主从接口（MSI）模块

目录

本章包括下列主题：

注： 本系列参考手册章节旨在用作对器件数据手册的补充。本文档适用于所有dsPIC33/PIC24器件。

请参见最新器件数据手册中**“主/从接口（MSI）”**章节开头的注释，以确定本文档是否支持您所使用的器件。

器件数据手册和系列参考手册的相关章节可从Microchip网站下载：
http://www.microchip.com。

1.0 简介

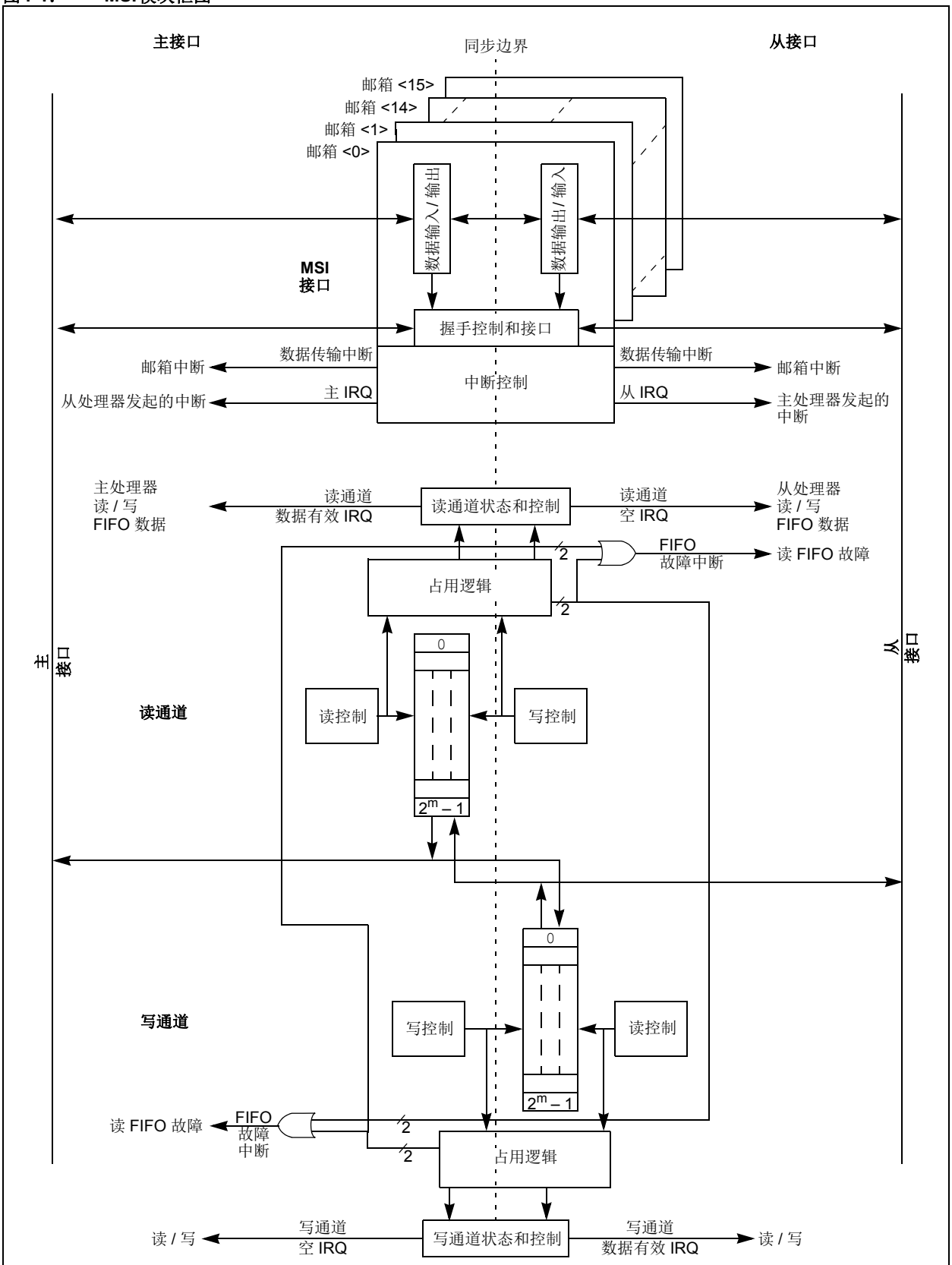
主从接口（Master Slave Interface，MSI）模块用于桥接主、从处理器系统，每个处理器系统在独立时钟域内工作。

主、从处理器系统均有自己的寄存器，用于在MSI模块之间通信；主MSI寄存器位于主SFR空间，从MSI寄存器位于从SFR空间。

主从接口（MSI）包括以下特性：

- 16个单向数据邮箱寄存器：
  - 每个邮箱寄存器的方向可通过熔丝选择
  - 可按字节寻址和字寻址
- 8个邮箱数据流控制协议块：
  - 单独熔丝使能
  - 写端口有效；读端口无效（即，无需读数据请求）
  - MSI时钟边界处采用中断驱动（或轮询）的自动数据流控制机制
  - 熔丝可分配给任一邮箱寄存器；支持任何长度的数据缓冲区（最大为可用邮箱寄存器的数量）
  - 兼容DMA传输
- 通过应答数据流控制来响应主处理器发送到从处理器以及从处理器发送到主处理器的中断请求
- 可选2通道FIFO存储器结构
- FIFO的深度为16-128字（请参见数据手册确认实际实现的FIFO深度）：
  - 1个读通道和1个写通道
  - 通过空和满状态及中断来实现循环操作
  - 通过主/从中断来实现上溢/下溢检测
  - 基于中断、软件轮询或兼容DMA传输
- 主/从处理器跨边界控制和状态：
  - 两个处理器均具有可读工作模式状态
  - 从处理器通过主处理器使能（需要满足硬件写互锁序列）
  - 从处理器在代码执行过程中复位时，主处理器将中断
  - 主处理器在代码执行过程中复位时，从处理器将中断
- 可选择通过熔丝解除主/从复位的关系，POR/BOR/$\overline{\text{MCLR}}$始终会复位主和从处理器；其余运行时复位对从使能的影响可通过熔丝设定

图1-1： MSI模块框图



注： 要确定FIFO的深度，请参见具体器件数据手册。

2.0 主/从处理器配置寄存器

2.1 MSI主处理器配置寄存器

以下寄存器与MSI主处理器模块相关联，位于主SFR空间：

- 寄存器2-1：MSI1CON
- 寄存器2-2：MSI1STAT
- 寄存器2-3：MSI1KEY
- 寄存器2-4：MSI1MBXS
- 寄存器2-5：MSI1MBXnD
- 寄存器2-6：MSI1FIFOCS
- 寄存器2-7：MRSWFDATA
- 寄存器2-8：MWSRFDATA

2.1.1 寄存器映射

表2-1简要汇总了与MSI主处理器模块相关的寄存器。此汇总后面是相应寄存器及其详细说明。

表2-1： MSI主处理器寄存器映射

寄存器名称	位范围	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
MSI1CON	15:0	SLVEN	—	—	—	RFITSEL1	RFITSEL0	MTSIRQ	STMIACK	SRSTIE	r	r	r	r	r	r	r
MSI1STAT	15:0	SLVRST	SLVWDRST	SLVPWR1	SLVPWR0	VERFERR	SLVP2ACT	STMIRQ	MTSIACK	SLVDBG	—	—	—	—	—	—	—
MSI1KEY	15:0	—	—	—	—	—	—	—	—	MSI1KEY<7:0>							
MSI1MBXS	15:0	—	—	—	—	—	—	—	—	DTRDY<H:A>							
MSI1MBXnD	15:0	MSI1MBXnD<15:0>															
MSI1FIFOCS	15:0	WFEN	—	—	—	WFOF	WFUF	WFFULL	WFEMPTY	RFEN	—	—	—	RFOF	RFUF	RFFULL	RFEMPTY
MRSWFDATA	15:0	MRSWFDATA<15:0>															
MWSRFDATA	15:0	MWSRFDATA<15:0>															

图注： — = 未实现，读为0； r = 保留位。

寄存器2-1：　　MSI1CON：MSI1主处理器控制寄存器

R/W-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
SLVEN	—	—	—	RFITSEL1	RFITSEL0	MTSIRQ	STMIACK
bit 15							bit 8

R/W-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
SRSTIE	—	—	—	—	—	—	—
bit 7							bit 0

图注：	r = 保留位		
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15　**SLVEN**：从处理器使能位
1 = 使能从处理器，释放从复位并且允许执行
0 = 禁止从处理器并保持复位

bit 14-12　**未实现**：读为0

bit 11-10　**RFITSEL<1:0>**：读FIFO中断阈值选择位
11 = 当FIFO在从处理器写操作后已满时触发数据有效中断
10 = 当FIFO在从处理器写操作后75%满时触发数据有效中断
01 = 当FIFO在从处理器写操作后50%满时触发数据有效中断
00 = 当从处理器向FIFO写入第一个数据时触发数据有效中断

bit 9　**MTSIRQ**：主处理器发送到从处理器的中断请求位
1 = 主处理器向从处理器发出中断请求
0 = 主处理器未向从处理器发出中断请求

bit 8　**STMIACK**：中断应答位（用于应答从处理器中断）
1 = 如果STMIRQ = 1：主处理器应答从处理器中断请求，否则产生协议错误
0 = 如果STMIRQ = 1：主处理器尚未应答从处理器中断请求，否则从处理器发送到主处理器的中断请求均不会处于等待处理状态

bit 7　**SRSTIE**：从处理器复位事件中断允许位
1 = 当从处理器进入复位状态时向主处理器产生从处理器复位事件中断
0 = 当从处理器进入复位状态时不向主处理器产生从处理器复位事件中断

bit 6-0　**保留**：保持为0

寄存器2-2：　　MSI1STAT：MSI1主处理器状态寄存器

R-0	R/HS/C-0	R-0	R-0	R/HS/C-0	R-0	R-0	R-0
SLVRST	SLVWDRST(1)	SLVPWR1	SLVPWR0	VERFERR	SLVP2ACT	STMIRQ	MTSIACK
bit 15							bit 8

R-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
SLVDBG	—	—	—	—	—	—	—
bit 7							bit 0

图注：	HS = 硬件置1位	C = 可清零位	
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15　**SLVRST**：从处理器复位状态位
1 = 从处理器处于复位状态
0 = 从处理器未处于复位状态

bit 14　**SLVWDRST**：从处理器看门狗定时器（Watchdog Timer，WDT）复位状态位(1)
1 = 从处理器已通过从处理器WDT复位
0 = 从处理器未通过WDT复位

bit 13-12　**SLVPWR<1:0>**：从处理器低功耗工作模式状态位
11 = 从处理器处于深度休眠模式
10 = 从处理器处于休眠模式
01 = 从处理器处于空闲模式
00 = 从处理器未处于低功耗模式

bit 11　**VERFERR**：PRAM验证错误状态位
1 = 执行VFSLV（PRAM写验证）指令期间检测到错误
0 = 执行VFSLV（PRAM写验证）指令期间未检测到错误

bit 10　**SLVP2ACT**：从处理器PRAM分区2有效状态位
该位用于反映从处理器NVM控制器状态位P2ACTIV（NVMCON<10>），后者在成功执行BOOTSWP指令（从处理器PRAM在线更新操作期间）后翻转。
1 = 从处理器NVM控制器状态位P2ACTIV = 1
0 = 从处理器NVM控制器状态位P2ACTIV = 0

bit 9　**STMIRQ**：从处理器发送到主处理器的中断请求的状态位
1 = 从处理器向主处理器发出中断请求
0 = 从处理器未向主处理器发出中断请求

bit 8　**MTSIACK**：中断应答状态位（从处理器进行应答）
1 = 如果MTSIRQ = 1：从处理器应答主处理器中断请求，否则产生协议错误
0 = 如果MTSIRQ = 1：从处理器尚未应答主处理器中断请求，否则主处理器发送到从处理器的中断请求均不会处于等待处理状态

bit 8　**SLVDBG**：从处理器调试模式状态位
1 = 从处理器工作在调试模式下
0 = 从处理器工作在任务或应用模式下

bit 6-0　**未实现**：读为0

注 1： 该位由硬件置1，由软件或POR/BOR复位清零。如果禁止从处理器（SLVEN（MSI1CON<15>）= 0），则该位将不受影响。

寄存器2-3：　　MSI1KEY：MSI1主处理器互锁密钥寄存器(1)

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
MSI1KEY<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15-8　**未实现：**读为0

bit 7-0　**MSI1KEY<7:0>：**MSI1密钥位
MSI1KEY<7:0>因特定写入值而受到监视。

注 1： 这不是物理寄存器；该寄存器将始终读为00h。

寄存器2-4：　　MSI1MBXS：MSI1主处理器邮箱数据传输状态寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DTRDY<H:A>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15-8　**未实现：**读为0

bit 7-0　**DTRDY<H:A>：**数据就绪状态位

1 = 数据发送器已指示MSI1MBXnD中有数据可供数据接收器读取（当数据发送器处理器写入分配的MSI1MBXnD时，DTRDYx自动置1）。意味着配置为：
——发送器时：已写入数据。等待接收器读取。
——接收器时：有新数据可供读取。

0 = MSI1MBXnD中没有数据可供接收器读取（或握手协议逻辑块被禁止）

寄存器2-5：　　MSI1MBXnD：MSI1主处理器邮箱n数据寄存器（主处理器，n = 0至15）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
MSI1MBXnD<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
MSI1MBXnD<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15-0　**MSI1MBXnD<15:0>：**MSI1主处理器邮箱n数据位

当已设定配置位FMBXMx = 1时：
邮箱数据方向：主处理器读取，从处理器写入主处理器；MSI1MBXnD<15:0>变为R-0（主处理器写入MSI1MBXnD<15:0>将不起作用）。

当已设定配置位FMBXMx = 0时：
邮箱数据方向：主处理器写入，从处理器读取主处理器；MSI1MBXnD<15:0>变为R/W-0。

寄存器2-6：　MSI1FIFOCS：MSI1主处理器FIFO控制/状态寄存器1

R/W-0	U-0	U-0	U-0	R/C-0	R-0	R-0	R-1
WFEN(1)	—	—	—	WFOF(2)	WFUF(2)	WFFULL(2)	WFEMPTY(2)
bit 15							bit 8

R/W-0	U-0	U-0	U-0	R-0	R/C-0	R-0	R-1
RFEN	—	—	—	RFOF	RFUF	RFFULL	RFEMPTY
bit 7							bit 0

图注：	C = 可清零位		
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15　**WFEN：**写FIFO使能位(1)
1 = 使能（主处理器）写FIFO
0 = 禁止并初始化（主处理器）写FIFO

bit 14-12　**未实现：**读为0

bit 11　**WFOF：**写FIFO上溢位(2)
1 = 检测到写FIFO上溢
0 = 未检测到写FIFO上溢

bit 10　**WFUF：**写FIFO下溢位(2)
1 = 检测到写FIFO下溢
0 = 未检测到写FIFO下溢

bit 9　**WFFULL：**写FIFO满状态位(2)
1 = 写FIFO已满；主处理器对写FIFO（WFDATA）执行的最后一次写操作写入的是最后一个空闲存储单元
0 = 写FIFO未满

bit 8　**WFEMPTY：**写FIFO空状态位(2)
1 = 写FIFO为空；从处理器对写FIFO（WFDATA）执行的最后一次读操作清空了FIFO的全部有效数据或FIFO被禁止（并初始化为空状态）
0 = 写FIFO包含尚未被从处理器读取的有效数据

bit 7　**RFEN：**读FIFO使能位
1 = 使能（主处理器）读FIFO
0 = 禁止并初始化（主处理器）读FIFO

bit 6-4　**未实现：**读为0

bit 3　**RFOF：**读FIFO上溢位
1 = 检测到读FIFO上溢
0 = 未检测到读FIFO上溢

bit 2　**RFUF：**读FIFO下溢位
1 = 检测到读FIFO下溢
0 = 未检测到读FIFO下溢

bit 1　**RFFULL：**读FIFO满状态位
1 = 读FIFO已满；从处理器对读FIFO（RFDATA）执行的最后一次写操作写入的是最后一个空闲存储单元
0 = 读FIFO未满

bit 0　**RFEMPTY：**读FIFO空状态位
1 = 读FIFO为空；主处理器对读FIFO（RFDATA）执行的最后一次读操作清空了FIFO的全部有效数据或FIFO被禁止（并初始化为空状态）
0 = 读FIFO包含尚未被主处理器读取的有效数据

注 1： 清零WFEN还会导致WFEMPTY状态位置1。随后将WFEN置1后，WFEMPTY将保持置1状态，直到主处理器将数据写入写FIFO。
2： 置1后，可通过设置WFEN = 0将这些位清零。

寄存器2-7：　　MRSWFDATA：主处理器读（从处理器写）FIFO数据寄存器

R/W-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
MRSWFDATA<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
MRSWFDATA<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15-0　　**MRSWFDATA<15:0>：**读FIFO数据输出寄存器位

寄存器2-8：　　MWSRFDATA：主处理器写（从处理器读）FIFO数据寄存器

R/W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
MWSRFDATA<15:8>							
bit 15							bit 8

W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
MWSRFDATA<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15-0　　**MWSRFDATA<15:0>：**写FIFO数据输入寄存器位

2.2 MSI从处理器配置寄存器

以下寄存器与从处理器MSI模块相关联，位于从SFR空间：

- 寄存器2-9：SI1CON
- 寄存器2-10：SI1STAT
- 寄存器2-11：SI1MBXS
- 寄存器2-12：SI1MBXnD
- 寄存器2-13：SI1FIFOCS
- 寄存器2-14：SWMRFDATA
- 寄存器2-15：SRMWFDATA

2.2.1 寄存器映射

表2-1简要汇总了与MSI从处理器模块相关的寄存器。此汇总后面是相应寄存器及其详细说明。

表2-2： MSI从处理器寄存器映射

寄存器名称	位范围	Bit 15	Bit 14	Bit 13	Bit 12	Bit 11	Bit 10	Bit 9	Bit 8	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
SI1CON	15:0	—	—	—	—	RFITSEL1	RFITSEL0	STMIRQ	MTSIACK	MRSTIE	r	r	r	r	r	r	r
SI1STAT	15:0	MSTRST	—	MSTPWR1	MSTPWR0	—	—	MTSIRQ	STMIACK	—	—	—	—	—	—	—	—
SI1MBXS	15:0	—	—	—	—	—	—	—	—	DTRDY<H:A>							
SI1MBXnD	15:0	SI1MBXnD<15:0>															
SI1FIFOCS	15:0	SRFEN	—	—	—	SRFOF	SRFUF	SRFFULL	SRFEMPTY	SWFEN	—	—	—	SWFOF	SWFUF	SWFFULL	SWFEMPTY
SWMRFDATA	15:0	SWMRFDATA<15:0>															
SRMWFDATA	15:0	SRMWFDATA<15:0>															

图注： — = 未实现，读为0； r = 保留位。

寄存器2-9：　　SI1CON：MSI1从处理器控制寄存器

U-0	U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	—	RFITSEL1	RFITSEL0	STMIRQ	MTSIACK
bit 15							bit 8

R/W-0	r-0	r-0	r-0	r-0	r-0	r-0	r-0
MRSTIE	—	—	—	—	—	—	—
bit 7							bit 0

图注：	r = 保留位		
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15-12　**未实现：**读为0

bit 11-10　**RFITSEL<1:0>：**读FIFO中断阈值选择位

11 = 当FIFO在主处理器写操作后已满时触发数据有效中断
10 = 当FIFO在主处理器写操作后75%满时触发数据有效中断
01 = 当FIFO在主处理器写操作后50%满时触发数据有效中断
00 = 当主处理器向FIFO写入第一个数据时触发数据有效中断

bit 9　**STMIRQ：**从处理器发送到主处理器的中断请求位

1 = 从处理器向主处理器发出中断请求
0 = 从处理器未向主处理器发出中断请求

bit 8　**MTSIACK：**主处理器到从处理器的中断应答位

1 = 如果MTSIRQ = 1：从处理器应答主处理器中断请求，否则产生协议错误
0 = 如果MTSIRQ = 1：从处理器尚未应答主处理器中断请求，否则从处理器发送到主处理器的中断请求均不会处于等待处理状态

bit 7　**MRSTIE：**主处理器复位事件中断允许位

1 = 当主处理器进入复位状态时向从处理器产生主处理器复位事件中断
0 = 当主处理器进入复位状态时不向从处理器产生主处理器复位事件中断

bit 6-0　**保留：**保持为0

寄存器2-10： SI1STAT：MSI1从处理器状态寄存器

R-0	U-0	R-0	R-0	U-0	U-0	R-0	R-0
MSTRST	—	MSTPWR1	MSTPWR0	—	—	MTSIRQ	STMIACK
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15 **MSTRST：**主处理器复位状态位
1 = 主处理器处于复位状态
0 = 主处理器未处于复位状态

bit 14 **未实现：**读为0

bit 13-12 **MSTPWR<1:0>：**主处理器低功耗工作模式状态位
11 = 保留
10 = 主处理器处于休眠模式
01 = 主处理器处于空闲模式
00 = 主处理器未处于低功耗模式

bit 11-10 **未实现：**读为0

bit 9 **MTSIRQ：**主处理器中断从处理器位
1 = 主处理器向从处理器发出中断请求
0 = 主处理器未向从处理器发出中断请求

bit 8 **STMIACK：**主处理器应答状态位
1 = 如果STMIRQ = 1：主处理器应答从处理器中断请求，否则产生协议错误
0 = 如果STMIRQ = 1：主处理器尚未应答从处理器中断请求，否则从处理器发送到主处理器的中断请求均不会处于等待处理状态

bit 7-0 **未实现：**读为0

寄存器2-11： SI1MBXS：MSI1从处理器邮箱数据传输状态寄存器

U-0	U-0	U-0	U-0	U-0	U-0	U-0	U-0
—	—	—	—	—	—	—	—
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
DTRDY<H:A>							
bit 7							bit 0

图注：	HS = 硬件置1位		
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15-8 **未实现：**读为0

bit 7-0 **DTRDY<H:A>：**数据就绪状态位

1 = 数据发送器已指示MSI1MBXnD中有数据可供数据接收器读取（当数据发送器处理器写入分配的MSI1MBXnD时，DTRDYx自动置1）。意味着配置为：
——发送器时：写入数据。等待接收器读取。
——接收器时：已有新数据可供读取。

0 = MSI1MBXnD中没有数据可供接收器读取（或握手协议逻辑块在配置位中被禁止）

寄存器2-12： SI1MBXnD：MSI1从处理器邮箱n数据寄存器（从处理器，n = 0至15）

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SI1MBXnD<15:8>							
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SI1MBXnD<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15-0 **SI1MBXnD<15:0>：**MSI1从处理器邮箱n数据位

当已设定配置位FMBXMx = 1时：
邮箱数据方向：主处理器读取，从处理器写入主处理器；SI1MBXnD<15:0>变为R-0（主处理器写入SI1MBXnD<15:0>将不起作用）。

当已设定配置位FMBXMx = 0时：
邮箱数据方向：主处理器写入，从处理器读取主处理器；SI1MBXnD<15:0>变为R/W-0。

寄存器2-13：　SI1FIFOCS：MSI1从处理器FIFO状态寄存器

R-0	U-0	U-0	U-0	R/C-0	R/C-0	R-0	R-1
SRFEN(1,2)	—	—	—	SRFOF(3)	SRFUF	SRFFULL(4)	SRFEMPTY
bit 15							bit 8

R-0	U-0	U-0	U-0	R-0	R/C-0	R-0	R-1
SWFEN	—	—	—	SWFOF	SWFUF	SWFFULL	SWFEMPTY
bit 7							bit 0

图注：	C = 可清零位		
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15　**SRFEN：**从处理器读（主处理器写）FIFO使能位(1,2)
1 = 使能从处理器读FIFO（主处理器写）
0 = 禁止从处理器读FIFO（主处理器写）

bit 14-12　**未实现：**读为0

bit 11　**SRFOF：**从处理器读（主处理器写）FIFO上溢位(3)
1 = 检测到从处理器读FIFO上溢
0 = 未检测到从处理器读FIFO上溢

bit 10　**SRFUF：**从处理器读（主处理器写）FIFO下溢位
1 = 检测到从处理器读（主处理器写）FIFO下溢
1 = 未检测到从处理器读（主处理器写）FIFO下溢

bit 9　**SRFFULL：**从处理器读（主处理器写）FIFO满状态位(4)
1 = 从处理器读（主处理器写）FIFO已满；主处理器对从处理器读FIFO（SRMWFDATA）执行的最后一次写操作写入的是最后一个空闲存储单元
0 = 从处理器读（主处理器写）FIFO未满

bit 8　**SRFEMPTY：**从处理器读（主处理器写）FIFO空状态位
1 = 从处理器读（主处理器写）FIFO为空；从处理器对读FIFO（SRMWFDATA）执行的最后一次读操作清空了FIFO的全部有效数据或FIFO被禁止（并初始化为空状态）
0 = 从处理器读（主处理器写）FIFO包含尚未被从处理器读取的有效数据

bit 7　**SWFEN：**从处理器写（主处理器读）FIFO使能位
1 = 使能从处理器写（主处理器读）FIFO
0 = 禁止从处理器写（主处理器读）FIFO

bit 6-4　**未实现：**读为0

bit 3　**SWFOF：**从处理器写（主处理器读）FIFO上溢位
1 = 检测到从处理器写（主处理器读）FIFO上溢
0 = 未检测到从处理器写（主处理器读）FIFO上溢

bit 2　**SWFUF：**从处理器写（主处理器读）FIFO下溢位
1 = 检测到从处理器写（主处理器读）FIFO下溢
0 = 未检测到从处理器写（主处理器读）FIFO下溢

注 1： SRFEN是一个只读位，在主处理器使能其写FIFO（WFEN（MSI1FIFOCS<15>）= 1）时置1。只有当主处理器清零WFEN位时，该位才会清零。
2： 当主处理器设置RFEN（MSIFIFOCS<7>）= 1时，SRFEN位置1。
3： 当缓冲区已满且主处理器额外再发送一个数据但从处理器未读取SRMWFDATA寄存器时，上溢位置1。
4： 当从处理器读缓冲区已满时，SRFFULL位置1。它可以在从处理器读取缓冲区时清零（使用SRMWFDATA寄存器）。

寄存器2-13：　SI1FIFOCS：MSI1从处理器FIFO状态寄存器（续）

bit 1　**SWFFULL：**从处理器写（主处理器读）FIFO满状态位
1 = 从处理器写（主处理器读）FIFO已满；从处理器对FIFO（SWMRFDATA）执行的最后一次写操作写入的是最后一个空闲存储单元
0 = 从处理器写（主处理器读）FIFO未满

bit 0　**SWFEMPTY：**从处理器写（主处理器读）FIFO空状态位
1 = 从处理器写（主处理器读）FIFO为空；主处理器对读FIFO执行的最后一次读操作清空了FIFO的全部有效数据或FIFO被禁止（并初始化为空状态）
0 = 从处理器写（主处理器读）FIFO包含尚未被主处理器读取的有效数据

注 1： SRFEN是一个只读位，在主处理器使能其写FIFO（WFEN（MSI1FIFOCS<15>）= 1）时置1。只有当主处理器清零WFEN位时，该位才会清零。
2： 当主处理器设置RFEN（MSIFIFOCS<7>）= 1时，SRFEN位置1。
3： 当缓冲区已满且主处理器额外再发送一个数据但从处理器未读取SRMWFDATA寄存器时，上溢位置1。
4： 当从处理器读缓冲区已满时，SRFFULL位置1。它可以在从处理器读取缓冲区时清零（使用SRMWFDATA寄存器）。

寄存器2-14： SWMRFDATA：从处理器写（主处理器读）FIFO数据寄存器

W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
SWMRFDATA<15:8>							
bit 15							bit 8

W-0	W-0	W-0	W-0	W-0	W-0	W-0	W-0
SWMRFDATA<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15-0 **SWMRFDATA<15:0>：**写FIFO数据输出寄存器位

寄存器2-15： SRMWFDATA：从处理器读（主处理器写）FIFO数据寄存器

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SRMWFDATA<15:8>							
bit 15							bit 8

R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
SRMWFDATA<7:0>							
bit 7							bit 0

图注：			
R = 可读位	W = 可写位	U = 未实现位，读为0	
-n = POR时的值	1 = 置1	0 = 清零	x = 未知

bit 15-0 **SRMWFDATA<15:0>：**写FIFO数据输出寄存器位

3.0 概述

主从接口（MSI）宏是主处理器和从处理器之间的数据网关，其主要用途是控制从处理器以及在两个处理器之间移动数据。主处理器和从处理器将以明显不同的时钟速度运行，因此MSI模块还包括用于同步两个时钟域之间的数据和信号的功能。宏由16个独立的单向邮箱式数据管道组成。数据方向逻辑分配可通过熔丝选择。

有时数据写处理器为主处理器，数据读处理器为从处理器；有时数据写处理器为从处理器，数据读处理器为主处理器；具体取决于邮箱方向。

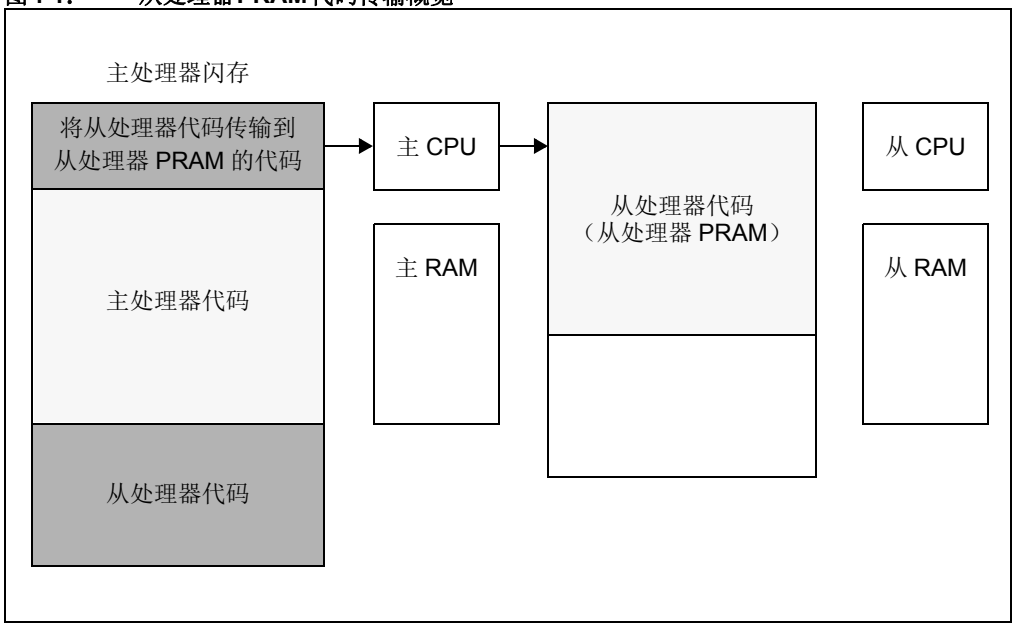
4.0 从处理器控制

MSI的MSI1CON寄存器中包含三个与从处理器控制相关的控制位。

4.1 从处理器使能（SLVEN）控制

当器件首次上电时，从处理器代码位于主处理器闪存中。首次上电后，主处理器中的用户代码会将数据从主处理器传输到从处理器。

图4-1： 从处理器PRAM代码传输概览



当此传输过程发生时，从处理器必须保持复位状态；通过MSI模块实现此目的。SLVEN（MSI1CON<15>）控制位为主处理器提供使能或禁止从处理器的方法。

从处理器在SLVEN = 0时禁止。在这种状态下：

- 从处理器保持复位状态
- 主处理器可访问从处理器PRAM（以在器件退出复位时向其装入数据）
- 从处理器复位状态位SLVRST（MSI1STAT<15>）= 1（主处理器寄存器）

从处理器在SLVEN = 1时使能。在这种状态下：

- 从处理器复位状态释放，它将开始在所配置的工作模式下执行代码
- 主处理器不能访问从处理器PRAM（在双分区模式下，主处理器将可访问非活动PRAM）
- 从处理器复位状态位SLVRST（MSI1STAT<15>）= 0

SLVRST位的状态指示从处理器何时处于复位状态。只有当从处理器从先前的未复位状态进入复位状态时，才会产生相关中断。即，只有从处理器先使能后，才能产生中断。

注： 只有在满足硬件写互锁后，才能修改SLVEN位，如例4-1所述。

通过基于MSI1KEY寄存器的软件解锁序列保护SLVEN位免受意外写入的影响。鉴于MSI控制接口的关键性质，MSI宏解锁机制独立于闪存控制器的相应机制，从而提高稳健性。

向MSI1KEY寄存器写入预定义的数据序列后，将打开一个窗口。SLVEN位应在解锁序列后的第一条指令写入。MSI1CON寄存器中的其他位均不受互锁的影响。MSI1KEY寄存器不是物理寄存器。读MSI1KEY寄存器将读取全0。

当使能SLVEN位锁定时（即，相应位锁定，无法修改），必须执行例4-1所示的指令序列以解除锁定。解锁序列是将目标控制位置1和清零的先决条件。

例4-1： MSI使能操作

```
//Unlock Key to allow MSI Enable control

MOV.b    #0x55, W0
MOV.b    WREG, MSI1KEY
MOV.b    #0xAA, W0
MOV.b    WREG, MSI1KEY
// Enable MSI

BSET     MSI1CON, SLVEN
```

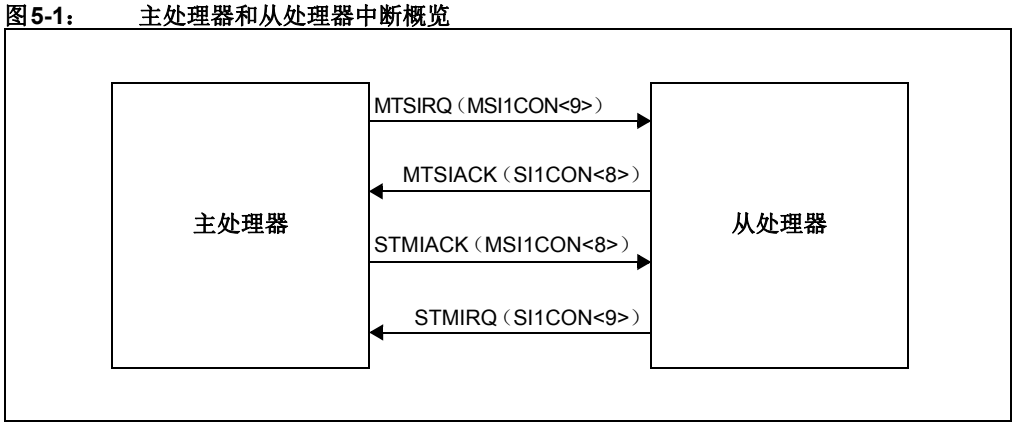
例4-2： 用C代码实现的MSI使能操作

```
#include <libpic30.h>
    _start_slave();
```

5.0 处理器间的中断请求和应答

主处理器和从处理器可以直接相互中断。主处理器可通过将MTSIRQ（MSI1CON<9>）控制位置为有效向从处理器发出中断请求。相似地，从处理器可通过将STMIRQ（SI1CON<9>）控制位置为有效向主处理器发出中断请求。对于主处理器发送到从处理器的中断请求，使用中断应答位STMIACK（MSI1CON<8>）来应答中断；对于从处理器发送到主处理器的中断请求，使用MTSIACK（SI1CON<8>）来应答中断（图5-1）。

图5-1： 主处理器和从处理器中断概览



6.0 传输模式

基于主处理器和从处理器之间的数据传输模式，传输可分为以下两种主要类型：

1. 基于邮箱的传输。
2. 基于FIFO的传输。

7.0 邮箱传输模式

邮箱由16个独立的单向数据寄存器组成。最多可从这些寄存器中选择8个与支持硬件就绪/应答协议的独立数据流控制逻辑配合工作，从而形成邮箱式数据管道。寄存器MSI1MBXS中有8个基于协议的通信状态位。共有8个硬件协议，依次命名为A至H。

注：	数据方向和数据流控制逻辑分配通过FMBXM<15:0>配置位选择，在执行代码前分配。

由于每个数据寄存器的方向可编程，在讨论数据传输协议时使用主处理器或从处理器术语没有意义。因此，之后使用术语“发送器”和“接收器”分别表示数据写处理器和读处理器。有时数据写处理器为主处理器，数据读处理器为从处理器；有时数据写处理器为从处理器，数据读处理器为主处理器；具体取决于数据寄存器方向。

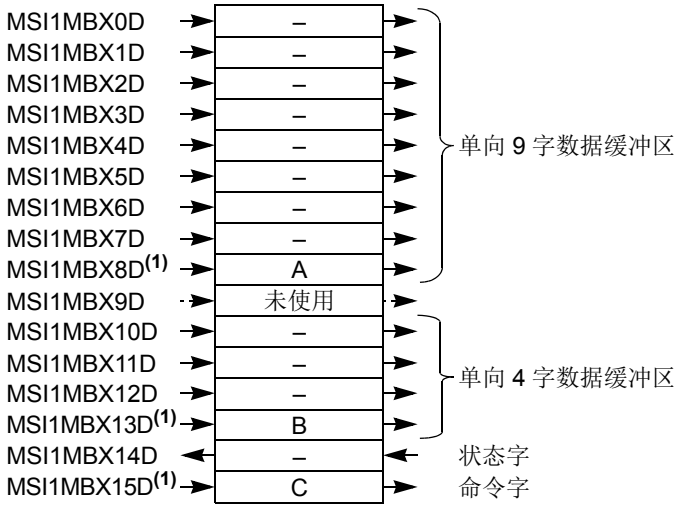
7.1 邮箱数据管道

对邮箱内数据寄存器的访问使用数据流控制协议来控制。因此，对基于邮箱的数据管道的访问具有互斥性（即，两个处理器不能同时对其进行访问）。每个处理器必须先完成其访问再将访问控制权交给另一个处理器。例如，如果主处理器已将MSI1MBX0D配置为发送器，则从处理器将必须等待中断（命令）才能接收SI1MBX0D寄存器的内容。

此外，对于由多个数据寄存器组成的邮箱，邮箱内所有数据寄存器的方向不必相同。数据流控制协议用于在主处理器和从处理器之间传输访问控制，但只有分配给协议硬件的数据寄存器才必须符合所需的数据方向规则。

图7-1所示为支持2个单向缓冲区、1个命令字和1个状态字的布局。数据流控制逻辑模块将分配给每个缓冲区中最后访问的字（请注意，在所支持邮箱数的限制范围内，缓冲区长度是任意的）。此外，数据流控制逻辑模块还会分配给命令字。但是，对状态字的访问是通过软件控制的，因此不需要数据流控制逻辑模块。MSI1MBX9D邮箱未使用。例如，读/写为协议分配的寄存器后，才能产生特定协议的中断。如图7-1所示，写入所有寄存器（MSI1MBX0D至MSIx7D）后，直到主处理器写入MSI1MBX8D寄存器（协议A寄存器），从处理器才会获得协议A中断。

图7-1： 邮箱结构示例



注 1： 应使用熔丝FMBXHS1将MSI1MBX8D分配给协议A。应使用FMBXHSx配置熔丝将MSI1MBX13D和MSI1MBX15D分配给其对应的协议。

注： 未经握手不得使用邮箱。应当注意，当发送内核正在发送时，不得读取接收内核。

7.2 邮箱数据寄存器

16个MSI邮箱数据寄存器MSI1MBXnD/SI1MBXnD（其中，0 ≤ n ≤ 15）相同，只是其数据方向可能不同。MSI宏包含8个数据流控制协议硬件模块，每个模块都可分配给任一数据寄存器以形成邮箱。这些邮箱的状态在DTRDYx位（MSI1MBXS<7:0>）中更新，其中x可以是A-H，表示8个数据流协议。

7.3 邮箱寄存器可访问性

所有MSI1MBXnD邮箱数据寄存器都是单向的，因此寄存器的内容始终不会被主端口和从端口读/写。每个MSI1MBXnD寄存器可通过主处理器（作为发送器）读/写以及通过从处理器（作为接收器）读取（只读），或者通过从处理器（作为发送器）读/写以及通过主处理器（作为接收器）读取（只读），具体取决于所选通道数据方向。这通过MBXM<15:0>配置位FMBXM<15:0>实现：

FMBXM MBXMn：邮箱数据寄存器n通道方向熔丝位（n = 0至15）

1 = 邮箱寄存器n配置为进行主处理器数据读取（从处理器到主处理器方向的数据传输——从处理器作为发送器）

0 = 邮箱寄存器n配置为进行主处理器数据写入（主处理器到从处理器方向的数据传输——主处理器作为发送器）

7.3.1 数据握手

注： 下文假定提及的数据寄存器（MSI1MBXnD/SI1MBXnD）采用正确的数据方向来支持所述操作。

支持使用自动数据流控制机制通过邮箱来控制数据流。8个数据流握手协议硬件模块中的每一个均用于控制位于MSI1MBXS和SI1MBXS寄存器中的2个数据就绪状态位（DTRDYx，其中x为A、B、C、D、E、F、G或H）。一个标志用于数据发送器，位于接口发送侧的MSI1MBXS/SI1MBXS寄存器中。另一个标志用于数据接收器，位于接口接收侧的MSI1MBXS/SI1MBXS寄存器中。

始终假定数据发送器为传输的发起方，因此不需要来自数据接收器的硬件数据请求。如果应用需要通过数据请求来发起传输，则必须通过软件进行处理。接收处理器软件将必须向发送处理器指示需要数据。这可以通过中断或基于邮箱的软件命令协议来实现。

7.3.1.1 使能握手协议硬件模块

每个握手协议硬件模块都有一个与其相关的熔丝使能功能。必须编程熔丝才能使能相应的握手协议硬件模块。FMBXHS1<3:0>配置位对应于握手协议硬件模块A，FMBXHS1<7:4>配置位对应于握手协议硬件模块B（有关熔丝详细信息，请参见器件数据手册）。

FMBXHS1 MBXHSA<3:0>：邮箱握手协议模块D寄存器分配位

1111 = MSI1MBXD15分配给邮箱握手协议模块A

…

0001 = MSI1MBXD1分配给邮箱握手协议模块A

0000 = MSI1MBXD0分配给邮箱握手协议模块A

FMBXHS1 MBXHSB<3:0>：邮箱握手协议模块B寄存器分配位

1111 = MSI1MBXD15分配给邮箱握手协议模块B

…

0001 = MSI1MBXD1分配给邮箱握手协议模块B

0000 = MSI1MBXD0分配给邮箱握手协议模块B

FMBXHSx MBXHSn<3:0>，其中n = A至H

（有关正确的等效配置设置，请参见器件数据手册）。

7.3.2 分配握手协议硬件模块

通过FMBXHSx寄存器中的8个4位位域（MBXHSn<3:0>）将8个协议模块中的每一个分配给特定的MSI邮箱数据寄存器：

1. 所选MSI邮箱寄存器称为邮箱协议数据寄存器。
2. 未分配的邮箱寄存器称为邮箱数据寄存器。

协议数据寄存器可能是单个邮箱或一组邮箱寄存器中的一个邮箱寄存器，通过软件定义为缓冲区。当邮箱定义为缓冲区时，最后一次缓冲区访问必须针对协议数据寄存器。类似地，当接收处理器检测到数据就绪且在访问邮箱时，最后一次缓冲区访问也必须针对协议数据寄存器。用户软件（主处理器和从处理器）必须确定应有多少个邮箱与每个协议数据寄存器相关。如果MSI1MBX0D和MSI1MBX1D选作与协议A相关的邮箱数据寄存器（MSI1MBX4D作为邮箱协议数据寄存器A），则发送器应确保针对邮箱数据寄存器（MSI1MBX0D和MSI1MBX1D）的操作完成后最后写入邮箱协议数据寄存器（MSI1MBX4D）。

类似地，当接收器正在接收数据时，接收器应确保针对邮箱数据寄存器（MSI1MBX0D和MSI1MBX1D）的操作完成后最后读取协议数据寄存器（MSI1MBX4D）。

7.4 使用中断实现邮箱数据传输

当没有处理器正在访问邮箱时，数据流控制硬件处于空闲状态（DTRDYx（MSI1MBXS<7:0>）= 0）。发送处理器现在可以访问邮箱以启动数据传输数据流控制。

数据流控制按下述方式工作，其中MSI1MBX0D至MSI1MBX4D寄存器分配为邮箱数据寄存器，MSI1MBX5D为邮箱协议数据寄存器（A）：

1. 发送处理器：
   a) 写入除最后一个数据字以外的所有数据字。
   b) DIN → MSI1MBX5D（最后一次数据写入）1 → DTRDYA向接收器发送读就绪中断（自动）。
2. 接收处理器：
   接收读就绪中断1
   a) 读取/发送除最后一个数据字以外的所有数据字（如果定义为缓冲区），MSI1MBX0D-MSI1MBX4D。
   b) MSI1MBX5D → DOUT（最后一次数据读取）0 → DTRDYA（自动）向发送器发送写就绪中断（自动）。
3. 发送处理器：
   返回步骤1

在图7-2中，使用FMBXM配置寄存器中的MBXMn位将MSI1MBX0D至MSI1MBX4D寄存器配置为进行主处理器到从处理器的发送操作，使用FMBXHS1<3:0> = 0101将MSI1MBX5D配置为协议A。例7-1和例7-2给出了允许在主处理器和从处理器之间进行邮箱传输的代码。

图7-2： 邮箱数据传输流

注 1： 邮箱数据寄存器可以是发送器或接收器，具体由FMBXM配置位分配。本示例将所有邮箱数据寄存器视为发送寄存器。

2： 邮箱协议A数据寄存器（由FMBXHS1<3:0> = 0101分配）。

在步骤3a之后，数据流控制完成，发送处理器可退出发送状态，也可继续发送更多数据（即，返回步骤1）。

如前文所述，通过发送器寄存器写入MSI1MBX5D寄存器时，接收器的相应邮箱将产生接收器数据流控制协议中断（读就绪），MSIAIF中断标志位置1且DTRDYA = 1。

类似地，当接收器读取MSI1MBX5D寄存器时（在读取邮箱数据寄存器之后），MSIAIF以及发送器的DTRDYA位将置1。

注： 与未使用协议硬件模块相关的中断应由用户在中断控制器中禁止。

例7-1： 使用协议A中断实现的邮箱传输（主处理器和从处理器之间的数据传输）

```
////////////////////////////////////////////////////////////////////////////////////////////
MASTER PROJECT
////////////////////////////////////////////////////////////////////////////////////////////
#include "p33CH128RA508.h"
#pragma  config MBXM0 = M2S    //(Master to Slave data transfer)
#pragma  config MBXM1 = M2S    //(Master to Slave data transfer)
#pragma  config MBXM2 = M2S    //(Master to Slave data transfer)
#pragma  config MBXM3 = M2S    //(Master to Slave data transfer)
#pragma  config MBXM4 = M2S    //(Master to Slave data transfer)
#pragma  config MBXM5 = M2S    //(Master to Slave data transfer) Protocol A assigned to mailbox 5
#pragma  config MBXM6 = S2M    //(Slave to Master data transfer)
#pragma  config MBXM7 = S2M    //(Slave to Master data transfer)
#pragma  config MBXM8 = S2M    //(Slave to Master data transfer)
#pragma  config MBXM9 = S2M    //(Slave to Master data transfer)
#pragma  config MBXM10 = S2M   //(Slave to Master data transfer)
#pragma  config MBXM11 = S2M   //(Slave to Master data transfer)
#pragma  config MBXM12 = S2M   //(Slave to Master data transfer)
#pragma  config MBXM13 = S2M   //(Slave to Master data transfer)
#pragma  config MBXM14 = S2M   //(Slave to Master data transfer)
#pragma  config MBXM15 = S2M   //(Slave to Master data transfer)

// FMBXHS1
#pragma  config MBXHSA = MBX5 //(MSIxMBXD5 assigned to mailbox handshake protocol block A)
#pragma  config MBXHSB = MBX15 //(MSIxMBXD15 assigned to mailbox handshake protocol block B)
#pragma  config MBXHSC = MBX15 //(MSIxMBXD15 assigned to mailbox handshake protocol block C)
#pragma  config MBXHSD = MBX15 //(MSIxMBXD15 assigned to mailbox handshake protocol block D)
// FMBXHS2
#pragma  config MBXHSE = MBX15 //(MSIxMBXD15 assigned to mailbox handshake protocol block E)
#pragma  config MBXHSF = MBX15 //(MSIxMBXD15 assigned to mailbox handshake protocol block F)
#pragma  config MBXHSG = MBX15 //(MSIxMBXD15 assigned to mailbox handshake protocol block G)
#pragma  config MBXHSH = MBX15 //(MSIxMBXD15 assigned to mailbox handshake protocol block H)
// FMBXHSEN
#pragma  config HSAEN = ON    //(Mailbox data flow control handshake protocol block enabled.)
#pragma  config HSBEN = OFF   //(Mailbox data flow control handshake protocol block disabled.)
#pragma  config HSCEN = OFF    //(Mailbox data flow control handshake protocol block disabled.)
#pragma  config HSDEN = OFF   //(Mailbox data flow control handshake protocol block disabled.)
#pragma  config HSEEN = OFF    //(Mailbox data flow control handshake protocol block disabled.)
#pragma  config HSFEN = OFF    //(Mailbox data flow control handshake protocol block disabled.)
#pragma  config HSGEN = OFF    //(Mailbox data flow control handshake protocol block disabled.)
#pragma  config HSHEN = OFF
```

例7-1： 使用协议A中断实现的邮箱传输（主处理器和从处理器之间的数据传输）（续）

```
unsignedint temp1,temp2,temp3,temp4,temp5,temp6,temp7,temp8,temp9,temp10,Flag;
int     main()
{
        IEC8bits.MSIAIE=1;     //enable interrupt for protocol A
        Flag=0;
        MSI1MBX0D=0xA0;
        MSI1MBX1D=0xA1;
        MSI1MBX2D=0xA2;
        MSI1MBX3D=0xA3;
        MSI1MBX4D=0xA4;
        Switch();              //waiting to start the to and fro transfer (writing to MSI1MBX5D will
                               //initiate the transfer to the slave and set MSIAIF of slave)

        MSI1MBX5D=0xA5;        //this will initiate transfer
while(1)
{
        while(Flag==0);        //Wait till the slave responds by reading MSI1MBX5D(which will generate
                               //the master MSIAIF interrupt)

           Flag=0;             //this flag is set in the MSAIF interrupt vector
        MSI1MBX0D=0xA0;
        MSI1MBX1D=0xA1;
        MSI1MBX2D=0xA2;
        MSI1MBX3D=0xA3;
        MSI1MBX4D=0xA4;
        MSI1MBX5D=0xA5;        //writing MSI1MBX5D will initiate transfer setting MSIAIF of the slave
}
/////////////////interrupt vector for protocol A/////////////////////////////////////
void __attribute__ ((interrupt, no_auto_psv)) _MSIAInterrupt(void)
{
        IFS8bits.MSIAIF=0;
        //Read the data from slave
        temp1=MSIMBX6D;
        temp2=MSIMBX7D;
        temp3=MSIMBX8D;
        temp4=MSIMBX9D;
        temp5=MSIMBX10D;
        temp6=MSIMBX11D;
        temp7=MSIMBX12D;
        temp8=MSIMBX13D;
        temp9=MSIMBX14D;
        temp10=MSIMBX15D;
        // set the flag for the next round of data transfer
        IFS8bits.MSIAIF=0;
        Flag=1;
}
```

例7-2： 使用协议A中断实现的邮箱传输（从处理器和主处理器之间的数据传输）

```
////////////////////////////////////////////////////////////////////////////////////////////////
SLAVE PROJECT (Slave side of the project)
////////////////////////////////////////////////////////////////////////////////////////////////

unsignedint temp1,temp2,temp3,temp4,temp5,temp6,temp7,temp8,temp9,temp10,Flag;
int     main(void)
{

               IEC8bits.MSIAIE=1;
               Flag=0;
    while (1)
{
        while(Flag==0);    //Wait till master initiates the transfer by writing data in MSI1MBX5D
        Flag=0;            //This flag is set in the MSIAIF interrupt
            SI1MBX6D=0x03;  //load all the data that need to be transfered to Master SI1MBX6D to SI1MBX15D
            SI1MBX7D=0x04;
            SI1MBX8D=0x05;
            SI1MBX9D=0x06;
            SI1MBX10D=0x07;
            SI1MBX11D=0x08;
            SI1MBX12D=0x08;
            SI1MBX13D=0x0A;
            SI1MBX14D=0x0B;
            SI1MBX15D=0x0C;
        temp5=SI1MBX5D;    // Reading the SI1MBX5D will generate MSIAIF interrupt for Master


    }
}

// MSAIF interrupt//////////////////////////////////////////////////////////////////////////////
void __attribute__ ((interrupt, no_auto_psv)) _MSIAInterrupt(void)
{
            temp0=MSI1MBX0D;
            temp1=MSI1MBX1D;
            temp2=MSI1MBX2D;
            temp3=MSI1MBX3D;
            temp4=MSI1MBX4D;
            // need to wait to read MSIMBX5D unless Master needs to be interrupted

            IFS8bits.MSIAIF=0;
            Flag=1;

}
```

7.5 使用软件轮询实现邮箱数据传输

尽管使用中断是管理邮箱数据流控制协议的主要方法，但也可以使用软件轮询状态位。在通过邮箱发送的数据要在周期性控制过程中使用的应用中，通过软件轮询邮箱数据流控制状态标志可能是首选方法。发送和接收处理器轮询软件应测试其相应的DTRDYx标志以确定数据流控制的状态。

发送处理器：

- DTRDYx = 1：未准备好发送数据（尚未读取邮箱）
- DTRDYx = 0：已准备好发送数据（邮箱为空）

接收处理器：

- DTRDYx = 1：数据可供读取（但尚未读取）
- DTRDYx = 0：已准备好接收数据（邮箱为空或数据过期）

7.6 邮箱数据寄存器、握手状态位和主/从处理器复位

MSI1MBXnD寄存器不受POR/BOR以外的任何器件复位的影响，因此可供接收器软件使用的数据将得以保留。这里的假设是，如果接收器读取已在进行（即，通过中断触发或DTRDYx轮询并检测到置1），最好返回有效（如果是旧数据）数据而不是复位值。

但是，所有DTRDYx流控制位（主处理器和从处理器）均受主处理器复位的影响。退出复位时，有必要初始化数据流协议模块。

当主处理器发生复位时，主处理器和从处理器侧的数据就绪状态标志位DTRDYx（MSI1MBXS<7:0>）均会复位。

当MSRE（FSLV1DEVOPT<15>）= 0时，从处理器复位与主处理器无关，MSRE熔丝 = 0（见**第10.0节“主处理器/从处理器复位交互”**），这样一来，如果主处理器（发送器）写入MSI1MBXnD寄存器后立即发生主处理器复位，则从处理器将在主处理器复位时继续运行，从处理器（接收器）侧中断请求将不会产生。

当从处理器发生复位时，主处理器和从处理器侧的数据就绪状态位（DTRDYx）均不会复位。如果从处理器（发送器）写入MSI1MBXnD寄存器后立即发生从处理器复位，则主处理器（接收器）侧中断请求仍将照常产生。

为避免可能出现的数据冲突情况，还需要通过主处理器复位将主处理器和从处理器数据就绪状态（DTRDYx）位复位。对于从处理器DTRDYx（MSI1MBXS<7:0>）标志而言，当主处理器复位和从处理器复位无关（配置熔丝MSRE = 0）且从处理器复位不会禁止从处理器（SSRE熔丝 = 0）时，如果要通过从处理器复位来复位从处理器DTRDYx标志，则可能会出现数据冲突情况。如果要通过从处理器复位来复位从处理器DTRDYx标志，则主处理器可能发生复位，从而复位主处理器侧的DTRDYx标志位，而非复位从处理器的DTRDYx标志位。这会使（仍在运行的）从处理器有机会处理从处理器DTRDYx标志位并读取相应的邮箱，上述操作可能在主处理器写入该邮箱时进行（假设由于主处理器DTRDYx = 0而使该邮箱为空）。

7.7 使用邮箱进行临时存储

只有当接收器DTRDYx = 1时，通过接收器读取MSI1MBXnD寄存器才会产生数据流控制协议中断（写就绪）。如果接收器DTRDYx = 0（首次从邮箱读取新数据后将出现这种情况），则之后通过接收器读取邮箱将不起作用（除了返回目标邮箱的数据内容外）。这允许接收器使用邮箱来临时存储通过其传送的最后一个数据值。

但是，从邮箱读取数据后，邮箱内容必须被视为过期，并且随时可能因发送器而发生变化。因此，为了成功管理邮箱临时存储的内容，假设存在这样一种软件数据传输协议：数据接收器可防止数据发送器用新数据任意改写邮箱的内容。例如，如果接收器必须向发送器请求数据（通过其他邮箱或中断），则发送器将不会改写邮箱的内容。

7.8 事务数据大小

与任何SFR一样，MSI1MBXnD寄存器可按字节访问或按字访问。为了在使用数据缓冲区时支持字节和字大小的数据事务，按最高有效字节（Most Significant Byte，MSB）或按字写入发送器协议寄存器会将相应的DTRDYx标志位置1。类似地，按MSB或按字读取接收器协议数据寄存器（MSI的另一侧）会将相应的DTRDYx标志位置1。

7.9 使用DMA控制器实现邮箱数据传输

如果DMA在器件上可用，则可使用DMA在MSI的主处理器侧或从处理器侧访问邮箱数据寄存器。邮箱数据流控制协议将产生与DMA操作兼容的中断，可在没有CPU干预的情况下传输各邮箱寄存器内的数据。

7.10 DMA数据传输序列

对于要传输的第一个DMA数据值（或块）而言，分配的发送器DMA通道可通过软件触发，也可通过用软件手动写入第一个数据值（或数据块）的方式触发。当DMA写入邮箱协议数据寄存器（在块传输过程中最后一次写入）时，相应的DTRDYx标志位将置1。将发送器DTRDYx标志位置1会在接口的接收器侧产生读就绪中断。

接收器读就绪中断（在通过发送器写入邮箱协议数据寄存器将DTRDYx置1后发起）将触发相应的接收器DMA通道并使其读取目标邮箱（数据块传输时为多个邮箱）。这样一来，会将相应的DTRDYx标志位清零。将接收器DTRDYx标志位清零会在接口的发送器侧产生发送器写就绪中断。这将触发所分配的发送器DMA通道写入下一个数据值（或数据块）并自动将DTRDYx标志位置1，从而再次启动序列。

8.0 FIFO传输模式

8.1 FIFO数据通道

MSI包含一个2通道FIFO，FIFO用于协调主处理器和从处理器之间的数据队列。假设FIFO不为空（或出现错误情况），则主处理器和从处理器可并行访问。因此，FIFO可提供比基于邮箱的数据管道更高的吞吐量，该数据管道必须由一个处理器先装入，才能供另一个处理器读取。

每个FIFO通道数据流是单向的，可简化设计和操作；一个通道是专门的读数据通道，另一个通道是专门的写数据通道。在下面的章节中，数据发送器是用于向FIFO写入数据的处理器。而数据接收器是从FIFO读取数据的处理器。

8.1.1 FIFO使能

当相应FIFO使能位（对于主处理器写FIFO为WFEN（MSI1FIFOCS<15>），对于读FIFO为RFEN（MSI1FIFOCS<7>））清零时，FIFO将被禁止。器件复位期间，FIFO使能控制位将清零。

主处理器最终负责使用配置位初始化和使能从处理器以及相关联的邮箱，因此，主处理器还负责使能（或禁止）MSI FIFO，无论数据流方向如何。

在正常工作条件下，FIFO将保持使能。但在出现FIFO错误时或者从处理器复位（或已停止响应并需要复位）时，WFEN（MSI1FIFOCS<15>）和RFEN（MSI1FIFOCS<7>）控制位可用于根据需要刷新和重新初始化FIFO。

FIFO禁止时，FIFO内容被擦除（复位为逻辑0），地址指针初始化为FIFO空状态，两个地址指针设置为彼此相等（在这种情况下为全0）。写FIFO的写FIFO空状态位（MSI1FIFOCS<8>）也会置1，读FIFO的RFEMPTY（MSI1FIFOCS<0>）置1。

使能FIFO后，空状态位将保持置1，直至第一个数据值写入FIFO。

在发生主处理器复位或FIFO禁止时，FIFO空状态标志位置1。但是，只要FIFO禁止，FIFO空中断便禁止。因此，退出复位时或FIFO禁止时，FIFO空中断始终不会处于等待处理状态。FIFO禁止时，FIFO下溢或上溢标志位将清零。

注： 接口的FIFO读取侧可检测到FIFO下溢。该状态必须先与写入侧时钟同步，然后才能被接口的FIFO写入侧检测到。因此，在接口的写入侧，当实际检测到FIFO下溢时，下溢状态会被延迟。

这种延迟的另一个含义是，当用户禁止写FIFO时，他们必须先等待下溢状态进行传播，之后才能重新使能FIFO。这是因为FIFO只能从接口的主处理器侧禁止。因此，WFEN信号必须传播到从处理器侧才能清零WFUF标志位，之后WFUF状态必须传播回主处理器侧才能检测到其清零。

8.2 FIFO数据寄存器可访问性

要从主处理器传输到从处理器的数据通过主处理器写入主处理器写FIFO数据寄存器（MWSRFDATA<15:0>）。之后，从处理器可从从处理器读FIFO数据寄存器（SRMWFDATA<15:0>）中读取数据。

要从从处理器传输到主处理器的数据通过从处理器写入从处理器写FIFO数据寄存器（SWMRFDATA<15:0>）。之后，主处理器可从主处理器读FIFO数据寄存器（MRSWFDATA）中读取数据。由于每次数据寄存器访问都会修改数据通道FIFO地址指针，因此数据将以单个实体（即，一个字或字节）的形式写入和读取。

写FIFO数据寄存器（主处理器MWSRFDATA<15:0>和从处理器SWMRFDATA<15:0>）为只写寄存器。读取这些寄存器将返回全0，不会影响FIFO地址指针。读FIFO数据寄存器（主处理器MRSWFDATA<15:0>和从处理器SRMWFDATA<15:0>）为只读寄存器。对这些寄存器的写操作将不起作用。

注： 不得对MWSRFDATA/MRSWFDATA或SRMWFDATA/SWMRFDATA寄存器执行读-修改-写操作。

8.2.1 FIFO数据大小

与任何SFR一样，FIFO数据寄存器可按字节访问或按字访问。为了在写入FIFO时支持字节和字大小的数据，按MSB或按字写入写FIFO数据寄存器将修改数据通道FIFO写地址指针。类似地，按MSB或按字读取读FIFO数据寄存器将修改数据通道FIFO读地址指针。

使用FIFO进行字节大小的数据传输时，必须先访问FIFO数据寄存器最低有效字节（Least Significant Byte，LSB），然后再访问MSB。

8.2.2 FIFO大小和寻址

FIFO作为循环缓冲区工作，每个缓冲区分别使用写和读地址指针确定下一个写和读地址。地址指针均在其相应的操作完成后进行修改。因此，在每次写操作之后，写地址指针将指向FIFO内的下一个空闲存储单元，并且在每次读操作之前，读地址指针将指向要读取的下一个数据存储单元。

注： dsPIC33CH系列的FIFO为32字深。有关具体器件的FIFO大小，请参见具体器件数据手册。FIFO地址指针无法由用户在外部控制或检测（但用户可通过禁止FIFO来复位指针（FIFO刷新））。

当读地址指针和写地址指针的值相等时，循环缓冲区被视为空，并且FIFO空状态位（写FIFO为WFEMPTY（MSI1FIFOCS<8>），读FIFO为RFEMPTY（MSI1FIFOCS<0>））置1。这还将在接口的写入侧为数据发送器处理器产生数据请求中断（MSIWFEIF（IFS8<11>））。

8.2.3 写入FIFO数据通道

当数据发送器写入写FIFO数据寄存器时（写FIFO为MWSRFDATA/SWMRFDATA，读FIFO为MRSWFDATA/SRMWFDATA），数据将被写入FIFO内的下一个空闲存储单元。

当向所寻址的FIFO存储单元装入数据时，写地址指针将进行调整以指向循环缓冲区内的下一个空闲存储单元。如果没有剩余的空闲存储单元，则写FIFO满状态位（写FIFO为WFFULL（MSI1FIFOCS<9>），读FIFO为RFFULL（MSI1FIFOCS<1>））置1。

注： 应用必须先测试FIFO空状态标志位的状态才能从FIFO读取数据。从空FIFO数据寄存器读取数据将导致数据下溢。

8.2.4 FIFO中断

FIFO空状态位用于为主处理器和从处理器产生中断。这些中断旨在用作数据传输协议的一部分。但是，如果应用不需要，则可在中断控制器（MSIxIE位）中禁止这些中断。表8-1给出了与FIFO相关的中断。

表8-1： MSI FIFO中断

内核	中断	备注
主	MSIFLTIF	主处理器读FIFO或写FIFO上溢或下溢故障中断
从	MSIFLTIF	从处理器读FIFO或写FIFO上溢或下溢故障中断
主	MSIWFEIF	主处理器写FIFO空中断
从	MSIWFEIF	从处理器写FIFO空中断
主	MSIDTIF	主处理器FIFO数据就绪中断
从	MSIDTIF	从处理器FIFO数据就绪中断

8.2.5 FIFO空中断（MSIWFEIF）

如果FIFO被视为空，则FIFO空状态标志位置1并且会为数据发送器处理器产生FIFO空中断。中断将在FIFO空状态标志位（写FIFO为WFEMPTY（MSI1FIFOCS<8>），读FIFO为RFEMPTY（MSI1FIFOCS<0>））从逻辑0变为逻辑1时产生。向FIFO写入数据将清零FIFO空状态标志位并向接收器发送数据有效中断。

在发生主处理器复位或FIFO禁止时，FIFO空状态标志位将设置为逻辑1。但是，只要FIFO禁止，FIFO空中断便禁止。因此，退出复位时或FIFO禁止时，FIFO空中断始终不会处于等待处理状态。

8.2.6 FIFO数据有效（就绪）中断（MSIDTIF）

如果将数据写入之前为空的FIFO，则FIFO空状态标志位清零并且会为数据接收器处理器产生FIFO数据有效中断。中断根据FIFO中可用数据的不同阈值产生。

中断逻辑有4种工作方式。逻辑由RFITSEL<1:0>位（主处理器为MSI1CON<11:10>，从处理器为SI1CON<11:10>）确定。

在主处理器写入FIFO或从处理器写入FIFO并且FIFO中的数据就绪后，可以在以下条件下产生中断：

1. 第一个FIFO数据就绪时中断。
2. FIFO 50%满时中断。
3. FIFO 75%满时中断。
4. FIFO 100%满时中断。

8.2.7 FIFO上溢和下溢状态以及中断

如果FIFO满状态位（写FIFO为WFFULL（MSI1FIFOCS<9>），读FIFO为RFFULL（MSI1FIFOCS<1>））置1后数据发送器向FIFO写入数据，则FIFO占用逻辑将检测到上溢条件并将FIFO上溢标志位（写FIFO为WFOF（MSI1FIFOCS<11>），读FIFO为RFOF（MSI1FIFOCS<3>））置1。请注意，数据写入将被忽略，FIFO写指针不会被修改，FIFO的内容将得以保留。

类似地，如果FIFO空状态位置1后数据接收器尝试从FIFO中读取数据，则FIFO占用逻辑将检测到下溢条件并将FIFO下溢标志位（写FIFO为WFUF（MSI1FIFOCS<10>），读FIFO为RFUF（MSI1FIFOCS<2>））置1。FIFO读指针不会在读操作前进行调整（典型做法），从而导致重新读取最近读取的FIFO地址。

FIFO有多种使用方法，下面介绍了其中一些方法。可以请求数据，也可将数据推送到处理器。可直接隐含数据应答（使用FIFO空状态位状态或处理器中断）。

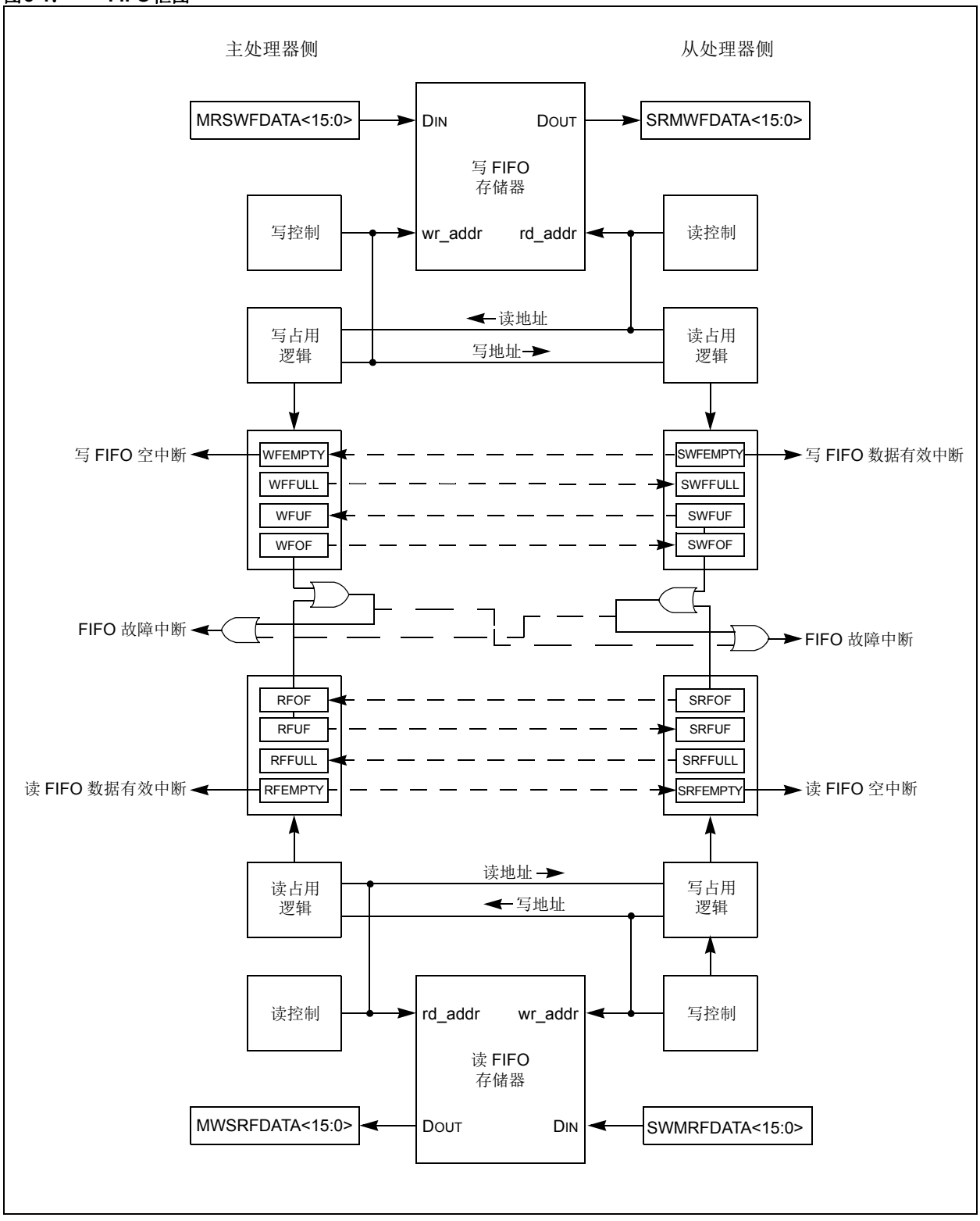
例8-1：　主处理器到从处理器的写入操作

```
////////////////////////////////////////////////////////////////////////////
Example of the Master project to write data to the Slave

#include "p33CH128RA508.h"
unsigned char Count;
int      main()
{
              MSI1FIFOCSbits.WFEN=1;
              Count=1;
while(Count<=32                           //buffer size of 32
{
              MWSRFDATA=Count;            //Master write to FIFO
              Count++;
}
while(1);

}

////////////////////////////////////////////////////////////////////////////
Example of the Slave project to Read the data written by the Master

#include "p33CH128RA508S1.h"
unsigned int  SRdata[32];
unsigned char Count;
int      main(void) {

         while(SI1FIFOCSbits.SRFFULL==0);    // Wait till the 32 but buffer is full
         Count=32;
         while(Count!=0)
         {
              SRdata[Count]=SRMWFDATA;
              Count--;
         }
         while(1);

}
```

例8-2：　从处理器到主处理器的写入操作

```
////////////////////////////////////////////////////////////////////////////
Example of the Slave project to Write data to Master (Master enables the Slave write bit)

#include "p33CH128RA508S1.h"

unsigned char  Count;
int     main(void)
{
        while(SI1FIFOCSbits.SWFEN==0); // wait till Master enables the Masters Read FIFO
        Count=1;
        while(Count<=32)                // Fill till the buffer is full
        {
        SWMRFDATA=Count;
        Count++;
        }

        while(1);
    }

////////////////////////////////////////////////////////////////////////////
Example of the Master project to read the data written by slave

#include "p33CH128RA508.h"
unsigned char  Count;
unsigned int   MRdata[32];
int     main()
{
        MSI1FIFOCSbits.RFEN=1;     // enable the read (Slave SWFEN)
        while(MSI1FIFOCSbits.RFFULL==0); // wait till the read buffer is full
        Count=32;
        while(Count!=0)
        {
        MRdata[Count]=MRSWFDATA;
        Count--;
        }
while(1);

}
////////////////////////////////////////////////////////////////////////////
```

图8-1： FIFO框图


主处理器侧
从处理器侧
MRSWFDATA<15:0>
DIN
DOUT
SRMWFDATA<15:0>
写 FIFO
存储器
写控制
wr_addr
rd_addr
读控制
读地址
写占用
逻辑
写地址
读占用
逻辑
写 FIFO 空中断
WFEMPTY
WFFULL
WFUF
WFOF
SWFEMPTY
SWFFULL
SWFUF
SWFOF
写 FIFO 数据有效中断
FIFO 故障中断
FIFO 故障中断
RFOF
RFUF
RFFULL
RFEMPTY
SRFOF
SRFUF
SRFFULL
SRFEMPTY
读 FIFO 数据有效中断
读 FIFO 空中断
读地址
写地址
读占用
逻辑
写占用
逻辑
读控制
rd_addr
wr_addr
写控制
读 FIFO
存储器
MWSRFDATA<15:0>
DOUT
DIN
SWMRFDATA<15:0>

8.2.8 将FIFO与DMA配合使用

MSI FIFO可与DMA模块配合使用。使用DMA执行FIFO写操作时，FIFO空中断用于触发相应的数据写DMA通道。FIFO变为空状态时，该中断即置为有效。DMA通道随后可将下一个数据块传输到FIFO。数据块大小任意，最大可达到FIFO的容量。

使用DMA执行FIFO读操作时，FIFO数据有效中断用于触发相应的数据读DMA通道。只要FIFO不再为空，该中断便会置为有效，并且在FIFO变为空状态前一直保持有效状态。这将允许重新触发DMA通道，以及继续传送数据，直至所有数据传送完毕（FIFO为空），或者DMA认为传输已完成。如上文所述，只要数据读DMA通道清空FIFO，FIFO空中断便会置为有效，并导致数据写DMA通道重新装入FIFO。

8.2.9 使用FIFO错误中断

出现FIFO上溢或下溢条件时，MSI FIFO将向主处理器和从处理器产生FIFO错误中断。

数据发送器处理器负责修正（写相关）上溢错误。在检测到错误后到再次使用FIFO之间的某一时刻，相应（粘住）上溢状态位必须清零。此外，数据发送器处理器还可检测FIFO下溢错误的状态。尽管数据发送器无法修正读下溢，但可以在错误持续存在和/或询问数据接收器处理器时停止向FIFO发送数据。

类似地，数据接收器处理器负责修正（读相关）下溢错误。在检测到错误后到再次使用FIFO之间的某一时刻，相应（粘住）下溢状态位必须清零。此外，数据接收器处理器还可检测FIFO上溢错误的状态。尽管数据接收器无法修正写上溢，但可以在错误持续存在和/或询问数据发送器处理器时停止从FIFO读取数据。

8.2.10 FIFO通道延时

从写入数据到数据可供接收之间存在一段延时。这段延时称为FIFO通道延时。同步延时为3个目标时钟。因此，对于从发送器传送到接收器的信号/数据，延时为3个接收器时钟。类似地，对于从接收器传送到发送器的信号/数据，延时为3个发送器时钟。

写操作所需的时间将为：

1. FIFO写周期（1个发送器时钟）。
2. 3个同步时钟（3个接收器时钟）。

8.2.11 FIFO的从处理器复位

由于读FIFO和写FIFO均从接口的主处理器侧控制，因此必须使主处理器了解从处理器复位情况才能重新启动FIFO通道。因此，除非基于邮箱的协议通知主处理器从处理器刚刚复位，否则主处理器应监视从处理器复位状态位SLVRST（MSI1STAT<15>）或使能从处理器复位事件中断允许位SRSTIE（MSI1CON<7> = 1），并在检测到从处理器复位时重新启动使能的FIFO。

9.0 处理器间的中断

主处理器和从处理器可以直接相互中断。主处理器可通过将MTSIRQ控制位（MS1CON<9>）置为有效来向从处理器发出中断请求。类似地，从处理器可通过将STMIRQ控制位（SI1CON<9>）置为有效向主处理器发出中断请求。

对于主处理器发送到从处理器的中断请求，使用中断应答控制位MTSIACK（MSI1CON<8>）来应答中断；对于从处理器发送到主处理器的中断请求，使用STMIACK（SI1CON<8>）来应答中断。

所有主处理器/从处理器中断控制/状态位均可通过任一处理器读取。中断请求位由请求的处理器读/写，中断应答位由被中断的处理器通过MSI1CON控制寄存器读/写。中断请求位只能由被中断的处理器读取，中断应答位只能由请求的处理器通过MSI1STAT状态寄存器读取。

9.1 主处理器到从处理器的中断协议

当主处理器将MTSIRQ位置为有效时，它将与从处理器时钟同步以产生从处理器中断。主处理器将MTSIRQ位（MSI1CON<9>）置1后，从处理器将在IFSx寄存器中的MSIMIF位置1时获得中断。从从处理器的角度来看，中断会将只读状态位（SI1STAT<9>）置1。处理中断时，从处理器必须在中断处理程序中的某一时刻将STMIACK位置1以应答中断。

同步到主处理器时钟域后，主处理器将检测到MTSIACK（MSI1STAT<8>）= 1，然后清零其MTSIRQ位，撤销请求。当从处理器检测到STMIRQ> = 0时，它将完成握手。此时，从处理器将清零STMIACK以撤销应答，中断处理程序可随后退出。

例9-1： 主处理器到从处理器的中断协议

```
//////////////////Master code Master to Slave interrupt Protocol//////////////////////////

while(1)
{
    MSI1CONbits.MSTIRQ=1;                 // Interrupt to slave
    while(MSI1STATbits.MTSIACK==0);       // wait till slave acknowledges
    MSI1CONbits.MSTIRQ=0;                 // clear the interrupt to repeat the next
    while(MSI1STATbits.MTSIACK==1);       //  wait till slave clears the acknowledge

}

//////////////////Slave code for Master to Slave interrupt Protocol//////////////////
while(1)
{
    while(IFS8bits.MSIMIF==0);            // wait for the interrupt
    IFS8bits.MSIMIF=0;
    SI1CONbits.MTSIACK=1;                 // Acknowledge the master interrupt
    while(SI1STATbits.MTSIRQ==1);         //wait till master clears the interrupt request
    SI1CONbits.MTSIACK=0;                 //
}
```

注： 用户必须清零MTSIRQ才能产生其他中断。即，在MTSIRQ位已置1的情况下向其写入逻辑1不会产生其他中断脉冲。

主处理器应（但并非必须）先等待从处理器撤销STMIACK，然后再将MTSIRQ重新置为有效（以产生其他中断，假定MTSIRQ之前已清零）。当使用上述握手时，未等待从处理器撤销STMIACK将使新中断保持等待处理状态，直到当前中断退出为止。

9.2 从处理器到主处理器的中断协议

当从处理器将STMIRQ位置为有效时，它将与主处理器时钟同步以产生主处理器中断（MSISxIF）。从处理器将STMIRQ位（SI1CON<9>）置1后，主处理器将在IFSx寄存器中的MSISxIF位置1时获得中断。从主处理器的角度来看，中断会将只读状态位（MSI1STAT<9>）置1。中断处理完成时，主处理器必须在中断处理程序的末尾将MTSIACK位置1以应答中断。

同步到从处理器时钟域后，从处理器将检测到STMIACK（SI1STAT<8>）= 1，然后清零其STMIRQ（SI1CON<9>）位，撤销请求。当主处理器检测到STMIRQ（MSI1STAT<9>）= 0时，它将完成握手。此时，主处理器将清零STMIACK（MSI1CON<8>）以撤销应答，中断处理程序可随后退出。

例9-2： 从处理器到主处理器的中断协议

```
//////////////////Master code Slave to Master Interrupt Protocol////////////////////////

while(1)
{
    while(IFS8bits.MSIS1IF==0);          // wait for the Slave interrupt
    IFS8bits.MSIS1IF=0;
    MSI1CONbits.STMIACK=1;               // ACK the slave
    while(MSI1STATbits.STMIRQ==1);       // wait till the slave clears the Interrupt request
    MSI1CONbits.STMIACK=0;

}

//////////////////Slave code for Slave to Master Interrupt Protocol//////////////////
while(1)
{
    SI1CONbits.STMIRQ=1;                 // Interrupt the Master
    while(SI1STATbits.STMIACK==0)        // Wait for ACK from the Master
    SI1CONbits.STMIRQ=0;                 // Clear the interrupt request
    while(SI1STATbits.STMIACK==1)        //wait till Master clears the acknowledge
}
```

10.0 主处理器/从处理器复位交互

工作在任何模式下时，用户均可基于2个熔丝的状态选择主处理器和从处理器的其他运行时复位（定义为非POR、BOR、$\overline{\text{MCLR}}$或$\overline{\text{SMCLR}}$复位（双调试模式下）的任何复位）如何影响SLVEN（MSI1CON<15>）控制位：主处理器从处理器复位使能位（MSRE，FSLV1DEVOPT<15>）和从处理器复位使能位（SSRE，FSLV1DEVOPT<14>）。

SLVEN位本质上是一个从处理器复位控制位，因此这两个熔丝可用于有效地耦合或解耦主处理器和从处理器运行时复位（MSRE），还可以确定发生从处理器运行时复位（SSRE）时从处理器继续工作还是自行禁止。默认状态（当MSRE和SSRE均未编程时）是允许主处理器复位和从处理器复位将SLVEN复位并禁止从处理器。

注： 当MSRE = 1时，任何主处理器复位都将复位从处理器（操作码复位、看门狗定时器超时复位、陷阱复位和非法指令复位）。当MSRE = 0时，如果主处理器发生复位，从处理器可以独立运行而不会复位。SSRE位决定从处理器复位期间是否禁止SLVEN位。如果SSRE = 1，从处理器产生的复位将复位从处理器复位使能位。如果SSRE = 0，从处理器产生的复位将不会复位MSI模块中的从处理器复位使能位。

10.1 从处理器复位耦合控制

在所有工作模式下，用户均可以通过使用主处理器从处理器复位使能（Master Slave Reset Enable，MSRE）熔丝耦合或解耦主处理器运行时复位和处理器复位。通过将所选复位源指向SLVEN位复位，可以有效地耦合复位。

在所有工作模式下，用户还可以通过使用从处理器复位使能（SSRE）熔丝选择是否在发生从处理器运行时复位时复位SLVEN位。

用户可以选择在从处理器复位时复位SLVEN，因为该事件可能指示从处理器执行过程中出现问题。从处理器将被置于复位状态，并通知主处理器（通过从处理器复位事件中断）尝试纠正问题。主处理器必须通过将SLVEN位再次置1来重新使能从处理器。

或者，用户也可以选择在从处理器复位时不停止从处理器，而是允许它在复位后重新开始执行并尽快继续工作。从处理器复位事件中断仍然会发生，但可被主处理器忽略。

表10-1： 应用模式SLVEN复位控制真值表

MSRE	SSRE	SLVEN位复位源	应用影响
0	0	POR/BOR/$\overline{\text{MCLR}}$	• 在POR、BOR或$\overline{\text{MCLR}}$复位时，从处理器将复位并禁止。主处理器必须重新使能从处理器。 • 从处理器运行时复位不会禁止从处理器。从处理器将复位并继续执行（可选择中断主处理器）。
1	0	主处理器复位(1)	• 在任何主处理器复位时，从处理器均将复位并禁止。主处理器必须重新使能从处理器。 • 从处理器运行时复位不会禁止从处理器。从处理器将复位并继续执行（可选择中断主处理器）。
0	1	从处理器复位(2)	• 在任何从处理器运行时复位时，从处理器均将复位并禁止（可选择中断主处理器）。主处理器必须重新使能从处理器才能执行从处理器代码。 • 主处理器运行时复位不会影响从处理器的工作。
1	1	主处理器复位(1)/从处理器复位(2)	• 在任何从处理器运行时复位或主处理器复位时，从处理器均将复位并禁止。主处理器必须重新使能从处理器。这代表默认状态（MSRE和SSRE未编程）。

注 1： 主处理器复位包括任何主处理器复位，例如POR/BOR/$\overline{\text{MCLR}}$复位。

2： 从处理器复位包括任何从处理器复位以及POR/BOR/$\overline{\text{MCLR}}$复位（在应用模式下）。

11.0 处理器间的工作模式状态

所有处理器的应用工作模式状态都可通过MSI1STAT寄存器获得。每个从处理器都可以检测到主处理器的工作状态，主处理器可以检测到每个从处理器的工作状态。从处理器无法直接查看任何其他从处理器的工作状态。

11.1 从处理器复位状态（对于主处理器）

通过检测主处理器侧的SLVRST位的状态，主处理器可以获得从处理器复位的状态。该位将保持置1，直至从处理器退出复位状态。当从处理器被禁止（SLVEN = 0）时，它将保持在复位状态，因此SLVRST将置1。

该位未映射到接口的从处理器侧，在主处理器侧为R-0。器件POR、BOR或$\overline{\text{MCLR}}$复位将始终复位主处理器和从处理器，从而复位SLVRST位。否则该位代表从处理器复位的状态。因此，如果从处理器也因主处理器复位而复位（或已经处于复位状态），或者从处理器因主处理器复位或在主处理器复位之前禁止（SLVEN = 0），则SLVRST位将复位为逻辑1。如果从处理器已使能（SLVEN = 1），则它不受主处理器复位的影响，SLVRST位将复位为逻辑0。

如果主处理器希望在从处理器复位时采取措施，SLVRST位可用于产生“从处理器复位事件”中断。要使此中断起作用，应将SRSTIE（MSI1CON<7>）位置1。该位使能时，在出现任何从处理器运行时复位事件（即，非POR/BOR/$\overline{\text{MCLR}}$）的前沿（仅限前沿）时，将为主处理器产生从处理器复位事件中断。

注：	主处理器中断控制器中的相关“从处理器复位事件”中断标志位必须在中断返回之前由中断服务程序（Interrupt Service Routine，ISR）清零，以避免重新进入中断。 为了避免在有意禁止从处理器时发生意外的“从处理器复位事件”中断，用户必须在禁止从处理器（SLVEN = 0）之前清零从处理器复位事件中断允许位（SRSTIE（MSI1CON<7>）= 0）。

SLVRST位旨在为主处理器提供一种方法来检查其在尝试与从处理器通信前从处理器能否进行响应。因此，它在整个从处理器复位事件期间保持有效，并且不能由主处理器清零。但是，它也是一个中断事件源，但仅在SLVRST从0转换为1时有效。如果它保持有效或被清零（即，退出从处理器复位状态时），则不会发生后续中断。

当SLVRST = 1时，主处理器可以：

- 等待ISR中的SLVRST = 0
- 记录事件并恢复应用操作，同时定期检查SLVRST状态位的状态
- 通过禁止从处理器（SLVEN = 0）来将其重新初始化，并在重新使能从处理器前向从处理器PRAM重新装入数据

11.1.1 主处理器处于休眠模式时的SLVRST

如果主处理器处于休眠模式且从处理器复位事件设置SLVRST = 1，则产生的“从处理器复位事件”中断将能够唤醒主处理器。

11.2 主处理器复位状态（对于从处理器）

通过检测从处理器侧的MSTRST（SI1STAT<15>）位的状态，从处理器可以获得主处理器复位的状态。该位将保持置1，直至主处理器退出复位状态。

该位未映射到接口的主处理器侧，在从处理器侧为R-0。除非MSRE熔丝 = 0，否则它将始终读为0（因为当MSRE = 1时，只要主处理器复位，从处理器也将复位）。器件POR或BOR复位将始终复位从处理器，从而复位MSTRST位。否则该位代表主处理器复位的状态（即，当MSRE = 0时）。

如果从处理器希望在主处理器复位时采取措施，MSTRST位可用于产生“主处理器复位事件”中断。该中断通过将STMIRQ置1（SI1CON<9> = 1）来允许。允许时，在出现任何主处理器运行时复位事件（即，非POR/BOR/$\overline{\text{MCLR}}$）的前沿时，将为从处理器产生“主处理器复位事件”中断。中断会将主处理器中断控制器宏中的相关中断标志位置1。

注： “主处理器复位事件”中断边沿敏感，仅在中断允许位置1时主处理器进入复位状态的情况下发生。但是，从处理器中断控制器中的相关“主处理器复位事件”中断标志位必须在中断返回之前由ISR清零，以避免重新进入中断。

11.2.1 MSTRST使用示例

MSTRST位旨在为从处理器提供一种方法（当因MSRE熔丝 = 0而独立复位时）来检查其在尝试与主处理器通信前主处理器能否进行响应。因此，它在整个主处理器复位事件期间保持有效，并且不能由从处理器清零。但是，它也是一个中断事件源，但仅在MSTRST从0转换为1时有效。如果它保持有效或被清零（即，退出主处理器复位状态时），则不会发生后续中断。

当MSTRST = 1时，从处理器可以：

- 记录事件并恢复应用操作，同时定期检查MSTRST状态位的状态
- 等待ISR中的MSTRST = 0（有效地将整个器件置于停止状态）
- 重启（当它因临时的故障状况而误读状态时）
- 验证PRAM的内容（即，校验和），然后停止或重启

如果用户需要了解过去的从处理器复位事件，则可以通过使用相关的ISR代码记录事件来实现此目的。

11.2.2 从处理器处于休眠模式时的MSTRST

如果将MSRE（主处理器从处理器复位使能）熔丝编程（为逻辑0），则主处理器和从处理器复位将无关。如果是这种情况，并且从处理器处于休眠模式，则主处理器复位事件将设置MSTRST = 1，并且产生的“主处理器复位事件”中断将能够唤醒从处理器。如果MSRE = 1，则主处理器复位和从处理器复位相关，因此主处理器复位也会复位从处理器（并退出休眠模式）。

11.3 系统看门狗定时器状态

通过检测主处理器侧的SLVWDRST（MSI1STAT<14>）位的状态，主处理器可以获得从处理器看门狗定时器（WDT）复位的状态。如果WDT已超时并强制进行从处理器复位，则该位将置1；它将保持置1状态，直至由主处理器清零。该位未映射到接口的从处理器侧，而是来自主处理器侧的R/C（仅限读取或清零）。

当主处理器复位置为有效时，SLVWDRST位将复位（从处理器的其余部分也将复位）。因此，主处理器WDT复位状态（针对从处理器）没有意义。

注： 如果禁止从处理器（SLVEN = 0），SLVWDRST将不受影响。

11.3.1 低功耗工作模式状态

从处理器低功耗工作模式状态由SLVPWR<1:0>（MSI1STAT<13:12>）位指示。这些位在接口的从处理器侧不可见，只能从主处理器侧读取。类似地，主处理器低功耗工作模式状态由MSTPWR<1:0>（SI1STAT<13:12>）位指示。这些位未映射到接口的主处理器侧，只能从从处理器侧读取。

12.0 相关应用笔记

本节列出了与手册本章内容相关的应用笔记。这些应用笔记可能并不是专为dsPIC33/PIC24器件系列而编写的，但是概念是相关的，通过适当修改即可使用，但在使用中可能会受到一定限制。当前与主从接口（MSI）模块相关的应用笔记有：

标题	应用笔记编号
目前没有相关的应用笔记。	N/A

注： 如需获取更多dsPIC33/PIC24系列器件的应用笔记和代码示例，请访问Microchip网站（www.microchip.com）。

13.0 版本历史

版本A（2016年8月）

这是本文档的初始版本。

版本B（2018年3月）

- 表：
  - 更新了表2-1和表2-2。
- 示例：
  - 增加了例4-2、例9-1和例9-2。
  - 更新了例7-1。
- 寄存器：
  - 更新了寄存器2-1和寄存器2-9。
- 章节：
  - 更新了**第8.2.6节“FIFO数据有效（就绪）中断（MSIDTIF）”**。

注：

请注意以下有关 Microchip 器件代码保护功能的要点：

- Microchip 的产品均达到 Microchip 数据手册中所述的技术指标。
- Microchip 确信：在正常使用的情况下， Microchip 系列产品是当今市场上同类产品中最安全的产品之一。
- 目前，仍存在着恶意、甚至是非法破坏代码保护功能的行为。就我们所知，所有这些行为都不是以 Microchip 数据手册中规定的操作规范来使用 Microchip 产品的。这样做的人极可能侵犯了知识产权。
- Microchip 愿与那些注重代码完整性的客户合作。
- Microchip 或任何其他半导体厂商均无法保证其代码的安全性。代码保护并不意味着我们保证产品是 “牢不可破”的。

代码保护功能处于持续发展中。Microchip 承诺将不断改进产品的代码保护功能。任何试图破坏 Microchip 代码保护功能的行为均可视为违反了 《数字器件千年版权法案 （Digital Millennium Copyright Act）》。如果这种行为导致他人在未经授权的情况下，能访问您的软件或其他受版权保护的成果，您有权依据该法案提起诉讼，从而制止这种行为。

Microchip 位于美国亚利桑那州 Chandler 和 Tempe 与位于俄勒冈州 Gresham的全球总部、设计和晶圆生产厂及位于美国加利福尼亚州和印度的设计中心均通过了 ISO/TS-16949:2009 认证。Microchip 的 PIC® MCU 与 dsPIC® DSC、KEELOQ® 跳码器件、串行 EEPROM、单片机外设、非易失性存储器和模拟产品严格遵守公司的质量体系流程。此外，Microchip 在开发系统的设计和生产方面的质量体系也已通过了 ISO 9001:2000 认证。

QUALITY MANAGEMENT SYSTEM
CERTIFIED BY DNV
═ ISO/TS 16949═

ISBN：978-1-5224-3358-3

全球销售及服务网点

美洲

公司总部 Corporate Office
2355 West Chandler Blvd.
Chandler, AZ 85224-6199
Tel: 1-480-792-7200
Fax: 1-480-792-7277
技术支持：
http://www.microchip.com/support
网址：www.microchip.com

亚特兰大 Atlanta
Duluth, GA
Tel: 1-678-957-9614
Fax: 1-678-957-1455

奥斯汀 Austin, TX
Tel: 1-512-257-3370

波士顿 Boston
Westborough, MA
Tel: 1-774-760-0087
Fax: 1-774-760-0088

芝加哥 Chicago
Itasca, IL
Tel: 1-630-285-0071
Fax: 1-630-285-0075

达拉斯 Dallas
Addison, TX
Tel: 1-972-818-7423
Fax: 1-972-818-2924

底特律 Detroit
Novi, MI
Tel: 1-248-848-4000

休斯敦 Houston, TX
Tel: 1-281-894-5983

印第安纳波利斯 Indianapolis
Noblesville, IN
Tel: 1-317-773-8323
Fax: 1-317-773-5453
Tel: 1-317-536-2380

洛杉矶 Los Angeles
Mission Viejo, CA
Tel: 1-949-462-9523
Fax: 1-949-462-9608
Tel: 1-951-273-7800

罗利 Raleigh, NC
Tel: 1-919-844-7510

纽约 New York, NY
Tel: 1-631-435-6000

圣何塞 San Jose, CA
Tel: 1-408-735-9110
Tel: 1-408-436-4270

加拿大多伦多 Toronto
Tel: 1-905-695-1980
Fax: 1-905-695-2078

亚太地区

中国 - 北京
Tel: 86-10-8569-7000

中国 - 成都
Tel: 86-28-8665-5511

中国 - 重庆
Tel: 86-23-8980-9588

中国 - 东莞
Tel: 86-769-8702-9880

中国 - 广州
Tel: 86-20-8755-8029

中国 - 杭州
Tel: 86-571-8792-8115

中国 - 南京
Tel: 86-25-8473-2460

中国 - 青岛
Tel: 86-532-8502-7355

中国 - 上海
Tel: 86-21-3326-8000

中国 - 沈阳
Tel: 86-24-2334-2829

中国 - 深圳
Tel: 86-755-8864-2200

中国 - 苏州
Tel: 86-186-6233-1526

中国 - 武汉
Tel: 86-27-5980-5300

中国 - 西安
Tel: 86-29-8833-7252

中国 - 厦门
Tel: 86-592-238-8138

中国 - 香港特别行政区
Tel: 852-2943-5100

中国 - 珠海
Tel: 86-756-321-0040

台湾地区 - 高雄
Tel: 886-7-213-7830

台湾地区 - 台北
Tel: 886-2-2508-8600

台湾地区 - 新竹
Tel: 886-3-577-8366

亚太地区

澳大利亚 Australia - Sydney
Tel: 61-2-9868-6733

印度 India - Bangalore
Tel: 91-80-3090-4444

印度 India - New Delhi
Tel: 91-11-4160-8631

印度 India - Pune
Tel: 91-20-4121-0141

日本 Japan - Osaka
Tel: 81-6-6152-7160

日本 Japan - Tokyo
Tel: 81-3-6880-3770

韩国 Korea - Daegu
Tel: 82-53-744-4301

韩国 Korea - Seoul
Tel: 82-2-554-7200

马来西亚 Malaysia - Kuala Lumpur
Tel: 60-3-7651-7906

马来西亚 Malaysia - Penang
Tel: 60-4-227-8870

菲律宾 Philippines - Manila
Tel: 63-2-634-9065

新加坡 Singapore
Tel: 65-6334-8870

泰国 Thailand - Bangkok
Tel: 66-2-694-1351

越南 Vietnam - Ho Chi Minh
Tel: 84-28-5448-2100

欧洲

奥地利 Austria - Wels
Tel: 43-7242-2244-39
Fax: 43-7242-2244-393

丹麦 Denmark - Copenhagen
Tel: 45-4450-2828
Fax: 45-4485-2829

芬兰 Finland - Espoo
Tel: 358-9-4520-820

法国 France - Paris
Tel: 33-1-69-53-63-20
Fax: 33-1-69-30-90-79

德国 Germany - Garching
Tel: 49-8931-9700

德国 Germany - Haan
Tel: 49-2129-3766400

德国 Germany - Heilbronn
Tel: 49-7131-67-3636

德国 Germany - Karlsruhe
Tel: 49-721-625370

德国 Germany - Munich
Tel: 49-89-627-144-0
Fax: 49-89-627-144-44

德国 Germany - Rosenheim
Tel: 49-8031-354-560

以色列 Israel - Ra'anana
Tel: 972-9-744-7705

意大利 Italy - Milan
Tel: 39-0331-742611
Fax: 39-0331-466781

意大利 Italy - Padova
Tel: 39-049-7625286

荷兰 Netherlands - Drunen
Tel: 31-416-690399
Fax: 31-416-690340

挪威 Norway - Trondheim
Tel: 47-7289-7561

波兰 Poland - Warsaw
Tel: 48-22-3325737

罗马尼亚 Romania - Bucharest
Tel: 40-21-407-87-50

西班牙 Spain - Madrid
Tel: 34-91-708-08-90
Fax: 34-91-708-08-91

瑞典 Sweden - Gothenberg
Tel: 46-31-704-60-40

瑞典 Sweden - Stockholm
Tel: 46-8-5090-4654

英国 UK - Wokingham
Tel: 44-118-921-5800
Fax: 44-118-921-5820