 被置位，这样当 MCU 进入到停止模式后背景调试逻辑的系统时钟仍然在工作，所以背景调试通信仍然是可能的。此外，电压调试器没有进入低功耗待机状态，而是保持满负荷工作中。如果用户试图在 ENBDM 置位的情况下进入 STOP2，MCU 将进入 STOP3 。

大部分后台命令在停止模式下是无效的。存储器访问状态命令不允许访问存储器，但它们会报告一个错误，表明 MCU 在停止或等待模式。如果 ENBDM 置位，BACKGROUND 指令可以用于将 MCU 从停止模式中唤醒进入激活背景模式。一旦进入背景调试模式后，所有后台命令都可用。

3.6.2 STOP2模式

在如表 3-1 的条件下执行一条 STOP 指令即可进入 STOP2 模式。大多数 MCU 的内部电路在 STOP2 时都处于掉电状态，除了 RAM 以外。进入 STOP2 模式，所有的 I/O 引脚控制信号被锁定，以在 STOP2 模式期间保留引脚状态。

离开 STOP2 模式可以依靠下面两个唤醒引脚中任意一个：$\overline{\text{RESET}}$ 或 IRQ。当一个程序使用 STOP2 模式状态，$\overline{\text{RESET}}$ 或 IRQ 引脚必须事先预设置为一个输入引脚来进入 STOP2 模式。一个模拟连接从 $\overline{\text{RESET}}$ 或 IRQ 垫连接到电源管理控制器唤醒引脚——如果该引脚被设置为 GPIO 输出引脚，它可以防止 STOP2 的操作发生。

另外，RTC 中断如果被允许，它也可以将 MCU 从 STOP2 模式中唤醒。

将系统从 STOP2 模式唤醒后，MCU 将作为上电复位（POR）启动：

- 所有的模块控制和状态寄存器将被复位

- 如果 V_{DD} 低于 LVD 的跳变点（低跳变点的选择取决于 POR），LVD 复位功能将被允许，MCU 将保持在复位状态。
- CPU 获得复位向量

除上述以外，从 STOP2 模式被唤醒后，SPMSC2 的 PPDF 位被置位。这个标志被用来指向用户代码进入一个 STOP2 的一个恢复例程。PPDF 保持置位、I/O 引脚状态保持锁存，直到逻辑 1 写入 SPMSC2 的 PPDACK 位。

进入 STOP2 模式前，要保持被设定为通用 I/O 引脚的 I/O 状态，用户必须在写 PPDACK 位之前将已被保存在 RAM 上的 I/O 端口寄存器中的内容写回端口寄存器。在写 PPDACK 之前，如果端口寄存器没有存储来自 RAM 的内容，那么在写 PPDACK 位时，I/O 引脚将转换到其复位状态。

引脚被设定为外设 I/O 引脚，在写 PPDACK 位之前，用户必须重新设定接口到该引脚的外设模块。在写 PPDACK 之前，如果外设模块未启用，当 I/O 锁存开放时，引脚将被其相关的端口控制寄存器控制。

3.6.3 停止模式下片内外设模块

当 MCU 进入任何 STOP 模式，到内部外设模块的系统时钟都被关闭。甚至在例外情况下（ENBDM=1），背景调试逻辑的时钟继续运行，外设系统时钟也会停止以降低功耗。停止模式下系统行为的具体信息请参阅 3.6.2 节，“STOP2 模式”，和 3.6.1，“STOP3 模式”。表 3-4 给出了停止模式下的行为。

表3-2 停止模式的系统行为

外设	模式	
	STOP2	STOP3
CPU	关闭	待机
RAM	待机	待机
Flash	关闭	待机
并行端口寄存器	关闭	待机
MCG	关闭	可选[1]
RTC	可选[4]	可选[2]
SCI	关闭	待机
MTIM	关闭	待机
SPI	关闭	待机
TPM	关闭	待机
系统电压调节器	关闭	待机
XOSC	关闭	可选[3]
I/O引脚	保持状态	保持状态
USB（SIE和PHY）	关闭	可选[4]
USB 3.3V 调节器	关闭	待机
USB RAM	待机	待机
CRC	关闭	待机

1 MCGC1 中的 IREFSTEN 被置位，否则待机。

2 RTCSC 中的 RTCPS[3:0]在进入停止模式前不等于 0，否则关闭

3 MCGC2 中的 EREFSTEN 被置位，否则待机。在 STOP3 模式下，高频范围（MCGC2 中的 RANGE 被置位）要求 LVD 必须被开启。

4 CTL 中的 USBEN 和 USBCTL0 中的 USBPHYEN 被置位，否则关闭。

第四章 存储器

4.1 简介 MC9S08JS16系列存储器映像

图 4-1 给出了 MC9S08JS16 系列 MCU 的存储器映像。MC9S08JS16 系列产品中的片内存储器包括 RAM，用于非易失性数据存储的 FLASH 程序存储器，I/O 和控制/状态寄存器。这些寄存器可分为以下三类：

直接页寄存器（0x0000-0x007F）

高页寄存器（0x1800-0x185F）

非易失性寄存器（0xFFB0-0xFFBF）

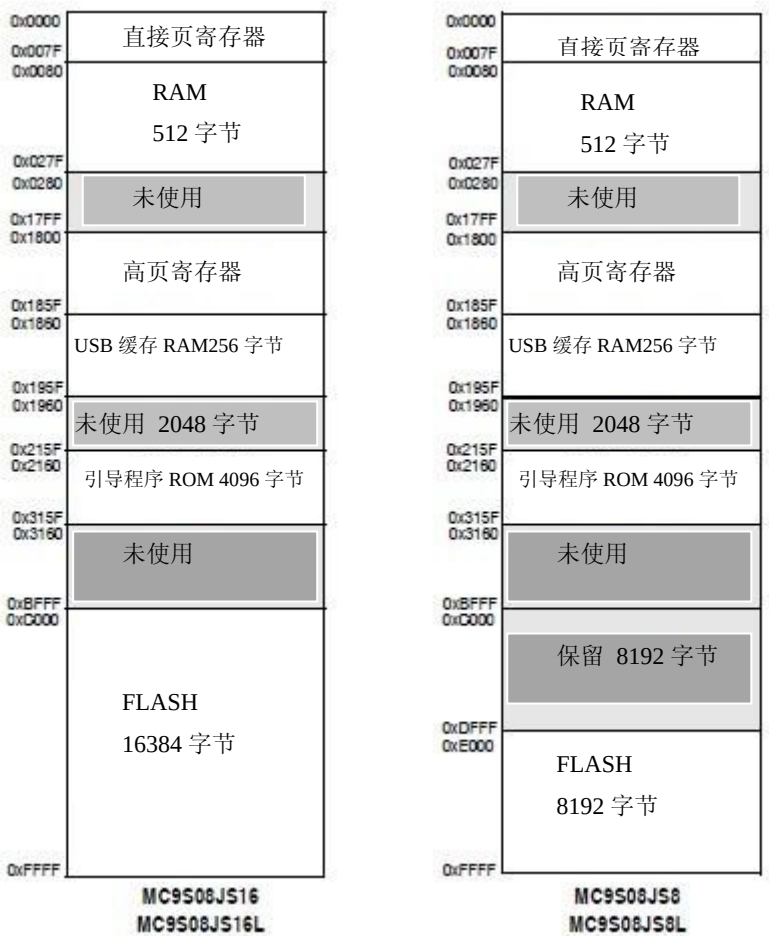


图4-1 MC9S08JS16系列存储器映像

4.1.1 复位和中断向量分配

表 4-1 为复位和中断向量地址的分配情况。该表中使用的向量名即为飞思卡尔提供的 MC9S08JS16 系列通用文件中使用的标签。关于复位，中断，中断优先级，和中断屏蔽控制的更多细节，请参阅第 5 章，“复位，中断，和系统配置”。

表4-1 复位和中断向量

地址（高/低）	向量	向量名
0xFFC0:0xFFC1	未使用的向量空间[1]	

到 0xFFC2:FFC3		
0xFFC4:FFC5	RTC	Vrtc
0xFFC6:FFC7	未使用的向量空间	—
0xFFC8:FFC9	未使用的向量空间	—
0xFFCA:FFCB	MTIM	Vmtim
0xFFCC:FFCD	KBI	Vkeyboard
0xFFCE:FFCF 到 0xFFD2:FFD3	未使用的向量空间	—
0xFFD4:FFD5	SCI 发送	Vscitx
0xFFD6:FFD7	SCI 接收	Vscirx
0xFFD8:FFD9	SCI 错误	Vscierr
0xFFDA:FFDB 到 0xFFE6:FFE7	保留	—
0xFFE8:FFE9	TPM 溢出	Vtpmovf
0xFFEA:FFEB	TPM 通道 1	Vtpmch1
0xFFEC:FFED	TPM 通道 0	Vtpmch0
0xFFEE:FFEF	未使用的向量空间	—
0xFFF0:FFF1	USB 状态	Vusb
0xFFE6:FFE7	TPM1 通道 4	Vtpm1ch4
0xFFE8:FFE9	TPM1 通道 3	Vtpm1ch3
0xFFEA:FFEB	TPM1 通道 2	Vtpm1ch2
0xFFEC:FFED	TPM1 通道 1	Vtpm1ch1
0xFFEE:FFEF	TPM1 通道 0	Vtpm1ch0
0xFFF0:FFF1	预留	-
0xFFF2:FFF3	USB状态	Vusb
0xFFF4:FFF5	SPI	Vspi
0xFFF6:FFF7	MCG锁损失	Vlol
0xFFF8:FFF9	低电压检测	Vlvd
0xFFFA:FFFB	IRQ	Virq
0xFFFC:FFFD	SWI	Vswi
0xFFFE:FFFF	复位	Vreset

1 未使用的向量空间可以作为通用 FLASH 存储空间来使用。然而，S08 系列 MCU 其他的芯片也许使用了这些地址作为中断向量。因此，如果要将代码移植到其他 MCU 必须小心使用这些地址空间。

4.2 寄存器地址和位分配

MC9S08JS16 系列产品中的寄存器可分为以下三组：

直接页寄存器，位于存储器映像的前 128 个位置上，这些寄存器可以通过高效的直接寻址方式指令访问。

高页寄存器不经常使用，因此位于存储器映像中 0x1800 以上。在直接页中为频繁使用

的寄存器和变量留出了更多的空间。

非易失性寄存器，由 FLASH 存储器中 0xFFB0-0xFFBF 之间 16 个地址段组成。

非易失性寄存器地址包括：

在复位时，加载到工作寄存器的三个值

一个 8 字节的后门对比密钥，可选择性的允许用户拥有对安全内存的受控制的访问权限。

由于非易失性寄存器位于 FLASH 存储器中，它们必须像其它 FLASH 存储器地址一样进行擦除和编程。

直接页寄存器可以通过高效的直接寻址方式指令访问。位操纵指令可用于访问任何直接页寄存器中的任何位。表 4-2 总结了所有用户可访问的直接页寄存器和控制位。

表 4-2 所列的直接页寄存器可以使用更高效的直接寻址方式，只需要地址的低位字节。正因为如此，第 1 栏中地址的较低字节用粗体显示。在表 4-3 和表 4-4 中，第 1 栏中的整个地址都用粗体显示。在表 4-2,表 4-3,和表 4-4 中，第 2 栏中的寄存器名称用粗体显示以便与右侧的位名称区分。与所列出的位不相关的单元格用阴影显示。带有 0 的阴影单元表示这个未使用的位始终应为 0。带有破折号的阴影单元表示未使用的或预留的位，可以是 1 或 0。

表4-2 直接页寄存器汇总

地址	寄存器名称	位 7	位 6	位 5	位 4	位 3	位 2	位 1	位 0
0x0000	PTAD	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
0x0001	PTADD	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
0x0002	PTBD	0	0	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
0x0003	PTBDD	0	0	PTBDD5	PTBDD4	PTBDD3	PTBDD2	PTBDD1	PTBDD0
0x0004- 0x0007	预留	— —	— —	— —	— —	— —	— —	— —	— —
0x0008	MTIMSC	TOF	TOIE	TRST	TSTP	0	0	0	0
0x0009	MTIMCLK	0	0	CLKS		PS			
0x000A	MTIMCNT	COUNT							
0x000B	MTIMMOD	MOD							
0x000C	CRCH	位 15	14	13	12	11	10	9	位 8
0x000D	CRCL	位 7	6	5	4	3	2	1	位 0
0x000E	预留	—	—	—	—	—	—	—	—
0x000F	预留	—	—	—	—	—	—	—	—
0x0010	TPMSC	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
0x0011	TPMCNTH	位 15	14	13	12	11	10	9	位 8
0x0012	TPMCNTL	位 7	6	5	4	3	2	1	位 0
0x0013	TPMMODH	位 15	14	13	12	11	10	9	位 8
0x0014	TPMMODL	位 7	6	5	4	3	2	1	位 0
0x0015	TPMC0SC	CH0F	CH0IE	MS0B	MS0A	ELS0B	ELS0A	0	0
0x0016	TPMC0VH	位 15	14	13	12	11	10	9	位 8
0x0017	TPMC0VL	位 7	6	5	4	3	2	1	位 0
0x0018	TPMC1SC	CH1F	CH1IE	MS1B	MS1A	ELS1B	ELS1A	0	0
0x0019	TPMC1VH	位 15	14	13	12	11	10	9	位 8
0x001A	TPMC1VL	位 7	6	5	4	3	2	1	位 0
0x001B	IRQSC	0	IRQPDD	IRQEDG	IRQPE	IRQF	IRQACK	IRQIE	IRQMOD
0x001C	KBISC	0	0	0	0	KBF	KBACK	KBIE	KBMOD

0x001D	KBIPE	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0
0x001E	KBIES	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
0x001F	预留	—	—	—	—	—	—	—	—
0x0020	SCIBDH	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
0x0021	SCIBDL	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
0x0022	SCIC1	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
0x0023	SCIC2	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
0x0024	SCIS1	TDRE	TC	RDRF	IDLE	OR	NF	FE	RF
0x0025	SCIS2	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
0x0026	SCIC3	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
0x0027	SCID	位 7	6	5	4	3	2	1	位 0
0x0028-0x002F	预留	—	—	—	—	—	—	—	—
0x0030	SPIC1	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
0x0031	SPIC2	SPMIE	SPIMODE	0	MODFEN	BIDIROE	0	SPISWAI	SPC0
0x0032	SPIBR	0	SPPR2	SPPR1	SPPR0	0	SPR2	SPR1	SPR0
0x0033	SPIS	SPRF	SPMF	SPTEF	MODF	0	0	0	0
0x0034	SPIDH	位 15	14	13	12	11	10	9	位 8
0x0035	SPIDL	位 7	6	5	4	3	2	1	位 0
0x0036	SPIMH	位 15	14	13	12	11	10	9	位 8
0x0037	SPIML	位 7	6	5	4	3	2	1	位 0
0x0038-0x003F	预留	—	—	—	—	—	—	—	—
0x0040	MCGC1	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
0x0041	MCGC2	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
0x0042	MCGTRM	TRIM							
0x0043	MCGSC	LOLS	LOCK	PLLST	IREFST	CLKST		OSCINIT	FTRIM
0x0044	MCGC3	LOLIE	PLLS	CME	0	VDIV			
0x0045-0x0047	预留	—	—	—	—	—	—	—	—
0x0048	RTCSC	RTIF	RTCLKS		RTIE	RTCPS			
0x0049	RTCCNT	RTCCNT							
0x004A	RTCMOD	RTCOMD							
0x004B-0x004F	预留	—	—	—	—	—		—	—
0x0050	USBCTL0	USBRESET	USBPU	USBRESMEN	LPRESF	0	USBVREN	0	USBPHYEN
0x0051-	预留	—	—	—	—	—	—	—	—

0x0057									
0x0058	PERID	0	0	ID5	ID4	ID3	ID2	ID1	ID0
0x0059	IDCOMP	1	1	NID5	NID4	NID3	NID2	NID1	NID0
0x005A	REV	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
0x005B-0x005E	预留	—	—	—	—	—	—	—	—
0x005C	IICD	DATA							
0x005F	预留	—	—	—	—	—	—	—	—
0x0060	INTSTAT	STALLF	0	RESUMEF	SLEEPF	TOKDNEF	SOFTOKF	ERRORF	USBRSTF
0x0061	INTENB	STALL	0	RESUME	SLEEP	TOKDNE	SOFTOK	ERROR	USBRST
0x0062	ERRSTAT	BTSERRF	—	BUFERRF	BTOERRF	DFN8F	CRC16F	CRC5F	PIDERRF
0x0063	ERRENB	BTSERR	0	BUFERR	BTOERR	DFN8	CRC16	CRC5	PIDERR
0x0064	STAT	ENDP				IN	ODD	0	0
0x0065	CTL	—	—	TSUSPEND	—	—	CRESUME	ODDRST	USBEN
0x0066	ADDR	0	ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0
0x0067	FRMNUML	位 15	14	13	12	11	10	9	位 8
0x0068	FRMNUMH	位 7	6	5	4	3	2	1	位 0
0x0069-0x006C	预留	—	—	—	—	—	—	—	—
0x006D	EPCTL0	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x006E	EPCTL1	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x006F	EPCTL2	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x0070	EPCTL3	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x0071	EPCTL4	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x0072	EPCTL5	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHK
0x0073	EPCTL6	SPRF	SPMF	SPTEF	MODF	0	0	0	0
0x0074-0x007F	预留	—	—	—	—	—	—	—	—

表 4-3 中列出的高页寄存器的访问频率比其它 I/O 和控制寄存器低很多，因此存放在可直接寻址的内存空间外，从 0x1800 开始。

注：

MCG 工厂调整值将会在任何复位发生之后自动装载进 0x180C 和 0x180D 寄存器中。

表4-3 高页寄存器汇总

地址	寄存器名称	位 7	6	5	4	3	2	1	位 0
0x1800	SRS	POR	PIN	COP	ILOP	ILAD	LOC	LVD	—
0x1801	SBDFR	0	0	0	0	0	0	0	BDFR
0x1802	SOPT1	COPT		STOPE	1	0	SPIFE	0	0
0x1803	SOPT2	COPCLKS	COPW	0	0	0	SPIFE	0	0
0x1804	预留	—	—	—	—	—	—	—	—
0x1805	预留	—	—	—	—	—	—	—	—
0x1806	SDIDH	—	—	—	—	ID11	ID10	ID9	ID8
0x1807	SDIDL	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
0x1808	预留	—	—	—	—	—	—	—	—
0x1809	SPMSC1	LVWF	LVWACK	LVWIE	LVDRE	LVDSE	LVDE	0	0
0x180A	SPMSC2	0	0	LVDV	LVWV	PPDF	PPDACK	0	PPDC
0x180B	预留	—	—	—	—	—	—	—	—
0x180C	为 FTRIM 存储预留	—	—	—	—	—	—	—	—
0x180D	为 MCGTRIM 存储预留	TRIM							
0x180E	FPROTD	—	—	—	—	—	—	—	—
0x180F	SIGNATURE	SIGNATURE 信号量							
0x1810	DBGCAH	位 15	14	13	12	11	10	9	位 8
0x1811	DBGCAL	位 7	6	5	4	3	2	1	位 0
0x1812	DBGCBH	位 15	14	13	12	11	10	9	位 8
0x1813	DBGCBL	位 7	6	5	4	3	2	1	位 0
0x1814	DBGFH	位 15	14	13	12	11	10	9	位 8
0x1815	DBGFL	位 7	6	5	4	3	2	1	位 0
0x1816	DBGC	DBGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
0x1817	DBGT	TRGSEL	BEGIN	0	0	TRG3	TRG2	TRG1	TRG0
0x1818	DBGS	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
0x1819-0x181F	预留	—	—	—	—	—	—	—	—
0x1820	FCDIV	DIVLD	PRDIV8	DIV5	DIV4	DIV3	DIV2	DIV1	DIV0
0x1821	FOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
0x1822	预留	—	—	—	—	—	—	—	—
0x1823	FCNFG	0	0	KEYACC	0	0	0	0	0
0x1824	FPROT	FPS7	FPS6	FPS5	FPS4	FPS3	FPS2	FPS1	FPDIS
0x1825	FSTAT	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
0x1826	FCMD	FCMD7	FCMD6	FCMD5	FCMD4	FCMD3	FCMD2	FCMD1	FCMD0
0x1827-0x183F	预留	—	—	—	—	—	—	—	—
0x1840	PTAPE	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0

0x1841	PTASE	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
0x1842	PTADS	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
0x1843	预留	—	—	—	—	—	—	—	—
0x1844	PTBPE	0	0	PTBPE5	PTBPE4	PTBPE3	PTBPE2	PTBPE1	PTBPE0
0x1845	PTBSE	0	0	PTBSE5	PTBSE4	PTBSE3	PTBSE2	PTBSE1	PTBSE0
0x1846	PTBDS	0	0	PTBDS5	PTBDS4	PTBDS3	PTBDS2	PTBDS1	PTBDS0
0x1847- 0x185F	预留	— —	— —	— —	— —	— —	— —	— —	— —

表4-4 非易失性寄存器汇总

地址	寄存器名称	位 7	6	5	4	3	2	1	位 0
0xxFFAE	为 FTRIM 存储保留	0	0	0	0	0	0	0	FTRIM
0xxFFAF	为 MCGTRIM 存储保留	TRIM							
0xFFB0- 0xFFB7	NVBACKKEY	8 字节比较密钥							
0xFFB8-		FLASH 块高字节和校验值							
0xFFBC		FLASH 块低字节和校验值							
0xFFBA		旁路和校验值							
0xFFBB	为用户数据存储预留	用户数据							
0xFFBC	为用户数据存储预留	用户数据							
0xFFBD	NVPROT	FPS7	FPS6	FPS5	FPS4	FPS3	FPS2	FPS1	FPS0
0xFFBE		FLASH 部分擦除信号量							
0xFFBF	NVOPT	KEYEN	FNORED	0	0	0	0	SEC01	SEC00

表 4-4 列出的非易失性 FLASH 寄存器，位于 FLASH 中。这些寄存器包括 1 个 8 字节的后门密钥，可选择性用于访问安全的内存资源。在复位过程中，FLASH 中非易失性寄存器区域的 NVPROT 和 NVOPT 内容被转移到高页寄存器中相应的 FPROT 和 FOPT 工作寄存器中，以控制安全和块保护选项。

MCG 出厂跳变值被存储在 FLASH 信息行（IFR[1]），在任何复位发生之后被装载进入 MCGTRM 和 MCGSC 寄存器。在 FLASH，TRIM，和 FTRIM 中的内部参考跳变值可以被第三方程序重新编程改变，且必须被用户程序拷贝到相应 MCG 寄存器来覆盖出厂值。

注：

当 MCU 处于激活 BDM 模式中，IFR 中的跳变值不会被装载，MCGTRM 寄存器复位值为 0x80，MCGSC 中的 FTRIM 位会复位为 0。

1 IFR — 非易失性信息存储空间，仅能在产品测试中被访问。在产品测试期间，系统初始化，设置和测试信息被存储在 IFR。这个信息不能在普通用户模式或背景调试模式中被读取或修改。

如果密钥启用（KEYEN）位为 1，那么 8 字节对比密钥可用于暂时脱离内存安全的限制。这种密钥机制只能通过在安全内存中运行的用户代码来访问。（安全密钥不能通过背景调试命令直接输入。）这个安全密钥可通过将 KEYEN 位设为 0 来完全禁用。如果这个安全密钥被禁用，那么如果需要，脱离安全限制的唯一方式是整体擦除 Flash（通常通过背景调试接口）并确认 Flash 已为空。为了避免在下一次复位后返回到安全模式，应该将安全位（SEC01：SEC00）设置为非安全状态 （1：0）。

0xFFBB 和 0xFFBC 被用来存储用户数据，例如用户的 MCG 跳变值。

4.3 RAM（系统RAM）

MC9S08JS16 系列包括静态 RAM。RAM 中 0x0100 以下的地址可以使用更高效的直接寻址模式访问，而这一区域中的任何单一比特可以通过位操作指令（BCLR、BSET、BRCLR 和 BRSET）访问。首选的方式是在这一区域中查找 RAM 最频繁访问的程序变量。

在 MCU 处于低功耗的等待、Stop2 或 Stop3 模式时，RAM 会保留数据。加电启动时，RAM 中的内容不会被初始化。如果电源电压没有降低到支持 RAM 保留的电压最低值以下，RAM 数据就不会受到任何复位的影响。

为了实现与 M68HC05MCU 的兼容性，HCS08 会将堆栈指针复位为 0x00FF。在 MC9S08JS16 系列中，将堆栈指针重新初始化到 RAM 顶部，以便经常被访问的 RAM 变量和位寻址的程序变量可以使用直接寄存器。复位初始化程序 （其中的 RamLast 等于飞思卡尔通用文件中 RAM 的最高地址）中包含以下两个指令序列。

```
LDHX    #RamLast+1    ;point one past RAM
TXS                   ;SP<-（H:X-1）
```

在启用了安全性的情况下，RAM 被认为是一种安全的内存资源，不能通过 BDM 或从非安全内存中执行代码来访问。若欲了解有关安全特性的更详尽描述，请参见 4.7，“安全性”。

4.4 USB RAM

4.5 引导程序ROM

引导程序 ROM 中装有擦除和编写 FLASH 存储器的代码。引导程序 ROM 从一个类如一台 PC 机的 USB 主设备处提供了快速而可靠的 FLASH 擦除和编写的过程

飞思卡尔提供了一个 PC GUI（图形用户接口），可以通过 UCB 接口与 ROM 中的引导程序进行通讯（PC 作为主 USB 设备）。

用 PC GUI 工作，用户可以：

整体除整个 FLASH 阵列

部分擦除 FLASH 阵列——除开始 1KB 外擦除所有 FLASH 块

对 FLASH 编程

复位 MCU

注：

如果引导程序功能被使用，MCU 提供电压必须高于 4V。内部 USB 3.3V 电压调节器在进入引导程序模式时将被使能。

注：

USB 引导程序需要一个外部振荡器，且频率必须为 2 MHz，4 MHz，6 MHz，8 MHz，12 MHz，或 16 MHz。引导程序代码可以自动识别外部振荡器。如果没有使用引导程序，就没有这样的限制。

注：

对引导程序的 USB 描述符是固定的：VID 是 0x15A2，PID 是 0x0038.用户程序只能使用它自身的描述符。

4.5.1 外部信号描述

引导程序 ROM 的$\overline{\text{BLMS}}$引脚决定了 MCU 是否在上电复位期间直接进入引导程序模式。这个引脚只在上电复位（POR）期间被检查。

表 4-5 显示了引导程序 ROM 的信号特征。

表4-5 信号特征

信号	功能	I/O
BLMS	引导程序模式选择	1

4.5.2 操作模式

任何复位之后，MCU 跳至引导程序 ROM 地址，在那里几个限制因子被检查来决定是跳至引导程序代码还是用户代码。引导程序 ROM 可以在引导程序模式或用户模式下被访问。这一节描述了在这两种模式下的非法操作和保护机制。

以下四个将会在每次 MCU 复位以后被检查

- $\overline{\text{BLMS}}$引脚
- SIGNATURE 信号量字节
- FLASH 块 CRC 校验和
- CRC 旁路字节

4.5.2.1 引导程序模式

引导程序模式可以在以下四种条件中进入：

1 .上电复位（POR）期间，当$\overline{\text{BLMS}}$引脚为低电平且 BKGD/MS 没有被拉低，将无条件的进入引导程序模式。

2. 上电复位（POR）期间，当$\overline{\text{BLMS}}$引脚和 BKGD/MS 引脚为高电平时，一个 CHECKSUMBYPASS FLASH 空间将被检查。如果它的值不等于 0x00 或 0xFF，则进入引导程序模式。

3. 上电复位（POR）期间，当$\overline{\text{BLMS}}$引脚和 BKGD/MS 引脚为高电平时，一个 CHECKSUMBYPASS FLASH 空间将被检查。如果它的值等于 0xFF，将计算 FLASH 阵列的 CRC 值并与一个 16 位 FLASHCRC 字相比较。如果不匹配，则进入引导程序模式。

4. 复位过后（除了上电复位以外），系统将检查 SIGNATURE 信号量。如果等于 0xC3，则进入引导程序模式。

4.5.2.2 用户模式

用户模式可以在以下三种条件下进入：

1. 上电复位（POR）期间，当$\overline{\text{BLMS}}$引脚和 BKGD/MS 引脚为高电平，且 CHECKSUMBYPASS 字节等于 0x00 时，进入用户模式。

2. 上电复位（POR）期间，当$\overline{\text{BLMS}}$引脚和 BKGD/MS 引脚为高电平，且 CHECKSUMBYPASS 字节等于 0xFF 时，将计算 FLASH 阵列的 CRC 值并与一个 16 位 FLASHCRC 字相比较。如果结果匹配，进入用户模式。

3.当一个复位发生（除了上电复位以外），如果 SIGNATURE 信号量不等于 0xC3，则进入用户模式。

4.5.2.3 激活背景模式和引导程序模式仲裁

上电复位（POR）期间，当$\overline{\text{BLMS}}$引脚和 BKGD/MS 引脚为低电平，则进入激活背景模

式。

4.5.2.4 禁止FLASH保护

通过向 FPROTD（FLASH 保护失败寄存器）地址连续写入 0x55 和 0xAA，然后将 FPROT 的 FPDIS 位置 1，来禁止 FLASH 区段的保护。

4.5.3 FLASH存储映像

引导程序的通用 FLASH 存储映像显示于图 4-2。

- FLASH 块校验和存储于 0xFBB8（校验和高字节）和 0xFBB9（校验和低字节）
- 校验和旁路信息存储于 0xFFBA
- FLASH 部分擦除信号量存储于 0xFFBE

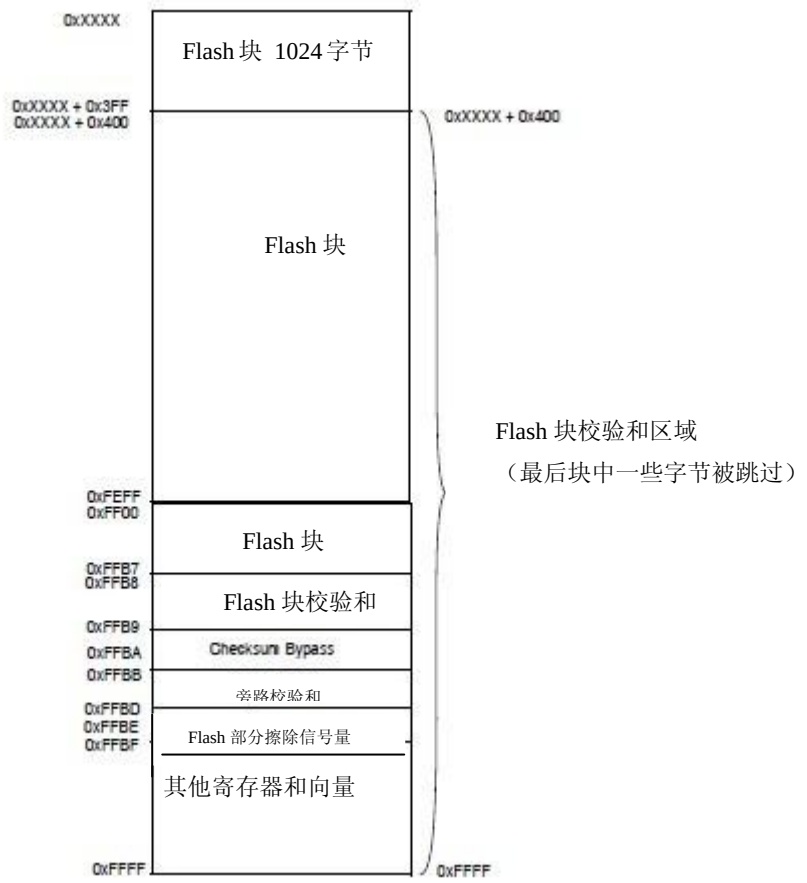


图4-2 通用FLASH存储器映像

旁路校验和的值靠用户编程设定。这个值只在上电复位之后被引导程序检查。这个字节的旁路校验和的值显示了 FLASH 块校验和是否会被计算，和在开始的 1KB 地址空间之后的 FLASH 区段是否被用作 EEPROM。当这个值不是 0x00 或 0xFF 时，不计算 FLASH 块的校验和，系统直接跳至引导程序入口。当值等于 0xFF 时，在上电复位之后 FLASH 块的校验和将会被计算。当值等于 0x00 时，不计算 FLASH 块的校验和，MCU 跳至用户程序入口处，在开始的 1KB 地址空间之后的 FLASH 区段将被用作 EEPROM。

芯片的出厂值为 0xFF。

注：

开始的 1KB FLASH 段不进行 CRC 校验计算，所以用户可以将这个区域用作 EEPROM。如果用户需

要用除此之外的其他 FLASH 空间，他们就需要重新计算 CRC，且重新编写 CRC 校验和来保证上电复位之后对地址空间的可靠访问。

4.5.4 引导程序操作

这节描述了引导程序机制和引导程序流程图。

引导程序位于引导程序 ROM。用户可以对 FLASH 进行擦除和写入当：

- 进入引导程序模式
- 计算出的 FLASH 块校验和和预置的 FLASH 块校验和在上电复位之后不匹配。
- 寄存器中的 SIGNATURE 信号量匹配。

4.5.4.1 FLASH块校验和

上电复位（POR）时，如果 BLMSS=0 且旁路校验和的值为 0xFF 时，引导程序将计算 FLASH 的校验和。这个校验和从开始 1KB 之后的 FLASH 开始算起到 0xFFFF 为止（FLASH 最末页中部分字节被跳过）。开始的 1KB FLASH 不包含在校验和当中，所以用户可以将它当做伪 EEPROM 使用。这个计算出来的校验和将和事先写入 FLASH 中两个字节（FLASHCRC）的值相比较，如果校验和匹配，之前的引导程序操作就是成功的，MCU 跳至用户程序的入口处开始执行用户代码。如果校验和不匹配，它就会跳至引导程序入口处等待指令。

FLASH 块校验和的计算使用 16 位 CRC。

JS16FLASH 块校验和的范围是 0xC400-0xFFAD 和 0xFFC0-0xFFFF，JS8FLASH 块校验范围是 0xE400-0xFFAD 和 0xFFC0-0xFFFF。

4.5.4.2 SIGNATURE 信号量寄存器

在常规复位期间，引导程序检查 SIGNATURE 信号量寄存器。如果 SIGNATURE=0xC3，则 MCU 跳至引导程序入口处等待指令。如果不是，则它跳至用户程序入口处开始执行用户代码。

用户被要求在他们的程序代码中提供一个机制来将 SIGNATURE 的值设为 0xC3，以及开始一次复位当他们希望在一次用户代码被成功写入之后重新进入引导程序模式。可选方案是，进入 BKGD 模式，SIGNATURE 可以用 BDM 指令来更新，BKGD 引脚为高电平时可以引发复位。

4.5.4.3 FLASH部分擦除信号量

FLASH 部分擦除的值依靠用户写入。只有当这个值被写为 0x00 时，引导程序支持部分擦除 FLASH 阵列命令。

芯片的出厂值为 0xFF。

4.5.4.4 流程图

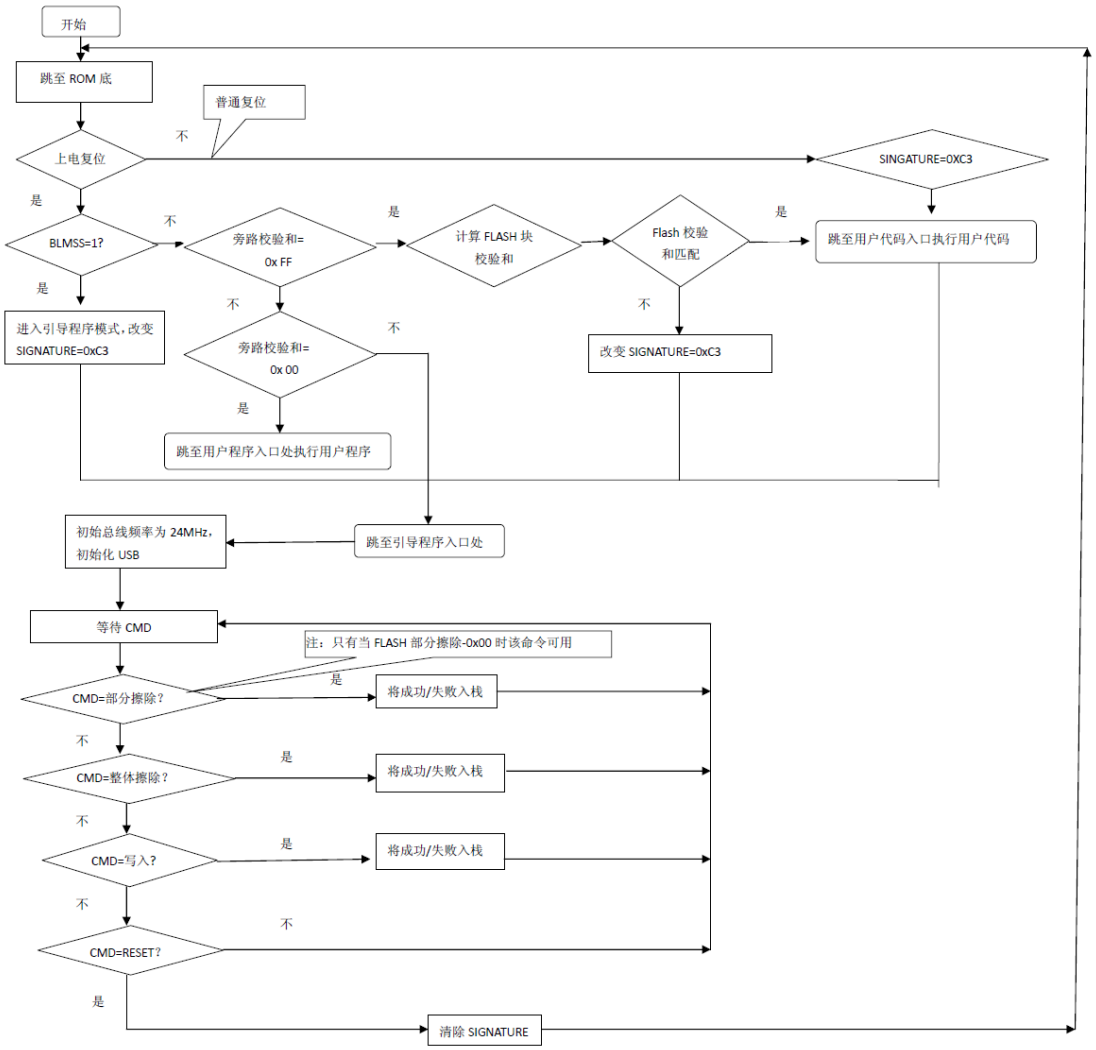


图4-3引导程序流程图

4.6 FLASH

FLASH 存储器主要用于保存程序。在线编程使正在运行的程序可以在应用产品的最终组装完成后上载到 Flash 中。我们可以通过单线背景调试接口对阵列进行编程。由于 FLASH 擦除和编程操作不需要特殊的电压，所以也可以通过其他软件控制的通信路径来实现应用编程。有关在线和应用内编程的更详尽描述，请参见“HCS08 系列参考手册，第 1 卷”（ 飞思卡尔半导体文件，编号 HCS08RMv1）。

4.6.1 特点

FLASH 存储器特点包括：

- FLASH 大小
  ——MC9S08JS16- 16384 字节（32 页，每页 512 字节） -
  ——MC9S08JS8-8192 字节（16 页，每页 512 字节）
- 单电源写入和擦除

- 命令界面提供快速写入和擦除操作
- 在典型的电压和温度下有最多可达 100000 个写入/擦除周期
- 灵活的块保护
- FLASH 和 RAM 的安全特性
- 自动关闭电源以获得低频读取

4.6.2 写入和擦除时间

在接受任何写入或擦除命令前，必须通过写 FLASH 时钟分频寄存器 （FCDIV） 以将 Flash 模块的内部时钟设置为 150kHz ～200kHz 之间的频率（f_{FCLK}）（请参见 4.8.1“FLASH 时钟分频寄存器（FCDIV）”）。这个寄存器只能写入一次，因此这一写入操作通常是在复位初始化过程中执行的。如果 FSTAT 寄存器的 FACCERR 访问错误标志位被置位，FCDIV 不能被写入。用户必须确保在写入 FCDIV 寄存器之前没有将 FACCERR 置为 1。命令处理器使用时钟 （$1/f_{FCLK}$）的一个周期来对写入和擦除脉冲定时。命令处理器利用这些定时脉冲的一个整数值来完成编程或擦除命令。

表 4-6 给出了写入和擦除时间。总线时钟频率和 FCDIV 决定 FCLK 的频率（f_{FCLK}）。一个 FCLK 周期为 $t_{FCLK} = 1/f_{FCLK}$。定时器显示为多个 FCLK 循环和一个绝对时间（t_{FCLK} = 5 μs）。显示的写入和擦除时间包括命令状态机的开销及写入和擦除电压的启用及禁用的时间。

表4-5 编程和擦除时间

参数	FCLK循环	FCLK = 200 kHz 时的时间
字节程序	9	45 μs
批量程序	4	20 μs[1]
页擦除	4000	20 ms
整体擦除	20,000	100 ms

1 包括启动/终止开销

4.6.3 编程和擦除命令的执行

下面列出执行任何命令的步骤。在执行命令之前 FCDIV 寄存器必须被初始化，任何错误标志都需要被清除。命令执行的步骤是：

1、将一个数据值写入到 Flash 阵列中的一个地址中。该地址和写入的数据信息被锁定到 Flash 接口上。这一写入操作是任何命令序列中要求的第一步。对于擦除和空白检查命令，这些数据的值并不重要。对于页擦除命令，地址可以是将要擦除的 Flash512 字节页面中的任何地址。对于整体擦除和空检查命令，地址可以是 Flash 存储器中的任何地址。512 字节的整个页面是 FLASH 中能被擦除的最小块。

注意：

在擦除操作成功之后，不要对 FLASH 中的任意字节写入超过一次。如果没有先对 FLASH 存储器进行整体擦除或擦除字节所在的页面，对已经写入的字节重新写入是不被允许的。写入前没先擦除可能会扰乱储存在 FLASH 中的数据。

2、将命令代码写入到 FCMD 中。 6 个有效的命令分别是空白检查 （0x05）、字节写入（0x20）、突发模式写入 （0x25）、页擦除（0x40）、整体擦除 （0x41）和页擦除终止 （0x47）。命令代码被锁定到命令缓冲器中。

3、将一个 1 写入到 FSTAT 中的 FCBEF 位上，以清除 FCBEF 并发起命令 （包括其地址和数据信息）。

在写内存阵列之后到写 1 清零 FCBEF 发起完整命令之前的任何时候，可以通过向

FCBEF 中写入一个 “0”，来使部分命令序列失效。以这种方式终止一个命令会将 FACCERR 访问错误标记置 1，而这个标记必须在开始一个新命令之前清零。

整个过程必须遵守严格的监控流程，否则命令将不会被接受。通过这种方式可以最大限度地降低无意中修改 FLASH 存储器内容的可能性。命令完整标记 （FCCF）用于指示一条命令是否完成。通过清除 FCBEF 来开始执行命令，命令序列被完成。 图 4-4 是除突发模式写入和页擦除终止以外的所有命令的执行流程。在使用任何 FLASH 命令之前，FCDIV 寄存器必须初始化。复位后，这只能执行一次。

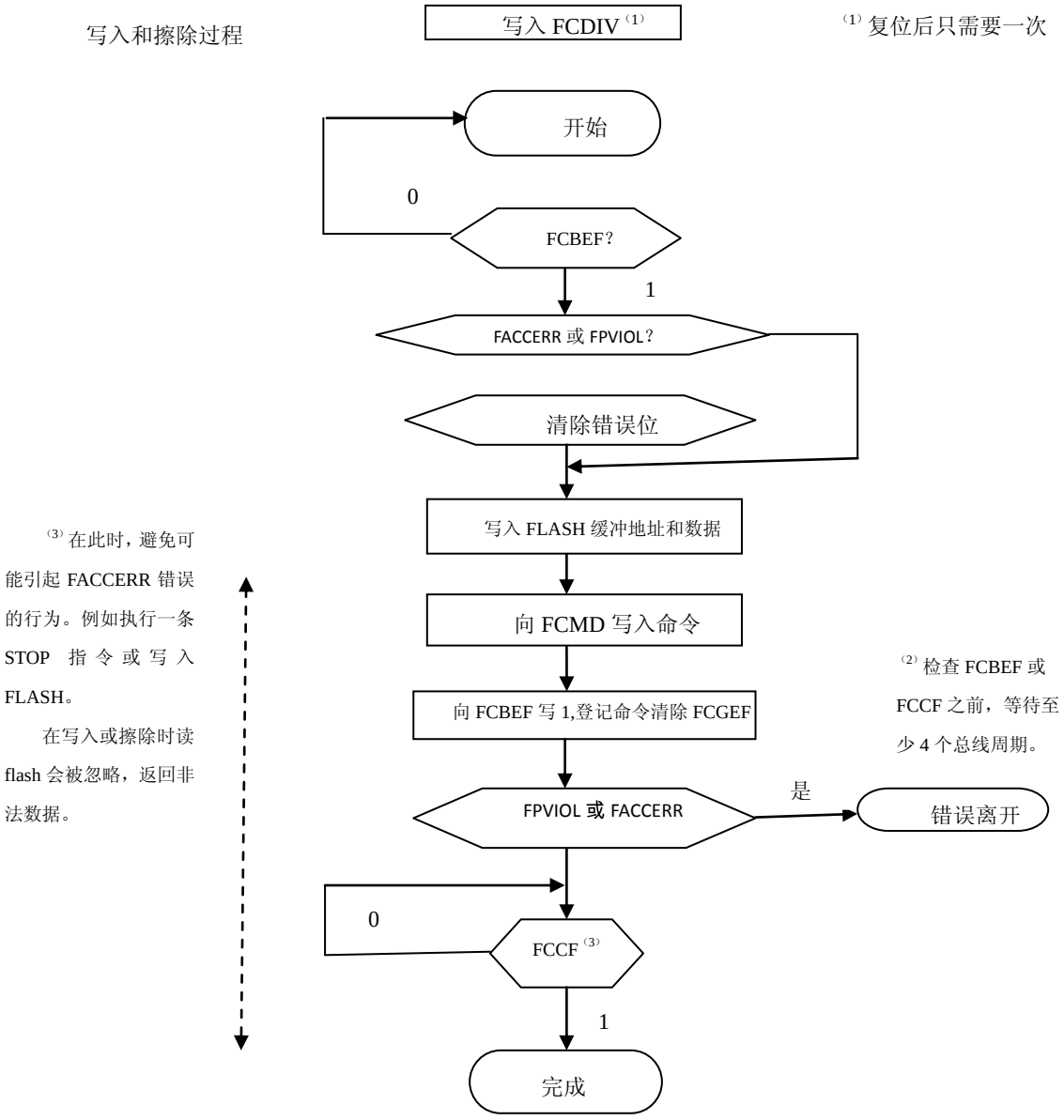


图4-4 FLASH写入和擦除流程图

4.6.4 批量写入

突发模式写入命令用来写入连续的字节，相比较标准写入命令需要较少的时间。因为不需要屏蔽两次写入操作之间加在 Flash 阵列的高电压。通常情况下，当发起一个写入或擦除命令，必须启用与 Flash 相关的一个内部电荷泵用于为阵列提供高电压。命令执行完成后，该电荷泵会被关闭。如果满足下面两个条件，当突发模式写入操作完成，电荷泵仍然保持打

开：

- 在本次操作完成之前下次，下次突发模式写入命令已经排队等候。
- 下一个连续地址选择了和当前正在写入的字节具有相同的物理行的字节。FLASH 内存行包括 64 个字节。每行的字节通过地址的 A5 到 A0 选择。一个新行的起始地址中的 A5 到 A0 全零。

在突发模式中写入一系列连续字节中的第一个所花的时间与正常模式中写入一个字节的时间是相同的，假设满足上述两个条件，后续字节将以突发模式写入。在下一个连续的地址是另一行的情况下，字节的写入时间将会是标准时间而不是突发时间。这因为加在阵列上的高电压将被屏蔽然后重新开启。如果在第一次命令结束之前新的命令没有排在队列中，电荷泵将被屏蔽，阵列上的高电压将被移除。

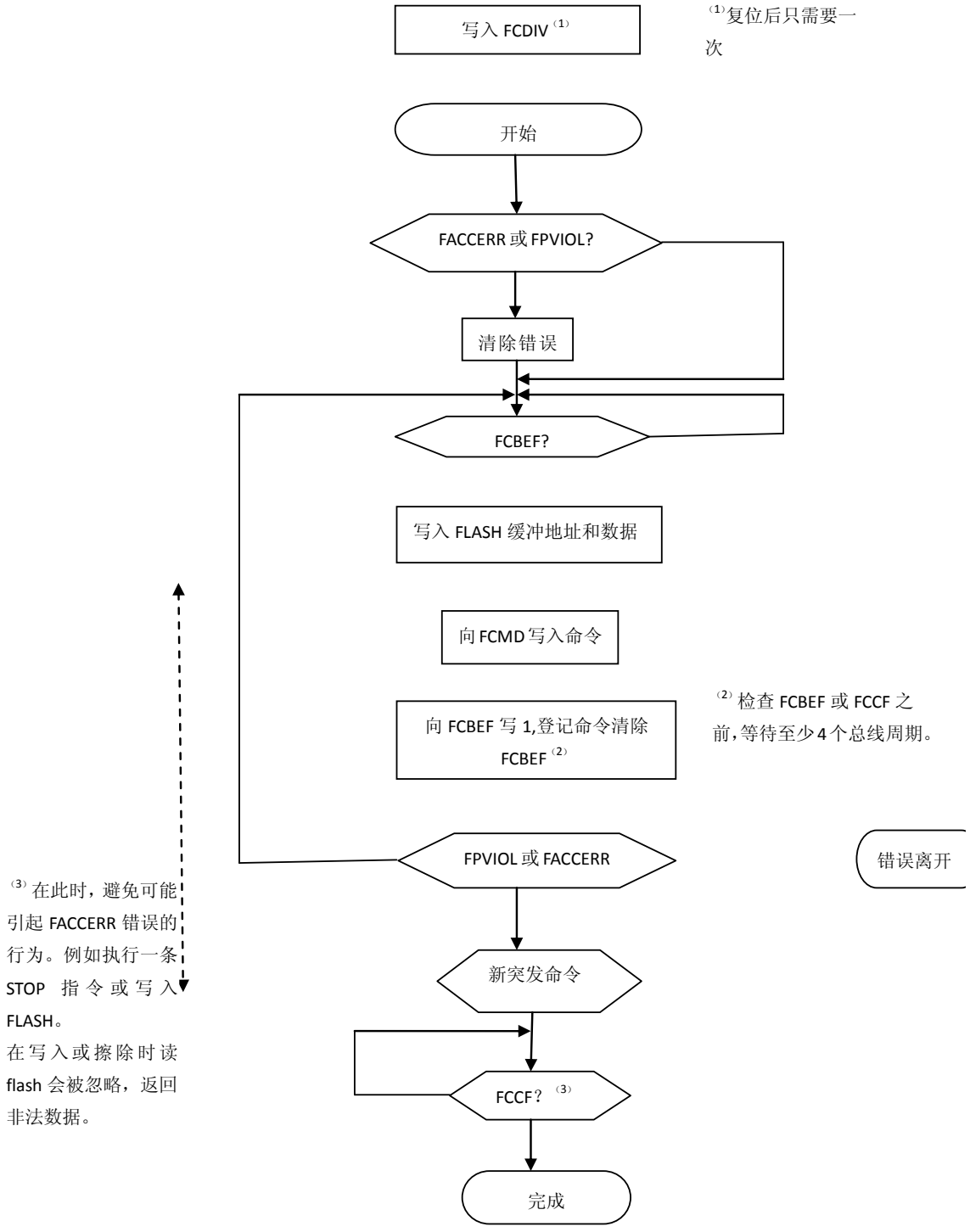


图4-5 FLASH突发写入流程图

4.6.5 访问错误

无论何时只要违反了命令执行协议,将产生访问错误.

任何下列动作将会导致 FSTAT 中的访问错误标志（FACCERR）置位。在任何命令执行之 前，FACCERR 位必须通过对 FSTAT 中的 FACCERR 写 1 的方式清除。

- 在写 FCDIV 寄存器设置内部 FLASH 时钟频率之前,写 FLASH 地址

- 写 FLASH 地址的同时 FCBEF 没有被设置（直到命令缓存清空,才能开始一个新命令.）
- 在登记先前的命令之前,第二次写 FLASH 地址（每一个命令只对 FLASH 写一次.）
- 在登记先前的命令之前,第二次写 FCMD（每一次命令只对 FCMD 写一次.）
- 写 FLASH 地址之后设置除 FCMD 以外其他任何 FLASH 控制寄存器
- 写（0x05, 0x20, 0x25, 0x40, or 0x41）之外的任何命令代码到 FCMD
- 在写命令到 FCMD 后访问（读或写）任何 FLASH 控制寄存器而不是在写入 FSTAT（为了清除 FCBEF 和登记命令）
- 当正在执行写入和擦除命令时 MCU 进入停止模式.（命令失败.）
- 当 MCU 处于安全状态时，用背景调试命令写字节写入、突发模式写入和页擦除这些命令（当 MCU 处于安全状态时，背景调试控制器只能进行空白检测和整体擦除命令）。
- 向 FCBEF 位写 0 取消一个未完成的命令。

4.6.6 FLASH块保护

块保护特点防止了 FLASH 的保护块被写入和擦除改变。块保护由 FLASH 保护寄存器（FPROT）控制。当使能，块保护从任何低于 FLASH 最后字节，0xFFFF，的任何 512 个字节边界开始。（参看 4.8.4，“FLASH 保护寄存器（FPROT 和 NVPROT）”）。

退出复位后，FPROT 用位于 FLASH 存储器中非易失寄存器块的 NVPROT 位置的内容加载。FPROT 不能用应用软件直接改变，所以一个失控的程序不能改变块保护的设置。因为 NVPROT 在 FLASH 最后 512 字节之中，如果任何数量的存储器被保护，NVPROT 自己也被保护，并且不能被应用软件（有意的或无意的）改变。背景调试命令能够写 FPROT，这是一种擦除和编程保护了的 FLASH 存储器的方法。

下图举例说明了块保护机制。FPS 位用作未保护存储区最后一个地址的高位。如图所示地址由 FPS7:FPS1 连接上逻辑 1 组成。例如，为了保护存储器中最后的 1536 个字节（地址从 0xFA00 到 0xFFFF）， FPS 位需设置为 1111100，产生的值 0xF9FF 是存储器中最后未保护的地址。除了对 FPS 位写入正确的值，FPDIS（NVPROT 中的 0 位）必须被写入为逻辑 0 以确保块保护。因此值 0xF8 必须写入到 NVPROT 以确保地址 0xFA00 到 0xFFFF 被保护。

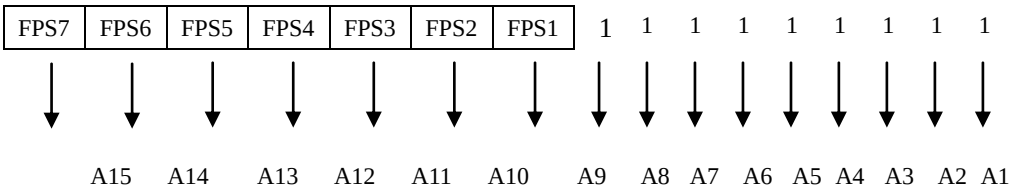


图4-6 块保护机制

块保护的一个用途是为引导程序提供一个保护了的 FLASH 存储区域。这样引导程序就可以用于擦除并且重新写剩余的 FLASH 存储器。因为引导程序被保护，所以即使 MCU 在写入或者擦除操作过程中掉电，该程序也是完整的。

4.6.7向量重定向

无论何时使能块保护,复位和中断矢量将会被保护。矢量重定向允许用户不用未保护引导程序和复位矢量空间就能修改中断矢量信息。矢量变向通过对位于 0xFFBF 地址的 NVOPT 中 FNORED 位写入为 0 使能。为产生重定向，通过对位于 0xFFBD 的 NVPROT 写入使至少

一些而不是全部 FLASH 存储区受到块保护。所有的中断矢量(位于存储器 0xFFC0–0xFFFD）都会重定向，而复位矢量（位于存储器 0xFFFE–0xFFFF）则不会重定向。

例如，如果 FLASH 的 512 字节受到保护，保护的区域地址是从 0xFE00 到 0xFFFF。中断矢量（0xFFC0–0xFFFD）重定向到位于 0xFDC0–0xFDFD。以 TPM1 溢出中断为例，0xFDE0:FDE1 中的值而不是 0xFFE0:FFE1 中的值将被中断向量使用。这就允许用户将新的程序代码和新的向量值写入 FALSH 中未被保护的部分，而包含默认向量地址的被保护区域不被改写。

4.7 安全性

MC9S08JS16 使用硬件电路来防止对 FLASH 存储器和 RAM 存储器的数据的非法访问。当使用安全机制时，FLASH 和 RAM 被认为是被保护资源资源。直接页寄存器，高页寄存器和背景调试控制器被认为是非被保护资源。在被保护存储器中执行的程序可正常访问任何 MCU 存储器和资源。试图用非被保护存储空间中执行的程序或通过背景调试接口对一个被保护存储器的访问将会被屏蔽（写被忽略,读返回 0）。

是否采取安全机制取决于非易失性寄存器 FOPT 中两个位（SEC01:SEC00）的状态。在复位过程中，非易失 的 NVOPT 的内容从 FLASH 复制到在高页面寄存器空间的正在工作的 FOPT 寄存器。用户能在对 FLASH 存储器写入的同时通过写入 NVOPT 来采取安全机制。1:0 状态解除安全机制，其它三种组合则采取安全机制。注意擦除态（1:1）也让 MCU 被保护。在开发过程中，无论何时只要 FLASH 被擦除，立即对 NVOPT 中的 SEC00 位编程为 0，使 SEC01 :SEC00 = 1:0。这让 MCU 在随后的复位后保持非被保护状态。

当 MCU 处于被保护态时，片内调试模块将不能被使能。这个独立的背景调试控制器仍然用于对未保护资源执行背景存储器访问命令。

用户可以通过 8 字节的后门密钥来允许或者禁止安全机制。如果 NVOPT 或者 FOPT 中的非易失性位 KEYEN 是 0，后门密钥是无效的，这时如果没有擦除整个的 FLASH 就没有方法离开安全模式。如果 KEYEN 是 1，一个在安全模式下的用户程序可以通过以下的方法暂时离开安全模式：

用户可以通过 8 字节的后门密钥来允许或者禁止安全机制。如果 NVOPT 或者 FOPT 中的非易失性位 KEYEN 是 0，后门密钥是无效的，这时如果没有擦除整个的 FLASH 就没有方法离开安全模式。如果 KEYEN 是 1，一个在安全模式下的用户程序可以通过以下的方法暂时离开安全模式：

1.对 FCNFG 寄存器中 KEYACC 写 1。这让 FLASH 模块对后门比较钥匙位置（NVBACKKEY 到 NVBACKKEY+7）的写操作解释为与钥匙相比较的值，而不是 FLASH 编程或者擦除命令的第一步。

2. 写用户进入钥匙的值到 NVBACKKEY 到 NVBACKKEY+7 的位置。这些写必须按照开始于 NVBACKKEY,结束于 NVBACKKEY+7 上的值这样的顺序。这些写中将不会用到 STHX，因为这些写不能在相邻总线周期中完成。用户软件通常从 MCU 系统外面通过一种通讯接口，例如串口,获得钥匙代码。

3. 写 FCNFG 寄存器中的 KEYACC 为 0。如果刚写下的 8 字节钥匙与存储在 FLASH 中的钥匙相匹配， SEC01 :SEC00 自动改变为 1:0，在下次复位前都会禁用安全机制。

安全密钥只能从安全存储器（RAM 或 FLASH）写入。如果没有安全用户程序的协调不能由背景调试命令进入。

后门密钥（NVBACKKEY 到 NVBACKKEY+7）分布在 FLASH 存储器的非易失性寄存器空间中，因此用户可以象写其他 FLASH 寄存器一样写这些寄存器。这个非易失性的寄存

器与复位和中断向量一起都位于同样的一块 512 字节的 FLASH 存储区域中，所以块保护这块区域也就保护了后门密钥。块保护不能通过用户的应用程序改变，因此，如果向量空间是块保护空间，这个后门安全密钥机制永远不能改变块保护、安全设置和后门密钥。

安全模式可以通过背景调试接口按照下面的步骤来消除：

1. 通过写 FPROT 屏蔽任何块保护。在 MC9S08JS16 系列中，当 FPROTD 被置位，FPROT 可以被改变。

2. 如果必要,则整体擦除 FLASH。

3. 空检查 FLASH。如果 FLASH 被完全擦除，直到下次复位不采取安全机制。

为避免下次复位后返回安全模式，写入 NVOPT 使 SEC01 :SEC00 = 1:0。

4.8 FLASH寄存器和控制位

FLASH 模块有 9 个 8 位的寄存器位于高地址页寄存器空间，2 个位于 FLASH 存储器的非易失性寄存器空间，当复位时，内容被复制到相应的高页区域的控制寄存器。同样也有一个 8 字节的密钥在 FLASH 存储器中。表 4-3 和表 4-4 列出了所有 FLASH 寄存器的绝对地址。这里通过这些寄存器和控制位的名称来引用它们。通常，Freescale 提供一个通用文件或头文件把它们的名称翻译为绝对地址。

4.8.1 FLASH时钟分频寄存器（FCDIV）

该寄存器的第 7 位是一个只读标记。 6:0 位可以在任何时候读取但只能写入一次。在开始任何擦除或写入操作之前，写入该寄存器以将非易失性内存系统的时钟频率设置在可接受的限度内。

	第7位	6	5	4	3	2	1	第0位
读	DIVLD	PRDIV8	DIV5	DIV4	DIV3	DIV2	DIV1	DIV0
写								
复位	0	0	0	0	0	0	0	0

=未使用或保留

图4-7 FLASH时钟分频寄存器（FCDIV）

表4-7 FCDIV寄存器字段描述

字段	描述
7 DIVLD	**Divisor 加载状态标识** — 当置 1 时，这个只读状态标记指出 FCDIV 寄存器自从复位后已有写入。复位会清除该位而且第一次写入该寄存器的操作将导致该位被置 1，不管写入什么数据。 0 FCDIV 自复位后没有写入；Flash 和 EEPROM 的擦除和写入操作被禁止。 1 FCDIV 自复位后已写入；Flash 和 EEPROM 的擦除和写入操作已开启。
6 PRDIV8	**Flash时钟8预分频 （分频）** 0 Flash 钟分频器的时钟输入为总线速率时钟。 1 Flash 时钟分频器的时钟输入为总线速率时钟除以8。
5:0 DIV[5:0]	**时钟分频位——**FLASH时钟分频器通过DIV5：DIV0这六位加上1的值来对总线时钟频率（或者当PRDIV8=1时是总线时钟频率的八分之一）进行分频。在对FLASH进行操作时，内部时钟必须降到150KHz~200KHz。擦除/写入操作的时序脉冲是内部FLASH工作时钟的一个时钟周期，所以擦/写的时间相应的在6.7μs~5μs。自动的写入逻辑使用这个时钟周期的整数倍时间完成擦除或写入操作。参考等式4-1，等式4-2和表4-7。

if PRDIV8 = 0 — fFCLK = fBus ÷ （DIV + 1） Eqn. 4-1

if PRDIV8 = 1 — fFCLK = fBus ÷ （8 × （DIV + 1）） Eqn. 4-2

表 4-8 列出了对 PRDIV8 和 DIV5:DIV0 设置不同数值时的 FLASH 时钟频率。

表4-7 FLASH时钟分频器设置

f_{BUS}	PRDIV8（二进制）	DIV（十进制）	f_{FCLK}	编程/擦除时序脉冲（最小5 μs, 最大6.7 μs）
24 MHz	1	14	200 kHz	5. μs
20 MHz	1	12	192.3 kHz	5.2 μs
10 MHz	0	49	200kHz	5 μs
8 MHz	0	39	200kHz	5 μs
4 MHz	0	19	200kHz	5 μs
2 MHz	0	9	200kHz	5 μs
1 MHz	0	4	200kHz	5 μs
200 kHz	0	0	200kHz	5 μs
150 kHz	0	0	150 kHz	6.7 μs

4.8.2 FLASH选项寄存器（FOPT和NVOPT）

在复位中，在非易失位置中 NVOPT 的内容从 FLASH 复制到 FOPT。第 2 至 5 位不使用并总是读出 0。在任何时间这个寄存器都可能被读出，但写入没有意义或影响。要改变这个寄存器的值，照常擦除和重写入 FLASH 存储器中 NVOPT 位置，然后发起一个新的 MCU 复位。

	第7位	6	5	4	3	2	1	第0位
读	KEYEN	FNORED	0	0	0	0	SEC01	SEC00
写								
复位	在复位时，这个寄存器从非易失性区域NVOPT中装载							

=未使用或保留

图4-8 FLASH选项寄存器（FOPT）

表4-9 FOPT寄存器字段描述

字段	描述
7 KEYEN	**使能后门钥匙机制** — 当这位为0，后门钥匙机制不能用来解除安全机制。后门钥匙机制仅能从用户（保密的）固件访问。BDM命令不能用来写将解锁后门钥匙的钥匙比较值。更多关于后门钥匙机制的细节，参看4.7，“安全性”。 0 不允许后门钥匙访问。 1 如果用户固件写了一个8字节的值，匹配非易失的后门钥匙（按顺序NVBACKKEY到NVBACKKEY+7），直到下次MCU复位，安全机制被暂时取消。
6 FNORED	**屏蔽矢量变向**— 当此位为1，屏蔽矢量重定向。 0 使能矢量重定向。 1 屏蔽矢量重定向。
5:0 SEC0[1 :0]	**安全状态代码** — 这两位域确定了如表4-10所示的MCU保密状态。当MCU处于安全态时，RAM和FLASH存储器中的内容不能通过来自任何非被保护资源的包括背景调试接口的指令来访问。关于保密的更多细节，参看4.7， “保密性”。

表4-10 安全状态码

SEC01:SEC00	描述
0：0	保密的
0：1	保密的
1：0	非保密的
1：1	保密的

在一次成功的后门密钥进入或 FLASH 空白检测后 SEC01:SEC00 变为 1:0。

4.8.3 FLASH配置寄存器（FCNFG）

第 5 位到第 7 位可以在任何时间被读写。第 0 位到第 4 位通常读为 0 且不能被写。

	第7位	6	5	4	3	2	1	第0位
读	0	0	KEYACC	0	0	0	0	0
写								
复位	0	0	0	0	0	0	0	0

=未使用或保留

图4-9 FLASH配置寄存器（FCNFG）

表4-11 FOPT寄存器域描述

域	描述
5 KEYACC	**写访问密钥允许位**——该位使能写访问密钥。更详细的描述见4.6节安全性。 0 写0xFFB0–0xFFB7被认为是FLASH写入或擦写命令的开始。 1 写NVBACKKEY（0xFFB0–0xFFB7）被认为是进行密码比较。

4.8.4 FLASH保护寄存器（FPROT和NVPROT）

在复位过程中，非易失位置 NVPROT 的内容从 FLASH 复制到 FPROT。在任何时间这个寄存器可被读，但用户程序的写入没有任何意义。背景调试命令可以对 FPROT 进入写入。

	7	6	5	4	3	2	1	0
读	FPS7	FPS6	FPS5	FPS4	FPS3	FPS2	FPS1	FPDIS
写	(1)	(1)	(1)	(1)	(1)	(1)	(1)	(1)
复位	在复位时，这个寄存器从非易失性区域NVOPT中装载							

（1） 背景命令可以改变FPRO寄存器中的这些位

图4-10 FLASH保护寄存器（FPROT）

表4-12 FPORT寄存器字段描述

字段	描述
7:1 FPS[7:1]	**FLASH保护选择位** — 当FPDIS = 0，这个7位的域确定了在FLASH高地址处的未保护的区段的结束地址。保护的FLASH位置不能被擦除或者写入。
0 FPDIS	**屏蔽FLASH保护** 0 由FPS[7:1]确定的FLASH块为受保护的块.（不被允许的写入和擦除）。 1 没有FLASH块被保护

4.8.5 FLASH状态寄存器（FSTAT）

位 3，1，0 总是读出 0，并写入操作没有意义或影响。其余 5 位是状态标志位可以在任何时间被读出。对这些位写入有特别的意义，在位说明中被描述。

	第7位	6	5	4	3	2	1	第0位
读	FCBEF	FCCF	FPVIOL	FACCERR	0	FBLANK	0	0
写								
复位	1	1	0	0	0	0	0	0

=未使用或保留

图4-11 FLASH状态寄存器（FSTAT）

表4-13 FSTAT寄存器字段描述

字段	描述
7 FCBEF	**FLASH命令缓存空标志** — FCBEF位用来开始命令。当执行突发时，这表明了命令缓存为空，一个新的命令序列能被执行。向FCBEF位写1或一个突发写入命令进入写入数组可以清FCBEF位。只有突发写入命令可以被缓冲。 0 命令缓存满（没准备好另外命令） 1 一个新的突发写入命令能够被写到编程缓存
6 FCCF	**FLASH命令完成标志** — 当命令缓存为空且没有正在执行的命令，FCCF自动置1。当开始一个新的命令，FCCF自动清零（当通过对FCBEF写1登记一个命令）。写FCCF没有意义和影响。 0 进行中的命令 1 所有命令结束
5 FPVIOL	**保护侵犯标志Flag** — 当一个命令被写入试图擦除或写入一块被保护的块时（忽略错误命令），FPVIOL位被自动置位。通过向FPVIOL写1可以自动清除FPVIOL位。 0 没有保护被侵犯 1 试图擦除或者写入一个受保护的位置
4 FACCERR	**访问错误Flag**—当没有遵守正确的命令顺序（忽略错误命令），如果在FCDIV寄存器初始化之前试图进行写入或者擦除操作，或者命令正在进行时MCU进入停止状态，FACCERR自动置位。更多关于针对对访问错误的正确的动作的细节讨论，参看4.6.5， “访问错误”。FACCERR通过对其写1被清除，.对FACCERR写0没有意义或者影响。 0 没有访问错误 1 产生一个访问错误
2 FBLANK	**FLASH空白标志位** — 当全部FLASH被确认为擦除，在一个空检查完成时FBLANK自动置1。FBLANK通过清除FCBEF来清除，然后写新的有效的命令.。对FBLANK写0没有意义或者影响。 0 在一个空检查命令完成和FCCF = 1，FBLANK = 0表明FLASH阵列没有完成擦除 1 在一个空检查命令之后FCCF = 1， FBLANK = 1表明FLASH阵列完全擦除（都是0xFF）

§4.8.6 FLASH命令寄存器（FCMD）

表 4-15 中说明了用户模式中仅仅能识别的五种命令代码。关于 FLASH 编程和擦除操作中的细节讨论，参考 4.6.3， “写入和擦除命令的执行”。

	第7位	6	5	4	3	2	1	第0位
读 写	FCMP7	FCMP6	FCMP5	FCMP4	FCMP3	FCMP2	FCMP1	FCMP0
复位	0	0	0	0	0	0	0	0

图4-12 FLASH命令寄存器（FCMD）

表4-14 FCMD寄存器字段描述

字段	描述
FCMD[7:0]	FLASH命令位 — 见表4-15

表4-15 FLASH命令

命令	FCMD	等同的文件标签
空检查	0x05	mBlank
字节写入	0x20	mByteProg
字节写入—突发写入	0x25	mBurstProg
页擦除（512字节/页）	0x40	mPageErase
整体擦除（所有FLASH）	0x41	mMassErase

所有其他命令代码为非法和产生一个访问错误。

在一个整体擦除操作之后没有必要执行一个空检查命令。只有当作为安全解锁机制的一部分时空白检测命令才是需要的。

第五章 复位、中断和系统控制

5.1 引言

本章讨论了 MC9S08JS16 系列中基本的复位和中断机制以及不同的复位和中断源。一些源自外围模块的中断在本手册的其他章节中给出更加详细的讨论。为了便于查阅，本节在一个地方集中给出关于所有复位和中断源的基本信息。一些复位和中断源，包括计算机运行正确（COP）看门狗，有它们本身的章节，不是片内外围设备系统的一部分，但是是系统控制逻辑的一部分。

5.2 特点

复位和中断特点包括：

- 为了灵活的系统控制和可靠运行的多个复位源：
- 复位状态寄存器（SRS）指出最近的复位源
- 每个模块独立的中断矢量（减少了轮询开销） （参看表 5-1）

5.3 MCU复位

对 MCU 进行复位提供了一种方法来从一组已知的初始条件开始处理。在复位期间中，大部分控制和状态寄存器强制置为初始值并从复位矢量处（0xFFFE:0xFFFF）加载程序寄存器。片内外围模块被屏蔽， I/O 引脚初始化配置成通用，高阻的输入，其下拉功能被屏蔽。为了屏蔽可屏蔽中断，条件代码寄存器（CCR） 中的 I 位置 1，以致用户程序有机会初始化堆栈指针（SP）和系统控制设置。复位中 SP 强制置为 0x00FF。

MC9S08JS16 系列有如下复位源：

- 上电复位（ POR ）
- 低电压检测（ LVD ）
- 计算机运行正确（ COP ）定时器
- 非法的操作码检测 （ILOP）
- 后台调试强制复位
- 复位引脚（ $\overline{\text{RESET}}$ ）
- 时钟发生器的锁丢失和时钟复位丢失 （LOC）

每一个这样的源，除了背景调试强制复位，在系统复位寄存器（SRS）中都有一个相关的位。

5.4 计算机正常运行（COP）看门狗

当应用软件与期望的运行不相符时，COP 看门狗试图强制系统复位。为防止系统从 COP 定时器复位，应用软件必须周期的重置 COP 计数器。如果应用程序丢失且在 COP 计数器到达中止时间之前重置 COP 计数器失败，将产生一个系统复位强制系统回到一个已知的起始点。

任何复位之后，COP 计数器都会被激活（更多信息参考 5.7.4 节，“系统选项寄存器 1（SOPT1）。如果 COP 看门狗在一个程序中没有使用，那可以通过将 SOPT1 中的 COPT1 清零来禁止它。

在可选的超时范围内，可以通过向 SRS 的寄存器写入 0x55 和 0xAA （以此顺序）来重置 COP 计数器。这种写入并不影响到只读的 SRS 中的数据。写入一旦完成，COP 超时周期就被重新启动。如果程序没能在超时范围内做到这个，那 MCU 就会被复位。同样的，如果任何其他的值被写入 SRS，MCU 同样会立刻复位。

SOPT2 中的 COPCLKS 位（关于“系统选项寄存器 2（SOPT2），” 见 5.7.5 节参考更多信息）选择了 COP 计时器的时钟源。时钟源可以是总线时钟或者内部 1 kHz LPO 时钟源。每个时钟源都有三种相关的超时范围被 SOPT1 的 COPT 位控制。表 5-6 总结了 COPCLKS 和 COPT 位的控制功能。COP 看门狗的缺省选项是 1 kHz LPO 时钟源和最长的超时范围（2^{10} 周期）。

当总线时钟源被选择，可通过将 SOPT2 中的 COPW 位置 1 来获得窗口 COP。在这种模式下，对 SRS 寄存器写入来将 COP 计时器清零的行为只能发生在选定的超时周期中的最后 1/4 周期内。不成熟的写入将立刻复位 MCU。当 1 kHz LPO 时钟源被选择时，无法获得窗口 COP。

COP 计数器的初始化是在第一次对 SOPT1 和 SOPT2 寄存器写入以及任何系统复位发生之后发生的。之后对 SOPT1 和 SOPT2 的写入对 COP 操作没有任何的影响。即使应用程序将使用 COPT，COPCLKS，COPW 位的复位缺省选项，用户也必须在复位初始化过程中对一次性写入寄存器 SOPT1 和 SOPT2 进入写入来锁定这些设定。这会在应用程序丢失的情况下防止偶然的改变。

服务于（清除）COP 计数器的写 SRS 操作不能被放置在中断服务例程（ISR），因为即使主要的应用程序失败，ISR 也可能继续被周期地执行。

如果选择了总线时钟源，那么当 MCU 处于背景调试模式或系统处于停止模式时 COP 计数器不会计数。等到 MCU 从背景调试模式或停止模式中恢复，COP 计数器才恢复计数。

如果 1 kHz LPO 时钟源被选择，那么当 MCU 进入背景调试模式或停止模式时 COP 计数器将会清零且直到系统离开这两种模式计数器才会重新从 0 开始计数。

5.5 中断

中断提供了一种方法来保存当前 CPU 状态和寄存器，执行一个中断服务子程序（ISR），然后恢复 CPU 状态，以便返回执行中断之前的处理。与只是程序指令的软件中断（SWI）不同，中断由硬件事件触发，如 IRQ 引脚沿或者一个定时器溢出事件。调试模块在某种环境下也能产生 SWI。

如果在使能的中断源中发生一个事件，与之关联的只读的状态标志会被置位。CPU 当且仅当本地中断使能被置 1 来允许中断时才会响应。 CCR 中的 I 位为 0 来允许中断。CCR 中的全局中断屏蔽（I 位）复位后初始化设置为 1，屏蔽（预防）了所有可屏蔽中断源。用户程序在清除 I 位以允许 CPU 响应中断之前初始化堆栈指针和完成其他系统设置。

当 CPU 收到一个许可的中断请求，它在相应中断之前要完成当前指令。中断顺序遵守与 SWI 指令相同的循环顺序，包括：

- 在堆栈中保存 CPU 寄存器
- CCR 中的 I 位置 1,屏蔽更多中断
- 在目前等待的中断中取出最高优先级中断的中断矢量
- 用从中断矢量位置中取出地址处开始的程序信息的前 3 个字节填满指令队列

当 CPU 响应中断，I 位自动置 1，以避免其他中断打断 ISR（这被叫做中断嵌套）。通常，当 CCR 从 ISR 入口的堆栈值恢复时，I 位恢复成 0。在很少的情况下，I 位能在 ISR 中被清零（在清除产生中断地状态标志之后），以便其他中断服务能够在第一个服务子程序没有完成时进入。这种方法只被推荐给具有丰富编程经验的人，因为这容易导致难于调试的隐蔽的错误。

中断服务子程序以一条中断返回（ RTI ）指令结束，其通过读先前存储在堆栈中的信息恢复 CCR，A，X 和 PC 寄存器为中断前的值。

注：

为了与 M68HC08 器件兼容，H 寄存器不被自动保存和恢复。好的编程经验是在中断服务子程序（ISR）开始时把 H 压到堆栈中，在用来从 ISR 中返回的 RTI 之前立即恢复 H。

当 I 位被清零,两个或更多中断等待时，最高优先级的源被首先服务（参看表 5-1）。

5.5.1 中断堆栈的帧

图 5-1 说明了一个堆栈帧的内容和组织。在中断之前，堆栈指针（SP）指向了堆栈中下一个有效的字节位置。CPU 寄存器的当前值存储在堆栈，其开始于程序计数器（PCL）的低位字节，结束于 CCR。压栈之后，SP 指向堆栈中的下一个有效位置，这是一个比保存 CCR 的地址小 1 的地址。压栈的 PC 值是主程序中在中断不发生时将执行的下一条指令所在地址。

当执行一个 RTI 指令，这些值以相反的顺序恢复。作为 RTI 序列的一部分，CPU 用从堆栈恢复的 PC 地址得到程序信息的三个字节填充指令流水。

引起中断的状态标志在从 ISR 返回前一定要被拒绝（清除）。典型的是，在 ISR 开始的时候标志被清除，以便如果同样的源产生另一个中断，中断会被登记,直到当前 ISR 完成之后才会开始服务。

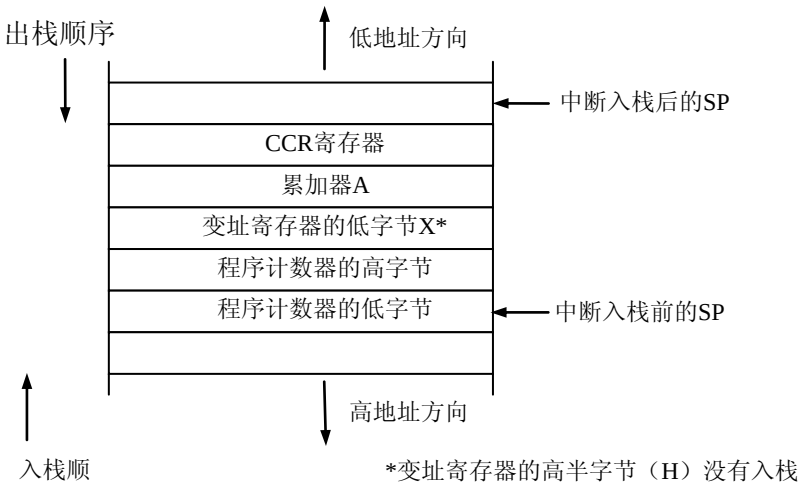


图5-1 中断堆栈框架

5.5.2 外部中断请求（IRQ）引脚

外部中断由 IRQ 状态和控制寄存器 IRQSC 管理。当使能 IRQ 功能，同步逻辑监控引脚是边沿触发还是边沿/电平触发。当 MCU 处于停止状态和系统时钟关闭时，使用一个独立的异步通路以便 IRQ （如果使能）能唤醒 MCU。

5.5.2.1 引脚配置选项

为了 IRQ 引脚作为中断请求（IRQ）输入，IRQSC 中的 IRQ 引脚使能（IRQPE）控

制位必须为 1。作为一个 IRQ 输入，用户可选择引脚检测边沿触发或边沿/电平触发（IRQMOD），以及一个事件是否导致一个中断或者只是将 IRQF 位置 1，这能被软件轮询。

IRQ 引脚，当被允许时，缺省使用一个内部上拉或下拉设备（IRQPDD=0），上拉还是下拉取决于其极性选择。如果用户希望用一个外部上拉或下拉设备，可以向 IRQPDD 写入 1 来关闭内部设备。

BIH 和 BIL 指令能够在 IRQ 引脚脚被配置成 IRQ 输入时用来检测引脚上的电平。

注：

该引脚不含一个接 V_{DD} 的固定二极管，且电压不能驱动到高于 V_{DD}。在内部拉升的 IRQ 引脚上测得的电压可能和 V_{DD} 同样低——0.7V。与此引脚相连接的内部门电路被拉至 V_{DD}。如果要求 IRQ 引脚被拉至 VDD 电平，就必须使用一个外部上拉设备。

注：

当允许 IRQ 引脚工作时，IRQF 位必须被置 1，且必须事先被清零来允许中断。当把该引脚设置为在一 5V 系统内下降沿和电平敏感，就必须在清零标志位和允许中断间等待至少 6 个周期。

5.5.2.2 边沿和电平敏感

IRQMOD 控制位重配置监测逻辑以便监测边沿事件和引脚电平。在这种边沿监测模式中，当一个边沿被监测到时（当 IRQ 引脚从未声明电平变为声明电平） IRQF 状态标志被置 1，但是标志位当 IRQ 保持声明电平时一直被置 1（不能被清除）。

5.5.3 中断向量，源和本地屏蔽

表 5-1 总结了所有中断源。较高优先级的源位于表格下方。中断服务程序地址的高位字节位于向量地址栏的第一个地址，中断服务程序地址的低阶字节位于下一个较高位地址中。

当出现中断时，相关标记位被 1 置。如果相关的本地中断允许位是 1，中断请求会发送到 CPU。在 CPU 中，如果全局中断屏蔽 （CCR 中的 I 位）是 0， CPU 将完成当前指令；将 PCL、PCH、 X、 A 和 CCR CPU 寄存器入栈；设置 I 位；然后为挂起的最高优先级中断获取中断向量。然后处理中断服务程序。

表5-1 向量总结（从最低优先级到最高优先级）

向量号	地址（高/低）	向量名	模块	源	使能	描述
31 到 30	0xFFC0:FFC1 到 0xFFC2:FFC3	未使用向量空间（用户程序有效）[1]				
29	0xFFC4:FFC5	Vrtc	系统控制	RTIF	RRIE	RTC实时中断
28 到 27	0xFFC6:FFC7 到 0xFFC8:FFC9	未使用向量空间（用户程序有效）				
26	0 xFFCA: FFCB	Vmtim	MTIM	TOF	TOIE	MTIM溢出
25	0xFFCC/FFCD	Vkeyboard	KBI	KBF	KBIE	键盘中断
24 到 22	0xFFCE/FFCF 到 0xFFD2:FFD3	未使用向量空间（用户程序有效）				
21	0xFFD4:FFD5	Vsci1tx	SCI1	TDRE TC	TIE TCIE	SCI1 传送
20	0xFFD6:FFD7	Vsci1rx	SCI1	IDLE RDRF	ILIE RIE	SCI1接收
19	0xFFD8:FFD9	Vsci2err	SCI2	OR NF	ORIE NFIE	SCI2错误

				FE PF	FEIE PFIE	
18 到 12	0xFFDA:FFDB 到 0xFFE6:FFE7	未使用向量空间 （用户程序有效）				
11	0xFFE8:FFE9	Vtpmovf	TPM	TOF	TOIE	TPM溢出
10	0xFFEA:FFEB	Vtpmch1	TPM	CH1F	CH1IE	TPM通道 1
9	0xFFEC:FFED	Vtpmch0	TPM	CH0F	CH0 IE	TPM通道 0
8	0xFFEE:FFEF	未使用向量空间 （用户程序有效）				
7	0xFFF0:FFF1	Vusb	USB	STALLF RESUMEF SLEEPF TOKDNEF SOFTOKF ERRORF USBRSTF	STALL RESUME SLEEP TOKDNE SOFTOK ERROR USBRST	USB 状 态
6	0xFFF2:FFF3	未使用向量空间 （用户程序有效）				
5	0xFFF4:FFF5	Vspi	SPI	SPRF MODF SPTEF SPMF	SPIE SPIE SPTIE SPMIE	SPI
4	0xFFF6:FFF7	Vlol	MCG	LOLS	LOLIE	MCG锁损失
3	0xFFF8:FFF9	Vlvd	系统控制	LVWF	LVWIE	低电压检 测
2	0xFFFA:FFFB	Virq	IRQ	IRQF	IRQIE	IRQ引脚
1	0xFFFC:FFFD	Vswi	内核	SWI指令	—	软件中断
0	0xFFFE:FFFF	Vreset	系统控制	COP LVD 复位引脚 非法编码 非法地址 LOC POR BDFR	COPE LVDRE — ILOP ILAD CME POR	看门狗计时器 低电压检测 外部引脚 非法编码 非法地址 时钟损失 上电复位 BDM强制复位

1 未使用向量空间可以与普通 FLASH 存储空间一样被使用。然而，S08 系列其他 MCU 芯片可能使用这个地址作为中断向量。因此，如果要把程序移植到其他 MCU，必须小心使用这个址。

5.6 低电压检测（LVD）系统

MC9S08JS16 系列包括一个防止低电压的系统，以便在电源电压不稳时保护存储器内容、控制 MCU 系统状态。该系统由加电复位 （POR）电路和 LVD 电路组成，其中 LVD 电路带跳变电压用于警告和检测。当 SPMSC1 中的 LVDE 设置为 1 时，LVD 电路使能。当进入停止模式时，LVD 禁止，除非 SPMSC1 中设置了 LVDSE。如果同时设置了 LVDSE 和 LVDE，那么 MCU 不能进入 stop2 （进入 stop3）， LVD 激活的 stop3 模式更耗电。

5.6.1 上电复位操作

当首次接通 MCU 的电源时，或当电源电压低于加电复位准备电压 V_{POR} 时， POR 电路会引发复位。随着电源电压升高， LVD 电路让 MCU 保持复位状态，直到电源高于低

压检测阈值 V_{LVDL}。 POR 后， SRS 中 POR 位和 LVD 位同时被设置。

5.6.2 LVD复位操作

通过把 LVDRE 设置为 1，可以在检测到低压情况时配置 LVD 以发起复位。低压检测阈值由 LVDV 位决定。在 LVD 复位后， LVD 系统会让 MCU 保持复位状态，直到电源电压高于低压检测阈值。LVD 复位或 POR 后都会在 SRS 寄存器将 LVD 位中置 1。

5.6.3 低电压警告（LVW）

LVD 系统有一个低压警告标记，告知用户电源电压正接近低电压条件。当一个低电压警告条件被检测到且设置为中断操作允许时（LVWIE=1），SPMSC1 中的 LVWF 会被置 1，LVW 中断请求将会发出。

5.7 复位，中断，系统控制寄存器以及控制位

直接页寄存器空间内的一个 8 位的寄存器和高页寄存器空间内的 8 个 8 位寄存器和复位及中断系统相关联。

寄存器绝对地址的分配参见本手册的第四章，“存储器”有关直接页寄存器的总结。本节通过名称来引用各寄存器和控制位。飞思卡尔提供的通用或头文件可用来将他们的名字与绝对地址对应起来。

一些 SOPT1 和 SPMSC2 寄存器与操作模式相关。尽管这里提供了对这些位的简短的描述，一些相关功能的详细讨论见第三章，“操作模式”。

5.7.1中断引脚请求状态和控制寄存器（IRQSC）

这个直接页寄存器包括状态和控制位，用来设置IRQ功能，返回状态，应答IRQ事件。

	第7位	6	5	4	3	2	1	第0位
读	0	IRQPDD	IRQEDG	IRQPE	IRQF	0	IRQIE	IQRMOD
写						IRQACK		
复位	0	0	0	0	0	0	0	0

图5-2 中断请求状态和控制寄存器（IRQSC）

表5-2 IRQSC寄存器字段描述

字段	描述
6 IRQPDD	**中断要求（IRQ）上拉设备禁止**——这个读/写控制位被用来在IRQ引脚允许（IRQPE=1）时使用外部设备禁止内部上拉设备 0 IRQ上拉设备允许 当IRQPE=1 1 IRQ 上拉设备被禁止 当IRQPE=1
5 IRQEDG	**中断请求 （IRQ） 边沿选择** — 这个读/写控制位选择导致 IRQF 置1的 IRQ 引脚上的边沿或电平的极性。IRQMOD 控制位决定 IRQ 引脚对边沿和电平都敏感还是只对边沿敏感。当 IRQ 引脚作为 IRQ 输入被使能且配置来检测上升边沿时，可选的上拉电阻就被重置为可选下拉电阻。 0 IRQ 是下降边沿，或者对下降边沿 / 低电平敏感； 1 IRQ 是上升边沿，或者对上升边沿 / 高电平敏感。
4 IRQPE	**IRQ 引脚使能** — 这个读写控制位使能 IRQ 引脚。当设置了该位时， IRQ 引脚可以用作中断请求。 0 IRQ 引脚禁止； 1 IRQ 引脚使能。
3	**IRQ 标志** — 该只读状态位显示中断请求事件何时发生。

IRQF	0 无 IRQ 请求； 1 检测到 IRQ 事件。
2 IRQACK	IRQ 确认 — 这个只写位用来确认中断请求事件 （写 1 来清除 IRQF）。写入 0 则没有任何意义或影响。读总是返回 0。如果选择了边沿和电平检测 （IRQMOD = 1），则在IRQ保持为声明电平时不能清除 IRQF。
1 IRQIE	**IRQ 中断使能** — 这个读 / 写控制位决定 IRQ 事件是否生成中断请求。 0 当 IRQF = 1 时不产生中断请求 （使用轮询）； 1 当 IRQF = 1 时产生中断请求。
0 IRQMOD	**IRQ 检测模式** — 这个读 / 写控制位选择只边沿检测还是边沿和电平检测。请参见5.5.2.2“边沿和电平敏感度”。 0 只下降边沿或上升边沿的 IRQ 事件； 1 下降边沿和低电平 IRQ 事件或上升边沿和高电平 IRQ 事件。

5.7.2 系统复位状态寄存器（SRS）

这个寄存器包括七个只读状态标志位来指明大多数最近复位的来源。当一个调试主机通过向 SBDFR 中的 BDFR 写入 1 来强制复位时，SRS 中的任何位都不置 1。除了 0x55 和 0xAA 以外向这个寄存器地址写入任何值都会引起 COP 复位，如果 COP 允许的话。向这个地址写入一个 0x55-0xAA 序列会将 COP 看门狗清零而不会影响寄存器的内容。这些位的复位状态依赖于是什么引起 MCU 复位。

	第7位	6	5	4	3	2	1	第0位
读	POR	PIN	COP	ILOP	ILAD	LOC	LVD	—
写	向该寄存器地址写入任意值可清零COP看门狗计数器							
上电复位	1	0	0	0	0	0	1	0
低电压复位	U	0	0	0	0	0	1	0
其他复位	0	(1)	(1)	(1)	0	(1)	0	0

U=不被复位影响

（1）在复位时任何有效的复位源都将导致相应的位被置位。同时相应的不有效的位被清零。

图5-3 系统复位状态（SRS）

表5-3 SRS寄存器字段描述

字段	描述
7 POR	**加电复位** — 复位由上电检测逻辑引起。由于此时内部电源电压此时发生跳跃，所以低压复位（LVD）状态位被置1，以显示当内部供应电压下降到LVD限度时复位发生。 0 非 POR 造成的复位。 1 POR POR 造成的复位。
6 PIN	外部复位引脚 — 复位由外部复位引脚上的活动低电平造成。 0 非外部复位引脚造成的复位。 1 外部复位引脚造成的复位。
5 COP	**计算机正常操作** （COP） 看门狗 — 复位由 COP 看门狗定时器超时导致。该复位源可以由 COPE = 0 拦截。 0 非 COP 超时造成的复位。 1 超时造成的复位。
4 ILOP	**非法操作码** — 复位由试图实施未实现或非法操作码导致。如果停止模式由 SOPT 寄存器里 STOPE = 0 禁止，STOP指令被视为非法。如果激活背景模式由 BDCSC 寄存器里 ENBDM = 0 禁止， BGND 指令指令被视为非法。 0 不是由非法操作码造成的复位。 1 非法操作码造成的复位。
3 ILAD	**非法地址**—复位因试图访问在一个未实现存储地址中的数据或指令引起。 0 复位不是由非法地址引起 1 复位由非法地址引起
2 ICG	**时钟损失复位** —由外部时钟损失造成的复位。 0 不是由外部时钟损失造成的复位。 1 由外部时钟损失造成的复位

1 LVD	**低压检测** — 如果设置了 LVDRE 位且电源降到 LVD 跳变电压以下，就会发生 LVD 复位。该位也可以由 POR 设置。 0 不是由 LVD 门槛或 POR 造成的复位。 1 由 LVD 门槛或 POR 造成的复位。

5.7.3 系统背景调试强制复位寄存器 （SBDFR）

这个寄存器包括一个单独的只写位。一个串行的背景命令例如 WRITE_BYTE 必须被用来写入 SBDFR。从用户程序中试图写入这个寄存器将被忽略。读取的返回值永远为 0x00。

	第7位	6	5	4	3	2	1	第0位
读	0	0	0	0	0	0	0	0
写								BDFR (1)
复位	0	0	0	0	0	0	0	0

=未使用或保留

（1） BDFR只能通过背景调试命令而不能通过用户程序写入

图5-4 系统背景调试强制复位寄存器（SBDFR）

表5-4 SBDFR寄存器字段描述

字段	描述
0 BDFR	**背景调试强制复位** — 可以使用串行背景命令，如 WRITE_BYTE ，使外部调试主机强制进行目标系统复位。在该位中写入 1 就能强制进行 MCU 复位。该位不能从用户程序中写入。

5.7.4 系统选项寄存器1 （SOPT1）

这个寄存器可以在任何时间读取。位 4 和位 3 不被使用且永远为 0。这是一个一次性写入寄存器所以只有当在复位之后的第一次写入才是合法的。之后任何写入 SOPT 的企图（有意或无意）都会被忽略以防止改变这些敏感的设定。SOPT 必须在用户复位初始化程序期间被写入设定需要的控制位，即便这些需要的设定与复位设定是一样的。

读 写	COPT		STOPE	1	0	BLMSS	BKGDPE	RSTPE
复位	1	1	0	1	0	0	1	u (1)
POR	1	1	0	1	0	0或1 (2)	1	0
LVR	1	1	0	1	0	0	1	u

[1] 不影响

[2] 依赖于PTB3/BLMS引脚状态

图5-5 系统选项寄存器（SOPT1）

表5-5 SOPT1寄存器字段描述

字段	描述
7：6 COPT[1:]	**COP看门狗超时** — 这些只写一次位选择了COP的超时周期。SOPT2中的COPT和COPCLKS定义了COP超时范围。见表5-6
5 STOPE	停止模式使能 — 这个单次写入有效的位用来使能停止模式。如果停止模式禁止且用户程序试图实施 STOP 指令，则会强制进行非法操作码复位。 0 停止模式禁止。 1 停止模式使能。
2 BLMSS	**BLMS引脚状态** — 这个只读位在上电复位期间（POR）显示了PTB3/$\overline{\text{BLMS}}$引脚状态。 0 在POR期间$\overline{\text{BLMS}}$引脚是高电平 1 在POR期间$\overline{\text{BLMS}}$引脚是低电平且MS引脚是高电平
1 BKGDPE	**背景调试模式引脚使能** — 这个只写一次位当被置1时可以将PTB2/BKGD/MS引脚设为BKGD/MS功能引脚。当它被清零时，这个引脚的功能将是备选的单输出功能。在任何MCU复位之后，这个引脚的缺省功能是BKGD/MS功能。

	0 PTB2/BKGD/MS引脚功能为PTB2 1 PTB2/BKGD/MSG引脚功能为BKGD/MS
0 RSTPE	**RESET引脚使能** — 这个只写一次位被置1时PTB1/$\overline{\text{RESET}}$引脚为/$\overline{\text{RESET}}$引脚。当被清零时，这个引脚的功能是它备选功能之一。POR复位后这个引脚被作为一个通用输入端口引脚。当被设置为/$\overline{\text{RESET}}$引脚，这个引脚就不被LVR或其他内部复位所影响。当RSTPE被置1，一个内部上拉设备在/$\overline{\text{RESET}}$引脚上被使能。 0 PTB1//$\overline{\text{RESET}}$引脚作为PTB1引脚 1 PTB1//$\overline{\text{RESET}}$引脚作为/$\overline{\text{RESET}}$引脚

表5-6 COP设置选项

控制位		时钟源	COP窗口[1]打开（COPW=1）	COP溢出计数
COPCLKS	COPT[1:0]			
N/A	0:0	N/A	N/A	COP被禁止
0	0:1	1 kHz	N/A	2^5周期（32ms[2]）
0	1:0	1 kHz	N/A	2^8周期（256ms[1]）
0	1:1	1 kHz	N/A	2^{10}周期（1024ms[1]）
1	0:1	总线	6144周期	2^{13}周期
1	1:0	总线	49152周期	2^{16}周期
1	1:1	总线	196608周期	2^{18}周期

1 窗口 COP 操作需要用户在选定超时周期的最后 25%时间内清零 COP 计时器。这列显示了在窗口 COP 模式（COPW=1）COP 计时器可以被重置前需要的最小时钟计数值。
2 在毫秒中显示的值基于 T_{lpo}=1 ms。在 MC9S08JS16 手册附录 A.12.1，“控制计时”中获得 T_{lpo} 的容限。

5.7.5 系统选项寄存器2 （SOPT2）

读	COPCLKS[1]	COPW[1]	0	0	0	SPIFE	0	0
写								
复位	0	0	0	0	0	1	0	0

=未使用或保留

[1] 这一位在复位后只能写一次。多余的写入无效。

图5-6 系统选项寄存器2（SOPT2）

表5-7 SOPT2寄存器域描述

域	描述
7 COPCLKS	**COP 看门狗时钟选择——**这个位仅可写入一次，选择了COP看门狗的时钟源 0 COP时钟源是内部1kHz时钟 1 COP时钟源是总线时钟
6 COPW	**COP 窗口——**该位只可写入一次，选择了COP的操作模式。当被置1，0x55-0xAA向SRS寄存器写入字节必须发生在选定周期的后25%时间内。任何其他时间写入到SRS寄存器的行为都会使MCU复位。 0 普通COP操作 1 窗口 COP操作
2 SPI1FE	**SPI 端口输入滤波器使能** 0 禁止在SPI端口引脚上的输入滤波，允许更高的最大SPI波特率 1 允许在SPI端口引脚上的输入滤波，限制噪音限制最大SPI波特率

5.7.6 FLASH保护失败寄存器（FPROTD）

	第7位	6	5	4	3	2	1	第0位
读	0	0	0	0	0	0	0	0
写								Bit0
复位	0	0	0	0	0	0	0	0

=未使用或保留

图5-7 FLASH保护失败寄存器（FPROTD）

表5-8 FPROTD 寄存器寄存器字段描述

域	描述
0	被保护FLASH区段可以通过先向FPROTD地址中背靠背写入0x55和0xAA然后将FPROT中的FPDIS位置1来禁止。这位只能在0x55和0xAA成功的背靠背写入之后才能置1。任何除此之外的写入操作都会将该位清零。 0 FPROT中位FPDIS不可以被置1 1 FPROT中位FPDIS可以被置1

5.7.7 SIGNATURE寄存器（SIGNATURE）

	第7位	6	5	4	3	2	1	第0位
读 写	SIGNATURE 信号量							
复位	0	0	0	0	0	0	0	0

=未使用或保留

图5-8 SIGNATURE寄存器（SIGNATURE）

图5-9 SIGNATURE寄存器字段描述

域	描述
7:0	SIGNATURE信号量杯用于跳向引导程序模式或不在常规复位之后的信号量。这个字节可以被上电复位（POR）清零，内容在常规复位之后被保留。

5.7.8 系统设备识别寄存器（SDIDH,SDIDL）

这个只读寄存器被包含，以便主机开发系统能识别 HCS08 的衍生及修改号。这使得开发软件可以在相应的 CPU 中识别出特定存储块，寄存器及控制位在什么地方。

	第7位	6	5	4	3	2	1	第0位
读					ID11	ID10	ID9	ID8
写								
复位	-	-	-	-	0	0	0	0

=未使用或保留

图5-9 系统标识寄存器-高（SDIDH）

表5-10 SDIDH寄存器字段描述

字段	描述
7:4预留的	**位7:4被预留**。读这些位将会导致一个不确定值。写入操作没有意义。
3:0ID[11:8]	**部件识别编号** — 每一个HCS08的派生型有一个独特的辨识号。MC9S08JS16固件编号为值0x024。也参考表5-11中的ID号。

	第7位	6	5	4	3	2	1	第0位
读	ID7	ID6	ID5	ID4	ID3	ID2	ID1	ID0
写								
复位	0	0	1	0	0	1	0	0

=未使用或保留

图5-10系统标识寄存器-低（SDIDL）

表5-11 SDIDL寄存器字段描述

字段	描述
7:0 ID[7:0]	部件识别编号 — 每一个HCS08的派生物有一个独特的辨识号。MC9S08JS16固定的编号为值0x024。也参考表5-10中的ID号。

5.7.9 系统能源管理状态和控制1寄存器（SPMSC1）

这个高页寄存器包括状态及控制位来支持低电压检测功能。这个寄存器必须在用户复位初始程序期间被写入来设置需要的控制位，即便需要的设定和复位设定是一样的。

	第7位	6	5	4	3	2	1（1）	第0位
读	LVWF[1]	0	LVWIE	LVDRE[2]	LVDSE	LVDE	0	0
写		LVWACK						
复位	0	0	0	1	1	1	0	0

=未使用或保留

1 LVWF位会在V_{Supply}转换低于跳变点或复位后V_{Supply}比V_{LVW}低时被置位。
2 这个位复位后只能写一次。其他的写入被忽略。

图5-11 系统电源管理状态控制寄存器1（SPMSC1）

表5-12 SPMSC1寄存器域描述

域	描述
7 LVWF	**低电压警告标志位——**LVWF位指示了低电压警告状态 0 低电压警告不存在 1 低电压警告存在或曾存在
6 LVWACK	**低电压警告应答——**如果LVWF=1,一个低电压条件已经发生。为了应答低电压警告，将1写入LVWACK，当低电压警告消失它会自动清零LVWF
5 LVWIE	低电压警告中断允许——这位允许了为LVWF硬件中断请求 0 硬件中断禁止（轮询） 1 提出硬件中断请求当LVWF=1
4 LVDRE	**低电压探测复位允许——**这个只写一次位允许LVD事件可以产生一个硬件中断（LVDE=1） 0 LVD事件不产生硬件复位 1 当一个被允许的低电压检测事件发生时强制MCU复位
3 LVDSE	**停止模式低电压检测允许——**LVDE=1时，可读/写位决定了当MCU位于停止模式时低电压检测功能是否继续 0 LVD逻辑禁止 1 LVD逻辑允许
2 LVDE	**低电压检测允许——**这个只写一次位允许了低电压检测逻辑并使寄存器其他位符合要求 0 LVD逻辑禁止 1 LVD逻辑允许

5.7.8 系统能源管理状态和控制2寄存器（SPMSC2）

这个寄存器用来报告低电压警告功能的状态，以及设置 MCU 在停止模式的行为。

	第7位	6	5	4	3	2	1	第0位
读	0	0	LVDV	LVWV	PPDF	0	0	PPDC[1]
写						PPDACK		

上电复位	0	0	0	0	0	0	0	0
LVD复位	0	0	U	U	0	0	0	0
其他复位	0	0	U	U	0	0	0	0

=未使用或保留　　u=未使用

1 这个位复位后只能写一次。其他的写入被忽略。

图5-12 系统电源管理状态和控制寄存器2（SPMSC2）

表5-13 SPMSC2寄存器字段描述

域	描述
5 LVDV	**低电压探测电压选择——**这个位选择低电压探测（LVD）跳变点设置。它同样选择警报电压范围，见表5-14
4 LVWV	**低电压警告电压选择——**这位选择低电压警告（LVW）跳变点电压。见表5-14
3 PPDF	**部分掉电标志位——**这个只读状态位指示了MCU已从STOP2模式返回 0 MCU没有从STOP2模式返回 1 MCU 从STOP2模式返回
2 PPDACK	**部分掉电应答——**将1写入PPDACK清零PPDF位。
0 PPDC	**部分掉电控制——**这个只写一次位控制对STOP2和STOP3模式的选择 0 STOP3使能 1 STOP2,部分掉电模式下使能。

1

表5-12 LVD和LVW跳变点典型值

LVDV:LVWV	LVW跳变点	LVD跳变点
0:0	V_{LVW0}=2.74V	V_{LVD0}=2.56V
0:1	V_{LVW1}=2.92V	
1:0	V_{LVW2}=4.3V	V_{LVD1}=4.0V
1:1	V_{LVW3}=4.6V	

1 见 MC9S08JS16 系列手册获得最小和最大值。

第六章 并行输入/输出控制

6.1 介绍

本小节解释了与并行输入 / 输出（I/O）相关的软件控制。 MC9S08JS16 系列 有 2 个 I/O 端口，总共包含 14 个通用 I/O 引脚。如需了解这些引脚的逻辑和硬件分配的更多信息，请参见第 2 章，“引脚和连接”。

很多 I/O 引脚都与片内外设功能复用，如表 2-1 所示。外设模块优先于 I/O，所以当一个外设模块被允许时，I/O 功能被禁止。

复位过后，共用的外围设备被禁止所以引脚执行并行输入/输出功能。所有的并行输入/输出都被设置成输入（PTxDDn=0）。每个引脚的引脚控制功能被如下设置：速率转换控制使能（PTxSEn=1），低驱动电流选择（PTxDSn=0），内部上拉禁止（PTxPEn=0）。

注：

为避免额外电流从输入引脚上消耗，应用程序中的用户复位初始化例程必须要么使能片内上拉设备要么将未连接引脚的方向改成输出防止引脚移动。

注：

当 PTB0 引脚被设置为输出引脚时，它是开漏引脚。

数据方向控制位决定了引脚的输出驱动是否被允许。每个端口引脚都有一个数据方向寄存器位。当 PTxDDn=0 时，相应的引脚是个输入引脚，对 PTxD 的读取返回引脚的值。当 PTxDDn=1 时，相应的引脚是个输出引脚，对 PTxD 的读取返回该端口数据寄存器的最后一次被写入的值。当一个外围模块或系统功能控制该端口引脚，数据方向寄存器位仍然控制着读取这个端口数据寄存器的返回值，即便外围系统已经控制了这个引脚的实际方向。

当一个共用的模拟设备被允许，所有引脚的数字功能被禁止。读取端口数据寄存器所有引脚上共用模拟设备被允许的位返回 0。总之，任何时候一个引脚被一个数字功能一个模拟功能共享，模拟设备都有更高的优先权，因此，当数字和模拟设备都被允许时，模拟设备控制该引脚。

在改变一个端口引脚为输出前写入数据到端口数据寄存器。它保证了引脚不会突然被一个恰好存放在寄存器中的旧数据所驱动。

6.3 端口控制

端口控制寄存器位于存储器中的高页寄存器块中。这些寄存器用来控制上拉设备，速率转换以及 I/O 引脚的驱动强度。引脚控制寄存器独立于并行通用 I/O 寄存器。

6.3.1 内部上拉允许

每个端口引脚都有一个内部上拉设备，它可以通过设置一个上拉允许寄存器（PTxPEn）的相关位来允许。如果引脚被并行 I/O 控制逻辑或任何共用的外围设备设置成一个输出引脚，不管相应的上拉允许寄存器位被如何设置，上拉设备都会被禁止。如果引脚被模拟设备控制上拉设备也同样会被禁止。

6.3.2 输出速率转换控制允许

每个端口引脚的速率转换控制都可以通过设置一个速率转换控制寄存器（PTxSEn）来

被允许。当速率转换控制被允许，转换控制可以限制输出速率的转换来减少 EMC 辐射。在设置为输入的引脚上速率转换控制没有任何意义。

6.3.3 输出驱动强度选择

一个输出引脚可以可以通过设置一个驱动力选择寄存器（PTxDSn）中的相关位来选择较高输出驱动力。当一个高驱动被选择时，引脚可以流汇更大的电流。即便每个 I/O 引脚可以选择为高驱动，但用户必须保证片内流汇总电流不超过限制。驱动力选择会影响 DC 或 I/O 引脚的行为。然后，AC 的行为不受影响。高驱动允许一个引脚在同样的转换速度下驱动高负荷，低驱动则让引脚驱动低负荷。因此，EMC 排放也会受到高驱动力引脚的影响。

6.4 在停止模式下引脚的行为

执行 STOP 指令后，I/O 的功能因停止模式的不同而不同。不同停止模式下 I/O 行为的不同解释如下：

- STOP2 模式是一个部分掉电模式，借此 I/O 锁定被保持在 STOP 指令执行之前它们的状态。执行 STOP 模式使 MCU 处于 STOP2 模式前，CPU 寄存器状态和 I/O 寄存器状态必须被保存在 RAM 中。当从 STOP2 中恢复时，在用任何 I/O 前，用户必须检查 SPMSC2 中的 PPDF 位的状态。如果 PPDF 位为 0，I/O 必须被初始化就如上电复位发生一样。如果 PPDF 位为 1，I/O 寄存器状态必须从在 STOP 指令被执行前它们被存放的 RAM 中装载回来。外围设备也许需要初始化或重新装载回它们的 STOP 前状态。用户必须在 SPMSC2 寄存器中的 PPDACK 位写入一个 1。用户程序再一次可以使用 I/O。
- STOP3 模式，所有的 I/O 都会保持，因为内部逻辑电路保持有电。当恢复时，用户可以使用 I/O 功能。

6.5 并行I/O和引脚控制寄存器

这节提供了关于了与并行 I/O 端口和引脚控制功能相关的寄存器的信息。这些并行 I/O 寄存器位于存储器映像的页 0，而引脚控制寄存器位于存储器中的高页寄存器段。

参考第四章，“存储器”中的表格，了解所有并行 I/O 和引脚控制寄存器的绝对地址分配信息。这节仅仅用这些寄存器及控制位的名字来提到它们。一份飞思卡尔提供的通用文件或头文件一般用来把这些名字和它们相对应的绝对地址对应起来。

6.5.1 端口A I/O寄存器（PTAD 和PTADD）

并行 I/O 端口 A 功能被下表中的寄存器控制。

	第7位	6	5	4	3	2	1	第0位
读 写	PTAD7	PTAD6	PTAD5	PTAD4	PTAD3	PTAD2	PTAD1	PTAD0
复位	0	0	0	0	0	0	0	0

图6-2 端口A数据寄存器（PTAD）

6－1 PTAD寄存器字段描述

字段	描述

7:0 PTAD[7:0]	**端口A数据寄存器位** — 对于配置为输入的A端口引脚，读数返回引脚上的逻辑电平。对于配置为输出的A端口引脚，读数返回写入寄存器的最后一个值。 写入值被锁定在本寄存器的所有位中。对于配置为输出的A端口引脚，逻辑电平驱动相应的MCU 引脚。 复位强制 PTAD都为0，但是这些0未被驱出相应的引脚，因为复位还会将所有端口引脚配置为上拉被禁止的高抗阻输入。

	第7位	6	5	4	3	2	1	第0位
读 写	PTADD7	PTADD6	PTADD5	PTADD4	PTADD3	PTADD2	PTADD1	PTADD0
复位	0	0	0	0	0	0	0	0

图6-3 端口A数据方向寄存器（PTADD）

表6－2 PTADD寄存器字段描述

字段	描述
7:0 PTADD[7:0]	**端口A数据方向位** — 这些读/写位控制着A端口引脚的方向以及为 PTAD 读数读取的内容。 0 输入（输出驱动被禁止），读数返回引脚值。 1 A端口位输出驱动使能， PTAD 读数返回 PTADn 内容。

6.5.2 端口A 引脚控制寄存器（PTAPE,PTASE,PTADS）

除了 I/O 控制，端口 A 引脚还被下列的寄存器控制。

	第7位	6	5	4	3	2	1	第0位
读 写	PTAPE7	PTAPE6	PTAPE5	PTAPE4	PTAPE3	PTAPE2	PTAPE1	PTAPE0
复位	0	0	0	0	0	0	0	0

图6-4 端口A上拉允许寄存器（PTAPE）

表6－3 PTAPE寄存器字段描述

字段	描述
7:0 PTAPE[7:0]	**端口A内部上拉使能位** — 对于 PTA引脚，这些控制位决定着为相关PTA引脚是否允许内部上拉设备。对于配置为输出的A端口引脚脚，这些位不会产生影响，同时内部上拉设备被禁止。 0 A端口n位内部上拉设备被禁止。 1 A端口n位内部上拉设备使能。

	第7位	6	5	4	3	2	1	第0位
读 写	PTASE7	PTASE6	PTASE5	PTASE4	PTASE3	PTASE2	PTASE1	PTASE0
复位	0	0	0	0	0	0	0	0

图6-5 端口A输出速率转换控制寄存器（PTASE）

表6－4 PTASE寄存器字段描述

字段	描述
7:0 PTASE[7:0]	**端口A输出转换率使能位** — 这些控制位决定着是否为相关PTA引脚是否使能输出转换率控制。对于配置为输入的A端口引脚，这些位不会产生任何影响。 0 A端口n位输出转换率控制被禁止。 1 A端口n位输出转换率控制使能。

	第7位	6	5	4	3	2	1	第0位
读 写	PTADS7	PTADS6	PTADS5	PTADS4	PTADS3	PTADS2	PTADS1	PTADS0
复位	0	0	0	0	0	0	0	0

图6-6 端口A输驱动选择寄存器（PTADS）

表6－5 PTADS寄存器字段描述

字段	描述
7:0 PTADS[7:0]	**端口A输出驱动强度选择位** — 这些控制位为相关PTA引脚选择低输出驱动和高输出驱动。对于配置为输入的A 端口引脚，这些位不会产生任何影响。 0 A端口n位选择低输出驱动强度。 1 A端口n位选择高输出驱动强度。

6.5.3 端口B I/O寄存器（PTBD 和PTBDD）

并行 I/O 端口 B 功能被下表中的寄存器控制。

	第7位	6	5	4	3	2	1	第0位
读 写	0	0	PTBD5	PTBD4	PTBD3	PTBD2	PTBD1	PTBD0
复位	0	0	0	0	0	0	0	0

=未使用或预留

图6-7 端口B数据寄存器（PTBD）

表6－6 PTAD寄存器字段描述

字段	描述
5:0 PTBD[5:0]	**端口B数据寄存器位** — 对于配置为输入的B端口引脚，读数返回引脚上的逻辑电平。对于配置为输出的B端口引脚，读数返回写入寄存器的最后一个值。 写入值被锁定在本寄存器的所有位中。对于配置为输出的B端口引脚，逻辑电平驱动相应的MCU 引脚。 复位强制 PTBD 都为0，但是这些0未被驱出相应的引脚，因为复位还会将所有端口引脚配置为上拉被禁止的高抗阻输入。

	第7位	6	5	4	3	2	1	第0位
读 写	0	0	PTBD5	PTBD4	R	R	PTBD1	PTBD0
复位	0	0	0	0	0	0	0	0

=未使用或预留

图6-8 端口B数据方向寄存器（PTBDD）

表6－7 PTBDD寄存器字段描述

字段	描述
5：4， 1：0 PTBDD[5:4,1:0]	**端口B数据方向位** — 这些读/写位控制端口B引脚的方向和读取PTBD的值 0 输入（输出驱动被禁止），读取返回值是引脚值 1 B端口n位输出驱动允许，PTBD读取返回PTBDn位的内容

6.5.4 端口B 引脚控制寄存器（PTBPE,PTBSE,PTBDS）

除了 I/O 控制，端口 B 引脚还被下列的寄存器控制。

	第7位	6	5	4	3	2	1	第0位
读	0	0	PTBPE5	PTBPE4	R	R	PTBPE1	PTBPE0
写								
复位	0	0	0	0	0	0	0	0

=未使用或预留

图6-9 端口B上拉允许寄存器（PTBPE）

表6－8 PTBPE寄存器字段描述

字段	描述
5：4， 1：0 PTBPE[5:4,1:0]	**端口B内部上拉使能位** — 对于PTB引脚，这些控制位决定着为相关PTB引脚是否允许内部上拉设备。对于配置为输出的B端口引脚脚，这些位不会产生影响，同时内部上拉设备被禁止。 0 B端口n位内部上拉设备被禁止。 1 B端口n位内部上拉设备使能。

	第7位	6	5	4	3	2	1	第0位
读	0	0	PTBSE5	PTBSE4	R	R	PTBSE1	PTBSE0
写								
复位	0	0	0	0	0	0	0	0

=未使用或预留

图6-10 端口B速率转换控制寄存器（PTBSE）

表6－9 PTBSE寄存器字段描述

字段	描述
5: 0 PTBSE[5:1]	**端口B输出转换率使能位** — 这些控制位决定着是否为相关PTB引脚是否使能输出转换率控制。对于配置为输入的B端口引脚，这些位不会产生任何影响。 0 B端口n位输出转换率控制被禁止。 1 B端口n位输出转换率控制使能。

	第7位	6	5	4	3	2	1	第0位
读	0	0	PTBDS5	PTBDS4	R	R	PTBDS1	PTBDS0
写								
复位	0	0	0	0	0	0	0	0

=未使用或预留

图6-11 端口B驱动选择寄存器（PTBDS）

表6－10 PTBDS寄存器字段描述

字段	描述
5: 0 PTBDS[5:1]	**端口A输出驱动强度选择位** — 这些控制位为相关PTB引脚选择低输出驱动和高输出驱动。 0 B端口n位选择低输出驱动强度。 1 B端口n位选择高输出驱动强度。

第七章 中央处理单元（S08CPUV2）

7.1 介绍

这部分提供有关 HCS08 些列 CPU 寄存器，寻址方式，指令设置摘要信息。更详细的信息参考 HCS08 些列参考手册 1，飞思卡尔半导体文件序号 HCS08RMV1/D。HCS08 CPU 与 M68HC08 的 CPU 是完全源和目标代码兼容。一些指令和增强寻址模式的加入以提高 C 编译器效率和支持一个新的背景调试系统以代替早期 M68HC08 的监控模式。

7.1.1 特征

HCS08 CPU 特征

- 目标代码完全向上兼容 M68HC05 和 M68HC08 家族
- 所有寄存器和存储器映射到一个单一的 64 KB 的地址空间
- 16 位堆栈指针（随时随地在 64k 字节的地址空间任何规模的栈）
- 16 位索引寄存器（H:X）带强大的索引地址模式
- 8 位累加器（A）
- 许多指令把 X 作为通用 8 位寄存器
- 7 种寻址模式：
  - 内在寻址方式
  - 相对 8 位有符号偏移量寻址
  - 立即寻址方式
  - 直接寻址方式
  - 扩展寻址方式
  - H:X 相对变址寻址方式
  - SP 相对变址寻址方式
- 内存-内存数据转移指令的四个寻址模式组合
- 溢出，半进位，负，零，和进位条件码支持有条件的分支上带符号，无符号，BCD 码操作数的结果
- 高效率的位操作指令
- 快速 8 位乘 8 位和 16 位除 8 位指令
- STOP 和 WAIT 指令调用低功耗运行模式

7.2 编程结构和CPU寄存器

如图 7-1 所示 5 个 CPU 寄存器，这些 CPU 寄存器不在微处理器的存储器中

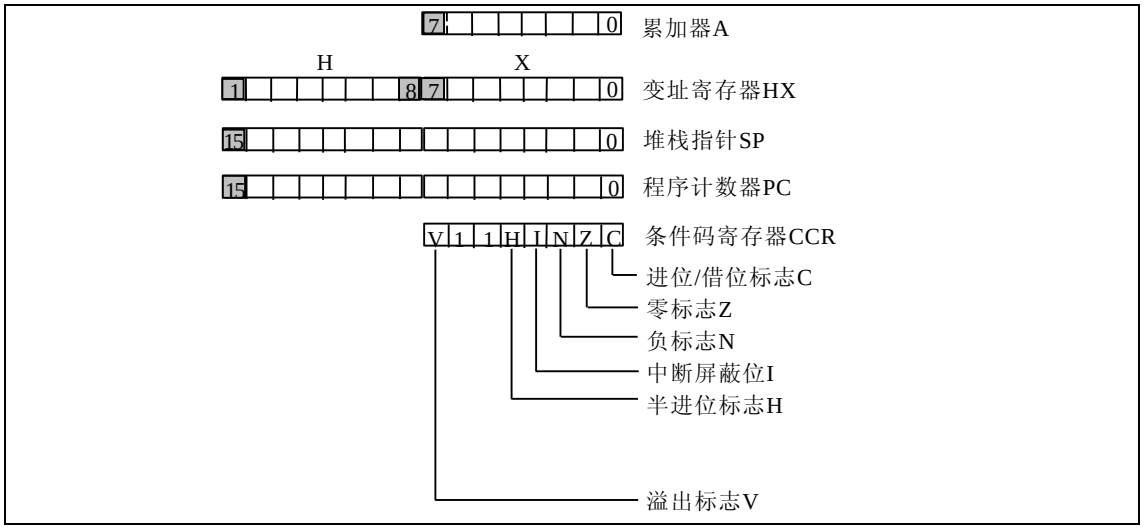


图7-1CPU寄存器

7.2.1 累加器（A）

这个累加器（A）是通用的 8 位寄存器。一个操作数输入到算术逻辑单元（ ALU ）是与累加器和 ALU 结果相关的，在算术和逻辑运算后往往存放入 A 累加器。累加器可以用不同的寻址模式指定地址从存储器中转入数据，或者累加器 A 中的数据可以根据不同的寻址模式制定的地址装入到存储器中。复位键对累加器中的数据无影响。

7.2.2 索引寄存器（H:X）

这个 16 位寄存器实际上是两个 8 位寄存器（H,X）的组合，常作为一个 16 位地址指示器，H 装地址的高字节，X 装地址的低字节。所有索引寻址方式指令用 H:X 中的 16 位全值作为索引参考值。无论怎样，为了和早期 M68HC05 系列兼容，一些指令只能在低 8 位（X）上运行。许多指令把 X 作为通用 8 位寄存器用来存储 8 位数据。X 可以被清零，加，减，取反，移位，循环移位。转移指令允许在算术逻辑操作被执行时数据可以转移到或转移出累加器 A。为了和早期 M68HC05 系列兼容，H 在复位时被设为 0X00。复位对 X 无影响。

7.2.3 堆栈指针（SP）

在一个自动后进先出（LIFO）堆栈中这个 16 位地址指针寄存器指向下一个可用地址。堆栈可以映射到有 RAM 和规模最高限额为可用内存的 64 字节任意地址空间。堆栈可以自动保存子程序调用返回值，中断地址返回值和 CPU 寄存器值，和本地变量。AIS 指令加 8 位有符号立即数给 SP。这是最常用的为本地堆栈变量分配或回收空间。为了兼容早期的 M68HC05 系列 SP 复位后被设为 0XFF。在复位初始化释放直接页 RAM（从片上 RAM 最低地址到 0X00FF）时，HCS08 程序在片上 RAM 中通常将 SP 中的值改为最后地址值（高地址）。RSP（复位对战指针）指令和 M68HC05 系列兼容且很少用在 HCS08 程序中，因为它仅影响对战指针的低位地址。

7.2.4 程序计数器（PC）

程序计数器是一个16位寄存器，包含下一条指令或待取操作数的地址。通常在程序执行

期间，程序计数器在每次数据或指令被取出时会自动增加到存储器的下个连续位置。跳转，分支，中断和返回操作装载一个带地址而不是下个连续位置的程序计数器。这就是流程的改变。

复位时，程序计数器从0XFFFE和0XFFF处装载复位向量。向量存储位置的第一个指令将在退出复位状态后被执行。

7.2.5 条件码寄存器（CCR）

8 位条件码寄存器包含中断屏蔽位 I 和五位状态标志位。第 6 位和第 5 位永远为逻辑 1。下图简要描述了 CCR 各位的信息及功能。详细的用指令设置 CCR 各位，请参阅 HCS08 系列参考手册 1，飞思卡尔半导体文件序号 HCS08RMv1。

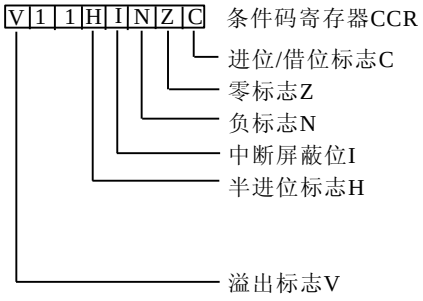


图 7-2条件码寄存器（CCR）

表7-1.CCR寄存器位功能描述

位置	描述
7 V	**溢出标志位**—当二进制补码溢出发生，CPU将设置溢出标志位。有符号指令BGT、BGE、BLE、BLT使用溢出标志。 无溢出 溢出
4 H	**半进位标志**—执行加法指令（ADD）和带进位加法指令（ADC）时，如果累加器D3向D4有进位，CPU设置半进位标志。半进位标志在BCD码算术运算中很有用。DAA指令用H,C的状态来自动调整BCD值。
3 I	**中断屏蔽位**—当中断屏蔽位被设置时，禁止CPU中断。当中断屏蔽位被清除，开放CPU中断。当中断发生时，在CPU寄存器值被保存到堆栈后中断屏蔽被自动设置，但在第一个中断服务例程被执行之前。在任何指令正在清I（CLI、TAP）时，中断不被认可。这确保了CLI或TAP后的指令执行时不被干扰。 允许中断 禁止中断
2 N	**负标志**—CPU进行运算过程中，如果产生负结果则将负标志置1，设置位7的结果。如果装载或存储值的较多重要位为1的话，简单的装载或存储8位或16位值也会引起N置1 。
1 Z	**零标志**—CPU进行运算过程中，如果数据或运算结果为零，零标志置1，否者置零。 无零结果 有零结果 有零结果
0 C	**进位/借位标志**—当进行加法时，在最高位D7上有进位；或在进行减法运算时需要向更高位借位，则CPU将进位/借位标志C置1。一些指令如位测试，跳转，移位指令等也会影响该标志。 无进位/借位 有进位/借位

7.3 寻址模式

寻址模式确定了 CPU 存取数据和操作数的路径。在 HCS08 系列芯片中，所有的存储器，状态和控制寄存器，I/O 口共享一个单一的 64 字节线性地址空间，所以一个 16 位二进制地址可以唯一确定一个存储位置。这样安排意味着存取 RAM 中变量的相同指令同时也能存取 I/O 和控制寄存器或非易失性程序空间。一些指令有不止一种寻址方式。举例来说，MOVE 指令用一种寻址方式来指定源操作数和另一种寻址方式来确定目标地址。当假设条件为真时，一些指令如 BRCLR、BRSET、CBEQ、DBNZ，用一种寻址方式来确定一个操作数地址然后用相对寻址方式来确定转移目标地址。象 BRCLR、BRSET、CBEQ、DBNZ，列在指令设置表上的寻址方式是一种存取操作数来测试，相对寻址方式隐含着转移目标地址的寻址方式。

7.3.1 内在寻址方式（INH）

内在寻址方式，指令中已经包含了操作数所在之处，所以 CPU 不需要存取存储器中的操作数。

7.3.2 相对寻址方式（REL）

相对寻址方式用于为转移指令确定目的地址。8 位有符号的偏移值在操作码之后立即在存储器中被定位。在执行期间，如果转移条件为真，有符号偏移值扩展为 16 位有符号值且加到当前程序计数器内容中，导致程序在转移目标地址处完成。

7.3.3 立即寻址方式（IMM）

立即寻址方式中，在指令操作码之后紧跟着需要完成目标代码中的指令的操作数。在 16 位立即操作数情况下，高位字节在操作码之后定位在写个存储位置，低位字节定位在其之后的下个存储位置。

7.3.4 直接寻址方式（DIR）

直接寻址方式中，在直接页（0X0000-0X00FF）中指令包含地址的低 8 位。在运行期间，通过连接隐含的高位地址 0X00 和指令的直接地址得到一个 16 位地址以获取目标操作数。这比给操作数分配一个全 16 位地址要更快，存取效率更高。

7.3.5 扩展寻址方式（EXT）

在扩展寻址方式中，操作数的 16 位地址被包含在操作码后的两字节目标码内（高位在先）

7.3.6 变址寻址方式

编制寻址方式包含七种，其中五种用到 16 位 H:X 变址寄存器，两种用到堆栈作基值参考。

7.3.6.1 无偏移量变址方式（IX）

在这种变址方式中，变址寄存器 H:X 的内容被用于访问操作数的地址。

7.3.6.2 无偏移量变址、变址加1寻址方式（IX+）

在这种变址方式中，H:X 变址寄存器的内容用来访问目的操作数，然后 H:X 寄存器自增 1。CBEQ 和 MOV 指令是唯一使用这种变址方式的指令。

7.3.6.3 8位偏移量变址方式（IX1）

在这种变址方式中，一个无符号的 8 位偏移量与 H:X 寄存器相加，得到要访问的操作数地址。

7.3.6.4 8位偏移量变址、变址加1寻址方式（IX1+）

在这种变址方式中，一个无符号的 8 位偏移量与 H:X 寄存器相加，得到要访问的操作数地址。在操作数被取出后编制寄存器增 1。CBEQ 是唯一使用这种寻址方式的指令。

7.3.6.5 16位偏移量变址方式（IX2）

在这种变址方式中，一个无符号的 16 位偏移量与 H:X 寄存器相加，得到要访问的操作数地址

7.3.6.6 8位偏移量堆栈寻址方式（SP1）

在这种变址方式中，指令集提供一个无符号的 8 位偏移量与堆栈指针（SP）中的 16 位值相加，得到要访问的操作数地址。

7.3.6.7 16位偏移量堆栈寻址方式（SP2）

在这种变址方式中，无符号的 16 位偏移量与堆栈指针（SP）中的 16 位值相加，得到要访问的操作数地址。

7.4 特殊操作

CPU 的大部分行为在指令集中被详尽描述，但仍需考虑特殊指令的操作。另外一些指令如 STOP，WAIT 直接影响其他 MCU 的电路。这部分提供这些操作的额外信息。

7.4.1 复位序列

复位可以由上电复位、COP 看门狗超时、外部复位引脚引起。当复位事件发生，复位事件强制 CPU 立即停止正在执行的任何事件。想要了解更详细的关于 MCU 如何确认复位及其来源的信息，可参阅复位，中断和系统结构章节。当序列确定复位源是否来自内部和复位引脚不再低电平时复位事件被认为结束。由一个复位事件可知，CPU 需执行一个 6 周期的序列来从 0XFFFE，0XFFFF 处取出复位向量和在准备执行第一个程序指令时填满指令队列。

7.4.2 中断序列

当中断请求发生时，在响应中断之前 CPU 先完成当前指令。这时程序计数器指向下条指令的开始位置，即 CPU 中断返回地址。CPU 对中断的响应是执行相同的操作序列的，软中断（SWI）指令也是一样，除非当中断序列开始时，由高优先级中断决定的用于向量获取的址是未定的。CPU 中断顺序如下：

1. 按序存储 PCL、PCH、X、A、CCR 的内容到堆栈中。
2. 设置 CCR 中的 I 位。
3. 获取中断向量高位。

4. 获取中断向量低位。
5. 延迟一个空闲总线周期。
6. 为中断服务例程将程序信息的首 3 字节填入指令队列为第一条指令执行作准备。

CCR 中的内容被压入堆栈后，在进行中断服务例程时设置 CCR 的 I 位防止其它中断的干扰。虽然也可以通过指令在中断服务例程中清 I，但是可能致嵌套中断发生（这是不被认可的，因为它导致程序难以调试和维护）。为了和早期的 M68HC05MCU 兼容，H:X 编制寄存器中的 H 不作为中断序列的一部分保存到堆栈中。用户必须在终端服务例程开始时用 PSHH 指令保存 H 内容，然后在 RTI 返回之前用 PULH 指令结束中断服务例程。如果你确定中断服务例程不用任何指令或自增寻址方式来改变 H 的值，H 值也可以不被保存。除了软中断不能被屏蔽外，其它和硬中断一样，它是和程序的指令操作码相联系的，所以它不是异步程序执行。

7.4.3 等待模式

WAIT 指令通过清零 CCR 的 I 位是使能中断。然后关闭 CPU 时钟以节省功耗，CPU 维持在低功耗状态下直到中断或复位唤醒它。当中断或复位事件发生时，CPU 从等待模式被唤醒，事件将被正常执行。当 CPU 在等待模式中如果串行的背景调试指令通过背景调试界面流入到 MCU，CPU 时钟将被开启并进入其它串行后台指令能被处理的活跃后台模式。这样确保即使在等待模式中主机开发系统仍能进入目标 MCU。

7.4.4 停止模式

通常在停止模式下，系统的所有时钟包括晶振都被关闭以使功耗最小化。在这样的系统中，外部电路需要控制时间，当继续处理进程时，确保有信号唤醒目标 MCU。不像早期的 M68HC05 和 M68HC08，HCS08 在停止模式中可以设定运行一批最小值时钟。这就允许一个内部周期信号从停止模式中唤醒目标 MCU。当一个主机调试系统连接到背景调试引脚（BKGD）且通过背景调试接口串行命令设置 ENBDM 位时，在 MCU 进入停止模式时振荡器被迫保持活跃。在这种情况下，当 CPU 在停止模式中如果后台命令通过背景调试接口流入到 MCU 中，CPU 时钟将被开启并进入其它串行后台指令能被处理的活跃后台模式。这样确保即使在等待模式中主机开发系统仍能进入目标 MCU。跳出停止模式依靠特殊的 HCS08 和停止模式中振荡器是否被停止，详细参阅操作模式章节。

7.4.5 背景模式

与 M68HC08 相比，对 HCS08 来说背景调试指令是一个新指令。背景不会用在普通用户程序因为它强迫 CPU 停止处理用户指令和进入活跃背景模式。重新继续运行用户程序的唯一方式是通过复位或主机调试系统的背景调试接口输入的 GO、TRACE1、TAGGO 串行命令。软断点可以通过用背景调试操作码替换目标断点地址操作码来设置。当程序到达断点地址时，CPU 更可能进入活跃背景模式而不是继续用户程序。

7.5 HCS08指令设置摘要

指令设置摘要术语表 7-2 中指令术语

操作符：

（）=寄存器或存储器内容显示在圆括号里

← =被装入（读：获得）

& =布尔与
丨=布尔或
⊕=布尔异或
：=连接
x =乘
÷=除
+ =加
- =减

CPU 寄存器：

A=累加器
CCR=条件码寄存器
H=变址寄存器高 8 位
X=变址寄存器低 8 位
PC=程序计数器
PCH=程序计数器高 8 位
PCL=程序计数器低 8 位
SP=堆栈指针

存储和寻址：

M=依赖寻址方式的存储位置或绝对数据
M:M+0X0001=在两个连续存储位置的 16 位值，高 8 位定位到 M 的地址，低 8 位定位到下个高连续地址。
条件寄存器（CCR）位
V=溢出标志，位 7
H=半进位标志，位 4
I=中断屏蔽标志，位 3
N=负标志位，位 2
Z=零标志，位 1
C=进位/借位标志，位 0

CCR 常用符号：

– =位不影响
0 =位改为 0
1 =位改为 1
=根据操作结果位设置或清零
U=操作后未定义

机器码符号：

dd =直接地址 0X0000-0X00FF 低 8 位（假设高字节为 0X00）
ee =16 位偏移量高 8 位
ff =16 位的低 8 位或 8 位
ii = 一字节立即数
jj =16 位立即数值高位字节
kk =16 位立即数值低位字节
hh =16 位扩展地址高位字节

ll =16 位扩展地址低位字节
rr =相对偏移量

源格式：

源格式中除去用斜体表示的表达式其它文字信息必须用汇编源文件正确表示。原始的 3 至 5 个字母一直是一个文字信息。所有的逗号，井号（#），圆括号，加号（+）都是文字字符。

n — 任何用来估值 0-7 中信号数的标签或表达式
opr8i — 任何用来估值 8 位立即数值的标签或表达式
opr16i — 任何用来估值 16 位立即数值的标签或表达式
opr8a — 任何用来估值 8 位立即数值的标签或表达式。指令把 8 位值当作 64 字节直接页地址空间（0X00XX）地址的低 8 位。
opr16a — 任何用来估值 16 位立即数值的标签或表达式。指令把这个值当作 64 字节地址空间的地址
oprx8 —任何用来估值 8 位立即数值的标签或表达式用于变址寻址
oprx16 —任何用来估值 16 位立即数值的标签或表达式。因为 HCS08 有 16 位地址总线，所以是有符号值或无符号值
rel —任何标签或表达式所涉及地址在当前指令末子节目标操作码后的下个地址起-128 至+127 之间。汇编程序将会计算 8 位有符号偏移量的值和把它包含在这条指令飞目标码中。

寻址方式：

INH =内在寻址方式
IMM =立即寻址方式
DIR =直接寻址方式
EXT =扩展寻址方式
IX =无偏移量变址方式
IX+ =无偏移量变址、变址加 1 寻址方式
IX2 =16 位偏移量变址方式
REL =相对寻址方式
SP1 =8 位偏移量堆栈寻址方式
SP2 =16 位偏移量堆栈寻址方式

§

表 7-2 HCS08 指令设置摘要（1/7）

源格式	操作	描述	CCR						寻址方式	操作码	操作数	总线周期
			V	H	I	N	Z	C				
ADC#opr8i	进位加	A □ （A）+（M）+ （C）	↕	↕	–	↕	↕	↕	IMM	A9	ii dd	2
ADCopr8a									DIR	B9	hh	3
ADCopr16a									EXT	C9	ll ee	4
ADCoprx16,X									IX2	D9	ff ff	4
ADC oprx8,X									IX1	E9		3
ADC ,X									IX	F9	ee	3
ADCoprx16,SP									SP2	9ED9	ff	5
ADC oprx8,SP									SP1	9EE9	ff	4
ADD #opr8i	无进位加	A □ （A）+（M）	↕	↕	–	↕	↕	↕	IMM	AB	ii	2
ADD opr8a									DIR	BB	dd	3
ADD opr16a									EXT	CB	hh	4
ADDoprx16,X									IX2	DB	ll ee	4
ADD oprx8,X									IX1	EB FB	ff ff	3
ADD ,X									IX	9EDB		3
ADDoprx16,SP									SP2	9EEB	ee	5
ADD oprx8,SP									SP1		ff	4
											ff	
AIS #opr8i	将立即数（有符号）压入堆栈 r	SP □ （SP） +（M）M 是有符号扩展为 16 位值	–	–	–	–	–	–	IMM	A7	ii	2
AIX #opr8i	将立即数（有符号）压入编制寄存器（H:X）	H:X□ （H:X）+（M）M 是有符号扩展为 16 位值	–	–	–	–	–	–	IMM	AF	ii	2
AND #opr8i	逻辑与	A □ （A）&（M）	0	–	–	↕	↕	–	IMM	A4	ii dd	2
AND opr8a									DIR	B4	hh	3
AND opr16a									EXT	C4	ll ee	4
ANDoprx16,X									IX2	D4	ff ff	4
AND oprx8,X									IX1	E4		3
AND ,X									IX	F4	ee	3
ANDoprx16,SP									SP2	9ED4	ff ff	5
AND oprx8,SP									SP1	9EE4		4

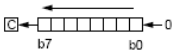
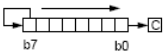
ASLopr8a ASLA ASLX ASLoprx8,X ASL ,X ASL oprx8,SP	算术左移（Same as LSL）		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	38 48 58 68 78 9E68	dd ff ff 	5 1 1 5 4 6
ASRopr8a ASRA ASRX ASRoprx8,X ASR ,X ASR oprx8,SP	算术右移		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	37 47 57 67 77 9E67	dd ff ff 	5 1 1 5 4 6
BCC rel	进位位清零则转移	（C） = 0 则转移	–	–	–	–	–	–	REL	24	rr	3

表 7-2 HCS08 指令设置摘要（2/7）

源格式	操作	描述	CCR						寻址方式	操作码	操作数	总线周期
			V	H	I	N	Z	C				
BCLR n,opr8a	存储器中位 n 清零	Mn □ 0	–	–	–	–	–	–	DIR （b0） DIR （b1） DIR （b2） DIR （b3） DIR （b4） DIR （b5） DIR （b6） DIR （b7）	11 13 15 17 19 1B 1D 1F	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BCS rel	进位位置 1 则转移（Same as BLO）	（C） = 1 则转移	–	–	–	–	–	–	REL	25	rr	3
BEQ rel	相等则转移	（Z） = 1 则转移	–	–	–	–	–	–	REL	27	rr	3
BGE rel	大于等于则转移（Signed Operands）	（N □ V̨ □ ˜则转移	–	–	–	–	–	–	REL	90	rr	3
BGND	ENBDM = 1 则进入活跃背景调试状态	等待和处理 BDM 指令直到 GO, TRACE1, TAGGO	–	–	–	–	–	–	INH	82		5+
BGT rel	大于则转移（Signed Operands）	（Z） | （N □ V̨ □ ˜则转移	–	–	–	–	–	–	REL	92	rr	3
BHCC rel	半进位清零则转移	（H） = 0 则转移	–	–	–	–	–	–	REL	28	rr	3
BHCS rel	半进位置 1 则转移	（H） = 1 则转移	–	–	–	–	–	–	REL	29	rr	3
BHI rel	为高则转移	（C） | （Z） = 0 则转移	–	–	–	–	–	–	REL	22	rr	3
BHS rel	高于或相同则转移（Same as BCC）	（C） = 0 则转移	–	–	–	–	–	–	REL	24	rr	3
BIH rel	IRQ 引脚为高则转移	IRQ pin = 1 则转移	–	–	–	–	–	–	REL	2F	rr	3
BIL rel	Branch if IRQ Pin Low	IRQ pin = 0 则转移	–	–	–	–	–	–	REL	2E	rr	3

BIT#opr8i BITopr8a BITopr1a BIToprx16,X BIT oprx8,X BIT ,X BIToprx16,SP BIT oprx8,SP	位测试	（A）&（M） CCR 更新但操作数不变	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX SP2 SP1	A5 B5 C5 D5 E5 F5 9ED5 9EE5	ii dd hh ll ee ff ff ee ff ff	2 3 4 4 3 3 5 4
BLE rel	小于等于则转移（Signed Operands）	（Z）|（N □ V□ 1 则转移	–	–	–	–	–	–	REL	93	rr	3
BLO rel	为低则转移（Same as BCS）	（C） = 1 则转移	–	–	–	–	–	–	REL	25	rr	3
BLS rel	为低或相同则转移	（C）| （Z） = 1 则转移	–	–	–	–	–	–	REL	23	rr	3
BLT rel	小于则转移（Signed Operands）	（N □ V□ 1则转移	–	–	–	–	–	–	REL	91	rr	3
BMC rel	中断屏蔽位清零则转移	（I） = 0 则转移	–	–	–	–	–	–	REL	2C	rr	3
BMI rel	负则转移	（N） = 1 则转移	–	–	–	–	–	–	REL	2B	rr	3
BMS rel	中断屏蔽位置 1 则转移	（I） = 1 则转移	–	–	–	–	–	–	REL	2D	rr	3
BNE rel	不等则转移	（Z） = 0 则转移	–	–	–	–	–	–	REL	26	rr	3
BPL rel	正则转移	（N） = 0 则转移	–	–	–	–	–	–	REL	2A	rr	3
BRA rel	一直转移	无测试	–	–	–	–	–	–	REL	20	rr	3

表 7-2 HCS08 指令设置摘要（3/7）

源格式	操作	描述	CCR						寻址方式	操作码	操作数	总线周期
			V	H	I	N	Z	C				
BRCLR n,opr8a,rel	存储器中位 n 清零	（Mn） = 0 则转移	–	–	–	–	–	↕	DIR （b0） DIR （b1） DIR （b2） DIR （b3） DIR （b4） DIR （b5） DIR （b6） DIR （b7）	01 03 05 07 09 0B 0D 0F	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd	5 5 5 5 5 5 5 5
BRN rel	从不转移	占用 3 个总线周期	–	–	–	–	–	–	REL	21	rr	3
BRSET n,opr8a,rel	存储器中位 n 置 1 则转移	（Mn） = 1 则转移	–	–	–	–	–	↕	DIR （b0） DIR （b1） DIR （b2） DIR （b3） DIR （b4） DIR （b5） DIR （b6） DIR （b7）	00 02 04 06 08 0A 0C 0E	dd rr dd rr dd rr dd rr dd rr dd rr dd rr dd	5 5 5 5 5 5 5 5

BSET n,opr8a	存储器中位 n 置 1	Mn□ 1	–	–	–	–	–	–	DIR（b0） DIR（b1） DIR（b2） DIR（b3） DIR（b4） DIR（b5） DIR（b6） DIR（b7）	10 12 14 16 18 1A 1C 1E	dd dd dd dd dd dd dd dd	5 5 5 5 5 5 5 5
BSR rel	转移到子程序	PC □ （PC） + 0x0002 push （PCL）; SP □ （SP） – 0x0001push （PCH）; SP □ （SP）– 0x0001PC □ （PC） + rel	–	–	–	–	–	–	REL	AD	rr	5
CBEQopr8a,rel CBEQA#opr8i,rel CBEQX#opr8i,rel CBEQoprx8,X+,rel CBEQ ,X+,rel CBEQoprx8,SP,rel	相等则比较转移	（A） = （M）则转移 （A） =（M）则转移（X）=（M）则转移 （A） = （M） 则转移（A） = （M） 则转移	–	–	–	–	–	–	DIR IMM IMM IX1+ IX+ SP1	31 41 51 61 71 9E61	dd rr ii rr ii rr ff rr rr ff	5 4 4 5 5 6
CLC	清进位位	C □ 0	–	–	–	–	–	0	INH	98		1
CLI	清中断屏蔽位	I □ 0	–	–	0	–	–	–	INH	9A		1
CLRopr8a CLRA CLRX CLRH CLRoprx8,X CLR ,X CLR oprx8,SP	清位	M□ 0x00 A□ 0x00 X□ 0x00 H□ 0x00 M□ 0x00 M□ 0x00 M□ 0x00	0	–	–	0	1	–	DIR INH INH INH IX1 IX SP1	3F 4F 5F 8C 6F 7F 9E6F	dd ff ff 	5 1 1 1 5 4 6
CMP #opr8i CMP oprr8a CMP opr16a CMPoprx16,X CMP oprx8,X CMP ,X CMP oprx16,SP CMP oprx8,SP	存储器与累加器相比	（A）&（M） CCR 更新但操作数 不变	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2 SP1	A1 B1 C1 D1 E1 F1 9ED1 9EE1	ii dd hh ll ee ff ff ee ff ff 	2 3 4 4 3 3 5 4

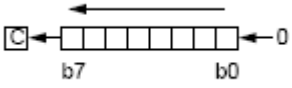
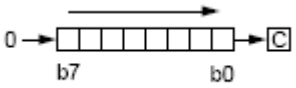
COMopr8a COMA	求补	M□（M）=0xFF–（M） A□							DIR INH	33	dd	5
COMX	（One’s	（A）= 0xFF–（A） X□（X）							INH IX1	43		1
COMoprx8,X	Complement）	= 0xFF–（X） M□（M）	0	–	–	↕	↕	1	IX SP1	53	ff ff	1
COM ,X		=0xFF–（M） M□（M）								63		5
COM oprx8,SP		=0xFF–（M） M□（M）								73		4
		=0xFF–（M）								9E63		6

表 7-2 HCS08 指令设置摘要（4/7）

源格式	操作	描述	CCR						寻址方式	操作码	操作数	总线周期
			V	H	I	N	Z	C				
CPX #opr8i CPX									IMM DIR	A3	ii dd	2
opr8a CPX opr16a		（X）-（M）							EXT	B3	hh ll	3
CPX oprx16,X CPX	X 与存储器相比								IX2	C3	ee ff	4
oprx8,X CPX ,X		CCR 更新但操作数不变	↕	–	–	↕	↕	↕	IX1	D3	ff	4
DAA	在 BCD 码 ADD,ADC 操作后转换累加器内容到	（A）10	U	–	–	↕	↕	↕	INH	72		1
DBNZ opr8a,rel									DIR INH	3B	dd rr	7
DBNZA rel		A, X, or M 自减 （result） □ 0则							INH	4B	rr	4
DBNZX rel	不为零则自减转移	转移	–	–	–	–	–	–	IX1	5B	rr	4
DBNZ oprx8,X,rel		DBNZX 影响 X 不影响 H							IX	6B	ff	7
DEC opr8a DECA		M □ （M） – 0x01							DIR INH	3A	dd	5
DECX		A □ （A） – 0x01							INH IX1	4A		1
DEC oprx8,X	自减	X □ （X） – 0x01	↕	–	–	↕	↕	–	IX SP1	5A	ff ff	1
DEC ,X		M □ （M） – 0x01								6A		5
DIV	除	A □ （H:A）□（X） H □ 余数	–	–	–	–	↕	↕	INH	52		6
EOR #opr8i EOR									IMM DIR	A8	ii dd	2
opr8a EOR opr16a									EXT IX2	B8	hh ll	3
EOR oprx16,X EOR	带累加器的存储器异或								IX1	C8	ee ff	4
oprx8,X EOR ,X		A □ （A □ M）	0	–	–	↕	↕	–	IX	D8	ff	4
INC opr8a		M □ （M） + 0x01							DIR INH	3C	dd	5
INCA		A □ （A） + 0x01							INH IX1	4C		1
INCX	自增	X □ （X） + 0x01	↕	–	–	↕	↕	–	IX SP1	5C	ff ff	1
INC oprx8,X		M □ （M） + 0x01								6C		5
JMP opr8a JMP									DIR EXT	BC CC	dd	3
opr16a JMP									IX2	DC EC	hh ll	4
oprx16,X JMP	跳转	PC □ 跳转地址	–	–	–	–	–	–	IX1	FC	ee ff	4
JSR opr8a JSR									DIR EXT	BD CD	dd	5
opr16a JSR oprx16,X		PC □ （PC） + n （n = 1, 2, or 3）							IX2	DD ED	hh ll	6
JSR oprx8,X	跳转到子程序	Push （PCL）； SP □ （SP）	–	–	–	–	–	–	IX1	FD	ee ff	6

LDA #opr8i LDA opr8a LDA opr16a LDA oprx16,X LDA oprx8,X LDA ,X	从存储器装入累加器	A □ （M）	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX	A6 B6 C6 D6	ii dd hh ll ee ff ff	2 3 4 4
LDHX #opr16i LDHX opr8a LDHX opr16a LDHX ,X	从存储器装入变址寄存器	H:X □ ¸M:M + 0x0001	0	–	–	↕	↕	–	IMM DIR EXT IX IX2 IX1	45 55 32 9EAE	jj kk dd hh ll	3 4 5 5

表 7-2 HCS08 指令设置摘要（5/7）

源格式	操作	描述	CCR						寻址方式	操作码	操作数	总线周期
			V	H	I	N	Z	C				
LDX #opr8i LDX opr8a LDX opr16a LDX oprx16,X LDX oprx8,X LDX ,X	从存储器中装入 X	X □ （M）	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX	AE BE CE DE EE FE 9EDE	ii dd hh ll ee ff ff	2 3 4 4
LSL oprx8a LSLA LSLX LSL oprx8,X	逻辑左移（Same as ASL）		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX	38 48 58 68	dd ff ff	5 1 1 5
LSR oprx8a LSRA LSRX LSR oprx8,X LSR ,X	逻辑右移		↕	–	–	0	↕	↕	DIR INH INH IX1 IX	34 44 54 64	dd ff ff	5 1 1 5
MOV oprx8a,oprx8a MOV oprx8a,X+	Move	（M）destination □ （M）source H:X □ （H:X） + 0x0001 in	0	–	–	↕	↕	–	DIR/DIR DIR/IX+	4E 5E	dd dd dd	5 5
MUL	无符号相乘	X:A □ （X） □ （A）	–	0	–	–	–	0	INH	42		5
NEG oprx8a NEGA NEGX NEG oprx8,X NEG ,X	取负（Two’s Complement）	M □ – （M） = 0x00 – （M） A □ – （A） = 0x00 – （A） X □ – （X） = 0x00 – （X） M □ – （M） = 0x00 – （M） M □ – （M） = 0x00 –		–	–	↕	↕	↕	DIR INH INH IX1 IX SP1	30 40 50 60	dd ff ff	5 1 1 5
NOP	空操作	Uses 1 Bus Cycle	–	–	–	–	–	–	INH	9D		1
NSA	累加器半位元组交换	A □ （A[3:0]:A[7:4]）	–	–	–	–	–	–	INH	62		1
ORA #opr8i ORA opr8a ORA opr16a ORA oprx16,X ORA oprx8,X ORA ,X	累加器或上存储器	A □ （A） | （M）	0	–	–	↕	↕	–	IMM DIR EXT IX2 IX1 IX	AA BA CA DA EA FA	ii dd hh ll ee ff ff	2 3 4 4
PSHA	把累加器压入堆栈	Push （A）; SP □ （SP） – 0x0001	–	–	–	–	–	–	INH	87		2
PSHH	把 H 压入堆栈	Push （H）; SP □ （SP） – 0x0001	–	–	–	–	–	–	INH	8B		2
PSHX	把 X 压入堆栈	Push （X）; SP □ （SP） – 0x0001	–	–	–	–	–	–	INH	89		2
PULA	累加器出栈	SP □ （SP + 0x0001）; Pull¸A	–	–	–	–	–	–	INH	86		3

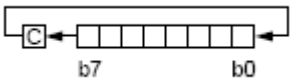
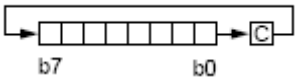
PULH	H 出栈	SP □ （SP + 0x0001）; Pull H	–	–	–	–	–	–	INH	8A		3
PULX	X 出栈	SP □ （SP + 0x0001）; Pull X	–	–	–	–	–	–	INH	88		3
ROL opr8a ROLA	进位循环左移		↕	–	–	↕	↕	↕	DIR INH	39	dd	5
ROLX									INH	49		1
ROL oprx8,X									IX1	59	ff ff	1
ROL ,X									IX	69		5

表 7-2 HCS08 指令设置摘要（6/7）

源格式	操作	描述	CCR						寻址方式	操作码	操作数	总线周期
			V	H	I	N	Z	C				
ROR opr8a RORA RORX ROR oprx8,X ROR ,X	进位循环右移		↕	–	–	↕	↕	↕	DIR INH INH IX1 IX	36 46 56 66	dd ff ff 	5 1 1 5
RSP	堆栈复位	SP □ 0xFF （高字节不影响）	–	–	–	–	–	–	INH	9C		1
RTI	中断返回	SP □ （SP） + 0x0001; Pull （CCR） SP □ （SP） + 0x0001; Pull （A） SP □ （SP） + 0x0001; Pull （X）	↕	↕	↕	↕	↕	↕	INH	80		9
RTS	从子程序返回	SP □ SP + 0x0001□ Pull(PCH) SP □ SP + 0x0001; Pull （PCL）	–	–	–	–	–	–	INH	81		6
SBC #opr8i SBC opr8a SBC opr16a SBC oprx16,X SBC oprx8,X SBC ,X	带进位减	A □ （A） – （M） – （C）	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1	A2 B2 C2 D2	ii dd hh ll ee ff ff	2 3 4 4
SEC	进位位置位	C □ 1	–	–	–	–	–	1	INH	99		1
SEI	中断屏蔽位置位	I □ 1	–	–	1	–	–	–	INH	9B		1
STA oprx16,X STA oprx8,X STA ,X	将累加器中内容存储到存储器	M □ （A）	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX	B7 C7 D7 E7	dd hh ll ee ff ff	3 4 4 3
STHX opr8a STHX opr16a STHX	存储编制寄存器内容	（M:M + 0x0001） □ （H:X）	0	–	–	↕	↕	–	DIR EXT SP1	35 96	dd hh ll	4 5
STOP	中断使能 停止处理涉及 MCU 的文件	I bit □ 0; Stop Processing	–	–	0	–	–	–	INH	8E		2+
STX opr8a STX opr16a STX oprx16,X STX oprx8,X STX ,X STX oprx16,SP STX	存储 X 到存储器	M □ （X）	0	–	–	↕	↕	–	DIR EXT IX2 IX1 IX SP2	BF CF DF EF FF 9EDF 9EEF	dd hh ll ee ff ff	3 4 4 3 2

SUB #opr8i SUB opr8a SUB opr16a SUB oprx16,X SUB oprx8,X SUB ,X SUB oprx16,SP SUB oprx8,SP	减	A □ （A） – （M）	↕	–	–	↕	↕	↕	IMM DIR EXT IX2 IX1 IX SP2	A0 B0 C0 D0 E0 F0	ii dd hh ll ee ff ff	2 3 4 4 3 3
SWI	软中断	PC □ （PC） + 0x0001 Push（PCL）; SP □ （SP） – 0x0001 Push（PCH）; SP □ （SP） – 0x0001 Push （X）; SP □ （SP） – 0x0001 Push （A）; SP □ （SP） – 0x0001 Push（CCR）; SP □ （SP） – 0x0001	–	–	1	–	–	–	INH	83		11

表 7-2 HCS08 指令设置摘要（7/7）

源格式	操作	描述	CCR						寻址方式	操作码	操作数	总线周期
			V	H	I	N	Z	C				
TAP	转移累加器到 CCR	CCR □ （A）	↕	↕	↕	↕	↕	↕	INH	84		1
TAX	转移累加器到 X	X □ （A）	–	–	–	–	–	–	INH	97		1
TPA	转移 CCR 到累加器	A □ （CCR）	–	–	–	–	–	–	INH	85		1
TST opr8a TSTA TSTX TST oprx8,X	测试零或负数	（M） – 0x00 （A） – 0x00 （X） – 0x00	0	–	–	↕	↕	–	DIR INH INH IX1 IX SP1	3D 4D 5D	dd ff ff	4 1 1
TSX	转移 SP 到 H:X.	H:X □ （SP） + 0x0001	–	–	–	–	–	–	INH	95		2
TXA	转移 X 到累加器	A □ （X）	–	–	–	–	–	–	INH	9F		1
TXS	转移 H:X 到 SP	SP □ （H:X） – 0x0001	–	–	–	–	–	–	INH	94		2
WAIT	中断使能，等待中断	I bit □ 0; Halt CPU	–	–	0	–	–	–	INH	8F		2+

总线的时钟频率是 CPU 时钟频率一半。

表 7-3 操作码图（1/2）

Bit-Manipula		Br	Read-M					C		Regist					
00 5 BRS ET0	10 5 BS ET0	20 3 B RA	30 5 N EG	40 1 NE GA	50 1 NE GX	60 5 N EG	70 4 N EG	80 9 R TI	90 3 B GE	A0 2 S UB	B0 3 S UB	C0 4 S UB	D0 4 S UB	E0 3 S UB	F0 3 S UB
01 5 BRC LR0	11 5 BC LR0	21 3 B RN	31 5 CB EQ	41 4 CB EQA	51 4 CB EQX	61 5 CB EQ	71 5 CB EQ	81 6 R TS	91 3 B LT	A1 2 C MP	B1 3 C MP	C1 4 C MP	D1 4 C MP	E1 3 C MP	F1 3 C MP
02 5 BRS ET1	12 5 BS ET1	22 3 B HI	32 5 LD HX	42 5 M UL	52 6 D V	62 1 N SA	72 1 D AA	825+ BG ND	92 3 B GT	A2 2 S BC	B2 3 S BC	C2 4 S BC	D2 4 S BC	E2 3 S BC	F2 3 S BC
03 5 BRC LR1	13 5 BC LR1	23 3 B LS	33 5 C OM	43 1 CO MA	53 1 CO MX	63 5 C OM	73 4 C OM	8311 S WI	93 3 B LE	A3 2 C PX	B3 3 C PX	C3 4 C PX	D3 4 C PX	E3 3 C PX	F3 3 C PX
04 5 BRS ET2	14 5 BS ET2	24 3 B CC	34 5 L SR	44 1 LS RA	54 1 LS RX	64 5 L SR	74 4 L SR	84 1 TA P	94 2 T XS	A4 2 A ND	B4 3 A ND	C4 4 A ND	D4 4 A ND	E4 3 A ND	F4 3 A ND
05 5 BRC LR2	15 5 BC LR2	25 3 B CS	35 4 ST HX	45 3 LD HX	55 4 LD HX	65 3 CP HX	75 5 CP HX	85 1 T PA	95 2 T SX	A5 2 B IT	B5 3 B IT	C5 4 B IT	D5 4 B IT	E5 3 B IT	F5 3 B IT
06 5 BRS ET3	16 5 BS ET3	26 3 B NE	36 5 R OR	46 1 RO RA	56 1 RO RX	66 5 R OR	76 4 R OR	86 3 PU LA	96 5 ST HX	A6 2 L DA	B6 3 L DA	C6 4 L DA	D6 4 L DA	E6 3 L DA	F6 3 L DA

07 5 BRC LR3	17 5 BC LR3	27 3 B EQ	37 5 A SR	47 1 AS RA	57 1 AS RX	67 5 A SR	77 4 A SR	87 2 PS HA	97 1 TA X	A7 2 AI S	B7 3 S TA	C7 4 S TA	D7 4 S TA	E7 3 S TA	F7 2 S TA
08 5 BRS ET4	18 5 BS ET4	28 3 BH CC	38 5 L SL	48 1 LS LA	58 1 LS LX	68 5 L SL	78 4 L SL	88 3 PU LX	98 1 C LC	A8 2 E OR	B8 3 E OR	C8 4 E OR	D8 4 E OR	E8 3 E OR	F8 3 E OR
09 5 BRC LR4	19 5 BC LR4	29 3 BH CS	39 5 R OL	49 1 RO LA	59 1 RO LX	69 5 R OL	79 4 R OL	89 2 PS HX	99 1 S EC	A9 2 A DC	B9 3 A DC	C9 4 A DC	D9 4 A DC	E9 3 A DC	F9 3 A DC
0A 5 BRS ET5	1A 5 BS ET5	2A 3 B PL	3A 5 D EC	4A 1 DE CA	5A 1 DE CX	6A 5 D EC	7A 4 D EC	8A 3 PU LH	9A 1 C LI	AA 2 O RA	BA 3 O RA	CA 4 O RA	DA 4 O RA	EA 3 O RA	FA 3 O RA
0B 5 BRC LR5	1B 5 BC LR5	2B 3 B MI	3B 7 DB NZ	4B 4 DB NZA	5B 4 DB NZX	6B 7 DB NZ	7B 6 DB NZ	8B 2 PS HH	9B 1 S EI	AB 2 A DD	BB 3 A DD	CB 4 A DD	DB 4 A DD	EB 3 A DD	FB 3 A DD
0C 5 BRS ET6	1C 5 BS ET6	2C 3 B MC	3C 5 IN C	4C 1 IN CA	5C 1 IN CX	6C 5 IN C	7C 4 IN C	8C 1 CL RH	9C 1 R SP		BC 3 J MP	CC 4 J MP	DC 4 J MP	EC 3 J MP	FC 3 J MP
0D 5 BRC LR6	1D 5 BC LR6	2D 3 B MS	3D 4 T ST	4D 1 TS TA	5D 1 TS TX	6D 4 T ST	7D 3 T ST		9D 1 N OP	AD 5 B SR	BD 5 J SR	CD 6 J SR	DD 6 J SR	ED 5 J SR	FD 5 J SR
0E 5 BRS ET7	1E 5 BS ET7	2E 3 B IL	3E 6 CP HX	4E 5 M OV	5E 5 M OV	6E 4 M OV	7E 5 M OV	8E 2+ ST	9E Pag e 2	AE 2 L DX	BE 3 L DX	CE 4 L DX	DE 4 L DX	EE 3 L DX	FE 3 L DX
0F 5 BRC LR7	1F 5 BC LR7	2F 3 BI H	3F 5 C LR	4F 1 CL RA	5F 1 CL RX	6F 5 C LR	7F 4 C LR	8F 2+ WA IT	9F 1 T XA	AF 2 AI X	BF 3 S TX	CF 4 S TX	DF 4 S TX	EF 3 S TX	FF 2 S TX

INH 隐含寻址
IMM 立即寻址
DIR 直接寻址
EXT 扩展寻址
DD 直接-直接寻址
IX+D 变址-直接-变址加

INH 隐含寻址
IMM 立即寻址
DIR 直接寻址
EXT 扩展寻址
DD 直接-直接寻址
IX+D 变址-直接-变址加

INH 隐含寻址
IMM 立即寻址
DIR 直接寻址
EXT 扩展寻址
DD 直接-直接寻址
IX+D 变址-直接-变址加

用16进制表示操作字节数

F0		3
	SUB	
1		IX

HCS08 指令周

寻址方式

表 7-3 操作码图（1/2）

Bit-Manipulation		Branch	Read-Modify-Write					Control		Register/Memory					
						9E60 6 NSA							9ED0 5 SUB	9EE0 4 SUB	
						9E61 6 CBEQ							9ED1 5 CBEQ	9EE1 4 CBEQ	
													9ED2 5 SUB	9EE2 4 SUB	
						9E63 6 CBEQ							9ED3 5 CBEQ	9EE3 4 CBEQ	9EF3 6 CPHX
						9E64 6 LDHX							9ED4 5 LDHX	9EE4 4 LDHX	
													9ED5 5 LDHX	9EE5 4 LDHX	
						9E66 6 LDHX							9ED6 5 LDHX	9EE6 4 LDHX	
						9E67 6 STHX							9ED7 5 STHX	9EE7 4 STHX	
						9E68 6 LDHX							9ED8 5 LDHX	9EE8 4 LDHX	
						9E69 6 STHX							9ED9 5 STHX	9EE9 4 STHX	
						9E6A 6 DBNZX							9EDA 5 DBNZX	9EEA 4 DBNZX	
						9E6B 8 DBNZX							9EDB 5 DBNZX	9EEB 4 DBNZX	

						9E6C 6 I									
						9E6D 5 T									
										9EAE 5 LD	9EBE 6 LD	9ECE 5 LD	9EDE 5 L	9EEE 4 L	9EFE 5 LDHX
						9E6F 6 C							9EDF 5 S	9EEF 4 S	9EFF 5 STHX

INH 隐含寻址
IMM 立即寻址
DIR 直接寻址
EXT 扩展寻址
DD 直接-直接寻址
IX+D 变址-直接-变址加

INH 隐含寻址
IMM 立即寻址
DIR 直接寻址
EXT 扩展寻址
DD 直接-直接寻址
IX+D 变址-直接-变址加

INH 隐含寻址
IMM 立即寻址
DIR 直接寻址
EXT 扩展寻址
DD 直接-直接寻址
IX+D 变址-直接-变址加

用16进制表示操作码 9E60 6 NEG
字节数 3 SP1

HCS08指令周
寻址方式

第八章 键盘中断（S08KBIV2）

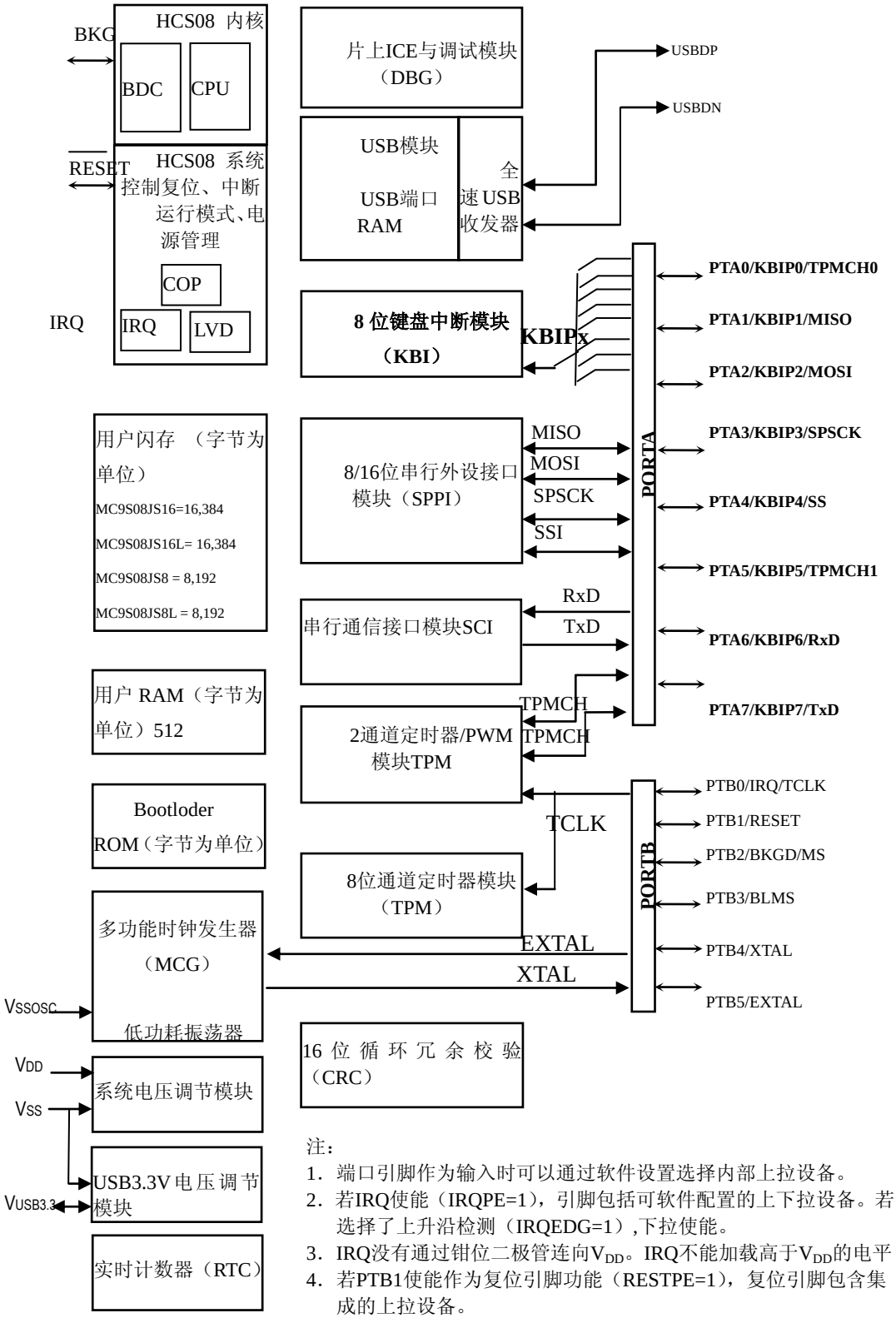
8.1 简介

键盘中断（KBI）模块提供了多达 8 个独立使能的外部中断源。MC9S08JS16 系列芯片包含了一个多达 8 个中断源的 KBI 模块。

注：

当使能 KBI 引脚时，KBF 将被置位，而且必须事先被清除以允许中断发生。当在一个 5V 系统中设置该引脚为下降沿电平敏感时，在清除标志位和允许中断之前必须等待至少一个周期。

图 8-1 显示了芯片层面上的框图，KBI 模块被高亮显示。



注：

1. 端口引脚作为输入时可以通过软件设置选择内部上拉设备。
2. 若IRQ使能（IRQPE=1），引脚包括可软件配置的上下拉设备。若选择了上升沿检测（IRQEDG=1）,下拉使能。
3. IRQ没有通过钳位二极管连向V_{DD}。IRQ不能加载高于V_{DD}的电平
4. 若PTB1使能作为复位引脚功能（RESTPE=1），复位引脚包含集成的上拉设备。
5. 引脚包含集成上拉设备。
6. 当KBI使能（KBIPEn=1）而且相关引脚被配置为上拉设备使能，那么上升沿（KBEDGn=1）被用于将上拉设备配置为下拉设备。

图 8-1. MC9S08JS16系列框图KBI模块和引脚高亮显示

8.1.1 特征

KBI 的特征包括

- 多达 8 个键盘中断引脚，每个引脚有独立的允许位。
- 每个键盘中断引脚只能被写为下降沿（或上升沿），或下降沿低电平（或上升沿高电平）中断敏感。
- 一个软件允许键盘中断。
- 从低电压模式离开。

8.1.2 操作模式

这节定义了 KBI 在等待，停止和背景调试模式中的操作。

8.1.2.1 KBI在等待模式中

如果 KBI 在执行 WAIT 指令前被使能，则 KBI 在等待模式中继续操作。因此，在 KBI 中断被允许的时候（KBIE=1），一个被允许的 KBI 引脚（KBPEx=1）可以被用来将 MCU 从等待模式中唤醒。

8.1.2.2 KBI在停止模式中

如果在执行 STOP 指令前 KBI 被使能则 KBI 在 STOP3 模式中不同步的进行操作。因此，在 KBI 中断被允许的时候（KBIE=1）一个被允许的 KBI 引脚（KBPEx=1）可以被用来将 MCU 从 STOP3 模式中唤醒。

不管在 STOP1 还是 STOP2 模式中，KBI 都被禁止。在一些系统中，与 KBI 相关的引脚可能被用来将系统从 STOP1 或 STOP2 模式中唤醒，见“操作模式”章停止模式小节。当从 STOP1 或 STOP2 模式中被唤醒时，KBI 模块将会处于复位状态。

8.1.2.3 KBI在激活背景模式中

当微控制器处于激活背景模式，KBI 将会继续正常操作。

8.1.3 框图

图 8-2 显示了键盘中断模块的框图。

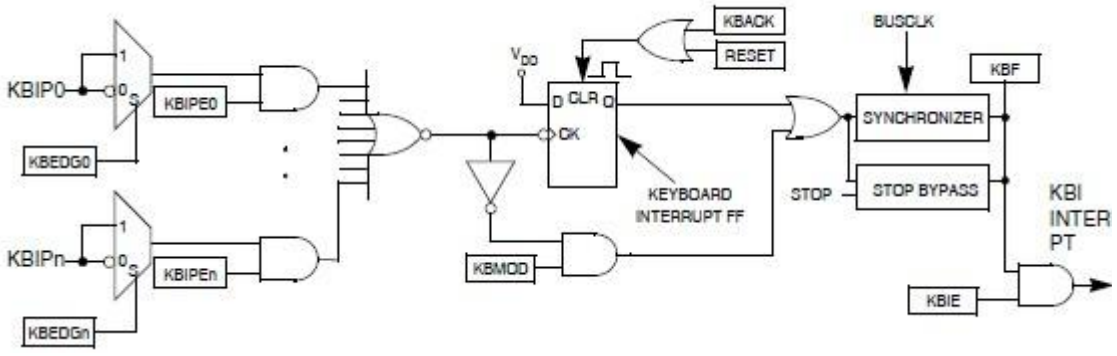


图8-2 KBI框图

8.2 外部信号描述

KBI 输入引脚可以用来检测下降沿或下降沿低电平中断请求。KBI 输入引脚可同样被用于检测上升沿或上升沿高电平中断请求。

KBI 的信号特征列于表 8-1。

表8-1 信号特征

信号	功能	I/O
KBIPn	键盘中断引脚	I

8.3 寄存器定义

KBI 包含 3 个寄存器：

- 一个 8 位的引脚状态和控制寄存器
- 一个 8 位的引脚使能寄存器
- 一个 8 位的沿探测寄存器

参考第四章，“存储器”中的直接页寄存器总结见所有 KBI 寄存器的绝对地址。这一节仅通过它们的名字来提到它们。

一些 MCU 也许包含不止一个 KBI，所以寄存器名字中会包含占位描述符来指明提到的是哪个 KBI。

8.3.1 KBI状态和控制寄存器（KBISC）

KBISC 包含状态位和控制位，控制位用于设置 KBI 功能。

	第7位	6	5	4	3	2	1	第0位
读	0	0	0	0	KBF	0	KBIE	KBIMOD
写						KBACK		
复位	0	0	0	0	0	0	0	0

=未使用

图8-3 KBI状态控制寄存器（KBISC）

表 8-2. KBI1SC寄存器字段描述

字段	描述
7:4	未使用寄存器位，永远读为0。
3 KBF	**键盘中断标志**—KBF显示了键盘中断何时被检测到。对该位写入没有任何意义。 0 没有KBI中断被检测到。 1 KBI中断被检测到。
2 KBACK	**键盘中断确认—**写1到KBACK是标志位清零机制的一部分。KBACK位永远读为0。
1 KBIE	**键盘中断使能**—KBIE决定了一个键盘中断请求是否被允许。 0 键盘中断请求被禁止。 1 键盘中断请求被允许。
KBIMOD	**键盘检测模块**—KBMOD（与KBEDG位一起）控制键盘中断引脚的检测模式。 0 键盘只探测沿。 1 键盘同时探测沿河电平。

8.3.2 KBI引脚使能寄存器（KBIPE）

	第7位	6	5	4	3	2	1	第0位
读	KBIPE7	KBIPE6	KBIPE5	KBIPE4	KBIPE3	KBIPE2	KBIPE1	KBIPE0

写								
复位	0	0	0	0	0	0	0	0

图8-4 KBI引脚使能寄存器（KBIPE）

表 8-3. KBIPE寄存器字段描述

字段	描述
7:0 KBIPEn	**键盘引脚使能位—**每个KBIPEn位使能对应的键盘中断引脚。 0 引脚键盘中断功能被禁止。 1 引脚键盘中断功能被允许。

8.3.3 KBI沿探测寄存器（KBIES）

KBIES 包含了沿选择控制位。

	第7位	6	5	4	3	2	1	第0位
读 写	KBEDG7	KBEDG6	KBEDG5	KBEDG4	KBEDG3	KBEDG2	KBEDG1	KBEDG0
复位	0	0	0	0	0	0	0	0

图8-5 KBI沿探测寄存器（KBIES）

表8-4 KBIES寄存器域描述

域	描述
7:0 KBEDGn	**键盘边沿探测**——每个KBEDGn位为对应引脚选择探测下降沿/低电平或上升沿/高电平。 0下降沿/低电平 1 上升沿/高电平

8.4.1 沿敏感

同步逻辑用来检测边沿。当一个使能键盘中断（KBIPEn=1）输入信号在一个总线周期内为逻辑 1（未声明电平），然后在下一个总线周期内为逻辑 0（声明电平）时，检测下降沿。当一个使能键盘中断（KBIPEn=1）输入信号在一个总线周期内为逻辑 0（未声明电平），然后在下一个总线周期内为逻辑 1（声明电平）时，检测上升沿。在第一个沿被检测到之前，所有使能键盘中断输入信号必须在未声明逻辑电平上。沿被检测到以后，所有使能键盘中断输入信号必须在下一个沿被探测到之前重回未声明电平。

一个在使能KBI引脚上的合法边沿将会将KBISC中的KBF置位。如果KBISC中的KBIE被置位，一个中断请求将被提交给 CPU。向 KBISC 中 KBACK 写 1 清零 KBF。

8.4.2 边沿电平敏感

一个在使能的 KBI 引脚上的合法的边沿或电平会将 KBISC 中的 KBF 置位。如果 KBISC 中的 KBIE 被置位，一个中断请求将被提交给 CPU。假定所有的使能 KBI 输入都在它们的未声明电平，向 KBISC 中 KBACK 写 1 清零 KBF。当任何使能的 KBI 引脚在声明电平时，向 KBACK 写 1 试图清除 KBF 无效，KBF 继续置 1。

8.4.3 KBI上拉/下拉电阻

使用相应的 I/O 端口上拉允许寄存器，KBI 引脚可以被设置来包含一个内部上拉/下拉

电阻。如果一个内部电阻被允许，则 KBIES 寄存器就被用来选择电阻为上拉电阻（KBEDGn=0）或下拉电阻（KBEDGn=1）。

8.4.4 KBI初始化

当一个键盘中断引脚开始被允许时，可能得到一个错误的键盘中断标志位。为了防止在键盘初始化期间发生错误的中断请求，用户必须照如下做：

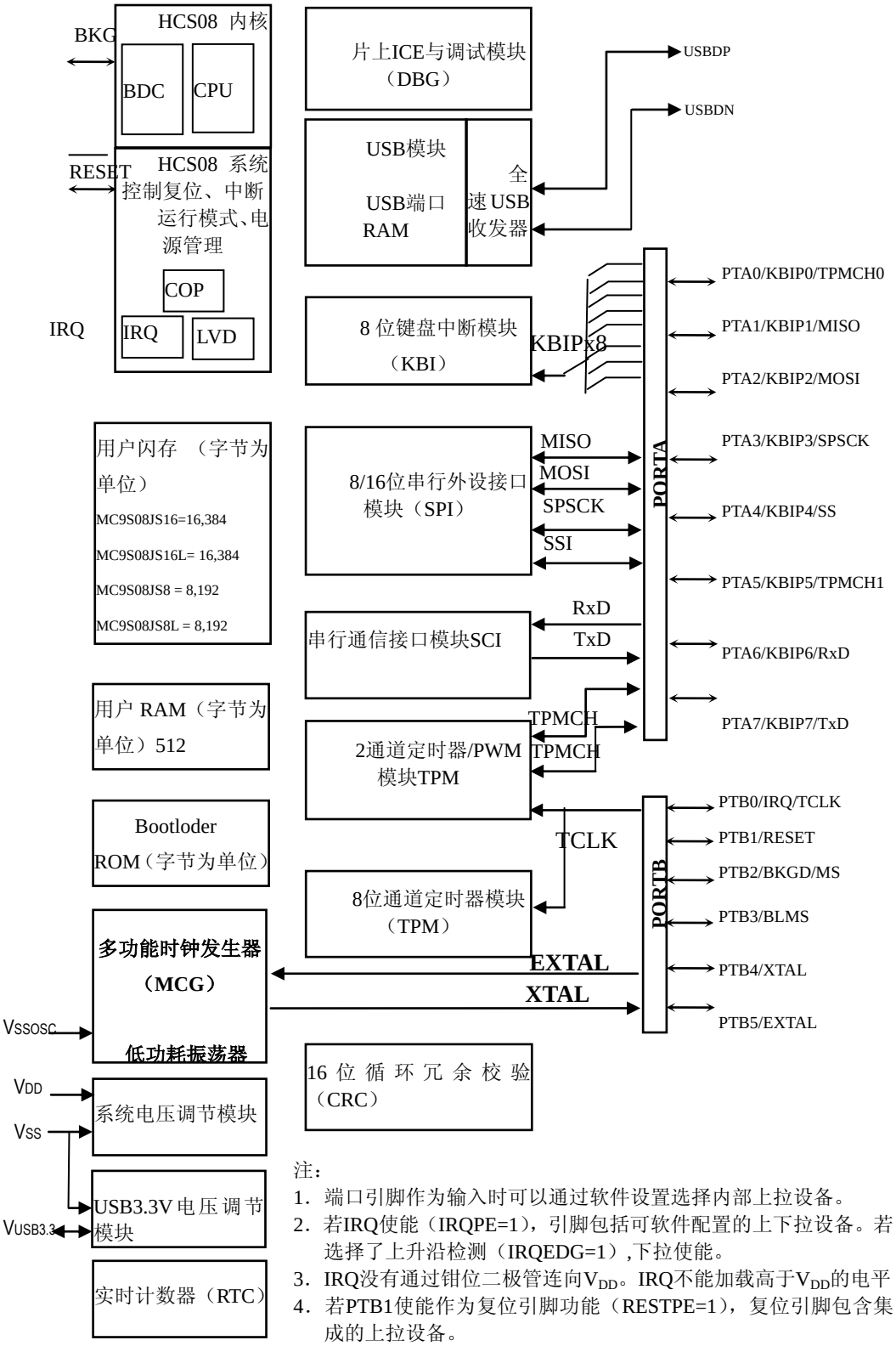
1. 将 KBISC 中的 KBIE 位清零屏蔽键盘中断。
2. 将 KBIES 中相应 KBEDGn 置位来允许 KBI 的极性。
3. 如果使用内部上拉/下拉设备，设置 PTxPE 中相应上拉允许位。
4. 将 KBIPE 中相应 KBIPEn 位置位来允许 KBI 引脚。
5. 向 KBISC 中 KBACK 写入来清除错误的中断。
6. 将 KBISC 中 KBIE 置位来允许中断发生。

第9章　多功能时钟发生器（S08MCGV1）

9.1 简介

多功能时钟发生器（MCG）模块提供了多种微控制器时钟源选择。其中包含一个频率锁定环（FLL）和锁相环（PLL）电路。该模块可以选择 FLL 或 PLL 的时钟，或者是内部或外部参考时钟作为单片机系统时钟。无论选择哪个时钟源，它都要经过能够分频一个较低输出频率时钟的总线分频器。在 MCG 还控制由晶体和振荡器组成的外部晶体振荡器（XOSC）来自作为外部参考时钟。

对于 MC9S08JS16 系列的 USB 操作，MCG 的必须配置为锁相环外部占用模式，以实现 MCGOUT 频率达到 48MHz。



注：

1. 端口引脚作为输入时可以通过软件设置选择内部上拉设备。
2. 若IRQ使能（IRQPE=1），引脚包括可软件配置的上下拉设备。若选择了上升沿检测（IRQEDG=1）,下拉使能。
3. IRQ没有通过钳位二极管连向V_{DD}。IRQ不能加载高于V_{DD}的电平
4. 若PTB1使能作为复位引脚功能（RESTPE=1），复位引脚包含集成的上拉设备。
5. 引脚包含集成上拉设备。
6. 当KBI使能（KBIPEn=1）而且相关引脚被配置为上拉设备使能，那么上升沿（KBEDGn=1）被用于将上拉设备配置为下拉设备。

图 9-1. MC9S08JS16系列框图MCG模块和引脚高亮显示

9.1.1 特点

MCG 的模块的主要特点是：

频率锁定环（FLL）

- 0.2％解析，利用内部 32 千赫参考
- 2％过压和温度偏差，利用内部 32 千赫参考

-内部或外部参考可用于控制 FLL

锁相环路（PLL）

-电压控制振荡器（VCO）

-模的 VCO 分频器

-相位/频率检测器

-综合环路滤波器

-有中断能力的锁定检测器

内部参考时钟

-九位修正准确性

-可选定为 MCU 的时钟源

外部参考时钟

-控制外部振荡器

-带复位功能的时钟监测

-可选定为 MCU 的时钟源

提供参考分频器

选择的时钟源可以被 1，2，4，或 8 分频

BDC 时钟（MCGLCLK）被看作一个 DCO 输出中 2 个的持续分频，不论是在 FLL 还是 PLL 模式。

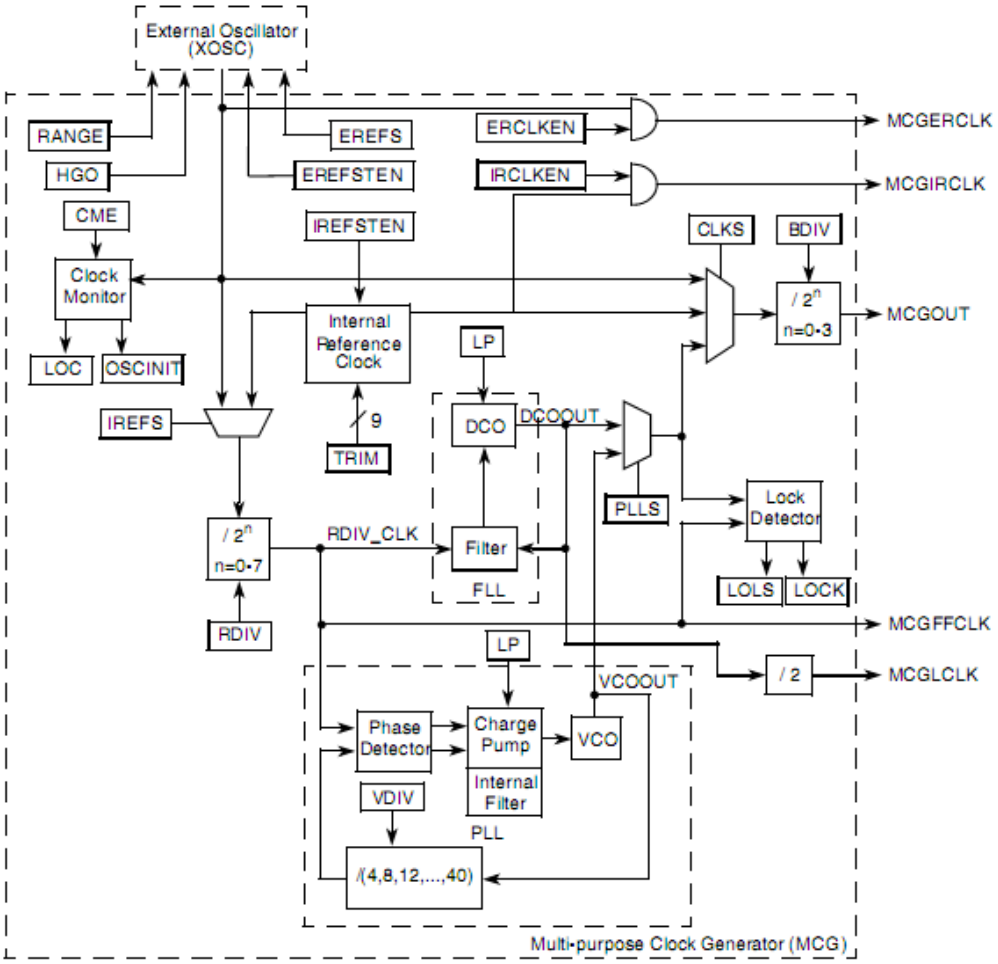


图9-2. 多用途时钟产生器（MCG）框图

9.1.2 运行模式

MCG 总共有 9 个允许模式

- FLL 内部占用（FEI）
- FLL 外部占用（FEE）
- FLL 内部忽略（FBI）
- FLL 外部忽略（FBE）
- PLL 外部占用（PEE）
- PLL 外部忽略（PBE）
- 内部忽略第功耗（BLPI）
- 外部忽略低功耗（BLPE）
- 停止

更多细节请参考 9.4.1 节：操作模式。

9.2 外部信号描述

在芯片之外没有 MCG 信号。

9.3 寄存器定义

9.3.1 MCG控制寄存器1（MCGC1）

	7	6	!	4	3	2	1	0
R W	CLKS		RDIV			IREFS	IRCLKEN	IREFSTEN
复位	0	0	(	0	0	0	0	0

图9-3.MCG控制寄存器1（MCGC1）

表9-1.MCG控制寄存器1位域描述

位域	描述
7:6	**时钟源选择** – 选择系统时钟源 00 编码 0 – FLL 或者 PLL 被选择 01 编码 1 – 内部参考时钟被选择 10 编码 2 – 外部参考时钟被选择 11 编码 3 – 保留，默认为 00
5:3	**参考分频因子** – 选择由 IREFS 位确定的参考时钟的分频因子。如果 FLL 被选择，结果频率必须在 31.25kHz 到 39.0625kHz 范围之内内。如果 PLL 被选择，结果频率必须在 1MHz 到 2MHz 之间。 000 编码 0 – 参考时钟 1 分频（复位默认） 001 编码 1 – 参考时钟 2 分频 010 编码 2 – 参考时钟 4 分频 011 编码 3 – 参考时钟 8 分频 100 编码 4 – 参考时钟 16 分频 101 编码 5 – 参考时钟 32 分频 110 编码 6 – 参考时钟 64 分频 111 编码 6 – 参考时钟 128 分频
2 IREFS	**内部参考电源选择** 1 选择内部参考时钟 0 选择外部参考时钟
1 IRCLKEN	**内部参考时钟使能 – 使能内部参考时钟用作 MCGIRCLK** 1 MCGIRCLK 激活 0 MCGIRCLK 禁止
0 IREFSTEN	**内部参考停止使能** – 控制当 MCG 进入停止模式的时候是否选择内部参考时钟保持使能。 1 如果在进入停止模式之前 IRCLKEN 被设置或者是 MCG 在 FEI,FBI,或者是 BLPI 模式，在停止模式中内部参考时钟保持使能， 0 内部参考时钟在停止模式中无效

9.3.2 MCG控制寄存器2（MCGC2）

	7	6	5	4	3	2	1	0
R W	BDIV		RANGE	HGO	LP	EREFS	ERCLKEN	EREFSTEN
复位:	0	0	0	0	0	0	0	0

图9-4 MCG控制寄存器2（MCGC2）

表9-2.MCG控制寄存器2位域描述

位域	描述
7:6 BDIV	**总线频率分频因子** – 选择由 MCGC1 寄存器中 CLKS 位决定的时钟源的分频。这控制总线频率。 00 编码 0 – 时钟 1 分频 01 编码 1– 时钟 2 分频（复位后默认） 10 编码 2– 时钟 4 分频 11 编码 3 – 时钟 8 分频
5 RANGE	**频率范围选择** – 选择外部振荡器或者外部时钟源的频率范围。 1 选择 1MHz 到 16MHz 外部振荡器的频率范围。（1MHz 到 40MHz 的外部时钟电源）的高频率范围 0 选择 32kHz 到 100kHz 外部振荡器的频率范围。（32kHz 到 1MHz 的外部时钟电源）的低频率范围
4 HGO	**高增益振荡器选择** – 控制外部振荡器操作模式。 1 配置外部振荡器为高增益运行 0 配置外部振荡器为低功耗运行
3 LP	**低功耗选择** – 控制在忽略模式中 FLL（或者 PLL）是否为无效 1 FLL（或 PLL）在忽略模式（低功耗）中为无效的。 0 FLL（或 PLL）在忽略模式中为无效的。
2 EREFS	**外部参考时钟选择** – 为外部参考选择时钟源 1 选择振荡器 0 选择外部时钟源
1 ERCLKEN	**外部参考时钟使能** – 使能外部参考时钟作为 MCGERCLK 1 MCGERCLK 激活 0MCGERCLK 无效
0 EREFSTEN	**外部参考时钟停止使能**-当 MCG 进入停止模式时控制外部参考时钟是否保持有效。 1 如果 ERCLKEN 被设置或者是 MCG 处于 FEE，FBE，PEE，PBE，或者是 BLPE 模式中而没有进入停止模式是外部参考时钟保持有效。 0 外部参考时钟在停止模式下无效。

9.3.3 MCG修正寄存器

	7	6	5	4	3	2	1	0
R	TRIM							

W								
复位:	0	0	0	0	0	0	0	0
POR	1	0	0	0	0	0	0	0

图9-5 MCG修正寄存器（MCGTRM）

表9-3.MCG修正寄存器位域分析

位域	描述
7:0 TRIM	**MCG 校正设置 –** 通过内部参考时钟周期来控制内部参考时钟频率。TRIM 位是二进制(即对位 1 的调整是为 0 调整的两倍)。增加 TRIM 二进制值将增加周期，减少值将减少周期。MCGSC 寄存器额外的校正位和 FTRIM 为一样有效。 如果 TRIM[7:0]值存储在非易失性存储器中，用户应该拷贝值到寄存器中。

9.3.4 MCG状态和控制寄存器（MCGSC）

	7	6	5	4	3	2	1	0
R	LOLS	LOCK	PLLST	IREFST	CLKST		OSCINIT	FTRIM
W								
复位:	0	0	0	0	0	0	0	0

图9-6 MCG状态和控制寄存器（MCGSC）

表9-4 MCG状态和控制寄存器位域分析

位域	描述
7 LOLS	**丢失锁定状态 –** 此位标志了 FLL 或者是 PLL 的锁定状态。当在要求锁定之后锁定使能时 LOLS 被置位，FLL 或者 PLL 输出频率降低解锁超出了频率范围。D_{unl} LOLIE 决定当通过复位或者是写逻辑以到 LOLS 将 LOLS 清除时是否中断响应。想 LOLS 写 0 没有影响。
6 LOCK	**锁定状态 –** 指示 FLL 或者是 PLL 要求锁定。当 FLL 和 PLL 无效时锁定检测无效。当改变以下任何位时锁定状态位被置位：IREFS，PLLS，RDIV[2:0],TRIM[7:0（如果处于 FEI 或者 FBI 模式中），或者是 VDIV[3:0]（如果处于 PBE 或者 PEE 模式中），将引起锁定状态为清除并且保持直到 FLL 或者 PLL 要求锁定。进入停止模式同样能引起 BCK 状态位清除并保持到 FLL 或者 PLL 要求锁定。进入 BLPI 或者 BLPE 模式将同样导致锁定位清除并保持直到 MCG 退出这些模式并且 FLL 或者 PLL 要求锁定。 0 FLL 或者 PLL 当前未锁定 1 FLL 或者 PLL 当前锁定
5 PLLST	**PLL 选择状态 –** PLLST 位指示了 PLLS 时钟的当前时钟源。当向 PLLS 位写操作时 PLLST 位不会立即更新。 0 FLL 时钟作为 PLLS 的时钟源 1 PLL 时钟作为 PLLS 的时钟源
4 REFST	**内部参考状态 –** IREFST 位指示了参考时钟的当前时钟源。在向 IREFS 位写后，IREFST 不会立即更新。 0 参考时钟源是外部参考时钟（由 MCGC2 寄存器中 EREFS 位决定的振荡器或者是外部时钟源）

	1 参考时钟源是内部参考时钟
3:2 CLKST	**时钟模式状态** – CLKST 位指示了当前时钟模式。在想 CLKS 位写入后，CLKST 位不会立即更新。 00 编码 0 – FLL 的输出被选择 01 编码 1 – 内部参考时钟被选择 10 编码 2 – 外部参考时钟被选择 11 编码 3 – PLL 的输出被选择
1 SCINIT	**OSC 初始化** – 如果外部参考时钟被选择（通过置位 ERCLKEN 或者通过 MCG 处于 FEE，FBE，PEE，PBE,BLPE 模式中），如果 EREFS 被置位，在外部振荡器时钟完成的初始化周期 EREFS 将被置位。此位只有当 EREFS 被清除或者是当 MCG 在 FEI，FBI，BLPI 模式中并且 ERCLKEN 被清除的时候才会被清除。
0 FTRIM	**MCG 精密校正** – 控制内部参考时钟频率的最小的校正。设置 FTRIM 将增加周期，清除 FTRIM 将会降低周期。 如果 FTRIM 值被存储在非易失性的存储器中，用户应该将其拷贝到寄存器的 FTRIM 位。

9.3.5 MCG控制寄存器3（MCGC3）

	7	6	5	4	3	2	1	0
R	LOLIE		PLLS	CME	0	VDIV		
W								
Reset:	0	0	0	0	0	0	0	0

图9-7 MCG 锁相环寄存器（MCGPLL）

表9-5 MCG锁相环寄存器位域描述

位域	描述
7 LOLIE	**锁定丢失中断使能** – 决定当出现丢失锁定时是否产生一个中断。LOLIE 位只有当 LOLS 被设置时才有作用 0 不响应丢失锁定中断 1 产生丢失响应中断
6 PLLS	**PPL 选择** – 控制 PLL 还是 FLL 被选择。如果 PLLS 位清除，PLL 在所有模式中无效。如果 PLLS 被设置，FLL 在所有模式中无效。 1 PLL 被选择 0 FLL 被选择
5 CME	**时钟监视器使能** – 决定当丢失外部时钟出现时是否产生复位请求。无论当 MCG 使用外部时钟并处于操作模式（FEE,FBE,PEE,PBE 或者 BLPE）还是外部参考时钟有效（ERCLKEN=1 在 MCGC2 寄存器中），MCE 位只能被设置为逻辑 1.无论什么时候 CME 位被设置为逻辑 1，MCGC2 寄存器中 RANGE 位的值不应该被改变。 0 时钟监视器被禁止 1 当丢失外部时钟时产生一个复位请求
3:0 VDIV	**VCO 分频因子** – 选择 PLL 的输出 VCO 的分频因子。VDIV 位确定参考时钟频率的倍频因子（M） 0000 编码 0 -保留 0001 编码 1 -4 倍频

	0010 编码 2 – 8 倍频 0011 编码 3 - 12 倍频 0100 编码 4 -16 倍频 0101 编码 5 – 20 倍频 0110 编码 6 – 24 倍频 0111 编码 7 – 28 倍频 1000 编码 8 – 32 倍频 1001 编码 9 – 36 倍频 1010 编码 10 – 40 倍频 1011 编码 11 – 保留（默认 M=40） 11xx 编码 12-15 –保留（默认 M=40）

9.4 功能描述

9.4.1 操作模式

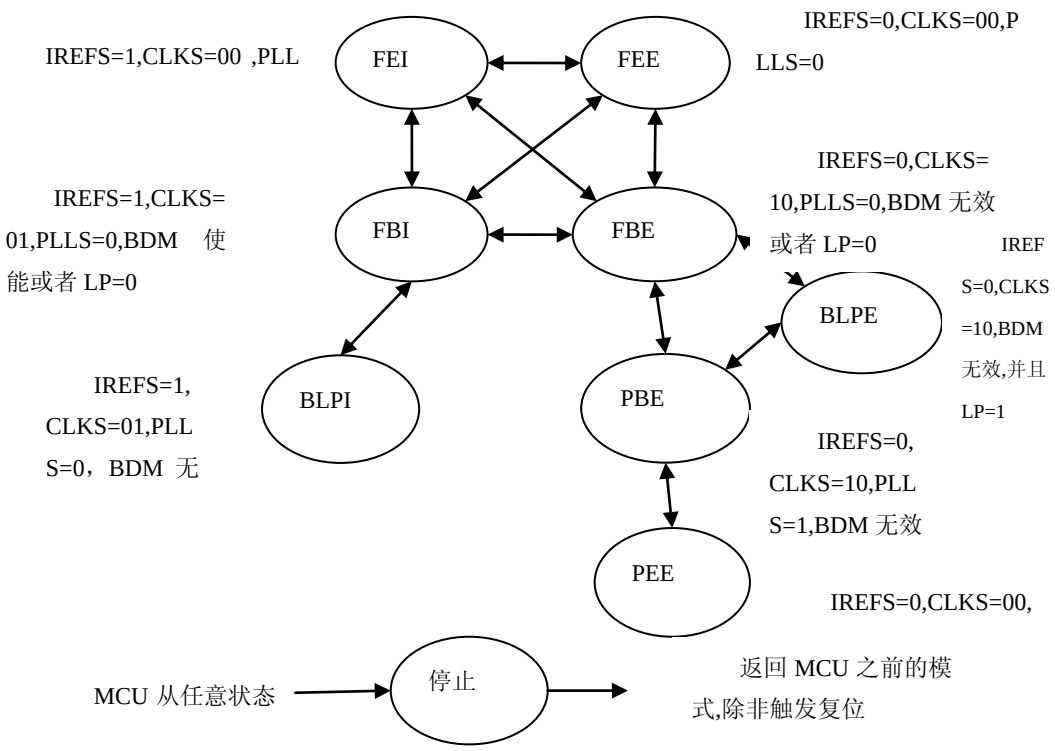


图9-8 时钟转换模式

MCG 的第九中状态被显示为一个状态图并且描述如下。箭头表示出两个状态之间的允许的动作

9.4.1.1 FLL内部占用（FEI）

FLL 内部占用模式是运行的默认模式并且当以下条件条件满足时进入。

- CLKS 位被写 00
- IREFS 位被写 1
- PLLS 位被写 0
- RDIV 位被写 000。既然在被修正后内部参考时钟频率已经在 31.25kHz 到 39.625 范围之内，那么就不需要继续分频。

在 FLL 内部占用模式，MCGOUT 时钟来自由内部参考时钟控制的 FLL 时钟。FLL 时钟频率被锁定为参考频率的 1024 倍，由 RDIV 位选择。MCGLCLK 来自 FLL，并且 PLL 在低功耗状态下无效。

9.4.1.2 FLL外部占用（FEE）

进入 FLL 外部占用模式必须满足以下条件。

- CLKS 位被写 00
- IREFS 位被写 0
- PLLS 位被写 0
- RDIV 各位用来设置将参考时钟分频为从 31.25KHz 到 39.0625KHz 范围之内。

FLL 外部占用模式，MCGOUT 时钟来自受外部参考时钟控制的 FLL 时钟。外部参考时钟可以是晶振也可能是另外一个外部时钟源。FLL 时钟频率被锁定为参考频率的 1024 倍，由 PDIV 各位选择。MCGLCK 来自 FLL 并且 PLL 在低功耗状态下无效。

9.4.1.3 FLL内部忽略（FBI）

在 FLL 内部忽略模式之下，MCGOUT 时钟来自内部参考时钟并且 FLL 是可操作的但是它的输出时钟未被使用。这个模式对当 MCGOUT 时钟来自内部参考时钟时允许 FLL 变为要求的频率是有用的。

当满足以下条件时，进入 FLL 内部忽略模式：

- CLKS 位写 01
- IREFS 位写 1
- PLLS 位写 0
- RDIV 各位写 000，因为当被修正后内部参考时钟频率已经在 31.25kHz 到 39.0625 范围之内，那么就不需要继续分频。
- LP 位写 0

在 FLL 内部忽略模式之下，MCGOUT 时钟来自内部参考时钟。PLL 时钟由内部参考时钟控制，FLL 时钟频率被锁定为参考频率的 1024 倍，由 RDIV 各位来选择。MCGLCLK 来自 FLL 并且 PLL 在低功耗状态下无效。

9.4.1.4 FLL外部忽略（FBE）

在 FLL 外部忽略模式之下，MCGOUT 时钟来自外部参考时钟，并且 FLL 是可运行的当时它的输出时钟没有用。这个模式对当 MCGOUT 时钟来自外部参考时钟时而允许 FLL 达到目的频率的情况是有用的。

FLL 外部忽略模式只有当以下条件满足是才能进入：

- CLKS 位写 10
- IREFS 位写 0
- PLLS 位写 0
- RDIV 各位是用来将参考时钟分频为 31.25kHz 到 39.0625kHz 范围之内。

- LP 位写 0

在 FLL 外部忽略模式之下，MCGOUT 时钟来自外部参考时钟。外部参考时钟可以是外部晶振或者是另外一个外部时钟源。FLL 时钟由外部参考时钟控制，并且 FLL 时钟频率被锁定为参考频率的 1024 倍，由 RDIV 各位所设定。MCGLCLK 来自 FLL，并且 PLL 在低功耗状态下无效。

注意：当参考时钟频率高于指定最高频率时，可能要暂时地工作在 FBE 模式。在 PEE 模式之下操作使用频率大于 5MHz 的外部晶振，这在应用中是必须的。详细例子请参考 9.5.2.4　例#4：从 FEI 到 PEE 模式：外部晶振=8MHz，总线频率=8MHz。

9.4.1.5 PLL外部占用（PEE）

进入外部 PLL 占用必须满足以下条件：

- CLKS 位写 00
- IREFS 位写 0
- PLLS 位写 1
- RDIV 各位用来将参考时钟分频在 1MHz 到 2MHz 范围之内

在 PLL 外部占用模式，MCGOUT 时钟来自受外部参考时钟控制的 PLL 时钟。外部参考时钟可以是外部晶振也可以是另外一个外部时钟源。PLL 时钟频率可被锁定由倍数因子（由 VDIV 各位来选择），倍数和参考频率（由 RDIV 各位来选择）。如果 BDM 开启，那么 MCGLCLK 来自 DCO（开环模式）的二分频。如果 BDM 未开启，那么 FLL 在低功耗状态下无效。

9.4.1.6 PLL外部忽略（PBE）

在 PLL 外部忽略模式之下，MCGOUT 时钟来自外部参考时钟，并且 FLL 是可运行的但是它的输出时钟没有用。这个模式对当 MCGOUT 时钟来自外部参考时钟时而允许 FLL 达到目的频率的情况是有用的。

PLL 外部忽略模式只有当以下条件满足是才能进入：

- CLKS 位写 10
- IREFS 位写 0
- PLLS 位写 1
- RDIV 各位是用来将参考时钟分频为 1KHz 到 2KHz 范围之内。
- LP 位写 0

在 PLL 外部忽略模式，MCGOUT 时钟来自受外部参考时钟控制的 PLL 时钟。外部参考时钟可以是外部晶振也可以是另外一个外部时钟源。PLL 时钟频率可被锁定由倍数因子（由 VDIV 各位来选择），倍数和参考频率（由 RDIV 各位来选择）。如果 BDM 开启，那么 MCGLCLK 来自 DCO（开环模式）的二分频。如果 BDM 未开启，那么 FLL 在低功耗状态下无效。

9.4.1.7 内部忽略低功耗（BLPI）

BLPI 模式只有当以下条件满足是才能进入：

- CLKS 位写 10
- IREFS 位写 0
- PLLS 位写 0 或 1
- LP 位写 1
- BDM 模式未被激活

在 BLPI 模式，MCGOUT 时钟来自受外部参考时钟控制的 PLL 时钟。外部参考时钟可以是外部晶振也可以是另外一个外部时钟源。

PLL 和 FLL 在 BLPE 模式之下是无效的，并且 MCGLCLK 对于 BDC 通讯来说是无效的。如果 BDM 变成有效，本模式将转换为外部忽略模式中的一个（由 PLLS 位的状态来决定）。

9.4.1.9 停止

只要 MCU 进入 STOP 模式，系统处于 STOP 模式。在本模式中，除了以下的情况，FLL 和 PLL 都禁止，并且 MCG 时钟信号是静态的。

当以下条件满足时，MCGIRCLK 将被激活在停止模式中。

- IRCLKEN=1
- IREFSTEN=1

当以下条件满足时，MCGERCLK 将被激活在停止模式中。

- ERCLKEN=1
- EREFSTEN=1

9.4.2 模式转换

当在内部占用模式和外部占用模式之间切换时，IREFS 位是随时可以改变的，但 RDIV 位必须同时改变，以便参考频率的停留由锁相环位状态所需范围（31.25 kHz 至 39.0625 千赫如果 FLL 被选中，或 1 MHz 至 2MHz 如果 PLL 被选中）。在 IREFS 值变化之后，当转换完成之后，FLL 或 PLL 将被重新锁定。开关的完成显示在 IREFST 位。

在转换到 FBE 模式之后立即进入停止模式这种特殊情况，如果外部时钟

和内部时钟在停止模式下是无效的（EREFSTEN = 0，IREFSTEN = 0），有必要在 IREFST 位被清除后延时 100us，以便内部参考时钟关闭。在于大多数情况下延时时间足够了。

该 CLKS 位也可以随时改变，但为了使 MCGLCLK 正确配置，RDIV 位必须同时改变，以便参考频率保持在由 PLLS 位状态要求的频率范围之内（31.25 KHz 至 39.062KHz 如果 FLL 被选择，或 1MHz 至 2MHz 如果 PLL 被选中）。实际切换到新选定的时钟将由 CLKST 位显示。如果新选定的时钟不可用，以前的时钟将继续被选中。

详情参见图 9-8。

9.4.3 总线频率分频器

BDIV 各位能够随时改变，选择的新频率将立即生效。

9.4.4 低功耗位的使用

低功耗位（LP）的用来使 FLL 或 PLL 被禁用，从而节省能源当这些系统没有被使用时。但是，在某些应用中可能需要启用 FLL 或 PLL，并允许它在切换到一个参与模式之前最准确的锁定。向 LP 位写 0 可以达到这个目的。

9.4.5 内部参考时钟

当 IRCLKEN 位被置 1 时，内部参考时钟信号将被作为 MCGIRCLK，并且可以作为一种额外的时钟源。该 MCGIRCLK 频率可以重新指定，通过修正内部参考时钟的周期。这可以通过在 MCGTRM 寄存器中的 TRIM 位写入新值得到。写一个较大的值将减少 MCGIRCLK 频率，写一个较小的值到 MCGTRM 寄存器会增加 MCGIRCLK 频率。如果 MCG 处于 FEI 模式或者 FBI 模式或者 BLPI 模式之下，TRIM 位将影响 MCGOUT 的频率。TRIM 和 FTRIM

的值将由 POR 初始化，但不受其它的设置影响。

在 MCGIRCLK 被修正前，写入低参考分频因子可能导致 MCGOUT 频率超出芯片级频率的最高值这是违反时钟修正规范的（参加器件概述章节）。

如果 IREFSTEN 和 IRCLKEN 位都被设置，内部参考时钟将在停止模式下继续允许，以便能迅速的从停止模式当中恢复。

9.4.6 外部参考时钟

在 FEE 模式和 FBE 模式之下，MCG 模块可以支持 31.25KHz 到 5 MHz 范围内的外部参考时钟 ，在 PEE 模式和 PBE 模式当中支持的范围为 1MHz 到 16MHz，在 BLPE 模式当中支持的范围为 0 到 40MHz。当 ERCLKEN 被置高，、外部参考时钟信号将作为 MCGERCLK，可作为额外的时钟源。当 IREFS = 1，外部参考时钟将不能被 FLL 或 PLL 使用，并且只能被当作 MCGERCLK。在这些模式下，频率可以等于芯片级时序规格支持的最高频率（见器件概述章）。

如果 EREFSTEN 和 ERCLKEN 位都设置或 MCG 是在 FEE，FBE，PEEPBE 或者 BLPE 模式之下，那么外部参考时钟将在停止模式下继续运行，以便能够从停止模式中快速恢复。

如果 CME 位写入 1，时钟监视器启用。如果外部参考时钟低于某一频率（f_{loc_high} 或 f_{loc_low} 取决于 MCGC2 寄存器中的 RANGE 位），MCU 将复位。SRS 寄存器中的 LOC 位将被置位来指示错误。

9.4.7 混合频率时钟

MCG 将被分频的参考时钟作为 MCGFFCLK 用作一个额外的时钟源。MCGFFCLK 频率必须低于 MCGOUT 有效频率的 1/4.因为这个要求，MCGFFCLK 在忽略模式中是无效的，由 BDIV 和 RDIV 的值共同决定：

- BDIV = 00（1 分频）,RDIV<010
- BDIV = 01（2 分频）,RDIV < 011 当 MCGFFCLK 为有效时，MCGFFCLKVALID 被置为 1.否则被置为 0。

9.5 初始化/应用信息

本节描述了在应用中如何初始化和配置 MCG 模块。接下来的节包含了如何初始化 MCG 和在各种有效状态下作合适转换的例程。

9.5.1 MCG模块初始化序列

MCG 由 FEI 模式中将 BDIV 设置为 2 分频而出。在 FLL 获得锁定之前，内部参考将保持稳定在 t_{irefst} 微秒内。只要内部参考是稳定的，FLL 将获得锁定在 t_{fll_lock} 毫秒内。

至于 POR，内部参考频率将要求校正来产生一个准确的时钟。飞思卡尔推荐使用 FLASH 中 0xFFAE 来存储精确的校正位，MCGSC 寄存器中 FTRIM，和 0xFFAF 来存储 8 位校正 MCGTRM 寄存器中的数据。MCU 将不会自动的将这些值拷贝到各自的寄存器。因此，用户必须自己将这些数据从 FLASH 拷贝到寄存器中。

注意：在内部参考时钟被校正前，BDIV 值不能被设置为 1 分频。这样的操作是不符合 MCU 的操作规格的。

9.5.1.1 MCG的初始化

因为 MCG 从 FEI 模式的复位中出来，能够直接转换重置的模式是 FEE，FBE 和 FBI 模式（见图 12-8）。到达任何其他方式首先配置这三个初步的 MCG 的模式之一。必须注意核实有关情况，在 MCGSC 位寄存器反映在每个模式下所有配置的更改。

要改变模式，从 FEI 模式到 FEE 或 FBE 模式，请按照此过程：

1. 通过设置 MCGC2 相应位来启用外部时钟源。
2. 设置 MCGC1 选择时钟模式。

-如果进入 FEE 模式，适当设置 RDIV，清除 IREFS 位切换到外部参考，而离开 CLKS 位 00％，所以，FLL 的输出被选择用来作为系统时钟来源。

-如果进入了 FBE 模式，清除 IREFS 位切换到外部参考，改变 CLKS 到 10％，使外部参考时钟作为系统时钟源。那个 RDIV 位也应适当地设置根据外部参考频率，因为虽然是绕过的 FLL，但仍处于 FBE 模式。

-内部参考可以设置 IRCLKEN 位来有选择地保持运行。如果应用要在内部和外部模式之间来回转换，这是非常有用的。为了达到最小的功耗，就要使得内部产靠时钟无效当处于外部时钟模式的时候。

3.　在设置好相关的位之后，等待 MCGSC 寄存器中相关的位改变，说明 MCG 已经转换到了合适的模式。

-如果 ERCLKEN 在 1 中已经设置好了，或者是 MCG 已经处于 FEE,FBE,PEE,PBE,BLPE 模式之中，并且 EREFS 同样在 1 中已经设置好了，等待 OSCINIT 位被设置，说明外部时钟源已经完成了初始化周期并且稳定了下来。典型的周期设置次数在附录 A 中:“电器特性”。

-如果处于 FEE 模式当中，确保在继续之前清除 IREFST 位。

-如果在 FBE 模式中，确保清除 IREFST 位，设置 LOCK 位，CLKST 位被改为%10，说明外部参考时钟被合适的选择。虽然在 FBE 模式中是忽略 FLL 的，但是它仍然开启并且锁定在 FBE 模式之中。

从 FEI 时钟模式转换到 FBI 时钟模式，请遵循以下的步骤：

1. 改变 CLKS 位到%01 以便内部参考时钟被选为系统时钟源。
2. 等待 MCGSC 寄存器中的 CLKST 位改变到%01，表明内部参考时钟被合适的选择。

9.5.2 MGC模式转换

当在 MCG 可运行模式之间转换，默写设置为必定被改变。每次任何位改变的时候（PLLS，IREFS,CLKS,或者是 EREFS），在 MCGSC 寄存器中相关的位（PLLST，IREFST,CLKST,或者是 OSCINIT）在应用程序继续之前必须被检查。

另外，必须小心的确保参考时钟分频因子（RDIV）是合适的。例如，在 PEE 模式中，如果使用 4MHz 的晶振，RDIV 必须被设置为%001（2 分频）或者是%010（4 分频），是为了将外部参考时钟降至要求的 1MHz 到 2MHz。

在改变 PLLS 位之前 RDIV 和 IREFS 位总是要被合适的设置。以便 FLL 或者 PLL 时钟有一个合适的参考时钟来转换。

下表显示了 MCGOUT 频率计算，通过设置 RDIV,BDIV,和 VDIV，在每个时钟模式中。总线频率等于 MCGOUT 的 2 分频。

表12-6.MCGOUT频率计算选项

时钟模式	f_{MCGOUT}	说明
FEI（FLL 内部占用）	（f_{int} *1024）/B	典型f_{MCGOUT} =16MHz RDIV 位设置为%000

FEE（FLL 外部占用）	（f_{ext}/R*1024）/B	f_{ext}/R 必须在 31.25kHz 到 39.0625kHz 范围之内
FBE（FLL 外部忽略）	f_{ext}/B	f_{ext}/R 必须在 31.25kHz 到 39.0625kHz 范围之内
FBI（FLL 内部忽略）	f_{int}/B	典型f_{ext}=32kHz
PEE（PLL 外部占用）	[（$\frac{f_{ext}}{R}$）*M]/B	f_{ext}/R 必须在 1MHz 到 2MHz 范围之内
PBE（PLL 外部忽略）	f_{ext}/B	f_{ext}/R 必须在 1MHz 到 2MHz 范围之内
BLPI（内部第电压忽略）	f_{int}/B	
BLPE（外部低电压忽略）	f_{ext}/B	

本节将包含 3 个模式转换的例子使用 4MHz 的外部晶振。如果使用的外部时钟源小于 1MHz，MCG 则不应该被设置为 PLL 模式（PEE 和 PBE）。

9.5.2.1 例#1：从FEI到PEE模式：外部晶振=4MHz，总线频率=8MHz

本例中，在竞争参考频率被设置达到总线频率 8MHz 之后，MCG 将从 FEI 模式转换为 PEE 模式。因为 MCG 在 FEI 模式之中脱离复位。本例将显示如何初始化 MCG 的 PEE 模式脱离复位。首先，代码序列将被描述。然后有个流程图将被用来说明序列。

1. 首先，FEI 必须转换到 FBE 模式：
   a) MCGC2 = 0x36（%00110110）
      - BDIV（第 7 和第 6 位）设置为%00，或者是 1 分频
      - RANGE（第 5 位）设置为 1，因为 4MHz 的频率位于高频率范围之内
      - HGO（第 4 位）设置为 1，来设置外部振荡器高速运转
      - EREFS（第 2 位）设置为 1，因为使用了晶振
      - ERCLKEN（第 1 位）设置为 1，用来保证外部参考时钟是激活的

   b） 循环直到 MCGSC 中 OSCINIT（第 1 位）为 1，说明由 EREFS 位选择的晶振被初始化了。

   c） MCGC1 = 0xB8（%10111000）

   - CLKS（第 6，7 位）设置为%10，用来选择外部参考时钟作为系统时钟源

   - RDIV（第 5-3 位）设置为%111，即被 128 分频，因为 4MHz/123=21.25kHz，这在 FLL 要求的 31.25kHz 到 39.0625kHz 的范围之内。

   - IREFS（第 2 位）清零，选择外部参考时钟

   d） 循环直到 MCGSC 中的 IREFST（第 4 位）位 1,指示外部参考时钟作为当前的参考时钟

   e） 循环知道 MCGSC 中的 CLKST（第 3，2 位）为%10，指示外部参考时钟选择作为 MCGOUT

2. 然后，FBE 必须转为 PBE 模式，或者是经过 BLPE 模式再转为 PBE 模式：

   a）BLPE:如果经过 BLPE 模式，首先要设置 MCGC2 中的 LP（第 3 位）为 1

   b）BLPE/PBE:MCGC1 = 0x90（%10010000）

   - RDIV（第 5-3 位）设置为%010，即 4 分频，因为 4MHz/4 = 1MHz，这在 PLL 要求的 1MHz 到 2MHz 之间。在 BLPE 模式 RDIV 的设置不重要。因为 FLL 和 PLL 都是无效的。改变他们只能设置 PLL 的分频因子，在模式 PBE 中。

   c） BLPE/PBE: MCGC3= 0x44（%01000100）

   - PLLS（第 6 位）设置为 1，选择 PLL。在 BLPE 模式中，改变这位只是让 MCG 准备在 PBE 模式中 PLL 的使用。

- VDIV（第 3-0 位） 设置为%0100，即 16 倍频，因为 1MHz*16 = 16MHz。在 BLPE 模式中，VDIV 的设置不重要。

d） BLPE:如果转换经过 PLPE 模式，清除 MCGC2 中的 LP（第 3 位），用来转换 PBE 模式。

e）循环直到 MCGSC 中的 PLLST（第 5 位）为 1，此时 PLL 作为当前 PLLS 的时钟源。

f）PBE:然后循环直到 MCGSC 中 LOCK（第 6 位）为 1，此时 PLL 被要求锁定。

3．最后，PBE 模式转换为 PEE 模式：

a） MCGC1 = 0x10（%00010000）

- MCGSC1 中 CLKS（第 7 和 6 位）设置为%00，用来选择 PLL 的输出作为系统时钟源。

- 循环直到 MCGSC 中 CLKST（第 3，2 位）位%11，说明 PLL 输出被选择来作为 MCGOUT 在当前的时钟模式中。

b)现在，RDIV 是 4 分频，BDIV 是 1 分频，VDIV 是 16 倍频，MCGOUT = [（4MHz/4）*16]=16MHz,总线频率是 MCGOUT/2，或者是 8MHz。

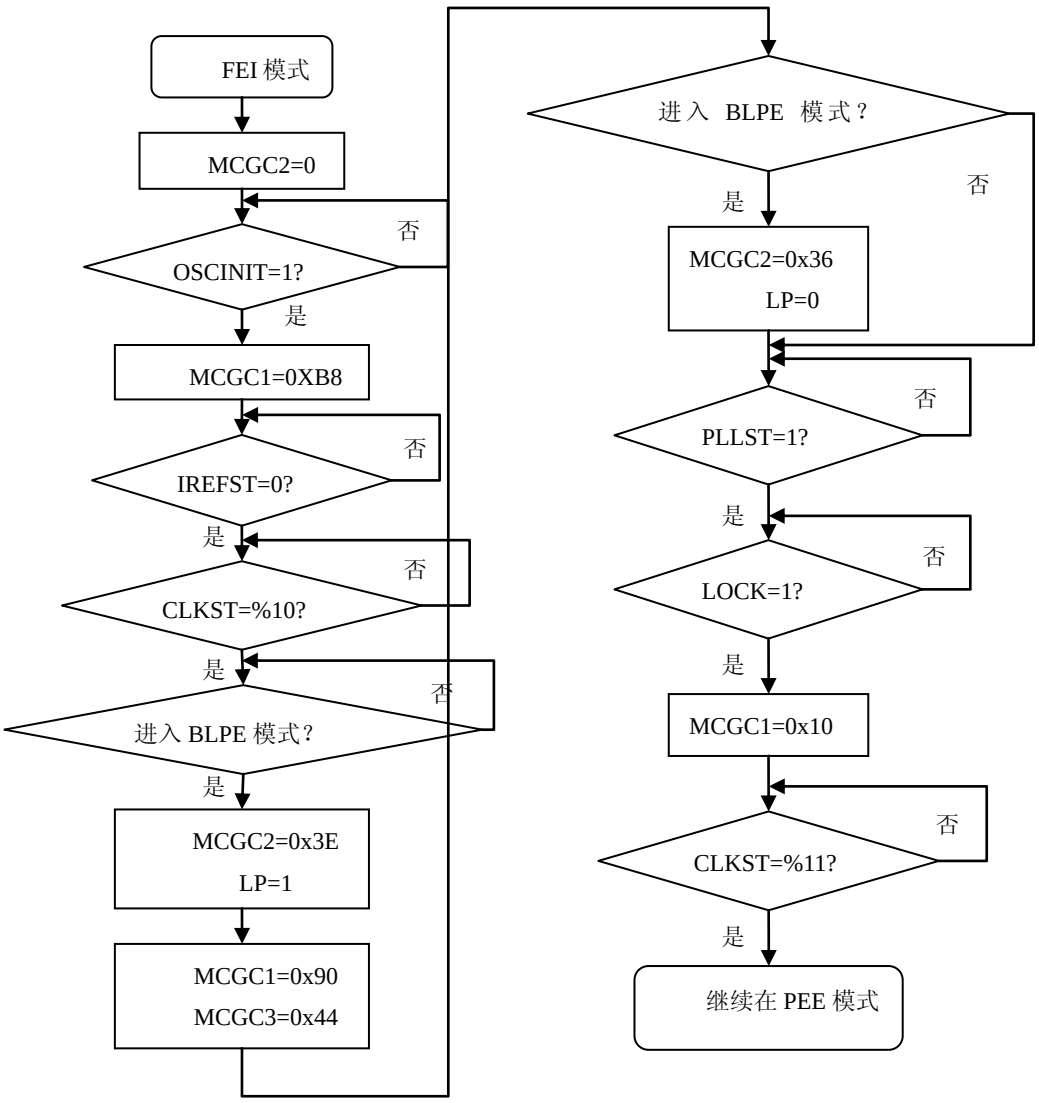


图9-9.4MHz晶振下FEI到PEE模式转换的流程图

9.5.2.2 例#2：从PEE模式到BLPI模式：外部晶振=4MHz,总线频率=16kHz

在本例中，MCG 将从 PEE 模式（4MHz 晶振配置为 8MHz 总线频率）到 BLPI 模式（16kHz 总线频率）。首先，代码序列将会被描述。然后将有流程图说明序列的含义。

1. 首先，PEE 必须转换为 PBE 模式：

   a） MCGC1=0x90（%10010000）

   -CLKS（第 7 和 6 位）被设置为%10，用来将系统时钟源转换为外部参考时钟。

   b）循环直到 MCGSC 的 CLKST（第 2 和 3 位）位%10，表示外部参考时钟作为 MCGOUT。

2. 然后，PBE 必须被直接转换为 FBE 模式或者经过 BLPE 模式然后转换为 PBE 模式：

   a）BLPE：如果从 BLPE 模式转换，首先在 MCGC2 寄存器中设置 LP（第 3 位）为 1.

   b）BLPE/FBE：MCGC1=0xB8（%10111000）

   -RDIV（第 5-3 位）设置为%111，即 128 分频，因为 4MHz/128=31.25kHz，在 31.25kHz 到 39.0626kHz 范围之间。满足 FLL 的要求。在 BLPE 模式，RDIV 的设置不重要，因为 FLL 和 PLL 是无效的。只有在 FBE 模式中为 FLL 使用设置分频器才能改变他们。

   c）BLPE/FBE：MCGC3=0x04（%00000100）

   -PLLS（第 6 位）清零用来选择 FLL，在 BLPE 模式中，只有在 FBE 模式中准备好 MCG 为了 FLL 的使用才需要改变这些位。当 PLLS=0 时，VDIV 值不重要。

   d）BLPE:如果经过 BLPE 模式转换，在 MCGC2 中清除 LP（第 3 位）来转换到 FBE 模式。

   e）FBE:循环知道 MCGSC 寄存器中的 PLLST（第 5 位）被清除，说明当前的 PLLS 时钟是 FLL。

   f）FBE:循环知道 MCGSC 寄存器中的 LOCK（第 6 位）被设置，说明 FLL 被要求锁定。虽然 FLL 在 FBE 模式中是忽略的，他仍然能够开启和运行。

3. 下一步，FBE 模式转换为 FBI 模式：

   a）MCGC1=0x44（%01000100）

   -MCGSC1 寄存器中 CLKS（第 7 和 6 位）设置为%01，用来将系统时钟转换为内部参考时钟。

   -IREFS（第 2 位）设置为 1 来选择内部参考时钟作为参考时钟源。

   -RDIV（第 5-3 位）设置为%000，即 1 分频，因为校正内部参考时钟应该在 31.25kHz 到 39.0625kHz 范围之内，z 这是由 FLL 要求的。

   b）循环直到 MCGSC 寄存器中 IREFST（第 4 位）为 1。表明内部参考时钟被选择作为参考时钟源。

   c）循环直到 MCGSC 寄存器中 CLKST（第 3 和 2 位）为%01，说明内部参考时钟被选择来作为 MCGOUT。

4. 最后，FBI 转换到 FBILP 模式

   a）MCGC2= 0x08（%00001000）

   -MCGSC 寄存器中 LP（第 3 位）为 1

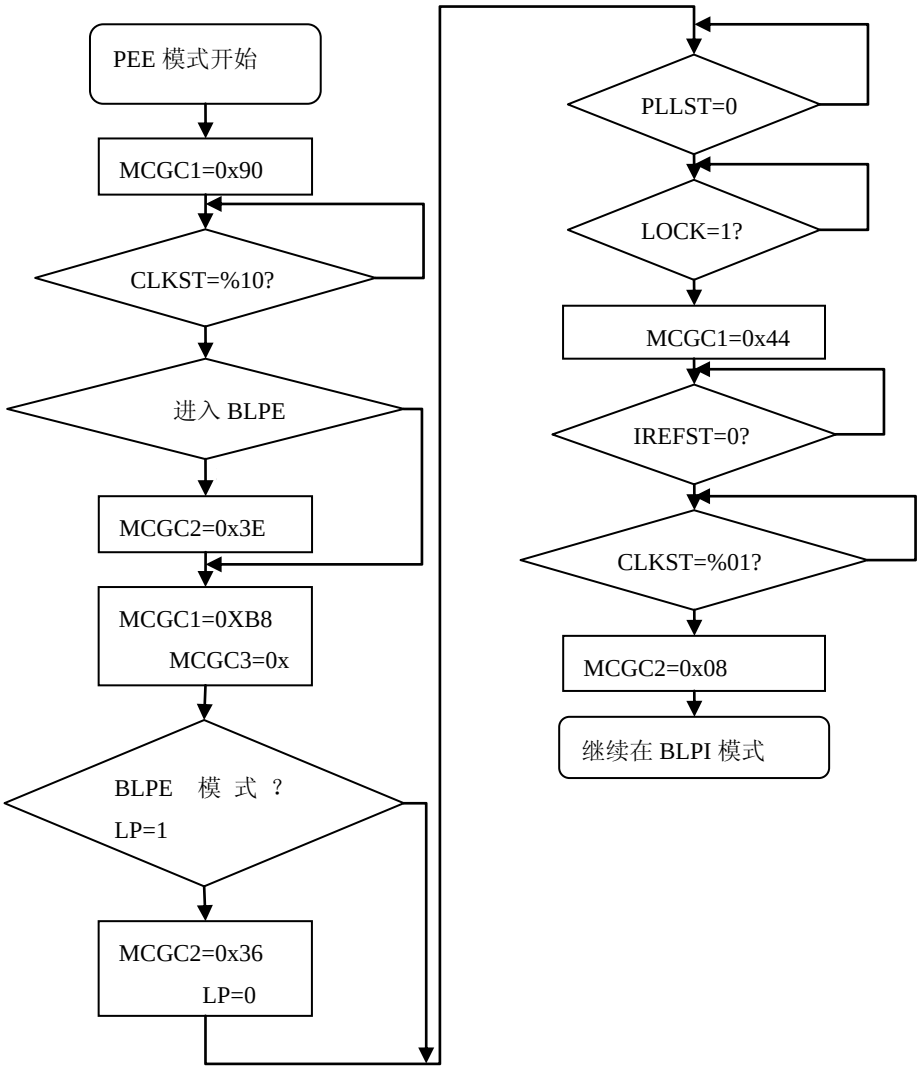


图9-10.从PEE到BLPI模式转换的流程图（使用4MHz晶振）

9.5.2.3 例#3: 从BLPI到PEE模式的转换: 外部晶振=4MHz，总线频率=16MHz

在本例中，MCG 将从 BLPI 模式（16kHz 总线频率）转换到到 FEE 模式（4MHz 晶振配置为 16MHz 总线频率）。首先，代码序列将被描述，然后流程图将给出。

1. 首先，BLPI 必须转换到 FBI 模式
   a) MCGC2=0x00（%00000000）
      i. MCGSC 寄存器中 LP（第 3 位）为 0
   b) 循环直到 MCGC 寄存器中 LOCK（第 6 位）为 1，显示 FLL 要求锁定。虽然 FLL 在 FBI 模式中是被忽略的，它仍然能够开启和运行。
2. 下一步，FBI 将转换到 FEE 模式。
   a) MCGC2=0x36（%00110110）
      i. RANGE（第 5 位）设置为 1，因为 4MHz 的频率在高频范围之内
      ii. HGO（第 4 位）设置为 1，用来配置外部振荡器高增益运行。
      iii. EREFS（第 2）位设为 1，因为晶振正在被使用。
      iv. ERCLKEN（第 1 位）设置为 1，确保外部参考时钟是激活的

b） 循环知道 MCGSC 寄存器中 OSCINIT（第 1 位）为 1，表明有 EREFS 位选择的晶振已经被初始化。

c） MCGC1=0x38（%00111000）

-CLKS（第 7 和 6 位）设置为%00，用来选择 FLL 的输出作为系统时钟源。

-RDIV（第 5-3 位）设置为%111，即 128 分频，因为 4MHz/128=31.25kHz，在 FLL 所要求的 31.25kHz 到 39.0625kHz 的范围之内。

-IREFS（第 1 位）清零，选择外部参考时钟。

d） 循环直到 MCGSC 寄存器中的 IREFST（第 4 位）为 0，表示外部参考时钟是参考时钟的当前时钟源。

e） 循环直到 MCGSC 中的 LOCK（第 6 位）为 1，表示 FLL 要求锁定。

f） 循环直到 MCGSC 寄存器中 CLKST（第 2 和 3 位）为%00，表示 FLL 的输出被选择为 MCGOUT。

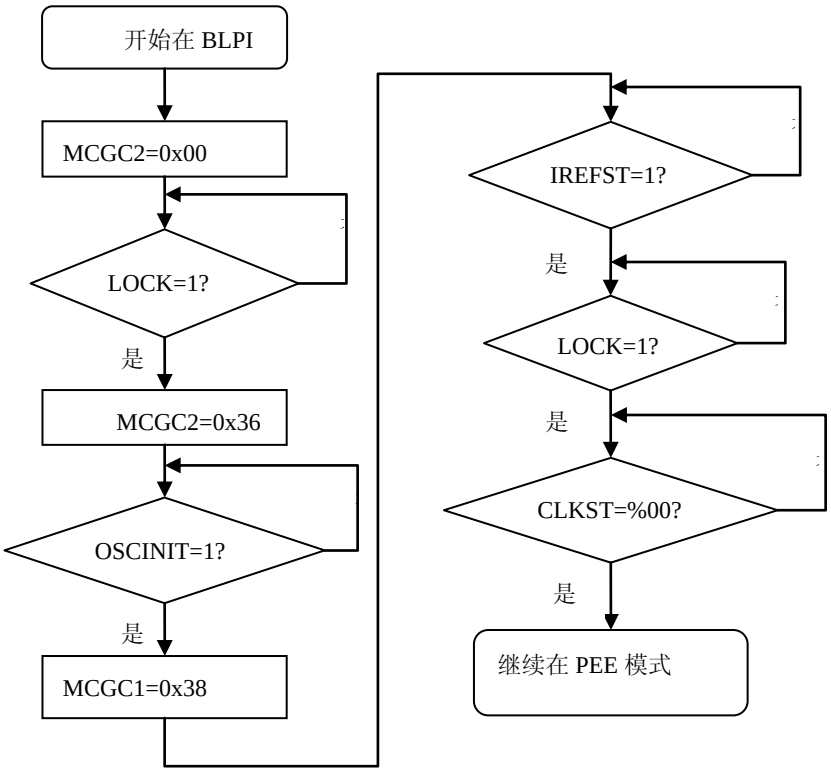


图9-11.从BLPI到FEE模式转换的流程图（使用4MHz晶振）

9.5.2.4 例#4: 从FEI到PEE模式：外部晶振=8MHz，总线频率为8MHz

在本节中，在 8MHz 晶振参考频率被设置到达 8MHz 后，MCG 将从 FEI 模式转换为 PEE 模式。

本例和例#1 相似，除了本例中外部晶振频率为 8MHz 而不是 4MHz。因为从 FEI 模式转换为 PEE 模式有一个周期时间 FLL 运行在一个 FLL 所允许的最高频率之上，所以有必要进行一个特殊的考虑。出现这种情况是因为当使用 8MHz 的外部晶振时，如果用最高的 128 分频结果为 62.5kHz（比最大的 39.0625kHz 还大）。当 FLL 运行在这种条件下时，必须小心的使用软件来尽量降低在这上面花费的时间。

下面的代码序列描述了如何从 FEI 模式转换为 PEE 模式，当 8MHz 晶振参考频率被设

置达到 8MHz 的总线频率。因为 MCG 在 FEI 模式中，本例亦显示了如何在 PEE 模式中初始化 MCG。首先代码序列将被描述，然后将有个流程图说明序列的原理。

1. 首先，FEI 必须转换为 FBE 模式：

a) MCGC2=0x36（%00110110）

i. BDIV（第 6 和 7 位）设置为%00，即 1 分频

ii. RANGE（第 5 位）设置为 1，因为 8MHz 的频率在高频率范围之内。

iii. HGO（第 4 位）设置为 1，配置外部振荡器作高增益操作

iv. EREFS（第 2 位）设置为 1，因为晶振已经被使用

v. ERCLKEN（第 1 位）设置为 1，来确保外部参考时钟有效

b）循环直到 MCGSC 寄存器中 OSCINIT（第 1 位）为 1，表示由 EREFS 位选择的晶振被初始化。

3. 模块中断（通过设置 CCR 中的中断位来开启）

4. MCGC1=0xB8（%10111000）

-CLKS（第 7 和 6 位）设置为%10，用来选择外部参考时钟作为系统时钟源

-RDIV（第 5-3 位）设置为%111，即 128 分频

注意：8MHz/128=62.5MHz 超出了 FLL 的最高频率。因此当到 FBE 模式的转换完成之后，软件必须立即通过设置 MCGC2 中的 LP 位来转换的 BLPE 模式。

-IREFS（第 2 位）清零。选择外部参考时钟

e）循环直到 MCGSC 寄存器中 IREFST（第 4 位）为 0，表示外部参考时钟是参考时钟的当前时钟。

f）循环直到 MCGSC 中 CLKST（第 2 和 3 位）为 i%10，表示外部参考时钟被选择作为 MCGOUT。

2. 然后，FBE 模式转换为 BLPE 模式

a）MCGC2=0x3E（%00111110）

-MCGC2 中的 LP（第 3 位）设置为 1（进入 BLPE 模式）

注意：在第一步的 d 到第二步的 a 之间必须没有其他操作（包括中断）。

b）开启中断（通过清除 CCR 中的中断位）

c）MCGC1=0x98（%10011000）

-RDIV（第 5-3 位）设置为%011，即 8 分频，因为 8MHz=1MHz 在 PLL 要求的 1MHz

到 2MHz 之间。在 BLPE 模式中，RDIV 的设置不重要因为 FLL 和 PLL 都无效。只有当在 PBE 模式中为 PLL 设置分频因子时才要改变他们。

d）MCGC3=0x44（%01000100）

-PLLS（第 6 位）设置为 1，选择 PLL。在 BLPE 模式中，改变这位只是让 MCG 在 PBE 模式中准备好 PLL 的使用。

-VDIV（第 3-0 位）设置为%0100，即 16 倍频，因为 1MHz*16=16MHz。在 BLPE 模式中，VDIV 的设置不重要，因为 PLL 无效。改变他们只能为 PBE 模式中的 PLL 设置倍频因子。

e）循环直到 MCGSC 寄存器中 PLLST（第 5 位）为 1，表明 PLLS 时钟的当前时钟源是 PLL。

3. 然后 BLPE 模式转换为 PBE 模式：

a）在 MCGC2 寄存器中设置 LP（第 3 位）为 0，转换到 PBE 模式

b）然后循环知道 MCGSC 寄存器中 LCOK（第 6 位）为 1 ，表示 PLL 已经要求锁定。

4. 最后，PBE 模式转换为 PEE 模式：

a）MCGC1=0x18（%00011000）

-MCGSC1 寄存器中 CLKS（第 6 和第 7 位）设置为%00，为了选择 PLL 的输出作为系统时钟源。

-循环直到 MCGSC 中 CLKST（第 2 和 3 位）为%11，表明 PLL 输出在当前时钟模式被作为 MCGOUT。

b）现在，RDIV 8 分频，BDIV 1 分频，VDIV 16 倍频，MCGOUT=[（8MHz/8）*16]/1=16MHz,总线频率是 MCGOUT/2，即 8MHz。

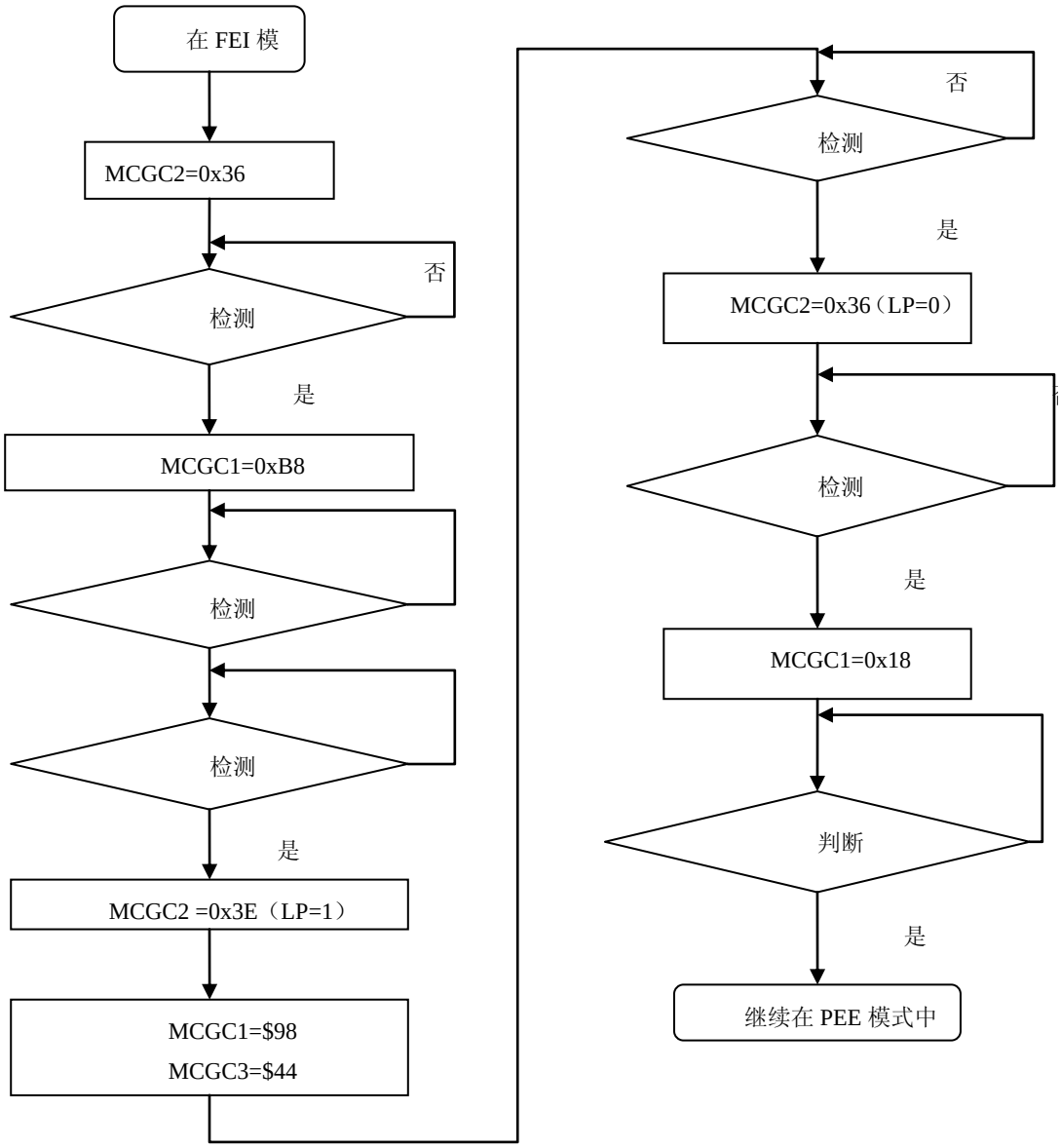


图9-12FEI到PEE模式转换的流程图（使用8MHz晶振）

9.5.3 测定内部参考时钟（IRC）

通过首先向 MCGTRM 寄存器设置来测定 IRC，然后使用 FTRIM 位来确定频率。我们将借助 9 位数据（从 0x000 到 0x1FF）来作为修正数据，此时 FTRIM 位即为 LSB。

在 POR 之后修正数据总是 0x100（MCGTRM= 0x80 FTRIM 0 ）。写入较大的数据将增加频率，较小的数据将降低品路。除了在封装测试过程中产生的在修正值和周期之间的一些微小的非线性变化，修正值和周期是成线性关系的。这些非线性现象说明要多次校正从而找到最好的校正数据。在本节例#4 中，就要这种情况。

在修正数据被找到之后，它能够被保存在 FLASH 内存中。如果器件切断电源，IRC 能够很容易的将数据从内存中保存到 MCG 寄存器。Freescale 一致推荐使用 FLASH 来保存校正数据。当器件在工厂被校正后，校正数据也保存在这些位置。详细位置请参考内存映射表。

9.5.3.1 例#5 ： 内部参考时钟校正

对于那些有严格频率要求的应用，校正程序能够提供一个精确的内部时钟源。本节将描述校正内部振荡器。

在下例中，MCG 校正将由 9 位的 MCGTRM 和 FTRIM 集合来校准。这个值将被当作 TRMVAL。

1) ATE 时钟，500us 周期
2) MCG 用 8MHz 配置内部参考时钟

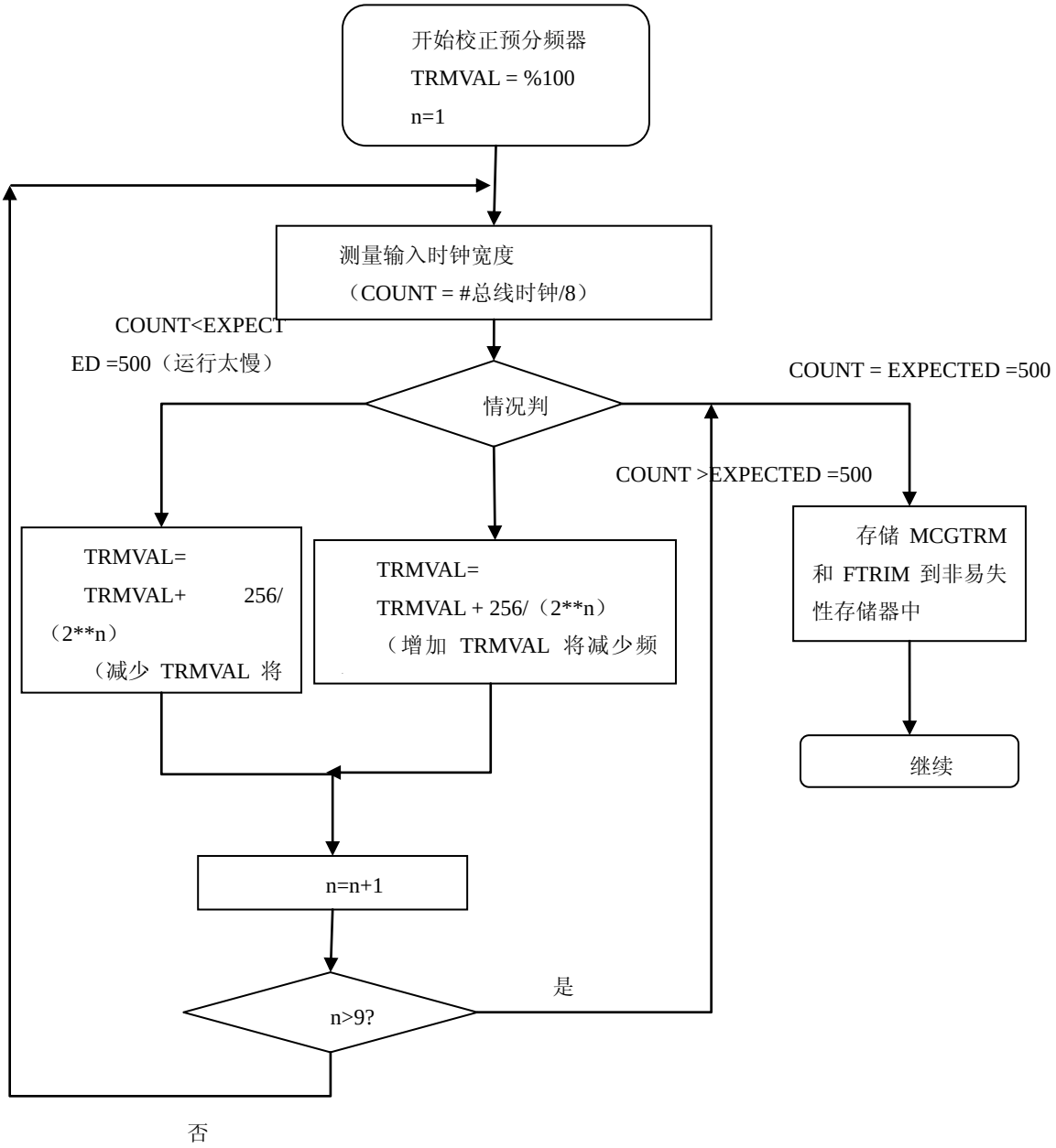


图9-13.校正程序

在这个特别的例子中，MCU 被安置在 PCB 中，并且整个系统被测试装最终测试过。提供给 MCU 的各个信号和消息由用户软件控制。当测试中提供指示时，MCU 开始校正程序如图 12-13 所示。

如果想要的总线频率在器件的最高频率附近，推荐使用参考分频因子的两倍值来校正。当校正完成之后，参考分频因子可以保存。从而防止意外的情况。

第十章 模数定时器（S08MTIMV1）

10.1 简介

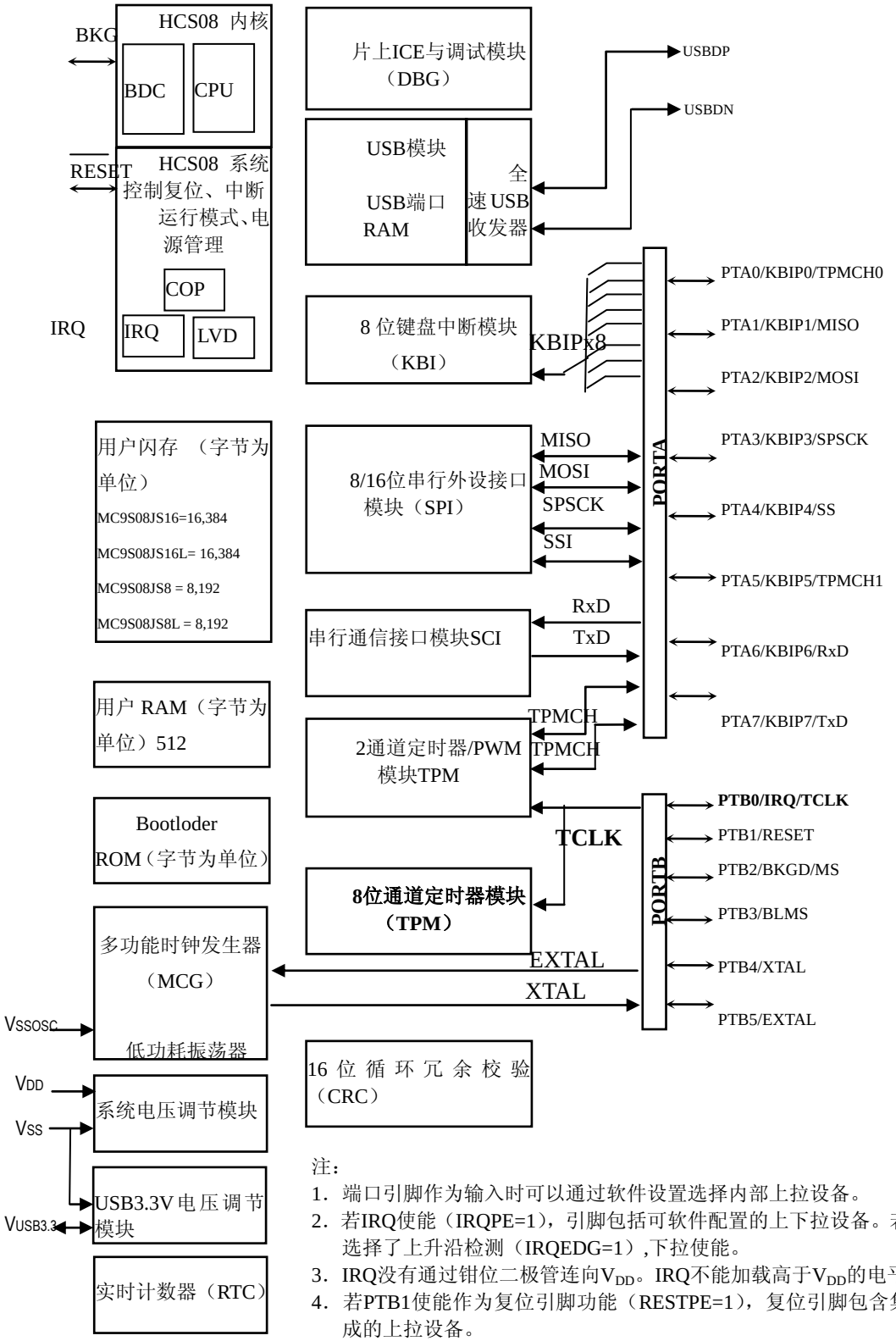
MTIM 是一个 8 位定时器，它有几个软件可选的时钟源，和一个可编程中断。

MTIM 的核心是一个 8 位的计数器，这个计数器能当作一个自由计数器或者一个模数计数器。定时器溢出中断能够产生周期的中断，用于基于时间的软件循环。

图 10-1 显示了 MCGS08JS16 系列框图，高亮显示 MTIM。

10.1.1 MTIM配置信息

TCLK 是 MTIM 模块的外部时钟，由设置 MTIMCLK 寄存器中 CLKS=1:1 或者 1:0 来选择的，这选择 TCLK 引脚为输入。PTB0 的 TCLK 输入能够同时的作为 MTIM 和 TPM 模块的外部时钟。



注：

1. 端口引脚作为输入时可以通过软件设置选择内部上拉设备。
2. 若IRQ使能（IRQPE=1），引脚包括可软件配置的上下拉设备。若选择了上升沿检测（IRQEDG=1）,下拉使能。
3. IRQ没有通过钳位二极管连向V_{DD}。IRQ不能加载高于V_{DD}的电平
4. 若PTB1使能作为复位引脚功能（RESTPE=1），复位引脚包含集成的上拉设备。
5. 引脚包含集成上拉设备。
6. 当KBI使能（KBIPEn=1）而且相关引脚被配置为上拉设备使能，那么上升沿（KBEDGn=1）被用于将上拉设备配置为下拉设备。

图 10-1. MC9S08JS16系列框图MTIM模块和引脚高亮显示

10.1.2 特性

定时器特性包括：

- 8 位向上计数器
  - 自由运行或者 8 位模数限制
  - 溢出时软件可控中断
  - 计数器复位位（TRST）
  - 计数器停止位（TSTP）
- 四个软件可选的时钟源到预分频器：
  - 系统总线时钟-上升沿
  - 混合频率时钟（XCLK）-上升沿
  - TCLK 引脚上的外部时钟-上升沿
  - TCLK 引脚上的外部时钟-下降沿
- 九种预分频值：
  - 时钟源可以被 1，2，4，8，16，32，64，128 或者 256 分频

10.1.3 操作模式

本节定义了 MTIM 在停止（STOP），等待（WAIT）和背景调试（BACKGROUND DEBUG）模式中的操作。

10.1.3.1 等待（WAIT）模式下的MTIM

MTIM 将持续运行在等待（WAIT）模式下，直到完成等待（WAIT）模式下的操作。所以,如果定时器溢出中断被允许的话，MTIM 能够被用来将 MCU 带出等待（WAIT）模式。为了降低功耗，如果在等待（WAIT）模式下，MTIM 不需要被当作中断源时，应该被停止。

10.1.3.2 停止（STOP）模式下的模式

在所有停止（STOP）模式中，MTIM 都是禁止的，无论在完成停止（STOP）操作之前是什么设置。因此，MTIM 被能够将 MCU 从停止（STOP）模式当中唤醒。

当 MCU 从 stop1 和 stop2 模式中唤醒时，MTIM 将进入复位状态。如果 stop3 因为复位退出，MTIM 同样将处于复位状态。如果 stop3 因为中断而退出，MTIM 将保持进入 stop3 模式时的状态。如果当进入 stop3 是计数器是激活的，计数器将从当前的值继续计数。

10.1.3.3 主动背景模式下的MTIM

MTIM 将暂计数停直到微控制器回到正常用户运行模式。只要 MTIM 复位不发生，将恢复计数。（TRST 被写 1，或者是 MTIMMOD 被置位）。

10.1.4 框图

图 10-2 显示了计数器模块的框图

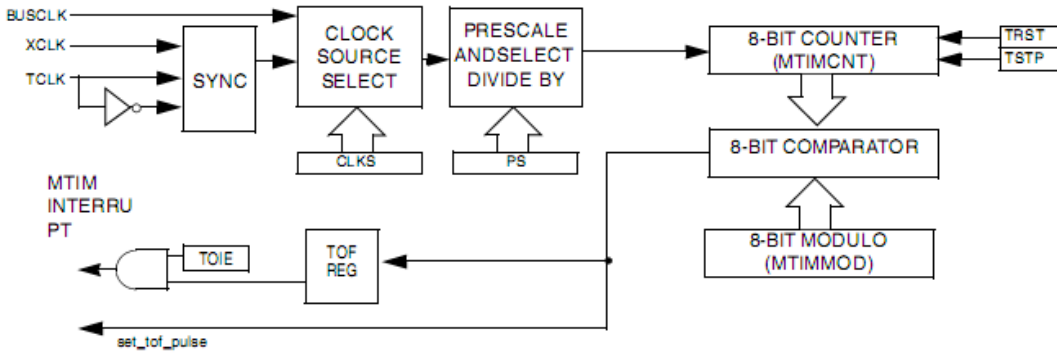


图10-2.定时器模块（MTIM）框图

10.2 外部信号描述

MTIM 包括了一个外部信号：TCLK，当被选择作为 MTIM 时钟源时被用来输入外部时钟。TCLK 的信号属性如表 10-1 所示。

表10-1.信号属性

信号	功能	I/O
TCLK	外部时钟源输入到 MTIM	1

TCLK 输入必须和总线时钟保持一致。同样，周期的变化和时钟的毛刺必须被校正。因此，TCLK 信号必须被限制在总线频率的四分之一。TCLK 引脚能够是一个普通的引脚。参看引脚和连接章节。

10.3 寄存器定义

图 10-3 是 MTIM 寄存器的概述。

图10-3.MTIM寄存器概述

名称		7	6	5	4	3	2	1	0
MTIMSC	R	TOF	TOIE	0	TSTP	0	0	0	0
	W			TRST					
MTIMCLK	R	0	0	CLKS		PS			
	W								
MTIMCNT	R	COUNT							
	W								
MTIMMOD	R	MOD							
	W								

每个 MTIM 包括四个寄存器

- 一个 8 位状态和控制寄存器
- 一个 8 位时钟配置寄存器
- 一个 8 位计数器寄存器
- 一个 8 位模块寄存器

如果要得到 MTIM 寄存器绝对地址的分配，请在内存章节中参考直接页寄存器概述。被节所涉及的寄存器和控制位都是通过名称和相对地址来进行的。

某些 MCU 可能拥有不止一个 MTIM，所以寄存器名称包含占位字符来识别所涉及的 MTIM。

10.3.1 MTIM状态和控制寄存器（MITMSC）

MTIMSC 包含溢出状态标志和控制位，该控制位被用来配置中断使能，复位计数器和停止计数器。

	7	6	5	4	3	2	1	0
R	TOF	TOIE	0	TSTP	0	0	0	0
W			TRST					
复位:	0	0	0	1	0	0	0	0

图10-4.MTIM状态和控制寄存器

表10-2.MTIM状态和控制寄存器位域描述

位域	描述
7 TOF	**MTIM 溢出标志** – 当计数器达到 MTIM 模数寄存器值时，MTIM 计数器溢出复位为 0x00 并将此只读位写为 1。 0 MTIM 计数器没有达到 MTIM 模数寄存器中的溢出值。 1 MTIM 计数器达到了 MTIM 模数寄存器中的溢出值。
6 TOIE	**MTIM 溢出中断使能** – 此读写位使能 MTIM 溢出中断。如果 TOIE 被置位，然后当 TOF=1 时一个中断产生。复位时 TOIE 被清除。如果 TOF=1 的话就不要设置 TOIE. 首先清除 TOF,然后设置 TOIE。 0 TOF 中断禁止。使用软件获得 1 TOF 中断使能
5 TRST	**MTIM 计数器复位** – 当此只写位被写 1 时，MTIM 计数器寄存器复位为 0x00 并且 TOF 被清除。读此位总是返回 0。 0 没有影响。MTIM 计数器保持当前的状态。 1 MTIM 计数器复位为 0x00
4 TRST	**MTIM 计数器停止位** – 当被置位是，此读写位将停止 MTIM 计数器。当 TSTP 被清除时，计数器将继续计数。复位将设置 TSTP 来阻止 MTIM 计数。 0 MTIM 计数器有效 1 MTIM 计数器停止
3:0	未使用位，总是为 0

10.3.2 MTIM时钟配置寄存器（MTIMCLK）

MTIMCLK 包括时钟选择位（CLKS）和预分频器选择位（PS）。

	7	6	5	4	3	2	1	0
R	0	0	CLKS		PS			
W								
复位:	0	0	0	0	0	0	0	0

图10-5.MTIM时钟配置寄存器

表10-3.MTIM时钟配置寄存器位域描述

位域	描述
7:6	未使用位，总为 0。
5:4	**时钟源选择** – 这两个读写位选择四种时钟源中的一种作为 MTIM 预分频器的输

CLKS	入。当计数器运行时，这两位的改变不会清除计数器的数值。计数器将继续用新的时钟源计数。复位后 CLKS 将被清除为 0x00。 00 编码 0.总线时钟（BUSCLK） 01 编码 1.混合频率时钟（XCLK） 10 编码 3.外部时钟源（TCLK 引脚），下降沿 11 编码 3.外部时钟源（TCLK 引脚），上升沿 其他编码默认为总线时钟（BUSCLK）。
3:0 PS	**时钟源预分频器** – 这四位读写位从 8 位域分频器中的九种输出中选择一种。同样，改变预分频器的值将不会清除运行中计数器的数值。复位将清除 PS 为 0x000。 0000 编码 0.MTIM 时钟源 1 分频 0001 编码 0.MTIM 时钟源 2 分频 0010 编码 0.MTIM 时钟源 4 分频 0011 编码 0.MTIM 时钟源 8 分频 0100 编码 0.MTIM 时钟源 16 分频 0101 编码 0.MTIM 时钟源 32 分频 0110 编码 0.MTIM 时钟源 64 分频 0111 编码 0.MTIM 时钟源 128 分频 1000 编码 0.MTIM 时钟源 256 分频 其他编码默认将 MTIM 时钟 256 分频。

10.3.3 MTIM计数器寄存器（MTIMCNT）

MTIMCNT 是 MTIM8 位计数器的当前值，他是只读的。

	7	6	5	4	3	2	1	0
R	COUNT							
W								
复位:	0	0	0	0	0	0	0	0

图10-6.MTIM计数器寄存器

表10-4.MTIM计数器寄存器位域描述

位域	描述
7:0 COUNT	**MTIM 计数器** – 这把个只读位包含 8 位计数器的当前值。写无效。复位将清除计数值为 0x00。

10.3.4 TIM模数寄存器（MTIMMOD）

	7	6	5	4	3	2	1	0
R	MOD							
W								
复位:	0	0	0	0	0	0	0	0

图10-7.MTIM模数寄存器

表10-5.MTIM模数寄存器位域描述

位域	描述
7:0	**MTIM 模数** – 这 8 个读写位包括重设计数值和设置 TOF 的模值。0x00 值将置 MTIM 与自由运行模式。向 MTIMMOD 写将复位 COUNT 为 0x00

	并且清除 TOF。复位将设置模数寄存器为 0x00。

10.4 功能描述

MTIM 由一个包含一个 8 位模数寄存器的主 8 位向上计数器，一个时钟源选择器，和一个有九个可选值的预分频器组成。MTIM 模块也包括软件可选的中断逻辑。

MTIM 计数器（MTIMCNT）有三个运行模式：停止（STOPPED）,自由运行（FREE-RUNNING）和模数（MODULO）。复位后，计数器停止。如果在计数器开始前没有像模式寄存器中写入数值，那么计数器将处于自由运行（FREE-RUNNING）模式。当计数器运行时模数寄存器中的值不为 0x00 时，计数器将处于模数模式。

当 MCU 复位后，计数器停止并且复位为 0x00，并且模数寄存器被设置为 0x00。总线时钟被选择作为默认的时钟源并且预分频器为 1 分频。要在自由运行（FREE-RUNNING）模式中开支 MTIM，只要向 MTIM 状态和控制寄存器（MTIMSC）置位并且清除 MTIM 停止位（TSTP）。

四种软件可选的时钟源：内部总线时钟，复合频率时钟（XCLK），和一个 TCLK 引脚上的外部时钟，选择上升沿或者下降沿作为计数增长方式。MTIM 在 MTIMSC 寄存器中时钟源选择位（CLKS1:CLKS0）用来选择预想的时钟源。如果当一个新的时钟源被选择时计数器是有效的（TSTP=0），计数器将使用新时钟源预分频后的频率继续计数。

九种预分频器值软件可选：时钟源可以被 1，2，4，8，16，32，64，128，或者是 256 分频。MTIMSC 寄存器中预分频器选择位（PS[3:0]）选择想要的预分频值。当一个新的预分频值被选择时如果计数器是有效的（TSTP=0），那么计数器将用新的预分频值继续计数。

MTIM 模数寄存器（MTIMMOD）允许将溢出比较值设置为 0x01 到 0xFF 中的任何值。复位将清除模数值为 0x00，这将计数器处于自由运行（FREE-RUNNING）模式。

当计数器运行时（TSTP=0），计数器将以一个选的速率持续计数直到计数值达到模数值。当达到时，计数器溢出变为 0x00 并且继续计数。当计数器溢出时，MTIM 溢出标志（TOF）被设置。当模数值变为 0x00 的过程中，标志位将被设置。向 MTIMMOD 写数值时，如果计数器在运行则它将被设置为 0x00 并且清除 TOF 位。

清除 TOF 位要经过两个步骤。第一个步骤是当 TOF 被设置时读 MTIMSC 寄存器。第二个步骤是向 TOF 写 0。如果另一个溢出发生在两个步骤之间发生，那么清除过程将复位并且 TOF 在第二个步骤完成后将保持设置。这将避免第二个事件丢失。当 TRST 被写 1 或者 MTIMMOD 寄存器写入任何值时，TOF 也要被清除。

当 TOF 被设置时 MTIM 允许产生一个可选的中断。要使能 MTIM 溢出中断，在 MTIMSC 寄存器中设置 MTIM 溢出中断使能位（TOIE）。当 TOF=1 时，TOIE 绝不能写 1。相反，TOF 应该首先被清除然后 TOIE 能够写 1。

10.4.1 MTIM运行实例

本节将呈现一个 MTIM 运行实例，在此例中计数器从模数寄存器计数到匹配值。

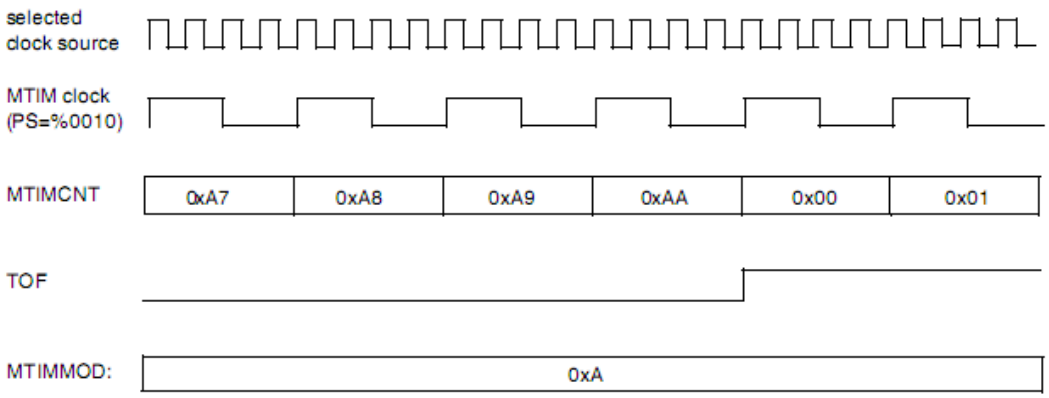


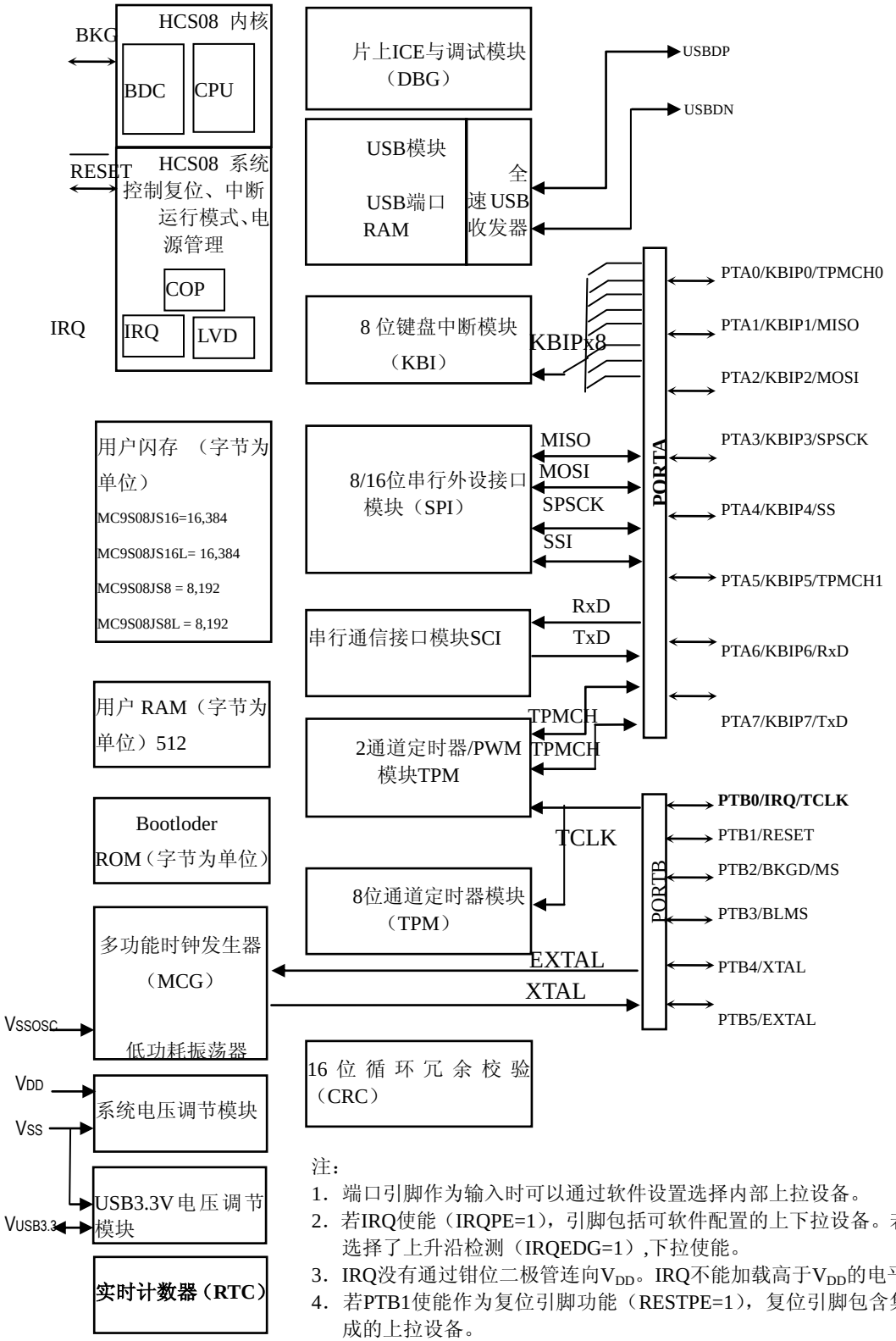
图10-8MTIM计数器溢出实例

在图 10-8 的例子中，时钟源可以为五种可以选择的任意一种。预分频器被设置为%0010 即 4 分频。MTIMMOD 寄存器中模数值被设置为 0xAA。当计数器（MTIMCNT）达到模数值 0xAA，计数器溢出到 0x00 并且继续计数。当计数器从 0xAA 变为 0x00 时，计数器溢出标志（TOF）被设置。如果 TOIE=1,当 TOF 被设置时将产生一个 MTIM 溢出中断。

第11章 实时计数器（S08RTCV1）

11.1 介绍

实时计数器（RTC）包含一个 8 位计数器，一个 8 位比较器，几个 2 进制和 10 进制预分频器，两个时钟源，和一个可编程周期中断。本模块能够被用作一天内的时间，日历，或者任何任务日程安排功能。它同样能够作为一个周期地将 MCU 唤醒的服务，不需要外部器件。



注：

1. 端口引脚作为输入时可以通过软件设置选择内部上拉设备。
2. 若IRQ使能（IRQPE=1），引脚包括可软件配置的上下拉设备。若选择了上升沿检测（IRQEDG=1）,下拉使能。
3. IRQ没有通过钳位二极管连向V_{DD}。IRQ不能加载高于V_{DD}的电平
4. 若PTB1使能作为复位引脚功能（RESTPE=1），复位引脚包含集成的上拉设备。
5. 引脚包含集成上拉设备。
6. 当KBI使能（KBIPEn=1）而且相关引脚被配置为上拉设备使能，那么上升沿（KBEDGn=1）被用于将上拉设备配置为下拉设备。

图 11-1. MC9S08JS16系列框图RTC模块和引脚高亮显示

11.1.1 特性

RTC 模块的特性：

- 8 位向上计数器
  - 8 位模块匹配极限
  - 软件可控周期匹配中断
- 三种软件可选的输入时钟源，用可选择的二进制和十进制的分频因子来作预分频
  - 1KHz 内部低功耗振荡器（LPO）
  - 外部时钟（ERCLK）
  - 32KHz 内部时钟（IRCLK）

11.1.2 运行模式

本节定义了停止（STOP），等待（WAIT）和背景调试（BACKGROUND DEBUG）模式。

11.1.2.1 等待模式

在完成合适的操作之前，RTC 将一直运行在等待模式。应此如果实时中断被允许的话 RTC 能够将 MCU 带出等待模式。为了降低功耗，RTC 模块如果在等待模式不需要作为一个中断源的话，应该被软件停止。

11.1.2.2 停止模式

如果 RTC 在退出停止模式之前激活，RTC 将一直运行在 stop2 或者 stop3 模式。因此，如果实时时钟中断激活的话，RTC 能够将 MCU 退出停止模式而且不需要外部器件。

LPO 时钟能够被用在 stop2 和 stop3 模式。ERCLK 和 IRCLK 时钟只能在 stop3 模式下有效。

当所有时钟源激活的话功耗是很低的，在这种情况下，实时中断不能够将 MCU 从停止模式中唤醒。

11.1.2.3 活跃后台模式

RTC 在活跃后台模式时暂停所有计数。只有当 TRCMOD 寄存器没有被写并且 RTCPS 和 RTCLKS 位没有被改变时，计数器恢复计数。

11.1.3 框图

RTC 模块框图如图 11-2 所示。

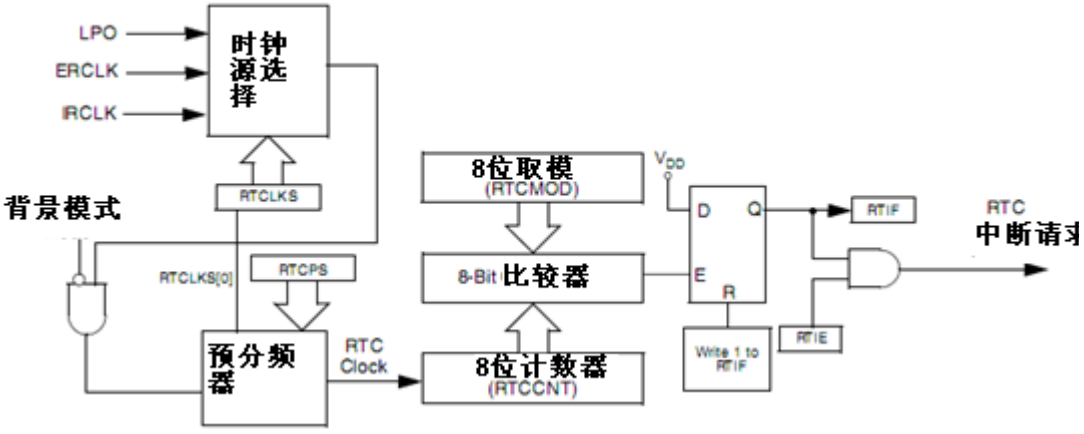


图11-2 实时计数器（RTC）框图

11.2 外部信号描述

RTC 不包括任何外部信号。

11.3 寄存器定义

RTC 包含一个状态和控制寄存器，一个 8 位计数寄存器，和一个 8 位模块寄存器。
参考直接页寄存器中的 RTC 寄存器绝对地址总览在内存章节。本节只使用它们的名字。

表11-1 RTC寄存器概述

名称		7	6	5	4	2	1	0
RTCSC	R	RTIF	RTCLKS		RTIE	RTCPS		
	W							
RTICNT	R	RTCCNT						
	W							
RTCMOD	R	RTCMOD						
	W							

11.3.1 RTC状态和控制寄存器（RTCSC）

RTCSC 包括一个实时中断状态标志（RTIF），时钟选择为（RTCLKS），实时中断使能位（RTIE），并且预分频选择位（RTCPS）。

	7	6	5	4	3	2	1	0
R	RTIF	RTCLKS		RTIE	RTCPS			
W								
复位:	0	0	0	0	0	0	0	0

图11-3 RTC状态和控制寄存器（RTCSC）

表11-2.RTCSC位域描述

位域	描述

7 RTIF	**实时时钟中断标志**，本状态为指示了 RTC 控制寄存器 RTC 的值到达 RTC 模块寄存器。写逻辑 0 无效。写逻辑 1 清除位并且产生实时时钟中断请求。
6-5 RTCLKS	**实时时钟源选择**。这两位读写为选择输入到 RTC 预分频器的时钟源。改变时钟源能够清楚预分频器和 RTCCNT 计数器。当选择一个时钟源时，确保那个时钟源是可用的来确保 RTC 正确运行。复位将清除 RTCLKS 位。 00 实时时钟源为 1kHz 低功耗振荡器（LPO） 01 实时时钟源为外部时钟（ERCLK）1x 实时时钟源为内部时钟（IRCLK）
4 RTIE	**实时中断使能。**本读写位使能实时中断。如果 RTIE 被设置，那么当 RTIF 被设置时将产生一个中断。复位将清除 RTIE。 0 实时中断请求被禁止。使用软件测试。 1 实时中断请求被允许。
3-0 TRCPS	**实时时钟预分频器选择。**这四位读写位选择二进制和十进制形式将时钟源分频。见表 13-3.改变预分频器的数值，将清除预分频器和 RTCCNT 计数器。复位将清除 RTCPS。

表11-3RTC预分频因子值

RTCLKS[0]	**RTCPS**															
										9	**1 0**	**1 1**	**1 2**	**1 3**	**1 4**	**1 5**
0	ff	2^3	2^5	2^6	2^7	2^8	2^9	2^{10}		2	2^2	0	2^4	10^2	$5 * 10^2$	10^3
0	ff	2^{10}	2^{11}	2^{12}	2^{13}	2^{14}	2^{15}	2^{16}	10^3	$2 * 10^3$	$5 * 10^3$	10^4	$2 * 10^4$	$5 * 10^4$	10^5	$2 * 10^5$

11.3.2 RTC计数器寄存器（RTCCNT）

RTCCNT 是 8 位计数器的 RTC 当前计数的只读值。

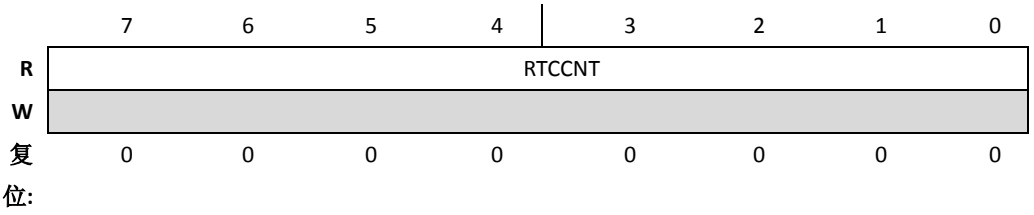
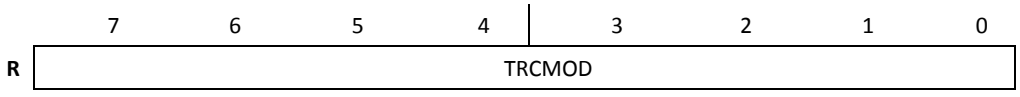


图11-4 RTC计数寄存器（RTCCNT）

表11-4 RTCCNT位域描述

位域	描述
7:0	**RTC 计数**。这八位只读为包括了 8 位计数器的当前值。向这些寄存器写是无效的。复位，向 RTCMOD 写，或者写不同的值到 RTCLKS 和 RTCPS 将清除计数器为 0x00.

11.3.3 RTC模块寄存器（RTCMOD）



W								
复位:	0	0	0	0	0	0	0	0

图11-5 RTC模块寄存器（RTCMOD）

表11-5 RTCMOD位域描述

位域	描述
7:0 RTCMOD	**RTC 模块**。这八位读写位包括一个用于将计数器复位为 0x00 的模块的值和一个设置 RTIF 状态位。0x00 在每个预分频器输出的每个上升沿设置 RTIF 位。向 RTCMOD 写数据将复位预分频器和 RTCCNT 计数器到 0x00.复位将设置本模块为 0x00.

11.4 功能描述

RTC 有一个带 8 位的模块寄存器的 8 位向上计数器，一个时钟源选择器，和一个带二进制和十进制的可选择值的预分频模块。模块也包括一个软件可选的中断逻辑。

在 MCU 复位后，计数器变为 0x00，模块寄存器变为 0x00，并且预分频器关闭。1kHz 的内部振荡器作为默认的时钟源。向预分频器选择位（RTCPS）写任何非零值就可以启动预分频器。

三种软件可选的时钟源：低功耗震荡时钟（LPO），外部时钟（ERCLK）,和内部时钟（IRCLK）。RTC 时钟选择位(RTCLKS)选择所需的时钟源。一旦一个新值被写入 PTCLKS，预分频器和 RTCCNT 计数器就被复位至 0x00。RTCPS 和 RTCLKS[0]位选择分频因子。如果一个新值被写入 RTCPS 预分频器和 RTCCNT 计数器被复位至 0x00。表 13-6 显示了不同预分频周期的值。

RTCPS	1-kHz 内部时钟 (RTCLKS = 00)	1-MHz 内部时钟 (RTCLKS = 01)	32-kHz 内部时钟 (RTCLKS = 10)	32-kHz内部时钟 (RTCLKS = 11)
0000	Off	Off	Off	Off
0001	8 ms	1.024 ms	250 µs	32 ms
0010	32 ms	2.048 ms	1 ms	64 ms
0011	64 ms	4.096 ms	2 ms	128 ms
0100	128 ms	8.192 ms	4 ms	256 ms
0101	256 ms	16.4 ms	8 ms	512 ms
0110	512 ms	32.8 ms	16 ms	1.024 s
0111	1.024 s	65.5 ms	32 ms	2.048 s
1000	1 ms	1 ms	31.25 µs	31.25 ms
1001	2 ms	2 ms	62.5 µs	62.5 ms
1010	4 ms	5 ms	125 µs	156.25 ms
1011	10 ms	10 ms	312.5 µs	312.5 ms
1100	16 ms	20 ms	0.5 ms	0.625 s
1101	0.1 s	50 ms	3.125 ms	1.5625 s
1110	0.5 s	0.1 s	15.625 ms	3.125 s
1111	1 s	0.2 s	31.25 ms	6.25 s

表11-6 预分频周期

RTC 模块寄存器（RTCMOD）允许比较值被设置为 0x00 到 0xFF。当计数器激活时，计数器以一个选择的速率来确定增量直到计数达到模块预定义的值。当达到时，计数器复位为 0x00，并且继续计数。当匹配发生时，实时中断标志为被设置。模数值变为 0x00 时标志位被设置。向 RTCMOD 写后将复位预分频器和置 RTCCNT 计数器为 0x00。

当 RTIF 被设置时，RTC 允许产生中断。在 RTCSC 寄存器中设置实时中断使能位（RTIE）就能够开启实时中断。向 RTIF 写 1 就能够清除 RTIF。

11.4.1 RTC运行模式

本节显示一个 RTC 运行实例，这个实例从模块寄存器达到预设数值。

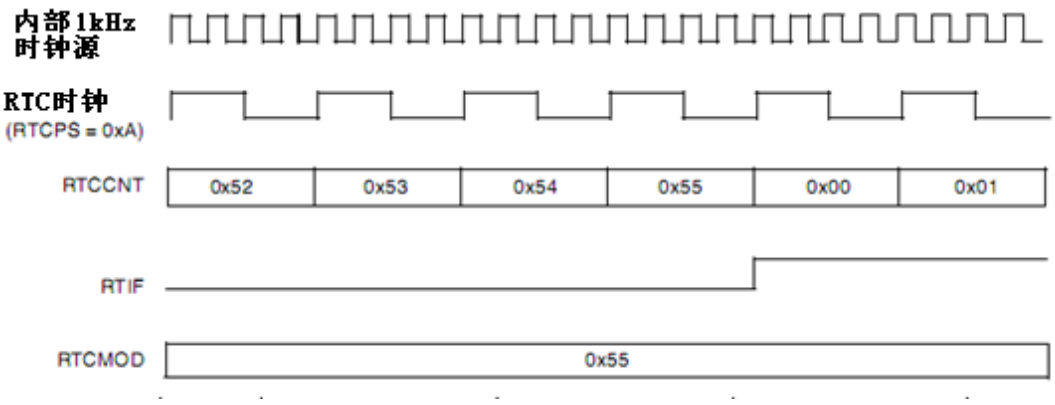


图11-6 RTC计数器溢出实例

在图 11-6 所示的例子中，时钟源为 1kHz 的能不振荡器时钟源。预分频器（RTCPS）被设置为 0xA 即 4 分频。RTCMOD 寄存器的模块值被设置为 0x55。当计数器（RTCCNT）达到模块值 0x55 时，计数器溢出变为 0x00 并且继续计数。当计数器值从 0x55 变为 0x00 时，实时中断标志（RTIF）被设置。当 RTIF 设置时，将产生一个实时中断。

11.5 初始化/应用信息

本节将提供一个样例代码向用户显示如何初始化和配置 RTC 模块。样例使用 C 语言实现。

一下样例显示如何用 RTC 实现一天内的时间，使用 1kHz 时钟源来达到最小的功耗。因为 1kHz 的时钟源不如晶振那样精确，可用软件调整。为了不使用调整从而降低功耗，外部时钟（ERCLK）或者内部时钟（IRCLK）能够被用来作为时钟源，但要选择合适的预分频和模块值。

```
/* Initialize the elapsed time counters */
Seconds = 0;
Minutes = 0;
Hours = 0;
Days=0;
/* Configure RTC to interrupt every 1 second from 1-kHz clock source */
RTCMOD.byte = 0x00;
RTCSC.byte = 0x1F;
/*************************************************************************
```

```
Function Name : RTC_ISR
Notes : Interrupt service routine for RTC module.
*************************************************************************/
#pragma TRAP_PROC
void RTC_ISR（void）
{
/* Clear the interrupt flag */
RTCSC.byte = RTCSC.byte | 0x80;
/* RTC interrupts every 1 Second */
Seconds++;
/* 60 seconds in a minute */
if （Seconds > 59）{
Minutes++;
Seconds = 0;
}
/* 60 minutes in an hour */
if （Minutes > 59）{
Hours++;
Minutes = 0;
}
/* 24 hours in a day */
if （Hours > 23）{
Days ++;
Hours = 0;
}
```

§

第12 章 串行通信接口（S08SCIV4）

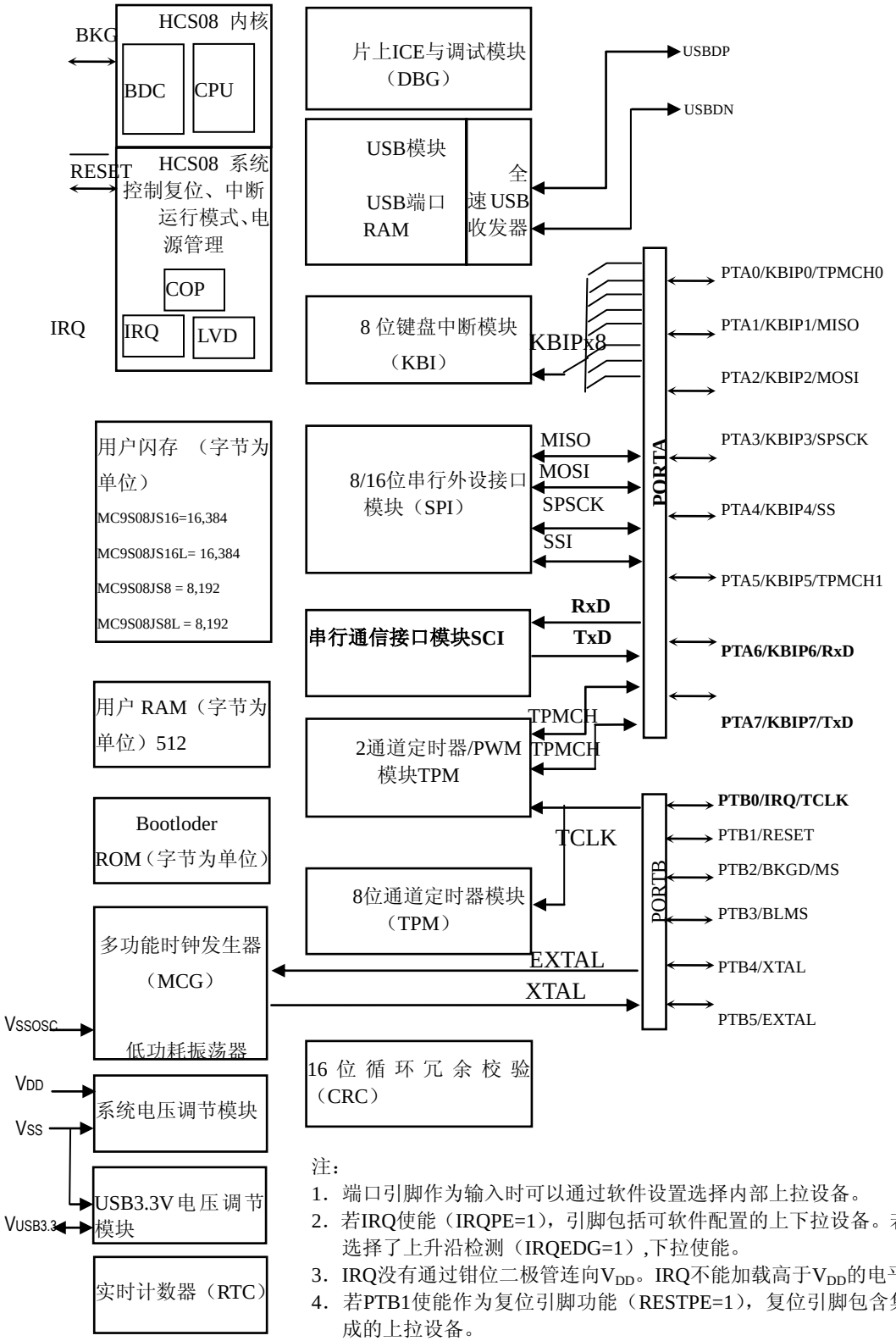
12.1 简介

MC9S08JS16 系列包含两个独立的串行通信接口模块（SCI）,也被称为通用异步收发器（UART）。通常，这些系统连接到个人计算机或工作站的 RS232 串行输入输出口（I/O），但也可以和其他嵌入式控制器通信。

灵活的 13 位模块化的波特率产生器支持超过 115.2 kB 的标准波特率。SCI 的发送和接收使用相同的波特率，并且每个 SCI 模块都有各自独立的波特率产生器。

本 SCI 提供了多种高级特性，这些特性很少在其他嵌入式控制器的异步串行（I/O）设备中出现。接收器使用先进的数据采样技术确保可靠的通信和噪音检测。也包括硬件奇偶校验，接收器唤醒，发送和接收的双缓冲。

注意：MC9S08JS16 系列器件运行在一个较高的电压范围（2.7V 到 5.5V）并且不包括 stop1 模式，因此，请忽略使用 stop1 模式。



注：

1. 端口引脚作为输入时可以通过软件设置选择内部上拉设备。
2. 若IRQ使能（IRQPE=1），引脚包括可软件配置的上下拉设备。若选择了上升沿检测（IRQEDG=1）,下拉使能。
3. IRQ没有通过钳位二极管连向V_{DD}。IRQ不能加载高于V_{DD}的电平
4. 若PTB1使能作为复位引脚功能（RESTPE=1），复位引脚包含集成的上拉设备。
5. 引脚包含集成上拉设备。
6. 当KBI使能（KBIPEn=1）而且相关引脚被配置为上拉设备使能，那么上升沿（KBEDGn=1）被用于将上拉设备配置为下拉设备。

图 12-1. MC9S08JS16系列框图SCI模块和引脚高亮显示

12.1.1 特性

SCI 模块具有以下特性：

- 全双工，标准的不归零 （NRZ）格式。
- 发送器和接收器内具有各自的双缓冲，并可分别使能。
- 可编程的波特率 （13 位模因子） 。
- 中断驱动或轮询操作：
- 发送数据寄存器空和发送完成
- 接收数据寄存器满
- 接收溢出，奇偶校验错误，帧错误，和干扰错误
- 接收器空闲检测
- 接收引脚上的有效边沿
- 间断检测支持 LIN
- 硬件奇偶产生和校验。
- 可编程的 8 位或 9 位的字符长度。
- 空闲线或地址标记唤醒接收器。
- 可选的 13 位间断字符产生或 11 位的间断字符检测。
- 可选的发送器输出极性。

12.1.2 操作模式

在下列模式中 SCI 操作的细节，参见第 12.3 节功能描述。

- 8 位或 9 位的数据模式
- 停止模式操作
- 轮询模式
- 单线模式

12.1.3 框图

图 12-2 为 SCI 的发送器部分。

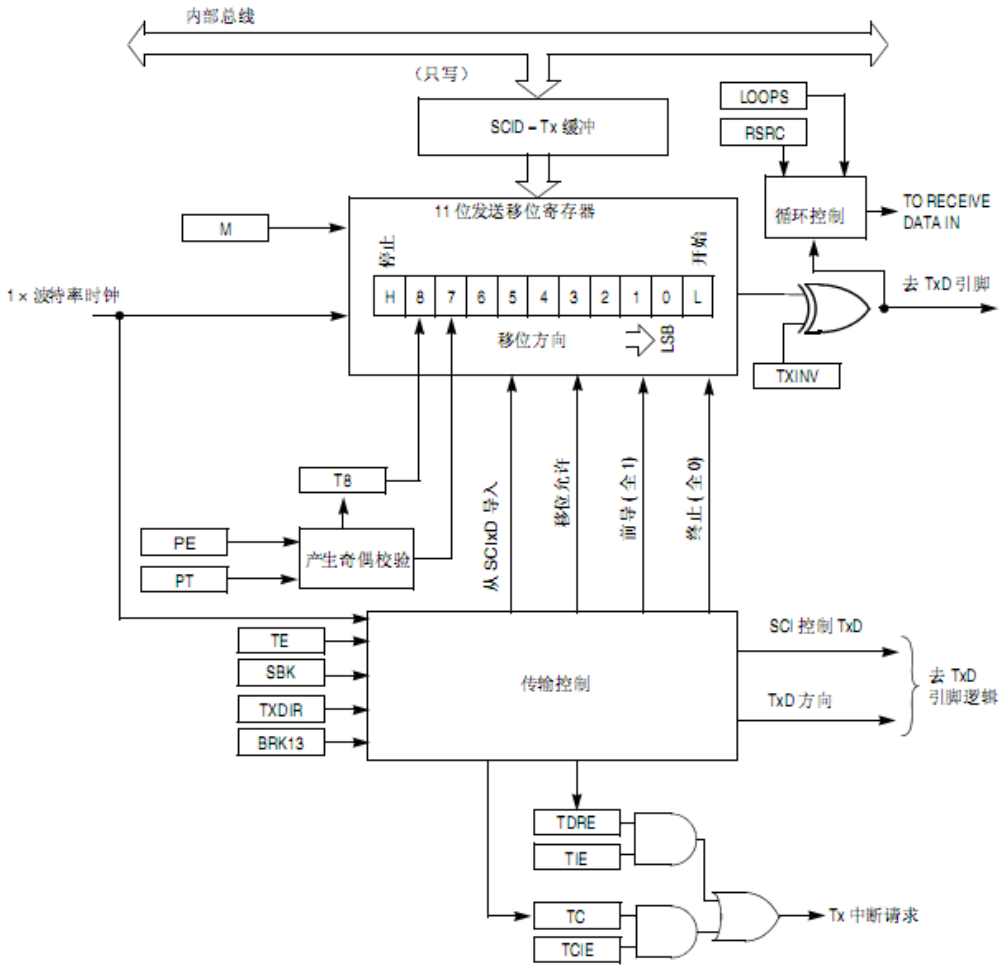


图12-2.SCI发送器框图

图 12-3 显示了 SCI 接收部分。

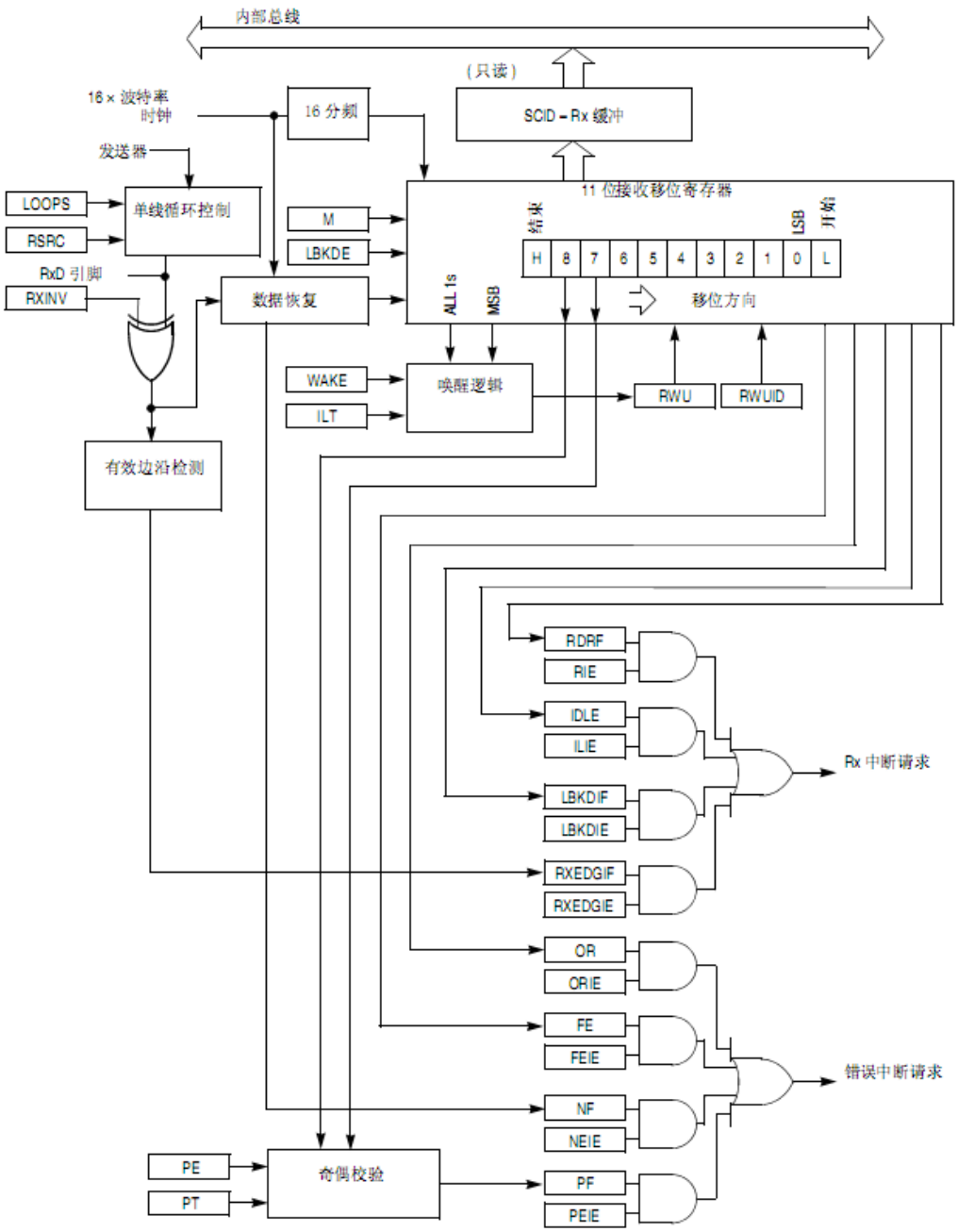


图12-3.SCI接收模块

12.2 寄存器定义

SCI 有 8 个 8 位寄存器，分别用于控制波特率、选择 SCI 选项、报告 SCI 状态和发送/接收数据。

SCI 寄存器的绝对地址参见本数据手册内存一章中的直接页寄存器的概述。这里通过这些寄存器和控制位的名称来引用它们。通常， Freescale 提供一个头文件把它们的名称翻译为绝对地址。

12.2.1 SCI 波特率寄存器 （SCIBDH,SCIBDL）

这两个寄存器控制产生 SCI 波特率的预分频因子。设置 13 位的波特率[SBR12:SBR0]，

先写新值的高半部分到 SCIBDH，然后写 SCIBDL。直到写完 SCIBDL，SCIBDH 的值，波特率才改变。

SICBDL 复位后的值非零，所以复位后波特率产生器仍然禁止，直到接收或发送第一次被允许（SCIC2 的 RE 或 TE 置 1）。

	7	6	5	4	3	2	1	0
R	LBKDIE	RXEDGIE	0	SBR12	SBR11	SBR10	SBR9	SBR8
W								
复位:	0	0	0	0	0	0	0	0

=未使用或者是保留的

图12-4.SCI波特率寄存器（SCIBDH）

表12-1.SCIBDH位域描述

位域	描述
7 LBKDIE	**LIN 中止检测中断使能 （对于 LBKDIF）** 0 禁止来自 LBKDIF 的硬件中断（使用轮询）。 1 当 LBKDIF 标志为 1 时，发送硬件中断请求。
6 RXEDGIE	**RxD 输入有效边中断使能 （对 RXEDGIF）** 0 禁止来自 RXEDGIF 的硬件中断 （使用轮询） 。 1 当 RXEDGIF 标志为 1 时，发送硬件中断请求。
4:0 SBR[12：8]	**波特率分配因子**——13 位的 SBR[12:0] 记作 BR，可以设置 SCI 波特率产生器的分频率。当 BR=0 时，SCI 波特率产生器被禁止以减少电路消耗。BR 可以从 1 到 8191，SCI 波特率 =BUSCLK/（16*BR）。其他的 BR 位请参见表 12-2。

	7	6	5	4	3	2	1	0
R	SBR7	SBR6	SBR5	SBR4	SBR3	SBR2	SBR1	SBR0
W								
复位:	0	0	0	0	0	0	0	0

图12-5.SCI波特率寄存器（SCIBDL）

表12-2.SCIBDL位域描述

位域	描述
7:0 SBR[7:0]	**波特率分配因子**——13 位的 SBR[12:0]记作 BR，可以设置 SCI 波特率产生器的分频率。当 BR=0 时，SCI 波特率产生器被禁止以减少电路消耗。BR 可以从 1 到 8191，SCI 波特率=BUSCLK/（16*BR）。其他的 BR 位请参考表 12-1。

12.2.2 SCI 控制寄存器 1（SCIC1）

可读写寄存器，用于控制多个 SCI 的可选功能。

	7	6	5	4	3	2	1	0
R	LOOPS	SCISWAI	RSRC	M	WAKE	ILT	PE	PT
W								
复位:	0	0	0	0	0	0	0	0

图12-6.SCI控制寄存器（SCIC1）

表12-3.SCIC1位域描述

位域	描述
7 LOOPS	**循环模式选择**——选择循环模式和正常的全双工模式。当 LOOPS=1 时，此时发送器的输出连接到接收器的输入。 0 正常操作 --- RxD 和 TxD 采用不同的引脚。 1 循环或者单线模式，此时发送器的输出连接到接收器的输入。
6 SCISWAI	**等待模式下 SCI 停止位** 0 在等待模式下， SCI 时钟继续运行，所以 SCI 可以作为唤醒 CPU 的中断源。 1 在等待模式下， SCI 时钟停止运行。
5 RSRC	**接收源选择**——仅在 LOOPS 位为 1 时有意义。当 LOOPS=1 时，接收器输入在内部连接到 TxD 引脚上，RSRC 决定引脚是否连接到发送器输出。 0 如果 LOOPS=1，RSRC=0 时，内部循环模式和 SCI 不使用 RxD 引脚。 1 单线 SCI 模式， TxD 引脚连接到发送器输出和接收器输入。
4 M	**9 位或 8 位模式选择** 0 正常——开始位 +8 位数据位 （首先传输最低有效位 LSB） +停止位。 1 接收和发送使用 9 位数据字符开始位+8 位数据位 （首先传输最低有效位 LSB）+9 位数据位 +停止位。
3 WAKE	**KE 接收器唤醒方式选择**——更多信息参见 12.3.3.2 节 接收唤醒。 0 空闲线唤醒。 1 地址符号唤醒。
2 ILT	**空闲线类型选择**——设置这个位为 1 保证字符后面的停止位和逻辑 1 的个数不超过 10 或 11 位 （空闲线检测逻辑需要的高电平） 。更多信息参见 12.3.3.2.1 节 空闲线唤醒 0 空闲字符位从 “开始位”开始计数。 1 空闲字符位从 “停止位”开始计数。
1 PE	**奇偶校验允许**——允许硬件奇偶校验产生和校验。当允许奇偶校验时，数据字符（第 8 或 9 位）的最高有效位（MSB）作为校验位。 0 不允许奇偶校验。 1 允许奇偶校验。
0 PT	**奇偶校验类型**——如果奇偶校验允许（PE=1） ，这个位选择奇校验或偶校验。奇校验是数据字符中 1 的总数（包括奇偶位）是奇数。偶校验是数据字符中 1 的总数 （包括奇偶位）是偶数。 0 偶校验。 1 奇校验。

12.2.3 SCI控制寄存器2（SCIC2）

本寄存器可以在任何时间被读和写。

7	6	5	4	3	2	1	0

R W	TIE	TCIE	RIE	ILIE	TE	RE	RWU	SBK
复位:	0	0	0	0	0	0	0	0

图12-7.SCI控制寄存器2（SCIC2）

表12-4.SCIC2位域描述

位域	描述
7 TIE	**发送中断允许 （用于 TDRE）** 0 TDRE 的硬件中断禁止 （使用轮询） 。 1 当 TDRE 标志位是 1，发送硬件中断请求。
6 TCIE	**发送完成中断允许 （用于 TC）** 0 TC 的硬件中断禁止 （使用轮询） 。 1 当 TC 标志位是 1，发送硬件中断请求。
5 RIE	**接收中断允许 （用于 RDRE）** 0 RDRE 的硬件中断禁止（使用轮询） 。 1 当 RDRE 标志位是 1，发送硬件中断请求。
4 ILIE	**空闲线中断允许 （用于 IDLE）** 0 IDLE 的硬件中断禁止（使用轮询）。 1 当 IDLE 标志位是 1，发送硬件中断请求。
3 TE	**发送允许** 0 发送器关。 1 发送器开。 使用 SCI 发送器则 TE 要置 1，当 TE=1 时， SCI 强制 TxD 引脚作为 SCI 系统的输出。当 SCI 设置位单线操作（LOOPS=RSRC=1），TXDIR 控制单个 SCI 传输线（TxD 引脚）的传输方向。TE=0 也可以表示一个空闲字符在排队，TE=1 则传输在处理中。更多细节参见 12.3.2.1 节 发送间隔和等待空闲。当 TE=0 时，在引脚变回通用 I/O 引脚前，发送器一直控制 TxD 引脚，直到有数据，等待空闲或等待间隔符排队完成传输。
2 RE	**接收允许**——当接收器关时， RxD 引脚变回通用 I/O 引脚。如果 LOOPS=1，即使 RE 等于 1RxD 也会变回通用 I/O 引脚。 0 接收关。 1 接收开。
1 RWU	**接收唤醒控制**——该位置 1， SCI 接收器进入待命状态，等待扫描到唤醒条件。唤醒条件是信息之间的空闲线（WAKE=0，空闲线唤醒）或者字符的最高有效位是逻辑 1（WAKE=1，地址符号唤醒） 。应用软件置位 RWU，（通常）硬件唤醒条件自动清零 RWU。更多信息参见 12.3.3.2 节 接收唤醒。 0 正常 SCI 接收操作。 1 待命的 SCI 接收器等待唤醒条件。
0 SBK	**发送终止**——向 SBK 写一个 1 然后一个 0，一个间隔符插入传输数据流。只要 SBK=1、10 或 11（13 或 14）个逻辑 0 的间隔符也会插入传输数据流。第二个间隔符可能在软件清 SBK 前产生，这依赖于 SBK 在正传输的信息时复位和清零的时间选择。更多信息参见 12.3.2.1 节 发送间隔和等待空闲。 0 正常发送操作。 1 发送对了终止字符。

12.2.4 SCI状态寄存器（SCIS1）

该寄存器有 8 个只读状态标志。写无影响。专门的软件序列（不是写该寄存器）用来清零这些状态位。

	7	6	5	4	3	2	1	0
R	TDRE	TC	RDRF	IDLE	OR	NF	FE	PF
W								
复位:	1	1	0	0	0	0	0	0

=未使用或者是保留

图12-8.SCI状态寄存器1（SCIS1）

表12-5.SCIS1位域描述

位域	描述
7 TDRE	**发送缓冲区空标志**——TDRE 在复位时置位。而且当发送数据从发送数据缓冲区传到发送移位寄存器为新字符留出空间时，TDRE 也置位。SCIS1 的 TDRE=1 然后写 SCI 数据寄存器可以清零 TDRE。 0 发送数据寄存器（缓冲区）满。 1 发送数据寄存器（缓冲区）空。
6 TC	**发送完成标志**——TC 在复位时置位。而且当 TDRE=1 并且没有数据，前导符或间隔符发送时， TC 也置位。 0 发送器忙 （发送数据，前导符或间隔符）。 1 发送器空闲。读 SCIS1 的 TC=1 然后做下面的 3 件事情之一，TC 就会自动清零。 • 写 SCI 数据寄存器 （SCID）发送新数据。 • 改变 TE 从 0 到 1 插入前导符。 • 置 1SCIC2 的 SBK 插入间隔符。
5 RDRF	**接收器满标志**——当字符从接受移位寄存器传输到接收数据寄存器（SCID）时， RDRF 置位。读 SCIS1 的 RDRF=1 然后读 SCI 数据寄存器（SCID）可以清 RDRF。 0 接收数据寄存器空。 1 接收数据寄存器满。
4 IDLE	**接收器空闲标志**——SCI 工作一段时间后，如果接收线空闲，IDLE 则置位。当 ILT=0 时，接收器开始计数开始位后的空闲时间。所以如果接收字符全 1，这些位的时间和停止位的时间达到了一个字符的逻辑高（10 或 11 位依赖于 M 控制位），这会导致接收器认为检测到空闲，当 ILT=1 时，接收器直到停止位后才开始数空闲时间。所以停止位和先前字符的逻辑高不会到达一个字符逻辑高的时间，从而不会被检测为空闲线。读 SCIS1 的 IDLE=1 然后读 SCI 数据寄存器（SCID）可以清零 IDLE。IDLE 清零后，直到接收到一个新的字符并且 RDRF 被置位，IDLE 才能再次置位。即使接收线长时间保持空闲，IDLE 也只会被置位一次。 0 未检测到空闲线。 1 检测到空闲线。
3 OR	**接收器溢出标志**——当新的字符准备传到接收数据寄存器，而前面接收的数据还没有从 SCID 中读走，OR 置位。在这种情况下，

	因为没有空间把这些数据移到 SCID 中，所以新的字符 （和所有的相关错误信息）丢失。读 SCIS1 的 OR=1 然后读 SCI 数据寄存器（SCID）可以清零 OR。 0 未溢出。 1 接收溢出 （新数据丢失）。
2 NF	**噪音标志位**——在接受器中使用先进的采样技术，开始位采样 7 次，每个数据位和停止位采样 3 次。在接收到数据时，RDRF 置位的同时，如果在帧中某个采样和其他的采样不同，则标志 NF 置位。读 SCIS1 然后读 SCI 数据寄存器（SCID）可以清零 NF。 0 未检测到噪音。 1 SCID 中接收到的字符检测到噪音。
1 FE	**帧错误标志**——接收器在停止位检测到逻辑 0 时，FE 和 RDRF 置位。这表明接收器没有对齐字符帧。读 SCIS1 的 FE=1 然后读 SCI 数据寄存器（SCID）可以清零 FE。 0 未检测到帧错误。这并不确保帧时正确的。 1 帧错误。
0 PF	**奇偶错误标志**——当奇偶校验使能（PE=1）并且接收到的数据中的校验位和正确的校验位不同，PE 和 RDRF 置位。读 SCIS1 然后读 SCI 数据寄存器 （SCID）可以清零 PF。 0 无奇偶校验错误。 1 有奇偶校验错误。

12.2.5 SCI状态寄存器2（SCIS2）

本寄存器只有一个只读状态位。

	7	6	5	4	3	2	1	0
R	LBKDIF	RXEDGIF	0	RXINV	RWUID	BRK13	LBKDE	RAF
W								
复位:	0	0	0	0	0	0	0	0

=未使用或者是保留

图12-9.SCI状态寄存器2（SCIS2）

表12-6.SCIS2位域描述

位域	描述
7 LBKDIF	**LIN 终止检测中断标志**——当使能 LIN 间隔检测电路，并且检测到一个 LIN 间隔符，LBKDIF 置位。向 LBKDIF 写 1，LBKDIF 清零。 0 未检测到 LIN 间隔符。 1 检测到 LIN 间隔符。
6 RXEDGIF	**RxD 引脚有效边沿中断标志**——当 RxD 引脚上一个有效边沿发生时（如果 RXINV=0，下降沿，如果 RXINV=1，上升沿） RXEDGIF 置位。向其写 1，RXEDGIF 清零。 0 在接收引脚上无有效边沿发生。 1 在接收引脚上有效边沿发生。
4	**接收数据反转**——该位置位则接受的数据输入的极性反转。

RXINV[1]	0 接收的数据未反转。 1 接收的数据反转。
3 RWUID	**接收唤醒空闲检测**——RWUID 控制空闲字符是否置位 IDLE 位。 0 在接收待命状态（RWU=1）期间，检测到空闲字符时， IDLE 不置位。 1 在接收待命状态（RWU=1）期间，检测到空闲字符时， IDLE 置位。
2 BRK13	**间隔符长度**——BRK13 可以选择更长的发送间隔符长度。该位的状态不影响帧错误的检测 0 间隔符长度是 10 字节 （M=1 则为 11） 。 1 间隔符长度是 13 字节 （M=1 则为 14） 。
1 LBKDE	**LIN 间隔检测使能**——LBKDE 可以选择更长的间隔符检测长度。当 LBKDE 置位时，可以防止帧错误 （FE）和接收数据寄存器满（RDRF）标志置位。 0 间隔符的检测长度是 10 字节 （M=1 则为 11） 。 1 间隔符的检测长度是 11 字节 （M=1 则为 12） 。
0 RAF	**接收器有效标志**——当 SCI 接收器检测到有效位的开始，RAF 置位；当接收器检测到空闲线，RAF 清零。该状态位可以用来检查 MCU 进入停止模式前， SCI 字符是否正被接收。 0 SCI 接收器空闲等待开始位。 1 SCI 处于活动中（RxD 输入无效）。

[1]置位 RXINV 反转 RxD 输入的各种情况：数据位，开始位和停止位，终止和空闲。

当在 LIN 系统中使用内部振荡器时，有必要把间断检测极限值增加一位。在最坏的情况下，LIN 允许的的定时条件可能把 0x00 数据字符看做 10.26 位字节，因为在从机上运行比主机快 14%。这将触发正常的间隔检测电路 （用来检测 10 位间隔标志） 。当 LBKDE 位置位时，帧错误可以减少，间隔检测阈从 10 位变成 11 位以防止把 0x00 数据字符错误地检测成 LIN 间隔符号。

12.2.6 SCI控制寄存器3（SCIC3）

	7	6	5	4	3	2	1	0
R	R8	T8	TXDIR	TXINV	ORIE	NEIE	FEIE	PEIE
W								
复位:	0	0	0	0	0	0	0	0

=未使用或者是保留

图12-10.SCI控制寄存器3（SCIC3）

表12-7.SCIC3位域描述

位域	描述
7 R8	**接收器的第九数据位**——当 SCI 设置为 9 位数据时 （M=1），R8 可以看作 SCID 寄存器中数据的最高有效位左边的第九位接收数据。当读 9 位数据时，因为读 SCID 会自动完成清零标志 （允许在 R8 和 SCID 中写入新数据），在读 SCID 之前先读 R8。
6 T8	**发送器的第九数据位**——当 SCI 设置为 9 位数据时 （M=1），T8 可以看作 SCID 寄存器中数据的最高有效位左边的第九位发送数据。当写 9 位数据时，在写 SCID 之后整个 9 位值传到 SCI 移位寄存器，所以 T8 应该在写 SCID 之前先写 （如果 T8 需要改变它的值） 。如果 T8 不需要改变值（比如当它用于产生标记或空间奇偶校验）则不需要每次写 SCID

	都重写。
5 TXDIR	**单线模式 TxD 引脚方向**——当 SCI 被设置为单线半双工操作时（LOOPS=RSRC=1），该位决定 TxD 引脚上的数据方向。 0 单线模式下 TxD 引脚为输入。 1 单线模式下 TxD 引脚位输出。
4 TXINV[1]	**发送数据反转**——该位置位则发送数据的输出的极性反转。 0 发送数据未反转。 1 发送数据反转。
3 ORIE	**溢出中断使能**——该位使能溢出标志 （OR），产生硬件中断请求。 0 禁止 OR 中断（使用轮询）。 1 当 OR=1 时，发出硬件中断请求。
2 NEIE	**噪音错误中断使能**——该位使能溢出标志 （NF），产生硬件中断请求。 0 禁止 NF 中断（使用轮询）。 1 当 NF=1 时，发出硬件中断请求
1 FEIE	**帧错误中断使能**——该位使能帧错误标志 （FE），产生硬件中断请求。 0 禁止 FE 中断（使用轮询）。 1 当 FE=1 时，发出硬件中断请求
0 PEIE	**奇偶校验错误中断使能**——该位使能奇偶校验错误标志 （PF），产生硬件中断请求。 0 禁止 PF 中断（使用轮询）。 1 当 PF=1 时，发出硬件中断请求

[1]置位 TXINV 反转 TxD 输出的各种情况：数据位，开始位和停止位，终止和空闲。

12.2.7 SCI数据寄存器（SCID）

该寄存器实际上是两个独立的寄存器。读操作读到只读接收数据缓冲区的内容，写操作写入发送数据缓冲区。该寄存器的读和写也和 SCI 状态标志的标识自动清零机制有关。

	7	6	5	4	3	2	1	0
R	R7	R6	R5	R4	R3	R2	R1	R0
W	T7	T6	T5	T4	T3	T2	T1	T0
复位:	0	0	0	0	0	0	0	0

图12-11.SCI数据寄存器（SCID）

12.3功能描述

SCI 是一种全双工，异步， NRZ 的串行通信。用于 MCU 和其他远程设备 （包括其他 MCU）通信。 SCI 包括一个波特率产生器，发送器和接收器。虽然使用相同的波特率产生器，发送器和接收器独立工作。正常工作时， MCU 监控 SCI 的状态，写数据发送，处理接收的数据。接下来描述每个 SCI 模块。

12.3.1 波特率发生

如图 12-12 所示 SCI 波特率产生器的时钟源为总线速率时钟。

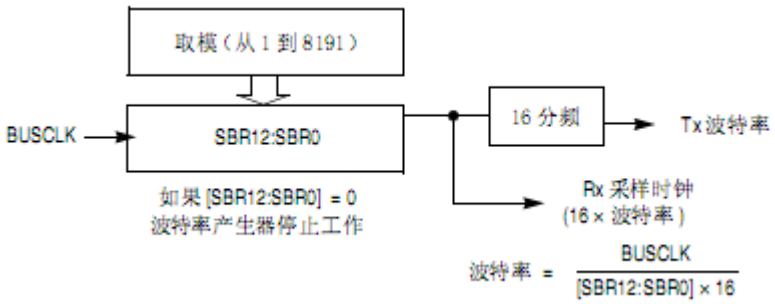


图12-12.SCI波特率产生

SCI 通信要求发送器和接收器（通常从独立的时钟源获得波特率）使用相同的波特率。所允许的波特频率由两个部分决定，接收器怎样同步开始位的有效沿和位采样怎样操作。

MCU 在每个高到低转换时都会重新同步位边界。但是在最坏的情况下，在 10 或 11 位的字符帧中没有这种转换，所以在整个字符期间任何波特率的不匹配都会积累起来。在飞思卡尔半导体的 SCI 系统中，其总线频率有一个晶振得到，所允许的波特率不匹配是 4.5%（8 位数据格式）和 4% （9 位数据格式） 。虽然波特率分频因子的设置并不总是产生正好匹配标准的波特率，但是通常偏差都在很小的百分比，可以保障可靠的通信。

12.3.2 发送功能描述

本节描述 SCI 接收器的框图和发送终止和空闲字符的功能。发送器的框图图 12-2 所示。

发送器输出 （TxD）空闲状态默认是逻辑高 （复位后 TXINV=0） 。设置 TXINV=1，发送器输出被反转。置位 SCIC2 的 TE，发送器使能。发送器发出一个前导符（一个空闲状态的满字符帧）。直到发送数据缓冲区的数据准备好，发送器保持空闲。写 SCI 数据寄存器 （SCID）可以将数据存到发送数据缓冲区。

SCI 发送器的主要部分是发送移位寄存器， 10 位或 11 位长 （由 M 控制位决定） 。在本节以后的部分，我们假定 M=0（选择正常的 8 位数据模式） 。在 8 位数据模式下，移位寄存器包含 1 个开始位，8 个数据位，1 个停止位。当发送移位寄存器允许接受新的 SCI 字符时，等待在发送数据寄存器的值传到移位寄存器 （由波特率时钟同步）并且发送数据空 （TDRE）状态标志置位以表示 SCID 内的发送数据缓冲区可以写入另一个字符。

TxD 引脚发出停止位后，如果没有数据等待发送数据缓冲区，发送器置位发送完成标志并进入空闲状态，TxD 逻辑高，等待传输字符。

写 0 到 TE 并不能立即使引脚成为通用 I/O 引脚，必须先完成正在进行的发送过程。这包括正在处理的数据，等待空闲字符和等待间隔符。

12.3.2.1发送间隔和等待空闲

SCIC2 的 SBK 用于发送间隔符 （原来用于与电传打字机进行通信初始化）。间隔符是一个字符的逻辑 0（10 位包含开始位和停止位） 。置位 BRK13=1，则间隔符可以达到 13 位的长度。通常，程序等待 TDRE 置位（表示信息的最后一个字符移到发送移位寄存器），然后向 SBK 位先后写 1 和 0。这样的话，只要移位寄存器可用，就会发送间隔符。如果当间隔符进入移位寄存器 （由波特率时钟同步） SBK 仍为 1，则有另外一个间隔符在等待。如果接收设备是另一个飞思卡尔半导体的 SCI，则接受的 8 位间隔符全 0，并且产生帧错误（FE=1） 。

当使用空闲线唤醒时，两个信息之间需要一个空闲 （逻辑 1）的字符，以唤醒处于睡

眠状态的接收器。通常，程序等待 TDRE 置位（表示信息的最后一个字符移到发送移位寄存器），然后向 TE 位先后写 0 和 1。这样的话，只要移位寄存器可用，就会发送空闲符。只要移位寄存器的字符没有完成 （TE=0） ， SCI 发送器就不会释放对 TxD 引脚的控制。如果移位寄存器有完成的可能性 （TE=0） ，设置为通用 I/O 控制，则引脚和 TxD 复用，输出逻辑 1。这确保了 TxD 线看起来像正常的空闲线，即使 SCI 在向 TE 写 0 和写 1 之间失去对

引脚的控制。

间隔符的长度受 BRK13 和 M 位影响，如表 12-8 所示。

表12-8.间隔符长度

BRK13	M	间隔位长度
0	0	10 位
0	1	11 位
1	0	13 位
1	1	14 位

12.3.3 接收功能描述

在该节中，接收器框图 （图 12-3）是全面描述接收器的功能一个指南。然后，更详细地描述数据采样技术 （用于得到接收器的数据） 。最后，解释了两种接收器唤醒功能。

通过置位 RXINV=1，接收器输入被反转。置位 SCIC2 的 RE，接收器使能。字符帧包括一个开始位（逻辑 0） ， 8（或 9）数据位 （首先最低有效位），和停止位 （逻辑 1） 。关于 9 位数据模式的信息，参见 12.3.5.1 节 8 位和 9 位数据模式。在以后的讨论中，我们假定 SCI 设置为正常的 8 位数据模式。

接受移位寄存器接收到停止位后，如果接收数据还未满，数据字符传到接收数据寄存器并且接收数据寄存器满 （RDRF）状态标志置位。如果 RDRF 已经置位表明接收数据寄存器 （缓冲区）已经满了，溢出（OR） ，状态标志置位并且新数据丢失。因为 SCI 接收器时双缓冲，在 RDRF 置位后和读接收数据缓冲区的数据前，程序有一个字符的时间避免接收器溢出。

当程序检测到接收数据寄存器满时 （RDRF=1），读 SCID 可以得到接收数据寄存器的数据。 RDRF 通过两步骤的序列（通常在控制接收数据的用户程序中）自动清零，更多关于标志清零的信息参见 12.3.4 节 中断和状态标志。

12.3.3.1 数据采样技术

SCI 接收器使用 16 倍的波特率时钟进行采样。接收器以 16 倍波特率采样逻辑电平，发现 RxD 串行输入引脚上的下降沿。下降沿被定义为 3 个逻辑 1 采样后的逻辑 0 采样。16 倍波特率时钟把位时间分成 16 段（从 RT1 到 RT16）。当定位了一个下降沿时，有 3 个采样（RT3、RT5、 RT7）确保这是开始位而不是噪音。如果 3 个采样中不少于 2 个是 0，则接收器假定与接收字符同步。

然后接收器采样每个位时间，包括开始位和停止位（在 RT8、RT9、RT10 决定改为的电平） 。逻辑电平由位时间的采样结果的多数采样决定。采样开始位时，如果 RT3、RT5、RT7 这三个中至少两个是 0，则可以认为该位为 0，即使 RT8、RT9、RT10 中的一个或全部为 1。如果在字符帧中任何一个位时间（包括开始位和停止位）的任何一个采样无法得到一致的逻辑电平，当接收的字符传到接收数据缓冲区时，噪音标志（NF）置位。

下降沿检测逻辑不断检测下降沿，如果检测到了，采样时钟同步到位时间。这提高了接收器在噪音和波特率不匹存在时的可靠性。这不能提高最糟情况的分析，因为有一些字符在

字符帧中都不会出现下降沿。

在帧错误的情况下，如果接收的字符不是间隔符，查找下降沿的采样逻辑填充了 3 个逻辑 1，如此以至于几乎可以立即检测一个新的开始位。

在帧错误的情况下，直到帧错误清零，接收器都会被阻止接收新的字符。接受移位寄存器继续工作，但是如果 FE 置位，一个完成的字符不能转移到接收数据缓冲区。

12.3.3.2 接收唤醒

接收器唤醒是一种硬件机制，它允许 SCI 接收器忽略那些供不同 SCI 接收器使用的信息中的字符。在这个系统中，所有的接收器判断每个信息的第一个字符，只要认为信息是提供给不同 SCI 的，则将 SCIC2 中的接受唤醒 （RWU）置 1。当 RWU 置位，和接收器相关的状态（除了空闲位 （IDLE） ，当 RWUID 位置位）禁止置位，因此消除控制不重要字符的软件开支。在信息的最后，或者是下一个信息的开始，所有的接收器自动强制 RWU 为 0，所以所有的接收器按时唤醒，查看下一个信息的第一个字符。

12.3.3.2.1 空闲线唤醒

当 WAKE=0，接收器被设置为空闲线唤醒。在这种模式中，当接收器检测到一个满字符的空闲线电平时， RWU 自动清零。 M 控制位选择 8 位或 9 位数据模式，这决定了多少个空闲位组成了一个满字符时间（加上开始位和停止位共 10 位或 11 位）。

当 RWU 为 1 和 RWUIN 为 0 时，唤醒接收器的空闲条件不置位 IDLE。接收器唤醒并等待下一个信息的第一个字符 （将会置位 RDRF，如果允许可以产生一个中断），当 RWUID 为 1 时，无论 RWU 是 0 还是 1，任何空闲条件置位 IDLE 标志并产生一个中断（如果允许） 。

空闲线类型 （ILT）控制位选择两种检测空闲线方法中的一种。 ILT=0，空闲位从开始位开始计数，停止位和字符后面的逻辑 1 的个数组成了空闲线满字符。 ILT=1，空闲位不是从开始位而是停止位，所以空闲线检测不受前一个信息的最后的字符的数据的影响。

12.3.3.2.2 地址标志唤醒

当 WAKE=1，接收器被设置为地址标志唤醒。在这种模式中，当接收器检测到接收数据的大多数有意义位是逻辑 1 时（M=0，第 8 位； M=1，第 9 位） ， RWU 自动清零。

地址标识唤醒允许信息包含空闲位，但是需要 MSB 保留为地址帧使用。在接收到停止位并且置位 RDRF 标志前，地址帧的 MSB 逻辑 1 清零 RWU 位。在这种情况下，即使接收器在大多数字符时间都在休眠，MSB 字符仍被接收。

12.3.4 中断和状态标志

SCI 系统有三个独立的中断向量，减少了分析中断原因的软件编程。一个中断向量和发送器的 TDRE 和 TC 事件相关。另一个中断向量和接收器的 RDRF、IDLE、RXEDGIF 和 LBKDIF 事件相关。第三个向量用于 OR、 NF、 FE 和 PD 错误条件。通过本地中断使能掩码，这十个中断源可以独立掩码。当本地掩码被清零，禁止中断请求的产生时，这个标志可以通过软件轮询的方式得到。

SCI 发送器有两个状态标志，可以产生硬件中断请求。发送数据寄存器空 （TDRE）表示什么时候发送数据缓冲区有空间可以写另一个发送字符到 SCID。如果发送中断使能（TIE）位置位，只要 TDRE=1，就会产生一个硬件中断请求。发送完成 （TC）表示所有的数据，前导符和间隔符都发送完成，发送器空闲 （TxD 处于无效电平） 。该标志位通常用于有调制解调器的系统，用于决定何时关闭调制解调器是安全的。如果发送完成中断使能位 （TCIE）置位，只要 TC=1，就会产生一个硬件中断请求。如果 TIE 或 TCIE 本地中断掩码是 0，软件轮询可以代替硬件中断请求，可以用于检测 TDRE 和 TC 状态位。

当程序检测到接收数据寄存器满 （RDRF=1），程序读 SCID 可以得到接收数据寄存器的数据。当 RDRF=1 并且读了 SCID 后，读 SCIS1 可以清零 RDRF 标志。

当使用轮询方式时，这个顺序可以很自然的满足用户程序的一般过程。如果使用硬件中断方式， SCIS1 必须在中断服务例程 （ISR）中读。通常这必须在 ISR 中完成已检测接收错误，所以这个顺序可以自动满足。

当 RxD 线在一段时间保持空闲， IDLE 状态标志包含了可以避免重复置位的情况。当 IDLE=1 并且读了 SCID 后，读 SCIS1 可以清零 IDLE。IDLE 清零后，直到接收器至少接收了一个新字符并且 RDRF 置位，才能再次置位。

如果在导致 RDRF 置位的接收字符中检测到了相应的错误，错误标志——噪音标志（NF），帧错误（FE）和奇偶校验错误标志（PF）将会和 RDRF 同时置位。在溢出情况，这些标志不置位。

如果 RDRF 置位，当一个新字符准备从接收移位寄存器到接收数据缓冲区，溢出标志（OR）置位，和数据相关的 NF、 FE 或 PF 条件丢失。

在任何时候， RxD 串行数据输入引脚上的有效边沿会导致 RXEDGIF 标志置位。向 RXEDGIF 标志写 1，可以清零 RXEDGIF。这个功能不依赖于接收器被允许 （RE=1） 。

12.3.5 其他 SCI功能

下面描述其他的 SCI 功能。

11.3.5.1 8 位和9 位数据模式

通过置位 SCIC1 的 M 控制位， SCI 系统 （发送器和接收器）可以被设置为 9 位数据模式。在 9 位数据模式中，第九位数据位在 SCI 数据寄存器的最高有效位的左边。对于发送数据缓冲区，该位存储在 SCIC3 的 T8。对于接收器，该位存储在 SCIC3 的 T8。

为了一致地写发送数据缓冲区，在写 SCID 以前写 T8 位。

如果要发送的新字符的第九位的值和前一个字符相同，没有必要再写 T8。当数据从发送数据缓冲区传到发送移位寄存器中时，在数据从 SCID 传到移位寄存器的同时， T8 的值被复制。

9 位数据模式通常用于连接奇偶校验位，可以允许 8 位的数据加上第九位的奇偶校验位。或用于地址标志唤醒，所以第九位也可以作为唤醒位。在定制协议中，第九位也作为一个由软件控制的掩码器。

12.3.5.2 停止模式操作

在所有的停止模式中， SCI 模块的时钟暂停。

在 stop1 和 stop2 模式中，所有的 SCI 寄存器数据丢失。当从这两种停止模式中恢复时，所有的寄存器必须重新初始化。在 stop3 模式中， SCI 模块寄存器不会受影响。

在 stop3 模式中，接收输入有效边沿检测电路仍然有效，但是在 stop2 模式下无效。如果中断没有被屏蔽（RXEDGIE=1） ，接收输入上的一个可以将 CPU 从 stop3 模式中唤醒。

注意，因为时钟暂停，当从停止模式 （仅在 stop3 模式中）中退出，SCI 模块将重新开始操作。当有字符正在被发送出或接收进 SCI 模块，软件应该确保不进入停止模式。

12.3.5.3 循环模式

当 LOOPS=1 时，同一个寄存器的 RSRC 位选择循环模式 （RSRC=0）或单线模式（RSRC=1）。循环模式有时用于检测软件，独立于外部系统的连接，可以帮助分析系统问题。在这个模式中，发送器的输出在内部连接到接收器的输入， RxD 不使用，所以它作为

通用 I/O 引脚。

12.3.5.4 单线操作

当 LOOPS=1 时，同一个寄存器的 RSRC 位选择循环模式 （RSRC=0）或单线模式（RSRC=1）。单线模式半双工的串行连接。接收器在内部连接到发送器的输出 TxD 引脚。RxD 引脚不使用，作为通用 I/O 引脚。

在单线模式中， SCIC3 的 TXDIR 位控制 TxD 引脚上数据的方向。当 TXDIR=0， TxD 引脚作为接收器的输入并且发送器和 TxD 引脚临时断开，所以一个外部设备可以向接收器发送数据。当 TXDIR=1，TxD 作为接收器的输出。在单线模式中，从发送器到接收器的内部环路使接收器接收发送器所发送的字符。

第13章 16位串行外设接口（S08SPI16V1）

13.1 简介

8 或 16 位串行外设接口模块（SPI）为 MCU 与外设提供全双工的、同步的、串行通信。这些外设可以拥有其它微控制器、模数转换器、移位寄存器、传感器、内存等。

SPI 在主机模式下波特率为总线时钟频率的 1/2 而在从机模式下工作频率有总线的 1/4。软件可以查看状态标志或者 SPI 操作可以中断驱动。

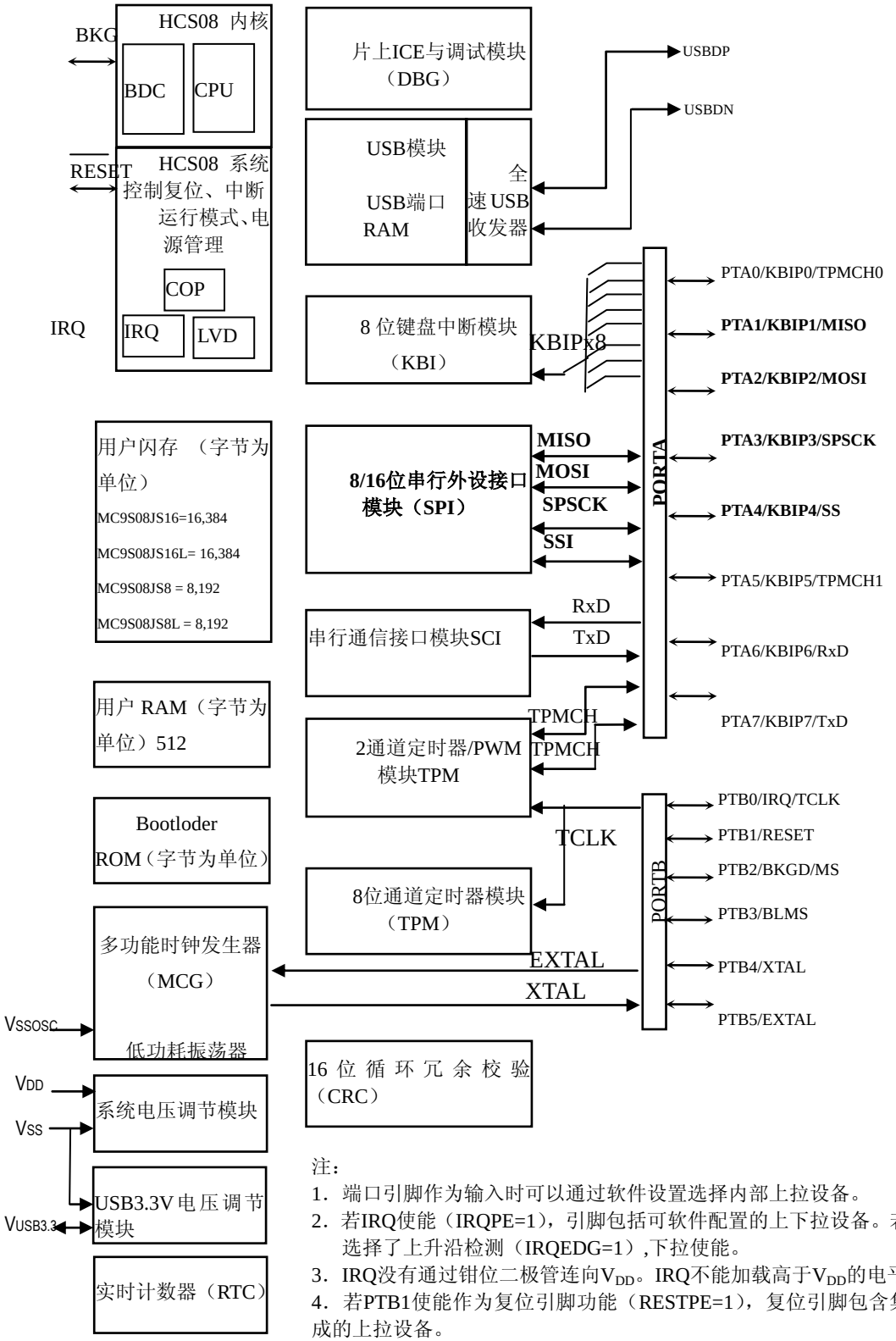
SPI 支持 8 或 16 位且有为接收缓存匹配硬件的特性。

MC9S08JS16 有 1 个串行外设接口模块（SPI）。其相关引脚为 PTA[4:1]。欲了解 SPI 电器参数可以查看”MC9S08JS16 系列数据手册”。

13.1.1 SPI端口配置信息

当配置 SPI 波特率大于 6MHz，输入滤波器应该禁止（清 SOPT2 的 SPIFE）并且使能相关高输出驱动强度选项的 SPI 引脚。由此可以使数据传送在一个较高的频率但这会有增加噪声的风险。如果 SPI 运行在 8MHz 波特率或者更高，应用必须使能高输出驱动增强，这个由 SPI 相关的引脚来选择。

默认情况下，SPI 端口引脚的输入滤波器将会使能（SPIFE=1），这使得 SPI 的波特率在 6MHz，但是消除了噪声干扰。



注：

1. 端口引脚作为输入时可以通过软件设置选择内部上拉设备。
2. 若IRQ使能（IRQPE=1），引脚包括可软件配置的上下拉设备。若选择了上升沿检测（IRQEDG=1）,下拉使能。
3. IRQ没有通过钳位二极管连向V_{DD}。IRQ不能加载高于V_{DD}的电平
4. 若PTB1使能作为复位引脚功能（RESTPE=1），复位引脚包含集成的上拉设备。
5. 引脚包含集成上拉设备。
6. 当KBI使能（KBIPEn=1）而且相关引脚被配置为上拉设备使能，那么上升沿（KBEDGn=1）被用于将上拉设备配置为下拉设备。

图 13-1. MC9S08JS16系列框图SPI模块和引脚高亮显示

Module Initialization (Slave):

Write:	SPIxC1	to configure	interrupts, set primary SPI options, slave mode select, and system enable.
Write:	SPIxC2	to configure	optional SPI features, hardware match interrupt enable, and 8- or 16-bit data transmission length
Write:	SPIxMH:SPIxML	to set	hardware compare value that triggers SPMF (optional) when value in receive data buffer equals this value.

Module Initialization (Master):

Write:	SPIxC1	to configure	interrupts, set primary SPI options, master mode select, and system enable.
Write:	SPIxC2	to configure	optional SPI features, hardware match interrupt enable, and 8- or 16-bit data transmission length
Write:	SPIxBR	to set	baud rate
Write:	SPIxMH:SPIxML	to set	hardware compare value that triggers SPMF (optional) when value in receive data buffer equals this value.

Module Use:

After SPI master initiates transfer by checking that SPTEF = 1 and then writing data to SPIDH/L:

Wait for SPRF, then read from SPIDH/L
Wait for SPTEF, then write to SPIDH/L

Data transmissions can be 8- or 16-bits long, and mode fault detection can be enabled for master mode in cases where more than one SPI device might become a master at the same time. Also, some applications may utilize the receive data buffer hardware match feature to trigger specific actions, such as when command data can be sent through the SPI or to indicate the end of an SPI transmission.

SPIxC1	SPIE	SPE	SPTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
	Module/interrupt enables and configuration							
SPIxC2	SPMIE	SPIMODE		MODFEN	BIDIROE		SPISWAI	SPC0
	Additional configuration options.							
SPIxBR		SPPR2	SPPR1	SPPR0		SPR2	SPR1	SPR0
	Baud rate = (BUSCLK/SPPR[2:0])/SPR2[2:0]							
SPIxDH	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
SPIxDL	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPIxMH	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
SPIxML	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	Hardware Match Value							
SPIxS	SPRF	SPMF	SPTEF	MODF				

Figure 15-2. SPI Module Quick Start

图13-2. SPI模块快速启用

13.1.2 特性

SPI 模块的特性包括：

主或辅模式运行

全双工或单线双向选项

可编程发送波特率

双缓冲发送和接收

串行时钟相位和极性选项

辅选择输出

CPU 中断下的模式错误标志
等待模式下的 SPI 操作控制
可选择的 MSB 在先或 LSB 在先转换
可编程的 8 或 16 位数据传输长度
接收缓存的硬件匹配特性

13.1.2 结构图

SPI 有三个功能模式：运行模、等待模式、停止模式。
等待模式：
此模式为基本模式
等待模式：
操作在此模式下时，SPI 配置在低功耗模式，由 SPIC2 寄存器 的 SPISWA 位 控制。在此模式下，当 SPISWA 位被清零，那么此时就和 RUN 模式下一样。如果 SPISWA 被置位，SPI 就会进入保守状态，并且 SPI 时钟周期性的关闭。如果 SPI 工作在主机模式下，所有的传输将停止，但是当 CPU 进入 RUN 模式时就会被唤醒。如果 SPI 被配置成从机，将继续接收传输一个字节，这样松鸡保持和主机的同步。
停止模式
为了减少能耗 SPI 进入 stop3 模式下时将不被激活。如果 SPI 被配置成一个主机，所有的传输将停止，但是当 CPU 进入 RUN 模式时候又会被恢复。如果 SPI 被配置成从机，将继续接收或发送有一个数据，这样从机就能保持与主机的同步。
当在其它停止模式的时候 SPI 是完全禁止的。当 CPU 从这些模式下被唤醒时所有的寄存器将会被重置。
这些只是从宏观上描述，更加详细的操作模式在 13.4.9 部分的“低功耗模式”会讲到。

13.1.3 SPI结构图

这部分包含了展示 SPI 系统连接、内部模块组织、控制主机模式速率的 SPI 时钟发生器的结构图。

13.1.4.1 SPI系统结构图

图 13-2 所示为两个以主从分布方式连接的 MCU 构成的 SPI 模块。主器件启动所有的 SPI 数据传输。在一次传输过程中，主器件把数据移出（在 MOSI 管脚）到从器件同时从从器件传送数据进入主器件（在 MISO 管脚）。这种传输有效的交换了两个 SPI 系统中 SPI 移位寄存器里面的数据。SPSCK 信号是从主器件的一个时钟输出，也是从器件的时钟输入。从器件必须通过一个从选择输入管脚（$\overline{SS}$ 管脚）上的低电平选中。在这个系统中，主器件已经配置它的$\overline{SS}$管脚为一个可选的从选择输出。

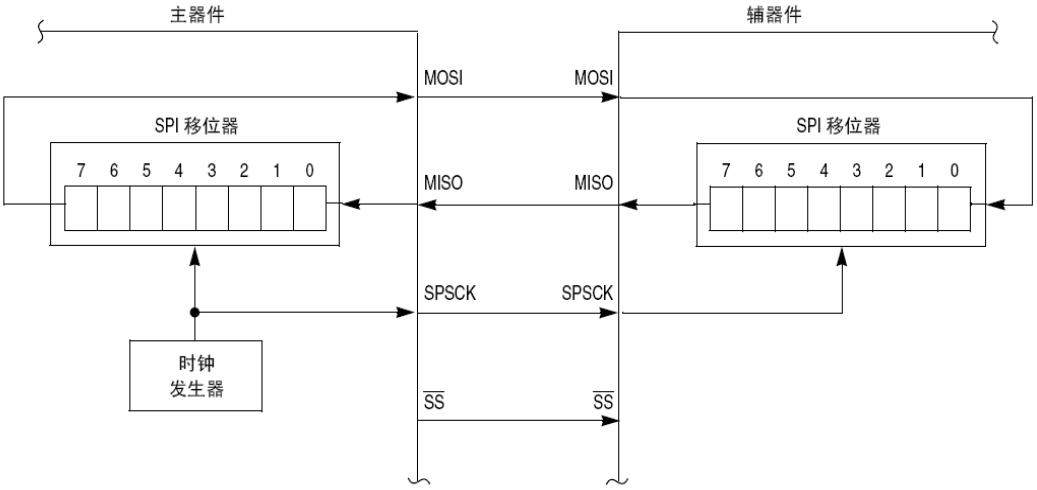


图13-3. SPI 系统连接

SPI 系统最常见的使用包括连接简单移位寄存器，以增加输入或输出端口，或者连接小型外围器件，如串行 A/D 或 D/A 移位器。尽管图 13-2 显示了一个在两 MCU 间交换数据的系统，但许多实际的系统的连接更简单，数据要么从主 MCU 单向传输到从 MCU，要么从从 MCU 单向传输到主 MCU。

13.1.4.2 SPI模块结构图

图 13-4 是 SPI 模块的结构图。SPI 的中心元件是 SPI 移位寄存器。数据写入双缓冲发送器（写入 SPID），然后转移到位于数据传输起点的 SPI 移位寄存器。在数据字节中转换后，数据被传输到双缓冲接收器，在这里数据可以被读取（从 SPID 读取）。管脚复用逻辑控制着 MCU 管脚和 SPI 模块间的连接。

当 SPI 配置为主 SPI 时，时钟输出被路由到 SPSCK 管脚，移位器输出被路由到 MOSI，移位器输入从 MISO 管脚路由出来。

当 SPI 配置为从 SPI 时， SPSCK 管脚为时钟输出。MOSI 为移位输出，MISO 管脚为移位器输入。

在外部 SPI 系统，只需将所有 SPSCK 管脚彼此连接，所有 MISO 管脚连接起来，所有 MOSI 管脚连接起来就可以来。外围器件通常为这些管脚使用略有不同的名称。

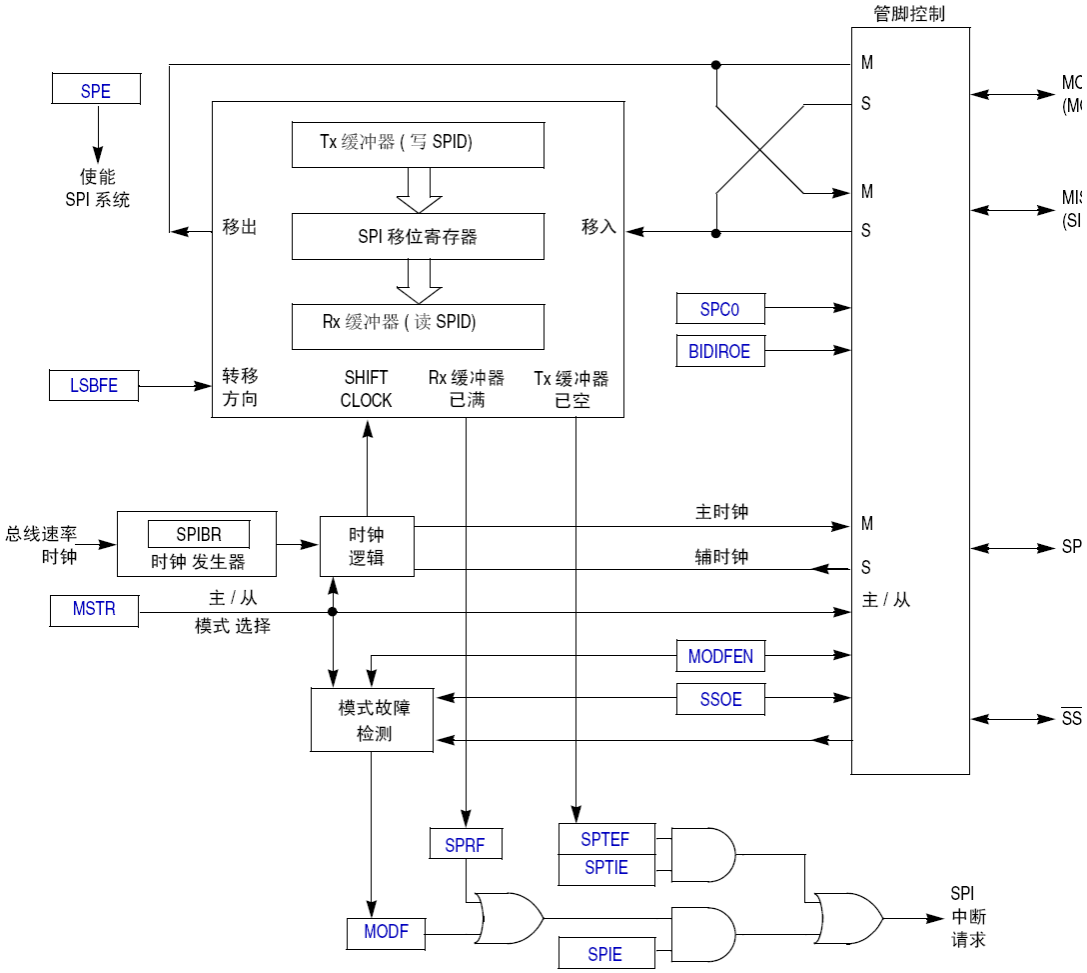


图13-4. SPI 模块结构图

13.2 外部信号描述

SPI 共享 4 个端口管脚。这些管脚的功能取决于 SPI 控制位的设置。当 SPI 禁止（SPE = 0）时，这 4 个管脚恢复为不受 SPI 控制的通用端口 I/O 管脚。

13.2.1 SPSCK — SPI 串行时钟

当 SPI 作为从 SPI 被使能时，该管脚是串行时钟输入。当 SPI 作为主 SPI 被使能时，该管脚是串行时钟输出。

13.2.2 MOSI —主数据输出，从数据输入

当 SPI 作为主 SPI 被使能且 SPI 管脚控制零（SPC0）为 0（不是双向模式）时，该管脚是串行数据输出。当 SPI 作为从 SPI 被使能且 SPC0 = 0 时，该管脚为串行数据输入。如果 SPC0 =1，选择单线双向模式并选择了主模式，该管脚就成为双向数据 I/O 管脚（MOMI）。同样，双向模式输出使能位决定该管脚作为输入（BIDIROE = 0）管脚还是输出管脚（BIDIROE = 1）。如果 SPC0 = 1 且选择了辅模式，该管脚不会被 SPI 使用，并恢复为通用端口 I/O 管脚。

13.2.3 MISO —主数据输入，从数据输出

当 SPI 作为主 SPI 被使能且 SPI 管脚控制零（SPC0）为 0 （不是双向模式）时，该管脚是串行数据输入。当 SPI 作为从 SPI 被使能且 SPC0 = 0 时，该管脚是串行数据输出。如果 SPC0 =1，选择单线双向模式并选择了辅模式，该管脚变成双向数据 I/O 管脚（SISO），双向模式输出使能位决定该管脚作为输入（BIDIROE = 0）管脚还是输出管脚（BIDIROE = 1）。如果 SPC0= 1 且选择了辅模式，该管脚不会被 SPI 使用，并恢复为通用端口 I/O 管脚。

13.2.4 $\overline{SS}$—从模式选择

当 SPI 作为从 SPI 被使能时，该管脚是低电平有效的从选择输入。当 SPI 作为主 SPI 被使能且模式默认使能关闭（MODFEN = 0）时，该管脚不被 SPI 使用，并恢复为通用端口 I/O 管脚。当 SPI 作为主 SPI 被使能时， MODFEN = 1，辅选择输出使能位决定该管脚是作为模式默认输入（SSOE = 0）还是辅选择输出（SSOE = 1）。

13.3 寄存器定义

SPI 有 5 个 8 位寄存器用于选择 SPI 选项、控制波特率、报告 SPI 状态和用来传送/接收数据。所有 SPI 寄存器的绝对地址分配请参考本技术手册中“存储器”一章的“直接页面寄存器总结”。在本节中仅通过它们的名字对寄存器和控制位进行引用。一个飞思卡尔提供的等同的或者头文件用来映射这些名字为适当的绝对地址。

13.3.1 SPI控制寄存器1（SPI1C1）

	7	6	5	4	3	2	1	0
读 写	SPIE	SPE	PTIE	MSTR	CPOL	CPHA	SSOE	LSBFE
复位	0	0	0	0	0	1	0	0

图 13-5 SPIC1字段描述

表13-1. SPI1C1 字段描述

字段	描述
7 SPIE	**SPI 中断使能 （ 用于 SPRF 和 MODF）** — 这是 SPI 接收缓冲器已满（SPRF）和模式故障（MODF）事件的中断使能。 0 禁止从 SPRF 和 MODF 中中断（使用轮询） 1 当 SPRF 或 MODF 为 1，请求硬件中断
6 SPE	**SPI 系统使能** —此位使能 SPI 系统并专门的 SPI 引脚控制 SPI 功能。如果 SPE 被清，SPI 被禁止并强制进入空闲模式，所有 SPIS 寄存器中的状态位将被重置。 0 SPI 系统禁止 1 SPI 系统使能
5 SPTIE	**SPI 发送中断使能** — 这是 SPI 发送缓冲器空（SPTEF）的中断使能位。 0 禁止从 SPTEF 中中断（使用轮询） 1 当 SPTEF 为 1 时，请求硬件中断
4 MSTR	主/从模式选择——此为为主从模式操作位 0 SPI 模块配置为从 SPI 器件 1 SPI 模块配置为主 SPI 器件

3 CPOL	**时钟极性**——这个位有效地对从主 SPI 到从 SPI 器件的时钟信号串联放置一个反相器。更多信息请参考 13.4.5 一节，“SPI 时钟格式”。 0 高有效 SPI 时钟（低无效） 1 低有效 SPI 时钟（高无效）
2 CPHA	**时钟相位** — 该位选择两种时钟格式的一种用于不同类型的同步串行外围器件。详情请参见 13.4.5，“SPI 时钟格式” 。 0 SPSCK 上的第一个边沿出现在 8 周期数据传输的第一个周期的中央 1 SPSCK 上的第一个边沿出现在 8 周期数据传输的第一个周期的起点
1 SSOE	**辅选择输出使能** — 该位的使用结合 SPCR2 中的模式故障使能（MODFEN）位和主从（MSTR） 控制位，以确定$\overline{SS}$ 管脚的功能，如表 13-2 所示。
0 LSBFE	LSB 优先（移位器方向）——此位并不影响 MSB 和 LSB 在寄存器中的位置读写数据寄存器总是 MSB 在第七位（或十六位模式下的第十五位） 0 SPI 串行数据传输以最高有效位开始 1 SPI 串行数据传输以最低有效位开始

表13-2. $\overline{SS}$引脚功能

MODFEN	SSOE	主模式	辅模式
0	0	通用 I/O（非 SPI）	从选择输入
0	1	通用 I/O（非 SPI）	从选择输入
1	0	模式故障的$\overline{SS}$输入	从选择输入
1	1	自动$\overline{SS}$ 输出	从选择输入

13.3.2 SPI控制寄存器2（SPI1C2）

该读/写寄存器用来控制 SPI 系统的任选功能。第 7、6、5 和 2 位无效，总为 0。

图13-6. SPI 控制寄存器2 （SPI1C2）

	7	6	5	4	3	2	1	0
读	0	0	0	M	BI	0	S	S
写				ODFEN	DROE		PISWAI	PC0
复位	0	0	0	0	0	0	0	0

=未执行或保留位

表13-3. SPIC2 寄存器字段描述

字段	描述
7 SPIMIE	**SPI 匹配寄存期使能—**此位中断使能接收缓冲硬件匹配（SPMF）功能 0 从 SPMF 中中断（用轮询方式） 1 当 SPMF=1 时，产生一个硬件中断请求
6 SPIMODE	**SPI8 或 16 位模式—**此位允许用户选择 8 或者 16 位数据传输长度。在主模式下改变此位将放弃当前传输，强制 SPI 进入空闲模式，重置 SPIS 寄存器的所有状态位。详细信息请参考 13.4.4 部分的“数据传输程度”。 0 8-bit SPI 移位寄存器、匹配寄存器、缓存 1 16-bit SPI 移位寄存器、匹配寄存器、缓存
4 MODFEN	**主模式故障功能使能—**当为辅模式配置 SPI 时，该位没有意义或影响 （$\overline{SS}$管脚是从选择输入）。在主模式中，该位决定$\overline{SS}$ 管脚的使用方式（如需了解更多信息，参见表表 13-2 。

	0 模式故障功能禁止，主$\overline{SS}$ 管脚恢复为不受 SPI 控制的通用 I/O 1 模式故障功能使能，主$\overline{SS}$ 管脚用作模式故障输入或辅选择输出
3 BIDIROE	**双向模式输出使能—**双向模式由 SPI 管脚控制 0 （SPC0 = 1）使能时， BIDIROE 决定 SPI 数据输出驱动器是否被使能为单个双向 SPI I/O 管脚。根据 SPI 是配置为主 SPI 还是从 SPI，它将 MOSI（MOMI）或 MISO（SISO）管脚分别用作单个 SPI 数据 I/O 管脚，当 SPC0 = 0, BIDIROE 没有意义或影响。 0 输出驱动器禁止，因此 SPI 数据 I/O 管脚作为输入 1 SPI I/O 管脚作为输出使能
1 SPISWAI	SPI 停止在等待模式中 0 在等待模式中 SPI 时钟继续运行 1 当 MCU 进入等待模式时 SPI 时钟停止
0 SPC0	**SPI 管脚控制 0—**SPC0 位用于选择单线双向模式。如果 MSTR=0（从模式），SPI 将 MISO（SISO）管脚用于双向 SPI 数据传输。如果 MSTR=1（主模式），SPI 将 MOSI（MOMI）管脚用于双向 SPI 数据传输。当 SPC0=1 时， BIDIROE 被用于为单个双向 SPI I/O 管脚使能或禁止输出驱动器。 0 SPI 使用单独管脚用于数据输入或输出 1 SPI 配置为单线双向操作

表13-4. 双向端口寄存器

引脚	SPC0	BIDIR0E	MISO	MOXI
主机模式工作				
常态	0	X	主入	主出
双向	1	0	此时 SPI 不用 MISO	主入
		1		主 I/O
从机模式下工作				
常态	0	X	从出	从进
双向	1	0	从入	此时 SPI 不使用 MOSI
		1	从 I/O	

13.3.3 SPI波特率寄存器（SPI1BR）

该寄存器用于为一个 SPI 主机设定预定标器和位速率分频因子。它可以在任何时间被读取或写入。

	7	6	5	4	3	2	1	0
读	0	SP	SP	SP	0	SP	S	S
写		PR2	PR1	PR0		R2	PR1	PR0
复位	0	0	0	0	0	0	0	0

=未执行或保留位

图13-7. SPI 波特率寄存器 （SPI1BR）

表13-5. SPI1BR 寄存器字段描述

字段	描述
6:4	**SPI 波特率预分频系数 —** 该 3 位字段为 SPI 波特率预分频器选择 8 个系数中的一个，如

SPPR[2:0]	表 13-6 所示。该预分频器的输入是总线速率时钟 （BUSCLK）。该预分频器的输出驱动 SPI 波特率系数的输入（图 13-5）。详细请见 13.4.6“SPI 波特率概述”。
2:0SPR[3:0]	**SPI 波特率系数 —** 该 4 位字段为 SPI 波特率系数选择 8 个系数中的一个，如表 13-7 所示。该被除数的输入来自 SPI 波特率预分频器（见图 13-15）。该被除数的输出是主模式的 SPI 波特率时钟。详细请见 13.4.6 的“SPI 波特率概述”。

表13-6. SPI 波特率预分频系数

SPPR2:SPPR1:SPPR0	预分频器系数
0:0:0	1
0:0:1	2
0:1:0	3
0:1:1	4
1:0:0	5
1:0:1	6
1:1:0	7
1:1:1	8

表13-7. SPI 波特率系数

SPR2:SPR1:SPR0	速度系数
0:0:0	2
0:0:1	4
0:1:0	8
0:1:1	16
1:0:0	32
1:0:1	64
1:1:0	128
1:1:1	256

13.3.4 SPI状态寄存器（SPI1S）

该寄存器有三个只读状态位，位 6、3、2、1 和 0 没有定义，总是读 0，写入无效。

	7	6	5	4	3	2	1	0
读	SPRF	0	SPTEF	MODF	0	0	0	0
写								
复位	0	0	0	0	0	0	0	0

=未执行或保留位

图13-8. SPI 状态寄存器 （SPI1S）

表13-8. SPI1S 寄存器字段描述

字段	描述
7 SPRF	**SPI 接收缓冲器满标志—**在一次 SPI 传输完成时，SPRF 被置位，表明接收到的数据可以从 SPI 数据寄存器（SPID）读取。当 SPRF 被置位时，通过读 SPRF，然后读取 SPI 数据寄存器，可将其清除。 0 在接收数据缓冲器中无可用数据

	1 在接收数据缓冲器中的数据可用
5 SPTEF	**SPI 发送缓冲器空标志—**当发送数据缓冲区空时，这个位被置位。PPTEF 被置位时，通过读取 SPI1S，然后向 SPI1D 的发送缓冲器写入一个数值，可将其清除。SPTEF=1 时，SPI1S 必须在向 SPI1D 写入数据之前被读取，否则 SPI1D 写操作将被忽略。如果 SPIC1 中的 SPTIE 位也被置位，SPTEF 会产生一个 SPTEF，CPU 中断请求。当一个数据字节从发送缓冲器传输进入发送移位寄存器时，SPTEF 被自动置位。对于一个空闲 SPI（在发送缓冲器或移位寄存器中没有数据，且无进行中的传输），写入 SPID 中的数据几乎立即被传输至移位器，因此，SPTEF 在两个总线周期内被置位，从而允许第二个 8 位数据值被排入发送缓冲器。在将移位寄存器中的值传输完成后，从发送缓冲器排列的值将自动转移至移位器，且 SPTEF 将被置位以表明在发送缓冲器中有空间可用于新数据传输。如果没有新数据在发送缓冲器中等待，SPTEF 只是保持置位状态，而不会有数据从缓冲器转移至移位器。 0 SPI 发送缓冲器非空 1 SPI 发送缓冲器空
4 MODF	**主模式故障标志—**如果 SPI 被设置为主模式，且从模式选择输入变为低电平时，MODF 被置位，这表明某个其它的 SPI 器件也被设置成了主模式。只有 MSTR=1，MODFEN=1，且 SSOE=0 时，$\overline{SS}$管脚才作为为模式故障错误输入；否则，MODF 将不会被置位。MODF 为 1 时，通过读 MODF，然后写入 SPI 控制寄存器 1 （SPIC1），可将其清空。 0 无模式故障错误 1 检测到模式故障错误

13.3.5 SPI 数据寄存器（SPIDH : SPIDL）

	7	6	5	4	3	2	1	0
读 写	Bit15	14	13	12	11	10	9	Bit8
复位	0	0	0	0	0	0	0	0

图13-9. SPI 数据寄存器高字节（SPIDH）

	7	6	5	4	3	2	1	0
读 写	Bit7	6	5	4	3	2	1	Bit0
复位	0	0	0	0	0	0	0	0

图13-10. SPI 数据寄存器低字节（SPDL）

SPI 数据寄存器（SPIDH：SPIDL）分别是 SPI 输入、输出数据寄存器。一次写入发送数据缓冲区就要写此寄存器来允许数据在排队和传输。

当 SPI 被配置成主机模式，数据缓冲区中的排队数据将在前一数据传出后很快传输。

当数据缓冲区接收到新的数据时，SPIS 寄存器的传输缓冲区空标志将独占。在写 SPI 数据寄存器前 SPTEF 被置位，此时 SPIS 应该被读取，否则写操作将会被忽略。

在 SPRF 被置位并且另一次传输完成之前任何时间，数据可从 SPIDH：SPIDL 中读取。在一次新传输结束前从接收缓冲区读取数据失败将导致超限并且新传输过来的数据可能会丢失。

在 8 字节模式下，只有 SPIDL 是有效的。读取 SPIDH 将全部返回为 0。 写 SPIDH 将被忽略。

在 16 位模式下，读取 SPIDH 或 SPIDL 到字节缓冲区。只有当另外读取一个数据的时

候这个缓冲区的数据才会被改写。写 SPIDH 或 SPIDL 到一个缓冲区。当两个字节的数据都被写了，它们将传输一个一致的 16 位值到发送缓冲区。

13.3.5 SPI 匹配寄存期（SPIMH : SPIML）

当值的 SPI 接收数据缓冲区收到的值等于在 SPIMH：SPIML 寄存器的值时，这些读/写寄存器包含硬件比较值，确定了匹配的 SPI 标志（SPMF）。

在 8 位模式中，只有 SPIML 可用。读取的 SPIMH 将返回所有 0s。写入 SPIMH 会被忽视。

在 16 位模式下，读取字节（SPIMH 或 SPIML）锁存到缓冲区，直到新的其他字节锁存读取。将 SPIMH 或 SPIML 的值，写入到一个缓冲区。当两个字节写入完毕，则发送到 SPI 的匹配寄存器。

	7	6	5	4	3	2	1	0
读 写	Bit15	14	13	12	11	10	9	Bit8
复位	0	0	0	0	0	0	0	0

图13-11. SPI 匹配寄存器高字节 （SPIDH）

	7	6	5	4	3	2	1	0
读 写	Bit7	6	5	4	3	2	1	Bit0
复位	0	0	0	0	0	0	0	0

图13-12. SPI 匹配寄存器低字节 （SPIDL）

13.4功能描述

通过设置 SPI 控制寄存器 1 的使能位（SPE）开启 SPI 系统。当 SPE 被置位，SPI 相关的四个引脚被 SPPI 功能独占为：

—从机选择（$\overline{SS}$）

—串行时钟（SPSCK）

—主出/从入（MOSI）

—主入/从出（MISO）

当 SPTEF = 1 时，然后将数据写入到发送数据缓冲区（写入 SPIDH：SPIDL），通过读取 SPI 的状态寄存器（SPIS），主机发起一次 SPI 传输。当一次传输完成后，收到的数据被转移到接收数据缓冲区。在 SPIDH：SPIDL 寄存器读取时为 SPI 接收数据缓冲区；写入时为传输数据的 SPI 写入缓冲区。

在 SPI 控制寄存器 1（SPIC1)中的时钟相位控制位(CPHA)和一个时钟极性控制位(CPOL)选择四种可能的时钟的格式之一 为 SPI 系统使用。CPOL 位简单的选择一个非反相或反向时钟。通过抽取在奇数 SPSCK 边缘或偶数 SPSCK 边缘数据，CPHA 用来存放两个根本不同的协议。

SPI 可以被配置成主机或者是从机。当 SPI 控制寄存器 1 的 MSTR 位被置位，那就意味着选择了主机模式；当 MSTR 被清零，那就是选择了从模式。

13.4.2 主机模式

当 MSTR 被置位时，SPI 工作在主机模式下。只有在主机模式下才能发起一次传输。通

过在 SPTEF=1 时读取 SPIS 寄存器的值并写到主 SPI 的数据寄存器中来开始一次传输。如果移位寄存器为空，传输字节迅速发送到移位寄存器中。数据在时钟的控制下移出到 MOSI 引脚。

——SPSCK

SPI 波特率寄存器中的 SPR2，SPR1 和 SPR0 波特率选择位与 SPPR2，SPPR1， SPPR0 波特率预选位结合，控制波特率发生器并确定传输速度。该 SPSCK 脚是 SPI 时钟输出引脚。通过 SPSCK 引脚，主机的波特率发生器控制从机外设的移位寄存器。

——MOSI，MISO 引脚

在主机模式下，串行数据输出引脚（MOSI）和窜行数据输入引脚的功能是决定于 SPC0 和 BIDIROE 控制为。

——$\overline{SS}$引脚

如果 MODFEN 和 SSOE 位被置位，$\overline{SS}$引脚被配置为从机选择输出。在每次传输时$\overline{SS}$输出为低电平；当 SPI 空闲时输出高电平。

如果 MODFEN 被置位并且 SSOE 被清零，$\overline{SS}$引脚被配置为输入，用于检测模式错误。如果$\overline{SS}$输入变成低电平这就意味着有别的主机试图驱动 MOSI 和 SPSCK 线并导致了模式错误。在这种情况下，SPI 通过清 MSTR 位并禁止从机输出缓冲 MISO（或者在双向模式下的 SISO），马上切换到从机模式下。这样导致的结果是所有的输出都被禁止并且 SPSCK，MOSI，MISO 均为输入。如果当错误发生时候正好有数据在传送，这次传输将被放弃且 SPI 被强制进入空闲模状态。

模式错误也会使得状态寄存器 SPIS 的模式标志位（MOD）置位。如果当模式位 MOD 被置位时 SPI 的终端使能也被置位，那么还将需要一个 SPI 中断序列。

当写入主机数据寄存器发生时，将有半个 SPSCK 循环的延时。之后，主机开启 SPSCK。其余的传输操作大同小异，取决于由 SPI 控制寄存器 1 中的 SPI 时钟相位（CPHA）指定的时钟格式 （详见 13.4.5“SPI 时钟格式”）

注意：对位 CPOL、CPHA，SSOE，LSBFE，MODFEN，SPC0、置 BIDIROE 的 SPC0 位，SPIMODE，SPPR2-SPPR0 和 SPR2- SPR0 在主模式下的改变将中止进行中的传输并强制进入空闲状态。远程从机无法检测到这一点，因此，主机要确保远程从机重新设置为空闲状态。

13.4.3 从机模式

当 SPI 控制寄存器的 MSTR 位被清零时 SPI 工作于从机模式

——SPSCK

在从机模式下，SPSCK 是主机输出的时钟。

——MISO，MOSI 引脚

在从机模式下，串行数据输出引脚的（MISO）和串行数据输出（MOSI）由 SPI 控制寄存器 2 的 SPSC0 位和 BIDROE 位决定。

——$\overline{SS}$引脚

$\overline{SS}$引脚是从机选择输入。在数据传输发生以前，从机 SPI 的$\overline{SS}$引脚必须为低电平。$\overline{SS}$应该保持低一直到传输结束。如果$\overline{SS}$变高，SPI 将强制进入空闲状态。

$\overline{SS}$输入也控制串行数据输出引脚，如果$\overline{SS}$为高（没有被选中），串行数据输出引脚为高阻态；如果$\overline{SS}$是低电平，SPI 数据寄存器的第一位被移出串行数据发送引脚。同样的，如果从机没有被选择（$\overline{SS}$为高电平），那么 SPSCK 将被忽略而且移位寄存器没有内部转移发生。

尽管 SPI 有能力进行双向操作，但是某些 SPI 外设只能工作于从机模式接收 SPI 数据。

对于这些简单的设备，就没有数据输出引脚。

注意：当外围设备处于双向工作时，小心不要同时使能两个串行输出驱动同一个数据输出总线的线接

收器。

只要不超过一个从设备驱动系统从机的串行数据输出线，几个从机获得了从主相同的传输是可能的，虽然主机不能得到接收从机的返回信息。

如果 SPI 控制寄存器 1 的时 CPHA 位被清零，奇数的 SPSCK 输入边缘导致数据在串行数据输入引脚被锁定。偶数边导致以前锁存的串行数据输入引脚的值根据 LSBFE 位转移到 LSB 或 SPI 移位寄存器的 MSB。

如果时钟相位位被置位，偶数的 SPSCK 输入边缘导致串行数据输入引脚的数据被锁定。奇数个数的边缘导致以前的串行数据输入引脚的值根据 LSBFE 位，转移到 LSB 或 SPI 移位寄存器的 MSB。

当时钟相位 CPHA 被置位，第一个边缘被用来为串行数据输出引脚获取第一个数据位。当时钟相位 CPHA 被清零且$\overline{SS}$输入为低电平（选择从机模式），第一个 SPI 数据位输出串行数据输出引脚。在第 8 位（SPIMODE = 0）或 16 位（SPIMODE = 1）移位之后，传输被认为是完成了并且接收到的数据也被传输到 SPI 数据寄存器。为了表示传输完成，在 SPI 的状态寄存器 SPRF 标志备置位。

注意:对位 CPOL，时钟相位 CPHA，SSOE，LSBFE，MODFEN，SPC0, 置 BIDIROE 的 SPC0 位并且从模式下的 SPIMODE 位的改变将会破坏当前正在进行的传输而且尽量避免类似操作。

13.4.4 数据传输长度

SPI 能支持 8 或者 16 位的数据传输。数据长度可以通过配置 SPIC2 寄存器的 SPIMODE 位来改变。

在 8 位模式下（SPIMODE=0），SPI 数据寄存器由一个字节组成（SPIDL）。SPI 匹配寄存器也是由一个字节组成(SPIML)。读取 SPIDH 和 SPIML 将返回 0。对它们的写操作会被忽略。

在 16 位模式下（SPIMODE=1）下， SPI 数据寄存器由两个字节组成：SPIDH 和 SPIDL。读字节 SPIDH 或 SPIDL 的两个字节的内容锁存到缓冲区，他们将保持锁存直到其他字节被读取。写入任何字节（SPIDH 或 SPIDL），会将其值锁存到缓冲区中。当两个字节均被写入，则传输一个前后一致的 16 位值到发送数据缓冲区。

在 16 位模式下， SPI 匹配寄存器同样由两字节组成：SPIMH 和 SPIML。读取字节 SPIMH 或 SPIML 将锁存两个字节的内容到缓冲区中，他们将一直锁定到其他字节被读取。写入任 SPIMH 或 SPIML 任何一个字节，其值将会被锁存到缓冲区中。当两个字节都被写入，则传输一个对应的 16 位值到数据发送缓冲区。

在主模式中的 8 位和 16 位数据长度之间的任何转换（由 SPIMODE 位控制）将被中止传输，迫使 SPI 进入闲置状态，并重置在 SPIS 寄存器中的所有的状态位。在对 SPIMODE 位写入后启动一次传输，通过置 SPTEF = 1，读取 SPIS 寄存器，数据必须以 16 位模式写入 SPIDH：SPIDL（SPIMODE = 1）或以 8 位模式（SPIMODE = 0）写入 SPIDL。

在从模式下，用户程序只应该写 SPIMODE 位一次，以避免中断当前的数据传输。

注意：如果主机和从设备之间的长度不一致，可能发生数据丢失。

13.4.5 SPI时钟格式

为了适应从不同制造商的各种同步串行外设，SPI 系统有一个时钟极性（CPOL）位和时钟相位（CPHA）控制位来选择 4 个数据传输时钟格式的一种。 CPOL 选择性地插入一个与时钟串联的反相器。时钟相位 CPHA 选择两种不同的时钟和数据之间时钟相位关系。

图 13-13 显示了时钟格式时 SPIMODE = 0（8 位模式）和 CPHA = 1。在图顶部，显示了作为参考的 8 位。第一位在第一次 SPSCK 沿开始计数而在第 16 个 SPSCK 沿后的半个 SPSCK

周期时第 8 位结束。 MSB 的优先和 LSB 优先总线指明 SPI 数据位的顺序。这个顺序取决于在 LSBFE 的设置。 SPSCK 极性的两种变化情况都得到了显示，但只有其中一种波形适用于特定的传输，这取决于 CPOL 的值。那些 SAMPLE IN 波形适用于从机输入或主机输入。MOSI 波形适用于主机 MOSI 输出引脚而 MISO 波形适用于从机 MISO 输出。$\overline{SS}$ OUT 输出波形适用于主机对从机的选择输出（假设 MODFEN 且 SSOE = 1）。主机$\overline{SS}$输出在开始传输之前变成低电平有效，并保持半个 SPSCK 周期并且在第 8 位传输结束时间返回高电平。$\overline{SS}$ IN 波形适用于从机的从机选择输入。

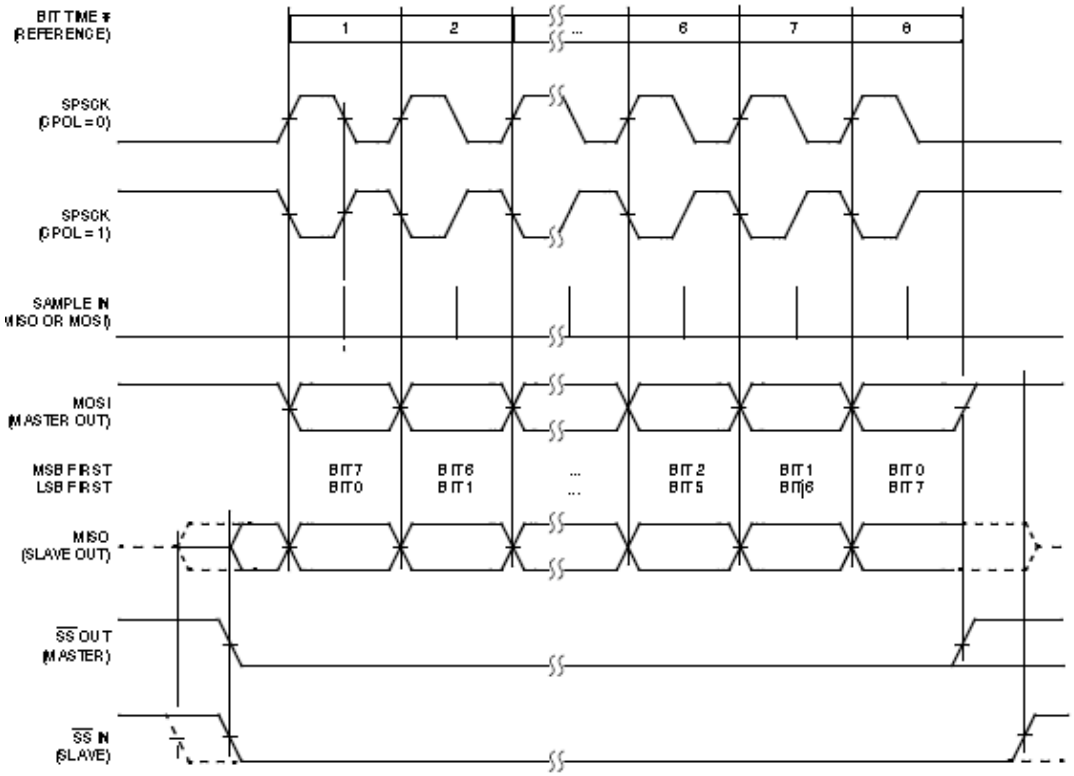


图13-13 SPI时钟格式（CPHA = 1）

当 CPHA=1，$\overline{SS}$进入低电平有效时，从器件开始驱动它的 MISO 输出，但是直到第一个 SPSCK 边沿时数据才被定义。第一个 SPSCK 沿将数据的第一位从移位器向上转移至主器件的 MOSI 输出端和从器件的 MISO 输出端。下一个 SPSCK 沿引起主器件和从器件分别在它们的 MISO 和 MOSI 输入端同时对数据位值进行采样。在第三个 SPSCK 沿，SPI 移位器在刚才被采样过的数值内移位一个位的位置，并且将第二个数据位的值从移位器的另一端移出，分别转移至主器件和从器件的 MOSI 和 MISO 输出端。当 CHPA=1 时，从器件的$\overline{SS}$ 输入端在两次传输之间不需要进入它的无效高电平状态。

图 13-11 所示为当 CPHA=0 时的时钟格式。在图的顶端，显示了用于参考的 8 位，当从模式被选择好时（$\overline{SS}$ IN 进入低电平）第一位开始；第 8 位在最后一个 SPSCK 沿结束。MSB 优先和 LSB 优先线显示了取决于 LSBFE 设置情况的 SPI 数据位的顺序。SPSCK 极性的两种变化情况都得到了显示，但是只有其中一种波形应用于一种特殊的传输，这取决于 CPOL 的值。SAMPLE IN 波形应用于从器件的 MOSI 输入或主器件的 MISO 输入。MOSI 波形应用于主器件的 MOSI 输出管脚，而 MISO 波形应用于从器件 MISO 输出管脚。$\overline{SS}$ OUT 波形应用于主器件的从模式选择输出（如果 MODFEN 和 SSOE=1）。主器件的$\overline{SS}$ 输出在

第一位传输开始时变为低电平有效；并在发送第 8 位结束后返回高电平，且保持半个 SPSCK 周期。$\overline{SS}$IN 波形应用于从器件的从模式选择输入。

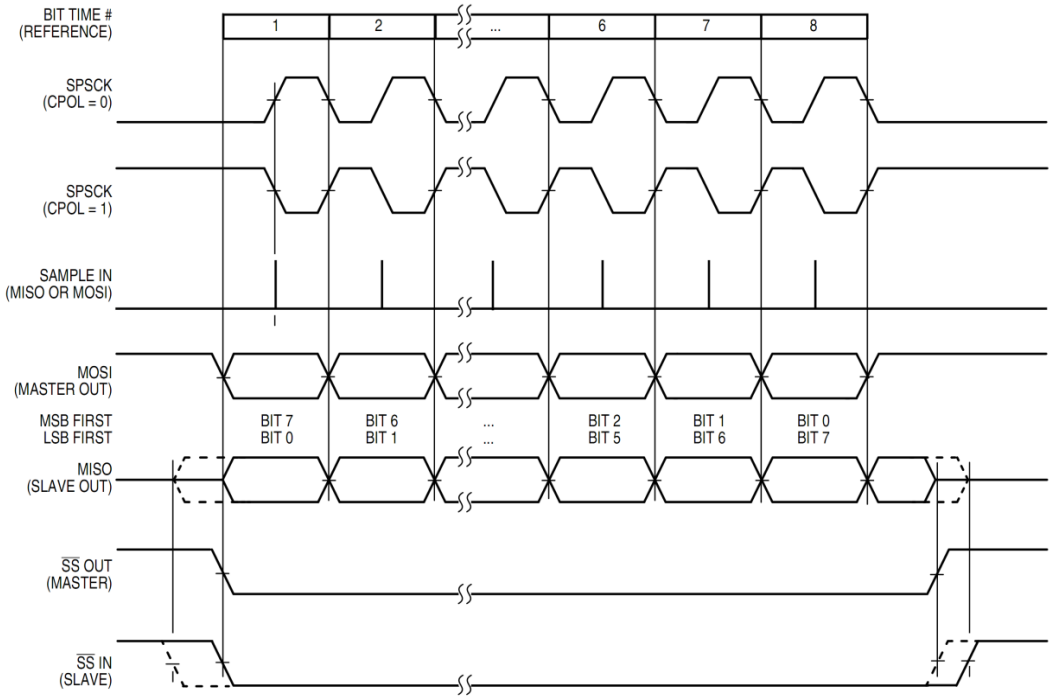


图 13-14. SPI 时钟格式（CPHA = 0）

当 CPHA=0， $\overline{SS}$进入低电平有效时，从器件从第一个数据位值开始驱动它的 MISO 输出（MSB 还是 LSB 取决于 LSBFE）。第一个 SPSCK 沿引起主器件和从器件分别在它们的 MISO 和 MOSI 输入端对数据位值进行采样。在第二个 SPSCK 沿，SPI 移位器在刚才被采样过的数值内移位一个位的位置，并且将第二个数据位的值从移位器的另一端移出，分别转移至主器件和从器件的 MOSI 和 MISO 输出端。当 CHPA=0 时，从器件的$\overline{SS}$ 输入端在两次传输之间必须进入它的无效高电平状态。

13.4.6 SPI波特率的产生

如图 13-15， SPI 波特率发生器的时钟源为总线时钟。三个预分频位（SPPR2：SPPR1：SPPR0）从 1，2，3，4，5，6，7，或 8 中选择一个预分频因子。预分频器输出除以 2，4，8，16,32,64,128 或 256 阶段的获得内部 SPI 主机模式比特率时钟。这三个预分频选择位(SPR2：SPR1：SPR0）决定了除数的大小。

波特率发生器只有当 SPI 处于主机模式下并且正在进行串行发送。在其余情形，分频器是不能减小 Idd 电流。

波特率分频器如下：

波特率分频系数=（SPPR+1）*2^（SPR+1）

波特率可以用下面的方程式计算：

波特率=总线时钟/波特率分频系数

图13-15.SPI波特率产生器

13.4.7 特性

13.4.7.1 $\overline{SS}$ 输出

$\overline{SS}$的输出特性，在传输过程中选择外部设备时自动驱动$\overline{SS}$引脚低电平；在空闲时驱动为高电平以取消外部设备。当选中$\overline{SS}$输出，$\overline{SS}$ 输出引脚连接到外部设备的$\overline{SS}$输入引脚。

如表 13-2 所示，在正常 SPI 操作时通过插入 SSOE 和 MODFEN 位，$\overline{SS}$输出只有在主模式下才有效。

当$\overline{SS}$输出使能时，模式错误特性被禁止。

注意：当在多主机系统中使用$\overline{SS}$输出特性一定要注意。因为模式错误不可用于检测主机之间的系统错。

13.4.7.2 双向模式（MOMI或SISO）

当 SPI 控制寄存器 2（见表 13-9）的 SPC0 位被置位时，双向模式被选中。 在这个模式下，SPI 仅仅用一个数据引脚作为与外围设备的接口。MSTR 位决定了用哪个引脚。在主机模式下，MOSI 引脚用作串行数据 I/O（MOMI）引脚；而在从机模式下，MISO 引脚用作串行数据 I/O（SISO）引脚。 此时，这些引脚并不是被 SPI 使用。

当 SPE=1	主机模式 MSTR=1	从机模式 MSTR=0
正常模式 SPC0=0	串行输出 SPI 串行输入 MOSI MISO	串行输入 SPI 串行输出 MOSI MISO
双向模式 SPC0=1	串行输出 SPI 串行输入 MOMI Bidiroe	串行输入 SPI 串行输出 Bidiroe SIS

表13-9 正常模式和双向模式

每个 I/O 引脚的方向均由 BIDROE 位定义。如果引脚被配置成输出，移位寄存器的串行数据被传输到引脚。此引脚也是移位寄存器的输入引脚。

SPSCK 在主机模式下是用作输出而在从模式下是输入。

$\overline{SS}$在主机模式下可以是输入或者输出；但是在从机模式下总是输入。

双向模式并不影响 SPSCK 和$\overline{SS}$功能。

注意：在双向模式下，随着模式错误的使能，MOSI 和 MISO 可以被 SPI 占用，虽然 MOSI 通常用于在双

向传输模式并且 MISO 不被 SPI 使用。如果模式发生故障时，SPI 会自动切换到 slave 模式，在这种情况下 MISO 被 SPI 占用和 MOSI 不被使用。如果 MISO 引脚用于其他目的。这个必须考虑。

13.4.8 错误发生条件

SPI 只有一种错误发生条件：模式故障错误

13.8.1 模式故障错误

如果当 SPI 配置为主机模式时， $\overline{SS}$输入变为低电平，这就意味着一个系统错误。不止一个主机可能正在试图同时驱动 MOSI 与 SPSCK 线。在正常运行时这种状况不允许而且如果 MODFEN 位被置位，SPI 状态寄存器的 MODE 位被自动置位。

在特殊情况下，SPI 在主控模式并且 MODFEN 位被清除，那么$\overline{SS}$引脚不被 SPI 使用。在这个特殊情况下,模式错误功能被禁止和 MODF 位将保持。在 SPI 的系统配置位从机模式,$\overline{SS}$引脚为一个独占的输入管脚。故障模式错误不会出现在从机模式。

如果发生故障模式错误,SPI 将切换到从机模式，并产生从机缓冲区禁用异常。因此 SPSCK，MISO 和 MOSI 引脚被迫高阻抗输入，以避免与别的输出驱动冲突。进行中的传输中断，SPI 被迫进入空闲状态。

如果模式故障错误发生在 SPI 主控模式下的双向模式，MOMI 输出使能（MOSI 在双向模式下）如果它曾被置位那么将会被清除。当 SPI 在从机模式下的双向模式时，故障模式错误不会发生。

通过先读 SPI 的状态寄存器（MODF 被置位），接着写入 SPI 控制寄存器 1，该模式故障标志将被自动清除。如果模式故障标志被清除，SPI 将成为一个正常的主或从模式了。

13.4.9 低功耗模式

13.4.9.1 SPI的运行模式

在运行模式下，SPI 控制寄存器的使能位（SPE）被清零，SPI 系统进入低功耗、禁止状态。SPI 寄存器仍然可以被访问，但是此模块的核心时钟被禁止了。

13.4.9.2 SPI的等待模式

SPI 在等待模式的工作状态取决于 SPI 控制寄存器 2 的 SPISWAI 位的状态。

如果 SPISWAI 被清零，当 CPU 在等待模式时 SPI 能正常工作。

若 SPISWAI 被置位，当 CPU 在等待模式时，停止产生 SPI 时钟，且 SPI 模块进入能　量储存状态。

若 SPISWAI 被置位且 SPI 被配置为主机,任何正在进行的传输和接收将在等待模式的入口被停止。当 SPI 推出等待模式时传输和接收将被恢复。

若 SPISWAI 被置位且 SPI 杯配置成从机，若果 SPSCK 仍然被主机驱动，那么任何正在进行的传输和接收将继续。这样就保证了从机和主机的同步。

若当从机在等待模式下，主机传输数据，从机将继续发出与在等待开始的运作模式一致的数据（即，如果从机正在向主机发送其 SPIDH:SPIDL，它将继续发送相同的字节。否则，如果从机正在发送从主机发送过来的最新的数据，它将发送此前的所有接收的数据。）

注意：当从机在等待模式或者 STOP3 模式下等待主发送的数据时需要注意。尽管如此移位寄存器将继续运行，SPI 的其余部分关闭（即 SPRF 中断将不会生成除非退出停止或等待模式）。此外，从移位寄存器中的数据将 不被复制到 SPIDH：SPIDL 除非从机 SPI 退出或停止模式等。如果在一次传输中进入或退出等待模式时，将会产生 SPRF 标志和 SPIDH：SPIDL 复制。如果从机在空闲模式时进入等待模式和空闲模式下

退出等待模式，SPRF 或 SPIDH：SPIDL 复制都不会发生。

13.4.9.3 SPI的停止模式

Stop3 模式取决于 SPI 系统。在进入 stop3 模式，SPI 模块时钟被禁用（保持高位或低位）。如果 SPI 是在主控模式和交换数据时，CPU 进入停止模式，传输被冻结，直到 CPU 退出停止模式。停止后，到外部的 SPI 和来自外部的 SPI 的数据被正确的交换。在从模式下，SPI 将与主机保持同步。

停止模式并不依赖于 SPISWAI 位。

在所有的停止模式中，SPI 模块被完全禁止。在停止后，所有的寄存器都被重置到他们的默认值并且 SPI 模块必须被初始化。

13.4.9.4 复位

寄存器和信号的复位值在 13.3“寄存器定义”中已经讲述，其中详细讲解了寄存器和其位段的定义。

若在复位后数据传输发生在从机模式，并没有写入到 SPIDH:SPIDL，那将传输垃圾数据，或者是最后收到的主机的数据。

在复位后读取 SPIDH:SPIDL 总是返回 0 值。

13.4.9.5 中断

SPI 中断只有 SPI 使能（SPIC1 的 SPE 位被置位）时 SPI 才产生中断请求。以下描述 SPI 时如何产生中断请求以及 MCU 如何感知这一请求。中断向量偏移和中断优先级均由芯片决定。

13.4.10 SPI中断

有四个标志为，三个中断掩码位，一个与 SPI 系统相关联的中断向量。SPI 中断使能掩码（SPIE）使能 SPI 接收满标志（SPRE）和模式故障标志（MODF）。SPI 传输中断使能掩码（SPTIE）使能中断传输缓冲区空标志（SPTEF）。SPI 匹配中断使能掩码位（SPIME）使能 SPI 匹配标志（SPMF）。当任意一个标志碑置位且相关的中断掩码位也被置位，CPU 将收到一个硬件中断。若中断掩码都被清零，软件可以轮询相关的标志位，而不是以中断方式。SPI 中断服务例程（ISR）应该检查标志位以确定什么时间导致此次中断。服务例程应该在返回之前清除标志位（一般是在 ISR 开始时）

13.4.10.1 模式

当主机检查到$\overline{SS}$引脚一个错误时，MODF 将发生。主机 SPI 应该被配置成 MODF 特性（见表 13-2）。一旦 MODF 被置位，但前的数据传输将会中止，随后的数据也将改变。

MSTR=0，SPIC1 中的主机位被置位。

MODF 中断与状态寄存器的 MODF 标志对应。清标志将同样清除中断。在 MODF 标志被重置前中断将一直有效。MODF 用一个自动清除进程，曾在 13.3.4“SPI 状态寄存器”中描述。

13.4.10.2 SPRF

SPI 接收数据缓冲区以准备好接收新的数据时 SPRF 将发生。在 8 位模式下，只有在所有 8 位数据从 SPIDL 移出到移位寄存器 SPRF 才被置位。在 16 位模式，只有在所有 16 位从 SPIDH：SPIDL 移出到移位寄存器 SPRF 才被置位。

一旦 SPTEF 被置位，除非被服务否则将不会被清除。SPTEF 有一个自动清除进程。详见 13.3.4“SPI 状态寄存器（SPIS）”

13.4.10.3 SPMF

只有当接收数据缓冲区中的数据等于在匹配寄存器中的数据时 SPMF 才会发生。在 8 位模式下，接收数据缓冲区数据的 8-0 位等于 SPIML 的值时，SPMF 才会被置位。在 16 位模式下，接收数据缓冲区数据的 15-0 位等于 SPIMH：SPIML 的值时 SPMF 才会置位。

13.5 初始化/应用信息

13.5.1 SPI模块初始化例子

13.5.1.1 初始化序列

在 SPI 模块能被使用以前，下面等待初始化步骤是必须的。

1. 更新控制寄存器 1（SPIC1）以使能 SPI 以及控制中断使能。这个寄存器同样设置 SPI 位一个主机或者从机模式。决定时钟相位，极性，培植 SPI 的主要选项。
2. 更新控制寄存器 2 （SPIC2）以使能附加的 SPI 功能，比如：SPI 匹配中断特性，主机的模式错误功能，双向模式输出 8 或 16 位选择，别的一些控制功能也在此定义。
3. 更新波特率寄存器（SPIBR）以设置预分频器和位率除数。
4. 更新硬件匹配寄存期（SPIMH：SPIML）.若硬件中断使能，当寄存器中的数据与接收的数据相等，则触发中断。
5. 在主机模式，当 SPTEF=1 时读取 SPIS 的值，然后写入数据寄存器（SPIDH:SPIDL）以开始传输。

13.5.1.2 伪代码举例

在这个例子里，SPI 模块将会配置成竹模式并只有硬件匹配中断使能。SPI 将以时钟总线最大波特率的 1/2 工作与 16 位模式下。时钟相位和极性将被设置成高有效的 SPI 时钟，既当第一个 SPSCK 时钟沿发生在第一次数据传输循环。

SPICI=0x54（%01010100）

Bit7	SPIE	=0	禁止接收和模式错误中断
Bit6	SPE	=1	使能 SPI 系统
Bit5	SPTIE	=0	禁止 SPI 传输中断
Bit4	MSTR	=1	配置 SPI 模块为主机模式
Bit3	CPOL	=0	配置 SPI 时钟为高有效
Bit2	CPHA	=1	第一次数据传输循环的第一个时钟沿
Bit1	SSOE	=0	当模式错误使能决定$\overline{SS}$引脚功能
Bit0	LSBFE	=0	SPI 串行数据传输以最重要位开始

SPIC2=0xC0（% 11000000）

Bit7	=0	未生效
Bit6:4	=000	设置预分频除数为 1
Bit3	=0	未生效
Bit2:0	=000	设置预分频除数为 2

SPIS=0x00（% 00000000）

Bit7	SPRF	=0	未生效
Bit6	SPMF	=000	设置预分频除数为 1

Bit5	SPTEF	=0	未生效
Bit4	MODF	=000	设置预分频除数为 2
Bit3:0		=0	未生效

SPIMH = 0xXX

在 16 位模式下，寄存器保持硬件匹配缓冲区中的 8-15 位。在 8 位模式下，写这个寄存器将被忽略。

SPIML = 0xXX

保持硬件缓冲区中的 0-7 位

SPIDH = 0xXX

在 16 位模式下，保持发送缓冲区中将要发送和接收缓冲区中接收的 8-15 位数据

SPIDL = 0xXX

保持发送缓冲区中将要发送和接收缓冲区中接收的 0-7 位数据。

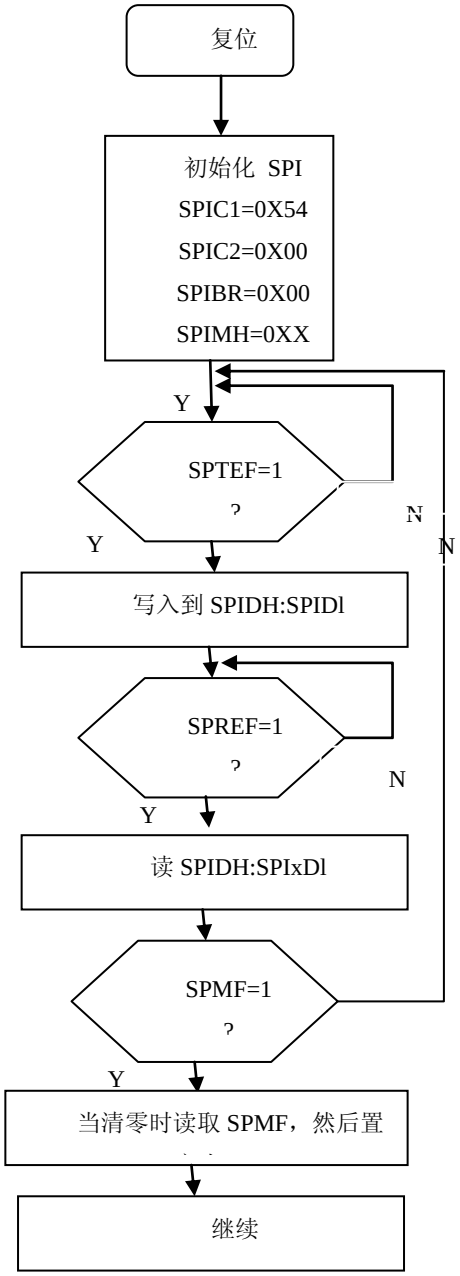


图 13-16. 16位模式下的初始化流图

第十四章 定时器/脉冲调制器（S08TPMV3）

14.1 介绍

该 MC9S08JS16 系列包括一个独立的定时器/脉宽调制（TPM）的模块，它支持传统输入捕捉，输出比较，或缓冲边沿对齐在每声道脉宽调制（PWM）。每个 TPM 有一个控制位来配置通道作为中心对齐的 PWM 功能来运行。在这两个 TPM 芯片，每个计时的功能是基于一个单独的 16 位计数器。由预分频器和系数特性来控制频率和时间参考范围（两次溢出时间间隔）。

用固定的系统时钟，XCLK，作为 TPM 模块时钟源将允许预分频器在不超过 MCGOUT 四分频的频率下工作。（见 9.4.7 节,”固定频率时钟”）。

注意：在 TPM 章节，TPM 与 MTIM 的外部输入时钟源 TCLK，被参考为 TPMCLK 时钟。

14.2 特性

MC9S08JS16 系列系统时钟包含一个 2-通道的 TPM。时钟系统包含以下几个不同的特性：

总共 2 个通道：

— 每个通道可以是输入捕捉、输出比较或边沿对齐 PWM 模式

— 上升沿、下降沿或任何沿输入捕捉触发

— 设置、清除、切换输出比较操作

— PWM 输出上的可选择性

在所有通道上，每个 TPM 可配置为缓冲、中央对齐脉冲宽度调制（CPWM）

每个 TPM 预分频器的钟源可独立选作为总线时钟，固定式系统时钟，或外部引脚：

—可选的预分频因子 1、2、4、8、16、32、64 或 128

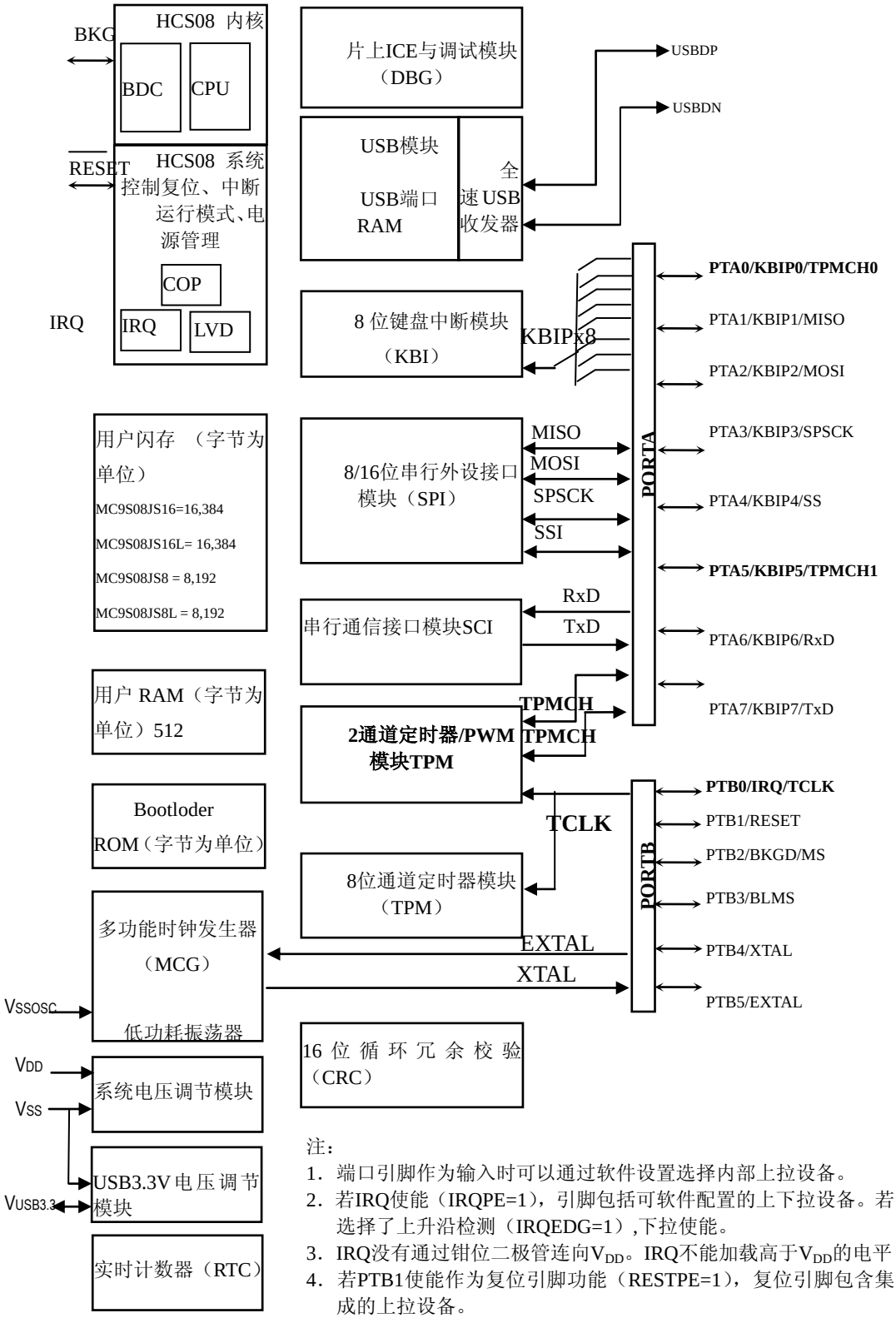
—外部时钟输入

16 位自由运行或模数向上/ 向下（CPWM）计数器

控制计数范围的 16 位模寄存器

定时系统启动

每个通道外加一个计数器中断



注：

1. 端口引脚作为输入时可以通过软件设置选择内部上拉设备。
2. 若IRQ使能（IRQPE=1），引脚包括可软件配置的上下拉设备。若选择了上升沿检测（IRQEDG=1）,下拉使能。
3. IRQ没有通过钳位二极管连向V_{DD}。IRQ不能加载高于V_{DD}的电平
4. 若PTB1使能作为复位引脚功能（RESTPE=1），复位引脚包含集成的上拉设备。
5. 引脚包含集成上拉设备。
6. 当KBI使能（KBIPEn=1）而且相关引脚被配置为上拉设备使能，那么上升沿（KBEDGn=1）被用于将上拉设备配置为下拉设备。

图 14-1. MC9S08JS16系列框图TPM模块和引脚高亮显示

14.3 TPMV3与以前版本的不同之处

TPMV3 是 Timer/PWM 模块的最新版本，处理了以前版本的错误。在下面的部分突出显示了 TPMV3 和 TPMV2 模块，和一些在端口编码时应该注意的问题。

表14-1. TPMV2和TPMV3 端口操作注意事项

事件	TPMV3	TPMV2
写入 TPMCnTH:L 寄存器1		
仅仅写入 TPMCnTH 或 TPMCnTL 寄存器	清除 TPM 计数器（TPMCNTH:L）和预分频计数器	仅清除 TPM 计数器（TPMCNTH:L）
读 TPMxCNTH:L 寄存器1		
在 BDM 模式下，任何对 TPMCnTH:L 寄存器的读取	返回 TPM 被冻结的计数器的值	如果在激活 BDM 模式前仅读取 TPMCnTH:L 寄存器的一个字节，从读缓冲区中返回 TPMCnTH:L 锁存值（而不是被冻结的 TPM 计数器值）。
在 BDM 模式下，写 TPMSC, TPMCNTH or TPMCNTL 寄存器	清除读一致机制	不清除读一致机制
读 TPMCnVH:L 寄存器 2		
在 BDM 模式下，仅读取 TPMCnVH:L 寄存器	返回 TPMCnVH:L 寄存器的值	如果在激活 BDM 模式前仅读取 TPMCnVH:L 寄存器的一个字节，则从读缓冲区中返回 TPMCnVH:L 锁存值（而不是被冻结的 TPM 计数器值）。
在 BDM 模式下，写 TPMCnSC 寄存器	清除读一致机制	不清除读一致机制
写 TPMCnVH:L 寄存器		
输入捕捉模式下，写 TPMCnVH:L 寄存器 3	不允许	允许
输出比较模式下，当（CLKSB:CLKSA 不等于 0:0）时，写 TPMCnVH：L 寄存器 3	在下一个 TPM 计数器第二个字节被改变时，更新 TPMCnVH：L 寄存器为写缓冲区的值(在预分频计数结束时)。	总是在第二个字节被写入时更新寄存器
在边沿对齐 PWM 模式下，当（CLKSB:CLKSA 不等于 00）时，写 TPMCnVH:L 寄存器	在两个字节被写入且 TPM 计数器由（TPMMODH:L – 1）变为（TPMMODH:L）时，TPMCnVH:L 寄存器被更新为其写缓冲区的值。 注意：若 TPM 计数器为自由运行计数器，那么当 TPM 计数器从 0xFFFE 变为 0xFFFF 时，将做此更新。	在两字节都被写入且 TPM 计数器从 TPMMODH:L 变为 0x0000 时，将会更新寄存器。
在中心对齐 PWM 模式下，	在两个字节被写入且 TPM 计数器	在两字节都被写入且 TPM 计数器

CLKSB:CLKSA 不等于 00），写 TPMCnVH:L 寄存器 4	由（TPMMODH:L – 1） 变为（TPMMODH:L） 时，TPMCnVH:L 寄存器被更新为其写缓冲区的值。 注意：若 TPM 计数器为自由运行计数器，那么当 TPM 计数器从 0xFFFE 变为 0xFFFF 时，将做此更新。	从 TPMMODH:L 变为 TPMMODH:L– 1 时，将会更新寄存器。
中心对齐 PWM		
当 TPMCnVH:L = TPMMODH:L 5 时	产生 100%的占空周期	产生 0%的占空周期
当 TPMCnVH:L = （TPMMODH:L – 1）6	产生一个接近 100%的占空周期	产生 0%的占空周期
TPMCnVH:L 从 0x0000 变为非零值	等待开始一个新的 PWM 周期以开始产生新的占空周期设置的 PWM 波形	在当前 PWM 周期的中央改变输出通道（当计数器值达到 0x0000）
TPMCnVH:L 从非零值变为 0x0000	用以前的占空周期设置来结束当前 PWM 周期	用新的占空周期设置来结束当前 PWM 周期
在 BDM 模式下写 TPMMODH:L 寄存器	清 TPMxMODH:L 寄存器的写一致机制	不清除写一致机制

1 详情参考 14.5.2 节“TPM-计数寄存器”（TPMCNTH:TPMCNTL）[SE110-TPM case 7]

2 详情参考 14.5.5 节“TPM 通道值寄存器（TPMCNTH:TPMCNTL）”

3 详情参考 14.6.2.1 节“输入捕捉模式”

4 详情参考 14.6.2.4 节“中心对齐 PWM 模式”

5 详情参考 14.6.2.4 节“中心对齐 PWM 模式” [SE110-TPM case 1]

6 详情参考 14.6.2.4 节“中心对齐 PWM 模式” [SE110-TPM case 2]

7 详情参考 14.6.2.4 节“中心对齐 PWM 模式” [SE110-TPM case3 和 5]

14.3.1 从TPMV1迁移

除了 14.3 节“TPMV3 与以前版本的区别”以外，当从一个 TPMV1 设备迁移时需要注意一下一些问题。

—当作为 TPMV2 时，不是 TPMV3，定时器不是在输入捕捉模式下，可以写通道值寄存器（TPMCnV）。

—在边沿或中心对齐模式下，只有当定时器由 TPMMOD-1 变到 TPMMOD 时或一个自由运行的计数器从 0xFFFE 变为 0xFFFF 时，通道值寄存器（（TPMCnV）才会被更新。

—同样的，当配置 TPM 模块时，最好在 TPMCnV 以前写入 TPMSC，因为写 TPMxSC 将复位 TPMCnV 的一致性机制。

何时...	行为/最优操作
写通道寄存器（TPMCnV）	定时器必需处于输入捕捉模式
在边沿对齐或中心对齐模式下更新通道值寄存器（TPMCnV）寄存器	仅仅在定时器从 TPMMOD – 1 变为 TPMMOD 或自由运行定时器从 0xFFFE 到 0xFFFF 时才会发生。

通道值寄存器（TPMCnV）的复位一致性机制	写 TPMSC
配置 TPM 模式	先写 TPMSC，然后是 TPMCnV 寄存器

表14-2.迁移到TPMV3时注意事项

14.3.2 特性

TPM 包含以下几个不同的特性：

总共 8 个通道：

— 每个通道可以是输入捕捉、输出比较或边沿对齐 PWM 模式

— 上升沿、下降沿或任何沿输入捕捉触发

— 设置、清除、切换输出比较操作

— PWM 输出上的可选择性

在所有通道上，每个 TPM 可配置为缓冲、中央对齐脉冲宽度调制（CPWM）

每个 TPM 预分频器的钟源可独立选作为总线时钟，固定式系统时钟，或外部引脚：

— 除以 1、2、4、8、16、32、64 或 128 的预分频器值相位

—固定的系统时钟源与总线时钟用过片上同步电路同步

—外部时钟引脚可以与其它定时器通道引脚或者独立的输入引脚分享

16 位自由运行或模数向上/ 向下（CPWM）计数操作

16 位模寄存器来控制计数范围

定时系统启动

每个通道外加一个计数器中断

14.3.3 操作模式

总体上，TPM 通道可以被独立的配置来操作输入捕捉，输出比较，或边沿对齐 PWM 模块。通过一个控制为可使 TPM(所有的通道)转换到中心对齐 PWM 模式。当选择中心对齐模式后，输入捕捉，输出比较，边沿对齐 PWM 功能在此模块的其它通道上均无效。

当微控制器在 BDM 背景调试或 BDM 前台模式时，TPM 临时挂起所有计数直到微控制器回到正常用户操作模式。在 STOP 模式期间，所有系统时钟，包括主振荡器，都停止。因此，TPM 被禁止，直到时钟恢复。在等待模式期间，TPM 继续正常运作。假设 TPM 不需要一个参考时钟或提供中断源需要将 MCU 从等待模式下唤醒，用户在进入等待模式之前可以通过禁止 TPM 功能节省能耗。

输入捕捉模式:

当一个已选择的边沿事件发生在相关的 MCU 引脚，当前 16 寄存器的计数器的值被捕捉到通道寄存器并置相关标志位。上升沿，下降沿，任何沿或没有沿变（禁用通道）可以被选作活动沿来触发输入捕捉。

输出比较

当定时器计数器的值和通道寄存器的值匹配，一个中断位被置位并且 MCU 相关引脚上将强制输出一个选择的行为。输出比较行为可以选择强制引脚为 0，强制引脚为 1，或者忽略引脚之（用于软件定时功能）

边沿对齐 PWM 模式:

16 位模数寄存器的值拉升为 1，置 PWM 输出信号期间为 1。通道值寄存器设置 PWM 输出信号的占空比。用户也可以选择 PWM 输出信号的极性。在周期结束时和在占空比跳变点时允许中断。这种类型的 PWM 信号被称为边沿对齐 PWM，因为 TPM 内所有通道的相位都同时开启。

中心对齐 PWM:

设置 PWM 输出的周期为 16 位模数寄存器值的 2 倍，用通道寄存器的值来设置半占空比时间。该定时器计数器计数一直累加到等于模数寄存器的值，然后倒计时，直至达到零。当计数值和数通道值寄存器值匹配，PWM 输出变为不活动状态。这种类型的 PWM 信号称为中心对齐 PWM，因为所有通道的活动周期的中心以零计数方式对齐。小型电器的马达需要这种类型的 PWM。

这只是表层描述。详见后面的操作模式。

14.3.4 模块图

定时器每个通道对应一个 I/O 引脚有，TPMxCHn （定时器通道 n）表示通道 1-8。定时器 TPM 的 I/O 引脚与通用 I/O 引脚复用 。（参见芯片规范的 I/O 引脚描述）

表 14-2 展示了 TPM 的结构。定时器的核心结构是一个 16 位的计数器，它可以是自由运行的计数器或模数上/下的计数器。TPM 计数器（当工作在正常向上计数模式下）可以为输入捕捉，输出比较，PWM 功能提供时钟。时钟计数器模数寄存器 TPMxMODH:TPMxMODL,控制模数寄存器的值（值的范围在 0x0000 到 0xFFFF）软件可以随时读取计数器的值而不影响计数许序列。不管写入什么数值，任何对 YPMxCNT 的写操作将重置计数器。

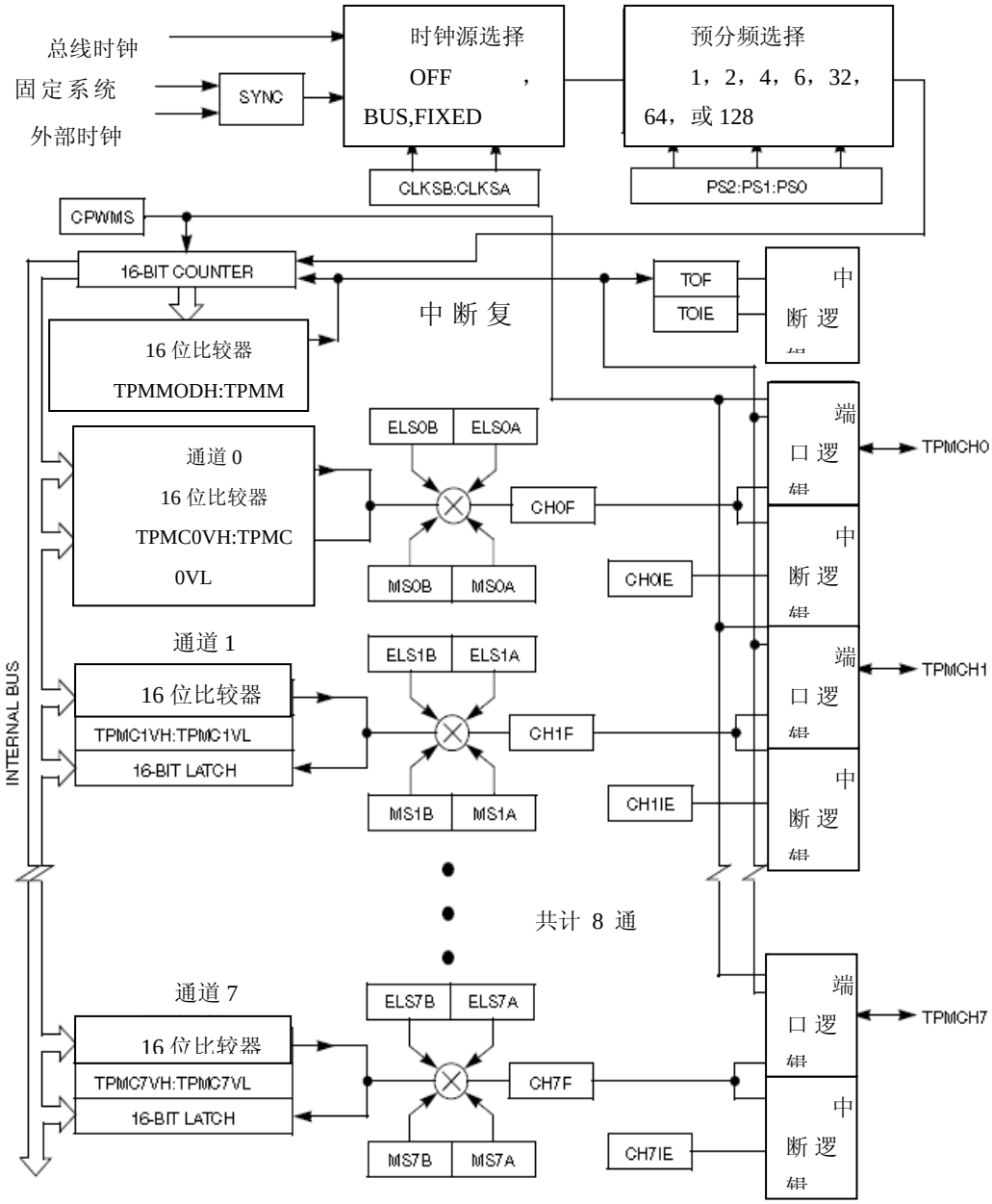


图14-2 TPM模块图

TPM 通道输入捕捉，输出比较，或边沿对齐 PWM 通道是独立可编程的。TPM 可以被配置为所有通道上生成 CPWM 输出。当 TPM 配置为 CPWMs，计数器不能作为向上/向下计数器，输入捕捉，输出比较，EPWM 功能。

如果一个通道作为输入捕捉，内部上拉设备设置可启用该通道。细节如何与引脚控制模块的交互依赖于芯片的实现，因为 I / O 引脚和相关通用 I / O 控制不属于此模块的一部分。请参考芯片规范的 I /O 端口逻辑的讨论。

由于中心对齐 PWM 一般用于驱动 3 相模拟电机和无刷直流电机，它们是运用 3 或 6 通道的典型案例。

14.4 信号描述

表 14-3 展现了用户可以访问的 TPM，通道号由 1-8。当包含外部时钟时，外部时钟可以被任意 TPM 通道的此类引脚共享。然而，它也可以连接到独立的输入引脚上。参考芯片规范

I/O 引脚描述。

表14-3 信号属性

名称	功能
EXTCLK1	可选 TPM 计数器的外部时钟源
TPMxCHn2	I/O 引脚第 n 个通道

1 当预设时，这个信号可以共享任意通道引脚；然而，也可以连接到独立的引脚；这取决于整片的实现

2 n=通道号 1-8

详细信息参考整片文档中对复位状态，断口连接，以及引脚上是否有上拉设备的描述。

TPM 通道引脚可以和通用 I/O 口关联，也可以允许被动的上拉设备。当 TPM 或通用 I/O 口已经配置相关引脚为输入时，此功能由控制为使能。当 TPM 允许用通信引脚时，引脚就被通用 I/O 来控制。包括端口数据和数据方向寄存器。在重置以后，TPM 功能被使能，因此，所有的相关引脚变为 I/O 控制引脚。

14.4.1 详细信号描述

这部分详细描述每个用户访问引脚。尽管 16-1 组合了所有的引脚通道，任意引脚可以与外部时钟源信号共享引脚。由于 I/O 引脚逻辑不是 TPM 的一部分，TPM 引脚交互的详细信息请参考整片文档，I/O 控制大致目的包括端口数据，数据方向和上拉控制。

14.4.1.1 EXTCLK –外部时钟源

定时器状态控制位和控制寄存器允许用户选空（禁用定时器），总线频率时钟（正常时的默认源），晶振时钟，或外部时钟作为时钟。这些时钟用于驱动 TPM 预分频和 16 位 TPM 计数器。在 TPM 内部外部时钟源市同步的。该总线的时钟同步，外部时钟得源频率应该小于总线频率的 1/4 以满足奈奎斯特准则和允许抖动。

外部时钟信号共享相同的引脚作为 I/O 引脚，因此当选择外部时钟源时通道引脚将不会被用作通道 I/O 功能 。用户有义务去避免这样的设置。若这个引脚被用作外部时钟源（CLKSB：CLKSA=1：1），那么通道仍然可以作为软件定时器（ELSnB:ELSnA=0:0）被用于输出比较模式

14.4.1.2 TPMxCHn-TPM 通道n I/O引脚

每个 TPM 通道都对应 MCU 上一个引脚。这个引脚通道配置所决定。TPM 与通用 I/O 共享引脚。每个引脚有一个端口数据寄存器位，数据方向控制位，当引脚工作在输入时，端口允许选择被动上拉。

当（ELSnB:ELSnA=0:0）或（CLKSB：CLKSA=0:0）时，TPM 通道并不控制 I/O 引脚，而是被置位通用 I/O 控制。当 CPWMS=1（并且 ELSnB:ELSnA 不等于 0:0），所有的通道都被[配置成中心对齐 PWM，TPM 系统控制着 TPMxCHn 引脚。当 CPWMS=0，MSnB：MSnA 控制位决定了通道是否被配置成输入捕捉，输出比较，或边沿对齐 PWM。

当通道被配置为输入捕捉（CPWMS = 0，MSnB：MSnA = 0:0 和 ELSnB：ELSnA 不为 0:0），该 TPMxCHn 引脚被迫作为边缘敏感输入引脚。 ELSnB：ELSnA 控制位确定什么极性边缘或哪些边缘会引发输入捕获事件。一种基于总线时钟的同步器用于输入边缘与总线时钟同步。这意味着输入捕捉引脚的最小脉冲带宽每 4 个总线周期可以可靠地被检测到（理想的时钟脉冲下接近两个总线时钟可以被检测到）。TPM 用此引脚作为输入捕捉输入来重写该引脚的端口数据以及数据方向控制寄存器。

当通道被配置成输出比较（SPWMS=0,MSnB:MSnA=0:1 而且 ELSnB:ELSnA 不等于 0:0），相关的数据方向控制被覆盖，TPMxCHn 引脚被认为是输出，ELSnB：ELSnA 控制位决定了怎么控

制此引脚。其余下的三位组合控制 TPMxCHn 引脚，每次 16 位通道值寄存器匹配时钟计数器时决定是否被翻转，清除或置位。

当输出比较翻转模式被选择，引脚上以前的值将被输出直到下一个输出比较时间发生，那时引脚被翻转。

当通道被配置成边沿对齐 PWM（CPWMS=0,MSnB=1 且 ELSnB：ELSnA=0:0），数据方向被覆盖，TPMxCHn 引脚被强制为输出控制。ELSnA 引脚控制 PWM 输出信号的极性。当 ELSnB：ELSnA=1:0，在每个周期的开始（TPMxCHn=0x0000）TPMxCHn 引脚被强制为高.当通道值寄存器与时钟计数器的值匹配时，此引脚被强制为低。当 ELSnA=1，TPMxCHn 引脚在每个周期的开始（TPMxCHn=0x0000）强制为低；当通道值寄存起的值与定时器计数器匹配时，引脚强制为高电平。

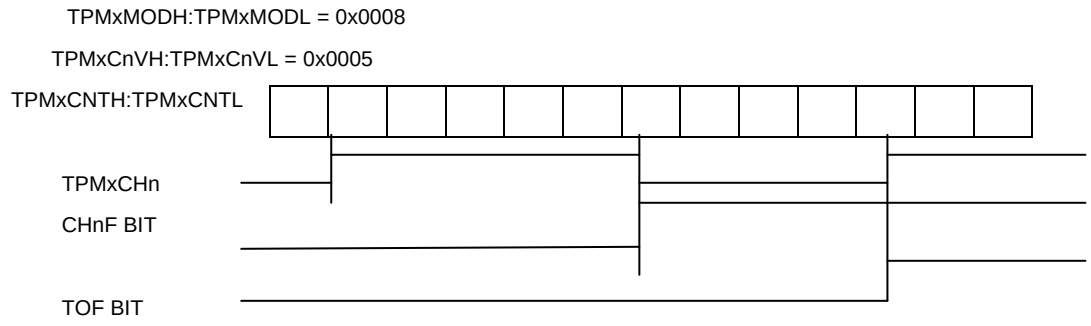


图14-3.边沿对齐的高真脉冲

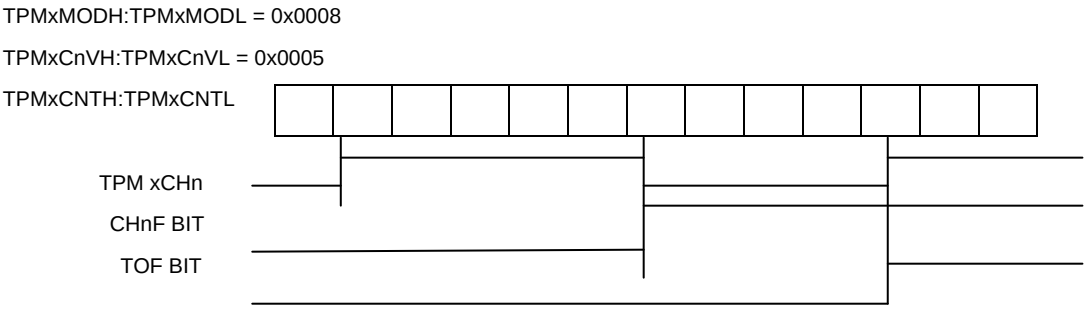


图14-4. 边沿对齐PWM低真脉冲

当 TPM 被配置为中心对齐的 PWM（并且 ELSnB：ELSnA 不等于 0:0），在这 TPM 的所有通道数据方向均被覆盖。TPMxCHn 引脚被迫由 TPM 控制输出。而 ELSnA 位控制每个 TPMxCHn 输出的极性。如果 ELSnB：ELSnA = 1:0，相应 TPMxCHn 脚被清零时，定时器计数器向上计数，和通道的值与定时器计数器匹配;TPMxCHn 引脚被置位，定时器计数器向下计数。通道值寄存与定时器计数器相匹配。如果 ELSnA = 1，当定时器计数器向上计数并和通道值寄存起相匹配，相应的 TPMxCHn 引脚被置位；计时器计数器向下计数并和通道值寄存器匹配，TPMxCHn 引脚被清零。

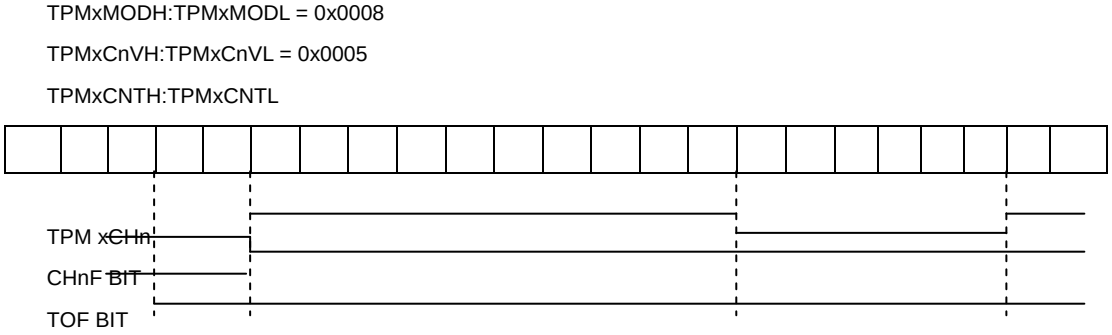


图14-5 中心对齐PWM高真脉冲

TPMxMODH:TPMxMODL = 0x0008

TPMxCnVH:TPMxCnVL = 0x0005

TPMxCNTH:TPMxCNTL

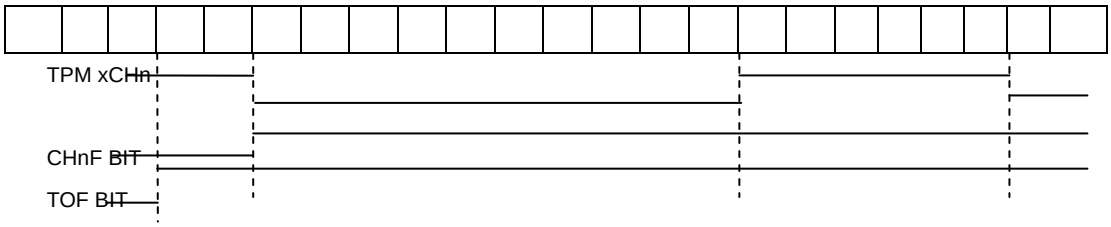


图14-6 中心对齐PWM低真脉冲

14.5 寄存器定义

这部分讲述寄存器地址顺序。一个典型的MCU系统可能有多个TPM，每个TPM有1-8个通道，所以寄存器名字包含地址信息，参考的是哪个寄存器哪个通道。例如，TPMxCnSC意思是定时器（TPM）x，通道n。TPMxCnSC可能是定时器1的通道2的状态和控制寄存器。

14.5.1 TPM状态和控制寄存器（TPMxSC）

TPMxSC包含溢出状态标志和控制位，用于配置中断使能，TPM配置，时钟源，以及预分频因子。在此模块里这些控制设计所有的通道。

	7	6	5	4	3	2	1	0
读 写	TOF	TOIE	CPWMS	CLKSB	CLKSA	PS2	PS1	PS0
复位	0	0	0	0	0	0	0	0

图14-7.TPM状态和控制寄存器（TPMxSC）

表14-4.TPMxSC字段描述

字段	描述
7 TOF	**定时器溢出标志**。在收到 TPM 模数寄存器中的可编程模数值后 TPM 计数器重置为 0X0000，此时重置读写标志位。当 TOF 被置位并且写逻辑值 0 到 TOF 时，通过读取 TPM 状态和控制寄存器清 TOF 标志。若在清除标志完成之前，另一个溢出中断发生了，这个序列将被重置。因此，TOF 仍然可以在清除后被重置。因此，这就要求 TOF 中断在前一个 TOF 清除序列期间不被丢失。重置清零 TOF。写一个逻辑 1 到 TOF 是无效的。 0 TPM 计数器值未到达模数值或溢出 1 TPM 计数器溢出了
6 TOIE	**定时器溢出中断使能**。读写此位使能定时器的中断。若 TOIE 被置位，当 TOF 等于 1，将产生中断。重置清除 TOIE 0 TOF 中断禁止（用软件轮询） 1 TOF 中断使能
5 CPWMS	**选择中心对齐 PW**。当出现，读写位选 CPWM 操作模式。默认情况下，TPM 运行在输入捕捉，输出比较，边沿对齐 PWM 的向上计数模式。设置 CPWM 重新配置 TPM 使之运行在 CPWM 的向下计数模式。重新清除 CPWMS。 0 所有通道运行在输入捕捉，输出比较，边沿对齐 PWM 模式。由每个通道的状态和控制寄存器的 MSnB：MSnA 位控制 1 所有通道运行在中心对齐的 PWM 模式下
4-3 CLKS[B:A]	**时钟源选择**。如表 14-4 中所示，这两个字段被用来禁止 TPM 或选择三种时钟源中的一种来驱动预分频器。固定系统时钟源只有在基于锁相环的系统中才有意义。当没有锁相环 PLL，固定系统时钟和总线时钟一样。外部时钟通过 TPM 模块和总线时钟保持同步；而固定时钟源（当有 PLL 时）是通过偏上同步电路实现与总线的同步。当有 PLL 电路，但是没有被使能时，固定系统时钟源和总线频率时钟一样。
2-0 PS[2:0]	**预分频选择**。这三位字段选择八个分频因子中的一个，如表 14-4 所示。分配器是在时钟源同步或时钟源选择之后被定位，因此，它影响着选择的驱动系统的时钟源。当新的值被更新到寄存器位后，新预分频因子将影响鲜一个系统时钟的时钟源。

表14-5. TPM时钟源选择

CLKSB：CLKSA	预分配器输入TPM时钟源
00	选空（TPM禁止）
01	总线速率时钟
10	固定系统时钟
11	外部时钟源

表14-6. 预分频因子选择

PS2：PS1：PS0	TPM时钟源
000	1
001	2
010	4
011	8
100	16

101	32
110	64
111	128

14.5.2 TPMM计数寄存器（TPMxCNTH: TPMxCNTL）

两个TPM只读计数寄存器包含了TPM中的计数器高、低字节的值。 读任意（TPMxCNTH或TPMxCNTL）字节锁存这两字节的内容到缓冲区直到另一半被读取否则一直被锁定。这使得不管是在大端还是小端顺序中读取一致的16位，对各种编译器的实现都十分友好。通过重置MCU或任何些定时器状态控制寄存器（TPMxSC），这种一致的读取机制将自动重新启动。

重置清除TPM的计数寄存器。写任意值到TPMxCNTH或TPMxCNTL也将清除TPM的计数器（TPMxCNTH：TPMxCNTL）并且重置协作机制，不管写入什么数据。

图 14-8. TPM计数寄存器高字节（TPMxCNTH）

	7	6	5	4	3	2	1	0
读	Bit15	14	13	12	11	10	9	Bit8
写	任何对TPMxCNTH的写入将清除16位计数器							
复位	0	0	0	0	0	0	0	0

	7	6	5	4	3	2	1	0
读	Bit7	6	5	4	3	2	1	Bit0
写	任何对TPMxCNTH的写入将清除16位计数器							
复位	0	0	0	0	0	0	0	0

图 14-9. TPM计数寄存器低字节（TPMxCNTL）

当BDM模式被激活，定时器计数将被冻结（这就是用户所读到的值）；即使在BDM模式下读取一个或两个半字节。一致性机制被冻结，以至于缓冲区在BDM处于活动状态时锁存它们的状态。这可确保如果用户正在读取一个16位寄存器的半个字节时，BDM此时被活跃，恢复正常后它将读取另外一半相应的值。

在BDM模式下，写入任何值到TPMxSC，TPMxCNTH或TPMxCNTL将重置寄存器读一致性机制的TPMxCNTH：L寄存器，无论写入什么数据。

14.5.3 TPM计数器模数寄存器（TPMxMODH:TPMxMODL）

读/写TPM模数寄存器包含TPM计数器的模数值。TPM计数器达到的模数值后，在下一个时钟TPM的计数器恢复从0x0000计数，并且溢出标志（TOF）被置位。写入TPMxMODH或TPMxMODL禁止TOF位和溢出中断直到其他字节写入。重置TPM的计数器模数寄存器为0x0000，使得定时器在自由运行（模数禁用）。

不管写入任何字节，（TPMxMODH或TPMxMODL）锁存到缓冲区中的值并且根据CLKSB：CLKSA位，寄存器都更新它们的值到写缓冲区，因此：

如果（CLKSB：CLKSA = 0:0），那么第二个字节写入时寄存器将更新

如果（CLKSB：CLKSA不为 0:0），那么两个字节都写入后，寄存器更新并且TPM计数器由（TPMxMODH：TPMxMODL－1）变成（TPMxMODH： TPMxMODL）。如果TPM计数器是自由运行计数器，当TPM 计数器由0xFFFE变为 0xFFFF时，定时器将更新。

锁存机制可以通过写TPMxSC地址手工重置（而不管BDM是否激活）

当BDM处于激活状态时，即使在BDM模式下模数寄存器被写入一个或两个半字节，一致性

机制被冻结（除非通过写入TPMxSC寄存器方式复位）以至于缓冲锁存它们在BDM处于活动状态时的状态。当BDM出于激活状态，任何写入模数寄存器的数据将不会被缓冲区锁存并直接写入模数寄存器。

	7	6	5	4	3	2	1	0
读 写	Bit15	14	13	12	11	10	9	Bit8
复位	0	0	0	0	0	0	0	0

图14-10 TPM计数器模数寄存器高字节（TPMxMODH）

	7	6	5	4	3	2	1	0
读 写	Bit7	6	5	4	3	2	1	Bit0
复位	0	0	0	0	0	0	0	0

图14-11 TPM计数器模数寄存器高字节（TPMxMODL）

重置TPM模数寄存器前复位TPM计数器以避免关于何时第一个计数器溢出的冲突。

14.5.4 TPM通道n状态和控制寄存器（TPMxCnSC）

TPMxCnSC包含通道中断状态标志和控制位用于配置中断使能，通道配置，引脚功能。

	7	6	5	4	3	2	1	0
读	CHnF	CHnIE	MSnB	MSnA	ELSnB	ELSnA	0	0
写	0							
复位	0	0	0	0	0	0	0	0

=未执行或保留位

图 14-12. TPM通道n状态和控制寄存器（TPMxCnSC）

表14-7. TPMxCnSC字段描述

字段	描述
7 CHnF	**通道 n 标志**—通道 n 被配置为输入捕获时，当一个活动边沿在通道 n 管脚出现时，此标志位被置位。通道 n 为一个输出比较或边沿排列 PWM 通道时，当 TPM 计数寄存器中的值与 TPM 通道 n 数值寄存器中的植匹配时，CHnF 被置位。此标志很少用于中心排列 PWM，因为每当计数器与通道数值寄存器匹配时，对应于活动周期的两个沿，该标识都将被置位。当 CHnF 被置位且中断被使能（CHnIE=1）时，会进行相应的中断请求。当 CHnF 被置位时，通过读 TPMCnSC，然后向 CHnF 写 0，可清除 CHnF。如果在清除操作完成之前出现另一个中断请求，清除操作被复位，因此，CHnF 将保持置位状态直到对前一 CHnF 的清除操作完成。这样可以使得一个 CHnF 中断请求不会因为清除前一个 CHnF 而丢失。复位可清除 CHnF。对 CHnF 写 1 无效。 0 在通道 n 无输入捕获或输出比较事件发生。 1 在通道 n 发生输入捕获或输出比较事件。
6 CHnIE	**通道 n 中断使能**—该读/写位使能来自通道 n 的中断。复位可清除 CHnIE。 0 通道 n 中断请求禁止（使用软件轮询） 1 通道 n 中断请求使能
5 MSnB	**为 TPM 通道 n 模式选择 B**—当 CPWMS=0 时，MSnB=1 配置 TPM 通道 n 为边沿排列 PWM 模式。通道模式和设置控制的概述，参照表 14-6。

4 MSnA	**TPM 通道 n 模式选择 A**—当 CPWMS=0 时，MSnB=1 配置 TPM 通道 n 为输入捕捉或输出比较模式。通道模式和设置控制的概述，参照表 14-6 注意：如果变为输入捕捉模式之前，关联的端口引脚大于两间总线时钟周期不稳定， 模式，有可能得到出乎意料的边沿触发迹象。
3-2 ELSnB ELSnA	**跳变沿/输出电平选择位**—根据 CPWMS:MSnB:MSnA 设置的定时器通道的运行方式，如表 10-5 所示，这些位可选择触发输入捕获事件的输入边沿极性、选择响应输出比较匹配的驱动电平、或选择 PWM 输出的极性。 将 ELSnB:ELSnA 设置为 0:0 可将相关定时器管脚配置为通用 I/O 管脚，而与定时器通道功能无关。当相关定时器通道设置为无需使用管脚的软件定时器时，此功能典型应用于暂时禁止一个输入捕获通道，或当 TPMCH0 管脚用作外部时钟输入时，此时并不需要用到引脚，可以使得定时器管脚成为可用的通用 I/O 管脚。

表 14-8 模式、边沿、级别选择

CPWMS	MSnB:MSnA	ELSnB:ELSnA	模式	配置
x	xx	00	不是TPM引脚-而是通用I/O或别的外围控制	
0	00	01	输入捕捉	仅上升沿捕捉
		10		仅下降沿捕捉
		11		上升或下降沿捕捉
	01	01	输出比较	触发器比较输出
		10		清除比较输出
		11		置位比较输出
	1x	10	边沿对齐PWM	高真脉冲（清除比较输出）
		X1		低真脉冲（置位比较输出）
1	xx	10	中心对齐PWM	高真脉冲（清除输出compare-UP引脚）
		X1		低真脉冲（置位输出compare-UP引脚）

14.5.5 TPM通道值寄存器（TPMxCnVH:TPMxCnVL）

这些可读/写寄存器包含在输入捕捉时捕捉TPM计数器值或在输出比较时的比较值或PWM功能。当复位时所有通道会被清零。

	7	6	5	4	3	2	1	0
读 写	Bit15	14	13	12	11	10	9	Bit8
复位	0	0	0	0	0	0	0	0

图 14-13 TPM通道值寄存器高字节（TPMxCnVH）

	7	6	5	4	3	2	1	0
读 写	Bit7	6	5	4	3	2	1	Bit0
复位	0	0	0	0	0	0	0	0

图 14-14 TPM通道值寄存器高字节（TPMxCnVL）

在输入捕捉模式，读取任意（TPMxCnVH或TPMxCnVL）一字节将锁存其内容到缓冲区，直到另一半被读取才解除锁定。当TPMxCnSC寄存器被写入（不论BDM模式是否有效）这锁存机制也会被重置（变成解锁）。在输入捕捉模式下，任何对通道寄存器的写入将被忽略。

当BDM是有效的，一致性机制将被冻结。（除非通过对TPMxCnSC寄存器写入产生复位）即使在BDM模式下读取一个或两个半字节，一致性机制也会被冻结，以至于缓冲区锁存它们在BDM处于活动状态时的状态。这可确保如果用户是在读取16位寄存器时BDM被激活，恢复正常执行后它将从中读取另外半字节值。在BDM模式下从TPMxCnVH:TPMxCnVL寄存器读取的值，是它们寄存器的值而不是缓冲区中的值。

在输出比较或PWM模式，写入（TPMxCnVH或TPMxCnVL）的任意字节，值将锁存到缓冲区。当两个字节都被写入后，它们作为一致对应的16位值，根据CLKSB：CLKSA位的值和选定的模式转入定时器通道寄存器。因此：

如果（CLKSB：CLKSA = 0:0），那么当第二个字节写入时，寄存器被更新。

如果（CLKSB：CLKSA不为 0:0并处于输出比较模式），则第二个字节写入和TPM的计数器变化（预分频器计数结束时），寄存器备更新。

如果（CLKSB：CLKSA不为 0:0并处于EPWM或CPWM模式），则当两个字节都被写入，以及TPM的计数器从（TPMxMODH：TPMxMODL－1）变至（TPMxMODH：TPMxMODL），更新寄存器值。如果TPM计数器是一个自由运行计数器，当TPM计数器由0xFFFE 到0xFFFF时，那么寄存器将被更新。

可通过写入TPMxCnSC寄存器（不论BDM 模式有效）手动复位锁存机制。不管是大端或小端顺序这种机制允许锁定一致连续的16位，对各种编译器的实现都很友好。

当BDM处于激活状态时，即使在BDM模式下模数寄存器被写入一个或两个半字节，一致性机制被冻结（除非通过写入TPMxSC寄存器方式复位）以至于缓冲锁存它们在BDM处于活动状态时的状态。当BDM处于激活状态，任何写入模数寄存器的数据将不会被缓冲区锁存并直接写入模数寄存器。当BDM有效时，一旦恢复正常执行，写入通道寄存器的值被用于在PWM和输出比较操作。当BDM是有效时，写入通道寄存器不干预一致性序列部分完成。一致性机制已完全执行，用用户写入的值更新通道寄存器（BDM没有被激活）。

14.6 功能描述

所有TPM的功能与允许灵活的时钟选择源和预分频因子的中央16位计数器关联。还有一个16位模数寄存器与中央计数器关联。

CPWMS控制位T为TPM（CPWMS = 1）的所有通道或通用定时功能（CPWMS = 0）选择中心对齐PWM操作，其中每个通道都可以独立的配置为在输入捕捉，输出比较，或边沿对齐PWM模式。CPWMS 控制位位于主TPM的状态和控制寄存器，因为它影响到位于TPM内所有通道且影响主要寄存器的操作方式。（在CPWM模式，改变向上/向下模式而不是用于通用定时器功能的向上计数模式）。

以下各节描述了主要计数器和每个定时器的操作模式（输入捕捉，输出比较，边沿对齐PWM，和中心对齐脉宽调制）。由于管脚操作细节和中断活动取决于操作模式，这些议题将包括在相关的模式说明部分。

14.6.1 计数器

计数器所有的功能都是基于16位计数器（TPMxCNTH:TPMxCNTL）。这个部分讨论了时钟源的选择，计数器溢出，向上计数模式和上/下计数模式以及手工重置计数器。

14.6.1.1 计数器时钟源

定时器状态和控制寄存器中的CLKSB:CLKSA两位字段选择了三个可能的时钟源或OFF（定时器禁止）之一。详见表 14-3。在任何MCU复位后，CLKSB:CLKSA=0:0，因此没有选择时钟源。TPM处在低功耗状态。这些控制位可以在任何时候被读/写。禁止定时器（写00到CLKSB:CLKSA）并不影响计数器的值和别的定时器。

CLKSB:CLKSA	分频器输入的TPM时钟源
00	没选择时钟（禁止计数器）
01	总线时钟
10	固定系统时钟
11	外部时钟

表14-9 TPM时钟源选择

总线速率时钟是微控制器的主系统总线时钟。这时钟源不需要同步，因为它用于所有的MCU内部活动，包括CPU操作的时钟和总线。

在无锁相环PLL或PLL不工作的微控制器中，固定系统时钟源和总线速率时钟源是相同的，并且它不经过同步器。当锁相环存在并工作，独立时钟源的晶振和定时器计数器之间需要同步器，使得计数器切换能够正确的和总线时钟切换对齐。同步器将用于在芯片级来同步晶体源有关的时钟与总线时钟。

外部时钟源，可连接到TPM的任何通道引脚。这时钟源总是通过同步器，以确保定时器切换正确的与总线时钟切换对齐。总线速率时钟驱动同步器，因此，为满足奈奎斯特准则即便外部时钟频率有抖动，时钟频率不得高于总线税率的1/4。理想情况下，外部时钟可以等于总线时钟的1/4。

当外部时钟源分享TPM的通道引脚，该引脚不能用于其他通道定时功能。例如，输入捕捉时通道配置为0，而TPM的通道0脚也被用作定时器外部时钟源，这将模棱两可。（这是用户的责任为了避免这种设置。）TPM的通道仍可以在输出比较模式下用于软件模式比较定时功能（引脚控制设置并不影响TPM的通道引脚）。

14.6.1.2 计数器溢出和模数复位

一个中断标志和使能都和16位的主要计数器关联。标志（TOF）是一个软件访问标记，表明定时器计数器溢出。使能信号在软件轮询（TOIE = 0），没有硬件中断产生或中断驱动操作（TOIE = 1），当标志等于1时将产生一个静态的硬件产生中断，之间的选择。

14.6.1.3 计数模式

主要定时器计数器有两种计数模式。当中心对齐的PWM选择（CPWMS = 1），计数器工作在上/下计数模式。否则，计数器作为一个简单的向上计数模式运行。作为一个向上计数器，定时器计数器计数从0x0000到其终端数值，然后继续返回0x0000计数。终端数值为0xFFFF或TPMxMODH：TPMxMODL的模块值。

当中心对齐PWM工作模式被指定，计数器从0x0000到其终端数值，然后又返回为0x0000开始向上计数。0x0000和终端数值都是正常长度计数（1个定时器时钟周期）。在这种模式下，定时器溢出标志（TOF）在终端数周期的结束时被置位（和计数变化到下一个较低的计数值一样）。

14.6.1.4 手动计数器复位

通过写任意值到TPMxCNTH或TPMxCNTL的任何部分，任何时候主定时器计数器可以手动重置。重置计数器之前只读取到一半的值时，以这种方式重置计数器同时重置一致性机制。

14.6.2 通道模式选择

假设CPWMS=0，通道n的状态控制寄存器中的MSnB以及MSnA控制位决定了通信通道的基本操作模式。可以选择输入捕捉，输出比较和边沿对齐PWM。

14.6.2.1 输入捕捉模式

在输入捕捉模式下，TPM可以捕捉到事件发生的时刻。当输入捕捉通道由活动沿变发生时，TPM所存TPM计数器的内容到通道值寄存器（TPMxCnVH:TPMxCnVL）中。可以选择上升沿，下降沿或任何沿作为活动沿用与触发一次输入捕捉事件。

在输入捕捉模式下，TPMxCnVH 和 TPMxCnVL寄存器均为只读的。

当16位寄存器被读取一半时，另外一半被锁存到缓冲区中以支持大端或小端顺序的一致连续的16位访问。通过读通道状态和控制寄存器（TPMxCnSC），一致连续的序列可以被手工复位。

一次输入捕捉事件会重置CHnF标志位，这会可选择的产生CPU中断请求。

当处于BDM模式下时，输入捕捉功能如用户配置的那样工作。当外部事件发生时，TPM锁存TPM计数器的值（由于在BDM下而被禁止）到通道寄存器并置相应标志位。

14.6.2.2 输出比较模式

输出比较功能时，TPM 可产生计时脉冲，这些计时脉冲是位置、极性、持续时间和频率可编程的。当计数器达到一个输出比较通道的通道数值寄存器内的值时，TPM 能设置、清除或连接通道管脚。

在输出比较模式下，只有在一个16位寄存器的两个8位字节都被写入后，数值才会被转移至相应的定时器通道数值寄存器。按照CLKSB:CLKSA的值，因此：

若CLKSB:CLKSA=0:0，当第二个字节被写入时，寄存器值被更新。

若CLKSB:CLKSA不为0:0，在第二个字节被写入后，在TPM计数器的下一次变化时（预分频计数结束时）更新寄存器值。

通过重写通道状态和控制寄存器，一致连续的序列可以被手动复位。

一个输出比较事件可设置一个标志位（CHnF），该标志位可随意产生一个 CPU 中断请求。

14.6.2.3 边沿对齐PWM模式

这一 PWM 输出类型使用计时器的正常增计数模式（CPWMS = 0），且当同一 TPM 中的其它通道配置为输入捕获或输出比较功能时可用。该 PWM 信号的周期由模寄存器（TPMMODH:TPMMODL）的设置情况确定。占空系数由定时器通道数值寄存器（TPMCnVH:TPMCnVL）的设置情况确定。PWM 信号的极性由 ELSnA 控制位的设置情况确定。占空系数的可能值在 0% 与 100% 之间。

如图 14-15 所示，TPM 通道寄存器中的输出比较数值确定了 PWM 信号的脉冲宽度（占空系数）。模溢出和输出比较之间的时间是脉冲宽度。如果 ELSnA=0，计数器溢出强制 PWM 信号为高电平，输出比较强制 PWM 信号为低电平。如果 ELSnA=1，计数器溢出强制 PWM 信号为低电平，输出比较强制 PWM 信号为高电平。

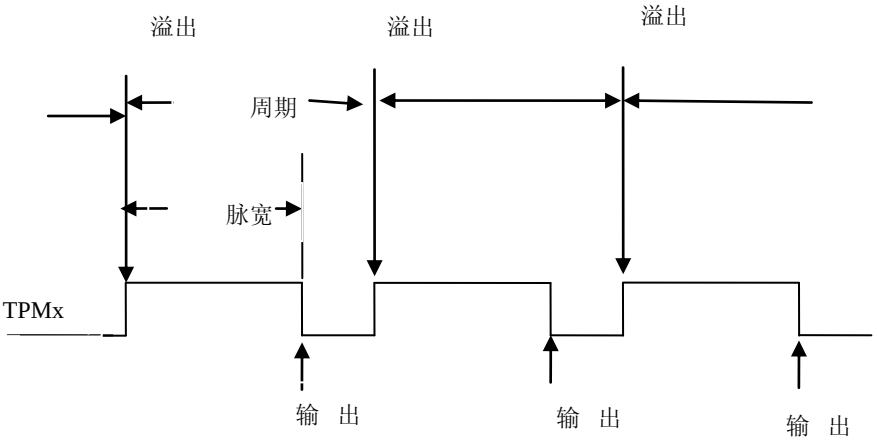


图14-15 PWM周期和脉冲带宽 （ELSnA=0）

当通道数值寄存器设置为 0x0000 时，占空系数为 0%。通过将定时器通道数值寄存器（TPMCnVH:TPMCnVL）设置为一个大于所设模值的数值，可使占空系数达到 100%。这表示，为了获得 100% 的占空系数，设置的模值必须小于 0xFFFF。

因为定时器经常被用于 8 位 MCU，定时器通道寄存器的设置被缓存起来，以确保连续 16 位数据更新，并避免出现意外的 PWM 脉冲宽度。写 TPMCnVH 或 TPMCnVL 中的任意一个寄存器，也就是写缓冲寄存器。在边沿对齐模式下，只有在一个 16 位寄存器的两个 8 位字节都被写入后，按照 CLKSB:CLKSA 位，计数值被转移至相应的定时器通道寄存器。因此：

若CLKSB：CLKSA=0:0，当第二个字节被写入时，寄存器值被更新。

若CLKSB：CLKSA不为0:0，则当两个字节都被写入，以及TPM的计数器从（TPMxMODH：TPMxMODL – 1）变至（TPMxMODH：TPMxMODL），更新寄存器值。如果TPM计数器是一个自由运行计数器，当TPM计数器由0xFFFE 到0xFFFF时，那么寄存器将被更新。

14.6.2.4 中心排列 PWM 模式

此类 PWM 输出使用计时器的增/减计数模式（CPWMS=1）。TPMCnVH:TPMCnVL 中的输出比较值决定 PWM 信号的脉冲宽度（占空系数），周期由 TPMMODH :TPMMODL 中的值确定。

TPMMODH:TPMMODL 中的值应该保持在范围 0x0001 至 0x7FFF 内，因为在此范围外的值会产生模糊的结果。ELSnA 将确定 CPWM 输出的极性。

pulse width =2 x （TPMCnVH:TPMCnVL） Eqn. 16-1

period = 2 x （TPMMODH:TPMMODL）;

for TPMMODH:TPMMODL = 0x0001–0x7FFF Eqn. 16-2

如果通道数值寄存器 TPMCnVH:TPMCnVL 为 0 或负值（第 15 位被置位），占空系数将为 0%。如果 TPMCnVH:TPMCnVL 为正值（第 15 位被清 0）且大于设置的模值（非零），占空系数为 100%，因为不会发生占空系数比较。这表明，通过模寄存器设置的可用周期范围在 0x0001 与 0x7FFE（如果不必产生 100% 的占空系数可为 0x7FFF）之间。这不是一个重要的限制，因为结果周期远远比正常应用所需要的周期长。

TPMMODH:TPMMODL=0x0000 为一不能被用于中心排列 PWM 模式的特殊情况。当 CPWMS=0 时，这种情况对应计数器从 0x0000 计数至 0xFFFF；但是当 CPWMS=1 时，计数器需要一个与模寄存器匹配的有效值，用于在非 0x0000 的某一位置从增计数转向为减计数。

图 10-12 所示为 TPM 通道寄存器中的输出比较值（乘以 2），该值确定了 CPWM 信号

的脉冲宽度（占空系数）。如果 ELSnA=0，当向上计数强制 CPWM 输出信号为低时或者当向下计数强 CPWM 输出信号为高时，比较得到匹配。计数器向上计数至 TPMMODH:TPMMODL 中设置的值，然后向下计数至 0。这将周期设置等于 TPMMODH:TPMMODL 的两倍。

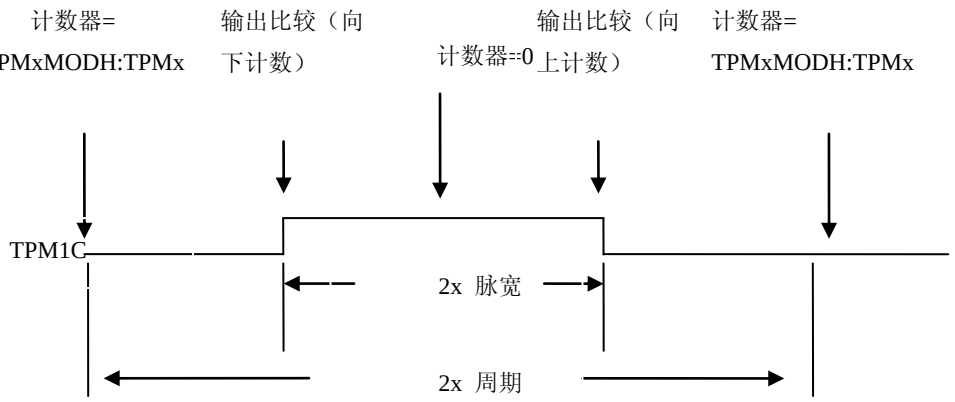


图 14-16. CPWM 周期和脉冲宽度（ELSnA = 0）

中心排列 PWM 输出产生的噪声小于各种边沿排列 PWM，因为在同一系统时钟边沿，排列的 I/O 传送管脚更少。对于一些类型的电动机，此类 PWM 也是需要的。

当计数器工作在上/下计数模式时，输入捕捉，输出比较以及边沿对齐PWM功能都不能被感知。因此这意味着当CPWMS=1时，一个TPM中所有活动的通道应该被用在CPWM模式下。

TPM 可以用在 8 位 MCU 中，定时器通道寄存器中的设置被缓存下来，以保证连续 16 位数据更新，并避免意外的 PWM 脉冲宽度出现。对寄存器 TPMMODH、TPMMODL、TPMCnVH 和 TPMCnVL 中的任意一个进行写操作，实际是对缓冲寄存器进行写操作。

在中心对齐模式下，按照 CLKSB：CLKSA 位，TPMCnVH:L 寄存器随着写入缓冲区的值变化而更新，因此：

若CLKSB：CLKSA=0:0，当第二个字节被写入时，寄存器值被更新。

若CLKSB：CLKSA不为0:0，则当两个字节都被写入，以及TPM的计数器从（TPMxMODH：TPMxMODL – 1）变至（TPMxMODH：TPMxMODL），更新寄存器值。如果TPM计数器是一个自由运行计数器，当TPM计数器由0xFFFE 到0xFFFF时，那么寄存器将被更新。

另外，当 TPMCNTH:TPMCNTL=TPMMODH:TPMMODL 时，TPM 能在此次计数的结尾产生一个TOF中断。

写 TPMSC 可取消写入 TPMMODH 和/或 TPMMODL 中的任意值，并为模寄存器中的关联装置复位。写 TPMCnSC 可取消写入通道数值寄存器中的任意值，并为 TPMCnVH:TPMCnVL 中的关联装置复位。

14.7 复位总览

14.7.1 综述

一旦MCU复位发生时，TPM复位。

14.7.2 复位操作描述

重置清除TPMxSC寄存器，禁用TPM时钟和禁用定时器溢出中断（TOIE = 0）。 CPWMS，MSnB，MSnA，ELSnB和ELSnA都被清除。这些是配置TPM所有通道的输入捕获相关管脚与I/ O操作引脚逻辑断开（因此，微控制器中所有与TPM相关的引脚设置为通用I/O引脚）。

14.8 中断

14.8.1 综述

TPM 为主寄存器一处产生一个可选中断，并为每个通道产生一个中断。通道中断的意义取决于每个通道采用的运行模式。如果通道配置为输入捕获，每当所选输入捕获边沿被识别时，中断标志被置位。如果通道配置为输出比较或 PWM 模式，每当主计时器与16位通道数值寄存器中的值匹配时，中断标志被置位。

在表14-10中列出的所有TPM中断，列出了任何能阻止离开TPM的中断要求，并得到独立的中断处理逻辑认可的局部启动的中断的名称。

中断	局部启动	源	描述
TOF	TOE	计数器溢出	在每次计数器值等于中断值时被置位（在切换到下一个计数值时，一般为0X0000）
CHnF	CHnIE	通道事件	在通道n上，发生一次输入捕捉或输出比较事件

表14-10 中断概要

TPM模块将提供一个高真中断信号。在芯片集成时，中断模块定义了向量和优先级。因此，完整的文档资可以参考用户中断模块或芯片规范

14.8.2 中断操作描述

对TPM中每个断源，当中断条件被识别，则相应标志位被置位。如：定时器溢出，通道输入捕捉，或输出比较事件。此标志可以由软件读取（轮询）以确定发生的事件或相关使能位（TOIE或CHnIE）被置位以便于使能硬件中断发生。一旦中断使能为被置位，当相关中断标志等于1时，将会产生一个静态中断。用户软件应该在中断从中断例程返回之前执行一个序列的步骤去清除中断标志。

TPM的中断标志是由两个步骤清除的，包括一个标志位读取数据的同时它被置位（1）随后写入零（0）到此位来清除标志。如果在这两个步骤之间检测出一个新的事件，则复位该序列，在第二个步骤之后，中断标志保持置位，以避免丢失可能的新发生的事件。

14.8.2.1 定时器溢出中断（TOF）描述

TOF中断的意义以及详细操作在不同的操作模式下有略微的区别(通用I/O对比中心对齐PWM）。通过上述步骤可清除相应标志位。

14.8.2.1.1 正常情况

正常的当定时器计数其从0xffff变化到0x0000时TOF被置位。TPM不是工作在中心对齐模式下（CPWMS=0），当定时器计数器从中断数值变为0x0000时，TOF被置位。这种情况下对应于正常意义的定时器溢出。

14.8.2.1.2 中心对齐PWM情况

当CPWMS=1,定时器计数器在终端数值（模数寄存器中的值）由向上计数变成向下计数，TOF将被置位。在这种情况下，TOF对应于PWM周期的最后时刻。

14.8.2.2 输入捕捉事件

通道中断的意义取决于所处的工作模式。（输入捕捉，输出比较，边沿对齐PWM，中心对齐PWM）

14.8.2.2.1 输入捕捉事件

当通道被配置成一个输入通道，ELSnB:ELSnA控制位选择无边沿（OFF），上升沿，下降沿或任何沿作为触发事件的沿。当所选择的沿被发现后，中断标志碑置位。通过16.6.2“中断操作描述”的说明来清除标志位。

14.8.2.2.2 输出比较事件

当一个通道被配置为输出比较通道,每次主定时器的计数值和通道值寄存器的16位匹配时，中断标志被置位。通过16.6.2“中断操作描述”的两步操作可清除标志位。

14.8.2.2.3 PWM占空周期结束事件

对于通道被配置为PWM操作有两种可能性。当通道被配置为边沿对齐脉PWM，定时器计数器与通道值寄存器匹配，通道标志位被置位。通道值寄存器标记着现役的占空循环周期的结束。当通道配置为中心对齐PWM，则每个PWM周期，计时器计数通道值与通道值寄存器匹配两次。在此CPWM 情况下，当定时器计数器的值与通道值寄存器匹配，在现役周期的开始和结束时，通道标志设置被置位。通过19.6.2节所述的“中断操作描述“两步操作，该标志可被清除。

1. 写 TPMxCnTH:L 寄存器（14.5.2 节,“TPM 计数寄存器（TPMCNTH:TPMCNTL）”）[SE110-TPM case 7]
2. 读 TPMxCNTH:L 寄存器（14.5.2 节,“TPM 计数寄存器（TPMCNTH:TPMCNTL）”）
   —在 TPMV3，在 BDM 模式任何读 TPMxCNTH:L 操作将回被冻结的 TPM 计数器值。在 TPMV2，在 BDM 模式激活前，如果仅仅只读取 TPMxCNTH:L 寄存器的一个字节，那么在 BDM 模式下的任何对 TPMxCNTH:L 寄存器的读操作将返回 TPMxCNTH:L 对应缓冲区的锁存值，而不是被冻结的 TPM 计数器值。
   —在 BDM 模式下，如果写 TPMxSC, TPMxCNTH 或 TPMxCNTL 寄存器，读一致性机制将被清除。相反，在 TPMV2 中并不清除这一机制。
3. 读取 TPMxCnVH:L 寄存器（14.5.5 节,“TPM 通道值寄存器”（TPMCnVH:TPMCnVL））
   —在 TPMV3 中，在 BDM 模式下，任何对 TPMxCnVH:L 寄存器的读操作将返回 TPMxCnVH:L 寄存器的值。在 TPM v2 中，在 BDM 模式激活之前如果仅仅只读取 TPMxCnVH:L 寄存器一个字节，那么在 BDM 模式下任何对 TPMxCnVH:L 的读操作将返回锁存在缓冲区中的 TPMxCnVH:L 值，而不是 TPMxCnVH:L 寄存器的值。
   —在 BDM 模式下，如果写 TPMxSC 寄存器，读一致性机制将被清除。反，在 TPMV2 中并不清除这一机制。
4. 写 TPMxCnVH:L 寄存器
   —输入捕捉模式（14.6.2.1 节，“输入捕捉模式”）
   在此模式下 TPMV3 不允许写 TPMxCnVH:L 寄存器。相反，TPMV2 可以允许写操作。
   —输出比较模式（14.6.2.2 节，“输出比较模式”）
   在此模式下，如果 CLKSB:CLKSA 不为 0:0），TPMV3 将用写缓冲区的值更新 TPMxCnVH:L 寄存器，这在 TPM 计数器的第二个字节被写入后的下一次改变时（即预分频技术结束时）更新。相反，TPMV2 将总是在第二个字节被写入后更新寄存器。
   —边沿对齐 PWM（14.6.2.3 节，“边沿对齐 PWM 模式”）

在此模式下且若（CLKSB:CLKSA 不为 00），TPMV3 将用其写缓冲寄存器的值更新 TPMxCnVH:L 寄存器。这在寄存器两字节均被写入且定时器从 TPMxMODH:L-1 变为 TPMxMODH:L 时发生。若定时器为自由运行定时器，并从 0xFFFE 到 0xFFFF 时，也将更新寄存器。相反，在 TPMV2 总是在两个字节都被写入且 TPM 计数器从 TPMxMODH:L 变为 0x0000 时更新。

—中心对齐 PWM（14.6.2.4，“中心对齐 PWM 模式）

在此模式下且若（CLKSB:CLKSA 不为 00），TPMV3 将用其写缓冲寄存器的值更新 TPMxCnVH:L 寄存器。这在寄存器两字节均被写入且定时器从 TPMxMODH:L–1 变为 TPMxMODH:L 时发生。若定时器为自由运行定时器，并从 0xFFFE 到 0xFFFF 时，也将更新寄存器。相反，在 TPMV2 总是在两个字节都被写入且 TPM 计数器从 TPMxMODH:L 变为 TPMxMODH:L-1 时更新。

5. 中心对齐 PWM（14.6.2.4 节，“中心对齐 PWM 模式”）

—TPMxCnVH:L = TPMxMODH:L [SE110-TPM case 1]

在此情况下，TPMV3 产生 100%占空周期。相反，TPMV2 产生 0%占空周期。

—TPMxCnVH:L = （TPMxMODH:L - 1） [SE110-TPM case 2]

在此情况下，TPMV3 产生几乎 100%占空周期。相反，在 TPMV2 产生 0%占空周期。

—TPMxCnVH:L 从 0x0000 变为非零值[SE110-TPM case 3 and 5]。

在此模式下，TPMV3 等待开始一个新的 PWM 周期以开始产生新的占空周期设置的 PWM 波形。相反，TPMV2 在当前 PWM 周期的中央改变输出通道（当计数器值达到 0x0000）

—TPMxCnVH:L 从非零值变为 0x0000 [SE110-TPM case 4]

在此情况下，TPMV3 用以前的占空周期设置来完成当前 PWM 周期。相反，TPMV2 用新的占空周期设置来完成当前 PWM 周期。

6. 在 BDM 模式下写 TPMxMODH:L 寄存器（14.5.3 节，“TPM 计数器模数寄存器”（TPMMODH:TPMMODL）

在 TPMV3 在 BDM 模式下，写 TPMxSC 寄存器将清除 TPMxMODH:L 寄存器的写一致性机制。相反，TPMV2 此机制不受写操作影响。

第十五章 通用串行总线设备控制器（S08USBV1）

15.1 引言

这章主要描述一个基于通用串口总线规范2.0的通用串行总线设备控制器（S08USBV1）模块。设计USB总线的目的是取代用于PC机外围设备的总线接口，比如：RS-232，PS/2，以及IEEE1248总线。

USB模块支持全速（FS）USB信令，且集成了片上FS USB2.0兼容收发器。收发器有一个3.3V稳压器。USB模块提供了USB收发器和调节器的相关控制选择。

15.1.1 时钟要求

S08USBV1 需要两个时钟源，24MH总线时钟和48MHz总线时钟。48MHz时钟源是由MCGOUT直接产生。为了获得48MHz时钟速率，MCG必须被适当配置为PLL使能的外部占用（PEE）模块，外接一晶振。

对USB的操作，例如，配置MCG使用PEE模式包括：

2MHz晶振-RDIV=000和VDIV=0110

4NHz晶振-RDIV=001和VDIV=0110

15.1.2 USB悬挂模式的电流消耗

在USB的悬挂模式下，USB设备的电流消耗被限制在500uA以内。当USB设备进入悬挂模式，为了符合USB悬挂模式的电流消耗要求，固件通常进入STOP3模式。

注意：

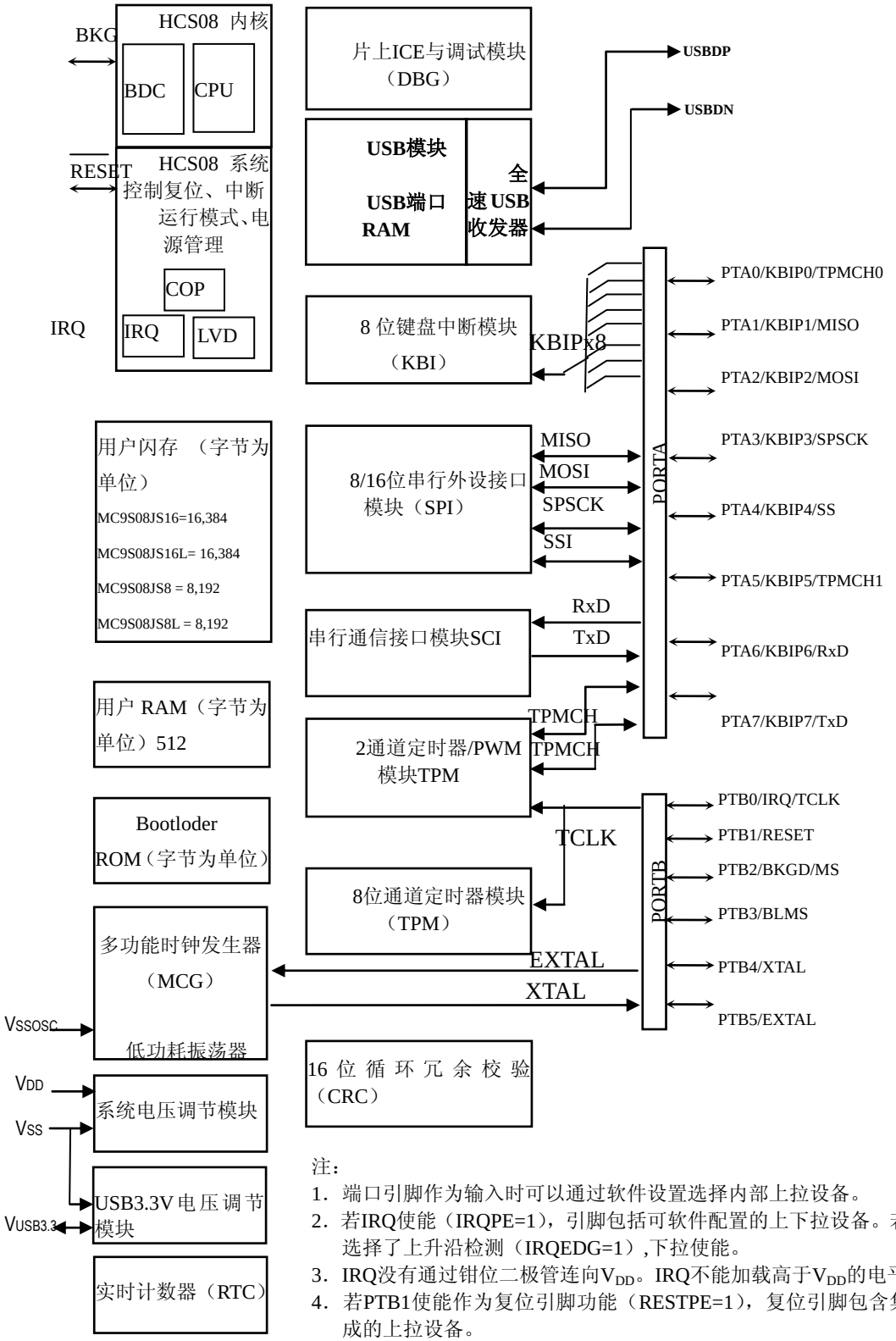
使能LVD将增加在STOP3模式下的电流消耗。所以，为了满足USB悬挂模式下的电流消耗需求，当在试图进入STOP3模式之前要禁止LVD。

15.1.3 3.3V 稳压器

如果使用外部3.3 V稳压器作为VUSB33输入（仅当USBVREN = 0），供电电压VDD，不得低于在VUSB33引脚的输入电压。如果使用内部3.3 V稳压器 （USBVREN = 1），确保不要连接外部VUSB33供电引脚。在这种情况下，为保证内部3.3 V稳压器的正常运行，VDD必须介于3.9 V和5.5 V之间。

表 15-1. USBVREN配置

USBVREN	3.3V 稳压器	VDD提供电压的范围
0	外围3.3V稳压器（作为Vusb333引脚输入）	Vusb333=<VDD
1	内部3.3V稳压器（无外部供电连接到Vusb333引脚）	3.9V=< VDD =<5.5V



注：

1. 端口引脚作为输入时可以通过软件设置选择内部上拉设备。
2. 若IRQ使能（IRQPE=1），引脚包括可软件配置的上下拉设备。若选择了上升沿检测（IRQEDG=1）,下拉使能。
3. IRQ没有通过钳位二极管连向V_{DD}。IRQ不能加载高于V_{DD}的电平
4. 若PTB1使能作为复位引脚功能（RESTPE=1），复位引脚包含集成的上拉设备。
5. 引脚包含集成上拉设备。
6. 当KBI使能（KBIPEn=1）而且相关引脚被配置为上拉设备使能，那么上升沿（KBEDGn=1）被用于将上拉设备配置为下拉设备。

图 15-1. MC9S08JS16系列框图USB模块和引脚高亮显示

15.1.4 特性

USB模式特性包括：
遵循USB2.0 协议
—12Mbps全速（FS）数据速率
—USB数据控制逻辑
—打包确认和解码/生成
—CRC生成和校验
—NRZI（反向不归零）加密/解密
—位填充
—同步监测
—结束包监测
7个USB端点
—双向端点0
—6个可配置的单向数据端点，可配置为中断，块，或同步
—5、6号端点支持双缓冲
USB RAM
—256字节的缓冲RAM，可在系统和USB间共享
USB 复位选项
—由MCU复位引起的USB模块复位
—主机产生的总线复位，触发CPU中断
支持远程唤醒的悬挂和恢复操作
收发器特性
—将不同的电压转换为数字逻辑
片上USB上拉电阻
片上3.3V稳压器

15.1.5 操作模式

表15-2. 操作模式

模式	描述
Stop1	USB 模块不可用。在进入 stop1 之前，内部 USB 的电压调节器和 USB 收发器进入关闭模式。因此， USB 的电压调节器和 USB 收发器必须由固件禁用
Stop2	USB 模块不可用。在进入 stop2，内部 USB 的电压调节器和 USB 收发器进入关闭模式。因此，电压调节器的 USB 和 USB 收发器必须由固件禁用
Stop3	在 stop3 模式下，USB 模块可用。每个符合 USB 2.0 版本规范的 USB 在悬挂模式下，可请求减少电流消耗，而 stop3 模式对获得较低的 MCU 电流消耗和整个 USB 设备电流消耗是很有用处的。在经固件进入 stop3 前，用户必须确保该设备被配置为 stop3，以实现的 USB 悬挂模式下电流消耗的目标。 当 SLEEPF 标志被置位时，USB 模块会通知进入悬挂模式，这种情况将在 USB 总线空闲 3 毫秒之后发生。该设备在 USB 悬挂模式下电流消耗水平要求由 USB 规范 Rev 2.0 所定义（启用远程唤醒功能的情况下，低功耗为 500μA，高功率为 2.5 毫安）。 如果 USBCTL0 寄存器中的 USBRESMEN 被置位并且在 USB 总线上发现 K-状态（恢复信令）时，USBCTL0 寄存器的 LPRESF 位将被置位。这会触发一个异步的中断，这个中断将控制器

	从 stop3 模式唤醒，并使能 USB 模块时钟。为清除 LPRESF 标志位，在 stop3 恢复后，USBRESMEN 位必须被立即清除。
等待（wait）	USB 模块可用。

15.1.6 框图

图15-2 为一个USB模块的框图。

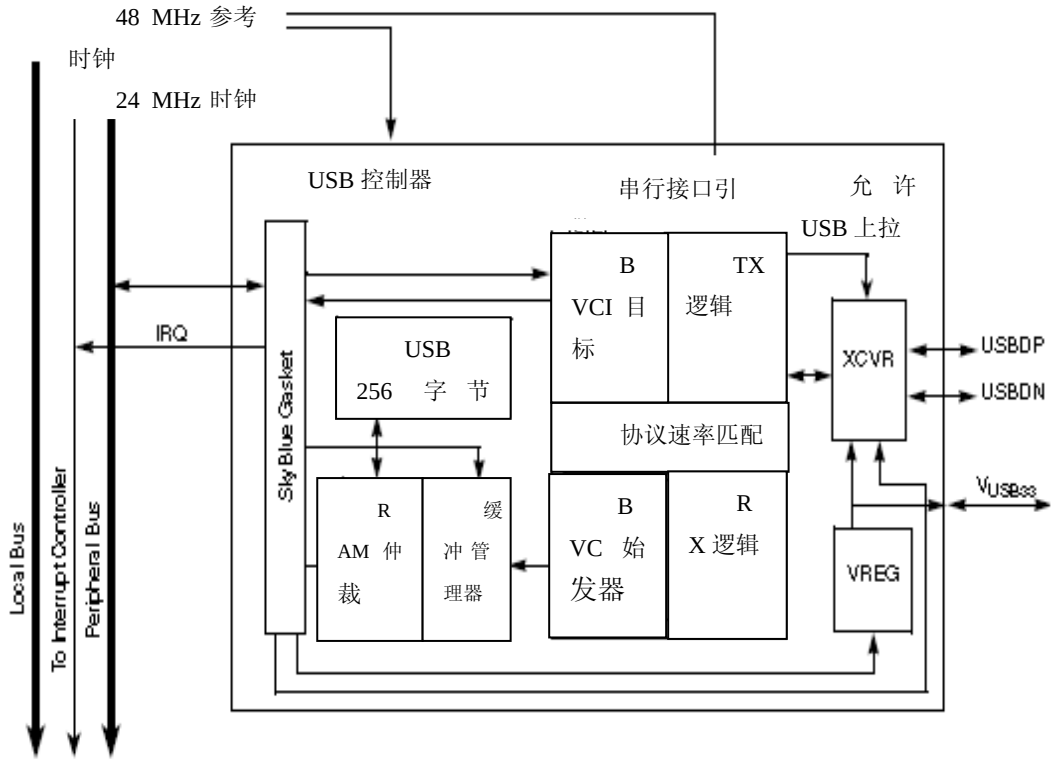


图15-2. USB模块框图

15.2 外部信号描述

USB模块要求数据和电源引脚。下表描述每个USB外部引脚

名称	端口	方向	功能	复位状态
正极差分信号	USBDP	I/O	USB差分信令	高阻态
负极差分信号	USBDN	I/O	USB差分信令	高阻态
USB稳压器电源引脚	Vusb333	电源	3.3Vusb稳压器输出或3.3V USB收发器/电阻供应输入	——

15.2.1 USBDP

USBDP是USB正差分信号引脚。在USB外设应用中，在此信号引脚上串联一个33Ω ±1%电阻以满足USB规范2.0版的实现要求。

15.2.2 USBDN

USBDN是USB负差分信号引脚。在USB外设应用中，在此信号引脚上串联一个33Ω ±1%电阻以满足USB规范2.0版的实现要求。

15.2.3 Vusb33

Vusb33连接到片上3.3V稳压器（VREG）。VUSB33保持3.3V电压输出，但仅为USB内部收发器（XCVR）和USB上拉电阻提供电流。如果VREG被软件禁用，应用程序必须输入一个外围3.3 V电源，通过USB模块的VUSB33引脚的给USB供电。

15.3 寄存器定义

这部分描述了内存映射和USB模块的控制/状态寄存器

15.3.1 USB控制寄存器0（USBCTL0）

	7	6	5	4		2		0
R	0	USBPU	USBRESMEN	LPRESF		USBVREN		USBPHYEN
W	USBRESET							
RESET	0	0	0	0		0		0

图15-3. USB传输和稳压器控制寄存器0 （USBCTL0）

表15-3. USBCTL0 字段描述

字段	描述
7 USBRESET	**USB 复位**—此位产生一个 USB 模块的硬件复位。USBPHYEN 和 USBVREGEN 位也会被清零。（需要重新开启 USB 收发器和 USB 稳压器）当被设置为 1 时，复位发生时此位自动被清零。 0 USB 模块正常运行 1 返回 USB 模块复位状态
6 USBPU	**上拉源控制位**—此位决定了 USBDP 线的上拉电阻的电流源。 0 内部 USBDP 上拉电阻被禁止；应用程序可以利用外部上拉电阻 1 内部 USBDP 上拉电阻使能。
5 USBRESMEN	**USB 低电压唤醒事件使能控制位**—当 USB 模块检测 LPRESF 位被置位，表明 USB 总线上为 K 状态，置此位，将使能 USB 模块以向 MCU 发送一个异步唤醒中断。在设置 SLEEPF = 1 之后 ，进入低功耗 stop3 模式之前该位必须被置位（USB 正在进入悬挂模式）。为了清除低功耗恢复标志位，在 stop3 恢复后必须立即清除此位。 0 禁止从悬挂模式异步唤醒 USB 1 使能在悬挂模式异步唤醒 USB
4 LPRESF	**低功电压恢复标志**—在 USB 悬挂模式下，如果 USBRESMEN=1 且在 USB 总线上检测到 K 状态，此时此位被置位，表明设备在 Stop3 模式下发出了恢复信令。为了清除

	LPRESF 位，固件必须先清除 USBRESMEN 位。 0 当设备在 Stop3 模式下，且 USB 处于悬挂状态，没有在 USB 总线上检测到 K 状态。 1 当设备在 Stop3 模式下，且 USB 处于悬挂状态，USBRESMEN=1，在 USB 总线上检测到 K 状态
2 USBVREN	**USB 稳压器使能**—此位使能片上 3.3VUSB 稳压器 0 片上稳压器被禁止（OFF 模式） 1 片上稳压器使能作为活动模式或待机模式。
0 USBPHYEN	**USB 收发器使能**—当 USB 收发器（XCVR）被禁止，USBDP 和 USBDN 都为高阻态。建议在设置 CTL 寄存器的 USBEN=1 之前，先使能 XCVR。当进入 USB SUSPEND 模式下时，固件必须允许保持 XCVR 使能 0 片上 XCVR 禁止 1 片上 XCVR 使能

15.3.2 外设ID寄存器（PERID）

寄存器PERID读取并返回值0X04。这个是为了USB外设模块而定义的。

	7	6	5	4	3	2	1	0
读	0	0	ID5	ID4	ID3	ID2	ID1	ID0
写								
复位	0	0	0	0	0	1	0	0

=未执行或保留

图15-4.外设ID寄存器（PERID）

表15-4 PERID字段描述

字段	描述
5:0 ID[5:0]	外设配置编号— 此编号被设置为0x04，表示外设处于全速模式

15.3.3 外设ID反码寄存器（IDCOMP）

IDCOMP 读取返回外设 ID 寄存器的反码。USB 模块的外设地址是 0XFB。

	7	6	5	4	3	2	1	0
读	1	1	NID5	NID4	NID3	NID2	NID1	NID0
写								
复位	1	1	1	1	1	1	1	1

=未执行或保留

图 15-5. 外设ID反码寄存器（IDCOMP）

表15-5. IDCOMP字段描述

字段	描述
5:0 NID[5:0]	ID号的反码—ID[5:0]的反码

15.3.4 外设版本寄存器（REV）

读取REV将返回USB外设版本号值。

	7	6	5	4	3	2	1	0
读	REV7	REV6	REV5	REV4	REV3	REV2	REV1	REV0
写								
复位	1	1	1	1	1	1	1	1
		=未执行或保留						

图15-6. 外设版本寄存器（REV）

表15-6. REV字段描述

字段	描述
8-0 REV[7:0]	版本—USB模块版本号

15.3.5 中断状态寄存器（INTSTAT）

寄存器INTSTAT包含了USB模块的中断源的每一位。这些位由各自的中断允许位控制（见中断允许寄存器）。寄存器的所有位一起进行逻辑或运算，形成一个独立的微控制器中断源。一旦中断位被置位，它只能写1到相应的中断位，中断位才被清除。这个寄存器在复位后的值为0x00。

	7		5	4	3	2	1	0
	STALLF		RESUMEF	SLEEPF	TOKENEF	SOFTOKF	ERRORF	USBRSTF
位	0		0	0	0	0	0	0

图15-8. 中断状态寄存器（INTSTAT）

表15-8. INTSTAT字段描述

字段	描述
7 STALLF	**STALL 标志**—STALL 中断被用在设备模式。在设备模式下，当串行接口引擎（SEI ）发送一个 STALL 握手信号时，STALL 标志被置位。 0 STALL 握手信号没有被发送 1 STALL 握手信号被发送
5 RESUMEF	**恢复标志**—在 USB 模块重启后 2.5us，伴随发送一个恢复信令，此时此位被置位。它可以用来表示 USB 总线上的远程唤醒信号。只有当 USB 模块进入悬挂模式（检测通常 SLEEPF 中断），此时此中断被使能。 0 没有检测到恢复信令 1 检测到恢复信令（在 USBDP/USBDN 持续 2.5 微秒检测到 K 状态信号）
4 SLEEPF	**睡眠标志—**如果 USB 模块检测到持续 3 毫秒空闲状态，此位被位置。表明 USB 模块将进入悬挂模式。在 USB 总线被激活时会引起睡眠定时器复位。 0 USB 总线没有检测到持续 3 毫秒空闲状态 1 一直在 USB 总线上检测恒定的 3 毫秒空闲状态
3 TOKDNEF	**令牌完成标志**—当前传输事务完成时此位被置位。固件必须立即读取 STAT 寄存器，以确定端点及 BD 信息。清除此位（通过设置它为 1）STAT 寄存器倍清除或 STAT 先进先出保

	持寄存器将被载入到 STAT 寄存器。 0 没有处理完成的令牌 1 当前令牌处理完成
2 SOFTOKF	**SOF 令牌标志**—若 USB 模块接收到一个开始帧（SOF）,此位被置位。 0 USB 模块没有收到 SOF 令牌 1 USB 模块收到 SOF 令牌
1 ERRORF	**错误标志—**ERRSTAT 寄存器的任何错误发生条件被满足，则此位被置位。固件必须读取 ERRSTA 寄存器以确定错误源。 0 ERRSTAT 寄存器中没有错误条件发生 1 检测到 ERRSTAT 寄存器中有错误条件发生
0 USBRSTF	**USB 复位标志**—当 USB 模块检测到一个有效的 USB 复位信号，此位被置位。若确定后，此位将通知 MCU，以自动向地址寄存器中写入 0X00 并且使能端点 0。若连续检 2.5 微秒测到 USB 复位信号，USBRSTF 将被置位。在 USB 复位条件被移出之前，不会再次置位，若被移出，则重置。 0 没检测到复位信号 1 检测到复位信号

15.3.6 中断使能寄存器（INTENB）

寄存器INTENB包含USB模块每个中断源允许位。设置其中任何位将使能各自中断源INTSTAT寄存器。复位后该寄存器值为0x00，也就是禁用所有中断。

	7	6	5	4	3	2	1	0
读 写	STALL	0	RESUME	SLEEP	TOKENE	SOFTOK	ERROR	USBRST
复位	0	0	0	0	0	0	0	0

图15-9.中断使能寄存器（INTENB）

表15-9. INTENB字段描述

字段	描述
7 STALL	**STALL中断使能**—设置此位将使能STALL中断 0 禁止中断 1 使能中断
5 恢复	**恢复中断使能**—设置此位将使能恢复中断 0 禁止中断 1 使能中断
4 睡眠	**睡眠中断使能**—设置此位将使能睡眠中断 0 禁止中断 1 使能中断
3 TOKDNE	**TOKDNE 中断使能**—设置此位使能令牌中断 0 禁止中断 1使能中断
2 SOF令牌	**SOF令牌中断使能**—设置此位将使能SOF令牌中断 0 禁止中断 1 使能中断

1 错误	**错误中断使能**—设置此位将使能错误中断 0 禁止中断 1 使能中断
0 USBRST	**USB复位中断**—设置此位将使能USB复位中断 0 禁止中断 1 使能中断

15.3.7 错误中断状态寄存器（ERRSTAT）

寄存器ERRSTAT的每一位包含了USB模块中所有的错误源。这些位中的每一位与各自的错误使能位相一致（见15.3.8 中“错误中断使能寄存器（ERRENB ）”）。这些结果被或在一起，然后发送到INTSTAT寄存器中的错误位。 一旦中断位被置位，只能通过写1到相应标志为来清除此位。在错误条件被检测到时就立即置相应位。因此，正常情况下中断将与最后处理的令牌不一致。在复位后寄存器为0X00。

	7	6	5	4	3	2	1	0
读 写	BTSERRF	RESENED	BUFFERR	BTOERRF	DFN8F	CRC16F	CRC5F	PIDERRF
复位	0	0	0	0	0	0	0	0

图15-10.错误状态寄存器（ERRSTAT）

表 15-10. 错误中断状态寄存器（ERRSTAT）

字段	描述
7 BTSERRF	**位填充错误标志**—检测到一个位填充错误将被置位。若置位，对应的包将因为一个位填充错误而被丢弃。 0 没有检测到位填充错误 1 检测到位填充错误
5 BUFERRF	**缓冲错误标志**—如果USB模块已要求读取内存中新的BD，但在USB模块需要接收或传输数据之前没有得到总线控制权，此位将被置位。如果处理一个TX（输入端点）传输，这将导致传输数据下溢条件。或者，如果处理的接收Rx（OUT端点）传输，这将导致接收数据溢出条件。如果主机发送或接收的数据包大于在BD分配的缓冲区大小。在这种情况下，在放入缓冲存储器之前数据包将被截断。 0 没有检测到缓冲区错误 1 缓冲区发生错误
4 BTOERRRF	**总线翻转超时错误标志**—如果总线翻转错误发生那么此位将被置位。USB模块使用总线翻转定时器来保持跟踪IN令牌的SETUP，TOKEN或数据以及握手相位的令牌和数据脉冲。如果多于16位，时间就会从前一个EOP开始计数而不是从IDLE开始计数，总线的一次翻转超时错误将会发生。 0 没有检测到总线翻转超时 1 检测到总线翻转超时
3 DFN8	**DFN8中断使能**—设置此位将使能DFN8（数据字段错误）中断 0 禁止中断 1 使能中断
2	**CRC16中断使能**—设置此位将使能CRC16中断

CRC16	0 禁止中断 1 使能中断
1 CRC5	**CRC5中断使能**—设置此位将使能CRC5中断 0 禁止中断 1 使能中断
0 PIDERR	**PIDERR中断使能**—设置此位将使能PIDERR中断 0 禁止中断 1 使能中断

15.3.8 错误中断使能寄存器（ERRSTAT）

	7	6	5	4	3	2	1	0
读 写	BTSERRF	0	BUFFER	BTOERRF	DFN8F	CRC16F	CRC5F	PIDERRF
复位	0	0	0	0	0	0	0	0

图15-11. 错误中断使能寄存器

表15-11.ERRSTA字段描述

字段	描述
7 BTSERR	**BTSERR 中断使能**—设置此位将使能 BTSERR 中断 0 中断禁止 1 使能中断
5 BUFFER	**BUFFER 中断使能**—设置此位将使能 BUFFER 中断 0 中断禁止 1 中断使能
4 BTOERR	**BTOERR 中断使能**—设置此位将使能 BTOERR 中断 0 中断禁止 1 中断使能
3 DNF8	**DNF8 中断使能**—设置此位将使能 DNF8 中断 0 中断禁止 1 中断使能
2 CRC16	**CRC16 中断使能**—设置此位将使能 CRC16 中断 0 中断禁止 1 中断使能
1 CRC5	**CRC5 中断使能**—设置此位将使能 CRC5 中断 0 中断禁止 1 中断使能
0 PIDERR	**PIDERR 中断使能**—设置此位将使能 PIDERR 中断 0 中断禁止 1 中断使能

15.3.9 状态寄存器 （STAT）

寄存器STAT 能够显示USB模块内的传输状况。当MCU收到TOKDNE中断请求，MCU读取STAT寄存器，以确定先前端点的通信状态。只有在TOKDNEF中断标志被确定时，状态寄存器的数

据才有效。STAT实际上是在窗口中读取由USB模块维护的状态FIFO。当USB模块使用BD时，它将更新状态寄存器。如果其他的USB事务在TOKDNE中断服务之前完成，USB模块将存储下一个事务的状态到STAT FIFO中。因此，STAT寄存器实际上是一个4字节的FIFO，允许微控制器处理一个事务，而串行接口引擎（SIE）的处理下一个STAT值。清除INTSTAT寄存器的TOKDNEF位，将导致SIE用下一个STAT的值更新STAT寄存器。如STAT FIFO中的下个数据有效的，SIE士将立即重新确定TOKDNE中断。

	第 7 位	6	5	4	3	2	1	第 0 位
读	END[3:0]				IN	ODD	0	0
写								
复位	0	0	0	0	0	0	0	0

图 15-12.状态寄存器（STAT）

表 15-12. STAT字段描述

字段	描述
7-4 ENDP[3:0]	**端点号**—这四位用于将先前接收到或发送令牌的端点地址进行编码。这将允许微控制器决定哪个BDT入口被最后一次USB事务更新 0000端点0 0001端点1 0010端点2 0011端点3 0100端点4 0101端点5 0110端点6
3 IN	**In/Out事务**—此位表明究竟被更新的BDT是发送（IN）传输还是接收（OUT）传输。 0 上一事务是接收（OUT）数据传输 1 前一BDT为发送传输（IN）
2 ODD	**奇/偶事务**—此位表明究竟最新的缓冲描述符寄存更新是在BDT的奇数存储区还是偶数个存储区。 0 最新被更新的缓冲区表述符为偶数存储区 1 最新被更新的缓冲区表述符为奇数存储区

15.3.10 控制寄存器（CTL）

寄存器CTL为USB模块提供了各种控制和配置信息。

	7	6	5	4	3	2	1	0
读 写			TSUSPEND			CRESUME	ODDRST	USBEN
复位	0	0	0	0	0	0	0	0

图15-13.控制寄存器（CTL）

表 15-13. CTL字段描述

字段	描述

5 TSUSPEND	**事务悬挂**— 当收到 setup 令牌时，此位被串行接口引擎（SIE）置位。并在恢复令牌处理之前，允许软件解队 BDT 中任何等待的事务包。TSUSPEND 位通知处理器，SIE 已禁用包传输和接收。清除此位将允许 SIE 继续处理令牌。 0 允许 SIE 继续处理令牌 1 收到 setup 令牌时; SIE 置此位，SIE 已禁用数据包传输和接收。
2 CRESUME	**恢复信令**— 设置此位将允许 USB 模块执行恢复信令。这将允许 USB 模块的远程唤醒功能。软件必需置 CRESUME 为 1，持续 USB 规范 2.0 所要求的时间，然后再清除此位。 0 不执行远程唤醒功能 1 执行远程唤醒信令
1 ODDRST	**奇复位**-设置此位将重置所有的缓冲描述符的奇 ping-pong 位为 0，届时将指定偶描述符存储区。此位是使用双缓冲端点 5 和 6。该位对端点 0 到 4 没有影响。 0 不复位 1 重置所有缓冲描述符器奇 ping/pong 位为 0，然后指定偶描述符存储区
0 USBEN	**USB 使能**— 设置的此位将使 USB 模块操作。设置此位，将导致 SIE 重置 ODD 位到 BDTs。因此，设置此位将重置在 SIE 中的许多逻辑。 0 禁用 USB 模块 1 启用 USB 模块的操作，不会影响收发器和 VREG。

15.3.11 地址寄存器（ADDR）

地址寄存器ADDR包含了唯一的7位地址，这个地址将像USB一样被设备识别。在复位输入处于活动状态或USB复位信令被解码之后，寄存器复位为0x00。寄存器如USB规范要求的那样初始化，以解析地址0x00。当固件处理SET_ADDRESS请求时，将改变其值。

	7	6	5	4	3	2	1	0
读		ADDR6	ADDR5	ADDR4	ADDR3	ADDR2	ADDR1	ADDR0
写								
复位	0	0	0	0	0	0	0	0

图 15-14.地址寄存器（ADDR）

表15-14. ADDR字段描述

字段	描述
6:0 ADDR[6:0]	**USB 地址**—这7位定义了USB模块解析的USB设备地址

15.3.12 帧号寄存器（FRMNUML,FRMNUMH）

帧号寄存器包含11位帧号。帧号寄存器要求两个8位寄存器来实现。低位字节包含于FRMNUML中，高阶字节包含于FRMNUMH中。每当接收到SOF令牌时，将更新这些寄存器为当前帧号。

	7	6	5	4	3	2	1	0
读	FRM7	FRM6	FRM5	FRM4	FRM3	FRM2	FRM1	FRM0
写								
复	0	0	0	0	0	0	0	0

位								

图 15-15. 帧号寄存器低字节（FRMNUML）

表 15-15. FRMNUML 字段描述

字段	描述
7-0 FRM[7-0]	帧号—这些位出现在11位帧号的低八位

	7	6	5	4	3	2	1	0
读	0	0	0	0	0	FRM10	FRM9	FRM8
写								
复位	0	0	0	0	0	0	0	0

图 15-16. 帧号寄存器高字节（FRMNUMH）

表 15-16. FRMNUMH字段描述

字段	描述
2-0 FRM[10:8]	帧号—这些位出现在11位帧号的高八位

15.3.13 端点控制寄存器（EPCTLn, n=0-6）

端点控制寄存器包含每个可用端点的端点控制位(EPCTLDIS,EPRXEN,EPTXEN和EPHSHK)这些位组成USB模块解码地址。这四位定义了任何端点控制所需的信息。寄存器的格式见下表。端点0（ENDP0）与控制管道0，这是由USB的所有功能所要求的。因此，在USBRST中断收到之后，微控制器必须设置EPCTL0 为0x0D。

	7	6	5	4	3	2	1	0
读	0	0	0	EPCTLDIS	EPRXEN	EPTXEN	EPSTALL	EPHSHIK
写								
复位	0	0	0	0	0	0	0	0

图 15-17.端点寄存器（EPCTLn,n=0-6）

表 15-17. EPCTLn字段描述

字段	描述
4 EPCTLDIS	**端点控制**—此位定义了端点是否被使能和端点方向。端点使能和端点控制定义在表15-19中
3 EPRXEN	**端点RX使能**—此位定义了OUT传输端点是否被使能。端点使能/方向控制定义在15-19中。
2 EPTXEN	**端点Tx使能**—此位定义了IN传输端点是否被使能。端点使能/方向控制定义在表15-19中。
1 EPTALL	**端点拖延**—当被置位，则此位表明端点被拖延。在端点控制寄存器中此端点比所有其它位有更高的优先级，但仅当EPTXEN = 1或EPRXEN = 1时有效。任何对此端点的访问，将导致USB模块返回一个拖延握手信号。一旦端点被拖延,需要来自主控制器的干预。 0 端点n没有被拖延 1 端点n被被拖延

表 15-17. 端点使能/方向控制

位名			端点使能/方向控制
4 EPCTLDIS	3 EPRXEN	2 EPTXEN	
X	0	0	禁止端点
X	0	1	使能IN（TX）传输端点
X	1	0	使能OUT（RX）传输端点
0	1	1	使能IN,OUT端点以及SETUP传输
0	1	1	保留（RESERVED）

15.4 功能描述

本节描述了USB模块的功能。它记录了端点0和数据端点的数据包处理，USB的挂起和恢复状态，SOF令牌处理，复位条件和中断。

15.4.1 模块描述

图15-2为其框图。以下各节主要描述该模块的子块和外部信号。该模块涉及几个主要模块- USB收发器（XCRV），USB串行接口引擎（SIE），3.3 V稳压器（VREG），端点缓冲区管理，共享RAM仲裁，USB RAM和SkyBlue gasket。

15.4.1.1 USB串行接口引擎（SIE）

SIE有两个主要功能：TX逻辑和RX逻辑。下面将给出这些主要功能更多的细节。 Tx和Rx逻辑通过USB协议引擎连接在一起。USB协议引擎连接管理USB模块包流入流出。SIE通过内部基本虚拟组件接口（BVCI）和创建者的总线与系统挂接在一起。BVCI接口用于配置USB SIE并向CPU提供状态和中断信息。BVCI创建者接口提供了完整的DMA控制访问缓冲区描述表BDT的通路，并输入或输出USB数据到USB RAM。

15.4.1.1.1 串行接口引擎SIE发送器逻辑

SIE发送器逻辑有两个主要功能。首先是格式化被储存在端点缓冲区的USB数据包。二是要通过USB收发器收发数据包。

所有必需的USB数据格式化操作都是由SIE发送器逻辑执行，包括：

—不归零倒相方式解码

—位填充

—CRC计算

—增加SYNC字段

—增加包结束信息（EOP）

CPU通常将数据放置在端点缓冲区作为应用程序的一部分。当缓冲区作为IN类型缓冲区且USB主机请求发送一个数据包，那么SIE用一个被正确格式化的数据包作为响应。

发送器逻辑也被用与从USB主机接收的数据包后产生应答。当从USB主机接收格式正确的数据包后，发送器逻辑回应适当的应答包，非应答包或拖延握手信号作为应答。

当SIE发送逻辑正为一个特殊端点从缓冲区发送数据时，不推荐CPU操作相应端点缓冲区。

15.4.1.1.2 串行接口引擎（SIE）接收器逻辑

SIE接收器逻辑用于接收USB数据并存储在USB RAM中以便于CPU和应用软件处理。收发器的串行数据转换为字节宽度的并行数据流，进行正确检查，并存储在USB RAM中。

接收位流过程包含一下操作：
—解码NRZI USB 串行数据流
—Svnc检测
—位填充移除（和错误检测）
—包结束信息（EOP）检测
—CRC有效性
—PID检测
—其他USB协议层检测
SIE接收器逻辑提供错误检测包括：
—错误的CRC
—EOP超时检测
—位填充违规
如果接收到正确格式的数据包时，接收器逻辑发起一个到主机握手应答。如果由于位填充违规，CRC错误或其它包层问题，包没有被正确的解码，接收器将忽略此包。 USB主机将响应一个最终超时等待并重传该包。

15.4.1.2 MCU/存储器接口

15.4.1.2.1 Sky Blue gasket

Sky Blue gasket用于链接USB模块和SOC内部外设总线。

15.4.1.2.2 端点缓冲区管理器

每个端点是由USB设备传输的来往于缓冲区的数据支持的，这些数据保存在共享缓冲存储器。串行接口引擎（SIE）使用描述符表，缓冲描述符表（BDT），描述每个端点的特性，这些表也保存在USB的RAM中。端点缓冲区管理器负责把端点缓冲描述符映射到USB RAM块物理地址中。

15.4.1.2.3 RAM 仲裁

15.4.1.3 USB RAM

USB模块包含256字节的高速RAM，可以被串行接口引擎（SIE）和CPU存取。USB RAM以两倍总线时钟的速率运行，这样可以允许CPU和SIE交错非阻塞存取。USB RAM用于保存缓冲区描述符表（BDT）和端点缓冲区。如果USB模块被禁止，这时整个USB RAM可能是作为不安全的系统存储器使用。

15.4.1.4 USB收发器（XCVR）

USB收发遵循通用串行总线规范2.0。这块为USB通信提供了2-线差分NRZI信令。收发器是片内提供的用于价格敏感的单芯片USB外设解决方案。

15.4.1.5 USB片内电压调节器（VREG）

片内3.3V调节器为USB内部收发器提供稳定的电源并为内部或外部终端提供上拉电阻。当片内收发器被允许时，它要求3.9V到5.5V电源输入，而电压调节器的输出范围在3.3V到3.6V之间。

带有一个专用的片内USB3.3V调节器和一个独立的MCU电源输入，MCU和USB可以在不同的电压下工作(有关USB电器特性请参考USB电压调节器电器特性)当片内3.3V调节器被禁用时，3.3V电源必须通过Vusb3.3引脚传递给USB收发器。这种情况下，MCU的电源电压不能低于Vusb33引脚的输入电压。

3.3V调节器包含三种模式：
激活模式—当USB激活时进入此模式。目前要求满足电源供应收发器和USBDP上拉电阻。

备用模式—电压调节器备用模式是在当USB设备处在挂起模式时自动进入的。当USB设备被USB总线强迫进入挂起模式时，固件必须配置MCU进入Stop3模式。在备用模式，要求通过一个900Ω （最坏打算）上拉，使USBDP引脚上的电压保持在3.0V到3.6V之间。

电源关闭模式—当处于STOP2或Stop1模式，或电压调节被禁用时，将进入此模式。

15.4.1.6 USB片内USBDP上拉电阻

USBDP线上的上拉电阻是全速操作要求的，是USB2.0规范中规定的，它可以根据具体应用的要求选择使用MCU内部或者外部的。一个片内上拉电阻是在USB2.0电阻ECN规范中指定的设备，它是随时由固件配置为可用的。或者将片内上拉电阻禁用，USB可以为USBDP线使用外部上拉电阻作为替代。如果在USBDP线上使用一个外部电阻，电阻必须遵循USB2.0电阻ECN的要求，在http://www.usb.org上可以找到。

寄存器USBCTL0的USBPU位用于表明MCU使用内部还是外部上拉电阻。如果USBPU被清楚，USBDP上的内部上拉电阻被禁用，这时可以使用外部USBDP上拉电阻。当使用一个外部USBDP上拉电阻，如果电压调节器 被使能，Vusb33引脚使出电压可以被USBDP上拉电阻使用。一般情况推荐使用内部USBDP上拉电阻，下图为USB设备使用连接到Vusb33引脚的外部USBDP上拉电阻的配置图。

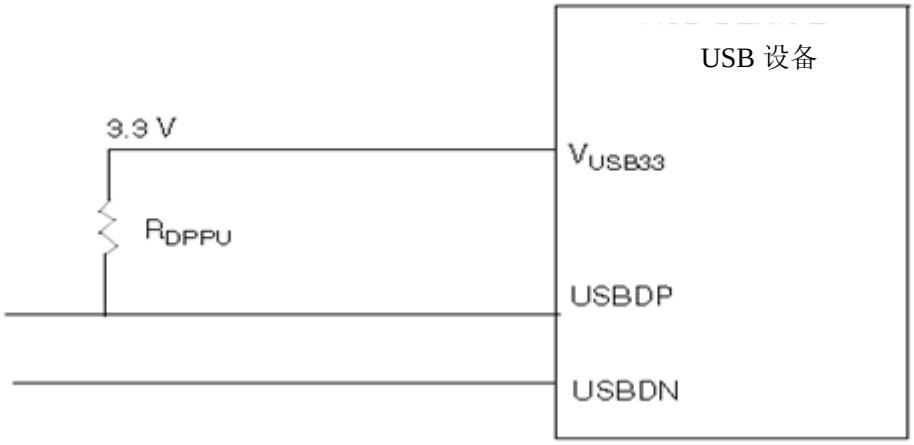


图 15-18. USB模块USBDP/USBDN 上拉电阻配置

15.4.1.7 USB电源和USBDP上拉电阻使能选择

USB模块为USB设备应用提供了单芯片解决方案，可以使用自供电或者总线供电。为了使能或者禁用USBDP线上的上拉电阻，USB设备需要知道什么时候它有一个合法USB连接。对于设备上的USB模块，USBDP的上拉电阻仅在一个合法的VBUS连接被识别的时候使用，这是USB规范要求的。

在总线供电的应用中，系统电源必需来自VBUS。在一个合法的来组主机到设备USB连接被建立，VBUS检测被建立，而且USBDP上拉电阻也相应的被允许时VBUS才是可用的。

在自供电应用时，判定合适一个合法连接被建立与总线供电应用是不同的。在自供电应用中，VBUS检测必须在应用中建立。例如，一个KBI引脚中断能够被初始化（如果可用）。当一个VBUS连接被建立，KBI中断能够通知应用程序，一个合法的VBUS连接可用而且内部上拉电阻可以通过USBPU位使能。如果一个外部上拉电阻替代内部上拉电阻被使用，VBUS检测机制必须被包含在系统设计中。

表15-19. 对不同USB供电模式USBDP上拉允许

电源	USBDP上拉	上拉允许

总线供电（VBUS检测中建立）	内部	设置USBPU位
	外部	在应用中建立
自供电（VBUS检测在应用中建立）	内部	设置USBPU位
	外部	在应用中建立

15.4.2 缓冲区描述表（BDT）

为了提高管理USB端点通信，USB模块配备一个缓冲区描述表（BDT），该表是由存储在USB RAM中的缓冲区描述符（BD）组成。BD实体为相应端点提供状态或控制信息。BD实体也提供端点缓冲区的地址。下一节将详细描述BDT格式。

软件API通过更新BDT管理缓冲区。这使得USB模块可以高效的处理数据发送和接受。

因为缓冲区是由微控制器和USB模块共享的，使用简单的信号量机制来区分哪一个被允许更新共享缓冲中的BDT和缓冲区。信号量位，即OWH，在BD实体被微控制器占有时会将此位清零。当OWN位为0时允许微控制器读写BD实体和数据缓冲区。当OWN位为1时，BD实体和数据缓冲区被USB模块占有。这时USB模块可以进行完全的读写存取，而微控制器禁止修改BD或者它的相应数据缓冲区。

15.4.2.1 单一端点的多缓冲描述表

每个端点需要至少一个三个字节的缓冲描述实体。因此，端点0，一个双向控制端点需要一个IN 类型BDT实体和外向BDT实体。

使用两个BD实体也允许双缓冲。双缓冲区BDs使USB模块很容易的在USB模块提供的最大吞吐量速度下传输数据。双缓冲区允许MCU处理一个BD的同时,USB模块处理另一个BD。

为了实现双缓冲区，每一个端点方向都需要两个缓冲描述符（BD）实体。一个是偶BD实体，另一个是奇BD实体。

15.4.2.2 缓冲描述符表实体编址

BDT编址是固定连接到USB模块的。BDT占据了USB RAM的第一部分。为了通过USB或MCU访问端点数据，需要理解缓冲区描述符表的编址机制。

所有的IN和OUT端点实体都被索引到BDT中，以此允许经由MCU或USB模块的存取简单化。下图说明了USB RAM的组织。此图说明了USB RAM的第一个实体是专门用于存储BDT实体的。例如，USB RAM开始的30个字节（0X00到0X1D）被用于实现BDT。

USB RAM 偏移		USB RAM内容描述
0X00	BDT	端点 0 IN
		端点 0 OUT
		端点 1
		端点 2
		端点 3
		端点 4
		端点 5，偶缓冲
		端点 5，奇缓冲
0X1D		端点 6，偶缓冲
		端点 6，奇缓冲
0X1E		保留
0X1F		保留
0X20		端点缓冲可用USB RAM

0XFF	

当USB模块在一个使能端点接收一个USB令牌，它将询问BDT。USB模块读相应的端点BD实体，判断它是否拥有BD和相应的数据缓冲区。

15.4.2.3 缓冲描述符格式

缓冲描述符格式（BDs）是一组寄存器，它们为USB模块和MCU提供端点缓冲区控制信息。谁在读取存储器中的BD将决定BD的不同含义。

USB模块是用存储在BD中的数据决定：

-谁在系统存储器中占缓冲区；

-数据0或数据1 PID；

-接收的令牌PID；

-多少数据将被传送或接收；

-缓冲区分配在缓冲RAM的哪一部分；

BDT由缓冲描述符（BD）组成，BD用于定义和控制USB RAM空间中的实际缓冲区。BDs一直作为3字节块产生。图 15-19 是一个BD实例，端点0 IN从USB RAM偏移量0X00开始。

缓冲描述符的格式在表15-22中说明。

Offset			6	5	4	3	2	1	0
0	R	OWN	DATA0/1	BDTKPID[3]	BDTKPID[3]	BDTKPID[1]	BDTKPID[0]	0	0
x00	W			0	0	DTS	BDTSTALL		
0	R	BC[7:0]							
x01	W								
0	R	EPADR[9:4]							
x02	W								

图15-19. 缓冲描述符实例

表15-20. 缓冲描述符表字段

字段	描述
OWN	**OWN**—OWN位决定了当前谁（MCU或USB模块）占据缓冲区。USB的SIE通常当它完成令牌发送后向这一位写0.当OWN=0时，USB模块忽略所有BD中的其它字段。一旦BD已经标记为USB模块（OWN=1），MCU不能再对它有任何修改。虽然硬件不会在USB的SIE占据缓冲区时阻止MCU读写BD，但是这样将引起为之操作而且一般情况下也不推荐这样做。 0 MCU具有对BD的唯一访问权 1 USB模块具有对BD的唯一访问权
DATA0/1	**数据toggle**—此位定义了是否DATA0（DATA0/1=0）字段或者DATA1（DATA0/1=1）字段被传输或接收。它不会被USB模块修改。 0 Data0包 1 Data1包
BDTKP[3:0]	一次传输完成后，当前令牌PID被USB模块写回到BD。写回的值来自USB指定的令牌PID：0x1 表示一个OUT令牌，0x9表示IN令牌，0xd表示初始令牌
DTS	**数据toggle同步**—此位使能数据toggle同步。 0 无数据toggle同步 1 数据toggle同步被执行
BDTSTALL	**BDTSTALL**—如果 SIE 收到一个令牌，设置此位将引起 USB 模块产生一个延时握手。当 BDTSTALL 位被置位时，BDT 不会被 SIE 更新（OWN 位保留而 BD 中剩下的不变）。

	0 BDT 延时被禁止 1 果令牌被 SIE 收到，USB 将会引起一次延时握手信号，这将会在这个位置使用 BDT
BC[7:0]	**字节计数**—字节计数代表一个8位字节计数。USB模块串行接口引擎（SIE）根据接收到的数据长度更新字段。但是注意，对于同步端点，当USB数据包达到1023字节时，USB模块限制包大小为64字节。
EPADR[9:4]	**端点地址**—端点地址表示本地 USB RAM 内 10 位缓冲地址的高六位。EPADR 的比特[3:0]总线是 0，因此缓冲地址必需从本地 RAM 的 16 字节对齐地址开始。USB 模块不改变这些位，这些不是系统总线存储器中的地址。EPADR 是相对于本地 USB RAM 开始的。

15.4.3 USB 传输

当USB模块传送或接收数据时，它会首先根据端点号，数据方向，偶或奇缓冲计算BDT地址，然后读取BD。

一旦BD已经被读取如果OWN=1，串行接口引擎（SIE）将会传送包数据到缓冲区或接收来自缓冲区的包数据，地址为BD中的EPADR字段指向的地址。当USB令牌处理结束，USB模块会更新BDT并且置OWN位为0。

如果TOKDNE中断置位，那么STAT寄存器被更新。当微控制器处理TOKDNE中断时，它会读状态寄存器。这将会把要处理的端点的所有信息传递给CPU。此时微控制器可以分配一个新的BD，其他的USB数据可以被发送或接收，而且它可以处理先前的BD。图15-20说明了一个典型USB令牌被处理的时序。

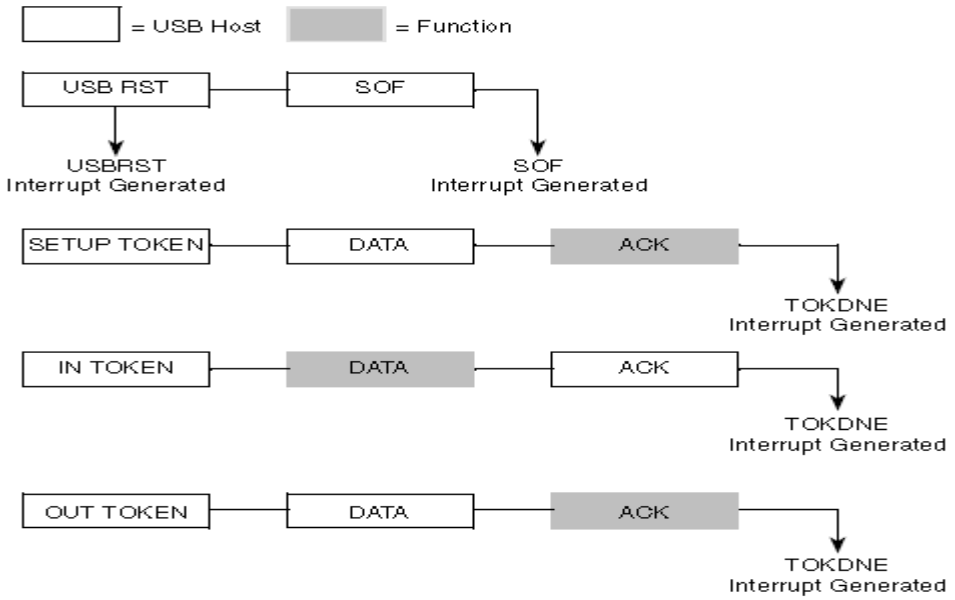


图15-20. USB数据包流程

USB有两种数据过载错误：

USB RAM接口响应时间太长导致接收缓冲溢出。

接收到的包可能比规定的MAXPACKET容量大。这是由软件bug所引起的。

在第一钟情况下，USB会回复一个NAK或者总线超时（BTO）作为传输级的应答。寄存器ERRSTAT中的BTOERR位将会被置位1.根据INTENB和ERRENB寄存器的值,USB模块可能会产生一个中断来把这个错误通知CPU。在设备模式BDT不被写回也不被TOLDNE中断触发，因为它被假设可能在第二次尝试后成功。

在第二种过大数据包的情况下，USB规范假设两边都是正确的软件驱动程序。过速不是由于存储器的响应时间而是缺少存储多余数据的空间。不应答的包可能会导致已经过载的数据包重传。为了响应过载，对于不同步传输USB模块将会一直应答这个包。写向存储器的数

据被分拆为MAXPACKET（最大包）大小，以免破坏缓冲区空间。USB 模块将会声明ERRSTAT寄存器中的BUFERRF位（它可以触发上述中断）和TOKDNE中断失败。BDT中的BDTKPID字段将不会是“1111”因为BUFFERRF不是由于延时造成的错误。写回BDT的包长度字段将会是MSXPACKET的值，用来代表被分拆后写入寄存器数据的实际长度值。从这以后软件可以为将来的传输（延时端点， 取消传输，禁止端点等等）决定合适的操作。

为USB设备处理数据包包括管理IN（到USB主机）和OUT传输（到USB设备）缓冲。USB是主从的结构，数据方向以主机为主导。包处理进一步分为在端点0的请求处理，在数据端点的数据处理。

15.4.4.1 USB数据管道处理

数据管道处理本质上是一个缓冲管理任务。固件负责管理共享缓冲RAM，以确保BD总是准备好硬件处理（OWN=1）.

设备在共享RAM中分配缓冲区，建立缓冲描述符，等待中断。在收到一个TOKDNE中断以后，固件读取寄存器STAT，决定哪一个端点被影响，然后读相应的BDT实体决定下一步该做什么。

当处理数据包时，固件负责按USB规范和模块物理限制管理包缓冲大小。所有端点支持64字节的包。同步端点也能仅仅指定在64字节的包容量。

固件同样负责为BDT设置适当的位。对于多数应用来说使用大容量（bulk）包（控制，大容量，中断类型传输），使用固件会为每个BD设置DTS,BC和EPADR字段。对于等时包，固件会设置BC和EPADR字段。在所有情况下，固件都会设置OWN位，以允许端点传输数据。

15.4.4.2 在端点0的请求处理

在多数情况下，给USB设备的命令直接传给端点0.主机使用“标准请求”（USB规范第九章描述）列举和配置设备。在主机上运行的类驱动或产品指定的驱动程序发送类（HID，大容量存储器，映像）和销售商指定的命令给设备的端点0。

USB请求总是遵循一个指定的格式：

主机发送一个SETUP令牌，跟随一个8字节SETUP包，而设备硬件能够发送一个握手包。

如果SETUP指定一个数据阶段，主机和设备可能传送超过64Kbytes的数据（要么是IN，要么是OUT，不是全部）。

请求由状态阶段终止

设备固件监控INTSTAT和STAT寄存器，端点0缓冲描述符（BD‘s），和执行当前主机请求的初始令牌内容。

处理端点0请求的流程如下所示：

1. 为端点0 OUT分配8字节缓冲区
2. 为端点0 OUT建立BDT实体，并且设置DT和OWN位为1
3. 等待TOKDNE中断
4. 读STAT寄存器

   —状态寄存器必须显示端点0，RX。如果没有这，判断端点控制器中的EPSTALL位。
5. 读端点0的OUT BD

   —核实令牌是否是SETUP令牌。如果不是，声明端点控制寄存器中的EPSTALL位。
6. 解码并处理建立包

   —如果建立包中的方向字段表示包是OUT传输，那么将进入数据阶段，接受在建立阶段所只是的数据长度。

   —若果初始包中的方向字段表示此包是IN传输，那么进入数据阶段，发送不多于规定长度的数据。

7. 数据阶段之后是状态阶段

—如果数据阶段位OUT事务，那么会设置BC为0，OWN=1.如果数据阶段为IN事务，主机会发送给从机一个确认信号。

15.4.4.3 端点0异常条件

USB模块包含了一定数量的错误检查和恢复机制，以此保证数据传输的可靠性。当主机发送一个SETUP包给设备时一个异常产生了，主机将不会接受来自设备的应答握手。这种情况下，主机将重新发送SETUP包。

端点0请求设备上的操作者（handers）必须考虑到在接受一个正确的SETUP包后，它们能在数据阶段真正开始前收到另一个SETUP包。

15.4.5 起始帧处理

USB主机把时间分成1.0ms的叫作“帧”的时间片，目的是为了包的调度。USB主机使用叫作SOF（起始帧）的广播令牌开始每一帧，它包括11位的序列号码。

TOKSOF中断是用于通知固件什么时候SOF令牌被收到。

固件可以读来自FRMNUL/FRMNUMH寄存器的当前帧编码。

一般来说，SOF中断仅被设备禁止，使用等时端点帮助确保设备和主机 保持同步。

15.4.6 挂起/恢复

USB支持一个单独的低电压模式叫作挂起。进入和离开起始状态在下面描述。

15.4.6.1 挂起

USB主机可以在任何时候把段度的设备或整个总线上的设备变为挂起状态。为了降低功耗MCU支持挂起模式。当USB数据处于空闲状态超过3ms时将进入挂起模式。进入挂起状态由INTSTAT寄存器中的SLEEPF标志位声明。

每一种USB规范，一个低电压总线供电的USB设备要求在挂起状态吸收不少于500uA电流。一个高功率设备，支持远程唤醒并且允许主机使用远程唤醒功能，这将消耗2.5mA电流。在开头的3ms空闲以后，USB设备在7ms内达到这一状态。这种低电流设备意味着固件将负责在SLEEPF标志被置位1时，赶在USB模块进入挂起模式前进入Stop3模式。

在收到来自USB的恢复信号以后，USB 模块能够产生一个异步中断给MCU，这个中断将把设备带出停止模式并且唤醒时钟。在SLEEPF位被置位，允许这个异步标志信息后，立即设置USBCTL0寄存器中的USBRESMEN为1.USB恢复信号将会引起LPRESF位被置位，表明一个低功耗挂起恢复，这将把CPU从Stop3模式唤醒。在正常操作下，当主机存在发送SOF包时，USB模块不会进入挂起模式。

15.4.6.2 恢复

有三种办离开挂起模式。当USB模块处于挂起状态时，即使所有时钟被禁止并且MCU处于STOP3模式，恢复检测处于活动状态。MCU也能从挂起状态被正常的总线活动激活，例如一个USB复位信号，或者逆流（upstream）复位（远程唤醒）。

15.4.6.2.1 主机初始恢复

一个来主机的从挂起恢复信号，它是一种初始恢复信号（K状态）持续至少20ms并伴随一个标准低速EOP信号。这20ms确保USB网络上的所有设备都被唤醒。在总线恢复以后，主机必需在3ms内开始发送总线运输以防止设备重新进入挂起模式。

根据设备在挂起时的电源模式，主机初始恢复提供的通知方式也是不同的：

运行模式—RESUME必须在SLEEPF被置位允许RESUMEF中断后才能置为1.有了恢复信号后，在USBDP/USBDN线上出现持续2.5us的状态后RESUME中断将会触发。

Stop3模式—SLEEPF被置位以保护LPRESF位,USBRESMEN必须被置位。在总线上处于K状态时，当设备在Stop3模式，LPRESF位将会被置位，表示从低电压挂起状态恢复。这将触发一个异步中断把CPU从Stop3模式唤醒并恢复USB模块时钟。

注意：

作为一种防范措施，在LPRESF被置位以后，固件必须检查USB总线的状态，查看一个瞬时事件的结果是否为K状态并且不是一个真实的主机初始恢复。如果是这种情况，若是必需的，设备能够回复到Stop3模式。为了实现这些，RESUME中断可以与USBRESMEN特性一起被允许。在LPRESF被置位时，并且在时钟重新启动后仍然检测到大约2.5us的K状态，固件能够检查RESUMEF中断已经触发，这表明来自主机的恢复信号。

15.4.6.2.2 USB复位信号

复位可以把一个设备从挂起状态唤醒

15.4.6.2.3 远程唤醒

通过写CRESUME位，USB设备可以发送一个恢复事件给主机。固件必须首先设置此位为1持续USB2.0规范要求的时间（15.1.7.7节）然后清除它。

15.4.7 复位

模块支持多种类型的复位。第一种是由USB主机产生的总线复位，第二种是由MCU产生的模块复位。

15.4.7.1 USB总线复位

在任何时候，USB主机可能向一个或者所有连接到总线上的设备发出复位信号。USB总线复位信号的定义：电缆上朝过2.5us的SE0信号。当设备检测到复位信号，它把自己复位到未配置状态，设置它的USB地址为0.USB主机使用复位信号强制一个或所有连接到总线上的设备进入一个已知状态，进而进行枚举。

USB模块通过判断INTSTAT寄存器中的USBRST中断响应复位信号。中断服务程序应处理这一中断，确保USB模块的正确操作。

15.4.7.2 USB模块复位

USB模块复位用于片内初始化。模块复位期间，USB模块被配置为默认模式。通过设置USBCTL0寄存器中的USBRESET位，USB模块将强制进入复位状态。默认状态包含以下设置：

-中断屏蔽
-USB时钟允许
-USB稳压器禁用
-USB收发器禁用
-USBDP上拉电阻禁用
-端点禁用
-USB地址寄存器设置为0

15.4.8 中断

INTSTAT寄存器的中断标志位标志在正常操作时的中断事件包括：USB起始帧令牌（TOKSOF）,包完成（TOKDNE）,USB总线复位（USBRST）,端点错误（ERROR），挂起和恢复（SLEEP和RESUME），端点停止（STALL）。

ERRSTAT寄存器存储着具体的错误类型信息。所以应用程序可以正确的判断包传送失败

的原因—由于CRC错误,PID校验错误等等。

上面这两个寄存器都可以通过INTENB和ERRENB寄存器屏蔽。寄存器INTSTAT和ERRSTAT在两个层面上发出中断信号。即未被屏蔽的ERRSTAT中的中断想INTSTAT请求中断。

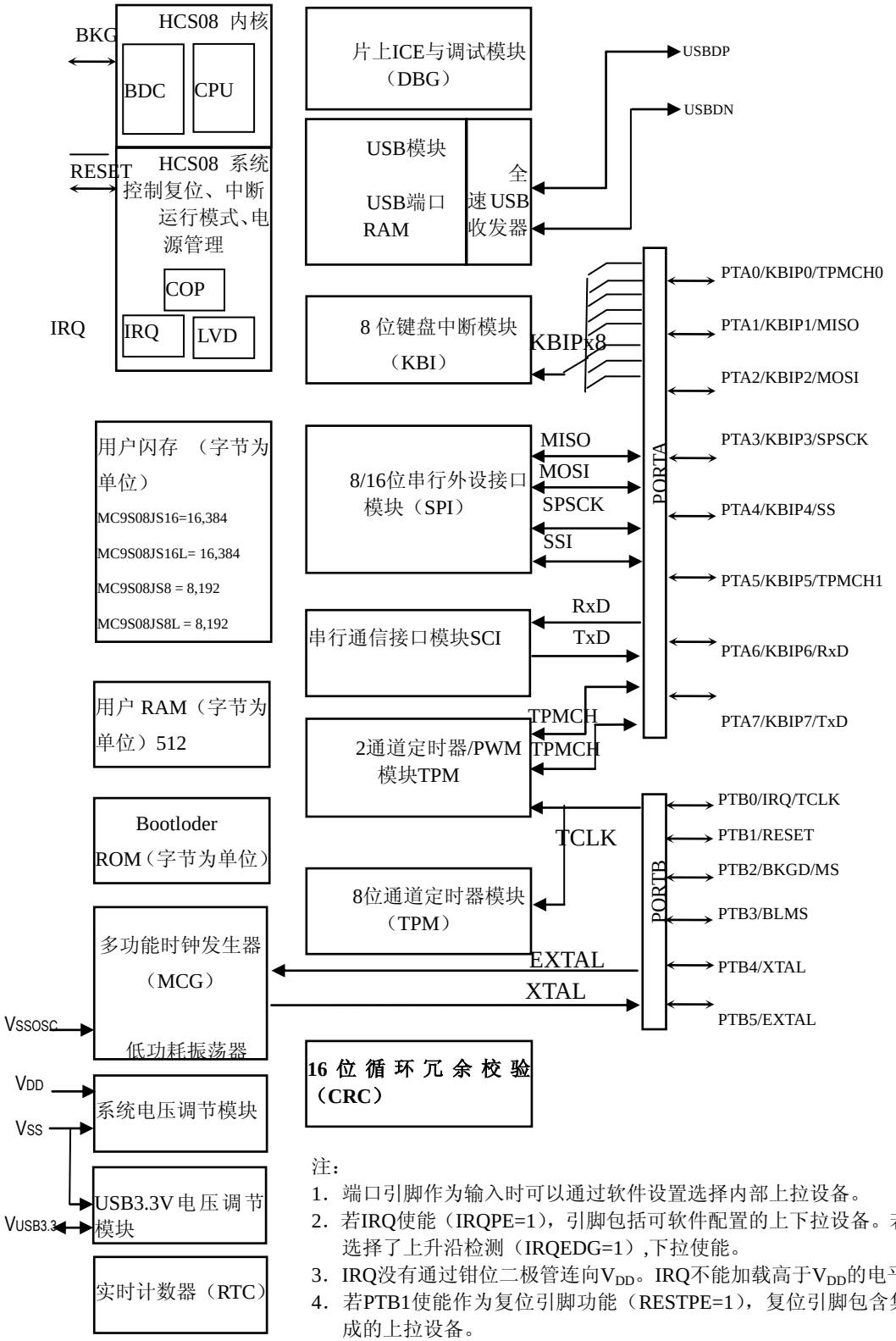
注意:中断寄存器和STAT寄存器配合工作。在收到INTSTAT中断以后，软件应检查STAT寄存器并决定哪一个BDT实体会受影响。

第十六章 环冗余校验发生器（S08CRCV2）

16.1 引言

本章介绍了循环冗余校验（CRC）发生器模块，此模块采用了 16 位循环冗余校验码多项式，X^16+ X^12+ X^5+ 1，生成一个错误检测 CRC 码。模块将 8 位数据计算一次得到一个 16 位代码，并为闪存或 RAM 中所有可访问的内存地址做一次简单检查。

注意：此设备不能使用在 Stop1 模式下。



注：

1. 端口引脚作为输入时可以通过软件设置选择内部上拉设备。
2. 若IRQ使能（IRQPE=1），引脚包括可软件配置的上下拉设备。若选择了上升沿检测（IRQEDG=1）,下拉使能。
3. IRQ没有通过钳位二极管连向V_{DD}。IRQ不能加载高于V_{DD}的电平
4. 若PTB1使能作为复位引脚功能（RESTPE=1），复位引脚包含集成的上拉设备。
5. 引脚包含集成上拉设备。
6. 当KBI使能（KBIPEn=1）而且相关引脚被配置为上拉设备使能，那么上升沿（KBEDGn=1）被用于将上拉设备配置为下拉设备。

图 14-1. MC9S08JS16系列框图CRC模块和引脚高亮显示

16.1.1 特性

CRC 特性包括：

- 16 位位移寄存器硬件产生电路
- CRC 16-CCITT（国际电信联盟-电信标准局）按照多项式 X^16+ X^12+ X^5+ 1 产生校验
- 错误检测，适合所有单、双、奇以及多位错误
- 可编程初始化种子值
- 高速的 CRC 计算

16.1.2 操作模式

此节定义了 CRC 模块在运行、等待、和停止模式下的操作情况。

- 运行模式—基本操作模式
- 等待模式—CRC 模块为可操作的
- Stop1 和 Stop2 模式—CRC 在此模式下不工作，将被置为复位状态直到从停止模式恢复。
- Stop3 模式—在此模式下，CRC 将进入低功耗待机状态。任何当前的 CRC 计算将停止，当 CPU 恢复到运行模式时，计算将被恢复。

16.1.3 模块图

图 16-2 提供了一个 CRC 模块的框图。

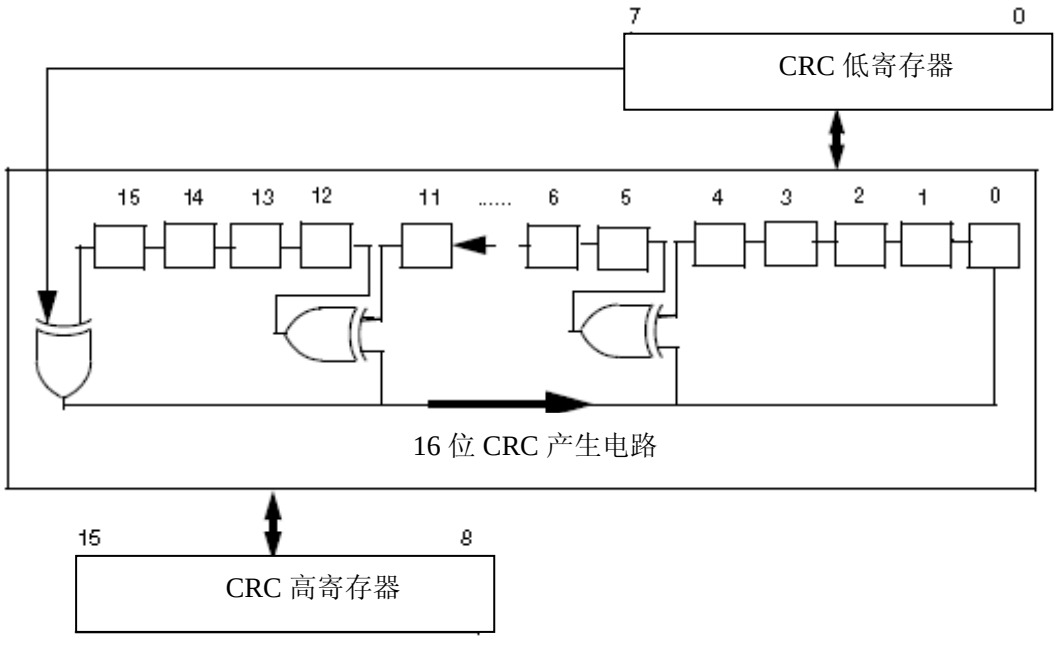


图 16-2. CRC模块框图

16.2 扩展信号描述

无片外的 CRC 信号。

16.3 寄存器定义

16.3.1 内存映射

名称		7	6	5	4	3	2	1	0
CRCH（offset=0）	R	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
	W								
CRCH（offset=1）	R	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
	W								

表 16-1.CRC寄存器概览

注意：偏移量4，5，7也被映射在CRCL寄存器。此别名仅在CF1Core使用（版本1 ColdFire内核），在HCS08内核应该被忽略。

16.3.2 寄存器描述

CRC 模块包括：

- 一个 16 位 CRC 结果和种子寄存器

参照第四节，“内存”，直接页寄存器概要获取所有的 CRC 寄存器绝对地址分配。此节仅仅参开它们的名称。飞思卡尔提供的通用文件或头文件用于将这些名称转化成相应的绝对地址。

16.3.2.1 CRC高字节寄存器（CRCH）

	7	6	5	4	3	2	1	0
读 写	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
复位	0	0	0	0	0	0	0	0

图 16-3.CRC高字节寄存器

表 16-2.寄存器字段描述

字段	描述
7:0 CRCH	**CRCH**—这是 16 位 CRC 寄存器的高字节。写 CRCH 将加载初始化 16 位种子值的高字节直接到 CRC 产生器的移位寄存器的 15-8 位。然后 CRC 产生器期望种子值低字节被写入到 CRCL 并直接加载到移位寄存器的 7-0 位。一旦双方的种子写入 CRCH:CRCL 被加载到 CRC 产生器且一个字节的数据已经写入 CRCL，移位寄存器将开始转移。对 CRCH 的读操作将直接从 CRC 产生器的移位寄存器读取计算结果的 1 5-8 位。

16.3.2.1 CRC低字节寄存器（CRCL）

	7	6	5	4	3	2	1	0
读 写	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
复位	0	0	0	0	0	0	0	0

16-4.CRC低字节寄存器（CRCL）

表16-3.寄存器字段描述

字段	描述
7:0 CRCL	**CRCL**—这是 16 位 CRC 寄存器的低字节。一般的，写 CRCL 将导致 CRC 产生器开始计时。特殊的，若写 CRCH 事件发生，接下来将写 CRCL 将价值寄存器值作为 16 位初始化种子的低字节直接加载到移位寄存器的 7-0 位。对 CRCL 的读操作将直接从 CRC 产生器

	的移位寄存器读取计算结果的 7-0 位。

16.4 功能描述

为了使能 CRC 功能，写 CRCH 寄存器将触发种子机制的上半部分。这将使得 CRCH 值直接存入到 CRC 产生器的移位寄存器的 15- 8 位。CRC 产生器将期望写入 CRCL 寄存器以完成种子机制。

一旦 CRCL 寄存器被写入，其值将直接加载到移位寄存器位 7-0 位，且种子机制下半部分将被完成。此 CRCH:CRCL 的值将为 CRC 产生器种子的初始值。

现在 CRC 计算将采用的第一个字节数据应该被写入到 CRCL 寄存器。种子机制的完成后的此次写入将触发模块的 CRC 校验过程。CRC 产生器将 CRCL 寄存器的值转到移位寄存器（MSB 优先）。在所有的 8 位被移入 CRC 产生器后（数据写入 CRCL 后的下个周期），移位后的值，或当前移位寄存器的值，可以从 CRCH:CRCL 中直接读取。包含在 CRC 计算中的下一个数据字节可以写入到 CRCL 寄存器。

然后，下个字节也通过 CRC 产生器的 16 位移位寄存器移位。在移位结束后，第二次计算的结果可以直接在 CRCH:CRCL 中被读取。

每个字节都转移完成后，一个新的 CRC 结果将出现在 CRCH：CRCL 中，且一个附加的字节可能会被写入到 CRCL 寄存器中。CRCL 寄存器包含在 CRC16-CCITT（国际电信联盟-电信标准局）计算中。每一个 8 位被移入移位寄存器时，CRCH：CRCL 中将产生一个新的结果。

开始一次 CRC 校验，写 CRCH 寄存器，CRC 校验的种子机制也将开始。

16.4.1 ITU-T（CCITT）使用建议以及期望的CRC结果

CRC 多项式 0x1021（x16 + × 12 + X5 的+ 1）俗称循环冗余校验码，众所周知它最初是由 ITU-T（原国际电报电话咨询委员会）委员会提议。

尽管 ITU – T 关于多项式 0x1021 的用途的建议都很明确，但是他们也接受已经被执行的各种方式。

- ITU – T V4.1 执行同一个电路，见“图 16-2”，但是建议用作 SEED=0x0000。
- ITU – T T3.0 和 ITU – T X.25 执行相同的电路见“图 16-2”，但是他们建议 CRC 最终结果取反。（一种反码操作）。同样的，他们推荐 SEED=0XFFFF。

此外，常常可以发现在手册中的电路与所建议的电路略有不同，但是同样为 CRC-CCITT 电路。（许多变化要求用零放大信号）。

在 CRC 模块中实现的电路正是由 ITU – T V.41 建议建议的，电路有一个灵活的可编程的 SEED。正如 ITU – T V.41，不需要放大，结果也没有取反。表 16-4 显示了一些预期的可用作参考结果。请注意，这些是 ASCII 字符串消息。例如，信息“123456789”编码用字节 0x31 到 0x39 编码（见 ASCII 表）。

表 16-4. 预期的CRC结果

ASCII 字符消息	SEED（初始 CRC 值）	CRC 结果
“A”	0x0000	0x58e5
“A”	0xFFFF	0xb915
“A”	0x1d0f	0xb915
“123456789”	0x0000	0x31c3
“123456789”	0xffff	0x29b1

“123456789”	0x1d0f	0xe5cc
256 个连续的大写字符”A”	0x0000	0xabe3
256 个连续的大写字符”A”	0xffff	0xea0b
256 个连续的大写字符”A”	0x1d0f	0xe938

1 RC-CCITT 的普通变量要求用 0 和 SEED=0xFFFF 放大消息。当 SEED=0x1D0F 时 CR 模块将给出相同的结果，且没有放大消息。

16.5 初始化信息

遵循下面的步骤可初始化 CRC 模块和初始化 CEC16-CCIT 计算。

1. 写种子值的高字节打牌 CRCH 中
2. 写种子值的低字节当 CRCL 中
3. 写即将被计算的数据的第一个字节到 CRCL 中（CFI 核提供了可选择的选项。见 16.5” 初始化信息”）
4. 在步骤 3 后的下一个总线周期后，如果需要，CRC 结果可以从 CRCH:CRCL 中读取
5. 重复 3-4 步骤直到数据被校验完毕。

第十七章 开发支持

17.1 引言

HCS08 中的开发支持系统包括背景调试控制（BDC）和片上调试模块（DBG）。BDC 提供单线调试接口，与为片上 FLASH 和其他非可变存储器编程提供方便接口的目标 MCU 相连。BDC 也是开发用的主要调试接口，允许以非介入式方式存取存储器数据和传统调试功能，如 CPU 寄存器修改、断点和单指令跟踪命令等。

在 HCS08 产品系列中，外部引脚上地址和数据总线信号并不能得到（即使在测试模式时也不行）。调试的执行是通过单线背景调试接口向目标 MCU 灌输命令来实现的。调试模块提供了一种有选择性地触发和捕获总线信息的方式，这样外部开发系统可以对 MCU 内发生的事件按周期进行重建，而不需要从外部存取 MCU 的地址和数据信号。

MC9S08JS16 系列的交替 BDC 时钟源是 ICGLCLK。更多 ICGLCLK 信息和如何选用时钟源，请参见第十二章，“内部时钟发生器（S08MCGV1）”。

17.1.1 特性

BDC 模块的特性包括：

- 单引脚进行模式选择和后台调试通信
- BDC 的寄存器不位于存储器映射地址中
- SYNC 命令确定目标通信速率
- 非介入式命令进行存储器存取
- 供 CPU 寄存器存取的激活背景模式命令
- GO 和 TRACE1 命令
- 背景调试命令可以将 CPU 从停止模式或等待模式中唤醒
- BDC 内置一个硬件地址断点
- 如果 BDC 使能，则振荡器运行在停止模式
- 处于激活背景调试模式时，COP 看门狗禁止

ICE 系统的特性包括：

- 两个触发比较器：两个地址+读/写（R/W）或一个完整地址+数据+R/W
- 灵活的 8 字 16 位 FIFO（先进先出）缓存，用于捕获信息：
  - 流程变化的地址或
  - 纯事件数据
- 两个类型的断点：
  - 指令操作码的标记断点
  - 任何地址存取的强制断点
- 九个触发模式：
  - 基本：只有 A，A 或 B
  - 顺序：A 然后 B
  - 全部：A 和 B 数据，A 和非 B 数据
  - 事件（存储数据）：纯事件 B，A 然后纯事件 B
  - 范围：在范围以内（A≤地址≤B），在范围以外（地址<A 或地址>B）

17.2 背景调试控制器（BDC）

HCS08 系列中的所有 MCU 都包含一个单线背景调试接口，它支持片上非易失性存储器的在线编程和先进的非介入式调试功能。与早期 8 位 MCU 的调试接口不同，这个系统不干扰正常的应用资源。它不使用任何用户内存或存储器映射中的地址，也不分享任何片上外设。

BDC 命令分为两大组：

- 激活背景调试模式命令要求目标 MCU 处于激活的背景调试模式（用户程序未运行）。激活背景调试模式命令允许读写 CPU 寄存器，允许用户一次跟踪一个用户指令，或从激活背景调试模式进入用户程序。
- 非介入式命令可以随时执行，即使用户的程序正在运行。非介入式命令允许用户在背景调试控制器中读写 MCU 存储器地址或存取状态和控制寄存器。

一般地，可以用相当简单的接口盒将来自主机的命令转换为与单线背景调试系统连接所需的用户串行接口命令。根据开发工具供应商的不同，这个接口盒也许采用标准 RS-232 串行端口，或是并行打印端口，或是一些其它类型的通信端口，如用来在 PC 和接口盒之间通信的通用串行总线（USB）接口。这个接口盒一般通过接地、BKGD 引脚、RESET，有时还有 V_{DD} 与目标系统连接。复位引脚的开漏连接允许主机强制目标系统复位，这有助于重新获得对失控目标系统的控制，或在片上非易失性存储器重新编程之前，控制目标系统的启动。有时可以用 V_{DD} 来允许接口盒使用目标系统的电源，避免再使用另一个隔离的电源。但是，如果单独对接口盒供电，它可以连接到一个正在运行的目标系统，而不必强制目标系统复位，否则会干扰正在运行的应用程序。

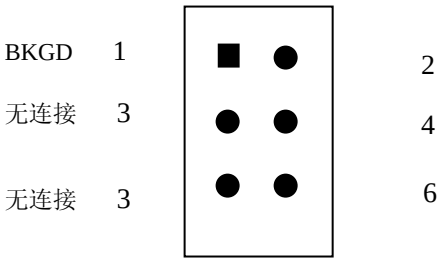


图17 -1.BDM 工具连接器

17.2.1 BKGD引脚描述

BKGD 是单线背景调试接口引脚。这个引脚的主要功能是实现激活背景调试模式命令和数据的双向串行通信。在复位过程中，这个引脚用来选择激活背景调试模式启动或启用用户的应用程序。这个引脚还用来请求定时同步响应脉冲来允许主机开发工具确定背景调试串行通信的正确时钟频率。

BDC 串行通信采用一个定制串行协议首先引入微处理器 M68HC12 系列。这个协议假定主机知道通信时钟速率，这个速率由目标 BDC 时钟速率所决定。所有通信通过主机启动和控制所有通信，主机驱动高到低边沿发出每个位时间开始信号。命令和数据以最重要的位先发（MSB 先发）的方式发送。有关通信协议的详细信息，请参见 17.2.2，“通信详细介绍”。

如果主机尝试与 BDC 时钟速率未知的目标 MCU 通信，可以发送 SYNC 命令给目标 MCU，请求定时同步响应信号，通过这个信号，主机可以判断正确的通信速率。

BKGD 是伪开漏引脚，有一个片上上拉，因此不需要外部上拉电阻。与典型的开漏引脚不同，引脚上的受外部容性的影响的外部 RC 时间常数，在信号上升时间上几乎不起作用。

定制协议提供简洁、活跃驱动加速脉冲，强制使这个引脚上有快速的上升时间，而没有驱动电平冲突风险。参见 17.2.2，“通信详细介绍”，了解更多详情。

当没有调试盒连接 6 引脚的 BDM 接口连接器时，BKGD 的内部上拉会选择正常的操作模式。当调试盒连接到 BKGD 时，可以在 MCU 复位后强制它进入激活背景调试模式。强制激活背景调试的特定条件取决于 HCS08 衍生产品（参见“开发支持”小节的介绍）。不必复位目标 MCU 来通过背景调试接口来与之通信。

17.2.2 通信详细介绍

BDC 串行接口需要外部控制器来在 BKGD 引脚上产生下降沿来指示每个位时间的开始。无论数据是发送或接收，外部控制器都会提供这个下降沿。

BKGD 是伪开漏引脚，可以被外部控制器或 MCU 来驱动。数据以 MSB 先发的形式且以每位 16 个 BDC 时钟周期的速率（标定速率）发送。如果来自主机的下降边沿之间产生 512 个 BDC 时钟周期，则该接口超时。如果出现超时，任何正在进行的 BDC 命令被中止，对目标 MCU 系统的存储器或操作模式没有影响。

定制串行协议要求调试盒知道目标 BDC 通信时钟速率。

BDC 状态和控制寄存器中的时钟开关（CLKSW）控制位允许用户选择 BDC 时钟源。BDC 时钟源可以是总线，或备用的 BDC 时钟源。

BKGD 引脚可以接收高或低电平，或发送高或低电平。下图显示了每种情况的时序。接口时序与目标 BDC 中的时钟同步，但是与外部主机异步。显示的内部 BDC 时钟信号是计数周期的参考。

图 17-2 显示了外部主机将逻辑 1 或 0 发送到目标 HCS08MCU 的 BKGD 引脚。主机与目标异步，因此主机生成的下降边沿与目标所认为的位时间起始点有 0 到 1 周期的延迟。10 个目标 BDC 时钟周期后，目标获得 BKGD 引脚的电平。一般地，主机在主机到目标方向的传输过程中主动驱动伪开漏 BKGD 引脚，以加快上升边沿。由于目标在主机至目标方向的传输周期中不驱动 BKGD 引脚，因此没有必要在此期间将线路作为开漏信号。

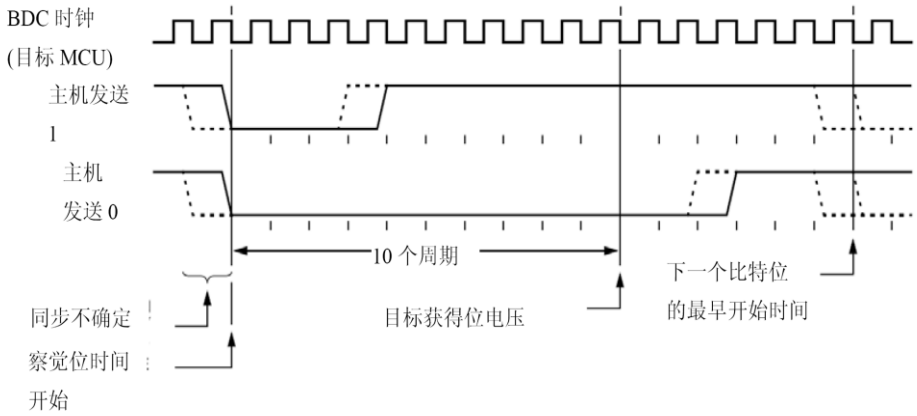


图17-2. BDC主机到目标机串行位时序

图 17-3 显示主机从目标 HCS08MCU 收到逻辑 1。由于主机与目标 MCU 异步，因此主机在 BKGD 引脚上产生下降边沿与目标 MCU 所认为的位时间起始点有 0 到 1 个周期的延迟。主机保持低 BKGD 引脚足够长的时间，使目标识别它（至少两个目标 BDC 周期）。主机必须在目标 MCU 所认定的位计时开始后，驱动简单主动高态加速脉冲七个周期前，释放低电平驱动。主机应该在其位启动时间约 10 个周期后采样位电平。

图17-3. BDC目标到主机串行位时序（逻辑1）

图 17-4 显示了主机从目标 HCS08MCU 收到逻辑 0。由于主机与目标异步，因此主机在 BKGD 引脚上产生下降沿与目标 MCU 所认为的位时间起始点有 0 到 1 个周期的延迟。主机启动位时间，但是目标 HCS08MCU 完成它。由于目标希望主机接收逻辑 0，它保持 BKGD 引脚低电平 13 个 BDC 时钟周期，然后驱动 BKDG 引脚置高，加速上升沿。主机在启动位时间约 10 个周期后采样位电平。

图17-4. BDM目标到主机串行位时序（逻辑0）

17.2.3 BDC命令

BDC 指令由主机串行发向目标 HCS08MCU 的 BKGD 引脚。所有的命令和数据都通过定制 BDC 通信协议先发送最高位。激活背景模式命令需要目标 MCU 当前处在激活背景模式，而非介入式命令可在任何时候发出，无论目标 MCU 是处在激活背景模式还是在运行用户应用程序。

表 17-1 显示了所有 HCS08 BDC 命令，并简要描述了它们的编码结构，以及每个命令的含义。

代码结构术语。这些术语在表 17-1 中使用来描述 BDC 命令的编码结构。

指令开始于主机到目标机方向（先发送最高位）的 8 位十六进制代码

/	==	命令分隔符
d	==	延时 16 个目标 BDC 时钟周期
AAAA	==	主机向目标机方向传送的 16 位地址
RD	==	目标机向主机方向发送的 8 位读取数据
WD	==	主机向目标机方向发送的 8 位写入数据，
RD16	==	主机向目标机方向发送的 16 位读取数据，
WD16	==	主机向目标机方向发送的 16 位写入数据，
SS	==	目标机向主机发送方向（状态） BDCSCR 的内容
CC	==	主机向目标机发送方向（控制）对 BDCSCR 写入的 8 位数据
RBKP	==	目标机向主机方向传送（来自 BDCBKPT 断点寄存器）的 16 位读取数据
WBKP	==	主机向目标机方向传送（写入 BDCBKPT 断点寄存器）的 16 位写入数据

表17-1. BDC命令概要

命名	激活 DBM/非介入式	指令结构	描述
SYNC	非介入式	n/a[1]	请求定时参考脉冲以决定目标 BDC 通信速率
ACK_ENABLE	非介入式	D5/d	使能响应协议。参见飞思卡尔文档编号 HCS08RMv1/D。
ACK_DISABLE	非介入式	D6/d	禁止响应协议。参见飞思卡尔文档编号 HCS08RMv1/D。
BACKGROUND	非介入式	90/d	如果允许，则进入背景模式（如果 ENBDM=0，则忽略）
READ_STATUS	非介入式	E4/SS	从 BDCSCR 读取 BDC 状态
WRITE_CONTROL	非介入式	C4/CC	写 BDC 控制位到 BDCSCR 中
READ_BYTE	非介入式	E0/AAAA/d/RD	从目标内存中读一个字节
READ_BYTE_WS	非介入式	E1/AAAA/d/SS/RD	读一个字节并报告状态
READ_LAST	非介入式	E8/SS/RD	从地址重新读字节，仅读和报告状态
WRITE_BYTE	非介入式	C0/AAAA/WD/d	写一个字节到目标内存
WRITE_BYTE_WS	非介入式	C1/AAAA/WD/d/SS	写一个字节并报告状态
READ_BKPT	非介入式	E2/RBKP	读 BDCBKPT 断点寄存器
WRITE_BKPT	非介入式	C2/WBKP	写 BDCBKPT 断点寄存器
GO	激活 BDM	08/d	执行开始于 PC 所指示的当前地址的用户应用程序
TRACE1	激活 BDM	10/d	跟踪 PC 中地址处的 1 条用户指令，然后回到激活背景模式
TAGGO	激活 BDM	17/d	与 GO 相同，但激活外部标签（HCS08 设备没有外部标签引脚）
READ_A	激活 BDM	68/d/RD	读累加器（A）
READ_CCR	激活 BDM	69/d/RD	读条件代码寄存器（CCR）

READ_PC	激活 BDM	6B/d/RD16	读程序计数器（PC）
READ_HX	激活 BDM	6C/d/RD16	读 H 和 X 寄存器（H:X）
READ_SP	激活 BDM	6F/d/RD16	读堆栈指针（SP）
READ_NEXT	激活 BDM	70/d/RD	以 1 为基数递增 H:X，然后读位于 H:X 的存储器字节
READ_NEXT_WS	激活 BDM	71/d/SS/RD	以 1 为基数递增 H:X，然后读位于 H:X 的存储器字节。报告状态和数据
WRITE_A	激活 BDM	48/WD/d	写累加器（A）
WRITE_CCR	激活 BDM	49/WD/d	写条件代码寄存器（CCR）
WRITE_PC	激活 BDM	4B/WD16/d	写程序计数器（PC）
WRITE_HX	激活 BDM	4C/WD16/d	写 H 和 X 寄存器（H:X）
WRITE_SP	激活 BDM	4F/WD16/d	写堆栈指针（SP）
WRITE_NEXT	激活 BDM	50/WD/d	以 1 为基数递增 H:X，然后写位于 H:X 的存储器字节
WRITE_NEXT_WS	激活 BDM	51/WD/d/SS	以 1 为基数递增 H:X，然后写位于 H:X 的存储器字节。报告状态和数据

[1].SYNC 指令是一种特殊的操作，它没有指令代码。

SYNC 命令与其它 BDC 命令不同，因为主机没有必要知道 BDC 通信的正确通信速率，直到它分析完 SYNC 命令的响应后。

要发出 SYNC 命令，主机：

- 保持 BKGD 引脚为低电平至少 128 个周期，而且是以最慢的可能的 BDC 时钟来计（最慢的时钟一般是参考振荡器 64 分频或自时钟速率 64 分频。）
- 驱动 BKGD 达到高电平，实现瞬态加速，快速上升时间（这个加速脉冲一般是系统中最快时钟的一个周期）。
- 去除 BKGD 引脚的所有驱动，这样它可回复到高阻抗。
- 监视 BKGD 引脚得到同步响应脉冲

当检测到主机的 SYNC 请求（比在正常 BDC 通信过程中发生的慢时钟要长），则目标：

- 等待 BKGD 返回到逻辑高电平
- 延迟 16 个周期，允许主机停止驱动高电平加速脉冲
- 驱动 BKGD 低态 128 个 BDC 时钟周期
- 驱动一个周期的高电平加速脉冲，在 BKGD 上实现快速上升时间
- 去除 BKGD 引脚的所有驱动，这样它可回复到高阻抗

主机测量这个 128 个周期的响应脉冲的低电平时间，判断进行后续 BCD 通信的正确通信速率。主机一般可以确定正确的通信速率，与实际目标速率的误差只有百分之几，通信协议能够接受百分之几的速率误差。

17.2.4 BDC硬件断点

BDC 包括一个相对简单的硬件断点，它将 CPU 地址总线看成是 BDCBKPT 寄存器中的 16 位匹配值。这个断点可以生成强制断点或标记断点。强制断点使 CPU 在存取断点地址后的第一个指令边界进入激活背景调试模式。标记的断点使指令操作码在断点地址被标记，这样当 CPU 到达指令队列的终点时，将进入激活背景模式，而不是执行该指令。这意味着标记的断点只能放置在指令操作代码的地址上，而强制断点可以设置在任何地址。

BDC 状态和控制寄存器（BDCSCR）中的断点使能（BKPTEN）控制位用来激活断点逻辑（BKPTEN=1）。当 BKPTEN=0，即复位后它的默认值，断点逻辑禁止，无论其它 BDC 断点中的值是多少，也不管控制位如何，均不请求断点。BDCSCR 中的强制/标记选择（FTS）控制位用来选择强制（FTS=1）或标记（FTS=0）类型断点。

片上调试模块（DBG）包括两个额外的硬件断点的电路，这两个硬件断点比 BDC 模块中的简单断点更灵活。

17.3 片上调试系统（DBG）

由于 HCS08 器件没有外部地址和数据总线，在线仿真器最重要的功能已经构建在 MCU 的芯片上。这种调试系统包含可以存储地址或数据总线信息的 8 级 FIFO，和一个确定何时捕获总线信息以及捕获哪些总线信息的灵活触发系统。这个系统依赖单线背景调试系统来存取调试控制寄存器，读取 8 级 FIFO 的结果。

调试模块包括控制和状态寄存器，可以在用户存储器映射中存取。这些寄存器位于高地址寄存器空间中，避免使用宝贵的直接页面存储器空间。

调试模块的大多数功能在开发过程使用，用户程序很少存取调试模块的任何控制和状态寄存器。一个例外就是调试系统可以提供一种手段来实施某种形式的 ROM 补丁。15.3.6，“硬件断点”中对此有更详细的描述。

17.3.1 比较器A和B

两个 16 位比较器（A 和 B）可以选择用 R/W 信号或一个操作码跟踪电路来鉴定。单独的控制位允许忽略每个比较器的 R/W 位。操作码跟踪电路可选地允许被规定，如果操作码在规定的地址实际执行，而不是只从存储器读到指令队列中，则触发将发生。比较器还能够进行庞大的比较，支持范围内和范围外触发模式。在所有 BDC 存取过程中，比较器被临时禁止。

比较器 A 总是与 16 位 CPU 地址相关联。比较器 B 根据所选的触发模式比较 CPU 地址或 8 位 CPU 数据总线。由于 CPU 数据总线分为只读数据总线和写数据总线， RWAEN 和 RWA 控制位有一个额外的目的，在完整地址加上数据比较中，它们被用来确定其中哪些总线用在比较器 B 数据总线比较中。如 RWAEN=1（激活），RWA=0（写），则使用 CPU 的写数据总线，否则用 CPU 的只读数据总线。

当比较器检测到合格的匹配条件时，当前选择触发模式确定调试器逻辑做什么。匹配可以导致以下情况：

- 生成 CPU 断点
- 将数据总线值存储到 FIFO 中
- 开始将流变化地址存储到 FIFO 中（开始类型跟踪）
- 停止将流变化地址存储到 FIFO 中（结束类型跟踪）

17.3.2 总线捕获信息和FIFO操作

使用 FIFO 的通常方式是建立触发模式和其它控制选项，然后打开调试器。当 FIFO 填满后，或调试器停止将数据存储到 FIFO 后，你可以按信息存储到 FIFO 的顺序从中读取信息。状态位指示数据所在的 FIFO 中的有效信息的字数。如果在 FIFO 满（CNT=1:0:0:0）之前将 ARM 写为 0，以人工停止跟踪，信息移动一个位置，主机必须执行（（8-CNT）-1）FIFO 虚读操作，使信息进入到 FIFO 中的第一个重要入口。

在大多数触发模式中，存储在 FIFO 中的信息包含 16 位流变化地址。在这些情况中，先读 DBGFH 然后读 DBGFL，从 FIFO 中获得一个一致的信息字节。读 DBGFL（FIFO 数据端口的低阶字节）会使 FIFO 移动，这样下一个信息字可以在 FIFO 数据端口得到。在仅事件触发模式（参见 15.3.5，“触发模式”）中，8 位数据信息存储在 FIFO 中。在这些情况中，FIFO（DBGFH）的高半部分没有被使用，仅仅通过读 DBGFL 来从 FIFO 中读出数据。每次读 DBGFL 时，FIFO 都会移动，这样通过 DBGFL 的 FIFO 数据端口可以获得下一个数据值。

在触发模式中，FIFO 保存流变化地址，CPU 地址与 FIFO 的输入端有一个延迟。由于这个延迟，如果触发事件本身是一个流变化地址或在触发事件启动 FIFO 后下两个周期中出现了流变化地址，它将不保存在 FIFO 中。如果是结束跟踪的情况，当触发事件是一个流变化，则它将保存为运行的调试器的最后一个流变化入口。

当调试器没有打开时，FIFO 还可以用来生成所执行指令地址的分析。当 ARM=0，读 DBGFL 会使最近获取的操作码的地址保存在 FIFO 中。采用分析功能，主机调试器将从 FIFO 中读取地址，即以常规的间隔先读 DBGFH 然后读 DBGFL。前 8 个值将被丢弃，因为它们对应于初始需要填充 FIFO 的 8 个 DBGFL 读取。DBGFH 和 DBGFL 的其它周期读取则返回关于所执行指令的延迟信息，这样主机调试器可以对执行指令地址进行分析。

17.3.3 流变化信息

为了减少存储在 FIFO 中的信息数量，只保存与使正常的指令执行顺序发生变化的指令相关的信息。知道存储在目标系统中的源和对象代码程序后，外部调试器可以通过来自 FIFO 中存储的大量流变化信息的许多指令来重现执行路径。

对于采用了分支的条件分支指令（分支条件为真），则保存源地址（条件分支操作码的地址）。由于 BRA 和 BRN 指令不是条件的，这些事件不会使流变化信息存储在 FIFO 中。

间接 JMP 和 JSR 指令采用 H:X 索引地址寄存器对的内容，确定目的地址，这样调试系统为任何间接 JMP 或 JSR 保存运行时的目的地址。对于中断，RTI 或 RTS，目的地址作为流变化信息存储在 FIFO 中。

17.3.4 标记vs.强制断点和触发器

标记一词指当指令操作码被取到指令队列时识别它，但是不采取任何其它操作，直到并且除非指令被 CPU 真正执行才操作。这种区分非常重要，因为任何因跳转、分支、子例程调用、或中断而发生的流变化都会导致一些指令被取到指令队列，未执行就被丢弃。

强制类型的断点等待当前指令完成，然后执行断点请求操作。通常操作是进入激活后台调试模式，而不是继续用户应用程序中的下一个指令。

标记 vs.强制这一术语在调试模块的两种情况下使用。第一种情况指从调试模块向 CPU 发送断点请求。第二种情况指从比较器向调试控制逻辑发送匹配信号。当标记类断点发送给 CPU 时，信号与操作码一起进入指令队列，这样当这个操作码被执行时，CPU 将有效地用 BGND 操作码代替被标记的操作码，这样 CPU 进入激活后台调试模式，而不是执行被标记的指令。当 DBGT 寄存器中的 TRGSEL 控制位被设置为选择标记类操作，比较器 A 或 B 的输出被调试模块中的逻辑块鉴定，这个逻辑块跟踪操作码，如果比较地址的操作码被实际执行，则只向该调试器生成一个触发。每个比较器都有单独的操作码跟踪逻辑，这样整个指令队列一次不止一个比较事件被跟踪。

17.3.5 触发模式

触发模式控制在调试运行中的整体行为。DBGT 寄存器中的 4 位 TRG 字段选择九个触

发模式中的一个。当 DBGT 寄存器中的 TRGSEL=1，比较器的输出必须在触发 FIFO 操作前通过操作码跟踪电路传播。DBGT 中的 BEGIN 位选择当检测到合格的触发时 FIFO 是否开始存储数据（开始跟踪），或 FIFO 从其打开之时开始循环存储数据，直到检测到合格的触发（结束触发）。

写 1 到寄存器中的 ARM 位可启动调试运行，它设置 DBGS 中的 ARMF 标记，并清除 AF 和 BF 标记及 CNT 位。当 FIFO 满时，开始跟踪调试运行结束。在所选触发事件发生时，结束跟踪运行结束。任何调试运行均可通过写 0 到 DBGC 中的 ARM 或 DBGEN 位停止。

除纯事件模式外的所有触发模式中，FIFO 都会存储流变化地址。在纯事件触发模式中，FIFO 将数据存储在 FIFO 的低八位。

控制位在纯事件触发模式中被忽略，而且所有这样的调试运行都是开始类型跟踪。当 TRGSEL=1 选择操作码获取触发器，无需在比较中使用 R/W 位，因为操作码标签只应用于操作码获取，而这一直都是读周期。在采用全模式触发器时，规定 TRGSEL=1 也是不正常的，因为操作码的值通常在特定的地址可以知道。

下面的触发模式描述只说明了导致触发的主要比较器条件。比较器 A 或 B 通常都可以被 R/W 进一步鉴定，通过将 RWAEN（RWBEN）和相应的 RWA（RWB）值设置为与 R/W 相匹配。如果 BRKEN=1，来自比较器的带可选 R/W 鉴定的信号，用来请求 CPU 断点，TAG 决定 CPU 请求是标记请求还是强制请求。

仅 A——当地址匹配比较器 A 的值时触发

A 或 B——当地址匹配比较器 A 或 B 的值时触发

A 然后 B——当地址匹配比较器 B 但只能在另一个周期的地址匹配比较器 A 的值以后触发。可能在 A 匹配后 B 匹配前有许多周期。

A 和 B 数据（全模式）——这称为全模式，因为地址，数据和 R/W（可选）必须在同一个总线周期内匹配，才能产生触发事件。比较器 A 检查地址，比较器的低字节检查数据，如果 RWAEN=1，R/W 对照 RWA 进行检查。比较器 B 的高半部分没有使用。

在全触发模式中，规定标签类 CPU 断点（BRKEN=TAG=1）没有使用，但是如果使用了，就会忽略比较器 B 的数据匹配，导致向 CPU 发送标签请求，当比较器 A 地址匹配时发送 CPU 断点。

A 但非 B 数据（全模式）——地址必须匹配比较器 A，数据必须不能匹配比较器 B 的低位部分，如果 RWAEN=1，R/W 必须匹配 RWA。所有三个条件必须在同一个总线周期中达到才能引起触发。

在全触发模式中，规定标签类 CPU 断点（BRKEN=TAG=1）没有使用，但是如果被使用了，就会忽略比较器 B 数据匹配，导致向 CPU 发送标签请求，当比较器 A 地址匹配时发送 CPU 断点。

纯事件 B（存储数据）——当地址每次匹配比较器 B 的值时，触发事件发生。触发事件导致数据被捕获到 FIFO 中。当 FIFO 满时调试运行结束。

A 然后纯事件 B（存储数据）——当地址匹配比较器 A 中的值后，每次地址匹配比较器 B 中的值时，触发事件发生。触发事件导致数据被捕获到 FIFO 中。当 FIFO 满时调试运行结束。

范围内（A≤地址≤B）——当地址大于或等于比较器 A 的值，且小于等于比较器 B 的值时，触发发生。

范围外（地址<A 或地址>B）——当地址小于比较器 A 的值，或大于比较器 B 的值时，触发发生。

17.3.6 硬件断点

DBGC 寄存器中的 BRKEN 控制位可以设置为 1 来允许使用 15.3.5，“触发模式”所描述的任何触发条件，向 CPU 生成硬件断点请求。DBGC 中的 TAG 控制断点请求是否处理为标记类断点或强制类断点。标记断点使当前的操作码进入指令队列时被标记。如果标记的操作码达到队列的末端，CPU 执行 BGND 指令，进入激活后台调试模式，而不是执行被标记的操作码。强制类断点使 CPU 完成当前指令，然后进入激活后台调试模式。

如果后台调试模式没有被通过 BKGD 引脚的串行 WRITE_CONTROL 命令激活（ENBDM=1），CPU 将执行 SWI 指令，而不是进入激活背景调试模式。

17.4 寄存器定义

本小节包括 BDC 和 DBG 寄存器及控制位的描述。

参见本文数据表的设备描述章节中的高页寄存器概述，了解所有 DBG 寄存器的绝对地址分配。本小节只按名字涉及了寄存器和控制位。使用飞思卡尔提供的等式或头文件，将这些名称翻译为相应的绝对地址。

17.4.1 BDC寄存器和控制位

BDC 包括两个寄存器：

- BDC 状态和控制寄存器（BDCSCR）是一个 8 位寄存器包括背景调试控制器的控制和状态位。
- BDC 断点匹配寄存器（BDCBKPT）保存 16 位断点匹配地址。

这些寄存器通过专用的串行 BDC 命令访问，没有位于目标 MCU 的存储器空间（所以它们没有地址，不能被用户程序访问）。

BDCSCR 的一些位有写限制；否则，这些寄存器可以在任何时候读写。例如当 MCU 处于激活背景模式时，ENBDM 控制位可能不允许写。（这避免了当 MCU 已经处于激活背景模式时，防止控制位的模糊条件禁止激活背景模式的模糊条件。）同时 4 个状态位（BDMACT、WS、WSF 和 DVF）是只读状态标识位，不能通过 WRITE_CONTROL 串行 BDC 命令写。时钟开关（CLKSW）控制位随时都可读或写。

17.4.1.1 BDC状态和控制寄存器（BDCSCR）

该寄存器可以通过串行 BDC 命令（READ_STATUS 和 WRITE_CONTROL）读或写。但是不能被用户程序访问，因为它不位于 MCU 的普通内存映射中。

	7	6	5	4	3	2	1	0
读	ENBAM	BDMACT	BKPTEN	FTS	CLKSW	WS	WSF	DVF
写								
复位	0	0	0	0	0	0	0	0
BDM 复位		=未执行或保留						

图17-5. BDC状态和控制寄存器（BDCSCR）

表17-2. BDCSCR寄存器域描述

域	描述

7 ENBDM	**使能 BDM（允许激活后台模式）** — 通常，在调试会话开始后或只要调试主机复位目标并且直到正常复位使它清零时仍为 1，该位通过调试主机快速写 1。如果应用可进入停止模式，如果调试需要这位需要置位。 0 BDM 不能激活（允许非介入性命令）。 1 BDM 可以激活允许激活后台模式命令。
6 BDMACT	**后台激活模式状态** — 该位为只读位。 0 BDM 无效（用户程序正在运行）。 1 BDM 有效并且等待串行命令。
5 BKPTEN	**BDC 断点使能** — 如果该位被清零，则 BDC 断点不可用，同时 FTS 控制位和 BDCBKPT 匹配寄存器将被忽略。 0 禁止 BDC 断点。 1 允许 BDC 断点。
4 FTS	**强制/标签选择位** — 当 FTS=1 时，无论 CPU 地址总线是否与 BDCBKPT 匹配寄存器相匹配，都会产生断点。当 FTS=0 时，CPU 地址总线和 BDCBKPT 寄存器之间的匹配将会给已获得的操作码作标记。若该被标记的操作码没有到达指令队列的末尾，则 CPU 进入激活后台模式而不是执行已标记的操作码。 0 在断点地址处给操作码做标记，如果 CPU 尝试执行这条指令就进入激活后台模式。 1 在下一个指令边界，断点匹配强制执行激活后台模式（地址无需是一个操作码）。
3 CLKSW	**选择 BCD 通信时钟源** — CLKSW 默认 0，选择其它 BDC 时钟源。 0 其它 BDC 时钟源 1 MCU 总线时钟
2 WS	**等待/停止状态位** — 当目标 MCU 处于等待/停止模式时，大部分 BDC 指令将不被执行。然而， BACKGROUND 指令可用于使目标 CPU 跳出等待/停止模式并进入激活后台模式，在此所有的 BDC 指令都能运行。无论何时主机迫使目标 MCU 进入激活后台模式，主机在尝试其他 BDC 指令前都应当发出一个 READ_STATUS 指令来核实 BDMACT=1。 0 目标 CPU 正执行用户程序代码或正处于激活后台模式（当后台模式被激活时，不是等待/停止模式）。 1 目标 CPU 处于等待/停止模式，或者使用 BACKGROUND 指令使它从等待/停止模式变为激活后台调试模式。
1 WSF	**等待/停止失败状态位** — 如果内存访问指令时由于目标 CPU 在差不多同一时刻执行了一条等待或停止指令，而导致失败时则置位该位。通常的恢复策略是发出一个 BACKGROUND 指令来跳出等待或停止模式而进入激活后台模式，如果重复指令仍然失败，则回到用户程序。（如果需要的话，主机可以恢复 CPU 寄存器并用堆栈存储数值，同时重新执行等待或停止指令。） 0 内存访问没有与等待或停止指令发生冲突。 1 由于 CPU 进入等待或停止模式，内存访问指令失败。
0 DVF	**数据有效失败状态** — 这个状态没有在 MC9S08AW60 系列中使用，因为它没有缓存存储器。 0 存储器存取与缓存存储器不冲突 1 存储器存取命令失败，因为 CPU 没有完成缓存存储器的存取

17.4.1.2 BDC断点匹配寄存器（BDCBKPT）

这个 16 位的寄存器用于存储 BDC 硬件断点地址。BDCSCR 中的 BKPTEN 和 FTS 控制位用于使能和配置断点逻辑。专门的串行 BDC 指令（READ_BKPT 和 WRITE-BKPT）用于

读写 BDCBKPT 寄存器，但是用户程序不能存取它，因为它不在 MCU 的普通存储器映射空间中。当目标 MCU 在运行用户应用程序之前处于激活背景模式时正常设置断点。关于建立和使用 BDC 中的硬件断点逻辑的更多信息，请参见 15.2.4，“BDC 硬件断点”。

17.4.2 系统背景调试强制复位寄存器（SBDFR）

该寄存器仅包含一个只写控制位。必须要用一个串行背景模式命令，例如 WRITE_BYTE，来写 SBDFR。从用户程序试图写该寄存器将被忽略。读该寄存器总是返回 0x00。

	7	6	5	4	3	2	1	0
读	0	0	0	0	0	0	0	0
写								BDFR1
复位	0	0	0	0	0	0	0	0

1. BDFR 只有通过串行后台模式调用命令才可以写，不能通过用户程序来

图17-6 系统后台调试强制寄存器（SBDFR）

表17-3. 寄存器域描述

域	描述
0 BDFR	**后台调试强制复位** — 一系列激活后台模式命令，如 WRITE_BYTE 等，允许外部调试主机强制目标系统复位。向该位写 1，强制 MCU 复位。该位不能从用户程序写。

17.4.3 DBG寄存器和控制位

这个调试模块包括 9 个字节的寄存器空间，分别是三个 16 位寄存器和三个 8 位控制状态寄存器。这些寄存器位于普通存储器映射空间的高地址空间中，这样它们可以存取正常的应用程序。普通用户应用程序几乎从不接入这些寄存器，除了使用断点逻辑的 ROM 修补机制。

17.4.3.1 调试比较器A高寄存器（DBGCAH）

这个寄存器包含比较器 A 的高 8 位的比较值。在复位时，这个寄存器被强制设置为 0x00，除了 ARM=1 之外可以随时被读或写。

17.4.3.2 调试比较器A低寄存器（DBGCAL）

这个寄存器包含比较器 A 的低 8 位的比较值。在复位时，这个寄存器被强制设置为 0x00，除了 ARM=1 之外可以随时被读或写。

17.4.3.3 调试比较器B高寄存器（DBGCBH）

这个寄存器包含比较器 B 的高 8 位的比较值。在复位时，这个寄存器被强制设置为 0x00，除了 ARM=1 之外可以随时被读或写。

17.4.3.4 背景比较器B低寄存器（DBGCBL）

这个寄存器包含比较器 B 的低 8 位的比较值。在复位时，这个寄存器被强制设置为 0x00，除了 ARM=1 之外可以随时被读或写。

17.4.3.5 调试FIFO高寄存器（DBGFH）

这个寄存器提供对 FIFO 的高 8 位的只读接入。写该寄存器没有意义或无效。在纯事件触发模式中，FIFO 只将数据存储在每个 FIFO 字的低字节，因此这个寄存器不能使用，将

读 0x00。

读 DBGFH 不会导致 FIFO 移动到下一个字。当从 FIFO 中读出 16 位字时，在读 DBGFL 前先读 DBGFH，因为读 DBGFL 会导致前进到下个字的信息。

17.4.3.6 调试FIFO低寄存器（DBGFL）

这个寄存器提供对 FIFO 的低 8 位的只读存取。写该寄存器没有意义或无效。

读 DBGFL 会导致 FIFO 移动到下一个字的信息。当调试模块以纯事件模式运行时，只有 8 位数据存储在 FIFO（每个 FIFO 字的高字节部分没有使用）。当从 FIFO 中读出 8 位字时，只需重复地读 BDGFL，从 FIFO 中获得数据的连续的字节。在这种情况下，没有必要读 DBGFH。

当 FIFO 仍然打开时（打开之后，但 FIFO 满或 ARMF 被清除前）不要试图从其中读数据，因为在 DBGFL 读取过程中， FIFO 被阻止前进到下一个字信息。这可以干扰正常的 FIFO 的读取顺序。

在调试器没有打开的情况下读 DBGFL，会使最近获取的操作码的地址存储到 FIFO 中的最后位置。定期读取 DBGFH 之后读 DBGFL，外部主机软件可以开发程序执行的概况。在对 FIFO 进行八次读取后，第九次读取将返回当作第一次读取结果的信息。要使用分析功能，则需要读取 FIFO 八次，且不使用准备好顺序的数据，然后开始使用数据来获取已执行地址的延迟概貌。存储在 FIFO 中的关于 DBGFL （且 FIFO 没有打开）读取的信息就是最近所获操作码的地址。

17.4.3.7 调试控制寄存器（DBGC）

该寄存器可在任何时候被读或写。

	7	6	5	4	3	2	1	0
读 写	DBGEN	ARM	TAG	BRKEN	RWA	RWAEN	RWB	RWBEN
复位	0	0	0	0	0	0	0	0

图17-7. 调试控制寄存器（DBGC）

表17-4. DBGC寄存器域描述

域	描述
7 DBGEN	**调试模块启用**——用来启用调试模块。如果 MCU 是安全的，DBGEN 不能设置为 1。 0 DBG 禁用 1 DBG 启用
6 ARM	**打开控制**——控制调试器是否在 FIFO 中比较和存储信息。采用写操作来设置该位（和 ARMF 位），完成调试运行就自动清除它。将 ARM 或 DBGEN 写为 0，可以停止任何调试运行。 0 调试器没有打开 1 调试器被打开
5 TAG	**标记/强制选择**——控制送到 CPU 的中断请求是否为标签或强制型请求。如果 BRKEN=0，这个位就没有意义或无效。 0 CPU 中断请求作为强制型请求 1 CPU 中断请求作为标签型请求
4 BRKEN	**中断启用**——控制触发事件是否向 CPU 生成中断请求。触发事件可以使信息存储在 FIFO 中而不必向 CPU 生成中断请求。对于结束跟踪，如果比较器和 R/W 满足触发条件，则发出 CPU 中断请求。对于起始跟踪，则当 FIFO 满时发出 CPU 中断请求。TRGSEL

	不影响 CPU 中断请求的定时。 0 CPU 中断请求未启用 1 触发器触发向 CPU 发出中断请求
3 RWA	**比较器 A 的 R/W 比较值**——当 RWAEN=1，这个位确定是否用读或写接入来鉴定比较器 A，当 RWAEN=0，RWA 和 R/W 信号不影响比较器 A。 0 比较器 A 只在写周期上匹配 1 比较器 A 只在读周期上匹配
2 RWAEN	**启用比较器 A 的 R/W**——控制比较器 A 的匹配是否考虑这个水平的 R/W。 0 R/W 未用在比较 A 中 1 R/W 用在比较 A 中
1 RWB	**比较器 B 的 R/W 比较值**——当 RWBEN=1，这个位确定是否用读或写接入来鉴定比较器 B。当 RWBEN=0，RWA 和 R/W 信号不影响比较器 B。 0 比较器 B 只在写周期上匹配 1 比较器 B 只在读周期上匹配
0 RWBEN	**启用比较器 B 的 R/W**——控制比较器 B 的匹配是否考虑这个水平的 R/W。 0 R/W 未用在比较 B 中 1 R/W 用在比较 B 中

17.4.3.8 调试触发寄存器（DBGT）

这个寄存器在任何时候都可以读，但是只有当 ARM=0 时才可以写，除非位 4 和位 5 硬件连线至 0。

	7	6	5	4	3	2	1	0
读 写	TRGSEL	BEGIN	0	0	TRG3	TRG2	TRG1	TRG0
复位	0	0	0	0	0	0	0	0

图17-8. 调试触发寄存器（DBGT）

表17-4. DBGT寄存器域描述

域	描述
7 TRGSEL	**触发类型**——控制比较器 A 和 B 的匹配输入是否与调试模块中的操作码跟踪逻辑匹配。如果 TRGSEL 已设置，比较器 A 或 B 的匹配信号必须通过操作码跟踪逻辑传播，如果匹配地址的操作码实际已执行，则只有触发事件 发送到 FIFO 逻辑。 0 存取比较地址时触发（强制） 1 如果比较地址的操作码已执行（标签），则触发
6 BEGIN	**开始/结束触发选择**——控制 FIFO 在触发时开始填充还是以循环形式填充直到触发结束信息的捕获。在纯事件触发模式中，忽略这个位，所有调试运用都假定为起始跟踪。 0 数据存储在 FIFO，直到触发（结束跟踪） 1 触发启动数据存储（起始跟踪）
3:0 TRG[3:0]	**选择触发模式**——选择下面 9 个触发模式中的一个。 0000 只有 A 0001 A 或 B 0010 A 然后 B

	0011 只有事件 B（存储数据） 0100 A 然后只有事件 B（存储数据） 0101 A 和 B 数据（满模式） 0110 A 和非 B 数据（满模式） 0111 范围内：A≤地址≤B 1000 范围外：地址<A 或地址>B 1001-1111 （无触发）

17.4.3.9 调试状态寄存器（DBGS）

该寄存器为只读寄存器。

	7	6	5	4	3	2	1	0
读	AF	BF	ARMF	0	CNT3	CNT2	CNT1	CNT0
写								
复位	0	0	0	0	0	0	0	0

图17-9. 调试状态寄存器（DBGS）

表17-6. DBGS寄存器域描述

域	描述
7 AF	**触发匹配 A 标记**——在调试运行开始时请除 AF，指示打开控制后是否满足触发匹配 A 条件。 0 比较器 A 未匹配 1 比较器 A 匹配
6 BF	**触发匹配 B 标记**——在调试运行开始时请除 BF，指示打开控制后是否满足触发匹配 B 条件。 0 比较器 B 未匹配 1 比较器 B 匹配
5 ARMF	**打开标记**——当 DBGEN=1 时，这个位为 DBGC 中 ARM 的只读镜像。将 DBGC 中的 ARM 控制位写为 1（当 DBGEN=1）可设置该位，在调试运行结束时自动清除它。当 FIFO 为满时（始起跟踪），或当探测到触发事件时（结束跟踪），调度运行完成。将 DBGC 中的 ARM 或 DBGEN 写为 0，可以人工停止调试运行。 0 调试器没有打开 1 调试器被打开
3:0 CNT[3:0]	**FIFO 有效计数**——这些位在调试运行开始时清除，指示调试运行结束时 FIFO 中的有效数据的字数。当数据从 FIFO 中读出时，CNT 中的值不减少。当信息从 FIFO 中读出时，外部调试主机负责计数的跟踪。 0000 FIFO 中的有效字数=无有效数据 0001 FIFO 中的有效字数=1 0010 FIFO 中的有效字数=2 0011 FIFO 中的有效字数=3 0100 FIFO 中的有效字数=4