FT64F0AX

IIC Application note

目录

FT64F0Ax 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：

- 主机模式和从机模式
- 多主机支持
- 标准模式 (100kHz) 和快速模式 (400kHz)
- 7 位和 10 位地址模式
- General call 支持
- Clock stretching
- 发送 NACK (从机模式)

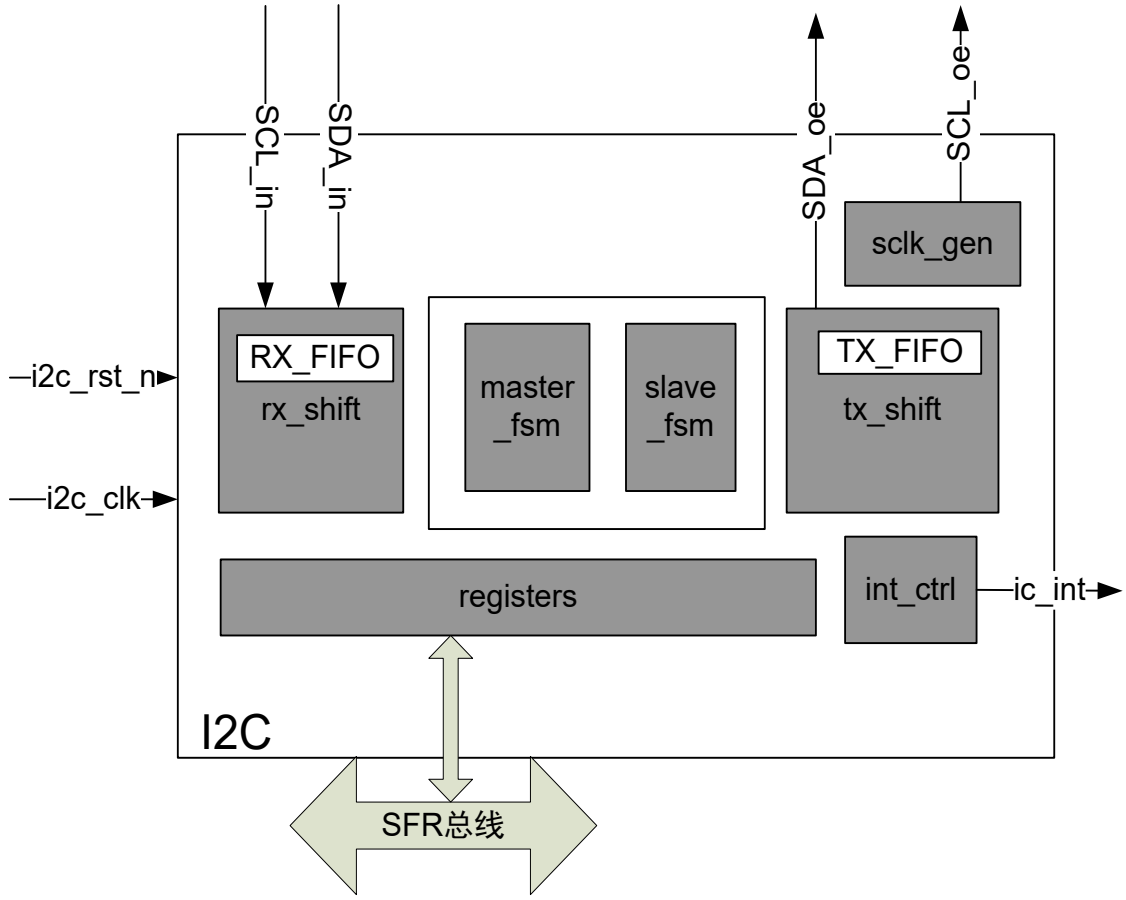

图 1-1 I2C 结构框图

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]	Bank 首地址 +0x0B	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) ADDRF = 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
ADDRF [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 = PB4	AFP0[0]	0x19E	RW-0
AFP0[5]	I2C_SCL 引脚 1 = PA2 0 = PB3	AFP0[5]	0x19E	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	地址	bit7	bit6	bit5	bit4	bit3	bit2	bit1	bit0	复位值
PCKEN	0x9A	UART2EN	I2CEN	UART1EN	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	---- -0-0
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	—	ADDRF	SBF	00-0 0-00
I2CSR2	0x418	—	—	—	TXABRT	OVR	AF	ARLO	BERR	---0 0000
I2CSR3	0x419	—	—	GCALL	—	—	RDREQ	ACTIVE	RXHOLD	--0- -000

表 1-4 I2C 相关寄存器地址

1.2. I2C 的工作原理

I2C 模块主要有四种工作模式，即主机接收、主机发送、从机发送、从机接收。每种模式下又包含了 7 位地址模式和 10 位地址格式。

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。

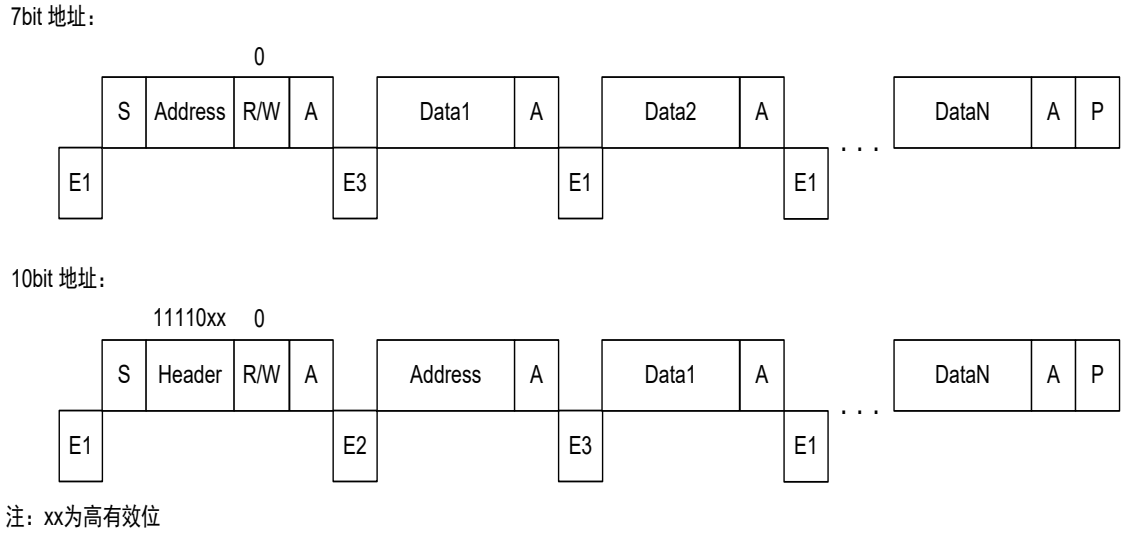

图 1-2 主机发送流程

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：ADDRF=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。

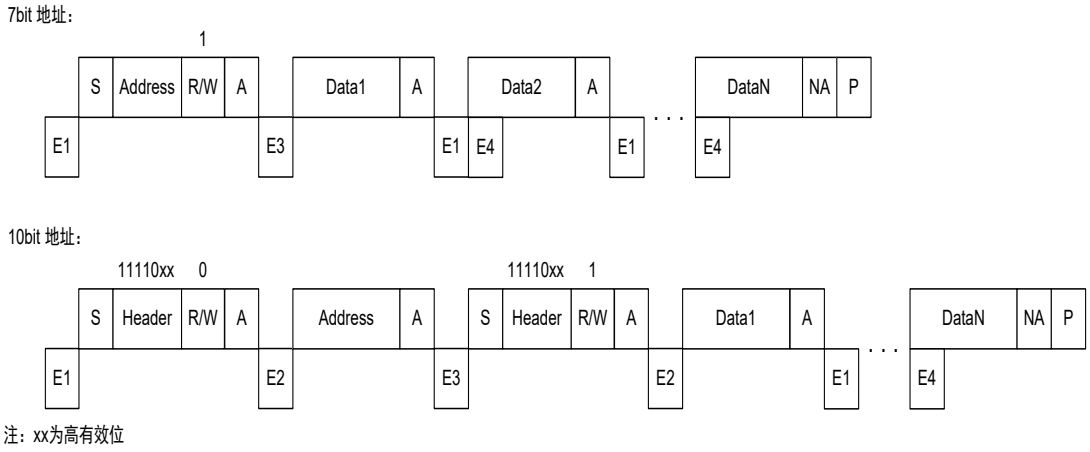

图 1-3 主机接收流程

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：ADDRF=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。

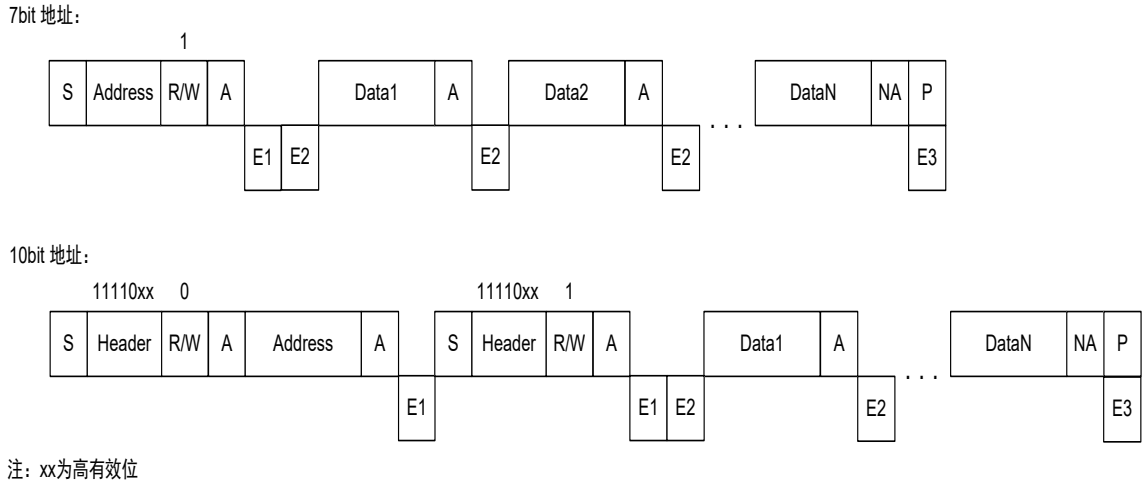

图 1-4 从机发送流程

注：

S：Start 信号；

A：ACK 信号；

P：Stop 信号；

E1：ADDRF=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。

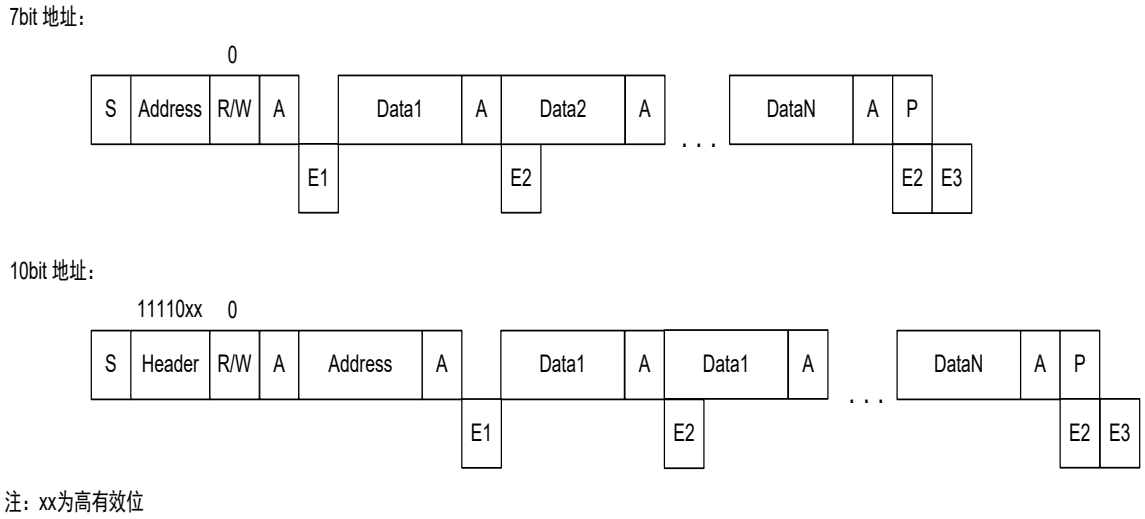

图 1-5 从机接收流程

注：

S：Start 信号；

A：ACK 信号；

P：Stop 信号；

E1：ADDRF=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)

General Call 模式在主机置位了 AGCALL 以后，就会向地址为 0x00 的地址发送数据，这种模式下主机只允许进行写数据，不允许读数据；从机模式下在置位了 AGCALL 以后，就会响应主机发来的 General Call；通信的过程跟主机发送，从机接收模式相同。

2. 应用范例

```
//*************************************************************
/* 文件名：TEST_64F0Ax_IIC.c
* 功能：    FT64F0Ax_IIC 功能演示
*  IC：     FT64F0A5 TSSOP20
* 内部：    16M/2T
* 说明：    此演示程序为 64F0Ax_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
*
*           FT64F0A5   TSSOP20
*              -------------------------
* NC----------|1(PA5)        (PA4)20|----------NC
* NC----------|2(PA6)        (PA3)19|----------NC
* NC----------|3(PA7)        (PA2)18|----------NC
* NC----------|4(PC0)        (PA1)17|----------NC
* NC----------|5(PC1)        (PA0)16|----------NC
* NC----------|6(PB7)        (PB0)15|----------NC
* GND---------|7(GND)        (PB1)14|----------NC
* NC----------|8(PB6)        (PB2)13|----------NC
* VDD---------|9(VDD)        (PB3)12|----IIC_SCL
* NC----------|10(PB5)       (PB4)11|----IIC_SDA
*              -------------------------
*/
//*************************************************************
#include      "SYSCFG.h";
#include      "FT64F0AX.h";
//**********************宏定义*****************************
#define  uchar    unsigned char

volatile  uchar    IICReadData;
/*-----------------------------------------------
 * 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;          //系统时钟选择为内部振荡器 16MHz,分频比为 1:1

     INTCON=0;                   //禁止所有中断

     PORTA=0B00000000;
     PORTB=0B00000000;
```

```
    PORTC=0B00000000;

    WPUA=0B00000000;          //弱上拉的开关，0-关，1-开
    WPUB=0B00000000;
    WPUC=0B00000000;

    WPDA=0B00000000;          //弱下拉的开关，0-关，1-开
    WPDB=0B00000000;
    WPDC=0B00000000;

    TRISA=0B00000000;         //输入输出设置，0-输出，1-输入
    TRISB=0B00000000;         //PB4:IIC-SDA   PB3:IIC-SCL
    TRISC=0B00000000;

    PSRC0=0B11111111;         //源电流设置最大
    PSRC1=0B11111111;
    PSRC2=0B00001111;

    PSINK0=0B11111111;        //灌电流设置最大
    PSINK1=0B11111111;
    PSINK2=0B00000011;

    ANSELA=0B00000000;        //设置对应的 IO 为数字 IO
}
/*---------------------------------------------
* 函数名：SPI_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;       //I2C_SCL,I2C_SDA 的开漏输出设置，高有效

    I2CCR1=0B00000001;        //主机模式，标准模式（100kHz），7 位地址格式
    I2CCR2=0B00000000;
    I2CCR3=0B00000000;        //禁用 I2C 模块，才能写目标从机地址，设置外设时钟频率以及 SCL
时钟周期
    I2COARL=0B01010000;       //从机地址
    I2COARH=0B00000000;
    I2CFREQ=0B00010000;       //外设时钟频率为 16MHz
    I2CCCRL=0B00000000;       //标准模式下，SCL 时钟周期为 2*CCR*Fmaster（Fmaster 为外设
时钟频率，此处为 2*128*16M）
```

```
    I2CCCRH=0B00000000;
    I2CITR=0B00000000;          //禁用所有中断
    ENABLE=1;                   //启用 I2C
}
/*-------------------------------------------------
* 函数名：DelayUs
* 功能：   短延时函数
* 输入：   Time 延时时间长度  延时时长 Time μs
* 输出：   无
-------------------------------------------------*/
void DelayUs(uchar Time)
{
    uchar a;
    for(a=0;a<Time;a++)
    {
        NOP();
    }
}
/*-------------------------------------------------
* 函数名：DelayMs
* 功能：   短延时函数
* 输入：   Time 延时时间长度  延时时长 Time ms
* 输出：   无
-------------------------------------------------*/
void DelayMs(uchar Time)
{
    uchar a,b;
    for(a=0;a<Time;a++)
    {
        for(b=0;b<5;b++)
        {
            DelayUs(197);
        }
    }
}
/*-------------------------------------------------
* 函数名：DelayS
* 功能：   短延时函数
* 输入：   Time 延时时间长度  延时时长 Time S
* 输出：   无
-------------------------------------------------*/
void DelayS(uchar Time)
{
    uchar a,b;
```

```
    for(a=0;a<Time;a++)
    {
        for(b=0;b<10;b++)
        {
            DelayMs(100);
        }
    }
}
/*-------------------------------------------------
 * 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
 -------------------------------------------------*/
uchar IIC_Read(uchar address)
{
    uchar 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(uchar address , uchar data)
{
    while(!IICTXE);
    I2CDR = address;
    I2CCMD= 0B00000000;
    while(!IICTXE);
    I2CDR = data;
    I2CCMD= 0B00000010;
    while(!IICTXE);
}

/*-------------------------------------------------
```

```
 * 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 初始化

    IIC_Write(0x12,0x55);         //0X55X 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