第 9 章：串行通信方式

井艳军

沈阳工业大学电气工程学院

主要内容

计算机串行通信基础

SPI 串行通信模块

I2C 串行通信模块

USART 串行通信模块

计算机串行通信基础

计算机串行通信基础

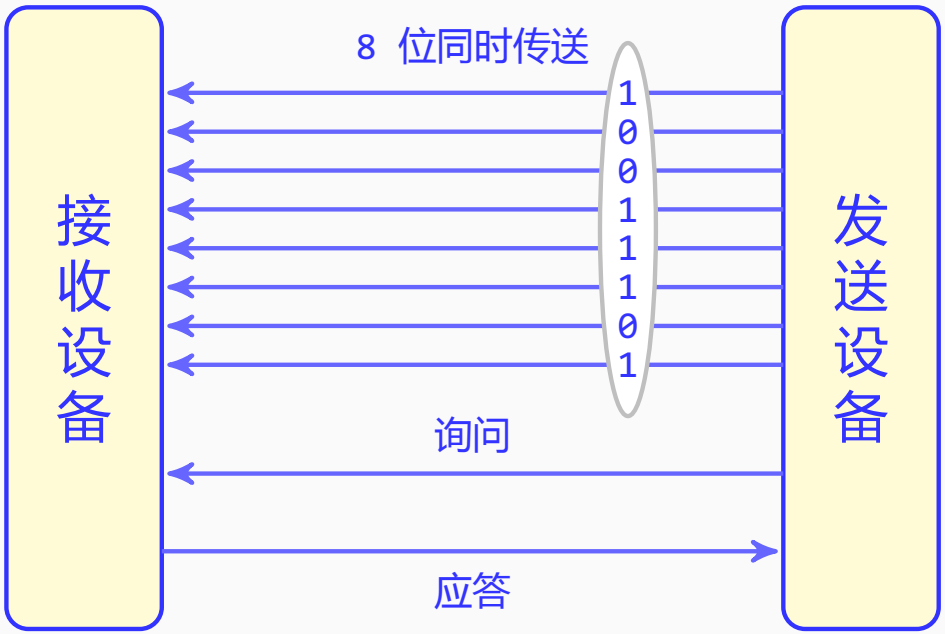
随着多微机系统的广泛应用和计算机网络技术的普及，计算机的通信功能愈来愈显得重要。计算机通信是指计算机与外部设备或计算机与计算机之间的信息交换。

计算机通信是将计算机技术和通信技术的相结合，完成计算机与外部设备或计算机与计算机之间的信息交换。

通信有并行通信和串行通信两种方式。在多微机系统以及现代测控系统中信息的交换多采用串行通信方式。

并行通信

并行通信将数据字节的各位用多条数据线同时进行传送。



特点 :控制简单、传输速度快；由于传输线较多，长距离传送时成本高且接收方的各位同时接收存在困难。

串行通信

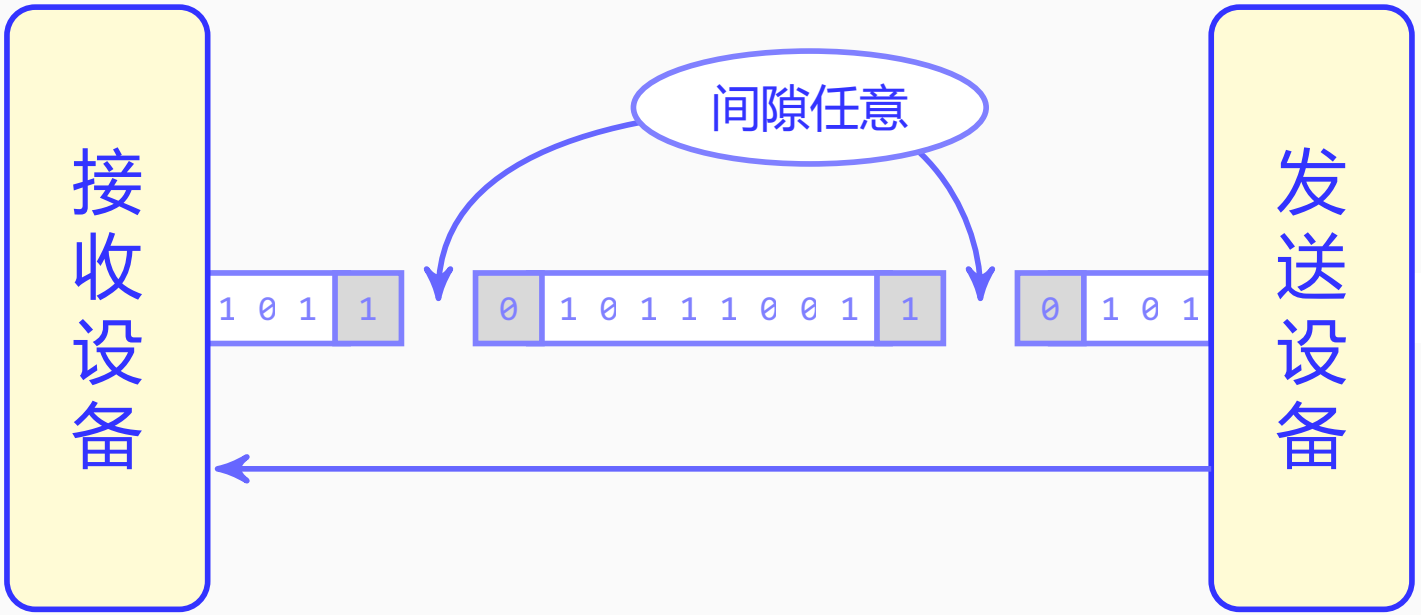
串行通信是将数据字节分成一位一位的形式在一条传输线上逐个地传送。

特点 :传输线少，长距离传送时成本低，且可以利用电话网等现成的设备，但数据的传送控制比并行通信复杂。

分类：异步通信与同步通信

异步通信

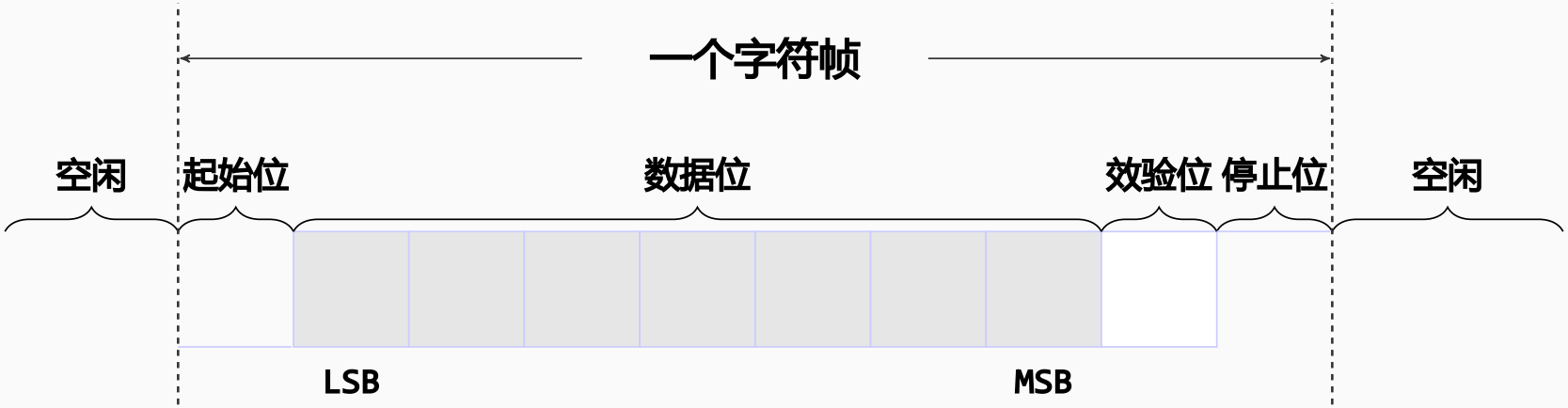
异步通信是指通信的发送与接收设备使用各自的时钟控制数据的发送和接收过程。为使双方的收发协调，要求发送和接收设备的时钟尽可能一致。



异步通信

异步通信是以字符（构成的帧）为单位进行传输，字符与字符之间的间隙（时间间隔）是任意的，但每个字符中的各位是以固定的时间传送的，即字符之间是异步的（字符之间不一定有“位间隔”的整数倍的关系），但同一字符内的各位是同步的（各位之间的距离均为“位间隔”的整数倍）。

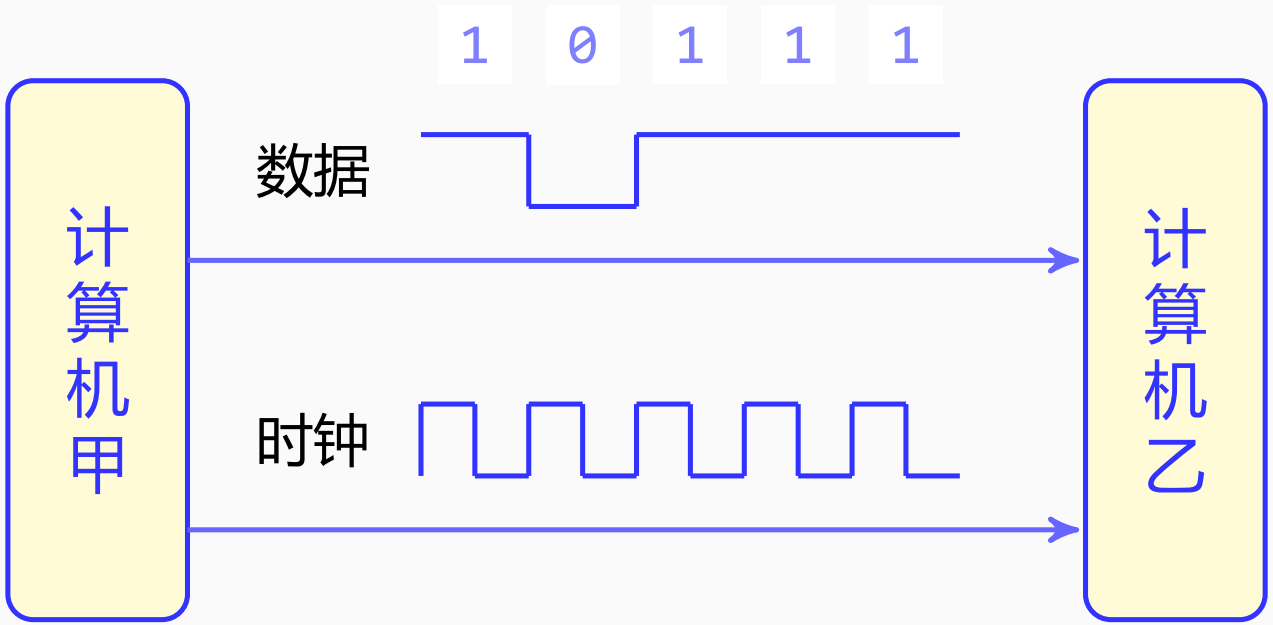
异步通信的数据格式



异步通信的特点：不要求收发双方时钟的严格一致，实现容易，设备开销较小，但每个字符要附加 2～3 位用于起止位，各帧之间还有间隔，因此传输效率不高。

同步通信

同步通信时要建立发送方时钟对接收方时钟的直接控制，使双方达到完全同步。此时，传输数据的位之间的距离均为“位间隔”的整数倍，同时传送的字符间不留间隙，即保持位同步关系，也保持字符同步关系。



面向字符的同步格式

此时，传送的数据和控制信息都必须由规定的字符集（如 ASCII 码）中的字符所组成。图中帧头为 1 个或 2 个同步字符 SYN（ASCII 码为 16H）。SOH 为序始字符（ASCII 码为 01H），表示标题的开始，标题中包含源地址、目标地址和路由指示等信息。STX 为文始字符（ASCII 码为 02H），表示传送的数据块开始。数据块是传送的正文内容，由多个字符组成。数据块后面是组终字符 ETB（ASCII 码为 17H）或文终字符 ETX（ASCII 码为 03H）。然后是校验码。典型的面向字符的同步规程如 IBM 的二进制同步规程 BSC。

面向位的同步格式

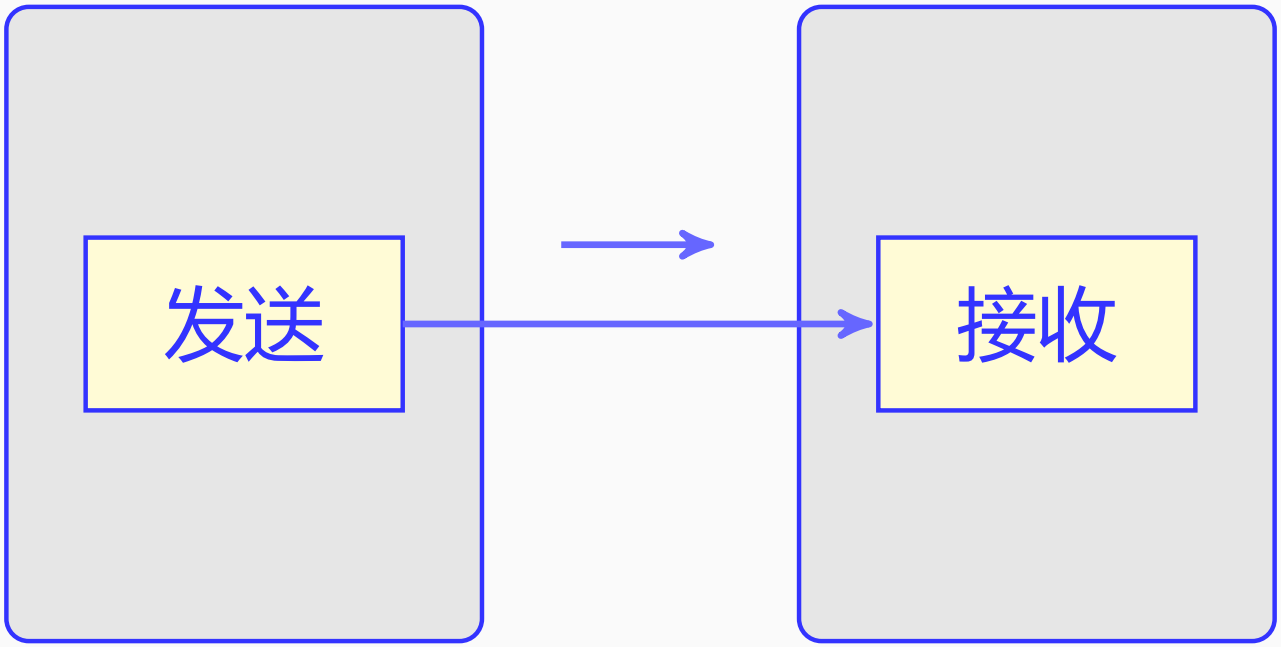
此时，将数据块看作数据流，并用序列 01111110 作为开始和结束标志。为了避免在数据流中出现序列 01111110 时引起的混乱，发送方总是在其发送的数据流中每出现 5 个连续的 1 就插入一个附加的 0；接收方则每检测到 5 个连续的 1 并且其后有一个 0 时，就删除该 0。

同步通信的特点是以特定的位组合“01111110”作为帧的开始和结束标志，所传输的一帧数据可以是任意位。所以传输的效率较高，但实现的硬件设备比异步通信复杂。

串行通信的传输方向

1、单工

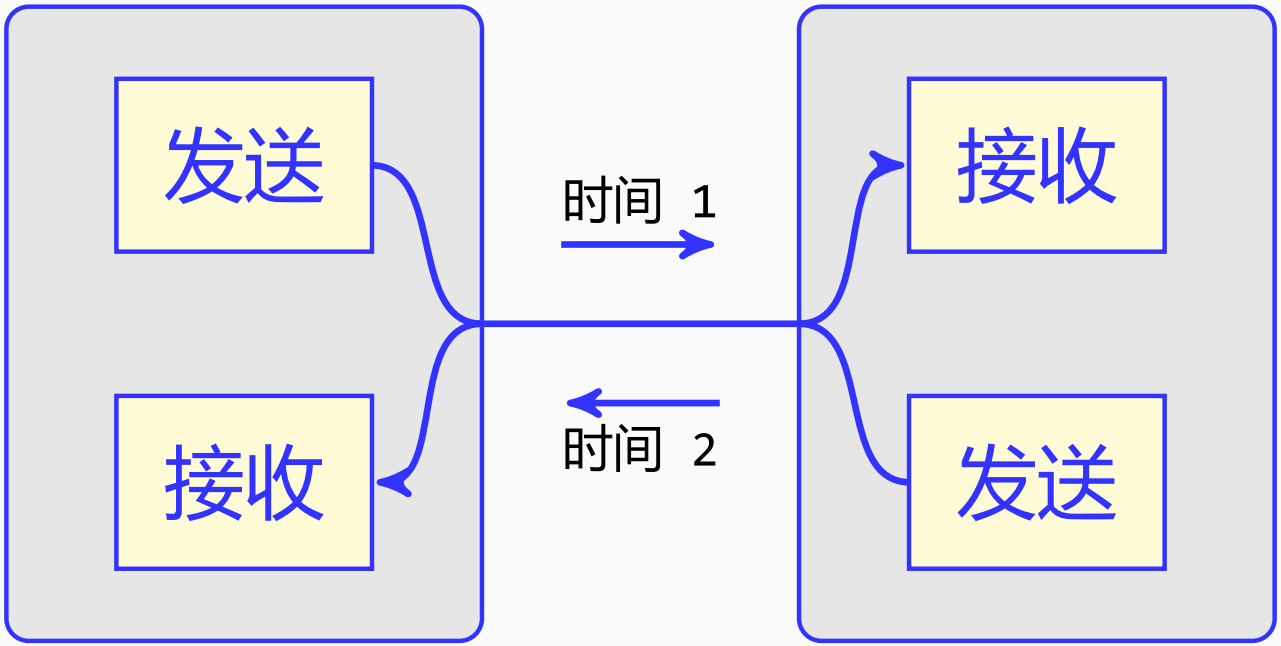
单工是指数据传输仅能沿一个方向，不能实现反向传输。



串行通信的传输方向

2、半双工

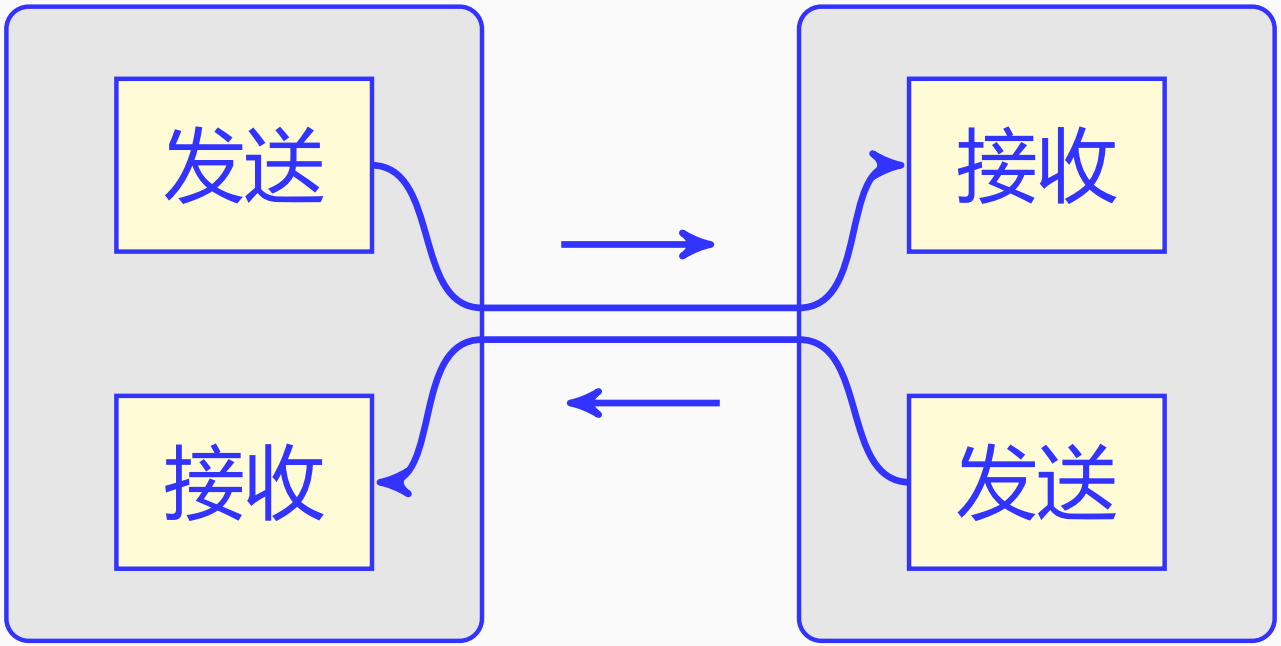
半双工是指数据传输可以沿两个方向，但需要分时进行。



串行通信的传输方向

3、全双工

全双工是指数据可以同时进行双向传输。



串行通信的错误校验

1、奇偶校验

在发送数据时，数据位尾随的 1 位为奇偶校验位（1 或 0）。奇校验时，数据中“1”的个数与校验位“1”的个数之和应为奇数；偶校验时，数据中“1”的个数与校验位“1”的个数之和应为偶数。接收字符时，对“1”的个数进行校验，若发现不一致，则说明传输数据过程中出现了差错

串行通信的错误校验

2、代码和校验

代码和校验是发送方将所发数据块求和（或各字节异或），产生一个字节的校验字符（校验和）附加到数据块末尾。接收方接收数据同时对数据块（除校验字节外）求和（或各字节异或），将所得的结果与发送方的“校验和”进行比较，相符则无差错，否则即认为传送过程中出现了差错。

串行通信的错误校验

3、循环冗余校验

这种校验是通过某种数学运算实现有效信息与校验位之间的循环校验，常用于对磁盘信息的传输、存储区的完整性校验等。这种校验方法纠错能力强，广泛应用于同步通信中。

串行通信的传输速率

波特率

在串行通信中，用“波特率”来描述数据的传输速率。所谓波特率，即每秒钟传送的二进制位数，其单位为 bps （ bits per second ）。它是衡量串行数据速度快慢的重要指标。有时也用“位周期”来表示传输速率，位周期是波特率的倒数。国际上规定了一个标准波特率系列：110 、300 、600 、1200 、1800 、2400 、4800 、9600 、14.4Kbps 、19.2Kbps 、28.8Kbps 、33.6Kbps 、56Kbps 。例如：9600bps ，指每秒传送 9600 位，包含字符的数位和其它必须的数位，如奇偶校验位等。

PIC877 串行通信模块

串行扩展通信接口是单片机与其它计算机之间进行数据交换的重要渠道，F877 单片机主要配置有 2 种形式的串行通信模块：

主控同步串行通信 MSSP（Master Synchronous Serial Port）

通用同步/异步收发器 USART（Universal Synchronous / Asynchronous Receiver Transmitter ）。

MSSP 模块

MSSP 模块主要应用于系统内部近距离的串行通信扩展，如 SPI、I2C 模式。USART 模块主要应用于系统之间的远距离串行通信，在外围接口电路及计算机通信中应用相当广泛。

SPI 串行通信模块

SPI 串行通信模块

SPI（Serial Peripheral Interface）是一种单片机外设芯片同步串行扩展接口，由摩托罗拉公司推出。采用 SPI 接口外围器件的特点是引脚性价比高等优点，因而在市场上得到了广泛的应用。

SPI 总线

SPI 总线是串行外围设备接口, 是一种高速的，全双工，同步的通信总线，并且在芯片的 SPI 的通信原理很简单，它以主从方式工作，通常有一个主设备和一个或多个从设备，需要至少 4 根线。管脚上只占用四根线.

SDO –主设备数据输出，从设备数据输入

SDI –主设备数据输入，从设备数据输出

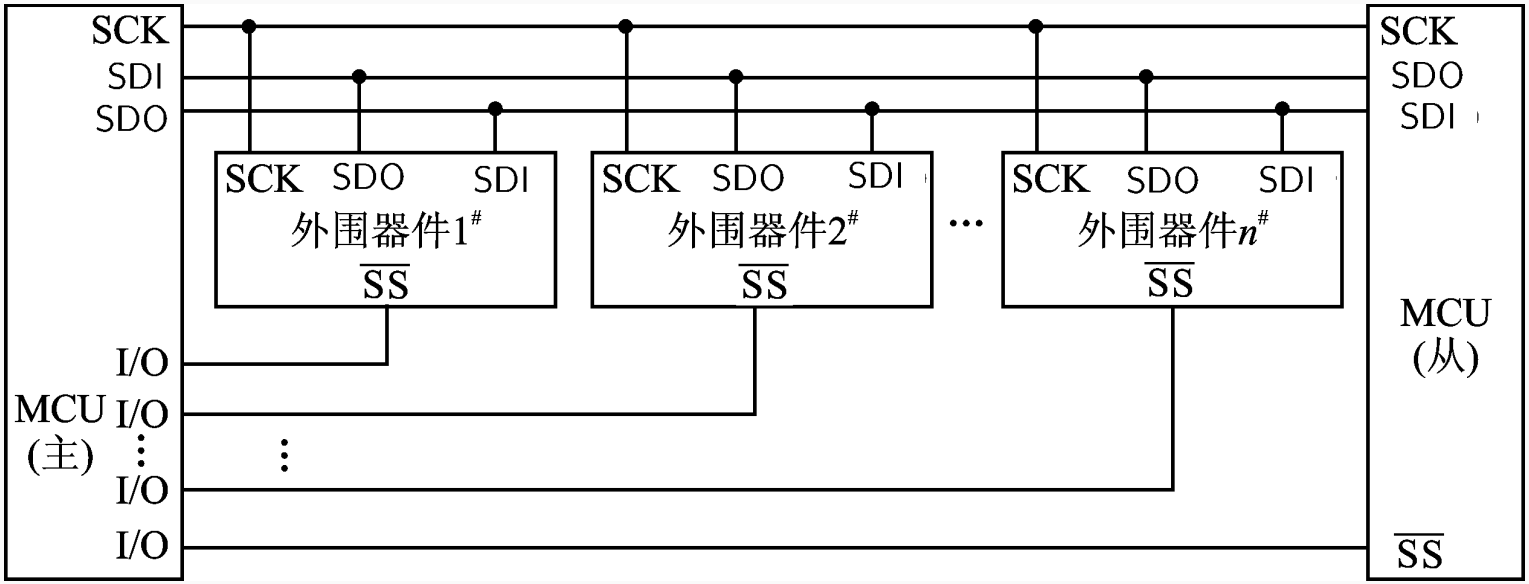
SCK –用来为数据通信提供同步时钟信号，由主设备产生

$\overline{SS}$ –从设备使能信号，由主设备控制

SPI 接口是全双工、同步、串口、单主机。

SPI 总线

SPI 总线添加从器件：每个从器件需要一个单独的从选择信号。总信号数最终为 n+3 个，其中 n 是总线上从器件的数量。在 SPI 总线上添加新的从器件也不方便。对于额外添加的每个从器件，都需要一条新的从器件选择线



SPI 总线

SPI 总线在一次数据传输过程中，接口上只能有一个主机和一个从机能够通信。并且，主机总是向从机发送一个字节数据，而从机也总是向主机发送一个字节数据。

在 SPI 传输中，数据是同步进行发送和接收的。

数据传输的时钟基于来自主处理器的时钟脉冲，

当 SPI 接口上有多个 SPI 接口的单片机时，应区别其主从地位，在某一时刻只能由一个单片机为主器件。

SPI 总线

从器件只能在主机发命令时，才能接收或向主机传送数据。

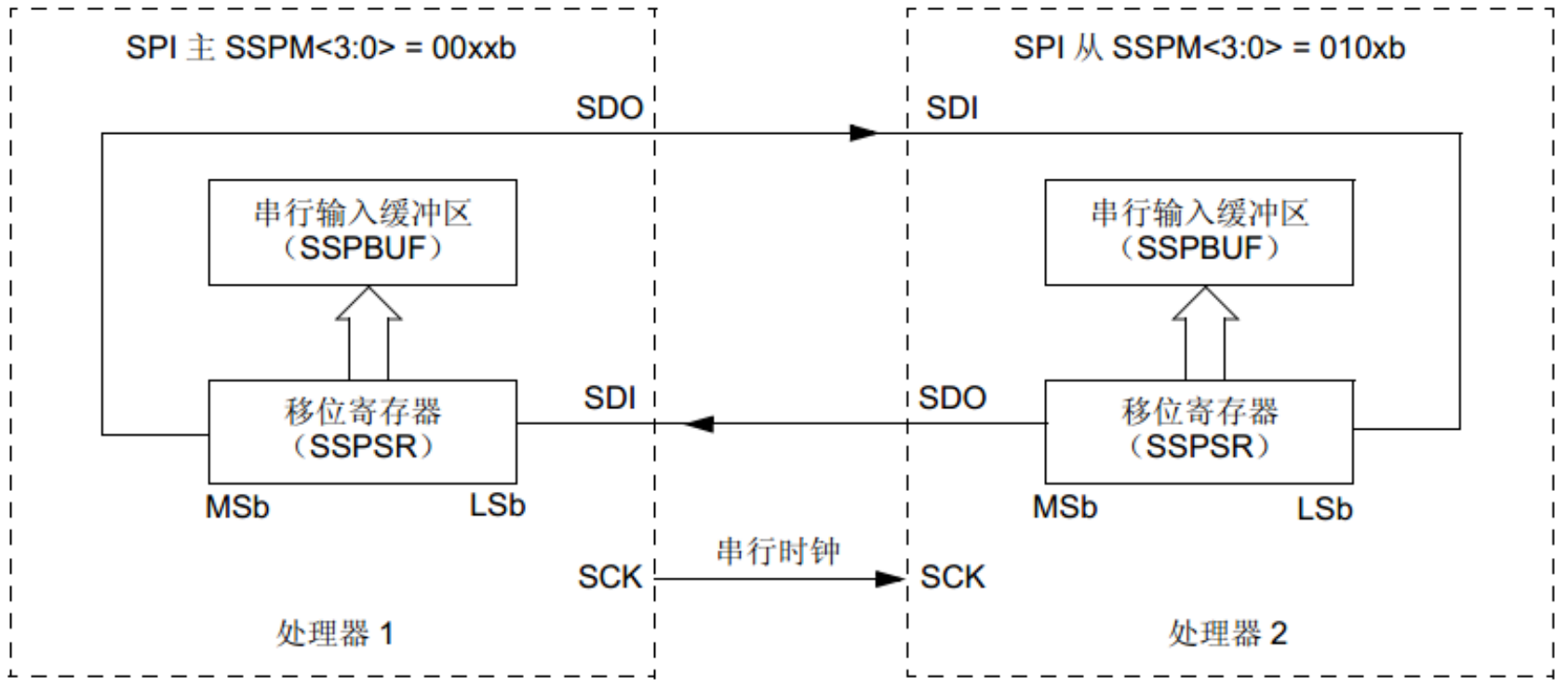
其数据的传输格式是高位（MSB）在前，低位 (LSB）在

SPI 接口的一个缺点：没有应答机制确认是否接收到数据。

如果只是进行写操作，主机只需忽略收到的字节；反过来，如果主机要读取外设的一个字节，就必须发送一个空字节来引发从机的传输。

SPI 总线

SPI 接口实际上是两个简单的移位寄存器，传输的数据为 8 位，在主器件产生的从器件使能信号和移位脉冲下，按位传输。



可以将 SSPBUF 比作生产线上运送包的工具，SSPSR 比作打包解包的工具，SDO 和 SDI 组成的环路比作传送带。但是这里的包是指数据字节，而传送带传送的是数据位，完成一个循环周期两个 SSBUF 数据发生交换。

SPI 总线

上面的上表示上升沿、下表示下降沿，sdi、sdo 相对于主机。

脉冲	主机 sbuff	从机 sbuff	sdi	sdo
0	1 0 1 0 1 0 1 0	0 1 0 1 0 1 0 1	0	0
1 上	0 1 0 1 0 1 0 x	1 0 1 0 1 0 1 x	0	1
1 下	0 1 0 1 0 1 0 0	1 0 1 0 1 0 1 1	0	1
2 上	1 0 1 0 1 0 0 x	0 1 0 1 0 1 1 x	1	0
2 下	1 0 1 0 1 0 0 1	0 1 0 1 0 1 1 0	1	0
3 上	0 1 0 1 0 0 1 x	1 0 1 0 1 1 0 x	0	1
3 下	0 1 0 1 0 0 1 0	1 0 1 0 1 1 0 1	0	1

SPI 总线

脉冲	主机 sbuff	从机 sbuff	sdi	sdo
4 上	1 0 1 0 0 1 0 x	0 1 0 1 1 0 1 x	1	0
4 下	1 0 1 0 0 1 0 1	0 1 0 1 1 0 1 0	1	0
5 上	0 1 0 0 1 0 1 x	1 0 1 1 0 1 0 x	0	1
5 下	0 1 0 0 1 0 1 0	1 0 1 1 0 1 0 1	0	1
6 上	1 0 0 1 0 1 0 x	0 1 1 0 1 0 1 x	1	0
6 下	1 0 0 1 0 1 0 1	0 1 1 0 1 0 1 0	1	0
7 上	0 0 1 0 1 0 1 x	1 1 0 1 0 1 0 x	0	1
7 下	0 0 1 0 1 0 1 0	1 1 0 1 0 1 0 1	0	1
8 上	0 1 0 1 0 1 0 x	1 0 1 0 1 0 1 x	1	0
8 下	0 1 0 1 0 1 0 1	1 0 1 0 1 0 1 0	1	0

SPI 模式下相关寄存器

在 SPI 模式下，有关的寄存器共有 10 个，其中无编址的只有一个 SSPSR。这 10 个寄存器中有 6 个寄存器是与其它模块共用的。另外有 4 个寄存器与 MSSP 模块相关，它们是与 I2C 模式共用的。

INTCON	GIE/GIEH	PEIE/GIEL	T0IE	INTE	RBIE	T0IF	INTF	RBIF
PIE1	—	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE
PIR1	—	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF
SSPBUF	同步串行端口接收缓冲器 / 发送寄存器							
SSPCON	WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0
SSPSTAT	SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF
TRISA	TRISA7	TRISA6	TRISA5	TRISA4	TRISA3	TRISA2	TRISA1	TRISA0
TRISC	TRISC7	TRISC6	TRISC5	TRISC4	TRISC3	TRISC2	TRISC1	TRISC0

SSPBUF（收/发数据缓冲器）

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
MSSP接收/发送数据缓冲空间							

SSPSTAT（同步串口状态寄存器）

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
SMP	CKE	D/A	P	S	R/W	UA	BF

SSPSTAT（同步串口状态寄存器）

Bit0/BF: 缓冲器满标志位，被动参数。仅仅用于 SPI 接收状态下：

0：缓冲器空；

1：缓冲器满。

SSPSTAT（同步串口状态寄存器）

Bit6/CKE：SPI 时钟沿选择和 I2C 总线输入电平选择位。

在 CKP = 0，静态电平为低时：

0：SCK 的下降沿发送数据；

1：SCK 的上升沿发送数据。

在 CKP = 1, 静态电平为高时：

0：SCK 的上升沿发送数据；

1：SCK 的下降沿发送数据。

SSPSTAT（同步串口状态寄存器）

Bit7/SMP：SPI 采样控制位兼 I2C 总线转换率控制位。

在 SPI 主控方式下：

0：在输出数据的中间采样输入数据；

1：在输出数据的末尾采样输入数据。

注意：在 SPI 从动方式下，SMP 位必须置位。

SSPCON（同步串口控制寄存器）

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0

SSPCON（同步串口控制寄存器）

SSPM3-SSPM0	SPI 工作方式	时钟
0 0 0 0	主控方式	fosc/4
0 0 0 1	主控方式	fosc/16
0 0 1 0	主控方式	fosc/64
0 0 1 1	主控方式	TMR2 输出/2
0 1 0 0	从动方式	SCK 脚输入，使能 SS 引脚功能
0 1 0 1	从动方式	SCK 脚输入，关闭 SS 引脚功能，SS 被用作普通数字 I/O 引脚

SSPCON（同步串口控制寄存器）

Bit4/CPK：时钟极性选择位。

0：表示空闲时时钟停留在低电平；

1：表示空闲时时钟停留在高电平。

Bit5/SSPEN：同步串口 MSSP 使能位。

在 SPI 模式下时，有关引脚必须正确的设定为输入或输出状态。

0：关闭串行端口功能，且设定 SCK、SOD、SDI 和 SS 为普通数字 I/O 脚；

1：允许串行端口工作，且设定 SCK、SOD、SDI 和 SS 为 SPI 接口专用。

SSPCON（同步串口控制寄存器）

Bit6/SSPOV：接收溢出标志位，被动参数。

0：未发生接收溢出；

1：发生接收溢出。

注意：所指的接收溢出是缓冲器 SSPBUF 中数据还未取出时，移位寄存器 SSPSR 中又收到新的数据，原 SSPSR 中的数据丢失。

SSPCON（同步串口控制寄存器）

Bit7/WCOL：写操作冲突检测位，被动参数。

在 SPI 从动方式下：

0：未发生冲突；

1：发生冲突。

注意：当 WCOL=1, 正在发送前一个数据时，又有新数据写入 SSPBUF，必须用软件予以清零。

SSPSR 移位寄存器

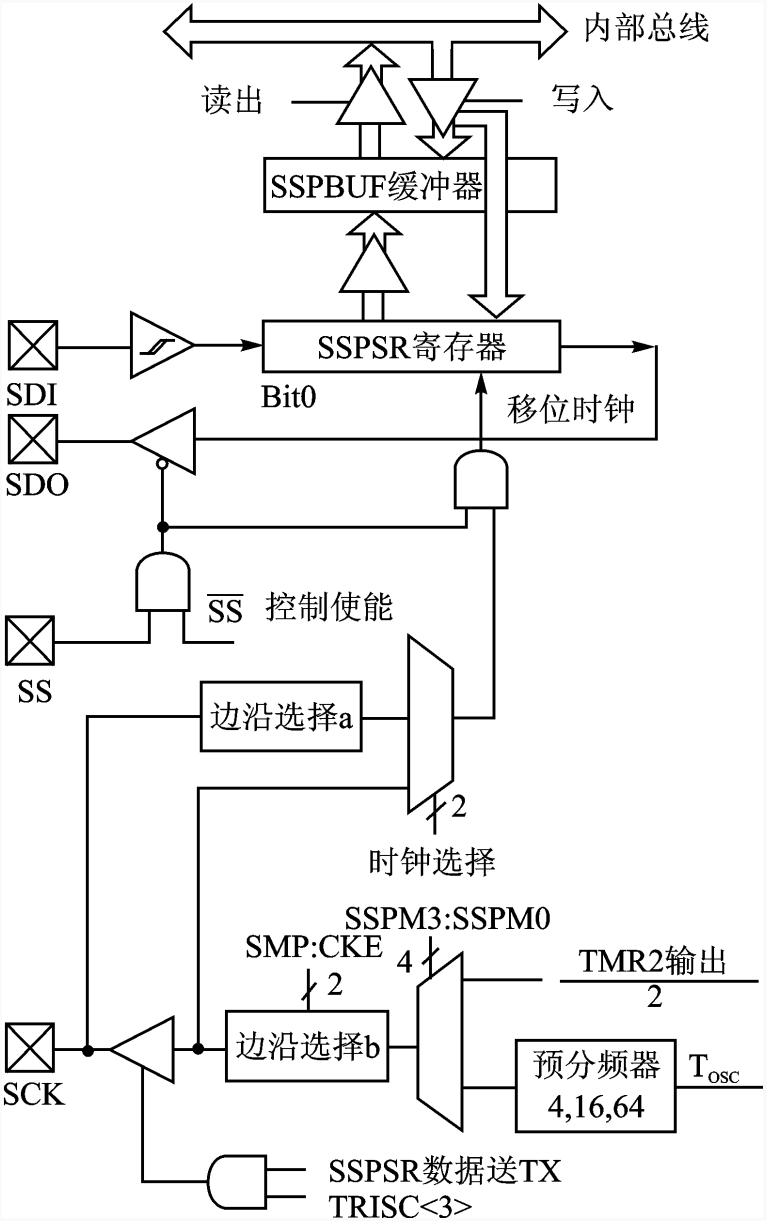
直接从端口引脚接收或发送数据，将已经成功接收到的数据送到缓冲器 SSPBUF 中，或者从缓冲器 SSPBUF 读取将发送的数据。

bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0
MSSP接收/发送数据串行移位空间							

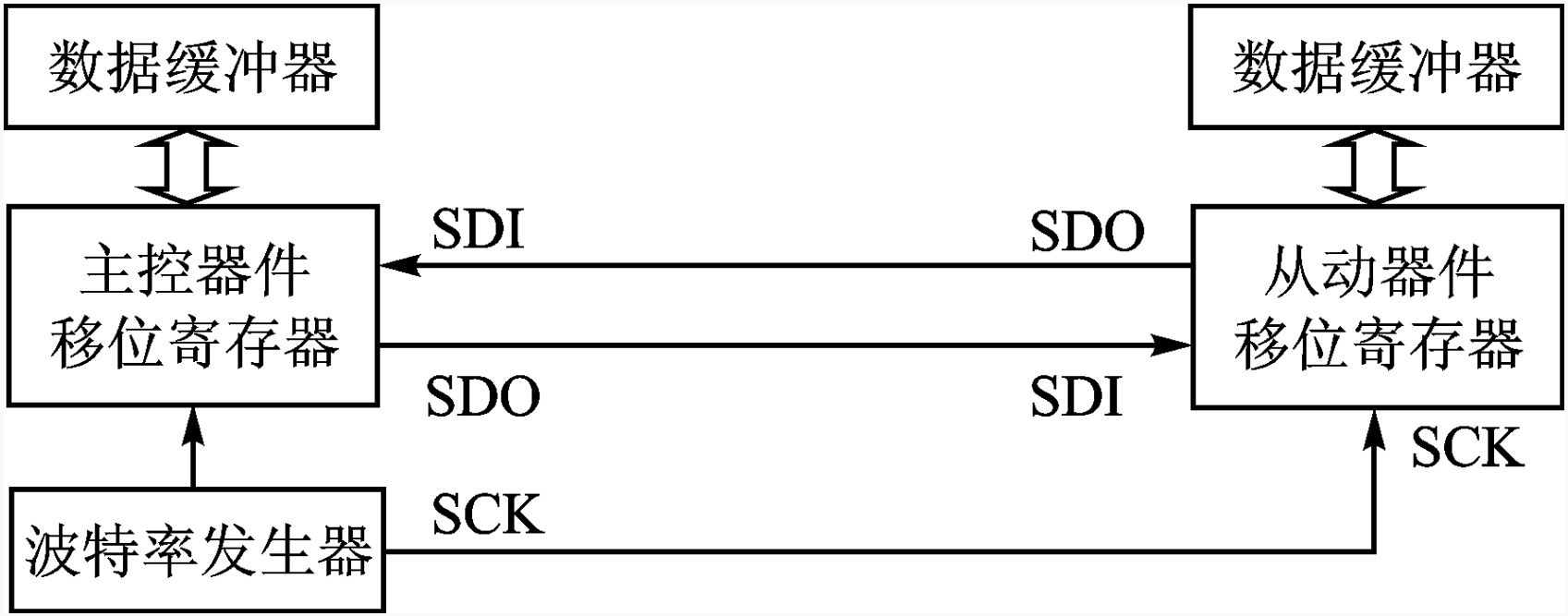
SPI 模式工作原理

要发送的数据通过数据总线送入发送缓冲器，然后自动传送到移位寄存器中；移位寄存器接收到数据自动传送到接收缓冲器，然后由程序读取收到的数据；移位寄存器有移入和移出两个端口，分别与收和发两条通信线路连接，负责收发数据。

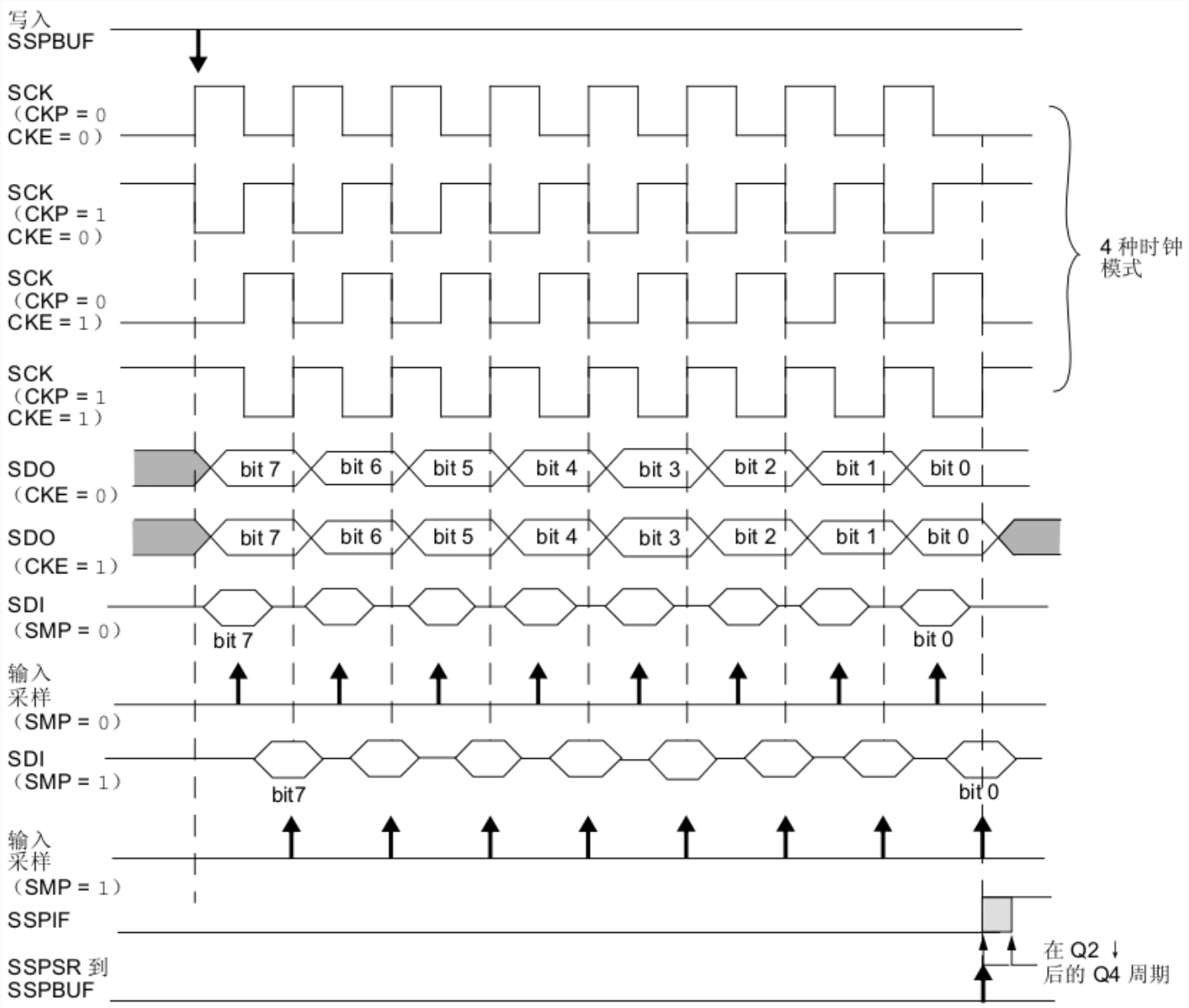
SPI 模式电路的基本结构



SPI 通讯方式结构框图



SPI 时序图 (主控模式)

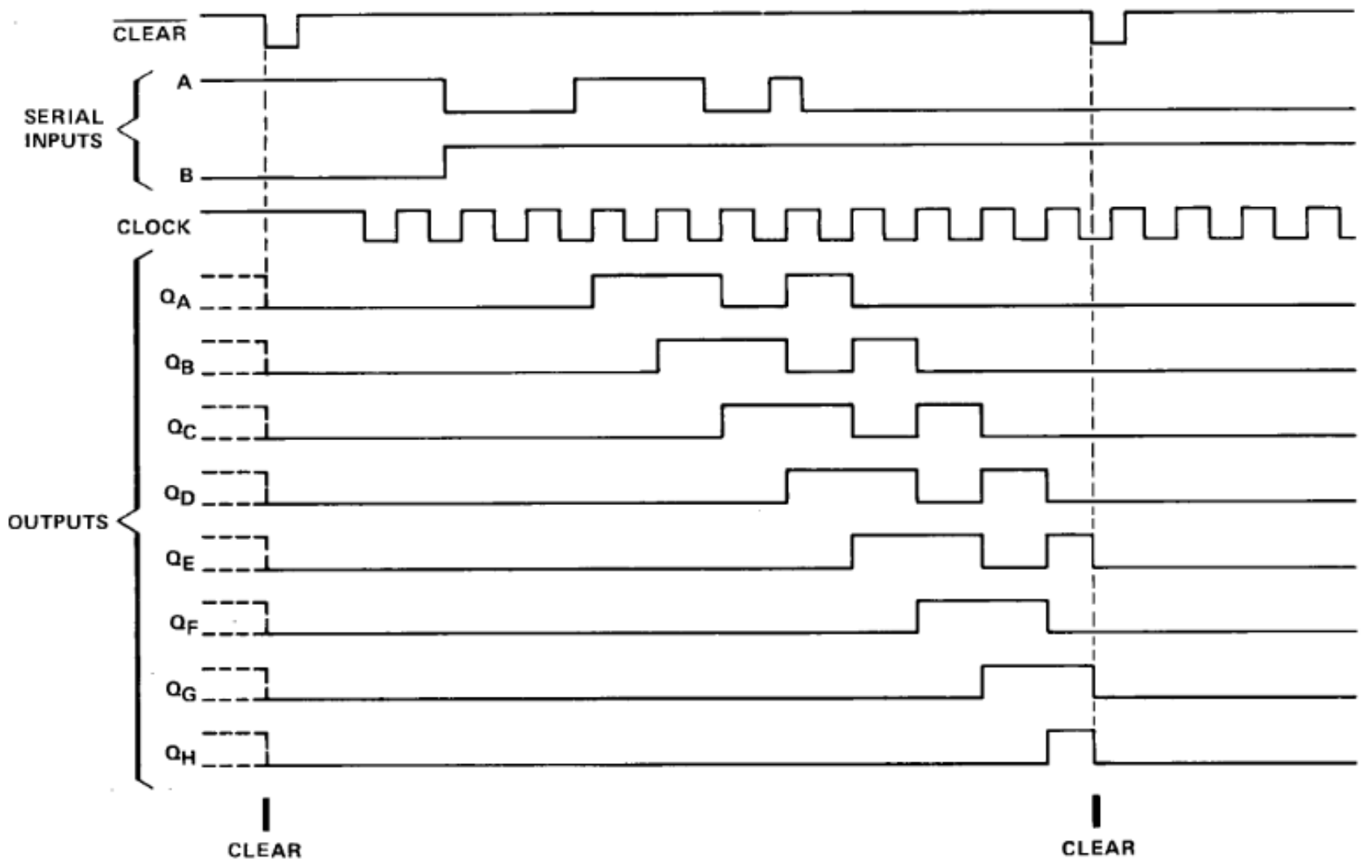

写入
SSPBUF
SCK
（CKP = 0
CKE = 0）
SCK
（CKP = 1
CKE = 0）
SCK
（CKP = 0
CKE = 1）
SCK
（CKP = 1
CKE = 1）
4 种时钟
模式
SDO
（CKE = 0）
SDO
（CKE = 1）
bit 7
bit 6
bit 5
bit 4
bit 3
bit 2
bit 1
bit 0
SDI
（SMP = 0）
输入
采样
（SMP = 0）
SDI
（SMP = 1）
输入
采样
（SMP = 1）
SSPIF
SSPSR 到
SSPBUF
在 Q2 ↓
后的 Q4 周期

应用实例

例题 9-1

利用 SPI 同步串行功能实现数码管数据串行传送，并通过 8 个 73LS164 组成的移位电路并行输出，达到数码数据的静态驱动显示。要求：在系统复位后，8 位数码管全暗，接着数码管 0～7 分别从最高位到最低位依次点亮，最后直接进入系统的监控状态，以在最高位出现“-”为标志。

74LS164 移位寄存器



程序头

```
1  LIST      P=16F877
2  INCLUDE   "P16F877.INC"
3  COUNTER   EQU     30H
4  FSR_TEP   EQU     31H
5            ORG     0000H
6            NOP
7  ST        BSF     STATUS, RP0
8            MOVLW   B'11010111'
9            MOVWF   TRISC
10           CLRF    SSPSTAT
11           BCF     STATUS, RP0
12           MOVLW   B'00110010'
13           MOVWF   SSPCON
14           CALL    CHUSHIHUA
15           GOTO    $
```

显示驱动程序

```
1  ;-------------------------------------------------------
2  ;  显示驱动程序
3  ;-------------------------------------------------------
4  XSHI        MOVLW     67H
5              MOVWF     FSR
6  LOOP        MOVF      INDF, W     ;  取出数据
7              CALL      BMA         ;  查询编码
8              CALL      OUTXSH      ;  SPI输出
9              DECF      FSR
10             BTFSS     FSR, 4
11             GOTO      LOOP
12             RETURN
```

SPI 方式输出编码数据子程序

```
1  ;-------------------------------------------------
2  ; SPI方式输出编码数据子程序
3  ;-------------------------------------------------
4  OUTXSH     MOVWF     SSPBUF
5  LOOP1      BSF       STATUS, RP0
6             BTFSS     SSPSTAT, BF
7             GOTO      LOOP1
8             BCF       STATUS, PR0
9             MOVF      SSPBUF, W
10            RETURN
```

编码查询

```
;-------------------------------------------------
; 编码查询
;-------------------------------------------------
BMA         ADDWF     PCL, F
            RETLW     3FH   ; "0"编码
            RETLW     06H   ; "1"编码
            RETLW     5BH   ; "2"编码
            RETLW     4FH   ; "3"编码
            RETLW     66H   ; "4"编码
            RETLW     6DH   ; "5"编码
            RETLW     7DH   ; "6"编码
            RETLW     07H   ; "7"编码
            RETLW     7FH   ; "8"编码
            RETLW     6FH   ; "9"编码
            RETLW     00H   ; "暗"编码
            RETLW     40H   ; "—"编码
```

键控提示符

```
;-------------------------------------------------
; 8位数码管全暗，仅最高位给出“-”键控提示符
;-------------------------------------------------
JKZT        MOVLW     60H
            MOVWF     FSR
TUN         MOVLW     0AH
            MOVWF     INDF
            INCF      FSR
            BTFSS     FSR, 3
            GOTO      TUN
            MOVLW     0BH
            MOVWF     67H
            RETURN
```

初始化

```
1  ;————————————————————————————————————————
2  ; 初始化子程序(67H—60H存储器赋值00—07)
3  ; 从数码最高位67H开始点亮，延时196ms
4  ;————————————————————————————————————————
5  CHUSHIHUA CALL     JKZT
6            CALL     XSHI
7            MOVLW    67H
8            MOVWF    FSR
9            MOVLW    00H
10           MOVWF    COUNTER
```

初始化

```
1  QT          MOVF      COUNTER, W
2              MOVWF     INDF
3              MOVF      FSR, W
4              MOVWF     FSR_TEP
5              CALL      XSHI
6              CALL      DELAY
7              MOVF      FSR_TEP, W
8              MOVWF     FSR
9              DECF      FSR
10             INCF      COUNTER
11             BTFSS     COUNTER, 3
12             GOTO      QT
13             CALL      JKZT
14             CALL      XSHI
15             RETURN
```

延时子程序

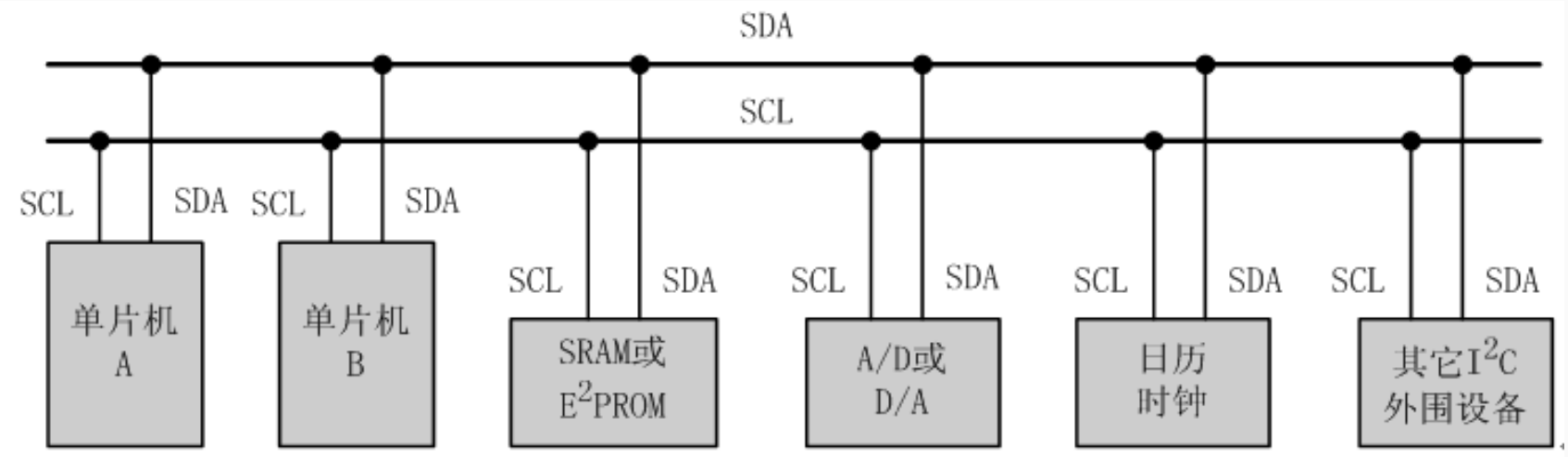
```
1  ;-------------------------------------------------
2  ; 196ms软件延时子程序
3  ;-------------------------------------------------
4  DELAY      MOVLW    01H
5             MOVWF    20H
6  LP1        MOVLW    0FFH
7             MOVWF    21H
8  LP2        MOVLW    0FFH
9             MOVWF    22H
10 LP3        DECFSZ   22H
11            GOTO     LP3
12            DECFSZ   21H
13            GOTO     LP2
14            DECFSZ   20H
15            GOTO     LP1
16            RETURN
```

I2C 串行通信模块

I2C 串行总线概述

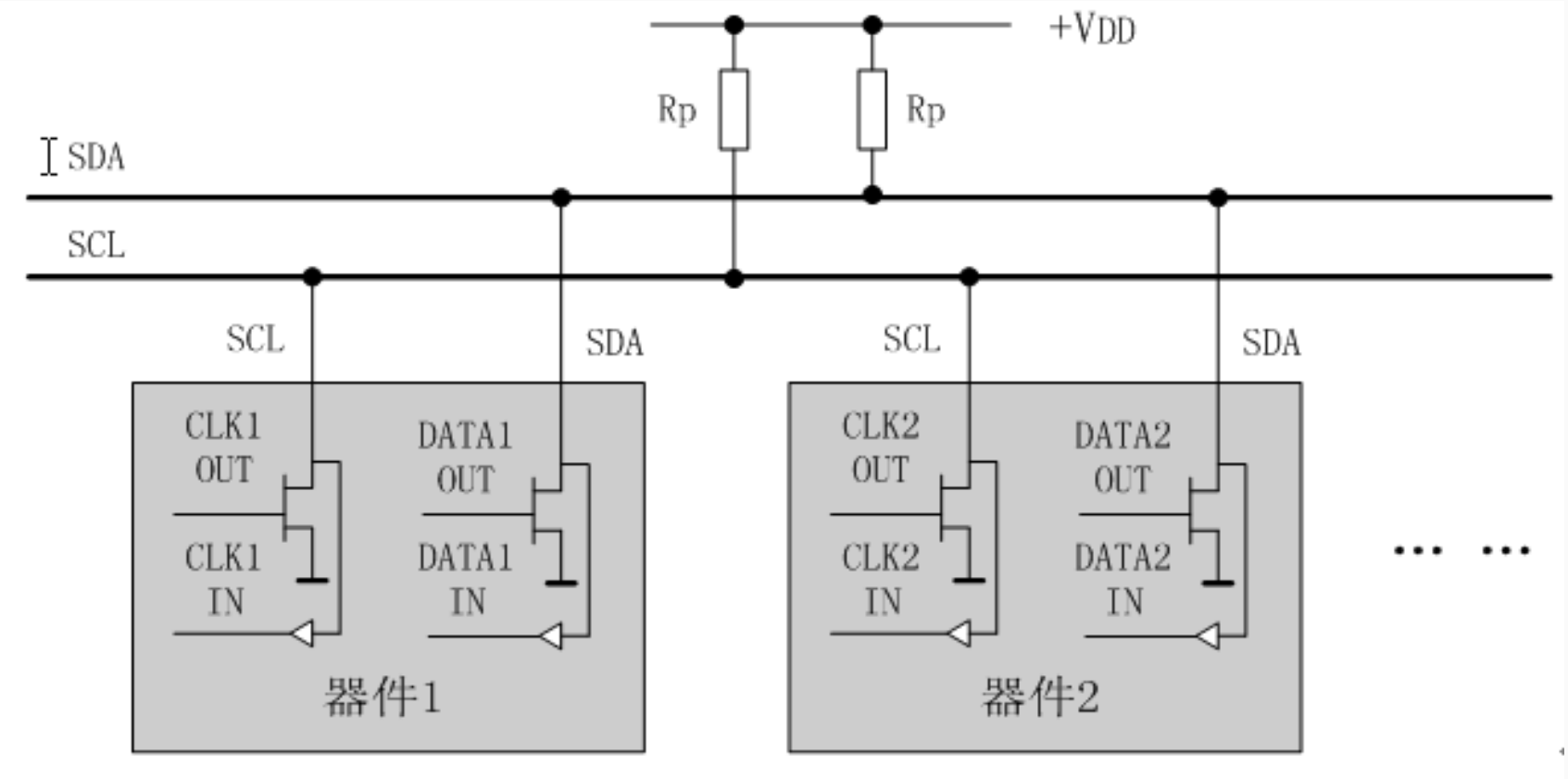
I2C(Inter Integrated Ciruit Bus) 总线是 PHLIPS 公司推出的一种串行总线，是具备多主机系统所需的包括总线裁决和高低速器件同步功能的高性能串行总线。

I2C 总线只有两根双向信号线。一根是数据线 SDA，另一根是时钟线 SCL。



I2C 串行总线概述

I2C 总线通过上拉电阻接正电源。当总线空闲时，两根线均为高电平。连到总线上的任一器件输出的低电平，都将使总线的信号变低，即各器件的 SDA 及 SCL 都是线“与”关系。



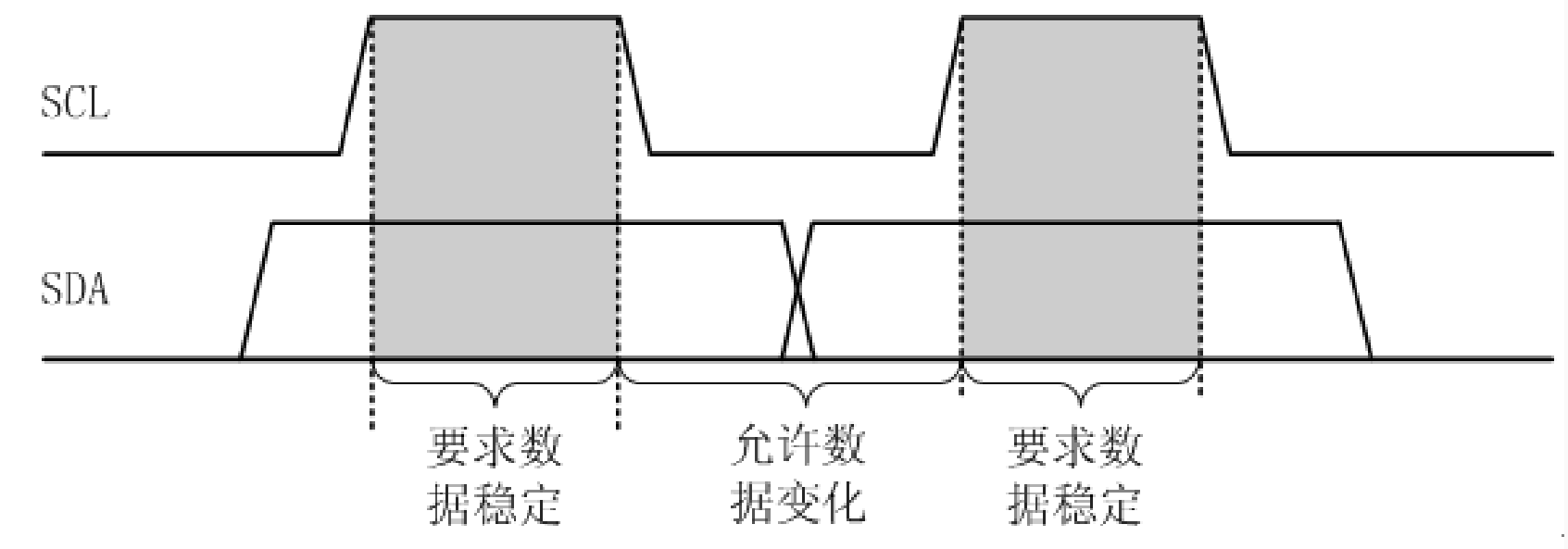
I2C 串行总线概述

每个接到 I2C 总线上的器件都有唯一的地址。主机与其它器件间的数据传送可以是由主机发送数据到其它器件，这时主机即为发送器。由总线上接收数据的器件则为接收器。

在多主机系统中，可能同时有几个主机企图启动总线传送数据。为了避免混乱，I2C 总线要通过总线仲裁，以决定由哪一台主机控制总线。

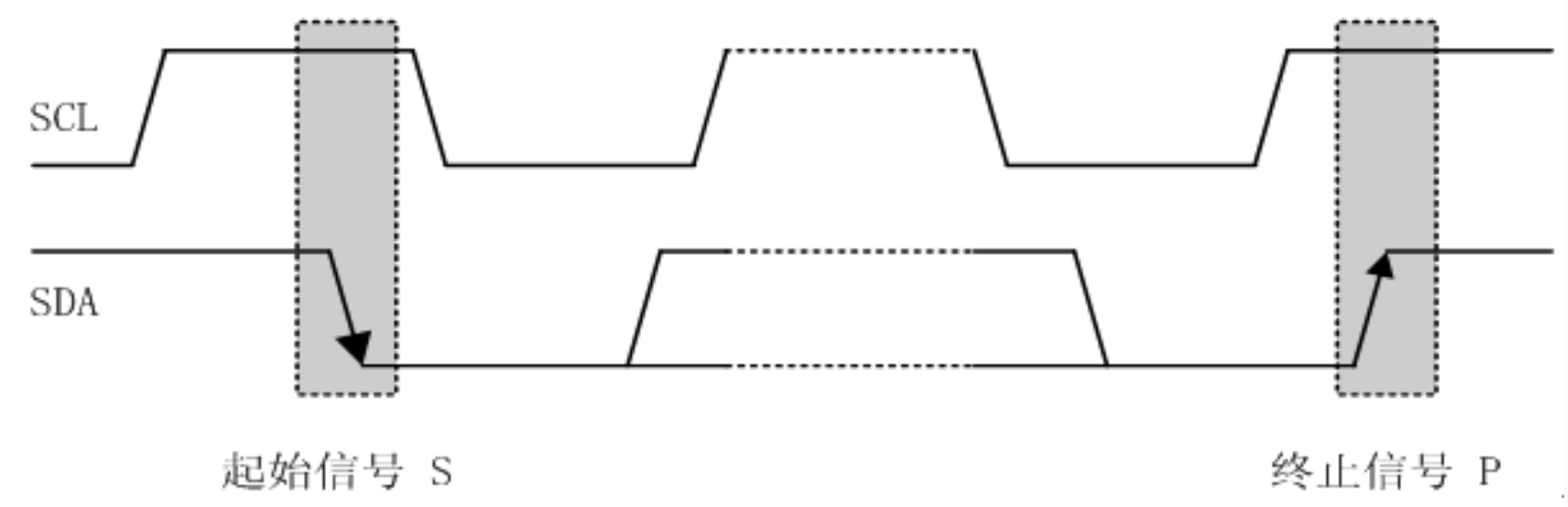
数据位的有效性规定

I2C 总线进行数据传送时，时钟信号为高电平期间，数据线上的数据必须保持稳定，只有在时钟线上的信号为低电平期间，数据线上的高电平或低电平状态才允许变化。



起始和终止信号

SCL 线为高电平期间，SDA 线由高电平向低电平的变化表示起始信号；SCL 线为高电平期间，SDA 线由低电平向高电平的变化表示终止信号。



起始和终止信号

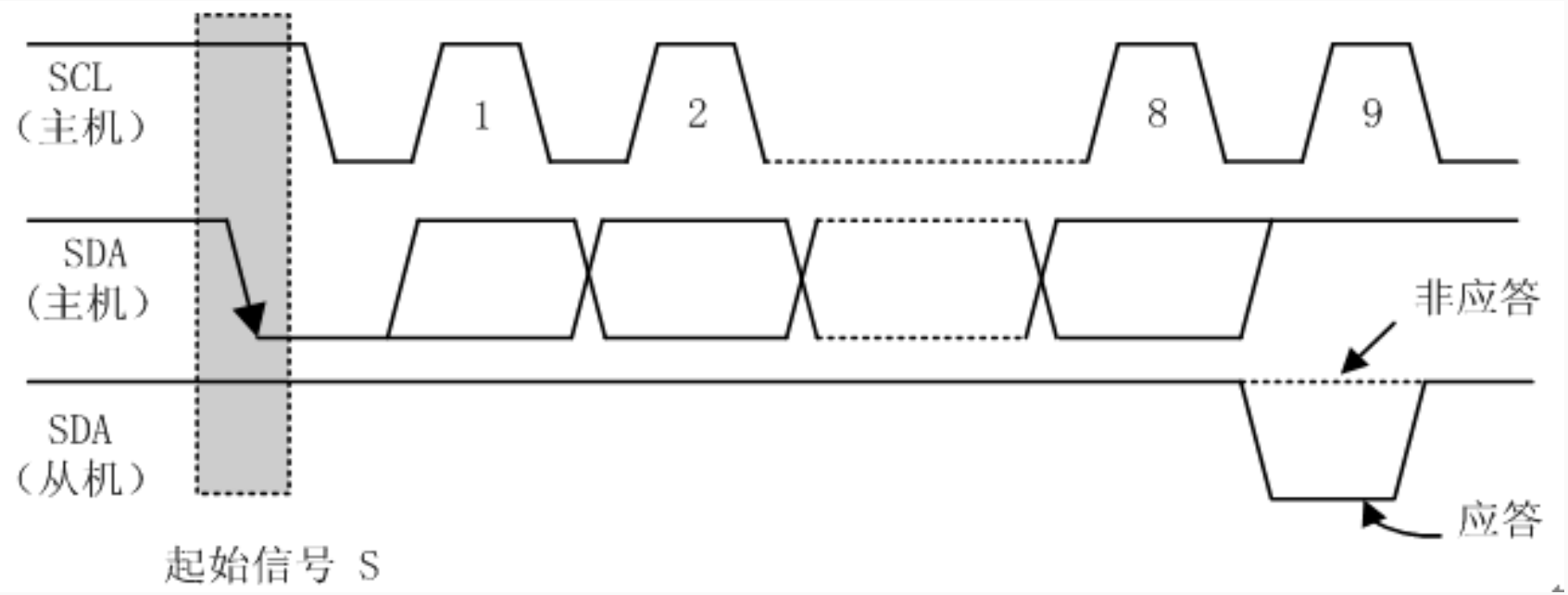
起始和终止信号都是由主机发出的，在起始信号产生后，总线就处于被占用的状态；在终止信号产生后，总线就处于空闲状态。

连接到 I2C 总线上的器件，若具有 I2C 总线的硬件接口，则很容易检测到起始和终止信号。

接收器件收到一个完整的数据字节后，有可能需要完成一些其它工作，如处理内部中断服务等，可能无法立刻接收下一个字节，这时接收器件可以将 SCL 线拉成低电平，从而使主机处于等待状态。直到接收器件准备好接收下一个字节时，再释放 SCL 线使之为高电平，从而使数据传送可以继续进行。

数据字节传送与应答

每一个字节必须保证是 8 位长度。数据传送时，先传送最高位（MSB），每一个被传送的字节后面都必须跟随一位应答位（即一帧共有 9 位）。



数据字节传送与应答

由于某种原因从机不对主机寻址信号应答时（如从机正在进行实时性的处理工作而无法接收总线上的数据），它必须将数据线置于高电平，而由主机产生一个终止信号以结束总线的数据传送。

如果从机对主机进行了应答，但在数据传送一段时间后无法继续接收更多的数据时，从机可以通过对无法接收的第一个数据字节的“非应答”通知主机，主机则应发出终止信号以结束数据的继续传送。

当主机接收数据时，它收到最后一个数据字节后，必须向从机发出一个结束传送的信号。这个信号是由对从机的“非应答”来实现的。然后，从机释放 SDA 线，以允许主机产生终止信号。

数据帧格式

I2C 总线上传送的数据信号是广义的，既包括地址信号，又包括真正的数据信号。

在起始信号后必须传送一个从机的地址（7 位），第 8 位是数据的传送方向位（R/T），用“0”表示主机发送数据（T），“1”表示主机接收数据（R）。每次数据传送总是由主机产生的终止信号结束。但是，若主机希望继续占用总线进行新的数据传送，则可以不产生终止信号，马上再次发出起始信号对另一从机进行寻址。

在总线的一次数据传送过程中，可以有以下几种组合方式：

数据帧格式

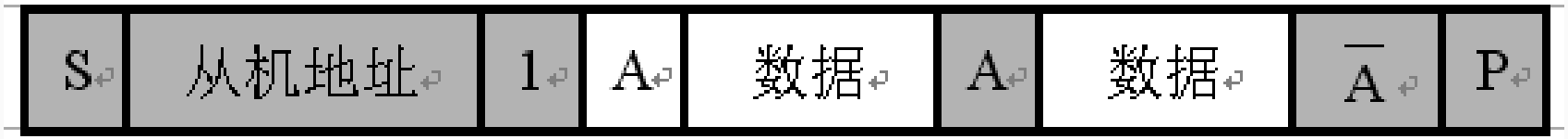
a、主机向从机发送数据，数据传送方向在整个传送过程中不变：

注：有阴影部分表示数据由主机向从机传送，无阴影部分则表示数据由从机向主机传送。

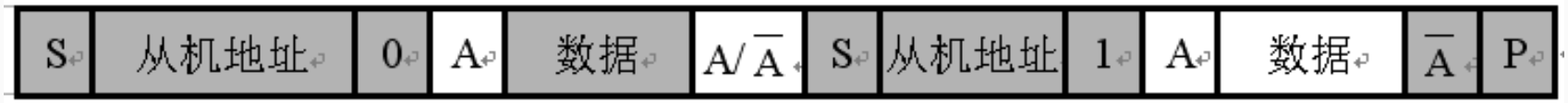
A 表示应答，A 非表示非应答（高电平）。S 表示起始信号，P 表示终止信号。。

数据帧格式

b、主机在第一个字节后，立即从从机读数据



c、在传送过程中，当需要改变传送方向时，起始信号和从机地址都被重复产生一次，但两次读/写方向位正好反相。



总线寻址

I2C 总线协议有明确的规定：采用 7 位的寻址字节（寻址字节是起始信号后的第一个字节）。

寻址字节的位定义

位：	7	6	5	4	3	2	1	0
	从机地址							R/$\overline{W}$

D7～D1 位组成从机的地址。D0 位是数据传送方向位，为“0”时表示主机向从机写数据，为“1”时表示主机由从机读数据。

总线寻址

主机发送地址时，总线上的每个从机都将这 7 位地址码与自己的地址进行比较，如果相同，则认为自己正被主机寻址，根据 R/T 位将自己确定为发送器或接收器。

从机的地址由固定部分和可编程部分组成。在一个系统中可能希望接入多个相同的从机，从机地址中可编程部分决定了可接入总线该类器件的最大数目。如一个从机的 7 位寻址位有 4 位是固定位，3 位是可编程位，这时仅能寻址 8 个同样的器件，即可以有 8 个同样的器件接入到该 I2C 总线系统中。

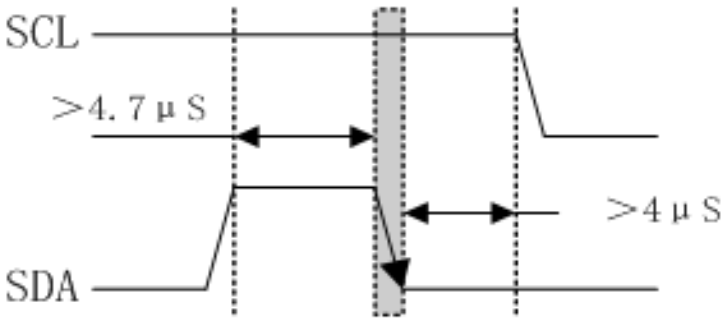
总线数据传送的模拟

主机可以采用不带 I2C 总线接口的单片机，利用软件实现 I2C 总线的数据传送，即软件与硬件结合的信号模拟。

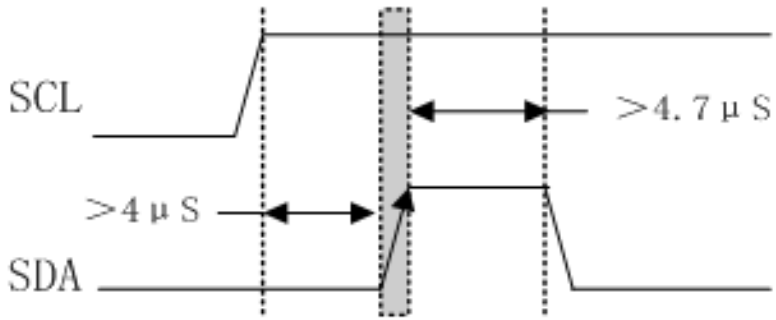
典型信号模拟

为了保证数据传送的可靠性，标准的 I2C 总线的数据传送有严格的时序要求。I2C 总线的起始信号、终止信号、发送“0”及发送“1”的模拟时序：

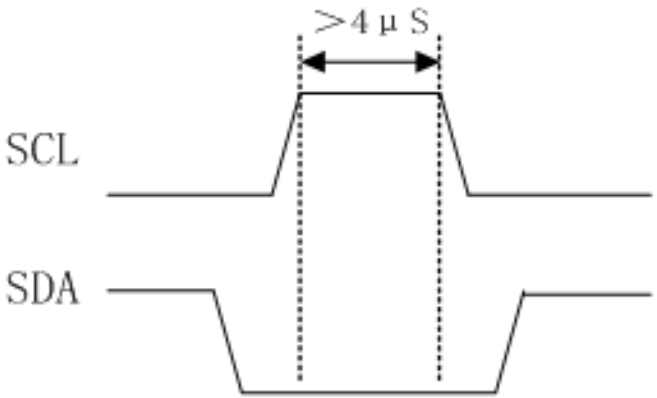
典型信号模拟



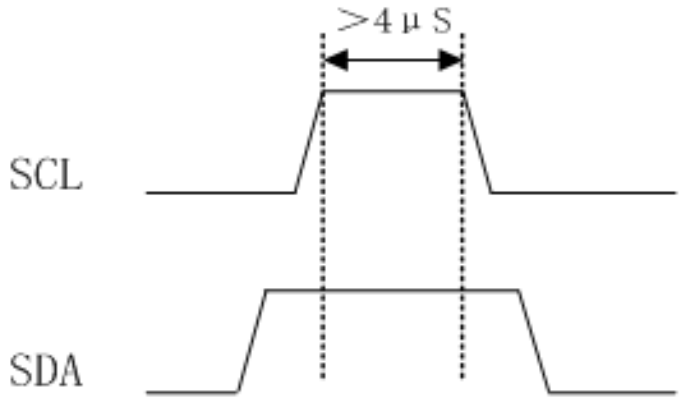
起始信号 S



终止信号 P



应答/“0”



非应答/“1”

I2C 工作方式

在 MSSP 模块的 I2C 工作方式中，完全实现了主/从功能，在硬件上实现开始位（START）和停止位（STOP）的中断，在多主机应用中判断总线的空闲，也实现了标准的 7 位或 10 位地址寻址。

在 I2C 工作方式中，采用两根引脚来传输数据，它们是时钟引脚 SCL 和数据引脚 SDA。在设置 I2C 工作方式时，自动设置 SCL 和 SDA 的功能。

寄存器与工作方式

设置 I2C 工作方式需要涉及六个寄存器，它们是：SSP 控制寄存器（SSPCON）、SSP 控制寄存器 2（SSPCON2）、SSP 状态寄存器（SSPSTAT）、串行接收/发送缓冲器（SSPBUF）、SSP 移位寄存器（SSPSR）和 SSP 地址寄存器（SSPADD）。其中 SSPSR 不可直接寻址。

控制寄存器 SSPCON 用于设置 I2C 工作方式，设置 SSPCON 的 D3:D0，可以选择工作方式：

- I2C 从动方式；
- I2C 主控方式。

I2C 从模式

在从模式，SCL 和 SDA 引脚必须定义为输入。当地址相吻合，或地址吻合后接收到数据，硬件设备自动产生一个应答脉冲/ACK，然后将当前在 SSPSR 中接收到的数据输送到 SSPBUF。

在这两种条件同时发生或其中一种发生，都会不产生/ACK：

1. 在传输数据接收到之前缓冲器满状态位 BF（SSPSTAT 的 D0）被置位；
2. 在传输数据接收到之前溢出状态位 SSPOV（SSPSTAT 的 D6）被置位。

如果 BF 被置位，SSPSR 寄存器的值没有送入 SSPBUF，但 SSPIF 和 SSPOV 状态位被置位。读 SSPBUF 寄存器，标志位 BF 自动清除，SSPOV 标志位必须用软件清除。

I2C 从模式寻址

一旦选择 MSSP 功能模块，CPU 等待开始信号 (START) 的出现，跟着开始信号是 8 位数据移入 SSPSR 寄存器。所有的输入数据位在时钟 SCL 的上升沿采样。SSPSR 寄存器的地址在第 8 个 SCL 脉冲下降沿与 SSPADD 寄存器内容进行比较。一旦相等，且 BF 和 SSPOV 标志位为零，就将完成如下的功能：

1. 在第 8 个 SCL 脉冲下降沿 SSPSR 内容送入 SSPBUF 寄存器；
2. 缓冲器满标志位 BF 在第 8 个 SCL 脉冲下降沿置位；
3. 产生应答脉冲；
4. 在第 9 个 SCL 脉冲下降沿 SSP 中断标志位 SSPIF（PIR1 的 D3）置位。

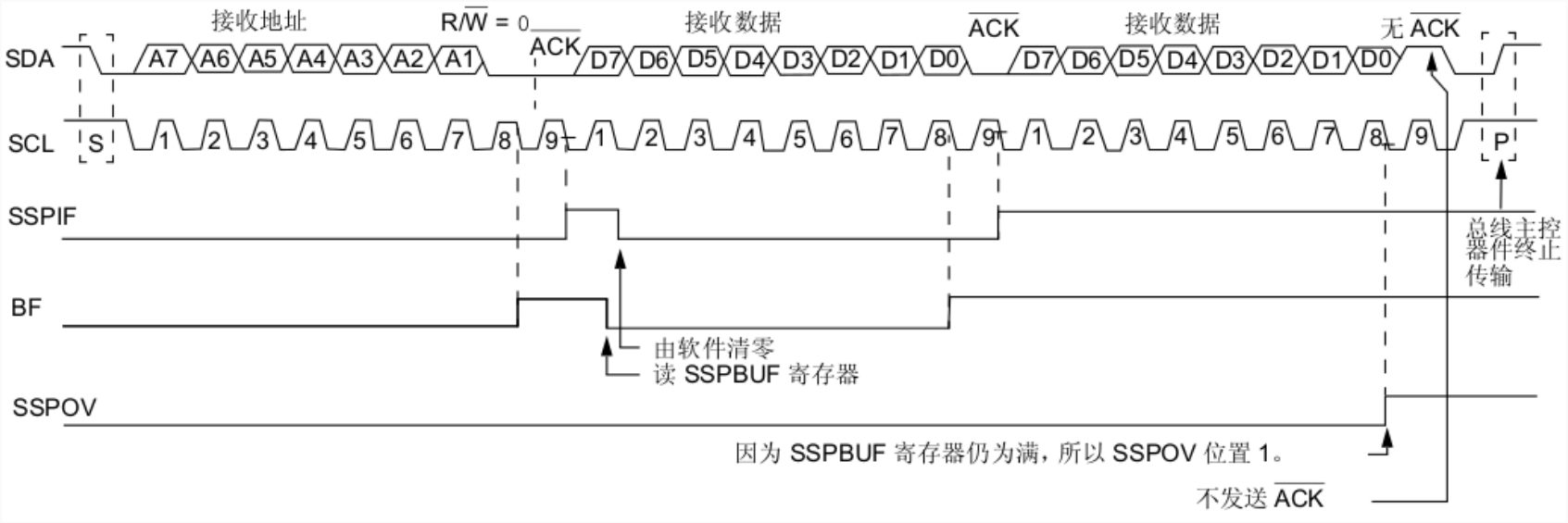
I2C 从模式接收

当地址字节的 $R/\overline{W}$ 位清零以及出现地址匹配，状态寄存器 SSPSTAT 寄存器的 $R/\overline{W}$ 位就被清零，同时把接收到的地址送入 SSPBUF 寄存器。

当状态寄存器 SSPSTAT 中的缓冲器满标志位 BF 被置位或 SSPCON 寄存器中的溢出标志位 SSPOV 被置位，称为溢出条件。当地址字节的溢出条件满足，从设备不会产生应答脉冲。

每传输一个数据，都会产生 SSP 中断而发出中断请求。中断标志位置位后必须用软件清零。状态寄存器 SSPSTAT 用于确定接收数据字节的状态。

I2C 从模式接收时序

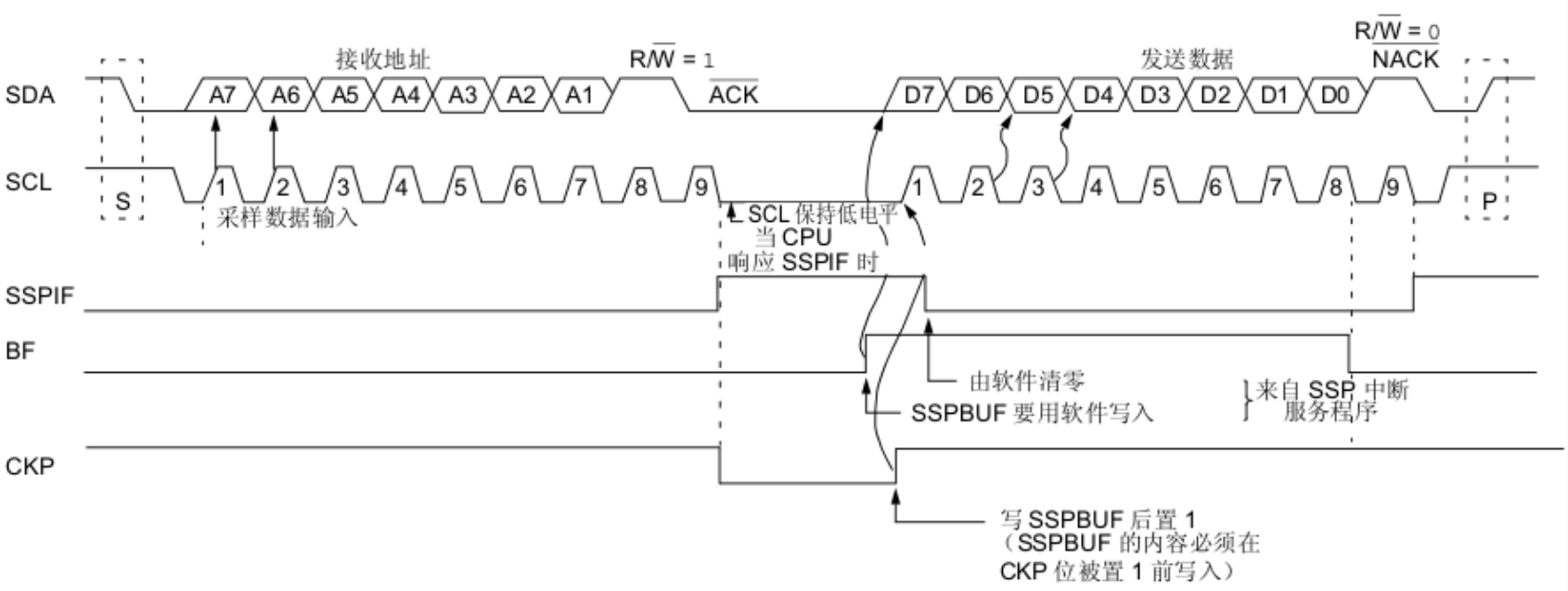

接收地址
R/W = 0
接收数据
ACK
接收数据
无 ACK
SDA
A7 A6 A5 A4 A3 A2 A1 ACK D7 D6 D5 D4 D3 D2 D1 D0 D7 D6 D5 D4 D3 D2 D1 D0
SCL
S 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 1 2 3 4 5 6 7 8 9 P
SSPIF
BF
由软件清零
读 SSPBUF 寄存器
总线主控器件终止传输
SSPOV
因为 SSPBUF 寄存器仍为满，所以 SSPOV 位置 1。
不发送 ACK

I2C 从模式发送

当地址字节的读/写控制位 $R/\overline{W}$ 位置位以及出现地址匹配，状态寄存器 SSPSTAT 寄存器的 $R/\overline{W}$ 位就置位。同时把接收到的地址送入 SSPBUF 寄存器。在时钟 SCL 的第九个脉冲时发送应答信号，同时时钟线 SCL 保持为低电平。发送的数据必须送入缓冲器 SSPBUF，同时也送入 SSPSR 寄存器，然后通过把控制寄存器 SSPCON 的 CKP 位 (D4) 置 l，允许 SCL 线工作。在 SCL 时钟输入脉冲的下降沿，8 位数据被依次移位串行输出。这就确保在 SCL 为高电平期间 SDA 信号肯定是有效的。

每个数据传输字节，都会在 SCL 脉冲的第 9 个脉冲下降沿把中断标志位 SSPIF 置 1，发出中断请求，而 SSPIF 标志必须用软件清零。状态寄存器 SSPSTAT 是用于确定发送数据字节的状态。

I2C 从模式发送时序


接收地址
R/W = 1
发送数据
R/W = 0
NACK
SDA
A7
A6
A5
A4
A3
A2
A1
ACK
D7
D6
D5
D4
D3
D2
D1
D0
SCL
S
1
2
3
4
5
6
7
8
9
P
采样数据输入
SCL 保持低电平
当 CPU
响应 SSPIF 时
SSPIF
BF
由软件清零
SSPBUF 要用软件写入
来自 SSP 中断
服务程序
CKP
写 SSPBUF 后置 1
（SSPBUF 的内容必须在
CKP 位被置 1 前写入）

I2C 主控模式

主控方式是通过检测“START”和“STOP”信号产生中断进行的。在 SSPSTAT 寄存器中用 S 位 (START) 和 P 位 (STOP) 来表示检测的情况。复位或关闭 SSP 模块将会对 S 和 P 清零。当 P 为 1 或 P 和 S 都为零而总线是空闲时，可以对 I2C 总线进行控制操作。

在主控模式中，SCL 和 SDA 是由 MSSP 硬件控制的。

I2C 主控模式

下列情况会引起中断标志被置 1(即 SSPIF 置位)，如果允许中断，便产生中断。

1. 起始 (START) 信号；
2. 停止 (STOP) 信号；
3. 发送/接收数据传输字节；
4. 应答信号发送；
5. 重 START 信号。

I2C 主控模式

在多主机模式中，利用检测到 START 信号和 STOP 信号就产生中断，可以判断总线何时空闲。当芯片复位或 MSSP 关闭都会使 STOP（P）和 START（S）标志位清零。当总线处于忙而 SSP 允许中断，一旦检测到停止信号，就产生中断。

在多主机操作中，仲裁总线控制需要检测 SDA 线，以判断它的电平是否是所希望的的输出电平。这种检测由硬件自动完成，结果放在 BCLIF 位中。

I2C 主控模式

下列情况会引起仲裁丢失：

1. 传输地址；
2. 传输数据；
3. 起始信号；
4. 重 START 信号；
5. 应答信号。

I2C 总线主控模式支持

通过置位和复位 SSPCON 寄存器中适当的 SSPM 位和置位 SSPEN，可以进入主控模式。一旦进入主控模式，用户可以有六种选择：

1. 在 SDA 和 SCL 线上声明起始（START）状态；
2. 在 SDA 和 SCL 线上声明重复的起始（START）状态；
3. 写 SSPBUF 寄存器，初始化数据/地址的传送；
4. 在 SDA 和 SCL 线上产生终止（STOP）信号；
5. 匹配 I2C 端口接收数据；
6. 在接收数据字节后产生应答信号。

I2C 总线主控模式操作

主控模式器件负责产生串行时钟、起始信号和终止信号。

在主控发送器模式，SCL 输出时钟信号，SDA 输出串行数据。第一个发送的字节包含接收器件的 7 位从地址和读/写方向位（$R/\overline{W}$），在发送数据方式，$R/\overline{W} = 0$（表示输出）。串行数据每次传输一字节（8 位），每字节传送完后，接收一个应答信号。输出起始（START）信号和终止（STOP）信号表示串行传输的开始于结束。

I2C 总线主控模式操作

在主控接收模式，第一个发送的字节包含发送数据器件的 7 位从地址和读/写方向位（$R/\overline{W}$），在接收数据方式（对主控器件而言），$R/\overline{W} = 1$（表示输入），即 7 位地址后在加一位 1。SCL 输出时钟信号，SDA 接收串行输入的 8 位数据。在每接收完一字节，发送一个应答信号。输出起始（START）信号和终止（STOP）信号表示串行传输的开始于结束。

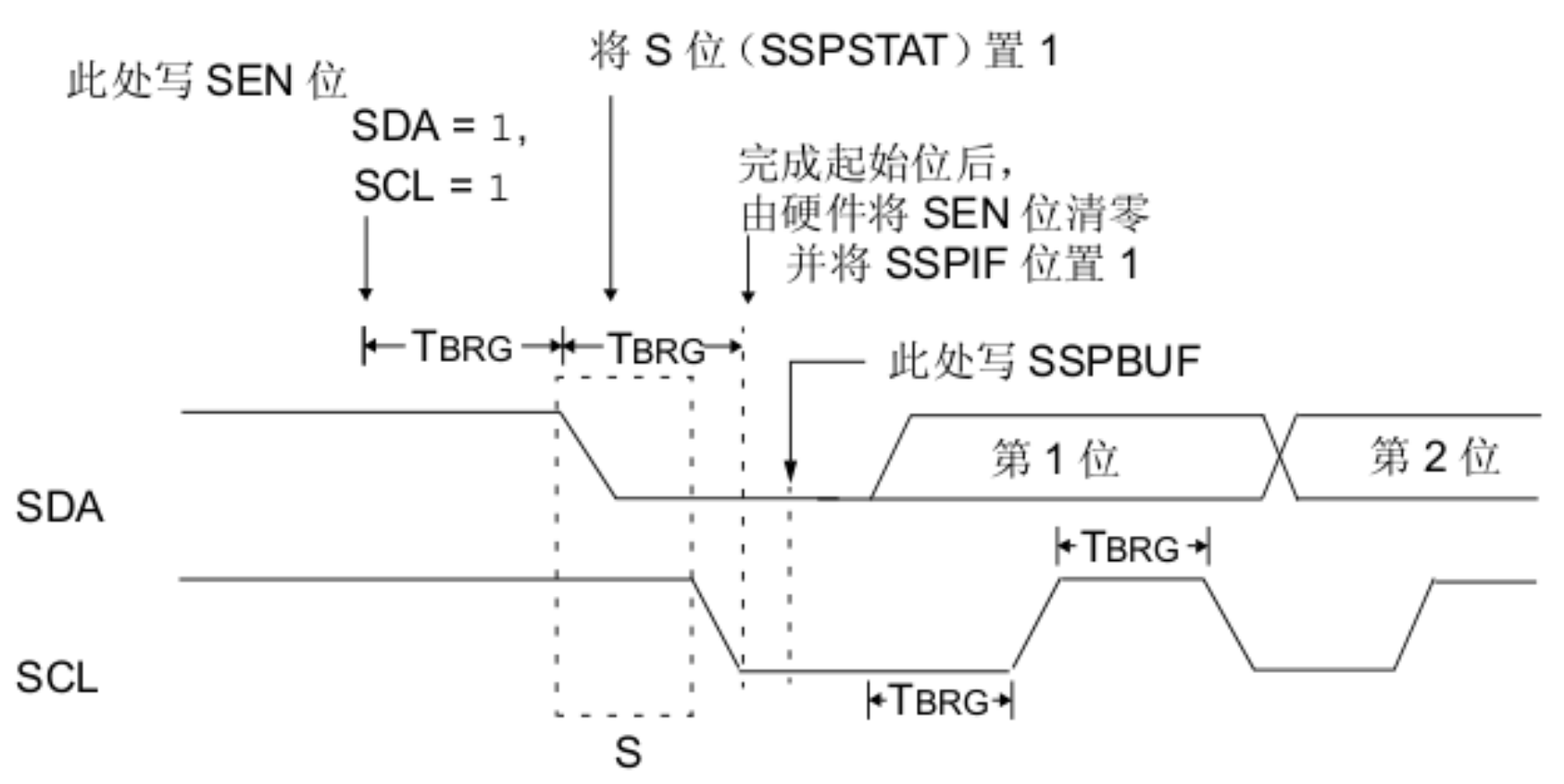
I2C 总线主控模式典型发送步骤

1. 置位 SSPCON2 的起始信号允许位 SEN，产生起始信号（START）；
2. 置位 SSPIF，等待起始时间；
3. 将从地址写入 SSPBUF；
4. 将地址移出 SDA 线；
5. 接收从器件返回的应答信号，并写入 SSPCON2 寄存器的 ACKSTAT 位;
6. 置位 SSPIF，在第九个时钟周期产生中断；
7. 用户把 8 位数据写入 SSPBUF；
8. 将 8 位数据移出 SDA 线；
9. 接收从器件返回的应答信号，并写入 SSPCON2 寄存器的 ACKSTAT 位;
10. 置位 SSPIF，在第九个时钟周期产生中断；
11. 置位 SSPCON2 的终止信号允许位 PEN，产生启终止信号（STOP）；
12. 一旦启终止信号（STOP）完成，将产生中断。

I2C 总线主控模式起始信号时序

置位 SSPCON2 中起始信号允许位 SEN（D0），初始化起始信号。如果检测到 SDA 和 SCL 读为高，就将 SSPADD 中低 7 位数据装入波特率发生器并开始计数。当波特率发生器计数时间（TBRG）到，而 SDA 和 SCL 都为高，SDA 将被置为低。在 SCL 高时 SDA 从高变为低就是起始信号，SSPSTAT 的 S 位（D3）被置位。同时波特率发生器重载 SSPADD 中低 7 位数据，重新计数。当波特率发生器计数时间（TBRG）又到，SEN 又硬件自动清零。波特率暂停工作，SDA 保持为低，产生起始信号完成。

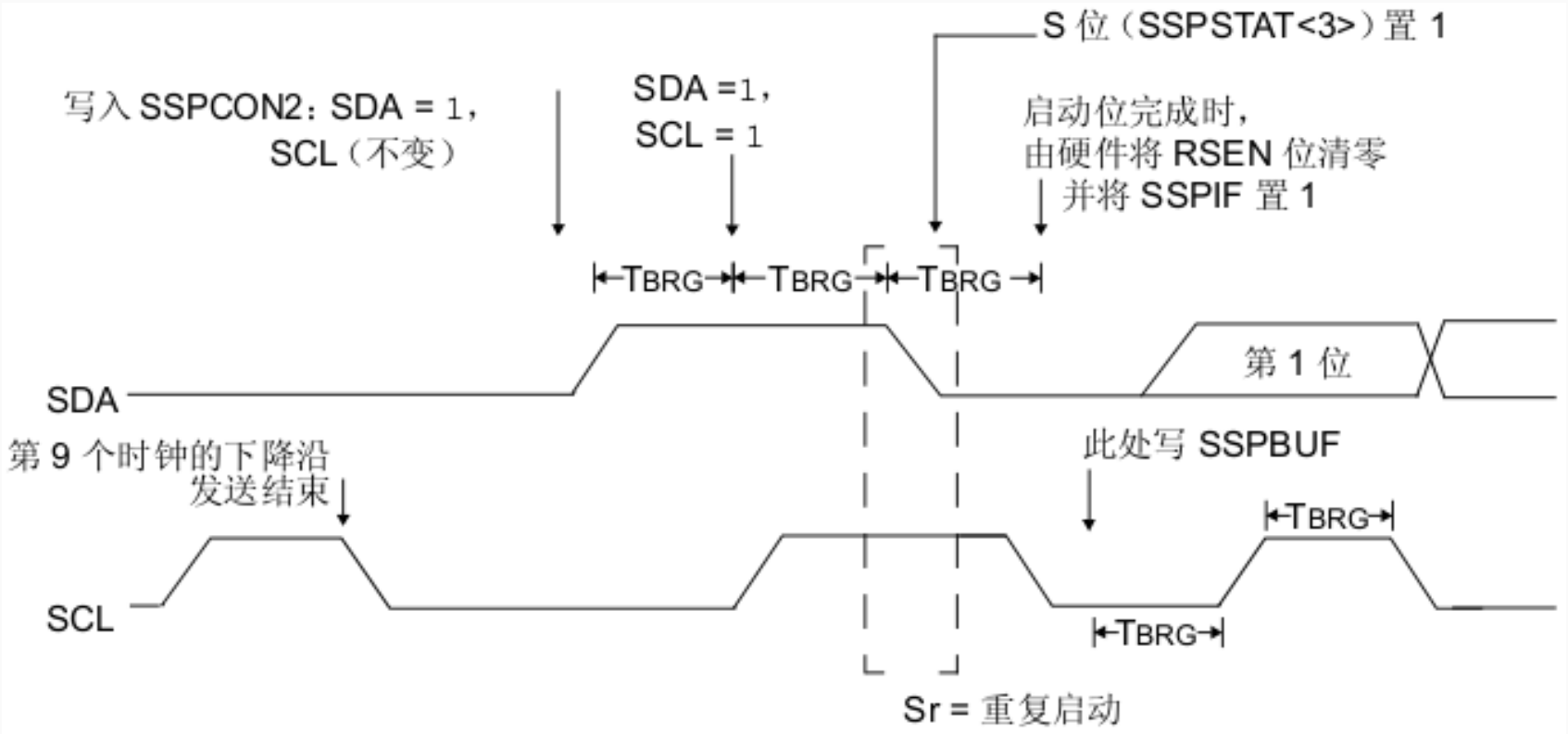
I2C 总线主控模式起始信号时序



I2C 总线主控模式重起始信号时序

当 SSPCON2 的 RSEN 位（D1）被置位，而此时 I2C 总线空闲，则会产生重起始信号。当 RSEN 置位，SCL 变为低，SCLA 变为低使得波特率发生器加载 SSPADD 的低 7 位数据，同时开始计数。SDA 被释放（为高）一个波特率发生器计数时间（TBRG）。当波特率发生器计数时间到，如果检测到 SDA 为高电平，SCL 将变为高电平。当检测到 SCL 为高电平，波特率发生器重载 SSPADD 的低 7 位数据并开始计数。SDA 和 SCL 同时保持高电平为一个 TBRG 时间，紧接着 SDA 变为低电平一个 TBRG 时间。这样 RSEN 被硬件自动清零，波特率发生器不再重载，SDA 保持低电平。一旦再 SDA 和 SCL 检测到起始信号（START），SSPSTAT 寄存器的 S 位（D3）置 1，如果波特率发生器的计数时间到，SSPIF 也置 1。

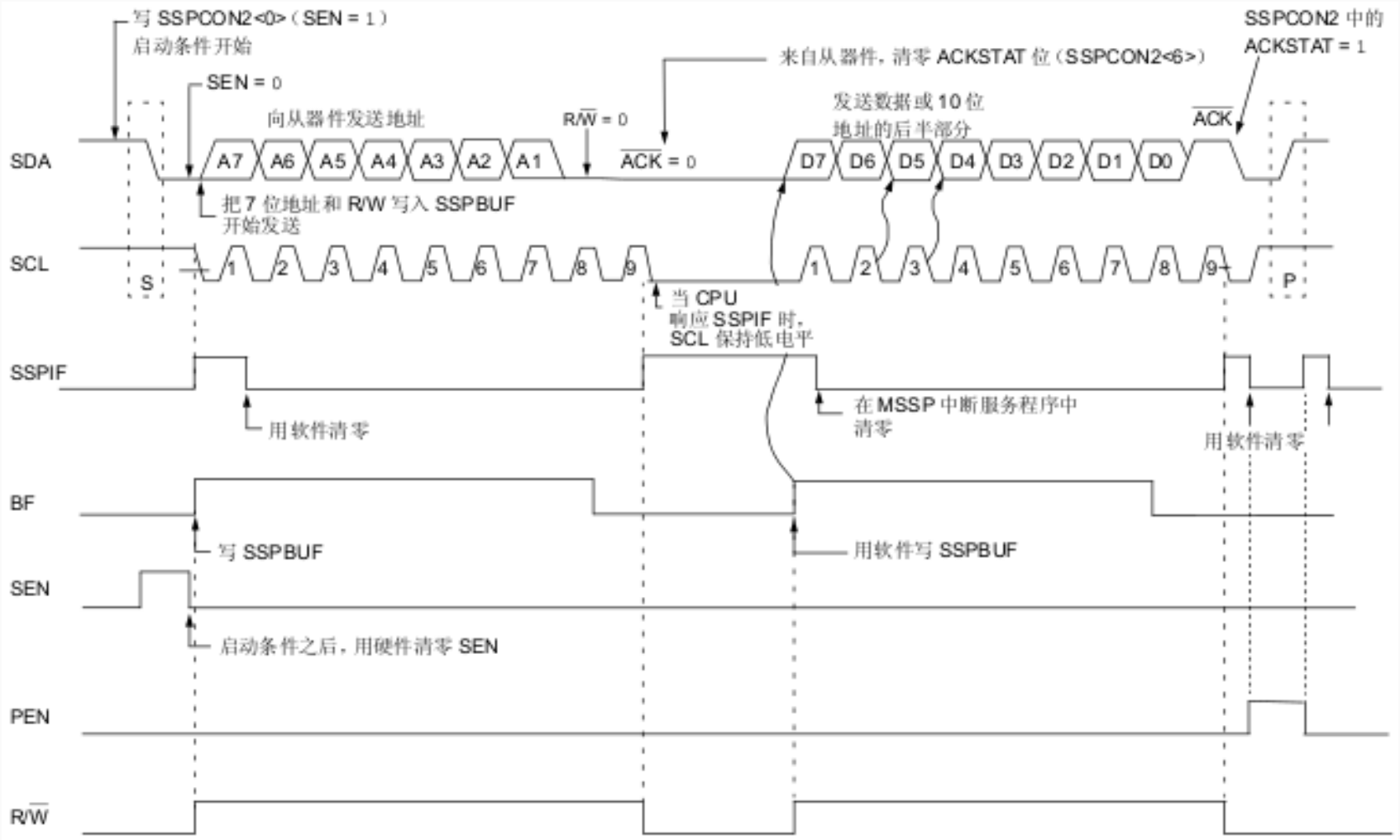
I2C 总线主控模式重起始信号时序



I2C 总线主控模式发送

将数据，或 7 位地址，或 10 位地址任一字节写入 SSPBUF 寄存器，就可以完成数据发送。写入 SSPBUF 使 BF 置 1，允许波特率发生器计数以及开始下一个发送。地址/数据的每一位在 SCL 的下降沿移出到 SDA 线上，SCL 保持一个 TBRG 时间低电平，数据在 SCL 变为高电平之前有效。当 SCL 变为高电平，也保持一个 TBRG 时间，在这期间 SDA 线上数据必须保持稳定，直到下一个 SCL 下降沿。当第八位移出 SDA 后（第八个时钟下降沿），BF 标志位清零，主控器件释放 SDA 数据线，允许被寻址的从动器件在地址匹配或接收到数据时在第九个时钟应答。应答信号在第九个时钟下降沿读入 ACKDT 位。如果主控器件接收到应答信号，应答状态位 ACKSTAT 被清零，否则状态置 1。第九个时钟后，SSPIF 标志位置 1，波特率发生器暂停直至下一个数据写入 SSPBUF，SCL 线保持低电平，SDA 线保持不变。

I2C 总线主控模式发送时序

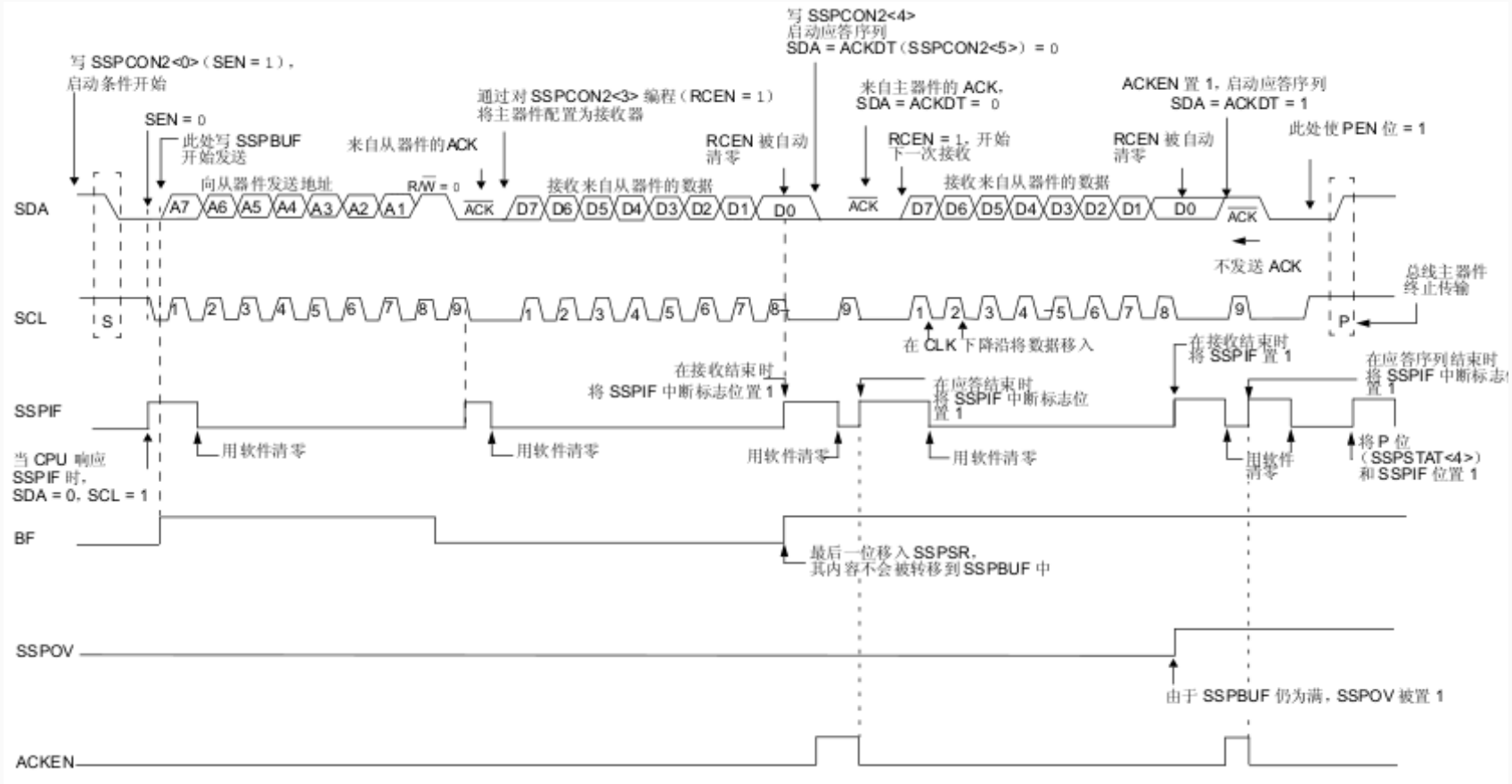


I2C 总线主控模式接收

置位 SSPCON2 的接收允许位 RCEN，进入主控接收模式。波特率发生器开始计数，一个周期后，SCL 线上的状态从低电平变为高电平或从高电平变为低电平，数据移入 SSPSR 寄存器。在第八个时钟下降沿，接收允许位自动清零，SSPSR 的内容输入 SSPBUF 寄存器，BF 和 SSPIF 置位，波特率发生器停止计数，SCL 保持低电平。SSP 处于空闲状态，等待下一个命令。当 CPU 读出缓冲器的值，BF 标志位自动清零。用户在接收结束时，通过置应答允许位 ACKEN，发送应答信号。

当正在接收数据（SSPSR 正在移入数据）时，用户写 SSPBUF 寄存器，则 WCOL 标志位被置 1，但 SSPBUF 寄存器的内容并没有改变（没有写入）

I2C 总线主控模式接收时序

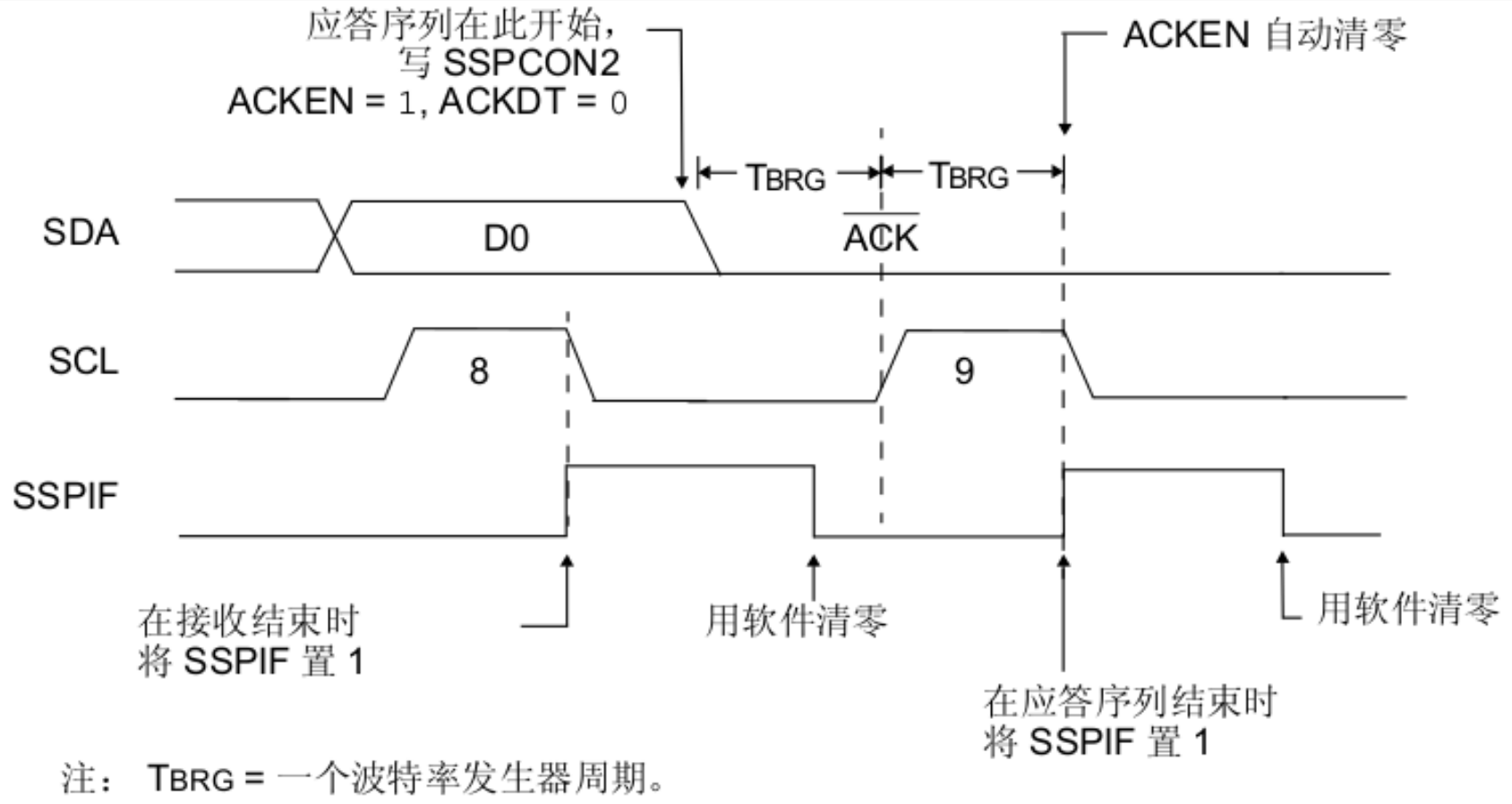


I2C 总线主控模式应答信号时序

置位应答信号序列允许位 ACKEN，就可以产生应答信号。当 ACKEN=1，SCL 线被拉到低电平，应答数据位的值发送到 SDA 线上。如果需要产生应答信号，使 ACKDT 复位，否则可以在开始应答之前置位 ACKDT。波特率发生器计数一个周期（TBRG），SCL 释放为高电平。当检测到 SCL 也为高电平（时钟仲裁），波特率发生器计数一个周期（TBRG），然后 SCL 被拉到低电平。这样，ACKEN 被自动清零，波特率发生器被关闭，SSP 模块进入空闲状态。

当正在产生应答信号而用户写 SSPBUF 寄存器，则 WCOL 标志位被置 1，但 SSPBUF 寄存器的内容并没有改变（没有写入）。

I2C 总线主控模式应答信号时序



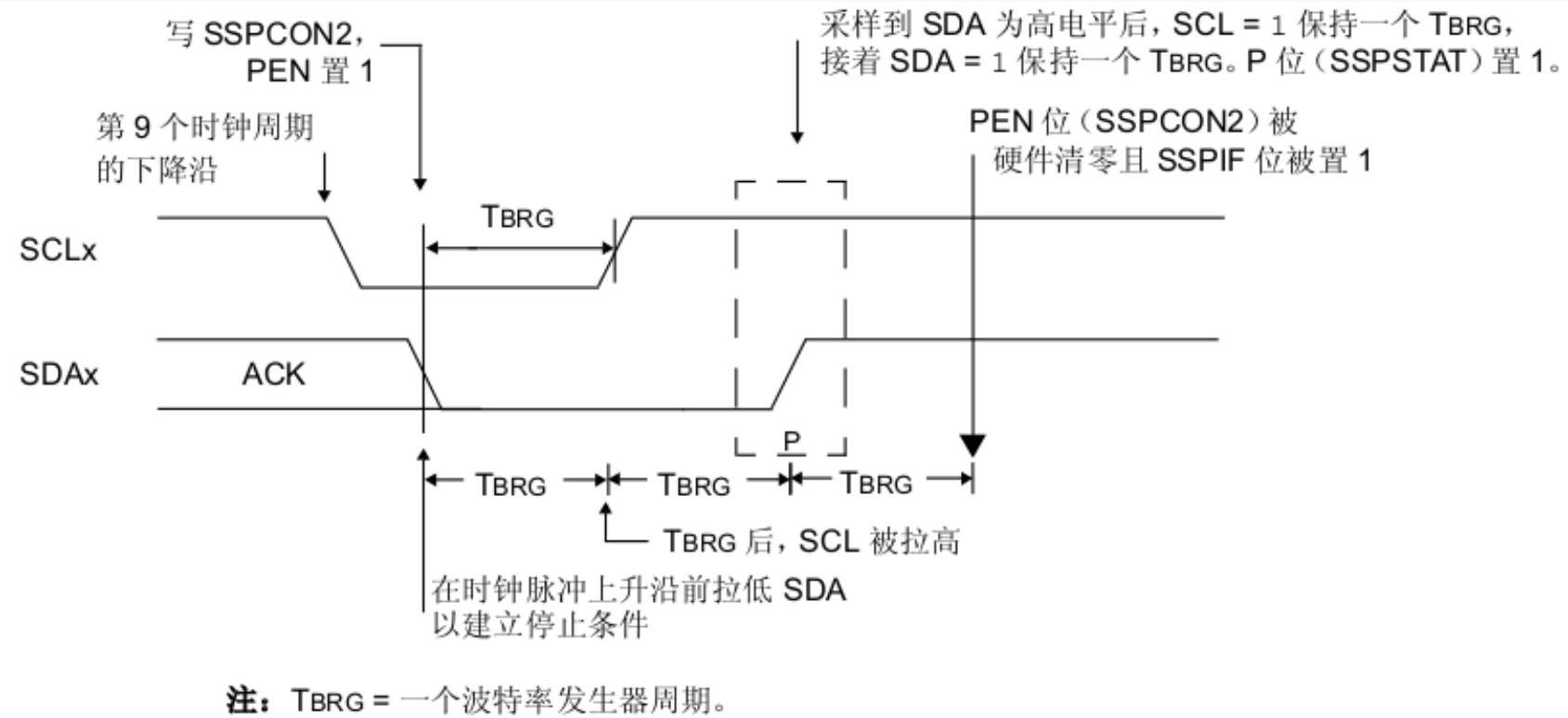
注： TBRG = 一个波特率发生器周期。

I2C 总线主控模式终止信号时序

在接收/发送结束时，置位 SSPCON2 的终止信号允许位 PEN，在 SDA 线上产生终止（STOP）信号。在接收/发送结束时，在第九个时钟的下降沿之后，SCL 保持低电平。当 PEN 置位，主控器件使 SDA 为低电平。当检测到 SDA 为低电平，波特率发生器重加载，减 1 计数到 0。当波特率发生器计数时间（TBRG）到，SCL 变为高电平。一个计数周期（TBRG）后，SDA 变为高电平。在 SCL=1 期间检测到 SCLA 为高，则 P 位置 1。一个计数周期（TBRG）后，PEN 清零，SSPIF 置位。

当正在产生终止（STOP）信号而用户写 SSPBUF 寄存器，则 WCOL 标志位被置 1，但 SSPBUF 寄存器的内容并没有改变（没有写入）。

I2C 总线主控模式终止信号时序

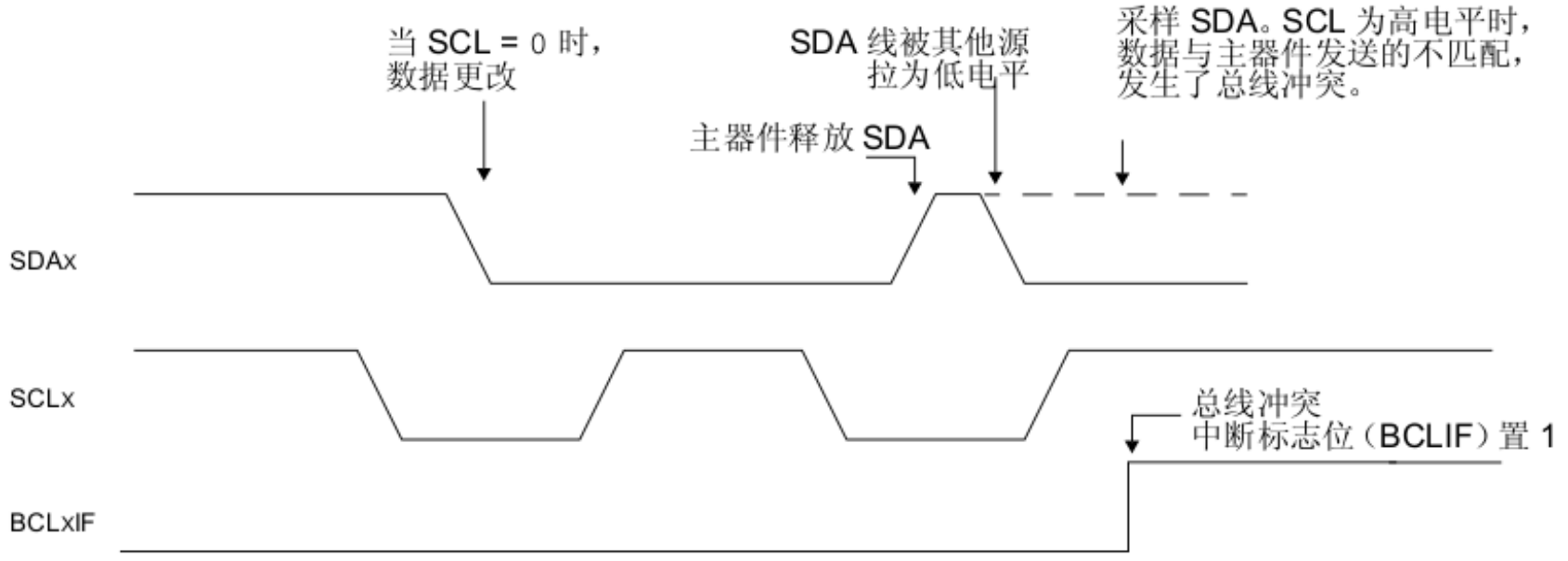


注： TBRG = 一个波特率发生器周期。

I2C 总线主控模式总线仲裁

通过总线仲裁首先多主机模式支持。当主机输出地址/数据到 SDA 线，通过使 SDA 浮空作为“1”，而另一主机输出“0”，这样就产生总线仲裁。当 SCL 浮空为高电平，数据必须稳定。如果期望的在 SDA 线上的数据是“1”，而实际检测到的 SDA=0，这就发生了总线碰撞。主机将置总线碰撞中断标志位 BCLIF，同时复位 I2C 端口到空闲状态。

I2C 总线主控模式总线仲裁



应用实例

例题 9-5

本例题电路原理图如图 9-5 所示，除连接有 8 位数码管显示和 16 个键盘电路以外，还利用 RC6 和 RC7 引脚组成一个 IIC 同步串行功能，实现对 24LC515 EEPROM 的串行数据传送。

编程要求：

首先将 64 个数据 00H～3FH 存入 EEPROM 单元 0000H～003FH 中；

然后再将 0010H～001FH 单元中的数据取出，存入数据存储器单元 40H～4FH 中；

最后逐个送往数码显示区的最后两位显示，每个数显示停留 1s。

变量定义

```
1     INCLUDE "p16f877a.inc"
2   ;-----------------------------------------------------------
3   EEPROM      EQU   2DH
4   DATA1       EQU   2EH
5   DATA0       EQU   2FH
6   ADDRH       EQU   2AH
7   ADDRL       EQU   2BH
8   DEVICE      EQU   2CH
9   TXBUF       EQU   28H
10  RXBUF       EQU   29H
11  CNT         EQU   27H
12  DI          EQU   07H
13  DO          EQU   06H
14  SCL         EQU   06H
15  SDA         EQU   07H
16  S1H         EQU   34H
17  S1L         EQU   35H
18  R1H         EQU   36H
19  R1Z         EQU   37H
20  R1L         EQU   38H
21  COUNTER     EQU   26H
22  TEMP        EQU   25H
```

主程序 1

```
1          ORG     0000H
2          NOP
3  START   MOVLW   0A8H
4          MOVWF   DEVICE
5          MOVLW   00H
6          MOVWF   ADDRH
7          MOVLW   00H
8          MOVWF   ADDRL
9          CALL    WRBYTE
10         MOVLW   10H
11         MOVWF   ADDRL
12         CALL    RDBYTE
13         MOVLW   40H
14         MOVWF   FSR
```

主程序 2

```
1  TP      MOVF    INDF, W
2          MOVWF   S1L
3          CLRF    S1H
4          CALL    BTOBCD
5          MOVF    R1L, W
6          ADDLW   0FH
7          MOVWF   60H
8          SWAPF   R1L, W
9          ANDLW   0FH
10         MOVWF   61H
11         INCF    FSR, F
12         BTFSS   FSR, 4
13         GOTO    TMT
14         GOTO    $
15 TMT     CALL    XSHI
16         CALL    DEL1S
17         GOTO    TP
```

向 24LC515 写入数据子程序 1

```
1  WRBYTE   MOVF    DEVICE, W
2           MOVWF   TXBUF
3           CALL    BSTART
4           CALL    TX
5           MOVF    ADDRH, W
6           MOVWF   TXBUF
7           CALL    TX
8           MOVF    ADDRL, W
9           MOVWF   TXBUF
10          CALL    TX
11          MOVLW   00H
12          MOVWF   COUNTER
```

向 24LC515 写入数据子程序 2

```
1 WRLOOP   MOVF     COUNTER, W
2          MOVWF    TXBUF
3          CALL     TX
4          INCF     COUNTER, F
5          BTFSS    COUNTER, 6
6          GOTO     WRLOOP
7          CALL     BSTOP
8          CALL     DEL1S
9          RETLW    00H
```

从 24LC515 读出数据子程序 1

```
1  RDBYTE   MOVLW    40H
2           MOVWF    FSR
3           MOVF     DEVICE, W
4           MOVWF    TXBUF
5           CALL     BSTART
6           CALL     TX
7           MOVF     ADDRH, W
8           MOVWF    TXBUF
9           CALL     TX
10          MOVF     ADDRL, W
11          MOVWF    TXBUF
12          CALL     TX
13          CALL     BSTART
14          MOVF     DEVICE, W
15          MOVWF    TXBUF
16          BSF      TXBUF, 0
17          CALL     TX
```

从 24LC515 读出数据子程序 2

```
1  RDLOOP   CALL     RX
2           MOVF     RXBUF, W
3           MOVWF    INDF
4           INCF     FSR, F
5           BTFSS    FSR, 4
6           GOTO     MACK
7           BSF      STATUS, RP0
8           MOVLW    B'00111111'
9           MOVWF    TRISC
10          BCF      STATUS, RP0
11          NOP
12          BSF      PORTC, SDA
13          NOP
14          BSF      PORTC, SCL
15          NOP
16          NOP
17          BCF      PORTC, SCL
```

从 24LC515 读出数据子程序 3

```
1           CALL    BSTOP
2           RETLW   00H
3   MACK    BSF     STATUS, RP0
4           MOVLW   B'00111111'
5           MOVWF   TRISC
6           BCF     STATUS, RP0
7           NOP
8           BCF     PORTC, SDA
9           NOP
10          BSF     PORTC, SCL
11          NOP
12          NOP
13          BCF     PORTC, SCL
14          GOTO    RDLOOP
```

字节发送子程序 1

```
1  TX      MOVLW   08H
2          MOVWF   CNT
3          BSF     STATUS, RP0
4          MOVLW   B'00111111'
5          MOVWF   TRISC
6          BCF     STATUS, RP0
7  TXLP    BCF     EEPROM, DO
8          BTFSC   TXBUF, 7
9          BSF     EEPROM, DO
10 BITOUT  NOP
11         BTFSS   EEPROM, DO
12         GOTO    BIT0
13         BSF     PORTC, SDA
14         GOTO    CLKOUT
15 BIT0    BCF     PORTC, SDA
```

字节发送子程序 2

```
1  CLKOUT   NOP
2           BSF     PORTC, SCL
3           NOP
4           NOP
5           NOP
6           NOP
7           NOP
8           BCF     PORTC, SCL
9           RLF     TXBUF, F
10          DECFSZ  CNT, F
11          GOTO    TXLP
12 BITIN    BSF     STATUS, RP0
13          MOVLW   B'10111111'
14          MOVWF   TRISC
15          BCF     STATUS, RP0
16          NOP
```

字节发送子程序 3

```
1  ACK      NOP
2           BCF      PORTC, SCL
3           NOP
4           NOP
5           NOP
6           BSF      PORTC, SCL
7           NOP
8           NOP
9           BTFSC    PORTC, SDA
10          GOTO     ACK
11          BCF      PORTC, SCL
12          RETLW    00H
```

字节接收子程序 1

```
1  RX      MOVLW   08H
2          MOVWF   CNT
3          CLRF    RXBUF
4          BSF     STATUS, RP0
5          MOVLW   B'10111111'
6          MOVWF   TRISC
7          BCF     STATUS, RP0
8          NOP
```

字节接收子程序 2

```
1  RXLP     BSF      PORTC, SCL
2           BCF      STATUS, C
3           NOP
4           NOP
5           NOP
6           NOP
7           NOP
8           BTFSC    PORTC, SDA
9           BSF      STATUS, C
10          RLF      RXBUF, F
11          BCF      PORTC, SCL
12          DECFSZ   CNT, F
13          GOTO     RXLP
14          RETLW    00H
```

发送开始子程序

```
1  BSTART  BSF     PORTC, SCL
2          BSF     PORTC, SDA
3          BSF     STATUS, RP0
4          MOVLW   B'00111111'
5          MOVWF   TRISC
6          BCF     STATUS, RP0
7          NOP
8          NOP
9          NOP
10         NOP
11         NOP
12         BCF     PORTC, SDA
13         NOP
14         NOP
15         NOP
16         NOP
17         NOP
18         BCF     PORTC, SCL
19         RETLW   00H
```

发送停止子程序

```
1  BSTOP   BCF     PORTC, SDA
2          BSF     STATUS, RP0
3          MOVLW   B'00111111'
4          MOVWF   TRISC
5          BCF     STATUS, RP0
6          BCF     PORTC, SCL
7          NOP
8          NOP
9          NOP
10         BSF     PORTC, SCL
11         NOP
12         NOP
13         NOP
14         BSF     PORTC, SDA
15         NOP
16         NOP
17         NOP
18         NOP
19         NOP
20         BCF     PORTC, SCL
21         NOP
22         RETLW   00H
```

二进制数转换成 BCD 码子程序 1

```
1  BTOBCD   MOVLW    10H
2           MOVWF    COUNTER
3           CLRF     R1H
4           CLRF     R1Z
5           CLRF     R1L
6  LOOP     RLF      S1L, F
7           RLF      S1H, F
8           RLF      R1L, F
9           RLF      R1Z, F
10          RLF      R1H, F
11          DECFSZ   COUNTER, F
12          GOTO     ADJDET
13          RETLW    00H
```

二进制数转换成 BCD 码子程序 2

```
1  ADJDET   MOVLW   R1L
2           MOVWF   FSR
3           CALL    ADJBCD
4           MOVLW   R1Z
5           MOVWF   FSR
6           CALL    ADJBCD
7           MOVLW   R1L
8           MOVWF   FSR
9           CALL    ADJBCD
10          GOTO    LOOP
```

二进制数转换成 BCD 码子程序 3

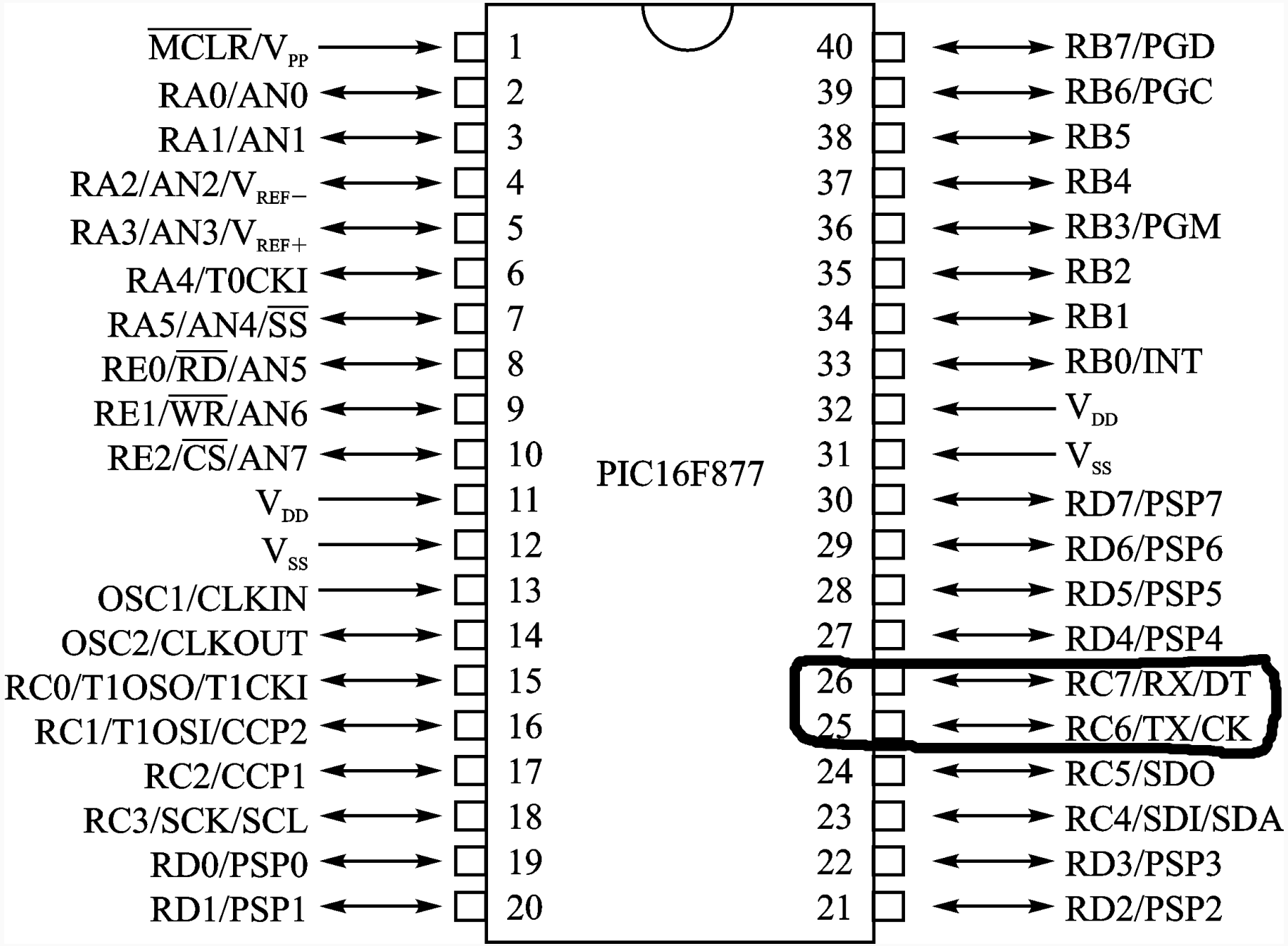
```
1  ADJBCD  MOVLW   03H
2          ADDWF   INDF, W
3          MOVWF   TEMP
4          BTFSC   TEMP, 3
5          MOVWF   INDF
6          MOVLW   30H
7          ADDWF   INDF, W
8          MOVWF   TEMP
9          BTFSC   TEMP, 7
10         MOVWF   INDF
11         RETLW   00H
```

USART 串行通信模块

USART 串行通信模块

PIC 系列芯片中，片内除了含有同步串行口 SSP（SPI，I2C）外，还有一个串行通信接口 SCI。这是一个通用同步 / 异步收发器，简称 USART，它是计算机最常用的通信接口之一。

USART 可工作于如下三种方式：全双工异步方式；半双工同步主控方式；半双工同步从动方式。


PIC16F877
MCLR/VPP 1
RA0/AN0 2
RA1/AN1 3
RA2/AN2/VREF− 4
RA3/AN3/VREF+ 5
RA4/T0CKI 6
RA5/AN4/SS 7
RE0/RD/AN5 8
RE1/WR/AN6 9
RE2/CS/AN7 10
VDD 11
VSS 12
OSC1/CLKIN 13
OSC2/CLKOUT 14
RC0/T1OSO/T1CKI 15
RC1/T1OSI/CCP2 16
RC2/CCP1 17
RC3/SCK/SCL 18
RD0/PSP0 19
RD1/PSP1 20
40 RB7/PGD
39 RB6/PGC
38 RB5
37 RB4
36 RB3/PGM
35 RB2
34 RB1
33 RB0/INT
32 VDD
31 VSS
30 RD7/PSP7
29 RD6/PSP6
28 RD5/PSP5
27 RD4/PSP4
26 RC7/RX/DT
25 RC6/TX/CK
24 RC5/SDO
23 RC4/SDI/SDA
22 RD3/PSP3
21 RD2/PSP2

与 USART 模块相关的寄存器

1. 发送状态兼控制寄存器：TXSTA
2. 接收状态兼控制寄存器：RCSTA
3. USART 发送缓冲寄存器：TXREG
4. USART 接收缓冲寄存器：RCREG
5. 波特率发生器的波特率定义值寄存器：SPBRG
6. 第一中断使能寄存器：PIE1
7. 第一中断标志寄存器：PIR1
8. 中断控制寄存器：INTCON
9. RC 口方向寄存器：TRISC

发送状态兼控制寄存器 TXSTA

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
CSRC	TX9	TXEN	SYNC	-	BRGH	TRMT	TX9D

发送状态兼控制寄存器 TXSTA

Bit0 / TX9D：发送数据的第 9 位（9 位数据帧结构）。

清零 0

当前发送第 9 位数据位为 0；

置位 1

当前发送第 9 位数据位为 1；

发送状态兼控制寄存器 TXSTA

Bit1 / TRMT：发送移位寄存器（TSR）“空”标志位。

清零 0
发送移位寄存器满；

置位 1
发送移位寄存器空。

发送状态兼控制寄存器 TXSTA

Bit2 / BRGH：高波特率选择位。

同步方式下，未用。异步模式下：

清零 0
低速；

置位 1
高速。

发送状态兼控制寄存器 TXSTA

Bit4 / SYNC：USART 同步/异步模式选择位。

清零 0

选择异步模式（USAT）；

置位 1

选择同步模式（USRT）。

发送状态兼控制寄存器 TXSTA

Bit5 / TXEN：发送使能位。

清零 0
关闭发送功能；

置位 1
使能发送功能。

发送状态兼控制寄存器 TXSTA

Bit6 / TX9：发送数据长度选择位。8 位数据加 1 位校验或标识位。

清零 0
8 位数据位发送；

置位 1
9 位数据发送。

发送状态兼控制寄存器 TXSTA

Bit7 / CSRC：时钟源选择位。

异步模式下，未用。同步模式下：

清零 0

选择被控（从属）模式（时钟来自外部输入信号）；

置位 1

选择主控模式（时钟来自内部波特率发生器）。

接收状态兼控制寄存器 RCSTA

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

接收状态兼控制寄存器 RCSTA

Bit0 / RX9D：所接收数据的第 9 位，可作校验位或标识位等。

清零 0
当前接收第 9 位数据位为 0；

置位 1
当前接收第 9 位数据位为 1；

接收状态兼控制寄存器 RCSTA

Bit1 / OERR：溢出标志位。

清零 0

未发生溢出错误；

置位 1

发生了溢出错误。

接收状态兼控制寄存器 RCSTA

Bit2 / FERR：帧格式错误标志位，被动参数。

清零 0

无帧格式错误；

置位 1

有帧格式错误。

接收状态兼控制寄存器 RCSTA

Bit3 / ADDEN：地址匹配检测使能位; 接收数据选择 9 位时，该位才起作用。

清零 0
取消地址匹配检测功能；

置位 1
启用地址匹配检测功能。

接收状态兼控制寄存器 RCSTA

Bit4 / CREN：连续接收使能位。

异步模式下：

清零 0
禁止连续接收功能；
置位 1
使能连续接收功能。

同步模式下：

清零 0
关闭连续接收；
置位 1
使能连续接收，直到该未被清 0 为止。优于 SREN 位。

接收状态兼控制寄存器 RCSTA

Bit5 / SREN：单字节接收使能位。

异步方式下未用，并且在同步从属接收方式下该位也无用。接收完成后该位即被清零。

同步方式下：

清零 0
禁止单字节接收功能；

置位 1
使能单字节接收功能。

接收状态兼控制寄存器 RCSTA

Bit6 / RX9：接收数据长度选择位。

清零 0
选择接收 8 位数据；

置位 1
选择接收 9 位数据。

接收状态兼控制寄存器 RCSTA

Bit7 / SPEN：串行端口使能位。

清零 0

禁止串行端口工作；

置位 1

允许串行端口工作。此时，RC7 和 RC6 作为 USART 的接收发送引脚。

USART 发送缓冲寄存器 TXREG

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0

每次发送的数据都是通过写入该缓冲器来实现的。

USART 接收缓冲寄存器 RCREG

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0

每次接收到的数据都可从该缓冲器读取出来的。

SPBRG 波特率寄存器

SPBRG 寄存器的设定值（0-255）与波特率成反比关系。在同步方式下，波特率仅由该寄存器来决定；而在异步方式下，则由 BRGH 位（TXSTA 寄存器的 bit2）和该寄存器共同确定。

Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
对于波特率发生器产生波特率的定义值							

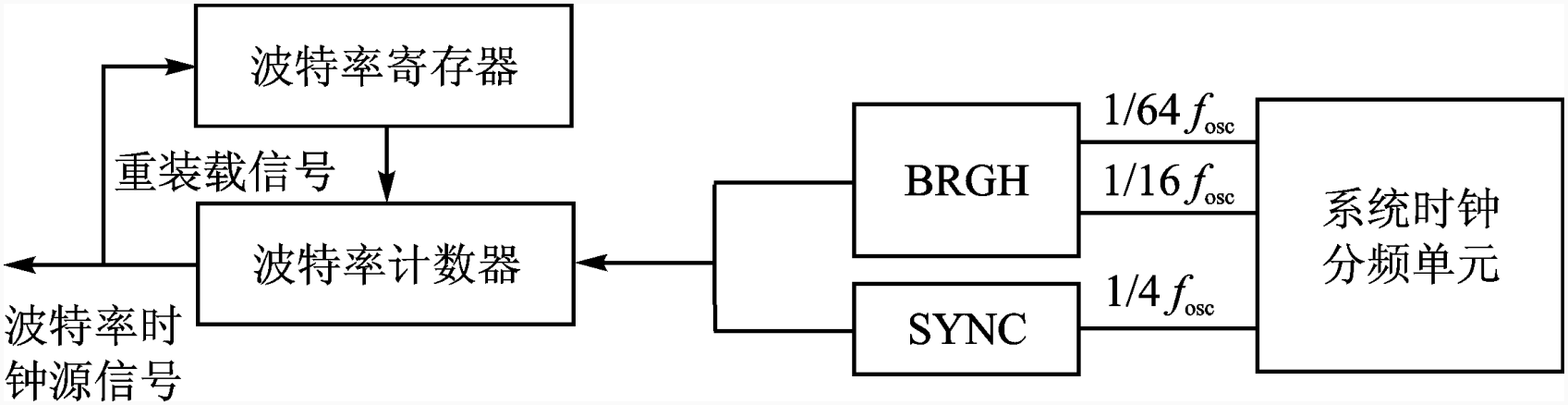
USART 波特率发生器

USART 模块带有一个波特率发生器 BRG（baud rate generator），用以产生串行传送所需的时钟，它支持 USART 的同步方式和异步方式。在异步方式下，BRGH 位用来控制波特率。在同步方式下，BRGH 位不起作用。

USART 波特率发生器

波特率发生器的逻辑结构如下图。其核心实际是一个递减的 8 位二进制计数器，其计数初始值是由寄存器 SPBRG 装入，在每次递减计数器到达 00H 之后的下一个计数脉冲到来时进行装入。系统时钟经过 6 位分频器后作为传递计数器的计数脉冲，其分频比可以由 BRGH 位和 SYNC 位设定为 1：4、1:16、或 1：64。

波特率时钟发生器示意图



同步方式波特率计算方式

波特率 $= f_{OSC}/[4(N+1)]$

$N = f_{OSC}/(4x$ 波特率 $) - 1$

注意，N 为 SPBRG 寄存器的初始值；BRGH 必须设置为 0，BRGH 为 1 无用。

异步方式波特率计算方式

BRGH = 0 时：

波特率 $= f_{OSC}/[64(N+1)]$

$N = f_{OSC}/(64x$ 波特率 $) - 1$

BRGH = 1 时：

波特率 $= f_{OSC}/[16(N+1)]$

$N = f_{OSC}/(16x$ 波特率 $) - 1$

异步方式波特率计算方式

例题

在某应用系统中，采用 4800 波特进行异步通信。假设单片机时钟频率为 fosc = 20MHZ，低速方式（BRGH=0）。

USART 的异步模式

在异步工作方式下，串行通信接口采用标准的不归零 (NRZ) 格式，即 l 位起始位、8 位或 9 位数据位和 l 位停止位，最常用的数据格式是 8 位。

片内提供的 8 位波特率发生器 BRG 可以用来驱动来自振荡器的时钟产生标准的波特率频率。

USART 接收和发送顺序是从最低位 (LSB) 开始。

USART 的发送器和接收器在功能上是独立的，但它们所用的的数据格式和波待率是相同的。

USART 的异步模式

波特率发生器可以根据 TXSTA 寄存器的 BRGH 位 (即 D0) 的设置产生两种不同的移位速率: 对系统时钟 16 分频和 64 分频的波特率时钟。

USART 硬件不支持奇偶校验，但可以用软件实现 (并存储作为第 9 位数据)。

在 CPU 处于休眠方式时, USART 不能用异步方式工作。

通过对 TXSTA 寄存器的 SYNC 位清零, 可选择 USART 异步工作方式。USART 异步工作方式由以下一些重要部件组成：波特率发生器 BRG；采样电路；异步发送器；异步接收器。

USART 异步发送模式

发送器的核心是发送移位寄存器 TSR。

移位寄存器从读/写发送缓冲器 TXREG 获得数据。

TXREG 的数据是用软件把要发送的数据送入的，等到把在这之前装入的停止位发送后，立即把 TXREG 中新的发送数据送入 TSR。

一旦 TXREG 把数据送入 TSR 后，TXREG 就为“空”状态，同时把发送中断标志位 TXIF 置 1。

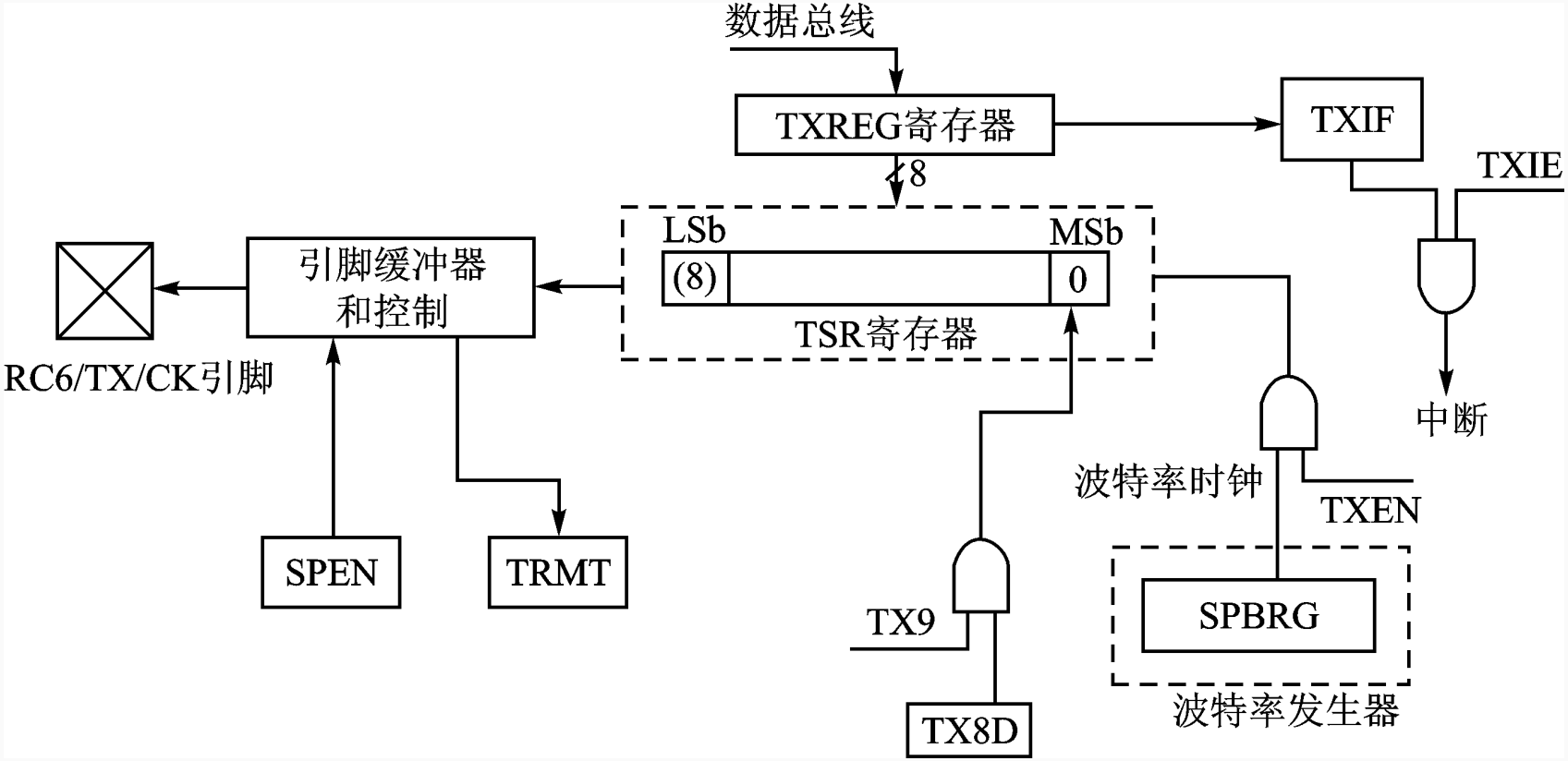
USART 异步发送模式

要让 SCI 工作于发送方式，必须先把 TXSTA 状态寄存器中的 TXEN 位 (即 TXSTA 的 D5) 置 I，不过真正的发送工作要等到 TXREG 已放有发送数据以及波特率发生器 BRG 发出移位脉冲后才开始。

也可以先把发送数据送入 TXREG，然后再把 TXEN 位置 I 来启动发送工作。

一般当第一次启动发送时 TSR 是空的，所以对 TXREG 的传输将立即把数据送入 TSR 中,TXREG 缓冲器变空。由此就可以用“背靠背”地连续传送两个数据。

USART 异步发送结构示意图



USART 异步发送步骤

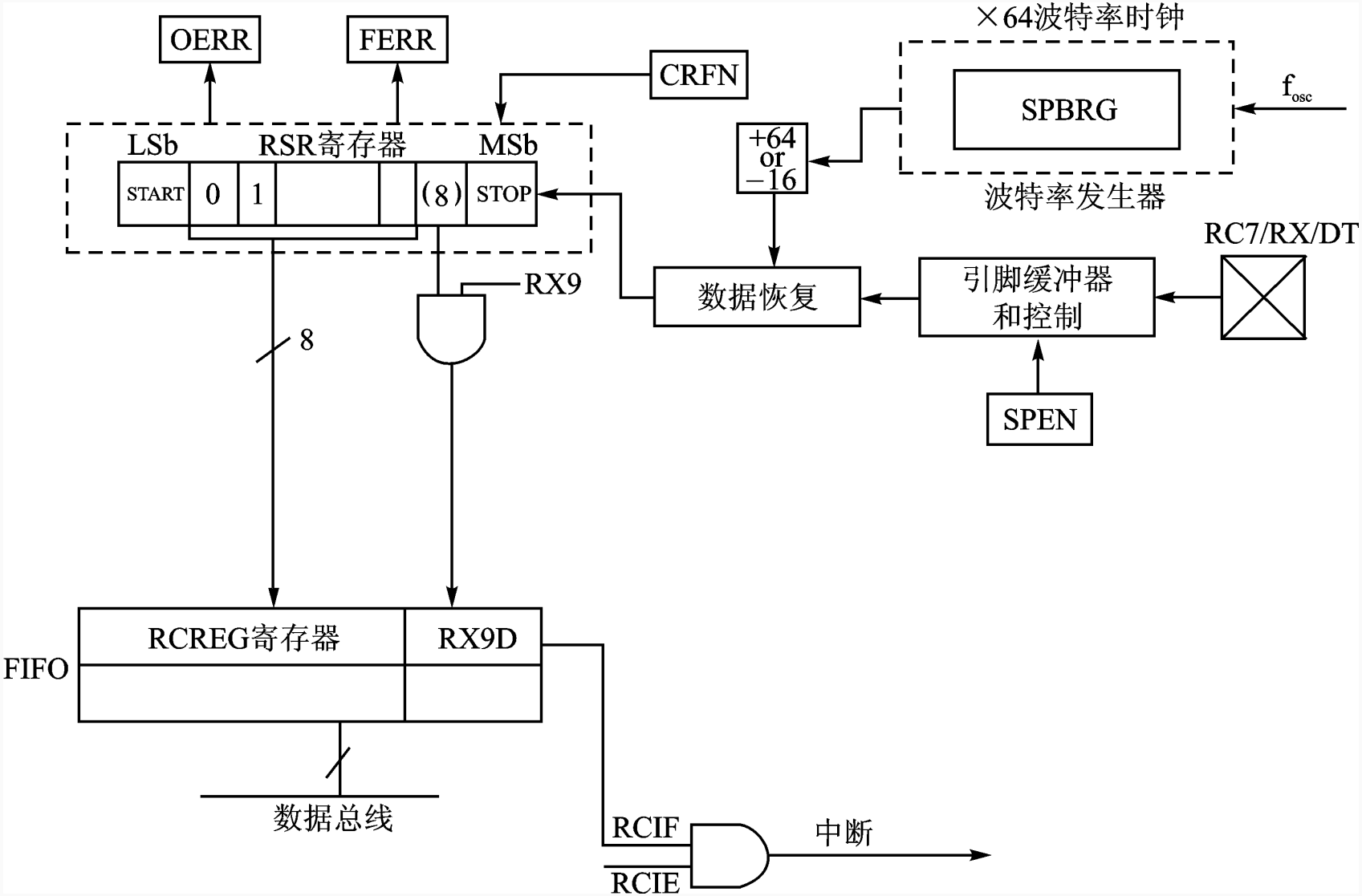
1. 选择合适的波特率，对 SPBRG 进行初始化，如果要求是高速波特率，置 BRGH=1；
2. 置 SYNC=0 和 SPEN=1，使其工作在异步串行口工作方式；
3. 若需要中断，置 TXIE=1；
4. 若要传送 9 位数据, 置 TX9=1；
5. 置 TXEN=1，使 SCI 工作在发送器方式；
6. 若选择发送 9 位数据，第 9 位应该写入 TX9D 位；
7. 把 8 位发送数据送入 TXREG 缓冲器 (启动发送)；
8. 如果使用中断，确保 GIE 和 PEIE 置位。

USART 异步接收模式

主要是由接收移位寄存器 RSR 和接收寄存器 RCREG 构成，串行信号从 RC7 / RX / DT 引脚接收，送入移位寄存器 RSR。

一旦收到停止位，RSR 就将收到的 8 位数据装载到 RCREG 中。同时，RCIF 置 1，表示收到一个数据，当 RCREG 被读出时 RCIF 被清零。

USART 异步接收结构示意图



USART 异步接收模式注意问题

如 RSR 将数据装载到 RCREG 时，RCREG 内已有 1 个数据，则产生数据传送溢出错误。OERR 置位。RSR 中数据将不能装入 RCREG；

OERR 置位后,RSR 不会接收新的数据。只有将 OERR 清 0 后才能接收新数据。OERR 清零的方法是将 CREN 清零再置位；

如收到停止位为 0，则 FERR 置位；

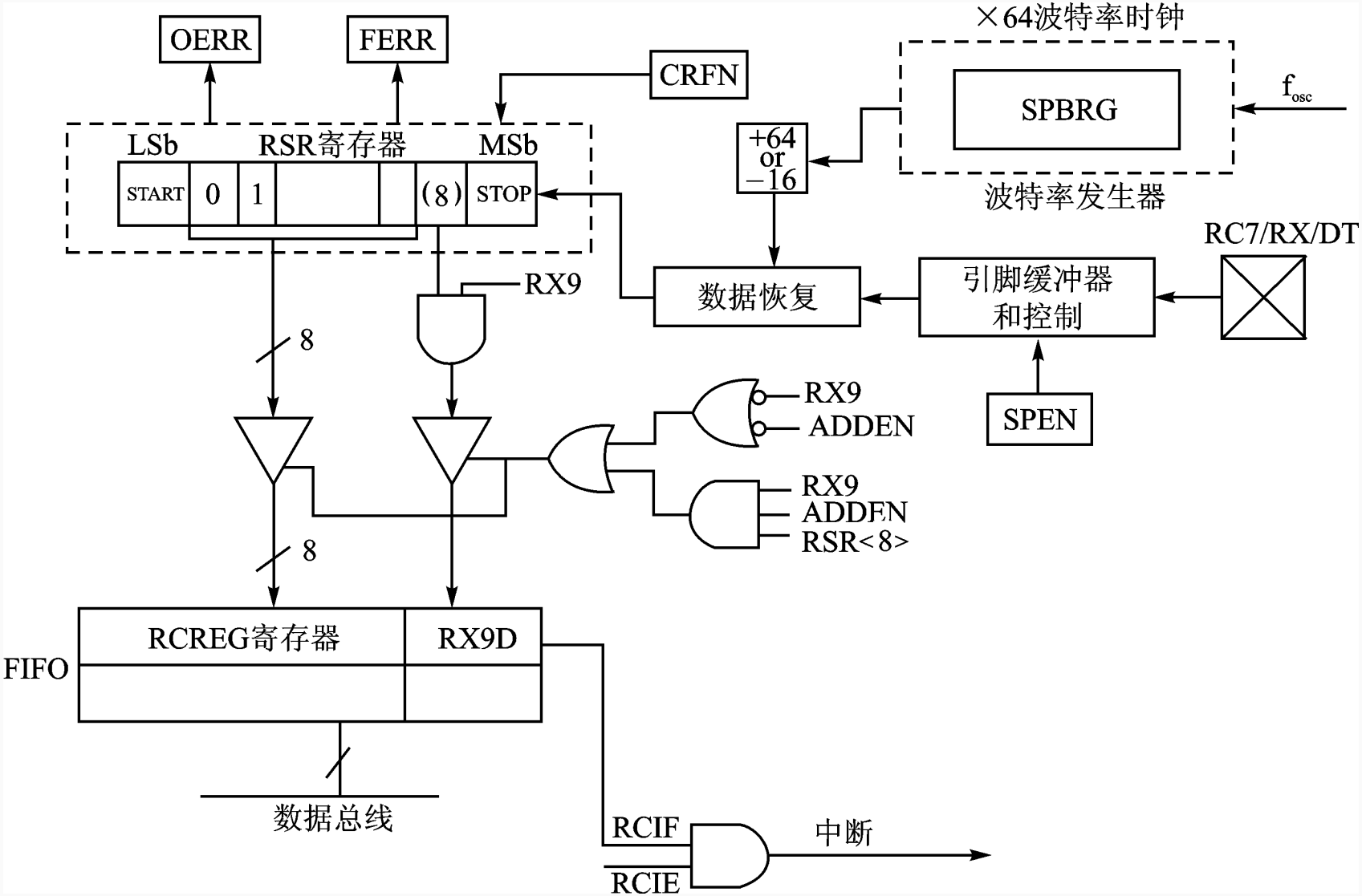
USART 异步接收步骤

1. 初始化 SPBRG 寄存器，选择合适的波特率；
2. 置 SYNC=0 和 SPEN=1，使其工作在异步串行口工作方式；
3. 若需要中断，置位 RCIE；
4. 若要接收 9 位数据，置位 RX9；
5. 置位 CREN，使 SCI 工作在接收器方式；
6. 当接收完成数据后，中断标志位 RCIF 置 1，如果 RCIE=1，便产生中断；
7. 如果设定接收 9 位数据，读 RCSTA 获取第 9 位数据并判断在接收操作中是否发生错误；
8. 读 RCREG 中的 8 位数据；
9. 如果发生某种接收错误，通过 CREN 清零以清除错误；

带地址检测的 9 位异步接收方式

主要由 RSR 移位寄存器，RCREG 寄存器即波特率发生器 BRG 组成。工作过程如下：从 RC7/RX31 引脚输入数据，在波特率时钟作用下，一位一位移入 RSR 寄存器，一旦收到停止位，就将收到的第 9 位数据分别装入 RCREG 和 RX9D 中，并置 RCIF=1。再读取 RCREG 中的数据时，自动将 RCIF 清零。

USART 带地址检测 9 位异步接收结构图



USART 带地址检测 9 位接收步骤

1. 初始化 SPBRG 寄存器，选择合适的波特率；
2. 置 SYNC=0 和 SPEN=1，使其工作在异步串行口工作方式；
3. 若需要中断，置位 RCIE；
4. 置位 RX9，允许接收第 9 位；
5. 置位 ADDEN，允许地址检测；
6. 置位 CREN，使 SCI 工作在接收器方式；
7. 当接收完成数据后，中断标志位 RCIF 置 1，如果 RCIE=1，便产生中断；
8. 读 RCSTA 获取第 9 位数据并判断在接收时是否发生错误；
9. 读 RCREG 中的 8 位数据，判断设备是否被寻址；
10. 如果发生某种接收错误，通过清零 CREN 以清除错误；

USART 同步模式

USART 的同步模式是指进行通信双方之间，除了有数据传输线以外，还有一条时钟专用线。起着同步发送/接收的作用。在同步方式下，数据格式可以使用 8 位或 9 位。由于有专用的时钟线同步，所以在串行字符中不再需要起始位和停止位。在同步方式下只能工作在半双工方式下。

USART 同步主控发送模式

USART 同步主控发送方式与异步发送方式基本相同，发送器的核心是串行发送移位寄存器 (TSR) 和发送寄存器 TXREG。用户将要发送的数据装入 TXREG，一旦 TSR 中空就会从 TXREG 中读出要发送数据装入 TSR。同时，TXIF 被置位。一旦 TXIF 置位后只有当有新的数据写入 TXREG 后，才能使 TXIF 清零。当 TSR 为空时 TRMT 置位，所以通过 TRMT 可查询 TSR 是否为空。

USART 同步主控发送步骤

1. 选择合适的波特率，初始化 SPBRG；
2. 置 SYNC=1、SPEN=1 和 CSRC=1，使 USART 工作在同步主控方式；
3. 若需要中断，置 TXIE=1；
4. 若要发送 9 位数据，置位 TX9；
5. 置位 TXEN，使 USART 工作在发送器方式；
6. 若选择发送 9 位数据方式，第 9 位应该先写入 TX9D 位；
7. 把 8 位发送数据送入 TXREG 缓冲器，启动发送；
8. 如果使用中断，确保中断控制寄存器的 D7(GIE) 和 D6(PEIE) 置位。

USART 主控接收模式

要使用 RSART 工作在同步主控接收模式时，首先选择同步方式，然后把 SREN 位或 CREN 位置位，即可进入同步主控接收状态，DT 数据线上的信号在时钟的下降沿被采样。如果 SREN = 1，仅接收一个字节。如果 CREN = 1，则可连续地接收数据，直到 CREN 被清零为止。如果 SREN 和 CREN 都被置位，则 CREN 状态优先于 SREN 状态，进行连续接收。

USART 同步主控接收步骤

1. 选择合适的波特率, 初始化 SPBRG；
2. 设置 SYNC=1、SPEN=1 和 CSRC=1，工作在同步主控方式；
3. 确保 CREN 和 SREN 清零；
4. 如果需要使用中断，置位 RCIE；
5. 如果 9 位模式接收，置位 RX9；
6. 如果单字节接收，置位 SREN；如果连续接收，置位 CREN；
7. 接收完成，中断标志位置位，如果 RCIF=1，产生中断请求；
8. 读 RCSTA 寄存器，获取第 9 位，并判断接收时是否有错误；
9. 读 RCREG 寄存器，得 8 位数据；
10. 如果出错，通过 CREN 清零清除错误；
11. 如果采用中断，确保中断控制寄存器的 GIE 和 PEIE 置位。

USART 同步从动发送模式

USART 的同步从动方式和同步主控方式的区别就是其时钟信号 CK 由外部提供，也就是由对方提供，因此就是本机在睡眠状态下仍可进行通信。

USART 同步从动发送步骤

1. 置位 SYNC 和 SPEN，复位 CSRC，使 USART 工作在同步从动方式；
2. 复位 CREN 和 SREN；
3. 若需要使用中断，置位 TXIE；
4. 若要使用 9 位数据传输模式，置位 TX9；
5. 置位 TXEN，使 USART 工作在发送器方式；
6. 若选择 9 位数据传输模式，第 9 位写入 TX9D 位；
7. 写入数据到 TXREG 缓冲器，启动发送；
8. 如果使用中断，确保中断控制寄存器的 GIE（D7）和 PEIE（D6）置位。

USART 同步从动接收模式

同步从动接收和同步主控接收的操作基本上是一样的，只是当 CPU 处于休眠方式下有所区别。另外，在从动接收方式下，没有用到 SREN 位。

USART 同步从动接收步骤

1. 设置 SYNC=1、SPEN=1 和 CSRC=0, 使其工作在同步从动串行口工作方式。
2. 若需要使用中断，置位 RCIE；
3. 若要使用 9 位数据传输模式，置位 TX9；
4. 置位 CREN，使 USART 工作在接收器方式；
5. 完成接收，标志位 RCIF 置位，如果中断开放，将产生中断请求；
6. 读 RCSTA 寄存器，获取第 9 位数据，判断在接收期间是否出错；
7. 读 RCREG 寄存器，获取 8 位接收数据；
8. 如果出错，通过 CREN 清零清除出错标志位；
9. 如果使用中断，确保中断控制寄存器的 GIE 和 PEIE 置位。

应用实例

例题 9-6

主、从单片机采用如图 9-33 所示硬件电路，基于 USART 同步/异步串行通信，通过电平变化中断实现双机同步 LED 快慢速加减计数显示。

主机程序头

```
1 LIST      P=16F877
2 INCLUDE   "P16F877.INC"
3 COUNTER   EQU      70H
4 COUNTER   EQU      71H
5           ORG      0000H
6           NOP
7           GOTO     ST
```

主机中断程序

```
1          ORG     0004H
2          BTFSC   INTCON, RBIF
3          GOTO    LRBIF
4          MOVF    RCREG, W
5          MOVWF   COUNTER
6          BCF     PIR1, RCIF
7          GOTO    RE
```

主机中断程序

```
1  LRBIF    BCF     INTCON, RBIF
2  BB4      BTFSS   PORTB, 4
3           GOTO    BB5
4           CALL    DELAY10MS
5           BTFSS   PORTB, 4
6           GOTO    BB5
7  PP4      BTFSC   PORTB, 4
8           GOTO    PP4
9           CALL    DELAY10MS
10          BTFSC   PORTB, 4
11          GOTO    PP4
12          CLRF    COUNTER
13          GOTO    RE
```

主机中断程序

```
1  BB5      BTFSS   PORTB, 5
2           GOTO    BB6
3           CALL    DELAY10MS
4           BTFSS   PORTB, 5
5           GOTO    BB6
6  PP5      BTFSC   PORTB, 5
7           GOTO    PP5
8           CALL    DELAY10MS
9           BTFSC   PORTB, 5
10          GOTO    PP5
11          MOVLW   01H
12          MOVWF   COUNTER
13          GOTO    RE
```

主机中断程序

```
1  BB6      BTFSS   PORTB, 6
2           GOTO    BB7
3           CALL    DELAY10MS
4           BTFSS   PORTB, 6
5           GOTO    BB7
6  PP6      BTFSC   PORTB, 6
7           GOTO    PP6
8           CALL    DELAY10MS
9           BTFSC   PORTB, 6
10          GOTO    PP6
11          MOVLW   02H
12          MOVWF   COUNTER
13          GOTO    RE
```

主机中断程序

```
1  BB7     BTFSS   PORTB, 7
2          GOTO    RE
3          CALL    DELAY10MS
4          BTFSS   PORTB, 7
5          GOTO    RE
6  PP7     BTFSC   PORTB, 7
7          GOTO    PP7
8          CALL    DELAY10MS
9          BTFSC   PORTB, 7
10         GOTO    PP7
11         MOVLW   03H
12         MOVWF   COUNTER
13         GOTO    RE
```

主机中断程序

```
1 RE      BSF     COUNT, 0
2         CLRF    PORTD
3         MOVLW   00H
4         CALL    OUTSHU
5         RETFIE
```

主机主程序

```
1  ST      BSF     STATUS, RP0
2          MOVLW   B'10000000'
3          MOVWF   TRISC
4          MOVLW   19H
5          MOVWF   SPBRG
6          MOVLW   B'00100100'
7          MOVWF   TXSTA
8          MOVLW   00H
9          MOVWF   TRISD
10         MOVLW   0F0H
11         MOVWF   TRISB
12         BSF     PIE1, RCIE
```

主机主程序

```
1        BCF     STATUS, RP0
2        MOVLW   B'10010000'
3        MOVWF   RCSTA
4        MOVLW   B'11001000'
5        MOVWF   INTCON
6        CLRF    PORTD
7        CLRF    COUNTER
8        CLRF    COUNT
```

主机主程序

```
1  LOOP    BTFSS   COUNT, 0
2          GOTO    $+3
3          CLRF    COUNT
4          CLRF    PORTD
5          MOVF    COUNTER, W
6          ADDWF   PCL, F
7          GOTO    XSH1
8          GOTO    XSH2
9          GOTO    XSH3
10         GOTO    XSH4
```

主机显示方式

```
1 ;-------------------------------------------------
2 ; 慢速自动加1（第一种显示方式）
3 ;-------------------------------------------------
4 XSH1      INCF     PORTD, F
5           MOVF     PORTD, W
6           CALL     OUTSHU
7           CALL     DELAY1S
8           GOTO     LOOP
```

主机显示方式

```
1 ;-------------------------------------------------
2 ; 慢速自动减1（第二种显示方式）
3 ;-------------------------------------------------
4 XSH2      INCF     PORTD, F
5           MOVF     PORTD, W
6           CALL     OUTSHU
7           CALL     DELAY1S
8           GOTO     LOOP
```

主机显示方式

```
1 ;-------------------------------------------------
2 ; 快速自动加1（第三种显示方式）
3 ;-------------------------------------------------
4 XSH3      INCF     PORTD, F
5           MOVF     PORTD, W
6           CALL     OUTSHU
7           CALL     DELAY10MS
8           GOTO     LOOP
```

主机显示方式

```
1 ;-------------------------------------------------
2 ; 快速自动减1（第三种显示方式）
3 ;-------------------------------------------------
4 XSH4      INCF     PORTD, F
5           MOVF     PORTD, W
6           CALL     OUTSHU
7           CALL     DELAY10MS
8           GOTO     LOOP
```

主机数据发送

```
1 ;-----------------------------------------------------
2 ; 数据发送子程序
3 ;-----------------------------------------------------
4 OUTSHU   MOVWF    TXREG
5          BSF      STATUS, RP0
6 LPTX     BTFSS    TXSTA, TRMT
7          GOTO     LPTX
8          BCF      STATUS, RP0
9          RETURN
```

主机 1s 延时子程序

```
1  DELAY1S  MOVLW   06H
2           MOVWF   20H
3  L1S1     MOVLW   0EBH
4           MOVWF   21H
5  L1S2     MOVLW   0ECH
6           MOVWF   22H
7  L1S3     DECFSZ  22H, F
8           GOTO    L1S3
9           DECFSZ  21H, F
10          GOTO    L1S2
11          DECFSZ  20H, F
12          GOTO    L1S1
13          RETURN
```

主机 10ms 延时子程序

```
1  DELAY10MS  MOVLW    0DH
2             MOVWF    20H
3  L10MS1     MOVLW    0FFH
4             MOVWF    21H
5  L10MS2     DECFSZ   21H, F
6             GOTO     L10MS2
7             DECFSZ   20H, F
8             GOTO     L10MS1
9             RETURN
10            END
```

从机程序头

```
1 ;-----------------------------------------------------
2       LIST P=16F877
3       INCLUDE "p16f877a.inc"
4 ;-----------------------------------------------------
5 COUNTER  EQU      70H
6 COUNT    EQU      71H
7          ORG      0000H
8          NOP
9          GOTO     ST
```

从机中断服务程序

```
1         ORG     0004H
2         BTFSC   INTCON, RBIF
3         GOTO    LRBIF
4         MOVF    RCREG, W
5         MOVWF   PORTD
6         BCF     PIR1, RCIF
7         GOTO    RE
```

从机中断服务程序

```
1  LRBIF    BCF     INTCON, RBIF
2  BB4      BTFSS   PORTB, 4
3           GOTO    BB5
4           CALL    DELAY10MS
5           BTFSS   PORTB, 4
6           GOTO    BB5
7  PP4      BTFSC   PORTB, 4
8           GOTO    PP4
9           CALL    DELAY10MS
10          BTFSC   PORTB, 4
11          GOTO    PP4
12          CLRF    COUNTER
13          MOVF    COUNTER, W
14          CALL    OUTSHU
15          GOTO    RE
```

从机中断服务程序

```
1  BB5      BTFSS   PORTB, 5
2           GOTO    BB6
3           CALL    DELAY10MS
4           BTFSS   PORTB, 5
5           GOTO    BB6
6  PP5      BTFSC   PORTB, 5
7           GOTO    PP5
8           CALL    DELAY10MS
9           BTFSC   PORTB, 5
10          GOTO    PP5
11          MOVLW   01H
12          MOVWF   COUNTER
13          CALL    OUTSHU
14          GOTO    RE
```

从机中断服务程序

```
1  BB6      BTFSS   PORTB, 6
2           GOTO    BB7
3           CALL    DELAY10MS
4           BTFSS   PORTB, 6
5           GOTO    BB7
6  PP6      BTFSC   PORTB, 6
7           GOTO    PP6
8           CALL    DELAY10MS
9           BTFSC   PORTB, 6
10          GOTO    PP6
11          MOVLW   02H
12          MOVWF   COUNTER
13          CALL    OUTSHU
14          GOTO    RE
```

从机中断服务程序

```
1  BB7      BTFSS   PORTB, 7
2           GOTO    RE
3           CALL    DELAY10MS
4           BTFSS   PORTB, 7
5           GOTO    RE
6  PP7      BTFSC   PORTB, 7
7           GOTO    PP7
8           CALL    DELAY10MS
9           BTFSC   PORTB, 7
10          GOTO    PP7
11          MOVLW   03H
12          MOVWF   COUNTER
13          CALL    OUTSHU
14          GOTO    RE
```

从机中断服务程序

```
1  RE      BSF     COUNT, 0
2          CLRF    PORTD
3          MOVLW   00H
4          CALL    OUTSHU
5          RETFIE
```

从机主程序

```
1  ST      BSF     STATUS, RP0
2          MOVLW   B'10000000'
3          MOVWF   TRISC
4          MOVLW   19H
5          MOVWF   SPBRG
6          MOVLW   B'00100100'
7          MOVWF   TXSTA
8          MOVLW   00H
9          MOVWF   TRISD
10         MOVLW   0F0H
11         MOVWF   TRISB
12         BSF     PIE1, RCIE
```

从机主程序

```
1        BCF     STATUS, RP0
2        MOVLW   B'10010000'
3        MOVWF   RCSTA
4        MOVLW   B'11001000'
5        MOVWF   INTCON
6        CLRF    PORTD
7        CLRF    COUNTER
8        GOTO    $-1
```

从机数据发送子程序

```
1  ;-----------------------------------------------------
2  ; 数据发送子程序
3  ;-----------------------------------------------------
4  OUTSHU   MOVWF    TXREG
5           BSF      STATUS, RP0
6  LPTX     BTFSS    TXSTA, TRMT
7           GOTO     LPTX
8           BCF      STATUS, RP0
9           RETURN
```

从机 10ms 延时子程序

```
;-------------------------------------------------
; 10ms延时子程序
;-------------------------------------------------
DELAY10MS  MOVLW     0DH
           MOVWF     20H
L10MS1     MOVLW     0FFH
           MOVWF     21H
L10MS2     DECFSZ    21H, F
           GOTO      L10MS2
           DECFSZ    20H, F
           GOTO      L10MS1
           RETURN
           END
```