第 18 章 串行外设接口 （SPI）

目录

本章包括下列主题：

18.1 简介

串行外设接口（Serial Peripheral Interface，SPI）模块是用于同其他外设或单片机器件进行通信的同步串行接口。这些外设可以是串行 EEPROM、移位寄存器、显示驱动器和 A/D 转换器等。SPI 模块与 Motorola 的 SPI 和 SIOP 接口兼容。

根据器件型号不同，dsPIC33F 系列在单个器件上提供一个或两个 SPI 模块。两个模块（指定为 SPI1 和 SPI2）具有相同的功能。SPI1 模块在所有器件上都提供，而 SPI2 模块在许多高引脚数封装器件中提供。

注：	在本章中，SPI 模块统称为 SPIx，或分别称为 SPI1 和 SPI2。特殊功能寄存器也使用类似的符号表示。例如，SPIxCON 指 SPI1 或 SPI2 模块的控制寄存器。

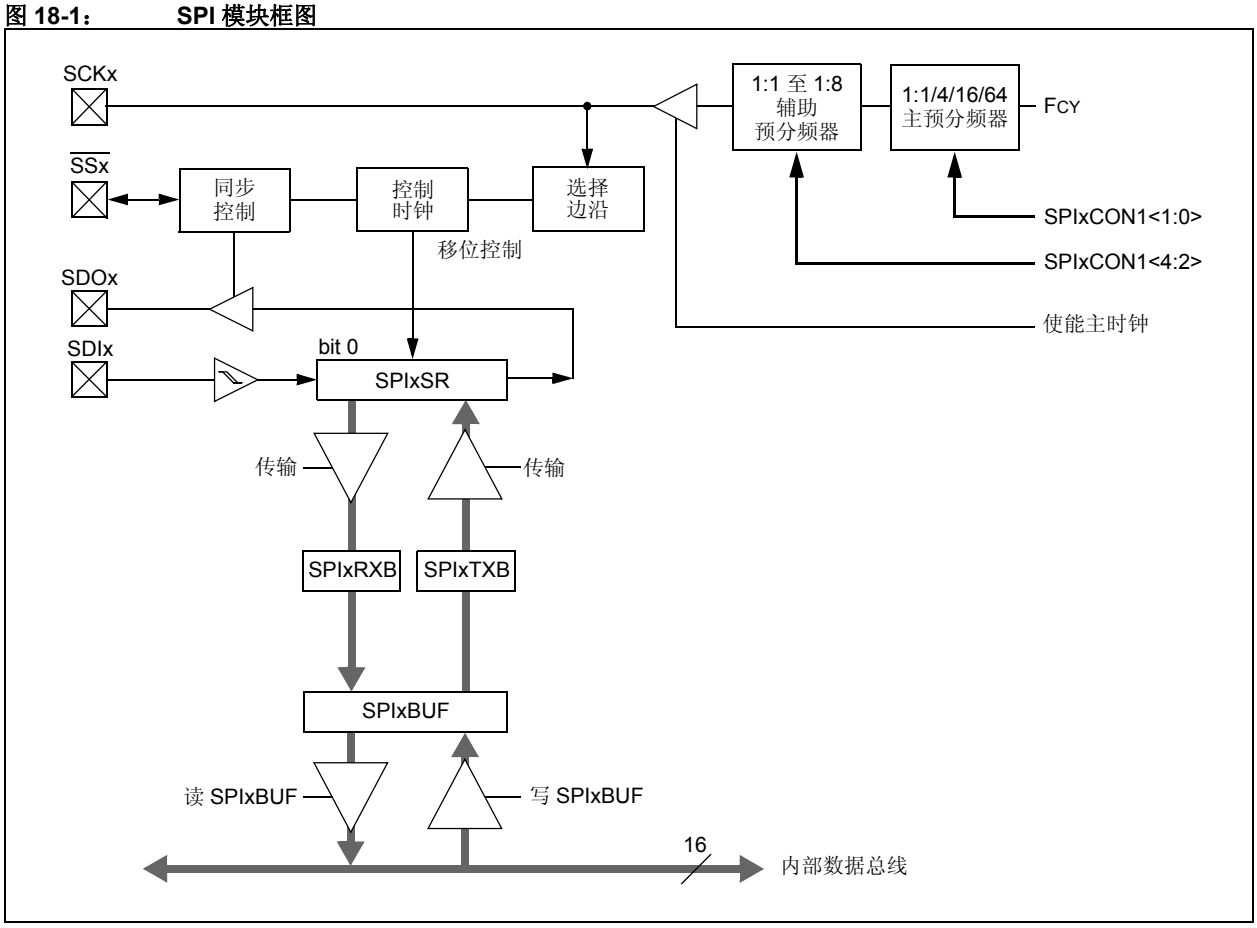
SPIx 串行接口由 4 个引脚组成：

- SDIx：串行数据输入
- SDOx：串行数据输出
- SCKx：移位时钟输入或输出
- $\overline{\text{SSx}}$/FSYNCx：低电平有效从选择或帧同步 I/O 脉冲

可将 SPIx 模块配置为使用 2、3 或 4 个引脚进行工作。在 3 引脚模式下，不使用 $\overline{\text{SSx}}$。在 2 引脚模式下，不使用 SDOx 和 $\overline{\text{SSx}}$。

图 18-1 给出了该模块的框图。

图 18-1： SPI 模块框图



18.2 SPI 寄存器

SPIxSTAT：SPIx 状态和控制寄存器

SPIx 状态和控制 （SPIxSTAT）寄存器用于指示各种状态条件，如接收溢出、发送缓冲区满以及接收缓冲区满。该寄存器还可用于指定空闲模式下模块的工作。该寄存器还包含一个可使能和禁止模块的位。

SPIxCON1：SPIx 控制寄存器 1

SPIx 控制寄存器 1 （SPIxCON1）用于指定时钟预分频比、主 / 从模式、字 / 字节通信、时钟极性和时钟 / 数字引脚操作。

SPIxCON2：SPIx 控制寄存器 2

SPIx 控制寄存器 2 （SPIxCON2）用于使能 / 禁止帧 SPI 操作。该寄存器还可用于指定帧同步脉冲的方向、极性和边沿选择。

SPIxBUF：SPIx 数据接收 / 发送缓冲寄存器

SPIx 数据接收 / 发送缓冲 （SPIxBUF）寄存器实际上是两个独立的内部寄存器：发送缓冲区（SPIxTXB）和接收缓冲区 （SPIxRXB）。这两个单向的 16 位寄存器共用 SPIxBUF 的 SFR 地址。如果用户应用程序将待发送数据写入 SPIxBUF 地址，在内部该数据会被写入 SPIxTXB 寄存器。

与此类似，当用户应用程序从 SPIxBUF 读取已接收数据时，在内部会从 SPIxRXB 寄存器读取该数据。

这种技术对发送 / 接收操作进行了双重缓冲，允许在后台连续地传输数据。发送和接收可同时进行。

此外，还有一个 16 位移位寄存器 （SPIxSR），该寄存器未被映射到存储空间。它用于将数据移入和移出 SPI 端口。

寄存器 18-1： SPIxSTAT：SPIx 状态和控制寄存器

R/W-0	U-0	R/W-0	U-0	U-0	U-0	U-0	U-0
SPIEN	—	SPISIDL	—	—	—	—	—
bit 15							bit 8

U-0	R/C-0	U-0	U-0	U-0	U-0	R-0	R-0
—	SPIROV	—	—	—	—	SPITBF	SPIRBF
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 **SPIEN**：SPIx 使能位
1 = 使能模块并将 SCKx、 SDOx、 SDIx 和 $\overline{SSx}$ 配置为串口引脚
0 = 禁止模块

bit 14 **未实现**：读为 0

bit 13 **SPISIDL**：空闲模式停止位
1 = 当器件进入空闲模式时，模块停止工作
0 = 在空闲模式下模块继续工作

bit 12-7 **未实现**：读为 0

bit 6 **SPIROV**：接收溢出标志位
1 = 一个新字节 / 字已完全接收并丢弃。用户软件没有读先前保存在 SPIxBUF 寄存器中的数据
0 = 未发生溢出

bit 5-2 **未实现**：读为 0

bit 1 **SPITBF**：SPIx 发送缓冲区满状态位
1 = 发送尚未开始， SPIxTXB 为满
0 = 发送已开始， SPIxTXB 为空
当 CPU 通过写 SPIxBUF 存储单元装入 SPIxTXB 时，该位由硬件自动置 1。
当 SPIx 模块将数据从 SPIxTXB 传输到 SPIxSR 时，该位由硬件自动清零。

bit 0 **SPIRBF**：SPIx 接收缓冲区满状态位
1 = 接收完成， SPIxRXB 为满
0 = 接收未完成， SPIxRXB 为空
当 SPIx 将数据从 SPIxSR 传输到 SPIxRXB 时，该位由硬件自动置 1。
当内核通过读 SPIxBUF 存储单元读 SPIxRXB 时，该位由硬件自动清零。

寄存器 18-2： SPIxCON1：SPIx 控制寄存器 1

U-0	U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
—	—	—	DISSCK	DISSDO	MODE16	SMP(1)	CKE(2)
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
SSEN	CKP	MSTEN	SPRE<2:0>			PPRE<1:0>	
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 15-13 **未实现：**读为 0

bit 12 **DISSCK：**禁止 SCKx 引脚位 （仅限 SPI 主模式）
1 = 禁止内部 SPI 时钟，引脚用作 I/O
0 = 使能内部 SPI 时钟

bit 11 **DISSDO：**禁止 SDOx 引脚位
1 = 模块不使用 SDOx 引脚；引脚用作 I/O
0 = SDOx 引脚由模块控制

bit 10 **MODE16：**字 / 字节通信选择位
1 = 采用字宽 （16 位）通信
0 = 采用字节宽 （8 位）通信

bit 9 **SMP：**SPIx 数据输入采样阶段位 (1)
主模式：
1 = 在数据输出时间的末端采样输入数据
0 = 在数据输出时间的中间采样输入数据
从模式：
当在从模式下使用 SPIx 时，必须将 SMP 清零

bit 8 **CKE：**SPIx 时钟边沿选择位 (2)
1 = 串行输出数据在时钟从工作状态转变为空闲状态时变化 （见 bit 6）
0 = 串行输出数据在时钟从空闲状态转变为工作状态时变化 （见 bit 6）

bit 7 **SSEN：**从选择使能位 （从模式）
1 = $\overline{SSx}$ 引脚用于从模式
0 = 模块不使用 $\overline{SSx}$ 引脚。引脚由端口功能控制

bit 6 **CKP：**时钟极性选择位
1 = 空闲状态时时钟信号为高电平；工作状态时为低电平
0 = 空闲状态时时钟信号为低电平；工作状态时为高电平

bit 5 **MSTEN：**主模式使能位
1 = 主模式
0 = 从模式

bit 4-2 **SPRE<2:0>：**辅助预分频比位 （主模式）
111 = 辅助预分频比 1:1
110 = 辅助预分频比 2:1
.
.
.
000 = 辅助预分频比 8:1

注 1：SMP 位只能在 MSTEN 位置 1 后置 1。如果 MSTEN = 0， SMP 位将保持清零。
2：在帧 SPI 模式下不使用 CKE 位。在帧 SPI 模式 （FRMEN = 1）下，将该位编程为 0。

寄存器 18-2： SPIxCON1：SPIx 控制寄存器 1 （续）

bit 1-0 **PPRE<1:0>：**主预分频比位 （主模式）
11 = 主预分频比 1:1
10 = 主预分频比 4:1
01 = 主预分频比 16:1
00 = 主预分频比 64:1

注 1：SMP 位只能在 MSTEN 位置 1 后置 1。如果 MSTEN = 0， SMP 位将保持清零。
2：在帧 SPI 模式下不使用 CKE 位。在帧 SPI 模式 （FRMEN = 1）下，将该位编程为 0。

寄存器 18-3： SPIxCON2：SPIx 控制寄存器 2

R/W-0	R/W-0	R/W-0	U-0	U-0	U-0	U-0	U-0
FRMEN	SPIFSD	FRMPOL	—	—	—	—	—
bit 15							bit 8

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	U-0
—	—	—	—	—	—	FRMDLY	—
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 **FRMEN**：帧 SPIx 支持位
1 = 使能帧 SPIx 支持 （$\overline{SSx}$ 引脚用作帧同步脉冲输入 / 输出）
0 = 禁止帧 SPIx 支持

bit 14 **SPIFSD**：帧同步脉冲方向控制位
1 = 帧同步脉冲输入 （从器件）
0 = 帧同步脉冲输出 （主器件）

bit 13 **FRMPOL**：帧同步脉冲极性位
1 = 帧同步脉冲为高电平有效
0 = 帧同步脉冲为低电平有效

bit 12-2 **未实现**：读为 0

bit 1 **FRMDLY**：帧同步脉冲边沿选择位
1 = 帧同步脉冲与第一个位时钟一致
0 = 帧同步脉冲比第一个位时钟提前

bit 0 **未实现**：禁止用户应用程序将该位设为 1。

寄存器 18-4： SPIxBUF：SPIx 数据接收 / 发送缓冲寄存器

R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
SPIx 发送和接收缓冲寄存器							
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
SPIx 发送和接收缓冲寄存器							
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 发送 / 接收缓冲区位

18.3 工作模式

SPI 模块使用以下灵活的工作模式：

- 8 位和 16 位数据发送 / 接收
- 主 / 从模式
- 帧 SPI 模式
- SPIx 只接收操作
- SPIx 错误处理

18.3.1 8 位与 16 位操作

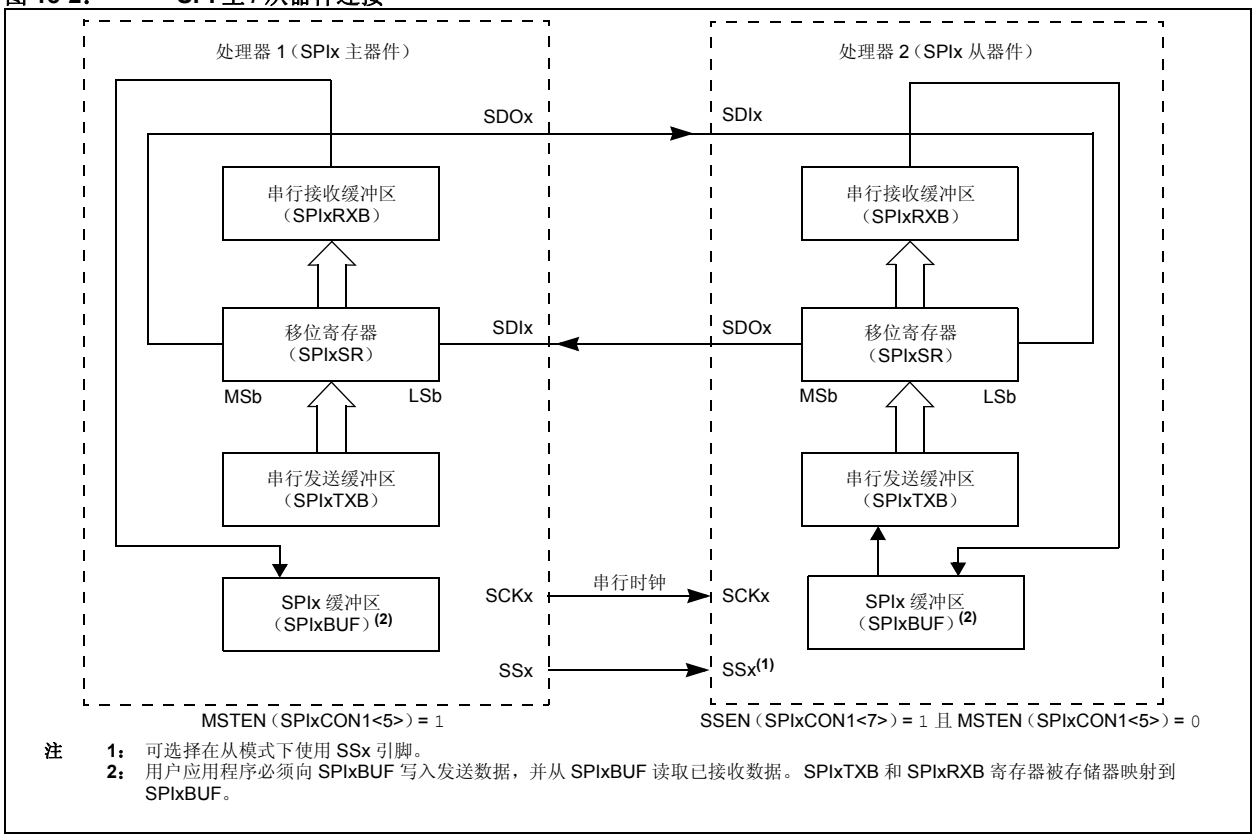
SPIx 控制寄存器 1 中的字 / 字节通信选择控制位 MODE16 （SPIxCON1<10>）允许模块在 8 位或 16 位模式下进行通信。除了接收和发送的位数外，两种模式的功能相同。在本文中：

- 当 MODE16 位的值改变时，模块会被复位。因此，在正常工作期间不应更改该位的值。
- 在 8 位工作模式下，数据是从 SPIx 移位寄存器（SPIxSR）的 bit 7 发送出去的；而在 16 位工作模式下，则是从 bit 15 （SPIxSR<15>）发送出去的。在两种模式下，数据都会移入 bit 0 （SPIxSR<0>）。
- 在 8 位模式下发送或接收数据时，需要在 SCKx 引脚上出现 8 个时钟脉冲以移入 / 移出数据。在 16 位模式下，需要在 SCKx 引脚上出现 16 个时钟脉冲。

18.3.2 主 / 从模式

数据的收发可看作在一个模块的移位寄存器的最高位（MSb）和另一个模块的移位寄存器的最低位 （LSb）之间采取了一个直接通路，然后将数据移到相应的发送或接收缓冲区。配置为主模块的模块根据需要为从器件提供串行时钟和同步信号。图 18-2 给出了主从模块的连接。

图 18-2： SPI 主 / 从器件连接



注 1： 可选择在从模式下使用 SSx 引脚。
2： 用户应用程序必须向 SPIxBUF 写入发送数据，并从 SPIxBUF 读取已接收数据。SPIxTXB 和 SPIxRXB 寄存器被存储器映射到 SPIxBUF。

18.3.2.1 主模式

在主模式下，系统时钟被预分频，然后被用作串行时钟。预分频基于 SPIx 控制寄存器 1 中主预分频比位 PPRE<1:0>（SPIxCON1<1:0>）和辅助预分频比位 SPRE<2:0>（SPIxCON1<4:2>）的设置。串行时钟通过 SCKx 引脚输出到从器件。仅当有待发送数据时才会产生时钟脉冲。更多信息，请参见**第 18.4 节“主模式时钟频率”**。CKP 和 CKE 位决定数据传输发生在时钟脉冲的哪个边沿。

将待发送数据写入 SPIxBUF 寄存器，或从 SPIxBUF 寄存器读取已接收数据。

主模式下 SPIx 模块的工作原理如下所述：

1. 一旦模块被设置为主工作模式并使能，待发送数据就会被写入 SPIxBUF 寄存器。SPIx 状态和控制寄存器中的 SPIx 发送缓冲区满状态位 SPITBF （SPIxSTAT<1>）被置 1。
2. SPIx 发送缓冲区 （SPIxTXB）的内容被移到 SPIx 移位寄存器 （SPIxSR），且模块将 SPITBF 位 （SPIxSTAT<1>）清零。
3. 一组 8/16 个时钟脉冲将 8/16 位发送数据从移位寄存器 SPIxSR 移出到 SDOx 引脚，同时将 SDIx 引脚的数据移入 SPIxSR。
4. 当传输结束时，中断控制器中会发生以下事件：
   a) 中断控制器中的相应中断标志位置 1：
      - 中断标志状态寄存器 0 中的 SPI1IF （IFS0<10>）置 1
      - 中断标志状态寄存器 2 （IFS2）中的 SPI2IF 置 1

      通过将相应的中断允许位置 1 来允许以下中断：
      - 将中断允许控制寄存器 0 中的 SPI1IE （IEC0<10>）置 1 来允许中断
      - 将中断允许控制寄存器 2 中的 SPI2IE （IEC2<1>）置 1 来允许中断

      SPIxIF 标志不会被硬件自动清零。
   b) 当正在进行的发送和接收操作结束时， SPIx 移位寄存器 （SPIxSR）的内容被移到 SPIx 接收缓冲区 （SPIxRXB）。
   c) SPIx 状态和控制寄存器中的 SPIx 接收缓冲区满状态位 SPIRBF（SPIxSTAT<0>）被模块置 1，指示接收缓冲区已满。一旦用户代码读取了 SPIxBUF 寄存器，硬件就会将 SPIRBF 位清零。
5. 当 SPIx 模块需要从 SPIxSR 传输数据到 SPIxRXB 时，如果 SPIRBF 位置 1（接收缓冲区满），模块会将接收溢出标志位 SPIROV（SPIxSTAT<6>）位置 1，指示产生了溢出条件。
6. 只要 SPITBF 位 （SPIxSTAT<1>）清零，用户软件就可以在任何时候将待发送数据写入 SPIxBUF。写操作可以与SPIxSR移出先前写入的数据同时发生，因此可以允许连续发送。

注：	用户应用程序不能直接写入 SPIxSR 寄存器。对 SPIxSR 寄存器的所有写操作均通过 SPIxBUF 寄存器执行。

18.3.2.1.1 主模式设置步骤

要将 SPIx 模块设置为主工作模式，请执行以下步骤：

1. 如果使用中断，需配置中断控制器：
   a) 将中断控制器中相应中断标志状态寄存器的 SPIx 中断标志状态位 SPIxIF（IFS0<10> 或 IFS2<1>）清零。
   b) 将中断控制器中相应中断事件控制寄存器的 SPIx 事件中断允许位 SPIxIE（IEC0<10> 或 IEC2<1>）置 1。
   c) 通过写中断控制器中相应中断优先级控制寄存器的 SPIx 事件中断优先级位 SPIxIP（IPC2<10-8> 或 IPC8<6-4>）来设置中断优先级。
2. 将 SPIxCON1 寄存器中的主模式使能 （MSTEN）位置 1 （SPIxCON1<5> = 1）。
3. 将 SPIxSTAT 寄存器中的接收溢出标志 （SPIROV）位清零 （SPIxSTAT<6> = 0）。
4. 通过将 SPIxSTAT 寄存器中的 SPIx 使能 （SPIEN）位置 1 （SPIxSTAT<15> = 1）使能 SPIx 工作。
5. 将待发送数据写入 SPIxBUF 寄存器。发送（和接收）在数据写入 SPIxBUF 寄存器时立即开始。

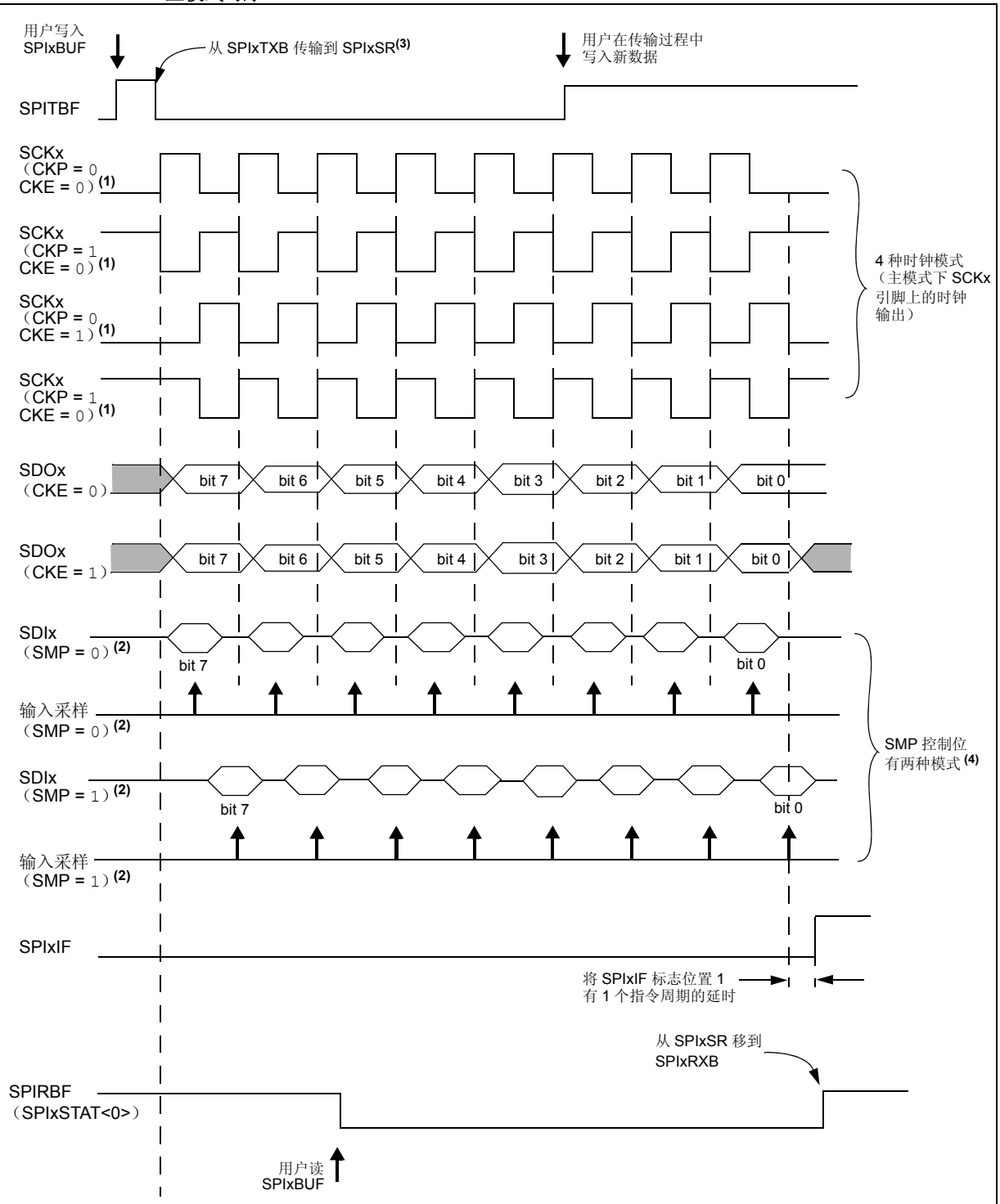
例 18-1 给出了为主模式配置 SPI 寄存器的代码片段。

例 18-1： SPI 配置——主模式

```
/* Following code snippet shows SPI register configuration for MASTER mode*/

IFS0bits.SPI1IF = 0;                  //Clear the Interrupt Flag
IEC0bits.SPI1IE = 0;                  //disable the Interrupt
                                      // SPI1CON1 Register Settings
SPI1CON1bits.DISSCK = 0;              //Internal Serial Clock is Enabled.
SPI1CON1bits.DISSDO = 0;              //SDOx pin is controlled by the module.
SPI1CON1bits.MODE16 = 1;              //Communication is word-wide (16 bits).
SPI1CON1bits.SMP = 0;                 //Input Data is sampled at the middle of data output time.
SPI1CON1bits.CKE = 0;                 //Serial output data changes on transition from
                                      //Idle clock state to active clock state
SPI1CON1bits.CKP = 0;                 //Idle state for clock is a low level;
                                      //active state is a high level
SPI1CON1bits.MSTEN = 1;               //Master Mode Enabled
SPI1STATbits.SPIEN = 1;               //Enable SPI Module
SPI1BUF = 0x0000;                     //Write data to be transmitted
                                      //Interrupt Controller Settings
IFS0bits.SPI1IF = 0;                  //Clear the Interrupt Flag
IEC0bits.SPI1IE = 1;                  //Enable the Interrupt
```

图 18-3：　SPIx 主模式时序



注　**1：** 图中所示 4 种 SPIx 时钟模式仅用于说明 CKP（SPIxCON1<6>）和 CKE（SPIxCON1<8>）位的功能。只能选择 4 种工作模式之一。

2： 图中所示 SMP 位（SPIxCON1<9>）取两个不同值时的 SDIx 和输入采样，仅用于说明目的。在工作期间只能选择 SMP 位的两种配置之一。

3： 如果没有待发送的数据，一旦用户应用程序写入 SPIxBUF， SPIxTXB 中的数据就会传输到 SPIxSR。

4： 图中所示为 8 位模式下的操作。除时钟脉冲数外， 16 位模式下的操作与此类似。

18.3.2.2 从模式

在从模式下，当 SCKx 引脚上出现外部时钟脉冲时发送和接收数据。 SPIx 时钟极性选择位 CKP（SPIxCON<6>）和 SPIx 时钟边沿选择位 CKE （SPIxCON<8>）决定数据传输发生在时钟脉冲的哪个边沿。将待发送数据写入 SPIxBUF 寄存器，或从 SPIxBUF 寄存器读取已接收数据。从模式下模块的其余操作与上述主模式下的操作相同。

注：	在从模式下，如果没有数据写入 SPIxBUF 寄存器，SPI 模式将在 SPI 主器件开始读操作时发送已写入 SPIxBUF 寄存器的上个数据。

18.3.2.2.1 从模式设置步骤

要将 SPIx 模块设置为从工作模式：

1. 将 SPIxBUF 寄存器清零。
2. 如果使用中断，需配置中断控制器：
   a) 将相应中断标志状态寄存器中的 SPIx 中断标志状态位 SPIxIF （中断控制器中的 IFS0<10> 或 IFS2<1>）清零。
   b) 将相应 IECn 寄存器中的 SPIx 事件中断允许 （SPIxIE）位置 1。
   c) 通过写相应 IPCn 寄存器中的 SPIx 事件中断优先级（SPIxIP）位来设置中断优先级。
3. 配置 SPIxCON1 寄存器：
   a) 将主模式使能 （MSTEN）位清零 （SPIxCON1<5> = 0）。
   b) 将数据输入采样阶段 （SMP）位清零 （SPIxCON1<9> = 0）。
   c) 如果时钟边沿选择 （CKE）位置 1，则将从选择使能 （SSEN）位置 1 （SPIxCON1<7> = 1）来使能 $\overline{SSx}$ 引脚。
4. 配置 SPIxSTAT 寄存器：
   a) 将接收溢出标志 （SPIROV）位清零 （SPIxSTAT<6> = 0）。
   b) 通过将 SPIx 使能 （SPIEN）位置 1 （SPIxSTAT<15> = 1）使能 SPIx 工作。

例 18-2 给出了为从模式配置 SPI 寄存器的代码片段。

例 18-2： SPI 配置——从模式

```
/* Following code snippet shows SPI register configuration for SLAVE Mode*/

SPI1BUF = 0;
IFS0bits.SPI1IF = 0;        //Clear the Interrupt Flag
IEC0bits.SPI1IE = 0;        //Disable The Interrupt
                            // SPI1CON1 Register Settings

SPI1CON1bits.DISSCK = 0;    //Internal Serial Clock is Enabled
SPI1CON1bits.DISSDO = 0;    //SDOx pin is controlled by the module
SPI1CON1bits.MODE16 = 1;    //Communication is word-wide (16 bits)
SPI1CON1bits.SMP = 0;       //Input Data is sampled at the middle of data
                            //output time.
SPI1CON1bits.CKE = 0;       //Serial output data changes on transition
                            //from Idle clock state to active clock state
SPI1CON1bits.CKP = 0;       //Idle state for clock is a low level; active
                            //state is a high level
SPI1CON1bits.MSTEN = 0;     //Master Mode disabled
SPI1STATbits.SPIROV=0;      //No Receive Overflow Has Occurred
SPI1STATbits.SPIEN = 1;     //En able SPI Module

                            //Interrupt Controller Settings
IFS0bits.SPI1IF = 0;        //Clear the Interrupt Flag
IEC0bits.SPI1IE = 1;        //Enable The Interrupt
```

18.3.2.2.2 从选择同步

$\overline{SSx}$ 引脚允许同步从模式。如果从选择使能（SSEN）位置 1（SPIxCON1<7> = 1），则仅当 $\overline{SSx}$ 引脚被驱动为低电平状态时才能使能从模式下的发送和接收（见图 18-5）。不能驱动端口输出或其他外设输出，以允许 $\overline{SSx}$ 引脚用作输入。如果 SSEN 位置 1 且 $\overline{SSx}$ 引脚被驱动为高电平，SDOx 引脚将不再被驱动并且将变为三态，即使模块在发送过程中也是如此。

下次 $\overline{SSx}$ 引脚被驱动为低电平时，将使用 SPIxTXB 寄存器中保存的数据重新尝试被中止的发送。如果 SSEN 位未置 1，则 $\overline{SSx}$ 引脚不影响从模式下模块的工作。

注：	为满足模块的时序要求，当 CKE = 1 时，在从模式下必须使能 $\overline{SSx}$ 引脚 （详细信息，请参见图 18-6）。

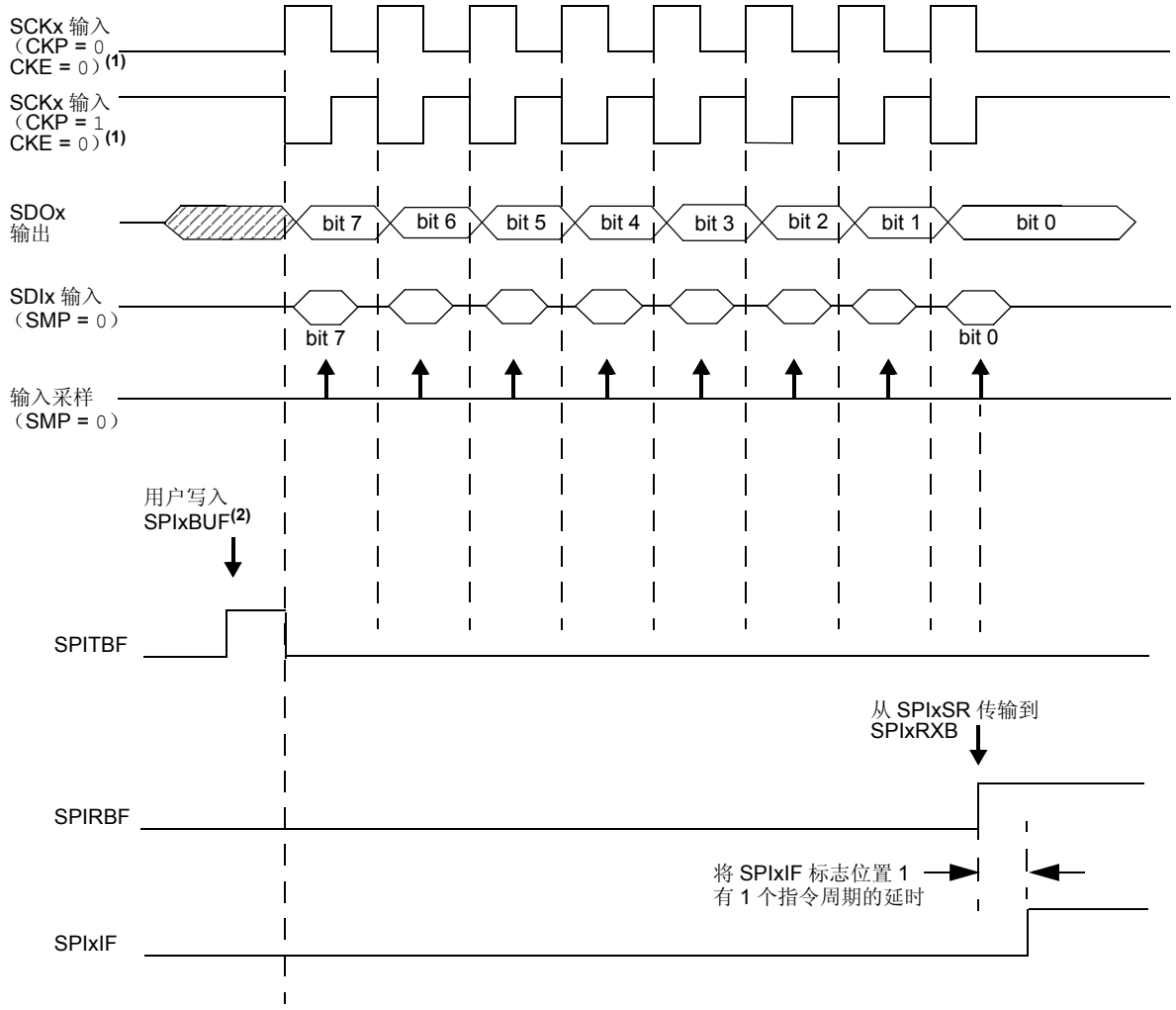
18.3.2.2.3 SPITBF 状态标志的操作

发送缓冲区满状态位 SPITBF （SPIxSTAT<1>）的功能在从模式下与主模式下有所不同。

如果 SSEN 清零 （SPIxCON1<7> = 0），当用户应用程序将数据装入 SPIxBUF 时， SPITBF 位将置 1。它在模块将 SPIxTXB 中的数据传输到 SPIxSR 时清零。这与主模式下 SPITBF 位的功能类似。

如果 SSEN 置 1 （SPIxCON1<7> = 1），当用户应用程序将数据装入 SPIxBUF 时， SPITBF 将置 1。但是，它只有在 SPIx 模块完成数据发送时才被清零。当 $\overline{SSx}$ 引脚变为高电平时，发送将被中止，但是可能在稍后重新尝试发送。每个数据字都保存在 SPIxTXB 中，直到所有位都被发送到接收器中。

图 18-4： SPIx 从模式时序 （禁止从选择引脚） (3)

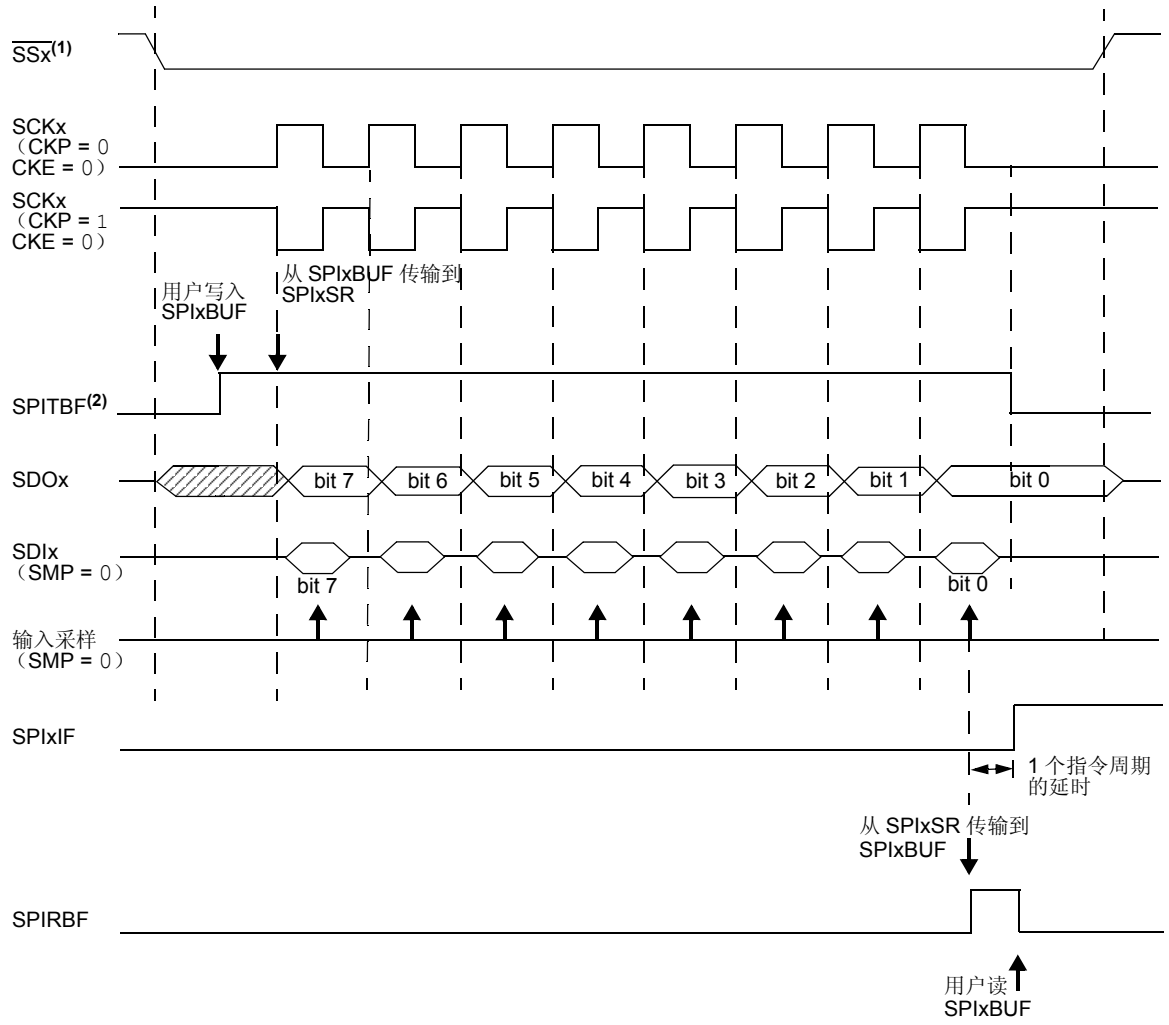


注 1： 图中所示两种 SPIx 时钟模式仅用于说明 CKP （SPIxCON<6>）和 CKE （SPIxCON<8>）位的功能。可选择 CKP 和 CKE 位的任意组合来配置模块的工作。

2： 如果没有待发送的数据或正在进行发送，一旦用户应用程序写入 SPIxBUF，SPIxBUF 中的数据就会传输到 SPIxSR。

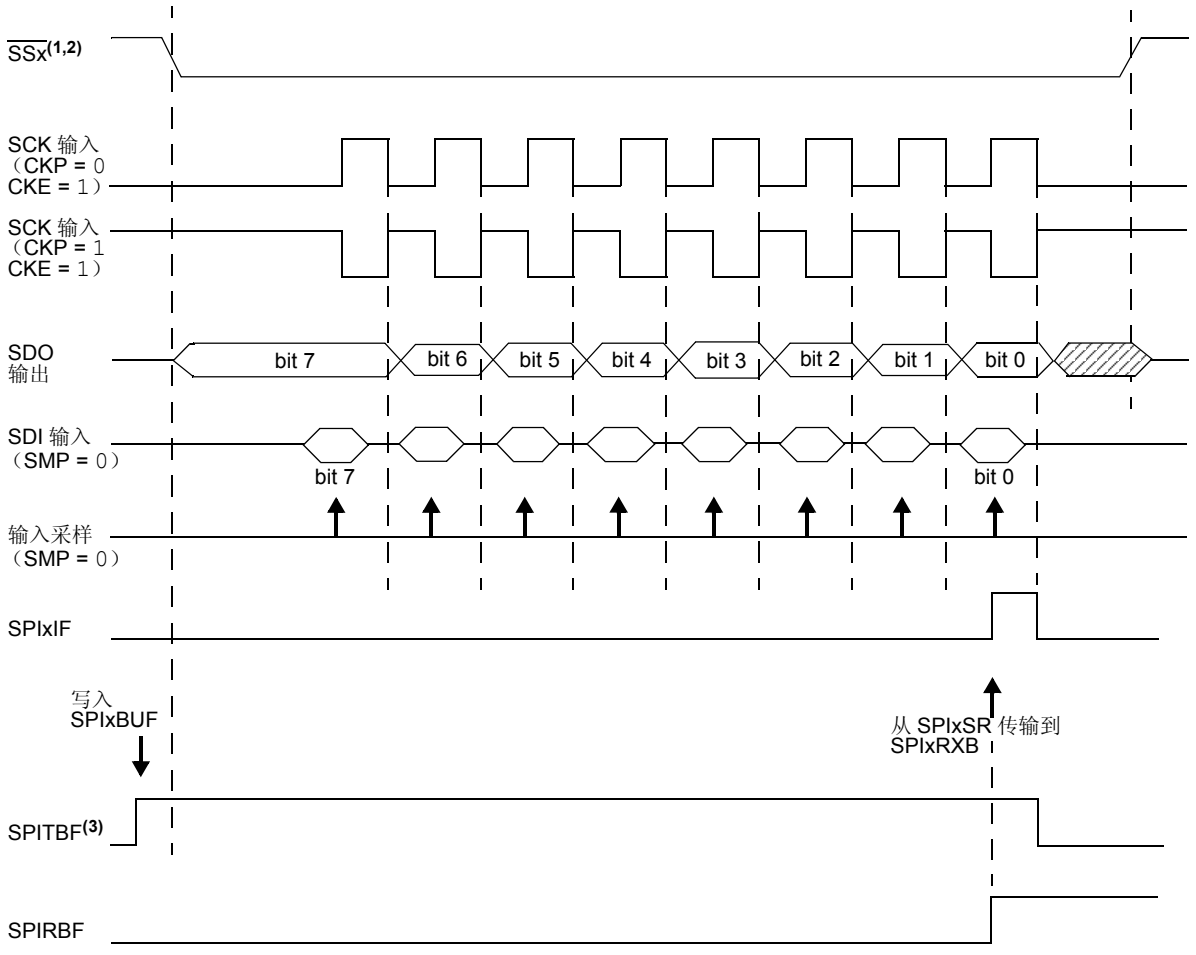
3： 图中所示为 8 位模式下的操作； 16 位模式下的操作与此类似。

图 18-5： SPIx 从模式时序 （使能从选择引脚）(3)



注 1： 当 SSEN （SPIxCON1<7>）位被设为 1 时，要使能从模式下的发送和接收， $\overline{SSx}$ 引脚必须被驱动为低电平。
2： 发送数据保存在 SPIxTXB 中， SPITBF 保持置 1 直到发送完所有位。
3： 图中所示为 8 位模式下的操作； 16 位模式下的操作与此类似。

图 18-6： SPIx 从模式时序 （CKE = 1） (4)



注 1： 当 CKE = 1 时， SSx 引脚必须用于从模式工作。
2： 当 SSEN （SPIxCON1<7>）位被设为 1 时，要使能从模式下的发送和接收， SSx 引脚必须被驱动为低电平。
3： 发送数据保存在 SPIxTXB 中， SPITBF 保持置 1 直到发送完所有位。
4： 图中所示为 8 位模式下的操作； 16 位模式下的操作与此类似。

18.3.3 帧 SPIx 模式

当 SPI 模块工作在主模式或从模式时，模块支持一个基本的帧 SPIx 协议。模块使用 4 个控制位来配置帧 SPIx 操作：

- 帧 SPIx 支持 （FRMEN）位
  FRMEN 位 （SPIxCON2<15>）用于使能帧 SPIx 模式并使 $\overline{SSx}$ 引脚用作帧同步脉冲输入或输出引脚。 SSEN （SPIxCON1<7>）的状态被忽略。
- 帧同步脉冲方向控制 （SPIFSD）位
  SPIFSD 位 （SPIxCON2<14>）决定 $\overline{SSx}$ 引脚是输入还是输出 （即，模块是接收还是产生帧同步脉冲）
- 帧同步脉冲极性 （FRMPOL）位
  FRMPOL （SPIxCON2<13>）用于选择单个 SPIx 数据帧的帧同步脉冲的极性 （高电平有效或低电平有效）
- 帧同步脉冲边沿选择 （FRMDLY）位
  FRMDLY （SPIxCON2<1>）用于选择同步脉冲是与第一个串行时钟脉冲一致还是在其之前出现

在帧主模式下， SPIx 模块产生帧同步脉冲并在 $\overline{SSx}$ 引脚上为其他器件提供此脉冲。

在帧从模式下， SPIx 模块使用 $\overline{SSx}$ 引脚上接收到的帧同步脉冲。

注：	在所有帧 SPIx 模式下必须使用 $\overline{SSx}$ 和 SCKx 引脚。

可同时支持帧 SPIx 模式和非帧主 / 从模式。这使用户可以使用以下 4 种帧 SPIx 配置：

- SPIx 主模式和帧主模式
- SPIx 主模式和帧从模式
- SPIx 从模式和帧主模式
- SPIx 从模式和帧从模式

这些模式决定 SPIx 模块是否产生串行时钟和帧同步脉冲。

当 FRMEN（SPIxCON<14>）= 1 且 MSTEN（SPIxCON<5>）= 1 时， SCKx 引脚变为输出引脚，且 SCKx 上的 SPI 时钟变为自由运行的时钟。

当 FRMEN = 1 且 MSTEN = 0 时，SCKx 引脚变为输入引脚。假设提供给 SCKx 引脚的时钟源是自由运行的时钟。

时钟的极性由 CKP（SPIxCON<6>）位选择。CKE（SPIxCON<8>）位在帧 SPI 模式下不使用，应该由用户软件编程为 0。

当 CKP = 0 时，帧同步脉冲输出和 SDOx 数据输出在 SCKx 引脚上时钟脉冲的上升沿改变。在串行时钟的下降沿，在 SDIx 输入引脚上采样输入数据。

当 CKP = 1 时，帧同步脉冲输出和 SDOx 数据输出在 SCKx 引脚上时钟脉冲的下降沿改变。在串行时钟的上升沿，在 SDIx 输入引脚上采样输入数据。

18.3.3.1 帧主模式和帧从模式

当 SPIFSD（SPIxCON2<14>）= 0 时，SPIx 模块处于帧主工作模式。在该模式下，当用户软件将发送数据写入 SPIxBUF 单元（从而将发送数据写入 SPIxTXB 寄存器）时，模块发出帧同步脉冲。在帧同步脉冲的末尾， SPIxTXB 中的数据被传输到 SPIxSR，同时开始发送 / 接收数据。

当 SPIFSD = 1 时，模块处于帧从模式。在该模式下，帧同步脉冲由外部时钟源产生。当模块采样到帧同步脉冲时，它会将 SPIxTXB 寄存器的内容传输到 SPIxSR，同时开始发送 / 接收数据。在接收到帧同步脉冲前，用户应用程序必须确保 SPIxBUF 中装入了正确的发送数据。

注：	无论数据是否写入 SPIxBUF，接收到帧同步脉冲时都将启动发送。如果没有执行任何写操作，将发送 SPIxTXB 的当前内容。

18.3.3.2 SPIx 主 / 帧主模式

在 SPI 主 / 帧主模式下， SPIx 模块同时产生时钟信号和帧同步信号，如图 18-7 所示。通过将 MSTEN 和 FRMEN 位设为 1 并将 SPIFSD 设为 0 来使能这种配置。

在该模式下，无论模块是否正在进行发送，都将在SCKx引脚连续输出串行时钟。当写入SPIxBUF 时， $\overline{SSx}$ 引脚将在适当的 SCKx 时钟发送边沿被驱动为有效状态 （由 FRMPOL 位确定），并在一个数据帧期间保持有效。如果 FRMDLY 控制位 （SPIxCON2<1>）清零，将在发送数据前产生帧同步脉冲，如图 18-8 所示。如果 FRMDLY 置 1，帧同步脉冲将与数据发送的开始一致，如图 18-9 所示。模块在 SCKx 的下一个发送边沿开始发送数据。

图 18-7： SPIx 主 / 帧主模式连接图

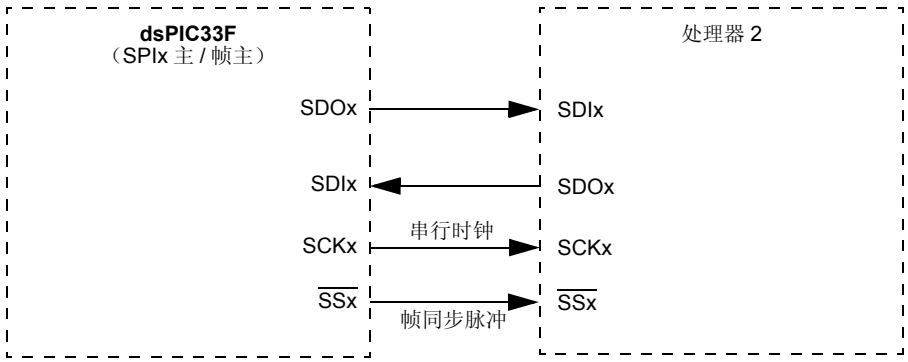


图 18-8： SPIx 主 / 帧主模式时序 （SPIFE = 0）

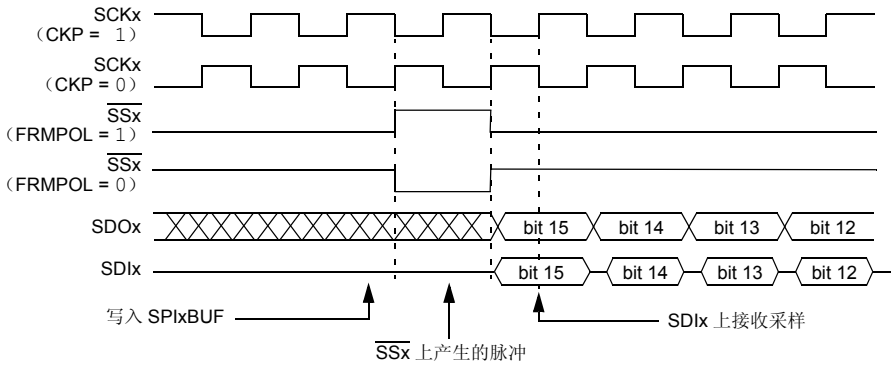
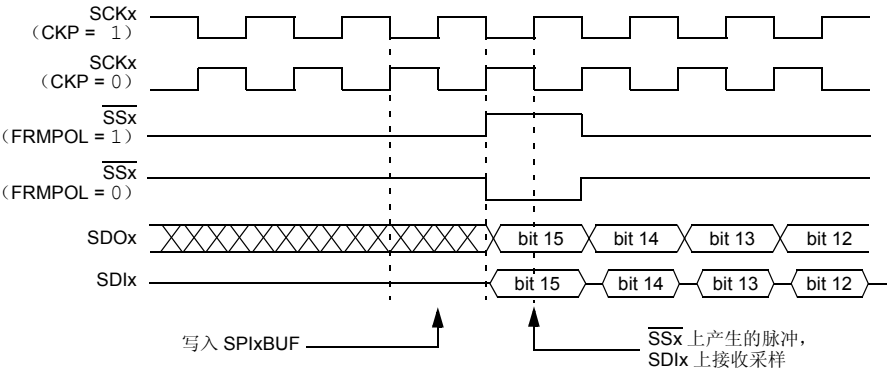


图 18-9： SPIx 主 / 帧主模式时序 （SPIFE = 1）



18.3.3.3 SPIx 主 / 帧从模式

在 SPI 主 / 帧从模式下，模块产生时钟信号，但是使用从模块的帧同步信号进行数据传输 （见图 18-10）。通过将 MSTEN、 FRMEN 和 SPIFSD 位设为 1 来使能该模式。

在该模式下， $\overline{SSx}$ 引脚为输入引脚，并且在 SPIx 时钟的采样边沿对其进行采样。当采样到有效状态时，在后续的 SPIx 时钟发送边沿就会发送数据。当发送完成时，中断标志 SPIxIF 置 1。在 $\overline{SSx}$ 引脚上接收到信号前，用户应用程序必须确保 SPIxBUF 中装入了正确的发送数据。

图 18-10： SPIx 主 / 帧从模式连接图

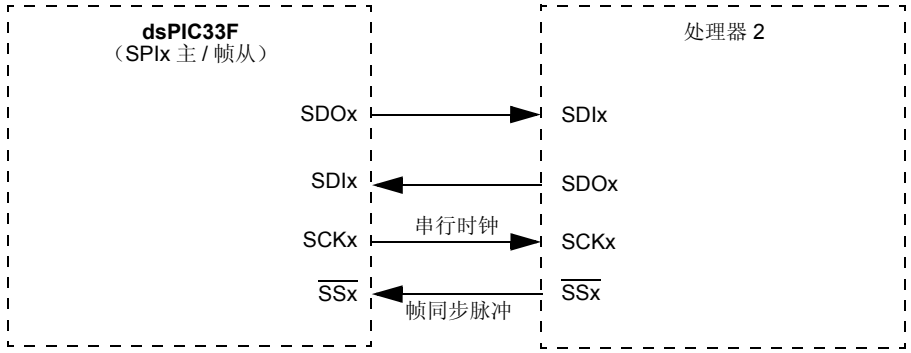


图 18-11： SPIx 主 / 帧从模式时序 （FRMDLY = 0）

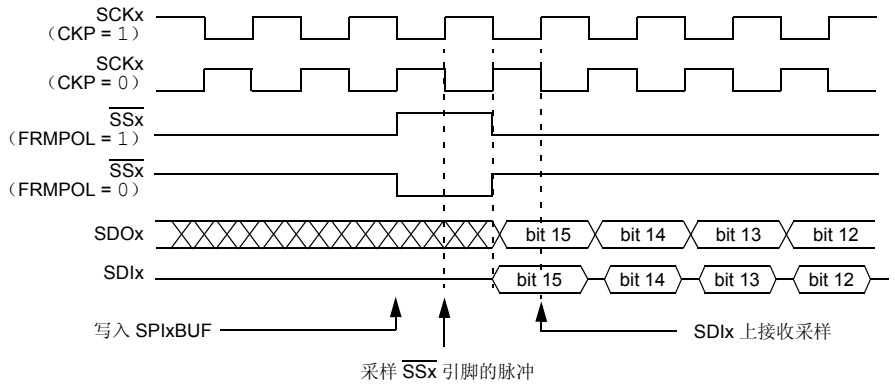
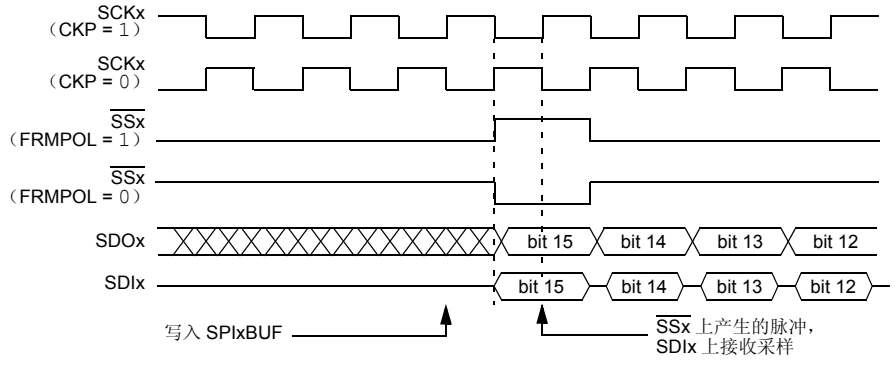


图 18-12： SPIx 主 / 帧从模式时序 （FRMDLY = 1）

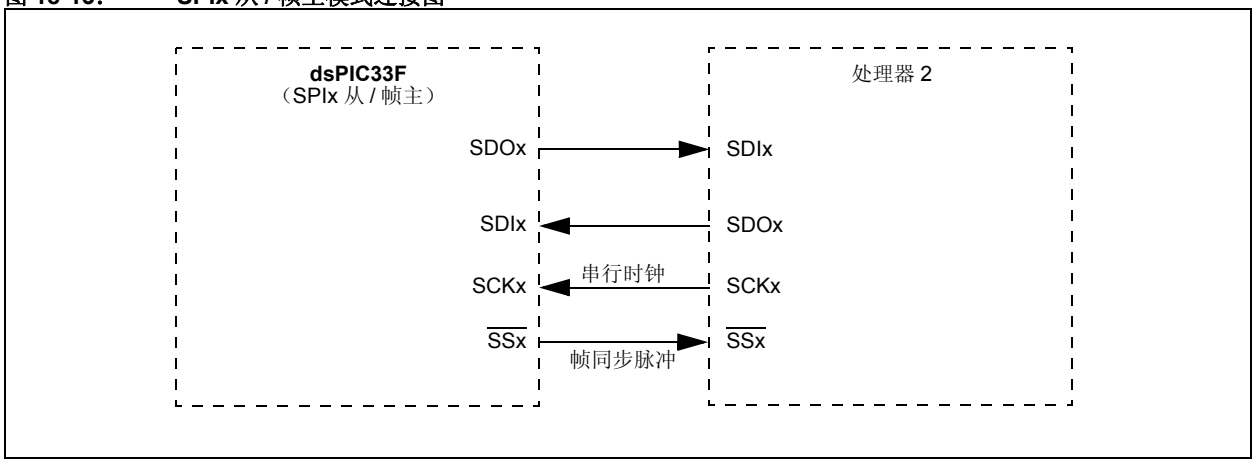


18.3.3.4 SPIx 从 / 帧主模式

在 SPI 从 / 帧主模式下，模块作为 SPIx 从模块工作，从其他 SPIx 模块获取时钟；但是，它将产生帧同步信号来控制数据传输 （见图 18-13）。通过将 MSTEN 位设为 0，将 FRMEN 位设为 1 并将 SPIFSD 位设为 0 来使能该模式。

在从模式下，输入 SPIx 时钟是连续的。当 SPIFSD 位为低电平时，$\overline{SSx}$ 引脚是输出引脚。因此，当写入 SPIxBUF 时，模块在适当的 SPIx 时钟发送边沿将 $\overline{SSx}$ 引脚驱动为有效状态，并保持一个 SPIx 时钟周期。在适当的 SPIx 时钟发送边沿开始发送数据。

图 18-13： SPIx 从 / 帧主模式连接图

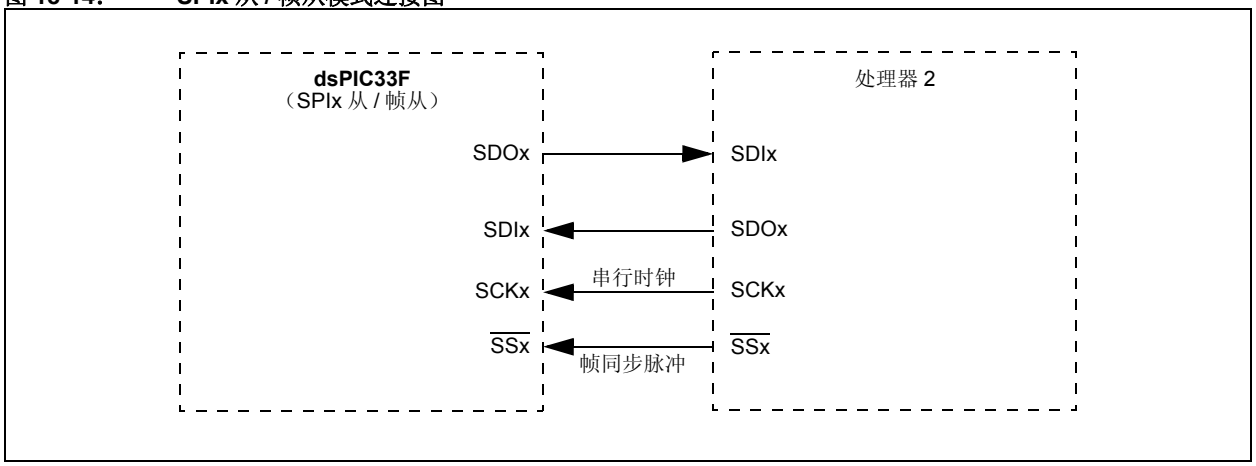


18.3.3.5 SPIx 从 / 帧从模式

在 SPI 从 / 帧从模式下，模块同时从主模块获取时钟信号和帧同步信号 （见图 18-14）。通过将 MSTEN 设为 0，将 FRMEN 设为 1 并将 SPIFSD 设为 1 来使能该模式。

在该模式下， SCKx 和 $\overline{SSx}$ 引脚都是输入引脚。在 SPIx 时钟的采样边沿对 $\overline{SSx}$ 引脚进行采样。当 $\overline{SSx}$ 上采样到有效状态时，将在适当的 SCKx 发送边沿发送数据。

图 18-14： SPIx 从 / 帧从模式连接图



18.3.4 SPIx 只接收操作

将 DISSDO 控制位（SPIxCON1<11>）置 1 将禁止 SDOx 引脚上的发送。这允许将 SPIx 模块配置为只接收工作模式。如果 DISSDO 位置 1，SDOx 引脚将由相应的端口功能控制。DISSDO 功能可应用于所有 SPIx 工作模式。

18.3.5 SPIx 错误处理

如果一个新的数据字已经移入 SPIxSR，但是先前的 SPIxBUF 内容尚未被读取，则 SPIROV 位（SPIxSTAT<6>）将被置 1。 SPIxSR 中接收到的任何数据都不进行传输，同时禁止后续的数据接收，直到 SPIROV 位清零。模块不会自动清零 SPIROV 位；它必须由用户应用程序清零。

当 SPIRBF（SPIxSTAT<0>）或 SPITBF（SPIxSTAT<1>）位置 1 时，SPIx 中断标志（SPIxIF）就会被置 1。中断标志不能由硬件清零。它必须用软件复位。仅当 IECn 控制寄存器中的相应 SPIxIE 位置 1 时，才会产生实际的 SPIx 中断。

此外，当 SPIROV 位置 1 时，SPIx 错误中断标志（SPIxEIF）就会被置 1。该中断标志必须用软件清零。仅当 IECn 控制寄存器中的相应 SPIxEIE 位置 1 时，才会产生实际的 SPIx 错误中断。

18.4 主模式时钟频率

在主模式下，提供给 SPIx 模块的时钟的周期就是指令周期（TCY）。然后此时钟信号由主预分频器（由主预分频比位 PPRE<1:0>（SPIxCON1<1:0>）指定）和辅助预分频器（由辅助预分频比位 SPRE<2:0>（SPIxCON1<4:2>）指定）进行预分频。经过预分频的指令时钟就变为串行时钟，并通过 SCKx 引脚提供给外部器件。

注： SCKx 信号时钟在正常 SPI 模式下不是自由运行的。它仅在 SPIxBUF 装入数据时运行 8 或 16 个脉冲时间；但在帧模式下，它会连续运行。

公式 18-1 可用于根据主预分频比和辅助预分频比设置来计算 SCKx 时钟频率。

公式 18-1： SPI 时钟频率

$$FSCK = \frac{FCY}{\text{主预分频比} * \text{辅助预分频比}}$$

表 18-1 列出了部分 SPIx 时钟频率示例（单位为 kHz）：

注： 并不支持所有时钟频率。更多信息，请参见具体器件数据手册中的 SPIx 时序规范。

表 18-1： SCKx 频率示例 (1)

FCY = 40 MHz		辅助预分频比设置				
		1:1	**2:1**	**4:1**	**6:1**	**8:1**
主预分频比设置	1:1	无效	无效	10000	6666.67	5000
	4:1	10000	5000	2500	1666.67	1250
	16:1	2500	1250	625	416.67	312.50
	64:1	625	312.5	156.25	104.17	78.125
FCY = 5 MHz						
主预分频比设置	1:1	5000	2500	1250	833	625
	4:1	1250	625	313	208	156
	16:1	313	156	78	52	39
	64:1	78	39	20	13	10

注 1： 表中 SCKx 频率的单位为 kHz。

18.5 使用 DMA 的 SPI 操作

在某些 dsPIC33F 器件上， DMA 模块在 CPU 和 SPI 之间传输数据而无需 CPU 协助。请参见具体的 dsPIC33F 器件数据手册，以查看您的特定器件上是否存在 DMA。关于 DMA 模块的更多信息，请参见**第 22 章 “直接存储器访问 （DMA）”**（DS70182）。

如果 DMA 通道与 SPI 接收器关联，则每次数据准备好可从 SPI 移入 RAM 时， SPI 都发出 DMA 请求。DMA 从 SPIxBUF 寄存器向 RAM 传输数据，并在预定义的传输次数之后发出 CPU 中断。

类似地，如果 DMA 通道与 SPI 发送器关联，则 SPI 在每次成功发送后发出 DMA 请求。每次 DMA 请求后，DMA 将新数据传输到 SPIxBUF 寄存器中，并在预定义的传输次数之后发出 CPU 中断。由于 DMA 通道是单向的，如果将 SPI 用于接收和发送，则需要两个 DMA 通道。每个 DMA 通道都必须如表 18-2 中所示进行初始化。

表 18-2： SPI 与 DMA 关联的 DMA 通道寄存器初始化

外设与 DMA 关联	DMAxREQ 寄存器的 IRQSEL<6:0> 位	要从外设读取 / 要写入外设的 DMAxPAD 寄存器值
SPI1TX/RX——SPI1 发送 / 接收	0001010	0x0248 （SPI1BUF）
SPI2TX/RX——SPI2 发送 / 接收	0100001	0x0268 （SPI2BUF）

与 SPI 外设之间的 DMA 传输的启动取决于 SPI 数据方向以及工作于从模式还是主模式。

- **在主模式下仅发送数据：**在该配置中，只有发送第一个 SPI 数据块之后，才会发出 DMA 请求。要启动 DMA 传输，用户应用程序必须先使用 DMA 手动传输模式发送数据，或者必须不依赖于 DMA 先向 SPI 缓冲区中写入数据。
- **在主模式下仅接收数据：**在该配置中，只有接收第一个 SPI 数据块之后，才会发出 DMA 请求。但是，在主模式下，只有 SPI 先发送数据之后才会接收到数据。要启动 DMA 传输，用户应用程序必须使用 DMA 空数据写模式，并启动 DMA 手动传输模式。
- **在主模式下接收和发送数据：**在该配置中，只有接收第一个 SPI 数据块之后，才会发出 DMA 请求。但是，在主模式下，只有 SPI 发送数据之后才会接收到数据。要启动 DMA 传输，用户应用程序必须先使用 DMA 手动传输模式发送数据，或者必须不依赖于 DMA 先向 SPI 缓冲区中写入数据。
- **在从模式下仅发送数据：**在该配置中，只有接收第一个 SPI 数据块之后，才会发出 DMA 请求。要启动 DMA 传输，用户应用程序必须先使用 DMA 手动传输模式发送数据，或者必须不依赖于 DMA 先向 SPI 缓冲区中写入数据。
- **在从模式下仅接收数据：**该配置会在第一个 SPI 数据到达之后立即产生 DMA 请求。用户应用程序不需要执行任何特殊步骤来启动 DMA 传输。
- **在从模式下接收和发送数据：**在该配置中，只有接收第一个 SPI 数据块之后，才会发出 DMA 请求。要启动 DMA 传输，用户应用程序必须先使用 DMA 手动传输模式发送数据，或者必须不依赖于 DMA 先向 SPI 缓冲区中写入数据。

18.5.1 使用 DMA 的 SPI 发送和接收

例 18-3 给出了使用 DMA 进行 SPI 发送和接收的示例。SPI 模块配置为主模式。将使用两个 DMA 通道，通道 0 用于数据发送，通道 1 用于数据接收。

为 SPI 发送配置 DMA 通道 0 时，使用以下参数：

- 从 RAM 向 SPI 连续传输数据
- 带后递增的寄存器间接寻址
- 使用两个乒乓缓冲区
- 每个缓冲区传输 16 次

为 SPI 接收配置 DMA 通道 1 时，使用以下参数：

- 从 SPI 向 RAM 连续传输数据
- 带后递增的寄存器间接寻址
- 使用两个乒乓缓冲区
- 每个缓冲区传输 16 次

例 18-3： 使用 DMA 的 SPI 发送和接收

设置 SPI1 主模式：

```
                              //Interrupt Controller Settings
IFS0bits.SPI1IF = 0;

                              // SPI1CON1 Register Settings
SPI1CON1bits.MODE16 = 1;    //Communication is word-wide (16 bits).
SPI1CON1bits.MSTEN = 1;     //Master Mode Enabled

                              // SPI1CON2 Register Settings
SPI1CON2bits.FRMEN = 0;     // Framed Mode Disabled

                              //SPI1STAT Register Settings
SPI1STATbits.SPISIDL =0;    //Continue module operation in Idle mode
SPI1STATbits.BUFELM =0;     //Buffer Length = 1 Word
SPI1STATbits.SPIROV =0;     //No Receive Overflow Has Occurred
SPI1STATbits.SPIEN = 1;     //Enable SPI Module

                              // Force First word after Enabling SPI
DMA0REQbits.FORCE=1;
while (DMA0REQbits.FORCE==1)

IEC0bits.SPI1IE = 1;
```

设置 DMA 通道 0 在连续乒乓模式下发送数据：

```
unsigned int TxBufferA[16] __attribute__((space(dma)));
unsigned int TxBufferB[16] __attribute__((space(dma)));

IFS0bits.DMA0IF = 0;
IEC0bits.DMA0IE = 1;
DMACS0 = 0;
DMA0CON = 0x2002;
DMA0STA = __builtin_dmaoffset(TxBufferA);
DMA0STB = __builtin_dmaoffset(TxBufferB);
DMA0PAD = (volatile unsigned int) &SPI1BUF;
DMA0CNT =15;
DMA0REQ = 0x000A;
DMA0CONbits.CHEN=1;
```

例 18-3： 使用 DMA 的 SPI 发送和接收 （续）

设置 DMA 通道 1 在连续乒乓模式下接收数据：

```
unsigned int RxBufferA[16] __attribute__((space(dma)));
unsigned int RxBufferB[16] __attribute__((space(dma)));

IFS0bits.DMA1IF =0;
IEC0bits.DMA1IE = 1;
DMA1CON = 0x0002;
DMA1STA = __builtin_dmaoffset(RxBufferA);
DMA1STB = __builtin_dmaoffset(RxBufferB);
DMA1PAD = (volatile unsigned int) &SPI1BUF;
DMA1CNT = 15;
DMA1REQ = 0x000A;
DMA1CONbits.CHEN=1;
```

SPI 和 DMA 中断服务程序：

```
void __attribute__((__interrupt__)) _SPI1Interrupt(void)
{
                        IFS0bits.SPI1IF =0;
}

void __attribute__((__interrupt__)) _DMA0Interrupt(void)
{
static unsigned int BufferCount = 0; // Keep record of which buffer
                                     // contains TX Data
if(BufferCount == 0)
{
TxData(BufferA);                     // Transmit SPI data in
                                     // DMA RAM Primary buffer
}
else
{
TxData(BufferB);                     // Transmit SPI data in
                                     // DMA RAM Secondary buffer
}
BufferCount ^= 1;
IFS0bits.DMA0IF = 0;                 // Clear the DMA0 Interrupt Flag
}

void __attribute__((__interrupt__)) _DMA1Interrupt(void)
{
static unsigned int BufferCount = 0; // Keep record of which buffer
                                     // contains RX Data
if(BufferCount == 0)
{
ProcessRxData(BufferA);              // Process received SPI data in
                                     // DMA RAM Primary buffer
}
else
{
ProcessRxData(BufferB);              // Process received SPI data in
                                     // DMA RAM Secondary buffer
}
BufferCount ^= 1;
IFS0bits.DMA1IF = 0;                 // Clear the DMA1 Interrupt Flag
}
```

18.5.2 空数据写模式下的 SPI 和 DMA

当 SPI 配置为主模式并且只对接收到的数据感兴趣时，必须将一些数据写入 SPI 发送缓冲区，以启动 SPI 时钟并接收外部数据。对于这种情况，请使用 DMA 的空数据写模式。关于 DMA 空数据写模式的更多信息，请参见**第 22 章 “直接存储器访问 （DMA）”**（DS70182）。

18.6 节能模式下的操作

dsPIC33F 系列器件有三种功耗模式：正常 （全功耗）模式和通过 PWRSAV 指令调用的两种节能模式。根据所选择的 SPIx 模式，进入节能模式也可能会影响模块的工作。

18.6.1 休眠模式

当器件进入休眠模式时，系统时钟被禁止。进入休眠模式的结果取决于调用休眠模式时模块被配置为哪种模式 （主模式还是从模式）。

18.6.1.1 主模式工作

以下为将 SPIx 模块配置为主模式工作时进入休眠模式的结果：

- SPIx 模块中的波特率发生器停止并复位
- 在休眠模式下发送器和接收器停止。在唤醒时，发送器或接收器不会继续进行部分完成的传输。
- 如果 SPIx 模块在发送或接收过程中进入休眠模式，则发送或接收将被中止。因为在发送或接收未完成时没有自动方式能防止 SPIx 模块进入休眠模式，所以用户软件必须将进入休眠模式与 SPIx 模块工作同步，以防止传输中止。

18.6.1.2 从模式工作

因为在从模式下， SCKx 的时钟脉冲由外部提供，所以模块在休眠模式下将继续工作。它将在进入休眠模式的转变期间完成所有事务处理。完成事务处理后， SPIRBF 标志置 1。因此， SPIxIF 位将被置 1。

如果允许了 SPIx 中断 （SPIxIE = 1），器件将从休眠模式唤醒。如果 SPIx 中断的优先级大于当前的 CPU 优先级，将从 SPIx 中断向量地址处恢复代码执行。否则，将从先前进入休眠模式时执行的 PWRSAV 指令后的下一条指令开始继续执行代码。如果模块作为从器件工作，则在进入休眠模式时它将不会复位。

当 SPIx 模块进入或退出休眠模式时，寄存器内容不受影响。

18.6.2 空闲模式

当器件进入空闲模式时，系统时钟源继续保持工作。 SPISIDL 位（SPIxSTAT<13>）用于选择在空闲模式下模块是停止工作还是继续工作。

如果 SPISIDL = 1，SPIx 模块将在进入空闲模式时停止通信。其工作方式将和处于休眠模式时相同。如果 SPISID = 0 （默认选择），模块将在空闲模式下继续工作。

18.7 与 SPI 模块相关的特殊功能寄存器

表 18-3： SPI1 寄存器映射

SFR 名称	地址	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	所有复位时的状态
SPI1STAT	0240	SPIEN	—	SPISIDL	—	—	—	—	—	—	SPIROV	—	—	—	—	SPITBF	SPIRBF	0000
SPI1CON1	0242	—	—	—	DISSCK	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE<2:0>			PPRE<1:0>		0000
SPI1CON2	0244	FRMEN	SPIFSD	FRMPOL	—	—	—	—	—	—	—	—	—	—	—	FRMDLY	—	0000
SPI1BUF	0246	SPI1 发送和接收缓冲寄存器																0000

表 18-4： SPI2 寄存器映射

SFR 名称	地址	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	所有复位时的状态
SPI2STAT	0260	SPIEN	—	SPISIDL	—	—	—	—	—	—	SPIROV	—	—	—	—	SPITBF	SPIRBF	0000
SPI2CON1	0262	—	—	—	DISSCK	DISSDO	MODE16	SMP	CKE	SSEN	CKP	MSTEN	SPRE<2:0>			PPRE<1:0>		0000
SPI2CON2	0264	FRMEN	SPIFSD	FRMPOL	—	—	—	—	—	—	—	—	—	—	—	FRMDLY	—	0000
SPI2BUF	0266	SPI2 发送和接收缓冲寄存器																0000

18.8 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 dsPIC33F 器件系列而编写的，但其概念是相近的，通过适当修改并受到一定限制即可使用。当前与串行外设接口（SPI）模块相关的应用笔记有：

标题	应用笔记编号
Interfacing Microchip's MCP41XXX and MCP42XXX Digital Potentiometers to a PIC® Microcontroller	AN746
Interfacing Microchip's MCP3201 Analog-to-Digital Converter to the PIC® Microcontroller	AN719

注： 如需获取更多 dsPIC33F 器件系列的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

18.9 版本历史

版本 A （2007 年 4 月）

这是本文档的初始版本。

版本 B （2008 年 7 月）

该版本包括以下内容更新：

- 寄存器：
  - SPIxCON1：SPIx 控制寄存器 1 （见寄存器 18-2）——增加了注 1。
- 对整篇文档进行了其他少量修正，如语言和格式的更新。

注：