第 19 章 I²C™

目录

本章包括下列主题：

19.1 概述

I²C 模块是用于同其他外设或单片机器件进行通信的串行接口。这些外设可以是串行 EEPROM、显示驱动器和 A/D 转换器等。

I²C 模块可在以下任意 I²C 系统中工作：

- 作为从器件
- 在单主机系统中作为主器件 （从器件也可同时工作）
- 在多主机系统中作为主 / 从器件 （提供总线冲突检测和仲裁）

I²C 模块包含相互独立的 I²C 主器件逻辑和 I²C 从器件逻辑，它们根据各自的事件产生中断。在多主机系统中，软件被简单地分为主器件控制程序和从器件控制程序。

当 I²C 主器件逻辑工作时，从器件逻辑也保持在工作状态，检测总线的状态，并可从其自身 （单主机系统中）或从其他主器件 （多主机系统中）接收报文。在多主机总线仲裁期间，不会丢失任何报文。

在多主机系统中，当检测到与系统中的其他主器件存在总线冲突时，模块提供了一种方法来终止报文，然后重新发送报文。

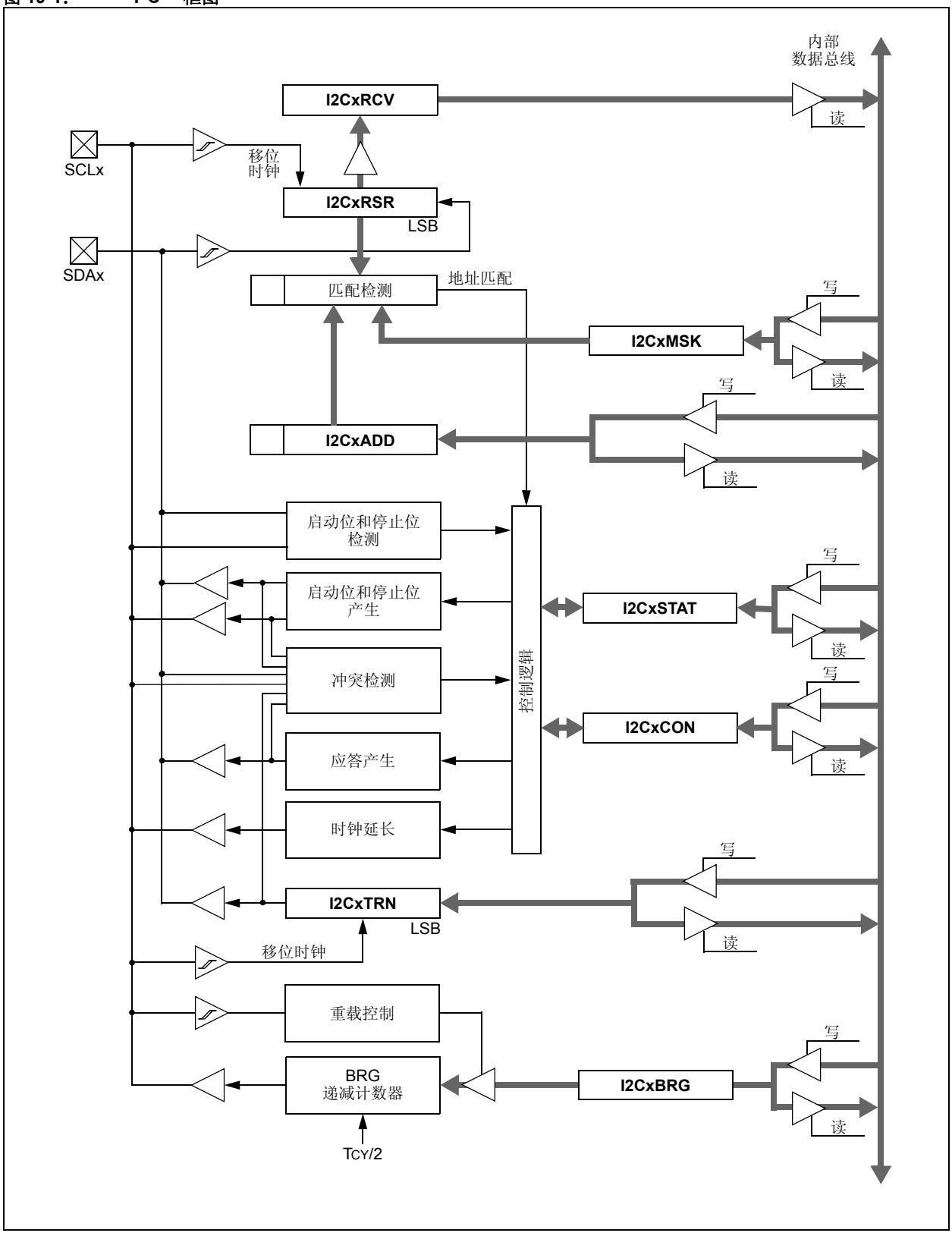
I²C 模块中包含一个波特率发生器（Baud Rate Generator，BRG）。I²C 波特率发生器并不耗用器件中的其他定时器资源。

I²C 模块具有以下主要特性：

- 主器件和从器件逻辑相互独立
- 支持多主机系统，可以防止在仲裁时丢失报文
- 在从模式下可检测 7 位和 10 位器件地址，并可配置地址掩码
- 检测 I²C 协议中定义的广播呼叫地址
- 具有总线转发器模式，允许模块作为从器件接收所有报文，与地址无关
- 具有自动 SCLx 时钟延长功能，为处理器提供延时以响应从器件的数据请求
- 支持 100 kHz 和 400 kHz 总线规范
- 支持智能平台管理接口 （Intelligent Platform Management Interface， IPMI）标准

图 19-1 给出了 I²C 模块的框图。

图 19-1：　I²C™ 框图


内部
数据总线
I2CxRCV
读
SCLx
移位
时钟
I2CxRSR
LSB
SDAx
匹配检测
地址匹配
写
I2CxMSK
读
写
I2CxADD
读
启动位和停止位
检测
写
启动位和停止位
产生
I2CxSTAT
读
冲突检测
写
控制逻辑
I2CxCON
读
应答产生
时钟延长
写
I2CxTRN
LSB
读
移位时钟
重载控制
写
BRG
递减计数器
I2CxBRG
读
TCY/2

19.2 I²C 总线特性

I²C 总线是二线制串行接口。图 19-2 给出了 PIC24H 器件和 24LC256 I²C 串行 EEPROM 之间的 I²C 连接示意图，这是任何 I²C 接口的典型示例。

接口使用一个综合协议，以确保数据的可靠发送和接收。进行通信时，一个器件作为主器件启动总线上的数据传输，并产生时钟信号以允许传输，而另一个（几个）器件作为从器件对数据传输作出响应。时钟线 SCLx 为主器件的输出、从器件的输入，虽然从器件偶尔也可驱动 SCLx 线。数据线 SDAx 可以是主器件和从器件的输出和输入。

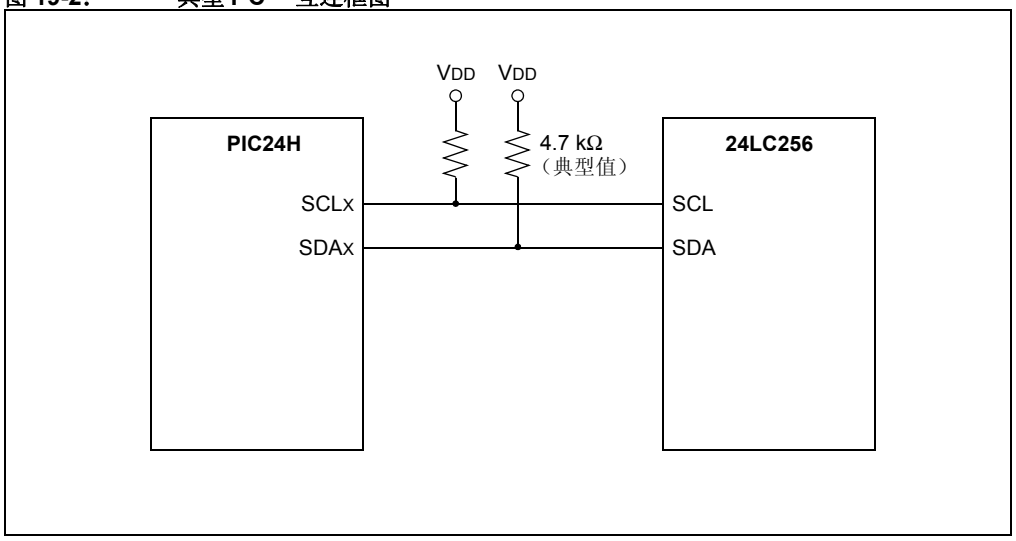
因为 SDAx 和 SCLx 线是双向的，所以驱动 SDAx 和 SCLx 线的器件的输出级必须为漏极开路输出，以便执行总线的线“与”功能。使用外接上拉电阻以确保在没有器件将数据线拉低时线路能保持高电平。

在 I²C 接口协议中，每个器件都有一个地址。当某个主器件要启动数据传输时，它首先发送它要与之进行通信的器件地址。所有的器件均会监听，看是否是自己的地址。在该地址中，bit 0 指定主器件是要自从器件读数据还是向从器件写数据。在数据传输期间，主器件和从器件始终处于相反的工作模式（发送器 / 接收器）。即，可以将它们看作是工作于以下两种关系之一：

- 主器件——发送器，从器件——接收器
- 从器件——发送器，主器件——接收器

在两种情况中，均由主器件产生 SCLx 时钟信号。

图 19-2： 典型 I²C™ 互连框图



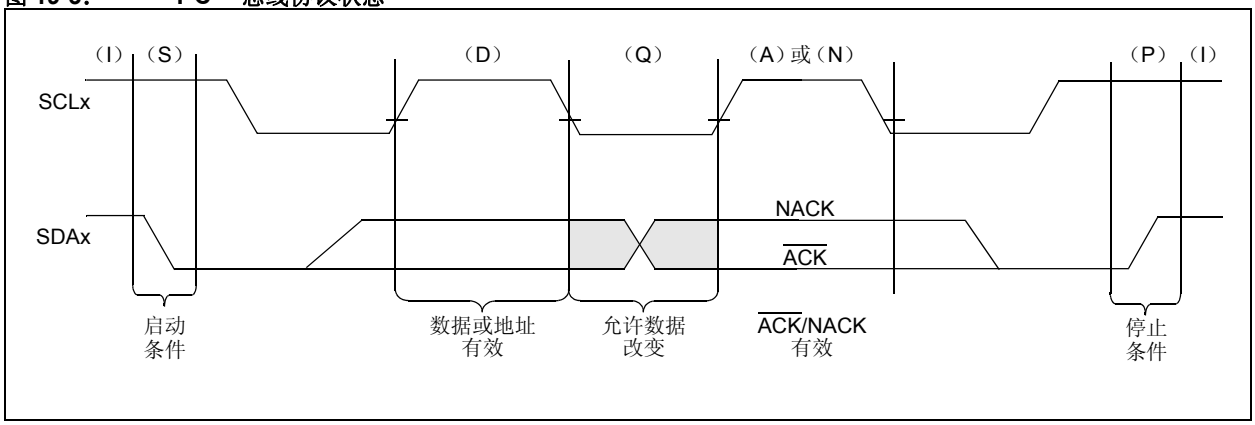
19.2.1 总线协议

I²C 总线协议定义如下：

- 只有在总线不忙时才能启动数据传输。
- 在数据传输期间，只要 SCLx 时钟线为高电平，数据线就必须保持稳定。在 SCLx 时钟线为高电平时，数据线的电平变化将被解析为启动或停止条件。

相应地，定义了如图 19-3 所示的总线条件。

图 19-3： I²C™ 总线协议状态



19.2.1.1 启动数据传输 （S）

在总线空闲状态之后，当时钟（SCLx）为高电平时，SDAx 线从高电平跳变到低电平产生启动条件。所有数据传输都必须以启动条件开始。

19.2.1.2 停止数据传输 （P）

当时钟（SCLx）为高电平时，SDAx 线从低电平跳变到高电平产生停止条件。所有数据传输都必须以停止条件结束。

19.2.1.3 重复启动 （R）

在等待状态之后，当时钟（SCLx）为高电平时，SDAx 线从高电平跳变到低电平产生重复启动条件。重复启动条件使主器件可在不放弃总线控制的情况下更改总线方向或所寻址的从器件。

19.2.1.4 数据有效 （D）

在启动条件之后，如果 SDAx 线在时钟信号的高电平期间保持稳定，则 SDAx 线的状态代表有效数据。每个 SCLx 时钟传送一位数据。

19.2.1.5 应答 （A）或不应答 （N）

所有的数据字节传输都必须由接收器应答 （$\overline{\text{ACK}}$）或不应答 （NACK）。接收器将 SDAx 线拉低则发出 $\overline{\text{ACK}}$，释放 SDAx 线则发出 NACK。应答为一位周期，使用一个 SCLx 时钟。

19.2.1.6 等待 / 数据无效 （Q）

数据线上的数据必须在时钟信号的低电平期间改变。器件也可以通过在 SCLx 上驱动低电平来延长时钟的低电平时间，使得总线处于等待状态。

19.2.1.7 总线空闲 （I）

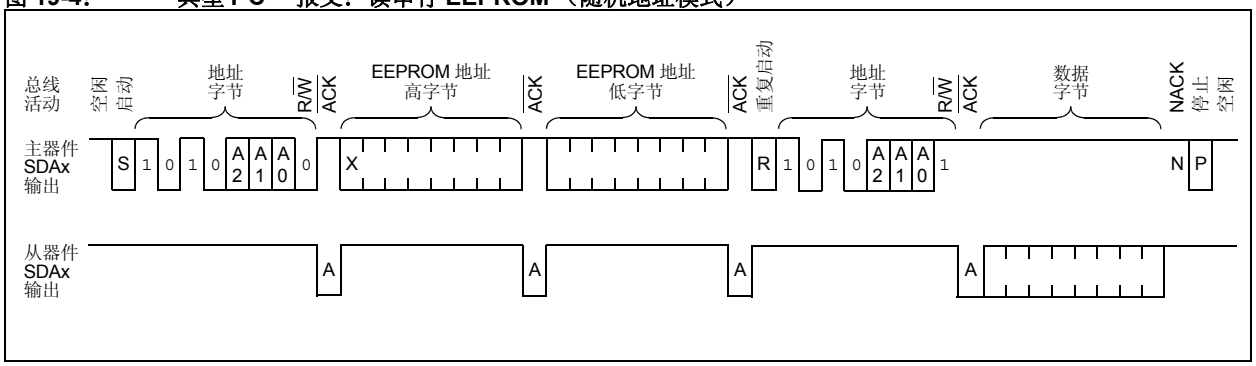
在停止条件之后、启动条件之前的时间内，数据线和时钟线都保持高电平。

19.2.2 报文协议

图 19-4 所示为典型的 I²C 报文。在该示例中，报文将从 24LC256 I²C 串行 EEPROM 读取指定字节。 PIC24H 器件将作为主器件，而 24LC256 器件作为从器件。

图 19-4 给出了由主器件驱动的数据和由从器件驱动的数据，要考虑到组合 SDAx 线上的数据是将主器件数据和从器件数据线“与”后的数据。主器件控制协议并产生协议序列。从器件仅在特别确定的时间驱动总线。

图 19-4： 典型 I²C™ 报文：读串行 EEPROM （随机地址模式）



19.2.2.1 启动报文

每个报文均由启动条件启动并由停止条件终止。在启动和停止条件之间传输的数据字节数由主器件确定。根据系统协议的定义，报文中的字节可能具有特殊的含义，如“器件地址字节”或“数据字节”。

19.2.2.2 寻址从器件

在图 19-4 中，第一个字节是器件地址字节，它必须是任何 I²C 报文的第一部分。它包含一个器件地址和一个 R/W 状态位。关于地址字节格式的更多信息，请参见《dsPIC30F 系列参考手册》中的**附录 A “I2C™ 概述”**（DS70074）（请访问 Microchip 网站 www.microchip.com 获得）。请注意，对于第一个地址字节， R/W = 0 表示主器件作为发送器，而从器件作为接收器。

19.2.2.3 从器件应答

在接收每个字节之后，接收器件必须产生应答信号 （ACK）。主器件必须产生与应答位关联的额外的 SCLx 时钟。

19.2.2.4 主器件发送

紧接的 2 个字节是由主器件发送到从器件的数据字节，包含所请求的 EEPROM 数据字节的地址。从器件必须对每个数据字节作出应答。

19.2.2.5 重复启动

在该时序点，从器件 EEPROM 已具有向主器件返回所请求数据字节所必需的地址信息。但是，第一个器件地址字节中的 R/W 状态位指定了主器件发送数据，而从器件接收数据。要让从器件向主器件发送数据，总线必须转换为另一方向。

要执行该功能而不结束报文传送，主器件需发送“重复启动”条件。重复启动条件后面跟随一个器件地址字节，其中包含与先前相同的器件地址，其中 R/W = 1，表示从器件发送数据，而主器件接收数据。

19.2.2.6 从器件答复

现在，从器件通过驱动 SDAx 线发送数据字节，而主器件继续产生时钟，但释放对 SDAx 的驱动。

19.2.2.7 主器件应答

在读数据期间，主器件必须通过对报文的最后一个字节作出 “不应答”（产生 “NACK”）来终止对从器件的数据请求。

19.2.2.8 停止报文

主器件发送停止条件来终止报文并将总线恢复为空闲状态。

19.3 控制和状态寄存器

I^2C 模块有 7 个用于操作的寄存器，可由用户应用程序访问。所有寄存器均可以字节或字模式进行访问。这些寄存器如下：

- 控制寄存器 （I2CxCON）：该寄存器 （寄存器 19-1）用于控制模块的操作。
- 状态寄存器 （I2CxSTAT）：该寄存器 （寄存器 19-2）包含状态标志，指示模块在操作期间的状态。
- 地址掩码寄存器 （I2CxMSK）：该寄存器 （寄存器 19-3）指定 I2CxADD 寄存器中的哪些位可以忽略，从而提供了多地址支持。
- 接收缓冲寄存器（I2CxRCV）：这是可从中读取数据字节的缓冲寄存器。I2CxRCV 寄存器是只读寄存器。
- 发送寄存器（I2CxTRN）：这是发送寄存器。在发送操作期间字节写入该寄存器。 I2CxTRN 寄存器是读 / 写寄存器。
- 地址寄存器 （I2CxADD）：该寄存器保存从器件地址。
- 波特率发生器重载寄存器（I2CxBRG）：保存 I^2C 模块波特率发生器的波特率发生器重载值。

I2CxTRN 是写入发送数据的寄存器。当模块作为主器件向从器件发送数据时，或作为从器件向主器件发送答复数据时，使用该寄存器。在报文的处理过程中， I2CxTRN 寄存器将移出各个数据位。因此，除非总线空闲，否则不能写入I2CxTRN寄存器。当发送当前数据时，可以重载I2CxTRN寄存器。

主器件或从器件正在接收的数据移入一个不可访问的移位寄存器 I2CxRSR。当接收到完整字节时，字节将传送到 I2CxRCV 寄存器。在接收操作中， I2CxRSR 和 I2CxRCV 寄存器共同构成一个双重缓冲接收器。这使得可以在读取所接收数据的当前字节之前开始接收下一字节。

如果在软件从 I2CxRCV 寄存器中读取前一字节之前，模块接收到另一完整字节，则发生接收器溢出， I2COV 位（I2CxSTAT<6>）置 1。 I2CxRSR 寄存器中的字节将丢失。除非模块收到总线上的启动 / 重复、启动 / 停止条件，将禁止进一步的接收与时钟延长。如果 I2COV 标志已清零，将继续正常接收。如果 I2COV 标志未清零，模块将正确接收下个字节，但会发送 NACK。随后，除非它检测到启动 / 重复、启动 / 停止条件，将不能接收更多字节或延长时钟。

I2CxADD 寄存器保存从器件地址。在 10 位寻址模式下，所有的位均有用。在 7 位寻址模式下，仅 I2CxADD<6:0> 位有用。请注意， I2CxADD<6:0> 位对应于地址字节中的高 7 位；读 / 写位没有包含在该寄存器内的值中。 A10M 位 （I2CxCON<10>）指定所期望的从器件地址模式。在两种从器件寻址模式下，通过配合使用 I2CxMSK 寄存器和 I2CxADD 寄存器，可从完全地址匹配中掩掉一个或多个位位置，从而使处于从模式的模块可以对多个地址作出响应。

寄存器 19-1： I2CxCON：I2Cx 控制寄存器

R/W-0	U-0	R/W-0	R/W-1, HC	R/W-0	R/W-0	R/W-0	R/W-0
I2CEN	—	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN
bit 15							bit 8

R/W-0	R/W-0	R/W-0	R/W-0, HC	R/W-0, HC	R/W-0, HC	R/W-0, HC	R/W-0, HC
GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN
bit 7							bit 0

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

bit 15 **I2CEN**：I2Cx 使能位
1 = 使能 I2Cx 模块，并将 SDAx 和 SCLx 引脚配置为串口引脚
0 = 禁止 I2Cx 模块；所有 I^2C 引脚由端口功能控制

bit 14 **未实现**：读为 0

bit 13 **I2CSIDL**：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 **SCLREL**：SCLx 释放控制位 （作为 I^2C 从器件工作时）
1 = 释放 SCLx 时钟
0 = 保持 SCLx 时钟为低电平 （时钟延长）
如果 STREN = 1：
该位可读可写（即软件可以写入 0 来启动时钟延长或写入 1 来释放时钟）。在从器件发送数据开始时和从器件接收数据结束时由硬件清零。
如果 STREN = 0：
该位可读且可被置 1 （即软件只能写入 1 来释放时钟）。在从器件发送开始时由硬件清零。

bit 11 **IPMIEN**：智能平台管理接口 （IPMI）使能位
1 = 使能 IPMI 支持模式；应答所有地址
0 = 禁止 IPMI 支持模式

bit 10 **A10M**：10 位从器件地址位
1 = I2CxADD 寄存器为 10 位从器件地址
0 = I2CxADD 寄存器为 7 位从器件地址

bit 9 **DISSLW**：禁止斜率控制位
1 = 禁止斜率控制
0 = 使能斜率控制

bit 8 **SMEN**：SMBus 输入电平位
1 = 使能符合 SMBus 规范的 I/O 引脚门限值
0 = 禁止 SMBus 输入门限值

bit 7 **GCEN**：广播呼叫使能位 （作为 I^2C 从器件工作时）
1 = 允许在 I2CxRSR 寄存器接收到广播呼叫地址时产生中断 （已使能模块接收）
0 = 禁止广播呼叫地址

bit 6 **STREN**：SCLx 时钟延长使能位 （仅适用于 I^2C 从模式；与 SCLREL 位配合使用）
1 = 使能软件或接收时钟延长
0 = 禁止软件或接收时钟延长

寄存器 19-1： I2CxCON：I2Cx 控制寄存器 （续）

bit 5 **ACKDT**：应答数据位 （I²C 主模式；仅适用于接收操作）
当软件启动应答序列时将发送的值
1 = 在应答时发送 NACK
0 = 在应答时发送 ACK

bit 4 **ACKEN**：应答序列使能位 （I²C 主模式接收操作）
1 = 在SDAx和SCLx引脚上发出应答序列，并发送ACKDT数据位（在主器件应答序列结束时由硬件清零）
0 = 应答序列不在进行中

bit 3 **RCEN**：接收使能位 （I²C 主模式）
1 = 使能 I²C 接收模式 （在主器件接收完数据字节的第 8 位时由硬件清零）
0 = 接收序列不在进行中

bit 2 **PEN**：停止条件使能位 （I²C 主模式）
1 = 在 SDAx 和 SCLx 引脚上发出停止条件 （在主器件停止序列结束时由硬件清零）
0 = 停止条件不在进行中

bit 1 **RSEN**：重复启动条件使能位 （I²C 主模式）
1 = 在 SDAx 和 SCLx 引脚上发出重复启动条件 （在主器件重复启动序列结束时由硬件清零）
0 = 重复启动条件不在进行中

bit 0 **SEN**：启动条件使能位 （I²C 主模式）
1 = 在 SDAx 和 SCLx 引脚上发出启动条件 （在主器件启动序列结束时由硬件清零）
0 = 启动条件不在进行中

寄存器 19-2： I2CxSTAT：I2Cx 状态寄存器

R-0, HSC	R-0, HSC	U-0	U-0	U-0	R/C-0, HS	R-0, HSC	R-0, HSC
ACKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10
bit 15							bit 8

R/C-0, HS	R/C-0, HS	R-0, HSC	R-0, HSC	R-0, HSC	R-0, HSC	R-0, HSC	R-0, HSC
IWCOL	I2COV	D/$\overline{A}$	P	S	R/$\overline{W}$	RBF	TBF
bit 7							bit 0

图注：		U = 未实现位，读为 0	
R = 可读位	C = 可清零位	HS = 硬件置 1 位	HSC = 硬件置 1/ 清零位
-n = 复位时的值	1 = 置 1	0 = 清零	x = 未知

bit 15 **ACKSTAT**：应答状态位
1 = 自从器件接收到 NACK
0 = 自从器件接收到 ACK
在从器件或主器件应答结束时由硬件置 1 或清零。

bit 14 **TRSTAT**：发送状态位 （I^2C 主模式发送操作）
1 = 主器件正在进行发送 （8 位 + $\overline{ACK}$）
0 = 主器件不在进行发送
在主器件发送开始时由硬件置 1 ；在从器件应答结束时由硬件清零。

bit 13-11 **未实现**：读为 0

bit 10 **BCL**：主器件总线冲突检测位
1 = 主器件工作期间检测到了总线冲突
0 = 未发生冲突
在检测到总线冲突时由硬件置 1。

bit 9 **GCSTAT**：广播呼叫状态位
1 = 接收到广播呼叫地址
0 = 未接收到广播呼叫地址
当地址与广播呼叫地址匹配时由硬件置 1 ；在检测到停止条件时由硬件清零。

bit 8 **ADD10**：10 位地址状态位
1 = 10 位地址匹配
0 = 10 位地址不匹配
在与匹配的 10 位地址的第二个字节匹配时由硬件置 1 ；在检测到停止条件时由硬件清零。

bit 7 **IWCOL**：写冲突检测位
1 = 因为 I^2C 模块忙，尝试写 I2CxTRN 寄存器失败
0 = 未发生冲突
当总线忙时写 I2CxTRN 寄存器会使硬件将该位置 1 （由软件清零）。

bit 6 **I2COV**：接收溢出标志位
1 = 当 I2CxRCV 寄存器仍然保存原先的字节时接收到了新字节
0 = 未溢出
尝试将数据从 I2CxRSR 寄存器传输到 I2CxRCV 寄存器时由硬件置 1 （由软件清零）。

bit 5 **D/$\overline{A}$**：数据 / $\overline{地址}$位 （I^2C 从模式）
1 = 表示上次接收的字节为数据
0 = 表示上次接收的字节为器件地址
在器件地址匹配时由硬件清零；接收到从器件字节时由硬件置 1，或在发送完成后由硬件置 1 且 TBF 标志清零。

bit 4 **P**：停止位
1 = 表示上次检测到停止位
0 = 上次未检测到停止位
当检测到启动、重复启动或停止条件时由硬件置 1 或清零。

寄存器 19-2： I2CxSTAT：I2Cx 状态寄存器 （续）

bit 3 **S**：启动位
1 = 表示上次检测到启动 （或重复启动）位
0 = 上次未检测到启动位
当检测到启动、重复启动或停止条件时由硬件置 1 或清零。

bit 2 **R/$\overline{W}$**：读 / $\overline{写}$信息位 （作为 I²C 从器件工作时）
1 = 读：数据传输自从器件输出
0 = 写：数据传输输入到从器件
接收到 I²C 器件地址字节后由硬件置 1 或清零。

bit 1 **RBF**：接收缓冲区满状态位
1 = 接收完成； I2CxRCV 寄存器为满
0 = 接收未完成； I2CxRCV 寄存器为空
用接收到的字节写 I2CxRCV 寄存器时由硬件置 1 ； 用软件读 I2CxRCV 寄存器时由硬件清零。

bit 0 **TBF**：发送缓冲区满状态位
1 = 发送正在进行中； I2CxTRN 寄存器为满
0 = 发送完成； I2CxTRN 寄存器为空
用软件写 I2CxTRN 寄存器时由硬件置 1 ； 数据发送完成时由硬件清零。

寄存器 19-3： I2CxMSK：I2Cx 从模式地址掩码寄存器

U-0	U-0	U-0	U-0	U-0	U-0	R/W-0	R/W-0
—	—	—	—	—	—	AMSK9	AMSK8
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
AMSK7	AMSK6	AMSK5	AMSK4	AMSK3	AMSK2	AMSK1	AMSK0
bit 7							bit 0

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

bit 15-10 **未实现：**读为 0

bit 9-0 **AMSKx：**地址中 bit x 的掩码选择位

对于 10 位地址：

1 = 使能输入报文的地址中 bit Ax 的掩码；在此位置上不需要位匹配

0 = 禁止 bit Ax 的掩码；在此位置上需要位匹配

对于 7 位地址 （仅 I2CxMSK<6:0>）：

1 = 使能输入报文的地址中 bit Ax + 1 的掩码；在此位置上不需要位匹配

0 = 禁止 bit Ax + 1 的掩码；在此位置上需要位匹配

19.4 使能 I²C 操作

通过将 I2CEN （I2CxCON<15>）位置 1 来使能 I²C 模块。

I²C 模块完全实现了所有主器件和从器件功能。当模块使能时，主器件和从器件功能同时工作，并根据软件或总线事件作出响应。

当初始使能时，模块会释放 SDAx 和 SCLx 引脚，将总线置为空闲状态。除非软件将 SEN 控制位置 1 并将数据装入 I2CxTRN 寄存器，否则主器件功能将保持在空闲状态。这两个操作会启动主器件事件。

当主器件逻辑处于工作状态时，从器件逻辑也处于工作状态。因此，从器件功能将开始监视总线。如果从器件逻辑在总线上检测到启动事件和有效的地址，从器件逻辑将开始从器件事务。

19.4.1 使能 I²C I/O

总线操作使用两个引脚。它们是 SCLx 引脚 （时钟线）和 SDAx 引脚 （数据线）。当模块使能时，假设没有其他更高优先级的模块在控制总线，则模块将获得对 SDAx 和 SCLx 引脚的控制。模块软件不需要关心引脚的端口 I/O 的状态，因为模块会改写端口状态和方向。在初始化时，引脚处于三态 （被释放）。

19.4.2 I²C 中断

I²C 模块可产生两种中断。第一种中断 MI2CxIF 被分配给主器件事件，另一种中断 SI2CxIF 被分配给从器件事件。这两种中断会将相应的中断标志位置 1，并在相应的中断允许位置 1 且相应的中断优先级足够高时中断软件执行过程。

MI2CxIF 中断在以下主器件报文事件完成时产生：

- 启动条件
- 停止条件
- 数据传输字节发送 / 接收
- 应答发送
- 重复启动
- 检测到总线冲突事件

SI2CxIF 中断在检测到地址为从器件地址的报文时产生，包括以下事件：

- 检测到有效的器件地址 （包括广播呼叫）
- 请求数据发送 （$\overline{\text{ACK}}$）或停止数据发送 （NACK）
- 接收到数据

19.4.3 作为总线主器件工作时设置波特率

当作为 I²C 主器件工作时，模块必须产生系统 SCLx 时钟。通常，I²C 系统时钟指定为 100 kHz、400 kHz 或 1 MHz。系统时钟速率指定为最小 SCLx 低电平时间加最小 SCLx 高电平时间。在大多数情况下，这由 2 个 TBRG 时间间隔确定。

波特率发生器的重载值为 I2CxBRG 寄存器值，如图 19-5 所示。当波特率发生器装入该值时，发生器递减计数至 0 并停止，直到发生另一次重载。发生器计数在每个指令周期（TCY）递减两次。在波特率重新启动时，将自动重载波特率发生器。例如，如果发生时钟同步，波特率发生器将在 SCLx 引脚采样为高电平时重载。

注：	I2CxBRG 寄存器的值不能小于 2。

要计算波特率发生器的重载值，可使用以下公式：

公式 19-1： BRG 重载值计算

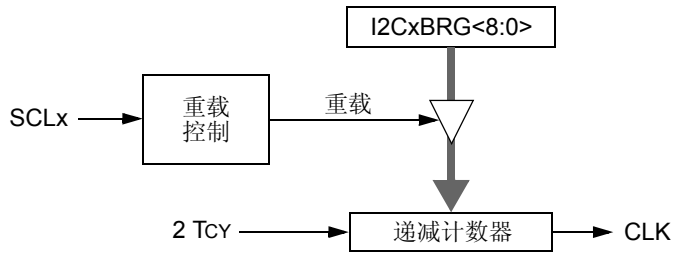
$$I2CBRG = \left(\frac{FCY}{FSCL} - \frac{FCY}{10{,}000{,}000} \right) - 1$$

表 19-1： I²C™ 时钟速率

必需的系统 FSCL	FCY	I2CBRG 十进制	I2CBRG 十六进制	实际 FSCL
100 kHz	40 MHz	395	0x18B	100 kHz
100 kHz	20 MHz	197	0x0C5	100 kHz
100 kHz	10 MHz	98	0x0b2	100 kHz
400 kHz	20 MHz	47	0x02F	400 kHz
400 kHz	10 MHz	23	0x017	400 kHz
400 kHz	5 MHz	11	0x00B	400 kHz
1 MHz	10 MHz	8	0x008	1 MHz

注：	公式 19-1 和表 19-1 仅用作指导。由于依赖于系统的一些参数，实际的波特率可能略有不同。需要进行测试来确定实际波特率符合系统要求。否则，可能需要调整 I2CxBRG 的值。

图 19-5： 波特率发生器框图



19.5 在单主机环境中作为主器件进行通信

I²C 模块在系统中的典型操作是使用 I²C 来与 I²C 外设（例如 I²C 串行存储器）进行通信。在 I²C 系统中，主器件控制总线上所有数据通信的序列。在该示例中， PIC24H 及其 I²C 模块在系统中作为唯一的主器件。作为唯一的主器件，它负责产生 SCLx 时钟和控制报文协议。

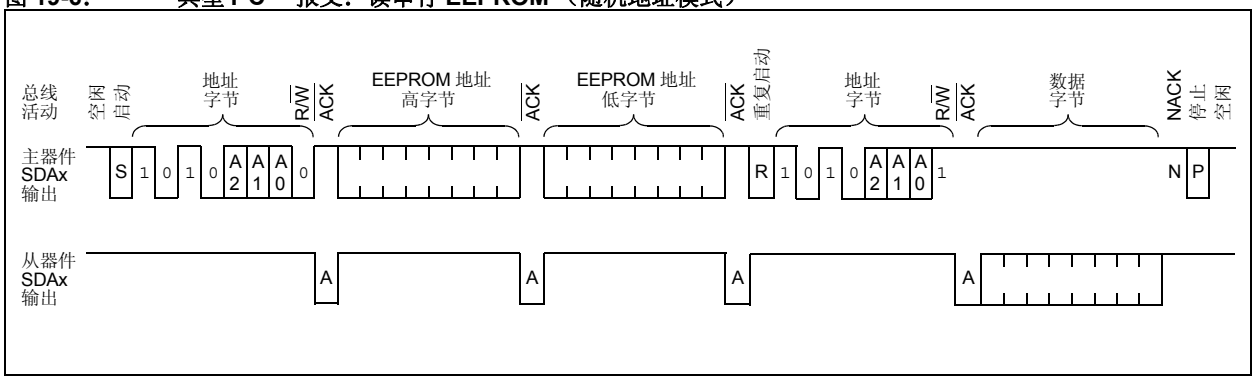
在 I²C 模块中，模块控制 I²C 报文协议的各个部分；但是，产生协议各组成部分的序列以构成完整的报文则需由软件完成。

例如，在单主机环境中，典型的操作可能是从 I²C 串行 EEPROM 读取字节。图 19-6 给出了该报文示例。

要完成该报文，软件通过以下步骤产生序列。

1. 在 SDAx 和 SCLx 上发出一个启动条件。
2. 向从器件发送带有写操作指示的 I²C 器件地址字节。
3. 等待并验证来自从器件的应答。
4. 向从器件发送串行存储器地址高字节。
5. 等待并验证来自从器件的应答。
6. 向从器件发送串行存储器地址低字节。
7. 等待并验证来自从器件的应答。
8. 在 SDAx 和 SCLx 上发出一个重复启动条件。
9. 向从器件发送带有读操作指示的器件地址字节。
10. 等待并验证来自从器件的应答。
11. 使能主器件接收，以接收串行存储器数据。
12. 在数据字节接收结束时产生 $\overline{\text{ACK}}$ 或 NACK 条件。
13. 在 SDAx 和 SCLx 上产生一个停止条件。

图 19-6： 典型 I²C™ 报文：读串行 EEPROM （随机地址模式）



I²C 模块有启动和停止条件发生器、数据字节发送功能、数据字节接收功能、应答发生器和波特率发生器，用以支持主模式通信。通常，软件会写入某个控制寄存器来启动特定的步骤，然后等待中断或查询状态来等待完成。

后续章节对这些操作进行了详细说明。

注：	I²C 模块不允许事件排队。例如，在启动条件完成前，不允许软件产生启动条件并立即写 I2CxTRN 寄存器以启动传输。在这种情况下，I2CxTRN 寄存器将不会被写入，且 IWCOL 状态位将被置 1，表示没有发生对 I2CxTRN 寄存器的写操作。

19.5.1 产生启动总线事件

要产生启动事件，软件应将启动使能位 SEN（I2CxCON<0>）置 1。在置 1 启动位之前，软件可以检查 P 状态位 （I2CxSTAT<4>）来确保总线处于空闲状态。

图 19-7 给出了启动条件的时序。

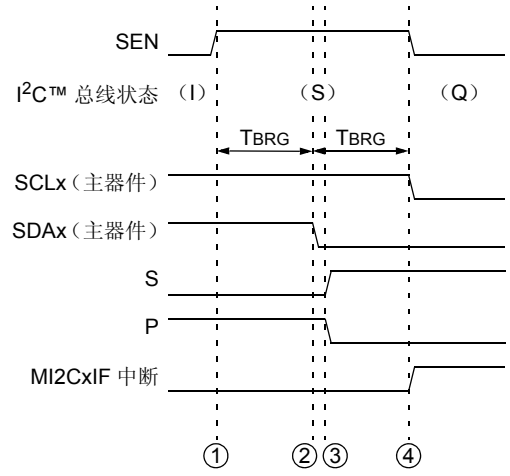
- 从器件逻辑检测到启动条件，将S状态位（I2CxSTAT<3>）置1，并将P状态位（I2CxSTAT<4>）清零。
- 在启动条件完成时， SEN 位自动清零。
- 在启动条件完成时，产生 MI2CxIF 中断。
- 在启动条件之后， SDAx 线和 SCLx 线保持为低电平 （Q 状态）。

19.5.1.1 IWCOL 状态标志

如果软件在启动序列进行过程中写 I2CxTRN 寄存器，则 IWCOL 状态位将置 1，同时发送缓冲区内容不变 （写操作无效）。

注：	由于不允许事件排队，在启动条件结束之前，不能写 I2CxCON 寄存器的低 5 位。

图 19-7： 主器件启动时序图



① 写 SEN = 1 将产生主器件启动事件。波特率发生器启动。
② 波特率发生器超时。主模块将 SDAx 驱动为低电平。波特率发生器重新启动。
③ 从模块检测到启动条件，并设置 S = 1， P = 0。
④ 波特率发生器超时。主模块将 SCLx 驱动为低电平，产生中断并清零 SEN。

19.5.2 向从器件发送数据

将适当的值写入 I2CxTRN 寄存器即可发送一个数据字节、一个 7 位器件地址字节或一个 10 位地址的第二个字节。在该寄存器中装入值将启动以下过程：

- 软件在 I2CxTRN 寄存器中装入要发送的数据字节。
- 写 I2CxTRN 寄存器会将缓冲区满标志位 TBF （I2CxSTAT<0>）置 1。
- 数据字节从SDAx引脚移出，直到发送完所有8个位。地址/数据的每一位将在SCLx的下降沿之后移出 SDAx 引脚。
- 在第 9 个 SCLx 时钟，模块将来自从器件的 $\overline{\text{ACK}}$ 位移入，并将其值写入 ACKSTAT 状态位（I2CxSTAT<15>）。
- 在第 9 个 SCLx 时钟周期结束时，模块产生 MI2CxIF 中断。

模块并不产生或验证数据字节。字节的内容和使用取决于由软件维护的报文协议的状态。

19.5.2.1 向从器件发送 7 位地址

发送 7 位器件地址需要向从器件发送一个字节。7 位地址字节必须包含 I²C 器件地址的 7 位和一个 R/W 状态位，该位定义报文是向从器件写数据 （主器件发送，从器件接收）还是自从器件读数据 （从器件发送，主器件接收）。

注： 在 7 位寻址模式下，使用 I²C 协议的每个节点都应配置为存储在 I2CxADD 寄存器中的唯一地址。

19.5.2.2 向从器件发送 10 位地址

发送 10 位器件地址需要向从器件发送 2 个字节。第一个字节中包含专为 10 位寻址模式而预留的 I²C 器件地址的 5 位和 10 位地址的 2 位。因为下一字节（含有 10 位地址的剩余 8 位）必须由从器件接收，所以第一个字节中的 R/W 状态位必须为 0，表示主器件发送，从器件接收。如果报文数据也是发送给从器件，则主器件可以继续发送数据。但是，如果主器件希望得到从器件的答复，则需要产生重复启动序列，且 R/W 状态位为 1，这可以将报文的 R/W 状态更改为读从器件。

注： 在 10 位寻址模式下，使用 I²C 协议的每个节点都应配置为存储在 I2CxADD 寄存器中的唯一地址。

19.5.2.3 接收来自从器件的应答

在第 8 个 SCLx 时钟的下降沿，TBF 状态位被清零，主器件会释放 SDAx 引脚，以允许从器件发出一个应答响应。然后，主器件会产生第 9 个 SCLx 时钟。

如果发生地址匹配或数据被正确接收，这就可使被寻址的从器件在第 9 个位时间发出一个 $\overline{\text{ACK}}$ 位响应。从器件在识别出其器件地址 （包括广播呼叫地址）或正确接收数据后，会发出一个应答。

$\overline{\text{ACK}}$ 的状态在第 9 个 SCLx 时钟的下降沿写入应答状态位 ACKSTAT （I2CxSTAT<15>）。在第 9 个 SCLx 时钟之后，模块会产生 MI2CxIF 中断并进入空闲状态，直到下一数据字节装入 I2CxTRN 寄存器。

19.5.2.4 ACKSTAT 状态标志

当从器件发送应答 （$\overline{\text{ACK}}$ = 0）时， ACKSTAT 状态位 （I2CxSTAT<15>）清零；当从器件不应答 （$\overline{\text{ACK}}$ = 1）时，该位置 1。

19.5.2.5 TBF 状态标志

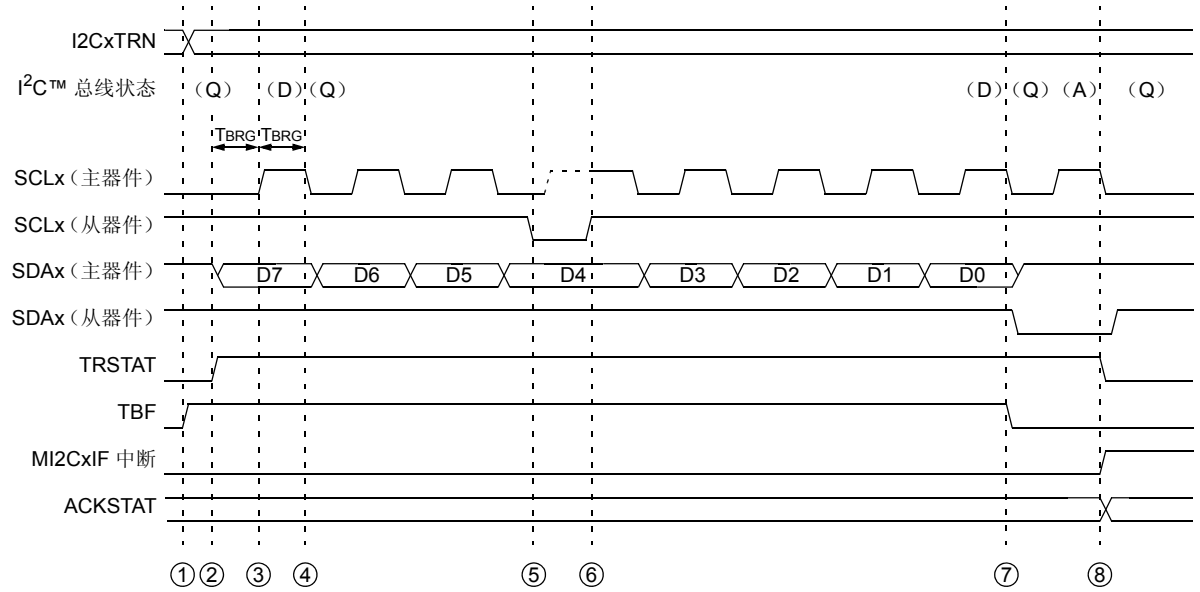
发送时， TBF 状态位 （I2CxSTAT<0>）在 CPU 写 I2CXTRN 寄存器时置 1，在所有 8 个位移出后清零。

19.5.2.6 IWCOL 状态标志

如果软件在发送进行过程中（即，模块仍在移出数据字节）试图写 I2CXTRN 寄存器，则 IWCOL 状态位将置 1，同时缓冲区内容不变 （写操作无效）。 IWCOL 状态位必须用软件清零。

注： 由于不允许事件排队，在发送条件结束之前，不能写 I2CxCON 寄存器的低 5 位。

图 19-8：　主器件发送时序图



① 写 I2CxTRN 寄存器将产生主器件发送事件。TBF 状态位置 1。

② 波特率发生器启动。I2CxTRN 寄存器的最高位驱动 SDAx。SCLx 保持为低电平。TRSTAT 状态位置 1。

③ 波特率发生器超时。SCLx 被释放。波特率发生器重新启动。

④ 波特率发生器超时。SCLx 被驱动为低电平。检测到 SCLx 为低电平后，I2CxTRN 寄存器的下一位驱动 SDAx。

⑤ 当 SCLx 为低电平时，从器件也可以将 SCLx 拉低以启动等待（时钟延长）。

⑥ 主器件已释放 SCLx，从器件可以释放 SCLx 以结束等待。波特率发生器重新启动。

⑦ 在第 8 个 SCLx 时钟的下降沿，主器件释放 SDAx。TBF 状态位清零。从器件驱动 $\overline{\text{ACK}}$/NACK。

⑧ 在第 9 个 SCLx 时钟的下降沿，主器件产生中断。SCLx 保持为低电平，直到发生下一个事件。
从器件释放 SDAx。TRSTAT 状态位清零。

19.5.3　接收来自从器件的数据

主器件在发送了 R/$\overline{\text{W}}$ 状态位的值为 1 的从器件地址之后，即可接收来自从器件的数据。这可通过将接收使能位 RCEN （I2CxCON<3>）置 1 来使能。主器件逻辑开始产生时钟，并且在 SCLx 的每个下降沿之前，对 SDAx 线进行采样并将数据移入 I2CxRSR 寄存器。

注：	在尝试将 RCEN 位置 1 之前，I2CxCON 寄存器的低 5 位必须为 0。这将确保主器件逻辑处于不工作状态。

在第 8 个 SCLx 时钟的下降沿之后，会发生以下事件：

- RCEN 位被自动清零。
- I2CxRSR 寄存器的内容被传送到 I2CxRCV 寄存器。
- RBF 状态位置 1。
- 模块产生 MI2CxIF 中断。

当 CPU 读缓冲区时， RBF 状态位被自动清零。软件可以处理数据，然后执行应答序列。

19.5.3.1 RBF 状态标志

接收数据时，当将器件地址或数据字节从 I2CxRSR 寄存器装入 I2CxRCV 寄存器时， RBF 状态位置 1。当软件读 I2CxRCV 寄存器时，该位被清零。

19.5.3.2 I2COV 状态标志

如果在 RBF 状态位保持置 1 且前一字节仍保留在 I2CxRCV 寄存器中时，在 I2CxRSR 寄存器中接收到另一字节，则 I2COV 状态位置 1， I2CxRSR 寄存器中的数据丢失。

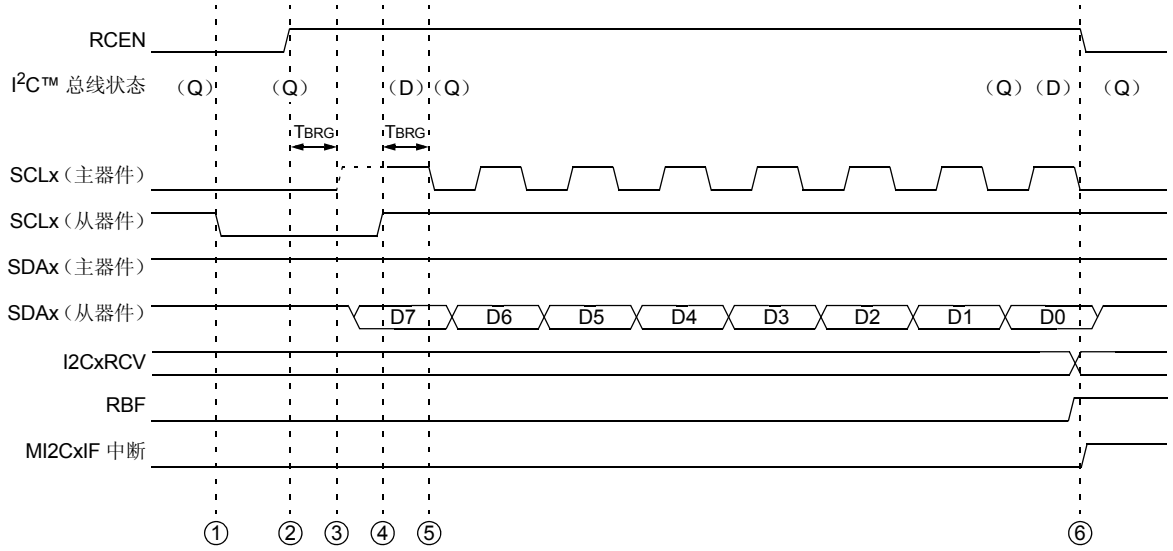
将 I2COV 状态位保持为置 1 并不会禁止继续接收数据。如果通过读 I2CxRCV 寄存器将 RBF 状态位清零，并且 I2CxRSR 寄存器接收到另一字节，则该字节将传送到 I2CxRCV 寄存器。

19.5.3.3 IWCOL 状态标志

如果软件在接收进行过程中（即， I2CxRSR 寄存器仍在移入数据字节）写 I2CxTRN 寄存器，则 IWCOL 状态位将置 1，同时缓冲区内容不变 （写操作无效）。

注：	由于不允许事件排队，在数据接收条件结束之前，不能写I2CxCON寄存器的低5位。

图 19-9： 主器件接收时序图



① 通常，从器件可以通过将 SCLx 拉低（时钟延长）来请求等待，以准备数据响应。
在准备就绪时，从器件将在 SDAx 上发送数据响应的最高位。

② 写 RCEN 位将产生主器件接收事件。波特率发生器启动。SCLx 保持为低电平。

③ 波特率发生器超时。主器件尝试释放 SCLx。

④ 当从器件释放 SCLx 时，波特率发生器重新启动。

⑤ 波特率发生器超时。响应的最高位移入 I2CxRSR 寄存器。SCLx 被驱动为低电平，等待下一个波特率时间间隔。

⑥ 在第 8 个 SCLx 时钟的下降沿，I2CxRSR 寄存器中的数据传送到 I2CxRCV 寄存器。模块将 RCEN 位清零。
RBF 状态位置 1。主器件产生中断。

19.5.4 应答产生

将应答使能位 ACKEN （I2CxCON<4>）置 1，将允许产生主器件应答序列。

注：	在尝试将 ACKEN 位置 1 之前，I2CxCON 寄存器的低 5 位必须为 0（主器件逻辑不工作）。

图 19-10 所示为 $\overline{\text{ACK}}$ 序列，图 19-11 所示为 NACK 序列。应答数据位 ACKDT （I2CxCON<5>）指定发送 $\overline{\text{ACK}}$ 还是 NACK。

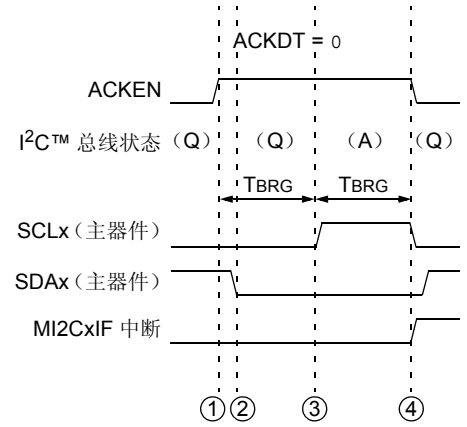
在两个波特率周期之后， ACKEN 位自动清零，模块产生 MI2CxIF 中断。

19.5.4.1 IWCOL 状态标志

如果软件在应答序列进行过程中写 I2CxTRN 寄存器，则 IWCOL 状态位将置 1，同时缓冲区内容不变 （写操作无效）。

注：	由于不允许事件排队，在应答条件结束之前，不能写 I2CxCON 寄存器的低 5 位。

图 19-10： 主器件应答 （$\overline{\text{ACK}}$）时序图



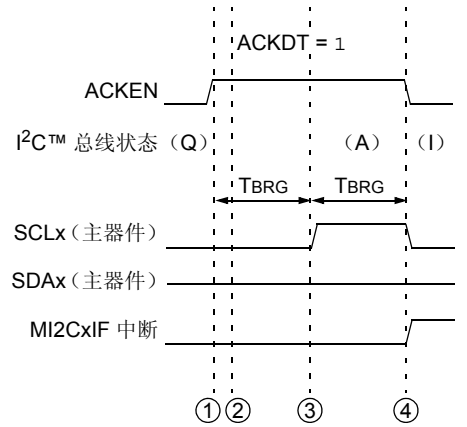
① 写 ACKDT = 0 指示发送 $\overline{\text{ACK}}$。
写 ACKEN = 1 指示产生了主器件应答事件。
波特率发生器启动。SCLx 保持为低电平。

② 当检测到 SCLx 为低电平时，模块驱动 SDAx 为低电平。

③ 波特率发生器超时。模块释放 SCLx。
波特率发生器重新启动。

④ 波特率发生器超时。
模块驱动 SCLx 为低电平，然后释放 SDAx。
模块将 ACKEN 清零。主器件产生中断。

图 19-11： 主器件不应答 （NACK）时序图



① 写 ACKDT = 1 指示发送 NACK。
写 ACKEN = 1 指示产生了主器件应答事件。
波特率发生器启动。

② 当检测到 SCLx 为低电平时，模块释放 SDAx。

③ 波特率发生器超时。模块释放 SCLx。
波特率发生器重新启动。

④ 波特率发生器超时。
模块驱动 SCLx 为低电平，然后释放 SDAx。
模块将 ACKEN 清零。主器件产生中断。

19.5.5 产生停止总线事件

将停止使能位 PEN （I2CxCON<2>）置 1，将允许产生主器件停止序列。

注： 在尝试将PEN位置1之前，I2CxCON寄存器的低5位必须为0（主器件逻辑不工作）。

当 PEN 位置 1 时，主器件会产生停止序列，如图 19-12 所示。

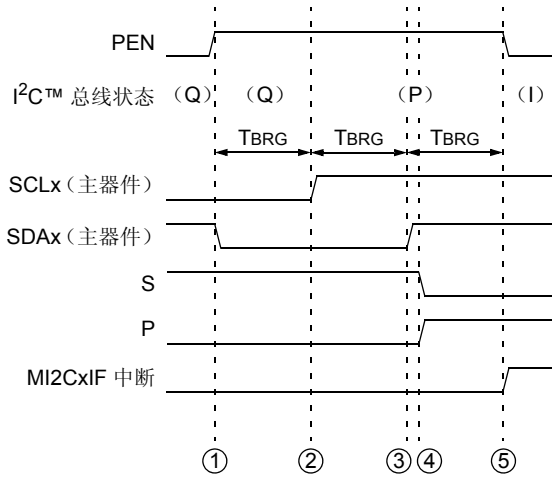
- 从器件检测到停止条件，将 P 状态位（I2CxSTAT<4>）置 1，并将 S 状态位（I2CxSTAT<3>）清零。
- PEN 位被自动清零。
- 模块产生 MI2CxIF 中断。

19.5.5.1 IWCOL 状态标志

如果软件在停止序列进行过程中写 I2CxTRN 寄存器，则 IWCOL 状态位将置 1，同时缓冲区内容不变 （写操作无效）。

注： 由于不允许事件排队，在停止条件结束之前，不能写 I2CxCON 寄存器的低 5 位。

图 19-12： 主器件停止时序图



① 写 PEN = 1 将产生主器件停止事件。
波特率发生器启动。模块驱动 SDAx 为低电平。

② 波特率发生器超时。模块释放 SCLx。
波特率发生器重新启动。

③ 波特率发生器超时。模块释放 SDAx。
波特率发生器重新启动。

④ 从器件逻辑检测到停止条件。模块设置 P = 1，S = 0。

⑤ 波特率发生器超时。模块将 PEN 清零。
主器件产生中断。

19.5.6 产生重复启动总线事件

将重复启动使能位 RSEN（I2CxCON<1>）置 1，将允许产生主器件重复启动序列（见图 19-13）。

注：	在尝试将 RSEN 位置 1 之前， I2CxCON 寄存器的低 5 位必须为 0 （主器件逻辑不工作）。

要产生重复启动条件，软件需将 RSEN 位 （I2CxCON<1>）置 1。模块将 SCLx 引脚拉为低电平。当模块采样到 SCLx 引脚为低电平时，模块释放 SDAx 引脚一个波特率发生器计数周期（TBRG）。当波特率发生器超时，并在模块采样到 SDAx 为高电平时，模块会释放 SCLx 引脚。当模块采样到 SCLx 引脚为高电平时，波特率发生器会重载并开始计数。 SDAx 和 SCLx 必须采样为一个计数周期 TBRG 的高电平。接下来的一个 TBRG，将 SDAx 引脚驱动为低电平，同时 SCLx 保持高电平。

以下是重复启动序列：

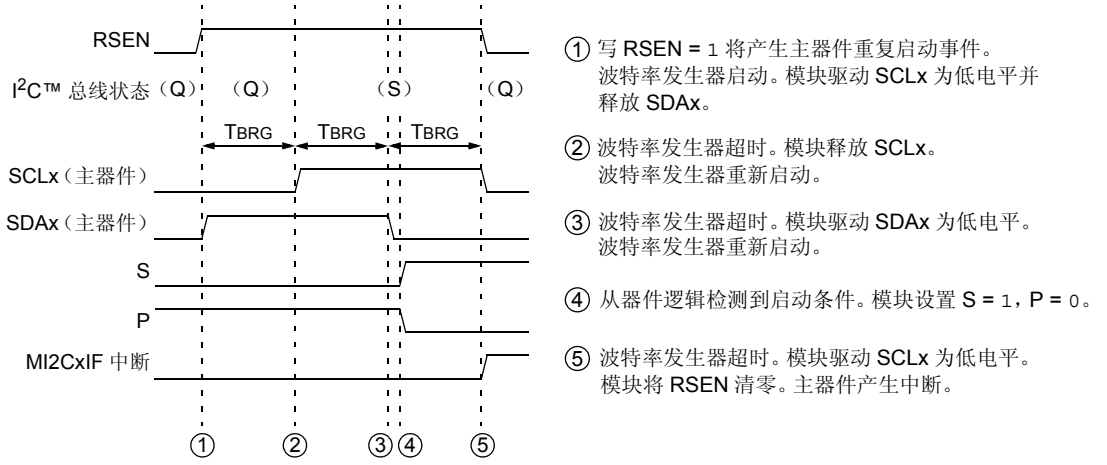
- 从器件检测到启动条件，将 S 状态位（I2CxSTAT<3>）置 1，并将 P 状态位（I2CxSTAT<4>）清零。
- RSEN 位被自动清零。
- 模块产生 MI2CxIF 中断。

19.5.6.1 IWCOL 状态标志

如果软件在重复启动序列进行过程中写 I2CxTRN 寄存器，则 IWCOL 状态位将置 1，同时缓冲区内容不变 （写操作无效）。

注：	由于不允许事件排队，在重复启动条件结束之前，不能写I2CxCON寄存器的低5位。

图 19-13： 主器件重复启动时序图



19.5.7 构造完整的主器件报文

如**第 19.5 节 “在单主机环境中作为主器件进行通信”**开头所述，由软件负责使用正确的报文协议构造报文。模块控制 I²C 报文协议的各个部分；但是，产生协议各组成部分的序列以构成完整的报文则需由软件完成。

在使用模块时，软件可以使用查询或中断方法。所显示的示例使用中断方法。

在报文传输过程时，软件可以使用 SEN、RSEN、PEN、RCEN 和 ACKEN 位（I2CxCON 寄存器的低 5 位）和 TRSTAT 状态位作为 “状态”标志。例如，表 19-2 给出了一些与总线状态相关的状态号的示例。

表 19-2： 主器件报文协议状态

示例 状态号	I2CxCON<4:0>	TRSTAT （I2CxSTAT<14>）	状态
0	00000	0	总线空闲或等待
1	00001	N/A	发送启动事件
2	00000	1	主器件发送
3	00010	N/A	发送重复启动事件
4	00100	N/A	发送停止事件
5	01000	N/A	主器件接收
6	10000	N/A	主器件应答

注： 状态号的示例仅供参考。用户软件可以根据需要分配状态号。

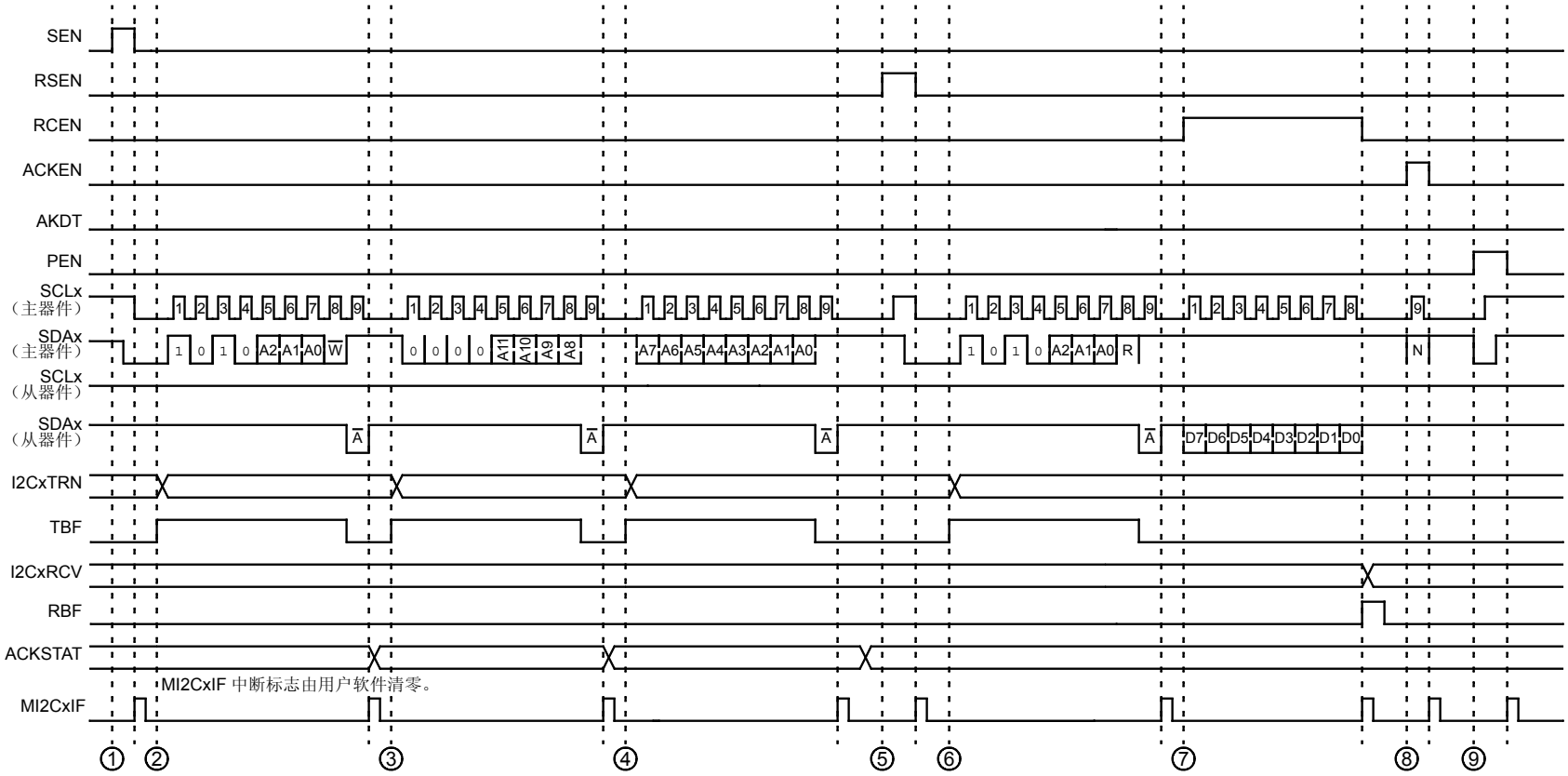
软件通过发出启动命令来开始发送报文。软件将记录对应于启动命令的状态号。

当每个事件结束并产生中断时，中断处理程序可以检查状态号。因此，对于启动状态，中断处理程序将确认启动序列的执行，然后启动主器件发送事件来发送 I²C 器件地址，并更改状态号以对应于主器件发送。

在下次中断时，中断处理程序将再次检查状态，确定主器件发送刚刚完成。中断处理程序将确认数据发送已成功，然后根据报文的内容继续执行下一事件。在这种方式中，每次中断时，中断处理程序将按报文协议进行处理，直到发送了完整的报文。

图 19-14 与图 19-6 的报文序列相同但提供了更详细的说明。图 19-15 所示为使用 7 位寻址格式的报文的一些简单示例。图 19-16 所示为向从器件发送数据的 10 位寻址格式报文的示例。图 19-17 所示为接收来自从器件数据的 10 位寻址格式报文的示例。

图 19-14：　主器件报文（典型 I²C™ 报文：读串行 EEPROM）



① 将 SEN 位置 1 产生启动事件。

② 写 I2CxTRN 寄存器启动主器件发送。数据为串行 EEPROM 器件地址字节，R/$\overline{W}$ 状态位清零，表示进行写操作。

③ 写 I2CxTRN 寄存器启动主器件发送。数据为 EEPROM 数据地址的第一个字节。

④ 写 I2CxTRN 寄存器启动主器件发送。数据为 EEPROM 数据地址的第二个字节。

⑤ 将 RSEN 位置 1 产生重复启动事件。

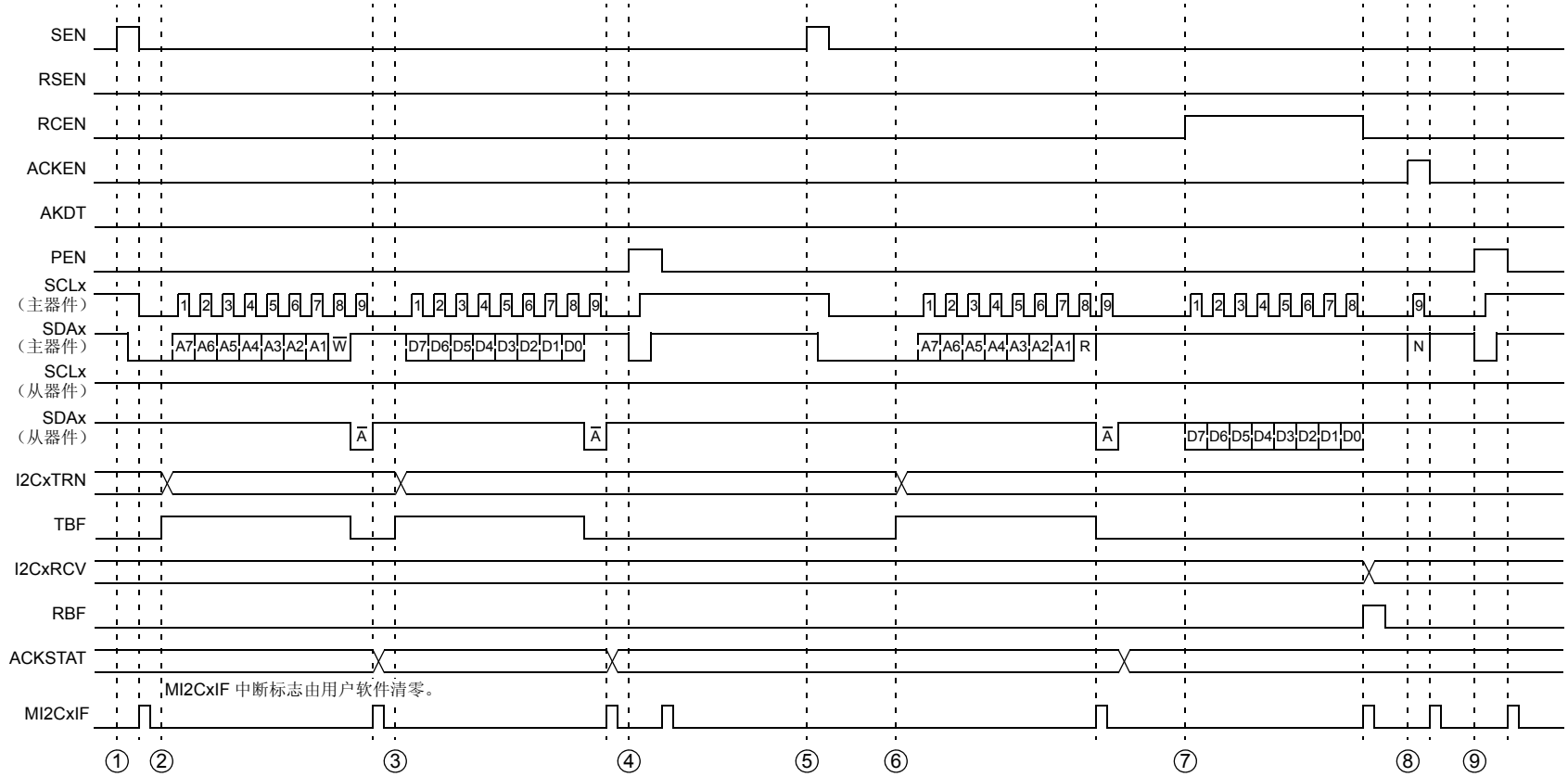
⑥ 写 I2CxTRN 寄存器启动主器件发送。重新发送串行 EEPROM 器件地址字节，但 R/$\overline{W}$ 状态位置 1，表示进行读操作。

⑦ 将 RCEN 位置 1 启动主器件接收。发生中断时，软件读 I2CxRCV 寄存器，这会清零 RBF 状态位。

⑧ 将 ACKEN 位置 1 产生应答事件。ACKDT = 1 发送 NACK。

⑨ 将 PEN 位置 1 产生主器件停止事件。

图 19-15：　主器件报文（7 位地址：发送和接收）



① 将 SEN 位置 1 产生启动事件。

② 写 I2CxTRN 寄存器启动主器件发送。数据为地址字节，且 R/$\overline{W}$ 状态位清零。

③ 写 I2CxTRN 寄存器启动主器件发送。数据为报文字节。

④ 将 PEN 位置 1 产生主器件停止事件。

⑤ 将 SEN 位置 1 产生启动事件。

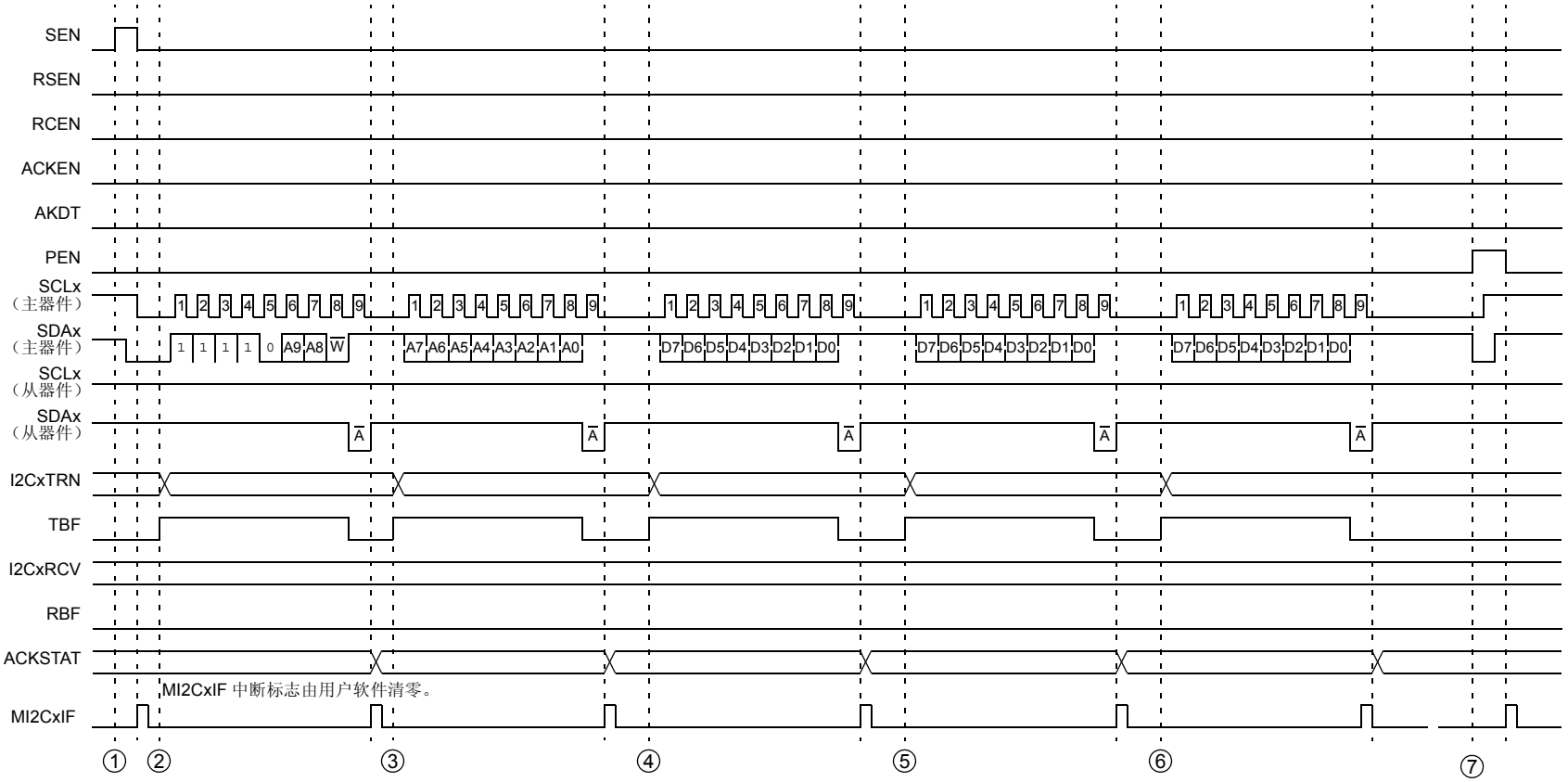
⑥ 写 I2CxTRN 寄存器启动主器件发送。数据为地址字节，且 R/$\overline{W}$ 状态位置 1。

⑦ 将 RCEN 位置 1 启动主器件接收。

⑧ 将 ACKEN 位置 1 产生应答事件。ACKDT = 1 发送 NACK。

⑨ 将 PEN 位置 1 产生主器件停止事件。

图 19-16： 主器件报文 （10 位发送）



① 将 SEN 位置 1 产生启动事件。

② 写 I2CxTRN 寄存器启动主器件发送。数据为地址的第一个字节。

③ 写 I2CxTRN 寄存器启动主器件发送。数据为地址的第二个字节。

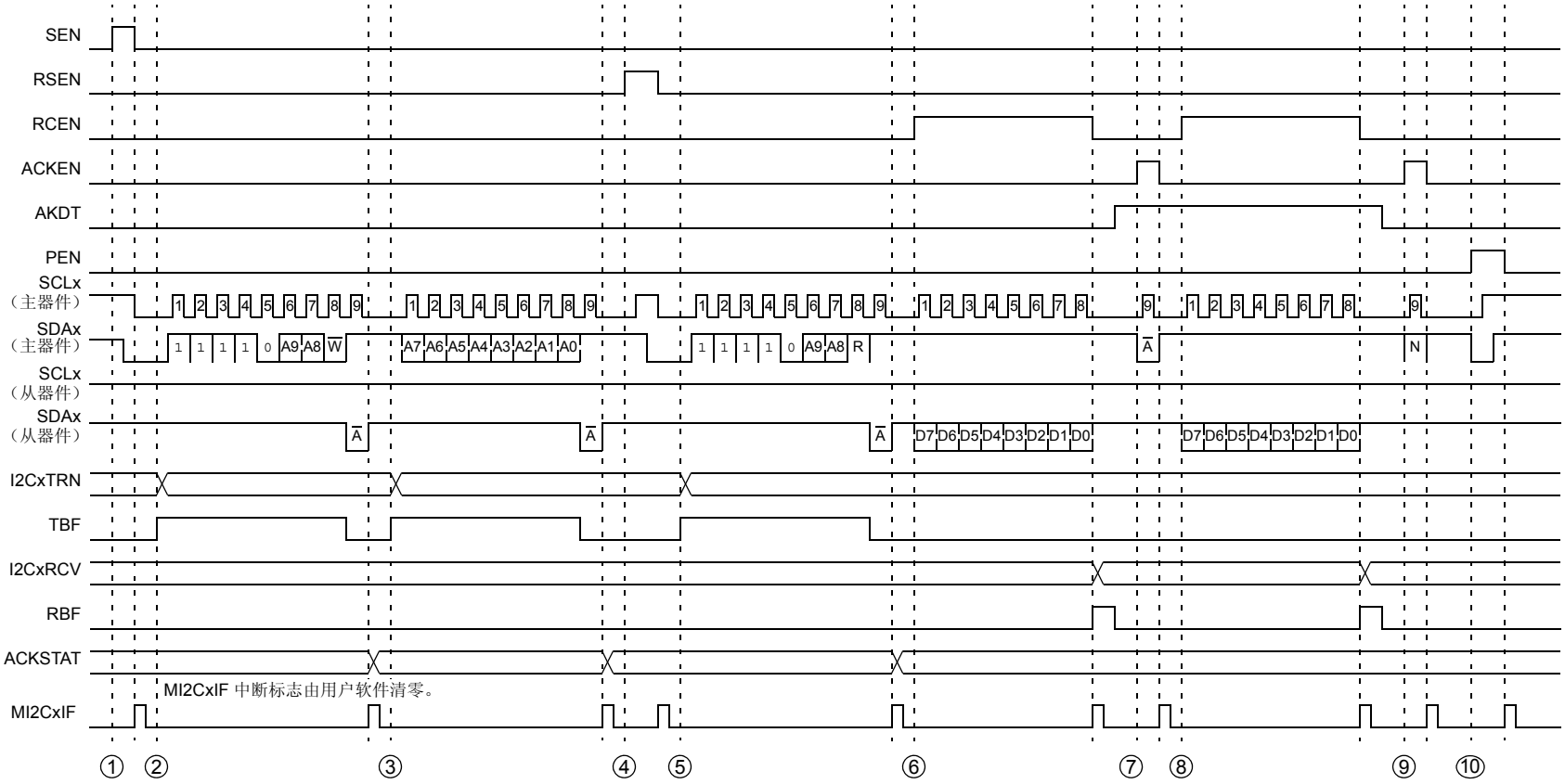
④ 写 I2CxTRN 寄存器启动主器件发送。数据为报文数据的第一个字节。

⑤ 写 I2CxTRN 寄存器启动主器件发送。数据为报文数据的第二个字节。

⑥ 写 I2CxTRN 寄存器启动主器件发送。数据为报文数据的第三个字节。

⑦ 将 PEN 位置 1 产生主器件停止事件。

图 19-17： 主器件报文 （10 位接收）



① 将 SEN 位置 1 产生启动事件。

② 写 I2CxTRN 寄存器启动主器件发送。数据为地址的第一个字节，且 R/$\overline{W}$ 状态位清零。

③ 写 I2CxTRN 寄存器启动主器件发送。数据为地址的第二个字节。

④ 将 RSEN 位置 1 产生主器件重复启动事件。

⑤ 写 I2CxTRN 寄存器启动主器件发送。重新发送第一个字节，且 R/$\overline{W}$ 状态位置 1。

⑥ 将 RCEN 位置 1 启动主器件接收。发生中断时，软件读 I2CxRCV 寄存器，这会清零 RBF 状态位。

⑦ 将 ACKEN 位置 1 产生应答事件。ACKDT = 0 发送 $\overline{ACK}$。

⑧ 将 RCEN 位置 1 启动主器件接收。

⑨ 将 ACKEN 位置 1 产生应答事件。ACKDT = 1 发送 NACK。

⑩ 将 PEN 位置 1 产生主器件停止事件。

19.6 在多主机环境中作为主器件进行通信

I^2C协议允许在系统总线上连接多个主器件。要考虑到主器件可以启动报文事务和产生总线时钟，而协议有应对多个主器件试图控制总线的方法。时钟同步可确保多个节点将其 SCLx 时钟同步，并在 SCLx 线上产生公共时钟。如果有多个节点试图启动报文事务，总线仲裁可以确保有且仅有一个节点能成功完成报文事务。其他节点将在总线仲裁中失败，产生总线冲突。

19.6.1 多主机操作

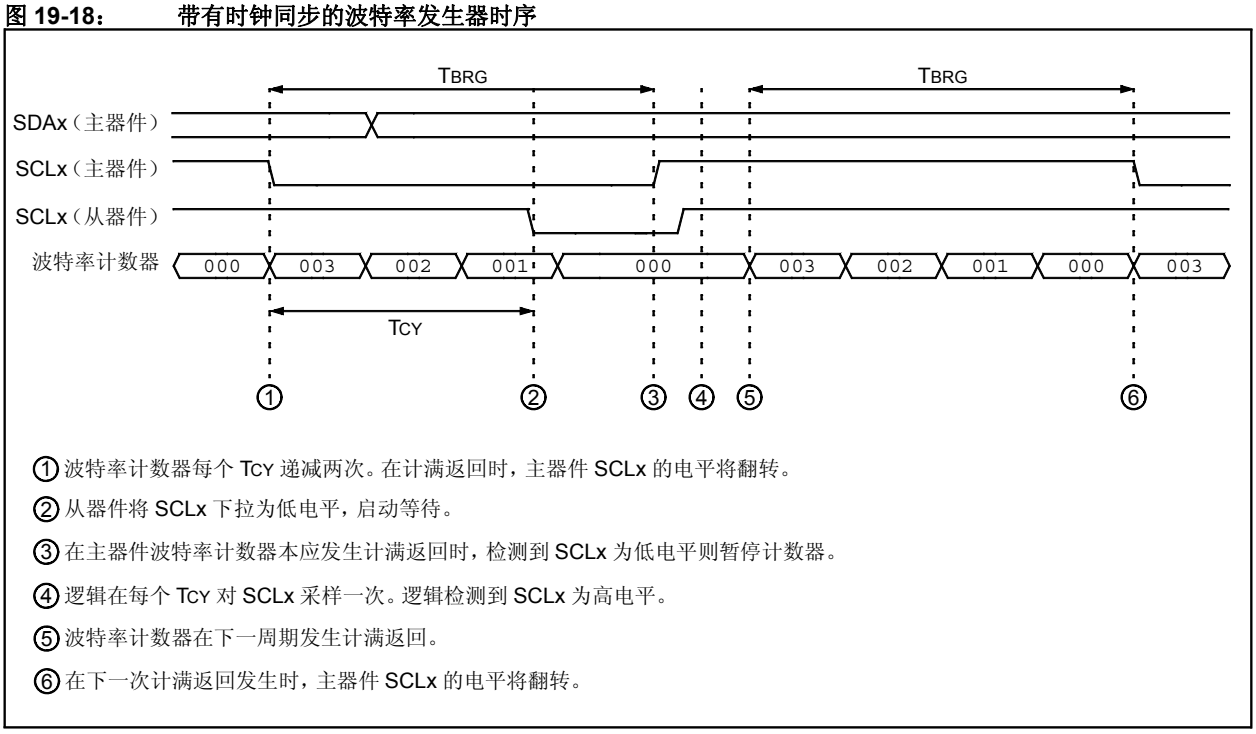
主模块没有使能多主机操作的特殊设置。模块一直执行时钟同步和总线仲裁。如果是在单主机环境中使用模块，则只会在主器件和从器件之间发生时钟同步，而不会发生总线仲裁。

19.6.2 主器件时钟同步

在多主机系统中，不同的主器件可能具有不同的波特率。时钟同步可确保这些主器件在尝试通过仲裁控制总线时，对它们的时钟将进行协调。

主器件释放 SCLx 引脚（SCLx 趋向于悬空为高电平）时发生时钟同步。当 SCLx 引脚被释放时，波特率发生器（BRG）将暂停计数，直到实际采样到 SCLx 引脚为高电平为止。当 SCLx 引脚采样为高电平时，波特率发生器将被重新装入 I2CxBRG<8:0> 的内容并开始计数。这可以确保当外部器件将时钟拉低时，SCLx 始终至少保持一个 BRG 计满返回周期的高电平，如图 19-18 所示。

图 19-18： 带有时钟同步的波特率发生器时序



① 波特率计数器每个 TCY 递减两次。在计满返回时，主器件 SCLx 的电平将翻转。

② 从器件将 SCLx 下拉为低电平，启动等待。

③ 在主器件波特率计数器本应发生计满返回时，检测到 SCLx 为低电平则暂停计数器。

④ 逻辑在每个 TCY 对 SCLx 采样一次。逻辑检测到 SCLx 为高电平。

⑤ 波特率计数器在下一周期发生计满返回。

⑥ 在下一次计满返回发生时，主器件 SCLx 的电平将翻转。

19.6.3 总线仲裁和总线冲突

总线仲裁支持多主机系统操作。SDAx 线的线“与”特性使其可以进行总线仲裁。当第一个主器件通过将 SDAx 悬空为高电平而在 SDAx 上输出 1，与此同时，第二个主器件通过下拉 SDAx 为低电平而在 SDAx 上输出 0，发生总线仲裁。SDAx 信号将变为低电平。这种情况下，第二个主器件在总线仲裁中获胜。第一个主器件在总线仲裁中失败，从而产生总线冲突。

对于第一个主器件，期望 SDAx 上的数据是 1，但在 SDAx 上采样到的数据却是 0。这即是总线冲突的定义。

第一个主器件会将总线冲突位 BCL（I2CxSTAT<10>）置 1，并产生主器件中断。主模块会将 I²C 端口复位为其空闲状态。

在多主机操作中，必须对 SDAx 线进行仲裁监视，以查看信号电平是否为期望的输出电平。该检查由主模块执行，检查结果置于 BCL 状态位中。

可能导致仲裁失败的情况是：

- 启动条件
- 重复启动条件
- 地址、数据或应答位
- 停止条件

19.6.4 检测总线冲突和重新发送报文

当发生总线冲突时，模块会将 BCL 状态位置 1 并产生主器件中断。如果在字节发送过程中发生总线冲突，则发送会被中止，TBF 状态位清零，SDAx 和 SCLx 引脚被释放。如果在启动、重复启动、停止或应答条件期间发生总线冲突，则条件会被中止，I2CxCON 寄存器中的相应控制位被清零，SDAx 和 SCLx 线被释放。

软件期待在主器件事件完成时产生中断。软件可以通过检查 BCL 状态位来确定主器件事件是已成功完成还是发生了总线冲突。如果发生总线冲突，软件必须中止发送剩余的待发报文，并准备重新发送整个报文序列，即在总线返回到空闲状态后从启动条件开始发送。软件可以通过监视 S 和 P 状态位来等待总线空闲。当软件执行主器件中断服务程序且 I²C 总线空闲时，软件可以通过发送启动条件重新开始通信。

19.6.5 启动条件期间的总线冲突

在发出启动命令之前，软件应使用 S 和 P 状态位检查总线是否处于空闲状态。可能出现两个主器件在差不多同一时刻尝试启动报文传输。通常，主器件将同步时钟并在发送报文期间继续进行总线仲裁，直到其中一个主器件仲裁失败。但是，以下条件会导致在启动条件期间发生总线冲突：

- 如果 SDA 和 SCL 引脚在启动条件开始时处于逻辑低电平状态，或者
- 如果 SCL 线在 SDA 线被驱动为低电平之前就处于逻辑低电平状态

在任何一种情况下，在启动条件期间仲裁失败的主器件会产生总线冲突中断。

19.6.6 重复启动条件期间的总线冲突

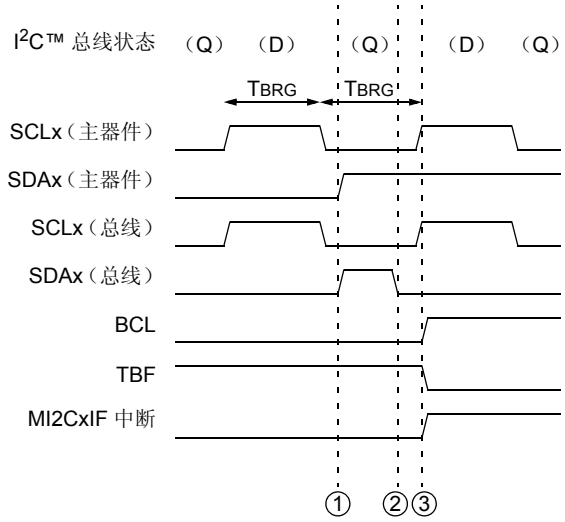
可能会有两个主器件在整个地址字节发送期间未发生冲突，但当一个主器件尝试发送重复启动条件，而另一个发送数据时可能发生冲突。在这种情况下，产生重复启动条件的主器件将仲裁失败，并产生总线冲突中断。

19.6.7 报文位发送期间的总线冲突

数据冲突最典型的情况是当主器件尝试发送器件地址字节、数据字节或应答位时。

如果软件能正确地检查总线状态，则不太可能会在启动条件期间发生总线冲突。但是，因为另一主器件可能会在非常相近的时间检查总线并产生其自身的启动条件，所以可能会发生 SDAx 仲裁，并对两个主器件的启动条件进行同步。在这种情况下，两个主器件都会开始并继续发送它们的报文，直到其中一个主器件在某个报文位仲裁失败。请记住， SCLx 时钟同步将使两个主器件保持同步，直到其中一个仲裁失败。图 19-19 给出了报文位仲裁的示例。

图 19-19： 报文位发送期间的总线冲突



① 主器件在下一个 SCLx 时钟发送位值 1。
模块释放 SDAx。

② 总线上的另一主器件在下一个 SCLx 时钟发送位值 0。
另一主器件将 SDAx 下拉为低电平。

③ 波特率发生器超时。模块尝试检查 SDAx 是否为高电平。
检测到总线冲突。模块释放 SDAx 和 SCLx。模块将 BCL 状态位置 1 并将 TBF 状态位清零。主器件产生中断。

19.6.8 停止条件期间的总线冲突

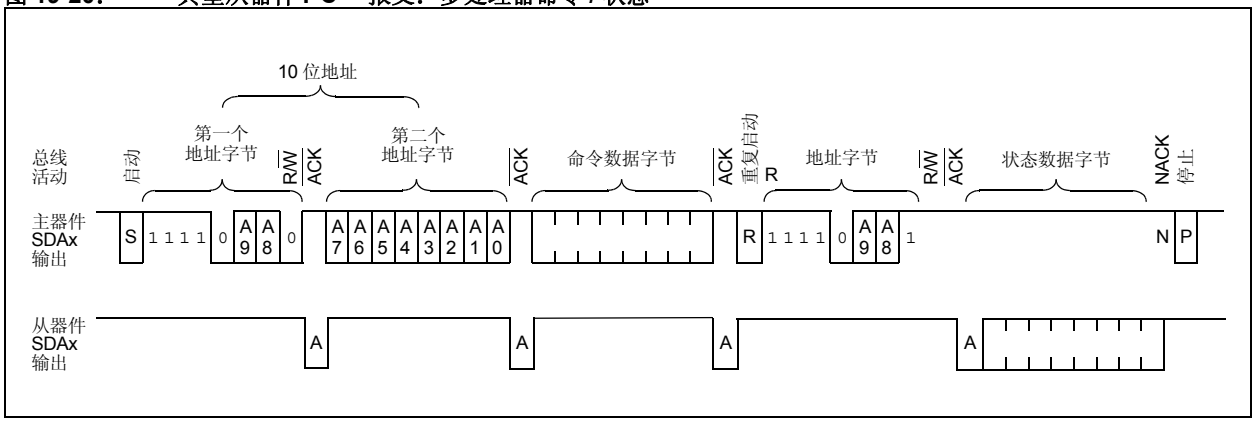
如果主器件软件失去了对 I2C 总线状态的跟踪，则在停止条件期间会有很多情况导致总线冲突。在这种情况下，产生停止条件的主器件将仲裁失败，并产生总线冲突中断。

19.7 作为从器件进行通信

在一些系统中，特别是有多个处理器相互通信的系统中， PIC24H 器件可以作为从器件进行通信（见图 19-20）。当模块使能时，从模块处于工作状态。从器件不能启动报文传输，它只能对由主器件启动的报文序列作出响应。主器件请求I²C协议中器件地址字节定义的特定从器件作出响应。从模块在协议所定义的适当时间对主器件作出答复。

与用作主模块时一样，软件产生用于答复的协议组成部分的序列。但是，当从器件地址与软件为它指定的地址匹配时，从模块会检测到。

图 19-20： 典型从器件 I²C™ 报文：多处理器命令 / 状态



在检测到启动条件之后，从模块会接收并检查器件地址。从器件可以指定 7 位地址或 10 位地址。当器件地址匹配时，模块将产生中断，通知软件其器件被选中。根据主器件发送的 R/W 状态位，从器件将接收或发送数据。如果是要求从器件接收数据，从模块会自动产生应答 （ACK），将 I2CxRSR 寄存器中接收到的当前值装入 I2CxRCV 寄存器，并通过中断通知软件。如果是要求从器件发送数据，则软件必须将数据值装入 I2CxTRN 寄存器。

19.7.1 采样接收数据

在时钟 （SCLx）线的上升沿采样所有的输入位。

19.7.2 检测启动和停止条件

从模块将检测总线上的启动和停止条件，并以S状态位（I2CxSTAT<3>）和P状态位（I2CxSTAT<4>）指示该状态。启动 （S）位和停止 （P）位在复位时或模块被禁止时清零。在检测到启动或重复启动事件之后，S 状态位置 1，P 状态位清零。在检测到停止事件之后，P 状态位置 1，S 状态位清零。

19.7.3 检测地址

一旦模块被使能，从模块就会等待启动条件发生。在检测到启动条件之后，从器件将根据 A10M 位 （I2CxCON<10>）的值尝试检测 7 位或 10 位地址。对于 7 位地址，从模块将比较一个接收字节；对于 10 位地址，从模块将比较两个接收字节。 7 位地址中还包含一个 R/W 状态位，该位指定跟在地址后的数据的传输方向。如果 R/W = 0，则指定进行写操作，从器件将接收来自主器件的数据。如果 R/W = 1，则指定进行读操作，从器件将向主器件发送数据。10 位地址中包含一个 R/W 状态位，但根据定义，它始终为 R/W = 0，因为从器件必须接收 10 位地址的第二个字节。

19.7.3.1 从器件地址掩码

I2CxMSK 寄存器用于掩码一些地址位，在 10 位和 7 位地址模式下将这些位指定为 “无关位”。当 I2CxMSK 寄存器中的某位置 1 （= 1）时，无论地址的相应位置中的位为 0 还是 1，从模块都会作出响应。例如，在 7 位从模式下， I2CxMSK = 0100000，模块将认为地址 0000000 和 0100000 有效。

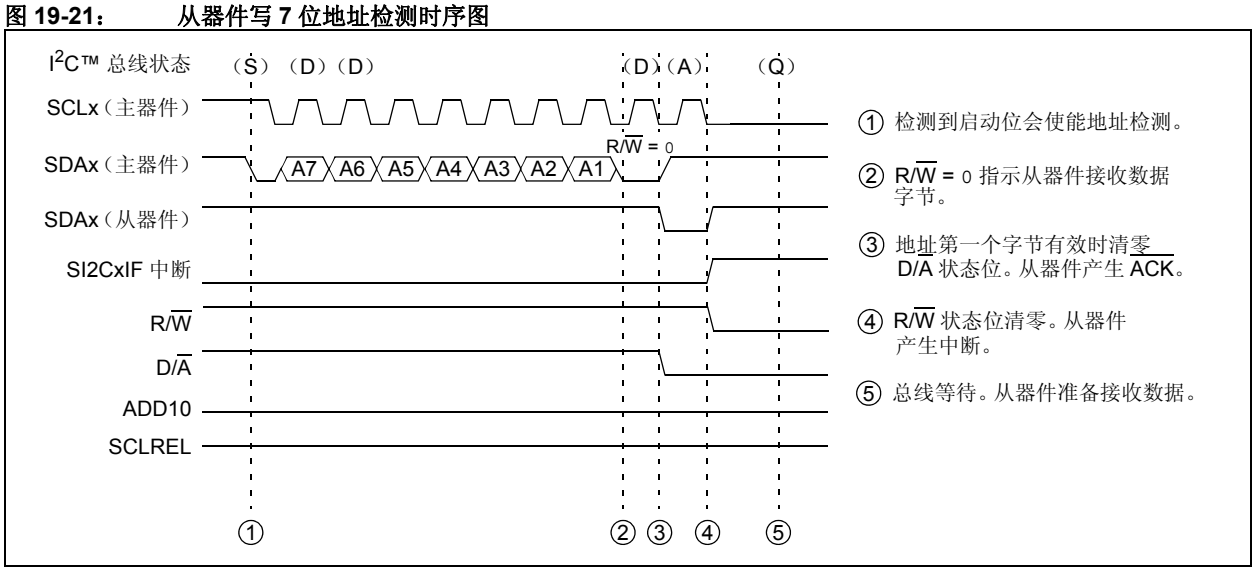
要使能地址掩码，必须通过清零 IPMIEN 位（I2CxCON<11>）禁止智能平台管理接口（IPMI）。

19.7.3.2 7 位地址和从器件写操作

在检测到启动条件之后，模块会将 8 个位移入 I2CxRSR 寄存器 （见图 19-21）。在第 8 个时钟（SCLx）的下降沿，根据 I2CxADD<6:0> 和 I2CxMSK<6:0> 寄存器的值求寄存器 I2CxRSR<7:1> 的值。如果地址有效 （即，所有未掩码位完全匹配），则发生以下事件：

1. 产生 $\overline{\text{ACK}}$。
2. D/$\overline{\text{A}}$ 和 R/$\overline{\text{W}}$ 状态位清零。
3. 在第 9 个 SCLx 时钟的下降沿，模块产生 SI2CxIF 中断。
4. 模块将等待主器件发送数据。

图 19-21： 从器件写 7 位地址检测时序图



19.7.3.3 7 位地址和从器件读操作

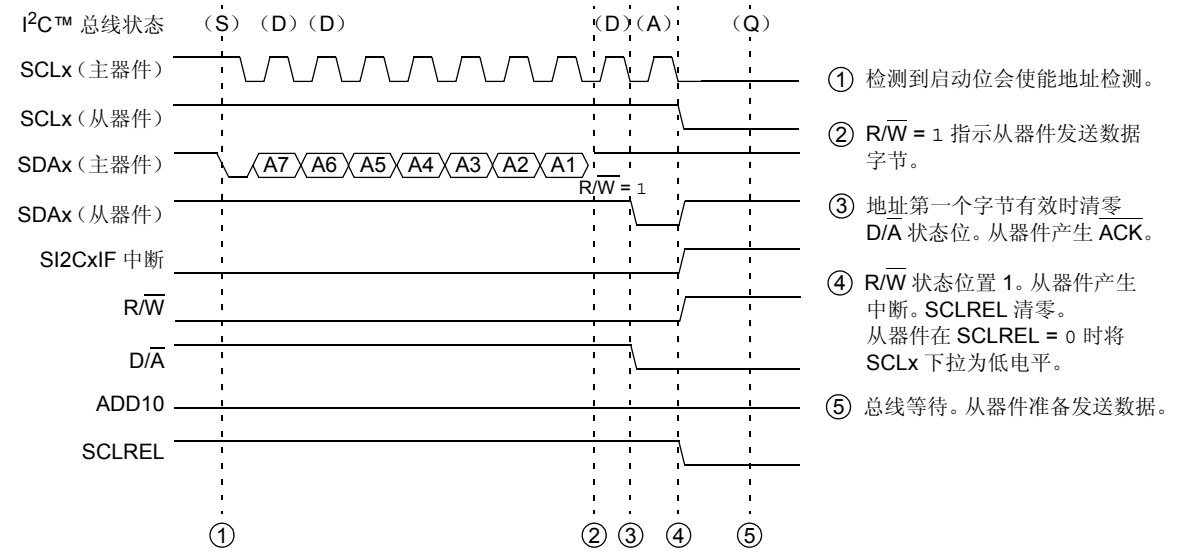
当 7 位地址字节中的 R/$\overline{\text{W}}$ = 1，指定执行从器件读操作时，检测器件地址的过程与从器件写操作类似 （见图 19-22）。如果地址匹配，则发生以下事件：

1. 产生 $\overline{\text{ACK}}$。
2. D/$\overline{\text{A}}$ 状态位清零， R/$\overline{\text{W}}$ 状态位置 1。
3. 在第 9 个 SCLx 时钟的下降沿，模块产生 SI2CxIF 中断。

由于希望从模块此时发送数据作为答复，因此必须暂停 I^2C 总线的操作，以使软件作好响应的准备。当模块清零 SCLREL 位时，这会自动完成。 SCLREL 为 0 时，从模块会将 SCLx 时钟线下拉为低电平，从而在 I^2C 总线上产生等待。从模块和 I^2C 总线将保持在该状态，直到软件在 I2CxTRN 寄存器中写入响应数据并将 SCLREL 位置 1。

注：	无论 STREN 位的状态如何，在检测到从器件读操作地址之后， SCLREL 都将自动清零。

图 19-22：　从器件读 7 位地址检测时序图



19.7.3.4　10 位寻址模式

在 10 位寻址模式下，从器件必须接收两个器件地址字节（见图 19-23）。第一个地址字节的高 5 位（MSb）将指定这是一个 10 位地址。地址的 R/W 状态位必须指定写操作，以使从器件接收第二个地址字节。对于 10 位地址，第一个字节应为 “11110 A9 A8 0”，其中 “A9” 和 “A8” 为地址的两个最高位。

I2CxMSK 寄存器可以掩码 10 位地址中的任何位。 I2CxMSK 的两个最高位用于掩码在第一个字节接收的进入地址的最高位。而寄存器的剩余字节用于掩码在第二个字节中接收的地址低字节。

在检测到启动条件之后，模块会将 8 个位移入 I2CxRSR 寄存器。 I2CxRSR<2:1> 位的值根据 I2CxADD<9:8> 和 I2CxMSK<9:8> 位的值求得，而 I2CxRSR<7:3> 位的值则与 “11110” 比较。地址求值在第 8 个时钟（SCLx）的下降沿进行。要使地址有效， I2CxRSR<7:3> 必须等于 “11110”，而 I2CxRSR<2:1> 必须与 I2CxADD<9:8> 中的任何未掩码位完全匹配。（如果这两个位都被掩码，则无需匹配。）如果地址有效，则发生以下事件：

1. 产生 ACK。
2. D/A 和 R/W 状态位清零。
3. 在第 9 个 SCLx 时钟的下降沿，模块产生 SI2CxIF 中断。

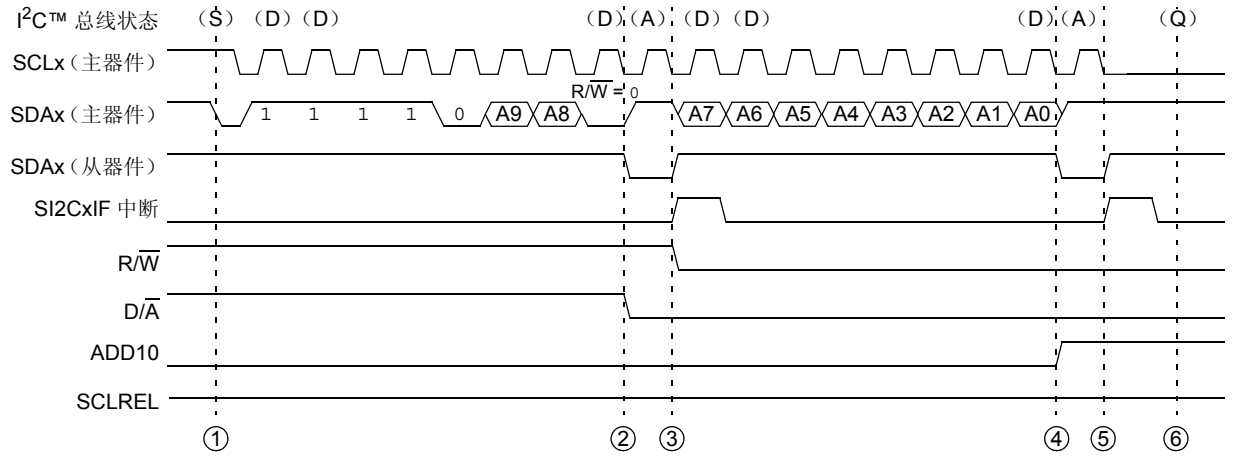
在接收到 10 位地址的第一个字节后，模块也会产生中断；但该中断几乎没有什么用处。

模块将继续接收第二个字节到 I2CxRSR 寄存器中。此时， I2CxRSR<7:0> 位根据 I2CADD<7:0> 和 I2CxMSK<7:0> 位进行求值。如果地址低字节有效 （如前文所述），则发生以下事件：

1. 产生 ACK。
2. ADD10 状态位置 1。
3. 在第 9 个 SCLx 时钟的下降沿，模块产生 SI2CxIF 中断。
4. 模块将等待主器件发送数据或发起重复启动条件。

注：	在 10 位寻址模式下检测到重复启动条件之后，从模块只会匹配前 7 位地址 “11110 A9 A8 0”。

图 19-23： 10 位地址检测时序图



① 检测到启动位会使能地址检测。

② 如果第一个字节地址匹配，则会清零 D/$\overline{A}$ 状态位并导致从器件逻辑产生 $\overline{ACK}$。

③ 接收到第一个字节时会清零 R/$\overline{W}$ 状态位。从器件逻辑产生中断。

④ 如果第一个字节和第二个字节地址匹配，则会将 ADD10 状态位置 1，并导致从器件逻辑产生 $\overline{ACK}$。

⑤ 接收到第二个字节即完成 10 位地址。从器件逻辑产生中断。

⑥ 总线等待。从器件准备接收数据。

19.7.3.5 广播呼叫操作

在 I²C 总线的寻址过程中，通常由启动条件后的第一个字节决定主器件将寻址哪个从器件。但广播呼叫地址例外，它能寻址所有器件。当使用这个地址时，所有已使能的器件都应该发送一个应答信号来响应。广播呼叫地址是由 I²C 协议为特定目的保留的 8 个地址之一。它由全 0 组成，且 R/$\overline{W}$ = 0。广播呼叫始终执行从器件写操作。

当广播呼叫使能位 GCEN （I2CxCON<7>）置 1 时，即识别为广播呼叫地址 （见图 19-24）。在检测到启动位之后，8 个位移入 I2CxRSR 寄存器，并且将地址与 I2CxADD 寄存器进行比较，同时也与广播呼叫地址进行比较。

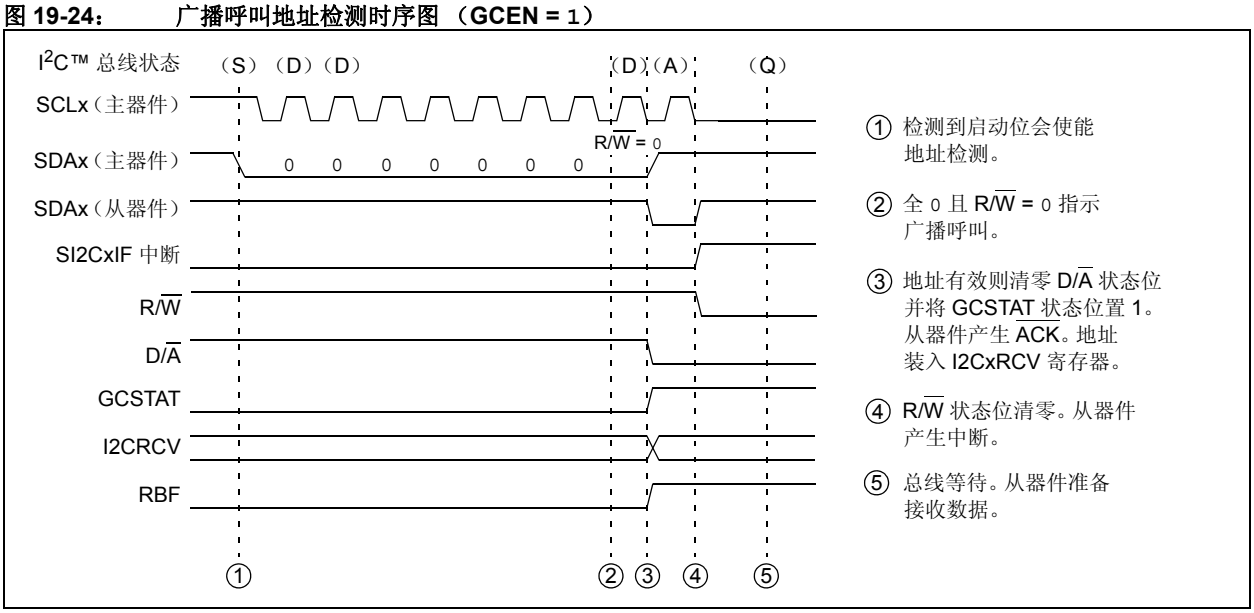
如果广播呼叫地址匹配，则发生以下事件：

1. 产生 $\overline{ACK}$。
2. 从模块会将 GCSTAT 状态位 （I2CxSTAT<9>）置 1。
3. D/$\overline{A}$ 和 R/$\overline{W}$ 状态位清零。
4. 在第 9 个 SCLx 时钟的下降沿，模块产生 SI2CxIF 中断。
5. I2CxRSR 寄存器中数据传送到 I2CxRCV 寄存器， RBF 状态位置 1 （在第 8 位期间）。
6. 模块将等待主器件发送数据。

当处理中断时，可以通过读 GCSTAT 状态位的内容来检查中断原因，以确定器件地址是特定于器件还是广播呼叫地址。

请注意，广播呼叫地址是 7 位地址。如果将从模块配置为 10 位地址，且 A10M 和 GCEN 位置 1，从模块还是会检测 7 位广播呼叫地址。

图 19-24： 广播呼叫地址检测时序图 （GCEN = 1）

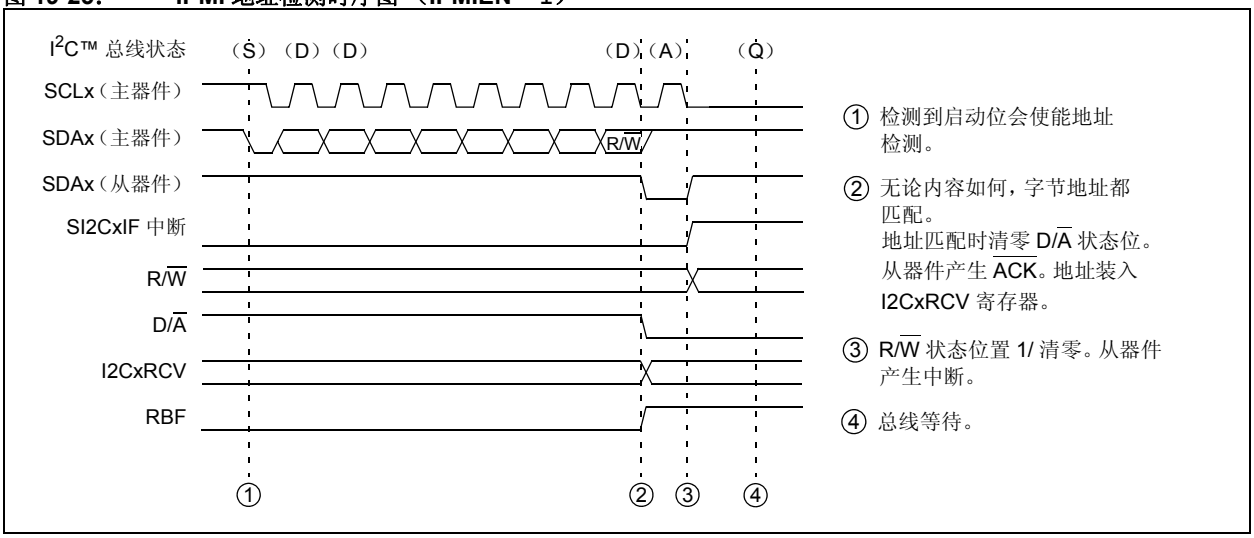


19.7.3.6 接收所有地址 （IPMI 操作）

一些 I²C 系统协议要求从器件对总线上的所有报文均进行操作。例如， IPMI （智能平台管理接口）总线使用 I²C 节点作为分布式网络中的报文转发器。要允许节点转发所有报文，从模块必须接受所有报文，无论器件地址如何。

要使能 IPMI 模式，需将 IPMIEN 位（I2CxCON<11>）置 1（见图 19-25）。无论 A10M 和 GCEN 位的状态如何，或装入 I2CxADD 寄存器的值如何，所有地址都会被接受。

图 19-25： IPMI 地址检测时序图 （IPMIEN = 1）



19.7.3.7 地址有效时

如果 7 位地址与 I2CxADD<6:0> 的内容不匹配，从模块将返回到空闲状态并忽略所有总线活动，直到检测到停止条件。

如果 10 位地址的第一个字节与 I2CxADD<9:8> 的内容不匹配，从模块将返回到空闲状态并忽略所有总线活动，直到检测到停止条件。

如果 10 位地址的第一个字节与 I2CxADD<9:8> 的内容匹配，但 10 位地址的第二个字节与 I2CxADD<7:0> 不匹配，从模块将返回到空闲状态并忽略所有总线活动，直到检测到停止条件。

19.7.3.8 保留为不掩码的地址

即使在使能掩码时，有几个地址在硬件中也被掩码排除在外。对于这些地址，无论掩码设置如何，始终不会发送应答。表 19-3 中列出了这些地址。

表 19-3： 保留的 I^2C 总线地址 (1)

7 位地址模式：		
从器件地址	**R/$\overline{W}$ 位**	**说明**
0000 000	0	广播呼叫地址 (1)
0000 000	1	启动字节
0000 001	x	CBUS 地址
0000 010	x	保留
0000 011	x	保留
0000 1xx	x	HS 模式主机码
1111 1xx	x	保留
1111 0xx	x	10 位从地址高字节 (2)

注 1： 只有 GCEN = 1 时才会应答地址。

2： 只有 10 位寻址模式下的高字节才会发生与该地址匹配。

19.7.4 接收来自主器件的数据

当地址字节的 R/$\overline{W}$ 状态位为 0 并发生地址匹配时，R/$\overline{W}$ 状态位（I2CxSTAT<2>）清零。从模块进入等待主器件发送数据的状态。在器件地址字节之后，数据字节的内容由系统协议定义，且仅由从模块接收。

从模块将 8 个位移入 I2CxRSR 寄存器。在第 8 个时钟 （SCLx）的下降沿，发生以下事件：

1. 模块开始产生 $\overline{ACK}$ 或 NACK。
2. RBF 状态位置 1，指示接收到数据。
3. I2CxRSR 寄存器字节传送到 I2CxRCV 寄存器，供软件访问。
4. D/$\overline{A}$ 状态位置 1。
5. 产生从器件中断。软件可以通过检查 I2CxSTAT 寄存器的状态来确定事件原因，然后清零 SI2CxIF 中断标志。
6. 模块将等待下一个数据字节。

19.7.4.1 应答产生

通常，从模块将通过在第 9 个 SCLx 时钟发送 $\overline{ACK}$ 对所有接收的字节作出应答。如果接收缓冲区溢出，则从模块不会产生该 $\overline{ACK}$。如果以下两种情况有一种（或同时）存在，则说明发生溢出：

1. 在接收到数据前，缓冲区满位 RBF （I2CxSTAT<1>）被置 1。
2. 在接收到数据前，溢出位 I2COV （I2CxSTAT<6>）被置 1。

表 19-4 显示了在给定 RBF 和 I2COV 状态位状态时，接收到数据传输字节时发生的情况。如果在从模块尝试向 I2CxRCV 寄存器传送数据时， RBF 状态位已经置 1，则不发生传送，但会产生中断，且 I2COV 状态位置 1。如果 RBF 和 I2COV 状态位均置 1，从模块会执行类似操作。阴影单元显示了在软件没有正确清除溢出条件时的情况。

读 I2CxRCV 寄存器会清零 RBF 状态位。 I2COV 状态位通过由软件写入 0 而清零。

表 19-4： 接收到传输数据后的操作

接收到数据字节时的状态位		将 I2CxRSR 值传送到 I2CxRCV	产生 $\overline{ACK}$	产生 SI2CxIF 中断（如果允许则发生中断）	RBF 置 1	I2COV 置 1
RBF	I2COV					
0	0	是	是	是	是	不变
1	0	否	否	是	不变	是
1	1	否	否	是	不变	是
0	1	是	否	是	是	不变

图注： 阴影单元显示了在软件没有正确清除溢出条件时的情况。

19.7.4.2 从器件接收期间的等待状态

当从模块接收到数据字节时，主器件可能会立即开始发送下一字节。这使控制从模块的软件可以有 9 个 SCLx 时钟周期来处理先前接收到的字节。如果该时间不够充足，从器件软件可以产生总线等待周期。

STREN 位（I2CxCON<6>）用于在从器件接收期间产生总线等待。当所接收字节的第 9 个 SCLx 时钟的下降沿 STREN = 1 时，从模块会清零 SCLREL 位。清零 SCLREL 位会使从模块将 SCLx 线下拉为低电平，从而产生等待。主器件和从器件的 SCLx 时钟将进行同步，如**第 19.6.2 节“主器件时钟同步”**中所示。

当软件准备好恢复接收时，软件将 SCLREL 置 1。这将使从模块释放 SCLx 线，从而主器件可以继续发送时钟。

19.7.4.3 从器件接收的示例报文

接收从器件报文是一个自动的过程。处理从器件协议的软件使用从器件中断来与事件同步。

当从器件检测到有效地址时，关联的中断将通知软件等待接收报文。在接收数据时，每个数据字节传送到 I2CxRCV 寄存器后，会产生中断通知软件读缓冲区。

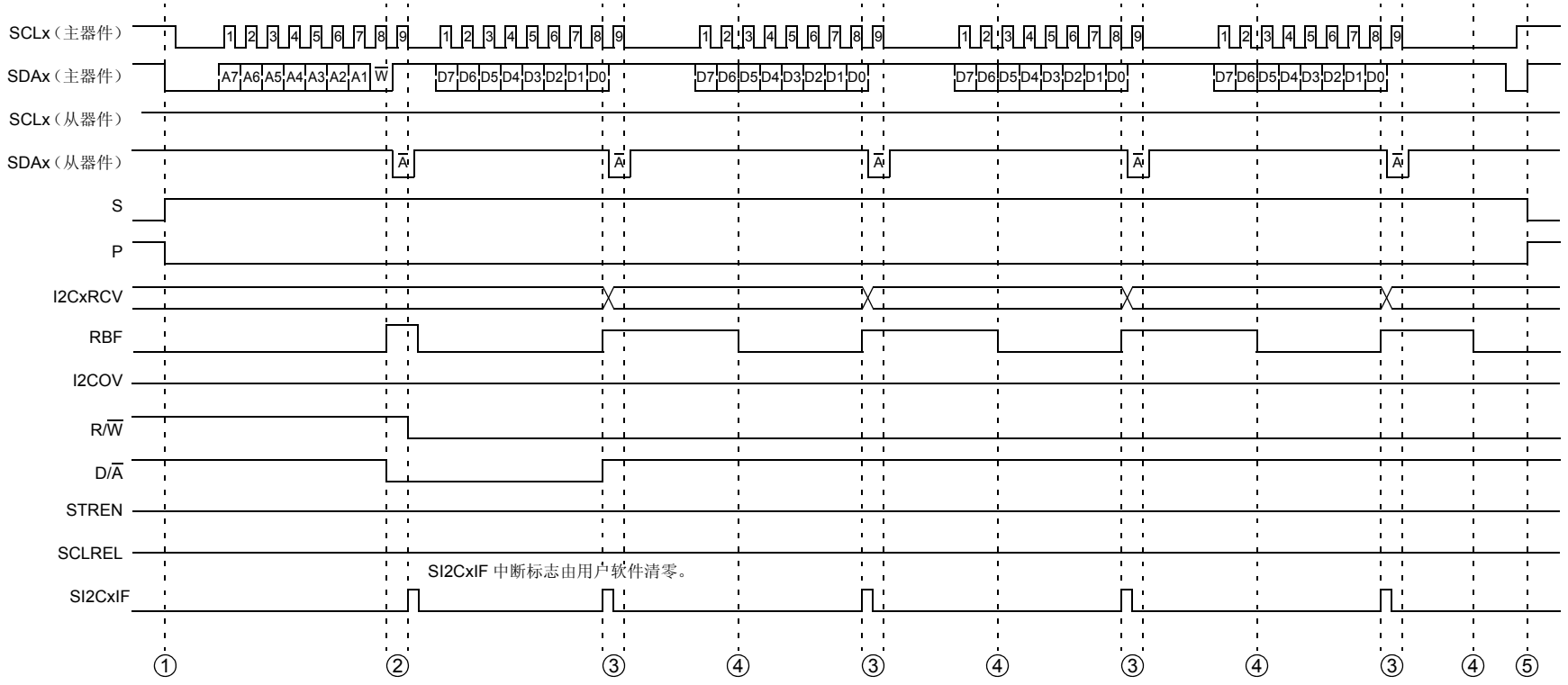
图 19-26 所示为一个简单的接收报文。由于是 7 位地址报文，接收到地址字节时仅产生一次中断。然后，每 4 个数据字节产生一次中断。在发生中断时，软件可以监视 RBF、 D/A 和 R/W 状态位来确定所接收字节的状态。

图 19-27 所示为使用 10 位地址的类似报文。在这种情况下，地址需要使用两个字节。

图 19-28 显示的情况是软件不对所接收字节作出响应，并且缓冲区溢出。在接收第二个字节时，模块将自动向主器件发送 NACK。通常，这会使主器件重新发送前一个字节。I2COV 状态位指示缓冲区发生溢出。I2CxRCV 寄存器缓冲区保留第一个字节的内容。在接收到第三个字节时，缓冲区仍然为满，模块将再次向主器件发送 NACK。在此之后，软件最终读取了缓冲区。读缓冲区将清零 RBF 状态位，但 I2COV 状态位仍保持置 1。软件必须清零 I2COV 状态位。下一个接收到的字节将移到 I2CxRCV 寄存器缓冲区，之后模块将发送 ACK 作为响应。

图 19-29 显示了在接收数据时的时钟延长。在前面的示例中 STREN = 0，这将在接收报文时禁止时钟延长。在该示例中，软件将 STREN 置 1 来使能时钟延长。当 STREN = 1 时，模块将在接收到每个数据字节之后自动延长时钟，使软件可以有更多时间从缓冲区移动数据。如果在第 9 个时钟的下降沿 RBF = 1，则模块将自动清零 SCLREL 位并将 SCLx 线下拉为低电平。如图所示，对于接收到的第二个数据字节，如果软件可以在第 9 个时钟的下降沿之前读缓冲区并清零 RBF 状态位，则不会产生时钟延长。软件也可以随时暂挂总线。通过清零 SCLREL 位，模块将在检测到总线 SCLx 为低电平之后将 SCLx 下拉为低电平。SCLx 线将保持低电平，暂挂总线上的事务，直到 SCLREL 位置 1。

图 19-26： 从器件报文（向从器件写数据：7 位地址；地址匹配； A10M = 0； GCEN = 0； IPMIEN = 0）



① 从器件识别启动事件； S 和 P 状态位相应地置 1/ 清零。

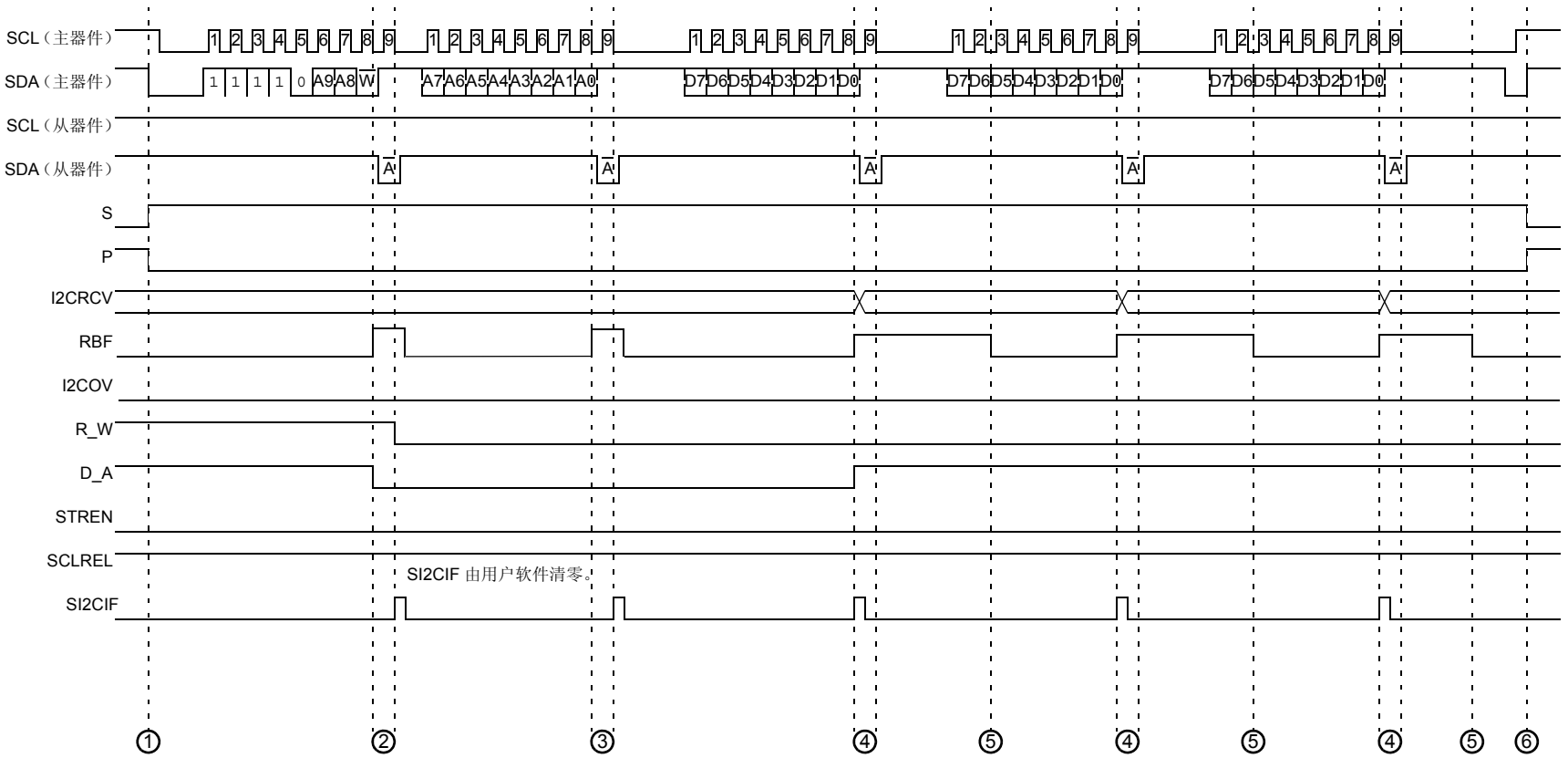
② 从器件接收地址字节。地址匹配。从器件应答并产生中断。地址字节移入 I2CxRCV 寄存器，且必须由用户软件读取，以防止缓冲区溢出。

③ 下一个接收字节为报文数据。字节移入 I2CxRCV 寄存器，RBF 状态位置 1。从器件产生中断。从器件应答接收。

④ 软件读 I2CxRCV 寄存器。RBF 状态位清零。

⑤ 从器件识别停止事件； S 和 P 状态位相应地置 1/ 清零。

图 19-27： 从器件报文（向从器件写数据：10 位地址；地址匹配； A10M = 1； GCEN = 0； IPMIEN = 0）



① - 从器件识别启动事件，S 和 P 位相应地置 1/ 清零。

② - 从器件接收地址字节。高字节地址匹配。
从器件应答并产生中断。地址字节移入 I2CRCV 寄存器，
且由用户软件读取，以防止缓冲区溢出。

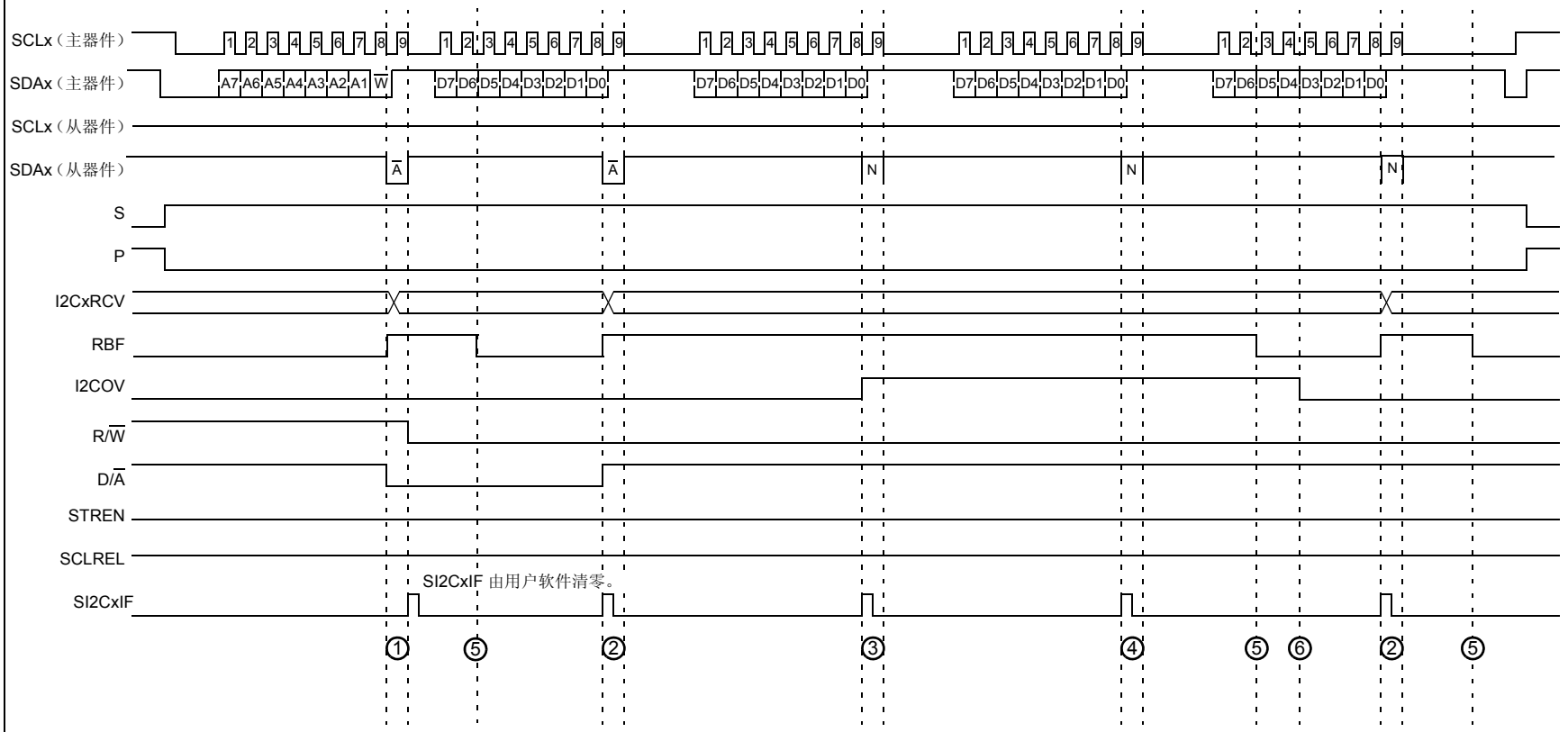
③ - 从器件接收地址字节。低字节地址匹配。
从器件应答并产生中断。地址字节移入 I2CRCV 寄存器，
且由用户软件读取，以防止缓冲区溢出。

④ - 下一个接收字节为报文数据。字节移入 I2CxRCV 寄存器，RBF 置 1。
从器件应答并产生中断。

⑤ - 软件读 I2CRCV 寄存器。RBF 位清零。

⑥ - 从器件识别停止事件，S 和 P 位相应地置 1/ 清零。

图 19-28： 从器件报文（向从器件写数据：7 位地址；缓冲区溢出； A10M = 0； GCEN = 0； IPMIEN = 0）



① - 从器件接收地址字节。地址匹配。从器件产生中断。地址字节移入 I2CxRCV 寄存器，且必须由用户软件读取，以防止缓冲区溢出。

② - 下一个接收字节为报文数据。字节移动到 I2CxRCV 寄存器，RBF 置 1。从器件产生中断。从器件应答接收。

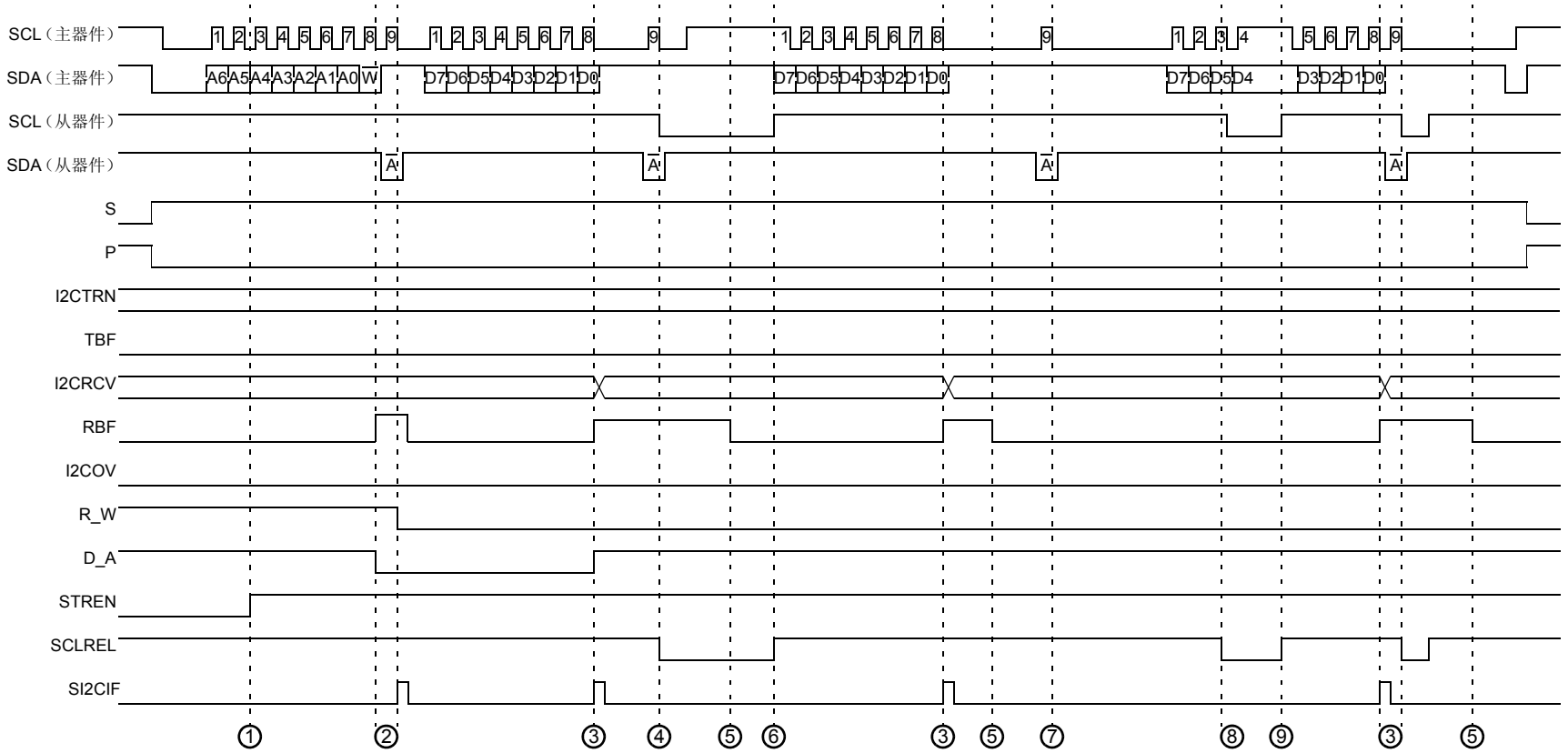
③ - 在软件读 I2CxRCV 之前接收到下一字节。I2CxRCV 寄存器不变。I2COV 溢出位置 1。从器件产生中断。从器件发送 NACK 响应接收。

④ - 在软件读 I2CxRCV 之前接收到下一字节。I2CxRCV 寄存器不变。从器件产生中断。从器件发送 NACK 响应接收。主器件状态机不应在以该方式接收 NACK 后，设定发送另一个字节。而是应该中止发送（使用停止条件），或发送重复启动条件并尝试重新发送数据。

⑤ - 软件读 I2CxRCV 寄存器。RBF 位清零。

⑥ - 软件清零 I2COV 位。接收仍无法正常继续，直到模块检测到停止 / 重复启动位。如果这些条件都不满足，会正确接收额外的发送，但会发送 NACK 并再次将 I2COV 位置 1。

图 19-29： 从器件报文（向从器件写数据：7 位地址；使能时钟延长； A10M = 0； GCEN = 0； IPMIEN = 0）



① - 软件将 STREN 位置 1 以使能时钟延长。

② - 从器件接收地址字节。I2CRCV 寄存器由用户软件读取，以防止缓冲区溢出。

③ - 下一个接收字节为报文数据。字节移入 I2CxRCV 寄存器，RBF 置 1。

④ - 因为在第 9 个时钟 RBF = 1，开始自动时钟延长。从器件清零 SCLREL 位。从器件将 SCL 线下拉为低电平以延长时钟。

⑤ - 软件读 I2CRCV 寄存器。RBF 位清零。

⑥ - 软件将 SCLREL 位置 1 以释放时钟。

⑦ - 从器件不清零 SCLREL，因为此时 RBF = 0。

⑧ - 软件可以清零 SCLREL 来产生时钟保持。模块必须先检测到 SCL 为低电平，之后才能将 SCL 下拉为低电平。

⑨ - 软件可以通过将 SCLREL 置 1 来释放时钟保持。

19.7.5 向主器件发送数据

当进入的器件地址字节的 R/$\overline{W}$ 状态位为 1，且地址匹配时，R/$\overline{W}$ 状态位（I2CxSTAT<2>）置 1。此时，主器件等候从器件通过发送数据字节作出响应。字节的内容由系统协议定义，且仅由从模块发送。

当地址检测产生中断时，软件可以通过向 I2CxTRN 寄存器写一个字节来启动数据发送。

从模块将 TBF 状态位置 1。 8 个数据位在 SCLx 输入的下降沿被移出。这可确保在 SCLx 为高电平期间 SDAx 信号是有效的。所有 8 个位都移出后， TBF 状态位被清零。

从模块在第 9 个 SCLx 时钟的上升沿检测来自主器件 - 接收器的应答。

如果 SDAx 线为低电平，指示应答 （$\overline{ACK}$），说明主器件需要更多的数据，报文尚未完成。模块会产生从器件中断，可检查 ACKSTAT 状态位来判断是否在请求更多数据。

在第 9 个 SCLx 时钟的下降沿产生从器件中断。软件必须检查 I2CxSTAT 寄存器的状态并清零 SI2CxIF 中断标志。

如果 SDAx 线为高电平，指示不应答（NACK），说明数据传输已完成。从模块复位并产生中断，它将等待直到检测到下一个启动位。

19.7.5.1 从器件发送期间的等待状态

在从器件发送报文期间，主器件在检测到有效地址且 R/$\overline{W}$ = 1 之后将期待立即返回数据。出于此原因，每当从模块返回数据时，从模块将自动产生总线等待。

在有效器件地址字节的第 9 个 SCLx 时钟的下降沿或主器件对发送字节产生应答时，发生自动等待，指示希望发送更多数据。

从模块清零 SCLREL 位。清零 SCLREL 位会使从模块将 SCLx 线下拉为低电平，从而产生等待。主器件和从器件的 SCLx 时钟将进行同步，如**第 19.6.2 节 “主器件时钟同步”**中所示。

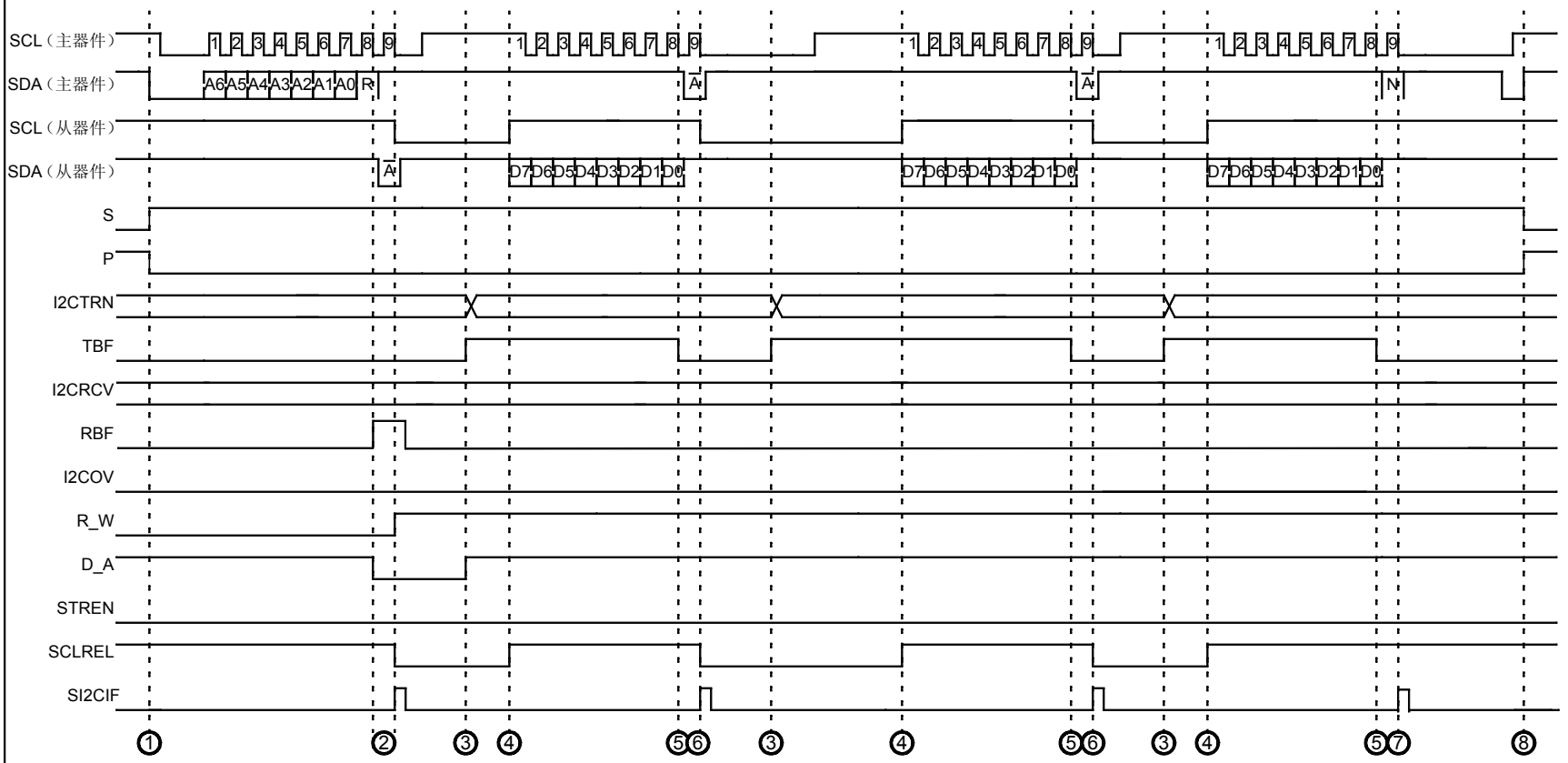
当软件在 I2CxTRN 中装入值并准备好恢复发送时，软件将 SCLREL 置 1。这将使从模块释放 SCLx 线，从而主器件可以恢复产生时钟。

19.7.5.2 从器件发送的示例报文

图 19-30 中显示了 7 位地址报文的从器件发送。当地址匹配且地址的 R/$\overline{W}$ 状态位指示进行从器件发送时，模块将通过清零 SCLREL 位自动产生时钟延长，并产生中断来指示需要发送响应字节。软件需将响应字节写入 I2CxTRN 寄存器。在发送完成时，主器件将发送应答作为响应。如果主器件答复为 $\overline{ACK}$，则说明主器件需要更多数据，此时模块将再次清零 SCLREL 位并产生另一中断。如果主器件发送 NACK 作为响应，则说明不再需要发送数据，此时模块将不会延长时钟，也不会产生中断。

对应于 10 位地址报文的从器件发送要求从器件首先识别 10 位地址。因为主器件必须发送两个字节的地址，所以地址的第一个字节中的 R/$\overline{W}$ 状态位必须指定执行写操作。要将报文更改为执行读操作，主器件需要发送重复启动条件并重复发送地址的第一个字节，这时第一个字节的 R/$\overline{W}$ 状态位应指定执行读操作。此时，从器件发送开始，如图 19-31 所示。

图 19-30：　从器件报文（自从器件读数据：7 位地址）



① - 从器件识别启动事件，S 和 P 位相应地置 1/ 清零。

② - 从器件接收地址字节。地址匹配。从器件产生中断。地址字节移入 I2CRCV 寄存器，且由用户软件读取，以防止缓冲区溢出。
R_W = 1，指示自从器件读数据。SCLREL = 0，暂挂主器件时钟。

③ - 软件向 I2CTRN 中写入响应数据。TBF = 1 指示缓冲区已满。
写 I2CTRN 将 D_A 置 1，指示是数据字节。

④ - 软件将 SCLREL 位置 1 以释放时钟保持。主器件继续发送时钟，从器件发送数据字节。

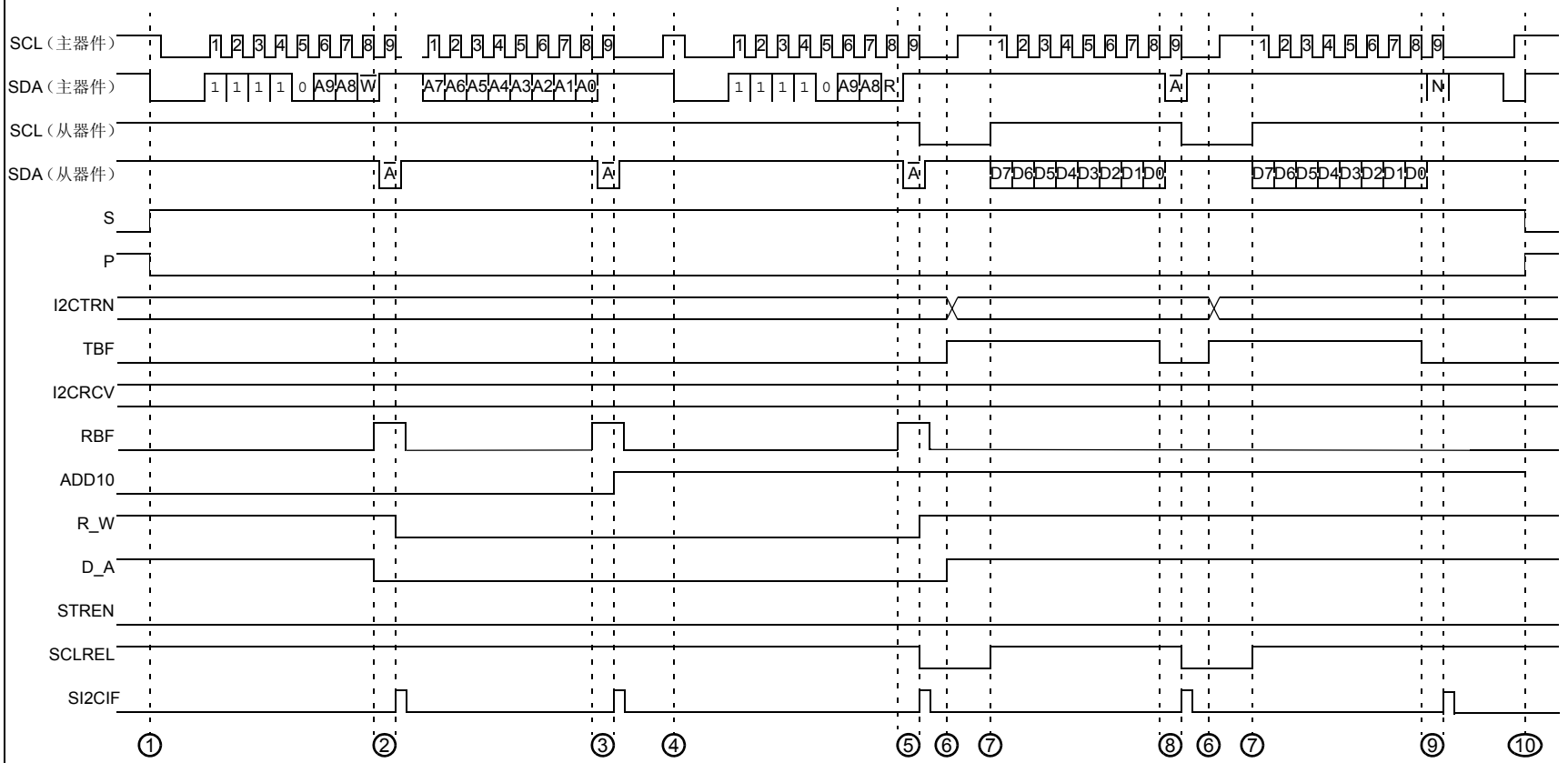
⑤ - 接收到最后一位之后，模块清零 TBF 位，指示缓冲区可用于接收下一字节。

⑥ - 在第 9 个时钟结束时，如果主器件发送 $\overline{\text{ACK}}$，模块将清零 SCLREL 以暂挂时钟。从器件产生中断。

⑦ - 在第 9 个时钟结束时，如果主器件发送 NACK，则说明不再需要发送数据。模块不会暂挂时钟，并将产生中断。

⑧ - 从器件识别停止事件，S 和 P 位相应地置 1/ 清零。

图 19-31： 从器件报文（自从器件读数据：10 位地址）



① - 从器件识别启动事件，S 和 P 位相应地置 1/ 清零。

② - 从器件接收第一个地址字节。指示执行写操作。从器件应答并产生中断。用户软件读 I2CRCV 寄存器。

③ - 从器件接收地址字节。地址匹配。从器件应答并产生中断。用户软件读 I2CRCV 寄存器。

④ - 主器件发送重复启动条件，以更改报文方向。

⑤ - 从器件接收重新发送的第一个地址字节。用户软件读 I2CRCV 寄存器。指示执行写操作。从器件暂挂时钟。

⑥ - 软件向 I2CTRN 中写入响应数据。

⑦ - 软件将 SCLREL 位置 1 以释放时钟保持。主器件继续发送时钟，从器件发送数据字节。

⑧ - 在第 9 个时钟结束时，如果主器件发送 $\overline{\text{ACK}}$，模块将清零 SCLREL 以暂挂时钟。从器件产生中断。

⑨ - 在第 9 个时钟结束时，如果主器件发送 NACK，则说明不再需要发送数据。模块不会暂挂时钟或产生中断。

⑩ - 从器件识别停止事件，S 和 P 位相应地置 1/ 清零。

19.8 I²C 总线的连接注意事项

因为 I²C 总线定义为按线与的方式进行连接，所以在总线上需要接有上拉电阻，该电阻为图 19-32 中的 RP。串联电阻（RS）是可选的，用于提高抗 ESD 能力。电阻 RP 和 RS 的阻值取决于以下参数：

- 供电电压
- 总线电容
- 所连接器件的数量 （输入电流 + 泄漏电流）
- 输入电平选择 （I²C 或 SMBus）

因为器件必须在电阻 RP 存在的情况下将总线下拉为低电平，所以，流过 RP 的电流必须大于器件输出级 I/O 引脚最小灌电流 （VOLMAX = 0.4V 时 IOL = 6.6 mA）。例如，电源电压为 VDD = 3V + 10% 时：

公式 19-2：

$$R_{PMIN} = (V_{DDMAX} - V_{OLMAX})/I_{OL} = (3.3V - 0.6V)/8.5\ mA = 439\Omega$$

在 400 kHz 系统中，最大上升时间规范为 300 ns ；在 100 kHz 系统中，此规范为 1000 ns。因为总电容 CB 的存在，必须选择 RP 以在最大上升时间 300 ns 内将总线电压拉高到（VDD – 0.7V），所以上拉电阻的最大阻抗 （RPMAX）必须小于：

公式 19-3：

$$-t_R/(C_B * (\ln(1 - (V_{DDMAX} - V_{ILMAX})) = -300\ ns/(100\ pF * \ln(1 - (0.99 - 3.3))) \text{ 或 } 2.5\ k\Omega$$

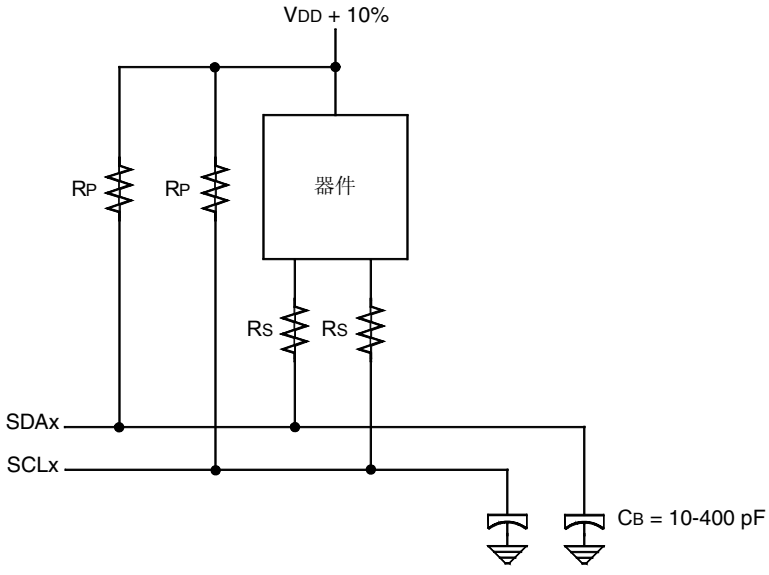
RS 的最大阻值由对应于低电平的期望噪声容限决定。RS 不能降低电压太多，使得器件 VOL 加 RS 上的电压大于最大 VIL。算术表示为：

公式 19-4：

$$R_{SMAX} = (V_{ILMAX} - V_{OLMAX})/I_{OLMAX} = (0.3\ V_{DD} - 0.4)/6.6\ mA = 89\Omega$$

为确保正常工作，SCLx 时钟输入必须满足最小高电平和低电平时间要求。I²C 总线规范的高低电平时间以及对 I²C 模块的具体要求，请参见具体数据手册的 “电气特性”章节。

图 19-32： I²C™ 总线器件配置示例



注： 输入电平与 VDD 相关的 I²C™ 器件必须具有公共的电源线，上拉电阻也连接到该电源线。

19.8.1 集成的信号调理

SCLx 和 SDAx 引脚具有输入毛刺滤波器。I²C 总线在 100 kHz 和 400 kHz 系统中都需要该滤波器。

在 400 kHz 总线中工作时，I²C 总线规范要求对器件引脚输出进行斜率控制。该斜率控制集成在器件中。如果 DISSLW 位（I2CxCON<9>）清零，则斜率控制处于有效状态。对于其他总线速度，I²C 总线规范不要求斜率控制，DISSLW 位应置 1。

一些实现 I²C 总线的系统需要 VILMAX 和 VIHMIN 具有不同的输入电平。在一般的 I²C 系统中，VILMAX 为 0.3 VDD，VIHMIN 为 0.7 VDD。与之不同，在 SMBus（系统管理总线）系统中，VILMAX 设为 0.8V，而 VIHMIN 设为 2.1V。

SMEN 位（I2CxCON<8>）用于控制输入电平。将 SMEN 置 1（= 1）会将输入电平更改为符合 SMBus 规范。

19.9 PWRSAV 指令期间的模块操作

19.9.1 从模式下的休眠模式

在模块配置为从模式时，如果进入休眠模式，模块将继续完全运行。由于所有数据位都是根据外部 SCL 信号而移入 / 移出，因此所有数据发送和接收操作都将继续。然后，模块应根据需要产生异步中断。

19.9.2 主模式下的休眠模式

如果在主器件发送过程中发生休眠，则状态机在时钟停止时部分进入发送状态，模块的行为将是未定义的。类似地，如果在主器件接收过程中发生休眠，则模块的行为也将是未定义的。在主模式下发生休眠时，发送器和接收器将停止。进入休眠模式或退出休眠模式不会影响寄存器的内容；在等待发送或接收时，没有任何自动方式可以阻止模块进入休眠模式。用户软件必须将休眠进入与 I²C 操作进行同步，以避免未定义的模块行为。

19.9.3 当器件进入空闲模式时

器件执行 PWRSAV 1 指令后，器件会进入空闲模式。在空闲模式下，模块将根据 I2CSIDL 位（I2CxCON<13>）进入节能状态。如果 I2CSIDL = 1，则模块进入节能模式，类似于在进入休眠模式时的操作。如果 I2CSIDL = 0，则模块不会进入节能模式，将继续正常工作。

19.10 外设模块禁止（PMD）寄存器

外设模块禁止（PMDx）寄存器提供了一种方法，可通过停止模块的所有时钟源来禁止 I²C 模块。通过相应的 PMDx 控制位禁止某个外设时，外设将处于最低功耗状态。与外设相关的控制和状态寄存器也会被禁止，因此写入这些寄存器不起作用，且读取值无效。只有 PMDx 寄存器中的 I2CxMD 位清零时，才会使能外设模块。

19.11 复位的影响

复位会禁止 I²C 模块并终止所有正在进行和待执行的报文活动。请参见 I2CxCON 和 I2CxSTAT 的寄存器定义，了解这些寄存器的复位条件。

注： 在本文中，“空闲”（Idle）指 CPU 节能状态。小写形式的 “空闲”（idle）指 I²C 模块不在总线上传输数据的时候。

19.12 寄存器映射

表 19-5 中提供了与 PIC24H I^2C 模块相关的寄存器汇总。

表 19-5： I2Cx 寄存器映射

寄存器名称	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
I2CxRCV	—	—	—	—	—	—	—	—	接收寄存器								0000
I2CxTRN	—	—	—	—	—	—	—	—	发送寄存器								00FF
I2CxBRG	—	—	—	—	—	—	—	波特率发生器									0000
I2CxCON	I2CEN	—	I2CSIDL	SCLREL	IPMIEN	A10M	DISSLW	SMEN	GCEN	STREN	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN	1000
I2CxSTAT	ACKSTAT	TRSTAT	—	—	—	BCL	GCSTAT	ADD10	IWCOL	I2COV	D/$\overline{A}$	P	S	R/$\overline{W}$	RBF	TBF	0000
I2CxADD	—	—	—	—	—	—	地址寄存器										0000
I2CxMSK	—	—	—	—	—	—	地址掩码										0000

图注： — = 未实现，读为 0。复位值以十六进制显示。

19.13 设计技巧

问 1： ***我将器件作为总线主器件工作并发送数据。为什么总是在同一时间产生从器件中断和接收中断？***

答 1： 主器件电路和从器件电路是相互独立的，从模块会从总线上接收主模块发送的事件。

问 2： ***我将器件作为从器件工作，在将数据写入 I2CxTRN 寄存器时，为什么数据未被发送？***

答 2： 在准备发送数据时，从器件会进入自动等待。请确保将 SCLREL 位置 1，以释放 I²C 时钟。

问 3： ***如何确定主模块处于何种状态？***

答 3： 检查 SEN、RSEN、PEN、RCEN、ACKEN 和 TRSTAT 位的值，通过这些位值可以确定主模块的状态。如果所有位的值均为 0，则说明模块处于空闲状态。

问 4： ***器件作为从器件工作时，当 STREN = 0 时接收到一个字节。如果软件无法在接收到下一字节之前处理该字节，软件应执行什么操作？***

答 4： 因为 STREN 为 0，所以在接收字节时，模块不会产生自动等待。但是，软件可以在报文传输期间的任意时刻将 STREN 置 1，然后再清零 SCLREL。这将在下一次同步 SCLx 时钟时产生等待。

问 5： ***我的 I²C 系统是多主机系统。为什么在我尝试发送报文时，报文被损坏？***

答 5： 在多主机系统中，其他主器件可能会导致总线冲突。在主器件的中断服务程序中，检查 BCL 状态位，以确保操作完成且未发生冲突。如果检测到冲突，则必须重新发送报文。

问 6： ***我的 I²C 系统是多主机系统。我应如何确定何时可以开始发送报文？***

答 6： 检查 S 状态位。如果 S = 0，则说明总线处于空闲状态。

问 7： ***我尝试在总线上发送启动条件，然后通过写入 I2CxTRN 寄存器发送一个字节。但字节并未发送。这是什么原因？***

答 7： 您必须等待 I²C 总线上的每个事件完成，然后才能启动下一个事件。在这种情况下，您应查询 SEN 位来确定启动事件何时完成，或者在将数据写入 I2CxTRN 寄存器之前等待主器件 I²C 中断。

19.14 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 PIC24H 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 I²C™ 模块相关的应用笔记包括：

标题	应用笔记编号
Use of the SSP Module in the I ²C™ Multi-Master Environment	AN578
Using the PICmicro® SSP for Slave I²C™ Communication	AN734
Using the PICmicro® MSSP Module for Master I²C™ Communications	AN735
An I²C™ Network Protocol for Environmental Monitoring	AN736

注： 如需获取更多 PIC24H 系列器件的应用笔记和代码示例，请访问 Microchip 网站（www.microchip.com）。

19.15 版本历史

版本 A （2007 年 2 月）

这是本文档的初始版本。

版本 B （2008 年 8 月）

该版本包括以下修正和更新：

- 更新了寄存器 19-2 中 ACKSTAT 位（I2CxSTAT<15>）和 D/$\overline{A}$ 位（I2CxSTAT<5>）的位定义。
- 将公式 19-1 中 I2CBRG 的分母从 1,111,111 更新为 10,000,000。
- 更新了表 19-1 中 I²C 时钟速率的值，删除了表注，而在表格后面增加了一条通用 “注”。
- 更新了**第 19.3 节“控制和状态寄存器”**中最后两段，以说明 I2CxRSR 寄存器中的匹配地址字节到 I2CxRCV 寄存器的移位。
- 更新了**第 19.4 节“使能 I²C 操作”**，以说明当 SEN 位置 1 且数据装入 I2CxTRN 寄存器时，主器件功能被使能。
- 更新了几个章节，以说明从模式下的 NACK 状态。受影响的章节有：
  - **第 19.4.2 节 “I²C 中断”**
  - **第 19.7.5 节 “向主器件发送数据”**
  - 图 19-28 至图 19-31
- IPMIEN 位曾被错误描述为智能*外设*管理接口使能位。所有出现之处都已更新为智能*平台*管理接口位。
- 更新了**第 19.9.2 节 “主模式下的休眠模式”**，以说明在发送时进入休眠模式时发生的情况。
- 更新了图 19-26 至图 19-31 中从器件报文 RBF 状态位的信息。
- 对整篇文档进行了其他少量修正，如语言和格式的更新。

注：