FT62F08X

IIC Application note

目录

FT62F08x IIC 应用

1. I2C 接口

I2C 为双线接口 (数据线 SDA 和串行时钟线 SCL)，可通过 I2C 协议与外部设备进行通信，特性如下：

- 主机模式、从机模式
- 多主机兼容
- 标准模式(100kHz)、快速模式(400kHz)
- 7 位或 10 位地址格式、广播呼叫 (General Call)
- 数据从高位开始发送/接收
- 可选时钟拉低扩展 (Clock stretching)
- 支持 I2C 接口 SCL / SDA 开漏输出
- 支持软件复位
- 事件中断：
  - ✓ TX-FIFO / RX-FIFO 状态为空/非空中断
  - ✓ 主机模式下：发送 Start 中断、地址发送完成中断、发送 10 位地址高 2 位中断
  - ✓ 从机模式下：接收地址匹配中断、识别到 General call 中断、检测到 Stop 中断
- 错误中断：
  - ✓ 检测到错位的 Start / Stop 中断
  - ✓ 主机仲裁失败中断
  - ✓ NACK 中断
  - ✓ 产生 Overrun 中断

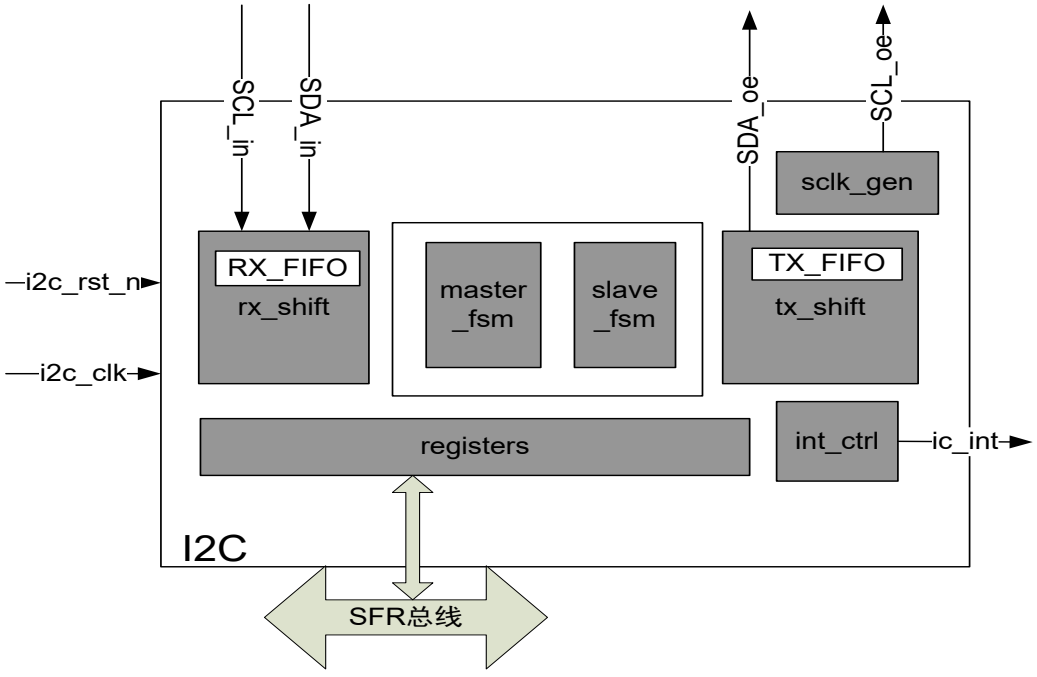

图 1-1 I2C 结构框图

1.1. I2C 接口相关寄存器汇总

名称	状态	寄存器	地址	复位值
MST10B [1]	主机发送地址格式 1 = 10 位　　0 = 7 位	I2CCR1[4]	0x40C	RW-0
SLV10B [1]	从机响应地址格式 1 = 10 位　　0 = 7 位	I2CCR1[3]		RW-0
SPEED [1]	I2C 通信速度 1 = 快速模式 (400kHz) 0 = 标准模式 (100kHz)	I2CCR1[1]		RW-0
MASTER [1]	工作模式 1 = 主机模式 0 = 从机模式	I2CCR1[0]		RW-0
SOFTRST	软件复位 (当 ACTIVE = 1 时可写) 1 = 复位 I2C 模块 0 = 无意义	I2CCR2[6]	0x40D	RW-0
AGCALL [1]	广播呼叫 (General call) 主机模式： 1 = 发送 General call 地址 (0x00) 0 = 发送正常的从机地址 从机模式： 1 = 响应 General call 0 = 不响应 General call	I2CCR2[5]		RW-0
SNACK [1]	接收应答 1 = 发送 NACK 0 = 发送 ACK (地址匹配或接收到数据)	I2CCR2[4]		RW-0
RXHLD [1]	时钟拉伸 (当 RX-FIFO 非空时) 1 = 使能 (拉低 SCL) 0 = 禁止 (新接收的数据将会丢失)	I2CCR2[1]		RW-0
EVSTRE	时钟拉伸 (当 SBF / ADDF / ADD10F 置位后) 1 = 使能 (拉低 SCL) 0 = 禁止	I2CCR3[2]	0x40E	RW-0
ENABLE	I2C 接口 1 = 使能 0 = 关闭	I2CCR3[0]		RW-0

[1] 当 ENABLE = 0 时可写。

<table>
<tr><th>名称</th><th>状态</th><th>寄存器</th><th>地址</th><th>复位值</th></tr>
<tr><td>ADD[7:0] [2]</td><td>从机地址低有效位 (LSB)
7 位地址：ADD[6:0]有效，ADD[7]忽略；
10 位地址：ADD[7:0] = 低 8 位；
注：主机模式下为目标从机地址，从机模式下为本机地址；</td><td>I2COARL[7:0]</td><td>0x40F</td><td>RW−0000 0000</td></tr>
<tr><td>ADD[9:8] 2</td><td>从机地址高有效位 (MSB)
7 位地址：ADD[9:8]忽略；
10 位地址：ADD[9:8] = 高 2 位；
注：主机模式下为目标从机地址，从机模式下为本机地址；</td><td>I2COARH[1:0]</td><td>0x410</td><td>RW−00</td></tr>
<tr><td>I2CEN</td><td>I2C 模块时钟
1 = 打开
0 = 关闭</td><td>PCKEN[6]</td><td>0x09A</td><td>RW−0</td></tr>
<tr><td>SYSON</td><td>睡眠模式下，系统时钟控制
1 = 保持运行
0 = 关闭</td><td>CKOCON[7]</td><td>0x095</td><td>RW−0</td></tr>
<tr><td>FREQ[5:0] 2</td><td>I2C 外设时钟频率 Fmaster
000000 = 禁止
000001 = 1MHz
000010 = 2MHz
……
011000 = 24MHz
> 011000 = 禁止
注：Fmaster 必须与 SysClk 相同</td><td>I2CFRWQ[5:0]</td><td>0x411</td><td>RW−0000 00</td></tr>
<tr><td>DUTY 2</td><td>快速模式下，占空比设置
1 = SCLL / SCLH = 16 / 9
0 = SCLL / SCLH = 2 / 1
注：标准模式下，SCLL / SCLH = 1 / 1</td><td>I2CCCRH[6]</td><td>0x415</td><td>RW−0</td></tr>
<tr><td>CCR[7:0] 2</td><td>主机模式下，SCL 时钟周期低 8 位</td><td>I2CCCRL[7:0]</td><td>0x414</td><td>RW−0000 0000</td></tr>
<tr><td>CCR[11:8] 2</td><td>主机模式下，SCL 时钟周期高 4 位
SCL 时钟周期公式：
<table>
<tr><th>模式</th><th>周期</th><th>SCLL</th><th>SCLH</th></tr>
<tr><td>标准模式</td><td>2*CCR*Fmaster</td><td>CCR*Fmaster</td><td>CCR*Fmaster</td></tr>
<tr><td>快速模式 (DUTY=0)</td><td>3*CCR*Fmaster</td><td>2*CCR*Fmaster</td><td>CCR*Fmaster</td></tr>
<tr><td>快速模式 (DUTY=1)</td><td>25*CCR*Fmaster</td><td>16*CCR*Fmaster</td><td>9*CCR*Fmaster</td></tr>
</table></td><td>I2CCCRH[3:0]</td><td>0x415</td><td>RW−0000</td></tr>
</table>

[2] 当 ENABLE = 0 时可写。

名称	状态	寄存器	地址	复位值
DR[7:0]	数据寄存器 写时：将新数据写入到 TX-FIFO 中 读时：返回 RX-FIFO 中未读的数据 注： TX-FIFO 和 RX-FIFO 的深度均为 1 写数据时，需先写 DR，再写 I2CCMD	I2CDR[7:0]	0x412	RW−0000 0000
RESTART	主机发送 Start / Restart 1 = 发送 0 = 不发送	I2CCMD[2]	0x413	WO−0
STOP	主机发送 Stop (字节传输后，或主机拉伸 SCL 时) 1 = 发送 (发送成功后自动清零) 0 = 不发送	I2CCMD[1]		WO−0
MSTDIR	主机模式，数据传输方向 (读写位 R/W) 1 = 读取 0 = 发送	I2CCMD[0]		WO−0
GCALL	从机模式接收到 General call 标志 1 = Yes (接收且 ACK 后置位) 0 = No 注：检测到 Start/Stop 或 ENABLE = 0 时硬件自动清零；	I2CSR3[5]	0x419	RO−0
RDREQ	从机模式，数据传输方向标志 1 = 发送 (从机接收地址字节的读写位为 1 时置位) 0 = 接收 注：检测到 Start/Stop 或 ENABLE = 0 时硬件自动清零；	I2CSR3[2]		RO−0
ACTIVE	主/从机状态 1 = Busy (繁忙) 0 = IDLE (空闲) 注：从机模式，地址匹配成功后即置位，接收到 Start / Restart / Stop 后清零；	I2CSR3[1]		RO−0
RXHOLD	RX-FIFO 非空保持标志 1 = 非空 (SCL 被拉低，读 DR 后释放) 0 = 空 (SCL 未被拉低)	I2CSR3[0]		RO−0

表 1-1 I2C 相关寄存器

名称	状态		寄存器	地址	复位值
GIE	全局中断	1 = 使能 (PEIE, ITBUFEN, ITEVEN, ITERREN 适用) 0 = 全局关闭 (唤醒不受影响)	INTCON[7]	0xN0B 0xN8B 0x60B 0xF8B (N=0~5)	RW-0
PEIE	外设总中断	1 = 使能 (ITBUFEN, ITEVEN, ITERREN 适用) 0 = 关闭 (无唤醒)	INTCON[6]		RW-0
ITBUFEN	FIFO 状态中断	1 = 使能 (当 IICTXE = 1 或 IICRXNE = 1 时产生中断) 0 = 关闭 (无唤醒)	I2CITR[2]	0x416	RW-0
IICTXE [3]	TX-FIFO 状态	1 = 空 0 = 非空	I2CSR1[7]	0x417	RO-0
IICRXNE [3]	RX-FIFO 状态	1 = 非空 0 = 空	I2CSR1[6]		RO-0
ITEVEN	事件中断	1 = 使能 0 = 关闭 (无唤醒) 事件中断产生条件： SBF = 1 (主机) ADD10F = 1 (主机) ADDF = 1 (主/从机) STOPF = 1 (从机)	I2CITR[1]	0x416	RW-0
STOPF [4]	主/从机模式， Stop 标志	1 = Yes 0 = No	I2CSR1[4]	0x417	RO-0
ADD10F [4]	主机模式，目标从机地址高有效位 MSB 匹配标志	1 = 匹配 (ACK 后置位) 0 = 不匹配，或未发送地址	I2CSR1[3]		RO-0
ADDF [4]	主机模式，目标从机地址低有效位 LSB 匹配标志	1 = 匹配 (ACK 后置位) 0 = 不匹配，或未发送地址	I2CSR1[1]		RO-0
	从机模式，本机地址匹配标志	1 = 匹配或识别到 General Call 0 = 不匹配			
SBF [4]	主机发送 Start 标志	1 = 已发送 0 = 未发送	I2CSR1[0]		RO-0

[3] 写 DR 或 ENABLE = 0 时硬件自动清零。

[4] 读 I2CSR1 或 ENABLE = 0 时硬件自动清零。

名称	状态		寄存器	地址	复位值
ITERREN	错误中断	1 = 使能 0 = 关闭 (无唤醒) 错误中断产生条件： OVR = 1 AF = 1 ARLO = 1 BERR = 1	I2CITR[0]	0x416	RW-0
TXARBT [5]	传输终止标志（发送过程中出错或异常原因导致)	1 = 发生终止 0 = 未发生终止	I2CSR2[4]	0x418	RW0-0
OVR [5]	Overrun 产生标志	1 = Yes 0 = No Overrun 产生条件： TX-over: 当 TX-FIFO 非空时仍写 DR; RX-over: 当 RX-FIFO 非空时仍接收数据; RX-under: 当 RX-FIFO 空时进行读操作;	I2CSR2[3]		RW0-0
AF [5]	主/从机模式，接收应答状态	1 = NACK 0 = ACK	I2CSR2[2]		RW0-0
ARLO [5]	主机模式，总线仲裁失败标志	1 = 仲裁失败 0 = 未发生仲裁失败	I2CSR2[1]		RW0-0
BERR [5]	总线错误（检测到错位的 Start / Stop) 标志	1 = 检测到（字节传输阶段检测到 Start/Stop 时置位) 0 = 未检测到	I2CSR2[0]		RW0-0

表 1-2 I2C 中断使能和状态位

名称	状态	寄存器	地址	复位值
AFP0[0]	I2C_SDA 引脚 1 = PB6 0 = PB3	AFP0[0]	0x19E	RW-0
AFP1[4]	I2C_SCL 引脚 1 = PA2 0 = PB2	AFP1[4]	0x19F	RW-0
I2COD	I2C_SCL，I2C_SDA 引脚开漏输出设置 1 = 使能 0 = 关闭	ODCON0[1]	0x21F	RW-0

表 1-3 I2C 接口引脚控制

[5] 写 0 清零，或 ENABLE = 0 时硬件自动清零。

名称	功能	默认值
I2CRMAP	复用引脚位置 • [I2C_SDA] = PA0，[I2C_SCL] = PA1 [SPI_MOSI] = PB3，[SPI_MISO] = PB2 (≥ I 版本芯片可选) • [I2C_SDA] = PB3，[I2C_SCL] = PB2 [SPI_MOSI] = PA0，[SPI_MISO] = PA1 (<I 版本芯片默认，不可更改)	• [I2C_SDA] = PA0， [I2C_SCL] = PA1， [SPI_MOSI] = PB3， [SPI_MISO] = PB2

表 1-4 I2C 接口初始化相关配置

名称	地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	复位值
PCKEN	0x9A	TKEN	I2CEN	UARTEN	SPIEN	TIM4EN	TIM2EN	TIM1EN	ADCEN	0000 0000
CKOCON	0x95	SYSON	CCORDY	DTYSEL		CCOSEL[2:0]			CCOEN	0010 0000
I2CCR1	0x40C	—	—	—	MST10B	SLV10B	—	SPEED	MASTER	---0 0-00
I2CCR2	0x40D	—	SOFTRST	AGCALL	SNACK	—	—	RXHLD	—	-000 —0-
I2CCR3	0x40E	—					EVSTRE	—	ENABLE	---- -000
I2COARL	0x40F	ADD[7:0]								0000 0000
I2COARH	0x410	—	—	—	—	—	—	ADD[9:8]		---- --00
I2CFREQ	0x411	—	—	FREQ[5:0]						--00 0000
I2CDR	0x412	DR[7:0]								0000 0000
I2CCMD	0x413	—	—	—	—	—	RESTART	STOP	MSTDIR	---- -000
I2CCCRL	0x414	CCR[7:0]								0000 0000
I2CCCRH	0x415	—	DUTY	—	—	CCR[11:8]				-0—0000
I2CITR	0x416	—					ITBUFEN	ITEVEN	ITERREN	---- -000
I2CSR1	0x417	IICTXE	IICRXNE	—	STOPF	ADD10F	—	ADDF	SBF	00-0 0-00
I2CSR2	0x418	—	—	—	TXABRT	OVR	AF	ARLO	BERR	---0 0000
I2CSR3	0x419	—	—	GCALL	—	—	RDREQ	ACTIVE	RXHOLD	--0- -000

表 1-5 I2C 相关寄存器地址

1.2. I2C 配置

主机和从机的 I2C 配置流程基本相同：

1. 使能 I2C 模块时钟（参阅” I2CEN”）；
2. 选择主机或从机模式（参阅” MASTER”）；
3. 设置主从机的时钟频率 Fmaster，需与 SysClk 相同（参阅” FREQ[5:0]”）；
4. 主机的通信速率设置为标准模式或快速模式（参阅” SPEED”）；
5. 主机配置 SCL 占空比及时钟周期（参阅” DUTY”, “CCR[7:0]”和“CCR[11:8]”）；
6. 主从机选择 7 位或 10 位地址格式（参阅” MST10B” 和 “SLV10B”）；
7. 设置主机的数据传输方向为发送或接收（参阅” MSTDIR”），从机则由接收地址字节的读写位控制；
8. 如需要，可选择 General call 模式（参阅” AGCALL”）；
9. 使能 I2C 模块（参阅” ENABLE”）；
10. 如需要，可使能相应的中断（参阅” GIE”, ” PEIE”, ” ITBUFEN” , ” ITEVEN”和” ITERREN”）；

注：

- 当 ENABLE =1 时，引脚 SCL / SDA 接口功能自动使能，SCL / SDA 分别对应引脚图中的 I2C_SCL / I2C_SDA；
- 为了产生正确的时序，I2C 模块的输入时钟 Fmaster 和时钟周期 CCR，必须满足以下设置条件：

	寄存器	标准模式	快速模式（DUTY=0）	快速模式（DUTY=1）
主机和从机	FREQ[5:0]	≥ 2MHz	≥ 8MHz	≥ 8MHz
主机	CCR[11:0]	≥ 9	≥ 9	—

- 如果 I2C 模块因异常原因导致一直处于活动状态(ACTIVE=1)，可以置位 SOFTRST 对发送和接收模块进行复位，对寄存器值无影响；

I2C 通信由主机发起数据传输并产生时钟信号，且由主机控制 Start 和 Stop 信号。串行数据传输以 Start 条件开始并以 Stop 条件结束，数据和地址均按 8 位/byte 进行传输，高位在前。

从机在检测到 Start 条件后，能识别自己的地址(可编程，7 位或 10 位）和 General Call 地址，且具有 Stop 检测功能。

在一个字节传输后的第 9 个时钟期间，接收器需回送一个应答位(ACK)给发送器。

I2C 模块的四种工作模式为：主机发送、主机接收、从机发送、从机接收。

1.2.1. 主机发送

主机发送模式下，输出时钟到 SCL，发送串行数据到 SDA。

7 位地址(MST10B = 0)，主机发送的第 1 个 byte 包括地址和读写位(0)，然后开始发送 8 位串行数据。

10 位地址(MST10B = 1)，主机发送的第 1 个 byte 包括地址头段序列和读写位(0)，第 2 个 byte 为低有效位地址，然后开始发送 8 位串行数据。

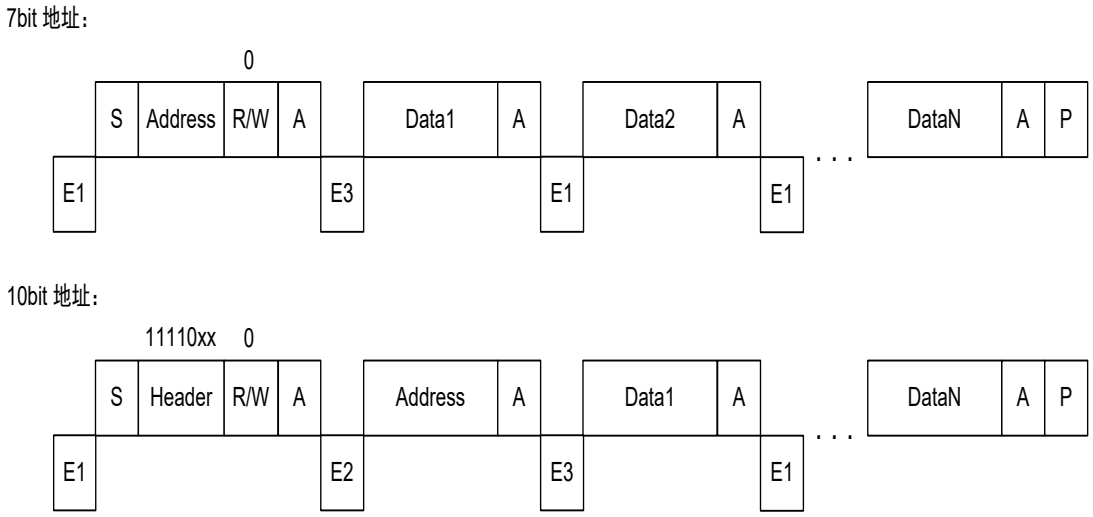

图 1-2 主机发送流程

注：

S：Start 信号；

A：ACK 信号；

P：Stop 信号；

E1：IICTXE=1，TX-FIFO 为空，写 DR 和 I2CCMD 将清零该标志；

E2：ADD10F=1，读 I2CSR1 将清零该标志；

E3：ADDF=1，读 I2CSR1 将清零该标志；

1.2.2. 主机接收

主机接收模式下，输出时钟到 SCL，从 SDA 线上接收串行数据。

7 位地址(MST10B = 0)，主机发送的第 1 个 byte 包括地址和读写位(1)，然后开始接收 8 位串行数据。

10 位地址(MST10B = 1)，主机发送的第 1 个 byte 包括地址头段序列和读写位(0)，第 2 个 byte 为低有效位地址，然后重新发送 Start 信号和地址头段序列和读写位(1)，开始接收 8 位串行数据。

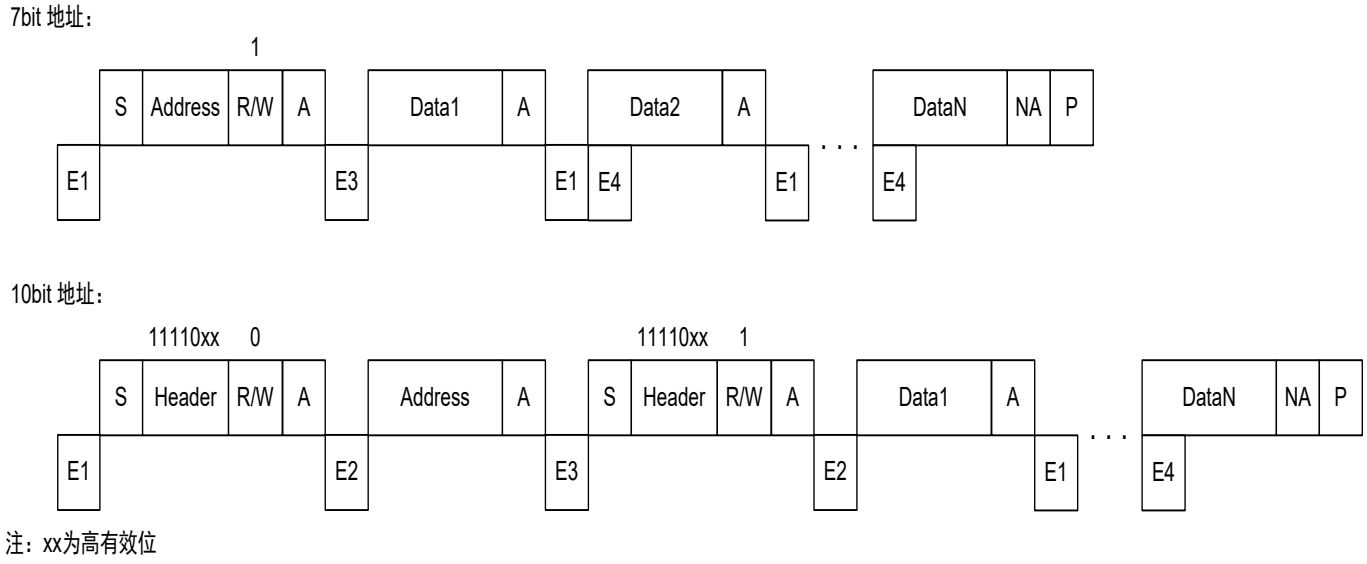

图 1-3 主机接收流程

注：

S：Start 信号；

A：ACK 信号；

P：Stop 信号；

E1：IICTXE=1，TX-FIFO 为空，写 DR 和 I2CCMD 将清零该标志；

E2：ADD10F=1，读 I2CSR1 将清零该标志；

E3：ADDF=1，读 I2CSR1 将清零该标志；

E4：IICRXNE=1，RX-FIFO 非空，读 DR 将清零该标志；

1.2.3. 从机发送

从机发送模式下，发送串行数据到 SDA。

7 位地址(SLV10B = 0)，从机接收的第 1 个 byte 包括地址和读写位(1)，然后开始发送 8 位串行数据。

10 位地址(SLV10B = 1)，从机接收的第 1 个 byte 包括地址头段序列和读写位(0)，第 2 个 byte 为低有效位地址，然后重新检测 Start 信号并接收地址头段序列和读写位(1)，开始发送 8 位串行数据。

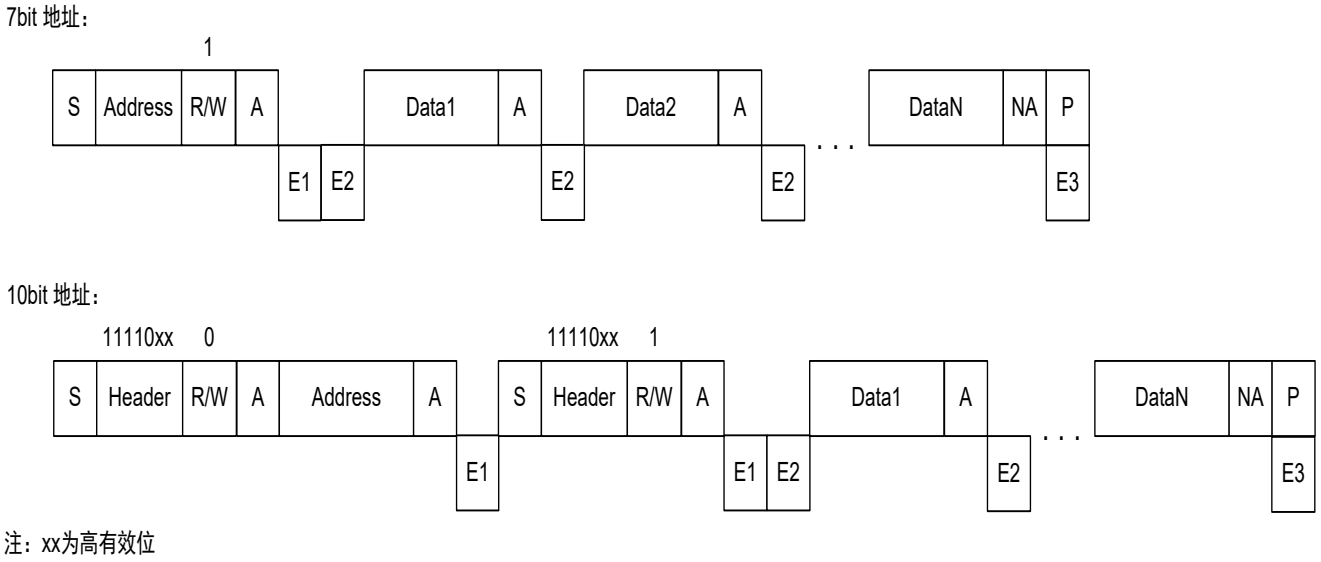

图 1-4 从机发送流程

注：

S：Start 信号；

A：ACK 信号；

P：Stop 信号；

E1：ADDF=1，拉低 SCL 线，读 I2CSR1 将清零该标志；

E2：IICTXE=1，TX-FIFO 为空，拉低 SCL 线，读 RDREQ 为 1，写 DR 和 I2CCMD 将清零该标志；

E3：AF=1，写 0 清零；

1.2.4. 从机接收

从机接收模式下，从 SDA 线上接收串行数据。

7 位地址(SLV10B = 0)，从机接收的第 1 个 byte 包括地址和读写位(0)，然后开始接收 8 位串行数据。

10 位地址(SLV10B = 1)，从机接收的第 1 个 byte 包括地址头段序列和读写位(0)，第 2 个 byte 为低有效位地址，然后开始接收 8 位串行数据。

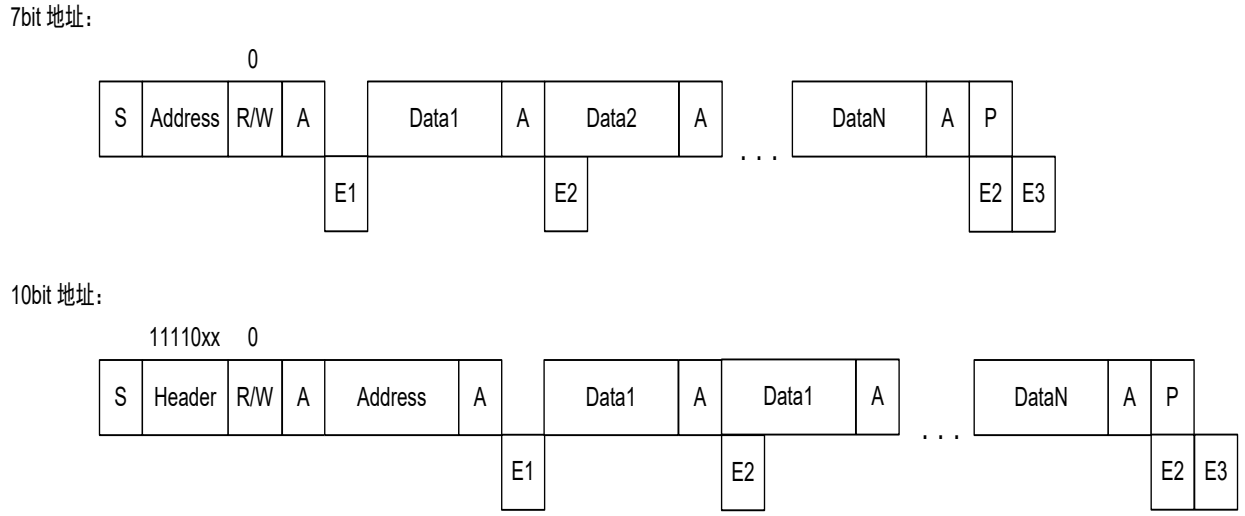

图 1-5 从机接收流程

注：

S：Start 信号；

A：ACK 信号；

P：Stop 信号；

E1：ADDF=1，读 I2CSR1 将清零该标志；

E2：IICRXNE=1，RX-FIFO 非空，读 DR 将清零该标志；

E3：STOPF =1，读 I2CSR1 将清零该标志；

1.2.5. 广播呼叫 (General Call)

主机/从机置位 AGCALL 后使能 General Call 模式：

- 主机向 0x00 地址发送数据，通信流程同主机发送；

从机响应主机发来的 General Call，向 0x00 地址写入数据，通信流程同从机接收；

2. 应用范例

```
//************************************************************
/* 文件名：TEST_62F08x_IIC.c
* 功能：    FT62F08x-IIC 功能演示
*  IC：     FT62F088 LQFP32
* 内部：    16M/2T
* empno：   500
* 说明：    此演示程序位 62F08x_IIC 的演示程序.
*           该程序把 0x55 写入(24C02)0x12 地址,后读 0x12 地址的值，判断是否写入成功
*
* 参考原理图  TEST_62F08x_sch.pdf
*/
//************************************************************
#include "SYSCFG.h"
//**********************宏定义***************************
#define   unchar      unsigned char

volatile    unchar      IICReadData;
/*-----------------------------------------------
 *  函数名：interrupt ISR
 *  功能：   中断处理，包括定时器 0 中断和外部中断
 *  输入：   无
 *  输出：   无
 -----------------------------------------------*/
void interrupt ISR(void)
{

}
/*-----------------------------------------------
 *  函数名：POWER_INITIAL
 *  功能：   上电系统初始化
 *  输入：   无
 *  输出：   无
 -----------------------------------------------*/
void POWER_INITIAL (void)
{
    OSCCON = 0B01110001;        //IRCF=111=16MHz 1:1
    INTCON = 0;                 //暂禁止所有中断

    PORTA = 0B00000000;
    TRISA = 0B00000000;         //PA 输入输出  0-输出  1-输入
    PORTB = 0B00000000;
    TRISB = 0B00000000;         //PB 输入输出  0-输出  1-输入
    PORTC = 0B00000000;
```

```
    TRISC = 0B00000000;         //PC 输入输出  0-输出  1-输入
    PORTD = 0B00000000;
    TRISD = 0B00000000;         //PD 输入输出  0-输出  1-输入

    WPUA = 0B00000000;          //PA 端口上拉控制  1-开上拉  0-关上拉
    WPUB = 0B00000000;          //PB 端口上拉控制  1-开上拉  0-关上拉
    WPUC = 0B00001000;          //PC 端口上拉控制  1-开上拉  0-关上拉
    WPUD = 0B00000000;          //PD 端口上拉控制  1-开上拉  0-关上拉

    WPDA = 0B00000000;          //PA 端口上拉控制  1-开下拉  0-关下拉
    WPDB = 0B00000000;          //PB 端口上拉控制  1-开下拉  0-关下拉
    WPDC = 0B00000000;          //PC 端口上拉控制  1-开下拉  0-关下拉
    WPDD = 0B00000000;          //PD 端口上拉控制  1-开下拉  0-关下拉

    PSRC0 = 0B11111111;         //PORTA,PORTB 源电流设置最大
    PSRC1 = 0B11111111;         //PORTC,PORTD 源电流设置最大

    PSINK0 = 0B11111111;        //PORTA 灌电流设置最大  0:最小，1:最大
    PSINK1 = 0B11111111;        //PORTB 灌电流设置最大  0:最小，1:最大
    PSINK2 = 0B11111111;        //PORTC 灌电流设置最大  0:最小，1:最大
    PSINK3 = 0B11111111;        //PORTD 灌电流设置最大  0:最小，1:最大

    ANSELA = 0B00000000;        //全为数字管脚
}
/*---------------------------------------------
 *   函数名：DelayUs
 *   功能：   短延时函数  --16M-2T--大概快 1%左右.
 *   输入：   Time 延时时间长度  延时时长 Time μs
 *   返回：   无
 ---------------------------------------------*/
void DelayUs(unsigned char Time)
{
    unsigned char a;
    for(a=0;a<Time;a++)
    {
        NOP();
    }
}



/*---------------------------------------------
 *   函数名：DelayMs
```

```
 *   功能：   短延时函数  快 1%
 *   输入：   Time 延时时间长度  延时时长 Time ms
 *   返回：   无
 ------------------------------------------------*/
void DelayMs(unsigned char Time)
{
    unsigned char a,b;
    for(a=0;a<Time;a++)
    {
        for(b=0;b<5;b++)
        {
            DelayUs(197);
        }
    }
}
/*------------------------------------------------
 *   函数名：IIC_READ
 *   功能：   IIC 读出特定位置的数据
 *   输入：   address
 *   输出：   读出 address 存储器里面的数据 iicdata
 ------------------------------------------------*/
 unsigned char IIC_READ(unsigned char address)
{
    unsigned char iicdata = 0;
    while(!IICTXE);
    I2CDR = address;
    I2CCMD= 0B00000110;
    while(!IICTXE);
    I2CCMD= 0B00000011;
    while(!IICRXNE);
    iicdata =I2CDR;
    return iicdata;
}
/*------------------------------------------------
 *   函数名：IIC_WRITE
 *   功能：   IIC 把数据 data 写入特定的位置 address
 *   输入：   address，data
 *   输出：   无
 ------------------------------------------------*/
void IIC_WRITE(unsigned char address,unsigned char data)
{
    while(!IICTXE);
    I2CDR = address;
    I2CCMD= 0B00000000;
```

```
    while(!IICTXE);
    I2CDR = data;
    I2CCMD= 0B00000010;
    while(!IICTXE);
}
/*-----------------------------------------------
 *  函数名：IIC_INITIAL
 *  功能：  初始化 IIC
 *  输入：  无
 *  输出：  无
 -----------------------------------------------*/
void IIC_INITIAL(void)
{
    PCKEN|=0B01000000;          //使能 I2C 模块时钟

    ODCON0|=0B00000010;
    I2CCR1 = 0B00000001;
    //Bit4:主机模式下地址格式
    //0：发送 7 位地址格式；
    //1：发送 10 位地址格式；

    //Bit3:从机模式下地址格式
    //0：响应 7 位地址格式；
    //1：响应 10 位地址格式；

    //Bit2:保留位，读 0

    //Bit1:I2C 通信速度模式
    //0：标准模式（100kHz）；
    //1：快速模式（400kHz）；

    //Bit0:主从机模式
    //0：从机模式；
    //1：主机模式；

    I2CCR2 = 0B00000000;
    I2CCR3 = 0B00000000;

    //Bit0:I2C 模块使能
    //0：禁用 I2C 模块；
    //1：使能 I2C 模块，相应的 IO 管脚会用作 I2C 的功能；

    I2COARL     = 0B01010000;   //从器件地址 A0
    I2COARH     = 0B00000000;   //从机地址高位
```

```
    I2CFREQ     = 0B00010000;    //外设时钟 16M

    I2CCCRL     = 0B10000000;    //SCL 周期=2*CCR*Fmaster
    I2CCCRH     = 0B00000000;
    I2CITR = 0B00000000;          //不使能 IIC 中断

    ENABLE=1;                     //使能 I2C
 }
/*-----------------------------------------------
 *   函数名：main
 *   功能：   主函数
 *  输入：   无
 *  输出：   无
 -----------------------------------------------*/
void main(void)
{
    POWER_INITIAL();              //系统初始化
    IIC_INITIAL();

    IIC_WRITE(0x12,0x55);         //0x55 写入地址 0x12
    DelayMs(10);
    IICReadData = IIC_READ(0x12);    //读取 0x12 地址 EEPROM 值

    while(1)
    {
      NOP();
    }
}
```

联系信息

Fremont Micro Devices Corporation

#5−8, 10/F, Changhong Building
Ke-Ji Nan 12 Road, Nanshan District,
Shenzhen, Guangdong, PRC 518057

Tel: (+86 755) 8611 7811
Fax: (+86 755) 8611 7810

Fremont Micro Devices (HK) Limited

#16, 16/F, Block B, Veristrong Industrial Centre,
34−36 Au Pui Wan Street, Fotan, Shatin, Hong Kong SAR

Tel: (+852) 2781 1186
Fax: (+852) 2781 1144

http://www.fremontmicro.com

* Information furnished is believed to be accurate and reliable. However, Fremont Micro Devices Corporation assumes no responsibility for the consequences of use of such information or for any infringement of patents of other rights of third parties, which may result from its use. No license is granted by implication or otherwise under any patent rights of Fremont Micro Devices Corporation. Specifications mentioned in this publication are subject to change without notice. This publication supersedes and replaces all information previously supplied. Fremont Micro Devices Corporation products are not authorized for use as critical components in life support devices or systems without express written approval of Fremont Micro Devices Corporation. The FMD logo is a registered trademark of Fremont Micro Devices Corporation. All other names are the property of their respective owners.