FT62F08X

Application note

目录

FT62F08X IIC 应用

1. I2C 接口

I2C 模块通过 SDA 和 SCL 管脚与外部 I2C 器件进行通信，数据以字节格式按照先传高位的原则进行传输。

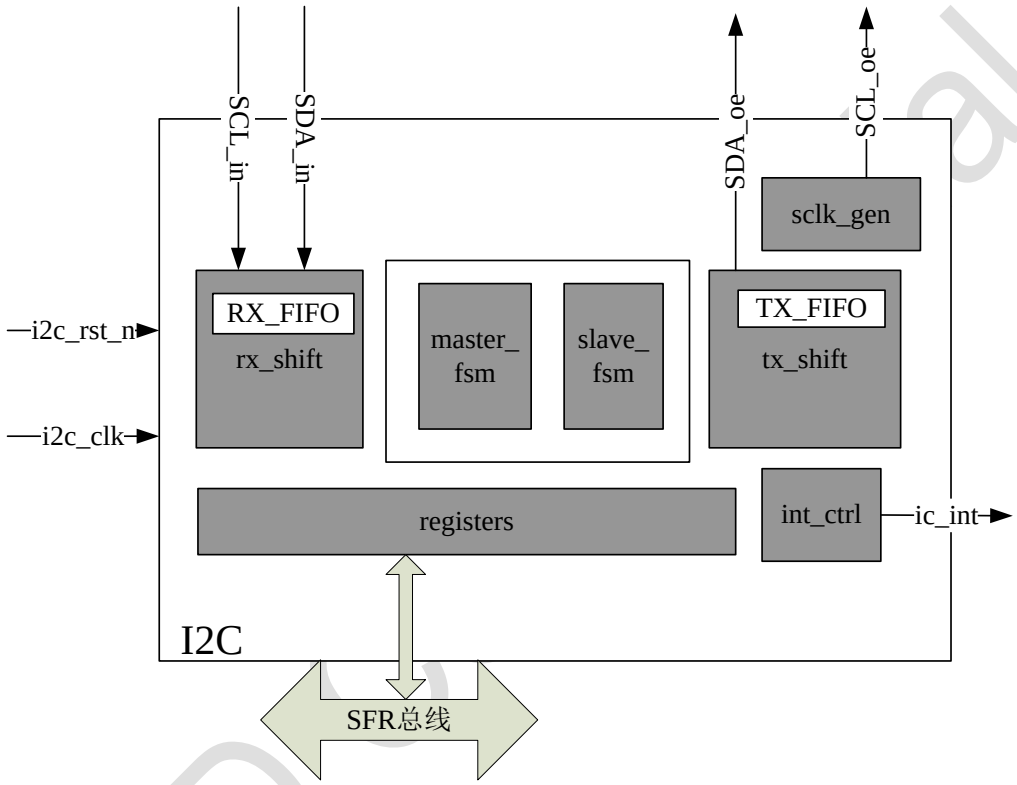

图 14.1 I2C 原理框图

I2C 模块包含以下功能：

- 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）

1.1. 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。

1.1.1. 主机发送

主机发送模式下，输出串行数据到 SDA，输出时钟到 SCL。主机发送的第一个字节包括从机地址和读写位，此模式下读写位为 0。然后主机发送 8 位的串行数据，每个数据字节后会接收到 ACK。同时，主机也会产生 Start 和 Stop。

当 I2CCR1 寄存器中 MST10B 位为 0 时，主机发送 7 位地址格式：

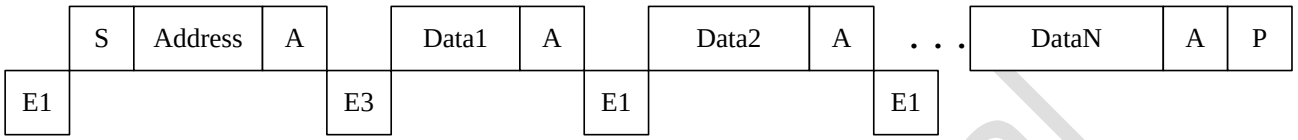

图 14.2 7 比特地址模式主机发送流程图

其中：

E1：TXE=1，写 DR 和 CMD 寄存器清零 TXE；

E3：ADDF=1，写零到 ADDF 清零 ADDF；

S：表示 START 信号；

A：表示 ACK 信号；

P：表示 STOP 信号；

当 I2CCR1 寄存器中 MST10B 位为 1 时，主机发送 10 位地址格式：

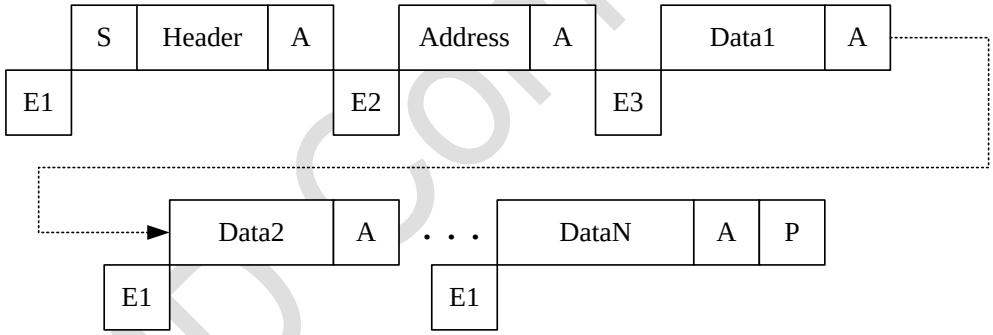

图 14.3 10 比特地址模式主机发送流程图

其中：

E1：TXE=1，写 DR 和 CMD 寄存器清零 TXE；

E2：ADD10F=1，写零到 ADD10F 清零 ADD10F；

E3：ADDF=1，写零到 ADDF 清零 ADD；

注意：主机发送模式下，软件可以不处理 SBF/ADDF/ADD10F 标志位，只关心 TXE。

1.1.2. 主机接收

主机接收模式下，从 SDA 线上接收串行数据，输出时钟到 SCL。主机首先发送从机地址和读写位，此模式下读写位为 1。然后主机接收 8 位的串行数据，每个数据字节后主机需要发送 ACK。同时，主机也会产生 Start 和 Stop。

当 I2CCR1 寄存器中 MST10B 位为 0 时，主机发送 7 位地址格式：

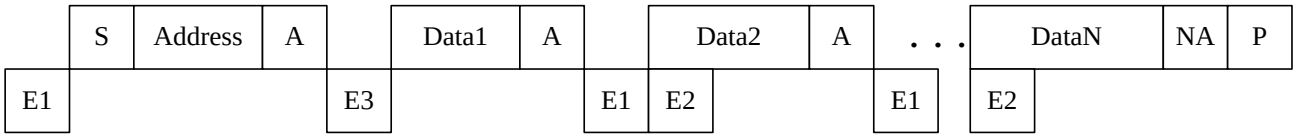

图 14.4 7 比特地址模式主机接收流程图

其中：

E1：TXE=1，写 DR 和 CMD 寄存器清零 TXE；

E3：ADDF=1，写零到 ADDF 清零 ADDF；

E2：RXNE=1，读 DR 寄存器清零 RXNE；

当 I2CCR1 寄存器中 MST10B 位为 1 时，主机发送 10 位地址格式：

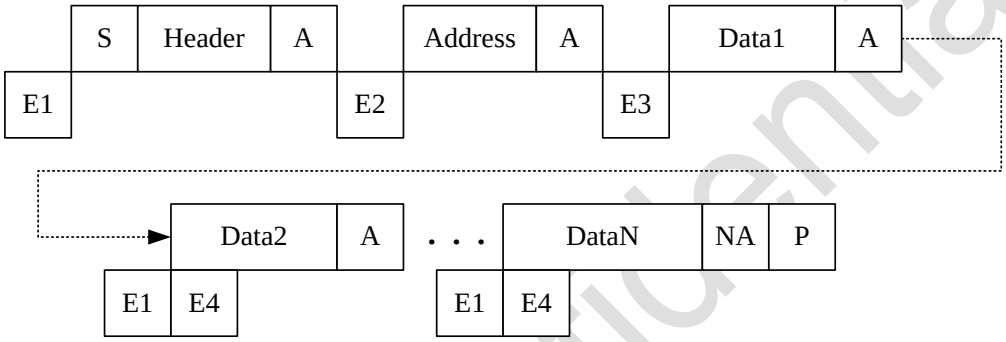

图 14.5 10 比特地址模式主机接收流程图

其中：

E1：TXE=1，写 DR 和 CMD 寄存器清零 TXE；

E2：ADD10F=1，写 1 到 ADD10F 清零 ADD10F；

E3：ADDF=1，写零到 ADDF 清零 ADDF；

E4：RXNE=1，读 DR 寄存器清零 RXNE；

注意：主机接收模式下，软件可以不处理 SBF/ADDF/ADD10F 标志位

1.1.3. 从机发送

从机发送模式下，发送串行数据到 SDA。从机在检测到 Start 条件后，首先接收地址字节和读写位，此模式下读写位应为 1。然后发送 8 位的数据字节，每个数据字节后会接收 ACK。同时如果检测到 Stop 条件，从机会结束通信，等待下一次 Start。

当 I2CCR1 寄存器中 SLV10B 位为 0 时，从机响应 7 位地址格式：

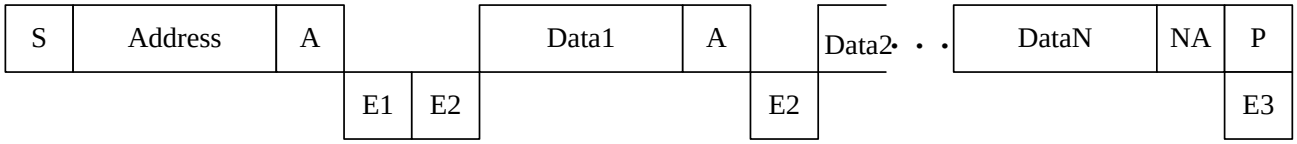

图 14.6 7 比特地址模式从机发送流程图

E1：ADDR=1,拉低 SCL 线，写零到 ADDR 清零 ADDRF；

E2：TXE=1，拉低 SCL 线，读 SR3 寄存器 rd_req 位为 1，需向 DR 寄存器中写数据清零 TXE；
E3：AF=1，写 I2CSR2 寄存器中 AF 位为 0 清零；

当 I2CCR1 寄存器中 SLV10B 位为 1 时，从机响应 10 位地址格式：

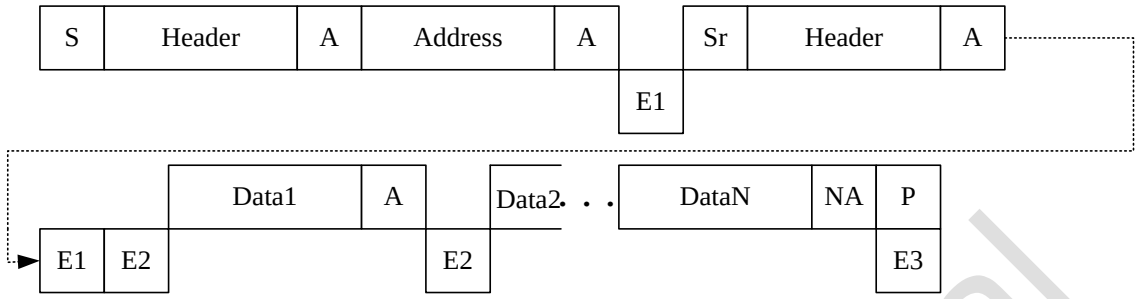

图 14.7 10 比特地址模式从机发送流程图

E1：ADDRF=1，拉低 SCL 线，写零到 ADDRF 清零；
E2：TXE=1，拉低 SCL 线，读 SR3 寄存器 rd_req 位为 1，需向 DR 寄存器中写数据清零 TXE；
E3：AF=1，写 SR2 寄存器中 AF 位为 0 清零；

1.1.4. 从机接收

从机接收模式下，从 SDA 线接收串行数据。从机在检测到 Start 条件后，首先接收地址字节和读写位，此模式下读写位应为 0。然后接收 8 位的数据字节，每个数据字节后需要发送 ACK。同时如果检测到 Stop 条件，从机会结束通信，等待下一次 Start。

当 I2CCR1 寄存器中 SLV10B 位为 0 时，从机响应 7 位地址格式：

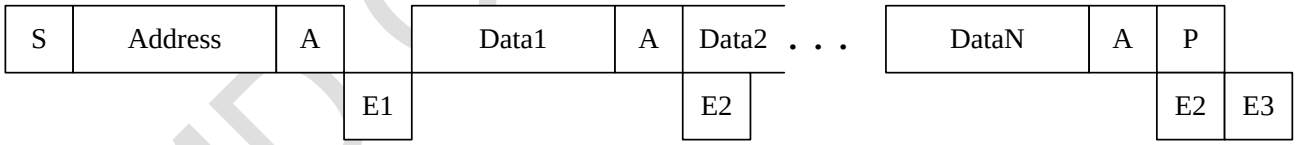

图 14.7 7 比特地址模式从机接收流程图

E1：ADDRF=1，写零到 ADDRF 清零；
E2：RXNE=1，读取 DR 寄存器中数据清零 RXNE；
E3：STOPF=1，写零到 STOPF 清零；

当 I2CCR1 寄存器中 SLV10B 位为 1 时，从机响应 10 位地址格式：

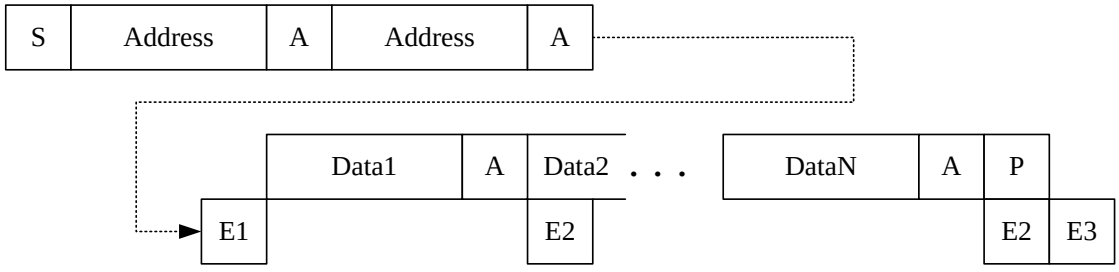

图 14.8 10 比特地址模式从机接收流程图

E1：ADDRF=1，写零到 ADDRF 清零；

E2：RXNE=1，读取 DR 寄存器中数据清零 RXNE；
E3：STOPF=1，写零到 STOPF 清零 STOPF；

1.1.5. 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。

1.2. 与 I2C 相关寄存器汇总

名称	地址	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0	复位值
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	—	ADDRF	SBF	00-0 0-00
I2CSR2	0x418	—	—	—	TXABRT	OVR	AF	ARLO	BERR	--00 0000
I2CSR3	0x419	—	—	GCALL	—	—	RDREQ	ACTIVE	RXHOLD	--0- -000

1.2.1. I2CCR1 寄存器，地址 0x40C

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	MST10B	SLV10B	—	SPEED	MASTER
Reset	—	—	—	0	0	—	0	0
Type	RO-0	RO-0	RO-0	RW	RW	RO-0	RW	RW

Bit	Name	Function
7:5	N/A	保留位，读 0
4	MST10B	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；
3	SLV10B	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；
2	N/A	保留位，读 0
1	SPEED	I2C 通信速度模式 0：标准模式（100KHz）； 1：快速模式（400KHz）；
0	MASTER	主从机模式 0：从机模式； 1：主机模式；

1.2.2. I2CCR2 寄存器，地址 0x40D

Bit	7	6	5	4	3	2	1	0
Name	—	SOFTRST	ACKGCALL	SLVNACK	—	—	RXHLD	—
Reset	—	0	0	0	—	—	0	—
Type	RO-0	RW	RW	RW	RO-0	RO-0	RW	RO-0

Bit	Name	Function
7	N/A	保留位，读 0
6	SOFTRST	软件复位 0：没有影响； 1：复位 I2C 控制模块； 注意：该复位不会复位寄存器的值
5	AGCALL	从机时，应答 General call 使能 0：不响应 General call； 1：响应 General call； 主机时，发送 General call 使能 0：发送正常的从机地址； 1：发送 General call 地址(0x00)；
4	SNACK	从机接收是否发送 NACK 0：字节接收后发送 ACK（地址匹配或数据字节）； 1：发送 NACK；
3:2	N/A	保留位，读 0
1	RXHLD	RX-FIFO 满控制位 0：RX-FIFO 满时不拉低 SCL，新接收的数据将会丢失； 1：RX-FIFO 满时拉低 SCL；
0	N/A	保留位，读 0

1.2.3. I2CCR3 寄存器，地址 0x40E

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	EVSTRE	—	ENABLE
Reset	—	—	—	—	—	0	—	0
Type	RO-0	RO-0	RO-0	RO-0	RO-0	RW	RO-0	RW

Bit	Name	Function
7:1	N/A	保留位，读 0
2	EVSTRE	SBF/ADD/ADD10 拉低 SCL 使能 0：SBF/ADD/ADD10 不拉低 SCL 1：SBF/ADD/ADD10 拉低 SCL
1	N/A	保留位，读 0
0	ENABLE	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；

1.2.4. I2COARL 寄存器，地址 0x40F

Bit	7	6	5	4	3	2	1	0
Name	ADD[7:0]							
Reset	0x00							
Type	RW							

Bit	Name	Function
7:0	ADD[7:0]	从机地址 7 位地址格式时：bit[7] 不关心； 10 位地址格式时：10 位地址的低 8 位； 注：如果模块用作主机，则该寄存器存储的是目标从机的地址，而在用作从机时表示的是本机的地址

1.2.5. I2COARH 寄存器，地址 0x410

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	—	ADD[9:8]	
Reset	—	—	—	—	—	—	2’b00	

Type	RO-0	RO-0	RO-0	RO-0	RO-0	RO-0	RW

Bit	Name	Function
7:2	N/A	保留位，读 0
1:0	ADD[9:8]	从机地址：10 位地址的高 2 位； 注：如果模块用作主机，则该寄存器存储的是目标从机的地址，而在用作从机时表示的是本机的地址

1.2.6. I2CFREQ 寄存器，地址 0x411

Bit	7	6	5	4	3	2	1	0
Name	—	—	FREQ[5:0]					
Reset	—	—	6’h0					
Type	RO-0	RO-0	RW					

Bit	Name	Function
7:6	N/A	保留位，读 0
5:0	FREQ[5:0]	外设时钟频率 6’b000000：不允许； 6’b000001：1MHz； 6’b000010：2MHz； … 6’b011000：24MHz； Higher values：不允许；

1.2.7. I2CDR 寄存器，地址 0x412

Bit	7	6	5	4	3	2	1	0
Name	DR[7:0]							
Reset	0x00							
Type	RW							

Bit	Name	Function
7:0	DR[7:0]	数据寄存器 写时：将该数据 push 到 TX-FIFO 中； 读时：读出 RX-FIFO 中数据； 注意： 主机模式下写数据时，push 动作是在写完 I2CCMD 寄存器之后，因此需要先写 I2CDR 寄存器，再写 I2CCMD 寄存器；发送和接收的 FIFO 深度都为 1

1.2.8. I2CCMD 寄存器，地址 0x413

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	RESTART	STOP	MSTDIR
Reset	—	—	—	—	—	0	0	0
Type	RO-0	RO-0	RO-0	RO-0	RO-0	W	W	W

Bit	Name	Function
7:3	N/A	保留位，读 0
2	RESTART	发送 Restart 0：字节传输后不发送 Restart； 1：字节传输后发送 Restart；
1	STOP	发送 Stop 0：字节传输之后不发送 Stop； 1：字节传输之后发送 Stop；
0	MSTDIR	主机模式时的传输方向 0：发送数据； 1：读取数据； **注意：请不要对该寄存器进行单比特操作，建议使用字节传输的指令对该寄存器进行操作，因为任何对该寄存器的写操作（无论是单比特还是字节操作）都会立即把这个数据写到 FIFO 中**

1.2.9. I2CCCRL 寄存器，地址 0x414

Bit	7	6	5	4	3	2	1	0
Name	CCR[7:0]							
Reset	0x00							
Type	RW							

Bit	Name	Function
7:0	CCR[7:0]	主机模式时 SCL 时钟周期的低 8 位；

1.2.10. I2CCCRH 寄存器，地址 0x415

Bit	7	6	5	4	3	2	1	0
Name	—	DUTY	—	—	CCR[11:8]			
Reset	—	0	—	—	4’h0			

Type	RO-0	RW	RO-0	RO-0	RW

Bit	Name	Function
7	N/A	保留位，读 0
6	DUTY	快速模式下占空比选择 0： Tlow/Thigh = 2； 1： Tlow/Thigh = 16/9； **注意：标准模式下 Tlow/Thigh = 1；**
5:4	N/A	保留位，读 0
3:0	CCR[11:8]	主机模式时 SCL 时钟周期的高 4 位 下表为具体的 SCL 时钟周期公式： 其中，Fmaster 为外设时钟频率；

	周期	SCLH	SCLL
标准模式	2*CCR*Fmaster	CCR*Fmaster	CCR*Fmaster
快速模式（DUTY=0）	3*CCR*Fmaster	CCR*Fmaster	2*CCR*Fmaster
快速模式（DUTY=1）	25*CCR*Fmaster	16*CCR*Fmaster	9*CCR*Fmaster

1.2.11. I2CITR 寄存器，地址 0x416

Bit	7	6	5	4	3	2	1	0
Name	—	—	—	—	—	ITBUFEN	ITEVEN	ITERREN
Reset	—	—	—	—	—	0	0	0
Type	RO-0	RO-0	RO-0	RO-0	RO-0	RO	RO	RO

Bit	Name	Function
7:3	N/A	保留位，读 0
2	ITBUFEN	FIFO 状态中断使能 0：TXE=1 或 RXNE=1 不产生中断； 1：TXE=1 或 RXNE=1 产生中断；
1	ITEVEN	事件中断使能 0：Disabled 事件中断使能； 1：使能事件中断； 事件中断产生条件： SB=1（master） ADDR=1（master/slave） ADD10=1（master） STOPF=1（slave）
0	ITERREN	错误中断使能 0：Disable 错误中断； 1：使能错误中断； 错误中断产生条件：

		BERR=1 ARLO=1 AF=1 OVR=1

1.2.12. I2CSR1 寄存器，地址 0x417

Bit	7	6	5	4	3	2	1	0
Name	IICTXE	IICRXNE	—	STOPF	ADD10F	—	ADDRF	SBF
Reset	0	0	—	0	0	—	0	0
Type	RO	RO	RO-0	RO	RO	RO-0	RO	RO

Bit	Name	Function
7	IICTXE	TX-FIFO 空 0：TX-FIFO 非空； 1：TX-FIFO 空； --发送条件下 TX-FIFO 空时置位； --软件写 I2CDR 寄存器，或者 disable i2c 时硬件清零； **注意：该寄存器位在数据传输过程中一直是 0，在当前数据传输完成后会置 1，因为 FIFO 的深度为 1，则推荐在该标志位为 1 时在向 FIFO 中写入数据，否则会导致数据写入失败，置位 OVR 标志位**
6	IICRXNE	RX-FIFO 非空 0：RX-FIFO 为空； 1：RX-FIFO 非空； --接收条件下 RX-FIFO 非空时置位； --软件读 I2CDR 寄存器，或者 disable i2c 时硬件清零；
5	N/A	保留位，读 0
4	STOPF	从机模式下 Stop 检测 0：没有检测到 Stop； 1：检测到 Stop； --从机模式时 ACK 之后检测到 Stop 时置位； --软件读 I2CSR1 寄存器，或者 disable i2c 时硬件清零；
3	ADD10F	主机模式下发送 10 位地址 Header； 0：没有发送 10 位地址 Header； 1：主机发送 10 位地址 Header； --主机发送 10 位地址 Header 时置位（ACK 之后）； --软件读 I2CSR1 寄存器，或者 disable i2c 时硬件清零；
2	N/A	保留位，读 0
1	ADDF	地址发送（主机）/地址匹配（从机）： 软件读 I2CSR1 寄存器，或者 disable i2c 时硬件清零； --地址匹配（从机）

		0：接收地址不匹配； 1：接收地址匹配； --接收地址匹配或者识别 General Call 后置位； --地址发送（主机）： 0：地址传输没有完成； 1：地址传输完成； --对于 10 位地址：第二个地址字节之后置位（ACK 后）； --对于 7 位地址：地址字节之后置位（ACK 后）； **注意：NACK 后不会置位 ADDF；**
0	SBF	主机模式下 Start 产生 0：没有发送 Start； 1：发送 Start； --主机模式时发送 Start 置位； --软件读 I2CSR1 寄存器，或者 disable i2c 时硬件清零；

1.2.13. I2CSR2 寄存器，地址 0x418

Bit	7	6	5	4	3	2	1	0
Name	—				OVR	AF	ARLO	BEFF
Reset	—				0	0	0	0
Type	RO-0				Rc-W0	Rc-W0	Rc-W0	Rc-W0

Bit	Name	Function
7:3	N/A	保留位，读 0
4	TXABRT	发送过程中由于出错或异常原因导致发送终止，写零清零或者 disable i2c 时硬件清零； 0：传输未发生终止 1：传输发生终止
3	OVR	Overrun 产生 0：没有 Overrun； 1：产生 Overrun； --以下条件时置位： tx-over：当 TX-FIFO 中有数据时仍写 I2CDR 寄存器； rx_over：RX-FIFO 中有数据时仍接收数据； rx_under：RX-FIFO 空时进行读操作； --软件该位写 0，或者 disable i2c 时硬件清零；
2	AF	无 ACK： 0：ACK 正常； 1：NACK 产生； --当产生 NACK 时硬件置位； --软件该位写 0，或者 disable i2c 时硬件清零；
1	ARLO	主机仲裁失败

		0：无仲裁失败产生； 1：产生仲裁失败； --主机仲裁失败时置位； --软件该位写 0，或者 disable i2c 时硬件清零；
0	BERR	总线错误 0：没有检测到错位的 Start/Stop； 1：检测到错位的 Start/Stop； --字节传输阶段检测到 Start/Stop 时置位； --软件该位写 0，或者 disable i2c 时硬件清零；

1.2.14. I2CSR3 寄存器，地址 0x419

Bit	7	6	5	4	3	2	1	0
Name	—	—	GCALL	—	—	RDREQ	ACTIVE	RXHOLD
Reset	—	—	0	—	—	0	0	0
Type	RO-0	RO-0	RO	RO-0	RO-0	RO	RO	RO

Bit	Name	Function
7:6	N/A	保留位，读 0
5	GCALL	从机模式接收到 General call --从机模式接收并且 ACK General call 时置位； --检测到 Start/Stop，或者 disable i2c 时硬件清零；
4:3	N/A	保留位，读 0
2	RQREQ	从机模式读请求 0：从机接收数据； 1：从机发送数据； --从机接收地址字节的读写位为 1 时置位； --检测到 Start/Stop，或者 disable i2c 时硬件清零；
1	ACTIVE	主从状态机状态 0：主从状态机处于 IDLE 状态，总线处于空闲状态； 1：主从状态机处于 Busy 状态； --主从状态机处于 IDLE 状态时置位； --主从状态机处于 Busy 状态时清零；
0	RXHOLD	接收满保持状态，此时 scl 被拉低，在读取数据寄存器后释放 0：接收未满，未出现 scl 被拉低的状态 1：接收 fifo 满，scl 被拉低状态

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
*    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
#define   unint         unsigned int
#define   unlong        unsigned long

#define   DemoPortOut  RB3
#define   DemoPortIn          RC3

//volatile    unchar mydata; //全局查看变量定义

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)                    //PIC_HI-TECH 使用
{

}


/*-------------------------------------------------
 *   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;     //WDT 32KHZ IRCF=111=16MHZ
                             //Bit0=1,系统时钟为内部振荡器
                             //Bit0=0,时钟源由 FOSC<2：0>决定即编译选项时选择
    INTCON = 0;              //暂禁止所有中断

    PORTA = 0B00000000;
    TRISA = 0B11111111;      //PA 输入输出  0-输出  1-输入
    PORTB = 0B00001100;
    TRISB = 0B11110011;      //PB 输入输出  0-输出  1-输入
    PORTC = 0B00000000;
    TRISC = 0B11111111;      //PC 输入输出  0-输出  1-输入
    PORTD = 0B00000000;
    TRISD = 0B11111111;      //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
    //BIT7~BIT6:PORTB[7:4]源电流能力控制,BIT5~BIT4:PORTB[3:0]源电流能力控制
    //BIT3~BIT2:PORTA[7:4]源电流能力控制,BIT1~BIT0:PORTA[3:0]源电流能力控制

    PSRC1   = 0B11111111;    //PORTC,PORTD 源电流设置最大
    //BIT7~BIT6:PORTD[7:4]源电流能力控制,BIT5~BIT4:PORTD[3:0]源电流能力控制
    //BIT3~BIT2:PORTC[7:4]源电流能力控制,BIT1~BIT0:PORTC[3:0]源电流能力控制

    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

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

```
 *   函数名称：DelayUs
 *   功能：      短延时函数  --16M-2T--大概快 1%左右.
 *   输入参数：Time 延时时间长度  延时时长 Time Us
 *   返回参数：无
 ----------------------------------------------------*/
void DelayUs(unsigned char Time)
{
    unsigned char a;
    for(a=0;a<Time;a++)
    {
        NOP();
    }
}
/*----------------------------------------------
 *   函数名称：DelayMs
 *   功能：      短延时函数
 *   输入参数：Time 延时时间长度  延时时长 Time ms
 *   返回参数：无
 ----------------------------------------------*/
void DelayMs(unsigned char Time)
{
    unsigned char a,b;
    for(a=0;a<Time;a++)
    {
        for(b=0;b<5;b++)
        {
            DelayUs(197); //快 1%
        }
    }
}
/*----------------------------------------------
 *   函数名称：DelayS
 *   功能：      短延时函数
 *   输入参数：Time 延时时间长度  延时时长 Time S
 *   返回参数：无
 ----------------------------------------------*/
void DelayS(unsigned char Time)
{
    unsigned char 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
 -----------------------------------------------*/
 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：
 *   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
    //外设时钟频率
    //6’b000000：不允许；
    //6’b000001：1MHz；
    //6’b000010：2MHz；
    //…
    //6’b011000：24MHz；
    //Higher values：不允许；

    I2CCCRL      =0B10000000;     //SCL 周期=2*CCR*Fmaster
    I2CCCRH      =0B00000000;     //
    I2CITR   =0B00000000;     //不使能 IIC 中断
    //I2CSR1 =0B00000000;     //

    //I2CSR2 =0B00000000;     //
    //I2CSR3 =0B00000000;
```

```
    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();
    }
}
```

Fremont Micro Devices (SZ) Limited

#5-8, 10/F, Changhong Building, Ke-Ji Nan 12 Road, Nanshan District, Shenzhen, Guangdong 518057
Tel: (86 755) 86117811
Fax: (86 755) 86117810

Fremont Micro Devices (Hong Kong) Limited

#16, 16/F, Blk B, Veristrong Industrial Centre, 34-36 Au Pui Wan Street, Fotan, Shatin, Hong Kong
Tel: (852) 27811186
Fax: (852) 27811144

Fremont Micro Devices (USA), Inc.

42982 Osgood Road Fremont, CA 94539
Tel: (1-510) 668-1321
Fax: (1-510) 226-9918

Web Site: http://www.fremontmicro.com/