

# 19 I<sup>2</sup>C

## 19.1 概述

I<sup>2</sup>C 模块实现 MCU 与外部 I<sup>2</sup>C 接口器件之间的同步通信，硬件实现串并转换。支持 I<sup>2</sup>C 的主机和从机模式，不支持多主机模式。

特点：

- 1 路独立 I<sup>2</sup>C 接口
- 支持主机和从机模式，不支持多主机模式
- 支持 7 位或 10 位从机地址
- 传输速度支持 standard mode(100Kbps), fast mode(400Kbps)和 Fm+(1Mbps)
- 支持 DMA，主机和从机独立 DMA 通道
- 低功耗从机设计，可以在没有系统时钟的情况下收发数据
- 支持异步从机地址匹配唤醒、数据帧接收完成唤醒或 START 检测唤醒

## 19.2 结构框图

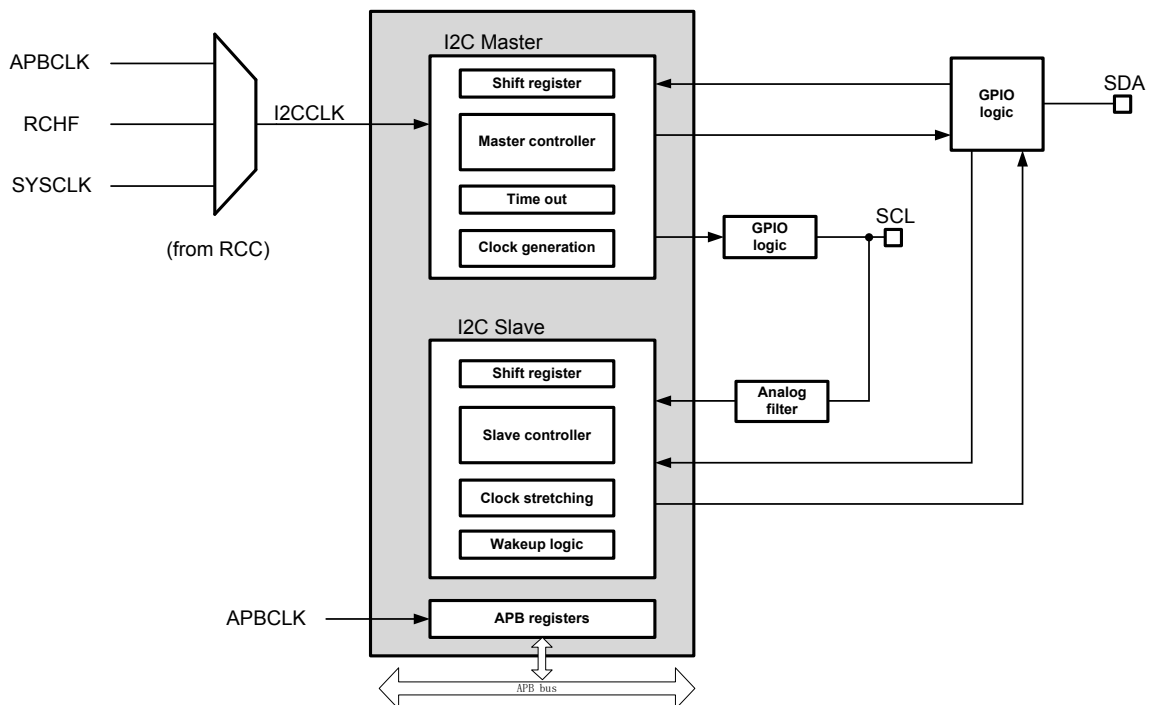


图 19-1 I<sup>2</sup>C 模块框图

### 19.3 引脚定义和上拉电阻范围

I2C 模块使用 2 个真开漏引脚与外部器件通信，工作时需要外接上拉电阻：

引脚	I2Cx	符号	功能
PA11	I2C	SCL	I2C 时钟
PA12		SDA	I2C 数据

表 19-1I2C 引脚列表

I2C 总线协议规定了 standard-mode、fast-mode 和 fast-mode plus 的信号最大上升时间，以及 IO 能够支持的最小 sink 电流。参见下表。

Symbol	Parameter	Conditions	Standard-mode		Fast-mode		Fast-mode Plus		Unit
			Min	Max	Min	Max	Min	Max	
$V_{IL}$	LOW-level input voltage <sup>[1]</sup>		-0.5	$0.3V_{DD}$	-0.5	$0.3V_{DD}$	-0.5	$0.3V_{DD}$	V
$V_{IH}$	HIGH-level input voltage <sup>[1]</sup>		$0.7V_{DD}$	<sup>[2]</sup>	$0.7V_{DD}$	<sup>[2]</sup>	$0.7V_{DD}$ <sup>[1]</sup>	<sup>[2]</sup>	V
$V_{hys}$	hysteresis of Schmitt trigger inputs		-	-	$0.05V_{DD}$	-	$0.05V_{DD}$	-	V
$V_{OL1}$	LOW-level output voltage 1	(open-drain or open-collector) at 3 mA sink current; $V_{DD}>2V$	0	0.4	0	0.4	0	0.4	V
$V_{OL2}$	LOW-level output voltage 2	(open-drain or open-collector) at 2 mA sink current <sup>[3]</sup> ; $V_{DD}\leq 2V$	-	-	0	$0.2V_{DD}$	0	$0.2V_{DD}$	V
$I_{OL}$	LOW-level output current	$V_{OL} = 0.4 V$	3	-	3	-	20	-	mA
		$V_{OL} = 0.6 V$ <sup>[4]</sup>	-	-	6	-	-	-	mA
$t_{of}$	output fall time from $V_{Ihmin}$ to $V_{ILmax}$		-	250 <sup>[5]</sup>	$1 \times (V_{DD} / 5.5 V)$ <sup>[6]</sup>	250 <sup>[5]</sup>	$1 \times (V_{DD} / 5.5 V)$ <sup>[6]</sup>	120 <sup>[7]</sup>	ns
$t_{SP}$	pulse width of spikes that must be suppressed by the input filter		-	-	0	50 <sup>[8]</sup>	0	50 <sup>[8]</sup>	ns
$I_i$	input current each I/O pin	$0.1V_{DD}< V_i< 0.9V_{Ddmax}$	-10	+10	-10 <sup>[9]</sup>	+10 <sup>[9]</sup>	-10 <sup>[9]</sup>	+10 <sup>[9]</sup>	$\mu A$

$C_i$	capacitance for each I/O pin <sup>[10]</sup>	-	10	-	10	-	10	pF
-------	--	---	----	---	----	---	----	----

表 19-2I2C 协议电参数表

下表则定义了总线信号允许的最大上升时间和下降时间。

Symbol	Parameter	Conditions	Standard-mode		Fast-mode		Fast-mode Plus		Unit
			Min	Max	Min	Max	Min	Max	
$f_{SCL}$	SCL clock frequency		0	100	0	400	0	1000	kHz
$t_{HD;STA}$	hold time (repeated) START condition	After this period, the first clock pulse is generated.	4.0	-	0.6	-	0.26	-	$\mu$ s
$t_{LOW}$	LOW period of the SCL clock		4.7	-	1.3	-	0.5	-	$\mu$ s
$t_{HIGH}$	HIGH period of the SCL clock		4.0	-	0.6	-	0.26	-	$\mu$ s
$t_{SU;STA}$	set-up time for a repeated START condition		4.7	-	0.6	-	0.26	-	$\mu$ s
$t_{HD;DAT}$	data hold time <sup>[2]</sup>	CBUS compatible masters (see Remark in <a href="#">Section 4.1</a> )	5.0	-	-	-	-	-	$\mu$ s
		I <sup>2</sup> C-bus devices	0 <sup>[3]</sup>	- <sup>[4]</sup>	0 <sup>[3]</sup>	- <sup>[4]</sup>	0	-	$\mu$ s
$t_{SU;DAT}$	data set-up time		250	-	100 <sup>[5]</sup>	-	50	-	ns
$t_r$	rise time of both SDA and SCL signals		-	1000	20	300	-	120	ns
$t_f$	fall time of both SDA and SCL signals <sup>[3][6][7][8]</sup>		-	300	$1 \times (V_{DD} / 5.5 V)$	300	$1 \times (V_{DD} / 5.5 V)$ <sup>[9]</sup>	120 <sup>[8]</sup>	ns
$t_{SU;STO}$	set-up time for STOP condition		4.0	-	0.6	-	0.26	-	$\mu$ s
$t_{BUF}$	bus free time between a STOP and START condition		4.7	-	1.3	-	0.5	-	$\mu$ s
$C_b$	capacitive load for each bus line <sup>[10]</sup>		-	400	-	400	-	550	pF
$t_{VD;DAT}$	data valid time <sup>[11]</sup>		-	3.45 <sup>[4]</sup>	-	0.9 <sup>[4]</sup>	-	0.45 <sup>[4]</sup>	$\mu$ s
$t_{VD;ACK}$	data valid acknowledge time <sup>[12]</sup>		-	3.45 <sup>[4]</sup>	-	0.9 <sup>[4]</sup>	-	0.45 <sup>[4]</sup>	$\mu$ s
$V_{nL}$	noise margin at the LOW level	for each connected device (including hysteresis)	$0.1V_{DD}$	-	$0.1V_{DD}$	-	$0.1V_{DD}$	-	V
$V_{nH}$	noise margin at the HIGH level	for each connected device (including hysteresis)	$0.2V_{DD}$	-	$0.2V_{DD}$	-	$0.2V_{DD}$	-	V

表 19-3I2C 协议时序参数表

根据以上协议规范，我们可以计算外部上拉电阻的合理范围。

假设总线信号从  $V_{IL}=0.3V_{DD}$  上升到  $V_{IH}=0.7V_{DD}$ ，则充电时间可以计算为：

$$V(t_1) = 0.3 \times V_{DD} = V_{DD} (1 - e^{-t_1 / RC}); \quad t_1 = 0.3566749 \times RC$$

$$V(t_2) = 0.7 \times V_{DD} = V_{DD} (1 - e^{-t_2 / RC}); \quad t_2 = 1.2039729 \times RC$$

$$T = t_2 - t_1 = 0.8473 \times RC$$

根据总线容性负载大小和协议对信号最大上升时间的要求，我们可以计算上拉电阻的最大值：

$$R_{p(max)} = \frac{t_r}{0.8473 \times C_b}$$

而上拉电阻的最小值则由总线电源电压  $V_{DD}$  和 IO 电流 sink 能力决定，FM33LC0XX 的 I2C 引脚 sink 能力是 20mA，而协议要求的最小 sink 能力是标准/快速模式 3mA，Fm+模式 20mA。

$$R_{p(min)} = \frac{V_{DD} - V_{OL(max)}}{I_{OL}}$$

## 19.4 时钟

I2C 主机和从机都采用了双时钟结构：

- 主机和从机的总线寄存器时钟用 PCLK 表示，来源于 APBCLK。当 CPU 或者 DMA 需要访问 I2C 内部寄存器时，必须使能 PCLK。
- 主机的数据收发时钟用 I2CCLK 表示，除了可以来源于 APBCLK，还可以来源于 RCHF、SYSCLK、RCMF，能够独立于 APBCLK 工作。必须使能 I2CCLK 才能进行数据收发。
- 从机的数据收发时钟使用 SCL 总线时钟输入，因此无需系统时钟就可以进行数据收发

PCLK 和 I2CCLK 的控制都在 CMU 模块内完成，进行 I2C 通信前必须正确配置相应的 CMU 控制寄存器。

采用双时钟结构，可以使 I2C 的工作不受限于 APBCLK 的配置，当某些外设需要工作在很高的 APBCLK 频率上时，I2C 仍可以工作在降低的频率上；或者反过来，CPU 工作在较低频率上，也不影响 I2C 以较高的波特率进行数据通信。

理论上 PCLK 和波特率时钟之间没有相对关系的约束，波特率时钟可以快于或者慢于 PCLK。但是应用需要注意当两者频率相差较大时，CPU 或者 DMA 是否来得及进行数据搬运。

# 19.5 接口时序

## 19.5.1 接口时序图

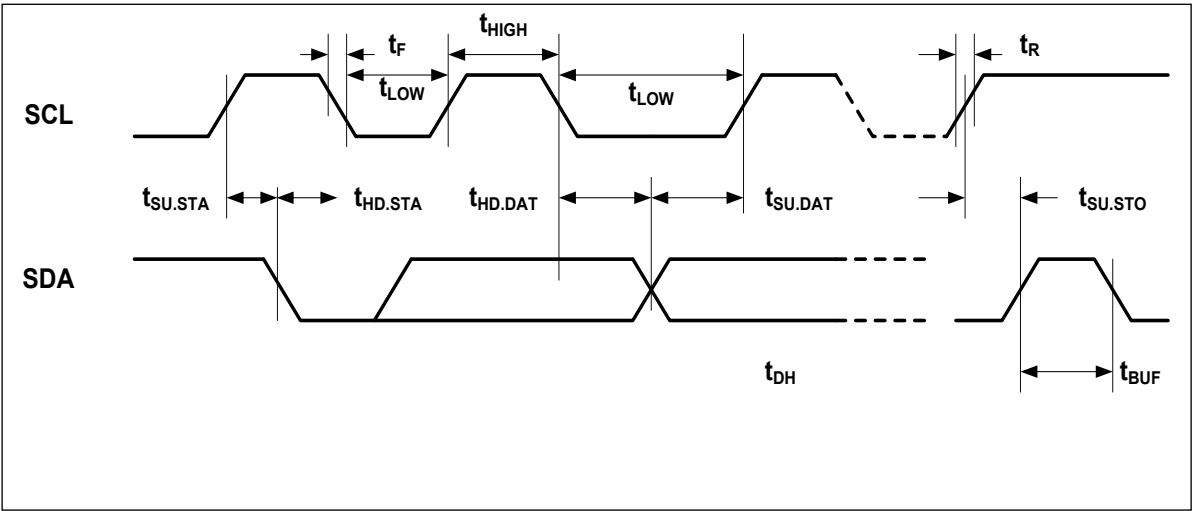


图 19-2 I<sup>2</sup>C 总线时序

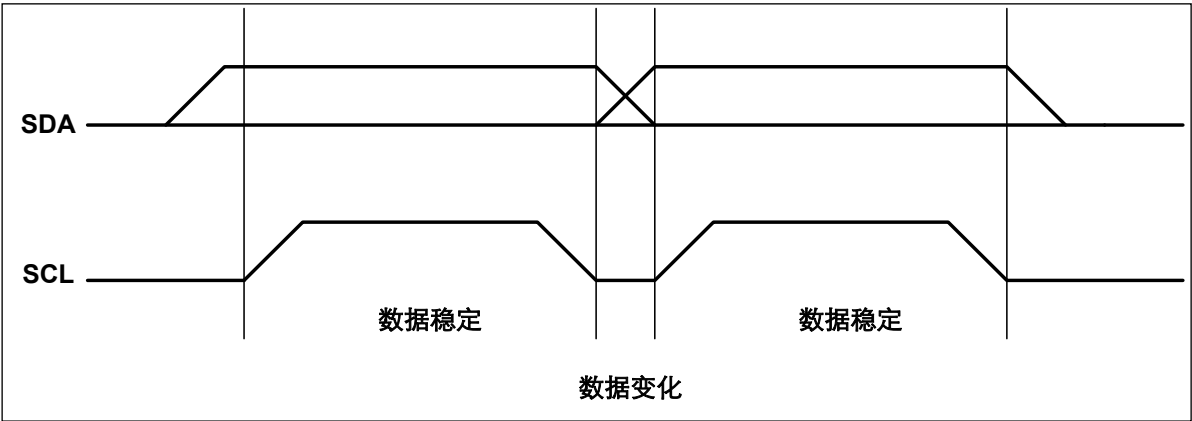


图 19-3 数据有效时序

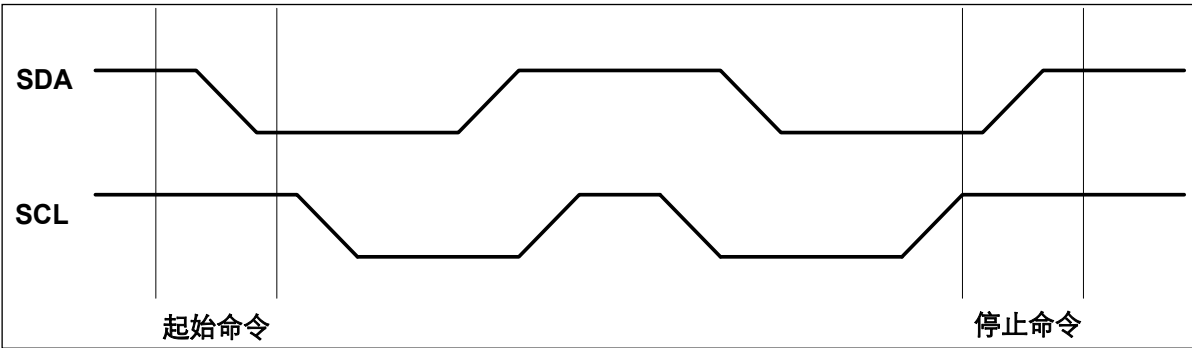


图 19-4 起始 (Start) 与停止(Stop)命令定义

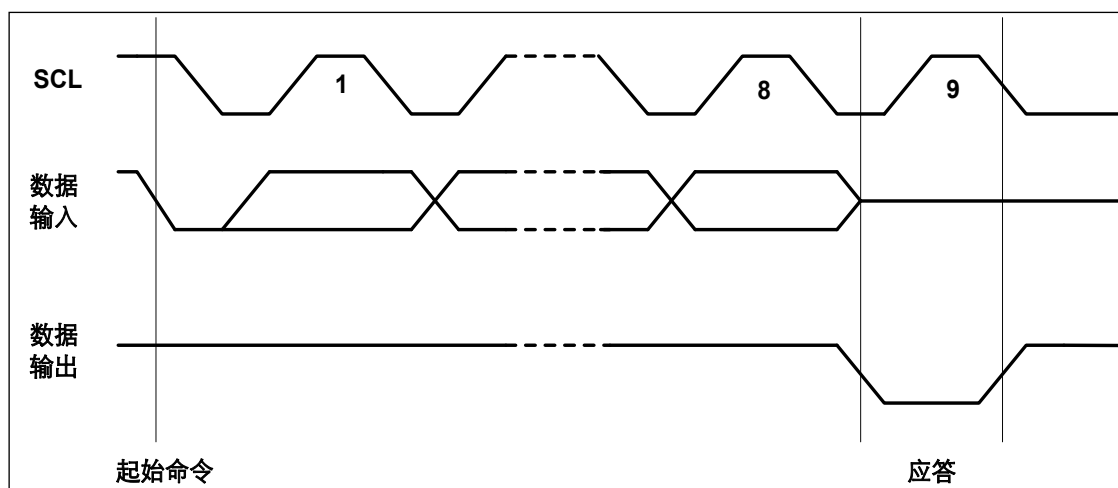


图 19-5 输出应答(ACK)

### 19.5.2 接口时序描述

**时钟有效时序:** SDA 引脚通常被外围器件拉高。SDA 引脚的数据应在 SCL 为低时变化(参见图 19-3); 当数据在 SCL 为高时变化, 将视为下文所述的一个起始或停止命令。

**起始命令:** 当 SCL 为高, SDA 由高到低的变化被视为起始命令, 必须以起始命令作为任何一次读/写操作命令的开始(参见图 19-4)。

**停止命令:** 当 SCL 为高, SDA 由低到高的变化被视为停止命令, 在一个读操作后, 停止命令会使 EEPROM 进入等待态低功耗模式(参见图 19-4)。

**输出应答:** SDA 上的数据都是以 8 位为一组串行输入和输出的, MSB 先发, 接收方在收完每个字节后应当在第 9 个周期回发一个回应 acknowledge 位(以下简称 ack), ack 的时钟由主机提供。发送方在 ack 期间悬空 SDA, 接收方须将 SDA 拉低, 确保 ack 时钟高电平期间 SDA 为低, 形成有效的 ack 信号(参见图 19-5)。

参数	符号	标准模式(100K)		快速模式(400K)		单位
		最小值	最大值	最小值	最大值	
SCL 时钟频率	$F_{SCL}$	0	100	0	400	kHz
启动条件建立时间	$T_{SU:STA}$	4.7	—	0.6	—	us
启动条件保持时间	$T_{HD:STA}$	4.0	—	0.6	—	us
时钟低电平时间	$T_{LOW}$	4.7	—	1.3	—	us
时钟高电平时间	$T_{HIGH}$	4.0	—	0.6	—	us
数据输入建立时间	$T_{SU:DAT}$	250	—	100 <sup>(4)</sup>	—	ns
数据输入保持时间	$T_{HD:DAT}$	5.0	—	—	—	us
		0 <sup>(2)</sup>	3.45 <sup>(3)</sup>	0 <sup>(2)</sup>	0.9 <sup>(3)</sup>	us
SDA 和 SCL 上升时间	$T_R$	—	1000	20+0.1Cb <sup>(5)</sup>	300	ns
SDA 和 SCL 下降时间	$T_F$	—	300	20+0.1Cb <sup>(5)</sup>	300	ns
停止条件建立时间	$T_{SU:STO}$	4.0	—	0.6	—	us
总线空闲时间	$T_{BUF}$	4.7	—	1.3	—	us

参数	符号	标准模式（100K）		快速模式(400K)		单位
		最小值	最大值	最小值	最大值	
总线的容性负载	Cb	—	400	—	400	Pf
噪声容限低值	V <sub>nL</sub>	0.1V <sub>DD</sub>	—	0.1V <sub>DD</sub>	—	V
噪声容限高值	V <sub>nH</sub>	0.2V <sub>DD</sub>	—	0.2V <sub>DD</sub>	—	V

表 19-4 I<sup>2</sup>C 接口时序要求



## 19.6 I<sup>2</sup>C 工作模式

I2C模块支持以下工作模式：

- 主机接收
- 主机发送
- 从机接收
- 从机发送

芯片上电后I2C模块默认关闭，主机和从机都不工作。软件需要根据应用选择模块工作模式，通过设置MSPEN来使能主机通信，或设置SSPEN来使能从机通信。

主机和从机不能同时工作，因为他们复用相同的IO引脚作为SCL和SDA，原则上禁止软件同时将MSPEN和SSPEN置1。

## 19.7 I<sup>2</sup>C 从机地址格式

I<sup>2</sup>C总线协议定义了以下保留地址，对其中多数保留地址，I<sup>2</sup>C从机硬件不做合法性判断，软件可以根据收到的地址进行自定义的处理。

但是对于10bit从机地址应用，即SSPCON.A10EN=1的情况下，要求1st字节必须以11110开头，否则将触发ADDR\_ERROR错误标志。而在SSPCON.A10EN=0的情况下，如果从机收到了11110开头的地址字节，也会置位ADDR\_ERROR错误标志。

从机地址	R/W_bit	描述
0000 000	0	General Call address
0000 000	1	START byte
0000 001	X	CBUS address
0000 010	X	Reserved for different bus format
0000 011	X	Reserved for future purpose
0000 1XX	X	HS-mode master code
1111 1XX	X	Reserved for future purpose
1111 0XX	X	10bit slave addressing

表 19-5I<sup>2</sup>C 从机保留地址定义

## 19.8 I<sup>2</sup>C 初始化

进行I<sup>2</sup>C通信前必须正确的初始化I<sup>2</sup>C模块，建议软件按照以下步骤进行初始化操作：

- 清零RCC模块的I2CRST寄存器，确保I2C模块不处于复位状态
- 置位RCC模块的I2C\_APBEN寄存器，使能I2C模块寄存器总线接口时钟
- 配置RCC模块的I2C\_CKSEL和I2C\_CKEN寄存器，选择并使能I2C工作时钟（如果是从机模式，不需要这个步骤）
- 根据需要配置模拟滤波使能（SCL和SDA输入模拟滤波，>50ns）

### 19.8.1 IO 配置

FM33LC0XX 最多有两组引脚用于数据传输，开始 I<sup>2</sup>C 通信前需将对应引脚的 FCR 寄存器设置为 AF：

SDA: PA12/PD12

SCL: PA11/PB15

注意，如果 PA11 和 PB15 同时配置为 SDA 功能，则 PA11 被连接到 I<sup>2</sup>C 模块，PB15 无效；如果 PA12 和 PD12 同时配置为 SCL 功能，则主机模式下两个引脚都会输出 SCL 信号，从机模式下只有 PA12 被连接到 I<sup>2</sup>C 从机的 SCL 输入。

PA11 和 PA12 是强驱动真 OD 引脚，必须搭配外部总线上拉电阻使用，并且具备 20mA sink 电流能力，能够支持 Fm+ 模式。

### 19.8.2 主机波特率配置

I<sup>2</sup>C 主机需要在使能前配置通信波特率，而从机不需要配置。

MSPBRG[8:0]波特率配置寄存器用于产生通信波特率。MSPBRG 是 9 bit 波特率分频系数，波特率计算公式如下：

$$T_{SCL} = 2T_{BRG}$$

$$T_{BRG} = 2 \times T_{I2CCLK} \times (MSPBRG[8:0] + 1); T_{I2CCLK} \text{ 为 I2C 工作时钟周期，即：}$$

$$MSPBRG = F_{I2CCLK} / (4 \times F_{SCL}) - 1$$

例如对于 100k 波特率，若 I<sup>2</sup>C 工作时钟为 8M，则 MSPBRG=19。

### 19.8.3 从机的输入模拟滤波和输出延迟

模拟滤波功能仅针对 SCL 引脚，并且只有从机的 SCLi 输入信号上可以使能模拟滤波功能。

同时，从机的 SDA 输出延迟，通过在 SDAo 上增加大于 300ns 的模拟延迟，来确保 SDA 相对于 SCL 下降沿的输出保持时间。

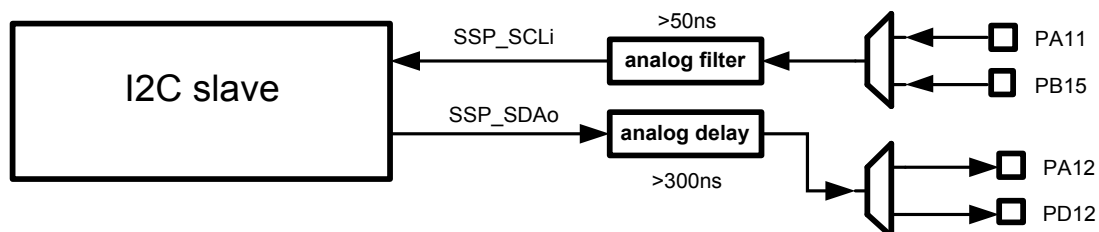


图 19-6 从机信号滤波

19.9 I<sup>2</sup>C 主机功能

FM33LC0XX的I2C主机模式不支持多主机总线，因此挂在总线上的其他设备都是从机。总线上总是由主机提供同步时钟SCL，SDA数据流方向可以是主机发送从机接收，或者从机发送主机接收。

I2C总线通信总是由主机发起，主机模式支持7bit或10bit寻址。

19.9.1 7bit 寻址

在7bit寻址时，主机发送的第一个字节包含从机地址和传输方向位（ $R/\overline{W}$ ），根据 $R/\overline{W}$ 决定后续传输是主机向从机写入数据（ $R/\overline{W}=0$ ）或主机从从机读取数据（ $R/\overline{W}=1$ ）。

名字	Slave Address Byte							
位	7	6	5	4	3	2	1	0
位名	address							R/W

位描述：

位号	助记符	功能描述
7-1	address	Slave device address
0	R/W	0: Write 表示发送数据（master 发送） 1: Read 表示请求数据（slave 回发）

主机向从机写入数据

典型的7bit寻址，主机向从机写入数据的帧结构如下图所示。

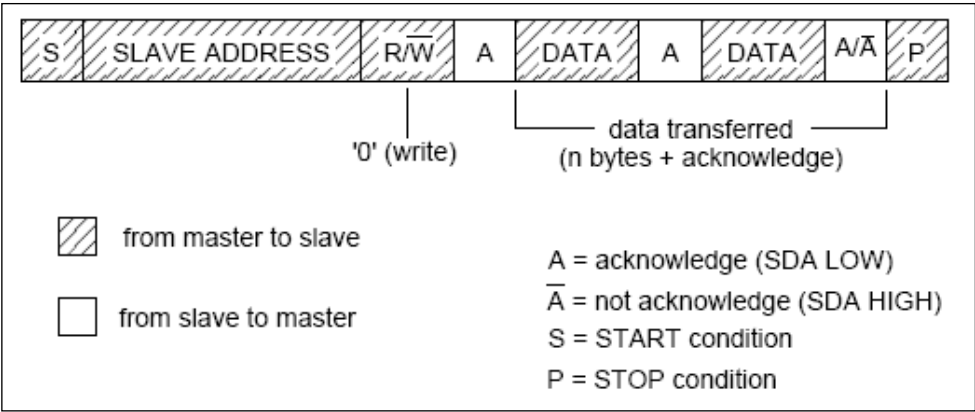


图 19-7 主机向 7 位地址从机写入数据时的帧格式

- 1、主机发起 START 时序
- 2、主机发送从机地址，从机地址包含 7 位从机地址和 1 位 R/W 标志位，发送数据时 R/W 位为 0
- 4、主机发送第一帧 8 位数据

- 5、主机在每次发送完 8 位数据后，会在第 9 个 SCL 判断是否检测到有效的 ACK，如果主机检测到 ACK 成功后，会继续输出下一字节数据
- 6、若从机无法响应 ACK，主机检测到 NACK 后应发送 STOP 时序终止发送
- 7、主机完成所有数据发送后，发送 STOP 时序

软件启动 I2C 主机发送的操作流程如下图：

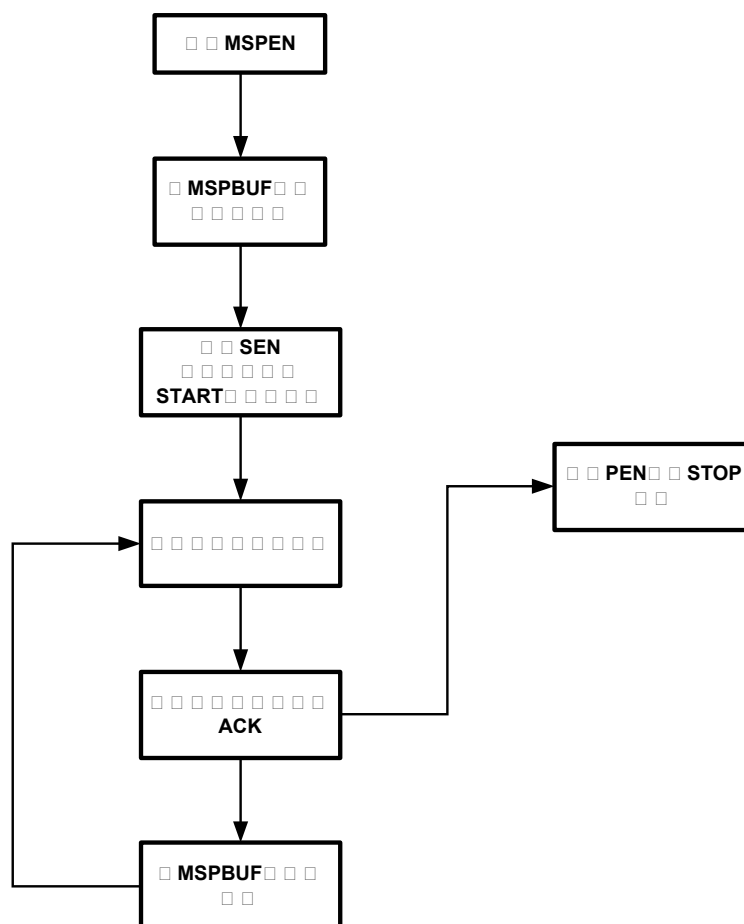


图 19-8I2C 软件发送数据流程图

I2C主机对7位地址从机写入数据的波形示意图如下：

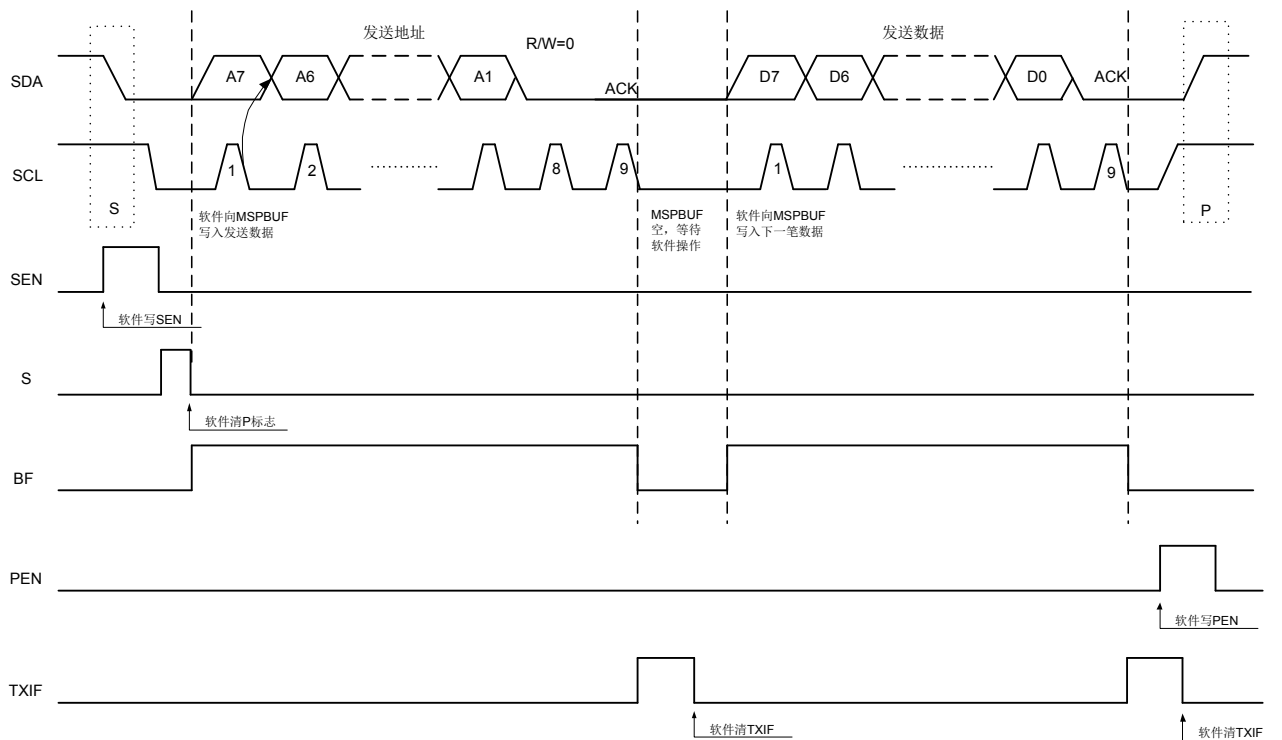


图 19-9I2C 主机对 7 位地址从机发送数据流图

### 主机从从机读取数据

典型的7bit寻址，主机从从机读取数据的帧格式如下图所示。

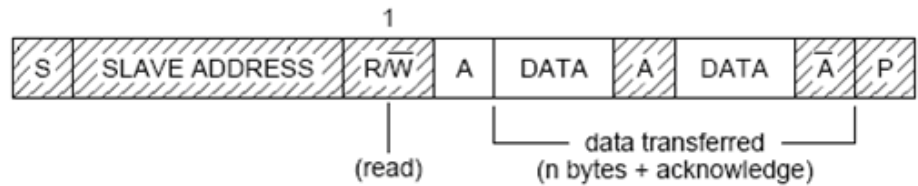


图 19-10 主机从 7 位地址从机读取数据时的帧格式

- 1、 主机发起 START 时序
- 2、 主机发送从机地址，从机地址包含 7 位从机地址和 1 位 R/W 标志位，数据读取时 R/W 位为 1
- 3、 此时设置 MSPCON.RCEN 为 1，主机自动转为接收状态
- 4、 主机开始接收第一字节 8 位数据，并在第 9 个 SCL 向从机发送有效 ACK,从而继续读取下一字节 8 位数据
- 5、 主机读取最后一个字节后，在第 9 个 SCL 向从机发送 NACK
- 6、 主机发送 STOP 时序终止读取

软件启动 I2C 接收的操作流程如下图：

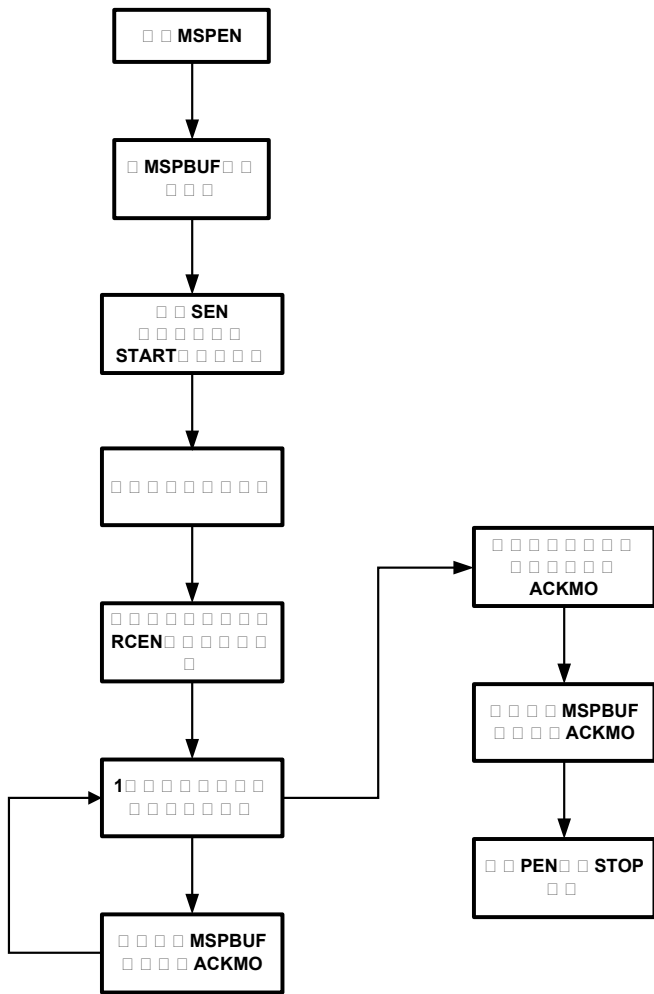


图 19-11I2C 软件发送数据流程图

主机每次接收完从机发送的数据后，根据ACKMO寄存器回发响应。ACKMO复位值为0，即默认状态下主机回发ACK。如果软件希望主机在接收完成后回发NACK，则需要在前一个字节接收完成中断中将ACKMO寄存器改写为1。ACKMO为1的情况下，主机在发送完响应后会自动清零ACKMO。

I2C主机从7位地址从机读出数据的波形示意图如下：



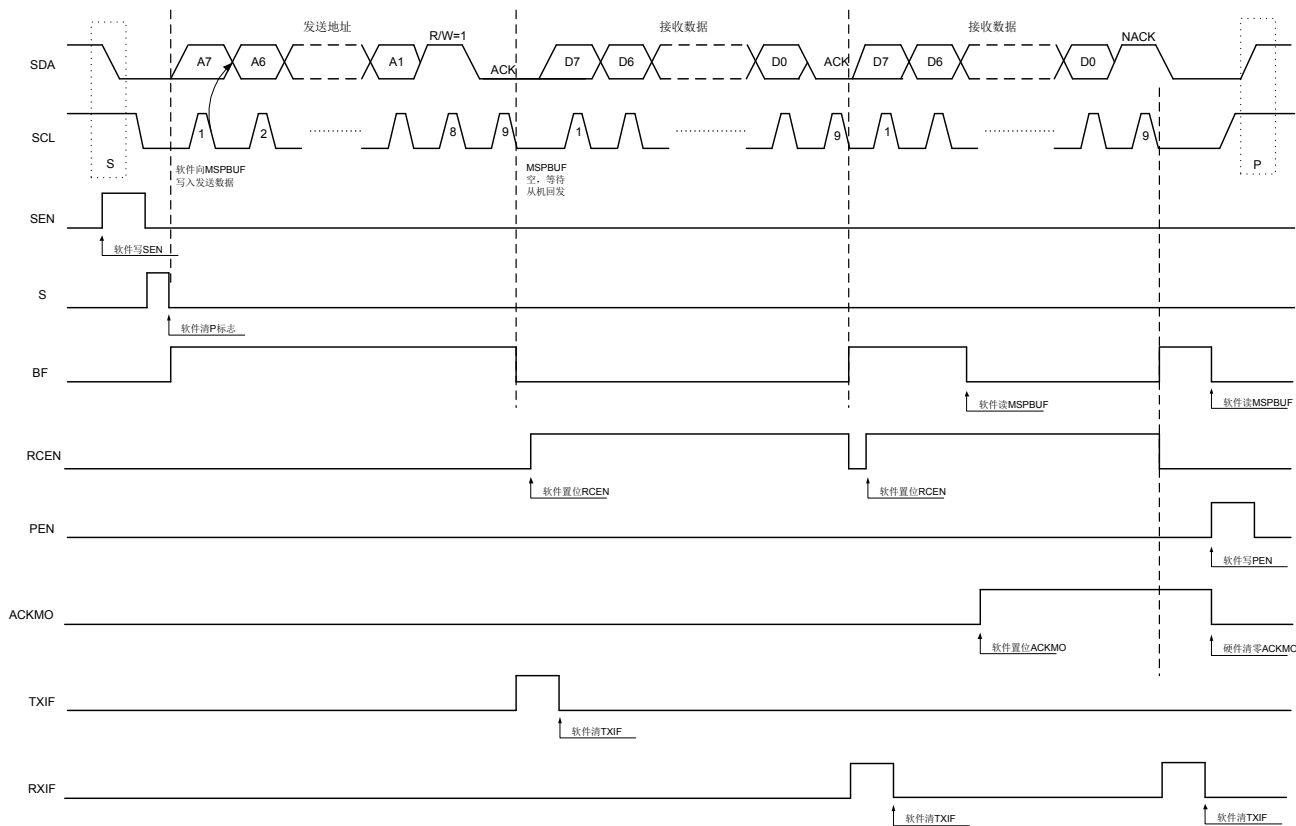


图 19-12I2C 从 7 位地址从机读取数据流程图

双向数据传输（组合模式）

典型的双向数据读写流程图如下图所示。在主机发送或读取数据过程中，主机可以通过发送 Repeated Start 时序来重新启动一次新的发送或读取通信，所以主机在一次通信中，即可以有数据发送也可以有数据读取。

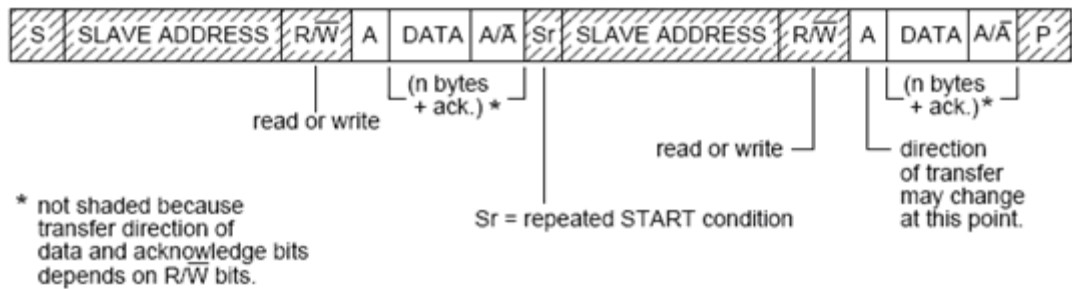


图 19-13 双向数据通信帧格式

组合传输的软件操作流程与单向传输类似，只是在某个字节收发完成后，通过发送ReSTART时序和从机地址字节来修改传输方向。

19.9.2 10bit 寻址

在10bit寻址时，主机发送的第一个字节包含部分从机地址（11110\_A9\_A8）和传输方向位（ $R/\overline{W}$ ），第二个字节包含剩余从机地址（A7~A0）。两个字节地址发送完成后，再进行数据传输。

主机向从机写入数据

典型的10bit寻址，主机向从机写入数据的数据流图如下图所示。

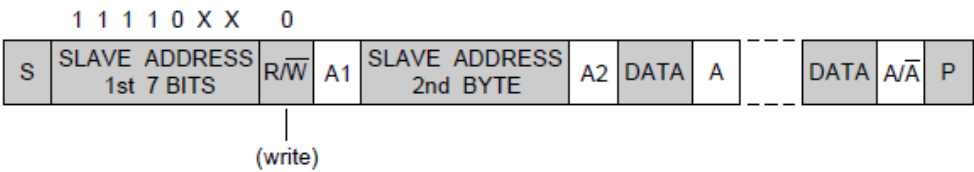


图 19-14 10bit 寻址，主机向从机写入数据

- 1、主机发起 START 时序
- 2、主机发送第一个从机地址字节，以 11110 开头，跟随 2bit 从机地址最高位，以及 R/W 标志位，发送数据时 R/W 位为 0
- 3、主机检查从机回发的 ACK
- 4、主机发送第二个从机地址字节，包含从机地址的低 8 位
- 5、主机检查从机回发的 ACK
- 6、主机继续向从机写入数据
- 7、主机完成所有数据发送后，发送 STOP 时序

软件启动 I2C 主机发送的操作流程如下图：



- 5、 主机再次发送第一字节从机地址，将 R/W 为改为 0
- 6、 此时设置 MSPCON.RCEN 为 1，主机转为接收状态
- 7、 主机开始接收第一字节 8 位数据，并在第 9 个 SCL 向从机发送有效 ACK,从而继续读取下一字节 8 位数据
- 8、 主机读取最后一个字节后，在第 9 个 SCL 向从机发送 NACK
- 9、 主机发送 STOP 时序终止读取

软件启动 I2C 接收的操作流程如下图：

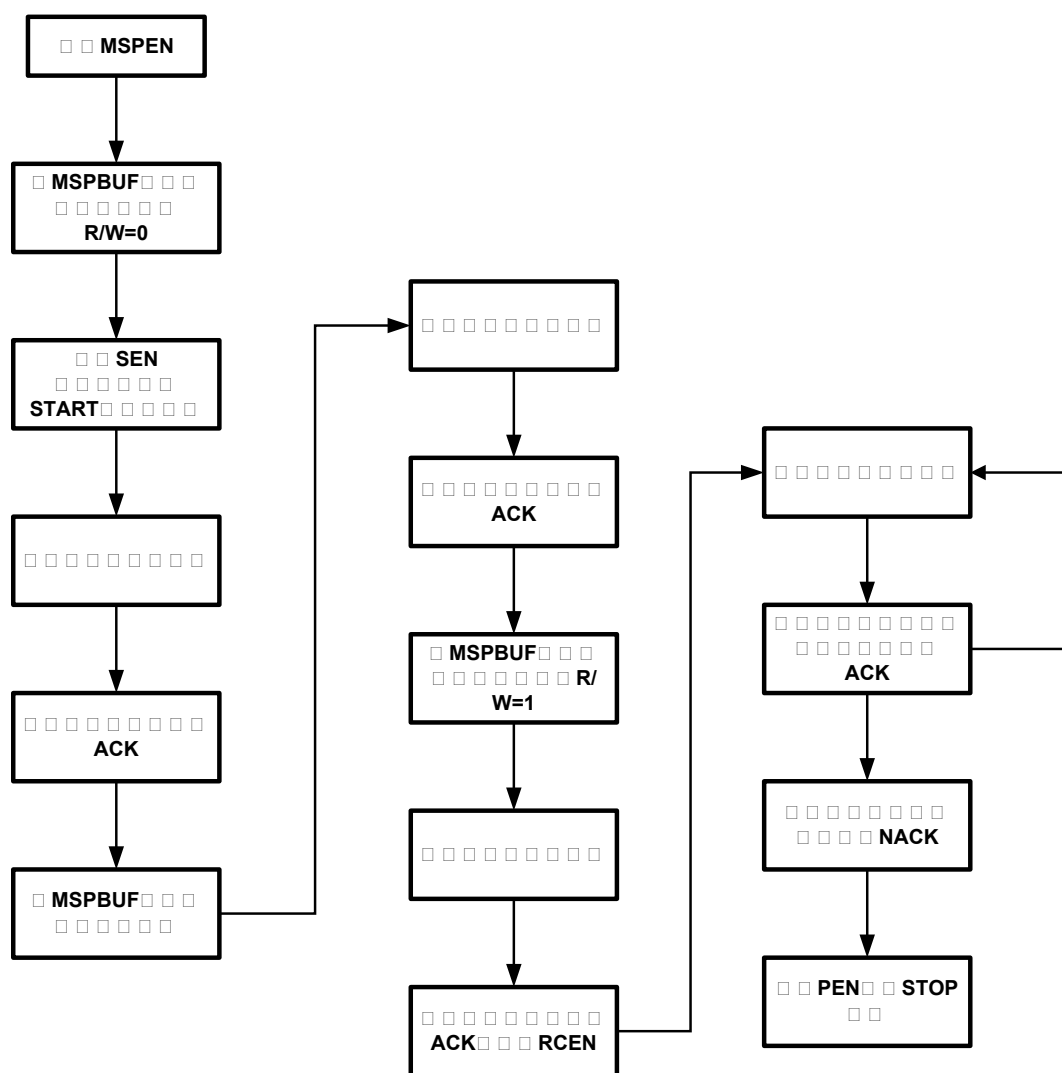


图 19-17I2C 软件发送数据流程图

## 双向数据传输（组合模式）

典型的双向数据读写流程图如下图所示。在主机发送或读取数据过程中，主机可以通过发送 Repeated Start 时序来重新启动一次新的发送或读取通信，所以主机在一次通信中，即可以有数据发送也可以

有数据读取。

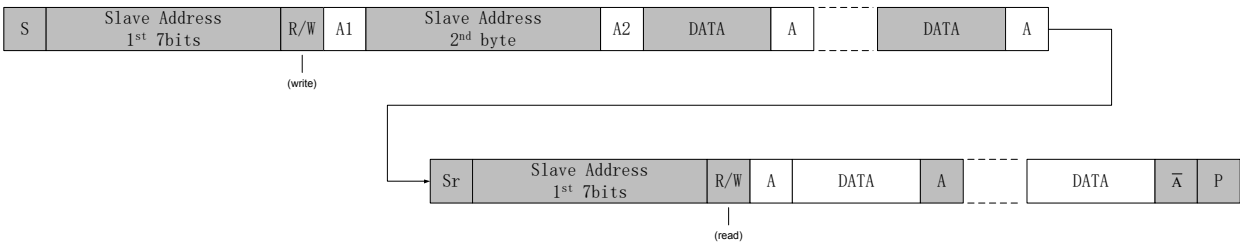


图 19-18I2C 软件发送数据流图

组合传输的软件操作流程与单向传输类似，只是在某个字节收发完成后，通过发送ReSTART时序和1st从机地址字节来修改传输方向。

### 19.9.3 DMA

I2C 主机支持 DMA，需要注意的是，必须在 I2C 模块的总线时钟（APBCLK）使能的情况下，才能使用 DMA 功能。

#### 主机使用DMA向从机写入数据

主机使用 DMA 发送数据时，包括从机地址字节和发送数据在内的所有数据都需要事先写入 RAM 中，并通过 DMA 请求发送出去。软件应事先将目标 DMA 通道配置为 I2C\_TX。

在 DMAEN=1 的情况下，MSPEN 置位，如果数据缓存寄存器 MSPBUF 为空，I2C 模块将产生 DMA 请求，DMA 模块响应请求后将 RAM 中的待发数据写入 MSPBUF，同时 I2C 模块自动置位 SEN 产生 START 时序，开始数据发送（第一个字节是从机地址）。DMA 发送模式下，I2C 并不检查发送数据的合法性，软件必须保证 RAM 中的数据是正确的。

每个字节发送完成后，I2C 检查从机 ACK，如果 ACK 正确则产生新的 DMA 请求，如果收到 NACK 则产生 NACK 中断，并不再产生 DMA 请求。

当DMA完成指定长度的数据发送后，产生DMA传输完成中断，此时可以又软件置位PEN产生STOP时序，也可以由 I2C 硬件根据 DMA 传输完成信号自动置位 PEN 产生 STOP 时序。可以通过设置 AUTOEND 寄存器来选择所需的策略。

主机使用 DMA 进行发送的流程如下图：

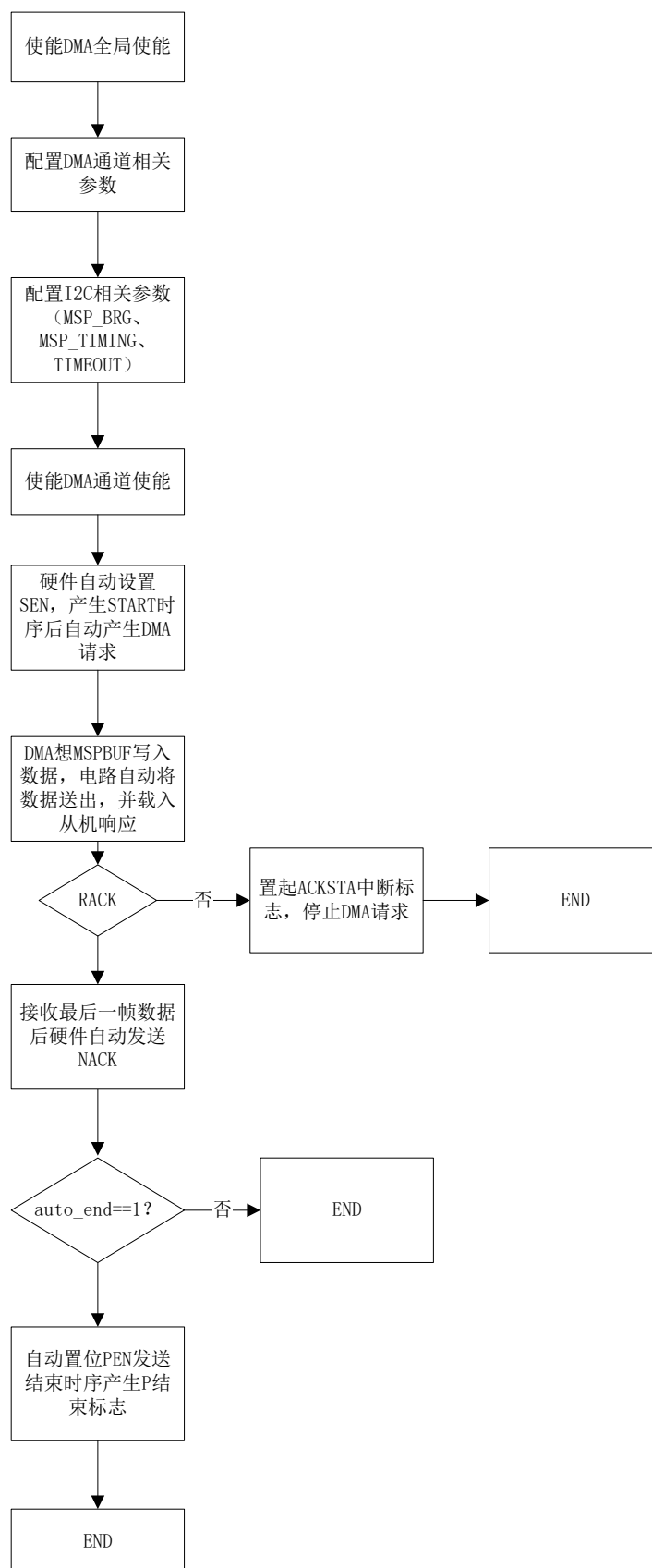


图 19-19 I2C 主机 DMA 发送流程图

## 主机使用DMA从从机读取数据

这种场景下，从机寻址字节必须由软件发送。软件应事先将目标 DMA 通道配置为 I2C\_RX。

软件首先发送完从机地址后，设置 MSP\_DMAEN=1，然后使能对应的 DMA 通道，I2C 自动进入接收模式，并在每个字节接收完成后产生 DMA 请求，通知 DMA 来读取 MSPBUF 内容，同时向从机回发 ACK。

当 DMA 传输达到指定长度后，DMA 的传输完成标志将通知 I2C 回发 NACK。随后根据 AUTOEND 寄存器配置，可以由软件或硬件置位 PEN 产生 STOP 时序。

*注意：当I2C主机通过DMA进行数据接收时，在不同AUTOEND配置和相同DMA传输长度（CHxTSIZE）配置下，DMA接收字节数会有差别。当AUTOEND=0时，接收字节数为CHxTSIZE+1；当AUTOEND=1时，接收字节数为CHxTSIZE。*

主机使用DMA进行接收的流程如下图：

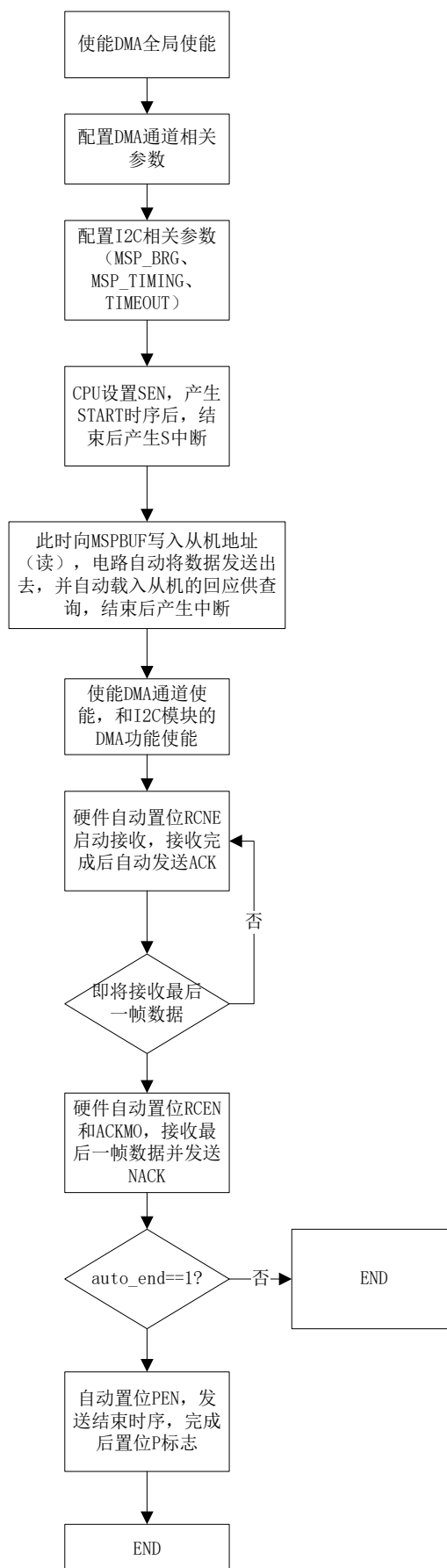


图 19-20 I2C 主机 DMA 接收流程图



### 19.9.4 SCL 延展 (Slave Clock Stretching)

I2C 总线运行低速从机通过拉低 SCL 的方式通知主机暂停数据通信。I2C 主机必须支持这一特性，因此在每个字节收发起始位置处，主机在尝试发送 SCL 高电平后，需要自动检查总线上 SCL 的实际电平，如果不是高电平，意味着从机正在进行 SCL 延展，主机会持续监控 SCL 电平，直到 SCL 为高，才开始后续操作。

**注意：**主机只在每字节收发的第一个 SCL 上升沿处进行 SCL 延展检查。

### 19.9.5 超时机制

I2C主机还实现了超时机制，即发现从机长时间拉低SCL导致总线无法通信的情况下，产生超时报警中断并返回IDLE状态。

当主机检测到SCL延展，其内部定时器开始计时，主机设定的SCL延展超时的时长最长是4096个SCL周期，假设波特率为100K，则超时周期大约是40ms，如果波特率是400K，则超时周期大约是10ms。通过12bit的TIMEOUT寄存器，软件可以设置超时周期。软件必须在MSPEN为0的情况下设置TIMEOUT寄存器，此寄存复位值为0xFFFF，即表示最长4096\*T<sub>SCL</sub>的超时周期，当检测到SCL延展后，TIMEOUT寄存器开始向下递减，当计数到0时，计数停止，TIMEOUT寄存器被复位到0xFFFF，同时触发超时中断。因此通过修改TIMEOUT初始值，可以设定超时周期。

$$T_{SCL\_STRETCHING\_TIMEOUT} = TIMEOUT[11:0] * T_{SCL}$$

当发生TIMEOUT中断时，建议软件复位I2C模块。

此功能可以被关闭，如果关闭硬件超时，软件也可以通过定时器结合SCL引脚状态判断来自行实现任意长度的超时判决。

### 19.9.6 可编程时序

I2C 模块的主机模式提供了灵活的时序编程特性，允许用户定义 SCL 时钟的低电平宽度、高电平宽度，SDA 数据的建立和保持时间。

通过 MSPBRG 寄存器可以设置 SCL 的低电平和高电平宽度，通过 SDAHD 寄存器可以配置 SDA 数据相对 SCL 时钟脉冲的保持和建立保持时间长度。

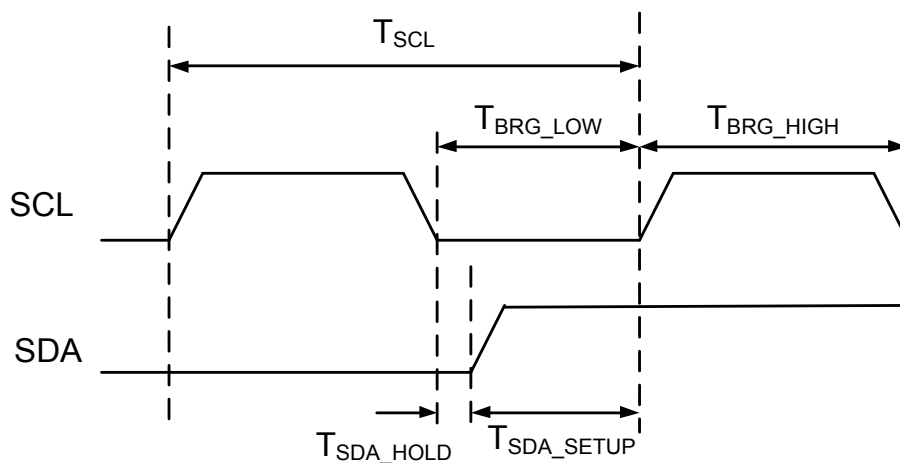


图 19-21 主机时序控制

上图中， $T_{SCL}$ 为通信波特率，各个参数可以由以下公式表达：

$$T_{SCL} = T_{BRG\_LOW} + T_{BRG\_HIGH}$$

$$T_{SDA\_SETUP} = T_{BRG\_LOW} - T_{SDA\_HOLD}$$

注意，应用中对MSPBGRH、MSPBRGL和SDAHD寄存器的配置必须满足以下要求，如果违反这些要求将导致异常的总线时序：

$$MSPBRGH \geq 2$$

$$MSPBRGL \geq 2$$

$$MSPBRGL - 1 \geq SDAHD \geq 1$$

$$TIMEOUT \geq 1$$

## 19.10 I<sup>2</sup>C 从机功能

I2C 从机的工作不需要系统时钟，因此可以在芯片休眠的状态下进行数据收发和唤醒。

从机接收完 1 字节数据后，产生中断通知 CPU 处理数据，在 CPU 取走数据前硬件可以将 SCL 拉低（软件控制使能），通知发送方正忙，发送方应暂停发送直到 SCL 放开。若接收方无法响应 ACK，发送方检测 ACK 失败后应发送 P 终止通信或者发送 Sr 开始新的通信。

从机发送完 1 字节数据后，产生中断通知 CPU，硬件拉低 SCL 令主机等待，CPU 响应中断并准备好下一字节数据后再放开 SCL，主机继续发送 SCL 使从机继续数据发送。

### 19.10.1 从机寻址

根据 SSPCON.A10EN 寄存器状态，从机可以支持 7bit 或者 10bit 寻址过程。从机地址由 SLAVE\_ADDR 寄存器定义。

对于 10bit 从机地址应用，即 SSPCON.A10EN=1 的情况下，要求 1st 字节必须以 11110 开头，否则将触发 ADDR\_ERROR 错误标志。而在 SSPCON.A10EN=0 的情况下，如果从机收到了 11110 开头的地址字节，也会置位 ADDR\_ERROR 错误标志。

### 19.10.2 从机发送数据

推荐操作流程：

- 从机接收到地址字节（R/W=1），回发ACK，产生地址匹配中断
- 由于R/W=1，硬件自动进行SCL延展，从机进入发送状态
- 软件响应中断，查询R/W标志，确认是从机发送
- 软件将待发送数据写入SSPBUF
- 硬件自动释放SCL
- 新的SCL到来，SSPBUF移位输出到SDA总线
- 接收ACK并产生发送完成中断
- 重复数据发送过程直到接收到STOP时序，或接收到主机NACK

下图是一个典型的从机数据发送波形示意图：

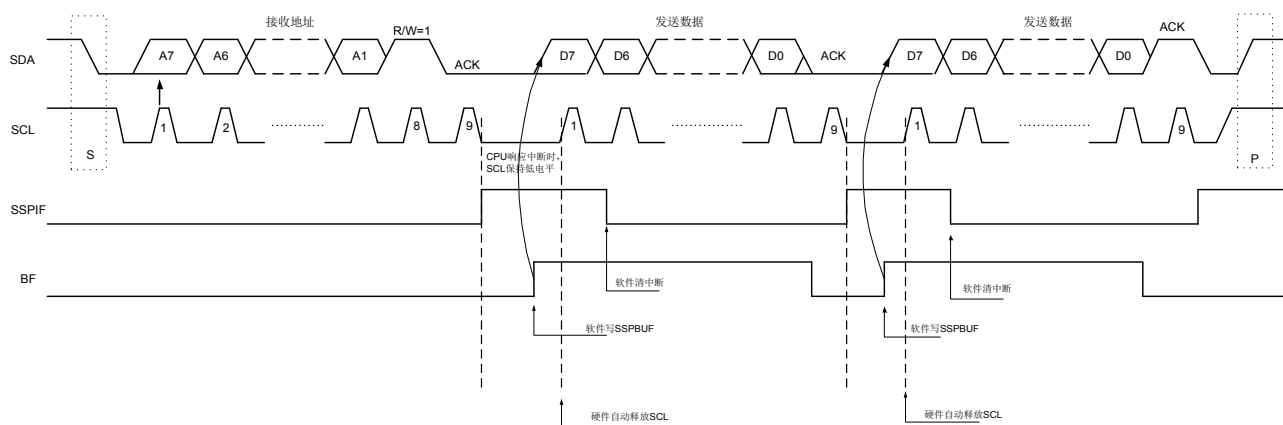


图 19-22 从机数据发送波形

在从机发送流程中，当从机收到正确地址时，ADM标志置位，地址字节不会被写入SSPBUF，因此BF标志不会置位。硬件自动拉低SCL信号等待软件写入SSPBUF，当软件写SSPBUF后BF置位，同时硬件释放SCL。

### 19.10.3 从机接收数据

推荐操作流程：

- 从机接收到地址字节（R/W=0），回发ACK，产生地址匹配中断
- 由于R/W=0，硬件自动进行SCL延展，从机保持接收状态
- 软件响应中断，查询R/W标志，确认是从机接收
- 软件读SSPBUF硬件自动释放SCL，开始接收数据
- 主机数据字节到来，字节接收完成后硬件置位BF标志
- 从机回发ACK，并产生接收完成中断
- 硬件自动进行SCL延展（SCLSEN=1）
- 软件响应中断，读取SSPBUF，硬件自动清零BF标志
- 硬件自动释放SCL
- 重复数据接收过程直到接收到STOP时序，或者软件将ACKEN置0

下图是一个典型的从机数据接收波形示意图（SCLSEN=1）：

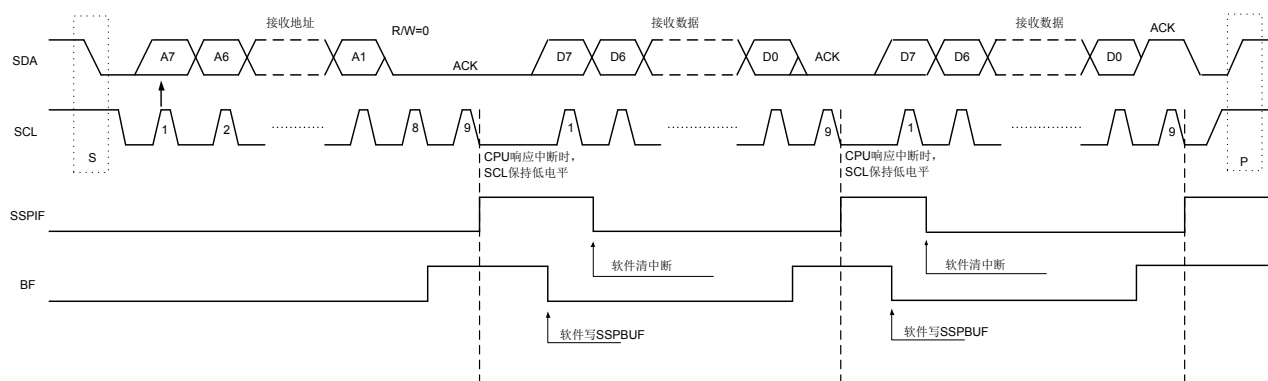


图19-23从机数据接收波形

从机接收过程中，从机首先收到地址字节，地址匹配的情况下，ADM标志置位，地址字节将被写入SSPBUF并置位BF标志，然后硬件拉低SCL。当软件读取SSPBUF后，BF标志自动清零，硬件释放SCL，可以进行后续数据接收。

*注意：从机接收流程中地址字节会被写入SSPBUF并导致BF置位，软件需要读取SSPBUF来清零BF并释放SCL。而从机发送流程中地址字节不会被写入SSPBUF因此也不会置位BF标志。*

从机接收数据可以被动结束通信或主动结束通信。

如果主机主动下发STOP，则从机被动结束本次通信。或者，软件在中断处理程序中将ACKEN寄存器清零，则从机在接收完下一个字节后，将回发NACK，主机接收NACK后将下发STOP结束本次通信。

### 从机SCL延展

I2C从机默认使能SCL延展（slave clock stretching），但是软件可以关闭这个功能（SCLSEN寄存器）以适应不支持从机SCL延展的主机。

当SCL延展使能的情况下，数据接收完成后，软件只有在SCL延展期间读取接收缓冲区时，才能清零BF标志。如果接收中出现了数据溢出，SSPOV标志置位，此时硬件回发NACK，并且SCL不再被延展，以便主机下发STOP；SSPOV置位的情况下，建议软件等待STOP标志置位，再读取接收缓冲区清零BF标志。

### 接收数据溢出

当从机接收缓冲区满（BF=1）时，如果又收到新的数据，则发生接收溢出，SSPOV标志置位。接收缓冲区中的老数据将被新的数据覆盖。只有在从机关闭了SCL延展功能的情况下，才有可能发生接收数据溢出。

下图是SCLSEN=0情况下发生数据接收溢出的示意图：

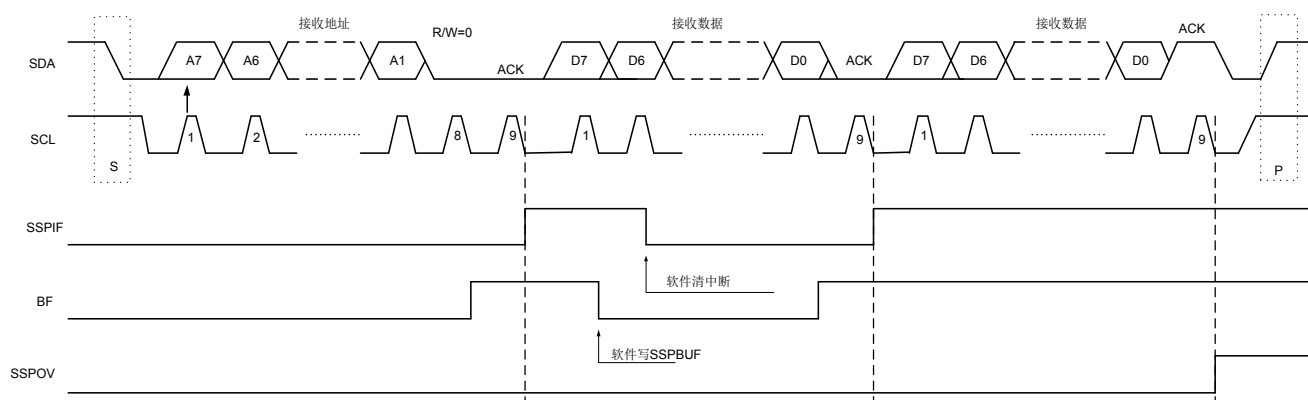


图19-24从机数据接收波形（SCLSEN=0，接收溢出）

#### 19.10.4 从机低功耗接收唤醒

由于I2C从机不需要系统时钟即可工作，因此可以在休眠模式下接收数据并唤醒MCU。

I2C从机支持START时序唤醒、地址匹配唤醒和数据接收完成唤醒。

软件设置流程：

- 关闭I2C主机
- 设置从机地址
- 根据所需的唤醒事件，设置SE、ADME或BFE中断使能
- 设置对应的GPIO为I2C功能
- 置位SSPEN，启动I2C从机
- 进入休眠模式等待数据接收
- 当唤醒事件到来后，软件查询唤醒源，处理I2C数据传输

#### 19.10.5 DMA

I2C 从机支持 DMA 操作，需要注意的是，必须在 I2C 的总线时钟（APBCLK）使能的情况下才能进行 DMA 操作。总线时钟被用于产生 DMA 请求并接收 DMA 应答。

##### 从机使用DMA接收数据

当 I2C 从机接收到正确的地址后，产生 ADM 中断标志，软件响应中断后，查询接收到的 R/W 位，如果为 0 表示主机准备向从机写入数据。此时软件可以配置特定 DMA 通道为 I2C\_RX，并使能 I2C 从机的 DMAEN；随后每次从机完成一个字节的接收，将产生 DMA 请求，通知 DMA 来读取 SSPBUF。

结束 DMA 从机接收有两种可能：

- 1) 数据传输长度还未达到 DMA 长度配置，主机就下发了 STOP 时序，软件应响应 STOP 中断并主动处理这种情况；
- 2) 数据传输长度达到 DMA 长度配置，但是由于 DMA 请求是在从机回发 ACK 后产生，所以软件应响应 DMA 传输完成中断，并将 ACKEN 清零，这样从机会在接收完下一个字节后，回发 NACK，结束本次通信。

从机使用 DMA 进行接收的流程如下图：

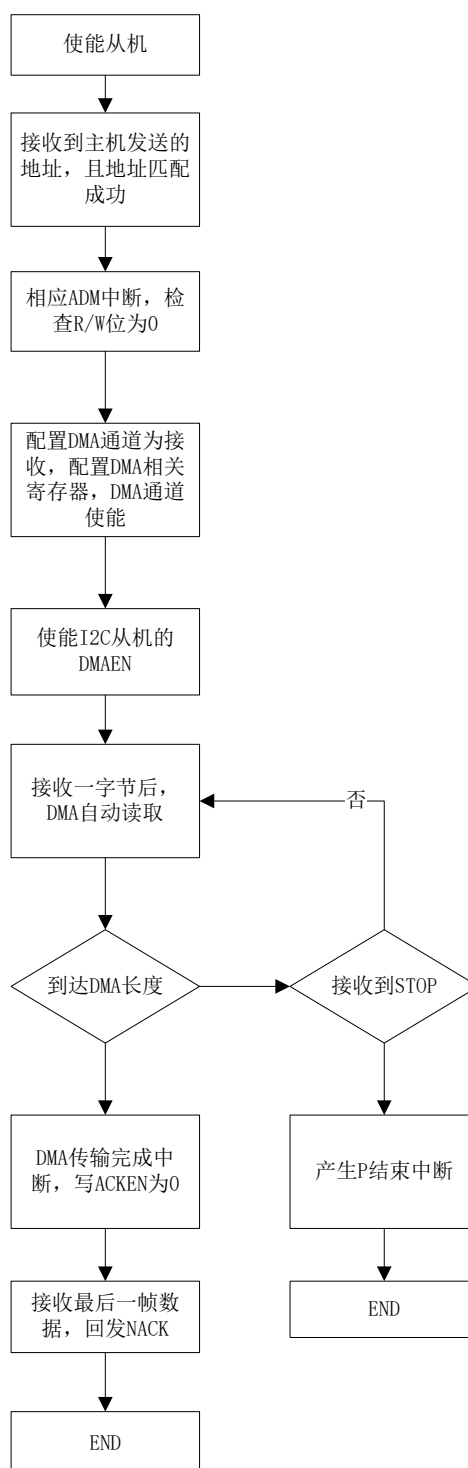


图 19-25 I2C 从机 DMA 接收流程图

### 从机使用DMA发送数据

当 I2C 从机接收到正确的地址后，产生 ADM 中断标志，软件响应中断后，查询接收到的 R/W 位，如果为 1 表示主机准备从从机读出数据。此时软件需要先读取 SSPBUF 清除 BF 标志，然后配置特定 DMA 通道为 I2C\_TX，并使能 I2C 从机的 DMAEN；随后当从机数据缓存 SSPBUF 为空时，将



产生 DMA 请求，通知 DMA 写入 SSPBUF。

只有主机回发 NACK 才能结束读取操作。当读取数据长度大于 DMA 设置的传输长度时，由于 DMA 不再响应 I2C 请求，从机将一直拉低 SCL，直到软件关闭 I2C 从机模块。

从机使用 DMA 进行发送的流程如下图：

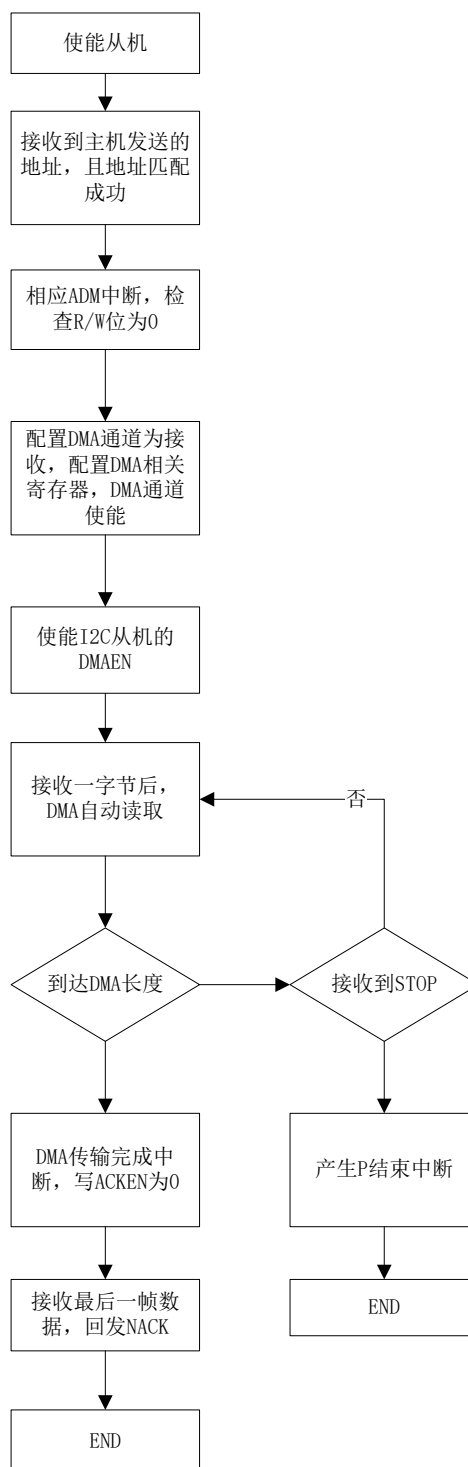


图 19-26 I2C 从机 DMA 发送流程图

### 19.10.6 从机时序

由于从机的数据收发只使用 SCL 进行，因此需要一些模拟延迟来实现 SDA 的数据建立和保持时间控制，而 SCL 的时序完全由主机控制。

从机时序控制如下图所示。根据 I2C 协议要求，SDA 相对于 SCL 下降沿的数据保持时间最小为 0ns，即从机使用 SCL 下降沿发送数据即可满足要求。但是考虑到总线上 SCL 波形的实际下降时间，为了更好的覆盖保持时间要求，在 SDA 输出上额外加入大于 300ns 的 RC 延迟。这个延迟仅需施加在 I2C 从机的 SDA 输出上（SSP\_SDAO）

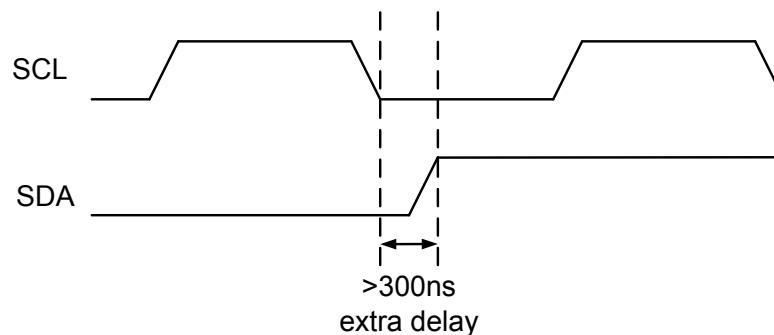


图 19-27 SDA 输出延迟波形

## 19.11 寄存器

offset 地址	名称	符号
<b>I2C(模块起始地址:0x40012400)</b>		
0x00000000	I2C 主机配置寄存器 (I2C Master Config Register)	I2C_MSPCFGR
0x00000004	I2C 主机控制寄存器 (I2C Master Control Register)	I2C_MSPCR
0x00000008	I2C 主机中断使能寄存器 (I2C Master Interrupt Enable Register)	I2C_MSPIER
0x0000000C	I2C 主机中断标志寄存器 (I2C Master Interrupt Status Register)	I2C_MSPISR
0x00000010	I2C 主机状态寄存器 (I2C Master Status Register)	I2C_MSPSR
0x00000014	I2C 主机波特率寄存器 (I2C Master Baud rate Generator Register)	I2C_MSPBGR
0x00000018	I2C 主机收发缓存寄存器 (I2C Master transfer Buffer)	I2C_MSPBUF
0x0000001C	I2C 主机时序控制寄存器 (I2C Master Timing Control Register)	I2C_MSPTCR
0x00000020	I2C 主机超时寄存器 (I2C Master Time-Out Register)	I2C_MSPTOR
0x00000024	I2C 从机控制寄存器 (I2C Slave Control Register)	I2C_SSPCR
0x00000028	I2C 从机中断使能寄存器 (I2C Slave Interrupt Enable Register)	I2C_SSPIER
0x0000002C	I2C 从机中断标志寄存器 (I2C Slave Interrupt Status Register)	I2C_SSPISR
0x00000030	I2C 从机状态寄存器 (I2C Slave Status Register)	I2C_SSPSR
0x00000034	I2C 从机收发缓存寄存器 (I2C Slave transfer Buffer)	I2C_SSPBUF
0x00000038	I2C 从机地址寄存器 (I2C Slave Address Register)	I2C_SSPADR

### 19.11.1 I2C 主机配置寄存器 (I2C\_MSPCFGR)

名称	I2C_MSPCFGR							
Offset	0x00000000							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-						AUTOEN D	MSP_D MAEN
位权限	U-0						R/W-0	R/W-0
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							
位权限	U-0							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0

位名	-	TOEN	MSPEN
位权限	U-0	R/W-0	R/W-0

位号	助记符	功能描述
31:18	-	RFU: 未实现, 读为 0
17	AUTOEND	主机 DMA 自动终止 (DMA automatic transfer end) 1: DMA 指定长度传输完成后, 自动发送 STOP 时序 0: DMA 指定长度传输完成后, 等待软件接管
16	MSP_DMAEN	主机 DMA 使能 (Master DMA enable) 0: 关闭 DMA 功能 1: 使能 DMA 功能
15:2	-	RFU: 未实现, 读为 0
1	TOEN	SCL 拉低超时使能 (Time Out enable) 1: 使能超时功能, 超时周期由 MSPTO 寄存器定义 0: 关闭超时功能
0	MSPEN	I2C 主机模块使能控制位 (Master enable) 1: I2C 主机使能 0: I2C 主机禁止

### 19.11.2 I2C 主机控制寄存器 (I2C\_MSPCR)

名称	I2C_MSPCR							
Offset	0x00000004							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							
位权限	U-0							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	-				RCEN	PEN	RSEN	SEN
位权限	U-0				R/W-0	R/W-0	R/W-0	R/W-0

位号	助记符	功能描述
31:4	-	RFU: 未实现, 读为 0
3	RCEN	主机接收模式下, 接收使能位 (Receive enable) 1: 主机接收使能 0: 接收禁止 主机通信中, 软件在发送完地址字节后, 通过置位 RCEN 将传输方向切换为主机接收, 然后可以接收来自于从机的数据。RCNE 在接收过程中保持为 1, 直到软件置位 PEN 发送 STOP 时序。
2	PEN	STOP 时序产生使能控制位, 软件写 1 发送 STOP 时序, 发送完成后硬件自动清零 (Stop Enable)
1	RSEN	Repeated START 时序产生使能控制位, 软件写 1 发送 Repeated START 时序, 发送完成后硬件自动清零 (Repeated Start)

位号	助记符	功能描述
		Enable)
0	SEN	START 时序产生使能控制位，软件写 1 发送 START 时序，发送完成后硬件自动清零 (Start Enable)

### 19.11.3 I2C 主机中断使能寄存器 (I2C\_MSPIER)

名称	I2C_MSPIER							
Offset	0x00000008							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							
位权限	U-0							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	-	WCOL	OVTE	SE	PE	NACK	TXIE	RXIE
位权限	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

位号	助记符	功能描述
31:7	-	RFU: 未实现, 读为 0
6	WCOL	WCOL 中断使能寄存器 (Write collision interrupt enable) 1: 允许写冲突中断 0: 禁止写冲突中断
5	OVTE	SCL 超时中断使能寄存器 (SCL overtime enable) 1: 允许超时中断 0: 禁止超时中断
4	SE	START 时序中断使能寄存器 (Start interrupt enable) 1: 允许 START 时序中断 0: 禁止 START 时序中断
3	PE	STOP 时序中断使能寄存器 (Stop interrupt enable) 1: 允许 STOP 时序中断 0: 禁止 STOP 时序中断
2	NACK	主机发送模式下 NACK 中断使能寄存器 (Non-ACK interrupt enable) 1: 允许收到 NACK 产生中断 0: 禁止产生 NACK 中断
1	TXIE	I2C 主机发送完成中断使能(Transmit done interrupt enable) 1: 允许发送完成中断 0: 禁止发送完成中断
0	RXIE	I2C 主机接收完成中断使能(Receive done interrupt enable) 1: 允许接收完成中断 0: 禁止接收完成中断

## 19.11.4 I2C 主机中断标志寄存器 (I2C\_MSPISR)

名称	I2C_MSPISR							
Offset	0x0000000C							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							
位权限	U-0							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	-	WCOL	OVT	S	P	ACKSTA	TXIF	RXIF
位权限	U-0	R/W-0	R-0	R-0	R-0	R/W-0	R/W-0	R/W-0

位号	助记符	功能描述
31:7	-	RFU: 未实现, 读为 0
6	WCOL	写冲突检测位, MCU 只能在完成 START 时序或发送完成一帧读写之后才能写 MSPBUF, 否则发生写冲突; 硬件置位, 软件写 1 清零(Write Collision Interrupt Flag, write 1 to clear) 1: 发送写冲突 0: 未发生冲突
5	OVT	SCL 超时中断标志, 仅在 TOEN 为 1 时工作(SCL OverTime Interrupt Flag) 1: 发生 SCL 超时 0: 没有发生 SCL 超时
4	S	START 时序发送完成中断标志, 硬件置位, 软件读取后清零(Start Interrupt flag)
3	P	STOP 时序发送完成中断标志, 硬件置位, 软件读取后清零(Stop interrupt flag)
2	ACKSTA	主控发送模式下, 来自从机的回应信号; 当主机发送后收到 NACK, 此标志可以产生中断; 硬件置位, 软件写 1 清零。(Acknowledge Status Flag ,write 1 to clear) 1: 从机回应 NACK 0: 从机回应 ACK
1	TXIF	I2C 主机发送完成中断标志, 硬件置位, 软件写 1 清零(Transmit done interrupt flag, write 1 to clear) 此标志寄存器在主机接收完从机回发的 ACK 或 NACK 后置位。
0	RXIF	I2C 主机接收完成中断标志, 硬件置位, 软件写 1 清零(Receive done interrupt flag, write 1 to clear) 此标志寄存器在主机回发完 ACK 或 NACK 后置位。

## 19.11.5 I2C 主机状态寄存器 (I2C\_MSPSR)

名称	I2C_MSPSR							
Offset	0x00000010							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							

位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							
位权限	U-0							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	-		BUSY	RW	-	BF	-	ACKMO
位权限	U-0		R-0	R-0	U-0	R-0	U-0	R/W-0

位号	助记符	功能描述
31:6	-	RFU: 未实现, 读为 0
5	BUSY	I2C 通信状态位 (Busy) 1: 接口处于读写状态, 正在进行数据传输, 0: 已完成数据传输
4	RW	I2C 传输方向状态位 (Read/Write) 1: 主机从从机读取数据 0: 主机向从机写入数据
3	-	RFU: 未实现, 读为 0
2	BF	缓冲器满状态位 (Buffer Full) 接收: 1: 接收完成, MSPBUF 满 0: 接收未完成, MSPBUF 空 发送: 1: 正在发送, MSPBUF 满 0: 发送完成, MSPBUF 空
1	-	RFU: 未实现, 读为 0
0	ACKMO	主控接收模式下, 主机回应信号的状态 (Ack Master output) 1: 主机回发 NACK 0: 主机回发 ACK  <b>注意: 必须在 P 标志寄存器被清零的情况下, 软件才能置位 ACKMO</b>

### 19.11.6 I2C 主机波特率寄存器 (I2C\_MSPBGR)

名称	I2C_MSPBGR							
Offset	0x00000014							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							MSPBRGH[8]
位权限	U-0							R/W-0
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	MSPBRGH[7:0]							
位权限	R/W-0001 0011							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							MSPBRGL[8]
位权限	U-0							R/W-0

位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	MSPBRGL[7:0]							
位权限	R/W-0001 0011							

位号	助记符	功能描述
31:25	-	RFU: 未实现, 读为 0
24:16	MSPBRGH	主机发送的 SCL 时钟高电平宽度, 以 I2C 工作时钟计数 (Master SCL High level length)
15:9	-	RFU: 未实现, 读为 0
8:0	MSPBRGL	主机发送的 SCL 时钟低电平宽度, 以 I2C 工作时钟计数 (Master SCL Low level length)

### 19.11.7 I2C 主机收发缓存寄存器 (I2C\_MSPBUF)

名称	I2C_MSPBUF							
Offset	0x00000018							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							
位权限	U-0							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	MSPBUF							
位权限	R/W-0000 0000							

位号	助记符	功能描述
31:8	-	RFU: 未实现, 读为 0
7:0	MSPBUF	MSPBUF[7:0]: 数据的读写通过对 MSPBUF 的操作完成。发送时, 对 MSPBUF 执行写操作, 同时也载入数据收发移位寄存器 (MSPSR); 接收时, MSPBUF 与 MSPSR 组成双缓冲结构, 读出数据为 MSPBUF 的数据。接收完一个字节的数, MSPSR 将数据载入 MSPBUF, 同时置位 I2CIF。MSPSR 不是直接寄存器, 没有物理地址。 (Master data Buffer)

### 19.11.8 I2C 主机时序控制寄存器 (I2C\_MSPTCR)

名称	I2C_MSPTCR							
Offset	0x0000001C							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							



位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							SDAHD[8]
位权限	U-0							R/W-0
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	SDAHD[7:0]							
位权限	R/W-0000 1010							

位号	助记符	功能描述
31:9	-	RFU: 未实现, 读为 0
8:0	SDAHD	定义 SDA 相对于 SCL 下降沿的保持时间参数, 以 I2C 工作时钟计数 (SDA hold delay) 注意: 最小有效值为 1, 最大有效值为 MSPBRGL

### 19.11.9 I2C 主机超时寄存器 (I2C\_MSPTOR)

名称	I2C_MSPTOR							
Offset	0x00000020							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-				TIMEOUT[11:8]			
位权限	U-0				R/W-1111			
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	TIMEOUT[7:0]							
位权限	R/W-1111 1111							

位号	助记符	功能描述
31:12	-	RFU: 未实现, 读为 0
11:0	TIMEOUT	定义从机 SCL 低电平延展超时周期, 软件可以在 MSPEN=0 的情况下改写 (SCL stretching Time Out) $T_{SCL\_STRETCHING\_TIMEOUT} = TIMEOUT[11:0] * T_{SCL}$

## 19.11.10 I2C 从机控制寄存器 (I2C\_SSPCR)

名称	I2C_SSPCR							
Offset	0x00000024							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-00000000							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-00000000							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-						SCLSE N	SSP_D MAEN
位权限	U-00000000						R/W-1	R/W-0
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	-			ACKEN	SDAO_ DLYEN	SCLI_ ANFEN	A10EN	SSPEN
位权限	U-0			R/W-1	R/W-0	R/W-0	R/W-0	R/W-0

位号	助记符	功能描述
31:10	-	RFU: 未实现, 读为 0
9	SCLSEN	I2C 从机时钟延展使能 (SCL Stretching Enable) 0: 禁止 slave clock stretching 1: 使能 slave clock stretching <i>注意: 当从机使用 DMA 通信时, 必须将 SCLSEN 置 1</i>
8	SSP_DMAEN	I2C 从机 DMA 使能 (Slave DMA enable) 1: 使能 DMA 功能 0: 关闭 DMA 功能
7:5	-	RFU: 未实现, 读为 0
4	ACKEN	ACK 使能位: (Slave Ack Enable) 1 = slave 接收完成后将回发 ACK 0 = slave 不回发 ACK
3	SDAO_DLYEN	SDA 从机输出延迟使能 (SDA output delay enable) 0: bypass 从机 SDA 输出延迟 1: 使能从机 SDA 输出延迟
2	SCLI_ANFEN	SCL 从机输入模拟滤波使能 (SCL input analog filter enable) 0: bypass 模拟滤波 1: 使能模拟滤波
1	A10EN	10 位地址使能: (10bit Slave address enable) 1 = slave 使用 10bit address 0 = slave 使用 7bit address
0	SSPEN	I2C 从机使能位 (Slave enable) 1: 使能 I2C 从机 0: 关闭 I2C 从机

## 19.11.11 I2C 从机中断使能寄存器 (I2C\_SSPIER)

名称	I2C_SSPIER
Offset	0x00000028

名称	I2C_SSPIER							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							
位权限	U-0							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	ADEE	SE	PE	WCOLE	SSPOV E	ADME	TXIE	RXIE
位权限	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

位号	助记符	功能描述
31:8	-	RFU: 未实现, 读为 0
7	ADEE	从机地址错误中断使能, 1 有效 (Address Error Enable)
6	SE	Start 中断使能, 1 有效 (Start interrupt enable)
5	PE	Stop 中断使能, 1 有效 (Stop interrupt enable)
4	WCOLE	WCOL 中断使能, 1 有效 (Write Collision Enable)
3	SSPOVE	SSPOV 中断使能, 1 有效 (Slave Buffer Overflow Enable)
2	ADME	从机地址匹配中断使能, 1 有效 (Address Match Enable)
1	TXIE	发送完成中断使能, 1 有效 (Transmit interrupt enable)
0	RXIE	接收完成中断使能, 1 有效 (Receive interrupt enable)

### 19.11.12 I2C 从机中断标志寄存器 (I2C\_SSPISR)

名称	I2C_SSPISR							
Offset	0x0000002C							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							
位权限	U-0							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	ADE	S	P	WCOL	SSPOV	ADM	TXIF	RXIF
位权限	R/W-0	R-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

位号	助记符	功能描述
31:8	-	RFU: 未实现, 读为 0
7	ADE	从机地址格式错误, 硬件置位, 软件写 1 清零 (Address Error flag, write 1 to clear) 在 7 位地址情况下收到 11110 开头的地址字节, 或者在 10 位

位号	助记符	功能描述
		地址情况下第一个字节不以 11110 开头时, 触发 ADEE
6	S	检测到 start 时序, 硬件置位, 软件读取后自动清零 (Start flag)
5	P	检测到 stop 时序, 硬件置位, 软件读取后自动清零 (Stop flag)
4	WCOL	写冲突标志, 硬件置位, 软件写 1 清零 (Write Collision flag, write 1 to clear) 1: 在 BF=1 的情况下, 软件向 SSPBUF 写入新的数据 0: 无写入冲突 当 WCOL 发生时, 新的数据将被丢弃
3	SSPOV	SSPBUF 溢出标志, 硬件置位, 软件写 1 清零 (Slave buffer overflow flag, write 1 to clear) 1: 在 BF=1 的情况下, 从机又接收到新的数据 0: 没有接收溢出 如果从机使能 SCL 延展, 不会出现接收数据溢出的情况; 因此 SSPOV 只可能在 SCLSEN=0 的情况下被置位。
2	ADM	从机地址匹配标志, 硬件置位, 软件写 1 清零 (Address Matched flag, write 1 to clear) 1: 接收到的 7bit 或 10bit 地址与 SLAVE_ADDR 寄存器内容一致 0: 接收到的地址与 SLAVE_ADDR 不一致
1	TXIF	I2C 从机发送完成标志, 硬件置位, 软件写 1 清零 (Transmit interrupt flag, write 1 to clear)
0	RXIF	I2C 从机接收完成标志, 硬件置位, 软件写 1 清零 (Receive interrupt flag, write 1 to clear)

### 19.11.13 I2C 从机状态寄存器 (I2C\_SSPSR)

名称	I2C_SSPSR							
Offset	0x00000030							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							
位权限	U-0							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	-				BUSY	RW	DA	BF
位权限	U-0				R-0	R-0	R-0	R-0

位号	助记符	功能描述
31:4	-	RFU: 未实现, 读为 0
3	BUSY	从机通信标志 (Busy) 1: 从机数据收发中 0: 从机空闲

位号	助记符	功能描述
2	RW	读写方向状态寄存器 (Read/Write) 1: slave 收到 R/W=1, slave 需要发送数据给 master 0: slave 处于接收数据状态
1	DA	data/address 帧指示 1: 上一字节收到的是数据 0: 上一字节收到的是地址
0	BF	从机数据缓冲区满标志 (Buffer Full flag) 1: SSPBUF 满 0: SSPBUF 空

#### 19.11.14 I2C 从机收发缓存寄存器 (I2C\_SSPBUF)

名称	I2C_SSPBUF							
Offset	0x00000034							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
位名	-							
位权限	U-0							
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	SSPBUF							
位权限	R/W-0000 0000							

位号	助记符	功能描述
31:8	-	RFU: 未实现, 读为 0
7:0	SSPBUF	SSPBUF[7:0]: 数据的读写通过对 SSPBUF 的操作完成。发送时, 对 SSPBUF 执行写操作, 同时也载入数据收发移位寄存器 (SSPSR); 接收时, SSPBUF 与 SSPSR 组成双缓冲结构, 读出数据为 SSPBUF 的数据。接收完一个字节的的数据, SSPSR 将数据载入 SSPBUF, 同时置位 I2CIF。SSPSR 不是直接寄存器, 没有物理地址 (Slave Buffer)

#### 19.11.15 I2C 从机地址寄存器 (I2C\_SSPADR)

名称	I2C_SSPADR							
Offset	0x00000038							
位	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
位名	-							
位权限	U-0							
位	Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
位名	-							
位权限	U-0							
位	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8

名称	I2C_SSPADR							
位名	-						SSPADDR[9:8]	
位权限	U-0						R/W-00	
位	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
位名	SSPADDR[7:0]							
位权限	R/W-0000 0000							

位号	助记符	功能描述
31:10	-	RFU: 未实现, 读为 0
9:0	SSPADDR	从机地址寄存器 (Slave Address) A10EN = 1 10 位地址都有效 A10EN = 0 仅低 7 位有效