AN-895
应用笔记

One Technology Way • P.O. Box 9106 • Norwood, MA 02062-9106, U.S.A. • Tel: 781.329.4700 • Fax: 781.461.3113 • www.analog.com

ADuC702x MicroConverter® I²C® 接口

作者：Michael Looney

简介

本应用笔记介绍如何利用 ADI 公司的 ADuC702x 系列精密微控制器实现 I²C（内部集成电路）接口的硬件主机和从机。本笔记还包含示例代码，显示主机和从机如何利用 I²C 接口相互通信（参见“串行 EEPROM 协议的实现”部分）。

I²C 总线的主要特性如下：

- 只需要两条总线线路：串行数据线 (SDA) 和串行时钟线 (SCL)。两条线路均为双向，即主机和从机均可以用作发射器或接收器。
- 一个 I²C 主机可以与多个从机通信。每个从机都有一个唯一的 7 位地址，因此即使在多从机环境中也始终存在一对一的主机 / 从机关系。
- 主机和从机的发送与接收速度可高达 400 kbps。
- 片内滤波可抑制 SDA 和 SCL 线上的 50 ns 以下尖峰，保护数据完整性。

I²C 接口的典型框图如图 1 所示。

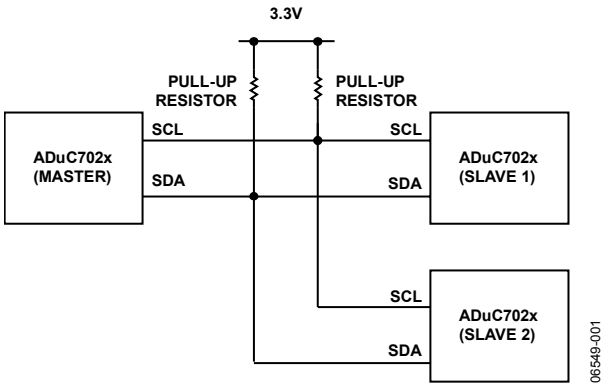

图 1. 单主机、多从机 I²C 框图

目录

I²C 接口概述

I²C 是由 Philips 开发的一种双线串行通信系统，允许多个主机和多个从机通过两条线（SCL 和 SDA）相连。在 I²C 接口中，至少有一个主机和一个从机。SCL 信号控制主机与从机之间的数据传输。

SCL 信号始终是从主机传送到从机。不过，如果从机尚未准备好开始下一次传输，则从机能够将此线拉低。这称为时钟延展功能。每传输一个数据位，就必须产生一个时钟脉冲。

SDA 信号用于发送或接收数据。SDA 输入在 SCL 高电平周期中必须保持稳定。当 SCL 为高电平时，SDA 线的跃迁被视为起始或停止条件（参见图 2 和图 3）。

I²C 基本原理

起始条件

I²C 接口的典型数据传输序列从起始条件开始。起始条件是指在 SCL 线为高电平时，SDA 线上发生的高电平至低电平跃迁（见图 2）。起始条件的产生始终由主机负责。在 SCL 线的高电平周期中，SDA 线只应在发生起始（和停止）条件时改变。在正常数据传输期间（包括从机寻址），SDA 线上的数据在 SCL 线的高电平周期中必须保持稳定。

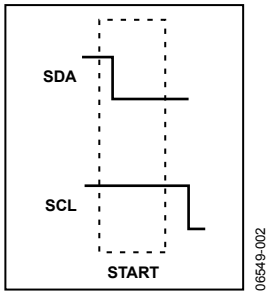

图 2. I²C 的起始条件

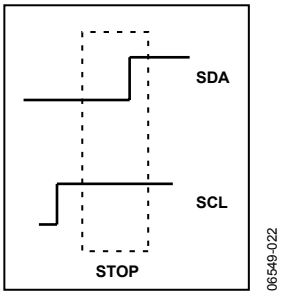

图 3. I²C 的停止条件

从机地址

发生起始条件之后，主机以最高有效位 (MSB) 优先方式在 SDA 线上发送一个字节，同时发送 8 个 SCL 脉冲。此字节的前 7 位为 7 位从机地址。只有当此 7 位地址与从机的地址（或四个从机地址中的一个）匹配时，从机才会响应主机。第 8 位（最低有效位 LSB）为 R/$\overline{W}$ 状态位，决定消息的方向。如果此位为 0，则主机对选定的从机写入数据。如果此位为 1，则主机期待接收来自从机的数据。两种情况下主机都会产生时钟。

如果从机接收到正确的地址，即来自主机的 7 个 MSB 与 I2C0ADR 存储器映射寄存器 (MMR) 的 7 个 MSB 匹配，则从机返回一个有效 ACK，将 SCL 线拉低，并设置 I2C0STA 中的标志位。

从机通过硬件自动完成 I²C 从机寻址的上述所有处理，而主机则负责输出适当的从机地址。

应答 (ACK)/ 非应答 (NACK)

如果从机地址与主机发送的地址匹配，从机会自动发送一个应答 (ACK) 信号，否则会发送一个非应答 (NACK) 信号。ACK 指 SDA 线在第 9 个时钟脉冲时为低电平。NACK 指 SDA 线在第 9 个时钟脉冲时为高电平（见图 4）。

在数据传输期间，ACK 或 NACK 始终由接收器产生。然而，ACK 所需的时钟脉冲始终由主机产生。在 ACK 时钟脉冲期间，发射器必须释放 SDA 线（高电平）。为产生有效的 ACK，接收器必须将 SDA 线拉低。

对于 ADuC702x 微转换器，ACK 和 NACK 均通过硬件在各字节接收结束时自动产生。

如果主机从一个从机接收器接收到 NACK（从机不响应从机地址或所传输的数据），主机应产生停止条件，中止传输（参见“数据传输”部分）。

如果从机为发射器，则在从机发送完最后一个字节之后，主机接收器应产生一个非应答 (NACK) 信号，向从机表明数据序列已结束。从机一旦接收到 NACK，就会释放 SDA 线，以便主机能够产生停止条件。

数据传输

在 I²C 中断服务程序 (ISR) 中，或者在轮询实现中，从机根据主机发送的 R/$\overline{W}$ 位的状态决定是否发送或接收。然后，从机在主机发送的每个时钟脉冲上发送或接收一位。主机负责提供 9 个时钟脉冲（8 个用于数据，1 个用于 ACK），以便从机与主机之间收发数据。从机每发送或接收一个有效数据字节时，I²C 中断位就会置 1。

同样要注意，在从机为发射器、主机为接收器的系统中，当从机发送完最后一个字节之后，主机接收器应发送 NACK，向从机表明数据序列已结束。从机一旦接收到 NACK，就会释放 SDA 线，以便主机能够产生停止条件。

如果一个主机想要中止数据传输，或者中断总线上另一个主机的数据传输，它可以先发送一个起始条件，再发送一个停止条件。

停止条件

数据传输序列由停止条件终止。停止条件是指在 SCL 为高电平时，SDA 线上发生的低电平至高电平跃迁（见图 3）。

停止条件始终由主机产生。主机一旦认为数据序列已结束，或者接收到来自从机的 NACK，就会发送停止条件。接收到停止条件后，从机复位到再次等待从机地址的状态。

ADuC702x 器件的 I²C 接口可以配置为在发生停止条件时产生中断。通过 I2CxCFG MMR 的位 14 可以使能这一特性。

图 5 所示为典型的传输序列。

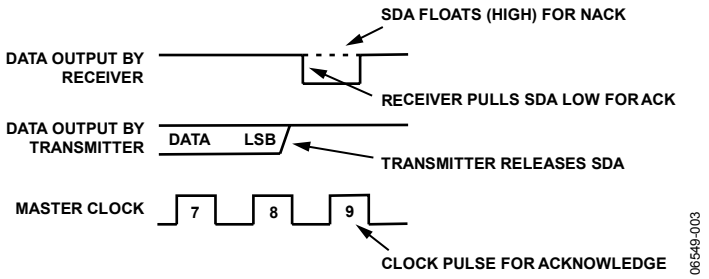

图 4. I²C 总线上的应答 (ACK) 和非应答 (NACK)

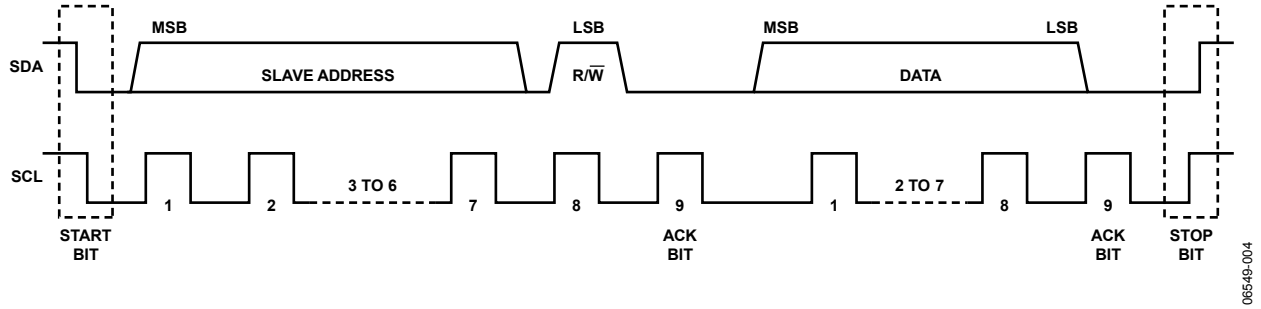

图 5. 典型 I²C 传输序列

重复起始条件

重复起始条件是指向从机发送第二个起始条件，且第一个和第二个起始条件之间没有发送停止条件。它允许主机通过改变 R/$\overline{W}$ 位逆转传输方向，但不必放弃对总线的控制权。

图 6 所示为一个传输序列示例。当发送至器件的第一个数据用于设置要读取的寄存器地址时，通常可以使用该传输序列。

当接收到（重复起始条件 + 从机地址）时，会产生一个中断。使用 I2CxSSTA MMR 的状态位可以区分是（重复起始条件 + 从机地址）时还是（起始条件 + 从机地址）。

注意，ADuC702x 器件在主机模式下无法直接产生 I²C 重复起始序列。

如果用户拥有对整个 I²C 总线的完全控制权，则可以使用一个变通办法，即在所需的第二个起始条件之前禁止产生停止条件。具体步骤如下：

1. 在 SDA 线上放置一个阻值较低的上拉电阻。例如，让 SDA 通过一个 4.7 kΩ 电阻连接到 V_{CC}，让 SCL 通过一个 20 kΩ 电阻连接到 V_{CC}。
2. 添加下列代码序列：

```
I2C0CCNT = 0x0;     // Sets Start/Stop condition counter value to 0 - minimum value.
I2C0ADR = 0xA0;     // Write sequence
I2C0MTX = 0x7;      // Load the Tx FIFO
while ((I2C0FSTA & 0x30) != 0x00) {}  // Wait for the Tx FIFO to empty
I2C0CNT = 0x1;      // Read 2 bytes from the slave
I2C0ADR = 0xA1;     // Send out the Read condition
I2C0CCNT = 0x80;    // Set the Start/Stop counter to a nonzero value to re-enable the Stop
                    Condition
```

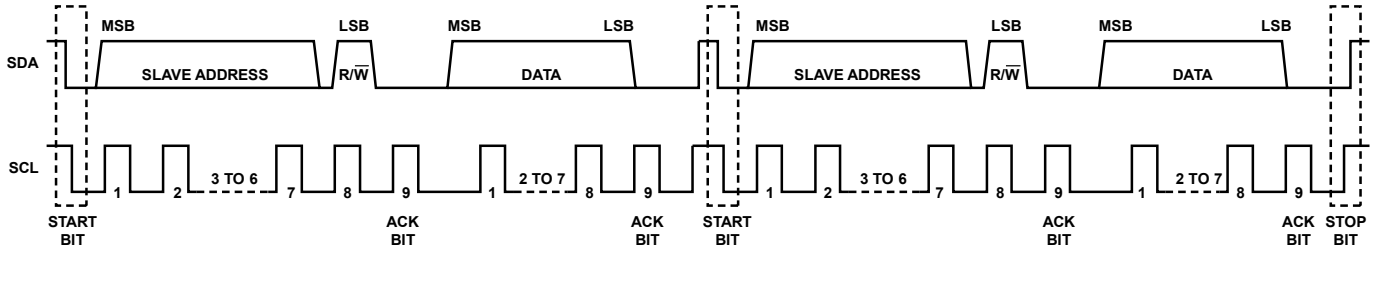

图 6. I²C 重复起始序列

时钟延展

在 I²C 通信中，主机决定时钟速度。与 RS232 不同，I²C 总线提供明确的时钟信号，因此主机和从机无需精确同步到预定的波特率。

但在某些情况下，I²C 从机无法以主机给出的时钟速度工作，需要放慢速度。这是通过一种称为“时钟延展”的机制来实现的。

如果 I²C 从机需要降低总线速度，它可以限制时钟速度。另一方面，主机在释放时钟线以使其回到高电平状态后，需要回读时钟信号，直到时钟线确实已变为高电平。

时钟延展功能通过 I2CxCFG MMR 的位 11 使能。

串行 EEPROM 协议

ATMEL AT24C 系列串行 EEPROM 支持下列五个命令：

- 随机写入
- 顺序写入
- 当前地址读取
- 随机读取
- 顺序读取

图 7 至图 11 说明了这五个命令。

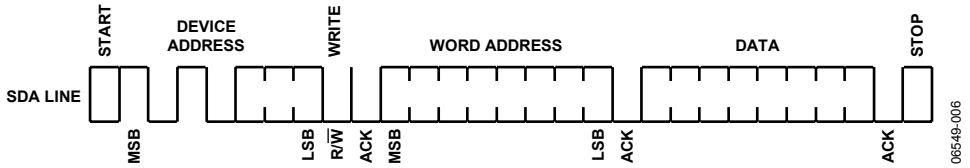

图 7. 随机写入

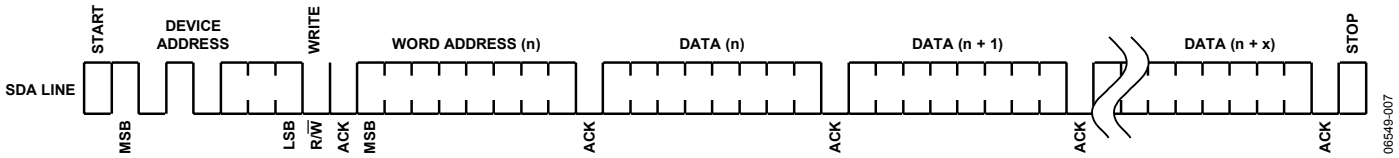

图 8. 顺序写入

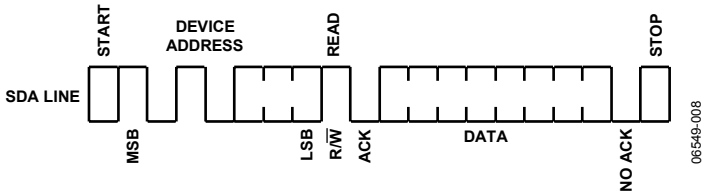

图 9. 当前地址读取

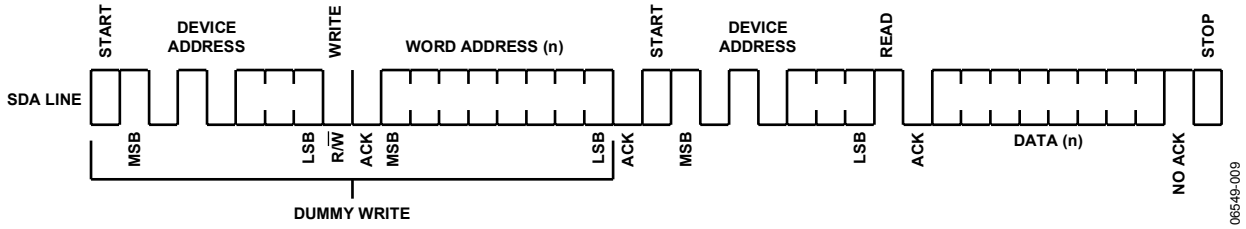

图 10. 随机读取

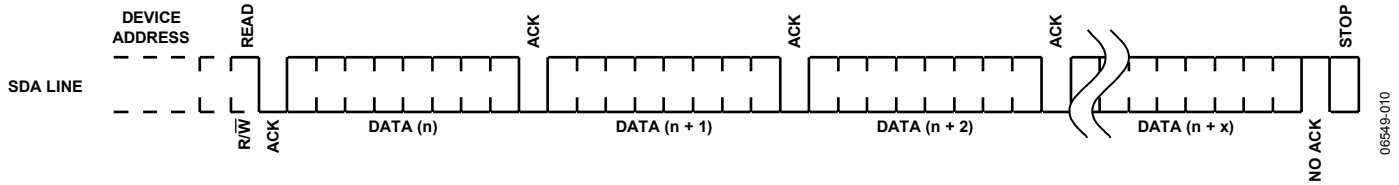

图 11. 顺序读取

ADuC702x 系列微转换器的 I²C 实现

ADuC702x 系列器件内置 2 个全硬件主机和从机 I²C 端口。各端口最多支持 4 个地址，并具有可配置的中断，可实现串行 EEPROM 命令。

在基本层面上，I²C 硬件接口类似于标准 UART。它含有接收和发送缓冲器，各缓冲器均包含 2 字节 FIFO。本应用笔记将详细说明四类通信（主机 / 从机接收 / 发送）以及 FIFO 的使用。

FIFO 的使用

每个 I²C 模块都有四个 2 字节 FIFO：

- 主机接收
- 主机发送
- 从机接收
- 从机发送

以下部分介绍发送 FIFO 和接收 FIFO。

表 1. I2CxFSTA MMR 位功能描述

位号	描述
31 至 10	保留。
9	主机发送 FIFO 清空位。 *置 1*，清空主机发送 FIFO；该位也可以清空从机接收 FIFO。 一旦主机发送 FIFO 被*清空*，则该位会自动清 0。
8	从机发送 FIFO 清空位。 *置 1*，清空从机发送 FIFO； 当从机发送 FIFO 已经被*清空*，该位自动清 0。
7 至 6	主发送 FIFO 状态位。 00 FIFO 空 01 向 FIFO 写字节数据 10 FIFO 中有 1 字节数据 11 FIFO 满
5 至 4	主机发送 FIFO 状态位。 00 FIFO 空 01 向 FIFO 写字节数据 10 FIFO 中有 1 字节数据 11 FIFO 满
3 至 2	从机接收 FIFO 状态位。 00 FIFO 空 01 向 FIFO 写字节数据 10 FIFO 中有 1 字节数据 11 FIFO 满
1 至 0	从机接收 FIFO 状态位。 00 FIFO 空 01 向 FIFO 写字节数据 10 FIFO 中有 1 字节数据 11 FIFO 满

发送 FIFO

要发送数据，必须加载 I2C0STX/I2C0MTX 寄存器。写入一个字节到发送寄存器相当于写入 FIFO 的字节 1（见图 12）。

- 如果字节 0 为空，字节 1 中的字节将被自动推入字节 0。这可以通过状态机来说明（见图 13）。注意，I2C0FSTA 寄存器中的状态对用户是可见的。
- 如果字节 0 已满，该字节将留在字节 1 中。再次写入发送寄存器将覆盖字节 1。

I2CFSTA 寄存器中的发送清空位置 1 将清空 FIFO。

当发生传输时，传输字节 0，字节 1 移入字节 0，FIFO 处于状态 2。

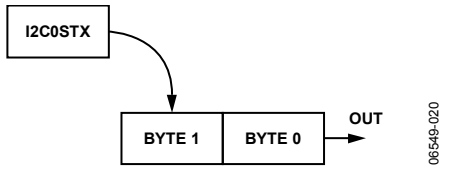

图 12. 发送 FIFO

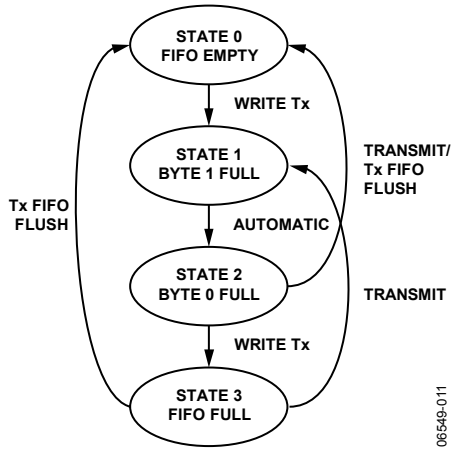

图 13. 发送 FIFO 状态机

接收 FIFO

接收数据时，数据到达字节 0。

- 如果字节 1 为空，字节 0 中的数据将自动移入字节 1。
- 如果字节 1 已满，该数据将留在字节 0 中，直到读取 I2C0SRX（相当于读取字节 1）。
- 如果在 FIFO 已满时有其它数据到达，从机将针对该数据发送一个 NACK，并且 I2C0SSTA 的位 4 置 1。

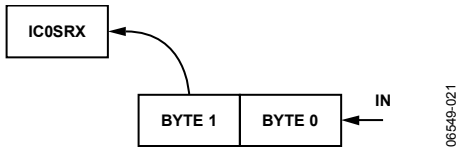

图 14. 接收 FIFO

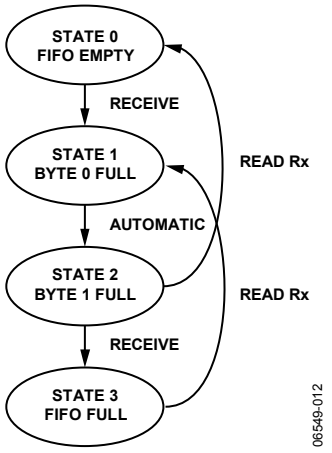

图 15. 接收 FIFO 状态机

主机发送

在发送字节前，必须首先将数据载入发送 FIFO。I2C0ADR 寄存器中必须指定从机的地址。对于数据写入，地址寄存器中的写操作 ($\overline{W}$) 位必须置 0。写入 I2C0ADR 寄存器会自动产生一个起始条件。

每个发送字节的第一个时钟脉冲上会产生一个 I^2C 中断。I2C0MSTA 的位 2 和位 1 置 1，表示主机刚刚发送一个字节，并且 FIFO 下溢。因此，用户可以向 FIFO 添加一个字节。

如果启动传输时 FIFO 中只有一个字节，则第一个 I^2C 中断发生在所发送地址的第一个时钟脉冲上。如果 FIFO 中有两个字节，则中断发生在所发送的第一个字节的第一个时钟脉冲上。

当 FIFO 为空时，传输结束。停止条件在发送完最后一个字节后再过 5.1 µs 时自动产生。

图 16 的流程图显示了这种操作的一个简单示例。

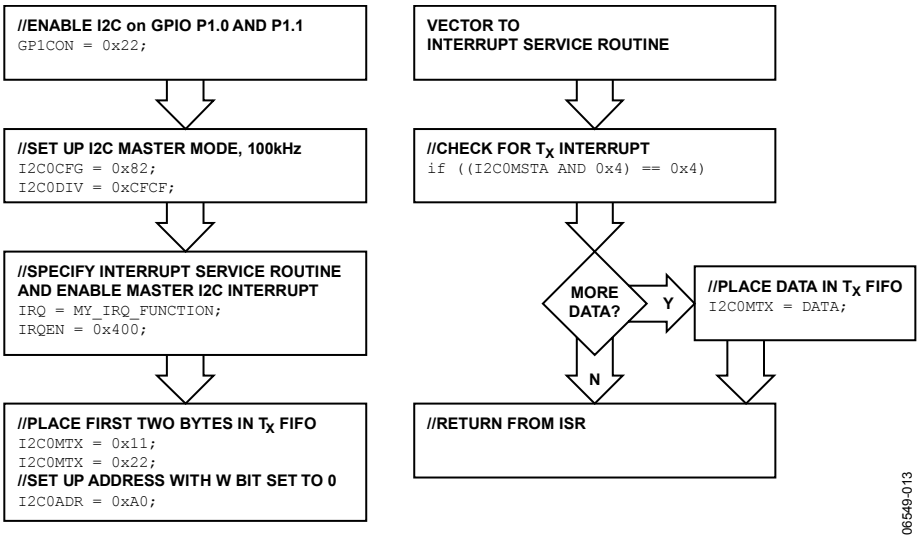

图 16. 主机发送流程图

从机接收

在 I²C 从机接收数据的过程中，当每个字节数据被置于接收 FIFO 中时，即在接收到每个字节的第 9 个时钟脉冲之后，就会产生一个中断。如果在接收第三个字节之前没有读取 FIFO，则该接口针对所发送的最后一个数据自动提供一个 NACK，并且 I2CSSTA 寄存器的位 4 置 1，表示接收 FIFO 溢出。

要从 FIFO 读取数据，应使用 I2C0RX 寄存器。I2C0SSTA 寄存器的位 3 表示从机已接收到数据。只有读取 I2C0SRX 才能使此位清零。清空 FIFO 并不能使位 3 清零。

发送完最后一个数据之后，主机自动发送一个停止条件。

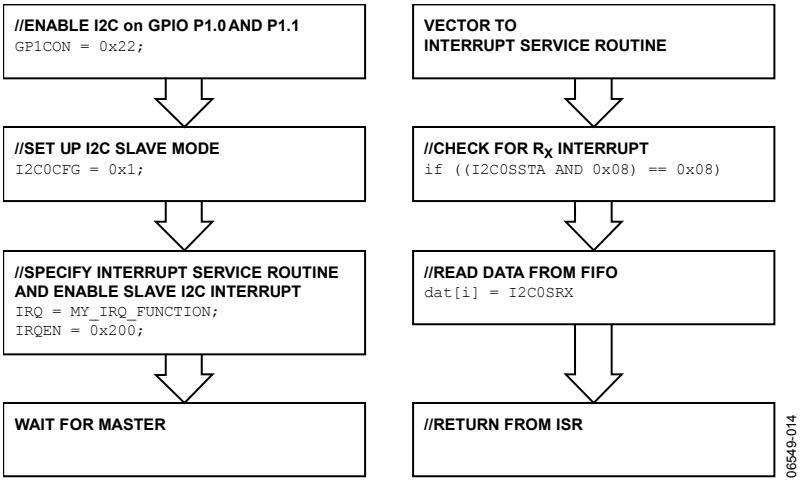

图 17. 从机接收流程图

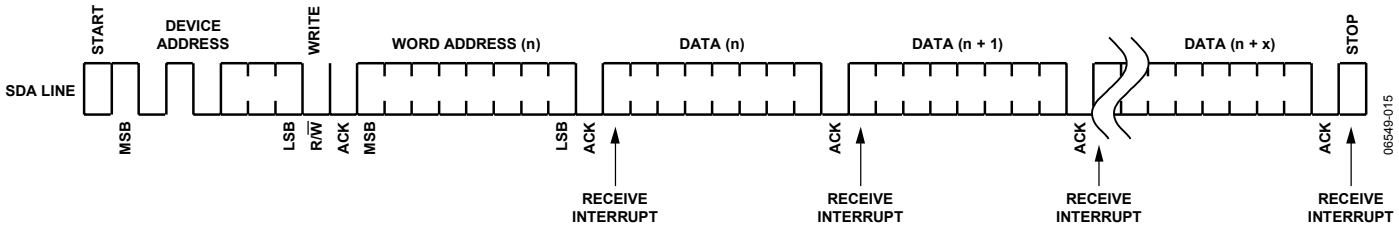

图 18. 从机接收示例

主机接收

在主机模式下，要从从机读取数据，可以使用类似的方法。首先，通过 I2C0CNT 寄存器配置要读取的字节数。其值表示要从从机读取的字节数加 1。它可以是 0 至 7 范围内的值，但在代码执行期间可以复位，以便读取大量数据。

为了开始接收数据，I2C0ADR 寄存器中的读操作 (R) 位应置 1。这将启动传输，并利用 I2C0ADR 寄存器所设置的地址和 R/$\overline{W}$ 位产生起始条件。接收到每个字节后（第 9 个时钟脉冲 ACK 或 NACK 之后），会产生一个中断。I2C0MSTA 的位 3 置 1，表示刚刚接收到一个字节。只有读取 I2C0MRX 才能使此位清零。

当主机不需要接收更多数据时，它针对最后接收到的字节自动产生 NACK。这相当于通知从机停止传输字节，以便主机能够产生停止条件。

如果没有及时读取所接收到的数据，并且 FIFO 已满，则主机将针对所接收到的额外数据提供一个 NACK。

图 19 显示了从从机接收四个字节的流程图。

I2C0CNT 是一个内置计数器的加载寄存器，用户无法访问该计数器。它是一个 3 位计数器，意味着主机只能被配置为一次发送 8 个字节。但是，在发送序列期间写入 I2C0ADR 可以重载该内部计数器。

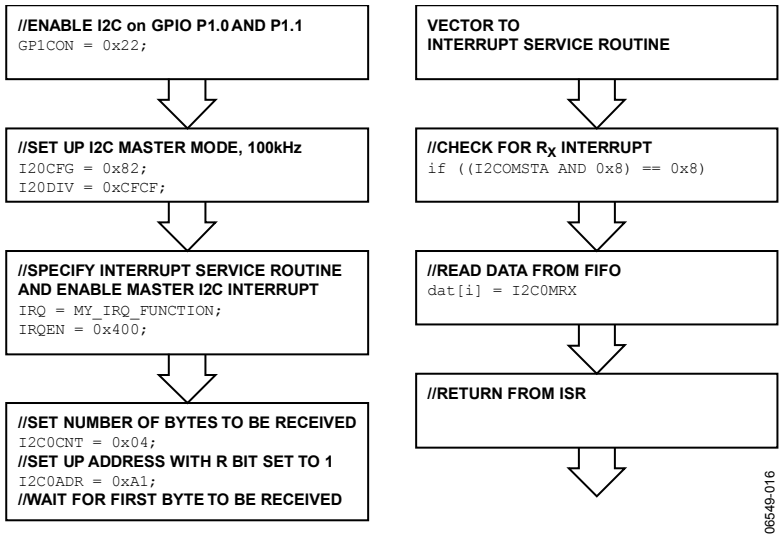

图 19. 主机接收流程图

从机发送

每次请求发送数据时，从机就会产生一个中断，第一个中断发生于对地址作出应答 (ACK) 之后，即在发送 FIFO 的字节 0 时。数据需要预载到从机发送 FIFO 中，否则主机的第一个读取请求将产生 NACK。如果 FIFO 预载有两组数据，则在对地址作出应答 (ACK) 后产生一个中断，然后在对所发送的每个字节作出应答 (ACK) 后产生一个中断。如果 FIFO 仅预载一组数据，则在对地址作出应答 (ACK) 后产生两个中断，发送第一个数据后 FIFO 清空。

发送完一个字节后，只要主机继续请求数据，就会产生一个中断。每次向主机发送一个字节时，I2C0MSTA 的位 2 就会置 1。

图 21 所示为从机响应主机数据请求的示例。

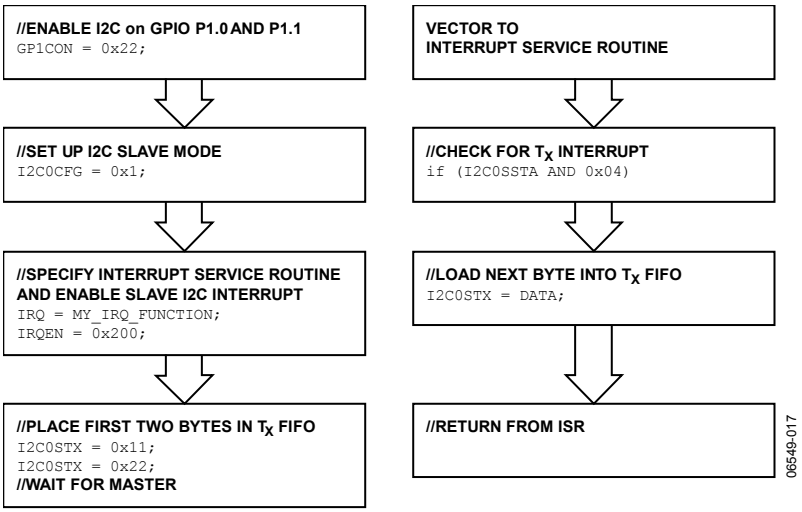

图 20. 从机发送流程图

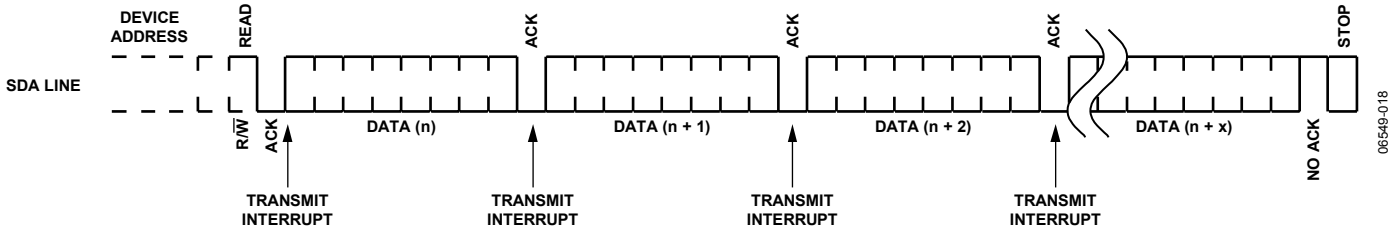

图 21. 从机发送示例

I²C 寄存器定义

I²C 外设接口由 15 个寄存器组成：

- 4 个寄存器，包括发送 / 接收主机 / 从机 MMR（I2CxSRX、I2CxSTX、I2CxMRX、I2CxMTX）。
- 3 个状态寄存器，包括主机 / 从机 /FIFO（I2CxMSTA、I2CxSSTA、I2CxFSTA）。
- 8 个配置寄存器，包括 4 个从机地址、1 个主机地址字节、1 个主机时钟分频器、1 个主机接收数据计数和 1 个 I²C 配置寄存器（I2CxID0、I2CxID1、I2CxID2、I2CxID3、I2CxADR、I2CxDIV、I2CxCNT、I2CxCFG）。
- 用于广播的其它寄存器不在本应用笔记的讨论范围之内。

以下部分将详细介绍其中的 4 个寄存器：

- I2CxDIV，时钟分频寄存器。
- I2CxMSTA，主机状态寄存器，见表 2。
- I2CxCFG，I²C 配置寄存器，见表 3。
- I2CxSSTA，从机状态寄存器，见表 4。

I2CxMSTA：主机状态寄存器

表 2. I2CxMSTA MMR 位功能描述

位号	描述
7	主机发送 FIFO 清空位。 *置 1*，清空主机发送 FIFO；该位也可以清空从机接收 FIFO。 一旦主机发送 FIFO 被*清空*，则该位会自动清 0。
6	主机忙。 如果主机忙，该位自动*置 1*。 该位自动*清 0*。
5	仲裁失效。 在多主机模式下，如果另一个主机占用总线，该位*置 1*。 如果总线空闲，则自动*清 0*。
4	非应答 NACK。 如果从机没有地址应答，该位自动*置 1*。 读 I2C0MSTA 寄存器后，自动*清 0*。
3	主机接收中断请求。 从机接收数据后，该位*置 1*。 读 I2C0MRX 寄存器后，自动*清 0*。
2	主机发送中断请求。 在一次发送结束时，该位*置 1*。 向 I2C0MTX 寄存器写入数据后，自动*清 0*。
1	主机发送 FIFO 下溢。 如果主机发送 FIFO 下溢，该位自动*置 1*。 向 I2C0MTX 寄存器写入数据后，自动*清 0*。
0	主机发送 FIFO 空。 如果主机发送 FIFO 为空，该位自动*置 1*。 向 I2C0MTX 寄存器写入数据后，自动*清 0*。

I2CxDIV，时钟分频寄存器

这是一个 16 位寄存器，包含两个 8 位值：DIVH 和 DIVL。此寄存器中的值可设置 I²C 总线的速度。设置公式如下：

$$f_{serialclock} = \frac{f_{UCLK}}{(2 + DIVH) + (2 + DIVL)}$$

其中：

f_{UCLK} 是分频之前的时钟。
DIVH 是时钟高电平周期。
DIVL 是时钟低电平周期。

因而，如果希望串行时钟为 100 kHz，那么

DIVH = DIVL =0xCF (I2C0DIV = 0xCFCF)

如果希望串行时钟为 400 kHz，那么

DIVH = 0x28 DIVL = 0x3C (I2C0DIV = 0x283C)

I2CxCFG：I²C 配置寄存器

表 3. I2CxCFG MMR 位功能描述

位号	描述
31 至 15	保留。这些位应该由用户写入 0。
14	停止中断使能位。 *置 1*，若在接收到一个有效的起始条件以及地址匹配之后接收到一个停止条件，则产生中断。 *清 0*，在接收到一个停止条件时不产生中断。
13 至 12	保留。这些位应该由用户写入 0。
11	时钟延展使能位 (SCL 保持低速率)。 *置 1*，使能 SCL 线时钟延展。如果 SCL 已经处于低速率或者即将进入低速率，此位指示 I²C 接口使 SCL 保持低速率。 *清 0* 则禁用 SCL 线时钟延展。
10	保留。此位应该由用户写入 0。
9	从机发送 FIFO 中断请求使能位。 *置 1*，禁用从机发送 FIFO 中断请求。 *清 0*，在 R/$\overline{W}$ 位的时钟负脉冲后产生一个中断请求。如果从机发送 FIFO 空时，用户可以向其中写入数据。在时钟速率为 400 kbps 并且内核时钟频率为 41.78 MHz 时，将中断延迟时间计算在内，用户有 45 个时钟周期的时间操作。
8	广播状态位清 0。 *置 1*，清 0 广播状态位。 在广播状态位*清 0* 后由硬件自动清 0。
7	主机串行时钟使能位。 *置 1*，在主机模式使能串行时钟。 *清 0*，在主机模式禁用串行时钟。
6	回送使能位。 *置 1*，内部发送端连接到内部接收端，用于测试用户软件。 *清 0*，正常工作。
5	启动延时取消位。 *置 1*，在多主机模式下，如果仲裁失效，主机立即尝试再次发送； *清 0*，启动延时。在仲裁失效后，主机在尝试再次发送数据前等待。
4	硬件广播使能位。 当该位和广播使能位置 1 时，如果已接收到一个广播信号（地址为 0x00）和 1 字节数据，器件将对接收寄存器中的数据和 I2C0ALT 中的数据进行比较。如果数据匹配，表明器件接收到一个硬件广播。当器件需要紧急呼叫一个主机而又不知道呼叫哪一个时，可使用该功能。ADuC702x 可监视这些地址。要求主机注意的器件会将自己的地址嵌入到消息中。所有的主机都会侦听这些消息，知道如何处理该器件要求的主机会与其从机通信并进行相应操作。根据 2000 年 1 月的 *I²C 总线规范* 2.1 版，I2C0ALT 寄存器的 LSB 应该始终写入 1。
3	广播使能位。 通过将该位*置 1*，可以让从机为 I²C 广播发送 ACK，写地址 0x00。然后器件将识别一个数据位。如果器件接收到的数据是 0x06，即由硬件复位和对从机地址的可编程部分进行写操作，那么根据 *I²C 总线规范*，此时 I²C 接口复位。这个命令可用于复位整个 I²C 系统。当产生任一广播时，广播中断状态位置 1。在复位后用户必须通过设置 I²C 接口进行合适的操作。如果接收到的数据为 0x04，即由硬件对从机地址可编程部分进行写操作，则当产生任一广播时，广播中断状态位置 1。用户必须通过重新编程器件地址来进行合适的操作。
2	保留。
1	主机使能位。 该位*置 1*，使能主机 I²C 通道。 该位*清 0*，禁用主机 I²C 通道。
0	从机使能位。 该位*置 1*，使能从机 I²C 通道。此时会监视从机传输序列中的数据以寻找存放在 I2C0ID0、I2C0ID1、I2C0ID2 和 I2C0ID3 中的器件地址。如果识别了器件地址，就会加入从机传输序列中。 该位*清 0*，禁用从机 I²C 通道。

I2CxSSTA：从机状态寄存器

注意：读取状态寄存器会更改其内容。只能在 ISR 期间读取一次并将其值保存在一个变量中。

表 4. I2CxSSTA MMR 位功能描述

位号	描述
31 至 15	保留。这些位应该由用户写入 0。
14	起始解码位。 如果器件接收到一个有效的起始条件 + 地址匹配，则由硬件对该位*置 1*。 当产生一个 I^2C 停止条件或 I^2C 广播复位时，该位*清 0*。
13	重复起始解码位。 如果器件接收到一个有效的重复起始条件 + 地址匹配，则由硬件对该位*置 1*。 当产生一个 I^2C 停止条件、读取 I2CxSSTA 寄存器或者 I^2C 广播复位时，该位*清 0*。
12 至 11	ID 解码位。 00 接收到的地址匹配 ID 寄存器 0。 01 接收到的地址匹配 ID 寄存器 1。 10 接收到的地址匹配 ID 寄存器 2。 11 接收到的地址匹配 ID 寄存器 3。
10	起始和地址匹配中断后停止。 在上一个 I^2C 起始条件 + 地址匹配后，如果从机接收到一个 I^2C 停止条件，则由硬件对该位*置 1*。 读取 I2CxSSTA 寄存器后该位*清 0*。
9 至 8	广播 ID。 00 无广播 01 广播复位和程序地址 10 广播程序地址 11 广播匹配可供选择的 ID
7	广播中断。
6	从机忙。 如果从机忙，该位*置 1*。 该位自动*清 0*。
5	非应答 NACK。 主机需要数据却无法获得数据时，该位*置 1*。 该位自动*清 0*。
4	从机接收 FIFO 溢出。 如果从机接收 FIFO 溢出，该位*置 1*。 读取 I2C0SRX 后，该位自动*清 0*。
3	从机接收中断请求。 从机接收数据后该位*置 1*。 读取 I2C0SRX 寄存器后自动*清 0*。
2	从机发送中断请求。 在一次发送结束时该位*置 1*。 向 I2C0STX 寄存器写入数据后，该位自动*清 0*。
1	从机发送 FIFO 下溢。 如果从机发送 FIFO 下溢该位*置 1*。 向 I2C0STX 寄存器写入数据后，该位自动*清 0*。
0	从机发送 FIFO 空。 如果从机发送 FIFO 为空，该位*置 1*。 向 I2C0STX 寄存器写入数据后，该位自动*清 0*。

串行 EEPROM 协议的实现

本节说明对于单一 I²C 地址，如何实现串行 EEPROM 规范所支持的五个命令：

- 当前地址读取
- 随机读取
- 随机写入
- 顺序读取
- 顺序写入

在中断服务程序中，必须读取一次状态寄存器并保存其值。中断服务程序的流程图如图 22 所示。

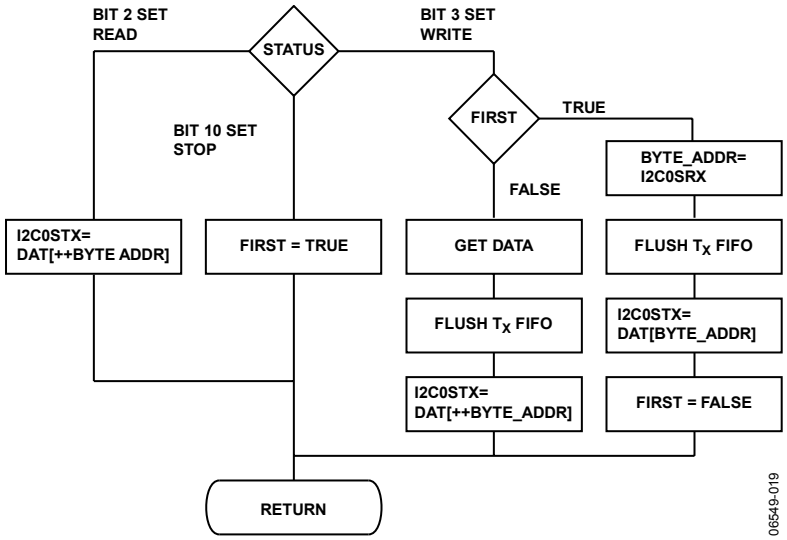

图 22. 从机中断服务程序

随附代码参见 AN-895 Companion Code.zip。

命令代码

I²C 配置

```
I2CCFG = 0x4001;   // Enable slave, enable STOP detect
I2C0ID0 = 0xA0;    // Slave ID
I2C0STX = dat[0];  // Set initial data
```

变量初始化

```
Byte_addr = 0
First = 1
```

使能 I²C 从机中断

```
IRQEN = 0x200;     // I2C0 Slave Interrupt
while (1)    // Wait for interrupt
```

如果系统符合 Philips 公司定义的 I²C 标准规范，则用户在购买 ADI 公司或其下属机构拥有 Philips 公司许可的 I²C 器件时，可以获得 Philips 公司 I²C 专利权之下的许可，以便在 I²C 系统中使用这些器件。

©2007 Analog Devices, Inc. All rights reserved. Trademarks and registered trademarks are the property of their respective owners.
AN06549-0-1/07(0)